SkyForm AIP

发布 *10.25.0*

天云融创软件

2025 年 10 月 17 日

目录

1 概览 **3**

1.1 简介 . . . 3
1.1.1 什么是 SkyForm 算力调度系统，简称 SkyForm AIP . . . 3
1.1.2 SkyForm AIP 的部署架构 . . . 4
1.1.3 SkyForm AIP 对资源占用 . . . 6
1.1.4 概念和术语 . . . 6
1.1.5 系统中的作业流 . . . 6
1.2 SkyForm 算力调度系统（AIP）快速安装指南 . . . 7
1.2.1 准备 . . . 7
1.2.2 安装 . . . 8
1.2.3 测试 . . . 9
1.2.4 Web 界面 . . . 10

2 管理 **11**

2.1 集群和节点环境准备 . . . 11
2.1.1 集群节点准备 . . . 12
2.1.2 创建 NFS 文件服务 . . . 13
2.1.3 安装 OpenLDAP 服务 . . . 14
2.1.4 在每个节点上配置 LDAP 客户端 . . . 16
2.1.5 集群 NAT 网关 . . . 17
2.2 安装 AIP . . . 18
2.2.1 安装前准备 . . . 18
2.2.2 准备文件系统 . . . 18
2.2.3 将 SkyForm AIP 安装到共享文件系统里 . . . 19
2.2.4 安装每台主机 . . . 19
2.2.5 测试 . . . 20
2.3 安装登录节点和 Web 门户 . . . 21
2.3.1 系统准备 . . . 21
2.3.2 如果 web 门户不是管理节点，安装 AIP 客户端 . . . 22
2.3.3 安装 Web 门户 . . . 22
2.3.4 测试 . . . 22
2.3.5 门户应用集成 . . . 23
2.3.6 集成 AIP 监控数据分析系统 . . . 23
2.3.7 排错 . . . 25
2.4 升级 AIP . . . 25
2.4.1 同版本更新 . . . 25

2.4.2 版本间升级 . . . 25
2.4.3 Web 门户代码版本升级 . . . 26
2.5 配置 . . . 26
2.5.1 调度器配置 cb.yaml . . . 26
2.5.2 计量计费配置 cbcrond.yaml . . . 30
2.5.3 典型的 EDA 负载集群配置 . . . 30
2.5.4 典型的 CAE（工程设计仿真）、HPC 集群的配置 . . . 31
2.5.5 典型的超算中心和智算中心集群配置 . . . 32
2.6 调度器动态配置参数 . . . 33
2.6.1 修改主机组 . . . 33
2.6.2 修改用户组 . . . 34
2.6.3 修改用户作业上限 . . . 37
2.6.4 修改队列 . . . 38
2.6.5 修改调度参数 . . . 39
2.6.6 修改部分 limit 参数 . . . 42
2.7 资源管理 . . . 43
2.7.1 什么是资源 . . . 43
2.7.2 资源定义 . . . 44
2.7.3 资源传感器 RESS . . . 44
2.7.4 资源配额 . . . 45
2.7.5 作业资源需求 . . . 46
2.7.6 多节异构作业资源需求 . . . 47
2.8 队列 . . . 47
2.8.1 队列配置的例子 . . . 47
2.8.2 调度主机顺序 . . . 52
2.8.3 大作业和独占作业作业槽预留 . . . 52
2.8.4 作业运行时长控制 . . . 52
2.8.5 作业内存使用控制 . . . 53
2.9 集群管理员命令 . . . 54
2.9.1 服务进程控制命令 . . . 54
2.9.2 资源使用量统计 . . . 54
2.9.3 动态修改配置（不重启服务或 daemon） . . . 55
2.10 安装共享数据分析和机器学习工具 . . . 56
2.10.1 准备 . . . 56
2.10.2 安装 Python . . . 56
2.10.3 安装机器学习库 . . . 56
2.10.4 安装 Jupyter . . . 56
2.10.5 安装 R . . . 57
2.10.6 安装 Julia . . . 57
2.10.7 安装 Spark . . . 57
2.10.8 安装 Jupyter 内核 . . . 57
2.10.9 安装 RStudio . . . 59
2.10.10 安装 Singularity . . . 59
2.10.11 安装 Git . . . 59
2.11 基于 License 的调度 . . . 60
2.12 监控仪表盘 . . . 62
2.12.1 安装 Prometheus . . . 62
2.12.2 安装 Grafana . . . 62
2.12.3 安装 AIP 监控包 . . . 62
2.12.4 浏览器连接 . . . 62
2.12.5 修改已有的仪表盘 . . . 63
2.12.6 aip-exporter 提供的数据 . . . 64
2.13 配置 Web 门户 . . . 64
2.13.1 门户配置基本参数 . . . 64

2.13.2 管理员门户管理配置 . . . 64
2.14 Web 门户应用集成 . . . 67
2.14.1 YAML 模板文件制作步骤 . . . 67
2.15 TCP/IP 代理服务 aipproxy . . . 74
2.15.1 概述 . . . 74
2.15.2 安装和配置 . . . 74
2.15.3 作业提交 . . . 75
2.16 节电调度插件 . . . 75
2.16.1 启动节电策略 . . . 76
2.16.2 节电策略插件案例 . . . 76
2.16.3 节电统计报表 . . . 78
2.17 大机群或者高通量负载集群操作系统调参 . . . 80
2.17.1 调度器主机 . . . 80
2.17.2 登录主机 . . . 81
2.18 故障处理 . . . 81
2.18.1 系统日志 . . . 81
2.18.2 常见问题 . . . 81

3 使用 85
3.1 作业定义和管理 . . . 85
3.1.1 系统命令 . . . 85
3.1.2 运行作业 . . . 86
3.1.3 高性能计算作业命令 . . . 87
3.1.4 查看作业 . . . 88
3.1.5 作业控制 . . . 91
3.1.6 查看系统资源 . . . 91
3.1.7 LSF 兼容命令 . . . 96
3.1.8 SLURM 兼容命令 . . . 96
3.1.9 PBS 兼容命令 . . . 97
3.1.10 作业提交资源需求 . . . 97
3.2 机器学习和数学分析作业 . . . 97
3.2.1 分布式 TensorFlow 的作业定义 . . . 97
3.2.2 分布式 MXNet 的作业定义 . . . 103
3.2.3 分布式 PyTorch 的作业定义 . . . 104
3.2.4 Intel Caffe MPI 作业 . . . 105
3.2.5 分布式 Ray 作业定义 . . . 105
3.2.6 Spark 集群作业 . . . 107
3.2.7 Megatron-deepspeed . . . 109
3.3 GPU 作业 . . . 111
3.3.1 GPU 资源自动检测 . . . 111
3.3.2 提交单机 GPU 作业 . . . 112
3.3.3 多节点 GPU 作业 . . . 112
3.3.4 GPU 资源使用限制 . . . 112
3.3.5 VGPU：多任务共享 GPU . . . 113
3.4 NVIDIA 多实例 GPU 调度 . . . 115
3.4.1 提交作业 . . . 115
3.5 MPI 作业 . . . 116
3.5.1 Intel MPI . . . 118
3.5.2 MPICH 和 MVAPICH . . . 118
3.5.3 OpenMPI 和 Platform/HP/IBM MPI . . . 119
3.5.4 MPI 和 OpenMP 混合作业 . . . 119
3.5.5 使用 SLURM 兼容命令提交 MPI 作业 . . . 120
3.6 Docker 容器作业 . . . 120
3.6.1 提交 Docker 容器作业 . . . 120

3.6.2 在运行中的 Docker 容器作业中执行命令 . . . 121
3.6.3 私有容器镜像库登录 . . . 121
3.6.4 从第三方的镜像库中引入镜像上传到本地镜像库 . . . 122
3.6.5 容器作业排错 . . . 122
3.7 Singularity/Apptainer 容器作业 . . . 122
3.7.1 制作镜像文件 . . . 122
3.7.2 提交 Singularity 简单作业 . . . 123
3.7.3 提交 Singularity MPI 作业 . . . 123
3.8 使用 Jupyter Lab . . . 123
3.8.1 启动 Jupyter Lab . . . 124
3.8.2 在 Singularity 容器里启动 Jupyter Lab . . . 124
3.8.3 切换成中文 . . . 124
3.8.4 安装 Python 库 . . . 124
3.8.5 安装 Julia 内核 . . . 125
3.8.6 安装其他 Anaconda 部署的 Python 内核 . . . 125
3.9 使用 VSCodes . . . 126
3.9.1 在本地机器 (PC 或工作站) 上安装微软 Visual Studio Code . . . 126
3.9.2 安装扩展 . . . 126
3.9.3 在 AIP 环境中启动可以 ssh 的容器作业 . . . 126
3.9.4 配置远程 ssh . . . 127
3.9.5 在 VSCode 中连接 . . . 128
3.9.6 打开远程文件 . . . 128
3.9.7 使用 code-server . . . 128
3.10 使用 RStudio . . . 129
3.10.1 启动 RStudio . . . 129
3.10.2 在 Singularity 容器里启动 RStudio . . . 129
3.10.3 退出或结束运行 . . . 130
3.10.4 安装 R 库 . . . 130
3.10.5 排错 . . . 130
3.11 使用 Nextflow . . . 130
3.11.1 在用户环境中安装 Nextflow . . . 130
3.11.2 Nextflow 利用 AIP 跑任务的例子 . . . 131
3.12 Cron 作业 . . . 133
3.12.1 cron 作业流程 . . . 133
3.12.2 cron 作业定义和策略 . . . 134
3.12.3 查看 cron 作业 . . . 136
3.12.4 批量 cron 作业 . . . 138

4 组件 139
4.1 VNC 和 SSH 访问门户 . . . 139
4.1.1 安装 . . . 139
4.1.2 用户 VNC 客户端 . . . 140
4.1.3 登录 . . . 140
4.1.4 启动 VNC 桌面 . . . 140
4.1.5 启动 ssh 访问 . . . 143
4.1.6 VNC 桌面和 ssh 会话的资源使用控制 . . . 144
4.2 SkyForm AIP 的 REST API 服务 . . . 144
4.2.1 系统简介 . . . 144
4.2.2 安装 aiprestd . . . 145
4.2.3 REST 接口 . . . 146
4.2.4 高可用方案 . . . 149
4.3 AIP 监控数据分析系统 . . . 150
4.3.1 系统简介 . . . 150
4.3.2 安装 . . . 150

4.3.3 与 AIP 门户的集成 . . . 157
4.3.4 维护和定制 . . . 157
4.3.5 故障排查 . . . 158
4.3.6 数据说明 . . . 159
4.4 AIP DNS . . . 165

5 参考 **167**
5.1 命令 . . . 167
5.1.1 bjobs . . . 167
5.1.2 cacct . . . 168
5.1.3 caddhost . . . 171
5.1.4 cadmin . . . 172
5.1.5 capps . . . 178
5.1.6 cbot . . . 179
5.1.7 cchkpnt . . . 180
5.1.8 ccluster . . . 182
5.1.9 cdexe . . . 183
5.1.10 cdimport . . . 184
5.1.11 cdlogin . . . 185
5.1.12 cexpand . . . 186
5.1.13 cgadd . . . 187
5.1.14 cgdel . . . 188
5.1.15 cgmod . . . 188
5.1.16 chinfo . . . 190
5.1.17 chist . . . 194
5.1.18 chosts . . . 199
5.1.19 cinfo . . . 203
5.1.20 cjdep . . . 205
5.1.21 cjgroup . . . 206
5.1.22 cjobs . . . 207
5.1.23 ckill . . . 224
5.1.24 cload . . . 227
5.1.25 cmgroup . . . 230
5.1.26 cmod . . . 232
5.1.27 cparams . . . 234
5.1.28 cpasswd . . . 235
5.1.29 cping . . . 236
5.1.30 cplace . . . 237
5.1.31 cpost . . . 238
5.1.32 cqueues . . . 239
5.1.33 crcp . . . 247
5.1.34 cread . . . 249
5.1.35 creduce . . . 250
5.1.36 creport . . . 251
5.1.37 crequeue . . . 256
5.1.38 cresources . . . 258
5.1.39 crestart . . . 260
5.1.40 cresume . . . 261
5.1.41 crmhost . . . 262
5.1.42 crun . . . 263
5.1.43 csadmin . . . 264
5.1.44 cstop . . . 271
5.1.45 csub . . . 273
5.1.46 cswitch . . . 297

5.1.47 ctask . . . 298
5.1.48 ctop . . . 299
5.1.49 cugroup . . . 300
5.1.50 cusers . . . 302
5.1.51 cview . . . 303
5.1.52 runtask . . . 304
5.1.53 vncsub . . . 306
5.2 配置文件 . . . 309
5.2.1 aiprestd.yaml . . . 309
5.2.2 cb.acct . . . 310
5.2.3 cb.yaml . . . 314
5.2.4 cbcrond.yaml . . . 350
5.2.5 gpu.yaml . . . 353
5.2.6 jservice.yaml . . . 355
5.2.7 olmon.conf . . . 355
5.3 服务进程 . . . 358
5.3.1 aip-exporter . . . 358
5.3.2 cbls . . . 362
5.3.3 cbls.remote . . . 364
5.3.4 cbexe . . . 366
5.3.5 cbjm . . . 367
5.3.6 cbps . . . 367
5.3.7 cbsched . . . 368
5.3.8 cbtt . . . 370
5.3.9 echkpnt . . . 371
5.3.10 erestart . . . 373
5.3.11 esub . . . 375
5.3.12 host-setup . . . 379
5.3.13 install . . . 380
5.3.14 jservice . . . 382
5.3.15 ress . . . 384

6 软件许可 387
6.1 天云融创软件许可 . . . 387
6.2 第三方软件许可 . . . 388

7 产品常见问题 389
7.1 什么是算力调度系统，它为什么重要 . . . 389
7.2 SkyForm 算力调度系统有哪些特色，提供哪些价值 . . . 389
7.3 SkyForm 算力调度系统是怎样收费的 . . . 390
7.4 对于 AI 模型训练，调度系统提供哪些功能 . . . 390
7.5 对于推理服务，调度系统提供哪些功能 . . . 390
7.6 对于 AI 应用开发，调度系统提供哪些功能 . . . 390
7.7 系统支持哪些类型的 CPU . . . 390
7.8 系统支持哪些类型的 GPU 或加速芯片 . . . 390
7.9 SkyForm 算力调度系统可以在不用虚拟机的场景中支持 vGPU 吗 . . . 391
7.10 系统怎样实现任务间的资源隔离 . . . 391
7.11 系统支持哪些容器类型 . . . 391
7.12 系统支持哪些操作系统 . . . 391
7.13 系统支持哪些 MPI 框架 . . . 391
7.14 系统支持哪些 AI 框架和大语言模型 . . . 391
7.15 如果算力节点出现故障，任务怎样恢复 . . . 391
7.16 调度系统有节能的功能吗 . . . 392
7.17 调度系统支持哪些接口 . . . 392

8 搜索 393

文档 PDF 下载

CHAPTER 1

概览

1.1 简介

1.1.1 什么是 SkyForm 算力调度系统，简称 SkyForm AIP

SkyForm 算力调度系统（SkyForm AIP）是天云融创软件自研的信创基础软件产品，是针对 AI、高性能计算和高性能数据分析的算力和任务调度系统。SkyForm 算力调度系统为企业 HPC（如 EDA、CAE、生物信息等）和 AI 应用开发者和高性能计算用户提供针对异构算力资源提供多维度智能调度策略，满足用户最复杂的和最优的算力调度场景，适用于 AI 模型的开发、训练和推理及利用大模型开发 AI 应用，高性能计算应用和高性能数据分析的算力分配和调度，使算力能力能发挥到极致，以加速企业产品研发和提高企业服务质量，缩短项目时间，节约算力成本。

SkyForm 算力调度系统提供多异构算力池的资源自动发现和监控、高速任务调度和大规模任务的分发，兼容业界传统超算调度器的命令行以快速集成各类应用。通过网络通讯转发机制实现安全的应用远程访问。内置 3D 可视化加速组件实现 3D 应用的远程可视化。这些多形态的应用支撑和快速集成增加了应用使用算力的效率。调度系统支持多层用户组织结构，自动同步企业内部的 AD/LDAP 用户管理的组织成员。通过配置针对组织结构的调度策略实现根据业务算力优先级的智能调度。调度器支持多种调度策略的组合：先进先出、优先级抢占、大任务资源自动预留、基于真实负载的任务调度、GPU 智能调度（专利技术）等大大增加调度的智能化，使算力效能最大化，运维成本最小化。SkyForm 算力调度系统提供可持续的每小时超过 1 百万个任务的高通量调度能力，这种调度能力保障生物信息、芯片设计和制造、金融数据分析领域的应用所需的算力使用效率。SkyForm 算力调度系统支持超过 50 万子任务的分布式并行任务的快速资源调度、子任务分发、监控和清理，超大任务分发和启动可在数分钟内完成，以保障大规模超算任务和 AI 大模型训练任务的有效可靠运行。不同种类的 CPU（如 X86 和 ARM）、不同种类的加速芯片（国产 GPU、GCU 等以及进口 GPU）和在同一算力池中进行算力调度，这种融合异构资源的融合算力池可使不同种类的算力应用，如高性能计算、高性能数据分析、交互式设计和开发、AI 训练、AI 推理等共享算力资源，提高吞吐、降低建设成本。为满足不同类型客户的需求，SkyForm 算力调度系统还内置了计量和计费功能，可以根据定价自动生成用户级月账单。

1.1.2 SkyForm AIP 的部署架构

常规的 HPC 集群含有至少一个登录节点、一个管理节点、和多个计算节点。计算节点的配置也各不相同，有些有大内存的“胖节点”，许多集群，尤其是机器学习的集群，含有 GPU 的 GPU 节点。

所有节点都接在共享文件系统上。共享系统可以使用如 IBM GPFS，开源 Lustre，或 Ceph FS，或者 NAS 一体机（如 NetApp）等。

用户认证服务（如 LDAP 服务）安装在管理节点上，所有其他节点都是用该服务进行用户认证（如 LDAP 客户端）。

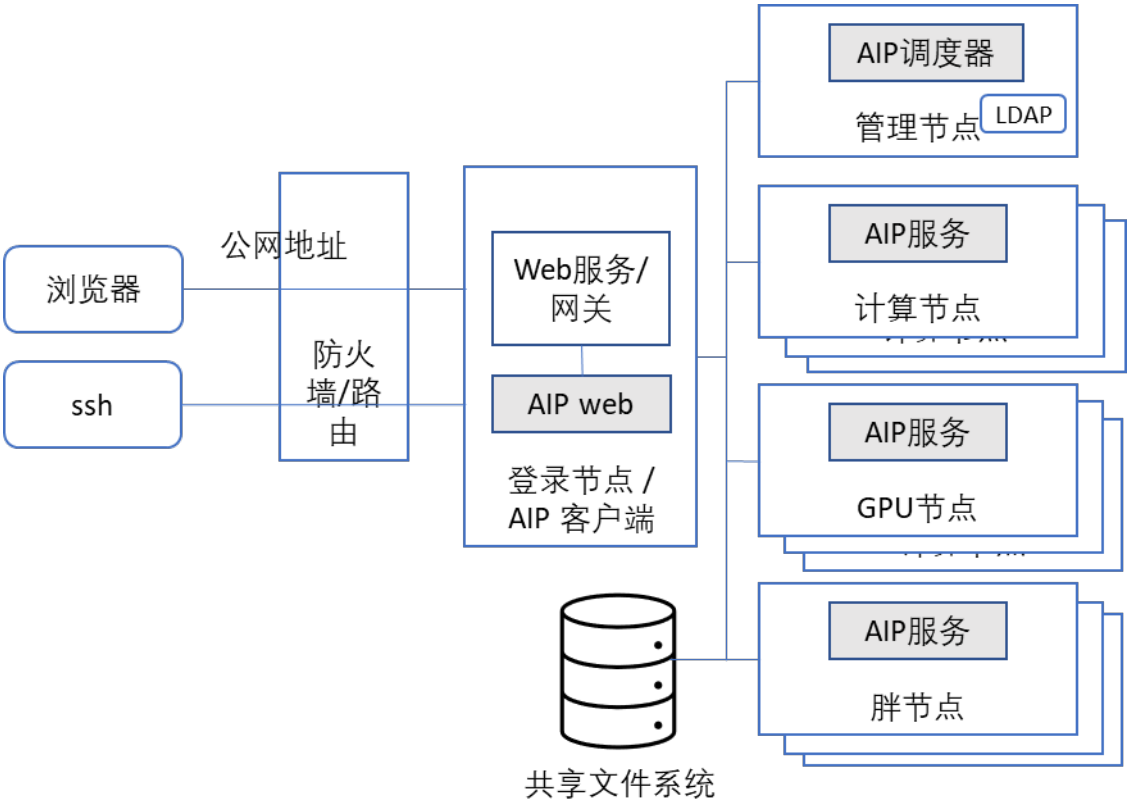


图 1: SkyForm AIP 部署架构

小的集群管理节点和登录节点可以合而为一，或者管理节点和计算节点可以合而为一。

AIP 的部署为：（1）管理节点和计算节点（2）登录节点。

备注： “节点”在本文中也叫做“主机”。这两个词会交替使用。

AIP 包中含有多个额外的组件。下图显示了 AIP 的其他几个组件，包括：分析数据采集服务和分析数据库、监控数据采集和可视化、REST API 服务、以及 VNC 门户等。

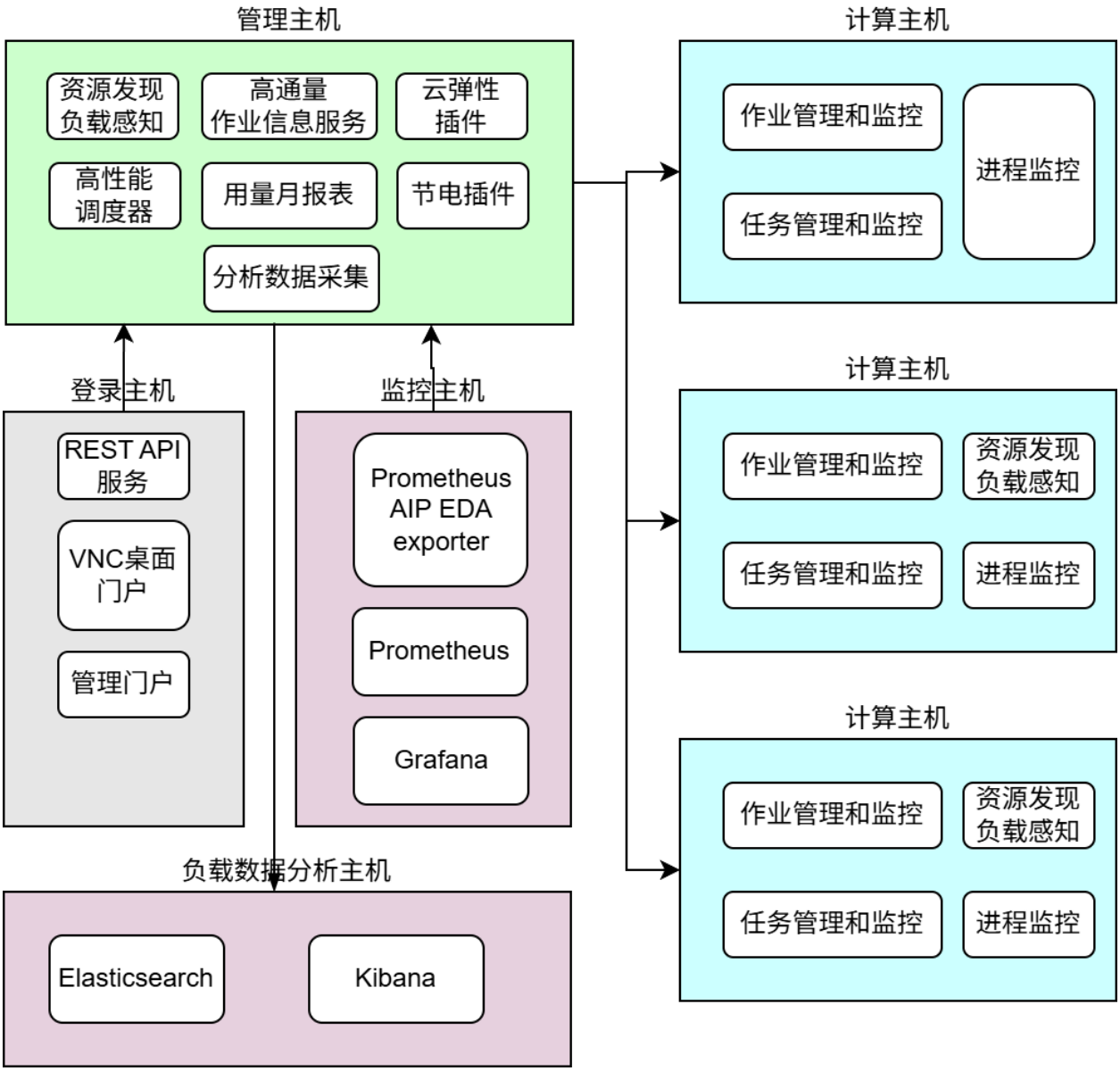


图 2: SkyForm AIP 组件

1.1.3 SkyForm AIP 对资源占用

与一般调度系统类似，SkyForm AIP 占用资源非常少。对于 50 节点以下的集群，在一般的节点上的内存，CPU 等没有特殊要求。SkyForm AIP 不用数据库，不用 Java，所以非常简洁。

1.1.4 概念和术语

在您开始使用 SkyForm 算力调度系统前，请先了解下表中所描述的概念和术语。

表 1-1 SkyForm 任务调度系统概念和术语

概念/术语	描述
集群（Cluster）	集群是一组主机的集合，这些主机上运行 SkyForm AIP 且通过 TCP/IP 网络互相连通。
Master 控制主机（主节点）	每个集群都需要一个 master 主机，master 主机控制着 SkyForm AIP 集群中的其他主机。
计算主机	集群中运行应用作业及任务的主机。
Remote Server 主机	主机上不安装 SkyForm AIP 服务，主机负载通过 Master 上的 cbls.remote 程序获取。分发到 remote server 上的作业先分发到 Master，然后由 Master 通过定制脚本转送到远程主机上。
Dynamic Server 主机	主机不在集群中预先配置，而是使用 caddhost 命令动态加入。
Client（客户端）主机	集群中仅提交作业及任务的主机。
作业执行主机	运行作业和任务的主机。
作业提交主机	作业在其上被提交的主机。
作业（Job）	提交至 SkyForm AIP 的应用。可占用一至多个 Job slot。
作业槽（Job slot）	SkyForm AIP 中处理器分配的最小单元。可以是一至多个物理处理器或物理处理器的一部分。默认情况下，一个作业槽就是一个处理器核。
队列（Queue）	网络范围内的一个存放作业的地方，负责对不同作业实施调度及管理调度策略。
集群主管理员	集群主要的管理员用户，具有修改集群配置文件、控制集群主机、和控制所有用户作业的权限。

1.1.5 系统中的作业流

SkyForm AIP 中的作业流如下图所示。

作业运行流程：

- 用户使用 aip job 命令或 csub 提交作业
- 作业被提交至队列中
- 为每个作业分配一个唯一的作业 ID
- 为每个作业的每个子任务分配一个通讯端口
- 调度器根据作业资源需求和系统中的可用资源为队列里的作业找合适一个或多个主机，将作业定义分发到找到的一个主机上，启动作业控制器。
- 作业控制器根据作业任务定义和调度器所分配的资源启动和管理作业任务。

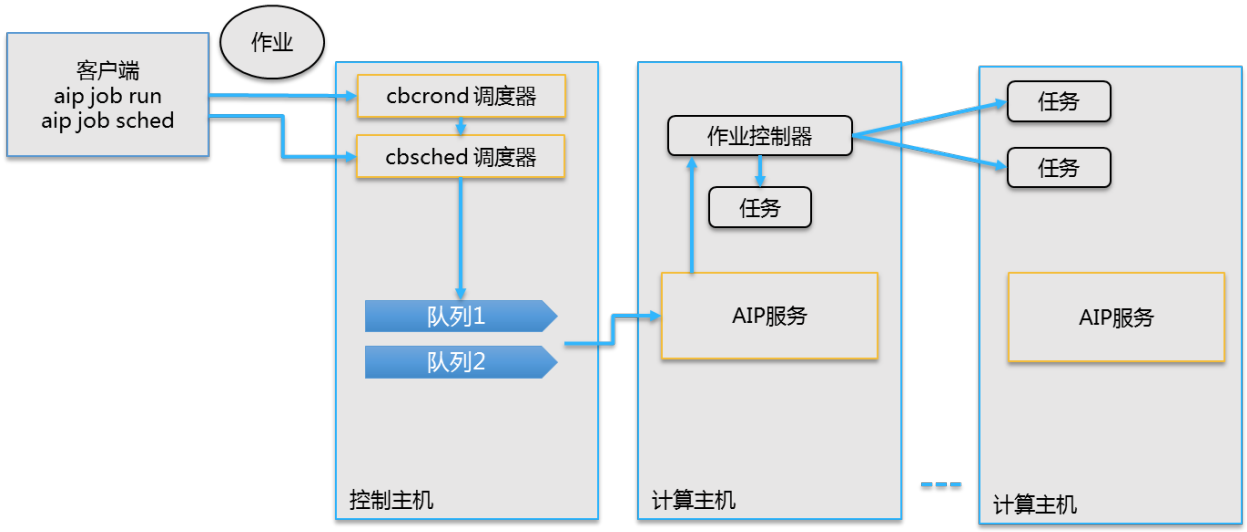


图 3: 作业流程图

1.2 SkyForm 算力调度系统（AIP）快速安装指南

1.2.1 准备

SkyForm 算力调度系统安装需要：

1. **所有内网（节点间）关闭防火墙、关闭 SELINUX**

- 在 EL（CentOS、Rocky Linux，Red Hat Enterprise Linux 等）上：

```
systemctl disable --now firewalld
setenforce 0
sed -i s/SELINUX=enforcing/SELINUX=disabled/ /etc/selinux/config
```

- Ubuntu 上：

```
ufw disable
setenforce 0
sed -i s/SELINUX=enforcing/SELINUX=disabled/ /etc/selinux/config
```

2. **安装所需的软件包**

- 在 EL7(如 CentOS 7) 中，安装 libyaml, psmisc, net-tools

```
yum -y install libyaml psmisc net-tools
```

- 在 EL8(如 Rocky Linux 8) 或以上版本中，安装 net-tools, libnsl

```
dnf install -y net-tools libnsl
```

- 在 Ubuntu 上：

```
apt install dmidecode numactl psmisc
```

3. **hosts 文件**，包含 AIP 集群所有节点的 IP 地址和节点名，格式与系统/etc/hosts 一样，例子：

```
192.168.20.100 exp001
192.168.20.101 exp002
192.168.20.102 exp003
```

> **警告：** hosts 文件中主管理节点必须放在头部。每个节点上的主机名 (hostname 命令的输出) 必须与以上 hosts 文件里的主机名一致。如果 hosts 文件没有提供，缺省安装基于本机的单节点集群。

4. **共享文件系统**，用于安装 AIP 文件和容错，缺省：/opt/skyformai_shared。如果只有一台主机，可以只用本地文件系统。参考集群和节点环境准备。
5. **统一的用户认证系统**：LDAP、AD、NIS、或各节点相同用户名和 ID 的本地用户。指定一个集群主管理员，缺省：cadmin。

> **小技巧：** 集群主管理员也可设成 root，安装时脚本会提示。如果安装时主管理员不存在，安装脚本会自动创建本地主管理员用户，缺省为 cadmin。

6. 指定一个联外网的节点作为 Web 服务器 (可选) Web 服务器上安装 php, python3, python36-numpy, php-pecl-yaml (先需要安装 epel-release)

```
yum install -y epel-release
yum install php php-pecl-yaml nginx php-fpm
systemctl enable nginx php-fpm
```

1.2.2 安装

安装分三步：

1. 把文件拷贝到共享文件系统中
2. 在每个节点上设置 AIP 服务和运行环境
3. 安装 web 界面

软件包下载

请到 SkyForm AIP 官方网站 https://skyformaip.com 下载软件包。

拷贝文件

1. 解压下载的文件：

```
mkdir aip
tar xfz skyformaip-10.xx.x.tar.gz -C aip
cp hosts aip  #可选
```

2. 运行安装脚本

```
# cd aip
./install
```

若不用缺省的路径、cadmin 作为主管理员，参考./install –help 显示的命令行参数或者在提示时输入。

每个节点的设置

在每个节点上运行 host-setup。这个脚本安装在 AIP 共享目录中，缺省/opt/skyformai_shared。

```
/opt/skyformai_shared/host-setup
```

备注： SkyForm AIP 软件包缺省为企业版，第一次安装自带 45 天的企业版功能。45 天后自动降级为免费版。若发现 key 有问题，可以修改 /opt/skyformai/etc/cb.yaml 里的参数 enterprise, 注释掉或者删除该参数，然后重启主管理节点上的 AIP 服务。

安装 web 界面

运行 AIP 的界面安装脚本

```
./portal-install
```

1.2.3 测试

登录成普通用户

1) 检查集群状态：

```
aip cluster info
```

或简写

```
aip c i
```

或 SLURM 命令：

```
sinfo
```

2) 提交作业

SLURM 命令方式

```
$ sbatch
#!/bin/bash
sleep 1h
^D
Submitted batch job 4342

$ squeue
JOBID PARTITION NAME      USER   ST TIME NODES NODELIST(REASON)
4342  medium    sleep 100 cadmin R  0:04     1 linux7

$ scancel 4342
```

LSF 命令方式

```
$ bsub -I hostname
Job <838> is submitted to default queue <medium>.
Job 838 is waiting to be started...
```

(下页继续)

(续上页)

```
Job 838 has started on host linux7.
linux7

$ bsub sleep 1h
Job <839> is submitted to default queue <medium>.

$ bjobs 839
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
839     cadmin  RUN   medium     dev         dev         sleep 1h   May  2 15:06

$ bjobs -l 839

Job <839>, User <cadmin>, Project <default>, User Group <comb>, Status <RUN>, Q
                     ueue <medium>, Job Priority <50>, Command <sleep 1h>
Fri May  2 15:06:52: Submitted from host <dev>, CWD <$HOME>;
Fri May  2 15:06:52: Started on <dev>, Execution Home </home/cadmin>, Execution
                     CWD </home/cadmin>;
Fri May  2 15:07:04: Total resource usage collected.
                     MEM: 356 Kbytes;  SWAP: 105 Mbytes;  NTHREAD: 1
                     PGID: 27491;  PIDs: 27491 27505 27506


 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   up    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

              gpu
 loadSched     -
 loadStop      -

 RESOURCE REQUIREMENT DETAILS:
 Combined: 1{select[type=="local"]order[slots]}
 Effective: 1{select[type=="local"]order[slots]}
```

1.2.4 Web 界面

AIP 的 web 登录使用 Linux 里的用户名和密码。浏览器连接到: https://< 服务器 >

CHAPTER 2

管理

2.1 集群和节点环境准备

算力集群的常规架构如下图所示。

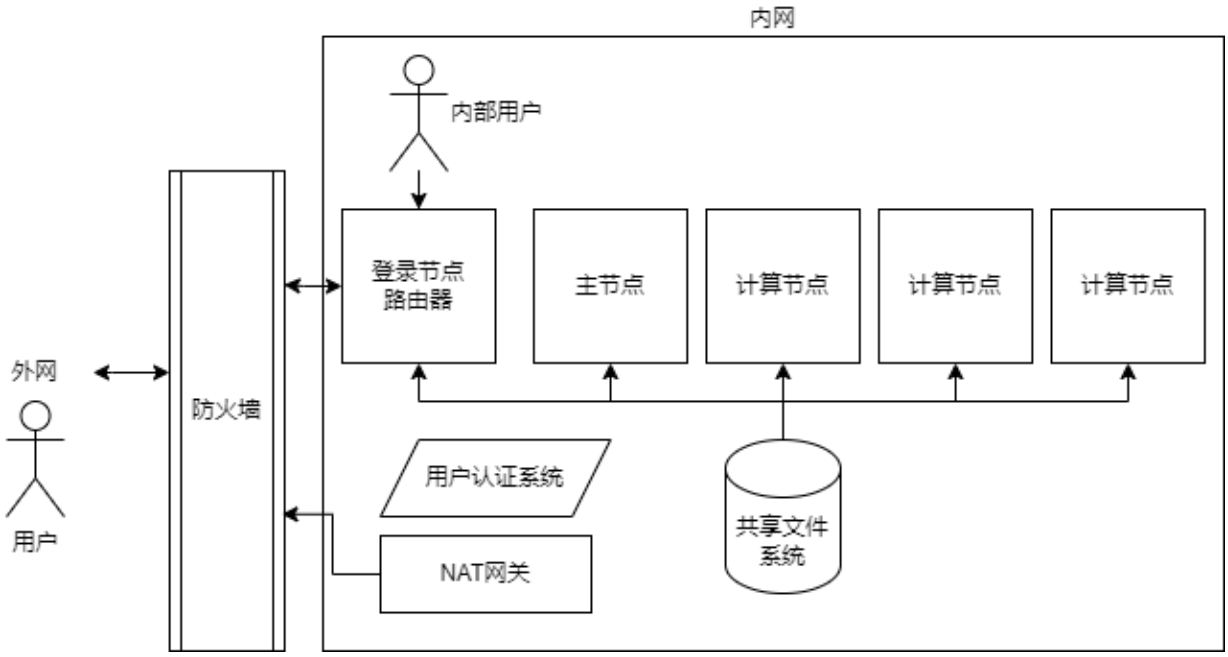


图 1: 集群架构

集群的所有节点都需挂载共享文件系统，以便（1）用户 HOME 目录以及所有 HOME 目录下的文件在所有节点上一致。（2）系统及应用软件，如多个 CUDA 版本，多个 Python 版本可以一次性安装后，在所有节点上都可以访问。（3）利用 POSIX 的用户 ID（uid、gid）控制文件访问权限，如每个用户的 HOME 只有用户自己可以访问。

集群的所有节点都用集中统一的用户认证系统，如 Active Directory、LDAP、FreeIPA、NIS 等，以保证用户在所有节点上访问以及权限的一致性。

整个集群被认为是同一个系统，其中所有节点间都是互信的，关闭防火墙，禁用 SELINUX。而集群与外部则需要防火墙，只开少量的端口，集群外部用户访问都必须通过前端的登录节点，通讯由登录节点转发。对外的 we 门户就放在登录节点上。由于登录节点上一般为无状态，任务的状态都由 AIP 管理，部署多个登录节点可以达到高可用的功能。

集群内部的节点访问互联网（下载软件等）需要通过 NAT 网关。

集群内部的访问则可在任意一个节点上完成。

备注： 本文档中 EL 表示基于 Fedora 的 Enterprise Linux，如 CentOS、Red Hat Enterprise Linux, Rocky Linux, Oracle Linux 等。

2.1.1 集群节点准备

1. 安装 AIP 依赖的软件包

- 在 EL7(如 CentOS 7) 中，安装 libyaml, libcurl, psmisc, net-tools

```
yum -y install libcurl libyaml psmisc net-tools
```

- 在 EL8(如 Rocky Linux 8) 或以上版本中，安装 net-tools, libnsl

```
dnf install -y libcurl libyaml psmisc net-tools libnsl
```

- 在 Ubuntu 上：

```
apt install dmidecode numactl psmisc
```

2. 关闭防火墙、关闭 SELINUX

在 EL（CentOS、Rocky Linux，Red Hat Enterprise Linux 等）上：

```
systemctl disable --now firewalld
setenforce 0
sed -i s/SELINUX=enforcing/SELINUX=disabled/ /etc/selinux/config
```

Ubuntu 上：

```
ufw disable
setenforce 0
sed -i s/SELINUX=enforcing/SELINUX=disabled/ /etc/selinux/config
```

2.1.2 创建 NFS 文件服务

对于测试集群，可以用第一个节点的本地盘作为 NFS 共享存储系统（或者选择集群中任意一台本地盘较大的节点创建 NFS 服务）。

小技巧： 把普通 Linux 服务器作为 NFS 共享存储服务器，考虑其性能和可靠性，不建议在生产环境中使用。生产环境中应该使用专用 NAS 或高性能并行文件系统，如 GPFS、Lustre 等。

在第一个节点上创建 NFS 文件服务

假设本地子网为 10.10.0.0/24 在 EL（CentOS、Rocky Linux、Red Hat 等）上：

```
yum -y install nfs-utils
```

在 Ubuntu 上：

```
apt install nfs-kernel-server
```

配置：

```
mkdir -p /share
cat <<EOF> > /etc/exports
/home 10.10.0.0/24(rw,no_root_squash)
/share 10.10.0.0/24(rw,no_root_squash)
EOF
systemctl enable --now rpcbind nfs-server
```

在集群其他几点上挂载 NFS

假设共享 NFS 的服务器地址为 10.10.0.4 在 EL（CentOS、Rocky Linux、Red Hat 等）上：

```
yum -y install nfs-utils
```

在 Ubuntu 上：

```
apt install nfs-common
```

继续配置：

```
mkdir -p /share
cat <<EOF >> /etc/fstab
10.10.0.4:/home /home nfs defaults 0 0
10.10.0.4:/share /share nfs defaults 0 0
EOF
systemctl daemon-reload
mount /home
mount /share
```

2.1.3 安装 OpenLDAP 服务

小集群可以在主节点上安装，大机群需要一个专用的管理节点安装。### 在 EL 上安装以下脚本安装提供 TLS 加密的 OpenLDAP 服务。执行前，请务必修改头部的部分参数，尤其是密码、域名等。

```
#!/bin/bash
ldaprootpw=Passw0rd               # LDAP Root 密码，请修改
managername=Manager               # LDAP管理员用户名，如必要，也请修改
managerpassword='Passw0rd123!'    # LDAP管理员密码，请修改
domain="dc=skyformaip,dc=com"     # 域名，请修改
oname=SkyFormAIP                  # 顶层组织名，请修改

yum -y install openldap-servers openldap-clients

workdir=/root
cd $workdir
cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
chown ldap. /var/lib/ldap/DB_CONFIG
systemctl enable --now slapd

rootpw="$(slappasswd -s $ldaprootpw)"

cat <<EOF > chrootpw.ldif
dn: olcDatabase={0}config,cn=config
changetype: modify
add: olcRootPW
olcRootPW: $rootpw
EOF

ldapadd -Y EXTERNAL -H ldapi:/// -f chrootpw.ldif
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/cosine.ldif
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/nis.ldif
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/inetorgperson.ldif

mngerpw="$(slappasswd -s $managerpassword)"

cat <<EOF > chdomain.ldif
dn: olcDatabase={1}monitor,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
↪"
  read by dn.base="cn=$managername,$domain" read by * none

dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: $domain

dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: cn=$managername,$domain

dn: olcDatabase={2}hdb,cn=config
changetype: modify
add: olcRootPW
olcRootPW: $mngerpw
```

(下页继续)

(续上页)

```
dn: olcDatabase={2}hdb,cn=config
changetype: modify
add: olcAccess
olcAccess: {0}to attrs=userPassword,shadowLastChange by
  dn="cn=$managername,$domain" write by anonymous auth by self write by * none
olcAccess: {1}to dn.base="" by * read
olcAccess: {2}to * by dn="cn=$managername,$domain" write by * read
EOF

ldapmodify -Y EXTERNAL -H ldapi:/// -f chdomain.ldif

dc=$(echo $domain | cut -d, -f1 | cut -d= -f2)

cat <<EOF > basedomain.ldif
dn: $domain
objectClass: top
objectClass: dcObject
objectclass: organization
o: $oname
dc: $dc

dn: cn=$managername,$domain
objectClass: organizationalRole
cn: $managername
description: Directory Manager

dn: ou=People,$domain
objectClass: organizationalUnit
ou: People

dn: ou=Group,$domain
objectClass: organizationalUnit
ou: Group
EOF

ldapadd -x -D cn=$managername,$domain -w $managerpassword -f basedomain.ldif

PASSPHRASE=$(echo $ldaprootpw | head -c 6)
cd /etc/pki/tls/certs
umask 77
openssl genrsa -aes128 -passout pass:$PASSPHRASE 2048 > server.key
openssl rsa -passin pass:$PASSPHRASE -in server.key -out server.key
openssl req -utf8 -new -key server.key -out server.csr \
  -subj "/C=CN/ST=Beijing/L=Beijing/O=$oname/OU=IT/CN=$(hostname)" \
  -passin pass:$PASSPHRASE
openssl x509 -in server.csr -out server.crt -req -signkey server.key -days 3650
cp /etc/pki/tls/certs/server.key \
   /etc/pki/tls/certs/server.crt \
   /etc/pki/tls/certs/ca-bundle.crt \
   /etc/openldap/certs/

chown ldap: /etc/openldap/certs/server.key \
    /etc/openldap/certs/server.crt \
    /etc/openldap/certs/ca-bundle.crt
cd $workdir
```

(下页继续)

(续上页)

```
cat <<EOF > mod_ssl.ldif
dn: cn=config
changetype: modify
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/openldap/certs/ca-bundle.crt
-
replace: olcTLSCertificateFile
olcTLSCertificateFile: /etc/openldap/certs/server.crt
-
replace: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/openldap/certs/server.key
EOF
ldapmodify -Y EXTERNAL -H ldapi:/// -f mod_ssl.ldif
sed -i 9d /etc/sysconfig/slapd
sed -i '9i SLAPD_URLS="ldapi:/// ldap:/// ldaps:///"' /etc/sysconfig/slapd
systemctl restart slapd
```

在 Ubuntu 上安装

请参考线上文档：https://www.server-world.info/en/note?os=Ubuntu_22.04&p=openldap&f=1

2.1.4 在每个节点上配置 LDAP 客户端

在 EL 上：

```
yum -y install sssd sssd-ldap oddjob-mkhomedir
```

在 Ubuntu 上：

```
apt install sssd sssd-ldap oddjob-mkhomedir
```

配置：

```
ldapserver_ip=10.10.1.3     #修改LDAP服务器的IP地址
domain="dc=skyformaip,dc=com" #修改域名，与LDAP服务设置的域名一致

cat <<EOF > /etc/sssd/sssd.conf
[domain/default]
id_provider = ldap
autofs_provider = ldap
auth_provider = ldap
chpass_provider = ldap
ldap_uri = ldaps://$ldapserver_ip:636
ldap_search_base = $domain
ldap_id_use_start_tls = False
cache_credentials = True
[sssd]
services = nss, pam, autofs
domains = default
[nss]
homedir_substring = /home
EOF
chmod 600 /etc/sssd/sssd.conf
```

(下页继续)

(续上页)

```
# for EL 7
authconfig --enablesssd --enablesssdauth --update
# for other OS
authselect select sssd with-mkhomedir --force
systemctl enable sssd oddjobd
systemctl restart sssd oddjobd
```

2.1.5 集群 NAT 网关

若硬件路由器不支持 NAT 网关，而内部节点要需要能够访问外网（外网不允许直接访问内网节点），可以在一台 Linux 服务器上设置 NAT 网关。如图二所示。

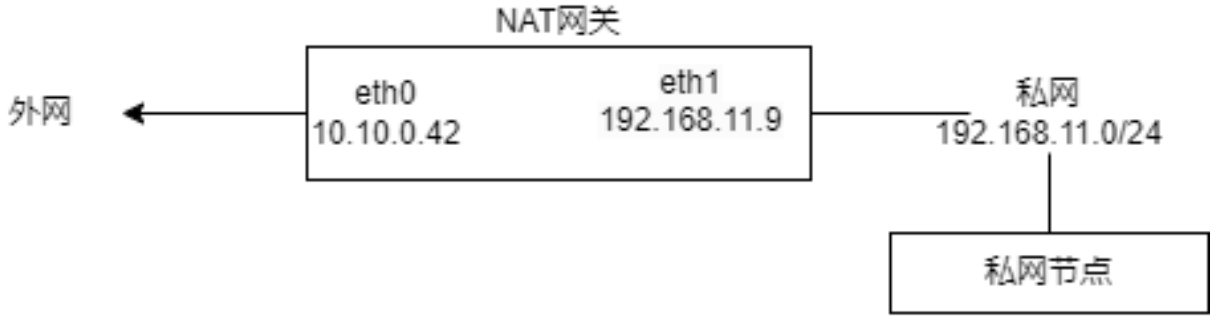


图 2: NAT 网关

1. 私网节点设置

把网络的 gateway 设成网关节点的内网网卡地址

```
systemctl disable --now firewalld
route add default gw 192.168.11.9   # 网关内网地址
```

2. NAT 网关节点设置

```
public_ip=10.10.0.42              # 网关外网地址
public_nic=eth0                   # 网关外网网卡设备名
private_subnet="192.168.11.0/24"  # 内网子网
private_nic=eth1                  # 网关内网网卡设备名
systemctl disable --now firewalld
cat <<EOF > /etc/init.d/nat
#!/bin/bash
# chkconfig: 345 99 10
case "$1" in
  start)
    echo 1 > /proc/sys/net/ipv4/ip_forward
    modprobe iptable_nat
    iptables -t nat -A POSTROUTING -s $private_subnet -o $public_nic -j SNAT --to-
↪source $public_ip
    iptables -A FORWARD -i $private_nic -j ACCEPT
    ;;
```

(下页继续)

(续上页)

```
  *)
    ;;
esac
exit 0
EOF
chmod +x /etc/init.d/nat
chkconfig nat on
service nat start
```

2.2 安装 AIP

2.2.1 安装前准备

在开始安装 SkyForm AIP 前，请先做好以下准备工作：

- 获取安装包

  从官方网站上下载 https://skyformaip.com。

- 对每一个集群里的主机：
  - 主机使用静态 IP 地址。
  - 集群主机中主机间内网的防火墙是关闭的。
  - 所有主机上的登录用户具有相同的用户名、用户号（uid）和用户组，用 LDAP 作为用户认证的系统无需单独设置。
  - 确保依赖软件包都已安装（详情请参考集群和节点环境准备)。所需的包为 libcurl, libyaml, psmisc, net-tools, libnsl。

    对于图形服务器，安装桌面（如 GNOME Desktop）和 openbox（需要先安装 epel-release)。
  - 挂载共享文件系统中的 SkyForm AIP 的目录（目录结构见下节的图)。
- 选择两个主机作为集群的主节点（master）候选机，SkyForm AIP 的调度器会自动在第一个主节点上运行，当第一个主节点失败后，调度器和监控服务会在第二个主节点上自动启动实现不间断地服务。用户命令会自动找到活的主节点。
- 在用户认证系统中生成一个用户 cadmin 作为 AIP 的管理员。

2.2.2 准备文件系统

SkyForm AIP 的容错依赖于共享文件系统。把可执行文件和配置文件安装在共享文件系统中也便于数据备份、软件升级和支持不同版本的操作系统。SkyForm AIP 对共享文件系统没有特殊要求，只要是 POSIX 兼容即可。

图 2 描述了 SkyForm AIP 的文件结构：

图的左侧是每个主机上文件的结构，右侧是在文件服务器上的文件结构。不同版本的二进制文件会被安装在不同的路径中，只要修改本地存储上的链接，并重启服务，就可在不改变用户环境的情况下切换 SkyForm AIP 的版本，即：

1. 在每个主机上 mount SkyForm AIP 的共享安装, 如/share/skyformai_shared。
2. 在每个主机上建立一个本地空目录以便让安装工具在里面生成链接，如/opt/skyformai。

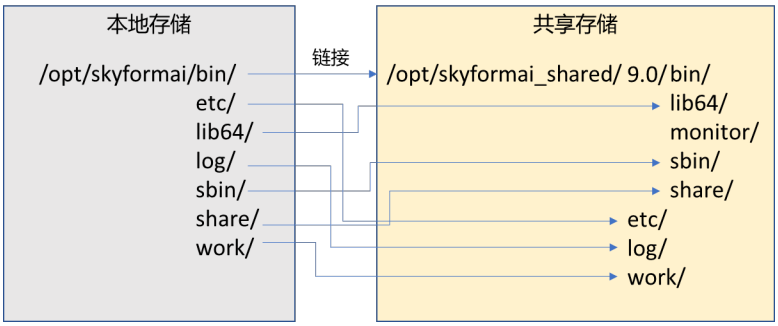


图 3: SkyForm AIP 安装文件目录结构

2.2.3 将 SkyForm AIP 安装到共享文件系统里

在任何一台 mount 好共享文件系统的主机上展开软件包：

```
# tar xvfz skyformai-10.*.tar.gz
```

在以上展开的同一目录下准备文件：

文本文件 hosts 列出 AIP 管理节点和计算节点的 IP 地址和主机名。在有多网络的情况下，仅列出供 AIP 内部通讯所用的网络的 IP 地址。主机名使用不带域名的短主机名，主机名必须相应主机上的 hostname -s 输出一致。文件中不要含 localhost 行。hosts 文件的例子：

```
192.168.20.100 exp001
192.168.20.101 exp002
...
192.168.20.102 exp010
```

警告： hosts 文件中主管理节点必须放在头部。

运行脚本 install，指定共享目录名（缺省为/opt/skyformai_shared）：

```
./install --shared=/share/skyformai_shared
```

小技巧： install 脚本的所有选项参考*install*

2.2.4 安装每台主机

到目前为止所有 SkyForm AIP 的文件已经在共享文件系统中安装完毕，现在需要在**每个主机**上做相应的本地配置。配置工具为 host-setup。运行时需要指定共享目录名（缺省为/opt/skyformai_shared）。例子：

```
/share/skyformai_shared/host-setup
```

host-setup 会完成以下几个动作：

1. 在 *local_top_dir* 中建立链接
2. 设置并启动 Linux 服务 aip，即 SkyForm AIP 的服务
3. 设置用户登录后用 SkyForm AIP 的环境（/etc/profile.d）

例子：

```
[root@linux7 tmp]# ./host-setup --shared=/opt/skyformai_shared
Selected distro is: 9.0
Created symlink from /etc/systemd/system/multi-user.target.wants/aip.service to
/usr/lib/systemd/system/aip.service.
```

备注： SkyForm AIP 软件包缺省为企业版，第一次安装自带 45 天的企业版功能。45 天后自动降级为免费版。若发现 key 有问题，可以修改 /opt/skyformai/etc/cb.yaml 里的参数 enterprise, 注释掉或者删除该参数，然后重启主管理节点上的 AIP 服务。

主机安装企业版功能

1. 可执行文件安装在本地

```
./host-setup --shared=/share/skyformai_shared --deploylocal
```

2. 安装 3D 图形远程可视化和 AI 开发应用环境

```
./host-setup --shared=/share/skyformai_shared --gui
```

3. ssh 登录控制能力，用户只有在某个节点上有作业运行时才可以 ssh 到该节点

```
./host-setup --shared=/share/skyformai_shared --sshcontrol
```

小技巧： host-setup 脚本的所有选项参考*host-setup*

2.2.5 测试

所有主机的安装都完成后，退出终端登录。重新登录以获取 AIP 环境，然后运行命令 aip cluster info 和 aip queue info 并以一般用户的身份提交作业来测试集群是否工作正常。

```
aip cluster info
```

```
[root@a0 aip]# aip cluster info
MASTER  ADMIN  NUM_NODES
a0      root   5

HOST   NCPUS  NGPUS  MAXMEM  MAXSWP  TAGS
a0     2      -      3771M   2048M   ()
a2     2      -      3771M   -       ()
a1     2      -      3771M   -       ()
a3     2      -      3771M   -       ()
win22  -      -      -       -       ()
```

```
aip queue info
```

```
[root@a0 aip]# aip queue info
NAME    PRIO  STATUS  MAX  TOTAL  WAIT  RUN  STOPPED  SL/U  SL/H
medium  3     OK      -    0      0     0    0        -     -
```

AIP 支持 SLURM 和 LSF 命令：

```
[root@a0 aip]# sinfo
PARTITION  AVAIL  TIMELIMIT  NODES  STATE  NODELIST
medium     up     infinite   5      idle   a0 a1 a2 a3 win22
[root@a0 aip]# bqueues
QUEUE_NAME      PRIO      STATUS      MAX  JL/U JL/P JL/H NJOBS  PEND  RUN  SUSP
medium          3     Open:Active      -    -    -    -     0     0     0     0
[root@a0 aip]# bhosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV    gpu
a0                 ok              -      2      0      0      0      0      0  0.000
a1                 ok              -      2      0      0      0      0      0  0.000
a2                 ok              -      2      0      0      0      0      0  0.000
a3                 ok              -      2      0      0      0      0      0  0.000
win22              unavail         -      1      0      0      0      0      0  0.000
```

备注： 集群超过 500 个主机，或者活动作业量超过 100,000 个作业，或者有多个脚本调用 bjobs 命令频繁查询作业状态的集群，master 主机的操作系统需要调参。详见大机群或者高通量负载集群操作系统调参 。

2.3 安装登录节点和 Web 门户

AIP 自带的 web 门户只提供简单功能，可作为演示或一般小集群和少量用户使用。全功能的 web 门户由 SkyForm 算力应用平台提供。

2.3.1 系统准备

安装所需系统软件和服务：

安装依赖的软件包：- EL7

```
yum -y install epel-release
yum -y install nginx php-fpm php php-pecl-yaml
systemctl enable nginx php-fpm
```

- EL8+

```
yum -y install epel-release
yum -y install nginx php-fpm php php-json
systemctl enable nginx php-fpm
```

找到 aip 包里的 php-yaml.el8.x86_64.tar.gz

```
tar xfz php-yaml.el8.x86_64.tar.gz -C /
```

- Ubuntu 20+

```
apt install nginx php-fpm php php-json php-yaml apache2
```

备注： 由于门户集成了对 Grafana 和 Kibana 的可视化嵌入集成，NGINX 中配置了端口 13000 和 15601 的转发。若有其他服务使用这些端口，则需要把其他服务去掉或者修改端口。门户的这两个端口是写死的，修改较困难。

2.3.2 如果 web 门户不是管理节点，安装 AIP 客户端

备注： 如果节点上已经安装了 AIP 服务，请跳转下一节。

在 AIP 包解压的目录下，运行 client-install。运行过程中需要输入主 master 和第 2master 的主机名和相应的 IP 地址。例子见下：

```
[root@a4 aip]# ./client-install
Master host name:linux7
Master host IP address:192.168.20.70
2nd master host name:linux8
2nd master host IP address:192.168.20.80
Found package: 9.20.tar.gz
Press enter to install...
```

安装完后，重新登录更新环境，就可以运行测试命令 aip cluster info。

2.3.3 安装 Web 门户

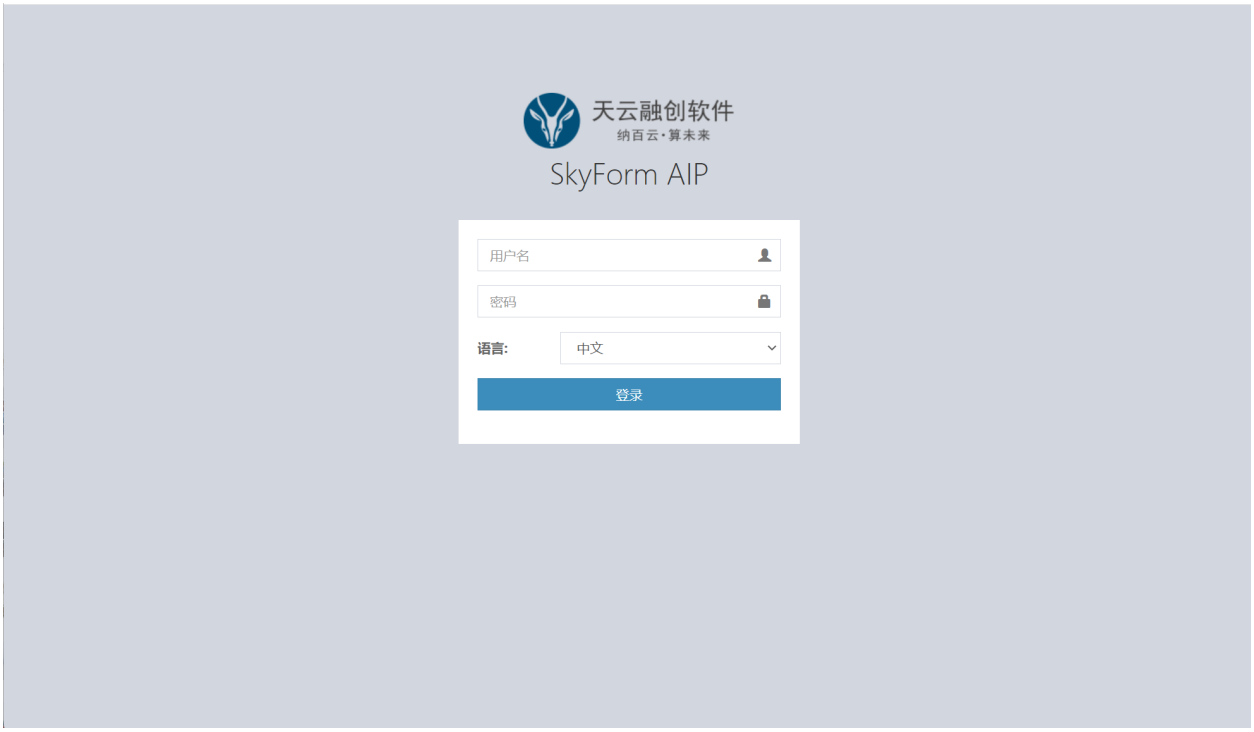
安装 GUI 文件

```
./portal-install
```

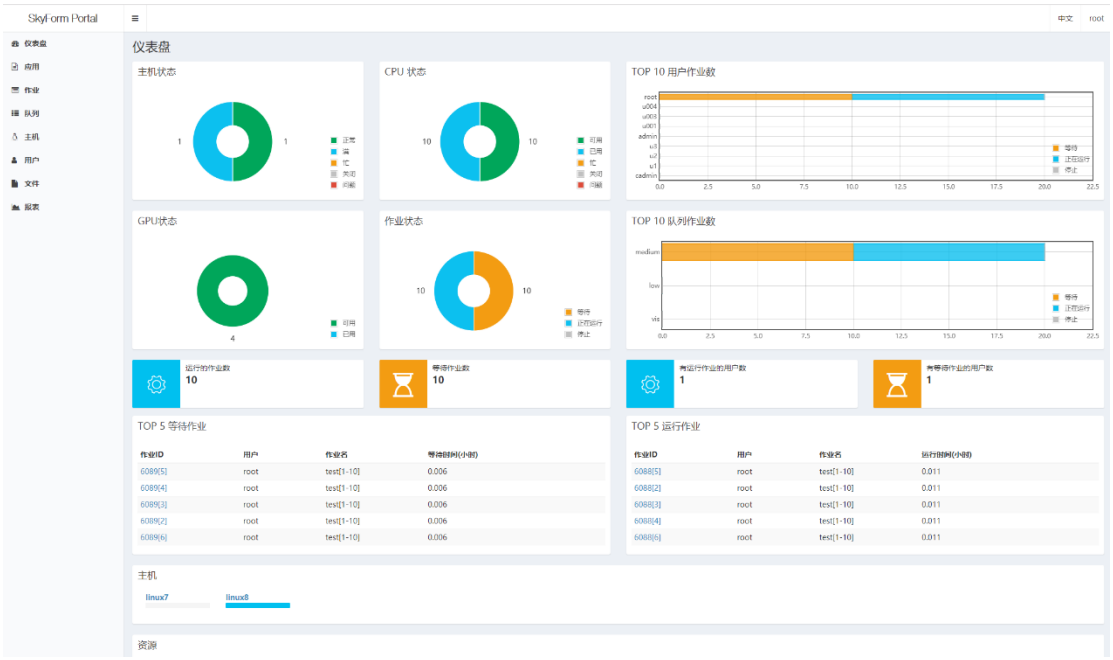
2.3.4 测试

在浏览器中使用机器 IP 访问安装好的 SkyForm Web 用户门户：

https://<gui_server>



Web 门户使用操作系统的用户和密码登录。登陆后 AIP 管理员会显示系统仪表盘



而一般用户登录后显示用户的作业（任务）列表。

2.3.5 门户应用集成

请参考*Web 门户应用集成* 。

2.3.6 集成 AIP 监控数据分析系统

AIP 监控数据分析系统 采集 AIP 调度器的主机负载和作业数据，存放到 Elasticsearch 中，使用 Kibana 做数据分析，并用 Grafana 实现可视化的报表。

AIP 门户自动检测/opt/skyformai/etc/olmon.conf 文件，自动集成 Kibana 和 Grafana 里的可视化图表。

警告： 如果 opt/skyformai/etc/olmon.conf 文件存在，而 Elasticsearch、Kibana、或 Grafana 其中任何一个服务不能访问，门户会卡住。

备注： 如果要禁止与监控数据分析系统的集成功能，可以把/var/www/html/dashboard.php 的第 45 行改成: `$_SESSION['es'] = FALSE;`

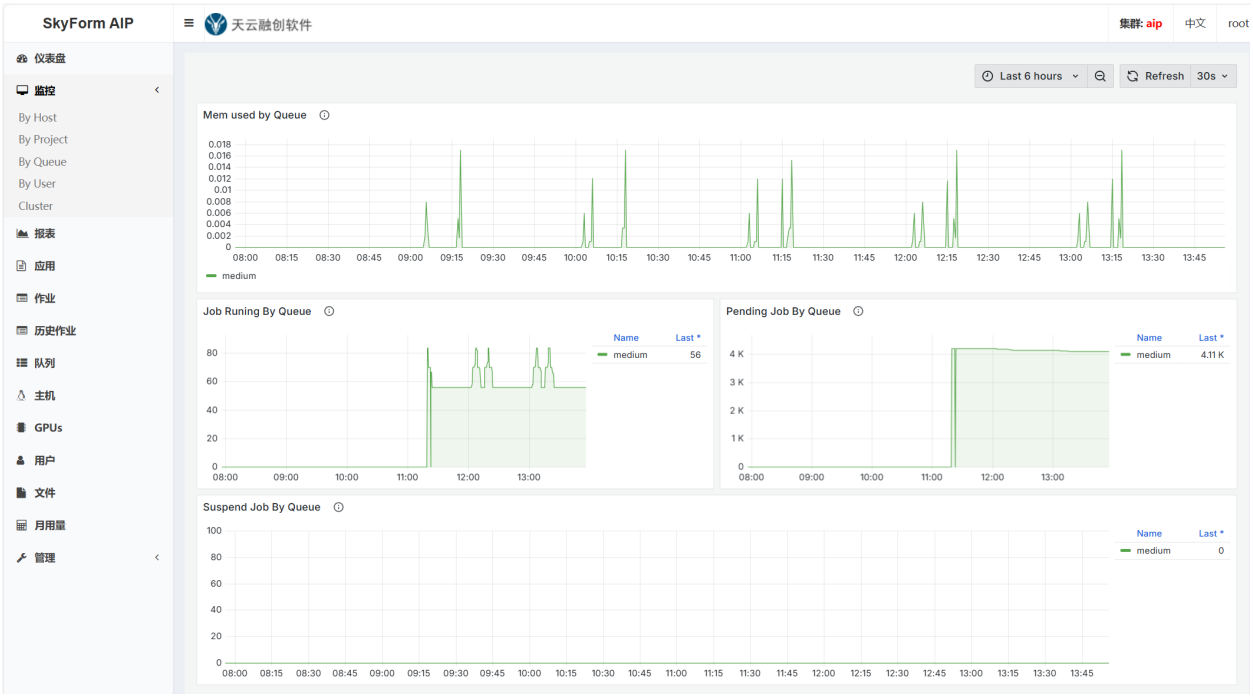
由于门户使用 HTTPS，而一般 Kibana 和 Grafana 使用 HTTP，需要在门户的 NGINX 中实现转发。

- 修改/etc/nginx/nginx.conf 的第 60 行里的 Kibana 访问 URL，如 `proxy_pass http://10.23.10.24:5601/;`。
- 修改/etc/nginx/nginx.conf 的第 79 行里的 Grafana 访问 URL，如 `proxy_pass http://10.23.10.24:3000/;`。

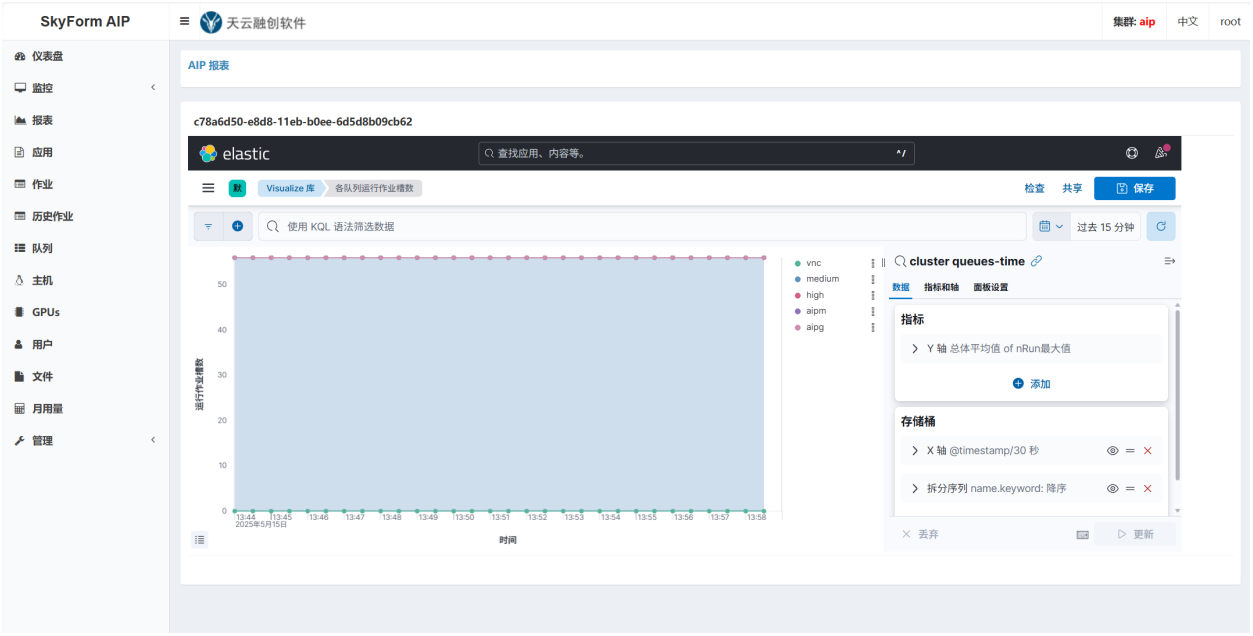
修改后重启 NGINX:

```
systemctl restart nginx
```

管理员登录门户后，可以看到左边菜单里的“监控”，列出 Grafana 里的仪表盘，点击任意一项可显示仪表盘的内容。



点击左侧菜单中的“报表”可以显示 Kibana 里配置的可视化标题，点击任意标题可显示可编辑的内容。



2.3.7 排错

如果用户无法登录，或者登录后看不到作业，用普通终端登录到管理主节点，检查 AIP 的状态。有些系统中如果安装过其他调度器，如 SLURM，里面会有与 AIP 相重的命令，如 lsid，这些命令会导致门户的失败，需要去掉这些命令。

2.4 升级 AIP

2.4.1 同版本更新

若版本不变，更新 SkyForm AIP 的代码只需停掉各主机上的 AIP 服务，更新 AIP 文件，然后重启 AIP 服务。更新不会影响正在运行或等待的作业。

1. 更新 AIP 文件：

   install –shared=*top_share_dir* –localtop=*local_top_dir*

   install 脚本不会清除或修改已有配置文件和作业数据

   例子：

```
./install --shared=/share/skyformai_shared --localtop=/opt/skyformai
```

   小技巧： install 脚本的所有选项参考*install*

2. 在 master 上重启所有主机上的 AIP 服务：

```
aip admin rs all # 重启所有主机上的AIP daemon
aip admin rcs    # 重启调度器
```

   警告： 升级过程中不要用 Linux 服务控制对 AIP 服务进行操作 systemctl start|stop|restart aip，否则正在运行的作业会退出。

2.4.2 版本间升级

升级 SkyForm AIP 的升级只需更新 AIP 文件，然后重新运行 host-setup 脚本，升级不会影响正在运行或等待的作业。

1. 更新 AIP 文件：

   install –shared=*top_share_dir* –localtop=*local_top_dir*

   install 脚本不会清除或修改已有配置文件和作业数据

   小技巧： install 脚本的所有选项参考*install*

2. 在集群的每台主机上运行 host-setup：

   ./host-setup –shared=*top_share_dir* –distro={new_version} –upgrade

   例子：

```
./host-setup --shared=/share/skyformai_shared --distro=10.25.0 --upgrade
```

若有 hosts 文件，其中每行列出集群里的 hostname，则可用以下的脚本在每个主机上自动运行 host-setup：

```
#!/bin/bash
for host in \`cat hosts \| awk '{print $2}'\`; do
    ssh $host ./host-setup --shared=/share/skyformai_shared --distro=10.24.0 --
↪upgrade
done
```

警告： 升级使用 host-setup 必须定义--upgrade 参数，否则正在运行的作业会退出。

小技巧： host-setup 脚本的所有选项参考*host-setup*

3. 在 master 上重启所有主机上的 AIP 服务和调度器：

```
aip admin rs all     # 重启所有主机上的AIP daemon
aip admin rcs        # 重启调度器
```

警告： 升级过程中不要用 Linux 服务控制对 AIP 服务进行操作 systemctl start|stop|restart aip，否则正在运行的作业会退出。

2.4.3 Web 门户代码版本升级

SkyForm AIP 的 web 门户代码包的安装脚本为 portal-install。升级代码只需在解压新版的 AIP 包后，以 root 在解压包的目录里运行命令：

```
./portal-install
```

2.5 配置

SkyForm AIP 系统的配置文件都是 YAML 格式，一般存放在/opt/skyformai/etc (环境变量 $CB_ENVDIR) 目录中。

2.5.1 调度器配置 cb.yaml

SkyForm AIP 安装后形成一个缺省的 cb.yaml 文件，这个文件是所有集群主机共享的，所以一般放在 *localtop*/etc 里（如：/opt/skyformai/etc）。cb.yaml 具有以下的结构。

本文列出 cb.yaml 基本的配置参数，**详细参数参考参考文档***cb.yaml* 。

集群定义

类名：cluster

Cluster 定义集群主机成员、管理员、集群参数等信息：

- 集群管理员（必填）：administrators，定义至少一个管理员用户。第一个用户为主管理员，AIP 安装的目录结构中 work/data 的 owner 必须是该用户。例子：

```
administrators:
- cadmin
- root
```

- 标签资源（可选项）：resources
  - 标签（必填）：name
  - 描述（可选项）：description
  - 类别：type（值必须是 tag）
- 主机（必填）：host
  - 主机名（必填）：name，大机群中主机名顺序数字结尾的，如 node0001，node0002，…，node0100，可以用一行定义多个主机名：node[0001-0100]
  - 最大作业槽数（可选项）：maxslots，若为数字（如”4”），则定义主机上的最多槽数，若为字串”cpu”，则为主机上 CPU 超线程数总和，缺省则为无限制。若每个主机不填，可以定义主机名为 default 的主机，通配 maxslots，所有没有定义 maxslots 的主机缺省都会使用该值作为最大作业槽数。
  - 主机具有的标签资源（可选）：tags
  - 主机类型（可选）：attr，可以选择 server（缺省）即计算服务器，client 即固定客户端。
- 允许任意客户机连接（缺省：不在 cb.yaml 里定义的主机不能与 AIP 服务连接）：clients: yes

> **警告：**若集群网络是开放的，clients: yes 有安全隐患，会允许任意联网的主机连接。

- 调度单元 CPU 核数（可选）：define_ncpus，可取的值为：
  - cores: 每个调度单元的 CPU 核对应于一个物理核 (缺省)。
  - threads：每个调度单元的 CPU 核对应与一个超线程 (hyperthread) 逻辑核，一般一个物理核为两个逻辑核。
- 企业版：enterprise，可取的值为 yes 或 no。如果值为 yes，则必须有合法的授权 key 文件才可以正常工作，否则用户不能提交作业。
- ssh 访问控制：sshcontrol（可选）：当安装 AIP 选择使用 ssh 控制（host-setup –sshcontrol）后的进一步控制。可取的值为：
  - none：取消 ssh 访问控制
  - job: 用户可以从任意地方 ssh 到有作业运行的主机上
  - 缺省：用户不能 ssh。只有同一作业的主机间可以互相 ssh，这个功能允许 MPI 作业使用 ssh 分发和控制子任务

队列定义

类名：queues

队列是作业的承载体。作业提交个 AIP 时必须指定一个队列，或用缺省的队列。

队列可以定义多个，每个队列的参数为：

- 队列名（必填）：name
- 优先级（必填）：priority 值为 1-100, 数值越大，优先级越高
- 队列作业槽配额（可选，缺省为无限制）：maxslots
- 队列可用主机（可选，缺省可用集群所有主机）：hosts。如：hosts: “node0001 node0002” 用空格隔开的多个主机名。企业版 AIP 这里也可以用主机组名。
- 可用队列的用户（可选，缺省为所有用户可用）：users。如：users: a1。企业版 AIP 这里也可以用用户组名。
- 等待中的大并行作业预留小作业结束后的作业槽时间（可选，缺省为不预留），单位：分钟：slotreservetime。这个功能对于有大的并行作业的进群很重要。否则空出来的作业槽会被队列中的小作业所占，大作业得到作业槽的机会很小。
- 作业运行时长限制（可选，缺省为无限）：runlimit，单位：分钟。超过运行时长的作业会被自动杀掉。
- 主机挂后作业自动重新调度（可选，缺省为不重新调度）：rerunonhostfail: yes
- 作业以某些退出码退出后重调度（可选，缺省为不重新调度）：rerunonexitcode: 1 3 5，这个例子是当作业退出码为 1、3、5 作业自动重新调度运行。
- 公平分享调度算法（企业版功能）：fairshare: ‘[default,1]’
- 抢占（企业版功能），定义可被抢占作业的低优先级队列名，如果没有资源，本队列里的等待作业可以抢占低优先级队列中运行作业的资源：preemption: low

通用调度参数

类名：general

- 缺省队列名 (不可缺省)：default_queue，其值必须是上面所定义的其中一个队列的值
- 调度器在内存里保存结束作业信息的时间：memperiod。缺省为一个小时
- 用 Linux cgroup 约束作业所用资源：cgroup，可选的参数为：
  - “acct”：调度器用 cgroup 跟踪作业进程，确保计量准确性。
  - “cpu”：调度器根据主机的 maxslots 计算作业任务可用 CPU 的份额，如 maxslots 为 8，则一个任务至少可用 1/8 主机里 CPU 的总和。
  - “gpu”：**AIP 企业版功能**。支持提交作业时使用 need[gpu=xx] 申请 GPU 的作业内约束只能使用所分配的 GPU。
- 禁止 root 用户运行作业。rootcanrun: no

  缺省：root 用户可与普通用户一样运行作业。
- 多 GPU 卡共享调度策略：gpu_share_spread，当一台主机上的多个 GPU 卡使用共享模式时，即每个 GPU 卡被多个作业共享，该参数值为” yes” 表示调度作业尽量分散到各个卡上，实现负载平衡。缺省取值为” no”，表示调度作业占用尽量少的 GPU 卡。

企业版的功能。详情参考文档*cb.yaml*。

用户组定义

企业版的功能。

类名：usergroups - 组名：name: 名字

- 成员：members: 以空格隔开的用户名
- 组管理员：administrator: 组管理员用户名。组管理员可以管理组内成员的作业。
- 组最大作业槽数：maxslots: 全组最多可用的 CPU 核数
- 每个组员最大作业槽数：usermaxslots: 每个组员最多可用的 CPU 核数
- 组最大作业数：maxjobs: 全组所有可运行作业的最大数
- 组最大 GPU 卡数：maxgpus：全组所有运行作业可用的 GPU 卡数总和

主机组定义

企业版的功能。

类名：hostgroups - 组名：name: 名字

- 成员：members: 以空格隔开的集群内配置的主机名

配额限制

企业版的功能。定义各个维度的配额限制。

类名：limits - 配额名：name: 名字

- 配额用于队列名：queues(配额用于所有列出的队列) 或 per_queue（配额用于每个列出的队列）：多个以空格隔开的队列名
- 配额用于主机名：hosts(配额用于所有列出的主机) 或 per_host（配额用于每个列出的主机）：多个以空格隔开的主机名
- 配额用于项目名：projects(配额用于所有列出的项目) 或 per_project（配额用于每个列出的项目）：多个以空格隔开的项目名
- 配额用于用户名：users(配额用于所有列出的用户) 或 per_user（配额用于每个列出的用户）：多个以空格隔开的用户名
- 配额用于主机名：apps(配额用于所有列出的应用) 或 per_app（配额用于每个列出的应用）：多个以空格隔开的应用名

配额定义：- slots：最大 CPU 核数

- jobs：最多运行作业数
- resources：资源定义，例子：[gpu,4] 最多 4 个 GPU 卡

2.5.2 计量计费配置 cbcrond.yaml

计量输出类名：usage

- enabled: true 或者 false，是否输出计量数据，计量数据存放在/opt/skyformai/work/usages
- outputperiod: 输出间隔分钟，一般为 15，即每 15 分钟一次

其他参数详见/opt/skyformai/etc/cbcrond.yaml *cbcrond.yaml* 里的例子。

2.5.3 典型的 EDA 负载集群配置

电子设计自动化（EDA）的负载一般为运行时间数分钟到数小时的作业，一般作业量都较大，几千到几百万不等。单个作业一般在单台主机上跑，有些需要大内存。

EDA 集群的配置有以下特点：

1. 用 CPU 逻辑核。缺省 AIP 使用 CPU 物理核调度作业，定义 define_cpus: threads：每个调度单元的 CPU 核对应与一个 CPU 逻辑核。
2. 不使用核绑定。缺省 AIP 的配置中，每台主机的 maxslots 与 CPU 核数相同（maxslots: cpu），这样每个主机都启用作业的 CPU 核绑定。需要定义 cpubind: no 来禁用作业 CPU 核绑定，这样一些多线程的 EDA 应用可以获得足够的 CPU 资源。
3. 操作系统系统打开交换区（swap），这个可以用*chinfo* 来查看，确保主机上的 maxswap 有值。打开 swap 是 Linux 内的配置，AIP 只是自动检测。关于如何配置 Linux 的 swap，请搜索互联网。
4. 很多 EDA 应用需要用大内存，为了防止内存过载，在队列或相应的主机中设置调度负载阈值，当内存低于一定值的时候停止调度作业。
5. LSF 兼容性：参数 lsf_compatible: yes，这是确保作业的输出最后提供与 LSF 兼容的输出结果，一些 EDA 应用会以此判断。
6. 一些 EDA 应用会运行命令 lsid 来判断是否是 LSF 环境，生成/opt/skyformai/etc/product 文件，里面放置一行”LSF 10.1”，这样 AIP 的 lsid 命令输出的第一行就会显示”LSF 10.1…”。
7. 一些企业使用脚本管理 EDA 工作流，提交作业后循环调用 bjobs 命令查询作业状态。这对调度器产生一定的压力，影响调度性能。启用 jservice，使 bjobs 命令从 jservice 获得作业状态，而不直接与调度器打交道，配置/opt/skyformai/etc/jservice.yaml, 具体例子见下。

cb.yaml:

```
cluster:
  name: aip
  administrator:
  - cadmin
  resources:
  - name: linux6
    type: tag
  - name: linux7
    type: tag
  define_cpus: threads # 用CPU逻辑核
  enterprise: yes
  clients: yes
  lsf_compatible: yes
  hosts:
  - name: mgt01
    maxslots: 0         # 主管理主机不运行作业
  - name: comp01
    tags:
```

(下页继续)

(续上页)

```
    - linux6
    thresholds:
    - "mem: 1G/0.7G"    # 内存小于1GB时停止接受新的作业，小于0.7G时暂停上面运行的作业
  - name: comp02
    tags:
    - linux7
    thresholds:
    - "mem: 2G/1G"      # 内存小于2GB时停止接受新的作业，小于1G时暂停上面运行的作业
  - name: default       # 所有主机的通用参数设置
    maxslots: cpu
    cpubind: no         # 不用CPU核绑定
queues:
- name: high
  priority: 10
- name: medium
  priority: 3
general:
  default_queue: medium
  mailprog: ":"         # 作业结束后不发邮件
  cgroup: acct          # 利用Linux cgroup跟踪作业进程，确保杀掉作业时没有进程遗漏
```

2.5.4 典型的 CAE（工程设计仿真）、HPC 集群的配置

HPC 应用一般为 MPI 作业，单个作业需要跨机运行。同时对于 CAD 图形应用，需要有 GPU 提供图形加速。HPC 的配置有以下考虑：

1. 使用 CPU 物理核调度作业。AIP 缺省用物理核调度。
2. 允许 CPU 绑定，以达到最佳性能。
3. 图形作业使用共享 GPU 时，在多 GPU 的主机上实现多 GPU 间的负载均衡。

cb.yaml:

```
cluster:
  name: aip
  administrator:
  - cadmin
  define_cpus: cores    # 用CPU物理核
  enterprise: yes
  clients: yes
  hosts:
  - name: mgt01
    maxslots: 0         #主管理主机不运行作业
  - name: comp01
  - name: comp02
  - name: default       # 所有主机的通用参数设置
    maxslots: cpu
    cpubind: yes        # 启用CPU核绑定
queues:
- name: high
  priority: 10
  memlimit: percore     # 作业内存限制按核平均分配
- name: medium
  priority: 3
```

(下页继续)

(续上页)

```
  memlimit: percore    # 作业内存限制按核平均分配
general:
  default_queue: medium
  gpu_share_spread: yes  # 每台主机上GPU负载均衡的调度
  mailprog: ":"        # 作业结束后不发邮件
  cgroup: acct         # 利用Linux cgroup跟踪作业进程，确保杀掉作业时没有进程遗漏
```

jservice.yaml:

```
jservice_enabled: no      # 禁用jservice，作业状态查询由调度器提供服务
cbcrond_enabled: yes      # 启动cbcrond提供用量统计
```

2.5.5 典型的超算中心和智算中心集群配置

超算和智算中心确保资源分配和资源使用控制。计量和计费为关键。配置有以下考虑:

1. 使用 CPU 物理核调度作业。AIP 缺省用物理核调度。
2. 允许 CPU 绑定，以达到最佳性能。
3. 利用 Linux cgroup 控制 GPU 的使用

cb.yaml:

```
cluster:
  name: aip
  administrator:
  - cadmin
  define_cpus: cores    # 用CPU物理核
  enterprise: yes
  clients: no           # 禁用浮动客户端
  hosts:
  - name: mgt01
    maxslots: 0         #主管理主机不运行作业
  - name: comp01
  - name: comp02
  - name: login
    attr: client        # 登录节点，配置成客户端
  - name: default       # 所有主机的通用参数设置
    maxslots: cpu
    cpubind: yes        # 启用CPU核绑定
queues:
- name: high
  priority: 10
  memlimit: percore    # 作业内存限制按核平均分配
- name: medium
  priority: 3
  memlimit: percore    # 作业内存限制按核平均分配
general:
  default_queue: medium
  gpu_share_spread: yes  # 每台主机上GPU负载均衡的调度
  mailprog: ":"        # 作业结束后不发邮件
  cgroup: acct gpu     # 利用Linux␣
↪cgroup跟踪作业进程，确保杀掉作业时没有进程遗漏，并控制GPU的使用
```

jservice.yaml:

```
jservice_enabled: no      # 禁用jservice，作业状态查询由调度器提供服务
cbcrond_enabled: yes      # 启动cbcrond提供用量统计
```

2.6 调度器动态配置参数

SkyForm AIP 调度器的部分参数可以动态修改。动态修改这些配置参数不会重启调度器，在调度器内存中的作业数据不会受到影响。参数调整后，调度器会自动更新或生成新的配置文件。这些配置文件都保存在/opt/skyformai/etc 中。原有的文件会被修改成 文件名.时间戳。

只有 root 或 AIP 集群管理员可以修改调度器参数。AIP 集群管理员定义在*cb.yaml* 的 cluster:adminnistrators 里。

配置修改的命令为 **aip** 。

2.6.1 修改主机组

命令：**aip hostgroup|hg**

当参数修改成功后，cb.yaml 文件中的 hostgroups 段落会被删除，所有主机组配置存放在/opt/skyformai/etc/hg.yaml 中。

新增主机组

命令：**aip hostgroup|hg create|c 主机组名主机名主机名…**

例子:

```
aip hg c research1 node10 node13
cmgroup research1
GROUP_NAME    HOSTS
research1     node10 node13
```

条件：主机组里的主机名必须是集群中合法的主机名。

删除主机组

命令：**aip hostgroup|hg delete|d 主机组名**

例子:

```
aip hg d research1
cmgroup research1
research1: No such user group or host group
```

条件：主机组名存在于调度器中，并且没有被配置到队列（queues）或 limits 中。

添加主机组成员

命令：**aip hostgroup|hg addmembers|am 主机组名主机名主机名…**

例子:

```
aip hg am research1 node14 node16
cmgroup research1
GROUP_NAME    HOSTS
research1     node10 node13 node14 node16
```

条件：主机组必须已经建立，添加的主机名必须是集群中合法的主机名，而且**这些主机上没有运行的作业**。

减少主机组成员

命令：**aip hostgroup|hg delmembers|dm 主机组名主机名主机名…**

例子:

```
aip hg dm research1 node10 node13
cmgroup research1
GROUP_NAME    HOSTS
research1     node14 node16
```

条件：主机组必须已经建立，减少的主机名必须是集群中合法的主机名，而且**这些主机上没有运行的作业**。

2.6.2 修改用户组

命令：**aip usergroup|ug**

当参数修改成功后，cb.yaml 文件中的 usergroups 段落会被删除，所有用户组配置存放在/opt/skyformai/etc/ug.yaml 中。

备注： 除了新增和删除用户组需要集群管理员的权限，用户组管理员有权限修改用户组其他参数，包括：增删组员和各种上限。用户管理员也可修改组管理员，修改后自己就失去了该组的管理员权限。

新增用户组

命令：**aip usergroup|ug create|c [JSON 用户组定义文件 | JSON 用户组定义字串]**

用户组定义的 JSON 格式如下:

```
{
  # 用户组名，值不能为空、与已有的用户同名，或者与系统中任何一个用户同名
  "Group": "组名",

  # 组员，值不能为空，可含多个用户、或已有的其他组名。
  # 如果成员名只有一个值为：@system，表示组名与操作系统中（如LDAP）的一个组同名，
  # 成员为系统中改组的成员。调度器每小时从系统中同步一次。同步间隔可以用cb.yaml里的
  # egroup_update_interval调整
  "Members": ["用户1", "用户2", "用户组2"],

  # 组管理员。可选填。用户必须是操作系统中已有的用户
```

(下页继续)

(续上页)

```
  "Admin": "",

  # Fairshare串，选填，一般为空
  "UserShares": "",

  # 用户组总运行作业槽上限，选填，缺省为-1，即无限
  "MaxSlots": -1,

  # 每个组成员用户运行作业槽上限，选填，缺省为-1，即无限
  "UserJobLimit": -1,

  # 用户组总运行作业数上限，选填，缺省为-1，即无限
  "MaxJobs": -1,

  # 用户组总GPU卡数上限，选填，缺省为-1，即无限
  "MaxGpus": -1
}
```

添加后可以用命令 **aip ug i** 或*cugroup* 查看结果。

条件：用户名必须是集群操作系统中合法的用户名。

删除用户组

命令：**aip usergroup|ug delete|d 用户组名**

例子:

```
aip ug d 我的组
```

条件：用户组名存在于调度器中，并且没有被配置到队列（queues）或 limits 中。

添加用户组成员

命令：**aip usergroup|ug addmembers|am 用户组名用户名用户名…**

例子:

```
aip ug am 我的组 u003 u004
```

条件：用户组必须已经建立，添加的用户名必须是集群操作系统中合法的用户名。

减少用户组成员

命令：**aip usergroup|ug delmembers|dm 用户组名用户名用户名…**

例子:

```
aip ug dm 我的组 u001 u002
```

条件：用户组必须已经建立，减少的用户名必须是集群操作系统中合法的用户名。

修改用户组管理员

命令：**aip usergroup|ug admin|a 用户组名管理员用户名**

用户组的管理员可以观察和管理组成员的作业、以及修改组成员、修改用户组资源上限、并修改用户组管理员（修改后自己就失去了管理员权限）。

备注： 用户组管理不一定必须是该用户组的成员。

例子:

```
aip ug a 我的组 cadmin
```

修改用户组资源限制

命令：**aip usergroup|ug grouplimit|gl 组作业槽上限 (MaxSlots) [组作业数上限 (MaxJobs) [组 GPU 卡上限 (MaxGpus)]]**

每一项的值：

- “-1”：无限
- “-2”：原有的值不变
- 任何 0 或其他正整数：修改的值

例子 1，修改组作业槽上限到 100:

```
aip ug gl 我的组 100
```

例子 2，修改组作业数上限到 20:

```
aip ug gl 我的组 -2 20
```

例子 3，修改组 GPU 卡上限到 10:

```
aip ug gl 我的组 -2 -2 20
```

例子 4，去掉所有上限:

```
aip ug gl 我的组 -1 -1 -1
```

例子 5，修改作业槽上限到 100，修改组 GPU 卡上限到 5:

```
aip ug gl 我的组 100 -2 5
```

修改用户组成员作业槽上限

命令：**aip usergroup|ug userlimit|ul 组员作业槽上限 (UserJobLimit)**

修改的值：

- “-1”：无限
- 任何 0 或其他正整数：修改的值

例子 1，修改组成员作业槽上限到 20:

```
aip ug ul 我的组 20
```

例子 1，去掉组成员作业槽上限:

```
aip ug ul 我的组 -1
```

2.6.3 修改用户作业上限

命令：**aip user|u**

当参数修改成功后，cb.yaml 文件中的 users 段落会被删除，所有用户作业上限配置存放在/opt/skyformai/etc/u.yaml 中。

修改用户运行作业槽上限

命令：**aip user|u limit|l 用户名运行作业槽上限**

例子 1，修改用户 u001 的运行作业槽上限到 64:

```
aip u l u001 64
cusers u001
USER/GROUP          MAXPEND     MAX   NJOBS    PEND     RUN   SSUSP   USUSP     RSV
u001                      -      64       0       0       0       0       0       0
```

例子 2，去除用户 u001 的运行作业槽上限:

```
aip u l u001 -1
cusers u001
USER/GROUP          MAXPEND     MAX   NJOBS    PEND     RUN   SSUSP   USUSP     RSV
u001                      -       -       0       0       0       0       0       0
```

修改用户最多等待的作业数

命令：**aip user|u pendlimit|pl 用户名等待作业数上限**

例子 1，修改用户 u001 的等待作业数上限到 1000:

```
aip u pl u001 1000
cusers u001
USER/GROUP          MAXPEND     MAX   NJOBS    PEND     RUN   SSUSP   USUSP     RSV
u001                   1000       -       0       0       0       0       0       0
```

例子 2，去除用户 u001 的运行作业槽上限:

```
aip u l u001 -1
```

2.6.4 修改队列

命令：**aip queue|q**

当参数修改成功后，cb.yaml 文件中的 queues 段落会被删除，所有队列配置存放在/opt/skyformai/etc/queue.yaml 中。

备注： 只有集群管理员才可以修改队列。

新增队列

命令：**aip queue create|c [JSON 队列定义文件 | JSON 队列定义字串]**

队列定义的 JSON 格式如下:

```
{
  # 队列名，值不能为空、与已有的队列同名
  "Name": "队列名",

  # 优先级，值不能为空。可以是任意正整数，数字越大，优先级越高
  "Priority": 3,

  # 说明。可选填
  "Description": "",

  # 主机，选填，多个主机或主机组名用空格隔开。如果为空，表示队列可用集群所有主机
  "Hosts": "主机1 主机组2",

  # 用户，选填，多个用户名或用户组名用空格隔开。如果为空，表示所有用户都可已用该队列
  "Users": "",

  # 队列运行作业槽上限，选填，缺省为-1，即无限
  "Maxslots": -1,

  # 每个用户在这个队列中运行作业槽上限，选填，缺省为-1，即无限
  "Usermaxslots": -1,

  # 是否为专属队列，选填，缺省为 “no”
  "Dedicated": "no",

  # 缺省资源需求，选填，缺省为无
  "Resreq": "",

  #␣
↪队列管理员用户名，多个用户名可用空格隔开，可以管理队列中其他用户的作业，选填，缺省为无
  "Administrators": ""
}
```

备注： 队列的其他参数可以后继通过编辑文件和重启调度器生效。

添加后可以用命令 **aip q i** 或*cqueues* 查看结果。

删除队列

命令：**aip queue|q delete|d 队列名**

例子:

```
aip q d queue1
```

条件：队列名存在于调度器中，不是缺省队列，并且队列中没有作业，队列没有被配置在 limits 中。

修改队列参数

命令：**aip queue update|u JSON 队列参数字串**

JSON 队列参数字串中 Name 必须指定，表示所修改的队列。参数只支持以上新增队列中可以指定的参数，其他参数需要通过编辑/opt/skyformai/etc/queue.yaml，并重启调度器生效。

例子，修改队列 testqueue 的用户和资源需求:

```
aip q u '{"Name":"testqueue","Users":"u001 u002","Resreq":"order[ut]"}'
```

2.6.5 修改调度参数

命令：**aip parameter|p update|u '参数 JSON 字串'**

> **警告:** 由于系统 yaml 包的兼容性问题，该功能只在 x86_64 上测试过，在 ARM 上不能正常工作。

当参数修改成功后，原 cb.yaml 文件备份至/opt/skyformai/etc/history 中，/opt/skyformai/etc/cb.yaml 中的 general 段落会被更新。

以下参数可以放到**参数 JSON 字串**里，每个参数的定义参考*cb.yaml* 里 general 的说明。

Memperiod

general:memperiod 参数，已完成作业在调度器内存中存放的时间，单位为秒。

Default_queue

general:Default_queue 参数，缺省队列名，多个名字放到同一字符串的值中，用空格隔开，如" medium vnc"。

Rootcanrun

general:rootcanrun 参数，值为”yes”或”no”。是否允许 root 用户提交作业。

Idle_action_trigger_duration

general:idle_action_trigger_duration 参数，值为大于 1 的正整数，单位分钟。定义检测作业 CPU 利用率不够后多长时间触发配置的脚本。

Elastic_job

general:Elastic_job 参数，值为”yes”或”no”，是否允许弹性作业调度。

Gpu_share_spread

general:Gpu_share_spread 参数，值为”yes”或”no”。在 GPU 分片或 VGPU 场景下，调度器是否在每台主机上调度 GPU 时考虑负载均衡。

Sched_interval

general:sched_interval 参数，值为正整数，单位秒。调度器两次调度作业的间隔。

Jm_interval

general:jm_interval 参数，值为正整数，单位秒。每台主机上的作业管理器（cbjm）定时检查作业状态的间隔。

Max_array_size

general:max_array_size 参数，值为正整数，作业阵列最大单元数。

Job_fail_close_host

general:job_fail_close_host 参数，作业连续失败后关闭主机的配置。具体语法参考*cb.yaml* 。

Max_cbdata_job_num

general:max_cbdata_job_num 参数，最大 cb.data 文件数。

Max_stream_records

general:max_stream_records 参数，/opt/skyformai/work/data/stream 中文件的最多记录（行）数。

Estream_interval

general:estream_interval 参数，cbsched 定期把作业、队列、主机等数据写到一个 estream_dir 下的间隔时间，单位：秒。

Max_periodic_task_interval

general:max_periodic_task_interval 参数，cbsched 定期检查、执行非调度任务（如作业清理，cb.data 滚动等）的时间间隔，单位：秒。

Max_jobclean

general:max_jobclean 参数，cbsched 每次从内存中清理结束的作业时，为了及时响应命令，内次清理的最多作业数。

User_view_alljobs

general:user_view_alljobs 参数，值为”yes”或”no”。是否允许一般用户查看所有用户的作业信息。

> **警告：** 这个参数调整后 jservice 需要手工重配置 (csadmin jsreconfig)，否则 bjobs 命令查到的作业数据不受影响。

例子:

```
aip p u '{"Memperiod":1000,"Max_jobclean":1500}'
```

以上命令修改 Memperiod 和 Max_jobclean，修改后可用命令”aip p i”查看:

```
aip p i
{
  "ParamConf": {
    "Memperiod": 1000,
    "Default_queue": "medium",
    "Cgroup": "acct",
    "Idle_action_trigger_duration": 1,
    "Elastic_job": "yes",
    "Sched_interval": 4,
    "Jm_interval": 3,
    "Max_array_size": 10000,
    "Preemption_suspend": "yes",
    "Job_terminate_interval": 10,
    "Mailprog": ":",
    "User_view_alljobs": "yes",
    "Max_cbdata_num": 1024,
    "Max_stream_records": 100000,
    "Max_cbdata_job_num": 100000,
    "Egroup_update_interval": 3600,
```

(下页继续)

(续上页)

```
    "Dedicated_queue_include_unavail": "yes",
    "Max_jobid": 2147483646,
    "Estream_interval": 30,
    "Max_periodic_task_interval": 300,
    "Estream_dir": "/opt/skyformai/work/data/estream",
    "Max_jobclean": 1500
  },
  "PowerConf": {},
  "ScaleConf": {}
}
```

2.6.6 修改部分 limit 参数

命令：**aip limit|l update|u '参数 JSON 字串'**

JSON limit 参数字串中 Name 必须指定，表示所修改的 limit。当参数修改成功后，cb.yaml 文件中的 limits 段落会被删除，所有 limit 配置存放在/opt/skyformai/etc/limit.yaml 中。

备注:

- limit 管理员有权限修改参数。limit 管理员也可修改 limit 管理员用户名，修改后自己就失去了管理该 limit 的权限。
- 增删 limit 或修改 limit 其他参数可以通过修改 cb.yaml 或 limit.yaml 文件，然后重启调度器生效：csasdmin reconfig

