PBS Pro使用指南

v2.0

广州景派科技有限公司

文档控制

更改记录

日期	版本	更改人	审批人	更新内容
2018/08/19	v1.0	陆炎		
2019/01/12	v2.0	陆炎		1. 第 5 章内容全部进行修改； 2. 添加附录《PBS Pro 提交脚本的内置变量》

目录

1. 系统概述

1.1 高性能计算机系统结构

高性能计算机一般由计算处理、互联通信、I/O 存储、监控诊断、基础架构、操作系统、编译器、运行环境、开发工具等多个软硬件子系统组成。

从资源管理系统的角度，高性能计算机中的节点按逻辑功能可分为管理节点、登录节点、计算节点和 I/O 节点；节点之间通过网络联接。图 1.1所示为高性能计算机的逻辑结构。

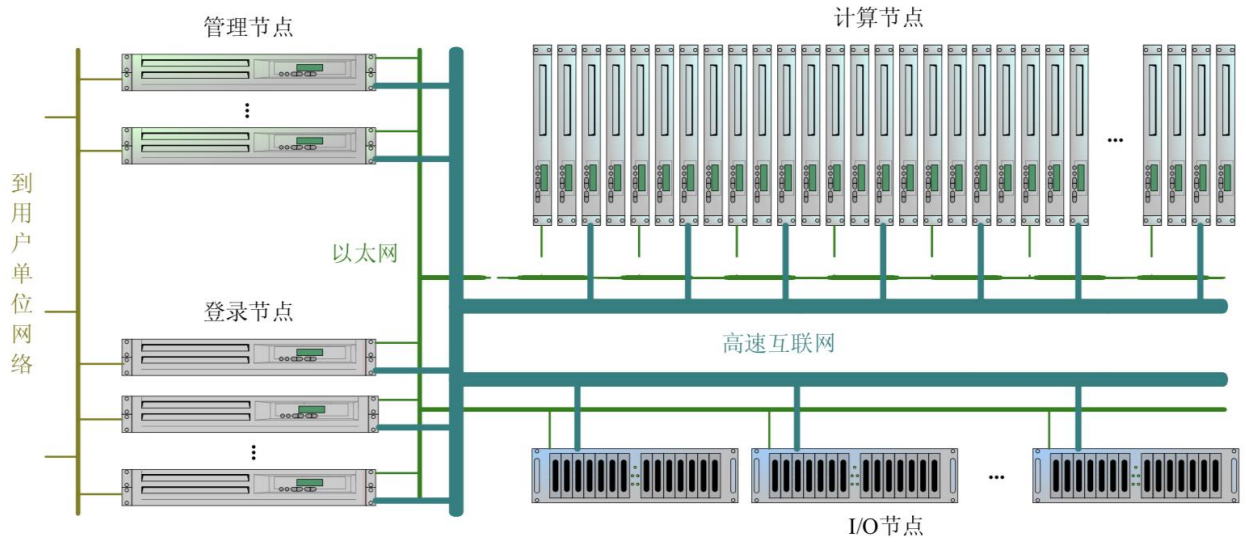


图 1.1: 高性能计算机系统逻辑结构

管理节点运行系统管理进程与相关的支撑服务。管理员在需要时登录管理节点,进行系统的管理与配置工作。普通用户不能登录管理节点。管理节点主机名字一般为mgt,mgt1,mgt2...等。

登录节点是用户使用高性能计算机的访问点，供用户登录进行程序编辑、编译，作业提交、运行，结果查看、分析等。登录节点的主机名字一般为ln0,ln1,...等。

计算节点提供高性能计算能力，运行用户作业的计算任务。资源管理系统所管理的计算资源就是指系统中的计算节点。计算节点的主机名字一般为cn0,cn1,...等。

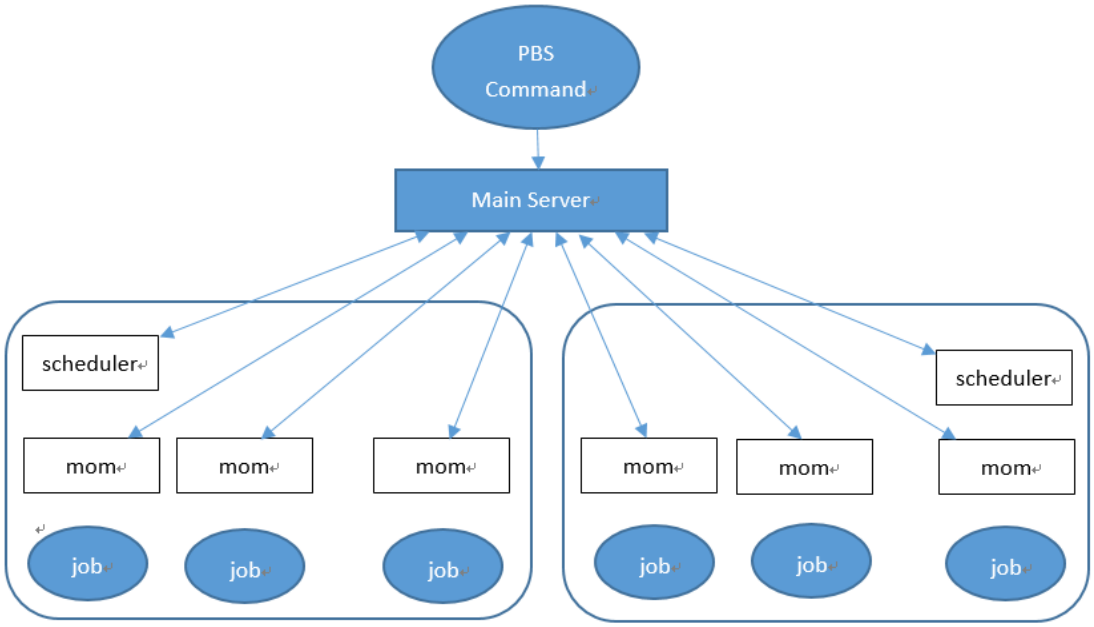
I/O 节点提供存储服务，在使用上表现为一个或多个全局共享的文件系统。登录节点和计算节点在相同的目录挂载这些文件系统。一般在较小的集群环境中，I/O节点的主机名字一般命名为ion0,ion1,…….等；在较大型的分布式文件系统的集群环境中，I/O节点的主机名字一般为mds0,mds1,...元数据服务器）以及 ost0,ost1,...（对象存储服务器）等。

系统中的节点之间通过高速互联网联接，提供高带宽、低延迟的计算和 I/O 通信链路。一般地，管理节点、登录节点和 I/O 节点之间还通过以太网进行联接，供系统的管理数据通信、调测试、监控诊断等使用。在某些系统中，计算节点也通过以太网与其它节点联接。管理节点和登录节点通过以太网联接到用户单位的内部网络，供管理员和用户远程访问。

注意：以上描述的是高性能计算机的逻辑结构。对于具体的系统，可以存在物理上的一个节点对应于多个逻辑节点的情况。例如，对于规模较小的系统，可能管理节点和登录节点是同一个物理节点；登录节点也可以同时作为计算节点。

1.2 PBS pro系统的组成架构

pbs pro的架构拓扑如下图所示：



PBS Professional是一个用于复杂和高性能计算环境的工作负载管理器和调度器。可以使用它来简化作业提交、跨多个平台集群分布工作负载，以及扩展到数百甚至数千个处理器。

本文将概述 PBS Professional用于满足组织需求的功能。它在单个或成组在一起的多个系统中支持作业。

PBS Professional的功能包括：

- 接受批作业
- 保持和保护作业，直到作业运行
- 运行作业
- 将输出交付给提交者

作为一个调度器， PBS Professional使用复杂的管理策略和调度算法来获得诸如此类的功能：

- 性能数据的动态收集
- 增强的安全性
- 高级别的策略管理
- 服务质量 (QoS)
- 处理有大量计算的任务
- 相关工作负载的透明分布
- 高级资源保留
- 支持检验点程序

PBS Professional中有 4 个主要的组件：

- 客户端命令 ——用于提交、控制、监视或删除作业，并可以安装在任意支持的平台上。其中包括诸如qsub 和qdel 之类的命令。
- 服务器 (pbs_server) ——为批服务（比如创建和修改作业，以及保护作业免遭系统崩溃的影响）提供主要的入口点。所有客户机和其他守护程序都要通过 TCP/IP 与该服务器通信。
- 调度器 (pbs_sched) ——控制用于通过网络提交作业的策略或规则。每个集群可以创建其自己的调度器或策略。开始时，调度器查询服务器获得将要运行的作业，查询执行程序了解系统资源的可用性。
- 作业执行程序 (pbs_mom，下文中也称为MoM) —— 通过模仿用户会话等于用户的逻辑会话而执行作业。它在定向之后将输出交付给调用者。

在典型的集群配置中， pbs_mom 将运行在作业将要运行在其上的每一个系统上。服务器和调度器可运行在相同的机器上，客户端命令可放置在将提交作业的机器上。 PBS Professional为想要实现其自己的调度策略的站点提供一个应用程序编程接口。

