COMPILADORES E MPI NA UNA

Daniel Merli Lamosa – PAD/CPTEC
julho – 2007

Compiladores disponíveis:

	C/C++	Fortran
Intel	icc	ifort
GNU	gcc	gfortran
PathScale	pathcc	pathf90/pathf95
Portland	pgcc	pgf90/pgf95

Uso do MPI:

Compilação:

A compilação de programas em FORTRAN que utilizam a biblioteca MPI pode ser feita com os diferentes compiladores listados na tabela acima. A forma padrão de uso é:

mpif90 -ccl (compilador) -o nome_arquivo_gerado

Exemplos:

mpif90 -ccl ifort -o arquivo_intel.x
mpif90 -ccl gfortran -o arquivo_gnu.x
mpif90 -ccl path90 -o arquivo_path.x
mpif90 -ccl pgf90 -o arquivo_pgi.x

O mesmo procedimento vale para compilações em C/C++:

mpicc -ccl (compilador) -o nome_arquivo_gerado (C)

mpic++ -ccl (compilador) -o nome_arquivo_gerado (C++)

Execução:

Execuções no *Cluster* Una são realizadas através de um sistema de filas configurado para distribuir os *jobs* entre os nós conforme os recursos disponíveis. Para entender o sistema de filas é necessário entender a configuração do equipamento.

A Una possui um total de 275 nós de processamento sendo cada um deles composto por 2 processadores Dual-Core Opteron 2218 (2.6GHz) com 1MB de cache e 8GB de memória. Nessa configuração, é possível disparar até 4 processos por nó (1 por core) num total de 1100 processos.

O sistema de filas foi montado de acordo com o número de processos disparados por nó. Foram criadas 6 filas diferentes:

- **mpi-npn1** = Distribuição dos recursos na forma 1 processo por nó
- **mpi-npn2** = Distribuição dos recursos na forma 2 processos por nó
- **mpi-npn3** = Distribuição dos recursos na forma 3 processos por nó
- **mpi-npn4** = Distribuição dos recursos na forma 4 processos por nó

- **mpi-fill** = Distribuir a quantidade de CPU's solicitadas em esquema de Fill Up. A quantidade de CPU's alocadas em cada nó será feita buscando a melhor distribuição mas isso dependendo da disponibilidade de CPU's em cada nó, ou seja, havendo nós com 4 CPU's disponíveis estes serão alocados, quando acabarem os nós com 4 CPU's disponíveis serão alocados os nós com 3 CPU's disponíveis e assim por diante até que a quantidade de CPU's solicitada tenha sido alocada

- **mpi-rb1** = Distribuir a quantidade de CPU's solicitadas em esquema de Round Robin. Uma CPU por nó será distribuída sendo que se houver requisição de mais cpu's que nós disponíveis uma nova rodada de alocação será feita iniciando pelo primeiro nó da rodada anterior, desde que o mesmo ainda tenha CPU's disponíveis, e assim por diante até que todas as CPU's requisitadas pelo usuário tenham sido alocadas.

Exemplo01: Para disparar um *job* com **200** processos

	Processos por nó	**Número de nós usados**
mpi-npn1	1	200
mpi-npn2	2	100
mpi-npn3	3	??? (200 não é divisível por 3!)
mpi-npn4	4	50
mpi-fill	4	50*
mpi-rb1	1	200*

* caso ninguém esteja usando a fila

Exemplo02: Para disparar um *job* com **300** processos

	Processos por nó	**Número de nós usados**
mpi-npn1	1	??? (existem apenas 275 nós)
mpi-npn2	2	150
mpi-npn3	3	100
mpi-npn4	4	75
mpi-fill	4	75*
mpi-rb1	1-2	250 (1 processo) + 25 (2 processos)*

- caso ninguém esteja usando a fila

É possível notar que usar a fila *mpi-npn1* com mais de 275 processadores não é viável, pois não temos mais que 275 nós. Erros como este podem ser identificados com o comando ***qstat -j***

Na fila mpi-fill sempre busca-se usar o máximo de CPU's por nó e na fila mpi-rb1 busca-se usar o maior número de nós antes de disparar um segundo processo em um nó.

A submissão de jobs é feita através do comando ***qsub*** de um *script* criado com os seguintes campos:

Cabeçalhos:
#$ -pe mpi-X N => Escolha da fila
X = Qual fila
N = número de processadores

#$ -N NomedoJob => Define nome para o *job (*mostrado com *qstat)*

#$ -o una1:/DIR/ArqSaida => Qual arquivo de saída
DIR = diretório de saída
ArqSaida = nome do arquivo

#$ -j y => Saída de erro direcionada ao arquivo de saída

Comando:
mpirun -np ${NSLOTS} -machinefile ${TMPDIR}/machines *caminho/exe*

Note que:

O número de processadores é obrigatoriamente múltiplo do número de processos utilizados por nó (ou seja, na primeira linha, N é múltiplo de X).

