CONTEÚDO

INTRODUÇÃO

O Tupã é um sistema de supercomputação adquirido pelo INPE – Instituto Nacional de Pesquisas Espaciais, com arranjo institucional para seu financiamento proveniente do FNDCT, projeto FINEP/SUPERCLIMA, encomendado através do MCT, e pelo Programa FAPESP de Pesquisa em Mudanças Climáticas Globais – PFPMCG.

É composto originalmente por um supercomputador Cray XT6, sistema de acesso interativo, sistema de processamento auxiliar e sistema de armazenamento, e está instalado nas dependências do Centro de Previsão de Tempo e Estudos Climáticos do INPE - CPTEC/INPE, em Cachoeira Paulista, no interior do estado de São Paulo.

A partir de maio de 2012, o sistema foi atualizado para o modelo XE6, com sistema de interconexão Gemini, operando em sua capacidade total com 14 gabinetes, 1.304 nós computacionais e 31.296 processadores.

CARACTERÍSTICAS

O sistema de supercomputação Cray XE6 do INPE é composto pelos seguintes itens:

- **13 nós de acesso interativo:**
  - Os nós de acesso interativo ou nós de login são utilizados para acesso ao sistema e para submissão de jobs e oferecem recursos para pré e pós processamento.
  - Possuem 1,664 TB de memória total agregada.
  - Podem ser acessados através do nome *tupa.cptec.inpe.br*:

```
hostname% ssh -l [username] tupa.cptec.inpe.br
Password:
```

  - Cada nó de acesso interativo possui:
    - 128 GB de memória compartilhada por nó ou 8 GB por core.
    - 4 processadores quad-core AMD Opteron.
    - Redes Ethernet de 1 Gigabit/s e 10 Gigabits/s.

- **20 nós de processamento auxiliar:**
  - Os nós de processamento auxiliar são utilizados para atender o processamento de jobs, cujas características em termos de paralelismo melhor se ajustam a este ambiente, e executar tarefas auxiliares que precedem ou sucedem a execução das simulações numéricas como, por exemplo, a coleta e o manipulação de dados atmosféricos para alimentar a simulação e a geração de figuras para disseminação dos resultados.
  - Possuem 2.56 TB de memória total agregada.
  - Cada nó de processamento auxiliar possui:
    - 128 GB de memória compartilhada por nó ou 8 GB por core.
    - 4 Processadores quad-core AMD Opteron.
    - Redes Ethernet de 1 Gigabit/s e 10 Gigabits/s.

- **768 nós computacionais Cray XE6 com:**
    - 32 GB de memória compartilhada por nó (1.333 GB por core).
    - 2 processadores 12-core AMD Opteron de 2.1 GHz por nó.
    - 1 módulo de interconexão Cray Gemini por nó.

- **32 nós de serviço com:**
    - 8GB de memória cada um.
    - 2 processadores dual-core AMD Opteron de 2.6 GHz por nó.
    - 1 módulo de interconexão Cray Gemini por lâmina.

- **Sistema de arquivos Lustre acessível por todos os nós computacionais, nós de acesso interativo e nós de processamento auxiliar.**

SISTEMA OPERACIONAL E SHELLS

O sistema operacional do supercomputador Tupã é baseado no SuSE Linux Enterprise Server e oferece os seguintes shells:

- sh
- ksh
- bash
- csh
- tcsh

O shell padrão é o bash. caso deseje alterar o bash padrão, por favor entre em contato com helpdesk@inpe.br

