UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

PEDRO HENRIQUE BUORO ALBERTINI

ESTUDO DE FERRAMENTAS DE SOFTWARE PARA MELHORIA DE DESEMPENHO EM AMBIENTE CLUSTER COMPUTACIONAL

FLORIANÓPOLIS, 2008

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

PEDRO HENRIQUE BUORO ALBERTINI

ESTUDO DE FERRAMENTAS DE SOFTWARE PARA MELHORIA DE DESEMPENHO EM AMBIENTE CLUSTER COMPUTACIONAL

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Orientador:
Prof. Dr. Mário Antônio Ribeiro Dantas

Membros da Banca:
Prof. Dr. José Mazzucco Junior
Alex Sandro Roschildt Pinto

FLORIANÓPOLIS, 2008

Pedro Henrique Buoro Albertini

ESTUDO DE FERRAMENTAS DE SOFTWARE PARA MELHORIA DE DESEMPENHO EM AMBIENTE CLUSTER COMPUTACIONAL

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Orientador: _____
Prof. Dr. Mário Dantas

Banca examinadora

_____
Prof. Dr. José Mazzucco Junior

_____
Alex Sandro Roschildt Pinto

À todos que acreditaram que um dia esse trabalho estaria pronto. E aos que não acreditaram também.

Agradecimentos

Agradeço à minha família pelo apoio, meu pai Mario Valter e minha mãe Maria Teodora, por terem me dado condições para chegar até aqui.

À minha namorada Renata, por todo amor e carinho, pelo apoio e incentivo para que esse trabalho ficasse pronto. E também pela compreensão do tempo que não pudemos passar juntos para que esse projeto se tornasse viável.

Ao professor Mário Dantas, orientador desse trabalho, pelo direcionamento e apoio.

Resumo

Os agregados computacionais vem cada vez mais sendo empregados como solução para a computação de alto desempenho. Mais baratos e com maior possibilidade de escalabilidade, os clusters são uma alternativa bem mais viável para aplicações que demandam grande quantidade de memória e processamento. Este trabalho mostra alguns conceitos básicos da computação distribuída de alto desempenho, a implementação de um cluster utilizando equipamentos comuns, sistema operacional Linux e o middleware de cluster Oscar, e as ferramentas para execução e monitoramento de aplicações nesse ambiente.

PALAVRAS-CHAVE: *Agregado Computacional. Oscar. Computação Distribuída. Computação de Alto-Desempenho.*

Abstract

Computer Clusters has been increasingly used as solution for high performance computing. Cheaper and easier to scale, cluster are a more viable alternative to run applications that require great amount of memory and processing. This work show some basic concepts of distributed high performance computing, a implementation of a cluster using ordinary equipments, opensource operational system, the Oscar cluster middleware and tools for execution and monitoring applications in this environment.

KEYWORDS: *Cluster Computing. Oscar Middleware. Distributed Computing. High Performance Computing.*.

Índice de figuras

Lista de Tabelas

Sumário

1 INTRODUÇÃO

Desde o surgimento das primeiras redes que interligavem microcomputadores, no início dos anos setenta, a idéia de rodar aplicaçóes distribuídas em maquinas separadas para otimizar o uso dos ciclos ociosos de cada uma, evitando o disperdício de recursos não utilizados já era visualizada. Nessa época onde ainda não se falava em Internet, a primeira rede Ethernet instalada pelo centro de pesquisas PARC(Palo Alto Research Center) [DANTAS, 2005] tinha como objetivo principal prover uma espinha dorsal para aplicações distribuídas.

Algum tempo depois, já nos anos noventa, alguns resultados interessantes foram obtidos por pesquisas que visavam resolver problemas conhecidos e difíceis de se resolver computacionalmente utilizando-se apenas uma máquina, o que tornou uma realidade o desenvolvimento de aplicações que requeriam o uso de diversas máquinas, como por exemplo o projeto Seti@Home(Search for Extraterrestrial Intelligence)[SETI,2008] que usa computadores de qualquer pessoa com acesso a Internet para analizar dados de satélites e buscar por indícios de vida inteligente fora do planeta.

Estamos vivendo um período de grande avanço tecnológico. Cada dia surgem inúmeros produtos e serviços que vem para facilitar nossa vida, mudando o modo de interação com as pessoas e organizações em geral. E nos deixando cada vez mais acostumados a isso. Porem nos últimos anos a tecnologia de microprocessadores vem esbarrando em limitações físicas, que fazem com que o ritmo de avanço tecnológico nessa área não consiga ser mantido. O que foi previsto por Gordon Moore em 1965, conhecido como a Lei de Moore[WIKIPEDIA-MOORE, 2008], de que a cada 18 meses a capacidade dos microprocessadores dobrariam, mantando os custos constantes funcionou por mais de 30 anos, mas começa a dar sinais de que não dura muito tempo.[SCHALLER, 1997]

Uma limitação na capacidade de processamento, não acompanhando a demanda da tecnologia leva a restrição de vários tipos de software, de Aplicativos domésticos a grandes servidores utilizados por empresas. O processamento distribuído vem como forma de contornar essas limitações. Pois um grupo de máquinas com tecnologia inferiores trabalhando juntas podem equivaler a uma máquina com processamento bem superior. O uso das maquinas em conjuntos vem sendo cada vez mais estudado pelo meio acadêmico e cientifico, visto a eminente quebra da lei de Moore.

1.1 Objetivos

1.1.1 Objetivo Geral

- ✓ Estudo sobre ferramentas de gerencia de ambientes de cluster computacional.

1.1.2 Objetivos Específicos

- ✓ Obter referencial teórico sobre ambientes distribuidos.
- ✓ Configurar um ambiente de cluster computacional.
- ✓ Demonstrar o uso das ferramentas de gerenciamento do cluster.
- ✓ Análise das informações

1.1.3 Metodologia

A pesquisa aqui desenvolvida terá caráter exploratório, que pode ser entendida como o "aprofundamento de conceitos preliminares sobre determinada temática". [RAUPP; BEUREN, 2004]. Envolvendo levantamentos bibliográficos, análise de exemplos. Segundo [GIL, 1999], pesquisa exploratória "[...] é desenvolvida no sentido de proporcionar uma visão geral acerca de determinado fato". Sendo assim, "a pesquisa exploratória visa prover o pesquisador de maior conhecimento sobre o tema ou problema de pesquisa em perspectiva". [MATTAR, 2000].

2 COMPUTAÇÃO DISTRIBUÍDA

2.1 Introdução

O processamento distribuído envolve a utilização descentralizada de hardware, software e diversos outros instrumentos podendo inclusive estar geograficamente separados. Esse tipo de computação vem cada vez mais se tornando realidade para diversos usuários, tanto para uso comercial quanto para pesquisa. Distribuindo e agregando através de complexas infra-estruturas, podemos disponibilizar às aplicações recursos em quantidades e qualidade nunca antes imaginados.

As configurações de alto desempenho podem ser classificada basicamente em dois grupos principais:

- ✓ **Clusters:** "Caracterizado pelo agrupamento físico considerando-se um pequeno limite geográfico, ou virtual, de inúmeros computadores para a execução de aplicações" [DANTAS, 2005].
- ✓ **Grids:** "Infra-estruturas com interoperabilidade entre ambientes computacionais dispersos em uma grande área geográfica". "Os grids computacionais podem ser vistos como cooperativas de clusters de diferentes organizações com objetivo de prover recursos e serviços únicos e diferenciados." [DANTAS, 2005].

Já as aplicações que rodam nesse ambiente se agrupam em duas classes:

- ✓ **Distribuídas:** fazem uso de recursos dispersos, e que não necessariamente se inter-relacionam. São assistidas por softwares (conhecidos por Middleware) que mediam a interação do usuário com a aplicação, escalonando as tarefas para os recursos ociosos que podem atende-las.
- ✓ **Paralelas:** é subdividida em pedaços menores e distribuído a vários processadores, que execultam de forma simultânea os sub-processos. Ao final o resultado de cada pedaço é agrupado de volta para formar o resultado final desejado.

Para o Melhor desempenho de determinada aplicação é necessário classifica-la corretamente como paralela ou distribuída, para que utilize os recursos adequados para sua melhor otimização

2.2 Arquitetura e Redes de Computadores

O projeto de infra-estrutura do ambiente é algo de extrema importância no sucesso da implementação da computação distribuída. A tecnologia de rede deve ser muito bem escolhida, visando alguns fatores importantes, sem os quais o nível do desempenho desejado pode ficar comprometido :

- ✓ Grande largura de Banda
- ✓ Baixo retardo de comunicação
- ✓ Escalabilidade

2.2.1 Ambientes de Clusters

Clusters (ou Agregados) Computacionais são uma forma local para aumento de desempenho na execução de aplicações paralelas e distribuídas. Eles podem ser caracterizados por algumas métricas:

- ✓ Uso dos componentes: Os nodos podem ou não ser dedicados a execução das tarefas submetidas ao cluster.
- ✓ Os componentes podem ser homogêneos ou heterogêneos, tanto a nível de hardware como de Sistema Operacional.
- ✓ Os limites de um agregado podem ser extensões variáveis, o que deve ser levado em conta tanto na execução das aplicações quanto principalmente na implantação do ambiente de rede que serve o cluster.
- ✓ A rede pode ter características única ou hibrida, visando o melhor desempenho e/ou custos.
- ✓ As aplicações a serem executadas no ambiente devem ter seu foco voltado para desempenho, disponibilidade ou ambos.

Atualmente grande clusters tem sido utilizados como forma de ligar computadores PC comuns para um ganho de desempenho, sem que os custos aumentem consideravelmente.

2.2.2 Configurações de Grids

"O termo grid computing foi criado a partir de uma proposta computacional que se baseia na similaridade às malhas de energia Elétrica" [DANTAS, 2005]. Diferente da estrutura Web e dos Serviços Web, caracteriza-se pela agregação de inúmeros ambientes e configurações distintas, que são usados para disponibilizar serviços e recursos sem limite geográfico definido ou conhecido. Os usuários apenas utilizam os serviços providos, sem necessidade de maiores informações detalhadas sobre os provedores dos mesmos.

Pode ser entendido como uma plataforma com interface única para acesso a serviços, esses podendo ser providos massivamente por organizações geograficamente dispersas, que compartilham serviços e informações.

2.3 Fundamentos Básicos

2.3.1 Introdução

Para realização de tarefas mais complexas ou série de tarefas quaisquer, quando se procura um desempenho superior o que pode ser feito é: Trabalhar mais, trabalhar melhor ou dividir a tarefa. Levando isso para a computação, na execução de aplicações as opções seriam utilizar um processador mais rápido, aumentar a eficiência do código, ou utilizar as técnicas de computação distribuída ou paralela.

O uso de computadores com processadores mais rápidos deixa de ser uma opção viável. As limitações que vem sendo encontradas atualmente no desenvolvimento de microprocessadores acabam por dificultar o aumento de desempenho que vinha sendo observado nas ultimas décadas. A lei de Moore que valeu por tanto tempo, onde a cada dois anos a capacidade de processamento dos processadores dobrava, mantendo-se o mesmo custo começa não valer mais [MOORE, 1965].

A princípio o uso de código mais otimizado parece uma alternativa mais razoável, porem na prática as empresas acabam não utilizando, pois existe uma dificuldade muito alta de se otimizar código de aplicações que já estejam em produção.

A opção de utilizar técnicas de computação distribuída ou paralela, agregando mais máquinas para executar as tarefas com um desempenho maior acaba sendo a melhor das opções, atingindo custo mais baixo e um melhor benefício.

2.3.2 Arquiteturas Computacionais

Existe uma grande variedade de arquiteturas computacionais paralelas e distribuídas propostas ao longo dos anos. Vários pesquisadores tentaram encontrar uma maneira de classificá-las, tentando uniformizar de maneira mais coerente as características dos diferentes sistemas computacionais, como: [FLYNN, 1972], [GAJSKI; PIER, 1985] e [TRELEAVEN, 1985)]. Porém o modelo de Flynn é o mais utilizado até hoje, embora alguns especialistas ainda o considerem uma visão muito superficial. Segundo Flynn, os computadores se classificam em:

- ✓ **SISD (Single Instruction Single Data):** Utilizado em máquinas que executam uma sequência de instruções única utilizando uma sequencia única de dados, sendo que para esse tipo de maquina não existe nenhum paralelismo. [TANEMBAUM, 2001]. Como exemplo podemos citar os computadores pessoais comuns, com processador convencional.
- ✓ **SIMD (Single Instruction Multiple Data):** neste tipo de arquitetura também é executado uma única seqüência de instruções, porém, devido à existência de facilidades em hardware para armazenamento (um vetor ou array), a mesma instrução é processada sob diferente itens de dados [DANTAS, 2005]
- ✓ **MISD (Multiple Instruction Single Data):** Arquitetura incomun, geralmente utilizada para tolerância a falhas. Sistemas heterogêneos operam sobre os mesmos itens de dados, e devem concordar no resultado. Um lugar onde são usados esse tipo são nos computadores de bordo de naves espaciais[WIKIPEDIA-FLYNN, 2008].
- ✓ **MIMD (Multiple Instruction Multiple Data):** Nessa arquitetura existem múltiplos processadores, cada um podendo executar independente dos demais. Exemplos são computadores com mais de um processador e configurações distribuídas de Clusters computacionais [DANTAS, 2005]

2.3.3 Multiprocessadores

Um multiprocessador é um sistema computacional que tem vários processadores em um único espaço de endereçamento visível a todos esses processadores. Nele roda apenas uma instancia de sistema operacional, coordenando o trabalho dos processadores. [TANEMBAUM, 2001].

Essa arquitetura é conhecida como fortemente acoplada, devido ao nível de interconexão entre os processadores e a memória por meio de seu sistema local de interconexão. Essa interconexão pode ser feita por configuração de barramento, ou de switch(comutador), mas essa diferença de interconexão não difere sua classificação. O número de processadores que pode ser associado nessa arquitetura pode variar de alguns até centenas. O diferencial dessa arquitetura é que a comunicação entre esses processadores é feita por instruções de acesso comum a memória, pois independente do número de processadores eles sempre compartilham a memória por inteiro. Esse compartilhamento na verdade é o que restringe a escalabilidade dos processadores. Computadores comerciais que se encaixam nessa arquitetura de multiprocessador são conhecidos como servidores de pequena e média capacidade.

Figura 1 - Multicomputador

2.3.4 Multicomputadores

Diferentemente dos multiprocessadores, nos multicomputadores não existe compartilhamento de memória. Toda a comunicação inter-processadores é feita através de troca de mensagens. O modelo de programação a ser adotado nesse abordagem deve ser radicalmente diferente, porém essa arquitetura suporta uma maior escalabilidade [TANEMBAUM, 2001].

Cada elemento, podendo ser formado por um ou mais processadores e sua memória local, é chamado de nó. Vários nós compõem um sistema Multicomputador [DANTAS,

2005]. Assim como nos Multiprocessadores, a conexão entre os nós pode ser feita tanto por uma barra como por um comutador.

2.3.5 Sistemas Distribuídos

São configurações que tem um enorme potencial de escalabilidade, pela agregação dos computadores de redes convencionais. Variações de configurações com máquinas semelhantes e/ou diferentes num mesmo ambiente permitem a formação de configurações interessantes de clusters e grids, deixando ampla gama de possibilidades aberta para que possam ser explorados como melhor modo de execução para distintos tipos de aplicações.

Os sistemas distribuídos são muita vezes empregados como um grande computador, chamado pela literatura de metacomputador (metacomputer) [DANTAS, 2005].

Embora o uso de ambientes distribuídos seja interessante no ponto de vista de utilização de recursos abundantes e ociosos, alguns aspectos devem ter um cuidado redobrado para garantir um bom funcionamento:

- ✓ **Segurança:** Deve-se tomar cuidado, pois aplicativos rodando de forma remota podem ficar expostos a pessoas ou processos não autorizados. Deve-se implementar formas de Autenticação, Criptografia, etc.
- ✓ **Retardo de Comunicação:** Devido a distribuição geográfica distante que componentes do ambiente podem estar, os retardos na comunicação são geralmente grandes. Uma forma de se evitar isso é executando apenas aplicações com granularidade alta, ou seja, que dependam muito mais de processamento do que de entrada e saída de dados
- ✓ **Confiabilidade e Disponibilidade:** Por ser composto de diversos componentes independentes e diferentes essa questão se torna complexa nesse ambiente. A probabilidade de ocorrência de falhas aumenta consideravelmente, comparando-se com um sistema comum.
- ✓ **Compatibilidade:** Diferenças em sistemas operacionais, arquiteturas de processadores, compiladores e interfaces de rede podem não interagir conforme solicitado, gerando inconsistências e mal funcionamento, podendo causar falhas nos outros itens.

2.3.6 Clusters

"As configurações de clusters, em termos de arquiteturas computacionais, podem ser entendidas como uma agregação de computadores de uma forma dedicada (ou não)." [DANTAS, 2005] .

A figura abaixo mostra um ambiente de cluster dedicado, montado com computadores IBM-PC convencionais, conectados entre si por um switch ethernet de 100Mbps.

Figura 2 - Cluster dedicado formado por 64 IBM-PCs comuns

A figura 2 ilustra uma configuração de cluster não dedicado, onde as máquinas além de executarem as atribuídas a elas pelo ambiente de cluster, podem rodar aplicações convencionais. O processamento de cada nó do cluster pode ser utilizado somente ao executar-se aplicações que solicitem um maior desempenho.

