
VERSATUS
HIGH-PERFORMANCE COMPUTING

Guia Rápido de Utilização

Cluster HPC “cajuina.ufpi.br”

Sobre este guia

Este guia tem o objetivo de apresentar aos usuários e administradores o recurso computacional “cajuina.ufpi.br” da UFPI descrevendo brevemente o cluster e o sistema instalado, além de explicar como utilizá-los.

Sumário

Descrição do Cluster

Hardware

O cluster **“cajuina.ufpi.br”** é composto de um head-node e quatro compute-nodes interligados por uma rede Gigabit Ethernet (1 Gbps).

O nó de gerenciamento (head-node) é um servidor SGI dual socket com 1 processador AMD Opteron(tm) Processor 6344, o processador possui 24 núcleos. É composto também por 8 pentes de 4 GB de memória RAM, 4 HDD's de 3 TB e 2 HDD's de 500 GB. No total estão disponíveis:

- 24 núcleos;
- 32 GB de memória RAM;
- 9 TB de espaço em disco para os diretórios de usuários (4 HDD's de 3 TB em RAID 5);
- 500 GB de espaço para o sistema (2 HDD's de 500 GB em RAID 1).

Os cinco nós computacionais (compute-nodes) são servidores SGI quad socket com 4 processadores AMD Opteron(tm) Processor 6376, cada processador possui 16 núcleos. E é composto também por 16 pentes de 8 GB de memória RAM e 1 HDD de 1 TB. No total estão disponíveis:

- 64 núcleos;
- 128 GB de memória RAM;
- e 1 TB de espaço em disco.

Ambiente de Trabalho

Arquivos e Armazenamento

O diretório padrão do usuário (HOME) é construído da seguinte forma:

/home/*usuário*

Ex: Para o usuário "usuário" temos o diretório "/home/usuário".

Esta área possui 9 **TB disponíveis** que são compartilhados entre todos os usuários do cluster. Há também uma outra área de trabalho destinada ao armazenamento temporário destes trabalhos que é chamado de scratch, o diretório está localizado em "/data2" (existente apenas nos nós computacionais). Este diretório tem a mesma organização encontrada em "/home".

Ex: Para o usuário "usuário" temos "/data2/usuário".

O scratch "/data2" é a área de trabalho reservada no disco de cada nó computacional, portanto **ele só é acessível no próprio nó e tem 813 GB de espaço disponível**.

Atenção: tenha em mente que o diretório de scratch é um **espaço temporário.** Ele é checado diariamente e arquivos com mais de 30 dias são apagados. Arquivos importantes devem ser movidos para o diretório "/home" que possui um nível maior de segurança dos dados.

Os Nós de Processamento

Os nós deste cluster estão nomeados no formato "n00X", onde "X" representa o número que identifica o nó.

Ex: n003 (nó 3).

Redes

Os nós estão conectados a 3 redes distintas que são relacionadas apenas para completude das informações:

1. Ethernet;
2. IPMI

Em geral, a rede IPMI é utilizada apenas procedimentos administrativos do cluster, enquanto que a rede Ethernet é utilizada para trocar arquivos entre os nós e também mensagens MPI. A seleção das redes é feita de forma automática, portanto os usuários não precisam se preocupar com essas distinções.

Forma de Acesso

O acesso é feito via SSH, utilizando o comando:

```
ssh -p 5523 usuário@200.137.162.107
```

Pode-se também acessar de dentro da rede interna da UFPI, utilizando o comando:

```
ssh -p 5523 usuário@10.40.2.1
```

Por fim, transferência de arquivos pode ser feito com o seguinte comando:

```
scp -P 5523 arquivo.zip usuário@200.137.162.107:~/
```

Ou então, de dentro da rede interna da UFPI com o seguinte comando:

```
scp -P 5523 arquivo.zip usuário@10.40.2.1:~/
```

Softwares Instalados

A maioria dos softwares pré-instalados no cluster estão disponíveis em “/opt/versatushpc”. Dentro deste diretório as aplicações estão divididas entre bibliotecas (“libraries”), compiladores (“compilers”), códigos-fonte (“sources”) e “softwares”.

Compiladores

A família de compiladores GCC e suas ferramentas estão integradas ao sistema e localizadas nos diretórios padrão: “/usr/bin”, “/usr/lib64” e “/usr/include”. Também está disponível o compilador do CUDA (“nvcc”) localizado no diretório “/opt/versatushpc/compilers/cuda-5.0”.

