集群环境：安装了 Intel 编译器（intel2019_update5） 。

一、关于存储的说明：

用户主目录在 SSD 固态硬盘中,适合存放小文件和安装软件包。用户主目录下面的软连接 low.iops.files -> /home.low.iops/username 指的是机械硬盘，适合存放大文件。

二、关于集群 PBS 相关说明

PBS 可用队列: 1、cu_fat （包含 26 个节点 CPU 节点，每个节点 80 核 CPU 。 单个作业允许的申请的最大 CPU 核数为 160 核，单个作业允许的最大运行时长为 24 小时，每个用户最大允许提交 5 个作业，最多允许同时运行 3 个作业，最多允许排队等待 2 个作业）

2、cu_slim（与 cu_fat 共享 26 个节点 CPU 节点，每个节点 80 核 CPU。单个作业允许的申请的最大 CPU 核数为 4，单个作业允许的 最大运行时长为 5000 小时，每个用户最大允许提交 5 个作业，最多允许同时运行 3 个作业，最多允许排队等待 2 个作业）

3、gpu (包含 10 个 GPU 节点，每个节点有 32 核 cpu 和 4 块 2080TI 显卡，只有申请使用 GPU 的用户才能够使用该队列)

注：请用户根据自己程序运行的需求选择合适的队列提交作业！请注意不要不指定队列直接提交作业,因为系统默认的队列是 batch ,该队列无法使用,提交到该队列的作业会一直排队无法运行!!!!

1、PBS 命令

PBS 提供 4 条命令用于作业管理。

（1）qsub —— 用于提交作业脚本

命令格式：

qsub [-a date_time] [-c interval] [-C directive_prefix]

[-e path] [-I] [-j join] [-k keep] [-l resource_list] [-m mail_options]

[-M user_list][-N name] [-o path] [-p priority] [-q destination]

[-r c][-S path_list] [-u user_list][-v variable_list] [-V]

[-W additional_attributes] [-z]

[script]

（2）qstat —— 用于查询作业状态信息

命令格式：

qstat [-f][-a][-i][-n][-s][-R][-Q][-q][-B][-u]

参数说明：

-f *jobid* 列出指定作业的信息

-a 列出系统所有作业

-i 列出不在运行的作业

-n 列出分配给此作业的结点

-s 列出队列管理员与 scheduler 所提供的建议

-R 列出磁盘预留信息

-Q 操作符是 destination id，指明请求的是队列状态

-q 列出队列状态，并以 alternative 形式显示

-au *userid* 列出指定用户的所有作业

-B 列出 PBS Server 信息

-r 列出所有正在运行的作业

-Qf *queue* 列出指定队列的信息

-u 若操作符为作业号，则列出其状态。若操作符为 destination id，则列出运行在其上的属于 user_list 中用户的作业状态。

例：# qstat -f 211 查询作业号为 211 的作业的具体信息。

（3）qdel —— 用于删除已提交的作业

命令格式：

qdel [-W 间隔时间] 作业号

命令行参数：

qdel -p 强制清除某个作业号，一般不建议使用

例：# qdel -W 15 211　　15 秒后删除作业号为 211 的作业

（4）qhold & qrls —— 作业挂起 & 作业释放

使用 qhold 命令可以挂起作业，使其不被调度执行；

使用 qrls 命令可以将挂起的作业释放，使之可以被调度执行；

命令格式：

qhold *jobid1 jobid2* ...

qrls *jobid1 jobid2* ...

