University of Toronto Cluster in Computational Technology
High Performance Aerospace Computing Facility (HPACF)

HPACF User's Guide

University of Toronto Institute for Aerospace Studies
4925 Dufferin Street, Toronto, Ontario, Canada M3H 5T6

November 5, 2007

Contents

Chapter 1

Introduction to the HPACF

Synopsis: An overview of the HPACF is provided and guidelines for the usage of this parallel computing facility are given.

1.1 HP Beowulf Cluster

The Hewlett Packard (HP) computational cluster (known as HPACF for High Performance Advanced Computing Facility) is a Beowulf-class cluster consisting of:

- 14 x 4-way ES40 SMP HP AlphaServers,
  - 4 x 667 MHz Alpha EV67 with 8 MBytes L2 Cache
  - 13 nodes 2 GBytes, 1 node 8 GBytes RAM
- 12 x 4-way ES45 SMP HP AlphaServers,
  - 4 x 1000 MHz Alpha EV68CB with 8 MBytes L2 Cache
  - 10 nodes 4 GBytes RAM, 1 node 16 GBytes, 1 node 32 GBytes RAM
- 35 x 4-way Integrity rx4640 SMP HP Itanium Servers,
  - 4 x 1500 MHz Itanium 2 with 6 MBytes L3 Cache
  - 33 nodes 8 GBytes RAM, 1 node 32 GBytes, 1 node 64 GBytes RAM

for a total of 61 nodes, 244 processors, and 482 GBytes of distributed RAM. The total disk space for the system is 4422 GBytes. A high-bandwidth low-latency Myrinet network is used to connect the servers in the cluster for parallel computation with MPI, and Gigabit Ethernet connections are used for file I/O. All of the Alpha servers are running Redhat Linux 7.2 with a customized 2.4.19 kernel, and all of the Integrity servers are running RedHat AS 5.0 with a stock 2.6.18 kernel.

You may access the HPACF cluster directly from the UTIAS CFD network by logging on to **hpcfd.cfd** or you can also access the parallel cluster from an outside network through its "firewall" by logging into:

hpacf.utias.utoronto.ca.

The firewall, **hpacf.utias.utoronto.ca**, is an AMD Linux box, that provides a secure connection to the cluster. Once you have logged on to **hpacf**, you can then login either to one of the two Alpha nodes, **fe01** or **fe02**, or one of the two Itanium nodes, **fe03** or **fe04**, which act as front-ends to the

parallel cluster. **hpcfd.cfd** is in fact **fe03** and logging into **hpcfd** from the UTIAS CFD network is equivalent to logging into **fe03**. Please do not use **hpacf.utias.utoronto.ca** for compiling and running jobs, as it is only a firewall to the computational cluster.

The four front-end nodes (8 Alpha and 8 Itanium CPUs or processors) are reserved for interactive use, code development and debugging, running interactive jobs and test cases, and for submitting jobs to the batch queues. Please refrain form starting serial or parallel jobs on the front-ends which will run longer than one hour (wall clock time). In addition, do not set up scripts to keep running jobs in your absence. Also, run at low priority when running on the front-ends (use `nice +19` under `tcsh` or `csh` and `/usr/bin/nice -n +19` under other shells). If jobs running longer than the limit or jobs which are run by a script are spotted on the front-ends, they may be terminated without warning. The remaining 24 Alpha and 33 Itanium nodes in the cluster are compute nodes and are reserved for the batch queues.

1.2 Cluster Layout: Front-End and Compute Nodes

The layout of the HPACF cluster can be summurized as follows:

- firewall: `hpacf.utias.utoronto.ca` or `hpacf`
- front-ends: `hpcfd.cfd` or `hpcfd` or `fe01`, `fe02` (Alpha), `fe03`, and `fe04` (Itanium)
- serial nodes: `sn01 - sn03` (Alpha), `sn05 - sn08` and `pn22 - pn50` (Itanium)
- parallel nodes: `sn04`, `pn01 - pn12` and `pn14 - pn21` (Alpha), and `sn07`, `sn08`, `pn22 - pn50` (Itanium)

You may use both `fe01` and `fe02` as an interactive Alpha compute server and both `fe03` and `fe04` as an interactive Itanium compute server. There is some advantage to using `fe03` as the preferred Itanium front-end if you are in the UTIAS CFD group. This node provides nfs mounts of all user file systems in the UTIAS CFD network. The other frontends `fe02`, `fe03`, and `fe04` do not.

1.2.1 Operating System File Space

Each of the Alpha nodes (`fe, sn, pn`) use either a 9 GByte or 18 GByte SCSI disk for the Operating System (OS) with 2 GBytes used for memory swap space. The Itanium nodes have 73 GByte SCSI disks for the OS with 1024 MBytes used for memory swap space.