1.3 PBS pro相关目录及其描述、权限

默认情况下，PBS Professional的配置文件为/etc/pbs.conf，如果在配置文件中定义了PBS_MOM_HOME属性的话，则pbs_mom相关的文件所有文件都将部署在PBS_MOM_HOME属性指定的目录下。需要注意的是，该目录下的所有文件属主都应该为root。PBS_MOM_HOME目录下相关的文件或文件夹的描述以及权限如下：

File/Directory	Description	Permissions
aux	Directory	0755
checkpoint	Directory	0700
checkpoint script	File	0755
mom_logs	Directory	0755
mom_priv	Directory	0751
mom_priv/jobs	Directory	0751
mom_priv/config	File	0644
mom_priv/prologue	File	0755
mom_priv/epilogue	File	0755
pbs_environment	File	0644
spool	Directory	1777 (drwxrwxrwt)
undelivered	Directory	1777 (drwxrwxrwt)

Version 2 configuration files	Files	0755
PBS reserved configuration files	Files	----
Job temporary directory	Directory	1777

2. 配置MoMs与vnode

本章节主要介绍怎样配置Pbs MoMs以及配置vnode。

2.1 配置node

可以通过以下方式来配置一个计算节点node：

1）通过/etc/pbs.conf文件配置pbs相关的环境变量，以及集群角色、启动进程等相关信息。通常/etc/pbs.conf文件会有以下的内容：

```
PBS_SERVER=mgt                  #指定pbs服务器名
PBS_START_SERVER=1              #本节点是否启动pbs server进程
PBS_START_SCHED=1               #本节点是否启动pbs sched进程
PBS_START_COMM=1                #本节点是否启动pbs comm进程
PBS_START_MOM=1                 #本节点是否启动pbs MoM进程
PBS_EXEC=/opt/pbs               #pbs的执行文件目录
PBS_HOME=/var/spool/pbs         #pbs运行时的家目录
PBS_CORE_LIMIT=unlimited
PBS_SCP=/bin/scp
```

2）通过修改PBS_HOME/mom_priv/config文件，来配置MoM的运行参数。通常该文件的默认参数如下：

```
$clienthost mgt                 #指定允许该主机访问MoM
```

$restrict_user_maxsysid 999　#指定用户 UID 如果小于等于该值的话，可以不受$restrict_user 参数限制

3）使用qmgr命令来创建node：

qmgr创建node的命令格式如下：

qmgr -c ‘create node <vnode name> [<attribute> = <value>]’

其中<attribute> = <value>内容为可选内容。

或者在命令行输入qmgr命令，可以进入qmgr的交互式配置模式中（此时命令行提示符为”Qmgr:”），在交互式模式下执行：

create node <vnode name> [<attribute> = <value>]

如：create node cn1 resources_available.ncpus = 24

node的调度颗粒度为主机级别（host-level），如果需要对调度的资源再细化就需要配置vnode。

2.2 什么是Vnode

vnode即virtual node（虚拟节点）。

一个vnode，是一个资源集的载体，可以是一台主机、服务器中可用的一部分；可以是实体主机，也可以是云服务器。对于一台主机，它可以包含有多个vnode，每一个vnode都可以被pbs独立地管理和调度使用。

pbs_mom执行文件在PBS_EXEC/sbin下，且只能用root运行。

2.2.1 主机与vnode的关系

一般意义上，主机（host）就是我们所指的计算机这个实体。这些host由包含cpu、内存、gpu、存储空间等组成。但是在pbs pro的调度系统中，一个vnode可以是其中的任何一个独立部件，也可以是其中的几个部件共同构成，也可以是其中任何部

件的其中一部分。

每一个vnode都有与之相关联的属性集（attribute）与资源集（resources），运行在该host上的软件也可以被看做是vnode上的资源。

因此，有些主机有一个vnode，有些主机有多个vnode。Pbs Pro对所有的vnode在大部分场景都一视同仁。

2.2.2 Natural Vnode

在一台有多个vnode的主机上（host），会有一个特别的vnode叫做natural vnode。natural vnode不与任何的硬件有关联关系，它的作用仅仅是用于pbs_server来标识host上多个vnode的定义信息、位置信息等信息映射作用。

一般 natural vnode 通常还用于定义 host-level（主机级别）的 resource，在 <sched_priv directory>/sched_config 文件中 mom_resources 一行定义的 resource 将会在所有该主机上的 vnodes 都共享其定义。

2.3 PBS Pro的配置文件类型

pbs有三种不同类型的配置文件：

- Version 1 Configuration file
- PBS reserved Configuration file
- Version 2 Configuration file

Version 1 类型配置文件通常在PBS_HOME/mom_priv/config目录下，主要用于控制 MoM的行为。该配置文件的权限必须安全，用户ID以及组ID均要小于10，且不能人人可写。

PBS reserved 类型的配置由PBS自己生成，一般前缀带有PBS。如果对这一类型的配置文件进行修改的话，PBS很可能会报错。

Version 2类型的配置文件由管理员创建，文件中包含有vnode的属性设置选项。建议不要直接编辑这些文件，你可以先创建一个文件，然后使用pbs_mom -s insert [option]命令，pbs会创建为你创建一个新的配置文件。version 2类型配置的属性或资源可以通过使用pbs_mom -s option命令来查询或添加、修改、删除。

MoM启动时加载配置文件的顺序如下：

version 1类型配置文件==>reserved类型配置文件==>version 2类型配置文件

后来加载的重复属性和内容会覆盖先加载的内容

2.3.1 Version 2类型的MoM配置文件

version 2的配置文件开头必须如下：

$configversion 2

剩余的内容格式如下：

<vnode ID> : <attribute name> = <attribute value>

其中：

- <vnode ID>为不包含":"的字符序列。vnode ID在同一个pbs_server集群中是唯一标识
- <attribute name>为开始于字母或数字，包含_-@[]#^/\.的字符序列
- <attribute value>为不包含=的字符序列

version 2类型的配置文件可以使用pbs_mom -s insert [option]命令导入配置。

2.4 配置Vnodes

配置vnode一般可以有两种方法：

- 使用qmgr命令来进行配置vnode

- 通过编辑Version 2类型的配置文件，然后使用pbs_mom -s insert 命令来导入vnode设置来进行配置

2.4.1 配置单Vnode节点

1）通过/etc/pbs.conf文件配置pbs相关的环境变量，以及集群角色、启动进程等相关信息。

2）通过配置Version 1类型的参数文件，来配置MoM的运行参数。

3）使用pbs_mom -s insert来导入vnode配置，或使用qmgr命令来设置vnode的属性（attributes）和全局的资源（resources）

qmgr命令设置vnode属性格式如下：

qmgr -c ‘set node <vnode name> <attribute> = <value>’

如：set node cn1 resources_available.ncpus = 24

2.4.2 配置多Vnode节点

1）通过/etc/pbs.conf文件配置pbs相关的环境变量，以及集群角色、启动进程等相关信息；

2）通过配置Version 1类型的参数文件，来配置MoM的运行参数；

3）使用pbs_mom -s insert来导入vnode配置，或使用qmgr命令来设置vnode的属性（attributes）和全局的资源（resources）；

4）可以使用pbsnodes 命令来设置多vnode节点上的natural vnode的状态以及其他主机级别（host-level）的属性。

2.4.3 pbsnodes设置主机offline

设置一台主机offline可以使用下面的命令:

pbsnodes -o <hostname [hostname ...]>

删除一台或多台主机的offline属性，可以使用下面的命令

pbsnodes -r <hostname [hostname ...]>

注意pbsnode命令只能作用于host，不能作用于vnode等级，指定到vnode基级别的需要使用qmgr命令。

2.5 查看Vnode信息

1) 使用pbs_mom -s list命令列出当前加载的version 2配置，命令结果是当前已加载的scriptname，如：

   # pbs_mom -s list

   cn9_vnodes

其中cn9_vnodes为当前已加载的version 2配置scriptname。

2) 使用pbs_mom -s show scriptname命令可以列出列出该配置文件的详细信息。

   如：

   # pbs_mom -s show cn9_vnodes

以下为输出内容：

```
$configversion 2

cn9: resources_available.ncpus = 0

cn9: resources_available.mem = 0



cn9[0]: resources_available.ncpus = 6

cn9[0]: resources_available.mem = 14gb

cn9[0]: resources_available.ngpus = 1

cn9[0]: resources_available.gpu_id = gpu0

cn9[0]: sharing = default_exclusive
```

```
……....
```

3) 另外，如果使用pbsnodes -v <vnode ID>可以查看到具体vnode ID对应vnode的详细信息，如：

#pbsnodes -v cn9[0]

以下为输出内容：

```
cn9[0]
     Mom = cn9.gpyhpc.sysu
     Port = 15002
     pbs_version = 18.1.2
     ntype = PBS
     state = free
     pcpus = 1
     resources_available.arch = linux
     resources_available.gpu_id = gpu0
     resources_available.host = cn9
     resources_available.mem = 14gb
     resources_available.ncpus = 6
     resources_available.ngpus = 1
     resources_available.vnode = cn9[0]
     ……
```

