MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

ESTUDO DA DINÂMICA DE FULIGEM EM ESCOAMENTO COM FORTE GRADIENTE DE TEMPERATURA: SIMULAÇÃO

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA
(PIBIC/CNPq/INPE)

Celso Thiago Silva Barbosa (Unisal, Bolsista PIBIC/CNPq)
E-mail: ct.barbosa@gmail.com

Dr. Fernando Fachini Filho (LCP/INPE, Orientador)
E-mail: fachini@lcp.inpe.br

Julho de 2007

SUMÁRIO

1. INTRODUÇÃO

Este trabalho tem como objetivo o aprendizado da computação paralela e para isso utilizará o estudo da dinâmica das partículas de fuligem geradas durante a combustão de gota individuais. Este caso fundamental tem uma grande importância na determinação da formação de fuligem na combustão de sprays. Este tipo de combustão é encontrado tanto no transporte de superfície bem quanto no aéreo. A preocupação com as condições ambientais em nossos dias obriga os equipamentos de combustão a serem menos poluentes. Este fato está levando ao detalhamento dos processos de formação de fuligem e o de transporte. A necessidade de se empregar código com alta performance e paralela se deve ao enorme numero de partículas a serem acompanhadas ao mesmo tempo. Nesta fase do projeto (Janeiro/2007 a Julho/2007) foi desenvolvido um cluster utilizando 4 computadores com o sistema operacional Linux, e será mostrado em capítulo abaixo a sua instalação, configuração e seus testes.

2. **CLUSTER**

2.1 Conceito

Cluster é um conjunto de computadores interligados em rede, cujo objetivo é distribuir todo processamento de uma aplicação aos demais computadores de forma que se pareçam com um supercomputador e assim obter os resultados rapidamente.

Segundo Alecrim (2004), cada computador de um cluster é denominado nó ou nodo. Todos devem ser interconectados, de maneira a formarem uma rede, de qualquer topologia. Essa rede precisa ser criada de uma forma que permita o acréscimo ou a retirada de um nó (em casos de danos, por exemplo), mas sem interromper o funcionamento do cluster.

Yokokura (2005) afirma que os *clusters* podem ser divididos em duas categorias básicas: Alta Disponibilidade (*HA- High Availability*) e Alta Performance de Computação (*HPC- High Performance Computing*).

Os *Clusters HA* tem a finalidade de manter um determinado serviço de forma segura com maior durabilidade de tempo. O *cluster* de *HPC* é uma configuração designada a prover grande performance de computação, superior ao que um microcomputador individual poderia oferecer. (SESTARI, 2005)

Foto de um *cluster* com 16 computadores

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

Vantagens em se utilizar Cluster de Computadores

Atualmente grandes empresas necessitam de um grande poder computacional a um baixo custo. Empresas de menor porte, com poucos recursos financeiros podem conseguir um processamento de alto desempenho, a custo baixo, aproveitando computadores que já possuem.

Desempenho e tolerância a falhas, que são os requisitos mais importantes e pretendidos, são oferecidos pelos clusters. Essa vantagem é obtida através do simples acréscimo de computadores ao sistema. (SANTANA; MOCELLIN, 2005)

Dentre as inúmeras vantagens em se utilizar clusters, pode-se se destacar algumas:

- **Alto Desempenho** – possibilidade de se resolver problemas muitos complexos através do processamento paralelo, o que diminui o tempo de resolução do problema;
- **Escalabilidade** – possibilidade de que novos componentes sejam adicionados à medida que cresce a carga de trabalho;
- **Tolerância a Falhas** – o aumento de confiabilidade dos sistema como um todo, caso alguma parte falhe;
- **Baixo custo** – a redução de custo para se obter processamento de alto desempenho utilizando PCs;
- **Independência de fornecedores** – utilização de hardware aberto, software de uso livre e independência de fabricantes e licenças de uso. (PITANGA, 2004)

Áreas de Aplicação

Os *clusters* podem ser utilizados em diversas áreas, ou seja, em qualquer aplicação que necessite um grande poder computacional.

