O Cluster GAUSS: Guia do Usuário

Lindolfo Meira
Centro Nacional de Supercomputação

22 de março de 2019

1 Panorama do Sistema

O cluster GAUSS opera com o sistema Novell SUSE Linux Enterprise Server 11-SP1 e conta, basicamente, com uma unidade de conexão e 64 unidades de processamento. Cada qual com 64 GB de RAM e 2 processadores dodecacore AMD Opteron (32 unidades com o modelo 6176 SE, de 2.3 GHz e 32 outras com o modelo 6238, de 2.9 GHz de frequência), totalizando 1536 núcleos de processamento e um desempenho teórico de 24 TFlops (precisão simples). Cada uma das 64 unidades de processamento possui discos locais dedicados a *scratch*, montados no endereço `/tmp/scratch`. O padrão de conexão das redes de produção no GAUSS é InfiniBand, e cada uma opera à taxa de 40 Gbps (são 2 redes de produção: uma dedicada a I/O, outra à troca de mensagens por processos operando em memória distribuída). Apesar de as submissões ao sistema de filas precisarem ser feitas sempre a partir da pasta `$DADOS` de cada usuário (ver Seção 4), dados sensíveis devem, depois de processados, ser transferidos ao `$HOME`. A partição que armazena as pastas `$DADOS` tem caráter temporário e, portanto, nunca é submetida a rotinas de backup.

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 gauss.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@gauss.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

O sistema foi projetado de modo a dispensar a necessidade de interação do usuário com o processo de configuração do ambiente computacional. No que diz respeito ao processamento em memória distribuída, três diferentes implementações da interface de troca de mensagens (MPI) são disponibilizadas: MPT, IMPI e OpenMPI. A MPT (Message Passing Toolkit) é baseada em software GNU, e otimizada pela própria SGI para o hardware do GAUSS. A IMPI baseia-se em software Intel com algumas extensões GNU; e a OpenMPI baseia-se em software GNU, mas é disponibilizada no CESUP em duas distribuições distintas: uma compilada inteiramente com a suíte GNU, outra com a suíte Intel. O ambiente padrão utiliza MPT. Usuários que necessitarem ou preferirem as implementações alternativas podem alterar seu respectivo ambiente através do comando `mpi-selector-menu`.

Em contraste com o que acontece com outras implementações MPI, a MPT não possui os *wrappers* `mpicc/mpif90` e afins. Assim sendo, a compilação de códigos MPI é feita pelos compiladores convencionais, com a *linkagem* declarada explicitamente na linha de comando, como segue:

```
$ compilador -lmpi nome_programa
```

O termo `compilador` no exemplo representa quaisquer um dos compiladores das suítes disponíveis. A saber: `gfortran`, `g++` e `gcc` (GNU); e `ifort`, `icpc` e `icc` (Intel). A compilação de códigos MPI em C++ requer, adicionalmente, a inserção do parâmetro `-lmpi++` à linha de comando. No caso de processamento em memória compartilhada via OpenMP (não confundir com OpenMPI), o parâmetro `-lmpi` deve ser substituído por `-fopenmp` quando o compilador utilizado for GNU, e `-openmp` quando for Intel.

3.1 Softwares

Uma ampla variedade de softwares voltados ao processamento científico de alto desempenho estão disponíveis aos usuários do GAUSS. Entre eles destacam-se o o GAMESS, *General Atomic and Molecular Electronic Structure System*; o GAUSSIAN, software orientado à modelagem de estruturas eletrônicas; o SIESTA, *Spanish Initiative for Electronic Simulations with Thousands of Atoms*, compilado com suporte a cálculos de transporte

balístico eletrônico (TranSIESTA); e o QE (*Quantum Espresso*), uma suíte voltada ao cálculo de estruturas eletrônicas e modelagem de materiais em nanoescala, cujas particularidades acerca do modo de operação fazem com que seja necessário o emprego de técnicas instrumentais bastante específicas. Usuários deste software, após a leitura integral deste manual, devem contatar a equipe de suporte (`suporte@cesup.ufrgs.br`), solicitando informações acerca dos procedimentos adequados à sua utilização.

3.2 Bibliotecas

Entre as bibliotecas de otimização, destacam-se a suíte AMD Core Math Library (ACML), 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 implementadas pela FFTW (Fastest Fourier Transform in the West), a qual também é disponibilizada. A ACML é equivalente à Math Kernel Library (MKL), mas otimizada para o hardware AMD (a despeito de a MKL ser otimizada para os processadores Intel, o CESUP também a disponibiliza). O Centro disponibiliza ainda o Automatically Tuned Linear Algebra Software (ATLAS), a GNU Scientific Library (GSL), a GNU Multiple Precision Arithmetic Library (GMP), o Basic Linear Algebra Communication Subprograms (BLACS), a Scalable LAPACK (ScaLAPACK), o Network Common Data Form (NetCDF) e o 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 `$HOME` e `$DADOS` 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 GAUSS, que opera com a versão 11.0.2 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).

Usuários dos softwares SIESTA e QE devem submeter seus jobs, respectivamente, às filas `siesta.q` e `quantum.q`, substituindo a linha reservada `SIESTA/QE` no script abaixo, pela instrução `#PBS -q siesta.q` ou `quantum.q`, conforme o caso. Os demais usuários não precisam se preocupar com a seleção de filas, que é feita automaticamente pelo sistema.

Para submeter um job, após transferir-se à sua respectiva pasta na partição de dados (`cd $DADOS`), 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 adotada
# no script e deve ser sempre a primeira linha do mesmo, não podendo
# ser precedida nem mesmo por espaços ou linhas em branco.
#
# Linhas que começam com "#PBS" são interpretadas como comandos
# 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
#
# Linha reservada SIESTA/QE
#
# Requisição de alocação de 48 slots para um programa MPI e
# concatenação do erro padrão na saída padrão (-j).
#PBS -l select=48
#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 execução do
# script, senão pela invocação do comando abaixo
cd $PBS_O_WORKDIR
#
# Linha de disparo de um programa com suporte a processamento
```

```
# distribuído (note que se o símbolo ./ não for prefixado ao nome
# do seu executável, é necessário então fornecer o caminho absoluto
# do arquivo).
mpiexec ./programa_mpi
```

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.

O argumento `select`, na instrução `-l`, estabelece que o sistema deve empregar uma regra de alocação do tipo `ROUND_ROBIN` (que minimiza o tempo de espera pela disponibilidade do recurso). Nestes casos, o valor atribuído à `select` deve refletir o número de núcleos de processamento desejado. Uma regra de alocação do tipo `FILL_UP` é estabelecida pelo esquema `nodes/ppn`, onde `nodes` deve assumir o valor do número de nós computacionais pelos quais se deseja distribuir o job, e `ppn` assume o valor do número de núcleos em cada nó (24 é o máximo no caso desta máquina). Logo, a alocação de, por exemplo, 48 slots pela regra `FILL_UP` daria-se pela substituição da instrução `#PBS -l select=48` por `#PBS -l nodes=2:ppn=24`.

No caso do OpenMP (processamento em memória compartilhada), usa-se o esquema `nodes/cpp`, sendo que o valor de `nodes` deve ser mantido sempre em 1, enquanto o valor de `cpp` pode escalar até 24 (número máximo de núcleos de processamento dos nós). O sistema trata de estabelecer o valor de `OMP_NUM_THREADS` de acordo com o valor de `cpp`. Inútil dizer que programas escritos em OpenMP são disparados invocando-se somente o nome do executável, sem `mpiexec`.

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 ☺
```