Slurm

作业调度系统

张鼎

武汉大学计算机学院

2021 年 05 月 26 日

Outline

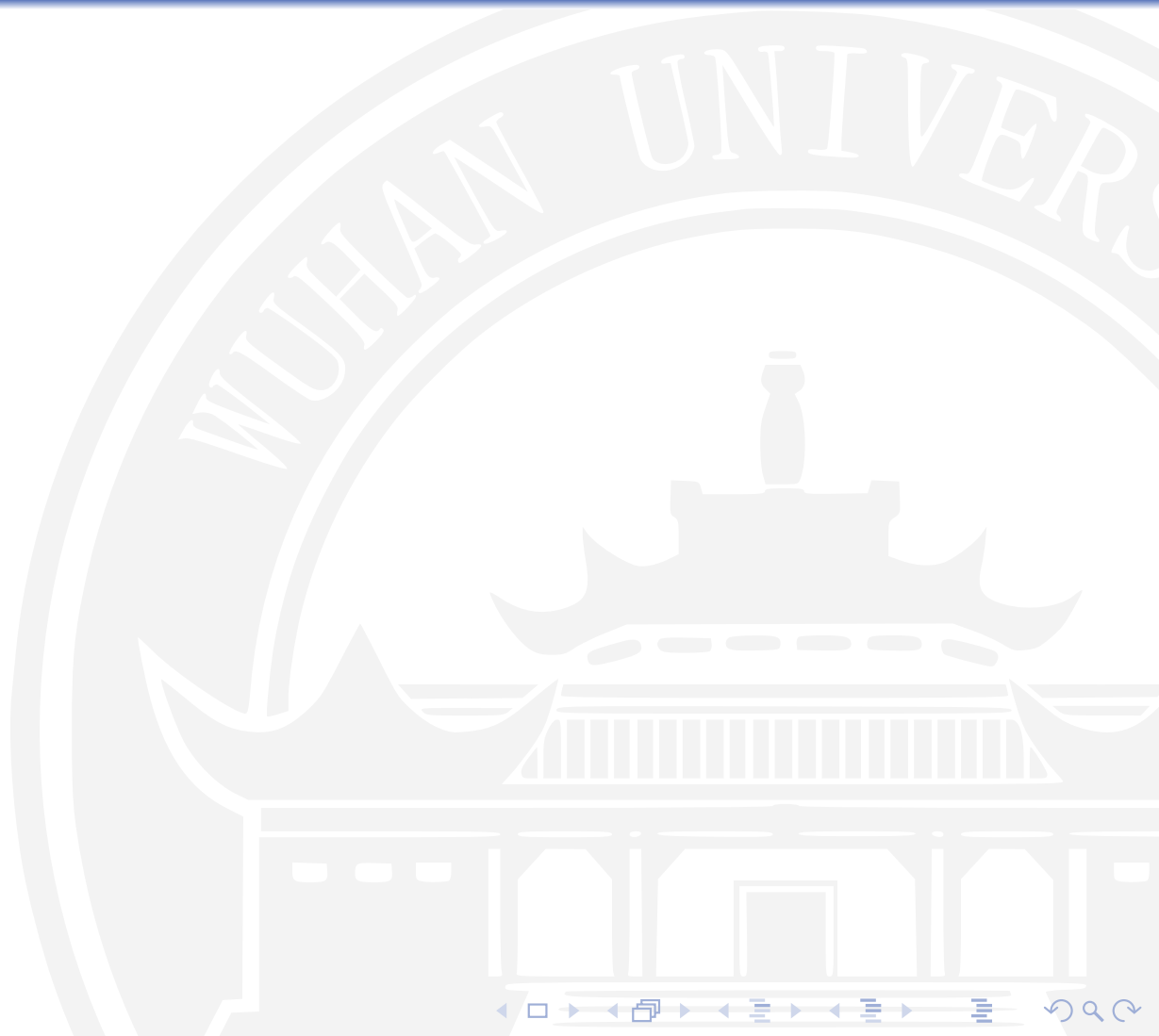
Outline

管理 & 使用服务器

- 硬件
- 软件
- 用户账号 & 密码
- 环境
- 运行程序

管理 & 使用服务器

- 1 台服务器
  - ★ 硬件
  - ★ 软件



管理 & 使用服务器

- 1 台服务器
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码

管理 & 使用服务器

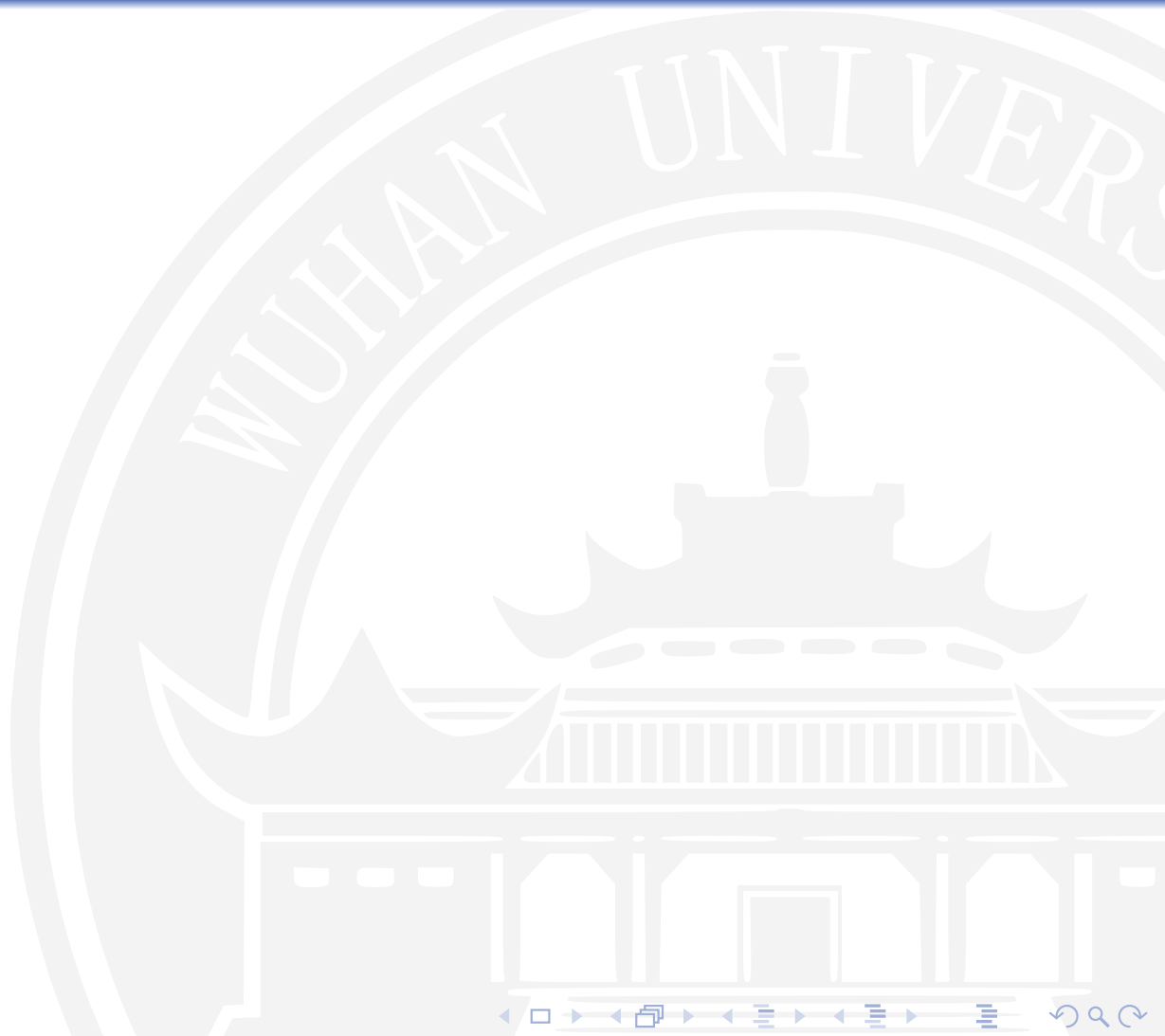
- 1 台服务器
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码
  - ★ 环境

管理 & 使用服务器

- 1 台服务器
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码
  - ★ 环境
  - ★ 运行程序

管理 & 使用服务器

- 1 台服务器
- 2 台服务器
  - ★ 硬件
  - ★ 软件



管理 & 使用服务器

- 1 台服务器
- 2 台服务器
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码

管理 & 使用服务器

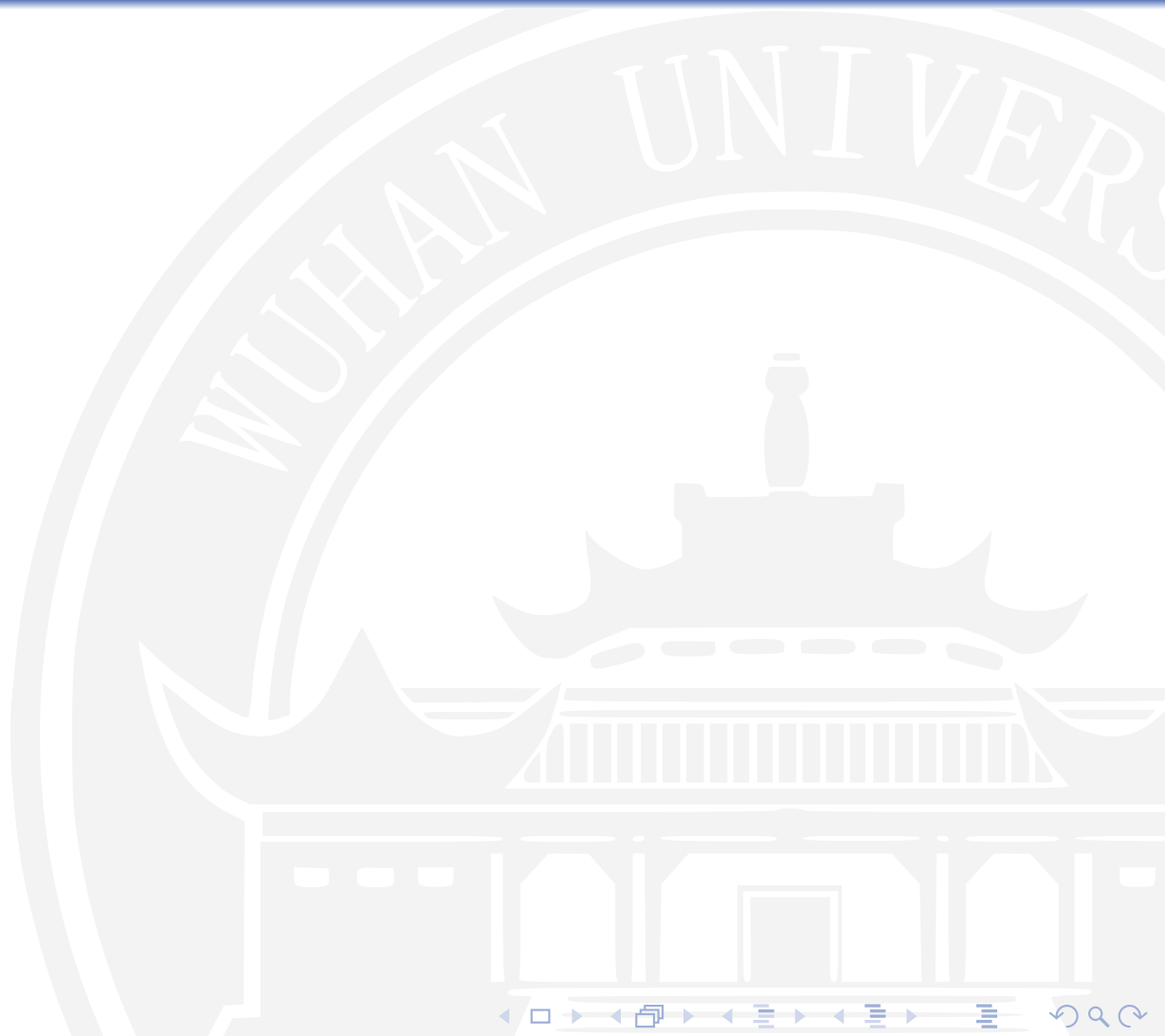
- 1 台服务器
- 2 台服务器
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码
  - ★ 环境

管理 & 使用服务器

- 1 台服务器
- 2 台服务器
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码
  - ★ 环境
  - ★ 运行程序

管理 & 使用服务器

- 1 台服务器
- 2 台服务器
- 4 台服务器
  - ★ 硬件
  - ★ 软件



管理 & 使用服务器

- 1 台服务器
- 2 台服务器
- 4 台服务器
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码

