Universidade Federal do Triângulo Mineiro – UFTM

Fábio José Roncolato

Otimização de suprimentos de Tecnologia da Informação por meio da implementação de Cluster computacional na UFTM

Uberaba

2013

Fábio José Roncolato

Otimização de suprimentos de Tecnologia da Informação por meio da implementação de Cluster computacional na UFTM

Dissertação apresentada junto à Comissão de Avaliação do Programa de Mestrado Profissional em Inovação Tecnológica – PMPIT, tendo como área de concentração Inovação Tecnológica e linha de pesquisa Modelagem Matemática Aplicada à Logística/Gestão de Operações do Instituto de Ciências Tecnológicas e Exatas – ICTE da Universidade Federal do Triângulo Mineiro – UFTM como requisito parcial para a obtenção do título de mestre.

Professor Orientador: Dr. Wagner Fernando Delfino Angelotti

Uberaba

2013

Catalogação na fonte: Biblioteca da Universidade Federal do Triângulo Mineiro

Roncolato, Fábio José
R679o Otimização de suprimentos de tecnologia da informação por meio da implementação de cluster computacional na UFTM / Fábio José Roncolato. – 2013.
115 f. : il., fig., graf.

Dissertação (Mestrado Profissional em Inovação Tecnológica) – Universidade Federal do Triângulo Mineiro, Uberaba, MG, 2013.
Orientador: Prof. Dr. Wagner Fernando Delfino Angelotti

1. Tecnologia da informação. 2. Universidades e faculdades. 3. Cluster (Sistema de computador). 4. Simulação (Computadores). I. Angelotti, Wagner Fernando Delfino. II. Universidade Federal do Triângulo Mineiro. III. Título.

CDU 004

FÁBIO JOSÉ RONCOLATO

OTIMIZAÇÃO DE SUPRIMENTOS DE TECNOLOGIA DA INFORMAÇÃO POR MEIO DA IMPLEMENTAÇÃO DE CLUSTER COMPUTACIONAL NA UFTM

Trabalho de conclusão apresentado ao Programa de Mestrado Profissional em Inovação Tecnológica da Universidade Federal do Triângulo Mineiro, como requisito para obtenção do título de mestre.

Uberaba, 22 de maio de 2013

Banca Examinadora:

Prof. Dr. Wagner Fernando Delfino Angelotti
Orientador – PMPIT – UFTM

Prof. Dr. Moacyr Comar Junior
Membro titular – Universidade Federal de São João Del- Rei – UFSJ

Prof. Dr. Luciano Xavier Medeiros
Membro Titular – UFTM

Dedico à minha Mãe Maria Hilda Rodrigues Roncolato e à minha esposa Rayssa Manzan de Mello Roncolato que sempre estão ao meu lado, acreditam no meu potencial e torcem por meu sucesso.

AGRADECIMENTOS

Agradeço primeiramente a Deus que me concedeu saúde e disposição para desenvolver mais esse projeto que me proporcionará muitas conquistas.

Agradeço a meus familiares, à minha esposa, aos amigos e colegas de trabalho que compreenderam minhas ausências e meus anseios para dedicar uma parte significativa do meu tempo à concretização desse projeto.

Agradeço também a meu orientador que não mediu esforços para me acompanhar nessa caminhada num projeto diferente da sua área de atuação efetiva.

Por fim, deixo meu agradecimento a todos que direta ou indiretamente me deram força e contribuíram de alguma maneira para que eu pudesse superar algumas dificuldades encontradas na evolução desse projeto.

RESUMO

As instituições públicas enfrentam entraves na gestão de processos e elaboração de projetos para atender à demanda de equipamentos de Tecnologia da Informação (TI) como bens e serviços. A ausência de equipamentos de grande porte, necessários para o processamento computacional em grande escala, contrapõe-se a um grande número de computadores de pequeno porte com poder computacional subutilizado, destinado a tarefas básicas como navegação na internet, edição de textos ou planilhas e acesso a sistemas de menor demanda de processamento. O custo para a aquisição destes equipamentos é elevado e inviabiliza experimentos computacionais que poderiam fornecer subsídios para comparação ou comprovação de resultados experimentais. Como alternativa, pode-se compartilhar o processamento computacional de microcomputadores de laboratórios de ensino de graduação, por exemplo, que em alguns períodos ficam ociosos e oferecer uma solução, com custo monetário reduzido, beneficiando projetos de pesquisa e colaborações científicas na instituição. Sendo assim, o objetivo principal foi desenvolver um modelo para otimização do uso de recursos computacionais subutilizados na Universidade Federal do Triângulo Mineiro (UFTM) via implementação de um modesto cluster computacional. A partir de pesquisas exploratórias e bibliográficas, estudo de ferramentas, modelos e técnicas para dar suporte e propor soluções aos desafios da implementação de cluster computacional, definiu-se o Mosix na plataforma Linux como padrão de configuração. Criou-se um cluster com alguns microcomputadores “ociosos” de um laboratório do Instituto de Ciências Tecnológicas e Exatas para o processamento de fenômenos químicos comprovando sua eficiência com custo financeiro mínimo. Posteriormente, o objetivo é aumentar o número de computadores no cluster e estender sua aplicação para outros fenômenos e sistemas mais complexos.

Palavras-chave: Tecnologia da informação. Universidades e faculdades. Cluster (Sistema de computador). Simulação (Computadores).

ABSTRACT

Public institutions face barriers to managing processes and elaborating projects in order to meet the demand of Information Technology equipments (IT) such as goods and services. The absence of large apparatus, necessary to the computational processing in large scale, is in contrast to the great number of small size computers with the computational power underutilized, destined to basic tasks such as internet browsing, edition of texts or worksheets and access to systems of lower processing demand. The cost for acquisition of these equipments is high and invalidates computational experiments which could offer subsidies for comparison or authentication of the experimental results. As an alternative, it is possible to share the computational processing of micro-computers from graduation teaching laboratories, for example, which in some periods become idle, and offer a solution, with reduced monetary cost, benefiting research projects and scientific collaborations at the institution. So, the main goal was to develop a model for optimization of the use of computer resources underutilized at the Federal University of Triângulo Mineiro (UFTM) by implementation of a computational cluster model. From the exploratory and bibliographic researches, tool studies, models and techniques to support and propose solutions to the challenges of the implementation of the computational cluster, it was defined Mosix in Linux Platform as pattern of configuration. It was created a cluster with some idle micro-computers from a laboratory of the Institute of Exact Science and Technology for the processing of chemical phenomena proving its efficiency with minimum financial cost. Afterwards, the goal is to increase the number of computers in cluster and extend its application to other phenomena and more complex systems.

Key words: Information technology. Universities and colleges. Cluster (Computer system). Simulation (Computers).

LISTA DE FIGURAS

Figura 1: Representação de um cluster 21
Figura 2: Identificação de processos do GAMESS em execução paralelizada 54
Figura 3: Monitoramento dos processos e recursos entre os nós do cluster 55
Figura 4: Funcionamento do cluster para um teste genérico 57
Figura 5: Comportamento do cluster com Mosix na execução de processos simultâneos 62
Figura 6: Identificação de processos simultâneos controlados pelo Mosix 63
Figura 7: Identificação de processos migrados pelo Mosix 64
Figura 8: Identificação de processos simultâneos pelo Linux 64
Figura 9: Identificação de processos remotos controlados pelo Mosix 65
Figura 10: Obtenção do Mosix2 para sistemas operacionais de 32 bits 79
Figura 11: Obtenção de Mosix3 para sistemas operacionais de 64 bits 80
Figura 12: Visualização de performance de um nó do cluster sem interação 94
Figura 13: Visualização de performance de 03 nós do cluster interagindo 95

Quadro 1: História do Mosix 30
Quadro 2: Configuração de máquinas virtuais 44
Quadro 3: Passos para configuração de rede do cluster 47
Quadro 4: Resultados de testes de fenômenos físico-químicos no Linux 53
Quadro 5: Resultados de testes genéricos para validação do funcionamento do cluster 56
Quadro 6: Resultados dos processos simultâneos utilizando métodos genéricos e Mosix 58
Quadro 7: Resultados de testes com fenômenos físico-químicos no cluster 59
Quadro 8: Resultados dos processos simultâneos utilizando método de Monte Carlo Quântico e Mosix 61
Quadro 9: Configuração do NFS e MPICH 98
Quadro 10: Processo de licitação de um computador de grande porte pela UFTM 107
Quadro 11: Processo de licitação de microcomputadores pela UFTM 109

Tabela 1: Energia Total (E) e potencial de tripla ionização (PTI) de vários estados triplamente ionizados, com participação de elétron de caroço, da molécula de amônia. 60

LISTA DE ABREVIATURAS E SIGLAS

APT-GET – Recurso desenvolvido originalmente para a distribuição Debian que permite a instalação e a atualização de pacotes (programas, bibliotecas de funções, etc.) no Linux de maneira fácil e precisa.

BSD – Berkeley Software Distribution (um sistema derivado do Unix). Utilizado para identificar licença de código aberto.

CH_4 – Gás Metano, um composto orgânico que tem um átomo de carbono ligado covalentemente a quatro átomos de hidrogênio.

CPU – Central Processing Unit (Unidade Central de Processamento).

CSH – C Shell. Sintaxe Shell do Linux modelada segundo a linguagem de programação C.

DNS – Domain Name System (Sistema de Nomes de Domínios).

FTP – File Transfer Protocol (Protocolo de Transferência de Arquivos).

GPLV2 – General Public License Version 2.0 (Licença Pública Geral versão 2.0).

GNU – Sistema Operacional tipo Unix em desenvolvimento pelo Projeto GNU.

GPU – Graphics Processing Unit (Unidade de Processamento Gráfico).

HA – High Availability (Alta Disponibilidade).

HP – Hewlett-Packard©.

HPC – High Performance Computing (Alta Performance de Computação).

HTTP – Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto).

I386 – Processador Intel 80386 (ou 386) que trouxe como principal diferencial de seu antecessor a capacidade de executar multitarefa preemptiva (mais conhecida como multitarefa de antecipação), que em poucas palavras seria atender a diversos processos ao mesmo tempo por prioridade e, além disto, ele utilizava o barramento de 32 bits e memória em modo protegido.

I/O – Input/Output (Entrada/Saída).

IBM – International Business Machines, Marca IBM©.

IGC – Índice Geral de Cursos.

IP – Internet Protocol (Protocolo de Internet).

IPV4 – Internet Protocol Version 4. Versão do Protocolo de Internet onde os endereços IP são compostos por quatro blocos de 8 bits (32 bits no total), que são representados por meio de números de 0 a 255, como "200.156.23.43" ou "64.245.32.11".

ISO – International Organization for Standardization (Organização Internacional de Padronização).

LAN – Local Area Network – Rede Local.

MC – Método de Monte Carlo.

MC68K – É uma família Motorola de microprocessadores CISC 32-bit utilizados em uma ampla gama de dispositivos, concorrendo principalmente com a família x86 da Intel.

MOPI – Biblioteca Entrada/Saída para processamento paralelo no Mosix.

MOSIX – Multicomputer Operating System for UnIX (Sistema Operacional Multicomputador para Unix).

MOSRC – Versão Mosix para processamento em nuvens.

MPI – Message Passing Interface (Interface de Passagem de Mensagens).

MPICH – Message Passing Interface Chameleon.

MRC – MOSIX Reach the Clouds (Versão do Mosix para configuração em nuvens).

NASA – National Aeronautics and Space Administration (Administração Nacional da Aeronáutica e do Espaço).

NE – Neônio.

NFS – Network File System (Sistema de Arquivos de Rede).

NS32332 – Era uma série de microprocessadores de National Semiconductor ("NS", "Natsemi"), provavelmente os primeiros microprocessadores de 32 bits de uso geral no mercado.

OS – Operation System (Sistema Operacional).

PC – Personal Computer (Computador Pessoal).

PDP – Programmed Data Processor. Identificação de uma série de microcomputadores.

PH.D – Philosophy Doctor. Título de Doutor no Brasil.

PIB – Produto Interno Bruto.

PVM – Parallel Virtual Machine (Máquina Virtual Paralela).

QDR – Quad Data Rate (Transferência de Dados Quadruplicada). Refere-se a dispositivos capazes de realizar quatro transferências de dados por ciclo.

SBC – Smart Bitrate Control (Controle Inteligente de Taxa de Bits).

SHELL – Linguagem de script usada em vários Sistemas Operacionais, com diferentes dialetos, dependendo do interpretador de comandos utilizado.

SMP – Symmetric Multiprocessing (Multiprocessamento Simétrico).

SSH – Secure Shell. Programa de computador e Protocolo de Rede que permitem a conexão com outro computador na rede de forma a executar comandos de uma unidade remota.

TCSH – Shell do Unix baseado e compatível com o shell C (csh). É essencialmente o shell C com completa linha de comando programável, edição de linha de comando e algumas outras funcionalidades.

TI – Tecnologia da Informação.

TIC – Tecnologia da Informação e Comunicação.

UNESP – Universidade Estadual Paulista.

UNIX – Sistema Operativo (ou Sistema Operacional) portátil (ou portável), multitarefa e multiutilizador (ou multiusuário).

UTP – Unshielded Twisted Pair. Cabo de rede composto por 4 pares de cabos entrelaçados sem blindagem.

VCL – Framework orientada a objetos baseado em componentes visuais para o desenvolvimento de aplicativos do Microsoft Windows. Em desenvolvimento de software Framework é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

VI – Editor de texto do Sistema Operacional Unix e semelhantes.

VM – Virtual Machine (Máquina Virtual).

X86 – Conjunto básico de instruções utilizado nos processadores para computadores pessoais independentemente de serem Intel ou AMD, Pentium ou 486.

X86-64 – É o conjunto de instruções utilizado pelos processadores de 64 bits da AMD e adotado também pela Intel.

SUMÁRIO

1 INTRODUÇÃO E JUSTIFICATIVA 14
2 OBJETIVO 18
3 FUNDAMENTOS SOBRE CLUSTER 19
3.1 APLICAÇÕES 26
3.2 BENEFÍCIOS DE CLUSTERS 27
3.3 MODELOS DE CLUSTER 28
3.4 TIPOS DE COMPOSIÇÃO DE CLUSTER 29
3.4.1 Computação em grade 29
3.4.2 MOSIX 30
3.4.3 Cluster Beowulf 36
3.4.4 OpenMosix 37
3.5 PARALELIZAÇÃO 39
3.5.1 MPI 39
3.5.2 MPICH 40
3.5.3 PVM 41
3.6 PROBLEMAS E DESVANTAGENS DE CLUSTERS 41
4 PROCEDIMENTOS METODOLÓGICOS 43
4.1 DEFINIÇÕES DO PROJETO 43
4.1.1 Revisão bibliográfica 43
4.1.2 Elaboração do Projeto 43
4.1.3 Adoção de máquinas virtuais para testar cluster 44
4.1.4 Definição do sistema operacional 44
4.1.5 Definição de ferramenta de administração de cluster 45
4.2 INSTALAÇÃO DO CLUSTER 46
4.2.1 Configuração dos equipamentos para composição do cluster 46
4.2.2 Configuração da rede para integração dos elementos do cluster 47
4.2.3 Instalação e configuração do Mosix 49
4.2.4 Configuração de cluster via SSH, NFS e MPICH 51
5 RESULTADOS E DISCUSSÕES 52

5.1 CONFIGURAÇÃO E INSTALAÇÃO DO SOFTWARE GAMESS PARA TESTES DE SIMULAÇÃO DE PROBLEMAS EM FÍSICO-QUÍMICA 52
5.2 TESTES GENÉRICOS 55
5.3 TESTES UTILIZANDO O MÉTODO DE MONTE CARLO 58
5.4 COMPORTAMENTO DO CLUSTER COM TESTE SIMULTÂNEO E MOSIX 62
6 CONCLUSÕES 66
7 CONSIDERAÇÕES FINAIS 68
REFERÊNCIAS 71
APÊNDICE A – VirtualBox 75
APÊNDICE B – Criação e configuração das máquinas virtuais 78
APÊNDICE C – Configuração do cluster por meio do Mosix 79
APÊNDICE D – Instalação e configuração de Cluster via SSH, NFS e MPICH 98
APÊNDICE E – Instalação e configuração do GAMESS 100
ANEXO A – Instalação do Debian 105
ANEXO B – Aquisição de computador de grande porte 107
ANEXO C – Aquisição de microcomputadores 109
ANEXO D – Arquivo de saída de um processamento via GAMESS 110
ANEXO E – Arquivo de saída do teste com programa em Fortran 113

1 INTRODUÇÃO E JUSTIFICATIVA

Administração Pública é a ordenação, direção e controle dos serviços do governo, no âmbito federal, estadual e municipal, segundo os preceitos do direito e da moral, e de acordo com as estruturas legais que, por meio de seu planejamento estratégico, pode dispor de forma eficaz de recursos orçamentários, recursos humanos e materiais que fazem parte de um conjunto de ações planejadas para o cumprimento dos programas e políticas do governo. O poder público se constitui de um conjunto de ações administrativas e jurídicas que possui seus próprios serviços, por meio dos órgãos da Administração direta ou descentralizadamente, por intermédio das entidades autárquicas, fundacionais e empresas estatais que integram a Administração indireta (LIMA, 2009).

As instituições públicas constituem-se de empresas com o foco na prestação de serviços de interesse, utilização e propriedade coletiva, buscando atingir diferentes objetivos (econômicos, políticos, educacionais, sociais, judiciais, culturais, etc.).

Para a realização de suas ações, as instituições públicas adquirem bens, serviços ou suprimentos sob demanda por meio de uma cadeia de processos. Entretanto, a gestão dessas cadeias enfrenta algumas particularidades, na maioria das vezes, incomuns em empresas privadas, ou seja, essas aquisições não ocorrem a qualquer tempo, devendo seguir um rigoroso protocolo que consome maior tempo e onera os processos. Esse protocolo envolve viabilização de recursos, processos licitatórios e imparcialidade na decisão por marcas, modelos e fornecedores, entre outras coisas. Isso nem sempre garante qualidade e prestatividade, visando apenas vantagens nos custos.

Entre tantos bens, serviços e suprimentos, a infraestrutura de Tecnologia da Informação (TI) é imprescindível para a gestão e execução dos processos produtivos em praticamente todos os níveis das organizações. Assim sendo, a TI pode ser compreendida como o envolvimento de "todos os aspectos de computadores (*hardware* e *software*), sistemas de informação, telecomunicação e automação de escritórios" (PALVIA, 1997). Tecnologia da Informação muitas vezes é identificada também pela sigla TIC, que significa Tecnologia da Informação e Comunicação.

Um forte entrave para o bom funcionamento desta cadeia de processos, da qual TI faz parte, é o fato de que todos os governos estão sendo submetidos às restrições orçamentárias, ou seja, fazer mais com menos. Os contratos públicos no Brasil representam 9,5% do PIB, portanto, um processo público que precisa ser bem administrado. Isso requer uma gestão

integrada, segundo os princípios de agilidade, transparência e economia com uso intensivo da TIC (TRIDAPALLI, J.P.; BORINELLI, B., 2010).

A realização de obras, serviços e aquisição de bens realizada pela Administração Pública deve passar previamente por um processo seletivo, que é a licitação pública, normatizada pela Lei n 8.666, de 21 de junho de 1993, e está regulamentada no Art. 37, inciso XXI da Constituição Federal, que instituiu normas para licitações e contratos da Administração Pública (MORIKANE, 2008).

De acordo com Morikane (2008), a Norma ISO 20000 foi criada no contexto das empresas privadas cujas teorias administrativas rogam pela flexibilidade e rapidez, na tomada de decisão e em suas ações. As leis, estatutos e procedimentos inerentes ao setor público, apesar de procurar regulamentar as ações de seus administradores, fazem com que a Administração Pública seja enrijecida e, portanto, existam dificuldades para as tomadas de decisão. Esta norma em uma instituição pública é possível de ser aplicada desde que sejam feitas considerações devido a fatores inerentes ao setor. Conscientizar-se de que os resultados esperados em relação a essa norma não se farão presentes em curto prazo, e sim, em médio e longo prazo, devido às diversas barreiras encontradas, principalmente pelas leis impostas para o setor público, é necessário (MORIKANE, 2008).

No estudo dos preceitos da TI percebe-se que o comprometimento dos serviços executados pelo setor ou a escassez de equipamentos de TI pode afetar a estrutura da cadeia de suprimentos e, principalmente, a gestão de cadeias de procedimentos dentro da instituição. Um eficaz gerenciamento de cadeia de suprimentos requer um eficiente e intensivo uso de TI mediante um sistema integrado que calcule, monitore e interligue todos os fluxos para garantir a entrega dos serviços ou produtos aos clientes nas condições esperadas, dentro dos prazos previstos, e com os menores custos possíveis. Nesse contexto, a TI, ao mesmo tempo em que é imprescindível para a gestão da cadeia de suprimentos, requer uma gestão eficiente dos seus insumos devendo ser parte integrante do nível macro de gestão da instituição.

Muitas atividades de pesquisa nas Instituições Públicas de Ensino dependem de recursos computacionais para o seu desenvolvimento, porém, encontram dificuldades relacionadas aos recursos disponíveis de TI para garantir sua execução com eficiência. Na UFTM esses problemas não são diferentes.

Anteriormente denominada Faculdade de Medicina do Triângulo Mineiro – FMTM, fundada em 1953, foi transformada em Universidade Federal do Triângulo Mineiro – UFTM, no ano de 2005, tendo sua dedicação ao ensino reconhecida por diferentes indicadores nacionais, como o IGC 2008 (Índice Geral de Cursos), que classificou a Instituição com

conceito máximo, posicionando-a entre as melhores, com a 3ª colocação de Minas Gerais e a 6ª do País. Além da tradição no ensino, conquistou, ao longo de 57 anos de existência, o reconhecimento nacional e internacional das atividades de pós-graduação, pesquisa e extensão que desenvolve. Na pesquisa, tem dedicado especial atenção à doença de Chagas, à Esquistossomose, à Leishmaniose e outras doenças tropicais comuns na região.

Em pleno processo de desenvolvimento, a UFTM reconhecidamente mantém sua qualidade, expandindo-a para novas áreas do conhecimento e aumentando a oferta de um ensino que busca contribuir para a ciência e para o desenvolvimento da sociedade (UNIVERSIDADE FEDERAL DO TRIÂNGULO MINEIRO, 2012). Recentemente, com a oferta de cursos de Engenharia, surgiu a demanda por recursos computacionais que garantam processamento "em grande escala". São necessários normalmente equipamentos robustos para processamento de imagens, diagnósticos ou experiências médicas ou de engenharias, cálculos matemáticos complexos, simulação de fenômenos químicos, físicos, biológicos, entre outros. Por outro lado, a aquisição desses tipos de equipamentos é dificultada pelo custo elevado, somada às dificuldades financeiras e, muitas vezes, a ineficiência de políticas que busquem maiores volumes financeiros para investimentos em TI.

Segundo Pitanga (2004 *apud* PEREIRA, 2010), uma forma de proporcionar alto poder computacional é a utilização de Máquinas Paralelas, que utilizam mais de um processador para executar uma mesma tarefa e, assim, conseguir um desempenho melhor. Os tipos mais comuns de máquinas paralelas são: os supercomputadores e os clusters de computadores.

Os supercomputadores são máquinas desenvolvidas com tecnologias proprietárias e, geralmente, possuem finalidades pré-definidas pelo fabricante que acabam gerando desvantagens, como:

a) utilização de softwares proprietários e caros;
b) alto custo de manutenção;
c) total dependência de fornecedores;
d) dificuldade de atualização.

Em contrapartida, clusters são mais acessíveis ao mercado, pois são máquinas construídas com utilização de dois ou mais microcomputadores comuns e de custo mais baixo. Estes são interligados por uma rede e trabalham em paralelo para resolver um determinado problema. O poder de processamento de um cluster é comparável ao de um supercomputador, pois esse utiliza técnicas de processamento distribuído e conta com uma gerência inteligente na distribuição de processos para os equipamentos interligados, que são chamados de nós.

Para muitos problemas enfrentados, uma solução prática, com custo mínimo, pode ser encontrada na otimização, em sua totalidade, dos recursos de TI já disponíveis, aproveitando, assim, o potencial eventualmente desperdiçado em sua ociosidade. Com isso, muitas atividades que dependam de recursos elevados para aquisição de equipamentos com grande capacidade de processamento podem ser colocadas em prática por meio da oferta de recurso computacional gerada com a implementação de cluster computacional utilizando equipamentos subutilizados.

Diante das dificuldades supracitadas e para atender à demanda por computadores de alto desempenho, em particular na UFTM, uma hipótese para viabilizar as referidas atividades é otimizar a utilização de suprimentos de TI mediante a implementação de um cluster computacional composto por microcomputadores subutilizados na instituição que apresentam intervalos ociosos de uso. Assim, mediante uma técnica para composição de um cluster com esses equipamentos, suas capacidades computacionais poderão ser somadas oferecendo condições para aportar processamentos em “grande escala” demandados pelas atividades de pesquisa de diferentes áreas da instituição. Com isso, muitas atividades complexas poderão ser executadas praticamente a custo monetário mínimo pelo reaproveitamento desses equipamentos e utilização de ferramentas gratuitas de gerenciamentos dos mesmos.

A seguir, o trabalho é exposto da seguinte forma: o Capítulo 2 explora os objetivos gerais e específicos; o Capítulo 3 descreve as diversas características e fundamentos sobre cluster; o Capítulo 4 descreve os materiais e métodos utilizados para a elaboração de um cluster computacional; o Capítulo 5 fornece os resultados e discussões sobre as etapas envolvidas na elaboração deste cluster; os Capítulos 6 e 7 apresentam as conclusões e algumas considerações finais sobre o trabalho, respectivamente; e, por fim, a bibliografia do trabalho, apêndices e anexos correspondentes a várias etapas deste trabalho.

2 OBJETIVO

Objetivo Geral

O objetivo do presente trabalho é desenvolver um modelo para otimização do uso de recursos computacionais subutilizados na UFTM mediante a implementação de um cluster computacional.

Objetivos específicos

a) identificar, analisar e representar dados que comprovem o contexto e as dificuldades para aquisição de suprimentos de bens e serviços de TI na UFTM;
b) implementar um cluster computacional otimizando o uso dos recursos de TI subutilizados;
c) elaborar para a UFTM um modelo para implementação de cluster computacional composto por microcomputadores “ociosos”;
d) comparar, se possível, os resultados obtidos entre a técnica desenvolvida e outros recursos existentes para comprovar a viabilidade do projeto.

3 FUNDAMENTOS SOBRE CLUSTER

No início da década de 90, os supercomputadores proprietários eram uma solução muito eficaz para obter maior desempenho em processamento de dados, porém, apresentam um custo demasiadamente alto e também são pouco flexíveis para atualizações do sistema. Assim, nessa mesma década, houve uma tendência gradual em termos de migração por arquiteturas de menor custo financeiro e mais flexíveis como os clusters (TEXEIRA, 2002, *apud* COQUEIRO, 2005).

Originado do inglês, cluster quer dizer: grupo compacto de coisas do mesmo tipo; agrupar (-se); juntar (-se). Especificamente neste caso, cluster é um conjunto de computadores interligados via rede que trabalham em conjunto trocando informações entre si em torno de um único objetivo (PEREIRA FILHO, 2004, *apud* PEREIRA, 2010).

Esses computadores que compõem o cluster são de um modelo tradicional, como por exemplo, de uso doméstico ou de escritório, que possuem sistema completo, processador, memória, unidade de armazenamento de dados e interligados via tecnologia de rede para executarem as tarefas de forma cooperativa, sendo identificados como nós do cluster.

As principais razões para a utilização dos clusters foram:

a) a evolução acentuada em termos de desempenho de estações de trabalho e computadores pessoais, ocorrendo a fabricação de processadores cada vez mais rápidos, contribuindo para que os nós dos clusters tornem-se cada vez mais poderosos em termos de processamento individual;
b) evolução das tecnologias de redes locais, possibilitando a diminuição do custo e o aumento da largura de banda para tráfego de dados, isto é, diminuindo o problema da latência da rede que causa a limitação do desempenho do cluster (TEXEIRA, 2002, *apud* COQUEIRO, 2005);
c) normalização de ferramentas de desenvolvimento para a arquitetura de clusters e sua rápida evolução de acordo com Teixeira (2002, *apud* COQUEIRO, 2005), ou seja, a exemplo da biblioteca de troca de mensagem (MPI), cujo objetivo é padronizar a forma de comunicação dos processos;
d) alternativas para menor custo financeiro em comparação aos supercomputadores proprietários, que são de alto custo.