1.2.2 User File Space

The front-end `fe01 has two LVM volume groups, one composed of four SCSI 67 GByte disks (268 GB in total) and the other composed of four SCSI 18 GByte disks (72 GB in total), both of which are used for scratch space.`

`On the other hand, fe02 has two SCSI 18 GByte disks, two SCSI 36 GByte disks, and an LVM volume group composed of four SCSI 67 GByte disks for user space. For scratch, fe02 has one LVM volume group made up of four SCSI 67 GByte discs.`

`The front-end fe03 has two SCSI 135 GB disks collected in an LVM volume group (270 GB in total), and a separate four SCSI 67 GByte disks collected in an LVM volume group (268 GB in total).`

`Finally, fe04 has one 67 GByte disk for scratch space as well.`

On fe03, the 268 GB and 270 GB LVMs are reserved for the UTIAS CFD Group (Profs. Zingg and Groth). On fe02, one 18 GB drive is reserved for the UTIAS Spacecraft Dynamics and Microsatellite Technology Group (Prof. Damaren), and the other 18 GB drive is reserved for UTIAS Flight Dynamics and Control Group (Prof. Liu). In addition, one 36 GB disk is reserved for users in various UTIAS CFD Groups, other than the ones working with Profs. Zingg and Groth, and the other is reserved for the UTIAS Combustion and Propulsion Group. Finally, one 268 GB LVM on fe02 is reserved for the University of Toronto Computatio Technology Cluster group and Biomechanics group.

The firewall, **hpacf**, has some space set aside on its hard disk, which can be used by the UTIAS CFD, Spacecraft Dynamics and Microsatellite Technology, and Flight Dynamics and Control Groups to archive computational results. Other users are can arrange to have their own SATA drives installed on **hpacf** for archiving results.

A summary of the available disk space, and the various mount points is given below.

- **fe01**
  - **/d2**: 67 GBytes, Scratch space for users' jobs
  - **/d3**: 268 GBytes, Scratch space for users' jobs
- **fe02**
  - **/d1**: 18 GBytes, UTIAS Spacecraft Dynamics and Microsatellite Technology Group (CJD)
  - **/d2**: 36 GBytes, Other UTIAS CFD Groups, Structural Mechanics and Advanced Composite Materials (JSH)
  - **/d3**: 18 GBytes, UTIAS Flight Dynamics and Aircraft Design Group (HTL)
  - **/d4**: 268 GBytes, University of Toronto Computational Technology Cluster Group
  - **/d5**: 36 GBytes, UTIAS Combustion and Propulsion Group (JPS)
  - **/d6**: 268 GBytes, Scratch space for users' jobs
- **fe03**
  - **/d1**: 270 GBytes, UTIAS CFD Group (DWZ & CPTG)
  - **/d2**: 268 GBytes, UTIAS CFD Group (DWZ & CPTG)
  - **/d1**: 135 GBytes, Scratch space for users' jobs
- **fe04**
  - **/d1**: 67 GBytes, Scratch space for users' jobs
- **hpacf**
  - **/d1,/d2,/d3**: 215 GBytes, UTIAS CFD, Spacecraft Dynamics and Microsatellite Technology, and Flight Dynamics and Aircraft Design Groups

If users from other groups require a significant amount of space for their calculations then they will have to purchase an additional hard drive for installation on the cluster and pay a portion of the cost of the mounting hardware.

1.3 Cluster Usage Guidelines

Currently, the controls on cluster usage are very lax and use of the system is based on the honor system. Some general guidelines for the use of the system are as follows:

- **PLEASE READ THESE GUIDELINES!**
- Do not alter in anyway the default mechanisms for system access. For example the use of a .rhosts file is forbidden and has been disabled.
- Do not change the way system resources are allocated. For example using your own version of the mpirun script.
- Do not stack up too many jobs without monitoring their progress. For example creating a script that automatically resubmits jobs when one is finished should be avoided as you may have submitted a run that crashes or is problematic (i.e., dumps a large core file) and will continue to resubmit itself.
- Please exercise care in how you use the computational resources of this facility in terms of memory and disk space. Users must be careful to avoid filling disk drives or having their jobs exceed the memory capacity of the individual nodes (currently 2 GBytes or 4 GBytes per node on the Alpha machines, and 8GBytes on the Itanium machines).
- Please refrain form starting serial or parallel jobs on the front-ends which will run longer than one hour (wall clock time). In addition, do not set up scripts to keep running jobs in your absence. Also, run at low priority when running on the front-ends (use nice +19 under tcsh or csh and /usr/bin/nice -n +19 under other shells). If jobs running longer than the limit or jobs which are run by a script are spotted on the front-ends, they may be terminated without warning.

1.4 SGI Altix 3000

The SGI Altix computer is a powerful parallel computer with 32 Itanium 2 Intel CPUs. It has 32GB of RAM, there are 237GB of user space available, and it is connected to the local network using Gigabit LAN.

This computer can be accessed through hpacf.utias.utoronto.ca. The host name of the machine is basalt . In order to acquire an account on basalt, you need to contact Prof. Clinton Groth. You should send your correspondence to **groth@utias.utoronto.ca**. Please indicate whether you already have an account on hpacf, because if you do not, one has to be created for you. On the other hand, if you do have an account on hpacf, we have to make sure that you share your user space with the account on hpacf.

1.5 Basalt Usage Guidelines

Currently, there is no scheduler that ensures proper distribution of resources on basalt. Users are encouraged to respect others' work, and to ensure that the needed resources for their jobs are available by using the top and ps commands. This information is subject to change in the future.

Chapter 2

Batch Queuing System on HPACF

Synopsis: The batch queuing system on HPACF is summarized and procedures for submitting jobs to the compute nodes are given.

2.1 TORQUE (PBS)

You cannot login directly to any of the compute nodes. In order to make use of and interact with the compute nodes you must use TORQUE (based on PBS, the Portable Batch System) and the batch queues that have been created for your use. TORQUE/PBS is software for multi-node and multi-processor clusters that can be used to submit jobs to predefined batch queues. Currently, TORQUE version 2.1 is installed on the system and used to control job submission to the computed nodes.

