UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DANIEL LOPES PEREIRA

COMPARAÇÃO DO IMPACTO DO CANAL DE COMUNICAÇÃO NO DESEMPENHO DE *GRIDS*

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2014

DANIEL LOPES PEREIRA

COMPARAÇÃO DO IMPACTO DO CANAL DE COMUNICAÇÃO NO DESEMPENHO DE *GRIDS*

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Richard Duarte Ribeiro

PONTA GROSSA

2014

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharel em Ciência da Computação

TERMO DE APROVAÇÃO

COMPARAÇÃO DO IMPACTO DO CANAL DE COMUNICAÇÃO NO DESEMPENHO DE *GRIDS*

por

DANIEL LOPES PEREIRA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 12 de novembro de 2014 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Richard Duarte Ribeiro
Prof. Orientador

_____
Tânia Lúcia Monteiro
Membro titular

_____
Marcus Vinicius Drissen Silva
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

RESUMO

PEREIRA, Daniel Lopes. **Comparação do Impacto do Canal de Comunicação no Desempenho de *Grids***. 2014. 69 páginas. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2014.

A alta qualidade dos recursos de *hardware* que equipam os computadores atuais não é suficiente para resolver certos tipos de problemas de computação em tempo hábil. Dentre as tecnologias existentes para a computação de tais problemas, está o *grid computing*. Tal tecnologia une diferentes dispositivos como: computadores, *smartphones* e *tablets* e, se utiliza de seus recursos para formar um ambiente de alto poder computacional. Este trabalho realiza um estudo sob a influência do canal de comunicação em *grids*, usando um simulador para estudar a influência das variações de velocidade e largura de banda no desempenho final de um *grid*.

Palavras-chave: *Grid computing*. Desempenho. Canal de comunicação.

ABSTRACT

PEREIRA, Daniel Lopes. **Comparison of the Impact of Communication Channel on Performance in Grids**. 2014. 69 pages. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Federal Technology University - Parana. Ponta Grossa, 2014.

The high quality of the hardware present in the individual computers is not enough to solve certain types of calculation problems within the required time. Among the existing technology capable of solving those problems is the Grid Computing. This technology put together different devices, such as computers, smartphones and tablets, using their available resources to form an environment with high computational power. This work studies the influence of the communication channel in grids, using a simulator to study the influence of variations like speed and bandwidth on the final performance of a grid.

Keywords: Grid computing. Performance. Communication channel.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

SETI@hom	*Search for Extraterrestrial Intelligence*
WCG	*World Community Grid*
GGF	*Global Grid Forum*
OGSI	*Open Grid Services Infrastructure*
OGSA	*Open Grid Services Architecture*
CERN	*Conseil Européen pour la Recherche Nucléaire*
SPRACE	São Paulo *Research and Analyzes Center*
JVM	*Java Virtual Machine*
XML	*Extensible Mark-up Language*
FLOPS	Operações de Ponto Flutuante por Segundo
IPS	Instruções por Segundo
MIPS	Milhões de instruções por Segundo
kBps	kilobits por segundo
MBps	megabits por segundo
us	microssegundos
ms	milissegundos
Mf	mega FLOPs
Gf	giga FLOPs

SUMÁRIO

1 INTRODUÇÃO

A computação atingiu uma alta qualidade de recursos, levando ao patamar atual, cujos processadores que equipam os computadores possuem capacidade de processamento antes impensadas. Similarmente, as redes de computadores evoluíram, tornando-se rápidas e disseminadas. O *software* também evoluiu tornando-se mais modular, configurável e dinâmico.

A Figura 1 ilustra a evolução do processamento atrelado ao número de núcleos do processador.

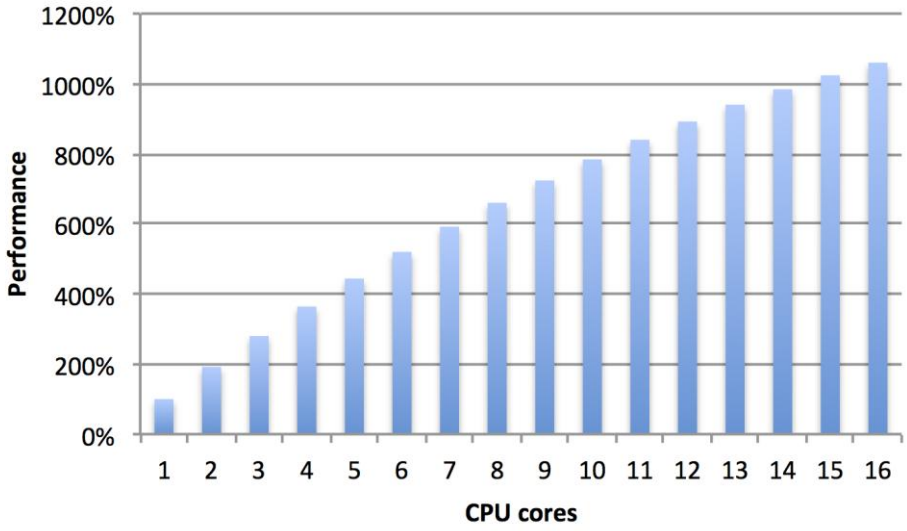

Figura 1 – Evolução dos processadores
Fonte: Errata *Security* (2014)

Mesmo com tal avanço, o *hardware* disponível ainda não é suficiente para resolver certos problemas de computação em tempo hábil, como por exemplo em alguns problemas avançados de física ou ciência, que demandam meses e até anos para serem processados.

Tal cenário cria a necessidade de se unir diversos computadores para trabalharem colaborativamente. Uma forma de implementar esta solução é através do uso de *grids*, que permitem a conexão de computadores diferentes através do uso de redes, e consequentemente de seus recursos computacionais (BUYYA; VENUGOPAL, 2005).

Grid é um modelo computacional capaz de dividir tarefas (através do uso de redes) entre as máquinas que lhe compõe, alcançando uma alta taxa de processamento, distribuindo as tarefas para os clientes (WALDROP, 2002). A grande vantagem do seu uso é, que o mesmo pode transcender o uso de redes de um único domínio (ex. domínio administrativo particular).

Um dos grandes benefícios do uso dessa tecnologia é permitir o compartilhamento de recursos diferentes, mesmo que estejam espalhados por diversos domínios, estendendo assim as fronteiras físicas e geográficas (WALDROP, 2002).

Grids são utilizados por diversos projetos científicos nos mais variados campos do conhecimento, tais como estudos médicos, ambientais, físicos, etc. Abaixo seguem alguns exemplos.

- SETI@home (*Search for Extraterrestrial Intelligence*): Projeto que procura por vida inteligente fora da terra, através da análise de sinais de rádio de banda estreita captados por telescópio. É um dos mais antigos e conhecidos projetos que utiliza os recursos de *grid* (ANDERSON *et al*, 2002).
- World Community Grid: Projeto patrocinado pela IBM, tem o objetivo de unir as pessoas que desejam dispor seus recursos de processamento para cálculos científicos. É um conjunto de projetos com fins humanitários. Atualmente possui mais de 2.7 milhões de computadores, *smartphones* e *tablets* contribuindo voluntariamente em 80 países (WORLD COMMUNITY GRID, 2014).
- Samsung Power Sleep: Projeto usado por milhares de pessoas em seus dispositivos móveis. Criado pela Samsung em parceria com a Universidade de Viena, utiliza o poder de processamento dos dispositivos móveis quando estes não estão sendo utilizados, para calcular sequências de proteínas (SAMSUNG POWER SLEEP, 2014).

Pelo fato de *grids* fazerem uso intensivo das redes para usar os recursos remotos (WALDROP, 2002), este trabalho visa estudar se o desempenho de um *grid* pode ser afetado pela estrutura de rede que o compõe.

1.1 OBJETIVOS

O objetivo desse trabalho é simular um ambiente *grid,* com variações na largura de banda e latência do canal de comunicação, a fim de representar as redes utilizadas tanto por computadores quanto por dispositivos móveis, e estudar o impacto que o canal de comunicação pode gerar no desempenho de processamento de um *grid*.

1.1.1 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Estudar o simulador SimGrid (CASANOVA, 2001) para a criação do ambiente *grid;*
- Simular diferentes larguras de banda e latências do canal de comunicação com possíveis perdas de conexão em um *grid*;
- Analisar os resultados das simulações no desempenho do *grid*.

1.2 JUSTIFICATIVA

Sempre existiram e existirão situações em que o poder computacional de uma máquina isolada não é suficiente para atender as necessidades de seus usuários, mesmo com o uso de supercomputadores. Contudo, montar *grids* para aumentar o poder computacional disponível pode ser uma tarefa custosa e complicada, imaginando-se que isso implique em reunir vários computadores diferentes. Todavia, a grande evolução dos dispositivos móveis veio mudar esse entendimento.

Hoje em dia, *smartphones* e *tablets* se tornaram versões menores de computadores pessoais, possuindo grande poder de processamento e memória disponível. Além disso, os mesmos estão, em geral, permanentemente conectados à internet, possibilitando seu uso na composição de *grids.*

A criação de um *grid* não é baseada unicamente no poder de processamento e memória disponível de suas partes. A velocidade de troca de dados que ocorre entre as partes também é fundamental.

Sabendo-se que os dispositivos que compõe um *grid* podem estar localizados em lugares distintos, se torna árduo inferir a respeito da qualidade da infraestrutura de rede utilizada por cada dispositivo. Tendo esse cenário como base, é necessário que se analise o impacto da disponibilidade e qualidade da infraestrutura de rede usada pelo *grid.*

1.3 ESTRUTURA DO TRABALHO

Este trabalho encontra-se subdividido em cinco capítulos que abordam os seguintes temas: o primeiro capítulo apresenta a introdução e os objetivos do trabalho. O segundo capítulo descreve a revisão bibliográfica acerca de *grids* e simuladores de *grids*. O terceiro capítulo descreve as ferramentas utilizadas e as estratégias empregadas para as simulações. O quarto capítulo aborda os resultados, os métodos de avaliação e interpretação. O quinto capítulo apresenta a conclusão do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo explica algumas definições importantes, além de fundamentar e dar consistência ao estudo desenvolvido. Seu objetivo é apresentar o problema a ser pesquisado sob o aspecto teórico e embasá-lo em outros estudos e pesquisas já realizadas. Neste trabalho serão utilizados conteúdos envolvendo sistemas distribuídos, *grid computing*, canal de comunicação e simulação. Assim, serão descritos características e funcionalidades.

2.1 GRID COMPUTING

A evolução dos computadores e das redes, nos permitem montar grandes sistemas computacionais compostos por grande quantidade de máquinas interconectadas por uma rede. Tais sistemas são chamados de sistemas distribuídos, e fornecem serviços de alto desempenho ao usuário (WALDROP, 2002).

O modelo de *grid computing* foi proposto nos anos 90 com o objetivo de auxiliar atividades de pesquisa e desenvolvimento científico, reduzindo os custos computacionais através do compartilhamento de recursos, os quais podem estar geograficamente distribuídos, mas integrados através de uma rede (WALDROP, 2002).

A palavra *grid* se relaciona com o termo "*Eletrical Power Grid"*, o qual remete a rede elétrica. Tal analogia deve-se ao fato de não se preocupar com fonte geradora de energia, assim como em um ambiente *grid computing*, se faz necessário preocupar-se apenas com a disponibilidade dos recursos (WALDROP, 2002).

2.1.1 Histórico

Historicamente, o uso de supercomputadores foi mais utilizado para a solução de problemas que utilizassem computação intensiva. Porém estas máquinas eram excessivamente caras o que não dava a possibilidade de universidades menores e/ou empresas de pequeno porte adquirirem, ficando restritas a grandes

corporações, universidades privilegiadas e centros de pesquisas. Mesmo com a evolução dos computadores, e com isso a queda dos preços das tecnologias, esta situação é ainda vigente. Assim, universidades e/ou usuários que não adquiram estes aparelhos pelo preço ou pela esporadicidade do uso de computação de alto desempenho, são excluídos da possibilidade do uso desta (GOLDCHLEGER, 2004).

Apesar do preço e da capacidade de supercomputadores eles tem o mesmo fim que todos os computadores: o defasamento e a obsolência. Segundo a Lei de Moore que se fez verdadeira desde 1965, a cada 18 meses dobra-se a capacidade dos processadores (MOORE, 1965), o que juntamente com as mudanças de arquitetura, ajuda no avanço da tecnologia, mas também alavanca o defasamento dos computadores atuais. O resultado disso é que as universidades e/ou empresas de pequeno porte raramente têm a possibilidade de aquisição de supercomputadores. Uma alternativa a esses problemas seria a união de computadores de pequeno porte espalhados em rede, formando o que vem a ser conhecido como um sistema distribuído (GOLDCHLEGER, 2004).

Sistemas distribuídos são sistemas cujas características principais são as de acesso remoto a recursos e informações. Componentes de *hardware* e *software* são encontrados em computadores remotos, conectados por rede. Estes computadores provêm comunicação e coordenação de suas ações através da troca de mensagens (COULOURIS *et al,* 2012). *Grid computing* e *Cluster* são exemplos de sistemas distribuídos.

2.1.2 Conceitos

Cluster é um sistema para processamento paralelo e distribuído que consiste de uma coleção de computadores interconectados e trabalhando juntos como um único recurso computacional integrado. Utilizando-se de tais computadores é possível realizar processamentos similares aos obtidos em supercomputadores (GOLDCHLEGER, 2004).

Em um *cluster*, todos os computadores pertencentes a ele devem possuir o mesmo sistema operacional instalado, e a heterogeneidade dos equipamentos é diretamente proporcional ao nível de complexidade para a implantação deste. O que traz uma restrição de uso e implantação (GOLDCHLEGER, 2004).

Grid computing, é um modelo que se utiliza da rede para tornar possível a criação de uma “máquina virtual” com alta taxa de processamento, permitindo o desenvolvimento de aplicações antes restritas a supercomputadores. O mesmo pode também ser definido como uma infraestrutura que utiliza recursos ociosos de computadores independentes, sem a preocupação de localização física (FOSTER *et al*, 2002).

Máquinas virtuais são máquinas implementadas e construídas através de *software*, sendo capazes de processarem e executarem programas como computadores físicos. Isso é possível através do mapeamento virtual de componentes como processador, memória e dispositivos de entrada e saída (SMITH, 2005).

O objetivo do *grid* é combinar o poder de processamento de vários computadores ligados em rede para conseguir executar tarefas que não poderiam ser realizadas, ou pelo menos não com um desempenho satisfatório em um único computador e, ao mesmo tempo, executá-las a um custo mais baixo do que de um supercomputador de potência semelhante (IOSUP; EPEMA, 2011).

A Figura 2 ilustra a ideia de computação em *grid*. O usuário faz a requisição de um certo serviço para o ambiente e o mesmo se encarrega de encontrar o(s) nodo(s) que possui (possuem) o serviço. O ambiente utiliza uma política universalmente aceita para selecionar os recursos compartilhados como: *cluster*, organização virtual e supercomputadores.

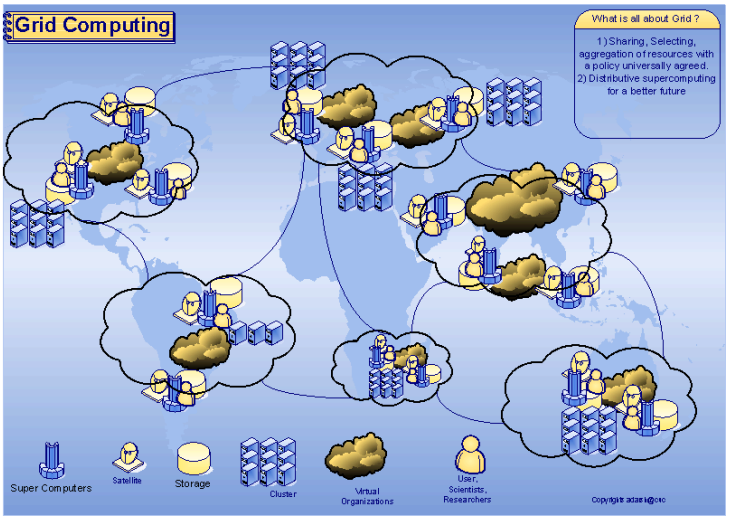

Figura 2 – Ambiente *grid*
Fonte: *Adarsh Grid Computing Research* (2014)

2.1.3 Canal de Comunicação

Em comunicação de dados, um canal de comunicação ou simplesmente “canal”, refere-se ao meio utilizado para transmitir e concretizar a comunicação entre o emissor e o receptor. A comunicação pode ser efetuada utilizando um meio físico de transmissão, como um fio, ou uma conexão lógica através de um meio multiplexado, com um canal de rádio (VALKENBURG, 1974).

