MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2017/07.07.22.07-NTC

GUIA BÁSICO PARA CLUSTER HPC-PLASMAS NO INPE - VERSÃO 1

Varlei Everton Menconi - Desenvolvedor
Odim Mendes Júnior - Supervisor
Margarete Oliveira Domingues - Co-supervisora

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3P8GT3S>

INPE
São José dos Campos
2017

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2017/07.07.22.07-NTC

GUIA BÁSICO PARA CLUSTER HPC-PLASMAS NO INPE - VERSÃO 1

Varlei Everton Menconi - Desenvolvedor
Odim Mendes Júnior - Supervisor
Margarete Oliveira Domingues - Co-supervisora

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3P8GT3S>

INPE
São José dos Campos
2017

GUIA BÁSICO PARA CLUSTER HPC-PLASMAS NO INPE, versão 1

Um guia básico para acesso e uso dos clusters
HPC PLASMAS
(orion-d11.cea.inpe.br e helios.cea.inpe.br)

Índice dos Tópicos abordados

Índice de Tabelas

1) Sistemas de Cluster e suas caracteríticas

Cluster (ou clustering) é, em poucas palavras, o termo técnico dado a uma estrutura computacional ou sistema que relaciona um **conjunto de computadores e outros recursos (hardwares e middlewares)** para que estes passem a trabalhar de maneira conjunta, porém coordenada, no intuito de processar uma determinada tarefa.

Em um cluster, dois termos que se sobressaem nesta arquitetura são:

1.1) Head-Node

Onde um dos servidores de processamento deste conjunto é dedicado exclusivamente a administrar e gerenciar todo o cluster. No head-node são instalados uma camada de middlewares responsáveis por monitorar e gerenciar os processos em execução; conexões e usuários no sistema; monitorar os outros hardwares que compõem o sistema; gerenciar outros recurssos disponiveis, etc.

1.2) Work-Nodes

É o termo dado aos outros computadores ou recursos de processamento do conjunto do cluster, e que serão dedicados apenas para o processamento das tarefas em execução.

Geralmente, uma camada de software no head-node divide o processamento de uma determinada tarefa/aplicação em blocos de processamento menores e atribui a execução destes vários blocos aos work-nodes, paralelamente. O head-node monitora estes nodes e, após a conclusão de todos os blocos, ter-se-á o processamento da tarefa/aplicação como um todo.

Neste conceito, há vários tipos de clusters, disponiveis para diferentes objetivos. Por exemplo:

1.1) Tipos de clusters

1.1.1 - Clusters de Alta Disponibilidade (HAC - High Availability Computing Cluster)

Onde o objetivo está em sempre manter uma aplicação crítica em pleno funcionamento e que exigem disponibilidade de pelo menos 99,999% do tempo a cada ano. Não é aceitável que estes sistemas parem de funcionar. Como exemplo, pode-se citar aplicações ecônomicas e bancárias, sistemas web, etc.

Para atender a esta exigência, os clusters HAC podem contar com diversos recursos: ferramentas de monitoramento que identificam nós defeituosos ou falhas na conexão, replicação (redundância) de sistemas e computadores para substituição imediata de máquinas com problemas, uso de geradores para garantir o funcionamento em caso de queda de energia, entre outros recursos.

Em determinadas circunstâncias, é permissível que o sistema apresente algum grau de perda de desempenho, especialmente quando esta situação é consequência de algum esforço para

manter a aplicação em atividade em eventual manutenção do sistema.

1.1.2 - Clusters de Balanceamento de Carga (LBC - Load Balancing Cluster)

São clusters onde são executadas aplicações que atendam à um volume muito grande de requisições. Assim, este tipo de cluster (LBC) tem a capacidade de receber os pedidos de requisição de serviços de vários clientes e distribuir (balancear a carga de trabalho) este tráfego / volume entre os nodes que estão executando estas mesmas aplicações.

Desta forma, cada node recebe e responde à uma cota do volume de requisições, evitando delays no retorno das requisições e/ou principalmente, sobrecarga de requisições em uma única máquina.

Clusters deste tipo são frequentemente encontrados em projetos / sistemas de Web Hosting ou Comércio Eletrônico; aplicações que requerem interações com usuários, etc.

1.1.1 - Clusters de Alto Desempenho (HPC - High Performance Cluster)

Clusters de Alto Desempenho (HPC) – Este tipo de cluster tem uma arquitetura destinada a fornecer uma grande capacidade computacional. Clusters HPC são então direcionados a aplicações bastante exigentes em processamento, com o objetivo de permitir que este processamento seja concluído e forneça seus resultados em tempo hábil para análise em uma determinada condição.

Nesta arquitetura, um middleware instalado no head-node do cluster divide as atividades de processamento de cada aplicações em execução entre os nodes de processamento e que por sua vez executam este trabalho de forma cooperada e simultânea / paralela.

Modelos e simulações numéricas utilizados em pesquisas científicas por exemplo, podem se beneficiar deste tipo de cluster por necessitarem processar uma grande variedade de dados e realizar cálculos bastante complexos, diminuindo o tempo de resolução do problema.

Uma observação importante que se aqui se faz necessária é saber que para se obter os benefícios de processamento em cluster HPC, a aplicação tem que ser desenvolvida com suporte a processamento paralelo. São estas técnicas de desenvolvimento (programação paralela e/ou programação distribuida) que permitem a divisão do código da aplicação em blocos de processamento simultâneos.

Aplicações não desenvolvidas para processamento paralelo são executados nos clusters como aplicações sequênciais, como em outro computador qualquer. Só são executadas mais rapidamente porque a arquitetura do cluster em frequência mais rápida.

2) Os clusters HPC PLASMAS

"**HPC PLASMAS**" é a designação nominal ao conjunto de dois clusters HPC adquiridos pela CEA . Deste conjunto, os clusters foram separados em dois equipamentos independentes indentificados com os hostnames: **orion-d11.cea.inpe.br** e **helios.cea.inpe.br.**

Ambos tem as mesmas configurações de hardware, porém serão utilizados com propósitos e aplicações conceitualmente diferentes.

