FUNDAÇÃO EURÍPIDES SOARES DA ROCHA
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL BENITO

ANÁLISE DE DESEMPENHO E IMPLEMENTAÇÃO DE UM SISTEMA CLUSTER BASEADO EM UMA REDE AD HOC

MARÍLIA
2011

FUNDAÇÃO EURÍPIDES SOARES DA ROCHA
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL BENITO

ANÁLISE DE DESEMPENHO E IMPLEMENTAÇÃO DE UM SISTEMA CLUSTER BASEADO EM UMA REDE AD HOC

Trabalho de curso apresentado ao curso de Bacharelado em Ciência da Computação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Ms. MAURÍCIO DUARTE

MARÍLIA
2011

Benito, Rafael.

Análise de desempenho e implementação de um sistema cluster baseado em uma rede Ad hoc. / Rafael Benito; Orientador: Dr. Maurício Duarte. Marília, SP: 2011

51 f .

Trabalho de Curso (Graduação em Bacharelado em Ciência da Computação) - Curso de Ciência da Computação, Fundação de Ensino “Eurípides Soares da Rocha”, mantenedora do Centro Universitário Eurípides de Marília - UNIVEM, Marília, 2009.

1. Sistemas Distribuídos. 2. Cluster. 3. Rede Wireless. 4. Ad Hoc. 5. MPI.

CDD

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Rafael Benito

ANÁLISE E IMPLEMENTAÇÃO DE UM SISTEMA CLUSTER BASEADO EM UMA REDE AD HOC

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 8,0 (oito)

Orientador: Mauricio Duarte

1°. Examinador: Emerson Alberto Marconato

2°. Examinador: Fábio Dacêncio Pereira

Marília, 28 de novembro de 2011.

AGRADECIMENTOS

Agradeço primeiramente à minha família pelo apoio e por serem exemplo de honestidade, respeito, educação e integridade, que foram determinantes para o meu crescimento e futuro.

Mãe, obrigado por ser um exemplo de vida e determinação. Lara, obrigado por estar sempre do meu lado, me ajudando e me dando conselhos e Ju, você é um exemplo de profissional e grande modelo pra mim, obrigado.

Ao meu orientador Prof. Ms. Maurício Duarte, pela competência, sabedoria e excelência na orientação deste trabalho.

Aos amigos da faculdade, principalmente aos amigos de pelada, Zé, Honda, Márcio e Jafa, pelo companheirismo, apoio e amizade.

Maria Rita Capobianco, que foi imprescindível para o desenvolvimento deste trabalho e esteve sempre ao meu lado quando precisei, muitíssimo obrigado.

E não menos importante, gostaria de agradecer a todos os amigos que me acompanharam durante toda a faculdade: Gustavo Mantovani, Guilherme Mantovani, Filipe Garcia Lopes, Rafael Ferrari, Bruno Conti, Rafael Oliveira, Enio, André Rubira, Gustavo Caliman, Kahue e Vinicius.

Epígrafe

“To Infinity… And Beyond.”

Buzz Lightyear

RESUMO

É cada vez mais frequentes pesquisas e estudos que buscam esclarecer como redes wireless podem se tornar mais viáveis e eficientes. Com a expansão do paralelismo e da computação distribuída tornou-se mais comum a utilização de arquiteturas distribuídas. Este projeto tem como objetivo a criação e análise de um sistema Cluster, baseado em uma rede wireless ad hoc, utilizando sistema operacional Windows 7 e ambiente de passagem de mensagem MPI (Message Passing Interface). Trata-se de uma pesquisa exploratória e prática, com revisão bibliográfica e a construção de uma arquitetura distribuída de alto desempenho. Um Algoritmo paralelo foi desenvolvido para validar o projeto. Os resultados demonstram que o sistema cluster implementado, embora funcional, não apresentou dados que indicassem viabilidade para se obter elevado desempenho devido às taxas de transmissão de redes wireless. Dessa forma, julga-se interessante que mais estudos sejam feitos nesta área uma vez que essas tecnologias são cada vez mais utilizadas e imprescindíveis no mundo da tecnologia de informação.

Palavras-chave: Sistemas Distribuídos; Cluster; Rede Wireless; Ad Hoc; MPI.

ABSTRACT

Nowadays, there has been an increase number of research and studies that aim to elucidate how wireless network may become more practicable and efficient. With the expansion of parallelism and distributed computing it has been taken for granted the use of distributed architecture. This work has its aim on the establishment and analysis of a cluster on ad hoc wireless network using Windows 7 as operational system and MPI (message passing interface). This research covers fact-finding data and praxis including bibliographic revision and the assembly of a high performance distributed architecture. A parallel algorithm was developed to validate this work. The outcomes reveal that the cluster system implemented, although functional, have not demonstrated data that suggest feasibility to obtain high performance due to wireless network transmission rates. Therefore, it is indeed thought to have more studies developed in this area once these technologies are more and more used and are essential in the world of information technology.

Key-words: Distributed Systems; Cluster; Wireless network; Ad Hoc; MPI.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS E GRÁFICOS

SUMÁRIO

INTRODUÇÃO.

Com a crescente substituição de redes cabeadas por redes wireless, tem-se tornado cada vez mais freqüente pesquisas e estudos que procuram elucidar como redes wireless podem se tornar mais viáveis e eficientes, tanto para a tecnologia da informação quanto para usuários comuns.

Com os avanços tecnológicos, sistemas sequenciais foram se tornando obsoletos, e os conceitos de paralelismo e computação distribuída cada vez mais utilizados. Segundo Stallings (2003), uma das áreas mais novas e promissoras de projeto de sistemas de computação é a de *Cluster*. A organização de cluster constitui uma alternativa para os multiprocessadores simétricos como abordagem para prover alto desempenho e alta disponibilidade.

Tendo em vista os benefícios de redes wireless e sistemas distribuídos *cluster,* é de se pensar que a união destas ferramentas resulte em algo relevante para a área de tecnologia da informação.

Esta pesquisa teve como objetivo montar um sistema distribuído *cluster* baseado em uma rede wireless ad hoc. Para tanto construiu-se uma aplicação distribuída, um algoritmo de ordenação shellsort, para ser executada em tal rede.

Com a finalidade de observar se este sistema é viável bem como o desempenho apresentado por ele utilizado a linguagem de programação C e ambiente de passagem de mensagem MPI (Message Passing Interface), realizou-se um estudo de caso.

Além disso, para melhor entendimento da pesquisa, procurou-se situar o leitor em relação a historicidade de sistemas computacionais através de uma pesquisa bibliográfica sobre o tema, para, posteriormente, analisar e discutir os dados obtidos.

Organização da Monografia

Capítulo 1 – Arquiteturas de Alto Nível: Apresenta um pequeno histórico sobre o surgimento do conceito de paralelismo e uma breve introdução sobre arquiteturas de alto desempenho.

Capítulo 2 – Sistemas Cluster: aborda definições, características, aplicações e tipos de cluster.

Capítulo 3 – Rede Wireless: apresenta características, aplicações e definições de rede wireless dando ênfase em redes ad hoc, rede utilizada no projeto.

Capítulo 4 – Metodologia e Projeto: apresenta o método utilizado para a pesquisa e um passo a passo de configuração do sistema *cluster*.

Capítulo 5 – Resultados: analisa e discorre sobre os resultados obtidos na pesquisa.

Capítulo 6 – Conclusão: aponta as conclusões feitas a partir da análise dos resultados e dos estudos.

CAPÍTULO 1 – ARQUITETURAS DE ALTO DESEMPENHO.

Segundo Omoto (2009), desde o aparecimento dos primeiros computadores, quando estes pertenciam somente a grandes empresas, buscou-se sempre a redução de custos e a melhora do desempenho computacional, dessa maneira, com o crescente avanço tecnológico, diversos paradigmas importantes surgiram como grande solução para resolver este desafio.

Este capítulo tem como objetivo situar o leitor quanto a historicidade de sistemas computacionais até o surgimento do paralelismo e da computação distribuída.

1.1 – Histórico

Em 1946 surgiu o primeiro computador, famoso ENIAC[1], fabricado nos EUA com a finalidade de realizar cálculos sobre armas de fogo. Era composto por 18 mil válvulas e pesava 30 toneladas, ficando em funcionamento até 1952, quando então se tornou obsoleto e foi desativado.

O ENIAC não tinha a capacidade de modificar ou carregar programas, já que não havia um "lugar" específico destinado ao armazenamento dos mesmos. Seria ideal um "lugar" (diga-se memória) para que essas instruções fossem armazenadas.

Na década de 40, John von Neumann e alguns colegas começaram o projeto de um novo computador, mais conhecido como IAS[2]. Junto com o IAS, surgiu um novo modelo computacional chamado de "modelo de Von-Neumann", esse novo modelo tornou-se a base para desenvolvimentos computacionais posteriores. O modelo é constituído por uma memória principal, uma unidade lógica aritmética, uma unidade de controle e dispositivos de entrada e saída como mostrado na figura 1. (Pitanga, 2008)

[1]Electronic Numerical Integrator and Computer (ENIAC).
[2] Institute for Advanced Study (IAS).

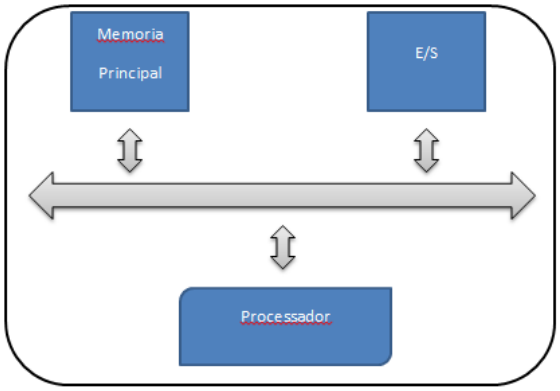

Figura 1. Arquitetura de Von-Neumann (Fonte Própria).

Para atender às necessidades da época, este modelo era ideal, porém, conforme foram surgindo necessidades maiores, foi-se modernizando e criando-se novas tecnologias. Por exemplo, dispositivos de entrada e saída eram lentos e tarefas não eram executadas paralelamente, dessa forma, foram surgindo avanços tecnológicos, até ser criado um novo conceito chamado de paralelismo.

Pode-se dizer que paralelismo significa agrupar vários dispositivos a fim de somar suas forças de processamento, suprindo assim, as necessidades que não vinham a ser resolvidas até então, fato esse que foi essencial para o desenvolvimento de computadores.

1.2 – Modelos de Arquiteturas de Computador – Paralelismo

Segundo Pitanga (2008), pode-se definir o paralelismo como uma técnica utilizada em grandes e complexas tarefas para obter resultados na forma mais rápida possível, dividindo-se então em tarefas pequenas que serão distribuídas em vários processadores para serem executadas simultaneamente.

Pode-se então concluir que o paralelismo significa fazer com que uma tarefa seja divida em várias outras tarefas menores e processadas em outros processadores simultaneamente.

Arquiteturas, ou modelos de arquiteturas de computador são definidos pelos dados que apresentam e pelos fluxos de instruções, sendo que é desta maneira que se dá a divisão das tarefas. Essas definições foram feitas por Michael Flynn em 1972 (PITANGA, 2008). Elas são divididas em quatro categorias:

- SISD -Single Instruction Single Data.

Única instrução, único fluxo de dados. Esse modelo de arquitetura é o mesmo descrito anteriormente como “Modelo de Von Neumann”. É o tipo de arquitetura mais simples pois funciona com apenas um dado por instrução. Um exemplo é mostrado na Figura 2.

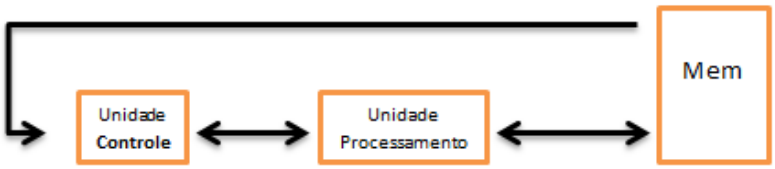

Figura 2. Exemplo de Arquitetura SISD - Von Neumann. (PITANGA, 2008).