2.6 修改Vnode属性

修改vnode属性可以通过修改version 2配置文件或直接使用qmgr命令修

改。

需要注意的是，修改version 2类型的配置文件需要重启或SIGHUP信号才能生效，并且pbs server不会马上知道MoM上对vnode进行的修改，需要等一个MoM的投票周期（Polling Cycle）时间才能接收到vnode的配置变化。MoM Polling Cycle的时间由version 1配置文件中的$min_check_poll以及$max_check_poll参数共同决定。一般$min_check_poll默认为1秒，$max_check_poll默认为120秒。

如果使用qmgr命令，则命令格式如下：

qmgr -c ‘set node <vnode ID> <attribute> = <value>’

如：qmgr -c ‘set node cn9[0] queue = gpu_reserve’

2.7 删除Vnode

使用pbs_mom -s remove scriptname命令删除vnode，如：

pbs_mom -s remove cn9_vnodes

2.8 配置Vnode注意事项

1) 不要直接修改编辑pbs保留、version2两种配置文件，而是使用pbs_mom -s 选项；
2) 使用pbs_mom -d选项来改变PBS_HOME,如果使用-d选项，MoM将会查找指定路径下的所有pbs reserved以及version 2类型的配置文件；
3) 启动的时候可以使用pbs_mom -c指定pbs_mom要读的配置文件；
4) 不要修改PBS的reserved类型配置文件；
5) 使用qmgr命令做的配置会覆盖配置文件的相关配置；
6) 3种配置文件的格式语法以及信息都是不一样的，需要区分；

7) 当你使用一个预先配置好的version 2类型的vnode配置文件，请确定该配置文件指定了所有的需要信息。使用pbs_mom -s命令创建的配置文件会覆盖预先配置的选项，如果新生成的配置没有指定的资源或属性，则该vnode将不可用；

8) version 2类型的配置文件可以被移动。可以使用以下的方法来从一个实例移动到另一个MoM实例：

1）在计算节点上使用pbs_mom -s list 选项列出当前的vnodes

2) 再使用pbs_mom -s show scriptname来列出详细的vnode配置，可以利用重定向保存成文件；

3）使用pbs_mom -s insert 命令来导入刚才保存的配置。

9) 当编辑pbs任意版本的配置文件时，必须要在文件的最后要有一行新行（空白行）；

10) 配置文件的修改必须要重启或SIGHUP信号才能生效。

2.9 示例 - 按GPU卡数调度的Vnode配置

在cn7主机上配置8个vnode，每个vnode ID对应为cn7[0]、cn7[1]、cn7[2]……cn7[7]，其中每个vnode上通过resources_available.ngpus = 1属性来指定一个vnode对应有1张GPU计算卡，从而达到根据ngpus数量来调用vnode。同时，8张gpu即使有数张gpu被调用了，其余的gpu资源也能够被其他job进行调用：

```
$configversion 2

cn7: resources_available.ncpus = 0

cn7: resources_available.mem = 0



cn7[0]: resources_available.ncpus = 6
```

cn7[0]: resources_available.mem = 11gb

cn7[0]: resources_available.ngpus = 1

cn7[0]: resources_available.gpu_id = gpu0

cn7[0]: sharing = default_exclusive

cn7[1]: resources_available.ncpus = 6

cn7[1]: resources_available.mem = 11gb

cn7[1]: resources_available.ngpus = 1

cn7[1]: resources_available.gpu_id = gpu1

cn7[1]: sharing = default_exclusive

cn7[2]: resources_available.ncpus = 6

cn7[2]: resources_available.mem = 11gb

cn7[2]: resources_available.ngpus = 1

cn7[2]: resources_available.gpu_id = gpu2

cn7[2]: sharing = default_exclusive

cn7[3]: resources_available.ncpus = 6

cn7[3]: resources_available.mem = 11gb

cn7[3]: resources_available.ngpus = 1

cn7[3]: resources_available.gpu_id = gpu3

cn7[3]: sharing = default_exclusive

cn7[4]: resources_available.ncpus = 6

cn7[4]: resources_available.mem = 11gb

cn7[4]: resources_available.ngpus = 1

cn7[4]: resources_available.gpu_id = gpu4

cn7[4]: sharing = default_exclusive

cn7[5]: resources_available.ncpus = 6

cn7[5]: resources_available.mem = 11gb

cn7[5]: resources_available.ngpus = 1

cn7[5]: resources_available.gpu_id = gpu5

cn7[5]: sharing = default_exclusive

cn7[6]: resources_available.ncpus = 6

cn7[6]: resources_available.mem = 11gb

cn7[6]: resources_available.ngpus = 1

cn7[6]: resources_available.gpu_id = gpu6

cn7[6]: sharing = default_exclusive

cn7[7]: resources_available.ncpus = 6

cn7[7]: resources_available.mem = 11gb

cn7[7]: resources_available.ngpus = 1

```
cn7[7]: resources_available.gpu_id = gpu7

cn7[7]: sharing = default_exclusive
```

注意，添加gpu资源，需要先在$PBS_HOME/server_priv/resourcedef配置文件中定义，在配置文件中添加一行：

ngpus type=long flag=nh

同时，在$PBS_HOME/sched_priv/sched_config配置文件中的resource一行修改为：

resources: “ncpus, mem, arch, host, vnode, ngpus”

修改完成后，pbs server需要重启生效。

编写完以上的version 2配置文件，并且修改了资源类型后，使用以下命令来导入version 2的配置：

/opt/pbs/sbin/pbs_mom -s insert cn7_vnodes cn7_vnodes

用户在提交job时，使用以下方式调用gpu资源：

qsub -l select=1:ngpus=3 <job name>

3. 配置queue

用户提交的作业，会变提交到队列（queue）中。

默认情况下，Pbs Pro中有一条默认队列workq，如果job提交不指名提交到的队列，将自动提交到默认队列上。

3.1 PBS Pro的队列类型

Pbs Pro主要有两种队列类型：routing 、execution。

- routing类型的队列实际只负责将job路由到其他的execution队列中，同时routing类型的队列也可以把job路由到其他不同pbs server中；
- execution类型的队列。作业必须要最终运行在execution类型的队列中，即使作业正在运行中，它也仍然驻留在队列中。

3.2 如何创建队列

创建队列的几个例子：

使用qmgr命令来创建，创建队列需要指定队列的类型（queue_type）。

qmgr创建队列的格式如下：

Qmgr: create queue <queue name>

Qmgr: set queue <queue_name> queue_type = <execution or route>

或：

Qmgr: create queue <queue name> queue_type = <execution or route>

队列创建后必须要设置enable才能接受job提交，默认队列创建后是disalbe的状态：

Qmgr: set queue <queue name> enabled = True

队列即使enable了，但默认队列上的job是不允许执行的，需要执行以下命令允许队列job运行：

Qmgr: set queue <queue name> started = True

示例1，创建一条execution类型的队列，队列名称为exec_queue：

Qmgr: create queue exec_queue

Qmgr: set queue exec_queue queue_type = execution

Qmgr: set queue exec_queue enabled = True

Qmgr: set queue exec_queue started = True

示例2，在创建了上面的execution队列后，再创建一条新的routing队列，将接收到的job路由到exec_queue队列上：

Qmgr: create queue routing_queue

Qmgr: set queue routing_queue queue_type = route

Qmgr: set queue routing_queue route_destinations = exec_queue

3.2.1 限定队列只接收从routing队列提交的job

默认情况下，pbs允许使用qmove命令从其他队列移动job到一个execution队列中。

通过设置队列属性from_route_only=True，则该队列只接收routing队列的任务提交

Qmgr: set queue <queue name> from_route_only = True

3.3 Routing队列的特征

routing队列有以下的特征：

1) 可以有多个路由目标；

2) 通过RR轮询调用不同的目标（根据order list的顺序）；

3) 可以路由到execution队列，也可以路由到routing队列；

4) 可以路由job到另一个pbs server；

5) 使用qstat -Q命令可查看队列路由的情况。

指定路由目标的方式可以类似下面形式，通过指定route_destinations属性实现：

route_destinations = Q1

route_destinations = Q1@Server1

route_destinations = "Q1, Q2@Server1, Q3@Server2"

route_destinations += Q1

route_destinations += "Q4, Q5@Server3"

3.4 Execution类型队列

Execution类型的队列实际又有多种类型：

- Reservation queues，只用于执行reservation类型的job；
- Dedicated time queues，挂起作业，直到指定的时间才运行的队列；
- Primetime queues，挂起作业，只在设置好的黄金时段才执行的作业队列；
- Non-primetime queues，挂起作业，只在设置好的非黄金时段才执行的作业队列；
- Anytime queues，不受专用时间和黄金时段限制的作业队列；
- Express queues，高优先级队列，优先运行；
- Anti-express queue，低优先级队列，当没有其他job运行时才运行的队列。

影响以上几种execution类型的相关配置：

1) holiday time，在sched_priv目录下的holidays文件中定义。默认情况下pbs定义有默认的holidays.holiday，相当于non-primetime。non-primetime队列在holiday时间运行；

2) dedicated Time，在<sched_priv directory>/dedicated_time中定义dedicated_time时间段，Dedicated time queues将在这个时间段运行；

3) primetime，在<sched_priv directory>/holidays文件中定义相关时间。Primetime队列的任务将在这个时间段运行，反之Non-Primetime队列则在primetime以外的时间段运行；

