Tesla GPU 集群服务器使用手册

v0.93 (2010-1-11)

一、系统环境简介

Tesla GPU 集群服务器域名为 tesla.sccas.cn，IP 地址为 159.226.49.76（暂定），内部用户可以在办公网内直接使用 SSH 登录该集群，外部用户同样需要经过防火墙身份认证之后（认证过程请参见《深腾 7000 远程登录指南》）再进行 SSH 登录。数据的上传与下载仍然是通过 scp 或者 sftp 方式进行。Tesla GPU 集群服务器的系统环境如下：

1）硬件环境：

● 头节点 1 个，机器名 console，配备一颗 Intel Xeon E5504 四核处理器，2.0GHz 主频，2*4MB 缓存，8G 内存，6 块 300GB SAS 硬盘，工作于 Raid5 模式。

● 计算节点 90 个，存在两种不同硬件配置，其机器名分别如下：

c0101-c0110、c0201-c0203、c0301-c0305（共 18 个节点）

以上节点配置一颗 AMD Phenom 9850 四核处理器，2.5GHz 主频，4*256KB 二级缓存，4MB 三级缓存，3 块 Tesla C1060 GPU 处理器，8GB 内存，一块 500GB SATA 硬盘。

c0204-c0233、c0401-0442（共 72 个节点）

以上节点配置一颗 Intel Xeon E5410 四核处理器，2.33GHz 主频，2*6MB 二级缓存，2 块 Tesla C1060 GPU 处理器，8GB 内存，一块 500GB SATA 硬盘。

各节点间通过 DDR 4X Infiniband 高速网络和千兆以太网进行连接，分别用于计算数据和系统管理信息的通讯。

2）软件环境：

RHEL 5.3 x64 操作系统，内核版本 2.6.18-128.el5。

GNU C/C++/Fortran 编译器。

Nvidia CUDA Toolkit 2.1 开发工具。

Mvapich/OpenMPI 并行编程环境。

Atlas/GotoBlas 数学函数库。

Torque/Maui 资源管理系统及作业调度器。

Ganglia 集群监控系统。

3）文件系统：

除 console 外所有节点通过 NFS 挂载 console 的/export 目录，包括 console 在内的所有节点的/home 目录为/export/home 目录的软链接。由于文件系统性能不高，建议不要在该集群上运行会造成大规模并行或复杂 I/O 的应用程序。

二、程序开发及调试环境

1）基本编译环境

目前在 Tesla GPU 集群上各节点均提供用于编译通用程序代码的 GCC 编译器，能够编译 C／C++／Fortran 程序，对应的命令为 gcc/g++/gfortran，该编译器为系统默认安装，安装路径位于/usr 下。

另外在所有计算节点（不包括 console）可以使用 Nvidia CUDA Toolkit 开发工具包提供的用于编译 CUDA GPU 加速程序 CUDA 编译器，对应的命令为 nvcc。Nvidia CUDA Toolkit 安装在/export/cuda 下，在计算节点上不用再另外设置环境变量，console 上无法使用。

2）并行程序编译环境

Tesla GPU 集群上安装了两套开源 MPI 编译并行环境，即 Mvapich 和 OpenMPI，用户需要在自己主目录下建立名为.mpi_type 文件（该文件为隐藏文件），在文件中指明使用哪套并行环境。如果该文件不存在或不合法，系统将默认使用 Mvapich 并行编译环境。

~/.mpi_type 文件示例如下：

```
#Here we set openmpi environment.
MPITYPE=”openmpi”
```

（注：修改完.mpi_type 文件之后，需要退出并重新登录系统才能使设置生效）

要查看当前并行编译环境设置是否生效，可以简单的执行“which mpicc”并通过返回信息中的路径来进行判定。

三、作业提交运行

Tesla GPU 集群目前安装的是 Torque 资源管理系统和 Maui 作业调度器。Torque 是著名的开源软件 OpenPBS 的后续开源版本（PBS Pro 是 OpenPBS 的商业化版本），命令基本兼容于 OpenPBS 和 PBS Pro。Maui 则是一套通用的集群作业调度器，可以结合各种资源管理系统进行安装，并为集群实现复杂的可配置的作业调度功能。