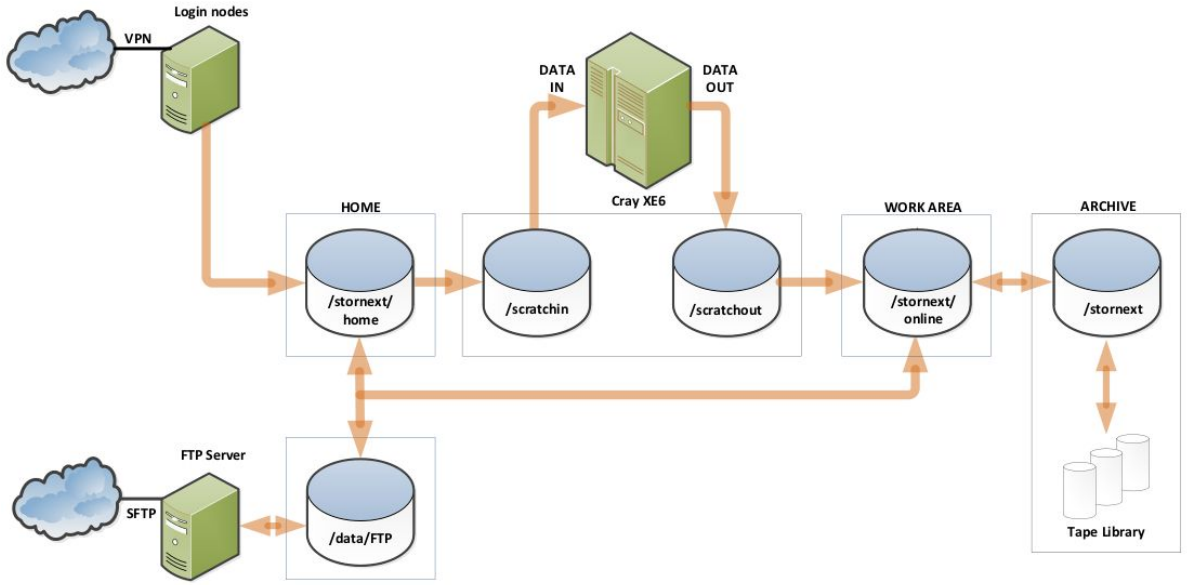
INFORMAÇÕES E STATUS DO SISTEMA

Notícias relacionadas ao sistema e o status atual estão disponíveis durante o processo de login ou na página http://sssn.cptec.inpe.br/monitoramento/supercomputador/, disponível na rede interna do CPTEC ou com acesso VPN.

ARMAZENAMENTO

Há duas áreas de armazenamento no supercomputador Tupã que possuem **a mesma estrutura de diretórios** e possuem basicamente três diferentes propósitos:

Storage Systems and Data Flow



Áreas de armazenamento

- **SAN:** Armazenamento primário de alta capacidade.
- **NAS Lustre:** Sistema de arquivos paralelo de alta performace.

Estrutura de diretórios

- **/stornext:** Área dedicada ao armazenamento de dados dos usuários (áreas /home, áreas de trabalho primárias (/online), áreas operacionais e de produção, grupos e projetos, com quotas que dependem da utilização). É utilizado também para o armazenamento de resultados transferidos do /scratchout para tratamento, pós-processamento ou armazenamento histórico (HSM) de longo prazo.
- **/scratchin:** Área destinada ao armazenamento de programas executáveis, scripts e conjuntos de dados de entrada que são utilizados na submissão de jobs, com quotas e sem implementação de backups.
- **/scratchout:** Área utilizada para armazenamento temporário dos resultados produzidos pela execução dos programas. É uma área de saída, sem quota definida, e o tempo de permanência dos dados nesta área é de 8 dias. Após este período os dados são removidos sem consulta prévia aos usuários. Esta área não possui backup.

Existem diferentes variáveis de ambiente apontando para as diferentes áreas de armazenamento disponíveis no supercomputador Tupã, conforme exemplos abaixo:

Nome	Descrição	Quota	Política de Retenção	Política de Backup	Política de HSM

$HOME	*$HOME* são diretórios utilizados para armazenar executáveis, bibliotecas ou scripts, pequenos conjuntos de dados e arquivos comuns relacionados às contas de usuários.	5, 15 ou 30 GB	Não removido	SIM	NÃO
$SUBMIT_HOME	*$SUBMIT_HOME* é um diretório no sistema de arquivos compartilhado utilizado para armazenar os dados e programas necessários para a submissão de um jobs para processamento.	5TB	Não removido	NÃO	NÃO
$WORK_HOME	*$WORK_HOME* aponta para uma área do sistema de arquivos compartilhado paralelo de alta performance, acessível por todos os nós. Esta área deve ser utilizada para armazenar os resultados das execuções.	Não há	Remoção a cada 15 dias	NÃO	NÃO

NOTAS:

1. Solicitações para aumento de quotas devem ser encaminhadas ao helpdesk.

Além dessas variáveis relacionadas à **HOME**, existem outras relacionadas a áreas operacionais, de produção, grupos e projetos, conforme exemplos abaixo:

Nome	Descrição	Quota	Política de	Política de	Política

			Retenção	Backup	de HSM
$SUBMIT_*	*$SUBMIT_** é um diretório no sistema de arquivos compartilhado utilizado para armazenar os dados e programas necessários durante a submissão de um job para processamento.	500 GB	Não removido	NÃO	NÃO
$WORK_*	*$WORK_** aponta para uma área do sistema de arquivos compartilhado paralelo de alta performance, acessível por todos os nós. Esta área deve ser utilizada para armazenar os resultados das execuções.	Atualmente não possui quota	Remoção a cada 15 dias	NÃO	NÃO
$ONLINE_*	Armazenamento primário de dados dos grupos.	De acordo com a necessidade de cada grupo	Não removido	NÃO	NÃO

NOTAS:

1. (*) Refere-se ao nome da área operacional, de produção, de grupos ou projetos, por exemplo, a área de saída do modelo operacional **ETA**, *$WORK_ETA*, apontando para */scratchin/oper/tempo/ETA*.