O cluster **orion-d11.cea.inpe.br** (ip de acesso 150.163.55.208) está dedicado ao desenvolvimento de conceitos e aplicações, modelos científicos e testes de simulações. Conta com uma plataforma instalada de softwares e ferramentas computacionais especificas para este fim, tais como compiladores das linguagens fortran, C e C++, java, bibliotecas diversas, ferramentas para testes destas aplicações.

Já o cluster **helios.cea.inpe.br** (ip de acesso 150.163.55.207) está dedicado a processar todas as aplicações e modelos de produção do Programa Clima Espacial da CEA. É o hélios que gera todas os dados e resultados das informações disponíveis nos sites dos Programas EMBRACE, e Clima Espacial, Divisão de Geofísica Espacial, Divisão de Astrofísica e Aeronomia e o INPE como um todo.

2.1) Composição de hardware dos clusters

Cada um dos clusters acima é composto por:

- ✔ 12 (doze) servidores de processamento Supermicro, sendo então um destes servidores, o head-node e os outros 11 servidores, como work-nodes de processamento. Cada servidor (node) é composto por 2 processadores Intel Xeon, modelo E5-2660 v2, penta-core, com 2.2 ghz de frequência. Assim, no conjunto, cada cluster comporta 24 processadores (2 x 12), com 240 núcleos de processamento (24 x 10 cores). Os nodes são nomeados como node001 a node011;
- ✔ 3 (três) Gpu's Nvidia, série Tesla, modelo K20, processadores K110. Cada Gpu contém 2.496 cuda cores; clock de 706 mhz; 5 GB de memória GDRAM; interface de 320-bits. As Gpuś estão instaladas nos nodes 001, 002 e 003. (uma por node);
- ✔ Cada node tem 64 GB de memória RAM, totalizando 768 GB de memória total;
- ✔ Rede ethernet, de 1 gbps dedicada para o sistema operacional e middleware gerenciamento dos recursos e processos em execução;
- ✔ Rede ethernet, de 1 gbps dedicada ao monitoramento de hardware, via protocolo IPMI (Intelligent Platform Management Interface);
- ✔ Rede infiniband, de 40 gbps dedicada a troca de informações, instruções e dados das aplicações em processamento;
- ✔ Storage de armazenamento de dados com capacidade para 7.2 Terabytes;

- ✔ Sistema de No-Break APC de 15 kva's, com autonomia média de 6,37 hs;
- ✔ Sistema Operacional GNU/Cent/OS 6.7 x64, com middleware Bright Cluster Manager 7.0 como software de gerenciamento de cluster.

2.2) Acessando os clusters remotamente

Primeiro, é necessário ter-se uma conta de usuário e uma senha de acesso ao cluster. A criação de uma conta deve ser solicitada ao administrador dos clusters.

Notebooks e outros dispositivos que tenham IP's atribuídos por servidores DHCP das redes wireless do INPE (WIFI-INPE, GUEST_LAN, etc), via de regra e por políticas de segurança, não conseguem acesso remoto aos clusters; salvo mediante autorização especifica solicitada ao admnistrador.

É possivel também solicitar um acesso a partir de um computador configurado com IP externo à rede INPE. Para isso, o usuário deve solicitar autorização ao administrador do cluster, justificando os motivos de acesso, informar o MAC-ADDRESS da interface de rede e as configurações de IPv4 fixo, atribuídos a este computador remoto.

O acesso ao cluster é feito remotamente, via protocolo ssh, a partir de um computador configurado com um número de IPv4 válido da rede interna e cabeada do INPE.

Para acesso, abra um terminal de console em seu computador remoto (cliente) e use o comando ssh, na seguinte forma:

Formato do comando:	*Exemplo:*
ssh -X -l <user-name> <hostname ou ip de acesso ao head-node do cluster>	*ssh -X -l suporte orion-d11.cea.inpe.br* *ssh -X -l suporte 150.163.55.208*
ssh -X <user-name>@<hostname ou ip de acesso ao head-node do cluster>	*ssh -X suporte@helios.cea.inpe.br* *ssh -X suporte@150.163.55.207*

Para completar a conexão, será solicitado a senha deste user-name no cluster. Validada a senha, o prompt de comando estará em um console / terminal, já no head-node. Aqui começa a sua sessão de trabalho no cluster.

2.3) Sessão de Trabalho e os módulos de softwares.

Neste Guia, a abordagem será sobre o cluster **orion-d11.cea.inpe.br**, uma vez que este deverá ser o equipamento para uso da maioria dos usuários. Porém, os comando e tópicos discutidos serão válidos também para o **helios.cea.inpe.br** já que a suas arquiteturas são bastante compatíveis.

2.3.1 - Módulos de Softwares

Módulos de softwares são em essência, mapeamentos que apontam o path para onde as várias aplicações e pacotes disponíveis no sistema foram efetivamente instalados.

O software de gerenciamento de cluster usado tanto no **orion-d11** quanto no **helios**, é o Bright Cluster Manager (http://www.brightcomputing.com.br).

Vários pacotes, softwares, ferramentas e aplicações estão instalados no Sistema Operacional para atender às diversas atividades dos usuários. O Bright Cluster Manager disponibiliza estes módulos de softwares para mapear os paths destas instalações na sessão de login atual do usuário.

São estes módulos de software que irão compor o seu ambiente de trabalho para esta sessão, carregando ou removendo estes módulos. Ao ser carregado, o módulo reconfigura as váriaveis de ambiente da sessão (PATH, LD_LIBRARY_PATH, etc) com os paths de onde estes softwares do módulo foram instalados.

Assim, o Bright disponibiliza módulos de compiladores, módulos de bibliotecas de desenvolvimento, módulos de gerenciamento de jobs, módulos de IDE's de desenvolvimento, módulos de softwares para monitoramento e testes de aplicações, módulos para configuração de ambientes, módulos para customização de hardware, etc.

Por exemplo; se a sua atividade nesta sessão logada for a de compilar o código fonte em um programa em C++, será necessário primeiro carregar o módulo do compilador C++ que voce quer usar (GNU/GCC, INTEL C++, PGI C++, etc) para poder dar o comando de compilação. (pois só assim, o sistema saberá onde o compilador escolhido está instalado) e carregá-lo.

Se o módulo correspondente não for previamente carregado, o sistema retornará erro, pois ele não tem como mapear os paths de instalação dos programas e arquivos que compõem este pacote/software. (embora eles estejam fisicamente instalados no sistema.)