管理 & 使用服务器

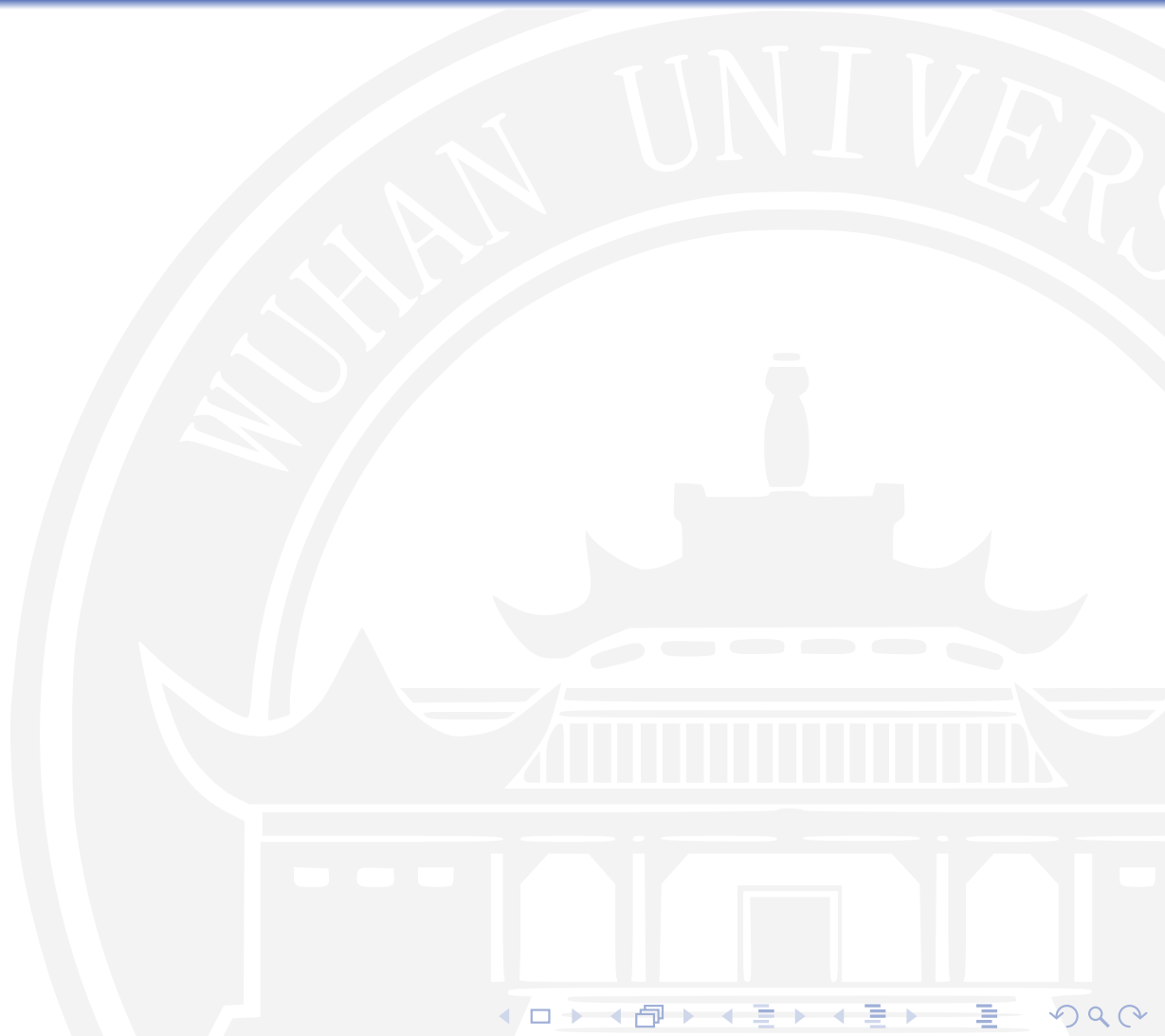
- 1 台服务器
- 2 台服务器
- 4 台服务器
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码
  - ★ 环境

管理 & 使用服务器

- 1 台服务器
- 2 台服务器
- 4 台服务器
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码
  - ★ 环境
  - ★ 运行程序

管理 & 使用服务器

- 1 台服务器
- 2 台服务器
- 4 台服务器
- n 台服务器……？
  - ★ 硬件
  - ★ 软件



管理 & 使用服务器

- 1 台服务器
- 2 台服务器
- 4 台服务器
- n 台服务器……？
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码

管理 & 使用服务器

- 1 台服务器
- 2 台服务器
- 4 台服务器
- n 台服务器..... ?
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码 – nis

管理 & 使用服务器

- 1 台服务器
- 2 台服务器
- 4 台服务器
- n 台服务器..... ?
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码 – nis
  - ★ 环境

管理 & 使用服务器

- 1 台服务器
- 2 台服务器
- 4 台服务器
- n 台服务器..... ?
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码 – nis
  - ★ 环境 – module

管理 & 使用服务器

- 1 台服务器
- 2 台服务器
- 4 台服务器
- n 台服务器..... ?
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码 – nis
  - ★ 环境 – module
  - ★ 运行程序

管理 & 使用服务器

- 1 台服务器
- 2 台服务器
- 4 台服务器
- n 台服务器..... ?
  - ★ 硬件
  - ★ 软件
  - ★ 用户账号 & 密码 – nis
  - ★ 环境 – module
  - ★ 运行程序 – job scheduler

Outline

Outline

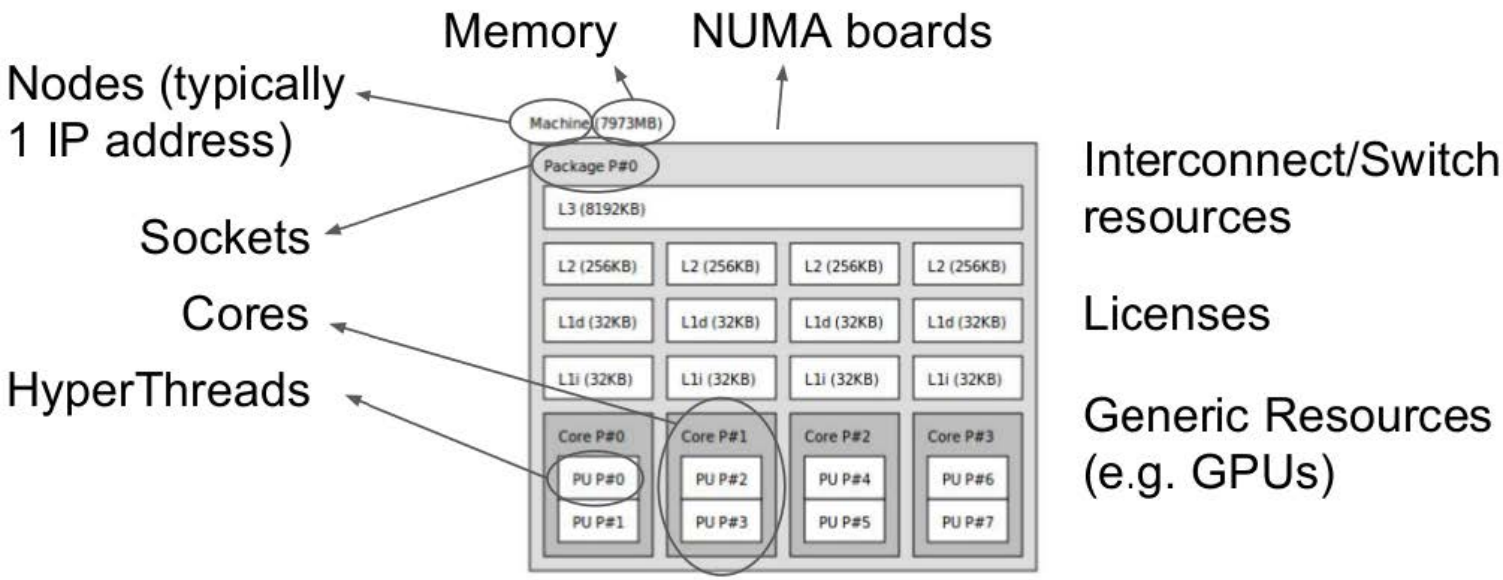
资源管理的任务

- 并行计算机执行并行作业的“胶水”

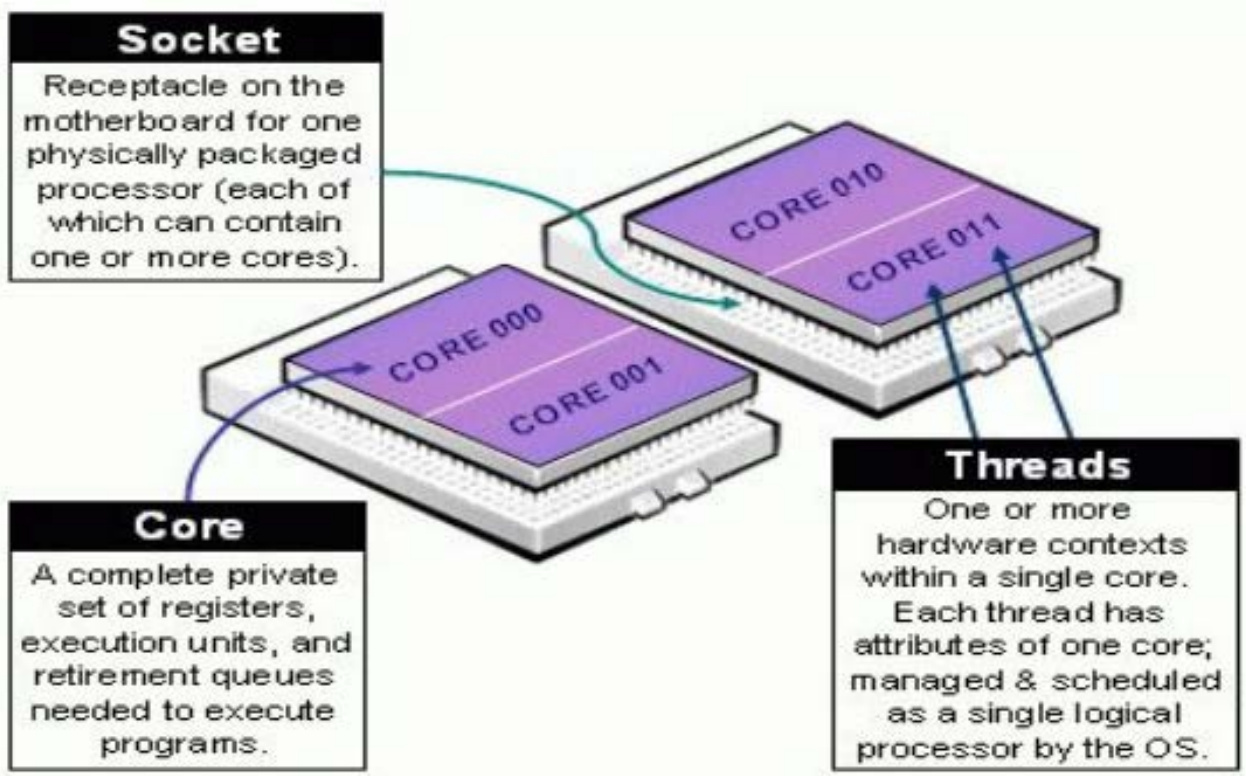
资源管理的任务

- 并行计算机执行并行作业的“胶水”
- 分配集群内的资源

资源管理的任务


Memory
NUMA boards
Nodes (typically 1 IP address)
Sockets
Cores
HyperThreads
Machine (7973MB)
Package P#0
L3 (8192KB)
L2 (256KB)
L1d (32KB)
L1i (32KB)
Core P#0
Core P#1
Core P#2
Core P#3
PU P#0
PU P#1
PU P#2
PU P#3
PU P#4
PU P#5
PU P#6
PU P#7
Interconnect/Switch resources
Licenses
Generic Resources (e.g. GPUs)

资源管理的任务


Socket
Receptacle on the motherboard for one physically packaged processor (each of which can contain one or more cores).
CORE 010
CORE 011
CORE 000
CORE 001
Core
A complete private set of registers, execution units, and retirement queues needed to execute programs.
Threads
One or more hardware contexts within a single core. Each thread has attributes of one core; managed & scheduled as a single logical processor by the OS.