Quota

Para verificar sua quota digite o comando *uquota*:

```
tupa% uquota
```

TRANSFERÊNCIA DE DADOS

Para usuários nas redes locais dentro do prédio do CPTEC de Cachoeira Paulista

Dentro do prédio do CPTEC em Cachoeira Paulista - SP o usuário deve seguir as instruções abaixo:

Copiando dados de outro sistema computacional para o Tupã:

```
outro_sistema% scp file.bin [username]@tupa.cptec.inpe.br:/<área>
```

Copiando dados do supercomputador Tupã para outro sistema computacional:

```
outro_sistema% scp [username]@tupa.cptec.inpe.br:/<área>/file.bin
<área-do-outro-sistema>
```

NOTAS:

1. *<área>* refere-se a uma das áreas descritas na seção Armazenamento. Por exemplo, a área de armazenamento primário de dados do grupo do modelo operacional **ETA** é *$ONLINE_ETA*.

O *acesso* ao supercomputador Tupã através de outras redes do INPE (e.g. COPDT) deve ser feito através de *ssh e VPN*.

MODELOS DE PROGRAMAÇÃO PARALELA

Vários tipos de paralelismo podem ser explorados no supercomputador Tupã utilizando diferentes modelos e métodos de programação, conforme listados na tabela abaixo:

Nível de Hardware	Modelo	Descrição
Shared-memory node	Auto	Habilita paralelização automática em programas de memória compartilhada. Loops básicos podem ser automaticamente paralelizados, mas para melhoria adicional de desempenho, tente utilizar diretivas **OpenMP**. Esta opção está disponível nos ambientes **PGI** e **Pathscale**.
Shared-memory node	OpenMP	Esta é uma forma de programação paralela explícita na qual o programador insere diretivas no programa para disparar múltiplos threads de memória compartilhada, tipicamente a nível de loop. É um método comum de paralelização, portável e relativamente fácil de implementar. Por outro lado, ele requer memória compartilhada cujos limites escalam ao número de processadores em apenas um nó. **OpenMP** pode ser utilizado em conjunto com a autoparalelização e está disponível com os ambientes **PGI**, **Pathscale** e **GNU**.
Distributed memory	MPI	Este é o método mais comum e portável para paralelização de códigos para sistemas de memória distribuída escalável. **MPI** é uma biblioteca de subrotinas para troca de mensagens, operações coletivas e outras formas de comunicação inter-processadores. O programador é responsável por implementar a distribuição de dados, sincronização e remontagem dos resultados utilizando chamadas explícitas **MPI**. Utilizando **MPI**, o programador pode amplamente ignorar a organização física dos processadores nos nós e simplesmente tratar o sistema como uma coleção de processadores

		independentes. **MPI** está disponível com os ambientes **PGI**, **Pathscale** e **GNU**.
Distributed memory	SHMEM	Esta biblioteca suporta comunicações *one-sided* entre as tarefas.

AMBIENTE DE PROGRAMAÇÃO

Estão disponíveis no supercomputador Tupã as seguintes ferramentas de programação. Certifique-se de trocar para o *module* apropriado quando alternar entre os compiladores.

Item	PGI	PathScale	GNU Compilers
Fortran 77/90/95 (Compute Node)	*ftn*	*ftn*	*ftn*
C compiler (Compute Node)	*cc*	*cc*	*cc*
C++ compiler (Compute Node)	*CC*	*CC*	*CC*
Fortran 77/90/95 (Login Node Only)	*pgf90*	*pathf90*	*gfortran*
C compiler (Login Node Only)	*pgcc*	*pathcc*	*gcc*
C++ compiler (Login Node Only)	*pgCC*	*pathCC*	*g++*
Debuggers	*totalview*	*totalview*	*totalview*
Performance Analysis	*CrayPat*	*CrayPat*	*CrayPat*
Default module (*)	*PrgEnv-pgi* (default)	*PrgEnv-pathscale*	*PrgEnv-gnu*
Batch queueing system	*PBS Professional 10.2*		

Nota: O ambiente de programação **PGI** é carregado por padrão para todas as contas.

Módulos

O supercomputador Tupã possui o pacote *module* instalado. Esta ferramenta permite a um usuário de maneira rápida e fácil alternar entre diferentes versões de um pacote ou compiladores, por exemplo. O pacote *module* define variáveis comuns de ambiente utilizadas por aplicações, tais

como *PATH*, *MANPATH*, etc. O módulo *PrgEnv-pgi* é carregado por padrão para todas as contas. Ele carrega a versão atual dos compiladores **PGI** com suporte a **MPI** no seu *PATH*.