---

以下参数可以放到**参数 JSON 字串**里，每个参数的定义参考*cb.yaml* 里 limit 的说明。

Name

所修改 limit 的名字，必填。该名字的 limit 必须已经存在与系统中。

Admin

limit 管理员用户名，可修改。

Slots

修改 slots 上限。

Jobs

修改作业数上限。

ResName

修改资源上限，定义资源名，如 gpu。ResLimit 必须同时定义。

ResLimit

修改资源使用上限。ResName 必须同时定义。

例子 1，用户 u001 是 limit vnc 的管理员:

```
$ cresources
LIMIT  vnc
ADMIN: u001
APPS     : cubevnc
PER_USER : all
  expand : root u002 u003 cadmin u001
JOBS       : 2

$ aip l u '{"Name":"vnc","Jobs":1}'      #把作业数上限改成1
```

例子 2，修改 GPU 使用上限:

```
$ cresources
LIMIT  gpu1
ADMIN  u002
USERS    : test1
  expand : u002 u003
  RESOURCE     : gpu : 1.0

$ aip l u '{"Name":"gpu1","ResName":"gpu","ResLimit":2}'
```

2.7 资源管理

通过 SkyForm AIP 集群资源管理，将分散在不同节点上的物理资源聚合起来，根据系统内存、闲置 CPU 容量、可用 GPU、磁盘空间等参数，以及用户的作业定义来为应用作业调度并分配资源，支持数百万个作业，数千台主机的超大规模集群。

2.7.1 什么是资源

在 SkyForm AIP 系统中，CPU、GPU、内存、作业槽、存储等均为系统中的资源。集群则可以看作是一堆资源的集合。

SkyForm AIP 系统持续的进行供需匹配。一旦集群中的某一个或多个主机可提供的资源能满足某个作业定义的资源要求，且该作业满足当前生效的调度策略，该作业就被分配至这个或这些主机上执行。

2.7.2 资源定义

资源分标签资源和动态资源。标签资源定义在 SkyForm AIP 的配置文件 cb.yaml 里，而基本主机动态资源由 aip 服务自动从各主机自动获取或由 RESS（资源传感器）引入到系统中。

根据值的类型，资源被划分为：

- 数值型（number）：该类型资源的值为一个数值，比如可用的内存、主机 CPU 等。
- 字串型（text）：该类型资源的值为一个字符串，比如主机类型、主机状态等。
- 标签型（tag）：表示某些具体特性是否可用。

资源可以定义范围：主机绑定，集群共享，部分主机共享。

小技巧： 定制动态资源由 RESS 定义和提供。

2.7.3 资源传感器 RESS

资源传感器（RESS）是一个定期输出 YAML 资源说明的可运行程序，只要该程序按照一定的规则命名和输出，定制的资源就自动加入到 SkyForm AIP 中，可被作业定义，由调度器进行调度。

例如，RESS 通过输出以下内容增加了两个集群资源 netres2 和 lic：

通过命令，我们可以看到这两个资源的值：

SkyForm AIP 服务启动时，会自动检测在其 daemon 安装目录（如/opt/skyformai/sbin）里有以下文件名的可执行文件：

- ress.master: 会在第一调度器主机上执行，并读取其输出。所有集群级的资源应该由 ress.master 检测更新并传递给 SkyForm AIP。
- ress.host: 会在每个主机上启动，并读取其输出。每个主机特有的资源（如 FPGA）应该由 ress.host 检测更新并传递给 SkyForm AIP。
- ress.hostname: 会在主机名为 hostname 的主机上启动，并读取其输出。用户部分集群主机共享的资源，如一个硬件加速设备可同时被多个主机共享，就可在其中一台主机上增加 ress.hostname，并在输出项 locale 中说明被哪几台主机共享。

RESS 里定义的资源具有以下属性：

- 资源名 (resource)：作业可以用资源名来指定对此资源的需求。
- 资源描述 (description)：以描述资源和提示资源的使用。
- 类型 (type)：可分数值型（number），字串型（text），标签型（tag）。
- 数值升降 (direction)：描述数值型资源的多少属性，如可用内存是数值越大资源越多 (decrease)，而网络流量是数值越大资源越少 (increase)。
- 释放 (release)：在作业暂停时作业是否释放资源。如作业暂停时会释放 CPU 和网络流量，但不会释放内存。
- 指定型 (assign)：有些资源如网络带宽或内存在分配时没有指定的单位，而 GPU 和端口在分配时由调度器指定具体某个资源，以免各个作业间互相发生冲突。
- 任务资源 (slotresource)：即每个任务需要分配一份资源，如内存，GPU 等，而有些资源是作业级的，即不管一个作业有多少个任务，都只需分配一份资源，如存储容量。
- 资源值 (value)：当前资源的值。

- 范围 (locale)：节点类资源（如可用内存，GPU）或是集群资源，如网络流量和存储容量是整个集群共享的像 FPGA、GPU 等人工智能加速硬件就可以以资源的方式通过 RESS 加到 SkyForm AIP 中。

RESS 的例子：/opt/skyformai/sbin/ress.master

```
#!/bin/bash
while true; do
    echo "- resource: clksnd"
    echo "  description: The second value of the current clock"
    echo "  type: number"
    echo "  direction: increase"
    echo "  value:" `date +%S`
    echo "  locale: master node01"
    echo "---"
    sleep 10
done
```

以上的例子中，管理主机上的 cbls 会启动 ress.master 脚本，插入名为 clksnd 的数值型资源，这个资源只在 master 和 node01 两台主机上有效。

ress 脚本编写好后，可以直接运行，确保输出的 YAML 格式正确，然后再放到/opt/skyformai/sbin 目录中，并重启相应主机的 cbls。以上的例子，重启管理主机 master 上的 cbls:

```
cadmin lsrestart master
```

2.7.4 资源配额

SkyForm AIP 可对数值型资源设置配额，限制对资源的使用，防止某个用户或用户组过多占据资源。

系统里可以设置任意多的资源配额，每个配额为消费者限定每时每刻能使用的最多资源量。

消费者可以是以下单元的任意组合：

1. 队列群或单个队列
2. 用户/用户组群或单个用户/用户组
3. 主机/主机群或单个主机/主机群
4. 项目名
5. 应用名

可设资源配额为：

1. 作业槽总量
2. 作业数总量
3. 数值型资源（如 GPU 数），包括 RESS 插入的数值型资源的总量。

小技巧： 关于队列、用户/用户组、和主机/主机组的设置，请参考 * 配置 *。

资源配额设置在系统配置文件 cb.yaml 的底部。

例子 1：

```
- name: limit1
  queues: medium low
  users: all ~u001
  slots: 130
```

定义在队列 medium 和 low 上，除了用户 u001 以外的所有用户的所有作业操数为 130。

例子 2：

```
- name: limit2
  per_queue: medium low
  per_host: all
  jobs: 2
```

定义对 medium 和 low，每个队列里，在每个主机上的作业数为 2。

例子 3：

```
- name: limit3
  per_user: all
  resources: [mem,5000] [gpu, 2]
```

定义每个用户内存配额为 5GB，GPU 数为 2。

配额还可定义有效的时间窗。例子：

```
slots: '[2 1:8:0-5:18:0] [5 5:18:01-1:7:59]'
```

以上的例子定义在周一的 8：00 到周五的 18：00，作业槽配额为 2，在周五 18：01 到周一的 7:59，作业槽配额为 5。

```
resources: '[mem,500000 1:8:0-5:18:0] [gpu,5 9:00-18:00]'
```

以上的例子定义在周一的 8：00 到周五的 18：00，内存的配额为 500GB，每天 9 点到 18 点，GPU 的配额数为 5 个。

2.7.5 作业资源需求

作业提交时可以定义作业所需的资源来帮助调度器为作业安排最优的运行资源，作业提交的资源需求可以用 csub -R “字串” 参数定义。字串中可以含有以下字节：

- 选择部分（select）。选择部分指定选择执行主机的标准：select[…]
- 排序部分（order）。排序部分说明对符合选择标准的主机进行排序: order[…]
- 资源使用部分（usage）。资源使用部分确定作业的预期资源消耗: rusage[…]
- 作业跨越部分（span）。作业跨越部分指示并行作业是否应该跨越多个主机: span[…]
- 资源组替代选择（same）。多台主机为并行作业选择的资源应具有相同的资源值: same[…]

例子 1：

```
csub -R "type==x86_64Windows order[ut] rusage[mem=100]" myjob
```

选择 Windows 主机，按 CPU 利用率排序，作业使用（预留）100MB 内存。

例子 2：

```
csub -n 16 -R "span[hosts=1] rusage[gpu=8]" myjob
```

作业需要 16 个 CPU，16 个 CPU 必须在同一台主机上，即整个作业需要 8 个 GPU。

例子 3：

```
csub -n 64 -R "same[model] myjob
```

作业需要 64 个 CPU 核，64 个核必须分布在相同的 CPU 型号的主机上（集群由不同型号 CPU 的服务组成）

集群中的可用资源可以用命令 cinfo 列出。

2.7.6 多节异构作业资源需求

对于类似 TensorFlow 的深度学习应用，一个作业由多个部分组成。提交此类作业时，可以使用多节资源需求。多节资源需求的语法为：n1{资源需求} n2{资源需求}…

其中 n1，n2 为 CPU 核数，如：

```
4{mem>10} 5{rusage[gpu=1]}
```

表示作业一共需要 9 个核，前面 4 核需要在内存大于 10MB 的主机上，后面 5 核每核需要 1 个 GPU。

更多例子，请查阅*csub*。

2.8 队列

队列是 AIP 作业的载体。用户提交作业到队列中，由 AIP 的调度器根据作业资源需求、集群中可用的资源、和*cb.yaml* 里配置的调度策略调度和分发队列里的作业到合适的计算主机上。

AIP 里的队列由调度器管理。缺省队列与集群挂钩，这样队列不会受某个主机的状态而影响作业提交。

小技巧： 修改队列（*cb.yaml* 中 qeueus 部分的参数）后，可以重配置调度器使其生效:

```
csadmin reconfig
```

2.8.1 队列配置的例子

AIP 中可以根据不同的需求配置多个队列，队列配置里和很多参数。常用的参数有：用户 (users)、主机 (hosts)、和缺省资源需求 (resspec)。

一般不同的应用需要不同的资源，如一些应用需要大内存，一些需要 GPU 等。一般的队列设置一应用类别来设置不同的队列。每一类应用，即每个队列，根据其资源需求设置队列所用的主机。同时可以根据用户使用应用的权限配置可用该队列的用户。

例子 1：按应用分队列

计算中心给 3 类应用：

1. MPI 类的 HPC 应用，如计算流体力学 OpenForm 这类应用需要使用大量的 CPU，而对内存的需求不太大。
2. 图形设计仿真应用，如 FreeCAD、用户图形桌面这类应用需要较大的内存和用于图形加速的 GPU。
3. AI 模型训练和一些可以用 GPU 加速的应用这类应用需要已使用 GPU 为主。

cb.yaml（企业版）配置:

```
# 配置3个主机组，每一个针对一种应用选用转为这类应用准备的服务器
# 如果要调整主机的用途，只要修改这部分配置即可
hostgroups:
- name: compute           # 运行MPI作业的CPUCPU计算类主机
  members: c[01-10]       # 10 台主机，名为c01, c02, ... c10

- name: gcompute          # 运行AI所用的GPU主机
  members: g[01-04]       # 4 台GPU主机：g01, g02, g03, g04

- name: desktop           # 用户交互任务的主机
  members: d01 d02 d03    # 3 台主机

# 配置使用不同应用的用户组
usergroups:
- name: cae               # 使用LDAP中的用户组cae
  members: "@system"      # @system表示组员由操作系统决定，AIP
                          # 每小时会与系统同步一次组员信息，如果
                          # LDAP里用户组成员修改了，1小时内会
                          # 自动同步到AIP中
  usermaxslots: 64        # 每个用户最多用64个作业槽（CPU核），
                          # 防止一个用户把这个作业槽都占了

- name: ml                # 可以跑AI训练的用户
  members: user01 user02 user03
  administrator: user01   # 这个用户组有一个管理员，可以控制
                          # 组内其他用户的作业

# 配置队列
queues:
- name: gui               # 所有用户都可以使用
  description: 运行交互式桌面或图形应用
  priority: 3             # 队列优先级，由于队列使用专属的主机
                          # 优先级数字不重要
  hosts: desktop          # 使用desktop主机组的主机
  usermaxslots: 2         # 每个用户最多用2个作业槽
  memlimit: percore       # 按照CPU核数平均分配内存，并限制内存
                          # 使用，超过内存限制的作业会被调度器
                          # 杀掉

- name: comp
  description: 运行并行作业的CPU队列
  priority: 3
  hosts: compute
  users: cae
  fairshare: "[default, 1]"  # 公平分享策略，每个用户等同的份额
```

(下页继续)

(续上页)

```
- name: gpu
  description: GPU计算为主的应用
  priority: 3
  hosts: gcompute          # 使用GPU主机组
  users: ml                # 只限ml组的用户使用
  resspec: rusage[gpu=0.25]  # 按作业槽平均分配GPU，这里每台主机
                           # 上有8个GPU，32个CPU核（作业槽），每个
                           # 作业槽预留1/4个GPU

# 调度器通用配置
general:
  default_queue: gui       # 如果提交作业时不指定队列，用缺省的队列gui
  cgroup: acct gpu         # 使用Linux cgroup控制作业GPU的使用
  mailprog: ":"            # 作业结束后不自动送邮件

# 集群配置
cluster:
  name: aip
  administrators:
  - cadmin
  hosts:
  - name: mgt[01-02]       # 两台管理节点，不运行作业
    maxslots: 0
  - name: c[01-10]
  - name: g[01-04]
  - name: d[01-03]
  - name: default
    maxslots: cpu
 enterprise: yes
```

例子 2：优先级抢占

EDA 公司的集群提供 VNC 桌面和 EDA 计算资源。有一个小组的工程师不是需要跑一些高优先级的作业。高优先级作业需要抢占普通 EDA 计算作业所用的资源。

cb.yaml（企业版）配置:

```
# 配置两个主机组
hostgroups:
- name: desktop              # 用于VNC桌面的主机组
  members: d01 d02 d03

- name: compute              # EDA计算主机
  members: all ~desktop      # 集群中除了VNC桌面主机外的其他主机

# 配置3个队列
queues:
- name: medium
  description: 通用计算队列
  priority: 3
  fairshare: "[default,1]"  # 每个用户有相同的份额，防止单一用户占据整个集群
  hosts: compute             # EDA计算主机组
  rerunonhostfail: yes       # 如果某台主机出故障，自动重调度主机上跑的作业

- name: vnc
```

(下页继续)

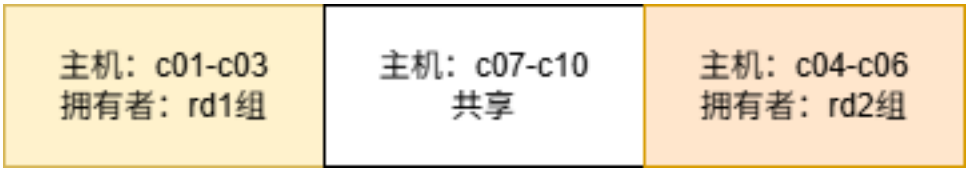
(续上页)

```
  description: 桌面队列
  priority: 3
  hosts: desktop
  usermaxslots: 2            # 每个用最多两个作业槽

- name: high
  description: 高优先级队列，可以抢占普通作业所占的资源
  priority: 10               # 优先级只要高于medium队列即可
  preemption: medium         # 如果集群里没有资源，杀掉并让medium队列中的
                             # 一部分作业，并让被杀掉的作业自动重排队
  users: user03 user04       # 只用用户user03和user04可以使用这个队列

# 调度器通用配置
general:
  default_queue: medium      # 一般作业都缺省提交到medium队列

# 集群配置
cluster:
  name: aip
  administrators:
  - cadmin
  hosts:
  - name: mgt01
    maxslots: 0
  - name: c[01-10]           # 由于集群较小，c01和c02作为管理节点
                             # 的备选主机（缺省hosts列表中前3台为master主备机）
  - name: d[01-03]
  - name: default
    define_ncpus: threads
    maxslots: cpu
  enterprise: yes
```

例子 3：主机拥有、共享、和抢占

两个小组 (rd1 和 rd2) 把自己拥有的服务器合并成一个集群，但每个小组在自己拥有的主机上都有优先级，并可以抢占其他用户在上面跑的作业。

公司另外有一批公共所有的主机也加到了集群中，这部分主机所有用户共享。



如上图所示，rd1 和 rd2 都有自己拥有的资源池 (c01-c03 和 c04-c06)，中间（c07-c10）有共享的资源池。

1. 这 3 个资源池共属一个集群。
2. 每个组里的资源池中若自己组的人不用，可共享给其他组的人用。
3. 自己组里的任务在自己的资源池中可以抢占其他组里任务的资源，抢占只限于自己的组的资源池中。

cb.yaml（企业版）配置:

```
# 用户组
usergroups:
```

(下页继续)

(续上页)

```
- name: rd1
  members: "@system"       # LDAP中rd1组
- name: rd1
  members: "@system"       # LDAP中rd2组

# 配置3个队列
queues:
- name: medium
  description: 普通队列，队列可以使用集群的所有主机
  priority: 3

- name: qrd1
  description: rd1用的队列
  priority: 10
  hosts: c[01-03]+ others    # 优先使用自己拥有的主机
  users: rd1                 # 只有rd1组可以用这个队列
  ownership_hosts: c[01-03]  # 拥有主机c01 c02 c03
  preemption: medium qrd2    # 可以在拥有的机器上抢占其他队列中的作业

- name: qrd2
  description: rd2用的队列
  priority: 10
  hosts: c[04-06]+ others    # 优先使用自己拥有的主机
  users: rd2                 # 只有rd2组可以用这个队列
  ownership_hosts: c[04-06]  # 拥有主机c04 c05 c06
  preemption: medium qrd1    # 可以在拥有的机器上抢占其他队列中的作业

# 调度器通用配置
general:
  default_queue: qrd1 qrd2 medium  # 用户提交作业而不指明队列时，AIP按这里定义的
                                   # 的顺序查找第一个用户可以使用的队列。
  mailprog: ":"                    # 作业结束后不自动送邮件
  user_view_alljobs: y             # 允许普通用户查看所有用户的作业信息

# 集群配置
cluster:
 name: aip
 administrators:
 - cadmin
 hosts:
 - name: mgt[01-02]    # 两台管理节点，不运行作业
   maxslots: 0
 - name: c[01-10]
 - name: default
   maxslots: cpu
 enterprise: yes
```

缺省抢占的动作为 requeue，即杀掉作业然后放回对接中重新调度。若抢占为暂停进程，在 general 里增加 preemption_suspend: y

2.8.2 调度主机顺序

利用队列参数 resspec，您可以控制调度时选择主机的顺序。

例子 1：作业调度到集群主机时，实现负载均衡。

AIP 缺省的主机选择顺序是按照 r15s:mem 排序，即先按主机负载 r15s 排序，若 r15s 一样，再按可用内存排序。这种排序是一个调度周期排一次，在同一个调度周期中，则是先把一台主机填满。在作业少的情况下（一个调度周期内所有等待作业都能被调度和分发），作业调度就不能实现真正的负载均衡。

在队列的参数中使用：**resspec: order[slots]**

这个参数让调度器在同一个调度周期中每个作业都对主机按其可用作业槽做一次排序，从而实现同调度周期内的负载均衡。

例子 2：尽量把作业往同一个主机上放，留出空间给大作业。比如尽量让 AI 训练任务用同一台主机上的 8 个 GPU。

这个例子是典型的“去碎片化”。可以使用队列的参数：**resspec: order[-slots]**

这个参数让调度器尽量用可用作业槽数少的机器，把作业槽数多的机器留给大作业。

类似可以用：**resspec: order[-gpu]** 来让作业尽量跑在可用 GPU 数少的机器上。

2.8.3 大作业和独占作业作业槽预留

大作业是指需要作业槽多的作业，这样的作业可以是单机，或者是跨机运行的。

由于调度器每次都去匹配队列中的作业资源需求与主机可用资源，小的作业比较容易找到可用的资源而往往先被调度，这样导致大作业长期得不到所需的资源而等待。

通过配置队列参数：**slotreservetime: N** (N 是保留的分钟数)，调度器在小作业结束后会为队列中的大作业保留空出的作业槽 N 分钟，从而增加大作业被调度的可能性。

如果过了保留时间还是没有足够的作业槽给大作业，所保留的作业槽会被释放，调度器进入新一轮的作业槽预留。

最佳时间是 slotreservetime 为集群中的平均作业运行时长。

独占作业是指不管作业使用多少资源，作业必须独占一台或多台主机。这类作业不会被调度到有作业运行的主机上，而一旦开始运行，所运行的主机上也不会再被调度其他作业。

队列中必须设置参数：**exclusive: yes**，用户才可以提交独占作业到这个队列中，提交的命令行为：“csub -x myjob”。

独占作业与大作业非常类似，在繁忙的集群中很难空出整台主机可以运行独占作业。队列参数 **slotreservetime** 也可以启动独占作业的作业槽预留。

2.8.4 作业运行时长控制

为了防止作业运行时应用异常但不退出，长期占用资源，队列中可以设置 runlimit 参数控制作业最长运行时长。超过运行时限的作业会被调度器杀掉。

备注： 在运行时限到达时，调度器向作业发送信号 SIGUSR2 (12)，如果作业不退出，5 分钟后或者 runaway_job_grace_period（*cb.yaml* ）后，调度器连续发送 SIGINT(2), SIGTERM(15), 和 SIGKILL(9), 这些信号间隔为 10 秒或者 job_terminate_interval （*cb.yaml* ），最后所有作业进程都会被杀掉，包括多机作业上所有属于该作业的进程。

参数的单位为分钟。

例子:

```
queues:
- name: medium
  runlimit: 1440     # 24 小时
```

备注: runlimit 参数可以定义两个值，一个缺省，一个最大。具体使用参考*cb.yaml* 。

2.8.5 作业内存使用控制

当主机上的可用内存耗尽时，主机会挂掉，这样一个超用内存的作业会影响同一主机上的其他作业。

内存控制有以下几种场景:

企业内部集群用户无法预测作业所需内存

用户无法为作业预留内存 (csub -R rusage[mem=xxx])，这种场景可以利用负载阈值的调度方法。

内存调度阈值可以定义在主机的配置里，或者定义在队列里。队列里定义的阈值是不能区分不同主机的不同阈值的。

队列中定义阈值的队列参数是：**thresholds**。例子:

```
queues:
- name: medium
  thresholds:
  - "mem 1G/0.5G"
```

1. 第一个值表示：当主机内存小于这个值后，主机不再接受新的作业，不管主机上是否还有空的作业槽。
2. 第二个值表示：当主机内存小于这个值后，主机上的作业会被暂停，以防主机挂掉。

也可以之选配第一个值，因为暂停作业不会释放内存，从而导致作业处于停顿状态。主机可以通过使用交换区（SWAP）避免主机因内存而挂掉。

对外服务的计算中心严格控制作业内存使用

通过使用作业级的内存使用限制保证主机内存不会超载。超过内存限额的作业会被调度器杀掉。

推荐使用的参数为：**memlimit: percore**

这是根据作业所用核数来确定内存够使用限制。如主机总共 56 个核，内存总量为 1T，则每个核的内存限制为约 18GB。一个 16 核（作业槽）的作业的内存限制是：288GB。

具体每个核的内存限制会因主机的硬件配置不同而不同。如果过一个作业跨多个主机，而每台主机的硬件配置不同，调度器会按照作业运行的第一台主机的硬件配置计算相应的内存使用上限。

备注: 调度器每 15 秒检查一次作业内存使用。检查时如果过发现作业所有进程，包括多台主机上的远程进程使用内存的总核超过限制，调度器向作业发送信号 SIGINT(2), SIGTERM(15), 和 SIGKILL(9), 这些信号间隔为 10 秒或者 job_terminate_interval （*cb.yaml* ），最后所有作业进程都会被杀掉，包括多机作业上所有属于该作业的进程。

2.9 集群管理员命令

2.9.1 服务进程控制命令

集群管理员可以用命令 aip admin 来控制各个主机上的 AIP 服务，并通过重启主控制主机上的服务或调度器进程来使 AIP 服务使用新的配置。

```
[cadmin@ai0 tensorflow]$ aip admin
NAME:
   aip admin - Administrative tool to control SkyForm AI daemons

USAGE:
   aip admin command [command options] [arguments...]

COMMANDS:
     start          Start SkyForm AI daemons on the local host or specified host
     stop           Stop SkyForm AI daemons on the local host or specified host
     restart        Restart SkyForm AI daemons on the local host or specified host
     reconfigsched  Update SkyForm AI scheduler related configurations
     reconfig       Update configuration and restart SkyForm AI daemons on all hosts in the clus
ter

OPTIONS:
   --help, -h  show help
```

- aip admin start <host> | all：在 host 主机或所有主机上启动 AIP 服务
- aip admin stop <host> | all：在 host 主机或所有主机上停止 AIP 服务
- aip admin restart <host> | all：在 host 主机或所有主机上重启 AIP 服务
- aip admin reconfig: 与 admin restart all 功能一样。当在 cb.yaml 文件中修改任何 cluster 里的参数，因为有可能影响所有主机，则需要运行该命令。

  备注： 重启所有主机上的 AIP 服务会造成集群数分钟的中断，所以需要谨慎操作。

- aip admin reconfig: 当在 cb.yaml 文件中修改 queues 或 general 参数时，只对调度器有影响，可以用该命令来只重启调度器，减少对整个集群的影响。

2.9.2 资源使用量统计

资源使用的统计可以用作计费。AIP 管理员可以使用以下命令得到资源使用的统计：

cacct

使用参数如下。具体详细说明请参考*cacct* 。

-d: 统计正常结束的作业

-e: 统计异常结束的作业

-C: 定义作业结束的时间段

-m: 定义作业运行的主机，多个主机用空格隔开，并使用引号把所有主机括起来

-P: 定义一个或多个项目名

-q：定义一个或多个队列名

-u: 定义一个或多用户或用户组名

-j：定义一个或多个作业号

```
SUMMARY:     (time unit: second)
 Total number of done jobs:     26
 Total number of exited jobs:   13
 Total number of run jobs:      39
 Total CPU time consumed:       5.17
   Average CPU time consumed:   0.13
   Maximum CPU time of a job:   6.00
   Minimum CPU time of a job:   0.00
 Total memory used (MB):        1478.988
   Average memory used (MB):    37.923
   Maximum memory used (MB):    624.395
   Minimum memory used (MB):    0.000
 Total job run time:            6157855
   Average job run time:        157893.7
   Maximum job run time:        3037768
   Minimum job run time:        0
 Total job wait time in queues: 3704.0
   Average wait time in queues: 95.0
   Maximum wait time in queues: 2380
   Minimum wait time in queues: 0
 Average turnaround time:       119024.1
   Maximum turnaround time:     1519110
   Minimum turnaround time:     0
 Average hog factor of a job:   0.02 (cpu time / turnaround time)
   Maximum hog factor of a job: 0.18
   Minimum hog factor of a job: 0.00
 Total throughput:              0.03 (jobs/hour) during 1469.50 hours

Time window: Oct 16 17:34 - Dec 16 22:03
```

自动生成计费用的用量和月账单

用量计费和账单生成修改配置文件/opt/skyformai/etc/cbcrond.yaml。修改后重启 master 上的服务，即可生效。参考*cbcrond.yaml*

2.9.3 动态修改配置（不重启服务或 daemon）

在不启动 AIP 服务和调度器的情况下，可使用以下命令修改调度器配置：

1. “aip p u”：修改调度器参数（cb.yaml 中的 general 参数、power 参数、scale 参数）。动态修改后的参数自动存于/opt/skyformai/etc/params.yaml（通用参数），/opt/skyformai/etc/power.yaml（节电调度参数），和/opt/skyformai/etc/scale.yaml（弹性伸缩调度策略参数）里，cb.yaml 中相应数据将视为无效。
2. “aip q c | d | u”：新增、删除、或修改部分队列参数。动态修改后的参数自动存于/opt/skyformai/etc/queue.yaml 里，cb.yaml 中 queues 中的参数将视为无效。
3. “aip ug c | d | a | am | dm | gl | ul”：新增、删除用户组，修改用户组管理员，增加、减少用户组员，修改用户组作业槽上限，每个用户作业槽上限。动态修改后的参数自动存于/opt/skyformai/etc/ug.yaml 里，cb.yaml 中 usergroups 中的参数将视为无效。
4. “aip hg c | d | am | dm”：新增、删除主机组，增加、减少主机组成员。动态修改后的参数自动存于/opt/skyformai/etc/hg.yaml 里，cb.yaml 中 hostgroups 中的参数将视为无效。

动态修改参数后，若需修改其他参数，可以直接在 cb.yaml 中修改，然后重启 AIP 服务。若要进一步手工编辑以上经过动态修改过的参数，则需要编辑相应的 yaml 文件（queue.yaml, params.yaml, power.yaml, scale.yaml, ug.yaml. hg.yaml 等）。删除相应的动态参数文件后若重启调度器（cbsched），则回到 cb.yaml 中的参数。调度器启动时先读 cb.yaml，然后读取其他.yaml 文件覆盖 cb.yaml 中的参数。

2.10 安装共享数据分析和机器学习工具

2.10.1 准备

选择共享文件系统目录。我们使用/share/apps 作为例子，以后所有的工具都将安装在这个目录下

```
mkdir -p /share/apps
```

安装任意版本的 anaconda，这个版本只是为了把所有工具安装在以上目录中，不影响最终用户的 conda 版本。

```
yum -y install epel-release
yum -y install conda
```

2.10.2 安装 Python

建立 Anaconda Python3.9 环境，并安装在这个环境中的 Anaconda

```
conda create -p /share/apps/python python=3.9
conda init bash
```

退出当前终端，再进入，然后运行：

```
conda activate /share/apps/python
pip3 install -U pip
conda install conda
```

2.10.3 安装机器学习库

```
pip3 install pandas numpy scipy matplotlib seaborn scikit-learn xgboost
```

- 安装 TensorFlow 和 TensorBoard

```
pip3 install tensorflow
pip3 install tensorboard
```

- 安装 PyTorch

```
pip3 install torch
```

2.10.4 安装 Jupyter

安装 Jupyter Lab

```
pip3 install jupyterlab jupyterlab-language-pack-zh-CN
```

2.10.5 安装 R

安装 R 4.3

```
conda install -c conda-forge R
```

2.10.6 安装 Julia

- 从网页 https://julialang.org/downloads/ 下载 Generic Linux on x86 - 64 bit (glibc)
- 解压，例子

```
tar xfz julia-1.9.3-linux-86+64.tar.gz -C /share/apps
```

2.10.7 安装 Spark

访问Spark 下载网页，下载 “Pre-built for Apache Hadoop”，如：spark-3.5.0-bin-hadoop3.tgz

解压到共享目录:

```
tar xfz spark-3.5.0-bin-hadoop3.tgz -C /share/apps
```

2.10.8 安装 Jupyter 内核

安装 Python 2 内核

1. 安装 Python 2 在安装前，确保退出原有的 Conda 环境

```
conda deactivate
```

建立 Python 2 环境

```
conda create -p /share/apps/python2 python=2
```

2. 安装 Python 2 内核

```
/share/apps/python2/bin/pip install ipykernel
/share/apps/python2/bin/python2 -m ipykernel install
```

3. 迁移内核到 Conda Python 3 和 Jupyter 的环境中

```
mv /usr/local/share/jupyter/kernels/python2 /share/apps/python/share/jupyter/kernels/
```

安装 R 内核

- 安装

```
conda activate /share/apps/python
conda install -c r r-irkernel
```

- 修改 R 运行命令行，修改/share/apps/python/share/jupyter/kernels/ir/kernel.json 的第一行，加入 R 运行路径:

```
{"argv": ["/share/apps/python/lib/R/bin/R", "--slave", "-e", "IRkernel::main()", "--
↪args", "{connection_file}"],
 "display_name":"R",
 "language":"R"
}
```

- 增加 R 运行设置，在/share/apps/python/lib/R/etc 中添加文件 Rprofile:

```
options(bitmapType='cario')
options(jupyter.plot_mimetypes="image/svg+xml")
```

确保运行 Jupyter 的主机上安装了 X11 的库（yum group install "Server with GUI"）

安装 Scala 内核 (Spark)

- 进入 Python3 环境

```
conda activate /share/apps/pythnon
```

- 安装软件

```
pip3 install spylon_kernel
python -m spylon_kernel install
conda install openjdk
pip3 install pyspark
```

- 迁移内核到 Conda Python 3 和 Jupyter 的环境中

```
mv /usr/local/share/jupyter/kernels/spylon-kernel /share/apps/python/share/jupyter/
↪kernels/
```

- 在内核定义文件中增加 Spark 位置，编辑/share/apps/python/share/jupyter/kernels/spylon-kernel/kernel.json，在 env 里增加 SPARK_HOME 和 JAVA_HOME。例子:

```
{"argv": ["/share/apps/python/bin/python", "-m", "spylon_kernel", "-f", "{connection_
↪file}"],
"display_name": "spylon-kernel","env": {"SPARK_HOME":"/share/apps/spark-3.5.0-bin-
↪hadoop3", "JAVA_HOME":"/share/apps/python", "PYTHONUNBUFFERED": "1", "SPARK_SUBMIT_
↪OPTS": "-Dscala.usejavacp=true"}, "language": "scala", "name": "spylon-kernel"}
```

安装 Julia 内核

由于 Julia 的不支持系统级与用户级运行环境的融合（如 Python 和 R 都可以支持系统与用户级的融合），安装 Julia 内核必须有需要使用的用户自行完成。详见 Jupyter 使用。

2.10.9 安装 RStudio

- 下载网页：https://posit.co/download/rstudio-server/
- 选择 OS 版本，拷贝下载命令，如

```
wget https://download2.rstudio.org/server/rhel8/x86_64/rstudio-server-rhel-2023.09.1-
↪494-x86_64.rpm
```

- 展开 rpm 包

```
rpm2cpio rstudio-server-rhel-2023.09.1-494-x86_64.rpm | cpio -idmv
```

- 把文件移到共享文件夹中

```
mv usr/lib/rstudio-server /share/apps
```

2.10.10 安装 Singularity

在 Linux 上安装 Singularity

```
yum install -y epel-release
yum install -y singularity-ce
```

下载集群中 Linux 同版本的镜像

```
singularity pull /share/apps/containers/centos7.sif docker://centos:7
singularity pull /share/apps/containers/rocky8.sif docker://rockylinux:8
```

2.10.11 安装 Git

常规操作系统中 Git 为 v1 版，很多开发者需要使用 Git v2。我们使用 conda 安装 Git v2

```
conda activate /share/apps/python
conda install git
```

2.11 基于 License 的调度

AIP 通过使用外插的 RESS 机制（参考*ress* ）把可用的应用浮动许可证数量变成 AIP 的资源，作业提交时请求该资源。如果没有可用的许可证，作业不会被调度。这样避免作业因无可用应用许可证而失败。

以下的例子通过 ress.master 把 Siemens StarCCM+ 的 ccmppower 许可证插入到 AIP 中。

1. 编写名为 ress.master 的脚本:

```
#!/bin/bash
# 这个 ress 例子用lmstat命令获取Siemens StarCCM+ license的使用情况，
# 计算出可用的feature ccmppower的数量。
# 通过定期输出（每10秒）资源配置名为star_ccmppower的YAML格式把可用
# 的ccmppower数量插入AIP调度器

STAR_LM_BIN_PATH="/share/apps/Siemens/14.06.012-R8/FLEXlm/11_14_0_2/bin"
STAR_LIC_SERVER_1="1999@mgt01"
STAR_LIC_FEATURE_1="ccmppower"
STAR_LIC_RES_NAME_1="star_ccmppower"
STAR_LIC_RES_VALUE_1="0"

OLD_IFS=$IFS

while true; do

  IFS=$'\n';
  STR_VAR1=(`${STAR_LM_BIN_PATH}/lmstat -c ${STAR_LIC_SERVER_1} -a`)

  for i in ${STR_VAR1[@]};
  do
     echo $i | grep ${STAR_LIC_FEATURE_1} > /dev/null 2>&1;
     if [ $? == 0 ]; then
        LIC_TOTAL=`echo $i | awk '{print $6}'`;
        LIC_USED=`echo $i | awk '{print $11}'`;
        STAR_LIC_RES_VALUE_1=$((LIC_TOTAL-LIC_USED))
        break;
     fi
  done

  IFS=$OLD_IFS

  echo "- resource: ${STAR_LIC_RES_NAME_1}"
  echo "  description: ${STAR_LIC_RES_NAME_1}, license feature resource of ${STAR_
↪LIC_FEATURE_1} "
  echo "  type: number"
  echo "  direction: decrease"
  echo "  release: no"
  echo "  assign: no"
  echo "  slotresource: no"
  echo "  value: ${STAR_LIC_RES_VALUE_1}"
  echo "  locale: all"
  echo "---"

  sleep 10
done
```

2. 编写好脚本后现在 shell 下运行测试。脚本不停地循环，每 10 秒输出类似以下的结果（连续）：

```
./ress.master
- resource: star_ccmppower
  description: star_ccmppower, license feature resource of star_ccmppower
  type: number
  direction: decrease"
  release: no"
  assign: no"
  slotresource: no"
  value: 15
  locale: all"
---
- resource: star_ccmppower
  description: star_ccmppower, license feature resource of star_ccmppower
^C
```

Ctrl-C 中断运行。

3. 把脚本放到/opt/skyformai/sbin 中。注意脚本名必须是 ress.master, root 必须可执行。
4. 不用重启 AIP，等 10-15 秒后用命令 cload -s 查看:

```
cload -s
RESOURCE                                VALUE       LOCATION
star_ccmppower                          15          dev node[01-10]
```

5. 确认调度器里也有相应的值:

```
chosts -s
RESOURCE                 USABLE       RESERVED       LOCALE
star_ccmppower               15            0.0       dev node[01-10]
```

6. 提交作业，用-R 预留作业所需的 star_ccmppower 资源（详见*csub* 里的-R 参数）:

```
csub -n 16 -R "rusage[star_ccmppower=1:duration=1]" mystarccmp
```

上面 rusage 中的 duration 是个关键的参数。这个作业先在调度器中预留 1 个 star_ccmppower 资源（即 ccmppower 许可），等作业调度开始运行后，应用到许可证服务器上拿走一个 ccmppower 许可，这样 ress.master 获的值就会少一个，所以在应用拿走 license 后，调度器里预留的值需要释放。duration 的单位是分钟，指资源预留时间，时间到后会释放。duration 的值一定大于应用启动到从许可证服务器获取许可的时间，但又不能太长，否则造成许可浪费（调度器过多预留资源）。

2.12 监控仪表盘

SkyForm AIP 使用 Prometheus 和 Grafana 作为监控系统。安装步骤如下：

2.12.1 安装 Prometheus

从 https://prometheus.io/download/下载最新版的 Prometheus 包，如 prometheus-3.3.1.linux-amd64.tar.gz。

解压后把可执行文件 prometheus 放到/usr/lib/prometheus 目录下。

```
tar xfz prometheus-3.3.1.linux-amd64.tar.gz
mkdir -p /usr/lib/prometheus
cp prometheus-3.3.1.linux-amd64/prometheus /usr/lib/prometheus
```

2.12.2 安装 Grafana

从 https://grafana.com/grafana/download 下载最新版的 Grafana RPM 包，如 grafana-enterprise-10.3.3-1.x86_64.rpm

安装 RPM 并允许服务自动启动。

```
yum localinstall -y grafana-enterprise-10.3.3-1.x86_64.rpm
systemctl enable grafana-server
```

2.12.3 安装 AIP 监控包

在 AIP 解压后的子目录 prometheus-aip-exporter 下运行 install 脚本。

```
cd prometheus-aip-exporter
./install
```

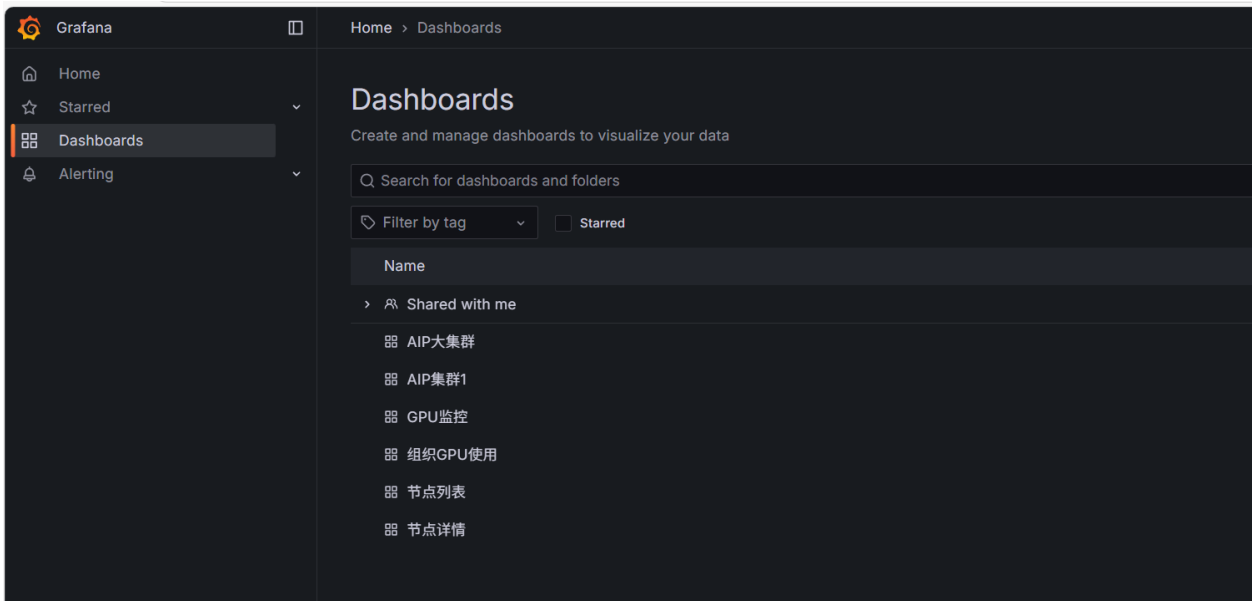
这个安装脚本会：

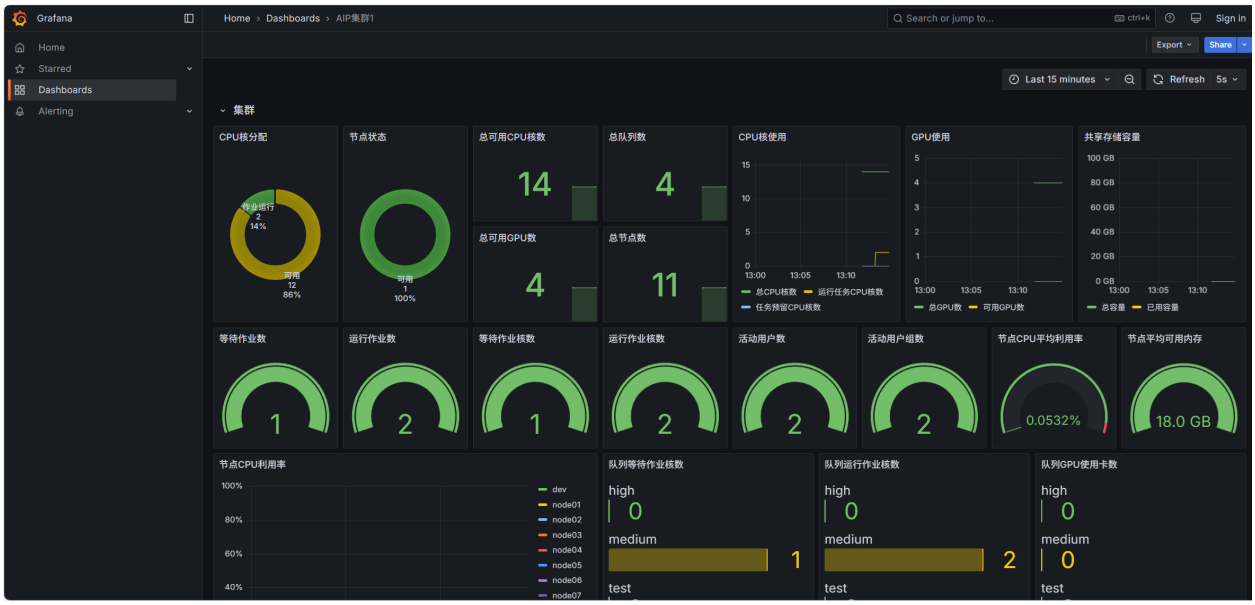
1. 在/etc/grafana/provisioning/dashboards 中设置仪表盘，这些仪表盘不能在 Grafana 中直接修改。
2. 安装并启动 aip-exporter 服务，该服务在 10610 端口上为 Prometheus 提供数据，数据通过调用 AIP 命令获得。
3. 配置 (/usr/lib/prometheus/prometheus.yml) 并启动 prometheus 服务。
4. 配置 (/etc/grafana/grafana.ini) 并启动 grafana 服务。

缺省 grafana 的管理员为 admin，密码为 admin

2.12.4 浏览器连接

浏览器连接到 http://安装机器的 IP_ 地址:3000，连接 Grafana 的仪表盘 dashboard。





2.12.5 修改已有的仪表盘

如果在 Grafana 中修改了产品带的仪表盘图，只能下载成 JSON 文件，然后替换/etc/grafana/provisioning/dashboards 中的 JSON 文件，重启 grafana-server 服务后修改的图可以生效。

2.12.6 aip-exporter 提供的数据

以下命令可以获得 aip-exporter 的指标数据:

```
curl localhost:16010/metrics
```

请参考：*aip-exporter*

2.13 配置 Web 门户

门户安装好后内置了一些基本配置，放置在文件/var/www/html/config.yaml 中。这些参数一般不需要调整。其他的配置可以通过 AIP 管理员（参考*cb.yaml* 里的 cluster: administrators）在门户上操作。

2.13.1 门户配置基本参数

安装完成后，管理员通过修改配置文件：/var/www/html/config.yaml 控制访问：

- **root_access**
  为了安全，缺省的安装允许 root 登录门户。若要禁止 root 用户登录，修改/var/www/html/config.yaml:

  ```
  root_access: no
  ```

- **admin_portal_for_users**
  如果禁止普通用户使用门户，而只允许管理员使用，可以把这个参数的值改成：no。
- **user_system**
  门户中可以管理 LDAP、AD、或本地（门户运行的主机）用户。可用的值为：“ldap”、“ad”、或“local”。

2.13.2 管理员门户管理配置

AIP 管理员登录门户后，左侧的菜单中的“管理”菜单有几个主要的功能：

- 数据管理配置
- 管理 LDAP 连接
- 管理 AIP 的一些配置

数据管理配置

数据管理配置可以控制用户的角色、可以通过门户浏览的服务器最上层目录、以及是否可以从服务器上下载文件。

缺省：

- 每个用户通过门户浏览的服务器最上层目录是用户的家目录
- 用户可以下载文件

点击“新增”按键，在弹窗里配置：

- 用户名：这个用户名是操作系统中合法的用户名。可以用保留词”default”表示缺省的配置
- 角色：可以选普通用户：“user”，或者数据管理员：“dataadmin”

- 路径：用户通过门户访问的最高路径。用“~”表示用户系统中配置的用户的家目录
- 路径权限：可选“readonly”**只能上传、不能下载**或“read-write”**可以上传下载**

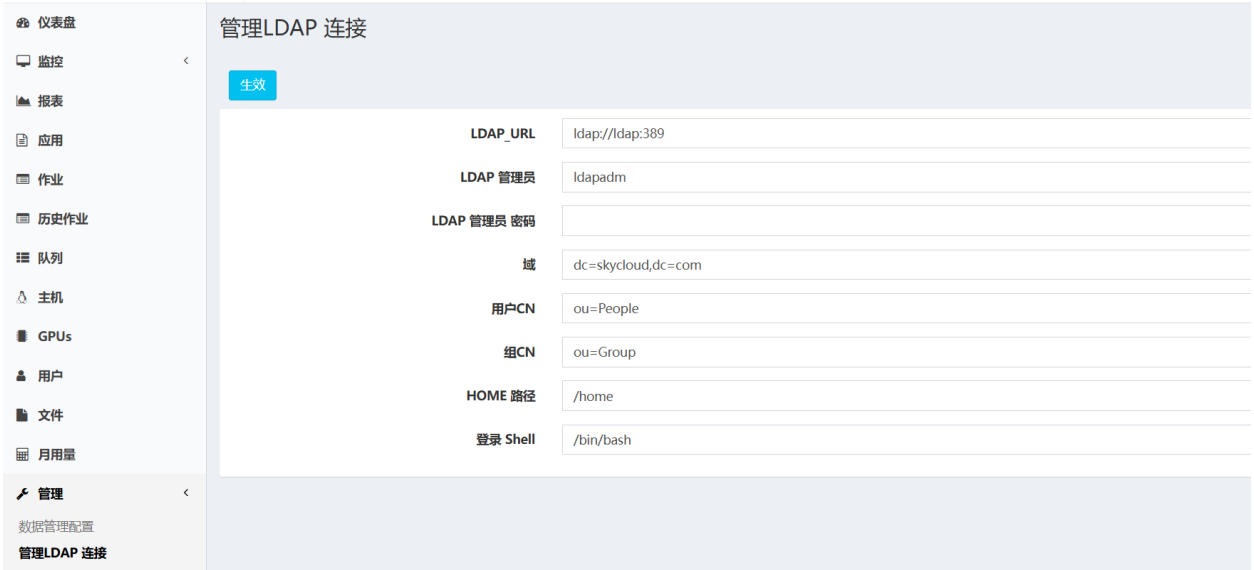
以上的例子中，缺省每个用户的门户文件访问最高路径为其家目录，只能上传文件，不能下载。

点击“新增”按键，增加一个“dataadmin”的角色的用户，把“路径”设成家目录的上一层，如/home，权限设成“read-write”。这个数据管理员可以为用户下载文件。

管理 LDAP 连接

门户可已利用 AIP 的工具有效管理单个 OU 下的 LDAP 用户和用户组。也可支持 AD 服务器的 LDAP 连接。

第一次使用点击“管理 LDAP 连接”，在页面上填入信息后，点击上方的“生效”键。



门户的后台会用这些参数运行 AIP 工具命令:

```
cbtool ldap config --url LDAP_URL --user LDAP管理员 --passwd LDAP管理员密码 --domain␣
↪域 \
--userCN 用户CN --groupCN 组CN --homeDir home路径 --loginShell 登录Shell --
↪isSambalEnabled=false \
--type ldap
```

这个命令在/opt/skyformai/etc 下生成文件 ldap.json，里面配置了以上这些访问 LDAP 的参数。后继的 LDAP 用户和用户组管理会使用这些参数由“aip lu”命令执行。

备注： 一旦/opt/skyformai/etc/ldap.json 文件生成后，门户中的“管理 LDAP 连接”的菜单就不显示了。如果需要重新配置，可以手工在把/opt/skyform/etc/ldap.json 文件更名或移除。再重新登录门户配置。

管理 LDAP 用户

配置 LDAP 连接后，就可以“新增”或删除 LDAP 用户了。新增用户时，可以上传 CSV 文件批量添加用户。CSV 文件的格式为：用户名（必要项），密码（必要项），家目录（可选项，缺省为 LDAP 连接配置中的家路径 +/用户名），登录 shell（可选）。

这个页面也可以重置用户密码。

管理 AIP 的用户组、主机组、或队列

管理用户组，必选项为组名和成员，其他可选。多个组的成员名用空格隔开。

新增主机组时，选择多个主机可以按 Ctrl 键选择。

新增队列的必选项为队列名和优先级，其他可选。多个用户、多台主机名间用空格隔开。

2.14 Web 门户应用集成

Web 门户中用户可以使用的应用，有模板生成。模板是一个 YAML 文件，里面含有提交页面的参数、应用图表、和作业提交命令等。

应用模板文件名为：应用名.yaml ，存放在/var/www/html/up/apps 里。用户登录后会自动扫描该目录中所有的.yaml 文件，在桌面上生成相应的应用图标。

2.14.1 YAML 模板文件制作步骤

应用模板的 YAML 文件分两类：

1. 需要用户输入集群参数的，如队列名，CPU 核数等。
2. 直接跳转到一个 URL 的，比如 php/terminal.php。

不管是哪一类，里面一个必要项，是应用图标参数：icon。

第一步：制作应用图标

应用的图标需要用图形工具制作一个 64x64 的 PNG 图形文件。运行 base64 命令获得图表的代码:

```
base64 python.png > Python程序.yaml
```

以上的例子是把 base64 转换后的图表放到名为“Python 程序”的模板文件中。

然后编辑这个 YAML 文件，插入第一行 `icon: "image/png;base64,` 如下:

```
icon: "image/png;base64,
iVBORw0KGgoAAAANSUhEUgAAAEAAAABGCAYAAAB8MJLDAAAAw3pUWHRSYXcgcHJvZmlsZSB0eXBl
IGV4aWYAAHjabVBbDgMhCPz3FD2CMujicdyuTXqDHr8obLO2nYThZUYg9NfzEW4DlDhw3qTUUqKC
```

在最后一行的尾部加上双引号，如下:

```
JgAAAABJRU5ErkJggg=="
```

这样模板中就有了应用图标的数据。

icon 参数是必须要的参数。

后面步骤中增加的所有参数都可放在 icon 参数的前面，以方便编辑 YAML 文件。

第二步：标识交互式应用

添加参数 gui。如果是交互式应用，如图形，web 等应用，则为 `gui： true` 。例子:

```
gui: false
```

gui 参数是必须要的参数。

备注： gui 参数会影响作业提交后的跳转。`gui： true` 作业提交后跳转到作业详情，否则跳转到作业列表。

第三步：添加集群参数

集群参数名字为：cluster_params。这个参数定义作业提交命令（csub）里的部分选项。

cluster_params 是可选参数。

cluster_params 中可以有以下的可选参数：

- **queue**

  取值为：`auto_list` （程序列出 AIP 中所有队列供选择）或者队列名列。

  例子 1:

  ```
  cluster_params:
    queue: auto_list
  ```

  例子 2:

  ```
  cluster_params:
    queue:
    - medium
    - high
  ```

  这个参数的值会转换成*csub* 的-q 参数。

- **distribution**

  取值为：`smp`，即单主机作业，或 `dmp`，即多主机作业。

  例子:

  ```
  cluster_params:
    distribution: smp
  ```

  参数值为 smp 时，程序转换成*csub* 的 `-R span[hosts=1]` ，值为 dmp 时，则不加 csub 的 span 参数，有调度决定分配单机或多机。

- **mincpu**

  让用户选择 CPU 核数：1、2、4、6、8、10、12、16、20、24、28、32、40、48、64、128、256。这个列表是固定的，若不能满足需求，请不要添加这个参数，而用 params（见下）来配置。

  参数的值为：`1,`。这个值是固定的，不能修改。

  程序把这个参数转换成*csub* 的-n 参数。

- **cwd**

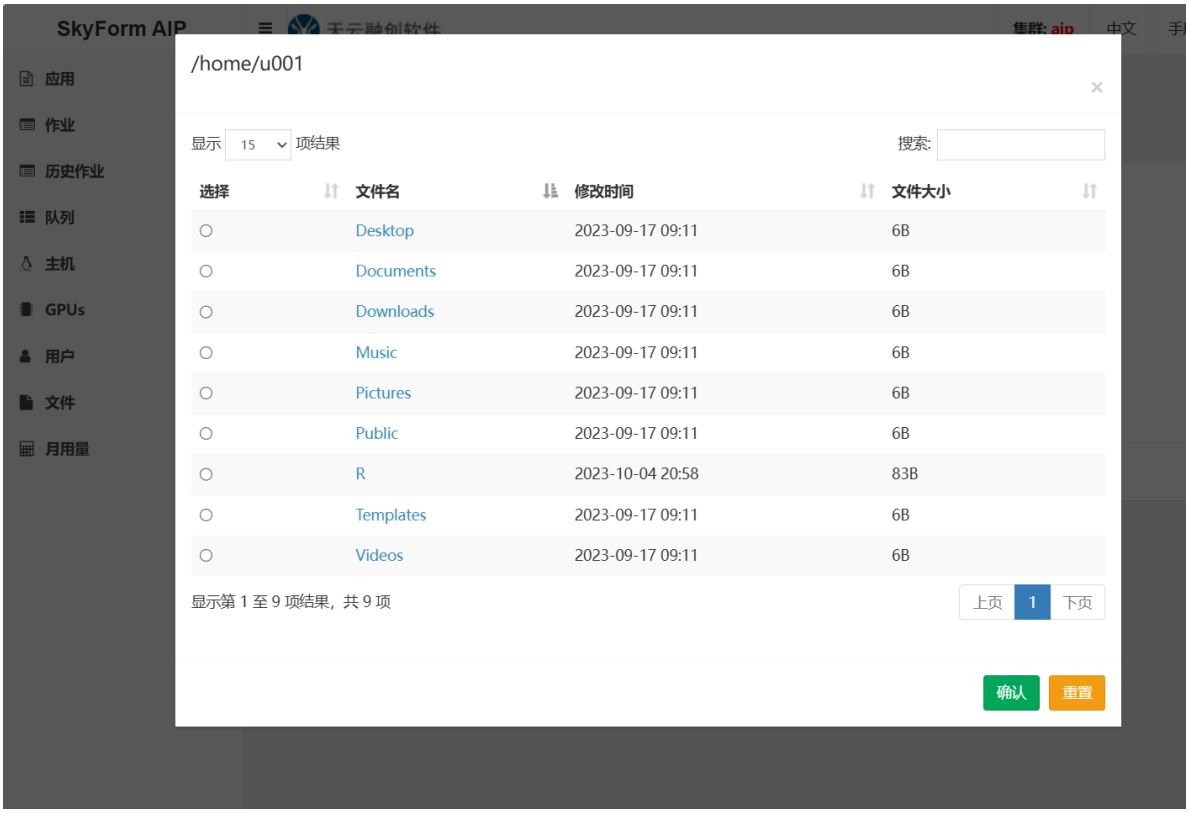
  让用户选择作业运行时的工作目录，取值为可选的最高层目录路径。值若为 `$HOME` 表示用户只能选择其 HOME 以下的任何目录，若让用户选择其他的目录，如项目存储区的一个目录，则把项目存储区的做高层路径作为参数的值。

  例子:

```
cluster_params:
  cwd: $HOME
```

  以上例子中，用户看到的应用参数如下图所示：

  点击工作路径参数值，用户可以浏览并选择目录：



程序把这个参数转换成*csub* 的-cwd 参数。

- **runlimit**

  让用户填入作业运行最长时间，单位为分钟。程序把这个参数转换成*csub* 的-W 参数。时限到后作业或被自动停止。

第三步 B：应用直接跳转 URL

如果应用不适用常规的作业提交，而是有其他程序来处理，或者跳转到另外的 URL，则可以放置参数 url，而不需要上面的第三步、和下面的第四、五步的操作。

备注： 指定 url 参数，务必指定 gui: true。

例子:

```
url: https://192.168.10.10:5601
```

url 里的内容会以 iframe 的方式显示在页面上。

备注： Web 门户使用 NGINX 的 HTTPS 提供服务的，如果过嵌入的页面不是 HTTPS，或者在其他域名下，则需要配置 NGINX 做转发，URL 为 NGINX 的转发 URL。嵌入受限于浏览器、所嵌网页和 iframe 等的规范。

第四步：应用参数

应用参数有两部分。第一部分为参数类别 params_category，第二部分是参数列表 params。

参数类别用于把参数在页面上按类别分开排列。params_category 的每一项有两个参数：name（类别名），和 id（类别号）。例子:

```
params_category:
- name: 应用参数
  id: 1
- name: 运行参数
  id: 2
```

每个 params 里的参数必须有以下的子参数：

- **category_id**

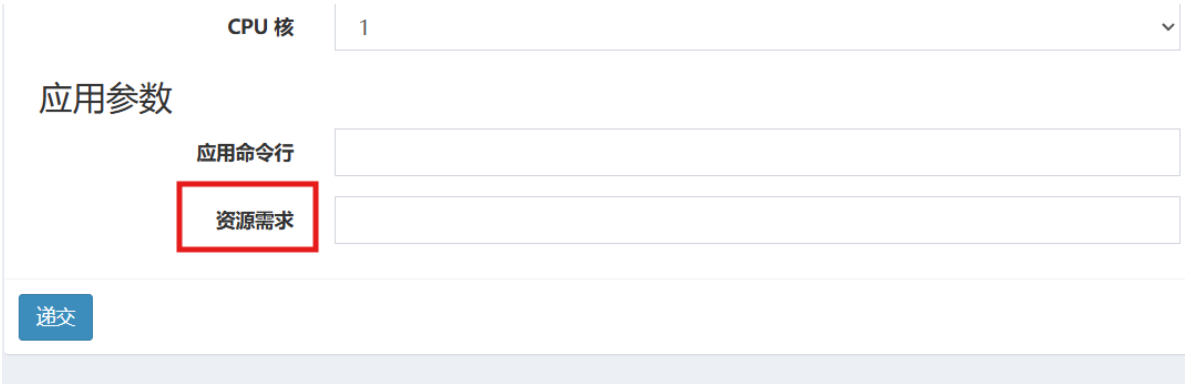
  参数的值对应于 params_category 中参数类别的 id 的值。例子：`category_id: 1` 。

- **id**

  参数号，这个号在第五步里会用到。例子：`id: 11` 。

- **filed_label**

  参数说明。这是任意的字串，会在应用页面上显示，如下：



- **filed_type**

  参数类别。可取的值为：

  - input: 输入文字。
  - select_single: 从多个选项中选一个，与参数 value_range 配合使用。
  - value_range: 多个可选值用逗号隔开，如 “1，4，8”。与 select_single 配合使用，这个例子中，用户可以选择 1，4，或 8 中的一个值。
  - check_box: 勾选，即该参数的值会被使用。与参数 value 配合使用，只有用户勾选这个参数后，其 value 的值会被命令行用上。
  - file-remote: 让用户选择服务器上的文件。用户点击后会弹出服务器文件浏览窗供用户选择。与参数 value 配合使用。

    value 定义最高路径，用户只能选择该路径以下（包含子目录）的文件。若最高目录为用户的 HOME，则 `value: $HOME` 。

    如果让用户选择目录，而不是文件，则需要增加参数 `value_range: /dir/*` 。
  - value: 缺省的参数值。如果过与 file-remote 配合使用，则为最高可浏览的目录。

  - app_cli: 定义参数怎样放到 csub 命令行中。有两个选项：“arg” 或 “var”。
    * arg：把转换后的参数值（见下面的 app_cli_convert）放到命令行的最后。多个参数值按序排放。
    * var：命令行中使用 “@ 参数号” 来指定参数值在命令行中的位置。

    详见第五步里的例子。
  - app_cli_convert: 对用户输入值的转换。转换可以使用 bash 命令行语法。用保留词 “$value” 代替用户输入的参数值。

备注： 参数 value_type 已经不再使用，可以忽略。

- **required**

  是否必填参数。若值为 “true”，表示若用户不填，作业不能提交。

第五步：命令行

下面两个参数配合使用。

- **command_path**

  作业提交命令，一般为 csub、vncsub、或 dcvsub。

- **common_params**

  作业提交参数。这里可以应用上面参数的值。

  如果参数的 app_cli 是 arg，则无需在此引用，程序会自动把参数的值添加到命令行尾部。

  如果参数的 app_cli 是 var，则可以用 “@ 参数号” 来指定参数值在命令行中的位置。如: `common_params: -o %J.out @12`，这里 @12 指把参数号为 12 的经过转换后的用户输入的值填放在命令行的这个位置。

警告： 如果过是作业提交参数，则必须定义为 var，因为这些参数不能放到作业应用的命令行后面。

例子 1:

```
params_category:
- name: 应用参数                          #␣
↪用户点击应用图标后的页面中会有一个名为 “应用参数” 的区列出下面定义的参数
  id: 1                                  # 参数类别号

params:
- category_id: 1                         # 这个参数放到 “应用参数” 区里
  id: 11                                 # 参数号为11
  filed_label: 应用命令行                 # 参数显示的标签
  filed_type: input                      # 参数值由用户自由填入
  value_type: 字串行                      # 忽略
  required: true                         # 必填参数
  app_cli: arg                           # 参数值放到命令行最后
  app_cli_convert: $value                # 直接使用用户输入的参数值，不做任何转换

- category_id: 1                         # 这个参数放到 “应用参数” 区里
  id: 12                                 # 参数号为12
  filed_label: 工作目录                   # 参数显示的标签
```

(下页继续)

(续上页)

```
  filed_type: file-remote           # 用户可以浏览服务器文件系统
  value_range: /dir/*               # 用户远程浏览服务器只能选择目录
  value_type: 字串行                # 忽略
  value: $HOME                      # 从用户HOME开始浏览
  required: true                    # 必填参数
  app_cli: var                      # 放到命令行中的某个位置
  app_cli_convert: -cwd $value      # 插入命令时的值为 "-cwd 用户输入的值"

command_path: csub                  # 用csub提交作业
common_params: -o %J.out @12        # 最后的提交命令行为:
                                    # csub -o %J.out -cwd <工作目录的值>
↪<应用命令行的值>
```

以上例子在页面上的效果如下图所示。

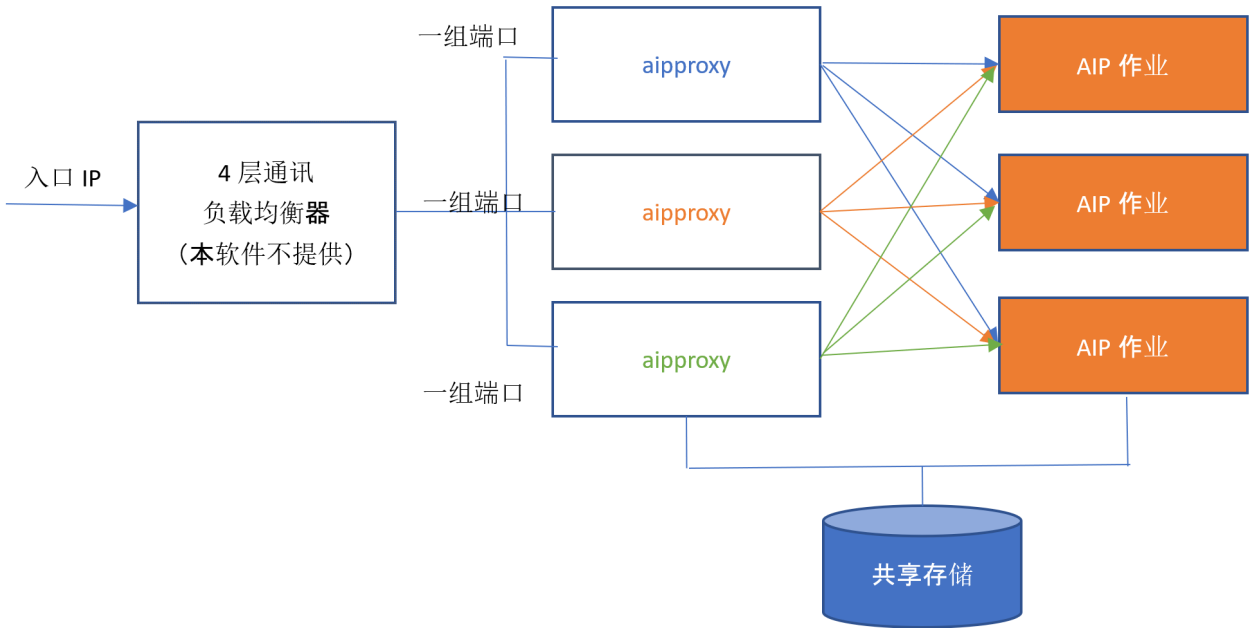
小技巧: 在开发应用模板式，可参考已安装的/var/www/html/up/apps 中的 yaml 文件，并对照页面上的效果。

2.15 TCP/IP 代理服务 aipproxy

2.15.1 概述

AIP TCP/IP 代理服务针是对 4 层网络通讯的转发和代理，7 层网络（如 http）转发由 NGINX 支持，不在该服务代理范围内。

服务架构见下图。



左边的负载均衡器由第三方提供。中间层是 aipproxy 代理服务，每个代理具有完全一致的配置。每个代理配置一组相同的对外开放的端口，如 1000 个端口：11333-12332，用于最多 1000 个并发作业的转发。这些端口由 AIP 为所需转发的作业统一分配和调度。

AIP 分配的端口通过共享文件系统通知 aipproxy。

2.15.2 安装和配置

1. 明确入口 IP 地址，和对外开放的端口范围
2. 配置 AIP
    1. cb.yaml 里增加共享资源，这个资源是 AIP 用于为需要转发的作业调度外网端口的。

```
cluster:
  resources:
  - name: proxy
    description: Number of ports configured in Proxy
    type: number
    direction: decrease
    assign: yes
    slotresource: no
    instances: "1000@[all]"
```

上面的 1000 是外网开放的端口总数。修改配置后需要重启 master 的 cbls:

```
cadmin lsrestart
```

验证：命令 chinfo -s 和 chosts -s 输出 proxy 配置的值，如 1000。

2. 在/opt/skyformai/etc/里增加一个配置文件 proxy.yaml:

```
pubip: 10.2.2.3
ports: 11333-12332
```

上面两行定义外网入口 IP（pubip）和对外开放的端口范围（ports），端口总数应该和 cb.yaml 里配置的 proxy 里 instances 的数字一致。

3. 安装和配置 aipproxy 运行 aipproxy 的主机必须是 AIP 的客户端（或集群主机），可以访问 aip 集群的 /opt/skyformai/etc 和/opt/skyformai/work（共享 aip 集群的这两个目录，按照原路径格式挂过来）。常规实践在是应用平台的 NGINX 的主机上部署。AIP 安装包解压后，cd aipproxy，然后运行./aipproxy_install

安装后检查：systemctl status aipproxy 服务日志：/var/log/aipproxy 目录下

2.15.3 作业提交

需要使用转发的作业需要增加参数 proxy。例子：

sshdjob:

```
csub -proxy sshdjob
```

cread 的结果会自动使用外网入口 IP 和对外开放的端口。输出分两段，用分号隔开:

```
ssh -i key -p 11333 u001@10.2.2.3;ssh -i key -p 16331 u001@192.168.10.10
```

第一段是外网访问命令行，第二段是内网访问命令行。

vncsub:

```
vncsub -proxy xterm
```

cread 的结果里 vncip 指向对外开放的端口，如 10.2.2.3，vncport 指向对外开放的端口，如 11333，本机 IP 地址和端口放在 localip 和 localport 里，例子:

```
/aj/192.168.10.10/16000/crv.html?path=/aj/192.168.10.10/16000?token=token5240&
↪autoconnect=true&password=bDLrbPUQEqOn4n.Y74crwi51YDMXY7O0&quality=9&compression=5&
↪vncip=192.168.0.10&vncport=10001&localip=192.168.10.10&localport=5901&resize=remote
```

如果 vncsub.yaml 里定义了 vncpubweb_ips，vncsub -prioxy 里的转发将不起作用。

2.16 节电调度插件

AIP 节电策略根据 AIP 集群中的主机的作业管理开启和关闭主机电源，以达到数据中心节电的功效。AIP 支持自动在主机闲置一段时间后关机，节省能源，并根据作业需要打开主机的电源，随着工作负载的变化采取相应的电源管理操作。

缺省的调度策略针对集群中所有主机采用同样的策略。在异构集群中，往往需要根据不同主机的应用场景使用不同的节电策略。使用外插的节电调度程序可以解决这类问题。

使用外插节电调度程序，在*cb.yaml* 中配置：

- 调度器调用节电策略的时间间隔
- 开机命令
- 关机命令
- AIP 支持节电策略插件：自定义脚本 power_ext_sched

AIP 根据 cb.yaml 配置的节电策略周期间隔，定期调用 power_ext_sched，并输入当前集群各队列主机和作业状态，power_ext_sched 评估当前集群状态，决定是否执行开关机动作。power_ext_sched 调用 AIP 命令执行开关机动作。

AIP 记录由节电策略触发的开关机，配置机器空闲功率后，可以统计总节电量：

总节电 = 机器空闲功耗 * 关机时间 * 机器数量

2.16.1 启动节电策略

修改配置文件/opt/skyformai/etc/cb.yaml，添加以下参数：

- power_sched_interval

  电源节能调度器调用的时间间隔，单位为秒。默认情况下，它会调用内置的节能调度器。如果存在可执行文件 /opt/skyformai/sbin/power_ext_sched，则会在此时间间隔内调用它以替代内置的节能调度器。

  当调用 power_ext_sched 时，cbsched 会将数据以 JSON 格式通过管道传递给 power_ext_sched 程序，即 power_ext_sched 应该从其标准输入中读取 JSON 数据，以便外部调度器做出决策并采取行动。

  该 JSON 的示例输出为 /opt/skyformai/sbin/power_ext_sched.input.json。

  默认值：60 秒。

- power_up_cmd

  在主控主机上执行的开机命令。该命令以主管理员身份执行，即 administrators 部分中定义的第一个用户。命令语法为”command host1 host2 …”.

- power_down_cmd

  在主控主机上执行的关机命令。该命令以主管理员身份执行，即 administrators 部分中定义的第一个用户。命令语法为”command host1 host2 …”.

2.16.2 节电策略插件案例

节电策略需求

1. L20 的主机的队列：保持空闲 3 台开机，空闲 12 小时关机，另外 7 台开启节电，集中分配。
2. A100 主机全部专属队列：没用到的机器全部关机，g001 和 g002 是子网管理器，至少保持一台开机。
3. 天数 GPU 的主机的队列：保持空闲 3 台开机，空闲 12 小时关机，另外 7 台做计算队列，开启节电，集中分配
4. 20 台属于 login 队列的主机，不配置节电。
5. 其他机器都属于计算队列，开起节电，按照等待作业需求开机，分批。

2. power_ext_sched 定制开发

配置策略

配置字典定义了不同队列的电源管理策略，包括：

- max_idle_hours：主机最大允许空闲时间，超过则考虑关机。
- min_idle_hosts：队列中需保持的最小空闲主机数量。
- mg_hosts：管理主机列表，这些主机在关机时有特定的保护或限制。
- up_batch_size：每批次开机的主机数量，防止同时开机导致电路故障。

  power_ext_sched 代码中，针对以上例子需求列配置如下：

```
config = {
  "g02": {
    "max_idle_hours": 12,
    "min_idle_hosts": 3,
    "mg_hosts": [],
    "up_batch_size": 0,
  },
  "g01": {
    "max_idle_hours': 12,
    "min_idle_hosts": 1,
    "mg_hosts": ['g001', 'g002'],
    "up_batch_size": 0,
  },
  "g21": {
    "max_idle_hours": 12,
    "min_idle_hosts": 3,
    "mg_hosts": [],
    "up_batch_size": 0,
  },
  "c01": {
    "max_idle_hours": 12,
    "min_idle_hosts": 0,
    "mg_hosts": [],
    "up_batch_size": 5,
  },
}
```

数据库

用 SQLite 数据库，文件放在：/opt/skyformai/work/power.db，存储主机开关机历史。用于：

- 选择最久未开机的主机进行开机，确保主机轮流使用，避免某些主机长时间未使用导致故障。
- 记录关机时的时间，防止频繁开关机。

数据库 schema:

```
PWDB = '/opt/skyformai/work/power.db'
PWTABLE = 'power_hist'
PWCOLS = '(host, queue, ncpu, ngpu, memmb, uptime, offtime)'
CREATE_TABLE_SQL = 'CREATE TABLE IF NOT EXISTS %s ( \
   host text PRIMARY KEY, \
   queue text NOT NULL, \
   ncpu integer, \
   ngpu integer, \
```

(下页继续)

(续上页)

```
    memmb text, \
    uptime integer, \
    offtime integer \
)' % (PWTABLE)
```

主机选择策略

开机选择

- 优先选择最久关机的主机，以平衡主机使用。
- 根据待处理作业需求，动态调整开机主机数量。
- 限制一次开机的主机数量，避免同时开机引发电路故障。

关机选择

- 针对长期空闲的主机，尤其是非管理主机，进行关机操作。
- 随机选择主机关机，防止特定主机长期空闲积累问题。

执行开关机动作

- 关机命令“csadmin pwsave $host”

  调度器会执行 cb.yaml 配置的关机命令 power_down_cmd

  “csadmin hhist”可以查到关机记录

- 开机命令“csadmin pwup $host”

  调度器会执行 cb.yaml 配置的关机命令 power_up_cmd

  “csadmin hhist”可以查到开机记录

2.16.3 节电统计报表

/opt/skyformai/bin/pwacct 用于统计和报告系统中的电源节约情况。通过解析历史记录数据，计算各主机在指定时间段内的电源节约时长和相应的电力节约量，并以 JSON 格式输出统计结果。脚本支持多种命令行参数，包括指定时间范围、输出详细信息以及调试模式。

功率配置

pwacct 中的 pwr_config 以队列或主机名的方式配置不同主机类型的功率。用于后续计算每个主机的电力节约量。

```
pwr_config = {
  'queue': {
  'g01': 640,
  'g02': 160,
  'g21': 480,
  'c01': 100
  },
  'host': {
    'node2': 100,
```

(下页继续)

(续上页)

```
  }
}
```

pwacct 支持的查询参数

- –start：指定开始日期，格式为 YYYY-MM-DD。
- –end：指定结束日期，格式为 YYYY-MM-DD。
- –all：统计所有时间段的数据。
- –long：以详细格式输出结果，包含每台主机的节电量。
- –debug：启用调试模式，输出调试日志。

例子 1:

```
pwacct
{
    "save_hours": 123.88,
    "save_pw": 12388.0,
    "num_hosts": 1
}
```

例子 2:

```
pwacct --long
{
    "save_hours": 123.88,
    "save_pw": 12388.0,
    "num_hosts": 1,
    "hosts": [
        {
            "host": "node2",
            "power": 100,
            "save_hours": 123.88,
            "save_pw": 12388.0
        }
    ]
}
```

例子 3:

```
pwacct --all
{
    "save_hours": 126.16,
    "save_pw": 12616.0,
    "num_hosts": 1
}
```

例子 4:

```
pwacct --all --long
{
    "save_hours": 126.16,
    "save_pw": 12616.0,
    "num_hosts": 1,
```

(下页继续)

(续上页)

```
    "hosts": [
        {
            "host": "node2",
            "power": 100,
            "save_hours": 126.16,
            "save_pw": 12616.0
        }
    ]
}
```

2.17 大机群或者高通量负载集群操作系统调参

2.17.1 调度器主机

当集群中主机数量超过 500，或者活动作业（等待和运行作业总和）数超过十万（100,000），或者有频繁的命令交互，如多个脚本频繁通过调用 bsub 命令从*jservice* 获取作业状态，调度器主机（master）的 Linux 操作系统需要调整网络通讯参数。

安装完成 AIP 后，运行以下命令设置参数，并使这些参数立即生效:

```
cat <<EOF >> /etc/sysctl.d/20-aip.conf
net.core.somaxconn=65535
net.ipv4.tcp_max_syn_backlog=8192
net.core.netdev_max_backlog=10000

net.core.wmem_max=16777216
net.core.rmem_max=16777216

net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_fin_timeout=5
net.ipv4.ip_local_port_range=16500 65535

net.ipv4.tcp_keepalive_time=600
EOF

sysctl -q  -p /etc/sysctl.d/20-aip.conf
```

备注: 所有调度器的备用主机，一般为集群配置*cb.yaml* 中 hosts 部分的前面 3 台主机，都需要运行以上命令调整参数

如果过集群中有持续海量作业提交，为了提供及时的作业提交响应，可以在把调度器的调度周期略微拉长一些。推荐为 10-15 秒（缺省为 4 秒）。在/opt/skyformai/etc/cb.yaml 中修改参数如下:

```
general:
  sched_interval: 10
```

然后重启调度器:

```
csadmin reconfig
```

当集群主机数超过 1000 台时，调整*cb.yaml* 中主机负载采集时间:

```
cluster:
  load_interval: 15
```

2.17.2 登录主机

如果用户在这些主机上运行脚本，里面有大量 AIP 的命令，如 bjobs 查询作业状态，需要修改缺省的 Linux 最大打开文件数。可以用以下命令修改最大打开文件数:

```
cat <<EOF >> /etc/security/limits.conf
* soft nofile 600000
* hard nofile 600000
EOF
```

用户登录后可以检查参数 ulimit -n。很多 Linux 发行版缺省值为 1024，对于生产系统，这个值太小。

小技巧: 通过 AIP 启动的作业中的最大打开文件数由 AIP 服务/lib/systemd/system/aip.service 文件控制，这个最大值为 40000。可以运行命令检查：csub -I ulimit -n

---

2.18 故障处理

2.18.1 系统日志

SkyForm AIP Daemon 日志存放 AIP 操作相关的告警或出错信息。当系统中有异常行为出现时，可以查看日志。

日志文件的位置在 **localtop/log**（如/opt/skyformai/log）。

日志文件名的格式为：daemon.host.log，其中 daemon 为 AIP daemon 进程的名字，host 为主机名，如/opt/skyformai/log/cbls.mgt01.log 为 CBLS 在 master 主机 mgt01 上的日志。

备注: AIP 的日志文件每 24 小时自动检查一次文件大小，检查时若超过 10MB，会自动滚动。滚动时，AIP 的 daemon 的日志只会保留两份，一份是前一个的的，一份是当前的。

2.18.2 常见问题

Daemon 问题

现象:

命令 systemctl status aip 或 service aip status 显示某些 daemon 进程没有启动

定位:

- 主机名是否能被解析？(执行 **hostname –fqdn** 命令必须输出正确的主机名）？
- /opt/skyformai/etc/hosts 文件是否配置，并且包含所有的集群主机的 IP 地址和主机名？
- <*daemon.hostname*> log file 中是否有上报的错误？
- 主机是否能访问 **cb.yaml** 文件？

- 防火墙是否关闭?
- 所需的软件包是否在各个主机上已安装?
- cb.yaml 里定义的用户名是否存在并在所有主机上都一致?

cbls 问题

问题 1

现象:

Received request from invalid host

定位:

- 主机是否使用多块网卡?
- 主机是否在 **cb.yaml** 文件中定义?
- 是否已重启 AIP 服务以识别新的主机?

cbsched 问题

现象:

cbsched 进程没有正常启动

定位:

- **cb.yaml** 文件中是否提示有配置错误 (可看命令 aip reconfigsched 的输出是否报错)?
- cbsched.<master>.log 的日志文件中是否出现错误信息?
- cbls 是否运行正常?
- 目录 **localtop/work** 是否为 SkyForm AIP 第一管理员所有且具有写权限?

用户的典型问题

问题 1

现象:

作业提交被拒

定位:

- 查看作业定义语法。
- 查看等待作业是否超出限制。
- 查看 cbsched 是否没有正常运行。

问题 2

现象：

作业长时间等待

定位：

运行 cjobs -lp 作业 ID 查看作业等待原因。列出的原因为调度在各个可以被调度的主机上未能成功调度的原因。

原因只列出作业可以被调度的主机，如队列只限制部分主机，或者用户提交作业时指定某些主机。另外 maxslots: 0 的主机不会被考虑在调度的主机范围内。

- 用户是否要求了过多的资源？例如
  - 要求的内存多于主机上的内存总量
  - 定义的资源限制太过严格
- 用户 ID 在作业运行主机上是否有效?
- 用户是否请求了过多的作业执行?
- 使用 aip job info -l 查看作业等待的原因。

问题 3

现象：

我的作业失败了

定位：

- 确认从执行主机上可以访问应用及其数据文件。
- 使用 **cjobs -l** 查看上报的退出代码（exit code）。
- 常见的 exit code
  - 127 –命令找不到
  - 128 –命令不能执行
  - by signal N –命令被信号 N 中断（查看 man signal 和 kill -l 了解 Linux 系统信号的信息）
  - 其他退出码 - 这些退出码是应用程序自身的退出码，请询问应用程序的作者

提交作业失败，参考*csub* 尾部的排错部分。

作业不运行，参考*cjobs* 里的作业等待原因。

作业异常退出，参考*cjobs* 里的作业退出原因。

问题 3

现象：

我有异常作业

定位：

异常作业一般为超时限运行、内存使用不足、CPU 使用不足等。可以使用*cjobs* 的-S 搜索表达式定义搜索条件，找到符合条件的作业，再做详细分析。

CHAPTER 3

使用

3.1 作业定义和管理

用户通过作业定义来向系统请求资源，用户也可用命令查看系统信息，作业及任务的状态等。

3.1.1 系统命令

SkyForm AIP 的命令接口是 aip。

```
[root@linux7 key]# aip
NAME:
  aip - SkyForm AI Platform that manages cluster resources for distributed AI ap
plications and Docker containers

USAGE:
  aip [global options] command [command options] [arguments...]

VERSION:
  9.0 (5c0e6c293b43e95b9287b0be568405d5593dbf3d)

BUILD TIME:
  2019-02-04 23:34:57 -0500 EST

COMMANDS:
    cluster, c   Display cluster information
    cronjob, cj  Run, control and display time-based jobs
    job, j       Run, control and display jobs
    host, h      Display host information
    queue, q     Display queue information
    admin, a     Administrative tool to control SkyForm AI daemons
    help, h      Shows a list of commands or help for one command

GLOBAL OPTIONS:
  --help, -h     show help (default: false)
  --version, -v  print the version (default: false)

COPYRIGHT:
  SkyForm AI (c) 2018
```

普通用户能够使用 aip cluster, aip host, 和 aip queue 来查询系统中的资源，用 aip job 来运行和管理自身的作业。

系统管理员除了可以运行以上命令外，还可以运行 aip admin 来重启远程的 AIP 服务和让 AIP 服务重读修改过的系统配置。

aip 的例子：

```
[u001@centos65 ~]$ aip cluster info
MASTER       ADMIN      NUM_NODES
centos65     cadmin     2

HOST         NCPUS     NGPUS     MAXMEM     MAXSWP     TAGS
centos65     3         4         987M       1983M      (cs)
ecp1         4         -         974M       4095M      ()
[u001@centos65 ~]$ aip cluster res
RESOURCE     USABLE     RESERVED       LOCALE
netres2      301        0.0            centos65 ecp1
lic          4          0.0            centos65 ecp1
gpu0         3          0.0            centos65
gpu0         3          0.0            ecp1
```

```
[u001@centos65 ~]$ aip host info
HOST         STAT     CPU      GPU      MEM      MAX     TOTAL    RUN     USTOPPED     SSTOPPED
centos65     ok       0.13%    0.00%    663M     3       0        0       0            0
ecp1         ok       0.20%    -        727M     6       0        0       0            0
[u001@centos65 ~]$ aip host res
HOST centos65
            TOTAL     AVAIL     RESERVED
SLOTS       3         3         0
GPU         4         4.0       0.0
MEM         663M      663M      0M
netres2     -         301       0.0
lic         -         4         0.0
gpu0        -         3         0.0

HOST ecp1
            TOTAL     AVAIL     RESERVED
SLOTS       6         6         0
GPU         -         -         -
MEM         727M      727M      0M
netres2     -         301       0.0
lic         -         4         0.0
gpu0        -         3         0.0
```

3.1.2 运行作业

作业需要一个 JSON 格式的作业定义。

直接运行命令 aip job run 会启动默认的文本编辑器让用户编辑 JSON 作业定义。最基本必填的作业定义参数是作业命令行 Command。如下例子：

```
aip job run
```

```
{
  "Command": "hostname",        # Mandatory. Command to run.
  "JobName": "",               # Defaults to Command for regular job.
                               # Mandatory for cron job.
  "Interactive": true,         # Defaults to true.
                               # Specify false to submit job in batch mode.
  "MinNumSlots": 1,            # Minimum number of required job slots.
                               # Defaults to 1.
  "MaxNumSlots": 1,            # Maximum number of required job slots.
                               # Defaults to MinNumSlots.
  "Resource":                  # Resource Requirement of the job.
  {
    "Select": "",              # Select execution host.
    "Need": ""                 # Resource needed for the job.
  },
  "Envs": [],                  # A list of environment variables.
  "Tasks": [],                 # A list of task specification.
  "Debug": false               # Enable debug.
}
```

写入文件退出后，作业开始运行并显示作业命令的输出：

3.1.3 高性能计算作业命令

SkyForm AIP 提供了一组高性能计算作业管理命令。具体使用请参考在线 man page。

- csub：作业提交。例子：

```
csub sleep 1h
```

```
[cadmin@amaster ~]$ csub sleep 1h
Job 427 has been submitted to the default queue [medium].
```

- cjobs：查询作业状态。例子：

```
cjobs -l 427
```

> **备注：** cjobs 和 LSF 兼容命令 bjobs 得到相同的结果。

```
[cadmin@amaster ~]$ bjobs -l 427

Job <427>, User <cadmin>, Project <default>, Status <RUN>, Queue <medium>, Comm
                     and <sleep 1h>
Wed Aug  7 09:26:55: Submitted from host <amaster>, CWD <$HOME>;
Wed Aug  7 09:26:55: Started on <an0003>, Execution Home </home/cadmin>, Execut
                     ion CWD </home/cadmin>;
Wed Aug  7 09:37:25: Total resource usage collected.
                     MEM: 3 Mbytes;  SWAP: 241 Mbytes;  NTHREAD: 3
                     PGID: 68510;  PIDs: 68510 68513 68514


 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

              gpu
 loadSched     -
 loadStop      -

 RESOURCE REQUIREMENT DETAILS:
 Combined:
 Effective:

 RESOURCE ALLOCATION:
[{"ResSpec":"","MinSlots":1,"MaxSlots":1, "Hosts":[{"Name":"an0003","Port":16331}]}]
```

- ckill：杀掉作业
- cstop：暂停作业
- cresume：回复暂停的作业
- ctop：把自己的作业放到最前面作为第一个被调度的作业。这只是调整自身作业的调度顺序，不影响其他用户作业的优先级。
- cbot：把自己的作业放到最前面作为最后被调度的作业。这只是调整自身作业的调度顺序，不影响其他用户作业的优先级。
- cswitch：作业切换队列。

3.1.4 查看作业

运行作业后，可以执行以下命令 aip job info 查看作业：

```
[cadmin@ai0 tensorflow]$ aip job info
JOBID  USER     STAT  QUEUE   RUN_HOST  SUBMITTED          STARTED       ENDED
1735   cadmin   RUN   medium  ai3*3     Sep 14 15:20       Sep 14 15:20  -
                             ai2*1
                             ai1*2
       TASKID   STAT  CPNT    RUN_HOST  COMMAND            STARTED       ENDED
       9996.1   RUN   chief   ai3       ...mnist/model.py  Sep 14 15:20  -
       9994.1   RUN   ps      ai1       ...mnist/model.py  Sep 14 15:20  -
       9992.1   RUN   ps      ai1       ...mnist/model.py  Sep 14 15:20  -
       9990.1   RUN   worker  ai2       ...mnist/model.py  Sep 14 15:20  -
       10000.1  RUN   worker  ai3       ...mnist/model.py  Sep 14 15:20  -
       9998.1   RUN   worker  ai3       ...mnist/model.py  Sep 14 15:20  -
```

命令 aip job info -l 输出作业 JSON 格式的详细信息。

命令 aip job log 看作业的输出。如：

```
aip j log 5902
```

另一组作业查看命令是 cjobs 和 bjobs。具体命令使用请参考 cjobs 的 man page（用命令 man cjobs）。

例子 1：作业 434 处于暂停状态：

```
[cadmin@amaster ~]$ cjobs -l 434

Job <434>, User <root>, Project <default>, Status <PSUSP>, Queue <medium>, Comm
                     and <sleep 1h>
Wed Aug  7 11:44:46: Submitted from host <amaster> on hold, CWD <$HOME>;
 PENDING REASONS:
 The job was stopped by the user while in stopped state: 1 host;

 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

              gpu
 loadSched     -
 loadStop      -

 RESOURCE REQUIREMENT DETAILS:
 Combined:
 Effective:

 PENDING PART:
[{"ResSpec":"","MinSlots":1,"MaxSlots":1}]
```

例子 2：作业 437 正在运行，是有远程任务的 MPI 作业。

```
Job <437>, User <cadmin>, Project <default>, Status <RUN>, Queue <medium>, Comm
                     and <mpirun ./mpitest>
Thu Aug  8 08:42:28: Submitted from host <amaster>, CWD <$HOME>, 6 Processors R
                     equested, Requested Resources <6{span[ptile=2]}>;
Thu Aug  8 08:42:28: Started on 6 Hosts/Processors <2*an0003> <2*an0002> <2*an0
                     001>, Execution Home </home/cadmin>, Execution CWD </home/
                     cadmin>;
Thu Aug  8 08:42:38: Task information.
                     Component: default
                       Task: 127474.1
                         Command: /usr/lib64/mpich/bin/hydra_pmi_proxy --contro
                     l-port an0003:32913 --rmk lsf --launcher lsf --demux poll
                     --pgid 0 --retries 10 --usize -2 --proxy-id 1
                         Host: an0002
                         Status: RUN
                         Start time: Thu Aug  8 08:42:28
                         MEM: 6 Mbytes;  SWAP: 102 Mbytes;  NTHREAD: 3
                         PGID: 79171;  PIDs: 79171
                         PGID: 79172;  PIDs: 79172
                         PGID: 79173;  PIDs: 79173
                       Task: 127475.1
                         Command: /usr/lib64/mpich/bin/hydra_pmi_proxy --contro
                     l-port an0003:32913 --rmk lsf --launcher lsf --demux poll
                     --pgid 0 --retries 10 --usize -2 --proxy-id 2
                         Host: an0001
                         Status: RUN
                         Start time: Thu Aug  8 08:42:28
                         MEM: 6 Mbytes;  SWAP: 102 Mbytes;  NTHREAD: 3
                         PGID: 82878;  PIDs: 82878
                         PGID: 82879;  PIDs: 82879
                         PGID: 82880;  PIDs: 82880
Thu Aug  8 08:42:38: Total resource usage collected.
                     MEM: 37 Mbytes;  SWAP: 581 Mbytes;  NTHREAD: 16
                     PGID: 127468;  PIDs: 127468 127471 127472
                     PGID: 127473;  PIDs: 127473
                     PGID: 127474;  PIDs: 127474 127483
                     PGID: 127475;  PIDs: 127475 127481
                     PGID: 127476;  PIDs: 127476
                     PGID: 127477;  PIDs: 127477
                     PGID: 79171;  PIDs: 79171
                     PGID: 79172;  PIDs: 79172
                     PGID: 79173;  PIDs: 79173
                     PGID: 82878;  PIDs: 82878
                     PGID: 82879;  PIDs: 82879
                     PGID: 82880;  PIDs: 82880


 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

              gpu
 loadSched     -
 loadStop      -

 RESOURCE REQUIREMENT DETAILS:
 Combined: 6{span[ptile=2]}
 Effective: 6{span[ptile=2]}

 RESOURCE ALLOCATION:
[{"ResSpec":"span[ptile=2]","MinSlots":6,"MaxSlots":6, "Hosts":[{"Name":"an0003","Port":16331},{
"Name":"an0003","Port":16332},{"Name":"an0002","Port":16331},{"Name":"an0002","Port":16332},{"Na
me":"an0001","Port":16331},{"Name":"an0001","Port":16332}]}]
```

3.1.5 作业控制

- 杀掉作业：aip job kill *jobid*
- 重运行作业，即杀掉目前正在运行的作业，让其重新排队：aip job rerun {jobid}
- 暂停和恢复后台运行作业：aip job stop *jobid*; aip job resume *jobid*

  SkyForm AIP 的作业状态和说明见下表：

状态	说明
WAIT 或 PEND	作业在队列中等待。等待原因可用命令 aip j i -l 或 cjobs -lp 查看
RUN	作业在运行
FINISH 或 DONE	作业运行结束，exit 码为 0
WSTOP 或 PSUSP	作业在队列等待时被暂停调度。恢复可用 aip j rs
EXIT	作业被杀掉，或以非 0 的 exit 码结束
USRSTOP 或 USUSP	运行作业被用户或系统管理员暂停
SYSSTOP 或 SSUSP	运行作业被调度器按调度策略暂停

作业状态间的关系见下图。

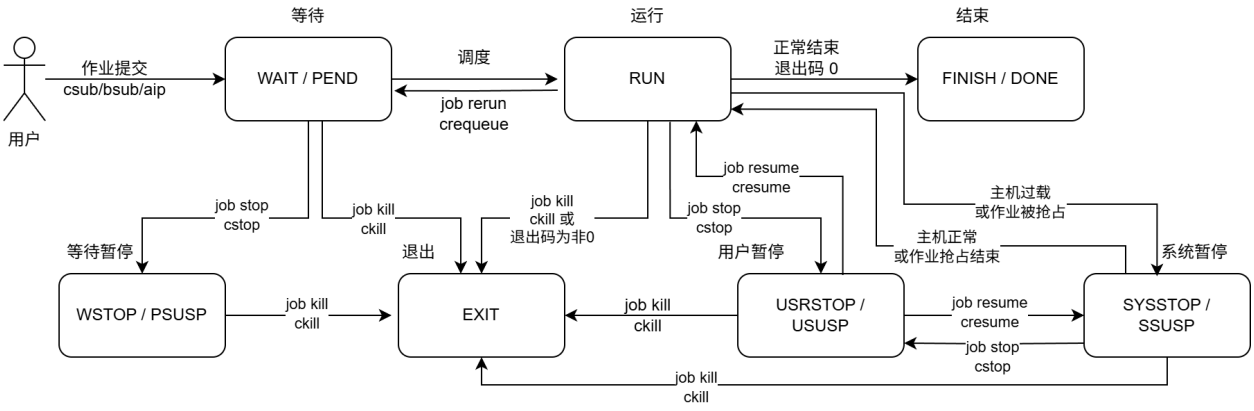


图 1: 作业状态间的关系

3.1.6 查看系统资源

集群例子的资源情况可以用命令 aip 来查看。

查看集群配置

命令：

```
aip cluster info
aip cluster res
```

例子：

```
[cadmin@ai0 tensorflow]$ aip cluster
NAME:
   aip cluster - Display cluster information

USAGE:
   aip cluster command [command options] [arguments...]

COMMANDS:
     info, i  Display cluster information
     res, r   Display custom resources

OPTIONS:
   --help, -h  show help

[cadmin@ai0 tensorflow]$ aip cluster info
MASTER    ADMIN     NUM_NODES
ai0       cadmin    4

HOST    NCPUS    NGPUS    MAXMEM    MAXSWP    TAGS
ai0     2        -        3933M     2047M     ()
ai1     1        -        1982M     2047M     (ps)
ai2     1        2        1982M     2047M     ()
ai3     1        4        1982M     2047M     ()
[cadmin@ai0 tensorflow]$ aip cluster res
2018/09/14 13:30:45 No resource defined.
[cadmin@ai0 tensorflow]$
```

查看系统配置的资源，包括通过从个主机上 RESS 得到的定制资源：cinfo

```
[root@d16 ~]# cinfo
RESOURCE_NAME   TYPE    DIR.  SLOT_RES ASSIGNED DESCRIPTION
r15s            Number  Inc      Yes       No   15-second CPU run queue length
r1m             Number  Inc      Yes       No   1-minute CPU run queue length (alias: cpu)
r15m            Number  Inc      Yes       No   15-minute CPU run queue length
ut              Number  Inc      Yes       No   1-minute CPU utilization (0.0 to 1.0)
pg              Number  Inc      Yes       No   Paging rate (pages/second)
io              Number  Inc      Yes       No   Disk IO rate (Kbytes/second)
ls              Number  Inc      Yes       No   Number of login sessions (alias: login)
it              Number  Dec      Yes       No   Idle time (minutes) (alias: idle)
tmp             Number  Dec      Yes       No   Disk space in /tmp (Mbytes)
swp             Number  Dec      Yes       No   Available swap space (Mbytes) (alias: swap)
mem             Number  Dec      Yes       No   Available memory (Mbytes)
gpu             Number  Dec      Yes      Yes   Number of GPUs
ncpus           Number  Dec       No       No   Number of CPUs
ndisks          Number  Dec       No       No   Number of local disks
maxmem          Number  Dec       No       No   Maximum memory (Mbytes)
maxswp          Number  Dec       No       No   Maximum swap space (Mbytes)
maxtmp          Number  Dec       No       No   Maximum /tmp space (Mbytes)
cpuf            Number  Dec       No       No   CPU factor
rexpri          Number  N/A       No       No   Remote execution priority
ngpus           Number  Dec       No       No   Number of GPUs
slots           Number  Dec       No       No   Available slots (job slots resource)
maxslots        Number  Dec       No       No   Available Maximum job slots (MXJ resource)
server             Tag  N/A       No       No   AIP server host
cs                 Tag  N/A       No       No   compute server
type              Text  N/A       No       No   Host type
model             Text  N/A       No       No   Host model
status            Text  N/A       No       No   Host status
hname             Text  N/A       No       No   Host name

TYPE_NAME
LINUX

MODEL_NAME      CPU_FACTOR      ARCHITECTURE
xeon                100.00      x86 64
```

查看各主机负载和资源使用

命令:

```
aip host info
aip host res
```

例子:

```
[cadmin@ai0 tensorflow]$ aip host info
HOST     STAT     CPU      GPU      MEM      MAX     TOTAL    RUN      USTOPPED     SSTOPPED
ai0      ok       0.20%    -        3651M    2       0        0        0            0
ai1      ok       0.00%    -        1808M    2       0        0        0            0
ai2      ok       0.00%    0.00%    1805M    2       0        0        0            0
ai3      ok       0.20%    0.00%    1795M    4       0        0        0            0
[cadmin@ai0 tensorflow]$ aip host res
HOST ai0
         TOTAL    AVAIL    RESERVED
SLOTS    2        2        0
GPU      -        -        -
MEM      3651M    3651M    0M

HOST ai1
         TOTAL    AVAIL    RESERVED
SLOTS    2        2        0
GPU      -        -        -
MEM      1808M    1808M    0M

HOST ai2
         TOTAL    AVAIL    RESERVED
SLOTS    2        2        0
GPU      2        2.0      0.0
MEM      1805M    1805M    0M

HOST ai3
         TOTAL    AVAIL    RESERVED
SLOTS    4        4        0
GPU      4        4.0      0.0
MEM      1795M    1795M    0M

[cadmin@ai0 tensorflow]$
```