Resumindo, a sua sessão de trabalho só poderá executar uma determinada atividade, se voce carregar os módulos de softwares necessários ao suporte desta atividade.

Note que os módulos são configuram apenas a sessão de login ativa. Ao se desconectar do cluster, estas configurações são perdidas. Ao logar-se novamente, os módulos deverão ser novamente carregados reconfigurar esta nova sessão.

Alguns módulos padrões (que definidos pelo administrador do cluster) são carregados automaticamente pelo Bright após o seu login.

O Bright permite também, criar ou customizar novos módulos de software para os programas especificos, externos ao sistema; como acontece com programas de usuários ou programas de terceiros, que são instalados no sistema e que poderão ficar disponiveis para uso por outros usuários. A criação ou customização de Módulos de softwares devem ser solicitadas ao administrador de sistema.

2.3.2 - Comandos para manipular módulos de software

O Bright Cluster Manager disponibiliza o comando ***module*** e seus parâmetros para

manipular módulos. São eles:

2.3.2.1) module list

list é o parâmetro que lista os módulos ativos em sua sessão. Todos os programas e arquivos cobertos pelos módulos listados por este comando já podem ser utilizados, pois já estão mapeados no sistema.

Formato: *module list*

2.3.2.2) module avail

O parâmetro **avail** lista todos os módulos disponiveis pelo sistema, independentemente de estarem ou não, carregados em sua sessão. Este é o comando usado para consultar o nome dos módulos (e seus respectivos releases) existentes no sistema uso.

Formato: *module avail*

2.3.2.3) module load

load é o parâmetro usado para carregar o módulo especificado no comando na sessão de trabalho ativa.

Formato: *module load <module-name/release>*

2.3.2.4) module add

Semelhante ao parâmetro load, o parâmetro **add** pode ser usado também para carregar o módulo especificado na sessão de trabalho ativa.

Formato: *module add <module-name/release> <module-name/release>*

2.3.2.5) module remove

Parâmetro que **remove** o módulo especificado no comando da sessão de trabalho ativa.

Formato: *module remove <module-name/release>*

2.3.3 - Carregando módulos de software automaticamente

É possivel carregar os módulos de software usados frequentemente, na maioria de suas sessões. Para isso, basta editar o arquivo ***.bash_profile*** (ou ***.bashrc***, dependendo do bash configurado para o seu usuário no sistema) do seu usuário e acrescentar ao fim deste arquivo, linhas do comando module add <module-name/release> para os módulos que voce necessita.

Assim, após o usuário completar o login em uma nova sessão no cluster, o próprio sistema se encaregará de carregar estes módulos na sessão ativa.

Formato: *nano /home/<username>/.bash_profile*
...
...

module add <module-name/release> <module-name/release>
module add <module-name/release>

3) O gerenciador Bright Cluster Manager (BCM)

Bright Cluster Manager (BCM) é o middleware utilizado em ambos os clusters do CEA, para gerenciamento de todos os recursos computacionais disponibilizados pelos sistemas.

O BCM é responsável pelas contas de usuários; pelo monitoramento dos processos em execução; pelo controle dos nodes ativos (ou não); pelos outros dispositivos de hardware que compõem o cluster; pelas restrições de uso e acesso aos recursos; pelas regras de monitoramento do sistema; pelas imagens de sistema carregadas; pelo monitoramento e quota de uso do espaço e volume de armazenamento de dados; enfim, pelo controle geral dos recursos dos clusters.

O BCM tem alguns recursos bastante completos e que estão descritos no User Guide do programa. Este manual pode ser solicitado aos administradores do cluster, ou ainda, baixado diretamente no site http://www.brightcomputing.com, empresa desenvolvedora do software.

3.1) A interface gráfica *cmgui*

O BCM disponibiliza uma interface gráfica que pode ser acessada pelo usuário e que pode ser carregada pelo módulo de software **cmgui/<release>**, comando **cmgui**. Para isso, use a sequência de comandos abaixo, em um terminal de console, já logado a um dos clusters.

- *module avail* – (verificar o nome e a referência de release atual do módulo cmgui/<release>
- *module list* - (verificar se o módulo cmgui/<release> já está carregado em sua sessão.
- *module load* cmgui/<release> - (para carregar o módulo, caso necessário)
- *cmgui* – (comando para carregar a interface gráfica do BCM)

Nesta interface gráfica é possível monitorar os recursos dos clusters e também permitir algumas intervenções/operações sobre objetos ou atividades que estão sobre a instância do usuário, tais como cancelar algum job em execução; suspender a execução temporáriamente; gerar algum log sobre a execução do processo; criar regras de monitoramento; enfim, várias atividades que estão sob o dominio user.

3.2) O shell cmsh (cluster manager shell)

O BCM disponibiliza ainda uma série de comandos exclusivos deste software, que permitem:

- ✔ Gerir recursos, controles e operações não disponíveis na interface gráfica;
- ✔ Alterar parâmetros de operação e funcionamento do BCM;
- ✔ Criar ou customizar regras e filtros mais especificos de monitoramento;
- ✔ Criar novos objetos de otimização de uso e recursos do cluster;

Para isso, o BCM disponibiliza ainda o módulo de software **cmsh/<release>**, comando **cmsh**.

O **cmsh** é o shell (interpretador de comandos) exclusivo do BCM e ao ser carregado, abre um terminal para que sejam entrados os comandos para execução. Estes comandos e potencial de recursos disponibilizados no cmsh (shell) estão descritos detalhadamente no User Guide do BCM, que pode ser solicitado aos administradores do cluster para consulta.

Para isso, use a sequência de comandos abaixo, em um terminal de console, já logado a um dos clusters.

- *module avail* – (verificar o nome e a referência de release atual do módulo cmsh/<release>
- *module list* - (verificar se o módulo cmsh/<release> já está carregado em sua sessão.
- *module load* cmsh/<release> - (para carregar o módulo, caso necessário)
- *cmsh* – (comando para carregar a interface gráfica do BCM)

4) O Torque Resource Manager e o Maui Scheduler.