4) 定义primetime队列，队列名称的前缀为“p_”；而non-primetime队列，队列名称前缀为“np_”,；dedicated time 队列名称的前缀为“ded”；修改这些前缀名称，可以在<sched_priv directory>/sched_config配置文件中通过修改相关的primetime_prefix 、nonprimetime_prefix、dedicated_prefix参数；

5) 使用Qmgr: set queue <queue name> priority = <value>来设定队列的优先级，数值越大优先级越高，preempt_queue_prio的默认值是150，队列的priority值在routing jobs时不发挥作用。优先级值必须是integer数值。一个队列如果优先级设置得比 <sched_priv directory>/sched_config文件中的preempt_queue_prio参数要高的话，那么他就是一个Express Queues，反之则为Anti-express队列；

6) anytime队列为无特殊前缀，不指定优先级的队列，可以运行在任何时间段；

3.5 限制指定用户使用指定队列

可以通过创建队列acl_users和acl_groups，限制其他用户访问该队列资源，acl_users和acl_groups的默认行为是禁止list之外的用户访问队列：

Qmgr: set queue <queue name> acl_users = “User1@*.example.com, User2@*.example.com”

Qmgr: set queue <queue name> acl_user_enable = True

Qmgr: set queue <queue name> acl_groups = “Group1@*.example.com, Group2@*.example.com”

Qmgr: set queue <queue name> acl_group_enable = True

3.6 关联vnode与queue

一个vnode关联一个队列的情况，可以通过设定vnode的queue属性来进行配置：

Qmgr: set node cn9[0] queue = gpu_reserve

通过上面的配置，将cn9[0]这个vnode与gpu_reserve队列进行了关联，那么将无法再通过gpu_reserve以外的队列调用到cn9[0]这个vnode。同时，需要说明的是，队列配置了关联vnode外，则该队列就只有配置了关联的vnode资源可用。

要取消vnode与队列的关联，只需要使用下面的命令：

Qmgr: unset node <vnode name> queue

3.7 如何查看队列状态

如需要查看一个队列的状态，包括队列的类型、目前job状态统计、运行作业数等，可以通过以下命令进行查看（通常只显示非默认的属性值）：

qstat -Qf <queue name>

如果需要查看所有队列的情况，则可以不指定队列名：

qstat -Qf

可以使用以下命令来查询所有队列上job的运行情况：

qstat -a

job的运行状态在查询结果的State字段中，有E (enabled) 、D (disabled.) 、 R (running与started相同) 、 S (stopped)等几种常见状态。

3.8 删除队列

删除队列使用命令：

Qmgr: delete queue <queue name>

注意，如果队列正在使用中则不能被删除。另外，如果队列被其他vnode通过vnode的queue属性所关联则不能删除该队列。要删除该关联，保存pbsnodes -a的输出信息，通过这些信息来查找相关vnode。unset相关联的vnode的queue属性即可。

3.9 配置队列的几个注意事项

1. 确保routing队列route_destinations中的队列要先创建；
2. 确保route_destinations中的队列要先enabled；

3. 队列名必须是字母或数字，必须开始于字母；一个server必须要有execution类型的队列；
4. 队列的属性有execution、routing两种类型共有的属性，也有各自专用的属性。一旦execution的队列使用了routing队列的属性（或反之），PBS会忽略相应的属性，并且在stderr中提示冲突以及warning信息；
5. 应避免路由循环或自己调用自己，路由最多经过20跳就会丢弃掉；
6. 如果路由到不同的Pbs server中，确保两个pbs的版本不一样；
7. 建议限制最严格的队列排在路由队列的最前面。

4. 配置Resource与作业调度

Resource（资源）可以是所有能用于job的东西，包括CPU、内存、gpu、运行软件、磁盘空间等。Pbs Pro中有定义好的内置资源（built-in resources），同时也允许用户自定义resource。

Pbs Pro的资源可以设定在server、queue、vnode上，这些资源中的任何一种都可以是静态/动态的、内置/自定义的、可消耗（consumable）/非消耗的（non-consumable）、全局/局部的。

4.1 Pbs Pro的内置资源

Pbs pro中的内置资源包括：ncpus（CPU的核心数）、mem（内存）、vmem（虚拟内存）mini_walltime（最小运行时间）、max_walltime（最大运行时间）、mpiprocs（mpi进程数）、pmem（一个进程的最大运行物理内存大小）、pvmem（一个进程的最大运行虚拟内存大小）等。

4.2 设定resource属性

- 可以使用qmgr命令来设置resource的属性，格式如下：

  qmgr -c "set <object> ressources_available.<resource name> = <value>"

  或在qmgr交互模式下：

  Qmgr: set <object> resources_available.<resource name> = <value>

- 取消配置可以使用下面的命令格式：

  qmgr -c "unset <object> resources_available.<resource name>"

  或在qmgr交互模式下：

  Qmgr: unset <object> resources_available.<resource name>

- 如果是string_array类型的的资源，可以使用下面的格式来定义：

  Qmgr: set <object> resources_available.<resource name> = '<value,value>'

  或：

  Qmgr: set <object> resources_available.<resource name> += <value>

以上的object可以是server、queue以及vnode里的任意一种，使用qmgr命令设定的资源会马上生效。

如定义队列存在的资源，可使用下面的命令格式：

Qmgr: set queue <queue name> resources_available.<resource name> = <value>

例（定义QueueA上存在Matlab资源）：

Qmgr: set queue QueueA resources_available.Matlab = True

另外，也可以通过version 2类型的配置文件来定义资源。

4.3 资源的限制使用

1. 除了mini_walltime与max_walltime外，其他的资源都可以使用下面的方法来进行限制：

   set <object> resources_min.<resource name> =<value>

   set <object> resources_max.<resource name> =<value>

当Job被提交后，判断能否进入一个队列，都要判断job的资源请求是否大于等于resources_min. <resource name>以及小于等于resources_max. <resource name>(只有job提交时有具体资源请求或继承defaults请求中涉及到的资源才会进行判断)。

如果这两个限制条件被设置为一个字符数组，则请求的值必须在这个数组中存在才符合条件。

例如，当进行以下配置：

set resources_min.str_arr=“blue,red,black”

set resources_max.str_arr =“blue,red,black”

则job需要在请求时指定了–l str_arr=blue或-l str_arr=red或or –l str_arr=black才能被需要提交到这个队列中。

对于job的运行时间限制由 resources_max.walltime以及resources_min.walltime来进行限制。

4.4 关于资源请求继承与资源限制继承

通过创建resources的default值来限定不指定资源请求的任务，这些任务通常会继承设定的resources default值。

例如，指定resource walltime是12小时，如果请求job时不指定walltime，则会自动继承default walltime值为12小时，如果指定则不会继承这个值。

Default resources还可以指定到server和每一个queue上。server上定义的default resources作用于所有jobs，queue上定义的则作用于运行在该队列上的jobs。通过以下命令格式来设定server的资源默认值：

Qmgr: set server resources_default.<resource name>=<value>

还可以指定到块级（vnode级别）资源：

Qmgr: set server　default_chunk.ncpus=1

Qmgr: set server　default_chunk.mem=1gb

Qmgr: set queue small default_chunk.ncpus=1

Qmgr: set queue small default_chunk.mem=512mb

你也可以指定默认资源去添加qsub提交时的默认参数，当用户不指定该请求相关的选项时默认添加：

Qmgr: set server default_qsub_arguments=<string containing arguments>

如：Qmgr: set server default_qsub_arguments= ”-r y -N MyJob”

如：Qmgr: set server default_qsub_arguments=”-l Red=False” （指定默认不在

Red的资源上运行）

如果一个job在提交的时候没有指定资源限制，并且server以及resource都没有设定defaults时，如果server跟queue都有resources_max.<resource name>限制，则job会继承其中的最大值；如果queue有设定resources_max.<resource name>值，则job会继承自queue值，queue无设定则继承自server设定的resources_max.<resource name>值。

4.5 配置示例

示例：用户提交job时如不指定队列与运行时间，则被默认将运行时间设置为4小时，同时被提交至ShortQ队列（运行时间低于5小时的队列），而如果运行时间大于或等于5小时的job会被提交至LongQ中。如对于运行时间不确定的，可以在提交时指定提交任务到WorkQ队列中：

1) 创建一条Routing类型的队列，同时将其设置为默认队列:

Qmgr: create queue RouteQ queue_type = route

Qmgr: set server default_queue = RouteQ

2) 创建两条execution类型的队列LongQ 和 ShortQ。 LongQ 队列用于较长运行时间的jobs，ShortQ队列则用于较短运行时间的jobs:

Qmgr: create queue LongQ queue_type = execution

Qmgr: create queue ShortQ queue_type = execution

Qmgr: set queue LongQ resources_min.walltime = 5:00:00

Qmgr: set queue ShortQ resources_max.walltime = 4:59:00

3) 将LongQ和ShortQ设置为只能从routing类型的队列中接收job提交：

Qmgr: set queue LongQ from_route_only = True

Qmgr: set queue ShortQ from_route_only = True

4) 设定RouteQ的目的队列为LongQ和ShortQ：

Qmgr: set queue RouteQ route_destinations = “ShortQ, LongQ”