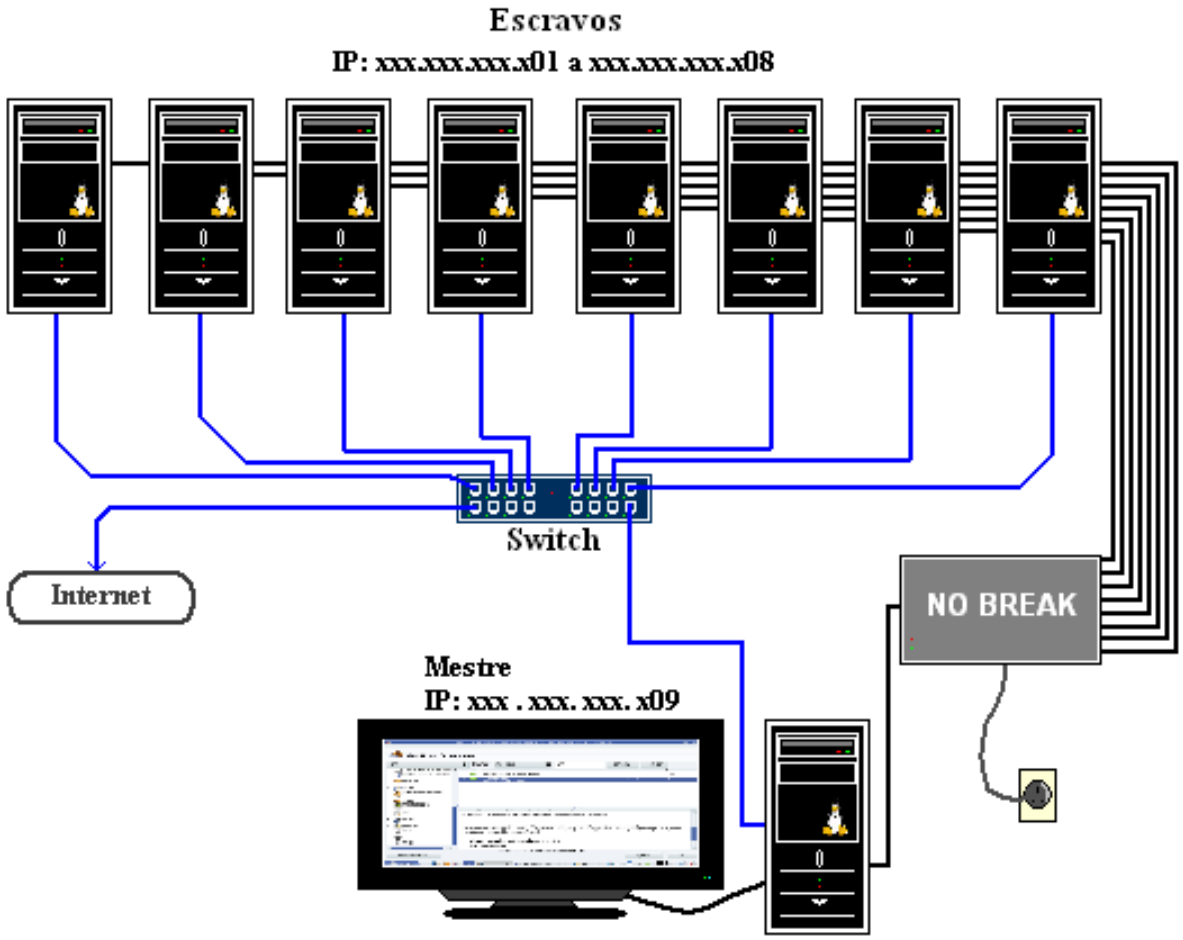
Na sua forma mais básica, um cluster executa aplicações ou serviços, de tal forma que para os usuários que os utilizam tenham a impressão que somente um único sistema responde

para eles, criando, assim, uma ilusão de um recurso único (computador virtual). Este conceito é denominado **transparência do sistema**. Como características fundamentais para a construção destas plataformas incluem-se elevação da confiança, distribuição de carga e desempenho (PITANGA, 2003).

Quando se fala em processamento de alto desempenho, o qual, neste contexto, pode ser entendido como processamento paralelo (conceito explorado mais adiante) e processamento distribuído, imaginam-se muitas pessoas e grandes máquinas dedicadas, que custam milhões de dólares, difíceis de serem operadas e com salas superprotegidas. Entretanto, hoje em dia, devido aos clusters, os custos foram reduzidos com pouca perda de desempenho, o que viabiliza o uso de processamento de alto desempenho na solução de problemas em diversas áreas.

Conforme mostra a Figura 1, normalmente existe um nó mestre, ou centralizador, que distribui as tarefas entre os demais nós, chamados de escravos, que servem apenas para processar isoladamente partes da tarefa como um todo, enviando apenas os resultados para o nó mestre (OLIVEIRA, 2004, *apud* PEREIRA, 2010). Esses elementos do cluster são interconectados numa rede local (LAN) do tipo ethernet por meio de cabeamento UTP (cabo de par trançado) a um switch ethernet 10/100 utilizando o protocolo IPV4. Alguns destes conceitos serão detalhados mais adiante.

Figura 1: Representação de um cluster



Fonte: (TONIDANDEL, 2008)

É vital em um sistema computacional distribuído a existência de uma rede para a conectividade entre os nós. Embora essa habilidade de comunicação entre os nós, direta e indiretamente, seja de extrema importância para o sistema distribuído, o caminho no qual será implementado este canal de comunicação pode variar de sistema para sistema.

Dentre as tecnologias de redes mais comuns usadas em sistemas computacionais distribuídos, segundo Aluvino (2008 *apud* Pitanga, 2002), destacam-se as seguintes:

a) **Ethernet, Fast-Ethernet e Gigabit-Ethernet** – Essas tecnologias são atualmente as mais utilizadas em redes locais (LAN), sendo que Ethernet possui uma taxa de transmissão de 10 Mbps (Megabits por segundo) e foram muito empregadas nos primeiros sistemas distribuídos, porém, essa largura de banda não é mais adequada para se utilizar em computação de alto desempenho, por ser pequena se comparada com recursos computacionais disponíveis atualmente. Fast-Ethernet fornece uma taxa de transmissão de 100 Mbps a um baixo custo, enquanto que Gigabit-Ethernet

fornece largura de banda de 1 Gbps (Gigabits por segundo) a um custo ainda considerado alto.

b) **Giganet** – Foi desenvolvida pela 10 Gigabit Ethernet Alliance. Seus adaptadores de rede suportam uma largura de banda de 1,25 Gbps, além de possuírem switches e 25 ferramentas para gerenciar e testar a rede local. São produtos especialmente desenvolvidos para serem utilizados em sistemas distribuídos de alto desempenho.

c) **Myrinet** – É uma rede de 1,28 Gbps full-duplex (um modo de operação onde dados podem ser transmitidos e recebidos simultaneamente), da empresa Myricom. Esta placa controladora utiliza tecnologia baseada na comunicação via pacotes. É considerada uma rede de alto desempenho, possuindo canais robustos de comunicação com controle de fluxo, controle de erro, baixa latência (tempo de uma rotação), interfaces que podem mapear a rede, rotas selecionadas, tradução de endereços da rede para essas rotas, bem como manipulação de tráfego de pacotes e software que permite comunicação direta entre os processos no nível do usuário e da rede. Tem tolerância a falhas devido ao mapeamento automático da configuração da rede, o que facilita a tarefa de configuração. Devido as suas diversas qualidades, a tecnologia Myrinet é recomendada tanto para sistemas computacionais distribuídos como construção de backbones (termo utilizado para identificar a rede principal, pela qual os dados de todos os clientes da Internet passam, ou a espinha dorsal da Internet) e transferência de imagens, devido à alta taxa de comunicação, utilizada por essas aplicações.

d) **Fibre Channel** – É um conjunto de padrões produzidos pela Fibre Channel Industry Association, que definem uma tecnologia de conexão para transporte de dados de alto desempenho, que pode transportar muitos tipos de dados a velocidades acima de 1 Gbps por intermédio de fios de cobre ou cabos de fibra ótica.

Os principais tipos de equipamentos que podem ser utilizados para a interconexão dos elementos do cluster são:

a) **Hub**: um dispositivo que tem a função de interligar os computadores de uma rede local numa forma de trabalho mais simples se comparado ao switch e ao roteador. Ele recebe dados vindos de um computador e os transmite às outras máquinas. No momento em que isso ocorre, nenhum outro computador consegue enviar sinal. Sua liberação acontece após o sinal anterior ter sido completamente distribuído;

b) **Switch**: um aparelho muito semelhante ao hub, cuja grande diferença é que os dados vindos do computador de origem somente são repassados ao computador de destino. Isso porque os switches criam uma espécie de canal de comunicação exclusiva entre a origem e o destino. Desta forma, a rede não fica "presa" a um único computador no envio de informações. Isso aumenta o desempenho da rede, já que a comunicação está sempre disponível, exceto quando dois ou mais computadores tentam enviar dados simultaneamente à mesma máquina. Essa característica também diminui a ocorrência de erros (colisões de pacotes, por exemplo);

c) **Roteador** (ou *router*): um equipamento utilizado em redes de maior porte, mais "inteligente" que o switch, pois além de poder fazer a mesma função deste, também tem a capacidade de escolher a melhor rota que um determinado pacote de dados deve seguir para chegar a seu destino. É como se a rede fosse uma cidade grande e o roteador escolhesse os caminhos mais curtos e menos congestionados.

Um elemento importante também na composição da estrutura de um cluster é o nobreak, um equipamento que armazena energia para provê-la a outro como uma fonte de alimentação ininterrupta. É bastante comum conectar computador a nobreak para que, em caso de queda de energia, seja possível desligar a máquina manualmente, diminuindo os riscos de dano.

A montagem do cluster pode passar por algumas complicações, denominadas de gargalos (a geração de ociosidade de uma ou mais partes de um sistema), como, por exemplo, a largura de banda, a vazão, a eficiência da rede, a latência, a sobrecarga da CPU e multithreading (significado explorado abaixo), que são os principais itens para avaliação de um projeto em rede de alta performance para clusters, e que devem ser tratados com muito cuidado, para que a rede de interconexão do cluster atenda perfeitamente os requisitos de alto desempenho. Uma breve descrição sobre cada um deles é apresentada a seguir:

a) **Largura de Banda**: é a capacidade de transporte de dados em uma rede, normalmente medida em bits por segundo (bps); em um cluster, quanto maior a largura de banda melhor deve ser a performance alcançada. Com isso, é possível enviar uma quantidade de dados maior entre os nós que compõem o cluster. Uma comparação prática é de uma autoestrada com 4 pistas, paralela a uma pista de mão única para trafegar a mesma quantidade de carros, todos desenvolvendo a mesma velocidade, onde, obviamente, tem-se um tempo máximo para um determinado percurso 4 vezes maior para os veículos na pista simples;

b) **Vazão**: é o termo utilizado para definir a quantidade de pacotes de dados livres de erros que trafegam por uma rede em unidade de tempo (Mbits/s, MB/s, GB/s). Outro tópico relacionado é a precisão com que os dados são transmitidos entre os nós. Um exemplo prático de vazão é a quantidade de água escoada por uma tubulação, normalmente medida em litros por segundo (l/s);

c) **Eficiência da rede**: é a medida da quantidade de esforço utilizado para que os dados fluam com uma boa vazão de dados. A eficiência depende muito da tecnologia da camada de enlace utilizada, cuja camada trata de algoritmos que permitem uma comunicação eficiente e confiável entre os computadores adjacentes. O nível de granularidade dos dados, ou seja, o nível de detalhamento dos dados, deve estar relacionado diretamente à topologia lógica escolhida para a configuração da rede, a fim de equilibrar o seu funcionamento e garantir o alto desempenho. Para exemplificar compara-se essa eficiência ao esforço que ocorre na condução de um líquido relacionando sua densidade à viscosidade da tubulação que o conduz e à potência de uma bomba para impulsioná-lo por um longo trajeto;

d) **Latência**: é outro fator de grande importância na comunicação entre os nós dentro do cluster, definida como o intervalo de tempo que um nó transmite seu pacote de dados pela rede até o momento da recepção pela outra extremidade (ns, μs, ms). Fatores como o meio físico empregado na transmissão, comutação dos pacotes e protocolos de acesso ao meio tendem a aumentar o atraso na comunicação. A latência depende muito da velocidade dos circuitos internos, da CPU, do tipo de switch, da pilha de protocolos e até mesmo do comportamento particular de cada Sistema Operacional na administração desses pacotes de dados durante a comunicação. E com isso, a latência gera uma influência negativa em outro termo muito utilizado em redes de alto desempenho, o tempo de resposta. Em um exemplo claro, supondo duas pessoas conversando por meio da Internet, à medida que o atraso aumenta, as conversas tendem a se entrelaçar, ou seja, uma pessoa não sabe se a outra a ouviu e continua falando. Após alguns milisegundos vem a resposta do interlocutor sobre a primeira pergunta efetuada, misturando as vozes. Num atraso muito grande, as pessoas devem começar a conversar utilizando códigos, por exemplo “câmbio”, quando terminam de falar e passam a palavra ao outro;

e) **Sobrecarga da CPU**: Se todo o processamento de uma rede está acontecendo na CPU de um host (hospedeiro), então esta carga competirá com o conjunto de

instruções dentro da própria aplicação. Muitas placas inteligentes de rede podem controlar vários pedaços de processamento de rede, o que alivia o trabalho da CPU. Para uma rede altamente ocupada, como é o caso de uma rede para cluster, isso pode ser um critério crítico na busca de performance. Por isso, algumas placas de rede possuem um processador dedicado na sua estrutura, de tal forma que diminua a sobrecarga da CPU e cuide da passagem de mensagem entre os nós do cluster. Outra significante otimização da sobrecarga da CPU é a utilização de "protocolos leves" (light-weight protocols) que são projetados para transportar mensagens relativamente pequenas por uma topologia de rede controlada, como no caso dos clusters. Tomando como exemplo, o acúmulo de atividades de motorista de ônibus e cobrador a uma única pessoa demonstra-se uma situação semelhante de sobrecarga de atribuições que compromete a eficiência da tarefa a ser realizada. Por outro lado, ainda usando essa mesma situação do exemplo, pode-se ter um grande ganho adotando-se catracas eletrônicas que permitiriam um melhor desempenho de todas as atribuições;

f) **Multithreading**: faz referência à capacidade de uma rede transmitir múltiplas mensagens concorrentemente e também na habilidade de um nó usar os dispositivos da rede sobre diferentes contextos. Certas placas de rede e seus drivers (rotina utilizada para interface e gerenciamento de um dispositivo de entrada ou saída) correspondentes são capazes de se comunicar com múltiplos processos de alto nível para troca de mensagens, considerando que algumas placas de rede ficam presas a um processo até que o mesmo tenha completado sua comunicação e a libere posteriormente. Enquanto a chamada sincronizada é executada em uma thread (linha de instrução), outras partes do programa que não dependem dessa chamada são executadas em threads diferentes. A execução da aplicação progride ao invés de ficar estagnada, até que a chamada sincronizada esteja completa. Dessa maneira, uma aplicação multithread maximiza a eficiência da CPU ou da rede porque ela não fica ociosa se qualquer thread da aplicação estiver pronta para ser executada.

3.1 APLICAÇÕES

Clusters, ou combinações de clusters, são usados quando os conteúdos são críticos ou quando os serviços têm de estar disponíveis e/ou processados o mais rápido possível. Internet *Service Providers* (Provedores de Internet), ou sites de comércio eletrônico, frequentemente requerem alta disponibilidade e balanceamento de carga de forma escalável. Os clusters paralelos têm uma importante participação na indústria cinematográfica para renderização de gráficos de altíssima qualidade e animações, relembrando que uma das versões do filme Titanic foi renderizado dentro desta plataforma nos laboratórios da Digital Domain. Os clusters Beowulf, que serão explorados posteriormente, por exemplo, são usados nas mais variadas áreas acadêmicas e financeiras para atuarem em projetos de desdobramento de proteínas, dinâmica de fluídos, redes neurais, análise genética, estatística, economia, astrofísica dentre outras (PITANGA, 2003). Além dessas, algumas outras áreas onde os *clusters* podem ser implementados e serão muito úteis são: computação gráfica, aerodinâmica, análise de elementos finitos, inteligência artificial e automação, exploração sísmica, oceanografia, previsão do tempo, pesquisas militares, problemas e pesquisa básica (física, química, engenharia e matemática), segurança de reatores nucleares, etc. (PITANGA, 2008, *apud* PRADO, C. L.; SILVA, J. M., 2010).

Além disso, pesquisadores, organizações e empresas estão utilizando os clusters porque precisam incrementar sua escalabilidade, gerenciamento de recursos, disponibilidade ou processamento em nível supercomputacional a um preço acessível (PITANGA, 2003).

As aplicações para os clusters são muito diversificadas, e envolvem todas as áreas, seja de pesquisa, indústria ou ensino. Em qualquer local que aparecer um grande problema computacional em que o processamento paralelo seja indicado, o *cluster* será recomendado para o serviço (PITANGA, 2008, *apud* PRADO, C. L.; SILVA, J. M., 2010). Por outro lado, como observado por Dantas (2005, *apud* PRADO, C. L.; SILVA, J. M., 2010), a implementação de *clusters* deve ser estabelecida a partir das necessidades das empresas e instituições.

Para Pitanga (2008, *apud* PRADO, C. L.; SILVA, J. M., 2010), empresas de menor porte e universidades, com poucos recursos financeiros, podem recorrer a uma alternativa para obtenção do processamento de alto desempenho, a custos baixos, aproveitando os equipamentos (*hardware*) existentes. Isto se deve ao barateamento das redes locais e da evolução dos processadores.

3.2 BENEFÍCIOS DE CLUSTERS

São inúmeros os benefícios que um aglomerado de computadores pode trazer para uma organização ou instituição. Porém, antes do desenvolvimento do cluster, a finalidade do mesmo já deve estar previamente definida, pois devem ser desenvolvidos de acordo com um modelo específico e considerando os processos e aplicações a que serão submetidos (DANTAS, 2005, *apud* PRADO, C. L.; SILVA, J. M., 2010).

Os benefícios mais importantes que os *clusters* podem proporcionar são:

a) baixo custo: a redução de custo para se obter processamento de alto desempenho utilizando-se simples PCs (computadores pessoais), cujo hardware é de fácil disponibilidade;
b) disponibilidade: devido ao cluster possuir no mínimo dois nós, caso ocorra qualquer problema, a disponibilidade dos serviços não deve ser prejudicada e deve ser garantida o maior tempo possível onde há uma grande dependência dos computadores;
c) escalabilidade: a configuração pode crescer à medida que mais recursos estiverem disponíveis em função da facilidade de adicionar novos nós para melhoria da performance, à medida que se cresce a carga de trabalho;
d) tolerância a falhas: o aumento de confiabilidade do sistema como um todo, caso alguma parte falhe, de forma que o cluster não fique prejudicado, sendo que os demais nós disponíveis redistribuem entre si a fração de processamento não realizada pelo nó que falhou;
e) alto desempenho: possibilidade de se resolver problemas muito complexos via processamento paralelo, oferecendo menor tempo de resolução;
f) balanceamento de carga: distribuição equilibrada do processamento em todos os nós que compõem o cluster;
g) independência de fornecedores: utilização de hardware aberto, software livre e independência de fabricantes e licenças de uso.

Dentre os seus benefícios, dois destacam-se: o aumento de performance, onde o processamento é distribuído entre os computadores para que cada um realize apenas uma parte, fazendo com que o sistema funcione como um único computador de maior porte; o aumento da disponibilidade, onde o mau funcionamento de um deles não representaria

problema, pois outro assumiria o lugar sem afetar o sistema, tornando-o assim estável e confiável (VOGELS *et al*., 1998, *apud* PEREIRA, 2010).

O processamento de alto desempenho pelo cluster é o benefício que permite o desenvolvimento de pesquisas, quando estas exigem cálculos complexos, nas instituições e centros de ensino e pesquisa. Nota-se com essas características que a utilização de clusters é uma opção muito flexível e vantajosa financeiramente, pois o número de nós é diretamente proporcional ao poder de processamento requerido. Outros fatores que corroboram com esse pensamento são a utilização de hardware aberto e software livre.

3.3 MODELOS DE CLUSTER

Um cluster pode ser configurado em diferentes modelos, tanto na montagem do hardware quanto na configuração do sistema. De acordo com Pitanga (2003), os modelos mais comuns de Clusters Linux (Sistema Operacional) são:

a) alta disponibilidade *(High Availability (HA) and Failover*): estes modelos de clusters são construídos para prover uma disponibilidade de serviços e recursos de forma ininterrupta mediante uso da redundância implícita ao sistema. A ideia geral é que se um nó do cluster vier a falhar (*failover*), aplicações ou serviços possam estar disponíveis em outro nó. Estes tipos de cluster são utilizados para base de dados de missões críticas, correio, servidores de arquivos e aplicações;
b) balanceamento de carga (*Load Balancing*): este modelo distribui o tráfego entrante ou requisições de recursos provenientes dos nós que executam os mesmos programas entre as máquinas que compõem o cluster por intermédio de algoritmo próprio e escalonador. Todos os nós estão responsáveis por controlar os pedidos. Se um nó falhar, as requisições são redistribuídas entre os nós disponíveis no momento. Este tipo de solução é normalmente utilizado em fazendas de servidores de web (*web farms*);
c) combinação *HA* & *Load Balancing*: como o próprio nome diz, combina as características dos dois tipos de cluster, aumentando assim a disponibilidade e escalabilidade de serviços e recursos. Este tipo de configuração de cluster é bastante utilizado em servidores de *web*, *mail*, *news* ou FTP;
d) processamento distribuído ou processamento paralelo: este modelo de cluster aumenta a disponibilidade e performance para as aplicações, particularmente as

grandes tarefas computacionais. Uma grande tarefa computacional pode ser dividida, por meio de algoritmos e bibliotecas específicas para troca de mensagens, em pequenas tarefas que são distribuídas ao redor das estações (nós), como se fosse um supercomputador massivamente paralelo. É comum associar este tipo de cluster ao projeto *Beowulf* da NASA. Estes clusters são usados para computação científica ou análises financeiras, tarefas típicas que exigem alto poder de processamento.

3.4 TIPOS DE COMPOSIÇÃO DE CLUSTER

De acordo com as ferramentas e técnicas adotadas para a composição de um cluster de computadores, eles podem ser classificados, resumidamente, como: computação em grade, Mosix, Beowulf e OpenMosix. A seguir será dada uma breve explanação sobre cada um destes modelos.

3.4.1 Computação em grade

Segundo (DIAS, 2006), as grades são construídas como um agrupamento de serviços básicos independentes, chamados de *Grid Services*, disponíveis uniformemente graças aos ambientes distribuídos na grade. Esses serviços básicos, tais como a alocação de recursos, o gerenciamento de processos, a comunicação, a autenticação e a segurança, são agrupados em um sistema integrado, chamado de *middleware*. Considerando que a computação em grade permite o uso de recursos heterogêneos, o papel principal dos *middlewares* de grade é operar sobre os diversos sistemas operacionais, arquiteturas e redes de comunicação, abstraindo seus detalhes e facilitando o desenvolvimento de aplicações para a grade. Essa aplicação terá o trabalho de fazer a interface entre o sistema operacional e as funcionalidades da grade.

Portanto, pode-se dizer que o cluster é um caso específico de grade. Isso porque um cluster é caracterizado por ser um sistema distribuído e paralelo que permite o compartilhamento, seleção e agregação de recursos de forma estática, ou seja, para fazer qualquer tipo de alteração em um cluster será necessário realizar o trabalho de configuração do recurso e do cluster para que a alteração seja considerada.

Além disso, um cluster precisa que todos os recursos trabalhem de forma dedicada, e este conjunto é utilizado em todo o seu potencial para formar um supercomputador. Por sua

vez, uma grade possui maior flexibilidade, os recursos são utilizados de acordo com sua disponibilidade e poder de processamento, ou seja, dentro de uma grade a capacidade, desempenho e disponibilidade de um recurso são considerados e cada recurso é selecionado para realizar um processamento de acordo com suas características, o que não acontece com um cluster.

A ideia de se obter um supercomputador virtual permanece no conceito de grade, porém, quando alterações são realizadas pode-se esperar que estas modificações sejam reconhecidas e consideradas, de forma que, quando se retira um recurso da grade, automaticamente ela deixará de contar com o mesmo e passará a dividir o trabalho entre os recursos restantes.

3.4.2 MOSIX

MOSIX é a sigla para *Multicomputer Operating System for UnIX*. Trata-se de um conjunto de ferramentas de cluster para Linux, voltado ao tipo Balanceamento de Carga. Uma de suas principais características é a não necessidade de aplicações e recursos de software voltados ao cluster, como acontece com o *Beowulf*.

Ele surgiu de um projeto de pesquisa sobre a migração de processos iniciado em 1977. Um protótipo de sistema operacional, com base em Unix da Bell Lab 6 foi desenvolvido e os testes foram realizados em um PDP-11/45 e um PDP-11/10 sem disco, que foram ligados em I/O paralela, uma interface de comunicação paralela entre um computador e um periférico. Na comunicação em paralelo, grupos de bits são transferidos simultaneamente (em geral, byte a byte) por meio de diversas linhas condutoras dos sinais. Desta forma, como vários bits são transmitidos simultaneamente a cada ciclo, a taxa de transferência de dados (throughput) é alta. Este estudo demonstrou que, mesmo com links de comunicação lentos, é possível obter melhorias substanciais de desempenho mediante a migração de processos em execução em nós remotos.

A partir daí surgiram as evoluções do Mosix, conforme listado no quadro 1:

Quadro 1: História do Mosix

Ano	Versão	Nome	Descrição da configuração do cluster com Mosix
1977–1979	Versão 0	UNIX with Satellite	• Configuração: PDP-11/45 + PDP-11/10 • Compatível com: Unix 6 da Bell Lab

		Processors	
1981–1983	Versão 1	MOS	• Configuração: PDP-11/45 + 4 PDP-11/23 • Compatível com: Unix 7 da Bell Lab
1983–1984	Versão 2	MOS	• Configuração: 8 computadores pessoais MC68K • Compatível com: Unix 7 da Bell Lab com algumas extensões BSD 4.1
1987–1988	Versão 3	NSMOS	• Configuração: vários computadores baseados em NS32332 • Compatível com: *AT & T Unix System V* Release 2
1988	Versão 4	MOSIX-1	• Configuração: Cluster de VAX-780 e 4 VAX-750 ligados por *Ethernet* • Compatível com: *AT & T Unix System V Release 2*
1988–1989	Versão 5	MOSIX	• Configuração: Cluster com 16 computadores NS32532 conectados por *VMEbus* e *Pronet* • Compatível com: *AT & T Unix System V Release 2*
1991–1993	Versão 6	MOSIX	• Configuração: Cluster com 8 486 SBC conectados por *Multibus* II + *32 PCs Pentium* ligados por *Myrinet* • Compatível com: BSD / OS
1998–1999	Versão 7	MOSIX	• Configuração: Cluster com 64 PCs x86 ligados por Myrinet • Compatível com: Linux 2.2
2000–2001	Versão 8	MOSIX	• Configuração: Cluster com 96 PCs x86 ligados por *Myrinet* • Compatível com: Linux 2.4
2003	Versão 9	MOSIX	• Configuração: Cluster com 100 estações de trabalho e servidores x86 ligados por Ethernet de 1 Gb • Compatível com: Linux 2.4

2004–2006	Versão 10	MOSIX-2	• Configuração: *Cloud* com 100 estações de trabalho e servidores x86 ligados por Ethernet de 1 Gb • Compatível com: Linux-2.6
2008–2009	MOSIX (MOSRC)	MOSRC	• Configuração: Nuvem de multi-cluster com várias estações de trabalho e servidores i386 e x86_64 ligados por 1 Gb Ethernet e QDR *InfiniBand* • Compatível com: Linux-2.6
2011–2012	Versão 11	MOSIX-3	• Configuração: Nuvem de *Multi-cluster* com estações de trabalho e servidores x86_64 conectados por 1 Gb *Ethernet* e *QDR InfiniBand* • Compatível com: Linux-3

Fonte: (HISTORY ..., 2013)

Alguns termos referenciados no Quadro 1 são descritos a seguir:

- **INFINIBAND** – Barramento serial que oferece 2.5 Gigabits por segundo (312 MB/s) por par de cabos, onde um envia e outro recebe dados;
- **MULTIBUS** – Barramento de computador usado em sistemas industriais;
- **VAX** – Arquitetura de computadores de 32 bits que suporta um conjunto de instruções ortogonais (código de máquina) e endereçamento virtual (ou seja, exige memória virtual paginada);
- **VMEbus** – Barramento de computador padrão inicialmente desenvolvido para uma linha de CPU do Motorola 68000;
- **PDP-11 –** Foi uma série de minicomputadores de 16 bits vendida pela Digital Equipment Corporation (DEC) de 1970 até a década de 1990, um de uma sucessão de produtos na série PDP. O PDP-11 substituiu o PDP-8 em muitas aplicações em tempo real, embora ambas as linhas de produtos fossem vendidas em paralelo por mais de 10 anos. O PDP-11 tinha várias características excepcionalmente inovadoras, e era mais fácil de programar do que seus antecessores, com o uso de registros gerais. Seu sucessor no nicho minicomputador de gama média foi a de 32 bits VAX-11 (WIKIPEDIA, 2013).

