slurm集群用户使用指南

发布日期 2020-10-12 by 余航

修改日期 2022-03-13 by 余航

主要说明`slurm`提交任务方法，如果是新手，请先查看 **服务器与集群入门指南**

> 注意！！！不要在登录的地方（master 节点）运行你的程序，请使用 slurm 提交！！！
> 注意！！！不要在登录的地方（master 节点）运行你的程序，请使用 slurm 提交！！！
> 注意！！！不要在登录的地方（master 节点）运行你的程序，请使用 slurm 提交！！！
> 注意！！！不要在登录的地方（master 节点）运行你的程序，请使用 slurm 提交！！！

我只列出来简单、常用的命令，以及一些自己遇到的坑，更深入的可以百度 slurm，一般用不到，如果遇到问题，请首先联系与自己课题相关的师兄，如果无法解决，再联系管理员

了解课题组的服务器

slurm 安装目录`usr/local/slurm`

查看整个 slurm 的使用情况

```
sinfo-s
```

你会看到这个

```
NODELISTPARTIT AVAIL       STATE    MEMORY  ALLOCMEM  FREE_MEM   CPUS(A/I/O/T)
 node1  ptt1    up       mixed   1031221     71680    635979     6/138/0/144
 node2  ptt1    up       mixed   1031221    931840    509399     70/74/0/144
 node3 ptt2*    up       mixed    257125     10240    175912     72/72/0/144
 node4 ptt2*    up       mixed    257125     10240    174097     72/72/0/144
 node5  ptt3    up   allocated    257125     36000      3689     144/0/0/144
 node6  ptt3    up       mixed    257125     35500       724     142/2/0/144
 node7  ptt3    up       mixed    257125      5000      5044    20/124/0/144
 node8  ptt3    up        idle    257125         0      9559     0/144/0/144
```

> `AVAIL`这一列如果出现`down`的标识，请联系管理员恢复节点

这里列出了各个节点更加详细的信息，总内存（MEMORY），可用内存（FREE_MEM），CPU 负载（CPU_LO），各结点可用核心情况（占用/空闲/其它/总量）。

我们的这组服务器一共有 9 台机器，

其中 1 台 master，是用来登录的，不要在上面运行程序，即使是编译程序，也不要用所有的核心一起跑
另外 8 台是用来计算的，命名为 nodeX即$node1$、$node2$、$node3$、$node4$、$node5$、$node6$、$node7$、$node8$

上面那个命令输出的可以看出

node1、node2 被划分在一起即`ptt1`
node3、node4 被划分在一起即`ptt2`
node5、node6、node7、node8 被划分在一起即`ptt3`

> 大家平时需要看明白的是最后一列 `CPUS(A/I/O/T)`：核心情况（占用/空闲/其它/总量） 如果空闲为 0，你提交在这里的程序需要排队，即等待别人计算完毕后自动运算，当然，如果当前分区（节点） `ALLOCMEM` 小于你申请的运行内存，程序也会排队

这是之前王宜森师兄发的 ppt，群里还有

> 请严格按照这个进行提交，

集群分区名	总核心数	总运行内存	功能定位及排队规则
Ptt1 node[1,2]	144	2T	大内存任务分区。运行单核内存 `大于4G` 的任务，鼓励提交耗时较短任务，限制每位用户耗时较长任务所占 核数比例。
Ptt2 node[3,4]	144	512G	小内存耗时长任务分区。主要运行单核内存 `小于4G` 但耗时长于 14 天的任务，限制每位用户耗时较长任务 所占核数比例
Ptt3 node[5-8]	288	1T	单个程序最长运行时间 `14天`，超过会被系统 `强制杀死`！！！ 小内存耗时短任务分区。主要运行单核内存小于 4G 且耗时少于 14 天的任务，限制每位用户排队总核时。

普通提交程序

1. 以下**仅为提交命令说明**，具体关于 matlab、lammps、comsol 的相关提交方式， **点我**

   ```
   srun -p ptt3 -c 4 --mem=5000M --job-name=run2_12 matlab -r main
   ```

   含义:

   1. 执行程序的命令为 `matlab -r main`
   2. `srun` 为 slurm 调度系统固定要求
   3. `-p ptt3`：程序提交到 `ptt3` 分区（无需指定到具体节点，指定以后可能会导致排队时间增加），

      > 注意：`ptt3` 分区单个程序最大运行时间 14 天，超时会被系统强制终止程序运行

   4. `-c 4`：4 个 cpu 线程，但是 `matlab` 会默认一个线程占用一个核心（服务器开启了超线程，平时一个核心两个线程），每个节点144个线程，`matlab` 只能申请72个，对于使用 `openmpi` 的并行，如 `lammps`，可以使用 `-n 4 --ntasks-per-core 1`
   5. `--mem=5000M`：使用 `500M` 内存，一般预估以下自己使用的内存，超出内存系统自动杀死程序，防止无意义运算
   6. `--job-name=run2_12` 这个是给自己的程序加个名字，防止一次提交多个程序想取消时不知道取消哪一个
2. 为了关闭终端程序仍在运行

   ```
   # 此时，会多一个nohup.out的文件，里面的内容是平时在终端运行时看到的内容，每次运行前删除这个文件，因为内容会累加
   nohup srun -p ptt3 -c 4 --mem=5000M --job-name=run2_12 matlab main &
   ```

   > 退出终端一般使用 `ctrl D` 快捷键，而不是直接关闭，直接关闭仍然有可能导致程序被关闭

3. 一般情况下，程序在某个目录，所以

   ```
   # 其中 ~/run2/21.3  为 待运行程序 main 所在目录
   (cd ~/run2/21.3; nohup srun -p ptt3 -c 4 --mem=5000M --job-name=run2_12 matlab main)&
   ```

4. 如果一次提交多个程序，建议把一行行这些命令放在一个文件里，文件名 `s.sh`，然后与程序一起上传到服务器，之后在服务器上直接运行 `sh s.sh` 即可一次性提交所有程序

其他操作

查看

查看所有人提交的程序，输入它，回车

```
squeue
```

或

```
squeue-sq
```

只看某个人的

```
squeue -u 把我改为那个人的用户名
```

或

```
squeue-sq| grep 把我改为那个人的用户名
```

输出类似如下

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
156980      ptt1   matlab chenzhen  R 6-18:36:05      1 node1
156979      ptt1   matlab chenzhen  R 6-18:36:19      1 node2
156978      ptt1   matlab chenzhen  R 6-18:36:34      1 node2
156977      ptt1   matlab chenzhen  R 6-18:36:47      1 node2
156976      ptt1   matlab chenzhen  R 6-18:37:02      1 node2
156975      ptt1   matlab chenzhen  R 6-18:37:20      1 node2
156974      ptt1   matlab chenzhen  R 6-18:37:39      1 node2
156973      ptt1   matlab chenzhen  R 6-18:38:08      1 node2
156964      ptt1   matlab chenzhen  R 6-20:54:36      1 node2
156963      ptt1   matlab chenzhen  R 6-20:54:58      1 node2
156962      ptt1   matlab chenzhen  R 6-21:05:47      1 node2
156961      ptt1   matlab chenzhen  R 6-21:06:03      1 node2
156960      ptt1   matlab chenzhen  R 6-21:06:17      1 node2
156959      ptt1   matlab chenzhen  R 6-21:06:35      1 node2
156958      ptt1   matlab chenzhen  R 6-21:07:26      1 node2
```

其中

JOBID：如`156980`每个程序唯一 id，取消、查找程序时需要，由系统自动分配

PARTITION：如`ptt1`，该程序在哪个分区上，由用户提交时指定

> 注意：`ptt3`分区单个程序最大运行时间 14 天，超时会被系统强制终止程序运行

NAME：如`matlab`，提交时可以指定，请务必在提交时，为每个程序取一个不一样的名字，否则取消时，都不知道具体哪个

USER：如`chenzhen`，这个程序是谁提交的，即`/home/`路径下的用户名，有些的名字长，没有显示完全

ST：这个程序状态，`R`此程序正在运行，`PD`在排队

TIME：已经运行时间，如：`6-18:36:05`：6 天 18 个小时 36 分钟 5 秒

> 注意：`ptt3`分区单个程序最大运行时间 14 天，超时会被系统强制终止程序运行

NODES：用户提交时，对这个程序申请了几个节点

NODELIST：目前在哪儿节点运行，提交程序时用户可指定，但是建议只指定分区即可，指定节点，可能导致排队时间增加，通过这个确定在哪个节点上，然后输入`topn 2`，可以查看`node2`上运行时的详细信息

查看其他节点程序运行情况，比如看节点 1