GCC (“gcc”, “g++”, “gfortran”)

Versão 4.4.7;
Executáveis: instalados no diretório “/usr/bin”;
Bibliotecas: instaladas nos diretórios “/usr/lib” e “/usr/lib64”;
Cabeçalhos: instalados no diretório “/usr/include”;

PGI (“pgcc”, “pgc++”, “pgfortran”)

Versão: 14.10
Executáveis: '/opt/versatushpc/compilers/pgi-14.10/linux86-64/14.10/bin''.
Bibliotecas: '/opt/versatushpc/compilers/pgi-14.10/linux86-64/14.10/libso'.
Cabeçalhos: '/opt/versatushpc/compilers/pgi-14.10/linux86-64/14.10/include'.
Arquivo de module : '/export/versatushpc/data/modulefiles/compilers/pgi-14.10'.

Bibliotecas

As bibliotecas estão localizadas em “/opt/versatushpc/libraries”:

Ex: “/opt/versatushpc/libraries/openmpi-1.5.4/gnu-4.4”

Estão pré-instalados no cluster as seguintes bibliotecas:

OpenMPI

Essa biblioteca é uma implementação de código aberto do padrão MPI. A versão instalada é a 1.5.4 (compilada com o GCC 4.4) localizada no diretório “/opt/versatushpc/libraries/openmpi-1.5.4/gnu-4.4”, tanto no head-node quanto nos compute-nodes.

Softwares Científicos, Visualizadores e Editores

Os softwares científicos estão instalados em “/opt/versatushpc/softwares”, seguindo o esquema mostrado abaixo:

Abinit 7.8

Todos os executáveis estão em: “/opt/versatushpc/softwares/abinit/7.8-gnu-4.4/bin”.

Quantum ESPRESSO 5.1

Todos os executáveis estão em: “/opt/versatushpc/softwares/espresso/5.1-gnu-4.4/bin”.

Octopus 4.1.2

Todos os executáveis estão em: “/opt/versatushpc/softwares/octopus-4.1.2/gnu-4.4/bin”.
Os arquivos auxiliares estão em: “/opt/versatushpc/softwares/octopus-4.1.2/gnu-4.4/share”.

SIESTA 3.2

Todos os executáveis estão em: “/opt/versatushpc/softwares/siesta/3.2-gnu-4.4/bin”.

SIESTA-ldau308

Todos os executáveis estão em: “/opt/versatushpc/softwares/siesta-ldau308/gnu-4.4/bin”.

SIESTA-trunk453

Todos os executáveis estão em: “/opt/versatushpc/softwares/siesta-trunk453/gnu-4.4/bin”.

Variáveis de Ambiente

Em ambientes de computação de alto desempenho é comum que haja diversos compiladores e bibliotecas diferentes para a mesma rotina. Por exemplo, para compilação de códigos em C, temos disponíveis o “gcc”. Além dessa variação de softwares, que desempenham a mesma função, também é comum ter diferentes versões de compiladores, bibliotecas e softwares. Com isso, surge a necessidade de selecionar quais arquivos queremos que o sistema operacional disponibilize para o uso, de acordo com a atividade que iremos desempenhar. No cluster “cluster-sgi” está disponível uma lista com todas as bibliotecas e compiladores presentes na máquina, basta o usuário selecionar quais ferramentas ele irá utilizar. Essa seleção é feita por sessão, ou seja, a cada job as ferramentas devem ser selecionadas e ao final do job elas voltam ao valor padrão de “nenhuma seleção”. O comando utilizado para este gerenciamento é o “module”, ele pode listar, carregar, descarregar ou substituir uma ferramenta dentre as disponíveis.