Conforme publicado em (MOSIX, 2013a), MOSIX é um sistema de gerenciamento de cluster, como um sistema operacional de cluster, alvo de computação de alto desempenho (HPC) em clusters Linux x86, multi-clusters e Nuvens. Quando combinado com a VCL pode também ser utilizado para gerir clusters GPU. Suporta tanto processos interativos concorrentes, onde dois ou mais processos interagem entre si para resolverem uma determinada tarefa, quanto processos em lote, onde um processamento de dados ocorre graças a um lote de tarefas enfileiradas, de modo que o sistema operacional só processa a próxima tarefa após o término completo da tarefa anterior. Ele fornece uma imagem de sistema único, incorporando a descoberta de recursos automáticos e distribuição de carga dinâmica por migração de processo preemptivo (o sistema pode interromper um processo em execução para que outro processo o utilize).

O MOSIX é implementado como uma camada de software, uma camada inferior que contém componentes como sistemas operacionais, bancos de dados, interfaces para hardware específico, etc. Isso permite que os aplicativos sejam executados em nós remotos como se eles executassem localmente. Os usuários podem iniciar aplicações (sequencial e paralela) em um nó, enquanto o MOSIX automaticamente busca recursos e executa-os em outros nós de forma transparente. Não é necessário modificar os aplicativos, copiar arquivos, fazer o login ou atribuir processos para nós remotos – tudo é feito automaticamente. Alocação de aplicações para nós são supervisionadas por um conjunto abrangente de algoritmos on-line que monitoram o estado dos recursos e tentam melhorar o desempenho global por alocação dinâmica de recursos como, por exemplo, balanceamento de carga.

Uma característica única do MOSIX é que ele opera sobre o nível do processo, ao contrário dos sistemas que operam no nível de tarefa. A diferença entre processo e tarefa é que processo é um software que executa alguma ação e que pode ser controlado de alguma maneira, pelo usuário, pelo aplicativo correspondente ou pelo sistema operacional, enquanto que a tarefa em execução é representada por um processo. Além disso, nem toda tarefa tem uma relação direta com algum aplicativo, sendo que muitas são executadas em segundo plano mantendo o sistema operacional funcionando. Isto significa que se adapta e redistribui a carga de trabalho quando o número de processos de uma tarefa (e/ou suas demandas) muda. Isto é especialmente útil para trabalhos paralelos. Como exemplos de tarefas, pode-se citar a leitura de sensores, tratamento de protocolos de comunicação, leitura de teclado, acionamento de dispositivos luminosos, escrita em telas de saída de dados (monitores), comunicações seriais entre equipamentos, gravação de informações em disco, etc. Por outro lado, exemplos de

processos podem ser a impressão de um documento, a cópia de um arquivo, a compilação de um programa ou a execução de uma rotina qualquer manual ou automática.

A última versão do MOSIX pode gerenciar clusters, multi-clusters e nuvens. A gestão flexível permite aos proprietários de diferentes grupos compartilharem os seus recursos computacionais, preservando a autonomia para desligar seus clusters, a qualquer momento, sem interromper programas já em execução. Um multi-cluster MOSIX pode se estender indefinidamente, enquanto houver confiança e domínio da estrutura entre os proprietários ou administradores de seus clusters.

Esse sistema pode funcionar em ambientes não virtualizados ou em máquina virtual. A máquina virtual é um software totalmente isolado que pode executar os próprios sistemas operacionais e aplicativos como se fosse um computador físico, podendo ser executada nos sistemas operacionais Linux ou Windows. Ela se comporta exatamente como um computador físico e também contém CPU, memória RAM, disco rígido e NIC (Network Interface Card, placa de interface de rede) virtuais (isto é, com base em software). Um ambiente não virtualizado exige o *patch* (programa criado para atualizar ou corrigir um software) do kernel do Linux e oferece melhor desempenho. Além disso, ele é mais adequado para a execução de aplicativos HPC com baixa a moderada quantidade de I/O. É particularmente adequado para utilização eficiente de recursos de todo o conjunto, execução de aplicações com requisitos de recursos imprevisíveis ou tempos de execução, executar (e preservar) processos longos e combinar nós de diferentes velocidades.

O MOSIX é também eficiente na tarefa de distribuição dinâmica de processamento entre os computadores do cluster. Esse tipo, assim como o *Beowulf*, é muito utilizado por universidades em projetos de pesquisas. Por basear-se em Linux, sua implementação é transparente, além de ser relativamente “fácil” de instalar.

De maneira generalizada, o MOSIX é uma extensão para Linux (ou sistemas baseados em Unix) de um sistema de cluster que trabalha como se fosse um único supercomputador, por meio de conceitos de distribuição de processos e balanceamento de carga (ALECRIM, 2004). Essa ferramenta faz balanceamento de carga mediante a utilização de migração preemptiva de processos e balanceamento dinâmico de carga.

A migração preemptiva de processos consiste em transportar um processo de um nó escasso de recursos para um que tenha maior disponibilidade. A migração de processos, que o MOSIX suporta para balancear carga, cria novas potencialidades de escalabilidade de processos paralelos de I/O (entrada e saída) que sejam apropriadas para as aplicações que necessitam processar grandes volumes de dados. O processamento paralelo de MOSIX utiliza

biblioteca de I/O "MOPI", que fornece meios para dividir os processos (de forma transparente) para diferentes nós e permite acesso paralelo aos diferentes segmentos de um arquivo (BARAK, 2002 *apud* PASINI, 2009). Quando um processo é iniciado, o Mosix escolhe qual o melhor nó que poderá executá-lo, o envia para tal e monitora sua execução. Assim, o usuário não vê nada, o cluster faz tudo de forma transparente.

Outra característica que pode ser destacada é que o Mosix pode ser implementado em máquinas onde serão usadas outras tarefas (desktops, por exemplo), pois o mesmo é totalmente imperceptível ao usuário, e em redes heterogêneas, com todos os tipos de computadores, inclusive com os que possuem processadores de 64bits, ou mesmo os novos processadores multi-core (vários núcleos de processamento). Entre as características do Mosix destacam-se:

a) usuários podem fazer login em qualquer nó do cluster sem a necessidade de saber da existência do Mosix;
b) não existe nó controlador;
c) não é necessário modificar ou apontar os aplicativos às bibliotecas especiais;
d) pode-se incluir ou remover algum nó a qualquer momento sem afetar o funcionamento do sistema;
e) não é necessário copiar arquivos para nós remotos do cluster;
f) promove balanceamento de carga usando migração de processos;
g) migração de processos dos nós escassos de recursos para os mais rápidos;
h) possui algoritmo de monitoramento que identifica, entre outras coisas, a velocidade de cada nó, a carga da CPU e a memória livre disponível;
i) tem suporte para trabalhos em *batch*, ou seja, um arquivo de computador contendo um conjunto de comandos para serem executados sequencialmente, ou em lote, para automatizar tarefas;
j) possui ferramentas de instalação e configuração automáticas.

Algumas das aplicações que se beneficiam do Mosix são:

a) processos *CPU-bound*: processos com longos tempos de execução e baixo volume de comunicação entre processos, utilizando muito processador; exemplo: aplicações científicas e outras aplicações que demandem alto desempenho de computação;
b) grandes compilações;

c) processos que misturam longos e rápidos tempos de execução ou com moderadas quantias de comunicação interprocessos, somado ao uso das bibliotecas MPI/PVM, que serão discutidas com mais detalhes na Seção 3.5;
d) processos I/O *bound* misturados com processos da CPU quando executados por meio do servidor de arquivos, usando o sistema de arquivos distribuídos, utilizando muita leitura e gravação e acesso a disco;
e) banco de dados que não usem memória compartilhada;
f) processos que podem ser migrados manualmente.

Algumas desvantagens do *Mosix*, segundo (PASINI, 2009), são:

a) processos com baixa performance, quando aplicativos com alta comunicação interprocessos são executados;
b) aplicações dependentes do hardware, que necessitam de acesso a um periférico de um nó em especial, não podem ser distribuídas;
c) aplicações com muitos *threads* (explorado abaixo) não ganham desempenho;
d) não se ganha desempenho quando se executa um único processo, tal como o browser (aplicativo para navegar na internet).

Threads são múltiplos caminhos de execução que são executados concorrentemente na memória compartilhada e que compartilham os mesmos recursos e sinais do processo pai ou ainda, pode ser dito que é um processo simplificado, mais leve ou "light", que custa pouco para o sistema operacional, sendo fácil de criar, manter e gerenciar (BUENO, 2012).

Mais informações sobre o Mosix estão disponíveis no site www.mosix.org e, além disso, Carvalho (2007) adotou alguns modelos de análise de comportamento de cluster que podem ser seguidos.

3.4.3 Cluster Beowulf

O nome Beowulf vem de um herói muito valente que tinha a missão de derrotar um monstro, num texto inglês antigo. Esse tipo de cluster, voltado à computação paralela, foi fundamentado em 1994, pela NASA, com a finalidade de processar as informações espaciais que a entidade recolhia. Desde então, grandes empresas (como HP e IBM) e universidades (como a brasileira UNESP) vêm construindo clusters deste tipo e com cada vez mais nós. O

que distingue o Cluster Beowulf dos outros tipos são as seguintes características (que são aplicadas de acordo com a finalidade do cluster):

a) a conexão dos nós pode ser feita por redes do tipo Ethernet (mais comum);
b) existe um servidor responsável por controlar todo o cluster, principalmente quanto à distribuição de tarefas e processamento (pode haver mais de um servidor, dedicado a tarefas específicas, como monitoração de falhas). Este servidor é chamado de *Front-end*;
c) o sistema operacional é baseado em Linux, sendo necessário que ele contenha todos os programas para cluster;
d) não é necessário usar equipamentos próprios para clusters. Pode-se usar computadores comuns e tradicionais, inclusive modelos considerados obsoletos.

De maneira generalizada, o Cluster Beowulf permite a construção de sistemas de processamento que podem alcançar altos valores de *gigaflops* (um *gigaflop* equivale a 1 bilhão de instruções de ponto flutuante executadas por segundo). Isso tudo com o uso de computadores comuns e de um sistema operacional com código-fonte livre, ou seja, além de gratuito, pode ser melhorado para a sua finalidade. Tais características fizeram do Cluster Beowulf um tema muito explorado em universidades e, claro, aplicado para vários fins.

Entre os requisitos para o sistema operacional de um Cluster Beowulf, estão a necessidade de se ter as bibliotecas *Parallel Virtual Machine* (PVM) ou *Message Passing Interface* (MPI). Ambos os tipos são usados para a troca de mensagens entre os nós do cluster. Estas bibliotecas serão exploradas com mais detalhes a seguir.

Para mais informações sobre o Cluster Beowulf, visite www.beowulf.org.

3.4.4 OpenMosix

O OPENMOSIX é uma extensão do projeto Mosix, anteriormente explorado, baseado no GPLv2, iniciado em 10 de fevereiro de 2002, e coordenado pelo Ph.D Moshe Bar, para manter os privilégios da solução Linux para cluster, sendo disponível com o software de código aberto (OPENMOSIX, 2008, *apud* ALUVINO, 2008).

O OPENMOSIX é uma extensão do kernel do Linux, que faz com que um cluster de computadores funcione como um grande e único supercomputador via balanceamento dinâmico de carga e da migração de processos de forma preemptiva, ou seja, um esquema de processamento computacional onde o kernel tem o controle do tempo que será usado por cada processo segundo sua relação de prioridades.

A implementação da migração preemptiva de processos permite que os processos do usuário possam migrar a qualquer tempo e para qualquer nó de forma transparente, mediante algoritmos de balanceamento de carga e de prevenção contra falta de memória.

Se um programa que está sendo executado consumir muitos recursos da máquina, o sistema faz uma busca na rede para procurar outra máquina mais disponível no momento, em termos de CPU e memória, deslocando o programa ou parte dele, para ser executado remotamente, aumentando o desempenho do sistema.

Cada nó é um mestre para os processos que são criados localmente e um servidor para processos remotos, migrados de outros nós do cluster. Isto significa que podemos acrescentar ou remover máquinas a qualquer instante, com um mínimo transtorno no sistema como um todo. Todos os nós desta configuração possuem algoritmo de monitoramento de carga da CPU, velocidade de cada nó, a comunicação entre processos, memória disponível e a velocidade de acesso de cada processo. Como no OpenMosix as operações são transparentes para as aplicações, podem-se executar aplicações sequenciais ou paralelas como se fosse um único computador SMP (Multiprocessamento Simétrico).

Não existe a necessidade de se saber onde os processos estão sendo executados, pois os algoritmos de balanceamento se encarregam de tal missão. Logo depois de iniciados os processos, eles são enviados para um melhor nó na rede e esses algoritmos continuam a monitorar os novos e os demais processos movimentando-os pelos computadores com menor carga de trabalho, otimizando os processos e melhorando a performance do sistema.

Porém, o projeto Open Mosix anunciou seu fim em 01 de março de 2008 (OPENMOSIX, 2008, *apud* ALUVINO, 2008). Mesmo assim, todas as funcionalidades existentes continuam disponíveis, apenas o desenvolvimento de novas funcionalidades é que não serão implementadas por este projeto. Uma alternativa a este projeto é o uso do MOSIX2 que continua em pleno desenvolvimento e que oferece uma cópia gratuita de avaliação, limitada a 6 nós, para arquitetura x86, sem data de expiração, para uso sem fins lucrativos.

Por fim, como informação adicional, a próxima seção trata de algumas características sobre bibliotecas de paralelização para otimizar o fenômeno em estudo e que são importantes para certos tipos de composição de cluster como, por exemplo, o *Beowulf*.

3.5 PARALELIZAÇÃO

Uma característica comum aos problemas de otimização é a dificuldade de encontrar uma solução ótima. O grande número de variáveis e restrições torna tais problemas muito complexos, exigindo uma grande quantidade de recursos computacionais para resolver até mesmo os de pequeno ou médio porte. Como nem toda empresa tem os recursos necessários para financiar um ambiente computacional compatível com as necessidades impostas pelos problemas referidos, há que se tirar o máximo proveito dos recursos existentes.

A norma hoje em dia nas empresas é possuir uma rede de comunicação formada por pontos (computadores) de baixo e médio custo que permitam compartilhar os recursos computacionais como CPUs (Unidade de Processamento Central) e impressoras, ou seja, uma rede local. Essa topologia substituiu a organização centralizada dos recursos computacionais em um mainframe (computador de grande porte usado no processamento de dados em grande escala) que possuía um alto poder de processamento e uma boa taxa de E/S (entrada e saída). Esses dois quesitos foram sacrificados nessa mudança e a busca por melhor desempenho, maior confiabilidade e maior modularidade impulsionou o desenvolvimento de arquiteturas distribuídas e paralelas (FRANÇA, [20--]).

O processamento paralelo consiste em dividir uma tarefa em partes independentes e a execução de cada uma destas partes em diferentes processadores. Para isso são necessárias algumas bibliotecas instaladas e configuradas para o funcionamento do paralelismo. As bibliotecas essenciais para a programação paralela devem ser referenciadas nos códigos fontes dos programas que suportam essa funcionalidade. São citadas a seguir algumas das mais conhecidas dessas bibliotecas.

3.5.1 MPI

De acordo com Bueno (2002), MPI (*Message Passing Interface*) é um método que inclui conceitos novos como *rank* (cada processo tem uma identificação única, crescente), *group* (conjunto ordenado de processos) e *communicator* (uma coleção de grupos), que permitem um gerenciamento mais complexo (e inteligente) do uso de cada máquina do cluster.

O MPI tem opções mais avançadas que o PVM, explorado na seção 3.5.3, como o envio de mensagens *broadcast* (para todas as máquinas do cluster) e *multicast* (para um grupo

específico de máquinas), assim como um melhor controle sobre o tratamento que cada mensagem terá ao ser recebida por outro ponto do cluster. A configuração do MPI depende da implementação utilizada e algumas delas chegam a instalar *front-ends* para compiladores em C e Fortran, mas a forma geral de uso é semelhante. Abaixo segue uma breve descrição de algumas características do MPI.

Requisitos: Requer o conhecimento de um sistema bastante complexo de troca de mensagens, além de ser um método explícito;

Vantagens: É o novo padrão para processamento distribuído, embora ainda seja menos utilizado que o PVM;

Desvantagens: Na prática significa aprender uma nova linguagem de programação. É um padrão da indústria com várias implementações individuais. O seu aprendizado não é trivial.

3.5.2 MPICH

A biblioteca de programação paralela MPICH foi desenvolvida pelo *Argonne National Laboratory* em conjunto com *Missisipi State University* e implementa todas as funcionalidades especificadas no padrão MPI. Um dos objetivos da MPICH foi criar uma biblioteca eficiente que, ao mesmo tempo, pudesse ser facilmente portada para outras plataformas. O "CH" em MPICH vem de "Chameleon", símbolo desta adaptabilidade.

O desenvolvimento da MPICH começou paralelamente com a definição da especificação MPI, como uma forma de avaliar o cumprimento das metas já estabelecidas para o padrão. A MPICH2 é a última versão desta implementação, refletindo a especificação MPI–2, versão mais recente do padrão.

Como o padrão MPI é bastante extenso, dificultando a tarefa de portá-lo diretamente para várias plataformas, uma solução viável foi definir um arcabouço que minimize o impacto ao portar uma implementação para outra plataforma. Assim a MPICH alcançou as metas de portabilidade e eficiência com a criação de uma arquitetura estruturada em camadas. Na arquitetura da MPICH todas as funções MPI são implementadas em termos de macros e funções definidas nas camadas de mais baixo nível (CARDOSO, 2008). Na Ciência da Computação, macro é uma abstração que define como um padrão de entrada deve ser substituído por um padrão de saída, de acordo com um conjunto de regras.

3.5.3 PVM

De acordo com Bueno (2002), PVM (*Parallel Virtual Machine*) é a biblioteca mais utilizada para processamento distribuído. É o padrão, de fato, da indústria de software.

O PVM se baseia em duas primitivas básicas: i) envie mensagem e ii) receba mensagem. É de fácil utilização, mas não é tão poderoso quando comparado com o MPI. O usuário deve configurar as máquinas para que sejam as mais idênticas possíveis, facilitando a manutenção e estabelecendo uma relação de confiança entre elas. Usar rhosts e rsh é a forma mais simples de conseguir isso. O rhosts é um arquivo que pode permitir que usuários específicos remotos e/ou hospedeiros executem comandos na máquina local. O rsh (Remote Shell) é um programa de computador de linha de comando que pode executar comandos shell como outro usuário e em outro computador via rede de computadores.

O usuário pode executar o gerenciador do PVM, adicionar máquinas ao cluster e depois simplesmente executar o programa feito usando as bibliotecas PVM. Como no MPI, segue uma descrição simplificada de características desta biblioteca.

Requisitos: Para o desenvolvimento dos programas é necessário conhecer a biblioteca PVM. É também um sistema explícito, ou seja, cabe ao programador dividir as tarefas por meio da troca de mensagens;

Vantagens: Possibilita o uso do processamento distribuído. É o mais utilizado. Alguns programas de Engenharia e Matemática geram código automaticamente para o PVM;

Desvantagens: Não é mais o padrão. O desenvolvimento dos programas fica bem mais complicado quando comparado com *threads*.

3.6 PROBLEMAS E DESVANTAGENS DE CLUSTERS

Conforme Bacellar (2010), os clusters de computadores, como toda tecnologia, também possuem desvantagens e cabe ao pesquisador ou projetista fazer o levantamento de prós e contras na hora de implementar um sistema desse tipo. São levantadas, a seguir, algumas das principais desvantagens:

a) manutenção de equipamentos: pelo fato do cluster ser facilmente expansível, o sistema computacional pode se tornar muito grande e a sua manutenção pode se tornar uma tarefa volumosa, pois cada máquina em um cluster deve ter todos os seus componentes em perfeito estado de funcionamento;

b) monitoração dos nós: monitorar as informações trocadas em cada nó pode ser um problema dependendo de como foi configurado o cluster, levando em consideração a expansibilidade e as ferramentas adotadas para a gestão do mesmo;
c) gargalos de troca de informações: como a comunicação no cluster de computadores ocorre por uma tecnologia de rede, a troca de informação se transforma no principal gargalo, uma vez que a transmissão de rede é mais lenta se comparada à troca de informação com um barramento de um sistema de memória compartilhada em um computador de grande porte, embora seja possível realizar ajustes de granulosidade para diminuir esse problema.

4 PROCEDIMENTOS METODOLÓGICOS

Este projeto foi concebido na Universidade Federal do Triângulo Mineiro (UFTM) a partir de pesquisa exploratória e bibliográfica, análise de documentos e de bancos de dados e testes em laboratório. Além disso, foram estudadas ferramentas, modelos e técnicas para dar suporte e propor soluções aos desafios da gestão de recursos de TI e sua otimização com a implementação de cluster computacional.

4.1 DEFINIÇÕES DO PROJETO

A definição desse projeto envolveu várias fases e estão descritas nos subtópicos seguintes.

4.1.1 Revisão bibliográfica

A primeira fase foi dedicada à pesquisa bibliográfica relacionada ao tema central desse trabalho que é otimização de recursos de Tecnologia da Informação. Após um determinado estágio das pesquisas, houve um direcionamento específico para busca de literatura sobre implementação de cluster. Com um refinamento maior tendeu-se à busca de soluções com cluster para instituições públicas, objetivando a implementação de um projeto viável para superar as dificuldades com recursos financeiros e tecnológicos como requisitos aos projetos de pesquisa que envolvam processamento em grande escala.

4.1.2 Elaboração do Projeto

Baseado nas condições já mencionadas anteriormente da UFTM, na literatura encontrada e na viabilidade do projeto em virtude da demanda de processamento em grande escala e existência de computadores ociosos, esse projeto foi formatado. Diante dessa proposta foi necessária a formalização da solicitação de utilização dos laboratórios do Instituto de Ciências Tecnológicas e Exatas, ICTE, da Universidade para execução de testes e implementação do produto desse trabalho. Essa solicitação foi concedida pelos departamentos responsáveis marcando o início do desenvolvimento.

4.1.3 Adoção de máquinas virtuais para testar cluster

Buscando a redução de custos com energia por dispensar o funcionamento de vários microcomputadores, procurando garantir a portabilidade do projeto e viabilização das reuniões com orientador e outras pessoas que deram suporte ao projeto, bem como o andamento do mesmo em locais diferentes do laboratório até o seu amadurecimento e concretização, foram utilizadas máquinas virtuais. Mediante a adoção do Virtualbox (VIRTUALBOX, [200-?]) foram criadas máquinas virtuais em um computador pessoal, com configurações reduzidas de processamento, simulando as configurações a serem definidas nos microcomputadores do laboratório onde o projeto seria implantado efetivamente na UFTM.

Foram criadas 3 máquinas virtuais e configuradas em cluster. A partir do momento que foi atingida a configuração "ideal", este projeto foi migrado para as máquinas físicas de um dos laboratórios do ICTE. Outras informações sobre o Virtualbox são listadas no Apêndice A e as instruções para a criação das máquinas virtuais são listadas no Apêndice B. As configurações dessas máquinas virtuais e do hospedeiro estão descritas no Quadro 2 a seguir:

Quadro 2: Configuração de máquinas virtuais

Hospedeiro	**Máquina Virtual**
Memória: 4GB Processador: Intel Core I3 M350 2.27GHZ (4 núcleos/CPU) Disco Rígido: 320 GB Placa de Rede: Realtek PCIe FE Family Controller Placa-mãe: Intel® 5 series Sistema Operacional: Windows 7 64 bits	Memória: 384 MB Processador: 1 Núcleo/CPU Intel Core I3 M350 2.27GHZ Disco Rígido: 5 GB Placa de Rede: Realtek PCIe FE Family Controller em modo Bridge Placa-mãe: Intel® 5 series Sistema Operacional: Linux Debian Squeeze

Fonte: Quadro elaborado pelo autor

4.1.4 Definição do sistema operacional

A busca por alternativas e ferramentas que proporcionassem a configuração de um cluster exigiu testes em diferentes versões de sistemas operacionais. Com respaldo da

literatura e da busca por soluções de menor custo, definiu-se o Linux como sistema Operacional, por ter distribuições livres.

Inicialmente, tentou-se o Linux Opensuse para adequar-se a um modelo de cluster chamado Openmosix, proposto em uma das referências bibliográficas pesquisadas. Porém, como esse modelo teve seu suporte descontinuado e alguns problemas encontrados na configuração do mesmo não puderam ser solucionados, ele foi descartado.

Outra aposta foi adotar o sistema operacional Linux, distribuição Fedora, em função de literatura que o apresentava como ideal para uma determinada configuração de cluster. Porém, esse método adotado envolvia configurações complexas que inviabilizavam a manutenção do cluster. Devido às variações de versões do Kernel do Linux, essa distribuição também não se compatibilizou com o Mosix, que foi a ferramenta adotada para a gestão do cluster.

As pesquisas e estudos continuaram até encontrar uma referência com a compatibilidade da configuração do Mosix em uma distribuição Debian Squeeze para a implementação do cluster. Embora ele possa ser instalado em quase todos os sistemas operacionais baseados em UNIX, inclusive heterogeneamente com hardwares diferentes, essa compatibilidade com o Linux é destacada na literatura por garantir melhores condições de instalação e manutenção do cluster a qualquer momento, dotado ainda de ferramentas de administração e monitoramento do mesmo.

A escolha por esta distribuição foi também por sua robustez, estabilidade e pela facilidade de trabalhar com pacotes pré-compilados nativos (.deb), de fácil manuseio e atualização do sistema. Essa distribuição foi obtida do sítio http://www.debian.org e instalada com todas as opções padrões do assistente de instalação em modo gráfico, conforme descrito no Anexo A.

Todo o estudo e pesquisa procuraram reunir sempre ferramentas e sistemas operacionais livres para atingir o ideal de propor uma solução viável e de custo mínimo à Instituição.

4.1.5 Definição de ferramenta de administração de cluster

Em função da literatura encontrada nas pesquisas realizadas e, como citado anteriormente, em função do sistema operacional compatível, a ferramenta de administração de cluster adotada foi o Mosix devido à “fácil” instalação, configuração, manutenção e

monitoramento, principalmente na distribuição Debian do Linux, além das demais vantagens já relacionadas no item 3.4.2.

4.2 INSTALAÇÃO DO CLUSTER

A instalação de um Cluster envolve vários passos para que, tanto o hardware quanto o software, sejam configurados adequadamente. Quanto ao hardware, embora se considerando que a estrutura física (*switch*, cabos de rede, *nobreak*, etc.) já estava devidamente implementada na sala para alocar o cluster, serão listados os passos básicos necessários para a instalação física:

- os computadores são conectados aos nobreaks e estes à rede de alimentação, atentando-se para a tensão de trabalho, 127V ou 220V;
- para a ligação física da rede, basta conectar uma extremidade do cabo de rede à placa ethernet e a outra à porta do switch;
- para acesso à internet basta conectar um cabo do switch a um ponto de acesso e, após configurada a rede, todas as máquinas terão acesso.