Um canal é utilizado para transmitir um sinal de informação, por exemplo, uma sequência de bits digitais a partir de um emissor (podendo ser um ou vários) para o receptor (de igual modo, um ou vários). A transmissão de dados a partir de um local para outro requer algum meio/canal, seja cabo, fibra óptica, micro-ondas, satélite, rádio, entre outros. Cada canal possui uma certa capacidade para a transmissão de informação, e muitas vezes tal medida é dada pela sua largura de banda, a qual pode ser limitada pela latência (VALKENBURG, 1974).

Largura de banda é um termo utilizado para referenciar a capacidade de transmissão de um determinado meio, estipulando a velocidade que os dados trafegam pelo canal de comunicação utilizado para a transmissão de dados. A largura de banda é medida em *bits*, os quais determinam a capacidade de transmissão do meio por unidade de tempo, e na maior parte das vezes, a medida de tempo é o segundo. Quanto maior a banda, maior a velocidade de transmissão (VALKENBURG, 1974).

A latência de uma rede de comunicação de dados pode ser mensurada através do tempo demandado pelo envio de um pacote da fonte ao destino, ou o tempo requerido de ida e volta (o tempo da origem para o destino, somado ao tempo do destino de volta para a fonte). Pode-se dizer que é o intervalo de tempo entre o estímulo e a resposta. A latência mais utilizada é a de ida e volta, uma vez que a mesma pode ser medida de um único ponto (VALKENBURG, 1974).

2.1.4 Características

Ian Foster (2002) elaborou três características para definir o modelo de *grid computing*, os quais são:

- **Recursos coordenados que não se sujeitam a um controle centralizado**: sistemas em *grid* integram e coordenam recursos e usuários pertencentes a diferentes domínios;
- **Utilizar padrões abertos com interface e protocolos de propósito geral:** a utilização de protocolos e padrões abertos é essencial para que os sistemas *grid* possam realizar funções fundamentais como autenticação, autorização, descobrimento de recursos sem perder a capacidade de escalar e interagir com diferentes plataformas de *hardware* e *software;*
- **Prover o mínimo de qualidade de serviços:** o ambiente deve coordenar o uso dos recursos e promover a qualidade para diferentes tipos de serviços abrangendo segurança, tempo de acesso, disponibilidade entre outros.

Além de tais características elencadas por Ian Foster, existem alguns outros aspectos importantes em *grid computing*, os quais são (BUYYA, 2002):

- **Heterogeneidade:** um *grid* envolve uma multiplicidade de recursos que são heterogêneos, e envolvem uma grande variedade de tecnologia;
- **Escalabilidade:** um *grid* deve crescer de algumas dezenas de recursos para milhares de recursos sem perda de desempenho;

- **Compartilhamento de recursos:** os recursos de um *grid computing* não podem ser dedicados para nenhuma aplicação específica, devem estar disponíveis para qualquer usuário que o desejar;
- **Múltiplos domínios administrativos:** os recursos estão distribuídos geograficamente em múltiplos domínios, onde cada organização possui suas próprias restrições e regras de uso de recursos, que devem ser respeitadas;
- **Controle distribuído:** não existe um gerenciador centralizado que possui uma visão global do sistema. Assim, cada componente do *grid* deve ser autônomo;
- **Dinamicidade e adaptabilidade:** a falha de um recurso é uma regra. Portanto, as aplicações e gerenciadores de recursos devem mudar seu comportamento de acordo com a disponibilidade dos recursos.

2.1.5 Arquitetura

Um *grid* oferece algumas funcionalidades básicas como: armazenamento remoto, autenticação e autorização de acesso, transparência de acesso a recursos remotos entre outros. Foster e Kesselman (2004) apresentam uma proposta de arquitetura e componentes para prover tais funcionalidades em um *grid computting* (Figura 3), são eles:

- A camada de construção é o nível mais baixo, são os recursos e dispositivos físicos, cujo os usuários desejam compartilhar e acessar;
- A camada de conectividade e recursos é responsável pela comunicação e autenticação (validação de usuários e controle de operações de compartilhamento), a fim de garantir a troca de recursos com segurança;
- A camada de cooperação possui protocolos e realiza serviços responsáveis pelas transações de recursos (descoberta e alocação de recursos, replicação de dados, políticas de privilégios de usuários);
- A camada de aplicação é o mais alto nível, sua maior funcionalidade é invocar e prover o acesso às outras camadas.

Figura 3 – Arquitetura de *grid*
Fonte: (FOSTER; KESSELMAN, 2004).

2.1.6 Funcionalidades

O modelo *grid* pode ser classificado em três categorias, segundo as funcionalidades que oferecem (BAKER *et al*, 2002):

- ***Grids* de processamento:** esta categoria está relacionada com o poder de processamento na execução de aplicações. Subdivide os sistemas em computação de alto desempenho (*high performance*) e computação de alta vazão (*high throughput*). Enquanto o primeiro tem o objetivo de promover o desempenho máximo às aplicações paralelas, o segundo distribui as aplicações a fim de promover o desempenho do sistema como um todo. Um aspecto que deve ser considerado se tratando de *grids* de processamento é a implementação de estratégias de alocação de recursos, tanto na distribuição de aplicações como no uso dos recursos. Um escalonador de aplicação determina os recursos necessários para execução de uma aplicação e encaminha a requisição ao gerenciador de recursos. Portanto o papel de controlar os recursos a serem utilizados cabe ao gerenciador de recursos local. O escalonador somente faz uso dos recursos disponibilizados pelo gerenciador;

- ***Grids* de dados:** é formada por uma infraestrutura que fornece serviços de gerenciamento de dados (armazenamento e acesso) para as aplicações. Esta infraestrutura possui alguns protocolos que trabalham em função da transferência de arquivos. O *grid* precisa reunir informações espalhadas, sintetizá-las e torná-las disponíveis ao cliente como se fizessem parte da sua máquina local;
- ***Grids* de serviços:** uma infraestrutura que fornece serviços que não podem ser providos por apenas uma máquina. Um serviço computacional trata de qualquer recurso ou outro serviço que possa ser acessado remotamente e descrito através da interface de um provedor, a qual pode ser interpretada de forma automática por um cliente. Uma arquitetura baseada em serviços já é uma tecnologia de grande interesse na área computacional porque o uso de serviços possibilita a construção de infraestruturas para computação colaborativa sob demanda.

2.1.7 Componentes Básicos de um *Grid*

Com o objetivo de estabelecer uma padronização em *grid computing*, o *Global Grid Forum* (GGF) foi criado e especificou a OGSA (*Open Grid Services Architecture*) e a OGSI (*Open Grid Services Infrastructure*). A OGSA estabelece uma arquitetura para os serviços oferecidos pelo *grid*, definindo uma semântica uniforme para esses serviços (Open Grid Forum, 2014).

A OGSI define mecanismos para a criação, gestão e intercâmbio de informações entre as entidades chamadas serviços de *grid*. A OGSI trabalha em conjunto com o OGSA fornecendo a interação necessária ao ambiente de *grid*. Serão abordados e descritos de forma sucinta os principais serviços necessários para a construção de um *grid* (Open Grid Forum, 2014).

2.1.8 Segurança

Um dos componentes mais importantes de um *grid computing* é a segurança. Para que um usuário possa executar uma aplicação em *grid*, deve-se

garantir que outras aplicações ou usuários não terão acesso aos dados de sua aplicação. Da mesma forma, um recurso oferecido à aplicação do usuário não deve sofrer interferências em seus dados privados. Alguns requisitos mínimos para a segurança em grids são (BAKER *et al*, 2002):

- **Autenticação**: verifica se o usuário é realmente quem ele diz ser;
- **Autorização**: gerencia o acesso, ou seja, assegura que cada usuário ou computador que irá utilizar um recurso possua permissão para fazê-lo;
- **Confiabilidade dos dados**: remete a ocultar os dados e informações importantes;
- **Integridade dos dados**: preocupa-se em manter os dados irrepreensíveis, longe de qualquer alteração sem o consentimento.

2.1.9 Sistema de Monitoramento

O sistema de monitoramento funciona gerenciando os recursos e aplicações do ambiente grid, garantindo a dinamicidade do ambiente. Mantém conhecimento a respeito da disponibilidade dos recursos e sua utilização corrente.

O monitoramento de estados e da utilização de recursos, pode ser utilizado tanto para auxiliar nas decisões do escalonador de aplicações, como para calcular o quanto um usuário faz uso de uma estrutura (BAKER *et al*, 2002).

2.1.10 Gerenciamento de Recursos

O sistema de gerenciamento de recursos é responsável por administrar os recursos disponíveis e associá-los às aplicações. Atua como uma interface, provendo facilidades na alocação de recursos, informação a respeito do estado da execução da aplicação. Provê meios de cancelar, parar a execução de uma aplicação e outros níveis de gestão. A organização dos computadores no *grid* pode influenciar no padrão de comunicação de um sistema de gerenciamento de recursos (BAKER *et al*, 2002).

2.1.11 Gerenciamento de dados

Se tratando de *grid* de dados, cujo o objetivo é agrupar dados distribuídos, as aplicações não requerem processamento. Logo, o sistema de gerenciamento de dados pode conter apenas serviços essenciais, como o de transferência, descobrimento, manipulação, criação e gerenciamento de cópias de dados.

Por exemplo, os dados produzidos e armazenados no CERN [Frances: *Conseil Européen pour la Recherche Nucléaire*] (Organização Europeia de Pesquisa Nuclear) são distribuídos para centros regionais de várias partes do mundo e, são repassados para centros de pesquisa e instituições.

2.1.12 Escalonamento

O gerenciamento da utilização de recursos é de grande importância em *grid computing*, os recursos precisam ser geridos de maneira a reduzir o tempo de execução das aplicações e aumentar a quantidade de execuções no sistema. O escalonamento relaciona recursos e aplicações de modo a otimizar uma única aplicação e o ambiente como um todo.

2.1.13 Notoriedade

A computação em *grid* vem ganhando destaques nos últimos anos. No meio científico, pode-se encontrar vários *grids* em funcionamento espalhadas por inúmeros países.

Como exemplo, há o Datagrid, do CERN, que é um projeto financiado pela Comunidade Europeia, com o objetivo de atuar em áreas de pesquisa como astronomia, física e biologia (CERN, 2014).

No Brasil, um bom exemplo é o SPRACE (São Paulo *Research and Analyzes Center*) (SPRACE, 2014), projeto implementado em 2003, que participa do processamento, armazenamento e análise dos dados provenientes do projeto D0 (DZero, 2014).

O projeto D0 reúne pesquisadores do mundo todo para analisar os dados gerados pelo acelerador de alta energia *Tevatron Collider*, localizado em *Illinois,* Estados Unidos (DZero, 2014).

2.1.14 Ciclo de Vida

Os *grids* podem ser permanentes ou temporários, podem ser formados para executar uma tarefa específica e, em seguida, serem desfeitos. Assumindo que todos os computadores estejam previamente ligados em rede.

A criação e dissolução do *grid* é apenas questão de ativar e depois desativar o software responsável em cada computador, disponibilizando assim, toda a capacidade de processamento do dispositivo para o dono.

2.1.15 Organização Virtual

A organização virtual é uma agregação de organizações autônomas e, independentes ligadas entre si por intermédio de uma rede de comunicação. A existência de tal organização limita-se a um período de tempo, correspondente ao tempo necessário à satisfação do seu propósito (IOSUP; EPEMA, 2011).

O conceito de *grid computing* permite que diversas entidades disponibilizem seus recursos no âmbito de uma organização virtual, de forma a otimizar sua utilização. Tal possibilidade permite a viabilidade de projetos que, de outro modo, seriam inviáveis em termos financeiros ou de complexidade técnica. A computação em *grid* é uma forma de computação distribuída que, permite a partilha e coordenação de recursos para a resolução de problemas complexos (IOSUP; EPEMA, 2011).

No entanto, a possibilidade de inserir novos usuários e recursos no *grid* de forma dinâmica, em paralelo com a necessidade de cada administrador manter o controle sobre os seus recursos, incrementa substancialmente a complexidade na gestão dos mesmos recursos (CASANOVA *et al*, 2008).

2.2 SIMULAÇÕES E SIMULADORES

A questão chave em computação distribuída é avaliar cientificamente a qualidade de várias soluções, no que diz respeito a uma determinada métrica (teste de heurísticas, de escalonamento, probabilidade de disponibilidade de um serviço, tempo de respostas das consultas de pesquisas) (CASANOVA *et al*, 2008).

Em pouquíssimos casos tal avaliação pode ser abordada através da análise teórica. Portanto, a maior parte dos resultados de pesquisas, são obtidos da avaliação empírica através de experimentos (CASANOVA *et al*, 2008).

Uma abordagem óbvia para a obtenção de dados experimentais válidos, é a realização de experimentos em ambientes/plataformas reais de produção. Infelizmente, tal alternativa é inviável. Ambientes reais não estão disponíveis para a realização de experiências, de modo a não interromper o seu uso.

Além disso, mesmo que tais espaços estivessem disponíveis, experiências em ambientes reais podem ser consideradas muito difíceis de serem realizadas, especialmente se um grande número de experimentos é necessário para explorar diversos cenários, com significância de estatística. Diante de tais dificuldades, surge a necessidades por parte dos pesquisadores de obter os resultados e avaliações por meio de simulações.

A simulação surge como uma maneira viável de analisar algoritmos em larga escala de sistemas distribuídos sobre recursos heterogêneos. Ao contrário de usar o sistema existente, em tempo real, a simulação funciona sem necessitar de um complexo mecanismo de análise, evitando a sobrecarga de coordenação de recursos reais.

Tal metodologia também é eficaz no trabalho com grandes problemas hipotéticos que requerem o envolvimento de um grande número de usuários e recursos ativos, o qual é difícil de coordenar e construir em ambiente de pesquisa em larga escala (BUYYA; MURSHED, 2002).

Legrand (2006) define simulação como a tentativa de prever aspectos do comportamento de algum sistema, criando um modelo aproximado do mesmo.

2.2.1 Dificuldades

A inerente natureza do *grid,* ou seja, a heterogeneidade de *hardware* e *software*, manipulação de amplos recursos e o número de dispositivos envolvidos desafiam os pesquisadores e os simuladores de *grid*. Alguns desafios ao *grid* e consequentemente aos simuladores são (BUYYA; MURSHED, 2002):

- **Confiabilidade do *grid***: quando uma falha atinge uma parte do *grid*, os demais dispositivos podem continuar operando normalmente.
- **Agendamento de tarefas**: agendamento de tarefas para cada dispositivo que compõe o *grid*;
- **Balanceamento de carga**: política para prover uma melhor distribuição na utilização de recursos, minimizando a ociosidade e utilização excessiva dos dispositivos;
- **Monitoramento de recursos**: obtêm informações dos dispositivos, com tais informações é possível monitorar o ambiente e fornecer aos usuários, detalhamento da máquina na qual irá submeter suas tarefas;

Os desafios não são limitados aos descritos acima. Existem vários outros desafios como a falta de padronização para simulações, custo oculto, definição e alcance da computação em *grid* (LEGRAND, 2006).

2.2.2 GridSim

GridSim é um conjunto de ferramentas de simulação que se utiliza da biblioteca SimJava baseada em *Java* (SABHARWAL, 1998). Java, por sua vez, é uma linguagem de programação criada na década de 90, sua principal característica é a orientação a objeto. Diferentemente de outras linguagens de programação, as quais são compiladas para serem executadas diretamente no computador, a linguagem *java* é compilada para um *bytecode* que é executado por uma máquina virtual, denominada *Java Virtual Machine* (JVM) (SABHARWAL, 1998).

O simulador GridSim oferece facilidades para a modelagem e simulação de recursos, diferentes capacidades, configurações e domínios. As capacidades mais salientes deste conjunto de ferramentas são as modelagens de recursos

heterogêneos, multitarefas, provisão de recursos agendados e modelos de aplicação paralela (BUYYA; MURSHED, 2002).

As principais funcionalidades do GridSim são (BUYYA; MURSHED, 2002):

- Capacidade de simular serviços computacionalmente intensivos, como também serviços que usam e transferem grande quantidade de dados;
- Possibilidade de modelar recursos heterogêneos;
- Suporte a catálogos hierárquicos, réplicas e políticas de alocação.

