UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Projeto e Desenvolvimento de uma Ferramenta de Baixa Intrusão para Administração e Gerência de Aglomerados de Computadores

LEONARDO MARCUS RIBEIRO DA SILVA

Orientadora: Profa. Dra. Maria Stela Veludo de Paiva

Dissertação apresentada à Escola de Engenharia Elétrica de São Carlos - Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Janeiro de 2006

Ficha catalográfica preparada pela Seção de Tratamento da Informação do
Serviço de Biblioteca - EESC/USP

S586p

Silva, Leonardo Marcus Ribeiro da
Projeto e desenvolvimento de uma ferramenta de baixa intrusão para administração e gerência de aglomerados de computadores / Leonardo Marcus Ribeiro da Silva. -- São Carlos, 2006.

Dissertação (Mestrado) -- Escola de Engenharia de São Carlos-Universidade de São Paulo, 2006.
Área : Engenharia Elétrica.
Orientador: Profª. Drª. Maria Stela Veludo de Paiva.

1. Ferramenta de baixa intrusão. 2. Administração de aglomerados de computadores. 3. Sistemas distribuídos. I. Título.

FOLHA DE JULGAMENTO

Candidato: Bacharel **LEONARDO MARCUS RIBEIRO DA SILVA**

Dissertação defendida e julgada em 22-02-2006 perante a Comissão Julgadora:

Aprovado

Profa. Dra. **MARIA STELA VELUDO DE PAIVA (Orientador)**
(Escola de Engenharia de São Carlos/USP)

Aprovado

Prof. Dr. **RODRIGO FERNANDES DE MELLO**
(Instituto de Ciências Matemáticas e de Computação/USP)

Aprovado

Prof. Dr. **GONZALO TRAVIESO**
(Instituto de Física de São Carlos /USP)

Prof. Associado **HOMERO SCHIABEL**
Vice-Coordenador do Programa de Pós-Graduação
em Engenharia Elétrica em exercício.

Profa. Titular **MARIA DO CARMO CALIJURI**
Presidente da Comissão de Pós-Graduação

Dedicatória

À minha linda esposa
Claudia, ao meu filho
Pedro Henrique e à
minha grande família.

Agradecimentos

Agradeço primeiramente a Deus por me conceder essa oportunidade, à minha esposa querida e ao meu filho, à minha família pelo suporte e incentivo nas horas de dificuldades, à Professora Maria Stela pela atenção e colaboração, à Denise e Marisa da secretária da Pós Graduação pelo trabalho realizado e a todos que direta ou indireitamente contribuiram para a realização deste trabalho.

Faço um agradecimento especial ao amigo e Prof. Dr. Rodrigo Fernandes de Mello pela co-orientaçao nesse trabalho, e pela sua paciência e atenção.

Resumo

SILVA,L.M.R.(2005). Projeto e Desenvolvimento de uma Ferramenta de Baixa Intrusão para Adminstração e Gerência de Aglomerados de Computadores. Dissertação de Mestrado - Universidade de São Paulo, Escola de Engenharia de São Carlos – Departamento de Engenharia Elétrica.

Este trabalho apresenta uma ferramenta denominada FAGAC que se destina à administração e gerência de aglomerados de computadores, através de uma interface Web. A ferramenta tem a características de ser pouco intrusiva no ambiente, ou seja, consumir poucos recursos computacionais a fim de não causar atrasos na execução dos serviços e processos do sistema. Inclui também funcionalidades que geram informações para o cliente ou administrador do sistema, a respeito do estado de ocupação de memória e de CPU, monitoramento do estado da carga de cada computador, tráfego gerado na rede, espaços em disco, informações de hardware e configurações do sistema. A validação da ferramenta foi feita por meio de experimentos comparativos das cinco principais funcionalidades comums entre o FAGAC e o Ganglia, mostrando melhores resultados nas cinco funcionalidades, e que FAGAC é menos intrusivo que o Ganglia.

Palavra chave: Ferramenta de baixa intrusão, administração de aglomerados de computadores, sistemas distribuídos.

Abstract

SILVA, L.M.R.(2005). Project and Developing of Tools with Low Intrusion for Administration and Manage of Agglomerate's Computers. Master Thesis -Universidade de São Paulo, Escola de Engenharia de São Carlos – Departamento de Engenharia Elétrica.

This research presents a tool named FAGAC for cluster management and administration of agglomerated of computers, through a Web interface. This tool has the characteristic of being little intrusive in the environment, what means that it should consume a little computational resource in order to not delay the services and processes in execution at the system. The tool has functions to inform the customer or system administrator about the status of memory and CPU occupation, monitorating the load of each computer, the traffic generated in the net, disk space, hardware informations and configurations of the system. It was validated by comparing the results of the experiments from the main similar functions between FAGAC and Ganglia, showing best results for five functions tested, and that FAGAC is less intrusive than Ganglia.

Key Word: Little intrusive tool, cluster computers management, distributed computing.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Durante a década de 1980, acreditava-se que a única maneira de se construir sistemas de alto desempenho seria com o desenvolvimento de processadores mais eficientes. Esse conceito foi alterado com o advento dos sistemas paralelos que trouxeram consigo novas abordagens relacionadas a desempenho das aplicações [1].

A década de 1990 trouxe novos conceitos para a construção de sistemas computacionais de alto desempenho. Esses conceitos estão relacionados ao uso de redes de computadores e à aplicação de técnicas de paralelismo para se obter alto desempenho. Os fatores que direcionaram os estudos de alto desempenho para redes de computadores incluem: aumento da capacidade dos computadores uniprocessados, o aumento da largura de banda nas redes de computadores, o desenvolvimento de novos protocolos de comunicação, a facilidade de interconexão de computadores em redes e o alto custo das máquinas com poder de processamento paralelo [1].

Os novos conceitos introduzidos foram, então, adotados na construção de ambientes com alto poder computacional, tornando-se alternativa aos supercomputadores. A evolução desses conceitos deram origem aos sistemas distribuídos [22], os quais atualmente são utilizados para os mais diferentes tipos de aplicações.

Em sistemas distribuídos processos executam de maneira cooperativa sobre recursos computacionais interconectados via rede a fim de resolver o mesmo problema computacional. O principal objetivo desses sistemas é oferecer uma imagem única para o usuário final, que o observa como único recurso computacional [22].

Por ser uma área relativamente recente, pesquisas tem sido concentradas na solução dos diferentes problemas da área, evidenciados pela utilização desses sistemas, podendo-se destacar as seguintes sub-áreas: escalonamento de processos [31], balanceamento de carga [35, 2], protocolos de baixa latência [37], memória compartilhada distribuída [36], administração e gerenciamento desses sistemas [9].

Este trabalho enquadra-se na área de administração e gerenciamento e apresenta uma infraestrutura de software implementada para simplificar a administração e gerenciamento de computadores e processos em sistemas de alto desempenho, através de uma interface web. Apesar das diversas ferramentas existentes para esse fim, não foi encontrada na literatura ferramentas Web que possibilitem o gerenciamento dos elementos de processamento.

As próximas seções apresentam o objetivo e a estrutura desse tra-

balho.

1.1 Objetivos

O objetivo desse trabalho foi o de implementar uma ferramenta denominada FAGAC para monitoramento e gerência de aglomerados de computadores por meio de uma interface Web. A ferramenta desenvolvida possui as seguintes características:

- é pouco intrusiva no ambiente;
- contém recursos para gerar as seguites informações para o clientes ou administrador do sistema:
  - estado de ocupação de memória e CPU;
  - estado da carga de cada computador;
  - tráfego gerado na rede;
  - espaço em disco;
  - informações sobre o hardware;
  - configurações do sistema;
  - shell para administração e gerência dos elementos de processamento por meio de comando do sistema operacional linux.

1.2 Estrutura do Trabalho

Este trabalho é constituido por oito capítulos:

- no primeiro capítulo foi realizada a introdução ao assunto de maneira à situar o trabalho dentro da área e apresentando o objetivo dessa pesquisa;
- no capítulo dois são apresentados os principais tipos de aglomerados de computadores;
- no capítulo três é apresentada a evolução das redes de computadores e modelo de padronização adotado pela *ISO (International Standartization Organization)*;
- no capítulo quatro são abordados os dois principais protocolos para gerenciamento de redes de computadores, SNMP e CMIP;
- o capítulo cinco apresenta os principais trabalhos sobre gerenciamento e administração de aglomerados de computadores;
- o capítulo seis descreve as funcionalidades e metodologia adotada para desenvolvimento da ferramenta FAGAC;
- no capítulo sete descreve o procedimento para avaliação de intrusão e desempenho da ferramenta desenvolvida;
- o capítulo oito descreve as conclusões e trabalhos futuros.

Essa ferramenta deve ser agregada ao trabalho de Mello[2] para simplificar a administração e gerência de aglomerados de computadores,

contudo não se limita somente a esse domínio.

Capítulo 2

Aglomerados de Computadores e Computação Paralela

2.1 Considerações Iniciais

Este capítulo apresenta os principais tipos de aglomerados de computadores, suas funcionalidades, aplicações e características.

2.2 Tipos de Aglomerados e suas Aplicações

A palavra *cluster* refere-se a um aglomerado de computadores, ou conjunto de computadores que são utilizados para obter alta disponibilidade [33] e alto desempenho. Na maior parte dos casos, tais aglomerados são montados para a realização de tarefas que necessitam de alto poder computacional.

O interesse crescente pelos aglomerados de computadores deve-se principalmente aos seguintes fatores: boa relação custo-benefício quando comparados às máquinas paralelas, a existência de redundância física de recursos motiva o uso de técnicas de confiabilidade como por

exemplo, sistemas tolerantes a falhas e ao crescimento incremental do sistema de acordo com novas necessidades[32].

No entanto, o projeto e implementação desses sistemas são mais complexos que os programas seqüenciais. Programadores devem se preocupar com diversas linhas de controle que executam simultaneamente e concorrentemente e, além disso, com detalhes essenciais que permitam obter o maior desempenho possível[4]. Esse tipo de estrutura tem sido aplicada na resolução de problemas computacionais com muitos cálculos científicos, análises sísmicas, renderização de imagens, processamento de imagens e sinais [16].

A rede de aglomerados de computadores deve ser estruturada de forma a permitir trocas, inserções e retiradas de computadores (ou nós, ou ainda nodos) do sistema, sem haver prejuízo ou interromper seu funcionamento, simplificando a tarefa de administração. Aglomerados podem ser implementados em diferentes sistemas operacionais, com particularidades a serem resolvidas para cada sistema operacional adotado. Há uma extensa base de conhecimento para a montagem desse tipo de estrutura sobre o sistema operacional Linux [2], sendo esse sistema o pioneiro, utilizado para o construção dos *clusters Beowulf* [28].

Existem basicamente duas técnicas para a implementação de aglomerados de computadores. A primeira baseia-se no uso de soluções de software que realizam alterações no núcleo do sistema operacional. Essas soluções permitem distribuir processos legados, automaticamente

pelo sistema. Um sistema que usa essa técnica é o OpenMosix [30], o responsável por distribuir os processos. A segunda técnica baseia-se no uso de ambientes e bibliotecas de comunicação tal como *PVM* (*Parallel Virtual Machine*) e *MPI* (*Message Passing Interface*) [16].

Essas soluções de software que realizam alterações no núcleo do sistema operacional são muito utilizadas para a distribuição de aplicações legadas que não foram construídas para suportar execuções utilizando toda a capacidade de processamento dos aglomerados. Nesse tipo de solução, o objetivo é ocupar todos os recursos disponíveis no ambiente e não exatamente aumentar o desempenho de uma aplicação em particular.

O uso de ambientes e bibliotecas de comunicação permite o desenvolvimento de aplicações modeladas especificamente para executar sobre aglomerados. Essas aplicações, em geral, são construídas para utilizar todos os recursos computacionais do sistema e atingir o máximo de desempenho. Esses ambientes são integrados a outras ferramentas (tais como RSH - *Remote Shell*) para iniciar os processos, de maneira transparente, nos diversos computadores que compõem o aglomerado. Uma vez esses computadores iniciados, eles sincronizam-se e iniciam sua execução. Essa solução é indicada para fins, cujo desempenho é essencial.

Alguns tipos de aglomerados que podem ser implementados são:

- **Cluster de Alta Disponibilidade:** disponibilidade refere-se à quantidade de tempo que o sistema se mantém ativo e em condições de uso. Alta disponibilidade é característica de sistemas que se mantém acima de 99,99% do tempo em condições de uso [29]. Normalmente, esse tipo de aglomerado é usado para aplicações de missão crítica.

- **Cluster para Balanceamento de Carga:** Balanceamento de Carga refere-se à distribuição eqüitativa da carga de trabalho sobre os diversos computadores que compõem o aglomerado. Esse tipo de solução é muito usada em servidores Web. Um exemplo de técnica aplicada é com o uso do *LVS* (*Linux Virtual Server*) [15];

- **Cluster Combo:** é uma modalidade de cluster que combina características de cluster de alta disponibilidade e cluster de balanceamento de carga [15].

- **Cluster Mosix:** usa uma alteração no núcleo do sistema operacional para realizar a distribuição automática de processos. Permite a migração de processos entre computadores para atingir equilíbrio na ocupação do sistema [38].

- **Cluster Beowulf:** em 1994, Sterling e Becker [28] decidiram criar um aglomerado contendo 16 computadores pessoais para realizarem pesquisas. Cada computador continha um microprocessador 486, usando o sistema operacional Linux e uma rede padrão Ethernet. Esses pesquisadores desenvolveram, então, uma

técnica para a implementação de aglomerados denominada *Beowulf*. Nessa técnica, uniram bibliotecas para troca de mensagens (*PVM, MPI e BSP/MPI*) livres[1] e outros softwares que colaboravam no gerenciamento do sistema.

- **Cluster de Alto Desempenho:** utilizados em atividades que requerem um grande processamento, devido à complexidade e do grande volume de dados, existentes em determinadas aplicações. Exemplos: previsão meteorológica (previsão do tempo e condições climáticas), simulações geotérmicas (ou seja, simulação de eventos no solo), renderização de efeitos especiais (muito usado em filmes), simulações financeiras, etc.

2.3 Considerações Finais

Este capítulo apresentou os principais tipos de aglomerados de computadores e suas características específicas. O sistema proposto nesse trabalho de mestrado pode ser utilizado para monitoramento e gerência de qualquer um dos tipos de aglomerados apresentados.

[1] O termo livre for associado a um software está relacionado a uma licença de software livre.

Capítulo 3

Conceitos de Redes de Computadores

3.1 Considerações Iniciais

Este capítulo apresenta a evolução das redes de computadores e o modelo de padronização adotado pela *ISO (International Standartization Organization)*. Essa evolução e o surgimento de várias tecnologias, culminaram em alta heterogeneidade entre padrões de diversas tecnologias, o que levou a uma padronização, sendo a *ISO* responsável por determinar esse padrão.

3.2 Camadas do Modelo OSI

No final da década de 70, ocorreu um grande crescimento nas pesquisas e desenvolvimento de tecnologias para redes de computadores. Nesse período verificou-se a heterogeneidade de padrões entre fabricantes, o que impossibilitava a interconexão de seus produtos. Assim, com o objetivo de atingir interoperabilidade, interconectividade, portabi-

lidade de aplicações e crescimento incremental (escalabilidade) a *ISO* trabalhou no sentido de desenvolver um padrão, denominado modelo de referência *OSI* (*Open System Interconnection*)[8].

O modelo de referência *OSI* define uma arquitetura de protocolos de comunicação entre computadores. Esse modelo é composto por sete camadas, apresentadas na figura 3.1.

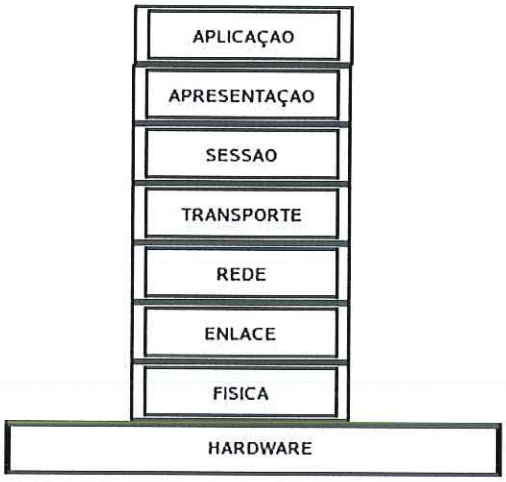

Figura 3.1: Representação do Modelo OSI

Cada camada desse modelo oferece as seguintes funcionalidades:

1. **Camada de Aplicação:** fornece ao usuário acesso ao ambiente e permite a implementação de sistemas distribuídos de informação. Nessa camada, o protocolo é composto por comandos de alto nível e por blocos de código que manipulam tais comandos.

2. **Camada de Apresentação:** provê independência da aplicação em relação às diferentes formas de representação dos dados.

3. **Camada de Sessão:** provê estrutura de controle de diálogo entre aplicações. Nessa camada, existe um protocolo que mantém o histórico da comunicação, permitindo que clientes e servidores tenham conhecimento sobre operações previamente executadas.

4. **Camada de Transporte:** responsável pela transferência de dados entre dois pontos de forma transparente e confiável com funções como controle de fluxo, correção de erros, multiplexação e demultiplexação da informação.

5. **Camada de Rede:** fornece para as camadas superiores independência das tecnologias de transmissão e comutação usadas para conectar os sistemas. Essa camada é responsável pelo endereçamento, localização de computadores e roteamento.

6. **Camada de Enlace de Dados:** responsável pela transmissão confiável de informação através do enlace físico. Envia blocos de dados (frames) com cabeçalho para controle de erro e de fluxo.

7. **Camada Física:** responsável pela transmissão de uma seqüência de bits em um meio físico. Trata das características mecânicas (especificar cabos, conectores e interfaces), elétricas (para representação de "0s"e "1s"lógicos), funcionais e procedurais para acessar o meio físico.

3.3 Modelo TCP/IP

O modelo TCP/IP surgiu da pesquisa administrada pela Agência de Projetos de Pesquisa Avançada de Defesa dos Estados Unidos denominada *DARPA*. Esse modelo foi implementado e utilizando inicialmente na interconexão entre departamentos de governos dos Estados Unidos. O objetivo era a obtenção de um meio de conexão entre universidades e instituições governamentais no caso de catástrofes, que pudessem afetar os meios de comunicação do país. O modelo TCP/IP é constituído de quatro camadas: aplicação, transporte, rede e acesso à rede. A figura 3.2 apresenta a estrutura de organização das camadas e exemplos de protocolos nelas contidos.

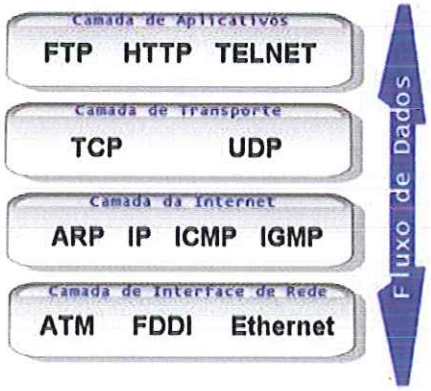

Figura 3.2: Modelo TCP/IP

1. **Camada de Aplicação:** essa camada não tem um padrão comum, cada aplicação define seu padrão tal como: o *FTP, o TELNET, o SNMP e o HTTP.* É na camada de aplicação que se estabelece o tratamento das diferenças entre a representação dos

formatos de dados. Essa camada usa a infra-estrutura de portas de comunicação provida pela camada de transporte.

2. **Camada de Transporte:** nesta camada, faz-se o controle da comunicação entre as aplicações. A camada de transporte utiliza dois protocolos: o TCP (*Transmission Control Protocol*) e o UDP (*User Datagram Protocol*). O primeiro é orientado à conexão, confiável e tem controle de congestionamento. O segundo não apresenta essas características, apenas permite o uso direto da camada de rede. Ambos podem servir a mais de uma aplicação simultaneamente. O acesso das aplicações à camada de transporte é feito através de portas. A maneira como a camada de transporte transmite dados das várias aplicações simultâneas é por intermédio da multiplexação, cujas várias mensagens são repassadas para a camada de rede (especificamente ao protocolo IP) que se encarrega de empacotá-las e enviá-las para uma ou mais interface de rede. Chegando ao destinatário, o protocolo IP repassa para a camada de transporte que demultiplexa para portas (aplicações) destino.

3. **Camada de Rede:** A camada de rede é responsável pelo endereçamento, roteamento dos pacotes, controle de envio e recepção (erros, bufferização, fragmentação, seqüência, reconhecimento). Essa camada não é orientada à conexão.

4. **Camada de Acesso à Rede:** também denominada camada de abstração de hardware, tem como função principal a interface

com os diversos tipos de redes (*X.25, ATM, FDDI, Ethernet, Token Ring, Frame Relay*, sistema de conexão ponto-a-ponto). Dada a existência de uma grande variedade de tecnologias de rede, que utilizam diferentes velocidades, protocolos e meios de transmissão, essa camada não é normatizada pelo modelo, o que provê uma das grandes vantagens do modelo TCP/IP: a possibilidade de interconexão e interoperação de redes heterogêneas.

3.4 Considerações Finais

Este capítulo apresentou a arquitetura de protocolos de comunicação entre computadores, adotada para a padronização de diversas tecnologias de redes pela *ISO*. As sete camadas do modelo *ISO* foram descritas com suas principais características.

Capítulo 4

Gerência de Redes de Computadores

4.1 Considerações Iniciais

Devido à grande evolução das redes de computadores e da heterogeneidade de tecnologias adotadas, ferramentas e protocolos de gerenciamento foram desenvolvidos para administrar e gerenciar redes. Este capítulo apresenta características de atuação desses protocolos e aborda os dois principais protocolos para o gerenciamento de redes de computadores o *SNMP* e *CMIP*.