São abordados, a seguir, os aspectos de software para implementar o cluster.

4.2.1 Configuração dos equipamentos para composição do cluster

Inicialmente, foi montado um cluster em um laboratório da Diretoria de Tecnologia da Informação, DTI, da UFTM para a realização de alguns testes. Porém, por fatores de logística, ele foi montado definitivamente utilizando microcomputadores de um laboratório de informática do Instituto de Ciências Tecnológicas e Exatas, ICTE, que, durante alguns intervalos do dia, não são utilizados. É classificado como homogêneo e possui a seguinte configuração de hardware: memória RAM de 4.0 GB, processador Intel Pentium Dual Core E6300 2.8 GHz (2 núcleos), disco rígido de 80 GB e placa de rede *Fast Ethernet onboard* 10/100.

Para preservar a integridade desses microcomputadores e não comprometer os sistemas já instalados durante as etapas de testes, seus discos rígidos foram substituídos por outros usados retirados de equipamentos sucateados fornecidos pela DTI da UFTM.

4.2.2 Configuração da rede para integração dos elementos do cluster

As máquinas selecionadas para compor o cluster foram configuradas num perfil de rede para se comunicarem livremente sem bloqueios e restrições e garantir o compartilhamento de processos.

Na UFTM, a configuração de rede é particular de cada laboratório da instituição em virtude da localização dos prédios e da logística adotada para o fornecimento de conectividade. Para o laboratório do ICTE foram seguidos os passos listados no quadro 3, após a instalação do sistema operacional:

Quadro 3: Passos para configuração de rede do cluster

Comando	**Descrição**
mii-tool	Esse comando identifica a placa de rede disponível do equipamento
more /etc/udev/rules.d/70-persistent-net.rules	Esse arquivo lista a identificação da placa de rede pelo sistema operacional no hardware específico
rm /etc/udev/rules.d/70-persistent-net.rules	Caso haja conflito na identificação da placa de rede, esse comando permite excluir o arquivo e reinicializar tal identificação
reboot ou halt	Com esses comandos o sistema operacional é reinicializado ou desligado para que na próxima inicialização carregue as configurações padrões do *hardware* de rede
ifconfig eth0 down	Esse comando desabilita a placa de rede eth0 para receber as configurações de rede
vi /etc/network/interfaces	Com esse comando é editado o arquivo responsável por armazenar toda a configuração de rede do sistema operacional. Todas as informações a seguir devem constar nesse arquivo podendo mudar apenas os números de IP dependendo da rede onde o computador está instalado
Conteúdo do arquivo interfaces	*# The loopback network interface* #auto lo eth0

Observação: Essa configuração de rede foi a mesma para os quatro microcomputadores do cluster, alterando-se exclusivamente o número do IP address, sendo que os demais foram configurados na sequência com os números 192.168.11.54, 192.168.11.55 e 192.168.11.56	auto lo iface lo inet loopback *# The primary network interface* allow-hotplug eth0 iface eth0 inet static address 192.168.11.53 netmask 255.255.255.0 gateway 192.168.10.254 network 192.168.10.0 broadcast 192.168.11.255 dns-nameservers 200.225.197.34 200.225.197.37
:wq	Esse comando é interno do editor de texto e permite gravar as alterações e sair
vi /etc/resolv.conf	Esse comando edita o arquivo responsável por armazenar as informações de DNS da rede, onde o que eventualmente pode mudar é o IP da rede na qual o computador está instalado.
Conteúdo do arquivo resolv.conf	nameserver 200.225.197.34 200.225.197.37
:wq	Esse comando é interno do editor de texto e permite gravar as alterações e sair
ifconfig eth0 up	Esse comando habilita a placa de rede após receber a configuração
Ifconfig eth0	Esse comando lista a configuração de rede definida para a placa de rede
/etc/init.d/networking stop	Esse comando finaliza o serviço de rede
/etc/init.d/networking start	Esse comando inicializa o serviço de rede
ifconfig eth0 192.168.11.53 netmask 255.255.255.0	Esse comando é uma alternativa para alterar o IP da placa de rede
route add default gw 192.168.10.254 eth0	Esse comando define a rota ou gateway da rede
Configuração do nome do computador Editar o arquivo /etc/hostname vi /etc/hostname	Cada máquina teve sua configuração particular, sendo incluídos os seguintes nomes em cada máquina na ordem dos IPs

	debian1 debian2 debian3 debian4
Conversão de um IP em um nome Editar o arquivo /etc/hosts de cada nó e preencher com o conteúdo da célula ao lado vi /etc/hosts	192.168.11.53 debian1 192.168.11.54 debian2 192.168.11.55 debian3 192.168.11.56 debian4
Reinicializar os computadores para atualizar os nomes configurados no arquivo hostname	shutdown –r now
Caso os computadores já possuam os seus respectivos nomes, a rede pode ser reinicializada por meio dos comandos	cd /etc/init.d ./networking restart

Fonte: Desenvolvido pelo autor

4.2.3 Instalação e configuração do Mosix

O primeiro passo foi instalar nas máquinas virtuais uma versão do Mosix, obtida do site www.mosix.org, compatível ao Kernel do Linux instalado, conforme detalhado no Apêndice C. Após a confirmação da funcionalidade por meio da realização de testes, toda a estrutura foi refeita em máquinas físicas.

Por meio de um instalador automático "*mosix.install*", contido no diretório principal originado da descompactação do pacote de instalação, são feitos todos os procedimentos necessários para instalação e configuração do Kernel do Linux, bem como sua integração com o sistema operacional para o funcionamento do Mosix.

Mediante um utilitário de configuração em modo texto acessado com o comando *mosconf* no terminal, são exibidas as várias opções de configuração do Mosix, conforme descrição a seguir:

a) na primeira opção configura-se quais nós estão no cluster, adicionando novos nós ou removendo;
b) a segunda faz referência à segurança, onde é fornecida uma senha de acesso aos nós, para evitar o acesso indevido por outros meios que não sejam o Mosix;
c) a opção três é responsável pela numeração lógica do cluster ou mapeamento, onde é criada uma regra de numeração para a identificação de cada nó na rede;

d) no item de número quatro existe a opção de indicar um nó do cluster para gerenciar as tarefas, e mais configurações a respeito de gerenciamento de processos;
e) a quinta define as políticas de congelamento; caso algum processo tenha que ser interrompido antes do seu término, essas políticas serão responsáveis pela retomada do processo de onde ele parou;
f) a sexta opção é onde são configurados vários parâmetros, como a velocidade do processador, IP associado ao nó, *logs* para chamadas de processos do cluster e reserva de espaço em disco para uso dos processos privados do cluster;
g) a sétima opção é a configuração do nó para participar de uma grade organizacional de clusters, onde ele pode fazer parte de outros clusters em outras localidades.

Todas as configurações feitas pelo utilitário são automaticamente aplicadas ao nó. Pode-se também configurar ou alterar a configuração do Mosix diretamente em seus arquivos de configuração, onde os mais importantes são:

- mosix.map – trata-se da lista dos nós do cluster;
- mosip – o endereço IP usado pelo Mosix;
- features – indica as características do nó;
- secret – arquivo que guarda a senha de acesso ao cluster;
- ecsecret – senha para autenticação de clientes de serviço batch;
- essecret – senha para autenticação de servidor de serviços batch;
- freeze.conf – configuração de políticas de congelamento de processos;
- queue.conf – indica o gerenciador de fila de processos padrão;
- userview.map – indicação lógica dos nós para serem exibidos no aplicativo monitor.

Após qualquer modificação em algum dos arquivos de configuração deve ser executado o comando "setpe" para aplicar as modificações ao nó.

Nota-se, portanto, que a instalação e configuração do Mosix pode ser feita de forma totalmente automática por meio do comando "*mosix.install*", ou de forma manual mediante a edição de cada um dos arquivos de configuração, conforme relacionado anteriormente. Pode ocorrer ainda que o procedimento automático ou manual de instalação do Mosix se depare com algum conflito de configuração do sistema operacional ou do hardware onde está sendo instalado e resulte em erro na compilação do Kernel. Para solucionar esse tipo de problema é

necessária a compilação manual do Kernel do Linux conforme as instruções relacionadas no item 12 do Apêndice C.

4.2.4 Configuração de cluster via SSH, NFS e MPICH

Uma alternativa à configuração do cluster com Mosix foi configurar a rede, os serviços SSH e NFS e instalar a biblioteca de paralelização e troca de mensagens MPICH para definir um cluster nos moldes do cluster *Beowulf*, de acordo com (RENDERFARM, 2009). A relação das instruções para tal configuração está disponível no Apêndice D.

5 RESULTADOS E DISCUSSÕES

Os principais resultados desse projeto foram obtidos após uma série de testes iniciais para avaliar o funcionamento do cluster e, posteriormente, para analisar os resultados positivos da estrutura montada com a ferramenta de gestão MOSIX devidamente instalada e configurada. Os primeiros testes foram feitos utilizando-se as máquinas virtuais. Após a definição do modelo de configuração ideal a ser adotado, essa configuração foi replicada nas máquinas físicas do laboratório do ICTE.

Entre alguns testes iniciais realizados para a comprovação do funcionamento do cluster, os principais foram: testes com o software GAMESS, testes genéricos e testes utilizando o método de Monte Carlo Quântico.

5.1 CONFIGURAÇÃO E INSTALAÇÃO DO SOFTWARE GAMESS PARA TESTES DE SIMULAÇÃO DE PROBLEMAS EM FÍSICO-QUÍMICA

Conforme Schmidt (*et al.*, 1993), o GAMESS é um programa para cálculos de estrutura eletrônica, ou seja, é um programa de Química Quântica. Para o funcionamento, esse software requer instalação e configuração de um compilador Fortran (gfortran), de uma biblioteca matemática (ATLAS) e de pacotes complementares e dependentes para a correta configuração, tais como editor de *shell script* como *csh* ou *tcsh*, sugeridos em seu manual de instalação. Para isso, o sistema operacional deve ter configurado o arquivo */etc/apt/sources.list*, para que o comando *apt-get* funcione corretamente garantindo a instalação dos pacotes necessários ao funcionamento do GAMESS, bem como de suas dependências.

No Linux Debian esses pacotes podem ser obtidos da seguinte maneira:

- *apt-get install –f gfortran*
- *apt-get install –f libatlas-base-dev*
- *apt-get install –f tcsh*

Após efetuar o download do programa do site http://www.msg.chem.iastate.edu/GAMESS/download/register/ e descompactá-lo, deve-se seguir as instruções no arquivo *../gamess/machines/readme.unix*. Mais informações sobre essa ferramenta podem ser obtidas no site

http://www.msg.chem.iastate.edu/GAMESS/GAMESS.html e instruções mais detalhadas sobre a instalação e configuração do GAMESS encontram-se no Apêndice E.

Para testar o cluster via o software GAMESS (SCHMIDT *et al.*, 1993), escolheu-se para a simulação de um fenômeno real, para cálculo de energia total do sistema e outras propriedades físico-químicas de interesse, uma molécula de porte médio (39 átomos, 146 elétrons) em meio ao efeito de solvente (acetato de etila). Conforme os comandos a seguir, os testes foram executados considerando os núcleos do processador de uma máquina como nós com o objetivo de identificar a diferença dos resultados de processamento relacionada à quantidade de nós disponíveis, conforme pode ser visto no quadro 4. Os comandos foram:

- comandos para execução do teste com apenas um núcleo do processador:
  - *./rungms exemplo.inp 01 1 >& exemplo.log*
- comandos para execução do teste com dois núcleos do processador:
  - *./rungms exemplo.inp 01 2 >& exemplo.log*

Quadro 4: Resultados de testes de fenômenos físico-químicos no Linux

Núcleos	**Tempo total de CPU**	**Tempo total de CLOCK**	**Utilização da CPU**
01 nó	2688 s (~45 min.)	8788 s (~145 min.)	30,58%
02 nós	1354 s (~23 min.)	8554 s (~143 min.)	15,83%

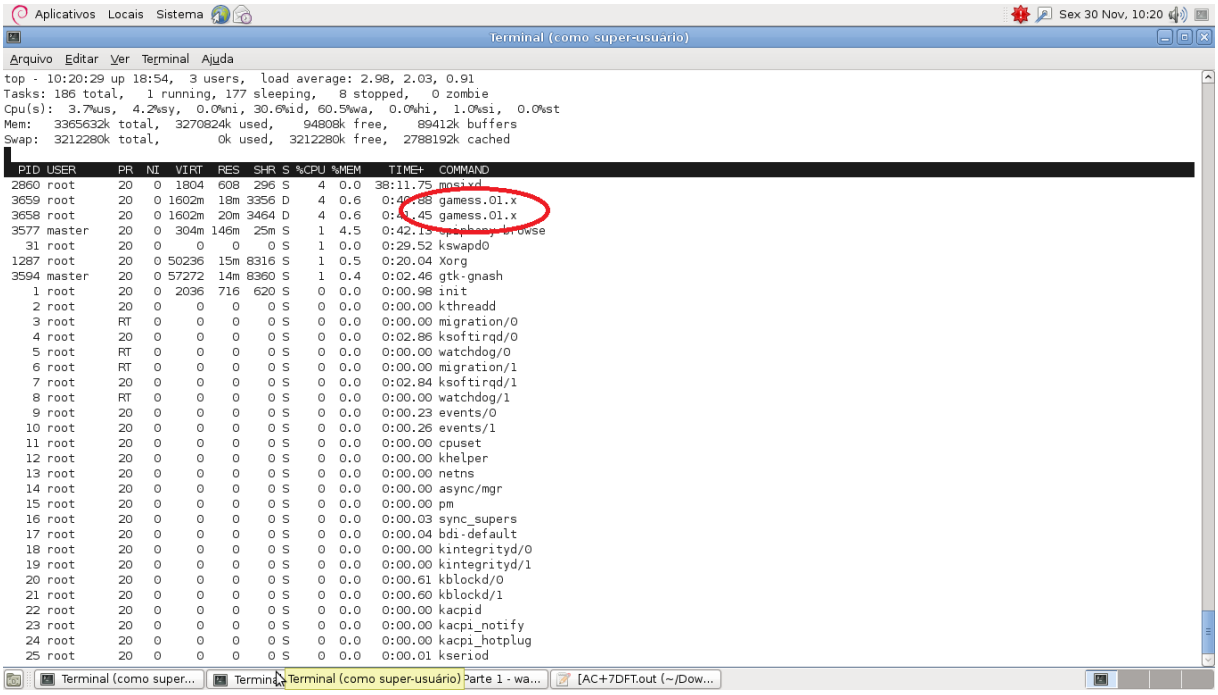
Fonte: Quadro elaborado pelo autor

Resumidamente, o tempo total de CPU é o tempo total que as unidades de computação (núcleo do processador) estão ativamente executando um trabalho. Para trabalhos mais simples, este é menor que o tempo real gasto (tempo total de clock), pois algum tempo é perdido em I/O e operações do sistema. Já o tempo total de clock indica o tempo total de execução de um trabalho. Como informação adicional, para trabalhos executados em paralelo o tempo de CPU é normalmente maior que o de clock, pois vários núcleos do processador estão ativos na execução destes trabalhos ao mesmo tempo, sendo que o quociente entre os tempos de CPU e clock mede a eficiência desta execução em paralelo.

Ou ainda, em síntese, o tempo total de CPU é descrito como o tempo total gasto executando linhas de código (instruções) de um programa do usuário (tempo de usuário) somado ao tempo gasto com tarefas necessárias do sistema operacional (tempo de sistema) e o tempo total de clock é o sinal usado para coordenar as ações de dois ou mais circuitos eletrônicos.

A figura 2 exibe o resultado do comando *top*, nativo do Linux, e mostra a paralelização dos processos *gamess.01.x*, usando dois nós, conforme destaque, durante a execução dos testes.

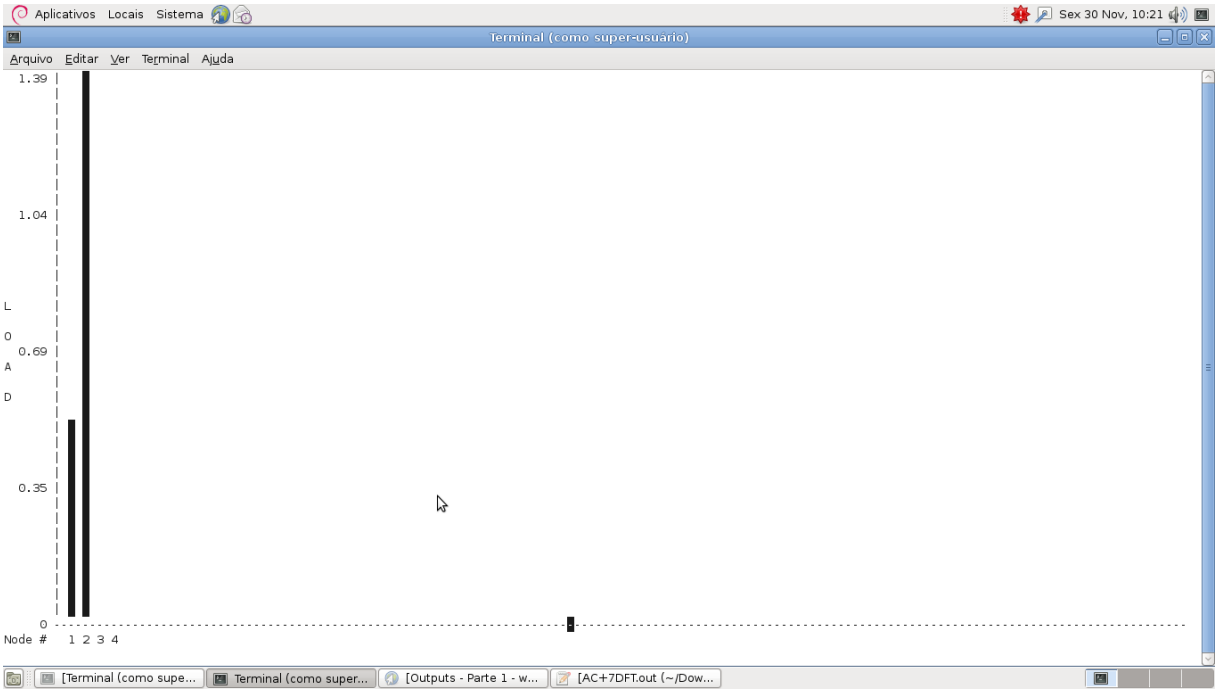
Figura 2: Identificação de processos do GAMESS em execução paralelizada



Fonte: Tela capturada pelo autor

A performance dos nós do cluster pode ser visualizada no gráfico gerado pelo Mosix, a partir do comando *mon*, conforme mostra a figura 3. Nessa figura identifica-se uma carga maior no nó 02 por ser o nó do cluster onde se originou o processo.

Figura 3: Monitoramento dos processos e recursos entre os nós do cluster



Fonte: Tela capturada pelo autor

Um exemplo de um arquivo de saída completo com os resultados de um processamento de moléculas via execução do GAMESS encontra-se no anexo D.

5.2 TESTES GENÉRICOS

O próprio Mosix fornece um aplicativo chamado *testload* que promove uma grande quantidade de carga à CPU para testes. Esse teste não permite fazer comparação de tempo quanto à execução local ou distribuída, pois ele é orientado a tempo de execução e não a volume de dados.

Foram feitos testes genéricos com algoritmos relativamente pequenos, cuja diferença de tempo entre processamento local e distribuído é pequena, porém, notável. Com esses algoritmos menores notou-se que o tempo de execução distribuído era maior que o tempo de execução localmente, devido ao grande uso da rede para comunicação entre os nós, tornando inviável o uso do processamento distribuído para processos pequenos, causando queda de performance.

Na fase em que esses testes foram executados, o Mosix ainda não estava funcionando perfeitamente em função da dificuldade ainda existente com a compilação do kernel do Linux e os ajustes das versões compatíveis de Mosix e Kernel. Portanto, foram apenas testes

contendo instruções computacionais repetitivas simples executadas em laços encadeados para demonstrar o funcionamento do cluster, ou seja, uma constatação da interação entre os nós do cluster. Os comandos foram os listados a seguir, sendo que eram armazenadas num arquivo de texto a data e a hora de início e fim do teste como parâmetro para calcular o tempo gasto no procedimento:

date >> teste.txt;
awk "BEGIN {for(c=0; c<50000; c++) for (d=0; d<50000; d++)}";
date >> teste.txt

Os resultados desses testes são apresentados no quadro 5, sendo que deve ser destacado que as referências ao Mosix eram tentativas de aproveitar algum recurso eventualmente disponível antes da compilação do kernel do Linux, essencial para o funcionamento efetivo do Mosix. Porém, poucas funcionalidades são disponibilizadas nessa ocasião, como por exemplo, a geração de gráficos com dados de comportamento do cluster como pode ser visto na figura 4.

Quadro 5: Resultados de testes genéricos para validação do funcionamento do cluster

Testes	Condições	Tempo
1	Usando 3 máquinas com o Mosix, inicializado pelo parâmetro *mosrun –e* antes do comando awk.	02 min. 18s
2	Usando 3 máquinas sem o parâmetro do mosrun do Mosix.	02 min. 17s
3	Usando apenas 1 máquina sem rede e sem o Mosix.	02 min. 27s

Fonte: Quadro elaborado pelo autor

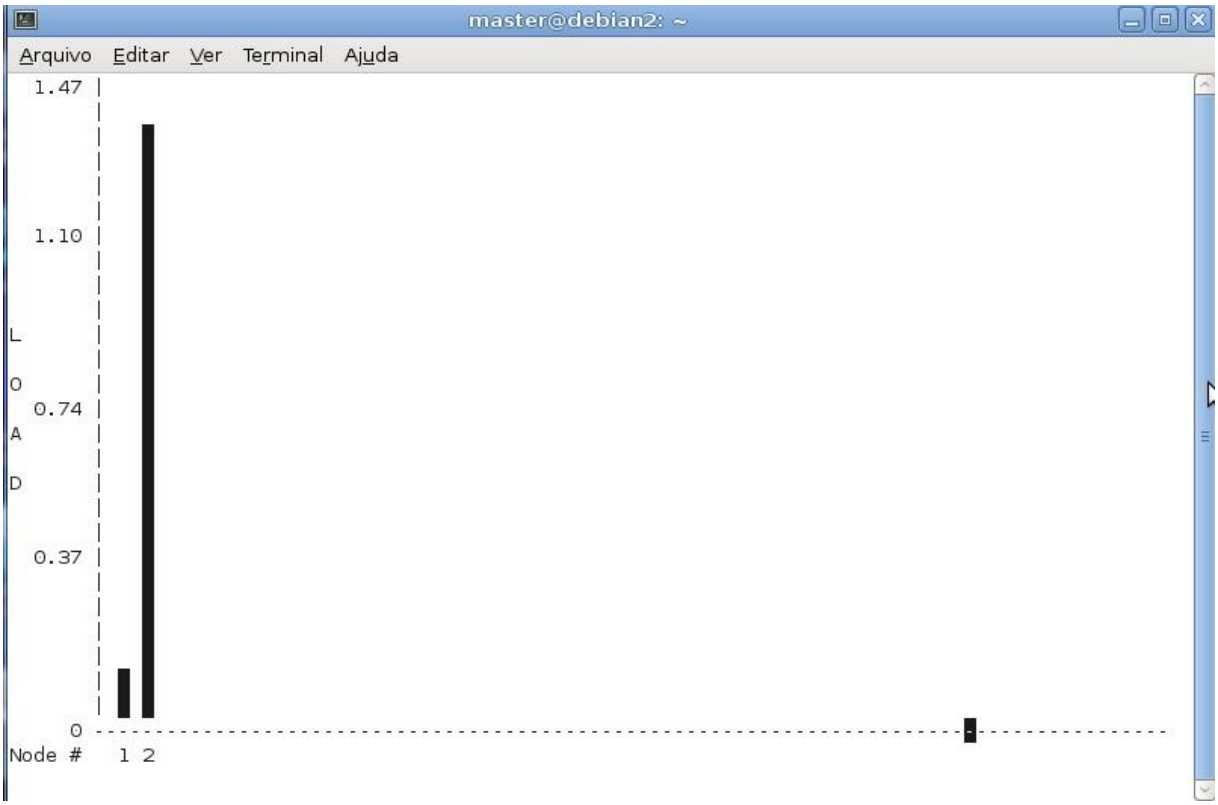
Todos os testes referidos anteriormente foram realizados inicialmente nas máquinas virtuais e, posteriormente, submetidos também ao cluster composto por máquinas reais.

Por meio do comando *mon* (ou *mosmon* nas versões mais novas do MOSIX), um dos principais aplicativos do Mosix, que é o monitor do cluster, são fornecidas muitas informações importantes sobre os nós, como a carga de cada uma das CPU’s e também a memória em uso para cada nó acessado. Essa aplicação permite monitorar a distribuição de carga e a performance dos nós do cluster durante os testes.

O gráfico da figura 4 foi gerado pelo comando *mon*, nativo do Mosix, e ilustra a interação entre dois nós ativos do cluster cujo comportamento foi capturado no momento dos

testes. No eixo horizontal são listados os nós ativos componentes do cluster no momento da execução desse comando e no eixo vertical são listados os valores de carga de processamento calculados instantaneamente pelo sistema. Neste caso, como o teste foi executado a partir do nó 2, ele está consumindo uma maior carga que o nó 1, mas internamente estão se comunicando e compartilhando recursos de forma que, quando os recursos necessários ao processamento não forem suficientes para a execução apenas pelo nó 2, ocorrerá a migração de processos de forma transparente para o nó 1, resultando em um balanceamento de carga.

Figura 4: Funcionamento do cluster para um teste genérico



Fonte: Tela capturada pelo autor

O teste com processos simultâneos se deu com um *shell script* (linguagem de programação de alto nível interpretada pelo *Shell* do Linux). A execução local desse *shell script* demandou um tempo bem elevado, e distribuidamente o mesmo teste foi concluído num tempo bem reduzido, conforme será demonstrado a seguir. Esse teste foi feito após a compilação do kernel com o Mosix, quando esse passou a funcionar efetivamente e todas as funcionalidades do cluster foram ativadas. Os modelos de testes adotados inicialmente nesse tópico foram repetidos, porém, submetendo uma maior carga de processamento, via comandos a seguir, cujos resultados podem ser vistos no quadro 6:

date >> teste.txt;
mosrun -e awk "BEGIN{for(c=0;c<100000;c++)for(d=0;d<100000;d++)}";
date >> teste.txt

Neste caso, o comando é precedido pela instrução "*mosrun -e*", que inicializa o Mosix para operacionalizar o cluster com todas suas funcionalidades de balanceamento de carga e migração de processos, entre outras, ajustando todos os processos do sistema operacional. Como pode ser visto no quadro 6, com o cluster e o Mosix devidamente configurados, para o teste em questão tem-se um ganho de aproximadamente 40% do tempo ao executar 03 procedimentos simultaneamente em relação à soma de tempo necessária para a realização de cada um desses procedimentos individualmente. Isso demonstra a eficiência do cluster com Mosix para a execução de grandes volumes de processamento.

Quadro 6: Resultados dos processos simultâneos utilizando métodos genéricos e Mosix

Qtde. Processos	**Tempo**	
	Individualmente	**Simultaneamente**
01	496 s (~ 8 min.)	496 s (~ 8 min.)
03	1488 s (~ 25 min.)	889 s (~ 15 min.)
Ganho		599 s (~ 10 min.) (40 %)

Fonte: Quadro elaborado pelo autor

5.3 TESTES UTILIZANDO O MÉTODO DE MONTE CARLO

Outros testes que também foram aplicados ao cluster para justificar seu efetivo funcionamento, foram testes que utilizaram como ferramenta o método de Monte Carlo aplicado a problemas de estrutura eletrônica.

A denominação "método de Monte Carlo" tornou-se uma expressão geral associada ao uso de números aleatórios e do conceito de probabilidade. Para que uma simulação de Monte Carlo esteja presente em um estudo, basta que este faça uso de números aleatórios na verificação de algum problema. Este termo foi criado pelos pesquisadores S. Ulam e Nicholas Metropolis em homenagem à atividade mais popular de Monte Carlo, Mônaco, os jogos (GUJARATI, 2002, *apud* LIMA *et al*, 2008). Este método é conhecido há um bom tempo, e uma de suas aplicações mais "notáveis" foi feita por Ulam e Von Neumann no Projeto

Manhattan, durante a Segunda Guerra Mundial, na construção da bomba atômica (YORIYAZ, 2009).