Tais funcionalidades fazem o GridSim um bom simulador para avaliar o desempenho de vários algoritmos de escalonamento ou, heurísticas de recursos com base no prazo e, as restrições com base no orçamento (BUYYA; MURSHED, 2002).

2.2.2.1Características

A grande desvantagem desta ferramenta é a execução de cada tarefa do *grid* como um segmento separado no *Java Virtual Machine*, fazendo o uso excessivo de *threads* (SABHARWAL, 1998), limitando a escalabilidade e restringindo o número máximo de máquinas simuladas (DEPOORTER *et al*, 2008). Em ciência da computação, *thread* é definido como um pequeno conjunto de instruções programadas que podem ser gerenciadas de forma independente. Em resumo, é uma forma do processo dividir a si mesmo em tarefas (SABHARWAL, 1998).

2.2.2.2Componentes

A arquitetura do GridSim, mostrado na Figura 4, é construído como um conjunto de componentes em camadas, as quais são (BUYYA; MURSHED, 2002):

- A base da arquitetura é a biblioteca SimJava, usada para a comunicação entre os componentes GridSim;
- A segunda camada é formada pelos principais elementos utilizados para modelar os recursos do *grid*;
- A terceira e quarta camada, modelam serviços computacionais específicos e dados do *grid*;

- A quinta e sexta camada, consistem de componentes para ajudar o usuário na implementação, teste de algoritmos, definição de cenários entre outras funcionalidades do GridSim.

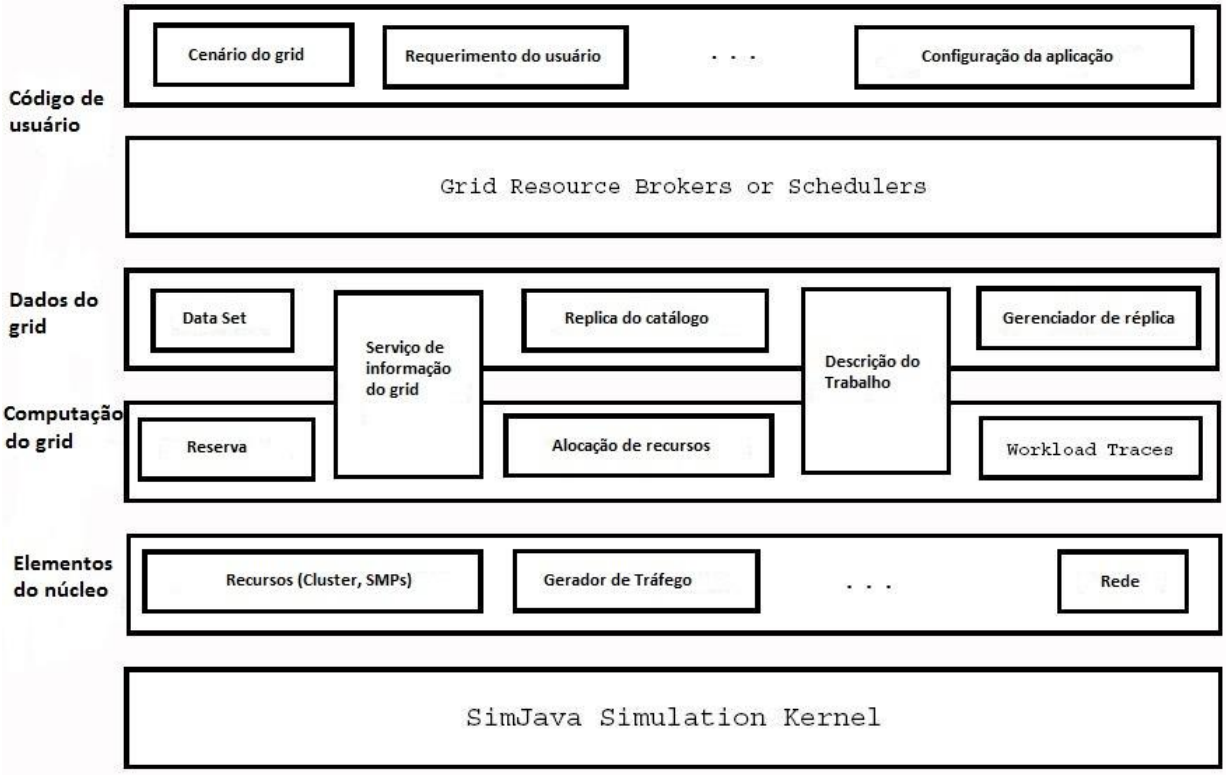

Figura 4 – Arquitetura do GridSim
Fonte: *The GridSim Toolkit* (2014)

2.2.3 CloudSim

A computação em nuvem é uma tecnologia emergente que transformará grande parte da indústria de tecnologia da informação (FARIA, 2012). Tal tecnologia define infraestrutura, plataformas e *software* como serviços, os quais podem ser acessados e utilizados por clientes. Alguns aplicativos baseados em nuvem como as redes sociais já estão sendo desenvolvidos com sucesso.

O CloudSim é um simulador para nuvem o qual utiliza GridSim como base. A nuvem é um paradigma diferente de *grid*, uma das principais diferenças é o provimento dinâmico de máquinas virtuais, para a computação na nuvem (FARIA, 2012). O CloudSim providencia para o usuário:

- Aplicação de teste em um ambiente fechado e repetível;
- Localização de gargalos do sistema;

- Modelagem e simulação de centros de dados em grande escala;
- Inserção de elementos de simulação dinâmica;
- Políticas definidas pelo usuário para a alocação de máquinas virtuais.

2.2.3.1Características

Dados os problemas de desempenho do GridSim, e o uso excessivo de *threads* pelas entidades do SimJava, tentou-se minimizar o número de entidades no CloudSim, porém, o simulador ainda apresenta uso elevado de *threads*, além da alocação exponencial de memória (FARIA, 2012).

O CloudSim é implementado usando bibliotecas e estruturas de simulação, a fim de lidar com os requisitos de baixo nível do sistema, como o GridSim utiliza a biblioteca SimJava existente. Os componentes de manipulação de eventos e mensagens que o SimJava utiliza, também podem ser reutilizados no GridSim (FARIA, 2012).

2.2.3.2Componentes

Os principais componentes da infraestrutura da computação em nuvem, os quais são implementados no CloudSim, demonstrados na Figura 5, são:

- ***Deta center:*** no CloudSim, *data center* é usado para modelar os serviços básicos ao nível do sistema de infraestrutura. Consistem em um conjunto de *hosts* que geram um conjunto de máquinas virtuais, cujas tarefas são os processamentos de baixo nível, e pelo menos um *deta center* deve ser criado para iniciar a simulação;
- ***Host:*** este componente é usado para atribuir capacidades de processamento (é especificado em milhões de instruções por segundo que o processador poderia executar), memória e uma política de escalonamento para alocar diferentes números de processamento em várias máquinas virtuais;
- ***Virtual Machine:*** este componente gerencia a alocação de diferentes máquinas virtuais, diferentes *hosts*, de modo que os núcleos de processamento possam ser agendados (pelo *host*) para as máquinas

virtuais. Tal configuração depende da aplicação, e a política padrão de alocação de máquinas virtuais;

- ***Datacenter broker:*** a responsabilidade do *broker* é ponderar entre usuários e prestadores de serviços, de acordo com a exigência de qualidade de serviço que o usuário especifica. O *broker* deve identificar qual o provedor de serviço é adequado para o usuário com base nas informações da nuvem, e assim negociar com os fornecedores, sobre os recursos que satisfaçam os requisitos do usuário;
- ***Cloudlet:*** este componente representa o serviço de aplicação.
- ***CloudCoordinator:*** gerencia a comunicação entre outros serviços *CloudCoordenators* e corretores e, monitora o estado interno de um centro de dados que irá ser feito periodicamente, em termos de tempo de simulação.

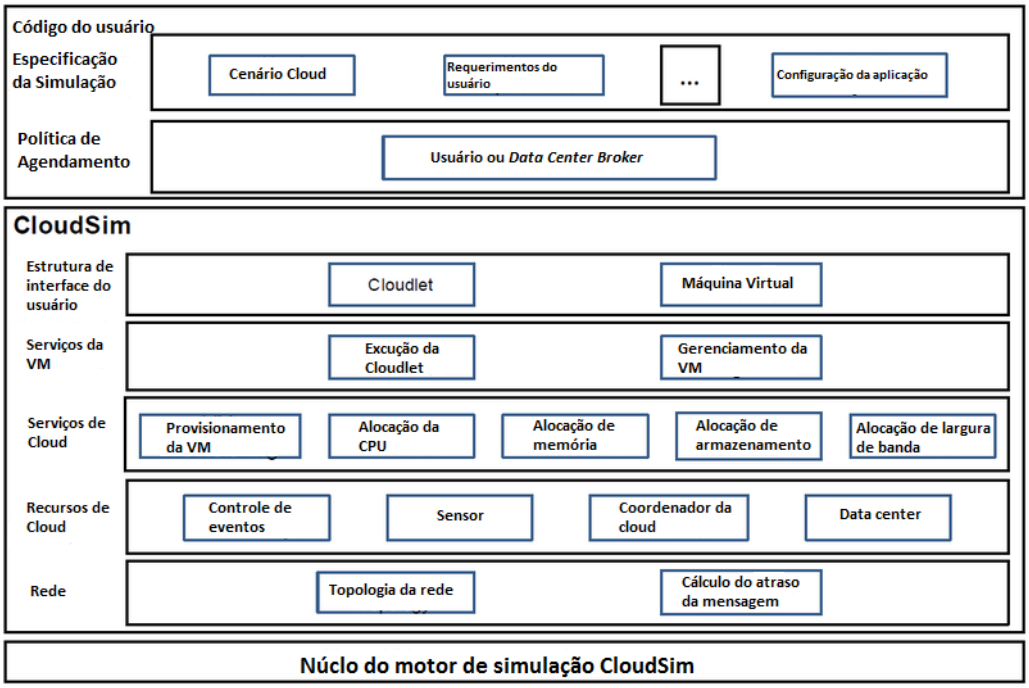

Figura 5 – Componentes do CloudSim
Fonte: *Cloud-Simulation-Frameworks* (2014)

2.2.4 SimGrid

O projeto SimGrid iniciou-se em 1999, com a finalidade de permitir o estudo de algoritmos de escalonamento para plataformas heterogêneas. A versão 1 do

SimGrid facilitou para protótipos de heurística de escalonamento, testes em variadas aplicações expressado em gráficos de tarefas (CASANOVA, 2001).

A partir de 2003, foram lançadas novas versões, estendendo as capacidades do simulador. Adicionou-se mecanismos para estudar a programação não-centralizado e, outros tipos de processos sequenciais. O motor de simulação foi reescrito visando uma melhor modularidade, velocidade e escalabilidade, suportando recursos dinâmicos e falhas (CASANOVA, 2001).

2.2.4.1Características

Algumas características chaves do SimGrid são (CASANOVA *et al*, 2008):

- Um motor de simulação escalável e extensível, o qual implementa vários modelos de simulações, tornando possível simular topologias arbitrárias de rede, computação dinâmica, recursos de redes disponíveis e falhas de recursos;
- Interfaces de alto nível para os pesquisadores de computação distribuída, facilitando e agilizando as simulações;
- Mecanismos para desenvolvedores simularem suas aplicações, o mais próximo das condições encontradas em ambientes reais.

2.2.4.2Componentes

Os principais componentes implementados no SimGrid são demonstrados na Figura 6, os quais são listados a seguir (CASANOVA *et al*, 2008):

- ***SimDag:*** provido desde a versão 1, é projetado para simular heurísticas de escalonamento para aplicações estruturadas. É possível criar tarefas com dependências, recuperar informações sobre a plataforma e computar o tempo de execução;
- ***MSG:*** adicionada na versão 2, permite o estudo de aplicações do SimGrid;
- ***GRAS (Grid Reality and Simulation):*** permite o desenvolvimento de aplicações distribuídas (ex. serviço de informações sobre recursos) dentro do simulador, o mais próximo do ambiente real;

- ***SMPI:*** habilita a simulação direta de aplicações que utilizam paralelismo;
- ***XBT:*** usado em todo o *software*, implementa um conjunto clássico de dados, mecanismo de registro e de execução, suporte para configuração e portabilidade;
- ***SURF:*** provê um mecanismo de simulação;
- ***SIMIX:*** módulo interno que fornece mecanismos sob o *SURF*, o qual facilita a simuladores implementação e abstração de vários processos simultâneos;

SMURF: permite a distribuição de processos simulados, aproveitando a memória de vários computadores. Tal componente, permite melhorar a escalabilidade do SimGrid.

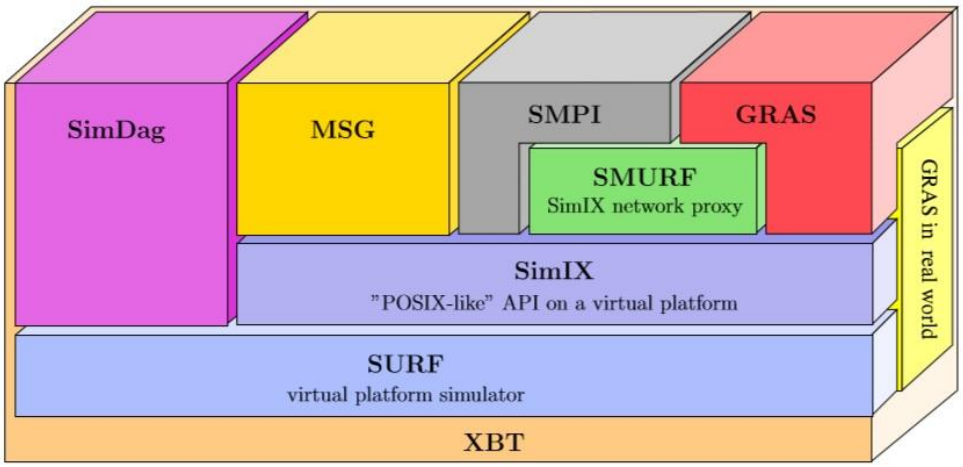

Figura 6 – Componentes do SimGrid
Fonte: (CASANOVA *et al*, 2013)

3 DESENVOLVIMENTO

Neste capítulo, são apresentadas as ferramentas empregadas para a criação da simulação que constitui o objetivo principal deste trabalho. A metodologia adotada é explorada através de análises descritivas das variáveis produzidas pelo simulador, além de uma descrição dos ambientes a serem simulados.

3.1 FERRAMENTAS

A execução deste trabalho foi realizada unicamente através da utilização de ferramentas de *software* de código aberto. Tal característica dá ao programador total liberdade para modificação e adaptação de seus componentes, caso os mesmos não atendam integralmente as necessidades do projeto.

3.1.1 Escolha do Simulador

Após pesquisar as principais ferramentas existente de simulações de *grids*, e descrever os principais componentes, pontos fortes e fracos, como também a melhor utilização, escolheu-se a ferramenta mais adequada que auxiliou o desenvolvimento deste trabalho, simulando todos os modelos criados com as características desejadas.

Como descrito no capítulo anterior, o simulador GridSim permite o estudo de aplicações em *grid*, entretanto, possui diversos problemas de escalabilidade, restringindo o número máximo de máquinas simuladas (DEPOORTER *et al*, 2008). Por outro lado, o simulador CloudSim que também permite o estudo de aplicações em *grid*, possui problemas de escalabilidade, que resultam em uso intensivo de *threads* e alocação exponencial de memória (FARIA, 2012).

Entretanto o simulador SimGrid, além de não possuir os problemas encontrados nos simuladores descritos acima, disponibiliza componentes para auxiliar a criação de ambientes com recursos heterogêneos e um *kernel* modular, que permite a adição e uso de novos recursos sem mudanças no código do usuário (CASANOVA *et al*, 2013).

O Quadro 1 resume as principais diferenças entre as ferramentas de simulação pesquisadas e discutidas.

Simuladores	**Auxilio para simulações**	***Kernel* modular**	**Consumo de memória**	**Consumo do processador**	**Latência**
GridSim	Possui	Não possui	Alto	Alto	Configurável
CloudSim	Possui	Não possui	Médio	Alto	Não configurável
SimGrid	Possui	Possui	Baixo	Baixo	Configurável

Quadro 1 – Comparação dos simuladores de *grids*
Fonte: Autoria Própria.

O simulador utilizado no desenvolvimento deste trabalho visou a realização de simulações de ambiente de computação distribuída, cujo a virtualização desempenha uma importante função. Tal ambiente é característico do *grid computing*, cuja as máquinas virtuais são criadas e executadas pelos computadores que o compõe, e atendem a usuários em escala global.