Os módulos de variáveis atualmente disponíveis são:

```
[usuário@cluster-sgi ~]$ module avail

--------------------- /opt/versatushpc/data/modulefiles --------------------

compilers/pgi-14.10                     softwares/espresso/5.1-gnu-4.4
softwares/abinit/7.8-gnu-4.4            softwares/siesta/3.2-gnu-4.4
softwares/octopus-4.1.2/gnu-4.4         softwares/siesta-trunk453/gnu-4.4
softwares/siesta-ldau308/gnu-4.4
libraries/openmpi-1.5.4/gnu-4.4
```

A sintaxe do “module” é:

- module **av**, **avail** - lista todos os módulos disponíveis;
- module **list** - lista todos os módulos atualmente em uso;
- module **add**, **load *caminho/do/módulo*** - carrega as configurações e prepara o ambiente de acordo com o módulo selecionado;
- module **rm**, **unload *caminho/do/módulo*** - remove as configurações específicas deste módulo;
- module **purge -** remove todos os módulos carregados ;
- module **help -** exibe todos os subcomandos do module;
- module show ***caminho/do/módulo*** - exibe o conteúdo do arquivo de módulo

Exemplos:

Cenário 1: Um programa para ser compilado com MPI necessita do “mpicc”, “mpif90”, etc, então o usuário deve carregar o módulo da biblioteca MPI, compilada com o respectivo compilador (o módulo do compilador será carregado automaticamente):

```
[usuário@cluster-sgi ~]$ module load libraries/openmpi-1.5.4/gnu-4.4
```

Utilizando o Cluster

Submetendo e Administrando Jobs

Para que os recursos computacionais sejam divididos de maneira razoável entre todos os usuários do cluster, é necessário uma forma de organizar e priorizar as requisições de uso, comumente chamadas de jobs. Para isso, o cluster conta com o sistema de gerenciamento de recursos Torque [http://www.adaptivecomputing.com/products/open-source/torque/] e o agendador de tarefas Maui [http://www.adaptivecomputing.com/products/open-source/maui/] que juntos gerenciam os jobs que serão executados, respeitando as políticas estabelecidas. O Torque trabalha com o conceito de filas, que são estruturas para classificar e agrupar os jobs sob critérios como: número de processadores requisitados, quantidade de nós, quantidade de memória RAM, tempo de processamento e assim por diante. Os limites são configurados de acordo com as políticas definidas pelos administradores do cluster.

As políticas atuais das filas estão definidas da seguinte forma:

1. Os jobs em que não tiverem o tempo de walltime especificado durante submissão serão automaticamente configurados para tempo infinito;
2. Os jobs que requisitem um maior número de núcleos e/ou um menor tempo terão maior prioridade na fila de espera;
3. Os jobs com menor prioridade podem ser executados antes desde que não atrasem o início dos jobs de maior prioridade, ou seja , o gerenciador desconsidera a prioridade do job se o mesmo não atrasar o tempo estimado de início de execução de outros jobs que tenham maior prioridade;
4. Não é necessário indicar a fila onde o job será submetido. O gerenciador de recursos faz isso, sem intervenção do usuário.

É possível exibir as filas do sistema e o seu status com o comando:

```
[usuário@cluster-sgi ~]$ qstat -q
```

Diretivas e Variáveis do Job

Dentro dos jobs existem algumas instruções que são dedicadas ao Torque, entre outras coisas para instruí-lo onde, como e por quanto tempo rodar. Essas instruções são chamadas diretivas, e estão nas linhas que se iniciam com: “#PBS”. O Torque irá considerar todas as diretivas até que encontre uma linha executável, depois deste ponto, será ignorada qualquer outra diretiva.

Diretivas comuns são:

- #PBS -N nome (Nome do job)
- #PBS -lnodes=4:ppn=8 (Número de nós/processadores)
- #PBS -lwalltime=24:00:00 (Tempo de execução hh:mm:ss)

Além disso, quando o job é submetido, o Torque coloca a disposição do job algumas variáveis de ambiente que tem o intuito de facilitar algumas operações. Algumas delas, por exemplo são:

- $PBS_NODEFILE → esta variável contém a lista de nós e CPU's que foram alocadas ao job, é útil, por exemplo para ser usada como parâmetro para o “mpirun”:

```
mpirun -machinefile $PBS_NODEFILE meuprograma
```

- $PBS_O_WORKDIR → esta variável aponta para o diretório de onde o script foi submetido.

- $PBS_JOBNAME → contém o nome especificado na diretiva “#PBS -N *nome*”.