资源管理的任务

- 并行计算机执行并行作业的“胶水”
- 分配集群内的资源
- 启动并以其他方式管理作业

资源管理的任务

- 并行计算机执行并行作业的“胶水”
- 分配集群内的资源
- 启动并以其他方式管理作业
- 需要知道系统硬件和软件的具体信息

Outline

作业调度的任务

- 管理作业队列

作业调度的任务

- 管理作业队列
- 支持复杂的调度算法

作业调度的任务

- 管理作业队列
- 支持复杂的调度算法
  - ★ 针对网络拓扑进行优化，公平调度，高级预订，抢占，帮派调度（时间片工作）等

作业调度的任务

- 管理作业队列
- 支持复杂的调度算法
  - ★ 针对网络拓扑进行优化，公平调度，高级预订，抢占，帮派调度（时间片工作）等
- 支持资源限制、服务质量 (QoS)

作业调度的任务

- 管理作业队列
- 支持复杂的调度算法
  - ★ 针对网络拓扑进行优化，公平调度，高级预订，抢占，帮派调度（时间片工作）等
- 支持资源限制、服务质量 (QoS)
  - ★ 队列

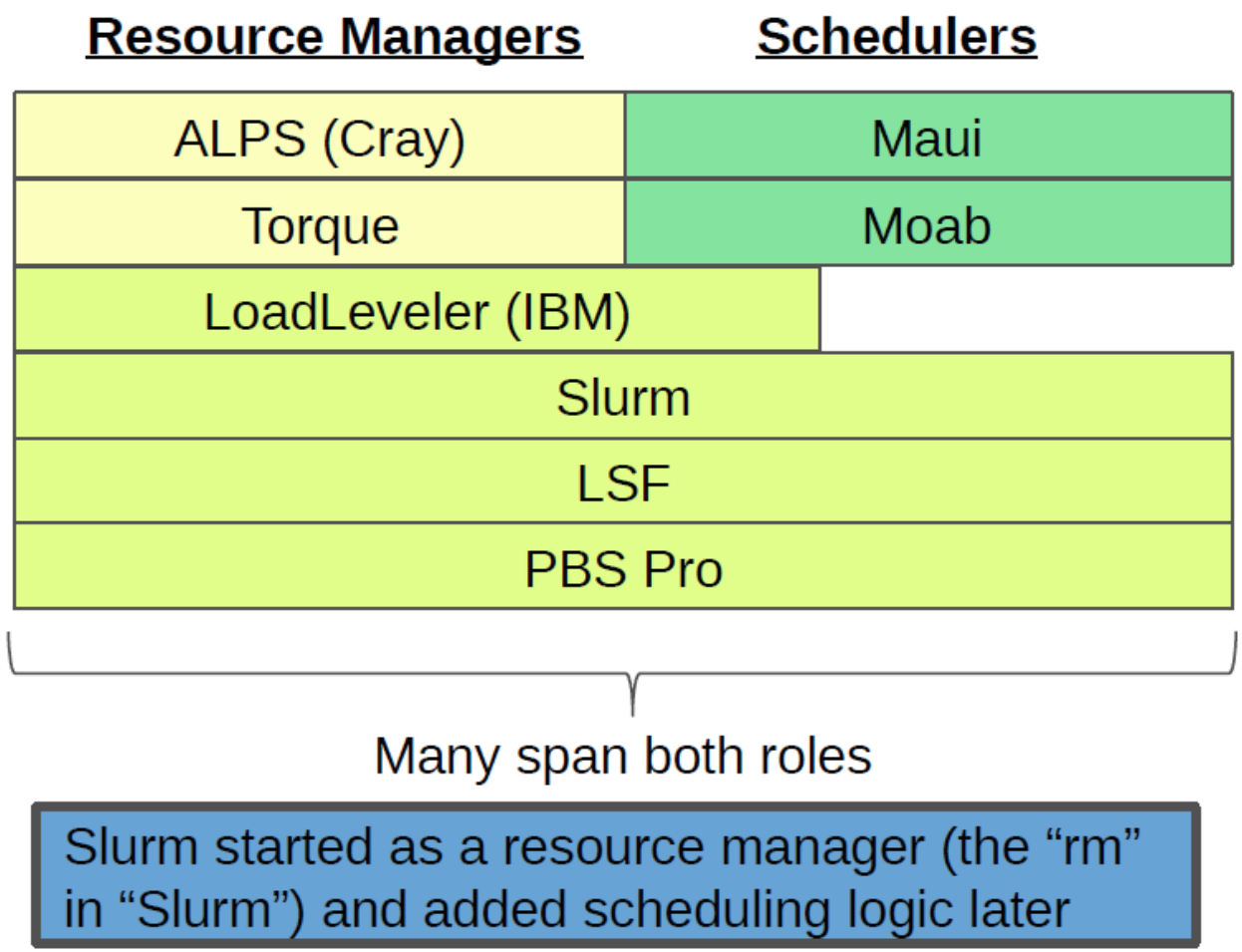
作业调度的任务

- 管理作业队列
- 支持复杂的调度算法
  - ★ 针对网络拓扑进行优化，公平调度，高级预订，抢占，帮派调度（时间片工作）等
- 支持资源限制、服务质量 (QoS)
  - ★ 队列
  - ★ 用户

作业调度的任务

- 管理作业队列
- 支持复杂的调度算法
  - ★ 针对网络拓扑进行优化，公平调度，高级预订，抢占，帮派调度（时间片工作）等
- 支持资源限制、服务质量 (QoS)
  - ★ 队列
  - ★ 用户
  - ★ 用户组
  - ★ 等等

资源管理器与作业调度器举例



Outline

Outline

Slurm 是什么

- Slurm
  - ★ ***S***imple ***L***inux ***U***tility for ***R***esource ***M***anagement

Slurm 是什么

- Slurm
  - ★ ***S***imple ***L***inux ***U***tility for ***R***esource ***M***anagement
- $>$ 550,000 行 C 代码

Slurm 是什么

- Slurm
  - ★ ***S***imple ***L***inux ***U***tility for ***R***esource ***M***anagement
- $>$ 550,000 行 C 代码
- 支持 Linux 和有限支持其他 Unix 发行版

Slurm 是什么

- Slurm
  - ★ ***S***imple ***L***inux ***U***tility for ***R***esource ***M***anagement
- > 550,000 行 C 代码
- 支持 Linux 和有限支持其他 Unix 发行版
- TOP 500 超级计算机中约 60% 使用 Slurm 管理作业负载。包括天河 1、2 号。(太湖之光使用 LSF 进行作业管理)

Slurm 功能

- 为用户分配资源（计算机节点）的独占和/或非独占访问权限，以便执行作业。

Slurm 功能

- 为用户分配资源（计算机节点）的独占和/或非独占访问权限，以便执行作业。
- 提供了一个框架，用于在一组分配的节点上启动，执行和监视工作（通常是并行作业，例如 MPI）。

Slurm 功能

- 为用户分配资源（计算机节点）的独占和/或非独占访问权限，以便执行作业。
- 提供了一个框架，用于在一组分配的节点上启动，执行和监视工作（通常是并行作业，例如 MPI）。
- 通过管理待处理作业队列来仲裁资源争用。

Slurm 功能

- 为用户分配资源（计算机节点）的独占和/或非独占访问权限，以便执行作业。
- 提供了一个框架，用于在一组分配的节点上启动，执行和监视工作（通常是并行作业，例如 MPI）。
- 通过管理待处理作业队列来仲裁资源争用。
- 使用基于希尔伯特曲线调度或胖树网络拓扑的最佳拟合算法来优化并行计算机上任务分配的局部性。

Outline

Slurm 设计目标

- 简单（取决于配置）
- 高度可扩展

Slurm 设计目标

- 简单（取决于配置）
- 高度可扩展
  - ★ Intel Single-Chip Cloud Computer ("Cluster on a chip")

Slurm 设计目标

- 简单（取决于配置）
- 高度可扩展
  - ★ Intel Single-Chip Cloud Computer ("Cluster on a chip")
  - ★ 管理天河 2 号上 310 万核心

Slurm 设计目标

- 简单（取决于配置）
- 高度可扩展
  - ★ Intel Single-Chip Cloud Computer ("Cluster on a chip")
  - ★ 管理天河 2 号上 310 万核心
  - ★ 可以管理超过 3300 万核心（2014 年，模拟验证）

Slurm 设计目标

- 简单（取决于配置）
- 高度可扩展
  - ★ Intel Single-Chip Cloud Computer ("Cluster on a chip")
  - ★ 管理天河 2 号上 310 万核心
  - ★ 可以管理超过 3300 万核心（2014 年，模拟验证）
- 开源 GPLv2。`https://github.com/SchedMD/slurm`

Slurm 设计目标

- 简单（取决于配置）
- 高度可扩展
  - ★ Intel Single-Chip Cloud Computer ("Cluster on a chip")
  - ★ 管理天河 2 号上 310 万核心
  - ★ 可以管理超过 3300 万核心（2014 年，模拟验证）
- 开源 GPLv2。`https://github.com/SchedMD/slurm`
- 对系统管理员友好

Slurm 设计目标

- 简单（取决于配置）
- 高度可扩展
  - ★ Intel Single-Chip Cloud Computer ("Cluster on a chip")
  - ★ 管理天河 2 号上 310 万核心
  - ★ 可以管理超过 3300 万核心（2014 年，模拟验证）
- 开源 GPLv2。`https://github.com/SchedMD/slurm`
- 对系统管理员友好
- 安全

Slurm 设计目标

- 简单（取决于配置）
- 高度可扩展
  - ★ Intel Single-Chip Cloud Computer ("Cluster on a chip")
  - ★ 管理天河 2 号上 310 万核心
  - ★ 可以管理超过 3300 万核心（2014 年，模拟验证）
- 开源 GPLv2。`https://github.com/SchedMD/slurm`
- 对系统管理员友好
- 安全
- 没有单点故障

Slurm 设计目标

- 简单（取决于配置）
- 高度可扩展
  - ★ Intel Single-Chip Cloud Computer ("Cluster on a chip")
  - ★ 管理天河 2 号上 310 万核心
  - ★ 可以管理超过 3300 万核心（2014 年，模拟验证）
- 开源 GPLv2。`https://github.com/SchedMD/slurm`
- 对系统管理员友好
- 安全
- 没有单点故障
- 高性能 (600 作业/s，1000 作业/s 提交)

Slurm 插件

- 提供具有通用插件机制的调度框架

Slurm 插件

- 提供具有通用插件机制的调度框架
- 根据配置文件和用户选项在运行时加载动态链接对象

Slurm 插件

- 提供具有通用插件机制的调度框架
- 根据配置文件和用户选项在运行时加载动态链接对象
- 110+ 个，30 种不同类别的插件
  - Network topology: 3D-torus, tree, etc
  - MPI: OpenMPI, PMI2, PMIx
  - Process tracking: cgroup, linuxproc, pgid, ipmi, etc
  - Accounting storage: MySQL, PostgreSQL, text file, etc

Slurm 插件

- 提供具有通用插件机制的调度框架

- 根据配置文件和用户选项在运行时加载动态链接对象
- 110+ 个，30 种不同类别的插件
  - Network topology: 3D-torus, tree, etc
  - MPI: OpenMPI, PMI2, PMIx
  - Process tracking: cgroup, linuxproc, pgid, ipmi, etc
  - Accounting storage: MySQL, PostgreSQL, text file, etc

Slurm Kernel (65% of code)				
Authentication Plugin	MPI Plugin	Job Submit Plugin	Topology Plugin	Accounting Storage Plugin
MUNGE	PMI2	Lua	Tree	MySQL

Outline

Slurm 概念

- Jobs：资源分配请求

Slurm 概念

- Jobs：资源分配请求
- Job steps：一组（通常是并行的）任务
  - ★ 通常是 MPI，UPC 和/或多线程应用程序
  - ★ 从作业分配到的资源中分配资源

Slurm 概念

- Jobs：资源分配请求
- Job steps：一组（通常是并行的）任务
  - ★ 通常是 MPI，UPC 和/或多线程应用程序
  - ★ 从作业分配到的资源中分配资源
  - ★ 一个作业可以包含多个作业步，这些步骤可以顺序或同时执行

Slurm 概念

- Jobs：资源分配请求
- Job steps：一组（通常是并行的）任务
  - ★ 通常是 MPI，UPC 和/或多线程应用程序
  - ★ 从作业分配到的资源中分配资源
  - ★ 一个作业可以包含多个作业步，这些步骤可以顺序或同时执行
  - ★ 有些作业可能会有上千个作业步

Slurm 概念

- Jobs：资源分配请求
- Job steps：一组（通常是并行的）任务
  - ★ 通常是 MPI，UPC 和/或多线程应用程序
  - ★ 从作业分配到的资源中分配资源
  - ★ 一个作业可以包含多个作业步，这些步骤可以顺序或同时执行
  - ★ 有些作业可能会有上千个作业步
  - ★ 比作业轻量

Slurm 概念

- Jobs：资源分配请求
- Job steps：一组（通常是并行的）任务
  - ★ 通常是 MPI，UPC 和/或多线程应用程序
  - ★ 从作业分配到的资源中分配资源
  - ★ 一个作业可以包含多个作业步，这些步骤可以顺序或同时执行
  - ★ 有些作业可能会有上千个作业步
  - ★ 比作业轻量
- Partitions：具有限制和访问控制的作业队列

Slurm 概念

- Jobs：资源分配请求
- Job steps：一组（通常是并行的）任务
  - ★ 通常是 MPI，UPC 和/或多线程应用程序
  - ★ 从作业分配到的资源中分配资源
  - ★ 一个作业可以包含多个作业步，这些步骤可以顺序或同时执行
  - ★ 有些作业可能会有上千个作业步
  - ★ 比作业轻量
- Partitions：具有限制和访问控制的作业队列
- QoS：限制、策略

Slurm 概念举例

- 用户提交作业 (Job) 到作业队列 (Partitions / queue)

Slurm 概念举例

- 用户提交作业 (Job) 到作业队列 (Partitions / queue)