Para a realização das simulações deste trabalho, o simulador que melhor atendeu as necessidades foi o SimGrid. O mesmo é resultado de um projeto com mais de 10 anos de existência (CASANOVA, 2001), possuindo a capacidade de simular 2.000.000 de nós em uma única máquina com 16Gb de memória. Além disso, o SimGrid simula diversas condições encontradas em uma infraestrutura de rede como latência e largura de banda (CASANOVA *et al*, 2008). A versão utilizada do SimGrid neste projeto é a 3.10.

3.1.2 Sistema Operacional

O simulador escolhido para auxiliar no desenvolvimento deste trabalho foi instalado no sistema operacional Debian 7.6 desenvolvido pelo projeto Debian. Tal projeto é composto por uma associação de pessoas que possuem uma causa comum: criar um sistema operacional livre. Debian é uma das distribuições Linux mais populares para computadores pessoais e servidores de rede, e tem sido utilizado como base para várias outras distribuição Linux (DEBIAN, 2014).

3.1.3 VirtualBox

Virtualbox é um aplicativo (um pacote de *software*) de virtualização que pode ser instalado em diversos tipos de sistemas operacionais. Foi criado por innotek GmbH e comprado pela *Sun Microsystems*. Tal aplicativo é instalado em um sistema operacional que não está virtualizado, e estende as capacidades deste computador, permitindo através da virtualização que, carregue e execute diversos outros tipos de sistemas operacionais e seus respectivos aplicativos, podendo estes serem executados ao mesmo tempo (VIRTUALBOX, 2014).

Para o presente trabalho utilizou-se o VirtualBox versão 4.3.16, disponibilizou-se 2GB ddr3 1060mhz de memória, 4 núcleos físicos do processador intel i7 2630QM para processamento, 10GB de espaço em disco rígido e 128MB de memória para aceleramento gráfico. O aplicativo foi instalado sob o sistema operacional Arch Linux.

3.1.4 A Instalação do Simulador

O simulador escolhido para auxiliar no desenvolvimento deste trabalho foi o SimGrid. O mesmo foi instalado sob o sistema operacional Debian que por sua vez está virtualizado pelo aplicativo VirtualBox. Os progrmas (dependências) necessários para a instalação do simulador são:

- *Build-essentials;*
- *Perl* e *libpcre;*
- Compilador *C* e *C*++;
- Ccmake;
- Tar;
- Plataformas: Windows, MacOS e Linux.

Para a instalação efetuou-se os seguintes passos:

1. Utilizou-se o repositório de pacotes do Debian para instalar as dependências listadas a cima;
2. Após a instalação das dependências, utilizou-se novamente o repositório de pacotes do Debian para a instalação do SimGrid:
    a. sudo apt-get install simgrid;

3. Após a instalação, efetuou-se o *download* da versão 3.10 do simulador SimGrid no site do desenvolvedor (http://simgrid.gforge.inria.fr/), uma vez que a versão disponibilizada para instalação no repositório de pacotes do Debian era a 3.7;
4. Instalou-se a nova versão do simulador com os seguintes comandos:
    a. tar xf SimGrid-3.10.tar.gz
    b. cmake -DCMAKE_INSTALL_PREFIX=<path> ./
    c. make
    d. make install
5. Atribuiu-se as variáveis do sistema como segue:
    a. export SIMGRID_ROOT="/usr/local"
    b. export SIMGRID_JAVA_ROOT="/usr/local"
    c. export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$SIMGRID_ROOT/lib:$SIMGRID_JAVA_ROOT/java

3.1.5 Linguagem XML

A linguagem utilizada pelo SimGrid para criar e especificar os ambientes de simulação é o *XML* (*Extensible Mark-up Language*) (XML, 2014). *XML* é uma linguagem de marcação, a qual define um conjunto de regras para criar documentos em um formato que humanos e máquinas possam entender (XML, 2014).

O objetivo do projeto *XML* é enfatizar simplicidade, generalidade e usabilidade sobre a *internet*. É um formato estrutural de dados com forte suporte para diferentes linguagens humanas. A *W3C* (*World Wide Web Consortium*) é a principal organização internacional de padronização para a *internet*, e recomenda o uso do *XML* para gerar linguagens de marcação com necessidades e características especiais (XML, 2014).

Uma vantagem do uso do *XML* é que seus campos não são pré-definidos. Assim, propiciando liberdade ao programador e à aplicação para fazer uso da linguagem para criar e definir campos que lhe convêm.

3.2 SIMULAÇÃO

Como já descrito neste trabalho, *grid computing*, é um modelo que utiliza recursos ociosos de computadores independentes (FOSTER *et al*, 2002). Além disso, é possível aplicá-lo em diferentes plataformas de processamento (exemplo: computadores e dispositivos móveis) localizados em diferentes países.

Tal característica, agrega ao modelo um alto nível de heterogeneidade, tanto de *hardware* quanto de *software*. Memórias e processadores distribuídos (compartilhados) podem ser heterogêneos em diversas formas. Podem ser de diferentes arquiteturas. Podem ser da mesma arquitetura, mas de diferentes modelos. Podem ser da mesma arquitetura e modelo, mas rodar diferentes instruções de processamento. Podem ser da mesma arquitetura, modelo e rodar as mesma instruções de processamento, mas usar diferentes programas básicos (compiladores, bibliotecas em tempo de execução, etc.) (DONGARRA; LASTOVETSKY, 2006).

A complexidade da heterogeneidade é ainda maior pelo fato das redes que interconectam os recursos dos dispositivos não serem homogêneas. Os canais de comunicação providos pela rede não possuem as mesmas características entre quaisquer pares dos recursos compartilhados do *grid*. As redes podem ser heterogêneas em latência, largura de banda, propósito de sua implantação, meio para o transporte dos dados, protocolos de comunicação e etc. (DONGARRA; LASTOVETSKY, 2006).

Todas essas diferenças impactam diretamente na performance do *grid*, aumentando a complexidade do modelo e fazendo com que cada ambiente *grid* possua suas particularidades em capacidade de processamento, transferência de dados e estrutura operacional.

Inerente a heterogeneidade do *grid*, está a dificuldade de especificar-se o ambiente ou modelo que melhor representa um *grid computing* real. Segundo Legrand (2006) não existe uma padronização e, nenhuma definição do melhor ambiente para simulações de *grid computing*.

Tendo essas questões como primícias, utilizou-se o paradigma mestre-escravo (LIU; SCHMIDER, 2011) e, buscou-se a máxima heterogeneidade na especificação da capacidade de processamento e, infraestrutura de comunicação pertencentes aos dispositivos que compõe o modelo da simulação do *grid*.

O simulador SimGrid utilizado no desenvolvimento deste trabalho, utiliza três arquivos para especificar o modelo a ser simulado. São eles:

- Código fonte: deve conter a implementação do paradigma desejado para as políticas de computação dos dados;
- Plataforma: especifica as capacidades de processamento e a infraestrutura pertencentes aos dispositivos da simulação;
- Implantação: especifica as propriedades das tarefas a serem computadas como também os dispositivos que irão compor o modelo (mestres e escravos).

3.2.1 Paradigma Mestre-Escravo

Para que um programa seja executado em paralelo, é necessário paralelizar tarefas que possuem fragmentos de códigos independentes, os quais permitem a paralelização e processamento fragmentado. Essas tarefas (trechos de código), são partes de problemas maiores que também podem ser processados independentemente, e trocam informações de tempo em tempo, de acordo com a sincronização dos processos. A comunicação dos processadores é avaliada pela granularidade (LIU; SCHMIDER, 2011).

A granularidade é a quantidade de processamento realizado por cada processador, em relação à quantidade de comunicação realizada entre os processadores. Quando os processadores executam poucas instruções e comunicam muito é dito que o programa é muito granular, caso contrário, é pouco granular (LIU; SCHMIDER, 2011).

O paradigma utilizado em computação paralela é chamado de *Dividir para Conquistar*. O mesmo consiste em distribuir uma grande tarefa em dois ou mais subproblemas para cada processador, de forma que, os subproblemas admitam resolução independente, e seus resultados parciais possam ser combinados chegando ao resultado final. Esses subproblemas são instâncias do problema original (LIU; SCHMIDER, 2011).

Um dos princípios básicos da computação paralela é evitar os *workloads imbalance* (tradução direta: desequilíbrio de cargas de trabalho). Se as tarefas de computação são distribuídas entre um número de processos independentes e, existe

uma chance das tarefas se diferirem substancialmente em seus tempos de execução, se faz necessário atentar-se para que as alocações de tarefas não sejam feitas estaticamente. Se esta regra não for seguida, alguns dos processos podem gastar muito tempo esperando obter mais trabalho (LIU; SCHMIDER, 2011).

O modelo mestre-escravo de computação paralela consiste em duas entidades: processador mestre e múltiplos processadores escravos. O mestre deve realizar a decomposição do problema em pequenas tarefas, distribuir essas tarefas entre os escravos e aguardar o recebimento dos resultados parciais. Tais resultados quando coletados em sua totalidade, compõe a solução do problema (LIU; SCHMIDER, 2011).

Os escravos recebem as mensagens com a tarefa, processam a tarefa, enviam o resultado ao mestre e obtém uma nova tarefa. Se os tempos de comunicação para a sinalização do mestre, e fornecimento de escravos com novas tarefas são pequenos se comparado com o tempo de execução da tarefa, este modelo está evitando *workload imbalances*. Geralmente a comunicação é efetuada apenas entre mestres e escravos (LIU; SCHMIDER, 2011). Uma representação deste modelo pode ser visualizada na Figura 7.

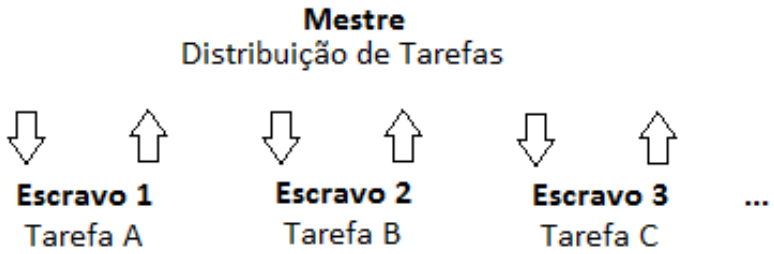

Figura 7 – Representação Mestre Escravo
Fonte: Autoria própria.

3.2.2 Especificação da Plataforma

Uma vez que o código fonte tenha sido desenvolvido com o paradigma desejado e, escolhido os recursos dos dispositivos, se faz necessário:

1. Instanciar cada dispositivo com os recursos e parâmetros escolhidos e apropriados;
2. Descrever/definir as interconexões entre os recursos.

A instanciação dos recursos dos dispositivos é realizado especificando alguns parâmetros. Para criar um dispositivo de processamento e especificar a capacidade computacional do processador no modelo de simulação, basta acrescentar uma linha no arquivo definindo o identificador da máquina através do marcador *host id,* e sua capacidade de processamento com o marcador *power*. As medidas de processamento devem ser especificadas em FLOPS ou MIPS. Todas as especificações da plataforma são feitas utilizando a linguagem XML. O Quadro 2 demonstra a criação e especificação de um dispositivo identificado como “Paul” e, com o poder computacional de 98.652Mf (Mega FLOPS):

```
<host id=”Paul” power=”98.652Mf”/>
```

Quadro 2 – Especificação de Dispositivos de Processamento
Fonte: Autoria Própria.

Em computação, FLOPS é uma sigla para operações de ponto flutuante por segundo (*FLoating point OPerations per Second*). O FLOPS é uma medida de desempenho de um computador, especialmente em áreas de cálculos científicos, os quais fazem uso intenso de ponto flutuante (DELL, 2012).

Um ponto flutuante é um método de codificação de números reais dentro dos limites de precisão finita disponíveis em computadores. É possível calcular o FLOPS, multiplicando o número de núcleos físicos do processador pela sua frequência, pelo seu FLOPs e dividir pelo o seu ciclo. O Quadro 3 demonstra a fórmula para o cálculo de FLOPs (DELL, 2012):

```
FLOPS = (núcleos x frequência x FLOPs)/ciclos
```

Quadro 3 – Especificação do cálculo de FLOPS
Fonte: Autoria Própria.

Instruções por segundo (IPS) é uma medida de velocidade do processador de um computador. É possível encontrar na literatura pesquisadores relatando valores de IPS para representar “picos” de execução de instruções. O termo IPS é comumente usado em associação com um valor numérico, como milhões de instruções por segundo (MIPS) (DELL, 2014).

Para criar e definir os recursos de rede estruturais que ligarão os dispositivos e irão compor o modelo estrutural do *grid,* basta também acrescentar uma linha no arquivo e definir o identificador da rede através do marcador *link id*, a latência com o marcador *latency* e a largura de banda com o marcador *bandwidth*.

O identificador da rede pode conter letras e/ou números. Contudo, a especificação da latência e largura de banda, podem conter somente números associados a valores numéricos como "*µs*" (microssegundos para latência) e "MBps" (megabit por segundo para largura de banda). O Quadro 4 mostra um exemplo para a criação e especificação de um canal de comunicação utilizado na simulação. Tal canal é identificado no modelo como "*link1*", possui 10.234MBps de largura de banda e latência de 156.456us. No simulador SimGrid, a letra grega "µ" é representada pela letra "u".

```
<link id=”link1” bandwidth=”10.234MBps” latency=”156.456us”/>
```

Quadro 4 – Especificação do canal de comunicação
Fonte: Autoria Própria.

Após a definição dos dispositivos e dos canais de comunicação, se faz necessário definir as ligações topológicas desses elementos, ou seja, os caminhos de redes permitidos. Tais ligações tornarão possível o compartilhamento de recursos entre os dispositivos, as distribuições de tarefas como o recebimento dos resultados do processamento das mesmas, o balanço de carga e a criação de fato de um *grid computing*.

O Quadro 5 mostra como atribuir o canal de comunicação "*link1"* entre os dispositivos denominados "*host1"* e "*hosts2"*, criando assim a rota entre os dispositivos. Todos os elementos que participarão da atribuição, devem estar previamente criados e definidos com os seus respectivos valores/propriedades.

```
<route src=”host1” dst=”host2”>
<link:ctn id=”link1”/> </route>
```

Quadro 5 – Especificação da topologia
Fonte: Autoria Própria.

O arquivo final deve conter todo o conjunto de dispositivos e recursos, instanciados com as suas respectivas características/propriedades, as interconexões

topológicas e todos os elementos devem estar acessíveis à rede (*hosts,* roteadores, links e etc.)

3.2.3 Especificação da Implantação

O arquivo de implantação, necessário para executar a simulação, consiste basicamente em especificar qual processo irá ser executado, isto é, as propriedades do problema a ser computado e, quais os dispositivos que irão participar da simulação, recebendo as tarefas para serem processadas.

A especificação do arquivo deve seguir uma ordem exigida pelo simulador, como segue:

1. O primeiro argumento deve definir o número de tarefas que deverão ser processadas;
2. O segundo argumento deve definir o tamanho em processamento de cada tarefa especificada anteriormente, isto é, quantas instruções serão exigidas do processador para processá-la;
3. O terceiro argumento deve definir qual o tamanho da comunicação das tarefas, neste argumento é possível definir se o problema a ser executado será muito granular ou não.
4. A partir do quarto argumento, deve-se especificar todos os dispositivos que participarão da execução da tarefa (escravos). É importante ressaltar que, tais dispositivos devem já estar especificados no arquivo da plataforma, descrito anteriormente.

O Quadro 6 mostra como especificar o arquivo de implantação, seguindo a ordem definida acima. Neste exemplo, está sendo criado um problema com 20 tarefas a serem executadas e paralelizadas, é exigido 50.000.000 instruções do processador para processar cada tarefa das 20 especificadas. O tamanho da comunicação entre as tarefas é de 1.000.000 e os escravos definidos para receberem e processar as tarefas vão do *host1* ao *host5*.

```
    <argument value="20"/>
    <argument value="50000000"/>
    <argument value="1000000"/>
    <argument value="host1"/>
    <argument value="host2"/>
    <argument value="host3"/>
    <argument value="host4"/>
    <argument value="host5"/>
```

Quadro 6 – Especificação da implantação
Fonte: Autoria Própria.

3.2.4 Especificação da Plataforma no Trabalho

Os parâmetros de poder de processamento dos dispositivos, ou de especificação do canal de comunicação, podem ser definidos como constantes ou amostra de probabilidade relevantes ao propósito da simulação. Assim, é possível especificar qualquer modelo de simulação com ambientes particulares de computação, e estudar as variáveis produzidas pelo mesmo.