Toda aplicação submetida a execução deve definir uma lista de recursos necessários para a sua execução (quantidade de processadores, quantidade de processos paralelos, montante de memória, quantidade de nós, espaço de armazenamento, entre outros), a sua prioridade, o tempo de execução desejado e as suas dependências. Essas informações guiam o escalonador na escolha de quais aplicações devem executar em quais recursos.

Para gerir todos os recursos computacionais dos clusters e disponibilizá-los ao processamento dos jobs (sendo job, qualquer aplicação ou processo em execução), nos clusters do CEA (até a edição deste Guia, em Setembro/2016), estão sendo utilizados os softwares **Torque Resource Manager** e o **Maui Scheduler**.

O Torque é o software gerenciador destes recursos e que cria o mecanismo responsável por estruturar o processamento de um job. Deve-se criar um arquivo contendo diretivas *#PBS* (ou variaveis de ambiente com nomes iniciados por *PBS_*) que especificam as alocações dos recursos computacionais necessários no processamento da aplicação em paralelo. Este arquivo deve ser gravado em formato texto (ASCII) e deve ser atribuído a extensão .pbs (que identifica scripts Torque).

O arquivo .pbs deve ainda conter a linha de execução do programa/aplicação a ser processado.

O exemplo abaixo ilustra o conteúdo de destes arquivos.

cat /home/support/runModelOgino.pbs

linha	*comando*
1	*#!/bin/bash*
2	*clear*
3	*cd $HOME/Ogino3D*
4	*pwd*
5	*PBS_O_WORKDIR=$HOME/ogino3D*
6	*#PBS -N ogino3D*
7	*#PBS -m abe*
8	*#PBS -M support@dge.inpe.br*
9	*#PBS -l nodes=2:ppn=16*
10	*#PBS -l mem 32gb*

linha	*comando*
11	*#PBS -q longq*
12	*/cm/shared/apps/openmpi/gcc/64/1.8.1/bin/mpiexec $HOME/ogino3D/runs/mh3 < $HOME/ogino3D/data/inputParams.txt*

Table 4.1: Exemplo de arquivo .pbs (Torque)

Observe acima que as diretivas do Torque são inicializadas por #PBS ou PBS_. Os parâmetros que as seguem fazem as requisições dos recursos computacionais exigidas para execução da aplicação. No caso citado no exemplo acima, estão sendo reservados 2 nodes (na linha 9) e 32 Gb de RAM por node (linha 10).

Observe também que a última linha do arquivo (linha 12) é a linha de execução do programa junto com os parâmetros necessários para executá-lo.

Nota importante: Os recursos computacionais requisitados nos arquivos .pbs, o Torque assume como **requisitos mínimos para execução da aplicação** para que possa ser executada com estabilidade. Assim, no exemplo acima, este job só será executado quando o cluster estiver com 2 nodes computacionais livres, e ainda, cada node com 32 GB livres.

Enquanto o sistema/cluster não disponibilizar estes todos os recursos solicitados, este job permanece na fila de execução. Outros jobs que requisitarem menor poder computacional passarão a ser executados àfrente deste.

Assim, os recursos solicitados em um arquivo .pbs funcionam como um operador **.and.** ou **acumulativo**.

Para submeter uma aplicação para a fila de execução, use o comando ***qsub*** seguido do nome do arquivo .pbs.

Exemplo: *qsub -np 32 ./runModelOgino.pbs*

O comando qsub é um dos programas inclusos no pacote do Torque. Assim, o Torque irá interpretar o arquivo .pbs e, caso todos os seus parâmetros estejam corretos, submete a aplicação à queue de execução.

Os arquivos .pbs podem também funcionar como um shell script comum de sistema. Isso porque, quando um shell Linux (bash, dash, csh, etc) encontra uma diretiva #PBS o shell a ignora, passando a execução para a próxima linha do arquivo. Para isso, basta dar permissão de execução ao arquivo .pbs e rodá-lo normalmente (./<nome do arquivo.pbs>).

Resumindo, se o arquivo .pbs for rodado com o comando **qsub**, ele será direcionado ao Torque. Se o arquivo .pbs for executado diretamente, no terminal, ele será interpretado como um shell-script Linux comum, apenas com a extensão não convencional (.sh).

Já o software Maui Scheduler, trata-se de um gerenciador de processos de código aberto para clusters e surgiu com o objetivo de suprir algumas carências no desempenho nas políticas de escalonamento de alguns softwares gerenciadores de recursos tais como o Torque. Assim,

Maui atua como um componente/ferrramenta complementar ao Torque, estendendo as capacidades de gerenciamento de recursos, tais como configuração de acesso aos recursos; configuração de qualidade de serviço; estabelecer condições para disponibizar recursos; implementação de politicas de BackFill, entre outras.

4.1) Principais diretivas de um script de submissão

4.1.1 - Variáveis de Ambiente Torque

Criar um script de submissão de job, será necessário criar um arquivo e gravá-lo em padrão ASCII, tendo por default, a extensão **.pbs**

Este script de submissão deve conter as diretivas e/ou as váriaveis de ambiente Torque que irão estruturar a execução do job em paralelo. As principais diretivas do Torque estão descritas abaixo.

Opcionalmente, um script de submissão pode funcionar como um shell-script de sistema. Para isso, a primeira linha do arquivo deve fazer chamada a um shell de sistema (bash, csh, dash, etc). Deve também ser atribuído a permissão de execução (atributo x) ao arquivo deste script.

Variáveis de ambiente Torque, para configuração do job.

