UNIVERSIDADE DE SÃO PAULO

FACULDADE DE FILOSOFIA, CIÊNCIAS E LETRAS DE RIBEIRÃO PRETO

DEPARTAMENTO DE FÍSICA E MATEMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM FÍSICA APLICADA À MEDICINA E BIOLOGIA

Otimização de um cluster de alto desempenho para o uso do programa PGENESIS em simulações biologicamente plausíveis em larga-escala de sistemas neurais

Vladimir Fabrício Pereira de Carvalho

Orientador: Prof. Dr. Antônio Carlos Roque da Silva Filho

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto da USP, como parte das exigências para a obtenção do título de Mestre em Ciências, Área: Física Aplicada à Medicina e Biologia.

Ribeirão Preto – SP

2007

Pereira de Carvalho, Vladimir F.

Otimização de um cluster de alto desempenho para o uso do programa PGENESIS em simulações biologicamente plausíveis em larga-escala de sistemas neurais./Vladimir Fabrício Pereira de Carvalho. - Ribeirão Preto, 2007. 101 p.

Dissertação (Mestrado) - Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, 2007.

Orientador: Prof. Dr. Antônio Carlos Roque da Silva Filho.

1. Clusters de computadores. 2. Simulação computacional

Dedico este trabalho ao meu padrinho que partiu cedo demais.

AGRADECIMENTOS

Agradeço ao meu orientador professor Roque pela paciência, apoio, oportunidade e imensurável ajuda.

Aos meus pais Antonio Carlos e Magali pelo incentivo incondicional e por nunca terem deixado que eu sequer cogitasse a desistência.

A minha amada esposa Claudia, pela força "extra" em todas as etapas e por ser a grande incentivadora do meu crescimento acadêmico, profissional e pessoal.

Aos irmãos Vanderson, Vitor e Vanessa pelos momentos em família.

Aos irmãos de velha data Douglas, Ivan, Renato, Edvaldo, Marcelo, Zé e Tio pela descontração nos momentos de escape e pelo entendimento da ausência nestes últimos tempos.

Aos meus avôs Antonio e Geraldo e avós Zulmira e Maria que serviram de modelo de perseverança.

Aos demais familiares e amigos que mesmo não cabendo uma dedicatória exclusiva me acompanharam até este momento.

Aos colegas de laboratório SISNE pelas conversas produtivas e perguntas bem colocadas que me obrigaram a encontrar respostas ao longo do trabalho.

Aos professores da pós graduação que me forneceram as ferramentas necessárias para este estudo.

À coordenação do departamento de Física Médica, principalmente ao professor Alexandre e a "moça da secretaria" Nilza.

Aos companheiros de trabalho da Proservices Informática que presenciaram o inicio do trabalho e da HPB Engenharia que estiveram do meu lado no final do processo.

A Deus por ter me permitido chegar até aqui.

A Capes pelo apoio financeiro no momento mais necessário.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

GENESIS – GEneral NEural SImulation System

PGENESIS – Parallel GEneral NEural SImulation System

Sisne – Laboratório de Sistemas Neurais

SMP – symmetric multi-processor, ou multi-processador simétrico

HT – hyperthreading

DSM – distributed shared memory

PVM – Parallel Virtual Machine

MPI – Message Passing Interface

APIC - Advanced Programmable Interrupt Controller

RSH – remote shell

SSH – secure remote shell

HPL – High performance Linpack

HD – hard disk ou disco rígido

SO – sistema operacional

MPICH - Message Passing Interface Chameleon

LAM - Local Área Multicomputer

TCP - Transmission Control Protocol

CPU - Central Processing Unit em inglês, ou Unidade Central de Processamento

Vnc - Virtual Network Computing

Gpl - General Public License (Licença Pública Geral)

Gnu - projeto GNU é um projeto iniciado por Richard Stallman em 1984 , com o objetivo de criar um sistema operacional totalmente livre

Flops - Floating point operations per second que, em português, quer dizer operações de ponto flutuante por segundo

RESUMO

A utilização de clusters de computadores na simulação de redes neurais biologicamente realistas desponta como uma solução para atender aos recentes estudos nesta área que estão desenvolvendo modelos biologicamente detalhados de células nervosas e sistemas neurais, modelos que estão se tornando cada vez mais complexos e consequentemente exigem um maior grau de processamento computacional para sua simulação.

Este trabalho teve como objetivo testar e avaliar o desempenho do cluster do Laboratório de Sistemas Neurais – SisNe na execução de uma simulação de uma rede de larga-escala de neurônios modelados segundo o formalismo de Hodgkin-Huxley, que pode ser tida como um protótipo das simulações realizadas de modo geral utilizando o neuro-simulador GENESIS em sua versão paralela PGENESIS. Utilizando de ajustes no hardware e principalmente de uma otimização do software utilizado no cluster foi possível melhorar o seu desempenho consideravelmente, provando que o uso de um cluster com uma simulação paralela é viável para o estudo de redes neurais biologicamente realistas.

ABSTRACT

The use of computer clusters for simulations of biologically realistic neural networks promises to be a solution to support recent studies in this field, which are based on nervous cell models with increasing levels of realism and complexity that demand large processing power to be implemented.

The objective of this work is to test and evaluate the performance of the cluster of the *Laboratório de Sistemas Neurais* (SisNe) in running a program that simulates a large-scale network of Hodgkin-Huxley like neurons, which can be seen as a generic prototype of realistic network models run under PGENESIS – the parallel version of the GENESIS neurosimulator. Via hardware adjustments and principally optimizations in the software utilized in the cluster it was possible to considerably improve the initial cluster performance, showing that the use of a cluster together with parallel programming is a viable alternative for biologically realistic neural network simulations.

1 Introdução

Os avanços na tecnologia de processamento e o barateamento dos dispositivos de informática têm levado ao aparecimento de sistemas computacionais cada vez mais velozes e poderosos, fazendo com que seja possível realizar simulações biologicamente detalhadas de complexas estruturas do sistema nervoso que até pouco tempo atrás eram apenas modeladas matematicamente e de forma simplificada (Bower e Beeman, 1998; Koch e Segev, 1998).

Como exemplos de ferramentas que surgiram recentemente para auxiliar na modelagem computacional do sistema nervoso têm-se os neuro-simuladores, como o GENESIS (Bower e Beeman, 1998) e o NEURON (Hines, 1998). Os neuro-simuladores são sistemas computacionais especializados para a simulação de modelos multi-compartimentais de células nervosas (Koch e Segev, 1998) em que os processos biofísicos relativos a cada neurônio são simulados em escala microscópica. A extensão espacial de cada neurônio é considerada pela sua partição em vários compartimentos, como o soma, o axônio e vários compartimentos dendríticos. Cada compartimento é modelado por equações diferenciais similares às equações de Hodgkin-Huxley (1952), que representam o comportamento da membrana celular e dos seus canais iônicos. Para a simulação de uma rede com muitos neurônios acoplados sinapticamente, um sistema de várias equações diferenciais acopladas tem que ser resolvido numericamente e programas como o GENESIS e o NEURON simplificam em muito essa tarefa.

Em particular, visando aproveitar melhor a capacidade de processamento e manipulação de dados tornada possível recentemente com o advento da computação paralela, tanto o GENESIS como o NEURON possuem versões desenhadas para

trabalhar com processamento paralelo, o PGENESIS (Bower e Beeman, 1998; Hood, 2002) e o parallel NEURON (Carnevale e Hines, 2006).

Segundo estudos de custo-benefício sobre simulações de sistemas neurais em diferentes plataformas de hardware, os supercomputadores ainda levam vantagem no quesito desempenho (Roth et al., 1995; Jahnke et al., 1998), mas ficam em grande desvantagem ao se considerar o custo. Uma tendência alternativa tem sido o desenvolvimento de hardwares específicos para a simulação de sistemas neurais, os chamados neurocomputadores (Hartmann et al., 1997; Schaefer et al., 2002). Porém, além de também ser uma tarefa de custo e demanda temporal relativamente alto, o desenvolvimento de neurocomputadores tem direcionado seu foco mais para aplicações em bioengenharia, como a construção de próteses de retinas, cócleas, sistemas óculo-motores, geradores centrais de padrões, etc do que para a pesquisa mais básica envolvendo simulações (experimentos *in silico*) de sistemas neurais (Omondi, 2000; Breslin e O'Lenskie, 2001).

Desta forma, uma solução de compromisso para se implementar simulações biologicamente plausíveis de sistemas neurais com um número razoável de neurônios e sinapses (da ordem de algumas dezenas de milhares) é a utilização de clusters de processadores rodando programas como o PGENESIS, por exemplo. Esta foi a solução adotada pelo Laboratório de Sistemas Neurais (SisNe), coordenado pelo orientador desta dissertação, que adquiriu recentemente um cluster com 10 nós de processamento para a execução de simulações de sistemas neurais em larga-escala (Oliveira e Roque, 2002; Mazza et al., 2004; Simões de Souza e Roque, 2004a,b; Simões de Souza, 2005; de Pinho et al., 2006; Oliveira, 2006; Publio et al., 2006).

Um cluster pode ser definido como um conjunto de nós processadores (PCs ou estações) autônomos e que interligados comportam-se como um sistema de imagem

única. O conceito de imagem única dita que um sistema paralelo ou distribuído, independente de ser composto por vários processadores ou recursos geograficamente distribuídos, deve comportar-se com um sistema centralizado do ponto de vista do usuário. Dessa forma, todos os aspectos relativos à distribuição de dados e de tarefas, comunicação e sincronização entre tarefas e a organização física do sistema devem ser abstraídos do usuário, ou seja, devem ser transparentes a ele (Pitanga, 2004).

Existem várias maneiras de montar um cluster, usando formas de comunicação e plataformas diferentes. Como exemplos de bibliotecas de sub-rotinas de comunicação, pode-se citar:

- *Parallel Virtual Machine* (PVM): a idéia do PVM é fazer com que uma coleção de *n* processadores heterogêneos (seriais, paralelos, vetoriais) seja vista como uma única máquina virtual (Geist et al., 1994). O PVM é composto de duas partes principais (Pitanga, 2004): (1) um processo do tipo *daemon*, chamado PVMD3, instalado em cada nó pertencente ao cluster e responsável pela coordenação das tarefas dos nós "mestre" e "escravos"; (2) o conjunto de bibliotecas (libpvm3, libfpvm3 e libgpvm3) que contêm as rotinas operáveis pelo usuário para, por exemplo, sincronizar tarefas ou enviar e receber mensagens entre os nós.
- *Message Passing Interface* (MPI): seu principal objetivo é definir um padrão de troca de mensagens com sintaxe definida, mas preservando características exclusivas de cada arquitetura, inclusive para arquiteturas de memória compartilhada (Snir e Gropp, 1998). O MPI é uma biblioteca de sub-rotinas padronizada, cada fabricante de computadores é responsável por desenvolver e otimizar uma biblioteca MPI para o seu ambiente paralelo de processamento.

Como exemplos de plataformas, pode-se citar o Microsoft Windows 2003 Datacenter, o IBM AIX e o GNU\Linux, sendo este último uma plataforma aberta e de distribuição gratuita de seus códigos-fontes, permitindo assim uma montagem com um custo reduzido e a possibilidade de alterações na sua estrutura visando uma otimização dos recursos disponíveis.

Como o sistema PGENESIS foi originalmente concebido para o uso com a biblioteca de troca de informação PVM e com a plataforma Unix e seus derivados (GNU/Linux, por exemplo), o cluster recentemente montado no Laboratório SisNe adotou o conjunto Linux e PVM como ponto de partida para seus trabalhos, porém uma versão do PGENESIS usando MPI foi recentemente desenvolvida (Hood, 2002) e existe a possibilidade de que ela seja testada pelo laboratório.

Apesar de já estar montado[1] fisicamente, o uso eficiente do cluster pelo Laboratório SisNe para a execução das suas simulações de sistemas neurais em larga-escala envolve configuração lógica da plataforma escolhida (Linux/PVM) e a otimização de muitas variáveis – tanto em nível de hardware como de software –, sendo que boa parte delas influencia diretamente no desempenho das simulações.

Com relação ao hardware, os processadores utilizados, do tipo Intel Xeon, possuem a característica *hyperthreading* (HT) que permite que um processador passe a executar um processo sem ter terminado o anterior, parecendo deste modo para o sistema operacional que existem dois processadores ao invés de apenas um. Isso permite aumentar a velocidade de processamento eliminando a latência, porém a configuração do *hyperthreading* tem que ser habilitada para o kernel do Linux. Ainda com relação aos processadores, os computadores usados no cluster são dual-processados, usando a tecnologia SMP (*Symmetric Multi-Processor Systems*) tendo na mesma placa dois

[1] O autor desta dissertação participou ativamente da montagem do cluster no Laboratório SisNe.

processadores. Como os dois processadores possuem a tecnologia *hyperthreading*, o sistema operacional pode enxergar, efetivamente, quatro processadores. Cada conjunto possui também duas placas de rede Gigabit e um Switch Gigabit que suporta tal velocidade, permitindo desta forma uma troca de informação muito rápida entre os nós que o compõem.

Com relação ao software, o conjunto Linux, PVM ou MPI e PGENESIS contêm uma enorme quantidade de variáveis lógicas que são decisivas para o uso adequado do hardware citado acima. Somando tudo, o cluster e o software constituem uma tecnologia avançada, mas que sozinha não garante bons resultados. É preciso uma personalização e otimização dos recursos disponíveis para obter do conjunto mais do que se conseguiria usando as partes isoladas.

O objetivo do trabalho descrito nesta dissertação foi o de testar e avaliar o desempenho do cluster do Laboratório SisNe, com diferentes configurações, na execução de uma simulação de uma rede de larga-escala de neurônios modelados segundo o formalismo de Hodgkin-Huxley, que pode ser tida como um protótipo das simulações realizadas pelo laboratório.

Os resultados do trabalho, apresentados ao longo do texto, fornecem uma avaliação detalhada do uso do cluster para a finalidade específica de realizar simulações em larga-escala de sistemas neurais biologicamente plausíveis. Espera-se, portanto, que eles oferecem uma contribuição para a área de desenvolvimento de sistemas computacionais para a modelagem computacional biologicamente plausível de estruturas neurais.

2 Conceitos básicos de computação, computação paralela e neurociência computacional

Antes de passar ao trabalho da dissertação propriamente dito, é interessante rever alguns fatos e definições que servem de suporte ao trabalho feito. Eles serão apresentados nas próximas seções deste capítulo.

2.1 Organização dos computadores

A primeira geração de computadores estabeleceu os conceitos básicos de organização dos computadores eletrônicos e na década de cinqüenta do século passado apareceu o modelo computacional que se tornaria a base de todo o desenvolvimento subseqüente, denominado “modelo de Von Neumann” (Pitanga, 2004).

Em sua proposta, von Neumann sugeriu que as instruções fossem armazenadas na memória do computador. Até então elas eram lidas de cartões perfurados e executadas uma a uma. Armazená-las na memória, para então executá-las, tornaria o computador mais rápido, já que, no momento da execução, as instruções seriam obtidas com rapidez eletrônica.

A maioria dos computadores de hoje em dia segue o modelo proposto por Von Neumann. Esse modelo define um computador seqüencial digital em que o processamento das informações é feito passo a passo, caracterizando um comportamento determinístico (ou seja, os mesmos dados de entrada produzem sempre a mesma resposta).

O modelo de Von Neumann é composto basicamente por um processador, uma unidade de memória e pelos dispositivos de entrada e saída de dados ou periféricos. O processador executa as instruções sequencialmente, de acordo com a ordem estabelecida por uma unidade de controle (Figura 1).

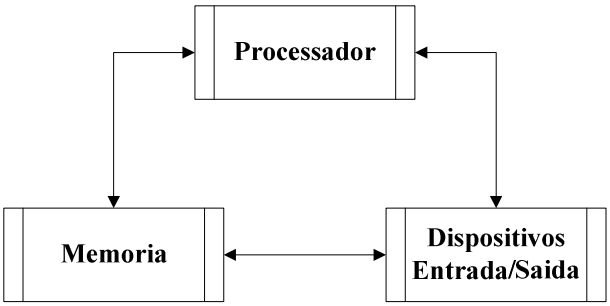


Figura 1 – Modelo de Von Neumann

Um refinamento do modelo de von Neumann é o chamado modelo de barramento de sistema. Neste modelo, a comunicação entre os três componentes (processador, memória e dispositivos de entrada e saída) é realizada através de um caminho compartilhado chamado de barramento de sistema (*bus*), constituído do barramento de dados (*data bus*), do barramento de endereços (*address bus*) e do barramento de controle (*control bus*). Existe também um barramento de energia e algumas arquiteturas podem ter um barramento de entrada/saída separado. Esta adaptação do modelo de von Neumann é a mais usada atualmente nos computadores de mercado (Figura 2).

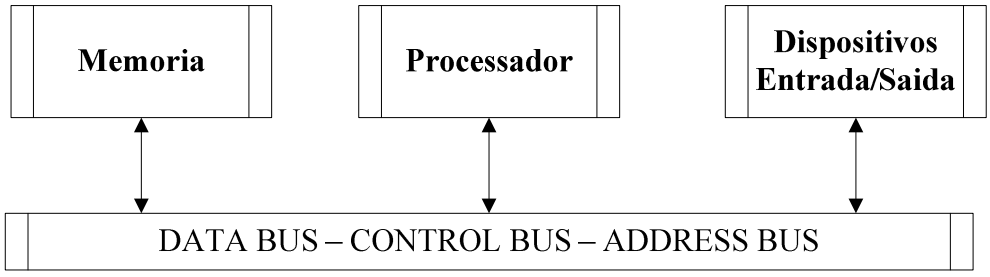


Figura 2 – Modelo barramento Sistema

Em 1972, Michael Flynn (Flynn, 1972) propôs caracterizar os modelos de arquitetura de computador segundo o número de fluxos de dados e de instruções existentes em cada instante, produzindo quatro classes de computadores, descritas a seguir.

SISD

Single Instruction Stream/Single Data Stream (Fluxo único de instruções/Fluxo único de dados). Corresponde ao tradicional modelo de Von Neumann. Um processador executa seqüencialmente um conjunto de instruções sobre um conjunto de dados (Figura 3).(Almasi, 1994)

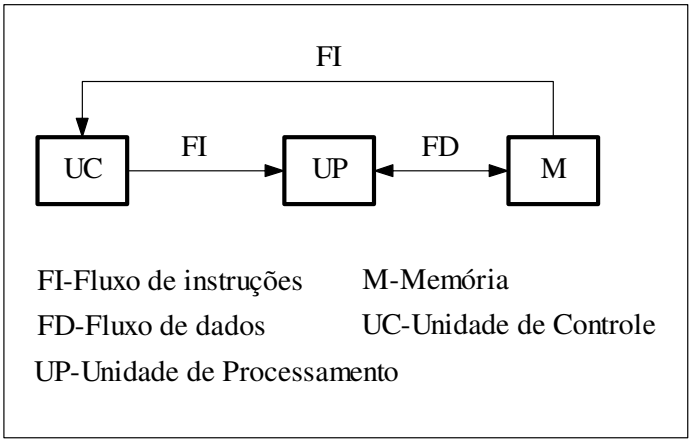


Figura 3 - Modelo Computacional SISD.(Adaptado Almasi, 1994)

SIMD

Single Instruction Stream/Multiple Data Stream (Fluxo único de instruções/Fluxo múltiplo de dados). Envolve múltiplos processadores (escravos) sob o controle de uma única unidade de controle (mestre), executando simultaneamente a mesma instrução em diversos conjuntos de dados. Arquiteturas SIMD são utilizadas, por exemplo, para manipulação de matrizes e processamento de imagens (Figura 4). (Almasi, 1994)

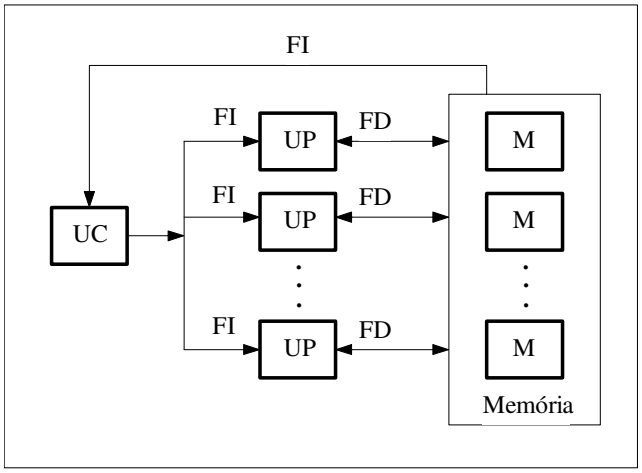


Figura 4 - Modelo Computacional SIMD.(Adaptado Almasi, 1994)

MISD

Multiple Instruction Stream/Single Data Stream (Fluxo múltiplo de instruções/Fluxo único de dados). Envolve múltiplos processadores executando diferentes instruções em um único conjunto de dados (Figura 5). Geralmente, nenhuma arquitetura é classificada como MISD, isto é, não existem representantes desta categoria. Alguns autores consideram arquiteturas *pipeline* como exemplo deste tipo de organização (Almasi, 1994).

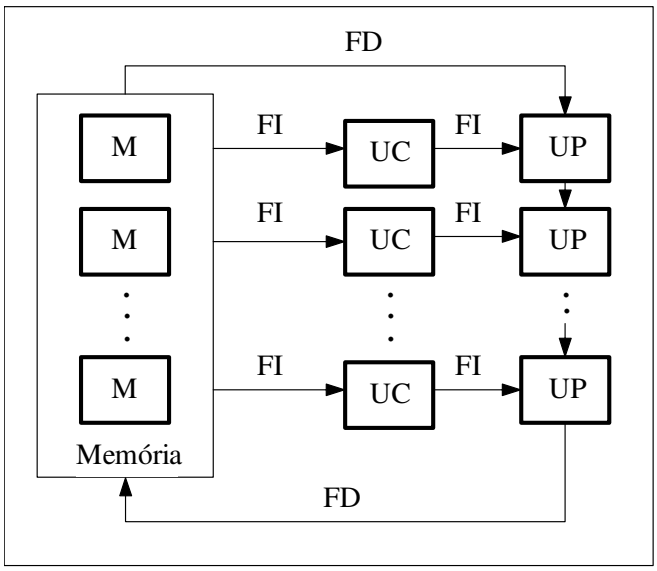


Figura 5 - Modelo Computacional MISD.(Adaptado Almasi, 1994)

MIMD

Multiple Instruction Stream/Multiple Data Stream (Fluxo múltiplo de instruções/Fluxo múltiplo de dados). Envolve múltiplos processadores executando diferentes instruções em diferentes conjuntos de dados, de maneira independente. Esta classe engloba a maioria dos computadores paralelos (Figura 6). (Almasi, 1994)

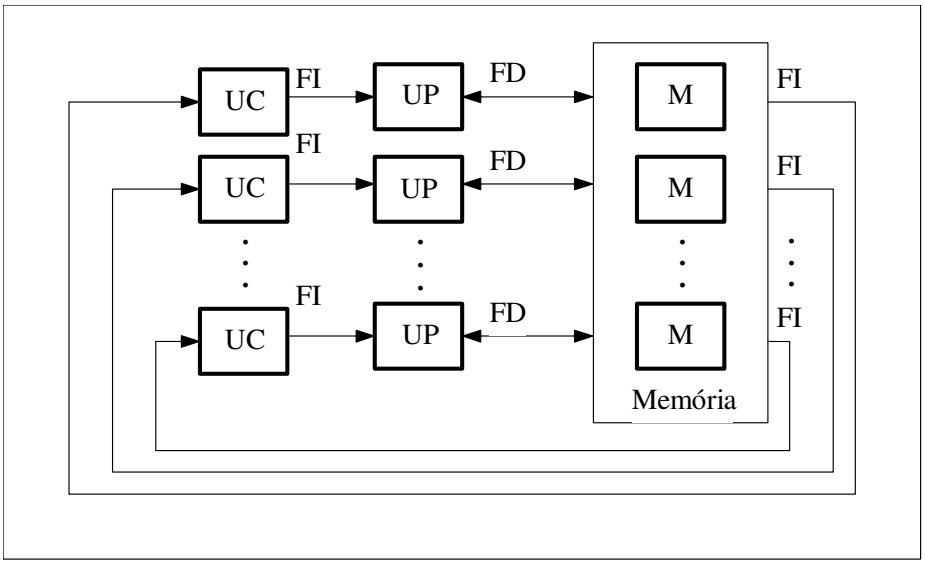


Figura 6 - Modelo Computacional MIMD.(Adaptado Almasi, 1994)

Todos os tipos de organização computacional apresentam vantagens e desvantagens. Arquiteturas SIMD, por apresentar fluxo único de instruções, oferecem facilidades para a programação e depuração de programas. Além disso, seus elementos de processamento são simples, pois são destinados à computação de pequena granulação. Por outro lado, arquiteturas MIMD apresentam grande flexibilidade para a execução de algoritmos paralelos (arquiteturas SIMD geralmente se destinam a processamento de propósitos específicos), e apresentam bom desempenho em virtude de seus elementos de processamento serem assíncronos (Almasi, 1994).

Embora a classificação de Flynn seja conveniente em uma primeira instancia, é bastante imprecisa para classificar as variedades de multiprocessadores existentes. A

categoria MIMD engloba uma grande diversidade de maquinas, havendo várias tentativas de classificá-las. Uma das formas mais usadas é uma classificação que usa como critério a forma como a memória central esta fisicamente organizada e o tipo de acesso que cada processador tem da totalidade da memória. Segundo essa abordagem, a arquitetura MIMD pode ser classificada em arquitetura de memória compartilhada e arquitetura de memória distribuída (Pitanga, 2004).

Memória compartilhada e memória distribuída

Os sistemas de memória compartilhada possuem uma única memória física que é acessível para todas as unidades processadores do sistema, sendo que a comunicação entre estes dispositivos é feita através de um barramento de comunicação de alta velocidade (Figura 7). Um exemplo deste tipo de sistema são os sistemas SMP (ver adiante).