2.2 Serial Queues

Two batch queues have been created to manage job submission to the Alpha serial nodes, and another two batch queues have been set up to manage job submission to the Itanium serial nodes. There are three Alpha compute nodes (12 cpus) reserved for serial (sequential jobs, nodes **sn01 - sn03**. On the other hand, almost all of the Itanium compute nodes can be either serial or parallel nodes, except for **sn05 - sn06** which are reserved for serial jobs. The three exclusively serial Alpha nodes and the two exclusively serial Itanium nodes have slightly different hardware configurations from the rest of the compute nodes, and the serial queues have been created to maximize user access to these serial nodes.

The two serial queues for the Alpha nodes and their respective properties are:

- Queue name: serial (for node sn03, EV67 667MHz, 8 GByte RAM)
  - resources_max.walltime = 24:00:00 (maximum of 24 hrs of wall clock time per job)
  - resources_max.nodect = 1 (maximum of one node per job)
  - resources_default.walltime = 01:00:00 (default wall clock time requested)
  - max_group_run = 4 (maximum of number of jobs allowed to run at one time per group)

- Queue name: serial_bigmem (nodes sn01 and sn02, EV68 1000MHz, 32 and 16 GByte RAM respectively; reserved access)
  - resources_max.walltime = 48:00:00 (maximum of 48 hrs of wall clock time per job)
  - resources_max.nodect = 1 (maximum of one node per job)
  - resources_default.walltime = 01:00:00 (default wall clock time requested)
  - max_group_run = 4 (maximum of number of jobs allowed to run at one time per group)

The two serial queues for the Itanium nodes and their respective properties are:

- Queue name: iserial (for nodes sn07, sn08, and pn22 - pn50, Itanium 2, 1.5 GHz, 8 GByte RAM)
  - resources_max.walltime = 48:00:00 (maximum of 24 hrs of wall clock time per job)
  - resources_max.nodect = 1 (maximum of one node per job)
  - resources_default.walltime = 01:00:00 (default wall clock time requested)
  - max_group_run = 6 (maximum of number of jobs allowed to run at one time per group)
- Queue name: iserial_bigmem (nodes sn05 and sn06, Itanium 2, 1.5 GHz, 64 & 32 GByte RAM respectively; reserved access)
  - resources_max.walltime = 48:00:00 (maximum of 48 hrs of wall clock time per job)
  - resources_max.nodect = 1 (maximum of one node per job)
  - resources_default.walltime = 01:00:00 (default wall clock time requested)
  - max_group_run = 4 (maximum of number of jobs allowed to run at one time per group)

For most users, using a serial value for the property directive in the -l flag (see Section 2.6 entitled Job Submission) and specifying the queue to be serial will be fine. Your job will then be submitted to **sn03**. Please, keep in mind that **sn03** has 2 GBytes of memory per CPU, and please ensure that your memory usage does not conflict with other jobs running on any of the serial nodes. Users with acess to the other Alpha serial queue serial_bigmem, are required to specify either serial-32G and serial-16G with the -l flag in place of the usual serial. This will provide the job with access to **sn01** and **sn02**, for serial jobs with memory needs up to 4 GBytes and 8 GBytes per job, respectively. Analogously, in order to run on the Itanium nodes serially, you would have to use an itanium-8G value for the property directive in the -l flag (see examples below) and to specify the queue to be iserial. The limit on RAM on the serial Itanium machines is 2 GBytes of RAM per job (the iserial queue), and if you require more, you would need access to the iserial_bigmem queue. Submitting to that queue with either itanium-64G and itanium-32G as the -l directive would execute your job on **sn05** or **sn06** with memory needs up to 16 GBytes and 8 GBytes of main memory per job respectively.

2.3 Parallel Queues

There are 21 Alpha compute nodes (80 CPUs) reserved for parallel jobs. They are **sn04, pn01 - pn12** (52 CPUs) and **pn14 - pn21** (32 CPUs). The Itanium compute nodes which are available for parallel execution are **sn07**, **sn08**, and **pn22 - pn50** (124 CPUs).

The queues intended for execution of parallel programs are split into day queues and night queues. The day is considered to run from 08:00 until 17:00, and the night is the rest of the 24 hour period. Weekends (17:00 on Friday until 8:00 am Monday) are considered nights. In order to save time for people who are making short runs during the day, and in order to increase cluster availability, only relatively short jobs are allowed to run during the day. In addition, the more processors a job requires, the less time it is allowed to run. Users may only submit to a queue named qmaster for the Alphas and to a queue called iqmaster for the Itaniums, which route their job accordingly. The day and night queues and their respective properties are outlined below. Although there are separate parallel queues for the Alphas and Itaniums, they are analogous and have the same properties. Of course, since the queues are separate, it means that the max_group_run property is counted separately for each architecture. For example, each group can run two 8 processor jobs on the Alphas and another two 8 processor jobs on the Itaniums simultaneously.

Please note that at this moment, the Alpha day queues are disabled and the night queues run 24 hours per day. This is due to a decrease in code development observed on the Alpha machines, and is intended to provide users with longer running times for their production runs.

