



Slurm用户手册

文档版本 **V1.0**

发布日期 **2020-07-****01**

尊敬的用户：

版权 © 浪潮 2017. 版权所有

未经事先书面同意，本文档的任何部分不得复制或以任何形式或任何方式修改、外传

注:您购买的产品、服务或特性等应受浪潮集团商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，浪潮集团对本文档内容不做任何明示或默示的声明或保证。由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。浪潮保留本软件的所有权和所有相关的知识产权，获得、使用本软件的任何主体，均应依照法律法规之规定、依照合法协议之约定，尊重并保护浪潮及相关权利人对本软件及相关知识产权的权利，该尊重和保护应不低于浪潮对其他权利人所负担之义务。

Inspur和“浪潮”是浪潮集团的注册商标。

|  |  |
| --- | --- |
| 技术服务电话： | 4008600011 |
| 地 址： | 中国济南市浪潮路1036号  浪潮电子信息产业股份有限公司 |
| 邮 编： | 250101 |

# 前言

本文档主要介绍了Slurm的功能，应用场景及其操作指导。

## 读者对象

本文档主要适用于以下工程师：

* 技术支持工程师
* 维护工程师

## 修改记录

|  |  |  |
| --- | --- | --- |
| **文档版本** | **发布日期** | **修改说明** |
| V1.0 | 2020-07-01 | 第一次正式发布 |

# 目录

[前言 II](#_Toc44524712)

[读者对象 II](#_Toc44524713)

[修改记录 III](#_Toc44524714)

[目录 IV](#_Toc44524715)

[1 文档指南 6](#_Toc44524716)

[2 资源控制 7](#_Toc44524717)

[2.1 关联关系 7](#_Toc44524718)

[2.2 设置服务质量 8](#_Toc44524719)

[2.3 添加账号和用户 9](#_Toc44524720)

[3 脚本和任务 11](#_Toc44524721)

[3.1 脚本样例 11](#_Toc44524722)

[3.2 运行脚本 13](#_Toc44524723)

[4 命令组成 13](#_Toc44524724)

[4.1 命令概览 13](#_Toc44524725)

[4.2 基本术语 14](#_Toc44524726)

[4.3 常用指令参考 15](#_Toc44524727)

[4.3.1 作业提交 15](#_Toc44524728)

[4.3.2 记帐信息 23](#_Toc44524729)

[4.3.3 作业管理 24](#_Toc44524730)

[4.4 命令详解 25](#_Toc44524731)

[4.4.1 sacct 25](#_Toc44524732)

[4.4.2 sacctmgr 27](#_Toc44524733)

[4.4.3 salloc 27](#_Toc44524734)

[4.4.4 sattach 28](#_Toc44524735)

[4.4.5 sbatch 28](#_Toc44524736)

[4.4.6 sbcast 29](#_Toc44524737)

[4.4.7 scancel 30](#_Toc44524738)

[4.4.8 sinfo 31](#_Toc44524739)

[4.4.9 squeue 36](#_Toc44524740)

[4.4.10 sreport 40](#_Toc44524741)

[4.4.11 srun 43](#_Toc44524742)

[4.4.12 sshare 44](#_Toc44524743)

[4.4.13 sstat 45](#_Toc44524744)

[4.4.14 strigger 47](#_Toc44524745)

[5 状态监控 48](#_Toc44524746)

[5.1 节点状态 48](#_Toc44524747)

[5.2 作业状态 49](#_Toc44524748)

[5.3 作业异常原因 50](#_Toc44524749)

[6 资源管理 52](#_Toc44524750)

[6.1 节点：Node 52](#_Toc44524751)

[6.1.1 主要输出项 52](#_Toc44524752)

[6.1.2 查看节点详细信息 53](#_Toc44524753)

[6.1.3 修改节点信息 53](#_Toc44524754)

[6.2 分区/队列：Partition 53](#_Toc44524755)

[6.2.1 主要输出项 54](#_Toc44524756)

[6.2.2 查看队列详细信息 55](#_Toc44524757)

[6.2.3 新建队列 55](#_Toc44524758)

[6.2.4 更新队列 56](#_Toc44524759)

[6.2.5 删除队列 56](#_Toc44524760)

[A 如何获取帮助 56](#_Toc44524761)

[A.1 收集必要的故障信息 56](#_Toc44524762)

[A.2如何使用文档 57](#_Toc44524763)

[A.3获取技术支持 57](#_Toc44524764)

[B 术语和缩略语 57](#_Toc44524765)

# 文档指南

表 1‑1 文档指南

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **文档名称** | **内容介绍** | **手册获取** |
| 了解产品 | 用户手册 | 介绍Slurm基本配置与操作 | 《Slurm用户手册》 |

# 命令组成

## 命令概览

|  |  |
| --- | --- |
| **命令** | **说明** |
| sacct | 在Slurm作业记帐日志或Slurm数据库中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。 |
| sacctmgr | 用于查看和修改Slurm帐户信息。 |
| salloc | 获取Slurm作业分配（一组节点），执行命令，然后在命令完成后释放分配。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。 |
| sbatch | 将批处理脚本提交给Slurm。此脚本一般也可含有一个或多个srun命令启动并行任务。 |
| sbcast | 将文件传输到分配给Slurm作业的节点。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 | 用于表示受Slurm控制的作业或作业步骤。 |

## 基本术语

|  |  |
| --- | --- |
| **术语** | **说明** |
| socket | CPU插槽，可以简单理解为CPU |
| core | CPU核，单颗CPU可以具有多颗CPU核 |
| job | 作业 |
| job step | 作业步，单个作业（job）可以有个多作业步 |
| tasks | 任务数，单个作业或作业步可有多个任务，一般一个任务需一个CPU核，可理解为所需的CPU核数 |
| rank | 秩，如MPI进程号 |
| partition | 队列、分区。作业需在特定队列中运行，一般不同队列允许的资源不一样，比如单作业核数等 |
| stdin | 标准输入文件，一般指可以通过屏幕输入或采用<文件名方式传递给程序的文件，对应C程序中的文件描述符 |
| stdout | 标准输出文件，程序运行正常时输出信息到的文件，一般指输出到屏幕的，并可采用>文件名定向到的文件，对应C程序中的文件描述符1。 stderr：标准出错文件，程序运行出错时输出信息到的文件，一般指也输出到屏幕，并可采用2>定向到的文件（注意这里的2），对应C程序中的文件描述符2 |

## 常用指令参考

### 作业提交

**指令说明**

|  |  |
| --- | --- |
| **指令** | **说明** |
| salloc | 为需实时处理的作业分配资源，提交后等获得作业分配的资源后运 行，作业结束后返回命令行终端 |
| sbatch | 批处理提交，提交后无需等待立即返回命令行终端 |
| srun | 运行并行作业，等获得作业分配的资源并运行，作业结束后返回命令行终端 |

**主要参数**

• ­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。

• ­­accel­bind=<options>： srun特有，控制如何绑定作业到GPU、网络等特定资源，支持同时多个选项，支持的选项如下：

– g：绑定到离分配的CPU最近的GPU

– m：绑定到离分配的CPU最近的MIC

– n：绑定到离分配的CPU最近的网卡

– v：详细模式，显示如何绑定GPU和网卡等等信息

• ­­acctg­freq：指定作业记账和剖面信息采样间隔。支持的格式为­­acctg­freq=<datatype>=<interval>其中<datatype>=<interval>指定了任务抽样间隔或剖面抽样间隔。多个<datatype>=<interval>可以采用,分隔（默认为30秒）：

– task=<interval>：以秒为单位的任务抽样（需要jobacct\_gather插件启用）和任务剖面（需要acct\_gather\_profile插件启用）间隔。

– energy=<interval>：以秒为单位的能源剖面抽样间隔，需要acct\_gather\_energy插件启用。

– network=<interval>：以秒为单位的InfiniBand网络剖面抽样间隔，需要acct\_gather\_infiniband插件启用。

– filesystem=<interval>：以秒为单位的文件系统剖面抽样间隔，需要acct\_gather\_filesystem插件启用。

• ­B ­­extra­node­info=<sockets[:cores[:threads]]>：选择满足<sockets[:cores[:threads]]>的节点， \*表示对应选项不做限制。对应限制可以采用下面对应选项：

– ­­sockets­per­node=<sockets>

– ­­cores­per­socket=<cores>

– ­­threads­per­core=<threads>

• ­­bcast[=<dest\_path>]：srun特有，复制可执行程序到分配的计算节点的[<dest\_path>]目录。如指定了<dest\_path>，则复制可执行程序到此；如没指定则复制到当前工作目录下的“slurm\_bcast\_<job\_id>.<step\_id>”。如srun ­­bcast=/tmp/mine ­N3 a.out将从当前目录复制a.out到每个分配的节点的/tmp/min并执行。

• ­­begin=<time>：设定开始分配资源运行的时间。时间格式可为HH:MM:SS，或添加AM、 PM等，也可采用MMDDYY、 MM/DD/YY或YYYY­MM­DD格式指定日期，含有日期及时间的格式为： YYYY­MM­DD[THH:MM[:SS]]，也可以采用类似now+时间单位的方式，时间单位可以为seconds（默认）、 minutes、 hours、days和weeks、 today、 tomorrow等，例如：

– ­­begin=16:00： 16:00开始。

– ­­begin=now+1hour： 1小时后开始。

– ­­begin=now+60： 60秒后开始（默认单位为秒）。

– ­­begin=2017­02­20T12:34:00： 2017­02­20T12:34:00开始。

• ­­bell：分配资源时终端响铃，参见­­no­bell。

• ­­cpu­bind=[quiet,verbose,]type： srun特有，设定CPU绑定模式。

• ­­comment=<string>：作业说明。

• ­­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）。

• ­­cores­per­socket=<cores>：分配的节点需要每颗CPU至少<cores>CPU核。

• ­­cpus­per­gpu=<ncpus>：每颗GPU需<ncpus>个CPU核，与­­cpus­per­task不兼容。

• ­c, ­­cpus­per­task=<ncpus>：每个进程需<ncpus>颗CPU核，一般运行OpenMP等多线程程序时需，普通MPI程序不需。

• ­­deadline=<OPT>：如果在此deadline（start > (deadline ­ time[­min]）之前没有结束，那么移除此作业。默认没有deadline，有效的时间格式为：

– HH:MM[:SS] [AM|PM]

– MMDD[YY]或MM/DD[/YY]或MM.DD[.YY]

– MM/DD[/YY]­HH:MM[:SS]

– YYYY­MM­DD[THH:MM[:SS]]]