查看主机详细信息:

```
chosts -l
```

```
[root@amaster ~]# chosts -l
HOST  amaster
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV DISPATCH_WINDOW
closed_Adm     100.00     -      2      0      0      0      0      0      -

 CURRENT LOAD USED FOR SCHEDULING:
             r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem    gpu
 Total        0.0   0.0   0.0    1%   0.0     0    0 2e+08 3596M 1024M 1582M    0.0
 Reserved     0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M    0.0


 LOAD THRESHOLD USED FOR SCHEDULING:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

             gpu
 loadSched    -
 loadStop     -


HOST  an0001
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV DISPATCH_WINDOW
ok             100.00     -      4      0      0      0      0      0      -

 CURRENT LOAD USED FOR SCHEDULING:
             r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem    gpu
 Total        0.0   0.0   0.0    0%   0.0     0    0 2e+08 6608M 1023M  748M    0.0
 Reserved     0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M    0.0


 LOAD THRESHOLD USED FOR SCHEDULING:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

             gpu
 loadSched    -
 loadStop     -
```

查看队列情况

命令:

```
aip queue info
```

例子:

```
[cadmin@ai0 tensorflow]$ aip queue info
NAME       PRIO     STATUS     MAX     TOTAL     WAIT     RUN     STOPPED     SL/U     SL/H
high       5        OK         -       0         0        0       0           -        -
short      4        OK         -       0         0        0       0           -        -
medium     3        OK         -       0         0        0       0           -        -
low        2        OK         -       0         0        0       0           -        -
[cadmin@ai0 tensorflow]$
```

队列详细信息:

```
cqueues -l
```

```
[root@amaster ~]# cqueues -l medium

QUEUE: medium
  -- No description provided.

PARAMETERS/STATISTICS
PRIO NICE     STATUS       MAX JL/U JL/P JL/H NJOBS  PEND  RUN  SSUSP USUSP  RSV
   3   0    Open:Active      -    -    -    -     0     0     0     0     0    0

Schedule delay for a new job is 0 seconds
Interval for a host to accept two jobs is 0 seconds

SCHEDULING PARAMETERS
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

              gpu
 loadSched     -
 loadStop      -

SCHEDULING POLICIES:  FAIRSHARE
USER/GROUP  SHARES     PRIORITY    PEND     RUN   USUSP   SSUSP      RUNTIME
cadmin           1        1.500       0       0       0       0        0.000
root             1        1.500       0       0       0       0        0.057
default          1        3.000       0       0       0       0        0.000

USERS:  all users
HOSTS:  all hosts used by the AIP system
```

3.1.7 LSF 兼容命令

SkyForm AIP 提供与 LSF 兼容的命令。具体使用可参考 LSF 的使用说明：

- bsub ： 提交作业
- bjobs：查看作业状态
- bkill：杀掉作业
- bqueues：查看队列状态
- bpost/bread：发布与作业有关的消息，读取与作业有关的消息
- bpeek：查看实时作业输出
- lsid：产品和集群信息
- blaunch：MPI 的远程任务执行工具

3.1.8 SLURM 兼容命令

- sbatch: 作业提交
- salloc: 交互作业提交
- srun: 作业中执行的远程任务管理（常用于 MPI）
- sinfo: 集群概况
- squeue: 队列信息
- scontrol: 主机、作业信息详情和控制

- scancel: 杀作业
- sacct: 作业计量和历史

3.1.9 PBS 兼容命令

- qsub: 作业提交
- qstat：作业、队列状态
- qdel: 杀作业
- pbsnodes: 主机详情

3.1.10 作业提交资源需求

参考*csub* 里-R 参数的说明。

3.2 机器学习和数学分析作业

SkyForm AIP 提供了一些作业定义的例子和深度学习样板数据。这些材料在软件包 https://skyformaip.com/aip/skyformaip_examples.tar.gz 里。

3.2.1 分布式 TensorFlow 的作业定义

TensorFlow 2.x 是一个与 TensorFlow 1.x 使用体验完全不同的框架，TensorFlow 2.x 不兼容 TensorFlow 1.x 的代码。SkyForm AIP 同时支持 TensorFlow 2.x 和 TensorFlow 1.x 分布式作业。

分布式 TensorFlow1.x 的作业定义

分布式 TensorFlow 是由多个任务组成的。任务有 Master，Parameter Server（ps）和 Worker 三类。在有 GPU 的情况下，Master 和 Worker 利用 GPU 可以提高性能。

下面是 TensorFlow 学习 mnist 的作业定义例子:

```
{
  # Mandatory.
  # Job type for tensorflow job
  "JobType": "tensorflow",

  # Mandatory if task level command is not specified.
  # Tensorflow job command
  # The job level command will be inherited by each task if the task does
  # not specify a command. Otherwise, task level command will overwrite
  # job level command
  "Command":"python mnist/model.py",

  # Optional.
  # A list of environment variables for the tensorflow job.
  # The environment variables will be set for each task.
  "Envs":
  [
```

(下页继续)

(续上页)

```
    "TF_DATA_DIR=mnist/data",
    "TF_MODEL_DIR=mnist/model",
    "TF_TRAIN_STEPS=2000"
  ],

  # Optional.
  # The shared directory where the output of tasks will be written to.
  # Each task will have a separate output file generated under:
  # <TaskLogDir>/<JobID>.<User>/<Component>.<Index>.<Host>.<PID>
  "TaskLogDir": "log",

  # Optional.
  # Suppress standard output and error of tasks. If TaskLogDir is specified,
  # the standard output and error of tasks will still be written to the
  # directory
  "SuppressTaskOutput": false,

  # Optional.
  # Buffer standard output and error of tasks every second before printing
  # out by task for better readability.
  # If SuppressTaskOutput is set to true, this field is ignored.
  "BufferTaskOutput": false,
  # A list of task specification
  "Tasks":
  [
    {
      # Mandatory.
      # Component name for the task
      "Component":"PS",
      "MinNumTasks": 1,
      "MaxNumTasks": 2,
      "Resource":
      {
        # 若系统中主机没有定义ps标签，删除以下这行。可用chinfo查看主机标签
        "Select": "ps"
      }
    },
    {
      "Component":"MASTER",
      "MaxNumTasks": 1,
      "Resource":
      {
        "Need": "gpu=2"
      }
    },
    {
      "Component":"WORKER",
      "MinNumTasks": 2,
      "MaxNumTasks": 4,
      "Resource":
      {
        "Need": "gpu=1"
      }
    }
  ]
}
```

参数说明：

JobType：
必须定义为 tensorflow，以便于作业控制器针对 TensorFlow 的任务做合理的处理，如为每个任务设置端口。

Command：
定义运行 Python 的程序

Envs：
定义 TensorFlow 所需的环境变量：TF_DATA_DIR，TF_MODEL_DIR，和 TF_TRAIN_STEPS。请参考 TensorFlow 的说明设置这些变量。

TaskLogDir：
定义一个目录放置各个任务的输出。每个任务的输出会写到单独的一个文件里，以便观察和排错。若不定义该目录，各个任务的输出会显示在命令 aip job run 的终端上，各个任务的输出的每行的前段会增加一个判别该任务的字串，以方便识别。因为各个任务是并行执行的，各个任务的输出行会交替。

SuppressTaskOutput:
禁止任务输出到终端上，以便排错。一般不需要修改该参数。

BufferTaskOutput:
为了方便观察各任务交替输出的结果，这个参数设成 true 时各个任务的输出会在各自的缓冲区里，每秒集中输出一次，这样每个任务在每秒内的输出集中显示到终端上，方便观察。

Tasks: 定义各类任务数和所需资源。

Component:
任务类别名，值必须是 MASTER、PS、或 WORKER。

MinNumTasks/MaxNumTasks:
此类任务的最小和最大任务数。在集群的环境中，任务多的作业不容易有机会拿到大块资源而长期等待。而任务数若定义少了又会减少并行度，影响作业运行时间。定义最小和最大任务数可以让调度器根据最大任务数调度资源，当资源不能满足最大任务数是，减少任务数目，直到最小值为止，这样能让作业的等待时间缩短并最大可能地得到可用资源。

当 MinNumTasks 没有定义时，默认为 1。当 MaxNumTasks 没有定义时，默认于 MinNumTasks 相等。

Resource：
定义每个任务所需的资源。以上的例子中，MASTER 任务只有一个，需要两个 GPU, 而每个 WORKER 需要 1 个 GPU。

运行作业:

```
aip job run tensorflow.mnist.json
```

当作业开始运行时，作业控制器会：

1. 根据调度器分配的主机槽位启动任务
2. 根据调度器分配的各主机上的端口为 TensorFlow 任务设置所用端口
3. 根据调度器分配的各主机上的 GPU 为每个需要 GPU 的任务设环境变量 CUDA_VISIBLE_DEVICES 控制任务可用的 GPU。
4. 监控各个任务状态和资源使用（CPU 和内存）情况。

除了作业的终端输出外，我们可以在另一个终端里看到各个任务的输出内容：

```
[cadmin@ai0 1004.cadmin]$ ls -l
total 32
-rw-r--r-- 1 cadmin cadmin 6371 Sep 13 17:18 chief.0.ai3.1370
-rw-r--r-- 1 cadmin cadmin 3883 Sep 13 17:11 ps.0.ai0.1368
-rw-r--r-- 1 cadmin cadmin 3883 Sep 13 17:11 ps.1.ai0.1369
-rw-r--r-- 1 cadmin cadmin 5015 Sep 13 17:18 worker.0.ai1.1371
-rw-r--r-- 1 cadmin cadmin 5075 Sep 13 17:18 worker.1.ai1.1367
[cadmin@ai0 1004.cadmin]$
```

作业屏幕输出例子:

```
worker.1 (ai1.1371) INFO:tensorflow:loss = 0.28630352, step = 804 (87.786 sec)
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.55739
worker.2 (ai1.1367) INFO:tensorflow:loss = 0.25960848, step = 825 (85.810 sec)
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.44942
chief.1 (ai3.1370) INFO:tensorflow:loss = 0.37597466, step = 984 (44.934 sec)
chief.1 (ai3.1370) INFO:tensorflow:Saving checkpoints for 1000 into mnist/model/model.ckpt.
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.49571
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.19368
chief.1 (ai3.1370) INFO:tensorflow:loss = 0.33835834, step = 1186 (47.497 sec)
worker.1 (ai1.1371) INFO:tensorflow:loss = 0.1323377, step = 1205 (92.580 sec)
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.24629
worker.2 (ai1.1367) INFO:tensorflow:loss = 0.19895774, step = 1218 (90.860 sec)
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.32089
chief.1 (ai3.1370) INFO:tensorflow:loss = 0.09624856, step = 1388 (46.669 sec)
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.37233
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.4402
chief.1 (ai3.1370) INFO:tensorflow:loss = 0.13250685, step = 1588 (45.227 sec)
worker.2 (ai1.1367) INFO:tensorflow:loss = 0.1337761, step = 1608 (88.752 sec)
worker.1 (ai1.1371) INFO:tensorflow:loss = 0.062080424, step = 1610 (92.144 sec)
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.44061
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.30593
chief.1 (ai3.1370) INFO:tensorflow:loss = 0.13399652, step = 1790 (46.510 sec)
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.2673
chief.1 (ai3.1370) INFO:tensorflow:global_step/sec: 4.17405
chief.1 (ai3.1370) INFO:tensorflow:loss = 0.041079972, step = 1992 (47.677 sec)
chief.1 (ai3.1370) INFO:tensorflow:Saving checkpoints for 2001 into mnist/model/model.ckpt.
chief.1 (ai3.1370) INFO:tensorflow:Saving checkpoints for 2003 into mnist/model/model.ckpt.
worker.2 (ai1.1367) INFO:tensorflow:loss = 0.2949481, step = 1999 (91.553 sec)
worker.2 (ai1.1367) INFO:tensorflow:Loss for final step: 0.2949481.
worker.2 (ai1.1367) Extracting mnist/data/train-images-idx3-ubyte.gz
worker.2 (ai1.1367) Extracting mnist/data/train-labels-idx1-ubyte.gz
worker.2 (ai1.1367) Extracting mnist/data/t10k-images-idx3-ubyte.gz
worker.2 (ai1.1367) Extracting mnist/data/t10k-labels-idx1-ubyte.gz
chief.1 (ai3.1370) INFO:tensorflow:Loss for final step: 0.13002016.
chief.1 (ai3.1370) Extracting mnist/data/train-images-idx3-ubyte.gz
chief.1 (ai3.1370) Extracting mnist/data/train-labels-idx1-ubyte.gz
chief.1 (ai3.1370) Extracting mnist/data/t10k-images-idx3-ubyte.gz
chief.1 (ai3.1370) Extracting mnist/data/t10k-labels-idx1-ubyte.gz
worker.1 (ai1.1371) INFO:tensorflow:Loss for final step: 0.10378957.
worker.1 (ai1.1371) Extracting mnist/data/train-images-idx3-ubyte.gz
worker.1 (ai1.1371) Extracting mnist/data/train-labels-idx1-ubyte.gz
worker.1 (ai1.1371) Extracting mnist/data/t10k-images-idx3-ubyte.gz
worker.1 (ai1.1371) Extracting mnist/data/t10k-labels-idx1-ubyte.gz
INFO[2018-09-13T17:18:41.112859-04:00] The jobcontroller is stopped.
```

分布式 TensorFlow2.x 的作业定义

Tensorflow2.x 提供分布式策略接口，用户只需要改动较少代码就能分布现有模型和训练代码，让单机运行的训练任务在多个 GPU、多台机器上进行分布式训练。

分布式 TensorFlow2.x 是由多个任务组成的，所有的任务都是 Worker。其中，第一个 Worker 是“主要”工作者，除了常规的训练工作之外，还承担了更多的责任，比如保存检查点和为 TensorBoard 编写摘要文件。在有 GPU 的情况下，Worker 利用 GPU 可以提高性能。

多机分布式 TensorFlow2.x 作业使用 csub 命令提交，使用 tf-run.sh 命令运行，同时支持 GPU 或者 CPU 运行训练任务。

下面是 TensorFlow2.x 学习 mnist 的作业定义例子:

```
csub -n 4,8 -R "span[ptile=2] rusage[gpu=1]" -o %J.out -e %J.err tf-run.sh multi.py
```

参数说明:

-n:
需要的最小和最大 worker 数。比如例子中“4,8”，最少需要 4 个 worker，最多需要 8 个 worker。当最小 worker 数都不满足时，作业处于 pending 状态，等待空闲资源；如果最小 worker 数满足，训练任务开始执行；如果集群有更多空闲资源，调度器会尽可能多的启动 worker，同时不超过设定的最大 worker 数。如果只指定最小 worker 数，不指定最大 worker 数，比如“-n 4”，指定 worker 数为 4，即不使用弹性资源分配。

-R:
定义作业运行的资源需求。比如例子中“span[ptile=2]”，要求每个节点启动两个 worker；“rusage[gpu=1]”，要求给每个 worker 分配一块 GPU。“-R”支持更多的资源需求表达式，具体用法请参考*csub* 里-R 的说明。

-o:
定义标准输出写入文件的路径。%J 指定文件名为作业号。

-e:
定义标准错误输出写入文件的路径。%J 指定文件名为作业号。

tf-run.sh:
AIP 例子 里 tensorflow2 目录下提供的 TensorFlow2.x 多机分布式训练任务的 wrapper，负责将训练任务在调度器分配的资源上启动，并监控训练任务状态和收集资源使用信息。TensorFlow2.x 多机分布式训练任务必须通过 tf-run.sh 启动。

multi.py:
运行训练任务的 Python 程序，后面可以跟程序参数，程序名和参数都是用户自定义的，比如“main.py –epoch=10”。

csub 命令支持更多参数，具体用法请参考*csub* 。

当提交作业并开始运行时，tf-run.sh 会:

1. 根据调度器分配的主机槽位启动任务
2. 根据调度器分配的各主机上的端口为 TensorFlow 任务设置所用端口
3. 根据调度器分配的各主机上的 GPU 为每个需要 GPU 的任务设环境变量 CUDA_VISIBLE_DEVICES 控制任务可用的 GPU。
4. 监控各个任务状态和资源使用（CPU 和内存）情况。

以下功能是训练任务程序自己实现的，tf-run.sh 不会涉及，SkyForm AIP 只提供基于分布式文件系统的共享目录:

1. 在分布式文件系统中保存检查点文件，以便在重新启动先前失败的实例后，将获得以前的状态，继续训练。

2. 为可视化工具 TensorBoard 编写摘要文件，包含模型，数据和 graph 等信息。
3. Epoch、batch_size、learning_rate 等跟训练相关的参数。
4. 下载训练数据和保存模型等跟训练相关的功能。

TensorBoard 是一套 Web 应用程序，用于检查和了解模型运行和图形。TensorBoard 当前支持五种可视化：标量，图像，音频，直方图和图形 。通过 TensorBoard 可以监控训练的收敛性，方便调整参数 epoch、batch_size、learning_rate 等。

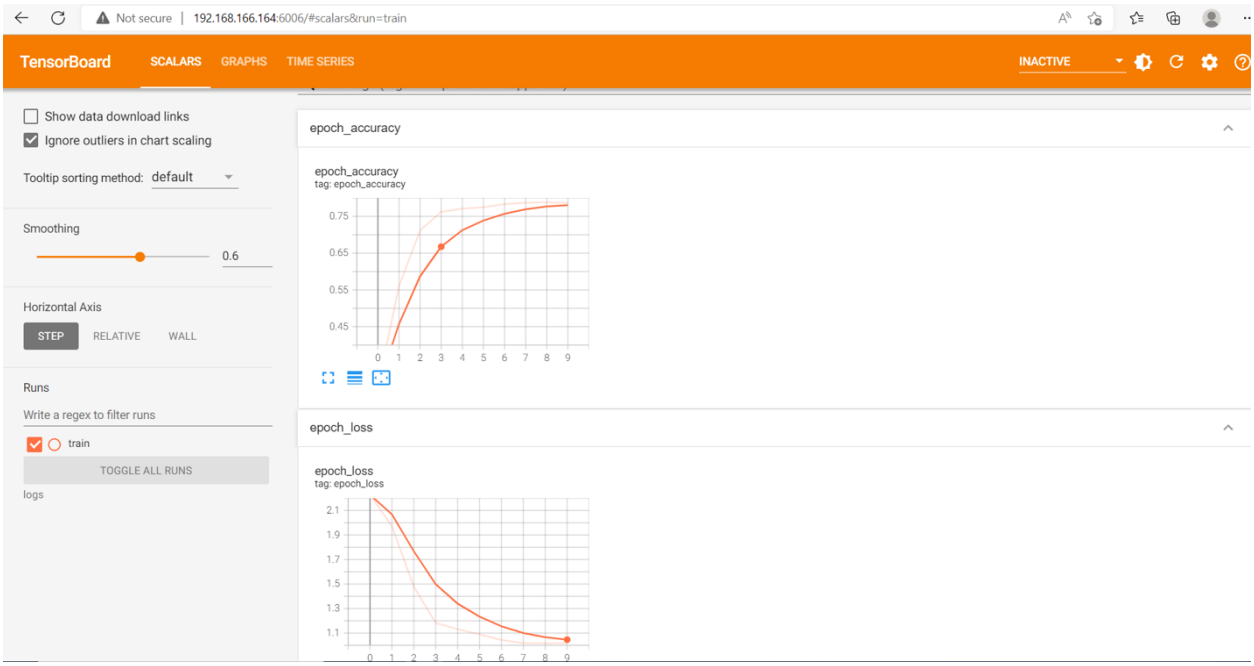
可以通过 AIP 启动 TensorBoard:

```
csub tbjob logs
Job 7455 has been submitted to the default queue [medium].

cread 7455
Messages posted to jobID 7455
POST_TIME: May  2 22:17:47  MESSAGE: /ajj/192.168.10.100/16332/
↪18e19598ce5aa95108fd30a9630e93e9
```

把上面的 MESSAGE 里的 URL 拷贝到 AIP 门户的主机 URL 后，如:

```
https://10.1.22.3/ajj/192.168.10.100/16332/18e19598ce5aa95108fd30a9630e93e9
```

以上的例子中，AIP 门户安装在 10.1.22.3 上。



3.2.2 分布式 MXNet 的作业定义

类似于 TensorFlow，分布式 MXNet 的任务分成 scheduler、server、和 worker。下面是一个包含 1 个 scheduler，2 个 server 和 2 个 worker 的例子:

```
{
  # Mandatory.
  # Job type for mxnet job
  "JobType": "mxnet",

  "Interactive": false,

  # Mandatory if task level command is not specified.
  # Mxnet job command
  # The job level command will be inherited by each task if the task does
  # not specify a command. Otherwise, task level command will overwrite
  # job level command
  "Command":"python image_classification.py --dataset cifar10 --model vgg11 --epochs␣
↪1 --kvstore dist_sync",

  # Optional.
  # A list of environment variables for the mxnet job.
  # The environment variables will be set for each task.
  # Specify NIC for mxnet communication if machines have multiple NICs
  # "Envs": ["DMLC_INTERFACE=enp0s8"],

  # Optional.
  # The shared directory where the output of tasks will be written to.
  # Each task will have a separate output file generated under:
  # <TaskLogDir>/<JobID>.<User>/<Component>.<Index>.<Host>.<PID>
  "TaskLogDir": "log",

  # Optional.
  # Suppress standard output and error of tasks. If TaskLogDir is specified,
  # the standard output and error of tasks will still be written to the
  # directory
  "SuppressTaskOutput": false,

  # Optional.
  # Buffer standard output and error of tasks every second before printing
  # out by task for better readability.
  # If SuppressTaskOutput is set to true, this field is ignored.
  "BufferTaskOutput": false,

  # A list of task specification
  "Tasks":
  [
    {
      # Mandatory.
      # Component name for the task
      "Component":"scheduler",
      "MaxNumTasks": 1
    },
    {
      "Component":"server",
      "MaxNumTasks": 2
    },
    {
```

(下页继续)

(续上页)

```
      "Component":"worker",
      "MaxNumTasks": 2
    }
  ]
}
```

我们加了一个参数“Interactive”：false 让作业在后台运行。

```
aip job run mxnet.json
```

3.2.3 分布式 PyTorch 的作业定义

PyTorch 有 master 和 worker 两类任务，一般 master 是一个任务，worker 则有多个。以下是 PyTorch 的作业定义:

```
{
  # Mandatory.
  # Job type for pytorch job
  "JobType": "pytorch",

  # Mandatory if task level command is not specified.
  # Pytorch job command
  # The job level command will be inherited by each task if the task does
  # not specify a command. Otherwise, task level command will overwrite
  # job level command
  "Command":"python main.py --epochs=3",

  # Optional.
  # A list of environment variables for the pytorch job.
  # The environment variables will be set for each task.
  "Envs": [],

  # Optional.
  # The shared directory where the output of tasks will be written to.
  # Each task will have a separate output file generated under:
  # <TaskLogDir>/<JobID>.<User>/<Component>.<Index>.<Host>.<PID>
  "TaskLogDir": "",

  # Optional.
  # Suppress standard output and error of tasks. If TaskLogDir is specified,
  # the standard output and error of tasks will still be written to the
  # directory
  "SuppressTaskOutput": false,

  # Optional.
  # Buffer standard output and error of tasks every second before printing
  # out by task for better readability.
  # If SuppressTaskOutput is set to true, this field is ignored.
  "BufferTaskOutput": false,

  # A list of task specification
  "Tasks":
  [
    {
      # Mandatory.
```

(下页继续)

(续上页)

```
      # Component name for the task
      "Component":"master",
      "MaxNumTasks": 1
    },
    {
      "Component":"worker",
      "MinNumTasks": 3
    }
  ]
}
```

然后用命令提交作业:

```
aip job run pytorch.json
```

TensorBoard 已经集成到 PyTorch 项目中，PyTorch 也可以使用 TensorBoard 的可视化功能。PyTorch 程序负责保存摘要文件，比如保存到目录 runs，启动 TensorBoard 检查 PyTorch 运行情况的方法与 TensorFlow2.x 的例子相同:

```
csub tbjob runs
```

3.2.4 Intel Caffe MPI 作业

Intel Caffe 是利用 Intel MPI（Message Passing Interface）并行框架的 Caffe 分支。其作业定义较为简单，如以下例子:

```
{
  "Command": "mpirun build/tools/caffe train --solver=cifar10/cifar10_full_solver.
↪prototxt", # Mandatory. Command to run.
  "MinNumSlots": 2, # Minimum replicas of the job to run.
                    # Defaults to 1.
  "MaxNumSlots": 4, # Maximum replicas of the job to run.
}
```

在这个例子中，我们定义最小任务数为 2，最大任务数为 4。调度器至少调度最小任务数，并会最大可能满足最大任务数。

作业运行用命令:

```
aip job run caffe.json
```

3.2.5 分布式 Ray 作业定义

Ray 是一个针对强化学习以及类似学习过程而设计的一个分布式计算框架，囊括强化学习主要计算并分布式化，包含模型训练，模型推理和仿真。Ray 并不替代深度学习框架，而是叠合使用、无缝集成，即同样的强化学习算法既可以使用 TensorFlow 的框架，也可以使用 PyTorch 的框架。

Ray 的优势之一就是能够将一个程序运行在多机器集群中，集群由一个 head 节点和多个 worker 节点组成。Head 节点需要先启动，然后 worker 节点使用 head 节点的地址启动以形成集群。在有 GPU 的情况下，worker 节点利用 GPU 可以提高性能。

SkyForm AIP 可以为 Ray 程序构建满足资源需求的集群，启动程序运行在集群上，并在任务结束后自动销毁集群。

利用 AIP 例子 里 ray 目录下的 ray-run.sh 脚本，以下是 Ray 利用 PyTorch 框架学习 mnist 的作业定义例子:

```
csub -R "2 {span[hosts=1]} 4 {span[ptile=2] rusage[gpu=1]}" -I ray-run.sh train_
↪fashion_mnist_torch.py
```

参数说明:

-R:

定义作业运行的资源需求。Ray 作业包含 head 节点和 worker 节点，节点数量和资源需求都不同，需要使用 SkyForm AIP 多节资源需求。

多节资源需求的语法为：n1{资源需求} n2{资源需求}…

其中 n1，n2 为 CPU 核数。比如例子中 “2 {span[hosts=1]} 4 {span[ptile=2] rusage[gpu=1]}”，表示 head 节点需要 2 个核，分布在一台机器上；worker 节点需要 4 个核，每个节点两个核，同时每个核对应一块 GPU。

-I:

交互式作业。作业标准输出重定向到终端，可选。也可以通过-o 将作业标准输出重定向到文件中。

ray-run.sh:

SkyForm AIP 提供的 Ray 多机分布式计算的 wrapper，负责将 head 节点和 worker 节点在调度器分配的资源上启动，启动 Ray 程序，并监控任务状态和收集资源使用信息。当 Ray 程序运行结束后，销毁 Ray 集群。Ray 多机分布式计算任务必须通过 ray-run.sh 启动。ray-run.sh 根据调度器分配的机器和端口，在启动 Ray 程序前会设置环境变量 “ip_head”，供 Ray 程序初始化时使用，比如：ray.init(address=os.environ[“ip_head”])

Train_fashion_mnist_torch.py:

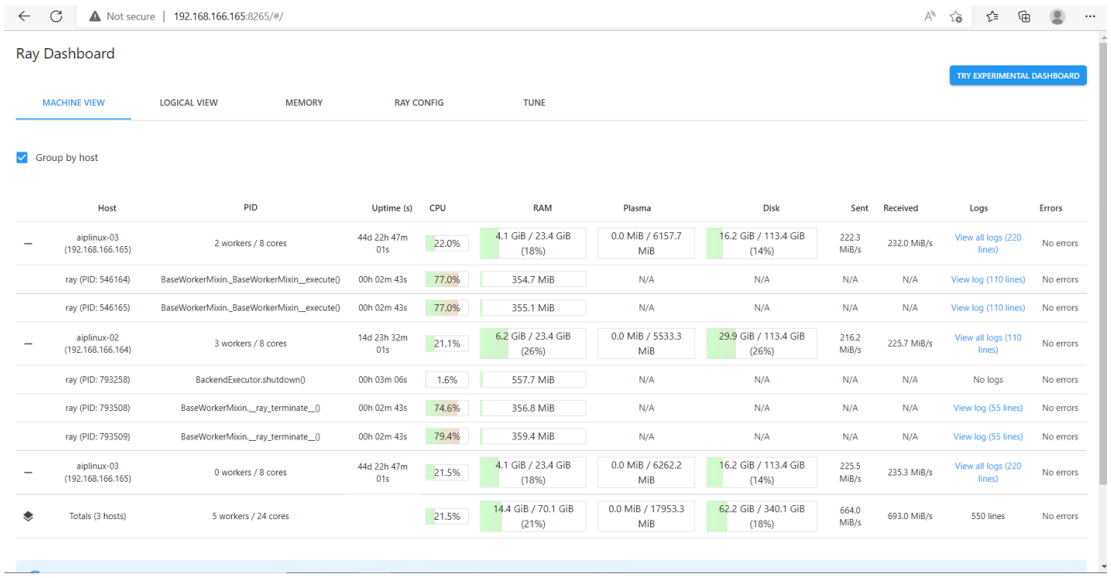
Ray 的 Python 程序，后面可以跟程序参数，程序名和参数都是用户自定义的。

作业屏幕输出例子:

```
2022-07-22 15:36:14,375 INFO trainer.py:243 -- Trainer logs will be logged in: /gpfs-data/cadmin/ray_results/train_2022-07-22_15-36-14
(BaseWorkerMixin pid=536816) 2022-07-22 15:36:27,148    INFO torch.py:347 -- Setting up process group for: env:// [rank=0, world_size=4]
(BaseWorkerMixin pid=536817) 2022-07-22 15:36:27,157    INFO torch.py:347 -- Setting up process group for: env:// [rank=1, world_size=4]
(BaseWorkerMixin pid=536816) 2022-07-22 15:36:27,148    INFO torch.py:347 -- Setting up process group for: env:// [rank=0, world_size=4]
(BaseWorkerMixin pid=536817) 2022-07-22 15:36:27,157    INFO torch.py:347 -- Setting up process group for: env:// [rank=1, world_size=4]
(BaseWorkerMixin pid=366123, ip=192.168.166.163) 2022-07-22 15:36:53,006        INFO torch.py:347 -- Setting up process group for: env:// [rank=3, world_size=4
(BaseWorkerMixin pid=366122, ip=192.168.166.163) 2022-07-22 15:36:53,005        INFO torch.py:347 -- Setting up process group for: env:// [rank=2, world_size=4
2022-07-22 15:36:53,056 INFO trainer.py:249 -- Run results will be logged in: /gpfs-data/cadmin/ray_results/train_2022-07-22_15-36-14/run_001
(BaseWorkerMixin pid=366123, ip=192.168.166.163) 2022-07-22 15:36:56,778        INFO torch.py:98 -- Moving model to device: cpu
(BaseWorkerMixin pid=366123, ip=192.168.166.163) 2022-07-22 15:36:56,778        INFO torch.py:132 -- Wrapping provided model in DDP.
(BaseWorkerMixin pid=366122, ip=192.168.166.163) 2022-07-22 15:36:56,777        INFO torch.py:98 -- Moving model to device: cpu
(BaseWorkerMixin pid=366122, ip=192.168.166.163) 2022-07-22 15:36:56,777        INFO torch.py:132 -- Wrapping provided model in DDP.
(BaseWorkerMixin pid=536816) 2022-07-22 15:36:56,778    INFO torch.py:98 -- Moving model to device: cpu
(BaseWorkerMixin pid=536816) 2022-07-22 15:36:56,779    INFO torch.py:132 -- Wrapping provided model in DDP.
(BaseWorkerMixin pid=536817) 2022-07-22 15:36:56,779    INFO torch.py:98 -- Moving model to device: cpu
(BaseWorkerMixin pid=536817) 2022-07-22 15:36:56,779    INFO torch.py:132 -- Wrapping provided model in DDP.
(BaseWorkerMixin pid=536816) 2022-07-22 15:36:56,778    INFO torch.py:98 -- Moving model to device: cpu
(BaseWorkerMixin pid=536816) 2022-07-22 15:36:56,779    INFO torch.py:132 -- Wrapping provided model in DDP.
(BaseWorkerMixin pid=536817) 2022-07-22 15:36:56,779    INFO torch.py:98 -- Moving model to device: cpu
(BaseWorkerMixin pid=536817) 2022-07-22 15:36:56,779    INFO torch.py:132 -- Wrapping provided model in DDP.
(BaseWorkerMixin pid=366123, ip=192.168.166.163) loss: 2.311164  [    0/15000]
(BaseWorkerMixin pid=366122, ip=192.168.166.163) loss: 2.306050  [    0/15000]
(BaseWorkerMixin pid=536816) loss: 2.304104  [    0/15000]
(BaseWorkerMixin pid=536817) loss: 2.310245  [    0/15000]
(BaseWorkerMixin pid=536816) loss: 2.304104  [    0/15000]
(BaseWorkerMixin pid=536817) loss: 2.310245  [    0/15000]
(BaseWorkerMixin pid=536816) loss: 2.304392  [ 1600/15000]
(BaseWorkerMixin pid=536816) loss: 2.304392  [ 1600/15000]
(BaseWorkerMixin pid=536817) loss: 2.300596  [ 1600/15000]
(BaseWorkerMixin pid=536817) loss: 2.300596  [ 1600/15000]
(BaseWorkerMixin pid=366123, ip=192.168.166.163) loss: 2.303049  [ 1600/15000]
(BaseWorkerMixin pid=366122, ip=192.168.166.163) loss: 2.300127  [ 1600/15000]
(BaseWorkerMixin pid=366123, ip=192.168.166.163) loss: 2.305834  [ 3200/15000]
(BaseWorkerMixin pid=366122, ip=192.168.166.163) loss: 2.285485  [ 3200/15000]
(BaseWorkerMixin pid=536816) loss: 2.297839  [ 3200/15000]
(BaseWorkerMixin pid=536816) loss: 2.297839  [ 3200/15000]
(BaseWorkerMixin pid=536817) loss: 2.294649  [ 3200/15000]
(BaseWorkerMixin pid=536817) loss: 2.294649  [ 3200/15000]
(BaseWorkerMixin pid=366123, ip=192.168.166.163) loss: 2.297828  [ 4800/15000]
(BaseWorkerMixin pid=366122, ip=192.168.166.163) loss: 2.287170  [ 4800/15000]
(BaseWorkerMixin pid=536816) loss: 2.298700  [ 4800/15000]
(BaseWorkerMixin pid=536816) loss: 2.298700  [ 4800/15000]
```

Ray 提供内置的 dashboard，启动在 head 节点，默认端口 8265。

访问 dashboard：http://192.168.166.165:8265



3.2.6 Spark 集群作业

Apache Spark 是高速通用集群计算框架，广泛应用与大数据分析等领域。SkyForm AIP 可以根据用户的要求动态为作业生成一个 Spark 集群。作业结束时 Spark 集群则自行消失而释放资源。

准备 Scala 应用例子。使用工具 sbt 来创建应用:

```
curl https://bintray.com/sbt/rpm/rpm | \
tee /etc/yum.repos.d/bintray-sbt-rpm.repo
yum install sbt
```

准备应用 JAR 包:

```
sbt package
...
[info] Packaging {..}/{..}/target/scala-2.12/simple-project_2.12-1.0.jar
```

准备作业定义。以下例子里用一个 master 和 2 各 Slave。其中

1. 环境变量 SPARK_HOME 必须设。
2. 最后的一个 Component 为客户程序。

```
{
  # Mandatory.
  # Job type for Spark job
  "JobType": "spark",

  # Mandatory.
  # A list of environment variables for the Spark job.
  # The environment variables will be set for each task.
  # The environment SPARK_HOME is mandatory.
  "Envs": ["SPARK_HOME=/opt/skyformai_shared/spark"],

  # Optional.
  # The shared directory where the output of tasks will be written to.
  # Each task will have a separate output file generated under:
```

(下页继续)

(续上页)

```
  # <TaskLogDir>/<JobID>.<User>/<Component>.<Index>.<Host>.<PID>
  "TaskLogDir": "",

  # Optional.
  # Suppress standard output and error of tasks. If TaskLogDir is specified,
  # the standard output and error of tasks will still be written to the
  # directory
  "SuppressTaskOutput": false,

  # Optional.
  # Buffer standard output and error of tasks every second before printing
  # out by task for better readability.
  # If SuppressTaskOutput is set to true, this field is ignored.
  "BufferTaskOutput": false,

  # A list of task specification
  "Tasks":
  [
    {
      # Mandatory.
      # Component name for the task
      "Component":"master",
      "Command":"sparkcluster",
      "MaxNumTasks": 1
    },
    {
      # Mandatory.
      "Component":"worker",
      "Command":"sparkcluster",
      "MinNumTasks": 2
    },
    {
      # Mandatory.
      "Component":"client",
      "Command":"spark-submit --class SimpleApp target/scala-2.12/simple-project_2.12-
↪1.0.jar spark.json",
      "MaxNumTasks": 1
    }
  ]
}
```

提交作业:

```
aip job run spark.json
```

若是单机 Spark 应用，则可用简单的作业定义:

```
{"Command": "/opt/skyformai_shared/spark/bin/spark-submit --class SimpleApp target/
↪scala-2.12/simple-project_2.12-1.0.jar spark.json"}
```

3.2.7 Megatron-deepspeed

微软通过将其 DeepSpeed 库集成到 NVIDIA 的 Megatron-LM 框架中，开发出了 Megatron-DeepSpeed。

DeepSpeed 是微软的优化库，旨在简化和增强分布式训练和推理。DeepSpeed 引入了一系列优化措施，简化了流程，使其更加高效。

Megatron-LM 是 NVIDIA 的大型强大转换器。它可以处理海量模型和复杂的深度学习任务，是 DeepSpeed 带来进步的理想起点。

Megatron-DeepSpeed 的与众不同之处在于它全面支持一系列功能，从混合专家模型训练到课程学习。这使得它成为应对深度学习领域各种挑战的多功能工具。

使用 Megatron-DeepSpeed，您可以以前所未有的效率和规模训练更大规模的模型。

以下的作业启动程序分两个部分：把 AIP 在多台主机上为作业分配的 CPU、GPU、和端口转换成 Megatron-DeepSpeed 需要的环境变量；利用 AIP 的远程任务启动程序启动分布式任务。

mtds.sh:

```
#!/bin/bash
# AIP为作业设置了环境变量CB_ALLOCATION，描述资源分配详情
# 1. 准备环境变量，这部分的代码对所有分布式训练的作业基本一致，有通用性
nodes_array=($(echo $CB_ALLOCATION|python -c 'import json, sys; parsed = json.
↪load(sys.stdin); hosts = [d["Hosts"] for d in parsed]; print([dd["Name"] for dd in␣
↪hosts[0]])'|sed "s/'//g;s/\[//g;s/\]//g;s/,//g"))
ports_array=($(echo $CB_ALLOCATION|python -c 'import json, sys; parsed = json.
↪load(sys.stdin); hosts = [d["Hosts"] for d in parsed]; print([dd["Port"] for dd in␣
↪hosts[0]])'|sed "s/'//g;s/\[//g;s/\]//g;s/,//g"))

nodes_array=($CB_MCPU_HOSTS)
slots=0
nnodes=0
nodes=""
for (( i=0; i<${#nodes_array[*]}; i++ ));
do
  nodes=$nodes" "${nodes_array[$i]}
  nnodes=$(($nnodes+1))
  i=$((i+1))
  slots=$(($slots+${nodes_array[$i]}))
done
nproc_per_node=$(($slots / $nnodes ))

# PROC_NODE_LIST: 分配的主机名阵列，例子：node2 node2 node1 node1
export PROC_NODE_LIST="${nodes_array[@]}"
# PROC_PORT_LIST: 分配的端口阵列，例子：16331 16332 16331 16332
export PROC_PORT_LIST="${ports_array[@]}"
# NPROC_PER_NODE: 每台主机上的任务数，例子：2
export NPROC_PER_NODE=$nproc_per_node
# NPROCS: 总任务数，例子：4
export NPROCS=$slots
# NODE_LIST: 主机列表，例子：node2 node1
export NODE_LIST=$nodes
# NUM_NODES: 总主机数
export NUM_NODES=$(echo $nodes | wc -w)

# 2. 启动任务

n=0
```

(下页继续)

(续上页)

```
for host in $NODE_LIST; do
  runtask -z $host torchrun \
   --nproc_per_node=$NPROC_PER_NODE \
   --nnodes=$NUM_NODES \
   --node_rank=$n \
   --master_addr=$PROC_NODE_LIST[0] \
   --master_port=$PROC_PORT_LIST[0] \
   $@ &
  n=$((n+1))
done
wait
```

备注： AIP 分配 GPU 后，根据环境变量 GPU_ALLOCATION 的值，在作业的第一台主机上自动设置环境变量 CUDA_VISIBLE_DEVICES，通过 runtask 启动任务时，runtask 会解析 GPU_ALLOCATION 的内容，在运行任务前，设置相应的 CUDA_VISIBLE_DEVICES 值。所以以上的作业启动器 mtds.sh 里没有处理 GPU 分配的逻辑。

提交作业例子:

```
csub -n 4 -R "rusage[gpu=1] span[ptile=2]" -o %J.out ./mtds.sh pretrain_gpt.py \
  --num-layers 24 --hidden-size 1024 \
  --num-attention-heads 16 --micro-batch-size 8 --global-batch-size 64 \
  --seq-length 1024 --max-position-embeddings 1024 --train-iters 500 \
  --lr-decay-iters 320000 --save /ai/Megatron-DeepSpeed/checkpoints/gpt2 \
  --load /ai/Megatron-DeepSpeed/checkpoints/gpt2 \
  --data-path /ai/Megatron-DeepSpeed/data/meg-gpt2_text_document  \
  --vocab-file /ai/Megatron-DeepSpeed/data/gpt2-vocab.json \
  --merge-file /ai/Megatron-DeepSpeed/data/gpt2-merges.txt \
  --data-impl mmap --split 949,50,1 --distributed-backend nccl \
  --lr 0.00015 --lr-decay-style cosine --min-lr 1.0e-5 \
  --weight-decay 1e-2 --clip-grad 1.0 --lr-warmup-fraction .01 \
  --checkpoint-activations --log-interval 100 --save-interval 100 \
  --eval-interval 100 --eval-iters 10 --fp16
```

以上的例子的 csub 参数：

- -n：4 个子任务
- -R “rusage[gpu=1] span[ptile=2]”：每个任务 1 个 GPU，每台主机 2 个任务

使用容器

如果使用容器跑例子中的作业，则需要定义容器镜像，以及 docker run 的命令参数，其他命令不变。例子:

```
# CB_TASK_DOCKER_IMAG 指向容器镜像名
export CB_TASK_DOCKER_IMAGE=megatron-deepspeed:pytorch2.0.1-cuda11.7-cudnn8
# CB_TASK_DOCKER_OPTIONS 定义挂载到容器里的目录和传进容器的环境变量
export CB_TASK_DOCKER_OPTIONS="-v /share/home/cadmin/ai:/ai
  -e NCCL_SOCKET_IFNAME=enp1s0f0 -e GLOO_SOCKET_IFNAME=enp1s0f0
  -w /ai/Megatron-DeepSpeed -e HOME=/tmp"

# 作业提交与裸金属一样
csub -n 4 -R "rusage[gpu=1] span[ptile=2]" -o %J.out ./mtds.sh pretrain_gpt.py \
  --num-layers 24 --hidden-size 1024 \
```

(下页继续)

(续上页)

```
  --num-attention-heads 16 --micro-batch-size 8 --global-batch-size 64 \
  --seq-length 1024 --max-position-embeddings 1024 --train-iters 500 \
  --lr-decay-iters 320000 --save /ai/Megatron-DeepSpeed/checkpoints/gpt2 \
  --load /ai/Megatron-DeepSpeed/checkpoints/gpt2 \
  --data-path /ai/Megatron-DeepSpeed/data/meg-gpt2_text_document  \
  --vocab-file /ai/Megatron-DeepSpeed/data/gpt2-vocab.json \
  --merge-file /ai/Megatron-DeepSpeed/data/gpt2-merges.txt \
  --data-impl mmap --split 949,50,1 --distributed-backend nccl \
  --lr 0.00015 --lr-decay-style cosine --min-lr 1.0e-5 \
  --weight-decay 1e-2 --clip-grad 1.0 --lr-warmup-fraction .01 \
  --checkpoint-activations --log-interval 100 --save-interval 100 \
  --eval-interval 100 --eval-iters 10 --fp16
```

3.3 GPU 作业

3.3.1 GPU 资源自动检测

AIP 会自动探测集群节点上已经安装的 GPU 和 GPU 的一些常用参数，如型号、内存、温度等。

查看集群中的 GPU 可以使用命令 chinfo -g

```
chinfo -g
```

命令输出:

```
HOST_NAME    ID MODEL_NAME            TotMem FreeMem Temp GUT  MUT
win10         0 NVIDIA GeForce RTX 2    5934    5915   22 0.01 0.00
              1 NVIDIA GeForce GT 10    2001    1991   17 0.02 0.00
```

各种不同型号的 GPU 在 AIP 内部对应于不同的资源名。如以上的两种 GPU 对应于两种资源，这些资源名可以用于作业资源需求参数:

```
cinfo | grep gpu
```

命令输出:

```
gpu             Number   Dec      Yes      Yes   Number of GPUs
ngpus           Number   Dec       No       No   Number of GPUs
gpuGeForceGT    Number   Dec      Yes      Yes   GeForce GT 1030
gpuGeForceRTX   Number   Dec      Yes      Yes   GeForce RTX 2070
```

以上的例子中，gpu 是任何一种型号的 GPU，gpuGeForceRTX 为集群中的 “GeForce RTX 2080”，gpuGeForceGT 为集群中的 GeForce GT 1030。

3.3.2 提交单机 GPU 作业

单机 GPU 作业请求的 GPU 数为整个作业使用的 GPU 卡数。

```
csub -n 4 -R "rusage[gpu=2] span[hosts=1]" myjob
```

以上的例子中，-R 定义作业资源需求，作业需要 4 个 CPU 核，但必须在同一个节点上，整个作业需要 2 个任意型号的 GPU。

```
csub -n 2 -R "rusage[gpuGeForceRTX=1] span[hosts=1]" myjob
```

以上例子中，作业需要一个 GeForce RTX 2080 GPU，两个 CPU 核，在同一个节点上。

备注： 单节点的作业必须指定 -R span[hosts=1]**

3.3.3 多节点 GPU 作业

多节点的 GPU 请求需要把作业指定到各个任务。提交作业是不用 span[hosts=1]，请求的 GPU 数是每个任务的 GPU 所用的数。例子：

```
csub -n 16 -R rusage[gpu=0.5] myjob
```

以上的例子中，作业需要 16 个 CPU 核，8 块 GPU 卡，平均 2 个 CPU 核 1 块卡。若第一个节点分配 4 个核，第二个节点分配 12 个核，则两个节点上所需的 GPU 卡数为 2 和 6。

```
csub -n 4,8 -R rusage[gpu=1] myjob
```

以上例子中，作业可以最少使用 4 个 CPU 核，4 个 GPU，最多可用 8 个 CPU 核，8 个 GPU（即每个 CPU 核对应一个 GPU 卡）。AIP 会根据系统资源的情况灵活分配。

3.3.4 GPU 资源使用限制

系统管理员可以在 cb.yaml 中的 general 中配置 cgroup: gpu 来控制作业只能使用被分配的 GPU。例子：

```
general:
  cgroup: gpu acct
```

备注： cgroup: acct 用于确保 AIP 能够跟踪作业的所有进程。

修改 cb.yaml 后，重启调度器：

```
aip admin rcs
```

容器作业 GPU 的使用缺省就被限制的，不依赖于 cb.yaml 中 cgroup 的配置。

3.3.5 VGPU：多任务共享 GPU

对于较强算力的 GPU，单个作业可能不能用满其所有算力资源。可以通过 GPU 共享来提高 GPU 的资源利用率。

当提交的作业与其他作业共享 GPU 时，作业请求小于 1 个 GPU:

```
csub -R rusage[gpu=0.25] myjob
```

以上的例子中，作业请求 0.25 个 GPU，剩下的 0.75 个 GPU 可与其他几个作业作业共享，如与一个 0.5 GPU 的作业和另一个 0.25 GPU 的作业共享。

AIP 提供两种形式的多任务共享 GPU 资源控制方式:

1. 分时方式，即时间分片

每个作业都可以使用整个 GPU 卡，多个作业按调度器分配的额度（如 0.25）与其他作业分时使用同一个 GPU。分时由 GPU 自己控制，类似于 CPU 一个核上同时跑多个进程的方式。

这种方式是 AIP 缺省的方式。

2. 物理资源限制

每个作业可以使用的 GPU 的算力和显存得到强行的控制。

这种方式需要在相应的队列配置中设置参数 “vgpu: y”，具体参考*cb.yaml* 。

队列中配置了参数 vgpu，并重启调度器后，可以用 cqueues -l 队列名来查看，输出中有 vGPU is enabled:

```
cqueue -l gpu

QUEUE: gpu
  -- GPU queue.

PARAMETERS/STATISTICS
PRIO NICE     STATUS       MAX JL/U JL/P JL/H NJOBS  PEND  RUN  SSUSP USUSP  RSV ␣
↪PJOBS
   3   0    Open:Active     -    -    -    -     0     0     0     0     0    0       0

Schedule delay for a new job is 0 seconds
Interval for a host to accept two jobs is 0 seconds
Cgroup: gpu acct
vGPU is enabled
```

以下例子中 GPU 共有 6G 显存，当作业申请 0.25 个 GPU 时，在作业里只能看到 1.5G 的显存:

```
csub -R rusage[gpu=0.25]  -I nvidia-smi
Job 415 has been submitted to the default queue [medium].
Job 415 is waiting to be started...
Job 415 has started on host amdlinux.
Tue Aug 26 19:14:02 2025
+-----------------------------------------------------------------------------------------
↪----+
| NVIDIA-SMI 580.65.06              Driver Version: 580.65.06      CUDA Version: 13.0␣
↪    |
+-----------------------------------------+------------------------+------------------
↪----+
```

(下页继续)

(续上页)

```
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.␣
↪ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util ␣
↪Compute M. |
|                                         |                        |              ␣
↪MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 2060        On  |   00000000:08:00.0  On |                  ␣
↪N/A |
|  0%   49C    P8             12W /  170W |       0MiB /   1536MiB |      0%      ␣
↪Default |
|                                         |                        |                  ␣
↪N/A |
+-----------------------------------------+------------------------+------------------
↪----+

+---------------------------------------------------------------------------------------
↪----+
| Processes:                                                                            ␣
↪    |
|  GPU   GI   CI              PID   Type   Process name                          GPU␣
↪Memory |
|        ID   ID                                                                  Usage ␣
↪    |
|=======================================================================================|
|  No running processes found                                                           ␣
↪    |
+---------------------------------------------------------------------------------------
↪----+
```

同样的方法可以用在容器作业中（容器镜像 ubuntu，使用 NVIDIA Container Toolkit）：

```
csub -R rusage[gpu=0.5] -di ubuntu -do "--runtime=nvidia" -I nvidia-smi
Job 414 has been submitted to the default queue [medium].
Job 414 is waiting to be started...
Job 414 has started on host amdlinux.
Tue Aug 26 19:12:30 2025
+---------------------------------------------------------------------------------------
↪----+
| NVIDIA-SMI 580.65.06              Driver Version: 580.65.06      CUDA Version: 13.0␣
↪    |
+-----------------------------------------+------------------------+------------------
↪----+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.␣
↪ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util ␣
↪Compute M. |
|                                         |                        |              ␣
↪MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 2060        On  |   00000000:08:00.0  On |                  ␣
↪N/A |
|  0%   49C    P8             12W /  170W |       0MiB /   3072MiB |      0%      ␣
↪Default |
|                                         |                        |                  ␣
↪N/A |
```

(下页继续)

(续上页)

```
+-----------------------------------------+-----------------------+----------------------
↪----+

+---------------------------------------------------------------------------------------
↪----+
| Processes:                                                                            ␣
↪    |
|  GPU   GI   CI              PID   Type   Process name                            GPU␣
↪Memory |
|        ID   ID                                                                     Usage ␣
↪    |
|=======================================================================================|
|  No running processes found                                                           ␣
↪    |
+---------------------------------------------------------------------------------------
↪----+
```

3.4 NVIDIA 多实例 GPU 调度

NVIDIA 的 A100/A800 和 H100/H800 支持把一块 GPU 最多分成 7 个实例。实例大小有不同的组合。详细参见：https://docs.nvidia.com/datacenter/tesla/mig-user-guide/

AIP 自动检测 MIG 是否已经配置，然后把每个 MIG 作为一个 GPU 来调度。

观察可用 GPU MIG：

```
chosts -s
```

命令输出：

```
RESOURCE                 USABLE       RESERVED       LOCALE
gpuMIG1g5gb            7.000000            0.0       linux7
gpuMIG2g10gb           3.000000            0.0       linux7
gpuMIG3g20gb           2.000000            0.0       linux7
gpuMIG4g20gb           1.000000            0.0       linux
gpuMIG1c3g20gb         6.000000            0.0       linux
```

3.4.1 提交作业

验证 MIG 分配情况的作业例子。用环境变量查看作业环境中 CUDA_VISIBLE_DEVICES 的设置：

```
csub -I -R rusage[gpuMIG1g5gb=1] env | grep CUDA
```

命令输出：

```
Job 3390 is waiting to be started...
Job 3390 has started on host linux7.
CUDA_VISIBLE_DEVICES=MIG-30ff4a58-8039-5b2c-9623-241c3dbf8689
```

以上的例子展示 AIP 给作业分配的 MIG。一个作业也可请求多个 MIG，如: -R rusage[gpuMIG1g5gb=4]。

3.5 MPI 作业

运行 MPI 作业时，节点与 CPU 应由 SkyForm AIP 调度获得，而不是在命令行中指定。当节点工作不正常或节点上 CPU 被其他作业占用时，作业会等待而不能及时运行。另外，由 SkyForm AIP 调度的 MPI 作业，当节点失败时，作业可以有选择地自动在其他节点上重启。所以，除非万不得已，MPI 运行的节点应该由调度器来动态决定，而不是作业提交命令行中指定，或者在 mpirun 的命令行中指定。

可喜的是，很多常用的 MPI 能够自动识别 AIP 调度所分配的节点，用户的 mpirun/mpiexec 命令行可以变的很简单。

通过 AIP 运行 MPI 可以无需设置免密 ssh，确保系统安全并省去不必要的设置和排错。除此之外，对于跨节点的 MPI 作业，AIP 随时监控远程 MPI 任务的进程以及 MPI 任务进程的资源（CPU、内存、IO、文件读写、GPU 等）使用情况，以便执行系统设置的资源使用上限。当用户需要杀掉 MPI 作业时，AIP 也会保证自动清理远程 MPI 任务的所有进程。

AIP 包含一个名为 rsh 的远程 MPI 任务分发和控制工具，该工具只能在 AIP 的作业中运行。MPI 启动时只要定义使用 rsh（而不是缺省的 ssh）作为远程任务分发工具，就可以实现 MPI 进程在 AIP 里的资源使用监控和结束是的清理功能。

下表列出各种 MPI 与 AIP 的集成支持：

MPI	MPI 自动识别 AIP 分配的节点和 CPU	不依赖免密 ssh	** 监控并在杀作业是自动清理所有 MPI 远程进程 **
Intel MPI	支持	支持	支持
MPICH	支持	支持	支持
MVAPICH	支持	支持	支持
OpenMPI	MPI 启动脚本	支持	支持
P latform/HP/IBM MPI	MPI 启动脚本	支持	支持

很多商业软件如 Fluent、Abaqus、StarCCM+ 等内含 Intel MPI，求解程序在 AIP 里可以实现深度集成。

提交 MPI 作业前先设置 MPI 环境，如使用 module load 或 source 环境文件。在提交前设置的环境会自动复制到作业运行的环境中。然后在作业提交命令中指定需要的资源。作业提交时的资源请求描述常用的两个参数

- -n 指定 CPU 核数
- -R span[ptile=x] 指定每个节点上的核数

  例子：提交一个需要总数为 128 个核，每个节点上 32 个核的 MPI 作业（LSF 兼容命令，下同）。

```
module load mpi/mpich-x86_64
bsub -n 128 -R span[ptile=32] mpirun myprogram
```

以上例子中，mpirun 的命令行中无需定义-hostfile、-machinefile、-np 等不用调度器时必须定义的参数，MPI 可以自动识别 AIP 为作业分配的节点和 CPU。这样就不需要编写 MPI 作业提交命令。只要熟悉命令行就可以灵活的直接提交 MPI。

-R 参数定义 AIP 作业所需的资源，-R 参数中的其他语法参考 man csub 里-R 的说明。

备注： -R span[ptile=32] 不是必须的，若不指定，AIP 会先把一个节点的 CPU 分配完，然后再分配其他节点上的 CPU。如 bsub -n 64 的作业在每个节点含 56 个核的系统中会在第一个节点上分配 56 个核，在第二个节点上分配 8 个核。加上-R span[ptile=32] 后就能保证两个节点上各分配 32 个核。

当作业运行后，“cjobs -l 作业号”可以看到 mpirun 通过 AIP rsh 启动远程任务的命令，如下方红框所示。

```
Job <437>, User <cadmin>, Project <default>, Status <RUN>, Queue <medium>, Comm
                     and <mpirun ./mpitest>
Thu Aug  8 08:42:28: Submitted from host <amaster>, CWD <$HOME>, 6 Processors R
                     equested, Requested Resources <6{span[ptile=2]}>;
Thu Aug  8 08:42:28: Started on 6 Hosts/Processors <2*an0003> <2*an0002> <2*an0
                     001>, Execution Home </home/cadmin>, Execution CWD </home/
                     cadmin>;
Thu Aug  8 08:42:38: Task information.
                     Component: default
                       Task: 127474.1
                         Command: /usr/lib64/mpich/bin/hydra_pmi_proxy --contro
                     l-port an0003:32913 --rmk lsf --launcher lsf --demux poll
                     --pgid 0 --retries 10 --usize -2 --proxy-id 1
                         Host: an0002
                         Status: RUN
                         Start time: Thu Aug  8 08:42:28
                         MEM: 6 Mbytes;  SWAP: 102 Mbytes;  NTHREAD: 3
                         PGID: 79171;  PIDs: 79171
                         PGID: 79172;  PIDs: 79172
                         PGID: 79173;  PIDs: 79173
                       Task: 127475.1
                         Command: /usr/lib64/mpich/bin/hydra_pmi_proxy --contro
                     l-port an0003:32913 --rmk lsf --launcher lsf --demux poll
                     --pgid 0 --retries 10 --usize -2 --proxy-id 2
                         Host: an0001
                         Status: RUN
                         Start time: Thu Aug  8 08:42:28
                         MEM: 6 Mbytes;  SWAP: 102 Mbytes;  NTHREAD: 3
                         PGID: 82878;  PIDs: 82878
                         PGID: 82879;  PIDs: 82879
                         PGID: 82880;  PIDs: 82880
Thu Aug  8 08:42:38: Total resource usage collected.
                     MEM: 37 Mbytes;  SWAP: 581 Mbytes;  NTHREAD: 16
                     PGID: 127468;  PIDs: 127468 127471 127472
                     PGID: 127473;  PIDs: 127473
                     PGID: 127474;  PIDs: 127474 127483
                     PGID: 127475;  PIDs: 127475 127481
                     PGID: 127476;  PIDs: 127476
                     PGID: 127477;  PIDs: 127477
                     PGID: 79171;  PIDs: 79171
                     PGID: 79172;  PIDs: 79172
                     PGID: 79173;  PIDs: 79173
                     PGID: 82878;  PIDs: 82878
                     PGID: 82879;  PIDs: 82879
                     PGID: 82880;  PIDs: 82880


 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

              gpu
 loadSched     -
 loadStop      -

 RESOURCE REQUIREMENT DETAILS:
 Combined: 6{span[ptile=2]}
 Effective: 6{span[ptile=2]}

 RESOURCE ALLOCATION:
[{"ResSpec":"span[ptile=2]","MinSlots":6,"MaxSlots":6, "Hosts":[{"Name":"an0003","Port":16331},{
"Name":"an0003","Port":16332},{"Name":"an0002","Port":16331},{"Name":"an0002","Port":16332},{"Na
me":"an0001","Port":16331},{"Name":"an0001","Port":16332}]}]
```

另外，通过命令“aip j i -p 作业号”可以看到作业在每个节点给上每个进程的资源使用情况：

3.5.1 Intel MPI

Intel MPI 使用内置的 Hydra 分发和管理远程 MPI 任务。Hydra 能自动识别 AIP 为作业分配的节点和 CPU 核。

备注： 由于 Intel MPI 中 Hydra 的调度器集成 bug，对于超过 1024 个核的 MPI，需要使用 AIP 脚本 impi-mpirun 来替代 mpirun。

例子：

```
bsub -n 1024 impi-mpirun myprogram
```

Intel MPI 2018 和以前的版本

Intel MPI 2018 和以前的版本自动识别到 AIP 环境后，会自动使用 AIP 的远程任务分发工具（rsh）分发和管理远程 MPI 任务。为确认所有版本使用 AIP 的 rsh 为不是其他的工具，在 mpirun 命令行里定义 rsh：

```
bsub -n 128 -R span[ptile=32] mpirun -bootstrap rsh myprogram
```

或者直接使用 AIP 脚本 impi-mpirun 代替 mpirun:

```
bsub -n 128 -R span[ptile=32] impi-mpirun myprogram
```

Intel MPI 2019、2020 和 Intel oneAPI

Intel MPI 2019 以后的版本指定远程进程控制工具的参数略有变化，为-launcher。在 mpirun 命令行里定义 rsh：

```
bsub -n 128 -R span[ptile=32] mpirun -launcher rsh myprogram
```

或者直接使用 AIP 脚本 impi-mpirun 代替 mpirun:

```
bsub -n 128 -R span[ptile=32] impi-mpirun myprogram
```

3.5.2 MPICH 和 MVAPICH

备注： 由于 Hydra 的调度器集成 bug，对于超过 1024 个核的 MPI，需要使用 AIP 脚本 impi-mpirun 来替代 mpirun。例子：csub -n 1024 impi-mpirun myprogram

---

与 Intel MPI 2019 一样，使用-launcher 指定 rsh。

```
bsub -n 128 -R span[ptile=32] mpirun -launcher rsh myprogram
```

或者直接使用 AIP 脚本 impi-mpirun 代替 mpirun:

```
bsub -n 128 -R span[ptile=32] impi-mpirun myprogram
```

3.5.3 OpenMPI 和 Platform/HP/IBM MPI

这几种 MPI 需要编写作业脚本，把 AIP 分配的节点名转换生成 host 文件，并告诉 mpirun 总核数。以下的例子为名字是 ompi 的脚本。这个脚本可用于 OpenMPI、Platform/HP/IBM MPI。

```
#!/bin/bash
# For running OpenMPI, IBM MPI in SkyForm AIP
WORKDIR=`dirname ${CB_JOBFILENAME}`

rm -f ${WORKDIR}/mpi_machines.* # cleanup previous mpi_machines files

MACHFILE=${WORKDIR}/mpi_machines.${CB_JOBID}

# Make the Open MPI machine file by the using AIP tool
hostlist -o > ${MACHFILE}

mpirun -np $CB_DJOB_NUMPROC -machinefile ${MACHFILE} \
--mca plm_rsh_agent rsh $*
```

以上脚本中，红色部分是指定总核数，蓝色部分是指定运行节点，橙色部分是指定使用 AIP 的 rsh。

提交作业命令：

```
bsub -n 128 -R span[ptile=32] ./ompi myprogram
```

AIP 里面包含了脚本 ompi-mpirun 实现以上的功能，可以用于作业提交：

```
bsub -n 128 -R span[ptile=32] ompi-mpirun myprogram
```

3.5.4 MPI 和 OpenMP 混合作业

若 MPI 的任务进程是多线程的 OpenMP，提交作业时需要定义 3 个参数

（1）-n m 参数制定作业总 CPU 核数，m 值等于：

节点数 * 每节点任务数 * 每任务 CPU 核数

（2）-R spen[ptile=x] 其中 x 为每个节点上的分配的核数，等于：

每节点任务数 * 每任务 CPU 核数

（3）-ENV OMP_NUM_THREADS=y 其中 y 值等于：每任务 CPU 核数

例子：

```
bsub -n 128 -ENV OMP_NUM_THREADS=8 -R span[ptile=32] ompi-mpirun ./myprog
```

这个 OpenMPI 作业一共用 4 个节点，每个节点上 4 个任务，每个任务 8 个核。

3.5.5 使用 SLURM 兼容命令提交 MPI 作业

SkyForm AIP 提供与 SLURM 兼容的作业提交命令，支持一部分 SLURM 常用参数，例子：

```
sbatch
#!/bin/bash
#SBATCH --ntasks-per-node=4
#SBATCH --ntasks=16
#SBATCH --cpus-per-task=8
ompi-mpirun ./myprog
```

```
Submitted batch job 1237
[root@linux7 skyformai]# cjobs -l 1237

Job <1237>, User <root>, Project <default>, Status <DONE>, Queue <medium>, Job
                     Priority <50>, Command <#!/bin/bash;ompi-mpirun aaa>
Wed Aug 18 14:50:14: Submitted from host <linux7>, CWD <$HOME/skyformai>, 128 P
                     rocessors Requested, Requested Resources <span[ptile=32]>;


 ACTUAL_RUN_LIMIT
 120.0 min
Wed Aug 18 14:50:14: Started on 128 Hosts/Processors <32*node00010> <32*node000
                     09> <32*node00008> <32*node00007>, Execution Home </root>,
                     Execution CWD </root/skyformai>;
Wed Aug 18 14:50:14: Done successfully. The CPU time used is 0.0 seconds. The m
                     aximum memory used is 14 Mbytes. The average memory used i
                     s 14 Mbytes.
```

3.6 Docker 容器作业

为了安全，AIP 运行容器作业是使用与作业提交用户相同的 uid 和 gid，这样即使容器挂在了主机上的文件夹，容器里的用户也不可逾越文件访问权限。AIP 提供了部分命令替代少量的 docker 命令。

若用户需要自己制作镜像，则需要在自己的有 root 权限的服务器或虚拟机中执行 docker 命令。

3.6.1 提交 Docker 容器作业

提交容器作业时用户需要指定容器镜像名，容器镜像名可以用环境变量 CB_DOCKER_IMAGE 或者 csub 的参数-di。例子：

```
csub -I -di centos:6 cat /etc/redhat-release
```

命令输出：

```
Job 4354 has been submitted to the default queue [medium].
Job 4354 is waiting to be started...
Job 4354 has started on host linux7.
CentOS release 6.10 (Final)
```

在运行 Docker 作业时，容器的 CPU、GPU、内存等资源都由调度器根据资源调度来设定。用户可以在容器运行时看到所有容器进程和资源使用情况。命令 cjobs -l 显示容器 id 和所有容器中的进程号。aip j i -p 显示所有容器进程的详细资源使用。

容器作业可以使用所有作业命令对其操作，包括监控所有属于作业的容器进程的资源使用、暂停和恢复容器进程。杀掉作业会杀掉容器作业的所有进程。

3.6.2 在运行中的 Docker 容器作业中执行命令

如果容器作业可以远程访问，用户可以用以下命令模仿 docker exec 命令对容器作业进行操作，只是命令暂不支持 docker exec 的可选参数。例子：

```
cdexe 4354 bash
```

命令输出：

```
bash-4.2$ id
uid=997 gid=1006 groups=1006
bash-4.2$ exit
exit
```

3.6.3 私有容器镜像库登录

容器的镜像可以从任何仓库中自动下拉（缺省为 docker.io)。如果镜像为私有，AIP 支持使用与 HPC 环境共享用户认证的镜像库中下拉镜像。运行的步骤为：

1. 在 AIP 中注册密码，只需注册一次。如果密码修改了测序重新注册

```
cpasswd -p mypassword
```

2. 登录到镜像库：

```
cdlogin harbor:443
```

以上的例子 harbor 为本地镜像库。该镜像库已与本地 LDAP 服务器集成。

该命令也可用于登录到第三方的镜像库中。命令句法为：cdlogin -u 用户名 -p 密码服务器。其中“-p 密码”若不在命令行中，命令会提示输入密码：

```
cdlogin -u user1 harbor:443
```

```
Input Password:
Login Succeeded
```

3. 提交作业

```
csub -I -di harbor:443/first/centos7 id
```

以上 first/centos7 为私有镜像。

3.6.4 从第三方的镜像库中引入镜像上传到本地镜像库

AIP 提供的 cdimport 用于实现 docker pull; docker tag; docker push 命令组合。例子如下：

```
cdimport docker.io/mstormo/suse harbor:443/myproj/suse
```

其中 myproj 是本地镜像库 harbor 中已有的项目名。

3.6.5 容器作业排错

若容器作业失败，可以在/opt/skyformai/log/cbjm.< 作业运行节点 >.log 里查看作业的”docker run”参数，判断命令行是否有问题。例子:

```
06/02 12:31:14 I 10.24.0 2069 insDockerContainerHead: Job <14485> Docker command: /
↪usr/bin/docker run --name=CBJOB.14485 --network=host --init -i --rm --cpus=1  -e CB_
↪ALLOCATION -e CB_JOBID -e CB_SUBCWD -e CB_PORT -e CB_HOSTIP -e HOME -u 0:0␣
↪rockylinux:9
```

然后根据作业提交传入的 docker run 参数在命令行里执行以下，看结果。

3.7 Singularity/Apptainer 容器作业

Singularity/Apptainer 是专为 HPC 设计的容器技术，它有管理简便，不像 Docker 那样需要启动 Linux 服务（docker）。普通用户可以简便地上传容器镜像文件，让后直接使用。

以下以 Singularity 为例，Apptainer 与其类似。

对于 Enterprise Linux (RHEL、CentOS、Oracle Linux、Rocky Linux 等），Singularity 安装命令：

```
yum install -y epel-release
yum install -y singularity-ce
```

确保每台需要运行 Singularity 作业的节点上都安装了以上的 Singularity 软件包。

3.7.1 制作镜像文件

镜像文件制作需要 root 权限。系统管理员可以自行制作，然后把镜像文件放到共享文件系统中。用户也可在自己有 root 访问权限的 Linux 服务器上制作镜像，然后上传到自己的 HOME 目录中。

```
singularity build centos7.sif docker:centos:7
```

容器制作的方法请参考Singularity 文档。

3.7.2 提交 Singularity 简单作业

```
csub -I Singularity run /opt/containers/centos7.sif id
```

命令输出:

```
Job 4362 has been submitted to the default queue [medium].
Job 4362 is waiting to be started...
Job 4362 has started on host linux7.
uid=997(cadmin) gid=1006(grp_admin) groups=1006(grp_admin),1005(admin)
```

3.7.3 提交 Singularity MPI 作业

AIP 的 MPI 集成脚本 impi-mpirun(支持 Intel MPI、MPICH、MVAPICH）和 ompi-mpirun（支持 OpenMPI 和 IBM MPI）同时支持 Singularity 容器作业。

提交作业时有三个环境变量或 csub 参数可以使用:

环境变量	csub 参数	描述
CB_SINGULARITY_IMAGE	-Si	指定 Apptainer 镜像文件绝对路径
CB_SINGULARITY_OPTIONS	-So	指定 apptainer run 命令行可选参数
SINGULARITY_BIND	-Sb	指定容器中挂载主机上的目录 (HOME 不用挂载)

例子：提交 OpenMPI Singularity 作业，共需 128 个核，每个节点 32 个核，交互式作业, 作业输出文件为: 作业号.out:

```
source /opt/openmpi/openmpi.env
csub -Si /opt/containers/centos7.sif -Sb /opt/openmpi -n 128 -R map[perhost=32] -I -o
↪%J.txt ompi-mpirun ./mympi_prog
```

同样的方法也可用于 Intel MPI、MPICH、MVAPICH，只需把 ompi-mpirun 改成 impi-mpirun。

在以上的例子中，AIP 会用 4 个容器来运行作业，每个节点上一个 singularity 容器，每个容器中运行 32 个 MPI 任务。

与普通 MPI 作业类似，AIP 对整个作业在所有节点上的进程都进行监控，在杀死作业是确保所有节点上属于这个作业的进程都杀掉。

3.8 使用 Jupyter Lab

Jupyter Lab 通过 AIP 调度以作业的形式运行。

3.8.1 启动 Jupyter Lab

指定 jupyter 可执行文件路径，提交作业：

```
JUPYTER_PATH=/share/apps/python/bin/jupyter csub jupyterjob
```

命令输出：

```
Job 5363 has been submitted to the default queue [medium].
```

获取 URL：

```
cread
```

命令输出：

```
Messages posted to jobID 5363
POST_TIME: Oct 21 22:20:20  MESSAGE: /ajj/192.168.10.10/16332/lab?
↪token=f977ed7222b68285e35cbe0334bb22e222f94efd0ecae56c
```

在浏览其中输入 SkyForm 应用平台服务的 IP 地址或者 AIP Web 门户的 IP 地址，加上上面获得的 URL 字串：例子：

```
https://10.102.0.11/ajj/192.168.10.10/16332/lab?
↪token=f977ed7222b68285e35cbe0334bb22e222f94efd0ecae56c
```

3.8.2 在 Singularity 容器里启动 Jupyter Lab

除了指定 jupyter 可执行文件路径，指定 singularity 运行前缀。例子：

```
export JUPYTER_PATH=/share/apps/python/bin/jupyter
export CONTAINER_PREFIX=singularity run --pid -B /opt,/lib64,/usr,/share,/etc/profile.
↪d --env PATH=/usr/bin:/opt/skyformai/bin /share/apps/containers/rocky8.sif
csub jupyterjob
```

以上例子中，jupyterlab 在镜像为 rocky8.sif 的 Singularity 容器中运行。

3.8.3 切换成中文

Settings > Language > Chinese (Simplified, China)

点击蓝色按键 “Change and Reload”。

3.8.4 安装 Python 库

由于系统中有多个 Python 版本，务必确认要在哪个版本的 Python 环境中安装 Python 库。用 conda 命令切换环境。

Jupyter 内 Python 3 (ipykernel) 的环境：

```
conda activate /share/apps/python
pip install ...
```

3.8.5 安装 Julia 内核

系统中已经安装了 Julia 的一个版本，路径为/share/apps/julia-1.x.x/bin/julia，如果需要使用其他版本的 Julia，可以下载后解压到自己的 HOME 下。

在 Jupyter 里的启动页里点击“终端”图标打开命令行终端，设置 jupyter 运行路径，然后运行 julia。例子：

```
export PATH=/share/apps/python/bin:$PATH
/share/apps/julia-1.9.3/bin/julia
```

安装内核：

```
julia> import Pkg
julia> Pkg.build("IJulia")
julia> Pkg.add("IJulia")
```

检查启动页上的 Julia 内核图标

3.8.6 安装其他 Anaconda 部署的 Python 内核

例子：用 Anaconda 部署一个 Python 3.11 的版本，把内核插入到 Jupyter。

- 在 Jupyter 里启动页里点击“终端”图标打开命令行终端，然后运行 conda 命令：

```
conda create -p $HOME/python3.11 python=3.11
conda activate $HOME/python3.11
pip3 install ipykernel
python -m ipykernel install --prefix $HOME/python3.11
```

- 把内核移动到缺省位置：

```
mv $HOME/python3.11/share/jupyter/kernels/python3 $HOME/.local/share/jupyter/
↪kernels/python311
```

- 编辑内核定义文件，以区分与已有 Python 3.9 内核：

```
vi $HOME/.local/share/jupyter/kernels/python311/kernel.json
```

修改第 9 行

```
"display_name": "Python 3.11",
```

- 用浏览器刷新一下 Jupyter 页面，启动页就有了 Python 3.11 的图标。

备注： 若要在这个环境中安装 Python 库，务必先运行：conda activate $HOME/python3.11，然后再运行 pip install

---

这个方法可用于安装其他内核。关键点：

1. 在 conda 环境中安装好的内核必须移到缺省目录下 $HOME/.local/share/jupyter/kernels
2. 由于 Jupyter 启动内核时没有 conda 环境，kernels.json 里的环境变量和可执行文件路径需要增加或修改
3. 若在 Jupyter 里内核启动或使用失败，可以查看 Jupyter 日志：$HOME/.jupyter/session*jupyterjob* 作业号.log

3.9 使用 VSCodes

平台最佳使用 VSCode 的方法是：把 VSCode 安装在用户自己的机器上，通过 ssh 通道连接 AIP 管理的 HPC 资源。

如果想在计算环境中启动基于 Web 访问的 VSCode，最佳的方法是使用 code-server。详见本文最后部分。

3.9.1 在本地机器 (PC 或工作站) 上安装微软 Visual Studio Code

下载网页：https://code.visualstudio.com/download

3.9.2 安装扩展

按左侧第 5 个图块（把鼠标移到图块上，会显示提示 Extensions）或按快捷键 Ctrl+Shift+X，在左侧上方的搜索框里输入 ssh，在“Remote - SSH”的扩展区按 Install 键。安装完毕后，左侧图标列出现第 6 个图标，把鼠标放上去，会显示 Remote Explorer。

3.9.3 在 AIP 环境中启动可以 ssh 的容器作业

若使用 Singularity 容器，设置环境变量指定容器镜像路径，容器挂在主机文件系统的路径

```
export SINGULARITY_CONTAINER=/share/containers/rocky8.sif
export SINGULARITY_BIND=/opt,/lib64,/usr,/share,/etc/profile.d
```

提交作业

```
csub sshdjob
```

命令输出:

```
Job 5366 has been submitted to the default queue [medium].
```

获得 ssh 私钥

```
cview 5366
```

命令输出:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAy7Qccs9vs0dphX2/HD2mjUukTSyj7/HZoDlavZN/lp0j0U+o
ehO9onbNev4FMiejfE7cSke3zD+iiogpbJHjlF3ToRAevFPthBOP6zzdtMdVwS28
O0QsV5eE2fwH8QN3QiCZdUQrwYf4wdrfu/1EkcNGe1t0TohhEx3TmL+ui8KqbJA7
YVKSBTn6UydcgiY00OmhVZxEtwbo99OKaCmp6P1zqkTD+plrNIye/wGdyujs5A+w
lrPqOMLqlV3AGPH9tROQshmLbbAYrQ793PyHS8vfC/gmrxQ5mT2E4VnsPR0qsho+
thSgpOorGvEBiV+a8ZKJgfqH5Z6ZkRsS/oOAjwIDAQABAoIBAQCTPIpbighI41Mo
Q+3A8ebStlv51ExigMM2hn5KDLTqHobnIda69k3Zglmfjs Aesnv+u9mKbzAwl8Za
r3SRnxmKAg+XmVk98sP49VLPS11M78/SI8asZ+hNH53NGdh7om+vAnUpR68A4dEx
/Yw+CyBm42isfTwiD31rHeH1xIbbSP0xrYbcHfvjm0HdScDuXYelYoVRH0YPwMVg
at7Gb/rvq70jzEkDqunzMhMSh9K1qg9kh9p/lheCeIB14DjJESWuTse/+acuwQT1
AiyzEsco3593KXjmsDD+FCbXeQQ1od4WHXJ9J54Y15Fq8M9LgYpn1n7v79PXNt+N
9eVmkDBRAoGBAOuiKX4G+CoK7A8bv9mi/503HUObQ67d2YVxaasp2ZG9OWRidxIg
k9wywLbJt2qETaLAyNWbI6YqjyfpKjzg4PLmzeO5D9gxV40tloiZQ65M8uHd2I4M
```

(下页继续)

(续上页)

```
jeURRqvsjTTV7qzoN98VMzwwIcRRVXw28Vscs62PnP/bmsepI5jb3/WDAoGBAN1P
cIpDUMwe+LUNS99kXQYc7/GqinBOBhf6UqjhWDnPGccah5jXSZx2ig9gABRa+fz7
u4GcJ0mY4uhZYfFDtS0uCIKI0PA0Ve5JqNlz7lDwiy/3F5Y2f5ZgDY/hvriZa1LC
jDyz7Kt0ddr5+XTZHHj1PLof4D157C1zKXLIdGcFAoGAIxomOjsJjyHPZrTIkMBi
eeYy+tZPb+ZmuCVcC36Rhc99lEHC4TVvMXdyFAjaxQhulFdJ0+BnoCJo7xYiD62k
dSsC8vCntporI8Ht0e2bEoUuDY3B6+9c2AoJ0CTOKFfSVXnGkPoUhtvYu+kt5fI9
ZEhTUAC777WkSigdQHFI3McCgYEAuqgkyEisXe0FYVb8lkbE+sHvUkm/h6cXFqQn
kY62ZdOZ70cd9LgnutJz8THL/18YF9qNtGxq6nULdGLm66Fqrtdzj/0rFF1f2KHB
yt4vs46eJm9mMmh2xfLvnVoODQ5A6cVymEQ8qgOI348UagwwKojUljsKW2941oCa
Bdix7K0CgYEAtx06fRO5V+Elpmv7KoVTDLw/fKUiQKia98tUF42KVW0/WZD0iwaC
FqkrQ8wixzf3RPeaw6ppRLwQ3k9Uqi9tk7WXtPuvO1/1O7iuE9bL8OJki4cYd4QB
fRix5XjkjO9vObs1Oj7UggU/t8A1DDNH6pNSDV9LJzzMBFzfLtsK8do=
-----END RSA PRIVATE KEY-----
```

获得主机上的用户名，IP 地址，和端口

```
cread 5366
```

命令输出:

```
Messages posted to jobID 5366
POST_TIME: Oct 22 11:46:10  MESSAGE: ssh -i key -p 16331 cadmin@192.168.10.10
```

以上的例子中，用户名为 cadmin，主机 IP 为 192.168.10.10，端口为 16331。

3.9.4 配置远程 ssh

在安装 VSCode 的主机上编辑 ~/.ssh/config 文件。在 Windows 上，这个文件是：C:\Users\你的用户名\.ssh\config，如果 C:\Users\你的用户名\.ssh 目录不存在，请先生成目录。

在配置前，确保获取 ssh 主机上的用户名 ssh 私钥、主机 IP 地址、和端口（上一步）。

把私钥的内容存放到一个文件里，如 ~/.ssh/key (C:\Users\你的用户名\.ssh\key‘)

在 ~/.ssh/config 文件中增加一下内容。

备注： 若用 notepad 生成这个文件，确保生成后文件名没有后缀.txt。

```
Host myhost
  HostName 192.168.10.10
  IdentityFile ~/.ssh/key
  Port 16331
  User cadmin
```

以上：192.168.10.10 是上一步获得的主机 IP 地址，~/.ssh/key 是私钥文件路径，16331 是上一步获得端口，cadmin 是上一步获得的用户名。

3.9.5 在 VSCode 中连接

点击 VSCode 左侧第 6 个图标 Remote Explorer，点击 REMOTES (TUNNELS/SSH) 右侧的“Refresh”图标。

把鼠标移动“> SSH”上，点击”>“展开，就会看到配置的”myhost”，把鼠标移到”myhost”，点击后方的”->“图标开始 ssh 连接。

- 如果弹出窗口“Are you sure you want to continue?”，点击 Continue。
- 如果连接失败，试试删除本地文件：~/.ssh/known_hosts。

连接成功后，左下方显示 SSH: myhost

3.9.6 打开远程文件

点击左侧第一个图标 Explorer，点击“Open Folder”，选择远端主机上的目录或文件。若目录是一个 git 的仓库，则自动连接 git。

3.9.7 使用 code-server

系统配置

从官网 https://github.com/coder/code-server/releases 下载适合的.tar.gz 文件，如：https://github.com/coder/code-server/releases/download/v4.89.1/code-server-4.89.1-linux-amd64.tar.gz

解压到一个目录中，如

```
tar xfz code-server-4.89.1-linux-amd64.tar.gz -C /share/apps
ln -s /share/apps/code-server-4.89.1-linux-amd64 /share/apps/vscode
```

确保所有计算节点上的 SkyForm CRV 服务已经运行。若未运行，可以运行 host-setup –gui 来自动配置，例子：

```
/opt/skyformai_shared/host-setup --gui
```

修改文件/opt/skyformai/bin/vscodejob 的第 20 行里 code-server 的绝对路径，例子：

```
VSCODE_PATH=/share/apps/vscode/bin/code-server
```

使用

用户可以提交作业:

```
csub vscodejob
```

然后用命令查询链接：

```
cread 作业号
```

经过转发的链接的例子

```
/aj/192.168.10.100/16000/code.html?password=84bff127820db1371b00bff51d076e17&
↪port=16331
```

3.10 使用 RStudio

Rstudio 通过 AIP 调度以作业的形式运行。

3.10.1 启动 RStudio

指定 rstudio 安装路径，指定 R 执行文件的路径，提交作业：

```
export RSTUDIO_PATH=/share/apps/rstudio-server
export R_PATH=/share/apps/python/lib/R/bin/R
csub rstudiojob
```

命令输出:

```
Job 5364 has been submitted to the default queue [medium].
```

获取 URL：

```
cread
```

命令输出:

```
Messages posted to jobID 5364
POST_TIME: Oct 22 10:27:37  MESSAGE: /aj/192.168.10.10/16000/rstudio.html?port=16331&
↪username=cadmin&password=VxqrjCOVFV7XANwYYveZ6Fre1_cJeXna
```

在浏览其中输入 SkyForm 应用平台服务的 IP 地址或者 AIP Web 门户的 IP 地址，加上上面获得的 URL 字串：例子：

```
https://10.102.0.11/aj/192.168.10.10/16000/rstudio.html?port=16331&username=cadmin&
↪password=VxqrjCOVFV7XANwYYveZ6Fre1_cJeXna
```

3.10.2 在 Singularity 容器里启动 RStudio

除了指定 rstudio 安装路径和 R 执行文件的路径，指定 singularity 运行前缀。例子：

```
export RSTUDIO_PATH=/share/apps/rstudio-server
export R_PATH=/share/apps/python/lib/R/bin/R
export CONTAINER_PREFIX=singularity run --pid -B /opt,/lib64,/usr,/share,/etc/profile.
↪d --env PATH=/usr/bin:/opt/skyformai/bin /share/apps/containers/rocky8.sif
csub rstudiojob
```

以上例子中，RStudio 在镜像为 rocky8.sif 的 Singularity 容器中运行。

3.10.3 退出或结束运行

若退出了 RStudio，必须使用 URL 重新进入，而不是用重新登录的办法

若要结束运行，在外部杀掉作业

```
ckill 5364
```

3.10.4 安装 R 库

在 Console 里，安装即可，如:

```
install.packages('shiny')
install.packages('reticulate')
```

3.10.5 排错

查看日志 $HOME/.local/share/rstudio/log/rserver.log

3.11 使用 Nextflow

Nextflow 是一个用于创建可扩展、可移植且可重复的工作流系统。它采用数据流编程模型，让您能够专注于数据流和计算，从而简化并行和分布式流水线的编写。

本节介绍如何在 AIP 环境中使用 Nextflow。

3.11.1 在用户环境中安装 Nextflow

1. 检查和安装 Java 版本:

   ```
   java -version
   ```

   如果 Java 版本是 17 或以上，跳到第 2 步。

   如果系统的 Java 版本低于 17，或者 Java 没有安装，运行以下命令在用户环境中安装 Java 17:

   ```
   curl -s https://get.sdkman.io | bash
   source ~/.sdkman/bin/sdkman-init.sh
   sdk install java 17.0.10-tem
   java -version
   ```

   上面最后命令的输出:

   ```
   openjdk version "17.0.10" 2024-01-16
   OpenJDK Runtime Environment Temurin-17.0.10+7 (build 17.0.10+7)
   OpenJDK 64-Bit Server VM Temurin-17.0.10+7 (build 17.0.10+7, mixed mode, sharing)
   ```

2. 安装 Nextflow

   安装 Nextflow，把可执行文件移到 $HOME/.local/bin 里，确保 PATH 环境变量里带有 $HOME/.local/bin。以下的代码假设您用的是 bash，若使用 csh 或 tcsh，请按做相应修改。

```
curl -s https://get.nextflow.io | bash
mkdir -p $HOME/.local/bin/
mv nextflow $HOME/.local/bin/
echo 'export PATH=$PATH:~/.local/bin' >> ~/.bashrc
source ~/.bashrc
nextflow info
```

上面最后一条命令应该有类似这样的输出:

```
Version: 25.04.0 build 5944
Created: 08-05-2025 16:41 UTC (12:41 EDT)
System: Linux 3.10.0-1160.119.1.el7.x86_64
Runtime: Groovy 4.0.26 on OpenJDK 64-Bit Server VM 17.0.10+7
Encoding: UTF-8 (UTF-8)
```

3.11.2 Nextflow 利用 AIP 跑任务的例子

从 git 上下载一个例子:

```
git clone https://github.com/nextflow-io/training.git
cd training/hello-nextflow
```

修改 nextflow.config

```
docker.enabled = false

process {
  executor = 'lsf'          // 利用Nextflow与LSF的集成和AIP与LSF的兼容性
  queue = 'medium'          // 定义AIP的队列
  // memory = '4 GB'        **注意** 这个内存参数集成目前在LSF和AIP中都不工作！！
  cpus = 2                  // 每个任务的核数
  time = '1h'               // 任务运行时间限制
}
```

工作流文件

这里的例子中，我们使用 hello-world.nf，里面有 3 个子任务。

运行工作流

执行命令:

```
nextflow run hello-workflow.nf \
-with-report my_report.html \
-with-timeline my_timeline.html
```

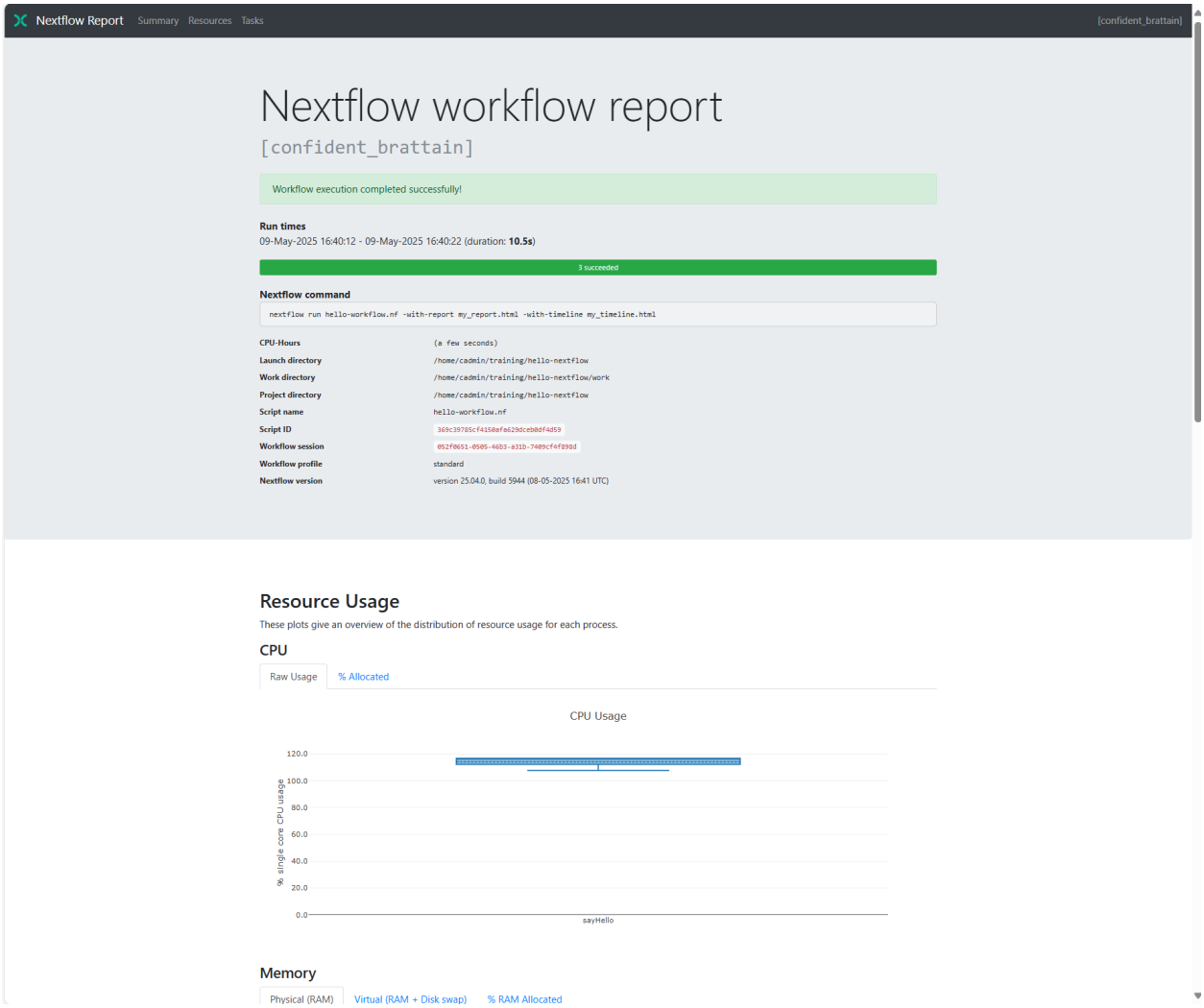
输出:

```
N E X T F L O W  ~  version 25.04.0

Launching `hello-workflow.nf` [confident_brattain] DSL2 - revision: 369c39785c

executor >  lsf (3)
[d0/efb916] sayHello (1) | 3 of 3 ✔
```

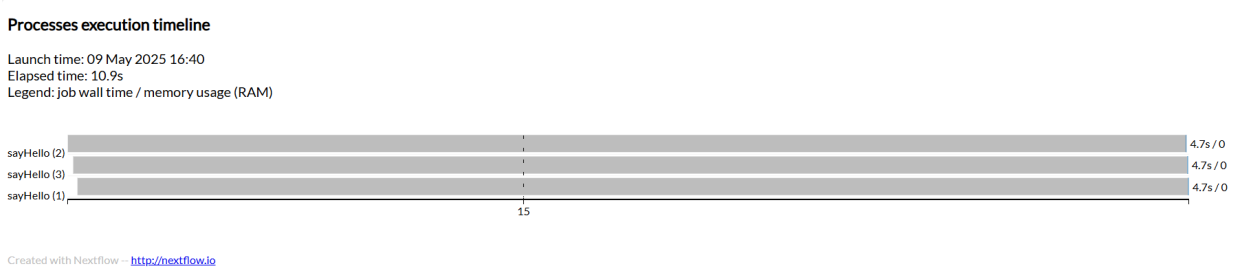
可以运行 AIP cjobs 命令查看作业（Nextflow 任务）运行状态:

```
cjobs -a
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
1277    cadmin  DONE  medium     dev         2*dev       *Hello_(2) May  9 16:40
1279    cadmin  DONE  medium     dev         2*dev       *Hello_(1) May  9 16:40
1278    cadmin  DONE  medium     dev         2*dev       *Hello_(3) May  9 16:40
```

生成的 my_report.html 文件:



生成的 my_timeline.html 文件:



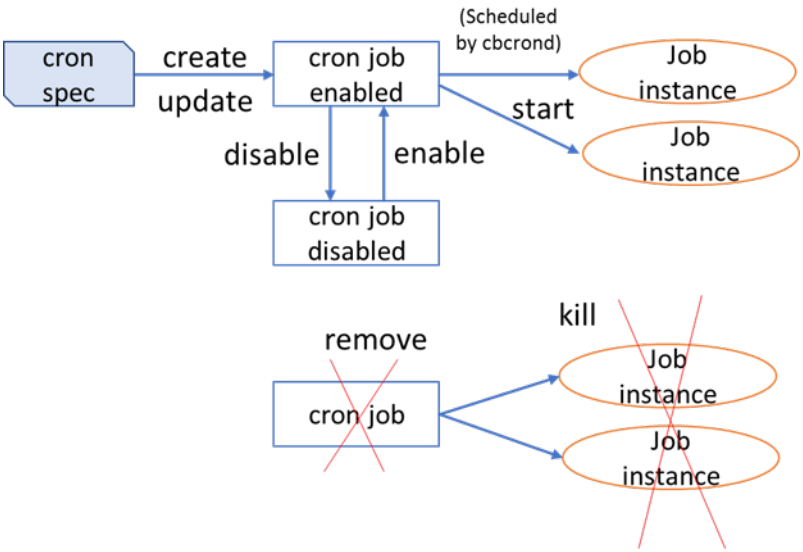
3.12 Cron 作业

Cron 作业类似于 Linux 里的 crontab，是由日历定义的重复性作业，一般用于定期的增量学习等。

SkyForm AIP 支持两种形式的 cron 作业定义，单个和批量。

3.12.1 cron 作业流程

Cron 作业的流程见下图：



备注： 每个 cron 作业都需要定义一个作业名。这个作业名必须是唯一，用于对此 cron 作业的操作。

1. 用户由 cron 作业定义用命令 aip cronjob create 一个 cron 作业。
2. Cron 作业调度器根据 cron 作业里 schedule 的定义把一个作业实例提交给 AIP 的调度器运行。实例受到 AIP 调度策略的制约。
3. 用户可以通过命令 aip cronjob disable | enable {cron job name} 来停止和启动 cron 的 schedule。
4. 用户可以用命令 aip cronjob start {cron job name} 来强制执行一次作业实例。这个实例不影响 cron schedule 生成的作业实例，但会影响 cron 作业策略（见下面关于策略的说明）
5. 命令 aip cronjob kill {cron job name} 杀掉所有正在运行的 cron 作业实例。
6. 若要删除系统里 cron 作业定义，用命令 aip cronjob remove {cron job name}

3.12.2 cron 作业定义和策略

在生成一个 cron 作业时，用户需要通过编辑 cron 作业定义，在定义中，需要指定一个唯一 cron 作业名和 AIP 作业定义，用户还可以配置此作业的策略。cron 作业的定义是一个 JSON 文件。

```
aip cronjob create
{
  # Mandatory.
  "Name": "",

  # Mandatory. A valid cron command with time and date fields.
  "Schedule": "",

  # Mandatory. The job specification. The value can be the path to a valid JSON
  # job specification file, or a JSON object that describes the job.
  "JobSpec": "",

  # Optional. The deadline in minutes to start an instance of a cron job if it
  # misses its scheduled run for any reason.
  # Default: unlimited.
  # "StartDeadline": 5,

  # Optional. The policy to start a new instance of a cron job if a previous
  # instance of the job is still running. Valid values are skip, run andreplace.
  # Default: skip.
  # "Overlap": "skip",

  # Optional. The maximum allowed number of concurrent running instances of a
  # cron job. This parameter only takes effect when Overlap is set to run.
  # Default: unlimited.
  # "MaxConcurrentRuns": 2,

  # Optional.
  # The maximum number of consecutive failures of a cron job (exits
  # with non-zero exit code) before it is disabled.
  # "MaxFails": 2,

  # Optional.
  # The number of finished instances of a cron job stored inside the cron
  # job daemon.
  # Default: 5.
  # "HistoryLimist": 5,

  # Optional.
  # Enable debug.
  # Default: false.
  "Debug": false
}
```

1. Name: cron 作业名，**必须项**，是一个由字母（可以大小写）和数字组成的字串，作业名不可含有其它字符。作业名必须定义，不定义作业名则不能生成 cron 作业。
2. Schedule：作业实例运行的时间表，**必须项**，用 Linux 里的 crontab 语法。例子：

   “5 0 * * *”：每天 0:05 运行

   “15 14 1 * *”：每月 1 日的 14:15 运行

   “0 22 * * 1-5”：每周一至周五 22:00 运行

“23 0-23/2 * * *”：0:23 开始运行，之后每 2 小时运行一次

“5 4 * * sun”：每周日 4:05 运行

3. JobSpec：AIP 的作业定义，**必须项**。可以是 JSON 字串或 AIP 作业定义文件。如：

   “JobSpec”：“./myjobspec.json”，

   “JobSpec”：{ “Command”:” python model.py” },

4. StartDeadline：可选项。定义若是由于某种原因（如系统资源不够，或 AIP 调度器忙等），cron 作业实例没有按时运行，多长时间内仍需要运行。单位是分钟。如：

   ” StartDeadline”：60,

   是指实例若延误 60 分钟后不运行了，cron 的作业实例会在下个时间点照常运行。缺省是无限长，即每个实例不管延误多久都得运行。

5. Overlap：可选项。可选值为：

   - “skip”：若到运行的时间时上一个时间点的实例还在运行，则跳过本实例。**这是缺省的策略**。
   - “run”：若到运行的时间时上一个时间点的实例还在运行，则本实例照常运行。这种情况就可能会有同属一个 cron 作业的 2 个或多个实例同时运行。
   - “replace”：若到运行的时间时上一个时间点的实例还在运行，杀掉前一个未结束的实例，启动本实例。

6. MaxConcurrentRuns：当多个实例的运行在时间上有重叠时，最多可重叠的实例个数。缺省是无限多。

7. MaxFails：可选项。当连续多个实例失败（不能提交给 AIP 调度器或运行结果和状态为非 0）后，cron 作业会被禁止。缺省是无限多，即不会被禁止。禁止后的 cron 作业可以用命令 aip cronjob enable {job name} 来恢复。

8. HistoryLimit：可选项。定义 cron 调度器内存里存放 cron 作业实例的数量，缺省是 5 个。定义大的数字增加调度器使用的系统内存。

9. Debug：可选项。提供进一步的排错信息，是给 AIP 开发人员使用的，缺省值是 “false”。

作业定义可以是文件，STDIN（标准输入），或命令行参数。如一个简单的 cron 可以用以下的方式提交:

```
aip cronjob create cronjobspec.json
```

```
aip cronjob create < cronjobspec.json
```

```
aip cronjob create '
\ {"Name":"mycron",
\ "Schedule":"1 0 \* \* \*",
\ "JobSpec":{"Command":"sleep 1"}
\ }'
```

在提交 cron 作业时，若要修改以前系统里有效的同名的 cron 作业，可以用命令:

```
aip cronjob create -o cronjobspec.json
```

若系统中没有同名的 cron 作业，则新建一个;

或:

```
aip cronjob update cronjobspec.json
```

若系统中没有同名的 cron 作业，命令会被拒绝

3.12.3 查看 cron 作业

例子:

```
aip cronjob info
```

```
USER    NAME    STAT      NUM_JOBS  LAST_FINISH             NEXT_RUN
cadmin  c0178   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0083   DISABLED  0         Oct  4 13:39 (Finish)   -
cadmin  c0897   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0519   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0282   DISABLED  0         Oct  4 13:39 (Finish)   -
cadmin  c0983   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0133   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0100   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0196   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0900   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0388   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0826   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0432   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0147   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0274   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0407   DISABLED  0         Oct  4 13:20 (Finish)   -
cadmin  c0621   DISABLED  0         Oct  4 13:20 (Finish)   -
```

```
aip cronjob info -l
```

```
{
  "CronJobSpec": {
    "name": "c0207",
    "user": "cadmin",
    "status": "DISABLED",
    "lastSucc": 1538673631,
    "numInstances": 0,
    "overlapPolicy": "skip",
    "startDeadline": 0,
    "maxConcurrentRuns": 0,
    "maxFails": 0,
    "historyLimit": 5,
    "schedule": "0-59/1 * * * *",
    "jobSpec": {
      "Command": "sleep 10"
    }
  },
  "Finished": [
    {
      "id": 1727,
      "status": "Done",
      "submitTime": "2018-10-04 13:16:03.039186881 -0400 EDT m=+1003.863218052",
      "startTime": "2018-10-04 13:16:21 -0400 EDT",
      "endTime": "2018-10-04 13:16:39 -0400 EDT",
      "exitCode": 0
    },
    {
      "id": 2728,
      "status": "Done",
      "submitTime": "2018-10-04 13:17:01.912797348 -0400 EDT m=+1062.736828429",
      "startTime": "2018-10-04 13:17:33 -0400 EDT",
      "endTime": "2018-10-04 13:17:45 -0400 EDT",
      "exitCode": 0
    },
    {
      "id": 3637,
      "status": "Done",
      "submitTime": "2018-10-04 13:18:02.514059785 -0400 EDT m=+1123.338090956",
      "startTime": "2018-10-04 13:18:19 -0400 EDT",
      "endTime": "2018-10-04 13:18:39 -0400 EDT",
      "exitCode": 0
    },
    {
      "id": 4579,
      "status": "Done",
      "submitTime": "2018-10-04 13:19:01.952831722 -0400 EDT m=+1182.776862894",
      "startTime": "2018-10-04 13:19:19 -0400 EDT",
      "endTime": "2018-10-04 13:19:35 -0400 EDT",
      "exitCode": 0
    },
```

3.12.4 批量 cron 作业

若需要用同一个 cron 的时间和策略定义运行多个作业，需要定义一个批量 cron 作业。批量作业共享的部分由 Common 里的变量定义，而不同的部分放到各自的文件里。例子:

1. 批量 cron 作业定义文件 packcron.json

```
{
  "Common":
  {
    "Schedule": "0 1 \* \* \*",
    "JobSpec": "",
    "StartDeadline": 60,
    "Overlap": "skip",
  },
  "Individual":
  {
    # Mandatory. The value is a valid file path and each line in the file is one␣
↪cronjob name.
    "Name": "cronnames.txt",

    # Mandatory. The individual fields of job specification.
    "JobSpec":
    {
      # Mandatory. The value is a valid file path and each line in the file is␣
↪one job command.
      "Command": "croncmds.txt"
    }
  }
}
```

2. 由于每个 cron 作业必须定义一个作业名，我们生成一个作业名列表文件，每行对应一个唯一的作业名 cronnames.txt

```
node0001cpu
node0001mem
node0001disk
node0002cpu
node0002mem
node0002disk
```

3. 产生一个 cron 作业命令行列表文件，每行对应 cronnames.txt 里作业名的作业命令

```
python monitors.py node0001 cpu
python monitors.py node0001 mem
python monitors.py node0001 disk
python monitors.py node0002 cpu
python monitors.py node0002 mem
python monitors.py node0002 disk
```

4. 用命令产生批量 cron 作业

```
aip cronjob pack packcron.json
```

查看作业:

```
aip cronjob info
```

作业定义的例子可见 AIP 例子 里 general/cronpackjobs.spec.json。

CHAPTER 4

组件

4.1 VNC 和 SSH 访问门户

VNC 和 SSH 访问门户是利用 AIP REST API 服务认证用户，协助用户在计算主机上提交和管理 VNC 桌面作业或者 SSH 作业的 web 门户。

4.1.1 安装

警告： 安装由系统管理员用 root 账号执行。VNC 和 SSH 访问门户不能与 AIP 门户在同一台主机上。

安装 VNC 和 SSH 访问门户自动在同一台主机上安装 AIP 的 REST API 服务：aiprestd。

1. 选择一台 AIP 的登录计算主机或登录主机。
2. 安装 nginx 和 httpd（或 apache2）。

   在 EL 上:

   ```
   yum install -y httpd nginx
   ```

   在 Ubuntu 上:

   ```
   apt install apache2 nginx
   ```

3. 在 AIP 包解压的目录中，运行 vncportal-install 脚本:

   ```
   ./vncportal-install
   ```

4. 在 AIP 集群中有一个名为 vnc 的队列，这个队列中的主机是可以运行 VNC 或 ssh 作业的主机。如果是 VNC 主机，确保主机在做 API 配置时使用脚本：host-setup --gui

4.1.2 用户 VNC 客户端

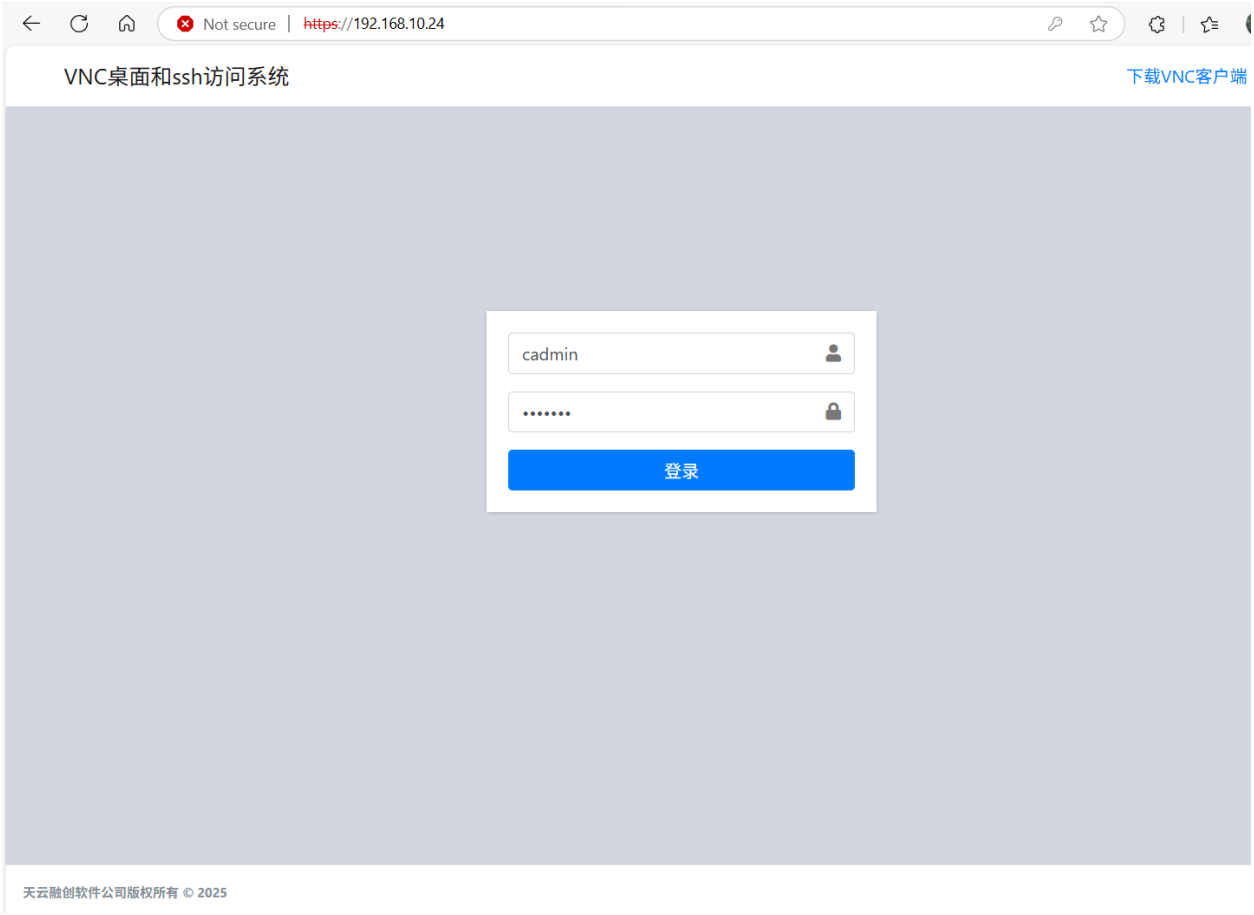
VNC 门户提供浏览器方式访问 VNC，用户也可以选择使用 VNC 客户端连接，客户端的图形效果略好。

VNC 客户端可以用 RealVNC。AIP 的安装包里带了 RealVNC Windows 和 Mac 的客户端，用户可以点击“下载 VNC 客户端”，展开压缩包，安装 RealVNC 客户端。

4.1.3 登录

用浏览器连接到 VNC 和 SSH 访问主机 https://*portal_ip*

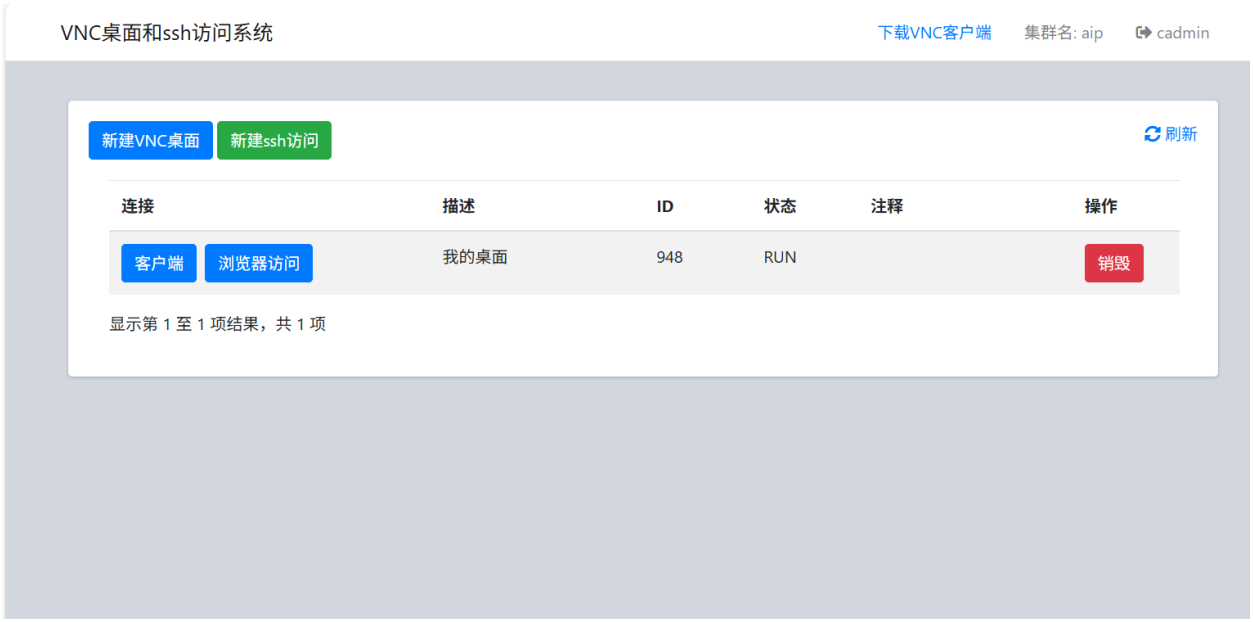
以操作系统中的用户和密码登录。



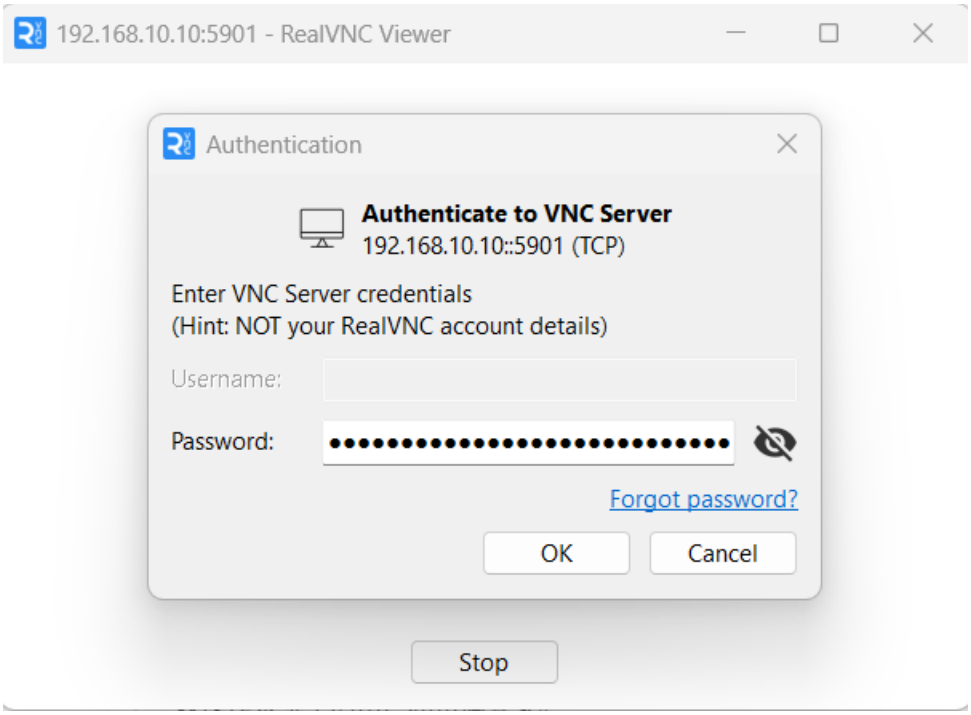
4.1.4 启动 VNC 桌面

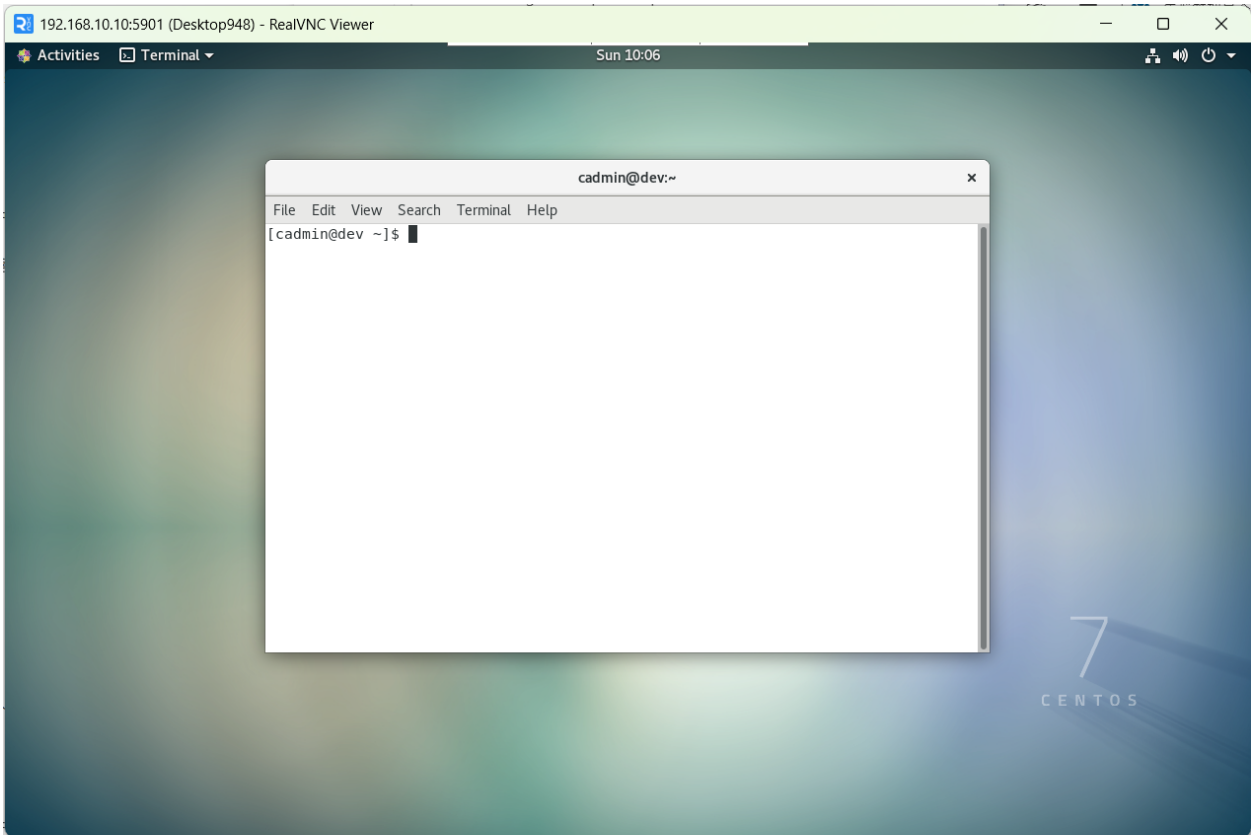
点击蓝色的“新建 VNC 桌面”，选填桌面名称，选择“提交”键启动一个桌面。

如果系统中有资源可以运行 VNC 桌面，几秒后，“客户端”和“浏览器访问”两个按钮就可以选择使用。



点击“浏览器访问”在浏览器中弹出一个标签页连接到 VNC 桌面，或者点击“客户端”打开 VNC 客户端。VNC 客户端启动后会提示密码，这个密码是随机生成的，点击“客户端”时浏览器已把密码复制到了系统的粘贴板上，在 VNC 客户端中只需按 Ctrl-V 就可把密码粘贴在密码提示框里，点击 OK 键就可以连接。



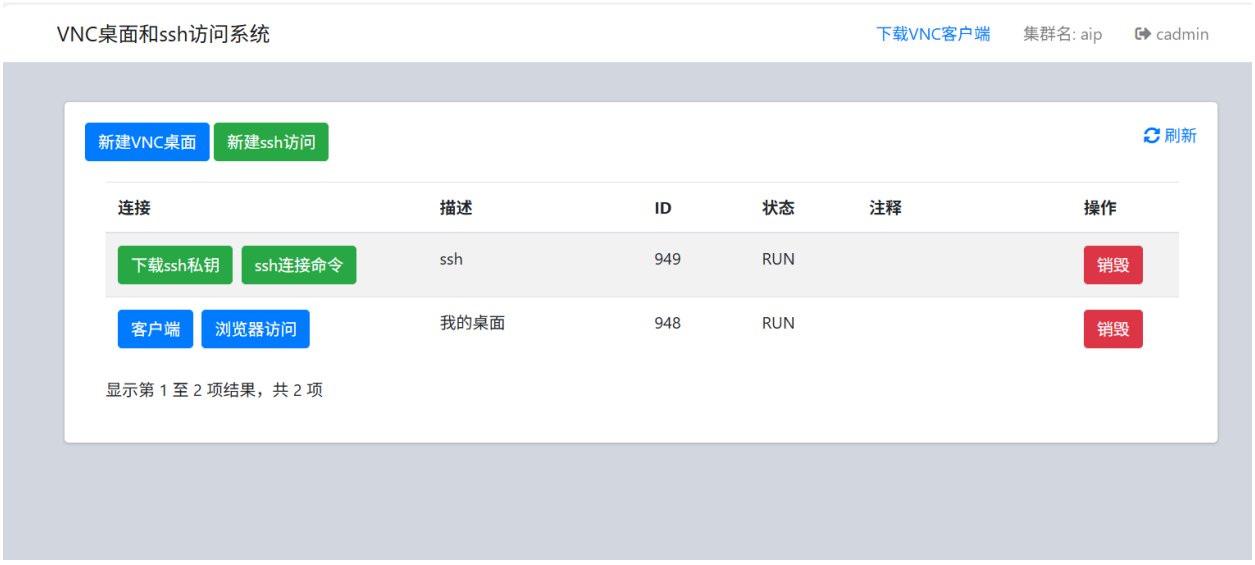


关闭桌面可以在 VNC 桌面里面选择 logout，或者在门户里面按红色“销毁”按钮。

4.1.5 启动 ssh 访问

点击绿色的“新建 ssh 访问”，选填描述名称，选择“提交”键启动一个 ssh 访问。

如果系统中有资源可以运行 ssh 访问，几秒后，“下载 ssh 私钥”和“ssh 连接命令”两个按钮就可以使用。



系统为每个 ssh 访问生成一个私钥。只有使用这个私钥才可以访问。

备注： 门户中生成的 ssh 访问不能使用用户密码访问。

1. 点击“下载 ssh 私钥”下载文件 key.pem。下载后的密钥保存在一个目录中。
2. 点击“ssh 连接命令”，可以选择“拷贝”按钮拷贝命令行。

3. 打开 Windows 的 cmd 或 powershell 终端。把当前目录切换到下载 key.pem 的目录中，粘贴上一个步骤拷贝的命令，连接 ssh

```
ssh -i key.pem -p 16332 cadmin@192.168.10.10
```

4.1.6 VNC 桌面和 ssh 会话的资源使用控制

VNC 桌面和 ssh 会话中用户会运行很多命令，推荐在队列中配置 cgroup 参数来限制其资源使用，这样主机上的多个卓面和会话不会把主机跑死。

在*cb.yaml* 队列中设置 cgroup 参数，cgroup 只控制本队列中的作业。例子:

```
queues:
- name: vnc                 # VNC和SSH队列
  priority: 3
  cgroup: cpu mem           # 限制每个桌面和会话的CPU和内存使用
  memlimit: percore         # 每个作业内存上限是主机总内存除以CPU核数
```

4.2 SkyForm AIP 的 REST API 服务

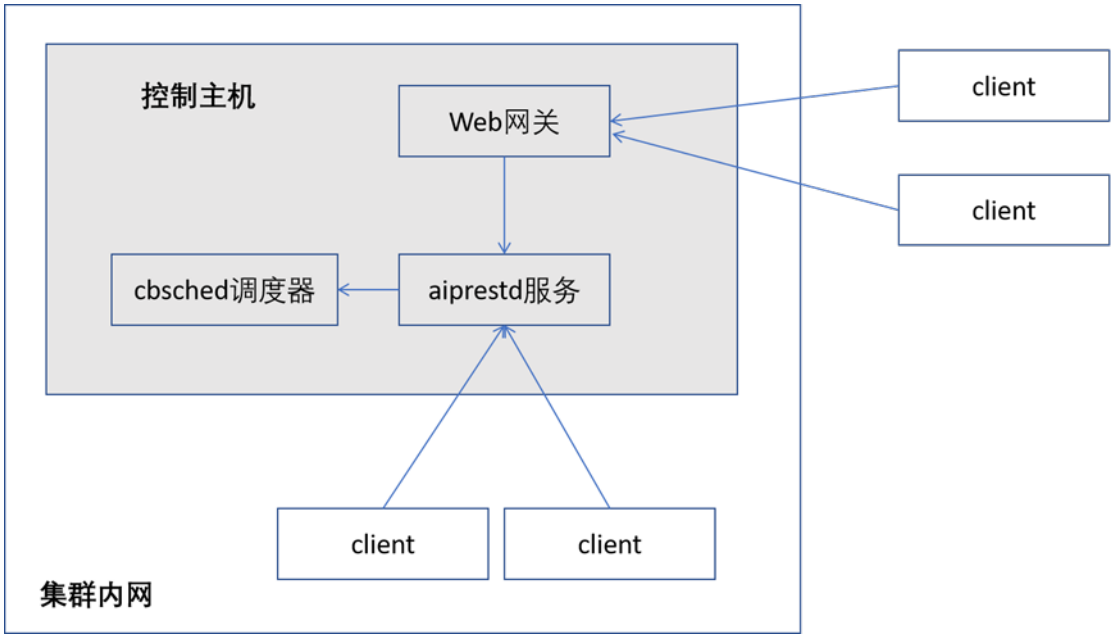
4.2.1 系统简介

SkyForm AIP 提供了标准的 RESTful Web 服务 aiprestd，以允许图形作业和 Batch 作业提交、作业信息查询、作业操作、查询和管理集群、主机、、用户、队列等。用户既可以从集群内网，也可以通过公网访问 aiprestd 服务。访问 aiprestd 服务的客户端主机可以是 Linux 也可以是 Windows，不需要安装 SkyForm AIP 软件，只要支持标准 HTTP 协议。

SkyForm aiprestd 服务是无状态的，不会在请求之间缓存或保存任何状态，任何请求都与 SkyForm AIP 调度器完全同步。

支持通过常用开源 web 网关，如 NGINX 等，让用户从公网访问 aiprestd 服务，确保数据安全。

图 1 展示了 aiprestd 的架构。



4.2.2 安装 aiprestd

安装

aiprestd 服务打包在 SkyForm API 安装包中。解压 AIP 的安装包，切换当前目录到解压包内的 rest 目录下，运行安装命令:

```
cd rest
./installaiprestd
```

安装脚本安装 aiprestd 服务，并启动它。

配置 aiprestd.yaml

修改配置文件/opt/skyformai/etc/aiprestd.yaml

- RESTful Web 服务 http 端口，缺省 8088:

```
# Public web port for disabling https. The default is 8088
# http_port: 8088
```

- 启用 SSL，缺省禁用:

```
#------------------------------------------------------------------
# Enabling https. The default is 0 (disabled). Valid value is 1 or 0
# ssl: 0
#------------------------------------------------------------------
# Public web port for disabling https. The default is 8088
# http_port: 8088
#------------------------------------------------------------------
# Public web secure port for enabling https. The default is 8043
# https_port: 8043
#------------------------------------------------------------------
```

(下页继续)

(续上页)

```
# Https certificate file path. If enable https, need to generate it
# cert: /opt/cert/server.crt
#-----------------------------------------------------------------
# Https key file path. If enable https, need to generate it
# key: /opt/cert/server.key
```

- Token 过期时间（分钟）, 缺省 30 分钟:

```
# Token timeout value. Integer. Unit is minute. Default value is 30 min
timeout: 300
```

- 日志级别，缺省 info:

```
# Log level. Valid values are fatal, error, warn, info, debug.
# Default value is info
# log_level: info
```

管理 aiprestd 服务

aiprestd 服务使用 Linux 的 systemd 管理。

- 检查 aiprestd 服务:

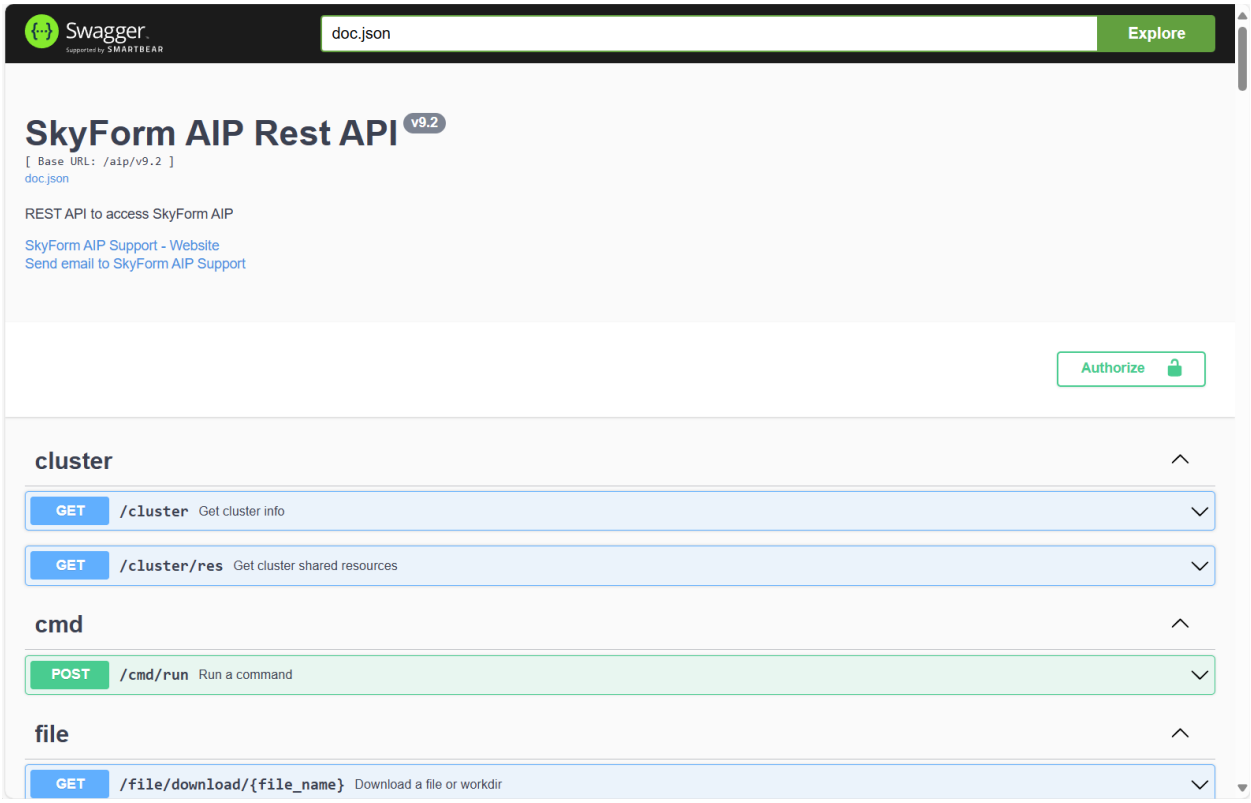
```
systemctl status aiprestd
```

- 查看 aipresrd 日志:

```
more /opt/skyformai/log/aiprestd.log.$HOSTNAME

journalctl -u aiprestd
```

4.2.3 REST 接口

本章提供几个 REST 接口示例，接口使用详情，请参考在线接口文档：`http://$HOSTNAME:8088/aip/v9.2/swagger/index.html`



认证

/aip/v9.2/login 获得 Token

- 所有 REST 接口 (除了认证接口/login 和在线文档/swagger/index.html) 调用都需要提供 Token。
- Token 的默认过期时间为 30 分钟。Token 过期时间指获得 token 后，一直没有任何操作的时间间隔。如果有调用 REST 接口，Token 过期时间会自动延期。如果 Token 已经过期，需要重新调用/login 获得新 Token。

Request:

```
curl -s -X 'POST' \
 'http://$HOSTNAME:8088/aip/v9.2/login' \
 -H 'accept: application/json' \
 -H 'Content-Type: application/json' \
 -d '{"password": "password", "username": "cadmin"}'
```

Response:

```
{
  "msg": "Success",
  "data": {
    "token": {
      "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJuYW1lIjoiY2FkbWluIiwiaWF0IjoxNjQ4MTA4NjcxLCJpc3MiOiJTa3lGb3JtIEFJUCJ9.
↪uXqBPRkcBcE0IenUZePwHsHmZTrTDv8_aYB_1Xw_8pY",
      "userName": "cadmin"
```

(下页继续)

(续上页)

```
    }
  }
}
```

提交可视化作业

/aip/v9.2/vis/submit

- 请求 header 添加“Authorization: Bearer $token”，$token 为/login 接口返回的 token 字符串
- 可视化作业提交成功后，返回作业号

Request:

```
curl -s -X 'POST' \
 'http://node1:8088/aip/v9.2/vis/submit' \
 -H 'accept: application/json' \\
 -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJuYW1lIjoiY2FkbWluIiwiaWF0IjoxNjQ4MTA4NjcxLCJpc3MiOiJTa3lGb3JtIEFJUCJ9.
↪uXqBPRkcBcE0IenUZePwHsHmZTrTDv8_aYB_1Xw_8pY' \
 -H 'Content-Type: application/json' \
 -d '{"command": "/bin/firefox", "type": "vnc"}'
```

Response:

```
{
  "msg": "Success",
  "data": {
    "vis": {
      "jobId": 22456
    }
  }
}
```

查询可视化作业 URL

/aip/v9.2/vis/$job_id

- 通过可视化作业提交返回的作业号，查询作业访问 URL
- 最终作业访问 URL 需要将 2.5 中部署的 httpd 服务地址和查询结果拼接，比如本示例最终作业 Web 访问 URL 为: `http://$hostname/novnc/vnc.html?autoconnect=true&host=10.211.59.111&port=42466&password=RQPxve1DcUztap5wtS3YyLltDO3kfwSX&resize=remote&quality=9&compression=5&exec=node1&sid=8`

Request:

```
curl -s -X 'GET' \
 'http://node1:8088/aip/v9.2/vis/url/22456' \
 -H 'accept: application/json' \
 -H 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJuYW1lIjoiY2FkbWluIiwiaWF0IjoxNjQ4MTA4NjcxLCJpc3MiOiJTa3lGb3JtIEFJUCJ9.
↪uXqBPRkcBcE0IenUZePwHsHmZTrTDv8_aYB_1Xw_8pY'
```

Response:

```
{
  "msg": "Success",
  "data": {
    "visurl": {
      "url": "/novnc/vnc.html?autoconnect=true&host=10.211.59.111&port=42466&
↪password=RQPxve1DcUztap5wtS3YyLltDO3kfwSX&resize=remote&quality=9&compression=5&
↪exec=node1&sid=8"
    }
  }
}
```

4.2.4 高可用方案

由于 REST API 服务依赖于 sqlite 数据库/opt/skyformai/work/aiprestd.db。这个文件缺省存放在共享文件系统内。高可用方案使用多台主机上部署 REST API 服务，所以必须把这个 sqlite 数据库文件移到本地。

1. 把 sqlite 数据库移到本地文件系统，并生成符号链:

```
systemctl stop aiprestd
rm -f /opt/skyformai/work/aiprestd.db
ln -sf /var/run/aiprestd.db /opt/lsf/work/aiprestd.db
```

2. 在集群的两到三台主机上部署 REST API 服务:

```
cd rest
./installaiprestd
```

3. 配置 NGINX 的一个新的端口转发，使用 keepalive 参数。例子/etc/nginx/conf.d/aiprest.conf:

```
upstream aiprest {
   server 192.168.10.10:8088;
   server 192.168.10.11:8088;
   keepalive 32;
}

server {
   listen 18088;
   location / {
      proxy_pass http://aiprest;
      proxy_http_version 1.1;
      proxy_set_header Connection "";
   }
}
```

以上例子中，访问 REST API 的 URL 为: http://host_name:18088/aip/v9.2/

4.3 AIP 监控数据分析系统

4.3.1 系统简介

SkyForm AIP 资源和作业管理调度系统常用于运行和调度模拟、分析、人工智能等应用和作业。为了确保系统自动根据配置分配资源、用户作业运行正常及集群资源的充分利用，图形化的监控，报告和分析工具是必备的。而传统的关系型数据库和数据仓库工具不足以高速地处理作业调度管理系统生成的大数据量，尤其是在系统中存在大量活动作业的情况下。

利用现代开源数据处理和可视化分析工具的优势，天云软件开发的 SkyForm AIP 监控系统是针对大型集群作业环境的作业负载监控、报告和分析软件。该监控系统具有活动的仪表盘，灵活的数据，可视化和可定制的数据分析，让用户能够轻松地产生报表以检测服务质量和规划未来的容量增长。

4.3.2 安装

安装前准备

在开始安装 SkyForm AIP 监控系统前，请先做好以下准备工作：

- 获取安装包

  Elasticsearch v8/v9、Grafana v11/v12、Kibana v8/v9 安装包，可到 elastic.co 和 grafana.com 官网下载

- 确保所有机器间能够通过 DNS，NIS 或/etc/hosts 使用主机名互相通信。
- Elasticsearch 机器可以不是集群中的机器，但是必须能够被 SkyForm AIP 机器所访问。建议您将 Elasticsearch 安装在专门的主机上，以保证 Elasticsearch 具有足够的资源提供服务。
- Elasticsearch 机器上需要安装 Java run time environment，安装 JRE:

```
yum install java
```

- 定义数据记录的时间间隔。

  默认情况下所有数据的记录间隔为 30 秒。对于某些环境，比如有 1000 个主机，尤其是当作业平均时间为 30 分钟的环境，间隔为 30 秒可能太短，因为这样会有大量的数据被记录。

  数据记录间隔在**/opt/skyformai/etc/olmon.conf** 文件中定义*olmon.conf* ，相关变量及其含义如下表所示。

变量	描述	默认值
job_update_interval	更新不在作业事件文件中的作业信息的时间间隔。	30 秒
host_update_interval	收集和主机信息的时间间隔。在大规模的环境中，数据量会很大。	30 秒
re source_update_interval	采集共享资源数据的时间间隔。	30 秒
rusagep reason_update_interval	记录作业等待原因和作业的资源使用情况数据的时间间隔。	60 秒
log_duration	数据收集时长（天）。仅最近一个收集时长内的数据被保存在数据库中。	90 天
eshosts	定义安装 Elasticsearch 的主机名，如：eshosts=host1,host2	
grafana	安装 Graf ana 的主机名或 IP 地址，web 用户必须能以这个地址或主机名访问到该主机	
kibana	安装 Kib ana 的主机名或 IP 地址，web 用户必须能以这个地址或主机名访问到该主机	
job_index	是否在 ES index jobs 里保存作业详细数据	1（是）

安装 Elasticsearch，Kibana，Grafana

SkyForm AIP 的监控系统只支持 Elasticsearch/Kibana 7.x 和 8.x。用户可直接从 elastic.co 官网下载最新的 Elasticsearch 和 Kibana，从 grafana.com 下载最新版的 grafana (如 v11)。

1. 在所有规划好的 Elasticsearch 机器上安装 Elasticsearch:

```
yum -y install ./elasticsearch*.rpm
```

或者:

```
dpkg -i ./elasticsearch*.deb
```

2. 在所有 Elasticsearch 机器上安装 Elasticsearch RPM 或 DEB 后，修改文件**/etc/elasticsearch/elasticsearch.yml**，配置和添加以下内容（假设 Elasticsearch 安装在 host1、host2、和 host3:

```
cluster.name: aip
node.name: ${HOSTNAME}
network.host: 0.0.0.0
discovery.seed_hosts: ["host1","host2","host3"]
cluster.initial_master_nodes: ["host1","host2","host3"]
# 关闭xpack
xpack.security.enabled: false
```

/etc/elasticsearch/jvm.options，配置 JVM 的可用内存，最佳时间为主机内存的一半。如主机内存为 8G，则设置 4G（原来文件中设的是 1G）。生产环境中最少设成 4G 以上:

```
-Xms4g
-Xmx4g
```

在每台 Elasticsearch 机器上启动 Elasticsearch 服务:

```
systemctl enable --now elasticsearch
```

3. 在某一台 Elasticsearch 机器上安装 Kibana 服务，例如选择第一台 Elasticsearch 机器安装。

```
yum -y install kibana*.rpm
# 或
dpkg -i kibana*.deb
```

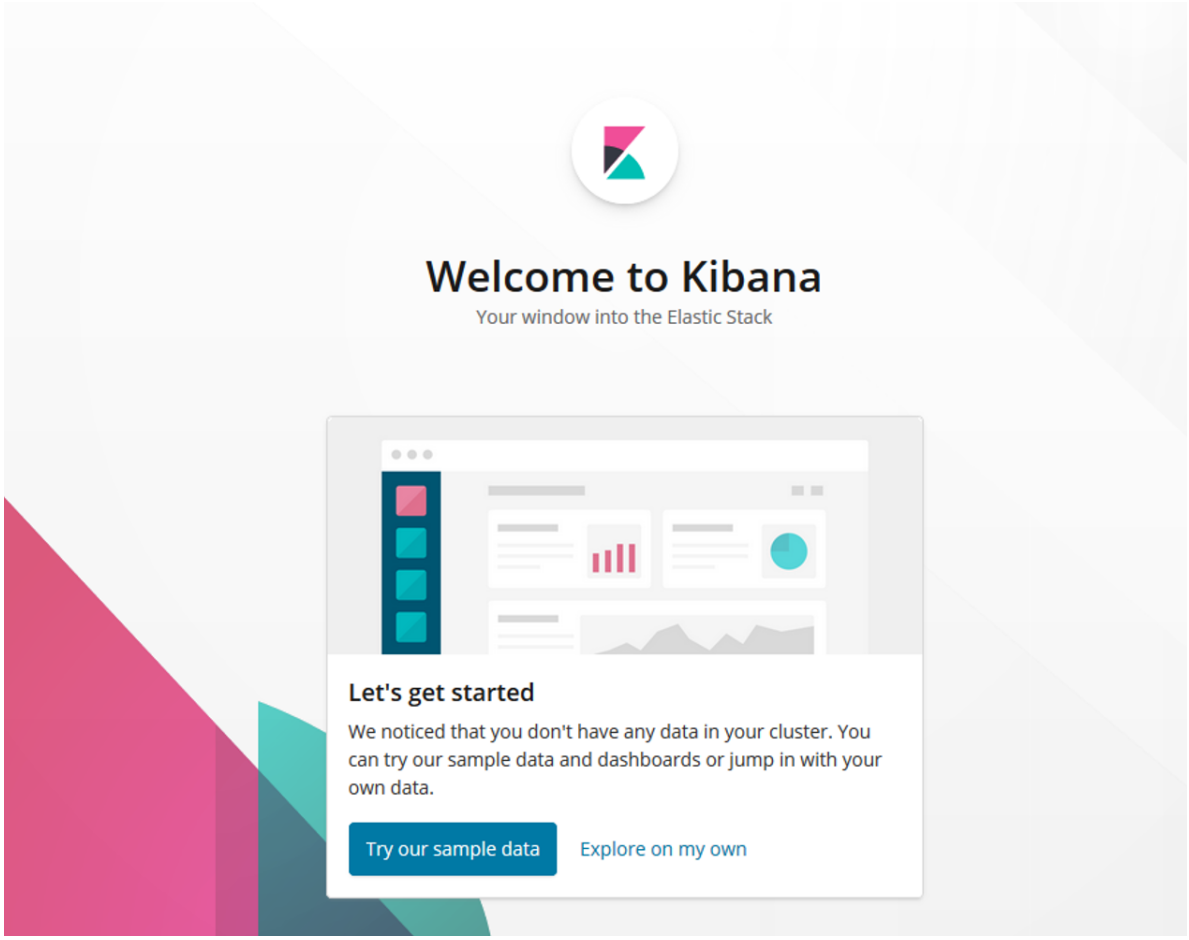
修改 Kibana 配置 /etc/kibana/kibana.yml:

```
# 允许所有机器都可以访问Kibana
server.host: “0.0.0.0”
# 把语言设成中文：
i18n.locale: "zh-CN"
```

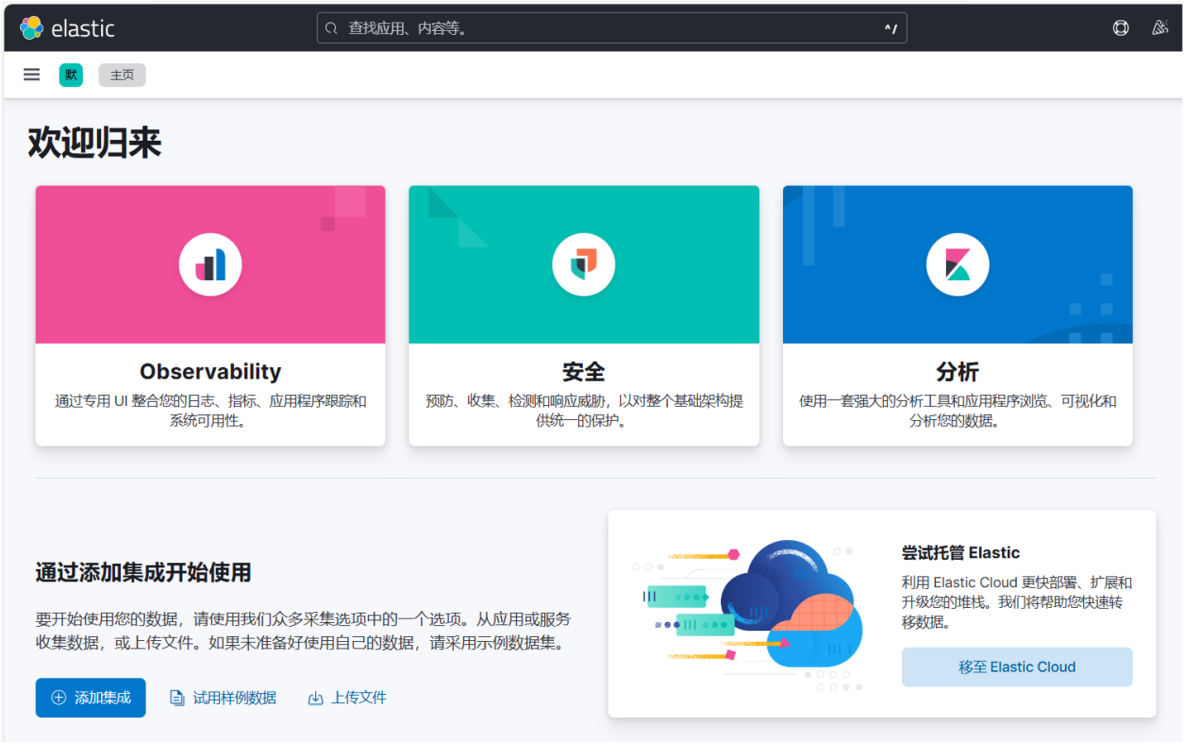
启动 Kibana 服务:

```
systemctl enable --now kibana
```

等待 2-3 分钟的 Kibana 初始化，然后用浏览器连接



选择 Explore on my own。



4. 在某一台 Elasticsearch 机器上安装 Grafana 服务，例如选择第一台 Elasticsearch 机器安装。

```
yum -y install grafana*.rpm
# 或
dpkg -i grafana*.deb
```

配置 Grafana，编辑/etc/grafana/grafana.ini:

```
[users]
default_theme = light
[security]
allow_embedding = true
[auth.anonymous]
enabled = true
```

启动 Grafana 服务:

```
systemctl enable --now grafana-server
```

备注： 以下的操作中，工具使用用户名 admin 和密码 admin 部署和提取 Grafana 里的 dashboard。用户 admin 的密码若修改会影响工具 upload_grafana_dbs 和 download_grafana_dbs 的使用。

5. 建议您在 Elasticsearch 机器及 Kibana/Grafana 机器上均关闭防火墙。

安装数据采集服务（data collector）

> **警告：** 数据采集服务（data collector）需要安装在 SkyForm AIP 集群的 master 上。

安装步骤如下：

1. 安装数据采集服务

数据采集服务所需的文件已经安装在 SkyForm AIP 所使用的共享文件系统里，如/opt/skyformai/monitor。

运行安装脚本:

```
cd /opt/skyformai/monitor
./installservice
```

1. 在 **$CB_ENVDIR** 目录下修改配置文件 **olmon.conf**。可以拷贝/opt/skyformai/etc/olmon.tmp 的内容，然后修改。参数可以参考安装前准备 的说明。添加至少以下内容:

   ```
   eshosts=host1,host2,host3
   log_duration=90
   kibana=192.168.20.70
   grafana=192.168.20.70
   ```

   其中：

   - **eshosts**：Elasticsearch 机器的主机名，以逗号（,）分隔（以上以 Elasticsearch 主机名为”host1”和“host2”为例）。
   - **log_duration**：数据保留最长天数。若想保留时间长，则 Elasticsearch 里的数据量会非常大，磁盘一旦满，数据就会出错。可以通过延长数据采集间隔减少数据量。

   缺省是每 30 秒采集一次，建议添加以下变量减少采集频率 (如 180 秒一次):

   ```
   job_update_interval=180
   host_update_interval=180
   resource_update_interval=180
   ```

   - **kibana**：Kibana 机器的 IP
   - **grafana**：Grafana 机器的 IP

   > **警告：** 不要在此文件中配置的主机名或 IP 后填写注释类信息，如“kibana=192.168.20.70 # kibana server IP”等，否则可能会影响主机名或 IP 的解析。

2. 检查安装是否成功。

   确保作业调度管理系统中有正在运行的作业。

   在前台运行数据采集服务，设置参数为“-2”。运行后等待一段时间直至显示 **data/stream/cb.stream.0** 文件被打开，然后输入 **Ctrl-C** 退出。

   执行以下命令:

   ```
   /opt/skyformai/sbin/olcol -2
   ```

   输出例子:

```
2024-05-08 13:39:38 109633 GB/memory unit=1048576.000000
2024-05-08 13:39:38 109633 olcol: started
2024-05-08 13:39:38 109633 Host query started
2024-05-08 13:39:39 109633 Shared resource query started
2024-05-08 13:39:39 109633 readevents: start to read file /opt/skyformai/work/
↪data/cb.data
2024-05-08 13:39:50 109633 readevents: processed 5000 lines
2024-05-08 13:39:50 109633 readevents: processed 10000 lines
2024-05-08 13:39:50 109633 readevents: processed 15000 lines
2024-05-08 13:39:53 109633 readevents: processed 20000 lines...
2024-05-08 13:40:07 109633 cb.data file completed
2024-05-08 13:40:07 109633 readevents: start to read file /opt/skyformai/work/
↪data /stream/cb.stream.0
2024-05-08 13:40:07 109633 readevents: processed 5000 lines
2024-05-08 13:40:08 109633 Job query started
^C
```

3. 启动 olmon 服务:

```
systemctl start olmon
```

查看日志文件**/opt/skyformai/log/olmon.log** 中是否有错误信息产生。

数据采集服务 OLMON 高可用

Olmon 服务自动定期检测当前运行主机是否为集群 master。如果不是，则不采集数据。

Olmon 服务的高可用方案为：在所有 AIP 集群 master 的候选机上都安装 olmon 服务，一个时候只有当前为 master 的主机上的 olmon 采集数据，其他主机上的 olmon 服务处于等待状态。当 AIP master 切换时，olmon 检测到当前的主机为 master，自动激活采集数据。

导入 Kibana&Grafana 配置

在运行 olmon 服务的主机上，运行 php -m。确保输出含有 yaml 和 json:

```
php -m | egrep "yaml|json"
json
yaml
```

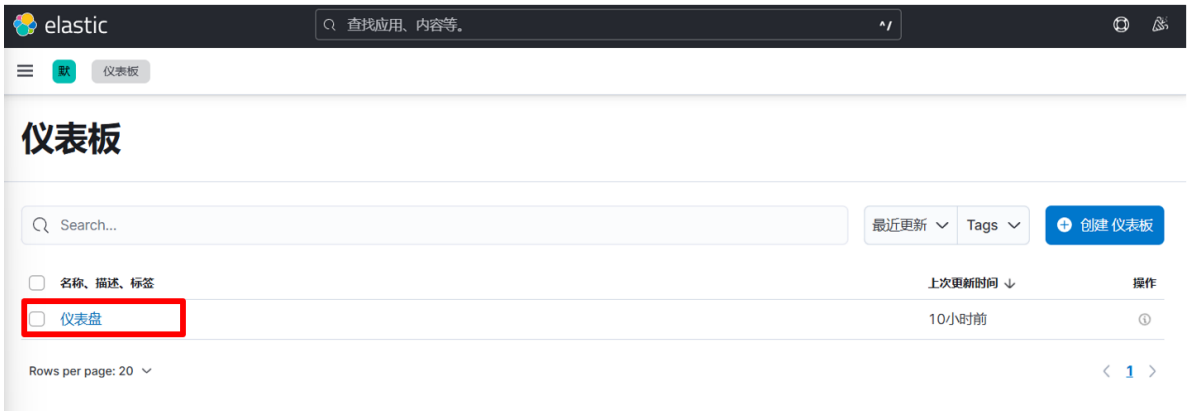
若缺少，需要安装:

```
# EL7
yum -y install epel-release
yum -y install php-pecl-yaml
# EL8
wget https://skyformaip.com/aip/php-yaml.el8.x86_64.tar.gz
tar xfz php-yaml.el8.x86_64.tar.gz -C /
dnf -y install php-json
# EL9, EL10
dnf -y install php-yaml
```

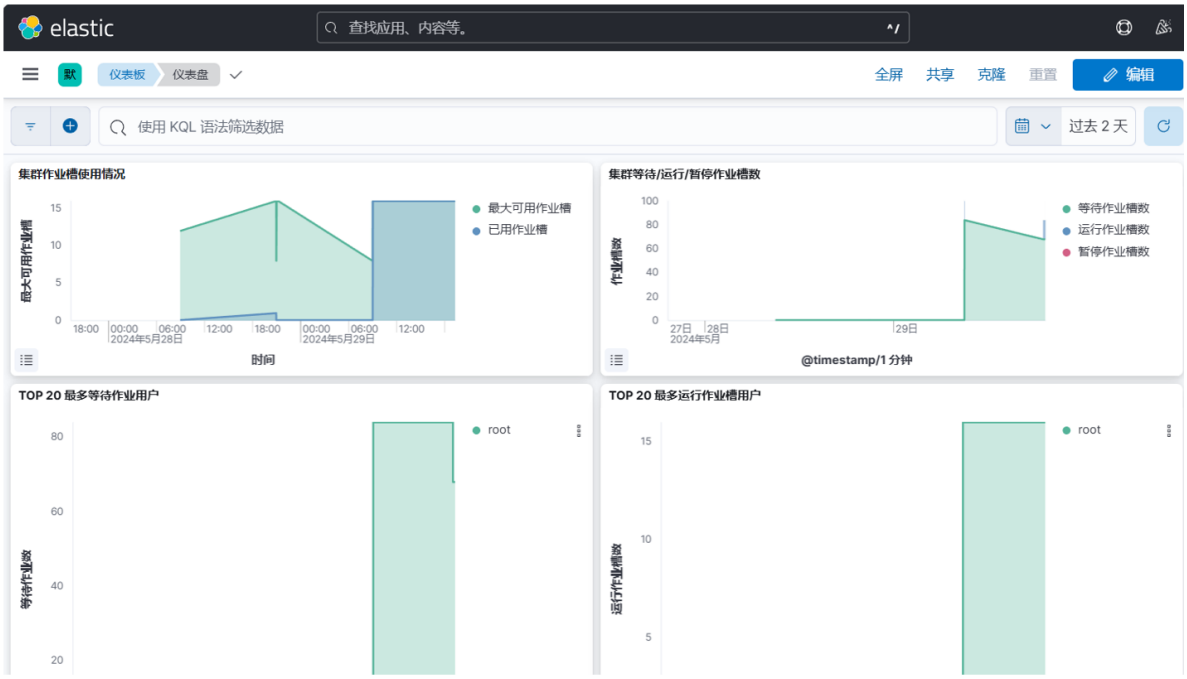
1. 运行以下命令使 Kibana 能从 Elasticsearch 中获取数据:

```
cd /opt/skyformai/monitor
./upload_kibana_conf
```

然后在 Kibana 页面上（http://kibana 主机:5601），选择左上方的菜单（三横线），然后选择 Analytics 里的 Dashboards。



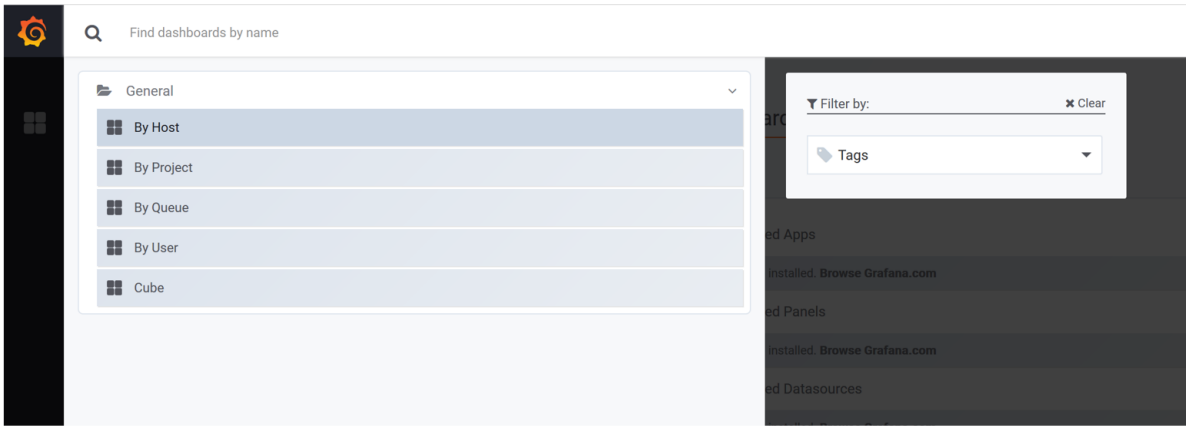
您将会看到 Kibana 已配置的仪表盘。



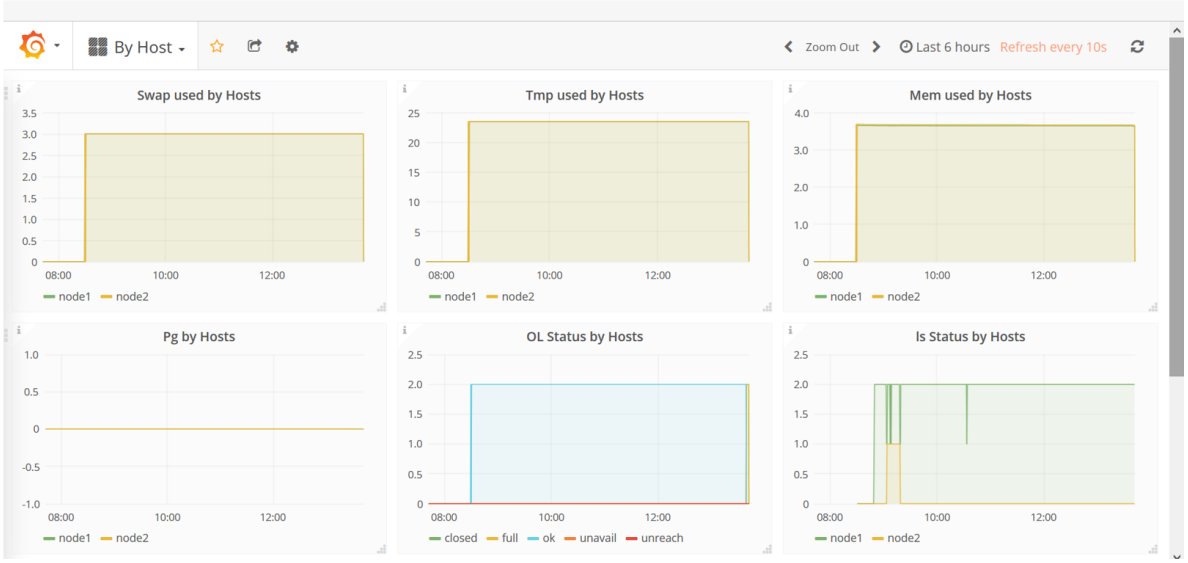
2. 运行以下命令配置 Grafana 仪表盘:

```
./upload_grafana_dbs
```

在浏览器中输入 http://<grafana_host>:3000 访问 Grafana GUI。点击左上角的 **Home**，将展示所有可显示的仪表盘。



选择一个要查看的仪表盘。



4.3.3 与 AIP 门户的集成

参考*集成 AIP 监控数据分析系统*。

4.3.4 维护和定制

修改 olmon.conf 时间参数后重新导入 Kibana&Grafana 配置

在 olmon 服务运行一段时间后，而 olmon.conf 里的时间参数修改了，Kibana 和 Grafana 里的图表配置必须修改。Vision 提供了工具帮助自动修改 Kibana 和 Grafana 里配置的时间间隔参数。

1. 导出配置:

```
cd /opt/skyformai/monitor
./download_kibana_conf
./download_grafana_dbd
```

小技巧: 若你修改和定制了 Kibana 和 Grafana 的图表，这个工具也可用于备份 Kibana 和 Grafana 的所有配置。

2. 重新导入:

```
./upload_kibana_conf
./upload_grafana_dbs
```

在 jobs 和 users-time 索引中增加外插用户组织信息

通过运行外部命令（如数据库访问等方式）获得用户的额外信息，如组织关系，可以选择插入到 jobs 和 users-time 索引的数据中。

配置方法如下:

在 olmon.conf 里配置两个参数:

- user_data_cmd = 可执行文件的全路径

  这个可执行文件（如脚本）执行后的输出格式为:

  第一行: 需要插入的索引名，如" jobs users-time"

  第二行: 数据名，多项用空格隔开，如 "tenant dept"

  第三行起，每行一个用户数据，格式为: 用户名，数据值，如" u001 skyform dev" , 其中 skyform 为 tenant 的值，dev 为 dept 的值。Olmon 自动把以下内容插入索引中用户名为 u001 的记录中: tenant: skyform, dept: dev

  可执行文件命令（脚本）输出例子:

```
users-time
tenant dept
u001 skyform dev
u002 skyform consult
u003 simforge test
u004 simforge test2
```

- user_data_interval = 更新间隔（olmon 服务运行以上脚本的间隔），单位为秒。缺省为一小时，3600 秒。

  例子:

```
user_data_interval = 300
```

4.3.5 故障排查

Elasticsearch 数据写入

数据采集系统服务每分钟监控数据采集进程一次。如果它发现一些进程死掉，会重新启动它。

当有数据写入 Elasticsearch 错误时，请查看以下文件进行故障定位或将其发送给技术支持人员:

监控系统数据采集器主机上的**/opt/skyformai/log/putbody-时间戳.txt** 文件。

Elasticsearch 的日志文件在 Elasticsearch 主机的**/var/log/elasticsearch** 目录下。

Elasticsearch 数据读取例子

假设 host-times 的数据更新间隔（host_update_interval）为 60 秒，读取最近一次的所有主机监控数据:

```
curl -XPOST -s 'http://eshost:9200/hosts-time/_search?size=30000' \
-H "Content-Type: application/json" \
-d '{"query":{"range":{"@timestamp":{"gt": "now-1m"}}}}'
```

假设 jobs-times 的数据更新间隔（job_update_interval）为 30 秒，读取最近一次的所有运行作业监控数据（最多 50 万个作业）:

```
curl -XPOST -s 'http://eshost:9200/jobs-time/_search?size=500000' \
-H "Content-Type: application/json" \
-d '{"query":{"range":{"@timestamp":{"gt": "now-30s"}}}}'
```

4.3.6 数据说明

本章描述监控在 Elasticsearch 里存放的数据字段。每一节描述一个 index。

jobs

Index 名：jobs（每个作业一条数据，随作业生命周期的时间而更新）。

```
{
 "cluster": "aip",      # 集群名
 "@timestamp": "2020-09-06T01:28:23.000Z", # 数据更新时间，时区为UTC
 "jobId": 225,          # 作业号
 "options": 33554435,   # aip内部使用
 "numProcessors": 1,    # 作业请求的最小CPU核数
 "submitTime": "2020-09-05 21:28:23", # 作业提交时间，本地时区
 "submitTimeUTC": 1599355703, # 作业提交时间戳
 "beginTime": "",       # 作业提交请求开始运行时间，空：未定义
 "beginTimeUTC": 0,     # 作业提交请求开始运行时间戳，0：未定义
 "termTime": "",        # 作业提交请求结束运行时间，空：未定义
 "termTimeUTC": 0,      # 作业提交请求结束运行时间戳，0：未定义
 "userName": "u003",    # 作业提交用户名
 "rLimits": [           # 作业提交进程资源限制参数表（详见getrlimit(2)
                        # 或ulimit）-1表示无限或用aip服务运行的缺省设置
                        # (/opt/skyformai/etc/limits.sh)
   -1,                  # CPU时间限制（秒）
   -1,                  # 生成文件大小限制
   -1,                  # 使用数据(data segment)大小限制
   -1,                  # 使用堆栈(stack segment)大小限制
   -1,                  # 生成core文件大小限制
   -1,                  # 内存使用限制
   -1,                  # 未来预留
   -1,                  # 未来预留
   -1,                  # 未来预留
   -1,                  # 运行时间限制
   -1                   # 未来预留
 ],
 "hostFactor": 61.9,    # 作业提交主机的CPU 性能参数
 "queue": "medium",     # 队列名
 "resReq": "",          # 作业提交资源需求定义字串（详见csub(1) -R参数）
```

(下页继续)

(续上页)

```
"memReq": 0,          # 提交内存预留值 (MB)
"fromHost": "aip-master", # 作业提交主机名
"cwd": "/root",       # 作业请求运行工作路径
"inFile": "",         # 标准输入 (stdin) 文件
"outFile": "",        # 标准输出 (stdout) 文件
"errFile": "",        # 标准错误输出 (stderr) 文件
"subHomeDir": "/home/u003", # 作业提交用户home目录
"numAskedHosts": 0,   # 提交时指定作业运行主机数
"askedHosts": [],     # 提交时指定作业运行主机名阵列
"dependCond": "",     # 作业依赖定义 (详见csub(1) -w参数)
"preExecCmd": "",     # 作业预处理程序路径 (csub(1) -E)
"command": "/bin/sleep 227", # 作业命令行
"mailUser": "",       # 作业提交指定邮箱通知的邮箱用户名
"projectName": "default", # 项目名
"interact": 0,        # 终端交互作业 0: 否, 1: 是
"maxNumProc": 1,      # 提交请求最大CPU核数
"loginShell": "",     # 指定的作业运行登录shell, 空为不用登录shell运行作业
"options2": 5136,     # AIP内部使用
"userPriority": -1,   # 用户定义的作业优先级, -1: 未定义
"userGroup": "",      # 指定作业用户组, 空: 未定义
"jobGroup": "",       # 作业所属作业组, 空: 未定义
"jobDesc": "",        # 作业描述字串, 空: 未定义
"status": "PEND",     # 当前作业状态。详见在线文档cjobs 描述
"startTime": "",      # 作业开始运行时间, 本地时区, 空: 尚未开始运行
"endTime": "",        # 作业结束运行时间, 本地时区, 空: 尚未结束运行
"numExHosts": 0,      # 运行占用作业槽数 (CPU核数)
"execHosts": [],      # 运行作业槽对应的主机名阵列
"reason": 0,          # 作业等待或暂停原因内部参数, 可调用AIP C API
                      # cbSuspReason()或cbPendReason()解析
"subreasons": 0,      # 作业等待或暂停原因内部参数, 可调用AIP C API
                      # cbSuspReason()或cbPendReason()解析
"jRusageUpdateTime": 0, # 最近作业资源使用情况更新时间戳
"pendReasons": " New job is waiting for scheduling: 1 host;^",
                      # 作业等待原因解析结果
"suspReasons":"",     # 作业暂停原因解析结果
"mem": 0,             # 当前作业内存使用量 (MB)
"stime": 0,           # 作业运行所用系统CPU时间 (秒)
"utime": 0,           # 作业运行所用用户CPU时间 (秒)
"swap": 0,            # 作业运行交换区用量 (MB)
"pidInfo": [],        # 作业所属的进程号和进程组号
"execCwd": "",        # 作业运行工作路径
"maxMem": 0,          # 作业运行最大内存用量 (MB)
"avgMem": 0,          # 作业运行平均内存用量 (MB)
"alloc": "",          # 调度器为作业分配的资源JSON串, 包括主机、CPU、GPU、端口等
"taskInfo": [],       # 子任务 (使用AIP分发远程MPI子任务或分布式深度学习作业)
                      # 的资源使用
"jobName": "test[108]", # 作业名
"idx": 108,           # 作业阵列index
"cpuUsage": 0.01,     # 作业CPU利用率0 - 1代表0 -100%
 “runtime” : 0,         # 作业运行时间 (秒)
 “idle” : false,        # 当队列配置了job_idle参数后作业限制检测
 “msg” : 0,             # 最后作业消息的发布时间戳
 “content” : "",        # 最后的作业消息内容
 “exitStatus” : 0,      # 作业退出码
 “exitStatusString” : "", # 作业退出exit code或者退出时接受到的signal,
                      # 值为: “Exit Code xxx”, 或“By Signal xxx”,
```

(下页继续)

(续上页)

```
                        # xxx为整数
  “exitReason” : "",      # 作业异常退出的原因说明
  “nthreads” : 0,         # 总线程数
  “app” : ""              # 作业提交时-A参数定义的应用名
}
```

jobs-time

Index 名：jobs-time（每 job_update_interval 每运行作业一条数据）

```
{
  "cluster": "aip",     # 集群名
  "@timestamp": "2020-09-06T01:28:23.000Z", # 数据更新时间，时区为UTC
   “jobId” : 225,         # 作业号
  "idx": 108,           # 作业阵列index
   “cpu” : 0.12,          # 作业CPU用时（秒）
   “mem” : 0.12,          # 作业所用内存GB
   “swap” : 0.23,         # 作业交换区使用GB
   “nthreads” : 2,        # 作业线程数
   “io” : 23,             # 作业存储IO率KB/秒 AIP 9.20.3新指标
   “gmem” : 4,            # 作业所用GPU内存MB AIP 9.20.3新指标
   “user” : “cadmin” ,    # 作业所属用户名
   “project” : “default” , # 项目名
   “descript” : “” ,       # 作业描述名
   “slots” : 4,           # 作业占用槽数
   “host” : “linux”        # 作业运行第一台主机名
   “cpuUt” : 0,           # 作业整个运行的CPU利用率（0-1表示0-100%）
   “smpl_cpuut” : 0.98,   # 作业在过去一个采样点的CPU利用率（0-1表示0-100%）
   “ps” :[{               # 作业进程详情
     “host” : "linux",    # 作业进程运行主机名
     “pids” : [{          # 主机上的所有该作业进程
       “pid” : 111,       # 进程号
       “pgid” : 111,      # 进程所属组号
       “cput” : 0,        # CPU 用时（秒）
       “mem” : 0,         # 内存使用量MB
       “swp” : 0,         # 交换区使用量MB
       “nthreds” : 3,     # 进程所含线程数
       “diskrd” : 0.3,    # 进程存储读速率KB/秒
       “diskwr” : 0.5,    # 进程存储写速率KB/秒
       “gmem” : 0,        # 进程GPU内存用量MB
       “cmd” : "(sleep)" # 进程命令
    }]
  }]
}
```

hosts-time

Index 名：hosts-time (每 host_update_interval 每主机一条数据)

```
{
   “@timestamp” : "2020-09-06T01:28:23.000Z", # 数据更新时间，时区为UTC
   “cluster” : "aip",      # 集群名
   “name” : "linux",       # 主机名
   “status” : "Ok",        # 主机状态（详见在线文档chosts描述）
                         # Ok: 良好
                         # Unavail: AIP服务异常
                         # Unreach: 作业服务cbjm异常
                         # Closed-Excl: 主机被独占作业占用
                         # Closed-Full: 主机所有作业槽都被占用
                         # Closed-Busy：主机负载超过所设阈值
   “hgroup” : "",          # 主机所属主机组名
   “maxJobs” : 56,         # 主机作业槽上限
   “numJobs” : 3,          # 运行、暂停、和保留的作业槽总数
   “numRUN” : 3,           # 运行作业槽数
   “numSSUSP” : 0,         # 被系统暂停（如过载）的作业槽数
   “numUSUSP” : 0,         # 用户或管理暂停的作业槽数
   “numRESERVE” : 0,       # 调度器为大作业预留的作业槽数
   “maxMem” : 100,         # 主机最大可用内存GB
   “memUt” : 0.3,          # 包含作业预留的内存使用率（0-1代表0-100%）
   “realmemUt” : 0.4,      # 实际内存使用率（0-1代表0-100%）
   “maxSwap” : 2,          # 最大可用SWAP GB
   “swapUt” : 0.1,         # SWAP使用率（0-1代表0-100%）
   “slotUt” : 0.054,       # 作业槽使用率（0-1代表0-100%）
   “r15s” : 0,             # CPU进程队列15秒平均
   “r1m” : 0,              # CPU进程队列1分钟平均
   “r15m” : 0,             # CPU进程队列15分钟平均
   “ut” : 0.1,             # CPU平均使用率（0-1代表0-100%）
   “mem” : 90,             # 可用内存GB
   “swp” : 1.2,            # 可用SWAP GB
   “up” : 13456,           # 正常运行时间（秒）
   “it” : 0,               # 空闲时间（秒）
   “tmp” : 45,             # /tmp可用存储GB
   “usedMem” : 12,         # 已用内存GB
   “reservedMem” : 10,     # 作业预留的内存GB
   “gpu” : 1,              # 可用gpu卡数
   “pg” : 0,               # paging rate (page/sec)
   “io” : 0,               # 网络IO KB/秒
   “cpus” : [{             # CPU核利用率
     “id” : 0,             # CPU核ID
     “ut” : 0.1}],         # 核利用率（0-1代表0-100%）
   “gpus” : [{             # GPU监控
     “id” : 0,             # GPU卡号
     “model” : "GTX3090", # GPU型号第2个以后的字段（如Tesla M60为M60）
     “maxmem” : 15098,     # GPU最大显存MB
     “mem” : 15020,        # GPU可用显存MB
     “usage” : 0,          # GPU作业分配律（0-1代表0-100%）
     “gut” : 0,            # GPU真实利用率（0-1代表0-100%）
     “memUt” : 0,          # GPU显存利用率（0-1代表0-100%）
     “temp” : 21}],        # GPU温度（摄氏）
   “nic” : [{              # 网络设备IO
     “device” : "eth0",    # 设备名
     “recv” : 0,           # 接收数据KB/秒
     “send” : 0}],         # 送出数据KB/秒
```

(下页继续)

(续上页)

```
   "diskio" : [{           # 本地存储设备IO
     "device" : "sda",     # 设备名
     "read" : 0,           # 读取数据KB/秒
     "write" : 0}],        # 写入数据KB/秒
   "pids" : [{             # 作业相关进程监控
     "pid" : 9090,         # 进程号
     "pgid" : 9090,        # 进程所属组号
     "cput" : 3,           # CPU 用时（秒）
     "mem" : 123,          # 内存使用量MB
     "swp" : 34,           # 虚拟内存使用量MB
     "nthreds" : 1,        # 进程所含线程数
     "diskrd" : 0,         # 进程存储读速率KB/秒
     "diskwr" : 0,         # 进程存储写速率KB/秒
     "gmem" : 0,           # 进程GPU内存用量MB
     "jobid" : 122,        # 进程所属作业号
     "idx" : 0,            # 进程所属作业阵列号（非阵列作业为0）
     "user" : "u001"}]     # 所属用户
}
```

users-time

Index 名：users-time (每 rusagepreason_update_interval 每用户每队列一条数据)

```
{
   "@timestamp" : "2020-09-06T01:28:23.000Z", # 数据更新时间，时区为UTC
   "user" : "cadmin",       # 用户名
   "queue" : "medium",      # 队列名
   "numRUN" : 2,            # 该用户在该队列中运行作业总槽数
   "numPEND" : 100,         # 该用户在该队列中等待作业总槽数
   "numSUSP" : 0,           # 该用户在该队列中暂停作业总槽数
   "jobRUN" : 1,            # 该用户在该队列中运行作业总数
   "jobPEND" : 50,          # 该用户在该队列中等待作业总数
   "jobSUSP" : 0,           # 该用户在该队列中暂停作业总数
   "cluster" : "aip"        # 集群名
}
```

preasons-time

Index 名：preasons-time （每 rusagepreason_update_interval 每个原因一条数据）

```
{
   "@timestamp" : "2020-09-06T01:28:23.000Z", # 数据更新时间，时区为UTC
   "reason" : "New job",    # 作业等待原因
   "counter" : 2,           # 该等待原因的作业数
   "cluster" : "aip"        # 集群名
}
```

shres-time

Index 名：shres-time (每 resource_update_interval 每种共享资源一条数据)。共享资源由 RESS 提供，主机级的 RESS 参数在 hosts-time 的每个主机的数据中。

```
{
   “@timestamp” : "2020-09-06T01:28:23.000Z", # 数据更新时间，时区为UTC
   “cluster” : "aip",       # 集群名
   “name” : "proxy",        # 资源名
   “value” : 100,           # RESS上报的值
   “rsvValue” : 39          # 被作业占用的值
}
```

queues-time

Index 名：queues-time (每 job_update_interval 每个队列一条数据)。

```
{
   “@timestamp” : "2020-09-06T01:28:23.000Z", # 数据更新时间，时区为UTC
   “cluster” : "aip",       # 集群名
   “name” : "medium",       # 队列名
   “priority” : 3,          # 队列优先级
   “status” : "Ok",         # 队列状态: OK, Open:Inactive, Closed:Active,
                          # Closed:Inactive, 参考cqueues
   “userJLimit” : -1,       # 每个用户的作业槽上限
   “availGPUs” : 1,         # 队列可用GPU数
   “maxAvaliGPUs” : 1,      # 队列最大可用GPU数
   “availMemGB” : 453,      # 队列可用内存总和 (GB)
   “maxAvailMemGB” : 543,   # 队列最大可用内存总和 (GB)
   “availSlots” : 23,       # 队列可用作业槽数
   “maxAvaliSlots” : 100,   # 队列最大可用作业槽数
   “maxJ” : -1,             # 队列配置的作业槽上限
   “nJobs” : 0,             # 等待、运行和暂停的作业槽数
   “nPend” : 0,             # 等待作业所需作业槽数
   “nStop” : 0,             # 暂停作业所占作业槽数
   “nRun” : 0               # 运行作业所占作业槽数
}
```

ugroups-time

Index 名：ugroups-time (每 rusagepreason_interval 每个用户或用户组一条数据)。

```
{
   “@timestamp” : "2020-09-06T01:28:23.000Z", # 数据更新时间，时区为UTC
   “cluster” : "aip",       # 集群名
   “name” : "ug1",          # 用户或用户组名
   “isgroup” : "y",         # 是否为用户组: y = 用户组，n = 用户
   “maxJ” : -1,             # 配置的用户可用作业槽上限
   “nJobs” : 8,             # 等待、运行和暂停的作业的作业槽数
   “nPend” : 8,             # 等待作业所需作业槽数
   “nStop” : 0,             # 暂停作业所占作业槽数
   “nRun” : 0               # 运行作业所占作业槽数
}
```

updatetime

Index 名：updatetime 除 jobs 外各个 index 数据最后更新的时间

```
{
    “index” : "ugroups-time", # index名，如jobs-time
    “last_update” : "2020-09-06T01:28:23.000Z" # 最近一次数据更新时间UTC
}
```

log-daemon 名-master 主机名

Index 名：log-daemin 名-master 主机名调度器主控（master）主机上 daemon 的日志。每个*olmon.conf* 中 logfile 配置的 daemon 名字一个 index

```
{
   "@timestamp": "2025-02-01T23:38:28.000Z",  # 日志时间戳，时区为UTC
   "cluster": "aip",                  # 集群名
   "loglevel": "ERR",                 # 日志级别
   "version": "10.25.0",              # 调度器版本
   "process": "2947471",              # daemon进程号
   "function": "main",                # 产生日志的C函数（function）名
   "message": "main: CBSCHED is started"  # 日志内容
}
```

警告： AIP 以前版本的日志中没有年份，日志上传时 olcol 程序会自动加入当前年份，老的日志的年份可能是错误的。最佳实践是删除老版本 AIP 的日志文件后再启动 olmon 服务。

4.4 AIP DNS

AIP 自身的服务和组件不依赖于 DNS 服务。主机名和 IP 地址的转换由/opt/skyformai/etc/hosts 文件控制。

对于云上动态主机，在加入集群时使用命令*caddhost* 的参数-a 指定新加主机的 IP 地址。这个地址会被加到 master 的 cbls 中，为 AIP 的其他组件提供内置的 DNS 服务。

如果外部的 DNS 有性能问题，而集群作业有需要 DNS 服务，可以用以下命令把/opt/skyformai/etc/hosts 里的内容自动同步到集群每台主机的本地/etc/hosts 中。命令以 root 运行，运行前确保所有主机上的 AIP 服务运行正常:

```
cd /tmp    # 确保当前路径所有主机上都存在
ctask -m all /opt/skyformai/sbin/synchostsfile
```

命令完成后，每台主机上的/etc/hosts 文件会新增以下类似的内容:

```
# --- Sync from AIP hosts file ---
192.168.10.10 dev
192.168.22.11 node01
...
```

以上命令在第一次安装和配置 AIP 后运行一次。在修改了/opt/skyformai/etc/hosts 后，也需要手动运行一次。

CHAPTER 5

参考

5.1 命令

5.1.1 bjobs

bjobs - 显示有关 AIP 作业的信息

概要

与 **cjobs** 相同

描述

显示有关作业的信息。

默认情况下，它的功能与 cjobs 命令完全相同。但是，当 CB_ENVDIR(/opt/skyformai/etc)/jservice.yaml 包含参数“jservice_enabled: yes”时，bjobs 会从 jservice 获取作业信息。从 jservice 获取的信息不包含作业进程信息或作业资源使用情况数据。

使用 jservice 是为了在高吞吐量环境中减轻 AIP 调度程序 cbsched 的负载。

例外

AIP 10.24.1：由于*jservice* 不包含所有作业信息，因此“bjobs -l jobid”将回退到“cjobs -l jobid”。“bjobs -l”或“bjobs -l jobid1 jobid2 …”将继续使用*jservice* 中的数据。

AIP 10.25.0 开始，当 jservice 允许后，bjobs 只从 jservice 获取作业数据。

另请参阅

jservice 、*cjobs*

5.1.2 cacct

命令

cacct - 显示作业统计信息

概要

cacct [**-d**] [**-e**] [**-l** | **-a** *previous_nth_month*] [**-C** *time0,time1*] [**-S** *time0,time1*] [**-L** *time0,time1*] [**-f** “*acctfile_name*”] [**-m** “*host_name …*”] [**-P** “*project_name …*”] [**-q** “*queue_name …*”] [**-u** “*user_name …*” | **-u all**] [**-j** “*jobId …*”]

cacct -ga

cacct [**-h**]

说明

显示调用该命令的用户提交的作业的帐户统计信息摘要。默认情况下，cacct 会显示所有由发出该命令并登录到所有 AIP 帐户文件（work/data/cb.acct*）的用户提交的作业的统计信息。

CPU 时间、作业运行时间、作业等待时间和周转时间均以秒为单位。

该命令从所有/opt/skyformai/work/data/cb.acct* 中读取数据。只有已经完成的作业才会在 cb.acct 文件中有记录。

选项

-d
: 显示退出代码为 0 的作业（即状态为 DONE 的作业）的统计信息。

-e
: 显示退出状态为 EXIT 的作业的统计信息。这包括已运行但以非零状态退出的作业以及等待期间被终止的作业。

-l
: 以多行格式列出每个作业的详细信息。

-a *previous_nth_month*
: 以 JSON 格式生成每月详细使用情况数据以用于计费。参数 *previous_nth_month* 指定当前月份之前的月份数。0 表示从当前月份到执行命令的时间。1 表示上个月，以此类推。例如，-a 0 表示当前月份，-a 1 表示上个月，-a 2 表示上个月前一个月。

例如，’cacct -a2 -u all’生成上个月前一个月的详细使用情况数据。

此选项通常用于生成月度账单。

-C *time0,time1*
显示在指定时间间隔内完成的作业的统计信息。有关时间格式规范，请参阅*chist* 。

-f *cb.acct* 文件路径
使用 *cb.acct* 指定文件路径中的数据。

-ga
以 JSON 格式列出滥用 GPU 的作业。滥用 GPU 的作业是指那些使用调度程序未调度的 GPU 的作业，这些 GPU 要么是已请求但未使用的 GPU，要么是使用了未分配给作业的 GPU。

-L *time0,time1*
显示在指定时间间隔内调度的作业的统计信息。有关时间格式规范，请参阅*chist* 。

-m “*host_name* …”
显示调度到指定主机的作业的统计信息。多个主机名必须用空格分隔，并用引号 (”) 或 (’) 括起来。

-P “*project_name* …”
显示属于指定项目的作业的统计信息。多个项目名称必须用空格分隔，并用引号 (”) 或 (’) 括起来。

-q “*queue_name* …”
显示提交到指定队列的作业的统计信息。多个队列名称必须用空格分隔，并用引号 (”) 或 (’) 括起来。

-S *time0,time1*
显示指定时间间隔内提交的作业的统计信息。有关时间格式规范，请参阅*chist* 。

-u “*user_group* | *user_name* …” | all
只有集群管理员可以使用此选项。如果发出该选项的用户不是集群管理员，则此选项将被忽略。显示由指定用户、用户组或所有用户（如果指定了关键字 all）提交的作业的统计信息。多个用户名必须用空格分隔，并用引号 (”) 或 (’) 括起来。

-j “*jobId* …”
显示具有指定作业 ID 的作业的统计信息。多个作业 ID 必须用空格分隔，并用引号 (”) 或 (’) 括起来。

-h
显示命令使用信息并退出。

输出

在 SUMMARY 部分，它显示以下作业统计信息：

- Total number of done jobs: 退出码为 0 的已完成作业总数
- Total number of exited jobs: 已退出作业总数，包含退出码为非 0 的作业和等待中被杀的作业
- Total number of run jobs: 运行过的作业总数
- Total CPU time consumed: 消耗的 CPU 总时间，包括平均值 (Average)、最大值 (Maximum) 和最小值 (Minimum)
- Total memory used: 使用的内存总值（以 MB 为单位），包括平均值 (Average)、最大值 (Maximum) 和最小值 (Minimum)
- Total job run time: 作业运行总时间，包括平均值 (Average)、最大值 (Maximum) 和最小值 (Minimum)
- Total job wait time in queues: 作业在队列中的等待总时间，包括平均值 (Average)、最大值 (Maximum) 和最小值 (Minimum)
- Average turnaround time: 平均周转时间即从作业提交到完成、退出或被杀的时间，最大和最小周转时间
- Average hog factor of a job: 作业的平均占用率 (CPU 用时/周转时间)、最大和最小占用率
- Total throughput: 总吞吐量，每小时作业数量

- Time window: 所有被统计的作业中，从第一个的提交作业时间，到最后一个结束的作业时间

在包含作业详细信息的长格式中，将显示各个作业的统计信息。其中包括：

- CPU_T：CPU 总时间（单位：MB）秒
- MAX.MEM：最大内存使用量（MB）
- WAIT：作业等待时间
- TURNAROUND：作业周转时间（秒）
- STATUS：完成或退出
- HOG_FACTOR：作业占用率

使用-a 参数的 JSON 输出的格式在以下的例子中:

```
  {
  "User": "root",                                 # 作业用户
  "JobId": "307",                                 # 作业号
  "SubmitTime": 1745347384,                       # 提交时间戳
  "StartTime": 1745347384,                        # 开始运行时间戳，如果从未开始运行，则为0
  "EndTime": 1745347499,                          #␣
↪结束运行时间戳，若无开始时间，则为被杀时间
  "Slots": 1,                                     # 使用的总作业槽数
  "SlotTime": 115,                                # 作业槽使用时间（秒） = 作业槽数 X␣
↪作业运行时长
  "MaxMem": 59800,                                # 最大内存使用KB
  "App": "",                                      # 应用名 csub -A
  "Allocation": [                                 # 调度器为作业分配的资源
    {
      "ResSpec": "",                                  # csub -R参数
      "MinSlots": 1,                                  # csub -n参数
      "MaxSlots": 1,                                  # csub -n参数
      "Hosts": [                                      # 分配的作业槽列表
        {
          "Name": "dev",                              # 作业槽所在主机
          "Port": 16331                               #␣
↪作业槽分配的端口，若分配了GPU，也会列出
        }
      ]
    }
  ],
  "Project": "default",                           # 项目名 csub -P
  "Queue": "medium",                              # 队列名 csub -q
  "Jobname": "",                                  # 作业名 csub -J
  "JobDescription": "",                           # 作业描述 csub -Jd
  "Command": "./calc",                            # 作业命令
  "FromHost": "dev",                              # 提交主机名
  "ResReq": "",                                   # 作业资源需求 csub -R
  "Cwd": "test",                                  # 作业工作目录（相对于HOME）
  "InFile": "",                                   # 标准输入文件路径
  "OutFile": "",                                  # 标准输出文件路径
  "ErrFile": "",                                  # 标准错误文件路径
  "Dependent": "",                                # 作业依赖定义 csub -w
  "CombinedResReq": "1{select[type==\"local\"]order[slots]}",    #␣
↪合并系统，队列，和作业的资源需求
```

(下页继续)

(续上页)

```
  "PreExecCmd": "",                          # 作业前处理命令 csub -E
  "ExitStatus": 0,                           # 作业退出码
  "MinReqSlots": 1,                          # 请求的最小作业槽数 csub -n
  "MaxReqSlots": 1,                          # 请求的最大作业槽数 csub -n
  "RunLimitSec": -1,                         # 作业运行时长限制（秒）csub -
↪W或者队列里的runlimit
  "UserGroup": "test2",                      # 用户组
  "ReqMemGB": 0,                             # 请求的内存预留GB
  "MaxUsedMemGB": 0.0005,                    # 最大使用内存GB
  "ReservedGPUs": 0,                         # 请求的GPU数
  "MaxUsedGPUs": 0,                          # 最多使用的GPU数
  "PendTimeSec": 0,                          # 作业等待总时长（秒）
  "RunTimeSec": 115                          # 作业运行总时长（秒）
}
```

5.1.3 caddhost

命令

caddhost - - 添加动态主机及其属性

概要

caddhost [**-m** 型号] [**-t** 类型] [**-f***CPU* 因子] [**-D** 磁盘数量] [**-R** “资源列表“] [**-w** “窗口“] [**-b** “主机繁忙“] [**-M***MXJ* 最大作业槽数] [**-a** *IPv4* 地址] 主机名

caddhost [**-h** | **-V**]

描述

将单个主机及其属性添加到现有集群，无需重新配置集群。

host_name 参数为必填项。如果未指定其他参数，则会将 host_name 添加到集群。caddhost 可以在主主机 (MASTER) 或任何其他已安装 AIP 包的计算（从属）主机上执行，在这种情况下，必须指定变量 CB_SERVER_HOSTS 以确定主主机的位置。使用其他参数，可以添加以下信息：主机型号、主机类型、CPU 因子、磁盘数量、资源列表、窗口、繁忙列表。请参阅 cb.yaml(5) 和 chosts(1)。

选项

-a ***IPv4_address***
: 使用“1.1.1.1”格式指定主机的 IPv4 地址。如果现有 DNS 或 hosts 文件无法解析主机名，则此选项非常有用。

-m
: 主机模型。如果未指定，则使用检测的主机模型。

-t
: 主机类型。如果未指定，则使用检测的主机类型。

-f
: CPU 因子。如果未指定，则使用检测的 CPU 因子。

-D
主机上安装的本地磁盘数量。如果未指定，则本地磁盘数量自动检测。

-R "资源列表"
集群上定义的资源列表，以空格分隔（标签 tag，静态或动态）。资源需求字符串的大小限制为 512 字节。

-w
主机的运行窗口。如果未指定，则不使用运行窗口。

-M
指定主机的 MXJ（最大作业槽位）。如果未指定，则添加的主机的最大作业槽位不受限制。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

host_name
要添加到集群的单个主机的主机名。若-a 参数没有指定，主机名必须通过 DNS 或 /etc/hosts 文件解析。

输出

显示以下内容：Host host_name 已添加。

在这种情况下，host_name 已成功解析并添加到集群。

否则，如果 host_name 未正确解析，则会显示以下内容：main: invalid hostname host_name

例子

在云上添加动态主机的例子:

```
caddhost -a $(hostname -I | awk '{print $1}') -M 54 $(hostname -s)
```

以上例子中，IPv4 地址和主机名由命令 hostname 获取。

排错

备注： 动态添加的主机在集群的/opt/skyformai/etc/cb.yaml 里没有配置，cbls 运行时必须加 -D 参数。在安装过程中，运行 host-setup –dynamic 会自动运行 caddhost 命令，并确保 cbls 启动时有-D 参数。

5.1.4 cadmin

管理员命令

cadmin - AIP 管理 cbls 和 cbexe 的工具

概要

cadmin 子命令

cadmin [**-h** | **-V**]

子命令列表

ckconfig [**-v**]
reconfig [**-f**] [**-v**]
lsstartup [**-f**] [主机名…| **all**]
lsshutdown [**-f**] [主机名…| **all**]
lsrestart [**-v**] [**-f**] [主机名…| **all**]
lslock [**-l** 时间秒数]
lsunlock
lsdebug [**-c** 类名…] [**-l** 调试级别] [**-f** 日志文件名] [**-o**] [主机名]
lstime [**-l** 计时级别] [**-f** 日志文件名] [**-o**] [主机名]
exestartup [**-f**] [主机名…| **all**]
exeshutdown [**-f**] [主机名…| **all**]
exerestart [**-f**] [主机名…| **all**]
exelogon [主机名…| **all**] [**-c** *cpu_time*]
exelogoff [*host_name* …| **all**]
exedebug [**-c** *class_name* …] [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**] [*host_name*]
exetime [**-l** *timing_level*] [**-f** *logfile_name*] [**-o**] [*host_name*]
help [*subcommand* …]
quit

说明

> **警告:** 此命令仅供 AIP 管理员在 $CB_ENVDIR/cb.yaml 文件发生更改时使用。

cadmin 是一个执行特权命令的工具，用于控制 AIP 集群中的 CBLS 和 CBEXE 操作。

如果没有为 cadmin 提供子命令，cadmin 会提示从标准输入中输入子命令。

对于可以指定多个主机名或主机组的子命令，请勿将多个主机名括在引号中。

选项

subcommand
: 执行指定的子命令。请参阅“用法”部分。

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 打印 AIP 发行版本并退出。

用法

ckconfig[-v]
检查 AIP 配置文件。

-v
显示有关配置文件检查的详细信息。

reconfig[-f] [-v]
在集群中的所有主机上重新启动 CBLS。您应该在更改配置文件后使用 reconfig 命令。在重新启动集群中的所有 CBLS 之前，会检查配置文件。如果配置文件不正确，则不会启动重新配置。

-f
禁用用户交互，并在未发现致命错误的情况下强制在集群中的所有主机上重新启动 CBLS。此选项在批处理模式下非常有用。

-v
显示有关配置文件检查的详细信息。

lsstartup[-f] [*host_name*…|all]
如果未指定参数，则在本地主机上启动 CBLS。

在指定主机或集群中所有主机上启动 CBLS（如果仅提供 all 参数）。系统将要求您确认。

该子命令只能由 root 运行，启动远程主机的 cbls 需要使用 ssh。

-f
禁用交互，启动 CBLS 时不要求您确认。

lsshutdown[-f] [*host_name*…| all]
如果未提供任何参数，则关闭本地主机上的 CBLS。

在指定主机或集群中所有主机上关闭 CBLS（如果指定了 all 参数）。系统将要求您确认。

关闭 CBLS 是向 CBLS 发送关闭请求，由 CBLS 自己退出，所以只需 AIP 管理员权限运行命令。

-f
禁用交互，关闭 CBLS 时不要求您确认。

lsrestart [-v] [-f] [*host_name*…| all]
如果未提供任何参数，则在本地主机上重新启动 CBLS。

如果指定了 all，则在指定主机或集群中的所有主机上重新启动 CBLS。系统将要求您确认。

重启 CBLS 是向 CBLS 发送重启请求，由 CBLS 自己退出后重新启动，所以只需 AIP 管理员权限运行命令。

-v
显示有关配置文件检查的详细信息。

-f
禁用用户交互，并在未发现致命错误的情况下强制 CBLS 重新启动。此选项在批处理模式下很有用。lsrestart -f all 与 reconfig -f 相同。

lslock [-l *time_seconds*]
如果未指定时间，则锁定本地主机上的 CBLS，直到明确解锁。主机锁定后，CBLS 的负载状态将变为 lockU。AIP 不会向锁定的主机发送任何作业。

-l*time_seconds*
主机锁定指定的时间（以秒为单位）。如果计算机正在运行需要所有可用 CPU 时间和/或内存的独占作业，则此功能非常有用。

lsunlock
解锁本地主机上的 CBLS。

exestartup[-f][*host_name* …| all]
如果未指定参数，则在本地主机上启动 CBEXE。

在指定主机或所有主机上启动 CBEXE 如果指定了 all ，则在集群中执行。系统将要求您确认。

该子命令只能由 root 运行，启动远程主机的 cbexe 需要使用 ssh。

-f
禁用交互，并且在启动 CBEXE 时不要求确认。

exeshutdown [-f] [*host_name*…| all]
如果未指定参数，则关闭本地主机上的 CBEXE。

关闭指定主机上的 CBEXE，如果指定了 all ，则关闭集群中所有主机上的 CBEXE。系统将要求您确认。

如果 CBEXE 正在运行，它将一直运行，直到所有远程任务都退出。

关闭 CBEXE 是向 CBEXE 发送关闭请求，由 CBEXE 自己退出，所以只需 AIP 管理员权限运行命令。

-f
禁用交互，并且在关闭 CBEXE 时不要求确认。

exerestart[-f] [*host_name*…| all]
如果未指定参数，则在本地主机上重新启动 CBEXE。

如果指定了 all，则在指定主机或集群中的所有主机上重新启动 CBEXE。系统将要求您确认。

如果 CBEXE 正在运行，它将一直运行，直到所有远程任务退出。在等待远程任务退出期间，将重新启动另一个 CBEXE 来处理新的查询。

重启 CBEXE 是向 CBEXE 发送重启请求，由 CBEXE 自己退出后重新启动，所以只需 AIP 管理员权限运行命令。

-f
禁用交互，并且在重新启动 CBEXE 时不要求确认。

exelogon [*host_name*…| all] [-c *cpu_time*]
如果未指定参数，则记录 CBEXE 在本地主机上执行的所有任务。

记录集群中指定主机或所有主机（如果指定 all）上 CBEXE 执行的任务。

CBEXE 会将任务的资源使用情况信息写入日志文件 cb.tacct.*host_name*。日志文件的位置为/opt/skyformai/log。或者 CBEXE 无法访问该目录，则日志文件将创建在 /tmp 中。

-c *cpu_time*
仅记录 CPU 使用率超过指定数量的任务。CPU 时间由 *cpu_time* 指定，以毫秒为单位。

exelogoff [*host_name*…|all]
如果未指定参数，则关闭本地主机上的 CBEXE 任务日志记录。

关闭集群中指定主机或所有主机（如果指定了 all）的 CBEXE 任务日志记录。

lsdebug [-c "*class_name*… "] [-l *debug_level*] [-f *logfile_name*] [-o] [" *host_name*… "]
设置 CBLS 的消息日志级别，以便在日志文件中包含更多信息。您必须是 root 权限或 AIP 管理员才能使用此命令。

如果该命令未使用任何选项，则使用以下默认值:

class_name = 0（不记录其他类别）

debug_level = 0（参数 CB_LOG_MASK 中的 LOG_DEBUG 级别）

logfile_name = 当前 AIP 系统日志文件，位于 CB_LOGDIR 指定的目录中，格式为: *daemon_name.host_name*.log。

host_name= 本地主机（提交命令的主机）

-c "*class_name*… "
指定要记录调试消息的软件类别。如果指定了类别列表，则必须将它们括在引号中并用空格分隔。

可能的类：

LC_AUTH - 记录身份验证消息

LC_CHKPNT - 记录检查点消息

LC_COMM - 记录通信消息

LC_EXEC - 记录作业执行的重要步骤

LC_FILE - 记录文件传输消息

LC_HANG - 标记程序可能挂起的位置

LC_SIGNAL - 记录与信号相关的消息

LC_TRACE - 记录重要的程序执行步骤

LC_RPC - 记录 XDR 传输的所有内容

默认值：0（不记录其他类）

-l *debug_level*
指定调试消息的详细程度。数字越高，记录的详细信息越多。较高级别包含所有较低级别。

可能的值：

0 - LOG_DEBUG 级别。

1 - 用于扩展日志记录的 LOG_DEBUG1 级别。较高级别包含较低日志记录级别。例如，LOG_DEBUG3 包含 LOG_DEBUG2、LOG_DEBUG1 和 LOG_DEBUG 级别。

2 - 用于扩展日志记录的 LOG_DEBUG2 级别。较高级别包含较低日志记录级别。例如，LOG_DEBUG3 包含 LOG_DEBUG2、LOG_DEBUG1 和 LOG_DEBUG 级别。

3 - 用于扩展日志记录的 LOG_DEBUG3 级别。较高级别包含较低日志记录级别。例如，LOG_DEBUG3 包含 LOG_DEBUG2、LOG_DEBUG1 和 LOG_DEBUG 级别。

默认值：0（LOG_DEBUG 级别）

-f *logfile_name*
指定要记录调试消息的文件名。可以指定带或不带完整路径的文件名。

如果指定了不带路径的文件名，则文件将保存在 /opt/skyformai/log 目录中。

将创建的文件名将采用以下格式：*logfile_name*.*daemon_name*.*host_name*.log

如果/opt/skyformai/log 路径无效，则日志文件将在/tmp 目录中创建。

默认值：当前 AIP 系统日志文件，位于 CB_LOGDIR 指定的目录中，格式为 *daemon_name*.*host_name*.log。

-o
关闭临时调试设置并将其重置为守护进程的启动状态。消息日志级别将重置为 CB_LOG_MASK 的值，类别将重置为 CB_DEBUG_RES 和 CB_DEBUG_LIM 的值。

日志文件将重置为默认日志文件。

"主机名… "
在指定的一个或多个主机上设置调试设置。

默认值：本地主机（提交命令的主机）

exedebug [-c "*class_name* "] [-l *debug_level*] [-f *logfile_name*] [-o] [" *host_name*… "]
设置 CBEXE 的消息日志级别，以便在日志文件中包含其他信息。您必须是 AIP 管理员才能使用此命令，而不是 root 用户。

有关选项的说明，请参阅 lsdebug 的说明。

lstime [-l *timing_level*] [-f *logfile_name*] [-o] [" *host_name* ··· "]

设置 CBLS 的计时级别，以便在日志文件中包含额外的计时信息。您必须是 root 权限或 AIP 管理员才能使用此命令。

如果使用此命令时不带任何选项，则使用以下默认值：

timing_level = 不记录计时信息

logfile_name = 当前 AIP 系统日志文件，位于 CB_LOGDIR 指定的目录中，格式为 *daemon_name*.log.*host_name*

host_name= 本地主机（提交命令的主机）

-l *timing_level*

指定日志文件中包含的计时信息的详细信息。计时消息指示软件中函数的执行时间，并以毫秒为单位记录。

有效值：1 | 2 | 3 | 4 | 5

数字越大，软件中计时并记录执行时间的函数越多。数字越小，包含更常用的软件函数。较高级别包含所有较低级别。

默认值：未定义（不记录计时信息）

-f *logfile_name*

指定要记录计时消息的文件名。可以指定带或不带完整路径的文件名。

如果指定了不带路径的文件名，则文件将保存在 /opt/skyformai/log 目录中。

将要创建的文件名将采用以下格式：*logfile_name*.*daemon_name*.*host_name*.log

如果指定的路径无效，则日志文件将在/tmp 目录中创建。

备注： 计时和调试消息都记录在同一个文件中。

默认值：当前 AIP 系统日志文件位于 /opt/skyformai/log 目录中，格式为 *daemon_name*.*host_name*.log。

-o

关闭临时计时设置并将其重置为守护进程的起始状态。计时级别将重置为相应守护进程的参数值（CB_TIME_LIM、CB_TIME_RES）。

日志文件将重置为默认日志文件。

"*host_name* ··· "

设置指定主机或主机的计时级别。

默认值：本地主机（提交命令的主机）

exetime [-l *timing_level*] [-f *logfile_name*] [-o] [" *host_name* ··· "]

设置 CBEXE 的计时级别，以便在日志文件中包含额外的计时信息。您必须是 AIP 管理员才能使用此命令，而不是 root 用户。

有关选项的说明，请参阅 lstime 的说明。

help [*subcommand* ···] | ? [*subcommand* ···]

显示指定命令的语法和功能。这些命令必须明确地提供给 cadmin。

在命令提示符下，您可以使用 help 或?。

quit

退出 cadmin 会话。

5.1.5 capps

命令

capps - 显示 $CB_ENVDIR (/opt/skyformai/etc) 和 $HOME/.cube 中 cb.apps 的内容

概要

capps [**-h** | **-V**] [**-l**] [**-v**] [*profile ···]*

描述

显示 $CB_ENVDIR (/opt/skyformai/etc) 和 $HOME/.cube 中 cb.apps 的内容

选项

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到标准错误输出并退出。

-l
: 以长格式显示内容。

-v
: 显示 cb.apps 文件中配置错误的行号。

profile
: 显示指定配置应用名称的相关信息。

输出

默认显示

$CB_ENVDIR/cb.apps 和 $HOME/.cube/cb.apps 中指定的配置文件包含以下字段:

PRF_NAME
: 作业配置应用的名称。

LOCATION
: 定义配置文件的 cb.apps 文件的路径。

DESCRIPTION
: 作业配置文件的描述。

BSUB_OPTIONS
: 配置文件中指定的 csub 命令选项

例子:

```
PRF_NAME        LOCATION              DESCRIPTION               BSUB_OPTIONS
fluent_wing      /cubetop/etc         CFD for wing              -n 32 -R span[ptile=4]␣
↪order[-slots]
```

-l 输出

如果指定了 -l 选项，则生成的长格式列表将包含以下附加字段：

环境变量
配置文件定义中指定的其他环境变量及其关联值。

例子:

```
PRF_NAME:          fluent_wing
Description        CFD for wing
Path of cb.apps: /opt/skyformai/etc
CSUB_OPTIONS:      -n 32 -R span[ptile=4] order[-slots]
Environments Vars:
  LSF              = false
  MPI              = /usr/lib64/intelmpi
```

5.1.6 cbot

命令

cbot - 将等待作业相对于队列中的最后一个作业移动

概要

cbot *job_ID* | "*job_ID*[*index_list*]" [*position*]

cbot [**-h** | **-V**]

描述

更改等待作业或等待作业数组元素的队列位置，以影响作业的调度顺序。

默认情况下，AIP 会按照到达顺序（即先到先得）调度队列中的作业，但前提是存在合适的服务器主机。

cbot 命令允许用户和 AIP 管理员手动更改作业的调度顺序。用户只能操作自己的作业，而 AIP 管理员可以操作任何用户的作业。用户只能更改自己作业的相对位置。

如果由 AIP 管理员调用，**cbot** 会将选定的作业移至提交到队列的最后一个具有相同优先级的作业之后。AIP 管理员可以更改队列中所有用户作业的位置。

如果由普通用户调用，**cbot** 会将选定的作业移至提交到队列的最后一个具有相同优先级的作业之后。

cjobs 会按照等待作业的调度顺序显示它们。

选项

***job_ID* | "*job_ID*[*index_list*]"**
必填。要操作的作业或作业数组的作业 ID。

对于作业数组，索引列表、方括号和引号是必需的。索引列表用于操作作业数组。索引列表是一个逗号分隔的列表，其元素的语法为 *start_index*[*-end_index*[**:***step*]]，其中 *start_index*、*end_index* 和 *step* 为正整数。如果省略 step，则假定 step 为 1。作业数组索引从 1 开始。作业数组索引的最大值为 1000。数组中的所有作业共享相同的 job_ID 和参数。数组的每个元素都通过其数组索引来区分。

position
可选。可以指定 *position* 参数来指示作业在队列中的位置。*position* 是一个正数，表示作业距离队列末尾的目标位置。这些位置仅与队列中适用的作业相关，具体取决于调用者是普通用户还是 AIP 管理员。默认值 1 表示该位置位于所有具有相同优先级的其他作业之后。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

另请参见

cjobs、*cswitch*、*ctop*

5.1.7 cchkpnt

命令

cchkpnt - 对一个或多个可设置检查点的作业进行检查点

概要

cchkpnt [**-f**] [**-k**] [**-p** 分钟 | **-p 0**] [作业 *ID*| "作业 *ID*[索引列表]"] …

cchkpnt [**-f**] [**-k**] [**-p** 分钟 | **-p 0**] [**-J** 作业名称] [**-m** 主机名 | **-m** 主机组] [**-q** 队列名称] [**-u** "用户名" | **-u all**] [**0**]

cchkpnt [**-h** | **-V**]

描述

对正在运行 (RUN) 或已暂停（SSUSP、USUSP 和 PSUSP）的可检查点作业执行检查点操作。集群管理员和 root 用户可以对其他用户提交的作业执行检查点操作。

默认情况下，对单个作业、最近提交的作业或同时满足其他指定选项（-m、-q、-u 和 -J）的最近提交的作业执行检查点操作。指定 0（零）可对多个作业执行检查点操作。

指定作业 ID 可对单个特定作业执行检查点操作。

默认情况下，作业在执行检查点操作后会继续执行。

要提交可检查点作业，请使用 csub -k 或将作业提交到检查点队列（*cb.yaml* 中的检查点）。使用*crestart* 启动已执行检查点操作的作业。

AIP 调用 CB_SERVERDIR (/opt/skyformai/sbin) 中的*echkpnt* 可执行文件来执行检查点操作。

选项

0
(零)。对多个作业执行检查点操作。所有满足其他指定选项（-m、-q、-u 和 -J）的作业都将执行检查点操作。

-f
即使存在无法执行检查点操作的条件（这些条件因操作系统而异），也强制对作业执行检查点操作。

-k
在成功执行检查点操作后终止作业。

-p 分钟 |-p 0****
启用定期检查点操作并指定检查点周期，或修改已执行检查点操作的作业的检查点周期。指定 **-p 0** (零）可禁用定期检查点操作。

检查点操作是一项资源密集型操作。为了使您的作业能够继续运行，同时仍提供容错功能，请指定 30 分钟或更长时间的检查点周期。

-J 作业名
仅对具有指定作业名称的作业执行检查点操作。

-m 主机名 | -m 主机组
仅对调度到指定主机的作业执行检查点操作。

-q 队列名
仅对从指定队列调度的作业执行检查点操作。

-u "用户名 "| -u all
仅对指定用户提交的作业执行检查点操作。关键字 all 指定所有用户。如果指定的作业 ID 不是 0（零），则忽略此参数。

***job_ID* | "*job_ID*[*index_list*]"**
仅对指定的作业执行检查点操作。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

示例

% **cchkpnt 1234**

对作业 ID 为 1234 的作业执行检查点操作。

% **cchkpnt -p 2.0 1234**

启用定期检查点操作或将作业 ID 为 1234 的作业的检查点周期更改为 2.0 分钟（2 小时）。

% **cchkpnt -m hostA -k -u all 0**

由 root 或集群管理员发出时，将执行检查点操作并终止 hostA 上所有可执行检查点操作的作业。当主机需要关闭或重启时，此功能非常有用。

另请参见

csub 、*cmod* 、*crestart* 、*echkpnt* 、*erestart*

5.1.8 ccluster

命令

ccluster - 显示当前 AIP 集群的配置信息

概要

ccluster [**-h** | **-V**] [**-l**]

描述

显示当前 AIP 集群的配置信息。

选项

-l
: 长格式。显示更多信息。

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到标准错误输出并退出。

默认输出

信息包括:

CLUSTER_NAME
: 集群名称。

STATUS
: 集群状态，值总是 ok。

MASTER_HOST
: 集群主管理主机

Admin
: 主集群管理员的用户名

HOSTS
: 集群中主机总数（包括配置的静态客户端主机）

SERVERS
: 集群中的服务器主机数量。

长格式 (-l)

如果指定此选项，该命令还会列出所有集群管理员用户名、可用的资源名称、主机类型、和主机型号。

5.1.9 cdexe

命令

cdexe - 在作业的 Docker 容器中运行交互式命令

概要

cdexe [**-t** *container_rank_id*] *jobId command* [*arguments* ···]

cdexe [**-h | -V**]

描述

在作业的 Docker 容器中运行交互式命令。

该命令相当于容器作业的 “docker exec”，但它不支持任何 docker-exec 选项。在普通用户没有权限运行 docker 命令的环境中，这个命令可以代替用户运行 docker exec。

发出该命令的用户必须是作业所有者。管理员无法针对其他用户的作业运行此命令。

示例:

```
cdexe -t2 1080 bash
[cadmin@dev /]#
```

上述命令在作业 ID 为 1080 的 Docker 容器中运行 bash。

选项

-t 容器顺序数
对于多容器作业，指定容器的顺序数。

如果未指定，则使用第一个容器等级的容器。

容器顺序基于 “cjobs -l” 或 “aip j i -l” 输出中列出的容器任务顺序。

容器顺序数从 1 开始。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

另请参阅

csub

诊断

cdexe 尝试使用 CBEXE 运行任务。如果 CBEXE 已关闭，cdexe 将失败。

5.1.10 cdimport

命令

cdimport - 将 Docker 镜像从公共镜像仓库导入本地镜像仓库

概要

cdimport 源 目标

cdimport [**-h | -V**]

描述

为 AIP 用户执行 “docker pull 源”、“docker tag 目标” 和 “docker push 目标” 三个命令。如果源或目标需要身份验证，请在执行此命令之前运行 **cdlogin**。

源和目标的语法遵循 “docker” 命令的语法。对于本地 Harbor 镜像仓库，目标通常为 *localserver:port/project/image_name*。

示例:

```
cdimport docker.io/rpmbuild/centos7 harbor_server:443/library/centos7
```

选项

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到标准错误输出并退出。

另请参见

csub cdlogin

5.1.11 cdlogin

命令

cdlogin - 登录到内部 LDAP 连接的容器仓库服务器

概要

cdlogin [**-u** *用户名*] [**-p** *密码*] *服务器*

cdlogin [**-h** | **-V**]

描述

登录到内部 LDAP 连接的容器仓库服务器，例如 Harbor 或任何第三方容器仓库服务。

默认情况下，用户名是发出命令的用户的名称，密码是存储在 AIP 中的密码。使用 -u 指定其他用户名，使用 -p 指定密码。如果指定了 -u 选项，但未指定 -p 选项，则命令将提示输入密码。如果使用当前用户，而 AIP 服务器没有用户的 LDAP 密码，请运行命令“cpasswd -p password”将密码添加到 AIP 服务器。

只有从私有容器仓库拉取或推送数据时才需要登录。访问公共 Docker 镜像无需登录。

示例:

```
cdlogin harbor:443
cdlogin -u myname registry.mycompany.com:1633
```

成功后，该命令会显示“登录成功”。用户随后可以使用在私有仓库中注册的镜像提交容器作业。

选项

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发布版本打印到标准错误输出并退出。

-u 用户名
: 指定登录用户名。

-p 密码
: 指定登录密码。

另请参见

csub cpasswd cdimport

5.1.12 cexpand

命令

cexpand - 为作业请求额外的作业槽位

概要

cexpand [**-R** *res_requirement_spec*] *job_ID*

cexpand [**-h** | **-V**]

描述

为作业请求额外的作业槽位，即增加作业的大小。

默认情况下，不使用 -R 选项时，将作业的大小增加一个槽位。如果原始作业的资源需求包含多个部分，则该槽位将被添加到资源需求的第一个部分（第 0 部分）。

选项

-R 资源需求
指定作业中每个资源需求部分可增长的最小和最大槽位。

资源需求的格式为“[min,]max{} [min,]max{} …”。要请求扩展资源需求的第 N 部分，此处指定的资源需求规范需要包含 N-1 次“0{}”，以指示无需扩展作业的前 N-1 个资源需求部分。

示例:

```
cexpand -R 0{} 5{} 101
```

表示将作业 101 资源需求的第二部分的作业槽增加 5。

job_ID
需要扩展的作业 ID。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

另请参见

csub 、*creduce*

5.1.13 cgadd

命令

cgadd - 创建作业组

概要

cgadd [**-L** limit] *作业组名*

cgadd [**-h** | **-V**]

描述

使用由 *作业组名*指定的作业组名称创建一个作业组。

您必须为新作业组提供完整的组路径名。路径的最后一部分是要创建的新组的名称。

在父作业组下创建子组之前，无需创建父作业组。如果作业组层次结构中不存在任何组，则所有组都将按指定的层次结构创建。

父子组之间以斜杠（/）隔开，如 /aaa/bbb，“aaa”为父作业组，“bbb”为子作业组，类似 Linux 里的文件目录结构。

选项

-L ***limit***
: 指定允许在作业组（包括子组）下运行的并发作业槽的最大数量。该选项限制作业组下已启动作业（RUN、SSUSP、USUSP）的数量。请指定一个介于 0 和 INT32_MAX-1 (2147483647) 之间的正数。如果指定的限制为零，则该作业组下的任何作业都无法运行。

  您无法为根作业组（/）指定限制。根作业组没有作业限制。添加的未指定限制的作业组将继承现有父作业组的任何限制。该选项仅限制创建的最低级别作业组。

  默认情况下，作业组没有作业槽位限制。限制在 cbsched 重启或重新配置后仍然有效。

job_group_name
: 作业组名称的完整路径。

-h
: 将命令用法打印到 stderr 并退出。

-V
: 将 AIP 发行版本打印到 stderr 并退出。

示例

在根组 / 下创建一个名为 mygroup 的作业组:

```
cgadd /mygroup
```

在作业组 /mygroup 下创建一个名为 subgroup1 的作业组:

```
cgadd /mygroup/subgroup1
```

另请参阅

cgdel 、*cgmod* 、*cjgroup*

5.1.14 cgdel

命令

cgdel - 删除作业组

概要

cgdel *作业组名*

cgdel [**-h** | **-V**]

描述

删除由 *作业组名*指定的作业组。

必须提供要删除的作业组的完整组路径名。只有在清理期结束后，从 cbsched 内存中清除所有属于该组的作业，并且删除所有子组后，删除操作才会生效。

可以显式或隐式创建作业组：

(1) 使用 cgadd 命令显式创建作业组。

(2) 当指定的组不存在时，使用 csub -g 或 cmod -g 命令创建作业组。

选项

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到标准错误输出并退出。

另请参阅

cgadd 、*cgmod* 、*cjgroup*

5.1.15 cgmod

命令

cgmod - 修改作业组

概要

cgmod [**-L** limit | **Ln**] 作业组

cgmod [**-h** | **-V**]

描述

使用由 作业组指定的作业组名称修改作业组。

只有 root 用户、AIP 管理员、作业组创建者或其父作业组的创建者可以使用 cgmod 修改作业组限制。

您必须提供被修改的作业组的完整组路径名。路径的最后一个部分是要修改的作业组的名称。

选项

-L *limit*
: 将 作业组的槽位限制更改为指定的 *limit* 值。如果该作业组有父作业组，则新的限制不能超过任何更高级别作业组的限制。同样，如果该作业组有子作业组，则新的限制值必须大于任何较低级别作业组的限制。

  limit 指定允许在作业组（包括子组）下运行的并发作业槽位的最大数量。该选项限制该作业组下已启动的作业（RUN、SSUSP、USUSP）的数量。请指定一个介于 0 和 INT32_MAX-1 之间的正数（2147483647）。如果指定的限制为零，则该作业组下的任何作业都无法运行。

  您无法为根作业组指定限制。根作业组没有作业限制。该选项仅限制指定的最低级别作业组。

-Ln
: 移除该作业组的现有作业限制。如果该作业组有父作业组，则修改后的作业组将自动继承其直接父作业组的任何限制。

作业组
: 作业组名称的完整路径。

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到标准错误输出并退出。

示例

修改组 /mygroup/subgroup/g1 的限制。此操作不会修改 /mygroup 或 /mygroup/subgroup 的限制。

```
cgmod -L 5 /mygroup/subgroup/g1
```

要修改 /mygroup 或 /mygroup/subgroup 的限制，您需要指定确切的组名称:

```
cgmod -L 5 /mygroup
```

或:

```
cgmod -L 5 /mygroup/subgroup1
```

另请参阅

cgadd 、*cgdel* 、*cjgroup*

5.1.16 chinfo

命令

chinfo - 显示主机和静态资源和 GPU 信息

概要

chinfo [**-w** | **-l** | **-e** | **-j**] [**-o** “格式”] [**-R** “资源需求”] [主机名] …

chinfo -s[共享资源名…]

chinfo -g

chinfo [**-h** | **-V**]

描述

显示主机的静态资源或 GPU 信息。

默认情况下，返回以下信息：主机名、主机类型、主机型号、CPU 因子、CPU 数量、总内存、总交换空间、主机是否为服务器主机以及静态资源。显示集群中所有主机的信息。请参阅 cb.yaml(5)。

-s 选项显示静态共享资源及其关联主机的信息。

-g 选项显示 GPU 及其关联主机的信息。

选项

-w
以宽格式显示主机信息。字段显示时不会截断。

-e
以多行长格式显示包含负载在内的扩展主机信息。它比使用 -l 选项显示更多信息。

-g
显示集群中的 GPU 信息。

-l
以长多行格式显示主机信息。除了默认字段外，还显示有关最大 /tmp 空间、本地磁盘数量、远程作业的执行优先级、负载阈值和运行窗口的信息。

-j
输出 JSON。

-o “字段名称…[delimiter=’分割符’]”
通过字段名称指定自定义输出格式。使用 *delimiter=* 来设置在不同标题和字段之间显示的分隔符。分隔符必须是单个字符。默认情况下，分隔符为空格。

如果字段没有值，则输出为短划线 (-)。

可用的字段名称包括：

host_name: 主机名

model: 主机的 CPU 型号

type: 主机类型，如 x86_64Linux, aarch64Linux 等

ncpus: CPU 或 vCPU 的数量，取决于 cb.yaml 中 define_ncpus 的值。define_ncpus 是 cores 则显示物理核数，否则显示逻辑核数

maxmem: 最大内存

maxswp: 最大交换空间

maxtmp: 最大 /tmp 空间

attr|server: 主机属性：Server（普通主机）, Dynamic（动态主机）, Remote（远程不运行 AIP 服务的主机）, 或 Client（客户端）。

-R "*res_req* "
仅显示满足资源需求表达式的主机信息。有关资源需求的更多信息，请参考资源需求字符串的大小限制为 512 字节。

AIP 支持对所有负载索引（包括外部负载索引，无论是静态的还是动态的）的资源需求进行排序。

主机名……
仅显示指定主机的信息。指定多个主机时请勿使用引号。

主机名可以采用 xxxx[001-100] 的格式指定。

-s[共享资源名…]
显示指定资源的信息。这些资源必须是静态共享资源。返回以下信息：资源名称、资源值以及资源位置。如果未指定共享资源，则显示所有共享资源的信息。

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 发布版本打印到 stderr 并退出。

输出

缺省显示的主机信息

显示以下字段：

HOST_NAME
主机名。如果主机名太长，则会被截断。

type
主机类型。如果主机类型太长，则会被截断。其值为 $archLinux 或 $archWindows，其中 $arch 是操作系统的值，如 x86_64、aarch64 等。

model
主机 CPU 型号。如果字串太长，则会被截断。

cpuf
CPU 因子。CPU 因子用于缩放 CPU 负载值，以便将 CPU 速度的差异考虑在内。CPU 速度越快，CPU 因子就越大。CPU 因子的数值是 Linux 的 lscpu 里的 BogoMIPS 除以 100。

未知主机类型的主机的 CPU 因子为 1.0。

ncpus
CPU 或 vCPU 的数量，取决于 cb.yaml 中 define_ncpus 的值。define_ncpus 是 cores 则显示物理核数，否则显示逻辑核数。

ngpus
GPUs 个数。若单张 GPU 卡上有两个 GPU，这数值为 2。

maxmem
最大内存。

maxswp
最大交换空间。

attr
“Server” 是指在*cb.yaml* 中配置的静态服务器主机。

“Remote” 是指通过主服务器或指定的代理主机进行代理的远程主机。

“Dynamic” 是指动态添加的主机。

“Client” 是指客户端主机。

RESOURCES
可用的布尔资源（以资源名称表示）以及外部数字和字符串静态资源的值。有关如何配置外部静态资源，请参阅*cb.yaml* 和*ress* 。

主机-l 或-e 选项长格式输出

除了上述字段外，-l 或-e 选项还显示以下内容：

HOST_IP
AIP 用于通讯的配置在/opt/skyformai/etc/hosts 里的主机 IPv4 地址。

ndisks
本地磁盘的数量。

maxtmp
主机上配置的最大 /tmp 空间（以 MB 为单位）。

rexpri
远程执行（ctask, runtask）优先级。这个指标总是 0。

本机共享资源
静态共享资源。

CPU Model
CPU 型号全名。

Sockets
CPU 颗数。

Cores per Socket
每颗 CPU 上的物理核数。

GPUs
GPU 信息列表。包含 ID、MODEL（型号）、TotalMem（最大显存）、FreeMem（可用显存）、Temp(C)（温度）、和 Power(W)（功耗瓦数）。

RUN_WINDOWS：
这个参数没有意义，只是为了保持与 LSF 的兼容性，值总是 always open。

LOAD_THRESHOLDS
这个参数没有意义，只是为了保持与 LSF 的兼容性，值总是空。

除了上述字段外，-e 选项还显示以下内容：

Core Ut
每个物理核或逻辑核（取决于 define_ncpus 的值，参考*cb.yaml* ）上的利用率。

Network Traffic：
各个网卡上的输出（send）和输入（recv）的瞬间速率（每秒 KB）。值是运行命令时 5 秒内的值。

Local Disk IO：
本地磁盘上的读（read）和写（write）的瞬间速率（每秒 KB）。值是运行命令时 5 秒内的值。

Master Candidate
如果是主控制主机（master），则显示这个信息，并显示用于 AIP key 的 Host ID。

资源选项-s 输出

显示静态共享资源。每行提供静态共享资源的值及其关联的主机。有关如何配置静态共享资源，请参阅*cb.yaml* 和*ress* 。

显示以下字段：

RESOURCE
资源名。

VALUE
静态资源的值。

LOCATION
与静态共享资源关联的主机。

GPU 资源选项-g 输出

显示集群中的 GPU 信息。

显示以下字段：

HOST_NAME
主机名。

ID
GPU 的 ID。

MODEL_NAME
GPU 型号。

TotMem
GPU 最大显存 MB。

FreeMem
GPU 可用显存 MB。

Temp
GPU 温度。

GUT
GPU 利用率。最小 0，最大 1。

MUT
GPU 显存利用率。最小 0，最大 1。

POWER
GPU 功耗瓦数。

5.1.17 chist

命令

chist - 显示 AIP 作业的历史记录

概要

chist [**-h**] [**-V**] [**-l**] [**-b**] [**-w**] [**-a**] [**-d**] [**-e**] [**-p**] [**-s**] [**-r**] [**-f** *cb.data 文件名* | **-n** *cb.data 文件数* | **-n** 最小日志文件, 最大日志文件] [**-C** 时间 *0*, 时间 *1*] [**-S** 时间 *0*, 时间 *1*] [**-D** 时间 *0*, 时间 *1*] [**-N** 主机规范] [**-P** 项目名称] [**-q** 队列名称] [**-m** 主机名] [**-J** 作业名] [**-u** 用户名 | **-u all**] [**-j**] [**-o** “格式”] [*jobId* | “*jobId[index]*” …]

chist [**-h**] [**-V**] **-t** [**-f** *cb.data 文件名*] [**-T** 时间 *0*, 时间 *1*] [**-j**]

描述

显示作业的历史记录。如果不带选项，则默认情况下，它会显示调用此命令的用户所拥有的所有等待、暂停和正在运行的作业的历史记录。如果用户没有特殊权限，chist 只能查看用户自己的作业历史。

选项

-h
将命令使用情况打印到 stderr 并退出。

-V
将 AIP 发布版本打印到 stderr 并退出。

-b
以简要格式显示作业历史记录。有关默认值，请参阅 -l。

-j
以 JSON **格式**显示作业历史记录。此选项不能与 -b 或 -l 选项一起使用。

-l
以长格式显示作业历史记录。每个作业的详细信息将以多行形式显示。如果未指定 -l 或 -b 选项，则仅显示“概要”中的字段（见下文）。

-w
以宽格式显示作业历史记录。

-a
显示已完成和未完成的作业。此选项会覆盖 -d、-p、-s 和 -r 选项。如果 -a 和 -d 均未指定，则不会显示已完成的作业。

-d
仅显示已完成的作业。

-e
仅显示退出（退出码为非 0 或等待是被杀）的作业。

-p
仅显示等待的作业。

-s
仅显示暂停的作业，并显示每个作业暂停的原因（如果指定了选项 -l 或 -b）。

-r
仅显示正在运行的作业。

-f *cb.data* 文件路径
显示指定事件日志文件中的作业历史记录信息。可以指定绝对路径名或相对路径名。默认使用 AIP 系统当前使用的事件日志文件：/opt/skyformai/work/data/cb.data。选项 -f 通常用于离线分析。

如果设置了环境变量 CB_SHAREDIR，chist 将在 $CB_SHAREDIR/data 中搜索文件。

-n *num_logfiles*
指定 chist 搜索的 cb.data 文件数量。将搜索最近的 num_logfiles 个 cb.data 文件。默认值为 1；即搜索当前事件 cb.data 文件 $(CB_SHAREDIR)/data/cb.data。如果 num_logfiles 为 2，chist 将搜索事件日志文件 cb.data 和 cb.data.（最近的时间戳）；如果 num_logfiles 为 3，chist 将搜索 cb.data 以及另外两个 cb.data.timestamp；依此类推。如果 num_logfiles 为 0，则搜索 $(CB_SHAREDIR)/data 中的所有 cb.data 文件。

-n *min_logfile,max_logfile*
指定 chist 搜索的 cb.data 文件范围。min_logfile 和 max_logfile 的值表示 cb.data 的索引。例如，2 表示 cb.data.(第 2 个带时间戳的文件)。

-C *time0,time1*
仅显示完成或退出时间在 time0 和 time1 之间的时间间隔内的作业（请参阅下面的时间格式）。

-S *time0,time1*
仅显示提交时间在 time0 和 time1 之间的时间间隔内的作业（请参阅下面的时间格式）。

-D *time0,time1*
仅显示开始运行时间在 time0 和 time1 之间的时间间隔内的作业（请参阅下面的时间格式）。

-N *host_spec*
以标准化值显示 CPU 时间。host_spec 可以是主机名、AIP 中定义的主机型号名称或 CPU 因子（使用*cinfo* 获取主机型号和 CPU 因子信息）。如果离线使用 chist，即不使用 CBLS，则 host_spec 的唯一有效用法是 CPU 因子。指定 host_spec 的适当 CPU 缩放因子用于标准化作业消耗的实际 CPU 时间。

-P *project_name*
仅显示属于指定项目 project_name 的作业。

-q *queue_name*
仅显示提交到指定队列 queue_name 的作业。

-m *host_name*
仅显示分派到指定主机 host_name 的作业。

-u *user_name* | -u all
只有集群管理员可以使用此选项。如果发出命令的用户不是集群管理员，则此选项将被忽略。显示由 user_name 指定的用户或所有用户（如果指定了保留用户名 all）提交的作业。默认显示调用此命令的用户提交的作业。

-t
按时间顺序显示作业事件。默认情况下仅显示上周的记录。如需显示不同时间段，请使用 -t 和 -T 选项。

-J *job_name*
显示具有指定作业名为 job_name 的作业。

-o “字段名称…[delimiter= ‘字符’]”
通过字段名称指定自定义输出格式。使用 *delimiter*= 指定在不同标题和字段之间显示的分隔符。分隔符必须是单个字符。默认情况下，分隔符为空格。

如果字段没有值，则输出为破折号 (-)。

可用的字段名称有：

jobid|jd：作业 ID

stat：作业状态

用户：提交用户

app：csub -A 指定的应用程序名称

user_group|ugroup：作业用户所属的用户组

queue：作业提交到的队列名称

job_name|name：csub -J 指定的作业名称

job_description|description：csub -Jd 指定的职位描述

proj_name|proj|project：csub -P 指定的项目名称

job_group|group：csub -g 指定的工作组名称

dependency：由 csub -w 指定的依赖关系

job_priority|priority：由 csub -sp 指定的作业优先级

command|cmd：作业命令

stripped_cmd|scmd：不带 #CSUB 选项的作业命令

pre_exec_command|pre_cmd：由 csub -E 指定的预执行

exit_code：作业退出代码

from_host：作业提交主机名

first_host：第一个作业执行主机

exec_host：作业执行主机

nexec_host：作业的执行主机数量

specified_start_time|sstart_time：csub -b 指定的作业开始时间

specified_terminate_time|sterminate_time：csub -t 指定的作业终止时间

finish_time：作业完成时间

cpu_used：使用的 CPU 时间

run_time：作业运行总时间（秒）

slots|nalloc_slot：已使用的槽总数

mem：已用内存（KB）

maxmem：完成作业所需的最大内存（KB）

avgmem：完成作业所用的平均内存（KB）

memlimit：csub -M 指定的内存限制

swap：操作系统为该作业分配的虚拟内存（以 KB 为单位）

swaplimit：由 csub -v 指定的虚拟内存限制（以 KB 为单位）

min_req_proc|nreq_slot：csub -n 请求的最小槽数

max_req_proc：csub -n 请求的最大插槽数

resreq：csub -R 指定的资源需求

filelimit：由 csub -F (ulimit -f) 指定的文件大小限制

corelimit：csub -C（ulimit -c）指定的核心文件大小限制

stacklimit：由 csub -S (ulimit -s) 指定的堆栈大小限制

processlimit：csub -p（ulimit -u）指定的最大用户进程数

input_file：csub -il-is 指定的输入文件名

output_file：csub -ol-oo 指定的输出文件名

error_file：csub -el-eo 指定的错误文件名

sub_cwd：运行 csub 或 csub -cwd 时的当前工作目录

exec_home：作业提交的主目录

exec_cwd：作业正在执行的当前工作目录

effective_resreq|eresreq：有效资源需求

pend_time：作业等待总时间（秒）

psusp_time：提交等待的总时间（秒）

ususp_time：用户暂停的时间（秒）

ssusp_time：调度程序暂停的时间（秒）

unkonwn_time：作业状态未知的时间（秒）

total_time：从提交到现在或作业结束的时间（以秒为单位）

usage_time：时隙时间（秒）* run_time

jobId …

仅显示指定的作业。此选项覆盖除 -N、-h 和 -V 之外的所有其他选项。与 -J 一起使用时，仅显示此处列出的具有指定 job_name 的作业。

每个作业在长格式输出（-l）最后的统计

统计该作业在各个状态下所花费的时间，单位为秒：

PEND

作业调度前的总等待时间（不包括用户暂停时间）

PSUSP

等待作业的用户暂停总时间

RUN

作业的总运行时间

USUSP

作业调度后用户暂停的总时间

SSUSP

作业调度后系统总暂停时间

UKNWN

作业的总未知时间（如果执行主机上的 cbjm 暂时无法访问，则作业状态变为未知）。

TOTAL

作业在所有状态下所花费的总时间；对于已完成的作业，这是周转时间（即从作业提交到作业完成的时间间隔）。

时间格式

-C、-S 和 -D 选项中的 *time0*,*time1* 必须符合以下内容：

time_form = ptime, ptime | ptime, |, ptime | itime

ptime = 日 | /日 | 月/ | 年/月/日 | 年/月/日/ | 时: | 月/日 | 年/月/日/时: | 年/月/日/时: 分 | 日/时: | 月/日/时: | 日/时: 分 | 时: 分 | 月/日/时: 分 | . | .-itime

itime = ptime 日，月，时，分 = 两位数

其中“ptime”代表特定的时间点，“itime”代表特定的时间间隔，“.”代表当前的月/日/时：分。

牢记以下规则将帮助您自由指定时间：

- 年份必须是 4 位数字，后跟 /
- 月份后面必须跟一个 /
- 日期前面必须跟一个 /
- 小时后面必须跟一个:
- 分钟前面必须跟一个:
- 当日期单独存在或日期后面跟一个 / 小时时，日期前的/ 可以省略：
- 时间格式中不允许有空格，也就是说，时间必须是单个字符串。

上述时间格式是为了方便灵活地指定时间而设计的。

请参阅以下示例：

假设当前时间是 2025 年 3 月 9 日 17:06:30。

1,8
: 从 2025 年 3 月 1 日 00:00:00 到 2025 年 3 月 8 日 23:59:00；从记录第一项工作的时间到 2025 年 3 月 4 日 23:59:00 的,4 或,/4；

6 或 /6
: 从 2025 年 3 月 6 日 00:00:00 到 2025 年 3 月 6 日 23:59:00；

2/
: 从 2025 年 2 月 1 日 00:00:00 到 2025 年 2 月 28 日 23:59:00；

12:
: 从 2025 年 3 月 9 日 12:00:00 到 2025 年 3 月 9 日 12:59:00；

2/1
: 从 2025 年 2 月 1 日 00:00:00 到 2025 年 2 月 1 日 23:59:00；

2/1,
: 从 2 月 1 日 00:00:00 到当前时间；

,. 或者,
: 从记录第一份工作的时间到当前时间；

,-2
: 从记录第一份工作开始到 2015 年 3 月 7 日 17:06:30；

,-2/
: 从记录第一份工作开始到 2015 年 1 月 9 日 17:06:30；

,2/10:
: 从记录第一份工作开始到 2015 年 3 月 2 日 10:59:00；

2024/11/25,2025/1/25
从 2024 年 11 月 25 日 00:00:00 到 2025 年 1 月 25 日 23:59:00。

5.1.18 chosts

命令

chosts - 显示调度器内主机的静态和动态信息

概要

chosts[**-w** | **-l** | **-c** | **-S** | **-E**] [**-j**] [**-o** “格式”][**-R** “资源需求“] [主机名 | 主机组] …

chosts -s [共享资源…]

chosts[**-h** | **-V**]

描述

显示主机信息。

默认情况下，返回所有主机的以下信息：主机名、主机状态、作业槽位限制以及作业状态统计信息。

-s 选项显示数值共享资源及其关联主机的信息。

显示主机信息范围

- root 或集群管理员可以访问所有主机信息。
- 当 cb.yaml 中 user_view_alljobs: yes，普通用户可以访问所有主机信息。
  - 当环境变量 CB_LIMIT_VIEW（或者在/opt/skyformai/etc/cb.conf 中设置）设任何值，普通用户只能访问自己可用队列所含主机的信息。

例子:

```
[u001@aipm ~]$ chosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV    ␣
↪gpu
aipg               ok              -    300      0      0      0      0      0  0.
↪000
aipm               ok              -    300      0      0      0      0      0  4.
↪000
centos6            ok              -    300      0      0      0      0      0  0.
↪000
node111            ok              -    300      0      0      0      0      0  0.
↪000
u22                ok              -    300      0      0      0      0      0  8.
↪000
w19                ok              -    300      0      0      0      0      0  0.
↪000
[u001@aipm ~]$ CB_LIMIT_VIEW=1 chosts
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV    ␣
↪gpu
```

(下页继续)

(续上页)

```
aipg                ok              -     300      0      0      0      0      0  0.
↪000
aipm                ok              -     300      0      0      0      0      0  4.
↪000
```

- 当 cb.yaml 中 user_view_alljobs: no，或者该参数没有设置，用户只能访问自己可用队列所含主机的信息。

选项

-w
以宽格式显示主机信息。字段显示时不会截断。

-l
以（长）多行格式显示主机信息。除了默认字段外，还显示有关 CPU 因子、调度窗口、当前负载和负载阈值的信息。

-j
显示每台主机上正在运行或暂停的作业 ID。短格式或宽格式输出仅显示作业 ID。长格式输出中，作业信息的格式为：作业 ID: 状态 (槽位数量)，例如 1743:RUN(2)

-c
仅显示由 AIP 管理员关闭的主机，以及正在运行的作业数量和管理员在命令“csadmin hclose”中发布的消息。

-o “输出项…[delimiter=’字符’]”
通过字段名称指定自定义输出格式。使用 *delimiter=* 来设置不同标题和字段之间显示的分隔符。分隔符必须是单个字符。默认情况下，分隔符为空格。

如果字段没有值，则输出为短划线 (-)。

可用的字段名称包括：

host_name: 主机名

status: 主机状态。详见 ** 输出 **

max: 主机上配置的最大作业槽数

njobs: 以被占用的作业槽数量

run: 正在运行的作业槽数量

ssusp: 调度器暂停的作业槽数量

ususp: 用户暂停的作业槽数量

rsv: 调度器保留的作业槽数量

hgroups: 主机所属的主机组

queues: 主机所属的队列名称

-S
显示不同状态的主机数量摘要。

-E
显示状态为“Unavail”和“Unreach”的主机名。

-R “*res_req* “
仅显示满足资源需求表达式的主机信息。有关资源需求的更多信息，资源需求字符串的大小限制为 512 字节。

AIP 支持对所有负载索引（包括静态或动态的外部负载索引）的资源需求进行排序。

主机名…| 主机组名…
仅显示指定主机或主机组的信息。对于主机组，将显示属于该组的主机名称，而不是主机组的名称。指定多个主机或主机组时，请勿使用引号。

host_name 可以采用 xxxx[001-100] 的格式指定。

-s [共享资源名…]
显示指定共享资源的信息。资源必须为数值。返回以下信息：资源名称、资源总量和预留量，以及资源位置。如果未指定共享资源，则显示所有数值共享资源的信息。

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 发布版本打印到 stderr 并退出。

输出

主机信息缺省输出

显示以下字段：

HOST_NAME
主机名称。如果主机正在运行作业，并且该主机已从配置中移除，则主机名将显示为 lost_and_found。

STATUS
主机的当前状态。作业只能调度到状态为 ok 的主机。主机状态的可能值如下：

ok
主机可以接受作业。

unavail
主机已关闭，或者主机上的 CBLS 和 CBJM 不可达。

unreach
主机上的 CBLS 正在运行，但 CBJM 无法访问。

closed
主机不允许接受任何远程作业。主机关闭的原因有多种（请参阅基于主机的 -l 选项）。

JL/U
主机可以为每个用户处理的最大作业槽位数量。

这些作业槽位可供正在运行的作业以及已预留槽位的暂停或等待作业使用。

MAX
主机可以处理的最大作业槽位数量。这些作业槽位供主机上正在运行和暂停的作业使用，以及供主机上已预留作业槽位的等待作业使用。

NJOBS
主机上已启动作业（包括正在运行和已暂停的作业）占用的作业槽位数量。

RUN
主机上运行的作业所使用的作业槽数。

SSUSP
主机上系统暂停的作业所使用的作业槽数。

USUSP
用户暂停的作业在主机上占用的作业槽位数量。作业可以由用户或 AIP 管理员暂停。

RSV
主机上预留了作业槽位的等待作业所使用的作业槽位数量。

主机信息-l 长格式输出

除了上述字段外，-l 选项还显示以下内容：

STATUS

closed
-l 选项显示的长格式给出了主机关闭的可能原因：

closed_Adm
该主机已被 AIP 管理员或 root 用户关闭（请参阅*csadmin* ）。任何作业都无法调度到该主机，但正在该主机上执行的作业不会受到影响。

closed_Lock
主机已被 AIP 管理员或 root 锁定（请参阅*cadmin* ）。主机上的所有作业均已被 AIP 暂停。

closed_Wind
主机通过其调度窗口关闭，调度窗口在配置文件 cb.yaml(5) 中定义。主机上的所有作业均被 AIP 系统暂停。

closed_Full
已达到主机上配置的最大作业槽数（请参阅下面的 MAX 字段）。

closed_Excl
主机目前正在执行独家任务。

closed_Busy
主机已过载，因为某些负载指标超出了配置的阈值（参见*cb.yaml* ）。显示的导致主机繁忙的阈值前面带有星号 (*)。

closed_LS
主机上的 CBLS 无法访问，但 CBJM 正常。

closed_Power
主机已被管理员关闭电源。

ok_Power
主机已根据省电调度策略关闭。等待的作业可以将其启动。

suspending
主机正在由管理员或省电策略关闭。

resuming
主机正在由管理员或省电策略启动。

CPUF
显示主机的 CPU 标准化因子（请参阅*chinfo* ）。

DISPATCH_WINDOWS
显示每个主机的调度窗口。调度窗口是每周可以在每个主机上运行作业的时间窗口。已启动的作业不受调度窗口的影响。调度窗口的默认值为无限制或始终开放（即每周 7 天，每天 24 小时）。有关调度窗口的规范，请参阅*cqueues* 中 -l 选项下 DISPATCH_WINDOWS 关键字的说明。

CURRENT LOAD
显示主机总负载和预留负载。

Reserved
您可以使用 csub -R 指定保留资源（参见*csub* ）。这些资源由主机上运行的作业保留。

Total
总负载的含义取决于负载指标是增加还是减少。

对于增加的负载指标（例如运行队列长度、CPU 利用率、分页活动、登录次数和磁盘 I/O），总负载等于已消耗负载加上预留负载。总负载等于当前负载与预留负载之和。当前负载是*cload* 看到的负载。

对于减少的负载指标（例如可用内存、空闲时间、可用交换空间和 tmp 中的可用空间），总负载等于可用负载。总负载等于当前负载与预留负载之差。此差值是*cload* 看到的可用资源。

LOAD THRESHOLD
显示调度阈值 loadSched 和暂停阈值 loadStop。此外，还会显示迁移阈值（如果已定义）以及检查点支持（如果主机支持）。

阈值的格式与批量作业队列的格式相同（参见*cqueues* 和*cb.yaml* ）。有关阈值和负载指标的说明，请参阅*cqueues* 中 -l 选项下“队列调度参数”关键字的说明。

Job CPU binding is set
显示 CPU 绑定是否已设。

Member of host group
显示主机所属的主机组

Member of queues
显示主机所属的队列

资源选项-s 的输出

-s 选项显示以下内容：用于调度的资源量、预留的资源量以及共享资源的关联主机。仅显示具有数值的共享资源。有关如何配置共享资源，请参阅*cb.yaml* 。

将显示以下字段：

RESOURCE
资源名。

TOTAL
用于调度的共享资源的值。这是共享资源的当前负载和预留负载的总和。

RESERVED
作业预留的资源量。您可以使用 csub -R 指定预留的资源（参见*csub* ）。

LOCATION
与共享资源关联的主机。

5.1.19 cinfo

命令

cinfo - 显示 AIP 资源配置信息

概要

cinfo [**-l**] [**-m** | **-M**] [**-r**] [**-t**] [*资源名…*]

cinfo [**-h** | **-V**]

描述

默认情况下，显示所有资源配置信息，包括资源名称及其含义、主机类型和型号以及系统已知的相关 CPU 因素。

默认情况下，显示所有资源的信息。资源信息包括资源名称、资源类型、描述以及资源的默认排序顺序。

您可以在任务放置请求中使用资源名称。

使用此命令及其选项可以选择性地查看已配置的资源、主机类型和主机型号。

选项

-l
: 以多行长格式显示资源信息。显示的其他参数包括资源是内置的还是已配置的，以及资源值是动态变化的还是静态的。如果资源值动态变化，则间隔表示评估频率。

-m
: 仅显示集群中存在的主机模型的信息。

-M
: 显示所有主机 CPU 型号的信息。

-r
: 仅显示有关已配置资源的信息。

-t
: 仅显示已配置主机类型的信息。类型为 x86_64Linux、x86_64Windows、aarch64Linux 等。

资源名…
: 仅显示有关指定资源的信息。

-h
: 将命令用法打印到 stderr 并退出。

-V
: 将 AIP 发布版本打印到 stderr 并退出。

使用-l 参数的长格式输出

TYPE
: 指示资源是 Number(数值)、Text(字符串)、还是 Tag(标签)。

DIR.

Inc
: 如果负载指标的数值随着其测量的负载增加而增加，例如 CPU 利用率 (ut)。

Dec
: 如果数值随着负载增加而减小。

N/A
: 如果资源不是数字。

INTERVAL
该指标更新间隔的秒数。负载指标每 INTERVAL 秒更新一次。值为 0 表示该值永不更改。

BUILTIN
如果 BUILTIN 为 Yes，则资源名称由 AIP 内部定义。如果 BUILTIN 为 No，则资源名称由管理员在*cb.yaml* 里定义，或者通过*ress* 插入。

DYNAMIC
如果 DYNAMIC 为 Yes，则资源是随时间变化的负载指标。如果 DYNAMIC 为 No，则资源表示固定的信息，例如主机上的总交换空间。资源为静态或布尔值。

RELEASE
仅适用于数字共享资源。指示当使用资源的作业暂停时，AIP 是否释放资源。当使用共享资源的作业暂停时，资源会被作业保留或释放。“Yes”表示资源被保留。“No”表示资源被释放。

SLOT_RES
如果 SLOT_RES 为“Yes”，则该资源与每个作业槽挂钩。每个作业槽会预留一份 csub -R rusage[] 里申请的资源数。这只对数值型资源有效。“No”表示资源与作业槽不挂钩，不管作业有多少作业槽，作业使用的资源数就是 rusage[] 里申请的值。

ASSIGNED
“Yes”表示每份资源对应一个顺序号，调度器会把资源的顺序号分配给作业。如 GPU 卡、端口号等属于这类资源，GPU 卡需要调度到第几个卡上，并把卡号用环境变量 ALLOCATION 放到作业环境中。

5.1.20 cjdep

命令

cjdep - 显示作业依赖关系

概要

cjdep *job_ID* | “*job_ID*[*index*]”

cjdep[**-h** | **-V**]

描述

允许用户显示所有作业依赖关系。它会输出该作业所依赖的其他作业的列表。

选项

job_ID | “***job_ID*[*index*]”**
必填。要操作的作业或作业数组的作业 ID。

对于数组作业，索引、方括号和引号是必需的。

显示此作业所依赖的所有作业。

示例

```
cjdep 135918
Dependencies of job 135918
DEPENDENCY   JOBID   JOB_STATUS    DEPENDENCY_STATUS
done         135917    PEND          false
exit         135816    PEND          false
done         135815    RUN           false
```

另请参阅

csub 、*cjobs*

5.1.21 cjgroup

命令

cjgroup - 显示作业组的信息

概要

cjgroup [**-h** | **-V**]

描述

显示作业组信息。

选项

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到标准错误输出并退出。

输出

显示作业组列表，其中包含以下字段：

GROUP_NAME
: 作业组的名称。

NJOBS
: 作业组中当前的作业槽位数量。

PEND
: 作业组中等待作业的作业槽位数量。

RUN
: 作业组中正在运行的作业的作业槽数量。

SUSP
作业组中已暂停的作业的作业槽数量。

EXITED
作业组中已退出的作业的作业槽数量。

DONE
作业组中已完成的作业的作业槽数量。

JLIMIT
cgadd -L 或 cgmod -L 设置的作业组限制。未配置限制或未使用限制的作业组以短划线 (-) 表示。

示例

```
cjgroup
GROUP    NJOBS   PEND    RUN     SUSP    EXITED DONE    JLIMIT
/u0      1       0       1       0       0       0       3
/ol      1       0       1       0       0       0       -
/bbb     1       0       0       0       1       0       -
/aaa     1       0       0       0       1       0       10
/ol/ol   1       0       1       0       0       0       -
/u0/u0   1       0       1       0       0       0       -
```

另请参阅

cgadd 、*cgdel* 、*cgmod*

5.1.22 cjobs

命令

cjobs ——显示有关 AIP 作业的信息

概要

cjobs [**-a**] [**-A**] [**-w** | **-l** | **-j** | **-o** “格式” | **-UF**] [**-WL** | **-WP** | **-WF**] [**-J** “作业名称”] [**-app** “应用名称”] [**-m** 主机名 | **-m** 主机组] [**-M**] [**-N** 主机名 |**-N** 主机型号 | **-N** *CPU* 因子] [**-P** 项目名称] [**-g** 作业组名称] [**-q** 队列名称] [**-u** 用户名 | **-u** 用户组 | **-u all**] [**-S** 搜索表达式] 作业 *ID*…

cjobs [**-d**] [**-p**] [**-r**] [**-s**] [**-A**] [**-w** | **-l** | **-UF**] [**-J** 作业名称] [**-app** 名称] [**-m** 主机名 | **-m** 主机组] [**-M**] [**-N** 主机名 | **-N** 主机型号 |**-N** *CPU* 因子] [**-P** 项目名称] [**-g** 作业组名称] [**-q** 队列名称] [**-u** 用户名 | **-u** 用户组 | **-u all**] [**-S** 搜索表达式] 作业 *ID*…

cjobs [**-d**] [**-p**] [**-r**] [**-s**] [**-A**] [**-noheader**] [**-J** 作业名称] [**-app** 配置文件名称] [**-m** 主机名 | **-m** 主机组] [**-M**] [**-N** 主机名 | **-N** 主机型号 |**-N** *CPU* 因子] [**-P** 项目名称][**-g** 作业组名称] [**-q** 队列名称] [**-u** 用户名 | **-u** 用户组 | **-u all**] [**-S** 搜索表达式] 作业 *ID*…

cjobs [**-h** | **-V**]

描述

显示有关作业的信息。

默认情况下，显示有关您自己的等待、正在运行和暂停的作业的信息。

命令显示在调度器内存中的作业数据，要显示已经被调度器从内存中清除的作业数据，请使用 chist。

显示作业信息范围

- root 或集群管理员可以访问所有作业信息。
- 当 cb.yaml 中 user_view_alljobs: yes，普通用户可以访问所有作业信息。
- **当 cb.yaml 中 user_view_alljobs: no，或者该参数没有设置：**
  - 用户组管理员可以访问本用户组所有用户的作业信息。
  - 队列管理员可以方问本队列所有用户的作业信息。
  - 普通用户只能访问自己提交的作业。

选项

-a
显示 cb.yaml 中 CLEAN_PERIOD 指定的时间间隔内（默认为 4000 秒）所有状态的作业信息，包括最近完成的作业。

-A
显示有关作业阵列的摘要信息。如果您使用作业阵列 ID 指定作业阵列，并且还指定 -A，则不要包含带有作业阵列 ID 的索引列表。

如果需要，您可以使用 -w 来显示完整的数组规范。

-app 配置文件名称
仅显示与职位配置文件名称匹配的职位。参数 *profile_name* 可以以通配符“*”开头或结尾，例如“*proj123”、“proj*”。

-d
显示在 cb.yaml 中 CLEAN_PERIOD 指定的时间间隔内（默认周期为 1 小时）最近完成的作业的信息。

-p
显示等待作业，以及导致每个作业在上一轮调度中未调度的等待原因。等待原因会显示导致该作业调度的主机数量；如果同时指定了-l 选项，则会显示主机名称。

每个等待原因都与一个或多个主机相关联，并说明这些主机未分配用于运行作业的原因。在作业请求特定主机（使用 csub -m）的情况下，用户可能会看到不相关主机的原因以及与所请求主机相关的原因一同显示。等待原因的生命周期在新的调度轮次开始后结束。该原因可能无法反映当前的负载情况，因为它的持续时间可能与 cb.yaml 中 SCHED_INTERVAL 指定的时间间隔一样长。

当作业阵列达到作业槽限制（csub -J “jobArray[indexList]%job_slot_limit”）时，将显示以下消息：The job array has reached its job slot limit.

-r
显示正在运行的作业。

-s
显示暂停的作业以及导致每个作业暂停的暂停原因。

作业暂停期间，暂停原因可能不会保持不变。例如，某个作业可能由于分页率（pg）而暂停，但分页率下降后，另一个负载指标可能会阻止该作业恢复。暂停原因将根据负载指标进行更新。原因可能与 cb.yaml 中 JM_INTERVAL 指定的时间间隔一样早。因此，显示的原因可能无法反映当前的负载情况。

-W

提供以下资源使用情况信息：PROJ_NAME、CPU_USED、MEM、SWAP、PIDS、START_TIME、FINISH_TIME。仅当您未以管理员身份登录时才显示属于您的作业。

-WF

显示使用 -W 选项（运行限制）提交的正在运行作业的预计完成时间。对于已完成或已退出的作业，显示实际完成时间。

-WL

显示使用 -W 选项（运行限制）提交的作业的估计剩余运行时间。

-WP

显示使用 -W 选项（运行限制）提交的作业的当前估计完成百分比。

-w

宽格式。显示作业信息而不截断字段。

-l

长格式。以多行格式显示每个作业的详细信息。

-l 选项显示以下附加信息：项目名称、作业命令、提交主机上的当前工作目录、等待和暂停原因、作业状态、资源使用情况、资源限制信息。

使用 cjobs -A -l 显示作业阵列的详细信息，包括作业阵列作业限制（%*job_limit*）（如果设置）。

-noheader

指定后，cjobs 将显示字段的值，但不显示字段名称。这在脚本解析时非常有用，因为不需要列标题。

此选项适用于没有选项的 cjobs 命令的输出，以及除 -l、-UF、-N、-h 和 -V 之外的所有带有短形式输出的 cjobs 选项的输出。

-j

JSON 格式。以 JSON 格式显示每个作业的详细信息。使用“jq”等工具将输出转换为可读格式。

-o “字段名称…[delimiter=’字符’]”

设置自定义输出格式。

指定要显示的 cjobs 字段（或字段别名，而非完整字段名）及其显示顺序。每个字段的输出宽度不受限制。

使用 **delimiter=** 来设置不同标题和字段之间的分隔符。分隔符必须是单个字符。默认情况下，分隔符为空格。

要在 csh 环境中指定特殊分隔符（例如 $），请在分隔符规范中使用双引号（“），并在 -o 语句中使用单引号（’）：

```
cjobs...-o 'field_name ... [delimiter="字符"]'
```

-o 选项不能与 cjobs 选项 -l、-UF、-WF、-W、-WL、-WP 一起使用。

如果字段没有值，则输出为破折号 (-)。

以下是用于指定要显示的 cjobs 字段的字段名称、可以代替字段名称使用的别名：

- **allocation|alloc**：资源分配 JSON
- **app**： csub -A 指定的应用程序名称
- **avgmem**：完成作业所用的平均内存（KB）
- **command|cmd**： 作业命令

- **%complete**：使用选项 -W 提交时作业完成的百分比
- **corelimit**：csub -C（ulimit -c）指定的核心文件大小限制
- **cpu_used**：每秒使用的 CPU 时间
- **dependency**： 由 csub -w 指定的依赖关系
- **end_reason|ereason**：作业结束原因
- **error_file**：csub -el-eo 指定的错误文件名
- **estimated_start_time|estart_time**：预计的作业开始时间
- **exec_cwd**：作业正在执行的当前工作目录
- **exec_home**：作业提交的主目录
- **exec_host**：作业执行主机
- **exit_code**： 作业退出码
- **filelimit**：由 csub -F (ulimit -f) 指定的文件大小限制
- **finish_time**：作业完成时间
- **first_host**：第一个作业执行主机
- **from_host**：作业提交主机名
- **gmem**：GPU 内存使用量（MB）
- **idle_factor**：CPU 使用时间/作业运行时间
- **input_file**：csub -il-is 指定的输入文件名
- **io**：作业的磁盘 I/O 范围（以 KB/s 为单位）
- **job_description|description**： csub -Jd 指定的作业描述
- **job_group|group**： csub -g 指定的作业组名称
- **job_name|name**： csub -J 指定的作业名称
- **job_priority|priority**： 由 csub -sp 指定的作业优先级
- **jobid|jd**： 作业 ID
- **last_message|msg|message**：cpost 发布的最后一条消息
- **last_ut|ut**：作业的最后一次 CPU 使用率
- **max_req_proc**：csub -n 请求的最大插槽数
- **maxmem**：完成作业所需的最大内存（KB）
- **mem**：已用内存（KB）
- **memlimit**：csub -M 指定的内存限制
- **min_req_proc|nreq_slot**：csub -n 请求的最小槽数
- **nexec_host**：作业的执行主机数量
- **nthread**：作业的线程数
- **output_file**：csub -ol-oo 指定的输出文件名
- **pend_reason|preason**：作业等待原因
- **pend_time**: 作业等待时间（秒）

- **pids**： 作业的进程 ID
- **pre_exec_command|pre_cmd**： 由 csub -E 指定的 pre_exec
- **processlimit**：csub -p（ulimit -u）指定的最大用户进程数
- **proj_name|proj|project**： csub -P 指定的项目名称
- **queue**： 作业提交到的队列名称
- **req_gpus**：csub -R 请求的 GPU 数量
- **reqmem**：保留内存（以 GB 为单位）
- **res_rusage|ress**：已分配的 GPU
- **resreq**：csub -R 指定的资源需求
- **run_time**：作业运行总时间（秒）
- **slots|nalloc_slot**：已使用的槽总数
- **specified_start_time|sstart_time**：csub -b 指定的作业开始时间
- **specified_terminate_time|sterminate_time**：csub -t 指定的作业终止时间
- **stacklimit**：由 csub -S (ulimit -s) 指定的堆栈大小限制
- **stat**： 作业状态
- **stripped_cmd|scmd**： 不带 #CSUB 选项的作业命令
- **sub_cwd**：运行 csub 或 csub -cwd 时的当前工作目录
- **susp_reason}sreason**：作业暂停原因
- **swap**：操作系统为该作业分配的虚拟内存（以 KB 为单位）
- **swaplimit**：由 csub -v 指定的虚拟内存限制（以 KB 为单位）
- **time_left**：使用选项 -W 提交时作业剩余运行时间
- **user**： 提交用户
- **user_group|ugroup**： 作业用户所属的用户组

字段名称和别名不区分大小写。

例如:

```
cjobs -o "id stat name start_time delimiter='^'"
JOBID^STAT^JOB_NAME^START_TIME
2730^RUN^my_testjob^Sep 8 11:59
```

-S 搜索表达式

按照搜索表达式定义的条件列出作业。搜索表达式使用变量（见下方的变量列表），组合运算符形成搜索条件。

例子 1:

```
cjobs -S "submit_time >= '2025/01/03' && submit_time <= '2025/06/10'"
```

列出提交日期为 2025 年 1 月 3 日到 2025 年 6 月 10 日本用户的活动作业。如果要包含已完成的作业，可以使用-a 选项。

例子 2:

```
cjobs -S "mem / reqmem > 1.2" -a -u all
```

列出内存使用超过预留内存 1.2 倍的作业，包括活动作业和已完成作业，并包含所有用户的作业。

例子 3:

```
cjobs -S "job_name == 'myjob*'"
```

列出我的活动作业中，作业名前缀为 myjob 的作业。

例子 4:

```
cjobs -S "user > u001" -u all
```

列出所有用户作业中作业用户名的字串排序“大于”u001 的作业。字串排序是根据字母循序排的。

表达式变量

- **app:** “csub -A”定义的应用名
- **command:** 作业命令
- **cpu_used:** CPU 用量，单位秒
- **description:** “csub -Jd”定义的作业描述
- **docker:** “docker > 0”表示 Docker 作业。“docker == 0”表示非 Docker 作业
- **exclusive:** “exclusive > 0”表示独占作业。“exclusive == 0”表示非独占作业
- **exec_cwd:** 作业运行时的工作目录
- **exit_code:** 作业退出码。如果被 Linux kill 消息（SIGNAL）所杀，退出码为 SIGNAL + 128
- **finish_time:** 作业结束时间。见下方的时间格式说明，如“2025/05/02 10:33”
- **first_host:** 作业运行主机名。若为多机的并行作业，这个变量只表示第一台主机
- **from_host:** 作业提交主机名
- **idle_factor:** 作业空闲指数，计算方法是：CPU 用时 / （作业运行时长 * 作业槽数）
- **interact:** “interact > 0”表示交互式作业（csub -I），“interact == 0”表示非交互式作业
- **job_group:** “csub -g”定义的作业组名
- **job_name:** “csub -J”定义的作业名
- **job_priority:** “csub -sp”定义的作业优先级
- **jobid:** 作业号
- **jobindex:** 阵列作业的序号
- **last_ut:** 过去 15 秒的作业 CPU 利用率，值为 0.0-1.0 表示 0 - 100%
- **mem:** 作业内存用量，单位为 GB
- **memlimit:** 作业内存限制，单位为 GB
- **npids:** 作业运行的进程数，一般为 3 个或以上
- **pend_time:** 作业等待时间，单位为分钟
- **project:** “csub -P”定义的项目名
- **queue:** “csub -q”定义的队列名，或者缺省队列名

- **req_gpus:** “csub -R rusage[gpu=X]”请求的 GPU 数，可以是分数
- **reqhost:** “csub -m”定义的第一个主机名
- **reqmem:** “csub -R rusage[mem=X]”请求的内存，单位为 GB
- **run_time:** 作业运行时长，单位为分钟
- **runlimit:** “csub -W”定义的时限，单位为分钟
- **slots:** 作业请求或使用的作业槽数
- **start_time:** 作业启动时间，格式见下方的说明
- **sub_cwd:** “csub -cwd”定义的作业工作目录，或者作业提交时的当前工作目录
- **submit_time:** 作业提交时间，格式见下方的说明
- **user:** 作业提交用户
- **user_group:** 作业用户所在的作业组名，或者“csub -G”参数指定的用户组名

时间格式

时间格式为 yyyy/mm/dd 间隔符 HH:MM:SS。间隔符可以是空格、-、或/。时分秒可选。

例子：

- ‘2025/03/03’
- ‘2025/03/04 10’
- ‘2025/04/02 10:30’
- ‘2025/04/02 10:30:12’

备注: 当日期后不带时间时，缺省的时间为 00:00, 小时后不带分钟，默认分钟为 00，分钟后不带秒，默认秒为 00。

运算符

- 数值运算 +，-，*，/ 加、减、乘、除
- 逻辑运算 &&，|| 与、或
- 判断运算 ==，!=，<，>，<=，>= 等于、不等于、小于、大于、小于或等于、大于或等于
- 计算顺序可以用 () 表示括弧中的表达式先计算结果

字符串前后缀

通配符“*”可用于定义字符串的前后缀。如“myjob*”表示前缀为 myjob 的字串，“*test”表示后缀为 test 的字串。

警告: 时间参数 submit_time、start_time、和 finish_time 由于是字符串，不能做加减乘除的运算。

-UF

无格式。显示无格式的作业详细信息。这使得编写用于解析 cjobs 关键字的脚本变得更容易。此选项的结果对输出没有宽度控制。每行从行首开始。SCHEDULING PARAMETERS 和 PENDING REASONS 的信息保持格式化。没有任何分隔符结尾的使用信息行会添加分号来分隔其各个部分。首行和所有以时间戳开头的行均以无格式显示在一行中。没有行长和格式控制。

***job_ID* …**
显示有关指定作业或作业阵列的信息。

如果使用 -A，请指定不带索引列表的作业阵列 ID。

-g 作业组名称
显示有关附加到由 job_group_name 指定的作业组的作业的信息。

-J 作业名称
显示有关指定作业名称或作业阵列的信息。作业名称可以在尾部用通配符”*”，例子:

```
cjobs -J proj*              #␣
↪显示本用户所有以proj开头的作业名称的活动作业（等待、运行、和暂停）
```

-m 主机名 | -m 主机组
仅显示分派到指定主机的作业。

要确定可用的主机和主机组，请使用 chosts 和 cmgroup。

-M
与 -l 选项结合使用时显示作业的最后一条发布消息。

-N 主机名 | -N 主机型号 | -N *CPU* 系数
显示作业消耗的标准化 CPU 时间。使用指定的 CPU 因子，或者指定主机或主机型号的 CPU 因子进行标准化。

-P 项目名称
仅显示属于指定项目的作业。

-q 队列名称
仅显示指定队列中的作业。命令 cqueues 返回系统中配置的队列列表，以及有关这些队列的配置信息。

-u 用户名 | -u 用户组 | -u all
仅显示由指定用户提交的作业。关键字 all 指定所有用户。该选项受集群用户权限限制，缺省权限配置时普通用户只能显示自己的作业信息。用户组管理员可以显示组内用户的作业信息，队列管理员可以显示队列中用户的作业信息。AIP 管理员不受限制。

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 发布版本打印到 stderr 并退出。

输出

等待作业将按其被考虑调度的顺序显示。高优先级队列中的作业优先于低优先级队列中的作业显示。相同优先级队列中的等待作业将按其提交的顺序显示，但可以使用 ctop 或 cbot 命令更改此顺序。如果将多个作业调度到一台主机，则该主机上的作业将按其队列优先级和调度时间的顺序列出。已完成的作业将按其完成的顺序显示。

默认显示

将显示包含以下字段的作业列表：

JOBID
AIP 分配给该作业的作业 ID。

USER
提交作业的用户。

STAT
作业的当前状态（请参阅下面的作业状态）。

QUEUE
作业所属的作业队列的名称。如果作业所属的队列已从配置中移除，队列名称将显示为 lost_and_found。使用 chist 获取原始队列名称。lost_and_found 队列中的作业将保持等待状态，直到使用 cswitch 命令将其切换到另一个队列。

FROM_HOST
提交作业的主机的名称。

EXEC_HOST
正在执行作业的一个或多个主机的名称（如果作业尚未调度，则此字段为空）。如果运行作业的主机已从配置中移除，则主机名将显示为 lost_and_found。使用 chist 获取原始主机名。

JOB_NAME
用户指定的作业名称，或默认指定的命令字符串（参见 csub(1)）。如果作业名称太长，此字段容纳不下，则仅显示作业名称的后半部分。

SUBMIT_TIME
作业的提交时间。

-l 输出

如果指定了 -l 选项，则生成的长格式列表将包括以下附加字段：

Job
作业 ID

User
提交作业用户名

Project
提交作业的项目。

Profile
csub -A 或 csub -app 定义的应用名。

Job Description
csub -Jd 定义的作业描述。

Pre-execute Command
csub -E 定义的 pre-exec 命令。

Command
作业命令。

CWD
提交主机上的当前工作目录。

RUNTIME
cjobs -WL, -WF, 或者-P 输出项

TIME_LEFT
估计的作业剩余运行时长

FINISH_TIME
估计的作业结束时间

%COMPLETE
估计的作业完成百分比

PENDING REASONS
作业处于 PEND 或 PSUSP 状态的原因。同时指定 -p 和 -l 选项时，将显示与每个原因关联的主机名称。

SUSPENDING REASONS
作业处于 USUSP 或 SSUSP 状态的原因。

loadSched
作业的负载调度阈值。

loadStop
作业的负载暂停阈值。

JOB STATUS
作业状态的可能值包括：

PEND
作业处于等待状态，即尚未开始。

PSUSP
该作业在等待期间已被其所有者或 AIP 管理员暂停。

RUN
该作业目前正在运行。

USUSP
该作业在运行时已被其所有者或 AIP 管理员暂停。

SSUSP
该作业已被 AIP 暂停。该作业因以下两个原因之一而被 AIP 暂停：

1. 根据为主机或队列定义的 loadStop 向量，执行主机上的负载情况已超过阈值。
2. 作业队列的运行窗口已关闭。参见*cqueues*、*chosts* 和*cb.yaml*。

DONE
作业已结束，状态为 0。

EXIT
作业已以非零退出码终止 - 它可能由于执行中的错误而被中止，或者被其所有者或 AIP 管理员终止。

UNKWN
CBSCHED 与运行该作业的主机上的 CBJM 失去联系，或者作业超时而无法被杀死。

ZOMBI
如果出现以下情况，作业将变为 ZOMBI：

- 当执行主机上的 CBJM 无法访问且作业显示为 UNKWN 时，不可重新运行的作业被 ckill 终止。
- 正在运行可重新运行作业的主机不可用，并且该作业已由 AIP 使用新的作业 ID 重新排队，就好像该作业作为新作业提交一样。

当执行主机可用后，AIP 将尝试终止 ZOMBI 作业。成功终止 ZOMBI 作业后，该作业的状态将更改为 EXIT。

RESOURCE USAGE

当前作业使用情况的值包括：

CPU time

作业中所有进程的累计 CPU 时间（以秒为单位）。

MEM

作业中所有进程的驻留内存使用量总计（以 MB 为单位）。

SWAP

作业中所有进程的虚拟内存总使用量（以 MB 为单位），包含所有操作系统为作业分配的虚拟内存总量。

DISKIO

当前作业中所有进程的磁盘 IO，以 KB/s 为单位。

GPUMem

当前作业中所有进程的 GPU 内存使用情况（以 MB 为单位）。

NTHREADS

当前作业中所有进程的线程数。

PGID

作业中当前活动的进程组 ID。

PIDs

作业中当前活动的进程。

RESOURCE LIMITS

对队列中的作业施加的硬资源限制（请参阅 getrlimit(2) 和*cb.yaml*）。这些限制是针对每个作业和每个进程施加的。

每个作业可能的限制是：

CPULIMIT

PROCLIMIT

MEMLIMIT

SWAPLIMIT

PROCESSLIMIT

Linux 每个进程可能的资源限制是：

RUNLIMIT

FILELIMIT

DATALIMIT

STAKLIMIT

CORELIMIT

如果提交到队列的作业指定了这些限制中的任何一个（请参阅*csub*），则将对该作业使用相应作业限制和队列限制中较低的一个。

如果没有指定资源限制，则假定资源是无限的。

SCHEDULING PARAMETERS

调度负载阈值，合并主机级和队列级配置的调度负载阈值。

loadSched
作业的负载调度阈值。

loadStop
作业的负载暂停阈值。

RESOURCE REQUIREMENT DETAILS
作业调度的资源需求。调度器用这些参数决定作业运行主机。

Combined：合并作业定义的资源需求 (csub -R)、队列中配置的资源需求 (resspec)、和系统内置的缺省调度资源需求。

Effective：如果合并的作业需求有多个“或”的选择，最终作业调度时的有效资源需求。

RESOURCE ALLOCATION
作业调度后的资源分配 JSON，每个作业槽的分配主机名、端口名、GPU、以及共享资源的分配。

PENDING PART
作业如果有等待部分，全部等待，或者可伸缩作业部分等待的 JSON。

THE MOST RECENT MESSAGE
作业的最后一个 cpost 的消息。

Time
消息发布的时间

Content
消息内容

作业阵列摘要信息

如果使用 -A，则显示有关作业阵列的摘要信息。将显示以下字段：

JOBID
作业阵列的作业 ID。

ARRAY_SPEC
数组规范的格式为 *名称* [*索引*]。数组规范可能被截断，请使用 -w 选项和 -A 选项一起显示完整的数组规范。

OWNER
作业阵列的所有者。

NJOBS
作业阵列中的作业数。

PEND
作业阵列中等待的作业数。

RUN
作业阵列中正在运行的作业数。

DONE
作业阵列中成功完成的作业数。

EXIT
作业阵列中未成功完成的作业数。

SSUSP
作业阵列的 AIP 系统暂停作业的数量。

USUSP
作业阵列中用户暂停的作业数。

PSUSP
作业阵列中等待时被暂停的作业数。

作业等待原因

当作业等待时，使用-p 列出作业等待原因。

用-l 或-lp 参数，则观察 **PENDING REASONS** 的结果。

作业参数和调度相关的原因

New job is waiting for scheduling
新的作业还未来得及调度

The job has a specified start time
作业定义的开始运行 (csub -b) 时间还没到

Job dependency condition not satisfied
作业依赖关系（csub -w）不满足

Dependency condition invalid or never satisfied
非法作业依赖关系（csub -w），作业永远不会被调度

The job’s pre-exec command exited with non-zero status
作业的前处理（csub -E）程序执行的退出码为非 0

Job execution initialization failed
CBJM 无法初始化作业，一般是由于作业运行主机的文件系统出现问题，或者主机没有可用资源（内存或本地存储）了

The schedule of the job is postponed for a while
调度器过忙，需等几分钟

Waiting for re-scheduling after switching queue
作业切换队列后重新调度 (cswitch)

队列相关的原因

The queue is inactivated by the administrator
队列已被系统管理员停用，不再调度作业

The queue is inactivated by its time windows
队列配置了运行时间窗，当前不在时间窗的时间内

The queue has reached its job slot limit
队列最大作业槽限制（maxslots）已经达到

User has reached the per-user job slot limit of the queue
队列所设每个用户的最大作业槽限制（usermaxslots）已经达到

The queue’s pre-exec command exited with non-zero status
队列的前处理（pre_exec）程序执行的退出码为非 0

Requeued job is waiting for rescheduling
作业被终止然后重调度：cswitch、运行主机挂了、或者作业退出码符合队列配置的重调度退出码（rerunonexitcode）

Job will not finish before queue’s run window is closed
作业定义了结束时间或者运行时限，队列定义了运行时间窗。作业在队列运行时间窗关闭前无法结束

Not allowed due to non-host related resource limit
作业受限于 limits 里定义的（与主机无关的）资源限制

Can’t run jobs submitted by root
集群配置了不允许 root 运行作业

用户或用户组配置相关的原因

The user has reached job slot limit
受限于用户作业槽限制

One of the user’s groups has reached its job slot limit
受限于用户组作业槽限制

Waiting for scheduling after resumed by user
用户恢复了作业，等待被调度

Unable to determine user account for execution
作业运行主机上无法认证用户账号

The user has no permission to run the job on remote host/cluster
用户在远程主机上没有作业运行权限

Host exausted mapped user accounts
配置了有限的账号映射，没有可用的账号可以映射了

主机相关的原因

这类原因可以用-lp 选项列出有相应原因的主机名。

小技巧： 作业等待只列出用户请求和队列配置的主机名，不会列出集群所有主机名。

Job’s resource requirements not satisfied
主机不满足作业的资源需求

Job’s requirement for exclusive execution not satisfied
主机不满足独占作业需求

Higher or equal priority jobs already suspended by system
作业抢占低优先级作业槽，已经没有可被抢占的作业槽

Cleaning up zombie job
主机正在清理僵尸作业

Closed by AIP administrator
主机被管理员关闭

Host is locked
主机被管理员锁住了（cadmin lslock)

Not enough job slot(s)
主机上作业槽不够

Job slot limit reached
主机作业槽上限已达到

Not usable to the queue
主机不能被队列所用，如作业提交指定的主机名不在队列配置的主机中

Just started a job recently
cb.yaml 中 job_accept_interval 不是 0，而主机刚接受了作业

Load information unavailable
CBJM 无法获取该主机的负载信息，原因可能是集群中的通讯短暂中断，或者 CBLS 需要重启

CBLS is unreachable now
主机的 CBLS 不正常

Queue’s resource requirements not satisfied
队列中配置的资源需求不满足

Not the same type as the submission host
除非特别指定，AIP 只把作业调度到与作业提交主机相同类型的主机上，这个主机的类型与作业提交主机不同

Not enough processors to meet the job’s spanning requirement
主机不满足 span 的资源需求

Not enough processors to meet the queue’s spanning requirement
主机不满足队列 span 的资源需求

Running an exclusive job
主机已被独占作业所占

Queue’s requirements for resource reservation not satisfied
主机不满足队列资源需求中 rusage 定义的条件

Job’s requirements for resource reservation not satisfied
主机不满足作业资源需求中 rusage 定义的条件

Power saving mode
主机处于节电关闭模式

Owned by other users with pending jobs
主机属于 owership 的队列，该主机的拥有者有等待作业

Unable to reach slave job server
主机 CBJM 工作不正常

Failed in talking to server to start the job
调度器无法把作业分发到该主机

Failed in receiving the reply from server when starting the job
分发作业后没有得到作业运行主机的回答

Unable to allocate memory to run job
运行主机没有可用内存了

Unable to fork process to run job
运行主机没有可用进程了

Load threshold reached
主机负载超过配置的阈值（主机阈值或者队列配置的阈值）

Preempted job is waiting to be resumed
被抢占的作业正在等待恢复

The job has been requeued
作业被重排队

The job group limit has been reached
作业组作业槽限额已到达

System is unable to set pending reason for job
未知原因

作业暂停原因

使用**-l** 参数可以显示暂停作业的暂停原因 **SUSPENDING REASONS**。

The job was suspended by user
作业由用户使用 cstop 命令暂停

The job was suspended by AIP administrator or root
作业由 AIP 管理员或 root 用户用 cstop 命令暂停

The run windows of the queue are closed
队列的作业运行时间窗关闭

The execution host is locked by AIP administrator now
作业运行主机被管理员锁住了

Waiting for re-scheduling after being resumed by user
用户运行 cresume 命令后，作业等待被调度

Host load exceeded threshold：负载指标
主机负载已到暂停阈值，并列住超载的负载名

Preempted by the scheduler
作业被抢占而暂停

作业结束原因

用-l 参数查看作业状态时间序列的最后一个序列，如果调度知道作业退出码为非 0 的原因（如由用户、管理员通过调度器对作业做的操作，或者是调度规则导致），会列出作业退出原因。作业退出原因也可使用 cjobs -o “end_reason” 查看。例子:

```
cjobs -o end_reason 1054
END_REASON
Signaled by the job owner
```

如果调度器无法判断作业结束原因，则不显示结果。

Requeued
作业被 crequeue 命令重排队

Rerun on host failure
作业由于运行主机出错而重调度

Rerun on configured exit code
作业退出码与配置的重运行退出码一致（队列参数 rerunonexitcode)

Requeued on checkpoint
检查点退出

Signaled by the
被以下个体所杀：

job owner：作业用户

system administrator：系统管理员

queue administrator：队列管理员

group administrator：用户组管理员

scheduling policy：调度策略

RUNLIMIT was reached
作业到达运行时限被杀

DEADLINE was reached
作业到达结束时间被杀

PROCESSLIMIT was reached
作业进程超限被杀

CPULIMIT was reached
CPU 用时超限被杀

MEMLIMIT was reached
内存使用超过限制被杀

Used unallocated GPU
使用了不是分配给本作业的 GPU 被杀

作业退出码

作业的退出码与作业命令，即作业主进程的退出码。**退出码是作业命令自身的，与调度器无关**。

常规的退出码定义为：

- 0：正常结束，AIP 把退出码为 0 的作业定义为 DONE，其他为 EXIT。
- 126：作业命令文件没有执行权限
- 127：作业命令找不到
- 129 - 192：AIP 根据 Linux 的规范认为时作业进程被信号（kill 命令）所杀，所杀的信号为：退出码 - 128。如: 退出码 130 是被信号 2，即 SIGINT(Ctrl-C) 所杀，等等。具体 Linux 的可用信号参见 kill -l 的输出。

示例

```
cjobs -pl
```

显示有关调用者的所有等待作业的详细信息。

```
cjobs -ps
```

仅显示等待和暂停的作业。

```
cjobs -u all -a
```

显示所有用户的所有作业。

```
cjobs -d -q short -m apple -u john
```

显示 john 提交到短队列并在主机 apple 上执行的所有最近完成的作业。

```
cjobs 101 102 203 509
```

显示 job_ID 为 101、102、203 和 509 的作业。

```
cjobs -o "id stat"
```

显示作业所有本用户的活动作业（等待、运行、或暂停）的 JOBID 和作业状态

5.1.23 ckill

命令

ckill - 发送信号以终止、暂停或恢复未完成的作业

概要

ckill[**-l**] [**-J** 作业名称] [**-m** 主机名 |**-m** 主机组] [**-q** 队列名称] [**-g** 作业组名称] [**-r** | **-s** (信号值 | 信号名称)] [**-u** 用户名 |**-u** 用户组 | **-u all**] [**-b** "批次大小 等待秒"] [作业 *ID*… | **0** | "*job_ID*[*index*]" …]

ckill[**-h** | **-V**]

说明

默认情况下，发送一组信号来终止指定的作业。在 Linux 上，会发送 SIGINT 和 SIGTERM 信号，让作业有机会在终止前进行清理，然后发送 SIGKILL 信号来终止作业。发送每个信号的时间间隔由*cb.yaml* 中的 job_terminate_interval 参数定义。

用户只能操作自己的作业。只有 root 和 AIP 管理员可以操作其他用户提交的作业。队列管理员可以操作所管队列中其他用户提交的作业。用户组管理员可以操作所管用户组中其他用户提交的作业。

如果信号请求未能到达作业执行主机，AIP 会在主机恢复连接后尝试执行该操作。AIP 会重试最近的信号请求。

如果无法终止作业，请使用 ckill -r 从 AIP 系统中移除该作业，而无需等待其终止，并释放作业的资源。

ckill 必须指定作业 ID 或 -m、-u、-q、-g 或 -J。

选项

-b "批次大小 [间隔等待秒数]"
当终止大量作业（例如，jobid 为 0）时，请指定批次大小，并在每批终止作业之间等待一秒。这是为了减少调度程序的开销。

默认情况下，等待功能处于禁用状态，ckill 命令会继续与调度程序交互以终止作业。

-l
显示 ckill 支持的信号名称。这是 /bin/kill 支持的信号子集，并且与平台相关。

-g *job_group_name*
仅对 *job_group_name* 指定的作业组中的作业执行操作。

ckill 不会终止路径中较低级别作业组中的作业。例如，作业附加到作业组 /mygroup 和 /mygroup/subgroup:

```
csub -g /mygroup myjob
Job 1021 has been submitted to the default queue [medium].
csub -g /mygroup/subgroup myjob2
Job 1022 has been submitted to the default queue [medium].
```

以下 ckill 命令仅终止 /mygroup 中的作业，而不会终止子组 /mygroup/subgroup:

```
ckill -g /mygroup 0
Job <1021> is being terminated
ckill -g /mygroup/subgroup 0
Job <1022> is being terminated
```

-J *job_name*
仅对具有指定 *job_name* 的作业执行操作。如果在 *job_ID* 选项中指定的作业 ID 不是 0，则 -J 选项将被忽略。

-m 主机名 |-m** 主机组名**
仅对调度到指定主机或主机组的作业执行操作。

如果未指定 *job_ID*，则仅对最近提交的符合条件的作业执行操作。如果在 *job_ID* 选项中指定的作业 ID 不是 0，则 -m 选项将被忽略。有关主机和主机组的更多信息，请参阅*chosts* 和*cmgroup* 。

-q 队列名
仅对指定队列中的作业执行操作。

如果未指定 *job_ID*，则仅对最近提交的符合条件的作业执行操作。

如果在 *job_ID* 选项中指定的作业 ID 不是 0，则 -q 选项将被忽略。

有关队列的更多信息，请参阅*cqueues* 。

-r
从 AIP 系统中删除作业，而不等待该作业在操作系统中终止。

发送与不带 -r 选项的 ckill 相同的一系列信号，不同之处在于，作业会立即从系统中移除，作业会被标记为 EXIT，并且 AIP 监控的作业资源会在 AIP 收到第一个信号后立即释放。

也适用于已发出 ckill 命令但 CBJM 无法对其进行操作的作业（处于 ZOMBI 状态的作业）。如果 CBJM 在作业完全移除之前恢复，AIP 会忽略使用 ckill -r 终止的僵尸作业。

仅对无法在操作系统中终止的作业或无法使用 **ckill** 以其他方式移除的作业使用 **ckill -r**。

-r 选项不能与 -s 选项一起使用。

-s (*signal_value* | *signal_name*)
将指定的信号发送到指定的作业。您可以指定一个名称（去除 SIG 前缀，例如 KILL），或一个数字（例如 9）。

符合条件信号名称由 **ckill -l** 列出。

-s 选项不能与 **-r** 选项一起使用。

使用 ckill -s 命令，通过适当的信号（而不是 cstop 或 cresume）来暂停和恢复作业。发送 SIGCONT 信号与使用 cresume 相同。向顺序作业发送 SIGSTOP 信号或向并行作业发送 SIGTSTP 信号与使用 cstop 相同。

您无法暂停已暂停的作业，也无法恢复未暂停的作业。对处于 USUSP 状态的作业使用 SIGSTOP 或 SIGTSTP 信号无效，对未处于 PSUSP 或 USUSP 状态的作业使用 SIGCONT 信号也无效。有关作业状态的更多信息，请参阅*cjobs* 。

-u 用户名 |-u 用户组名 | -u all
仅对指定用户或用户组提交的作业执行操作（请参阅*cugroup* ），如果指定了保留用户名 all，则仅对所有用户提交的作业执行操作。

如果未指定 *job_ID*，则仅对最近提交的符合条件的作业执行操作。如果 *job_ID* 选项中指定的作业 ID 不是 0，则 -u 选项将被忽略。

备注： 用户必须有相应的权限才能用此参数。

***job_ID*…| 0 | "*job_ID*[*index*]" …**
仅对由 *job_ID* 或 "*job_ID*[*index*]" 指定的作业进行操作，其中" *job_ID*[*index*]" 指定选定的作业数组元素（参见*cjobs* ）。对于作业数组，必须用引号将作业 ID 和索引括起来，并且索引必须用方括号括起来。

任何用户提交的作业都可以在此处指定，而无需使用 -u 选项。如果使用保留的作业 ID 0，则所有满足其他选项（即 -m、-q、-u 和 -J）的作业都会被操作；所有其他作业 ID 都将被忽略。

如果指定的作业 ID 不是 0，则选项 -u、-q、-m 和 -J 无效。作业 ID 在作业提交时返回（参见*csub* ），也可以使用 cjobs 命令获取（参见*cjobs* ）。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

示例

% **ckill -s 17 -q night**

向调用者提交到队列 night 的最后一个作业发送信号 17。

% **ckill -q short -u all 0**

终止队列 short 中的所有作业。

% **ckill -r 1045**

强制移除无法终止的作业 1045。

5.1.24 cload

命令

cload - 显示主机的负载信息

概要

cload [**-l**] [**-w**] [**-N** | **-E**] [**-o** “格式”] [**-I***load_index*[**:***load_index*] …] [**-n** 主机数] [**-S**] [**-R** 资源需求] 主机名……

cload -s [资源名…]

cload [**-h** | **-V**]

描述

显示主机的负载信息。负载信息可以按主机或按资源显示。

默认情况下，显示本地集群中所有主机的负载信息（按主机排序）。

默认情况下，显示原始负载指标。

默认情况下，资源的负载信息根据 CPU 和分页负载显示。

选项

-l
长格式。显示不带截断的负载信息，以及 I/O 和外部负载指标的附加字段。

此选项将覆盖使用 -I 选项指定的指标名称。

-w
以宽格式显示负载信息。字段显示不带截断。

-N
显示规范化的 CPU 运行队列长度负载指标。规范化的 CPU 运行队列长度负载是把从操作系统获得的指标除以 CPU 因子。

-E
显示有效的 CPU 运行队列长度负载指标。选项 -N 和-E 互斥。有效的 CPU 运行队列长度负载指标是把从操作系统获得的指标除以 CPU 核数（物理核或者逻辑核，取决于*cb.yaml* 里定义的 define_ncpus 参数）。

-I ***load_index*[:*load_index*] …**
仅显示指定负载索引的负载信息。负载索引名称必须用冒号分隔（例如，r1m:pg:ut）。

-n 主机数
仅显示请求数量的主机的负载信息。最多显示 主机数个最符合资源要求的主机的信息。

-o “字段…[delimiter=’分隔符’]”
通过字段名称指定自定义输出格式。使用 **delimiter=** 来设置显示在不同标题和字段之间的分隔符。分隔符必须是单个字符。默认情况下，分隔符为空格。

如果字段没有值，则输出为短划线 (-)。

可用的字段名称包括：

host_name：主机名

status：状态。参见输出

r15s：”r15s”的负载。参见输出

r1m：”r1m”的负载。参见输出

r15m：”r15m”的负载。参见输出

ut：”ut”的负载。参见输出

io：”io”的负载。参见输出

tmp：”tmp”的负载。参见输出

mem：”mem”的负载。参见输出

swp：“swp”的负载。参见输出

pg：“pg”的负载。参见输出

up：“up”的负载。参见输出

-R 资源需求
仅显示满足指定资源要求的主机的负载信息。有关内置资源名称列表，请参阅*cinfo* 。

主机的负载信息根据指定资源的负载排序。

如果 *资源需求*包含特殊资源名称，则仅显示提供这些资源的主机的负载信息（请参阅*chinfo* 以了解每个主机上可用的资源）。

如果指定了一个或多个主机名，则仅显示满足资源要求的主机的负载信息。

-S
显示“正常”主机数量和“不可用”主机数量的摘要。

主机名…
仅显示指定主机的负载信息。

主机名也可以采用 xxxx[001-100] 的格式指定。

-s [资源名…]
显示集群中配置的所有动态共享资源的信息。

如果指定了资源，则仅显示指定资源的信息。*资源名*必须是动态共享资源名称。

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 发行版本打印到 stderr 并退出。

输出

基于主机的输出（默认输出）

显示数值型动态非共享资源。*res_req* 的选择和排序部分控制显示哪些主机信息以及如何排序。

显示的默认负载信息包含以下字段：

HOST_NAME
AIP 使用的标准主机名。

Status
主机状态。状态前面可能出现减号 (-)，表示主机上的 AIP 远程执行服务器 (CBEXE) 未运行。

可能的状态包括：

ok
主机处于正常状态，可以接受远程作业。

lockU
主机已被 AIP 管理员或 root 锁定。

unavail
主机已关闭电源或 AIP 的 CBLS 未正常运行。

负载指标值
内置负载指标包括 r15s、r1m、r15m、ut、pg、io、up、it、swp、mem 和 tmp（见下文）。外部负载指标由 RESS 配置（参见*ress* ）。

r15s
15 秒指数平均 CPU 运行队列长度。

r1m
1 分钟指数平均 CPU 运行队列长度。

r15m
15 分钟指数平均 CPU 运行队列长度。

ut
过去一分钟内 CPU 利用率的指数平均值，单位是百分数（%），介于 0 到 100 之间。

pg
过去一分钟内内存分页速率的指数平均值，以每秒页数为单位。

io
过去一分钟内网络 I/O 速率的指数平均值，以每秒 KB 为单位（仅当指定 -l 选项时可用）。

up
主机正常运行时间 uptime（分钟）。

it
在 Linux 系统中，主机的空闲时间（所有登录会话中未触摸键盘的时间），以分钟为单位。

swp
可用的交换空间大小。

mem
可用内存大小。

tmp
/tmp 中的可用空间大小。

external_index
任何站点配置的全局外部负载索引（请参阅*ress* ）。仅当使用 -l 选项或带有索引名称的 -I 选项时可用。

备注： *external_index* 不应包含共享资源。

基于资源的输出 (cload -s)

显示有关动态共享资源的信息。每行提供资源实例的值及其关联的主机。有关配置动态共享资源的信息，请参阅 ress(8) 和 cb.yaml(5)。

显示的信息包含以下字段：

RESOURCE
资源名称。

VALUE
资源实例的值。

LOCATION
与资源实例关联的主机。

示例

% **cload -R "select[r1m<=0.5 && swp>=20 && type==ALPHA]"**

或者，使用受限格式：

%**cload -R r1m=0.5:swp=20:type=ALPHA**

显示交换空间至少为 20 MB，且 1 分钟运行队列长度小于 0.5 MB 的 ALPHA 主机的负载。

% **cload -R "select[(1-swp/maxswp)<0.75] order[pg]"**

显示交换空间利用率低于 75% 的主机的负载。结果主机按分页率排序。

%**cload -I r1m:ut:io:pg**

显示集群中所有主机的 1 分钟 CPU 原始运行队列长度、CPU 利用率、磁盘 I/O 和分页率。

%**cload -E**

显示所有主机的负载，按 r15s:pg, 排序，其中 CPU 运行队列长度为有效运行队列长度。

%**cload -s verilog_license**

显示所有 verilog_license 动态共享资源实例的值和位置。

诊断

如果检测到 AIP 问题或指定了无效的资源名称，则退出状态为 -10。

如果指定了无效参数，则退出状态为 -1，否则 cload 返回 0。

5.1.25 cmgroup

命令

cmgroup - 显示主机组信息

概要

cmgroup [**-r**] [**-s**] [**-o** “格式”] [主机组名…]

cmgroup[**-h** | **-V**]

描述

显示主机组及其主机名。

默认情况下，显示所有主机组的信息。

选项

-s
以竖排格式显示。

-r
递归扩展主机组。扩展后的列表仅包含主机名。它不包含子组的名称。重复的名称仅列出一次。

-o “字段名…[delimiter=’* 分隔符 *’] [-**a]”**
通过字段名称指定自定义输出格式。使用 **delimiter=** 来设置显示在不同标题和字段之间的分隔符。分隔符必须是单个字符。默认情况下，分隔符为空格。

如果字段没有值，则输出为短划线 (-)。

使用选项“-a”，输出还会包含一行，表示组名为“_allhosts”的整个集群。

可用的字段名称包括：

group_name|name|hgroup：主机组名称

members：组成员主机

maxavail_slots：组中最大可用作业槽位数量

used_slots：作业占用的作业槽位数量

maxavail_memgb：组中所有主机配置的最大内存（以 GB 为单位）

used_memgb：组中所有主机已用内存（以 GB 为单位）

weightd_ut|ut：组中所有可用主机的加权 CPU 利用率

***host_group*…**
仅显示指定主机组的信息. 指定多个主机组时请勿使用引号。

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 发行版本打印到 stderr 并退出。

输出

在主机列表中，名称后跟斜杠 (/) 表示子组。输出字段为：

GROUP_NAME
主机组名

HOSTS
主机组成员主机名。多个主机名也可能用 node[001-100] 的各是输出。名称后跟斜杠 (/) 表示子组。

文件

主机组在配置文件*cb.yaml* 中定义，配置文件也可能是 hg.yaml。

5.1.26 cmod

命令

cmod - 修改作业的提交选项

概要

cmod[*csub options*] [*job_ID*| "*job_ID*[*index*]"]

cmod [**-h** | **-V**]

选项列表

[**-B** | **-Bn**]
[**-N** | **-Nn**]
[**-r**| **-rn**]
[**-x** | **-xn**]
[**-b** 开始时间 | **-bn**]
[**-C** *core* 限制 | **-Cn**]
[**-c** [小时**:**] 分钟 [**/**主机名 | **/**主机型号] | **-cn**]
[**-D** 数据限制 | **-Dn**]
[**-e** 错误文件 | **-en**]
[**-E** "预执行命令 [参数…] " | **-En**]
[**-f** "本地文件操作 [远程文件] "…| **-fn**]
[**-F** 文件限制 | **-Fn**]
[**-i** 输入文件 | **-in** | **-is** 输入文件 | **-isn**]
[**-J** 作业名称 | **-J** "**%** 作业限制 " | **-Jn**]
[**-Jd** "作业描述" | **-Jdn**]
[**-k** 检查点目录 | **-k** "检查点目录 [检查点周期] " | **-kn**]
[**-L** 登录外壳 | **-Ln**]
[**-m** "主机名 [**+**[优先级]] |* 主机组 *[**+**[优先级]] … " | **-mn**]
[**-M** 内存限制 | **-Mn**]
[**-n** 处理器数量 | **-nn**]

[**-o** 输出文件 | **-on**]
[**-P** 项目名称 | **-Pn**]
[**-p** 进程限制 | **-Pn**]
[**-q** “队列名称… “ | **-qn**]
[**-R** “资源请求 “ | **-Rn**]
[**-sp** 优先级 | **-spn**]
[**-S** 堆栈限制 | **-Sn**]
[**-t** 终端时间 | **-tn**]
[**-u** *mail_user* | **-un**]
[**-w** ‘依赖表达式 ‘ | **-wn**]
[**-W** 运行时限 [/主机名 | /主机型号 *l*] | **-Wn**]
[**-Z** “新作业命令 “ | **-Zs** “新作业命令 “ | **-Zsn**]
[*job_ID* | “*job_ID*[*index*]”]
[**-h**]
[**-V**]

描述

修改先前提交的作业的选项。有关可使用 cmod 修改的作业提交选项的完整说明，请参阅*csub*。

只有作业所有者或 AIP 管理员才能修改作业的选项。

提交时指定的所有选项均可更改。每个选项的值都可以用新值覆盖，只需指定选项即可，就像在 csub 中一样。要将选项重置为其默认值，请使用选项字符串后跟 “n”。重置选项时请勿指定选项值。

-i、-in 和 -Z 选项具有支持输入和作业命令文件假脱机的对应选项（-is、-isn、-Zs 和 -Zsn）。

即使未指定相应的 csub 选项，您也可以修改等待作业的所有选项。

默认情况下，您可以修改正在运行的作业的资源预留 (**-R** “*res_req* “)。要修改正在运行的作业的其他作业选项，请在 cb.yaml 中定义 CB_MOD_ALL_JOBS=Y。

以下是仅对正在运行的作业有效的 **cmod** 选项。作业调度后，您无法进行任何其他修改。

- 资源预留 (**-R** “*res_req* “)
- CPU 限制 (**-c**[*hour***:**]*minute*[*/host_name* | */host_model*])
- 内存限制 (**-M** *mem_limit*)
- 运行限制 (**-W** *run_limit*[*/host_name* | */host_model*])
- 标准输出文件名 (**-o** *output_file*)
- 标准错误文件名 (**-e** *error_file*)
- 可重新运行的作业 (**-r** | **-rn**)
- 已分配插槽数量 (**-n**)。修改后的插槽数量只能减少。

修改后的资源限制不能超过队列中定义的资源限制。

要修改正在运行的作业的 CPU 限制或内存限制，必须在 cb.yaml 中定义参数 CB_JOB_CPULIMIT=Y 和 CB_JOB_MEMLIMIT=Y。

如果您想通过数组名称指定数组依赖关系，请在 cb.yaml 中设置 JOB_DEP_LAST_SUB。如果您未设置此参数，则如果先前的数组名称相同但索引不同，则该作业将被拒绝。

选项

***job_ID*| "*job_ID*[*index*]"**
修改指定作业 ID 的作业。

修改由 "*job_ID*[*index*]" 指定的作业数组元素。

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 发行版本打印到 stderr 并退出。

另请参阅

csub

限制

不允许修改作业数组的 **-q** 选项。

如果在作业调度前未指定 **-e**，则无法修改正在运行的作业的作业错误文件的名称。

5.1.27 cparams

命令

cparams - 显示 cb.yaml general 部分中配置的调度器参数的信息

概要

cparams [**-l**]

cparams [**-h** | **-V**]

描述

显示 cb.yaml general 部分中配置的调度器参数的信息。

默认情况下，仅显示简单配置的参数。

缺省用户权限下，集群管理员可以查看配置的信息。普通用户无法查看配置的信息。只有在 cb.yaml 的 general 中 user_view_alljobs 值为 yes 时，普通用户才可以查看配置的信息。

选项

-l
长格式。显示 cb.yaml 中所有 general 下可配置参数的详细信息。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

缺省输出

Default Queue
缺省队列名

Default Host Specification
缺省主机 CPU 型号

Job Dispatch Interval
调度器调度间隔

Job Checking Interval
CBJM 检查作业状态间隔

Job Accepting Interval
主机从调度器接受一个作业后，第二个作业的接受最小间隔

另请参阅

cb.yaml

5.1.28 cpasswd

命令

cpasswd - 在调度系统中注册 Windows 用户密码

概要

cpasswd [**-u** *用户名*] [**-r** | **-p** *密码*]

cpasswd [**-h** | **-V**]

描述

该命令用于在调度系统中注册 Windows 用户密码，以便 Windows 系统上运行的作业能够获得正确的访问权限。

所有用户都可以注册、检查或取消注册自己的密码。管理员和 root 可以注册、检查或取消注册任何用户的密码。

如果没有注册密码，Windows 系统上的作业可能没有访问共享文件系统的权限，尽管不需要访问共享文件系统的纯命令行作业仍然可以正常运行。

默认情况下，如果不使用任何选项，该命令将检查当前用户的密码注册。

要注册新密码或更新密码，请使用 cpasswd -p password。

选项

-u 用户名
: 针对指定用户进行操作。只有管理员可以使用此选项。如果不使用 -p 选项，该命令将检查指定用户的密码注册状态。

-p 密码
: 注册或更新指定的密码。此选项不能与 -r 选项一起使用。

-r
: 取消注册当前用户或指定用户的密码（使用 -u 选项）。此选项不能与 -p 选项一起使用。

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到标准输出并退出。

5.1.29 cping

命令

cping - 检查主机 AIP 服务的各守护进程的是否响应

概要

cping [*主机名* | **master**]

cping [**-h** | **-V**]

描述

与主机上主要的 AIP 服务进程送握手信号，检测是否返回。

默认情况下，向本机的服务进程 cbjm、cbexe、和 cbjm 送握手信号。

若定义了主机名，则向该主机的服务进程 cbjm、cbexe、和 cbjm 送握手信号。若需要检测调度器进程 cbsched 和 jservice，则使用保留词 **master** 。

选项

[主机名 | master]
选填。指定主机名，或者 master - 调度器主机。

缺省：运行该命令的本机

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

另请参见

cbls、*cbexe*、*cbjm*、*cbsched*、*jservice*。

5.1.30 cplace

命令

cplace -显示可用于执行任务的主机

概要

cplace [**-L**] [**-n** 最小数 | **-n 0**] [**-R** 资源需求] [**-w** 最大数 | **-w 0**] [主机名…]

cplace [**-h** | **-V**]

描述

显示可用于 ctask 的主机，并暂时增加这些主机的负载（以避免快速连续地向同一主机发送过多的作业）。过高的负载会随着时间的推移而缓慢衰减，直到调度任务产生的实际负载反映在 CBLS 的负载信息中。对于多处理器主机，主机名可能会重复，以指示可以将多个任务放置在单个主机上。

默认情况下，仅显示一个主机名。

默认情况下，使用 AIP 默认资源要求。

选项

-L
尝试将任务放置在尽可能少的主机上。这对于分布式并行应用程序非常有用，可以最大限度地降低任务之间的通信成本。

-n 最小数 | -n 0
至少显示指定数量的主机。指定 0 可显示尽可能多的主机。

如果找不到所需数量的拥有所需资源的主机，则打印“当前符合条件的主机不足”并以状态 1 退出。

-R 资源需求
仅显示具有指定资源要求的主机。这个命令只从 CBLS 得到符合资源需求的主机名，用于调度的资源需求不能用，包括资源 slots、资源需求 rusage、span、和 same。

-w 最大数 | -w 0
显示不超过指定数量的主机。指定 0 可显示尽可能多的主机。

主机名…
仅显示指定主机名范围内的主机。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

使用

cplace 主要用于反引号中，用于选取主机名，然后将其传递给其他命令。

-w 和 -n 选项可以组合使用，分别指定要返回的处理器的上限和下限。例如，命令 cplace -n 3 -w 5 返回至少 3 个、最多 5 个主机名。

另请参阅

cinfo、*cload*

诊断

如果可用主机不足，cplace 将返回 1。如果在 AIP 中检测到问题，则退出状态为 -10；如果检测到其他错误，则返回 -1；否则返回 0。

5.1.31 cpost

命令

cpost -向作业发送消息

概要

cpost -d “message” *job_ID* | **“***job_ID***[***index***]”**

cpost[**-h** | **-V**]

描述

允许用户向自己的作业发送消息。消息可以一直发送，直到作业从 AIP 中清除为止。向作业发送消息的数量没有限制，但每条消息的大小不能超过 1023 个字符。

用户只能向自己的作业发送消息。用户不能向其他人提交的作业发送消息。只有 root 和 AIP 管理员可以向其他用户提交的作业发送消息。

cpost 命令被用于发布作业应用的 URL，如 Jupyter Lab、Tensorboard、AI 推理等。这类任务被调度运行后，前端代码可以调用 cread 命令或许 URL，连接到作业应用。

选项

***job_ID* | "*job_ID*[*index*]"**
必填。要发送消息的作业 ID。

对于阵列作业，索引、方括号和引号是必需的。

-d
指定要发送至作业的消息。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 版本号打印到标准错误输出并退出。

示例