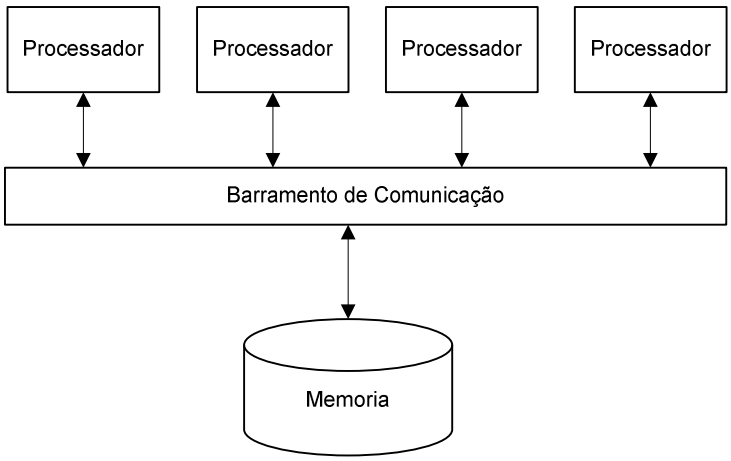


Figura 7 – Memória Compartilhada

Os sistemas de memória distribuída possuem várias unidades processadoras, cada uma com sua própria memória física, interligadas por de um dispositivo de comunicação (Figura 8).

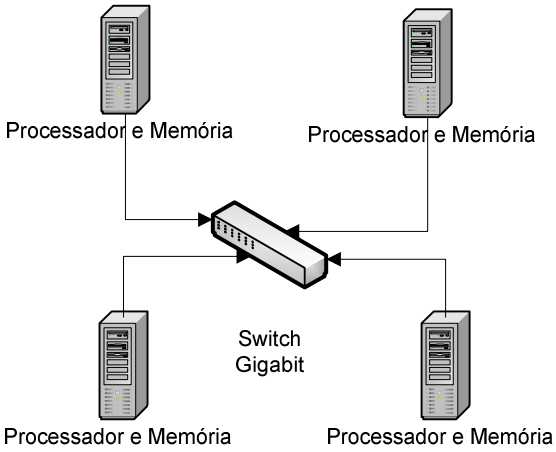


Figura 8 – Memória Distribuida

Existem também os sistemas de memória compartilhada e distribuída (DSM – *distributed shared memory*). Um sistema DSM é um conjunto de elementos que se comunicam através de uma rede de interconexão e que utilizam software de sistema distribuído. Cada elemento é composto por um ou mais processadores e uma ou mais memórias. As vantagens de um sistema DSM incluem a possibilidade de seu crescimento incremental, ou seja, grande escalabilidade, e a possibilidade de implementação de aplicações distribuídas e de sistemas tolerantes a falhas por meio de replicação de serviços. Um sistema DSM é um conjunto de computadores com seu próprio clock e que não possuem memória compartilhada. Ele é visto pelos seus usuários como um sistema de recurso único, que se comunicam com ele através de um barramento de troca de mensagens (Pitanga, 2004).

Tipos de Acoplamento

Nos chamados sistemas *fortemente acoplados*, existem vários processadores compartilhando uma única memória e gerenciados por apenas um sistema operacional. Múltiplos processadores permitem que vários programas sejam executados ao mesmo tempo, ou que um programa seja dividido em subprogramas, para execução simultânea em mais de um processador. Dessa forma, é possível ampliar a capacidade de

computação de um sistema, adicionando-se apenas novos processadores, com um custo muito inferior à aquisição de outros computadores (Barcellos, 2006). Fazem parte desta categoria os computadores SMP.

Os sistemas *fracamente acoplados* caracterizam-se por possuir dois ou mais sistemas de computação (multicomputadores), conectados através de linhas de comunicação. Cada sistema funciona de forma independente, possuindo seu(s) próprio(s) processador(es), memória e dispositivos. A utilização de sistemas fracamente acoplados já é caracterizada pelo processamento distribuído entre os seus diversos processadores. Dentre os sistemas fracamente acoplados têm-se os sistemas operacionais distribuídos, que têm como característica a existência de um relacionamento mais forte entre os seus componentes, onde geralmente os sistemas operacionais são os mesmos (Barcellos 2006). Para o usuário e suas aplicações, é como se não existisse uma rede de computadores, mas sim um único sistema centralizado. Fazem parte desta categoria os clusters de computadores como o que foi utilizado neste estudo.

Sistemas SMP

Na arquitetura SMP (*symmetric multi-processor*, ou multi-processador simétrico), um grupo de processadores trabalha conjuntamente, compartilhando uma única memória através de um barramento único, sendo possível para qualquer um deles executar uma parte do programa. Neste sistema, o barramento de memória é o eixo de simetria entre os processadores. Assim, nenhum poderá acessar diretamente ~~a~~ uma memória ou ~~a~~ um dispositivo de entrada/saída.

O multiprocessamento simétrico trata todos os processadores de forma igual. Qualquer processador pode fazer o trabalho de qualquer outro processador e as aplicações são divididas em correntes que podem rodar concorrentemente em qualquer

processador disponível. O SMP melhora o desempenho da própria aplicação, como também o processamento total do sistema. Os sistemas SMP requerem alguma forma de memória compartilhada e CACHE s de instrução locais, mas, mais importante, sistemas SMP requerem aplicações que podem tirar proveito do paralelismo.

Clusters de computadores

A palavra inglesa *cluster* significa agrupamento, de maneira que podemos chamar de *cluster de computadores* um sistema onde dois ou mais computadores trabalham de maneira conjunta para realizar processamento em paralelo. Em outras palavras, os computadores dividem as tarefas de processamento e trabalham como um único computador (Pitanga, 2004).

Os tipos mais comuns de cluster de computadores são listados a seguir.

Cluster Beowulf: Trata-se do primeiro tipo de cluster, quando tudo começou. Originou-se de um projeto financiado em parte pela NASA para estudar a viabilidade da computação paralela no processamento de grandes quantidades de dados pelos cientistas espaciais, que entrou em operação em 1994 (Merkey, 2004). Consiste de vários computadores conectados através de uma rede ethernet que trabalham trocando informações entre si e acessados através de um computador chamado de *frontend*, transparecendo para o operador se tratar de uma maquina única.

OpenMosix: A tecnologia OpenMosix foi criada em 2002 por Bar (2006), como uma alternativa *open-source* ao Mosix, que se tornou proprietário em 2001, e conta atualmente com vários desenvolvedores espalhados pelo mundo. Ao contrário do Beowulf, trata-se de uma arquitetura de clusters onde os computadores não são dedicados à execução dos *jobs*, podendo continuar a ser utilizados normalmente se utilizando, de forma transparente e quando necessário, dos ciclos de CPU ociosos dos outros nós do cluster. Além disso, sua arquitetura é

baseada na distribuição de processos e não na segmentação de um único processo em sub-tarefas que se comunicam entre si por trocas de mensagens, como no Beowulf. Ou seja, os programas precisam ser compostos por múltiplos processos para se aproveitar de um cluster OpenMosix, mas um programa já construído dessa forma não precisa ser modificado para realizar trocas de mensagens, por exemplo. Resumindo, um programa com três instâncias, cada uma em execução simultânea em um nó do cluster, terá seu tempo de execução reduzido a 1/3, em comparação à execução em uma única máquina (Bar, 2006).

Cluster de Alta disponibilidade: Sistema usado em tarefas de missão critica, onde a disponibilidade é o objetivo, normalmente formados por unidades de processamento independentes que utilizam a mesma base de dados. Uma unidade de processamento fica em estado de espera e só entra em operação em caso de falha da principal e um sistema de gerenciamento de informações garante a mudança de maquinas sem perda de dados. Exemplos desse tipo de cluster são os servidores de instituições financeiras, controle de trafego e segurança nacional.

Cluster de balanceamento de carga: Sistema usado para garantir o acesso da aplicação mesmo sob forte demanda. Possui uma configuração similar ao cluster de alta disponibilidade, mas não possui unidade em espera e sim duas ou mais unidades que se revezam no atendimento às solicitações dos usuários, mantendo desta maneira uma performance aceitável. São exemplos sites de busca, sites governamentais e servidores de DNS.

Cluster de alta performance: Sistema usado para resoluções de problemas e uso de aplicações que exigem alto grau de processamento. O cluster Beowulf e o Mosix se encaixam neste item e suas aplicações são das mais variadas, por exemplo, simulações do meio ambiente, renderização de imagens (atualmente muito usada em animações para os cinemas) e resolução de algoritmos complexos.

2.2 Elementos de programação paralela

Pode-se classificar os tipos de programação em três tipos.

Programação seqüencial: caracteriza-se pela execução de várias tarefas uma após a outra. É a mais utilizada e para que cada etapa do programa seja executada a anterior obrigatoriamente tem que ser finalizada.

Programação concorrente: caracteriza-se pela iniciação de várias tarefas sem que as anteriores tenham sido necessariamente terminadas. Pode trabalhar com sistemas monoprocessados, mas é feita visando à utilização de sistemas multiprocessados.

Programação paralela: Caracteriza-se pela iniciação e execução das tarefas em paralelo. Pode ser composta apenas de tarefas concorrentes ou de tarefas seqüenciais e concorrentes.

Para ilustrar a diferença entre a programação paralela e a seqüencial, dá-se a seguir um exemplo de um mesmo algoritmo para simulação de um fenômeno utilizando as visões da programação seqüencial e da paralela.

Supondo um fenômeno físico, que possa ser explicado por uma equação composta e simulado da seguinte forma:

Fenômeno = Equação 1 + Equação 2 + Equação 3.

Considerando que cada equação que compõe a simulação do fenômeno é independente das demais, é possível desenvolver um algoritmo independente para encontrar a solução de cada uma. Portanto, o resultado da simulação do fenômeno pode ser obtido com a resolução desses algoritmos e pela posterior soma dos resultados.

Caso essa simulação seja projetada para ser executada em um único computador ela seguirá o processo de programação seqüencial e o programa que contém os três algoritmos desenvolvidos seguiria o seguinte fluxo.

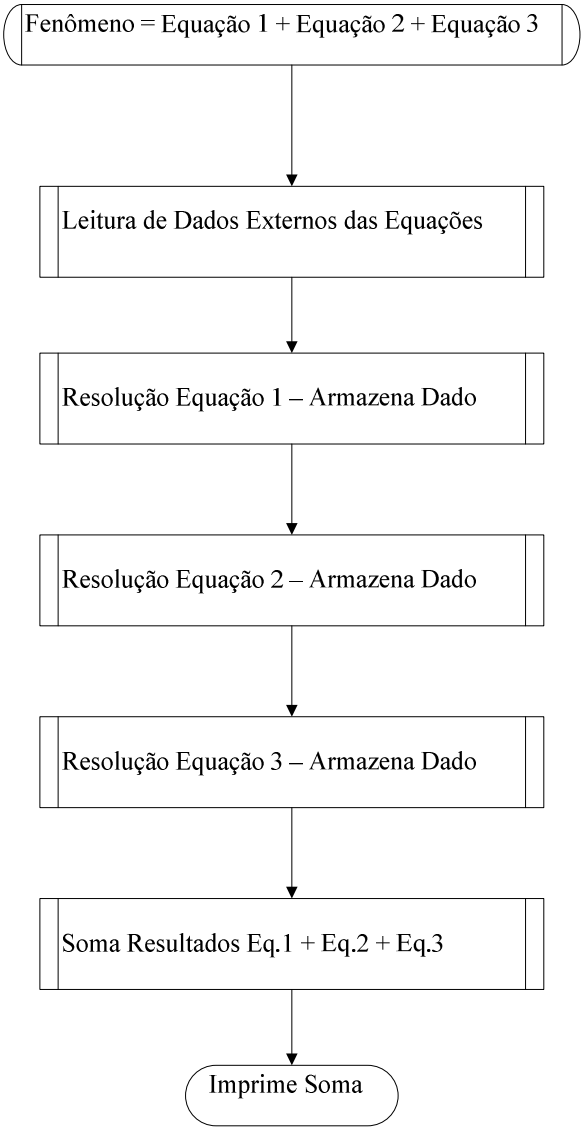


Figura 9 – Programação seqüencial

Como dito anteriormente, as três equações que compõem o fenômeno são independentes entre si. Desta forma, elas podem ser processadas paralelamente sem alterar o valor final da equação, tornando possível usar um algoritmo que utiliza programação paralela para agilizar a resolução do problema. Uma maneira possível de efetuar isso é criar processos filhos independentes a partir do processo pai (que seria a simulação do fenômeno). Utilizando esta maneira de visualizar o problema, cria-se um processo pai e três processos filhos, um para cada equação do fenômeno, resultando em um novo fluxo dado abaixo.

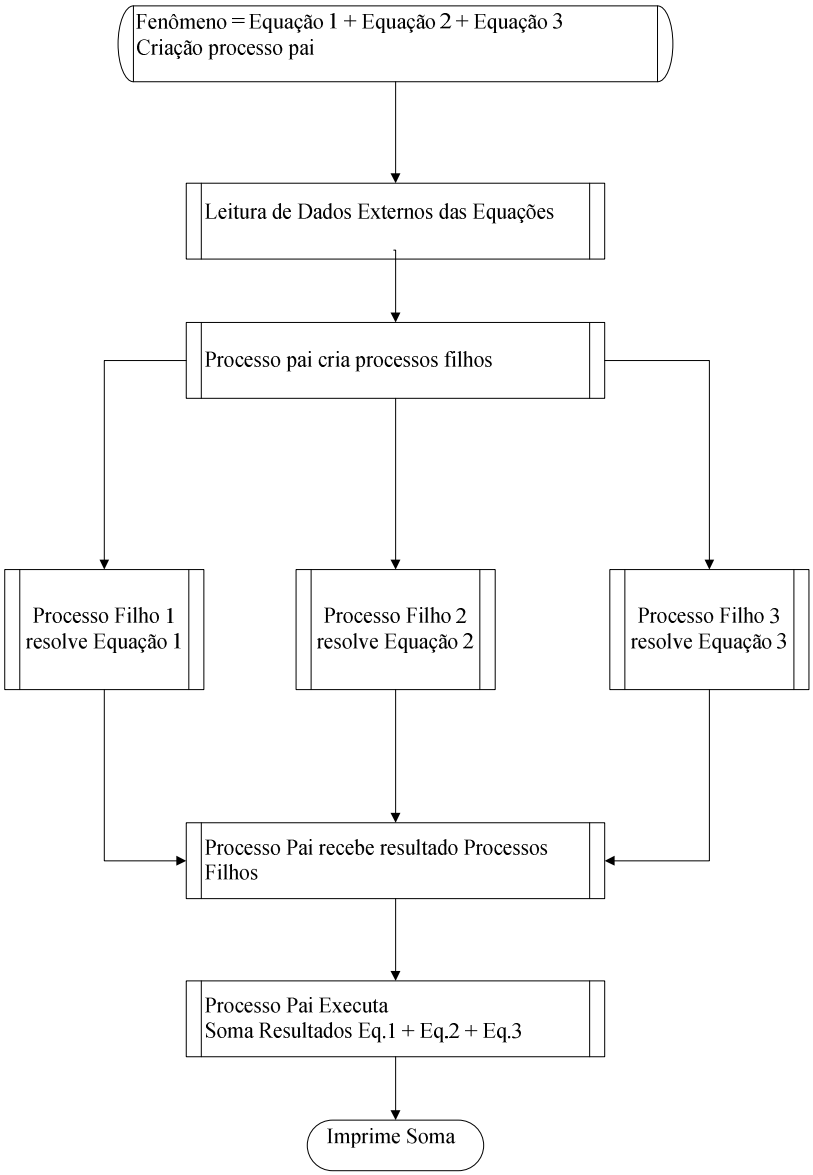


Figura 10 – Programação Paralela

Note-se que este último algoritmo usa tanto a programação seqüencial (execução apenas do processo pai) como a concorrente (execução dos processos filhos).

2.3 Neurociência computacional e o GENESIS

A neurociência computacional usa equações matemáticas e simulações computacionais para modelar as funções cerebrais. Nos chamados modelos biologicamente plausíveis, cada neurônio é modelado por compartimentos elétricos conectados entre si por resistências elétricas que simulam a resistência do interior da célula (Figura 11).

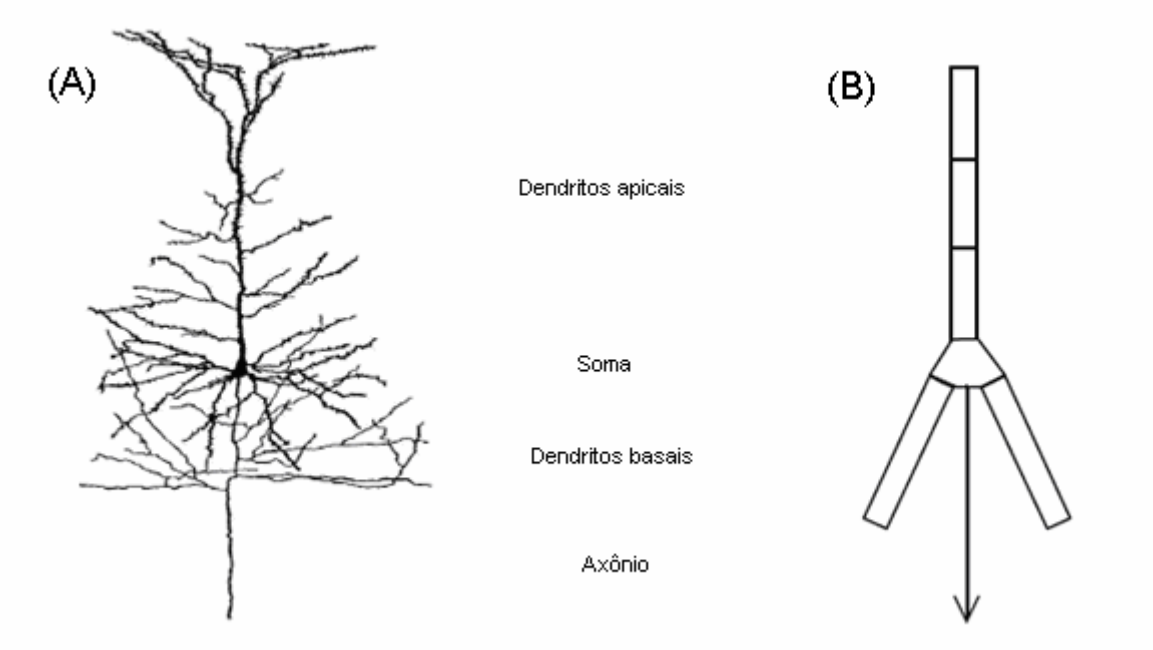


Figura 11. (A) Uma célula piramidal, com dendritos, soma (corpo celular), e axônio. (B) Um modelo compartimental da mesma célula. Figura adaptada de Bower e Beeman (1998).

Cada compartimento é modelado como um elemento isopotencial cujas propriedades elétricas são representadas por um circuito elétrico equivalente como o do modelo original de Hodgkin-Huxley (Figura 12).

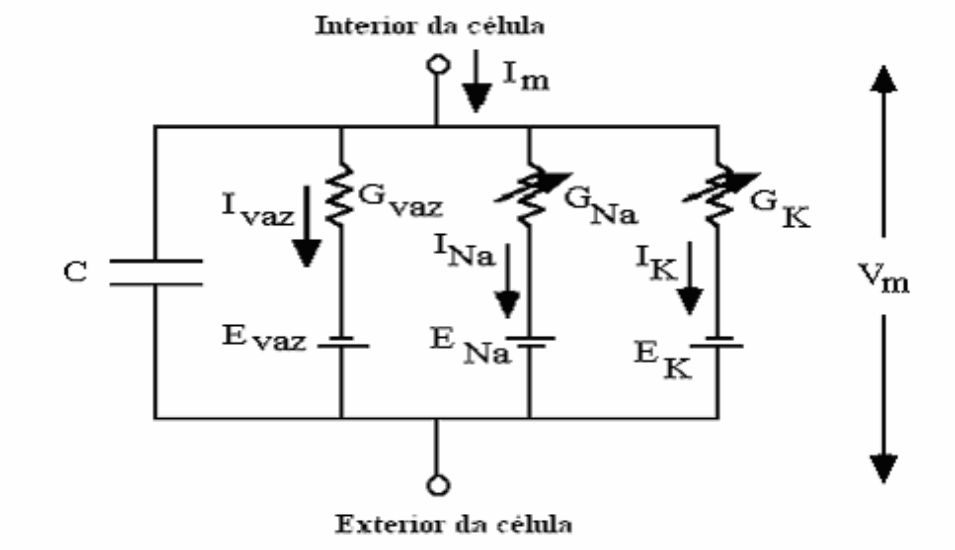


Figura 12. Circuito elétrico equivalente para o modelo de Hodgkin-Huxley. Figura adaptada de Bower e Beeman (1998).

No caso do modelo original de Hodgkin-Huxley (Figura 11) o compartimento possui três canais iônicos, de potássio, de sódio e de vazamento ou fuga (*leakage*), cujas correntes são dadas respectivamente por

$$I_K = G_K\left(V - E_K\right) = \overline{g}_K n^4 \left(V - E_K\right), \tag{1}$$

$$I_{Na} = G_{Na}\left(V - E_{Na}\right) = \overline{g}_{Na} m^3 h\left(V - E_{Na}\right), \tag{2}$$

$$I_{vaz} = G_{vaz}\left(V - E_i\right) = \overline{g}_i\left(V - E_i\right). \tag{3}$$

Nestas equações, G_i representa a condutância do canal de tipo i, E_i representa o potencial de reversão do canal de tipo i, $\overline{g}_i$ representa a condutância máxima do canal de tipo i, n, m e h são as variáveis de ativação e inativação dos canais de potássio e de sódio (dependentes da voltagem e do tempo) e V é a voltagem através da membrana, medida em relação à voltagem de repouso (Bower e Beeman, 1998; Koch e Segev, 1998).

A variação desta voltagem no tempo é determinada resolvendo-se o sistema de quatro equações acopladas:

$$C\frac{dV}{dt} = -\overline{g}_K n^4\left(V - E_K\right) - \overline{g}_{Na} m^3 h\left(V - E_{Na}\right) - \overline{g}_V\left(V - E_V\right) + I_{ext}, \tag{4}$$

$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V)n, \tag{5}$$

$$\frac{dm}{dt} = \alpha_m(V)(1-m) - \beta_m(V)m, \tag{6}$$

$$\frac{dh}{dt} = \alpha_h(V)(1-h) - \beta_h(V)h. \tag{7}$$

Nestas equações, I_{ext} representa uma possível corrente externa injetada no compartimento. As funções α e β, determinadas por Hodgkin e Huxley a partir do ajuste entre as equações do modelo e os seus dados experimentais, são:

$$\alpha_n(V) = 0{,}01\frac{10-V}{e^{(10-V)/10}-1} \qquad \text{e} \qquad \beta_n(V) = 0{,}125e^{-V/80}, \tag{8}$$

$$\alpha_m(V) = 0{,}1\frac{25-V}{e^{(25-V)/10}-1} \qquad \text{e} \quad \beta_m(V) = 4e^{-V/18}, \tag{9}$$

$$\alpha_h(V) = 0{,}07e^{-V/20} \qquad \text{e} \qquad \beta_h(V) = \frac{1}{e^{(30-V)/10}+1}. \tag{10}$$

Quando um neurônio é modelado por vários compartimentos acoplados por resistências, a voltagem de cada compartimento é modelada por equações como as acima. Por exemplo, a equação para a voltagem do *j*-ésimo compartimento seria:

$$\begin{aligned} C_j\frac{dV_j}{dt} &= g_{j-1,j}V_{j-1} - \left(\bar{g}_V + \bar{g}_K n^4(V_j,t) + \bar{g}_{Na}m^3(V_j,t)h(V_j,t) + g_{\sin_j}s_j(t) + g_{j-1,j} + g_{j,j+1}\right)V_j \\ &\quad + g_{j,j+1}V_{j+1} + \bar{g}_V E_V + \bar{g}_K n^4(V_j,t)E_K + \bar{g}_{Na}m^3(V_j,t)h(V_j,t)E_{Na} + g_{\sin_j}s_j(t)E_{\sin_j} + I_{ext_j}. \end{aligned} \tag{11}$$

Nesta equação, $g_{j\text{-}1,j}$ indica o valor da condutância axial entre os compartimentos $j-1$ e j e a corrente sináptica que o compartimento recebe é dada por

$$i_{\sin_j} = g_{\sin_j}s_j(t)\left(V_j - E_{\sin_j}\right). \tag{12}$$

Em geral, a função $s_j(t)$ é modelada por uma função chamada de função alfa (Bower e Beeman, 1998; Koch e Segev, 1998):

$$s_j(t) = \frac{t}{\tau_j}e^{-t/\tau_j}, \tag{13}$$

onde τ_j é uma constante característica do compartimento *j*. Cada compartimento *j* em que a célula é subdivida obedece a uma equação diferencial como a (11). Desta forma, uma célula que é subdividida em *N* compartimentos é descrita por um sistema de *N* equações diferenciais de primeira ordem acopladas. Esse sistema pode ser escrito na forma matricial como,

$$\dot{V} = A\vec{V} + \vec{b}, \tag{14}$$

onde $\vec{V}$ é um vetor coluna cujas componentes são as voltagens dos *N* compartimentos, $\dot{V}$ é a derivada temporal de $\vec{V}$, *A* é a matriz dos coeficientes que multiplicam a voltagem de um compartimento e de seus dois vizinhos e $\vec{b}$ é um vetor coluna contendo os termos que envolvem os potenciais de reversão e a corrente injetada.

O neuro-simulador GENESIS (Bower e Beeman, 1998) foi construído para permitir a resolução numérica de um sistema de equações como as acima para diferentes modelos de neurônios (com diferentes números de compartimentos e canais iônicos) e arquiteturas de redes neurais, alterando-se apenas alguns parâmetros ou opções, por exemplo, o método numérico usado para resolver as equações e o tamanho do passo de integração. Além disso, ele possui uma interface gráfica com o usuário para a visualização dos objetos modelados e dos resultados das simulações, por exemplo, a visualização dos valores dos potenciais de cada compartimento de um neurônio ou do soma de cada neurônio em uma rede.

O PGENESIS (Bower e Beeman, 1998; Hood, 2002) possui estrutura para paralelizar as simulações feitas no GENESIS e executá-las em cluster de computadores, reduzindo e otimizando o tempo de simulação para o caso de simulações em larga-escala, seja de um único neurônio composto por milhares de compartimentos com milhares de canais iônicos (por exemplo, o modelo de uma célula de Purkinje do cerebelo construído por De Schutter e Bower (1994) com o GENESIS (Figura 13)

possui 4550 compartimentos e 8021 canais iônicos), seja de uma rede composta por milhares de neurônios modelados por dezenas de compartimentos (por exemplo, o modelo da área cortical primária (V1) construído por Oliveira e Roque (2006) possui 59821 neurônios com um número variável de compartimentos entre 6 e 8).