• ­d, ­­dependency=<dependency\_list>：满足依赖条件<dependency\_list>后开始分配。<dependency\_list>可以为<type:job\_id[:job\_id][,type:job\_id[:job\_id]]>或<type:job\_id[:job\_id][?type:job\_id[:job\_id]]>。依赖条件如果用,分隔，则各依赖条件都需要满足；如果采用?分隔，那么只要任意条件满足即可。可以为：

– after:job\_id[:jobid...]：当指定作业号的作业结束后开始运行。

– afterany:job\_id[:jobid...]：当指定作业号的任意作业结束后开始运行。

– aftercorr:job\_id[:jobid...]：当相应任务号任务结束后，此作业组中的开始运行。

– afternotok:job\_id[:jobid...]：当指定作业号的作业结束时具有异常状态（非零退出码、节点失效、超时等）时。

– afterok:job\_id[:jobid...]：当指定的作业正常结束（退出码为0）时开始运行。

– expand:job\_id：分配给此作业的资源将扩展给指定作业。

– singleton：等任意通账户的相同作业名的前置作业结束时。

• ­D, ­­chdir=<path>：在切换到<path>工作目录后执行命令。

• ­e, ­­error=<mode>：设定标准错误如何重定向。非交互模式下，默认srun重定向标准错误到与标准输出同样的文件（如指定）。此参数可以指定重定向到不同文件。如果指定的文件已经存在，那么将被覆盖。参见IO重定向。 salloc无此选项。

• ­­epilog=<executable>： srun特有，作业结束后执行<executable>程序做相应处理。

• ­E, ­­preserve­env：将环境变量SLURM\_NNODES和SLURM\_NTASKS传递给可执行文件，而无需通过计算命令行参数。

• ­­exclusive[=user|mcs]：排他性运行，独占性运行，此节点不允许其他[user]用户或mcs选项的作业共享运行作业。

• ­­export=<[ALL,]environment variables|ALL|NONE>： sbatch与srun特有，将环境变量传递给应用程序

– ALL：复制所有提交节点的环境变量，为默认选项。

– NONE：所有环境变量都不被传递，可执行程序必须采用绝对路径。一般用于当提交时使用的集群与运行集群不同时。

– [ALL,]environment variables：复制全部环境变量及特定的环境变量及其值，可以有多个以,分隔的变量。如： “­­export=EDITOR,ARG1=test”。

• ­­export­file=<filename | fd>： sbatch特有，将特定文件中的变量设置传递到计算节点，这允许在定义环境变量时有特殊字符。

• ­F, ­­nodefile=<node file>：类似­­nodelist指定需要运行的节点，但在一个文件中含有节点列表。

• ­G, ­­gpus=[<type>:]<number>：设定使用的GPU类型及数目，如­­gpus=v100:2。

• ­­gpus­per­node=[<type>:]<number>：设定单个节点使用的GPU类型及数目。

• ­­gpus­per­socket=[<type>:]<number>：设定每个socket需要的GPU类型及数目。

• ­­gpus­per­task=[<type>:]<number>：设定每个任务需要的GPU类型及数目。

• ­­gres=<list>：设定通用消费资源，可以以,分隔。每个<list>格式为“name[[:type]:count]”。name是可消费资源； count是资源个数，默认为1；

• ­H, ­­hold：设定作业将被提交为挂起状态。挂起的作业可以利用scontrol release<job\_id>使其排队运行。

• ­h, ­­help：显示帮助信息。

• ­­hint=<type>：绑定任务到应用提示：

– compute\_bound：选择设定计算边界应用：采用每个socket的所有CPU核，每颗CPU核一个进程。

– memory\_bound：选择设定内存边界应用：仅采用每个socket的1颗CPU核，每颗CPU核一个进程。

no multithread：在in­core multi­threading是否采用额外的线程，对通信密集型应用有益。仅当task/affinity插件启用时。

– help：显示帮助信息

• ­I, ­­immediate[=<seconds>]： salloc与srun特有，在<seconds>秒内资源未满足的话立即退出。格式可以为“­I60”，但不能之间有空格是“­I 60”。

• ­­ignore­pbs： sbatch特有，忽略批处理脚本中的“#PBS”选项。

• ­i, ­­input=<mode>： sbatch与srun特有，指定标准输入如何重定向。默认， srun对所有任务重定向标准输入为从终端。参见IO重定向。

• ­J, ­­job­name=<jobname>：设定作业名<jobname>，默认为命令名。

• ­­jobid=<jobid>： srun特有，初始作业步到某个已分配的作业号<jobid>下的作业下，类似设置了SLURM\_JOB\_ID环境变量。仅对作业步申请有效。

• ­K, ­­kill­command[=signal]： salloc特有，设定需要终止时的signal，默认，如没指定，则对于交互式作业为SIGHUP，对于非交互式作业为SIGTERM。格式类似可以为“­K1”，但不能包含空格为“­K 1”。

• ­K, –kill­on­bad­exit[=0|1]： srun特有，设定是否任何一个任务退出码为非0时，是否终止作业步。

• ­k, ­­no­kill：如果分配的节点失效，那么不会自动终止。

• ­L, ­­licenses=<license>：设定使用的<license>。

• ­l, ­­label： srun特有，在标注正常输出或标准错误输出的行前面添加作业号。

• ­­mem=<size[units]>：设定每个节点的内存大小，后缀可以为[K|M|G|T]，默认为MB。

• ­­mem­per­cpu=<size[units]>：设定分配的每颗CPU对应最小内存，后缀可以为[K|M|G|T]，默认为MB。

• ­­mem­per­gpu=<size[units]>：设定分配的每颗GPU对应最小内存，后缀可以为[K|M|G|T]，默认为MB。

• ­­mincpus=<n>：设定每个节点最小的逻辑CPU核/处理器。

• ­­mpi=<mpi\_type>： srun特有，指定使用的MPI环境， <mpi\_type>可以主要为：

– list：列出可用的MPI以便选择。

– pmi2：启用PMI2支持

– pmix：启用PMIx支持

– none：默认选项，多种其它MPI实现有效。

• ­­multi­prog： srun特有，让不同任务运行不同的程序及参数，需指定一个配置文件，参见MULTIPLE PROGRAM CONFIGURATION。

• ­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。

• ­­nice[=adjustment]：设定NICE调整值。负值提高优先级，正值降低优先级。调整范围为： +/­ 2147483645。

• ­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。 ­­cpus­per­task选项可以改变此默认选项。

• ­­ntasks­per­core=<ntasks>：每颗CPU核运行<ntasks>个任务，需与­n, ­­ntasks=<number>配合，并自动绑定<ntasks>个任务到每个CPU核。仅对作业起作用，不对作业步起作用。

• ­­ntasks­per­node=<ntasks>：每个节点运行<ntasks>个任务，需与­n, ­­ntasks=<number>配合。仅对作业起作用，不对作业步起作用。

• ­­ntasks­per­socket=<ntasks>：每颗CPU运行<ntasks>个任务，需与­n, ­­ntasks=<number>配合，并绑定<ntasks>个任务到每颗CPU。仅对作业起作用，不对作业步起作用。

• ­­no­bell： salloc特有，资源分配时不终端响铃。参见­­bell。

• ­­no­shell： salloc特有，分配资源后立即退出，而不运行命令。但Slurm作业仍旧被生成，在其激活期间，且保留这些激活的资源。用户会获得一个没有附带进程和任务的作业号，用户可以采用提交srun命令到这些资源。

• ­o, ­­output=<mode>： sbatch与srun特有，指定标准输出重定向。在非交互模式中，默认srun收集各任务的标准输出，并发送到吸附的终端上。采用­­output可以将其重定向到同一个文件、每个任务一个文件或/dev/null等。参见IO重定向。

• ­­open­mode=<append|truncate>： sbtach与srun特有，对标准输出和标准错误输出采用追加模式还是覆盖模式。

• ­O, ­­overcommit：采用此选项可以使得每颗CPU运行不止一个任务。

• ­­open­mode=<append|truncate>：标准输出和标准错误输出打开文件的方式：

– append：追加。

– truncate：截断覆盖。

• ­p, ­­partition=<partition\_names>：使用<partition\_names>队列

• ­­prolog=<executable>： srun特有，作业开始运行前执行<executable>程序，做相应处理。

• ­Q, ­­quiet：采用安静模式运行，一般信息将不显示，但错误信息仍将被显示。

• ­­qos=<qos>：需要特定的服务质量(QS)。

• ­­quit­on­interrupt： srun特有，当SIGINT (Ctrl­C)时立即退出。

• ­r, ­­relative=<n>： srun特有，在当前分配的第n节点上运行作业步。该选项可用于分配一些作业步到当前作业占用的节点外的节点，节点号从0开始。 ­r选项不能与­w或­x同时使用。仅对作业步有效。

• ­­reservation=<name>：从<name>预留资源分配。

• –requeue： sbtach特有，当非配的节点失效或被更高级作业抢占资源后，重新运行该作业。相当于重新运行批处理脚本，小心已运行的结果被覆盖等。

• ­­no­requeue：任何情况下都不重新运行。

• ­S, ­­core­spec=<num>：指定预留的不被作业使用的各节点CPU核数。但也会被记入费用。

• ­­signal=<sig\_num>[@<sig\_time>]：设定到其终止时间前信号时间<sig\_time>秒时的信号。由于Slurm事件处理的时间精度，信号有可能比设定时间早60秒。信号可以为10或USER1，信号时间sig\_time必须在0到65535之间，如没指定，则默认为60秒。

• ­­sockets­per­node=<sockets>：设定每个节点的CPU颗数。

• ­T, ­­threads=<nthreads>： srun特有，限制从srun进程发送到分配节点上的并发线程数。

• ­t, ­­time=<time>：作业最大运行总时间<time>，到时间后将被终止掉。时间<time>的格式可以为：分钟、分钟:秒、小时:分钟:秒、天­小时、天­小时:分钟、天­小时:分钟:秒

• ­­task­epilog=<executable>： srun特有，任务终止后立即执行<executable>，对应于作业步分配。

• ­­task­prolog=<executable>： srun特有，任务开始前立即执行<executable>，对应于作业步分配。

• ­­test­only： sbatch与srun特有，测试批处理脚本，并预计将被执行的时间，但并不实际执行脚本。

• ­­thread­spec=<num>：设定指定预留的不被作业使用的各节点线程数。

• ­­threads­per­core=<threads>：每颗CPU核运行<threads>个线程。

• ­­time­min=<time>：设定作业分配的最小时间，设定后作业的运行时间将使得­­time设定的时间不少于­­time­min设定的。时间格式为： minutes、minutes:seconds、hours:minutes:seconds、days­hours、days­hours:minutes和days­hours:minutes:seconds。

