O Cluster FERMI: Guia do Usuário

Lindolfo Meira
Centro Nacional de Supercomputação

3 de abril de 2019

1 Panorama do Sistema

O cluster FERMI opera com o sistema OpenSUSE Leap, versão 15.0, e conta com uma unidade de conexão, um sistema de arquivos paralelo composto por 6 unidades (cada qual contribuindo um arranjo RAID-6, com 16 discos de 4 TB cada, rodando GLUSTER sobre XFS), e 24 unidades de processamento (cada qual com 96 GB de RAM, um processador dodecacore Intel Xeon Silver 4116 de 2.1 GHz de frequência e suporte à *hyper threading*, mais 2 GPUs nVidia Pascal P100). Ao todo, são 576 núcleos de processamento em CPUs, mais 172 mil e 32 núcleos em GPUs, atingindo um desempenho teórico de 466 TFlops (precisão simples). O padrão de conexão da rede de produção no FERMI é InfiniBand, e opera à taxa de 56 Gbps.

2 Acesso Remoto

Para acesso remoto, a partir de uma estação com sistema operacional baseado em UNIX, utiliza-se o programa `ssh`, usando-se a seguinte instrução, digitada diretamente na linha de comando:

```
$ ssh fermi.cesup.ufrgs.br -l username
```

Sendo que `username` (argumento da opção `-l`) deve ser substituído pelo nome de usuário associado à conta disponibilizada pelo CESUP.

De modo similar, a transferência de arquivos desde a estação do usuário até o cluster pode ser feita pelo comando `scp`, da seguinte maneira:

```
$ scp arquivo username@fermi.cesup.ufrgs.br:
```

Isto copiará o arquivo `arquivo` à pasta `$HOME` do usuário `username` no cluster. Ao copiar-se uma pasta inteira, a instrução `scp` deve ser substituída por `scp -r`.

Usuários de sistemas Windows devem instalar em suas estações o cliente SSH/SCP de sua preferência. Uma consulta a qualquer mecanismo de busca na Internet retornará inúmeras opções de clientes para este sistema.

3 Ambiente Computacional

Em virtude das características do hardware, o equipamento é preferencialmente empregado no processamento de códigos que operam em regime de memória compartilhada, com a utilização adicional de aceleradores. Além do kit de desenvolvimento CUDA (Compute Unified Device Architecture), as suítes de compilação GNU e PGI também são disponibilizadas. No entanto, somente a suíte PGI (`pgcc, pg++, pgfortran`) tem suporte ao Open Accelerated Computing (OpenACC). De modo muito similar ao que ocorre no caso do OpenMP, a tecnologia viabiliza a programação de dispositivos aceleradores por meio de diretivas[1], sem a necessidade de alterações drásticas ao código original. A compilação de códigos OpenACC é feita pela inserção do parâmetro `-ta=tesla:cc60` à linha de comando.

No que diz respeito ao processamento em memória distribuída, somente a implementação OpenMPI (não confundir com OpenMP) é disponibilizada. Porém, em duas distribuições distintas: uma compilada inteiramente com a suíte PGI (*default* do ambiente), outra com a suíte GNU. Usuários que necessitarem ou preferirem a distribuição GNU podem alterar seu respectivo ambiente através do comando `mpi-selector-menu`. A compilação de códigos MPI é feita pelos *wrappers* `mpicc/mpic++/mpif90/mpifort`, sem a necessidade de quaisquer parâmetros adicionais à linha de comando.

O sistema de arquivos paralelo abriga a partição com as pastas `$DATA` de cada usuário, de onde as submissões ao sistema de filas devem ser feitas (ver Seção 4). Contudo, como a partição tem caráter temporário e nunca é submetida a rotinas de backup, dados sensíveis devem ser transferidos ao `$HOME` após o processamento.