Para especificar a plataforma de simulação no presente trabalho, cujo objetivo é estudar o impacto que o canal de comunicação tem no desempenho de um *grid computing* e, sabendo que não existe um modelo que especifique tal ambiente ou que melhor se aproxime do mesmo (LEGRAND, 2006), distribuiu-se valores aleatoriamente, sem aplicar qualquer tipo de amostra de probabilidade.

Tais valores foram especificados tanto para o poder de processamento dos dispositivos, quanto para as definições do canal de comunicação (latência e largura de banda). O principal objetivo da distribuição aleatória para os parâmetros, é especificar um *grid computing* com o maior grau de heterogeneidade e, representar os mais variados dispositivos que podem fazer parte do modelo.

O modelo de simulação é composto por 191 canais de comunicação que interligam todos os elementos e, garantem que todos os escravos participem das computações da tarefa. Os canais de comunicação não foram organizados de forma lógica a fim de representar qualquer tipo de topologia, buscou-se apenas a ligação dos escravos entre si e entre o mestre.

Para a especificação da largura de banda, definiu-se valores de 274kBps (kilobits por segundo) até 50MBps (megabits por segundo). Os valores baixos representados pela escala numérica kBps, foram definidos com o objetivo de representar canais de comunicação utilizados por dispositivos móveis. De igual modo, definiu-se valores para a latência, variando entre 6ms (milissegundos) até 900us (microssegundos). Os valores definidos possuem até quatro casas de precisão e representação numérica.

A definição dos parâmetros de poder de processamento pertencentes aos dispositivos (escravos), foram especificados com valores de 22Mf (mega FLOPs) até 137Gf (giga FLOPs). Os valores foram atribuídos de modo heterogêneo, com o objetivo de representar dispositivos de baixa capacidade computacional, até grandes máquinas com grande poder de computação. O modelo de simulação possui 98 dispositivos para computação das tarefas.

A especificação do problema a ser computado para todos os modelos de simulação, é composto por 100.000 tarefas, cada tarefa requer 50.000.000 de instruções para serem processadas e, possui o tamanho de comunicação de 1.000.000. É importante ressaltar que, todas as tarefas especificadas podem ser paralelizadas entre os 98 dispositivos de computação do modelo.

3.2.5 Estratégia de Simulação

Para atingir o objetivo deste trabalho e, estudar o impacto que o canal de comunicação tem no desempenho de um *grid*, criou-se diversos ambientes de simulação utilizando-se da seguinte estratégia:

1. Especificou-se um ambiente para simulação, visando a máxima heterogeneidade (como já descrito anteriormente) para os canais de comunicação e as capacidades computacionais dos dispositivos. Tal modelo foi chamado de “ambiente inicial”;
2. Mantiveram-se inalterados para todas as simulações criadas, todos os parâmetros do poder de processamento dos dispositivos, como também as tarefas a serem processadas, ou seja, todas as simulações computaram as mesmas tarefas e processaram com o mesmo poder computacional;

3. Conservou-se os valores dos parâmetros da largura de banda e aplicou-se um fator de correção (porcentagem) para todos os parâmetros de latência, especificado em cada *link* de comunicação. Para cada fator de correção aplicado, criou-se um novo ambiente de simulação, o qual foi executado a fim de contabilizar as horas necessárias para computar as tarefas;
4. Restaurou-se os valores inicias da latência e, aplicou-se um fator de correção (porcentagem) para todos os parâmetros da largura de banda, especificado em cada *link* de comunicação. Para cada fator de correção aplicado, criou-se um novo ambiente de simulação, o qual foi executado a fim de contabilizar as horas necessárias para computar as tarefas;
5. Restaurou-se os valores inicias de latência e largura de banda e, aplicou-se simultaneamente os fatores de correções antes utilizados individualmente para latência e largura de banda. De igual modo, após a aplicação e combinação dos fatores de correção, executou-se a simulação a fim de contabilizar as horas;

Além da estratégia para criar os modelos de simulação citada a cima, se faz necessário ter uma métrica como referência (um valor padrão), a fim de compará-la com as demais simulações. Para extrair tal métrica, executou-se a simulação utilizando o “ambiente inicial” (descrito no item 1 da especificação da estratégia), o qual não possui nenhum emprego de fator sobre os parâmetros do canal de comunicação.

O emprego dos fatores de correções nos parâmetros do canal de comunicação, tem o objetivo único de piorar a qualidade do canal em relação aos valores estabelecidos inicialmente. Após a simulação e extração da métrica de tempo, é possível avaliar a influência (em relação ao tempo contabilizado do “ambiente inicial”) do emprego de tal fator sobre o canal de comunicação, uma vez que, todos os demais parâmetros são mantidos constantes.

Para a extração do tempo de cada simulação, utilizou-se a função “MSG_get_clock()” provida pelo simulador SimGrid. Quando invocada a função, a mesma retorna em segundos todo o tempo demandado para a simulação, isto é, transferência, computação e recebimento de todas as tarefas definidas. Após o

retorno da função, converteu-se o resultado de segundos para horas, com oito casas de precisão numérica.

Foram escolhidos cinco fatores de correções para serem aplicados no canal de comunicação do “ambiente inicial”. Todos os fatores aplicados tinham como escala 20% (razão 0.2) de variação. O emprego dos fatores no canal gerou 15 simulações, sendo:

- Cinco simulações com a variação da largura de banda e latência constante;
- Cinco simulações com a variação da latência e largura de banda constante;
- Cinco simulações combinando simultaneamente os fatores utilizados individualmente.

A largura de banda é a capacidade de transmissão de um determinado meio e, refere-se a vazão de dados pelo canal (VALKENBURG, 1974). Com o objetivo de piorar a qualidade do canal, aplicou-se os fatores 0.8, 0.6, 0.4 e 0.2 sobre os parâmetros iniciais da largura de banda, reduzindo a capacidade de transmissão, e consequentemente estreitando a vazão de dados pelo canal.

O Quadro 7 demonstra alguns canais de comunicação. Sejam tais canais um exemplo de um “ambiente inicial”. A fim de exemplificação e facilitação do entendimento, serão utilizados valores inteiros e de fácil representação para os parâmetros.

```
<link id=”link1” bandwidth=”10MBps” latency=”100us”/>
<link id=”link2” bandwidth=”20MBps” latency=”110us”/>
<link id=”link3” bandwidth=”30MBps” latency=”120us”/>
<link id=”link4” bandwidth=”40MBps” latency=”130us”/>
```

Quadro 7 – Valores iniciais do canal de comunicação
Fonte: Autoria Própria.

Após o emprego do fator 0.8 nos parâmetros da largura de banda, é gerado um novo ambiente de simulação com um decréscimo de 20% dos valores iniciais. O Quadro 8 demonstra a variação dos parâmetros no novo ambiente de simulação, é possível também verificar que a latência permanece constante.

```
<link id=”link1” bandwidth=”8MBps” latency=”100us”/>
<link id=”link2” bandwidth=”16MBps” latency=”110us”/>
<link id=”link3” bandwidth=”24MBps” latency=”120us”/>
<link id=”link4” bandwidth=”32MBps” latency=”130us”/>
```

Quadro 8 – Emprego do fator 0.8 sobre a largura de banda
Fonte: Autoria Própria.

A latência de uma rede de comunicação de dados pode ser mensurada através do tempo demandado pelo envio de um pacote da fonte ao destino (VALKENBURG, 1974). Para alterar os parâmetros da latência nas mesmas escalas de 20% utilizadas para a largura de banda e, piorar o canal em relação aos valores inicias, é preciso aplicar os fatores 1.2, 1.4, 1.6 e 1.8 para incrementar os parâmetros e consequentemente, demandar mais tempo para o transporte dos dados.

Considerando novamente o Quadro 7 como um exemplo de um “ambiente inicial”, o Quadro 9 representa o novo ambiente de simulação gerado após a aplicação do fator 1.2 sobre os parâmetros da latência. O emprego de tal fator, gera um incremento de 20% sobre o valor inicial do parâmetro. É possível também verificar que a largura de banda permanece constante.

```
<link id=”link1” bandwidth=”10MBps” latency=”120us”/>
<link id=”link2” bandwidth=”20MBps” latency=”132us”/>
<link id=”link3” bandwidth=”30MBps” latency=”144us”/>
<link id=”link4” bandwidth=”40MBps” latency=”156us”/>
```

Quadro 9 – Emprego da fator 1.2 sobre a latência
Fonte: Autoria Própria.

E por fim, o Quadro 10 demonstra a combinação do fator de 20% para a largura de banda e latência (considerando os valores do Quadro 7 como um “ambiente inicial”), o qual gerou uma nova simulação. Vale ressaltar que, de igual modo, foi realizado simulações com as combinações dos fatores 40, 60 e 80%.

```
<link id=”link1” bandwidth=”8MBps” latency=”120us”/>
<link id=”link2” bandwidth=”16MBps” latency=”132us”/>
<link id=”link3” bandwidth=”24MBps” latency=”144us”/>
<link id=”link4” bandwidth=”32MBps” latency=”156us”/>
```

Quadro 10 – Emprego dos fatores de 20% para latência e largura de banda
Fonte: Autoria Própria.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos das simulações realizadas com o simulador SimGrid. Os resultados permitem validar o estudo proposto por este trabalho sobre o canal de comunicação, além disso, também é possível verificar se o desempenho do *grid* é afetado com os diferentes modelos de simulações.

Todos os resultados que serão apresentados, foram extraídos dos modelos de simulações demonstrados e especificados no capítulo anterior. Todos os modelos foram criados seguindo rigorosamente a estratégia descrita para a criação das simulações.

Através da análise dos resultados das simulações e, a sua representatividade no modelo, será possível mensurar o grau de importância da infraestrutura no desempenho final do *grid*. Tal conhecimento pode aprimorar as regras do *grid* referente à distribuição de tarefas, e os balanços de cargas entre os dispositivos do modelo.

4.1 RESULTADOS OBTIDOS

Após a realização das simulações, é extraído a quantidade de horas demandada para a computação das tarefas. A Figura 8 apresenta os resultados obtidos com a variação da largura de banda. O gráfico ilustra a quantidade de horas, representado pelo eixo cartesiano Y (“Horas”), em relação a largura de banda utilizada, representado pelo eixo cartesiano X (“Largura de Banda”). Cada ponto do gráfico representa uma simulação realizada.

Os pontos plotados com valores em “X” de 80, 60, 40 e 20%, representam o respectivo fator de correção aplicado nos parâmetros da largura de banda. Por exemplo, no ponto 20%, todos parâmetros da largura de banda, possuem 20% dos valores estabelecidos no “ambiente inicial”. É importante ressaltar que, o fator é aplicado em todos os canais de comunicação definidos no modelo, como descrito em 3.2.5.

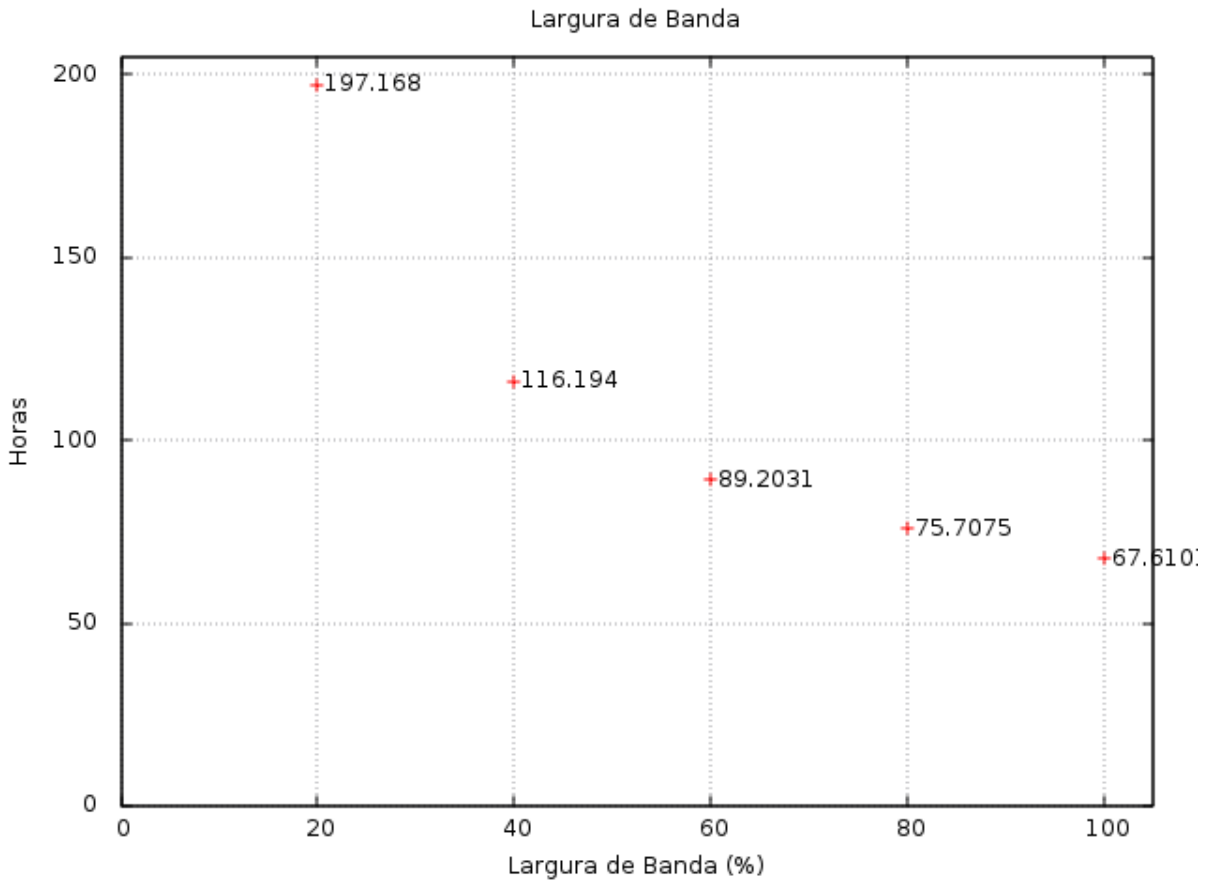

Figura 8 – Resultado 1 das simulações com a variação da largura de banda
Fonte: Autoria própria

Para um melhor entendimento, a Figura 9 apresenta em gráfico de barras os mesmo resultados das simulações demonstradas na Figura 8.

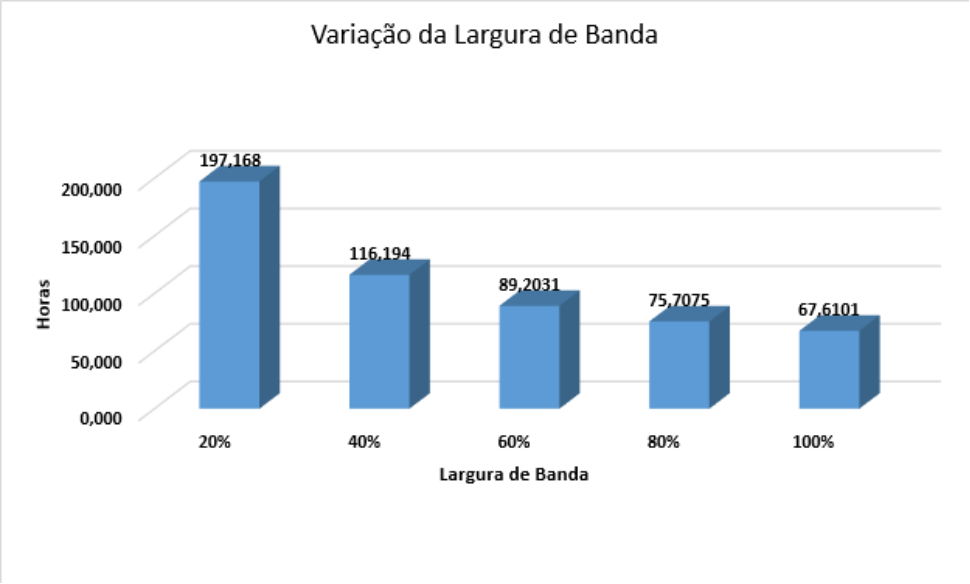

Figura 9 – Resultado 2 das simulações com a variação da largura de banda
Fonte: Autoria própria

De igual modo, a Figura 10 apresenta os resultados obtidos com a variação da latência. O gráfico ilustra a quantidade de horas, representado pelo eixo cartesiano Y (“Horas”), em relação latência utilizada, representado pelo eixo cartesiano X (“Latência”). Cada ponto do gráfico representa uma simulação realizada.