Exemplo de algumas áreas onde a utilização de um *cluster* pode ser indicada:

- **Servidores da Internet** – o grande crescimento de utilização da internet revelou fragilidade nos sites muito visitados. Um cluster pode distribuir a carga e aumentar a capacidade de resposta;

- **Segurança** – a grande capacidade que o processamento paralelo oferece beneficiará qualquer processo para identificação de quebra na segurança e verificação de possíveis soluções;
- **Bases de Dados** – pesquisas intensivas em Banco de dados podem demorar muito tempo em um sistema comum. A utilização de um cluster pode reduzir esse tempo significativamente;
- **Computação Gráfica** – nessa área, é muito comum o tempo de processamento ser uma limitação para a evolução da qualidade dos projetos. A utilização de um cluster pode diminuir o tempo de renderização de imagens durante a elaboração de um filme, por exemplo;
- **Aerodinâmica** – produção de novas capacidades tecnológicas e econômicas na pressão enfrentada em aeronaves, no lançamento de naves espaciais e nos estudos de turbulência;
- **Análise de elementos finitos** – cálculos de barragens, pontes, navios, aviões, grandes edifícios, veículos espaciais;
- **Aplicações em sensoramento remoto** – análise de imagens de satélite, para obtenção de informações sobre agricultura, florestas, geologia, fontes híbridas;
- **Inteligência artificial e automação** – processamento de imagens, reconhecimento de padrões, visão por computador, reconhecimento de voz máquinas de inferência;
- **Engenharia genética** – projeto Genoma;
- **Exploração sísmica** – empregado especialmente pelas companhias petrolíferas para determinação de local de poços de petróleo;
- **Oceanografia e astrofísica** – exploração de recursos dos oceanos, estudo da formação da terra, dinâmica das galáxias;
- **Previsão do tempo** – um processo demorado e com pouca precisão quando gerado através de computadores seqüenciais;
- **Pesquisas militares** – projeto de armas nucleares, simulação de efeitos das armas, em especial as radioativas, processamento de sinais de radares para comandos de mísseis antibalísticos, geração automática de mapas, acompanhamento de submarinos submersos;
- **Problemas de pesquisa básica** – em química, física e engenharia, tais como mecânica quântica, mecânica estatística, química de polímeros, crescimento de cristais, análise de trajetória, dinâmica de fluidos, teoria do campo quântico, dinâmica

molecular, equações de circuitos de grande escala, distribuição de conexões e circuitos VLSI;

- **Segurança de reatores nucleares** – análise das condições do reator, controle automático, treinamento através de simulações de operações, atuação rápida em caso de acidentes. (PITANGA, 2004)

Alecrim (2004) diz que basicamente, qualquer tipo de aplicação crítica, ou seja, aplicações que não podem parar de funcionar ou não podem perder dados (como os sistemas de bancos, por exemplo), podem utilizar as tecnologias de *cluster*, desde que devidamente configurados para não serem sujeitas à falhas graves. Assim, o *cluster* deve contar com *nobreaks* ou geradores que garantam o funcionamento do sistema mesmo nos casos de queda de energia, além de meios de manutenção e detecção de falhas eficientes.

2.2 TIPOS DE CLUSTERS

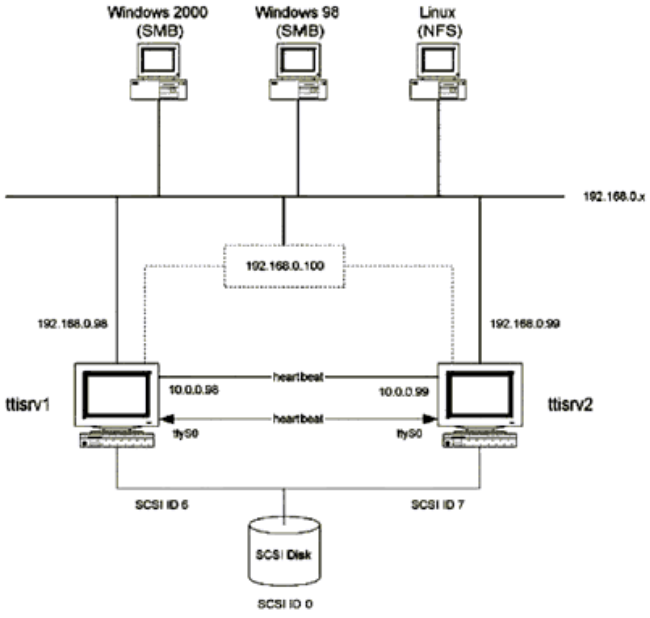
2.2.1 Cluster de Alta Disponibilidade

Com o crescimento do uso de computadores em ambientes empresarias, comercias e até mesmo doméstico, a parada não planejada de equipamentos pode causar danos e prejuízos na qualidade dos serviços.