3.1 Softwares

Uma ampla variedade de softwares voltados ao processamento científico de alto desempenho estão disponíveis aos usuários do cluster FERMI. Entre eles destacam-se o GROMACS, *Groningen Machine for Chemical Simulations*; o NAMD, *Nanoscale Molecular Dynamics*; e o TENSORFLOW, uma biblioteca matemática voltada ao cálculo simbólico, amplamente empregada por aplicações nas áreas de *aprendizagem automática* e *redes neurais*.

3.2 Bibliotecas

Entre as bibliotecas de otimização, destacam-se a suíte Intel Math Kernel Library (MKL), que implementa, entre outros, o Linear Algebra Package (LAPACK), o Basic Linear Algebra Subprograms (BLAS, níveis 1, 2 e 3) e um Random Number Generator (RNG), além de uma interface Fast Fourier Transforms (FFTs), que contém um subconjunto das funções originalmente

[1] Maiores informações em `https://www.openacc.org`

implementadas pela FFTW (Fastest Fourier Transform in the West), a qual também é disponibilizada. O Centro disponibiliza ainda a Data Analytics Acceleration Library (DAAL), a Threading Building Blocks (TBB), a Integrated Performance Primitives (IPP), todas desenvolvidas pela Intel, além do Automatically Tuned Linear Algebra Software (ATLAS), da GNU Scientific Library (GSL), da GNU Multiple Precision Arithmetic Library (GMP), do Basic Linear Algebra Communication Subprograms (BLACS), da Scalable LAPACK (ScaLAPACK), do Network Common Data Form (NetCDF) e do Hierarchical Data Format (HDF5), entre outros. Informações adicionais acerca do modo de utilização destas bibliotecas são fornecidas somente por meio da documentação original, específica a cada uma.

3.3 Cotas

É importante observar que as partições que abrigam as pastas `$HOME` e `$DATA` possuem cotas, e cada usuário é limitado à utilização de 512 GB em cada uma delas. O monitoramento das estatísticas de ocupação das partições deve ser feito com o comando `quota -s`, e é de inteira responsabilidade do usuário, que deve providenciar de antemão a transferência de seus arquivos a mídias externas, de modo a evitar que o volume de seus dados atinja o valor da cota.

4 Sistema de Filas

Todas as tarefas computacionais executadas no FERMI, que opera com a versão 18.1.3 do Altair PBS-Pro, devem ser submetidas ao processamento somente via sistema de filas (com exceção de compilações e afins). Os comandos de submissão e monitoramento dos jobs são, respectivamente, o `qsub` e o `qstat` (para maiores informações, acesse os manuais online digitando, após a conexão, `man` seguido do nome do comando de interesse).

Para submeter um job, após transferir-se à sua respectiva pasta na partição de dados (`cd $DATA`), o usuário deve editar o modelo de script lá existente (`script.sh`) adaptando-o a seu respectivo caso. Feito isso, basta invocar o comando `qsub`, informando o nome do arquivo recém editado como argumento. Após a submissão, o usuário tem seu *prompt* de comando liberado, podendo desconectar-se do cluster sem comprometer seu trabalho. Abaixo uma versão comentada do arquivo `script.sh`:

```
#!/bin/sh
# A linha acima, incluindo o símbolo #, define a sintaxe ado-
# tada no script e deve ser sempre a primeira linha do mesmo,
# não podendo ser precedida por espaços ou linhas em branco.

# Linhas que começam com "#PBS" são interpretadas como coman-
```

```
# dos internos pelo PBS. Linhas que começam somente com # são
# tratadas como comentários (exceto no caso da primeira linha
# do script), e linhas em branco são totalmente ignoradas pelo
# sistema.

# Definição do interpretador utilizado internamente pelo PBS e
# do nome que se queira dar ao job (não deve conter espaços).
#PBS -S /bin/sh
#PBS -N job_exemplo

# Requisição de alocação de 6 núcleos e uma GPU para um código
# com suporte a processamento em memória compartilhada e acele-
# ração, com concatenação do erro padrão na saída padrão (-j).
#PBS -l select=1:ncpus=6:ngpus=1
#PBS -j oe

# Digamos que seu executável esteja em /dados/$USER/tmp e que
# seu script é submetido a partir desta pasta. Mesmo adotando-
# a como pasta de trabalho, o PBS não transfere a ela a execu-
# ção do script, senão pela invocação ipsis litteris do coman-
# do abaixo.
cd $PBS_O_WORKDIR

# Linha de disparo de um programa com suporte a processamento
# acelerado (note que se o símbolo ./ não for prefixado ao no-
# me do seu executável, é necessário então fornecer o caminho
# absoluto do arquivo).
./programa_openacc
```