Diretiva	*Descrição*
PBS_O_QUEUE	Variável de ambiente do Torque que define o nome da queue de execução a qual o job será submetido. Variável equivalente à diretiva #PBS -q <queue-name> em um script. *Formato:* *PBS_O_QUEUE=<queue-name>* *Exemplo:* *PBS_O_QUEUE=longq*
PBS_O_WORKDIR	Variável de ambiente do Torque que define o diretório de trabalho da aplicação, e que será usado (como default) para gerar os arquivos de saída. *Formato:* *PBS_O_WORKDIR=<directory-path>* *Exemplo:* *PBS_O_WORKDIR=$HOME/tests/Cyne*
PBS_O_PATH	Variável de ambiente Torque que especifica o path do job para execução. *Formato:* *PBS_O_PATH=<directory-path>* *Exemplo:* *PBS_O_PATH=$HOME/tests*
PBS_NODEFILE	Especifica o arquivo que contém a lista de nomes dos nós a serem

Diretiva	*Descrição*
	atribuídos à tarefa. O Torque tenta alocar os nós especificados para executar o job. *Formato: PBS_NODEFILE=<path/filename>* *Exemplo: PBS_NODEFILE=$HOME/runs/runNodes.txt*
PBS_O_JOBNAME	Especifica um nome definido pelo usuário para ser atribuído ao job. Variável equivalente à diretiva #PBS -N <"job-name"> em um script .pbs. *Formato: PBS_JOBNAME=<"job-name">* *Exemplo: PBS_JOBNAME="Ogino2048"*
PBS_JOBID	Esta variável é criada pelo Torque e retorna o id de identificação de um job que foi submetido para a queue de execução. *Formato: PBS_JOBID* *Exemplo: echo $PBS_JOBID*

Table 4.2: Principais Variáveis de ambiente Torque.

4.1.2 - Diretivas de script Torque

As diretivas abaixo podem ser incluídas em um script de submissão, para alocar recursos necessários para a execução do job no cluster. As linhas das diretivas são iniciadas pelo prefixo **#PBS**, sempre em caixa alta.

Diretiva	*Descrição*
#PBS -N <job-name>	Especifica um nome definido pelo usuário para ser atribuído ao job. Variável equivalente à diretiva #PBS -N <"job-name"> em um script .pbs. Diretiva equivalente à variável **PBS_O_JOBNAME.** *Formato: #PBS -N <"job-name">* *Exemplo: #PBS -N "Ogino2048"*
#PBS -q <queue-name>	Diretiva Torque que define o nome da queue de execução a qual o job será submetido. Diretiva equivalente à variável PBS_O_QUEUE em um script. *Formato: #PBS -q <queue-name>* *Exemplo: #PBS -q longq*
#PBS -l mem=<size-memory>	Esta diretiva indica a quantidade de memória RAM será

Diretiva	*Descrição*
	disponibilizada em cada node para a execução do job. É indicada pelo valor, seguido da unidade “kb”, “mb”, “gb”, “tb”. *Formato: #PBS -l mem=<size-memory>* *Exemplo: #PBS -l mem=32gb* *#PBS -l mem=64mb* Cada node dos clusters HPC PLASMAS tem 64 gigabytes de ram, totalizando 768 gigabytes por cluster.
#PBS -l nodes=<number-nodes> **#PBS -l nodes=<hostnames>**	Esta é uma das principais diretivas a ser definida no script de submissão. Ela define o número de nodes que serão utilizados no processamento do seu job. (Ver item 2.1 deste manual). Neste caso, o gerenciador de jobs Maui utilizará quaisquer nodes que estiverem disponíveis no sistema para processamento do job. *Formato: #PBS -l nodes=<number-nodes>* *Exemplo: #PBS -l nodes=4* Um outro formato possível para esta diretiva é especificar quais nodes serão utilizados no processamento do job, via o hostname destes nodes. Neste caso, o gerenciador de jobs Maui executará o job em todos os nodes especificados, assim que o conjunto estiver disponível (free). *Formato: #PBS -l nodes=<hostmame1, hostname2, …>* *Exemplo: #PBS -l nodes=node001,node002,node005*
#PBS -l np=<number-cores>	A diretiva ***np*** (number of processors) indica quantos núcleos de processamento de um microprocessador multicore serão usados no processamento do job. No caso dos clusters HPC PLASMAS, cada node de processamento é composto por 2 (dois) processadores multicore, Intel Xeon, série E5-2660 v2, com 10 núcleos/cores de processamento cada processador. Esta diretiva indica então, quantos destes 20 núcleos de processamento serão utilizados no processamento do job.

Diretiva	*Descrição*
	Formato: #PBS -l np=<number-cores> *Exemplo: #PBS -l np=6* Caso este parâmetro não seja indicado, o gerenciador de jobs tenta usar o maior número de núcleos/cores disponíveis nos nodes.
#PBS -l ppn=<number-processes>	A diretiva ***ppn*** (processes per node) indica em quantos processos paralelos por node o job será executado. *Formato: #PBS -l ppn=<number-of-processes>* *Exemplo: #PBS -l ppn=6*
#PBS -l gpus=<quantity gpus>	Esta diretiva indica a quantidade de gpus que seram utilizadas para o processamento do job. *Formato: #PBS -l gpus=<quantity gpus>* *Exemplo: #PBS -l gpus=2* Há uma gpu Nvidia Kepler K20M instalada nos node001, node002 e node 003 (totalizando 3 gpu's) em cada um dos clusters HPC PLASMAS. (ver item 2.1 deste manual)
Nota: É possível também montar uma única linha no script .pbs, combinando as diretivas do parâmetro -l (*nodes, np, ppn, mem, gpus, etc*). Para isso, separe as diretivas de uma mesma configuração pelo caracter : (dois pontos). Diretivas gerais são separadas por virgula. *Formato: #PBS -l nodes=<number-nodes>:np=<number-cores>:ppn=<number-processes>:mem=<size-memory>* *Exemplo: #PBS -l nodes=4:np=7:ppn=10:mem=32gb, gpus=2* *ou* *Formato: #PBS -l nodes=<hostnames>:np=<number-cores>:ppn=<number-processes>* *Exemplo: #PBS -l nodes=node001:np=7:ppn=10:gpu=1:mem=20gb, node002:np=7:ppn=10,node003:ppn=10:gpus=1* *Variações permitidas:* *#PBS -l nodes=3:node001:np=6:ppn=10:mem=24gb,node002:np=6:ppn=10:mem=12gb, node005:ppn=12* *#PBS -l nodes=11:ppn=12, gpus=2, mem=20gb* *#PBS -l nodes=11, mem=48gb*	

Table 4.3: Diretivas de script Torque

4.2) Comandos Torque para manipulação e controle dos jobs

Segue alguns dos principais comandos do pacote Torque, que permitem manipular os jobs de execução, além de fornecer informações sobre os recursos computacionais disponiveis no cluster.

4.2.1 - qsub

Qsub submete um job para processamento no cluster, seguindo os critérios especificados pelas diretivas do script .pbs indicado no comando.