Para solucionar este problema, podem-se utilizar *clusters* de alta disponibilidade, que garantem o funcionamento do sistema, mesmo com a paralisação do parcial do *cluster*, seja ela de *hardware* ou *software.*

Segundo Pitanga (2004), *clusters* de alta disponibilidade os equipamentos são usados em conjunto para manter um serviço ou equipamento sempre ativo, replicando serviços e servidores, o que evita máquinas paradas, ociosas, esperando apenas o outro equipamento ou serviço paralisar, passando as demais responder normalmente. É claro que com isso poderemos ter perda de performance e poder de processamento, mas o principal objetivo será alcançado, ou seja, não paralisar o serviço.

De maneira geral um servidor de boa qualidade mantém uma disponibilidade de 99, 5%, enquanto uma solução através de *clusters* de computadores mantém 99, 99% de disponibilidade (PITANGA, 2004).



Cluster de Alta Disponibilidade

Fonte: http://www.clubedohardware.com.br/artigos/153

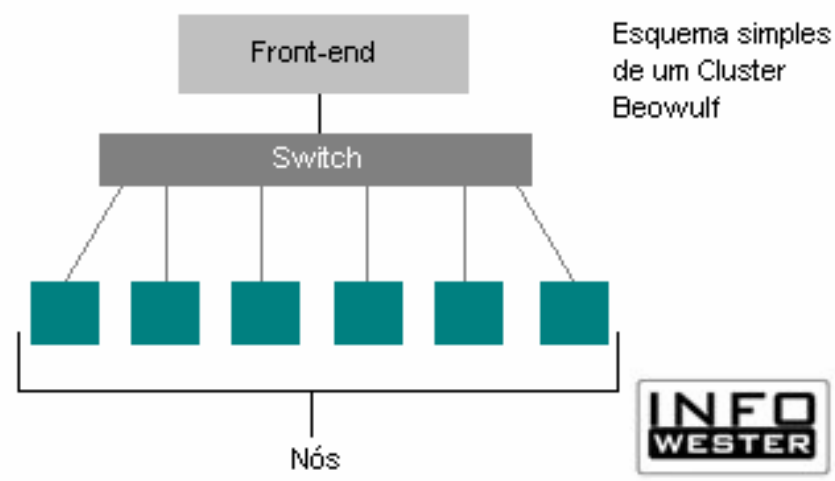
2.2.2 Cluster de Alta Performance de Computação

O *cluster HPC* é um tipo de sistema para processamento paralelo ou distribuído que consiste em uma coleção de computadores interconectados, unificados como um recurso de computação simples e integrado. Um nó do *cluster* pode ser um simples sistema multiprocessador (*PC's*, estações de trabalho ou *SMP's*) com memória, dispositivos de entrada/saída de dados e um sistema operacional. No entanto esse sistema pode oferecer características e benefícios (serviços rápidos e confiáveis), encontrados somente em sistemas com memória compartilhada, como os supercomputadores. (PITANGA, 2004).

2.2.3 Cluster Beowulf

Com o aumento do desempenho e o baixo custo de computadores pessoais, torna-se cada vez mais viável o uso destes equipamentos no desenvolvimento de um cluster (supercomputador). Com essa vantagem e a necessidade de atingir altos índices de processamento em diversas áreas, surge então o Cluster Beowulf.