Comando	Exemplo	Propósito
module avail	`tupa% module avail`	Lista todos os módulos disponíveis no sistema.
module load *pkg*	`tupa% module load PrgEnv-pathscale`	Carrega um módulo do ambiente.
module unload *pkg*	`tupa% module unload PrgEnv-pgi`	Remove um módulo do ambiente.
module list	`tupa% module list`	Mostra os módulos atualmente carregados.
module switch *old new*	`tupa% module switch PrgEnv-pgi PrgEnv-gnu`	Troca o módulo *old* pelo módulo *new* no ambiente.
module purge	`tupa% module purge`	Restaura o ambiente ao estado original, removendo todos os módulos carregados após o estado inicial.

Veja **man module** para mais informações.

Tipicamente, os módulos precisam ser carregados apenas durante o login. Isto pode ser feito colocando os comandos de *module* no seu *.cshrc* (*csh*/*tcsh*), *.bashrc* (*bash*) ou *.profile* (*ksh*).

PGI - Compilando e linkando programas em Fortran

O compilador Fortran 90 é o *ftn*.

Este compilador possui suporte a **OpenMP**, **MPI** e **Shmem**.

Veja um comando típico mostrando várias opções:

```
tupa% ftn -O3 prog.f90
```

Portland Group Fortran 90 Compiler Options

Option	Description
-c	Generate intermediate object file does not attempt to link.
-g	Adds information for debugging to the object file and/or executable.
-I<*directory*>	Tells the preprocessor to search in *directory* for include or module files.
-L<*directory*>	Tells the linker to search in *directory* for libraries.
-r8	Promotes REALs from the default size of 4 bytes to 8 bytes.
-i8	Promotes INTEGERs from the default size of 4 bytes to 8 bytes.
-default64	Passes the -r8 and -i8 options to the compiler.
-O3	Higher level of optimization than -O2 (the default optimization level).
-fast	Higher optimization level than -O3.
-tp barcelona-64	Use optimizations for the AMD Barcelona Quad Core processor (default).
-Mipa	Tells the compiler to perform interprocedural analysis. Can be very time consuming to perform. This flag should also be used in both compilation and linking steps.
-Mconcur	Enables autoparallelization. Additional options can be used with -Mconcur to provide more fine-grained control of autoparallelization, see *man pgf90* for details.
-mp=nonuma	Enables parallelization via OpenMP directives.
-Minfo	Displays useful information to stderr, see *man pgf90* for details.
-Mneginfo	Displays information on why a particular optimization was not performed.

NOTA: Muitas outras opções de compilação estão disponíveis. Veja as páginas de manual para informações adicionais:

man pgf90

man ftn

PGI - Compilação e linkedição de programas em C/C++

Os compiladores C são chamados utilizando o seguinte comando *cc*.

Compilar em C++ requer o comando *CC*.

Estes compiladores tem suporte a **OpenMP**, **MPI**, **Shmem** e **Pthreads**.

Aqui estão alguns exemplos de comandos de compilação mostrando as opções mais comuns:

```
tupa% cc -O3 -o prog prog.c

tupa% CC -O3 -o prog prog.cpp
```

Option	Description
-c	Generate intermediate object file but does not attempt to link.
-g	Adds information for debugging to the object file and/or executable.
-I<*directory*>	Tells the preprocessor to search in *directory* for include or module files.
-L<*directory*>	Tells the linker to search in *directory* for libraries.
-O3	Higher level of optimization than -O2 (the default optimization level).
-fast	Higher level optimization (default is -O2). This flag should be used in both compilation and linking steps.
-Mipa	Tells the compiler to perform interprocedural analysis. This option can be very time consuming to perform. This flag should be used in both compilation and linking steps.
-Mconcur	Enables autoparallelization. Additional options can be used with -Mconcur to provide more fine-grained control of autoparallelization, see *man pgcc* or *man pgCC* for details.
-mp=nonuma	Enables parallelization via OpenMP directives.
-Minfo	Displays useful information to stderr, see *man pgf90* for details.
-Mneginfo	Displays information on why a particular optimization was not performed.

Nota: Muitas outras opções de compilação estão disponíveis. Veja as páginas de manual para informações adicionais:

man pgcc

man pgCC

man cc

man CC

PathScale - Compilação e linkedição de programas em Fortran

O compilador Fortran 90 pode ser chamado através do comando *ftn*.