Figura 3 - Cluster não dedicado

A utilização de clusters vem para atender um mercado existente relativo a sistemas com alto grau de escalabilidade e de disponibilidade, que deixou de ser suprido ou por não atingir patamares aceitáveis, ou por custos impeditivos por computadores de arquitetura simples. Seu objetivo principal é aumentar facilmente o poder de processamento, a capacidade de memória e tamanho de disco disponível a qualquer momento [PATTERSON; HENNESSY, 2000].

Normalmente os clusters são compostos por agregados de computadores convencionais, onde o custo para se adicionar um nodo a mais para agregar mais potencial é relativamente baixo. Mas nada impede que os nós do cluster sejam compostos por máquinas multiprocessadas, o que agregaria mais valor ainda ao cluster [DANTAS, 2005].

A escalabilidade é a palavra chave dos ambientes de cluster, visto que todo ambiente pode crescer a medida em que novos recursos estiverem disponíveis, de forma incremental, sem impacto nenhum para a infra-estrutura já existente.

2.3.7 Grids

Arquiteturalmente, Grids podem ser entendidas como um compartilhamento de recursos (Processadores, Memórias, Dados, etc.) em níveis globais. Podendo ser imaginado como um ambiente de alto desempenho que possibilita utilizar todos os recursos computacionais disponíveis, onde quer que estejam. São novas formas de se oferecer serviços por meio de agregação de recurso em ambientes geograficamente distribuídos, mas com foco na qualidade de serviços.

Ambientes de grid compartilham a vantagem da escalabilidade com os clusters, ainda em um nível maior, onde a ordem de nós é bem superior. Porém devido a distancia que pode haver entre eles, o retardo na comunicação é algo limitante para os ambientes em Grid. Isso torna essa arquitetura mais eficiente para aplicações de alta granularidade.

2.3.8 Escalonamento

Em ambientes com recursos abundantes disponíveis, faz-se necessário um bom gerenciamento dos mesmos, para que sejam aproveitados da melhor maneira possível. A atribuição das tarefas a cada elemento deve ser feita de forma eficiente, visando diminuir o

tempo global de execução das aplicações. O objetivo a ser alcançado deve ser maximizar a utilização dos recursos computacionais, e diminuir os custos relativos a comunicação, que são mais expendiosos nesses ambientes.

Segundo a classificação de Casavant e Kuhl [CASAVANT; KUHL, 1988 apud DANTAS, 2005] inicialmente os métodos de escalonamento são divididos em local e global. Escalonamento local refere-se ao problema de atribuição das fatias de tempo (time-slices) de um processador aos processadores. É aquele realizado pelo sistema operacional. Já o escalonamento global refere-se ao problema de decidir em qual nodo deve ser executado um processo, sendo, portanto, aplicáveis em sistemas distribuídos.

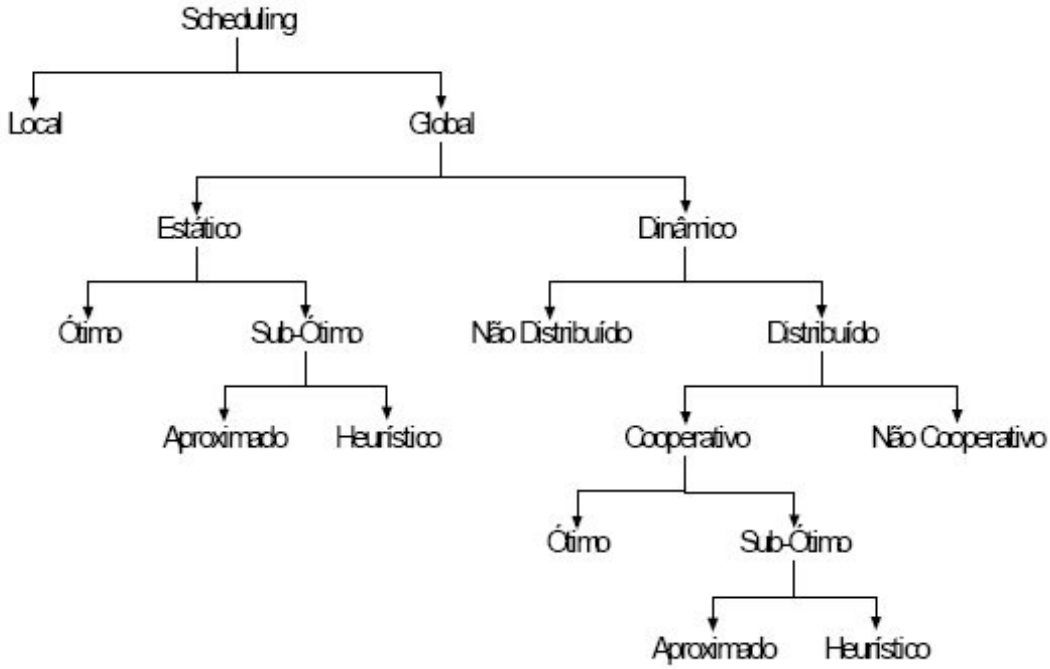


Figura 4 - Classificação dos métodos de escalonamento. Fonte: [CASAVANT; KUHL, 1988 apud DANTAS, 2005]

1 Escalonamento Estático

Nesse tipo de escalonamento a atribuição de processos aos processadores [e realizada antes da execução do programa se iniciar, fazendo com que seja necessário que se tenha informações sobre o tempo de execução dos processos e elementos de processamento (nós) em tempo de compilação. Depois de feito o escalonamento, este não poderá ser alterado em tempo de execução.

O escalonamento pode ser compreendido como uma abordagem de solução que está

incluída na classe de problemas de decisão, onde deseja-se uma resposta do tipo sim ou não. Em geral, para se obter escalonamentos ótimos depara-se com um problema NP-completos, que não possuem algoritmo simples para resolver. Por isso em geral são empregados os métodos subótimos. Esses podem ser classificados em escalonamento aproximado, onde é feita uma busca no espaço da solução em amplitude ou profundidade, ou escalonamento heurístico, onde regras empíricas são usadas para orientar a busca por uma solução quase ótima [DANTAS, 2005].

2 Escalonamento Dinâmico

No escalonamento dinâmico é assumido que pouco se sabe sobre as necessidades do processo ou do ambiente em que ele irá rodar. No escalonamento dinâmico é realizada a redistribuição de processos aos processadores durante a execução do programa. Em geral é adotado o balanceamento de carga entre os nós, visando a melhoria do desempenho da aplicação. Pela dificuldade de se estimar tempo de execução e pelo dinamismo dos recursos, os métodos dinâmicos redistribuem os processos, tirando dos nós mais carregados e encaminhando aos mais ociosos.

Segundo Dantas [DANTAS, 2005], três principais tipos de política são empregadas no escalonamento dinâmico:

- ✓ **Informação:** Define quais dados sobre a carga deve ser utilizado, a freqüência com que é atualizado e quem recebe e trata esses dados.
- ✓ **Transferência:** Define as condições necessárias para que um processo seja transferido de um nodo para outro.
- ✓ **Colocação:** Define para qual nó determinado processo deve ser transferido.

O balanceamento de carga pode ser decidido tanto de forma centralizada quanto de forma distribuída, ou combinação de ambos, onde cada política pode ser decidida de uma forma. No caso do balanceamento distribuído, se os elementos tomam decisões independente, são considerados escalonamentos não cooperativos. No caso do balanceamento cooperativo a decisão é tomada em conjunto com os outros nós, buscando atingir uma meta global [DANTAS, 2005].

2.3.9 Dependabilidade

A utilização de computadores e sistemas na sociedade hoje em dia vem cada vez mais aumentando. Esse aumento nos leva cada vez mais a uma dependência. Alguns exemplos de áreas onde essa utilização mais tem trazido benefícios são: transações bancárias, sistemas integrados de bolsa de valores e mercadorias, desenvolvimento e controle automotivo e aeronáutico, projeto e inspeções de plataformas marítimas, controle de reatores nucleares, sistemas de defesa, controle de tráfego aéreo e sistemas de suporte à vida. Nessas atividades um mau funcionamento pode trazer enormes prejuízos, tanto financeiros quanto inclusive perdas de vidas humanas [DANTAS, 2005].

Essa dependência crescente tem levado pesquisadores e organizações a buscarem cada vez mais por sistemas computacionais mais confiáveis. O termo dependabilidade foi criado pelo segmento de estudo e desenvolvimento de técnicas sobre este aspecto da computação distribuída para expressar a relação de dependência entre os componentes que fazem parte de um sistema computacional.

Segundo Dantas [DANTAS,2005] "dependabilidade é uma propriedade dos sitemas computacionais que define a capacidade dos mesmos de prestar um serviço no qual se pode justificadamente confiar".

O serviço prestado pelo sistema é o seu comportamento, como percebido pelos seus usuários. O usuário de um serviço pode ser outro sistema ou um ser humano, através de uma interface. A função do sistema traduz nos diz para o que ele foi projetado e deve ser descrito na sua especificação. A dependibilidade do sistema envolve um conjunto de conceitos que alguns autores costumam dividir em três grupos: atributos, meios pelos quais será alcançada e ameaças. A figura abaixo ilustra uma taxonomia proposta por [AVIZIENIS, 2001], bastante aceita nessa área.

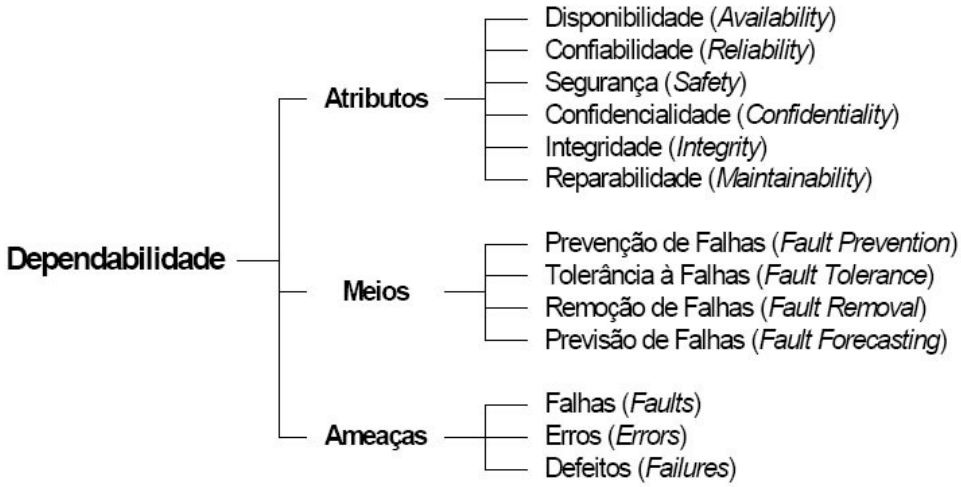


Figura 5 - Taxonomia de dependabilidade. Fonte: [AVIZIENIS, 2001 apud DANTAS, 2005]

3 Ameaças

Existem vários aspectos que podem ameaçar a dependabilidade de um sistema distribuído. Um sistema pode ser considerado defeituoso quando seu serviço desvia-se da especificação do sistema. Segundo Dantas [DANTAS, 2005] "o objetivo da especificação de ameaça a dependabilidade é a identificação mais precisa das falhas, dos erros e, por conseguinte, dos defeitos que podem ocorrer em um sistema distribuído".

Falha é o elemento que ocasiona o erro, que provoca uma transição de estado não planejada do sistema. O sistema pode apresentar uma ou mais falhas e não vir a apresentar erros, onde nesse caso a falha é considerada latente. Já quando a falha leva ao erro ela é considerada uma falha ativa.

Os tipos de falhas e suas origens são bastante variados, podendo ser causadas por falhas elétricas, descargas na rede, ação humana. Na figura a seguir temos um sistema de classificação de falhas proposto no trabalho de [AVIZIENIS, 2001].

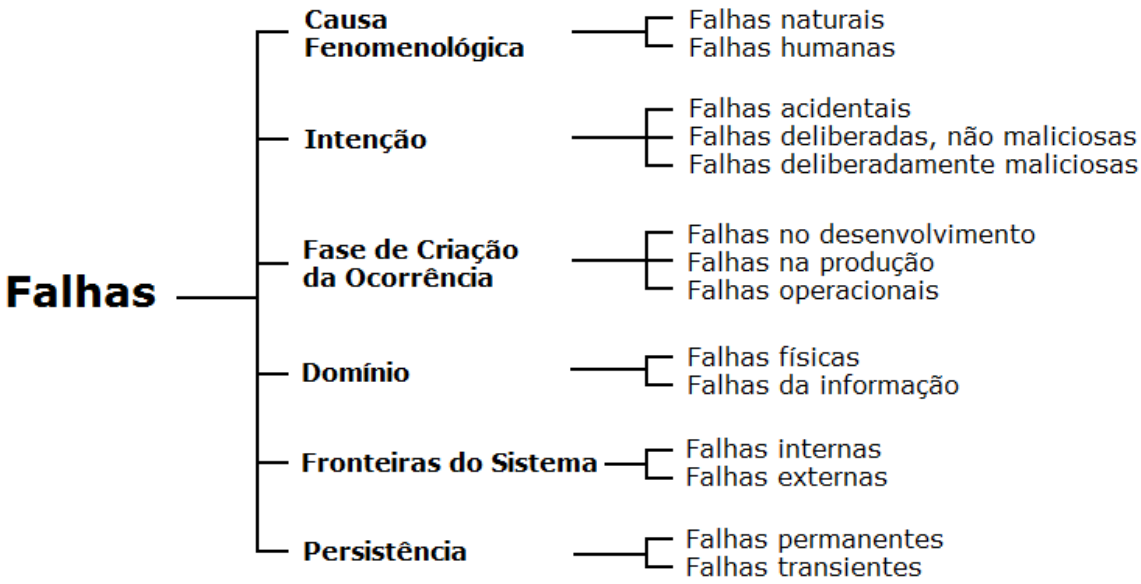


Figura 6 - Diagrama de Falhas. Fonte: [AVIZIENIS, 2001 apud DANTAS, 2005]

Um defeito somente ocorre quando um erro existente no sistema alcança a interface e altera um serviço prestado. O erro pode ou não vir a causar um defeito. O tempo entre a ocorrência do erro e manifestação do defeito ocasionado por ele é chamado pela literatura de latência de erro. Os defeitos apresentados por um sistema também podem ser de diversos tipos, sendo que por isso alguns autores propõem taxonomias para classificá-los, baseados em seu comportamento. Parâmetros comuns para essa classificação são: domínio, percepção ao usuário e conseqüências no ambiente, como pode ser visto na figura abaixo, que representa a classificação proposta por [AVIZIENIS, 2001]:

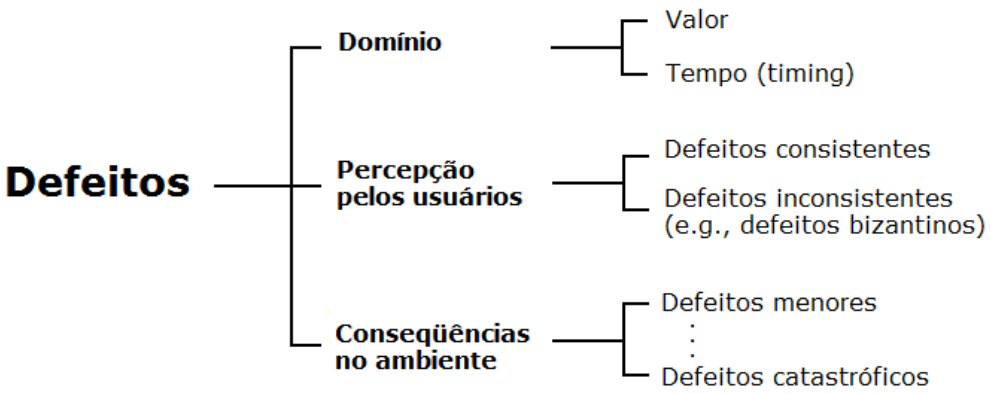


Figura 7 - Diagrama de defeitos. Fonte: [AVIZIENIS, 2001 apud DANTAS, 2005]

4 Meios

Os meios da classificação de dependabilidade nos ajudam a trabalhar na prevenção, tolerância, remoção ou previsão das falhas. Têm como objetivo impedir as falhas, que ao ocorrer podem gerar erros que por fim podem gerar os defeitos.

- ✓ **Prevenção de falhas:** Tem como objetivo aumentar a confiabilidade dos sistemas ainda nas etapas de projeto e desenvolvimento, empregando técnicas de controle de qualidade. Nessa política já partimos do pressuposto que não há como eliminar a totalidade de falhas. São definidos procedimentos manuais de reparo a fim de restaurar o sistema à condição de serviço correto.

  Normalmente somente a prevenção a falhas não é suficiente para a maioria dos sistemas, pois o tempo até o reparo do sistema ou os custos podem ser impeditivos. Para minimizar essas conseqüências pode-se complementar essa técnica com o emprego da tolerância a falhas.

- ✓ **Tolerância a falhas:** "O paradigma de tolerância a falhas pode ser definido como a habilidade que um sistema tem de apresentar um comportamento muito bem definido na ocorrência de falhas ativas." [DANTAS, 2005]

  A primeira forma de tolerância a falhas é o mascaramento, que encobre (oculta) as falhas, para que o sistema como um todo não venha a apresentar defeito. É a forma mais completa e desejada, porém a mais custosa. Alguns autores consideram apenas essa forma como tolerância a falhas, visto que as demais modificam o serviço realizado.

Outra forma é a não tolerância a falhas e é considerada extrema, sendo a menos desejada. Na ocorrência de falhas o sistema apresentará o defeito e ficara não operacional e/ou em um estado inseguro.