- Day and night queues for jobs which request 8 processors.
  - resources_max.walltime = 2:00:00 (day) 8:00:00 (night) (maximum of 2 and 8 hrs of wall clock time per job during the day and night respectively)
  - resources_max.nodect = 2 (maximum of two nodes per job)
  - resources_default.walltime = 00:20:00 (day) 01:00:00 (night) (default wall clock time requested)
  - max_group_run = 2 (maximum of number of jobs allowed to run at one time per group)
- Day and night queues for jobs which request 16 processors.
  - resources_max.walltime = 1:00:00 (day) 6:00:00 (night)
  - resources_max.nodect = 4 (maximum of four nodes per job)
  - resources_min.nodect = 2 (minimum of two nodes per job)
  - resources_default.walltime = 00:20:00 (day) 01:00:00 (night)
  - max_group_run = 1
- Day and night queues for jobs which request 48 processors.
  - resources_max.walltime = 00:30:00 (day) 4:00:00 (night)
  - resources_max.nodect = 12
  - resources_min.nodect = 4

- resources_default.walltime = 00:10:00 (day) 01:00:00 (night)
- max_group_run = 1

- Day and night queues for jobs which request 80 processors.
  - resources_max.walltime = 00:20:00 (day) 2:00:00 (night)
  - resources_max.nodect = 20
  - resources_min.nodect = 12
  - resources_default.walltime = 0:05:00 (day) 01:00:00 (night)
  - max_group_run = 1

Users have to specify whether they want their job to run during the day or during the night. This is done by specifying a flag called **other**. The flag can take on the values of **day** for jobs which should run during the day, and **night** for jobs which should run during the night. The **other** flag is specified either as a flag to qsub, or in the submission script, as detailed in **Job Submission** below. In addition, users are allowed to submit only to the qmaster or iqmaster queues. That is why actual day/night queue names are irrelevant. How to specify that a job should be submitted to the qmaster or iqmaster queues is also discussed in Section 2.6 below.

Please note that due to low usage of the day queues on the Alpha nodes, they have been turned off. When submitting to qmaster (the Alpha nodes), please always specify **other** to be **night**.

Since the Alpha parallel nodes have two different clock speeds, the four Alpha parallel queues can be used in three different ways. If node property parallel-2G is specified by the user, the job will be run only on nodes with 667 MHz CPUs and 2 GBytes of RAM. If however the user specifies the node property parallel-4G, the job will be run only on nodes with 1000 MHz CPUs and 4 GBytes of RAM. Finally, if the property parallel is specified by the user, a mixture of the nodes can be used to execute the parallel task. The parallel property should only be used when requesting resources for programs that have been developed to take into account variable processor clock speeds when load balancing the task, otherwise the execution of the tasks on the faster processors will be limited by the execution speed of the slower processors, wasting the system resources. Examples in the following section will provide further information on how and when to use each of these node property flags. The Itanium CPUs are all at the same clock speed of 1.5 GHz. Therefore, only the node property itanium-8G is available.

Note that the organization of the batch queues will evolve over time. The queuing system needs to change in order to maximize the computational resources of the HPACF and we would appreciate any feedback on changes to the existing queues that you would like to see in order for you to maximize your throughput.

2.4 Summary of HPACF computational facilities

2.5 Maui Scheduler

The Maui cluster scheduler (www.supercluster.org) is used to schedule jobs for execution on the HPACF cluster. It is an extremely advanced and customizable scheduler, and

Queue	Architecture	CPU Frequency	Memory, GB	Node names
N/A	AMD Athlon	2,000	1	hpacf
N/A	Alpha EV68	1,000	4	fe01
N/A	Alpha EV68	1,000	4	fe02
N/A	Itanium 2	1,500	8	fe03
N/A	Itanium 2	1,500	8	fe04
serial_bigmem	Alpha EV68	1,000	32	sn01
serial_bigmem	Alpha EV68	1,000	16	sn02
serial	Alpha EV67	667	8	sn03
qmaster	Alpha EV68	1,000	4	pn14-pn21
qmaster	Alpha EV67	667	2	pn01-pn12, sn04
iserial_bigmem	Itanium 2	1,500	64	sn05
iserial_bigmem	Itanium 2	1,500	32	sn06
iserial	Itanium 2	1,500	8	sn07, sn08, pn22-pn50
iqmaster	Itanium 2	1,500	8	sn07, sn08, pn22-pn50

Table 2.1: HPACF facilities, as of January 2007. Key: fe="front-end", sn,pn="computational nodes".

its configuration will evolve over time to best satisfy the requirements on the system and to maximize its usage.

The Maui scheduler uses a mixture of credentials and fair share in order to prioritize jobs scheduled to run. In other words, how jobs submitted by a certain group are prioritize depends on a group's past and present usage of the system (the time frame for past usage is in the order of a couple of days). In addition, group credentials (shares) are used to give the group a certain ``base priority''.

Per user credentials may be set up at any time in order to further fine tune the system's performance, and if divisions among individuals within the same group become a necessity.

Another advanced feature the Maui scheduler, which is put to use at HPACF is backfill. Backfill allows Maui to schedule the next job (according to priority) in the future, by reserving the resources that job has requested when they become free. However, if gaps are formed by the reservation, during which resources are not utilized, Maui will schedule short jobs which fit within those gaps in order to utilize system time more efficiently. This is proven to reduce waiting time for short jobs even further.