Este compilador tem suporte a **OpenMP**, **MPI**, **Shmem** e **Pthreads**.

```
tupa% ftn -O3 prog.f90
```

PathScale Fortran 90 Compiler Options

Option	Description
-show-defaults	List default compiler options for the compiler and exits.
-c	Generates intermediate object file but does not attempt to link.
-g	Adds information for debugging to the object and/or executable.
-I<*directory*>	Tells the preprocessor to search in *directory* for include or module files.
-L<*directory*>	Tells the linker to search in *directory* for libraries.
-r8	Promotes REALs from the default size of 4 bytes to 8 bytes.
-i8	Promotes INTEGERs from the default size of 4 bytes to 8 bytes.
-default64	Passes the -i8 and -r8 options to the compiler.
-O3	Higher level of optimization than -O2 (the default optimization level).
-cpp	Preprocess files with the C preprocessor. Enabled by default for files ending in .F, .F90 or .F95.
-ftpp	Preprocess files with the Fortran preprocessor. Useful when portions of the Fortran code could be misinterpreted as C preprocessor directives (e.g. "//").
-O3	Higher level of optimization than -O2 (the default optimization level).
-O3-OPT;Ofast	Higher optimization level than -O3.

-ipa	Tells the compiler to perform interprocedural analysis. Can be very time consuming to perform. This flag should also be used in both compilation and linking steps. Not recommended for programs over 100,000 lines for the current compiler release.
-intrinsic=PGI	Enables intrinsic functions that are available in the PGI compiler which are not ANSI standard (e.g. rand).
-apo	Enables autoparallelization.
-mp	Enables parallelization via OpenMP directives.

Nota: Muitas outras opções de compilação estão disponíveis. Veja as páginas de manual para informações adicionais:

man pathf90

man eko

man ftn

PathScale - Compilação e linkedição de programas em C/C++

Os compiladores C são chamados utilizando o seguinte comando *cc*.

Para compilar em C++ use o comando *CC*.

Este compilador suporta **OpenMP**, **MPI**, **Shmem** e **Pthreads**.

Aqui está um exemplo de comando mostrando algumas opções mais comuns:

```
tupa% cc -O3 -o prog prog.c

tupa% CC -O3 -o prog prog.cpp
```

Option	Description
-show-defaults	List default compiler options for the compiler and exits.
-c	Generate intermediate object file but does not attempt to link.
-g	Adds information for debugging to the object file and/or executable.
-I<*directory*>	Tells the preprocessor to search in *directory* for include or module files.
-L<*directory*>	Tells the linker to search in *directory* for libraries.

-O3	Higher level of optimization than -O3 (the default optimization level).
-Ofast	Higher level optimization (default is -O2). This flag should be used in both compilation and linking steps.
-ipa	Tells the compiler to perform interprocedural analysis. This option can be very time consuming to perform. This flag should be used in both compilation and linking steps. Not recommended for programs over 100,000 lines for the current compiler release.
-apo	Enables autoparallelization.
-mp	Enables parallelization via OpenMP directives.

Várias outras opções estão disponíveis. Veja as páginas de manual para informações adicionais.

man pathcc

man pathCC

man eko

man cc

man CC

PathScale - Outras ferramentas

As seguintes ferramentas são disponibilizadas com a suíte do compilador **PathScale**.

Command	Notes
assign	Alters the way Fortran I/O is performed.
explain	Provides extended information for compile error and diagnostic messages.
pathhow-compiled	Show the compiler options used to produce an object file. e.g.: `tupa% pathhow-compiled sample.o`

PathScale - Variáveis de tempo de execução

A tabela abaixo mostra algumas variáveis de ambiente que afetam a função de variáveis de tempo de execução (*runtime*) de executáveis compilados com compiladores **PathScale**.

Environment Variable	Languages	Description
PSC_OMP_AFFINITY	OpenMP codes	*PSC_OMP_AFFINITY* specifies whether or not **OpenMP** tasks should be bound to particular processors. Valid values are *TRUE* or *FALSE*.
PSC_FDEBUG_ALLOC	Fortran codes	*PSC_FDEBUG_ALLOC* specifies the initial value to be placed in Fotran memory allocations for debugging purposes. Valid values are *ZERO*, *NaN* or *NAN8*.
FILENV	Fortran codes	*FILENV* specifies the file name which contains assign options to be used by a Fortran program.

Bibliotecas

No supercomputador Tupã existem bibliotecas disponíveis geralmente para a suíte de compiladores **PathScale**, **Portland Group** e **GNU**. A maioria das versões atuais de bibliotecas e arquivos de include estão disponíveis nos seguintes diretórios:

PathScale Compilers	Libraries:	*/usr/local/pathscale/lib*
	Include Files:	*/usr/local/pathscale/include*
PGI Compilers	Libraries:	*/usr/local/pgi/lib*
	Include Files:	*/usr/local/pgi/include*
GNU Compilers	Libraries:	*/usr/local/gnu/lib* */usr/local/lib*
	Include Files:	*/usr/local/gnu/include* */usr/local/include*

ANÁLISE DE DESEMPENHO

A ferramenta de análise de desempenho mais indicada no supercomputador Tupã é a **CrayPat**. Veja aqui os passos básicos requeridos para construir e instrumentar executáveis:

1. Carregue o módulo *xt-craypat*:

```
tupa% module load xt-craypat
```

2. Recompile o código como você faria normalmente para gerar um executável:

```
tupa% ftn mycode.f90 -o mycode
```

3. Use o comando *pat_build* para gerar um executável instrumentado:

```
tupa% pat_build -g mpi -u mycode
```

Isto gera um executável instrumentado chamado *mycode+pat*. A opção *-g* habilita o tracegroup **MPI**. Veja *man pat_build* para os tracegroups disponíveis.