Note-se que as linhas sem caracteres especiais no início são comandos normais do sistema operacional (obedecendo a sintaxe do interpretador definido na primeira linha do script) e são absolutamente transparentes ao PBS.

Aplicações que operam puramente em regime de memória compartilhada e/ou aceleração necessitam de um esquema de alocação do tipo *fill up*. Logo, o argumento do parâmetro `select`, na instrução `-l`, assume sempre o valor 1 no caso de jobs OpenMP ou OpenACC, enquanto o argumento do parâmetro `ncpus` pode variar até o número máximo de núcleos de processamento nos nós do sistema. O PBS encarrega-se de estabelecer o valor de `OMP_NUM_THREADS` de acordo com o valor de `ncpus`. Para aplicações OpenACC/CUDA com o devido suporte, `ngpus` pode assumir o valor 2, mas na maioria dos casos assume o valor 1.

No caso do MPI (processamento em memória distribuída), o emprego de uma regra de alocação do tipo *round robin* pode ajudar a minimizar o tempo de espera pela disponibilidade do recurso. Nestes casos, o valor atribuído

à `select` deve refletir o número de núcleos de processamento desejado e o parâmetro `ncpus` pode ser omitido, pois assume automaticamente o valor 1. Convém mencionar que programas MPI precisam ser invocados pelo *wrapper* `mpiexec/mpirun`.

Para processamento híbrido (em memória distribuída e compartilhada, mais aceleração), como no caso do GROMACS no cluster FERMI, usa-se uma forma mista da instrução `-l`, que permite que cada *rank* MPI distribua sua carga de processamento entre *threads* OpenMP, que por sua vez podem atribuir parte da carga deste processamento aos aceleradores (i.e., *descarregar*). A regra de alocação nestes casos incluiria os parâmetros `select`, `mpiprocs`, `ncpus` e `ngpus`. O parâmetro `select` deve ser pensado como o número de nós computacionais a ser alocado, e serve como fator multiplicativo aos demais parâmetros. O parâmetro `mpiprocs` representa o número de *ranks* MPI que se deseja alocar. O parâmetro `ncpus` representa o número de *threads* OpenMP (núcleos de processamento) sobre as quais cada *rank* MPI distribuirá seu processamento. E o parâmetro `ngpus` representa o número de aceleradores que se deseja alocar em cada nó.

O monitoramento do *status* dos jobs no sistema de filas é feito pelo comando `qstat`. Quando disparado sem argumentos, o sistema retorna o status de todos os jobs em processamento no momento do disparo. Empregando-se o parâmetro `-u user`, o sistema apresenta somente o status do(s) job(s) do usuário `user`. O parâmetro `-f JOB_ID` fornece informações detalhadas acerca do job `JOB_ID`. Para cancelar um job em espera ou execução utiliza-se `qdel JOB_ID`, sendo `JOB_ID` o número de referência do job, assim como retornado pelo `qstat`. Uma série de parâmetros adicionais podem ser utilizados com ambos os comandos aqui mencionados. Para informações mais detalhadas, sugere-se a leitura dos manuais online (i.e., uma vez conectado digite, na linha de comando do terminal, `man comando_de_interesse`).

```
if [ $BUG || $? ]; then mail meira@cesup.ufrgs.br; fi ☺
```