4.2 Áreas de Atuação da Gerência de Redes de Computadores

No final da década de 70 e início da década de 80, as redes de computadores deixaram de ser aplicadas somente com o objetivo de estudo, para serem utilizadas na interconexão de empresas e demais instituições. A integração de redes de computadores formando maiores

redes foi denominada internet.

A popularização do uso da tecnologia e interconexão de redes de menor porte para formar internets gerou novas necessidades tais como técnicas de gerenciamento de redes, monitoramento de serviços de software e recursos de hardware. A complexidade para o desenvolvimento dessas técnicas encontra-se intimamente relacionada ao crescimento incremental das internets. Surgiram, então, algumas abordagens que visavam automatizar esses processos tais como os protocolos de gerenciamento de redes.

Com o desenvolvimento contínuo dos protocolos de gerenciamento de redes, algumas instituições desenvolveram padrões. Segundo a *ISO* (*International Standartization Organization*), esses protocolos atuam em cinco áreas distintas, que são:

1. Gerenciamento de falhas;
2. Gerenciamento de configuração;
3. Gerenciamentode segurança;
4. Gerenciamento de desempenho;
5. Gerenciamento de contabilização.

Essas áreas vão ao encontro das necessidades dos usuários do sistema, que são: um sistema seguro, interface amigável e facilidade de implantação. As áreas anteriormente mencionadas são descritas a seguir.

- **Gerenciamento de Falhas:** uma falha é uma condição anormal de operação do sistema, cuja operação de restauração exige um processo de gerenciamento. Geralmente, tem como principal ocorrência operações incorretas ou erros sucessivos. Dentre as técnicas utilizadas para minimizar falhas, encontram-se o uso de componentes de redundância, rotas alternativas de comunicação e planos de contingência para restauração do sistema. Esses planos devem colocar o sistema em estado de uso o mais rápido possível e com menor prejuízo possível.

- **Gerenciamento de Configuração:** visa a ativação, desativação e avaliação do estado dos serviços de software de cada computador na rede. Essas operações não devem afetar outros serviços ativos. Configurações são, geralmente, necessárias para atualização, recuperação de falha ou testes de segurança.

- **Gerenciamento de Segurança:** realiza a coleta e o armazenamento de registros de informações do sistema para fins de auditoria. Essa área de gerenciamento inclui o uso de políticas de manutenção e trocas de senhas de acesso, geração e distribuição de chaves criptografadas.

- **Gerenciamento de Desempenho:** monitora os serviços e recursos de hardware da rede visando verificar o tempo de resposta de aplicativos, canais de comunicação e a existência de gargalos no sistema de comunicação.

- **Gerenciamento de Contabilização:** evita o monopólio de recursos da rede para benefício de um único usuário ou grupo de usuários. Essa técnica visa conhecer os hábitos dos usuários do sistema, com o objetivo de planejar o crescimento na infra-estrutura de software e hardware.

As áreas apresentadas permitem definir o gerenciamento de redes como o processo de controle dos aspectos lógicos (serviços de software) e físicos (hardware) de uma rede de computadores, de forma a obter maior eficiência e produtividade.

4.3 Protocolos para o Gerenciamento de Redes de Computadores

Dentro da pilha de protocolos TCP/IP, dois protocolos de gerenciamento de redes são oferecidos. O *SNMP* (*Simple Network Management Protocol*) e o *CMIP* (*Common Management Information*). Esses protocolos permitem o monitoramento e avaliação dos serviços em execução nos computadores, obtenção de informações de hardware e aspectos de configuração. As seções 4.4 e 4.5 apresentam detalhes e características específicas desses protocolos.

4.4 *Simple Network Management Protocol*

Com o advento das grandes redes de computadores, com estruturas mais complexas e de diferentes tipos, surgiu a proposta de um pro-

tocolo, denominado *SNMP* (*Simple Network Management Protocol*), com o objetivo de gerenciá-las. Esse protocolo surgiu em meados dos anos 80, e sua principal proposta foi a de uma solução emergencial, pois até então nenhum sistema ou protocolo de gerenciamento de redes existia. Esse protocolo, atua na camada de aplicação da pilha de protocolos TCP/IP, tendo sido definido e projetado pela IETF (*Internet Engineering Task Force*).

Versões do Protocolo *SNMP*

Sua simplicidade, flexibilidade e popularidade fizeram com que esse protocolo fosse disseminado tonando-se uma referência na área de gerência de redes. A segunda versão do *SNMP (SNMPv.2)* buscou corrigir algumas deficiências da primeira versão. Essa versão surgiu basicamente da evolução do *SNMPv.1 e do RMON* (*Remote Monitoring*). Além disto, a segunda versão permite a criação e exclusão de objetos, juntamente com a comunicação entre gerentes através da chamada *Manager to Manager MIB* (conjunto de variáveis usadas para representar informações estáticas ou dinâmicas vinculadas a um determinado dispositivo gerenciável). As mensagens e a segurança foram melhor implementadas. Nas versões denominadas *SNMPv2u e SNMPv2**, existem níveis de segurança por usuário (*user-based*), o que só permite a realização de operações por usuários específicos. A grande vantagem oferecida é a possibilidade de monitoração das camadas de rede à aplicação da pilha de protocolos.

A terceira versão do *SNMP (SNMPv.3)* trouxe como principais

vantagens aspectos ligados à segurança. Nesse caso a segurança busca evitar a alteração das mensagens enviadas e bloquear o acesso para a execução de operações de controle, que são realizadas através da primitiva *SetRequest*. Evita-se também a leitura das mensagens por parte de pessoas não autorizadas, além de se garantir ao gerente o direito de alteração da senha dos agentes. A segurança é conseguida através da introdução de mecanismos de criptografia com o *DES* (*Data Encryption Standard*) e de algoritmos de autenticação que podem ser tanto o *MD5* quanto o *SHA* (*Secure Hash Algorithm*).

Após o *SNMP*, surgiram outros protocolos com restrições de configuração, desempenho e facilidade de implementação. Dessa forma, o protocolo manteve-se no mercado e tornou-se uma referência. Atualmente, os principais equipamentos e sistemas operacionais suportam o protocolo *SNMP*.

O *SNMP* permite a troca de mensagens entre agentes e gerentes. Essas informações são trocadas ou transportadas pelo *SNMP* permitindo aos administradores da rede gerenciar o desempenho da rede, configurações de serviços e hardware, obtenção de informações do sistema de forma remota. Assim os profissionais podem encontrar as soluções mais adequadas, diagnosticar e planejar melhorias. Os padrões que definem a estrutura de gerenciamento de redes internet são descritos nos documentos RFC-1155 (*Structure of Management Information*), RFC-1156 (*Management Information Base*), RFC-1157 (*Simples Network Management Protocol*)[7].

4.4.1 Agente *SNMP*

O agente (a figura 4.1) é uma aplicação que recebe requisições de um aplicação gerente, pertencente a uma determinada comunidade de gerenciamento. O agente verifica se as requisições são válidas e envia respostas apropriadas. A aplicação agente utiliza rotinas de instrumentação que manipulam estruturas de dados locais para recuperar ou resgatar informações de vários objetos denominados *MIBs* (*Management Information Bases*).

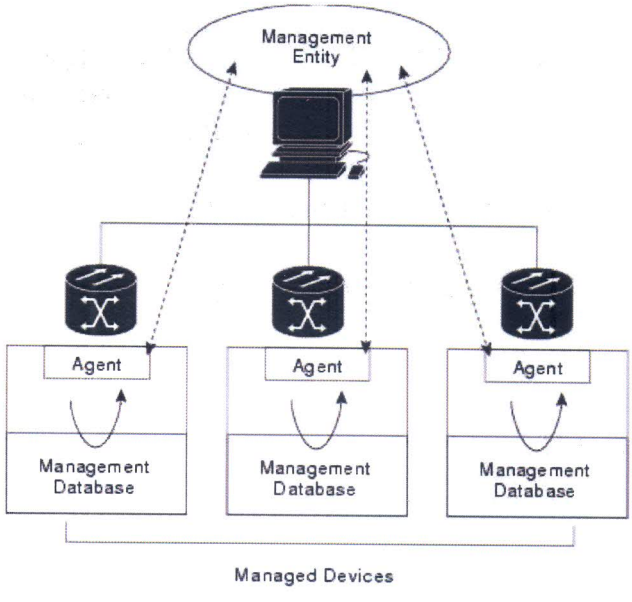

Figura 4.1: Modelo Agentes-Gerentes - *SNMP* [11]

4.4.2 Gerente *SNMP*

A aplicação gerente (figura 4.2) utiliza protocolos da camada de acesso à rede, da camada de rede (IP), e da camada de transporte, além do próprio protocolo *SNMP* (localizado na camada de aplicação) para

a comunicação com agentes. A aplicação gerente realiza operações simples de leitura e escrita de variáveis gerenciadas pelo agente. O gerente usa o esquema de segurança provido pelo agente. Esse esquema requer a identificação ou senha denominada *community*, que o gerente deve informar no início da transação.

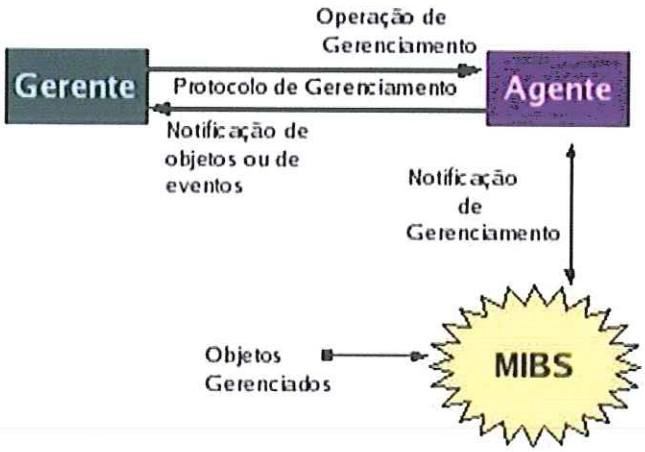

Figura 4.2: Modelo de trocas de mensagens entre Agentes e Gerentes

O protoloco usado pelo *SNMP* na camada de transporte é o *UDP* (*User Datagram Protocol*)[11]. *UDP* é geralmente usado por programas que transmitem pequenas quantidades de dados sem a necessidade de garantias de confiabilidade e com alto desempenho. Nessas situações, a baixa sobrecarga do *UDP* (pois esse não faz verificações realizadas pelo TCP) e a capacidade de *broadcast* do *UDP* (envio do mesmo datagrama a vários destinatários) são adequadas. Assim, o campo *Request ID da PDU* é usado pela entidade gerenciadora para numerar as requisições que faz a um agente; *Request Identification* é usado pela entidade gerenciadora, para detectar requisições e respostas

perdidas.

A entidade gerenciadora é quem decide se retransmite uma requisição caso nenhuma resposta correspondente for recebida após um dado intervalo de tempo. O protocolo *SNMP* em particular não obriga nenhum procedimento específico de retransmissão.

As mensagens UDP são encapsuladas e enviadas em datagramas IP, conforme apresentado na figura 4.3.

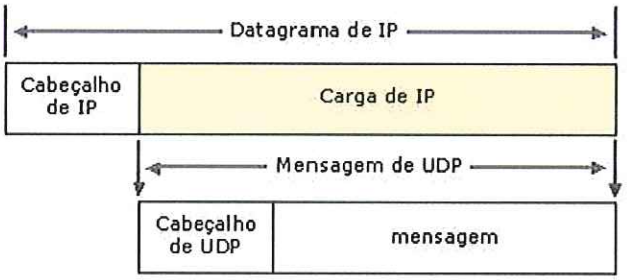

Figura 4.3: Mensagens UDP em pacotes IP [10]

- **Vantagens do UDP:** quando a quantidade de dados a serem transmitidos é pequena, o cabeçalho e a garantia de confiabilidade da entrega podem ser maiores que o trabalho de retransmissão de todo o conjunto de dados. Nesse caso, o UDP é a melhor escolha de protocolo da camada de transporte. Aplicações que encaixam em um modelo de pergunta-resposta [32] também são fortes candidatas a usarem UDP. A resposta pode ser usada como reconhecimento positivo para a pergunta. Se uma resposta não chega em num período de tempo estipulado (estouro de *timeout*), a aplicação envia outra pergunta.

- **Desvantagens do UDP:** não fornece garantia de entrega e nem de verificação de dados. Essa desvantagem, por outro lado agrega maior desempenho para esse protocolo, tornando-o mais rápido por não ter a sobrecarga de verificação dos dados.

A tabela 4.1 apresenta um comparativo entre o UDP e o TCP.

Tabela 4.1: Comparativo entre UDP e TCP

UDP	TCP
Serviço sem conexão; nenhuma sessão é estabelecida entre os hosts.	Serviço orientado por conexão;uma sessão é estabelecida entre os hosts.
UDP não garante ou confirma a entrega ou seqüência os dados.	TCP garante a entrega através do uso de confirmações e entrega seqüenciada dos dados.
Os programas que usam UDP são responsáveis por oferecer confiabilidade necessária ao transporte de dados	Os programas que usam TCP têm garantia de transporte confiável de dados.
UDP é rápido, tem baixa sobrecarga de verificação e pode oferecer suporte à comunicação ponto a ponto e ponto a vários pontos.	TCP é mais lento, tem de maior sobrecarga de verificação e pode oferecer suporte apenas à comunicação ponto a ponto.

4.4.3 Base de Informações de Gerenciamento - MIB

A *MIB* (*Management Information Base*) compreende um conjunto de variáveis usadas para representar informações estáticas ou dinâmicas vinculadas a um determinado dispositivo gerenciado. Grande parte das funcionalidades de um software – gerente/agente destinam-se à troca de dados existentes na base de informações de gerenciamento.

A figura 4.4 apresenta a estrutura organizacional das MIBS.

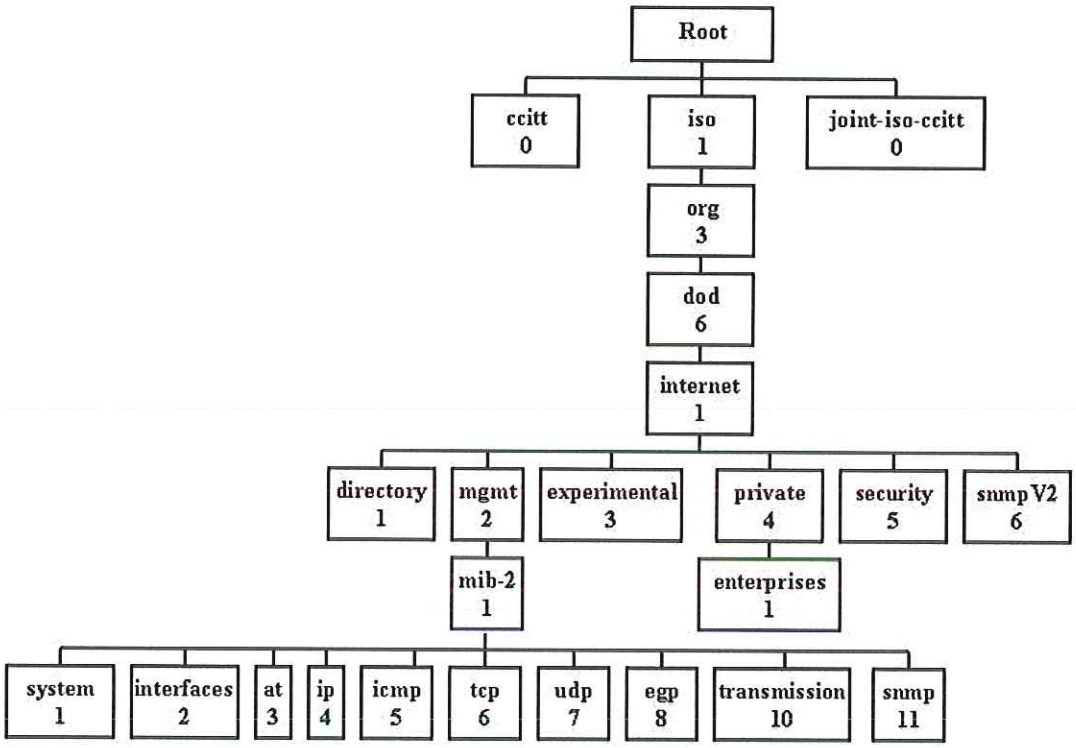

Figura 4.4: Modelo Esquemático da Organização das MIBS

4.4.4 Vantagens da Utilização do Protocolo *SNMP*

A vantagem mais expressiva do protocolo *SNMP* é sua simplicidade de uso aliada à facilidade de implementação em redes de grande porte. Mesmo com sua simplicidade, pode-se executar tarefas disponíveis no protocolo *CMIP*, de maior complexidade, tal como a implementação e a programação de variáveis a serem monitoradas.

De maneira geral as informações providas pelas MIBS consistem em:

1. Título da variável;
2. Tipo de dado da variável (por exemplo: inteiro, string);
3. Acesso da variável (leitura apenas ou de leitura/escrita);
4. Valor da variável.

Outra vantagem significativa do *SNMP* é sua ampla aplicação, que naturalmente gera uma maior base de conhecimento técnico disponível. Essa popularidade foi concretizada a partir do momento que nenhum outro protocolo de gerência de redes o substituiu.

Outra característica positiva desse protocolo é o grande número de fabricantes que visam padronizar seus produtos de hardware para serem gerenciados com *SNMP*.

4.4.5 Desvantagens do Protocolo *SNMP*

A principal deficiência do *SNMP* está relacionada à segurança, pois não tem um bloqueio bem definido para o acesso de informações por parte de intrusos. A segunda versão desse protocolo (*SNMP v2*) agregou mecanismos e políticas de segurança para tratar as limitações de: privacidade dos dados, impedindo que intrusos ganhem acesso à informação transmitida ao longo da rede; autenticação, para impedir que os intrusos emitam dados falsos através da rede e possam assim invadir o sistema; controle de acesso, que restringe o acesso de variáveis particulares a certos usuários.

A simplicidade deste protocolo faz com que sua capacidade seja posta à prova, pois muitos acreditam que as informações obtidas não sejam confiáveis. Esse problema também foi reparado na segunda versão do *SNMP*, a qual permite a especificação de variáveis com maiores detalhes, incluindo o uso de estrutura de dados de tabela para uma recuperação mais simples.

4.5 *Common Management Information Protocol*

O *CMIP* (*Common Management Information Protocol*) é um protocolo de gerenciamento de redes de computadores, projetado para substituir o *SNMP* no fim dos anos 80. Esse projeto foi financiado por governos e empresas [9]. E tinha como objetivo o desenvolvimento de um protocolo robusto com maior número de funcionalidades. Sua estru-

tura é similar ao *SNMP*, que por meio de PDU's (*Protocol Data Unit*) são empregadas variáveis para monitorar uma rede. *CMIP* contém onze tipos de PDU's e *SNMP*, cinco [9].

Para a manipulação de dados gerenciais, além da *MIB, CMIS e do CMIP* também são definidos recursos adicionais, que permitem selecionar o grupo de objetos, sobre os quais se aplica uma determinada operação. As variáveis são vistas como estruturas de dados muito complexas e sofisticadas, compostas por: 1) atributos da variável - que representam seus tipos de dados; 2) comportamentos da variável - alertas associados à variável; 3) notificações - a variável pode gerar um relatório de ocorrências de eventos (por exemplo, um encerramento de terminal causaria um evento de notificação da variável).

4.5.1 Vantagens do Protocolo *CMIP*

A principal contribuição do protocolo *CMIP* é que ele é integrado a outras ferramentas de software que permitem utilizar as informações de gerenciamento da rede. No caso do *SNMP* isso é possível, contudo, demanda a criação de alguns scripts e modificações em aplicações legadas.

O *CMIP* monitora recursos de maneira periódica, detecta falhas e encaminha alarmes aos administradores do sistema. No caso do *SNMP*, o administrador deve executar cada operação de monitoramento e avaliar seus resultados. Dessa forma, o *CMIP* minimiza esforços do responsável na administração do ambiente.

Outro fator que agrega vantagens ao *CMIP* é a implementação de melhorias no *SNMP*, incorporando suporte a dispositivos de gerência de segurança. Tais dispositivos suportam o controle de acesso e registros de segurança.

4.5.2 Desvantagens do *CMIP*

A principal desvantagem do *CMIP* é necessitar de maior quantidade de recursos para execução, quando comparado ao *SNMP*. Há possibilidades de minimizar esse custo, alterando suas especificações e diminuindo suas funcionalidades.

4.6 Considerações Finais

Este capítulo apresentou as principais áreas de atuação dos protocolos de gerência de redes de computadores e os dois principais protocolos de gerência (*SNMP e CMIP*), esclarecendo suas vantagens e desvantagens. Esse estudo permitiu a escolha do protocolo *Simple Network Management Protocol - SNMP*, para ser utilizado nesse trabalho, com o objetivo de resgatar informações e também por não requerer recursos excessivos dos elementos de processamento a serem monitorados.

Capítulo 5

Sistema de Monitoramento de Aglomerados de Computadores

5.1 Considerações Iniciais

Este capítulo apresenta os trabalhos mais relevantes para a gerência e monitoramento de aglomerados de computadores, os quais foram utilizados como base de conhecimento para o desenvolvimento deste trabalho.

5.2 Ganglia

Ganglia é um sistema de monitoramento distribuído e escalável para sistemas de computação de alto desempenho, tais como aglomerados de computadores e grades computacionais. O Ganglia foi desenvolvido para suportar diferentes sistemas operacionais e arquiteturas de hardware [18].

O Ganglia armazena dados obtidos dos elementos de processamento em bases *RRDtool* (*Round Robin Database*). Tal sistema permite o armazenamento de seqüências temporais de dados de forma compacta em um banco de dados circular. Além disso *RRDtool* gera gráficos dos dados armazenados dos elementos de processamento, mostrando a evolução de uma ou mais métricas.