Esta técnica pode ser utilizada largamente na avaliação de projetos, onde os riscos envolvidos podem ser expressos de forma simples e de fácil leitura, e as simulações auxiliam a decisão, como por exemplo na previsão de fluxo de caixa operacional, na análise de risco em custos. Além disso, pode ser utilizado em aplicações em física médica e cálculo de área de superfície, entre outros. Assim, os indicadores deixam de ser determinísticos e passam a ser estocásticos, probabilísticos.

Matematicamente, o método de Monte Carlo (MC) está frequentemente associado à solução de integrais multidimensionais, comuns em problemas de quântica. A associação do método com integrais é feita de maneira simples e intuitiva e sua vantagem está na possibilidade de reduzir sistemas com grande número de dimensões mediante a determinação de uma média. O método de Monte Carlo recebe a designação particular de Monte Carlo Quântico (MCQ) quando empregado na resolução da equação de Schrödinger (ANGELOTTI *et al.*, 2008, *apud* HAMMOND *et al.*, 1994).

Os testes envolvendo o método de Monte Carlo Quântico foram feitos por meio de um programa de simulação escrito em linguagem Fortran - FORmula TRANslation (em português: tradução de fórmulas) - e que calcula, como principal propriedade físico-química, a energia total do sistema em processos físico-químicos, sendo dois exemplos específicos: a molécula de metano (CH_4) e o átomo de neônio (Ne). Abaixo seguem os comandos para a execução do programa e os resultados são apresentados no quadro 7:

root@debian2:#cd /home/master/testefortran

root@debian2:/home/master/testefortran# ./a.out

Quadro 7: Resultados de testes com fenômenos físico-químicos no cluster

Nós	**Elementos**	
	CH_4	Ne
01	1853 s (~31 min.)	746 s (~ 12 min.)
03	1841 s (~31 min.)	724 s (~ 12 min.)

Fonte: Quadro elaborado pelo autor

As diferenças de tempo para a conclusão desses testes foram pequenas, imperceptíveis se usado o minuto como unidade de medida de tempo, os quais foram realizados ora com um nó, ora com três nós, dispensando a realização com dois nós por causa dessa mínima

diferença. Essas pequenas diferenças são atribuídas ao pequeno tamanho das moléculas, além de outros fatores como interferências na rede, que anulam as vantagens obtidas pelo acréscimo de nós. A opção por esses testes simples foi exclusivamente para certificar-se da configuração e do funcionamento do cluster, não tendo como foco, nessa circunstância, o ganho de tempo ou performance.

Um exemplo de um arquivo de saída completo de um processamento de moléculas feito por esse método encontra-se no anexo E.

Um outro teste mais robusto envolvendo cálculo de energia em fenômenos físico-químicos envolvendo a molécula de amônia foi aplicado nesse cluster, cujo resultado pode ser visualizado na Tabela 1. Esse teste retratou a eficiência do cluster pelo fato de que várias tarefas foram submetidas simultaneamente e as condições computacionais oferecidas pelo cluster, ainda sem o Mosix funcionando perfeitamente, permitiram obter o resultado final desse processamento num tempo bem menor do que o total que seria gasto para a execução dessas mesmas tarefas individualmente.

Tabela 1: Energia Total (E) e potencial de tripla ionização (PTI) de vários estados triplamente ionizados, com participação de elétron de caroço, da molécula de amônia.

Estado	E (u.a.)1	PTIC1 (eV)1	E (u.a.)2	PTIC2 (eV)2	PTIR (eV)3	ΔTI (eV)4	
Neutro	-56,572		-56,572			ΔTI1 (eV)	ΔTI2 (eV)
2A1	-39,819	455,849	-39,550	463,168	470,431	14,582	7,263
4A2	-39,425	466,569	-39,285	470,379	470,810	4,241	0,431
2B1	-39,265	470,923	-39,186	473,073	474,026	3,103	0,953
2B2	-39,188	473,018	-39,154	473,943	475,207	2,189	1,264
2A2	-39,478	465,127	-39,248	471,386	476,080	10,953	4,694
2A2	-39,024	477,481	-38,993	478,324	479,240	1,759	0,916
2B2	-39,023	477,508	-38,936	479,875	479,320	1,812	0,555
2B1	-39,021	477,562	-38,935	479,902	480,593	3,031	0,691
2A1	-38,822	482,977	-38,829	482,787	482,519	0,458	0,268
4B1	-38,650	487,657	-38,635	488,065	487,973	0,316	0,092
4A2	-38,742	485,154	-38,709	486,052	488,317	3,163	2,265
2A1	-38,905	480,719	-38,748	484,991	491,578	10,859	6,587

4B2	-38,586	489,399	-38,541	490,623	492,381	2,982	1,758
2A1	-38,637	488,011	-38,495	491,875	497,398	9,387	5,523
2A2	-38,052	503,929	-38,028	504,582	508,638	4,709	4,056

Fonte: (ANGELOTTI, 2013)

1. Energia total (E) e potencial de tripla ionização (PITC1) calculado com Monte Carlo de Difusão usando orbitais Hartree-Fock.
2. Energia total (E) e potencial de tripla ionização (PITC2) calculado com Monte Carlo de Difusão usando orbitais Kohn-Sham.
3. Resultados teóricos de tripla ionização de Tashiro et al. (Tashiro, M.; Ueda, K. M.; Ehara, M. J Chem Phys 2011, 135, 154307).
4. Diferença (abs) entre a tripla ionização obtida com Monte Carlo de Difusão usando orbitais Hartree-Fock e Kohn-Sham, respectivamente, e de Tashiro et al.

Após a compilação do Kernel do Linux com uma versão compatível do Mosix, quando o cluster passou a ter o funcionamento administrado pelo Mosix, esse tipo de teste também foi submetido, porém com um maior volume de processamento. A seguir são listados os comando adotados para a execução desses testes, cujos resultados são mostrados no quadro 8:

1) *date >> teste.txt; mosrun -e ./a1.out; date >> teste.txt*

2) *date >> teste.txt; mosrun -e ./a2.out; date >> teste.txt*

Nestes testes, os programas foram precedidos pela instrução "*mosrun -e*" que inicializa as funções de gestão do cluster pelo Mosix. No comando 1 o programa "*a1.out*" é constituído de um conjunto de instruções em Fortran para cálculo de energia do átomo de Boro com duas configurações enquanto que no comando 2 o programa "*a2.out"* envolve três configurações.

Quadro 8: Resultados dos processos simultâneos utilizando método de Monte Carlo Quântico e Mosix

Identificação Processo	**Tempo**	
	Individualmente	**Simultaneamente**
1	2715 s (~ 45 min.)	
2	3965 s (~ 66 min.)	
1 e 2	6680 s (~ 111 min.)	4334 s (~ 72 min.)
Ganho		2346 s (~ 39 min.) (35 %)

Fonte: Quadro elaborado pelo autor

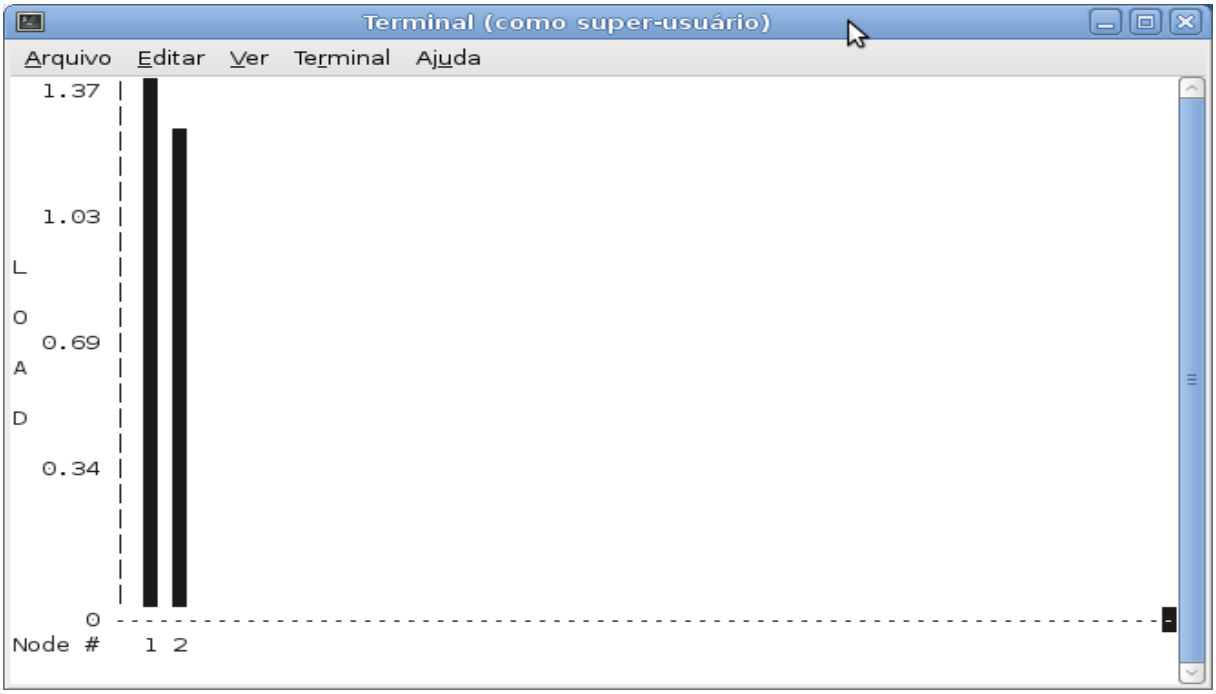
Como pode ser visto no quadro 8, houve um ganho de tempo de aproximadamente 35% com o processamento simultâneo dos testes em relação ao tempo total gasto com os processamentos individuais dessas simulações. Isso justifica a eficiência da configuração do

cluster com Mosix para processamento em grande escala e, principalmente, de processos simultâneos.

5.4 COMPORTAMENTO DO CLUSTER COM TESTE SIMULTÂNEO E MOSIX

Para fazer uma análise mais detalhada do comportamento do cluster com Mosix, após a compilação do Kernel, foi adotado o processamento dos cálculos realizados no item 5.3. A figura 5 é um gráfico gerado pelo comando *mosmon* (nativo do Mosix) apresentando o comportamento do cluster durante a execução dos referidos testes no cluster com dois nós ativos. Observa-se uma distribuição balanceada de carga de processamento entre esses nós que é determinada exclusivamente pelo Mosix que se encarrega de migrar os processos, decidir qual nó é mais eficiente para a execução dos processos entre outras ações, tudo de forma transparente ao usuário que submeteu as tarefas.

Figura 5: Comportamento do cluster com Mosix na execução de processos simultâneos

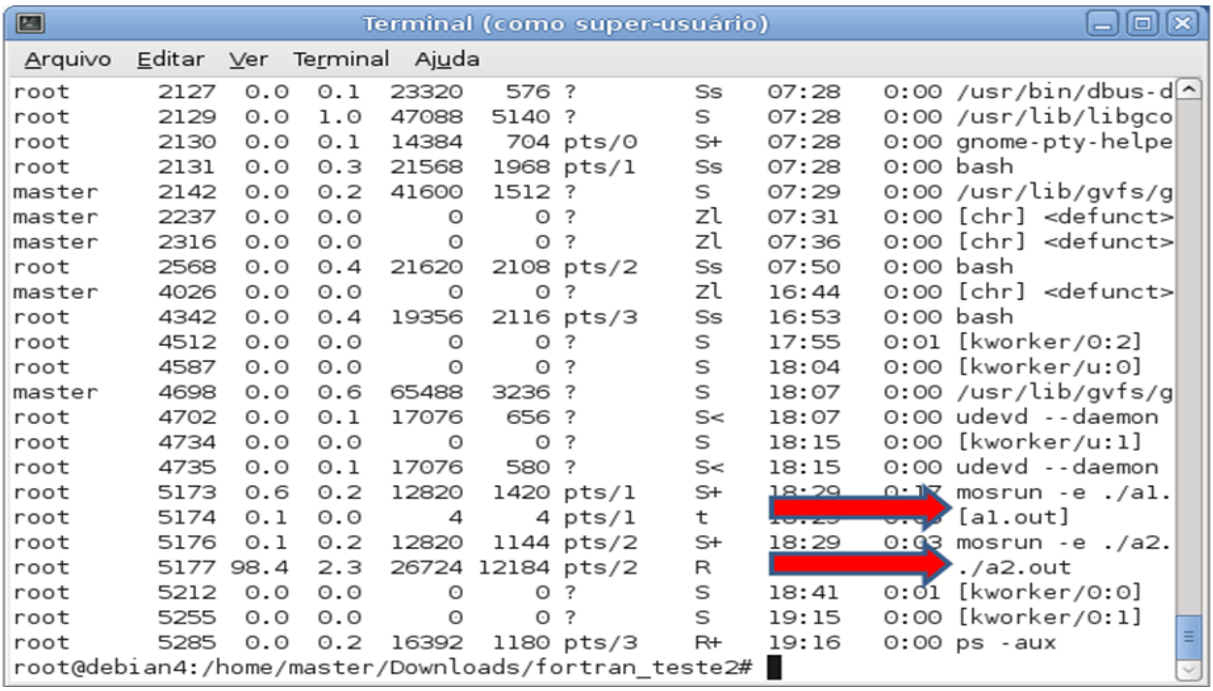


Fonte: Tela capturada pelo autor

A figura 6 refere-se a uma tela capturada com o resultado do comando “ps -aux” (nativo do Linux) que lista todos os processos em execução pelo sistema operacional. O

destaque mostra os processos "*a1.out*" e "*a2.out*" sendo executados simultaneamente. No caso do processo "*a1.out*", nota-se que é apresentado entre colchetes o que significa que esse processo foi iniciado no nó corrente, porém, o Mosix migrou sua execução para outro nó do cluster. Ao final do processamento o resultado é exibido no nó de origem.

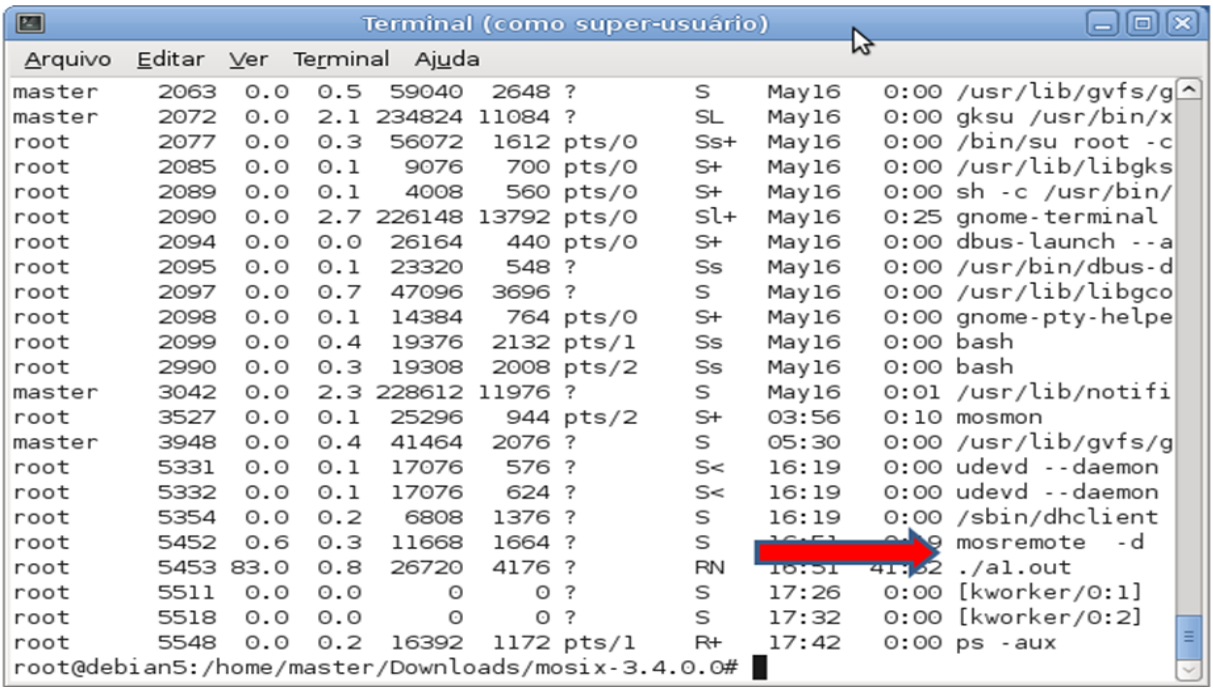
Figura 6: Identificação de processos simultâneos controlados pelo Mosix



Fonte: Tela capturada pelo autor

Sob a perspectiva do nó que recebeu o processo migrado, o resultado do comando "*ps -aux*" mostra a relação de processos conforme ilustrado na figura 7, onde nota-se a execução do programa "*a1.out*" e um processo associado "*mosremote –d*" conforme o destaque nessa figura. Esse processo associado identifica que o processo "*a1.out*" veio migrado de outro nó por atribuição do Mosix.

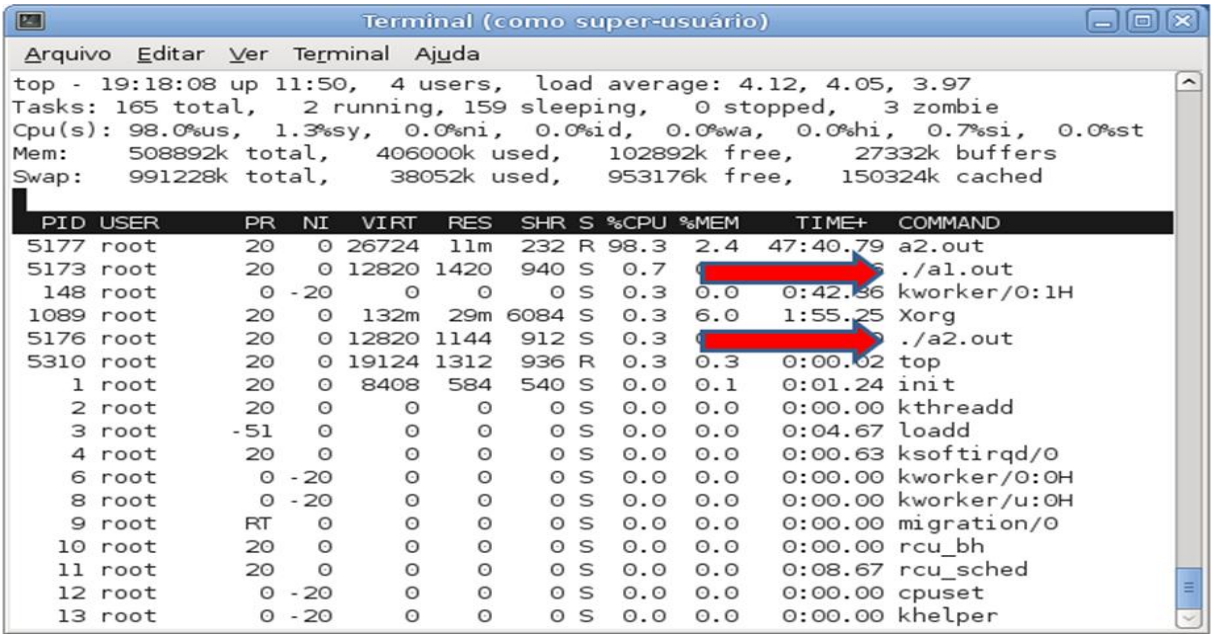
Figura 7: Identificação de processos migrados pelo Mosix



Fonte: Tela capturada pelo autor

Numa outra visão, por meio do comando "*top*" (nativo do Linux) obtém-se como resultado a relação de processos mostrada na figura 8, onde pode-se ver os dois processos referentes aos programas "*a1.out*" e "*a2.out*", em destaque. Porém, nessa visão não é exibida nenhuma identificação de migração dos processos.

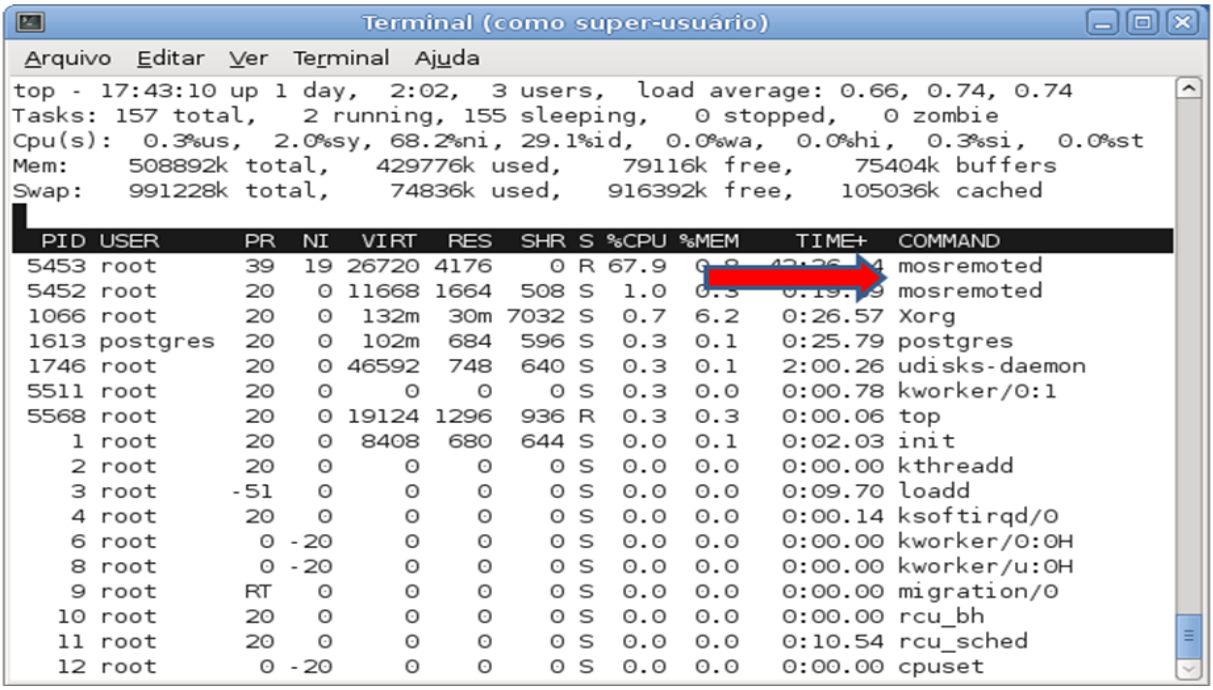
Figura 8: Identificação de processos simultâneos pelo Linux



Fonte: Tela capturada pelo autor

Na visão dos resultados do comando "*top*" executado no nó de destino dos processos migrados observa-se apenas as referências "*mosremoted*", conforme destaque na figura 9, que identifica a existência de processos migrados executados em segundo plano, ou seja, de forma transparente ao usuário.

Figura 9: Identificação de processos remotos controlados pelo Mosix



Fonte: Tela capturada pelo autor

6 CONCLUSÕES

Buscando representar a viabilidade desse projeto, em termos financeiros, foram pesquisados e comparados os custos para aquisição de computadores de grande porte e microcomputadores.

Referências de compras efetuadas por órgãos públicos são encontradas no site www.comprasnet.gov.br. Neste site foram localizados dois processos de compra de bens desse tipo e, baseados nos valores encontrados, constata-se que o valor de R$ 11.950,00 gasto para a aquisição de um computador de grande porte, conforme o Anexo B, é suficiente para a compra de aproximadamente 13 microcomputadores padrões, de acordo com o Anexo C, no valor de R$ 934,00 cada.

Neste caso, um computador de grande porte, chamado *Mainframe*, é um tipo de computador para trabalho pesado em grandes instituições, como bancos e órgãos de governo. Já microcomputadores podem ser utilizados em atividades básicas, em atividades administrativas ou em laboratórios de informática para execução de aplicativos comuns de diversas áreas. Como essas atividades não ocorrem a todo o momento, esses microcomputadores apresentarão ociosidade no uso, cabendo, portanto, otimizar seu uso aproveitando-os na implementação de clusters que substituem os supercomputadores nas atividades de pesquisa. Por outro lado, quando ocorre a aquisição de equipamentos de grande porte, é exclusivamente para atender demandas críticas, como servidores de dados e aplicações, ainda não sendo prioridade como aquisição para áreas de pesquisa.

É notável o custo reduzido desse projeto com a possibilidade de utilização de recursos e equipamentos já disponíveis dentro da própria Instituição como algo real, dispensando aquisições de equipamentos de grande valor, com altos valores de manutenção e uma alternativa quando se tem restrições financeiras mais elevadas e busca-se fazer mais gastando menos. Com a eficiência da implementação do cluster desenvolvido nesse projeto, e comprovada nos testes, os resultados serão positivos quanto à economia de recursos e atendimento às demandas de diversas áreas de pesquisa da UFTM.

Algumas desvantagens ou dificuldades desse projeto podem ser a manutenção dos equipamentos do cluster sempre em perfeito funcionamento, cabendo, nesse caso, a necessidade de um técnico que tenha conhecimentos de toda a estrutura para eventuais atualizações, configurações ou substituição dos equipamentos. Outro fator limitador pode ser o nível de conhecimento e habilidade dos usuários para interagir com as ferramentas de

administração do cluster para submeter as tarefas a serem processadas. Além disso, devem ser consideradas medidas para superar os eventuais gargalos apresentados pela estrutura que possam comprometer o funcionamento do cluster.

7 CONSIDERAÇÕES FINAIS

Após entender o funcionamento de um cluster de computadores, o ambiente de processamento de alto desempenho e suas características, usabilidade, requisitos de hardware e de software, conseguiu-se projetar um cenário útil para sua aplicação e funcionamento.

Os cálculos complexos, as simulações e modelagem computacionais não conseguiriam grandes resultados sem a ajuda do processamento de alto desempenho. Com o resultado desse projeto, muitas áreas, aos poucos, podem se beneficiar com os equipamentos que permitem processar a informação de uma maneira mais rápida.

Dentro das instituições de ensino, o uso básico desta ferramenta, vem acompanhado de pesquisas acadêmicas e, portanto, essas instituições, mais precisamente as universidades, devem repensar como as pesquisas acadêmicas são encaradas, visto que o incentivo à pesquisa e à divulgação cientifica é um dos tripés do Ensino Superior. A viabilidade da implantação deve estar em pauta em novas propostas para laboratórios de pesquisa que envolvam processamento de alto desempenho, principalmente em instituições com fortes restrições financeiras.

Portanto, a junção de instituições de ensino, pesquisa acadêmica e cluster se aplicam no momento em que ambas se acharem importantes no desenvolvimento científico, tecnológico e profissional.

Um dos resultados desse projeto foi a criação de um cluster com alguns microcomputadores “ociosos” de um laboratório do Instituto de Ciências Tecnológicas e Exatas na UFTM para o processamento de fenômenos químicos, como exemplo inicial, sem nenhum custo adicional, visto que os computadores já existentes para atividades acadêmicas puderam ser compartilhados com esse projeto. Outra forma de dispensar custos foi o uso de softwares livres para todas as aplicações. Economizou-se consumo de energia elétrica, dispensando o uso de todos esses equipamentos durante a concepção desse projeto, por meio do uso de máquinas virtuais instaladas em um computador portátil pessoal, onde eram testadas as configurações até definir um modelo para ser adotado nos microcomputadores que compuseram esse cluster.

Outro resultado positivo foi a elaboração de instruções ou manuais necessários para a configuração de um cluster computacional relacionando procedimentos para a instalação de sistema operacional “ideal”, softwares gratuitos para a instalação, configuração e gestão do cluster, além de outras ferramentas para oferecer recursos necessários para o processamento

dos testes ou de outras tarefas no cluster. Essas instruções contemplam comentários, procedimentos e informações adicionais ausentes nos manuais de instalação das próprias ferramentas e que, assim, possibilita que usuários sem muitos conhecimentos dessa área tenham condições de implementar essa solução de cluster para o desenvolvimento de seus projetos. Após a conclusão, são disponibilizadas como apêndices as instruções sobre o Virtualbox, instruções para criação e configuração de máquinas virtuais, instruções para instalação e configuração de cluster com o Mosix, instruções para configuração de cluster via SSH, NFS e MPICH, instruções otimizadas para instalação, configuração e utilização do GAMESS como ferramenta de referência para processamento de fenômenos físico-químicos genéricos e instruções para instalação e configuração de sistema operacional e da rede de computadores para o cluster.