Torque 资源管理系统和 Maui 调度器的官方网站是：

http://www.clusterresources.com/pages/products/torque-resource-manager.php

http://www.clusterresources.com/products/maui-cluster-scheduler.php

在上面的网站上可以下载这两个软件的源码和配置使用文档。如需进一步了解，可自行进行下载和阅读学习。

下面简单说明如何在 Tesla GPU 集群上使用 Torque 提交运行作业：

1）作业脚本

在 Torque 中，作业脚本用来描述运行作业（程序）所需执行的命令和程序，也可以用

来配置该作业的参数（参数一般在提交作业时通过命令行直接指定）。用户通过使用 qusb 提交该作业脚本，使脚本文件中所写的程序和命令得到执行。（与深腾上的 LSF 不同，bsub 直接提交可执行程序的名称，这一点请区分开）

实际上，当作业得到调度执行之后，系统将远程登录到被分配的主计算节点并执行所提交的作业脚本中的内容。特别需要注意的是，与平时正常登录一样，在执行作业时系统自动远程登录计算节点后的初始目录（也就是开始执行作业脚本时的目录）仍然是用户的主目录，而不是用户提交作业时所在的目录。而用户提交作业时所在的目录则被保存为 $PBS_O_WORKDIR 环境变量传递给执行作业的登录进程。因此，当用户提前作业时的工作目录不是用户的主目录时，作业脚本里正式内容的第一句，通常应该是将工作目录切换至之前提交作业的目录，即：

```
cd $PBS_O_WORKDIR
```

对于串行程序和纯 OpenMP 并行程序，作业脚本只需按照的 shell 脚本书写方法，在脚本中调用程序使其执行即可，假设程序名为当前目录的 foo_se，则脚本中可以这么写（文件名请任意指定）：

```
#假设该脚本文件名为 job.sh1
cd $PBS_O_WORKDIR
#如果是 OpenMP 程序，此处先设置 OpenMP 执行变量
./foo_se arg1 arg2 ...
```

对于 MPI 并行程序，在计算化学集群上的 Torque 系统中是通过 mpiexec 软件包使程序得到执行，调用该软件包的过程已经封装在了 mpijob 这个命令脚本中。mpijob 命令默认以 Mvapich 方式执行 MPI 程序。如并行环境配置使用的是 OpenMPI 的话，则需加上相应的-openmpi 参数。如：

```
#假设该脚本文件名为 job.sh2
cd $PBS_O_WORKDIR
#执行 Mvapich 并行程序
mpijob ./foo_mpi_mva arg1 arg2 ...
```

或是：

```
#假设该脚本文件名为 job.sh3
cd $PBS_O_WORKDIR
#执行 OpenMPI 并行程序
mpijob -openmpi ./foo_mpi_open arg1 arg2 ...
```

2）作业提交

对于 Torque 系统，使用 qsub 命令提交作业，最常用的格式如下：

```
qsub -l nodes=X:ppn=Y  -q QUEUE SCRIPT
```

其中 X 代表所需节点数，Y 代表每节点使用 CPU 数，QUEUE 代表队列名，SCRIPT 是

作业脚本名。

Tesla GPU 集群中目前有三个队列，分别如下：

队列名	资源配置	最小规模	默认规模	最大规模	最大时长	默认时长
all	集群中所有计算节点	4x1	4x4@intel	90x4	不超过 7 天	1 天
amd	18 个 AMD 平台计算节点	1x1	1x4	16x4	不超过 7 天	2 天
intel	72 个 Intel 平台计算节点	1x1	1x4	36x4	不超过 7 天	2 天

其中，amd 队列是默认队列，所有没有使用-q 参数指定提交到哪个队列的作业将会被提交到 amd 队列当中。

另外，如果作业提交到 all 队列，Torque 在默认情况下将随机分配空闲节点给作业运行，而不去管该节点是什么平台的处理器。用名可以在提交作业时使用-l 参数并且再加上:amd 或:intel 选项，即-l nodes=X:ppn=Y:intel 或-l nodes=X:ppn=Y:amd，这样就能够指定使用何种平台的计算节点运行作业。如果需要混合使用两种不同平台的节点，可以通过指定-l nodes=X1:ppn=Y1:intel+X2:ppn=Y2:amd，这样系统就会将作业分配 X1 个 intel 节点和 X2 个 amd 节点上运行。