Pode-se classificar a implementação do Ganglia em duas partes ou dois *daemons* básicos: uma estrutura receptora de informações recebidas dos agentes denominada *front-end*, desenvolvida em linguagem PHP e outra composta por pequenos utilitários. Entre esses utilitários estão o *Ganglia Monitoring Daemon (gmod) e o Ganglia Meta Daemon (gmetad).*

1. *Ganglia Monitoring Daemon (gmod)*: é um *daemon* multitarefa que executa em cada um dos computadores que se deseja monitorar. Esse *daemon* utiliza uma base de dados redundante e proprietária, denominada *Gmod*, para armazenar informações localmente coletadas. O *daemon* apresenta quatro características principais:

   (a) monitora mudanças de estado dos computadores do aglomerado;

   (b) escuta requisições de todos os elementos de processamento do cluster seja esse por meio de *unicast* ou *multicast*;

   (c) monitora os computadores através de comunicação em grupo (*multicast*);

(d) gera um arquivo XML com informações de estado dos computadores [18].

2. *Ganglia Meta Daemon (gmetad)*: é um *daemon* que cria uma topologia virtual de conexões, na forma de árvore, entre cada computador do sistema. Um computador no nível n dessa árvore, periodicamente, recolhe dados de seus filhos (computadores no nível $n + 1$). Os dados são centralizados na raiz da árvore e, posteriormente, enviados a um servidor para auxiliar na administração.[18].

3. *Ganglia PHP WEB FrontEnd*: esse *daemon* apresenta informações recolhidas pelo *gmod* e *gmetad* na forma de gráficos na Web. Esse *daemon* é implementado através de páginas Web dinâmicas, que apresentam informações atualizadas do sistema. Algumas das informações apresentadas são: carga de trabalho de cada computador, utilização de recursos de memória, processos em execução e estatísticas do uso da rede [18].

A figura 5.1 ilustra a interface Web de Ganglia para visualização das métricas monitoradas. Na parte superior dessa interface observa-se um menu de opções que pode selecionar a métrica e a granulosidade de tempo a ser analisada. Na parte central estão as métricas globais do sistema e, na parte inferior, a visualização dos estados de cada recurso computacional com relação a uma métrica escolhida no menu superior.

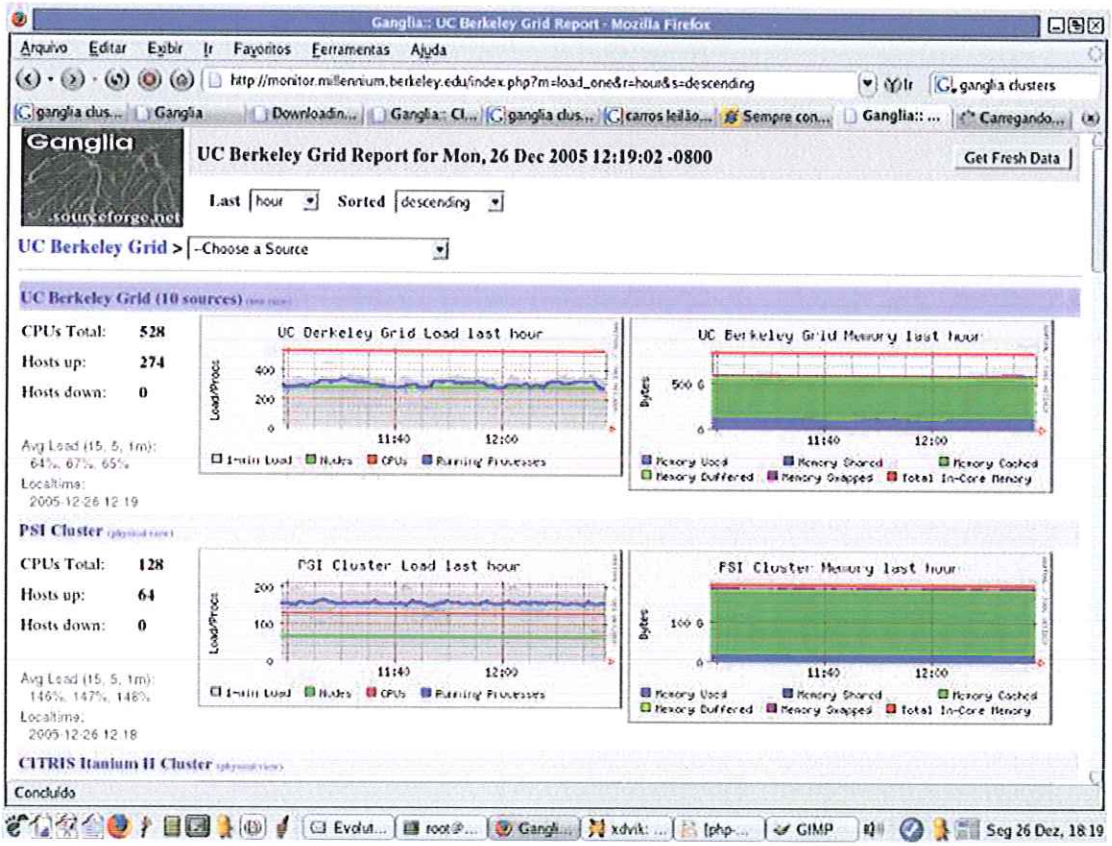

Figura 5.1: Característica do Ambiente Ganglia [18]

5.3 *Smile Cluster Management Systems*

O *SCMS* (*SMILE Cluster Management Systems*) visa oferecer recursos para administração de aglomerados de computadores. Essa ferramenta oferece suporte para navegar e interagir com componentes internos, envia comandos e coleta seus resultados de execução, emite alarmes sobre o estado crítico de algum recurso ou sobre problemas de desempenho. É indicada para ambientes compostos por até 16 computadores, tornando-se inviável para um número superior a esse, ocasionando queda no desempenho [40].

As principais funcionalidades dessa ferramenta são: comandos unix paralelos, alarme, monitoração, serviço de nomes e serviço de eventos distribuídos, que são apresentados nas tabelas 5.1 e 5.2.

Tabela 5.1: Funcionalidade do Smile Cluster Management System

Funcionalidade	Descrição da Funcionalidade
Node Status	Mostra o *status* atual do elemento de processamento
Control Panel	Acessa o painel central de cada elemento de processamento
Disk Space	Mostra o espaço disponível em disco
FTP	FTP do nodo
Node File System	Mostra o sistema de arquivos montado
Process Status	Mostra o *status* dos processos
Reboot	Reinicia o elemento de processamento
RPM Package	Gerenciador de pacotes RPM
Shutdown	Desliga a elemento de processamento
Telnet	Telnet no elemento de processamento
User Check	Verificar o usuário no elemento de processamento
Start Daemon	Inicia o *daemon* de monitoração.
System Monitor	Mostra o monitoramento do sistema.
Show Config	Mostra as configurações
Update Config	Atualiza as configurações de Hardware
Paralel GUI	GUI para todos os comandos paralelos.
Alarm GUI	GUI para sistema de alarmes.
Motherboard	Mostra o status da *motherboard.*

Tabela 5.2: Comando que o Sistema Smile Cluster Management System implementa

Comando	Descrição
Pcat	Mostra o conteúdo dos arquivos
Pcp	Distribui arquivos para os elementos de processamento
Pexec	Executa comando para o grupo
Pfind	Procura arquivos em vários elementos de processamento
Pfsp	Procura por processos através do nome do processo
Pkill	Elimina um processo dentro do cluster através de seu nome
Pkillu	Elimina um processo no cluster usando o nome do usuário
Pls	Lista a estrutura de arquivos no cluster
Pload	Mostra o relatório de carga do sistema
Ppred	Executa comando em múltiplos elementos de processamento do cluster
Pps	Mostra informações de processos em cada elementos de processamento
Prm	Apaga um ou vários arquivos no elementos de processamento

No *SCMS* o sistema de alarme consiste em um *daemon*, denominado *alarm manager*, responsável por criar *daemons* detectores em cada computador do sistema. Esses são responsáveis por gerar relatórios de anomalias encontradas dentro do ambiente. Os relatórios de alarmes podem ser enviados por email ou por meio dos comandos Unix.

5.4 PARMON

PARMON é uma ferramenta projetada para ser portável, flexível, interativa, escalável, transparente de localização e capaz de monitorar aglomerados compostos por muitos computadores. Essa ferramenta segue a metodologia cliente/servidor, que permite monitorar múltiplas instâncias do mesmo componente, tais como diversas CPUs [21].
Ela monitora, também, atividades do sistema e utilização de recursos tais como: atividades dos processos, logs do sistema e atividades do núcleo do sistema operacional (kernel).

Os dois principais componentes da implementação do *PARMON* são parmon-server, que tem como principal atividade prover informações sobre o uso de recursos do sistema, e o parmon-client, responsável por coletar dados e apresentá-los graficamente. A implementação do cliente foi feita na linguagem Java, e o servidor, em C, usou POSIX *threads*.

Algumas características da implementação do *PARMON* para o monitoramento de aglomerados de computadores são:

1. Visualização dos recursos: geração de gráficos e observação de métricas;

2. Atividades dos processos: avaliação da utilização de recursos de memória de processamento;

3. Log do sistema: permite processar mensagens de sistema (syslog) por entradas que ocorreram em uma hora específica, ou por entradas que contém uma palavra chave;

4. Atividades do kernel: suporta instrumentação de software dos recursos do sistema (como CPU, memória, disco e rede) e suas atividades;

5. Visão de componentes físicos e lógicos: mostra representações do sistema físico e de alguns de seus componentes, os quais ajudam na administração do sistema;

6. Geração de eventos: permite ao administrador definir eventos como enviar e-mail quando o usuário ultrapassa limites de utilização de recursos;

7. Representação de dados: usa gráficos de pizza, barra e linha para representar uso de recursos como disco e memória;

8. Outros: permite executar comandos em computadores; obtém lista de usuários ativos em cada computador; suporta envio simultâneo do mesmo comando para múltiplos computadores; obtém informações de sistema.

A figura 5.2 representa o modelo esquemático de *PARMON*.

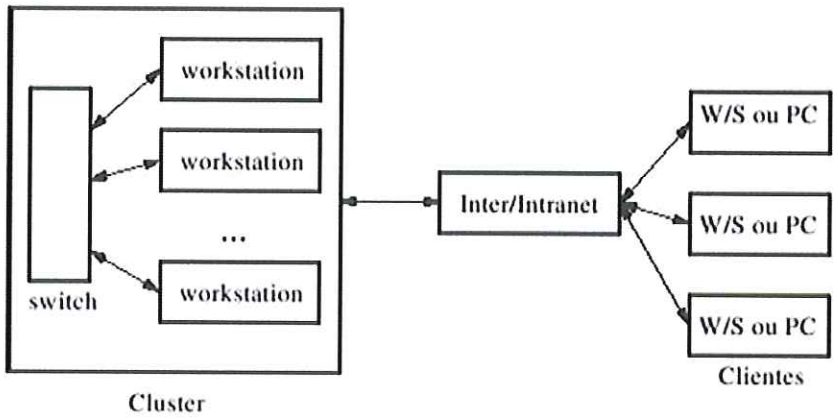

Figura 5.2: Modelo Esquemático do PARMON

5.5 openMosix

O openMosix é um conjunto de funcionalidade embutidas em *patch* para o kernel do sistema operacional Linux. Esse *patch* faz com que múltiplos computadores sejam monoprocessados ou multiprocessados operem de maneira cooperativa na execução das tarefas, sendo formado por diversos algoritmos, para compartilhamento de recursos. Tais algoritmos são desenvolvidos para aproveitar melhor os recursos computacionais disponíveis pelos elementos de processamento do cluster. O openMosix trabalha de maneira transparente na migração de processos de um elemento de processamento para o outro, procurando sempre manter um balanceamento de carga e evitar operações de paginação de memória (*swap*) [49].

A tecnologia implementada no openMosix consiste basicamente em

duas partes: 1) um mecanismo de migração que preempta processos; e 2) algoritmos de compartilhamento de recursos. Ambas aplicações são implementadas no kernel do sistema operacional Linux, fazendo com que não haja necessidade de modificação da aplicação para utilizar o openMosix [41].

A MPP(*Massively Parallel Processors*) é a ferramenta principal para os algoritmos de gerência de recursos, pois esses utilizam de maneira otimizada os recursos disponíveis no sistema. É ainda capaz de migrar qualquer processo, a qualquer instante, para qualquer elemento de processamento disponível. Normalmente, essas operações ocorrem de maneira automatizada pelo sistema, com base em informações providas por algoritmos de compartilhamento de recursos. Toda operação realizada pelo openMosix é feita sem a necessidade de um controle central, ou seja, não existe nenhum relacionamento do tipo mestre/escravo ou gerente/agente. Cada elemento de processamento do cluster opera de maneira autônoma, tomando as suas próprias decisões de forma independente. Tal estrutura proporciona a realização de configurações dinâmicas, em que os elementos de processamento podem ser adicionados ou removidos do conjunto de computadores, sem ocasionar grandes impactos ao sistema.

Os algoritmos de compartilhamento de recursos do openMosix destacam-se principalmente pelo balanceamento de carga e o de memória. O algoritmo de balanceamento dinâmico de carga tenta reduzir a diferença de carga entre os pares de elementos de processamento, migrando os processos de um elemento de processamento sobrecarregado

para outros menos sobrecarregados [41].

O algoritmo de memória, previne a insuficiência de memória evitando assim problemas de paginação ou utilização de memória virtual. Ele é ativado quando um elemento de processamento começa a executar muita paginação em virtude da insuficiência de memória livre. Assim, o algoritmo faz com que esse processo seja migrado para um elemento de processamento onde possua memória livre, mesmo que para isso possa ocasionar um desajarranjo na distribuição de carga do sistema [49].

O algoritmo implementado no openMosix é de grande utilidade, pois concilia principios baseados em economia e análise de competividade. Para chegar em tal conclusão, os recursos de comunicação são medidos em termos de largura de banda, memória em termos de espaços, e CPU em termos de ciclos. A principal idéia é fazer com que esses recursos heterogêneos sejam convertidos em custos homogêneos. Tal recurso faz com os processos possam ser transferidos para o local onde exigir menor custo (menor ocupação de memória, ocupação de CPU, processos em execução, etc) [46].

5.5.1 Vantagens e desvantagens

O openMosix implementa uma gama de recursos novos, ao kernel do sistema operacional Linux, que faz dessa uma estrutura de cluster ideal, pois permite fazer uso dos processadores de maneira totalmente transparente. Entretanto, tais recursos não são aplicavéis a qualquer

tipo de processo, principalmente para as seguintes aplicações:

- Processos com baixo custo de processamento, como aplicativos que fazem uso de alta comunicação interprocessos;
- Aplicações com memória compartilhada, geralmente os *Web Servers ou Database Servers*;
- Aplicações dependentes do hardware, que necessitam de acesso a um periférico em algum elemento de processamento específico do cluster;
- Aplicações com diversas linhas de execução (*threads*) não ganham desempenho;
- A execução de um único processo, como o *browser*, não ganha performance.

Apesar de parecer muito desvantajoso em um primeiro momento, os ganhos obtidos de sua implementação são inúmeros, pois as vantagens existentes estão relacionadas a questões de paralelismo nas execução das aplicações. Essas vantagens são listadas a seguir:

- Processos de uso intensivo da CPU, os quais ocasionam longos tempos de execução e baixo volume de comunicação;
- Compilação de programas extensos;
- Processos que possuem tempo de execução longos e rápidos ou com quantidade moderada de comunicação interprocessos;

- Processos com muita operações de entrada e saída (E/S);
- Banco de dados que não utilizam memória compartilhada;
- Processos que podem ser migrados manualmente.

5.5.2 Softwares de gerenciamento

openMosixview

Para manter tal arquitetura, é importante adotar-se um software para gerenciamento do cluster. O software mais utilizado para gerenciamento de cluster, para openMosix é o gerenciador gráfico openMosixview. Com essa aplicação também é possível ajustar os principais parâmetros do cluster, além de permitir a visualização de carga do sistema, a visualização dos processos migrados e de executar ou transferir processos manualmente.

O conjunto de ferramentas que fazem parte do openMosixview são:

- openMosixview: aplicação principal, que além de dar acesso às outras, permite a visualização de desempenho do cluster;
- openMosixprocs: permite o gerenciamento de processos;
- openMosixcollector: por meio de um *daemon* que coleta informações dos elementos de processamento, permite a exibição do comportamento do cluster durante um período de execução;
- openMosixanalyzer: utilitário utilizado para analisar dados coletados pelas aplicações do openMosix;

- openMosixhistory: exibe um histórico dos processos que estão executando no sistema;
- openMosixmigmon: permite visualizar, por meio de um mapa, as ocorrências de migração de processos.

O openMosixview apresenta uma interface gráfica bem intuitiva e de simples manuseio, como demonstra a figura 5.3. No lado esquerdo, na janela da aplicação, são exibidos os IPs dos elementos de processamento que compõem o cluster, e ao seu lado, há uma numeração indicando se o mesmo está operacional ou não (verde para funcionando e vermelho para indicar alguma falha).

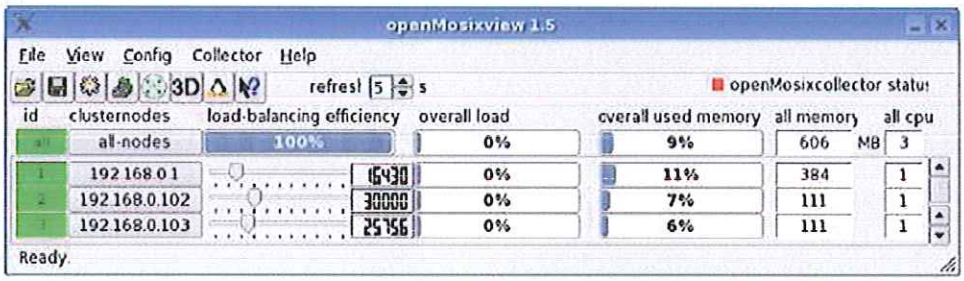

Figura 5.3: Tela principal do openMosixview

Em seguida, com o botão deslizante, pode-se modificar a velocidade para cada elemento de processamento. O balanceamento de carga do sistema é realizado com base nesses valores, por exemplo, um processo pode migrar facilmente para um elemento de processamento cujo valor de velocidade esteja mais alto em comparação aos outros. Cabe ressaltar que essa velocidade não representa a capacidade física do equipamento, e sim a velocidade com que o openMosix avalia o elemento de processamento em questão.

As barras de progresso, encontradas no meio da janela, informam o quanto estão sobrecarregados os elementos de processamento do sistema e o consumo de memória de cada um deles. Na seqüência, os números indicam o total de memória e a quantidade de CPUs para cada elemento de processamento, respectivamente. Cabe ressaltar que a primeira informação, localizada logo acima desses valores, indica o desempenho do sistema em sua plenitude.

Pode-se ainda realizar algumas configurações inerentes a cada elemento de processamento desejado. Ao selecionar o endereço IP do elemento de processamento, será exibido uma tela de configuração semelhante ao da figura 5.4, onde os parâmetros como o mecanismo de migração automatizada, migração de processos locais, comunicação com outros elementos de processamento, dentre outros, podem ser ativados ou desativados.

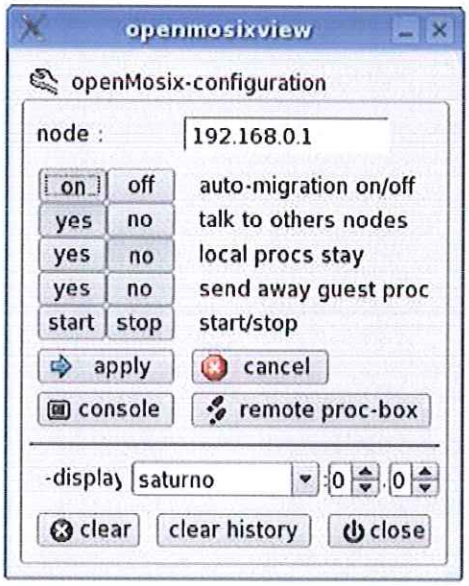

Figura 5.4: Janela de configuração de um elemento de processamento do cluster

5.5.3 Gerenciando Processos

O openMosixprocs é a aplicação responsável pelo gerenciamento dos processos que estão sendo executados no cluster. É possível visualizar nessa aplicação informações como a identificação de cada processo, o elemento de processamento em que está sendo executado, a situação atual, o comando referente ao processo, entre outros.

Na primeira coluna, contida na figura 5.5, nota-se a presença de alguns ícones para cada processo. Aqueles que estão representados por um cadeado não podem ser migrados.

Os ícones de engrenagem representam processos que podem ser migrados, e estão na cor verde para processos remotos (migrados) e laranja para processos locais.

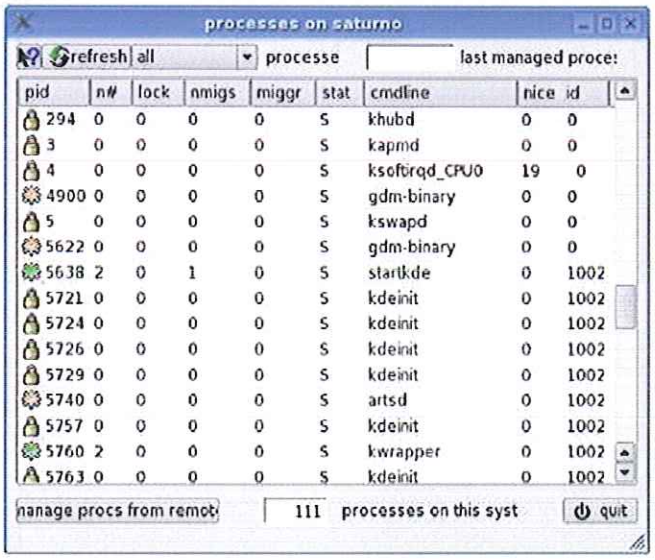

Figura 5.5: Janela de gerenciamento de processos

Ao disparar um duplo clique do mouse em um dos processos da lista, a janela de migração da figura 5.6 será apresentada. Com ela, poderão ser realizada tarefas tais como travar um processo, via botão *sigstop*, e reiniciá-lo, via botão *sigcont*, bem como migrar processos manualmente. Com o botão deslizante, localizado na parte inferior, pode-se atribuir um valor de prioridade para o processo, em que o número ou valor -20 significa prioridade máxima e o valor 20 prioridade mínima.