A lista completa de diretivas e variáveis pode ser encontrada em (em inglês) [http://docs.adaptivecomputing.com/torque/Content/topics/2-jobs/exportedBatchEnvVar.htm]

Além das variáveis padrões do Torque está disponível uma variável $SCRATCH que aponta para o diretório “/scratch/local/*usuário*” nos nós. Você pode usá-la para acessar o seu diretório ou em seus scripts (Assim como pode usar a $HOME para o seu diretório “/home/*usuário*”). Ex:

```
[usuário@cluster-sgi ~]$ cp -ar $HOME/meuprimeirojob $SCRATCH
```

Considerando que o usuário a executar este comando seja o usuário “usuário”, o efeito final seria:

```
[usuário@cluster-sgi ~]$ cp -ar /home/usuário/meuprimeirojob
/scratch/local/usuário
[usuário@cluster-sgi ~]$ ls $SCRATCH
meuprimeirojob
```

Tipos de Jobs

Quanto aos jobs, existem dois tipos mais comuns:

Batch Job

Um arquivo de script que contém comandos para executar aplicações específicas, usado pelo programa “qsub” para informar ao torque as características do teu job. Exemplo de script:

```
#PBS -S /bin/bash
#PBS -l walltime=5:00:00
#PBS -l nodes=1:ppn=8
#PBS -N layers

module load softwares/siesta/3.2-gnu-4.4

cd $PBS_O_WORKDIR

mpirun -np $PBS_NP siesta < $PBS_JOBNAME > $PBS_JOBNAME-$PBS_JOBID”.out”
```

- A primeira linha informa ao Torque que o shell escolhido pelo usuário para rodar o job é o bash (“/bin/bash”);
- A segunda linha indica, em horas reais, qual a duração do job;
- A terceira linha indica, respectivamente, a quantidade de nós e o número de núcleos por nó para este job;
- A quarta linha indica o nome dado a este job;
- A sexta linha indica que o módulo do Siesta 3.2 deve ser carregado, configurando então as variáveis de ambiente deste software;
- A variável $PBS_O_WORKDIR tem por valor padrão o diretório onde o job foi submetido. Logo após seguem os comandos para a execução do job propriamente dito;
- Variáveis de ambiente necessárias à execução do job devem ser declaradas/definidas antes do job.

Exemplo da submissão de um batch job:

```
[usuário@cluster-sgi ~]$ cd teste
[usuário@cluster-sgi teste]$ ls teste
input.inp                 script.pbs
[usuário@cluster-sgi teste]$ qsub script.pbs
```

Dica: Sempre crie um diretório em seu diretório HOME e coloque todos os arquivos relativos ao job (input, script de submissão, etc) nele, entre neste diretório e submeta o script. Assim, garante que arquivos de outros jobs não se misturem.

Job Interativo

É executado como um “batch job”, porém o terminal do usuário é conectado ao host de execução, similar a uma sessão de login. A partir daí o usuário envia as opções do script de job como comandos individuais, o que facilita ao usuário depurar um script com problemas.

O parâmetro para submeter um job Interativo é “-I” (“i” maiúsculo), se o script for aceito, o Torque retornará um número identificador do job, mais conhecido como “id”.

Exemplo da submissão de um job interativo:

```
[usuário@cluster-sgi ~]$ qsub -I -l nodes=1 -l ncpus=8 -l walltime=5:00:00
```

Exemplos de Script de Submissão

Abaixo são apresentados scripts de submissão com ideias que podem ser mescladas para se construir um script ideal para cada tipo de trabalho, além dos já presentes no diretório "/opt/versatushpc/data/jobscripts".

Exemplo 1

Neste exemplo é utilizado um nome base para os dados, o arquivo de entrada recebe o sufixo “.inp” enquanto o arquivo de saída recebe o sufixo “.out”.

