Universidade de Passo Fundo
Instituto de Ciências Exatas e Geociências
Curso de Ciência da Computação

Instruções para uso do MPI
- Relatório Técnico -

ComPaDi – Grupo de Pesquisa em Computação Paralela e Distribuída

Passo Fundo - RS
Agosto de 2002

SUMÁRIO

1. Obtenção 3
2. Instalação 4
3. Configuração 5
4. Compilação e Execução do MPI 7
5. Rotinas Básicas do MPI 8
5.1. MPI_INIT 8
5.2. MPI_Comm_Rank 8
5.3. MPI_Comm_Size 9
5.4. MPI_Send 9
5.5. MPI_Recv 10
5.6. MPI_Finalize 10

1. OBTENÇÃO

Neste tutorial vamos abordar a versão MPILAM. http://www.lam-mpi.org é o site oficial do MPILam, nele você encontrará tudo documentações, howtos, fontes do mpi, softwares para downloads entre outras coisas.

- Download do software: http://www.lam-mpi.org/download/

A versão mais recente do MPILam, é a 6.5.7, e está disponível no formato tar.gz, sendo o nome completo do arquivo: lam-6.5.7.tar.gz. Abaixo estão descritos os procedimentos para a instalação do MPI em ambiente UNIX, sendo que foram feitos teste em computadores com sistemas operacionais Linux, distribuições Red Hat 7.3 e Yellow Dog, além do sistema operacional Solaris, da SUN.

2. INSTALAÇÃO

Após ter obtido o arquivo.tar.gz cós os arquivos do MPI, é só instalar seguindo os seguintes passos.

```
[root@dunga teste]# gzip -d lam-6.5.6.tar.gz
[root@dunga teste]#tar -xvf lam-6.5.6.tar
[root@dunga lam]# mkdir /tools
[root@dunga lam]# ./configure --prefix=/tools/lam656 --without-fc
[root@dunga lam]# make
[root@dunga lam]# make install
```

Observações:

- gzip –d lam-6.5.6.tar.gz – Descompacta o arquivo .gz
- tar –xvf lam-6.5.6.tar – Desarquiva o conjunto de arquivos .tar
- mkdir /tools – Cria a pasta onde será feita a instalação como foi definido no .bashrc do usuário.
- ./configure –prefix=/tools/lam656 –without-fc – Define o local de instalação do mpi
- make – 1º passo para instalação
- make install – Instalação do MPI
- Feito tudo isso o MPILam já esta instalado

3. CONFIGURAÇÃO

Para o funcionamento do MPILam devemos configurar os seguintes arquivos:

- /etc/hosts

Nesse arquivo devem ser colocados os nomes das máquinas e o endereço IP correspondente a cada uma delas.

```
# Exemplo do arquivo hosts (/etc/hosts)
# Do not remove the following line, or various programs
# that require network functionality will fail.
192.168.115.240  lcp240.upf.tche.br      lcp240
192.168.115.172  sparc10.upf.tche.br     sparc10
192.168.115.173  pcserver2.upf.tche.br   pcserver2
192.168.115.15   pcserver.upf.tche.br    pcserver
192.168.115.4    imac.upf.tche.br  imac
```

Observação: As máquinas acima citadas poderão ou não fazer parte da Máquina Virtual, utilizada pelo MPI.

- /etc/hosts.equiv

O MPI necessita que esteja habilitado e rodando o serviço RSH. O rsh, ou Remote Shell, é usado para possibilitar a execução de uma aplicação a partir de um computador em outro. Este serviço deve ser configurado para não necessitar de login e senha.

Após ter instalado o rsh, é necessário fazer a configuração do mesmo e para isso, basta habilitar os serviços rlogin e rsh com o comando *ntsysv* e marcar estas opções. Feito isso já está habilitado o serviço de rsh, agora falta liberar este serviços para as máquinas acessarem. Para isso é necessário criar o arquivo hosts.equiv no diretório /etc. Criado este

arquivo deve-se editá-lo, colocando o nome das máquinas que podem acessar a máquina sendo configurada.

```
#Exemplo do arquivo hosts.equiv (/etc/hosts.equiv)
lcp240
imac
pcserver
pcserver2
sparc10
```

Esse procedimento deve ser feito em todas as máquinas que se pretende usar o MPI.

Observação: Estes nomes que serão incluídos no arquivo hosts.equiv, devem estar obrigatoriamente no arquivo hosts, porque é a partir deste arquivo que ele verifica o endereço IP de cada máquina.

- /home/usuário/.bashrc

  Nesse arquivo deverão inseridos alguns parâmetros necessários para o MPI.

  - LAMHOME="/tools/lam656"
  - PATH=$PATH:$LAMHOME/bin
  - ENV=$HOME/.bashrc
  - USERNAME=""export USERNAME ENV PATH LAMHOME

4. COMPILAÇÃO E EXECUÇÃO DO MPI

- Compilar o programa

```
mpicc -o nomedoprograma nomedoprograma.c
```

- Criar a máquina virtual paralela

```
lamboot -v mpi_hosts
```

- Rodar a aplicação

```
mpirun -np 5 nomedoexecutavel
```

- Destruir a máquina virtual paralela

```
wipe -v mpi_hosts
```

5. ROTINAS BÁSICAS DO MPI

5.1. MPI_INIT

Função de inicialização de processo MPI. Portanto, deve ser a primeira a ser chamada por cada processo, pois estabelece o ambiente necessário para executar o MPI. Ela também sincroniza todos os processos na inicialização de uma aplicação MPI.

- int MPI_INIT(int *argc, char *argv[])
    - argc – Apontador para a quantidade de parâmetros de linha de comandos;
    - argv – Apontador para um vetor de strings;

5.2. MPI_Comm_Rank

Identifica um processo MPI dentro de um determinado grupo. Retorna sempre um valor inteiro entre 0 e n-1, onde n é o número de processos.

- int MPI_Comm_Rank(MPI_Comm comm, int *rank)
    - comm - Comunicador do MPI;
    - rank – Variável inteira com o numero de identificador do processo;

5.3. MPI_Comm_Size

Retorna o número de processos dentro de um grupo.

- int MPI_Comm_Size(MPI_Comm comm, int *size)
    - comm – Comunicador do MPI;
    - size – Variável interna que retorna o numero de processos iniciados pelo MPI;

5.4. MPI_Send

Rotina básica para envio de mensagem.

- int MPI_Send(void *sndbuf, int count, MPI_Datatype dtype, int dest, int tag, MPI_Comm comm)
    - sndbuf – Identificação do buffer (endereço inicial do “application buffer” – de onde os dados serão enviados);
    - count – Número de elementos a serem enviados;
    - dtype – Tipo de dado;
    - dest – Identificação do processo destino;
    - tag – Rótulo (label) da mensagem;
    - comm – MPI Communicador;

5.5. MPI_Recv

Rotina básica para o recebimento de mensagem.

- int MPI_Recv(void *sndbuf, int count, MPI_Datatype dtype, int dest, int tag, MPI_Comm comm, status)
    - sndbuf – Identificação do buffer (endereço inicial do “application buffer” – onde os dados serão recebidos);
    - count – Número de elementos a serem enviados;
    - dtype – Tipo de dado;
    - dest – Identificação do processo destino;
    - tag – Rótulo (label) da mensagem;
    - comm – MPI Communicador;
    - status – Status;

5.6. MPI_Finalize

Finaliza um processo MPI.

- int MPI_Finalize(void)
    - Não tem parâmetros;