Além dessas duas existem outras duas opções intermediárias, uma que garante que o sistema continuará em um estado seguro, mas não garante seu estado operacional, chamado de defeito seguro e outra que garante que o sistema ira permanecer operacional, ainda que esteja em um estado inseguro, que é conhecido como tolerância a falhas sem mascaramento. Em geral a opção pelo defeito seguro é sempre melhor, pois a segurança geralmente é mais importante.

A redundância pode ser usada para contornar falhas, e existem três formas mais utilizadas: estrutura, informação e tempo. Cada uma podendo ser implementada tanto em hardware como em software. A redundância de estrutura é baseada na replicação de partes de componentes de um sistema ou até mesmo do sistema inteiro, podendo ser replicas idênticas ou estruturalmente diferentes, mas que funcionem de forma semelhante para gerar os mesmos resultados, que são submetidos aos mesmos dados de entrada e os resultados são comparados depois. A redundância de informação utilizada nas técnicas de código de paridade como CRC e Checksum por exemplo consistem em adicionar nos dados informações redundantes que permitam detectar e tratar erros. A redundância em tempo consistem na repetição da mesma atividade, para detectar erros provenientes de falhas transientes.

- ✓ **Remoção de falhas:** Pode ser aplicada tanto na fase de desenvolvimento quanto no decorrer da vida operacional do sistema. Durante o desenvolvimento, é aplicada com verificação, diagnóstico e correção. Na verificação (também conhecida como validação) acontecem checagens para conferir se o sistema atende certas propriedades definidas. Caso alguma propriedade não seja atingida, é realizado o diagnóstico e por último é feita a correção com base no diagnóstico**.**

A verificação dos mecanismos de tolerância a falhas é um ponto importante a ser verificado, sendo nesse caso submetidos a testes de injeção de falhas (fault injection) onde são criadas falhas programadas para verificar a eficiência dos métodos de tratamento de falhas.

A remoção de falhas depois que o sistema já está em produção é feita através de atividades de manutenção, que podem ser corretivas, quando detectadas durante a execução do sistema, ou preventivas, quando se tratar de erros latentes que podem vir

a acontecer futuramente.

- ✓ **Previsão de falhas:** é o último meio utilizado para se alcançar a dependabilidade. Consiste na avaliação do sistema em relação a ocorrência e tratamento de falhas, em termos quantitativos, medindo com métodos probabilísticos o atendimento aos atributos da dependabilidade, e qualitativos, visando classificar e ordenar por relevância as causas de defeitos nos sistemas. Tais avaliações têm como meta o levantamento de informações que possam validar a abordagem feita na estrutura de dependabilidade do sistema, e, sendo verificado inconsistências, subsidiar modificações que venham a melhorar o sistema [DANTAS, 2005].

5 Atributos

Para os atributos da dependabilidade usam-se medidas probabilísticas para ponderá-los relativamente, visto que sua natureza é não determinística. Dependendo da aplicação os pesos considerados para cada atributo devem ser diferenciados.

- ✓ **Disponibilidade:** é definido como o tempo em um período em que o sistema encontrava-se funcionando normalmente. O fator importante é a fração de tempo na qual um sistema deverá ter condições de apresentar o serviço correto. Normalmente são usados métricas de tempo médio para ocorrência de defeito e tempo médio para reparo para o levantamento da disponibilidade.
- ✓ **Confiabilidade:** pode ser traduzido como a probabilidade de o sistema não apresentar defeito durante o intervalo de tempo considerado. Na tolerância a falhas a redundância age diretamente nesse atributo de confiabilidade.
- ✓ **Segurança:** é considerado sobre dois aspectos, contra catástrofes e convencional. No primeiro é a probabilidade do sistema não apresentar defeito que leve a conseqüências catastróficas contra os usuários ou meio ambiente em um intervalo de tempo. Na convencional, é a probabilidade de que não ocorra acesso ou mudança indevida no estado do sistema.
- ✓ **Confidencialidade:** é probabilidade de não ocorrer divulgação não autorizada dos dados do sistema.
- ✓ **Integridade:** é a probabilidade de não ocorrerem alterações impróprias de estado no sistema.

- ✓ **Reparabilidade:** avalia o quanto um sistema pode ser restaurado, retornando ao estado correto, supondo que partiu do estado com defeito. É a capacidade de passar por reparos e modificações

3 AMBIENTES DE SOFTWARE

Nesse capítulo serão descritos alguns ambientes de sofware são necessárias na execução de aplicações distribuídas e paralelas integradas baseadas em ambientes de grids e clusters. Como não existe uma taxonomia própria para essa classificação, os ambientes serão categorizados segundo a classificação utilizada por Dantas [DANTAS, 2005], que separa os ambientes em ambiente de programação, ferramentas e middlewares.

3.1 Ambientes de Programação

A configuração computacional nos ambientes paralelos e distribuídos é bem mais complexo do que nas configurações centralizadas, tornando desafiadores os esforços dos ambientes de programação, principalmente por causa da possibilidade de fazer uso de recursos que não estão disponíveis localmente, requerendo acesso e autenticação para ser utilizados.

Diversos modelos de programação devem serconsiderados para as diferentes arquiteturas dos sistemas, em relação a possíveis direren;cãs de largura de banda, possíveis retardos de comunicação, diferenças de hardware, sistemas operacionais e linguagens e diferentes políticas de segurança. Serão mostrados explorados nesse capitulo três modelos que são considerados por Dantas [DANTAS, 2005] os mais expressivos, Web Services para ambientes distribuídos e Parallel Virtual Machine (PVM) e Message Passing Interface (MPI) para ambientes paralelos.

3.1.1 Web Services

"Serviços Web são aplicações de serviços que permitem seu acesso por intermédio do uso de protocolos web padrão" [DANTAS, 2005] Esse modelo impulsionou a utilização de serviços distribuídos, pela simplicidade e facilidade inerente. Permitem assegurar a interoperabiliade de aplicações rodando sobre ambientes com características diferentes em

relação a rede, plataformas de hardware e software.

O acesso aos serviços é feito utilizando-se de padrões amplamente utilizados e aceitos pela comunidade, o que garante a integração de diversos ambientes distintos, que comumente implementam esses padrões. Esses protocolos, como HTTP e XML são padrões abertos e aceitos na comnidade de TI. Os padrões empregados no desenvolvimento desses serviços são propostos por entidades sem fins lucrativos compostos por membros de diversos segmentos da indústria e acadêmicos da área de software.

Existe uma padronização da interface e do protocolo de comunicação que pode ser acionado pelos clientes dos sérvios. A utilização da linguagem WSDL (Web Service Definition Language) adiciona um nível de abstração entre a interface e a implementação, tornando possível uma maior flexibilidade, em conseqüência do fraco aclopamento entre as aplicações.

As principais especificações que sáo utilizadas nos Web Services são o XML, SOAP, WSDL e UDDI. Estas serão explicadas logo abaixo, devido sua importância para a o modelo.

- ✓ **XML** (eXtensible Markup Language)**:** é uma especificação que visa simplificar o formato de documentos. Define que um documento siga regras determinadas para facilitar a compreensão das pessoas. São utilizados pelos Serviços Web para comunicação, descrição de interfaces e para definir suas mensagens. Protocolos como SOAP, UDDI e WSDL empregal as interfaces e mensagens XML que qualquer outra aplicação pode interpretar. O XML permite que desenvolvedores possam criar seus próprios rótulos, sem ter que seguir padrões de nomenclatura para as tags. Esta liberdade torna possível a definição, transmissão, validação e interpretação de dados entre aplicações distintas.
- ✓ **SOAP:** é um protocolo leve baseado no XML que permite a troca de informações entre aplicações através de HTTP. Pode ser compreendido como um padrão que contém informações sobre o conteúdo da mensagem que transporta, quando esta vai de um serviço produtor para serviços consumidores. Por não ser uma arquitetura distribuída é necessário que o protocolo SOAP faça parte de uma implementação maior. Além de descrever o conteúdo das mensagens e como processa-las o protocolo oferece facilidades de transporte para troca de mensagens, além de possuir um conjunto de regras para expressar instâncias de tipos de dados de aplicações predefinidas e uma representação de chamadas a procedimentos remotos (RPC).

  Esse protocolo por fazer uso de chamadas não orientadas ao estado da conexão têm um grande potencial de crescimento, que se traduz em uma grande escalabilidade.

Por ser um protocolo de muito alto nível pode sofrer alguma degradação no seu desempenho, mas a interoperabilidade do protocolo em ambientes heterogêneos é um fator diferencial.

- ✓ **WSDL:** utilizado para descrever serviços de rede como um conjunto de operações finais em mensagens contendo uma informação de um documento ou procedimento. Essas mensagens são descritas de forma abstrata, e depois são integradas a um protocolo de rede real onde um formato da mensagem possui uma definição de seus pontos de acesso. Por sua natureza abstrata é que esse formato é de interesse para descrição de Serviços Web.
- ✓ **UDDI:** formado por um conjunto de protocolos para representar um sérvio de orientação, provendo uma listagem com os sérvios disponíveis em uma rede. Tem como ponto principal fornecer um ponto único de referência dos sérvios disponíveis em um ambiente. Permite que mesmos serviços rodem em versões distintas e que sinônimos para os serviços possam ser criados.

3.1.2 Parallel Virtual Machine (PVM)

O objetivo principal do PVM é disponibilizar facilidades de um ambiente de programação paralela em conjunto comum mecanismo transparente para o desenvolvedor de agregação de inúmeras máquinas de diversas arquiteturas diferentes.

Pode ser dividido em duas partes. A biblioteca PVM, que é utilizado em conjunto de outras linguagens de programação para integração a arquitetura servindo de biblioteca de troca de mensagem. E o ambiente paralelo computacional (Parallel Programming Environment), que é na verdade um conjunto de componentes que são disponibiliza para a biblioteca PVM as facilidades de gerenciamento e interligação de conexões com máquinas remotas, criando uma abstração para o usuários das aplicações.

Segundo Dantas [DANTAS, 2005]: “O PVM é um pacote de software que permite a execução de sistemas operacionais tais como Unix e/ou Windows em um conjunto de computadores heterogêneos possibilitando esses serem agregados de uma rede e utilizados como um único e grande computador paralelo”

É considerado um pacote com grande portabilidade, podendo rodar desde um notebook com configurações reduziadas até supercomputadores. Devido a simplicidade de suas chamadas e por sua interface gráfica para acompanhamento da execução é uma

ferramenta propícia para o ensino de programação paralela.

3.1.3 Message Passing Interface (MPI)

O padrão denominado MPI foi um esforço conjunto de empresas da área de produtos de alto desempenho, universidades e centros de pesquisa que utilizam amplamente aplicações distribuídas e paralelas. Semelhante a outras iniciativas de padrão de facto, um exemplo clássico é o TCP/IP, o MPI é disponibilizado gratuitamente através de instituições de pesquisas ou comercializado por empresas privadas.

Essa padronização é referente apenas ao nível de troca de mensagens, deixando opção para que seja executado sob qualquer ambiente computacional paralelo, inclusive sobre um ambiente que utiliza o ambiente paralelo do PVM.

Existem algumas ferramentas existentes que utilizam MPI, citadas por [DANTAS, 2005]:

- ✓ **PyPar:** ambiente amigável para o programador e eficiente para utilizar o padrão MPI dentro da linguagem Python.
- ✓ **SKAMPI:** é um pacote de software de benchmark. Pode ser utilizado para auxiliar a otimização das aplicações, facilitando o reconhecimento de gargalos e identificação de processos ineficientes. Faz medidas de desempenho da grande maioria das operações existentes no padrão MPI.
- ✓ **ETNUS:** é um software depurador que ajuda no desenvolvimento de aplicações que utilizam implementações do padrão MPI.
- ✓ **XMPI:** é uma ferramenta disponível na implementação LAM do MPI que auxilia a visualização e depuração de problemas em programas paralelos implementados em MPI.

3.2 Ferramentas

Segundo Dantas, [DANTAS 2005] antes do uso de qualquer ferramenta para fins de configurações distribuídas devem ser questionados o que são essas ferramentas e para que fins específicos a classe de ambiente dessa ferramenta é utilizada. Podemos considerar que se enquadram como ferramentas de software aquelas que ajudam na instalação de aplicativos distribuídos, no gerenciamento dos recursos, escalonamento de tarefas, balanceamento de carga e monitoração e visualização da execução dos processos. Elas fazem parte de um

escopo maior, o middleware. Elas são agrupadas e orquestradas com objetivo de esconder a complexidade do ambiente distribuído, seja qual for a arquitetura (cluster ou grid), hardware e sistema operacional.

3.2.1 Gerenciadores de Tarefas e Recursos

Ferramentas conhecidas como RMS (do inglês Resource Management and Systems) são ferramentas utilizadas para a gerëncia das terefas e recursos geograficamente distribuídos nas configurações dos ambientes distribuídos. É amplamente utilizado como ferramentas nos pacotes de middleware.

Nas configurações de multicomputadores todos os nós têm seus próprios sistemas operacionais, e dentro deles o seu próprio gerenciador local de tarefas e recursos, que trabalha indiferente a configuração distribuída. As ferramentas de gerenciamento de tarefa são necessárias para suprir a fata nativa de cooperação dos escalonadores.

Outro fator importante para a utilização desses ssistemas gerenciadores é a possibilidade de fornecer sérvios diferenciados e transparentes para o usuário. Eles auxiliam na distribuição e execução das tarefas, sem agregar complexididade alguma. O usuário interage com o gerenciador de tarefas ao executar uma aplicação, e o mesmo cuida para escolher o melhor caminho para execução da aplicação no ambiente distribuído, seguindo a política de escalonamento global aplicada ao ambiente.

Os ambientes RMS são projetados para auxiliar os programadores de aplicações a fazer o melhor aproveitamento das configurações distribuídas. Podem ser utilizados tanto pacotes comerciais, de licença paga, como pacotes gratuitos, que serviram de base para os comerciais. Normalmente são caracterizados por disponibilizar: [DANTAS, 2005]

- ✓ Suporte a sistemas operacionais distintos.
- ✓ Ambientes de batch, paralelo e interativo.
- ✓ Verificação e migração de processo.
- ✓ Balanceamento de carga.
- ✓ Limite de execução de tarefas simultâneas.
- ✓ Interface gráfica (GUI).

Esses serviços apresentadas são úteis para sistemas que rodam em ambientes onde a carga de cada nó muda constantemente, principalmente em sistemas de nós não dedicados. Também se fazem valiosos em sistemas que necessitam de várias operações de entrada e saída

de dados, podendo ter grande diferença no tempo de execução das aplicações, por mudar sempre a tarefa para uma máquina que atenda melhor os requisitos da aplicação.

O RMS geralmente tem sua arquitetura composta por elementos de interface gráfica, política de administração do ambiente, filas de execução, computadores pertencentes ao ambiente, tarefas, recursos disponíveis, políticas de distribuição e técnicas de escalonamento e balanceamento de carga. Na tabela a seguir [DANTAS, 2005] é apresentado um sumário dos componentes dessa arquitetura.

Tabela 1 - Elementos da arquitetura RMS

Componente	Descrição Funcional
GUI	Interface através da qual o usuário solicita que suas tarefas sejam submetidas, monitoradas, excluídas ou colocadas em estado suspenso temporário.
Administração	Módulo que contém as características dos computadores pertencentes à configuração, acesso permitido para os usuários e suas tarefas, limitações de recursos para tarefas e usuários, controle de tempo de uso de recursos e funcionamento de filas.
Filas	O conceito de filas é empregado nos sistemas RMS, visando organizar a forma de execução de tarefas submetidas à configuração distribuída e prover uma abordagem eficiente de controle durante a execução das tarefas.
Computadores	São os elementos aonde serão executadas as tarefas, submetidas ao RMS. Como esses podem ter configurações heterogêneas em Termos de hardware e software, o módulo de Administração auxilia na filtragem dos sérviços e recursos solicitados a cada computador.
Tarefas	Representam a unidade de solicitação de serviços e recursos distribuídos que são submetidos pelos pacotes RMS às configurações distribuídas.
Recursos	Processadores, quantidade de memória, sistemas de armazenamento e outros tipos especiais de dispositivos que podem ser oferecidos para que tarefas de usuários remotos possam utilizar.
Políticas	Geralmente as políticas que são consideradas pelos RMS são a de utilização e escalonamento. Quanto a política de utilização considera-se o módulo de Administração como uma base para sua implementação. Por outro lado, a política de escalonamento pode ser baseada na forma estática ou dinâmica.
Escalonamento – Balanceamento de Carga	O escalonamento pode ser baseado em algoritmos tais como: o primeiro a chegar será o primeiro a ser servido; selecionar o ambiente com menor carga; utilizar uma seleção fixa baseada em algum conhecimento de administração do ambiente ou empregar um algoritmo híbrido considerando uma mistura dos algoritmos anteriores. Quanto ao balanceamento de carga, podemos utilizar formas que considerem: informações de pesos previamente estabelecidos ou séries históricas de carga do ambiente.

3.3 Middleware

O ambiente de Middleware como definido por Dantas:

"Ambiente que provê para os usuários dos clusters e grids computacionais uma transparência segura de acesso, facilidade de submissão das aplicações, gerenciamento de recursos e serviços da configuração distribuída geograficamente. Os ambientes de middlewares também têm como função o controle e permissão de instalação de pacotes de software nos nós da configuração, provendo um mecanismo específico de tratamento de falha de funcionamento do ambiente, quando do mau funcionamento de um determinado componente de hardware ou software." [DANTAS, 2005]

Ou seja, é um pacote de software que auxilia desenvolvedores a abstraírem as características específicas do ambiente. Com essa abordagem a preocução da execução do aplicativo em ambientes diferentes não existe, pois o middleware gerencia essas diferenças.