Priority ordered queue of jobs

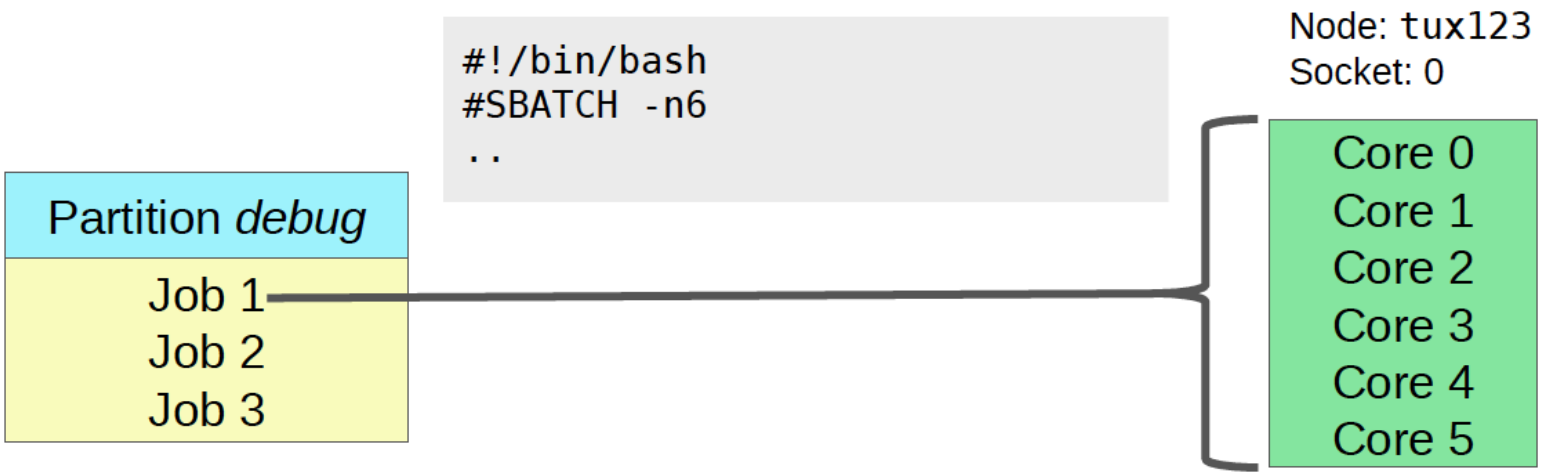
Partition *debug*
Job 1 Job 2 Job 3

Slurm 概念举例

- 分配资源给作业 (Resource allocation)

Slurm 概念举例

- 分配资源给作业 (Resource allocation)

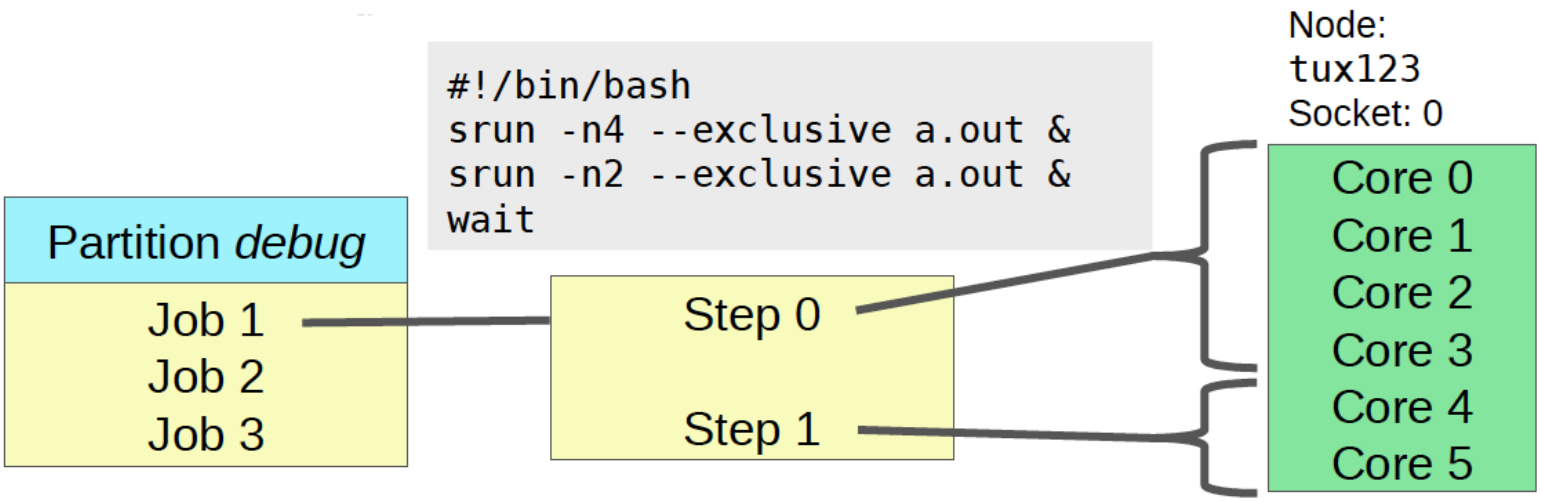


Slurm 概念举例

- 作业又可以分为更小的作业步 (Job step)
- 作业步从作业分配到的资源中获取资源

Slurm 概念举例

- 作业又可以分为更小的作业步 (Job step)
- 作业步从作业分配到的资源中获取资源



节点状态信息

- 主板、socket、核心、线程
- CPUs
- 内存大小
- 通用资源
- 特征（任意字符串。例如：系统版本、CPU 类型等）
- 状态（例如：UNKNOWN、ALLOCATED、DOWN、IDLE、DRAIN、COMPLETING、NO_RESPOND 等）
  - ★ 原因、时间、userID
    - ◇ e.g. "Bad PDU [operator@12:40:10T07/05/2021]"

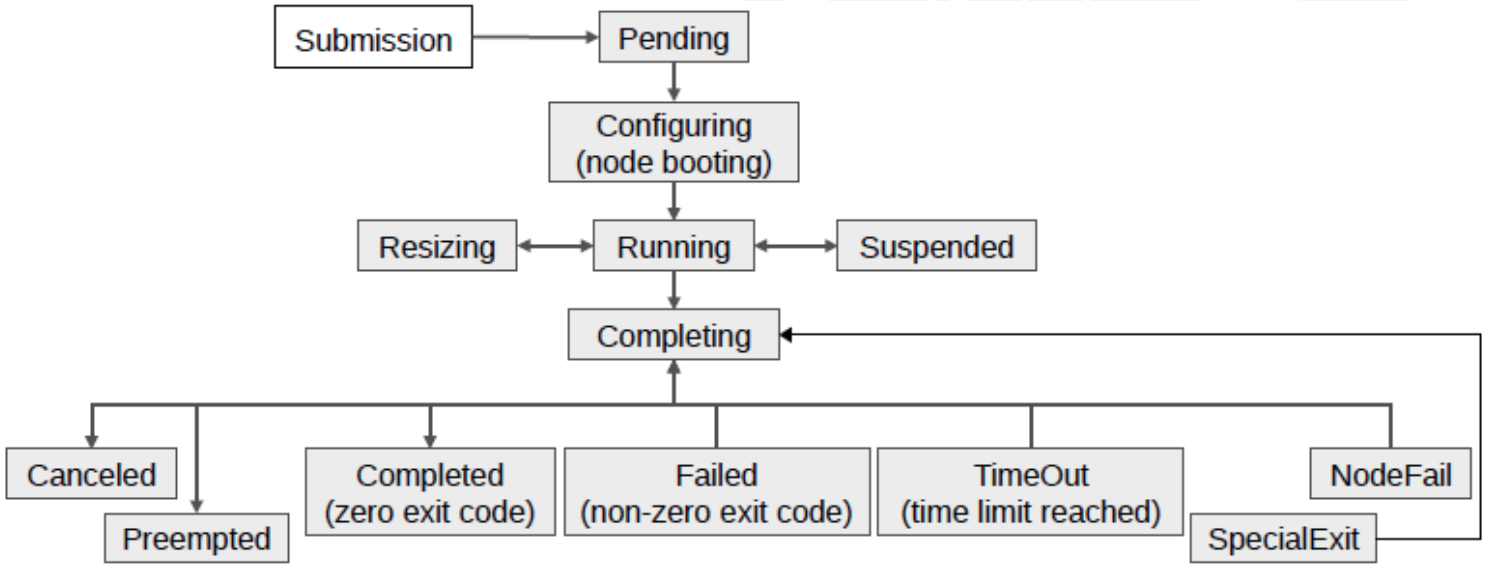
分区/队列状态信息

- 节点集合信息
  - ★ 节点可以在多个分区中
- 作业大小和时间限制（例如：某些分区的大小和时间限制较小，而其他分区的大小和时间限制较大）
- 访问控制列表（按帐户，QoS 或 Linux Group）
- 抢占规则
- 状态信息（例如：UP、DOWN、Drain 等）
- 超额预订和帮派调度规则

作业状态信息

- ID（数字）
- 名称
- 时间限制
- 大小信息（例如：节点、CPU、socket、核心、线程）
- 要包括或排除在分配中的特定节点名称
- 分配中需要的节点功能
- 作业依赖
- 账户名称
- QoS
- 状态（例如：PENDING、RUNNING、CANCELLED、COMPLETED、FAILED、TIMEOUT 等）

作业状态


Submission
Pending
Configuring
(node booting)
Resizing
Running
Suspended
Completing
Canceled
Preempted
Completed
(zero exit code)
Failed
(non-zero exit code)
TimeOut
(time limit reached)
NodeFail
SpecialExit

作业步状态信息

- ID（数字）：<job_id>.<step_id>
- 名称
- 时间限制
- 大小信息（例如：节点、CPU、socket、核心、线程）
- 要包括或排除在分配中的特定节点名称
- 分配中需要的节点功能

Slurm 概念 Summary

- 作业被提交到 Slurm 队列/分区
- 为作业分配资源（核心，内存等）
- 作业步使用作业的资源执行应用程序
- 作业和相关资源受到监视：
  - 通过插件：JobAcctGather，Cgroup，NHC 等

Outline

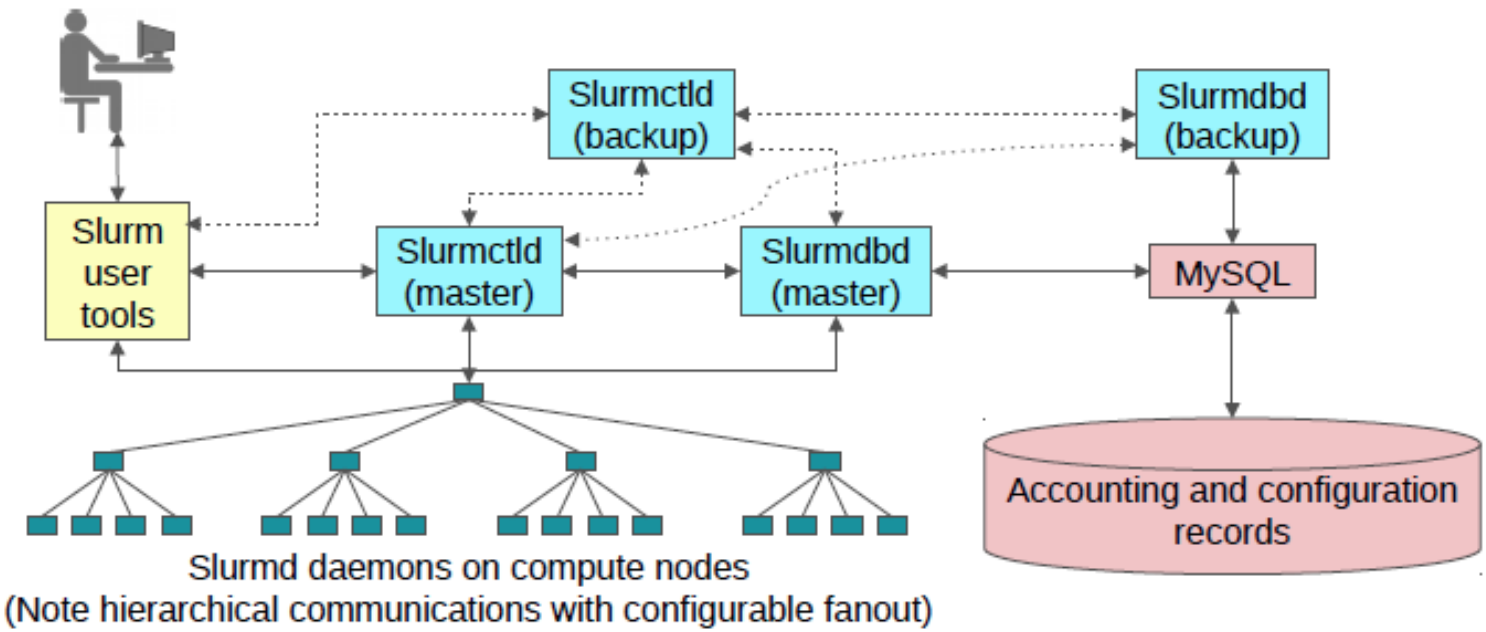
Daemons

- **slurmctld** - 中央控制程序（通常每一个集群一个）
  - 监控资源状态
  - 管理作业队列
  - 分配资源
- **slurmdbd** - 数据库守护程序（通常每一个集群一个）
  - 统计计费信息
  - 管理计费配置（限制、份额等）
    - 将计费配置推送到 slurmctld

