Carbono

Guia do Usuário

versatus
HPC

Published by VersatusHPC

VERSATUSHPC.COM

Revisão 2. 05 Fevereiro, 2024

Contents

versatus
HPC

I

Descrição do *Cluster*

1. *Hardware*

1.1 *Headnode*

1 (um) servidor Dell EMC PowerEdge R7525 configurado com:

- 2 (dois) processadores AMD EPYC 7313
- 128 GB de memória RAM DDR4
- 1 (um) BOSS com 2x SSDs de 480GB em RAID 1
- 8 (oito) SSDs de 8TB
- 2 (duas) portas 1 Gigabit Ethernet
- 2 (duas) portas 25 Gigabit Ethernet
- 1 (uma) porta Mellanox InfiniBand ConnectX

1.2 Nós Computacionais

Tipo A) 4 (quatro) servidores Dell EMC PowerEdge R6525 configurados com:

- 2 (dois) processadores AMD EPYC 7763
- 512 GB de memória RAM DDR4
- 2 (duas) portas 1 Gigabit Ethernet
- 2 (duas) portas 10 Gigabit Ethernet

Tipo B) 3 (três) servidores Dell EMC PowerEdge R6525 configurados com:

- 2 (dois) processadores AMD EPYC 7763
- 512 GB de memória DDR4
- 2 (duas) portas 1 Gigabit Ethernet
- 2 (duas) portas 10 Gigabit Ethernet
- 2 (duas) portas Mellanox InfiniBand ConnectX

Tipo C) 1 (um) servidor Dell EMC PowerEdge R7525 configurado com:

- 2 (dois) processadores AMD EPYC 7763
- 512 GB de memória DDR4
- 2 (duas) portas 1 Gigabit Ethernet
- 2 (duas) portas 10 Gigabit Ethernet
- 2 (duas) portas Mellanox InfiniBand ConnectX
- 2 (duas) GPUs NVIDIA A30.

Tipo D) 1 (um) servidor Dell EMC PowerEdge R7525 configurado com:

- 2 (dois) processadores AMD EPYC 7763
- 512 GB de memória DDR4
- 2 (duas) portas 1 Gigabit Ethernet
- 2 (duas) portas 10 Gigabit Ethernet
- 2 (duas) portas Mellanox InfiniBand ConnectX
- 2 (duas) GPUs NVIDIA A40

2. Tabela de Particionamento

Headnode:

```
Filesystem      Type     Size  Used Avail Use% Mounted on
/dev/sda4       xfs      442G   67G  375G  16% /
/dev/sda2       xfs     1014M  428M  587M  43% /boot
/dev/sda1       vfat     599M  5.1M  594M   1% /boot/efi
beegfs_nodev    beegfs   582T  4.1T  578T   1% /home
```

3. Arquivos e Armazenamento

O diretório padrão do usuário é construído da seguinte forma:

`/home/<usuário>/`

Ex: `/home/vhpc/`

Esta área possui **582 TB de espaço disponível** que são compartilhados entre todos os usuários.

4. Nós Computacionais

Os nós desse *cluster* nomeados no formato "n0X". Por exemplo, `n03`.
O *cluster* Carbono possui 9 nós computacionais, conforme mencionado na subseção 1.2:

Hostname	**Tipo de máquina**
n[01-02,06-07]	A
n[03-05]	B
gn01	C
gn02	D

5. Redes

Os nós estão conectados a 2 redes distintas que são relacionadas apenas para completude das informações:

1. **Gigabit Ethernet**
2. **IPMI**

A rede IPMI é utilizada pelos procedimentos administrativos do cluster via protocolo IPMI, enquanto a rede GbE são utilizadas para trocas de dados entre os nós e também mensagens MPI.

6. Formas de acesso

O acesso remoto é feito através de SSH (*Secure Shell*):

```
ssh <usuario>@carbono.ufabc.int.br
```

versatus
HPC

II

Softwares Instalados

7. *Softwares* Instalados

O sistema tem um grande conjunto de softwares instalados de acordo com as necessidades dos usuários. Os softwares disponíveis em produção deste guia estão listados abaixo.