4. Rode o executável instrumentado com o aprun via **PBS**:

```
tupa% aprun -n 80 ./mycode+pat
```

Isto gera um arquivo de saída instrumentado. (ex: *mycode+pat+2007-12tdt.xf*)

5. Use *pat_report* para mostrar as estatísticas do arquivo de saída:

```
tupa% pat_report mycode+pat+2007+12tdt.xf > mycode.pat_report
```

Veja *man pat_build* para opções adicionais de instrumentação.

EXECUTANDO JOBS INTERATIVOS

É possível utilziar o sistema de filas para executar um job interativo utilizando o comando:

```
tupa% qsub -q debug -l mppwidth=16 -A <Código da Instituição> -I
```

Uma vez que o seu job é iniciado, você pode executar comandos interativamente no nós computacionais que o **PBS** designou para a sua sessão. Por exemplo:

```
tupa% aprun -n 16 ./myprog
```

EXECUTANDO JOBS EM BATCH

Todo o trabalho operacional, de produção e pesquisa no supercomputador Tupã é executado através do **PBS**. O sistema de filas PBS oferece mais memória e maiores limites de tempo que os nós interativos. Jobs executados através do sistema de filas também salvam convenientemente os *stdout*/*stderr* em arquivos, para cada rodada, e continuam mesmo após o *log off*.

Um job em batch é um shell script com um prefácio que declara os requerimentos de recursos necessários e contém instruções que serão utilizadas pelo **PBS** para gerenciar o job.

Scripts **PBS** são submetidos para processamento através do comando *qsub*.

Conforme mostrado abaixo, existem cinco passos comuns na maioria dos processamentos básicos em batch:

1. Crie um script batch:

Normalmente, todos os comandos **PBS** são embutidos dentro do script batch.

No script batch, conforme a sintaxe **PBS**, todas as opções **PBS** devem preceder os comandos shell. Cada linha contendo uma opção **PBS** deve iniciar com o caracter string, "**#PBS**" seguido por um ou mais espaços e seguido pela opção.

Os scripts **PBS** iniciam a execução do seu diretório home e então, o primeiro comando shell script executável é usualmente um *cd* para o diretório *$SUBMIT_* ou *$WORK_*. A variável de ambiente *$PBS_O_WORKDIR* é setada para o diretório de onde o script foi submetido.

Lembre-se que ao submeter um job para os nós computacionais é necessário fazê-lo a partir de um diretório que esteja dentro de */scratchin*.

PBS Script - MPI

```
#!/bin/bash
#A opção –q define a fila na qual o job será executado
#PBS -q <fila>
#Define 48 cores, e tempo de execução de 4 horas
#PBS -l mppwidth=48
#PBS -l walltime=4:00:00
#PBS -A <Código da Instituição>
#A opção -j junta os arquivos de erro e de saída
#PBS -j oe
#Muda o diretório para o diretório de trabalho inicial
cd $PBS_O_WORKDIR
#Executa o programa MPI
aprun -n 32 ./myprog
```

PBS Script - OpenMP

```
#!/bin/bash
#A opção –q define a fila na qual o job será executado
#PBS -q <fila>
#Define 8 cores, e tempo de execução de 8 horas
#PBS -l mppwidth=8
#PBS -l walltime=8:00:00
#PBS -A <Código da Instituição>
#A opção -j junta os arquivos de erro e de saída
#PBS -j oe
#Muda o diretório para o diretório de trabalho inicial
cd $PBS_O_WORKDIR
#Define o n° de threads OpenMP que será usado
export OMP_NUM_THREADS=8
#Executa o programa OpenMP
aprun –n 1 –d 8 ./myprog
```

2. Submeta um batch script para o PBS, usando qsub:

O script pode possuir qualquer nome. Se no exemplo acima foi dado o nome *myprog.pbs*, ele seria submetido para processamento com o comando:

```
tupa% qsub –A <Código da Instituição> myprog.pbs
```

3. Monitore o job:

Para verificar o status do job **PBS** submetido, execute o comando:

```
tupa% qstat –a
```