```
topn 1
```

取消程序

`提交的程序id`，即前面查看的时候第一个数字$JOBID$

```
scancel 提交的程序id

# 取消自己所有的程序
scancel -u 自己的用户名
```

查询程序详细信息

> 只能看自己的

```
scontrol show job 267672
```

一般能看到如下，包含程序内存、cpu、强制被关闭的时间，工作目录等详细信息

```
JobId=267672 JobName=65
   UserId=yuh(1061) GroupId=yuh(1062) MCS_label=N/A
   Priority=1 Nice=0 Account=icpcs QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
   RunTime=6-08:58:18 TimeLimit=14-00:00:00 TimeMin=N/A
   SubmitTime=2021-12-20T09:13:39 EligibleTime=2021-12-20T09:13:39
   AccrueTime=Unknown
   StartTime=2021-12-20T09:13:39 EndTime=2022-01-03T09:13:39 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   LastSchedEval=2021-12-20T09:13:39
   Partition=ptt3 AllocNode:Sid=master:176566
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=node6
   BatchHost=node6
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=1,mem=1G,node=1,billing=1
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=1 MinMemoryNode=1G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=lmp
   WorkDir=/home/yuh/code/mcnt/mcnt_mst_shorter_mst15-15_ls1000-100_cnt16-7_ls2000-
200_temp300_span_rates0.1-0.3/l800_1600/t300/span60/c0
   Power=
```

环境变量说明

openmpi、matlab 应该是生效了，但是如果你用的不是 bash，大概需要自己设置
comsol、intel（含 ifort）、lammps 这个少数人使用的程序只是配置了相关的环境变量，但是没有生效

什么是环境变量

正常来说，不输入路径时，你只能运行的当前目录下有的程序、脚本，但是 matlab 等安装在其他目录，你却可以输入 `matlab` 仍然有效果，因为电脑如果发现当前目录下没有你运行的脚本，会到已经设置了环境变量中的路径中查询，如果找到了，即可运行

这时候问题也就出现了，比如 intel 相关的环境变量内容很多，甚至有些是和其他的程序相冲突，如果全部环境变量都生效，会导致一些 bug，而且启动终端速度极慢，所以一些环境变量只是配置到脚本sh文件中了，谁需要，谁添加

添加环境变量分两种，

一个是，我就用这一次，这时你可以按照说明，临时激活`source sh脚本文件路径`，下次打开终端不再生效
一个是，我经常使用，这时你可以把`source sh脚本文件路径`添加到`~/.bashrc`的文件末尾，当然，如果你是用的是`zsh`，默认中为`bash`，你需要修改的是`~/.zshrc`文件

配置示例

使用永久配置以后，查看添加是否成功

```
cat ~/.bashrc
```

最后一行应该有刚才的`source xxxxx`，但是如果你运行多次，这一行出现了多次，请自行百度如何 vim 编辑环境变量，或联系管理员

openmpi 环境变量

目前支持`openmpi4.1.2`，其中 mpi 版本为`3.2.3`，具体查看输入`ompi_info`，对于默认bash已经配置，对于zsh，请设置

临时

```
source /etc/profile.d/openmpi.sh
```

永久，只能运行一次

```
echo `source /etc/profile.d/openmpi.sh` >> ~/.zshrc
```

comsol 环境变量

临时

```
source /opt/public/sh/start_up/comsol.sh
```

永久，注意 bash 与 zsh 区别

```
echo `source /opt/public/sh/start_up/comsol.sh` >> ~/.bashrc
```

intel 环境变量

临时

```
source /opt/intel/oneapi/setvars.sh intel64
```

永久，注意 bash 与 zsh 区别

```
echo `source /opt/intel/oneapi/setvars.sh intel64` >> ~/.bashrc
```

查看是否生效，关闭终端重新打开，应该会看到类似下面的