- SIMD -Single Instruction Stream Multiple Data Stream.

Única instrução, múltiplos fluxo de dados. Funciona quando aplicada uma única instrução a um conjunto de dados de um vetor ou matriz. Pode ser chamada também de arquiteturas vetoriais.

As primeiras máquinas com esse modelo de arquitetura foram também os primeiros supercomputadores. Mais detalhes na figura 3.

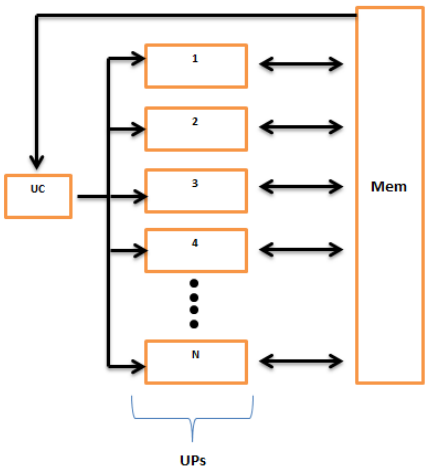

Figura 3. Exemplo de Arquitetura SIMD. (PITANGA, 2008).

- MISD - Multiple Instruction Stream Single Data Stream.

Múltiplas instruções, único fluxo de dados.Várias unidades de processamento executam diferentes operações sobre os mesmos dados. Um exemplo desse modelo é mostrado na figura 4.

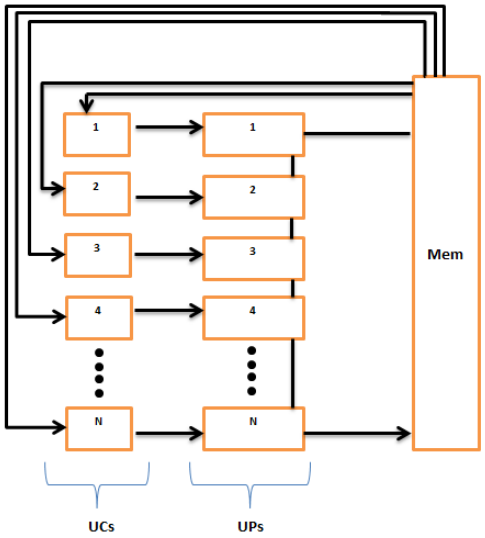

Figura 4. Exemplo Arquitetura MISD. (PITANGA, 2008).

- MIMD - Multiple Instruction Stream Multiple Data Stream.

Múltiplas instruções, múltiplos conjunto de dados.Todas as unidades de processamento têm uma unidade de controle própria, assim permite-se que várias instruções sejam executadas em paralelo pelas unidades de processamento. Mais detalhes na figura 5.

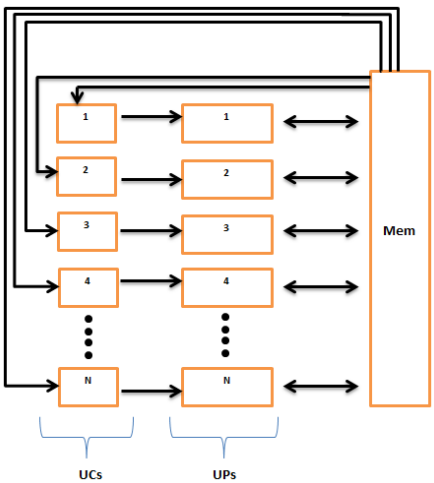

Figura 5. Exemplo de Arquitetura MIMD. (PITANGA, 2008).

1.3 – Sistemas Distribuídos.

Um sistema distribuído é um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente (TANENBAUM, 2007).

Assim, pode-se afirmar que um sistema distribuído nada mais é que vários dispositivos que se comunicam por algum tipo de rede, e que, na verdade, o usuário verá isso como apenas

um computador, para o usuário, a quantidade de processadores e memórias do sistema é transparente.

O próprio objetivo do sistema distribuído é criar essa ilusão de que há apenas um computador trabalhando, mas na verdade o que está ocorrendo é uma troca de mensagens e compartilhamento de memória.

Pode-se apontar como vantagens do sistema distribuído: maior poder computacional; separação física; menor atraso no acesso, entre outras.

A separação física impede que um computador já saiba do estado geral do sistema uma vez que a mensagem transmitida pode se perder devido a atrasos que ocorram ou em decorrência de alguma outra falha qualquer.

Pode-se afirmar ainda que sistemas com computação distribuída são uma alternativa de baixo custo em relação a supercomputadores,onde os sistemas são centralizados.

Segundo Omoto (2009), existem diversas vantagens e desvantagens de se utilizar sistemas distribuídos ao invés de centralizados. Entre as vantagens pode-se citar: *Economia* – aproveitamento de dispositivos ociosos, que interligados entre si é mais econômico que apenas um supercomputador-, *Potência de processamento* – entre um sistema distribuído pode ser maior do que em apenas um supercomputador -, *Confiabilidade* – mesmo que um equipamento apresente falhas ou problemas, é possível que o sistema continue executando uma aplicação-, *Crescimento incremental* – as inclusões de novas máquinas aumentam o poder computacional.

Já entre as desvantagens, pode-se citar: *Escassez de software* – Sistemas distribuídos necessitam de software próprio para rodar -, *Segurança* – Se houver muitos usuários pode haver problema quanto ao acesso do sistema-, *Rede de intercomunicação* – É possível que haja muitos usuários e ocorra problema de gargalo e saturação na rede.

1.4 – Supercomputadores

Supercomputadores são computadores de altíssimo poder/velocidade de processamento e grande capacidade e velocidade de acesso às memórias. Foram criados em 1960 por SeymorCray. (PITANGA, 2008). Eles contêm uma velocidade de processamento gigantesca, um enorme tamanho – que requer refrigeração adequada –, e há uma grande dificuldade de uso uma vez que esses não se tratam de um desktop doméstico. Esses

supercomputadores são usados para cálculos complexos e seus maiores usuários são grandes centros de pesquisas.

Há vários tipos de supercomputadores, tais como o PVP (Processadores Vetoriais Paralelos), SMP (Symetric Multiprocessors ou Multiprocessadores simétricos), DSM (Distributed Shared Memory ou Memória Compartilhada Distribuídas) e o MPP (Massively Parallel Processors ou Processadores Massivamente Paralelos). (PITANGA, 2008)

Existem alguns problemas ao se utilizar um supercomputador. Este é de alto custo, os softwares são caros e proprietários, há dificuldade de atualização, altos custo de manutenção e a dependência em relação aos fornecedores. Em decorrência desses fatores é que existe uma forte pesquisa sobre computação distribuída e cluster de computadores.

Acredita-se, de acordo com Pitanga (2008), que sistemas distribuídos, clusters, grids, entre outros tendem a aparecer mais nesse milênio.

1.5 – Considerações Finais

Arquiteturas de alto nível podem ser definidas como aquelas em que ocorre multi-processamento, podendo ser feito de maneira distribuída ou centralizada. Esta característica é extremamente útil quando se pretende solucionar problemas de processamento, uma vez que é possível agrupar o poder deste somando a força de vários dispositivos**.**

Dentre as arquiteturas de alto nível, são as dos supercomputadores que podem representar a melhor opção em situações que demandam alta velocidade e grande poder de processamento, uma vez que estas duas características estão presentes neste tipo de arquitetura. No entanto, sistemas distribuídos têm despertado interesse e estudos devido à praticidade e ao custo operacional que apresentam, uma vez que é possível criar ou, no mínimo, aproximar a velocidade de processamento às dos supercomputadores.

Neste projeto foi adotada a arquitetura computacional MIMD em multicomputadores.

CAPÍTULO 2 – SISTEMAS CLUSTER

Os Clusters de Computadores são máquinas construídas com utilização de dois ou mais microcomputadores comuns interligados por uma rede de interconexão, que trabalham juntos para resolver um problema. (RODRIGUES; GOUVÊA; PEREIRA, s.d).

Pode-se então dizer que um sistema cluster são vários dispositivos que, conectados por uma rede, agem em conjunto para atingir um objetivo em comum.

Uma das características sobre sistemas distribuídos é passar uma imagem transparente para o usuário, como se houvesse apenas um dispositivo trabalhando e não vários, como também foi ressaltado por Júnior e Freitas, (2005): “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.”.

Conclui-se então que Cluster são vários computadores ou dispositivos autônomos que operam em conjunto, interconectados por uma rede, para que o mesmo objetivo seja alcançado, porém passando uma imagem para o usuário de que apenas um computador esteja trabalhando.

A figura 6 mostra o Cluster da NASA, Columbia(2004).

Figura 6. Cluster Columbia da Nasa (Fonte: NASA).

2.1 – Funcionamento

Um cluster consiste na coleção de nós computacionais que são controlados e acessados por um nó denominado Nó Mestre (TANENBAUM, 2006).

A figura 7 mostra um exemplo de um sistema cluster.

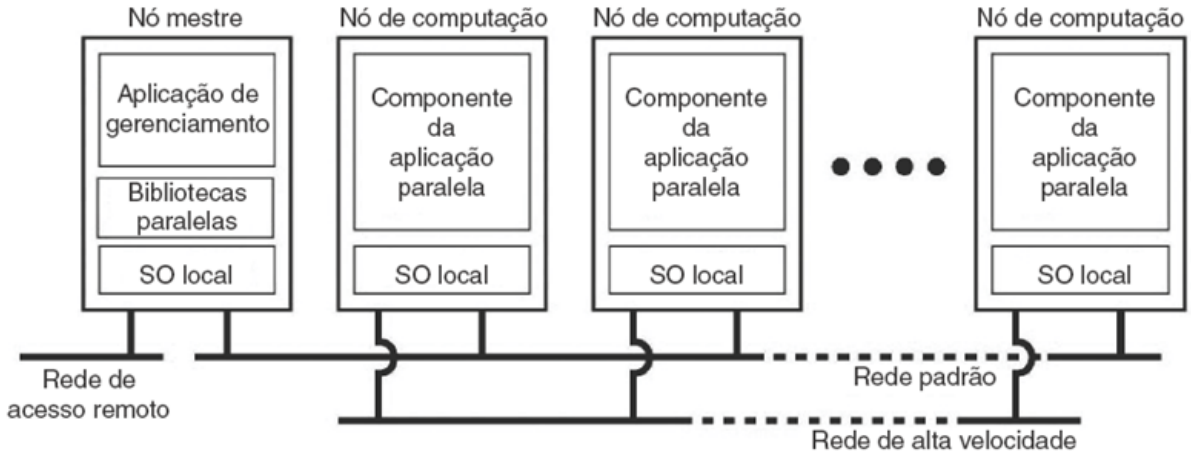

Figura 7. Exemplo de um Sistema Cluster (Fonte: Tanenbaum, 2006)

A Figura 7 mostra um exemplo de funcionamento de um Cluster, onde o nó mestre é quem gerencia e distribui a carga entre os nós de computação através de uma Rede. A figura 7 mostra também que o Nó mestre é quem contém a aplicação de gerência das bibliotecas paralelas e nos nós escravos contêm apenas os componentes da aplicação paralela.

Toda a troca de informações entre o nó mestre e os outros nós do sistema é feita através da interconexão de rede, e uma rede separada é feita para conectar o usuário ao sistema cluster. Pode-se usar a mesma rede para fazer a conexão, porém, se houver muito tráfego pode ocorrer perda de desempenho.

2.1.2 – Características de um Cluster

Um sistema cluster possui várias características, porém a mais importante, como dito anteriormente é a transparência, ou seja, o sistema deve parecer centralizado em apenas um dispositivo para o usuário.

Outras características, não menos importantes como a transparência são apontadas por (Omoto,2009):

- A *configuração dos nós*, onde os dispositivos podem ser multiprocessados ou uniprocessados.

- Quanto à escalabilidade do sistema ela pode ser:

a) *Absoluta*: possibilidade de construção de um sistema onde a capacidade de processamento supera a de dispositivos individuais.
b) *Incremental*: um sistema cluster pode ser formatado para ser expandido, ou seja, incrementar a quantidade de nós no sistema.