Beowulf é uma arquitetura de vários computadores utilizados para computação paralela, que consiste em um conjunto de máquinas, e é formado por um nó servidor e nós clientes conectados através de uma rede (Ethernet ou outra topologia qualquer), com um sistema operacional que implemente estruturas de Máquinas Virtuais Paralelas (PVM) e/ou Passagem de Mensagens, para realizar uma comunicação que otimize o processamento paralelo. (SANTANA; MOCELLIN, 2005)



Esquema de um *cluster* Beowulf

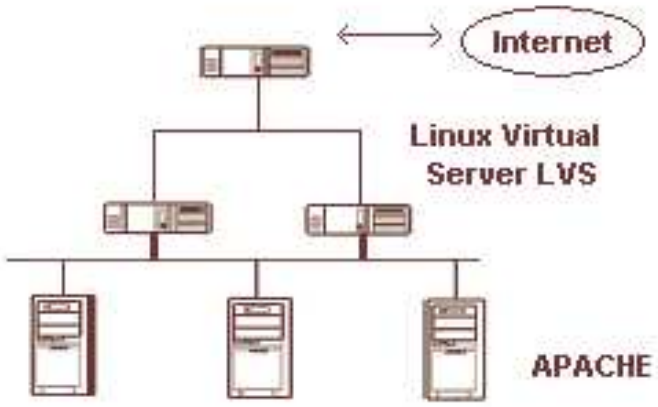
Fonte: http://www.infowester.com/cluster.php

2.2.4 Cluster de Balanceamento de Carga

Segundo Yokokura (2005), esse tipo de *cluster* é especialmente utilizado em serviços de comércio eletrônico e provedores de *internet*, que necessitam resolver diferenças de carga provenientes de múltiplas requisições de entrada em tempo real.

Abreu e Gorayeb (2004) afirmam este tipo de cluster consiste em dividir as requisições ou processos que chegam entre os computadores pertencentes ao sistema, para que não haja o gargalo de informações, ou seja, fazer com que o computador não fique muito atarefado prejudicando assim o desempenho do sistema. O cluster de balanceamento de cargas também segue o paradigma que para o usuário este balanceamento será totalmente transparente. Essa transparência pode ser feita através de um software específico ou por um simples redirecionamento de DNS (*Domain Name Service*).

Quando se colocam dois computadores com a mesma capacidade de resposta para atender uma mesma requisição, sem fazer o balanceamento de cargas, pode acontecer de os dois computadores responderem a mesma requisição, prejudicando assim a conexão. Para evitar isso, coloca-se um software de balanceamento de carga que fará o gerenciamento entre os servidores e o cliente. (ABREU; GORAYEB, 2004).



Cluster de Balanceamento de carga

Fonte: (ABREU; GORAYEB, 2004).

3. MPI – INTERFACE DE PASSAGEM DE MENSAGENS

O que é MPI?

O MPI é uma biblioteca com funções para troca de mensagens, responsável pela comunicação e sincronização de processos em um *cluster* paralelo. Dessa forma, os processos de um programa paralelo podem ser escritos em uma linguagem de programação seqüencial, tal como C ou Fortran. O principal objetivo do MPI é disponibilizar uma interface que seja largamente utilizada no desenvolvimento de programas que utilizem troca de mensagens. Além de garantir a portabilidade dos programas paralelos, essa interface deve ser implementada eficientemente nos diversos tipos de máquinas paralelas existentes (reais ou *clusters* de *workstations*). (PITANGA, 2004)

O MPI possui um ambiente prático, é portável para qualquer arquitetura e eficiente para o desenvolvimento de programas paralelos de alta performance. Possui ferramentas para análise de performance e tem 129 funções que oferecem os seguintes serviços:

- Comunicação ponto-a-ponto;
- Comunicação coletiva;
- Suporte para grupo de processos;
- Suporte para contextos de comunicação;
- Suporte para topologia de processos.

Distribuições do MPI

O padrão MPI possui diferentes implementações, sendo elas de domínio público ou privado. Dentre as implementações de domínio público as mais conhecidas são:

- MPICH – *Argonne National Laboratory Missipi State University*;
- LAM – *Ohio Supercomputer Center*;
- MPIAP – *Australian National Univerity*;
- MPI-FM – *University of Illinois*
- CHIMP/MPI – *Edinburgh Paralel Computing Center*

Implementações da IBM (MPI-F e MPL) e da HP estão no grupo das de domínio privado.

Conceitos

Rank: Todo processo possui uma identificação única recebida pelo sistema quando inicializado. Essa identificação é representada por número inteiro que varia de 0 até n-1 processos.