Todos os softwares, compiladores e bibliotecas pré-instalados no cluster estão disponíveis em '`/opt/ohpc/pub/`' e '`/opt/spack/opt/spack/linux-oracle8-x86\_64/`'.

7.1 Compiladores

A família de compiladores GNU e Intel e suas respectivas ferramentas, estão disponíveis nas seguintes versões:

7.1.1 GNU

- **Versão 8.5.0**

  Binários: `gcc`, `gfortran` e `g++`

- **Versão 12.2.0**

  Binários: `gcc`, `gfortran` e `g++`
  Module: `gnu12/12.2.0`

7.1.2 Intel oneAPI

- **Versão 2023.2.0**

  Module: `intel/2023.1.0`

7.2 Bibliotecas de Transferência de Mensagem

As bibliotecas MPI pré-instalados na *workstation* são as seguintes:

7.2.1 Intel MPI

- **Versão 2021.9 compilada com Intel oneAPI 2023.1**

  Module: `mpi/2021.9.0`

7.2.2 OpenMPI

OpenMPI é uma biblioteca de troca de mensagens de código aberto implementada pela OpenMPI Project:

- **Versão 4.1 compilada com GNU 12.2.0**

  Module: `openmpi4/4.1.4`

7.2.3 MPICH

MPICH é outra biblioteca de troca de mensagens aberta existente:

- **Versão 3.4 compilada com GNU 12.2.0**

  Module: `mpich/3.4.3-ucx`

7.3 Bibliotecas Matemáticas

Estão pré-instalados no *cluster* as seguintes bibliotecas:

7.3.1 FFTW

Essa biblioteca são sub-rotinas para cálculo das transformadas rápida e discreta de Fourier, além de transformadas discretas do seno e cosseno e transformadas de Hartley.

- **Versão 3.3.10 compilada com GNU 12.2.0**

  Module: `fftw/3.3.10`

7.3.2 HDF5

Hierarchical Data Format nome para um conjunto de formatos de arquivos e bibliotecas criadas para organização e armazenamento de grandes quantidades de dados numéricos.

- **Versão 1.10.8 compilada com GNU 12.2.0**

  Module: `hdf5/1.10.8`

7.3.3 Intel MKL

- **Versão 2023.1 compilada com Intel oneAPI 2023.1**

  Module: `mkl/2023.1.0`

7.3.4 NetCDF

Network Common Data Form (netCDF) é um conjunto de bibliotecas de softwares criadas para melhor organização de dados científicos.

- **Versão 4.9.0 compilada com GNU 12.2.0**

  Module: `netcdf/4.9.0`

7.3.5 OpenBLAS

O BLAS (*Basic Linear Algebra Subprograms*) são rotinas que fornecem blocos de construção padrão para a realização de operações básicas vetoriais e matriciais.

- **Versão 0.3.21 compilada com GNU 12.2.0**

  Module: `openblas/0.3.21`

7.3.6 PnetCDF

PnetCDF é uma bilbioteca de E/S (entrada/saída) paralelo de alta desempenho para acessar arquivos compatíveis com NetCDF.

- **Versão 1.12.3 compilada com GNU 12.2.0**

  Module: `pnetcdf/1.12.3`

7.3.7 ScaLAPACK

O *Scalable* LAPACK é uma biblioteca de rotinas de alto desempenho para álgebra linear baseada no LAPACK.

- **Versão 2.2.0 compilada com GNU 12.2.0**

  Module: `scalapack/2.2.0`

7.4 *Softwares* Científicos

Os *softwares* científicos instalados são mostrados em cada tópico abaixo com seus respectivos módulos de ambiente para carregamento via comando `module load`:

7.4.1 AMBER

- **Versão ZZ compilada com GNU 12.2.0**

  Module:

7.4.2 BoltzTraP2

- **Versão 22.12.1 compilada com GNU 12.2.0**

  Module: `boltztrap2/22.12`

7.4.3 Gaussian'09

- **Versão D.01**

  Module: `gaussian/09d01`

7.4.4 GROMACS

- **Versão 2023.1 compilada com GNU 12.2.0**

  Module: `gromacs/2023.1-gcc-12.2.0-lamolab`

7.4.5 LAMMPS

- **Versão 20220623.3 compilada com GNU 12.2.0**

  Module: `lammps/20220623.3-gcc-12.2.0-f5imquc`

- **Versão 20220623.3 com suporte à CUDA compilada com GNU 12.2.0**

  Module: `lammps/20220623.3-gcc-12.2.0-q7ehxc3`

7.4.6 ORCA

- **Versão 5.0.4**

  Module: `orca/5.0.4`

7.4.7 Quantum ESPRESSO

- **Versão 7.2 compilada com GNU 12.2.0**

  Module: `quantum-espresso/7.2-gcc-12.2.0-ohadjws`

- **Versão 7.2 com suporte à CUDA compilada com GNU 12.2.0**

  Module: `quantum-espresso/7.2-cuda`

7.4.8 SIESTA

- **Versão 4.0.2 compilada com GNU 12.2.0**

  Module: `siesta/4.0.2-gcc-12.2.0-p5cyvtd`

7.4.9 VASP

- **Versão 6.2.0 compilada com GNU 12.2.0**

  Module: `vasp/6.2.0`

- **Versão 6.2.0 com suporte à CUDA compilada com GNU 12.2.0**

  Module: `vasp/6.2.0-cuda`

III

Variáveis de Ambiente

8. 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 C/C++ e MPI, temos disponíveis “gcc” e “mpicc”. Além dessa variação de softwares que desempenham a mesma função, também é comum “colecionar” 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 Carbono 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 de ambiente que você necessita podem ser carregados manualmente na linha de comando. Ainda assim, é mais conveniente incluir os módulos para serem carregados nos scripts de submissão que você utilizar. Outra possibilidade é modificar o seu “`.bash_profile`” para carregar os módulos automaticamente no login (não esqueça de descarregar os módulos se há algum conflito com algum outro módulo que você precisa para um software específico).

Os módulos de variáveis atualmente disponíveis são:

```
[vhpc@carbono ~]$ module avail
...
```

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.

Exemplos:

Cenário 1: Compilar um programa utilizando o GCC 12.2, para esta compilação o usuário deve carregar o módulo do compilador com o comando:

```
[vhpc@carbono ~]$ module add gnu12/12.2.0
```

Cenário 2: O usuário deseja remover todos os módulos já carregados:

```
[vhpc@carbono ~]$ module purge
```

versatus
HPC

IV

Utilizando o *cluster*

9. Utilizando o *cluster*

9.1 Sistema Operacional

A versão do seu sistema operacional é o Oracle Linux 8.8 x86_64, distribuição baseada no Red Hat Enterprise Linux.

versatus
HPC

V

Gerenciamento de *Jobs*

10. Submetendo e Gerenciando *Jobs*

O Carbono conta com o sistema de filas instalado para submeter e administrar jobs via scripts. Para que os recursos computacionais sejam divididos de maneira razoável entre todos os usuários, é necessária uma forma de organizar e priorizar as requisições de uso, comumente chamadas de "jobs". Para isso, o cluster conta com o sistema de filas SLURM para gerenciar os jobs que serão executados, respeitando as políticas estabelecidas. O SLURM trabalha com o conceito de filas, que são estruturas para classificar e agrupar os jobs e nós computacionais sob critérios como: número de processadores requisitados, quantidade de nós, quantidade de memória RAM, tempo de processamento e assim por diante. Na configuração atual, estão configuradas partições com ordem de prioridade e em que apenas alguns grupos de usuários têm acesso a algumas delas (na seção 10.2 temos mais detalhes).

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

```
[vhpc@carbono ~]$ sinfo -a
```

O acesso via SSH aos nós é controlado pelo sistema de filas, ou seja, usuários comuns só serão permitidos acessar os nós se houver(em) *job*(s) em execução nestes nós.