5) 创建一条WorkQ队列，用于提交运行时间不确定的队列：

Qmgr: create queue WorkQ queue_type = execution

6) 可以用以下方式一起enable以及start队列：

Qmgr: active queue RouteQ,LongQ,ShortQ,WorkQ

Qmgr: set queue enabled = True

Qmgr: set queue started = True

7) 通过设定server的默认运行时间，如用户不指定job运行时间的将会继承server的默认运行时间，然后被分派到ShortQ队列中：

Qmgr: set server resources_default.walltime = 4:00:00

4.6 注意事项

- 默认情况下，PBS不会路由已挂起的jobs，你可以通过设定route队列的route_held_jobs=True来使队列支持路由挂起的jobs：

  Qmgr: set queue <queue name> route_held_jobs = True

- 默认情况下，PBS不会路由开始时间还没到来的jobs，可以通过设定route队列的route_waiting_jobs=True来使队列支持路由未来才能运行的jobs：

  Qmgr: set queue <queue name> route_waiting_jobs = True

- 队列重试时间的默认值为30 seconds,可以通过下面方式设置其值：

  Qmgr: set queue <queue name> route_retry_time = <value>

- 默认情况下，job在route队列中的存在时间不受限制，可通过下面的方式来改变这一行为：

  Qmgr: set queue <queue name> route_lifetime = <value>

5. 作业调度使用说明

5.1 PBS的基本命令

命令	作用
nqs2pbs	将NQS脚本转换成PBS脚本
mpiexec	通过PBS运行MPI程序
pbs_rsub	提交一个预约，以在预约的时间预留申请的资源给作业
pbs_rdel	删除一个预约
pbs_rstat	显示预约的状态
pbs_password	用于配置或更新pbs 用户的密码
pbs_python	用于从命令行调试hook脚本的Python解释器
pbsdsh	分布式shell工具，用于在一个或多个vnode节点执行指定命令或程序
qalter	用于修改已提交的job
qdel	用于删除已提交的job
qhold	用于挂起已提交的job
qmove	用于将已提交的job从所在队列移动到指定的队列
qmsg	发送信息（message）到指定的job
qorder	交换两个或多个作业在队列中的位置
qrls	删除或释放挂起的作业
qselect	根据指定的条件筛选作业

qsig	发送信号给指定的作业
qstat	显示作业、队列或PBS Server的状态
qsub	用于提交作业
tracejob	打印作业的详细日志信息
pbs_hostn	根据主机名打印详细的主机名、IP地址
pbs_migrate_users	将pbs的用户或Server密码迁移到其他的Server中
pbs_probe	打印对故障的诊断信息
pbs_tclsh	一个封装了PBS API的TCL shell
pbsnodes	管理PBS的vnode
printjob	打印作业的详细信息
qdisable	用于禁用指定的PBS队列
qenable	用于启用指定的PBS队列
qmgr	PBS的命令行主要管理工具
qrerun	qrerun会先杀死指定的作业进程，然后重新提交到执行队列
qrun	无视优先级以及资源需求，手动强制作业运行
qstart	开始一个队列
qstop	停止一个队列
qterm	终止PBS server运行

5.2 相关术语

- 块（**Chunk**）

块是一个单元分配给作业的一组资源。一个块内的所有部分都来自相同的主机。在典型的MPI作业中，每个MPI进程有一个块。

- **块级别资源（Chunk-level resource）**

块级别的资源是按作业分配的单元来进行划分的，一个块资源是应用于该块中正在运行作业的那部分资源。块级别的资源可以使用

qsub -l select=[N:][<chunk specification>][+[N:]<chunk specification>] 这样的select表达式中指定，也可以在作业提交脚本中以

#PBS -l select=[N:][<chunk specification>][+[N:]<chunk specification>] 语句类似的PBS指令来指定。

- **主机级别资源（host-level resource）**

主机级可用的资源，是以主机来划分的，例如一台主机中的cpu、内存、gpu、运行软件、磁盘空间等。主机级别资源只能作为块级资源被请求。

- **作业级别资源（Job-wide resource），服务器资源（server resource），队列资源（queue resource）**

作业级别资源、服务器资源以及队列资源在PBS Pro中实际是同一种资源，是服务器或队列上的整个作业都可以使用的资源。通常作业级别的资源会是可供整个作业使用，并且在服务器或队列上可用，但在主机级别上不可用的资源，例如某些不绑定到特定vnode的资源、cput和wall-time等。

用户请求作业级别资源时，可以使用qsub -l <resource1>=<value1>,<resource2>=<value2>的形式或在pbs提交脚本中以PBS指令#PBS -l <resource1>=<value1>,<resource2>=<value2>的形式请求。

用户请求的作业级别资源必须为可消费的（consumed），并且server或queue上要存在resources_available.<resource >属性匹配所请求的属性值，PBS server才能分配资源给请求的作业，否则会返回资源不满足的提示。

5.3 使用qsub提交作业

PBS的作业可以是一条或多条命令的集合，也可以是要执行的应用程序（applications）。PBS的作业可以运行在一个节点上，同时也可以运行在多个节点上。

在PBS系统中，用户使用qsub命令提交作业。以下是使用qsub命令提交作业的一个简单例子：

该作业需要请求400MB的运行内存，4个逻辑CPU数，计划运行时间在1小时内，运行的实际应用程序是vasp_std。

1) 先编辑一个提交作业文件，假设命名为run_vasp.sh，内容如下所示：

```
#!/bin/sh
#PBS -l walltime=1:00:00
#PBS -l mem=400mb,ncpus=4
source /etc/profile &> /dev/null
source /etc/profile.d/*.sh &> /dev/null
vasp_std &> out
```

注意，#PBS开头的行并不像普通的Shell脚本一样表示注释，在PBS系统中，#PBS开始的行相当于一行PBS的相关指令。

2) 使用qsub命令提交作业：

qsub run_vasp.sh

3) 提交作业后，PBS会返回提交的作业ID，作业的ID通常是以下的格式：

<sequence number>.<server name>

如： 1.mgt

4) 作业文件除了可以是Shell脚本外，还可以是python脚本、Perl脚本，如：

```
#!/usr/bin/python

#PBS -l select=1:ncpus=3:mem=1gb

#PBS -N HelloJob

print “Hello”
```

5.3.1 qsub命令使用方法

qsub命令的使用方法如下：

qsub [-a date_time] [-A account_string] [-c interval] [-C directive_prefix] [-e path] [-f] [-h] [-I [-G [-- <GUI application>]] | [-X]] [-j join] [-J range] [-k keep] [-l resource_list] [-m mail_events] [-M user_list] [-N name] [-o path] [-p priority] [-P project] [-q destination] [-r c] [-R remove] [-S path_list] [-u user_list] [-v variable_list] [-V] [-W additional_attributes] [-z] [script | -- executable [arglist for executable]]

qsub --version

qsub命令最后如果没有提交脚本或者可执行文件（或可执行文件列表）时，表示要从标准输入获取提交脚本名

常用参数及其作用：

选项	作用
-a	后接时间，格式为[[[[CC]YY]MM]DD]hhmm[.SS]。当使用该选项提交作业时，当指定时间还没到时，作业将等到指定的时间才会执行（作业状态为W）
-e	指定错误输出文件名，格式为[hostname:]path_home。Hostname是返回错误输出文件的主机名，path_home是错误输出文件 的绝对路径，如果指定了相对路径，则相对用户的主目录。不使用该选项时，默认是“job_name.e<sequence number>”文件

-f	在前台运行qsub命令。如果要运行一个很短时间的作业时，可以使用该选项
-o	指定输出文件名，格式为[hostname:]path_home。缺省值是在用户主目录下，以“job_name.o<sequence number>”命名的文件
-h	提交作业后先挂起
-I	指定作业以交互方式运行
-j	指定合并错误输出和实际输出。如果指定’oe’，则合并到标准输出文件中；如果指定’eo’，则合并到标准错误输出文件中
-k	指定执行主机是否保留错误输出和实际输出。 如果指定‘o’,则仅保留标准输出； 如果指定’e’，则仅保留标准错误输出； 如果指定’oe’或‘eo’,则保留标准输出和标准错误输出； 如果指定’n’,则不保留任何输出
-l	指定作业所需要的资源，如果不设置，则无限制。
-m	定义何时给用户发送有关作业的邮件。可设定的选项有： n 不发送邮件； a 当作业被批处理系统中断时，发送邮件； b 当作业开始执行时，发送邮件； e 当作业执行结束时，发送邮件
-M	指定发送有关作业信息的邮件用户列表。格式为user[@host][,user@[host],…]缺省值为提交作业的用户

-N	指定作业的名字。缺省值为脚本的名字，如果没有指定脚本，则为STDIN
-p	指定作业的优先级，优先级的范围是[-1024, +1023]。缺省值是0，即没有优先级
-q	指定作业提交到哪个队列。队列名是<queue name>@<server name>的形式，server name>可以省略，表示提交到默认server中。 如果不指定该选项，表示提交到默认server的默认队列中
-r	指定作业是否可重新运行。 指定‘y’时，作业可以重新运行； 指定‘n’时，作业不能重新运行。 默认值为’n’。
-v	后面接上一个变量列表，格式为variable1,variable2,…或variable1=value,variable2=value的形式，表示将指定的变量及其值传递到作业中
-V	指定qsub命令的所有的环境变量都传递到批处理作业中