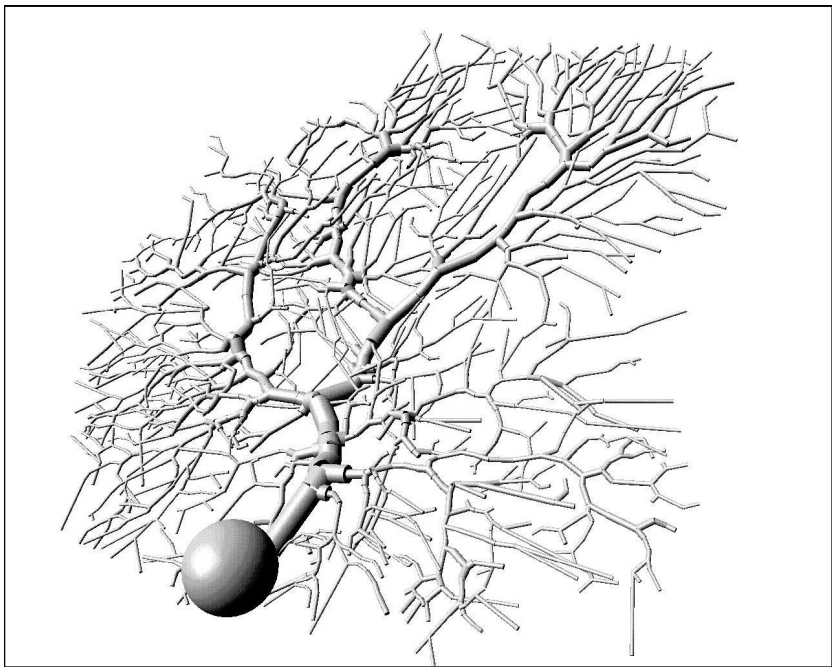


Figura 13 - Um modelo multi-compartimental detalhado de uma célula de Purkinje do cerebelo, criado com o GENESIS por De Schutter e Bower (1994). Figura adaptada de Bower e Beeman (1998).

Para situações como a desses modelos, o uso do PGENESIS pode fazer com que uma simulação que levaria dias implementada de forma seqüencial possa ser executada em algumas horas, dependendo do grau de paralelização que for possível alcançar.

3 Materiais e Métodos

A seguir é apresentado o equipamento utilizado neste trabalho e os métodos a serem utilizados

3.1 Materiais

O material necessário para este projeto foi conseguido junto à FAPESP (Processo Nº 03/12657-2) pelo orientador deste projeto e se encontra no Laboratório SisNe (Sistemas Neurais). Ele consiste de:

- 1 computador Dual Xeon 2.66 Ghz , 2x72 Gb SCSI com 2GB ram;
- 9 computadores Dual Xeon 2.66 Ghz, 40 Gb IDE com 2Gb Ram;
- 1 Switch Gigabit;
- 2 no-breaks de 2.2Kva;
- 1 módulo kvm de 16 portas;
- 1 Rack padrão 42 U.

Uma foto do cluster montado no laboratorio está dada na Figura 14.

Dos materiais listados acima, 7 computadores Dual Xeon com chipset E7501 foram utilizados neste estudo.

Figura 14. Foto da parte frontal do cluster

3.2 Tecnologia do equipamento

Os computadores são do modelo X5DPR-iG2 que é baseado no chipset E7501 da Intel, um chipset de alto desempenho projetado para processamento dual. A Figura 15 mostra um diagrama de bloco do chipset E7501(Data book Intel E7501, 2003).

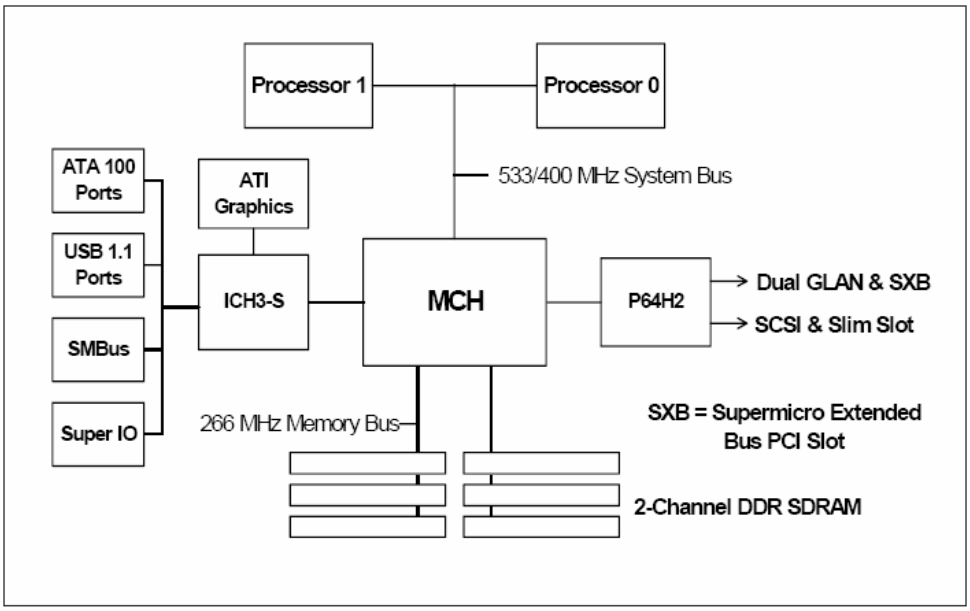


Figura 15 – Diagrama do chipset E7501

O chipset E7501 consiste de quatro componentes principais: o hub do controlador da memória (MCH), o iCubo 3 do controlador de I/O (ICH3), o hub 64-bit 2.0 de PCI-X (P64H2) e o adaptador host channel 82808AA (VxB).

Esse equipamento possui a tecnologia SMP (ver Seção 2.1). Isto significa que, ao contrário do sistema com um processador, um sistema SMP pode ter mais de um processo rodando ao mesmo tempo, neste caso dois processadores pentium Xeon de 2,66Ghz com tecnologia *hyperthreading* o que permite simular em um único processador físico dois processadores lógicos. Cada processador lógico recebe seu próprio controlador de interrupção programável (APIC) e conjunto de registradores.

Os outros recursos do processador físico, tais como, cache de memória, unidade de execução, unidade lógica e aritmética, unidade de ponto flutuante e barramentos, são compartilhados entre os processadores lógicos.

Em termos de software, isso significa que o sistema operacional pode enviar tarefas para os processadores lógicos como se estivesse enviando para processadores físicos em um sistema de multiprocessamento.

3.3 Estrutura Física do Cluster

Baseado nas definições apresentadas ate o momento pode-se classificar a configuração adotada para este equipamento como um cluster de computadores do tipo Beowulf com sistema de memória compartilhada distribuída, uma vez que cada máquina possui um sistema de memória compartilhada (SMP, tipo MIMD, fortemente acoplado) e que foram agrupadas para formar um tipo especifico de DSM (tipo MIMD, fracamente acoplada). O cluster é homogêneo já que todos os computadores utilizados possuem o mesmo hardware.

Para a utilização do cluster sem a necessidade de se estar fisicamente no laboratório para usar o console do equipamento, foi montado uma configuração de rede especial utilizando um equipamento do laboratório com endereço válido na internet para servir de ponte para a rede do cluster. Este computador possui um *firewall* configurado para permitir somente os acessos encaminhados para os serviços SSH e VNC, e estes serviços utilizam senhas com caracteres especiais para dificultar invasões.

Através de um repetidor VNC é possível acessar o console do mestre do cluster (ou qualquer nó) diretamente, tendo-se a sensação de estar em frente ao próprio equipamento, com todos os recursos disponíveis, inclusive a interface gráfica (neste caso Gnome).

Este cenário é mostrado na imagem a seguir (Figura 16). Apesar dessa configuração não influenciar diretamente o trabalho desenvolvido, sua configuração facilitou em muito um seguimento continuo das simulações.

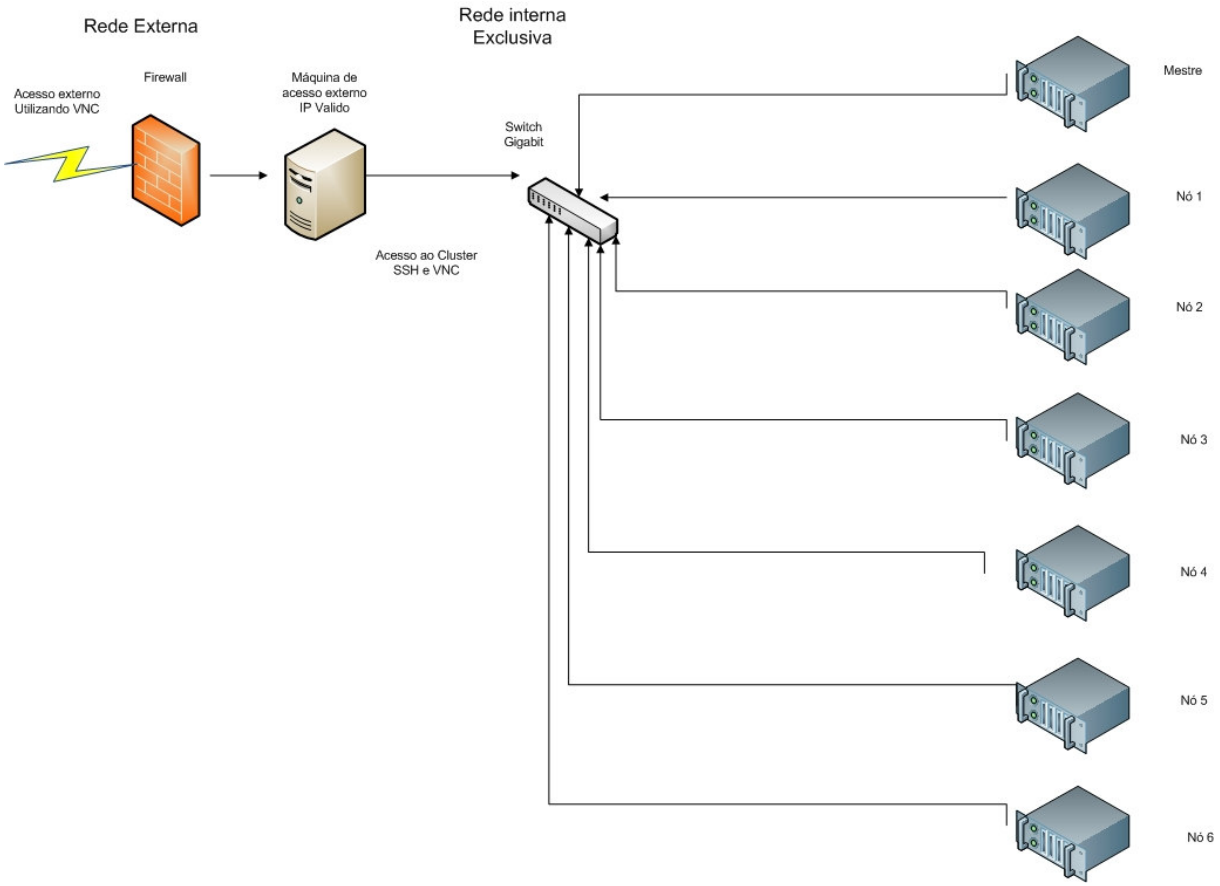


Figura 16 - Esquema do Cluster com Acesso externo.

3.4 Estrutura Lógica do Cluster

Como este trabalho foi desenvolvido sobre uma plataforma fixa de hardware, que possui limitações com relação aos experimentos possíveis de serem executados, estão na configuração lógica do cluster os principais pontos de análise e testes. Portanto, apresenta-se abaixo uma breve descrição sobre os componentes lógicos do cluster.

3.5 Componentes do Cluster

O sistema operacional instalado no cluster é o Linux (Campos, 2006). Neste trabalho foram escolhidas a distribuições Fedora 5 e Gentoo 5, devido a sua boa difusão e aceitação no meio acadêmico e ao fato de serem livres (*freeware*). Outras distribuições foram usadas ao longo do trabalho até a adoção final da Fedora 5 e do Gentoo 5. Por exemplo, os experimentos com o kernel 2.4 foram executados sob a distribuição Conectiva 9.

Para a comunicação entre os nós foram testadas as bibliotecas PVM e MPI (ver Seção 1). Como o conceito básico dessas bibliotecas é realizar computação paralela utilizando um conjunto de computadores totalmente heterogêneos, o protocolo TCP/IP é o canal de comunicação entre os nós através de uma rede física do tipo ethernet.

Devido a esta busca pela harmonia em um ambiente heterogêneo os pacotes que são enviados e recebidos pelos nós sofrem um empacotamento, o que, com certeza, causa uma sobrecarga no processamento e pode influenciar no ganho de processamento esperado com a paralelização da tarefa. Para que a comunicação entre os nós seja possível, alem do PVM ou MPI, é necessária a existência de uma estrutura de comunicação mínima entre os computadores que, no caso do cluster deste trabalho, consiste fisicamente de uma rede do tipo ethernet com um *switch* Gigabit (como descrito acima na Seção 3.3) e de um canal lógico aberto de comunicação entre as maquinas. Isto pode ser obtido pela configuração dos serviços RSH (*remote shell protocol*) ou SSH (*secure shell protocol*) e, por padrão, o PVM e o MPI utilizam o RSH, sendo necessária ainda a configuração para o acesso sem senha entre os membros do cluster (Geist,1994). (Ver Anexo I)

3.6 Avaliação do Desempenho do Cluster

Por se tratar de um sistema operacional aberto, o Linux possui várias ferramentas de *benchmarking* que podem ser utilizadas para analisar como se comporta o sistema durante uma simulação e medir qual a sua capacidade de processamento. Após a realização de uma pesquisa sobre as ferramentas disponíveis para análise de desempenho, diante da grande aceitação do Linpack esta foi a ferramenta escolhida.

O Linpack (Dongarra et al., 2001; Dongarra, 2005) é um dos *benchmarks* mais populares na avaliação do desempenho de computadores em aplicações científicas e em

engenharia, inclusive sendo utilizado nos testes das 500 máquinas mais rápidas existentes (Meuer et al., 2004). O Linpack é um pacote genérico de rotinas para a resolução numérica de sistemas lineares de *n*x*n* equações matriciais densas, $A\vec{x} = \vec{b}$ (Dongarra et al., 1989). Ele avalia o desempenho do sistema em executar operações em ponto flutuante e o resultado da avaliação é reportado em MFLOPS, GFLOPS ou TFLOPS.

Para se trabalhar com análise de desempenho, alguns termos e definições precisam ser explicitados. Eles estão listados a seguir:

***Speedup*:** É o aumento de velocidade ganho com a paralelização da tarefa, em relação à sua execução com um único processador. O *speedup* pode ser equacionado da seguinte forma (Colombet e Desbat, 1998):

$$S = \frac{T_1}{T_P}, \tag{15}$$

onde S é o coeficiente de *speedup*, T_1 é o tempo gasto para executar a tarefa usando um único processador e T_P é o tempo gasto com P processadores usando programação paralela. Em uma situação ideal o valor do coeficiente de *speedup* deveria ser igual a P, ou seja, o gráfico de S contra P deveria ser uma linha reta com inclinação unitária, mas em situações reais a curva *S versus P* costuma ficar abaixo da reta teórica (Colombet e Desbat, 1998).

Eficiência de um Sistema: É definida como sendo a razão entre o *speedup* conseguido em um sistema de P multiprocessadores e o número P de processadores do sistema (Colombet e Desbat, 1998):

$$E = \frac{S}{P}. \tag{16}$$

A eficiência de um sistema dá a fração do número de processadores que está efetivamente sendo usada para a execução de um programa paralelo, ou a taxa de

utilização de cada processador durante a execução da tarefa (Colombet e Desbat, 1998). Seu valor teórico seria constante e igual 1, mas na prática os valores de *E* costumam ficar abaixo disso. Se o valor de *E*, embora abaixo de 1, se mantém constante com o aumento do número *P* de processadores o sistema é chamado de *escalável* (Colombet e Desbat, 1998; Pitanga, 2004).

Granulação: Também chamada de *nível de paralelismo*, caracteriza o tamanho das unidades de trabalho submetidas aos processadores. Esta é uma definição muito importante na computação paralela, visto que está intimamente ligada ao tipo de plataforma (o porte e a quantidade de processadores) à qual se aplica o paralelismo (Kirner,1991). De maneira geral, a granulação costuma ser dividida em três níveis: fina, média e grossa (Kirner, 1991; Almasi, 1994). Plataformas com granulação grossa possuem, em geral, poucos processadores grandes e complexos e plataformas com granulação fina, ao contrário, possuem um grande número de processadores pequenos e simples. Plataformas com granulação média situam-se em um patamar entre as duas anteriores.

O PGENESIS pode ser classificado como um sistema de granulação média ou grossa, dependendo da forma como a simulação é construída. A metodologia usada para se avaliar o desempenho do cluster na execução de simulações com o PGENESIS foi similar à adotada por Jahnke et al. (1997), onde o tempo é o fator de análise primordial (Jahnke et. al., 1998). Nesta análise, os conceitos de *speedup* e eficiência foram amplamente usados.

3.7 O Modelo Usado na Avaliação

A simulação base para este estudo está disponível na documentação do GENESIS sob o nome de *Orient-tut* e está descrita no livro *The Book of GENESIS* (Bower e Beeman, 1998) em sua versão serial, 1slice.g, e em sua versão paralela, demo.g. Ela consiste de um

modelo simplificado para o fenômeno de seletividade a orientação no córtex visual primário (Oliveira, 2006). A simulação tem como finalidade a aprendizagem da modelagem de redes em larga-escala no GENESIS e modela um conjunto de neurônios do córtex visual primário (V1) sensíveis a diferentes orientações que recebem estímulos de uma camada de receptores que simula a retina. O estágio intermediário entre a retina e V1, o núcleo lateral geniculado do tálamo, não é modelado.

Os neurônios corticais do modelo são sensíveis a apenas duas orientações, uma vertical e a outra horizontal. Nessa versão simplificada, as células da retina são apenas geradores aleatórios de estímulos que transmitem seus sinais às células do córtex. Já os neurônios corticais são modelados como neurônios de Hodgkin-Huxley com três tipos de canais iônicos: de sódio, de potássio e de vazamento. Os neurônios corticais estão organizados em duas camadas modeladas por redes quadradas de *N*x*N* células com um valor de *N* que pode ser definido pelo modelador.

A maneira como a simulação foi paralelizada está descrita no livro *The Book of GENESIS* (Bower e Beeman, 1998). As redes que modelam a retina e as duas camadas do córtex são divididas em fatias ao longo do eixo horizontal, com as células de cada fatia sendo processadas por um dos nós do cluster (Fig. 17).

Esse modelo, apesar de simples, é um protótipo dos tipos de modelos em larga-escala de sistemas sensoriais desenvolvidos pelo orientador deste trabalho e seus alunos no Laboratório SisNe. Além disso, ele está completamente descrito e documentado no *The Book of GENESIS* (Bower e Beeman, 1998) e é amplamente conhecido pela comunidade internacional de modeladores em neurociência computacional, o que permite que os resultados aqui descritos possam ser reproduzidos por pesquisadores em outras partes do mundo. Por estes motivos, esse modelo do córtex visual primário foi escolhido como base deste trabalho.

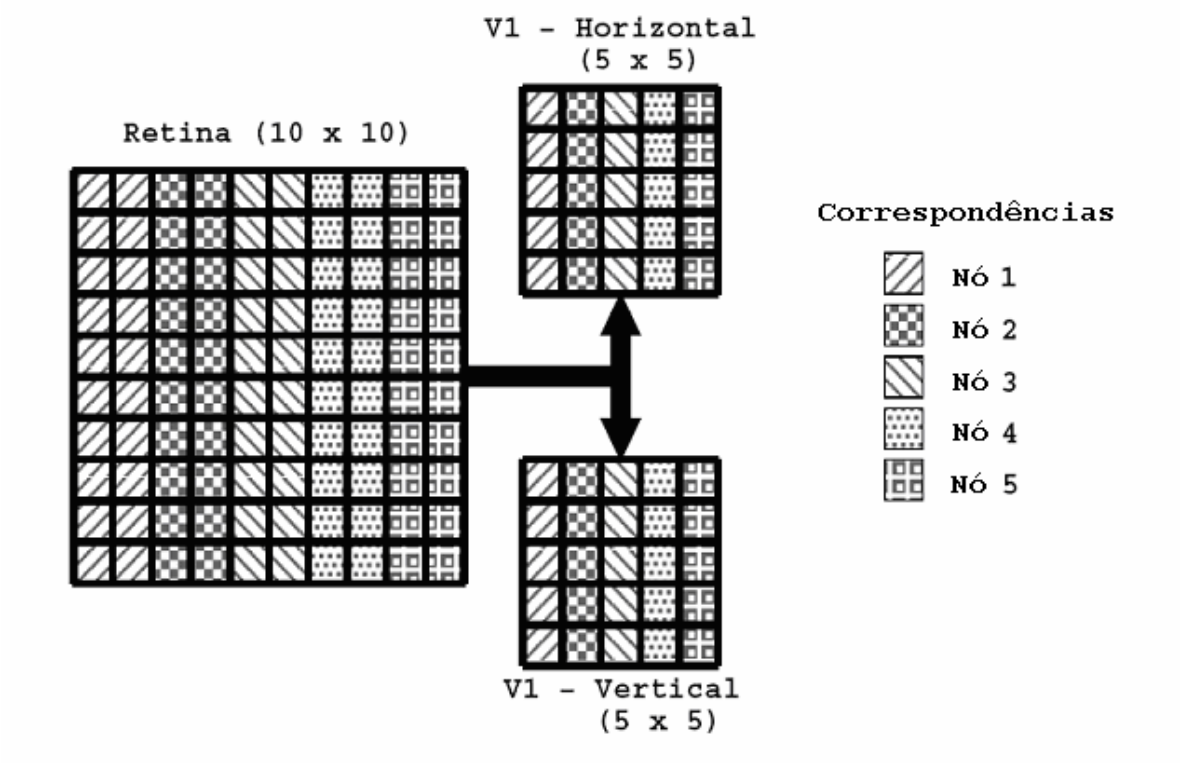


Fig. 17 – Esquema de paralelização da rede simulada. No caso da figura, a retina é formada por uma rede de 10x10 neurônios e cada uma das camadas corticais é formada por uma rede de 5x5 neurônios (esta configuração é representada, de forma abreviada, por "10x10, 5x5" e essa forma de indicar a arquitetura de uma simulação será usada no restante deste trabalho). As camadas são decompostas em fatias e as células de cada fatia são processadas por nós diferentes. No caso da figura, são usados cinco nós. Figura adaptada de Bower e Beeman (1998).

3.8 Forma de Análise dos Resultados

A estrutura do modelo de simulação usado neste trabalho permite sua expansibilidade pela alteração do parâmetro *N* que controla o número de neurônios nas camadas de V1, dando uma grande maleabilidade durante as simulações de análise de desempenho.

O esquema dos estudos feitos com o modelo é o seguinte: para um valor inicial de *N*, otimiza-se a simulação no PGENESIS e obtém-se o tempo de simulação para este caso. Posteriormente, aumenta-se o valor de *N* (o que implica em mais neurônios e sinapses simuladas) e uma nova avaliação do tempo gasto para simular uma fatia de tempo é feita. Desta forma, é possível levantar um gráfico de desempenho contendo,

num eixo, o tempo gasto para simular uma fatia de tempo e, no outro eixo, o tamanho do modelo. Esse procedimento permite que se faça um estudo sistemático do desempenho do conjunto cluster/PGENESIS/Linux na simulação do modelo, permitindo a identificação dos parâmetros dos componentes do conjunto (tanto de hardware como de software) a cujos valores uma simulação é mais sensível.

Uma vez levantados os valores "normais" de desempenho para os diferentes tamanhos de modelo considerados, teve inicio a fase de otimização dos componentes do cluster. Essa fase consistiu na realização de ajustes nos diferentes itens descritos nas seções anteriores e no levantamento de novos valores de desempenho para serem comparados com os do caso normal.

Na fase de comparação e otimização, foi adotada a análise por *speedup* e por eficiência para a classificação dos conjuntos usados nas simulações. Este método é bastante utilizado quando se pretende analisar o ganho de desempenho dentro de um mesmo conjunto (Pitanga 2004).

4 Apresentação e análise de resultados

4.1 Análise do Hardware

Memória

Em computação, o assunto *desempenho* logo é associado com a questão *memória*. Em geral, assume-se que quanto maior a quantidade de memória disponível mais rápido será o sistema. Esta conclusão se deve ao fato de a memória RAM trabalhar na velocidade do barramento de comunicação do computador e de ser algumas vezes mais rápida que o acesso a discos rígidos ou removíveis, porém, numa primeira análise do hardware ficou claro que nem sempre mais memória significa maior desempenho (Fig. 18).

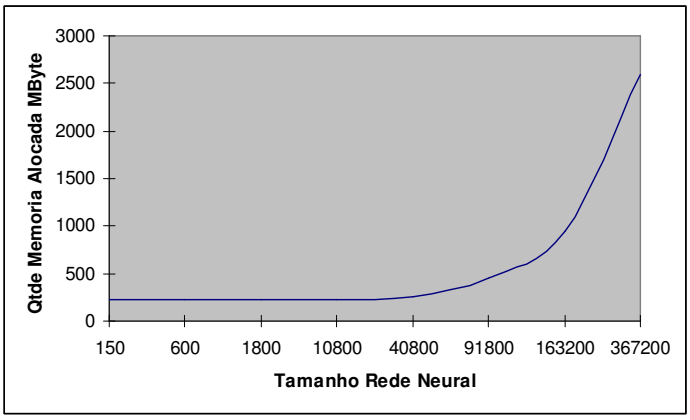


Figura 18 – Gráfico de Alocação de Memória X Tamanho da rede neural

O gráfico da Fig. 18 apresenta a relação obtida entre a quantidade de memória alocada pela aplicação, neste caso uma simulação no GENESIS do modelo do sistema visual descrito na Seção 3.7 usando apenas um processador com 2 GB de memória RAM, e o tamanho da rede neural utilizada na simulação (o número de neurônios,

contando-se tanto os neurônios de V1 como os geradores de sinais da retina). Enquanto o tamanho da rede se manteve abaixo de 40.000 neurônios não houve alteração no uso de memória, mas a partir daí houve um enorme aumento no uso de memória. Nesse processo explosivo de aumento de alocação de memória, toda a memória RAM disponível no equipamento acabou sendo utilizada e, para que fosse possível rodar uma simulação com uma rede de mais de 300.000 neurônios, a simulação teve que consumir espaço em disco na forma de memória de swap. O uso da memória swap gerou uma grande perda de desempenho, como pode ser visto no gráfico da Fig. 19 que dá o tempo gasto com a execução de uma simulação *versus* o tamanho da rede simulada.