IMPORTANT: In order to best benefit from the backfill feature, users are encouraged to specify relatively accurate wall-clock time requirements. In other words, users should attempt to estimate accurately how much execution time their job requires. Over-esti your time requirements is better than under-estimating them, however, so please err on the safe side. This is especially important during peak usage of the system, as it will guarantee better usage of the system resources, and shorter waiting times for users.

2.6 Job Submission

To submit a job to any of the queues you should make use of the PBS command qsub:

```
qsub [-q QUEUE] [-l RESOURCE_LIST] SCRIPT
```

where SCRIPT is a qsub script and QUEUE is the name of the queue to which you would like to submit the batch job. Examples of qsub scripts are given below in Sections 2.7 and 2.8. The qsub command accepts a number of flags aside from the ‘‘-q’’ flag (type man qsub for information on qsub flags). For serial jobs the ‘‘-q serial’’ or ‘‘-q iserial’’ flag must be specified and for parallel jobs, the ‘‘-q qmaster’’ or ‘‘-q iqmaster’’ flag must be used to access the Alphas or Itaniums respectively.

All flags to qsub may also be set in the qsub script file using PBS directives as illustrated in the script files given in Sections 2.7 and 2.8 below. Note that the default queue is serial and the default RESOURCE_LIST is nodes=1:ppn=1:serial, so for submission to the serial queue, the ‘‘-q’’ and ‘‘-l’’ flags are not required (only the ‘‘-l walltime=XX:XX:XX’’ flag is needed if the wall clock time should be changed from the default value of 1:00:00 (one hour). Specifying the ‘‘-l walltime=12:00:00’’ flag would overide the default wall clock time requested and request that the batch job be permitted to run for 12 hours. See the man pages for qsub, pbs, and pbs_resources_li for more information on submitting a job to the batch queues.

2.7 Serial Queue qsub Scripts

Examples of qsub script files for serial queue submission is given below for both the Alpha and Itanium architectures. The script uses several PBS directives to set qsub flags then changes directory to the directory for running the job and runs the executable pdes++, piping the standard output to the log file pdes++.log. The ‘‘-N groth_1’’ flag (set by the directive) assigns a name to the job. By default, the job is assigned the name of the qsub script file. The first example concerns the Alpha machines.

```
#!/bin/tcsh
#
#  Example of a PBS Job Submission Script for the Alpha Serial Queue
#  ------------------------------------------------------------------
#
# A job script may consist of PBS directives,
# comments and executable statements. A PBS
# directive (examples shown below) provides a
# way of specifying job attributes in addition
# to the command line options.
#
#PBS -N groth_1
#PBS -q serial
#PBS -l walltime=24:00:00,nodes=1:ppn=1:serial
#

cd $PBS_O_WORKDIR  # This command changes directory
                   # to where the qsub command was issued

/nfs/fe01/d1/cfd/groth/CFDkit+caboodle/src/pdes++ >& pdes++.log
```

The second example pertains to the Itanium machines.

```
#!/bin/tcsh
```

```
#
#  Example of a PBS Job Submission Script for the Itanium Serial Queue
#  ------------------------------------------------------------------
#
# A job script may consist of PBS directives,
# comments and executable statements. A PBS
# directive (examples shown below) provides a
# way of specifying job attributes in addition
# to the command line options.
#
#PBS -N groth_1
#PBS -q iserial
#PBS -l walltime=24:00:00,nodes=1:ppn=1:itanium-8G
#

cd $PBS_O_WORKDIR  # This command changes directory
                   # to where the qsub command was issued

/nfs/fe01/d1/cfd/groth/CFDkit+caboodle/src/pdes++ >& pdes++.log
```

2.8 Parallel Queue qsub Scripts

Example of qsub script files for parallel queue submission is given below for both the Alpha and Itanium architectures. The script changes directory to the directory for running the job and then runs the parallel executable BATSRUS.exe on 48 processors (12 nodes) using the mpirun command, piping the standard output to the log file BATSRUS.log Please note that the ‘‘other=day’’ or ‘‘other=night’’ attribute must be specified for parallel jobs. The first example is for the Alpha parallel machines.

```
#!/bin/tcsh
#
#   Example of a PBS Job Submission Script for a Parallel Queue
#   -----------------------------------------------------------
#
# A job script may consist of PBS directives,
# comments and executable statements. A PBS
# directive (examples shown below) provides a
# way of specifying job attributes in addition
# to the command line options.
#
#PBS -N groth_1
#PBS -q qmaster
#PBS -l walltime=4:00:00,nodes=12:ppn=4:parallel-2G,other=night

cd $PBS_O_WORKDIR  # This command changes directory
                   # to where the qsub command was issued

mpirun -np 48 BATSRUS.exe >& BATSRUS.log
```

The second example pertains to the Itanium parallel machines.

```
#!/bin/tcsh
#
#   Example of a PBS Job Submission Script for a Parallel Queue
#   -----------------------------------------------------------
#
# A job script may consist of PBS directives,
# comments and executable statements. A PBS
# directive (examples shown below) provides a
# way of specifying job attributes in addition
# to the command line options.
#
#PBS -N groth_1
#PBS -q iqmaster
#PBS -l walltime=2:00:00,nodes=16:ppn=4:itanium-8G,other=night

cd $PBS_O_WORKDIR  # This command changes directory
                   # to where the qsub command was issued

mpirun -np 64 BATSRUS.exe >& BATSRUS.log
```

SOME VERY, VERY IMPORTANT NOTES

It is important to note that the flag ‘‘-l nodes=XX:ppn=4:parallel_property’’, where XX is the number of requested nodes, must be set for every job submitted to any of the parallel queues. Failure to do so will result in incorrect parallel job management by the PBS queuing software and users will not be provided with proper access to the compute nodes. Also note that the ‘‘parallel_property’’ flag may take on values of parallel-2G, parallel-4G, or parallel for the Alphas as described above and permits control over the processor speed of the nodes to which the job is submitted. For the best usage of system resources, use values of either parallel-2G or parallel-4G when running on the Alphas, unless your parallel program can be efficiently load balanced across processors with varying clock speeds.