5.3.2 指定作业脚本的Shell

如果用户没有指定作业使用的默认Shell，PBS系统会默认使用/bin/sh。

用户可以选择任何类型的Shell，只要指定的Shell在当前环境下可以使用，也包括一些交互式的Shell，如Python、Perl。

要指定使用这些非默认的Shell，可以在提交脚本的第一行指定即可，如：

```
#!/usr/bin/perl
```

或：

```
#!/usr/bin/python
```

也可以通过qsub的-S选项来指定作业的脚本Shell，如：

qsub -S /bin/bash@mgt,/usr/bin/bash@cn1

5.3.3 传递环境变量到作业中

PBS的使用中，经常会遇到需要传递环境到作业脚本的情况，PBS也提供了可以传递环境变量到作业的多种方法。

假设需要在作业脚本中使用到p4vasp的相关python模块，假设都被安装在共享的目录/opt/ohpc/pub/apps/p4vasp/site-packages/中，那么需要传递PYTHONPATH环境变量至作业中，可以按以下几种方式进行操作：

◆ 在作业脚本中直接指定

1) 编辑作业提交脚本:

```
#!/bin/sh

#PBS -l walltime=1:00:00

#PBS -l mem=400mb,ncpus=4

source /etc/profile &> /dev/null

source /etc/profile.d/*.sh &> /dev/null

export PYTHONPATH=$PYTHONPATH: /opt/ohpc/pub/apps/p4vasp/site-packages/

cd $PBS_O_WORKDIR

python test.py
```

2) 使用qsub提交作业，假设作业脚本名为testP4vasp.sh：

qsub testP4vasp.sh

◆ 通过环境变量传递

1) 定义PYTHONPATH环境变量：

export PYTHONPATH=$PYTHONPATH: /opt/ohpc/pub/apps/p4vasp/site-packages/

2) 在qsub命令提交作业时，使用-v或-V参数。

qsub -v PYTHONPATH testP4vasp.sh

或：

qsub -V testP4vasp.sh

qsub的-V选项可以将当前提交用户的所有环境变量都会传递到作业中。

5.3.4 传递作业参数

有时候作业会需要传递参数给作业，尤其是当要运行的是一个可执行文件时。PBS Pro提供了以下的方式来直接在命令行指定作业的运行参数：

qsub [<options>] -- <executable> [<arguments to executable>]

其中，<executable>为可以执行文件。

例如，作业testjob的应用程序app1需要传递参数a、b，那么可以运行以下的qsub命令：

qsub -N testjob -- app1 a b

qsub命令中使用环境变量同样也可以生效。如，可执行文件a.out的运行需要1个参数作为计算的主要数据源，以下的操作方式PBS Pro也支持：

export INFILE=/data/data.in

qsub -- a.out $INFILE

5.3.5 请求资源

■ 请求作业级别资源

请求作业级别的资源可以在qsub命令或提交脚本的#PBS指令中以下面的格式指定：

-l <resource name>=<value>[,<resource name>=<value> ...]

如，请求运行作业的时间是1小时，可用以下命令提交作业testjob.sh：

qsub -l walltime=1:00:00 testjob.sh

需要注意的是，请求作业级别的资源时，请求的资源列表可以在-l后，但不能跟在-l select后，因为-l select是用于请求块级别的资源的。如果在提交作业时既要请求作业级别资源又要请求块级别的资源时，可以接上多个-l参数，如：

qsub -l walltime=1:00:00 -l select=2:ncpus=8 testjob.sh

■ 请求块级别资源

块资源的请求可以在qsub命令或提交脚本的#PBS指令中以下面的格式指定：

-l select=[N:]<chunk>[+[N:]<chunk> ...]

如果是只请求一个块的话，可写成以下方式：

-l select=<resource name>=<value>[:<resource name>=<value>...]

例如，请求6个块，每个块中有2个逻辑CPU以及4gb内存可以写成以下格式：

-l select=6:ncpus=2:mem=4gb

如果想请求多套不同的块集，可以按以下方式请求，以“+”来连接：

-l select=[<number of chunks>]<chunk specification>+[<number of chunks>]<chunk specification>

如：***-l select=6:ncpus=2:mem=4gb+3:ncpus=8:mem=4GB***

需要注意的是，必须要在一个select语句中指定所有的块，要注意**select语句中不要有空格**。

5.3.6 请求GPU资源

要在作业中请求GPU资源，前提必须是PBS集群中存在主机有GPU，并且PBS管理员已经正确配置了GPU资源。需要注意的是，PBS Pro并不提供将作业绑定到指定GPU资源的功能，这需要依赖于应用程序本身或CUDA库来进行绑定。

请求GPU资源可以按照以下方式提交作业：

```
qsub -lselect=1:ncpus=1:ngpus=2 -lplace=shared gpu_job.sh
```

或：

```
qsub -lselect=1:ncpus=1:ngpus=2 -lplace=excl gpu_job.sh
```

区别是，-lplace=shared可以与其他作业共用分配的块资源，而-lplace=excl请求的资源则不会与其他作业共用分配的块资源。

如果PBS管理员在配置vnode的GPU资源时，配置了gpu_id资源，那么也可以在请求时按gpu_id请求GPU资源。

```
qsub -lselect=4:ncpus=1:gpu_id=gpu0 gpu_job.sh
```

5.3.7 使用qsub请求资源的一些例子

- 提交一个10路的mpi作业，每个MPI task上分配一个块并且指定每个块分配1CPUS和2gb内存：

  -l select=10:ncpus=1:mem=2gb

- 请求将分配3个块，每个块有1个CPU和10GB内存，并且计划需要100MB的临时存储空间：

  -l scratch=100mb -l select=3:ncpus=1:mem=10GB

- 请求4个块，每个块需要4个CPU和2GB内存，并且要求每台主机只分配一个块：

  -lselect=4:mem=2GB:ncpus=4:arch=linux -lplace=scatter

- 请求在主机名为cn1的节点上，分配24CPUS和50GB内存：

  -l select=1:ncpus=24:mem=50gb:host=cn1

- 请求分配2个3CPUS6gb内存的块资源，并且要求2个块集中在一台主机上，一旦分配后，节点专用：

  -l select=1:ncpus=3:mem=6gb-l place=pack:excl

- 向cn1主机请求4个块，每个块分配4cpus和10gb内存，向cn2主机请求6个块，每个块分配4cpus和10gb内存：

  -l select=4:ncpus=4:mem=10gb:host=cn1+6:ncpus=4:mem=10gb:host=cn2

- 请求一个mpi作业，分配4个块，每个块分配2CPUS，并且每个块中运行2个MPI进程：

  -lselect=4:ncpus=2:mpiprocs=2

需要注意的是，以上的请求如果用旧版本的pbs请求格式是-lnodes=4:ppn=2:ncpus=1，但PBS Pro采用的基于块资源的请求方式，如果使用旧的语法提交作业可能会出现作业分配资源不匹配请求或提交失败的情况。并且PBS_NODEFILE环境变量的行数等于块数*mpiprocs的值，即：mpiprocs=N，则该块所在的主机名会在PBS_NODEFILE中出现N次。

同样，cpp的qsub提交语法也已经被ncpus语法代替，在使用时需要注意。

使用PBS请求资源运行MPI程序时，还可以指定MPI进程共享cpu来提升性能。例如，使用4个CPU和4个MPI进程请求一个块与使用1个CPU和1个MPI进程请求4个块是不同的。在第一种情况下，所有四个MPI进程共享所有四个CPU。在第二种情况下，每个进程都有自己的CPU。

- 需要请求3个块资源，第一个块需要2CPUS和20GB内存，第二个块需要4CPUS和100GB内存，第三个块需要1CPUS和5GB内存：

-lselect=1:ncpus=2:mem=20gb+ncpus=4:mem=100gb+mem=5gb

- 使用脚本提交作业

  Linux shell脚本：

  qsub <name of shell script>

  Linux Python或Perl脚本：

  qsub <name of Python or Perl job script>

- 在提交job时指定作业组ID：

  qsub -W group_list=<group list>

  #PBS group_list=<group list>

  举例:

  qsub -W group_list=grpA,grpB@jupiter my_job

- 提交到指定的队列以及服务器

  将job提交到指定队列可以使用以下的命令格式：

  qsub -q <queue name>[@<server name>]

  其中，[ ]扩住的部分为非必须内容，指定提交到的pbs server，不指定的话提交到默认的pbs server中。

  脚本中则可以使用下面的格式：

  #PBS -q <queue name>[@<server name>]