Em relação aos testes, estes foram realizados com um tradicional pacote computacional de Química Quântica e com um específico método programado em linguagem Fortran, ambos para sistemas atômicos e moleculares, porém, isto não faz parte dos objetivos deste trabalho, e foram apenas para mostrar o bom funcionamento e a viabilidade da montagem do cluster com poucos recursos financeiros. Outros tipos de testes podem ser realizados exigindo as devidas configurações dos pacotes necessários apenas para sua interação com o sistema operacional, dispensando modificações na configuração do cluster. Fica a critério do pesquisador que for executar outros testes a necessidade de acrescentar novos microcomputadores ao cluster para ganhar mais eficiência e redução de tempo. Porém, a quantidade de nós do cluster pode se tornar indiferente a partir de um determinado volume de processamento, visto que a carga de processamento é distribuída entre todos os nós e, caso ela não seja saturada, a diferença de tempo para realização dos testes pode ser mínima com o acréscimo de novos nós.

O Mosix contempla uma série de funcionalidades e comandos que não foram destacados nesse projeto pelo fato de que o objetivo foi mostrar a viabilidade de composição de cluster com equipamentos ociosos por meio da gestão do Mosix na sua configuração mais básica. Toda a documentação sobre as funcionalidades do Mosix está disponível em (MOSIX, 2013d).

Posteriormente, o objetivo é aumentar o número de computadores no cluster e estender sua aplicação para outros fenômenos e sistemas reais. Um projeto futuro também contempla a proposta de gerar uma mídia com instalação automática, contendo o sistema operacional e todas as ferramentas essenciais para o funcionamento do cluster e suportar a instalação de aplicativos como o Scilab ou outras plataformas necessárias para execução de atividades de

cálculos e processos de pesquisa, em geral, de diversas áreas. Desta forma, bastaria a inicialização dos equipamentos disponíveis carregando o sistema a partir dessas mídias, facilitando para qualquer usuário a utilização de qualquer microcomputador para compor cluster.

As dificuldades encontradas para o desenvolvimento desse projeto foram:

a) configuração e adaptação dos pacotes corretos e necessários para configuração do Mosix antes de encontrar uma distribuição compatível;
b) configuração dos pacotes do Linux corretos e necessários para a execução das ferramentas necessárias para execução dos testes;
c) conciliar os horários disponíveis para implementações e testes com os horários disponíveis dos laboratórios;
d) grande tempo para encontrar versões compatíveis entre Mosix e Kernel do Linux;
e) ajustar a configuração de rede nos ambientes onde foram executados os testes antes da definição de um local permanente para a instalação do mesmo;
f) compilar o Kernel do Linux para inserir os componentes do Mosix;
g) localizar e preparar testes eficientes para validar o funcionamento do cluster.

REFERÊNCIAS

ALECRIM, E. Cluster: principais conceitos. In: INFOWESTER. [S.l.], 2013. Disponível em: <http://www.infowester.com/printversion/cluster.php>. Acesso em: 23 fev. 2012.

_____. Diferenças entre hub, switch e roteador. In: INFOWESTER. [S.l.], 2004. Disponível em: <http://www.infowester.com/hubswitchrouter.php>. Acesso em: 23 fev. 2012.

ALUVINO, J. C. **Alto poder de computação com cluster de computadores usando Mosix.** 2008. 93 f. Trabalho de Conclusão de Curso (Graduação em Tecnologia em Informática) -- Faculdade de tecnologia de Guaratinguetá, Guaratinguetá, SP, 2008.

ANGELOTTI, W. F. D. Calculations of triple ionization with participation of electron from the inner shell by Diffusion Monte Carlo, 2013.

ANGELOTTI, W. F. D. et al. Uma abordagem simplificada do método Monte Carlo Quântico: da solução de integrais ao problema da distribuição eletrônica. **Química Nova**, São Paulo, v. 31, n. 2, p. 433-444, 2008.

BACELLAR, H. V. **Cluster: Computação de Alto Desempenho.** Instituto de Computação da Universidade Estadual de Campinas, Campinas, SP, Brasil, 2010.

BARAK, A.; LA'ADAN, O. The MOSIX multicomputer operating system for high performance cluster computing. **Future Generation Computer Systems,** v. 13, n. 4-5, p. 361-372, 1998. Disponível em: < ftp://www.moscoso.org/pub/os/cluster/mosix/mosultp/mosixhpcc.pdf>. Acesso em: 10 maio 2013.

BARAK, A.; SHILOH, A. The MOSIX cluster operating system for high-performance computing on Linux clusters, multi-clusters and clouds: a white paper. In: MOSIX: cluster operating system. [S.l.], 2013. Disponível em: <http://www.mosix.org/pub/MOSIX_wp.pdf>. Acesso em: 13 nov. 2012.

BUENO, A. D. **Introdução ao processamento paralelo e ao uso de clusters de workstations em sistemas GNU/LINUX:** parte I: Filosofia. Florianópolis: UFSC/LMPT, 2003. Disponível em: <http://itaporangasp.com/usp/programacao/programacao_avancada/ProgramacaoParalelaECluster-P1.pdf>. Acesso em: Acesso em: 23 fev. 2012.

CARDOSO, M. C. **MPICH-IG:** uma infra-estrutura de execução de aplicações paralelas do tipo MPI em grades computacionais oportunistas. 2008. 100 f. Dissertação (Mestrado em Ciência da Computação) -- Instituto de Informática, Universidade Federal de Goiás, Goiânia, 2008.

CARVALHO, V. F. P. de. **Otimização de um cluster de alto desempenho para o uso do programa PGENESIS em simulações biologicamente plausíveis em larga-escala de sistemas neurais.** 2007. 101 f. Dissertação (Mestrado em Ciências) --Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da USP, Ribeirão Preto, SP, 2007.

COMPRASNET: portal de compras do governo federal. **Pregão eletrônico nº 107/2011**: aquisição de materiais permanentes destinados aos Cefores. Brasília, DF: Ministério do Planejamento, 2011. Disponível em: <http://www.comprasnet.gov.br/> Acesso em: 15 mar. 2012.

COMPRASNET: portal de compras do governo federal. **Pregão eletrônico nº 54/2012**: aquisição de materiais permanentes destinados aos Cefores. Brasília, DF: Ministério do Planejamento, 2011. Disponível em: <http://www.comprasnet.gov.br/> Acesso em: 15 mar. 2012.

COQUEIRO, T. A. S. **Supercomputador de baixo custo financeiro:** implementação do cluster Beowulf "Iesam Master". 2005. 41 f. Monografia de Conclusão de Curso (Graduação em Engenharia da Computação) -- Instituto de Estudos Superiores da Amazônia, Belém, 2005.

CORRÊA, J. L.; BOMENTE FILHO, P. R. Virtualização: conceitos e princípios do SUN VirtualBox. In: CICLO DE SEMINÁRIOS ACME, 5., 2009, São José do Rio Preto. **[Palestras ...].** São José do Rio Preto, SP: UNESP, 2009. Disponível em: <http://apt.acmesecurity.org/sites/default/files/acme-seminario-virtualizacao-virtualbox-jorge-pedro.pdf>. Acesso em: 27 out. 2012.

DIAS, R. A. **Computação em grade para bioinformática**. 2006. 26 f. Dissertação (Mestrado em Ciência da Computação) -- Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2006.

FRANÇA, A. L. M. **Uma Introdução ao PVM:** como a paralelização pode ajudar a resolver problemas complexos de otimização. Campinas: UNICAMP, [20--]. Disponível em: <http://www.dsee.fee.unicamp.br/~morelato/IntroPVM.html>. Acesso em: 09 dez. 2012.

HAMMOND, B. L.; LESTER, W. A.; REYNOLDS, P. J. **Monte Carlo methods in ab initio quantum chemistry.** Singapore: World Scientific, 1994.

HISTORY of MOSIX. In: MOSIX: cluster operating system. [S.l.], 2013. Disponível em: <http://www.cs.huji.ac.il/wikis/MediaWiki/mosix/index.php/History_of_MOSIX>. Acesso em: 13 nov. 2012.

LIMA, E. C. P. et al. Simulação de Monte Carlo auxiliando a análise de viabilidade econômica de projetos. In: CONGRESSO NACIONAL DE EXCELÊNCIA EM GESTÃO, RESPONSABILIDADE SOCIOAMBIENTAL DAS ORGANIZAÇÕES BRASILEIRAS, 4., 2008, Niterói . **Anais ...** Niterói, RJ: UFF, 2008.

LIMA, W. A. D. **Planejamento estratégico no setor público.** Brasília: Escola Superior Aberta Do Brasil, 2009.

MORIKANE, C. K. **O gerenciamento de serviços de tecnologia da informação (TI) em uma instituição pública:** aplicabilidade da Norma ISO 20000 em uma instituição pública de ensino. 2008. 117 f. Dissertação (Mestrado em Gestão e Desenvolvimento Regional) -- Universidade de Taubaté, Taubaté, SP, 2008.

MOSIX: cluster operating system. **About mosix**. [S.l.], 2013. Disponível em: <http://www.mosix.org/txt_about.html>. Acesso em: 13 nov. 2012.

MOSIX: cluster operating system**. Distributions**: Older MOSIX-2 distributions for 32/64-bit can be found here. [S.l.], 2013. Disponível em: <http://www.mosix.org/mos2/index.php>. Acesso em: 13 nov. 2012.

MOSIX: cluster operating system**. Distributions**: Other MOSIX-3 distributions can be found here. [S.l.], 2013. Disponível em: <http://www.mosix.org/mos3/index.php>. Acesso em: 13 nov. 2012.

MOSIX: cluster operating system. **User's and administrator's guides and manuals**: revised for MOSIX-3.4.0.6. [S.l.], 2013. Disponível em: <http://www.mosix.org/pub/Guide.pdf>. Acesso em: 17 nov. 2012.

NAJIB, K.; AL ZAHRANI, S.; HUSSAIN, S. M. MOSIX evaluation on a linux cluster. **The International Arab Journal of Information Technology**, v. 3, n. 1, Jan. 2006. Disponível em: <http://www.ccis2k.org/iajit/PDF/vol.3,no.1/10-Najib.pdf>. Acesso em: 10 maio 2013.

PALVIA, P. C. Developing a model of the global and strategic impact of information technology. **Information & Management**, v. 32, n. 5, p. 229-244, 1997.

PASINI, R. M. **Mosix – Instalação e configuração de um cluster de balanceamento de carga**. In: VIVA o Linux. [S.l.], 2009. Disponível em: <http://www.vivaolinux.com.br/artigo/Mosix-Instalacao-e-configuracao-de-um-cluster-de-balanceamento-de-carga>. Acesso em: 12 jul. 2012.

PEREIRA, F. M. **Estudo de performance em um cluster OpenMosix**. 2010. 49 f. Monografia [Graduação em Ciências da Computação] – Faculdade Lourenço Filho, Fortaleza, 2010. Disponível em: <http://www.flf.edu.br/revista-flf/monografias-contabeis/monografia-felipe-maciel-pereira.pdf>. Acesso em: 23 fev. 2012.

PITANGA, M. Computação em cluster. In: CLUBE do Hardware. [Rio de Janeiro], 2003. Disponível em: <http://www.clubedohardware.com.br/artigos/153>. Acesso em: 23 fev. 2012.

PRADO, C. L.; SILVA, J. M. A. **Aplicação de cluster Beowulf em instituições de ensino**. 2010. 117 f. Monografia (Graduação em Tecnologia em Informática) -- Faculdade de Tecnologia de Guaratinguetá, Guaratinguetá, SP, 2010.

RENDERFARM. **Configuração – Cluster rápido Beowulf**. [S.l.], 2009. Disponível em: <https://code.google.com/p/renderfarm/downloads/detail?name=SO_Tutorial_01.pdf&can=2&q=>. Acesso em: 02 dez. 2012.

SCHMIDT, M. W. et al. General atomic and molecular electronic structure system. **J. Comput. Chem.,** New York, 14, n. 11, p. 1347-1363, 1993.

TONIDANDEL, D. A. V. **Manual de montagem de um cluster Beowulf sob a plataforma GNU/Linux.** 2008. 89 f. Monografia (Graduação em Engenharia de Controle e Automação) – Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, MG, 2008.

TRIDAPALLI, J. P.; BORINELLI, B. Gestão da cadeia de suprimento do setor público brasileiro: um estudo exploratório das funcionalidades e do nível de maturidade em governo eletrônico**.** CONGRESSO INTERNACIONAL DE ADMINISTRAÇÃO, 2010, Ponta Grossa, PR. **Anais ...** Ponta Grossa, PR: UEPG, 2010. Disponível em: <http://www.admpg.com.br/2010/down.php?id=966&q=1>. Acesso em: 17 jan. 2011.

UNIVERSIDADE FEDERAL DO TRIÂNGULO MINEIRO. **Apresentação**. Uberaba–MG, 2012. Disponível em: <www.uftm.edu.br>. Acesso em: 12 maio 2012.

VIRTUALBOX. [S.l.]: Oracle, [200-?]. Apresenta informações sobre o software virtualbox, notícias e downloads. Disponível em: <http://www.virtualbox.org>. Acesso em: 12 maio 2012.

YORIYAZ, H. Método de Monte Carlo: princípios e aplicações em física médica. **Revista Brasileira de Física Médica**, São Paulo, v. 3, n. 1, p. 141-149, 2009.

WIKIPEDIA: the free encyclopedia. **PDP-11**. [S.l.], 2013. Disponível em: <http://en.wikipedia.org/wiki/PDP-11>. Acesso em: 12 mar. 2013.

APÊNDICE A – VirtualBox

Segundo (CORRÊA, J. L., FILHO, P. R. B, 2009) o VirtualBox é uma coleção poderosa de ferramentas de máquina virtual e foi desenvolvido pela SUN Microsystems e disponibilizado sob licença GPL. É obtido a partir do site http://www.virtualbox.org/. Suas principais características são:

a) pode ser usado em computadores pessoais, servidores ou sistemas mais específicos (embarcados);
b) possibilita a virtualização de sistemas operacionais 32-bit e 64-bit, em processadores Intel e AMD;
c) pode usar recursos de hardware para virtualização (virtualização assistida em hardware) ou totalmente em software (virtualização total);
d) portabilidade: roda em vários sistemas (Linux, Windows, Mac OS e Solaris) podendo intercambiar máquinas virtuais (padrão OVF – *Open Virtualization Format*);
e) arquitetura simples: separação entre cliente e servidor. É possível iniciar a MV em GUI, controlar via CLI e acessar via RDP remotamente;
f) não é necessário o suporte a virtualização em hardware;
g) *Guest Additions*: pacote de softwares que otimiza o sistema hospedeiro;
h) suporte de hardware avançado: multiprocessamento no Guest (Sistema operacional instalado numa máquina virtual) OS (SMP), USB, ACPI, controles de vídeo, suporte iSCSI e boot PXE;
i) suporte a pontos de restauração de sistema: é possível gerar snapshots a qualquer momento e reverter o estado de uma MV;
j) acesso remoto via VRDP: protocolo RDP (desktop remoto) melhorado, como por exemplo, túnel USB;
k) sistemas hospedeiros suportados:
    - Windows:
        - Windows XP, all service packs (32-bit);
        - Windows Server 2003 (32-bit);
        - Windows Vista (32-bit e 64-bit);
        - Windows Server 2008 (32-bit e 64-bit);
        - Windows 7 beta (32-bit e 64-bit).

- Apple Mac OS X: processador Intel como requisito;
- Linux (32-bit e 64-bit), entre outros:
  - Debian GNU/Linux 3.1 ("sarge"), 4.0 ("etch") e 5.0 ("lenny");
  - Fedora Core 4 to 11;
  - Gentoo Linux;
  - Redhat Enterprise Linux 4 e 5;
  - SUSE Linux 9 and 10, openSUSE 10.3, 11.0 e 11.1;
  - Ubuntu 6.06 ("Dapper Drake"), 6.10 ("Edgy Eft"), 7.04 ("Feisty Fawn"), 7.10 ("Gutsy Gibbon"), 8.04 ("Hardy Heron"), 8.10 ("Intrepid Ibex"), 9.04 ("Jaunty Jackalope");
  - Mandriva 2007.1, 2008.0 e 2009.1.
- Solaris: Open Solaris e Solaris 10;
- FreeBSD;
- OpenBSD;
- OS/2 Warp 4.5.

l) sistemas visitantes suportados:
- Windows: NT 4.0, 2000, XP, Server 2003, Vista, Server 2008, 7 beta, DOS / Windows 3.x / 95 / 98 / ME;
- Linux: 2.4, 2.6;
- Solaris 10, OpenSolaris;
- FreeBSD;
- OpenBSD;
- OS/2 Warp 4.5.

m) o sistema que roda o VirtualBox é denominado “host” (hospedeiro);

n) o sistema virtual, instalado dentro do VirtualBox, é denominado “guest” (visitante);

o) o VirtualBox é capaz de prover virtualização de duas maneiras:
- inteiramente em software: todas as chamadas são interceptadas pelo VMM que fornece uma abstração;
- usando recursos de hardware, em processadores que suportam:
  - IntelVT-x e AMD-V;
  - Este suporte deve ser habilitado no BIOS.

p) Sobre os sistemas 64-bit como visitante:
   - se o processador for 64-bit, por padrão, pode-se instalar sistemas 64-bit como Guest OS normalmente.
     - habilitar o suporte a virtualização assistida em hardware;
     - configurar a MV para utilizar este recurso.
   - caso contrário, pode-se ainda utilizar um sistema visitante 64-bit rodando em um sistema hospedeiro 32-bit:
     - cria-se um *overhead* no VMM.
   - se as MVs forem criadas com o *Wizard*, muitas destas configurações são feitas automaticamente.

APÊNDICE B – Criação e configuração das máquinas virtuais

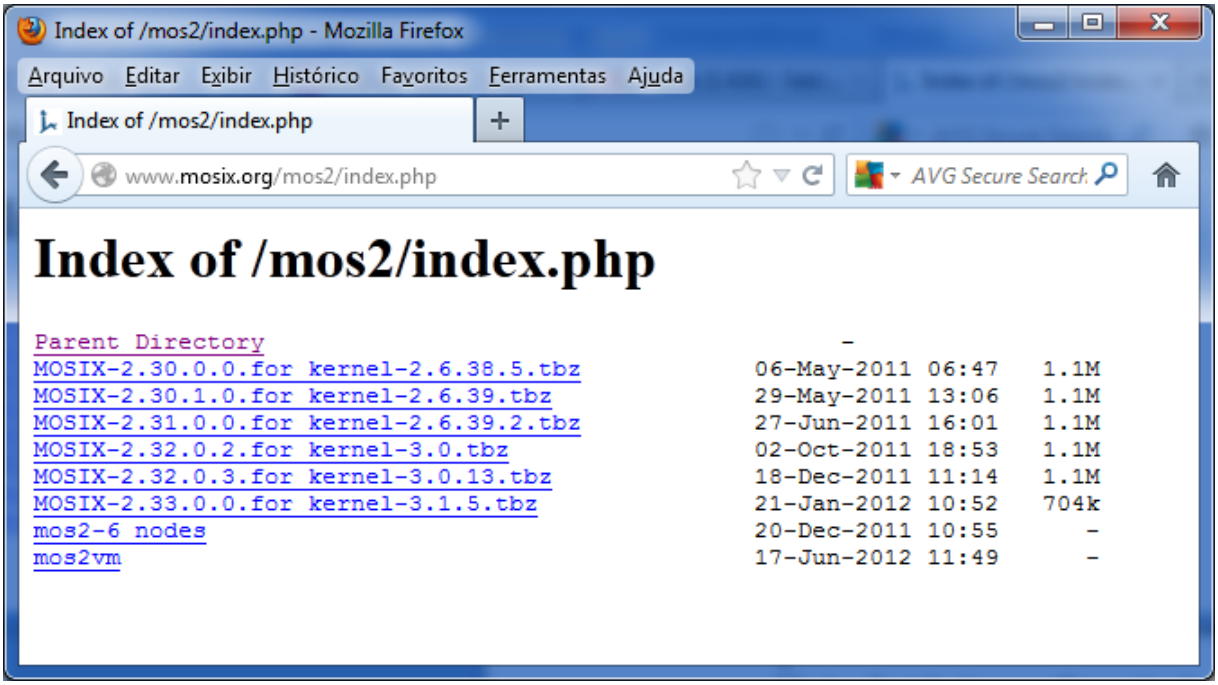
O processo de instalação é iterativo, bastando selecionar as opções listadas a seguir e clicar no botão Próximo em cada tela:

a) Selecione o botão Nova;
b) Preencha o campo Nome para identificação da máquina virtual;
c) Selecione o Sistema Operacional;
d) Selecione a versão do Sistema Operacional;
e) Defina a quantidade de memória e clique em Próximo;
f) Marque o tipo de arquivo que armazenará o disco da máquina virtual (Sugerido VMDK para otimização de espaço e criação de disco dinâmico);
g) Marque a opção Dinamicamente alocado;
h) No campo localização preencha com o caminho onde será armazenado o disco da máquina virtual ou selecione o caminho por meio da caixa de diálogo disponível com um clique no botão ao final do campo do endereço;
i) Defina o tamanho máximo do disco da máquina virtual;
j) Na tela de Sumário, clique no botão Criar para que o Virtualbox gere a máquina virtual com as configurações definidas;
k) Depois de concluída a criação, selecione a máquina virtual e clique no botão Configurações;
l) Por padrão, a máquina é criada com a placa de rede configurada como NAT para compartilhar conexão de rede via servidor DHCP do Virtualbox em conexão com a rede do Hospedeiro. Porém, como é necessário IP fixo, o ideal é a configuração da placa de rede no Modo *Bridge* possibilitando a atribuição de um IP único para a máquina em questão. Pode-se optar por configurar duas placas de rede, cada uma num modo de operação;
m) Para iniciar a instalação do sistema operacional, selecione no menu dispositivos a opção CD/DVD do hospedeiro ou USB do hospedeiro;
n) Com a mídia inserida no hospedeiro, basta Iniciar a máquina virtual e seguir os procedimentos comuns de instalação do sistema operacional na máquina virtual, da mesma forma que em uma máquina física.

APÊNDICE C – Configuração do cluster por meio do Mosix

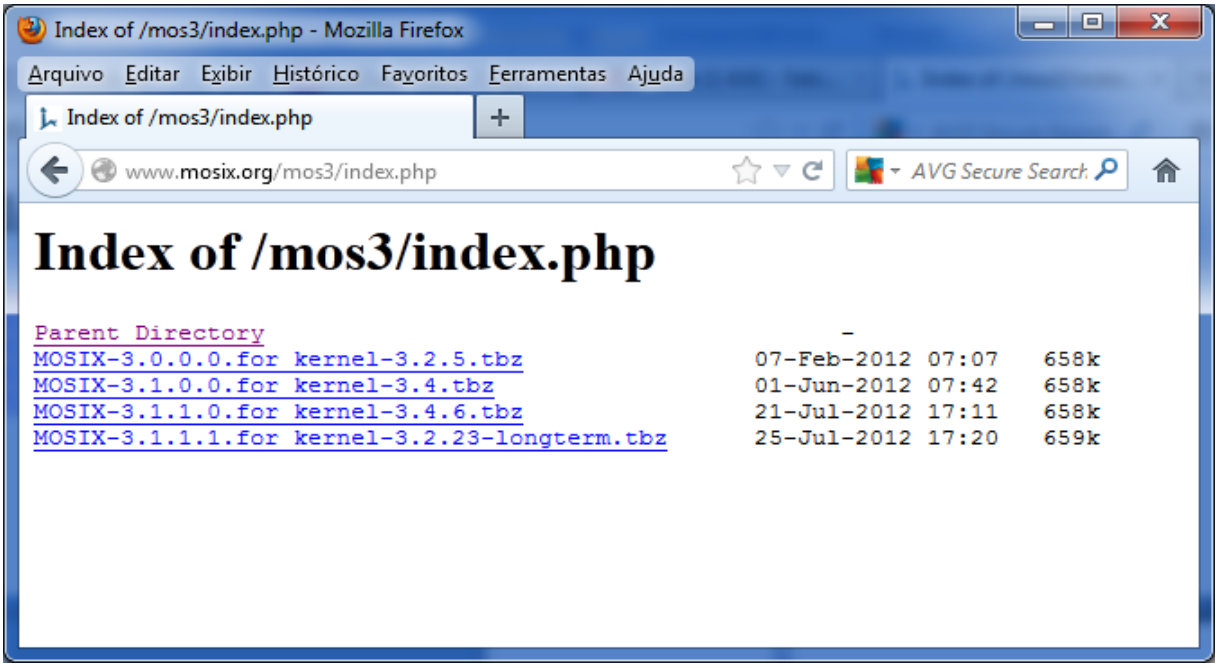
1. Acesse o sistema com o superusuário (usuário root);
   master@debian2:# su
2. Identifique a versão do Kernel
   root@debian2:/home/master/Desktop/Downloads/mosix-2.31.0.0# uname -r
   2.6.32-5-686
   root@debian2:/home/master/Desktop/Downloads/mosix-2.31.0.0# uname -a
   Linux debian2 2.6.32-5-686 #1 SMP Sun May 6 04:01:19 UTC 2012 i686 GNU/Linux
   *Neste caso, o termo i686 na resposta do comando indica sistema operacional de 32 bits.
3. Faça o download da versão do Mosix de acordo com a versão do kernel do Linux instalado a partir do site indicado nas figuras a seguir:

Figura 10: Obtenção do Mosix2 para sistemas operacionais de 32 bits



Fonte: (MOSIX, 2013b)

Figura 11: Obtenção de Mosix3 para sistemas operacionais de 64 bits



Fonte: (MOSIX, 2013c)

4. Acesse a pasta onde foi salvo o arquivo de instalação do Mosix:

root@debian2:# cd /home/master/Desktop/Downloads

*Esse caminho de diretório pode ser diferente dependendo da opção do usuário por salvar e descompactar em outro diretório.

5. Descompacte o arquivo com a versão do Mosix

root@debian2:/home/master/Desktop/Downloads/tar –xvf MOSIX-2.31.0.0.for_kernel-2.6.39.2.tbz

*Será criada uma pasta com o nome mosix-2.31.0.0 dentro desse diretório, que deve ser acessada em seguida. Os nomes listados dos pacotes de instalação ou das pastas podem ser diferentes de acordo com a versão baixada para o sistema.

6. Instale os pacotes complementares

*root@debian2:/home/master# apt-get install –f make**

*root@debian2:/home/master# apt-get install –f ncurses**

(ncurses package like kaya or libncurses)

root@debian2:/home/master# apt-get install –f fakeroot

7. A instalação automática do Mosix ocorre com as seguintes etapas:

root@debian2:/home/master/Desktop/Downloads/mosix-2.31.0.0# ./mosix.install
Installing MOSIX:
==========

If you are installing MOSIX for a set of nodes with a common root, then type the common root directory -
if you are installing only the local node, press <ENTER> :- ----> *Pressione Enter*

32-bit Installation.

Please choose how to prepare the kernel:

1. Fetch the Linux kernel (2.6.39.2) sources from the internet.
2. The Linux kernel (2.6.39.2) sources are already on my disk.
3. I already installed the kernel, or will install it separately.

Note for openSUSE users:
You do not need to compile a kernel - a ready MOSIX-kernel RPM for openSUSE is available at http://www.mosix.org/txt_eval.html:
it is best to select '3' now, then download and install it.

Option [1]: 1 --> *Opção 1 para efetuar eventuais atualizações do kernel*

--2012-11-13 00:20:57-- http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.39.2.tar.bz2
Resolvendo www.kernel.org... 149.20.20.133, 149.20.4.69
Conectando-se a www.kernel.org|149.20.20.133|:80... conectado.
A requisição HTTP foi enviada, aguardando resposta... 200 OK
Tamanho: 76096978 (73M) [application/x-bzip2]
Salvando em: “linux-2.6.39.2.tar.bz2.1”

100%[======================================>] 76096978 145K/s em 11m 31s

2012-11-13 00:32:35 (107 KB/s) - “linux-2.6.39.2.tar.bz2.1” salvo [76096978/76096978]

Extracting kernel sources from /home/master/Desktop/Downloads/mosix-2.31.0.0/linux-2.6.39.2.tar.bz2...