Os pontos plotados com valores em “X” de 120, 140, 160 e 180%, representam o respectivo fator de correção aplicado nos parâmetros da latência. Por exemplo, no ponto 120%, todos parâmetros da latência, possuem um incremento de 20% dos valores estabelecidos no “ambiente inicial”. É importante ressaltar que, o fator é aplicado em todos os canais de comunicação definidos no modelo, como descrito em 3.2.5.

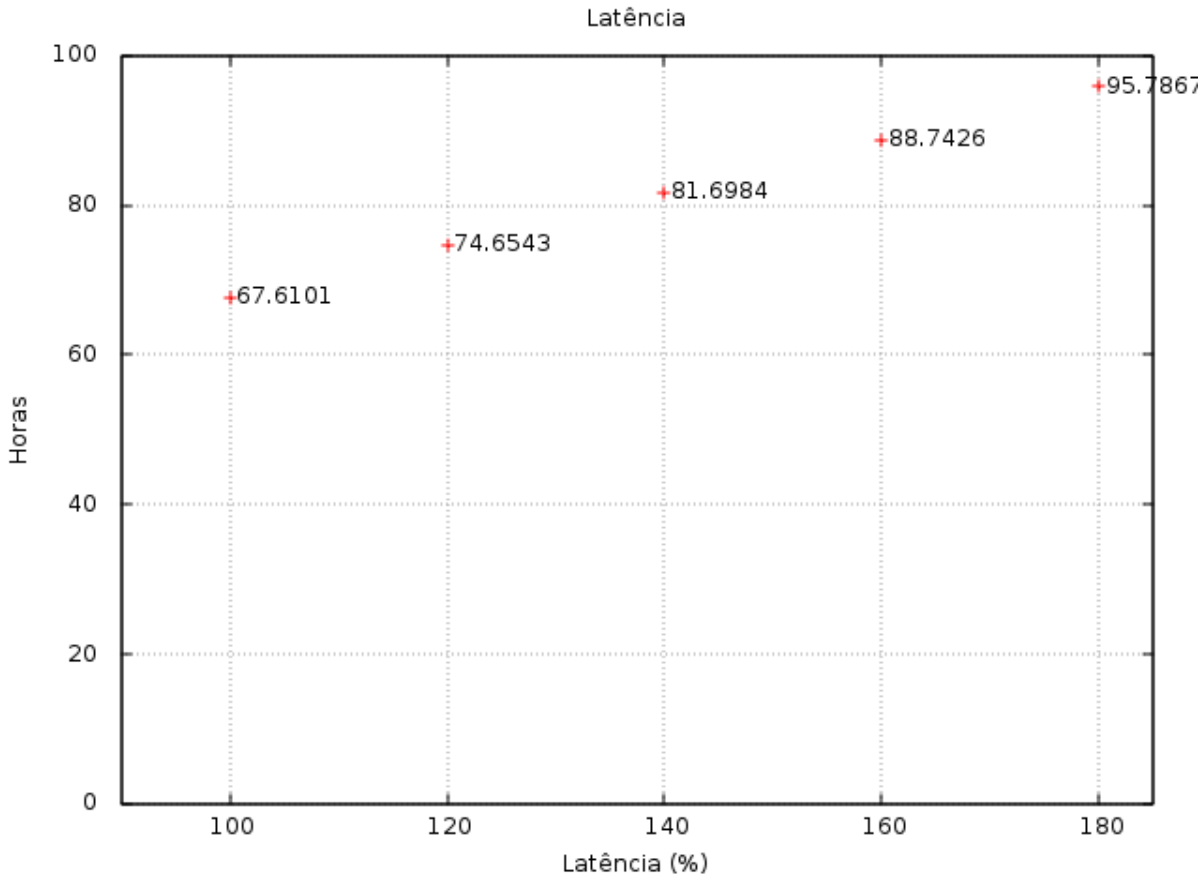

Figura 10 – Resultado 1 das simulações com a variação da latência
Fonte: Autoria própria

Para um melhor entendimento, a Figura 11 apresenta em gráfico de barras os mesmo resultados das simulações demonstradas na Figura 10.

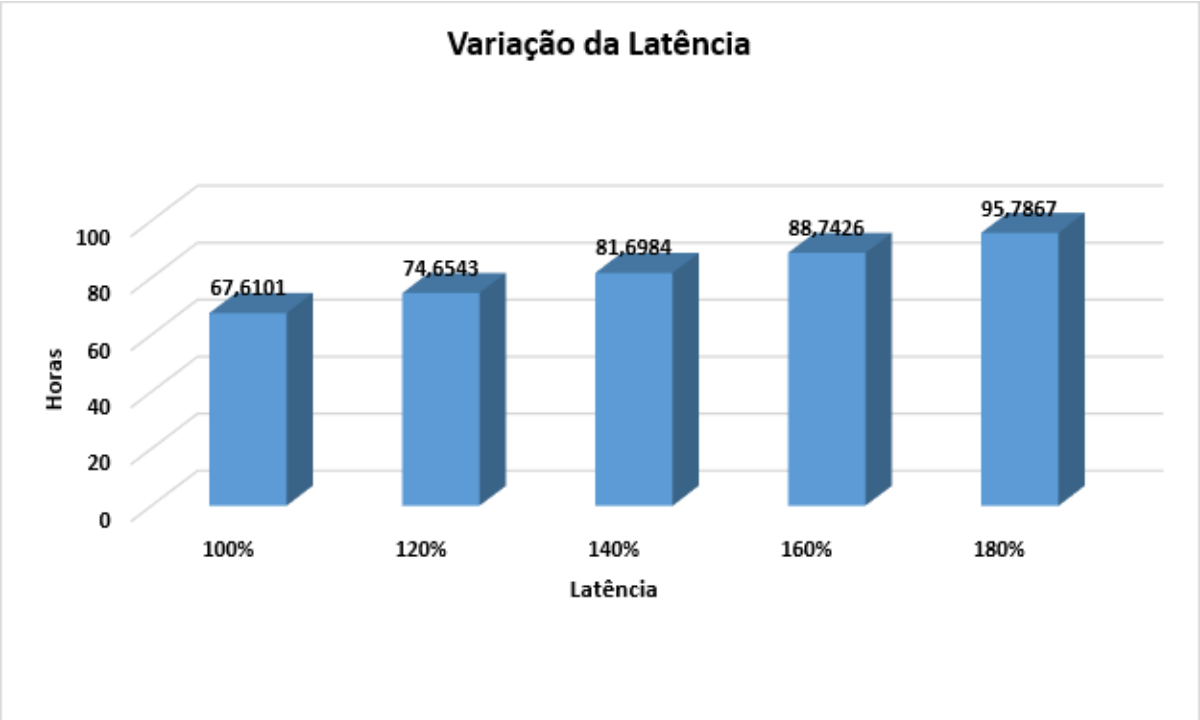

Figura 11 – Resultado 2 das simulações com a variação da largura de banda
Fonte: Autoria própria

A Figura 12 apresenta os resultados obtidos com a variação da largura de banda e latência simultaneamente. O gráfico ilustra a quantidade de horas, representado pelo eixo cartesiano Y (“Horas”), em relação a largura de banda e latência utilizada, representados respectivamente pelos eixos cartesianos X (“Largura de banda”) e Z (“Latência”). Cada ponto do gráfico representa uma simulação realizada.

Os pontos plotados, representam a aplicação dos respectivos fatores de correções nos parâmetros da largura de banda e latência, em relação aos valores dos parâmetros estabelecidos no “ambiente inicial”. É importante ressaltar que, a linha que interliga os pontos do gráfico é para a facilitação da leitura do mesmo.

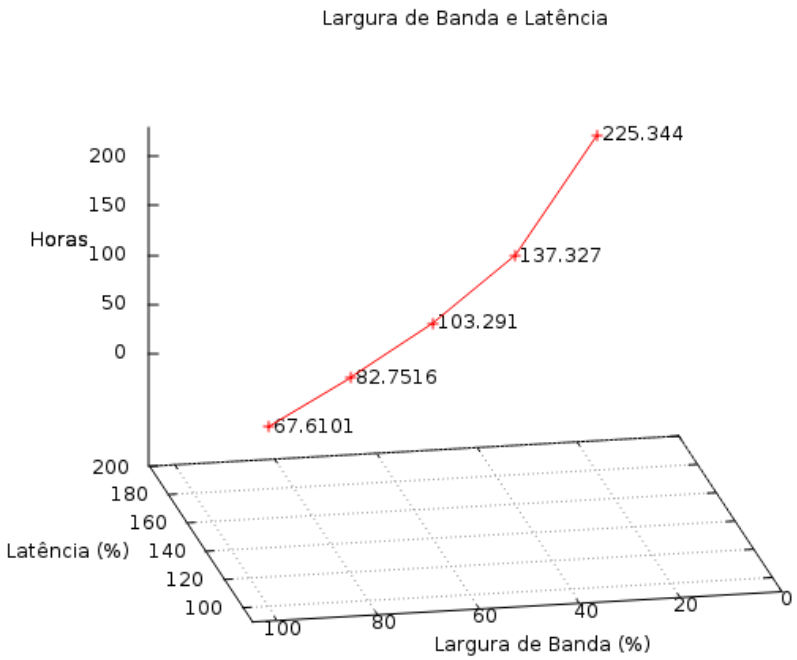

Figura 12 – Resultado 1 das simulações com a variação da largura de banda e latência
Fonte: Autoria própria

Para um melhor entendimento, a Figura 13 apresenta em gráfico de barras os mesmo resultados das simulações demonstradas na Figura 12.

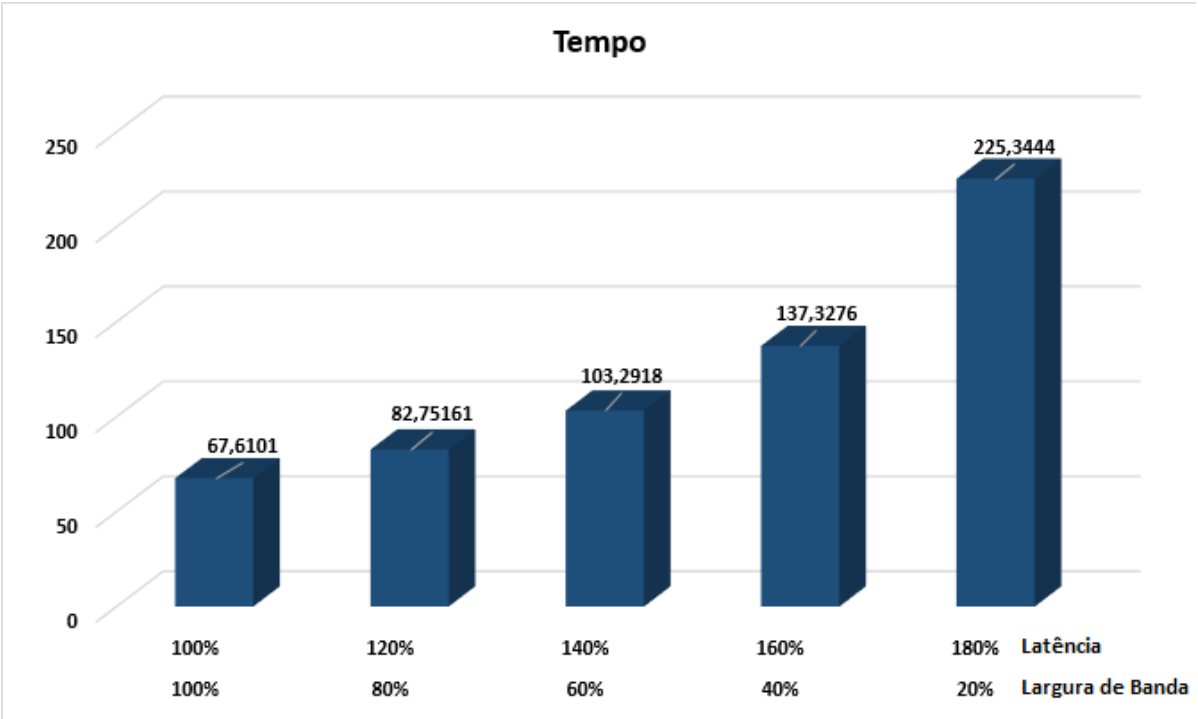

Figura 13 – Resultado 2 das simulações com a variação da largura de banda
Fonte: Autoria própria

4.2 ANÁLISE ESTATÍSTICA

Para a análise estatística dos resultados das simulações, escolheu-se o planejamento fatorial 2^k, com k=2. O planejamento fatorial é indicado para a fase inicial do procedimento experimental, quando há necessidade de se definir os fatores mais importantes e estudar os efeitos sobre a variável resposta escolhida. Além disso, é um modelo de efeitos fixos, isto é, a análise dos efeitos provocados pelos fatores não pode ser transferida para outros níveis que não os analisados no planejamento (BUTTON, 2001).

O processo experimental, consiste em realizar teste com cada uma das combinações da matriz experimental, para em seguida, determinar e interpretar os efeitos principais de interação dos fatores investigados e, assim, poder identificar as melhores condições experimentais do processo de fabricação (BUTTON, 2001). A matriz experimental das simulações está representada na Figura 14.

	Design: 2**(2-0) design		
Standard Run	Latência	Largura de Banda	Tempo
1	100.0000	20.0000	197.1680
2	180.0000	20.0000	225.3440
3	100.0000	100.0000	67.6101
4	180.0000	100.0000	95.7867

Figura 14 – Matrix experimental
Fonte: Autoria própria

A matriz experimental, tornou possível a construção de diversos gráficos de análise de dados, os quais serão apresentados a seguir. A partir da análise dos gráficos construídos, é possível analisar quem mais influência no desempenho do *grid*, se a latência ou largura de banda.

A Figura 15 ilustra as médias marginais da largura de banda associada à grandeza de tempo, a qual é estuda neste trabalho. Os gráficos das médias marginais, possuem junto com os outros gráficos a serem mostrados, o objetivo de ilustrar qual variável da matriz experimental (largura de banda ou latência), que mais afeta a “variável objetivo” (tempo).

A Figura 15 ilustra a média marginal da largura de banda distribuída, linearmente. É possível constatar que os fatores aplicados na largura de banda

impactaram no desempenho do *grid*. Além disso, fica evidente que quanto menor o fator de correção, maior é o impacto gerado.

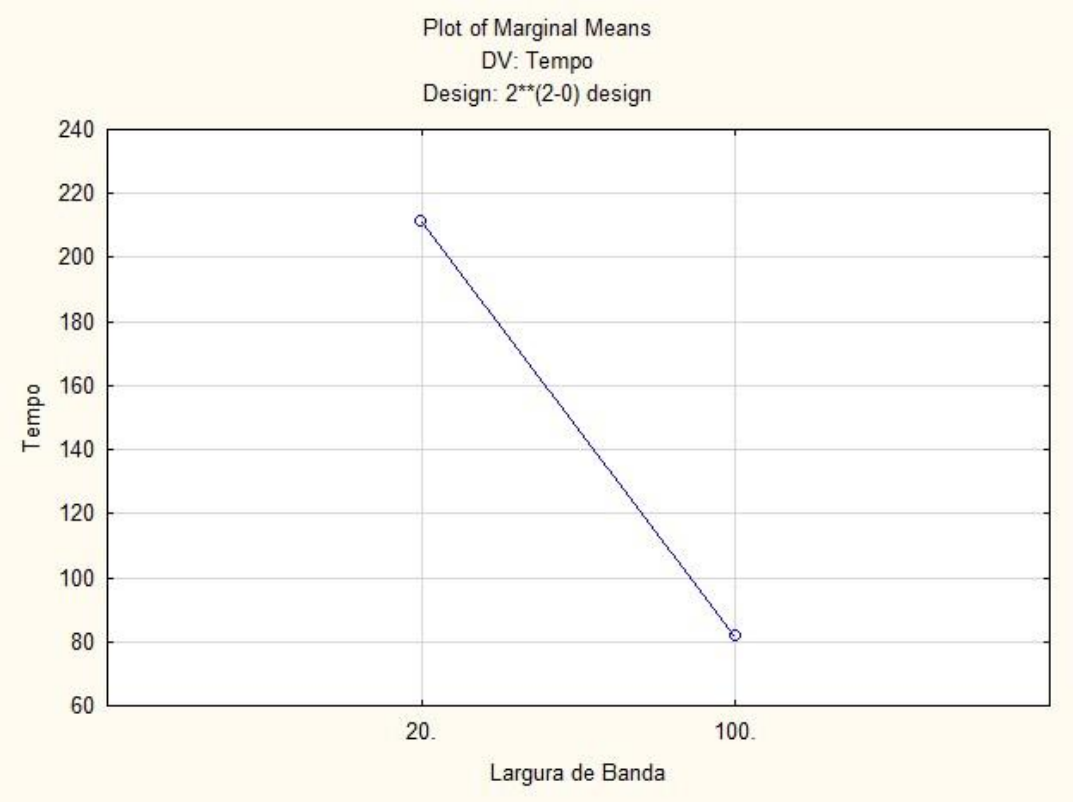

Figura 15 – Média marginal simples da Largura de Banda
Fonte: Autoria própria

A Figura 16 ilustra a média marginal da largura de banda pela latência, o gráfico demonstra as médias distribuídas linearmente em relação à latência. É possível verificar pela distância das retas, quais foram os fatores que mais impactaram no desempenho do *grid.*

Se compararmos a Figura 16 com a Figura 18, fica evidente que a largura de banda causou um impacto maior que a latência no desempenho do *grid*, uma vez que, as retas plotadas na Figura 16 se distanciam substancialmente mais do que as retas da Figura 18.