${NSLOTS} => número total de processos (o mesmo que N no cabeçalho); essa variável é atribuída automaticamente pelo sistema de filas;

${TMPDIR}/machines => Arquivo gerado pelo gerenciador de filas.

caminho/exe => Indica o caminho completo do arquivo executável

Existem outras variáveis que podem ser úteis, todas atribuídas automaticamente pelo sistema de filas:

${NHOSTS} **=>** Quantidade total de nós que formam o *machinefile*
${NPN} **=>** Quantidade de processos por nó.

Script pré-montado: *filaUna.nqs*

```
#$ -pe mpi-npn4 32
#$ -N Eta32p
#$ -o una1:/home/usuario/worketa/eta/diretoriodeTrabalho/scripts/saidaEta32p.txt
#$ -j y
cd /home/usuario/worketa/eta/diretoriodeTrabalho/scripts
./start.ksh 2006120412
```

Este arquivo está localizado no diretório scripts dentro do diretório da rodada, ou seja, quando se compila o Eta, este gera um diretório de rodada.
Ex: /home/usuario/worketa/eta/199X249X38_32proc/scripts

Para submeter usar:
qsub FilaUna.nqs

Acompanhamento da execução:

qstat => Lista os processos de todos os usuários.
qstat -j => Lista processos com problemas de execução
qstat -j n => Lista informações do processo de identificação *n* (ID)

Remoção de tarefas submetidas:

qdel n => Remove processo de identificação *n* (número do processo)

Sugestões específicas para o Eta WS:

O Eta WS está configurado para ser executado no cluster Una em apenas 3 passos:

1. Execute o *script* ***gera_set_parmeta.sh N1 N2*** no diretório *worketa/eta/install*. Note que *N1* e *N2* são, respectivamente, os valores de ***INPES*** e ***JNPES.*** Será criado um arquivo chamado ***set_parmeta_199X249X38_Nproc*** (*N = INPES*JNPES*).

   OBS: Não apagar o arquivo ***set_parmeta_orig***, pois ele é usado para gerar os demais.

2. Executar o script: ***./buildall 199X249X38_Nproc*** (cria diretório de trabalho *../199X249X38_Nproc*)

3. Entrar no diretório *../199X249X38_Nproc/scripts* e submeter na fila*:* ***qsub filaUna.nqs***

As configurações para submeter o *job* estão todas **automatizadas**, ou seja, definir o número de processadores e modificar os *scripts* de rodada não são necessários.

IMPORTANTE: O modelo esta preparado para rodar nos passos acima, apenas para a resolução ***199X249X38***, tendo como ponto central (-23º, -45º) localizado na Serra do Mar (vide *set_parmeta_orig*).

Exemplo dos arquivos gerados pelo script gera_parmeta.sh:

Arquivo set_parmeta_199X249X38_4proc:

```
.
.
.
############################
#
## These should both be =1 unless using multiple CPUs on a machine
## using "real" MPI (as opposed to the included dummy MPI library).
##
## INPES X JNPES = total number of CPUs to be used
##
INPES=1
JNPES=4
#
###################################
.
.
.
```

Arquivo *run.com_real_mpi :*

```
.
.
.
# Rodada do modelo usando arquivo de machinefile gerado pelo sistema ----------
# Arquivo salvo pelo script de submissao em fila no diretorio da rodada
# by Lamosa PAD/CPTEC/2007 ----------------------------------------------------
mpirun -np ${npr} -machinefile ../machines ${Model_exe}
.
.
.
```

npr => Contem o número de processadores

Arquivo *filaUna.nqs*:

```
#!/bin/ksh
#$ -pe mpi-fill 5
#$ -o una1:/home/usuario/worketa/eta/199X249X38_4proc/scripts/Eta_5p.out
#$ -j y
#$ -N EtaWs5
#$ -S /bin/ksh
#$ -V
#
cd /home/usuario/worketa/eta/199X249X38_4proc/scripts
cp ${TMPDIR}/machines ../machines
./start.ksh 2006120412
```

Note que o número de processos informados nos arquivos *run.com_real_mpi* e *filaUna.nqs* será 4 (nós de processamento) + 1 (servidor de I/O). Dessa forma, passamos para o modelo que apenas um nó será responsável pela escritas dos arquivos de saída. Esta é a forma comumente usada no CPTEC/INPE neste modelo.