10.1 Diretivas e Variáveis do *Job*

Dentro dos jobs existem algumas instruções que são dedicadas ao SLURM, 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: `#SBATCH`. O SLURM irá considerar todas as diretivas até que encontre uma linha executável, depois deste ponto, será ignorada qualquer outra diretiva. Diretivas comuns são:

`#SBATCH --job-name=<nome>` ⟶ Nome do job
`#SBATCH --ntasks=<número>` ⟶ Número total de núcleos
`#SBATCH --time=<dd-hh:mm:ss>` ⟶ Tempo de execução (dias-horas:min:seg)
`#SBATCH --partition=<fila>` ⟶ Fila onde o job será alocado

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

`$SLURM_JOB_NODELIST` ⟶ esta variável contém a lista de nós e núcleos que foram alocadas ao job. É útil, por exemplo, para ser usada como parâmetro para o `mpirun`:

```
$ mpirun -machinefile $SLURM_JOB_NODELIST ./meuprograma
```

`$SLURM_SUBMIT_DIR` ⟶ esta variável aponta para o diretório de onde o script foi submetido.

`$SLURM_JOB_NAME` ⟶ contém o nome especificado na diretiva `#SBATCH --job-name=`

A lista completa de diretivas, opções e variáveis pode ser encontrada em: [https://slurm.schedmd.com/sbatch.html].

10.2 Filas

As filas são o que permitem um bom desempenho, distribuição e agendamento de uso de recursos. No Carbono temos apenas as filas configuradas listadas abaixo:

Fila	Tempo limite	Mín. de núcleos alocados por *job*	Máx. de núcleos alocados por job	Máx. de núcleos alocados por usuário	Nó(s) associado(s)	Observação
grafite	1 hora	-	2	128	n[01-05]	sem acesso à GPU
grafeno	2 dias	-	16	128	gn[01-02]	-
nanotubo	2 dias	17	64	128	gn[01-02]	-
fulereno	2 dias	64	128	128	gn[01-02]	-
diamante	5 dia	128	384	-	gn[01-02]	apenas grupo "ccm_carbono_FINEP"
metano	2 dia	-	64	128	gn[01-02]	-
etileno	2 dia	-	64	128	gn[01-02]	-

10.3 *Scripts* de submissão

Um exemplo de *script* de submissão de *jobs* está localizado em `/opt/versatushpc/data/jobscripts/`. O *script* `template.slurm` pode ser utilizado como modelo para criação de *scripts* personalizadas para os *softwares*.

10.4 Tipos de *Jobs*

Quanto aos *jobs*, existem dois tipos mais comuns:

10.4.1 *Batch Jobs*

Um arquivo de *script* que contém comandos para executar aplicações específicas, usado pelo programa `sbatch` para informar ao SLURM as características do seu *job*. Exemplo de *script*:

```
#!/bin/bash
## Copyright (C) 2009-2024 VersatusHPC, Inc.

#SBATCH --time=24:00:00
#SBATCH --ntasks=8
#SBATCH --job-name=teste

cd $SLURM_SUBMIT_DIR

### Prepare o ambiente com os devidos modulos ###
module add openmpi4/4.1.4

### Exporte as variaveis relevantes para o job ###
```

```
export OMP_NUM_THREADS=1

### Execucao do job ###
mpirun -np $SLURM_NTASKS prime 200000 > teste.out
```

- A primeira linha informa ao SLURM 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 o número de núcleos para este job;
- A variável `$SLURM_SUBMIT_DIR` 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/exportadas antes do job.

Exemplo da submissão de um batch job:

```
[vhpc@carbono ~]$ sbatch script.slurm
```

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

O exemplo abaixo mostra um job simples de execução de um *software* genérico em 20 núcleos e *walltime* de 2 horas:

```
#/bin/bash
## Copyright (C) 2009-2024 VersatusHPC, Inc.

#SBATCH --job-name=TESTE
#SBATCH --ntasks=20
#SBATCH --time=2:00:00

module load openmpi4/4.1.4

cd $SLURM_SUBMIT_DIR

mpirun -np $SLURM_NTASKS ./meuprograma
```