Formato: *qsub <parameters> -np=<quantity processes> <script-name.pbs>*

Exemplo: qsub -q defaultq -np 32 runLaxLiu2048.pbs

Na linha acima, o comando qsub tentará gerar um job, tentando inseri-lo nas filas de execução do Torque/sistema. Caso o job seja aceito (sem nenhum erro nas ditetivas do script), o comando qsub retornará o ***id*** de identificação deste job (***jobId***) . Com este ***jobId*** será possivel monitorar o processo/job no sistema, pelos comandos do Torque (qstat, qlist , qremove, etc).

4.2.2 - qstat

O comando qstat é usado para solicitar o **estado de execução** dos jobs e filas em processamento no cluster. O comando retorna uma lista identificando os estados de execução; o owner dos jobs; em quais nodes estes jobs estão sendo executados; tempo de processamento destes jobs, entre outros parâmetros.

Formato: *qstat <parameters> <jobId>*

Exemplo: *qstat -f 2093.orion-d11*

Principais parâmetros qstat

Parameter	*Description*
-a	*Lista o estado e as informações básicas de **todos** os jobs do usuário ativo, aguardando execução nas queues de processamento.*
-f	*Lista todas as informações sobre o job indicado no comando.*
-n	*Além das informações básicas, são listados também os nodes em que este job está sendo executado.*

Os flags de estados possiveis no Torque são:

Flag	*Description*
C	*Job is completed after having run.*
E	*Job is exiting after having run.*
H	*Job is held.*
Q	*Job is queued, eligible to run or routed.*
R	*Job is running.*
S	*(Unicos only) Job is suspended*
T	*Job is being moved to new location.*
W	*Job is waiting for its execution time (-a option) to be reached.*

Table 4.4: Flags dos estados possiveis para execução dos jobs.

4.2.3 - pbsnodes

O comando pbsnodes lista o status dos nodes computacionais do cluster, de acordo com o parâmetro fornecido.

Principais parâmetros:

Parameter	*Description*
pbsnode -l all	Retorna uma lista com o nome e o status de todos os nodes computacionais que compõem o cluster. • ***busy*** - node está cheio e pode não aceitar novos jobs adicionais. • ***down*** - node desligado ou está detectando/reportando falhas locais. • ***free*** - node está pronto para aceitar novos jobs para processamento.

- ***job-exclusive*** **-** nodes cuja todos os processadores virtuais disponíveis neste node estão atribuído ao processamento de job exclusivo.

- ***job-sharing*** **-** nodes setados para processar múltiplos jobs compartilhadamente.

- ***offline*** **-** node setado para não receber/aceitar novos jobs para processamento.

- ***reserve*** **-** node está reservado para executar outras tarefas exclusivas do cluster

- ***time-shared*** **-** node permite executar vários jobs simultâneamente.

- ***unknown*** **-** node não pode ser detectado por alguma falha.

pbsnodes -l up	Retorna uma lista apenas dos nodes atualmente ligados, e que estão operando com status em um dos modos: free, job-exclusive, job-sharing, reserve e time-sharing.
pbsnodes -l down	Retorna uma lista apenas dos nodes atualmente desligados.
pbsnodes -l free	Retorna uma lista apenas dos nodes ativos, disponiveis para aceite de jobs para execução.
pbsnodes -o <nodename>	Altera o status do node indicado para OFFLINE.

Table 4.5: Comando pbsnodes e seus principais parâmetros

Formato: pbsnodes <parameters> <argument>

Exemplo: pbsnodes -l all

4.2.4 - qdel

O comando qdel exclui o job indicado das queues de execução.

Formato: *qdel <jobId>*

Exemplo: qdel 2093.orion-d11

4.2.5 - qalter

O comando qalter modifica os atributos/condições definidas originalmente, de um job aguardando execução nas queues de processamento. É possivel alterar os valores de todas as diretivas #PBS do script utilizado para submissão do job.

Assim, é possivel por exemplo, aumentarmos o número de nodes para processamento do job; modificar o tempo de processamento; quantidade de memória alocada; etc sem excluir o job da queue de execução.

Os atributos (e seus novos valores) são indicados como parâmetros do comando qalter. Contudo, se qualquer um dos atributos especificados no comando, não puder ser modificado (por qualquer razão), qalter cancela a operação e nenhum dos atributos listados será modificado no job.

O comando qalter tem uma diversidade de parâmetros. Indica-se então usra o comando qalter --help ou man qalter para consulta aos parâmteros e a forma de como indicar os novos valores para estes parâmetros.

Formato: *qalter <parameters> <jobId>*

Exemplo: qalter -h n -l nodes=4:ppn=10 2093.orion-d11

4.2.6 - qhold

Qhold é um atributo que suspende a execução do job indicado no comando, na queue de processamento, até que que este job seja novamente liberado para execução (comandos qrerun ou qalter).

Se o job já estiver em estado de execução, o atributo hold faz com que a execução seja suspensa imediatamente. Se o job estiver na queue, aguardando processamento, qhold altera o estado do job para suspenso. O job ainda ficará na fila, mas agurando liberação para execução.

Há 3 instâncias de hold: user, others e system e o sistema repeita a hierarquia de privilégios entre estas instâncias para liberação de re-execução do job. Estas instâncias são indicadas pelos respectivos caracteres: (***u***) para user; (***o***) others e (***s***) para system,

Formato: *qhold -h <hold list> <jobId>*

Exemplo: *qhold -h u 2093.orion-d11* *qhold -h us 2093.orion-d11*

4.2.7 - qrls

O comando qrls remove o atributo hold (estado de suspensão) de um job na queue de processamento, fazendo com que o job volte a aguardar processamento na queue.

O sistema permite atribuir a um job, três tipos de atributo hold (modo de suspensão): Um hold pode ser atribuído ao um job pelo:

- (***u***) referenciando-se ao usário proprietário do job (user);
- (***s***) referenciando-se ao administrador do sistema/clusters (system),
- (***o***) referenciando-se a outros usuários (others).

A combinação de uma ou mais destas letras de referências, formam o argumento hold-list solicitado pelo comando. Se o processo de remoção for bem sucedido, o comando qrls retorna um status de retorno com valor 0 (zero).