• ­­usage：显示简略帮助信息

• ­­tmp=<size[units]>：设定/tmp目录最小磁盘空间，后缀可以为[K|M|G|T]，默认为MB。

• ­u, ­­usage：显示简要帮助信息。

• ­u, –unbuffered： srun特有，该选项使得输出可以不被缓存立即显示出来。默认应用的标准输出被glibc缓存，除非被刷新(flush)或输出被设定为步缓存。9 提交作业命令共同说明 42

• ­­use­min­nodes：设定如果给了一个节点数范围，分配时，选择较小的数。

• ­V, ­­version：显示版本信息。

• ­v, ­­verbose：显示详细信息，多个v会显示更详细的详细。

• ­W, ­­wait=<seconds>：设定在第一个任务结束后多久结束全部任务。

• ­w, ­­nodelist=<host1,host2,... or filename>：在特定<host1,host2>节点或filename文件中指定的节点上运行。

• ­­wait­all­nodes=<value>： salloc与sbatch特有，控制当节点准备好时何时运行命令。默认，当分配的资源准备好后salloc命令立即返回。 <value>可以为：

– 0：当分配的资源可以分配时立即执行，比如有节点以重启好。

– 1：只有当分配的所有节点都准备好时才执行

• ­X, ­­disable­status： srun特有，禁止在srun收到SIGINT (Ctrl­C)时显示任务状态。

• ­x, ­­exclude=<host1,host2,... or filename>：在特定<host1,host2>节点或filename文件中指定的节点之外的节点上运行。

### 记帐信息

**指令说明**

sacct：在Slurm作业记帐日志或Slurm数据库中显示所有作业和作业步骤的记帐数据。

**常用参数**

­­– accounts=<name>：显示指定账户的作业。

­­– endtime=<time>：设定显示的截止时间之前的作业。

– ­­format=<spec>：格式化输出。

– ­­name=<jobname>：设定显示作业名的信息。

– ­­partition=<name>：设定采用队列的作业信息。

– ­­state=<state\_list>：显示特定状态的作业信息。

**指令说明**

sacctmgr：查看和修改Slurm帐户信息。

**常用参数**

选项：

--immediate 立即变更

--parseable 以"|"分隔的输出

命令：

create <ENTITY> <SPECS> 创建一个实体。

add <ENTITY> <SPECS> 增加一个实体，与create命令相同。

delete <ENTITY> where <SPECS>

list <ENTITY> [<SPECS>]

modify <ENTITY> where <SPECS> set <SPECS>

ENTITY：account、cluster、qos、user

### 作业管理

**指令说明**

salloc：为需实时处理的作业分配资源，提交后等获得作业分配的资源后运 行，作业结束后返回命令行终端。

sbatch：批处理提交，提交后无需等待立即返回命令行终端。

srun：运行并行作业，等获得作业分配的资源并运行，作业结束后返回命令 行终端

**常用参数**

* ***scancel*：取消作业**

jobid<job\_id\_list>：设定作业号。

name=<name>：设定作业名。

partition=<name>：设定采用队列的作业。

qos=<name>：设定采用的服务质量(QOS)的作业。

reservation=<name>：设定采用了预留测略的作业。

nodelist=<name>：设定采用特定节点名的作业。

* ***squeue*：查看作业信息**

format=<spec>：格式化输出。

jobid<job\_id\_list>：设定作业号。

name=<name>：设定作业名。

partition=<name>：设定采用队列的作业。

qos=<name>：设定采用的服务质量(QOS)的作业。

start：显示作业开始时间。

state=<state\_list>：显示特定状态的作业信息。

* ***scontrol*：查看作业、节点和队列等信息**

­­details：显示更详细信息。

­­oneline：所有信息显示在同一行。

show ENTITY ID：显示特定入口信息， ENTITY可为： job、 node、 partition等， ID可为作业号、节点名、队列名等。

update SPECIFICATION：修改特定信息，用户一般只能修改作业的。

## 命令详解

### sacct

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。

**主要参数说明**

• ­b, ­­brief：显示简要信息，主要包含：作业号jobid、状态status和退出码exitcode。

• ­c, ­­completion：显示作业完成信息而非记账信息。

• ­e, ­­helpformat：显示当采用 ­­format指定格式化输出的可用格式。

• ­E end\_time, ­­endtime=end\_time：显示在end\_time时间之前（不限作业状态）的作业。有效时间格式：

– HH:MM[:SS] [AM|PM]

– MMDD[YY] or MM/DD[/YY] or MM.DD[.YY]

– MM/DD[/YY]­HH:MM[:SS]

– YYYY­MM­DD[THH:MM[:SS]]

• ­i, ­­nnodes=N：显示在特定节点数上运行的作业(N = min[­max)]。

• ­j job(.step) , ­­jobs=job(.step)：限制特定作业号（步）的信息，作业号（步）可以以“,”分隔。

• ­l, ­­long：显示详细信息。

• ­n, ­­noheader：不显示信息头（显示出的信息的第一行，表示个列含义）。

• ­N node\_list, ­­nodelist=node\_list：显示运行在特定节点的作业记账信息。

• ­­name=jobname\_list：显示特定作业名的作业记账信息。

• ­o, ­­format：以特定格式显示作业记账信息，格式间采用,分隔，利用­e, ­­

helpformat可以查看可用的格式。各项格式中%NUMBER可以限定格式占用的

字符数，比如format=name%30，显示name列最多30个字符，如数字前有­则该列左对齐（默认时右对齐）。

• ­r, ­­partition：显示特定队列的作业记账信息。

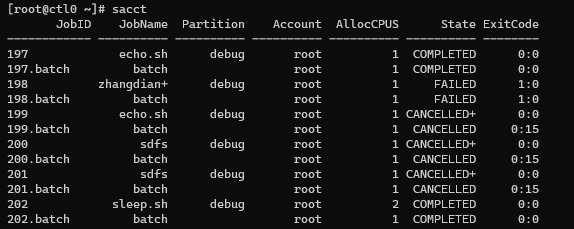
• ­R reason\_list, ­­reason=reason\_list：显示由于reason\_list（以,分隔）原因没有被调度的作业记账信息。

• ­s state\_list, ­­state=state\_list：显示state\_list（以,分隔）状态的作业记账信息。

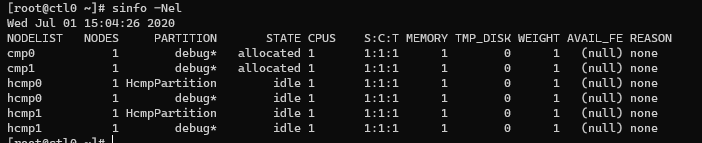
• ­S, ­­starttime：显示特定时间之后开始运行的作业记账信息，有效时间格式参见前面­E参数。

**示例**

1、默认的指令调用



2、报告面向节点的详细信息和精确匹配信息



### sacctmgr

sacctmgr用于查看或修改Slurm账户信息。帐户信息在数据库中维护，由slurmdbd（Slurm数据库守护程序）提供接口。这个数据库可以作为一个站点多台计算机的用户和计算机信息的中央仓库。Slurm账户信息是根据四个参数来记录的，这四个参数构成了所谓的关联。这些参数是用户、集群、分区和账户。用户是登录名。集群是slurm.conf配置文件中ClusterName参数指定的Slurm管理集群的名称。分区是该集群上Slurm分区的名称。预期的操作模式是启动 sacctmgr 命令，添加、删除、修改和/或列出关联记录，然后提交更改并退出。

注意：Slurm数据库的内容是以小写字母来维护的. 这可能会导致某些 sacctmgr 命令的输出与其他 Slurm 命令的输出不同。

**指令格式**

sacctmgr [<OPTION>] [<COMMAND>]

### salloc

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：

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。如果后面没有跟命令，那么将执行Slurm系统slurm.conf配置文件中通过SallocDefaultCommand设定的命令。如果SallocDefaultCommand没有设定，那么将执行用户的默认shell。

注意： salloc逻辑上包括支持保存和存储终端行设置，并且设计为采用前台方式执行。如果需要后台执行salloc，可以设定标准输入为某个文件，如：salloc ­n16 a.out </dev/null &。

**参数说明**

参见 3.3.1

### sattach

sattach可以吸附到一个运行中的Slurm作业步，通过吸附，可以获取所有任务

的IO流等，有时也可用于并行调试器。

**指令格式**

sattach [options] <jobid.stepid>

**参数说明**

• ­h, ­­help：显示帮助信息。

• --input-filter[=]<task number>、 --output-filter[=]<task number>、 --error-filter[=]<tasknumber>：仅传递标准输入到一个单独任务或打印一个单个任务中的标准输出或标准错误输出。

• ­l, ­­label：在每行前显示其对应的任务号。

• ­­layout：联系slurmctld获得任务层信息，打印层信息后退出吸附作业步。

• ­­pty：在伪终端上执行0号任务。与­­input­filter、 ­­output­filter或­­error­filter不兼容。

• ­Q, ­­quiet：安静模式。不显示一般的sattach信息，但错误信息仍旧显示。

• ­u, ­­usage：显示简要帮助信息。

• ­V, ­­version：显示版本信息。

• ­v, ­­verbose：显示冗余信息。

### sbatch

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。

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。

**参数说明**

参见 3.3.1

### sbcast

sbcast命令可以将文件同步到各计算节点对应目录。

当前，用户主目录是共享的，一般不需要此命令，如果用户需要将某些文件传递到分配给作业的各节点/tmp等非共享目录，那么可以考虑此命令。

命令的基本语法格式为：

sbcast [­CfFjpstvV] SOURCE DEST。

此命令仅对批处理作业或在Slurm资源分配后生成的shell中起作用。 SOURCE是当前节点上文件名， DEST为分配给此作业的对应节点将要复制到文件全路径。

**常用参数**

• ­C [library], ­­compress[=library]：设定采用压缩传递，及其使用的压缩库，[library]可以为lz4（默认）、 zlib。

• ­f, ­­force：强制模式，如果目标文件存在，那么将直接覆盖。

• ­F number, ­­fanout=number：设定用于文件传递时的消息扇出，当前最大值为8。

• ­j jobID[.stepID], ­­jobid=jobID[.stepID]：指定使用的作业号。

• ­p, ­­preserve：保留源文件的修改时间、访问时间和模式等。

• ­s size, ­­size=size：设定广播时使用的块大小。 size可以具有k或m后缀，默认单位为比特。默认大小为文件大小或8MB。

• ­t seconds, fB­­timeout=seconds：设定消息的超时时间。

• ­v, ­­verbose：显示冗余信息。

• ­V, ­­version：显示版本信息。

### scancel

Scancel用于发信号或取消作业，作业阵列或作业步骤。可以使用作业规范过滤器或特定作业和/或作业步骤ID的空格分隔列表来发信号通知任意数量的作业或作业步骤。如果使用数组ID值指定了作业数组的作业ID，则仅该作业数组元素将被取消。如果指定的作业数组的作业ID没有数组ID值，则所有作业数组元素将被取消。当异构作业处于挂起状态时，只能取消整个作业，而不是其单个组件。取消未处于挂起状态的异构作业的单个组件的请求将返回错误。作业开始执行后，可以取消单个组件。作业或作业步骤只能由该作业的所有者或用户root发出信号。

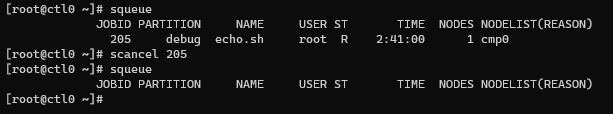
**指令格式**

scancel [OPTIONS...] [job\_id[\_array\_id][.step\_id]] [job\_id[\_array\_id][.step\_id]...]