Alertamos que a utilização constante do comando *qstat* gera sobrecarga no sistema PBS, causando o comprometimento do ambiente. Sugerimos algumas formas alternativas para a verificação do status dos jobs submetidos no PBS:

3.1. No script de submissão, inclua 2 novas declarações, uma logo após o cabeçalho (#PBS) e outra na última linha:

```
#PBS
.
.
.
#Atualiza o arquivo de status do job com RUNNING
echo "RUNNING" > ${PBS_O_WORKDIR}/${PBS_JOBID}
.
.
.
aprun ...
.
.
.
#Atualiza o arquivo de status do job com FINISHED
echo "FINISHED" > ${PBS_O_WORKDIR}/${PBS_JOBID}
```

3.2. Para submissão do script, execute:

```
job=`qsub <script>`
echo "QUEUED" > ${job}
```

A primeira linha do código acima executa o comando qsub e atribui o Job ID (nnnnnnn.sdb) à variável ambiente *$job*. A segunda linha cria um arquivo de status cujo nome é o Job ID, com o status de QUEUED.

3.3. Para executar uma ação após o término do script:

```
while [[ `grep -v FINISHED ${job} ]]
do
cat $job
sleep 60
done
rm ${job}
<ação seguinte>
```

O código acima executa o ciclo de espera até o encerramento do job.

3.4. Caso o monitoramento seja ao término de um conjunto de jobs:

a) certifique-se de que os scripts contém as linhas inseridas no item 3.1 acima;

b) certifique-se de que os arquivos de status serão criados para cada um dos jobs submetidos.

No diretório de submissão, execute:

```
jobs=*.sdb
#Executa o ciclo de espera até o encerramento de todos os jobs
while [[ `grep –v FINISHED ${jobs}` ]]
do
#Caso a quantidade jobs seja muito grande
echo "`grep QUEUED ${jobs} | wc -l` jobs em fila"
echo "`grep RUNNING ${jobs} | wc -l` jobs em execução"
echo "`grep FINISHED ${jobs} | wc -l` jobs encerrados"
sleep 300
done
#Uma vez encerrados todos os jobs, remova os arquivos de status
rm  ${jobs}
<ação seguinte>
```

3.5. Os comandos acima estão na sintaxe do bash. Para scripts em *csh*, mude apenas a sintaxe do comando *while* de:

```
while [[ `grep –v FINISHED ${jobs}` ]]
do
.
.
.
done
```

para

```
while (`grep –v FINISHED ${jobs} | wc -l`)
.
.
.
end
```

Para scripts em bourne shell (*/bin/sh*), mude a sintaxe do comando *while* para:

```
while test `grep -v FINISHED ${jobs}
do
.
.
.
done
```

3.6. Para configurar o envio de e-mails de notificação de jobs pelo PBS, adicione as seguintes opções no cabeçalho do script:

```
#PBS -m bae
#PBS -M <seuemail>@<domínio>
```

A primeira linha especifica os eventos que disparam o envio de e-mail: o job foi iniciado (b), abortado (a) ou encerrado (e). A segunda linha especifica o endereço de e-mail do destinário das mensagens.

***Atenção**: Não use essa opção quando submeter vários jobs ao mesmo tempo.*

4. Delete o job

Dado o número de identificação **PBS** (retornado quando se deu o comando *qsub* e mostrado em *qstat -a*), você pode deletar o job do sistema de filas com o comando:

```
tupa% qdel <PBS-ID>
```

5. Examine o Resultado

Quando o job termina, o **PBS** salva o *stdout* e *stderr* num arquivo no diretório de onde ele foi submetido. Estes arquivos serão nomeados usando o nome do script e o número de identificação **PBS**. Por exemplo:

```
tupa% myprog.pbs.o<PBS_ID>
```

FILAS PBS

Veja as filas disponíveis através do comando:

```
tupa% qstat -Q
```

Os detalhes sobre uma fila específica podem ser vistos através do comando:

```
tupa% qstat -Qf <fila>
```

Atualmente, as seguintes filas estão disponíveis:

Fila	Propósito
oper	Fila para submissão de jobs Operacionais. Dependendo do parâmetro *walltime*, deriva o job para execução numa das filas **oper.<*>**.
prod	Fila para submissão de jobs de Produção. Dependendo do parâmetro *walltime*, deriva o job para execução numa das filas **prod.<*>**.
pesq	Fila para submissão de jobs relacionados à Pesquisa & Desenvolvimento. Dependendo do parâmetro *walltime,* deriva o job para execução numa das filas **pesq.<*>**.
proj	Fila para submissão de jobs relacionados à Projetos de Pesquisa. Dependendo do parâmetro *walltime*, deriva o job para execução numa das filas **proj.<*>**.
expressa	Fila para submissão de jobs com necessidades emergenciais e alta prioridade. Sob controle do Suporte da Supercomputação.
especial	Fila de baixa prioridade para execução de jobs de longa duração. Sob controle do Suporte da Supercomputação.