Please use the preceding example as a template to ensure the flags are correctly set when creating your own parallel queue submission scripts. The usefulness of the PBS software depends on the correct and fair use of the queuing software by the users.

It should be pointed out that the use of the preceding flag in the submission of parallel jobs means that, regardless of the number of processors actually needed by the user to run the job, the queuing software will allocate to the user the processors of the compute node in multiples of 4. Consequently, all of the processors on the compute nodes associated with the job request are allocated for use (whether they are actually used or not) and they are not available for use by other users. This method of allocating the nodes is required to ensure the proper functioning of the high-speed Myrinet network.

The second very important note is that it is extremely important to set the flag ‘‘other={night,day}’’ because otherwise your job may be rejected by PBS due to a resource requirement mismatch.

The third note is that users who edit their qsub scripts in Windows should be careful about using them to submit their programs. In Windows, the editors do not add a newline,

which is required by UNIX programs, at the end of the file. Please edit your scripts on HPACF or the frontends using one of the editors there. Examples of such editors are pico, emacs, vi. You can read their manpages for help on how to use them.

2.9 Deleting a Job Using qdel

You can delete your job from the batch queue using the qdel command. Type:

qdel XX.hpacf.utias.utoronto.ca

where XX is your job number (XX.hpacf.utias.utoronto.ca is the job id). Refer to the qdel man pages for further details.

2.10 Monitoring a Job Using qstat

You can monitor the status of your batch job using the qstat command. Type:

qstat -a

or

qstat -f

or

qstat -n

to obtain information about your jobs. Refer to the qstat man pages for further details.

2.11 Reserving an Entire Node For a Serial Job

If you would like to reserve an entire node (i.e., all four processors and all of the memory) for a serial computation to perform single CPU timings and/or to use all of the available 2/4/8 GBytes of memory on a node, you can do this using the parallel queues (qmaster or iqmaster). Use the *parallel* submission script given above with ‘‘-l walltime=8:00:00,nodes=1:ppn=4:parallel_property,other=[day|night]’’ in order to submit your serial job. Please note that this gives you a maximum of 8 hours of runtime with ‘‘other=night’’ and 2 hours with ‘‘other=day’’ however. PBS will reserve an entire node for your serial calculation. The ‘‘parallel_property’’ flag can be used to specify the clock speed of the processors and memory size of the machine to be used in carrying out your job and may have values of parallel-2G, parallel-4G, parallel, or itanium-8G as described above. *Remember to request 1 node and 4 processors per node and set the parallel node property in your job submission script!*

2.12 Submitting Interactive Jobs to TORQUE/PBS

It is possible to run interactive jobs through the batch scheduler. One reason one may want to do this is to run an interactive job for more than one hour (see Section 1.3 for limit on jobs run interactively on the front-ends). Alternatively one may want to run an interactive job on more than 8 CPUs (remember that each architecture

has two front-end machines, allowing one to use a maximum of 8 CPUs from a particular architecture for interactive running).

When you run an interactive job, a shell is opened on the first node which TORQUE has reserved for you. Once you have the shell opened, you can launch your job and all I/O from your job will be directed to this terminal. Let's go through an example based on the second script (for the Itanium architecture). The directives in the script were:

```
#PBS -N groth_1
#PBS -q iqmaster
#PBS -l walltime=2:00:00,nodes=16:ppn=4:itanium-8G,other=night
```

The command in the script was:

```
cd /nfs/fe01/d1/cfd/groth/Test
mpirun -np 64 BATSRUS.exe >& BATSRUS.log
```

In order to launch the above job interactively, you would have to run the commands:

```
qsub -I -N groth_1 -q iqmaster \
     -l walltime=2:00:00,nodes=16:ppn=4:itanium-8G,other=night
```

Your job will then be queued, and when the nodes for its execution become available, the terminal in which the qsub command was invoked will execute the shell where you can run your interactive process. Once this is done, you can simply type into the shell the sequence of commands from the script (with ``>& BATSRUS.log'' removed because we would like to see the output interactively):

```
cd /nfs/fe01/d1/cfd/groth/Test
mpirun -np 64 BATSRUS.exe
```

and your process will proceed to execute on 16 Itanium machines and its standard I/O will be directed to the terminal where you executed the qsub command (and consequently where you typed the commands above).

Remember that any #PBS directive can be given to qsub as a command-line option (see man qsub for more information). Also, please make sure that you're asking for as many CPUs as you plan to run on! I.e. the ``nodes=16:ppn=4'' flag above means that you're asking TORQUE to reserve 64 CPUs for your interactive run. Therefore, following this request, your mpirun command should have the ``-np 64'' option.