**示例**

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，如：

hmli@login01:~$ scancel 7



### sinfo

sinfo用于查看运行Slurm的系统的分区和节点信息。

**指令格式**

sinfo [OPTIONS…]

**参数说明**

**-a, --all**

显示有关所有分区的信息。 这将导致显示有关配置为隐藏的分区和用户组不可用的分区的信息.

**-d, --dead**

如果设置，则仅报告非响应（死）节点的状态信息.

**-e, --exact**

如果设置，请不要在多个节点上对节点信息进行分组，除非要报告的配置相同。 否则，节点的cpu计数，内存大小和磁盘空间将以最小值列出，后面是带有相同分区和状态（例如“250+”）的节点的“ +”号.

**--federation**

如果联盟的成员之一，则显示联盟中的所有分区.

**-h, --noheader**

Do not print a header on the output.

**--help**

Print a message describing all sinfo options.

**--hide**

不显示有关隐藏分区的信息。 默认情况下，将不会显示配置为隐藏或用户组不可用的分区（即默认行为）.

**-i <seconds>, --iterate=<seconds>**

定期打印状态。在两次报告之间睡眠指定的秒数。默认情况下，打印带有标题的时间戳.

**--local**

仅显示此集群本地的作业。忽略此联盟中的其他集群（如果有）. Overrides --federation.

**-l, --long**

打印更多详细信息。如果指定了--format选项，则将忽略此设置.

**-M, --clusters=<string>**

向其发出命令的集群。多个群集名称可以用逗号分隔。值“all”将查询在所有群集上运行。请注意，SlurmDBD必须启动，此选项才能正常工作。此选项隐式设置--local选项.

**-n <nodes>, --nodes=<nodes>**

仅打印有关指定节点的信息。多个节点可以用节点范围表达式逗号分隔或表示。例如，“linux [00-07]”将指示八个节点，即“linux00”至“linux07”。当指定单个节点名称时，对于具有大量节点的系统，可以显着提高命令的性能.

**--noconvert**

请勿转换原始类型的单位（例如2048M不会转换为2G）.

**-N, --Node**

以面向节点的格式打印信息，每个节点和分区一行。 也就是说，如果一个节点属于多个分区，则每个节点-分区对将显示一行。 如果还指定了--partition，则此分区中每个节点仅显示一行。 默认设置是以面向分区的格式打印信息。 如果指定了--format选项，则将忽略此设置.

**-o <output\_format>, --format=<output\_format>**

使用sinfo格式字符串指定要显示的信息。 如果命令是在联合集群环境中执行的，并且要显示有关多个集群的信息，并且使用了-h，--noheader选项，那么集群名称将在下面显示的默认输出格式之前显示。 使用各种选项运行时，sinfo使用的格式字符串是

default

"%#P %.5a %.10l %.6D %.6t %N"

--summarize

"%#P %.5a %.10l %.16F %N"

--long

"%#P %.5a %.10l %.10s %.4r %.8h %.10g %.6D %.11T %N"

--Node

"%#N %.6D %#P %6t"

--long --Node

"%#N %.6D %#P %.11T %.4c %.8z %.6m %.8d %.6w %.8f %20E"

--list-reasons

"%20E %9u %19H %N"

--long --list-reasons

"%20E %12U %19H %6t %N"

**-p <partition>, --partition=<partition>**

仅打印有关指定分区的信息。 多个分区之间用逗号分隔.

**-r, --responding**

如果设置，则仅报告响应节点的状态信息.

**-R, --list-reasons**

列出节点处于关闭，耗尽，失败或失败状态的原因。 当节点处于这些状态时，Slurm支持管理员可选地包含“原因”字符串。 此选项将显示原因字段的前20个字符以及该节点的列表，这些原因默认情况下为所有处于关闭，耗尽，耗尽或故障状态的节点。 该选项可以与其他节点过滤选项（例如-r，-d，-t，-n）一起使用，但是，这些选项的组合会导致未关闭或耗尽或失败的节点列表不会产生任何结果。 输出。 与-l一起使用时，输出还包括当前节点状态.

**-s, --summarize**

仅列出分区状态摘要，没有节点状态详细信息。 如果指定了--format选项，则将忽略此设置.

**-S <sort\_list>, --sort=<sort\_list>**

报告记录顺序的说明。 这使用与<output\_format>相同的字段规范。可以通过列出多个用逗号分隔的排序字段来执行多种排序。字段规范的前面可以带有“ +”或“-”，分别用于升序（默认）和降序。分区字段规范“P”之前可以有一个“#”，以报告分区的顺序与在Slurm的配置文件slurm.conf中出现的顺序相同。例如，“+P,-m”的排序值要求以增加分区名称的顺序打印记录，并通过减小内存大小在分区内打印记录。sort的默认值为“#P,-t”（按配置顺序排序的分区，然后降低节点状态）。如果选择了--Node选项，则默认排序值为“N”（增加节点名称）.

**-t <states> , --states=<states>**

列出仅具有给定状态的节点。 多个状态可以用逗号分隔，并且比较不区分大小写。 可能的值包括（不区分大小写）：ALLOC，ALLOCATED，COMP，COMPLETING，DOWN，DRAIN（用于处于DRAINING或DRAIN状态的节点），DRAINED，DRAINING，FAIL，FUTURE，FUTR，IDLE，MAINT，MIX，MIXED，NO\_RESPOND，NPC ，PERFCTRS，POWER\_DOWN，POWERING\_DOWN，POWER\_UP，RESV，SERVED，UNK和UNKNOWN。 默认情况下，将报告处于指定状态的节点是否响应。 --dead和--responding选项可用于通过响应标志过滤节点。

**-T, --reservation**

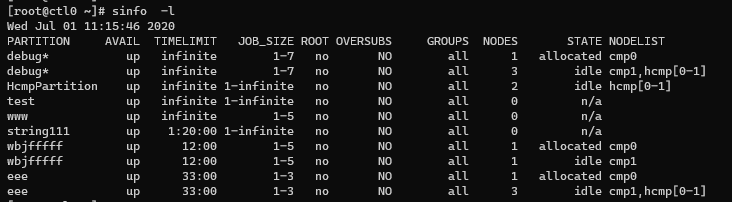
只显示有关Slurm保留的信息。

**输出字段说明**

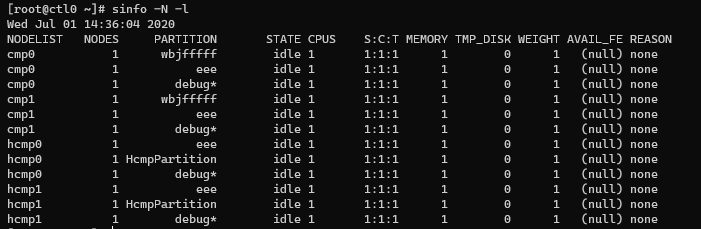
|  |  |
| --- | --- |
| **命令** | **说明** |
| AVAIL | 分区状态. 状态可以是 up, down, drain, or inact (for INACTIVE) |
| CPUS | 每个节点上的CPU（处理器）数 |
| S:C:T | 节点的sockets (S), cores (C), and threads (T) 总数 |
| SOCKETS | 节点的socket总数 |
| CORES | 节点的核总数 |
| THREADS | 节点的线程总数 |
| GROUPS | 此分区中的资源分配仅限于命名组。all表示所有组都可以使用此分区 |
| JOB\_SIZE | 可以分配给任何用户作业的最小和最大节点数。 单个数字表示最小和最大节点数相同。 infinite用于标识没有最大节点数的分区 |
| TIMELIMIT | 任何用户作业的最大时间限制，以days-hours:minutes:seconds为单位。infinite用于标识没有作业时间限制的分区 |
| MEMORY | 这些节点上的实际内存大小（以兆字节为单位） |
| NODELIST | 与此配置/分区关联的节点名称 |
| NODES | 具有此特定配置的节点数 |
| NODES(A/I) | 按节点状态以“可用/空闲”形式表示具有此特定配置的节点数 |
| NODES(A/I/O/T) | 按节点状态以“可用/空闲/其他/总数”形式表示具有此特定配置的节点数 |
| PARTITION | 分区名称。 请注意，后缀“\*”表示默认分区 |
| PORT | slurmd在节点上使用的本地TCP端口 |
| ROOT | 是否在此分区中分配资源的能力仅限于root用户，是或否 |
| OVERSUBSCRIBE | 将在此分区中分配资源的作业将超额分配那些计算资源（即CPU）。 NO：表示资源永远不会超额预定。 Exclusive表示整个节点都专用于作业（与srun --exclusive选项等效，甚至可以使用select / cons\_res管理单个处理器）。 强制表示始终有资源可供超额使用。  YES：表示根据作业的资源分配，资源可能超额预定 |
| STATE | 节点的状态。 可能的状态包括: allocated, completing, down, drained, draining, fail, failing, future, idle, maint, mixed, perfctrs, power\_down, power\_up, reserved, and unknown，以及其缩写形式: alloc, comp, down, drain, drng, fail, failg, futr, idle, maint, mix, npc, pow\_dn, pow\_up, resv, and unk. 后缀“\*”标识当前未响应的节点 |
| TMP\_DISK | 这些节点上的临时磁盘空间大小（以兆字节为单位） |

**示例**

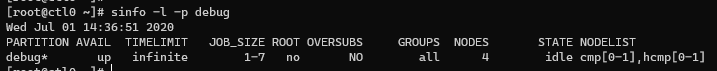
sinfo默认打印信息包含分区、节点概览信息；



以节点维度查询



以分区给度查询



### squeue

sque用于查看Slurm调度队列中管理的作业和作业步骤信息

**指令格式**

squeue [OPTIONS…]

**常用参数**

**-A <account\_list>, --account=<account\_list>**

指定要查看的作业的帐户。接受以逗号分隔的帐户名列表.