其中 *jobidX* 代表需要操作的作业号，可以一次操作多个作业。

（5）qmgr —— 用于队列管理

qmgr -c "create queue batch queue_type=execution"

qmgr -c "set queue batch started=true"

qmgr -c "set queue batch enabled=true"

qmgr -c "set queue batch resources_default.nodes=1"

qmgr -c "set queue batch resources_default.walltime=3600"

qmgr -c "set server default_queue=batch"

2、PBS 脚本文件

PBS 脚本文件由脚本选项和运行脚本两部分组成。

(1) PBS 作业脚本选项

（若无-C 选项，则每项前面加 '#PBS'）

-a *date_time*　　*date_time* 格式为：[[[[CC]YY]MM]DD]hhmm[.SS]

表示经过 *date_time* 时间后作业才可以运行。

-c *interval*　　定义作业的检查点间隔，如果机器不支持检查点，则忽略此选项。

-C *directive_prefix*　　在脚本文件中以 *directive_prefix* 开头的行解释为 qsub 的命令选项。若无此选项，则默认为 '#PBS'

-e *path*　　将标准错误信息重定向到 *path*

-I　　以交互方式运行

-j *join*　　将标准输出信息与标准错误信息合并到一个文件 *join* 中

-k *keep*　　定义在执行结点上保留标准输出和标准错误信息中的哪个文件。*keep* 为 o 表示保留前者，e 表示后者，oe 或 eo 表示二者都保留，n 表示皆不保留。若忽略此选项，二者都不保留。

-l *resource_list*　　定义资源列表，几个常用的资源种类：

cput=N　　请求 N 秒的 CPU 时间，也可以是 hh:mm:ss 的形式。

mem=N[K|M|G][B|W]　　请求 N {k|m|g}{bytes|words}大小的内存。

nodes=N:ppn=M　　请求 N 个结点，每个结点 M 个处理器。

-m *mail_option*　　*mail_option* 为 a：作业 abort 时给用户发信

为 b：作业开始运行发信

为 e：作业结束运行时发信

若无此选项，**默认为 a**。

-M *user_list*　　定义有关此作业的 mail 发给哪些用户

-N *name*　　作业名，限 15 个字符，首字符为字母，无空格。

-o *path*　　重定向标准输出到 *path*

-p *priority*　　任务优先级，整数，[-1024，1023]，若无定义则为 0

-q *destination*　　*destination* 有三种形式：queue；

@server；

queue@server

-r y|n　　指明作业是否可运行，y 为可运行，n 为不可运行。

-S *shell*　　指明执行运行脚本所用的 shell，须包含全路径。

-u *user_list*　　定义作业将在运行结点上以哪个用户名来运行。

-v *variable_list*　定义 export 到本作业的环境变量的扩展列表。

-V　　表明 qsub 命令的所有环境变量都 export 到此作业。

-W *additional_attributes*　　作业的其它属性

-z　　指明 qsub 命令提交作业后，不在终端显示作业号。

(2) 运行脚本同 Linux 下一般的运行脚本文件

[注]：脚本文件中的 mpirun_rsh 命令行中的节点列表文件要用环境变量表示。

$PBS_NODEFILE，这个环境变量表示由 PBS 自动分配给作业的节点列表；节点数为命令行中指定的进程数。

命令格式：　mpirun_rsh -np *进程数* -hostfile $PBS_NODEFILE *可执行程序名*

3、PBS 环境下运行示例

（1）脚本文件编辑示例

实例 1：运行 mpi 程序

命令行：#vi aaa.pbs

编辑的内容：

#PBS -N *myjob*

#PBS -o /home/jz/my.out

#PBS -e /home/jz/my.err

#PBS -q cu_fat　(指定需要提交作业的 队列，用户按需选择队列)

#PBS -l nodes=2:ppn=2 (这个意思申请 2 个节点，每个节点 2 核 CPU，那总共使用的 CPU 就是 4 核。这里用户可根据自己的需求和队列的实际情况填写，例如说我们 cu_fat 队列单个任务可以最大允许 160 核，而我们的一个节点是 80 核，如果你需要 160 核 CPU，你可以写 nodes=2:ppn=80，或是 nodes=4:ppn=40 ，或是 nodes=8:ppn=20 以此类推……)