```
:: initializing oneAPI environment ...
   zsh: ZSH_VERSION = 5.8
   args: Using "$@" for setvars.sh arguments:
:: advisor -- latest
:: ccl -- latest
:: clck -- latest
:: compiler -- latest
:: dal -- latest
:: debugger -- latest
:: dev-utilities -- latest
:: dnnl -- latest
:: dpcpp-ct -- latest
:: dpl -- latest
:: inspector -- latest
:: intelpython -- latest
:: ipp -- latest
:: ippcp -- latest
:: ipp -- latest
:: itac -- latest
:: mkl -- latest
:: mpi -- latest
:: tbb -- latest
:: vpl -- latest
:: vtune -- latest
:: oneAPI environment initialized ::
```

测试

```
ifort -V
```

看到类似如下

```

Intel(R) Fortran Intel(R) 64 Compiler Classic for applications running on Intel(R) 64, Version
2021.5.0 Build 20211109_000000
Copyright (C) 1985-2021 Intel Corporation.  All rights reserved.
```

lammps 环境变量

只有一个命令，这个对于 `bash` 来说默认配置了，对于zsh如下

临时

```
source /etc/profile.d/lmp.sh
```

永久

```
echo `source /etc/profile.d/lmp.sh` >> ~/.zshrc
```

提交示例

1. 先测试比较一下**多少个 cpu 最划算**，因为申请核心太多时，由于调度关系，速度反而变慢
2. 在自己电脑上优化测试一下你的程序，有时候比多用 cpu 快得多

> matlab 程序在关闭终端后有时候还是会失败，这样退出终端就没问题了，`Ctrl D` 快捷键！ 另外，在 zsh 等终端有时候会关闭即销毁，这时候可以输入 `bash` 切换到 bash，再提交程序

关于slurm的一些命令说明，请查看这里**普通提交程序**

matlab提交

> 运行文件为 `exe0303.m`，在 `~/mat/run2/500000--21.1` 这里

```
(cd ~/mat/run2/500000--21.1; nohup srun -p ptt3 -c 5 --mem=5000M matlab -nodesktop -nosplash -
logfile mat.log -r exe0303)&
```

`exe0303.m` 文件在 `~/mat/run2/500000--21.1` 这里，`matlab` 的日志输出到 `~/mat/run2/500000--21.1/mat.log`，文件 `exe0303.m` 在运行中不要文件后缀名 `.m`

在 `ptt3` 这里计算，使用 `5` 个cpu核心，但是因为超线程的问题，会占用 `10` 个线程，也就是说，一个结点 144 个线程，只能申请 `72` 个

comsol提交

> 很多时候，一个核心一个线程更快，超线程不好

```
(cd ~/code/run1/s1/; nohup srun -c 144 -p ptt3 --mem=100G comsol batch -inputfile in.mph -outputfile
out.mph)&
```

文件 `in.mph` 在 `~/code/run1/s1/` 这里，最大使用 `100G` 内存，在 `ptt3` 上计算，使用 `144` 个核心

lammps提交

编译命令为：

```
cmake -D PKG_PHONON=on -D PKG_KSPACE=on -D BUILD_SHARED_LIBS=on -D LAMMPS_EXCEPTIONS=on -D
PKG_PYTHON=on -D PKG_MANYBODY=yes -D CMAKE_INSTALL_PREFIX=/usr/local/lammps-2022-02-20  ../cmake/
```

支持视频输出，如果缺少某些功能，请联系管理员或者自行编译

默认bash已经设置环境变量，对于时zsh参考 **环境变量说明**

也要申请一个核心一个线程，才最快即

> lammps脚本 `file.in` 在 `~/code/run1/s1/`这里，提交命令如下

```
(cd ~/code/run1/s1/; nohup srun -n 72 --ntasks-per-core 1 -p ptt3 --mem=500M lmp -in file.in)&
```

`--ntasks-per-core 1` lammps文档建议不要超线程，最大只能申请72个线程，因为想当与关闭了超线程

---

icpcs 集群内部文档，主要说明`slurm` 提交任务方法，如果是新手，请先查看 **服务器与集群入门指南**

Department of Electronic Engineering, City University of Hong Kong