O arquivo de entrada é copiado para um diretório temporário ($SCRATCH) dentro do nó que irá executar o trabalho. O cálculo é executado no disco local do nó para minimizar perda de performance com transferência de dados pela rede, e após o seu termino os dados são retornados ao head-node. Os scripts com sufixo “.torque” (sem “-local”) são deste tipo abaixo:

```
#!/bin/bash
#PBS -S /bin/bash
#
## nodes = quantidade de nós requisitada.
## ppn = quantidade de núcleos por nó.
#PBS -l nodes=1:ppn=16
#
## walltime = quantidade de horas necessárias.
#PBS -l walltime=168:00:00
#
## Nome do job . Aparece na saída do comando 'qstat'.
## É recomendado, mas não necessário, que o nome do job
## seja igual ao nome do arquivo de input.
#PBS -N

## Variáveis padronizadas para todos os jobs
WRKDIR=$SCRATCH/$PBS_JOBID


# procura o nome o input baseado no nome do job (linha #PBS -N xxx acima).
INP=$PBS_JOBNAME".inp"


## O diretório que sera criado para rodar o job
## deve ser apagado quando o job terminar ?
## Por padrão será, ao menos que declare qualquer outro valor.
APAGA_SCRATCH=Y


## informações do job no arquivo de saída.
qstat -an -u $USER
cat $PBS_NODEFILE


########################################
#-------  Inicio do trabalho  ----- #
########################################

## Configura o nó de calculo.
module load softwares/siesta/3.2-gnu-4.4


## Cria o diretório em que o job rodará.
mkdir -p $WRKDIR
```

```
## Transfere os inputs e arquivos necessários
## para o diretório '/scratch'.
cd $PBS_O_WORKDIR

cp -r * $WRKDIR/

cd $WRKDIR

mpirun -np $PBS_NP siesta < $INP > $PBS_JOBNAME-$PBS_JOBID.out

cp -r $WRKDIR/ $PBS_O_WORKDIR/


## Apaga o diretório que o job rodou.
if [ x"$APAGA_SCRATCH" = x"Y" ]; then

      rm -rf $WRKDIR

else

      echo -e "\nO diretório \e[00;31m$WRKDIR\e[00m deve ser removido
manualmente
      para evitar problemas para outros jobs e/ou usuários. \n"

fi
```

Exemplo 2

O exemplo 2 é mais simples do que o exemplo 1 por não realizar a transferência dos arquivos para a área de “scratch” do nó, ou seja, é executado no próprio diretório que o job foi submetido. Os scripts com sufixo “-local.torque” são deste tipo abaixo:

```
#!/bin/bash
#PBS -S /bin/bash
#
## nodes = quantidade de nós requisitada.
## ppn = quantidade de núcleos por nó.
#PBS -l nodes=1:ppn=16
#
## walltime = quantidade de horas necessárias.
#PBS -l walltime=168:00:00
#
## Nome do job . Aparece na saída do comando 'qstat'.
## É recomendado, mas não necessário, que o nome do job
## seja igual ao nome do arquivo de input.
#PBS -N


# procura o nome o input baseado no nome do job (linha #PBS -N xxx acima).
INP="$PBS_JOBNAME.inp"


## informações do job no arquivo de saída.
qstat -an -u $USER
cat $PBS_NODEFILE


########################################
#-------  Inicio do trabalho  ----- #
########################################

## Configura o nó de calculo.
module load softwares/siesta/3.2-gnu-4.4

cd $PBS_O_WORKDIR

mpirun -np $PBS_NP siesta < $INP > $PBS_JOBNAME-$PBS_JOBID.out
```

Exemplo 3

O job deve executar um programa contra inputs dentro de 10 diretórios diferentes e guardar a saída respectiva dentro do diretório em questão

Supondo que se submeta o job do diretório “x”, que estará guardado dentro da variável $PBS_O_WORKDIR, haverá dentro deste local 10 subdiretórios:

1 2 3 4 5 6 7 8 9 10.

Logo, o programa seria executado 10 vezes, a primeira recebendo como parâmetro de entrada $PBS_O_WORKDIR/1/entrada.txt e como saída $PBS_O_WORKDIR/1/saída.txt

```
#PBS -S /bin/bash
#PBS -N array1-10
#PBS -lnodes=1:ppn=2
#PBS -lmem=4gb
#PBS -t 1-10

cd $PBS_O_WORKDIR/$PBS_ARRAYID

programa < entrada.txt > saída.txt
```

Exemplo 4

Este job é executado contra um input inicial e gera uma saída com um número sequencial que é usado como input para o próximo job. Todos os inputs e outputs são salvos dentro do mesmo diretório.

```
#PBS -S /bin/bash
#PBS -N grafeno
#PBS -lnodes=1:ppn=2
#PBS -lmem=4gb
#PBS -t 1-10
BASEN=grafeno

cd $PBS_O_WORKDIR

programa < $BASEN.$PBS_ARRAYID.txt > $BASEN.`expr $PBS_ARRAYID + 1`.txt
```

Monitorando os Jobs

Para verificar o estado do(s) seu(s) job(s), usa-se o comando "**qstat**". Algumas das opções mais usadas são:

- **"-a"** - Mostrar as informações mais comuns disponíveis (pode ser utilizado conjuntamente com outras opções) .
- **"-u *login*"** - Mostrar somente as informações referentes ao usuário "*login*".
- **"-f *jobid*"** - Mostra informações completas referentes ao job com a id "*jobid*".
- **"-n *jobid*"** - Mostra em qual nó está rodando o job com a id "*jobid*".