3.3.1 Sistemas de Imagem Única

Nas configurações distribuídas, principalmente nos clusters que se utilizam de computadores não dedicados, todas as máquinas possuem recursos e serviços que muitas vezes só ficam disponíveis para os usuários locais. O ideal é que os recursos e serviços possam ser utilizados não apenas pelos usuários locais, mas também por outros usuários da mesma rede. Para atingir o objetivo de se ter transparência de onde o sérvio é executado, é preciso abstrair algumas funções do sistema operacional em uma camada superior e abstrata para o usuário.

O compartilhamento pode significar um melhor aproveitamento da configuração, melhorando a produtividade da efetiva agregação distribuída dos recursos e serviços. Os sistemas de imagens única podem ser considerados middlewares que provêem uma grande abstração para os usuários dos clusters em relação a disposição física e a estrutura dos sistemas do cluster. São projetados para serem utilizados em ambientes com várias máquinas possivelmente heterogêneas onde os usuários tenham uma visão unificada do compartilhamento de recursos e serviços do ambiente. A Figura 8 mostra um esquema de funcionamento desse sistema.

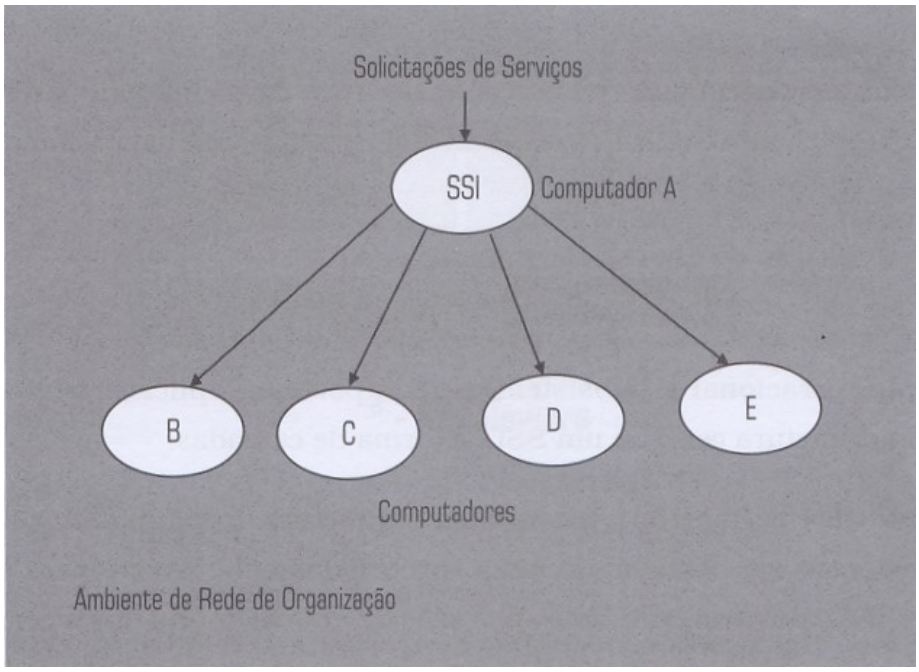


Figura 8 – Esquema de sistema de imagem única [DANTAS, 2005]

Dantas [DANTAS, 2005] define como diferenciais nos sistemas de imagem única os seguintes aspectos:

- ✓ Transparência na execução de aplicações em ambientes geograficamente distribuídos.
- ✓ Não requerem o conhecimento de localização de recursos.
- ✓ Reduzem a necessidade de gerenciamento do sistema.
- ✓ Têm embutidos sistemas de alta disponibilidade.
- ✓ Aumentam a robustez da configuração, reduzindo a intervenção de operadores
- ✓ Estabelecem um sistema de comunicação de troca de mensagem para controle do ambiente, melhorando o balanceamento de carga.
- ✓ Aceitam a utilização de diversas ferramentas RMS.

Essa arquitetura pode ser composta por níveis de hardware, sistema operacional e subsistemas de suporte às aplicações. O nível de hardware permite que o sistema de imagem única seja configurado baseado na arquitetura do computador, tornando possível, por exemplo, criar uma abstração de memória única compartilhada.

No nível de Sistema operacional são desejáveis a oferta de um pacote de software que gerencia o sistema e apresente um sistema de arquivo único e que disponibilize a abordagem de memória compartilhada distribuída para os usuários de uma configuração de multicomputadores. Na Figura 9, Dantas [DANTAS, 2005] mostra uma máquina abstrata que

faz parte de um cluster, rodando um middleware de imagem única

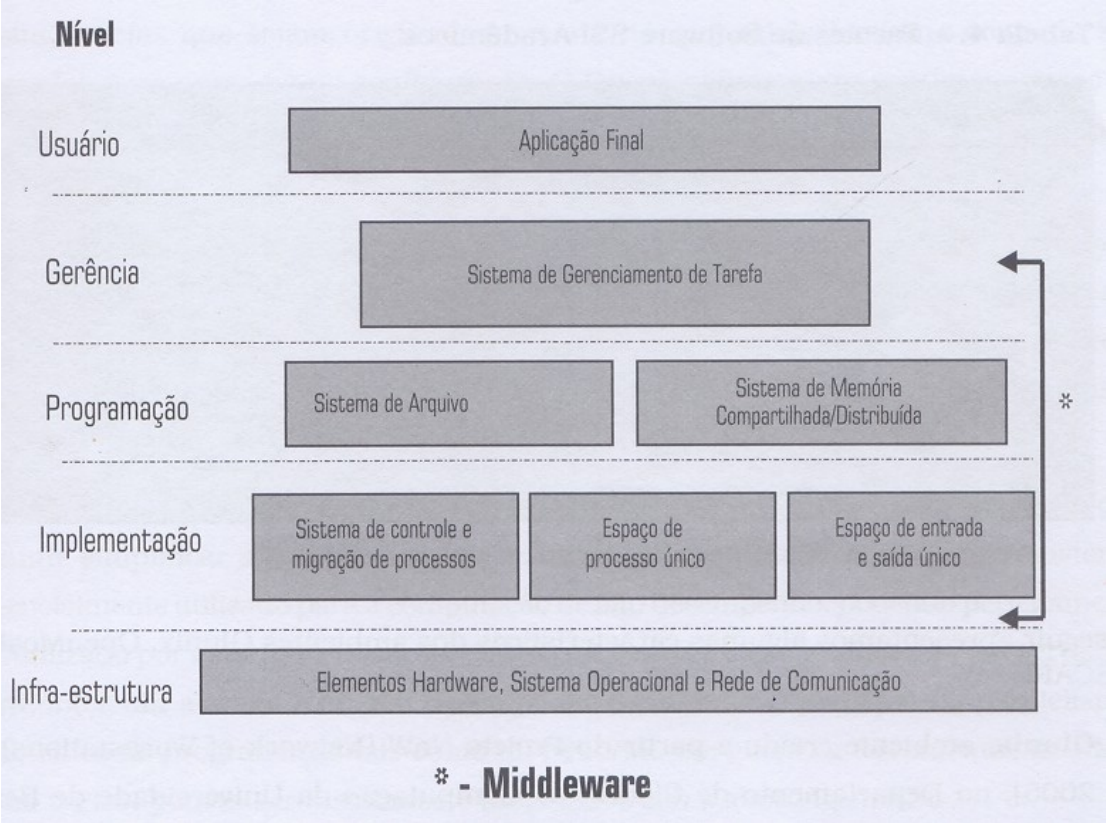


Figura 9 – Abstração de um nó de um sistema de imagem única [DANTAS, 2005]

Nos próximos Capítulos serão feitos comentários sobre os sistemas de imagem única disponíveis, tanto opções comerciais quanto acadêmicos.

6 Glunix

Ambiente criado a partir do Projeto NoW (Network of Workstations) [NOW, 2008] visa a criação de um nível de abstração acima do nível do sistema operacional, com objetivo de prover uma maior portabilidade e reduzir o tempo de execuç!ao de desenvolvimento de aplicações.

O grupo de pesquisa do projeto NoW idealizou o GLunix, pois tinham como alvo agregar recursos em conjunto de entidades que pudessem prover maior desempenho para aplicações seqüenciais e paralelas. O Glunix é uma abordagem de SSI que diminui a limitação dos diferentes sistemas operacionais executando nas diferentes máquinas de um cluster.

O Glunix difere por seu suporte agendamento de recursos para integrar aplicações

seqüenciais e parlelas, parametrização automática de recursos, mecanismos para enfileirar extensões do sistema de imagem única acima dos sistemas operacionais existentes, além do próprio sistema de imagem única, tolerância a falhas e gerência do sistema. [GLUNIX, 2008]

7 OpenMosix

Desenvolvido pela Hebrew University, é uma extensão do projeto Mosix, o OpenMosix é uma extensão do kernel do Linux para ambientes de imagem única de cluster. Segundo os pesquisadores do projeto ele faz com que uma rede de pcs padrão executando o pacote torne-se um supercomputador para aplicações executando sob o Linux. Uma vez instalado o software as máquinas da configuração trocam mensagens visando gerenciar entre si e fazer com que os processos executados se distribuam igualmente para todos os micros do cluster.

Desse modo as tarefas iniciadas em um nó que esta ocupado são executadas em algum outro que tiver recursos disponíveis. O OpenMosix se encarrega de fazer de tempos em tempos uma verificação e redistribuição dos processos de acordo com a carga em que cada nó se encontra. No que se refere a escalabilidade, um novo nó pode ser adicionado ao cluster e dinamicanente assim que for feita a ligação já poderá receber tarefas de outros nós.

O grande diferencial do OpenMosix é que as aplicações não precisam ser modificadas, já que ele opera sobre o Kernel do Linux, qualquer aplicação executada em Linux pode rodar dentro do ambiente do Openmosix.

8 Oscar

É um pacote de software que permite simplificar a complexa tarefa de utilização e gerenciamento de um cluster com as finalidades de instalação e configuração, manutenção, programação paralela, sistema de filas e escalonador. É essencialmente utilizado para computação de alto desempenho, podendo ser usado por qualquer aplicação que precise das funcionalidades de um agregado, para obter um aumento em seu desempenho através do paralelismo.

Existem vários pacotes no Oscar para ajudar o usuário a monitorar processos paralelos, compartilhar recursos do sistema, o que auxilia a melhora da performance do sistema. O pacote de software conta também com bibliotecas e ferramentas como MPI e PVM. Na figura abaixo esta representado um ambiente de cluster configurado utilizando-se o OSCAR.

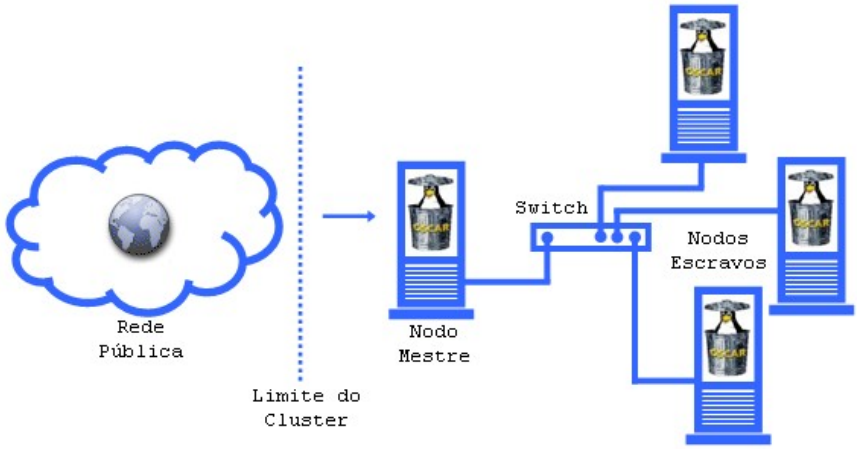


Figura 10 - Esquema de cluster utilizando OSCAR. [DENISE, 2007]

9 Solaris MC

Considerado um sistema operacional para computadores clusterizados, tem como objetivo prover transparência para configurações onde milhares de computadores são agregados para prover escalabilidade, facilidade de administração e alta disponibilidade. O Solaris MC é considerado uma extensão do sistema operacional Solaris que executa as aplicações de forma distribuída, dando idéia de um único super-computador.

Aplicações de gerenciamento de banco de dados online e suporte a decis'~ao são exemplos típicos de aplicações que podem ganhar com a escalabilidade e alta disponibilidade dessa abordagem de imagem única.

"O solaris provê uma imagem única global de todos os processadores e usuários da configuração que podem ser gerenciados dse um único ponto, assim facilitando a administração e gerência dos clusters." [SUN, 2005 apud DANTAS, 2005]

4 AMBIENTE DE CLUSTER

O processamento de alto desempenho é um paradigma da computação que tem como principal objetivo a execução de milhares aplicações simultaneamente e também o processamento de tarefas complexas paralelamente com significativa eficiência. Ao longo do

tempo vinha sendo empregados computadores paralelos específicos para resolução desses problemas, tais como SMPs e MPPs.

Com o custo elevado desse tipo de equipamento, atualmente a maneira de buscar esse processamento de alto nível mudou. O uso de ambientes de agregados compostos por máquinas comuns, em grande quantidade está cada vez mais sendo utilizado. A grande oferta de máquinas tipo IBM-PC e o custo delas viabiliza o agrupamento de grandes quantidades de máquinas, com custo relativamente baixo.

Essas configurações, conhecidas como cluster computacionais, tem como principal objetivo a agregação de recursos de diversas máquinas e disponibilização dos mesmos para melhorar o desempenho das aplicações.

Segundo Dantas [DANTAS, 2005] é interessante classificarmos os ambientes de cluster segundo algumas de suas características, tais como seu limite geográfico, modo de utilização dos nós, tipos de hardware e conexão, requerimento de recursos pelas aplicações e tipos de nós da configuração. Esta classificação é melhor representada pela figura abaixo:

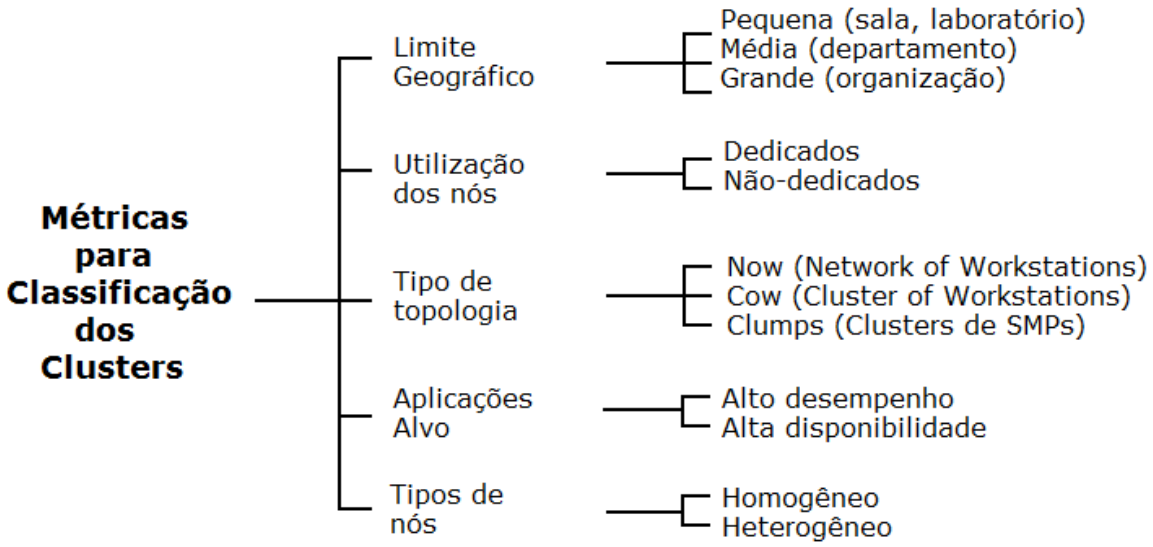


Figura 11 - Métricas para classificação de clusters [DANTAS, 2005]

4.1 Limite Geográfico

O uso de clusters vem crescendo de um tempo para cá, e sua utilização vem ganhando bastante força nas organizações. Normalmente as configurações de cluster iniciam-se em salas específicas e laboratórios, ambientes fisicamente próximos. São utilizados para isso desde computadores simples interligados por conexões de rede padrão através de hubs,

switches até dispositivos especiais de interligação. Após essa primeira configuração de cluster é normal que dentro da organização o ambiente cresça a nível departamental.

Nesse próximo nível os usuários da configuração já têm uma boa noção da importância da configuração e de que devem ser utilizadas políticas de uso para garantir o alto desempenho. Após o sucesso de um ambiente menor, o ambiente de cluster tende a crescer cada vez mais, com agregação de recursos mais espalhados dentro da organização.

Esse tipo de comportamento nos mostra que o cluster é um ambiente que independe do seu ambiente físico, e que boas políticas devem ser definidas para sua utilização [DANTAS, 2005].

4.2 Utilização dos Nós

É essencial que seja estabelecida a forma de participação dos nós no ambiente de cluster, de forma dedicada ou não dedicada. Essa definição influencia diretamente diversos outros fatores, tais como políticas de gerenciamento, segurança, alta-disponibilidade, escalonamento de processos, balanceamento de carga e tipo de middleware do sistema de imagem única.