```
cpost -d "message in a bottle" 7249
Message to job 7249 posted.
```

另请参阅

cread 、*cjobs*

5.1.32 cqueues

命令

cqueues - 显示有关队列的信息

概要

cqueues [**-w** |**-l**] [**-e**] [**-o** "格式"] [**-m** 主机名 |**-m** 主机组 |**-m all**] [**-u** 用户名 |**-u** 用户组 |**-u all**] [队列名…]

cqueues [**-h** |**-V**]

描述

显示有关队列的信息。

默认情况下，返回有关所有队列的以下信息：队列名称、队列优先级、队列状态、作业槽统计信息和作业状态统计信息。

队列名称和特性由管理员设置。

显示队列信息范围

- root 或集群管理员可以访问所有队列信息。
- 当 cb.yaml 中 user_view_alljobs: yes，普通用户可以访问所有队列信息。
  - 当环境变量 CB_LIMIT_VIEW（或者在/opt/skyformai/etc/cb.conf 中设置）设任何值，普通用户只能访问自己可用队列的信息。

  例子:

```
[u001@aipm ~]$ cqueues
QUEUE_NAME      PRIO      STATUS      MAX  JL/U JL/P JL/H NJOBS  PEND  RUN  SUSP
test             10    Open:Active      -    -    -    -     0     0     0     0
qm                4    Open:Active      -    -    -    -     0     0     0     0
qg                4    Open:Active      -    -    -    -     0     0     0     0
medium            3    Open:Active      -    -    -    -     0     0     0     0
vnc               3    Open:Active      -    -    -    -     0     0     0     0
[u001@aipm ~]$ CB_LIMIT_VIEW=1 cqueues
QUEUE_NAME      PRIO      STATUS      MAX  JL/U JL/P JL/H NJOBS  PEND  RUN  SUSP
qm                4    Open:Active      -    -    -    -     0     0     0     0
```

- 当 cb.yaml 中 user_view_alljobs: no，或者该参数没有设置，用户只能访问自己可用队列的信息。

选项

-w
以宽格式显示队列信息。字段显示时不带截断，并显示等待作业数（PJOBS）。

-l
以多行长格式显示队列信息。-l 选项显示以下附加信息：队列描述、队列特征和统计信息、调度参数、资源限制、调度策略、用户、主机、用户共享、窗口、相关命令以及作业控制。

-e
以长多行格式展开用户组和主机组。

-m 主机名 |-m 主机组 |-m all
显示可在指定主机或主机组上运行作业的队列。如果指定关键字 all，则显示可在所有主机上运行作业的队列。有关主机组列表，请参见*cmgroup*。

-o "字段名称…[delimiter='字符']"
通过字段名称指定自定义输出格式。使用 *delimiter=* 指定在不同标题和字段之间显示的分隔符。分隔符必须是单个字符。默认情况下，分隔符为空格。

如果字段没有值，则输出为破折号 (-)。

可用的字段名称有：

queue | q：队列名称

description：队列描述

priority：队列优先级

users：允许使用队列的用户

hosts：属于队列的主机

userjoblimit | ulj：每个用户作业槽限制

status：队列状态。参见 OUTPUT

maxslots | maxj：配置的最大作业槽限制

numjobs：作业槽数量（包括等待、运行、暂停和保留）

numpend：等待作业槽的数量

pjobs：等待作业的数量

numrun：正在运行的作业槽数

numssusp：系统暂停作业槽的数量

numususp：用户暂停作业槽的数量

numreserve：调度程序保留的作业槽数

fairshare：Fairshare 配置字符串

maxavail_slots：基于队列中的主机和主机可用性的最大可用作业槽

avail_slots：空闲职位槽

maxavail_gpus：主机上配置的队列使用的最大可用 GPU 数量

avail_gpus：队列中可用的空闲 GPU

maxavail_memgb：基于队列中的主机和主机可用性的最大可用内存（以 GB 为单位）

avail_memgb：队列中所有主机的可用内存（以 GB 为单位）

jlh：每个主机配置的作业槽限制

weighted_ut | ut：队列中所有可用主机的加权 CPU 利用率

-u 用户名 |-u 用户组 |-u all
显示可以接受来自指定用户或用户组的作业的队列（有关用户组的列表，请参阅*cugroup*。）如果指定了关键字“all”，则显示可以接受来自所有用户的作业的队列。

队列名..
显示有关指定队列的信息。

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 发布版本打印到标准输出并退出。

输出

默认输出

显示以下字段：

QUEUE_NAME
队列的名称。队列的命名与通常提交给它的作业类型或它提供的服务类型相对应。

lost_ane_found
如果 AIP 管理员从系统中删除队列，AIP 会创建一个名为 lost_and_found 的队列，并将被删除队列中的作业放入 lost_and_found 队列中。lost_and_found 队列中的作业除非被切换到其他队列（参见 cswitch），否则不会启动。

PRIO
队列的优先级。如果未配置作业优先级，则决定作业调度、暂停和恢复时的队列搜索顺序。作业调度和恢复时，会首先搜索优先级值较高的队列（这与 Linux 进程优先级顺序相反），作业暂停时，会最后搜索优先级值较高的队列。

STATUS
队列的当前状态。可能的值包括：

Open
打开：队列能够接受作业提交。

Closed
关闭：队列无法接受作业提交。

Active
激活：队列中的作业可被调度。

Inactive
非活动：队列中的作业暂时无法被调度。

每个队列在任何时候都处于“Open”或“Closed”状态，以及“Active”或“Inactive”状态。AIP 管理员可以使用 csadmin（参见*csadmin*）打开、关闭、停用和重新激活队列。当队列的调度窗口或运行窗口关闭时，队列将变为非活动状态（参见“-l 选项的输出”部分中的 DISPATCH_WINDOWS）。在这种情况下，无法使用 csadmin 激活队列。当队列的其中一个调度窗口和其中一个运行窗口再次打开时，AIP 会重新激活队列。队列在 AIP 启动时的初始状态设置为“打开”，并根据其窗口状态变为活动或非活动状态。

MAX
队列中作业可使用的最大作业槽数。这些作业槽由已调度但尚未完成的作业以及已预留槽的等待作业使用。顺序作业在调度到主机时将使用一个作业槽，而并行作业在调度时将使用 csub -n 所需的作业槽数量。详情请参阅*csub*。如果显示“-”，则表示没有限制。

JL/U
队列中作业的最大作业槽位数量。这些作业槽位将由已调度但尚未完成的作业以及已预留槽位的等待作业使用。如果显示“-”，则表示没有限制。

JL/P
处理器可以从队列中处理的最大作业槽数。这包括已调度但尚未完成的作业槽，以及为某些等待作业保留的作业槽。每个处理器的作业槽限制 (JL/P) 控制发送到每台主机的作业数量。此限制是按处理器配置的，以便多处理器主机自动允许运行更多作业。如果显示“-”，则表示没有限制。该指标在 AIP 配置中不能修改，仅为 LSF 兼容所用。

JL/H
主机可以从队列中处理的最大作业槽数。这包括已调度但尚未完成的作业槽，以及为某些等待作业保留的作业槽。每台主机的作业槽限制 (JL/H) 控制发送到每台主机的作业数量，无论主机是单处理器主机还是多处理器主机。如果显示“-”，则表示没有限制。该指标在 AIP 配置中不能修改，仅为 LSF 兼容所用。

NJOBS
队列中当前作业占用的作业槽总数。这包括等待、正在运行、暂停和预留的作业槽。在 n 个处理器上运行的并行作业计为 n 个作业槽，因为它占用队列中的 n 作业槽。有关作业状态的说明，请参阅*cjobs*。

PEND
队列中等待的作业槽的数量。

RUN
队列中正在运行的作业槽的数量。

SUSP
队列中暂停的作业槽的数量。

RSV
仅限宽格式（-w）。队列中由 AIP 为等待作业保留的作业槽数。

PJOBS
仅限宽格式（-w）。队列中等待作业的数量。

长输出-l 选项的输出

除了上述字段之外，**-l** 选项还显示以下内容：

Description
队列的典型用途的描述。

NICE
队列中作业运行时的 nice 值。这是 Linux 中用于降低进程优先级的 nice 值（参见 nice (1)）。

PARAMETETERS/STATISTICS

STATUS

Inactive
-l 选项的长格式给出了队列不活动的可能原因：

Inact_Win
队列超出了其调度窗口或运行窗口。

Inact_Adm
该队列已被 AIP 管理员停用。

SSUSP
队列中分配给由 AIP 暂停的作业的作业槽数。

USUSP
队列中分配给由作业提交者或 AIP 管理员暂停的作业的作业槽数。

RSV
队列中由 AIP 为等待作业保留的作业槽数量。

PJOBS
队列中等待作业的数量。

Migration threshold
在 AIP 尝试将作业迁移到其他主机之前，从队列中调度的作业将被系统暂停的时间（以秒为单位）。该数据是为 LSF 兼容所设，AIP 里没有对应配置参数。

Schedule delay for a new job
新作业提交后，调度会话的延迟时间（以秒为单位）。如果调度延迟时间为零，则作业提交到队列后会立即启动新的调度会话。在 AIP 中这个参数总是 0。

Interval for a host to accept two jobs
将一个作业调度到主机后，需要等待多少秒才能将第二个作业调度到同一主机。如果作业接受间隔为零，则主机可以在每个调度间隔内接受多个作业。请参阅 cb.yaml 中 queues 和 general 里的 job_accept_interval 参数。

RESOURCE LIMITS
对队列中的作业施加的硬资源限制（请参阅 getrlimit (2) 和*cb.yaml*）。这些限制是针对每个作业和每个进程施加的。

每个作业可能的限制是：

CPULIMIT

PROCLIMIT

MEMLIMIT

SWAPLIMIT

Linux 每个进程可能的资源限制是：

RUNLIMIT

FILELIMIT

DATALIMIT

STACKLIMIT

CORELIMIT

如果提交到队列的作业指定了这些限制中的任何一个（请参阅*csub*），则将对该作业使用相应作业限制和队列限制中较低的一个。

如果没有指定资源限制，则假定资源是无限的。

SCHEDULING PARAMETERS

队列的调度和暂停阈值。

调度阈值 loadSched 和暂停阈值 loadStop 用于控制作业的调度、暂停和恢复。队列阈值与主机定义的阈值结合使用（参见*chosts* 和*cb.yaml*）。如果同时配置了队列级别和主机级别的阈值，则应用最严格的阈值。

loadSched 和 loadStop 阈值具有以下字段：

r15s
15 秒指数平均有效 CPU 运行队列长度。

r1m
1 分钟指数平均有效 CPU 运行队列长度。

r15m
15 分钟指数平均有效 CPU 运行队列长度。

ut
过去一分钟内 CPU 利用率的指数平均值，以 0 到 1 之间的百分比表示。

pg
过去一分钟内内存分页率的指数平均值，以每秒页数为单位。

io
过去一分钟内网络 I/O 速率的指数平均值，以每秒千字节为单位。

up
主机正常运行时间（以分钟为单位）。

it
主机的空闲时间（所有登录会话中未触摸键盘），以分钟为单位。

tmp
/tmp 中的可用空间量（以 MB 为单位）。

swp 当前可用的交换空间量（以 MB 为单位）。

mem
当前可用内存量（以 MB 为单位）。

gpu
可用的 GPU 卡数

除了这些内部索引之外，如果在 RESS 中定义了外部索引，则也会显示它们（参见 ress(8)）。

loadSched 阈值指定了相应负载指标的作业调度阈值。如果值显示为 “-”，则表示该阈值不适用。如果主机所有负载指标的值均在队列和主机的相应阈值范围内（低于或高于，取决于负载指标的含义），则队列中的作业可以调度到该主机。相同的条件也适用于恢复已在该主机上暂停的队列调度作业。

类似地，loadStop 阈值指定了作业暂停的阈值。如果主机上的任何负载指标值超出了队列的相应阈值，则队列中的作业将被暂停。

SCHEDULING POLICIES

队列的调度策略。可以配置以下一个或多个策略：

IGNORE_DEADLINE

如果 IGNORE_DEADLINE 设置为 Y，则无论运行限制如何都启动所有作业。

EXCLUSIVE

如果用户在提交作业时指定，从独占队列调度的作业可以在某台主机上独占运行（参见*csub*）。独占执行意味着该作业将被发送到一台没有其他作业正在运行的主机上，并且在该作业运行期间，不会再有其他作业（无论是作业还是交互作业）调度到该主机。默认情况下，不允许独占作业。

NO_INTERACTIVE

此队列不接受批量交互式作业。（请参见*csub*）的 -I、-Is 和 -Ip 选项）。默认情况下，它同时接受交互式和非交互式作业。

ONLY_INTERACTIVE

此队列仅接受批量交互式作业。作业必须使用 csub (1) 的 -I、-Is 和 -Ip 选项提交。默认情况下，同时接受交互式和非交互式作业。

OWNED_HOSTS

队列拥有的主机，这些主机上拥有者的作业可以抢占其他队列或其他用户在这些主机上的作业。

FAIRSHARE

此队列使用公平共享策略来调度作业。屏幕上会显示公平共享树的用户、份额、优先级以及等待和正在运行的作业数量。参见:ref:*cb-yaml*。

PREEMPTION

此队列使用抢占策略，输出指定哪些队列将被抢占，其作业将被重新排队。参见*cb.yaml*。

DEFAULT HOST SPECIFICATION

用于规范所有作业的 CPU 时间限制的默认主机或主机模型。

如果要查看集群中主机定义的 CPU 因子（CPU Factor）列表，请使用*cinfo*。CPU 因子由 AIP 自动检测。

主机或主机型号的相应 CPU 缩放系数用于调整执行主机的实际 CPU 时间限制（请参阅*cb.yaml* 中的 CPULIMIT）。cb.yaml 中的 DEFAULT_HOST_SPEC 参数会覆盖 cb.yaml 中的系统 DEFAULT_HOST_SPEC 参数（请参阅*cb.yaml*）。如果用户在提交作业时使用 csub -c cpu_limit [/ host_name | / host_model] 明确指定了主机规范，则用户规范将覆盖 cb.yaml 和 cb.yaml 中定义的值。

RUN_WINDOWS

一周内有一个或多个运行窗口，在此期间队列中的作业可以运行。

当一个运行窗口结束时，队列中所有正在运行的作业都会被暂停，直到下一个运行窗口开始时才会恢复。默认设置是无限制，即始终开放。

DISPATCH_WINDOWS

队列的调度窗口。调度窗口是一周内可以调度队列中作业的时间窗口。

当队列超出其调度窗口时，队列中将不会调度任何作业。已调度的作业不受调度窗口的影响。默认设置是无限制或始终开放（即每周 7 天、每天 24 小时）。请注意，此类调度窗口仅适用于作业。可以为单个主机配置类似的调度窗口（参见*chosts*）。

窗口以 *begin_time* - end_time 格式显示。时间以 [day :] hour [: minutes] 格式指定，其中所有字段均为合法范围内的数字：day 为 0（星期日）-6，hour 为 0-23，minute 为 0-59。minute 的默认值为 0（整点）。day 的默认值为每周的每一天。窗口的 begin_time 和 end_time 以 “-” 分隔，中间不能有空格（SPACE 和 TAB）。窗口的 begin_time 和 end_time 必须同时存在。窗口以空格分隔。

USERS

允许向队列提交作业的用户和用户组列表。用 户组名称末尾会添加一个斜杠 (/)。请参阅*cugroup*。当使用-e 参数时（cqueues -l -e），这个列表会展开到单个用户名。

AIP 集群管理员可以默认将作业提交到队列。

HOSTS
可调度队列中作业的主机和主机组列表。主机组名称末尾会添加一个斜杠 (/)。请参阅*cmgroup*。当使用-e 参数时（cqueues -l -e），这个列表会展开到单个主机名。

ADMINISTRATORS
队列管理员列表。列出的用户有权操作队列中的作业以及队列本身。更多信息，请参阅*cb.yaml*

PRE_EXEC
队列的预执行命令。预执行命令在队列中的每个作业在执行主机（或为并行作业选择的第一个主机）上运行之前执行。有关更多信息，请参阅*cb.yaml*

POST_EXEC
队列的执行后命令。执行后命令在作业终止时运行。更多信息请参阅*cb.yaml*

REQUEUE_EXIT_VALUES
以这些值退出的作业将自动重新排队。更多信息请参阅*cb.yaml*

RES_REQ
队列的资源需求。只有满足这些资源需求的主机才能被队列使用，由 cb.yaml 里队列的 resspec 定义。

Maximum slot reservation time
队列中等待作业在 slot 中保留的最长时间（以秒为单位）。请参阅 cb.yaml 中的 slotreservetime 参数。

RESUME_COND
恢复队列中已暂停作业的阈值条件。该参数在 AIP 中暂无意义。

STOP_COND
队列中正在运行的作业的停止阈值条件。该参数在 AIP 中暂无意义。

JOB_STARTER
队列中正在运行的作业的作业启动命令。更多信息请参阅*cb.yaml* 里 queues 定义的 job_starter。

RERUNNABLE
如果 RERUNNABLE 字段显示为“yes”，则表示队列中的作业可重新运行。也就是说，如果执行主机不可用，队列中的作业将自动重新启动或重新运行。但是，如果您已从作业中移除 rerunnable 选项，则队列中的作业将不会重新启动。有关更多信息，请参阅*cb.yaml* 。

CHECKPOINT
如果显示 CHKPNTDIR 字段，则表示队列中的作业可以执行检查点操作。除非您指定其他值，否则作业将使用默认的检查点目录和周期。请注意，如果您已从作业中移除检查点选项，则队列中的作业将不会执行检查点操作。

CHKPNTDIR
使用绝对或相对路径名指定检查点目录。

CHKPNTPERIOD
指定检查点周期（以秒为单位）。

尽管 cqueues 的输出以秒为单位报告检查点周期，但检查点周期是以分钟为单位定义的（检查点周期通过 csub -k ” checkpoint_dir [checkpoint_period]” 选项定义，或在 cb.yaml 中定义）。

JOB CONTROLS
用于作业控制的已配置操作。请参阅 cb.yaml 中的 stop_action, resume_action, kill_action 参数。

配置的操作以 *[action_type* , command] 格式显示，其中 action_type 为 SUSPEND、RESUME 或 TERMINATE。

MAX AVAILABLE SLOTS
队列中工作正常的主机上所有作业槽总和。

FREE SLOTS
队列中工作正常的主机上所有可用作业槽数。

GPU Resources
队列中工作正常的主机上各个型号 GPU 的可用数/总数。

5.1.33 crcp

命令

crcp -使用 AIP 远程文件复制

概要

crcp [**-a**] [**-t**] [**-m** | **-v**] 源路径 目标路径

crcp [**-s**] [**-t**] [**-o**] [**-i** 秒] 源路径 目标路径

crcp [**-h** | **-V**]

描述—

使用 AIP 远程文件复制。

crcp 是一个支持 AIP 的远程文件复制程序，可在 AIP 集群中的主机之间传输单个文件或目录树。crcp 使用 AIP 主机上的 CBEXE 服务传输文件。如果主机上未安装 AIP 或 CBEXE 未运行，则 crcp 将使用 rcp 复制文件。

要使用 crcp，您必须对要复制的文件具有读取权限。

源文件和目标文件都必须属于发出该命令的用户。

crcp 使用 rcp 将源文件复制到另一个用户拥有的目标文件。详情请参阅下面的限制。

选项

-a
将 *source_path* 附加到 *target_path*。

-v
详细模式。显示源绝对路径和目标绝对路径。

-m
模拟模式。显示源绝对路径和目标绝对路径，但不复制文件。

-t
树模式。将所有文件和目录树从 *source_path* 复制到 *target_path*。*source_path* 必须是本地目录。

-s
流式传输模式。持续将 *source_path* 同步到 *target_path*，直至按下 CTRL-C 键。

-o
在流式传输模式下覆盖 *target_path*。

-i 秒数
在流式传输模式下将 *source_path* 同步到 *target_path* 的间隔。默认值为 2 秒。

source_path target_path
指定要复制的本地或远程主机上的现有文件，以及要将源文件复制到的目标文件。

文件格式如下：

[[*host_name*]**:**][*path*/][*file_name*]

host_name
文件所在的远程主机的名称。如果未指定 *host_name*，则使用发出命令的本地主机。

path
绝对路径名或相对于用户登录目录的路径名。本地或远程主机均不支持 Shell 文件名扩展。

如果 source_path 中未指定 file_name，则会复制除路径本身之外的整个子目录和文件树。

使用“/”将文件从 Linux 主机传输到 Linux 主机。例如:

```
crcp file1 hostD:/home/usr2/test/file2
```

file_name
源文件的名称。不支持文件名通配符。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

示例

% **crcp myfile @hostC:/home/usr/dir1/otherfile**

将文件 myfile 从本地主机复制到 hostC 上的文件 otherfile。

% **crcp user1@hostA:/home/myfile user1@hostB:otherfile**

将文件 myfile 从 hostA 复制到 hostB 上的文件 otherfile。

% **crcp -a user1@hostD:/home/myfile /dir1/otherfile**

将 hostD 上的文件 myfile 附加到本地主机上的文件 otherfile。

% **crcp /tmp/myfile user1@hostF:~/otherfile**

将文件 myfile 从本地主机复制到 hostF 上 user1 主目录中的文件 otherfile。

% **crcp -s /tmp/myfile hostB:/home/usr/myfile**

以流式传输模式将文件 myfile 从本地主机复制到 hostB 上的文件 myfile。该命令直到输入 CTRL-C 才会停止。

% **crcp -t /tmp/mydir hostB:/home/usr/mydir**

将文件和目录树 mydir 从本地主机复制到 hostB 上的目录 mydir。

% **crcp -t -s /tmp/mydir hostB:/home/usr/mydir**

以流式传输模式将文件和目录树 mydir 从本地主机复制到 hostB 上的目录 mydir。该命令直到输入 CTRL-C 才会停止。

另请参见

rsh(1)、rcp(1)

诊断

crcp 尝试使用 CBEXE 将 *source_path* 复制到 *target_path*。如果 CBEXE 关闭或无法复制 *source_path*，crcp 将使用 rsh（如果指定了 -a 选项）或 rcp（如果未指定 -a 选项）。

限制

以下情况下不支持使用 crcp 进行文件传输：

- 如果使用了 AIP 帐户映射；crcp 在其他用户帐户下运行时会失败
- 在 AIP 客户端主机上。AIP 客户端主机不运行 CBEXE，因此 crcp 无法联系提交主机上的 CBEXE
- 第三方复制。当源文件和目标文件都不在本地主机上时，crcp 不支持第三方复制。在这种情况下，将使用 rcp 或 rsh。如果 *target_path* 存在，crcp 将保留这些模式；否则，crcp 将使用经过源主机的 umask（参见 umask(2)）修改的 *source_path* 模式。

您可以执行以下操作：

如果 crcp 无法联系提交主机上的 CBEXE，它将尝试使用 rcp 复制文件。您必须设置 /etc/hosts.equiv 或 HOME/.rhosts 文件才能使用 rcp。有关使用 rcp 命令的更多信息，请参阅 rcp(1) 和 rsh(1) 手册页。

您可以将 crcp 替换为您自己的文件传输机制 (如 scp)，只要它支持相同的语法即可。这样做可能是为了利用更快的互连网络，或者克服现有 crcp 的限制。CBJM 会在 cb.yaml 文件中指定的 CB_BINDIR (/opt/skyformai/bin) 目录中查找 crcp 可执行文件。

5.1.34 cread

命令

cread -从作业中读取消息

概要

cread *job_ID* | "*job_ID*[*index*]"

cread[**-h** | **-V**]

描述

允许用户读取之前发布到作业的消息。在作业从 AIP 系统中清除之前，消息一直处于读取状态。用户可以读取其他用户提交的作业的消息。

选项

***job_ID* | "*job_ID*[*index*]"**
必填。读取消息的作业 ID。

对于数组作业，索引、方括号和引号是必需的。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

示例

```
cread 7249
Messages posted to jobID 7249
POST_TIME: Apr  6 14:03:42  MESSAGE: message in a bottle
```

另请参阅

cpost 、*cjobs*

5.1.35 creduce

命令

creduce -请求释放作业的部分作业槽位

概要

creduce [**-r** *release_slot_spec*] *job_ID*

creduce[**-h** | **-V**]

描述

请求释放作业的部分作业槽位，并为作业添加额外的作业槽位，例如，减少作业的大小。

默认情况下，如果不使用 -r 选项，则释放作业的最后一个作业槽位。

选项

-r *release_slot_spec*
指定要为作业释放的槽位。

release_slot_spec 资源需求的格式为："~ 最后的槽位数 | 资源需求段数:~ 最后的槽位数 | 槽位数 1, 槽位数 2,…"。

资源需求段数从 0 开始。槽位号也从 0 开始。

示例 1："-r ~5" 释放作业的最后 5 个槽位。

示例 2：“-r 0:~4 1:3,4,8”释放第一个资源需求段的最后 4 个槽位，即第二个资源需求段的槽位 #3、#4、#8。

job_ID
需要缩减的作业 ID。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发布版本打印到标准错误输出并退出。

另请参见

csub 、*cexpand*

5.1.36 creport

命令

creport - 生成与作业相关的报告

概要

creport [**-C** *time0,time1*] [**-f** “*cb.acct* 文件名…” |*cb.acct* 目录名 [+]] [**-a** “应用程序名…”] [**-m** “主机名…”] [**-P** “项目名…”] [**-q** “队列名…”] [**-u** “用户名…”] [**-G** “用户组名…”] [**-j** “*jobId* …”] [**-l**] [**-p** *gnuplot* 输出文件名] [**-r** 报告类型] [**-i** 数据点间隔 [unit]] [**-JSON**] [**-z en|zh**]

creport [**-h**] [**-V**]

描述

根据 AIP 会计和作业事件数据生成与作业相关的报告。

该工具默认以 csv 格式输出数据，以便轻松导入其他工具。它还集成了开源 gnuplot，可以图形化地绘制数据。

它可以生成以下报告。所有报告均基于指定的时间段。

- 已提交的作业数量
- 已完成、已退出或已完成的作业数量
- 等待作业的平均数量
- 已暂停作业的平均数量
- 作业槽位平均使用率

数据可以按以下方式分隔：

- 用户
- 队列
- 项目

- 用户组
- 应用程序名称

还可以通过指定以下范围来缩小报告范围：

- 用户名
- 队列名称
- 项目名称
- 用户组名称
- 应用程序名称
- 主机名
- 时间段

所需环境

如果您想创建 gnuplot 图表，您需要安装 gnuplot v5 并将 gnuplot 模板文件 bgnu.temp 复制到当前工作目录或 $CB_ENVDIR。

选项

-h
显示使用信息。如果任何参数有误，也会显示使用信息。

-V
显示版本信息。

-f "*file_name* …" |*path*[+]
指定 AIP cb.acct 文件名。可以使用引号指定多个文件名，并用空格分隔，例如 "cb.acct cb.acct1"。如果参数是目录路径，则将使用该目录中的所有 cb.acct* 文件。如果参数是目录路径加上 "+" 字符（例如 /opt/skyformai/work/data+），则将同时读取该目录中的 cb.acct* 文件以及 cb.events* 文件。

如果未指定此选项，则将使用当前安装的 AIP 环境中的 cb.acct 文件。

-C *time0,time1*
指定报告时间段。其语法与 bhist 命令中的时间语法相同。如果未指定此选项，则将报告事件或/和会计文件所涵盖的整个时间段。

-a "应用程序名…"
仅缩小指定应用程序名称的报告数据范围。可以在引号内指定多个应用程序名称，并用空格分隔。应用程序名称在 csub -A 或 csub -app 选项中指定。默认情况下，报告具有任何应用程序名称（包括空应用程序名称）的作业。

-m "主机…"
使用在指定主机上运行的作业的数据。可以在引号内指定多个主机，并用空格分隔。如果未指定，则使用 AIP 系统中的所有主机。

-P "项目名…"
使用具有指定项目的作业的数据。可以在引号内指定多个项目，并用空格分隔。如果未指定，则使用 AIP 系统中的所有项目。

-q "队列名…"
使用在指定队列中运行的作业的数据。可以在引号内指定多个队列，并用空格分隔。如果未指定，则使用 AIP 系统中的所有队列。

-u "用户名…"
使用由指定用户提交的作业的数据。多个用户可以用引号括起来并用空格分隔。如果未指定，则报告所有运行过作业的用户。

-G "用户组名…"
使用指定用户组运行的作业数据。多个用户组可以用引号括起来并用空格分隔。请注意，会计文件不包含用户组信息，因此无法使用此选项。

-j "*jobId* …"
使用指定作业 ID 的作业数据。多个作业 ID 可以用引号括起来并用空格分隔。对于数组作业，只能指定作业 ID。如果未指定此选项，则将使用 AIP 系统中的所有作业。

-l
列出所有已报告的作业，但不生成报告。

-JSON
输出 JSON 格式。

-z en | zh
选择 JSON 报告语言。"en" 为英语，这是默认语言。"zh" 为简体中文。

-p *gnuplot* 输出文件路径
使用 gnuplot 生成报告图表

-r 报告类型
指定报告类型。报告类型名称的格式为 "统计项: 类别"。

统计项可以是以下名称之一：

- **submit**：已提交作业数
- **done**：已完成且退出代码为 0 的作业数
- **exited**：已退出且退出代码非零的作业数
- **completed**：已完成和已退出的作业数
- **run**：平均作业槽位 ge
- **pending**：等待作业的平均数量
- **suspend**：暂停作业的平均数量

类别是以下名称之一：

- **user**：按用户分类的数据
- **queue**：按队列分类的数据
- **project**：按项目分类的数据
- **app**：按应用名称分类的数据
- **ugroup**：按用户组分类的数据
- **all**：数据未分类

如果不指定该选项，则默认生成的报告为 "**run:all**"。

-i *interval*[unit]
指定两个报告数据之间的时间间隔（以秒为单位）。例如："-i 3" 表示 3 秒。

单位为可选字符。

- 'm'：分钟
- 'h'：小时

- ‘d’：天

如果不指定选项，则会自动计算间隔，以便在整个报告周期内生成大约 160 个报告数据点。

使用

该工具可用于生成多种不同类型的报告。

按用户划分的槽位利用率

命令示例:

```
creport -r run:user -p run-user.png
```

报告：用户随时间变化的作业槽位使用情况。此报告显示 AIP 分配给用户的作业槽位随时间变化的份额。该命令从 AIP 集群中的 cb.acct 文件中读取数据，并调用 gnuplot 生成报告图像文件 run-user.png。

集群槽位利用率

同样，上图也显示了 AIP 集群的运行情况。

图表中的顶线表示随时间变化的作业槽位总数。当有作业运行时，使用率接近 100%。

如果您只想查看集群利用率的输出，而不查看按用户细分的数据，只需运行:

```
creport -r run -p run.png
```

或者，如果您想将数据导入电子表格或其他工具，您可以让工具将数据写入标准输出，然后将其捕获到文件中。例如:

```
creport -r run > run.csv
creport -r run:user > run-user.csv
creport -r run:queue > run-queue.csv
```

作业吞吐量

要报告 AIP 系统的作业吞吐量，请运行以下命令显示作业完成的速率:

```
creport -r comp:user -p comp-user.png -i 1h
```

在上图中，我们看到每小时完成（完成或退出）的作业数量，这实际上就是作业吞吐量。

作业失败

如果您想检查有多少作业以非零状态退出（表示失败），可以运行以下命令:

```
creport -r exit -C 2024/10/30 -i 1h
```

这将报告 2024 年 10 月 30 日退出的作业数量，每小时一行。如表 1 所示，您可以以秒、分钟、小时、天或月为间隔生成类似的统计数据。

调度器性能

通常，当等待作业足够多时，系统中的槽位使用率应该很高。有时，即使有空闲的作业槽位，作业也未进行调度。作业未调度的常见原因之一是资源需求未得到满足。

首先，我们会生成一份按资源需求划分的作业槽位使用情况报告。如果发现有空槽位，我们可能会生成另一份按资源需求划分的等待作业报告。

然后，我们来查看等待作业报告。

如果我们发现在所示时间段内有很多等待作业。

存在等待作业且集群未得到充分利用的事实表明，其他因素阻止了这些作业的调度。我们可能会查看 AIP 配置以进一步调查。

放大感兴趣的时间段

要放大到较小的时间段，请使用–C 选项指定时间段。要获取更多数据点，您还可以使用–i 开关缩短报告数据点之间的间隔。允许的最小间隔为 1 秒。

您还可以通过指定队列名称、项目名称、用户名、用户组名称、主机名、作业 ID 或任何组合来缩小数据范围。

时间规范

-C 选项中的“time0,time1”必须符合以下规定：

time_argument = ptime,ptime | ptime, | ,ptime | itime

ptime = 日 | /日 | 月/ | 年/月/日 | 年/月/日/ | 小时: | 月/日 | 年/月/日/小时: | 年/月/日/小时: 分钟 | 日/小时: | 月/日/小时: | 日/小时: 分钟 | 小时: 分钟 | 月/日/小时: 分钟 | -itime

itime = ptime 日、月、小时、分钟 = 两位数字

其中，“ptime”代表特定时间点，“itime”代表特定时间间隔，“.”代表当前月/日/小时: 分钟。

记住以下规则将帮助您自由指定时间：

- 年份必须为 4 位数字，后跟 /
- 月份必须后跟 /
- 日期必须前跟 /
- 小时后必须加:
- 分钟前必须加:
- 当 day 单独出现时，day 前的 / 可以省略 e 或当 day 后跟 /hour 时：
- 时间格式中不允许有空格，即时间必须为单个字符串。

上述时间格式旨在方便灵活地指定时间。

请参阅以下示例：

假设当前时间为 2025 年 3 月 9 日 17:06:30。

1,8

从 2025 年 3 月 1 日 00:00:00 到 2025 年 3 月 8 日 23:59:00；

,4 或,/4

从记录第一个作业的时间到 2025 年 3 月 4 日 23:59:00；

6 或 /6

从 2025 年 3 月 6 日 00:00:00 到 2025 年 3 月 6 日 23:59:00；

2/

从 2025 年 2 月 1 日 00:00:00 到 2025 年 2 月 28 日 23:59:00；

12:

从 2025 年 3 月 9 日 12:00:00 到 2025 年 3 月 9 日 12:59:00；

2/1

从 2025 年 2 月 1 日 00:00:00 到 2025 年 2 月 1 日 23:59:00；

2/1,

从 2 月 1 日 00:00:00 到当前时间；

,. 或,

从记录第一个作业的时间到当前时间；

,.-2

从记录第一个作业的时间到 2025 年 3 月 7 日 17:06:30；

,.-2/

从记录第一个作业到 2025 年 1 月 9 日 17:06:30；

,2/10:

从记录第一个作业到 2025 年 3 月 2 日 10:59:00；

2024/11/25,2025/1/25

从 2024 年 11 月 25 日 00:00:00 到 2025 年 1 月 25 日 23:59:00；

限制

目前，creport 仅使用 AIP 统计和事件数据文件作为信息来源。它有以下限制：

- AIP 统计和事件文件中不包含 AIP 配置信息。该工具无法以百分比形式报告插槽利用率。
- 如果您仅使用 cb.acct（会计文件）作为数据源，则其中没有用户组信息。作业暂停和恢复信息也会缺失。
- 没有作业等待原因的数据。
- 没有系统负载的数据。
- 报告中的时间值为本地时间。如果创建数据的地点与 creport 运行的地点处于不同的时区，请注意时差。例如，对于 11 月 1 日晚上 11:00 在新加坡运行的作业，如果在北美东部标准时间运行报告，则显示为 11 月 1 日上午 10:00。

另请参阅

cacct、*cb.acct*

5.1.37 crequeue

命令

crequeue -终止并重新排队作业

概要

crequeue [-J *job_name* **| -J "***job_name***[***index_list***]"] [-u** *user_name* **| -u all**][*job_ID* **| "***job_ID***[***index_list***]"][-d][-e][-r][-a][-H]**

crequeue [**-h | -V**]

描述

除非您是 root 或 AIP 管理员，否则您只能对您拥有的作业使用 crequeue。

终止正在运行 (RUN)、用户暂停 (USUSP) 或系统暂停 (SSUSP) 的作业，并将其返回到原始队列，并使用相同的作业 ID。被终止并重新排队的作业将保留其提交时间，但会根据其重新排队时间进行调度。重新排队时，作业将被分配 PEND 状态，如果使用了 -H 选项，则将被分配 PSUSP 状态。调度后，作业将从头开始执行。

使用 crequeue 重新排队作业阵列或其元素。

默认情况下，如果未指定 job_ID，则会终止并重新排队最近提交的作业。

选项

-J ***job_name*** **| -J "***job_name***[***index_list***]"**
对指定的作业进行操作。

由于作业名称不唯一，多个作业阵列可能具有相同的名称，但索引集不同或相同。

-u ***user_name*****|-u all**
对指定用户的作业或所有作业进行操作。

只有 root 和 AIP 管理员可以重新排队其他用户提交的作业。

job_ID**| "***job_ID***[***index_list***]"**
对指定的作业或作业阵列元素进行操作。

job_ID 的 0 值将被忽略。

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 发行版本打印到 stdout 并退出。

-d
将已完成运行且作业状态为 DONE 的作业重新排队。

-e
将异常终止且状态为 EXIT 的作业重新排队。

-r
将正在运行的作业重新排队。

-a
将所有作业重新排队，包括正在运行的作业、暂停的作业以及状态为 EXIT 或 DONE 的作业。

-H
将作业重新排队至 PSUSP 作业状态。

限制

crequeue 不能用于交互式作业；crequeue 只会终止交互式作业，而不会重新启动它们。

5.1.38 cresources

命令

cresources - 显示正在运行的作业的资源限制信息

概要

cresources [**-h** | **-V**]

描述

显示资源限制信息。

选项

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

默认输出

当前资源限制使用情况显示为以下字段:

LIMIT
资源限制的名称。

QUEUES 或 PER_QUEUE
cb.yaml 中定义的队列。

HOSTS 或 PER_HOST
cb.yaml 中定义的主机或主机组。

PROJECTS 或 PER_PROJECT
作业提交时定义的项目名（csub -P）。

USERS 或 PER_USER
cb.yaml 中定义的用户或用户组。

APPS 或 PER_APP
作业提交时定义的应用名（csub -A）。

expand
以上这些项的扩展值。如果定义中使用了用户组或主机组，则显示所有成员。如果组包含子组，则递归显示子组中的每个成员。如果消费者定义中使用了非运算符 (~)，则显示除指定主机/用户/队列/项目之外的所有成员。

**** CURRENT LIMIT RESOURCE USAGE****
此限制下项目的当前资源使用情况。

PROJECT
此限制下定义的项目已启动作业。

QUEUE
作业已在队列中启动。

USER
作业由用户提交。

HOSTHOST
作业正在主机上运行。

APP
作业定义的应用名

SLOTS
此项目/用户/主机/队列已使用的插槽数以及最多可使用的插槽数。

JOBS
此项目/用户/主机/队列已启动的作业数以及最多可使用的作业数。

示例

```
cresources

LIMIT  Limit1
HOSTS      : all ~grp1
    expand : host3 host4
USERS      : all ~AIP
    expand : u001 u002
SLOTS      : 0

LIMIT  Limit2
QUEUES     : normal
    expand : normal
PROJECTS   : Proj1 Proj2 Proj3
JOBS       : 10
CURRENT LIMIT RESOURCE USAGE:
    PROJECT     QUEUE       JOBS
    Proj1       normal      5/10
    Proj2       nornal      2/10

LIMIT  jenkinslimits
QUEUES     : normal owner
    expand : normal owner
PER_USER   : G1
    expand : u001 u002 u003 u004
SLOTS      : 20 09:00-17:00
SLOTS      : 10 17:00-09:00
CURRENT LIMIT RESOURCE USAGE:
    PROJECT       QUEUE        USER         HOST        APP       SLOTS       JOBS
    -             owner        u004         -           -         5/10        -
    -             normal       u004         -           -         5/10        -
```

另请参阅

cusers 、*cqueues* 、*chosts*

5.1.39 crestart

命令

crestart - 重启检查点作业

概要

crestart[*csub* 选项] [**-f**] *checkpoint_dir*[*job_ID* | "*job_ID*[*index*]"]

crestart [**-h** | **-V**]

选项列表

-B
-f
-N
-x
-b 开始时间
-C *core* 限制
-c [小时**:**] 分 [**/**主机名 |**/**主机模型]
-D 数据限制
-E "预执行命令 [参数…] "
-F 文件限制
-m "主机名 [**+**[优先级]] | 主机组 [**+**[优先级]] … "
-G 用户组
-M 内存限制
-q "队列名称… "
-S 堆栈限制
-t 终端时间
-w '依赖表达式 '
-W 运行时限 [**/**主机名 |**/**主机型号]| *checkpoint_dir*[*job_ID* | "*job_ID*[*index*]"]
[**-h** | **-V**]

描述

使用保存在 *checkpoint_dir/last_job_ID/* 中的检查点文件重新启动已进行检查点处理的作业。只有已成功进行检查点处理的作业才能重新启动。

作业将重新提交并分配新的作业 ID。检查点目录将使用新的作业 ID 重命名，即 *checkpoint_dir/new_job_ID/.*。

默认情况下，作业将使用与原始作业相同的输出文件和文件传输规范、作业名称、窗口信号值、检查点目录和周期以及重新运行选项重新启动。

要在另一台主机上重新启动作业，两台主机必须二进制兼容、运行相同的操作系统版本、有权访问可执行文件、有权访问所有打开的文件（AIP 必须使用绝对路径名定位它们）以及有权访问检查点目录。

作业重新启动时，环境变量 CB_RESTART 设置为 Y。

AIP 调用 CB_SERVERDIR (/opt/skyformai/sbin) 中的*erestart* 可执行文件来执行重新启动。

只有此处列出的 csub 选项可以与 crestart 一起使用。

选项

只有上面选项列表中列出的 csub 选项可以用于 crestart。除以下选项外，有关 crestart 选项的说明，请参阅*csub* 。

-f
即使存在不可重启的条件（这些条件特定于操作系统），也强制重启作业。

另请参阅

csub 、*cjobs* 、*cmod* 、*cqueues* 、*cchkpnt* 、*echkpnt* 、*erestart* 。

限制

在内核级检查点中，您无法使用 crestart 更改核心限制、CPU 限制、堆栈限制或内存限制的值。

5.1.40 cresume

命令

cresume - 恢复一个或多个已暂停的作业

概要

cresume [**-g** 作业组名称] [**-J** 作业名称] [**-m** 主机名][**-q** 队列名称] [**-u** 用户名 | **-u** 用户组 | **-u all**] [**0**]

cresume [作业 *ID* | “作业 *ID*[索引列表]”] …

cresume[**-h** | **-V**]

描述

发送 SIGCONT 信号以恢复一个或多个已暂停的作业。只有 root 和 AIP 管理员可以操作其他用户提交的作业。对非 PSUSP 或 USUSP 状态的作业使用 cresume 无效。

默认情况下，仅恢复一个作业，即您最近提交的作业。

您也可以使用 ckill -s 向作业发送恢复信号。

您无法恢复未暂停的作业。

选项

0
恢复所有满足其他选项（-m、-g、-q、-u 和 -J）的作业。

-g 作业组名称
仅恢复指定作业组中的作业。

-J 作业名称
仅恢复具有指定名称的作业。

-m 主机名
仅恢复已调度到指定主机的作业。

-q 队列名
仅恢复指定队列中的作业。

-u 用户名 | -u 用户组名 | -u all
仅恢复指定用户或组或所有用户的作业。

指定 all 为保留用户名。

***job_ID* ··· | "*job_ID*[*index_list*]" ···**
仅恢复指定的作业。任何用户提交的作业都可以在此处指定，而无需使用 -u 选项。

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 发行版本打印到 stderr 并退出。

示例

% **cresume -q night 0**

恢复用户在夜间队列中所有已暂停的作业。如果用户是 AIP 管理员，则恢复夜间队列中所有已暂停的作业。

另请参阅

cstop 、*ckill*

5.1.41 crmhost

命令

crmhost - 删除主机

概要

crmhost *hostname*

crmhost [**-h** | **-V**]

描述

从集群中删除单个主机，无需重新配置集群。

hostname 参数是必需的，并且必须列在集群的主机列表中。它会动态地从集群中删除指定的主机。参见*chinfo*。

选项

-h
: 将命令用法打印到 stderr 并退出。

-V
: 将 AIP 版本打印到 stderr 并退出。

hostname
: 要从集群中删除的单个主机的主机名。hostname 必须在集群中。

输出

显示以下内容：Host hostname removed.

在这种情况下，hostname 已成功从集群中移除。

否则，如果 host_name 不属于集群，则显示以下内容：main: invalid hostname host_name

5.1.42 crun

命令

crun - 强制作业立即运行

概要

crun [**-f**]**-m** "主机名… " *job_ID*

crun [**-f**]**-m** "主机名… ""*job_ID*[*index_list*]"

crun [**-h** | **-V**]

描述

> **警告:** 此命令仅供 AIP 管理员使用。

强制等待作业在指定主机上立即运行。

已强制运行的作业将被视为正在运行的作业，这可能违反用户、队列或主机的作业限制。

已强制运行的作业即使提交到可抢占队列，且其他作业也提交到抢占队列，也无法被其他作业抢占。

默认情况下，作业启动后，仍然受运行窗口和暂停条件的影响。

选项

-f
允许作业在不因运行窗口或暂停条件而暂停的情况下运行。

-m 主机名…
必需。指定要运行作业的一个或多个主机。

***job_ID*| "*job_ID*[*index_list*]"**
必需。指定要运行的作业，或指定作业数组中的一个元素。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

限制

您无法强制在 SSUSP、USUSP 或 PSUSP 状态下执行作业。

5.1.43 csadmin

管理员命令

csadmin - - AIP 调度系统 cbsched、cbjm、和 jservice 管理工具

概要

csadmin 子命令

csadmin[**-h** | **-V**]

子命令列表

ckconfig [**-v**]

reconfig [**-v**] [**-f**]

schedrestart [**-v**] [**-f**]

qopen [队列名称…| **all**]

qclose [队列名称…| **all**]

qact [队列名称…| **all**]

qinact [队列名称…| **all**]

qhist [**-t** *time0*,*time1*] [**-f** 日志文件名] [队列名…]

hopen [**-C** 消息] [主机名…| 主机组…| **all**]

hclose [**-C** 消息] [主机名…| 主机组…| **all**]

jmrestart [**-f**] [主机名…| **all**]

jmshutdown [**-f**] [主机名…| **all**]

jmstartup [**-f**] [主机名…| **all**]

jsrestart

jsreconfig

hhist [**-t** *time0*,*time1*] [**-f** 日志文件名] [主机名…]

schedhist[**-t** *time0*,*time1*] [**-f** 日志文件名]

hist [**-t** *time0*,*time1*] [**-f** 日志文件名]

help [命令…] | **?** [命令…]

quit

jmdebug [**-c** 类名…] [**-l** 调试级别] [**-f** 日志文件名] [**-o**] [主机名…]

scheddebug [**-c** 类名…] [**-l** 调试级别] [**-f** 日志文件名] [**-o**]

jmtime [**-l** 计时级别] [**-f** 日志文件名] [**-o**] [主机名…]

schedtime [**-l** 计时级别] [**-f** 日志文件名*]* [**-o**]

pwup [**-C** 消息] [**-f**] 主机名 [主机名…]

pwdown [**-C** 消息] [**-f**] 主机名 [主机名…]

pwrestart [**-C** *message*] [**-f**] *host_name* [*host_name* …]

描述

> **警告:** 此命令仅供 AIP 管理员使用。

csadmin 提供了一组用于控制和监控 AIP 的命令。如果没有为 csadmin 提供子命令，csadmin 会提示从标准输入中输入命令。

每个命令的相关信息可通过 help 命令获取。

csadmin 命令由一组特权命令和一组非特权命令组成。特权命令仅可由 root 或 AIP 管理员调用，具体定义在配置文件中（有关管理员信息，请参阅 cb.yaml(5)）。特权命令包括：

reconfig

schedrestart

qopen

qclose

qact

qinact

hopen

hclose

jmrestart

jmshutdown

jmstartup

jsrestart

pwup

pwdown

pwrestart

所有其他命令均为非特权命令，可由任何 AIP 用户调用。如果要由 AIP 管理员执行特权命令，则必须以 setuid root 身份安装 csadmin，因为它需要使用特权端口发送请求。

对于可以指定多个主机名或主机组的子命令，请勿将多个主机名或主机组括在引号中。

选项

子命令
执行指定的子命令。请参阅“用法”部分。

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 发行版本打印到 stderr 并退出。

用法

ckconfig [-v]
检查 AIP 配置文件。配置文件位于 CB_ENVDIR 目录中。

默认情况下，csadmin ckconfig 仅显示配置文件检查的结果。如果发现警告错误，csadmin 会提示您显示详细信息。

-v
详细模式。将配置文件检查的详细消息显示到 stderr。

reconfig [-v] [-f]

schedrestart [-v] [-f]
重新配置，重启 CBSCHED。检查配置文件中的错误，并将结果显示到 stderr。如果配置文件中未发现任何错误，则会向 CBSCHED 发送重新配置请求，并重新加载配置文件。

发出此命令后，CBSCHED 重新启动。

如果发现警告错误，csadmin 会提示您显示详细消息。如果发现致命错误，则不会执行重新启动，并且 csadmin 会退出。

-v
详细模式。显示有关配置文件状态的详细消息。如果不使用此选项，则默认显示配置文件检查的结果。所有配置文件检查的消息都会打印到标准错误输出 (stderr)。

-f
如果配置文件不包含致命错误，则禁用交互并继续重新启动。

qopen [*queue_name* ···| all]
打开指定的队列，如果指定了保留字 all，则打开所有队列。如果未指定队列，则默认使用系统默认队列（有关 default_queues，请参阅*cb.yaml* ）。队列只有在打开时才能接受作业。

qclose [*queue_name* ···| all]
关闭指定的队列，如果指定了保留字 all，则关闭所有队列。如果未指定队列，则默认使用系统默认队列。队列关闭后将不会接受任何作业。

qact[*queue_name* ···| all]
激活指定队列，如果指定了保留字 all，则激活所有队列。如果未指定队列，则默认使用系统默认队列。如果队列已激活，则队列中的作业可以被调度。被其运行窗口停用的队列无法通过此命令重新激活（有关 run_window，请参阅*cb.yaml* ）。

qinact [*queue_name* ···| all]
停用指定队列，如果指定了保留字 all，则停用所有队列。如果未指定队列，则默认使用系统默认队列。如果队列已停用，则队列中的作业无法调度。

qhist [-t *time0*,*time1*] [-f *logfile_name*] [*queue_name* ···]
显示指定队列的历史事件，如果未指定队列，则显示所有队列的历史事件。队列事件包括队列打开、关闭、激活和停用。

-t *time0*,*time1*
仅显示从 *time0* 到 *time1* 期间发生的事件。有关时间格式，请参阅*chist* 。默认显示事件日志文件中的所有队列事件（见下文）。

-f *logfile_name*
指定事件日志文件的文件名。可以指定绝对路径或相对路径。默认使用 AIP 系统当前使用的事件日志文件：/opt/skyformai/word/data/cb.events。选项 -f 对于离线分析非常有用。

hopen [-C *message*] [*host_name*···| *host_group*···| all]
打开服务器主机。指定任何服务器主机或主机组的名称（参见*cmgroup* ）。如果指定了保留字 all，则将打开所有服务器主机。如果未指定主机或主机组，则假定为本地主机。如果主机处于打开状态，则会接受作业。

选项 -C 将文本作为管理员消息记录到操作历史记录中。消息字符串的最大长度为 128 个字符。

hclose [-C *message*] [*host_name*···| *host_group*···| all]
关闭服务器主机。指定任何服务器主机或主机组的名称（参见*cmgroup* ）。如果指定了保留字 all，则所有服务器主机都将关闭。如果未指定任何参数，则默认使用本地主机。关闭的主机将不会接受任何新作业，但已调度到该主机的作业不会受到影响。注意，这与通过窗口关闭的主机不同，在这种情况下，该主机上的所有作业都将被暂停。

选项 -C 将文本作为管理员消息记录到操作历史记录中。消息字符串的最大长度为 128 个字符。

jmrestart [-f] [*host_name*…| all]
在指定主机上重新启动 CBJM，如果指定了保留字 all，则在所有服务器主机上重新启动。如果未指定主机，则默认使用本地主机。CBJM 将从头重新执行自身。这允许使用新的 CBJM 二进制文件。

该子命令向 CBJM 发送重启请求，由 CBJM 自身退出后重运行，只需 AIP 管理员权限即可。

-f
禁用交互，重启 CBJM 时无需确认。

jmshutdown [-f] [*host_name*…| all]
关闭指定主机上的 CBJM，如果指定了保留字 all，则关闭所有服务器主机上的 CBJM。如果未指定主机，则默认使用本地主机。CBJM 将在收到请求后退出。

该子命令向 CBJM 发送退出请求，由 CBJM 自身退出，只需 AIP 管理员权限即可。

-f
禁用关闭 CBJM 时无需确认。

jmstartup [-f] [*host_name*…| all]
在指定主机或所有服务器主机上启动 CBJM（如果指定了保留字 all）。只有 root 可以使用此选项，并且这些用户必须能够在所有 AIP 主机上使用 ssh。如果未指定主机，则假定使用本地主机。

-f
禁用交互，启动 CBJM 时无需确认。

jsrestart
如果 JSERVICE 正在运行，则重新启动 JSERVICE。如果 JSERVICE 未运行，则该命令失败。当在 $CB_ENVDIR/jservice.yaml 中启用 JSERVICE 时，在主服务器上重新启动 CBJM (csadmin jmrestart) 会冷启动 JSERVICE。

jsreconfig
让 JSERVICE 重读*jservice.yaml* 和*cb.yaml* 配置文件，更新参数。这个操作不改变 JSERVICE 内存里的作业数据。

hhist [-t *time0*,*time1*] [-f *logfile_name*] [*host_name*…]
显示指定主机的历史事件，如果未指定主机，则显示所有主机的历史事件。主机事件包括主机的开启和关闭。选项 -t 和 -f 与 qhist 的选项完全相同（见上文）。

schedhist [-t *time0*,*time1*] [-f *logfile_name*]
显示 CBSCHED 的历史事件。事件描述 CBSCHED 的启动和退出。选项 -t 和 -f 与 qhist 的选项完全相同（见上文）。

hist [-t *time0,time1*] [-f *logfile_name*]
显示所有队列、主机和 CBSCHED 的历史事件。选项 -t 和 -f 与 qhist 的选项完全相同（见上文）。

help [*command* …] | ? [*command* …]
显示指定命令的语法和功能。

quit
退出 csadmin 会话。

jmdebug [-c *class_name* …] [-l *debug_level*] [-f *logfile_name*] [-o] [*host_name* …]
设置 CBJM 的消息日志级别，以便在日志文件中包含更多信息。您必须是 root 或 AIP 管理员才能使用此命令。

如果该命令未使用任何选项，则使用以下默认值：

class_name = 0（不记录其他类）

debug_level = 0（参数 CB_LOG_MASK 中的 LOG_DEBUG 级别）

logfile_name = 当前 AIP 系统日志文件，位于 CB_LOGDIR 指定的目录中，格式为 *daemon_name*.log.*host_name*

host_name = 本地主机（提交命令的主机）

-c *class_name* ···

指定要记录调试消息的软件类。

class_name 的格式是类的名称，或类名称列表，以空格分隔并用引号括起来。

可能的类别：

LC_AUTH - 记录身份验证消息

LC_CHKPNT - 记录检查点消息

LC_COMM - 记录通信消息

LC_EXEC - 记录作业执行的重要步骤

LC_FILE - 记录文件传输消息

LC_HANG - 标记程序可能挂起的位置

LC_JLIMIT - 记录作业槽位限制消息

LC_LOADINDX - 记录负载索引消息

LC_PEND - 记录与作业等待原因相关的消息

LC_PERFM - 记录性能消息

LC_PIM - 记录 CBPS 消息

LC_SIGNAL - 记录与信号相关的消息

LC_SYS - 记录系统调用消息

LC_TRACE - 记录重要的程序执行步骤

LC_RPC - 记录 XDR 传输的所有内容

LC_SS - 与高级调度相关的日志消息

LC_POWER - 与电源操作相关的日志消息

备注： 类也列在 cube.h 中。

默认值：0（不记录其他类）

-l ***debug_level***

指定调试消息的详细级别。数字越高，记录的详细程度越高。较高级别包含所有较低级别。

可能的值：

0: LOG_DEBUG 级别。

1: LOG_DEBUG1 级别用于扩展日志记录。较高级别包含较低日志记录级别。
例如，LOG_DEBUG3 包含 LOG_DEBUG2、LOG_DEBUG1 和 LOG_DEBUG 级别。

2: LOG_DEBUG2 级别用于扩展日志记录。较高级别包含较低日志级别。例如，LOG_DEBUG3 包含 LOG_DEBUG2、LOG_DEBUG1 和 LOG_DEBUG 级别。

3: LOG_DEBUG3 级别用于扩展日志记录。较高级别包含较低日志级别。例如，LOG_DEBUG3 包含 LOG_DEBUG2、
LOG_DEBUG1 和 LOG_DEBUG 级别。

默认值：0（参数 CB_LOG_MASK 中的 LOG_DEBUG 级别）

-f *logfile_name*
指定要记录调试消息的文件名。可以指定带或不带完整路径的文件名。

如果指定了不带路径的文件名，则文件将保存在/opt/skyformai/log 目录中。

将要创建的文件的名称将采用以下格式：

logfile_name.daemon_name.host_name.log

如果指定的路径无效，则日志文件将在/tmp 目录中创建。

默认值：当前 AIP 系统日志文件，位于 CB_LOGDIR 指定的目录中，格式为 *daemon_name.host_name*.log。

-o
关闭临时调试设置并将其重置为守护进程的启动状态。消息日志级别将重置为 CB_LOG_MASK 的值，类别将重置为 CB_DEBUG_SCHED 和 CB_DEBUG_JM 的值。

日志文件也将重置为默认日志文件。

***host_name* …**
可选。在指定的一个或多个主机上设置调试设置。

主机名列表必须用空格分隔并用引号括起来。

默认值：本地主机（提交命令的主机）

scheddebug [-c *class_name* …] [-l *debug_level*] [-f *logfile_name*] [-o]
设置 CBSCHED 的消息日志级别，以便在日志文件中包含其他信息。您必须是 root 或 AIP 管理员才能使用此命令。

有关选项的说明，请参阅 jmdebug。

jmtime [-l *timing_level*] [-f *logfile_name*] [-o] [*host_name* …]
设置 CBJM 的计时级别，以便在日志文件中包含其他计时信息。您必须是 root 或 AIP 管理员才能使用此命令。

如果命令未使用任何选项，则使用以下默认值：

timing_level = 不记录计时信息

logfile_name = 当前 AIP 系统日志文件，位于 CB_LOGDIR 指定的目录中，格式为：

daemon_name.log.*host_name*

host_name= 本地主机（提交命令的主机）

-l *timing_level*
指定日志文件中包含的计时信息的详细信息。计时信息指示软件中函数的执行时间，并以毫秒为单位记录。

有效值：1 | 2 | 3 | 4 | 5

数字越大，软件中计时并记录执行时间的函数越多。数字越小，包含更常用的软件函数。较高级别包含所有较低级别。

默认值：未定义（不记录计时信息）

-f *logfile_name*
指定要记录计时消息的文件名。可以指定带或不带完整路径的文件名。

如果指定了不带路径的文件名，则文件将保存在 cb.conf 中 CB_LOGDIR 参数指定的目录中。

将要创建的文件名将采用以下格式：*logfile_name.daemon_name.host_name*.log

如果指定的路径无效，在 Linux 系统中，日志文件将创建在/tmp 目录中。

备注： 计时消息和调试消息都记录在同一个文件中。

默认值：当前 AIP 系统日志文件，位于 CB_LOGDIR 指定的目录中，格式为 *daemon_name.host_name*.log。

-o

可选。关闭临时计时设置并将其重置为守护进程的启动状态。计时级别将重置为相应守护进程的参数值（CB_TIME_SCHED、CB_TIME_JM）。

日志文件也将重置为默认日志文件。

host_name…

在指定的一个或多个主机上设置计时级别。

主机列表必须用空格分隔，并用引号括起来。

默认值：本地主机（提交命令的主机）

schedtime [-l *timing_level*] [-f *logfile_name*] [-o]

设置 CBSCHED 的计时级别，以便在日志文件中包含额外的计时信息。您必须是 root 或 AIP 管理员才能使用此命令。

有关选项的说明，请参阅 jmtime。

pwdown [-C *message*] [-f] *host_name* [*host_name …*]

使用*cb.yaml* 中配置的 power_down_cmd 命令关闭主机，并在主机上发布可选消息 (-C)（chosts -l 可以显示该消息）。

选项 -f 禁用用于在每个主机上确认操作的交互模式。

pwup [-C *message*] [-f] *host_name* [*host_name …*]

使用在 cb.yaml 中配置的 power_up_cmd 命令启动主机，并在主机上发布可选消息 (-C)（chosts -l 可以显示该消息）。只有通过 csadmin 命令或电源调度策略关闭的主机才能通过此命令恢复。

选项 -f 禁用交互模式，用于在每个主机上确认操作。

pwrestart [-C *message*] [-f] *host_name* [*host_name …*]

执行 cb.yaml 中配置的 power_restart_cmd 命令，并指定要在主机上发布的可选消息 (-C)（chosts -l 可以显示该消息）。

选项 -f 禁用交互模式，用于在每个主机上确认操作。

5.1.44 cstop

命令

cstop - 暂停未完成的作业

概要

cstop [**-a**] [**-d**] [**-R**] [**0**] [**-g** 作业组名称] [**-J** 作业名称] [**-m** 主机名 | **-m** 主机组] [**-q** 队列名称] [**-u** 用户名 | **-u** 用户组 | **-u all**] [作业 *ID* | "作业 *ID*[索引]"] ···

cstop | [**-h** | | -V**]

描述

暂停未完成的作业。

向顺序作业发送 SIGSTOP 信号，向并行作业发送 SIGTSTP 信号以暂停它们。

默认情况下，只暂停一个作业，即您最近提交的作业。

只有 root 和 AIP 管理员可以操作其他用户提交的作业。

对处于 USUSP 状态的作业使用 cstop 无效。

您还可以使用 ckill -s 向作业发送暂停信号，或发送恢复信号。您可以使用 cresume 恢复已暂停的作业。您无法暂停已暂停的作业。

如果信号请求未能到达作业执行主机，AIP 将在主机恢复连接后重试该操作。AIP 会重试最近的信号请求。

选项

-a
: 暂停所有作业。

-d
: 仅暂停已完成的作业（状态为 DONE 或 EXIT）。

0
: 暂停所有满足其他选项（-m、-g、-q、-u 和 -J）的作业。

-g 作业组名称
: 仅暂停指定作业组中的作业。

-J 作业名称
: 仅暂停指定名称的作业。

-m 主机名 | -m 主机组
: 仅暂停调度到指定主机或主机组的作业。

-q 队列名称
: 仅暂停指定队列中的作业。

-u 用户名 | -u 用户组 | -u all
: 仅暂停指定用户或用户组拥有的作业，如果指定了关键字 all，则暂停所有用户拥有的作业。

作业 *ID* ···| "作业 *ID*[索引]" ···
: 仅暂停指定的作业。任何用户提交的作业都可以在此处指定，而无需使用 -u 选项。

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到标准错误输出并退出。

示例

% **cstop 314**
暂停编号为 314 的作业。

% **cstop -m apple**
暂停调用者发送到主机 apple 的最后一个作业。

% **cstop -u smith 0**
暂停用户 smith 提交的所有作业。

% **cstop -u all**
暂停 AIP 系统中最后一个提交的作业。

% **cstop -u all 0**
暂停 AIP 系统中所有作业。

另请参见

csub、*cjobs*、*cqueues*、*chosts*、*cresume*、*ckill*

5.1.45 csub

命令

csub-向 AIP 提交作业

概要

csub [选项] 命令 [参数]

csub [**-h**| **-V**]

选项列表

-A 应用名

-a *esub* 参数

-app 配置文件名称

-B

-b [[月:] 日:] 小时: 分钟

-C *core* 限制

-c [小时:] 分钟 [/*host_name*|/*host_model*]

-cwd 作业工作目录

-cwdc 作业工作目录

-D 数据限制

-di *Docker* 镜像名

-do *Docker run* 参数

-dl

-e | -eo *err_file*

-E “*pre_exec_command* [参数…]”

-ENV | -env “none” | “~*var_name*[,~*var_name*]…,][*var_name*= 值 [,*var_name*= 值]…”

-f “本地文件 *op* [远程文件]” …

-F 文件大小限制

-G 用户组名

-g 作业组名

-H

-I | -Ip | -Is | -K

-i 输入文件 | **-is** 输入文件

-J 作业名 | **-J** “作业名称 [索引列表]% 作业限制 “

-Jd “作业描述”

-k “检查点目录 [检查点周期] [方法 = 方法名称]”

-L 登录 *shell*

-Lp

-M 内存限制

-m “主机名 [+[首选项级别]] | 主机组 [+[首选项级别]] …”

-N

-n 最小处理器数 [, 最大处理器数]

-o | -oo 输出文件

-P 项目名称

-p 进程限制

-pack 作业提交文件

-proxy

-q “队列名称…”

-R “资源请求 &” *[*-R** “资源请求” …]

-r

-rnc *resize_nodify_command*

-S *stack* 限制

-Si *Singularity* 镜像文件

-So *Singularuty* 选项

-Sb *Singularity_bind*

-sp 优先级

-t [[月:] 日:] 时: 分钟

-U 消息模板

-u 邮件用户

-v *swap_limit*

-W [小时:] 分钟 [/主机名 |/主机模型]

-w ‘依赖关系表达式‘

-XF

-x

-Zs

描述

提交执行的作业并为其分配唯一的数字作业 ID。

当作业、主机、队列和集群的所有条件均满足时，在满足作业所有要求的主机上运行作业。如果 AIP 无法立即运行所有作业，则 AIP 调度策略将决定调度顺序。作业将根据当前系统负载启动和暂停。

设置用户执行作业的环境，包括当前工作目录、文件创建掩码和所有环境变量，并在启动作业之前设置 AIP 环境变量。

作业运行时，命令行和 stdout/stderr 缓冲区存储在执行主机上的 home_directory/.cbsched 目录中。如果此目录无法访问，则使用 /tmp/.cbtmpuser_ID 作为作业的主目录。如果当前工作目录位于提交主机的主目录下，则当前工作目录也会设置为执行主机主目录下的相同相对目录。如果当前工作目录在执行主机上无法访问，则作业将在 /tmp 中运行。

如果 csub 没有提供命令，csub 会提示从标准输入中输入命令。在 LINUX 系统中，在新行输入 CTRL-D 即可终止输入。

默认情况下，AIP 假定集群中所有主机都拥有统一的用户名和用户 ID 空间。也就是说，给定用户提交的作业将在执行主机上使用同一用户的帐户运行。对于用户名和用户 ID 空间不一致的情况，必须使用帐户映射来确定用于运行作业的帐户。

默认情况下，使用命令名称作为作业名称。引号有意义。

默认情况下，该作业不可进行检查点操作。

默认情况下，自动选择合适的队列。如果您通过设置 CB_DEFAULTQUEUE 定义了默认队列列表，则会从您的列表中选择该队列。如果未定义 CB_DEFAULTQUEUE，则会从 AIP 管理员指定的系统默认队列列表中选择队列（请参阅*cb.yaml*中的 default_queues 参数）。

默认情况下，假定仅请求一个处理器。

默认情况下，作业不会启动登录 shell，而是在提交作业的执行环境下运行作业文件。

默认情况下，作业的输入文件是/dev/null（无输入）。

默认情况下，作业完成后会不向您发送邮件。若发邮件，目的地由 cb.yaml 中的 mailto 定义。邮件内容包括作业报告、作业输出（如有）以及错误消息（如有）。

默认情况下，将作业计入默认项目。默认项目是通过设置环境变量 CB_DEFAULTPROJECT 定义的项目。

使用**-n** 提交并行作业。

使用**-I**、**-Is** 或**-Ip** 提交交互式作业。

使用**-J** 为您的作业分配一个名称。

使用**-k** 指定可检查点的作业。

要终止使用 **csub** 提交的作业，请使用 **ckill**。

使用 **cmod** 修改通过 **csub** 提交的作业。cmod 使用的选项与 csub 类似。

选项

-A 应用程序名称
使用 * 应用程序名称 * 标记作业以便计费。该名称不会像 -app 选项那样通过 $CB_ENVDIR/cb.apps 或 $HOME/.cube/cb.apps 进行验证。

例子:

```
csub-A fluent myfluent_job
```

-app 配置名称
将指定的作业 * 配置名称 * 分配给作业。对于作业阵列，将相同的配置文件名称分配给作业阵列的所有元素。作业配置文件必须在 $CB_ENVDIR/cb.apps 和/或 $HOME/.cube/cb.apps 中配置。

-B
当作业被分派并开始执行时向您发送邮件。

-H
提交作业时，将其保持在 PSUSP 状态。除非您指示系统恢复该作业（参见*cresume*），否则该作业不会被调度。

-I | -Ip | -Is
提交交互式作业。交互式作业完成或终止后才能提交新作业。

将作业的标准输出（或标准错误）发送到终端。除非指定 **-N** 选项，否则作业完成后不会发送邮件。

交互式作业可以支持 tty 终端。

指定**-Ip** 选项时，提交交互式作业，并在作业启动时创建伪终端。某些应用程序（例如 **vi**）需要伪终端才能正常运行。

指定**-Is** 选项后，将提交交互式作业，并在作业启动时创建一个支持 Shell 模式的伪终端。提交交互式 Shell 或重新定义 CTRL-C 和 CTRL-Z 键的应用程序（例如 jove）时应指定此选项。

如果指定了**-i***input_file* 选项，则您无法通过终端与作业的标准输入进行交互。

如果指定了**-o***out_file* 选项，则将作业的标准输出发送到指定的输出文件。如果指定了**-e***err_file* 选项，则将作业的标准错误发送到指定的错误文件。

不能将**-I**、**-Ip** 或**-Is** 与**-K** 选项一起使用。

例子:

```
csub-Is bash
```

-K
提交作业并等待作业完成。提交作业时，向终端发送“等待调度”消息。作业完成后，向终端发送“作业已完成”消息。

在作业完成之前，您将无法提交其他作业。当需要完成作业才能继续执行（例如作业脚本）时，此功能非常有用。如果由于瞬时故障而需要重新运行作业，则 csub 会在作业成功完成后返回。csub 将以与

作业相同的退出代码退出，以便作业脚本能够根据退出代码采取适当的操作。如果作业在等待期间终止，则 csub 将以值 126 退出。

不能将**-K** 选项与**-I**、**-Ip** 或**-Is** 选项一起使用。

-N

作业完成后，通过邮件向您发送作业报告。当不带任何其他选项使用时，其行为与默认行为相同。

仅与不发送邮件的 -o、-I、-Ip 和-Is 选项一起使用，以强制 AIP 在作业完成时向您发送邮件消息。

-r

如果作业运行时执行主机不可用，则指定该作业将在另一台主机上重新运行。AIP 会将作业重新排入同一作业队列，并使用相同的作业 ID。找到可用的执行主机后，会重新运行该作业，就像提交了新作业一样。您会收到一封邮件，告知您主机故障以及作业已重新排入队列。

如果作业运行时系统出现故障，则指定系统重新启动时该作业将重新排队。

如果执行主机或系统出现故障，则重新运行作业；如果作业本身失败，则不会重新运行作业。

如果在作业执行检查点操作后执行主机不可用（参见 **csub -k** 和*cchkpnt* 命令），则该作业将从上一个检查点重新启动。重新启动的作业将重新排队等待执行，其方式与使用*crestart* 命令重新启动作业的方式相同。为了成功重新启动作业，作业的检查点目录必须位于接收重新启动作业的主机可访问的共享文件系统中。

-x

将运行作业的主机置于独占执行模式。

在独占执行模式下，您的作业会在主机上自行运行。它只会被调度到没有其他作业正在运行的主机上，并且 AIP 不会在该作业完成之前将任何其他作业发送到该主机。

要以独占执行模式提交作业，必须将队列配置为允许独占作业。

当作业调度时，*chosts*报告主机状态为 closed_Excl，*cload*报告主机状态为 lockU。

-a *esub_* 参数

该参数以 CB_SUB_ADDITIONAL 的 形式存储在作业的参数文件（即 CB_SUB_PARM_FILE ）中。该文件可以被 esub 读取。

-b [[月:] 日:] 时: 分

在指定的日期和时间或之后调度执行作业。日期和时间的格式为 [[月:] 日:] 时: 分，其中数字范围如下: 月 1-12，日 1-31，时 0-23，分 0-59。

必须至少指定两个字段。这些字段的格式假定为“小时: 分钟”。如果指定三个字段，则假定为“日: 小时: 分钟”，如果指定四个字段，则假定为“月: 日: 小时: 分钟”。

-C *core* 限制

为属于此作业的所有进程设置每个进程的（软）核心文件大小限制（请参阅 getrlimit(2)）。核心文件大小限制以 KB 为单位。

此选项的行为取决于特定于平台的 LINUX 系统。

在某些情况下，如果作业尝试创建大于指定限制的核心文件，则会向进程发送 SIGXFSZ 信号。SIGXFSZ 信号通常会终止该进程。

在其他情况下，核心文件的写入会在指定的限制处终止。

-c [小时:] 分钟 [/主机名 |/主机型号]

限制作业可使用的总 CPU 时间。此选项有助于防止作业失控或占用过多资源。当整个作业的总 CPU 时间达到限制时，会首先向该作业发送 SIGXCPU 信号，然后发送 SIGINT、SIGTERM 和 SIGKILL 信号。

如果 cb.yaml 中的 CB_JOB_CPULIMIT 设置为 n，则会禁用 AIP 强制执行的 CPU 限制，并且 AIP 会将限制传递给操作系统。当作业中的一个进程超出 CPU 限制时，操作系统将强制执行该限制。

CPU 限制的格式为 [小时:] 分钟。分钟数可以指定为大于 59 的数字。例如，三个半小时可以指定为 3:30 或 210。

或者，您可以提供 AIP 中定义的主机名或主机型号名称。必须在 CPU 限制与主机名或型号名称之间插入 "/"。(请参阅*cinfo* 获取主机型号信息。) 如果未提供主机名或型号名称，AIP 将使用队列级别定义的默认 CPU 时间标准化主机 (cb.yaml 中的 DEFAULT_HOST_SPEC) (如果已配置)；否则，使用集群级别定义的默认 CPU 时间标准化主机 (cb.yaml 中的 DEFAULT_HOST_SPEC) (如果已配置)；否则，使用提交主机。

您指定的 CPU 时间是标准化 CPU 时间。这样做的目的是，即使作业发送到 CPU 速度更快或更慢的主机，在给定的 CPU 限制下，其处理量也大致相同。每当指定标准化 CPU 时间时，执行主机上的实际时间等于指定时间乘以标准化主机的 CPU 因子，然后除以执行主机的 CPU 因子。

-cwd 工作目录
指定作业执行的当前工作目录。如果未指定该选项，AIP 将使用运行 csub 的当前工作目录来执行作业。如果作业执行主机上的用户无法访问指定的当前工作目录，则将使用 /tmp 作为当前工作目录。

如果指定的目录不是绝对路径，AIP 将使用用户的主目录 $HOME/job_working_directory 作为作业执行的当前工作目录。

目录路径可以包含以下动态模式，这些模式区分大小写:

* %J - 作业 ID

* %I - 索引 (默认值为 0)

* %P - 项目名称 (默认值为 "default")

* %U - 用户名

* %G - 用户组 (默认值与用户名相同)

* %H - 第一个执行主机名

* %A - csub -Al-app 中指定的应用程序名称 (默认为 "_")

* %D - csub -Jd 中指定的职位描述 (默认为 "_")

-cwdc 工作目录
与-cwd 相同，除非目录不存在，否则 AIP 将尝试以 0700 模式以作业执行用户身份创建目录 (只有用户具有该目录的读/写权限)。

如果目录不存在且由于任何原因无法创建 (例如用户没有创建权限)，则将使用/tmp。

例子:

```
csub -cwdc /home/u001/%J myjob
```

-D 数据限制
为作业中的每个进程设置每个进程的 (软) 数据段大小限制 (参见 getrlimit(2))。数据限制以 KB 为单位。调用 **sbrk** 函数将数据段扩展到超出数据限制的大小将返回错误。

-di *Docker* 镜像名
指定用于运行作业的 Docker 镜像名。这相当于设置环境变量 CB_DOCKER_IMAGE。如果设置了环境变量 CB_DOCKER_IMAGE，则此选项将覆盖环境变量的值。

使用此选项，作业命令将在 Docker 容器内运行。如果容器不执行任何命令，则必须将字符串 "null" 作为作业命令，这告诉调度程序仅将容器作为作业运行，而不执行任何命令。

默认情况下，Docker 容器出于性能考虑使用主机网络。对于 AI 影响作业，当您指定 `-do "-p port1:port2"` 时，容器将使用 Docker 桥接网络。您也可以指定 `-do "--network=xxx"` 来更改容器使用的网络。

出于安全原因，AIP 容器作业将以提交作业的用户身份运行。您无法使用 `-do "--user"` 更改此设置。

AIP 添加的主要 “docker run” 参数是：`--name=DOCKERNAME --init -i --rm -e CB_ALLOCATION -e CB_JOBID -e CB_SUBCWD -e CB_PORT -e CB_HOSTSIP`。根据 CPU 和 GPU 的分配情况，还会自动添加 `--cpu` 和 `--gpus` 或 `--device`。

-do *Docker_run_options*

指定命令 “docker run” 的选项。这相当于设置环境变量 CB_DOCKER_OPTIONS。如果设置了环境变量 CB_DOCKER_OPTIONS，则此选项将覆盖环境变量的值。

由于已被 AIP 使用，因此无法使用以下选项：“ `--name`、`--cpu`、`--gpus` 或 `--device` ”。

docker run 选项 -u 或 `--user` 将被忽略，因为在 AIP 中，容器始终以与提交作业的用户相同的 uid:gid 运行。

当你从 Docker 桥接网络向主机网络进行端口发布时，可以使用保留字 CB_PORT 来指示 AIP 为该作业分配的端口。例如：

AIP 分配的端口为 16331，您希望将容器端口 8088 绑定到该端口。选项为：“-do -p CB_PORT:8088”。如果您要发布多个端口，可以指定 CB_PORT+n。例如：“-do -p CB_PORT:8088 -p CB_PORT+300:8081”。这会将容器端口 8088 发布到主机端口 16331，并将容器端口 8081 发布到主机端口 16631。

-dl

在执行 docker run 命令前，先运行 docker login。如果设置了环境变量 CB_DOCKER_USERNAME 和 CB_DOCKER_PASSWORD，则使用环境变量值进行认证。如果未设置环境变量，则使用 cpasswd 命令注册的用户名和密码进行认证。

-e | -eo 错误文件

指定文件路径。如果指定了-e 选项，则将作业的标准错误输出附加到指定文件。如果指定了 -eo 选项，则该文件将被覆盖。

如果作业启动后无法在执行主机上访问当前工作目录，AIP 会将标准错误输出文件写入 /tmp/。

文件路径可以包含以下动态模式，这些模式区分大小写：

* %J - 作业 ID

* %I - 索引（默认值为 0）

* %P - 项目名称（默认值为 “default”）

* %U - 用户名

* %G - 用户组（默认值与用户名相同）

* %H - 第一个执行主机名

* %A - csub -Al-app 中指定的应用程序名称（默认为 “_”）

* %D - csub -Jd 中指定的职位描述（默认为 “_”）

例子:

```
csub-e %J-%I-%U.err
```

-E “*pre_exec_command* [参数…]”

在作业实际运行之前，先在作业的执行主机上运行指定的 pre-exec 命令。对于并行作业，pre-exec 命令会在为其选定的第一个主机上运行。如果 pre-exec 命令返回 0（零），则实际作业将在选定的主机上启动。否则，作业（包括 pre-exec 命令）将返回 “等待” 状态并重新安排。

如果您的作业重新进入 “等待” 状态，AIP 会在条件允许的情况下继续尝试运行预执行命令和实际作业。因此，请确保您的预执行命令可以多次运行而不会产生副作用。

pre-exec 命令的标准输入和输出指向与实际作业相同的文件。pre-exec 命令与实际作业在相同的用户 ID、环境、主目录和工作目录下运行。如果 pre-exec 命令不在用户的正常执行路径（$PATH 变量）中，则必须指定该命令的完整路径名。

例子:

```
csub-E  “/home/u001/my_preexec.sh” myjob
```

-ENV | -env “none” |” *~var_name*[,~*var_name*]…,][*var_name*= 值 [,*var_name*= 值]…”
为作业设置环境变量。以下多个短语可以用逗号 (,) 分隔。请注意 Shell 对特殊字符的解释，这可能会影响传递给 csub 命令的字符串的实际值。使用单引号 (’) 引用整个参数。

“none” 表示从提交环境中移除所有现有的环境变量。默认情况下，csub 会将作业提交时设置的所有当前环境变量复制到作业执行环境中。

~var_name 表示取消设置该作业的 环境变量 var_name 。

var_name=value 表示设置或更新环境变量。

例子:

```
csub -ENV "~XDG_RUNTIME_DIR,~XDG_SESSION_ID,MYPROJNAME=rndproject" myjob
```

-f “本地文件 操作 [远程文件]” …
在本地（提交）主机和远程（执行）主机之间复制文件。请指定绝对路径或相对路径，包括文件名。在非共享系统中运行时，应将远程文件指定为不带路径的文件名。

如果未指定远程文件，则默认为本地文件（必须指定）。使用多个**-f** 选项可以指定多个文件。

操作

指定是否将文件复制到远程主机或从远程主机复制回的操作符。该操作符必须用空格括起来。

下面对运算符进行描述:

> 在作业开始前将本地文件复制到远程文件。如果远程文件存在，则覆盖它。

< 作业完成后，将远程文件复制到本地文件。如果本地文件已存在，则覆盖。

« 作业完成后，将远程文件附加到本地文件。本地文件必须存在。

>< 在作业开始前将本地文件复制到远程文件。如果远程文件存在，则覆盖它。然后在作业完成后将远程文件复制到本地文件。覆盖本地文件。

<> 在作业开始前将本地文件复制到远程文件。如果远程文件存在，则覆盖远程文件。作业完成后，将远程文件复制到本地文件。覆盖本地文件。

如果您使用 **-i** *input_file* 选项，则无需使用 **-f** 选项将指定的输入文件复制到执行主机。AIP 会为您执行此操作，并在作业完成后从执行主机中删除输入文件。

如果您使用 **-e** *err_file* 或**-o** *out_file* 选项，并且希望在作业完成时将指定的文件复制回提交主机，则必须使用**-f** 选项。

本地文件名和远程文件名都可以包含动态模式（%J、%I、%P、%U、%G、%H、%A、%D），其规范与-cwd 选项中描述的相同。

如果提交主机和执行主机的目录结构不同，则必须确保放置远程文件和本地文件的目录存在。

如果本地和远程主机的文件名空间不同，则必须始终指定相对路径名。如果本地和远程主机不共享同一文件系统，则必须确保包含远程文件的目录存在。建议在异构文件系统中运行时，仅指定远程文件的文件名。这会将文件放置在作业的当前工作目录中。如果文件在提交主机和执行主机之间共享，则不会执行文件复制。

AIP 使用 **crcp** 传输文件（参见*crcp* 命令）。crcp 会联系远程主机上的 CBEXE 执行文件传输。如果 CBEXE 不可用，则使用 **rcp** （参见 rcp(1)）。用户必须确保 rcp 二进制文件位于执行主机上用户的 $PATH 中。

仅当允许 **rcp** 时，从 AIP 客户端主机提交的作业才应指定-f 选项。同样，如果使用帐户映射，也必须允许 **rcp** 。

例子:

```
csub-cwdc /home/u001/%J -f "/home/u001/myinput > /home/u001/%J" myjob
```

把整个本地 myinput 目录传到作业运行主机上的%J (作业号) 的目录中，-cwdc 参数会动态生成%J 目录。

-F 文件限制

为作业中的每个进程设置每个进程的（软）文件大小限制（参见 getrlimit(2)）。文件大小限制以 KB 为单位。如果作业进程尝试写入超出文件大小限制的文件，则会向该进程发送 SIGXFSZ 信号。SIGXFSZ 信号通常会终止该进程。

-i 输入文件 | -is 输入文件

从指定文件获取作业的标准输入。指定绝对路径或相对路径。输入文件可以是任何类型的文件，但通常是 Shell 脚本文本文件。

如果执行主机上存在该文件，AIP 将使用它。否则，AIP 将尝试将文件从提交主机复制到执行主机。为了成功复制文件，您必须允许远程复制 (rcp) 访问，或者必须从运行 CBEXE 的服务器主机提交作业。该文件将从提交主机复制到 JOB_SPOOL_DIR 参数指定的目录中的临时文件，或复制到执行主机上的 $HOME/.cbsched 目录中。作业完成后，AIP 会删除此文件。

-is 选项会将输入文件假脱机到 cb.yaml 中 JOB_SPOOL_DIR 参数指定的目录中，并将假脱机文件用作作业的输入文件。默认情况下，如果未指定 JOB_SPOOL_DIR，则输入文件将假脱机到 CB_SHAREDIR/cb_indir。如果 cb_indir 目录不存在，AIP 会在假脱机文件之前创建该目录。作业完成后，AIP 会删除假脱机文件。如果您需要在作业完成之前修改或删除输入文件，请使用-is 选项。删除或修改原始输入文件不会影响已提交的作业。

除非使用**-is**，否则您可以在输入文件的名称中使用以下模式:

* %J - 作业 ID

* %I - 索引（默认值为 0）

* %P - 项目名称（默认值为“default”）

* %U - 用户名

* %G - 用户组（默认值与用户名相同）

* %H - 第一个执行主机名

* %A - csub -Al-app 中指定的应用程序名称（默认为“_”）

* %D - csub -Jd 中指定的职位描述（默认为“_”）

-g 作业组名称

提交 *作业组名称*指定的作业组中的作业。提交作业前，该作业组不必存在。

例如:

```
csub-g /reg/hug/myg1 job
Job101 has been submitted to the default queue [medium]
```

将作业提交到作业组/reg/hug/myg1。

如果组/reg/hug/myg1 存在，则作业 101 将附加到该作业组。

作业组名称最多可包含 512 个字符。

-J 作业名称 | -J “作业名称 [索引列表]% 作业槽位限制 “

为作业分配指定的名称，并且对于作业阵列，指定作业阵列的索引以及可选地指定在任何给定时间可以运行的最大作业数。

作业名称不必是唯一的。

要指定作业阵列，请将索引列表括在方括号中，并将整个作业阵列规范括在引号中。索引列表是一个逗号分隔的列表，其元素的语法为 start[-end[: step]]，其中 start、end 和 step 均为正整数。如果省略 step，则假定 step 为 1。作业阵列索引从 1 开始。默认情况下，最大作业阵列索引为 2.00。

您还可以使用正整数来指定此作业阵列的系统范围作业槽限制（在任何给定时间可以运行的最大作业数）。

阵列中的所有作业共享相同的作业 ID 和参数。数组的每个元素都通过其数组索引来区分。

提交作业后，您可以使用作业名称来标识该作业。指定 " job_ID [index]" 可 处理特定数组的元素。指定 " job_name [index]" 可处理所有同名阵列的元素。由于作业名称不唯一，因此多个作业阵列可能具有相同的名称，但索引集不同或相同。

例子:

```
csub -J "a[1-1000,2]%100" myjob
```

提交一个 500 单元的阵列作业，阵列索引为 1，3，5，…, 999。同一时间不能超过 100 个单元运行。

-Jd "作业描述"
为作业分配指定的描述。对于作业阵列，为作业阵列的所有元素分配相同的作业描述。

-k "*checkpoint_dir*[*checkpoint_period*][method=*method_name*]"
使作业可进行检查点操作并指定检查点目录。如果省略检查点句点，则无需使用引号。请指定相对路径名或绝对路径名。

当作业执行检查点操作时，检查点信息存储在 *checkpoint_dir* / job_ID / file_name 中。多个作业可以对同一目录执行检查点操作。系统可以创建多个文件。

检查点目录用于重新启动作业（参见*crestart*）。

可选地，指定检查点周期（以分钟为单位）。请指定一个正整数。正在运行的作业会在每个检查点周期自动执行检查点操作。可以使用*cchkpnt* 更改检查点周期。由于检查点操作是一项重量级操作，因此您应该选择大于半小时的检查点周期。

可选地，指定用于作业的自定义检查点和重启方法。使用 **method=default** 表示使用 AIP 的默认检查点和重启程序 echkpnt.default 和 erestart.default，这两个程序通常与特定的检查点/重启方法相关联。

echkpnt.method_name 和 erestart.method_name 程序必须位于 CB_ENVDIR/../sbin 或 CB_ECHKPNT_METHOD_DIR 指定的目录中（环境变量或在 cb.yaml 中设置）。

如果已经使用 CB_ECHKPNT_METHOD（环境变量或 cb.yaml）指定了自定义检查点和重启方法，则使用 csub -k 指定的方法将覆盖该方法。

AIP 会调用 CB_ENVDIR/../sbin 中的 **echkpnt**（参见*echkpnt*）来对作业进行检查点操作。您可以通过将 CB_ECHKPNT_METHOD 和 CB_ECHKPNT_METHOD_DIR 定义为环境变量或在 cb.yaml 中将其指向您自己的 echkpnt 来覆盖作业的默认 echkpnt。这允许您使用其他检查点工具，包括应用程序级检查点。

-L *login_shell*
使用指定的登录 shell 初始化执行环境。指定的登录 shell 必须是绝对路径。这不一定是执行作业的 shell。使用这个参数后所有提交作业本地的环境变量都不会带到作业运行环境中。

例子:

```
csub -L /bin/bash myjob
```

-Lp
当 cb.yaml 中指定参数 "additional_ports" 来限制 AIP 使用的随机端口时，此参数会强制交互式作业 (-I/-Ip/-Is) 使用 "additional_ports" 范围内的端口。这是为了让交互式作业在作业提交主机和作业执行主机之间存在防火墙的情况下，仍然能够正常工作，并且只有 "additional_ports" 中指定的端口开放。

默认情况下，即使在 cb.yaml 中指定了 "additional_ports"，交互式作业也会使用随机端口。

例子:

```
csub -Lp -I hostname
```

-m "主机名 [+[首选项级别]] | 主机组 [+[首选项级别]] …"
在指定的主机之一上运行作业。

默认情况下，如果有多个主机可供候选，则在负载最低的主机上运行作业。要更改此设置，请在您希望使用的主机或主机组名称后添加加号 (+)，并可选地在其后添加优先级。对于优先级，请指定一个正整数，数字越大，表示这些主机的优先级越高。

例如，-m "hostA groupB+2 hostC+1" 表示 groupB 是最受欢迎的，而 hostA 是最不受欢迎的。

有关主机组的信息，请使用 **cmgroup**。

关键字 others 可以指定优先级，也可以不指定优先级，以引用未列出的其他主机。关键字 others 必须至少与一个主机名或主机组一起指定，不能单独指定。例如，-m "hostA+ others" 表示 hostA 优先于所有其他主机。

如果同时使用**-m** *host_name*[+[*pref_level*]] | *host_group*[+[*pref_level*]]…" 选项和 **-q** *queue_name* 选项，则必须将指定的队列配置为包含主机列表中的所有主机。否则，作业将无法提交。要查看为队列配置了哪些主机，请使用 cqueues -l。

-M 内存限制
指定内存限制（以 KB 为单位）。以 "M" 结尾表示限制以 MB 为单位。以 "G" 结尾表示限制以 GB 为单位。

例子:

```
csub -M 100G myjob
```

-n 最小作业槽 [, 最大作业槽]
提交并行作业，并指定运行该作业所需的最小和最大处理器数量（某些处理器可能位于同一台多处理器主机上）。如果未指定最大值，则您指定的数字代表要使用的处理器的确切数量。

如果最大处理器数量大于作业提交到的队列的进程限制，AIP 将拒绝该作业（请参阅*cb.yaml*中的 processlimit 参数）。

一旦至少有最低数量的处理器可用，作业就会被调度到第一个选定的主机。作业所选的主机名列表在环境变量 CB_HOSTS 和 CB_MCPU_HOSTS 中指定。作业本身应该在这些主机上启动并行组件，并在它们之间建立通信，可选地使用 CBEXE。

-o | -oo 输出文件
指定文件路径。如果指定了-o 选项，则将作业的标准输出附加到指定文件。如果指定了 -oo 选项，则覆盖该文件。如果文件不存在或系统无法写入，则通过邮件发送输出。

如果仅指定文件名，AIP 会将输出文件写入当前工作目录。如果作业启动后在执行主机上无法访问当前工作目录，AIP 会将标准输出文件写入 /tmp/。

如果使用**-o** 而不使用**-e**，则作业的标准错误将存储在输出文件中。

如果使用**-o** 而不使用**-N**，则作业报告将作为文件头存储在输出文件中。

如果同时使用**-o** 和**-N** 选项，输出将存储在输出文件中，作业报告将通过邮件发送。作业报告本身不包含输出，但报告会建议您在哪里查找输出。

文件路径可以包含以下动态模式，这些模式区分大小写:

* %J - 作业 ID

* %I - 索引（默认值为 0）

* %P - 项目名称（默认值为 "default"）

* %U - 用户名

* %G - 用户组（默认值与用户名相同）

* %H - 第一个执行主机名

* %A - csub -Al-app 中指定的应用程序名称（默认为“_”）

* %D - csub -Jd 中指定的职位描述（默认为“_”）

-P 项目名称
将作业分配给指定的项目。

例子:

```
csub-P "rnd project" myjob
```

-pack 作业提交文件
提交作业包而不是单个作业。指定作业提交文件的完整路径。在命令行中，此选项与任何其他 csub 选项不兼容。在作业提交文件中，每行定义一个作业请求，使用常规 csub 语法，但省略“csub”一词。对于文件中的请求，不支持以下 csub 选项：

-I-Ip-Is-XF-K-h-V

作业提交文件示例:

```
# 注释行
-o /dev/null -J myjob -R "rusage[license=1]" myjob
-o /dev/null -g /my/group -R "rusage[license=1:mem=3500]" herjob
-o out -W 1:30 -q queue -P runme ourjob
# end...
```

-G 用户组名
作业所属的 AIP 用户组名称（参见*cb.yaml*）。作业的用户必须是指定用户组的直接成员。如果用户属于为公平共享调度定义的多个用户组，则此选项允许用户为作业指定特定的用户组。如果未指定此选项，且用户属于多个用户组，则作业将调度到任意一个用户组下。如果用户所属的用户组未定义公平共享调度，则此选项无效。

-p 进程限制
将整个作业的进程数限制设置为 *process_limit*。默认值为无限制。超出限制将导致作业终止。

-proxy
提交需要设置 TCP/IP 代理的作业。详情请参阅/opt/skyform/etc/proxy.yaml。

-q “队列名称…”
将作业提交到指定的队列之一。如果只提交单个队列，引号可省略。如需查看可用队列的列表，请使用 **cqueues**。

当指定队列名称列表时，AIP 会根据作业的资源限制和其他限制（例如请求的主机、您对队列的可访问性、队列状态（关闭或打开）、队列是否可以接受独占作业等），从列表中为您的作业选择最合适的队列。队列的考虑顺序与这些队列的列出顺序相同。首先列出的队列将被首先考虑。

-R “*res_req*” [-R “*res_req*” …]
在满足指定资源要求的主机上运行作业。资源要求字符串指定作业所需的资源。AIP 使用资源要求来选择作业执行主机。资源要求字符串的大小限制为 512 字节。

资源需求字符串分为以下几个部分。每个部分都有不同的语法。

* 选择部分（select）。选择部分指定选择执行主机的标准

* 排序部分（order）。排序部分指示符合选择条件的主机应如何排序。

* 资源使用情况部分 (rusage)。资源使用情况部分指定任务的预期资源消耗。如果队列配置了 rusage 部分（请查看 cqueues -l queue_name 了解配置），则此部分将被忽略。但是，AIP 管理员可以在集群级别设置 job_rusage_override 参数，以启用作业 rusage 覆盖队列 rusage。

* 作业跨越部分（span）。作业跨越部分指示并行作业是否应跨越多个主机。

* 资源组备选选项（same）。并行作业选择的多个主机应具有相同的资源值。

* CPU 和内存亲和性资源部分（affinity）。亲和性部分指定作业的 CPU 和内存绑定要求。

资源需求字符串部分具有以下语法：

select[selection_string]

order[order_string]

same[resource_name]

rusage[usage_string [, usage_string][| usage_string] …]

span[span_string]

每个部分必须按所示方式输入方括号。每个资源需求部分之间必须用空格分隔。

备注： selection_string 的最小单元语法为: “*资源名 op 值*”或者“tag”。多个最小单元间可以用“&&“(与)、“||”（或）相连。资源名或者 tag 名为*cinfo* 输出中 RESOURCE_NAME 列出名字之一。

op 为：==(等于)，!=(不等于)，<(小于)，>(大于)，<=(小于或等于)，>=(大于或等于)。

tag 不能有 op 和 value。

所有的值都不能带单位，如 M、G 等。mem、swap、tmp 的单位为 MB。

您可以省略 select 关键字和方括号。

备注： order_string 的语法为：“*资源:资源:…*”。资源名必须是数值型资源（*cinfo* 的输出 TYPE 为 Number）。多个资源排序可以用冒号 (:) 隔开

备注： same 的 resource_name 必须是数值型或者字串型，不能是 tag。

备注： rusage_string 的语法为：“*数值型资源 = 值*:duration= *保留时间分钟*:decay= *每分钟减少保留的值*（mem 的单位为 MB）”。多个资源保留之间用逗号 (,) 隔开。

mem，swap，tmp 的保留值单位为 MB，但可以在数字尾部加 G 表示 GB。

备注： span_string 的语法为：“ptile=N”（表示每台主机分配 N 个作业槽）或“hosts=1”（表示所有作业槽必须分配在同一台主机上）。

例如:

```
csub-R "type==x86_64Linux order[ut] rusage[mem=100]" mjob
```

相当于以下内容:

```
csub -R "select[type==x86_64Linux] order[ut] rusage[mem=100]" mjob
```

资源需求中指定的任何特定于运行队列长度的资源（例如 r15s、r1m 或 r15m）均指规范化的运行队列长度。如果需要在字符串中包含连字符 (-) 或其他非字母字符，请将文本括在单引号中，例如，csub -R "select[hname!='node-x01']"。

select[] 和 rusage[] 部分中的内存 (mem)、交换 (swp) 和临时磁盘空间 (tmp) 限制以 MB 为单位指定。

名为 "slots" 的资源是 AIP 系统中的预定义资源。此资源在任何命令中均不可见。它可用于根据空闲作业槽的数量对候选作业执行主机进行排序。要提交在具有更多空闲作业槽的主机上运行的作业（即所谓的 "条带化" 调度策略），请执行以下操作：