Exemplo 2

O job abaixo é um exemplo de execução do software em 16 núcleos de 1 nó e walltime de 30 horas:

```
#!/bin/bash
## Copyright (C) 2009-2024 VersatusHPC, Inc.
##
#SBATCH --nodes=1              # quantidade de nodes
#SBATCH --ntasks-per-node=16   # quantidade de processos por node
#SBATCH --time=30:00:00        # quantidade de tempo
```

```
#SBATCH --job-name=<Job_Name> # nome do job

echo -e "\n[+] Job iniciado em $(date +'%d-%m-%Y as %T')\n"

## Variaveis de ambiente
SCRATCH=/scratch/local
WRKDIR=$SCRATCH/$SLURM_JOB_ID

# nome dos arquivos de input e output (se baseados no jobname)
INP=$SLURM_JOB_NAME".inp"
OUT=$SLURM_JOB_NAME".out"

# O diretorio onde o job sera executado sera apagado, por padrao
APAGA_SCRATCH=Y # 'Y' para apagar e 'N' para nao apagar

# Informacoes do job impressos no arquivo de saida.

echo -e "\n[+] Jobs ativos de $USER: \n"
squeue -a --user=$USER
echo -e "\n[+] Node de execucao do job:         $(hostname -s)\n
   "
echo -e "\n[+] Numero de tarefas para este job: $SLURM_NTASKS\n"

### INICIO DO TRABALHO

# Configura o ambiente de execucao do software.
module load @SOFTWARE_MODULE@

# Informacoes sobre o ambiente de execucao impressos no arquivo
   de saida.
echo -e "\n[+] Diretorio de submissao do job:
   $SLURM_SUBMIT_DIR \n"
echo -e "\n[+] Diretorio de scratch do job:     $WRKDIR \n"
echo -e "\n[+] Arquivo de input:                $INP \n"

# Cria o diretorio de scratch para o job.
mkdir -p $WRKDIR

# Transfere os inputs e arquivos necessarios para o diretorio de
    scratch
cd $SLURM_SUBMIT_DIR
cp $INP $WRKDIR/
cd $WRKDIR

# Execucao do software
mpirun ./mysoftware < $INP > $OUT

# Copia o diretorio de scratch para o diretorio original do job.
cp -r $WRKDIR/ $SLURM_SUBMIT_DIR/
```

```
# Apaga o diretorio de scratch do job.
if [ x"$APAGA_SCRATCH" = x"Y" ]; then
    rm -rf $WRKDIR
else
    echo -e "\n[+]O diretorio \e[00;31m$WRKDIR\e[00m deve ser
       removido manualmente para evitar problemas para outros
       jobs e/ou usuarios.\n"
fi
echo -e "\n[+] Job finalizado em $(date +'%d-%m-%Y as %T')\n"
```

10.4.2 *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.

Para submeter um *job* interativo basta executar o comando abaixo:

```
[vhpc@carbono ~]$ srun --pty bash -i
```

Para encerrar o job interativo, basta sair do terminal com o comando '`exit`'.

Outras opções podem ser utilizadas no comando '`srun`', referência completa: https://slurm.schedmd.com/srun.html.

10.4.3 Monitorando os seus *Jobs*

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

- `--user=<usuario>` ⟶ mostra somente as informações referentes ao usuário `usuario`
- `--jobs=<jobid>` ⟶ mostra informações completas referentes ao job com a id `jobid`
- `--name=<jobname>` ⟶ mostra informações completas referentes ao job com o nome `jobname`

Referência completa em: https://slurm.schedmd.com/squeue.html.

10.4.4 Cancelando *Jobs*

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

- `scancel <jobid>` ⟶ jobid é o número que o SLURM retorna quando o job é aceito e enfileirado
- `scancel --jobname=<jobname>` ⟶ cancela o job com nome "jobname"
- `scancel --usage` ⟶ mostra brevemente as opções do comando

Referência completa em: https://slurm.schedmd.com/scancel.html.

10.5 Problemas, Resoluções e Dúvidas

A documentação oficial do SLURM mantém uma compilação de problemas comuns e suas resoluções como também dúvidas mais frequentes sobre o sistema de filas.

Para problemas e resoluções: https://slurm.schedmd.com/troubleshoot.html.

Para dúvidas e questões frequentes: https://slurm.schedmd.com/faq.html.

versatus
HPC

VI

Obtendo Ajuda

11. Obtendo Ajuda

Em caso de contrato ativo de manutenção continuada os chamados podem ser abertos por telefone (+55 11 3436-0664) ou através de e-mail [suporte@versatushpc.com.br], fornecendo o ID do cliente e as informações abaixo:

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