  同样，[ ]扩住的部分为非必须内容，指定提交到的pbs server，不指定的话提交到

默认的pbs server中。

PBS管理员可以将作业从一个队列移动到另一个队列。您可以使用qstat <job ID >查看哪个队列具有作业。作业的队列属性包含作业驻留的队列的名称。

举例：

qsub -q queue my_job

qsub -q @server my_job

#PBS -q queue1

qsub -q queue1@myserver my_job

qsub -q queue1@myserver.mydomain.com my_job

- 指定使用Intel mpi

#PBS -l select=1:ncpus=1:host=hostA+1:ncpus=2:host=hostB

$INTEL_MPI_HOME/sbin/mpirun -np 3 /tmp/mytask

输出结果会在文件hostA和hostB当中

- 指定mpi使用的线程数

例：用64个MPI进程运行MPI应用程序，每个进程运行一个线程：

#PBS -l select=64:ncpus=1

mpiexec -n 64 ./a.out

例：使用64个MPI进程运行MPI应用程序，每个进程运行四个OpenMP线程：

#PBS -l select=64:ncpus=4

mpiexec -n 64 omplace -nt 4 ./a.out

或

#PBS -l select=64:ncpus=4:ompthreads=4

mpiexec -n 64 omplace -nt 4 ./a.out

5..4 通过PBS使用MPI

当作业在未与PBS集成的MPI下运行时，PBS仅限于管理主vnode上的作业，运行作业的资源跟踪、发信号给作业等功能都只局限在主vnode上的进程。一些集成的MPIs有稍微不同的命令行。

5.4.1 pbs_tmrsh命令

pbs_tmrsh命令用于集成mpi环境。

pbs_tmrsh命令是PBS模拟的rsh，利用PBS_TM接口直接与其他vnodes上的pbs_mom对话。pbs_tmrsh命令通知主mom和其他mom进行作业过程通讯。当作业使用pbs_tmrsh时，PBS可以跟踪所有作业进程的资源使用情况。

需要注意的是，有些应用程序将临时文件写入临时位置。PBS为此提供了一个临时目录，并将路径放在PBS_TMPDIR环境变量中。临时目录的位置与主机相关。如果您使用的MPI不是LAM MPI或Open MPI，并且您的应用程序需要scatch空间，那么作业的临时目录应该跨执行主机保持一致。PBS管理员可以配置$tmpdir 的MoM参数为每个主机上的临时目录指定根目录。在这种情况下，PBS_TMPDIR环境变量最好配置为全路径。注意，不要尝试在作业中配置修改PBS_TMPDIR。

PBS Pro注意通过变量的方式来集成各种MPI，但是集成的方式各不相同，注意某些类型的较新版本的MPI可能会不支持，同时某些类型旧版本的MPI支持可能也会在新版本的PBS Pro中移除，详细的信息请查看PBS Pro官方资料或相应MPI的官方资料。

5.4.2 PBS集成Intel MPI

PBS Pro集成Intel mpi的首要前提是，当前集群的Intel mpi环境配置正确，并能独立于PBS之外并行使用MPI程序。

PBS Pro集成intel mpi的步骤如下：

1) 环境变量指定rsh:

export I_MPI_HYDRA_BOOTSTRAP=rsh

2) 环境变量指定pbs_tmrsh:

export I_MPI_HYDRA_BOOTSTRAP_EXEC=/opt/pbs /bin/pbs_tmrsh

以上的环境变量配置也可以写在用户的profile文件或bashrc文件中。

5.4.3 PBS使用Intel MPI注意事项

PBS中使用Intel MPI需在qsub中明确指定select的数量，同时mpirun -np数值要与qsub -l select表达式中的块数值相同，否则将返回"too few entries in the machinefile"的类似错误。

PBS Pro中使用Intel mpirun的话，以下的一些mpirun选项会受到影响：

选项	说明
-host, -ghost	使用mpirun该参数指定执行的主机会被PBS Pro忽略
-machinefile	使用该参数会被PBS Pro忽略，并使用$PBS_NODEFILE指定的文件内容代替
mpdboot --totalnum=*	使用该参数会被PBS Pro忽略，并使用$PBS_NODEFILE指定的文件内容指定的主机数（去除重复主机）
mpdboot -f ,--file=*	使用该参数会被PBS Pro忽略，并以$PBS_NODEFILE值代替
-s	不支持使用该参数选项

-np	如果mpirun没有指定-np选项，则PBS Pro不提供默认值，由mpirun决定合理的值应该是什么，通常是1。 可以启动的最大数是$PBS_NODEFILE中的行数。

5.4.4 PBS使用Intel MPI示例

- 本示例运行一个Intel MPI作业，6个MPI进程分布在由PBS Pro分配的在$PBS_NODEFILE文件中列出的主机上：

提交脚本job.script内容大致如下：

```
#!/bin/bash

mpirun /path/to/app1 1200
```

qsub提交命令如下：

qsub -l select=3:ncpus=2:mpiprocs=2 job.script

- 本示例要运行的作业是需要运行多个可执行文件的Intel MPI作业

提交脚本job.script内容大致如下：

```
#!/bin/bash

mpirun -np 2 /tmp/mpitest1 : -np 2 /tmp/mpitest2 : -np 2 /tmp/mpitest3
```

qsub提交命令如下：

qsub -l select=3:ncpus=2:mpiprocs=2 job.script

PBS Pro分配的$PBS_NODEFILE文件内容大致会如下：

```
cn1

cn1
```

```
cn2

cn2

cn3

cn3
```

根据提交请求，PBS Pro将会cn1节点上运行mpitest1的2个实例，在cn2节点上运行mpitest2的2个实例，在cn3上运行mpitest3的2个实例。

5.5 查询队列信息

查询服务器上所有的队列信息可以使用以下的命令格式：

qmgr -c 'print server'

命令也可以简写成：

qmgr -c 'p s'

执行结果信息通常如下所示：

```
[root@mgt ~]# qmgr -c 'p s'

# Create and define queue workq

create queue workq

set queue workq queue_type = Execution

set queue workq enabled = True

set queue workq started = True

# Set server attributes.

set server scheduling = True

set server default_queue = workq
```

```
set server log_events = 511

set server mail_from = adm

set server query_other_jobs = True

set server resources_default.ncpus = 1

set server resources_default.place = scatter

set server default_chunk.ncpus = 1

set server scheduler_iteration = 600

set server resv_enable = True

set server node_fail_requeue = 310

set server max_array_size = 10000

set server default_qsub_arguments = -V

set server pbs_license_min = 0

set server pbs_license_max = 2147483647

set server pbs_license_linger_time = 31536000

set server eligible_time_enable = False

set server job_history_enable = True

set server max_concurrent_provision = 5
```

其中# Create and define queue <queue name>字段为定义好的<queue name>队列的详细信息。

另外，也可以使用qmgr -c 'print queue <queue name>'命令只打印指定的队列信息，其中<queue name>为队列名。

```
Qmgr: print queue workq
#
# Create queues and set their attributes.
#
#
# Create and define queue workq
#
create queue workq
set queue workq queue_type = Execution
set queue workq enabled = True
set queue workq started = True
Qmgr: _
```

5.6 查询作业状态

PBS中，通常使用qstat来查询作业的状态信息。

qstat命令的用法如下所示：

qstat [-f] [-J] [-p] [-t] [-x] [-E] [-F format] [-D delim] [job_identifier... | destination...]

qstat [-a|-i|-r|-H|-T] [-J] [-t] [-u user] [-n] [-s] [-G|-M] [-1] [-w]

[job_identifier... | destination...]

qstat -Q [-f] [-F format] [-D delim] [destination...]

qstat -q [-G|-M] [destination...]

qstat -B [-f] [-F format] [-D delim] [server_name...]

qstat --version

qstat命令的常用选项及其作用描述如下所示：

选项	作用
-a	列出所有队列中正在运行的作业
-f	详细显示分配给作业的资源情况
-Q	以默认格式打印各队列的状态
-q	以备用格式打印各队列的状态（包含内存、CPU time、Walltime信息）
-G	以G为单位显示备用打印格式
-M	以M为单位显示备用打印格式
-i	列出不在运行的作业
-n	列出分配给此作业的结点
-B	列出PBS服务器的相关信息
-r	列出正在运行的作业
-u	列出指定用户提交的作业状态

5.6.1 作业的状态

在PBS Pro中，作业主要有如下的几种状态：

状态	意义
* B	表示作业已经开始执行

* E	作业在运行后退出
* F	作业已完成
* H	作业被服务器或用户或者管理员阻塞
* M	作业已经被移动到另一个pbs server
* Q	作业正在排队中，等待被调度运行
* R	作业正在运行
* S	作业被服务器挂起，由于一个更高优先级的作业需要当前作业的资源
* T	作业被转移到其它计算节点了
* U	由于服务器繁忙，作业被挂起
* W	作业在等待状态，所请求的执行时间还没到
* X	只用于子作业，表示子作业完成

5.6.2 使用qstat查看状态

- 查询所有队列上作业的状态可以使用以下的命令：

  qstat -an

```
[root@mgt ~]# qstat -an

mgt:
                                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
574.mgt         manager  workq    test.sh     13033   1   1    --    --  R 00:00
   cn1/0
```

- 查看队列资源状态

qstat -Qf

```
[root@mgt ~]# qstat -Qf
Queue: workq
    queue_type = Execution
    total_jobs = 0
    state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0 Begun
        :0
    resources_assigned.mpiprocs = 0
    resources_assigned.ncpus = 0
    resources_assigned.nodect = 0
    enabled = True
    started = True
```