- A *disponibilidade*, caso ocorra algum tipo de erro em um nó, os outros nós continuam trabalhando para manter o sistema disponível.

- O *custo/beneficio*, dado a facilidade da construção de um sistema cluster, seja feito com dispositivos antigos ou novos, é possível a construção de sistemas com alto poder computacional e com custo menor se comparados a supercomputadores ou quaisquer outros computadores de grande porte.

- E a utilização de hardware aberto, softwares livres e sem a necessidade de licenças de uso torna o sistema *independente de fornecedores.*

2.1.3 – Aplicações

Pode-se concluir que clusters são bem aplicados diante de situações que exigem alto poder computacional, assim “em qualquer local onde tivermos um grande problema computacional em que o processamento paralelo seja considerado uma vantagem, pode ser indicado à utilização em um cluster.” (PITANGA, 2008).

Pitanga(2008) também cita exemplos de aplicações onde esses sistemas podem ser utilizados.São eles:

- Servidores da Internet: clusters podem dividir a carga de trabalho para otimizar os resultados quanto a resposta do servidor.

- Segurança: por oferecer grande poder de processamento, o cluster pode ser aplicado para identificar se houve quebra na segurança e apontar possíveis soluções para o ocorrido.

- Banco de Dados: *querrys* que demorariam a retornar algum resultado de um banco de dados muito grande, retornam de forma mais rápida se usado sistema cluster para busca.

- Computação Gráfica: tempo de processamento normalmente é uma limitação para qualidade de processos na computação gráfica, a utilização de cluster pode, por exemplo, aperfeiçoar o tempo de renderização de imagem.

- Análise de Elementos Finitos: cálculo de barragem, navios, veículos espaciais, edifícios entre outros que necessitam de grandes cálculos que podem levar muito tempo.

- Engenharia genética: projeto Genoma.

- Inteligência artificial e automação: reconhecimento de voz, processamento de imagens, reconhecimento de padrões.

- Previsão do tempo: em computadores seqüenciais esse processo pode ser lento e ter pouca precisão.

- Pesquisa militares: projeto de armas nucleares e simulações de efeitos das armas, geração automática de mapas, acompanhamento de submarinos submersos.

2.2 – Tipos de Cluster

Clusters normalmente são classificados quanto a sua aplicação final. Existem vários tipos de cluster, porém, pode-se mostrar as classificações mais conhecidas em três categorias

básicas: *Alta Disponibilidade (HA - High Availability), Alta Desempenho de Computação (High Performance Computing)* e *Balanceamento de Carga (LoadBalancing).*

2.2.1 – Alta Disponibilidade (High Availability)

Cluster de alta disponibilidade tem o objetivo principal de manter um sistema disponível o tempo todo sem falhas de indisponibilidade, assim "a alta disponibilidade tem como função essencial deixar um sistema no ar vinte e quatro horas por dia, sete dias por semana, ou que não suportem paradas de meia hora ou alguns minutos." (PITANGA, 2003).

Pode-se concluir então que, nos clusters de alta disponibilidade, os nós operam em conjunto para que o serviço esteja sempre ativo, caracterizando assim uma alta disponibilidade.

A figura 8 mostra um exemplo de um Cluster de Alta Disponibilidade

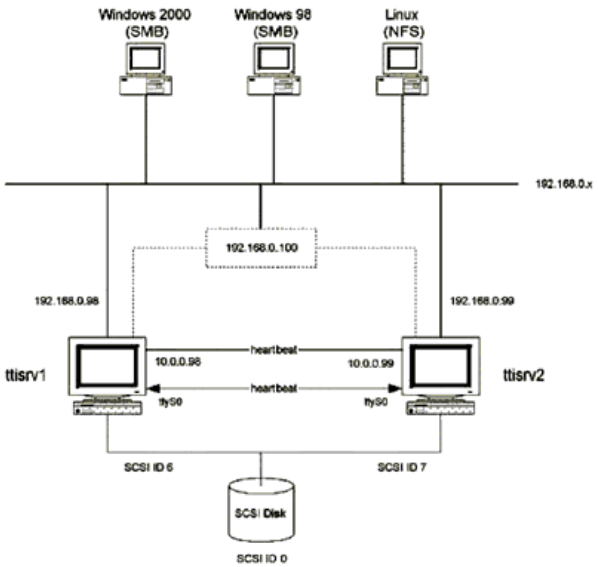

Figura 8. Cluster de Alta Disponibilidade.
(Fonte: http://www.clubedohardware.com.br/artigos/153).

A Grande justificativa de utilizar cluster de alta disponibilidade surgiu de empresas que necessitavam que seu serviço e/ou aplicação tivessem disponibilidade total. Se esta não ocorresse, por exemplo, em uma loja virtual, clientes dessa empresa provavelmente teriam a impressão de má qualidade de serviço.

Para Pitanga (2003), de maneira geral, um servidor de boa qualidade apresenta uma disponibilidade de 99,5%, e soluções em cluster apresentam disponibilidade de 99,99%. Conclui-se então que serviços que utilizam cluster para alta disponibilidade têm mais chances de atingir o objetivo de prover serviços sem interrupções.

2.2.2 – Alto Desempenho de Computação (*High Performance Computing*)

"Esta categoria tem como foco, o desenvolvimento de soluções cujo objetivo principal é o ganho de desempenho através do agrupamento de sistemas de processamento, auxiliado por algoritmos de processamento paralelo." (OMOTO, 2009).

Sendo assim, pode-se concluir que cluster de alto desempenho tem como objetivo a soma de poder processamento para resoluções de problemas cuja carga a ser processada é elevada.

HPC funciona paralelizando a aplicação a ser processada, ou seja, dividindo a aplicação em tarefas menores e a distribuindo entre os nós do sistema, cada nó processa sua parte e devolve o resultado. Dessa maneira, tarefas muito grandes são processadas mais rapidamente e sem custo excessivo se comparados a supercomputadores.

Um exemplo de cluster voltado para o desempenho é o cluster *beowulf*, como mostrado na figura 9:

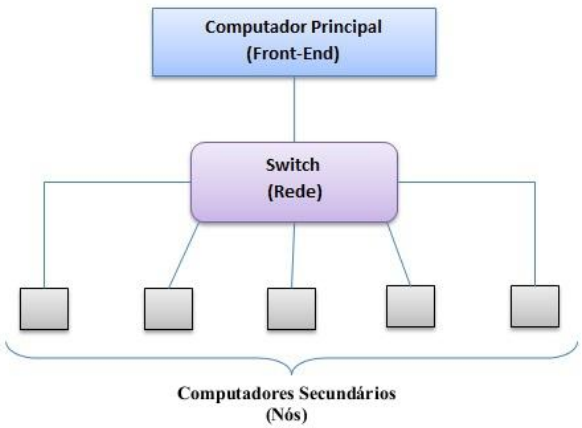

Figura 9. Exemplo cluster *Beowulf* (Fonte Própria).

Um cluster *beowulf* é baseado em uma infraestrutura de hardware comum, sendo assim, qualquer tipo de computador pode preencher o sistema, mesmo que tal computador não

tenha um processamento elevado, já que o poder de processamento será somado. Baseia-se também em uma rede privada e softwares livres.

2.2.3 – Balanceamento de Carga (*Loading Balancing*)

Nessa abordagem de cluster, baseado em balanceamento de carga, o sistema tem a função de distribuir o trafego entre as máquinas do sistema.

O cluster de Balanceamento de Carga tem como propósito a distribuição igualitária de processos ao longo dos nós do agrupamento de computadores. Evidentemente algoritmos de escalonamento de processos se fazem necessários (PITANGA, 2002 apud OMOTO, 2009).

Cada nó do sistema é interligado e cada solicitação requerida a ele é distribuída de forma equilibrada entre os nós, assim toda a carga é balanceada entre o sistema. Vale ressaltar que esse tipo de cluster utiliza um algoritmo de escalonamento para que o sistema balanceie a carga corretamente.

Sistemas cluster de balanceamento de carga são muito utilizados em situações onde há problema de inúmeras solicitações em tempo real, solucionando assim, cargas excessivas que poderiam prejudicar o sistema.

Algoritmos de escalonamento para esse tipo de sistemas são definidos segundo (BUYYA, 1999 apud OMOTO, 2009):

- **Least Connection**

  Nesse algoritmo de balanceamento, a preferência de destino da carga é para nós onde o número de solicitações é menor.

- **Round Robin**

  Encaminha as cargas de forma sequencial, ou seja, encaminhando a carga para o primeiro nó na sequência que esteja disponível.

- **Weighted Fair**

  A carga é encaminhada preferencialmente para nós de maior poder computacional, ou que tem maior capacidade de carga.

Um exemplo de sistema cluster de balanceamento de carga é mostrado na figura 10.

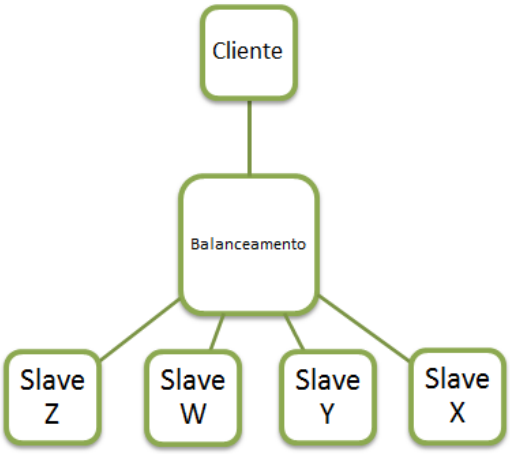

Figura 10. Diagrama de um Cluster Balanceador de Carga. (Fonte:Própria)

2.3 – Breve Introdução à Ambientes de Programação Paralela.

Para a implementação de um programa paralelo em um sistema distribuído é necessário que um ambiente de troca de mensagem seja definido e configurado, isso se dá pela necessidade de troca de informações entre nós do sistema. São esses ambientes os responsáveis pelo gerenciamento e da computação paralela, tais como a comunicação, sincronização e criação de processos.

Os ambientes de passagem de mensagens são ambientes de programação paralela para memória distribuída portátil, pois permitem o transporte de programas paralelos entre diferentes arquiteturas e sistemas distribuídos de maneira transparente (SANTANA, 1997 apud OMOTO, 2009).

As mais conhecidas e utilizadas bibliotecas de passagem de mensagem são o MPI (*MessagePassing Interface*) e o PVM (*Parallel virtual Machine*) assim como mostra Santana, (1997) apud Omoto (2009): “Os ambientes MPI e PVM obtiveram uma grande aceitação por proporcionar o desenvolvimento de aplicações paralelas a um custo relativamente baixo em relação à arquiteturas paralelas.”.

É interessante citar também que após 1997 Java RMI (Remote Method Invocation) obteve bastante destaque.[3]

Para o desenvolvimento deste projeto será dada ênfase para o MPI, o qual foi utilizado para a implementação do mesmo.

2.3.1 – Breve introdução à MPI

A sigla MPI quer dizer *MessagePassing Interface* ou em português: Interface de Passagem de Mensagem, é uma biblioteca desenvolvida para ser padrão em memória distribuída, em passagem de mensagem e em computação paralela. Passagem de mensagem portável para toda e qualquer arquitetura, oferece aproximadamente 125 funções para programação e algumas ferramentas para analisar o desempenho. Pode ser utilizada nas linguagens C e FORTRAN. Vale também ressaltar que a plataforma alvo para o MPI são os ambientes de memória distribuída, máquinas paralelas e sistemas cluster.

2.3.2 – Características do MPI

Algumas características do MPI podem ser apontadas (OMOTO, 2009):

- **Eficiência:** O MPI foi projetado para executar com eficiência e em diferentes arquiteturas, especificando somente o funcionamento lógico e deixando a implementação aberta, sendo assim, desenvolvedores podem aperfeiçoar o código de acordo com a sua necessidade.

- **Facilidade:** Interface com facilidade de uso.

- **Portabilidade:** É compatível com sistemas de memória distribuída ou compartilhada ou até mesmo uma rede de servidores.

[3] Informação Oral obtida em novembro de 2011 pelo Prof. Ms. Mauricio Duarte.