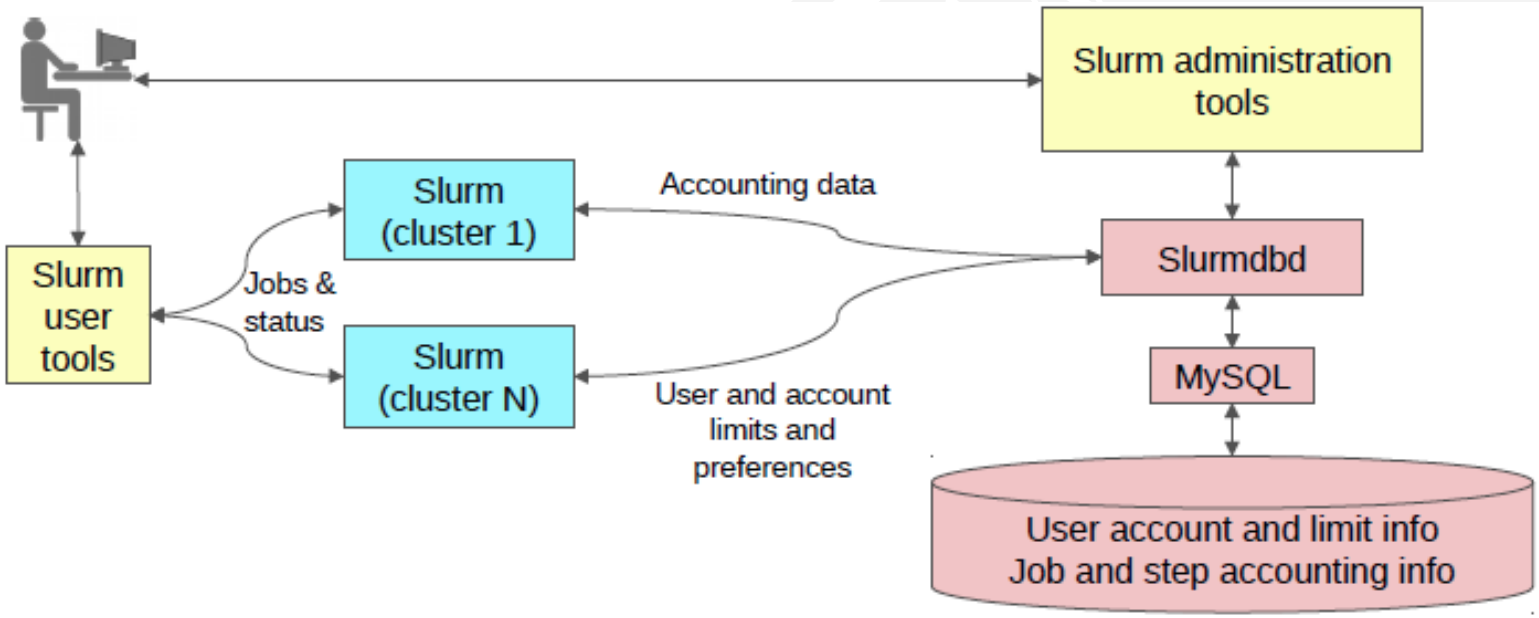
Daemons

- **slurmd** - 计算节点守护程序（通常每个计算节点一个）
  - 启动并管理 slurmstepd
  - 非常轻量
  - 启动后处于静默状态，除了一些计费统计管理
  - 支持分层通信、fanout
- **slurmstepd** - 作业步管理程序
  - 启动批处理作业和作业步
  - 启动用户应用程序
  - 管理计费信息，应用程序 IO，信号等

典型集群架构


Slurmctld
(backup)
Slurmdbd
(backup)
Slurm
user
tools
Slurmctld
(master)
Slurmdbd
(master)
MySQL
Accounting and configuration
records
Slurmd daemons on compute nodes
(Note hierarchical communications with configurable fanout)

典型企业集群架构


Slurm administration tools
Slurm (cluster 1)
Accounting data
Slurmdbd
Slurm user tools
Jobs & status
Slurm (cluster N)
MySQL
User and account limits and preferences
User account and limit info
Job and step accounting info

Outline

Outline

基本用户命令

- **sacct**：
  - 显示激活的或已完成作业或作业步的记账（对应需缴纳的机时费）信息。
- **salloc**：
  - 为需实时处理的作业分配资源，典型场景为分配资源并启动一个 shell，然后用此 shell 执行 srun 命令去执行并行任务。
- **sattach**：
  - 吸附到运行中的作业步的标准输入、输出及出错，通过吸附，使得有能力监控运行中的作业步的 IO 等。

基本用户命令

- **sbatch**：
  - 提交作业脚本使其运行。此脚本一般也可含有一个或多个 srun 命令启动并行任务。
- **sbcast**：
  - 将本地存储中的文件传递分配给作业的节点上，比如 /tmp 等本地目录；对于 /home 等共享目录，因各节点已经是同样文件，无需使用。
- **scancel**：
  - 取消排队或运行中的作业或作业步，还可用于发送任意信号到运行中的作业或作业步中的所有进程。

基本用户命令

- **scontrol**：
  - 显示或设定 Slurm 作业、队列、节点等状态。
- **sinfo**：
  - 显示队列或节点状态，具有非常多过滤、排序和格式化等选项。
- **squeue**：
  - 显示队列中的作业及作业步状态，含非常多过滤、排序和格式化等选项。
- **srun**：
  - 实时交互式运行并行作业，一般用于段时间测试，或者与 sallcoc 及 sbatch 结合。

Outline

三种模式

- 提交作业有三种模式：
  - 批处理模式（采用 sbatch 命令提交，最常用方式）
  - 交互式作业提交（采用 srun 命令提交）
  - 实施分配作业模式（采用 salloc 命令提交）

sbatch - 批处理模式 (最常用方式)

- 对于批处理作业（提交后立即返回该命令行终端，用户可进行其它操作）使用 sbatch 命令提交作业脚本，作业被调度运行后，在所分配的首个节点上执行作业脚本。
- 在作业脚本中也可使用 srun 命令加载作业任务。提交时采用的命令行终端终止，也不影响作业运行。

srun - 交互式作业提交

- 资源分配与任务加载两步均通过 srun 命令进行：当在登录 shell 中执行 srun 命令时，srun 首先向系统提交作业请求并等待资源分配，然后在所分配的节点上加载作业任务。
- 采用该模式，用户在该终端需等待任务结束才能继续其它操作，在作业结束前，如果提交时的命令行终端断开，则任务终止。一般用于短时间小作业测试。

salloc - 实时分配作业模式

- 实时分配作业模式类似于交互式作业模式和批处理作业模式的融合。
- 用户需指定所需要的资源条件，向资源管理器提出作业的资源分配请求。提交后，作业处于排队，当用户请求资源被满足时，将在用户提交作业的节点上执行用户所指定的命令，指定的命令执行结束后，运行结束，用户申请的资源被释放。在作业结束前，如果提交时的命令行终端断开，则任务终止。
- 典型用途是分配资源并启动一个 shell，然后在这个 shell 中利用 srun 运行并行作业。

三种模式

- salloc 后面如果没有跟定相应的脚本或可执行文件，则默认选择/bin/sh，用户获得了一个合适环境变量的 shell 环境。
- salloc 和 sbatch 最主要的区别是 salloc 命令资源请求被满足时，直接在提交作业的节点执行相应任务，而 sbatch 则当资源请求被满足时，在分配的第一个节点上执行相应任务。
- salloc 在分配资源后，再执行相应的任务，很适合需要指定运行节点和其它资源限制，并有特定命令的作业。

Outline

sinfo - 显示队列、节点信息

- sinfo 可以查看系统存在什么队列、节点及其状态。如 sinfo -l

```
zhangding@swarm01:~ %  sinfo -l
Sun May 23 20:11:37 2021
PARTITION AVAIL  TIMELIMIT   JOB_SIZE ROOT OVERSUBS     GROUPS  NODES       STATE NODELIST
hpxg*        up 1-00:00:00 1-infinite   no       NO        all     10       mixed n[0331,0334-0336,0347,0350,0353,0358-0359,0361]
hpxg*        up 1-00:00:00 1-infinite   no       NO        all     12   allocated n[0332-0333,0341-0342,0348-0349,0351-0352,0354-0355,0357,
0375]
hpxg*        up 1-00:00:00 1-infinite   no       NO        all     23        idle n[0337-0340,0343-0346,0356,0360,0362-0374]
hpib         up   infinite 1-infinite   no       NO        all      1    draining n0023
hpib         up   infinite 1-infinite   no       NO        all     26       mixed n[0007-0009,0014,0031,0038,0046,0056-0057,0068,0073,0078,
0107,0120-0121,0172-0174,0182-0183,0260-0261,0307,0319-0321]
hpib         up   infinite 1-infinite   no       NO        all    117   allocated n[0001-0006,0010-0013,0015-0022,0024-0030,0034,0039,0047-
0055,0058-0061,0069-0072,0074-0077,0079,0095-0098,0102-0106,0108-0119,0122-0123,0158-0169,0175-0181,0185-0188,0190,0253-0257,0259,0301-0305
,0308-0312,0315,0317-0318,0328,0330]
hpib         up   infinite 1-infinite   no       NO        all    124        idle n[0032-0033,0035-0037,0040-0045,0062-0067,0080-0094,0099-
0101,0124-0157,0170-0171,0184,0189,0258,0262-0300,0306,0313-0314,0316,0322-0327,0329]
pub          up   infinite 1-infinite   no       NO        all      8       mixed n[0376,0378,0383,0385,0391,0397,0403-0404]
pub          up   infinite 1-infinite   no       NO        all     12   allocated n[0396,0398-0402,0405-0410]
pub          up   infinite 1-infinite   no       NO        all     15        idle n[0377,0379-0382,0384,0386-0390,0392-0395]
sd530        up   infinite 1-infinite   no       NO        all      1    drained* n0416
sd530        up   infinite 1-infinite   no       NO        all      6       mixed n[0417-0422]
sd530        up   infinite 1-infinite   no       NO        all     21        idle n[0423-0440,0443-0445]
fat          up   infinite 1-infinite   no       NO        all      2       mixed m[001-002]
gpu          up   infinite 1-infinite   no       NO        all      1    drained* g0025
gpu          up   infinite 1-infinite   no       NO        all     42       mixed g[0001-0020,0022-0024,0027,0029-0034,0036-0040,0042-0043,
0045-0046,0090,0102-0103]
gpu          up   infinite 1-infinite   no       NO        all     17   allocated g[0021,0026,0035,0041,0081-0086,0105-0111]
gpu          up   infinite 1-infinite   no       NO        all      6        idle g[0028,0087-0089,0101,0104]
```

sinfo - 显示队列、节点信息

- 主要输出项：
  - PARTITION：队列名，后面带有 * 的，表示此队列为默认队列。
  - AVAIL：up 表示可用，down 表示不可用。
  - TIMELIMIT：作业运行墙上时间限制，infinite 表示没限制。
  - JOB_SIZE：可供用户作业使用的最小和最大节点数，如果只有 1 个值，则表示最大和最小一样，infinite 表示无限制。
  - ROOT：是否限制资源只能分配给 root 账户。
  - GROUPS：可使用的用户组，all 表示所有组都可以用。
  - NODES：节点数。
  - NODELIST：节点名列表。

sinfo - 显示队列、节点信息

- 主要输出项：cont.
  - OVERSUBSCRIBE：是否允许作业分配的资源超过计算资源（如 CPU 数）：
    - no：不允许超额。
    - exclusive：排他的，只能给这些作业用（等价于 srun exclusive）
    - force：资源总被超额。
    - yes：资源可以被超额。

sinfo - 显示队列、节点信息

- 主要输出项：cont.
  - STATE：节点状态，可能的状态包括：
    - allocated、alloc：已分配。
    - completing、comp：完成中。
    - down：宕机。
    - drained、drain：已失去活力。
    - draining、drng：失去活力中。
    - fail：失效。
    - failing、failg：失效中。
    - future、futr：将来可用。

sinfo - 显示队列、节点信息

- 主要输出项：cont.
  - STATE：节点状态，可能的状态包括：cont.
    - idle：空闲，可以接收新作业。
    - maint：保持。
    - mixed：混合，节点在运行作业，但有些空闲 CPU 核，可接受新作业。
    - perfctrs、npc：因网络性能计数器使用中导致无法使用。
    - power_down、pow_dn：已关机。
    - power_up、pow_up：正在开机中。
    - reserved、resv：预留。
    - unknown、unk：未知原因。
  - 注意，如果状态带有后缀 *，表示节点没响应。