Ex:

```
[usuário@cluster-sgi ~]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
48539.cluster-sgi         exemplo1         usuário             02:36:57 R
default
```

Ao executar o comando são mostrados os jobs com as respectivas informações sobre os mesmos. Geralmente, são exibidos o job ID, o nome do job, o usuário dono do job, o tempo de uso de CPU, o estado e a fila. Na coluna com estado do job é mostrado uma letra maiúscula que tem o seguinte significado: "Q" job aguardando execução; "R" job em execução; "S" job suspenso e "E" job em processo de término.

Referência completa (em inglês):
[http://docs.adaptivecomputing.com/torque/Content/topics/commands/qstat.htm]

Também é possível monitorar os nós do cluster, para isso é usado o comando:

```
$ pbsnodes -a
```

Este comando retornará o status de todos os nós, informando quais estão ocupados e quais estão livres, quais jobs estão sendo executados em quais nós e diversas outras informações.

Também pode ser consultado um nó específico com:

```
$ pbsnodes n001
```

Referência completa (em inglês):
[http://docs.adaptivecomputing.com/torque/Content/topics/commands/pbsnodes.htm]

Cancelando Jobs

O comando utilizado para apagar os jobs da fila é o **"qdel"**, a sintaxe é:

```
$ qdel jobid
```

Sendo "jobid" o número que identifica o job quando é aceito e enfileirado.

Existem casos em que o nó onde o job está sendo executado pode “travar”, nesse caso o **“qdel”** não é capaz de matar o job. Nessa situação, deve-se usar o parâmetro **“-p”**, porém só pode ser utilizado pelo administrador (root).

Referência completa(em inglês):
[http://docs.adaptivecomputing.com/torque/Content/topics/commands/qdel.htm]

Compilando seu Próprio Código

Compilação em Geral

É muito comum na hora da compilação, ser necessário ligar o código que está sendo compilado à bibliotecas externas. As opções comuns para fazer essas ligações, em ambos os compiladores são:

- "-L */caminho/da/biblioteca*"
- "-l *biblioteca*"
- "-I */caminho/do/cabeçalho/da/biblioteca/quando/necessário*"

A primeira opção é um "**L**" maiúsculo, a segunda opção um "**L**" minúsculo e a terceira opção um "**i"** maiúsculo. Essas opções são utilizadas na família de compiladores GCC, em qualquer uma das linguagens suportadas.

Um exemplo de link de uma biblioteca externa poderia ser:

```
$ gcc -L /home/usuário/minhasbibliotecas/lib -lbiblioteca1 -lbiblioteca2 -I
/home/usuário/minhasbibliotecas/include -o meuprograma meuprograma.c
```

Em todo o caso, a maioria dos programas utiliza de métodos de configuração automática como automake, "./configure", "Makefile", etc. Em alguns casos essas opções devem ser passadas diretamente para o configurador do programa.

GCC

Para usar os compiladores da família GNU certifique-se que nenhum outro compilador esteja configurado descarregando todos os módulos com o seguinte comando:

```
$ module purge
```

- C:

```
$ gcc -o helloworld helloworld.c
```

- C++:

```
$ g++ -lbibliteca2 -o helloworld helloworld.cpp
```

- Fortran:

```
$ gfortran -o helloworld helloworld.f
```

PGI

Para usar o compilador PGI carregue o módulo relacionado com o seguinte comando:

```
$ module load compilers/pgi-14.10
```

- C:

```
$ pgcc -o helloworld helloworld.c
```

- C++:

```
$ pgc++ -o helloworld helloworld.cpp
```

- Fortran:

```
$ pgfortran -o helloworld helloworld.f/f77/f90
```

Programas MPI

Para utilizar as bibliotecas MPI você deve antes carregar o módulo da mesma. Ex: OpenMPI com compilador GCC

```
$ module load libraries/openmpi-1.5.4/gnu-4.4
```

A compilação de códigos MPI pode ser feita de forma facilitada através dos programas wrappers que já ligam o código às bibliotecas MPI.