- **Transparência:** O programa pode ser executado em arquiteturas heterogêneas sem mudanças significativas nos resultados.

- **Segurança:** Provê uma comunicação confiável.

- **Escalabilidade:** Permite o crescimento em escala.

2.4 – Por que cluster?

Como descrito anteriormente, cluster são sistemas que podem ser construídos utilizando máquinas não tão poderosas como supercomputadores, mas que em conjunto, possam processar cargas excessivas que seriam impossíveis de processar em máquinas de arquitetura centralizadas.

Assim conclui-se que esses sistemas são altamente viáveis, uma vez que o paralelismo vem crescendo, a implementação não é complexa e seu custo não é excessivo.

CAPÍTULO 3 – REDE WIRELESS

"O mundo sem fio tem sido significantemente desenvolvido com produtos de redes de Rádio Freqüência (RF). O objetivo desses produtos é simplificar e transformar o modo de comunicação e condução de nossas vidas."(MOHAMMAD, 2003 apud CORRÊA et al. (s.d)).

"Essas redes buscam primeiramente, num âmbito mais abrangente, ocupar a lacuna que impossibilitava a implementação de redes cabeadas por questões de patrimônio público tombado, bem como pelo auto custo de se estabelecer uma rede fixa temporária."(CORRÊA et al. s.d)

"Nos últimos cinco anos (2002) o mundo tem se tornado mais móvel. Como resultado, tradicionais meios de rede vem provando serem inadequados para enfrentar os desafios colocados pelo novo estilo de vida coletivo".[4] (GAST, 2002).

A rede sem fio está presente no cotidiano através de celulares, comunicação TV-Satélite, rádio amadores, entre outros, porém o seu uso mais aparente é feito em rede de computadores.

Redes e dispositivos sem fio estão cada vez mais populares uma vez que o acesso à informação e à comunicação pode ser feito a qualquer hora e lugar.(MOHAPRATA, KRISHNAMURTHY, 2004).

De acordo com Mohaprata e Krishnamurthy (2004),o uso de redes e dispositivos sem fio vem ficando popular uma vez que o uso das redes wireless são mais práticas e não necessitam de infraestrutura complexa como a de uma rede cabeada. Dessa forma se torna possível o acesso à informação e comunicação instantânea.

Segundo Tanenbaum (2003), pode-se utilizar o critério de escala para classificar uma rede, como mostrado na figura 11.

[4] Tradução própria

Interprocessador Distância	Localização dos processadores	Exemplo
1 metro	Metro Quadrado	Área de Rede Pessoal
10 metros	Cômodo	Área de Rede Local
100 metros	Construção	
1 Km	Campus	
10 Km	Cidade	Área de Rede Metropolitana
100 km	País	Área de Rede Extensa
1 000 km	Continente	
10 000 km	Planeta	Internet

Figura 11. Classificação de processadores interconectados por escala. (Fonte: TANENBAUM, 2003).

Para as redes sem fio, Tanenbaum (2003), faz uma divisão em três categorias – Interconexão de Sistemas, LANs sem Fio, WANs sem Fio.

- Interconexão de Sistemas ou WPAN – Wireless Personal Area Network, utilizada normalmente na interação de dispositivos simples, e de curto alcance, tal como controle de carro ou bluetooth. (COSTA, [s.d.]).
- LANs sem Fio ou WLAN – Wireless Local Area Network, rede local sem fio. Uma rede local sem fio pode ser considerada uma extensão de uma rede local cabeada, que converte os pacotes de dados em ondas de rádio e os envia para um ponto de acesso que serve como uma conexão para uma rede local.(CORRÊA et al. s.d.).A Figura12mostra um exemplo de uma WLAN.
- WANs Sem Fio – Wireless Wide Area Network, a abrangência desse tipo é a de maior alcance, normalmente utilizada por operadoras de celular. CDMA[5] e GSM[6] são exemplos desse tipo de rede sem fio.

Dentro da Rede Wireless existem duas principais topologias, ou seja, dois principais modos de se montar essa rede. São elas:

[5] CDMA - CodeDivisionMultiple Access, ou Acesso Múltiplo por Divisão de Código.
[6] GSM - Global System for Mobile Communications, ou Sistema Global para Comunicações Móveis.

a) Infraestruturadas, as redes com infraestrutura são aquelas em que o nó móvel está em contato direto com uma Estação de Suporte à Mobilidade (ESM), também conhecida como ponto de acesso (AP). .(CORRÊA et al. s.d).

   A aplicabilidade dessa topologia de rede pode ser empregada em locais onde já existe uma rede local, mas haja a necessidade de oferecer flexibilidade e mobilidade para algumas estações. (CORRÊA et al. s.d).

b) Ad-Hoc, onde não existe a presença de nenhum dispositivo concentrador, os próprios dispositivos transmitem entre si.Elas são construídas dinamicamente, como resultado da detecção mútua de dois ou mais equipamentos móveis com interfaces sem fio na mesma vizinhança. (COULOURIS; DOLLIMORE; KINDBERG, 2007).

   Neste projeto a importância maior está sobre essa topologia Ad Hoc.

3.1 – Redes Ad Hoc.

As redes móveis ad hoc são a fronteiras finais em uma comunicação sem fio, e o fator chave na evolução desse paradigma (SANTINI, 2005). Essa abordagem de comunicação permite que os nodos se comuniquem diretamente entre si ou através de múltiplos saltos dentro da rede usando receptores e transmissores sem fio, sem a necessidade de uma infraestrutura fixa (CAMPBELL, 2003 apud CORRÊA et al. s.d).

Como afirmado por Corrêa et al. (s.d.), toda a transmissão entre os dispositivos é feita sem nenhuma infraestrutura, e isso faz com que os próprios dispositivos ajam como concentradores.

3.1.1 – Características das Redes Ad Hoc

Mohaprata e Krishnamurthy (2004), descrevem como características de redes Ad Hoc:

- **Mobilidade:**

  Configura-se como uma das principais características das redes ad hoc, uma vez que possibilita de maneira prática e rápida o reposicionamento e mobilidade dos nós. Em áreas com pouca infraestrutura, cujo objetivo é obter determinado resultado com um rápido desempenho, é necessário criar times "teams/swarms" que coordenem entre si para executar uma força tarefa ou missão. O modelo de mobilidade escolhida irá influenciar de forma crucial a decisão quanto ao esquema de roteamento que, consequentemente irá determinar o desempenho da tarefa a ser feita. Pode-se, assim, escolher entre mobilidades individuais aleatórias, mobilidades em grupo, movimento através de rotas pré-planejadas e outras.

- **Multi-hopping:**

  Dizer que uma rede possui características "múltiplo saltos" significa afirmar que os trajetos percorridos desde a origem até o destino final atravessam diversos nós. As redes ad hoc geralmente utilizam o recurso dos "múltiplos saltos" para solucionar obstáculos, conservar energia e reutilizar frequência (*spectrum reuse*). As operações secretas em campos de batalha, visando evitar e/ou dificultar que inimigos descubram seus conteúdos, também utilizam uma sequência de "saltos".

- **Auto-organização:**

  A rede ad hoc deve determinar automaticamente seus parâmetros de funcionamento, incluindo: endereçamento, roteamento, agrupamento, identificação de posicionamento, controle de potência, entre outras. Em alguns casos, nós especiais podem se coordenar e distribuir em determinada área geográfica para que haja cobertura de territórios desconectados.

- **Escalabilidade:**

  Em alguns casos a rede ad hoc pode expandir em centenas de milhares de nós em pouco tempo dependendo das aplicações a que são submetidas. A escalabilidade de rede sem fio infraestruturada é manipulada por construção hierárquica ou pode também ser manipulada utilizando um ip-móvel ou técnicas de *handoff*. Porém, como as redes Ad Hoc possuem muita mobilidade, se esta for pura, não irá tolerar ser manipulada por um ip-móvel ou qualquer hierarquia fixa. Levando estes fatores em

consideração, pode-se concluir que é problemático e trabalhoso utilizar mobilidade em alta escala de redes ad hoc.

- **Segurança**

  Os desafios de segurança nas redes sem fio já são bem conhecidos, e, como já se sabe, estas redes podem ser vistas e/ou ouvidas por intrusos. Dessa forma, muitos trabalhos relacionados à segurança estão sendo desenvolvidos para ambientes em rede sem fio infraestruturada e, por consequência, se estendendo para as redes ad hoc, uma vez que estas são mais vulneráveis do que aquelas. Porém, uma das dificuldades encontradas nas redes ad hoc é em relação aos protocolos, muito mais difícil de detectar do que nas redes infraestruturadas. Além disso, ataques passivos às redes ad hoc podem ser mais maliciosos/perigosos do que os ativos.

- **Restrições de Energia**

  A maioria dos nós ad hoc, tais como laptops e PDAs[7], possuem autonomia limitada e incapacidade de gerar sua própria energia. Dessa forma, é imprescindível que haja um eficiente sistema de fornecimento ou geração de energia para o sucesso e duração da missão.

3.1.2 – Aplicações

Uma vez que essas redes não requerem de infraestrutura, as aplicações destas se dão através de situações onde se necessita de uma rede rápida de montar e/ou não há outra maneira de se tê-la sem infraestrutura, ou até mesmo se houver uma rede infraestruturada que não possa ser utilizada por algum motivo em especifico como, por exemplo, a segurança. Algumas dessas aplicações serão descritas a seguir. (CORRÊA, et al. s.d).

- **Campo de Batalha em Operações Militares**

  As redes ad-hoc tiveram origem dentro do campo de batalha, a comunicação entre soldados é feita através desse tipo de rede. Além da comunicação entre soldados,

[7] PDA - Personal digital assistants, assistente pessoal digital ou palmtop.

pode-se haver a comunicação entre os soldados e aviões não tripulados (UAVs) que os ajudam quanto a localizações, terreno e entre outras informações.

- **Veículos não tripulados**

  Alguns dos aplicativos populares das redes ad hoc exigem componentes robóticos não tripulados. Todos os nós de uma rede genérica são, obviamente, capazes de criar redes autonomamente. Quando se adiciona também uma mobilidade autônoma, surgem interessantes oportunidades de se associar networking com deslocamento. Um exemplo são os chamados UAVs – Unmanned Airborne Vehicles, que são capazes de manter uma rede ad hoc de grande terreno interconectada apesar dos muitos obstáculos físicos, das irregularidades do canal de propagação e *enemyjamming*. Além disso, as UAVs conseguem conciliar uma performance relativamente restrita através de um posicionamento e *beaming* de antena adequados.

- **Sensores**

  O uso de sensores em determinado ambiente tem se mostrado popular a medida que permite coletar processos, operar de forma automática além de transmitir informações sem o suporte de uma infra-estrutura.

- **Conexão à Internet**

  Há algumas vantagens em expandir a rede infrestruturada com apêndices de ad hoc. Por exemplo, o alcance de uma LAN sem fio doméstica pode ser aumentada, se preciso – para a garagem, casa de um vizinho e outras – com roteadores portáteis. Estas extensões têm se tornado extremamente importantes e são significativas na medida em que surgem aplicações comerciais das mesmas. A integração de um protocolo ad hoc com padrões infra estruturados está, portanto, se tornando uma questão de grande relevância atualmente.

- **Assistência a Desastre**

  Redes ad-hoc se moldam muito bem nesse tipo de caso, uma vez que sua aplicação não necessita de infraestrutura elas podem ser altamente flexíveis para a troca de informações com sobreviventes ou até mesmo com o próprio grupo de resgate.

3.2 – Considerações Finais

Após a análise das características e aplicabilidades tanto das redes infraestruturadas quanto das ad hoc pode-se pontuar vantagens e desvantagens de cada. Nas redes infraestruturadas a segurança e o alcance são fatores positivos embora seja necessário o uso de concentradores. Já nas redes ad hoc, embora a segurança seja prejudicada, existe a vantagem de não precisar de nenhum concentrador, já que os próprios dispositivos irão agir como concentradores.

Dessa forma pode-se afirmar que redes ad hoc são muito bem utilizadas onde a aplicação de redes infraestruturadas não podem ser feitas.