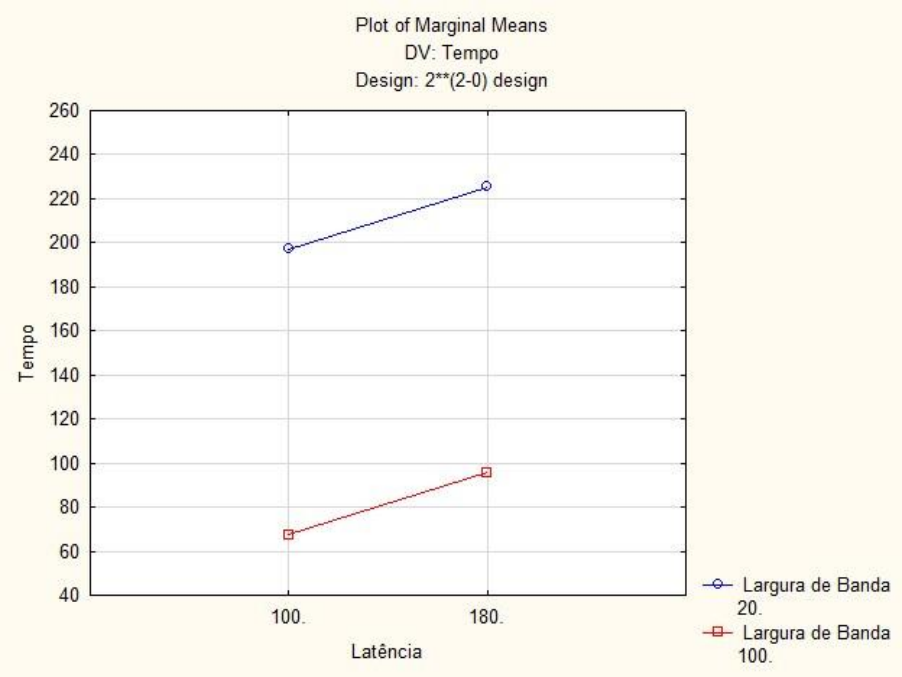

Figura 16 – Média marginal composta para Largura de Banda
Fonte: Autoria própria

A Figura 17 ilustra a média marginal da latência, distribuída linearmente. É possível constatar que os fatores aplicados na latência impactaram no desempenho do *grid*. Além disso fica evidente que quanto maior o fator de correção, maior é o impacto gerado.

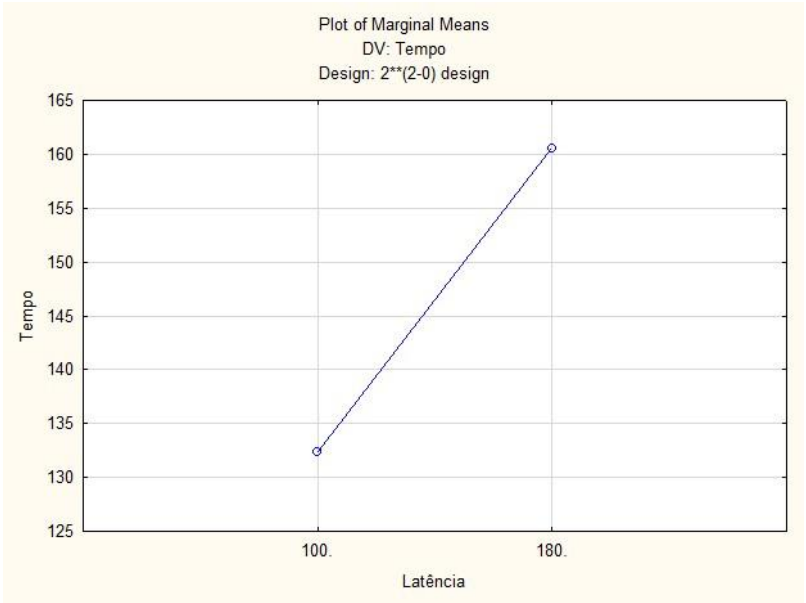

Figura 17 – Média marginal simples da Latência
Fonte: Autoria própria

A Figura 18 ilustra a média marginal da latência pela largura de banda, o gráfico demonstra as médias distribuídas linearmente em relação à largura de banda. É possível verificar pela distância das retas, quais foram os fatores que mais impactaram no desempenho do *grid.*

Como já descrito anteriormente, é possível verificar pela pequena distância entre as retas do gráfico da Figura 18, que a variação da latência não gerou tanto impacto no desempenho do *grid* quanto a largura de banda.

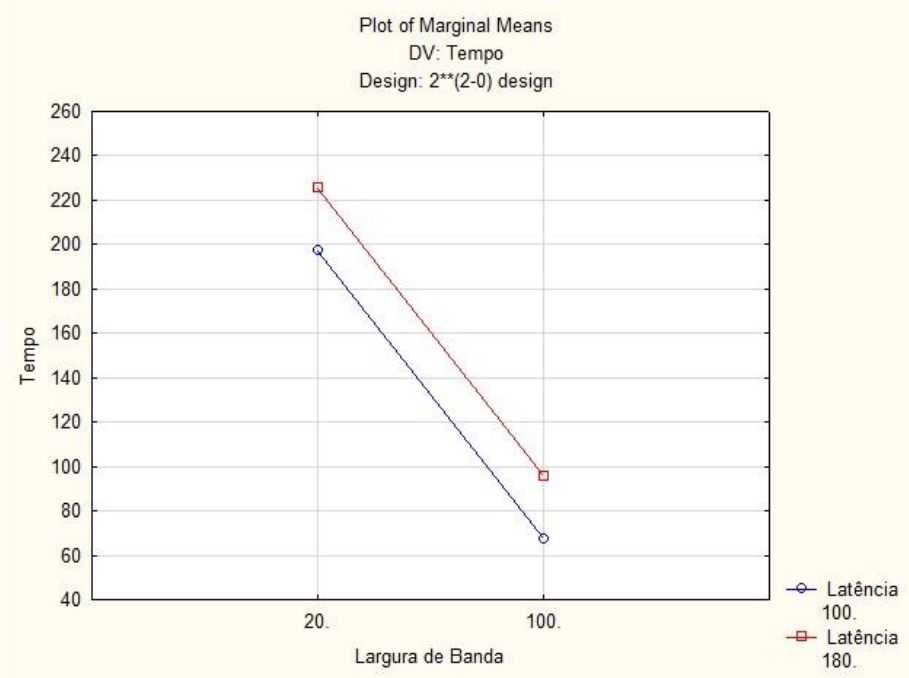

Figura 18 – Média marginal composta para Latência
Fonte: Autoria própria

A Figura 19 demonstra o gráfico de Pareto, o qual possui o principal objetivo compreender a relação ação/benefício. A ordenação das frequências de ocorrências, permite a localização de problemas vitais e a eliminação de perdas, assim é priorizada a ação que trará o melhor benefício.

O número negativo tem o objetivo de ilustrar o efeito causado pela largura de banda. A representação da escala negativa, significa que, quanto maior a largura de banda, menor será o tempo de execução, e consequentemente melhor será o desempenho do *grid*.

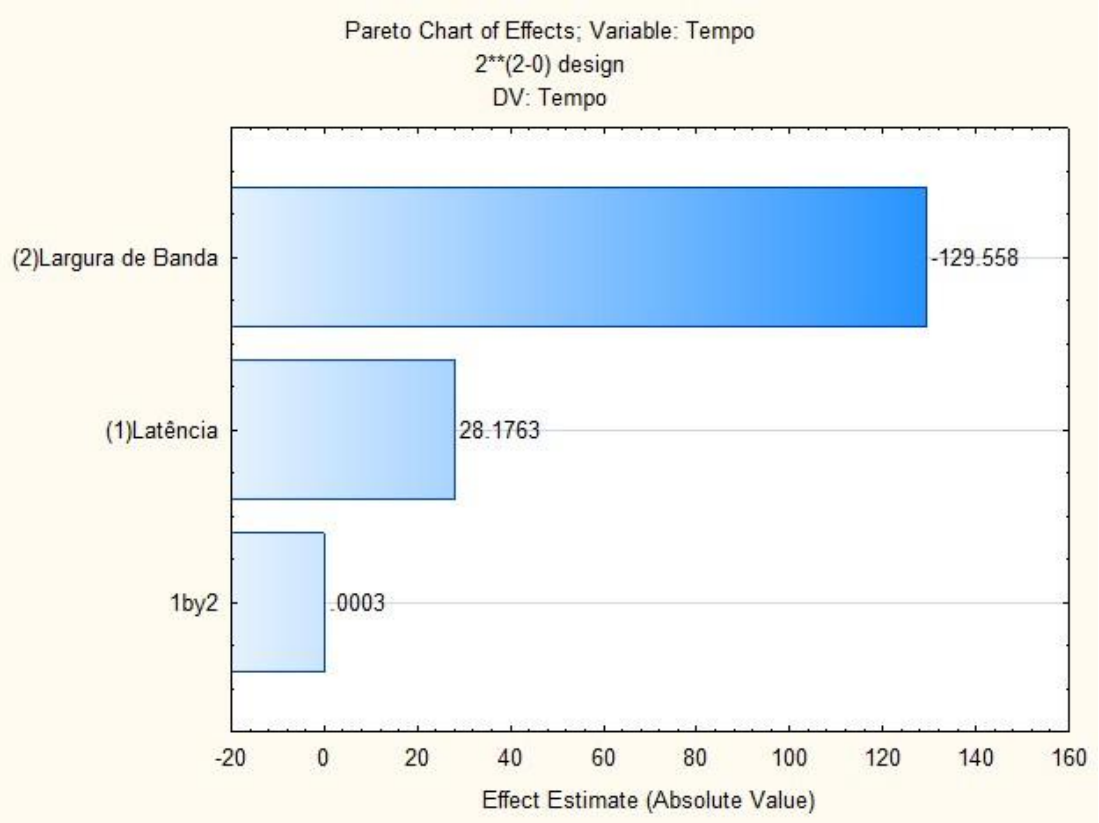

Figura 19 – Gráfico de Pareto
Fonte: Autoria própria

A Figura 20 representa a variável tempo em uma superfície tridimensional. O gráfico de superfície conecta os pontos das variáveis participantes (largura de banda e latência), formando uma espécie de mapa topográfico. É possível ver através do gráfico, o impacto gerado no tempo por todas as possíveis combinações de largura de banda e latência.

A área do gráfico representado pela cor verde escura, é a combinação da largura de banda e latência que obtiveram o melhor desempenho, ou seja, são os fatores de correções que geraram o menor tempo de execução.

O gráfico ainda permite ver o deslocamento da superfície (tempo) em relação aos fatores aplicados. Se deslocarmos no eixo da latência, a variação da superfície (tempo) será diferente se deslocarmos no eixo da largura de banda, isso se deve ao fato da largura de banda e latência impactarem diferentemente no tempo, como já visto nos gráficos anteriores.

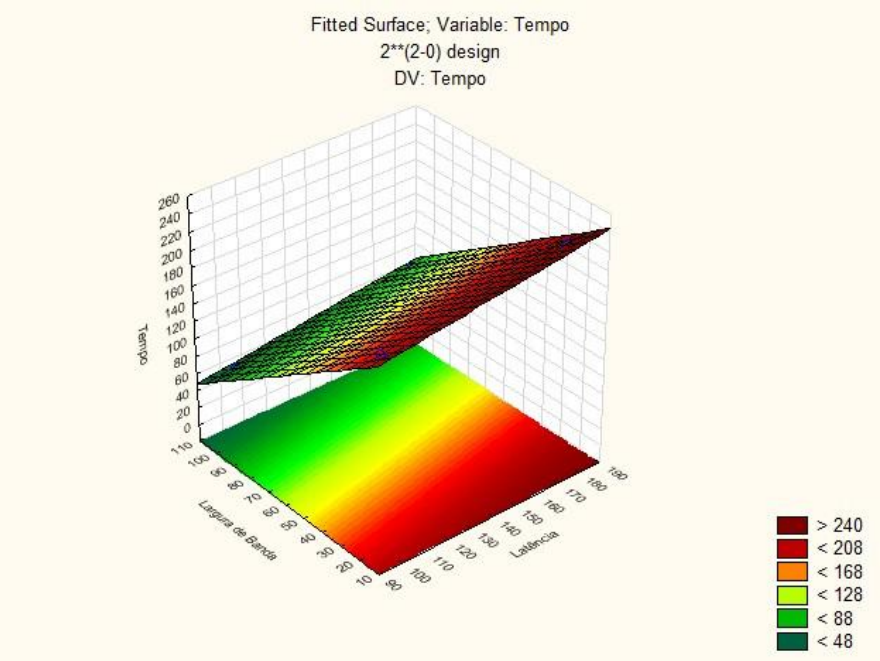

Figura 20 – Gráfico de superfície tridimensional
Fonte: Autoria própria

A Figura 21 tem o mesmo objetivo do gráfico apresentado na Figura 20, porém, é apresentado em uma superfície plana. Através da análise do gráfico na superfície plana, é possível verificar o impacto dos fatores de correções empregados nos parâmetros da largura de banda e latência.

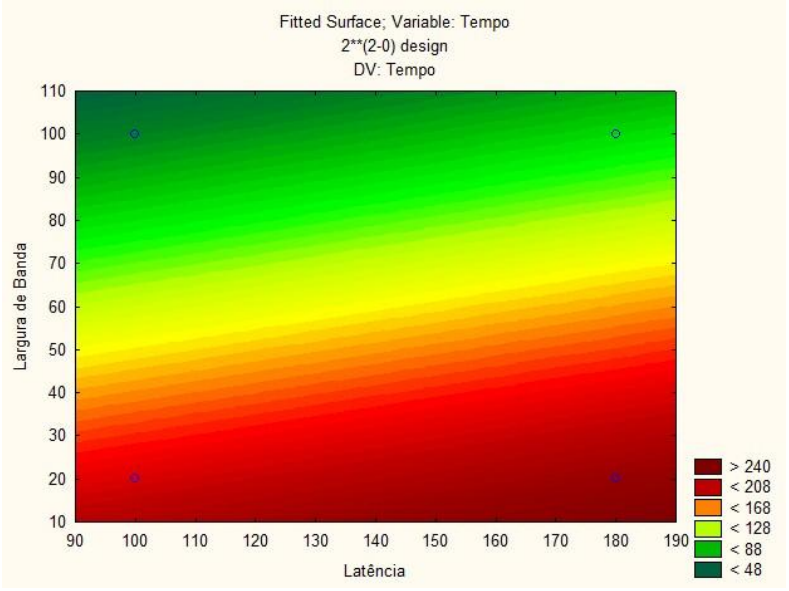

Figura 21 – Gráfico de superfície plana
Fonte: Autoria própria

Após a análise de superfície demonstrados nos gráficos anteriores, é possível estabelecer uma relação matemática da largura de banda, latência e tempo.

É possível saber a variação do tempo em relação ao aumento da largura de banda, ou a redução da latência.

A função matemática que expressa tal variação, está representada no Quadro 11. Fazendo o uso da função, é possível atribuir valores para y, que representa a largura de banda, ou para x, que representa a latência, a fim de estudar as variações causadas no tempo.

```
Z = 194.337665 + 0.352198125*x – 1.619483125*y
0.00000093749999986*x*y
```

Quadro 11 – Relação matemática para tempo, largura de banda e latência
Fonte: Autoria Própria.