sinfo - 显示队列、节点信息

- 主要参数：
  - **--help**
  - **-a**, **--all**：显示全部队列信息，如显示隐藏队列或本组没有使用权的队列。
  - **-i<seconds>**, **--iterate=<seconds>**：以 <seconds> 秒间隔持续自动更新显示信息。
  - **-l**, **--long**：显示详细信息。
  - **-s**：显示摘要信息。
  - **-p<partition>**, **--partition=<partition>**：显示 <partition> 队列信息。
  - **-R**, **--list-reasons**：显示不响应（down、drained、fail 或 failing 状态）节点的原因。

squeue - 查看队列中的作业信息

- 显示队列中的作业信息。如 squeue 显示：

```
~
zhangding@swarm01:~ %   squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          15453838      hpxg a.sbatch wukaixin PD       0:00      1 (PartitionTimeLimit)
          15473245      hpxg work01.s    hjluo PD       0:00      1 (PartitionTimeLimit)
          15473277       gpu test.sba  ouwenwu PD       0:00      1 (QOSMinGRES)
          15480799       gpu     bash  dubo_01 PD       0:00      1 (AssocGrpGRES)
          15491656      hpxg  test.sh     cyji PD       0:00      1 (QOSMaxCpuPerUserLimit)
          15491657      hpxg  test.sh     cyji PD       0:00      1 (QOSMaxCpuPerUserLimit)
          15503754      hpib active_8  dft_pg1 PD       0:00      3 (AssocGrpCpuLimit)
          15503928       gpu       mv    liuye PD       0:00      1 (AssocGrpGRES)
          15503969      hpxg  b040.sh zhanglia PD       0:00      1 (QOSMaxCpuPerUserLimit)
          15504060       gpu   sub.sh   macong PD       0:00      1 (Resources)
          15504137      hpxg  3-0-NEB     yqwu PD       0:00      2 (QOSMaxCpuPerUserLimit)
          15504174      hpxg l.sbatch    jsgao PD       0:00      1 (QOSMaxCpuPerUserLimit)
          15504181      hpib active_9  dft_pg1 PD       0:00      3 (AssocGrpCpuLimit)
          15504188       gpu tyzdemo2 zhongxia PD       0:00      1 (AssocGrpGRES)
          15504197      hpxg sh_run_p    lijie PD       0:00      1 (QOSMaxCpuPerUserLimit)
          15504198      hpxg sh_run_p    lijie PD       0:00      1 (QOSMaxCpuPerUserLimit)
          15504199      hpxg sbatch_f    lijie PD       0:00      1 (QOSMaxCpuPerUserLimit)
          14995122      hpib  Pro-1.5   magang  R 104-11:03:48      1 n0330
          15026396      hpib  Pro-3.0   magang  R 97-22:39:49       1 n0021
          15305275      hpib fluentZZ   guozhe  R 51-09:09:43       1 n0015
          15326169      hpib myjob.sh  yuyuewu  R 45-06:16:37       1 n0079
          15328740      hpib myjob.sh  yuyuewu  R 44-10:17:07       1 n0001
          15500898       gpu submit_f huxiaoli  R 1-18:20:27       4 g[0026,0035,0082,0105]
          15501884       gpu   rmnori zhongxia  R 1-07:26:50       1 g0090
          15501940       gpu myjob.sb   xiaoli  R 1-06:48:43       1 g0007
          15502024       gpu myjob.sh    mahui  R 1-05:59:35       1 g0019
          15502120       gpu runCM_No hyzhang2  R 1-05:27:00       1 g0002
```

squeue - 查看队列中的作业信息

- 主要输出项：
  - JOBID：作业号。
  - PARTITION：队列名（分区名）。
  - NAME：作业名。
  - USER：用户名。
  - TIME：已运行时间。

squeue - 查看队列中的作业信息

- 主要输出项：cont.
  - ST：状态。
    - PD：排队中，PENDING。
    - R：运行中，RUNNING。
    - CA：已取消，CANCELLED。
    - CF：配置中，CONFIGURING。
    - CG：完成中，COMPLETING
    - CD：已完成，COMPLETED。
    - F：已失败，FAILED。
    - TO：超时，TIMEOUT。
    - NF：节点失效，NODE FAILURE。
    - SE：特殊退出状态，SPECIAL EXIT STATE。

squeue - 查看队列中的作业信息

- 主要参数：
  - **--help**
  - **-a, --all**：显示所有队列中的作业及作业步信息，也显示被配置为对用户组隐藏队列的信息。
  - **-i<seconds>, --iterate=<seconds>**：以 <seconds> 秒间隔持续自动更新显示信息。
  - **-l, --long**：显示更多作业信息。
  - **-p<part_list>, --partition=<part_list>**：显示特定队列 <part_list> 信息，<part_list> 以，分隔。
  - **--start**：显示排队中的作业的预期执行时间。
  - **-u<user_list>, --user=<user_list>**：显示特定用户 <user_list> 的作业信息，<user_list> 以，分隔。
  - **-A<account_list>, --account=<account_list>**：显示用户 <account_list> 的作业信息，<account_list> 以，分隔。

scontrol show partition - 查看详细队列信息

- scontrol show partition 显示全部队列信息
- scontrol show partition PartitionName 或 scontrol show partition=PartitionName 显示队列名 PartitionName 的队列信息
- 输出类似:

```
~
zhangding@swarm01:~ %  scontrol show partition gpu
PartitionName=gpu
   AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
   AllocNodes=ALL Default=NO QoS=gpu
   DefaultTime=NONE DisableRootJobs=NO ExclusiveUser=NO GraceTime=0 Hidden=NO
   MaxNodes=UNLIMITED MaxTime=UNLIMITED MinNodes=0 LLN=NO MaxCPUsPerNode=UNLIMITED
   Nodes=g[0001-0043,0045-0046,0081-0090,0101-0111]
   PriorityJobFactor=1 PriorityTier=1 RootOnly=NO ReqResv=NO OverSubscribe=NO
   OverTimeLimit=NONE PreemptMode=OFF
   State=UP TotalCPUs=1320 TotalNodes=66 SelectTypeParameters=NONE
   JobDefaults=(null)
   DefMemPerNode=UNLIMITED MaxMemPerNode=UNLIMITED
```

scontrol show partition - 查看详细队列信息

- 主要输出项：
  - PartitionName：队列名。
  - MaxNodes：最大节点数。
  - MaxTime：最大运行时间。
  - MinNodes：最小节点数。
  - Nodes：节点名。
  - PriorityJobFactor：作业因子优先级。
  - PriorityTier：调度优先级。
  - OverSubscribe：是否允许超用。
  - PreemptMode：是否为抢占模式。
  - TotalCPUs：总 CPU 核数。
  - TotalNodes：总节点数。
  - DefMemPerNode：每个节点默认分配的内存大小，单位 MB。
  - MaxMemPerNode：每个节点最大内存大小，单位 MB。

scontrol show node - 查看详细节点信息

- scontrol show node 显示全部节点信息
- scontrol show node NODENAME 或 scontrol show node=NODENAME 显示节点名 NODENAME 的节点信息
- 输出类似:

```
~
zhangding@swarm01:~ %  scontrol show node g0111
NodeName=g0111 Arch=x86_64 CoresPerSocket=10
   CPUAlloc=20 CPUTot=20 CPULoad=9.21
   AvailableFeatures=(null)
   ActiveFeatures=(null)
   Gres=gpu:tesla:4
   NodeAddr=g0111 NodeHostName=g0111 Version=18.08
   OS=Linux 3.10.0-862.14.4.el7.x86_64 #1 SMP Wed Sep 26 15:12:11 UTC 2018
   RealMemory=120000 AllocMem=0 FreeMem=121323 Sockets=2 Boards=1
   State=ALLOCATED ThreadsPerCore=1 TmpDisk=0 Weight=1 Owner=N/A MCS_label=N/A
   Partitions=gpu
   BootTime=2021-02-27T10:02:56 SlurmdStartTime=2021-05-18T05:26:55
   CfgTRES=cpu=20,mem=120000M,billing=20,gres/gpu=4,gres/gpu:tesla=4
   AllocTRES=cpu=20,gres/gpu=4,gres/gpu:tesla=4
   CapWatts=n/a
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

scontrol show node - 查看详细节点信息

- 主要输出项：
  - NodeName：节点名。
  - Arch：系统架构。
  - CPUAlloc：分配给的 CPU 核数。
  - CPUTot：总 CPU 核数。
  - CPULoad：CPU 负载。
  - Gres：通用资源。如上面 Gres=gpu:tesla:4 指明了有四块 Tesla V100 GPU。
  - OS：操作系统。
  - RealMemory：实际物理内存，单位 GB。
  - AllocMem：已分配内存，单位 GB。
  - FreeMem：可用内存，单位 GB。
  - ThreadsPerCore：每颗 CPU 核线程数。
  - Weight：权重。
  - BootTime：开机时间。

scontrol show job - 查看详细作业信息

- scontrol show job 显示全部作业信息
- scontrol show job JOBID 或 scontrol show job=JOBID 显示作业号为 JOBID 的作业信息
- 输出类似下面：

```
~
zhangding@swarm01:~ %  scontrol show job 15500898
JobId=15500898 JobName=submit_faster_r101_2x_rfs.sh
   UserId=huxiaolin(52670) GroupId=huxiaolin(52670) MCS_label=N/A
   Priority=4293953095 Nice=0 Account=para QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=1-19:25:32 TimeLimit=UNLIMITED TimeMin=N/A
   SubmitTime=2021-05-22T02:03:42 EligibleTime=2021-05-22T02:03:42
   AccrueTime=2021-05-22T02:03:42
   StartTime=2021-05-22T02:56:23 EndTime=Unknown Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2021-05-22T02:56:23
   Partition=gpu AllocNode:Sid=paratera01:12964
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=g[0026,0035,0082,0105]
   BatchHost=g0026
   NumNodes=4 NumCPUs=80 NumTasks=80 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=80,node=4,billing=80,gres/gpu=16
   Socks/Node=* NtasksPerN:B:S:C=20:0:*:* CoreSpec=*
   MinCPUsNode=20 MinMemoryNode=0 MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/dat01/huxiaolin/tianchi/mmdetection_tianchi/submit_faster_r101_2x_rfs.sh
   WorkDir=/dat01/huxiaolin/tianchi/mmdetection_tianchi
   StdErr=/dat01/huxiaolin/tianchi/mmdetection_tianchi/slurm-15500898.out
   StdIn=/dev/null
   StdOut=/dat01/huxiaolin/tianchi/mmdetection_tianchi/slurm-15500898.out
   Power=
   TresPerNode=gpu:4
```

scontrol show job - 查看详细作业信息

- 主要输出项：
  - JobId：作业号。
  - JobName：作业名。
  - UserId：用户名（用户 ID）。
  - GroupId：用户组（组 ID）
  - Priority：优先级，越大越优先。如果为 0 则表示被管理员挂起，不允许运行。
  - Account：记账用户名。
  - QOS：作业的服务质量。

scontrol show job - 查看详细作业信息

- 主要输出项：cont.
  - Reason：原因。
  - Dependency：依赖关系。
  - RunTime：已运行时间。
  - TimeLimit：作业允许的剩余运行时间。
  - SubmitTime：提交时间。
  - EligibleTime：获得认可时间。
  - StartTime：开始运行时间。
  - SuspendTime：挂起时间。
  - NodeList：实际运行节点列表。
  - BatchHost：批处理节点名。
  - NumNodes：节点数。
  - NumCPUs：CPU 核数。
  - NumTasks：任务数。
  - TRES：显示分配给作业的可被追踪的资源。

scontrol show job - 查看详细作业信息

- 主要输出项：cont.
  - JobState：作业状态。
    - PENDING：排队中。
    - RUNNING：运行中。
    - CANCELLED：已取消。
    - CONFIGURING：配置中。
    - COMPLETING：完成中。
    - COMPLETED：已完成。
    - FAILED：已失败。
    - TIMEOUT：超时。
    - NODE FAILURE：节点失效。
    - SPECIAL EXIT STATE：特殊退出状态。

srun - 交互式提交并行作业

- srun 可以交互式提交运行并行作业。提交后，作业等待运行。等运行完毕后，才返回终端。语法为：