----> Nesta etapa será configurado o Kernel do Linux com o Mosix, devendo-se observar as seguintes situações na interface a ser exibida:

- Não alterar a opção “CONFIG_SECURITY”: ’Security Options’ / ‘enable different security models’
- Não configurar a opção “CONFIG_HEADERS_CHECK”: ’Kernel hacking’ / ‘Run ‘make headers_check’ when building vmlinux’

----> Esse é um processo demorado que lista todos os drivers e procedimentos da compilação. Caso bem sucedido, carrega automaticamente a interface de configuração do Mosix.

8. A configuração do Mosix é feita graças ao comando mosconf. Normalmente esse configurador é carregado automaticamente pelo instalador do Mosix ao final do processo de instalação.

root@debian2:/home/master/Desktop/Downloads/mosix-2.31.0.0# mosconf

MOSIX CONFIGURATION
==========

If this is your cluster's file-server and you want to configure MOSIX for a set of nodes with a common root, please type their common root directory. Otherwise, if you want to configure the node that you are running on, just press <ENTER> :-

What would you like to configure?
==========
1. Which nodes are in this cluster
2. Authentication
3. Logical node numbering
4. Queueing policies
5. Freezing policies
6. MOSIX Reach the Clouds (MRC)

7. Miscellaneous policies
8. Become part of a multi-cluster organizational Grid
9. Parameters of 'mosrun'

Configure what :- 1

Nodes in your cluster:
==========
1. 192.168.11.53
2. 192.168.11.54
3. 192.168.11.55
4. 192.168.11.56

To add a new set of nodes to your cluster, type 'n'.
To modify an entry, type its number.
To delete an entry, type 'd' followed by that entry-number (eg. d3).
To turn on advanced options, type '+'.
For help, type 'h'.
To save and exit, type 'q'. (to abandon all changes and exit, type 'Q')

Option :- n

Adding new node(s) to the cluster:

First host-name or IP address :- 192.168.11.53
Number of nodes :- 1

Nodes in your cluster:
==========
1. 192.168.11.53
2. 192.168.11.54
3. 192.168.11.55
4. 192.168.11.56

To add a new set of nodes to your cluster, type 'n'.
To modify an entry, type its number.
To delete an entry, type 'd' followed by that entry-number (eg. d4).
To turn on advanced options, type '+'.
For help, type 'h'.
To save and exit, type 'q'. (to abandon all changes and exit, type 'Q')

Option :- q

Cluster configuration was saved.

OK to also update the logical node numbers [Y/n]? Y

What would you like to configure next?
==========
1. Which nodes are in this cluster
2. Authentication
3. Logical node numbering
4. Queueing policies
5. Freezing policies
6. MOSIX Reach the Clouds (MRC)
7. Miscellaneous policies
8. Become part of a multi-cluster organizational Grid
9. Parameters of 'mosrun'
q. Exit

Configure what :- 2

MOSIX Authentication:
==========

To protect your MOSIX cluster from abuse, preventing unauthorized persons from gaining control over your computers, you need to set up a secret cluster-protection key. This key can include any characters, but must be identical throughout your cluster.

You already have a cluster-protection key:
To keep it, press <Enter>.
To change it, type the new key here :- debian
Your key is 6 characters long.

To allow your users to send batch-jobs to other nodes in the cluster, you must set up a secret batch-client key. This key can include any characters, but must match the 'batch-server' key on the node(s) that can receive batch-jobs from this node.

You already have a batch-client key:
To keep it, press <Enter>.
To delete it, type '-'.
To change it, type the new key here :- debian
Your key is 6 characters long.

For this node to accept batch jobs, you must set up a secret batch-server key. This key can include any characters, but must match the 'batch-client' key on the sending nodes.

You already have a batch-server key:
To keep it, press <Enter>.
To delete it, type '-'.
To change it, type the new key here :- debian
Your key is 6 characters long.

What would you like to configure next?
==========
1. Which nodes are in this cluster
2. Authentication
3. Logical node numbering
4. Queueing policies
5. Freezing policies
6. MOSIX Reach the Clouds (MRC)
7. Miscellaneous policies

8. Become part of a multi-cluster organizational Grid
9. Parameters of 'mosrun'
q. Exit

Configure what :- 3

Mappings of host-names/IP-addresses to MOSIX node numbers:
==========

Entry Node Host/IP # of nodes
---

1. 1 192.168.11.53 1
2. 2 192.168.11.54 1
3. 3 192.168.11.55 1
4. 4 192.168.11.56 1

To add a new mapping type 'n'.
To initialize the mappings to the nodes in this cluster, type 'i'.
To modify an entry, type its entry number.
To delete an entry, type 'd' followed by that entry-number (eg. d4).
For help, type 'h'. When finished, type 'q' (to abandon all changes type 'Q').

Option :- i

Mappings of host-names/IP-addresses to MOSIX node numbers:
==========

Entry Node Host/IP # of nodes
---

1. 1 192.168.11.53 1
2. 2 192.168.11.54 1
3. 3 192.168.11.55 1
4. 4 192.168.11.56 1

To add a new mapping type 'n'.
To initialize the mappings to the nodes in this cluster, type 'i'.

To modify an entry, type its entry number.
To delete an entry, type 'd' followed by that entry-number (eg. d4).
For help, type 'h'. When finished, type 'q' (to abandon all changes type 'Q').

Option :- q

Mappings saved.

What would you like to configure next?
==========
1. Which nodes are in this cluster
2. Authentication
3. Logical node numbering
4. Queueing policies
5. Freezing policies
6. MOSIX Reach the Clouds (MRC)
7. Miscellaneous policies
8. Become part of a multi-cluster organizational Grid
9. Parameters of 'mosrun'
q. Exit

Configure what :- 4

Queuing Configuration:
==========
Central queue manager is 192.168.11.53.

To modify the Central queue manager(s), type 'c'.
To set a default job priority, type 'p'.
User-ID's are uniform across the cluster. If not, type 's'.
To limit the maximum number of running processes, type 'm'.
To set a target number-of-processes per CPU, type 'x'.
To allow a number of priority-0 processes to run regardless, type 'z'.
To set a minimum number of jobs per-user to run regardless, type 'n'.

To enable the Fair-Share policy, type 'f'.
There are no users with a private interleave ratio. To add, type u.
The maximum number of queued jobs per user is 1000. To modify, type 't'.
For help about an option, type 'h{option}' (hc,hp,hs,hm,hx,hz,hn,hf,hu,ht).
To save and exit, type 'q'. To exit discarding all changes, type 'Q'.

Option :- q

Configuration saved.

What would you like to configure next?
==========
1. Which nodes are in this cluster
2. Authentication
3. Logical node numbering
4. Queueing policies
5. Freezing policies
6. MOSIX Reach the Clouds (MRC)
7. Miscellaneous policies
8. Become part of a multi-cluster organizational Grid
9. Parameters of 'mosrun'
q. Exit

Configure what :- 5

Freezing Policy:
==========

No automatic freezing policy is currently defined.
To create a freezing policy for a class of jobs, type the class number.

Advanced options:

Freezing to the default (/freeze) directory.

To define a freezing directory, type its path-name (starting with '/').
To perform freezing under different user-privileges, type 'u'.

For help, type 'h'. To save and exit, type 'q'. To discard changes, type 'Q'.

Option :- q

Configuration saved.

What would you like to configure next?
==========
1. Which nodes are in this cluster
2. Authentication
3. Logical node numbering
4. Queueing policies
5. Freezing policies
6. MOSIX Reach the Clouds (MRC)
7. Miscellaneous policies
8. Become part of a multi-cluster organizational Grid
9. Parameters of 'mosrun'
q. Exit

Configure what :- 6

MOSIX Reach the Clouds (MRC)
==========
The following callers are currently accepted:
        a1. 192.168.11.53 - 192.168.11.54
  Type 'a{number}' to modify accepted callers.
  Type 'da{number}' to delete accepted callers.
  Type 'a' to accept more callers.
To exclude specific callers (even if they appear above), type 'r'.
To manage access to directories, type '/'.
To manage user access, type 'u'.

To manage user-group access, type 'g'.
To create a special policy for specific callers, type 's'.
For pre-defined Top/Medium/Low security defaults, type 'T'/'M'/'L'.
Type 'h' for help; 'q' to exit; or 'Q' to exit without saving.

Option :- q

What would you like to configure next?
==========
1. Which nodes are in this cluster
2. Authentication
3. Logical node numbering
4. Queueing policies
5. Freezing policies
6. MOSIX Reach the Clouds (MRC)
7. Miscellaneous policies
8. Become part of a multi-cluster organizational Grid
9. Parameters of 'mosrun'
q. Exit

Configure what :- 7

Miscellaneous MOSIX policies:
==========
a. Processor speed is automatically detected.
b. No specific IP address is associated with this node.
c. Logging calls to 'mosrun' and migrations is disabled.
d. Storage allocation for Private-Temporary-Files (PTFs):
   Local processes use "/private", allowing by default up to 5GB per process.
   Guest processes use "/private", allowing by default up to 2GB per process.

Type <a>-<d> to modify a policy.
Type <ha>-<hd> for help about a policy.
Type 'q' to exit.

Option :- q

What would you like to configure next?

==========

1. Which nodes are in this cluster
2. Authentication
3. Logical node numbering
4. Queueing policies
5. Freezing policies
6. MOSIX Reach the Clouds (MRC)
7. Miscellaneous policies
8. Become part of a multi-cluster organizational Grid
9. Parameters of 'mosrun'

q. Exit

Configure what :- 8

There are no partners in your multi-cluster Grid yet:

==========

Please Type the name of a new partner to configure (or press <Enter> to exit) :-

What would you like to configure next?

==========

1. Which nodes are in this cluster
2. Partner clusters in your multi-cluster organizational Grid
3. Authentication
4. Logical node numbering
5. Queueing policies
6. MOSIX Reach the Clouds (MRC)
7. Freezing policies
8. Miscellaneous policies
9. Parameters of 'mosrun'

q. Exit

Configure what :- 9

Controlling the parameters of 'mosrun':
==========
This is where you can make certain parameters of 'mosrun' the default or even enforce some of them without permitting the user to override them.

* Queuing is currently the default but is not enforced.
* Selecting the best node to start on is currently the default, but is not enforced.
* Programs that attempt an unsupported system-call are currently killed (unless the user specifies the '-e' or '-w' flags).
* Memory-specification is currently not enforced.

Your options regarding queuing are to:
1. Not interfere whether or not your users use the '-q' argument.
2. Make queuing the default, but not enforce it.
3. Enforce queueing for all users.

Your choice [none/Default/enforce] :- Default

Your options regarding selecting the best node to start on are to:
1. Not interfere with the user's choice.
2. Make selecting the best node the default, but not enforce it.
3. Enforce selecting the best node on all users.

Your choice [none/Default/enforce] :- 2

Your options regarding the handling of unsupported system-calls are to:
1. Not interfere with the user's choice (by default, kill the process).
2. Make silent recovery-attempts the default.
3. Make recovery-attempts the default, with warnings to standard-error.

Your choice [Kill/proceed/warn] :- 1

Should memory-specification ('mosrun -m{mb}') be mandatory? [No/yes]:- n

Thank you, that is all: press <Enter> to continue.

What would you like to configure next?
==========
1. Which nodes are in this cluster
2. Partner clusters in your multi-cluster organizational Grid
3. Authentication
4. Logical node numbering
5. Queueing policies
6. MOSIX Reach the Clouds (MRC)
7. Freezing policies
8. Miscellaneous policies
9. Parameters of 'mosrun'
q. Exit

Configure what :- q

9. Para iniciar o Mosix, execute uma das opções:

- *root@debian2:/home/master/Desktop/Downloads/mosix-2.31.0.0# mosd*
- *root@debian2:#./etc/init.d/mosix restart*
- *root@debian2:#./etc/init.d/mosix stop root@debian2:#./etc/init.d/mosix start*
- *root@debian2:/home/master/Desktop/Downloads/mosix-2.31.0.0#reboot*

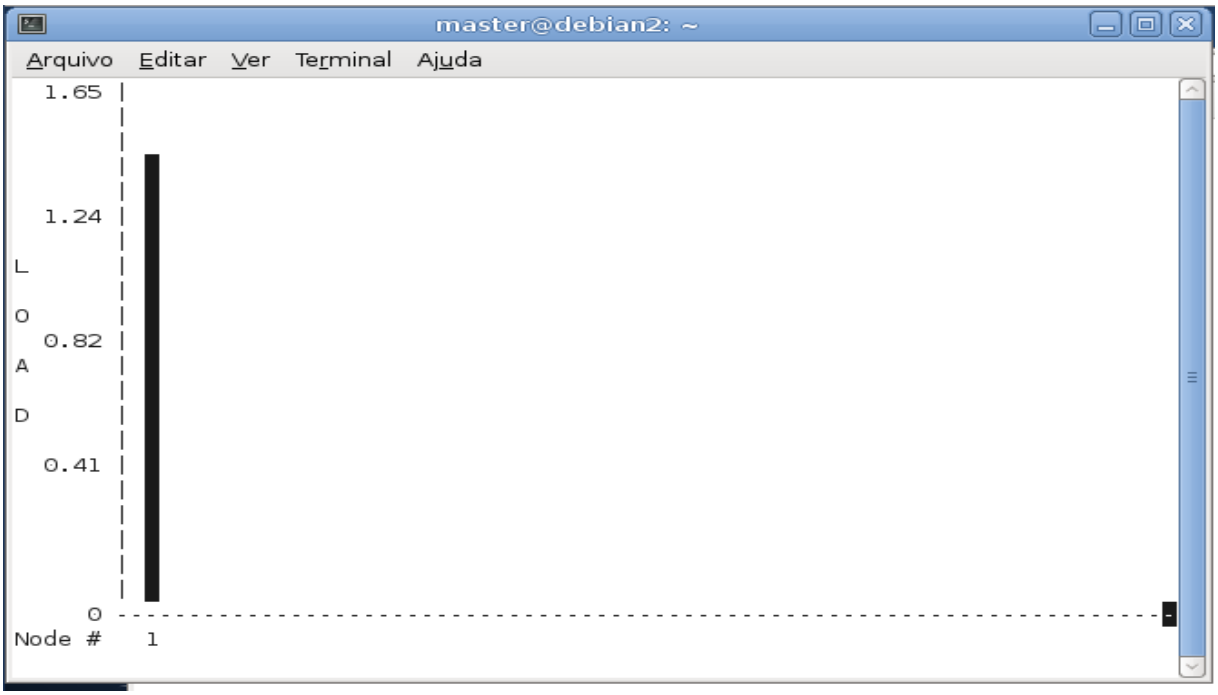
10. O teste do funcionamento do cluster pode ser feito por meio do comando mosrun, conforme os exemplos a seguir:
root@debian2:/home/master/Desktop/Downloads/mosix-2.31.0.0#mosrun testload
Mosrun: Running 'testload' as a standard Linux program!

root@debian2:/home/master/Desktop/Downloads/mosix-2.31.0.0#mosrun –e awk 'BEGIN {for(i=0;i<1000000;i++)for(j=0;j<1000000;j++)}'

11. Para visualizar o funcionamento do cluster uma interface gráfica é exibida mediante o comando *mon*.

root@debian2:/home/master/Desktop/Downloads/mosix-2.31.0.0#mon

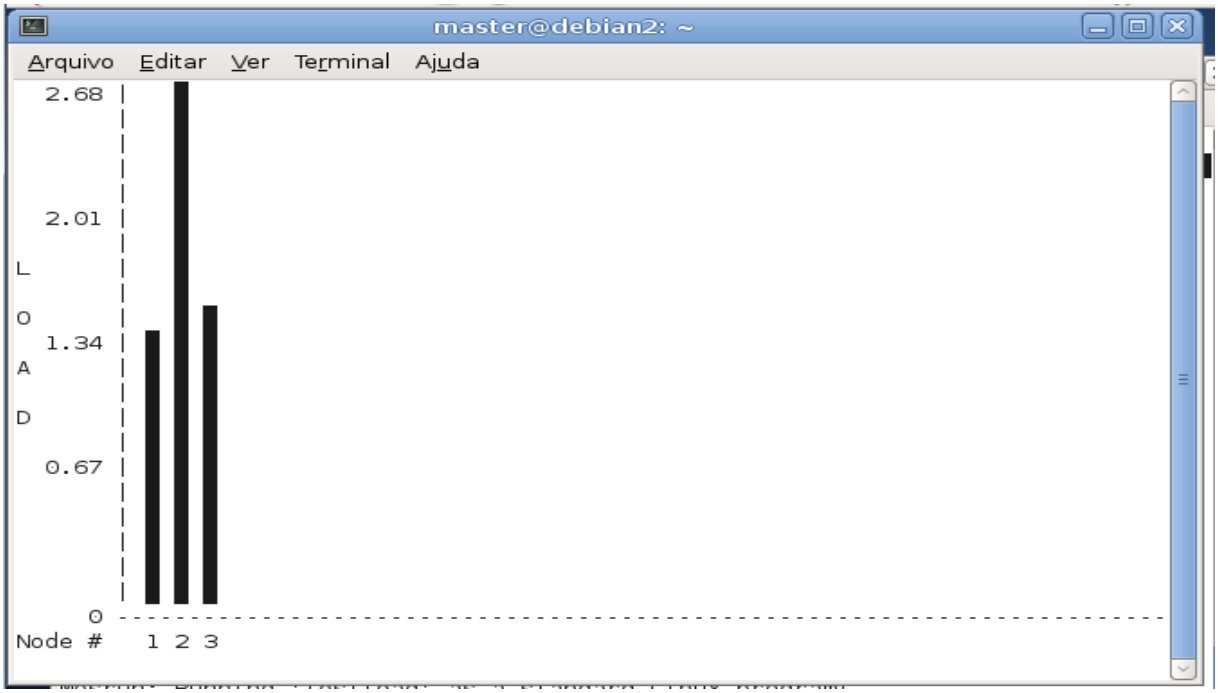
Figura 12: Visualização de performance de um nó do cluster sem interação



Fonte: Tela capturada pelo autor

Após inserir ou inicializar um novo nó deve-se verificar neste gráfico a execução do Mosix em todos os nós. Caso não tenha iniciado automaticamente, executar o comando *mosd* ou *mosixd* ou */etc/init.d/mosix restart*

Figura 13: Visualização de performance de 03 nós do cluster interagindo



Fonte: Tela capturada pelo autor

12. Caso ao executar os comandos do Mosix seja exibida a mensagem “Kernel is not MOSIX!”, significa que o kernel do Linux não foi compilado adequadamente para atender às especificações do Mosix e, para solucionar esse problema, devem ser feitos os seguintes passos para a compilação manual do kernel adequado para que o Mosix funcione perfeitamente.

Primeiro, deve-se obter um pacote essencial ao processo: o kernel-package.
 # apt-get install kernel-package

Antes da compilação, deve ser escolhido o método para configurar os parâmetros do kernel:

- Menuconfig é indicado para pessoas mais experientes que já sabem o que desejam do kernel e é apresentado em modo texto.
- Xconfig é bem mais intuitivo, dá explicações mais à mostra na tela sobre os módulos e para que servem.

Se for usar o Menuconfig, deve ser instalado pelo comando:
 # apt-get install libncurses5-dev

No caso do Xconfig:
 # apt-get install libqt3-mt-dev

Observação 1: Algumas dependências serão instaladas automaticamente.

Para escolher o kernel a ser compilado deve ser visitado o site http://kernel.org/ e, logo na entrada, o último kernel estável fica disponível, sendo que se deve buscar a versão compatível com o Mosix a ser instalado. Nesse projeto foi usada a versão 2.6.39.2 do kernel.

Para encontrar a versão correta deve ser acessada a página de download do kernel http://www.kernel.org/.../v2.6/

O download do kernel é feito clicando no link do arquivo linux-2.6.39.2.tar.bz2.

Deve ser descompactado em /usr/src com o comando:
 # tar xvjf linux-2.6.39.2.tar.bz2 -C /usr/src

Um diretório linux-2.6.39.2 é criado.

Deve ser criado um link para este diretório, para em seguida ser acessado por meio dos comandos:
$ cd /usr/src
 # ln -s linux-2.6.39.2 linux
 # cd linux

Observação 2: Para manter as configurações atuais do kernel vigente/antigo no que está prestes a compilar, faça o seguinte (senão pule para o próximo comando):
 # cp /boot/config-2[pressione TAB] .config* **TAB autocompletar*
 # make xconfig

Na tela exibida com os parâmetros do kernel clique em - File - Load - e carregue aquele arquivo de configuração antigo (.config) para carregar os drivers já configurados no kernel atual para que não carregue drivers desnecessários ou esquecer os fundamentais para o funcionamento da máquina. Salve e feche.

Deve ser criado um pacote com o novo kernel:
 # make-kpkg --initrd kernel_image

Este procedimento demora, mas ao terminar, saia do diretório atual:
$ cd ..

Instale o novo kernel por meio do pacote gerado:
 # dpkg-i kernel-image2.6.39.2.deb

Não são necessárias alterações nas configurações do gerenciador de boot, pois o Debian faz isso automaticamente. O computador deve ser reiniciado para verificar se ocorrerá algum erro.
Caso ocorra algum erro, o máximo que acontecerá é perder tempo, já que o kernel funcional continuará na máquina e o kernel com problema pode ser apagado do disco tão facilmente como apagaria um programa comum:
 # apt-get remove --purge kernel-image2.6.39.2

Também pode ser apagado o download feito e a pasta linux-2.6.39.2, pois não serão mais usados.
Após a conclusão desses procedimentos, todos os comandos do Mosix poderão ser executados adequadamente com suas funcionalidades interagindo corretamente com a versão do kernel compatível.

APÊNDICE D – Instalação e configuração de Cluster via SSH, NFS e MPICH

Para Renderfarm (2009), essa modalidade de configuração, conhecida como cluster *Beowulf*, inicialmente exige alguns pacotes que podem ser obtidos conforme descrito a seguir:

- Patch: *apt-get install patch*
- Make: *apt-get install make*
- xlibs-dev: *apt-get install xlibs-dev*
- libpng-dev: *apt-get install libpng-dev*
- Gcc: *apt-get install gcc*
- g++: *apt-get install g++*
- libstdc++2.10-dev: *apt-get install libstdc++2.10-dev*

Esta configuração utiliza como base 4 etapas:

1. Identificação dos computadores em ambiente de rede.
2. Compartilhamento dos arquivos que irão processar em cluster – NFS.
3. Configuração do acesso remoto – SSH.
4. Configuração da biblioteca de passagem de mensagem – MPICH.

Considerando que estejam configurados os computadores na rede e com o serviço SSH ativo, os demais passos para a configuração de um cluster com quatro nós estão relacionados no quadro 7.

Quadro 9: Configuração do NFS e MPICH

Ações	**Comandos**
Instalação do NFS para todos os nós Pacotes utilizados: *nfs-kernel-server* *nfs-common*	*apt-get install nfs-kernel-server* *apt-get install nfs-common*
Configuração do NFS no microcomputador debian1, considerando-o mestre caso tenha necessidade de compartilhar unidade com os demais nós do cluster Editar o arquivo */etc/exports* e acrescentar as linhas	*vi /etc/exports* */usr *(rw,sync,no_root_squash)* */lib *(rw,sync,no_root_squash)*
Inicializar ou reinicializar o serviço NFS	*cd /etc/init.d* *./nfs-kernel-server restart* *./nfs-common restart*
Montar o NFS nos nós *debian2*,	*mount mestre:/usr /usr –t nfs*

debian3 e debian4 considerando-os escravos	*mount mestre:/lib /lib –t nfs*
Configurar a montagem automática NFS para os nós escravos acrescentando as linhas	*Editar o arquivo /etc/fstab* *vi /etc/fstab* *mestre:/usr /usr nfs* *defaults 0 0* *mestre:/lib /lib nfs* *defaults 0 0*
Instalação do SSH em todos os nós para comunicação	*apt-get install ssh*
Gerar chave pública a partir do nó *debian1* para autenticação	*ssh-keygen –b 1024 –t RSA*
Copiar a chave pública para os nós e para o próprio mestre para que a utilização do *ssh* não solicite senha	*cd /root/.ssh* *scp /root/.ssh/id_rsa.pub* *debian1:/root/.ssh/authorized_keys* *scp /root/.ssh/id_rsa.pub* *debian2:/root/.ssh/authorized_keys* *scp /root/.ssh/id_rsa.pub* *debian3:/root/.ssh/authorized_keys* *scp /root/.ssh/id_rsa.pub* *debian4:/root/.ssh/authorized_keys*
Instalação do *mpich* no *debian1*	*apt-get install mpich*
Configurar no debian1 o arquivo *machines.LINUX*, responsável pelas máquinas que farão parte do cluster. Acrescentar as linhas *debian2, debian3 e debian4*. O *debian1* não precisa ser incluído nesse arquivo. Remover a linha *localhost*	*cd /usr/lib/mpich/share* *vi machines.LINUX* *debian2* *debian3* *debian4*
Alterar o arquivo */root/.bashrc* no *debian1* e acrescentar as linhas	*PATH=$PATH:/usr/lib/mpich/bin* *export PATH*

Fonte: Desenvolvido pelo autor

APÊNDICE E – Instalação e configuração do GAMESS

A seguir são listados os passos para instalação, configuração e execução do GAMESS, conforme executado nesse projeto.

```
master@debian1:~$ su
Senha:
root@debian1:/home/master# cd Downloads/gamess
root@debian1:/home/master/Downloads/gamess# ls
auxdata     ddi    INTRO.DOC  lked.out    object    rungms          vb2000
comp        ddikick.x  IRON.DOC   lked.txt    PROG.DOC  rungms1
compall     gms-files.csh  libcchem   machines    qmnuc     source
compall.log  graphics   lked       Makefile    REFS.DOC  tests
compall.out  INPUT.DOC      lked1      Makefile.in  runall    TESTS.DOC
config      install.info  lked.log   misc        runall1   tools
root@debian1:/home/master/Downloads/gamess# ./config
This script asks a few questions, depending on your computer system, to set up compiler
names, libraries, message passing libraries, and so forth.

You can quit at any time by pressing control-C, and then <return>.

Please open a second window by logging into your target machine, in case this script asks you
to 'type' a command to learn something about your system software situation.  All such extra
questions will use the word 'type' to indicate it is a command for the other window.

After the new window is open, please hit <return> to go on.

  GAMESS can compile on the following 32 bit or 64 bit machines:
axp64 – Alpha chip, native compiler, running Tru64 or Linux
cray-xt – Cray's massively parallel system, running CNL
hpux32– HP PA-RISC chips (old models only), running HP-UX
hpux64 – HP Intel or PA-RISC chips, running HP-UX
ibm32 – IBM (old models only), running AIX
ibm64 – IBM, Power3 chip or newer, running AIX or Linux
```

ibm64-sp – IBM SP parallel system, running AIX
ibm-bg – IBM Blue Gene (P or L model), these are 32 bit systems
linux32 – Linux (any 32 bit distribution), for x86 (old systems only)
linux64 – Linux (any 64 bit distribution), for x86_64 or ia64 chips
AMD/Intel chip Linux machines are sold by many companies
mac32 – Apple Mac, any chip, running OS X 10.4 or older
mac64 – Apple Mac, any chip, running OS X 10.5 or newer
sgi32 – Silicon Graphics Inc., MIPS chip only, running Irix
sgi64 – Silicon Graphics Inc., MIPS chip only, running Irix
sun32 – Sun ultraSPARC chips (old models only), running Solaris
sun64 – Sun ultraSPARC or Opteron chips, running Solaris
win32 – Windows 32-bit (Windows XP, Vista, 7, Compute Cluster, HPC Edition)
win64 – Windows 64-bit (Windows XP, Vista, 7, Compute Cluster, HPC Edition)
winazure – Windows Azure Cloud Platform running Windows 64-bit type 'uname -a' to partially clarify your computer's flavor.
please enter your target machine name: linux32

Where is the GAMESS software on your system?
A typical response might be /u1/mike/gamess, most probably the correct answer is /home/master/Downloads/gamess

GAMESS directory? [/home/master/Downloads/gamess]

Setting up GAMESS compile and link for GMS_TARGET=linux32
GAMESS software is located at GMS_PATH=/home/master/Downloads/gamess

Please provide the name of the build location.
This may be the same location as the GAMESS directory.