Formato: *qrls -h <hold list> <jobId>*

Exemplo: *qrls -h uo 2093.orion-d11* *qhold -h us 2093.orion-d11*

4.2.8 - qrun

O comando qrun é utilizado para forçar o software escalonador (Torque) a inicializar a execução de um job já submetido a queue de processamento, independentemente da posição que ele estiver na queue. É necessário privilégios de Operação ou Administração.

Se o parâmetro <node-name> não for especificado, o próprio sistema seleciona o(s) node(s) de menor carga para processar o job.

Formato: *qrun -H <node-name> <jobId>*

Exemplo: qrun -H node002 2093.orion-d11

4.2.9 - qrerun

O comando qrerun direciona os jobs especificados na linha de comando para que sejam re-executados/reprocessados, com os mesmos parâmetros do último processamento.

Um job pode estar marcado com atributo de não reexecução. Neste caso, o comando qrerun é retorna erro até que o atributo seja removido (comando qalter).

O parâmetro -f é usado para forçar a re-execução do job.

Formato: *qrerun <-f> <jobId>*

Exemplo: qrerun 2093.orion-d11 qrerun -f 2093.orion-d11

4.2.10 - qgpumode

O comando qgpumode especifica o modo de como a GPU indicada no comando irá operar. Uma GPU pode operar em 4 modos diferentes. São eles.

Parameters	*Description*
-m	Altera o modo de operação da GPU. São indicados pelos dígitos: • ***0*** (***Shared***) Modo de computação compartilhada. Múltiplas threads de múltiplos jobs estão habilitadas a usar o recurso cudaSetDevice(). Este é o modo default do sistema. • ***1*** (***Exclusive Thread***) Modo de computação exclusiva (por Thread) . Somente uma thread (por vez), de um único processo está habilitada a usar o recurso cudaSetDevice(). • ***2*** (***Prohibited***) Modo de computação "Proibida", onde nenhuma thread de enhum processo em execução terá acesso ao recurso cudaSetDevice(). • ***3*** (***Exclusive Process***) Mode de computação exclusiva (por processo). Todas as threads de um único processo terá acesso ao recurso cudaSetDevice(). CudaSetDevice() - Função de alocação & endereçamento das Threads. Retorna o gpuID da thread alocada em memória.
- H	Especifica o hostname/node-name onde a GPU está instalada.
-g	Espefifica a referência (gpuID) da GPU para operação. Esta referência (gpuID) pode variar em função da versão do kernel Linux instalado no sistema.

Table 4.6: qgpumode: Modos de computação de threads na GPU.

Formato: *qgpumode <-H node-name> <-g gpuID> < -m mode-value >*

Exemplo: qgpumode -H node002 -g 1 -m 3

Nos clusters PLASMAS (orion-d11.cea.inpe.br e helios.inpe.br) existem 6 (seis) GPU's Nvidia, linha Tesla, modelo Kepler-K20. Três destes dispositivos estão instaladas nos

node001, node002 e node003 de cada cluster, sendo então 3 GPU's por cluster, 1 por node (acima descritos).

4.2.11 - qgpureset

O comando qgpureset reseta os parâmetros atribuidos a uma GPU, voltando estes parâmetros para o modo default do sistema.

Parameters	*Description*
- ***H***	Especifica o hostname/node-name onde a GPU está instalada.
-***g***	Especifica a referência (gpuID) da GPU para operação. Esta referência (gpuID) pode variar em função da versão do kernel Linux instalado no sistema.
-***p***	Reseta a GPU com contagem de erro EEC permanente.
-***v***	Reseta a GPU com contagem de erro ECC volátil.

Table 4.7: qgpureset: Parâmetros do comando.

Formato: *qgpureset <-H node-name> <-g gpuID> <-p> <-v>*

Exemplo: qgpureset -H node002 -g 1 -p

4.2.12 - qmgr

O comando qmgr fornece uma interface de administração shell para consultar e configurar os parâmetros da daemon do Torque Server. Assim. É possível criar, excluir ou mover filas de processamento; ativar ou desativar recursos disponiveis; configurar (set/unset) atributos destes recursos; dentre outros.

Qmgr instância os objetos *servers*, *nodes* e *queues* criados no sistema.

Ao ser executado, qmgr abre um terminal shell, onde ele lê as diretivas da linha de comando. Em algumas operações, será necessário privilégios de Operador ou Administração do sistema. Para encerar a interface qmgr, use o comando quit. .

Formato: *qmgr <enter> → Entra no prompt de comandos do shell qmgr*

Line command*:* *<command> server <server-name > <attr OP value>, <attr OP value>...*

Line command: *<command> queue <queue-name> <attr OP value>, <attr OP value>...*

Line command: *<command> node <node-name> <attr OP value>, <attr OP value>...*

Command	*Description*
active	Define os objetos ativos indicado no comando. *Ex: active server s_orion*
create	Cria um novo objeto com o nome indicado no comando. Válido para queues e nodes. *Ex: create queue little <parameters>*
delete	Exclui um objeto existente. Comando aplicável apenas para queues e nodes. *Ex: delete queue little*
set	Define ou altera o valor de um atributo incorporado ao objeto indicado no comando. *Ex: set queue little resources_max.mem=8mw,resources_ max.cput =10* *Ex: set node node002 state="down,offline"*
unset	Exclui o valor de um atributo incorporado ao objeto indicado no comando. *Ex: unset queue little max_running*
list	Lista todos os atributos e os valores a eles atribuídos do objeto indicado no comando. *Ex: list node node002* *Ex: list queue little*
print	Imprime todos os atributos do objeto indicado, em um formato que pode ser usado como re-entrada do comando qmgr. *Ex: print server* *Ex: print queue longq*

Table 4.8: qmgr: Descrição dos comandos.

A tabela abaixo, descreve os parâmetros utilizados para alterar os atributos de objetos.