```
1 srun [OPTIONS...] executable [args...]
```

srun - 交互式提交并行作业

- 主要参数：
  - **--help**
  - **-A, --account=<account>**：指定此作业的责任资源为账户<account>，即账单（与计算费对应）记哪个名下，只有账户属于多个账单组才有权指定。
  - **-N, --nodes=<minnodes[maxnodes]>**：采用特定节点数运行作业，如没指定 maxnodes 则需特定节点数。注意，这里是节点数，不是 CPU 核数，实际分配的是节点数 * 每节点 CPU 核数。
  - **-n, --ntasks=<number>**：设定所需要的任务总数。默认是每个节点 1 个任务，注意是节点，不是 CPU 核。仅对作业起作用，不对作业步起作用。--cpus -per -task 选项可以改变此默认选项。
  - **--bell**：分配资源时终端响铃。
  - **--comment=<string>**：作业说明。

srun - 交互式提交并行作业

- 主要参数：cont.
  - **--contiguous** 需分配到连续节点，一般来说连续节点之间网络会快一点，如在同一个 IB 交换机内，但有可能导致开始运行时间推迟（需等待足够多的连续节点）。
  - **-D, --chdir=<path>**：在切换到 <path> 工作目录后执行命令。
  - **--epilog=<executable>**：作业结束后执行 <executable> 程序做相应处理。
  - **--prolog=<executable>**：作业开始运行前执行 <executable> 程序，做相应处理。
  - **--gres=<list>**：设定通用消费资源，可以以，分隔，每个 <list> 格式为 “name[[:type]:count]”。通常为 GPU 资源，如 --gres=gpu:4
  - **-I, --immediate[=<seconds>]**：在 <seconds> 秒内资源未满足的话立即退出。格式可以为 “I60”，但不能之间有空格如 “I 60”。

sbatch - 批处理方式提交作业

- Slurm 支持利用 sbatch 命令采用批处理方式运行作业。
- sbatch 命令在脚本正确传递给作业调度系统后立即退出，同时获取到一个作业号。作业等所需资源满足后开始运行。
- sbatch 提交一个批处理作业脚本到 Slurm。批处理脚本名可以在命令行上通过传递给 sbatch，如没有指定文件名，则 sbatch 从标准输入中获取脚本内容。

sbatch - 批处理方式提交作业

- 脚本文件基本格式：
  - 第一行以 #!/bin/sh 等指定该脚本的解释程序，/bin/sh 可以变为 /bin/bash、/bin/csh 等。
  - 在可执行命令之前的每行“#SBATCH”前缀后跟的参数作为作业调度系统参数。在任何非注释及空白之后的“#SBATCH”将不再作为 Slurm 参数处理。
  - 默认，标准输出和标准出错都定向到同一个文件 slurm%j.out，“%j”将被作业号代替。

sbatch - 批处理方式提交作业

- sbatch 脚本文件示例如下：

```
1  #!/bin/bash
2  #SBATCH --job-name=jobname                  # 指定作业的名字
3  #SBATCH --partition=gpu                     # 申请分区 'gpu' 的计算资源
4  #SBATCH --nodes=1                           # 申请 1 个节点
5  #SBATCH --ntasks-per-node=1                 # 申请每个节点上分配一个任务(进程)
6  #SBATCH --time=06:00:00                     # 计划最多运行 6 小时
7  #SBATCH --output=slurm-%A_%a.out            # 标准输出定向到文件的文件名
8  #SBATCH --error=slurm-%A_%a.error           # 标准错误定向到文件的文件名
9  #SBATCH --account=supervisor                # 关联付费账号 supervisor (导师的账号名)
10 #SBATCH --gres=gpu:X                        # 每个节点上 GPU 资源的数量，X 不能超过 4
11
12 module load pytorch/1.0_python3.7_gpu      # 加载系统预装 pytorch
13
14 cd $SLURM_SUBMIT_DIR                        # 进入提交脚本时所在的目录
15
16 python main.py
```

salloc - 分配式提交作业

- salloc 将获取作业的分配后执行命令，当命令结束后释放分配的资源。其基本语法为：

```
1 salloc [options] [<command> [command args]]
```

- command 可以是任何是用户想要用的程序，典型的为 xterm 或包含 srun 命令的 shell。
- 如果需要后台执行 salloc，可以设定标准输入为某个文件，如：

```
1 salloc -n16 a.out </dev/null &
```

sacct - 查看记账信息

- sacct 显示激活的或已完成作业或作业步的记账（与机时费对应）信息。

```
~
zhangding@swarm01:~ %  sacct
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
15504000              -       hpxg  zhangding          1     FAILED      1:0
15504000.ex+     extern              zhangding          1  COMPLETED      0:0
15504003           bash       hpxg  zhangding          4 CANCELLED+      0:0
15504006           bash       hpxg  zhangding          4  COMPLETED      0:0
15504006.ex+     extern              zhangding          4  COMPLETED      0:0
15504008           bash       hpxg  zhangding         16  COMPLETED      0:0
15504008.ex+     extern              zhangding         16  COMPLETED      0:0
15504008.0         bash              zhangding         16  COMPLETED      0:0
15504013           bash       hpxg  zhangding         32 CANCELLED+      0:0
15504014           bash       hpxg  zhangding         16  COMPLETED      0:0
15504014.ex+     extern              zhangding         16  COMPLETED      0:0
15504014.0     hostname              zhangding          2  COMPLETED      0:0
15504016       hostname       hpxg  zhangding         16  COMPLETED      0:0
15504016.ex+     extern              zhangding         16  COMPLETED      0:0
15504016.0     hostname              zhangding         16  COMPLETED      0:0
```

Outline

终止作业

- scancel job_id：终止作业
  - 如果想终止一个作业，可利用 scancel job_id 来取消，job_list 可以为以，分隔的作业 ID。

```
1 scancel 15504000
```

挂起排队中尚未运行的作业

- scontrol hold job_list：挂起排队中尚未运行的作业
  - scontrol hold job_list 命令可使得排队中尚未运行的作业（设置优先级为 0）暂停被分配运行。
  - 被挂起的作业将不被执行。
  - 被挂起的作业在用 squeue 命令查询时，显示的 NODELIST(REASON) 状态标志为 JobHeldUser（被用户自己挂起）或 JobHeldAdmin（被系统管理员挂起）。

```
1 scontrol hold 15504000
```

继续排队被挂起的尚未运行作业

- scontrol release job_list：继续排队被挂起的尚未运行作业
  - 被挂起的作业可以利用 scontrol release job_list 来取消挂起，重新进入等待运行状态。

```
1 scontrol release 15504000
```

重新排队作业

- scontrol requeue job_list：重新排队作业
  - 利用 scontrol requeue job_list 重新使得运行中的、挂起的或停止的作业重新进入排队等待运行。

```
1 scontrol requeue 15504000
```

重新挂起作业

- scontrol requeuehold job_list：重新挂起作业
  - 利用 scontrol requeuehold job_list 重新使得运行中的、挂起的或停止的作业重新进入排队。
  - 之后可利用 scontrol release job_list 使其运行。

```
1 scontrol requeuehold 15504000
```

最优先等待运行作业

- scontrol top job_list：最优先等待运行作业
  - 利用 scontrol top job_list 可以使得尚未开始运行的 job_list 作业排到用户自己排队作业的最前面，最优先运行。

```
1 scontrol top 15504000
```

等待某个作业运行完

- scontrol wait job_list：等待某个作业运行完
  - 利用 scontrol wait_job job_id 可以等待某个 job_id 结束后开始运行，一般用于脚本中。

```
1 scontrol wait 15504000
```

更新作业信息

- scontrol update SPECIFICATION：更新作业信息
  - 利用 scontrol update SPECIFICATION 可以更新作业、作业步等信息。
  - SPECIFICATION 格式为 scontaol show job 显示出的。

```
1 scontrol update JobId=15504000 JobName=NewName
```

Outline

Outline

武大超算中心介绍

- 主页：http://hpc.whu.edu.cn/
- 文档：http://docs.hpc.whu.edu.cn/

武汉大学超级计算中心集群计算系统，由 445 台 CPU 服务器、168 台 KNL 服务器和 125 台 GPU 服务器组成，总体计算能力理论峰值为 4600 万亿次。

硬件资源

CPU集群：8832个CPU核心，峰值计算能力350万亿次/秒				
类型	CPU	内存	硬盘	数量
Hp Appollo2000服务器	2*Intel Xeon E5-2630 v3 x86_64，2.4GHz，16核心，Infiniband互联	96GB DDR4 2133MHz ECC	240GB 固态硬盘	375
Inspur 英信服务器	2*Intel(R) Xeon(R) E5-2682 v4 x86_64，2.5GHz，32核心，Infiniband互联	128GB DDR4 2400MHz ECC	240GB 固态硬盘	40
Inspur 天梭服务器	8*Intel(R) Xeon(R) CPU E7-8880 v4，2.20GHz，176核心，Infiniband互联	4TB DDR4 2400MHz ECC	500GB 固态硬盘+8TB 机械硬盘	2
Lenovo SD530服务器	2*Intel(R) Xeon(R) CPU Gold 6248，2.50GHz，40核心，100G OPA互联	384GB DDR4 2933MHz ECC	480GB 固态硬盘	30

硬件资源

KNL集群： 11424个CPU核心，峰值计算能力500万亿次/秒，100G OPA互联				
类型	CPU	内存	硬盘	数量
Intel Knight Landing 7250服务器	1*Intel Xeon Phi Knight Landing，1.4GHz，68核心	96GB DDR4 2400MHz ECC	240GB 固态硬盘	168
GPU集群： 500块Nvidia Tesla V100 16GB，峰值计算能力3750万亿次/秒，100G OPA互联				
类型	CPU	内存	硬盘	数量
Supermicro/DELL GPU服务器	2*Intel(R) Xeon(R) E5-2640 v4 x86_64，2.4GHz，20核心，4*Nvidia Tesla V100 16GB	128GB DDR4 2400MHz ECC	240GB 固态硬盘	125

硬件资源

存储系统：30台IO节点，lustre并行文件系统，3PB				
类型	CPU	内存	硬盘	数量
DELL R730机架式存储， IO节点	Intel Xeon E5-2609 v2 x86_64，2.1GHz，6核	32GB DDR3 1600MHz	2*600GB SAS，16*6TB SATA	10
Inspur英信服务器，机架式存储， IO节点	Intel Xeon E5-2630 v4 x86_64，2.2GHz，10核	120GB DDR3 2400MHz	2*240GB SAS，10*6TB SATA	20

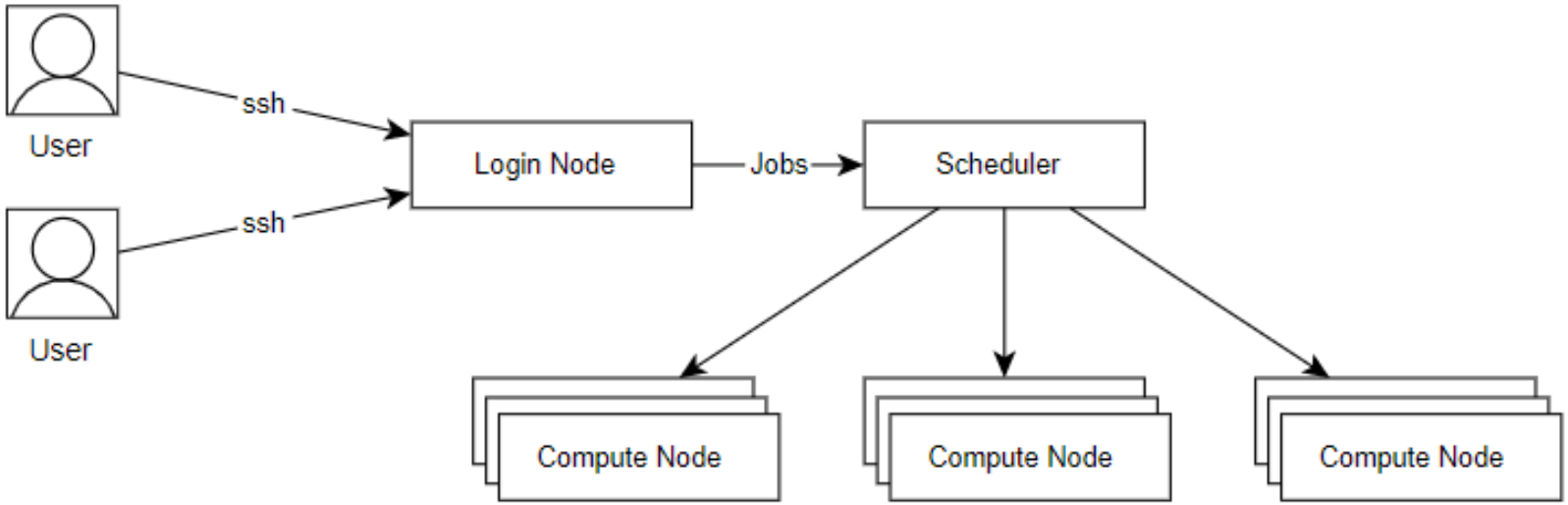
申请账号

账号分两类：

- 免费账号
  - ★ 最大使用 **16 CPU** 核，不能使用 **GPU**！
  - ★ 申请地址：https://hall.whu.edu.cn/infoplus/form/CSZXYHKTSQ/start
- 付费账号
  - ★ 只针对教工账户进行收费
  - ★ 开通办法：填写申请表并发送电子版到超算中心邮箱 hpc@whu.edu.cn 进行审批
  - ★ 校内学生账户可作为子账户关联至老师账户名下使用老师的计算资源
  - ★ 申请关联地址：https://hall.whu.edu.cn/infoplus/form/CSXSZHGLZHSQ/start

Outline

登陆超算


User
ssh
User
ssh
Login Node
Jobs
Scheduler
Compute Node
Compute Node
Compute Node

登陆超算

- ssh 登录地址:
  - **swarm.whu.edu.cn** (202.114.96.179) - 推荐
  - 202.114.96.178
  - 202.114.96.180

★ 在武大校园网内，可以直接连接

★ 在武大校园网外，需要通过 VPN 连接武大校园网后，再连接

存储空间

存储空间分为三个部分：

- ~/
  - 限额 1 GB，文件数量 10,000 个
  - 目录下的文件长期保存
  - 建议仅用来保存用户的环境变量等设置信息
- ~/project
  - 限额 1 TB，文件数量 1,000,000 个
  - 目录下的文件长期保存
  - 主要的计算程序和数据存储空间
- ~/scratch
  - 限额 3 TB，文件数量 1,000,000 个
  - 超过 **3** 个月的数据会自动清理
  - 额外的、临时的数据存储空间

存储空间

使用如下命令查询自己的存储空间使用情况。