4.2.1 Configurações não dedicadas

Nas configurações não dedicadas a principio não existe um pacote de software central que faça o controle lógico do ambiente. Os computadore estão ligados através da rede, porém cada computador é utilizado por usuários locais para execução de suas aplicações. Os recursos excedentes podem ser disponibilizados para auxiliar a execução de outros aplicativos do ambiente.

Em um ambiente não dedicado as diferenças de hardware, sistemas operacionais e principalmente carga local de cada um dos nós deve ser levada em conta. É necessário adotar um sistema de imagem única de cluster que adote políticas para trabalhar essas diferenças. Essa configuração caracterizada pela interligação virtual (não dedicada) é normalmente denomidado em termos de topologia como NoW (Network of Workstations). Os sistemas de imagem única que rodam nesse ambiente devem alem de considerar as diferenças sitadas, considerar o retardo na comunicação e a prioridade do usuário local sob processos remotos que estejam sendo executados em cada nó.

Essa configuração é interessante devido a seu custo-desempenho, pois é possível aproveitar recursos ociosos já disponíveis. Mas também se os nós estiverem sobrecarregados

com aplicações locais o desempenho das aplicações rodando distribuídas pode ser prejudicado. A falta de uma política única e bem definida no uso dos nós da configuração faz com que o middleware tenha que se esforçar em descobrir e ponderar paremtros como tipo dos processadores, espaço livre de memória, capacidade de armazenamento em disco e carga dos componentes do cluster.

4.2.2 Configurações dedicadas

"As configurações de cluster dedicados são as mais apropriadas para execução de aplicações críticas de uma organização." [DANTAS, 2005] . Essa configuração propicia um melhor ambiente para execução desses tipos de aplicação, graças ao controle do ambiente. Através de políticas de uso, o middleware dessa configuração pode:

- ✓ Distribuir melhor as aplicações para os processadores disponívies, baseado no conhecimento prévio do funcionamento da configuração e da prioridade dos aplicativos.
- ✓ Melhor controle de alta disponibilidade, com a identificação de nós espelhos.
- ✓ Facilita o crescimento do ambiente de cluster, com a configuração mais rápida de um novo nodo a ser adicionado.
- ✓ O Pacote de imagem única pode servir como elemento centralizador para efetuar cópias de segurança de processos submetidos para processadores da configuração e ainda cópias de backups.

Ambientes dedicados normalmente podem ser caracterizados por computadores sem componentes de interação, como mouse, teclado e monitores. O Controle e acesso a cada nó e feito por terminais, ligados a rede.

Nessas configurações pode ser adicionado um compenente adicional, que serve de interface para o ambiente e o usuário, onde ficam centralizados o controle de acesso e gerência do ambiente.

4.3 Tipos de Hardware

Pensando em Hardware existem dois aspectos que podem ser analisados nos ambientes de cluster computacional. Os tipos de dispositivos de interconexão entre os compoentes e a própria arquitetura de cada um dos nós do cluster.

Quanto ao dispositivo de interconexão quando o cluster é formado por multicomputadores ou multiprocessadores, o sistema de interconexão é algo que não pode ser mudado. Já em ambientes de CoWs o projetista deve analisar a melhor maneira de interconexão, visando o custo e necessidade de intercomunicação entre os nós para a execução das aplicações alvos. Dependendo do tipo de aplicação que irá rodar no cluster a velocidade de interconexão pode ser fator diferencial para a melhora no desempenho.

Quanto a arquitetura dos componentes existe hoje uma ampla gama de tipos diferentes de computadores que podem ser utilizados nesses ambientes. O mais adequado a ser decidido em um projeto de ambiente é uma análise dos aplicativos e o custo-benefício do ambiente projetado. Existem projetos onde a uniáão de computadores do tipo PC já é mais do que suficiente para suprir a demanda de desempenho necessário. Existem também equipamentos especializados, com arquiteturas preparadas exclusivamente para o trabalho em cluster, que podem ser utilizados quando o uso de computadores padrão não for suficiente.

4.4 Aplicações Alvo

Existem duas principais classes de aplicações que necessitam de ambiente de cluster para serem executadas. A Classe de aplicações que necessita de um grande poder de processamento, grande quantidade de memória e armazenamento em disco para ganhar em termos de desempenho. E a classe de aplicações que não podem parar sua execução, que são as que necessitam de alta disponibilidade.

- ✓ **Alta disponibilidade:** Caracterizados por sistemas críticos, onde o mal funcionamento do sistema pode representar um prejuízo muito grande em diversos aspectos. As aplicações tem que sempre estar disponíveis para acesso dos clientes. Requerem redundância no ambiente do cluster, para garantir a operabilidade constante.
- ✓ **Alto desempenho:** Caracterizados por aplicações que necessitam de um grande volume e rapidez de processamento, ou manipulação de enormes quantidades de informação, onde pode haver distribuição de tarefas.

4.5 Tipos de Nós

Uma maneira de se classificar os clusters computacionais é em relação a semelhança dos equipamentos constituem os nós desse ambiente, tanto em questões de hardware quanto

de software. Dessa semelhança ou diferença classificamos os ambientes em homogêneos ou heterogêneos, respectivamente.

Nos clusters homogêneos são utilizados computadores com arquiteturas equivalentes em todos os nós, como por exemplo PCs ou SMPs. É interessante manter essa similaridade entre os nós pois primeiro os esfor;os de interoperabilidade entre pacotes de software é inexistente, e também a reposição de componentes é mais fácil.

Os clusteres heterogênos, tanto em quesitos de software quanto hardware normalmente são utilizados em ambientes que cresceram de forma modular. Requerem um esforço maior em seu projeto para manter a transparência do ambiente em relação a essas diferenças, e para garantir que as diferenças arquiteturais não reflitam em erros durante a execução das aplicações.

4.6 Ambientes de Software para Cluster

A configuração de software do ambiente de cluster computacional é um fator fundamental para seu desempenho. Não basta apenas juntar os recursos se não existir uma política preestabelecida de uso dessa configuração. O Ambiente de software de cluster deve incluir não somente um sistema operacional, deve incluir um sistema de imagem única principalmente.

O sistema de imagem única pode ser estabelecido no nível de middeware, sistema operacional e hardware. A utilização de middeware como SSI traz maior flexibilidade de configuração, devido principalmente a independência de sistema operacional, a heterogeneidade e dedicação ou não dos componentes.

Diversos pacotes de software têm sido desenvolvidos focando a taxa de utilização ou paralelismo dos processos. Através da taxa de utilizaç!ao é buscado uma melhor distribuição de múltiplas tarefas independentes. Em ambientes paralelos a busca é pela melhor eficiência na execução de tarefas paralelas dependentes, para que o tempo final de execução seja o melhor. Esses ambientes podem ser representados pelo diagrama abaixo.

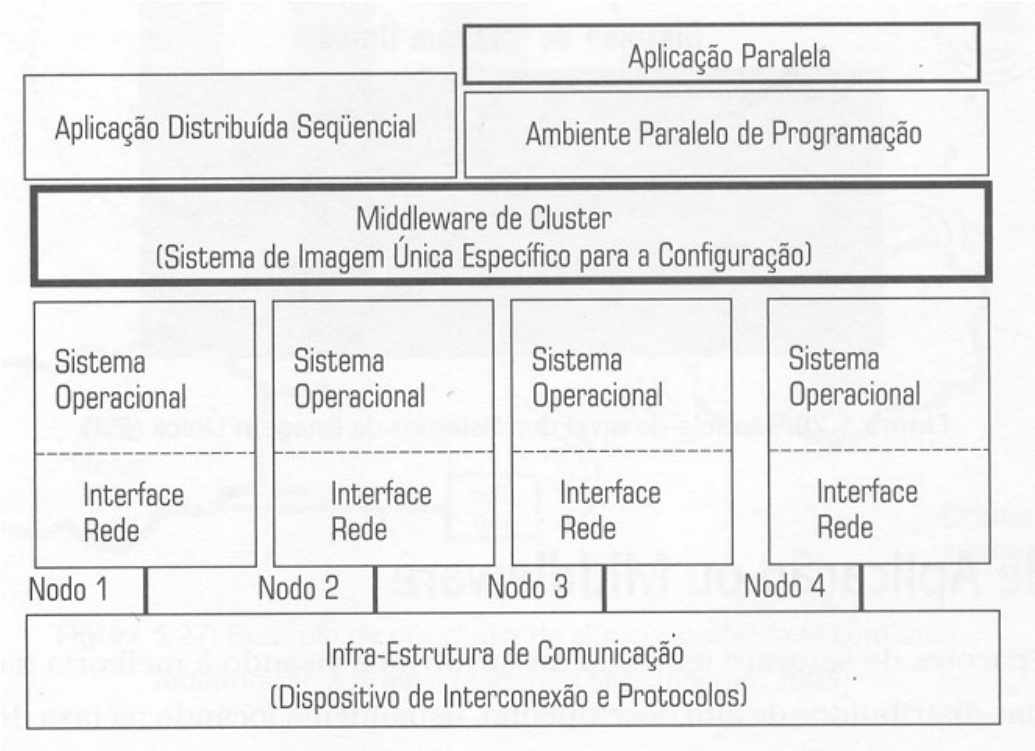


Figura 12 - Ambiente de Software para Clusters fonte: [DANTAS, 2005]

A nível de sistema operacional, todos os nós do ambiente de cluster deverão possuir a mesma versão de sistema operacional, esse deve possuir suporte para que o mecanismo de comunicação entre processos seja realmente funcional. A vantagem desse modo é a obtenção de um maior desempenho, visto que a solicitação das tarefas é feita diretamente ao kernel do sistema operacional.

5 METODOLOGIA

5.1 Ambiente Experimental

O Ambiente de cluster montado para esse experimento foi bem caseiro, utilizando apenas computadores comuns, os quais estamos acostumados a conviver no nosso dia a dia. Foram utilizados para compor o cluster um notebook HP, modelo tx1410, utilizado como servidor, e um PC padrão, ambos dispondo de 2Gb de memória, mais do que o suficiente para o que é desejado se demonstrar nesse trabalho. A rede utilizada é a extensão de uma rede residencial já existente.

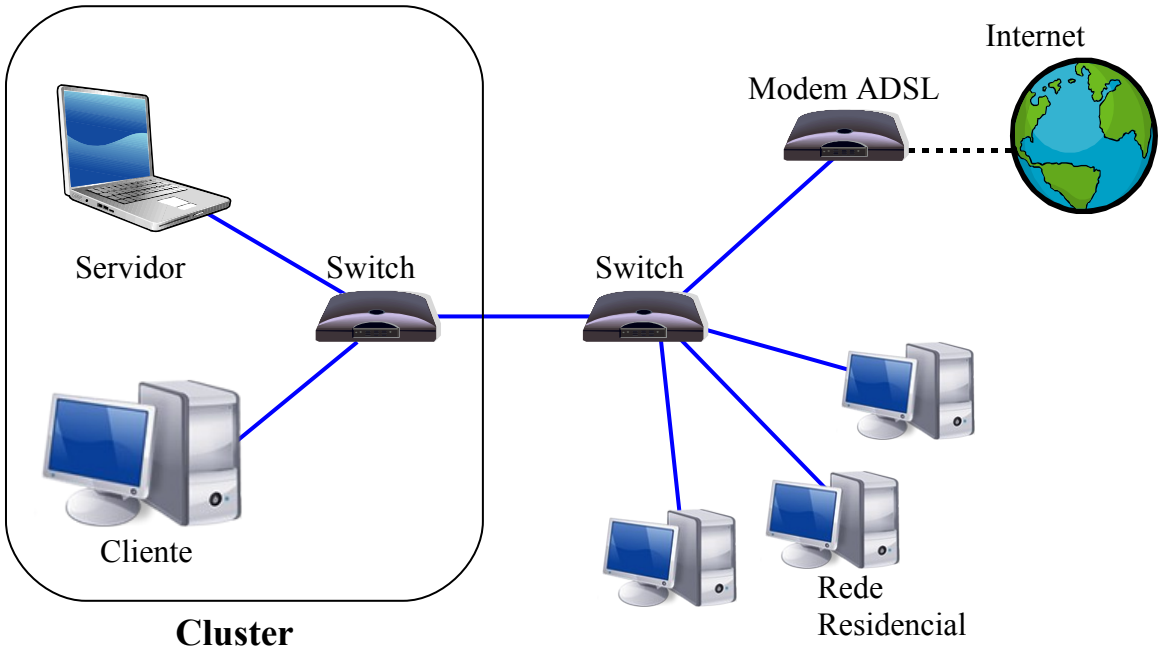


Figura 13 – Ambiente do Cluster Experimental

5.2 Configuração de Hardware

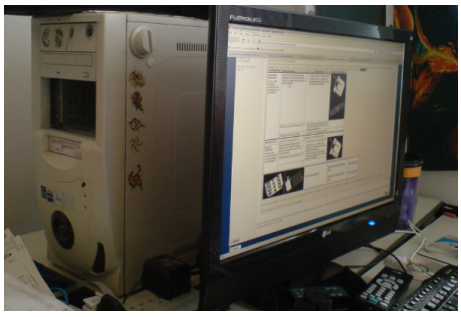
Servidor – pedrohnote.localdomain

Notebook HP tx1410

Processador AMD TX2 64bits 2200Mhz

2Gb RAM DDR 667Mhz

IP 10.1.1.60



Cliente - ocarnode1.localdomain

Processador AMD Athlon64x2 4800+

2Gb RAM DDR 800 128bit Dual-Channel

IP 10.1.1.61

O notebook foi escolhido para ser utilizado como servidor devido a sua mobilidade, pois a princípio a intenção era que ele poderia ser levado para outros ambientes com outras máquinas, e essas se conectariam a ele para fazer parte do cluster. Visto que as configurações do ambiente de cluster dependem basicamente da configuração do servidor, e que a adição de um novo cliente é mais simples e rápida.

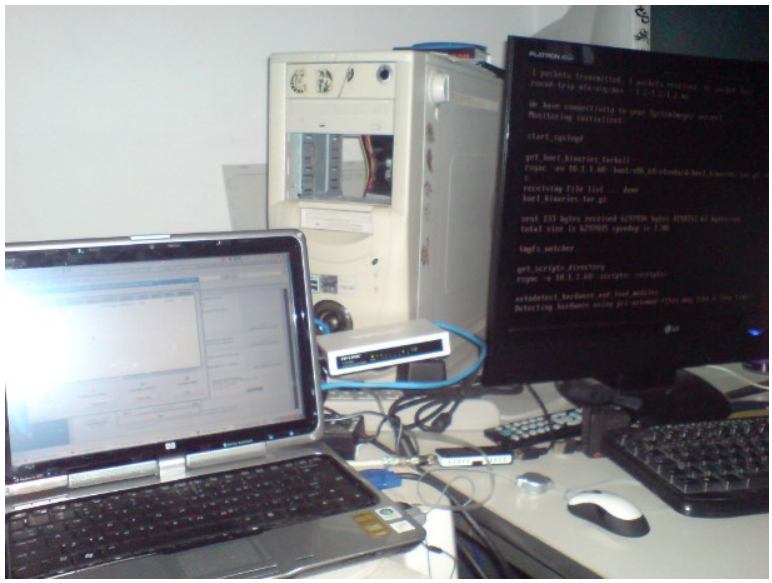


Figura 14 - Cluster

5.3 Infraestrutura de Rede

A infraestrutura de rede sobre a qual foi montada a rede consiste apenas em um Switch Ethernet 10/100, interligando as máquinas do cluster diretamente, e conectando-as com a internet através de outro switch, que compartilha a conexão de um modem ADSL. Os cabos utilzados para conectar as máquinas ao switch, switch a switch e switch a modem ADSL foram cabos padrão ethernet de par trançado, com conector RJ45.

A necessidade de acesso a internet foi verificada durante a fase de instalação e preparação do cluster, sendo que durante a execução das aplicações mesmo essa conexão não é necessária. Foi usado o equipamenteo que se encontrava disponível, de baixo custo, encontrado hoje em dia em qualquer residência.

Figura 15 – Switch do Cluster, Switch da Rede Residencial e Modem Adsl

5.4 Escolha do Sistema Operacional e Middleware

O projeto OSCAR visa suportar as mais diversas distribuições do Linux. Quando novas codificações são feitas elas são ajustadas e testadas em todas as distribuições suportadas anteriormente. Junto com as distribuições das versões do Oscar é distribuído um conjunto de scripts que ajuda a compilar as bibliotecas de pacotes do Oscar em qualquer distribuição do Linux. [OSCAR, 2008].

A versão do Oscar escolhida foi a última disponível em julho de 2008, que é a versão 5.1b. Apesar de ser uma versão beta, foi escolhida por suportar distribuições mais recentes do Linux. Já que o hardware tanto do Notebook quanto do PC que serão utilizados são mais recentes, e poderia haver incompatibilidades com distribuições mais antigas.

Porem existem para algumas distribuições, os pacotes RPMs já compilados. E para essas distribuições o pacote foi testado. Essas distribuições que possuem esses pacotes podem ser vistas na tabela abaixo:

Tabela 2 - Lista de Distribuições suportadas/testadas para o Oscar 5.1b. Fonte: [OSCAR, 2008]

Fedora Core 7
Fedora Core 8
RHEL 4
RHEL 5
YellowDogLinux 5.0
openSUSE 10.2

Por facilidade de instalação, para não ser necessário recompilar os pacotes do Oscar, foi escolhido então para ser instalado no ambiente a distribuição Fedora Core 8.