GAMESS build directory? [/home/master/Downloads/gamess]

Please provide a version number for the GAMESS executable.
This will be used as the middle part of the binary's name, for example: gamess.00.x

Version? [00] 01

Linux offers many choices for FORTRAN compilers, including the GNU compiler set ('g77' in old versions of Linux, or 'gfortran' in current versions), which are included for free in Unix distributions.

There are also commercial compilers, namely Intel's 'ifort', Portland Group's 'pgfortran', and Pathscale's 'pathf90'. The last two are not common, and aren't as well tested as the others.

type 'rpm -aq | grep gcc' to check on all GNU compilers, including gcc
type 'which gfortran' to look for GNU's gfortran (a very good choice),
type 'which g77' to look for GNU's g77,
type 'which ifort' to look for Intel's compiler,
type 'which pgfortran' to look for Portland Group's compiler,
type 'which pathf90' to look for Pathscale's compiler.
Please enter your choice of FORTRAN: gfortran

gfortran is very robust, so this is a wise choice.

Please type 'gfortran -dumpversion' or else 'gfortran -v' to detect the version number of your gfortran.
This reply should be a string with at least two decimal points, such as 4.1.2 or 4.6.1, or maybe even 4.4.2-12.
The reply may be labeled as a 'gcc' version, but it is really your gfortran version.
Please enter only the first decimal place, such as 4.1 or 4.6: 4.4
Alas, your version of gfortran does not support REAL*16, so relativistic integrals cannot use quadruple precision.
Other than this, everything will work properly.
hit <return> to continue to the math library setup.

Linux distributions do not include a standard math library.

There are several reasonable add-on library choices,
MKL from Intel for 32 or 64 bit Linux (very fast)

```
     ACML from AMD            for 32 or 64 bit Linux (free)
    ATLAS from www.rpmfind.net for 32 or 64 bit Linux (free)
```

and one very unreasonable option, namely 'none', which will use some slow FORTRAN routines supplied with GAMESS. Choosing 'none' will run MP2 jobs 2x slower, or CCSD(T) jobs 5x slower.

```
Some typical places (but not the only ones) to find math libraries are
Type 'ls /opt/intel/mkl'                to look for MKL
Type 'ls /opt/intel/Compiler/mkl'       to look for MKL
Type 'ls /opt/intel/composerxe/mkl'     to look for MKL
Type 'ls -d /opt/acml*'                 to look for ACML
Type 'ls -d /usr/local/acml*'           to look for ACML
Type 'ls /usr/lib/atlas'                to look for Atlas

Enter your choice of 'mkl' or 'atlas' or 'acml' or 'none': atlas
Where is your Atlas math library installed?  A likely place is
  /usr/lib/atlas
Please enter the Atlas subdirectory on your system: /usr/lib/atlas-base
Math library 'atlas' will be taken from /usr/lib/atlas-base

please hit <return> to compile the GAMESS source code activator

gfortran -o /home/master/Downloads/gamess/tools/actvte.x actvte.f
unset echo
Source code activator was successfully compiled.

Your configuration for GAMESS compilation is now in
    /home/master/Downloads/gamess/install.info
Now, please follow the directions in
    /home/master/Downloads/gamess/machines/readme.unix
root@debian1:/home/master/Downloads/gamess# more machines/readme.unix
root@debian1:/home/master/Downloads/gamess# cd ddi
root@debian1:/home/master/Downloads/gamess/ddi# ./compddi >& compddi.log
root@debian1:/home/master/Downloads/gamess/ddi# ls
```

```
compddi     compddi.out    ddikick.x          kickoff   oldddi      server  test compddi.log
data_server.x  ddi_test.x        libddi.a  readme.ddi  src
root@debian1:/home/master/Downloads/gamess/ddi# ls -lh
total 680K
-rwxr-xr-x 1 master master  45K Mar 14  2012 compddi
-rw-r--r-- 1 root   root    18K Nov  9 03:43 compddi.log
-rw-r--r-- 1 root   root    16K Set 27 20:44 compddi.out
-rwxr-xr-x 1 root   root    86K Nov  9 03:43 data_server.x
-rwxr-xr-x 1 root   root    28K Nov  9 03:43 ddikick.x
-rwxr-xr-x 1 root   root   130K Nov  9 03:43 ddi_test.x
drwxr-xr-x 2 master master 4,0K Nov  9 03:43 kickoff
-rw-r--r-- 1 root   root   191K Nov  9 03:43 libddi.a
drwxr-xr-x 4 master master 4,0K Nov 22  2011 oldddi
-rw-r--r-- 1 master master 117K Nov 22  2011 readme.ddi
drwxr-xr-x 2 master master 4,0K Nov 22  2011 server
drwxr-xr-x 2 master master 4,0K Nov  9 03:43 src
drwxr-xr-x 2 master master 4,0K Nov 22  2011 test
root@debian1:/home/master/Downloads/gamess/ddi# mv ddikick.x ..
root@debian1:/home/master/Downloads/gamess/ddi# cd ..
root@debian1:/home/master/Downloads/gamess# ./compall >& compall.log
root@debian1:/home/master/Downloads/gamess# more compall.log

------------------- done with all compilations --------------------
Sex Nov  9 04:10:58 BRST 2012
root@debian1:/home/master/Downloads/gamess# ./lked gamess 01 >& lked.log
root@debian1:/home/master/Downloads/gamess# more lked.log

Linker messages (if any) follow...

The linking of GAMESS to binary gamess.01.x was successful.
4.2u 8.8s 2:09.47 10.0% 0+0k 613672+48736io 11654pf+0w
root@debian1:/home/master/Downloads/gamess# ./runall 01 >& runall.log
```

ANEXO A – Instalação do Debian

(ALUVINO, 2008)

1– Verifique as configurações de setup (BIOS) do computador, deixando o CD_ROM como primeiro dispositivo de boot.
2– Faça um boot com CD do Debian.
3– Selecione <Escolher idioma/Choose language> e, em seguida, escolha o idioma: <Português do Brasil>.
4– Escolha <Brasil>.
5– Escolha o layout de teclado adequado ao seu hardware, ABNT2 (ç) ou Americano.
6– Será feita a detecção dos dispositivos e a configuração de rede via DHCP; se o instalador não encontrar o servidor DHCP da rede será solicitada a configuração manual.
7– Em seguida, informe o nome da máquina.
8– Forneça o nome do domínio.
9– Será iniciado o particionador de discos. Certifique-se que não há nenhum dado importante no HD em que o sistema será instalado. Selecione a opção Assistido – usar disco inteiro. Caso deseje utilizar somente uma parte do disco, selecione Manual.
10– Selecione o disco/partição.
11– Será criada uma área de troca (SWAP) e partição raíz "/" no restante do espaço do HD. Selecione **Finalizar o particionador e gravar as mudanças no disco**.
12– Clique em **sim** para prosseguir.
13– O sistema está formatando a partição para instalar o novo sistema.
14– Configurar o fuso horário, escolha São Paulo.
15– Informe a senha do Root (administrador do sistema).
16– Confirme a senha.
17– Crie um usuário.
18– Confirme o usuário.
19– Digite a senha para o usuário.
20– Confirme a senha do usuário.
21– Clique em sim para utilizar espelho de rede. No nosso caso, escolhemos **NÃO** e colocaremos o espelho depois no sources.list; siga para o **passo 24**, pois se clicar em Sim o sistema pode demorar muito, dependendo da velocidade da Internet.

22– Selecione o país do espelho de rede ou informe manualmente, caso tenha um.

23– Selecione o espelho.

24– Se existir Proxy, digite o seu endereço IP, caso contrário, clique em continuar.

25– Deixe selecionado apenas sistema básico.

26– Clique em sim para gravar uma entrada no GRUB.

27– Terminando a instalação e ejetando o CD.

28– Iniciando o Sistema.

ANEXO B – Aquisição de computador de grande porte

Quadro 10: Processo de licitação de um computador de grande porte pela UFTM

Órgão:	26254 – Universidade Federal do Triangulo Mineiro
Uasg:	153035 – Universidade Federal do Triangulo Mineiro
Data:	07 / 2012
Modalidade: 05 – Pregão	
Número da Licitação: 54/2012	Situação: Informado
CNPJ/CPF: 10.345.104/0001–91	
Razão Social/Nome: Mactecnology Comércio de Informática Ltda	
Item da licitação: 00003	Cod. do conjunto material: 256547
Identificação conjunto material: Servidor Arquivo	
Descrição detalhada do material: Marca: IBM / modelo: X3550M3 97944E8U). Prazo de entrega: 20 (vinte) dias /// Prazo de garantia: 03 (três) anos /// Validade da proposta: 60 (sessenta) dias /// Declaramos que os produtos ofertados são novos e sem uso; que nossa empresa possui recursos humanos, técnicos e materiais compatíveis com as necessidades estabelecidas no edital e que conhecemos e concordamos com todas as exigências e condições estabelecidas no edital do pregão eletrônico e documentos por nele referenciados. Nos valores propostos estão inclusos todos os custos operacionais, encargos previdenciários, trabalhistas, tributários, comerciais e quaisquer outros que incidam direta ou indiretamente no fornecimento do objeto. Mínimo 2 slots para processador. com 1 processador de mínimo 6 núcleos, frequência mínima de 2.40 GHZ e mínimo de 12MB de memória cache L3; mínimo. de 16GB mem. ram com capacidade. de expansão até mínimo de 192 GB e suporte a tecnologias de proteção avançada de memória. Advanced ECC, Rank Sparing, Memory Mirroring; 04 discos SAS de 2,5(mínimo de 7200 rpm) de 500 GB; controladora Raid, com portas SAS de 6GB com cache mínimo de 256 MB; compatibilidade com sistemas de virtualização, como Hyper V, Xen Server e Vmware; ocupar espaço físico máximo de 1U de altura em rack padrão de 19"; controladora Raid da mesma marca do fabricante do equipamento, compatível com discos rígido padrão SAS e SATA; equipamento sem sistema operacional; permitir configurações de Raid para, no mínimo dois padrões: 0, 1, 10 e 5; possuir no mínimo 2 interfaces de rede que operam em conexões UTP 10/100/1000; possuir no mínimo 4 portas USB 2.0, para conexão de dispositivos; deverá possuir unidade leitora óptica DVD-Rom; possuir fontes redundantes e	

hot plug, com potência suficiente para o funcionamento do equipamento em sua configuração máxima e operar nas faixas de tensão de entrada 200-240 VAC em 60 HZ; possuir ventiladores redundantes e hot plug, com adequados para a refrigeração do sistema interno do equipamento na sua configuração máxima e dentro dos limites de temperatura adequados para operação; possuir backplane para, no mínimo, 8 discos rígidos de 2,5"; possuir todos os acessórios para sua instalação em rack padrão de 19"; possuir, no mínimo, 2 slots de expansão do tipo PCI-Express X4/X8/X16; com mídia de inicialização e configuração do equipamento contendo todos os drives; manuais técnicos do usuário e de referencia com todas inform. do produto para instalação, configuração, operação e administração; com cabo de forca 2m tipo C13-C14; garantia 3 anos on-site com atendimento no local de instalação do equipamento, 36 meses, 24h por dia, 7 dias por semana, 4h para atendimento sem tempo de solução.	
Quantidade: 1	
Marca: IBM	Unidade: Unidade
Preço Unitário: R$ 11.950,00	Valor total: R$ 11.950,00

Fonte: (COMPRASNET, 2011b)

ANEXO C – Aquisição de microcomputadores

Quadro 11: Processo de licitação de microcomputadores pela UFTM

Órgão:	26254 – Universidade Federal do Triangulo Mineiro
Uasg:	153035 – Universidade Federal do Triangulo Mineiro
Modalidade: 05 – Pregão	
Número da Licitação: 107/2011	Situação: Informado
CNPJ/CPF: 07.315.550/0001-49	
Razão Social/Nome: Romaze Indústria e Comércio de Computadores Ltda EPP	
Item da licitação: 00021	Cod. do Conjunto Material: 150566
Identificação conjunto material: Microcomputador	
Descrição detalhada do material: Microcomputador processador núcleo duplo ou superior 2GB de memória Ram, disco rígido capacidade 500GB SATA, gravador de DVD, leitor de cartões, caixa de som teclado, mouse óptico PS2 e monitor de vídeo 18,5 polegadas (43,2cm) LCD	
Quantidade: 10	
	Unidade: Un
Preço unitário: R$ 934,00	Valor total: R$ 9340,00

Fonte: (COMPRASNET, 2011a)

ANEXO D – Arquivo de saída de um processamento via GAMESS

```
----- GAMESS execution script 'rungms' -----
This job is running on host debian2
under operating system Linux at Ter Dez 11 14:54:48 BRST 2012
Available scratch disk space (Kbyte units) at beginning of the job is
Sist. Arq.           1K-blocos      Usad Dispon.   Uso% Montado em
/dev/sda1             73791720   5959212  64084108   9% /
GAMESS temporary binary files will be written to /usr/local/src
GAMESS supplementary output files will be written to
/home/master/Downloads/gamess/scr
Copying input file exemplo.inp to your run's scratch directory...
cp exemplo.inp /usr/local/src/exemplo.F05
unset echo
/home/master/Downloads/gamess/ddikick.x
/home/master/Downloads/gamess/gamess.01.x exemplo -ddi 1 2 debian2:cpus=2 -
scr /usr/local/src

 Distributed Data Interface kickoff program.
 Initiating 2 compute processes on 1 nodes to run the following command:
 /home/master/Downloads/gamess/gamess.01.x exemplo

          ******************************************************
          *         GAMESS VERSION =  1 MAY 2012 (R1)          *
          *             FROM IOWA STATE UNIVERSITY             *
          * M.W.SCHMIDT, K.K.BALDRIDGE, J.A.BOATZ, S.T.ELBERT, *
          *   M.S.GORDON, J.H.JENSEN, S.KOSEKI, N.MATSUNAGA,   *
          *           K.A.NGUYEN, S.J.SU, T.L.WINDUS,          *
          *       TOGETHER WITH M.DUPUIS, J.A.MONTGOMERY       *
          *         J.COMPUT.CHEM.  14, 1347-1363(1993)        *
          ****************** 32 BIT LINUX VERSION ***************

  SINCE 1993, STUDENTS AND POSTDOCS WORKING AT IOWA STATE UNIVERSITY
  AND ALSO IN THEIR VARIOUS JOBS AFTER LEAVING ISU HAVE MADE IMPORTANT
  CONTRIBUTIONS TO THE CODE:
     IVANA ADAMOVIC, CHRISTINE AIKENS, YURI ALEXEEV, POOJA ARORA,
     ANDREY ASADCHEV, ROB BELL, PRADIPTA BANDYOPADHYAY, JONATHAN BENTZ,
     BRETT BODE, GALINA CHABAN, WEI CHEN, CHEOL HO CHOI, PAUL DAY,
     ALBERT DEFUSCO, TIM DUDLEY, DMITRI FEDOROV, GRAHAM FLETCHER,
     MARK FREITAG, KURT GLAESEMANN, DAN KEMP, GRANT MERRILL,
     NORIYUKI MINEZAWA, JONATHAN MULLIN, TAKESHI NAGATA,
     SEAN NEDD, HEATHER NETZLOFF, BOSILJKA NJEGIC, RYAN OLSON, MIKE PAK,
     JIM SHOEMAKER, LYUDMILA SLIPCHENKO, SAROM SOK, JIE SONG,
     TETSUYA TAKETSUGU, SIMON WEBB, SOOHAENG YOO, FEDERICO ZAHARIEV

  ADDITIONAL CODE HAS BEEN PROVIDED BY COLLABORATORS IN OTHER GROUPS:
     IOWA STATE UNIVERSITY:
          JOE IVANIC, LAIMUTIS BYTAUTAS, KLAUS RUEDENBERG
     UNIVERSITY OF TOKYO: KIMIHIKO HIRAO, TAKAHITO NAKAJIMA,
          TAKAO TSUNEDA, MUNEAKI KAMIYA, SUSUMU YANAGISAWA,
          KIYOSHI YAGI, MAHITO CHIBA, SEIKEN TOKURA, NAOAKI KAWAKAMI
     UNIVERSITY OF AARHUS: FRANK JENSEN
     UNIVERSITY OF IOWA: VISVALDAS KAIRYS, HUI LI
     NATIONAL INST. OF STANDARDS AND TECHNOLOGY: WALT STEVENS, DAVID GARMER
     UNIVERSITY OF PISA: BENEDETTA MENNUCCI, JACOPO TOMASI
     UNIVERSITY OF MEMPHIS: HENRY KURTZ, PRAKASHAN KORAMBATH
     UNIVERSITY OF ALBERTA: TOBY ZENG, MARIUSZ KLOBUKOWSKI
     UNIVERSITY OF NEW ENGLAND: MARK SPACKMAN
     MIE UNIVERSITY: HIROAKI UMEDA
     MICHIGAN STATE UNIVERSITY:
```

```
          KAROL KOWALSKI, MARTA WLOCH, JEFFREY GOUR, JESSE LUTZ, PIOTR
PIECUCH
     UNIVERSITY OF SILESIA: MONIKA MUSIAL, STANISLAW KUCHARSKI
     FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX:
          OLIVIER QUINET, BENOIT CHAMPAGNE
     UNIVERSITY OF CALIFORNIA - SANTA BARBARA: BERNARD KIRTMAN
     INSTITUTE FOR MOLECULAR SCIENCE:
          KAZUYA ISHIMURA, MICHIO KATOUDA, AND SHIGERU NAGASE
     UNIVERSITY OF NOTRE DAME: DAN CHIPMAN
     KYUSHU UNIVERSITY:
          HARUYUKI NAKANO,
          FENG LONG GU, JACEK KORCHOWIEC, MARCIN MAKOWSKI, AND YURIKO AOKI,
          HIROTOSHI MORI AND EISAKU MIYOSHI
     PENNSYLVANIA STATE UNIVERSITY:
          TZVETELIN IORDANOV, CHET SWALINA, JONATHAN SKONE,
          SHARON HAMMES-SCHIFFER
     WASEDA UNIVERSITY:
          MASATO KOBAYASHI, TOMOKO AKAMA, TSUGUKI TOUMA,
          TAKESHI YOSHIKAWA, YASUHIRO IKABATA, HIROMI NAKAI
     UNIVERSITY OF NEBRASKA:
          PEIFENG SU, DEJUN SI, NANDUN THELLAMUREGE, YALI WANG, HUI LI
     UNIVERSITY OF ZURICH:
          ROBERTO PEVERATI, KIM BALDRIDGE
     N. COPERNICUS UNIVERSITY AND JACKSON STATE UNIVERSITY:
          MARIA BARYSZ

 PARALLEL VERSION RUNNING ON     2 PROCESSORS IN     1 NODES.

 EXECUTION OF GAMESS BEGUN Tue Dec 11 14:54:49 2012
….
             -------------------------------------------
             -------   RESULTS OF PCM CALCULATION   -------
             -------------------------------------------

FREE ENERGY IN SOLVENT = <PSI| H(0)+V/2 |PSI>       =  -954.4186668781 A.U.
INTERNAL ENERGY IN SOLVENT = <PSI| H(0) |PSI>       =  -954.3974332851 A.U.
DELTA INTERNAL ENERGY =  <D-PSI| H(0) |D-PSI>       =     0.0000000000 A.U.
ELECTROSTATIC INTERACTION                           =    -0.0212335930 A.U.
PIEROTTI CAVITATION ENERGY                          =     0.0000000000 A.U.
DISPERSION FREE ENERGY                              =     0.0000000000 A.U.
REPULSION FREE ENERGY                               =     0.0000000000 A.U.
TOTAL INTERACTION (DELTA + ES + CAV + DISP + REP)   =    -0.0212335930 A.U.
TOTAL FREE ENERGY IN SOLVENT                        =  -954.4186668781 A.U.

FREE ENERGY IN SOLVENT       =     -598906.82 KCAL/MOL
INTERNAL ENERGY IN SOLVENT   =     -598893.50 KCAL/MOL
DELTA INTERNAL ENERGY        =          0.00 KCAL/MOL
ELECTROSTATIC INTERACTION    =        -13.32 KCAL/MOL
PIEROTTI CAVITATION ENERGY   =          0.00 KCAL/MOL
DISPERSION FREE ENERGY       =          0.00 KCAL/MOL
REPULSION FREE ENERGY        =          0.00 KCAL/MOL
TOTAL INTERACTION            =        -13.32 KCAL/MOL
TOTAL FREE ENERGY IN SOLVENT =     -598906.82 KCAL/MOL

           -----  ENERGY CHANGE FROM GAS PHASE TO SOLVENT  -----
              ELEC          ELEC+CAV    ELEC+CAV+DIS+REP
           -12.242           -12.242             -12.242 KCAL/MOL

-INTERNAL ENERGY IN SOLVENT-
   OMITS ES,CAV,DISP,REP (USES SOLVATED ORBITALS IN THE GAS PHASE
HAMILTONIAN)
```

```
-FREE ENERGY IN SOLVENT-
   INCLUDES ES+DISP+REP WHICH ARE IN THE SELF-CONSISTENT WAVEFUNCTION.
-TOTAL FREE ENERGY IN SOLVENT-
   INCLUDES ES+DISP+REP+CAV, MEANING CAVITATION IS NOT IN THE SCF ENERGY.

.... DONE PRINTING PCM SOLVENT SUMMARY ....
 CPU     0: STEP CPU TIME=     0.01 TOTAL CPU TIME=     1354.1 (22.6 MIN)
 TOTAL WALL CLOCK TIME=     8554.4 SECONDS, CPU UTILIZATION IS  15.83%
     2668057  WORDS OF DYNAMIC MEMORY USED
 EXECUTION OF GAMESS TERMINATED NORMALLY Tue Dec 11 17:17:23 2012
 DDI: 263224 bytes (0.3 MB / 0 MWords) used by master data server.

 ----------------------------------------
 CPU timing information for all processes
 ========================================
 0: 1248.990 + 105.126 = 1354.116
 1: 1252.322 + 103.738 = 1356.60
 ----------------------------------------
 ddikick.x: exited gracefully.
unset echo
----- accounting info -----
Files used on the master node debian2 were:
-rw-r--r-- 1 root staff     2693 Dez 11 14:54 /usr/local/src/exemplo.F05
-rw-r--r-- 1 root staff 10993920 Dez 11 17:17 /usr/local/src/exemplo.F10
-rw-r--r-- 1 root staff        8 Dez 11 16:31 /usr/local/src/exemplo.F26
-rw-r--r-- 1 root staff        0 Dez 11 14:54 /usr/local/src/exemplo.F27
ls: No match.
ls: No match.
ls: No match.
Ter Dez 11 17:17:26 BRST 2012
0.0u 0.0s 2:22:38.02 0.0% 0+0k 1360+16io 13pf+0w
```

ANEXO E – Arquivo de saída do teste com programa em Fortran

```
                          ------------------------------
                               DQMC after Variational
                          ------------------------------

Program written by Rogerio Custodio,
                   Wagner Angelotti and
                   Paulo F. B. Goncalves

Instituto de Quimica
Universidade Estadual de Campinas
13083-970 Campinas, Sao Paulo, Brazil

Last revision: February, 2010

Date: Mon Nov 12 22:36:26

Job: Ne

Elements and nuclear coordinates (a.u.)

Ne        0.0000000      0.0000000      0.0000000

Spin-orbitals:
    5 alpha spin functions
    5 beta spin functions

Linear combination of spin-orbitals

Atom   prim. l    m    n   nr        alfa
   1    1    0    0    0    1       11.42160
   1    2    0    0    0    1        8.50182
   1    3    0    0    0    2        3.56883
   1    4    0    0    0    2        2.19285
   1    5    1    0    0    2        4.68773
   1    6    0    1    0    2        4.68773
   1    7    0    0    1    2        4.68773
   1    8    1    0    0    2        2.05684
   1    9    0    1    0    2        2.05684
   1   10    0    0    1    2        2.05684

LCAO for Alpha Spin Orbitals - Open Shell Wave Function

                    1            2            3            4            5
      1           0.3864500    0.0172900    0.0000000    0.0000000    0.0000000
      2           0.6214200   -0.3122800    0.0000000    0.0000000    0.0000000
      3          -0.0015800    0.6586300   -0.0000100    0.0000000    0.0000000
      4           0.0021000    0.4341100    0.0000000    0.0000000    0.0000000
      5           0.0000000    0.0000000    0.0000000    0.3581600   -0.0853400
      6           0.0000000    0.0000000    0.3681800    0.0000000    0.0000000
      7           0.0000000    0.0000000    0.0000000    0.0853400    0.3581600
      8           0.0000000    0.0000000    0.0000000    0.6979300   -0.1662900
      9           0.0000000    0.0000000    0.7174700    0.0000000    0.0000000
     10           0.0000000    0.0000000    0.0000000    0.1662900    0.6979300

LCAO for Beta Spin Orbitals - Open Shell Wave Function

                    1            2            3            4            5
      1           0.3864500    0.0172900    0.0000000    0.0000000    0.0000000
      2           0.6214200   -0.3122800    0.0000000    0.0000000    0.0000000
      3          -0.0015800    0.6586300   -0.0000100    0.0000000    0.0000000
      4           0.0021000    0.4341100    0.0000000    0.0000000    0.0000000
      5           0.0000000    0.0000000    0.0000000    0.3581600   -0.0853400
```

```
       6           0.0000000      0.0000000      0.3681800      0.0000000      0.0000000
       7           0.0000000      0.0000000      0.0000000      0.0853400      0.3581600
       8           0.0000000      0.0000000      0.0000000      0.6979300     -0.1662900
       9           0.0000000      0.0000000      0.7174700      0.0000000      0.0000000
      10           0.0000000      0.0000000      0.0000000      0.1662900      0.6979300

------------------------------------------------------------------------
 Initial parameters for the simulation
 Number of walkers                  :       100
 Number of steps for dqmc           :     10000
 Number of steps for vqmc           :      1000
 Number of steps for thermalization :       300
 Initial time step for dqmc         :   0.00100
 Initial time step for vqmc         :   0.10000
 Desired acceptance rate for dqmc   :   0.95000
 Desired acceptance rate for vqmc   :   0.50000
 Initial seed                       : -19071968
------------------------------------------------------------------------

 Tau is optimized to produce a single acceptance rate of 0.5000 for VQMC

 Using default random number generator

------------------------------------------------------------------------

 Performing       300 thermalization steps

 Adjusted time step size :   0.06208

 Acceptance rate         :   0.53129
------------------------------------------------------------------------

 Running Difusion QMC after Variational QMC

 Running Variational QMC now
------------------------------------------------------------------------

 Average energy (a.u.)                 :         -128.511199
 Nuclear attraction energy             :         -313.391060
 Eletronic repulsion energy            :           54.096284
 Total potential energy                :         -259.294776
 Kinetic energy                        :          130.783578
 Virial theorem (v/t)                  :            1.982625
 Standard deviation                    :            0.017460
 Tau                                   :            0.062082
 Acceptance rate                       :            0.521315
 Number of walkers                     :                 100

 VQMC CPU time:         0.47256 minutes

 Running Difusion QMC now

 Performing     10000 production steps

 Initial reference energy         : -128.5111988 (u.a.)
------------------------------------------------------------------------

 Best estimate energy <e>        :  -128.9224839 a.u.      -3508.1574017 eV
 Nuclear attraction energy       :  -307.7237491 a.u.
 Eletronic repulsion energy      :    53.2990046 a.u.
 Nuclear repulsion energy        :     0.0000000 a.u.
 Total potential energy          :  -254.4247445 a.u.
```

```
 Kinetic energy                  :   125.5022605 a.u.
 Virial theorem (v/t)            :     2.0272523
 Standard deviation              :     0.8554708
 Tau                             :     0.0010000
 Effective tau                   :     0.0009794
 Acceptance rate                 :     0.9874901
 Average reference energy        :  -128.8602329 a.u.
 Number of steps                 :         10000
 Number of steps for average     :             0
 Average number of walkers       :           101

 DQMC CPU time      :        11.50199 minutes
 Branching CPU time :        10.90587 minutes

-------------------------------------------------------------------------

 QMC simulation time =        12.05742 minutes

 QMC simulation done
```