Algumas das características apontadas como vantagens da implantação das redes ad hoc podem se tornar desvantagens, dependendo do contexto de utilização das mesmas. Por exemplo, a característica de multi-hopping pode apresentar um peso no salto, o que não seria uma situação desejável ou ideal em determinadas ocasiões.

CAPÍTULO 4 – METODOLOGIA E IMPLEMENTAÇÃO

Toda pesquisa caracteriza-se por ser um procedimento racional cujo principal objetivo é encontrar respostas aos problemas que são propostos, além de confirmar ou refutar as hipóteses levantadas. (GIL, 2008).

Dessa forma, procurando observar de que maneira ocorre a implementação de um sistema cluster baseado em uma rede ad hoc e se este é viável quando comparado com um sistema centralizado, realizou-se uma pesquisa exploratória e prática.

A pesquisa exploratória tem como objetivo explorar um assunto ainda pouco conhecido, pouco estudado. Caracteriza-se por ser a primeira, ou uma das primeiras aproximações de um determinado tema e, por ser um tipo de pesquisa muito específica, quase sempre assume a forma de um estudo de caso.

Foram utilizados, portanto, dois procedimentos metodológicos para conduzir esta pesquisa: o levantamento bibliográfico e o estudo de caso. Estes, assim como qualquer outro procedimento metodológico, são utilizados a fim de juntar informações úteis para construir raciocínios e conclusões a respeito de um fenômeno.

O levantamento bibliográfico é um procedimento básico e obrigatório em qualquer pesquisa. Consiste na leitura e análise de livros, artigos, revistas, teses, entre outros e pretende-se, através deste, obter informações amplas e profundas a respeito de determinado assunto. É imprescindível para familiarizar o pesquisador em relação ao assunto escolhido e fundamentar seus estudos.

Já o estudo de caso consiste no estudo profundo e exaustivo de um caso em particular e permite conhecer o objeto de uma forma ampla e detalhada. Tem como finalidade explorar situações da vida real que não apresentam com clareza algum aspecto, descrever situações da realidade, explicar fenômenos específicos em situações também específicas que não poderiam ser obtidos de outra forma, dentre outros. É necessário ressaltar que os resultados não podem ser generalizados, uma vez que foram obtidos e observados em uma amostra não significativa.

De acordo com Yin (2001), o estudo de caso compreende três etapas: 1) a escolha do referencial teórico que se pretende utilizar – etapa esta que foi feita e explicada anteriormente

neste trabalho –; 2) a seleção do(s) caso(s) e forma de coleta de dados; 3) análise dos dados e interpretação dos resultados.

4.1 – Tecnologias utilizadas

Para a resolução do tema proposto, foram utilizados três computadores de arquitetura heterogênea, porém, sem muito desnível tecnológico.

Todos os computadores instalados o sistema operacional Windows 7 e todos os *softwares* necessários para a resolução de testes. Segue abaixo toda a configuração de *hardware* dos nós:

O primeiro nó do projeto é composto por um processador AMD Turion 64 X2 - 1.80 GHz com 2.00 GB de memória ram e placa de rede wireless Atheros AR5007EG, o segundo nó composto por um processador AMD Athlon 64 X2 Dual Core - 2.11 GHz com 2.00 GB de memória ram e um adaptador wireless BUFFALO WLI-UC-G30xN, e o terceiro e ultimo nó do projeto era composto por um processador Intel Core 2 Duo 1.73 GHz com 2 GB de memória ram e placa de rede wireless Atheros AR5007EG.

Para a análise desenvolvida neste projeto, foi utilizado uma implementação do padrão MPI chamada MPICH2.

4.1 – Configuração do sistema cluster baseado em rede Ad Hoc

Para a configuração do cluster em uma rede Ad Hoc foram necessárias duas configurações básicas, uma referente à rede Ad Hoc e outra referente ao ambiente de passagem de informação o MPICH2. Ambas as configurações são descritas a seguir:

4.1.1 – MPICH2

MPICH2 é uma implementação de alta performance e amplamente portátil do padrão MPI.

Pode-se concluir então que o MPICH é uma ferramenta de implementação do MPI. A ferramenta é *free*, portátil e está disponível para a maioria de sistemas UNIX e também para sistemas Windows.

Para o desenvolvimento do projeto, é necessária a configuração da ferramenta em questão. Toda a configuração é descrita a seguir:

Configuração do MPICH2

Para o desenvolvimento desse projeto foi utilizado a ferramenta MPICH2 na versão 1.4.1 para Windows 32 bits. A mesma versão deve ser instalada em todos os nós que serão utilizados no sistema cluster.

Antes de começar a instalação deve ser criada uma conta de usuário do windows com mesmo *login* e senha em todos os nós. Por exemplo, Computador Mestre (Login: Rafael , Senha: 123mudar), Computador1 (Login: Rafael , Senha: 123mudar) ... Computador N (Login: Rafael , Senha: 123 mudar). É muito importante que a conta de usuário criada seja um administrador ou pode surgir erros de credenciais. Após essa configuração pode-se então prosseguir ao passo de instalação.

A instalação deve ocorrer na conta de usuário criada, ou se estiver ocorrendo em outra conta, deve-se habilitar a opção de “instalar o MPICH para todos os usuários do computador”.

Depois de concluída a instalação, o executável “wmpiregister.exe” deve ser aberto. Ele está localizado na pasta de instalação do MPICH. São exibidos dois campos a serem preenchidos e quatro botões logo abaixo, como mostra a figura 13.

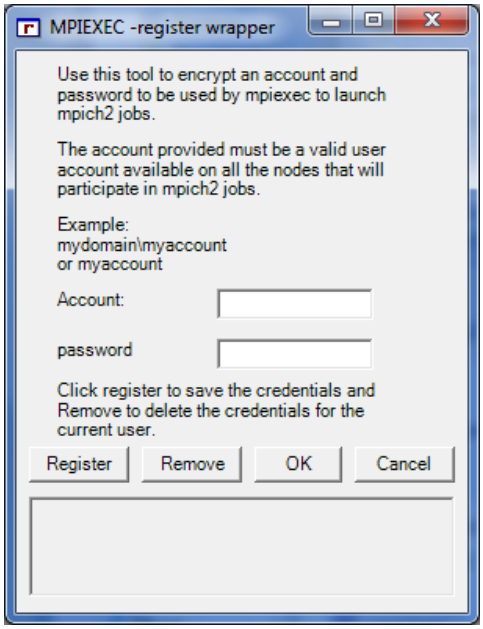

Figura 12. WMPIRegister.exe [Fonte: Própria]

No primeiro campo *"Account"* deve ser preenchido o nome de usuário do computador e no campo abaixo "*password"* deve ser preenchida a respectiva senha do nome de usuário informado. Vale ressaltar que esse nome e usuário devem ser iguais em todos os nós e todos eles devem ter uma conta de usuário no Windows com esses dados exatamente. Preenchido os campos, deve então apertar o botão "*Register"* e logo em seguida no botão "*OK"*. Feito isso todos os nós do sistema estarão com as credenciais necessárias para que ocorram conexões entre eles.

Após a criação das credenciais deve-se então montar a configuração entre os nós do sistema, elas podem ser feitas em outro executável, também localizado na pasta de instalação do MPICH, com o nome de "wmpiconfig.exe". A figura 13 mostra a interface desse executável:

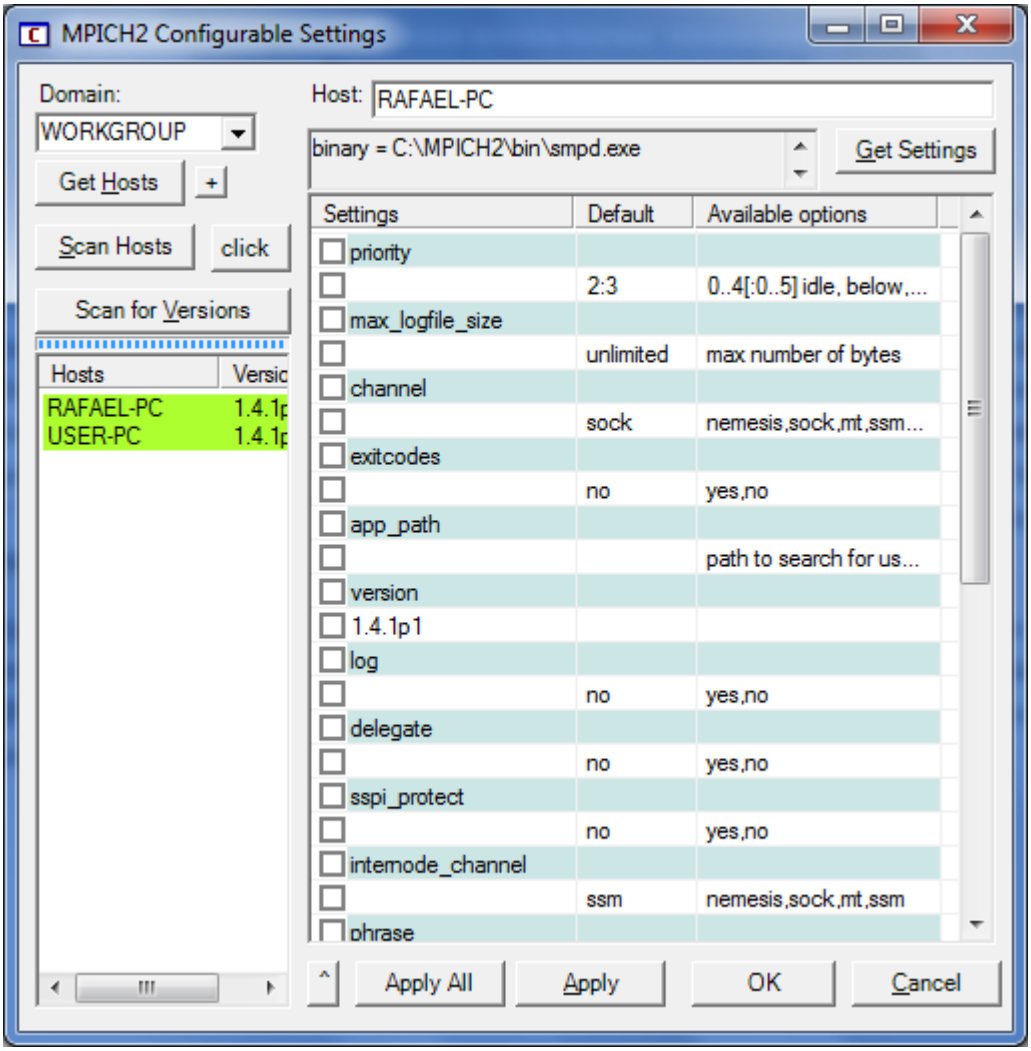

Figura 13. Interface do executável wmpiconfig.exe. [Fonte: Própria]

Primeiramente deve-se mudar o domínio para o grupo de trabalho onde estão localizados os nós. Para isso basta mudar o campo "*Domain"* para o grupo desejado, no exemplo da figura acima o campo está como *"WorkGroup"*. Feito isso, deve-se então clicar no botão "*Get Hosts"* e então em "*Scan Hosts"*, isso é necessário para verificar a disponibilidade dos hosts. Após clicar em "*Scan Hosts"* deve-se verificar a cor em que os nós aparecem no sistema, como padrão da ferramenta eles aparecem tarjados de verde se estão disponíveis ou de cinza se indisponíveis. No meio da interface encontram-se algumas configurações que podem ser feitas, para esse projeto todas as configurações foram as padrões do MPICH, ou seja, não foi alterado nada da configuração original que o MPICH disponibiliza.

Após a etapa de configuração estar pronta, cria-se uma pasta compartilhada no Windows onde toda a aplicação paralela que foi desenvolvida deve estar.

Configuração Rede Ad Hoc

A criação/configuração de uma rede Ad Hoc é relativamente simples se for feita pelo Windows 7. Deve-se acessar a “Centro de Redes e Compartilhamento”, em seguida clicar em “Configurar uma conexão ou Rede”. Posteriormente deve-se clicar em “Configurar uma rede ad hoc sem fio” e em seguida seguir os passos e preencher os campos necessários.