**-a, --all**

显示所有分区中有关作业和作业步骤的信息.

**-r, --array**

每行显示一个作业数组元素. 如果没有此选项，将优化显示以用于作业数组（挂起的作业数组元素将与使用正则表达式打印的数组索引值组合在一行输出中）

**--array-unique**

每行显示一个唯一的挂起作业数组元素。如果没有此选项，挂起的作业数组元素将分组到主阵列作业中以优化显示。这也可以用环境变量SQUEUE\_ARRAY\_UNIQUE设置.

**-h, --noheader**

不打印表头.

**--help**

显示帮助信息.

**--hide**

不要在所有分区中显示有关作业和作业步骤的信息。默认情况下，将不显示有关配置为隐藏分区或用户组不可用分区的信息（即，这是默认行为）.

**-i <seconds>, --iterate=<seconds>**

以指定的间隔（秒）重复收集和报告请求的信息。默认情况下，打印带有标题的时间戳.

**-j <job\_id\_list>, --jobs=<job\_id\_list>**

请求显示以逗号分隔的作业ID列表。默认为所有作业。--jobs=<job\_id\_list>选项可以与--steps选项一起使用，以打印有关特定作业的步骤信息。注意：如果提供了作业ID列表，则即使作业位于隐藏分区上，也会显示它们。由于此选项的参数是可选的，为了正确分析，单字母选项后面必须紧跟值，并且不能在它们之间包含空格。例如“-j1008”而不是“-j 1008”。作业ID格式为“job\_id[\_array\_id]”。当指定单个作业ID时，对于具有大量作业的系统，可以测量地提高该命令的性能。默认情况下，此字段大小将限制为64字节。使用环境变量SLURM\_BITSTR\_LEN指定更大的字段大小.

**--local**

仅显示此群集本地的作业。忽略此联盟中的其他群集（如果有）. Overrides --federation.

**-l, --long**

根据指定的任何约束，报告选定作业或作业步骤的更多可用信息.

**-L, --licenses=<license\_list>**

请求作业请求或使用一个或多个命名许可证。许可证列表由以逗号分隔的许可证名称列表组成.

**--me**

Equivalent to --user=<my username>.

**-M, --clusters=<string>**

要向其发出命令的群集。多个群集名称可以用逗号分隔。值“all”将查询以在所有群集上运行。此选项隐式设置--local选项.

**-n, --name=<name\_list>**

请求具有指定名称之一的作业或作业步骤。该列表由以逗号分隔的作业名称列表组成.

**--noconvert**

不要转换原始类型的单位（例如2048M不会转换为2G）.

**-o <output\_format>, --format=<output\_format>**

指定要显示的信息及其大小和位置（右对齐或左对齐）。另请参见下面描述的-O<output\_format>，--format=<output\_format>选项（它支持较少的格式化灵活性，但支持对所有字段的访问）。如果该命令在联邦集群环境中执行，并且要显示有关多个集群的信息，并且使用-h，--noheader选项，那么集群名称将显示在下面显示的默认输出格式之前.

各种选项的默认格式是:

default

"%.18i %.9P %.8j %.8u %.2t %.10M %.6D %R"

-l, --long

"%.18i %.9P %.8j %.8u %.8T %.10M %.9l %.6D %R"

-s, --steps

"%.15i %.8j %.9P %.8u %.9M %N"

**-O <output\_format>, --Format=<output\_format>**

**-p <part\_list>, --partition=<part\_list>**

指定要查看的作业或步骤的分区。接受分区名称的逗号分隔列表.

**-P, --priority**

对于提交到多个分区的挂起作业，请为每个分区列出一次作业。此外，如果作业按优先级排序，请同时考虑分区和作业优先级。此选项可用于按Slurm计划时考虑的相同顺序生成挂起作业的列表，并提供适当的附加选项（例如“-sort=-p，i--states=PD”）.

**-q <qos\_list>, --qos=<qos\_list>**

指定要查看的作业或步骤的qos。接受以逗号分隔的qos列表.

**-R, --reservation=reservation\_name**

指定要查看的作业的预约.

**-s, --steps**

指定要查看的作业步骤。 该标志指示后面要用逗号分隔的要查看的工作步骤列表，而没有等号。 作业步骤格式为“job\_id [\_array\_id] .step\_id”。 默认为所有作业步骤。 由于此选项的参数是可选的，因此为了正确解析，单字母选项必须紧跟其后的值，并且它们之间不应包含空格。 例如，“-s1008.0”而不是“-s 1008.0”.

**--sibling**

显示联合群集上的所有同级作业. Implies --federation

**-S <sort\_list>, --sort=<sort\_list>**

报告记录顺序的说明。 这使用与<output\_format>相同的字段规范。 长格式选项“集群”也可用于按群集名称对作业或作业步骤进行排序（例如，联合作业）。 可以通过列出多个用逗号分隔的排序字段来执行多种排序。 字段规范的前面可以带有“ +”或“-”，分别用于升序（默认）和降序。 例如，排序值“P,U”将按分区名称然后按用户ID对记录进行排序。 作业排序的默认值为“P,t,-p”（先增加分区名称，然后再在给定分区内通过增加作业状态然后降低优先级）。 作业步骤的排序的默认值为“P,i”（然后通过增加步骤ID在给定分区中先增加分区名称）。

**--start**

报告预期的开始时间和要为开始的作业分配的资源，以增加开始时间的顺序。 这等效于以下选项：--format =“%.18i %.9P %.8j %.8u %.2t %.19S %.6D %20Y %R”，-sort = S和--states = 等待中。 通过将--start选项与其他选项值组合（例如，使用不同的输出格式），可以根据需要显式更改这些选项中的任何一个。 仅当Slurm配置为使用回填计划插件时，挂起作业的预期开始时间才可用.

**-t <state\_list>, --states=<state\_list>**

指定要查看的作业状态。 接受以逗号分隔的状态名称或“全部”列表。 如果指定“全部”，则将报告所有状态的作业。 如果未指定任何状态，则将报告挂起，正在运行和完成的作业。 有关有效状态的列表，请参见下面的“作业状态代码”部分。 扩展形式和紧凑形式均有效。 请注意，提供的<state\_list>不区分大小写（“ pd”和“ PD”等效）.

**-u <user\_list>, --user=<user\_list>**

从逗号分隔的用户列表中请求作业或作业步骤。 该列表可以包含用户名或用户ID号。 当指定一个用户时，对于具有大量作业的系统，可以显着提高命令的性能.

**--usage**

打印简短的帮助消息，列出排队选项.

**-v, --verbose**

报告squeues动作的详细信息.

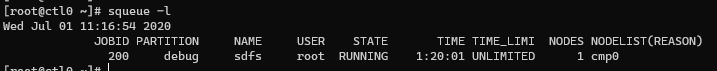
**-V , --version**

Print version information and exit.

**-w <hostlist>, --nodelist=<hostlist>**

仅报告分配给指定节点或节点列表的作业。 如果它们不同，则可以是slurm.conf配置文件中定义的NodeName或NodeHostname。 localhost的node\_name映射到当前主机名.

**示例**



输出字段说明：

JOBID：作业ID

PARTITION：队列名称

NAME：作业名

USER：作业所属用户

ST： 作业状态（作业状态参见4.2）

TIME：作业已运行时间

NODES：作业占用节点数

NODELIST（REASON）：作业运行的结点列表（不运行作业的原因）（参见4.3）

### sreport

sreport用于生成保存到Slurm数据库slurmdbd的Slurm作业的作业使用情况和集群利用率报告。

**指令格式**

sreport [OPTIONS…] [COMMAND…]

**[OPTIONS]：**

**-a, --all\_clusters**

使用所有群集，而不是仅使用执行命令的群集.

**--federation**

如果联盟的成员之一，则为其生成报告.

**-h, --help**

帮助信息.

**--local**

即使是联盟的一部分，也要为本地集群生成报告. Overrides --federation

**-M, --cluster**

要为其生成报告的集群。 默认值为本地集群，除非本地集群当前是联盟的一部分，在这种情况下，将为当前联盟中的所有集群生成报告。 如果联盟中包含的群集随时间变化，请使用此选项来标识要包括在报告中的群集. Implies --local.

**-n, --noheader**

不会向输出添加标题。默认操作是显示标题.

**-p, --parsable**

输出将以“|”分隔，结尾带“|”

**-P, --parsable2**

输出将以“|”分隔，结尾不带“|”

**-Q, --quiet**

不打印警告或参考消息，仅显示错误消息.

**-t <format>**

指定输出时间格式。时间格式选项不区分大小写，可以缩写。默认格式为分钟。下面的时间命令部分列出了支持的时间格式选项.