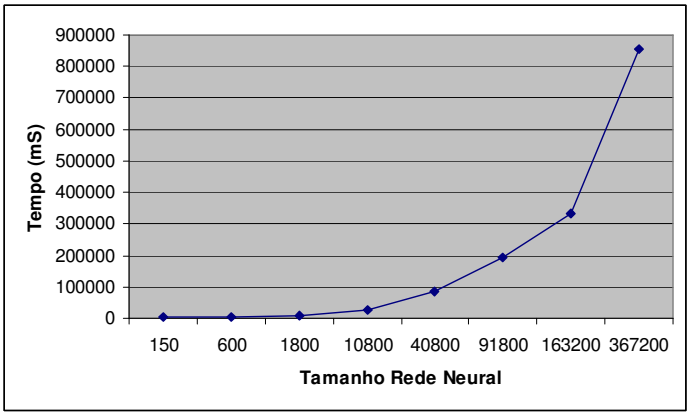


Figura 19 – Gráfico de Tempo x Tamanho da rede neural

Memória cache

Uma maneira de se melhorar o desempenho do sistema, ainda pensando-se em termos de memória, é aumentar a quantidade de memoria cache. A memória cache, no entanto, é uma memória cara e o seu incremento implica na troca do processador que contém a memoria cache.

A configuração da memória cache do processador para o modo *write-back* (Patterson e Hennessy, 2004) gerou um ganho de desempenho porque uma menor quantidade de informação necessita ser atualizada a cada ciclo. Para poder quantificar

este ganho, a simulação básica do tutorial *orient-tut2* do *The Book of GENESIS* (Bower e Beeman, 1998) foi executada nas duas situações, com cache no modo *write-back* e com cache no modo *write-through* (Patterson e Hennessy, 2004). Como pode ser visto na Tabela 1, o speedup conseguido com o modo *write-back* em relação ao modo *write-through* foi de aproximadamente 20, mostrando que para aplicações com alto uso de processamento a configuração no modo *write-back* da memória cache do processador é a melhor escolha.

Tabela 1 – Comparação do desempenho com memória cache nos modos *write-through* x *write-back*.

Máquina básica -Simulação *Orient-tut* 20x20 10x10 10x10 10 Slices			
Passos	Tempo write-through	Tempo write-back	Speedup
20000	2449300	124037	19,7465272
50000	6108703	307084	19,8926124

Hyperthreading (HT)

A utilização dos recursos incorporados pela tecnologia HT é muito proveitosa quando o assunto é desempenho. A Figura 20 mostra o speedup obtido no mesmo equipamento e rodando as mesmas simulações com o recurso HT habilitado e desabilitado. O uso do HT gerou um ganho que variou de aproximadamente 7% para as simulações maiores até quase 50% para as simulações menores, tendo como media aritmética das simulações um speedup de 1,25 (Figura 21).

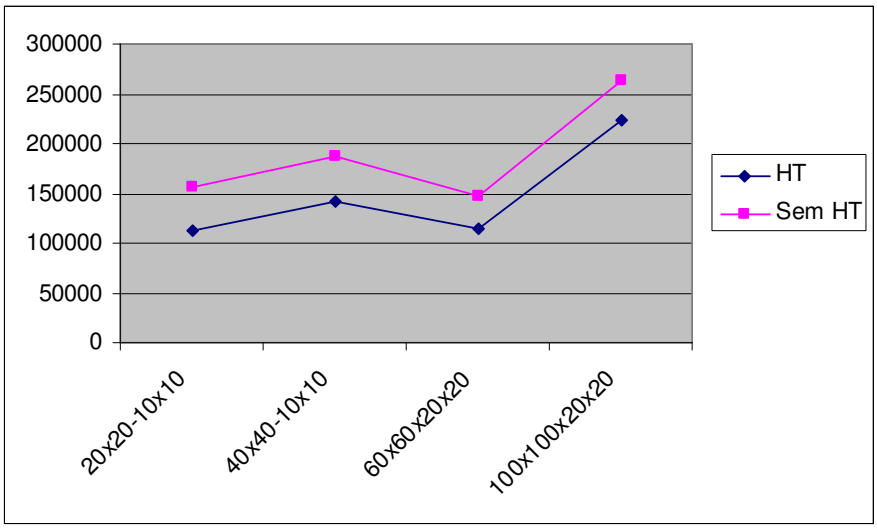


Figura 20 – Ganho com uso de HT (eixo horizontal: configuração da rede neural; eixo vertical: tempo em ms).

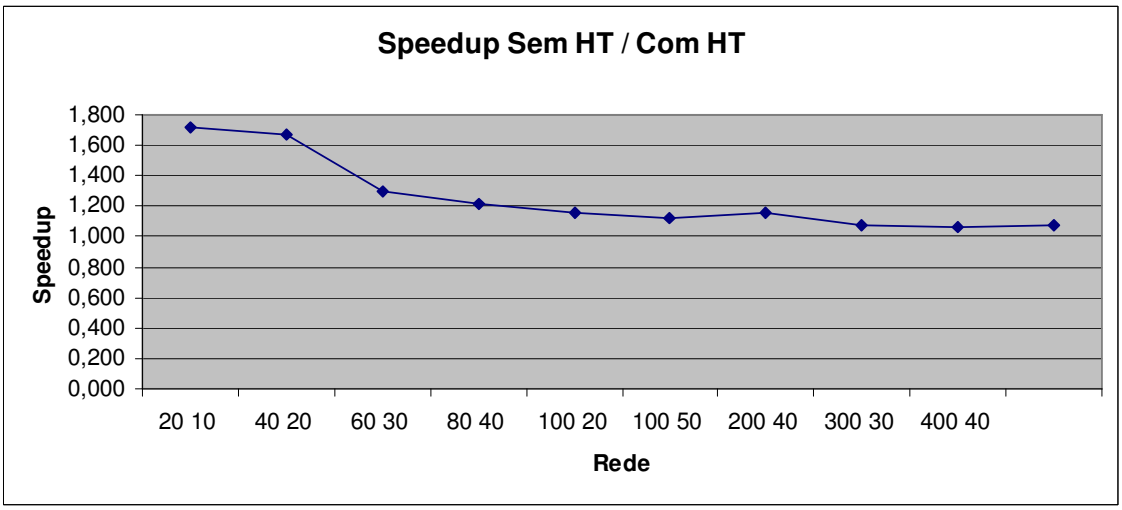


Figura 21 – Speedup com e sem HT. Entende-se usando como exemplo 60 20 10 como retina 60x60, V1_Vert 20x20, V1_Horiz 20x20 divididos em 10 Slices, assim por diante.

Velocidade de comunicação entre os nós

Todas as boas práticas descritas no *The Book of GENESIS* (Bower e Beeman, 1998) para uma simulação mais rápida foram aplicadas neste trabalho, inclusive a de minimizar a comunicação entre os nós durante a simulação. A comunicação, portanto, não foi o gargalho nos casos de estudo, mas mesmo assim a mudança de um switch de 100mbits/s para um de 1000mbits/s forneceu um aumento de desempenho em torno de 30%, para diferentes tamanhos da rede neural simulada.

4.2 Análise do Software

Kernel 2.4 x Kernel 2.6

Para todos os experimentos realizados, o kernel 2.6 apresentou, em média, 9% de ganho de performance em relação ao kernel 2.4 (Figura 22).

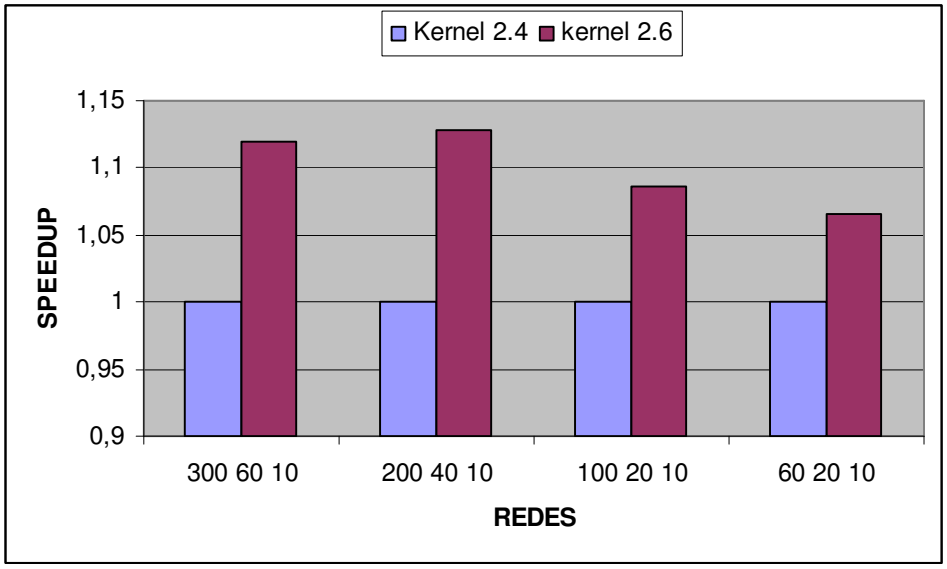


Figura 22 – Speedup com os *kernels* 2.4 e 2.6 Entende-se usando como exemplo 60 20 10 como retina 60x60, V1_Vert 20x20, V1_Horiz 20x20 divididos em 10 Slices, assim por diante.

Apesar de ambos os *kernels* possuírem suporte para a tecnologia HT, o kernel 2.6 possui um sistema de agendamento de processos mais elaborado e ágil, alem de já possuir suportes para novas tecnologias utilizadas no cluster, tais como processador Xeon e placa de rede gigabit. Em redes maiores, onde a tendência de comunicação entre os processos da simulação é maior, o ganho (speedup) tende a aumentar. Nas simulações realizadas este valor chegou a 16% (Figura 23).

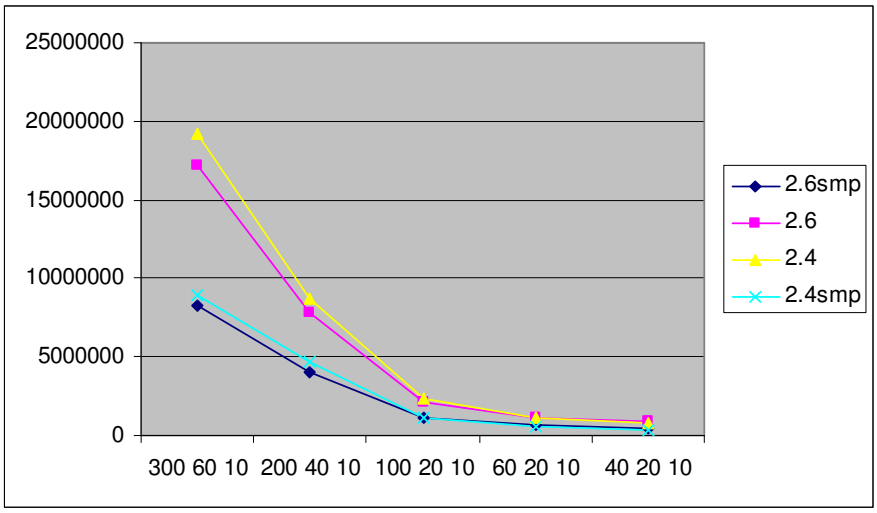


Figura 23 – Kernel 2.4 e 2.6: Gráfico tempo x rede neural Entende-se usando como exemplo 60 20 10 como retina 60x60, V1_Vert 20x20, V1_Horiz 20x20 divididos em 10 Slices, assim por diante.

SSH X RSH

A principal diferença com relação ao uso do SSH (*secure remote shell*) em comparação com o RSH (*remote shell*) é que toda a comunicação entre cliente/servidor é feita de forma encriptada usando-se chaves público-privadas para criptografia, garantindo uma transferência segura de dados.

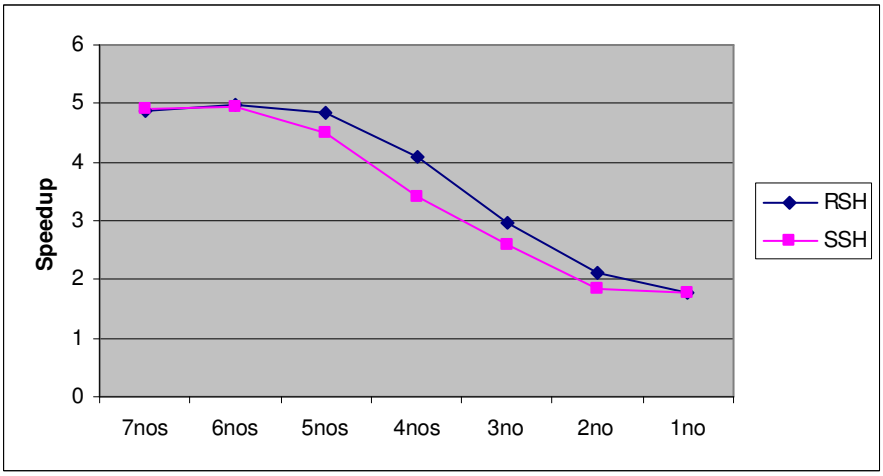


Figura 24 – Gráfico speedup RSH - SSH

O uso de comunicação RSH gerou, em média, um ganho de 10% em relação ao uso do SSH (Figura 24). Este resultado de maior desempenho com a comunicação RSH era esperado, porque a comunicação SSH acrescenta ao processo de envio da mensagem a etapa de criptografia e na etapa de recebimento da mensagem a tarefa de descriptografia, aumentando assim o tempo de comunicação.

Se o ambiente paralelo esta usando dispositivos de comunicação de uma rede aberta o uso do SSH é recomendado, mas no caso da maioria dos clusters que têm seu próprio dispositivo de comunicação que fica atrás de um firewall, o uso do RSH é recomendável por motivos de desempenho.

PVM X MPI (MPICH/LAM)

Para a simulação de rede neural usada neste trabalho, os resultados obtidos com o uso de PVM foram bem superiores aos obtidos com o uso de MPI, nas suas duas distribuições gratuitas testadas (a LAM, da Indiana University**,** e a MPICH, do Argonne National Laboratory/University of Chicago/MCS Division). O speedup conseguido com o uso de PVM variou em torno de 10 a 20.

Esta discrepância se deve ao fato das simulações usadas neste estudo serem originalmente desenhadas para uso com o PVM. Neste caso, portanto, o uso de MPI não se trata de uma opção e sim de uma adequação, isto é, as simulações podem ser adequadas para usar o MPI, mas sem a performance estimada.

Saindo do ambiente analisado e buscando na bibliografia outros casos de análises comparativas entre os sistemas PVM e MPI, encontram-se vários estudos[2] que parecem indicar que, de fato, não é sempre que um sistema com MPI possui desempenho superior a um sistema com PVM.

[2] Ver, por exemplo, a compilação de trabalhos levantada pelo Grupo de Pesquisa de Aplicações em Computação Paralela-GPACP, da UNESP de Rio Claro (GPACP, 2007).

Uso do Xodus

O uso da ferramenta de interface gráfica do GENESIS, Xodus (Bower e Beeman, 1998), é muito interessante para a aferição das simulações, além da óbvia vantagem do acompanhamento gráfico em tempo real do status das simulações.

No entanto, passada a etapa de análise das simulações e tendo início à etapa de coleta de dados esta ferramenta, outrora imprescindível, se torna um problema. Dados experimentais (Tabela 2 e Figura 25) mostram que a mesma simulação gravando o mesmo arquivo de resultados, mas sem a interface gráfica tem um speedup em torno de 4 em relação à original.

Tabela 2 – Análise do uso do Xodus

	Redes: Retina V1 Fatias				
Configuração da rede	400 40 10	300 60 10	200 40 10	60 20 10	20 10 5
Tempo de execução com o Xodus (mS)	90987654	35207338	11344676	6013531	1446346
Tempo de execução sem o Xodus (mS)	26765876	9849178	3114955	1467688	307084
Speedup na execução com 1 processador sem o Xodus	3,39939	3,57464	3,642	4,09728	4,70993

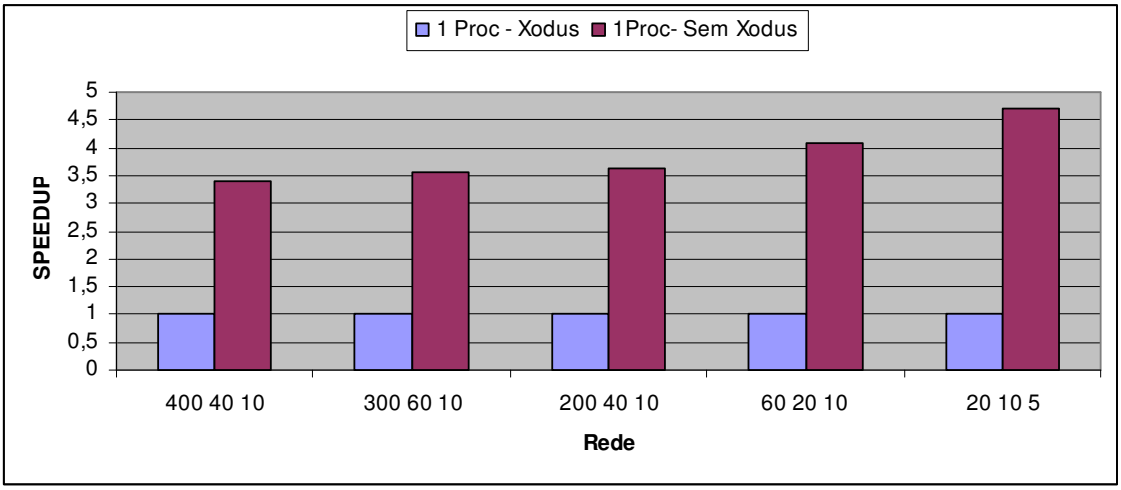


Figura 25 – Speedup na execução de simulações com diferentes configurações com e sem o uso do Xodus. Entende-se usando como exemplo 60 20 10 como retina 60x60, V1_Vert 20x20, V1_Horiz 20x20 divididos em 10 Slices, assim por diante.

4.3 Análise do desempenho do cluster rodando o PGENESIS

Nesta sessão é apresentada a análise do cluster propriamente dito, executando uma simulação em paralelo, com a apresentação e análise dos resultados obtidos com o acréscimo de mais computadores a maquina virtual.

Duas configurações distintas do cluster foram feitas para fins comparativos. Na primeira foi utilizado o sistema operacional Fedora Core 5 com kernel 2.6.20 pré-compilado (assim como as aplicações já instaladas, tais como, pvm, rsh, ssh, etc) e com a instalação padrão (alguns dispositivos secundários e softwares auxiliares foram instalados, tais como, openoffice, gnome, etc) e, na segunda, foi utilizado o sistema operacional Gentoo R5 com kernel 2.6.19 compilado no próprio equipamento (assim como as ferramentas necessárias ao cluster, tais como pvm, ssh, rsh, etc) e com uma instalação mínima (sem a interface gráfica e apenas com os dispositivos e aplicações realmente necessários instalados, tais como pvm, módulo de placa de rede, etc).

Inicialmente, serão apresentados e analisados os resultados obtidos com a primeira configuração. No Anexo I desta dissertação pode ser encontrado um manual que mostra, passo a passo, como montar um cluster utilizando a configuração mais simples com o Fedora Core 5.

Comparação SMP com 2 nós mono processados

Antes de se iniciar a análise do cluster com o acréscimo de maquinas, uma análise preliminar se fez necessária. Todos os equipamentos utilizados neste estudo são duo processados com tecnologia HT. Porém, com a compilação do kernel do linux de forma específica foi possível criar uma estação mono processada sem reconhecer a tecnologia HT e essa máquina foi utilizada como padrão para as comparações de speedup e eficiência do cluster.

As Figuras 26 e 27 mostram a comparação entre uma estação com os recursos SMP e HT habilitados no kernel e um cluster formado por duas estações utilizando o kernel citado acima, utilizando apenas um processador sem recurso HT.

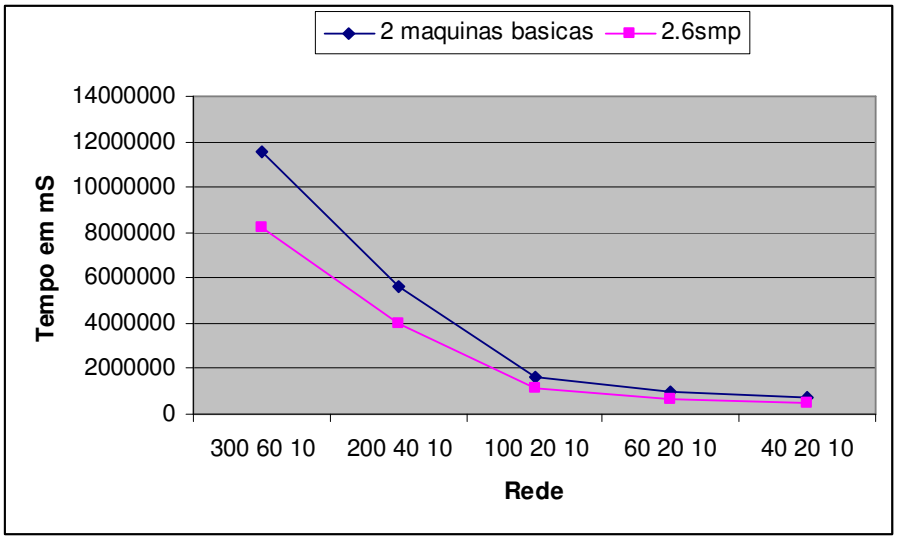


Figura 26 – Gráfico 2Mono X 1SMP - Tempo X Rede neural. Entende-se usando como exemplo 60 20 10 como retina 60x60, V1_Vert 20x20, V1_Horiz 20x20 divididos em 10 Slices, assim por diante.

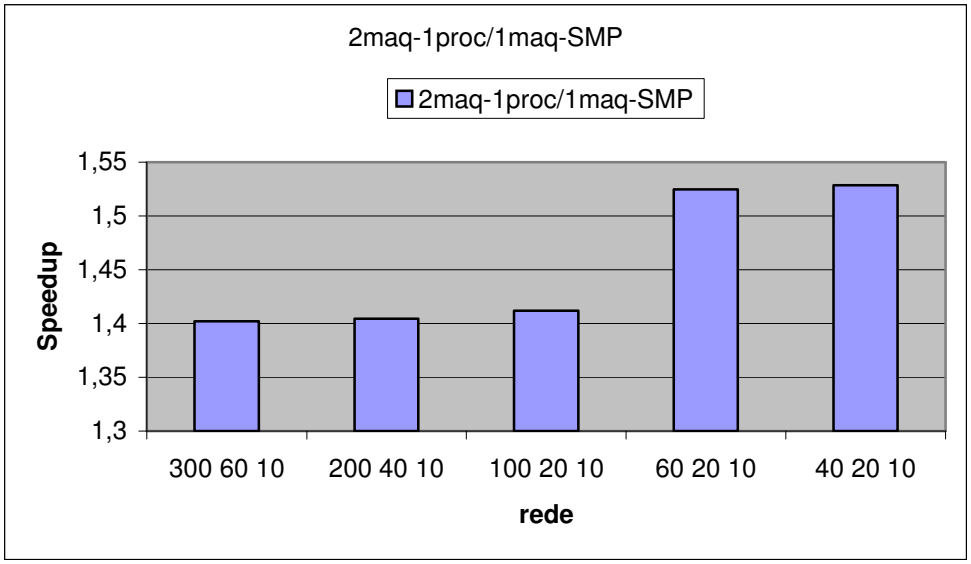


Figura 27 – Gráfico SPeedup 2Mono X 1 SMP. Entende-se usando como exemplo 60 20 10 como retina 60x60, V1_Vert 20x20, V1_Horiz 20x20 divididos em 10 Slices, assim por diante.

Uma única máquina utilizando as tecnologias SMP e HT é, em media 40%, mais rápida que duas máquinas usando um único processador. Em simulações com redes neurais pequenas este ganho chega a 50%.

Existem alguns motivos para que isto aconteça, os principais são:

- Tempo de troca de mensagens através da rede gigabit: a comunicação entre as máquinas mono processadas é mais lenta do que a realizada entre os processadores da máquina SMP que utilizam a velocidade de barramento para isso. A taxa máxima de transferência no barramento externo (processador - memória) das estações utilizadas é de 4,2 GB/s (6013-Manual, pág. 1-2), enquanto a taxa máxima de transferência obtida por um switch gigabit é de 1GB/s ou 2GB/s em modo Full-Duplex.
-O uso da tecnologia HT no equipamento SMP: como demonstrado anteriormente. o uso deste recurso fornece um ganho em torno de 7 a 50% em relação ao uso de um mono processador.

Cluster com 2 a 7 estações adicionadas à máquina virtual

Utilizando a simulação adotada como padrão (a mesma utilizada nas análises anteriores) chega-se aos dados mostrados Tabela 3, contendo resultados para diferentes arranjos de computadores. Todos utilizando o sistema operacional Fedora Core 5, exceto os que utilizaram o kernel 2.4 que foram configurados com o sistema operacional Conectiva 9.

Tabela 3 – Apresentação geral dos dados (Tempo de execução)

Tempo de Execução (mS)	Redes: Retina V1 Fatias				
Número de passos = 50000	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
Kernel 2.4 - Mono processado	19232254	8751889	2318268	1248337	876333
Kernel 2.6 - Mono processado	17190345	7758426	2134768	1171030	867730
2 máquinas mono processadas	11565374	5635473	1649101	973080	709559
2.4smp	8981128	4633020	1148784	557596	477909
2.6smp	8249178	4013306	1168129	638173	464230
2 máquinas fedora	6591006	3354652	960354	548318	468590
3 máquinas fedora	4276428	2046654	759479	398045	387368
4 máquinas fedora	4064475	2099612	557154,7	311894,9	238143
5 máquinas fedora	5235030	2454980	792456,2	482050,4	378109,1
6 máquinas fedora	3064247	1541961	590406,2	380491,9	381569,8
7 máquinas fedora	2055298	1035018	422444,3	317356,2	235016,3

Considerando a estação com o kernel 2.4 mono processado para análise de speedup e eficiência, chega-se aos dados mostrados na Tabela 4 e na Figura 28.