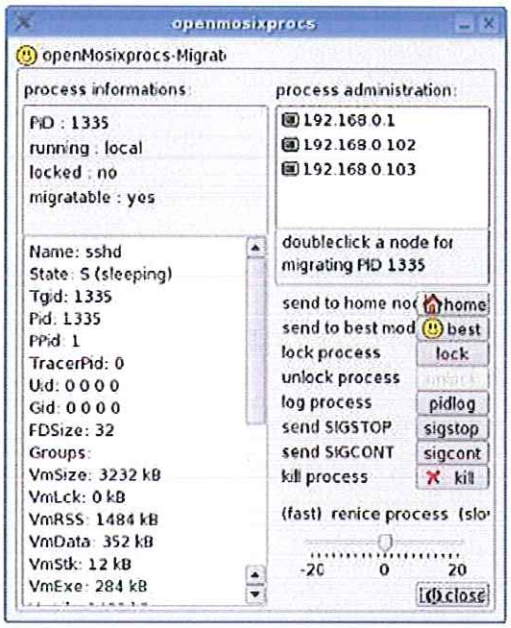

Figura 5.6: Informação inerente a um processos

5.5.4 Histórico de Utilização

Há um processo daemon chamado de openMosixcollector que cria arquivos de log que armazenam informações de carga de cada elemento de processamento do openMosix, de modo a manter assim um histórico

das atividades realizadas.

Posteriormente, esse histórico pode ser utilizado pelo analisador de logs *openMosixanalyzer*, que faz uma avaliação ininterrupta da carga, memória e processos do cluster. A tela da figura 5.7 mostra, por meio de gráficos estatísticos, um histórico da operacionalidade do cluster utilizando a ferramenta *openMosixanalyzer*.

Através dos logs gerados pelo processo *daemon* citado anteriormente, pode-se ainda visualizar os processos que foram executados durante o tempo de coleta, através da ferramenta *openMosixhistory* (figura 5.7).

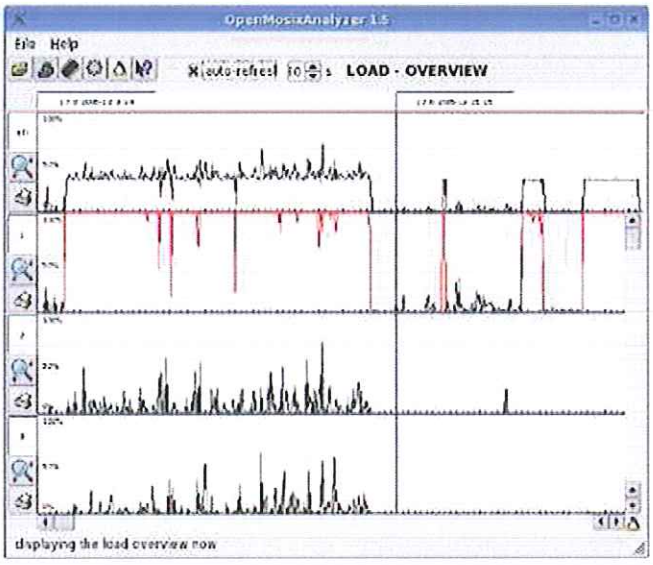

Figura 5.7: Histórico contendo a carga de cada elemento de processamento do cluster

5.5.5 Monitorando Migração

O openMosixmigmon é uma ferramenta usada para monitorar migrações efetuadas no cluster. São apresentados vários pingüins envoltos por um círculo representando os elementos de processamento, conforme ilustrada pela figura 5.8.

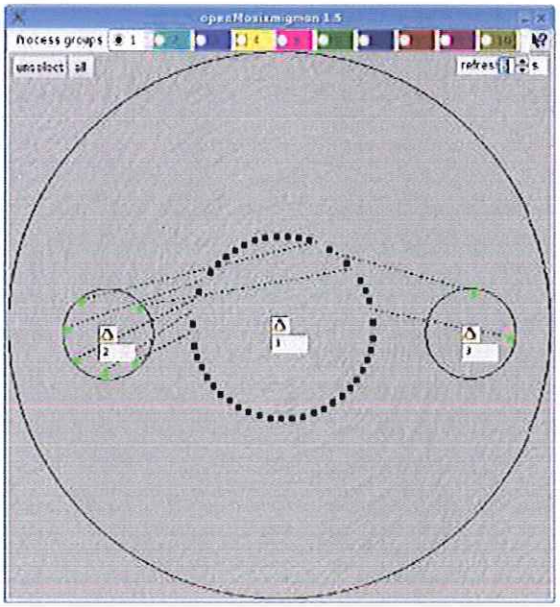

Figura 5.8: Histórico contendo a carga de cada elemento de processamento do cluster

O círculo maior, localizado no meio da imagem, identifica a máquina que está executando a ferramenta. Os pequenos quadrados pretos, formando a circunferência do elemento de processamento, representam os processos que podem ser migrados. Quando um processo migra para um dos elementos de processamento, é exibido uma ligação entre o processo principal (*deputy*) e o processo remoto (remote), sendo o último marcado por um quadrado verde.

Outra característica interessante dessa ferramenta é a possibilidade de arrastar o processo e solta-lo em um outro elemento de processamento.

5.6 Sistema de monitoramento de carga para clusters

Villas Boas e Travieso [4] propõem uma ferramenta para o monitoramento de processos em clusters. Tal ferramenta é constituída por duas partes: uma *interface* de programação que permite a fácil inserção de códigos em aplicações paralelas para realizar o monitoramento em um sistema de monitoramento de carga; e a mesma apresenta um sistema de monitoramento de cargas em tempo de execução, para recolher as informações de cargas dos computadores do ambiente distribuído.

A ferramenta baseia-se na arquitetura cliente/servidor, sendo que nesse caso, o servidor é o sistema de monitoramento que fornece o serviço de informação de carga do elemento de processamento que hospeda as informações, e o cliente é a parte das aplicações paralelas responsável por pedir as informações de carga dos elementos de processamento do sistema distribuído [4].

Foram utilizados para o desenvolvimento da ferramenta: linguagem de programação Java [5], *RMI - Remote Method Invocation* [6] e um cluster de computadores do tipo *Beowulf*. A linguagem Java foi adotada por ser portátil, orientada a objetos, oferecer facilidade de comunicação entre os elementos de processamento do sistemas dis-

tribuídos por meio de RMI, e pela facilidade de programação.

5.6.1 Interfaces

Interface LoadIndex

A ferramenta apresenta duas *interfaces* gráficas implementadas em Java: *LoadIndex* e *Resource*. A primeira representa o índice de carga, que mostra o quanto de recursos um determinado elemento de processamento está ocupando dentro do sistema. A segunda descreve objetos cujos métodos destinam-se a medir o determinado índice de carga, sendo que esses podem ser chamados remotamente. As classes que implementam essas *interfaces* são os tipos de medidas de carga, as quais se deseja realizar. Três tipos de medidas de cargas foram desenvolvidas: uso de CPU, memória e de rede.

Interface Resource

Essa *interface* de programação define métodos, para que classes implementem o cálculo do índice de carga adotando a métrica desejada (uso de CPU, de memória ou de rede).

O cliente (parte da aplicação paralela interessada no monitoramento de cargas), para determinar o tipo de medida de carga, só precisa da definição dessa *interface* para poder realizar chamadas ao método e posteriormente receber o índice de carga correspondente ao pedido efetuado. Três classes que implementam a *interface Resource* são apresentadas a seguir: *CPUResourceImpl, MemResourceImpl e*

NetResourceImpl.

Uma breve descrição de cada uma delas é apresentada a seguir:

- *CPUResourceImpl*: classe destinada a calcular o índice de carga de CPU. A medida de carga da CPU é realizada por meio da leitura de parte do arquivo */proc/stat.*
- *MemResourceImpl*: essa classe implementa a medida de carga de memória, sendo os objetos dessa classe os que definem três cálculos para determinar esse índice: memória *RAM*, *Swap* e *RAM + Swap.*
- *NetResourceImpl*: implementa a medida de carga de rede que é realizada através da leitura do arquivo */proc/net/dev.* Sendo observados a quantidade de *bytes* recebidos e a quantidade de *bytes* transmitidos por uma dada *interface* de rede.

Esse sistema de monitoramento de carga é representado pelo servidor *RMI*, o qual fornece os índices de cargas dos elementos de processamento do sistema distribuído. Vale ressaltar que o servidor e os clientes constituem a base para o funcionamento dessa ferramenta, assim o cliente é a parte da aplicação que pede as informações de carga aos elementos de processamento do sistema e o servidor, a parte da aplicação que é executada em todos os elementos de processamento para se medir um certo tipo de carga e atender às requisições dos clientes.

Dentre as características mais importantes dessa ferramenta de monitoramento de cargas em sistemas distribuídos destaca-se o fato de ela ser portátil, por poder ser utilizada em grandes sistemas distribuídos que sejam heterogêneos e que até possam ter elementos de processamento com sistemas operacionais diferentes (devido ao fato do Java trabalhar usando a máquina virtual Java).

A ferramenta desenvolvida além de fornecer a carga dos elementos de processamento de um sistema distribuído às aplicações paralelas, ou a um sistema de balanceamento de cargas, pode também ser usada para gerenciamento de um computador paralelo [4]. Apresentada ainda recurso para monitoramento de sistema em *Grids* computacionais.

5.7 Integração de Ganglia, libRastro e Pajé para o Monitoramento de Aplicações Paralelas

O uso integrado de diferentes ferramentas de monitoramento, permite aprimorar a capacidade de análise das execuções de aplicações paralelas. Essa intregração será ilustrada nessa seção com o processo de integração dos dados coletados pelo Ganglia, aos traços de execução gerados por libRastro, que é uma biblioteca de recursos para aplicações paralelas. A visualização dos dados integrados é feita com a ferramenta Pajé. Através de alguns exemplos de visualização integrados, demonstra-se que as informações sobre o estado do *cluster* permitem, inclusive, detectar eventuais problemas na execução da aplicação.

Esse processo de integração envolve dois tipos de ferramentas: aquelas especializadas no monitoramento de *clusters* de computadores, e aquelas que se destinam ao monitoramento de aplicações paralelas e distribuídas.

Detalhes sobre o Ganglia foram apresentados na seção 5.2.

A biblioteca libRastro [42] permite monitorar uma aplicação paralela através do resgistro de eventos observados durante a sua execução, como por exemplo, o envio ou recepção de mensagens, a criação de *threads*, etc. Em uma plataforma do tipo *cluster*, os eventos são resgistrados em cada elemento de processamento para, após o término da execução, serem sincronizadas e combinados em um único arquivo de traço. Uma das característica da libRastro é sua generalidade, isto é, sua total independência em relação à aplicação que a utiliza, à biblioteca de comunicação empregada e à ferramenta de visualização utilizada [42].

Pajé é uma ferramenta de visualização que permite observar o comportamento de aplicações em ambientes de execução paralelos e distribuídos. Para isso, Pajé utiliza traços gerados por outras ferramentas durante a execução das aplicações. Uma particularidade do Pajé é a combinação de três propriedades importantes no contexto da visualização de aplicações paralelas: interatividade, escalabilidade e extensibilidade. A interatividade de Pajé é garantida por uma *interface* gráfica que mostra diferentes eventos em um diagrama espaço-tempo, permitindo alterar a escala de visualização, avançar e voltar no tempo

e, também inspecionar os detalhes da cada evento [43]. A escalabilidade por outro lado, refere-se à habilidade de lidar com um grande número de objetos visuais (como por exemplo, aplicações com muitas *threads*). A extensibilidade é garantida pela possibilidade de visualizar informações provenientes de diferentes ferramentas de monitoramento, em diferentes níveis de abstração [43].

5.7.1 Integração de Ferramentas de Monitoramento

A utilização de diversas ferramentas distintas para o monitoramento de aplicações que faz coleta de dados, pode ser complementada e proporcionar ao desenvolvedor uma visão mais completa da execução de seus programas[24]. Dessa forma, da união dos resultados gerados pelas diferentes ferramentas pode-se obter uma análise mais detalhada e ampla. Essa integração depende fortemente das características de cada ferramenta utilizada.

Para a integração dessas diversas ferramentas Veiga, Scheid e Schnoor [24] capturaram os dados dos recursos oferecidos pelo Ganglia. Essa ferramenta armazena dados de monitoramento em uma base RRD (*Round Robin Database*) o que simplifica a obtenção dos dados[24]. Com a leitura das informações de uma única fonte foram obtidos os dados de todo o *cluster*.

A biblioteca libRastro registra os eventos gerados durante a execução de uma aplicação em um formato compacto, sendo necessário decodificar esses resultados para gerar um traço intermediário que possa ser

facilmente convertido para o formato aceito pela ferramenta Pajé. A integração das informações geradas pela libRastro e Ganglia foi feita de forma a gerar um único arquivo de traço, visualizável através do Pajé, e que permitisse cruzar as informações de forma a avaliar todos os elementos de processamento do *cluster*. Pajé permite a visualização de dados de forma hierárquica.

Para a sincronização dos dados coletados foi necessário implementar um pequeno programa monitor iniciado junto com a aplicação monitorada. Esse programa teve como finalidade coletar informações do sistema e as enviar, periodicamente, para o Ganglia. Por fim, foi feita a implementação e a sincronização dos eventos temporais gerados pela libRastro com os dados monitorados pelo Ganglia. Os dados coletados pelo Ganglia, em diferentes elementos de processamento, foram sincronizados no momento em que foram armazenados em um servidor central. A libRastro também tem um mecanismo de sincronismo que ajusta os tempos de todo os eventos com base no relógio de um dos processadores [24].

5.8 Rvision

Ferrero e De Rose [27] motivados pela inflexibilidade encontrada em sistemas de monitoração propuseram um estudo e melhorias para o Rvision. A arquitetura aberta do RVision provê monitoração genérica, a qual habilita a utilização de bibliotecas de monitoração próprias dos recursos dos elementos de processamento, assim como, a facilidade de

cliente ou administrador monitorar esses elementos conectando-se por meio da Internet [27].

5.8.1 Conceitos e Características

Durante o projeto do RVision, alguns novos conceitos de monitoração foram introduzidos, a fim de aprimorar a funcionalidade do sistema. Esses conceitos são:

- ***Sessão de Monitoração:*** A sessão de monitoração consiste em uma configuração específica para um cliente no ambiente de monitoramento. A configuração desse ambiente tem como objetivo principal definir a lista de elementos de processamento monitorados, informações a serem monitoradas e a frequência de aquisição de informação.
- ***Biblioteca de Monitoração:*** Consiste em uma coleção de rotinas responsáveis pela captura de informação. Essas bibliotecas podem ser implementadas pelo usuário a fim de adicionar novas funcionalidades ao monitor. A principal vantagem obtida por essa abordagem é a separação dos mecanismos de aquisição e transporte de informações, além da freqüência de monitoração.

As principais características apresentadas pelo RVision são:

- ***Múltiplas Sessões de Monitoração:*** o RVision pode ser utilizado por vários usuários de forma concorrente, cada um com sua

própria sessão de monitoração, que pode apresentar configurações diferentes.

- ***Monitoração Genérica:*** através da utilização de bibliotecas de monitoração, o RVision pode ser utilizado para monitorar qualquer tipo de informação, sendo necessária somente a criação de uma biblioteca de monitoração com função de fornecer a informação desejada.

- ***Análise de Informação:*** as informações podem ser analisadas durante a monitoração, isto é, *online*, ou após o encerramento da monitoração, chamada de análise *POST-MORTEM*. A análise a *posteriore* é realizada por meio de um arquivo com as informações capturadas durante a monitoração.

- ***Freqüência de Monitoração:*** as freqüências de monitoração suportadas pelo RVision são: cíclica, por alteração e por demanda. A cíclica captura as informações periodicamente; a por alteração também captura periodicamente, mas as informações somente são enviadas quando uma condição é satisfeita; e a por demanda consiste na aquisição de informações quando um cliente realiza uma requisição explícita.

5.8.2 Arquitetura e Implementação

A arquitetura do RVision é baseada no modelo cliente-servidor. Essa abordagem centralizada foi escolhida por apresentar menor complexidade em comparação à distribuida. Segundo os autores, Ferrero e De

Rose[27], essa abordagem é mais eficiente para *clusters* com dezenas de elementos de processamento. O RVision foi implementado em linguagem C para o sistema operacional GNU/Linux. A arquitetura é composta por cinco módulos, divididos em cliente de monitoração, interface de monitoração, RVCore, RVSpy e biblioteca de monitoração. O RVision é distribuído sob licença GNU/GPL [33].

5.8.3 Funcionamento

A figura 5.9 apresenta a arquitetura do RVision, constituída pelos os seus cinco módulos. O *Cliente de Monitoração* utiliza-se das funções providas pela *Interface de Monitoração* para se comunicar com o *RVCore*. As funções existentes na *Interface de Monitoração* provêem mecanismos básicos para gerenciamento da monitoração, tais como a configuração da sessão de monitoração, inicialização e encerramento da monitoração.

O *RVCore* requer autenticação dos clientes para uso do sistema de monitoramento. Após autentica-los ele cria uma sessão de monitoração para cada cliente. Essa sessão pode ser configurada de acordo com preferências do usuário e após essa etapa a monitoração é iniciada.

A etapa de inicialização informa ao módulo *RVSpy*, existente em todas as máquinas previamente configuradas na sessão do usuário, dados sobre bibliotecas de monitoração, informações e freqüências de monitoração utilizadas em cada elemento de processamento monitorado.

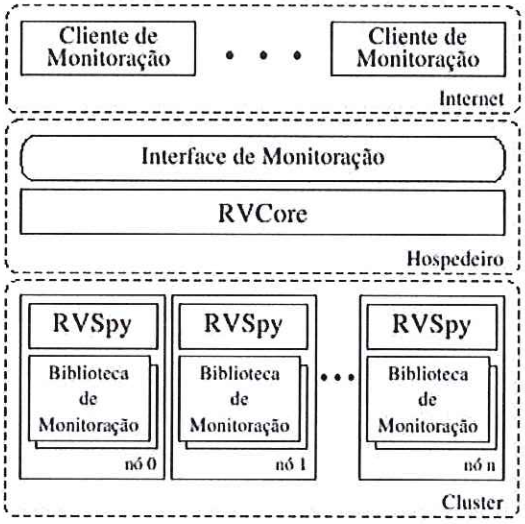

Figura 5.9: Arquitetura do RVision

O *RVSpy* recebe as preferências do usuário e realiza a ligação dinâmica das *Bibliotecas de Monitoração*, conforme definido na sessão do usuário. Esse módulo ainda dispara submódulos de coleta de informações, os quais agem de acordo com o tipo de freqüência e análise definidos. Os submódulos de coleta obtêm informações através das funções definidas nas *Bibliotecas de Monitoração*.

A *Biblioteca de Monitoração* consiste em uma *shared library* do sistema operacional GNU/Linux. Ela deve obedecer uma estrutura pré-definida, para que possa ser ligada dinamicamente ao *RVSpy*. No término da monitoração, o módulo *RVSpy* finaliza os submódulos de coleta de informações e limpa a área de memória utilizada.

5.9 Crono – Sistema de Gerência de Aglomerados

Crono é um sistema para gerência de aglomerados de computadores que tem como objetivo controlar o acesso a recursos. Esse tipo de controle se faz necessário, principalmente, para que as aplicações dos usuários não interfiram umas nas outras. O Crono fornece dois tipos de alocação de recursos: o primeiro, chamado de alocação espacial, define quais elementos de processamento podem ser utilizados pelos usuários; o segundo, chamado de alocação temporal, permite o compartilhamento de elementos de processamento por diferentes usuários. Esse último é utilizado, por exemplo, nos casos onde os usuários estejam apenas testando aplicações e, conseqüentemente, não necessitam de desempenho [34].

A arquitetura do sistema Crono é formada pelos seguintes componentes: interface do usuário, o gerenciador de acesso, o gerenciador de requisições, o gerenciador de execução e o gerenciador do elemento de processamento. Os componentes da arquitetura e o fluxo de informações, representados respectivamente pelos retângulos e pelas setas, são apresentados na figura 5.10.

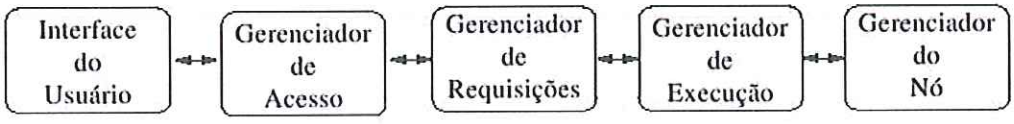

Figura 5.10: Arquitetura do Sistema de Gerência de Aglomerados Crono

A Interface do Usuário (IU) permite interação com o sistema de monitoramento por meio de uma interface gráfica ou utilizando um *shell* (bash, tcsh, etc.). Através desse componente são disponibilizadas ferramentas que possibilitam aos usuários e administradores executarem diversas tarefas, entre elas: mostrar informações relativas à fila de requisições, submeter aplicações para execução, liberar os recursos, alterar o tempo e a quantidade dos elementos de processamento utilizados, configurar o ambiente de execução e obter informações sobre direitos de acesso e nome dos elementos de processamento [34].