- 查看分配给正在运行作业的详细情况

qstat -f

```
[root@mgt ~]# qstat -f
Job Id: 576.mgt
    Job_Name = test.sh
    Job_Owner = manager@mgt
    resources_used.cpupercent = 0
    resources_used.cput = 00:00:00
    resources_used.mem = 0kb
    resources_used.ncpus = 4
    resources_used.vmem = 0kb
    resources_used.walltime = 00:00:00
    job_state = R
    queue = workq
    server = mgt
    Checkpoint = u
    ctime = Tue Jan 15 15:04:54 2019
    Error_Path = mgt:/home/manager/test.sh.e576
    exec_host = cn1/0*4
    exec_vnode = (cn1:mem=409600kb:ncpus=4)
    Hold_Types = n
    Join_Path = n
    Keep_Files = n
    Mail_Points = a
    mtime = Tue Jan 15 15:04:54 2019
    Output_Path = mgt:/home/manager/test.sh.o576
    Priority = 0
    qtime = Tue Jan 15 15:04:54 2019
    Rerunable = True
    Resource_List.mem = 400mb
    Resource_List.ncpus = 4
    Resource_List.nodect = 1
    Resource_List.place = pack
    Resource_List.select = 1:mem=400mb:ncpus=4
    Resource_List.walltime = 00:10:00
    schedselect = 1:mem=400mb:ncpus=4
    stime = Tue Jan 15 15:04:54 2019
    session_id = 13170
    jobdir = /home/manager
    substate = 42
    Variable_List = PBS_O_HOME=/home/manager,PBS_O_LANG=en_US.UTF-8,
        PBS_O_LOGNAME=manager,
        PBS_O_PATH=/opt/ohpc/pub/apps/phonopy/bin:/opt/ohpc/pub/apps/intel/com
        pilers_and_libraries_2019/linux/bin:/opt/ohpc/pub/apps/intel/compilers_
        and_libraries_2019.1.144/linux/bin/intel64:/opt/ohpc/pub/apps/intel/com
        pilers_and_libraries_2019.1.144/linux/mpi/intel64/libfabric/bin:/opt/oh
        pc/pub/apps/intel/compilers_and_libraries_2019.1.144/linux/mpi/intel64/
        bin:/opt/ohpc/pub/apps/intel/compilers_and_libraries_2019.1.144/linux/m
        pi/intel64/libfabric/bin:/opt/ohpc/pub/apps/intel/compilers_and_librari
        es_2019.1.144/linux/mpi/intel64/bin:/opt/ohpc/pub/apps/wien2k:/opt/ohpc
        /pub/apps/wien2k/SRC_structeditor/bin:/opt/ohpc/pub/apps/wien2k/SRC_IRe
```

- 查询PBS server的资源状态：

  qstat -fB

```
[root@mgt ~]# qstat -fB
Server: mgt
    server_state = Active
    server_host = mgt
    scheduling = True
    total_jobs = 0
    state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:0 Exiting:0 Begun
        :0
    default_queue = workq
    log_events = 511
    mail_from = adm
    query_other_jobs = True
    resources_default.ncpus = 1
    default_chunk.ncpus = 1
    resources_assigned.mpiprocs = 0
    resources_assigned.ncpus = 0
    resources_assigned.nodect = 0
    scheduler_iteration = 600
    FLicenses = 20000000
    resv_enable = True
    node_fail_requeue = 310
    max_array_size = 10000
    pbs_license_min = 0
    pbs_license_max = 2147483647
    pbs_license_linger_time = 31536000
    license_count = Avail_Global:10000000 Avail_Local:10000000 Used:0 High_Use:
        0
    pbs_version = 18.1.2
    eligible_time_enable = False
    max_concurrent_provision = 5
    power_provisioning = False
```

- 若要查看Vnode资源，请使用以下任一项：

  qmgr -c "print node <vnode name> [<attribute name>]"

  其中，<attribute name>为可选项。

```
Max open servers: 49
Qmgr:
Qmgr: print node pgn1[0]
#
# Create nodes and set their properties.
#
#
# Create and define node pgn1[0]
#
create node pgn1[0] Mom=pgn1
set node pgn1[0] state = free
set node pgn1[0] resources_available.arch = linux
set node pgn1[0] resources_available.gpu_id = gpu0
set node pgn1[0] resources_available.host = pgn1
set node pgn1[0] resources_available.mem = 7gb
set node pgn1[0] resources_available.ncpus = 4
set node pgn1[0] resources_available.ngpus = 1
set node pgn1[0] resources_available.vnode = pgn1[0]
set node pgn1[0] resv_enable = True
set node pgn1[0] in_multivnode_host = 1
Qmgr:
```

可以使用以下的命令来查询目前服务器上配置的vnode及其资源：

pbsnodes -av

```
pgn1[0]
     Mom = pgn1
     Port = 15002
     pbs_version = 18.1.2
     ntype = PBS
     state = free
     pcpus = 1
     jobs = 16060.pmgt/0
     resources_available.arch = linux
     resources_available.gpu_id = gpu0
     resources_available.host = pgn1
     resources_available.mem = 7gb
     resources_available.ncpus = 4
     resources_available.ngpus = 1
     resources_available.vnode = pgn1[0]
     resources_assigned.accelerator_memory = 0kb
     resources_assigned.hbmem = 0kb
     resources_assigned.mem = 0kb
     resources_assigned.naccelerators = 0
     resources_assigned.ncpus = 1
     resources_assigned.ngpus = 1
     resources_assigned.vmem = 0kb
     resv_enable = True
     sharing = default_shared
     in_multivnode_host = 1
     last_state_change_time = Wed Dec  5 14:03:48 2018
     last_used_time = Mon Jan 14 18:23:28 2019
```

5.7 删除作业

删除作业命令：

qdel <job ID>

注意事项：

1) 非root用户只能查看、删除自己提交的作业；

2) 强制删除作业，当某些作业无法删除时，可由root用户登录，使用qdel -p <job ID>来强制删除作业。

5.8 设置作业优先级

可以在作业提交时指定作业的优先级，在-p参数后附带优先级即可。例如：

qsub -p 120 job.sh

在脚本里可以用以下方式指定：

#PBS -p 300

优先级高的作业会优先运行。

5.8.1 修改作业在队列中的顺序

作业被提交到队列后，则需要使用qorder命令来修改作业在队列中的顺序。

qorder <job ID>1 <job ID2>

qorder命令会交换指定的2个作业在队列中的顺序，需要注意两个作业都必须在同一个PBS server中。

附录：《PBS Pro提交脚本的内置变量》

变量名	含义	备注
NCPUS	作业所在vnode的总线程数	
OMP_NUM_THREADS	同NCPUS	
PBS_ARRAY_ID	job array的唯一标识符	
PBS_ARRAY_INDEX	作业在job array中的索引值	
PBS_CONF_FILE	pbs.conf配置文件的所在路径	
PBS_CPUSET_DEDICATED	声明分配的cpuset为独占使用	由mpiexec生成
PBS_DEFAULT	默认的PBS server主机名	
PBS_DATA_SERVICE_USER	用于data服务的用户账号	安装时指定，默认为Admin
PBS_ENVIRONMENT	批处理作业为 PBS_BATCH， 交互式作业为 PBS_INTERACTIVE	
PBS_JOBCOOKIE	用于作业内部多个MoM间通信的惟一标识符	
PBS_JOBDIR	分配给作业执行的指定目录路径	
PBS_JOBID	作业或job array的唯一标识	由pbs server生成
PBS_JOBNAME	用户指定的作业名称	

PBS_MOMPORT	PBS MOM进程的监听端口	
PBS_NODEFILE	包含分配给该作业的所有节点的文件名，多个节点间以行分隔	
PBS_NODENUM	分配给该作业的vnode数量	
PBS_O_HOME	该作业提交时环境变量HOME的值	作业提交时决定
PBS_O_HOST	执行qsub命令所在主机的主机名	作业提交时决定
PBS_O_LANG	该作业提交时环境变量LANG的值	作业提交时决定
PBS_O_LOGNAME	该作业提交时环境变量LOGNAME的值，登录的用户名称	作业提交时决定
PBS_O_MAIL	该作业提交时环境变量MAIL的值	作业提交时决定
PBS_O_PATH	该作业提交时环境变量PATH的值	作业提交时决定
PBS_O_QUEUE	该作业最初提交到PBS的队列名称	作业提交时决定
PBS_O_SHELL	该作业提交时环境变量SHELL的值	作业提交时决定
PBS_O_SYSTEM	执行qsub命令提交作业时所在主机的操作系统类型	作业提交时决定
PBS_O_TZ	该作业提交时环境变量TZ的值	作业提交时决定
PBS_O_WORKDIR	执行qsub命令提交该作业时所在目录的绝对路径	作业提交时决定
PBS_QUEUE	该作业真正被执行所在的队列	

PBS_SERVER	默认的PBS server主机名	
PBS_TASKNUM	所在vnode上分配给该作业的task数（或进程数）	
PBS_TMPDIR	该作业的临时目录路径	