Tabela 4 – Apresentação geral dos dados (speedup)

	Speedup - Redes: Retina V1 Fatias				
	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
Kernel 2.4 - Mono processado	1	1	1	1	1
Kernel 2.6 - Mono processado	1,118782	1,12805	1,085958	1,066016	1,009914
2 máquinas mono processadas	1,662917	1,553	1,405777	1,282872	1,235039
2.4smp	2,141407	1,889025	2,018019	2,238784	1,833682
2.6smp	2,331415	2,180718	1,984599	1,956111	1,887713
2 máquinas fedora	2,917954	2,608881	2,413972	2,276666	1,870149
3 máquinas fedora	4,497271	4,276194	3,052445	3,136171	2,262275
4 máquinas fedora	4,731792	4,168337	4,160905	4,002429	3,679861
5 máquinas fedora	3,673762	3,564953	2,925421	2,58964	2,317672
6 máquinas fedora	6,27634	5,675816	3,926564	3,280851	2,296652
7 máquinas fedora	9,357404	8,455784	5,487749	3,933551	3,728818

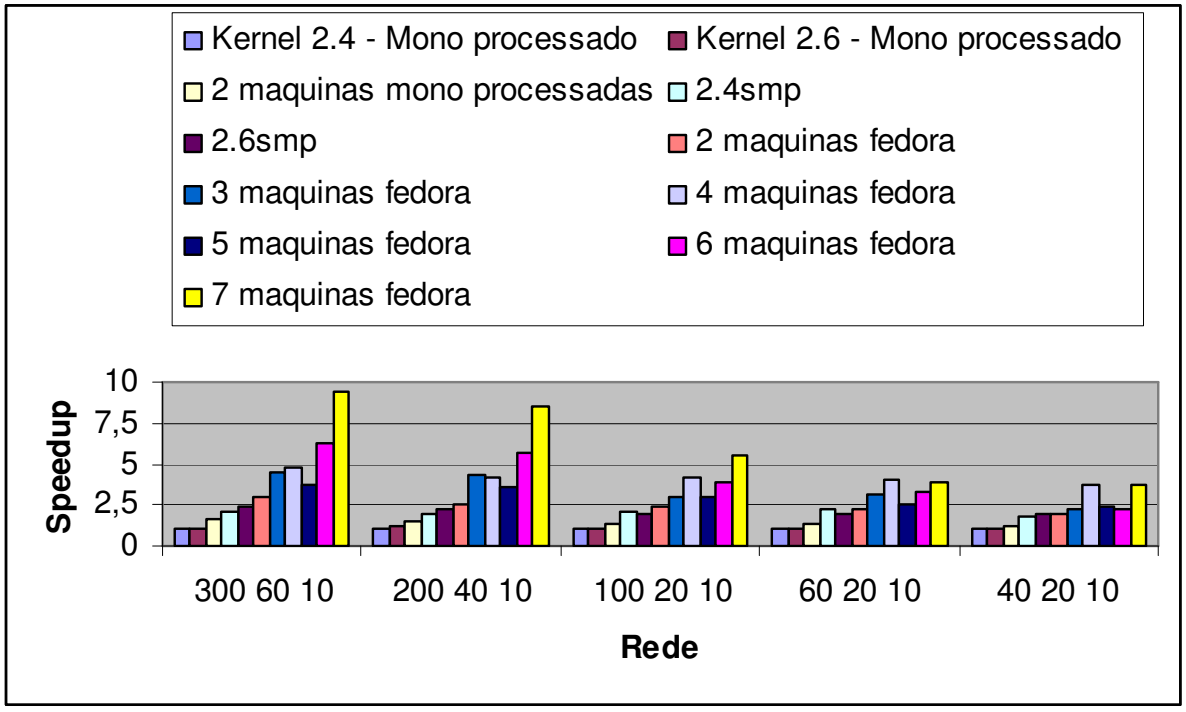


Figura 28 – Speedup para diferentes configurações do cluster. Entende-se usando como exemplo 60 20 10 como retina 60x60, V1_Vert 20x20, V1_Horiz 20x20 divididos em 10 Slices, assim por diante.

Tabela 5 – Apresentação geral dos dados (eficiência)

	Nº	Eficiência - Redes: Retina V1 Fatias				
	Proc	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
Kernel 2.4 - Mono processado	1	1	1	1	1	1
Kernel 2.6 - Mono processado	1	1,118782	1,12805	1,085958	1,066016	1,009914
2 maquinas mono processadas	2	0,831458	0,7765	0,702888	0,641436	0,617519
2.4smp	2	1,070704	0,944512	1,00901	1,119392	0,916841
2.6smp	2	1,165707	1,090359	0,9923	0,978055	0,943856
2 maquinas fedora	4	0,729489	0,65222	0,603493	0,569167	0,467537
3 maquinas fedora	6	0,749545	0,712699	0,508741	0,522695	0,377046
4 maquinas fedora	8	0,591474	0,521042	0,520113	0,500304	0,459983
5 maquinas fedora	10	0,367376	0,356495	0,292542	0,258964	0,231767
6 maquinas fedora	12	0,523028	0,472985	0,327214	0,273404	0,191388
7 maquinas fedora	14	0,668386	0,603985	0,391982	0,280968	0,266344

Analisando o conceito de eficiência, observa-se que o valor tido como ideal seria aquele em que todos os processadores do cluster que estão envolvidos na simulação tenham utilização total (100%), o que corresponde à eficiência de 1. Os resultados sobre a eficiência das diferentes configurações testadas estão mostrados na Tabela 5 e na Figura 29.

A Tabela 5 e a Figura 29 contêm valores de eficiência maiores do que 1. Isto se deve ao fato de que, além da comparação entre o speedup de uma máquina seqüencial e o de uma paralela, existe também a comparação com os valores referentes a uma máquina seqüencial otimizada. Portanto, como essa máquina otimizada possui o mesmo número de processadores da máquina seqüencial tida como padrão, valores de eficiência maiores do que 1 aparecem nos resultados.

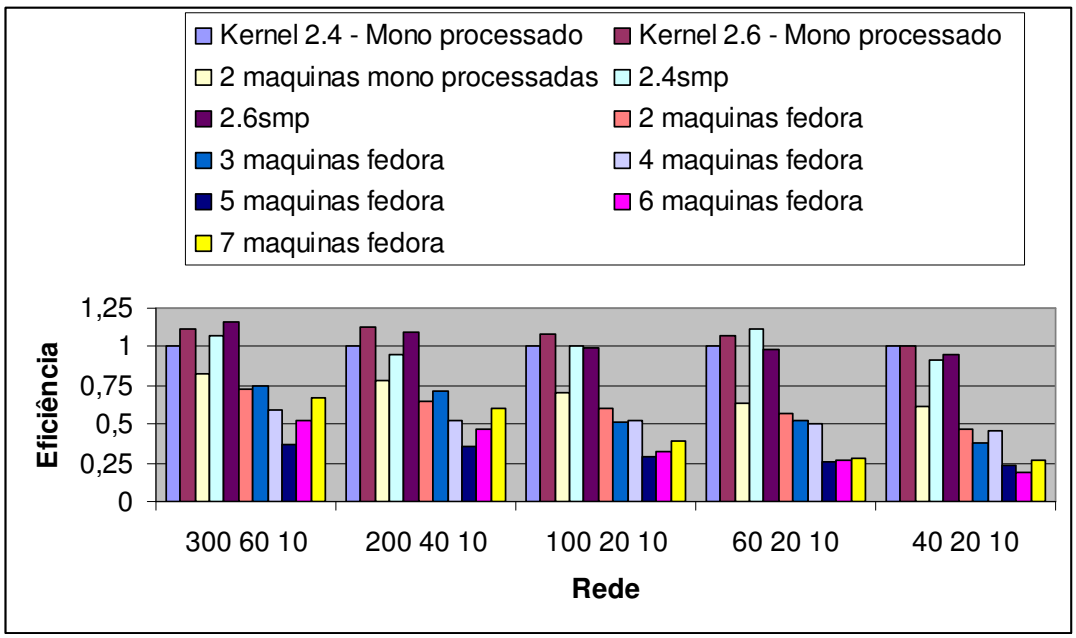


Figura 29 – Eficiência do cluster. Entende-se usando como exemplo 60 20 10 como retina 60x60, V1_Vert 20x20, V1_Horiz 20x20 divididos em 10 Slices, assim por diante.

4.4 Análise das tabelas e gráficos de resultados.

Analisando somente os dados de speedup (Tabela 4 e Figura 28) chega-se à conclusão de que quanto mais nós processadores estiverem trabalhando menor será o tempo de processamento, visto que os maiores coeficientes de speedup foram obtidos com o número máximo de nós utilizados.

Extraindo da Tabela 4 apenas os resultados obtidos com as máquinas com kernel 2.6 SMP (com HT), obtêm-se uma tabela simplificada (Tabela 6).

Tabela 6 – Cluster com kernel 2.6

Tempo de Execução (mS)	Redes = Retina V1 Fatias				
Número de passos = 50000	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
Kernel 2.6 - Mono processado	17190345	7758426	2134768	1171030	867730
Fedora	8249178	4013306	1168129	638173	464230
2 máquinas fedora	6591006	3354652	960354	548318	468590
3 máquinas fedora	4276428	2046654	759479	398045	387368
4 máquinas fedora	4064475	2099612	557154,7	311894,9	238143
5 máquinas fedora	5235030	2454980	792456,2	482050,4	378109,1
6 máquinas fedora	3064247	1541961	590406,2	380491,9	381569,8
7 máquinas fedora	2055298	1035018	422444,3	317356,2	235016,3

Os dados de speedup para o caso da Tabela 6, tendo um nó como referencia, estão mostrados na Tabela 7.

Tabela 7 – Speedup do cluster com o Kernel 2.6

	Speedup				
	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
Kernel 2.6 - Mono processado	1	1	1	1	1
Fedora	2,083886	1,933176	1,82751	1,834973	1,869181
2 máquinas fedora	2,608152	2,312736	2,222897	2,135677	1,851789
3 máquinas fedora	4,019791	3,790785	2,810832	2,941954	2,240066
4 máquinas fedora	4,229413	3,695172	3,831553	3,754566	3,643736
5 máquinas fedora	3,283715	3,160281	2,693862	2,429269	2,29492
6 máquinas fedora	5,609974	5,031531	3,615761	3,077674	2,274105
7 máquinas fedora	8,363919	7,495934	5,053372	3,689954	3,692212

O aumento de nós do cluster não produz um aumento linear no speedup e, consequentemente, no desempenho. O ganho chega a diminuir quando se passa de 4 para 5 nós, retomando o crescimento em seguida com um novo aumento de um nó, como mostrado na Figura 30.

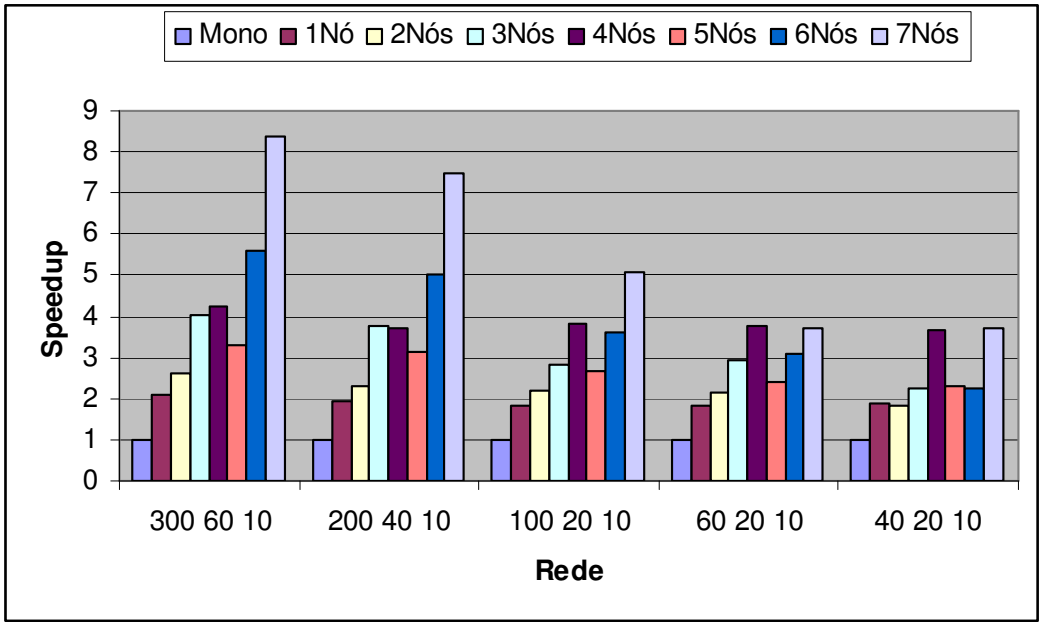


Figura 30 – Speedup do cluster com o Kernel 2.6. . Entende-se usando como exemplo 60 20 10 como retina 60x60, V1_Vert 20x20, V1_Horiz 20x20 divididos em 10 Slices, assim por diante.

Para redes pequenas (40-20-10 e 60-20-10) o maior índice de speedup foi obtido com 4 nós e a adição de novas máquinas provocou um acréscimo no tempo de execução. Isso indica que a afirmação genérica de que quanto mais nós adicionados ao cluster melhor o desempenho não procede, sendo que o desempenho do sistema depende muito do tipo e do tamanho da simulação que se deseja executar. No caso da simulação padrão deste estudo, para cada tamanho de rede se obteve um tamanho de máquina virtual ideal.

Ainda da Figura 30 percebe-se que números diferentes de nós fornecem coeficientes de speedup muito próximos, como, por exemplo, nos casos com 4 e 6 nós na rede com a configuração 100-20-10 e com 3 e 4 nós na rede com a configuração 300-60-10. Para uma melhor análise dos dados torna-se necessário o uso do conceito de eficiência junto com o de speedup e os dados de eficiência correspondentes aos dados da Figura 30 estão mostrados na Tabela 8.

Tabela 8 – Eficiência do cluster com o kernel 2.6

		Eficiência				
	Nº Proc	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
Kernel 2.6 - Mono	1	1	1	1	1	1
Fedora	2	1,041943	0,966588	0,913755	0,917486	0,934591
2 máquinas fedora	4	0,652038	0,578184	0,555724	0,533919	0,462947
3 máquinas fedora	6	0,669965	0,631798	0,468472	0,490326	0,373344
4 máquinas fedora	8	0,528677	0,461897	0,478944	0,469321	0,455467
5 máquinas fedora	10	0,328371	0,316028	0,269386	0,242927	0,229492
6 máquinas fedora	12	0,467498	0,419294	0,301313	0,256473	0,189509
7 máquinas fedora	14	0,597423	0,535424	0,360955	0,263568	0,263729

O uso conjunto dos coeficientes de eficiência e de speedup pode determinar qual o melhor tamanho da máquina virtual para cada simulação e tamanho de rede. Essa escolha permite otimizar os recursos do cluster de modo que uma simulação não utilize um número maior de nós do que ela realmente necessita para se obter o melhor desempenho. Portanto, buscando um maior desempenho a um menor custo é

interessante dimensionar o cluster para cada simulação em função dos valores dos coeficientes de speedup e de eficiência. Uma forma de se calcular o melhor valor é através da construção de uma tabela relacionando o produto dos dois coeficiente, como a Tabela 9.

Tabela 9 – Speedup x Eficiência do cluster com kernel 2.6

	Speedup X Eficiência				
	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
Kernel 2.6 - Mono processado	1	1	1	1	1
Fedora	2,17129	1,868584	**1,669897**	**1,683562**	**1,746919**
2 máquinas fedora	1,700614	1,337187	1,235318	1,140279	0,857281
3 máquinas fedora	2,693119	2,395009	1,316796	1,442515	0,836316
4 máquinas fedora	2,235992	1,706787	1,8351	**1,762096**	**1,659601**
5 máquinas fedora	1,078278	0,998737	0,725689	0,590135	0,526666
6 máquinas fedora	2,622651	2,109692	1,089478	0,78934	0,430963
7 máquinas fedora	**4,996795**	**4,013501**	**1,82404**	0,972554	0,973745

Para as redes menores mostradas na Tabela 9, o simples uso de um equipamento com dois processadores gera uma situação de boa eficiência com um speedup razoável, mas com o uso de 4 nós a eficiência continua boa e tem-se um melhor speedup. A mesma análise vale para a rede intermediária de configuração 100-20-10 com a diferença de que o coeficiente somente volta a melhorar com 7 nós. Já nas redes maiores o melhor coeficiente speedup x eficiência é obtido com 7 nós, demonstrando assim que para essas simulações o uso de todos os equipamento disponíveis resulta em considerável redução de tempo.

Infelizmente, não existe uma fórmula que permita calcular a melhor configuração do cluster para cada simulação, pois cada rede e sua simulação possuem características únicas, como forma dos neurônios e dinâmicas das sinapses, tamanho da rede e sua forma de divisão e comunicação para o uso com programação paralela. Portanto, o melhor caminho para a escolha do dimensionamento do cluster é o empírico: uma vez definido o tamanho da rede e a forma de comunicação entre os processos,

deve-se fazer estudos como os indicados neste trabalho para se dimensionar o tamanho do cluster apropriadamente.

Outro ponto interessante com relação ao dimensionamento do cluster é a forma em que se divide a simulação. No caso da rede padrão utilizada neste trabalho pode-se dividir as matrizes usadas em seus divisores comuns e a quantidade de divisões influencia no tempo de processamento da simulação, como mostrado na Figura 31.

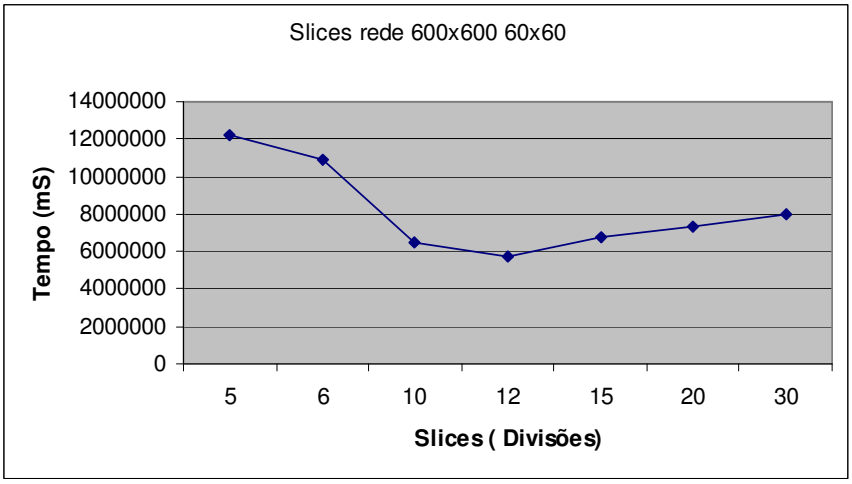


Figura 31 – Gráfico de Tempo X Fatias (slices)

Para a montagem do gráfico da Figura 31 foi usada uma rede considerada como grande (Retina 600X600, V1 vertical 60X60 e V1 horizontal de 60X60). Para um número crescente de fatias na faixa entre 5 a 12, o tempo de simulação diminui com o aumento do número de fatias. Isto se deve à melhor divisão de serviço entre os processadores que fazem parte do cluster. A partir de 15 e até 30 fatias tem-se um aumento no tempo de simulação e isto se deve à criação de um número maior de processos que, como conseqüência, provoca um aumento na demanda de comunicação entre eles e ocasiona um aumento na latência de resposta dos processadores, que são requisitados por um número maior de conexões.

O controle entre uma boa divisão da carga de processamento e um custo de comunicação dentro dos parâmetros aceitáveis de desempenho deve ser ponderado na elaboração e execução de uma simulação.

4.4.1 Cluster com Linux compilado e pré-compilado.

Ate o momento todos os dados foram obtidos com uma configuração do cluster utilizando o Linux pré-compilado Fedora Core 5. Apesar de sua instalação ser resumida, ela ainda não pode ser considerada otimizada para o equipamento em questão, já que tanto o kernel como os dispositivos, as aplicações e as ferramentas instaladas são pré-compilados (formato rpm). Para se obter o melhor desempenho dos equipamentos à disposição uma instalação otimizada foi feita utilizando o Linux Gentoo R5, sendo o kernel, as aplicações, os dispositivos e as ferramentas utilizadas todas compiladas no equipamento. Neste caso também se manteve a política da instalação mínima para atender às simulações e o sistema de acesso externo descrito anteriormente.

Para fins comparativos as mesmas redes das análises anteriores foram utilizadas neste que se pode chamar de cluster otimizado, além das configurações de memória, cache, modo de comunicação e dispositivos de redes selecionados nas análises anteriores deste capítulo. As Tabelas 10 e 11 mostram os dados obtidos nas simulações com o novo cluster e as obtidas anteriormente, eliminando-se os dados obtidos com a simulação mono processada, para comparação. Os valores de speedup da Tabela 11 foram calculados tendo como base 1 nó-SMP com Fedora.

Tabela 10 – Comparação entre cluster pré-compilado e compilado

Tempo de Execução (mS)	Redes = Retina V1 Fatias				
Número de passos = 50000	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
Fedora	8249178	4013306	1168129	638173	464230
2 máquinas fedora	6591006	3354652	960354	548318	468590
3 máquinas fedora	4276428	2046654	759479	398045	387368
4 máquinas fedora	4064475	2099612	557154,7	311894,9	238143
5 máquinas fedora	5235030	2454980	792456,2	482050,4	378109,1
6 máquinas fedora	3064247	1541961	590406,2	380491,9	381569,8
7 máquinas fedora	2055298	1035018	422444,3	317356,2	235016,3
Gentoo	8078474	3746250	1027372	529726	374808
2 máquinas gentoo	5907881	2958829	855778	424948	439102
3 máquinas gentoo	3728233	1671768	575114	321089	369528
4 máquinas gentoo	3643714	1863076	469905	290674	213490
5 máquinas gentoo	3095793	1437933	454758	292365	223599
6 máquinas gentoo	1844885	908828	329229	229082	229731
7 máquinas gentoo	1842530	908691	349140	295570	210687

Tabela 11 – Comparação entre speedups do cluster pré-compilado e compilado

	Speedup				
	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
Fedora	1	1	1	1	1
2 máquinas fedora	1,251581	1,19634	1,216353	1,163874	0,990695
3 máquinas fedora	1,928988	1,960911	1,538066	1,603268	1,198421
4 máquinas fedora	2,02958	1,911452	2,096597	2,046116	1,949375
5 máquinas fedora	1,575765	1,634761	1,474061	1,323872	1,227767
6 máquinas fedora	2,692074	2,602728	1,978517	1,677232	1,216632
7 máquinas fedora	4,013617	3,877523	2,765167	2,010904	1,97531
gentoo	1,021131	1,071286	1,137007	1,204723	1,238581
2 máquinas gentoo	1,396301	1,356383	1,364991	1,501767	1,057226
3 máquinas gentoo	2,212624	2,400636	2,031126	1,987527	1,256278
4 máquinas gentoo	2,263948	2,154129	2,485883	2,195494	2,174481
5 máquinas gentoo	2,664641	2,791024	2,568683	2,182795	2,076172
6 máquinas gentoo	4,471378	4,415914	3,548074	2,785784	2,020755
7 máquinas gentoo	4,477093	4,416579	3,345732	2,159126	2,203411

Para analisar este caso, um enfoque diferente dos anteriores foi seguido. Ao invés de uma análise de speedup e de eficiência, foi montada a Tabela 12 relacionando os resultados equivalentes entre o cluster chamado de pré-compilado e o chamado otimizado ou compilado, mostrando o ganho ou speedup entre cada configuração do cluster (razão entre itens equivalentes).

Tabela 12 – Ganho do cluster compilado

	Ganho Kernel Compilado / Pré-compilado				
	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
1 Nó - Fedora/Gentoo	1,021131	1,071286	1,137007	1,204723	1,238581
2 Nó - Fedora/Gentoo	1,115629	1,133777	1,1222	1,290318	1,067155
3 Nó - Fedora/Gentoo	1,147039	1,224245	1,320571	1,239672	1,048278
4 Nó - Fedora/Gentoo	1,115476	1,12696	1,185675	1,073006	1,115476
5 Nó - Fedora/Gentoo	1,691014	1,707298	1,742589	1,648796	1,691014
6 Nó - Fedora/Gentoo	1,660942	1,696648	1,7933	1,660942	1,660942
7 Nó - Fedora/Gentoo	1,115476	1,139021	1,209957	1,073709	1,115476

Pelos dados da Tabela 12 vê-se que, desde o modesto incremento de 2,1% obtido por 1 nó na rede de configuração 300-60-10 até os impressionantes 79,33% obtidos por 6 nós na rede de configuração 100-20-10, há uma melhora no desempenho quando se passa de um cluster pré compilado para um otimizado ou compilado.

4.4.2 Linpack para efeito comparativo com outros clusters

Como já mencionado, o Linpack é uma ferramenta de grande aceitação para a medição do desempenho de um computador, tanto de um supercomputador como de um cluster de computadores. Através da ferramenta HPL – High Performance Linpack, foram obtidos os resultados mostrados na Tabela 13.

Tabela 13 – Análise de desempenho Linpack

Linpack - Cluster 7 Nós	
Valor Teórico	40,65Gflops/s
Valor Medido	29,34Gflops/s

Entende-se como valor teórico o produto entre o valor obtido para um nó e o número de nós existentes no cluster. Neste caso o valor obtido para 1 nó foi de 5,807 Gflops/s, o que implica que para 7 o valor teórico seria 7 x 5,807 = 40,65 Gflops/s. O valor medido normalmente é bem menor que o teórico e não foi diferente no presente caso (ver Tabela 13), uma vez que existe todo o custo de processamento para envio e recebimento das mensagens entres os nós do cluster.

5 Conclusão

A seguir apresentam-se as considerações e conclusões obtidas deste trabalho, além da contribuição ao meio acadêmico esperada com o mesmo. Uma breve descrição das dificuldades obtidas e uma perspectiva para trabalhos futuros estão dadas em seguida.

5.1 Resumo do estudo

Utilizando uma simulação padrão foi realizada uma análise do comportamento do hardware alterando-se alguns parâmetros de configuração e dispositivos buscando o melhor desempenho com estes itens. Após isto, a análise foi com relação ao software, selecionando-se algumas aplicações e ferramentas e eliminando-se as que não seriam utilizadas, chegando-se deste modo a uma máquina "limpa" contendo todos os serviços necessários para a realização das simulações, mas utilizando-se apenas os pacotes de instalação chamados de pré-compilados. Como o cluster utilizado neste trabalho é um cluster homogêneo, a mesma instalação refinada foi repetida em todos os nós disponíveis. Neste cluster foi realizada uma análise do ganho de desempenho com a divisão da simulação em vários nós do cluster. Um novo cluster foi configurado utilizando-se desta vez apenas pacotes de instalação chamados de fontes, onde toda a compilação do kernel, aplicativos e ferramentas foi realizada fornecendo-se os dados específicos do hardware da máquina em questão. Neste novo cluster parte da análise feita anteriormente foi repetida e comparada com a anterior. As conclusões obtidas com este trabalho estão dispostas a seguir.