Group: É um conjunto ordenado de N processos, onde qualquer grupo é associado a um comunicador, que faz parte de um grupo preestabelecido chamado *MPI_COMM_WORLD*.

Communicator: Conjunto de processos que podem comunicar-se entre si No MPI, as mensagens somente serão transmitidas se estiverem dentro de um comunicador.

Application Buffer: É um endereço de memória que armazena um dado ou mensagem que o processo necessita enviar ou receber (Ex: uma variável).

System Buffer: É um endereço de memória reservado pelo sistema para armazenar as mensagens.

Blocking Comunication: Uma rotina de comunicação é denominada bloqueante se a finalização de uma chamada depender de determinados eventos.

Non-Blocking Communication: Uma rotina de comunicação é denominada **não-bloqueante** se a chamada retorna sem esperar qualquer evento que indique o fim ou o sucesso da rotina. Característica de uma comunicação assíncrona.

Synchronous Send: Bloqueia até que ocorra um "receive" correspondente no processo de destino. Pode ser "blocking" ou "non-blocking".

Buffered Send: O MPI permite que se usem buffers para armazenar temporariamente a mensagens até que elas possam ser enviadas, liberando a aplicação.

Standard Send: Operação básica de envio de mensagens usada para transmitir dados de um processo para o outro.

Ready Send: Se o destinatário tiver feito um receive, então a mensagem vai direto para ele.

Rotinas Básicas

MPI_INIT: Primeira rotina do MPI a ser chamada por cada processo. Estabelece o ambiente necessário para executar o MPI.

MPI_COMM_RANK: Identifica o processo, dentro de um grupo de processos.

MPI_COMM_SIZE: Retorna o número de processos dentro de um grupo de processos.

MPI_SEND: A rotina retorna após o dados ser enviado.

MPI_RECV: A rotina retorna após o dado ter sido recebido e armazenado.

MPI_FINALIZE: Finaliza o processo para o MPI. Última rotina a ser executada por uma aplicação MPI.

4. DESENVOLVIMENTO

4.1 Instalação e Configuração do Cluster

Tendo quatro computadores com o sistema operacional Linux Debian 3.0 instalado, a primeira coisa foi colocar todas as máquinas do cluster em rede e para isso todos os arquivos hosts (encontrados em /etc) foram alterados.

Em **/etc/hosts** estão os nomes e ips dos computadores que farão parte do cluster:

node0 10.0.0.1 node0.beowulf.lcp.inpe.br node0
node1 10.0.0.2 node1.beowulf.lcp.inpe.br node1
node2 10.0.0.3 node2.beowulf.lcp.inpe.br node2
node4 10.0.0.4 node0.beowulf.lcp.inpe.br node3

Em **/etc/hosts.allow** estão os ips dos computadores que terão acesso a rede do cluster:

portmap: 10.0.0.1, 10.0.0.2, 10.0.0.3, 10.0.0.4
mountd: 10.0.0.1, 10.0.0.2, 10.0.0.3, 10.0.0.4
lockd: 10.0.0.1, 10.0.0.2, 10.0.0.3, 10.0.0.4
rquotad: 10.0.0.1, 10.0.0.2, 10.0.0.3, 10.0.0.4
statd: 10.0.0.1, 10.0.0.2, 10.0.0.3, 10.0.0.4

Em **/etc/hosts.deny** o acesso a rede do cluster está fechado para qualquer computador:

portmap: ALL
mountd: ALL
lockd: ALL
rquotad: ALL
statd: ALL

No fim do arquivo **/etc/securetty** foram adicionados os comandos rsh e rlogin, para que o usuário root também possa acessar sem senha.

O arquivo **/etc/hosts.equiv** foi configurado para definir a relação de confiança entre os hosts através de equivalência, para que não haja a necessidade de autenticação

por senha. Este arquivo é requerido pelo protocolo de acesso RSH (Remote Shell) para ser capaz de acessar todas as máquinas do cluster.