Chapter 3

Program Development Tools on HPACF

Synopsis: Tools made available on the HPACF for the development of serial (sequential) and parallel programs and code are summarized.

3.1 Compilers

The gnu C, C++, and FORTRAN 77 compilers (gcc, g++, and g77) are available on the HP Alpha and Itanium machines for program and code development.

3.1.1 Alpha compilers and debugger

The HP Alpha compilers C, C++, and FORTRAN 90 compilers (ccc, cxx, and fort) which were developed by HP specifically for the Alpha cpu have also been installed. You are encouraged to use the HP Alpha compilers for all of your program development in order to achieve maximum performance from your code on the Alpha machines. HP also provides a debugger called ladebug. Refer to the man pages for the ccc, cxx, fort, and ladebug commands for further information concerning the HP Alpha compilers and debugger.

Note that on the Alphas, the gcc compiler version 2.96 and the gcc3 version 3.0.4 compiler are installed in /usr/bin. The HP Alpha compilers (ccc, cxx, and fort) use gcc version 2.96 as the back end. Please ensure that your path is set correctly to use the desired version of the gcc compiler.

/usr/bin/gcc -> gcc version 2.96 /usr/bin/gcc3 -> gcc version 3.0.4
/usr/bin/ld -> binutils version 2.11.91

3.1.2 Itanium compilers and debugger

The Intel C, C++, and FORTRAN 90 compilers (icc, icpc, and ifort) are also available on the Itanium machines. You are encouraged to use the Intel compilers for all of your program development in order to maximize the performance of your applications on the Itanium machines. Intel also provides a debugger called idb on the Itanium architecture. Refer to the man pages for the icc, icpc, ifort, and idb commands for further information.

Please note that the gcc version on the Itanium machines is 3.2.3. Also, note that the default version of the Intel compilers is 8.1. If you would like to utilize the version 9.0 compilers which are also installed, you would have to modify your startup scripts. Add the following lines to your .bashrc or .cshrc files (for bash and tcsh users respectively):

To .bashrc add:

```
if [ `expr $MACHTYPE == ia64-redhat-linux-gnu` -eq 1 ] ; then
  source /opt/intel/cc_90/bin/iccvars.sh
  source /opt/intel/fc_90/bin/ifortvars.sh
  source /opt/intel/idb_90/bin/idbvars.sh
fi
```

To .cshrc add:

```
if ( `expr $MACHTYPE == ia64` == 1 ) then
  source /opt/intel/cc_90/bin/iccvars.csh
  source /opt/intel/fc_90/bin/ifortvars.csh
  source /opt/intel/idb_90/bin/idbvars.csh
endif
```

You can check whether the compiler you are trying to use is the correct one with the "which" UNIX utility. For example, after adding the abovementioned lines, log out, log back in, and typing ``which icc'', you should see ``/opt/intel/cc_90/bin/icc as the output.

3.2 MPI

To create your own parallel executables for running on multiple processors, you can use MPI (the Message Passing Interface) on both the Alpha and Itanium architectures. This is a library of subroutines, available in C, C++, FORTRAN 77, and FORTRAN 90/95, which permit the program developer to control, coordinate, and send information between a group of processes that are participating in the computational solution of a problem. We have installed Myrinet's modified version of MPICH 1.2.6 (mpich-1.2.6..13b) and this works in conjunction with GM (GM 2.0.16, Myrinet's own message passing library) to provide communication over the high-speed Myrinet 2000 network switch. Two installations of MPICH-GM have been compiled and installed on the system. One of the installations of MPICH-GM was built with the HP Alpha compilers on the Alpha machines and with the Intel compilers on the Itanium machines. It is installed in /opt/mpich-1.2.6..13b and MPI documentation and manuals may be found in /opt/mpich-1.2.6..13b/doc. This is the default version of MPI on both architectures. Another version of MPICH-GM is also available on both architectures, which has been built with the GNU comilers (gcc) and installed in /opt/mpich-1.2.6..13b-gcc. Some users may wish to use this version of MPI, for example for debugging.

Please refer to this documentation to get started using MPI. The required MPI scripts are located in /opt/mpich-1.2.6..13b/bin or /opt/mpich-1.2.6..13b-gcc/bin. It is a good idea to ensure that this directory is included in your PATH environment variable. Here is a summary of some MPI commands:

MPICH-GM using HP Alpha/Itanium Compilers, Default

```
MPI installation directory = /opt/mpich-1.2.6..13b
MPI C Compiler             = /opt/mpich-1.2.6..13b/bin/mpicc
MPI C++ Compiler           = /opt/mpich-1.2.6..13b/bin/mpiCC
MPI FORTRAN 77 Compiler    = /opt/mpich-1.2.6..13b/bin/mpif77
MPI FORTRAN 90 Compiler    = /opt/mpich-1.2.6..13b/bin/mpif90
MPI run command            = /opt/mpich-1.2.6..13b/bin/mpirun
```

MPICH-GM using gnu Compilers (paths apply to both Alpha and Itanium architectures

```
MPI installation directory = /opt/mpich-1.2.6..13b-gcc
MPI C Compiler             = /opt/mpich-1.2.6..13b-gcc/bin/mpicc
MPI C++ Compiler           = /opt/mpich-1.2.6..13b-gcc/bin/mpiCC
MPI FORTRAN 77 Compiler    = /opt/mpich-1.2.6..13b-gcc/bin/mpif77
MPI run command            = /opt/mpich-1.2.6..13b-gcc/bin/mpirun
```

A VERY, VERY IMPORTANT NOTE

On the Alpha machines, the default MPICH-GM has been compiled using the HP Alpha compilers and the scripts mpicc, mpiCC, mpif77, and mpif90 call the ccc, cxx, and fort compilers in order to compile your parallel code. Compiler flags for the ccc, cxx, and fort compilers are passed through by the mpicc, mpiCC, mpif77, and mpif90 scripts.