5.2 Conclusões experimentais

Na análise de hardware observou-se que alguns componentes têm influencia direta sobre o ganho de desempenho, tais como:

- Processador: item relevante a ser levado em consideração quando o assunto for cluster de alto desempenho.

- Memória cache: a correta configuração de escrita de cache é fundamental para o bom rendimento do processador. O aumento da quantidade de memória cache também contribui para melhorar o desempenho do processador.

- Hardware paralelo: o uso de novas tecnologias que têm como base o uso de processamento paralelo como a SMP e os novos processadores que possuem características de processamento paralelo tais como a HT e a Core 2 Duo aumentam a capacidade de processamento para aplicações concorrentes ou paralelas, como no caso do cluster do estudo.

- Barramento de comunicação: para que o processador possa trabalhar na sua máxima capacidade uma placa com barramento adequado é essencial.

- Dispositivo de comunicação em rede: este tipo de cluster, que tem como forma de sincronização a troca de mensagens pela rede, e um dispositivo mais rápido diminui o tempo de sincronização diminuindo a latência em um processador quando este necessita de informação de outro nó.

Alguns componentes de hardware podem influenciar no desempenho caso estejam mal dimensionados para as simulações previstas, tais como memória RAM e espaço em disco rígido, porém estes itens só influenciam no desempenho após sua saturação.

Na análise de software a comunicação utilizando RSH se mostrou mais rápida do que a utilizando a comunicação SSH. Para clusters dedicados sem interface com a

internet esta é a melhor solução. Infelizmente, não foi possível obter um resultado conclusivo sobre a melhor escolha na utilização das bibliotecas de troca de mensagens PVM e MPI.

A divisão da simulação para rodar em mais nós do cluster trouxe um ótimo ganho de desempenho. Para o uso do cluster de modo a se ter o maior desempenho utilizando-se a quantidade de nós necessária é muito importante considerar, alem dos valores de speedup e da eficiência do sistema, o custo do processamento.

A compilação do sistema operacional Linux, que pode ser chamada de uma personalização ou otimização do Linux para o hardware do cluster, trouxe um ganho considerável nas simulações utilizando uma única maquina e, principalmente, nas simulações que utilizaram mais nós, sendo que este ganho chegou a 70% em relação ao cluster pré-compilado no caso do cluster com 7 nós (o máximo número de nós considerado neste estudo).

5.3 Contribuições deste trabalho

Com este trabalho espera-se que algumas contribuições sejam aceitas pelos colegas usuários de simulação computacional de redes neurais e de processamento paralelo em geral. Espera-se que:

- Os resultados e conclusões aqui apresentados sirvam de referencia para o dimensionamento de novos nós para o cluster ou na modelação de um novo cluster em outros laboratórios.

- Para os colegas que estejam iniciando a construção de seu modelo utilizando o GENESIS e que possuam acesso a um cluster, que neste texto possam enxergar que vale a pena dispor de um tempo maior para desenvolver o modelo visando a sua simulação em paralelo usando o PGENESIS. Durante a etapa de aquisição de dados rodando a

simulação proveniente do modelo desenvolvido, o tempo a mais gasto na programação em paralelo será recuperado e abrirá a possibilidade para um número maior de combinações uma vez que o tempo necessário para esta análise será diminuído.

- O manual de configuração do cluster com PGENESIS passo a passo nos moldes dos documentos “How to” utilizados mundialmente pela comunidade Linux (Anexo I) ajude os iniciantes a ter um cluster de bom desempenho a um baixo custo de implementação.

5.4 Dificuldades encontradas

Durante a execução deste trabalho algumas dificuldades foram encontradas e, a seguir, segue um resumo destas.

- A necessidade da utilização do Linux, além da constante adequação de ferramentas às necessidades do projeto. Apesar da grande quantidade de aplicativos e ferramentas disponíveis para o Linux, grande parte delas se encontra apenas no formato “fonte” sendo necessário sua compilação e configuração para uso.

- Ausência de simulações paralelas para o PGENESIS utilizando MPI. Devido ao fato de a implementação do MPI no PGENESIS ser recente, ainda não existe uma simulação modelo capaz de ser executada com PVM e MPI com total compatibilidade.

- Inicialmente tinha-se a idéia de usar alguns dos modelos desenvolvidos no próprio Laboratório SisNe para este estudo, os quais seriam transformados de suas versões feitas no GENESIS para o PGENESIS, utilizando-se tanto a biblioteca PVM como a MPI, mas após algumas análises ficou evidente que um modelo computacional cujo criador não tinha a visão de futuramente transformá-lo para uma versão usando programação paralela torna sua migração para esta versão uma tarefa muito complexa e

com resultados duvidosos, uma vez que a idéia original do desenvolvedor do modelo pode acabar se perdendo.

- Hardware homogêneo e sem possibilidade de alterações de componentes. Apesar de toda facilidade envolvida na configuração de um cluster com hardware homogêneo, esta característica dificulta algumas comparações. Como não se dispõe de elementos diversificados, não é possível analisar o ganho entre plataformas diferentes, como, por exemplo, processadores com freqüências e tecnologias diferentes, e o ganho com o aumento do cache dos processadores ou com a mudança do barramento de comunicação interna de 266 MHz para 400 MHz ou 800 MHz, já disponíveis hoje no mercado.

5.5 Sugestões para trabalhos futuros

Nesta linha de trabalho, seria muito interessante ampliar a análise para um cluster de hardware heterogêneo. Quando se trabalha com um cluster homogêneo todos os nós possuem a mesmas capacidades computacionais e a divisão das tarefas pode ser igual, o que facilita muito a programação da simulação e o gerenciamento do cluster, tanto que para este estudo nenhum sistema gerenciador de cluster se fez necessário. Com o uso de um cluster heterogêneo a visão da programação do modelo deve ser diferenciada e o uso de um sistema mais complexo de gerenciamento de recursos se faz necessário porque cada nó terá uma capacidade computacional diferente e uma correta distribuição de tarefas entre os nós terá grande diferença na diminuição do tempo da simulação.

Um ponto que ficou carente de uma melhor definição é o uso da biblioteca MPI na utilização do PGENESIS e, portanto, um estudo mais detalhado deste uso e sua comparação com a outra opção disponível que é a biblioteca PVM pode ser objeto de um trabalho futuro.

5.6 Considerações Finais

Utilizando de ajustes no hardware e, principalmente, de uma otimização do software utilizado no cluster é possível melhorar o seu desempenho consideravelmente. Da primeira configuração do equipamento utilizando o Linux kernel 2.4 sem suporte SMP até a última, utilizando Linux kernel 2.6 compilado com suporte SMP, obteve-se um ganho na mesma simulação e no mesmo equipamento de 228% (ver Tabela 14).

Para o cluster na sua configuração máxima com os 7 nós disponíveis para este estudo obteve-se, utilizando-se a configuração otimizada, até 79% de ganho em relação à configuração pré-compilada rodando a mesma simulação (ver Tabela 12). Desconsiderando o custo computacional envolvido e fazendo-se uma comparação para a mesma simulação executada na primeira configuração, utilizando o Linux kernel 2.4 sem suporte SMP, e na ultima, utilizando a configuração máxima otimizada do Cluster (7 nós com Linux compilado com suporte SMP), obteve-se o ganho de máximo de 1043% ou um speedup de 10,43 (ver Tabela 14).

Apesar da variância de ganho a versão otimizada sempre levou a um acréscimo de desempenho em comparação com a configuração básica, demonstrando desta forma que uma correta configuração do cluster buscando sua otimização é essencial para se obter toda a capacidade que o equipamento pode fornecer.

Tabela 14 – Speedup relativo entre as configurações mínima/máxima do cluster

Tempo de Execução	Redes = Retina V1 Fatias				
Número de passos = 50000	300 60 10	200 40 10	100 20 10	60 20 10	40 20 10
Kernel 2.4 - Mono processado	19232254	8751889	2318268	1248337	876333
Kernel 2.6 SMP - Compilado	8078474	3746250	1027372	529726	374808
7Nós Kernel SMP Compilado	1842530	908691	349140	295570	210687
	Speedup				
Kernel 2.4 / Kernel 2.6 Compilado	2,380679	2,336173	2,256503	2,356571	2,071282
Kernel 2.4 / Cluster 7 Nós Compilado	10,43796	9,631315	6,639938	4,22349	3,684769

O uso da ferramenta PGENESIS e de um cluster de computadores para sua execução mostrou-se altamente viável. O custo de implementação deste cluster comparado à alternativa de um supercomputador é bem menor, o que tornou possível sua aquisição.

De posse destes resultados positivos é possível estudar a ampliação do equipamento e a criação de um centro de alto processamento, disponibilizando o equipamento para outras áreas e laboratórios dentro e fora da Universidade. Com esta possível ampliação o cluster, devido à renovação tecnológica ocorrida desde sua montagem, se tornaria um cluster heterogêneo abrindo o leque de pesquisas na área de computação paralela possíveis de serem executadas nele.

6 Referências Bibliográficas

1. Almasi, G. S., *High Paralell Computing*, 2nd Ed, The Benjamin Cummings Publishing Company, San Francisco, CA, 1994.
2. Bar, M. The OpenMosix Project. Disponível em: <http://openmosix.sourceforge.net>. 2006. Consultado em 13/02/2007.
3. Barcellos, M. P. e Gaspary, L. P. Tecnologias de Rede para Processamento de Alto Desempenho. Anais da 3ª Escola Regional de Alto Desempenho (ERAD). Santa Maria, RS: UFRGS, 2006.
4. Bower, J. M. and Beeman, D., *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System*, 2nd. ed., Springer-Verlag, Santa Clara, CA, 1998.
5. Breslin, C. and O'Lenskie, A., Neuromorphic hardware databases for exploring structure-function relationships in the brain. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 356: 1249-1258, 2001.
6. Campos, A. O que é Linux. BR-Linux. Florianópolis, março de 2006. Disponível em <http://br-linux.org/linux/faq-linux>. Consultado em 13/02/2007.
7. Carnevale, T. and Hines, M. The NEURON Book. Cambridge University Press, Cambridge, 2006.
8. Colombet, L. and Desbat, L. Speedup and efficiency of large size applications on heterogeneous networks. *Theoretical Computer Science*, 196: 31-44, 1998.
9. Data book Intel E7501, 2003.
10. de Pinho M., Mazza, M. and Roque A. C., A computational model of the primary auditory cortex exhibiting plasticity in the frequency representation. *Neurocomputing*, 70: 3-8, 2006.

11. De Schutter E. and Bower J. M. An active membrane model of the cerebellar Purkinje cell. I. Simulation of current clamps in slice. *Journal of Neurophysiology*, 71: 375-400, 1994

12. Dongarra, J. J., Duff, I. S., Sorensen, D. C. and van der Vorst, A. Numerical Linear Algebra on High-Performance Computers. SIAM books, Philadelphia, PA, 1989.

13. Dongarra, J. J., Luszczeck, P. and Petitet, A. The LINPACK Benchmark: Past, Present and Future. Department of Computer Science, University of Tennessee, Knoxville, TN, 2001. Disponível publicamente em <http://www.netlib.org/utk/people/JackDongarra/papers.htm>. Acessado 06/08/2007.

14. Dongarra, J. Linpack Benchmark. Disponível em < http://performance.netlib.org/performance/html/linpack.data.col0.html>, 2005. Acessada em 06/08/2007.

15. Flynn, M. J. Some computer organizations and their effectiveness. *IEEE Transactions on Computing*, 9, 948–960, 1972.

16. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V., PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing. MIT Press, Cambridge, MA, 1994.

17. GPACP. <http://black.rc.unesp.br/gpacp/>. Acessado em 20/08/2007.

18. Hartmann, G., Frank, G., Schaefer, M. and Wolff, C., SPIKE 128K: an accelerator for dynamic simulation of large pulse-coded networks. In: Klar, H. and Konig, A. (Eds.), *Proceedings of MicroNeuro'97*, Dresden, Germany, 1997, pp. 130-139.

19. Hines, M. L., The Neurosimulator NEURON. In: Koch, C. and Segev, I. (Eds.), *Methods in Neuronal Modeling*, 2nd ed., MIT Press, Cambridge, MA, 1998, pp. 129-136.

20. Hodgkin, A. L., Huxley, A. F. and Katz, B., Measurement of current-voltage relations in the membrane of the giant axon of *Loligo*. *J. Physiol. (Lond.)* 116:424-448, 1952.

21. Hood, G. Parallel GENESIS: its uses and applications. Tutorial apresentado no *GENESIS Users Meeting*, 8-10 de novembro de 2002, San Antonio, TX, EUA. URL: http://www.genesis-sim.org/GENESIS/gum-tutorials/hood/index.html, 2002.

22. Jahnke, A., Schönauer, T., Roth, U., Mohraz, K. and Klar, H., Simulation of spiking neural networks on different hardware platforms. In: International Conference on Artificial Neural Networks (ICANN'97), Lausanne, Switzerland, Springer-Verlag, Berlin,1997.

23. Jahnke, A., Roth, U. and Schönauer T., Digital Simulation of Digital Spiking Neural Networks. In: Maass, W. and Bishop, C. M. (Eds), *Pulsed Neural Networks*, MIT Press, Cambridge, MA, 1998, ch 9.

24. Kirner, C., Arquiteturas de sistemas avançados de computação, *Anais da Jornada EPUSP/IEEE em Sistemas de Computação de Alto Desempenho*, pp. 307-353, 1991.

25. Koch C. and Segev I., *Methods in Neural Modeling: From Ions to Networks*, 2nd ed., MIT Press, Cambridge, MA, 1998.

26. Meuer, H. W., Strohmaier, E., Dongarra, J. J. and Simon, H. D. Top500 Supercomputer Sites, 2004. Disponível em <http://www.netlib.org/benchmark/top500.html>. Acessado em 06/08/2007.

27. Mazza, M., de Pinho, M., Piqueira, J. R. C. and Roque, A. C. A dynamical model of fast cortical reorganization. *Journal of Computational Neuroscience*, 16:177-201, 2004.

28. Merkey, P., *Beowulf History*. Disponível em <http://www.beowulf.org/overview/history.html>, 2004. Acessada em 26/07/2007.

29. Oliveira, R. F. and Roque, A. C., A biologically plausible neural network model of the primate primary visual system. *Neurocomputing*, 44-46: 957-963, 2002.

30. Oliveira, R. F., Modelação de Fenômenos de Plasticidade Rápida no Sistema Visual de Mamíferos. Tese de Doutorado. Programa de Pós-Graduação em Psicobiologia, Departamento Psicologia e Educação, Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, USP, 2006.

31. Omondi, A. R., Neurocomputers: a dead end?, *International Journal of Neural Systems*, 10: 475-481, 2000.

32. Patterson, D. A. and Hennessy, J. L., *Computer Organization & Design: The Hardware/Software Interface*, 3rd. Ed, Morgan Kauffman, San Francisco, CA, 2004.

33. Pitanga, M., *Construindo Supercomputadores com Linux*, 2ª Ed., Editora Brasport, Rio de Janeiro, 2004.

34. Publio, R., Oliveira, R. F. and Roque A. C., A biologically plausible simulation of rod photoreceptor for use in a large-scale retina network model. *Neurocomputing*, 69: 1020-1024, 2006.

35. Roth, U., Jahnke, A. and Klar, H., Hardware requirements for spike-processing neural networks. In: *Proceedings of IWANN'95*, Malaga, Spain, 1995, pp. 720-727.

36. Schaefer, M., Schoenauer, T., Wolff, C., Hartmann, G., Klar, H. and Rückert, U., Simulation of spiking neural networks: architectures and implementations. *Neurocomputing*, 48: 647-679, 2002.

37. Simões-de-Souza F. M. and Roque A. C., A biophysical model of vertebrate olfactory epithelium and bulb exhibiting gap junction dependent odor-evoked spatiotemporal patterns of activity. *BioSystems*, 73: 25-43, 2004a.

38. Simões-de-Souza F. M. and Roque A. C., Self-sustained waves in a computational model of the olfactory epithelium with gap junctions. *Neurocomputing*, 58-60: 1033-1039, 2004b.

39. Simões de Souza, F. M., Estudo da Origem e do Papel das Oscilações Elétricas em um Modelo Computacional do Sistema Olfativo de Vertebrados. Tese de Doutorado. Programa de Pós-Graduação em Psicobiologia, Departamento Psicologia e Educação, Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto, USP, 2005.

40. Snir, M. and Gropp, W., *MPI: The Complete Reference*, 2 Vols., MIT Press, Cambridge, MA, 1998.

Anexo I – Manual montagem Cluster Linux/PVM

Este manual de montagem de um cluster para rodar o neuro-simulador paralelo Pgenesis utilizando PVM e SSH foi escrito no formato HOW TO, comum na comunidade Linux e direcionado para usuarios com pouca ou nenhuma experiencia com Linux, trata-se de uma receita que seguida fielmente funciona. A escolha do Fedora 5 para este manual é devido a grande compatibilidade desta distribuição com vários Hardwares disponiveis, evitando desta maneira o problema comum no Linux de falta de drivers para video, rede e outros dispositivos, alem da facilidade do uso de ferramentas gráficas.Para este Cluster voce precisará no minimo.

-2 Computadores com placa de rede (PII 400Mhz com 128MbRAM ou superior, quanto melhor o processador e acessorios , memoria, placa rede, HD, melhor será o desempenho do seu cluster)

-Dispositivo de comunicação : Hub, Switch

- Acesso a internet (banda larga de preferencia para baixar os CDs de instalação)

- Gravador de CDs, com software que reconheça a extensão iso (nero por exemplo)

Inicialmente é necessario baixar os CDs do Fedora Core 5 no endereço abaixo:

http://download.fedora.redhat.com/pub/fedora/linux/core/5/i386/iso/

Baixe os arquivos .iso para gravar os CDs de instalação, voce vai precisar os arquivos disc1.iso ate disc5.iso. Grave os 5 CDs.

Com os CDs em mão vamos iniciar a instalação.

Supondo que as maquinas que serão utilizadas na montagem do cluster são dedicadas e portanto podem ser apagadas, nada impede o uso de Dual-boot para uso geral da outra partição, mas este manual não contempla este recurso.

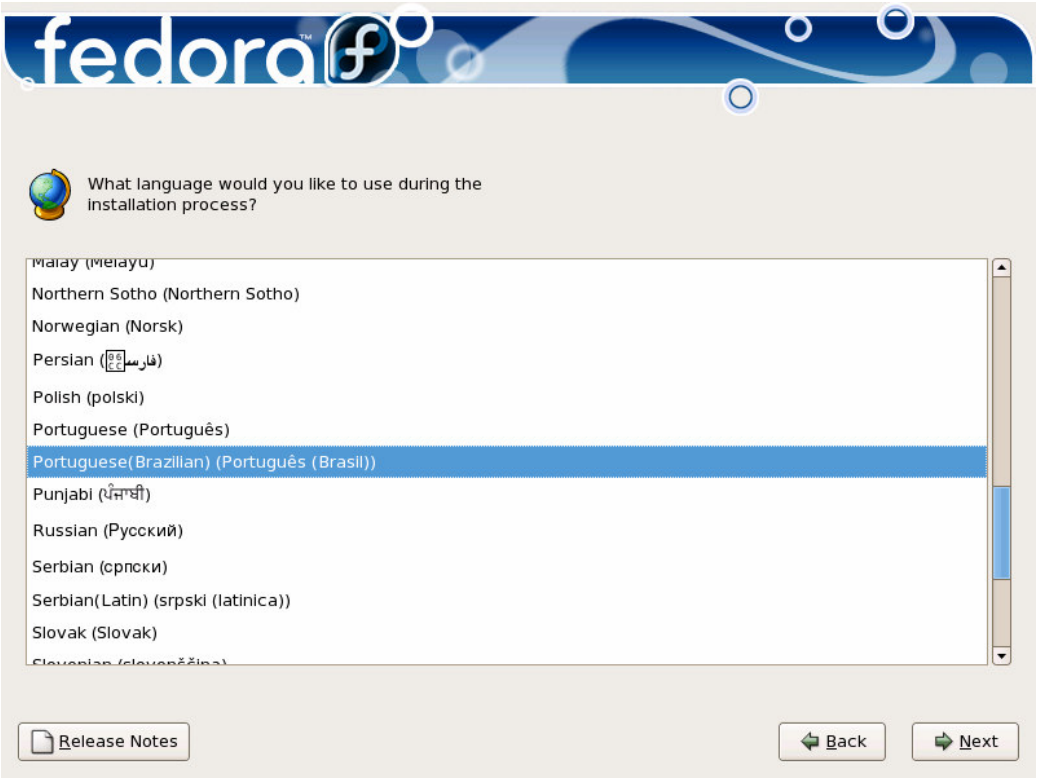
Insira o CD Fedora 1 e incie a maquina (se necessario acesse o setup da maquina e configure a opção boot por CD-Rom)

Seguindo as telas, tecle enter.

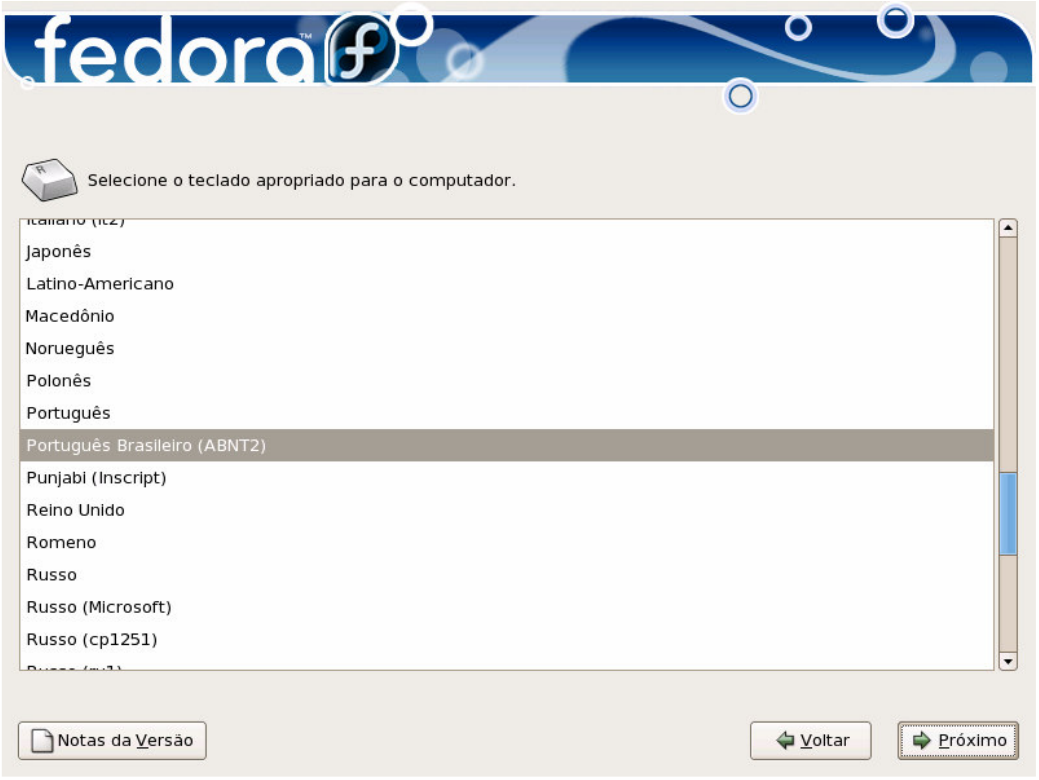
Mude para skip e tecle enter

Apenas tecle enter

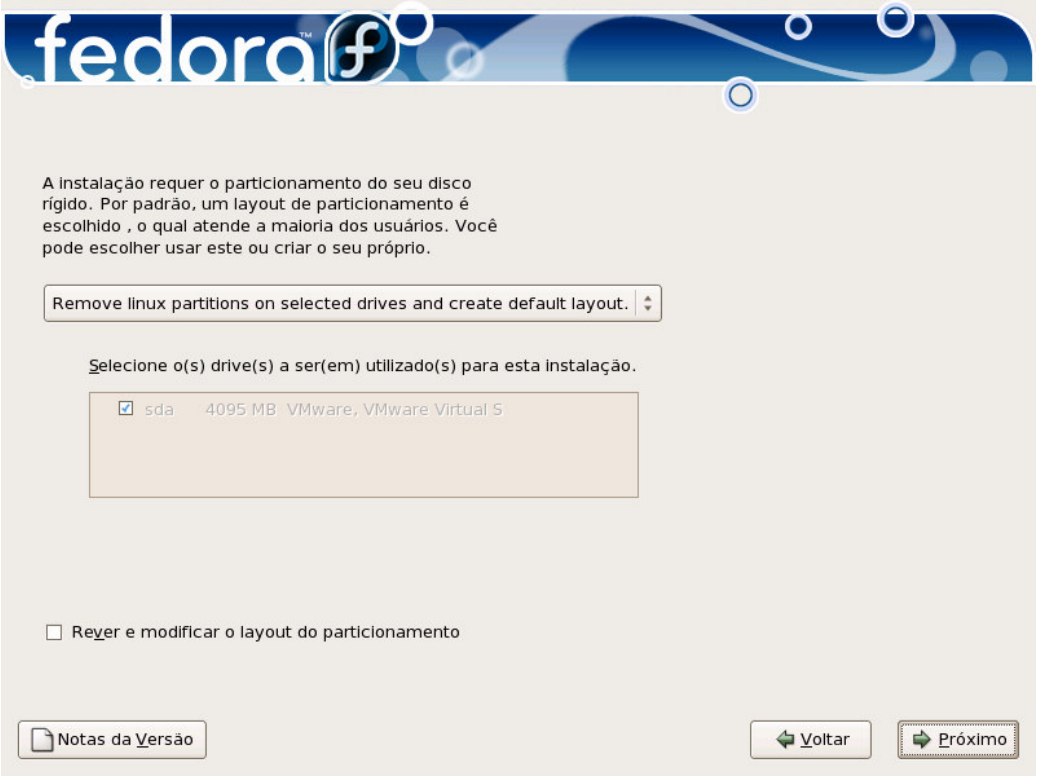
Selecione linguagem portugues-Brasil e tecle enter



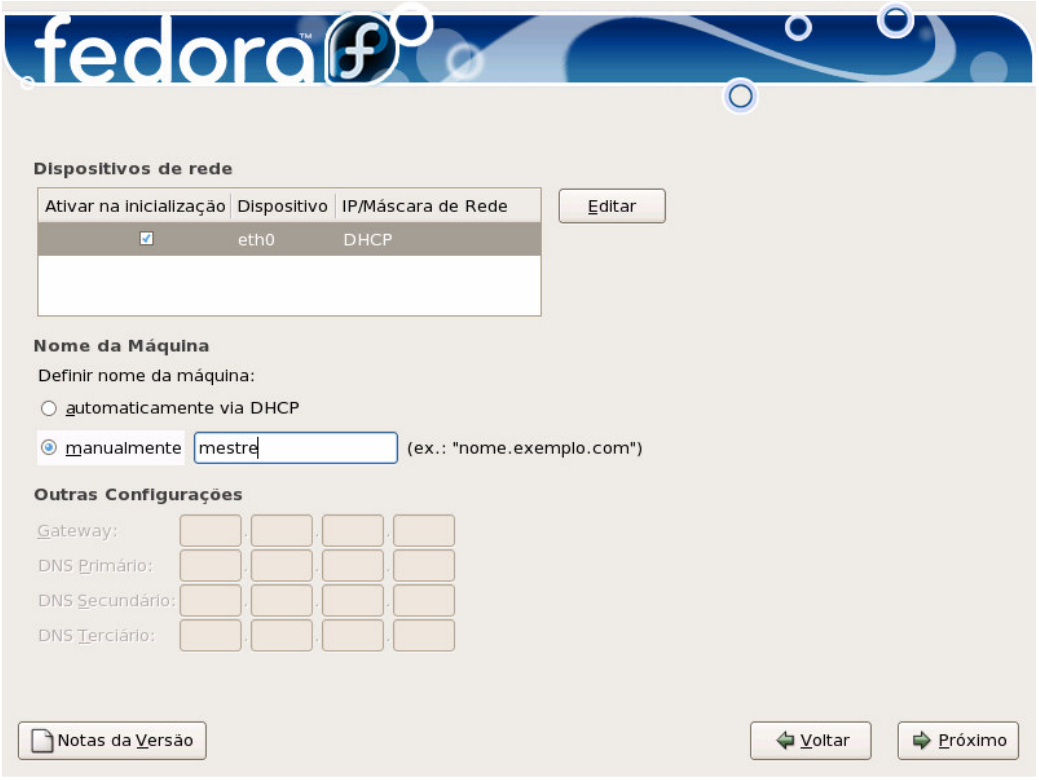
Selecione o teclado, na maioria dos casos selecione ABNT, telce proximo.