Quando pronto, a conexão ficará disponível para outros dispositivos que estão no raio de alcance dessa rede. Após a criação, a rede fica em um estado de aguardo, esperando que outro dispositivo se conecte à ela, quando isto ocorrer a rede estará pronta para o uso.

Feito todos os passos descritos anteriormente, a aplicação paralela estará pronta para ser executada.

4.1.2 – Algoritmo Distribuído

Antes de efetuar a execução do algoritmo distribuído na ferramenta MPICH é preciso explicar a funcionalidade deste.

O algoritmo pode ser encontrado no anexo A. Trata-se de um shellsort distribuído e toda a sua implementação foi feita na linguagem de programação C.

Vale ressaltar também que para esse projeto foi escolhido o algoritmo shellsort pois este não é o melhor, nem o pior método de ordenação existente.

Na figura 15 pode-se observar como o algoritmo distribuído funciona.

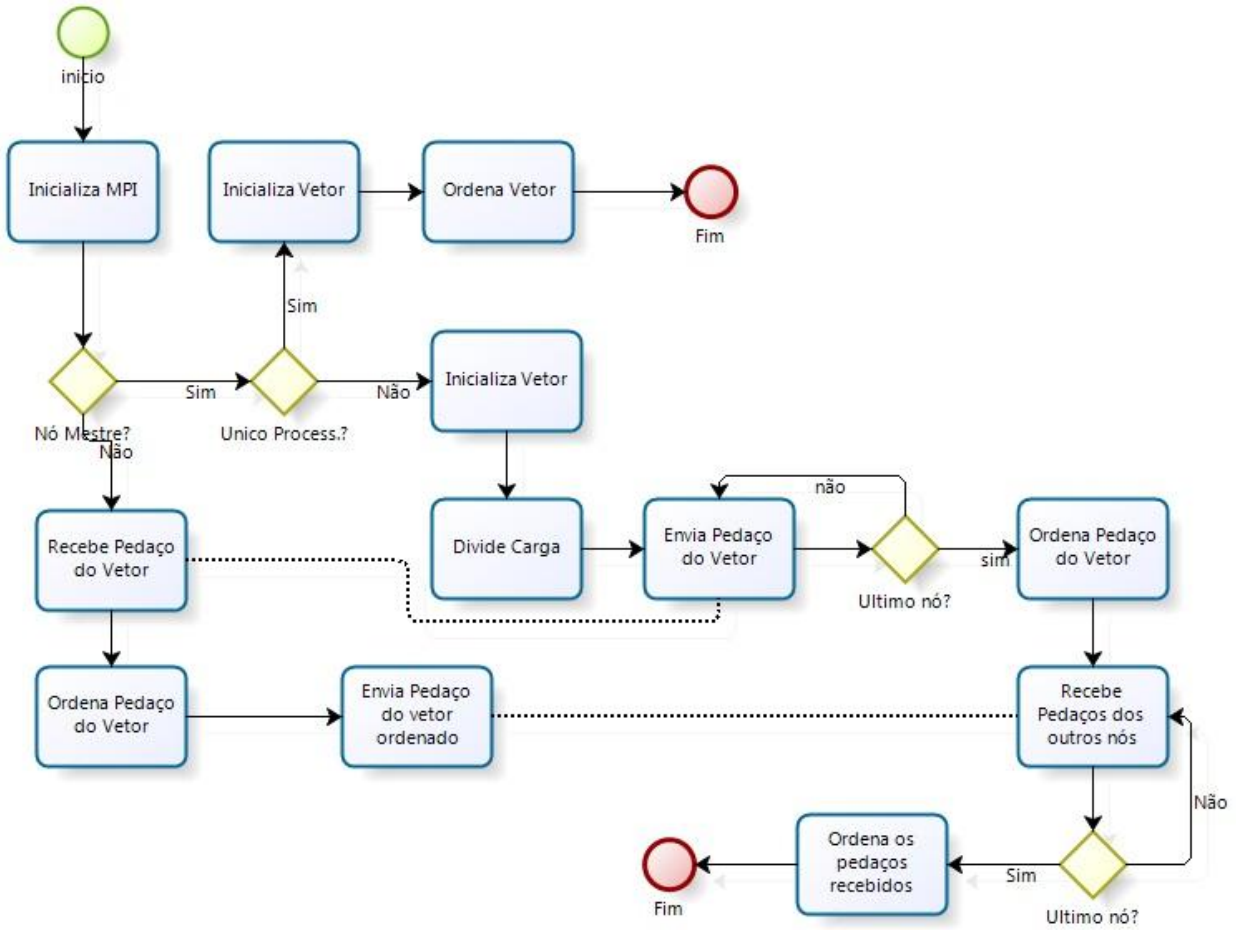

Figura 14. Diagrama do Algoritmo Distribuído Proposto (Fonte: Própria)

O algoritmo proposto funciona da seguinte maneira: Inicializa-se o programa, em seguida este inicializa o MPI com informações referentes ao nó que se encontra o algoritmo distribuído, após a inicialização do MPI pode-se percorrer entre dois caminhos dependendo de qual seja o nó.

Caminho 1 (o algoritmo está no nó mestre): Se o algoritmo estiver no nó mestre então há ainda outra comparação, se irá haver a paralelização do algoritmo ou se este será executado de forma centralizada.

Caminho 1.1 (algoritmo será executado de forma centralizada): Se o algoritmo for executado de forma centralizada, se inicializa o vetor, há a ordenação do vetor e chega-se ao fim do algortimo. Caso contrario segue-se o Caminho 1.2.

Caminho 1.2 (o algoritmo será executado de forma paralela): Se o algoritmo for executado de forma paralela então inicializa-se o vetor, divide-se a carga que será enviada aos outros nós e envia os pedaços para estes. O envio é feito de forma bloqueante, ou seja, o processo só é desbloqueado quando o nó receptor receber de fato a sua parte do vetor. Após enviar todas as partes aos seus respectivos destinos, é ordenada a parte restante do vetor,

exemplo, se há 3 nós no sistema, contando com o nó mestre, e nove posições de vetor, envia três posições para o nó 1, três para o nó 2, e três ficam para o nó mestre ordenar. Depois de feito a ordenação do pedaço de vetor restante, o nó mestre fica esperando a carga ordenada de todos os outros nós, a recepção também é bloqueante, o processo fica bloqueado até que toda a carga de todos os nós seja recebida, assim, quando retornadas, o nó mestre intercala os pedaços que foram recebidos e assim chegando ao fim do algorimo.

Caminho 2 (o algoritmo está no nó escravo): o nó escravo irá agir recebendo de forma bloqueante a sua parte do vetor, irá ordená-la a e em seguida enviar de forma bloqueante e com toda a carga ordenada para o nó mestre.

4.1.3 – Execução da Aplicação Paralela Usando MPICH2

Para execução do algoritmo não basta somente executar o programa que foi gerado. Depois de compilado o programa deve-se colocar o arquivo executável dentro de uma pasta compartilhada que foi criada durante o processo de configuração do MPICH2, feito isso abra então o executável “wmpiexec.exe” que está localizado na pasta de instalação do MPICH2. A figura 16 mostra a interface deste.

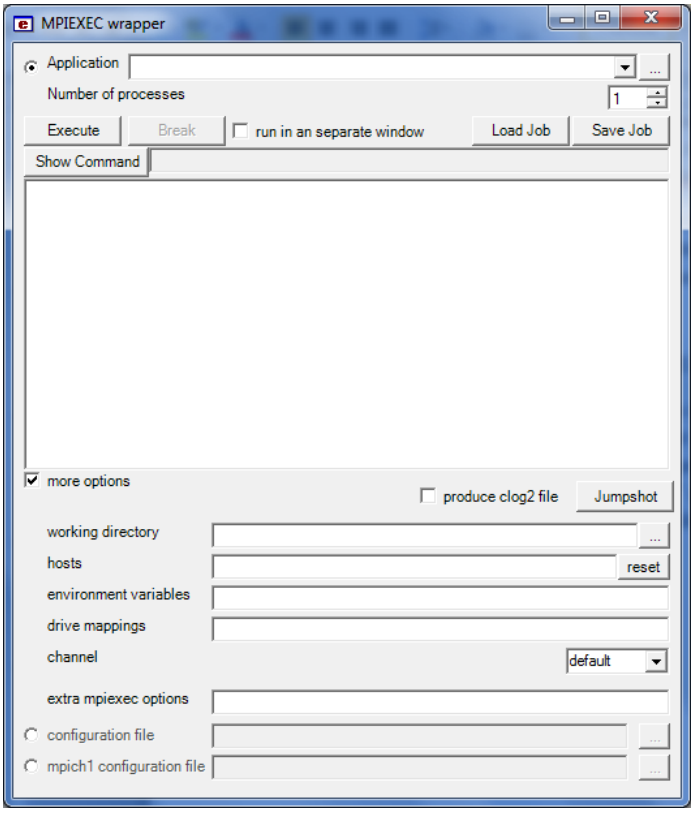

Figura 15. Interface “wmpiexec.exe”

No campo “Aplication” deve-se colocar o caminho de onde se encontra a aplicação distribuída. Logo abaixo deste campo há a opção de aumentar ou diminuir a quantidade de *processes*, este número deve ser igual a quantidade de nós que o sistema cluster possuir. Para a execução centralizada, basta deixar o campo “Number of Processes” atribuído com o valor 1. Após definido o numero de nós e o caminho para a aplicação paralela deve-se definir quem serão os nós do sistema.

No campo “hosts” é onde o nome ou ip do nó deve ser escrito, por exemplo, Rafael-PC User-PC 192.168.0.1, dessa maneira, sem virgula. Feito isso basta clicar em “Execute” e a aplicação distribuída irá ser executada.

CAPÍTULO 5 – RESULTADOS

Os resultados foram obtidos através de uma métrica de medida simples, com a seguinte fórmula:

$$Tempo \cong \frac{(double)(clock()Final - clock()Inicial)}{Clocks\_Per\_Second}$$

A hora do sistema foi capturada utilizando a função *clock()* encontrado na biblioteca *time.h* da linguagem de programação C.

O *clock()* inicial é capturado no nó mestre antes de ocorrer a inicialização do vetor, e o *clock()* final após a intercalação dos pedaços de vetores recebidos.

O atributo *Clocks_Per_Second* é um retorno de quantos pulsos de clock o processador tem em um segundo, esse atributo é encontrado na biblioteca nativa da linguagem de programação C.

O tempo foi determinado de maneira aproximado, isso se dá por que a função *clock(),* se utilizada no Windows, retorna um número inteiro da quantidade de pulsos de clock que o processador fez desde a inicialização da aplicação, dessa maneira o tempo não pode ser considerado exato, pois se o programa distribuído for bloqueado pelo sistema operacional por outro que tenha, por exemplo, maior prioridade do uso do processador, o *clock()* final terá contado também os pulsos que o processador deu durante o tempo que a aplicação distribuída ficou bloqueada. Além disso, a fim de obter uma métrica mais apurada realizou-se três execuções, e do tempo dado por elas fez-se uma média, tal como a fórmula[8]:

$$MediaTempo = \frac{Tempo1 + Tempo2 + \cdots + Tempon}{n}$$

Foram feitos quatro testes com quantidade de posições de vetores determinadas intencionalmente para verificação do funcionamento em relação ao tempo, escolhendo-se assim posições com intervalos significativos entre si (9.000; 150.000; 900.000 e 9.000.000).

Situação 1 – 9.000 posições no vetor

[8] Onde n é igual ao número de execuções.

Quando executado com 9.000 posições no vetor, houve melhor desempenho (Tempo x Quantidade de Nós) quando realizado com apenas um nó (0,002s), enquanto em dois nós o tempo foi de 0,011s e com três nós 0,028s. Isto pode ser observado na tabela 1 e gráfico 1.

Quantidade de Nós	Tempo
1 Nó	0,002s
2 Nós	0,011s
3 Nós	0,028s

Tabela 1 – Situação 1.