Porém após ter feito os procedimentos acima, o rsh sem senha não funcionou. A partir disso a comunicação das máquinas passou a ser feita via ssh (Secure Shell) sem senha, onde foi criada em cada uma das máquinas uma chave de acesso e esta chave deveria ser copiada para as demais máquinas. Em uma visita ao CTA, foi sugerida que a comunicação fosse feita via rsh, pois se houvesse a necessidade de adicionar mais computadores ao cluster, a configuração seria mais fácil e rápida. E para que o RSH funcionasse corretamente, foi preciso instalar dois pacotes, o rsh-server e rsh-client .

Durante a instalação dos pacotes foi necessário fazer uma atualização do sistema e nas máquinas escravas (back-end) foi instalado o sistema operacional Debian 3.1 e o sistema da máquina mestre (front-end) foi mantido.

Após a instalação do sistema operacional, toda a parte de rede foi configurada e os pacotes para o rsh sem senha foram instalados manualmente, pois somente a máquina mestre possuía acesso à internet. Para que todas as máquinas tivessem acesso à internet todos os IPs internos foram substituídos por IPs externos e no arquivo **/etc/resolv.conf** de cada computador foi incluído o nome do servidor de Internet.

Devido a mudança de ips, o computador node0 passou a se chamar linan.

Com a Internet funcionando os pacotes essenciais para o funcionamento do cluster foram instalados.

Após os pacotes serem instalados, foi criado em cada diretório dos usuários (/home e /root) o arquivo .**rhosts,** que contém o nome dos computadores que farão a comunicação sem autenticação.

.rhosts

linan

node1

node2

node3

Ainda temos que solucionar o problema com a rede de uma das máquinas, onde a conexão com as demais é perdida depois de um tempo, sendo necessário reinicializá-la. À partir disso todas as máquinas estavam se comunicando sem senha.

4.2 Instalação do MPI

Seguimos então para a instalação do MPI (Message Passing Interface), que é uma biblioteca de subrotinas de comunicação, desenvolvidas em linguagem C e que são utilizadas no desenvolvimento de programas para serem executados em mais de um processador, simultaneamente. Devido à biblioteca MPI ser privativa foi instalada a biblioteca MPICH que é de domínio público, através do seguinte comando:

apt-get install mpich

Com o mpich instalado, o arquivo **/etc/mpich/machines.LINUX** foi atualizado com os nomes dos computdores do cluster.

Foram feitos testes com programas bem conhecidos, como o cálculo do valor de PI e Hello World, para conhecer o desempenho da rede

A compilação do programa deverá ser feita da seguinte maneira:

mpif77.mpich -o executavel arquivo.f

O arquivo executável deverá ser copiado para todas máquinas em mesmo diretório.

Exemplo:
rcp arquivo usuário@node1:/home/usuário/programas
rcp arquivo usuário@node2:/home/usuário/programas
rcp arquivo usuário@node3:/home/usuário/programas

O programa é executado à partir da máquina mestre com o comando abaixo:

mpirun.mpich –v –np proc executável

Onde *proc* é o número de processadores que serão utilizados na execução do programa.

4.3 Programas de testes

Hello World

```
      program main

       INCLUDE 'mpif.h'
       INTEGER err

       CALL MPI_INIT(err)
       PRINT *, "Ola mundo!"
       CALL MPI_FINALIZE(err)

       end
```

Cálculo do valor de PI

```
      program main

      include 'mpif.h'

      double precision  PI25DT
      parameter        (PI25DT = 3.14159265358979323846264338327950288419716939937510d0)

      double precision  mypi, pi, h, sum, x, f, a
      integer n, myid, numprocs, i, rc
c                              function to integrate
      f(a) = 4.d0 / (1.d0 + a*a)

      call MPI_INIT( ierr )
      call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
      call MPI_COMM_SIZE( MPI_COMM_WORLD, numprocs, ierr )
      print *, 'Process ', myid, ' of ', numprocs, ' is alive'

      sizetype   = 1
      sumtype    = 2

 10   if ( myid .eq. 0 ) then
        write(6,98)
```

```
 98      format('Enter the number of intervals: (0 quits)')
         read(5,99) n
 99      format(i10)
       endif

       call MPI_BCAST(n,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
c                            check for quit signal
       if ( n .le. 0 ) goto 30

c                            calculate the interval size
       h = 1.0d0/n

       sum  = 0.0d0
       do 20 i = myid+1, n, numprocs
         x = h * (dble(i) - 0.5d0)
         sum = sum + f(x)
 20    continue
       mypi = h * sum

c                            collect all the partial sums
       call MPI_REDUCE(mypi,pi,1,MPI_DOUBLE_PRECISION,MPI_SUM,0,
      $     MPI_COMM_WORLD,ierr)

c                            node 0 prints the answer.
       if (myid .eq. 0) then
         write(6, 97) pi, abs(pi - PI25DT)
 97      format('  pi is approximately: ', F18.16,
     +          '  Error is: ', F18.16)
       endif
       goto 10

 30    call MPI_FINALIZE(rc)
       stop
       end
```