Analogously, on the Itanium machines, the default MPICH-GM has been compiled using the Intel compilers, and the scripts mpicc, mpiCC, mpif77, and mpif90 call the icc, icpc, and ifort compilers. The flags to the compilers are similarly passed through by the mpicc, mpiCC, mpif77, and mpif90 scripts.

3.3 The mpirun Script

The MPICH-GM mpirun script is used to run a parallel job with the ‘‘-np XX’’ flag set, where XX is the number of requested processors or cpus. For example,

```
mpirun -np 48 BATSRUS.exe >& BATSRUS.log
```

Note you can run short parallel jobs (compiled for the Alpha architecture) interactively on nodes **fe01** and **fe02** using the mpirun command with the "-np" flag set to the desired number of processors (up to a maximum of 8 cpus can be used in this case). Alternatively, you can run short parallel jobs compiled for the Itanium architecture on nodes **fe03** and **fe04** using the same syntax and subject to the same limits. You do not need to use PBS and the qsub command to run interactively on the front-end nodes.

3.4 Flags for mpirun

The MPICH-GM mpirun script accepts a number of command line arguments or flags. Given below is a list of some of useful flags. The list is not inclusive. Type mpirun --gm-h to obtain help and a complete list of mpirun flags.

- --gm-h: generates help message
- -np XX or --gm-np XX: specifiy number of processors used to run job where XX is the number of processors

- --gm-node-pack: Passing this flag will force the job to be run on the specified compute nodes in a ‘‘packed’’ or ‘‘clustered’’ fashion. This is the default behaviour. Thus, for ‘‘-np 8’’ on nodes **pn01** and **pn02**, the processors will be assigned to the two nodes in the following order:

  pn01
  pn01
  pn01
  pn01
  pn02
  pn02
  pn02
  pn02

- --gm-node-unpack: Passing this flag will the job to be run on the specified compute nodes in a ‘‘unpacked’’ or ‘‘interleaved’’ fashion. Thus for ‘‘-np 8’’ on nodes **pn01** and **pn02**, the processors will be assigned to the two nodes in the following order:

  pn01
  pn02
  pn01
  pn02
  pn01
  pn02
  pn01
  pn02

- --gm-no-shmem: This flag disables shared memory support. Shared memory support is compiled and enabled by default in MPICH-GM. It may be disabled at run-time using this flag. By default the shared memory device will be used for message passing between processors on the same node, rather than making use of the Myrinet high-speed switch. This is generally more efficient and most MPI users should use the default and not use this flag.

- --gm-kill <N>: This flag will ensure that all MPI processes are killed N seconds after the first MPI process exits. The README file for MPICH-GM states that this implementation of MPI makes no attempt at terminating the other nodes when MPI_Abort is called on one node. Only the calling process is terminated, and the others do not notice (and so will generally hang at some point). Although this is a buggish shortcut, some other MPI implementations behaved similarly. It is not simple to implement an MPI_Abort with a correct semantic. In the meantime, the MPICH-GM solution is to introduce the "--gm-kill" flag at runtime, on the mpirun command line. This flag tells mpirun to kill all of the MPI processes N seconds after the first one exits or aborts. Thus the abort is propagated to the rest of the MPI job but outside of the MPI code. Most MPI users should use this flag.

3.5 Other Installed Packages

3.5.1 PETSc

Portable, Extensible Toolkit for Scientific Computation, or PETSc, is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modelled by partial differential equations. On the Alpha front-ends, PETSc 2.1.5 has been compiled using both gnu and HP Alpha compilers and the libraries are located in the following directories on **fe01** and **fe02**:

gnu Version: /usr/local/PETSc/petsc-2.1.5/lib/libO/linux_alpha/
HP Alpha Version: /usr/local/PETSc/petsc-2.1.5/lib/libO/linux_alpha_dec/

In addition, PETSc 2.3.1 has been compiled on both Alphas and Itaniums using only their native compilers (cxx and icc respectively), and is available in the following directories on **fe01-fe04**:

/usr/local/PETSc/petsc-2.3.1-p9

Chapter 4

Questions, Requests, and Feeback

Synopsis: Contact information is given for reaching administrators of the HPACF.

4.1 Contact Information

4.2 User Accounts

You may request user a account on HPACF by contacting:

Joseph Chen
High Performance Parallel Cluster Administrator

Phone: (416)667-7796
E-mail: chen@utias.utoronto.ca

Address: Institute for Aerospace Studies
University of Toronto
4925 Dufferin Street
Toronto, ON M3H 5T6

Prof. Clinton P. T. Groth

Phone: (416) 667-7715 Fax: (416) 667-7799
E-mail: groth@utias.utoronto.ca
Home page: http://www.utias.utoronto.ca/~groth

Address: Institute for Aerospace Studies
University of Toronto
4925 Dufferin Street
Toronto, Ontario, Canada M3H 5T6