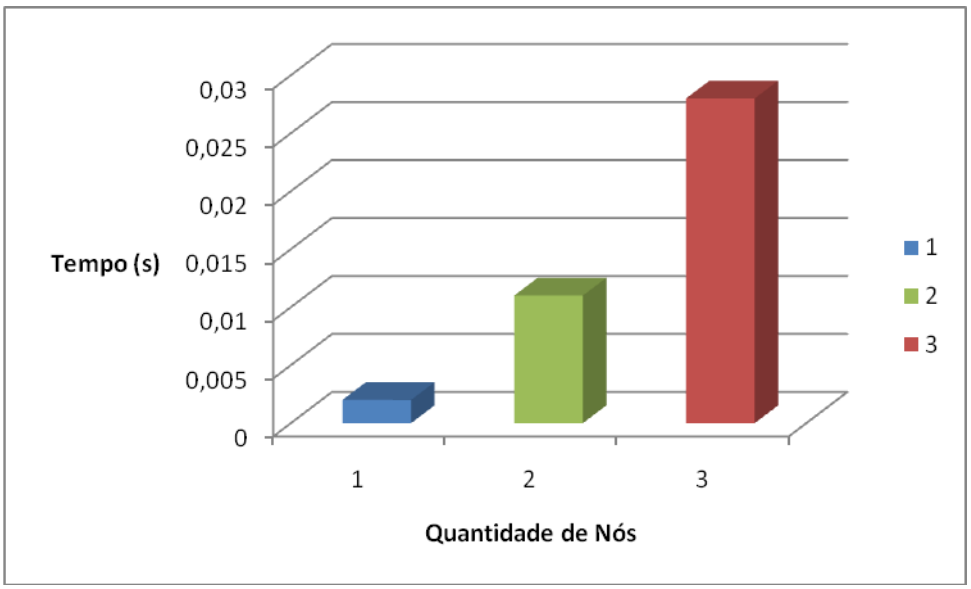

Gráfico 1 – Situação 1

Situação 2 – 150.000 posições no vetor

Com 150.000 posições no vetor, ainda é possível verificar um melhor desempenho (Tempo x Quantidade de Nós) quando executado com apenas um nó (0,118s). Executando com dois nós temos o tempo de 0,25s e com três nós o tempo é de 0,58s. A tabela 2 e gráfico 2 demonstram essas informações.

Quantidade de Nós	Tempo
1 Nó	0,118s
2 Nós	0,25s
3 Nós	0,58s

Tabela 2 – Situação 2

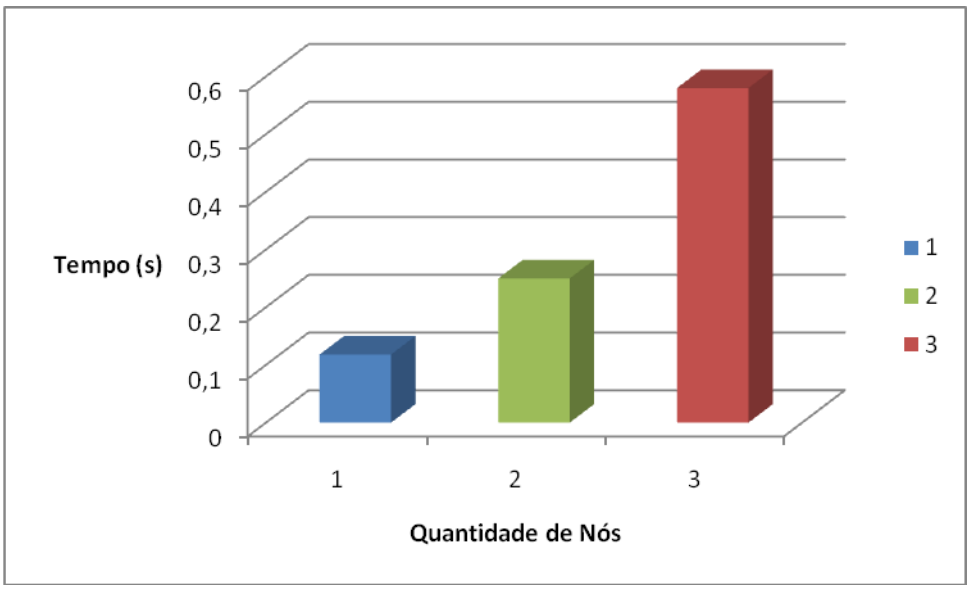

Gráfico 2 – Situação 2

Situação 3 – 900.000 Posições no vetor

Com 900.000 posições o desempenho (Tempo x Quantidade de Nós), quando executado em apenas um nó (0,908s), ainda é superior se comparado com dois nós (4,69s) e três nós (3,53). Nota-se que o desempenho em três nós foi superior que em dois nós, pode-se supor que isto ocorreu devido ao fato que com uma carga muito grande, não compensaria dividir apenas para um segundo nó, e sim para mais deles. A tabela 3 e gráfico 3 mostram os detalhes.

Quantidade de Nós	Tempo
1 Nó	0,908s
2 Nós	4,69s
3 Nós	3,53s

Tabela 3 – Situação 3

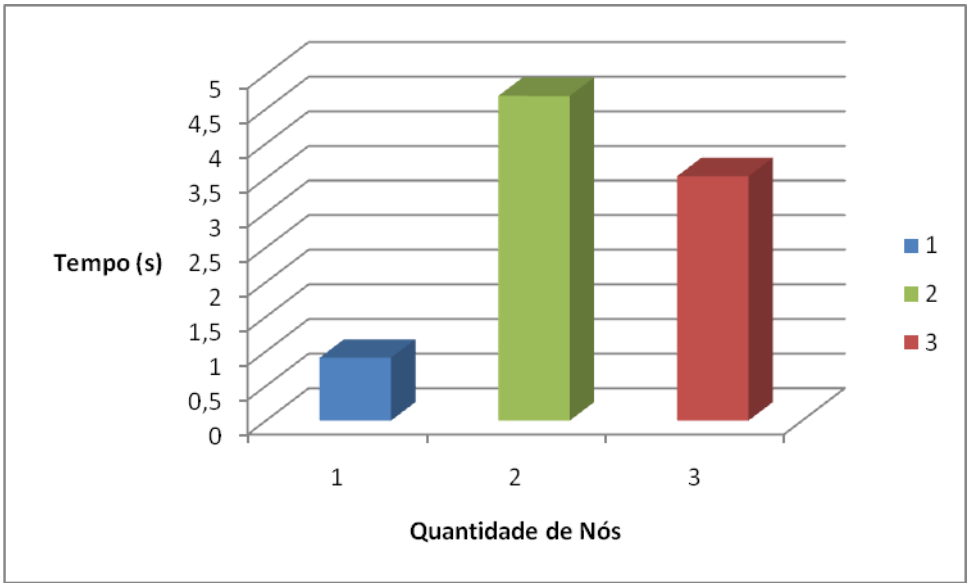

Gráfico 3 – Situação 3

Situação 4 – 9.000.000

Nessa situação, ocorreu um resultado semelhante a situação 3. Quando executado em apenas um nó (13,59s) o desempenho (Tempo x Quantidade de Nós) foi maior que se comparado com dois nós (48,19s) e com três nós (35,976). Ainda assim, quando executado com 3 nós o desempenho foi superior se comparado a dois nós.

Quantidade de Nós	Tempo
1 Nó	13,59s
2 Nós	48,19s
3 Nós	35,976s

Tabela 4 – Situação 4

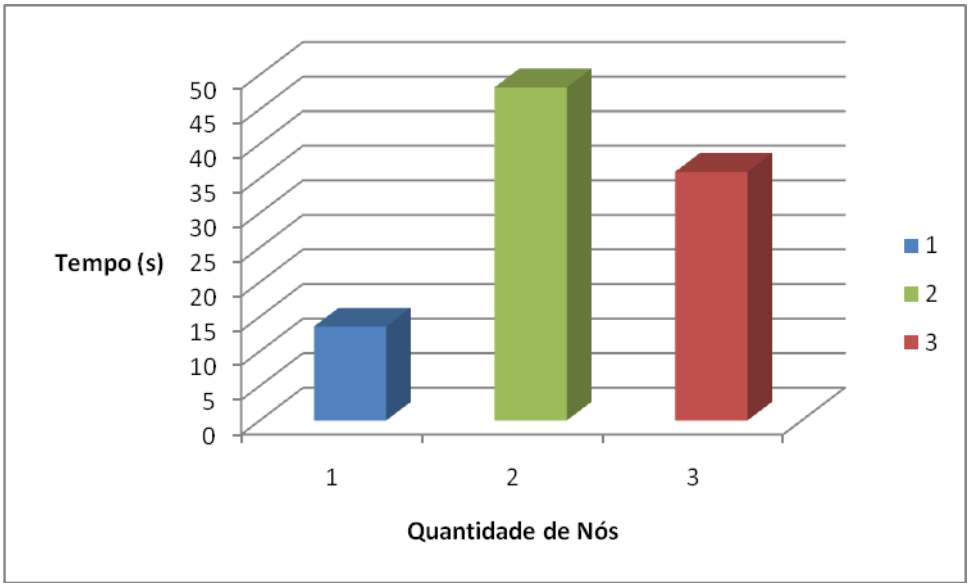

Gráfico 4 – Situação 4

Situação Geral

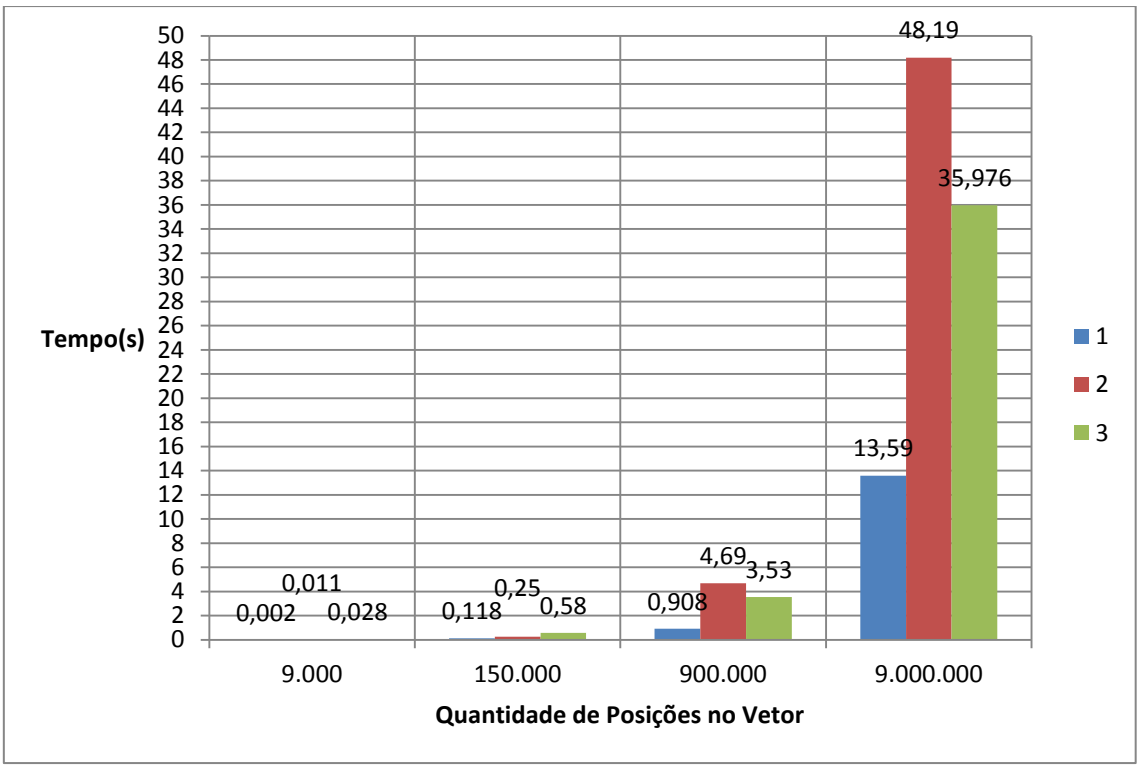

Gráfico 5 – Situação Geral

Pode-se observar que quando a aplicação distribuída é executada apenas no nó centralizado o desempenho foi superior que se executado de forma distribuída.