5. CONCLUSÕES E TRABALHOS FUTUROS

A bolsa, que teve início em janeiro de 2007, me proporcionou novos conhecimentos relacionados ao Linux e noções de programação paralela baseada em MPI. Por ter acesso, como usuário *root*, ao sistema operacional Linux, aprimorei meus conhecimentos em relação à administração do mesmo, que era inicialmente limitada aos comandos básicos, e também adquiri conhecimentos relacionados a montagem e configuração de uma rede de computadores.

Durante o período da bolsa, recebi todo o suporte necessário para a realização do trabalho que me foi designado. O ambiente de trabalho e os equipamentos oferecidos proporcionaram um bom desempenho durante toda a execução das atividades.

Após o termino da bolsa, em julho de 2007, pretendo continuar estudando o funcionando de *clusters* e aprofundar meus conhecimentos em programação paralela (MPI).

6. REFERÊNCIAS BIBLIOGRÁFICAS

ABREU, Harley Oliveira; GORAYEB Inácio Leite. *Construindo Cluster Beowulf com Software Livre.*2004. 66f. (Graduação em Ciência da Computação) – Universidade da Amazônia – Centro de Ciências Exatas e Tecnologia. Belém, 2004.

ALVES, Rodrigo Sanger. *Ferramenta de Gerenciamento de Clusters de Alto Desempenho Uitlizando SNMP.* 2003. 63f. (Graduação em Ciência da Computação) – Universidade Federal do Rio Grande do Sul – Instituto de Informática. Porto Alegre, 2003.

ANDRUCIOLI, Alexandre Pinaffi; PAZOTI, Mario Augusto. *Configurando e usando o PVM.* Disponível em: <http://www.comlinux.com.br/docs/comofazer/pvm.shtml>. Acesso em: 22 nov. 2005.

ALECRIM, Emerson. *Cluster: principais conceitos.* Disponível em: <http://www.linuxclube.com/colunas/13,06,2004.php>. Acesso em: 10 nov. 2005.

BALL, Bill; DUFF Hoyt. *Dominando Linux: Red Hat e Fedora.* São Paulo: Pearson Makron Books, 2004.

COSTA, Marcos Rogério. *Proposta de Solução do Processamento de Grandes Quantidades de Dados com o Uso de Aglomerado de Computadores por Meio da Hibridização de um Cluster de Alto Desempenho com balanceamento de Carga Utilizando Linux.* 2004. 103f. (Graduação em Sistema de Informação) – Sociedade Educacional de Santa Catarina – SOCIESC – Instituto Superior Tupy. Joinville, 2004.

CRAVEIRO, Gisele da Silva. *Um ambiente de execução para suporte à programação paralela com variáveis compartilhadas em sistemas distribuídos heterogêneos*. (Doutorado em Engenharia Elétrica). Escola Politécnica da Universidade de São Paulo, 2003.

DEITEL, Harvey M.; DEITEL, Paul. J.. *Java: como programar*. Porto Alegre: ed. Bookman, 2001.

FAVARI, Lúcio Alexandre. *Instalação de um Cluster de Alto Desempenho com Linux.* 2004. 21f. Monografia (Ciência da Computação) – Universidade São Francisco. São Paulo, 2004.

FILHO, Nélio Alves Pereira. *Serviços de Pertinência para Clusters de Alta Disponibilidade.* 2004. 248f. Dissertação (Ciência da Computação) – Instituto de Matemática e Estatística da Universidade de São Paulo. São Paulo, 2004.

JANG, Michael. *Dominando Red Hat Linux 9.* Rio de Janeiro: Ed. Ciência Moderna Ltda., 2003.