```
1 /bin/myDiskQuota
```

```
zhangding@swarm-xfe ~
0 %  ls -l
total 17
lrwxrwxrwx 1 zhangding zhangding 18 Mar 10 10:34 project -> /project/zhangding
lrwxrwxrwx 1 zhangding zhangding 18 Mar 10 10:34 scratch -> /scratch/zhangding

zhangding@swarm-xfe ~
0 %  myDiskQuota
     Dir    DiskUsed   DiskQuota    DiskFree       Files  FilesQuota
    home         43M        1024M         981M        1772       10000
 project          0G        1024G        1024G           2     1000000
 scratch          0G        3072G        3072G           1     1000000

zhangding@swarm-xfe ~
0 %
```

数据传输

武大超算设有专门用于文件传输的服务器：

- 202.114.96.177

如果需要传输大量的小文件，建议先打包成无压缩的单个文件，可以大大加快传输的速度，传输完成之后再解包。

使用预装软件

超算配置了 module 工具来调用系统预装的软件。

查看系统预装软件列表：

```
1 module avail
```

使用预装软件

超算配置了 module 工具来调用系统预装的软件。

查看系统预装软件列表：

```
1 module avail
```

查看已加载软件环境：

```
1 module list
```

使用预装软件

超算配置了 module 工具来调用系统预装的软件。

查看系统预装软件列表：

```
1 module avail
```

查看已加载软件环境：

```
1 module list
```

加载软件环境：

```
1 module load pytorch/1.0_python3.7_gpu
```

使用预装软件

超算配置了 module 工具来调用系统预装的软件。

查看系统预装软件列表：

```
1 module avail
```

查看已加载软件环境：

```
1 module list
```

加载软件环境：

```
1 module load pytorch/1.0_python3.7_gpu
```

卸载已加载的软件环境：

```
1 module unload pytorch/1.0_python3.7_gpu
```

使用预装软件

```
zhangding@swarm01:~
>  module avail
------------------------------------------------------------ /software/modulefiles -------------------------------------------------------------
anaconda/2.7                          java/jdk11(default)                   nvidia/cuda/10.2
anaconda/3.7(default)                 julia/1.0.5                           nwchem/6.8.1
anaconda/3.7.4                        julia/1.5.3                           openfoam/openfoam-6
bcftools/1.10.1                       lammps/201812/lmp_cuda_openmpi        openmpi/1.10.7
caffe/pycaffe                         lammps/201812/lmp_intelcpu_intelmpi   openmpi/2.1.5
cran/R-3.5.1                          lammps/201812/lmp_intelknl_intelmpi   openmpi/2.1.5_IB_gcc7.3
cran/R-3.6.1(default)                 lammps/202003/lmp_cuda_openmpi        openmpi/2.1.5_OPA_gcc7.3
cran/R-4.0.3                          lammps/202003/lmp_intelcpu_intelmpi   openmpi/2.1.5_OPA_gcc7.3_cuda9.0
ctan/texlive/2019(default)            lammps/202010/lmp_cuda_openmpi        openmpi/3.1.3_IB_gcc7.3_singularity3.0.1
dock/dock6.9                          lammps/202010/lmp_intelknl_intelmpi   openmpi/3.1.3_OPA_gcc7.3_cuda9.0_singularity3.0.1
google/go/1.11.2                      libsndfile/1.0.28                     openmpi/4.0.0_IB_gcc7.3_singularity3.0.1
google/go/1.12                        liggghts/3.8.0/lmp_gcc_openmpi        openmpi/4.0.0_OPA_gcc6.3_cuda10.0_singularity3.1.0
google/go/1.16                        make/4.0                              openmpi/4.0.0_OPA_gcc7.3_cuda9.0_singularity3.0.1
google/tensorflow/python2.7-cpu       make/4.2.1                            openmpi/4.0.5_OPA_gcc9.3
google/tensorflow/python2.7-gpu       make/4.3                              openmpi/4.1.0_IB_gcc9.3
google/tensorflow/python3.6-cpu       matlab/R2017b                         perl/5.30.0
google/tensorflow/python3.6-gpu       matlab/R2018b(default)                pgi/2020(default)
gromacs/4.6.7-cpu                     matlab/R2019b                         pytorch/1.0_python3.7_cpu
gromacs/2018.3-cpu                    matlab/R2020a                         pytorch/1.0_python3.7_gpu
gromacs/2019.1-knl                    microsoft/dotnet/dotnet-2.1           scl/gcc4.9
gromacs/2019.4-gpu                    microsoft/dotnet/dotnet-5.0           scl/gcc5.3
gromacs/2021-cpu                      mpich/3.4.1                           scl/gcc6.3
hdf5/1.8.16                           mpich/3.4.1-gcc9.3                    scl/gcc7.3
hdf5/1.8.20                           mpich/3.4.1-intel2017                 scl/gcc8.3
hdf5/1.10.5                           mpich/3.4.1-intel2021                 scl/gcc9.3
htslib/1.10.1                         namd/2.1.3-cpu(default)               singularity/2.6.0
intel/intelpython/2.7                 namd/2.1.3-gpu                        singularity/3.0.1
intel/intelpython/3.6(default)        netcdf-fortran/4.4.0                  singularity/3.1.0
intel/intelpython/2021                netcdf/4.4.0                          singularity/3.2.0
intel/oneapi/2021                     nvidia/cuda/9.0                       singularity/3.7(default)
intel/parallelstudio/2013             nvidia/cuda/9.1                       tools/rar
intel/parallelstudio/2017u8(default)  nvidia/cuda/9.2                       vtk/7.1.1_gcc7.3_openmpi3.1.3
intel/parallelstudio/2019             nvidia/cuda/10.0(default)             yade/2020.01a
java/jdk8                             nvidia/cuda/10.1
```

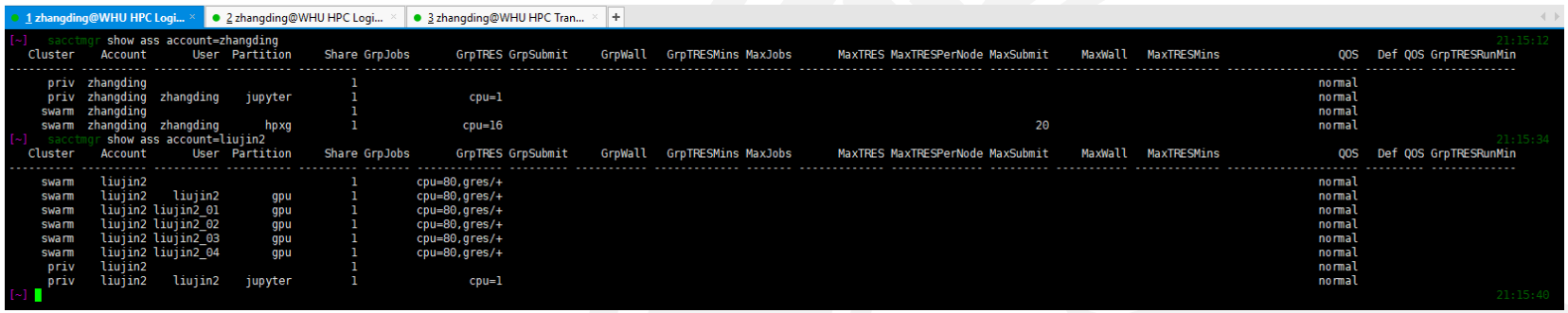
查看账户核心/资源限制信息

- 如何查看自己账户的核心/资源限制信息？

```
1 sacctmgr show ass account=xxxx
```

```
1 sacctmgr show ass user=‘whoami‘
```

查看账户核心/资源限制信息

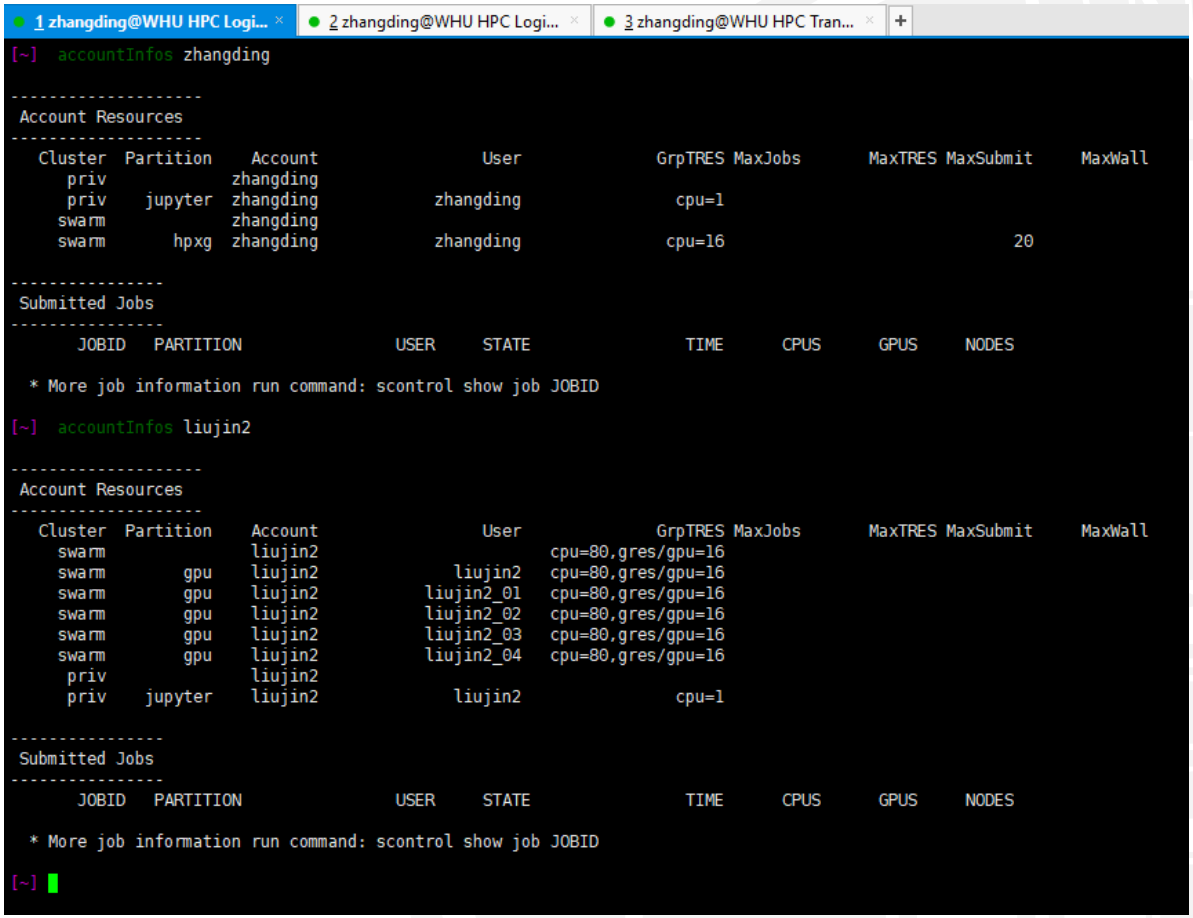


查看导师账号下资源及使用情况

- 如何查看导师 (付费) 账号下资源及使用情况？

```
1 accountInfos xxxx
```

查看导师账号下资源及使用情况



分区有空闲节点，为什么我的作业还在排队

- 为什么 sinfo 查看对应的分区有空闲节点，但是我的作业却还在排队？
  - ★ 整个队列中可能有需要占用多节点的高优先级任务正在等待资源，调度器会一定程度上为这些作业保留资源，以确保它们能够运行。

为什么我的作业没有运行

Some common reasons why jobs are pending:

Priority	Resources being reserved for higher priority job
Resources	Required resources are in use
Dependency	Job dependencies not yet satisfied Waiting for
Reservation	advanced reservation
AssociationJobLimit	User or account job limit reached
AssociationResourceLimit	User or account resource limit reached
AssociationTimeLimit	User or account time limit reached
QOSJobLimit	Quality Of Service (QOS) job limit reached
QOSResourceLimit	Quality Of Service (QOS) resource limit reached
QOSTimeLimit	Quality Of Service (QOS) time limit reached

Outline

Outline

Demo

Demo

Thanks!