O Gerenciador de Acesso (GA) é responsável por autenticar usuários e verificar os direitos de acesso sobre o sistema. Nesse componente o administrador, através de arquivos de configuração, pode criar grupos de usuários e atribuir restrições de acesso. Essas restrições são definidas pelo tempo máximo e quantidade máxima de elementos de processamento para alocação [34].

O Gerenciador de Requisições (GR) do Crono tenta aproveitar os recursos disponíveis (tempo e elementos de processamento). Neto, Barcelos e De Rose [34] afirmam que no uso do GR há menor desperdício em relação ao uso do esquema de alocação FCFS (*first-come first-served*), além disso, não prejudica usuários que estejam na fila de espera. Usuários são atendidos conforme agendamento. Esse componente também permite alteração dinâmica de reserva de recursos, isto é, usuários podem pedir maior tempo ou maior número de elementos de processamento para atender suas aplicações [34].

O Gerenciador de Execução (GE) recebe as requisições do GR, faz o tratamento de falhas (problemas nos elementos de processamento), prepara o ambiente de execução, e submete as aplicações para os elementos de processamento ou libera o acesso para execução de programas (dependendo se for modo *batch jobs* ou modo interativo, respectivamente) [34].

O Gerenciador do Nó (GN) é responsável por fornecer e alterar informações dos elementos de processamento do *cluster*, através de requisições feitas pelo gerenciadores de Requisições e Execução. A principal funcionalidade desse componente é controlar o acesso não autorizado de usuários [34].

A seguir é descrito o processo de controle de acesso aos elementos de processamento.

5.9.1 Controle de Acesso aos Nós dos Aglomerados

Para que não haja interferência no desempenho das aplicações de diferentes usuários, deve-se fazer a proteção dos elementos de processamento, ou seja, garantir que os usuários façam uso apenas dos recursos computacionais agendados. Esse controle de acesso é feito alterando arquivos de configuração do sistema GNU/Linux [48]. Quando é feito o processo de *login* em um elemento de processamento, o arquivo *login.access* é verificado a fim de controlar o acesso do usuário. Para garantir que apenas usuários com agendamento possam utilizar os recursos, o Crono altera o *login.access* para definir as permissões. Entre-

tanto, alterar o arquivo *login.access* não é o suficiente para restringir esse acesso. Os usuários podem ainda utilizar o comando rsh (*Remote Shell*) para executar operações remotamente. Isso porque o servidor do rsh não realiza todo processo de *login*, apenas a autenticação dos usuários. Assim, o *login.access* é ignorado.

Para resolver esse problema, o Crono também altera o arquivo *hosts.equiv* dos elementos de processamento. O servidor rsh analisa esse arquivo e, caso o usuário não esteja autorizado, o servidor rsh requisita uma senha, realizando então o processo de *login*. Ainda é analisado o arquivo *login.access*, que não permite que usuários não autorizados tenham acesso. Existe ainda um problema quanto ao arquivo .rhosts, pois o servidor rsh o analisa antes do *hosts.equiv*. Isso abre a possibilidade do usuário alterar seu próprio .rhosts e executar comandos remotamente, sem que seja solicitada sua senha. Isto é facilmente resolvido, bastando inicializar o servidor rsh com a opção de ignorar o arquivo .rhosts do usuário.

5.10 Considerações Finais

Este capítulo apresentou as principais funcionalidades implementadas por ferramentas de administração e gerência de aglomerados de computadores. O presente trabalho baseou-se principalmente no Ganglia, devido ao fato de ser uma ferramenta que proporciona a clientes e aos administradores do sistema uma visão global dos aspectos de monitoramento de um *cluster*, e incluiu como um recurso adicional a

possibilidade de gerenciamento dos elementos de processamento por meio de um shell linux.

Capítulo 6

Projeto e Implementação de uma Ferramenta de Administração e Gerência de Aglomerados de Computadores (FAGAC)

6.1 Considerações Iniciais

Nesse capítulo será descrita a ferramenta denominada FAGAC que foi desenvolvida para a administração e gerenciamento de aglomerados de computadores. É apresentada a metodologia, adotada para a sua implementação e o fluxo de nagevação, o qual permite observar todas as funcionalidades dessa ferramenta.

6.2 Objetivo

O objetivo do presente trabalho foi desenvolver uma nova ferramenta denominada FAGAC para o gerenciamento e administração de aglomerados de computadores, tendo como característica principal ser

acessível através da Web e não causar atrasos significativos em aplicações que executam nesse ambiente de aglomerados, ou seja, ser de baixa intrusão. A acessibilidade e simplicidade do uso da ferramenta é avaliada, bem como sua intrusão nos computadores do aglomerado.

6.3 Metodologia de Desenvolvimento

O desenvolvimento deste trabalho ocoreu dentro das seguintes etapas:

1. Trabalhos relacionados - De forma similar aos trabalhos apresentados no capítulo anterior (5), a ferramenta desenvolvida baseia-se no conceito de gerente/agente para monitoração dos recursos computacionais do ambiente. Nessa técnica, existe um agente que reporta os dados para um programa gerente, a partir do qual é feita a administração e gerência dos elementos de processamento do aglomerado. O *webbrowser* é utilizado para a exibição dos seguintes resultados obtidos: estado de carga dos computadores do algomerado, memória consumida pelos elementos de processamento, taxa de transferência, recebimento de pacotes das *interfaces* de rede, aspectos gerais do hardware, *status* de execução dos processos, etc. Através dos resultados informados pela ferramenta pode-se mensurar os estados e aspectos de carga dos computadores do aglomerado, determinado assim, quais computadores estão mais sobrecarregados dentro do aglomerado.

2. Protótipo da ferramenta - O desenvolvimento de um protótipo foi muito importante para o desenvolvimento final da ferramenta,

pois evidenciou inúmeras dificuldades que contribuiram para novas idéias de implementação. Com o protótipo foi possível fazer uso de *interfaces* que simplificaram a administração e visualização dos resultados obtidos. A *interface* de administração pode ser observada independente do sistema operacional do cliente e acessada através da *Web* fazendo com que a ferramenta seja flexível e de fácil acessibilidade. Os requisitos fundamentais levantados e implementados através da prototipagem da ferramenta foram:

- Monitoramento do estado de carga de cada computador;
- Monitoramento da quantidade de memória consumida;
- Monitoramento do tráfego gerado na rede;
- Monitoramento de espaço em disco;
- Informações sobre o hardware e configurações dos computadores.

3. Projeto e desenvolvimento da ferramenta - No desenvolvimento da ferramenta de administração e gerenciamento de aglomerados de computadores via *Web*, foi utilizada a linguagem PHP e o protocolo de gerenciamento de redes *SNMP*. A linguagem PHP foi adotada por permitir a criação de *websites* dinâmicos, interação por meio de formulários, execução do lado do servidor (e por isso, não necessitar de configurações específicas dos clientes) e uso de bancos de dados [18].

   O protocolo de gerenciamento de redes *SNMP*, foi escolhido por ser suportado em diversos sistemas operacionais e adotados em

softwares embarcados que executam ou implementam o protocolo *SNMP* sobre dispositivos utilizados em redes de computadores (tais como roteadores, switches, etc). O *SNMP* oferece um conjunto de *MIBs* (conjunto dos objetos gerenciados) que provêem informações sobre serviços de software, hardware e configurações. O envio de mensagem entre agentes e gerente é feita por meio da pilha de protocolo TCP/IP utilizando o protocolo UDP da camada de transporte. Vale lembrar que o protocolo UDP consome menos recursos na rede, contudo não apresenta confiabilidade, como visto na seção 4.4.

A figura 6.1 ilustra o funcionamento da linguagem PHP por meio de um navegador web em que o usuário requisita o acesso a uma página hospedada em um servidor. Esse servidor processa essa requisição, executa um *script* PHP e retorna o resultado para o cliente em linguagem HTML. Uma outra funcionalidade da ferramenta é o uso de um terminal idêntico a um terminal Linux implementado em Java proporcionando execuções tais como uso de linha de comando, uso de editores, inicialização e parada de serviços e demais operações disponíveis em terminais locais. Esse terminal utiliza o protocolo SSL (*Secure Socket Layer*) e permite a conexão segura com serviços SSH. Outras características desse terminal incluem: implementação em Java, sendo assim, portável para outros sistemas operacionais; executa sobre o servidor gráfico X11; suporta o protocolo SSH2; utiliza autenticação baseada em chave pública e privada; executa embutido em um na-

vegador; tem suporte a sftp (ftp seguro); compressão de pacotes; ferramenta licenciada sob GNU/GPL [40].

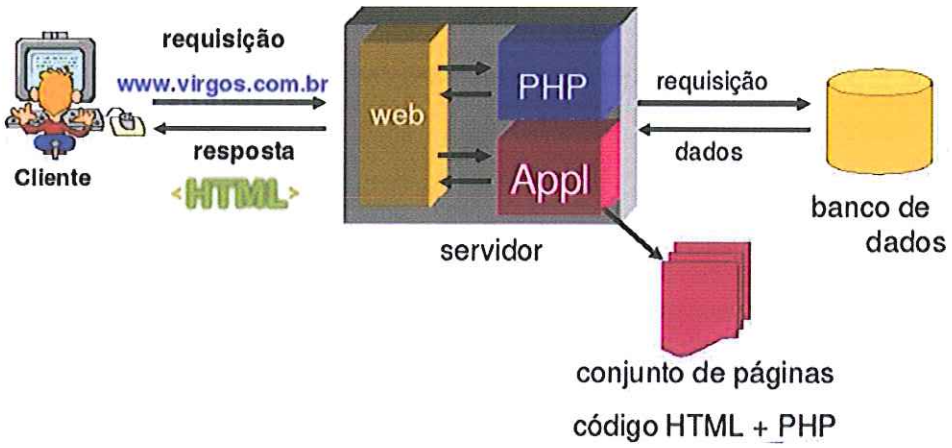

Figura 6.1: Arquitetura de funcionamento da linguagem PHP

A implementação das principais funcionalidades da ferramenta está definida e desenvolvida seguindo os seguintes modelos:

(a) Configuração dos agentes *SNMP*;

(b) Configuração do Gerente;

(c) Gerenciamento das MIBS;

(d) Configuração do *WebServer* e da linguagem PHP;

(e) Scripts para resgate das informações dos agentes atráves das informações providas pelas MIBS e exibição no ambiente *Web*.

A configuração dos agentes *SNMP* pode ser definida para ter um determinado tipo de acesso, que varia da seguinte forma: leitura, leitura e escrita, ou escrita. Isso determina o tipo de acesso à informação pelo usuário.

Antes que qualquer objeto possa ser lido ou possa ser escrito, o nome comunitário do *SNMP* deve ser conhecido. Esses nomes comunitários são configurados pelo administrador do sistema e podem ser vistos como senhas necessárias para anexar dados ao *SNMP*. No sentido exato, nomes comunitários existem para permitir que parte das *MIBs* no protocolo *SNMP* e subconjuntos de objetos sejam referenciados. Como o termo "comunitário"é requerido, espera-se que a verdadeira intenção desses valores sejam identificar comunitariamente entre os objetos *SNMP* configurados. Porém, é prática comum fazer esses nomes comunitários limitarem o acesso da capacidade do *SNMP* para usuários que não tenham permissão. Por meio do trecho do arquivo */etc/snmp/snmpd.conf* contido na Tabela 6.1, pode-se observar o uso desse termo aplicado para comunidade.

Atráves do trecho de configuração citado pela Tabela 6.1, pode-se então, acessar as *MIBs* dos computadores remotos para resgate de informações, que serão enviadas para o gerente *SNMP* o qual posteriormente repassa as mesmas para a aplicação.

Nesse trabalho, o gerente *SNMP* tem tanto a função de receber as informações dos agentes, quanto exibi-las por meio do uso da ferramenta de monitoramento e gerência de aglomerados de computadores.

No gerenciamento das MIBS, as informações de gerenciamento são vistas como variáveis e seus respectivos valores. As variáveis podem ser definidas de forma independente (individuais) ou na

Tabela 6.1: Configuração dos agentes *SNMP*

```
group context sec.model sec.level prefix read write notif
access notConfigGroup  any noauth exact systemview all all

sec.name source community
com2sec local localhost public
com2sec mynctwork 192.168.0/24 public

group.name sec.model sec.name
group MyRWGroup any local
group MyROGroup any mynetwork

incl/excl subtree mask
view all included .1 80

context sec.model sec.level prefix read write notif
access MyROGroup  any noauth 0 all all all
access MyRWGroup  any noauth 0 all all all
```

forma de tabelas. Variáveis individuais ou conjunto de variáveis (tabelas), são organizadas em grupos, os quais são constituídos e nomeados em função do grau de afinidade entre as informações. Uma determinada coleção de variáveis representa um módulo da *MIB*, que poderá ser constituída por diversos módulos.

Muitas informações de gerenciamento são abstrações, somente geradas quando requisitadas por uma determinada aplicação. Outras informações de gerenciamento são utilizadas como um argumento para iniciar ações em um sub-sistema gerenciado. Assim sendo, o termo *MIB* é conceitual, ou seja, não importa que tipo de armazenamento físico (memória principal, arquivos, base de dados, etc) é empregado para manter as informações de gerenciamento. Cada aplicação que desempenha o papel de agente tem

uma *MIB* associada, a qual compreende um ou mais módulos de informações de gerenciamento.

A identificação do conteúdo (coleção de variáveis e seus valores) de uma *MIB* pode ser iniciada mediante a identificação dos módulos que estão implementados, portanto, sendo controlados e/ou monitorados por um agente *SNMP*.

Para prover acessibilidade as informações resgatadas pelos agentes *SNMP*, foi desenvolvida a ferramenta de monitoramento e gerência de aglomerados de computadores. Dessa forma, para que as informações pudessem ser acessadas através da *web*, um *webserver* teve de ser configurado para prover hospedagem e tornar a aplicação acessível por um *webbrowser* conectado à internet.

O Apache é o servidor Web escolhido para esse trabalho por ser o mais usado: cerca de 60% do mercado utiliza o Apache, 20% o IIS da Microsoft e 20% utiliza outros (Sun/Netscape/AOL, etc)[39].

O Projeto Apache é o resultado do trabalho conjunto realizado por Universidades e Empresas (MIT, Berkeley, Stanford e empresas como IBM, Sun, HP, Compaq, RedHat)[25] e tem sido considerado lider em servidores web [39].

Suas principais características são as seguintes: é multiplataforma, robusto, tem boa performance e excelente documentação. Ele tem a vantagem em relação aos demais servidores, de possuir um código fonte completo, e uma licença irrestrita e melhor performance. É compatível com todas as especificações, permite

mudanças em suas características (flexibilidade) mesmo em suas partes mais internas (*core*) por ser implementado em módulos, e tem sua própria API padronizando toda programação interna.

PHP, *Personal Home Page*, ou, mais recentemente, um acrônimo recursivo para *"PHP: Hypertext Preprocessor"* é uma linguagem de programação de computadores interpretada, livre e muito utilizada para gerar conteúdo dinâmico na Web [26].

A linguagem surgiu por volta de 1994 como um subconjunto de scripts Perl criados por Rasmus Lerdof. As adições de Zeev Suraski e Andi Gutmans, dois programadores israelitas pertencentes ao Technion, o Instituto Israelita de Tecnologia, que reescreveram o parser, e assim, deram origem versão 3 lançada em 1997, sendo esta a primeira versão estável e similar à linguagem atual. Em maio de 2000 veio a público a versão 4, e em julho de 2004, a versão 5 [26].

Trata-se de uma linguagem extremamente modularizada, o que a torna ideal para instalação e uso em servidores web. É muito similar com a linguagem C. e com a C++, no que se refere a tipos de dados, sintaxe e mesmo funções. A partir da versão 5, inclusive, PHP conta com um suporte maior à orientação a objetos. PHP pode ser embutida no código HTML e além disso, destaca-se a extrema facilidade com que PHP lida com servidores de base de dados, como MySQL, Microsoft SQL Server e Oracle [26]. Vale lembrar ainda que existem versões do PHP disponíveis para ambientes Windows e Linux.

A interface gráfica da ferramenta de monitoramento e gerência de aglomerados de computadores é subdividida de duas partes (figura6.2):

(a) A primeira parte, à esquerda, destinada aos links para acesso à informações relacionadas aos computadores do aglomerado;

(b) A segunda parte destina-se à visualização das informações solicitadas.

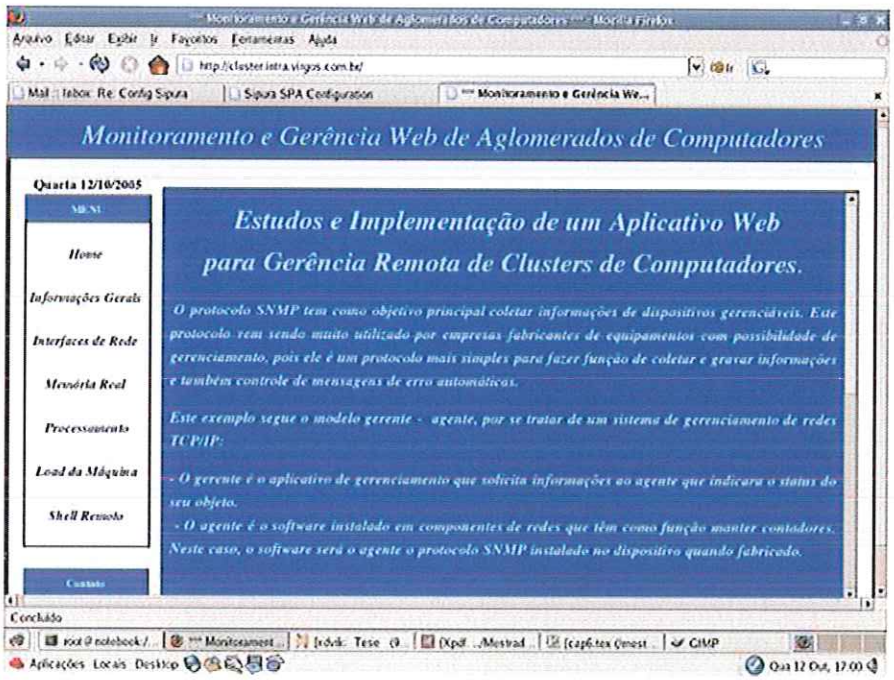

Figura 6.2: Interface Gráfica da Ferramenta

Através do acionamento do link de informações gerais à esquerda é exibido na área central algumas informações referentes a cada um dos computadores. Essas informações serão detalhadas no item 6.4.

4. Testes e correções - Como todo software, testes devem ser realizados, pois são de grande importância e determinam que os resultados providos pela aplicação possam ser utilizados de maneira confiável e segura. Através dos testes realizados observou-se algumas deficiências que foram corrigidas na aplicação e avaliadas novamente com novos testes.

5. Experimentos para avaliação da intrusão - A intrusão adotada nesse trabalho é conceitualizada e definida como sendo os atrasos que a execução da ferramenta de administração e gerência de aglomerados de computadores podem trazer aos serviços e processos do sistema.

6. Análise dos resultados obtidos da intrusão– Atráves dos experimentos comparativos realizados com outra ferramenta de mesma finalidade, avaliou-se o quanto de atrasos (grau de intrusão) essa ferramenta gerou.

6.4 Fluxo de Navegação

Através do diagrama de navegação, observar-se as funcionalidades da ferramenta de administração e gerência de aglomerados, o que facilita o entendimento sobre o que venha a ser abordado dentro de cada módulo da ferramenta. A figura 6.3 exibe um modelo esquemático para navegação e tem como finalidade representar, através do fluxo de navegação, as informações obtidas em cada sub-item da ferramenta.

Os módulos na figura 6.3 são enumerados através dos códigos T1, T2, T3, T4, T5, T6 e T7, que representam telas com as funcionalidades oferecidas aos clientes ou administradores do sistema para monitoramento e gerência de aglomerados de computadores. Essas informações são descritas nos itens seguintes:

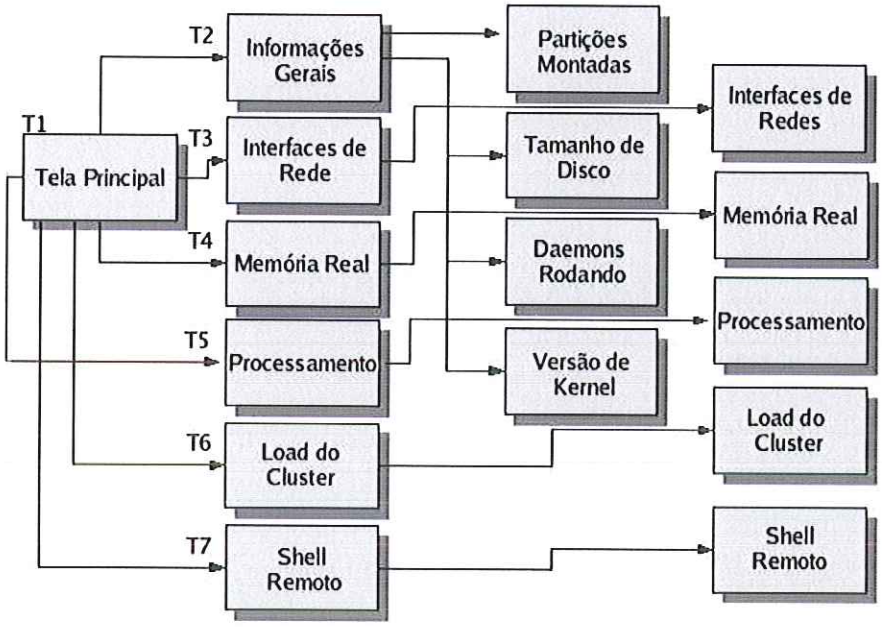

Figura 6.3: Modelo de esquemático de navegação da ferramenta de adminstração e gerência de aglomerados de computadores

- **T1:** - A tela T1 mostra a primeira janela da ferramenta, onde é apresentada uma breve descrição sobre as funcionalidades da ferramenta de monitoramento e gerência de aglomerados de computadores. Ainda pela tela T1 pode-se observar links à esquerda, que ao serem acionados, exibem as seguintes informações:

  1. Informações gerais relacionadas ao aspecto físico de cada computador;

2. Partições que estão montadas localmente ou remotamente;
3. Tamanho de disco do computador;
4. Estados de execução dos daemons;
5. Versão utilizada do kernel;
6. Endereço IP do computador, tipo do processador e informações do período de tempo que o computador se encontra em funcionamento.