5.5 Instalação do Linux no Servidor

A distribuição do Linux Fedora Core 8, escolhida como sistema operacional base para

rodar no ambiente de cluster pode ser encontrada para download no endereço eletrônico http://fedoraproject.org/. O Fedora foi instalado em uma partição root de 4gb, foi criado uma partição de 1 gb para swap, e uma partição de 20gb para dados.

A instalação foi feita a partir de uma imagem de DVD para a arquitetura x86_64. Foram encontrados dois problemas durante a instalação no notebook, que foram facilmente solucionadas utilizando instruções obtidas na ferramenta de busca Google.

O primeiro problema aconteceu durante o wizard de configuração na primeira execução depois da instalação do sistema. Logo após selecionar as configurações de localização, data e hora, o sistema congelava ao clicar no próximo. O problema foi resolvido adicionando o comando noapic no Grub, que carrega o sistema operacional.

O segundo problema ocorrido foi um conflito com a placa de vídeo, que fazia com que o cursor do mouse não aparecesse na tela. Esse problema foi resolvido editando-se o arquivo xorg.conf, trocando a sessão Device com a configuração abaixo:

```
...
Section"Device"  Identifier "Videocard0"  Driver "nvidia"  VendorName "Videocard
vendor"  BoardName "nVidia Corporation C51 PCI Express Bridge"  Option
"HWCursor" "off"  EndSection
...
```

A instalação foi tranqüila, mesmo com esses pequenos problemas encontrados, que foram facilmente solucionados.

5.6 Instalação do Oscar no Servidor

O primeiro passo para a configuração do cluster é a instalação do Oscar no notebook que será usado como servidor. Seguindo as instruções disponíveis em http://svn.oscar.openclustergroup.org/trac/oscar.

5.6.1 Preparando para instalar o Oscar

No repositório do Oscar existem os pacotes básicos da versão, e os pacotes compilados para versões especificas. Os arquivos necessários para a instalação no Fedora Core 8 são:

- ✓ oscar-base-5.1.tar.gz
- ✓ oscar-repo-common-rpms-5.1.tar.gz
- ✓ oscar-repo-fc-8-x86_64-5.1.tar.gz

As informações de instalação encontram-se no próprio site do Oscar e também em um arquivo pdf que está dentro do arquivo Oscar-base-5.1.tar.gz. Após baixar os pacotes e descompacta-los é necessário criar a estrutura de diretórios como indicado abaixo, onde serão armazenados os arquivos necessários para a instalação.

/tftpboot/oscar	Repositório dos pacotes(RPMs) básicos do Oscar. (oscar-base e common-rpms)
/tftpboot/distro	Repositório dos pacotes(RPMs) da distribuição do Linux sobre a qual vai ser instalado o Oscar

Dentro do diretorio /distro foi criado um link para o diretório /tftpboot/distro/fc8-x86_64, que apontava para o diretório /usr/pedroh/oscar/rpm, onde foram armazenados os arquivos RPM encontrados no DVD de instalação do fedora. Esses RPMs são necessários para a instalação dos pacotes básicos do Oscar.

5.6.2 Instalação do pacote básico do Oscar

Após criar os diretórios deve-se instalar o pacote básico do Oscar, através do comando:

```
yum install createrepo /tftpboot/oscar/common-rpms/yume*.rpm
yume –repo /tftpboot/oscar/common-rpms install oscar-base
```

O primeiro comando executado em principio não foi executado com sucesso. Foi apresentado a mensagem “Package yume-2.7-2.noarch.rpm is not signed”. Esse problema foi corrigido editando-se o arquivo /etc/yum.conf e alterando o parâmetro "gpgcheck=0". Essa configuração vem como padrão habilitada no fedora.

5.6.3 Configurações de rede do servidor

A maioria das distribuições do Linux usam como padrão “localhost” como nome de rede. Esse é o caso do fedora. Para a instalação do Oscar é necessário mudar esse nome. Para o notebook foi escolhido o nome pedrohnote, que já era o nome da máquina no Windows. O IP utilizado pelos nodos deve estar dentro da faixa de IPs privados, definidos pelo RFC1918 [OSCAR, 2008], que vai de 10.0.0.0 a 10.255.255.255, 172.16.0.0 a 172.31.255.255 e 192.168.0.0 a 192.168.255.255, sendo que o IP de inicio dessas faixas não

deve ser escolhido como o IP do servidor.

Seguindo as orientações, o IP escolhido para o servidor foi 10.1.1.60. O arquivo /etc/hosts foi editado como mostrado a seguir:

```
127.0.0.1 localhost.localdomain localhost
10.1.1.60 pedrohnote.localdomain pedrohnote
```

5.6.4 Wizard de instalação do Oscar

O Oscar vem com um script para ser executado após a configuração de rede no servidor, que prepara todo o ambiente e inicia uma interface visual (GUI) que facilita a configuração do ambiente de cluster disponibilizando alguns passos de instalação. Para iniciar o script, deve-se executar o comando abaixo, no diretório onde foram extraídos os pacotes básicos. Deve ser passado como parâmetro a interface de placa de rede a ser usada.

```
cd /opt/oscar
./install_cluster eth0
```

O script de instalação é responsavel por:

- ✓ Instalar no servidor os pacotes que prerequisitos
- ✓ Instalar os arquivos RPMs do Oscar
- ✓ Atualizar o arquivo `/etc/hosts` com os aliases do Oscar
- ✓ Atualizar o arquivo `/etc/exports`
- ✓ Adiciona os caminhos (paths) do OSCAR ao arquivo `/etc/profile`
- ✓ Atualiza os scripts de inicialização do sistema operacional (`/etc/rc.d/init.d`)
- ✓ Reinicia os serviços alterados

Durante a execução do script de inicalização foram encontrados alguns problemas. O script faz uso dos RPMs que foram copiados para o diretório /tftpboot/distro. Alguns RPMs tinham como pré-requisito outros pacotes que não estavam incluídos nos RPMs da imagem do Fedora Core 8, então alguns pacotes não puderam ser instalados, o que fez o script de instalação ser abortado. Os pacotes que acusaram erro foram libaio, libaio-devel, csh (/bin/csh) e rrdtool.

Como foram poucos pacotes, foram isntalados individualmente através do comando

```
yum install libaio
yum install libaio-devel
yum install csh
yum install rrdtool
```

Após resolver esse problema das dependências, o script executou com sucesso, abrindo a tela do Wizard, para configuração do cluster.

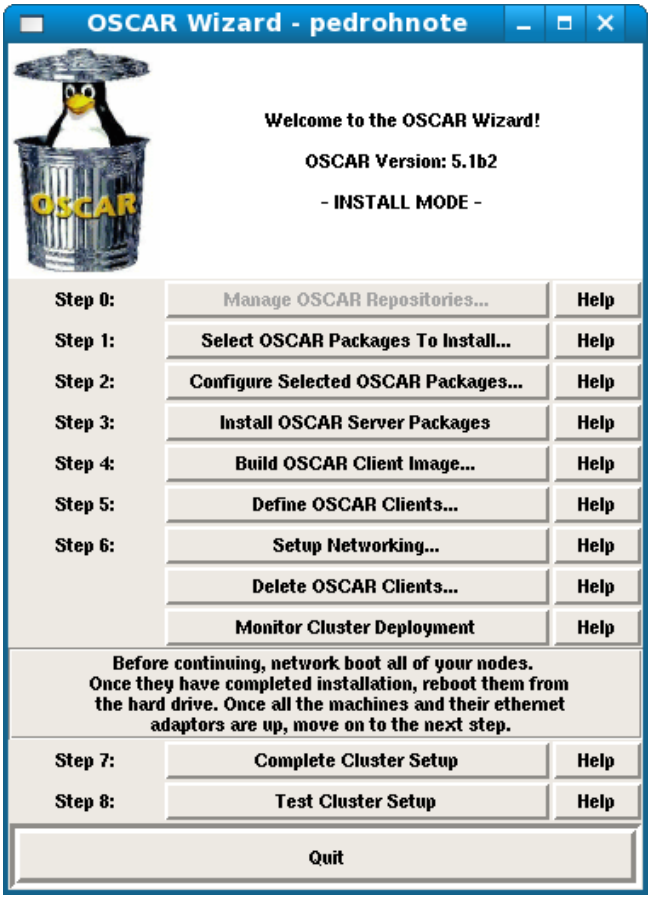


Figura 16 – Wizard de configuração do Oscar

Passo 1: "Select OSCAR Packages to Install"

Devem ser escolhidos os pacotes a ser instalados. Para o ambiente experimental não foi alterado nada da configuração padrão nesse passo.

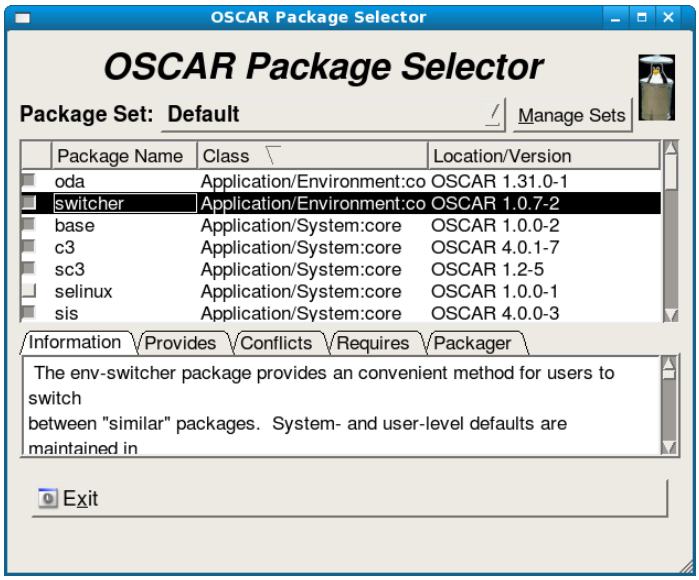


Figura 17 – Oscar Wizard - Escolha de Pacotes

Passo 2: "Configure Selected OSCAR Packages"

Permite a configuração de alguns parâmetros dos pacotes selecionados para ser instalado no passo anterior. Também não foram feitas alterações nesse passo.

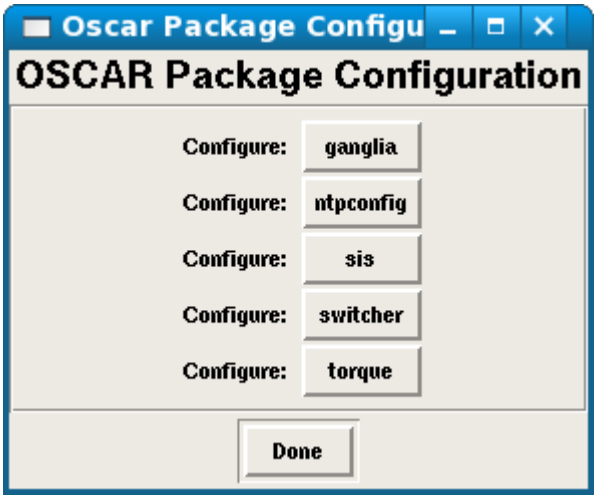


Figura 18 – Oscar Wizard – Configuração de pacotes

Passo 3: "Install OSCAR Server Packages"

Efetua a instalação dos pacotes selecionados e configurados nos passos anteriores. Ao executar esse passo, o log apontou diversas mensagens de falha, por conta de dependências de pacotes não encontrados nos repositórios copiados em /tftpboot/.

Para resolver esse problema, foi adicionado e instalado na mão as bibliotecas que o script acusou que estavam faltando, utilizando o diretório /tftpboot/distro, /tftboot/Oscar e endereço na internet do mirror da distruição do Fedora, apontando para o repositório Everything, que contem todos os pacotes disponíveis da distribuição. Foi rodado o comando

abaixo:

```
yume install --repo /tftpboot/oscar/common-rpms/  --repo /tftpboot/oscar/fc-8-x86_64/  --repo /tftpboot/distro/fedora-8-x86_64/ --repo http://fedora.c3sl.ufpr.br/linux/releases/8/Everything/x86_64/os/ gcc gcc-c++  gcc-gfortran glib libstdc++-devel ltrace screen strace xorg-x11-xdm opkg-apitest-client opkg-base-client  opkg-c3-client opkg-disable-services-client opkg-ganglia-client opkg-lam-client opkg-loghost-client opkg-maui-client opkg-mpich-client opkg-mtaconfig-client  opkg-netbootmgr-client opkg-ntpconfig-client opkg-oda-client opkg-openmpi-client opkg-opium-client opkg-pfilter-client opkg-pvm-client opkg-rapt-client opkg-sc3-client  opkg-sis-client opkg-switcher-client  opkg-sync-files-client  opkg-torque-client  opkg-yume-client
```

Após a instalação das dependências, o passo 3 executou sem exceções, instalando os pacotes do Servidor com sucesso.

Passo 4: "Build OSCAR Client Image"

Nesse passo é gerado a Imagem de instalação dos Clientes. Essa imagem é usada posteriormente pelo programa SystemImager para instalação e configuração automática dos nodos clientes do cluster.

Antes de executa-lo é preciso certificar-se de algumas condições:

- ✓ A configuração de SSH deve permitir login como root. O arquivo /etc/ssh/sshd config no servidor deve estar com a propriedade PermitRootLogin setada para "yes". Essa propriedade pode ser reconfigurada após a geraçao da imagem.
- ✓ As configurações de rede devem estar muito restritivas.
- ✓ Firewalls e a ferramenta SELinux devem estar desabilitados

Após serem feitas essas verificações, a primeira tentativa de gerar a imagem do cliente não foi bem sucedida. No log foram mostradas mensagens de erro parecidas com as mensagens do passo 3, de problemas com dependências de pacotes.

O problema foi resolvido criando um arquivo /tftpboot/distro/fedora-8-x86_64.url com o seguinte conteúdo:

```
file:/tftpboot/distro/fedora-8-x86_64
http://fedora.c3sl.ufpr.br/linux/releases/8/Everything/x86_64
```

Com isso durante a geração de imagem os pacotes que não estavam disponíveis locais foram achados no repositório remoto, e a imagem foi criada com sucesso.

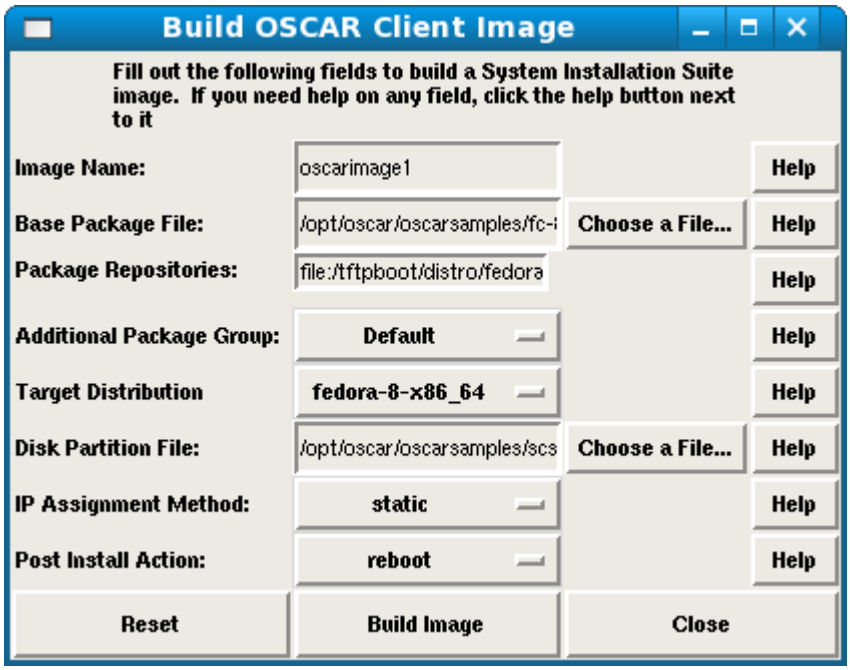


Figura 19 – Oscar Wizard – Imagem dos clientes

Passo 5: "Define OSCAR Clients"

Aqui devem ser configurados as informações dos clientes que farão parte do cluster. Devem ser selecionada qual imagem o cliente usará na sua instalação , IP do cliente e nome do domínio. No ambiente de cluster configurado, será configurado apenas um cliente, com IP 10.1.1.61

Figura 20 – Oscar Wizard – Definição dos clientes

Passo 6: "Setup Networking"

Os clientes devem ser mapeados, com o endereço físico da placa de rede (MAC

Adress). Isso para a instalação do systemimager funcionar corretamente. São escolhidos nesse passo a forma como a imagem será enviada para os clientes (rsync, multicast ou bittorrent) e é configurado o ambiente de Boot pela rede, ou a criação de cd de instalação do SystemImager.

Segundo o manual, para o Fedora foi necessário marcar a opção UYOK (Use Your Own Kernel), para que o kernel do systemimager q fosse instalado pudesse reconhecer os HDs IDEs. Seguindo esse conselho, foi habilitado e configurado o Ambiente de boot remoto.

Para a obtenção do MAC Adress do cliente, foi só reiniciar a maquina cliente, e entrar no modo de boot na rede, enquanto o cliente busca o servidor de PXE, nesse momento deve-se acionar o botão "Start Collecting MACs", então o MAC do cliente aparece na lista para ser atribuído a lista de configurações.