**-T, --tres <tres\_names>**

要报告其值的可跟踪资源（TRES）。默认情况下会报告CPU资源使用情况（保留报告除外）。默认情况下，保留保留的所有TRES类型都将显示，除非使用此选项重写）。可以使用逗号分隔的列表将多个TRES名称分隔开，但作业报告除外，作业报告只能支持单个TRES名称，或者对所有tre使用“all”。

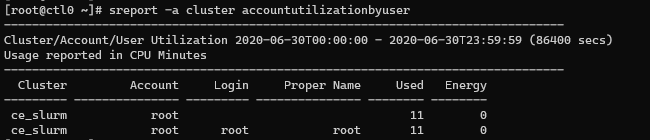
“报告的”计费树是根据每个节点的最大计费树乘以时间帧计算的。例如，如果一个节点是多个分区的一部分，并且每个分区定义了不同的TRESBillingWeights，那么该节点的计费TRES将是分区中最高的。如果未在节点的任何分区上定义TRESBillingWeights，则计费TRES将等于节点上的cpu数量。

不再以百分比格式或群集利用率报告TRES节点使用情况。请改用TRES CPU。使用节点的主要问题是在大多数配置中，多个作业能够在同一个节点上运行。这使得TRES节点会计在同一期间多次计算同一节点。在独占节点配置中，CPU记帐将返回与节点记帐相同的使用率。

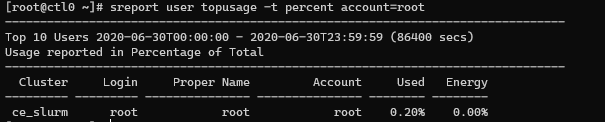
**[COMMAND]：**

|  |  |  |
| --- | --- | --- |
| cluster | AccountUtilizationByUser | 该报告将显示其在层次结构树中显示的帐户利用率。 默认情况下，从指定帐户或根帐户开始，此报告将列出基础用法以及每个级别的总和。 |
| UserUtilizationByAccount | 此报告将按使用情况按帐户显示用户，而不会按用户将多个帐户分组为一个帐户，而是将它们显示在单独的行中 |
| UserUtilizationByWckey | 此报告将按使用率按wckey显示用户，而不会按用户将多个wckey分组为一个，而是将它们显示在单独的行中 |
| Utilization | 该报告将显示选定群集的总使用情况，包括Allocated, Down, Planned Down, Idle, and Reserved time |
| WCKeyUtilizationByUser | 该报告将显示每个集群上每个用户的wckey利用率，按WCKey名称排序 |
| job | SizesByAccount | 该报告将显示“ grouping =”选项指定的作业范围所用的时间。 |
| SizesByAccountAndWcKey | 此报告与SizesByAccount非常相似，不同之处在于每个帐户都是与wckey成对的，因此标识符是account：wckey而不是仅仅是account，因此很可能会列出多个帐户，具体取决于所使用的wckey的数量。 |
| SizesByWckey | 此报告将显示由“ grouping =”选项指定的作业范围的每个wckey的时间。 |
| reservation | Utilization | 该报告将显示系统上预订的总使用量 |
| user | TopUsage | 显示集群上的主要用户。 使用分组选项将帐户分组在一起。 默认值是每个用户帐户组合都有不同的行 |

**示例**



报告指定默认一天内root帐户的最大使用量百分比，也可通过start 和 end 指令时间段：



### srun

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。

**指令格式**

srun [OPTIONS...] executable [args...]

**参数说明**

参见 3.3.1

### sshare

用于列出与群集关联的共享的工具。

sshare用于查看Slurm共享信息。此命令仅在与priority/multifactor插件一起运行时才有效。sshare信息来自一个数据库，该数据库的接口由slurmdbd（Slurm database daemon）提供，slurmdbd从slurmctld读入并用于处理给定关联可用的共享。sshare为每个关联提供Account(帐户)、User(用户)、RawShares(原始共享)、NormalizedShares(标准化共享)、RawUsage(原始使用)、NormalizedUsage(标准化使用)、EffectiveUsage(有效使用)、FairShare(公平共享因子)、GrpTRESMins、Partitions分区和TRESRunMins(当前累计运行TRES分钟)的Slurm共享信息

**说明参数**

**-A, --accounts=**

显示特定账户的信息（逗号分隔的列表）。

**-a, --all**

显示所有用户的信息。

**-l, --long**

长名单--包括标准化使用信息。

**-M, --clusters=<string>**

要向其发出命令的群集。请注意，SlurmDBD 必须启动，这个选项才能正常工作。

**-m, --partition**

如果系统中存在基于关联的分区，则打印它们的名称。

**-n, --noheader**

不会向输出添加标题。默认操作是显示标题.

**-o, --format=**

逗号分隔的字段列表（使用"--helpformat "查看可用字段列表）。

**-p, --parsable**

输出将以“|”分隔，结尾带“|”

**-P, --parsable2**

输出将以“|”分隔，结尾不带“|”

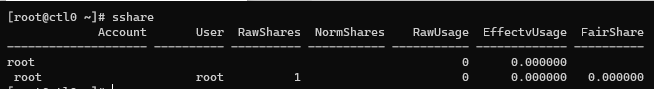
**-u, --users=**

显示特定用户的信息（逗号分隔的列表）。

**-U, --Users**

如果指定只打印用户信息，则不显示父辈和祖先。

**示例**



### sstat

显示正在运行的作业/步骤的各种状态信息

sstat命令显示用于分析的作业状态信息。sstat命令显示有关CPU、任务、节点、驻留集大小（RSS）和虚拟内存（VM）的信息。可以使用--fields=选项定制输出，以指定要显示的字段。

对于root用户，SSTAT命令显示在系统上运行的任何作业的作业状态数据。

对于非root用户，sstat输出仅限于用户的作业。

【注意】：sstat命令要求安装并运行jobacct\_gather插件。

【注意】：Cray ALPS不支持sstat命令。

**指令格式**

**sstat [OPTIONS...] –j <job(.stepId)>**

**参数说明**

**-a, --allsteps**

未指定步骤时打印给定作业的所有步骤.

**-e, --helpformat**

打印可使用“--format”选项指定的字段列表.

**-h, --help**

显示常规帮助消息.

**-i, --pidformat**

列出每个作业步骤运行的PID的预定义格式。 (JobId,Nodes,Pids)

**-j, --jobs**

格式为<job（.step）>。统计此作业步骤或以逗号分隔的作业步骤列表。此选项是必需的。如果未指定，步骤部分将默认为最低步骤运行，除非设置了--all steps标志，否则未指定步骤将导致显示所有运行步骤。

注意：步骤id为“batch”将显示有关批处理步骤的信息。

注意： 步骤id为“extern”将显示有关extern步骤的信息。此步骤仅在使用PrologFlags=contain时可用

**-n, --noheader**

不会向输出添加标题。默认操作是显示标题.

**--noconvert**

不要转换原始类型的单位（例如2048M不会转换为2G）.

**-o, --format, --fields**

以逗号分隔的字段列表。（对可用字段列表使用“--helpformat”）。

注意：当使用format选项列出不同的字段时，您可以在后面输入一个%NUMBER来指定应该打印多少个字符。

例如：format=name%30将打印30个字符的字段名右对齐。A-30将打印30个左对齐字符.

**-p, --parsable**

输出将以“|”分隔，结尾带“|”

**-P, --parsable2**

输出将以“|”分隔，结尾不带“|”

**示例**

> sstat --format=AveCPU,AvePages,AveRSS,AveVMSize,JobID -j 11

25:02.000 0K 1.37M 5.93M 9.0

> sstat -p --format=AveCPU,AvePages,AveRSS,AveVMSize,JobID -j 11

25:02.000|0K|1.37M|5.93M|9.0|

### strigger

**指令格式**

**strigger --set [OPTIONS...]  
strigger --get [OPTIONS...]  
strigger --clear [OPTIONS...]**

**--set**

根据提供的选项注册事件触发器。注意：事件只触发一次。必须为将来要处理的同一类型的事件设置新的事件触发器。只有当命令由用户SlurmUser运行时，才能设置触发器，除非SlurmUser配置为root用户。

**--get**

显示已注册的事件触发器。选项可用于筛选目的

**--clear**

清除或删除以前定义的事件触发器。必须指定--id、-jobid或--user选项来标识要清除的触发器。只有root用户或触发器的创建者才能删除触发器。

**示例**

1. 当主slurmctld失败时，执行/usr/sbin/primary\_slurmctld\_failure程序，发邮件通知。指令设置触了发器，一起触发器是一次性生效，所以需要在脚本中再进行一次设置。

> strigger --set --primary\_slurmctld\_failure \

--program=/usr/sbin/primary\_slurmctld\_failure

primary\_slurmctld\_failure脚本内容：

#!/bin/bash

# Submit trigger for next primary slurmctld failure event

strigger --set --primary\_slurmctld\_failure \

--program=/usr/sbin/primary\_slurmctld\_failure

# Notify the administrator of the failure using by e-mail

/bin/mail [slurm\_admin@site.com](mailto:slurm_admin@site.com) -s Primary\_SLURMCTLD\_FAILURE

# 状态监控

在Slurm系统中可以通过指令查询节点、分区、作业、作业步的运行状态。节点和分区的实时状态可通过sinfo 指令进行查询，作业和作业步的状态可通过squeue进行查询。

以下列出了相关的节点状态、作业以及作业异常状态原因。

## 节点状态

|  |  |
| --- | --- |
| **状态** | **说明** |
| \* | 该节点目前没有响应，也不会分配任何新工作。 如果该节点保持无响应，则它将处于DOWN状态（在COMPLETING，DRAINED，DRAINING，FAIL，FAILING节点的情况下除外）。 |
| ~ | 该节点当前处于节能模式（通常以降低的频率运行）. |
| # | 该节点目前正在加电或配置中. |
| % | 该节点目前正在关机. |
| $ | 该节点当前处于保留状态，其标志值为“ maintenance”。 |
| @ | 节点正在等待重启. |
| ALLOCATED | 该节点已分配给一个或多个作业. |
| ALLOCATED+ | 该节点已分配给一个或多个活动作业，并且正在完成一个或多个作业. |
| COMPLETING | 与此节点关联的所有作业都在完成中。 |
| DOWN | 该节点不可用。 如果发生某些故障，Slurm可以自动将节点置于此状态。系统管理员还可以明确地将节点置于此状态。如果节点恢复正常运行，Slurm可以自动将其恢复服务。 |
| DRAINED | 根据系统管理员的请求，该节点不可用。. |
| DRAINING | 该节点当前正在执行作业，但不会分配给其他作业。 当节点上的最后一个作业完成时，节点状态将更改为DRAINED状态。 节点根据系统管理员请求进入此状态。 |
| FAIL | 预计该节点很快会发生故障，并且无法根据系统管理员的请求使用该节点。 |
| FAILING | 该节点当前正在执行作业，但是预计很快就会失败，并且无法根据系统管理员的请求使用。 |
| FUTURE | 该节点当前尚未完全配置，但预计将在不确定的将来某个时候可用. |
| IDLE | 该节点未分配给任何作业，可供使用. |
| MAINT | 该节点当前处于保留状态，其标志值为“ maintenance”。 |
| REBOOT | 该节点当前计划重新引导。 |
| MIXED | 该节点分配了一些CPU，而其他资源则为IDLE。 |
| PERFCTRS (NPC) | 与该节点关联的网络性能计数器正在使用中，从而使该节点不可用于任何其他作业 |
| POWER\_DOWN | 该节点当前已关闭电源，无法运行任何作业。 |
| POWERING\_DOWN | 该节点当前正在关闭电源，无法运行任何作业。 |
| POWER\_UP | 该节点当前正在加电。 |
| RESERVED | 该节点处于高级保留状态，通常不可用。 |
| UNKNOWN | Slurm控制器刚刚启动，尚未确定节点的状态 |