A tela T1 pode ser observada na figura 8.1 no apêndice.

- **T2:** - A tela T2 exibe uma janela onde são apresentados aspectos gerais relacionados à estrutura física de cada computador dentro do aglomerado (figura 8.2 no apêndice). Resume-se tais características em: impressoras configuradas; discos espelhados; nome do fabricante do disco; informações referentes ao período em que os computadores estão em funcionamento; *interfaces* de redes existentes.

  Essa tela também contém links de páginas para visualização das partições montadas, tamanho real de disco, estrutura de daemons e versão do kernel, ilustradas em figuras que encontra no apêndice da maneira como segue:

  - Pela figura 8.3 pode ser visualizada a estrutura de partições montadas no computador; essas partições podem tanto ser partições locais quanto partições remotas, e podem ainda ser montadas por meio dos mais diferentes sistemas de arquivos.

- A figura 8.4 mostra a janela onde é apresentada o tamanho real do *hard disk* do elemento de processamento referenciado.
- A figura 8.5 mostra a janela onde é apresentada a estrutura de *daemons* que estão sendo gerenciados pelo sistema operacional de cada computador; esses daemons são representados através de processos. Tal estrutura tem como principal finalidade, exibir a origem do processo, ou seja, o caminho real da aplicação expresso também atráves do nome do processo, seguido pelo PID (*Process Identification*), sendo esse um valor numérico que faz referência ao nome do processo e o estado de execução do processo.
- A figura 8.6 apresenta a versão do kernel do elemento de processamento do cluster, sendo o núcleo do sistema operacional responsável pela gerência e funcionamento do computador, ou seja, faz a *interface* de comunicação entre hardware (impressora, monitor, mouse, teclado) e software (aplicativos em geral). Essa informação é importante, pois em cada versão são implementadas novas funcionalidades e melhorias nos processos já existentes.

- **T3:** - A tela T3 apresenta aspectos relacionados ao tráfego de rede gerado a partir das *interfaces* de redes listadas, podendo ser observada na figura 8.7 no apêndice. Após selecionar a *interface* desejada, essa nos exibe informações relacionadas à descrição da placa de rede, tamanho máximo dos diagramas suportados

pela placa, largura de banda da *interface*, *status* da *interface up/down*, taxa de descartes de entrada/saída, taxas de erros de entrada/saída e número de *bytes* enviados/recebidos. Essa página pode ser observada na figura 8.8 no apêndice.

- **T4:** - A figura 8.9, no apêndice, apresenta o *status* de ocupação de memória consumida versus a quantidade de memória real existente em cada computador do aglomerado. Pode-se então, monitorar o estado de ocupação de memória durante todo o processo de execução da tarefa.

- **T5:** - A figura 8.10, no apêndice, mostra a janela onde é apresentada o estado de ocupação de CPU por parte dos recursos alocados ao sistema, recursos alocados por processos de usuários do sistema e estado de ocupação livre de CPU. Dessa forma, verifica quanto de processamento está sendo utilizado pelo sistema, pelos usuários e a quantidade não utilizada por nenhum deles, ou estado de *idle* – livre para utilização.

- **T6:** - A figura 8.11, no apêndice, mostra a janela onde é apresentado o estado de carga em que os sistemas computacionais do aglomerado de computadores estão submetidos. Isso representa o quão ocupado esse sistema se encontra e conseqüentemente fornece o intervalo de tempo necessário para início de um novo processo no sistema.

- **T7:** - A figura 8.12, no apêndice, mostra a janela onde é apresentado o terminal SSH para conexão remota aos computadores

do aglomerado. O cliente ou administrador do sistema terá a capacidade de interação total com o ambiente.

6.5 Considerações Finais

Este capítulo apresentou toda a metodologia de implementação desse trabalho, bem como a descrição das funcionalidades empregadas na ferramenta de administração e gerência de aglomerados de computadores. Através do item 6.4 foi descrito todo o fluxo de navegação da ferramenta e apresentada a descrição de cada uma das páginas, proporcionando assim uma visão global das funcionalidades empregadas na ferramenta.

Capítulo 7

Avaliação de Intrusão e Desempenho

7.1 Considerações Iniciais

Este capítulo tem como principal objetivo analisar e validar a ferramenta de administração e gerenciamento de aglomerados de computadores implementada nesse trabalho, utilizando os resultados dos experimentos comparativos com a ferramenta Ganglia que tem características similares.

7.2 Aspectos de Configuração das Ferramentas

Para melhor auxiliar a compreensão e o entendimento, essa seção tem como objetivo apresentar os pacotes necessários para o funcionamento correto das ferramentas Ganglia e FAGAC. A seguir são apresentados os módulos que compõem as ferramentas e seus respectivos pacotes.

A escolha pelo Ganglia para experimentos comparativos com a fer-

ramenta FAGAC [1] se deu devido ao fato do Ganglia apresentar algumas funcionalidades similares às estruturas de desenvolvimento do FAGAC. Além disso, as linguagens adotadas entre as duas ferramentas proporcionam resultados similares para monitoração dos elementos de processamento, pois tem-se agentes que reportam dados a seus gerentes, esses, posteriormente, encaminham essas informações para o *frontend* para exibição dos resultados em um *webbrowser*.

7.2.1 Ganglia

A implementação do Ganglia pode ser dividida em duas partes principais ou dois *daemons*, os que resgatam informações provenientes dos computadores clientes e o *frontend* para exibição desses resultados. Os itens a seguir descrevem os pacotes necessários para o correto funcionamento das ferramemntas.

1. Gmod ou *Ganglia Monitoring Daemon - daemon* multitarefa que executa nos elementos de processamento do *cluster*, responsável por armazenar e coletar todas as informações provenientes de cada elemento. Acrescenta-se ainda para o seu funcionamento, a instalação do pacote *ganglia-gmod-3.0.2-1.i386.rpm* ou *ganglia-gmod-3.0.2-1.i386.tar.gz* [18], o primeiro baseado em distribuições RPM (*RedHat Package Management*) e o segundo trata do código fonte para compilação.

[1] Ferramenta de Administração e Gerência de Aglomerados de Computadores

2. Gmetad ou *Ganglia Meta Daemon* - trabalha na forma de árvore, sendo criada uma topologia em cada um dos elementos de processamento, recolhendo informações dos níveis $n + 1$. Para seu funcionamento é necessário a instalação do pacote *ganglia-gmetad-3.0.2-1.i386.rpm* ou *ganglia-gmetad-3.0.2-1.i386.tar.gz* [18];

3. RRDtool - armazena em base de dados circular uma seqüência de dados temporais de maneira compacta, e exibe gráficos mostrando a evolução dos dados armazenados. Para seu funcionamento é necessário a instalação do pacote *rrdtool-1.0.49-5.fc4.i386.rpm* ou *rrdtool-1.0.49-5.fc4.i386.tar.gz* [44, 45].

4. FrontEnd ou *Ganglia PHP Web FrontEnd* - apresenta as informações recolhidas pelo *gmod* e *gmedat* na forma de gráficos, sendo esses gerados a partir do RRDtool. Para seu funcionamento é necessário a instalação do pacote *ganglia-web-3.0.2-1.rpm* ou *ganglia-web-3.0.2-1.tar.gz* [18];

5. Instalação e configuração do *WebServer Apache* - para poder dar acesso aos clientes e administradores do sistema o *WebServer* é de fundamental importância, pois através dele é que o acesso às informações são visualizadas no *browser*. O arquivo principal de configuração do *Apache* é o */etc/httpd/conf/httpd.conf* [47], e é através desse arquivo que são especificados atributos gerais do servidor, como o número de porta, diretório no qual os diretórios do servidor *www* estão localizados, entre outras. Para o básico funcionamento do Apache é necessário a instalação do

pacote *httpd-2.0.54-10.2.i386.rpm* [44]. Devido à grande variedade de módulos suportados, é exibido na tabela 7.1 trecho da configuração.

Tabela 7.1: Configuração do WebServer Apache

Aspectos de Configuração	Comentário
Listen 80	porta em que o apache opera
Include conf.d/*.conf	diretório que contêm os arquivos de configuração
User apache	Usuário
Group apache	Grupo
ServerAdmin root@localhost	endereço para envio de alertas ou problemas
DocumentRoot "/var/www/html"	diretório www
DirectoryIndex index.php index.html index.htm	terminação arquivos index suportados
VirtualHost 200.245.158.170:80 ServerAdmin leonardo@virgos.com.br DocumentRoot /docs/sites/mmtlba ServerName 200.245.158.170:80 ServerAlias cluster.intra.virgos.com.br ErrorLog /var/log/httpd/error_log CustomLog /var/log/httpd/access_log ErrorDocument 404 VirtualHost	Virtual hosts Cria vários names para acesso a diferentes nomes do mesmo domínio

7.2.2 Ferramenta de Administração e Gerência de Aglomerados de Computadores

Para utilizar a ferramenta de administração e gerenciamento de aglomerados de computadores é necessário configurar os computadores que compõem o ambiente. Essa configuração é subdividida nas seguintes etapas:

1. Instalação do protocolo SNMP(*Simple Network Management Protocol*) no servidor. A instalação deve ser feita através do pacote

net-snmp-5.2.1.2-fc4.1.i386.rpm ou mesmo através de seu código fonte *net-snmp-5.2.1.2.tar.gz*. Além disso é preciso a instalação dos pacotes auxiliares *net-snmp-libs-5.2.1.2-fc4.1.i386.rpm, net-snmp-devel-5.2.1.2-fc4.1.i386.rpm* [44].

2. Instalação do SNMP nos clientes que compõem o *cluster*, seguido da instalação do pacote *net-snmp-utils-5.2.1.2-fc4.1.i386.rpm* [44], que é de fundamental importância, pois esse pacote contém os comandos para resgate das informações provenientes das mibs.

3. Configuração do cliente-servidor ou gerente-agente. Para que as informações possam ser trocadas entre agentes-gerentes, ressalta-se que o nome comunitário deve ser conhecido como sendo senhas para acesso às informações dos agentes comunitários. Através do trecho de configuração da Tabela 6.1 na seção 6.3 pode-se observar o uso do termo aplicado. Após a instalação e configuração é necessário iniciar o serviço através do comando *service snmpd start* ou */etc/init.d/snmpd start*.

4. Instalação e configuração da linguagem PHP (*Hypertext Preprocessor*). Para que os *scripts* php desenvolvidos para a ferramenta FAGAC possam ser interpretados através de uma requisição cliente, é necessário que o servidor tenha suporte para interpretar tais solicitações e processá-las devolvendo ao cliente somente as informações requisitadas. Para isso, os seguintes pacotes devem ser instalados: *php-5.0.5-2.1.i386.rpm* e o *php-snmp-5.0.5-2.1.i386.rpm* [44] ou pacotes com código fonte para compilação.

Terminada a instalação deve-se então, configurar a linguagem PHP através do arquivo */etc/php.ini*. Devido às várias funcionalidades oferecidas vale ressaltar a necessidade de mudar o parâmetro *"register_globals"* para *on* [20].

5. Instalação dos *scripts* desenvolvido para o FAGAC, colocando-os no diretório padrão do Apache, sendo o diretório */var/www/html* determinado através do parâmetro do Apache *DocumentRoot* [47]. Aspectos relacionados à configuração do Apache podem ser vistos na seção 7.2.1 acima.

6. Instalação e configuração da ferramenta *JpGraphs* através do pacote *jpgraph-2.0.tar.gz*. Essa ferramenta proporciona o suporte para a geração dos gráficos através das informações resgatadas dos clientes pelo protocolo SNMP.

O sistema operacional utilizado para todo o desenvolvimento e experimentos comparativos entre as duas ferramentas foi o Linux/Fedora na versão 4.

7.3 Configuração do Ambiente

Na realização dos experimentos foi utilizado um notebook Acer Aspire 3000 com as seguintes configurações:

- Processador

```
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 15
model           : 28
model name      : Mobile AMD Sempron(tm) Processor 2
stepping        : 0
cpu MHz         : 1607.466
cache size      : 256 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
```

- Memória:

```
MemTotal:       449088 kB
SwapTotal:      498004 kB
```

- Versão Kernel:

```
Linux notebook_2.6.13-1.1532_FC4 #1 Thu Oct 20 01:30:08 EDT 2005 i686 athlon i386 GNU/Linux
gcc version 4.0.1 20050727 (Red Hat 4.0.1-5)
```

- Disco Rigído:

```
LABEL=/                 /                       ext3    defaults        1 1
LABEL=/boot             /boot                   ext3    defaults        1 2
/dev/devpts             /dev/pts                devpts  gid=5,mode=620  0 0
/dev/shm                /dev/shm                tmpfs   defaults        0 0
/dev/proc               /proc                   proc    defaults        0 0
/dev/sys                /sys                    sysfs   defaults        0 0
/dev/hda3               swap                    swap    defaults        0 0
/dev/hdc                /media/cdrecorder       auto    pamconsole,exec,noauto,managed 0 0
```

Para se comparar as duas ferramentas é necessário identificar anteriormente quais são os pontos comuns existentes entre as mesmas. Na Tabela 7.2 são apresentados aspectos gerais sobre as funcionalidades de cada ferramenta.

Os experimentos e os resultados obtidos da comparação serão descritos na próxima seção.

Tabela 7.2: Funcionalidades apresentadas por cada ferramenta

Funcionalidades Monitoradas	Ganglia	FAGAC
Quantidades de CPUs	X	–
Fabricante da CPU	–	X
Host: Up/Down	X	–
Avg Load	X	X
Local Time	X	X
UpTime:	–	X
Ordenação Cresc./ Decresc.	X	–
Estado do Processos	–	X
CPU utilizado pelo sistema	X	X
CPU utilizado pelos usuários	X	X
CPU livre para utilização	X	X
Memória Usada	X	X
Memória Livre	X	X
Memória Shared	X	–
Memória Cached	X	–
Memória Buffered	X	–
Memória Swapped	X	–
Memória Total	X	–
Tamanho de Disco	X	X
Versão de Kernel	–	X
Partições locais e remotas	–	X
Endereço IP	–	X
Tráfego das interfaces de rede	X	X
Informações gerais de hardwares	X	X
Terminal Shell	–	X
Hostname	X	X

Através das funcionalidades acima descritas e da análise das funcionalidades das ferramentas foi possível encontrar apenas cinco funcionalidades no código fonte das aplicações em comuns, entre o FAGAC e o Ganglia, sendo esse alvo de experimentos na próxima seção.

7.4 Experimentos e Resultados

Os experimentos consistiram em determinar a quantidade de ocupação de CPU para cada funcionalidade analisada, considerando-se apenas resultados com valor diferente de valor nulo. Esse experimentos foram realizados para as seguintes cinco funcionalidades comuns, escolhidas entre as ferramentas Ganglia e a FAGAC especificadas na tabela 7.2 : carga de processamento, quantidade de memória principal livre, espaço em disco rígido, percentual de utilização de CPU para usuários e percentual de CPU livre.

Para a realização dos experimentos, foram feitas alterações no código fonte de ambas as ferramentas, para que as mesmas pudessem retornar a informação do estado de ocupação de CPU.

Na ferramenta Ganglia foi feita uma alteração no código do *daemon* **gmond.c**, e foi inserida uma função denominada *clock* que é responsável por determinar o tempo aproximado utilizado pelo processador. Uma alteração similar foi feita no arquivo **snmpd.c** para o FAGAC.

Os resultados da tabela 7.3 foram obtidos executando-se os dois *daemons* snmpd e gmond responsáveis por determinarem os valores de ocupação de CPU.

Para cada funcionalidade foram feitos 100 experimentos, e foi determinada a média e o desvio padrão dos resultados (DP), para cada ferramenta.

Tabela 7.3: Média e Desvio Padrão do estado de ocupação da CPU de cada funcionalidade

Funcionalidade	Ganglia (Média/DP)	FAGAC (Média/DP)
carga de processamento	$7667 \pm 43,02$	$1100 \pm 30,33$
memória principal livre	$1667 \pm 37,90$	$1237 \pm 29,93$
espaço em disco rígido	$8667 \pm 34,57$	$1033 \pm 30,45$
percentual de CPU	$6806 \pm 43,77$	$5065 \pm 45,53$
percentual de CPU livre	$7968 \pm 38,25$	$2742 \pm 36,12$

Os resultados apresentados na tabela 7.3 permitem observar que o FAGAC teve menor ocupação nas cinco das funcionalidades analisadas em comparação à ferramenta Ganglia.

Para comparar os custos de processamento das cinco funcionalidades comuns às ferramentas Ganglia e FAGAC foram aplicados testes de hipótese. A teoria sobre teste de hipótese é apresentada no Apêndice II.

O intuito desse teste foi o de detectar os casos em que a ferramenta FAGAC gera menores custos que o Ganglia.

Baseando-se na teoria sobre teste de hipótese foi escolhida a hipótese científica como sendo a seguinte:

$$\begin{cases} H_0: \text{ FAGAC tem o mesmo custo computacional que Ganglia} \\ H_1: \text{ FAGAC tem menor custo computacional} \end{cases}$$

A hipótese estatística é:

$$\begin{cases} H_0: \ \mu_{FAGAC,i} = \mu_{Ganglia,i} \\ H_1: \ \mu_{FAGAC,i} < \mu_{Ganglia,i} \end{cases}$$

Onde i é a funcionalidade avaliada.

Os resultados de testes de hipótese unilaterais são apresentados na tabela 7.4, juntamente com as médias de cada funcionalidade já apresentadas na tabela 7.3.

Tabela 7.4: Testes de hipótese para as cinco funcionalidades comuns

Funcionalidade	Ganglia (média)	FAGAC (média)	Unilateral
carga de processamento	7667	1100	0,0904
memória principal livre	1667	1237	0,3498
espaço em disco rígido	8667	1033	0,1431
% de utilização de CPU	6806	5065	0,2964
% de CPU livre	7968	2742	0,1637

A coluna unilateral ou teste de hipótese monocaudal é utilizada para definir uma hipótese que será considerada como aceitável, caso a hipótese nula seja rejeitada. E dentro das amostras coletadas dos experimentos, podemos observar pelos resultados apresentados na tabela 7.4 que essa hipótese ocorre nas cinco funcionalidades da ferramenta FAGAC, apresentando assim menor custo computacional para resgatar as informações, sendo essas respectivamente:

- Carga de processamento
- Memória principal livre
- Espaço de disco rígido
- Percentual de CPU
- Percentual de CPU livre

Na figura 7.1 pode-se observar os tempos médios de resposta das ferramentas para resgate da funcionalidades constatando de maneira gráfica as conclusões já apresentadas, referentes à tabela 7.3.

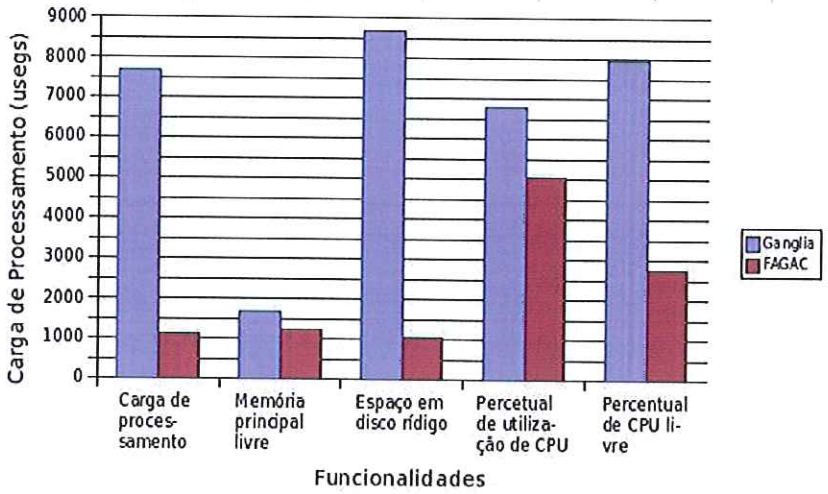

Figura 7.1: Tempos médios de resposta para resgate da informações

A partir dos resultados apresentados na coluna denominada unilateral da tabela 7.4 pode-se também estatísticamente comprovar que:

- $0,0904$ de 100 experimentos representam $90,96\%$ dos resultados de *carga de processamento.* Tal resultado indica que a ferramenta FAGAC tem menor custo que o Ganglia para resgatar a informação;

- $0,3498$ de 100 experimentos representam $65,02\%$ dos resultados de *memória principal livre.* Isso comprova que a ferramenta FAGAC tem menor custo para resgatar a informação do que o Ganglia;

- $0,1431$ de 100 experimentos representam $85,69\%$ dos resultados de *espaço de disco rígido.* Isso comprova que a ferramenta FAGAC tem menor custo para resgatar a informação do que o Ganglia;

- 0,2964 de 100 experimentos representam 70,36% dos resultados de *percentual de utilização de CPU*. Isso comprova que a ferramenta FAGAC tem menor custo para resgatar a informação do que o Ganglia;

- 0,1637 de 100 experimentos representam 83,63% dos resultados de *percentual de CPU livre*, o que indica que a ferramenta FAGAC tem menor custo que o Ganglia para resgate da informação;

7.5 Considerações Finais

Este capítulo apresentou resultados de simulações e experimentos comparativos de cinco funcionalidades comuns entre as ferramentas Ganglia e FAGAC. Os resultados dos experimentos comprovaram o menor consumo computacional por parte do FAGAC nas seguintes funcionalidades: (*carga de processamento, memória principal livre, espaço em disco rígido, percentual de utilização de CPU e porcentagem de CPU livre*) em comparação à ferramenta Ganglia.