作业提交举例如下（例中 job.sh1、job.sh2、job.sh3 脚本为前面举例所写的脚本）：

qsub -l nodes=1:ppn=1 -q amd job.sh1

（job.sh1 是之前所写串行程序脚本，即使用单个 AMD 平台节点上的单个 CPU 核心执行作业）

qsub -l nodes=2:ppn=4 -q intel job.sh2

（使用 2 个 Intel 平台的计算节点，每节点占用 4 个 CPU 核心，共 8 个核心执行 MPI 并行程序，job.sh2 中指定用 Mvapich 并行环境执行）

qsub -l nodes=4:ppn=4:amd+8:ppn=4:intel,walltime=3:0:0 -q all sh3

（使用 4 个 AMD 平台的计算节点，每节点占用 4 个 CPU 核心，以及 8 个 Intel 平台的计算节点，每节点也占用 4 个 CPU 核心来运行作业，并且设置作业时长为 3 天。这样一共使用了 12x4=48 个 CPU 核心，并且使用 OpenMPI 并行环境执行）

qsub 提交作业后，系统返回'1051.console'类似的输出，其中前面的数字 1051 代表作业号，作业号是 Torque 系统中每个作业所拥有的唯一的代号。

需要提示的是，如果程序执行过程中有标准输入过程（比如需要从键盘输入指定变量），那么建议自行使用重定向方式将其从文件输入。当然，qsub 同样提供-I 参数进行交互式作业提交，具体使用方法在此不再详叙，可参考官方手册自行尝试。

3）作业状态查看

使用 qstat 命令，可以看到系统中所有正在排队和运行的作业，qstat 默认输出类似以下信息：

```
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
1056.console              job.sh1          user1          10:02:03 C amd
1057.console              job.sh2          user2          25:13:27 R intel
1061.console              job.sh3          user4                 0 R all
1062.console              job.sh4          user1                 0 Q intel
```

上述信息分别代表的含义是作业号，作业名（默认为脚本名），用户名，使用 CPU 时间，状态（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），队列名。

如需查看指定作业号的作业，执行：

qstat jobid1 jobid2 ...

jobid1 和 jobid2 代表指定作业号，可以一次查看多个作业。

如需查看指定用户的作业，可以使用参数-u：

qstat -u user1

该方式输出和默认略有不同，但大同小异。

如需查看特定作业详细信息，则应使用-f 参数：

qstat -f jobid

该命令将会输出作业号为 jobid 的作业的详细信息。

4）作业挂起、释放和删除

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；而使用 qdel 命令即可删除作业，不论该作业是否正在运行当中。这些命令的具体格式为：

qhold jobid1 jobid2 ...

qrls jobid1 jobid2 ...

qdel 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）作业的输出结果

作业运行完成或异常退出之后，在用户提交作业的目录下会生成'jobname'.o'jobid'以及'jobname'.e'jobid'两个文件（比如 sh1.o1066 和 sh1.e1066），分别记录作业执行时写往标准输出设备和标准错误输出设备的输出信息，其中 jobname 是作业名，默认则是提

交作业时的作业脚本名，jobid 则是作业号。用户可以通过这两个文件查看和验证程序运行的结果。同时，建议在编写程序时尽量将程序运行结果输出到特定的磁盘文件而不是标准输出（屏幕）。

以上是 Torque 作业资源管理系统常用到的一些基本操作，更加复杂的操作可以参考 Torque 官方网站上提供的文档。另外，如果遇到作业不能正常提交、作业提交后不能被正常调度以及在使用 mpijob 命令配合 Torque 运行 MPI 并行程序时遇到了程序不能正确被执行的错误（指的是没有得到系统的正确执行，而非程序本身的错误）等异常情况，请及时来信或来电反馈相关信息，谢谢大家配合。

该文档如有遗漏或不当之处，请随时批评指出，谢谢。