```
csub-R "order[slots]" mjob
```

提交在具有较少空闲作业槽的主机上运行的作业（所谓的 "打包" 调度策略）：

```
csub-R "order[-slots]" mjob
```

另一个内置资源是 "maxslots"，它表示在 cb.yaml 中为每个主机配置的 MXJ（最大作业槽位数量）参数。此资源可用于作业提交时的资源需求字符串，或队列配置（cb.yaml）。例如，"order[maxslots]" 会根据最大作业槽位数量按降序对要调度的主机进行排序。

要提交在 CentOS 6 或 CentOS 7 上运行的作业：

```
csub-R "centos6 || centos7" mjob
```

提交一个在负载较轻（CPU 利用率）且至少有 100MB 可用交换空间的 centos7 主机上运行的作业。

```
csub -R "swap>100 && centos7 order[ut]" mjob
```

cb.base 中定义了一个名为 bigmem 的资源，在 cb.yaml.AIP 中将其作为 host002 的独占资源。使用以下命令提交在 host002 上运行的作业：

```
csub-R "bigmem" mjob
```

或者:

```
csub-R"defined(bigmem)"mjob
```

为 VCS 应用程序的许可证配置了一个名为 vcs 的静态共享资源。要提交在有可用许可证的主机上运行的作业，请执行以下操作:

```
csub-R "select[defined(vcs)] rusage[vcs=1]" mjob
```

以下作业请求在作业期间使用 20 MB 内存，并请求 1 个许可证使用 2 分钟:

```
csub-R"rusage[mem=20, license=1:duration=2]" mjob
```

以下作业请求 20 MB 内存和 50 MB 交换空间，持续 1 小时，以及 1 个许可证，持续 2 分钟:

```
csub-R "rusage[mem=20:swp=50:duration=1h,license=1:duration=2]" mjob
```

以下作业请求在作业期间使用 20 MB 内存、在 1 小时内使用 50 MB 交换空间以及在 2 分钟内使用 1 个许可证。

```
csub-R "rusage[mem=20,swp=50:duration=1h,license=1:duration=2]" mjob
```

以下作业请求 50 MB 的交换空间，在 2 小时的持续时间内线性减少保留量，并请求 1 个许可证，有效期为 2 分钟:

```
csub-R "rusage[swp=50:duration=2h:decay=1,license=1:duration=2]" mjob
```

以下作业请求两个持续时间相同但衰减不同的资源:

```
csub-R "rusage[mem=20:duration=30:decay=1,lic=1:duration=30]" mjob
```

您正在运行应用程序版本 1.5（作为名为 app_lic_v15 的资源），以及应用程序版本 2.0.1（作为名为 app_lic_v201 的资源）。2.0.1 版本的许可证密钥与 1.5 版本向后兼容，但 1.5 版本的许可证密钥不适用于 2.0.1 版本。

使用 || 运算符的作业级资源需求规范优先于任何队列级资源需求规范。

- 如果您只能使用一个版本的应用程序运行作业，请提交作业而不指定其他资源。要提交仅使用 app_lic_v201 的作业，请执行以下操作:

  ```
  csub-R "rusage[app_lic_v201=1]" mjob
  ```

- 如果您可以使用任一版本的应用程序运行作业，请尝试保留应用程序的 2.0.1 版本。如果不可用，您可以使用 1.5 版本。要提交先尝试 app_lic_v201 再尝试 app_lic_v15 的作业，请执行以下操作:

  ```
  csub -R "rusage[app_lic_v201=1||app_lic_v15=1]" mjob
  ```

- 如果不同版本的应用程序需要不同的系统资源，您可以在 rusage 字符串中指定其他资源。要提交一个使用 20 MB 内存（app_lic_v201）或 20 MB 内存和 50 MB 交换空间（app_lic_v15）的作业，请执行以下操作:

  ```
  csub-R "rusage[mem=20:app_lic_v15=1||mem=20:swp=50:app_lic_v201=1]" mjob
  ```

使用 affinity 资源要求的示例:

```
csub-R "rusage[mem=4734] span[hosts=1] affinity[core(1):cpubind=core]" myjob
```

AIP 会将作业 myjob 绑定到核心。

```
csub-R "affinity[core(1,exclusive=(core,
↪alljobs)):cpubind=core:membind=localonly:distribute=pack]"myjob
```

AIP 将为该作业分配一个独占的内核，即该内核不会与其他作业共享。系统会使用连接到该内核的本地内存，并会选择内核，使其尽可能“打包”到较少的插槽上。

cb.yaml 中，如果主机设置了 maxslots: cpu，则表示开启了 CPU 绑定。

备注: AIP 目前仅支持以上两个示例中的 affinity 符串。

当需要在一台机器上运行的多线程或并行作业（“span[hosts=1]”）使用 “-R rusage[mem=…]” 指定内存使用量时，内存使用量要求适用于单台机器，而不是乘以作业槽位数量。例如:

```
csub -n 4 -R "rusage[mem=100] span[hosts=1]" mymulti-threaded-job
```

调度程序将分配 100MB。

以上规则适用于所有基于插槽的资源，包括交换区、GPU 或任何插件插槽资源。当定义 span[hosts=1] 时，rusage 值将应用于整个作业，而非单个插槽。如果未指定 spen[hosts=1]，rusage 值将应用于单个分配的插槽。

- 当 span[hosts=1] 与 rusage[slot_resource* =⋯] 同时指定时，不允许使用 min,max 参数，否则调度器会困惑于如何为每个 slot 分配资源。以下作业提交无效:

```
csub-R "2,3{rusage[gpu=1] span[hosts=1]}" myjob
csub-n2,3 -R "rusage[mem=500] span[hosts=1]" myjob
```

- 多个主机具有相同的资源值如果所有主机具有相同的性能或相同的资源，则并行作业（例如 MPI 作业）将以最佳方式运行。资源需求的 same[] 部分可用于指示调度程序分配具有相同资源值的主机。

在同一节中定义的资源名称可以是任何内置文本/字符串资源或静态数字资源：type、model、cpuf、maxmem、maxswp、maxtmp、maxslots、ngpus、ncpus，以及通过 RESS 定义的任何外部资源（更多信息请参阅*ress*）。例如:

```
csub -n 64 "same[cpuf]" myjob
```

分配给该作业的所有主机将具有相同的 CPU 系数。

多种资源需求

对于某些应用程序（例如 AI 作业），每个应用程序组件都需要不同类型的资源。例如，TensorFlow 的参数服务器只需要 CPU，而 TensorFlow 的作业通常需要 GPU。在这种情况下，资源需求字符串需要包含多个部分。例如:

```
csub-R "4 {mem>10} 5 {rusage[gpu=1]}" myjob
```

在上面的例子中，每个资源需求都用括号 {} 括起来。括号前面的数字表示该资源需求所需的槽位数量。括号 {} 内的资源需求语法遵循上述规则（select、order、rusage、span 等）。

作业的资源需求和队列中定义的资源需求

队列配置中（参考*cb.yaml*) 的 resspec 参数可以定义队列级别的资源需求。作业级的资源需求和队列定义的资源需求的关系如下:

- select[]: 选择主机的过滤器，如 mem>100 && ut<0.3，这部分 bsub -R 里的和队列中 resspec 定义的是“与”的关系
- order[]: 作业定义的 order 优先对主机排序，然后再按队列 resspec 里定义的 order 对主机排序
- rusage[]: 作业定义的 rusage 里的参数覆盖队列中定义的，但作业中定义的值不能大于队列中定义的值，否则作业不能提交。若 cb.yaml 中有 cluster: job_rusage_override: yes，队列中定义的 rusage 值只是缺省值，作业提交的值可以大于队列中定义的值。
- span[]: 作业定义了 span，队列 resspec 中也定义了 span，作业将无法调度，只能有一个地方定义 span。
- same[]: 作业定义 same 的和队列 resspec 中的定义的 span 是“与”的关系。

-rnc 命令

指定当作业分配被修改（扩展和减少）时在第一个执行主机上调用的可执行文件的完整路径。

例子:

```
csub -n 4 -rnc /home/u001/jobrnc.sh myjob
```

-sp 优先级

指定用户分配的作业优先级，允许用户在队列中对其作业进行排序。优先级的有效值为 1 到 100 之间的任意整数。错误的作业优先级会被拒绝。AIP 和队列管理员可以指定 100 以上的优先级。

作业所有者可以更改其自己作业的优先级。AIP 和队列管理员可以更改队列中所有作业的优先级。

作业顺序是确定作业是否符合调度条件的首要考虑因素。无论作业优先级如何，作业仍需遵循所有调度策略。优先级相同的作业将按“先到先得”的顺序进行处理。

可以配置用户分配的作业优先级，并自动提升作业优先级，以自动提高已等待指定时间的作业的优先级。

-S 堆栈限制
为属于作业的每个进程设置每个进程（软）堆栈段大小限制（KB）（请参阅 getrlimit(2)）。

-Si *Singularity* 镜像文件路径
指定要运行的作业的 Apptainer /Singularity 镜像文件路径。

-So *Singularity_run_options*
指定“apptainer run”或“singularity run”命令的选项。

-Sb *Singularity/apptainer_run_bind* 值
指定 Apptainer /Singularity 容器可以访问的一组主机目录。一个好的做法是指定 AIP 安装目录（例如 /opt/skyformai、/opt/skyformai_shared）以及任何共享应用程序目录，例如 MPI 目录。

-t [[月:] 日:] 时: 分
指定作业终止的截止时间。如果 LINUX 作业在终止时间仍在运行，则会向该作业发送 SIGUSR2 信号，如果该作业在十分钟内未终止，则会被终止。（有关如何终止这些作业的详细说明，请参阅 ckill。）在队列定义中，可以配置“kill_action”（在 cb.yaml 中配置）操作来覆盖 **ckill** 默认操作（请参阅*cb.yaml* 中的 kill_action 参数）。

终止时间的格式为 [[month:]day:]hour:minute，其中数字范围如下：月份 1-12、日期 1-31、小时 0-23、分钟 0-59。

必须至少指定两个字段。这些字段的格式假定为“小时: 分钟”。如果指定三个字段，则假定为“日: 小时: 分钟”，如果指定四个字段，则假定为“月: 日: 小时: 分钟”。

-U 消息模板
作业调度后，使用此处指定的模板向作业发送消息。

message_template 可以包含任何带有以下保留字的字符。这些保留字将自动被 AIP 替换。

“JOBID”：作业 ID 字符串。

“HOSTIP”：作业执行的第一个主机的 IPv4 地址。

“PORT”：作业的第一个分配端口。此端口由 AIP 分配。支持“PORT+100”格式，如果 AIP 分配的端口为 16331，则此值将被替换为“16431”。

“USER”：提交作业的用户名。

“PROXY_IP”：当配置了 AIP 代理并使用“-proxy”选项提交作业时，此字符串将被代理公共 IP 地址替换。

“PROXY_PT”：当配置了 AIP 代理并使用“-proxy”选项提交作业时，此字符串将被作业分配的代理公共端口替换。

在 message_template 末尾添加“DELAY=n”。该帖子将在任务调度后“n”秒发布。示例：用户“cadmin”运行以下命令。

```
csub -U "ssh -o PORT USER@HOSTIPDELAY=10" myjob
```

发布的消息是：“ssh -o 16331 cadmin@192.168.10.10 ”，作业启动 10 秒后消息发布。

-u 邮件用户
将邮件发送到指定的电子邮件目的地。

-v *swap* 限制
将整个作业的进程虚拟内存总限制设置为 *swap_limit*（以 KB 为单位，或以 M 或 G 结尾，表示以 MB 或 GB 为单位）。默认值为无限制。超出限制会导致作业终止。

-w '依赖关系表达式'

除非依赖项表达式的计算结果为 TRUE，否则 AIP 不会放置您的作业。如果您指定的作业依赖 AIP 无法找到（例如尚未提交的作业），则作业提交会失败。

依赖表达式是由一个或多个依赖条件组成的逻辑表达式。要创建多个条件的依赖表达式，请使用以下逻辑运算符：

&&（和）

||（或）

!（不是）

如果需要，请使用括号来指示运算顺序。

将依赖项表达式括在单引号（'）中，以防止 Shell 解释特殊字符（空格、任何逻辑运算符或括号）。如果对依赖项表达式使用单引号，请对其中带引号的项（例如作业名称）使用双引号。

在依赖条件中，除非您是 AIP 管理员，否则作业名称仅指定您自己的作业。默认情况下，如果您使用作业名称指定依赖条件，并且您的多个作业具有相同的名称，则所有具有该名称的作业都必须满足测试。如果 cb.yaml 中的 JOB_DEP_LAST_SUB 设置为 1，则测试将针对最近提交的作业进行。使用双引号（""）括起以数字开头的作业名称。在作业名称中，在字符串末尾指定通配符星号 (*)，以指示名称以该字符串开头的所有作业。例如，如果您使用 jobA* 作为作业名称，则它指定名为 jobA、jobA1、jobA_test、jobA.log 等的作业。

使用带有依赖关系条件的 * 来定义作业阵列元素之间的一对一依赖关系，使得一个数组的每个元素都依赖于另一个数组的相应元素。作业阵列大小必须相同。例如，csub -w "done(myarrayA[*])" -J "myArrayB[1-10]" myJob2 表示，myArrayB 中的元素 1 必须先完成，myArrayA 中的元素 1 才能开始，依此类推。

提交数组后，您还可以使用 * 与 cmod 建立一对一的数组元素依赖关系。

如果您想通过数组名称指定数组依赖关系，请在 cb.yaml 中设置 JOB_DEP_LAST_SUB。如果您未设置此参数，则当先前的数组中存在同名但索引不同的数组时，作业将被拒绝。

在依赖条件中，变量 *op* 表示以下关系运算符之一：

>

>=

<

<=

==

!=

使用以下条件来形成依赖表达式。

done(job_ID| " job_name " ⋯ **)**

作业状态为 "DONE"。

AIP 指的是内存中 job_name 最旧的作业。

ended (job_ID | "job_name ")

作业状态为 EXIT 或 DONE。

exit (job_ID | "job_name" [**,** [op] exit_code])

作业状态为 EXIT，并且作业的退出代码满足比较测试。

如果指定没有运算符的退出代码，则测试是否相等（假定 ==）。

如果您仅指定作业，则任何退出代码都满足测试。

external(job_ID| “job_name “, “状态 “)

指定作业状态或消息描述的第一个字（无空格）。仅评估第一个字。

该作业具有指定的作业状态，或者该作业状态的文本以指定的单词开头。

job_ID| “job_name”

如果您指定一个没有依赖条件的作业，则测试针对的是 DONE 状态（AIP 默认采用“完成”依赖条件）。

numdone(job_ID，op | *)

对于作业阵列，处于 DONE 状态的作业数量满足测试。使用 *（不带运算符）指定数组中的所有作业。

numended(job_ID，op| *)

对于作业阵列，处于 DONE 或 EXIT 状态的作业数量满足测试。使用 *（不带运算符）指定数组中的所有作业。

numexit(job_ID ， op| *)

对于作业阵列，处于 EXIT 状态的作业数量满足测试。使用 *（不带运算符）指定数组中的所有作业。

numhold(job_ID ， op| *)

对于作业阵列，处于 PSUSP 状态的作业数量满足测试。使用 *（不带运算符）指定数组中的所有作业。

numpend(job_ID ， op| *)

对于作业阵列，处于“等待”状态的作业数量满足测试。使用 *（不带运算符）指定数组中的所有作业。

numrun(job_ID ， op| *)

对于作业阵列，处于 RUN 状态的作业数量满足测试。使用 *（不带运算符）指定数组中的所有作业。

numstart(job_ID ， op| *)

对于作业阵列，处于 RUN、USUSP 或 SSUSP 状态的作业数量满足测试。使用 *（不带运算符）指定数组中的所有作业。

post_done(job_ID| “作业名 “)

作业状态为 POST_DONE（指定作业的后处理已完成且没有错误）。

post_err(job_ID| “作业名 “)

作业状态为 POST_ERR（指定作业的后处理已完成，但有错误）。

started(job_ID| “作业名 “)

作业状态为：

- RUN、DONE、或 EXIT
- PEND 或 PSUSP，并且作业有一个正在运行的预执行命令 (csub -E)。

-W [小时:]* 分钟 *[/主机名 |/主机型号]

设置作业的运行时间限制。如果 LINUX 作业的运行时间超过指定的运行时间限制，则会向该作业发送 SIGUSR2 信号，如果该作业在十分钟内未终止，则会被终止。（有关如何终止这些作业的详细说明，请参阅 ckill。）在队列定义中，可以配置 TERMINATE 操作来覆盖 **ckill** 默认操作（请参阅*cb.yaml*中的 JOB_CONTROLS 参数）。

运行限制的格式为 [小时:] 分钟。分钟数可以指定为大于 59 的数字。例如，三个半小时可以指定为 3:30 或 210。

您也可以选择提供 AIP 中定义的主机名或主机型号名称。必须在运行限制与主机名或型号名称之间插入“/”。（请参阅*cinfo* 获取主机型号信息。）

您指定的 CPU 时间是标准化 CPU 时间。这样做的目的是，即使作业发送到 CPU 速度更快或更慢的主机，其处理量也大致相同。每当指定标准化 CPU 时间时，执行主机上的实际时间等于指定时间乘以标准化主机的 CPU 因子，然后除以执行主机的 CPU 因子。

如果作业还通过 **csub -t** 选项指定了终止时间，AIP 会判断该作业在终止时间之前是否能够实际运行运行限制所允许的指定时间长度。如果不能，则该作业将被中止。如果在*cb.yaml* 中设置了 IGNORE_DEADLINE 参数，则会覆盖此行为并忽略运行限制。

-XF

启用 ssh X11 转发。这允许作业通过 ssh X11 隧道将其显示内容发送回作业提交主机。当作业提交主机上的 X 服务器或 VNC 服务器不接受常规 X11 显示请求时（例如，X 服务器或 VNC 服务器以 "-nolisten tcp" 选项运行），此功能非常有用。

要使用此功能，请确保用户无需输入密码即可通过 ssh 连接到任何作业执行主机。

该作业以交互方式运行，即 csub 命令将被阻止，直到作业完成为止。输入 Ctrl-C 可终止该作业。

-Zs

将作业命令文件假脱机到 cb.yaml 中 JOB_SPOOL_DIR 参数指定的目录中，并将假脱机文件作为作业的命令文件。

默认情况下，如果未指定 JOB_SPOOL_DIR，则输入文件将被缓存到 CB_ENVDIR/../work/data/info 目录。如果 info 目录不存在，AIP 会在缓存文件之前创建该目录。作业完成后，AIP 会删除缓存文件。

嵌入式作业命令（#CSUB）不支持-Zs 选项，因为 AIP 无法确定嵌入式作业命令中要假脱机的第一个命令。

-h

将命令用法打印到 stderr 并退出。

-V

将 AIP 发布版本打印到 stderr 并退出。

命令 [参数]

作业可以通过命令行参数 command 指定，如果 command 不在命令行中，则可以通过标准输入指定。command 可以是提供给 UNIX Bourne shell 的任何内容（参见 **sh** (1) ）。command 假定以第一个不属于 csub 选项的单词开头。command 之后的所有参数都作为 command 的参数提供。

如果命令行中未指定作业，**csub** 将从标准输入读取作业命令。如果标准输入是控制终端，系统会提示用户输入 "csub>" 输入作业命令。输入完成后，在新行输入 CTRL-D 即可终止。您可以通过标准输入提交多个命令。命令将按照指定的顺序执行。如果行以 #CSUB 开头，也可以在标准输入中指定 csub 选项；例如，"#CSUB -x"。如果 csub 命令行和标准输入中都指定了选项，则命令行选项将覆盖标准输入中的选项。用户可以通过在标准输入的第一行指定 shell 路径名来指定运行命令的 shell，例如 "#!/bin/csh"。如果第一行未指定 shell，则使用 Bourne shell。标准输入功能可用于假脱机用户的作业脚本；例如，"csub < script"。请参阅下面的示例，了解通过标准输入指定命令的示例。

输出

如果作业提交成功，则显示作业 ID 和作业提交到的队列。

示例

% **csub sleep 2.0**

将 UNIX 命令 sleep 及其参数 2.0 作为作业提交。

%**csub -q short -o my_output "pwd;ls"**

将 LINUX 命令 pwd 和 ls 作为作业提交到名为 short 的队列，并将作业输出存储在 my_output 文件中。

%**csub -m "host1 host3 host8 host9" my_program**

提交 my_program 在以下候选主机之一上运行：host1、host3、host8 和 host9。

% **csub -I ls**

提交交互式作业，在用户终端显示 ls 的输出。

% **csub -Ip vi myfile**

提交交互式作业来编辑 myfile。

% **csub -Is csh**

提交一个交互式作业，以启动 csh 作为交互式 shell。

% **csub -b 20:00 -J my_job_name my_program**

提交 my_program 在晚上 8 点后运行，并为其分配作业名称 my_job_name。

% **csub my_script**

将 my_script 作为作业提交。由于 my_script 被指定为命令行参数，因此 my_script 文件不会被假脱机处理。在作业完成之前对 my_script 文件的后续更改可能会影响此作业。

% **csub < default_shell_script**

其中 default_shell_script 包含:

```
sim1.exe
sim2.exe
```

文件 default_shell_script 已假脱机，并且由于脚本的第一行未给出 shell 规范，因此命令将在 Bourne shell 下运行。

% **csub < csh_script**

其中 csh_script 包含:

```
#!/bin/csh
sim1.exe
sim2.exe
```

csh_script 已假脱机并且命令将在 /bin/csh 下运行。

% **csub -q night < my_script**

其中 my_script 包含:

```
#!/bin/sh
#CSUB -q test
#CSUB -o outfile -e errfile #我的默认stdout、stderr文件
#CSUB -m "host1 host2" #我的默认候选主机
#CSUB -f "input > tmp" -f "output << tmp"
#CSUB -D 200 -c 10/host1
```

(下页继续)

(续上页)

```
#CSUB -t 13:00
#CSUB -k "dir 5"
sim1.exe
sim2.exe
```

该作业被提交到夜间队列而不是测试，因为命令行覆盖了脚本。

% csub -b 20:00 -J my_job_name

```
csub> sleep 2.00
csub> my_program
csub> CTRL-D
```

作业命令以交互方式输入。

作业执行环境设置的环境变量

AIP 将大多数环境变量从提交主机传输到执行主机。除了从用户环境继承的环境变量外，AIP 还为作业设置了许多其他环境变量：

- CB_ERRORFILE：用 csub -e 指定的错误文件的名称。
- CB_JOBID：AIP 分配的作业 ID。
- CB_JOBINDEX：属于作业阵列的作业的索引。
- CB_CHKPNT_DIR：每次提交检查点作业时都会设置此变量。该变量的值为 chkpnt_dir/job_Id，即提交作业时指定的检查点目录的子目录。该子目录由已提交作业的作业 ID 标识。
- CB_EFFECTIVE_RUSAGE：当在资源需求字符串的 rusage 部分使用 OR 时，AIP 使用此变量来指示使用哪个 rusage 字符串来调度作业。
- CB_HOSTS：用于运行作业的主机列表。对于顺序作业，这只有一个主机名。对于并行作业，这包含多个主机名。
- CB_MCPU_HOSTS：与 CB_HOSTS 类似。格式为：“主机 1 槽位号 1 主机 2 槽位号 2 …”
- CB_JOBPID：作业的进程 ID
- CB_SUB_HOST：提交主机的名称。
- CB_QUEUE：调度作业的队列名称。
- CB_JOBNAME：作业的名称。
- CB_MAX_NUM_PROCESSORS：提交作业时请求的最大处理器数量。
- CB_PROJECTNAME：该作业的项目名称。
- CB_RESTART：如果作业是重新启动的作业或已迁移，则设置为“Y”。否则，此变量未定义。
- CB_EXIT_PRE_ABORT：设置为表示退出状态的整数值。如果预执行命令希望中止作业而不是将其重新排队或执行，则应以此值退出。
- CB_EXIT_REQUEUE：设置为队列的 REQUEUE_EXIT_VALUES 参数。如果队列未配置 REQUEUE_EXIT_VALUES，则此变量不定义。
- CB_INTERACTIVE：如果提交作业时使用了 -I 选项，则设置为“Y”。否则，未定义。
- CB_SUBCWD：这是提交主机上作业提交的目录。仅当目录未跨机器共享，或者由于账户映射导致执行账户与提交账户不同时，此目录才与 PWD 不同。
- CUBE_VERSION：表示 AIP 的当前版本。

- CB_ALLOCATION：描述调度程序分配的资源的 JSON 数据结构。
- CB_PORT：分配给作业的第一个端口。此信息也包含在 CB_ALLOCATION 中。
- GPU_ALLOCATION：GPU 分配信息。格式为 hostname:gpuID…例如：`d16:0,1 d17:2`

局限性

使用帐户映射时，命令*cview* 将不起作用。通过*csub* 的 -f 选项进行文件传输需要 rcp(1) 在提交主机和执行主机之间正常运行。使用 -N 选项请求邮件，和/或使用 -o 和 -e 选项分别指定输出文件和错误文件。

排错

csub 命令无法与调度器通讯时会在 stderr 上不断显示：AIP scheduler not responding. Keep trying…，直到可以与调度器通讯，或者用户用 Ctrl-C 中断命令。

作业提交失败可能会有以下原因：

Dependency condition syntax error
: 作业依赖定义语法错误

Queue does not accept exclusive jobs
: 作业提交的队列不接受独占作业

Job submission by root user is disabled
: 系统配置不允许 root 用户提交作业

No job Id can be used now
: 系统内作业太多，已经没有可用的作业号。这种情况一般是因为 cb.yaml 里设置的 max_jobid 太小造成的

Queue only accepts interactive jobs
: 队列配置不接受一般作业，只接受交互式作业（csub -Il-Isl-Ip）

Queue does not accept interactive jobs
: 队列配置不接受交互式作业

No such queue
: 作业提交的队列不存在

Queue has been closed
: 作业提交的队列已关闭，不再接受新的作业

User cannot use the queue
: 根据队列配置，用户不允许使用该队列

Invalid host name, host group name or cluster name
: 作业提交指定的主机名在系统中不存在

Too many processors are requested
: 作业需要的作业槽数超过集群内最大可用的作业槽数

Host or host group is not used by the queue
: 作业提交指定的主机不在队列配置的主机列表中

Queue does not have enough per-user job slots
: 作业所需的作业槽数超过了队列中配置的 usermaxslots

Requests from non AIP host are rejected
: 作业提交主机不属于 AIP 集群

Invalid argument
csub 命令行语法错误

Invalid time specification
提交参数中的时间定义语法错误

Job start time is later than termination time
作业定义了起始时间和结束时间，结束时间比起始时间早

Invalid job name
作业名语法错误

AIP library internal error
AIP 内部错误，请联系 AIP 技术支持

Failed to fork
提交主机进程数上限太小

RPC encode or decode error
csub 命令与调度器版本不一致

Invalid resource spec syntax
csub -R 定义的资源需求语法错误

Interactive job cannot be rerunnable
-I 参数与-r 参数冲突

User is not in the specified user group
用户不属于-G 指定的用户组

Request is aborted by esub
作业提交没有通过 esub 验证而失败

Job group does not exist 或 Invalid or empty job group name
-g 指定的作业组不存在或不合法

Job array index error
-J 定义的作业阵列语法错误

Job array index too large
作业阵列大小超过 cb.yaml 里定义的 max_array_size 参数值

Invalid user priority
-sp 定义的值不合法。值应该是 1-100

Invalid minimum and maximum slot requested
-n 定义的两个值错误

Single host job with slot resource reservation can’t define minimum and maximum slot requirements
如果作业请求与作业槽相关的资源（如 GPU），而要求作业在一台主机上，同时又定义了伸缩作业（-n min,max），这种作业定义 AIP 不支持

User has too many pending jobs
系统定义了用户最多等待作业的限制（如 10000）。提交作业的用户等到作业超过的这个限制

Key expired
企业版的 key 过期了，或者无效

5.1.46 cswitch

命令

cswitch - 将未完成的作业从一个队列切换到另一个队列

概要

cswitch [**-J** 作业名] [**-m** 主机名 | **-m** 主机组] [**-q** 队列名称] [**-u** 用户名 | **-u** 用户组 | **-u all**] 目标队列 [**0**]

cswitch 目标队列 [作业 *ID* | "作业 *ID*[索引列表]"] ···

cswitch [**-h** | **-V**]

描述

将一个或多个未完成的作业切换到指定队列。AIP 管理员和 root 可以切换其他用户提交的作业。

默认情况下，切换单个作业、最近提交的作业或同时满足其他指定选项（-m、-q、-u 和 -J）的最近提交的作业。指定 -0（零）可切换多个作业。

切换操作仅当指定的作业被新队列接受，如同已提交给新队列一样；并且，如果作业已调度到主机，则新队列可以使用该主机。如果切换操作失败，作业将保留在原处。

如果切换的作业尚未调度，则其行为将如同最初已提交到新队列一样。

如果已调度已切换的作业，则该作业将由新队列的 loadSched 和 loadStop 向量、PRIORITY、RUN_WINDOW 和其他配置参数控制，但其 nice 值和资源限制将保持不变，但 RUNLIMIT 将被重置为新队列的值。

此命令用于更改从队列继承的作业属性。

选项

0
（零）。切换多个作业。切换所有满足其他指定选项（-m、-q、-u 和 -J）的作业。

-J 作业名
仅切换具有指定作业名称的作业。

-m 主机名 | -m 主机组
仅切换调度到指定主机或主机组的作业。

-q 队列名
仅切换指定队列中的作业。

-u 用户名 | -u 用户组 | -u all
仅切换指定用户或组提交的作业，如果指定关键字 all，则切换所有用户提交的作业。

目标队列
必需。指定要将作业移动到的队列。

作业 *ID* ··· | "作业 *ID*[索引列表]" ···
仅切换指定的作业。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

另请参阅

cqueues、*chosts*、*cugroup*、*csub*、*cjobs*

5.1.47 ctask

命令

ctask - 在 AIP 集群中运行交非作业互式任务

概要

ctask [**-P**] [**-v**] [**-p**] [**-R** “资源需求”] [**-m** 主机…] 命令 [命令参数…]

ctask [**-h** | **-V**]

描述

在 AIP 库中的一台或多台主机上运行非作业交互式任务。

默认情况下，ctask 在与本地主机类型相同的主机上运行任务。如果存在多台相同类型的主机，则选择 CPU 和内存负载最低的主机。

如果本地主机满足指定的资源需求，或者远程执行失败，则在本地运行任务。

默认情况下，ctask 在运行任务时不会创建伪终端。

选项

-P
使用伪终端运行任务。这对于运行需要伪终端的程序（例如 vi）是必需的。

-v
详细模式。使用此选项将显示运行任务的主机名称。

-p
并行模式。如果多个主机满足选项 -R 指定的资源要求，则任务将在指定的主机上同时运行。

-R “资源需求”
在满足指定资源需求的主机上运行任务。默认情况下，仅在第一个主机上运行任务。如果指定了 -p，则所有满足资源要求的主机都将同时运行任务。

-m *hosts*…
在指定的一台或多台主机上同时运行任务。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

诊断

ctask 尝试使用 CBEXE 运行任务。如果 CBEXE 已关闭，ctask 将在本地主机上运行该任务。

5.1.48 ctop

命令

ctop - 将等待作业相对于队列中的第一个作业进行移动

概要

ctop *job_ID* | " *job_ID*[*index_list*]" [*position*]

ctop[**-h** | **-V**]

描述

更改等待作业或等待作业阵列元素的队列位置，以影响作业的调度顺序。

默认情况下，AIP 会按照作业到达的顺序（即先到先得）调度队列中的作业，但前提是存在合适的服务器主机。

ctop 命令允许用户和 AIP 管理员手动更改作业的调度顺序。用户只能操作自己的作业，而 AIP 管理员可以操作任何用户的作业。用户只能更改自己作业的相对位置。

如果由 AIP 管理员调用，**ctop** 会将选定的作业移至提交到队列的第一个具有相同优先级的作业之前。AIP 管理员可以更改队列中所有用户作业的位置。

如果由普通用户调用，**ctop** 会将选定的作业移至提交到队列的第一个具有相同优先级的作业之前。等待的作业由 cjobs 按照其将被考虑调度的顺序显示。

选项

job_ID **|** **"*job_ID*[*index_list*]"**
: 必填。要操作的作业或作业阵列的作业 ID。

  对于作业阵列，索引列表、方括号和引号是必需的。索引列表用于对作业阵列进行操作。索引列表是一个逗号分隔的列表，其元素的语法为 *start_index*[*-end_index*[**:***step*]]，其中 *start_index*、*end_index* 和 *step* 为正整数。如果省略 step，则假定 step 为 1。作业阵列索引从 1 开始。作业阵列索引的最大值为 1000。数组中的所有作业共享相同的 job_ID 和参数。数组的每个元素都由其数组索引区分。

position
: 可选。可以指定 *position* 参数来指示作业在队列中的位置。*position* 是一个正数，表示作业相对于队列开头的目标位置。这些位置仅与队列中适用的作业相关，具体取决于调用者是普通用户还是 AIP 管理员。默认值 1 表示该位置位于队列中所有其他具有相同优先级的作业之前。

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到标准错误输出并退出。

另请参见

cbot、*cjobs*、*cswitch*

5.1.49 cugroup

命令

cugroup - 显示用户组信息

概要

cugroup [**-r**] [**-l**] [*用户组名…*]

cugroup[**-h** | **-V**]

描述

显示用户组及其对应的用户名。

默认显示所有用户组信息。

显示用户组信息范围

- root 或集群管理员可以访问所有用户组信息。
- 当 cb.yaml 中 user_view_alljobs: yes，普通用户可以访问所有用户组信息。
- 当 cb.yaml 中 user_view_alljobs: no，或者该参数没有设置，用户只能访问自己所属的用户组信息。

选项

-l
: 输出作业槽使用信息。

-r
: 递归扩展用户组。扩展列表仅包含用户名，不包含子组名称。重复的用户名仅显示一次。

用户组名…
: 仅显示指定用户组的信息。指定多个用户组时请勿使用引号。

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到 stderr 并退出。

输出

输出字段如下。如果某个字段没有值，输出为‘-’。

在组员用户列表中，名称后跟斜杠 (/) 表示子组。

缺省输出

GROUP
用户组名。

ADMIN
用户组管理员用户名

G_MAX
用户组最大作业槽数限制

U_MAX
组内每个成员用户的最大作业槽数限制

CAT
组员配置的方式。

‘-’表示 AIP 配置文件定义。

‘SYS’表示由操作系统，如 LDAP 定义。

MEMBERS
组员用户列表。名称后跟斜杠 (/) 表示子组。

选项-l 的输出

GROUP
用户组名。

MAXSLOTS
用户组最大作业槽数限制。

MAXJOBS
用户组最大作业数限制。

MAXGPUS
用户组 GPU 个数限制。

USER_SL
用户组最大作业槽数限制。

NRUN_J
用户组里正在运行的作业数。

NRUN
用户组里正在运行的作业槽数。

NPEND
用户组里等待作业槽数。

NGPUS
用户组里作业使用的 GPU 数。

文件

用户组和用户共享在配置文件*cb.yaml* 中定义，或者在 ug.yaml 中。

5.1.50 cusers

命令

cusers - 显示用户和用户组的信息

概要

cusers [**-w**] [*用户名…*| *用户组…*| **all**]

cusers[**-h** | **-V**]

描述

显示用户和用户组的信息。

默认情况下，显示运行该命令的用户的信息。

选项

-w
以宽格式输出。在每行末尾显示每个用户或用户组的等待作业数量。

用户名…| 用户组…|all
显示指定用户或用户组的信息，或者如果指定为 all，则显示所有用户的信息。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

输出

显示用户和用户组列表，其中包含以下字段：

USER/GROUP
用户或用户组的名称。

JL/P
指定用户在每个处理器上可同时处理的最大作业槽数。这些作业槽由正在运行和暂停的作业或已预留作业槽的等待作业使用。此作业限制是按处理器配置的，以便多处理器主机拥有更多作业槽。如果显示短划线 (-)，则表示没有限制。作业/任务在 AIP 配置文件 cb.yaml(5) 中定义。

MAX
指定用户作业可同时处理的最大作业槽数。这些作业槽位可供正在运行和暂停的作业，或已预留作业槽位的等待作业使用。如果显示字符 “-”，则表示没有限制。MAX 由配置文件 cb.yaml(5) 中的 MAX_JOBS 参数定义。

NJOBS
指定用户的作业当前使用的作业槽位数量。等待的并行作业被计为 *n* 个作业槽位，因为它在调度时将使用队列中的 *n* 个作业槽位。

PEND
指定用户的作业使用的等待作业槽位数量。

RUN
指定用户的正在运行的作业使用的作业槽位数量。

SSUSP
指定用户的系统暂停作业使用的作业槽位数量。

USUSP
指定用户的用户暂停作业使用的作业槽位数量。

RSV
指定用户的等待作业所占用的作业槽位数量，这些作业槽位已预留。

PJOBS
仅限宽格式。指定用户的等待作业数量。

另请参阅

cugroup、*cb.yaml*

5.1.51 cview

命令

cview - 显示作业的标准输出 (stdout) 和标准错误输出 (stderr)

概要

cview[**-f**] [**-s** *size_limit*] [**-n** *start_line*] [**-q** *queue_name* | **-m** *host_name* | **-J** *job_name* | *job_ID* | "*job_ID*[*index_list*]"]

cview[**-h** | **-V**]

描述

显示在调用此命令之前，某个作业产生的标准输出和标准错误输出。

若作业提交时没有指定输出文件 (csub -o | -oo)，该命令只能用于运行中的作业。若作业指定了输出文件，该命令也可用于已完成但未被调度器从内存中清除的作业。

默认情况下，使用 cat 命令显示输出。

此命令可用于监控作业进度和识别错误。如果发现错误，可以通过终止错误作业来节省宝贵的用户时间和系统资源。

选项

-f
使用 tail -f 命令显示作业的输出。

-s 输出字节数
指定大小限制（以字节为单位）。如果超过指定的大小限制，则不显示。

-n 开始行
从指定的行开始输出。

-q 队列名
对指定队列中最近提交的作业进行操作。

-m 主机名
执行最近提交的、已调度到指定主机的作业。

-J 作业名
执行最近提交的、具有指定作业名称的作业。

***job_ID*｜ "*job_ID*[*index_list*]"**
执行指定的作业。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

5.1.52 runtask

命令

runtask - 在一组主机上启动远程并行作业子任务

概要

runtask [**-x**] [**-n**] [**-u** 主机列表文件 |**-z** 主机名… | 主机名] 命令 [命令参数…]

runtask[**-h** | **-V**]

描述

许多并行应用程序（包括 MPI 实现）都使用无密码 SSH 作为启动机制。runtask 命令提供了一种替代 SSH 的透明方法，用于在 AIP 中启动并行应用程序，如 MPI、多机 AI 模型训练、多机推理任务等。

runtask 支持命令行类似 SSH：ssh 主机名 命令…

所有其他 SSH 选项均不支持。

runtask 透明地直接连接到远程主机上的 CBEXE，然后创建并跟踪远程任务，并将连接返回到 AIP。runtask 会监控和收集完整的作业进程资源使用情况，可以通过 *cjobs -l* 查看。

runtask 仅在 AIP 下工作。它只能用于在作业分配的远程主机上启动任务。它不能用作独立命令，也不能在不属于作业主机分配的主机上运行任务。

选项

-n
标准输入来自 /dev/null。

-u 主机列表文件
在 主机列表文件中列出的所有主机上执行任务。

指定包含主机名列表的文件的路径。每个主机名必须在主机列表文件中单独一行列出。

此选项与 -z 选项互斥。

主机名
远程任务启动的主机名。

-x
此选项仅用于 SSH 兼容性，不影响 runtask 功能。

-z "主机名…"
在所有指定的主机（多个主机名以空格隔开）上执行任务。

指定要执行任务的主机列表。如果指定了多个主机名，则必须用引号掩码（"或'"）括起来，并用空格分隔。

此选项与 -u 选项互斥。

命令 [命令参数…]
指定要执行的命令。这必须是命令行中的最后一个参数。

-h
将命令用法打印到标准错误输出并退出。

-V
将产品发行版本打印到标准错误输出并退出。

环境变量

一下环境变量影响远程任务运行：

CB_TASK_DOCKER_IMAGE
指定用于在 Docker 容器内运行任务的 Docker 镜像名称。默认情况下，此变量未设置，任务将在不使用 Docker 的主机上运行。示例："CB_TASK_DOCKER_IMAGE=centos:7"。

CB_TASK_DOCKER_CONTAINER
该任务在现有的 Docker 容器内运行。此变量指示正在运行的 Docker 容器 ID。

CB_TASK_DOCKER_OPTIONS
在 Docker 容器中运行任务时，指定其他 docker 命令选项。此变量是可选的。例如："CB_TASK_DOCKER_OPTIONS='–mount type=bind,source=/share/apps,target=/share/apps'"。

CB_TASK_DOCKER_ENVS
在 "name=value" 格式中指定空格分隔的字符串作为 Docker 容器任务的环境变量。例如："CB_TASK_DOCKER_ENVS=' class=4 type=ty'"。

5.1.53 vncsub

命令

vncsub - 提交 VNC 会话图形作业

dcvsub - 提交 DCV 会话图形作业

概要

vncsub [*csub 参数*] [**-geometry** *MxN*] [**-vglrun**] [**-xstartup** 桌面启动脚本] [**-vnc** *vncserver 可执行文件所在目录路径*] [**-debug**] [**-command**] [**desktop**] [*作业命令…*]

dcvsub [*csub 参数*] [**-debug**] [**-command**] [**desktop**] [*作业命令…*]

vncsub [**-h**]

描述

提交图形化作业。作业运行前，会创建一个 VNC 会话或 DCV 会话。

作业提交后，vncsub 会在作业消息中发送 VNC 会话密码。

vncsub/dcvsub 会读取可选配置文件 vncsub.yaml。默认情况下，vncsub.yaml 位于调度程序的配置目录中，例如 /opt/skyformai/etc 或 /opt/lsf/conf。但是，如果 vncsub.yaml 无法安装在共享目录中（因为某些参数可能因主机而异），则应将其放置在 /etc/skyformaip、/etc/lsf 或 /etc/slurm 中。

VNC 作业依赖于 SkyForm CRV 服务，该服务运行在 VNC 作业的主机上，提供 VNC 到 Web 的转换。

vncsub.yaml 配置文件支持以下参数：

pp: 任何字串
: 使用 NGINX 将来自 CRV 服务的 Web 流量路由到与 Web 服务器相同的端口。

  默认情况下，不使用 NGINX 路由 CRV 流量。端口 16000 需要穿过 Web 服务器防火墙。

  默认值未指定。

crv_port: ***port***
: 默认情况下，SkyForm CRV 服务在端口 16000 上运行。如果它使用其他端口，vncsub 需要知道该端口。此参数可以配置为 CRV 使用的端口号（而不是默认的 16000）。

desktop: ***app|project***
: 提交桌面或桌面 + 命令作业时，请使用 csub 的 -A 参数或 -P 参数对作业进行分类。这将允许 cb.yaml 中的 limits 精细控制作业限制。例如，为某些用户禁用正在运行的桌面作业。

geometry: ***MxN***
: 指定 VNC 会话的大小（以像素为单位）。默认值为 1900x1060。这仅适用于 VNC 作业。DCV 作业不接受此参数。

vnc: 路径
: 指定包含可执行文件 vncserver、vncpasswd 和 Xvnc 的目录路径。默认使用 AIP 嵌入式 TurboVNC。如果使用系统 TigerVNC，请指定 -vnc /usr/bin。这仅适用于 VNC 作业。

desktopautokill: ***yes|no***
: 某些系统（例如 Kyrin OS）的 VNC 服务器 -autokill 选项存在问题。在这种情况下，应将此参数设置为“no”，否则 VNC 桌面将完全无法运行。默认情况下，该值为“yes”，这意味着当桌面用户注销时，作业将完成。如果将 *desktopautokill* 设置为“no”，则用户退出桌面后，作业仍会继续运行。需要手动终止桌面作业。

vglrun: *y|n*
指定是否应使用 VirtualGL 运行命令 *vglrun* 来启动应用程序。默认情况下，VNC 会话不在 VirtualGL 中运行。

skyformvnc: 路径
指定 skyformcrv 目录的路径。默认值为：/opt/skyformai/etc/skyformcrv。

dcvpubweb_ips: *yaml*
指定 DCV 公共 IP 地址。默认情况下，dcvsub 使用检测到的 IP 地址发布 DCV 访问 URL。此地址可能并不总是正确的。此参数用于提供正确的 Web 访问 IP 地址。以下示例包含 3 个 DCV 服务器:

```
dcvpubweb_ips:
- "host1: 11.23.1.2"
- "host2: 11.23.1.3"
- "host3: 11.23.1.4"
```

dcv_web_url: *URL* **前缀**
指定 DCV 服务器上 /etc/dcv/dcv.conf 文件中配置的 [connectivity] web-url-path 的值。该值支持使用环境变量，例如 $HOSTNAME。此参数用于 dcvsub 生成正确的 DCV 访问 URL。

默认情况下，此值在 DCV 配置中为空。

vncpubweb_ips: *yaml*
指定 VNC 公网 IP 地址。默认情况下，vncsub 使用检测到的 IP 地址发布 VNC 访问 URL。此地址可能并非总是正确。此参数用于提供正确的 Web 访问 IP 地址。以下示例包含 3 个 VNC 服务器:

```
vncpubweb_ips:
- "host1: 11.23.1.2"
- "host2: 11.23.1.3"
- "host3: 11.23.1.4"
```

watermark: 字串
为图形背景添加水印。动态变量包括：%U：用户名，%J：作业 ID，%T，VNC 会话的日期和时间。示例：watermark: SkyFormAIP_%U_%J。水印中的空格字符将转换为“_”。

xstartup: 桌面给启动命令
指定用于桌面会话启动的 xstartup 脚本。“null”表示使用默认值，尤其是在 Ubuntu 系统中。

选项

当不加任何参数时，命令显示 vncsub.yaml 中配置的参数:

```
Usage: vncsub [-geometry MxN] [-vnc vncserver_dir] [-vglrun|-novglrun] [-xstartup␣
↪script] [scheduler_submit_options] [-command] [desktop] job_command ...
Common parameters:
  Web forwarding is enabled.
  Session geometry: -geometry 1900x1060
  VNC desktop autokill: Yes
  vncserver directory: /cubetop/etc/skyformcrv/bin
  VNC pubweb addresses:
  - dev: 192.168.10.10
  - node01: 11.11.11.12
  Graphics compression (0-9): 5
  CRV port: 16000
```

vncsub/dcvsub 可用以下参数:

csub_arguments
指定作业提交选项。不支持交互式作业和暂停选项。例如，LSF 中的选项 -I -Is -Ip -K -XF 不受支持。

-geometry ***MxN***
指定 VNC 会话的大小（以像素为单位）。默认值为 1900x1060。

-vnc ***vncserver_dir***
指定目录包含可执行文件 vncserver、vncpasswd 和 Xvnc 的路径。默认使用 AIP 嵌入式 TurboVNC。如果使用系统 TigerVNC，请指定 vnc: /usr/bin **-xstartup** *file_path* 指定桌面会话的 xstartup 脚本。“null” 表示使用默认启动脚本。这在 Ubuntu 上非常有用。

-debug
显示额外的调试信息。

-pp yes|no
指定是否启用 Web 流量路由。这将覆盖 vncsub.yaml 中的配置。

-command
默认情况下，该工具假定调度程序作业提交选项具有值，例如 -n 3。如果最后一个调度程序选项没有值，请使用 -command 指示此作业命令之后的任何内容。

desktop
以 VNC/DCV 会话的形式启动完整桌面。环境变量 DESKTOPAPP 指定多个桌面应用程序定义文件路径之一。指定该变量后，这些应用程序将与桌面一起启动。例如：DESKTOPAPP=” /usr/share/applications/xterm.desktop /usr/share/applications/firefox.desktop”。

job_command ⋯
带有要运行的参数的作业命令。

如果指定了选项 “desktop”，但未指定任何作业命令，则 VNC/DCV 会话中仅启动桌面。

如果同时指定了选项 “desktop” 和作业命令，则作业命令将在 VNC/DCV 桌面会话中启动。作业命令完成后，桌面和整个作业都将保留。用户需要注销桌面或手动终止作业才能结束整个作业。

如果未指定选项 “desktop”，则 VNC/DCV 会话仅包含作业命令。没有其他桌面组件。当作业命令完成后，VNC/DCV 会话将被销毁，整个作业也随之完成。

示例

% **vncsub 桌面 xterm**

在 VNC 桌面中提交 xterm 作业。退出 xterm 可保持桌面会话运行。

% **vncsub xterm**

提交 xterm 作业。退出 xterm 可结束 VNC 会话。

% **vncsub /usr/bin/startkde**

提交 KDE 桌面。

% **dcvsub -ENV DESKTOPAPP=/usr/share/applications/xterm.desktop desktop**

提交正在运行 xterm 的 GHOME 桌面。

诊断

对于 vncsub，作业输出仅显示在 VNC 日志中。如果作业运行失败，请检查 $HOME/.vnc/*job_exec_host:sessionID*.log 进行故障排除。

5.2 配置文件

5.2.1 aiprestd.yaml

配置文件

aiprestd.yaml

概述

aiprestd 是一个提供 RESTFul API 的 HTTP 服务器。aiprestd.yaml 文件提供了 aiprestd 服务的配置信息。

该 yaml 文件采用严格的 YAML 格式。包含特殊字符的文本字符串需要用单引号括起来（例如 “xxx”）。

描述

在启动 aiprestd 服务之前，您需要检查 aiprest.yaml 文件，并确定 aiprestd 服务中使用的这些参数。

aiprestd 会读取配置文件 aiprestd.yaml。默认情况下，aiprestd.yaml 文件位于调度程序的配置目录中，例如：/opt/skyformai/etc

该配置文件支持以下参数：

log_level: 致命错误 | 警告 | 信息 | 调试
: aiprestd 会根据日志级别将 RESTFul API 的行为记录到日志文件中。默认情况下，aiprestd.log.HOST 位于调度程序的日志目录中，例如 /opt/skyformai/log。

  默认值为 info。

timeout: 分钟
: aiprestd 使用 JWT 令牌来控制 RESTFul API 的访问。访问 RESTFul API 之前，您必须通过调用 /login API 生成一个令牌。然后在发送业务请求时将此令牌放入 Authorization 标头中。

  我们设置令牌将在指定的超时间隔内过期。

  单位为分钟，默认值为 30。

web_url_path: 路径
: 此参数用于 aiprestd 组织 Web 上下文路径。默认情况下，web_url_path 为 /，Web 上下文路径将为/aip/v9.2。如果 web_url_path 为 /test，则 Web 上下文路径将为 /test/aip/v9.2。

  默认值为 /。

ssl: *0* | *1*
: 指定 Web 服务器是否使用 SSL。1 = 使用，0 = 不使用（默认）。

http_port: *port*
: 指定 aiprestd REST 服务 HTTP 端口（如果端口号不是默认的 8088）。

https_port: *port*
: 指定 aiprestd REST 服务 HTTPS 端口（如果端口号不是默认的 8043）。

cert: *path*
指定 SSL 证书文件路径。默认值为 none。

key: *path*
指定 SSL 密钥文件路径。默认值为 none。

示例

```
cat /opt/skyformai/etc/aiprestd.yaml
log_level: error
timeout: 120
web_url_path: /skyform
ssl: 1
https_port: 9443
cert: /opt/cert/server.crt
key: /opt/cert/server.key
```

诊断

如果 API 运行异常，请检查调度程序日志目录中的日志文件 aiprestd.log.HOST 进行故障排除。

您也可以运行 systemctl status aiprestd 命令来查看守护进程状态和日志。

5.2.2 cb.acct

数据文件

cb.acct - AIP 作业统计文件

描述

AIP 调度程序 CBSCHED 会为每个作业的完成或失败生成一条记录。该记录会附加到作业日志文件 cb.acct 中。该文件位于 CB_SHAREDIR/data 目录下，其中 CB_SHAREDIR 一般是/opt/skyformai/work/data。作业日志文件是一个 ASCII 文件，每行一条记录。记录的各个字段以空格分隔。如果某个字段的值不可用，则记录字符串为 “”，记录时间和数字为 0，记录资源使用情况为 -1。

cb.acct 文件为当前月份中的作业记录。cb.data.1 为上个月的作业记录，cb.data.2 为上上的月的记录，以此类推。缺省情况下，cb.data 文件可以有无数个。总个数由 cb.yaml 配置文件中的 max_cbacct_num 参数决定。

字段

每个作业记录的字段按以下顺序排列：

- **事件类型（%s）**
  始终为 “ JFIN ”
- **版本号（%s）**
  日志文件格式的版本号，如 10250。
- **记录时间 (%d)**
  记录事件的时间（Linux 时间戳）

- **作业号 (%d)**
  作业号 ID
- **用户 ID (%d)**
  提交者的 Linux 用户 ID
- **选项（%d）**
  作业处理的位标志（内部使用）
- **处理器数量（%d）**
  最初请求执行的处理器数量
- **提交时间 (%d)**
  作业提交时间
- **开始时间 (%d)**
  作业开始时间- 工作应在此时间或之后开始运行
- **结束时间 (%d)**
  作业终止期限- 工作应在此时间终止
- **开始运行时间 (%d) -**
  作业调度时间- 作业调度执行的时间
- **用户名 (%s)**
  提交者的用户名
- **队列（%s）**
  提交作业的作业队列的名称
- **作业资源需求（%s）**
  用户提交作业时指定的资源需求
- **依赖条件（%s）**
  用户提交作业时指定的作业依赖条件
- **预执行命令（%s）**
  用户指定的预执行命令（pre_exec）
- **来自主机（%s）**
  提交主机名
- **cwd (%s)**
  当前工作目录
- **输入文件 (%s)**
  标准输入文件名
- **输出文件（%s）**
  标准输出文件名
- **错误文件（%s）**
  标准错误输出文件名
- **作业文件（%s）**
  作业脚本文件名。这是 AIP 为作业生成的内部使用的作业启动脚本
- **请求主机数量 (%d)**
  限制作业调度的主机名数量
- **请求主机名（%s）**
  作业调度受限的主机名列表（每个主机名占%s）；如果最后一个字段值为 0，则为空白。如果有多个主机名，则每个附加主机名将在其自己的字段中返回

- **作业运行作业槽数量 (%d)**
  用于执行的作业槽数量
- **运行作业槽主机名（%s）**
  运行作业槽所在的主机名列表（每个主机名占%s）；如果上一个字段值为 0，则为空白
- **jStatus (%d)**
  作业状态。数字 32 表示 EXIT，64 表示 DONE
- **主机 CPU 因子 (%f)**
  第一个执行主机的 CPU 因子
- **作业名 (%s)**
  作业名称
- **命令（%s）**
  完成用户指定的作业命令
- **cbRusage**
  以下字段包含作业的资源使用情况信息。如果某些字段的值不可用（由于作业中止或操作系统差异），则会记录-1。时间以秒为单位，大小以 KB 为单位。

  ru_utime (%f)
  用户使用 CPU 时间

  ru_stime (%f)
  所用系统 CPU 时间

  ru_maxrss (%d)
  最大共享文本大小

  ru_ixrss (%d)
  共享文本大小随时间变化的积分（以千字节秒为单位）

  ru_ismrss (%d)
  共享内存大小随时间变化的积分（仅在 Ultrix 上有效）

  ru_idrss (%d)
  未共享数据大小随时间变化的积分

  ru_isrss (%d)
  非共享堆栈大小随时间变化的积分

  ru_minflt (%d)
  页面回收次数

  ru_magflt (%d)
  缺页次数

  ru_nswap (%d)
  进程被换出的次数

  ru_inblock (%d)
  块输入操作数

  ru_oublock (%d)
  块输出操作数

  ru_ioch (%d)
  读取和写入的字符数（仅在 HP-UX 上有效）

  ru_msgsnd (%d)
  已发送的 System V IPC 消息数

ru_msgrcv (%d)
收到的消息数量

ru_nsignals（%d）
接收信号数量

ru_nvcsw (%d)
自愿上下文切换次数

ru_nivcsw (%d)
非自愿上下文切换次数

ru_exutime (%d)
用户使用的准确时间（Linux 上无效）

- **邮件用户（%s）**
  收到工作相关邮件的用户的姓名
- **项目名称（%s）**
  项目名称
- **退出状态（%d）**
  Linux 作业的退出吗
- **最大处理器数量 (%d)**
  为作业指定的最大处理器数量
- **登录 shell（%s）**
  用于工作的登录 shell
- **idx（%d）**
  作业阵列索引
- **最大 RMem（%d）**
  作业中所有进程的最大驻留内存使用量（以 KB 为单位）
- **最大 RSwap (%d)**
  作业中所有进程的最大虚拟内存使用量（以 KB 为单位）
- **inFileSpool（%s）**
  假脱机输入文件
- **命令池（%s）**
  假脱机命令文件
- **运行时间（%d）**
  作业运行时间（以秒为单位）。运行时间是作业结束时间和开始时间之间的时长减去作业暂停的时间。
- **平均内存 (%d)**
  作业运行期间的平均内存使用量（以千字节为单位）。
- **运行时间限制（%d）**
  作业运行限制（以秒为单位）。
- **作业描述 (%s)**
  提交作业时指定的作业描述。如果使用命令“aip j r”提交作业，则作业描述为 JSON 格式的作业规范。
- **任务数量（%d）**
  通过 runtask 启动的作业的远程任务数（runtask，rsh，blaunch 的任务数）。
- **任务信息**
  远程任务信息（如果 numTasks > 0）。

- **应用名（%s）**
  在作业提交中通过-A 或 -app 选项指定的应用程序名称。
- **资源分配（%s）**
  调度程序以 JSON 格式分配作业资源。
- **ugroup（%s）**
  该作业所属的用户组。
- **gpu_reserved（%d）**
  作业保留的 GPU 数量。
- **最大 ngpus (%d)**
  作业使用的最大 GPU 数量。
- **req_memmb (%d)**
  作业请求的或队列中配置的内存量（以 MB 为单位）。
- **组合请求 (%s)**
  作业的综合资源需求，合并资源需求。

5.2.3 cb.yaml

配置文件

cb.yaml

概要

cb.yaml 文件是 SkyForm AI 服务的强制配置。它至少定义了集群结构和队列结构。每个部分都是必需的。 该文件采用严格的 YAML 格式。包含特殊字符的文本字符串需要用单引号引起来（例如‘xxx’）。

内容

- cluster 部分
- general 部分
- usergroups 部分
- users 部分
- hostgroups 部分
- queues 部分
- limits 部分
- apps 部分
- power 部分
- scale 部分

参数

cluster 部分

集群部分包含控制 cbls、cbexe、cbsched 和 cbjm 的集群配置参数。

additional_ports:

默认情况下，AIP 使用随机端口跨主机复制文件（crcp 和 csub -f）。每次并发复制都需要在源主机和目标主机上使用唯一的端口。如果主服务器和计算机服务器之间有防火墙，则需要开放防火墙上的一系列端口才能使 AIP 正常工作。

AIP 服务需要 16322 - 16329 之间的端口：

- cbls UDP: 16322
- cbexe TCP: 16323
- cbjm TCP: 16325
- cbsched TCP: 16324
- cbcrond TCP: 16326
- jservice TCP: 16329

AIP 个每个作业槽（一台主机上的作业槽数一般等于主机上总的 CPU 核数）分配的端口为 16331 + 作业槽 (0 ~ n)。

此参数指定的其他端口用于文件复制和 cview 命令。端口范围取决于 cview 和 crcp 的并发操作数。例如：“additional_ports: 16001-16300”，即 300 个端口。

例子:

```
additional_ports: 16001-16300
```

默认值：未定义

参数影响的 AIP daemon：cbls, cbexe, cbjm, cbsched

administrators:

administrators 参数是必需的。它定义了集群管理员用户名列表。至少需要定义一个用户。

列表中的第一位用户是主管理员。

例子:

```
administrators:
- cadmin
- root
```

参数影响的 AIP daemon：cbls，cbsched，影响的目录/opt/skyformai/work，这个目录的必须由第一个管理员拥有。

allow_docker_run_change_user:

默认情况下，Docker 容器以提交作业的用户身份运行。这是为了确保 Docker 容器不会将其权限提升为 root 用户，从而避免裸机环境中的安全问题。

如果此参数的值为“y”，则允许用户在“csub -do”选项中添加“–user uid:gid”来指定在 Docker 容器作业内运行的用户 ID。

例子:

```
allow_docker_run_change_user: yes
csub -do “--user root：root” -di centos:7 python3 myprog.py
```

默认值：no

参数影响的 AIP daemon：cbjm，cbexe

allowed_dynamic_cidrs:

默认情况下，AIP 不允许动态主机加入集群。通过指定此参数和 CIDR 列表，AIP 将允许 IP 地址属于该 CIDR 的主机动态加入集群。

例如:

```
allowed_dynamic_cidrs:
- 10.210.0.0/16
- 10.211.2.0/24
- 10.211.3.3/32
- 10.211.3.4/32
```

默认值：无，不允许动态主机

参数影响的 AIP daemon：cbls

define_ncpus:

如果已定义，则将每台主机上的 CPU 总数反映如下：“cores”表示主机上的核心总数，“threads”表示主机上超线程的总数。此参数会被主机级别参数“define_ncpus”覆盖。

默认值：cores

参数影响的 AIP daemon：cbls, cbjm

dynamic_host_timeout:

指定动态主机不可用后自动删除的时间间隔（以分钟为单位）。指定值-1 可禁用动态主机的自动删除功能。

默认值：5（分钟）

参数影响的 AIP daemon：cbls

enterprise:

如果设置为“y”，则禁用 AIP 免费版本。在此模式下，如果密钥无效或缺失，调度程序将退出并显示错误日志。

默认值：no

参数影响的 AIP daemon：cbls, cbsched

env_prefix:

指定 AIP 环境变量的前缀。

AIP 会为作业设置一些以“CB”为前缀的环境变量（例如 CB_JOBID）。在某些情况下，前缀“CB”已被作业的其他环境变量使用。请使用其他值（例如“AIP”）覆盖前缀“CB”，这样 CB_JOBID 就会被 AIP_JOBID 覆盖。

修改 env_prefix 参数时始终运行 csadmin reconfig。

默认值：无，即不覆盖前缀“CB”。

参数影响的 AIP daemon：cbsched, cbjm

gpu_action_cmd:

如果 AIP 检测到任何 GPU 多函数调用，它会调用一个外部程序，该程序的路径在 gpu_action_cmd 中指定。此命令以 root 身份运行，无需设置任何环境变量。

如果未指定此参数，则默认情况下，当检测到 GPU 出现故障时将不会采取任何措施。

该程序将使用参数“miss”调用 ($gpu_action_cmd miss)。当所有 GPU 都恢复后，该程序也会使用参数“full”调用 ($gpu_action_cmd full)。

参数影响的 AIP daemon：cbls

host_inactivity_limit:

表示 load_interval 倍数的整数，它控制计算节点将其负载信息发送到主节点的最大时间以及主节点向所有计算节点发送心跳消息的频率。

计算节点可以在 load_interval 到 (host_inactivity_limit - 2) * load_interval 秒之间的任何时间发送其负载信息。主节点将至少每 load_interval * host_inactivity_limit 秒向每个主机发送一次主节点公告。

默认值：3

参数影响的 AIP daemon：cbls

job_rusage_override:

默认情况下，当在 queue: resspec 中指定了队列级别的 rusage 部分时，作业级别的 rusage 部分将被忽略。将此参数设置为“y”后，当作业在 -R 选项中指定了 rusage 部分时，队列级别的 rusage 部分将被忽略。

默认值：n。不覆盖。队列级别使用部分始终有效。

参数影响的 AIP daemon：cbsched

load_interval:

指定主控主机与所有其他主机之间的负载交换间隔（以秒为单位）。该值不能小于 5。在极其繁忙的主机或网络中，或者在包含大量主机的集群中，负载可能会干扰主控主机与所有其他主机之间的周期性通信。将 load_interval 设置为较长的 值可以降低网络负载并略微提高可靠性，但代价是响应动态负载变化的速度会变慢。

默认值：5（秒）

参数影响的 AIP daemon：cbls

localdir:

启用调度程序作业数据的复制。指定仅存在于第一个 AIP Master 主机上的本地目录路径。此目录用于保存调度程序作业数据和会计文件的主要副本。副本将存储在 AIP_top/work 中。

修改 localdir 参数时始终运行 csadmin reconfig。

默认值：无，即不使用重复的数据副本。所有数据都存储在 AIP_top/work 目录中。

参数影响的 AIP daemon：cbsched

logdir:

AIP 守护进程日志（cbcrond 除外）的目录。默认情况下，AIP 守护进程日志存储在 AIP_top/log 中。对于大型集群（超过 1000 台主机），建议使用此参数将守护进程日志存储在 /var/log/aip 中。

默认值：/opt/skyformai/log

参数影响的 AIP daemon：cbls, cbexe, cbjm, cbsched, jservice, cbcrond

lsf_compatible:

如果此参数的值为“y”或“yes”，则作业输出将遵循 LSF 样式，包含作业执行和资源使用情况信息。作业内的环境变量都改成以 LSB_ 开头的，等同与设置了“env_prefix: LSB”。默认情况下，此参数的值为“n”，即作业输出不包含额外数据。

默认值：no

参数影响的 AIP daemon：cbjm

master_inactivity_limit: 一个整数，表示 load_interval 的倍数。如果计算节点在 (host_inactivity_limit + host_number * master_inactivity_limit) * load_interval 秒后仍未收到上一主节点的消息，它将尝试成为主节点，其中 host_number 是主机在 hosts 部分中的位置。主节点的 host_number 为 0。

默认值：2

参数影响的 AIP daemon：cbls

master_list:

指定以空格（' '）分隔的主候选主机名。

默认值：如果未指定此参数，则 hosts 部分中的前 3 个主机为 Master 候选。如果主机总数小于 3，则默认所有主机均为 Master 候选。

参数影响的 AIP daemon：cbls, cbsched, cbjm

mem_enforce_minfree:

指定此参数时，仅当作业执行主机的可用内存低于此值（以 MB 为单位）时才会强制执行作业内存限制。

默认值：未指定。默认情况下，无论主机可用内存有多少，都会强制执行作业内存限制。

例如：mem_enforce_minfree: 100。使用 csub -M 提交的作业带有内存限制，或者作业具有队列级别内存限制。仅当作业首次执行主机的可用内存小于 100MB 时，才会强制执行作业内存限制。否则，即使作业的内存使用量大于内存限制，作业仍会继续运行。

参数影响的 AIP daemon：cbsched, cbjm

name:

定义集群的名称。如果缺失，则默认集群名称为 “aip”。例如:

```
name: aip
```

参数影响的 AIP daemon：cbls

no_resreq_check:

默认情况下，作业提交会检查用户指定的资源需求，如果用户请求的资源量超过集群的资源量，则会拒绝提交。这要求作业提交查询主服务器以获取集群中所有资源的信息。在非常大的集群环境中，这将花费很长时间（超过 10 秒），从而导致作业提交速度变慢。要禁用此检查，请将此参数指定为任意值，例如 no_resreq_check: nocheck。

默认值：未指定，即 csub 检查资源需求。

参数影响的 AIP daemon：cbsched

runaway_job_grace_period:

当作业运行时间达到上限时，cbjm 会执行终止作业的操作，默认情况下，cbjm 首先发送 SIGUSR2 信号。在 runaway_job_grace_period 之后，它会发送 SIGINT 信号、SIGTERM 信号，最后发送 SIGKILL 信号。

cbjm 启动时，该值显示在 cbjm.log 中。

默认值：10 (分钟)

参数影响的 AIP daemon：cbjm

shared_fs:

默认情况下，AIP 会在主服务器上调用 Linux 的“df”命令来监控已挂载文件系统的可用空间。有时，它无法识别共享文件系统，例如 GPFS。在这种情况下，管理员可以配置此参数来告诉 AIP 需要监控哪个文件系统。

例如：shared_fs: gpfs1

可以通过指定以空格 (’‘) 分隔的文件系统字符串来监控多个文件系统。指定此参数将覆盖 AIP 识别共享文件系统（文件系统名称包含列字符 (:)）的默认规则。

默认：识别包含列字符（:）的共享文件系统名称（如 NFS）。

参数影响的 AIP daemon：cbls

shared_fs_interval:

默认情况下，系统每 5 秒检查一次共享文件系统。这可能会增加系统负载。要降低负载，请将 shared_fs_interval 指定为更大的秒数。

默认值：5 秒

参数影响的 AIP daemon：cbls

sshcontrol:

当使用–sshcontrol 设置主机（host-setup）时，会安装一个 pam 模块来控制用户何时可以 ssh 到主机。

- 当此参数的值为“yes”时，用户可以在
  1. 主机属于专用队列，并且用户可以使用专用队列，或者
  2. 用户在主机上有一个作业，并且该用户从运行该作业的主机 SSH 连接到运行同一作业的另一台主机（即并行作业的主机之间的 SSH）。不允许从同一主机发起 SSH 连接。

  所有其他 ssh 登录均被阻止。
- 如果该参数的值为“none”，则 ssh 控制被禁用。
- 如果此参数的值为“job”，则用户可以在以下情况下 ssh 到主机：
  1. 主机属于专用队列，并且用户可以使用专用队列，或者
  2. 无论 ssh 来自哪里，用户在主机上都有作业。
- 在所有情况下，除非被系统阻止，否则始终允许 root 和 AIP 管理员使用 ssh。

默认值：无（禁用 ssh 控制）

参数影响的 AIP daemon：cbsched

resources:

资源子节是可选的。它定义了要附加到集群中每个主机的标签类型资源列表。定义完成后，用户可以通过在资源规范中指定资源标签来运行作业。调度程序将选择带有关联标签的主机作为作业执行候选。

每个资源标签都具有以下属性（变量）：

name:
标签的名称。这是必填项。

description:
标签的描述。这是可选的。

type:
类型的值应为“tag”标签，或“number”数值。

direction:
资源数值的升降。描述数值型资源的多少属性，如可用内存是数值越大资源越多 (decrease)，而网络流量是数值越大资源越少 (increase)。

assign:
指定型资源。取值为“y”或“n”，缺省为 n。有些资源如网络带宽或内存在分配时没有指定的单位，而 GPU 和端口在分配时由调度器指定具体某个资源，以免各个作业间互相发生冲突。

slotresource:
作业槽资源。取值为“y”或“n”，缺省为 n。即每个任务需要分配一份资源，如内存，GPU 等，而有些资源是作业级的，即不管一个作业有多少个任务，都只需分配一份资源，如存储容量。

instances:
定义共享资源的主机。格式为：instance instance …。instance 的格式为 [all|default|others|host1 …]

对于静态资源，必须在此处定义资源总量。动态资源（如用 RESS 获取的资源）不能定义资源总量。

instance 是共享资源实例的主机名列表。可以为实例指定保留字 all、others 和 default：

- all: 表示整个集群中只有一个资源实例，并且此资源由所有主机共享。使用 not 运算符 (~) 将主机从 all 规范中排除。例如：(2@[all ~host3 ~host4]) 表示集群中由 host1 host2 …hostn 组成的所有服务器主机共享 2 个资源单元，host3 和 host4 除外。如果您拥有大型集群但只想排除少数主机，则此方法非常有用。

  规范中必须使用括号。not 运算符只能与 all 关键字一起使用。它与 others 和 default 关键字一起使用无效。

- others: 表示字段中未明确列出的其余服务器主机构成了该资源的一个实例。例如：2@[host1] 4@[others] 表示 Apple 主机上有 2 个资源单元，所有其他主机共享 4 个资源单元。

- default——表示集群中每个主机上都有一个资源实例，即资源实际上不共享，并且是每个主机的本地资源。

interval:
动态资源（RESS 提供）的采样间隔，单位为秒。最小值为 5 秒。

资源部分中的变量与 SkyForm AI 资源传感器 RESS 中使用的资源变量基本相同。更多详细信息，请参阅*ress*

例子:

```
- name: linux7                          # 标签资源
  description: CentOS 7
  type: tag
- name: proxy                           # 静态数值型资源
  description: Number of ports configured in Proxy
  type: number
```

(下页继续)

(续上页)

```
  direction: decrease
  assign: yes
  slotresource: no
  instances: "1000@[all]"
- name: localdisk                    # 动态数值型资源，由RESS更新资源值
  description: Available local disk in GB
  type: number
  direction: decrease
  instances: "[default]"
  interval: 15
```

参数影响的 AIP daemon：cbls

hosts:

hosts 子节是必需的。它定义了属于集群的主机列表。

每个主机具有以下属性（变量）：

name:

指定主机名，该名称必须在系统中可解析。这是必需的。该名称可以采用类似 “node[0001-0100]” 的值，系统会在内部将其扩展为 100 个主机。

maxslots:

指定主机上可用的最大作业槽位数量。此项为可选。

该值可以是数字或单词 “cpu”。如果未定义，则表示主机上的最大作业槽位数量不受限制。如果定义了单词 “cpu”，则表示主机上的最大作业槽位数量等于主机上安装的 CPU 核心数。这还可以将作业进程与 CPU 核心绑定，以确保作业执行性能。对于跨多台主机运行的并行作业，CPU 绑定仅在第一台执行主机上启用。

attr:

其值为 “server”、“client” 或 “remote”。“server” 表示主机运行 AIP 服务并可以运行作业。“client” 表示主机未运行 AIP 服务，或者 AIP 服务被忽略，主机无法运行作业。“remote” 表示主机是服务器，但不运行 AIP 服务。它要求 *aip_top* /sbin 中存在可执行文件 cbls.remote。AIP 守护进程从 clbs.remote 的输出中获取负载信息。当作业被调度到主机时，该作业会被调度到主服务器。需要配置作业启动器以远程启动作业到主机。详情请参阅*cbls.remote* 。

默认值：server

cpubind：

当指定“maxslots: cpu”时，是否启用 CPU 绑定（取值“y”或“n”）。如果未指定，则当 maxslots 等于主机 CPU 核心数时启用 CPU 绑定。否则，禁用 CPU 绑定。当 maxlots 不等于主机 CPU 核心数时，此参数不会覆盖行为。

默认：遵循“默认”主机。如果未指定，则使用“默认”主机。

define_ncpus：

它采用“cores”或“threads”的值。这将覆盖集群范围的参数“define_ncpus”。

tags:

与主机关联的标签资源列表。此项为可选。

例子:

```
tags:
-  linux7
-  compute
```

默认值：未指定

thresholds:

主机的调度阈值列表。阈值格式为：“load_namesched_threshold/stop_threshold”。

load_name 的值为“mem”、“swp”和“ut”（CPU 利用率在 0.0 - 1.0 范围内）。sched_threshold 和 stop_threshold 是浮点值。当负载达到“sched_threshold”时，调度程序将停止向主机调度更多作业。当负载达到“stop_threshold”时，调度程序将停止在主机上运行作业。

对于“mem”、“swp”、“tmp”，默认单位为 MB。您可以在数值后添加字母“G”将单位更改为 MB，或添加字母“T”将单位更改为 TB。

负载索引名称与“cinfo”的输出相同：“r15s”、“r1m”、“r15m”、“ut”、“pg”、“io”、“up”、“it”、“tmp”、“swp”或“mem”。

该阈值支持多个时间窗口，每个值的格式为：“sched_threshold1/stop_threshold1 [weekday:] 小时 [: 分钟]-[weekday:] 小时 [: 分钟] sched_threshold2/stop_threshold2 [weekday:] 小时 [: 分钟]-[weekday:] 小时 [: 分钟]”。

或者格式可以是：“sched_threshold/stop_threshold default (weekday1-weekday2)hour:minute-hour:minute;(weekday3,weekday4,week5)hour:minute-hour:minute”

日期数字代表星期几，类似于 crontab，即 0 或 7 代表星期日，1 代表星期一，等等。

示例 1:

```
thresholds:
- "ut 0.7/0.9"
- "mem 250/200"
```

在上面的例子中，作业调度条件是“mem >= 250 && ut <= 0.7”，作业停止（挂起）条件是“mem < 200 || ut > 0.9”。

示例 2:

```
- "mem1G/0.5G 1:8:00-5:20:00 0.6G/0.3G 5:20:00-1:8:00"
```

上面的例子意味着阈值为周一早上 8 点到周五晚上 8 点 1G/0.5G，周五晚上 8 点到周一早上 8 点 0.6G/0.3G。

示例 3:

```
- "mem 100/50default50/30 (1-5)8:00-20:00,(0,6)6:00-22:00"
```

以上示例表示，工作日 8:00 - 20:00 以及周末 6:00 - 22:00 的阈值分别为 50MB/30MB。除此以外，阈值（默认值）为 100MB/50MB。

proxy:

远程主机的代理主机名。提交到远程主机的作业将被分派到指定的代理主机。

例子:

```
proxy:host1
```

默认：对于远程主机，代理主机是主主机。

umap:

如果作业执行主机的用户空间与主服务器不同，则可以在此处指定可用于运行作业的用户列表，以供集群动态使用。命令 cumap <hostname> 会显示该主机上的动态用户映射。

例子:

```
umap: u[001-100] u123
```

默认值：主机未启用用户映射

map_to_single_user:

默认情况下，远程主机上的用户映射不允许不同的本地用户同时映射到同一个远程用户。如果远程用户数量受限，请将值设置为 1，以禁用上述控制，并将无限数量的本地用户映射到单个远程用户。

例子:

```
map_to_single_user: 1
```

参数影响的 AIP daemon：cbls, cbexe, cbsched, cbjm

clients:

允许 AIP 接受浮动客户端主机。值为“yes”。

例子:

```
clients: yes
```

在客户端主机上，需要以与服务器相同的方式安装 AIP。但必须停止并禁用 aip 服务。配置文件 cb.yaml 必须与 AIP 服务器使用的 cb.yaml 相同。

默认值：no，即不允许浮动客户端。

参数影响的 AIP daemon：cbls

general 部分

general 部分定义了调度程序的通用属性。更改此部分中的任何参数都需要重新配置调度程序才能生效。命令为：“csadmin reconfig”。

参数影响的 AIP daemon：cbsched

default_queues:

当用户在作业运行中未指定队列名称时，指定默认队列名称。此项为必填项。

当用户提交作业而未明确指定队列时，AIP 会将作业放入此列表中第一个满足作业规范的队列中，但要遵守其他限制，例如请求的主机、队列状态、允许的用户等。

例如：default_queues: medium foruser1 foruser2

memperiod:

指定已完成作业的信息在调度程序内存中保留的时间段（以秒为单位）。此项为可选。

将已完成的作业保留一小时的示例：memperiod：8000

默认值：4000（秒）

cgroup:

Linux cgroup 集成，用于强制执行 CPU 使用率、内存使用率和 GPU 使用率。值为 “cpu”、“mem”、“gpu”、“acct” 或以上任意词语的组合。默认情况下，作业的 cgroup 资源不强制执行。

例如：cgroup: acct cpu gpu

现代操作系统，Ubuntu 22、EL 9 及以上，cgroup 使用 v2。“gpu” cgroup 控制使用 cgroup v1。

当 cgroup “acct” 启用时，AIP 使用 CPU cgroup 来控制作业进程。“cpu” 不仅启用 “acct”，还会强制执行作业的 CPU 使用率。我们强烈建议启用 “cgroup: acct”，以防止作业进程逃逸。

AIP 在 Linux 内核 2.x（例如 CentOS 6）或更低版本上不支持 Cgroup。

默认：cgroup 未启用。

rootcanrun:

默认情况下，root 用户可以运行作业。要禁用此功能，请定义“rootcanrun: no”。

默认值：yes

egroup_update_interval:

定义自动同步系统用户组成员的间隔，单位秒。系统用户组是 usergroups 里与系统中同名的用户组，但组成员为”@system”的哪些用户组。这些组的成员由调度器从系统中定期同步。

默认值：3600（1 小时）

最小值：600（10 分钟）

idle_action_trigger_duration:

指定队列级别作业空闲操作运行的时间间隔（以分钟为单位）。此参数与队列部分中的“job_idle”一起使用。

例子:

```
idle_action_trigger_duration: 2
```

默认值：1（分钟）

gpu_share_spread:

默认情况下，当多个作业共享一个 GPU 时，即使用 -R “need[gpu=0.x]”参数提交作业时，调度程序会将作业打包到具有多个 GPU 的主机上尽可能少的 GPU 上。通过将此参数指定为“yes”，调度程序会在安装了多个 GPU 的主机上分散 GPU 分配比例（即作业所需的 GPU 少于一个）。例如，如果第一个作业计划使用 0.5 个 GPU #0，则第二个作业将被计划使用 #GPU1，即使第二个作业仅需要 0.5 个 GPU。

默认值：no

job_accept_interval:

将一个作业调度到主机后，在将第二个作业调度到同一主机之前等待的调度轮次数。默认情况下，主机在每个作业调度间隔内可以接受多个作业。由于主机上可运行的作业总数没有限制，因此可能会一次性将大量作业调度到主机。

如果该值设置为非零正整数，则意味着主机在调度一个作业后等待（job_accept_interval* sched_interval）秒，然后再调度第二个作业。

在队列级别（队列部分）设置的 job_accept_interval 将覆盖此部分中设置的 job_accept_interval 的值。

默认值：0

job_terminate_interval:

指定终止作业时发送 SIGINT、SIGTERM 和 SIGKILL 信号的时间间隔（以秒为单位）。作业终止时，系统会依次向该作业发送 SIGINT、SIGTERM 和 SIGKILL 信号，并在发送这些信号之间设置一个 job_terminate_interval 的休眠时间。这允许作业在必要时进行清理。

默认值：10（秒）

sched_interval:

调度器启动一个调度周期的时间间隔（以秒为单位）。在大型或繁忙的环境中，此值应该更长，例如 15。

默认值：4（秒）

jm_interval:

作业管理器每次更新作业状态和作业资源使用情况信息的时间间隔（以秒为单位）。在大型环境中，此数字应该更长，例如 30。

默认值：3（秒）

elastic_job:

通过提供“y”值来指定是否支持弹性作业。

默认值：no

estream_interval:

指定调度程序 CBSCHED 更新扩展流文件的间隔（以秒为单位）。这些文件将写入 **estream_dir** 目录中，缺省为 /tmp/.aipestream。它们包含最新的作业状态、队列状态、用户状态和主机状态。

默认值：30（秒）

最小值：10（秒）

estream_dir:

指定调度器扩展 stream 数据的目录。若修改了该参数，目录必须手工创立，并保证集群第一管理员有写的权限，否则 estream 不能正常工作，jservice 也不能正常工作。

默认值：调度器主机（master）上的/tmp/.aipestream

dedicated_queue_include_unavail

专属队列计量时，当专属队列中的某个主机 AIP 服务连不上时是否任然算全部使用。这个功能只对计量和计费有用。

默认值：no

mailonmaster:

值“yes”表示通知发送邮件或自定义程序从主控主机执行。“no”表示在作业的第一个执行主机上执行。

默认值：no

mailprog:

AIP 在作业开始和结束时用于发送电子邮件的邮件程序的路径和文件名。

这是 AIP 用于向用户发送系统消息的电子邮件程序。

当 AIP 需要向用户发送电子邮件时，它会调用此处定义的程序。您可以编写自己的自定义邮件程序，并将此参数设置为该程序的存储路径。

提供任何邮件程序的名称。为了方便起见，AIP 提供以下邮件程序：

- sendmail：支持 Linux 上的 sendmail 协议如果修改此参数，AIP 管理员必须在主机上运行 csadmin reconfig。
- 在 Linux 上：AIP 通常使用 /usr/lib/sendmail 作为邮件传输代理向用户发送邮件。AIP 使用以下参数调用“mailprog”：-oi -FAIP -f *job_user* mailto。

“mailprog”必须从标准输入读取邮件正文。邮件的结尾以文件结束符 (END-OF-FILE) 标记。任何能够接受参数和输入并正确投递邮件的程序或 Shell 脚本都可以使用。

任何用户都必须能够执行“mailprog”。

示例：mailprog: /serverA/tools/skyformai/bin/notify

默认：”:”，即不发送电子邮件

mailto:

当用户的作业完成或出现错误时，AIP 会向用户发送电子邮件；如果 AIP 系统出现严重错误，则会向 AIP 管理员发送电子邮件。默认情况下，AIP 会将邮件发送给运行守护进程的主机上提交作业的用户；这假设您的电子邮件系统会将邮件转发到中央邮箱。

“mailto”参数改变 AIP 使用的邮寄地址。“mailto”是用于构建邮寄地址的格式字符串。

常见格式有：

- “!U”- 邮件发送到本地提交用户的帐户名主机。如果找到子字符串“!U”，则将其替换为用户的帐户名。
- “!U@company_name.com”- 邮件发送给用户 @ company_name.com。
- “!U@!H”- 邮件发送至 用户 @submission_hostname 。子字符串!H 被替换为提交主机的名称。该值需要用引号引起来，因为字符“!”在 YAML 中具有特殊含义。所有其他字符（包括任何其他“!”）都会被准确复制。

如果修改了此参数，AIP 管理员必须通过执行命令“csadmin reconfig”重新配置调度程序。

mailsize_limit:

限制包含 AIP 作业输出信息的电子邮件的大小（以 KB 为单位）。

AIP 系统会将作业信息（例如 CPU、进程和内存使用情况、作业输出以及错误）通过电子邮件发送给提交用户帐户。某些作业可能会创建大量输出。为防止大型作业输出文件干扰您的邮件系统，请使用 “mailsize_limit” 设置包含作业信息的电子邮件的最大大小（以 KB 为单位）。请指定一个正整数。

如果作业输出电子邮件的大小超出 “mailsize_limit”，则输出将保存到 JOB_SPOOL_DIR 下的文件中；如果 JOB_SPOOL_DIR 未定义，则保存到默认的作业输出目录中。该电子邮件会告知用户作业输出的位置。

如果使用 **csub** 的**-o** 选项，则不会根据 “mailsize_limit” 检查作业输出的大小。

如果您使用 “mailprog” 参数指定的可以使用 CB_MAILSIZE 环境变量的自定义邮件程序，则无需配置 CB_MAILSIZE_LIMIT。

默认值：未启用。AIP 作业输出电子邮件的大小没有限制。

max_array_size:

用户在单次作业提交中可创建的作业阵列的最大索引值。该值必须介于 2 到 1,000,000,000 之间。

默认值：10000

max_cbacct_num:

work/data 目录中的最大 cb.acct 文件数量。

默认值：0，即无限制。

max_cbdata_num:

work/data 目录下 cb.data 文件的最大数量。设置为 0 或负数表示无限制。

默认值：1024

max_cbdata_job_num:

已从 cbsched 内存中清除的已完成作业的最大数量，以触发 cb.events 文件翻转。

当 cb.events 文件滚动更新时，所有已清理的作业将被写入新文件 cb.events.<timestamp>。新的 cb.events 包含 cbsched 内存中所有活动作业和已完成作业。

一旦达到限制，AIP 就会启动一个新的事件日志文件。旧的事件日志文件将保存为 cb.events.<timestamp>。事件日志记录将在新的 cb.events 文件中继续进行。

默认值：10000

max_jobclean:

定义调度器在每次从内存中清理已完成作业时最多清理的作业数。调度器在做清理时会停止对外的服务，如果过每次清理作业数太大，会影响对命令的响应。当每次清理的作业数到达这个最大值时，cbsched.*master*.log 中有一条清理所有时间的日志。

默认值：2000

最小值：10

max_jobid:

指定作业 ID 限制。作业 ID 限制是 AIP 分配的最大作业 ID，也是系统中作业的最大数量。

指定从 999,999 到 2,147,483,646 的任意整数（出于实际目的，指定小于该值的任何 9 位整数）。

默认值：2147483646

最小值：999999，最大值：2147483646

max_periodic_task_interval:

调度器运行非调度任务的最长间隔，如数据清理、输出扩展 stream、混动 cb.data 等。这个间隔的最小周期是 sched_interval （缺省为 4 秒）。这个参数定义最长时间，单位为秒。

默认值：600 （10 分钟）

最小值：10 （秒）

max_stream_records:

在写入由 max_stream_records 定义的记录数后，AIP 调度程序会写入一个 STREAM_END 事件，关闭文件 cb.stream.0，将其重命名为 cb.stream.1，并创建一个新的 cb.stream.0 来继续写入作业事件，之前的 cb.stream.1 将被覆盖。

默认值：100000

job_fail_close_host:

当一段时间内作业失败次数超过指定次数时，关闭主机。

语法：number_of_job_failes[time_period_minute]

第一个值是触发主机关闭的作业失败次数。第二个值是作业失败的测量时间段，以分钟为单位。默认情况下，如果未指定此参数，则不会测量任何作业失败。如果未指定 time_period_minute，则时间段为 5 分钟。

默认：未指定，即不启用该功能。

preemptable_resources:

用于启用基于资源的作业抢占的配置。只有在值（以空格分隔的资源名称列表）中定义的资源才能触发作业抢占。

当高优先级队列中的作业因其所需资源不可用而处于等待状态，且该资源在“preemptable_resources”中定义时，AIP 会将正在使用该资源的低优先级作业重新排队，直到资源可供高优先级作业运行。这通常用于 GPU 或应用程序许可证抢占。

要启用抢占策略，需要在高优先级队列配置中设置参数“preemption”。详情请参阅队列部分。

启用基于资源的作业抢占会自动禁用作业抢占暂停（“preemption_suspend”）。

例如：preemptable_resources: gpu

默认值：未指定可抢占资源。

preemption_suspend:

指定基于槽位的抢占行为。如果设置为“y”，则作业抢占会暂停被抢占的作业。如果设置为“n”或未设置（默认），则作业抢占会终止被抢占的作业，然后将其重新放入队列。

默认值：n，作业抢占将重新排队被抢占的作业。

user_view_alljobs:

如果值为“yes”，则普通用户可以查看所有作业信息、用户组信息、主机组信息、所有用户信息。

默认值：no

ugconf

它指定包含用户组部分的文件名。该文件应与 cb.yaml 位于同一目录中。目前仅支持“ug.yaml”文件名。

usergroup 部分

用户组部分指定用户和用户组的层次结构，可用于队列和限制部分。一个用户可以属于多个用户组。用户组可以嵌套，即一个组可以是另一个组的成员。

参数影响的 AIP daemon：cbsched

name:

指定用户组的名称。该名称不得与任何现有用户相同。由于此限制，使用 OS 用户组时请务必谨慎。

members:

指定属于该组的用户名。可以列出多个用户，每个用户之间用空格分隔。特殊术语“@system”表示该组是操作系统用户组，其成员信息应从操作系统用户数据库（例如 LDAP、NIS）中获取。

特殊术语“@external”表示组成员来自外部程序“/opt/skyformai/sbin/egroup”。egroup 由调度程序以“/opt/skyformai/sbin/egroup -u group_name”的格式执行。预期输出是以空格分隔的用户组成员列表。

成员定义也可以使用“all ~username”来定义除‘username’之外的所有用户。

administrator:

指定群组管理用户。该用户可以查看群组成员的作业信息。每个群组只能指定一名管理员。

默认值：该用户组无管理员

fairshare:

指定公平分享定义。

例如：fairshare:’[default,1]’

默认值：无公平分享

maxslots:

指定组的作业槽限制。

例如：maxslots: 10

默认值：对用户组无限制

usermaxpend:

指定组中每个用户的最大等待作业数。此项为可选。默认情况下，每个用户的等待作业数没有限制。

默认值：没有最大等待作业限制

usermaxslots:

指定组中每个用户的运行作业槽位限制。此项为可选。默认情况下，每个用户的运行作业槽位没有限制。

每位用户只能拥有一个职位槽位限制。如果用户属于多个用户组，则适用该用户所属的最后一个用户组的 usermaxslots 值。此限制也可能会被用户部分中指定的槽位限制所覆盖（见下文）。

默认值：每个用户无限制

maxjobs:

指定该组要运行的最大作业数。

默认值：无限制

maxgpus:

指定组一次可使用的最多 GPU 数量。

默认值：无限制

users 部分

users 部分指定用户作业槽位限制。如果同一用户在 usergroups 部分中指定了作业槽位限制，则该限制将被覆盖。在此部分中，值 -1 表示受限。

参数影响的 AIP daemon：cbsched

name:

指定用户名。保留字“default”表示该规范默认适用于所有用户，除非被特定用户条目覆盖。

maxslots:

指定用户运行作业槽的限制。

默认值：-1，即无限制

maxpend:

指定用户等待作业槽的限制。

默认值：-1，即无限制

hgconf

它指定包含 hostgroups 部分的文件名。该文件应与 cb.yaml 位于同一目录中。目前仅支持“hg.yaml”文件名。

hostgroups 部分

hostgroups 部分指定主机组，可用于队列和限制部分。一个主机可以属于多个主机组。主机组可以嵌套，即一个组可以是另一个组的成员。

参数影响的 AIP daemon：cbsched

name:

指定主机组的名称。该名称不能与此文件中配置的任何主机名相同。

members:

指定属于该组的主机名。可以列出多个主机名，每个主机名之间用空格分隔。

members 的值可以包含保留字“all”，表示读取 cb.yaml 时配置的所有主机，但不包括通过命令 caddhost 动态添加的主机。主机名前缀“~”表示“排除”。例如，“all ~h1”表示除主机/主机组 h1 之外的所有主机。之前定义的组名可以作为成员包含在“members”字段中。

queueconf

它指定包含队列部分的文件名。该文件应与 cb.yaml 位于同一目录中。目前仅支持“queue.yaml”文件名。

queues 部分

队列部分定义了集群的队列列表。每个队列都有一组属性。

参数影响的 AIP daemon：cbsched

name:

指定队列名称。这是必需的。

administrators:

队列管理员列表。队列管理员可以对队列中任何用户的作业执行操作。默认值为未定义，这意味着用户必须是集群管理员才能对队列中的作业进行操作。

赋值可以使用用户组名称。

默认值：无管理员

description:

指定队列的可选描述。

默认值：无描述

priority:

相对数字。数字越大，队列的优先级越高。此项为必填项。数字范围一般为 1-100。

maxslots:

队列中正在运行的作业的最大作业槽位数量。此项为可选。如果未定义，则没有运行作业槽位限制。

默认值：-1 无限制

fairshare:

指定队列的公平共享结构。如需均等共享，请使用文本“[default,1]”。此项为可选。

默认值：无公平份额

preemption:

指定一个或多个低优先级队列，这些队列之间用空格分隔，可以被抢占。拥有者抢占时可以抢占同有限的拥有者队列（见 ownership_hosts）。抢占可以在“general”部分中基于资源或作业槽位进行配置。有关详细信息，请参阅参数“preemptable_resources”和“preemption_suspend”。

默认值：未定义，即队列不抢占。

ownership_hosts:

指定队列拥有的一组主机（包括主机组）。将此参数与“preemption”参数结合使用，意味着如果这些拥有的主机被“preemption”参数指定的队列中的作业占用，则此队列中等待的作业将抢占“preemption”队列中的作业。

此外，最好在队列中指定主机偏好，以便优先拥有的主机来运行此队列中的作业。

例子:

```
- name: dev
  priority: 3
  preemption: testq
  ownership_hosts: hostgroup1
  hosts: hostgroup1+ others
```

在上面的例子中，dev 队列中的作业优先在 hostgroup1 的主机上运行。它拥有 hostgroup1 的所有主机。如果 testq 中的作业正在 hostgroup1 的任何主机上运行， 则 dev 队列中等待的作业将抢占 testq 中在 hostgroup1 主机上运行的作业，无论 testq 队列的优先级如何。

此功能可保证单个用户群的 SLA（服务水平协议）。

默认值：无所有权主机

resspec:

指定队列中所有作业的默认资源需求。此为可选参数。语法请参阅*cjobs*。-R 选项。与 csub -R 选项相同，多个部分之间以空格分隔。不支持多个资源需求（例如“4{mem>10} 5{rusage[gpu=1]}”）。

默认值：无资源要求

users:

指定哪些用户/用户组可以使用该队列。此项为可选。

默认：所有用户都可以使用该队列。

hosts:

指定队列中的作业可以运行在哪些主机/主机组上。此项为可选。

默认值：队列中的作业可以在集群中的所有主机上运行。

hostmaxslots:

指定队列在每台主机上运行作业的槽位限制。这是可选的。

默认值：队列主机没有运行作业槽限制。

usermaxslots:

指定每个用户的运行作业槽位限制。这是可选的。

默认值：每个用户没有运行作业槽限制。

rerunonhostfail:

如果值为“yes”，则当作业执行主机发生故障时，作业将自动重新运行。

默认值：no

rerunonexitcode:

指定一组作业退出代码，以空格分隔，用于自动重新运行作业。特殊值“!0”表示当作业的退出代码非零时，作业将自动重新运行。这通常用于服务作业。

默认值：作业在特定退出代码时不会重新排队

slotreservetime:

启用队列中等待的并行作业可以预留的处理器和内存预留时间（以分钟为单位）。在此时间之后，如果作业尚未积累足够的作业槽来启动，则会释放其所有预留的作业槽。

默认：未启用处理器预留。

cgroup:

指定队列里作业是否使用 cgroup 控制进程、CPU 使用、内存使用、和 GPU 使用。这个参数会覆盖系统级的 cgroup 参数（general: cgroup）。值为 acct、cpu、mem、gpu 的组合，多项之间用空格隔开。例子:

```
queues:
- name: medium
  priority: 2
  cgroup: cpu mem

general:
  cgroup: acct
```

这个例子里，系统级的 cgroup 是 acct，而队列 medium 中的作业只受 cpu 和 mem 的 cgroup 控制。

队列作业的 cgroup 控制缺省使用系统级设置的 cgroup 值。队列 cgroup 的定义覆盖系统级的。如果队列要去除系统级 cgroup 的控制，则 cgroup 的值为 **remove**。例子:

```
queues:
- name: medium      # 队列medium中的作业不受cgroup控制
  priority: 2
  cgroup: remove
- name: vnc         # 队列vnc中的作业继承系统中cgroup的控制
  priority: 2
general:
  cgroup: acct gpu
```

job_starter:

指定队列级作业包装程序的路径。作业启动器会在执行前为已提交的作业创建特定的环境。该参数的值可以是任何可用于启动作业的可执行文件（即，可以接受作业作为输入参数）。也可以选择指定其他字符串。

语法：job_starter: *stater* [*starter*] [“%USRCMD”] [*starter*]

默认情况下，用户命令在作业启动器之后运行。可以使用特殊字符串%USRCMD 来表示用户作业在作业启动器命令行中的位置。%USRCMD 字符串可以用引号括起来，也可以在其后跟其他命令。

例如:

```
job_starter: csh -c "%USRCMD;sleep 10"
```

job_idle:

定义作业空闲因子阈值，介于 0.0 和 1.0 之间，表示 CPU 时间/运行时间，以及可选的“检测后”值，表示仅在作业运行超过“检测后”分钟后才检测作业空闲，以及可选的“空闲操作”，由主集群管理员在检测到空闲作业后执行。

语法：job_idle: *idle_factor_threshold* [*detect_after* [*idle_action*]]

默认情况下，job_idle 未定义，即不检测作业空闲。如果定义了作业空闲因素阈值，则默认的“检测后”值为 20 分钟。如果未定义“空闲操作”，则检测到作业空闲后不会采取任何操作。

例如:

```
job_idle: 0.05 5 /opt/skyformai/sbin/idle_action_script
```

“idle_action”程序由集群的主管理员在主服务器上执行。环境变量 CB_IDLE_JOBS 在运行“idle_action”程序之前设置。它包含队列中所有检测到的空闲作业的作业 ID 和作业的空闲因子。多个作业之间用空格分隔。CB_IDLE_JOBS 值的示例为“102 0.01 103 0.00”。

exclusive:

定义一个独占队列。值为“enforce”，所有提交到该队列的作业都是独占的，即作业只会被调度到没有其他作业运行的主机上。若只是支持独占作业，请使用值“yes”。

默认情况下，队列不接受独占作业提交。

nice:

调整此队列中作业的 Linux 调度优先级。默认值 0（零）表示 Linux 交互式作业将保持默认调度优先级。此值会逐个队列调整作业的运行时优先级，以控制其对其他批处理或交互式作业的影响。更多详细信息，请参阅 nice (1) 手册页。

默认值：0

corelimit:

对于属于此队列中的作业的所有进程，每个进程（硬）core 文件的大小限制（以 KB 为单位）。

默认值：无限制。

filelimit:

此队列中所有属于某个作业的进程的（硬）文件大小限制（以 KB 为单位）（参见 getrlimit (2)）。您可以在数字末尾添加 M 或 G，以使限制单位为 MB 或 GB，例如 filelimit: 10G

默认值：无限制。

job_accept_interval:

将一个作业调度到某台主机后，在将第二个作业调度到同一主机之前，需要等待的调度轮次数。默认情况下，此值与 general 部分中设置的 job_accept_interval 相同，其默认值为 0。

队列级别设置的值将覆盖集群级别（在 general 部分）设置的 job_accept_interval。

memlimit:

内存驻留设置了属于该队列中某个作业的所有进程的大小限制。

值格式：[default_limit] maximum_limit | percore

设置可分配给作业进程的最大物理内存量。

默认情况下，如果指定了默认内存限制，则当达到默认内存限制时，提交到队列的没有作业级内存限制的作业将被终止。

如果只指定一个限制，则该限制为最大内存限制（或称硬限制）。如果指定两个限制，则第一个限制为默认内存限制（或称软限制），第二个限制为最大内存限制。

默认情况下，限制的单位是 KB。要指定 MB 或 GB，请在数字末尾添加 M 或 G，例如 14G。

如果值为 “percore”，则作业的内存使用量受运行主机上平均可用内存乘以请求的核心数的限制。主机上的可用内存量等于已安装内存减去主机级别或队列级别的内存调度阈值（以较大者为准）。

默认值：无限制。

processlimit:

限制可作为作业一部分的并发进程的数量。

值格式：[default_limit] maximum_limit

默认情况下，如果指定了默认进程限制，则当达到默认进程限制时，提交到队列的没有作业级进程限制的作业将被终止。

如果仅指定一个限制，则该限制为最大进程限制（或称硬限制）。如果指定两个限制，则第一个限制为默认进程限制（或称软限制），第二个限制为最大进程限制。

默认值：无限制。

runlimit:

指定作业运行时间的上限，以及可选的默认运行时间上限（以分钟为单位）。格式：[default_limit] maximum_limit。

默认情况下，处于 RUN 状态的时间超过指定的最大运行限制的作业将被 AIP 终止。您可以选择提供自己的终止作业操作来覆盖此默认设置。

如果提交的作业级别运行限制 (csub -W) 小于最大运行限制，则当达到最大运行限制时，作业将被终止。如果提交的作业级别运行限制大于最大运行限制，则将被队列拒绝。

如果指定了默认运行限制，则当达到默认运行限制时，提交到队列的没有作业级运行限制的作业将被终止。

如果仅指定一个限制，则该限制为最大运行限制（或称硬限制）。如果指定两个限制，则第一个限制为默认运行限制（或称软限制），第二个限制为最大运行限制。分钟数可以大于 59。因此，三个半小时可以指定为 3:30 或 210。

默认：无运行限制。

例如：runlimit: 20 60 # 最大运行限制为 60 分钟，默认 20 分钟

例如：runlimit: 60 # 最大运行限制为 60 分钟

pre_exec:

指定在作业执行前在执行主机上运行的命令。如果同时指定了作业级别预执行命令 (csub -E) 和队列级别预执行命令，则作业级别预执行将在队列级别预执行命令之后运行。预执行和后执行命令在 /bin/sh -c 和 /tmp 下运行。stdin/stdout/stderr 设置为 /dev/null。PATH 环境变量设置为：/bin;/usr/bin;/usr/sbin。

如果预执行命令以非零退出代码退出，则视为失败，并将作业重新排队至队列的头部。

默认值为空，即不执行预执行命令。

post_exec:

指定在作业执行后的第一个主机上运行的命令。其行为类似于 pre_exec（参见上文）。

默认值为空，即没有执行后命令。

pre_post_exec_user:

为队列级别的执行前和执行后命令指定与默认值不同的用户名。

默认值：执行作业的用户

kill_action:

更改作业终止操作的行为。该值可以是信号名称（例如 SIGTSTP），也可以是命令路径（不能是 ckill 或 crequeue，因为这会导致死锁）。

为作业设置的所有环境变量也可用于命令操作。以下附加环境变量已设置：CB_JOBPGIDS - 作业当前进程组 ID 列表。CB_JOBPIDS - 作业当前进程 ID 列表。

默认的作业终止操作是：发送 SIGINT 信号。如果作业仍在运行，则在 10 秒后发送 SIGTERM 信号。如果作业仍在运行，则再过 10 秒发送 SIGKILL 信号。

stop_action:

更改作业暂停操作的行为。值可以是信号名称（例如 SIGTSTP），也可以是命令路径（不能是 cstop，否则会导致死锁）。

为作业设置的所有环境变量也可用于命令操作。此外，还设置了额外的环境变量 CB_JOBPGIDS 和 CB_JOBPIDS（有关这些变量的含义，请参阅 kill_action）。CB_SUSP_REASONS 也已设置，它是一个整数，表示暂停原因的位图，定义在 cbsched.h 中。

默认的作业暂停操作是：对于普通串行作业发送 SIGSTOP，对于并行或交互式作业发送 SIGTSTP。

resume_action:

更改作业恢复操作的行为。该值可以是信号名称（例如 SIGTSTP），也可以是命令路径（不能是 cresume，因为这会导致死锁）。默认是向所有作业进程发送 SIGCONT 信号。

为作业设置的所有环境变量也适用于命令操作。此外，还设置了额外的环境变量 CB_JOBPGIDS 和 CB_JOBPIDS（有关这些变量的含义，请参阅 kill_action）。

默认的作业恢复操作是：发送 SIGCONT。

run_window:

允许队列中的作业运行的时间段。当窗口关闭时，AIP 会暂停队列中正在运行的作业，并停止从队列中调度作业。当窗口重新打开时，AIP 会恢复已暂停的作业并开始调度其他作业。可以在同一行上定义多个时间窗口，并用空格分隔。

时间窗口语法：[day:]hour[:minute]。注意 day=[0-6]：0 表示星期日，1 表示星期一，6 表示星期六。如果只有一个字段，则默认为 hour；如果有两个字段，则默认为 hour[:minute]。

例如：5:19:00-1:8:30 表示从星期五 19:00 到星期一 8:30。

默认情况下，运行窗口未定义，这意味着队列始终处于活动状态。

dispatch_window:

此队列中的作业被调度的时间窗口。一旦调度完成，作业将不再受调度窗口的影响。调度窗口的规范语法与 run_window 语法相同（参见上文）。默认情况下，dispatch_window 未定义，这意味着队列始终处于打开状态。

jobsizes:

一系列规范表示作业大小，以作业槽位数量 (csub -n) 以及队列可接受的 GPU 数量（可选）表示。提交的作业如果槽位请求数量与这些数量不同，则会被拒绝。格式为：job_slots/gpu_numbers。

示例:

```
jobsizes:
- 4/2 # 4个CPU和2个GPU
- 8/4 # 8个CPU和4个GPU
- 16 # 16个CPU和任意数量的GPU
- /8 #任意数量的CPU和8个GPU
```

缺省：没有定义，队列可以接受任意小于作业槽数和 GPU 数请求的作业。

interactive:

指定队列如何接收交互式作业。如果未指定，则队列可以接收交互式和非交互式作业。值“only”表示队列不接收非交互式作业。值“no”表示队列仅接收非交互式作业。

dedicated:

如果值为“y”或“yes”，则该队列为专属队列，专供特定用户组使用。全部 CPU 小时数和 GPU 小时数都为用户计量，无论是否有任何作业正在运行。这不会影响任何调度行为。

gpu_enforce:

如果设置为“yes”，则使用非调度 GPU 的作业进程将被调度程序强制终止。使用 GPU 的作业必须在提交作业时指定 -R rusage[gpu=x]。如果未设置，则默认为“no”，即使用非调度 GPU 的进程不会被调度程序终止。

备注： 这个参数已经被 cgroup 取代，以后的版本中会去除。

---

swaplimit:

此队列中作业的总虚拟内存限制（以 KB 为单位）。此限制适用于整个作业，无论该作业包含多少个进程。

当作业超出其交换限制时采取的操作是按顺序发送 SIGQUIT、SIGINT、SIGTERM 和 SIGKILL。

默认值：无限制。

terminate_when:

配置队列在指定情况下调用 kill_action 而不是 stop_action。

语法：terminate_when: WINDOW | LOAD

WINDOW 表示如果运行窗口关闭，则终止作业。LOAD 表示当负载超过暂停阈值时终止作业。使用 WINDOW 时，必须指定队列 run_window。使用 LOAD 时，必须指定负载阈值。

thresholds:

指定主机的调度阈值列表。阈值的格式为：“load_namesched_threshold/stop_threshold”。

load_name 的值为“mem”、“swp”和“ut”（CPU 利用率在 0.0 - 1.0 范围内）。sched_threshold 和 stop_threshold 是浮点值。当负载达到“sched_threshold”时，调度程序将停止向主机调度更多作业。当负载达到“stop_threshold”时，调度程序将停止在主机上运行作业。

对于“mem”、“swp”和“tmp”，单位均为 MB。对于“mem”、“swp”和“tmp”，默认单位均为 MB。您可以在数值后添加字母“G”将单位更改为 MB，或添加字母“T”将单位更改为 TB。

负载索引名称与“cinfo”的输出相同：“r15s”、“r1m”、“r15m”、“ut”、“pg”、“io”、“up”、“it”、“tmp”、“swp”或“mem”。

该阈值支持多个时间窗口，每个值的格式为：“sched_threshold1/stop_threshold1 [weekday:] 小时 [: 分钟]-[weekday:] 小时 [: 分钟] sched_threshold2/stop_threshold2 [weekday:] 小时 [: 分钟]-[weekday:] 小时 [: 分钟]”。

或者格式可以是：“sched_threshold/stop_threshold default (weekday1-weekday2)hour:minute-hour:minute;(weekday3,weekday4,week5)hour:minute-hour:minute”

日期数字代表星期几，类似于 crontab，即 0 或 7 代表星期日，1 代表星期一，等等。

示例 1:

```
thresholds:
- "ut 0.7/0.9"
- "mem 250/200"
```

在上面的例子中，作业调度条件是“mem >= 250 && ut <= 0.7”，作业停止（挂起）条件是“mem < 200 || ut > 0.9”。

示例 2:

```
- "mem 1G/0.5G 1:8:00-5:20:00 0.6G/0.3G 5:20:00-1:8:00"
```

上面的例子意味着阈值为周一早上 8 点到周五晚上 8 点 1G/0.5G，周五晚上 8 点到周一早上 8 点 0.6G/0.3G。

示例 3:

```
- "mem 100/50 default 50/30 (1-5)8:00-20:00,(0,6)6:00-22:00"
```

以上示例表示，工作日 8:00 - 20:00 以及周末 6:00 - 22:00 的阈值分别为 50MB/30MB。除此以外，阈值（默认值）为 100MB/50MB。

vgpu:

指定是否限制作业在共享 GPU 上资源的使用。当作业请求小于 1 个 GPU 卡时，这个 GPU 卡会由多个作业共享。共享时每个作业使用的 GPU 的算力以及 GPU 现存由 GPU 根据负载自行分配，必要时 GPU 会用分时的方法管理共享。

当 vgpu 参数配置成“y”时，调度器会使用第三方的 CUDA 插件限制作业使用的 GPU 算力和显存。如一个作业申请 0.5 个 GPU，则该作业被限制只能使用 0.5 个 GPU 卡的算力和 0.5 的 GPU 显存。

缺省：n，作业申请小于 1 个 GPU 时，作业使用的实际 GPU 资源由 GPU 自行决定。

limitconf

它指定包含 limits 部分的文件名。该文件应与 cb.yaml 位于同一目录中。目前仅支持“limit.yaml”文件名。

limits 部分

限制部分指定用户、项目、队列和主机或这些消费者的任意组合的作业槽、作业和资源消耗的限制。

指定消费者时，保留字“all”表示系统中的所有消费者单元。例如：queues 或 per_queue 中的“all”表示系统中配置的每个队列。

特殊字符’~’表示“除”。例如，“queues: ‘all ~low’”表示系统中配置的除“low”队列之外的所有队列。

所有值都必须用引号引起来，以避免 YAML 解析程序因特殊字符“~”、“[”、“]”等崩溃。

参数影响的 AIP daemon：cbsched

name:

指定限制名称。此为必填字段。

admin:

该限制的管理员用户名。管理员有权限可以通过命令 aip l u 修改部分该 limit 的参数。参考调度器动态配置参数 。

queues | per_queue:

以空格分隔的队列列表。这是可选的。

hosts | per_host:

以空格分隔的主机或主机组列表。这是可选的。

projects | per_project:

以空格分隔的项目列表。这是可选的。

apps | per_app:

按空格列出的应用程序名称。这是可选的。

users | per_user:

系统中的用户列表。用户组名称也可用于表示整个用户组及其子组。如果消费者类型为“per_user”，则限制将应用于指定用户组中的每个用户。此项为可选。

以上的定义值允许用“all”代表所有，并用 ~name 排除。如“all ~user1 ~user2”表示除了 user1 和 user2 外的所有用户。

slots:

使用可选的时间窗口指定作业槽位的限制。仅接受单个数字，可选时间窗

例如：slots: ‘[2 1:08:00-5:18:00] [5 9:00-19:00]’

jobs:

指定作业数量的限制。仅接受单个数字，可选时间窗

例如：jobs: '[2 1:08:00-5:18:00] [5 9:00-19:00]'

resources:

指定资源限制。格式：[res,limit time_window] [res,limits time_window] …

例如：资源："[mem,5000] [gpu,4 9:00-18:00]"

时间窗口的格式为：[day:]hour[:minute]-[day:]hour[:minute]。天数代表星期几，类似 crontab，即 0 或 7 为星期日，1 为星期一等。

例子 1：不允许指定项目以外的作业使用 priority 和 idle 队列。

```
- name: limit1
  projects: all ~p1 ~p2 ~p3
  queues: priority idle
  slots: 0
```

例子 2：每个项目最多 300 个作业槽。

```
- name: projectlimit
  per_project: all
  slots: 300
```

apps 部分

apps 部分控制应用的调度速率，避免每秒分发太多作业造成基础架构中某些资源的瓶颈，如 FLEXlm 许可证服务器、共享存储等。

参数影响的 AIP daemon：cbsched

name:

应用名。这个名字与作业提交 (*csub*) 时的-A 或-app 匹配。以这个应用名提交的作业收到 job_rate 的限制。

job_rate:

每秒分发的最大作业数。调度器控制每秒内分发的最多以这个应用名提交的作业数。

例子:

```
apps:
- name: vcs
  job_rate: 1000
```

以上的配置控制 csub -A vcs 的作业每秒不超过 1000 个作业被调度和分发。

power 部分

power 部分指定省电调度参数。此部分为可选。如果未指定（默认），则管理员通过 AIP 对主机执行的所有电源操作都将被禁用。

省电调度允许 AIP 在主机空闲时间达到配置的时间段时自动执行关机命令以关闭主机，并在队列中有等待的作业且有省电主机时自动执行开机命令。

参数影响的 AIP daemon：cbsched

idle_time:

指定 AIP 触发关机操作以节省电量的最短主机空闲时间（以分钟为单位）。配置“电源”部分时，此字段为必填项。

默认情况下，该值为 0，表示禁用省电功能。

pend_time:

指定触发节能主机恢复电源的最短作业挂起时间（以分钟为单位）。默认值为 10 分钟。

cycle_time:

指定最大电源循环时间（以分钟为单位）。如果在此时间段后发生电源操作，并且主机电源状态未发生变化，则 AIP 会认为电源操作失败，并在 cbsched.log.<master> 中记录一条消息。默认值为 5 分钟。

suspend_rate:

触发省电操作时每分钟关闭的最大主机数量。默认值为 60。

resume_rate:

当作业处于等待状态且存在节能主机时，每分钟启动的最大主机数量。默认值为 300。

exclude_hosts:

排除在节能策略之外的主机列表。如果未指定，则排除主主机和主候选主机。

如果指定了此参数，请确保主服务器和主服务器候选服务器在列表中。

power_down_cmd:

在主控主机上执行的用于关闭主机的命令。该命令以主管理员（即“administrators”部分中定义的第一个用户）的身份执行。命令语法为“command host1 host2 …”。

power_up_cmd:

在主控主机上执行的启动主机的命令。该命令以主管理员（即“administrators”部分中定义的第一个用户）的身份执行。命令语法为“command host1 host2 …a”。

power_restart_cmd:

在主控主机上执行的用于重启主机的命令。该命令以主管理员（即“administrators”部分中定义的第一个用户）的身份执行。命令语法为“command host1 host2 …”。此命令不用于省电调度。它是管理员手动重启主机的命令。

power_down_filter | power_up_filter:

这两个参数用于实现电源关闭和电源开启策略。如果已配置，调度程序将以电源关闭和电源开启候选主机作为参数执行命令。该命令应返回主机列表（以空格分隔），以指示哪些主机可以执行与电源相关的电源操作。

例如，如果您使用“power”参数来实现 AIP 与其他调度程序之间的资源共享，这些命令可用于检查“power up”主机是否有来自其他调度程序的工作负载，而这些调度程序无法加入 AIP 集群。

例子：

开机过滤器：ssh k8smaster k8sidlenodes

在上面的例子中，该命令应该返回没有运行 pod 并且可以加入 AIP 集群的节点数。

默认值：未定义，不执行任何检查。

power_sched_interval:

节能调度外插程序调用的时间间隔（以秒为单位）。默认情况下，它会调用内置的节能调度程序。如果存在可执行文件/opt/skyformai/sbin/power_ext_sched，则会以此间隔调用该文件来替换内置的节能调度程序。

当调用“power_ext_sched”时，cbsched 将 JSON 格式的数据通过管道传输到 power_ext_sched 程序，即 power_ext_sched 应该从其 stdin 读取 JSON 数据，以便外部调度程序做出决策并采取行动。

此 JSON 的示例输出是 /opt/skyformai/sbin/power_ext_sched.input.json。

默认值：60 秒。

scale 部分

scale 部分指定自动缩放调度参数。此部分是可选的。如果未指定（默认），则禁用自动缩放。

自动伸缩调度监控特定队列中的作业和动态主机空闲情况，根据配置触发自动伸缩动作。

参数影响的 AIP daemon：cbsched

scale_pend_time:

指定触发扩容操作前，作业在 *scale_queues* 中等待的最小分钟数。配置 *scale* 部分时，此字段为必填项。

有效值必须大于或等于 1。调度程序每分钟检查一次作业等待时间。如果 cbsched 重新启动或重新配置，则计时器将被重置。

scale_idle_time:

指定应通过 scale_down_action 删除动态添加的主机（横向扩展主机）的最小分钟数。

有效值必须大于或等于 1。调度程序每分钟检查一次空闲主机。如果 cbsched 重新启动或重新配置，则计时器会重置。默认值为 5 分钟。

scale_max_slots:

指定要扩展的最大作业槽数（这些可以通过自动扩展操作 *scale_up_action* 动态添加。

有效值必须等于或大于 1。默认值为无限制。

scale_up_action:

指定集群扩展（动态添加主机）的绝对路径。该可执行文件可以接受一个参数，该参数表示 scale_queues 中等待时间超过 scale_pend_time 的作业数量。

该操作由 cbsched 以主集群管理员用户身份执行。

此参数仅在指定 scale_pend_time 且有效时有效。若不指定此参数，则不进行任何扩容操作。默认值为 none，即自动扩容不采取任何操作。

scale_down_action:

指定缩减集群规模（从中移除动态添加的主机）的绝对路径。该可执行文件可以接受一个或多个参数，这些参数是空闲时间超过 scale_idle_time 的主机名。

该操作由 cbsched 以主集群管理员用户身份执行。

如果未指定此参数，则不会采取缩减操作。

scale_up_buffer:

指定在扩展时需要预配的作业槽数量（除等待作业所需的数量外）。默认值为 0。

scale_down_buffer:

指定即使检测到空闲也要保持正常运行的动态预配主机数量。默认值为 0。

scale_queues:

指定自动扩缩策略应监控的一个或多个队列。多个队列名称应以空格分隔。
默认情况下，系统中的所有队列均受监控。

scale_sched_interval:

指定调度器触发伸缩操作的分钟数，默认值为 5 分钟。

例子

见 AIP 安装完成后/opt/skyformai/etc/cb.yaml
单机集群简单配置:

```
cluster:
  name: aip
  administrators:
  - cadmin
  hosts:
  - name: mgt
  - name: default
    maxslots: cpu
  define_ncpus: threads
  clients: yes
  enterprise: no
general:
  default_queue: medium
  cgroup: acct
  mailprog: ":"
queues:
- name: medium
  priority: 3
```

5.2.4 cbcrond.yaml

配置文件

cbcrond.yaml

概述

cbcrond.yaml 文件定义了作业资源使用情况、实时成本和每月账单生成。

该文件采用严格的 YAML 格式。包含特殊字符的文本字符串需要用单引号括起来（例如 “xxx”）。

内容

· usage 部分

· charge 部分

· bill 部分

usage 部分

描述

usage 部分定义了定期作业使用情况生成的属性。

属性

enabled：
启用（值：true）或禁用（值：false）作业使用情况生成。

启用此功能后，cbcrond 将定期生成作业使用情况文件，包括在此期间的 CPU、内存、GPU 和应用程序资源使用情况。

outputperiod：
指定作业使用量生成的时间段（以分钟为单位）。允许的值介于 1 到 240 分钟之间。请确保此值小于 cb.yaml 中定义的 memperiod。默认情况下，outputperiod 为 60 分钟。

默认值为：60（分钟）

usagefilepath：
指定作业使用量文件路径。请确保 SkyForm AIP 管理员对此路径拥有写入权限。

默认值为：/opt/skyformai/work/usages

charge 部分

描述

charge 部分定义了实时定价的属性。

属性

enabled:
启用（值：true）或禁用（值：false）定期使用量成本生成。

启用此功能后，cbcrond 将在作业使用量文件中同时生成作业使用量成本和资源使用量。禁用此功能后，作业使用情况文件仅包含资源使用情况。

费用部分包含标准和用户折扣子部分。

standard:
standard 子部分定义标准价格。这部分包含 cpuperhour、mempergbhour、gpuperhour 和 appperhour 子部分。

cpuperhour:
cpuperhour 子节定义了队列的每 CPU 每小时价格列表。如果只有一个价格，则队列必须是全部。

每个价格包含以下属性：

· queue：队列名称或全部。

· price：每 CPU 每小时的价格。

mempergbhour:
mempergbhour 子节定义了一个队列的每 GB 每小时内存价格列表。如果只有一个价格，则该队列必须是全部。

每个价格包含以下属性：

· queue：队列名称或全部。

· price：内存每小时价格（以 GB 为单位）。

gpuperhour:
gpuperhour 子节定义了一个队列的每 GPU 每小时价格列表。如果只有一个价格，则该队列必须是全部。

每个价格包含以下属性：

· queue：队列名称或全部。

· price：每个 GPU 每小时价格。

appperhour:
appperhour 子节定义了一个应用程序每小时价格列表。

每个价格包含以下属性：

· app：应用程序。

· price：每个应用程序每小时价格。

userdiscounts:
用户折扣子部分定义了用户折扣列表。

· user：用户名。

· cpu：CPU 价格折扣。必须介于 0 到 1 之间。

· mem：内存价格折扣。必须介于 0 到 1 之间。

· gpu：GPU 价格折扣。必须介于 0 到 1 之间。

· apps：应用程序折扣列表。

· app：应用程序名称。

· rate：此应用程序的折扣。必须介于 0 到 1 之间。

bill 部分

描述

bill 部分定义了每月账单生成的属性。

属性

enabled：
启用（值：true）或禁用（值：false）每月账单生成。

启用此功能后，cbcrond 将定期为每个用户生成每月账单文件。

billpath：
指定每月账单文件路径。请确保 SkyForm AIP 管理员对此路径拥有写入权限。

默认值为：/opt/skyformai/work/bills

reconstruct：
在文件系统故障时，一些用户账单可能会生成失败。如果 usage（/opt/skyformai/work/usage）数据完整，用户帐单可以通过 usage 进行修复。把这个值设成 true，并且设置参数 lastupdatetime（见下面的参数说明）重启 cbcrond:

```
killall cbcrond; csadmin jmrestart
```

cbcrond 开始修复账单，这个过程需要持续数小时。当账单修复完成后，日志中会有“ReconstructBill is finished”。这时可以

1. 修改 cbcrond.yaml，把 reconstruct 的值改成 false，
2. 用上面的命令重启 cbcrond

lastusagetime：
与 reconstruct 一起使用，指示 cbcrond 从哪个 usage 文件开始使用 usage 数据修复账单。

usage 文件每 15 分钟生成一个，只有当月的（上一个月的保存在子目录 lastmonth 里）。文件名的格式为 ddhhmm.json。这里 dd 表示日，hh 表示时，mm 表示分。如果账单恢复从某个 usage 文件开始，打开这个文件，把头部” This_usage_time” 后面的时间戳（如 1747938600）配置在 cbcrond.yaml 的 lastusagetime 参数中。

警告： 恢复帐单后务必修改配置文件，重启 cbcrond。

5.2.5 gpu.yaml

配置文件

gpu.yaml

概述

定义 NVIDIA 以外 GPU 的适配参数。

描述

对于除了 NVIDIA 外的 GPU，包括国产 GPU 的支持，AIP 使用这个文件来定义 GPU 适配的各种参数

gpu.yaml 列出了所有支持的 GPU。AIP 的 cbls 在启动时按照 gpu.yaml 里的配置顺序寻找本机安装的 GPU。

例子:

```
gpuconf:
- vendor: ILUVATAR
  discovery_file: /usr/local/corex/bin/ixsmi
  metrics_cmd: LD_LIBRARY_PATH=/usr/local/corex/lib64 /usr/local/corex/bin/ixsmi --
↪query-gpu=uuid,name,memory.total,temperature.gpu,memory.free,utilization.gpu,
↪utilization.memory,power.draw --format=csv,nounits,noheader
  first_device: /dev/iluvatar0
- vendor: MUXI
  discovery_file: /usr/bin/mx-smi
  metrics_cmd: /opt/skyformai/sbin/mx-smi.py
  fixed_devices:
  - /dev/mxcd
  first_device: /dev/dri/renderD128
  second_device: /dev/dri/card1
```

cbls 先查 discovery_file，如果发现该文件存在，则认为本机安装了相应的 GPU，而不再继续查看 gpu.yaml 其他 GPU 的信息。cbls 定期运行 metrics_cmd 里定义的命令，返回 GPU 的信息。命令返回的信息必须是多行的 CSV，每个 GPU 一行，格式如下：Id, 型号, GPU 总显存 MB, 温度, 可用显存 MB, GPU 利用率 (0-100), GPU 显存利用率 (0-100), 功耗 (W) metrics_cmd 运行返回的例子:

```
GPU-06bb47c4-da16-6f49-9569-175db9c6a5b4, NVIDIA A800 80GB PCIe, 81920, 51, 65491, 0,␣
↪0, 43
GPU-4dd97756-eedd-4b67-6838-6e31d9be140a, NVIDIA A800 80GB PCIe, 81920, 57, 65497,␣
↪100, 0, 32
GPU-d22802be-6518-f889-e60a-50ae36b3cf9d, NVIDIA A800 80GB PCIe, 81920, 57, 65497,␣
↪100, 0, 33
GPU-7578cd1c-0319-6ca5-418d-e5b7b45c6860, NVIDIA A800 80GB PCIe, 81920, 54, 65497,␣
↪100, 0, 23
```

许多国产 GPU，设备命令返回不了上面格式的 csv，就需要开发一个格式转换脚本，一般用 Python 来开发。如上面例子中 MUXI 的 GPU 的命令/usr/bin/mx-smi 返回的值需要通过脚本/opt/skyformai/sbin/mx-smi.py 经过转换成 cbls 认识的 CSV 格式。

小技巧: 适配的一个关键点就是开发这个格式转换脚本。这个脚本调用原厂的设备信息命令行，把返回转换成 cbls 需要的格式。

在启动容器作业时，我们利用 docker run –device 参数指定容器可以访问的 GPU 设备。gpu.yaml 里的以下几个参数配置这些设备名。这些设备名也被 cgroup 使用来控制作业只能用被 AIP 分配的 GPU：

- fixed_devices 对于很多种 GPU，有一些所有卡都共用的设备名，这些设备列在这个参数下面。如 NVIDIA 的 GPU 有以下公用的设备：/dev/nvidiactl, /dev/nvidia-uvm, /dev/nvidia-uvm-tools, /dev/nvidia-modeset, /dev/nvidia-caps
- first-device 每卡相应的设备名的第一个。如 4 卡 NVIDIA GPU 有/dev/nvidia0 /dev/nvidia1 /dev/nvidia2 /dev/nvidia3。在这里只需配置第一个就可以了，其他的 AIP 会顺序发现。一些 GPU 品牌的设备名较怪，如 MUXI 的第一个 GPU 设备是/dev/dri/renderD128，第二个是/dev/dri/renderD129 等等。还有一些每个卡需要两个设备名，则第二个配置在 second_device 里（见上面 MUXI 的例子）

参数

vendor: 名字
: 任意的字串，定义 GPU 厂商名字。这个字段只用于标识，AIP 内部不使用该字段。

discovery_file: 文件路径
: GPU 独特的文件路径，以便 AIP 判断 GPU 厂商。AIP 扫描到该文件存在，便会认为主机上安装了这个厂商的 GPU。

metrics_cmd: *GPU* 参数命令路径
: 运行这个命令可以动态获取 GPU 的各项参数。这个命令的输出格式必须是多行的无头 CSV，每个 GPU 一行，格式如下：Id, 型号, GPU 总显存 MB, 温度, 可用显存 MB, GPU 利用率 (0-100), GPU 显存利用率 (0-100), 功耗 (W)。例子见上面的“描述”部分。

fixed_devices: *YAML* 设备列表
: 所有 GPU 公用的设备名列表。很多厂商的 GPU 公用设备有多个，每一个都必须列出，否则作业将无法访问 GPU。

first_device: 第一个 *GPU* 对应的设备名
: 第一个 GPU 对应的设备名。AIP 利用这个名字通过对设备名最后一个数字加一的方法自动计算第二、第三等 GPU 的设备名。

second_device：第一个 *GPU* 对应的额外设备名
: 一些厂商每个 GPU 多于一个的设备，这个参数用于提供第一个 GPU 的相应的第二个设备名。

诊断

如果命令 chinfo -g (*chinfo*) 没有显示某台主机上的 GPU，登录到这台 GPU 主机。

1. 运行 gpu.yaml 中相应 GPU 配置中 metrics_cmd 里的命令，查看输出。输出必须符合“描述”中例子的格式。
2. 运行/opt/skyformai/sbin/cblt -t。这个命令会显示 AIP 探测到的硬件配置。

小技巧:

- CBLS 按照 gpu.yaml 中配置的循序来判断主机上安装的 GPU。如果其他类型 GPU 中的 discover_file 在主机上出现，会影响 CBLS 的判断。
- 很多 GPU 的参数获取命令需要一定的环境变量，如 LD_LIBRARY_PATH。这个值必须在 metrics_cmd 里设置。操作系统中设的缺省环境变量不会被使用。

5.2.6 jservice.yaml

配置文件

jservice.yaml

概要

服务*jservice* 的配置文件。请参考*jservice* 。

5.2.7 olmon.conf

配置文件

olmon.conf

概述

SkyForm AIP 监控 - Olmon 是一款用于 SkyForm AIP 的监控和数据分析软件。olmon.conf 文件定义了 Olmon 的配置参数。

描述

olmon.conf 需要在安装过程中手动创建。该文件供 Olmon 使用。Olmon 仅支持 Elasticsearch v7 和 v8。

位置

olmon.conf 的位置为 $CB_ENVDIR(/opt/skyformai/etc)。

格式

olmon.conf 中的每个条目都采用以下格式之一：

NAME=VALUE

NAME=STRING1,STRING2,…

每个 NAME 后面必须跟等号 =

以井号 (#) 开头的行是注释，会被忽略。

参数

eshosts

语法：**eshosts=***host1,host2,⋯*

说明：指定 Elasticsearch 服务器主机列表。多个主机名之间用逗号 (,) 分隔。数据收集器默认将数据写入第一个主机。如果第一个主机不可用，则会尝试第二个主机、第三个主机，依此类推。

有效值：属于 Elasticsearch 集群的有效主机名。

默认值：未定义。如果没有此参数，olmon 服务将无法启动。

job_update_interval

语法：**job_update_interval=***time_seconds*

说明：Olmon 的数据收集器会定期向工作负载管理系统查询动态作业信息。此参数指定两次查询之间的时间间隔（以秒为单位）。受此参数影响的 Elasticsearch 索引更新包括：jobs-time 和 users-time。

有效值：大于 10 的任意正整数。

默认值：30

host_update_interval

语法：**host_update_interval=***time_seconds*

说明：指定 Olmon 从工作负载管理系统查询主机信息的时间间隔。Elasticsearch 索引主机时间的更新受此参数影响。

有效值：大于 10 的任意正整数。

默认值：30

resource_update_interval

语法：**resource_update_interval=***time_seconds*

说明：指定 Olmon 从工作负载管理系统查询共享资源的时间间隔。Elasticsearch 索引 shres-time 的更新受此参数影响。

有效值：大于 10 的任意正整数

默认值：30

rusagepreason_update_interval

语法：**rusagepreason_update_interval=***time_seconds*

说明：指定 Olmon 工作负载管理系统查询作业等待原因的时间间隔。此参数的最小值为 job_update_interval。Elasticsearch 索引 preasons-time 的更新受此参数影响。

有效值：大于 10 的任意正整数。

默认值：60

log_duration

语法：**log_duration=***time_days*

说明：指定 Olmon 保存数据的时间段（以天为单位）。任何超过此时间段的数据都将被覆盖。

有效值：大于 10 的任意正整数。

默认值：90

job_index

语法：**job_index=**0 | 1

说明：指定是否使用索引“jobs”来存储作业详情数据。值 1 表示“是”，值 0 表示“否”。

有效值：0 或 1。0 表示“否”。1 表示“是”。

默认值：1（是）

lmstat_path

语法：**lmstat_path=/opt/skyformai/bin/aip**

描述：指定许可证查询命令 **aip** 的绝对路径。

有效值：指定的路径必须可由主 AIP/LSF 管理员读取和执行。

默认值：无。许可证不受监控。

license_update_interval

语法：**license_update_interval=***time_seconds*

说明：指定从许可证服务器查询许可证使用情况的时间间隔。

有效值：大于 10 的任意正整数。

默认值：30

kibana

语法：**kibana=***IP_address*

说明：指定 Kibana 服务器的 IP 地址。Web 门户使用此参数导入 Kibana 可视化数据。

有效值：门户 Web 服务器可以访问的 Kibana 服务器 IP 地址。

默认值：无。Kibana 未启用

grafana

语法：**grafana=***IP_address*

说明：指定 Grafana 服务器的 IP 地址。此参数用于 Web 门户导入 Grafana 仪表板。

有效值：门户 Web 服务器可以访问的 Grafana 服务器 IP 地址

默认值：无。Grafana 未启用

user_data_cmd

语法：**user_data_cmd=***executable_path*

说明：指定列出附加用户 meta data 数据的可执行文件。

用户 meta data 数据的输出格式如下：

第一行列出需要包含用户元数据的 Elasticsearch 索引名称。多个索引名称之间用空格分隔。

第二行列出元数据字段名称，例如 “dept group proj”。

从第三行开始列出用户名及其关联的元数据。每行一个用户条目。

输出示例:

```
jobs users-time
dept team
u001 rd chip1
u002 rd chip2
u003 qa veri
```

有效值：以上述格式将用户 meta data 数据输出到标准输出的可执行文件的绝对路径。

默认值：无。用户 meta data 数据未启用。

user_data_interval

语法：**user_data_interval=***time_seconds*

说明：指定用 meta data 元数据的更新间隔（以秒为单位）。由于用户 meta data 数据通常不会更改，因此建议将此参数的值保持在 3600 秒（一小时）以上，以避免系统过载。

有效值：大于 300 的任意整数。

默认值：3600 秒

purge_interval

语法：**purge_interval=***time_days*

说明：清理 Elasticsearch 垃圾数据的间隔（以天为单位）。

有效值：大于 1 的任意整数。

默认值：7（天）

logfiles

语法：**logfiles=** "*daemon_name daemon_name* ⋯"

说明：把调度器 master 服务器的 daemon 日志传入 Elasticseatch。日志最多保留 log_duration 天（缺省 90 天）。日志的 index 名为 "log-daemon 名-主机名"。

有效值：字串。

默认值：无，即不上传 daemon 日志。

use_estream

语法：**use_estream=** "yes | no"

说明：olmon 是否使用 AIP 的扩展 stream 数据来获取最新的调度期内有关作业、主机、队列、用户组等的信息。

有效值：yes 或 no。

默认值：由*jservice.yaml* 里的 use_estream 参数决定。

> **警告：** 当使用 AIP 的扩展 stream 数据时，数据不是实时的，一般会有 30-600 秒的延迟，主要的作用是减低对调度器的压力。

5.3 服务进程

5.3.1 aip-exporter

服务

aip-exporter - AIP 系统的 Prometheus 数据输出服务

概要

aip-exporter systemd 服务

描述

aip-exporter 是 Linux systemd 的服务。服务文件位置：/lib/systemd/system/aip-exporter.service。

配置

服务的配置文件为/opt/skyformai/etc/aip-exporter.yaml。

配置的参数为：max_run_jobs: 值

参数取值为：

- 0: 不输出运行中每个作业的指标
- -1: 输出运行中每个作业的指标
- *N*: *N* 为正整数，输出最多 N 个运行作业的指标

这个参数的作用是为了减小 Prometheus 的负担，不管这个值是多少，aip-exporter 通过 cjobs 命令从 AIP 调度器的数据量是不变的。

> **警告：** 如果过系统较大（超过 500 台主机）或者活动作业数超过 10000，为了减小对调度的负担和较小 Prometheus 的压力，建议在/usr/lib/prometheus/prometheus.yml 中调大数据获取间隔。

例子:

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s
  scrape_timeout: 15s

scrape_configs:
- job_name: "prometheus"
  static_configs:
  - targets: ["localhost:9090"]
- job_name: "aip_exporter"
  scrape_interval: 120s
  scrape_timeout: 120s
  static_configs:
  - targets: ["localhost:16010"]
```

数据

输出的数据为：

集群指标：

- aip_cluster_maxslots 集群总作业槽数
- aip_cluster_nclosed 集群关闭的作业槽数
- aip_cluster_nresv 集群保留的作业槽数
- aip_cluster_nrun 集群运行的作业槽数
- aip_cluster_nunavail 集群不可用作业槽数
- aip_cluster_nfree 集群可用作业槽数

- aip_cluster_maxgpus 集群最大 GPU 数
- aip_cluster_usedgpus 集群已用 GPU 数
- aip_sharedfs_maxgb 所有共享存储上的最大容量（GB）
- aip_sharedfs_usedgb 所有共享存储上的已用容量（GB）

每台主机的指标：

- aip_host_cpuut 主机 CPU 利用率%（0-100）
- aip_host_ncpus 主机 CPU 核数
- aip_host_iokbps 主机网络 IO kb/s
- aip_host_maxmemgb 主机安装的内存 GB
- aip_host_maxslots 主机最大作业槽数
- aip_host_maxswapgb 主机配置的交换区 GB
- aip_host_nresv 主机预留作业槽数
- aip_host_nrun 主机运行作业槽数
- aip_host_nfree 主机可用作业槽数
- aip_host_r15m 主机 OS 进程队列 15 分钟平均值
- aip_host_r15s 主机 OS 进程队列 15 秒钟平均值
- aip_host_r1m 主机 OS 进程队列 1 分钟平均值
- aip_host_uptime 主机开机后的时间（分钟）
- aip_host_freememgb 主机可用内存 GB
- aip_host_freeswapgb 主机已用交换区 GB
- aip_host_tmpgb 主机可用/tmp 存储 GB
- aip_host_maxgpus 主机最大 GPU 数
- aip_host_freegpus 主机可用 GPU 数

以上每个指标都带有标签：

- host 主机名
- model 主机 CPU 型号
- 主机状态（参考*chosts* 输出）

每个队列的指标：

- aip_queue_npend 队列等待作业槽数
- aip_queue_nrun 队列运行作业槽数
- aip_queue_nsusp 队列挂起作业槽数
- aip_queue_pend_gpus 队列等待 GPU 的作业槽数
- aip_queue_ngpus 队列已用 GPU 数
- aip_queue_totalcpus 队列最多可用的作业槽数
- aip_queue_totalgpus 队列最多可用的 GPU 数

以上每个指标都带有标签：

- queue 队列名
- status 队列状态（参考*cqueues* 输出）

每个用户的指标：

- aip_user_npend 用户等待作业槽数
- aip_user_nrun 用户运行作业槽数
- aip_user_nsusp 用户挂起作业槽数
- aip_user_ngpus 用户作业使用的 GPU 数

以上每个指标都带有标签：

- user 用户名

每个用户组的指标：

- aip_ugroup_npend 用户组等待作业槽数
- aip_ugroup_nrun 用户组运行作业槽数
- aip_ugroup_nsusp 用户组挂起作业槽数
- aip_ugroup_ngpus 用户组作业使用的 GPU 数

以上每个指标都带有标签：

- ugroup 用户组名

每种应用名的指标：

- aip_cat_npend 应用名等待作业槽数
- aip_cat_nrun 应用名运行作业槽数
- aip_cat_nsusp 应用名挂起作业槽数
- aip_cat_ngpus 应用名作业使用的 GPU 数

以上每个指标都带有标签：

- category 应用名

作业数统计：

- aip_job_npend 总等待作业数
- aip_job_pgpus 等待作业 GPU 需求总数
- aip_job_nrun 总运行作业数
- aip_job_nsusp 总挂起作业数

每个运行中的作业指标（如果配置了输出每个作业指标）：

- aip_job_cpusec 作业 CPU 使用时间（秒）
- aip_job_cpuut 作业当前 CPU 利用率%
- aip_job_memgb 作业当前内存使用 GB
- aip_job_swapgb 作业虚存使用 GB
- aip_job_iokpbs 作业当前文件 IO kb/s
- aip_job_nthreads 作业当前线程数
- aip_job_gmemmb 作业使用 GPU 显存总量 MB

以上每个指标都带有标签：

- jobid 作业号

5.3.2 cbls

服务进程

cbls - AIP 系统的负载监控 (CBLS)

概要

CB_SERVDIR/cbls [-h] [-V] [-t] [-C] [-D] [-d *env_dir*] [**-** *debug_level*]

描述

CBLS 是一个运行在每个服务器主机上的服务进程。它提供系统配置信息、负载信息服务。所有主机上的 CBLS 协调收集和传输负载信息。负载信息以负载索引向量的形式在 CBLS 之间传输。这些内容在“负载索引”部分中有详细描述。

为了在主机上运行 CBLS，主机必须时集群的成员，配置在*cb.yaml* 中或者动态加入集群（参考*caddhost*）。

每个 AIP 集群都会选举一个主 CBLS。主 CBLS 接收来自所有从属 **CBLS** 的负载信息，并为所有主机提供服务。从属 **CBLS** 会定期检查自身的负载状况，并在观察到负载状况发生显著变化时向主 CBLS 发送负载向量。最小负载信息交换间隔为 15 秒。但是，用户站点可以通过 cb.yaml 里 load_interval 参数来重新定义此间隔。如果负载变化不大，实际交换间隔可以更长。

从属 CBLS 也为本地应用程序提供一些服务。例如，应用程序联系本地 CBLS 以获取本地负载信息、本地集群名称、任何主机上的可用资源以及主 CBLS 所在主机的名称。从属 CBLS 接受应用程序锁定或解锁本地主机的请求。从属 CBLS 还会监视主 CBLS 的状态，并在原主 CBLS 不可用时选举新的主 CBLS。

负载索引

CBLS 提供的负载信息包含以下负载索引：

r15s
: 15 秒内指数平均的 CPU 运行队列长度，已根据 CPU 速度进行归一化。

r1m
: 1 分钟内指数平均的 CPU 运行队列长度，已根据 CPU 速度进行归一化。

r15m
: 15 分钟内指数平均的 CPU 运行队列长度，已根据 CPU 速度进行归一化。

ut
: 过去一分钟内 CPU 利用率的指数平均值，介于 0 到 1 之间。

pg
: 过去一分钟内内存分页速率的指数平均值，以每秒页数为单位。

io
: 过去一分钟内网络 I/O 速率的指数平均值，以每秒千字节为单位。

up
: 主机开机时间（uptime），单位为分钟。

it
主机空闲时间（所有登录会话中未触摸键盘的时间），以分钟为单位。

tmp
/tmp 中的可用磁盘空间大小，以 MB 为单位。

swp
当前可用的交换空间大小，以 MB 为单位。

mem
当前可用的内存大小，以 MB 为单位。

gpu
当前可用的 GPU 数。cbls 根据 $CB_ENVDIR/gpu.yaml 检测主机上安装的 GPU 和各个 GPU 的负载。

此外，AIP 可以配置任意外部负载索引，如下所述。

外部负载信息管理器

可以通过添加自定义资源传感器 RESS 来扩展站点的负载指标监控。详情请参阅*ress*。

选项

-h
将命令用法打印到标准错误输出并退出。

-V
将 Cube 的发行版本打印到标准错误输出并退出。

-t
显示主机信息，例如主机架构、CPU 型号、物理处理器数量、核心数量、GPU 信息等。

-C
检查配置文件内容并显示详细错误报告，然后退出。

-d env_dir
从目录 *env_dir* 读取 **cb.conf**，而不是从默认目录 **/etc** 或 **CB_ENVDIR** 环境变量指定的目录读取。

-D
本机为动态主机，主机名在 cb.yaml 中没有配置。

-debug_level
设置调试级别。有效值为 1 和 2。如果指定，CBLS 将以调试模式运行。在调试模式下，CBLS 使用硬编码的端口号，而不是在系统服务中注册的端口号。此外，任何用户都可以执行特权操作，例如重新配置和主机锁定或解锁。如果未启用调试模式，则只有 root 和 **administrators**（在*cb.yaml* 中定义) 可以执行这些特权操作。如果 *debug_level* 为 1，CBLS 在后台运行，没有关联的控制终端。如果 *debug_level* 为 2，CBLS 在前台运行，并将错误消息打印到 tty。

注意

CBLS 需要对 **/dev/kmem** 或其等效文件的读访问权限。

文件

cb.yaml

5.3.3 cbls.remote

服务进程

cbls.remote - 远程主机负载传感器

概要

CB_SERVERDIR/cbls.remote

描述

cbls.remote 是 AIP 中自定义的远程主机负载传感器。它由 AIP 负载服务器 cbls 定期执行，以获取远程服务器的信息和负载。

cbls.remote 必须位于 **CB_SERVERDIR** 目录中，例如 /opt/skyformai/sbin，并且必须是可执行文件。

CBLS.REMOTE 代码逻辑

cbls.remote 代码逻辑应从所有远程主机获取所需数据。与 RESS 不同，它不应该是无限循环，也就是说，主 cbls 会定期调用它。它只能由主 cbls 调用。它应该输出 YAML 数据，然后退出。

cbls.remote 输出示例:

```
---
remote_hosts:
- name: node0013
  model: x6_5424_IntelRCoreTMi57200UCPU250GHz
  type: x86_64Linux
  status: ok
  cpuf: 54.2
  sockets: 4
  cores_per_socket: 1
  cpus: 4
  maxmem: 1980
  maxswap: 2047
  gpus:
  - id: 0
    model: GeForce RTX 2080 Ti
    totalmem: 11019
  - id: 1
    model: GeForce RTX 2080 Ti
   totalmem: 11019
  ut: 0.23
```

(下页继续)

(续上页)

```
  r1m: 1.23
  mem: 1311.0
  up: 42507
...
```

命令“cbls -y”提供了远程主机数据的示例输出。

输出格式

cbls.remote 的输出应遵循 YAML 标准。第一行始终为：’ remote_hosts:’。然后是一个数据数组。数组中的每个元素代表 cb.yaml 中配置的一个远程主机。

对于每个远程主机，其元素包括：

name:
主机名。主机名必须在 cb.yaml 中配置为远程主机，否则 cbls.remote 中的主机数据将被忽略。此字段为必填字段。

model：
CPU 型号。如果此字段缺失，远程主机将被标记为与主控主机相同的 CPU 型号。CPU 型号可以是任何字符串。此字段为可选字段。

type：
主机类型，例如：x86_64Linux。如果此字段缺失，远程主机将被标记为与主控主机相同的主机型号。此字段为可选字段。

status：
如果主机状态良好，可以运行作业，则该值应为“ok”。否则，请输入其他字符串。此字段为必填字段。

cpuf：
一个数值，表示主机的性能。Linux 中的典型值为“lscpu”输出中的 BogoMIPS 除以 100。如果缺失，则将采用与主控主机相同的值。此字段为可选字段。

sockets：
主机上的 CPU 插槽数量。此字段为可选字段。

cores_per_socket：
每个 CPU 插槽的 CPU 物理核数。此字段为可选字段。

cpus：
逻辑 CPU 核总数。此字段为可选字段。默认值为 1。

maxmem：
主机上安装的最大内存（以 MB 为单位）。此字段为必填字段。

maxswap：
主机上配置的最大交换空间（以 MB 为单位）。此字段为可选字段。

gpus：
GPU 数据数组。此字段为可选字段。

id：
GPU ID。通常为 0、1、2 等。如果存在 gpus 部分，则此字段为必填字段。

model：
GPU 型号。例如：Tesla P100。此字段为可选字段。

totalmem：
GPU 的安装总内存（以 MB 为单位）。此字段为可选字段。

ut:
过去一分钟内 CPU 利用率的指数平均值，介于 0 到 1 之间。此字段为可选字段。

r15s:
15 秒内 CPU 运行队列长度的指数平均值。此字段为可选字段。

r1m:
1 分钟内 CPU 运行队列长度的指数平均值。此字段为可选字段。

r15m:
15 分钟内 CPU 运行队列长度的指数平均值。此字段为可选字段。

pg:
过去一分钟内内存分页速率的指数平均值，以每秒页数为单位。此字段为可选字段。

io:
过去一分钟内网络 I/O 速率的指数平均值，以每秒 KB 为单位。此字段为可选字段。

up:
主机正常运行时间（分钟）。此字段为可选字段。

swap:
可用的交换空间大小，以兆字节为单位。此字段为可选字段。

mem:
可用的内存大小，以兆字节为单位。此字段为可选字段。

tmp:
/tmp 中的可用空间大小，以兆字节为单位。此字段为可选字段。

5.3.4 cbexe

服务进程

cbexe - AIP 系统的远程任务服务

概要

CB_SERVDIR/cbexe [-h] [-V] [-d *env_dir*] [**-***debug_level*]

描述

CBEXE 是一个远程任务执行服务，运行在参与负载分担的每个服务器主机上。它为顺序和并行作业提供远程执行服务。

应用程序可以使用 runtask 命令来连接 CBEXE。程序可以在一个或多个远程主机上顺序或并行启动远程任务。远程任务与客户端具有相同的执行环境，远程任务的状态信息将使用与“wait”(2) 相同的数据结构传递回客户端。信号可以透明地从客户端传递给所有远程任务，并且伪终端可用于支持远程交互式任务。

选项

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 的发行版本打印到标准错误输出并退出。

-d env_dir
从目录 *env_dir* 读取 **cb.yaml**，而不是从默认目录 **/etc** 或环境变量中设置的目录 **CB_ENVDIR** 读取。

-debug_level
设置调试模式和级别。有效值为 1 和 2。如果指定，普通用户可以在调试模式下运行 CBEXE。调试模式下不进行身份验证，因此 CBEXE 只能服务于一个用户。如果 debug_level 为 1，CBEXE 将在后台模式下运行，没有关联的控制终端。如果 debug_level 为 2，CBEXE 将在前台模式下运行，并将错误信息打印到 tty。

另请参阅

cbls、*cb.yaml*、*cadmin*

5.3.5 cbjm

请参考*cbsched*

5.3.6 cbps

服务进程

cbps - AIP 系统的进程监控服务器 (CBPS)

概要

CB_SERVDIR/cbps [-h] [-V] [-d *env_dir*] [**-***debug_level*]

描述

CBPS 是一个守护进程，由参与负载分担的每个服务器主机上的 **CBLS** 启动。CBPS 收集本地主机上运行的进程的资源使用情况。CBPS 收集的信息将被 AIP 的 **CBJM** 和 **CBEXE** 服务用于监控资源消耗并强制执行使用限制。

除非应用程序查询此信息，否则 CBPS 每 3 秒更新一次进程信息。如果应用程序请求该信息，CBPS 也可随时更新进程信息。

进程信息存储在 **/tmp/cbps.info.<hostname>** 中。容器进程的信息存储在 **/tmp/cbps.info.c.<hostname>** 中。CBPS 在启动时也会读取这些文件，以便累积现有进程组的死进程的资源使用情况。

选项

-h
将命令用法打印到 stderr 并退出。

-V
将 AIP 的发行版本打印到 stderr 并退出。

-d env_dir
从目录 *env_dir* 读取 **cb.conf**，而不是从默认目录 **/etc** 或 **CB_ENVDIR** 环境变量指定的目录读取。

-debug_level
设置调试级别。有效值为 1 和 2。如果指定，CBPS 将以调试模式运行。如果 *debug_level* 为 1，CBPS 将运行在后台，没有关联的控制终端。如果 *debug_level* 为 2，CBPS 将运行在前台，并将错误消息打印到 tty。

注意

CBPS 需要对 **/dev/kmem** 或其等效文件的读权限。

文件

/tmp/cbps.info.<hostname>

/tmp/cbps.info.c.<hostname>

5.3.7 cbsched

服务进程

CBSCHED、CBJM - AIP 系统的作业调度器和作业管理器

概要

CB_SERVERDIR/cbsched [-h] [-V] [-C] [-d *env_dir***] [-***debug_level***]**

CB_SERVERDIR/cbjm [-h] [-V] [-d *env_dir***] [-***debug_level***]**

描述

AIP 的每个服务器主机上都有一个作业管理器 CBJM。与底层的主负载服务器 (CBLS) 位于同一主机上的作业管理器负责在其主机上为本地 AIP 集群启动 AIP 调度程序 CBSCHED。CBJM 通常在系统启动时由 **aip.service** 启动。除非使用 **-C** 选项运行 CBSCHED 守护进程，否则切勿启动它。

您可以从任何主机向 CBSCHED 提交作业。CBSCHED 会联系 CBLS，以获取满足作业提交时提供的资源要求的主机的当前负载信息（参见*csub*）。如果未找到合适的主机，作业将被 CBSCHED 保留（即处于等待状态）。找到合适的执行主机后，CBSCHED 会将作业调度到所选主机上的一个或多个 CBJM 进行执行。

CBJM 接受来自 CBSCHED 的作业执行请求，启动作业，并监控作业的进度。根据 CBSCHED 在作业调度时指定的执行条件，CBJM 可以根据主机负载情况、用户交互活动以及队列或主机运行窗口的变化来停止或恢复作业。CBJM 的此类操作会报告给 CBSCHED。作业完成时（成功完成或出现错误，或由用户、AIP 管理员或队列管理员终止），作业输出会通过邮件发送给提交作业的用户，或存储在作业提交时指定的文件中（参见*csub*）。

除了处理用户请求外，CBSCHED 还执行 AIP 的大部分调度功能。AIP 具有容错功能：如果当前 CBSCHED 所在的主机发生故障，则会启动一个新的 CBSCHED，并且除故障主机上运行的作业外，不会丢失任何作业。这些作业可能会在其他主机上从头开始重新运行，或者，如果它们可设置检查点，则可在其他主机上从其最后一个检查点重新启动。

提交作业时，用户的大部分环境信息（例如当前工作目录、掩码和用户组）都会保留，并用于作业的执行。

CB_ENVDIR
配置文件的存储目录，一般为/opt/skyformai/etc。集群中的所有 AIP 服务器主机都应该能够访问目录 **CB_ENVDIR**。目录 **CB_ENVDIR** 必须由 root 用户拥有，并且所有用户都可以访问。

CB_SHAREDIR
调度程序 (CBSCHED)，一般为/opt/skyformai/work，存储作业数据和统计文件的目录。实际文件存储在目录 **CB_SHAREDIR/data** 中。目录 **CB_SHAREDIR** 应该能够从集群中的所有 AIP 服务器主机访问。目录 **CB_SHAREDIR** 必须由 root 用户拥有，并且所有用户都可以访问。目录 **CB_SHAREDIR/data** 必须由 AIP 主（第一）管理员拥有并可写入，并且所有用户都可以访问。

选项

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 的发行版本号打印到标准错误输出 (stderr) 并退出。

-C
此选项仅适用于 CBSCHED。如果指定，CBSCHED 将对 AIP 配置文件进行语法检查，并将详细信息打印到标准输出 (stdout)。检查完成后，CBSCHED 将退出。带有 **-C** 选项的 CBSCHED 不必在主控主机上运行。除非使用 **-C** 选项，否则切勿手动启动 CBSCHED。

-d env_dir
从目录 *env_dir* 读取 **cb.yaml** 文件，而不是从默认目录 **/etc** 或环境变量中设置的目录 **CB_ENVDIR** 读取。

-debug_level
调试模式级别。可能值为 1 或 2。设置调试模式后，守护进程将以调试模式运行，并可由普通用户（非 root 用户）启动。如果调试级别为 1，CBJM 启动后将转至后台运行。如果调试级别为 2，CBJM 将保持在前台运行。CBSCHED 始终由 CBJM 使用相同选项启动，因此具有相同的调试级别。

队列和日志

用户可以看到许多作业队列，作业可以提交到这些队列中。作业队列由 AIP 管理员在集群配置文件 **cb.yaml** 或 **queue.yaml** 中定义。有关队列配置的说明，请参阅*cb.yaml*。

CBSCHED 维护几种类型的数据文件：事件数据（cb.data*）、作业统计数据（cb.acct*）、作业流数据（cb.stream.*）。

错误报告

CBSCHED 和 CBJM 没有控制终端。错误会以日志级别写入文件 **/opt/skyformai/log/cbsched.<***hostname***>.log** 和 **/opt/skyformai/log/cbjm.<***hostname***>.log**。

文件

CB_ENVDIR/cb.yaml

CB_ENVDIR/queue.queues

CB_ENVDIR/ug.hosts

CB_ENVDIR/hg.users

CB_SHAREDIR/data/cb.data. 时间戳

CB_SHAREDIR/data/cb.acct. 序号

另请参阅

cb.yaml、*cb.acct*、*cbls*

5.3.8 cbtt

系统命令

cbtt - 用于 AIP 交互作业的网络 I/O 服务器

概要

CB_SERVERERDIR/cbtt

CB_SERVERDIR 一般为/opt/skyformai/sbin

描述

cbtt 是一个终端服务器，由 AIP 在首次启动远程执行时启动。**cbtt** 与客户端应用程序并发运行，透明地处理所有 I/O 以及往返于远程任务的信号。**cbtt** 是一个基于每个应用程序的服务器进程，也就是说，如果同一个应用程序有多个远程任务，则这些任务在客户端主机上共享同一个 **cbtt**。**cbtt** 在应用程序终止时终止。**cbtt** 在应用程序退出时退出。

I/O 重排
: **cbtt** 是应用程序所有远程任务的 I/O 代理。默认情况下，**cbtt** 从标准输入读取数据并将流转发给所有远程任务。**cbtt** 将所有远程任务的输出打印到标准输出 (stdout) 和标准错误 (stderr)。可以禁用或启用特定远程任务的标准输入。这样可以将标准输入转发到选定的远程任务。也可以禁用或启用远程任务读取标准输入。这对于某些应用程序（例如分布式 Shell）是必需的，因为 Shell 会从标准输入读取命令行，并将任务调度到远程主机执行，在此期间远程任务必须读取标准输入。

TTY 模式
: 当分布式应用程序在远程主机上运行时，**cbtt** 可以在本地模式和远程模式之间切换 TTY。在本地模式下，所有控制键均在本地解释。在远程模式下，输入流直接传递给远程任务。用户通常无需了解这些细节。

作业控制
: **cbtt** 透明地支持作业控制。它捕获信号（例如 SIGTSTP、SIGINT 和 SIGTERM）并将其传递给远程任务。因此，当用户按下作业控制键（例如 Ctrl-Z 或 Ctrl-C）时，远程任务会收到 SIGTSTP 或 SIGINT 信号。借助 **cbtt** 的支持，远程任务可以像在本地运行一样被暂停或恢复、置于后台或前台以及终止。

cbtt-客户端协调
对于需要在远程主机上使用伪终端的应用程序，必须小心同步 tty 的本地/远程模式设置。

注意

cbtt 必须安装 **CB_SERVERDIR** (/opt/skyformai/sbin) 中。

另请参阅

csub、*runtask*

5.3.9 echkpnt

系统命令

echkpnt - 调度器用户执行检查点操作的定制化命令

概要

echkpnt [**-k**] [**-d** *checkpoint_dir*] [**-x**] *process_group_ID*

echkpnt [**-h** | **-V**]

描述

echkpnt 是用于对作业执行检查点操作的接口。

默认情况下，作业在执行检查点操作后会继续执行。

概要中的某些选项仅受某些操作系统支持。

echkpnt

echkpnt 位于 CB_SERVERDIR (/opt/skyformai/sbin) 目录下。

echkpnt 将通过作业提交提供的检查点指令或通过队列参数发送到您定义的自定义 echkpnt 方法。通信通过概要中指示的通用语法实现。

echkpnt.method_name

echkpnt.*method_name* 是一个自定义 echkpnt，它尝试运行自定义检查点程序或脚本。

自定义 echkpnt 方法的名称必须是 echkpnt.*method_name*，其中 *method_name* 是您分配给自定义 echkpnt 的名称。method_name 必须使用 CB_ECHKPNT_METHOD 作为环境变量指定，或在使用 csub -k 选项提交作业时指定，例如 csub -k “mydir 120 method=myapp” job1。

echkpnt.*method_name* 必须支持概要中描述的语法。

echkpnt.*method_name* 位于 CB_SERVERDIR (/opt/skyformai/sbin) 中。

选项

-k
成功完成检查点后终止作业。如果指定此选项，并且检查点因任何原因失败，作业仍会继续执行。

-d *checkpoint_dir*
指定检查点目录 checkpoint_dir。请指定相对或绝对路径名。

作业检查点后，检查点信息存储在 *checkpoint_dir*/*job_ID*/*file_name* 中。多个作业可以在同一目录中执行检查点操作。系统可以创建多个文件。

检查点目录用于重新启动作业（参见*crestart*）。

-x
此选项仅用于兼容性。应忽略。

process_group_ID
要执行检查点操作的进程或进程组的 ID。

-h
将命令用法打印到标准错误输出并退出。

-V
将 AIP 发行版本打印到标准错误输出并退出。

另请参见

csub、*cchkpnt*、*crestart*、*erestart*

诊断

echkpnt

如果检查点操作成功，则返回 0。否则，返回 echkpnt.default 或 echkpnt.*method_name* 的值。

echkpnt.default

这应该链接到一个特定的 echkpnt.method，并设置为默认的检查点方法。

echkpnt.method_name

至少应该存在一个自定义检查点方法。如果 echkpnt.*method_name* 成功完成作业检查点，则退出并返回 0。非零值表示作业检查点失败。所有写入 stdout 和 stderr 的消息都将定向到 /dev/null 并被 AIP 忽略。

要保存 echkpnt.*method_name* 的标准错误和标准输出消息，请在提交作业之前设置 CB_ECHKPNT_KEEP_OUTPUT=y。echkpnt.*method_name* 生成的 stdout 和 stderr 输出将被重定向至：

- *checkpoint_dir*/$CB_JOBID/echkpnt.out
- *checkpoint_dir*/$CB_JOBID/echkpnt.err

限制

如果您使用 echkpnt.*method_name*，则作业提交后，将使用提交时指定的方法或参数 CB_ECHKPNT_METHOD 对作业进行检查点检查。您无法使用 bmod 命令更改该方法。

如果您提交作业但未指定自定义方法，并且 CB_ECHKPNT_METHOD 未定义，则将使用 echkpnt.default。您无法使用 bmod 更改此设置。

集群管理员有责任确保方法名称和方法目录组合在集群中是唯一的。

5.3.10 erestart

系统命令

erestart - 调度器用于重新启动检查点作业的定制化命令

概要

erestart [**-c**] [**-f**] *checkpoint_dir*

erestart [**-h** | **-V**]

描述

erestart 是一个接口，用于重新启动已执行检查点的作业。

作业将使用保存在 *checkpoint_dir*/*last_jobID* 中的检查点文件重新启动。作业将被重新提交，分配一个新的作业 ID，并使用新的作业 ID 将检查点目录重命名为 *checkpoint_dir*/*new_jobID*。

erestart

erestart 位于 CB_SERVERDIR (/opt/skyformai/sbin)。

erestart 将已执行检查点操作的作业的进程 ID 和进程组 ID 发送给 erestart.default。erestart 会设置环境变量 CB_RESTART_PID 和 CB_RESTART_PGID。

erestart 会将作业重启请求发送给 erestart.default 或您定义的自定义 erestart 方法。通信通过概要中所示的通用语法实现。

erestart.default

erestart.default 位于 CB_SERVERDIR 目录下，是指向 erestart.*method* 的链接。erestart.default 会重启该作业。

erestart.method_name

erestart.*method_name* 是一个自定义的 erestart 应用，其中 *method_name* 与其对应的 echkpnt.*method_name* 中指定的名称相同。

erestart.*method_name* 位于 CB_SERVERDIR 目录下。所有需要运行自定义 echkpnt 和 erestart 程序的用户都必须能够访问检查点方法目录。

erestart.*method_name* 必须支持概要中描述的语法。

erestart.*method_name* 创建文件 checkpoint_dir/$CB_JOB_ID/.restart_cmd，并将重新启动作业或进程组的命令写入此文件中，格式为 "CB_RESTART_CMD=*restart_command*"。CB_RESTART_CMD 必须使用引号 (")。

erestart 会尝试在.restart_cmd 中查找用于重新启动作业的命令。如果找不到该命令，它将使用原始作业文件和命令来重新启动作业。原始作业文件是在作业提交时在检查点目录中创建的。

如果 erestart 在.restart_cmd 中找到用于重新启动作业的命令，它将调用该命令来重新启动已设置检查点的作业。

erestart.*method_name* 必须能够访问用于启动作业的原始命令行。

erestart.*method_name* 必须返回，它不应运行应用程序来重新启动作业。

参数

如果您的应用程序需要其他参数，您可以使用以下环境变量来得到有关作业的更多信息：

CB_JOBID——分配给要重新启动的作业的新作业 ID

CB_OLD_JOBID——执行检查点后要重新启动的作业的作业 ID。

CB_ERESTART_USRCMD——作业提交时用于启动作业的原始命令。此环境变量由 erestart 设置，并由 erestart.*method_name* 读取。

CB_JOB_STARTER——以实际作业作为参数的可执行程序。仅当为提交作业的队列定义了作业启动器时，此环境变量才存在。

选项

-c
: 此选项仅在某些操作系统（例如 Convex）上受支持。

  将执行检查点的进程正在使用的所有文件复制到检查点目录。

-f
: 此选项仅用于兼容性，应忽略。

checkpoint_dir
: 指定检查点目录。请指定相对或绝对路径名。

  当作业执行检查点操作时，检查点信息存储在 *checkpoint_dir*/job_ID/*file_name* 中。多个作业可以在同一目录中执行检查点操作。系统可以创建多个文件。

  检查点目录用于重新启动作业（参见*crestart*）。

-h
: 将命令用法打印到标准错误输出并退出。

-V
: 将 AIP 发行版本打印到标准错误输出并退出。

另请参见

csub、*echkpnt*、*crestart*、*echkpnt*

诊断

erestart

如果作业重新启动成功，则退出并返回重新启动的作业的值。

如果作业重启由于 erestart.default 或 erestart.*method_name* 失败，则退出并返回 erestart.default 或 erestart.*method_name* 的值。如果作业重启由于其他原因失败，则退出并返回 -1。

erestart.default

如果使用了 AIP 的默认重启程序 erestart.default，并且作业成功重启，则 erestart.default 不会在其标准错误中打印除“pid=new_ pid pgid=new_pgid”之外的任何其他消息。如果 erestart.default 在其标准错误中打印了任何其他消息，则 AIP 认为作业重启失败。

erestart.method_name

如果使用了自定义的 erestart 程序（在检查点方法中，指定为 csub -k “mydir method=myapp” job1），如果 erestart.*method_name* 成功将作业重启命令写入，则退出时返回 0。文件 *checkpoint_dir*/$CB_JOB_ID/.restart_cmd，或者如果它故意不向文件写入任何内容。非零值表示 erestart.*method_name* 无法重新启动作业（作业重新启动失败）。

对于用户级检查点，erestart.*method_name* 必须收集作业的退出代码。然后，erestart.*method_name* 必须以与作业相同的退出代码退出。否则，作业的退出状态将无法正确报告给 AIP。内核级检查点的工作方式不同，不需要来自 erestart.*method_name* 的这些信息来重新启动作业。

所有写入 stdout 和 stderr 的消息都将定向到 /dev/null 并被忽略。

保存 echkpnt 的标准错误和标准输出消息。*method_name* 和 *erestart.method_name*，在运行 crestart 之前设置 CB_ECHKPNT_KEEP_OUTPUT=y。erestart. *method_name* 生成的 stderr 输出将被重定向至：

- *checkpoint_dir*/$CB_JOBID/erestart.out
- *checkpoint_dir*/$CB_JOBID/erestart.err

5.3.11 esub

系统命令

esub、**eexec** - 作业提交和执行时的外部可执行文件

概要

CB_SERVERDIR/esub

CB_SERVERDIR/eexec

CB_SERVERDIR 一般为：/opt/skyformai/sbin

描述

管理员可以编写这些外部提交 (esub) 和执行 (eexec) 可执行文件，以对作业执行特定于站点的操作。

CB_SERVERDIR 一般为：/opt/skyformai/sbin

提交作业时（参见*csub* 和*crestart* ），如果在 CB_SERVERDIR 中找到 esub，则会执行 eexec。在执行主机上，eexec 会在作业启动、完成以及启动检查点时运行。环境变量 **LS_EEXEC_T** 分别设置为 **START**、**END** 和 **CHKPNT**，以指示何时调用 eexec。如果 esub 需要向 eexec 传递一些数据，esub 可以简单地将数据写入标准输出；eexec 可以通过其标准输入获取数据。因此，AIP 实际上是在“esub | eexec”中实现了管道。

在设置作业的环境变量后，eexec 会以用户身份执行。父作业进程会等待 eexec 完成后再继续执行；因此，eexec 预计会完成。

调用 eexec 的进程的 pid 存储在环境变量 **CB_JOBPID** 中。如果 eexec 旨在监视作业的执行情况，则 eexec 必须 fork 一个子进程，然后让父 eexec 进程退出。在监视作业执行的同时，eexec 子进程应定期通过测试 **CB_JOBPID** 变量来检查作业进程是否仍然处于活动状态。作业完成后，eexec 子进程应退出。

除了 csub 和 crestart 之外，cmod 还会调用 esub。调用 esub 时，环境变量 **CB_SUB_ABORT_VALUE** 和 **CB_SUB_PARM_FILE** 会被设置。如果 esub 退出时返回 **CB_SUB_ABORT_VALUE** 的值，则作业请求将被中止。**CB_SUB_PARM_FILE** 指定包含作业请求参数的临时文件的名称。有关此文件的内容，请参阅 **PARAMETER FILE** 部分。**CB_SUB_ABORT_VALUE** 和 **CB_SUB_PARM_FILE** 变量的典型用途是禁用某些提交选项。

在交互式远程执行中调用 esub 时，**CB_SUB_ABORT_VALUE** 和 **CB_SUB_PARM_FILE** 变量未定义。

esub 可以通过写入环境变量 CB_SUB_MODIFY_FILE 指定的文件来更改任何作业选项。例如:

```
#!/bin/bash
# esub 修改 -R 选项的示例
source $CB_SUB_PARM_FILE          # 环境变量中获得所有作业提交参数
echo 'CB_SUB_RES_REQ="select[mem>100]"' > $CB_SUB_MODIFY_FILE # 修改csub -R 参数
exit 0
```

请注意，参数值需要用双引号括起来。

参数文件

作业提交和修改参数文件包含一行，格式为 **option_name=value**，用于指定每个作业选项。**value** 为“SUB_RESET” 表示此选项已被 cmod 重置。如果 esub 是 Bourne Shell 脚本，则运行 “.$CB_SUB_PARM_FILE” 将使 esub 可以使用这些变量。**option_name** 包括：

CB_SUB_JOB_NAME
为指定的作业名称 (csub -J)。

CB_SUB_QUEUE
为指定的队列名称 (csub -q)。

CB_SUB_IN_FILE
为指定的标准输入文件名 (csub -i | -is)。

CB_SUB_OUT_FILE
为指定的标准输出文件名 (csub -o | -oo)。

CB_SUB_ERR_FILE
为指定的标准错误文件名 (csub -e | -eo)。

CB_SUB_ADDITIONAL
为指定的 additional_esub_information(csub -a)。

CB_SUB_EXCLUSIVE
为“Y”表示独占执行 (csub -x)。

CB_SUB_NOTIFY_END
为“Y”，表示作业结束时发送邮件通知 (csub -N)。

CB_SUB_NOTIFY_BEGIN
为“Y”，表示作业开始时发送邮件通知 (csub -B)。

CB_SUB_USER_GROUP
表示指定的用户组名称 (csub -G)。

CB_SUB_CHKPNT_PERIOD
表示指定的检查点周期 (csub -k)。

CB_SUB_CHKPNT_DIR
表示指定的检查点目录 (csub -k)。

CB_SUB_COMMANDNAME
表示作业命令的第一个参数 (csub 作业命令第一个词)。

CB_SUB_COMMAND_LINE
表示作业命令及其参数。使用 vncsub/dcvsub 提交作业时，此变量会被省略。

CB_SUB_RESTART_FORCE
为“Y”表示强制重启作业 (crestart -f)。

CB_SUB_RESTART
为“Y”表示重启作业 (crestart)。

CB_SUB_RERUNNABLE
为“Y”表示可重新运行作业 (csub -r)。

CB_SUB_HOST_SPEC
表示指定的主机规范。

CB_SUB_DEPEND_COND
表示指定的依赖条件 (csub -w)。

CB_SUB_RES_REQ
表示指定的资源需求字符串 (csub -R)。

CB_SUB_PRE_EXEC
表示指定的预执行命令 (csub -E)。

CB_SUB_LOGIN_SHELL
是指定的登录 shell(csub -L)。

CB_SUB_MAIL_USER
是指定的发送电子邮件的用户 (csub -u)。

CB_SUB_MODIFY
为“Y”表示修改请求 (cmod)。

CB_SUB_MODIFY_ONCE
为“Y”表示修改一次请求 (cmod)。

CB_SUB_PROJECT_NAME
是指定的项目名称 (csub -P)。

CB_SUB_INTERACTIVE
为“Y”表示交互式作业 (csub -I)。

CB_SUB_PTY
为“Y”表示支持 PTY 的交互式作业 (csub -Ip)。

CB_SUB_PTY_SHELL
为“Y”表示支持 PTY Shell 的交互式作业 (csub Is)。

CB_SUB_HOSTS
是执行主机名列表 (csub -m)。

CB_SUB_NUM_PROCESSORS
是请求的最小处理器数量 (csub -n)。

CB_SUB_MAX_NUM_PROCESSORS
是请求的最大处理器数量 (csub -m)。

CB_SUB_BEGIN_TIME
为开始时间，以秒为单位，自 1970 年 1 月 1 日 00:00:00 GMT 时间起计算 (csub -b)。

CB_SUB_TERM_TIME
为终止时间，以秒为单位，自 1970 年 1 月 1 日 00:00:00 GMT 时间起计算 (csub -t)。

CB_SUB_OTHER_FILES
为“SUB_RESET”，以指示正在执行 cmod 操作以重置要传输的文件数 (csub/cmod -f)。

CB_SUB_OTHER_FILES_nn
nn 是指示特定文件传输的索引号。**值**是指定的文件传输表达式。例如，对于“csub -f “a > b” -f “c < d””，定义如下：

```
CB_SUB_OTHER_FILES_0="a > b"
CB_SUB_OTHER_FILES_1="c < d"
```

CB_SUB_RLIMIT_CPU
是指定的 CPU 限制 (csub -c)。

CB_SUB_RLIMIT_FSIZE
是指定的文件大小限制 (csub -F)。

CB_SUB_RLIMIT_DATA
是指定的数据大小限制 (csub -D)。

CB_SUB_RLIMIT_STACK
是指定的堆栈大小限制 (csub -S)。

CB_SUB_RLIMIT_CORE
是指定的核心文件大小限制 (csub -C)。

CB_SUB_RLIMIT_RSS
是指定的驻留大小限制 (csub -M)。

CB_SUB_RLIMIT_RUN
是指定的挂钟运行限制 (csub -W)。

CB_SUB2_CWD
是指定的当前工作目录 (csub -cwd)。

CB_SUB2_JOB_DESC
是指定的作业描述 (csub -Jd)。

CB_SUB2_APP
是指定的应用程序名称 (csub -A | -app)。

CB_SUB2_JOB_GROUP
是指定的作业组名称 (csub -g)。

另请参阅

csub , *crestart* , *cmod* 。

5.3.12 host-setup

AIP 安装命令

host-setup - AIP 软件在每台主机上执行的安装脚本

概要

AIP 安装的共享目录/**host-setup** [**--shared=** *共享目录*] [**--distro=** *适合本机的 AIP 的软件的子目录名*] [**--deploylocal**] [**--gui|--nogui**] [**--sshcontrol**] [**--dynamic** [**--mxj=** *主机最大作业槽*]] [**--upgrade**] [**--help**]

描述

AIP 软件用 install 脚本安装到共享文件系统中后，每台主机用 host-setup 脚本配置 AIP 服务和用户使用 AIP 的环境变量。

host-setup 脚本被安装在 AIP 共享文件目录中。在 AIP 集群的每台主机上必须以 root 身份运行，主机才可以加入到集群中。

选项

--help
显示所有参数项，并退出。

--deploylocal
把 sbin 里的服务进程安装到本地，这样做的目的是为了较少 AIP 服务对共享存储的依赖并增加的负担。对于超过 500 个主机的集群，建议安装到本地。

缺省：sbin 用符号链链接到本地目录。

--distro= 适合本机的 *AIP* 的软件的子目录名
在安装好的 AIP 共享目录中，每个 AIP 版本以及针对不同架构 CPU 的二进制可执行文件都被放到一个单独的子目录中。目录的名字为：**版本** [架构]。

例子:

```
10.25.0/        # AIP 10.25.0 x86_64 Linux内核3.x以上
10.25.0rhel6/   # AIP 10.25.0 x86_64 Linux内核2.x
10.25.0aarch64/ # AIP 10.25.0 ARM 64 Linux内核3.x以上
```

缺省：无，脚本停顿要求用户选择。如果系统中只有一个版本，则被自动选用。

--dynamic [--mxj= 主机最大作业槽]]
这个选项在公有云上建立弹性集群所用。这个选项把本主机自动加到 AIP 的集群给中。

因为动态主机不在*cb.yaml* 里配置，如果 maxslots 不与 CPU 核数一致，可用参数–mxj 定义本机的 maxslots。

缺省：主机不是动态主机，只有在 cb.yaml 中配置的主机才能加入到 AIP 集群中。

--gui|--nogui
增加一些设置本机为图形服务器的功能，安装并启动 SkyForm CRV 服务，参考*vncsub*。这个选项只在第一次安装有效。

缺省：–nogui，即本机不运行图形作业。

--shared= 共享目录
这个选项一般不需要使用，host-setup 会根据自己所在的目录自动决定这个参数的值。

缺省：host-setup 所在的目录。

--sshcontrol
设置 PAM 模块以便控制用户的 ssh 权限，参考*cb.yaml*。

缺省：不设置控制用户 ssh 权限的 PAM 模块。

--upgrade
设置主机后不重启 AIP 服务，以防本主机上运行的作业收到影响。管理员可以在合适的时候用管理命令 (*cadmin* 和*csadmin*) 来重启 AIP 的服务。

缺省：重启 AIP 服务。

另请参阅

install

5.3.13 install

AIP 安装命令

install - AIP 软件包的安装脚本

概要

install [**--shared=** 共享目录] [**--localtop=** 本地安装目录] [**--hosts=** 主机 *hosts* 文件路径] [**--admin=***AIP* 集群主管理员用户名] [**--ncpus=threads|cores**] [**--crv_port=***CRV* 服务端口] [**--eda**] [**--noint|--silent**] [**--help**]

描述

AIP 软件包首层目录中的 install 脚本用于把 AIP 软件安装到共享文件系统中。这个脚本也可用于升级，包括同版本更新（覆盖原有的可执行文件）或新的版本的安装。

install 脚本必须以 root 身份运行。

第一次安装 AIP，脚本会安装缺省配置文件。如果当前目录下没有 AIP 企业版的 key 文件，会自动生成有效期为 45 天的企业版 key，key 可以提供 hosts 文件中配置的主机数乘以 10 的单元。如 hosts 文件中有 3 台主机，则 key 里包含 30 个单元。

企业版每一颗 CPU 和一颗 GPU 需要一个 key 的单元。

若共享文件的目录下已有以前版本的 AIP 文件，install 脚本不会修改任何配置文件。

当同版本的 AIP 有更新的软件包时，可以用这个脚本更新和覆盖已经安装的二进制文件。

当安装新版的 AIP 时，脚本会把新的版本安装在一个新的目录中，如/opt/skyformai_shared/10.25.0。这里/opt/skyformai_shared 为共享文件系统中的目录，10.25.0 为新安装的目录。

选项

--help
显示所有参数项，并退出。

--admin=*AIP* 集群主管理员用户名
指定集群主管理员用户名。如果过用户名在系统中不存在，脚本会调用 useradd 添加本地用户。

缺省：cadmin

--crv_port=*CRV* 服务端口]
指定 SkyForm CRV 服务（参考*vncsub*）的端口号。这个选项只在第一次安装有效。因为后继的安装为更新或升级，不会修改已有的配置。

缺省：16000

--eda
生成适合 EDA 负载的配置文件，参考典型的 *EDA* 负载集群配置 。这个选项只在第一次安装有效。因为后继的安装为更新或升级，不会修改已有的配置。

缺省：适合超智算中心的配置。参考典型的超算中心和智算中心集群配置 。

--hosts= 主机 *hosts* 文件路径
提供集群中所有主机的 IPv4 地址和主机名列表文件路径。文件例子如下。

hosts:

```
192.168.20.100 mgt01
192.168.20.101 mgt02
192.168.20.102 node0001
192.168.20.103 node0002
192.168.20.103 gpu0001
```

缺省文件路径：当前目录下./hosts。

缺省：若文件不存在，以本机为目标自动生成单一主机的 hosts 文件。

--localtop= 本地安装目录
安装在本机的目录。这个目录在同一集群的所有异构主机上都是一致的，里面的 bin、sbin 等目录为符号链，链接到共享文件系统中适合本机的二进制文件目录，如在 arm 的主机上：bin -> /opt/skyformai_shared/10.25.0aarch。

缺省：/opt/skyformai

--ncpus=threads|cores
配置 cb.yaml 中集群级别的参数 define_ncpus 的值。参考*cb.yaml* 。这个选项只在第一次安装有效。因为后继的安装为更新或升级，不会修改已有的配置。

--noint|--silent
脚本运行不提问确认。除了用参数修改的安装方法外，所有参数都用缺省值。这个参数可用于自动化部署中。

缺省：参数需要用户确认。

--shared= 共享目录
所安装的共享文件系统上的路径。所有 AIP 软件都会被安装在这个目录下。

缺省：/opt/skyformai_shared

另请参阅

host-setup

5.3.14 jservice

服务进程

jservice - AIP 的作业状态服务 (JSERVICE)

概要

CB_SERVDIR/jservice [-h] [-V] [-d *env_dir*] [**-***debug_level*]

描述

JSERVICE 是一个运行在 AIP Master 主机上的服务器，仅提供“bjobs”命令的作业状态信息。

在作业吞吐量极高的环境中，它可以减轻 AIP 调度程序 cbsched 的负担，从而使 cbsched 有带宽来调度作业。

在作业吞吐量极高的环境中，还建议禁用 cbcrond，方法是在 /opt/skyformai/etc/jservice.yaml 中添加参数“cbcrond_enabled: no”。

jservice 服务由配置文件 /opt/skyformai/etc/jservice.yaml 控制。默认情况下，该服务未启用，即该服务未运行。

要启用它，请在 jservice.yaml 中添加参数“jservice_enabled: yes”，然后在 AIP 主服务器上重启守护进程 cbjm，方法如下：重启整个 AIP 服务：“systemctl restart aip”，或重启 cbjm：“csadmin jmrestart”。

启用后，命令“bjobs”（cjobs 的符号链接）将开始联系 jservice 获取作业信息，而不是从 AIP 调度程序 cbsched 获取作业状态。

配置文件 /opt/skyformai/etc/jservice.yaml

jservice 的行为由配置文件 /opt/skyformai/etc/jservice.yaml 控制。该文件包含以下参数：

jservice_enabled
值为“yes”表示 jservice 由 cbjm 在 Master 主机上自动启动。默认情况下，该参数为“no”，即禁用。

cbcrond_enabled
值为“yes”表示 cbcrond 由 cbjm 在 Master 主机上自动启动。默认情况下，该参数为“yes”。

jobinfo_interval
jservice 与 cbsched 同步作业状态的时间间隔（以秒为单位）。默认情况下，间隔为 30 秒。同步的数据来自 cbsched 定期输出的流数据，存放在/tmp/.aipestream 下。*cbsched* 输出的流数据的间隔由*cb.yaml* 中的 **estream_interval** 参数指定，缺省为 30 秒。

jobclean_interval
jservice 执行作业清理周期的时间间隔（以秒为单位）。这是 jservice 清理内存中已完成作业的时间，这些作业的时间超过“keep_finishjob_period”（见下文）。默认情况下，该值为 600，即 10 分钟。

keep_finishjob_period
jservice 将已完成作业保留在其内存中的时长（以秒为单位）。超过此时间后，已完成作业将从 jservice 的内存中删除。默认情况下，持续时间为 4 小时，即 14400。

max_requests_on_fork
每次处理收到作业信息请求时（fork 一个子进程），在一个子进程中处理最多的请求数。缺省为 10，即一个子进程会最多并行处理 10 个作业信息请求。这个值最多不能超过 256。每个子进程中使用线程（thread）并行处理请求，每个线程会占用一个 CPU 核，最佳实践是这个值应该是 master 主机的总 CPU 核数减去 5，即保留 5 个核给调度器和操作系统。

timing_level
如果需要在日志中输出每次 fork 子进程处理请求的时间，这个值需要设成 **1**。缺省：没有关于时间信息的日志。

> **警告：** 这个值设置后日志会有大量输出，建议一般环境把它注释。修改 jservice.yaml 后可以运行命令 **csadmin jsrestart** 重启 jservice。

use_estream
是否使用 AIP 的扩展 stream 数据。不使用时定期的作业数据同步会对调度器产生一定负载。

缺省：no (不使用）

选项

-h
将命令用法打印到标准错误输出 (stderr) 并退出。

-V
将 AIP 发行版本打印到标准错误输出 (stderr) 并退出。

-d env_dir
从目录 *env_dir* 读取 **jservice.yaml**，而不是从

默认目录 **/etc** 或 **CB_ENVDIR** 环境变量指定的目录读取。

-debug_level
设置调试级别（debug_level)。有效值为 2。如果指定，JSERVICE 将以调试模式运行。在调试模式下，服务器在前台运行，并将日志打印到标准错误输出 (stderr)。

文件

CB_ENVDIR/jservice.yaml 和 **CB_ENVDIR/cb.yaml** (*cb.yaml*)

控制

只有集群管理员可以控制 jservice。

重启命令：csadmin jsrestart

配置参数 jservice.yaml 修改后生效：csadmin jsreconfig

5.3.15 ress

服务进程

ress - SkyForm AIP 系统的资源传感器 (RESS)

概要

CB_SERVERDIR/ress.[master|host|*hostname*]

描述

RESS 是 AIP 负载服务器 (cbls) 的定制资源插件。它向 AIP 系统提供定制的资源信息，供作业使用。它是一个定制编写的程序，由 CBLS 调用，用于获取特定于站点的资源数据。

RESS 必须位于 **CB_SERVERDIR** 目录 (/opt/skyformai/sbin) 中，并且必须可执行。

RESS 必须具有以下文件名之一：

- **ress.master**，将由 CBLS 在主主机上执行。
- **ress.host**，将由 CBLS 在每个 AIP 主机上执行。
- **ress.***hostname*，将由 CBLS 在主机名为 *hostname* 的主机上执行。

CBLS 会定期在 $CB_SERVDIR 目录中查找可执行名称 ress.master、ress.host 和 ress.*hostname*，如果存在，则自动执行。

RESS 代码逻辑

RESS 代码逻辑应为一个循环，定期更新资源数据。在每个循环中，它会在其标准输出上提供 YAML 格式的资源数据，然后休眠几秒钟。之后，它会再次更新资源。

对于不经常更新的资源，两次循环之间的休眠时间可能很长，例如几个小时。

最短的数据更新时间为 **5** 秒。

它提供的资源数据必须是 YAML 格式。详细格式将在下一节中解释。

RESS 代码示例，**ress.master**：

```
#!/bin/bash
while true; do
  echo "- resource: clksnd"
  echo " description: 当前时钟的秒值"
  echo " type: number"
  echo " direction: increase"
  echo " value:" \`date +%S\`
  echo " locale: master node01"
  echo "---"
  sleep 10
done
```

输出格式

对于每次循环，RESS 在标准输出中的输出必须是完整的 YAML 文档。该文档应以一个序列开头，并以 “—” 作为分隔行结尾。这告诉 CBLS 停止读取内容并等待下一个循环。

每个序列描述一个资源。可以描述多个资源。

每个资源的属性包括：

resource:
定义资源的名称。资源名称的长度不应超过 32 个字符。必须以字母开头，且不能包含任何 **.!-=+*/[]@:&|{}’ “’** 字符。

description:
资源的描述。可以包含任何字符。总长度必须为 255 个字符或更短。

type:
资源类型。有效值为：

number，表示资源值为数字（整数或浮点数）；

text，表示资源值为少于 32 个字符的自由文本；

tag，表示资源不为值，资源名称将作为主机的标签。作业可以使用标签（即资源名称）来选择主机。

direction:
方向表示资源从最佳状态向最差状态移动的方向。其值为 “增加” 或 “减少”。“增加” 值表示数字越小，系统拥有的资源越多。例如，CPU 利用率一个 “增加” 类型的资源。“减少” 类型的资源表示数字越大，系统拥有的资源越多。例如，空闲内存就是一个 “减少” 类型的资源。此参数仅适用于类型为 “number” 的资源。

release
取值为 “yes” 或 “no”。当 **release** 指定为 “yes” 时，当请求该资源的作业被暂停时，资源会被释放。默认值为 “no”，即作业暂停时不会释放资源。

assign
仅适用于类型为 “number” 的资源。取值为 “yes” 或 “no”。当 **assign** 指定为 “yes” 时，资源单元会被枚举，调度程序会将特定的资源单元分配给请求该资源的作业。例如，GPU 的资源 “assign” 设置为 “yes”。如果某个作业在主机上请求 2 个 GPU 资源，调度程序将为该作业分配特定的 GPU 编号，例如 GPU 2 和 GPU 3。这样，该作业就知道要使用哪些 GPU，以避免与同一主机上其他请求 GPU 的作业发生冲突。默认值为 “no”，即不为每个任务（作业槽位）分配资源单元。

slotresource：
仅适用于 “number” 资源。如果该值设置为 “yes”，则为该作业预留的资源总量等于作业资源需求中请求的值乘以该作业调度的槽位数量。默认值为 “no”，即为该作业预留的资源总量与资源需求中指定的值相同。

value：
资源的当前值。对于 “number” 资源，该值可以是数字；对于 “text” 资源，该值可以是文本字符串。对于 “tag” 资源，该值无意义。

locale
表示本地

报告资源的位置。如果缺少此参数，系统将认为资源值是运行 RESS 的主机的地址。该地址是共享该资源的主机列表。对于主机基础资源，此参数可以是报告资源值的主机名，也可以不填。对于集群范围的资源，请使用保留字 “all”。如果某个资源由多个主机共享，请列出这些主机的主机名，并用空格分隔，以分钟为单位。

> **警告：** locale 必须是 yaml 输出中的最后一个字段。

备注： 如果资源以在 cb.yaml 中定义，RESS 的输出只需 resource 和 value 两个字段即可。

示例

以下示例报告了两个资源：一个“number”资源和一个“text”资源。这两个资源的“locale”都是运行 RESS 的主机。*hostname*。

```
#!/bin/bash
while true; do
  echo "- resource: gpu1"
  echo "  description: GPU availability"
  echo "  type: number"
  echo "  direction: increase"
  echo "  value: 0.8
  echo "  release: yes"
  echo "- resource: network"
  echo "  description: my network"
  echo "  type: text"
  echo "  value: IB"
  echo "- resource: localdisk"        # 这个资源以在cb.yaml中定义
  echo "  value: 30"
  echo "---"
  sleep 10
done
```

CHAPTER 6

软件许可

6.1 天云融创软件许可

使用 SKYFORM AIP 和相关支持服务，即表示您接受本协议并确认您已阅读并理解本协议。如果您代表一个组织行事，您就是被视为有权代表签署本协议该组织的。如果您不接受本条款同意，您不得使用天云融创软件或服务。

1.0 定义

“协议”指本软件订阅协议；

“天云融创软件”是指依照中华人民共和国法律组织的北京天云融创软件技术有限公司；

“文档”是指天云融创提供的所有描述软件使用的用户文档；

“软件”是指二进制的特定天云融创的计算机程序格式，由天云融创软件提供，包括任何文档；

“客户”是指使用天云融创软件提供的该软件的个人或组织。

2.0 使用授权

(a) 非生产用途。如果客户已订阅软件非生产目的，受本条款和条件的约束协议，天云融创软件授予客户个人的、有限的、在非生产环境，仅用于非生产目的软件之间的测试、评估和集成开发以及客户在订阅期内销售或许可的产品对应的时间段。在此类到期时订阅期限，天云融创软件可自行决定续订订阅期限为后续期限。

(b) 生产用途。如果客户已订阅软件生产使用，受本条款和条件的约束协议，天云融创软件授予客户个人的、有限的、仅为客户内部使用软件的独家许可商业目的，并且仅用于指定系统与服务服务器。本协议可以通过天云融创软件将根据客户的要求涵盖额外的软件服务支付相关费用。

3.0 限制

客户不得：(a) 出租、租赁、出借、转让、转让、分享或转售软件；(b) 未经事先天云融创软件的书面同意，将软件提供给第三方合作伙伴、除客户的内部业务目的外使用；(c) 基于本软件创作和分发衍生作品，或将本软件用于除按照本协议的规定；(d) 违反任何适用的法律、法规、行政命令的情况下出口或再出口。

4.0 许可和所有权

4.1 软件：每种类型的软件受制于授权或最终用户许可协议，包含许可条款或在本协议或适用订单的附录中的条款。

4.2 想法使用自由：除非客户对客户信息的权利尽管本协议中有任何其他的规定，想法、方法、概念、诀窍、结构、技术、发明、过程、发现、改进和其他信息和在任何订单过程中开发的材料可能被天云融创软件以某种方式使用，这样的使用被认为是适当的。

4.3 标记: 除非在其他相关协议中明确说明，在本协议中授予使用任何天云融创软件，天云融创软件关联公司、客户或第三方商号、服务标志或商标。

5.0 保修

(a) 天云融创软件保证：

(i) 对付费订阅软件的客户由合格人员以专业和熟练的方式进行技术支持；

(ii) 有权参与客户协议；

(iii) 按照天云融创软件已有认知，天云融创软件品牌的软件在交付给客户时不包含恶意或隐藏机制或代码的破坏软件。

(b) 除本协议另有规定外，本软件是按“原样”提供给客户，天云融创软件不承担所有其他责任、明示、默示、法定的保证、陈述和条件或以其他方式，包括但不限于任何保证或条件适销性、适销性质量或特定用途的适用性，或源自交易、使用或贸易实践的过程。特别是，天云融创软件不保证该软件将运行不间断或无错误、或软件中的缺陷一定可纠正的或在规定时间内被更正。

6.0 责任限制

在任何情况下任何一方的累积责任是否会超出该方的实际，直接的、可证明的损害赔偿金额，并且赔偿金额不超过客户支付对于软件订阅的金额。本责任限制也不得就有形个人财产的损害，人身伤害或死亡，或因故意不当行为引起的索赔，或重大过失提出索赔。

在任何情况下任何一方或天云融创软件对任何惩罚性、特殊的、后果性的、附带的或间接的损害或任何利润损失、业务或商誉损失，或数据丢失提供索赔。

7.0 用户信息

用户允许天云融创软件和天云融创软件的合作伙伴使用客户注册的电子邮箱向客户推送天云融创软件的最新动态和技术更新信息。客户可以选择退订信息的推送。

8.0 其他

(a) 未经天云融创软件事先书面同意，客户不得转让本协议或转让协议相关的权利。
(b) 本协议受中华人民共和国法律保护。

6.2 第三方软件许可

以下的软件许可适用于 Skyform AIP 里所用的开源组件：

- The 3-Clause BSD License
- The MIT License
- Apache License, Version 2
- GNU General Public License v2

CHAPTER 7

产品常见问题

7.1 什么是算力调度系统，它为什么重要

当一个计算系统中同时运行超过一个计算任务时，就需要对任务进行资源分配，防止任务间的计算资源冲突，这就需要有调度系统。单台服务器上的调度系统由操作系统提供，而多服务器组成的算力集群就需要算力调度系统或者说集群调度系统。若没有算力调度系统，多个计算任务就会互相争抢算力资源导致任务性能大幅度下降，系统平均利用率下降，浪费时间，浪费贵重的算力资源（如高端 GPU）。

算力调度的功能和性能对算力的利用率、任务利用算力的成本等多个因素都有影响。许多客户觉得既然有开源的算力调度系统，不愿意购买商业软件，实际上商业软件为算力中心每提高 5% 的利用率所产生的价值，就会高于购买商业调度软件本身的费用，同时由于算力利用率的提高，缩短了产品研发的周期，由此带来的价值更高。

7.2 SkyForm 算力调度系统有哪些特色，提供哪些价值

SkyForm 算力调度系统是支持多算力池、多类型应用、高性能、轻量化的信创产品，在大规模生产环境中稳定性和可靠性得到充分的验证。系统帮助用户提高算力利用率、减少运维成本，是应用可以充分利用底层的算力资源，帮助企业提高生产力、减少 IT 的成本。

SkyForm 算力调度系统支持各种类型对高性能算力有要求的应用，如工业仿真、基因和蛋白质研究、芯片设计、天气预报、大数据分析、人工智能训练和推理等。

算力资源以裸金属服务器、部分服务器、和无服务器方式提供。SkyForm 算力调度系统的 Python SDK 让 AI 应用开发中所需的算力通过 API 的方式获得。完整的资源使用监控和精细的秒级计量计费优化用户的开销。

SkyForm 算力调度系统以其高性能的调度能力和智能调度算法，比一般的开源的算力调度软件能够为算力池的平均利用率增加 5-10%，一个 1 千万成本的算力池意味着会节约 50-100 万的预算额度。

7.3 SkyForm 算力调度系统是怎样收费的

AIP 按照所管理的算力集群中的 CPU 颗数 +GPU 卡数收费。分为免费版和企业版（收费版），收费方式为：- 订阅：订阅按年和集群大小收取固定费用，订阅期间用户享受使用最新版本的 AIP，以支持最新的操作系统和最新的应用软件，并通过邮件、微信等方式的产品维保和产品技术支持。- 永久授权：永久授权为根据集群大小的一次性收费。购买永久授权后的第一年内享受产品维保和产品技术支持。第二年起可以按年选购产品维保和产品技术支持，年度费用为永久授权费用的 15%。以上费用不包含额外的服务，如客户现场安装、应用集成、系统集成、现场巡检等。如需此类服务，请联系天云融创提供专家服务和单独报价。

7.4 对于 AI 模型训练，调度系统提供哪些功能

调度系统对 AI 模型训练，尤其是大模型的训练做了优化，任务在出错后可以自动重启，在训练过程中提供各种资源监控以保证训练任务的顺利进行。对于大模型的训练，其独特的任务远程分发能力（支持多种子任务远程分发框架如 DeepSpeed、Ray 等）实现分布式训练任务的快速启动，缩短训练时间、减少资源浪费。

7.5 对于推理服务，调度系统提供哪些功能

调度系统为推理用户提供推理所需自动伸缩调度能力和访问端口转发等功能。推理可以使用容器或者直接使用裸金属服务。

7.6 对于 AI 应用开发，调度系统提供哪些功能

调度系统可以调度开发用的 Jupyter 任务，或者调度开可以接受 ssh 的应用开发容器以便用户 VSCode 的远程 ssh 连接。开发者可以使用 SDK 来使用资源进行开发训练、调试、和推理测试。

7.7 系统支持哪些类型的 CPU

系统目前支持 X86、ARM、申威、龙芯 CPU。其他的 CPU 类型请联系公司。

7.8 系统支持哪些类型的 GPU 或加速芯片

系统能够自动检测并调度的 GPU 或加速芯片有：英伟达、AMD、天数智芯、燧原等，其他的国产加速芯片的支持请联系公司。

7.9 SkyForm 算力调度系统可以在不用虚拟机的场景中支持 vGPU 吗

SkyForm 算力调度系统支持 NVIDIA 的 MIG GPU 分片。另外对于不支持 MIG 的 GPU，调度系统可以支持多个任务同时使用同一个 GPU，在这种场景中，多个任务在 GPU 内分时运行，这对于偶尔使用 GPU 的任务可以使 GPU 复用而节约开销。

7.10 系统怎样实现任务间的资源隔离

SkyForm 算力调度系统利用 Linux 的 cgroup（裸金属任务）或者容器实现任务间的资源隔离。

7.11 系统支持哪些容器类型

目前系统支持 Docker 和 Singularity。其他类型的容器技术可以通过类似的方法集成到调度系统中。

7.12 系统支持哪些操作系统

系统支持所有的商业和开源 Linux 操作系统，包括基于 Linux 的国产操作系统，并支持 X86 Windows。

7.13 系统支持哪些 MPI 框架

SkyForm 算力调度系统与 Intel MPI 有深度集成，同时支持 MPICH、OpenMPI、MVAPICH 等开源 MPI 框架。这些框架常用于工业仿真和开源科学计算软件中。

7.14 系统支持哪些 AI 框架和大语言模型

系统支持所有常用的 AI 框架，包括 TensorFlow、PyTorch、Ray、PaddlePaddle、DeepSpeed 等，现代大模型都是基于这些框架构建的，所以本系统可以轻松支持大语言模型。

7.15 如果算力节点出现故障，任务怎样恢复

如果算力节点出现故障，调度系统可以自动把在故障节点上运行的任务自动重新一高优先级排队，以便在其他好的节点上重新运行。

7.16 调度系统有节能的功能吗

在算力使用不饱满的时候，调度系统可以自动把空闲的节点通过 IPMI 设置成休眠或关机状态，并在有任务需要算力时自动恢复电源状态。

7.17 调度系统支持哪些接口

SkyForm 算力调度系统除了自身的命令行接口外，还支持部分常用的 SLURM、LSF、和 PBS 命令行接口。SkyForm 调度系统还有基于 Swagger 标准的 REST API 服务，以及为 AI 准备的 Python SDK。

CHAPTER 8

搜索

- search