## 作业状态

squeue 指令可以查看当前的作业状态，

|  |  |  |
| --- | --- | --- |
| **状态代码** | **状态** | **说明** |
| BF | BOOT\_FAIL | 作业由于启动失败而终止，通常是由于硬件故障(例如，无法启动节点或块而无法重新排队该任务) |
| CA | CANCELLED | 用户或系统管理员已明确取消作业。 作业可能已经启动，也可能尚未启动 |
| CD | COMPLETED | 作业已终止所有节点上的所有进程，退出代码为零 |
| CF | CONFIGURING | 已为作业分配了资源，但正在等待它们准备就绪（例如启动） |
| CG | COMPLETING | 作业正在完成中。 一些节点上的某些进程可能仍处于活动状态 |
| DL | DEADLINE | 作业正在完成中。 一些节点上的某些进程可能仍处于活动状态。 |
| F | FAILED | 作业在截止日期前终止 |
| NF | NODE\_FAIL | 作业由于一个或多个分配的节点故障而终止 |
| OOM | OUT\_OF\_MEMORY | 作业遇到内存不足错误。 |
| PD | PENDING | 作业正在等待资源分配 |
| PR | PREEMPTED | 作业由于抢占而终止 |
| R | RUNNING | 作业当前有分配 |
| RD | RESV\_DEL\_HOLD | 作业被保持 |
| RF | REQUEUE\_FED | 作业正在由联合群集重新排队 |
| RH | REQUEUE\_HOLD | 保持的工作正在重新排队 |
| RQ | REQUEUED | 正在完成作业 |
| RS | RESIZING | 作业即将更改大小 |
| RV | REVOKED | 由于其他集群启动了作业，因此已从集群中删除了同级节点 |
| SI | SIGNALING | 正在发出作业通知 |
| SE | SPECIAL\_EXIT | 该工作以特殊状态重新排队。 如果作业已终止并具有特定的退出值，则用户通常可以在EpilogSlurmctld中设置此状态 |
| SO | STAGE\_OUT | 作业正在暂存文件 |
| ST | STOPPED | 作业已分配，但执行已通过SIGSTOP信号停止。 该工作已保留了CPUS |
| S | SUSPENDED | 作业已分配，但执行已暂停并且已为其他作业释放了CPU |
| TO | TIMEOUT | 作业在达到其时间限制时终止 |

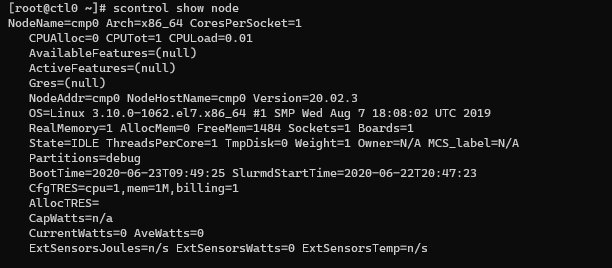
## 作业异常原因

|  |  |
| --- | --- |
| **异常代码** | **说明** |
| AssociationJobLimit | 作业的关联已达到其最大作业计数 |
| AssociationResourceLimit | 作业的关联已达到一定的资源极限 |
| AssociationTimeLimit | 该作业的关联已达到其时间限制 |
| BadConstraints | 作业的约束无法得到满足 |
| BeginTime | 尚未达到作业的最早开始时间 |
| Cleaning | 该作业正在重新排队，并且仍从先前的执行中清除 |
| Dependency | 这项作业正在等待相关工作完成 |
| FrontEndDown | 没有前端节点可用于执行此作业 |
| InactiveLimit | 作业已达到系统非活动限制 |
| InvalidAccount | 作业的帐户无效 |
| InvalidQOS | 作业的QOS无效 |
| JobHeldAdmin | 该作业由被系统管理员保持 |
| JobHeldUser | 该作业由被用户保持 |
| JobLaunchFailure | 该作业无法启动。 这可能是由于文件系统问题，程序名称无效等引起的。 |
| Licenses | 作业正在等待许可证。 |
| NodeDown | 作业所需的节点已关闭。 |
| NonZeroExitCode | 作业以非零退出代码终止。 |
| PartitionDown | 该作业所需的分区处于DOWN状态。 |
| PartitionInactive | 该作业所需的分区处于“非活动”状态，无法启动作业。 |
| PartitionNodeLimit | 该作业所需的节点数超出了其分区的当前限制。 也可以指示所需的节点为DOWN或DRAINED。 |
| PartitionTimeLimit | 作业的时间限制超出了其分区的当前时间限制。 |
| Priority | 为此分区或高级预留存在一个或多个更高优先级的作业。 |
| Prolog | 其PrologSlurmctld程序仍在运行。 |
| QOSJobLimit | 作业的QOS已达到最大作业数。 |
| QOSResourceLimit | 作业的QOS已达到一定的资源限制。 |
| QOSTimeLimit | 作业的QOS已达到其时间限制。 |
| ReqNodeNotAvail | 该作业明确要求的某些节点当前不可用。 该节点当前可能正在使用中，为其他作业保留，处于高级保留状态，“关闭”，“已耗尽”或没有响应。 DOWN，DRAINED或无响应的节点将被标识为“ UnavailableNodes”，作为工作的“原因”字段的一部分。 此类节点通常需要系统管理员的干预才能使用。 |
| Reservation | 保留作业正在等待其高级保留可用。 |
| Resources | 作业正在等待资源可用。 |
| SystemFailure | Slurm系统，文件系统，网络等发生故障 |
| TimeLimit | 这项工作已耗尽了时间。 |
| QOSUsageThreshold | 违反了所需的QOS阈值。 |
| WaitingForScheduling | 尚未为此工作设置任何原因。 等待调度程序确定适当的原因。 |

# 资源管理

## 节点：Node

即指计算节点；包含处理器、内存、磁盘空间等资源；具有空闲、分配、故障等状态；使用节点名字标识；通过scontrol 进行管理。节点信息展示如下：



### 主要输出项

|  |  |
| --- | --- |
| **输出项** | **说明** |
| NodeName | 节点名 |
| Arch | 系统架构 |
| CoresPerSocket | 每个Socket的核数 |
| CPUAlloc | 分配给的CPU核数 |
| CPUErr | 出错的CPU核数 |
| CPUTot | 总CPU核数 |
| CPULoad | CPU负载 |
| AvailableFeatures | 可用特性 |
| ActiveFeatures | 激活的特性 |
| Gres | 通用资源，如Gres=gpu:v100:2指明了有两块V100 GPU |
| NodeAddr | 节点IP地址 |
| NodeHostName | 节点名 |
| Version | Slurm版本 |
| OS | 操作系统 |
| RealMemory | 实际物理内存，单位MB |
| AllocMem | 已分配内存，单位MB |
| FreeMem | 可用内存，单位MB |
| Sockets | CPU颗数 |
| Boards | 主板数 |
| State | 状态 |
| ThreadsPerCore | 每颗CPU核线程数 |
| TmpDisk | 临时存盘硬盘大小 |
| Weight | 权重 |
| BootTime | 开机时间 |
| SlurmdStartTime | Slurmd守护进程启动时间 |

### 查看节点详细信息

查看所有节点信息

scontrol show node

根据节点名称查询指定节点的详细信息，节点名称作为唯一标识

scontrol show node [node\_name] 或 scontrol show NodeName=[node\_name]

### 修改节点信息

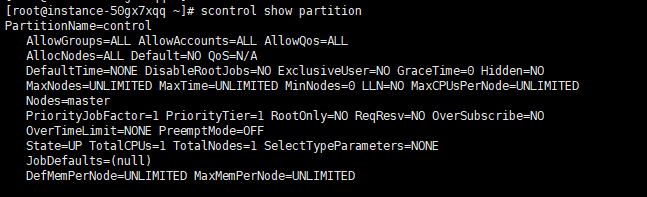
scontrol update nodeName=[node\_name] [params]=[value]

可更新节点参数：

ActiveFeatures、AvailableFeatures、CpuBind、Gres、NodeAddr、NodeHostname、Reason、State、Weight

## 分区/队列：Partition

节点的逻辑分组；提供一种管理机制，可设置资源限制、访问权限、优先级等；分区可重叠，提供类似于队列的功能；使用分区名字标识。分区信息展示如下：



### 主要输出项

|  |  |
| --- | --- |
| **输出项** | **说明** |
| PartitionName | 队列名 |
| AllowGroups | 允许的用户组 |
| AllowAccounts | 允许的账户 |
| AllowQos | 允许的QoS |
| AllocNodes | 允许的节点 |
| Default | 是否为默认队列 |
| QoS | 服务质量 |
| DefaultTime | 默认时间 |
| DisableRootJobs | 是否禁止root用户提交作业 |
| ExclusiveUser | 排除的用户 |
| GraceTime | 抢占的款显时间，单位秒 |
| Hidden | 是否为隐藏队列 |
| MaxNodes | 最大节点数 |
| MaxTime | 最大运行时间 |
| MinNodes | 最小节点数 |
| LLN | 是否按照最小负载节点调度 |
| MaxCPUsPerNode | 每个节点的最大CPU颗数 |
| Nodes | 节点名 |
| PriorityJobFactor | 作业因子优先级 |
| PriorityTier | 调度优先级 |
| RootOnly | 是否只允许Root |
| ReqResv | 要求预留的资源 |
| OverSubscribe | 是否允许超用 |
| PreemptMode | 是否为抢占模式 |
| State | 状态  – UP：可用，作业可以提交到此队列，并将运行。  – DOWN：作业可以提交到此队列，但作业也许不会获得分配开始运行。已运行的作业还将继续运行。  – DRAIN：不接受新作业，已接受的作业可以被运行。  – INACTIVE：不接受新作业，已接受的作业未开始运行的也不运行。 |
| TotalCPUs | 总CPU核数 |
| TotalNodes | 总节点数 |
| SelectTypeParameters | 资源选择类型参数 |
| 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 |

### 查看队列详细信息

查看所有队列信息

scontrol show partition

根据名称查看指定队列详情，通过队列名作为唯一标识

scontrol show partition <partition\_name> 或者 scontrol show PartitionName=<partition\_name>，

### 新建队列

scontrol create PartitionName=<partition\_name> [param]=[value]

新建可用参数：

AllowGroups、AllocNodes、Alternate、CpuBind、Default、DefaultTime、DefMemPerCPU、DefMemPerNode、DisableRootJobs、GraceTime、Hidden、JobDefaults、MaxMemPerCPU、MaxMemPerCNode、MaxNodes、MaxTime、MinNodes、Nodes、OverTimeLimit、OverSubscribe、PreemptMode、Priority、PriorityJobFactor、PriorityTier、QOS、RootOnly、ReqResv、Shared、State、TRESBillingWeights