Limpando partições e criando uma partição padrão, tecle proximo



Caso voce possua um servidor DHCP em sua rede apenas defina o nome da maquina.



Caso não tenha um servidor DHCP, atribua o ip, mascara, gateway e o DNS manualmente clicando em Editar.

Recomendo esta configuração para Cluster dedicado.

Este ponto merece um pouco mais de atenção. Tanto os CDs do fedora quanto o pvm e os arquivos do GENESIS e Pgenesis são baixados da internet, portanto uma forma de acesso a internet deve estar disponivel, voce pode baixar os arquivos em outra maquina com acesso a internet e instala-los no Cluster normalmente também, como fiz neste cluster modelo. Selecione a região para ajuse de horario e clique proximo.

Defina uma senha para o usuario root, use a mesma para todas as maquinas do Cluster.

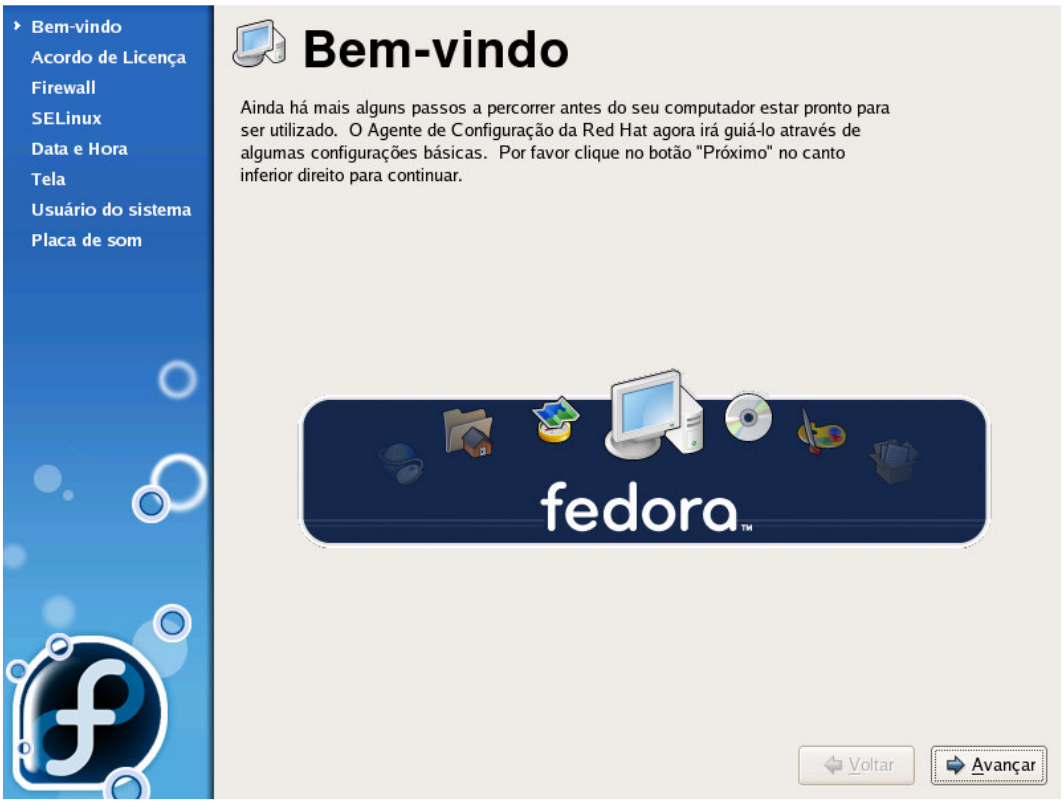
Clique em persolize agora.

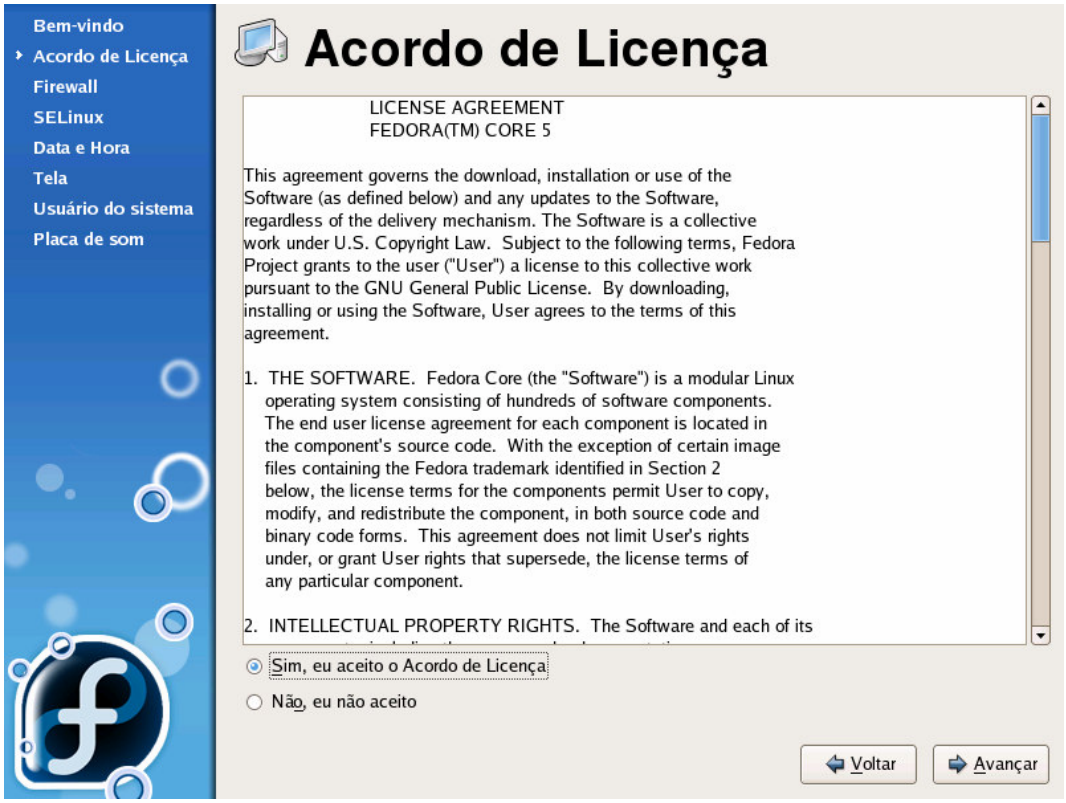
No item desenvolvimento selecione, bibliotecas de desenvolvimento, desenvolvimento de software Gnome, Desenvolvimento de Software X, desenvolvimento de software legado e ferramentas de desenvolvimento. Isto irá fornecer os compiladores e bilbiotecas necessarias para instalação do Pgenesis.

Clique em proximo para continuar a instação, troque os CDs conforme for solicitado.

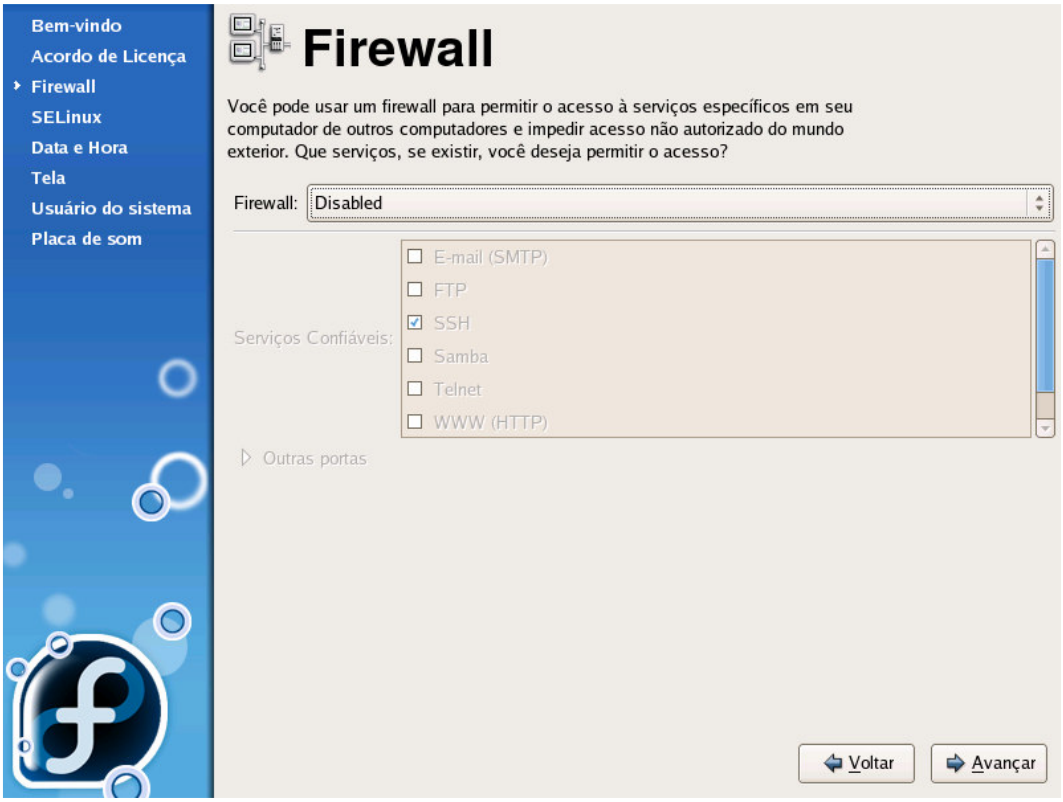
Quando a instalação estiver completa clique em reinciar

Clique em avançar, e depois clique Sim e avançar





Como estamos considerando este Cluster dedicado, para facilitar a comunicação e evitar a criação de regras vamos desativar o Firewall . Clique desabled e avançar

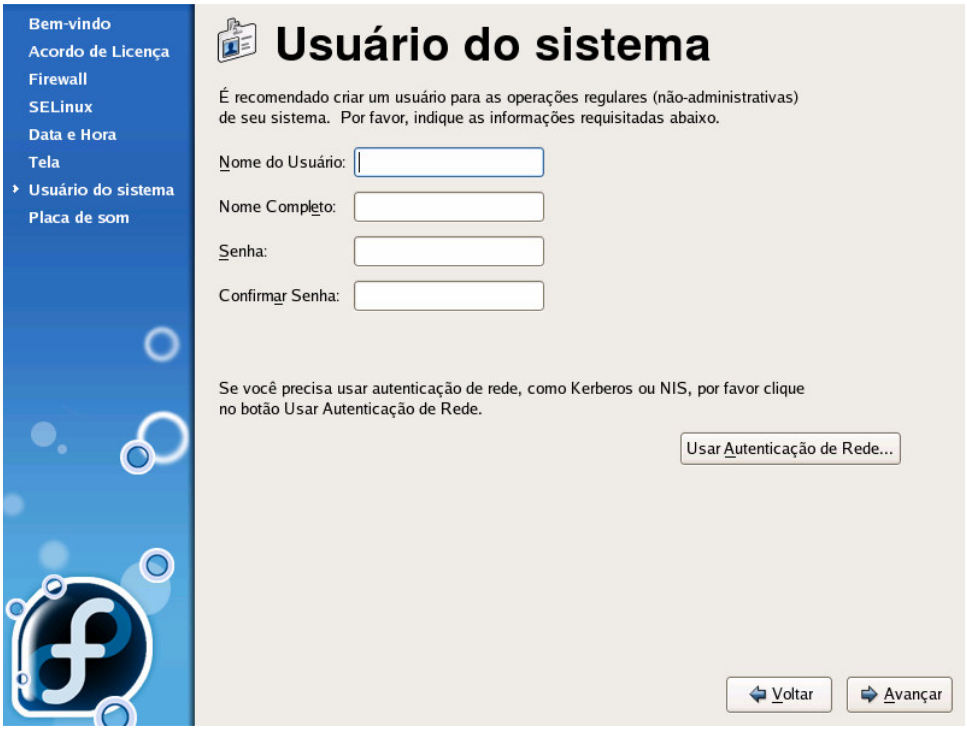


Clique em desativar os recursos avançados de segurança e avançar.

Ajuste a data e hora e depois clique em avançar.

Ajuste a configuração da tela no minimo em 800x600 e avançar

Como o cluster é dedicado vamos utilizar o usuario root para as simulações, portanto pode avançar sem a criação de um novo usuario.

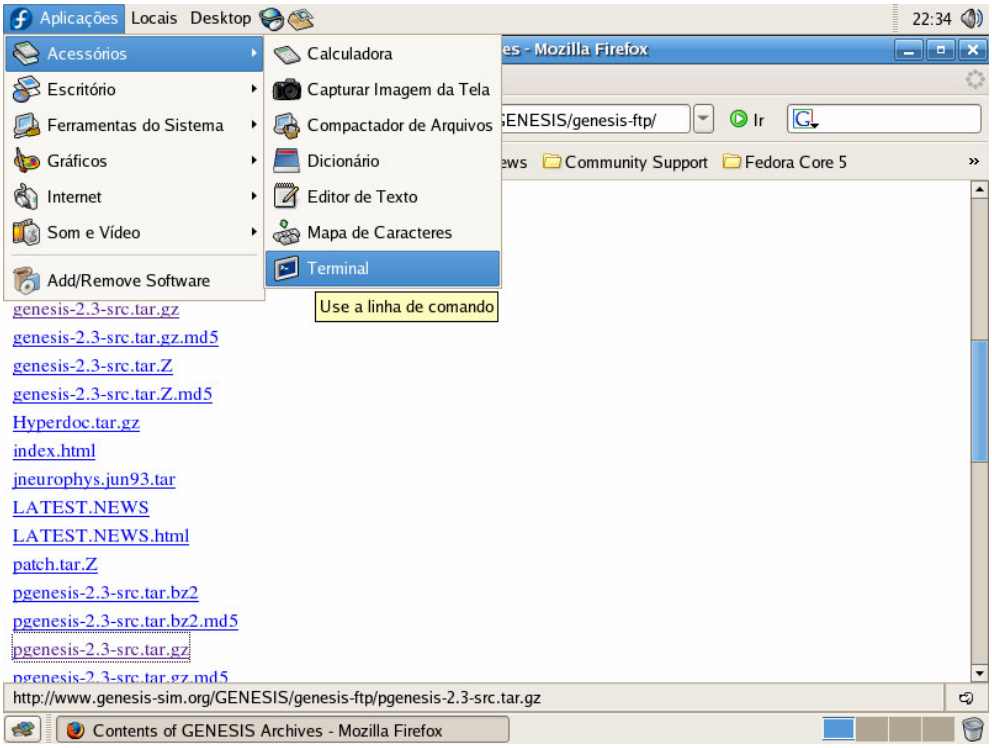


Nem sempre a placa de som é reconhecida, mas para a função desejada isto é irrelevante, pode terminar a configuração.

Abrir o navegador e digitar o seguinte site para fazer download do Pgenesis.

http://www.genesis-sim.org/GENESIS/genesis-ftp/

Abra uma janela de terminal.



Baixar os arquivos genesis-2.3.tar.gz e pgenesis-2.3.tar.gz, digitar os comandos onde baixou os arquivos (padrão /root/Desktop)

> tar –zxvf genesis-2.3.tar.gz
> mv genesis-2.3/genesis /usr/local/
> cd /usr/local/genesis/src
> cp Makefile.dist Makefile
> nano Makefile (use o vi se preferir)

Alterar a linha a seguir para

INSTALLDIR = /usr/local/genesis

Localize o cabeçalho

System: Linux 1.2.x and up on Intel x86-based, Xeon,

Descomente as linhas conforme mostrado abaixo

MACHINE=Linux
OS=BSD
XINCLUDE=-I/usr/X11R6/include
Choose ONE XLIB line to uncomment:
For 32-bit architectures
XLIB=/usr/X11R6/lib

```
## For 64-bit machines, probably need /usr/X11R6/lib64 here.
# XLIB=/usr/X11R6/lib64
 CC=cc
## Old (and probably broken) gcc installations may need the full
## path to cpp (preferably NOT one in /lib). If there isn't a
## [link to] cpp in the same directory as 'cc', you should consider
## [re]installing a newer gcc.
 CPP=cpp -P
## Choose ONE CFLAGS line to uncomment:
## For 32-bit architectures
 CFLAGS=-O2 -D__NO_MATH_INLINES
## For 64-bit architectures
# CFLAGS=-O2 -D__NO_MATH_INLINES -DLONGWORDS
 LD=ld
# !!!
## Don't uncomment the next line unless you get errors about
## libraries not being found. Setting this path may interfere with
## the default (probably correct) operation of the loader, but some
## 64-bit architectures may need /usr/lib64 here.
## LDFLAGS=-L/usr/lib
 RANLIB=ranlib
 AR=ar
 YACC=bison -y
 LEX=flex -l
 LEXLIB=-lfl
## Some linuxes (Gentoo?) may require -lSM and -lICE as well.
 LIBS= $(LEXLIB) -lm
 TERMCAP=-lncurses
 TERMOPT=-DTERMIO -DDONT_USE_SIGIO
## end   Linux 1.2.x and up on Intel x86-based systems
```

Salve e feche

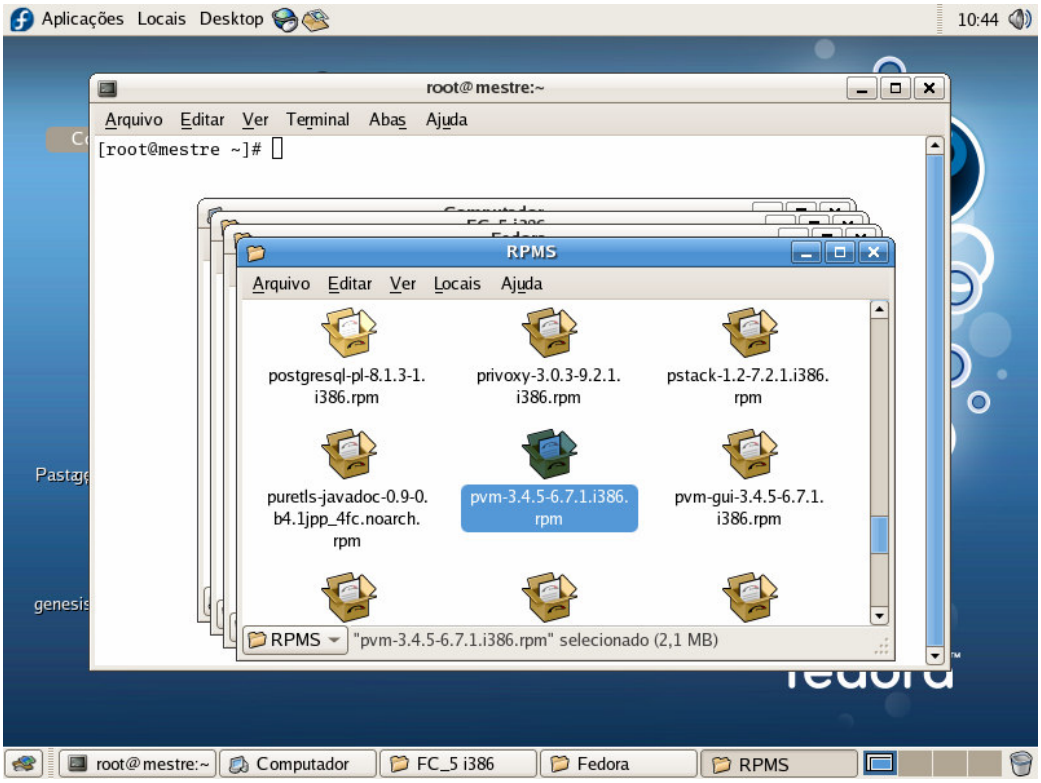
Digite make (o compilador vai executar as alterações feitas no arquivo Makefile)

Digite make install

Com isto o GENESIS já esta instalado.

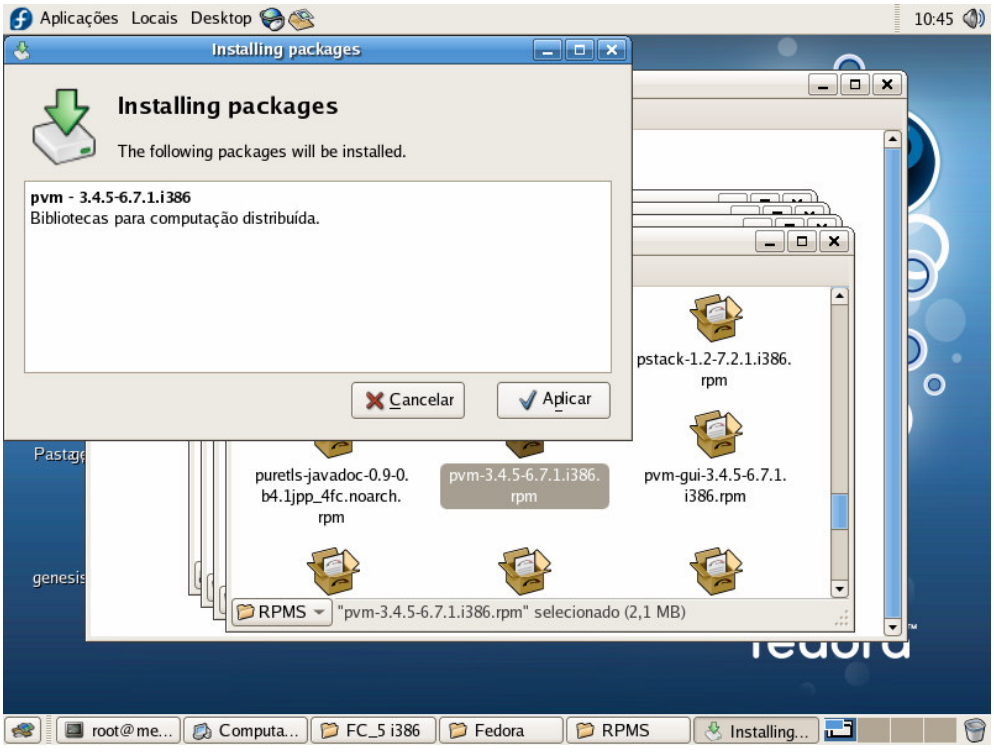
Coloque o CD Fedora Core 3

Abra atraves do navegador de arquivos a unidade de cd-rom, depois fedora e depois RPMS.



Execute o arquivo pvm-3.4.5.6.7.1.i386

Clique em aplicar para instalar o pacote.Com isto a biblioteca de troca de mensagens PVM será instalada.



Retornar para o terminal e digitar.

> *nano /root/.bashrc*
> *inserir as seguintes linhas*
> *PVM_ROOT=/usr/share/pvm3*
> *PVM_RSH=/usr/bin/ssh*

Criando a chave publica da maquina para posterior configuração Cluster.