Para compilar utilizando esses wrappers, utiliza-se, no lugar do “gcc”, “g++”, “gfortran”, os executáveis “mpicc”,”mpicxx”, “mpif77” e “mpif90”, respectivamente. A sintaxe é a mesma dos compiladores padrões. Ex:

```
$ mpicc -o helloworld helloworld.c
```

Programas que utilizam passagem de mensagens MPI devem ser executados sempre utilizando o “mpirun”:

```
$ mpirun -np 8 meuprograma
```

Tarefas Administrativas

Adicionando Usuários

Para adicionar um novo usuário ao sistema, execute o comando “adduser” e “passwd” para definir a senha do novos usuários. Estes comandos devem ser executados com permissões de administrador (root).

```
[root@cluster-sgi ~]# adduser novo_usuário
[root@cluster-sgi ~]# passwd novo_usuário
```

Assim que o(s) usuário(s) for(em) criado(s), execute o comando abaixo no head-node com permissões de administrador (root) para que as configurações do(s) usuário(s) recém-criado(s) sejam replicadas para todos os nós:

```
[root@cluster-sgi ~]# (cd /var/yp; make)
```

Obtendo Ajuda

Em caso de contrato ativo de **manutenção continuada**, os chamados podem ser abertos por telefone ou através de e-mail fornecendo o teu **ID de cliente** e as **informações abaixo**.

Telefone para **+55 11 2275-9733** ou **+55 81 4062-8443** ou envie e-mail para [**suporte@versatushpc.com.br**] informando:

1. O que você está tentando fazer (exemplos: executar um programa, copiar um arquivo, acessar um diretório, etc);
2. Passos que você está executando para reproduzirmos o problema;
3. Quais as mensagens de erro estão sendo exibidas;
4. Quais os passos alternativos você já tentou e quais os resultados.

Happy Clustering !
Versatus HPC
contato@versatushpc.com.br

Quero Começar a Usar o Cluster mas Estou Ansioso

Como acessar o cluster

O acesso é feito via SSH, utilizando o comando:

```
ssh -p 5523 usuário@200.137.162.107
```

Pode-se também acessar de dentro da rede interna da UFPI, utilizando o comando:

```
ssh -p 5523 usuário@10.40.2.1
```

Como executar comandos em todos os nós

Para executar um comando qualquer em um dos nós do cluster usa-se:

```
[usuário@cluster-sgi ~]$ pdsh -w  hostname “comando parâmetro”
```

Exemplo (executa o comando date no nó n001):

```
[usuário@cluster-sgi ~]$ pdsh -w  n001 “date”
```

Para executar um comando qualquer em todos os nós do cluster usa-se:

```
[usuário@cluster-sgi ~]$ pdsh -a “comando parâmetro”
```

Exemplo (executa o comando uptime em todos os nós do cluster):

```
[usuário@cluster-sgi ~]$ pdsh -a “uptime”
```

Filas

O cluster redireciona os jobs para as filas corretas de acordo com o número de núcleos e tempo solicitados. Não é necessário especificar a fila no momento da submissão.

Como submeter jobs

Existem exemplos de scripts de submissão disponíveis em “/opt/versatushpc/data/jobscripts”. O usuário pode copiá-los e alterá-los como desejar.

Compiladores disponíveis

Estão disponíveis os compiladores GCC e PGI. O GCC é integrado ao sistema e está disponível por padrão. Já para utilizar o compilador PGI execute o comando abaixo:

```
[usuário@cluster-sgi ~]$ module load compilers/pgi-14.10
```

Manipulando seus jobs

Para submeter seus jobs, é usado o comando:

```
[usuário@cluster-sgi ~]$ qsub script.pbs
```

Para verificar o status dos jobs submetidos utlize:

```
[usuário@cluster-sgi ~]$ qstat
```

Para excluir um job da fila:

```
[usuário@cluster-sgi ~]$ qdel jobid
```

Para verificar quais são os nós e suas disponibilidades, usamos o comando:

```
[usuário@cluster-sgi ~]$ qnodes
```