Lembre-se:

- Há limites de tempo de execução para cada fila.
- A acuracidade dos tempos especificados pelos usuários será avaliada e, assim que surgirem padrões de uso, novas filas ou ajustes e novas restrições poderão ser criadas para facilitar o uso eficiente do sistema.
- Quanto mais precisa forem as estimativas mais razoável o scheduler será para os usuários do sistema.

Ambiente Auxiliar

Para execução de jobs que necessitem de até 16 processadores (1 node), recomendamos utilizar a fila *pesq.q* do ambiente auxiliar:

```
tupa% module load aux
tupa% qstat -Qf pesq.q
```

CONTABILIZAÇÃO DO USO DOS RECURSOS

Inicialmente os recursos de supercomputação estão alocados de acordo com a seguinte distribuição mensal, considerando uma reserva técnica para uso interno do Serviço de Supercomputação e 1200 nós computacionais de 24 cores por nó, ou 26.400 horas MPP(1) por hora de relógio.

Usuário	Código da Instituição	Porcentagem	Número de segundos por Hora MPP
Previsão Numérica de Tempo e Clima Sazonal - Operação, Produção, Desenvolvimento e Pesquisa (CPTEC/INPE)	CPTEC	40%	41.472.000
Cenários de Mudanças Climáticas - Produção, Desenvolvimento e Pesquisa (CCST/INPE), Rede Clima e INCT	CCST	30%	31.114.000
Programa FAPESP Pesquisa Mudanças Climáticas	FAPESP	30%	31.114.000
TOTAL DE SEGUNDOS POR HORA MPP		100%	103.680.000

(1) Massive Parallel Processing

O uso dos recursos de supercomputação é contabilizado por usuário, de acordo com a atividade ou projeto indicado por ele em tempo de submissão de cada job através do parâmetro "-A", e debitado do percentual de uso definido para cada instituição, conforme tabela acima.

As atividades se caracterizam por: Operação, Produção, Desenvolvimento ou Pesquisa, e cada usuário está associado a uma ou mais atividades, de acordo com seu grupo de trabalho ou projeto, e deverá utilizar o parâmetro "-A" correspondente, por exemplo:

Para usuários do DIMNT, DIPTC e DISSM:

```
tupa% qsub -l walltime=4:00:00,mppwidth=48 -A CPTEC myprog.sh
```

Para usuários da DIIAV:

```
tupa% qsub -l walltime=4:00:00,mppwidth=48 -A CCST myprog.sh
```

Para participantes de um projeto de pesquisa:

```
tupa% qsub -l walltime=4:00:00,mppwidth=48 -A <Código da Instituição>
myprog.sh
```

Ao submeterem jobs para processamento via comando qsub, os usuários devem obrigatoriamente indicar pelo menos três parâmetros, que indicam o uso dos recursos e identificam a atividade ou projeto:

-l walltime=<hh:mm:ss> mppwidth=<número de processadores> -A <Código da Instituição>

```
tupa% qsub -l walltime=4:00:00,mppwidth=48 -A <Código da Instituição>
myprog.sh
```

Ao término de cada job, é debitado da Instituição correspondente a quantidade de recursos utilizada.

COMO EDITAR O .BASHRC PARA USO DO GRADS

Para habilitar o uso do grads em sua conta o usuário deve editar o arquivo .bashrc localizado em /stornext/home/<usuario> de sua conta, incluindo as seguintes informações:

module load tupa2

umask 023

export LANG=US

export PATH=$PATH:${HOME}:${HOME}/bin:./:/opt/grads/2.0.a9/bin

#VARIAVEIS DO GRADS

export GRADSB=/opt/grads/2.0.a9/bin

export GADDIR=/opt/grads/2.0.a9/dat

export GADLIB=/opt/grads/2.0.a9/lib

export GASCRP=/opt/grads/2.0.a9/lib

export GAUDFT=/opt/grads/2.0.a9/udf/udft

COMO EDITAR O .BASHRC PARA O USO DO NCL

Para habilitar o uso do ncl em sua conta o usuário deve editar o arquivo .bashrc localizado em /stornext/home/<usuario> da sua conta, incluindo as seguintes informações:

Configuracoes do NCL

NCARG_ROOT=/opt/ncl/5.2.0

PATH=$PATH:/opt/ncl/5.2.0/bin

export NCARG_ROOT PATH

DOCUMENTAÇÃO E MANUAIS ONLINE

- Documentos Cray Online: http://docs.cray.com
- Compiladores PGI: http://www.pgroup.com/resources/docs.htm
- Compiladores PathScale: http://pathscale.com/docs.html
- PBS - Guia do Usuário