> *cd*
> *ssh-keygen –b 1024 –t rsa*

Reiniciar a maquina digite
> *reboot*

Abrir novamente o terminal e digitar

> *cd /root/Desktop*

Instalando o PGenesis

> *tar –zxvf pgenesis-2.3.tar.gz*
> *mv genesis-2.3/pgenesis /usr/local/pgenesis*
> *cd /usr/local/genesis/pgenesis*
> *mv Makefile.dist Makefile*
> *nano Makefile*

Altere as seguintes linhas para

```
# ----------------------------------------------------------------------
# B. INSTALLATION AND MISC CONFIGURATION SETTINGS
# ----------------------------------------------------------------------

# the current working directory
CWD = /usr/local/genesis/pgenesis

# the location where serial genesis is installed.
GENESIS = /usr/local/genesis

# the directory to install pgenesis into
INST_DIR = /usr/local/genesis/pgenesis

# directory where pgenesis source tree root is
PARSRC_DIR = /usr/local/genesis/pgenesis

# ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
# System:         Linux 1.2.x and up on Intel x86-based, Xeon,
#         and AMD 64-bit systems.
# Compiler:       GCC
# ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

## 2000-05-23
## Termcap/ncurses issues: The shell library makes reference to the
## termcap library.  Some Linux distributions have an ncurses library
## which is includes termcap emulation.  GENESIS appears to work
## properly with the ncurses supplied with Red Hat Linux 5.1 and higher
```

```
## and Debian Linux (glibc2.1, egcs-2.91.66).  However, linking with
## ncurses is known to have resulted in core dumps in GENESIS in older
## Linux versions.
##
## If you encounter problems linking with the TERMCAP flags listed below
## or the GENESIS command line interface does not work, try the
## following alternatives:
##
## 1) TERMCAP = -ltermcap
##
## 2) (If you are using SuSE Linux)
##    TERMCAP = /usr/lib/termcap/libtermcap.a
##
## 3) (If you are using Red Hat Linux prior to version 6.0)
##    TERMCAP = /usr/lib/libtermcap.a
##

 MACHINE=Linux
 OS=BSD

 XINCLUDE=-I/usr/X11R6/include

## 64-bit machines probably need /usr/X11R6/lib64 here.
# XLIB=/usr/X11R6/lib

 CC=cc

## Old (and probably broken) gcc installations may need the full
## path to cpp (preferably NOT one in /lib). If there isn't a
## [link to] cpp in the same directory as 'cc', you should consider
## [re]installing a newer gcc.
 CPP=cpp -P
 CFLAGS=-O2 -D__NO_MATH_INLINES

## For 64-bit architectures
# CFLAGS=-O2 -D__NO_MATH_INLINES -DLONGWORDS

 LD=ld

## !!!
## Don't uncomment the next line unless you get errors about
## libraries not being found. Setting this path may interfere with
## the default (probably correct) operation of the loader, but some
## 64-bit architectures may need /usr/lib64 here.
 LDFLAGS=-L/usr/lib

 RANLIB=ranlib
 AR=ar
 CPLIB=cp

 YACC=bison -y
 LEX=flex -l
 LEXLIB=-lfl
 LIBS= $(LEXLIB) -lm

 TERMCAP=-lncurses
 TERMOPT=-DTERMIO -DDONT_USE_SIGIO

## end   Linux 1.2.x and up on Intel x86-based systems
```

Localize e descomente a linha

 # when the Makefile has been configured, uncomment this variable definition
EDITED = yes

Salvar e sair
Digite no diretório pgenesis

> *make install*

Neste momento o Pgenesis esta pronto para uso, para testar digite no console na mesma pasta da instalação.

> *./pgenesis*

O sistema ira executar e ficará aguardando outro comando digite.

> *quit*

Execute este mesmo processo nas outras maquinas que farão parte do Cluster.
Para usar o pvm vamos configurar o ssh para acesso sem senha entre as maquinas do cluster. Digite no console.

> *cd /root/.ssh*
> *cp id_rsa.pub authorized_keys*

Neste arquivo colocaremos todas as chaves publicas do cluster, no caso deste exemplo temos apenas duas maquinas a mestre e a no1. Antes de continuar, caso você esteja usando DHCP nada precisa ser feito, agora caso tenha atribuído o IP manualmente é necessário configurar o arquivo hosts com os números IPs das maquinas do cluster , digite.

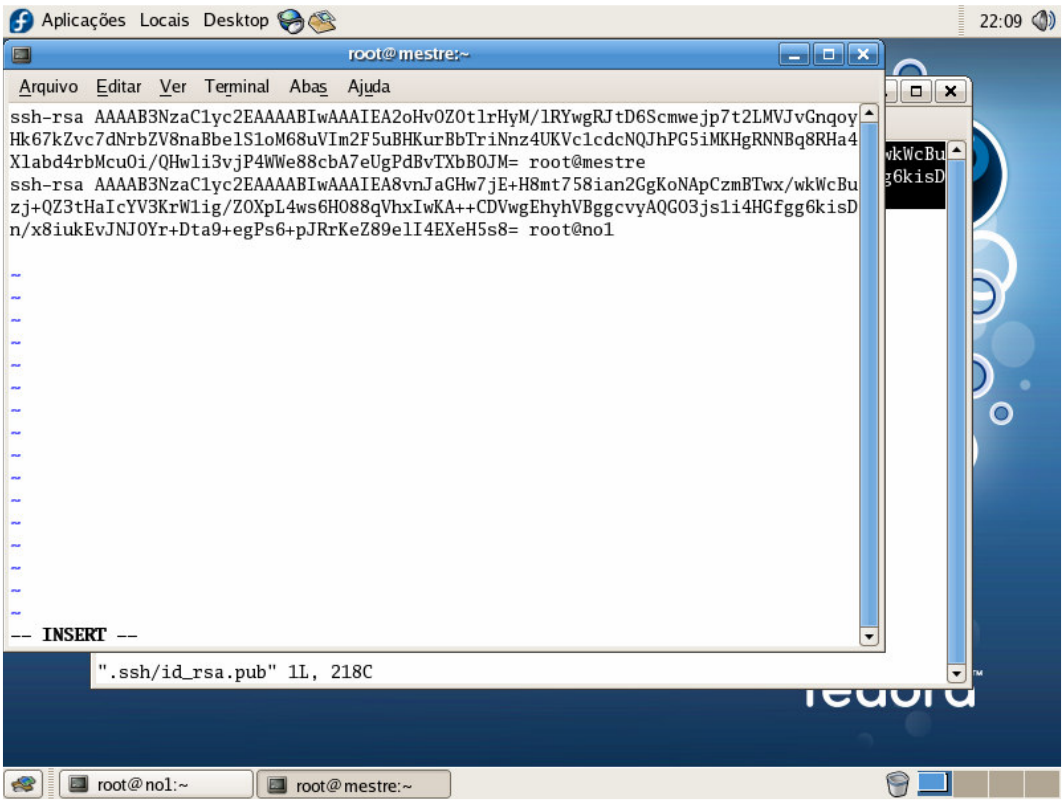
> *nano /etc/host*
mestre 192.168.0.101
no1 192.168.0.102

Isto em todas as maquinas no cluster.
Feito isto na maquina mestre digite.

> *scp root@no1:/root/.ssh/id_rsa.pub no1_id_rsa.pub* (digite a senha do usuário root)

Copie o conteúdo do arquivo no1_id_rsa.pub para o arquivo authorized_keys, em uma nova linha conforme tela abaixo.
Faça isto para toda maquina do cluster, feito isto acesse as outras maquinas no cluster, digitando
> *ssh no1 (digite a senha do usuário root)*
> *scp root@mestre:/root/.ssh/authorized_keys /root/.ssh/* (digite a senha do usuário root)

O arquivo final terá uma configuração como esta:



Agora entre as maquinas do Cluster o acesso ssh não pedirá mais senha, faça o teste entre as maquinas digitando.

> *ssh mestre*
> *ssh no1*

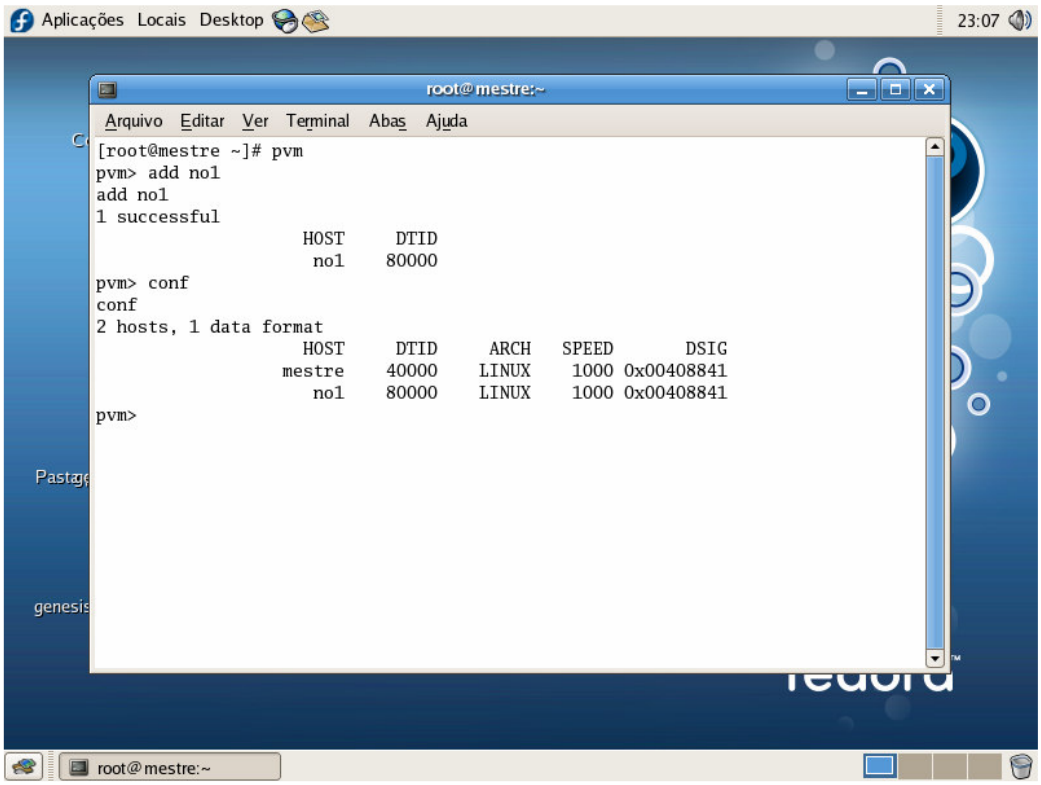
Vamos testar o Cluster, em qualquer maquina digite.

> *pvm*
> *conf*

Aparecerá as maquinas configuradas no cluster, adicione a outra maquina, digite

>*add no1*
>*conf*

Deverá aparecer a seguinte tela.



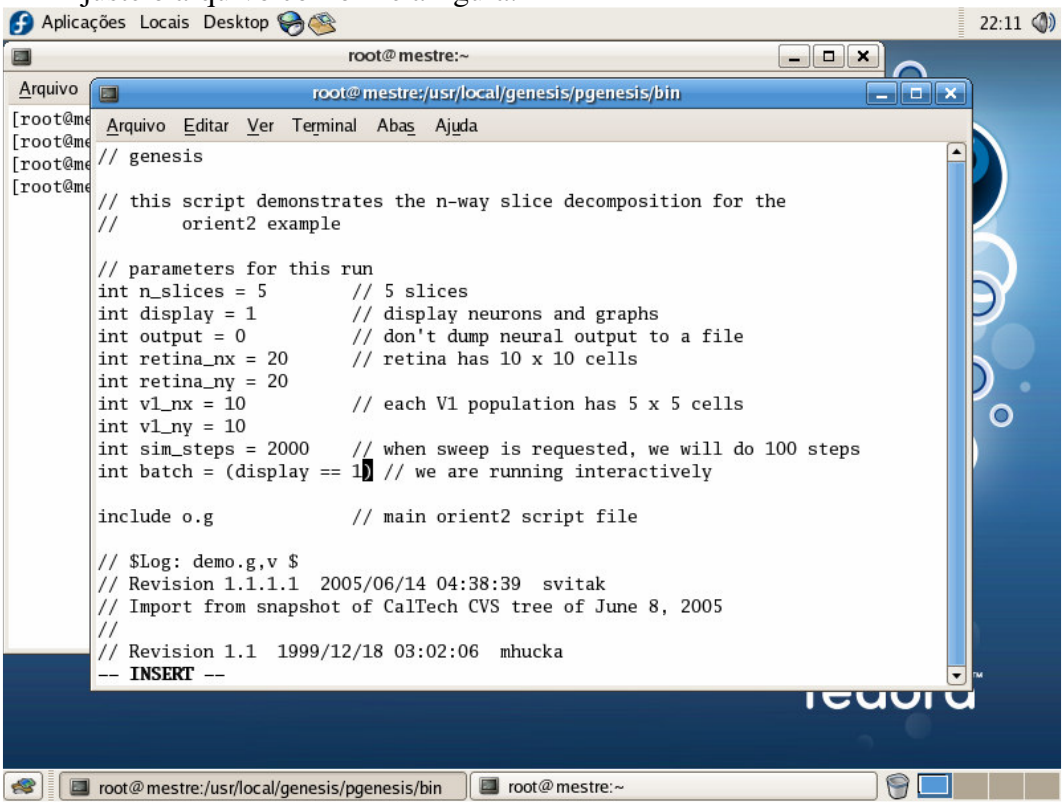
Isto demonstra que o pvm e o acesso sem senha esta funcionando, agora vamos fazer o teste com a simulação orient_tut do livro do GENESIS, digite.

>cp /usr/local/genesis/pgenesis/Scripts/orient2/ /usr/local/genesis/pgenesis/bin*
>cd /usr/local/genesis/pgenesis/bin

Vamos ajustar o arquivo de controle da simulação para uso da interface gráfica com uma rede de 20x20 células da retina , 10x10 V1 vertical e 10x10 V1 horizontal com 2000 passos e 5 slices, digite.

>nano demo.g

Ajuste o arquivo conforme a figura.



Você pode acompanhar a simulação nas maquinas através do comando top ou através do gnome-system-monitor, basta abrir um terminal em cada maquina do cluster e digitar.

> *top*
> *gnome-system-monitor*

Para rodar a simulação em duas maquinas vamos criar o arquivo de controle cluster assim.

> *nano cluster*

Digite dentro deste arquivo o nome das maquinas que serão usadas na simulação assim.

>*mestre*
>*no1*

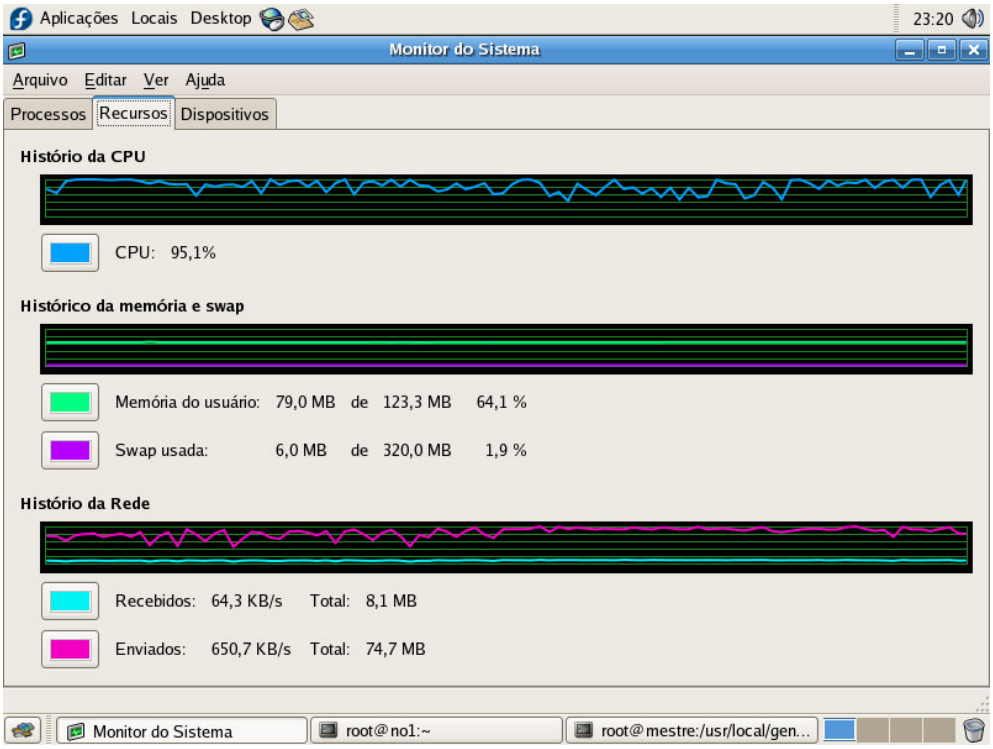
Salve e feche, para executar a simulação digite.

> *./pgenesis –config cluster demo.g*

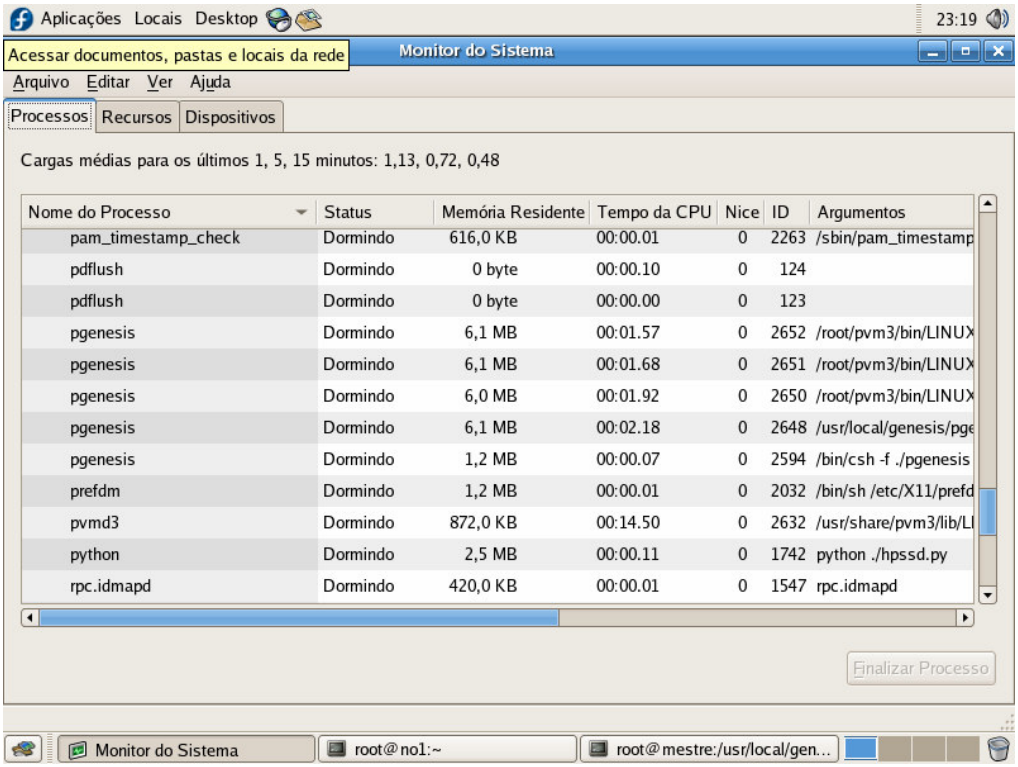
Caso queira saber o tempo da simulação digite

> *time ./pgenesis –config cluster demo.g*

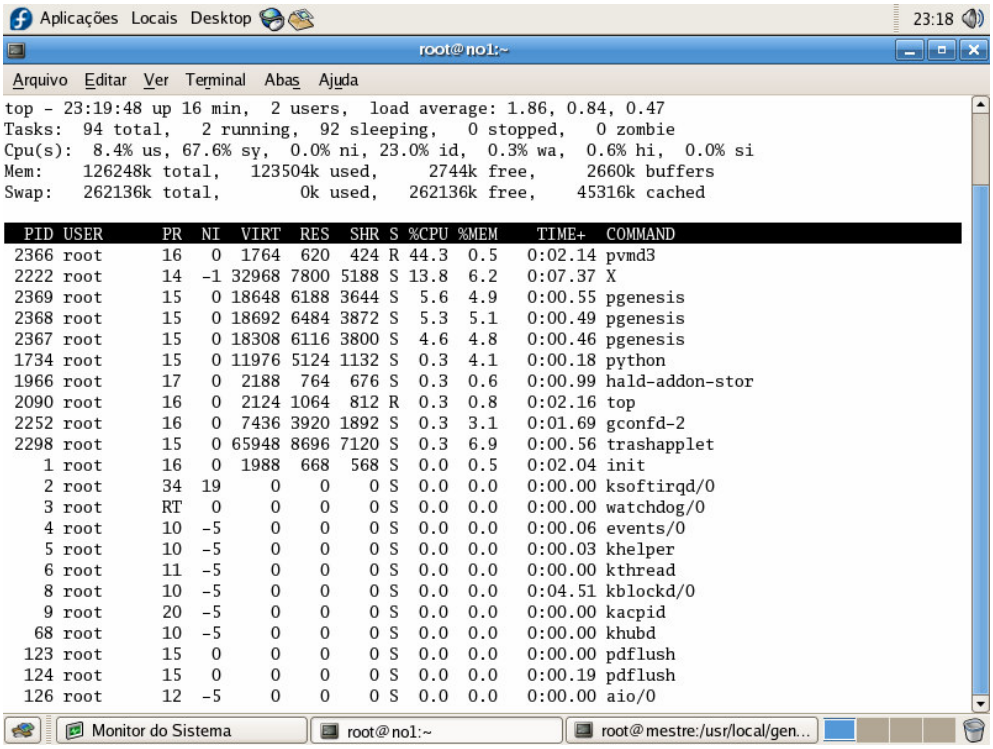
Exemplo de tela do gnome-system-monitor, mostrando uso do processador, uso da memória e trafego de rede.



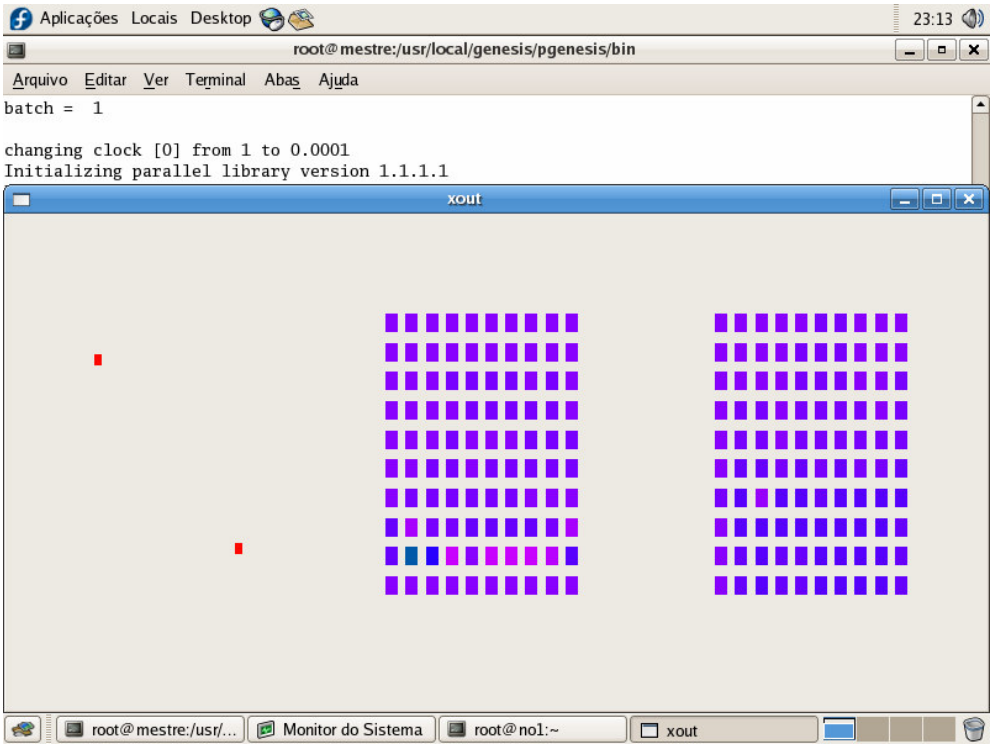
Tela do gnome-system-monitor mostrando os processos ativos, note os processos do pgenesis ativos na maquina mestre.



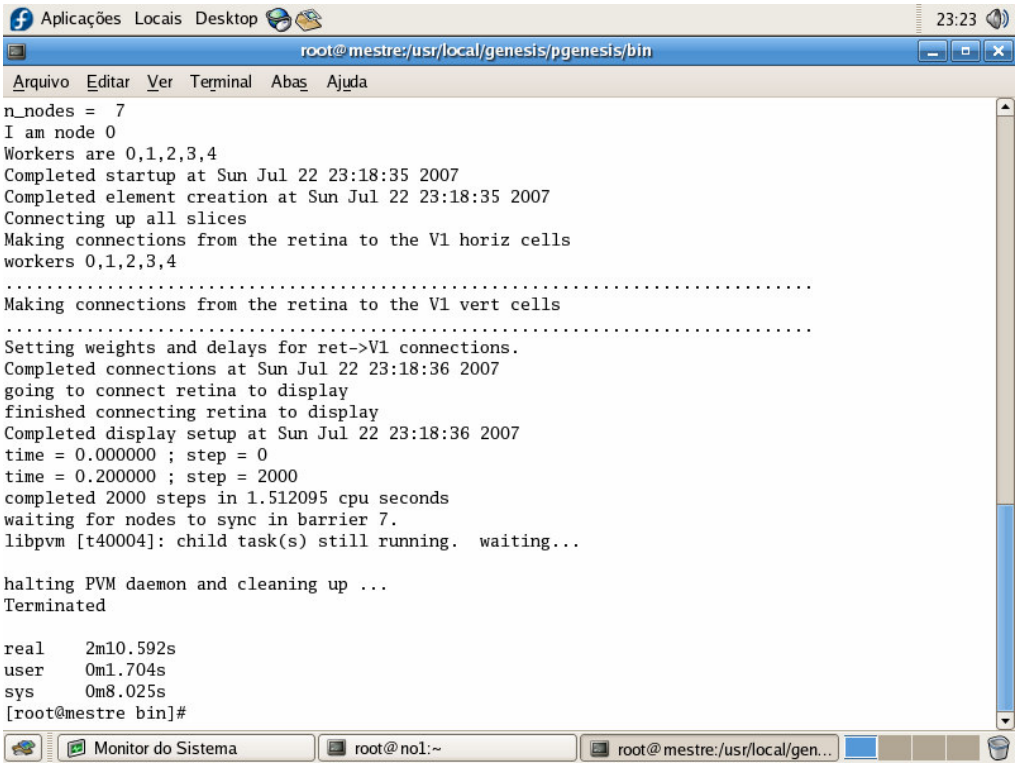
Idem a imagem anterior, mas desta vez usando o programa top mostrando os processos ativos na maquina no1.



Tela da interface gráfica Xodus do Pgenesis, à esquerda esta a matriz da retina com o gerador aleatório de sinais e a direita as duas matrizes de neurônios do tipo V1 com orientação vertical e horizontal.



Tela de resultado da simulação com tempo. Na configuração da simulação você pode optar por gerar um arquivo contendo as saídas da simulação.



Este manual trata de uma configuração genérica capaz de atender os usuários que possuem mais de um computador e gostaria de uma forma rápida configurá-los para uso do Pgenesis. Esta configuração acima foi o ponto de partida para o estudo realizado nesta dissertação para otimização do equipamento para as simulações, apenas de pouco otimizada possui um bom desempenho e escabilidade, basta ir adicionando mais nós repetindo os passos.