Uma das principais características do FAGAC é o baixo tempo de resposta para resgate das informações no ambiente, tornando-se uma ferramenta vantajosa em ambientes de monitoramento e gerência de aglomerados de computadores.

Capítulo 8

Conclusões

8.1 Conclusões sobre o Projeto

Nesse trabalho foi apresentada uma nova ferramenta para administração e gerência de aglomerados de computadores, denominado FAGAC (Ferramenta de Administração e Gerência de Aglomerados de Computadores). Além de apresentar as funcionalidades existentes em ferramentas similares, essa ferramenta proporciona uma interface web para interação do cliente ou administrador com os elementos de processamento.

Na implementação dessa ferramenta de adminstração e gerência de aglomerados de computadores (FAGAC) houve a preocupação de minimizar os recursos computacionais a fim da ferramenta ser o menos intrusiva em ambientes distribuídos, maximizando a ocupação desses recursos.

Essa ferramenta foi avaliada por meio de experimentos comparativos com a ferramenta Ganglia, comprovando que a mesma apresenta

redução nos tempos médios de resposta das funcionalidades resgatadas, e menor consumo dos recursos computacionais, comprovando ser menos intrusiva que a ferramenta Ganglia. Em resumo, essas são as principais contribuições resultantes deste projeto de mestrado aliada às funcionalidades de administração dos aglomerados.

8.2 Trabalhos Futuros

As contribuições apresentadas neste projeto também motivam, como trabalhos futuros, o armazenamento das informações resgatadas em uma base de dados e a possibilidade de monitoramento através de *Grid Computing*. Assim, poder-se-ia implementar um histórico do desempenho dos elementos de processamento, podendo monitorar e gerenciar *Grids Computing*.

Referências Bibliográficas

[1] Mello, R.F., Paiva, M. S., Silva, L.M.R., Trevellin, L.C., Barbosa L.A.F. Cluster Linux.
Disponivel em: http://www.revistadolinux.com.br/ed/048 /assinantes/cluster.php.
Acessado em: 26 Dez 2004.

[2] Mello, R.F, Proposta e Avaliação de Desempenho de um Algoritmo de Balanceamento de Carga para Ambientes Distribuídos Heterogêneos Escaláveis.Tese (Doutorado) Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo.

[3] Mello, R.F., Paiva, M. S., Trevellin, L.C. (2003). Comparative Analysis ofthe Prototype and the Simulator of a New Load Balancing Algorithm for Heterogeneous Computing Environment, aceito para publicação no Internacional Jornalof High Performance Computing and Network (IJHPCN), ISSN 1740-0562.

[4] Boas, P.R.V, Travieso Gonzalo. Workshop de Computação de Alto Desempenho, WSCAD 2004 - Uma ferramenta Orientada a Objetos para Monitoramento de Cargas em Sistemas Distribuídos - Foz do Iguaçu, de 27 a 29 de Outubro de 2004.

[5] D. Fanagan. Java in a Nutshell. O'Reilly, 3nd edition, May 1997.

[6] W.Grosso. Java RMI. O'Reilly & Associates, Inc., Sebastopol, CA. 2002.

[7] RFC.net repository of RFC STD BCP and FYI documents.
Disponivel em: http://rfc.net.
Acessado em: 27 Dez 2004.

[8] Modelo OSI de Arquitetura.
Disponível em: http://www.geocities.com/siliconvalley
Acessado em: 28 Dez 2004.

[9] Gerenciamento de Redes Teleinformática.
Disponível em: http://www.gta.ufrj.br/grad/991/rodrigo/geredes.htm.
Acessado em: 27 Dez 2004.

[10] Nacamura. L.J., Garcia O, Ferrasa A , Althaus R. A Desenvolvimento de um agente SNMP embarcado para o gerenciamento de carga em redes de distribuição elétrica.

[11] Modelo Agentes-Gerentes SNMP.
Disponível em: http://www.canon.com/bctv/canobeam
Acessado em: 29 Dez 2004.

[12] Aspectos Introdutórios do Protocolo CMIP
Disponível em:http://www.gta.ufrj.br/grad/cmip.html/snmp
Acessado em: 30 Dez 2004.

[13] BoxServer Internet Simples e Fácil
Disponível em: http://www.boxserver.com.br/cluster.htm
Acessado em: 4 Jan 2005.

[14] Tutorial de TCP/IP-UDP.
Disponível em: http://www.juliobattisti.com.br.
Acessado em: 30 Dez 2004.

[15] Cluster - Principais Conceitos
Disponível em:http://www.infowester.com/cluster.php
Acessado em: 4 Jan 2005.

[16] Sistemas de Processamento Distribuído
Disponível em:http://pt.wikipedia.org/wiki/Sistemadeprocessamento distribuído.html
Acessado em: 4 Jan 2005.

[17] GNU Bis MOSIX
Disponível em:http://www.gnubis.com.br/bin/view/VESLAC/Mosix
Acessado em: 4 Jan 2005.

[18] Ganglia
Disponível em:http://ganglia.sourceforge.net/docs/name
Acessado em: 8 Jan 2005.

[19] Puchong Uthayopas, Surachai Paisitbenchapol, Thara Angskun, and Jullawadee Maneesilp. System management framework and tools for beowulf cluster. In The Fourth International Confe-

rence/Exhibition on High Performance Computing in the Asia-Pacific Region, pages 935940 vol. 2, November 2000.

[20] PHP Hypertext Preprocessor
Disponível em:http://www.php.net
Acessado em: 13 Fev 2005.

[21] Rajkumar Buyya, Krishna Mohan, and Bindu Gopal. Parmon: A comprehensive cluster monitoring system. In Proceedings of the International Conference on High Performance Computing on Hewlett-Packard Systems (HiPer-98), Junho 1998.

[22] Rede Segura Linux, James Stanger, Ph.D. , Patrick T. Lane, Edgar Danielyan. Alta Books.

[23] Céline Boutros Saab, Xavier Bonnaire, and Bertil Folliot. A flexible monitoring platform to build cluster management services. In IEEE Cluster2000 Conference: Technology and Applications, pages 75-85, Janeiro 2002.

[24] Neves Marcelo, Scheid Tiago, Schnorr Mello Lucas. Integração de Ganglia, libRastro e Pajé para o Monitoramento de Aplicações Paralelas. 2004. Universidade Federal de Santa Maria Campus UFSM - Santa Maria - RS - Brasil.

[25] O Linux – Seu site de Linux na Internet
Disponível em : http://www.linuxsolutions.com.br
Acessível em: 16 Nov 2005.

[26] PHP - Wikipedia
Disponível em : http://www.wikipedia.org
Acessível em : 16 Nov 2005.

[27] Ferreto Tiago, De Rose César. Rvision: uma ferramenta aberta e configurável para monitoração de clusters. Pontifícia Universidade Católica de Rio Grande do Sul.

[28] Beowulf - Overview - History
Disponível em:http://www.beowulf.org/overview/history.html
Acessado em: 13 Fev 2005.

[29] Cluster Computing White Paper Status Final Release Version 2.0 Date 28th December 2000 Editor Mark Baker, University of Portsmouth, UK

[30] openMosix - a Free Softwar Linux Cluster Project
Disponível em:http://laurel.datsi.fi.upm.es/ rpons/openmosix
Acessado em: 13 Fev 2005.

[31] Casavant, T. L.; Kuhl, J.G. (1988). Taxonomy of Scheduling in Distributed Computing Systems. IEEE Transactions on Software Engineering, v. 14, n.2, p. 141-154, fev.

[32] Tanembaum, A. (1994). Distributed Operating Systems. Prentice Hall.

[33] Guia do Administrador do Sistema Linux
Autor : Rubens E. Ferreira ISBN: 85-7522-038-1 Ano: 2003

[34] Netto Marco A. S. , Barcelos Alex V. , De Rose César A. F., Desenvolvimento de Sistemas de Gerência de Aglomerados para Plataforma GNU/Linux. Pontifícia Universidade Católica do Rio Grande do Sul.

[35] Shivarati, N. G., Krueger, P. and Singhal, M. (1992). Load Distributing for Locally Distributed Systems. IEEE Computer, p. 33-44, dez.

[36] Whately, L., Pinto, R., Rangarajan, M., Iftode, L., Bianchini, R., and Amorim, C. L. (2001). Adaptive techniques for home-based software dsms. In 13th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2001), pages 164-171, Pirenópolis, Goiás, Brazil.Whately, L., Pinto, R., Rangarajan, M., Iftode, L., Bianchini, R., and Amorim, C. L. (2001). Adaptive techniques for home-based software dsms. In 13th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2001), pages 164-171, Pirenópolis, Goiás, Brazil.

[37] Gamma, E.; Helm, R.; Vlissides, J.; Johnson, R. Design patterns: elements of reusable object-oriented software . Addison-Wesley, 1994.

[38] GNU General Public License
Disponível em: http://www.gnu.org/copyleft/gpl.html/SEC1
Acessado em: 13 Fev 2005.

[39] Netcraft
Disponível em http://www.netcraft.com
Acessado em 16 Nov 2005.

[40] Puchong Uthayopas, Surachai Paisitbenchapol, Thara Angskun, and Jullawadee Maneesilp. System management framework and tools for beowulf cluster. In The Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region, pages 935-940 vol. 2, November 2000.

[41] Maya; Asmita; Anuradha; Snehal; Krushna. Migshm Project Report
Disponível em: http://mcaserta.com/maask/Migshm-Report.pdf
Acessado em: 16 Jun 2005.

[42] G.J da Silva and B. de Oliveira Stein. Uma biblioteca genérica de geração de rastros de execução para visualização de programas. Anais de I Simpósio de Informática de Região Centro, 2002.

[43] J. Chassim de Kergommeaux and B. de Oliverira Stein. Pajé: An extensible environment for visualizind multi-theaded programs executions. Lecture Notes in Computer Science, 133-153, 2001.

[44] About Rpmfind.Net WWW Server a.k.a. Rufus.W3.Org Disponível em: http://www.rpmfind.net
Acessado em: 16 Nov 2005.

[45] RRDtool
Disponível em: http://www.fastmirrors.org/rrdtool
Acessado em: 18 Nov 2005.

[46] Pitanga, Marcos Computação em Cluster: O Estado da Arte da Computação. Rio de Janeiro: Brasport, 2003.

[47] Welcome ! The Apache Software Foundation
Disponível em: http://www.apache.org
Acessado em: 19 Nov 2005.

[48] Ha, Bao e Nquyen, Tina "Slackware Linux Unleashed"Editora Sans, Dezembro de 1999.

[49] Yokokura, Alex Yuichi, Estudo de Viabilidade da Implantação de Técnicas de Cluster ao Projeto Servidor de Estações de Trabalho (SET), Fevereiro de 2005.

[50] P. L. Meyer, Probabilidade: Aplicações à Estatística, segunda edição, ano de 1983, Livros técnicos e Científicos Editora.

[51] M. N. Magalhães e A. C. P. De Lima, 2001, Noções de Probabilidade e Estatística 3 edição, Editora USP.

Apêndice I

Figura 8.1: Modelo de desenvolvimento da Ferramenta de Monitoramento e Gerência de Aglomerados de Computadores