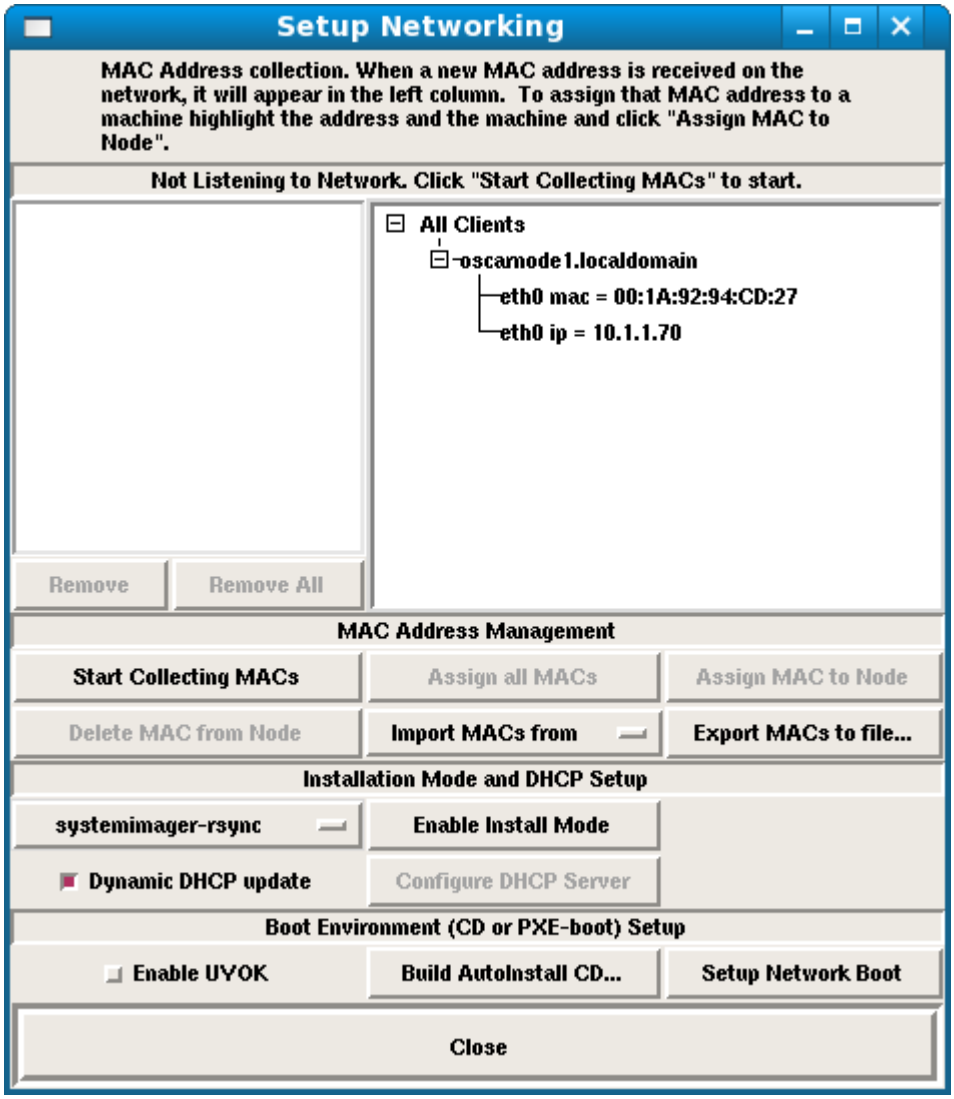


Figura 21 - Oscar Wizard - Configuraçao de rede

5.7 Configuração do Cliente

Para a configuração do cliente existem duas opções. Pode ser feito a configuração de boot remoto, ou caso a placa mãe da máquina cliente não disponibilize essa opção pode ser gravado um cd de instalação. A partir de ambos os métodos, a maquina cliente será inicializada, e começara a instalação a partir da imagem que será baixada da maquina servidor. Nenhuma configuração adicional precisou ser feita, após o termino da instalação no cliente, ele já estava pronto para ser utilizado como um dos nodos do cluster.

No caso do ambiente utilizado, ambos as maquinas tinham a mesma arquitetura, e optou-se por utilizar a mesma versão do Linux em ambas. Mas é possível montar um ambiente de cluster heterogêneo. Para isso basta copiar os pacotes de outras versões para o diretório /tftpboot/distro , e gerar a imagem equivalente a distriuição e arquitetura necessárias.

Após a instalação e reinicialização do cliente, deve ser executados os próximos passos do Wizard no servidor.

Passo 7: “Complete Cluster Setup”

Deve ser executado após a instalação e reinicialização de todos os clientes do cluster. São feitas configurações no servidor, e preparações para deixar o cluster pronto para produção.

Passo 8: “Test Cluster Setup”

Após concluído a instalação o Oscar fornece alguns testes para verificar se os serviços instalados estão em perfeito funcionamento.

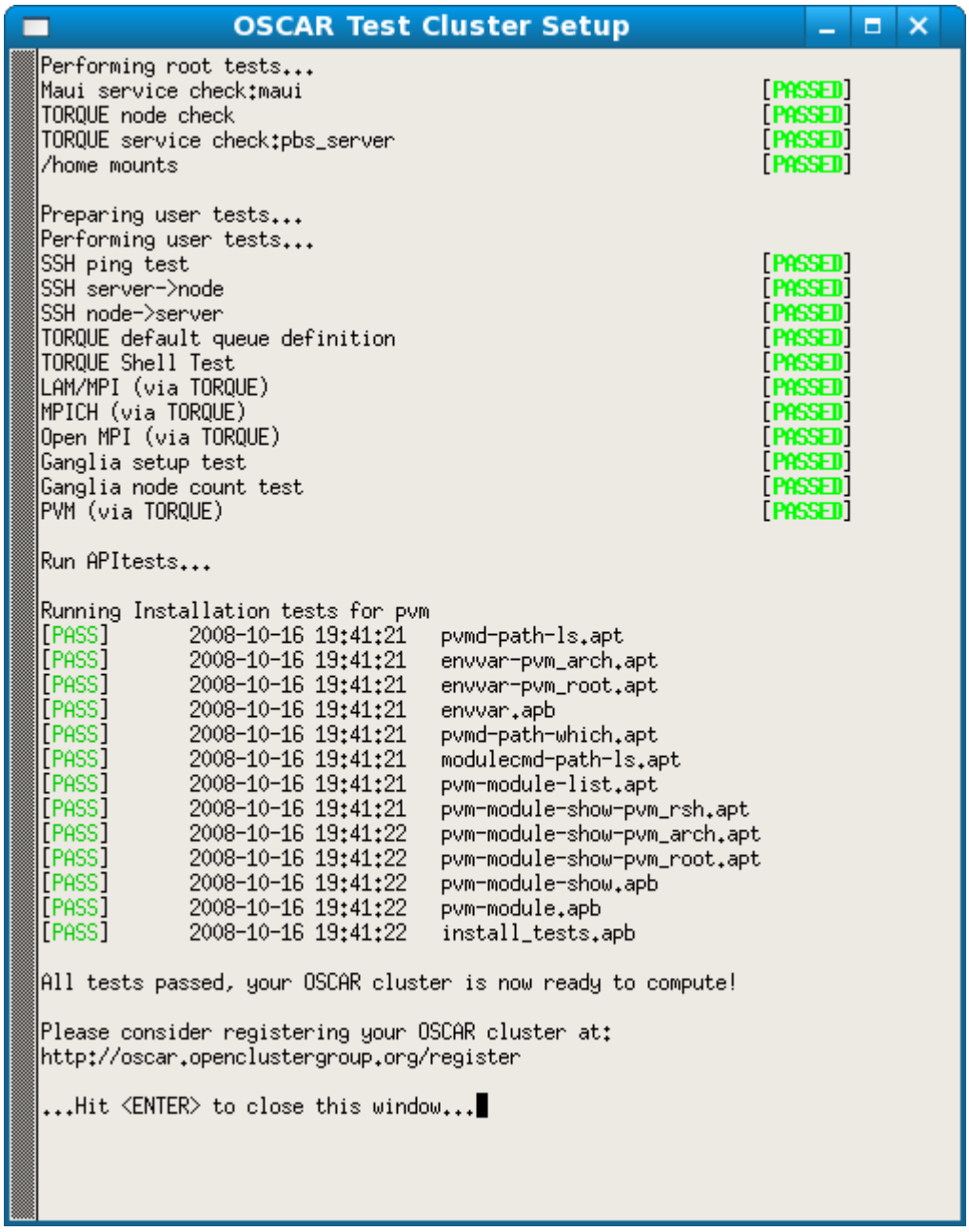


Figura 22 – Oscar Wizard – Teste do cluster

5.8 Utilização do Cluster

Existem algumas formas de se executar comandos e aplicações no cluster. Pode-se utilizar o Torque, que controla uma fila de "Jobs" e distribui-os entre os nodos do cluster, pode se forçar a execução de comandos através do "C3", pode-se executar programas feitos para a arquitetura MPI e ainda executar programas feitos para a arquitetura PVM.

5.8.1 Torque Resource Manager

Existem alguns comandos que podem ser executados para interagir com o Torque. Entre eles os principais são qsub, qdel, qstat, pbsnodes. Para submeter um processo para o Torque, é necessário rodar o comando qsub, e passar como parâmetro o arquivo de script do processo a ser executado, como no exemplo abaixo:

```
$ qsub -l nodes=X:ppn=Y:all,walltime=1:00:00 script.sh
```

Conteúdo do arquivo script.sh:

```
#!/bin/sh
#PBS -N my_jobname
#PBS -o my_stdout.txt
#PBS -e my_stderr.txt
#PBS -q workq
echo Launchnode is `hostname`
pbsdsh /path/to/my_executable
# All done
```

5.8.2 C3 Cluster Tools (Cluster Command Control)

As ferramentas do C3 são utilizadas tanto para administração quanto para suporte a aplicações. Possui ferramentas para execução de tarefas em todo o cluster, distribuição e compartilhamento de arquivos, termino de processos, desligamento e reinicialização remotos e atualizações de imagem.

Os principais comandos para essas ferramentas são:

- ✓ **qexec**: permite a execução de qualquer comando padrão do Linux nos nodos do cluster, podendo ser especificados quais nodos deverão executar.
- ✓ **cget**: recupera arquivos ou diretórios de todos os nodos ou dos que foram especificados.
- ✓ **ckill**: termina a execução de um processo especificado pelo usuário
- ✓ **cpush**: distribui arquivos ou diretórios para todos os nodos ou apenas para os nodos especificados.
- ✓ **cpushimage**: atualiza a imagem dos nodos do cluster, utilizando a imagem do SystemImager.
- ✓ **crm**: remove arquivos ou diretórios de todos os nodos ou apenas dos especificados.
- ✓ **cshutdown**: desliga um nodo especificado

- ✓ **clist**: retorna todos as informações do cluster.

A execução dos comandos quando não especificado nenhum nodo, é feita em todos ao mesmo tempo. Para fins de depuração, existe um comando equivalente ao cexec que executa os processos de forma serial. Esse comando é o cexecs.

5.8.3 LAM/MPI (Local Area Multicomputer)

LAM é o ambiente de programação MPI e o sistema de desenvolvimento para computadores heterogêneos em uma rede. Com essa implementação, o cluster ou infraestrutura de computadores pode ser utilizado como um único computador paralelo. É uma implementação completa do MPI-1 e implementa muita coisa do MPI-2.

Para rodar uma aplicação utilizando deve-se realizar os seguintes passos:

- ✓ Inicializar o Ambiente Run-time do LAM, com as configurações dos nodos que irão participar.

```
$ lamboot hostfile
```

- ✓ Compilar o programa a ser executado utilizando os wrappers fornecidos (mpicc,mpiCC ou mpi77). Eles adicionam flags no código e repassam para os compiladores reais.

```
mpicc myprogram.c –o myprogram
```

- ✓ Utilizar o comando mpirun para inicializar a aplicação no ambiente.

```
$ mpirun C myprogram
```

5.8.4 PVM (Parallel Virtual Machine)

PVM um pacote de sofwares que permitem que computadores heterogêneos, rodando Unix ou Windows, possam ser utilizados como um único computador. A instalação padrão do Oscar testa o PVM por meio de tarefas Torque/PBS. Porem e possível utiliza-lo fora desse contexto.

Para execução de uma aplicação deve se seguir o roteiro a seguir, baseado em uma aplicação de exemplo hello world:

Criar um diretório padrão para os executáveis do PVM

```
$ mkdir -p $HOME/pvm3/bin/$PVM_ARCH
```

Copiar os arquivos de exemplo para o diretório “hello”

```
$ cp $PVM_ROOT/examples/hello* $HOME/hello-example
$ cd $HOME/hello-example
```

Compilar o programa, utilizando os includes necessários e o path de busca de bibliotecas sendo as do PVM3

```
$ gcc -I$PVM_ROOT/include hello.c -L$PVM_ROOT/lib/$PVM_ARCH \
> -lpvm3 -o hello
$ gcc -I$PVM_ROOT/include hello_other.c -L$PVM_ROOT/lib/
$PVM_ARCH \
> -lpvm3 -o hello_other
```

Mover os arquivos gerados para o diretório padrão de busca do PVM

```
mv hello_other $HOME/pvm3/bin/$PVM_ARCH
```

Iniciar o PVM, adicionar hosts para a maquina virtual

```
$ pvm
pvm> add oscarnode1
add oscarnode1
1 successful
                        HOST DTID
                  oscarnode1 80000
pvm> quit
```

Rodar a aplicação

```
$ ./hello
i’m t40005
from t80002: hello, world from oscarnode1.localdomain
4.9.2 Using PVM 16
```

Finalizar o PVM

```
$ pvm
pvmd already running
pvm> halt
halt
Terminated
```

5.9 Monitoramento do Ambiente

Com a instalação padrão do Oscar existem algumas facilidades básicas para monitoramento do ambiente. Para um monitoramento mais completo, o Oscar também já vem com uma ferramenta bastante completa. O Ganglia, que já é instalado junto com o ambiente do Oscar, possui uma interface web com diversas informações sobre o ambiente do cluster e também sobre um ou mais nodos específicos.

Na Figura 23 é mostrado a tela principal de monitoramento do ambiente do Ganglia, onde são mostrados informações como número de processadores, memória total e tráfego de rede do cluster.

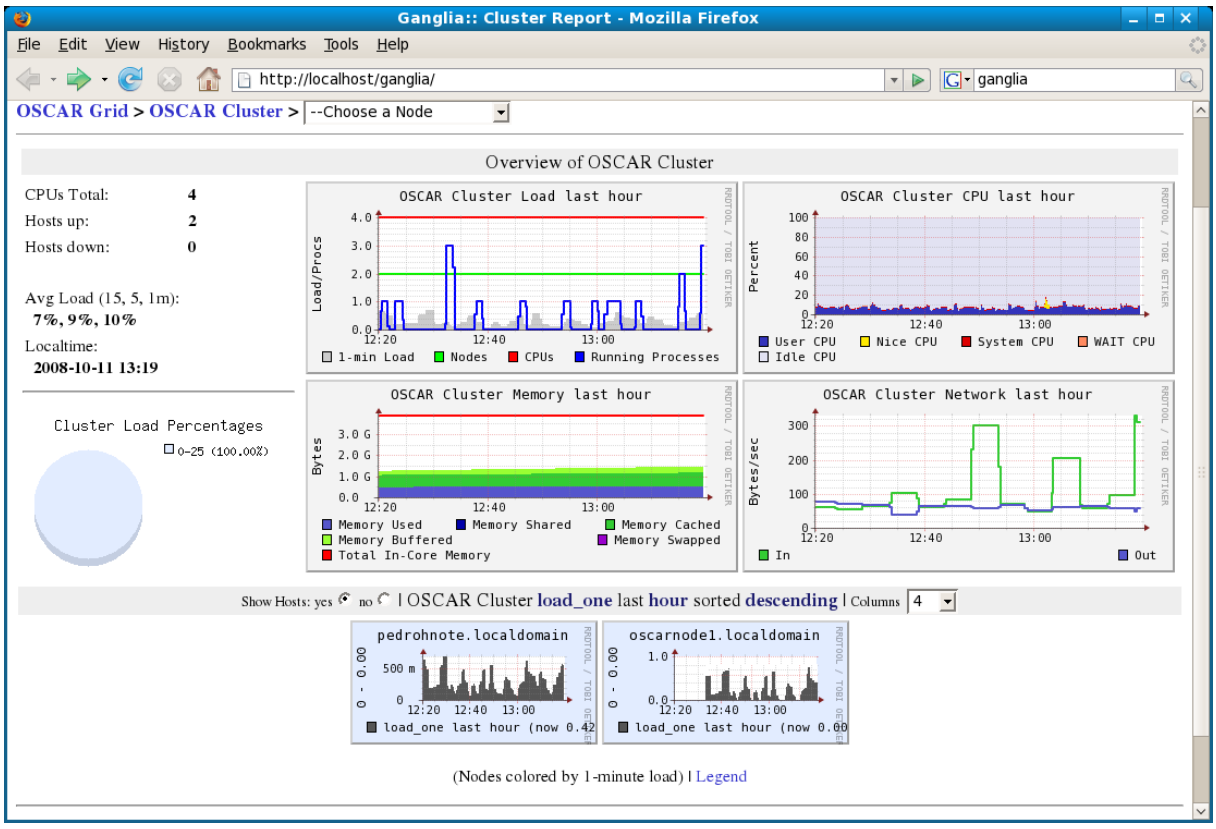


Figura 23 - Ganglia - Visão geral do cluster

Na Figura 24 e na Figura 25 são mostrados as telas que contem as informações específicas de um dos nodos do cluster, com diversos gráficos, o que facilita uma melhor análise do que está acontecendo com o processador. Essas informações são úteis para uma análise de performance do cluster.