PBS -W　x=GRES:gpu@1　（使用 GPU 的数量 1 块,不用 GPU 不写）

cd *目录*（脚本所在的目录）

mpirun -np 4 -hostfile $PBS_NODEFILE /home/jz/helloworld

解释：

原来我们都是在终端输入 mpirun；rsh.....这些命令执行程序的，现在只要把这些提交命令放在.pbs 配置文件的最后，由 PBS 来调度执行（自动分配节点和其它资源）。

myjob 是为要运行的程序起的任务名，可以改成你自己想要的名字。原先输出信息都是直接在屏幕上显示的，现在屏幕上的显示全部输出到文件中，上例中输出文件是/home/jz/my.out 文件，可以根据自己的需要修改（目录，文件名）。程序运行时遇到的一些错误会记录在.err 文件中。这样的好处是，因为对每个任务都设定了不同的输出文件，所以看结果只要打开相应文件看就可以了，不需要开多个终端，而且里面有任务的详细信息，比如实际分配的是哪些节点计算，运行时间等。

mpirun_rsh –np 4 –hostfile $PBS_NODEFILE /home/jz/helloworld

此例中-np 后的 4 是并行数（2×2＝4 个 cpu），-hostfile $PBS_NODEFILE 不需要改变。/home/jz/helloworld 是你编译好的可执行文件名，需修改。

对于每个你要运行的 mpi 程序都需要这样一个.pbs 配置文件，也就是说原来的操作是：mpirun……，现在改成 2 步走：

1）写个 PBS 配置文件（比如 xxx.pbs）；

2）向 PBS 提交（qsub xxx.pbs）

实例 2：运行非 mpi 程序

有些用户并不是自己编写 mpi 程序，同样也可以用 PBS 提交。比如物理系运行程序时一般输入的命令是：

RunDMol3.sh TiFeCp2-pbe-dspp-m=1-opt

那么配置文件可以这样写：

命令行：#vi job.pbs

编辑的内容：

#PBS -N physics_job

#PBS -o /home/physics/physics_job.out

#PBS -e /home/physics/physics_job.err

#PBS -q cu_fat （指定需要提交作业的 队列，用户按需选择队列 ）

#PBS -l nodes=2:ppn=2(这个意思申请 2 个节点，每个节点 2 核 CPU，那总共使用的 CPU 就是 4 核。这里用户可根据自己的需求和队列的实际情况填写，例如说我们 cu_fat 队列单个任务可以最大允许 160 核，而我们的一个节点是 80 核，如果你需要 160 核 CPU，你可以写 nodes=2:ppn=80，或是 nodes=4:ppn=40 ，或是 nodes=8:ppn=20 以此类推……)

PBS -W x=GRES:gpu@1 （使用 GPU 的数量 1 块,不用 GPU 不写）

#PBS -r y

cd *目录*（原来直接在节点上运行时所在的目录）

RunDMol3.sh TiFeCp2-pbe-dspp-m=1-opt

解释：

把原来在终端直接输入的命令放到 PBS 配置文件中，因为只要一个节点，所以 nodes=1，至于用哪个节点系统自动分配，可以用 qstat 命令查询（比如 qstat -n）。

（2）提交作业示例

命令行：#qsub aaa.pbs

（3）作业状态查询示例

qstat 后加不同参数可以查看不同的信息，查看作业的状态。

命令行：#qstat -a

解释：

Job id 211 是给提交的任务分配的任务号，S（常用状态：R 代表运行，Q 代表排队，E 代表正在退出，H 代表挂起，C 代表运行完毕）

命令行：#qstat -n 查看作业使用的节点

命令行：#qstat *jobid1 jobid2* … 查看指定作业号的作业（可一次查看多个作业）

命令行：#qstat -u *user1* 查看指定用户的作业

解释：该方式输出和默认略有不同，但大同小异。

命令行：#qstat -f *jobid*　　查看特定作业详细信息

解释：该命令将会输出作业号为 *jobid* 的作业的详细信息。