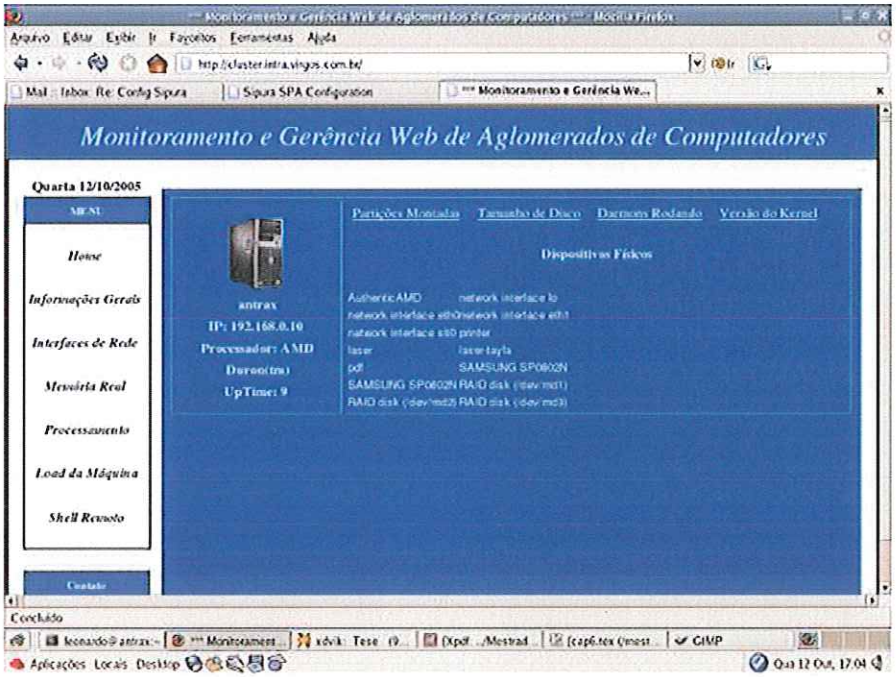

Figura 8.2: Informações da Arquitetura Física

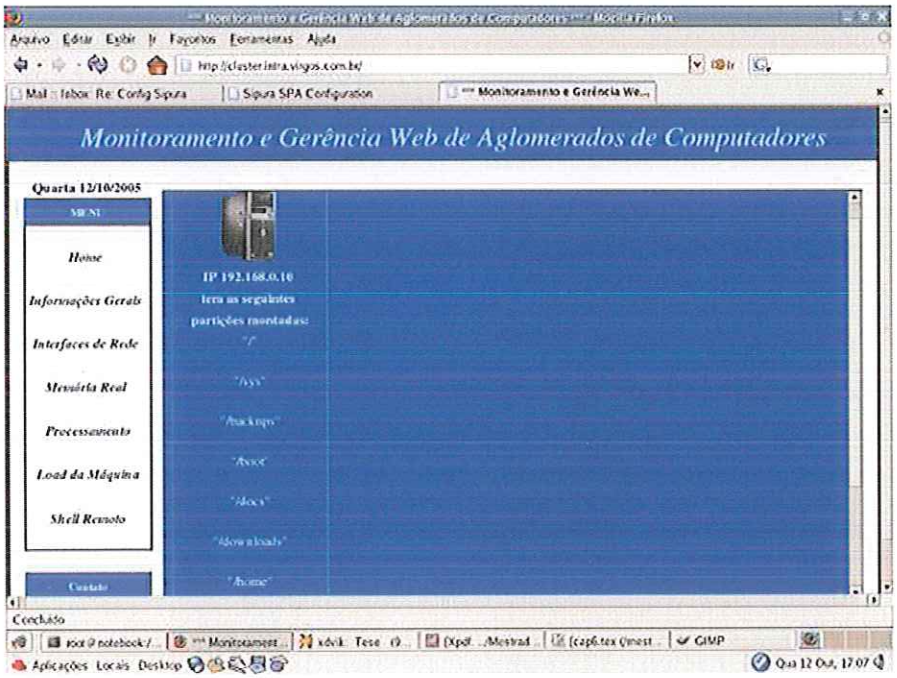

Figura 8.3: Partições Montadas no Computador

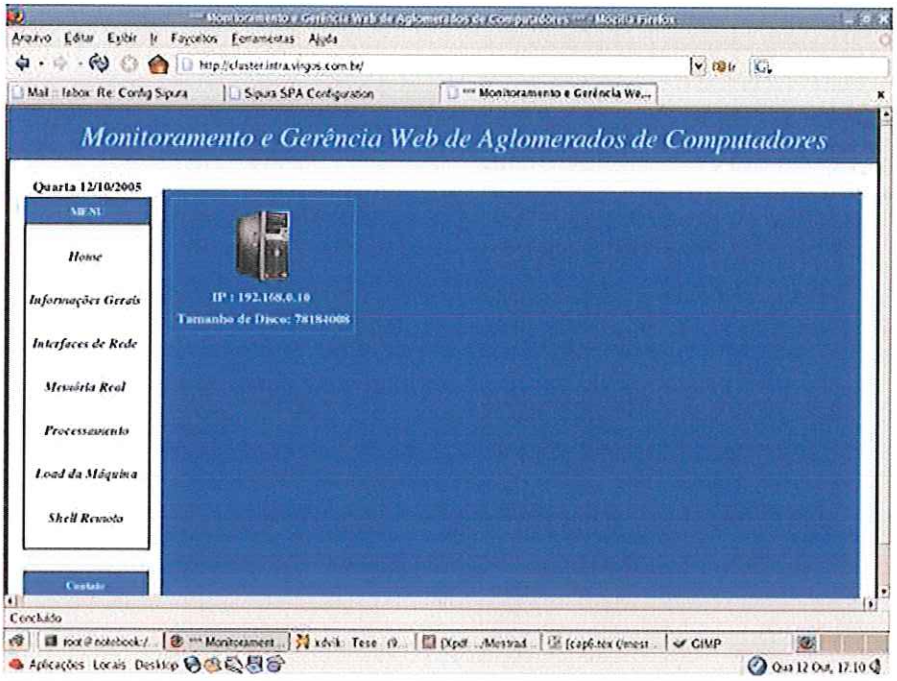

Figura 8.4: Tamanho Físico do Disco

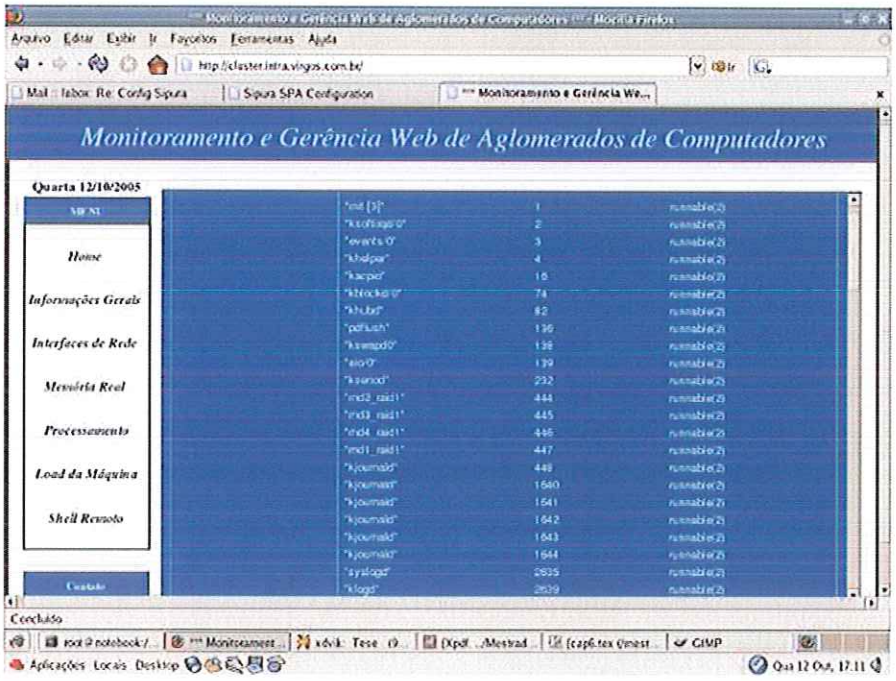

Figura 8.5: Daemons Rodando

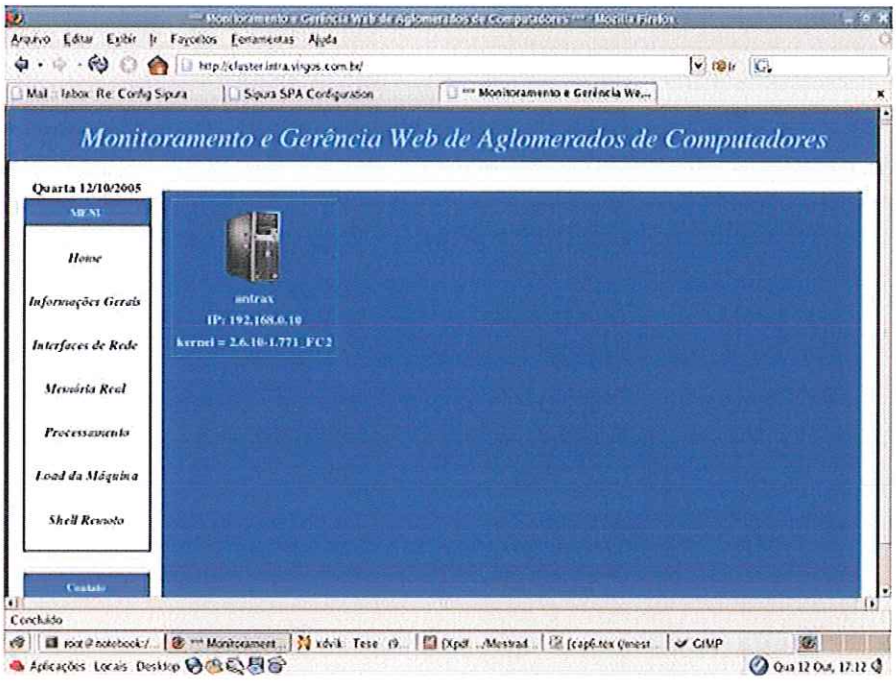

Figura 8.6: Versão do Kernel utilizada

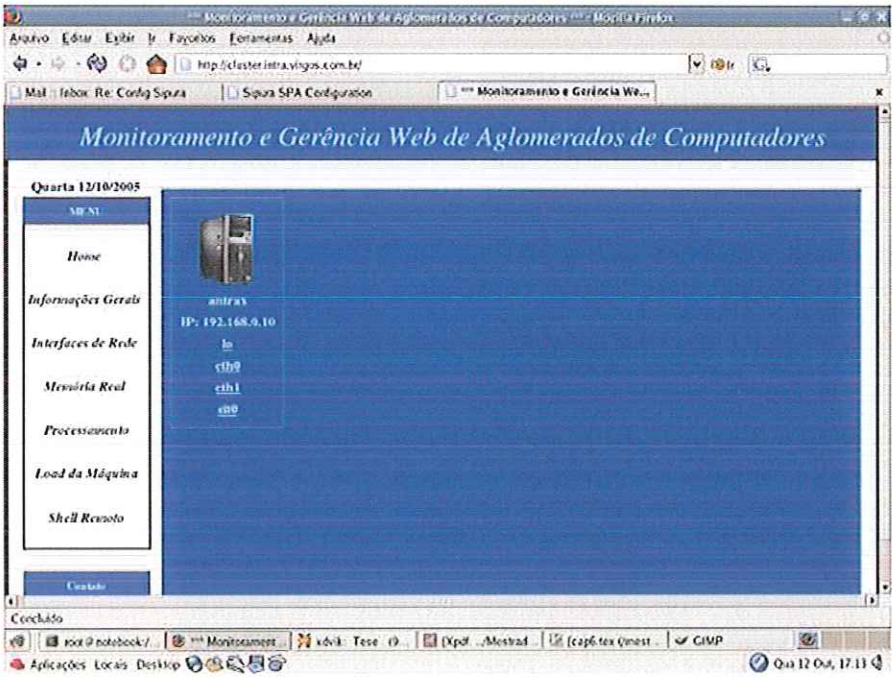

Figura 8.7: Interfaces de Redes

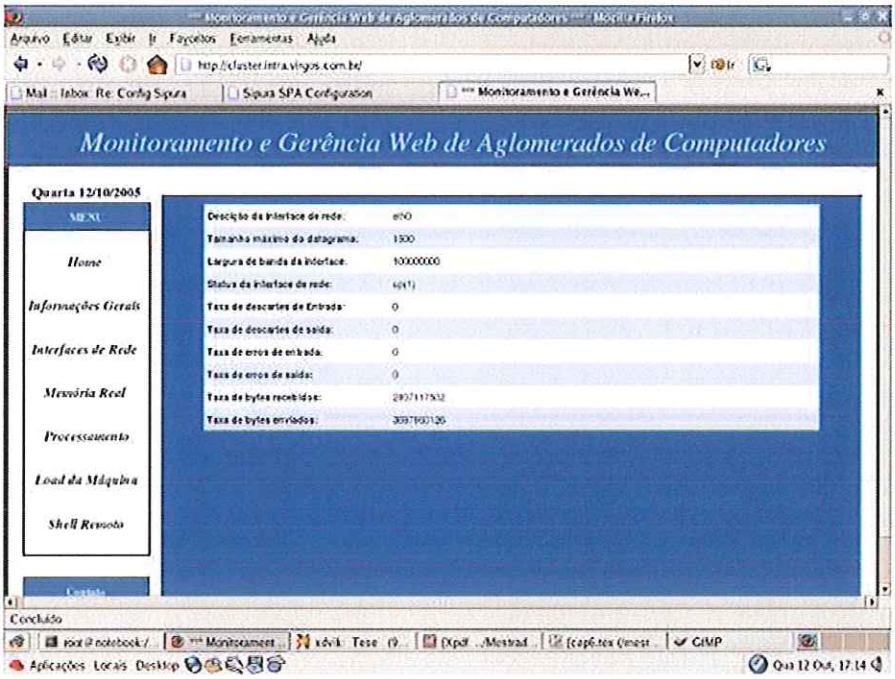

Figura 8.8: Status da Interface de Rede

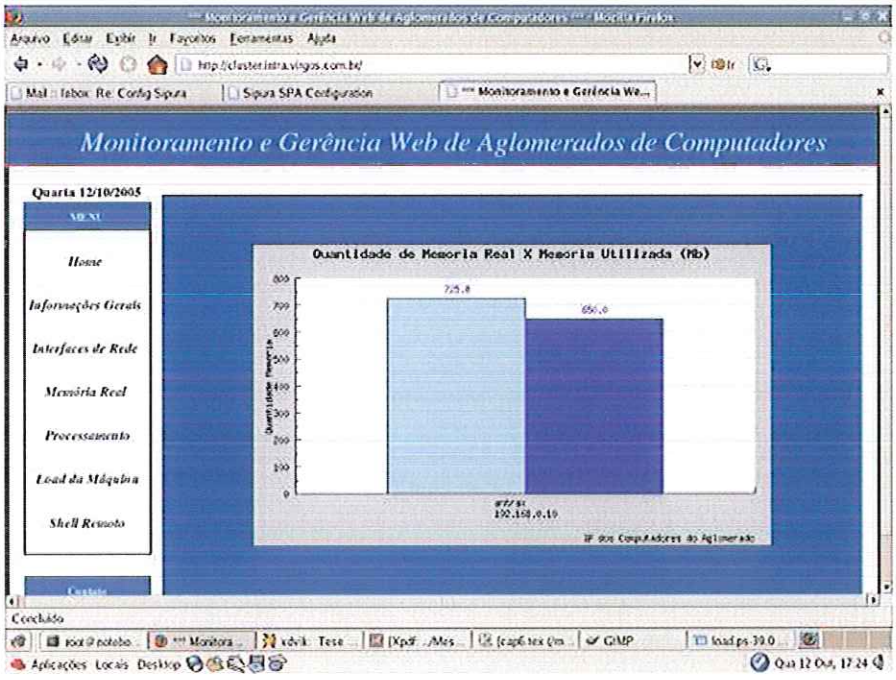

Figura 8.9: Informações da Quantidade de Memória X Consumida

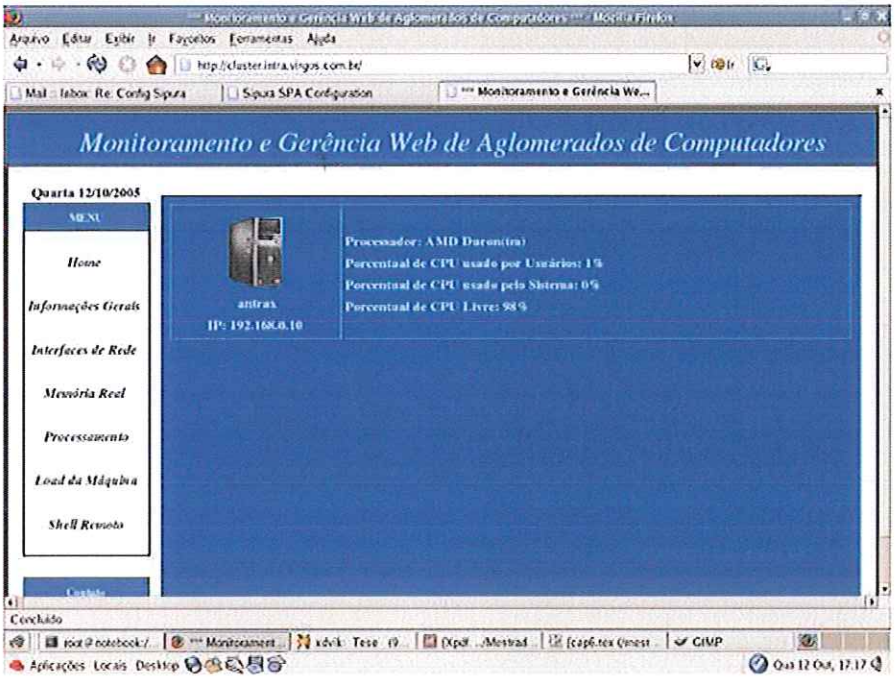

Figura 8.10: Informações do Status de CPU

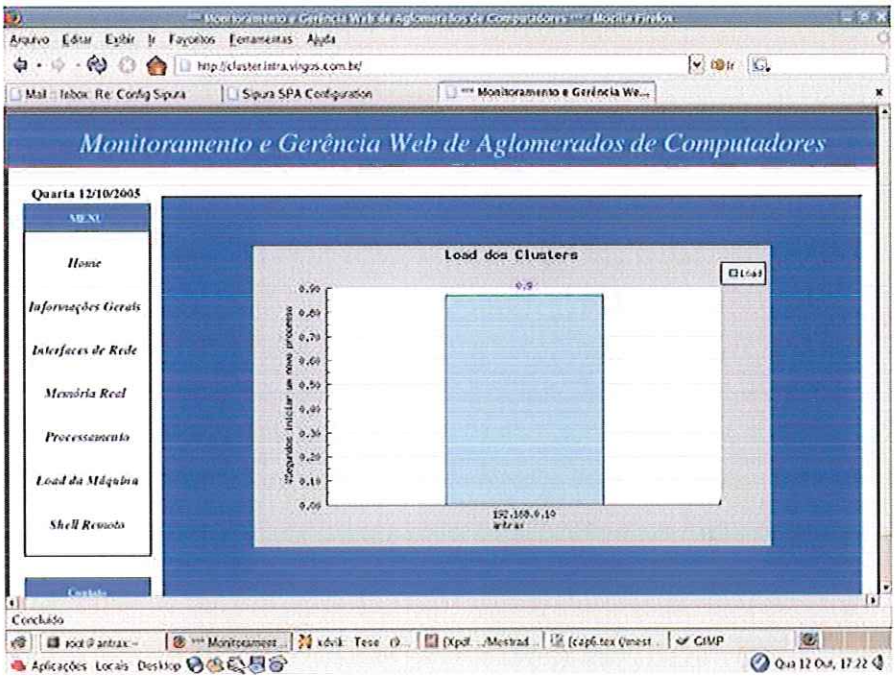

Figura 8.11: Informaçõcs do status Load

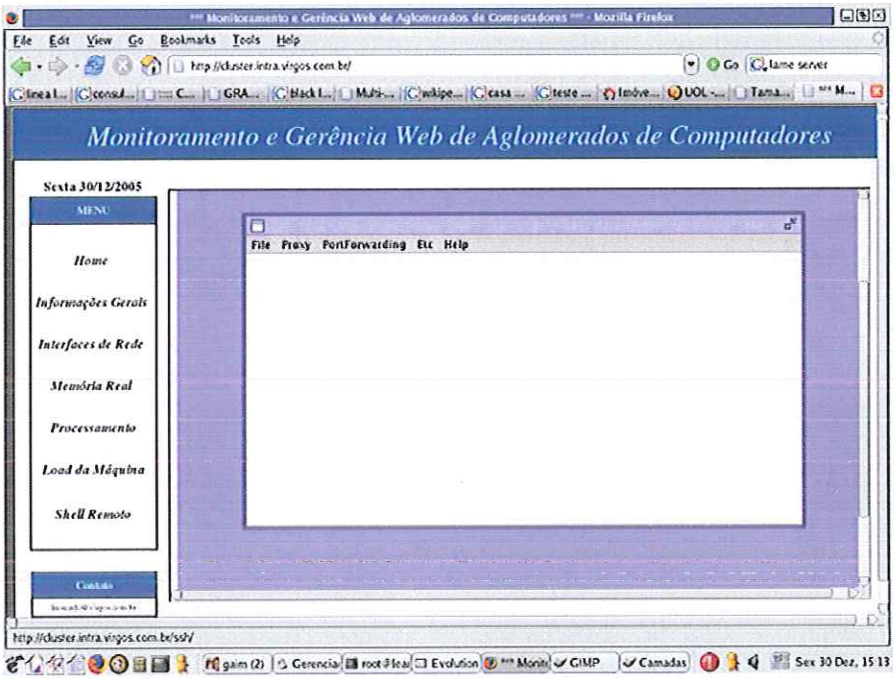

Figura 8.12: Shell para Administração do Sistema

Apêndice II

Para a validação da eficiência da ferramenta FAGAC em comparação ao Ganglia, foi utilizado teste de hipótese. Teste de hipótese esse que permite verificar qual ferramenta apresenta maior custo de processamento. Essa verificação é chamada de hipótese científica. A partir de um hipótese científica obtém-se a hipótese estatística [51, 50].

A hipótese estatística é uma asserção ou conjectura sobre um parâmetro, ou parâmetros. Pode também referir-se ao tipo, ou características de uma população, em que a população é um conjunto de resultados de experimentos executados.

Acrescenta-se ainda que uma hipótese estatística, denominada como hipótese nula – denotada por H_0 –, é usada para analisar alguma conjectura estabelecida inicialmente. Essa hipótese pode ser rejeitada ou não. A idéia de se estabelecer uma hipótese nula é comum mesmo em um raciocínio não-estatístico. Esses testes são adotados em processos criminais, em que um acusado (réu) é presumido ser inocente até que se prove o contrário [51, 50]. A pressuposição da inocência é uma hipótese nula.

Para desenvolver procedimentos para testar hipóteses estatísticas

deve-se sempre conhecer, exatamente, o que esperar quando uma hipótese é verdadeira.

A hipótese utilizada como alternativa nula, isto é, a hipótese aceita quando a nula é rejeitada, é denominada hipótese alternativa, denotada por H_1. Assim nesse exemplo do réu, formulam-se as hipóteses:

$$\begin{cases} H_0: \text{ O réu é inocente} \\ H_1: \text{ O réu é culpado} \end{cases}$$

Ao realizar testes de hipótese, pode-se cometer dois tipos de erros:

1. Rejeitar a hipótese H_0, quando ela é verdadeira.
2. Não rejeitar a hipótese H_0, quando ela é falsa.

A tabela 8.1 resume as situações acima.

	Aceitar H_0	**Rejeitar H_0**
H_0 verdadeira	decisão correta	Erro tipo I
H_0 falsa	Erro tipo II	decisão correta

Tabela 8.1: Tipos de erros

Se a hipótese H_0 é verdadeira e aceita, ou falsa e rejeitada, a decisão é correta; se H_0 é verdadeira e rejeitada, ou falsa e aceita, a decisão é errada. O primeiro desses erros é chamado de Erro Tipo I que é a probabilidade de cometê-lo denotada pela letra grega α (alfa); o segundo, chamado de Erro Tipo II, probabilidade de cometê-lo denotada pela letra grega β (beta).

Quando o erro tipo I é cometido, significa que a probabilidade de ocorrer esse evento é de α, por isso é fixado um valor baixo para o

nível de significância.

Dependendo das hipóteses alternativas, os valores críticos fornecem um critério de teste. Exemplo desse critério pode ser observado pelas figuras 8.13 , 8.14 e 8.15, que apresentam critérios para testes unilaterais e bilaterais, respectivamente.

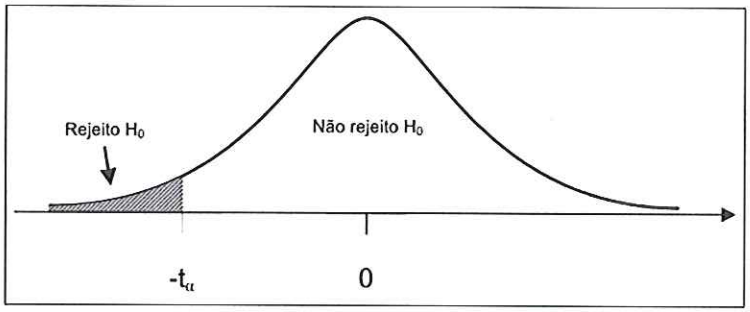

Figura 8.13: Hipótese Alternativa $\mu < \mu_0$

Figura 8.14: Hipótese Alternativa $\mu > \mu_0$

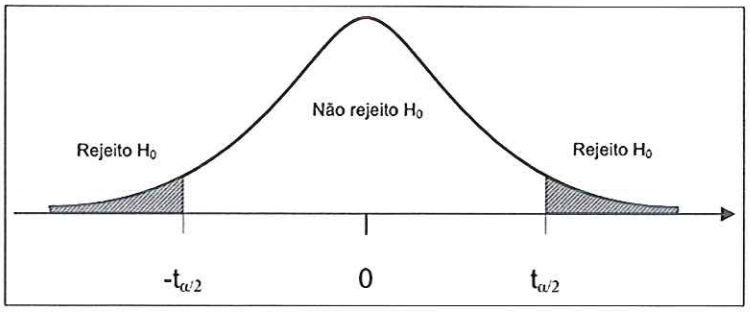

Figura 8.15: Hipótese Alternativa $\mu \neq \mu_0$

A Tabela 8.2 apresenta alguns critérios para o teste de hipótese.

Considerando um teste unilateral dado pelas hipóteses:

Tabela 8.2: Tabela de critérios para o teste de hipótese

Hipótese Alternativa	Rejeita H_0 se	Aceita H_o se
$\mu < \mu_0$	$t < t_\alpha$	$t \geq -t_\alpha$
$\mu > \mu_0$	$t > t_\alpha$	$t \leq -t_\alpha$
$\mu \neq \mu_0$	$t < -t_{\alpha/2}$ ou $t > t_{\alpha/2}$	$-t_{\alpha/2} \leq t \leq t_{\alpha/2}$

$$\begin{cases} H_0 : \ \mu = \mu_0 \\ H_1 : \ \mu < \mu_0 \end{cases}$$

A interpretação dos erros pode ser vista como:

α – (concluir que $\mu \leq \mu_0$ quando na verdade ele não é.)

β – (concluir que $\mu = \mu_0$ quando na verdade ele não é.)

A situação ideal é aquela em que ambas as probabilidades, α e β, são, na verdade, próximas de zero. No entanto, é fácil ver que à medida que se diminui α, β aumenta. A figura 8.16 apresenta esta relação.

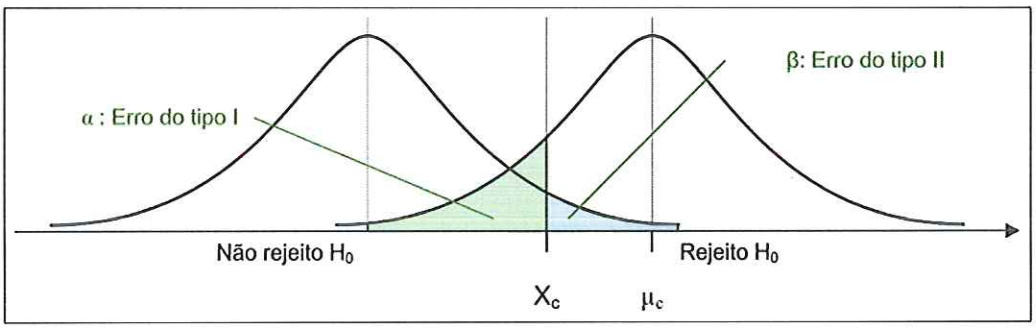

Figura 8.16: Gráfico dos dois erros possíveis

À probabilidade μ é dado o nome de nível de significância do teste. Supondo μ conhecido, tem-se condições de determinar o(s) valor(es)

crítico(s), ou seja, o(s) valores que divide(m) em duas, a região de decisão, denominada região de aceitação e região de rejeitação. A figura 8.17 representa as duas regiões para um valor fixo de α.

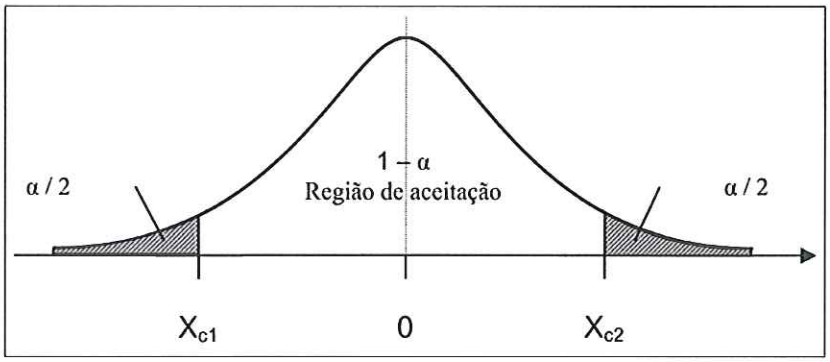

Figura 8.17: Gráfico da Região de aceitação

A expressão para o cálculo do teste quando o desvio padrão populacional (σ) é desconhecido é dada pela expressão 8.1.

$$t = \frac{\overline{X} - \mu_0}{\frac{s}{\sqrt{n}}} \tag{8.1}$$

Para facilitar a execução do teste, são fornecidos os seguintes passos:

1. Estabelecer as hipóteses, por exemplo:

$$\begin{cases} H_0 : \ \mu = \mu_0 \\ H_1 : \ \mu < \mu_0 \end{cases}$$

2. Fixar o nível de siginificância α.

3. Determinar a região crítica; nesse caso deve-se determinar o ponto crítico, como ilustrado na figura 7.6 :

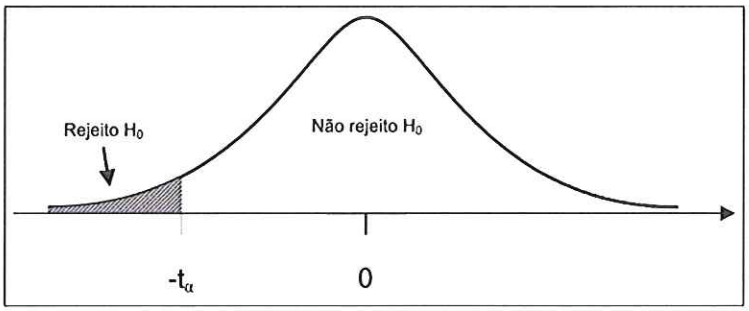

Figura 8.18: Gráfico de decisão do teste de hipótese da estatística t

4. Calcular, sob a hipótese nula, o valor:

$$t_{obs} = \frac{\overline{x} - \mu_0}{\frac{s}{\sqrt{n}}}$$

onde:

- $\overline{x}$: média amostral.
- μ_0: valor da média populacional sob a hipótese nula.
- s : desvio padrão amostral.
- n: tamanho da amostra.

5. Conclusão: Para $t_{obs} < -t_\alpha$ rejeita-se H_0 , caso contrário, aceita-se H_0.