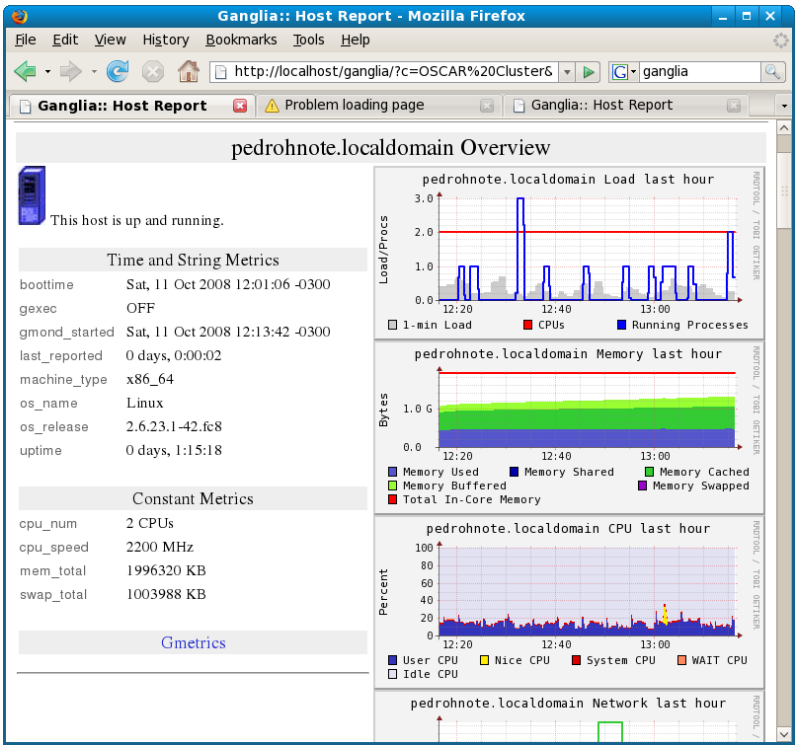


Figura 24 - Ganglia - Detalhe do nodo

Figura 25 - Ganglia - Graficos do nodo

6 CONCLUSÃO FINAL

Neste trabalho foram apresentados conceitos básicos para o entendimento do

funcionamento de um ambiente distribuído. Foi também detalhado a instalação, configuração e utilização de um ambiente de Cluster, utilizando ferramentas e sistemas operacionais gratuitos. Durante a instalação e configuração surgiram algumas dificuldades, a maioria contornada sem grandes dificuldades. Algumas delas por inexperiência com o ambiente Linux.

A configuração do ambiente de cluster foi feita utilizando-se maquinas e equipamentos comuns, fáceis de ser adquiridos e com um custo relativamente baixo. O propósito dessa escolha foi demonstrar que é possível criar um ambiente de grande potencial computacional sem arcar com grandes investimentos. Uma escola que possuía um laboratório com algumas máquinas simples de uso geral já poderia se beneficiar com a configuração de um cluster com as mesmas características do que foi demonstrado nesse trabalho. Um ambiente desse configurado pode ser útil para a execução de programas mais complexos e que normalmente levam um certo tempo, como por exemplo calculo de folha de pagamento e processamento de matrículas.

6.1 Trabalhos futuros

No ambiente experimental foram executados apenas aplicações simples, para demonstrar o funcionamento, futuramente pode-se criar aplicações mais complexas, para medição da melhora de desempenho ampliando-se a estrutura física do cluster.

Outro trabalho que pode ser realizado é a execução de uma mesma aplicação em um ambiente diferente, como por exemplo o Openmosix, para que as informações de tempo de execução e uso de recursos possam ser comparadas.

7 REFERÊNCIAS BIBLIOGRÁFICAS

[AVIZIENIS, 2001] AVIZIENIS, A.; LAPRIE, J.-C.; RANDELL, B. Fundamentos **Concepts of Dependability.** [S.l.]: Research Report, 2001.

[CASAVANT; KUHL, 1988] CASAVANT, T.; KUHL, J. **A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems.** [S.l.]: IEEE Transactions on Software Engineering, 1988. Vol. 14, p. 141-154.

[DANTAS ,2005] DANTAS, Mário . Computação **Distribuída de Alto Desempenho**. Editora Axcel, 2005.

[DENISE, 2007] FERREIRA, Denise Janson. **MAD-RSSF:** Uma Infra-estrutura de Monitoração Integrando Redes de Sensores Ad-Hoc e uma Configuração de Cluster Computacional. 2007. Dissertação (Bacharelado Ciências da Computação) UFSC, Florianópolis.

[FLYNN, 1972] FLYNN, M. **Some Computer Organizations and Their Effectiveness**. [S.l.]: IEEE Transation on Computers, 1972. Vol. C-21, pp. 948-960.

[GAJSKI,PIER; 1985] GAJSKI, D.; PIER, K. **Essencial Issues in Multiprocessors Systems**. [S.l.]: IEEE Computer Magazine, 1985. Vol. 19, pp. 9-27.

[GIL, 1999] GIL, Antonio Carlos. **Métodos e técnicas de pesquisa social**. 5. ed. São Paulo: Atlas, 1999.

[GLUNIX, 2008] **GLUnix**. Disponível em: http://now.cs.berkeley.edu/Glunix Acesso em ago 2008.

[MATTAR, 2000] MATTAR, Fauze N. **Pesquisa de marketing**: edição compacta. 2. ed. São Paulo: Atlas, 2000.

[MOORE, 1965] MOORE, G. E. **Cramming more Components onto Integrated Circuits**, Eletronics, Volume 38, Number 8, April 19, 1965. Disponível em:

<ftp://download.intel.com/research/silicon/moorespaper.pdf> Acesso em: julho 2008

[NOW, 2008] **NoW** Disponível em: http://now.cs.berkleley.edu. Acesso em ago. 2008

[OSCAR, 2008] **Oscar** - Open Source Cluster Apllication Resources. Disponível em <http:// http://svn.oscar.openclustergroup.org/trac/oscar>. Acesso em jul. 2008.

[PATTERSON; HENNESSY, 2000] PATTERSON, D. A.; HENNESSY, J. L. **Organização e Projeto de Computadores**. Rio de Janeiro, RJ: LTC, 2000. 551 p. Tradução de Nery Machado Filho.

[RAUPP; BEUREN, 2004]RAUPP, F.M., BEUREN, I. M. **Metodologia da pesquisa**. In: BEUREN, I. M. (Org.). Como elaborar trabalhos monográficos em contabilidade: teoria e prática. São Paulo: Atlas, 2004. p. 76 – 97.

[SETI, 2008] **Seti@Home**. Disponível em: http://setiathome.berkeley.edu. Acesso em: jul. 2008

[SCHALLER, 1997] SCHALLER, R.R. **Moore's law: past, present and future**. IEEE Spectrum, v. 34, n. 6, p.52-59, 1997.

[TANENBAUM, 2001] TANENBAUM, A. S. **Organização Estruturada de Computadores**. Rio de Janeiro, RJ: LTC, 2001. 398 p. Tradução de Nery Machado Filho.

[TRELEAVEN,1985] TRELEAVEN, P. **Control-Driven, Data-Driven, and Demand-Drivem Computer Architecture**. [S.l.]: Parallel Computing, 1985. Vol. 2.
TUECKE, S. et al. Open Grid Services Infrastructure

[WIKIPEDIA-FLYNN, 2008] **WIKIPEDIA.** Disponível em: http://en.wikipedia.org/wiki/Flynn'_taxonomy Acesso em: ago. 2008.

[WIKIPEDIA-MOORE, 2008] **WIKIPEDIA.** Disponível em: http://pt.wikipedia.org/wiki/Lei_de_Moore Wikipédia. Acesso em: Acesso em: jul. 2008.

Estudo de Ferramentas de Software para Melhoria de Desempenho em Ambiente Cluster Computacional

Pedro Henrique Buoro Albertini

Bacharelado em Ciências da Computação, 2008
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC), Brasil, 88040-900
pedroalbertini@inf.ufsc.br

Resumo

Os agregados computacionais vem cada vez mais sendo empregados como solução para a computação de alto desempenho. Mais baratos e com maior possibilidade de escalabilidade, os clusters são uma alternativa bem mais viável para aplicações que demandam grande quantidade de memória e processamento. Este trabalho mostra alguns conceitos básicos da computação distribuída de alto desempenho, a implementação de um cluster utilizando equipamentos comuns, sistema operacional Linux e o middleware de cluster Oscar, e as ferramentas para execução e monitoramento de aplicações nesse ambiente.

Palavras-chave: Agregado Computacional. Oscar. Computação Distribuída. Computação de Alto-Desempenho.

Abstract

Computer Clusters has been increasingly used as solution for high performance computing. Cheaper and easier to scale, cluster are a more viable alternative to run applications that require great amount of memory and processing. This work show some basic concepts of distributed high performance computing, a implementation of a cluster using ordinary equipments, opensource operational system, the Oscar cluster middleware and tools for execution and monitoring applications in this environment.

Keywords: Cluster Computing. Oscar Middleware. Distributed Computing. High Performance Computing.

Introdução

Desde o surgimento das primeiras redes que interligavam microcomputadores, no início dos anos setenta, a idéia de rodar aplicações distribuídas em maquinas separadas para otimizar o uso dos ciclos ociosos de cada uma, evitando o desperdício de recursos não utilizados já era visualizada. Nessa época onde ainda não se falava em Internet, a primeira rede Ethernet instalada pelo centro de pesquisas PARC(Palo Alto Research Center) [DANTAS, 2005] tinha como objetivo principal prover uma espinha dorsal para aplicações distribuídas.

Algum tempo depois, já nos anos noventa, alguns resultados interessantes foram obtidos por pesquisas que visavam resolver problemas conhecidos e difíceis de se resolver computacionalmente utilizando-se apenas uma máquina, o que tornou uma realidade o desenvolvimento de aplicações que requeriam o uso de diversas máquinas, como por exemplo o projeto Seti@Home (Search for Extraterrestrial Intelligence)[SETI,2008] que usa computadores de qualquer pessoa com acesso a Internet para analisar dados de satélites e buscar por indícios de vida inteligente fora do planeta.

Estamos vivendo um período de grande avanço tecnológico. Cada dia surgem inúmeros produtos e serviços que vem para facilitar nossa vida, mudando o modo de interação com as pessoas e organizações em geral. E nos deixando cada vez mais acostumados a isso. Porem nos últimos anos a tecnologia de microprocessadores vem esbarrando em limitações físicas, que fazem com que o ritmo de avanço tecnológico nessa área não consiga ser mantido. O que foi previsto por Gordon Moore em 1965, conhecido como a Lei de Moore[WIKIPEDIA-MOORE, 2008], de que a cada 18 meses a capacidade dos microprocessadores dobrariam, mantendo os custos constantes funcionou por mais de 30 anos, mas começa a dar sinais de que não dura muito tempo.[SCHALLER, 1997]

Uma limitação na capacidade de processamento, não acompanhando a demanda da tecnologia leva a restrição de vários tipos de software, de Aplicativos domésticos a grandes servidores utilizados por empresas. O processamento distribuído vem como forma de contornar essas limitações. Pois um grupo de máquinas com tecnologia inferiores trabalhando juntas podem equivaler a uma máquina com processamento bem superior. O uso das maquinas em conjuntos vem sendo cada vez mais estudado pelo meio acadêmico e cientifico, visto a eminente quebra da lei de Moore.

Clusters

"As configurações de clusters, em termos de arquiteturas computacionais, podem ser entendidas como uma agregação de computadores de uma forma dedicada (ou não)." [DANTAS, 2005].

A utilização de clusters vem para atender um mercado existente relativo a sistemas com alto grau de escalabilidade e de disponibilidade, que deixou de ser suprido ou por não atingir patamares aceitáveis, ou por custos impeditivos por computadores de arquitetura simples. Seu objetivo principal é aumentar facilmente o poder de processamento, a capacidade de memória e tamanho de disco disponível a qualquer momento [PATTERSON; HENNESSY, 2000].

Normalmente os clusters são compostos por agregados de computadores convencionais, onde o custo para se adicionar um nodo a mais para agregar mais potencial é relativamente baixo. Mas nada impede que os nós do cluster sejam compostos por máquinas multiprocessadas, o que agregaria mais valor ainda ao cluster [DANTAS, 2005].

A escalabilidade é a palavra chave

dos ambientes de cluster, visto que todo ambiente pode crescer a medida em que novos recursos estiverem disponíveis, de forma incremental, sem impacto nenhum para a infra-estrutura já existente.

Ambiente de Programação

A configuração computacional nos ambientes paralelos e distribuídos é bem mais complexo do que nas configurações centralizadas, tornando desafiadores os esforços dos ambientes de programação, principalmente por causa da possibilidade de fazer uso de recursos que não estão disponíveis localmente, requerendo acesso e autenticação para ser utilizados.

Diversos modelos de programação devem serconsiderados para as diferentes arquiteturas dos sistemas, em relação a possíveis direren;cãs de largura de banda, possíveis retardos de comunicação, diferenças de hardware, sistemas operacionais e linguagens e diferentes políticas de segurança. Serão mostrados explorados nesse capitulo três modelos que são considerados por Dantas [DANTAS, 2005] os mais expressivos, Web Services para ambientes distribuídos e Parallel Virtual Machine (PVM) e Message Passing Interface (MPI) para ambientes paralelos.

Middleware

O ambiente de Middleware como definido por Dantas:

"Ambiente que provê para os usuários dos clusters e grids computacionais uma transparência segura de acesso, facilidade de submissão das aplicações, gerenciamento de recursos e serviços da configuração distribuída geograficamente. Os ambientes de middlewares também têm como função o controle e permissão de instalação de pacotes de software nos nós da configuração, provendo um mecanismo específico de tratamento de falha de funcionamento do ambiente, quando do mau funcionamento de um determinado componente de hardware ou software." [DANTAS, 2005]

Ou seja, é um pacote de software que auxilia desenvolvedores a abstraírem as características específicas do ambiente. Com essa abordagem a preocução da execução do aplicativo em ambientes diferentes não existe, pois o middleware gerencia essas diferenças.

Ambiente Experimental

O Ambiente de cluster montado para esse experimento foi bem caseiro, utilizando apenas computadores comuns, os quais estamos acostumados a conviver no nosso dia a dia. Foram utilizados para compor o cluster um notebook HP, modelo tx1410, utilizado como servidor, e um PC padrão, ambos dispondo de 2Gb de memória, mais do que o suficiente para o que é desejado se demonstrar nesse trabalho. A rede utilizada é a extensão de uma rede residencial já existente.

O notebook foi escolhido para ser utilizado como servidor devido a sua mobilidade, pois a princípio a intenção era que ele poderia ser levado para outros ambientes com outras máquinas, e essas se conectariam a ele para fazer parte do cluster. Visto que as configurações do ambiente de cluster dependem basicamente da configuração do servidor, e que a adição de um novo cliente é mais simples e rápida.

O projeto OSCAR visa suportar as mais diversas distribuições do Linux. Quando novas codificações são feitas elas são ajustadas e testadas em todas as distribuições suportadas anteriormente. Junto com as distribuições das versões do Oscar é distribuído um conjunto de scripts que ajuda a compilar as bibliotecas de pacotes do Oscar em qualquer distribuição do Linux. [OSCAR, 2008].

A versão do Oscar escolhida foi a

última disponível em julho de 2008, que é a versão 5.1b. Apesar de ser uma versão beta, foi escolhida por suportar distribuições mais recentes do Linux. Já que o hardware tanto do Notebook quanto do PC que serão utilizados são mais recentes, e poderia haver incompatibilidades com distribuições mais antigas.

Por facilidade de instalação, para não ser necessário recompilar os pacotes do Oscar, foi escolhido então para ser instalado no ambiente a distribuição Fedora Core 8.

Conclusão

Durante a instalação e configuração surgiram algumas dificuldades, a maioria contornada sem grandes dificuldades. Algumas delas por inexperiência com o ambiente Linux.

A configuração do ambiente de cluster foi feita utilizando-se maquinas e equipamentos comuns, fáceis de ser adquiridos e com um custo relativamente baixo. O propósito dessa escolha foi demonstrar que é possível criar um ambiente de grande potencial computacional sem arcar com grandes investimentos. Uma escola que possuía um laboratório com algumas máquinas simples de uso geral já poderia se beneficiar com a configuração de um cluster com as mesmas características do que foi demonstrado nesse trabalho. Um ambiente desse configurado pode ser útil para a execução de programas mais complexos e que normalmente levam um certo tempo, como por exemplo calculo de folha de pagamento e processamento de matrículas.

Referências Bibliográficas

[DANTAS ,2005] DANTAS, Mário . **Computação Distribuída de Alto Desempenho**. Editora Axcel, 2005.

[SETI, 2008] **Seti@Home**. Disponível em: <http://setiathome.berkeley.edu>. Acesso em: jul. 2008

[WIKIPEDIA-MOORE, 2008] **WIKIPEDIA.** Disponível em: http://pt.wikipedia.org/wiki/Lei_de_Moore Wikipédia. Acesso em: Acesso em: jul. 2008.

[PATTERSON; HENNESSY, 2000] PATTERSON, D. A.; HENNESSY, J. L. **Organização e Projeto de Computadores**. Rio de Janeiro, RJ: LTC, 2000. 551 p. Tradução de Nery Machado Filho.

[OSCAR, 2008] **Oscar -** Open Source Cluster Apllication Resources. Disponível em <http://svn.oscar.openclustergroup.org /trac /oscar>. Acesso em jul. 2008.