CAPÍTULO 6 - CONCLUSÃO

O trabalho teve como objetivo a análise e implementação de um sistema distribuído cluster baseado em uma rede wireless ad hoc. Os sistemas distribuídos vêm despertando cada vez mais interesse em pesquisas devido à boa relação de custo x benefício que apresentam. Por sua vez as redes wireless despertam interesse devido a sua praticidade de uso, e em especial, as ad hoc pela não necessidade de infraestrutura.

Dessa forma, foram feitas análises de desempenho com o objetivo de unir esses dois conceitos e verificar como eles funcionam em conjunto.

Após a análise de resultados, pode-se concluir que neste caso, onde se utilizou um padrão 802.11g, não é viável a implementação de um sistema distribuído com base em redes wireless. Isso se da devido às taxas de transmissão que as redes wireless implementadas nesse padrão apresentam.

Além disso, é importante ressaltar que são necessárias mais pesquisas nessa área a fim de constatar ou estudar mais detalhadamente como sistemas distribuídos baseados em rede wireless podem se tornar mais viáveis e eficientes, uma vez que essas tecnologias são cada vez mais utilizadas e imprescindíveis no mundo da tecnologia de informação.

REFERÊNCIAS BIBLIOGRÁFICAS

ALECRIM, Emerson. **Cluster:** Principais Conceitos. 2004. Disponível em: http://www.infowester.com/cluster.php. Acesso em: 26 de Agosto de 2011.

BARROS, Andersown Becher Paes. **Computação em Cluster.** Instituto Cuiabano de Educação – Faculdades (ICE), Cuiabá, (s.d.).

CORRÊA, Underléia; PINTO, A.R.; CODAS, Andres; FERREIRA, D.J.; MONTEZ, Carlos. **Redes Locais Sem Fio:** Conceitos e Aplicações. Departamento de Automação e Sistemas – Programa de Pós-Graduação em Engenharia Elétrica Universidade Federal de Santa Catarina (UFSC), (s.d.).

COSTA, Jefferson. Wireless: **Redes Sem Fio.** Disponível em: http://www.jeffersoncosta.com.br/wireless.pdf. Acesso em: 13 de Julho de 2011.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim.**Sistemas Distribuídos:** Conceitos e Projeto.4.ed. Bookman, s.l., 2007.

FARIAS, Paulo César Bento. **Redes Wireless** – Parte I. 2005. Disponível em: http://www.juliobattisti.com.br/tutoriais/paulocfarias/redeswireless001.asp. Acesso em: 12 de Julho de 2011.

GAST, Matthew, S. 8002.11 **Wireless Network:** The Definitive Guide.1.ed. O'Reilly Media, 2002.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa.** 5. ed. Atlas: São Paulo, 2008.

JÚNIOR, Faustino Pereira Esli; FREITAS, Reinaldo Borges. **Construindo Supercomputadores com Linux:** Cluster Beowulf. Monografia (Centro Federal de Educação Tecnológica de Goiás). Goiânia, 2005.

JOHANSSON, Tomas; MOTYČKOVÁ-CARR, Lenka.**On Clustering in Ad Hoc Network**.Division of Computer Science and NetworkingLuleåUniversityofTechnology, Agosto, 2003.

MOHAPATRA, Prasant; KRISHNAMURTHY Srikanth.**Ad Hoc Networks:** Technologies and Protocols. UniversityofCalifornia, Riverside, 2005.

MORIMOTO, Carlos E. **Servidores em Cluster e Balanceamento de Carga.** 2007. Disponível em: http://www.hardware.com.br/artigos/cluster-carga/. Acesso em: 27 de Setembro de 2011.

OMOTO, Marcelo Augusto. **Configuração de um cluster para o algoritmo shellsort distribuído.** Monografia (Centro Eurípides de Marilia – UNIVEM) – Marília, SP, 2009.

PEREIRA, Ivana Carial de Miranda; PEDROSA, Aloyso de Castro P. **Aplicações Militares Empregando Redes Móveis Ad Hoc.** Universidade Federal do Rio de Janeiro. (s.d.).

PITANGA, Marcos. **Computação em Cluster.** 2003. Disponível em: http://www.clubedohardware.com.br/artigos/153. Acesso em: 10 de Setembro de 2011.

PITANGA, Marcos**. Construindo Supercomputadores com Linux.**3.ed. Brassport: Rio de Janeiro, 2008.

PONTE, Thiago Costa**. Algoritmos para clusterização distribuída em redes móveis ad hoc.** PUC-Rio, 2006.

RODRIGUES, André Soares; GOUVÊA, Thiago Maia; PEREIRA, Francislane. **Cluster e Grid Computing.** IV SEAC – Semana de Atividades Científicas. Resende, RJ, (s.d.).

SANTINI, Paolo.**The Tolpology Control in Wireless Ad Hoc and Sensor Network**, Instituto de Informatica e Telemática del CNR, Italy, John Wiley & Sons Ltd, 2005.

STALLINGS, William. **Arquitetura e Organização de Computadores.**5.ed. Prentice Hall: São Paulo, 2003.

TANENBAUM, Andrew S.; VAN STEEN, Maarten. **Distributed Systems:** Principles and Paradigms.2.ed. Vrije Universiteit. Amsterdam, Holanda, 2007.

TANENBAUM, Andrew S. **Rede de Computadores.** Pearson: 2003.

UNIVERSIDADE DE CAMPINAS. **Sistemas Distribuídos.** Campinas. Disponível em :http://www.ic.unicamp.br/~ranido/mc704/intro.pdf. Acesso em: 30 de Março de 2011.

_____. **MPICH2.** Disponível em: http://www.mcs.anl.gov/research/projects/mpich2/. Acesso em: 30 de outubro de 2011.

YIN, Robert K. **Estudo de caso: planejamento e métodos.** Bookman: Porto Alegre, 2001.

ANEXO A – Aplicação Distribuída – Shellsort.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include "mpi.h"

// Tamanho dos vetores
#define TAM 9000000

/*****************************************************************************/
/* Functions 'n Procedures                                                   */
/*****************************************************************************/

int *initVector(int vSize) {
  int i, *V;

  if ( vSize <= 0 )
    return NULL;

  V = (int *)malloc(vSize * sizeof(int));

  for ( i = 0; i < vSize; i++ )
    V[i] = rand() % vSize;

  return V;
}

void showVector(int *V, int vSize) {
  int i;

  fprintf(stdout, "V = \n");
  for ( i = 0; i < vSize; i++ )
    {
      fprintf(stdout, "%05d ", V[i]);

      if ( (i+1) % 10 == 0 )
       fprintf(stdout, "\n");
    }
  fprintf(stdout, "\n");
}

// ShellSort
void shellSort(int *vetor, int tamanho) {
  int i, j, valor;
  int gap = 1;
  do {
```

```
    gap = 3 * gap + 1;

  } while (gap < tamanho);
  do {
    gap /= 3;
    for (i = gap; i < tamanho; i++) {
      valor = vetor[i];
      j = i - gap;
      while (j >= 0 && valor < vetor[j]) {
      vetor [j + gap] = vetor[j];
       j -= gap;
      }
      vetor [j + gap] = valor;
    }
  } while (gap > 1);
}

int *merge(int **vec, int N, int *vSize) {

  int i, j, k, fSize;
  int *fVector;
  int *vecPointer;

  // Calculando tamanho final do vetor
  for ( i = 0, fSize = 0; i < N; i++ )
    fSize += vSize[i];

  // Vetor final
  fVector = (int *)malloc(fSize * sizeof(int));

  // Posição atual dos vetores
  vecPointer = (int *)malloc(N * sizeof(int));

  for ( i = 0; i < N; i++ )
    vecPointer[i] = 0;

  // Merge
  for ( i = 0; i < fSize; i++ )
    {
      // Escolha o menor elemento dentre os vetores
      k = -1;
      for ( j = 0; j < N; j++ )
{
  if ( (vecPointer[j] < vSize[j]) && (k == -1) )
    k = j;
  else
    if ( vecPointer[j] < vSize[j] )
      if ( vec[k][vecPointer[k]] > vec[j][vecPointer[j]] )
k = j;
}
```

```
      // Copiando o menor valor e avancando o apontador de onde
      // o elemento foi copiado
      fVector[i] = vec[k][vecPointer[k]++];
    }

  // Liberando memoria
  free(vecPointer);

  // Retornando o vetor
  return fVector;
}

/*****************************************************************************/
/* MAIN                                                                      */
/*****************************************************************************/
int main(int argc, char **argv) {

  //  int i, envia, recebe, procs, meurank, aux;
  int tag = 10;
  //  int vetor[TAM];
  //  int vetor2[TAM], vetor3[TAM];
  clock_t start, finish;
  float duracao;

  /***************************************/

  int i, tSize;
  int myRank, nProc;

  int *vector, *vSize, *fVector;
  int **vecPointers;

  // Inicializando gerador de numeros pseudo-aleatorios
  srand(time(NULL));

  // Inicializacao MPI
  MPI_Status status;
  MPI_Init(&argc, &argv);

  MPI_Comm_size(MPI_COMM_WORLD, &nProc);
  MPI_Comm_rank(MPI_COMM_WORLD, &myRank);



  if ( myRank == 0 )
    {
      if ( nProc == 1 )
{
      start = clock();
```

```
  // Sistema distribuido pra caralho
  vector = initVector(TAM);
  shellSort(vector, TAM);
  finish = clock();

  duracao = (float)(finish - start) / CLOCKS_PER_SEC;
  printf("Duraçao: %3.11f seconds\n", duracao);
            //showVector(vector, TAM);
}
     else
{
  // Marcando tempo de inicio
  start = clock();
  printf("Start: %d\n",start);

  // Inicializa vetores
  vector = initVector(TAM);

  // Inicializa tamanhos e apontadores
  vSize = (int *)malloc(nProc * sizeof(int));
  vecPointers = (int **)malloc(nProc * sizeof(int *));

  for ( i = 0, tSize = 0; i < nProc - 1; i++ )
    {
      vSize[i] = TAM / nProc;
      vecPointers[i] = &(vector[tSize]);
      tSize += vSize[i];
    }

  // Ultimo vetor pode ter um tamanho diferente se o tamanho do vetor inicial
  // nao for divisivel pelo numero de processos
  vSize[i] = TAM - tSize;
  vecPointers[i] = &(vector[tSize]);

  for ( i = 0; i < nProc; i++ )
    {
      fprintf(stdout, "vSize[%d]=%d\n", i, vSize[i]);
    }

  // Enviando pedacos do vetor
  printf("nPoc: %d \n", nProc);
  for ( i = 0; i < nProc - 1; i++ ) {
              MPI_Send(vecPointers[i], vSize[i], MPI_INT, i+1, tag, MPI_COMM_WORLD);



  }

  // Ordenando "meu" pedaco do vetor
            shellSort(&(vector[tSize]), vSize[i]);
```

```
  // Recebendo pedacos do vetor ordenados
  for ( i = 0; i < nProc - 1; i++ )
            MPI_Recv(vecPointers[i], vSize[i], MPI_INT, i+1, tag, MPI_COMM_WORLD,
                  &status);

  // Merge
  fVector = merge(vecPointers, nProc, vSize);

  finish = clock();
  printf("finish: %d\n", finish);

  duracao = (float)(finish - start) / CLOCKS_PER_SEC;
   printf("Duraçao: %3.11f seconds\n", duracao);

          //showVector(fVector, TAM);
    }
    }
  else
    {
      // Calculando tamanho do vetor
      vSize = (int *)malloc(1 * sizeof(int));
      vSize[0] = TAM / nProc;

      // Alocando espaço para o vetor
      vector = (int *)malloc(vSize[0] * sizeof(int));

      // Recebe o vetor
      MPI_Recv(vector, vSize[0], MPI_INT, 0, tag, MPI_COMM_WORLD, &status);

      // Ordena o vetor
      shellSort(vector, vSize[0]);

      // Envia o vetor
      MPI_Send(vector, vSize[0], MPI_INT, 0, tag, MPI_COMM_WORLD);
    }

  MPI_Finalize();

  return 0;
}
```