Command	*Description*
names	Lista um ou mais nomes dos objetos indicado na linha de comando, no formato: *[name][@server][,queue_name[@server]…]*
attr	Especifica o nome do atributo de um objeto para ser consultado sou modificado.
OP	Operações a serem realizadas para alterar os atuais valores dos atributos de um objeto. • "=" Define o novo valor ao atributo do objeto. O valor atual do atributo será substituido por este novo valor. • "+=" Substitui o valor atual do atributo, pela somatória do valor atual mais o valor especificado no comando. Incrementa (valor atual + novo valor.) • "-=" Substitui o valor atual do atributo, pela diferença do valor atual menos o valor especificado no comando. Decrementa (valor atual - novo valor.)
value	Define o valor a ser definido a um atributo. Se o valor incluir espaços em branco, vírgulas ou outros caracteres especiais, tais como o carácter "#", a cadeia de valor deve ser colocado entre aspas (").

Table 4.9: qmgr: Descrição dos parâmetros do comando.

4.2.13 - pbsdsh

O comando pbsdsh provê recursos para executar um ou mais jobs seriais, de forma paralela.

Jobs SERIAIS são os softwares cujo os códigos foram desenvolvidos sem nenhum tipo de recurso de desenvolvimento paralelo (implementações MPI 's ou OpenMP). São portanto, códigos sem comunicação interna entre processos durante a sua execução.

O comando pbsdsh então, gerência as várias execuções simultâneas destes jobs independentes distribuindo a carga de trabalho (workload) entre os núcleos/cores de processamento dos nodes alocados no script de submissão Torque.

Pbsdsh é tipicamente usado para:

- N execuções de um mesmo programa/job com um argumento diferente a cada rodada,
- iniciar N execuções de programas/jobs diferentes em cada processo ou
- N execuções de um mesmo programa/job, lendo diferentes arquivos de entrada.

O comando pbsdsh deve ser a última linha do script de submissão, tendo como parâmetro, o código do programa e os argumentos para execução deste job.

Exemplo de um script Torque, submetendo um job via comando pbsdsh.

```
#!/bin/bash
#PBS -q @nic-cluster.srv.mst.edu
#PBS -V
#PBS -l nodes=3
#PBS -d /nethome/users/USERNAME/pbsdsh_example/
#PBS -l walltime=0:05:00
/share/apps/job_data_prerun.py
pbsdsh -v $PBS_O_WORKDIR/per_node_runner.sh
```

Table 4.10: pbsdsh: Exemplo de script de submissão.

Neste mesmo conceito então, Jobs PARALELOS, são os códigos que incorporam algum tipo de tecnologia de paralelização (MPI, OpenMP e hibridas), de forma que os processos consigam trocar mensagens entre si. Aqui os blocos paralelos são especificados no próprio código. Estes tipos de códigos não são escopos/domínio do comando pbsdsh.

5) Roteiro básico para submissão de jobs no cluster

5.1) Logar-se no cluster

Use o comando ssh no formato abaixo para logar-se em um dos clusters PLASMAS. Maiores informações no item 2.2 deste manual.

ssh -X -l <login-name> <ip ou hostname do cluster>

Ex: ssh -X -l suporte orion-d11.cea.inpe.br ou ssh -X -l suporte helios.cea.inpe.br

5.2) Configurar a sessão de trabalho

Configure a sua sessão de trabalho ativa, carregando ou removendo os módulos de softwares necessários para o desenvolvimento e execução das suas atividades programadas para esta sessão. Veja o item 2.3 que descreve os comandos necessários:

module list - Lista os módulos de softwares já ativos em sua sessão.

module avail – Lista os módulos de softwares disponíveis no sistema e que podem ser carregados para sua sessão.

module add <module-name> - Adiciona o módulo de software indicado no comando para que ele seja carregado automaticamente na sua sessão de trabalho, após seu login. Use esta opção para módulos que voce necessite constantemente.

module load <module-name> - Carrega o módulo de software indicado no comando apenas para esta sessão ativa.

module remove <module-name> - Remove um módulo de software da sua sessão ativa.

5.3) Compilar seus códigos fontes

Lembre-se que o sistema operacional do cluster disponibiliza via recurso de módulos, a possibilidade de carregar em sua sessão de trabalho diferentes compiladores e releases de algumas linguagens de programação; diferentes bibliotecas de desenvolvimento, diferentes IDE's e diversas ferramentas para depuração do código.

Carregue o módulo do compilador e bibliotecas para criar a compilação de seus códigos fontes, verificando algum possivel erro de compilação ou erro em tempo de execução.

Uma vez que o código esteja compilado e estabilizado, crie um script de submissão para executá-lo em paralelo nos nodes dos clusters PLASMAS..

5.4) Montar o script de submissão para execução do código

Com base nas diretivas descritas no item 4.1 deste guia, crie um arquivo/script de submissão conforme descrito no capítulo 4. Este arquivo deve ser gravado em formato ASCII e com a extensão padrão .pbs. (veja exemplo no capitulo 4).

A última linha deste arquivo deve ser a linha de execução do seu programa, indicando o comando de execução paralelo (mpirun, mpiexec, openmpi, etc); o seu código executável e caso necessário, os parâmetros de entrada para rodá-lo.

5.5) Submeter o script com comando qsub

O próximo passo é enviar este arquivo/script de execução via comando qsub descrito no item 4.2.1 deste manual.

qsub <parameters> -np <total parallel processes> <script-name.pbs>

O total de processos paralelos do parâmetro -np é calculado pelo produto da quantidade de nodes pelo numero de processo por node (nodes * ppn) descrito no corpo do script.

Após o job ser submetido à fila de execução, o comando qsub retorna o job-id do processo. Este job-id será usado como referência para monitorar o processamento do job no sistema.

5.6) Monitorar a execução dos jobs submetidos

O item 4.2 deste guia descreve os comandos básicos do Torque para monitorar a execução e gerenciamento dos jobs submetidos.

Dentre eles, o comando qstat retorna status dos jobs, indicando o tempo de execução, etc.

qstat -f <job-id>

Referências:

http://www.infowester.com/cluster.php

http://pt.slideshare.net/luiz_arthur/tpicos-cluster-de-balanceamento-de-carga

https://labs.moip.com.br/blog/cluster-de-alta-disponibilidade-x-cluster-de-alto-desempenho-computacional-o-futuro/

http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/ipmi-cim-emerging-technologies.pdf

http://clusterresources.com

http://www.adaptivecomputing.com/products/open-source/torque/

https://www.adaptivecomputing.com/products/open-source/maui