### 更新队列

scontrol update PartitionName=<partition\_name> [param]=[value]

可更新的参数：

AllowGroups、AllocNodes、Alternate、CpuBind、Default、DefaultTime、DefMemPerCPU、DefMemPerNode、DisableRootJobs、GraceTime、Hidden、JobDefaults、MaxMemPerCPU、MaxMemPerCNode、MaxNodes、MaxTime、MinNodes、Nodes、OverTimeLimit、OverSubscribe、PreemptMode、Priority、PriorityJobFactor、PriorityTier、QOS、RootOnly、ReqResv、Shared、State、TRESBillingWeights

### 删除队列

scontrol delete PartitionName=<partition\_name>

# 资源限制

在slurm中可以对资源采取复杂的限制策略。具体来说，slurm采用多层次的限制：

1、分区的QoS限制

2、任务的QoS限制

3、用户的关联关系

4、账户的关联关系

5、Root/集群的关联关系

6、分区的限制

7、无限制

按照从上到下的顺序、先指定的限制会生效。

## 关联关系

记账和任务调度策略都是基于关联关系association的，关联关系是一个（集群，账户，用户，分区）四元组。

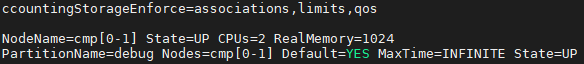
开启记账功能需要配置AccountingStorageType=accounting\_storage/mysql，开启任务调度策略限制需要配置AccountingStorageEnforce=associations,limits,qos,safe,wckeys。

请根据需要选择参数：

|  |  |
| --- | --- |
| **参数** | **说明** |
| associations | 限制没有association的用户提交作业 |
| limits | 强制association生效 |
| qos | 任务提交的时候需要指定qos |
| safe | 只让能在设置的时间内完成的任务运行 |
| wckeys | 只让有wckey权限的用户运行任务 |

设置了AccountingStorageEnforce需要再slurm.conf中指定每个计算节点的资源数目。样例如下：

配置slurm.conf样例



## 设置服务质量

可以给每个提交的任务指定服务质量QoS，下面是创建QoS的操作步骤：

sacctmgr add qos qos\_large

sacctmgr add qos qos\_small

这里创建了两个，qos\_large适用于占用资源较多的长时任务，优先级较低，qos\_small适用于占用资源较少的短时任务，优先级较高。按照这个思路为它们指定资源数量和优先级：

sacctmgr modify qos qos\_large set priority=10

sacctmgr modify qos qos\_large set MaxTRESPerUser=cpu=128,mem=512 MaxJobsPerUser=20 MaxSubmitJobsPerUser=30

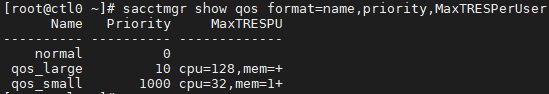
sacctmgr modify qos qos\_small set priority=1000

sacctmgr modify qos qos\_small set MaxTRESPerUser=cpu=32,mem=128 MaxJobsPerUser=2 MaxSubmitJobsPerUser=4

执行完上面的命令后，可以用下面的命令验证一下执行结果：

sacctmgr show qos format=name,priority,MaxTRESPerUser

命令截图



## 添加账号和用户

账号是slurm中跟踪和限制资源使用情况的，用户是Linux系统中的登陆用户。

用下面的命令添加账号：

sacctmgr add account name=account0 cluster=cluster0

添加用户到帐号并且给用户添加 qos：

sacctmgr add user name=user0 account=account0 cluster=cluster0 qos=qos\_large

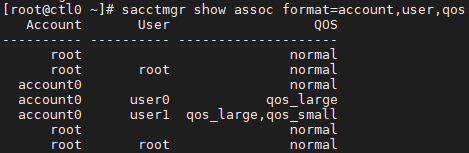
sacctmgr add user name=user1 account=account0 cluster=cluster0 qos=qos\_large,qos\_small

执行过程中命令行会提示是否创建Linux用户，请输入y确认。

执行完以后，可以用下面的命令确认执行结果：

sacctmgr show assoc format=account,user,qos

命令截图



可以用一下命令限制单个用户的资源数目：

sacctmgr modify user user9 set GrpTRES=cpu=1500,mem=200

之后可以用下面的命令取消资源数目限制：

sacctmgr modify user user9 set GrpTRES=cpu=-1,mem=-1

下面解释一下相关的参数：

1. 关联关系和QoS都适用的参数

|  |  |
| --- | --- |
| **参数** | **说明** |
| GrpTRES | 组TRES资源数目限制。达到使用限制后新任务必须排队等待其他任务释放资源 |
| GrpTRESMins | 组TRES分钟限制。达到限制后组内任务都会被杀死，并且不允许新的任务运行。 PriorityDecayHalfLife设置半衰期 |
| GrpTRESRunMins | 组TRES分钟运行限制。达到限制后停止运行新任务直到其他任务结束 |
| GrpJobs | 组同时运行的任务数目限制。达到数目限制后新任务必须排队等待先前的任务运行结束 |
| GrpSubmitJobs | 组提交任务数目限制。达到数目限制后新任务会被拒绝提交直到先前的任务运行结束 |
| GrpWall | 组挂钟时间限制。达到限制后新任务会被拒绝提交。PriorityDecayHalfLife设置半衰期 |
| MaxTRESMinsPerJob | 单个任务分钟限制。达到限制后，如果任务不是运行再safe模式下，会被杀死 |
| MaxTRESPerJob | 单个任务资源数目限制 |
| MaxTRESPerNode | 单个节点资源数目限制 |
| MaxWallDurationPerJob | 单个节点挂钟时间限制 |

1. 关联关系特定的参数：

|  |  |
| --- | --- |
| **参数** | **说明** |
| MaxJobs | 同时运行的任务数目限制。达到数目限制后新任务必须排队等待先前的任务运行结束 |
| MaxSubmitJobs | 提交任务数目限制。达到数目限制后新任务会被拒绝提交直到先前的任务运行结束 |
| QOS | 逗号分隔的QoS列表 |

1. QoS特定的参数

|  |  |
| --- | --- |
| **参数** | **说明** |
| MaxJobsPerAccount | 单个账户同时运行的任务数目限制 |
| MaxJobsPerUser | 单个用户同时运行的任务数目限制 |
| MaxSubmitJobsPerAccount | 单个账户提交任务（正在运行的任务加上挂起的任务）数目限制 |
| MaxSubmitJobsPerUser | 单个用户提交任务（正在运行的任务加上挂起的任务）数目限制 |
| MaxTRESPerAccount | 单个用户TRES资源数目限制 |
| MaxTRESPerUser | 逗号分隔的QoS列表 |
| MinTRESPerJob | 单个任务TRES资源最小数目限制 |

# 脚本和任务

可以用srun运行同步的命令，用sbatch运行脚本。本章节主要介绍sbatch的使用方法。

## 脚本样例

下面通过一个具体的脚本test.sh演示。

#!/bin/bash

#SBATCH --job-name="testJob"

#SBATCH --partition=debug

#SBATCH --qos=qos\_large

#SBATCH --nodes=2

#SBATCH --ntasks-per-node=2

#SBATCH --exclusive

#SBATCH --time=00:10:00

#SBATCH --output=job.%j.out

echo "current job: ${SLURM\_JOB\_ID}:${SLURM\_JOB\_NAME}"

echo " running on: ${SLURM\_JOB\_PARTITION}:${SLURM\_JOB\_NODELIST}"

echo " allocated resources: CPU ${SLURM\_CPUS\_ON\_NODE} / GPU ${SLURM\_GPUS\_PER\_NODE}"

sleep 60

脚本的前面可以包含多个以"#SBATCH"开头的指令行，直到第一行非注释非空白的行为止。常用的指令：

|  |  |
| --- | --- |
| **参数** | **说明** |
| --job-name | 指定任务名称 |
| --partition | 指定运行的分区 |
| --qos | 指定运行的QoS |
| --nodes | 指定需要的节点数 |
| --output | 指定输出文件名称模板，"%j"会被替换成当前任务的ID，"%x"会被替换成当前任务的名称 |
| MaxTRESPerUser | 逗号分隔的QoS列表 |
| MinTRESPerJob | 单个任务TRES资源最小数目限制 |

脚本运行过程中，slurm会设置一些环境变量。上面的脚本样例中使用的环境变量的含义：

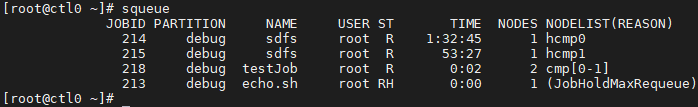
|  |  |
| --- | --- |
| **变量** | **含义** |
| SLURM\_JOB\_ID | 任务ID |
| SLURM\_JOB\_NAME | 任务名称 |
| SLURM\_JOB\_PARTITION | 运行的分区名称 |
| SLURM\_JOB\_NODELIST | 运行的节点列表 |
| SLURM\_CPUS\_ON\_NODE | 分配的CPU数目 |
| SLURM\_GPUS\_PER\_NODE | 分配的GPU数目 |

## 运行脚本

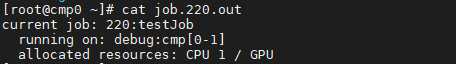
用sbatch命令运行上面的样例脚本。结果如下：



用squeue命令确认218号任务的运行情况。结果如下：



可以看到218号任务被分配到了cmp[0-1]两个节点运行，等任务运行完成以后，可以去cmp0节点下查看脚本输出，参考样例如下：



# A 如何获取帮助

## A.1 收集必要的故障信息

在进行故障处理前，需要收集必要的故障信息。

收集的信息包括：

* 客户详细名称、地址
* 联系人姓名、电话号码
* 故障发生的具体时间
* 故障现象的详细描述
* 设备类型及软件版本
* 故障后已采取的措施和结果
* 问题的级别及希望解决的时间

## A.2如何使用文档

浪潮电子信息产业股份有限公司提供全面的随设备发货的指导文档。指导文档能解决您在日常维护或故障处理过程中遇到的常见问题。为了更好的解决故障，在寻求浪潮技术支持前，建议充分使用指导文档。

## A.3获取技术支持

浪潮电子信息产业股份有限公司通过办事处、电话技术指导、远程支持及现场技术支持等方式向用户提供及时有效的技术支持。

浪潮电子信息产业股份有限公司技术支持体系包括：

* 客户服务中心：(+86)400-860-0011; (+86)800-860-0011
* 企业业务网站（https://www.inspur.com）

# B 术语和缩略语