4.3 ANÁLISE DE DESEMPENHO

Além das análises estatísticas demonstradas anteriormente, realizou-se a extração da função matemática que melhor se ajusta aos pontos obtidos como resultados das simulações (demonstrados na Figura 8 e Figura 10). A partir da função matemática, é possível saber o impacto que a largura de banda e latência podem causar no desempenho do *grid*.

Após a análise dos resultados obtidos pelas simulações com variações na latência, já era possível visualizar que a melhor modelagem matemática seria linear. Realizou-se a extração da função matemática que melhor se aproxima dos pontos, e como já esperado, a função matemática obtida e representado no Quadro 12, é linear.

```
0.352207*x + 32.3894
```

Quadro 12 – Modelagem matemática do impacto da latência
Fonte: Autoria Própria.

É possível verificar na Figura 22, que a função matemática modelada, se ajusta perfeitamente nos pontos obtidos das simulações.

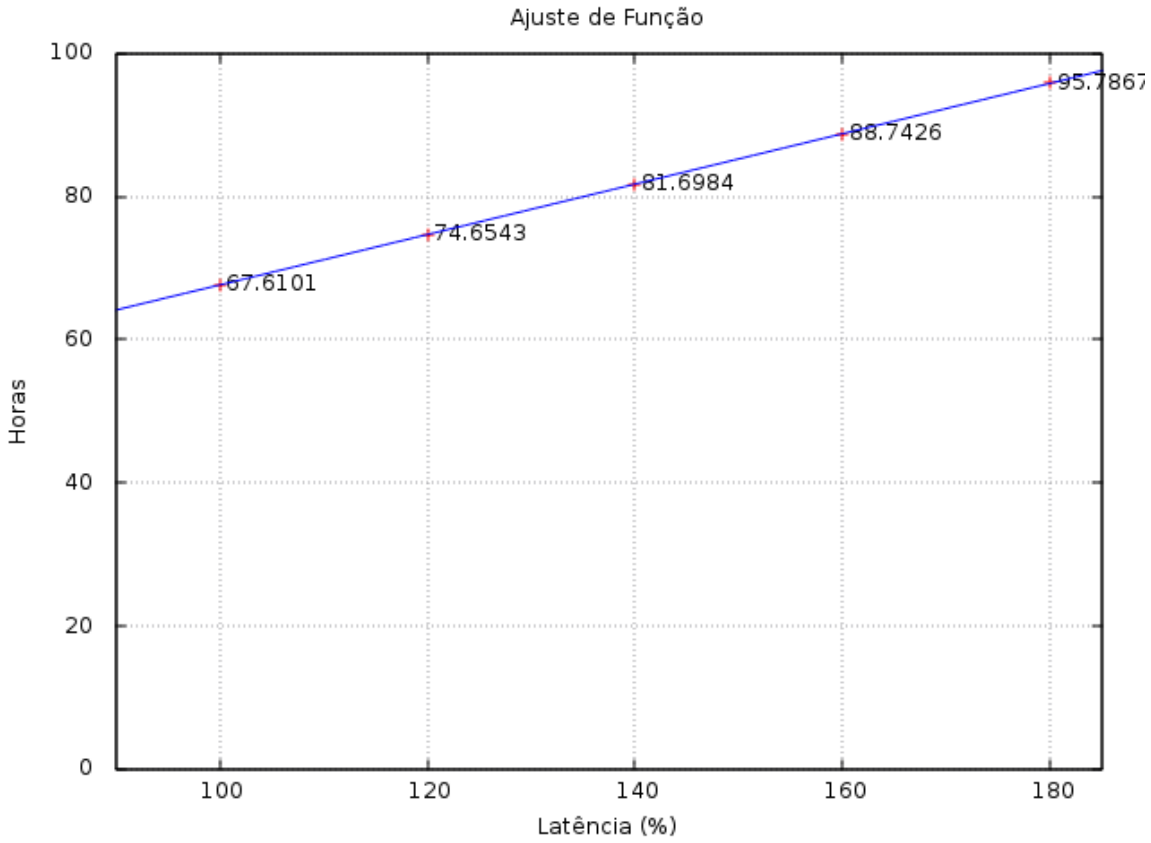

Figura 22 – Ajuste de função da latência
Fonte: Autoria própria

Após a análise dos resultado obtidos pelas simulações com variações na largura de banda, era possível imaginar que a função de ajuste seria exponencial. Após a modelagem matemática para funções exponenciais, verificou-se que a função modelada, demonstrada na Figura 23 e, representada no Quadro 13, não se ajustava aos pontos do gráfico.

$254.551e^{-0.0159109*x}$

Quadro 13 – Modelagem matemática para função exponencial (largura de banda)
Fonte: Autoria Própria.

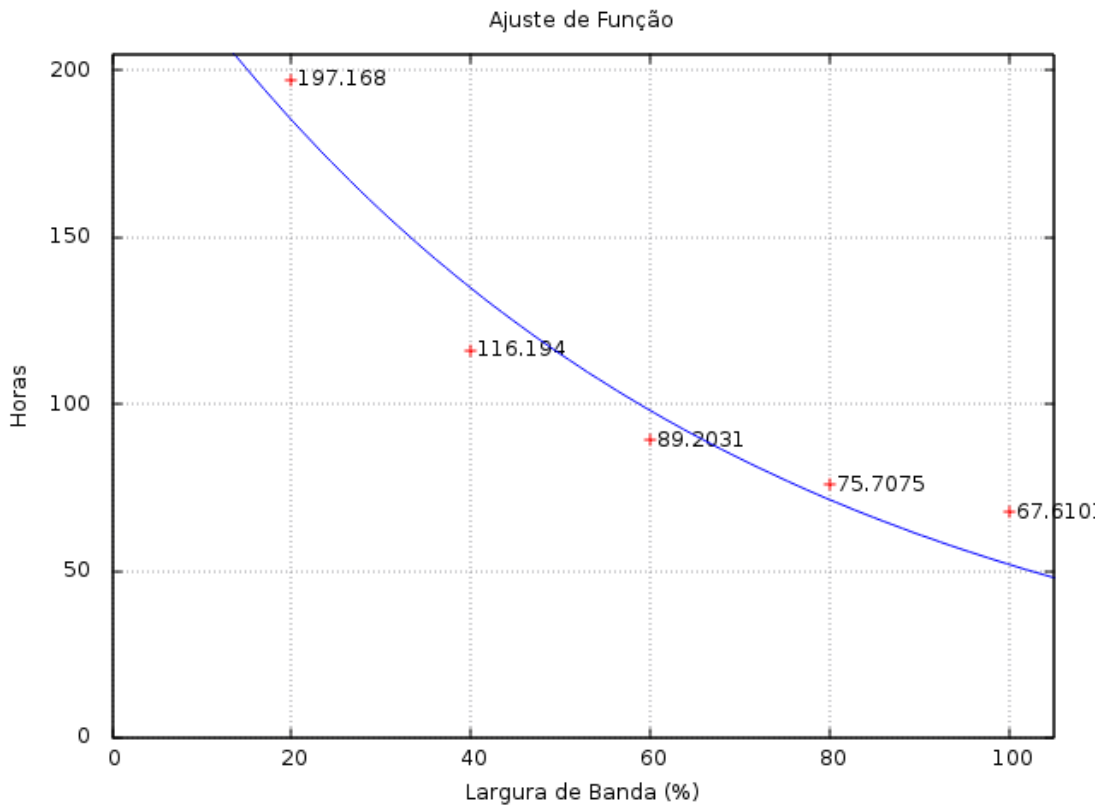

Figura 23 – Ajuste de função exponencial para a largura de banda
Fonte: Autoria própria

Realizou-se novamente a modelagem dos pontos obtidos pelas simulações da largura de banda, porém, a modelagem foi realizada para funções polinomiais. Após a modelagem, foi possível verificar que a função polinomial obtida, demonstrada no Quadro 14, se ajusta perfeitamente aos pontos do gráfico, como demonstrado na Figura 24.

$$8.43508*10^{-6}*x^4 - 0.00253051*x^3 + 0.286789*x^2 - 15.1828*x + 405.004$$

Quadro 14 – Modelagem matemática para função polinomial (largura de banda)
Fonte: Autoria Própria.

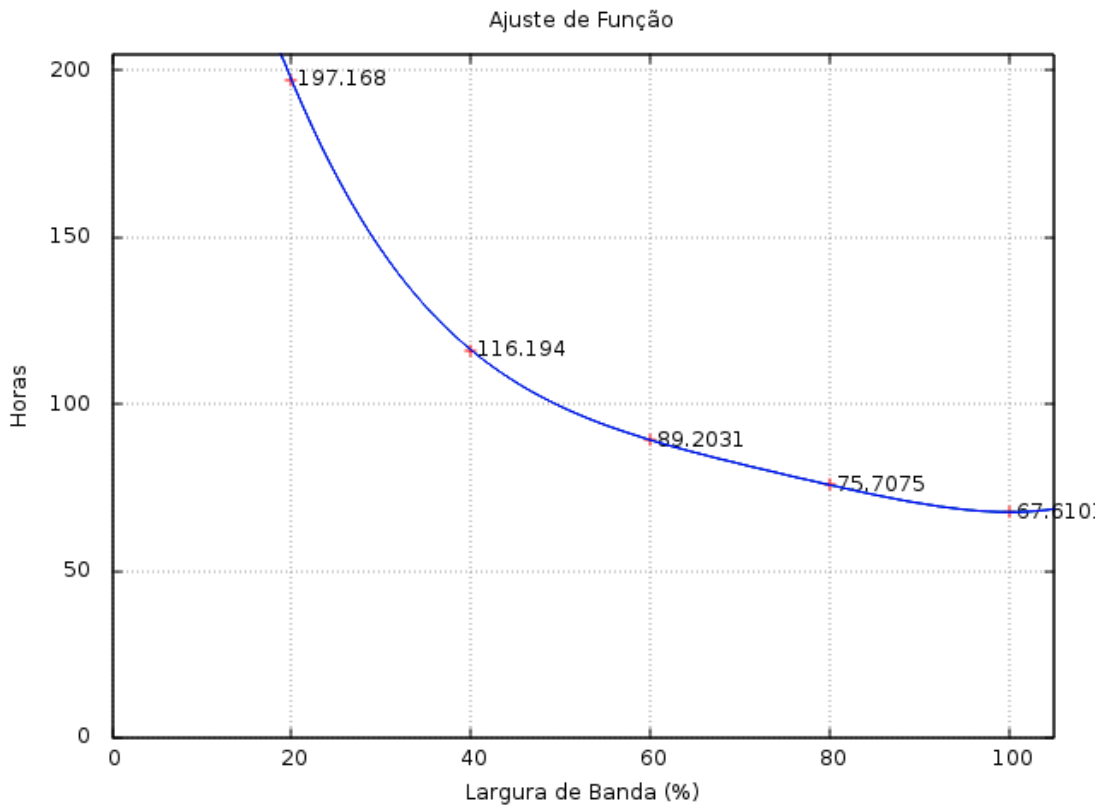

Figura 24 – Ajuste de função polinomial para a largura de banda
Fonte: Autoria própria

A extração do modelo matemático referente ao impacto causado pela largura de banda e latência, veio a confirmar que, a largura de banda tem uma maior influência do que a latência no desempenho final do *grid*. Além disso, é possível também saber o quão maior é o impacto causado, uma vez que, são conhecidas as funções modelada do impacto gerado no modelo.

5 CONCLUSÃO

Neste trabalho abordou-se fundamentos de simulação, processamento e comunicação de dados. Foram conceituados ambientes de processamento como *cluster* e *grid*, os quais ganham notoriedade devido ao baixo custo de implantação e, por serem bastante modulares.

Na abordagem de simulação de *grid*, apresentou-se temas fundamentais para sua compreensão. Os desafios para simular um ambiente com características elevadas de heterogeneidade e, a alta necessidade da simulação para estudos e pesquisas do modelo.

Para atingir os objetivos proposto deste trabalho, realizou-se simulações com o auxílio do simulador SimGrid. Através das variações no canal de comunicação, especificamente largura de banda e latência, foram criados modelos de simulações para serem estudados.

Estabeleceu-se uma estratégia para a criação dos modelos de simulações, com o objetivo de representar os mais variados dispositivos que possam fazer parte de um *grid computing*. Todos os critérios estabelecidos na estratégia foram obedecidos rigorosamente.

A partir dos resultados das simulações, foi possível aplicar métodos estatístico e extrair os significados das variáveis obtidas. Os métodos estatísticos auxiliaram a identificar e visualizar o impacto sofrido pelo *grid* após as variações do canal de comunicação, foi possível também verificar que a largura de banda impacta mais que a latência no desempenho do *grid*.

Após a análise estatística, realizou-se a modelagem matemática do impacto causado no *grid* com as variações da largura de banda e latência. A análise de complexidade demonstrou matematicamente o quanto que a largura de banda impacta a mais do que a latência.

A pesquisa realizada neste trabalho, pode contribuir para o aprimoramento nas regras/políticas de um *grid computing*. O modo de alocar o canal, ou distribuir as tarefas para os escravos, podem sofrer alterações sabendo da relevância que o canal tem no desempenho do *grid*. O aprimoramento de tais regras poderiam evitar alguns tipos de *workload imbalance*.

Além de poder contribuir para o aprimoramento das regras, o trabalho também pode direcionar em parte novas implantações de *grid computing*. A exemplo

do aplicativo *Power Sleep,* que é direcionado exclusivamente aos dispositivos móveis. Os novo aplicativos desse segmento, poderiam destacar a importância da infraestrutura no desempenho do *grid*, e pedir aos seus usuários que permaneçam ao máximo em uma rede de alta velocidade.

O canal de comunicação é alvo de estudo da computação há vários anos, é possível constatar de tempos em tempos, novas tecnologias e protocolos que aprimoram o modo de se comunicar. Assim, a influência do canal de comunicação no desempenho do *grid,* revelado por este trabalho, pode contribuir de diversas formas no modo que um *grid* se opera.

5.1 TRABALHOS FUTUROS

Como proposta de trabalhos futuros, os seguintes temas se destacaram durante a pesquisa realizada, são eles:

1. Estudar as possíveis mudanças nas regras de alocação de canal e distribuição de tarefas no *grid computing*.
2. Quando se trata de um *grid* em escala global, é impossível exigir que os usuários façam parte de infraestruturas de qualidade. Assim, poderia reduzir a influência do canal de comunicação no desempenho do *grid.*
3. Criar novos protocolos de comunicação para os diferentes dispositivos que podem compor um *grid*.

REFERÊNCIAS

ANDERSON, David P. *et al*. **SETI@home An Experiment in Public-Resource Computing**. Communications of the ACM, New York, p. 56-61, 2002.

BUTTON, S.T.. **Metodologia para planejamento experimental e análise de resultado**. Universidade Estadual de Campinas.São Paulo, 2001.

BUYYA, Rajkumar; VENUGOPAL, Srikumar. **A Gentle Introduction to Grid Computing and Technologies**. Computer Society of India, p. 1-11, 2005.

BUYYA, Rajkumar. **Economic-based Distributed Resource Management and Scheduling for Grid Computing**.3. ed. Melbourne. Monash University. 2002.

BUYYA, Rajkumar; MURSHED, Manzur. **GridSim: A Grid Simulation Toolkit for Resource Modelling and Application Scheduling for Parallel and Distributed Computing**. Cornell University Library, New York, p. 11-15, mar. 2002.

CASANOVA, Henri. **Simgrid: A Toolkit for the Simulation of Application Scheduling**. Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on, Brisbane, p. 30-37, 2001.

CASANOVA, Henri *et al*. **SimGrid: a Generic Framework for Large-Scale Distributed Experiments**. Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on, Cambridge, p. 126-131, abr. 2008.

CASANOVA, Henri *et al*. **SimGrid: a Sustained Effort for the Versatile Simulation of Large Scale Distributed Systems**. Cornell University Library, New York, p. 1-4, set. 2013.

CONSEIL EUROPÉEN POUR LA RECHERCHE NUCLÉAIRE (2014). CERN. Fonte: home.web.cern. Disponível em: <http://home.web.cern.ch/about>. Acesso em: 14 jul. 2014.

COULOURIS, George *et al*. **Distributed Systems Concepts and Design**. 5. ed. Boston. Pearson Education, 2012.

DEBIAN (2014). Debian Project. Fonte: Debian. Disponível em: <https://www.debian.org/>. Acesso em: 15 set. 2014.

DELL (2012). DELL Project. Fonte: dell. Disponível em: <http://en.community.dell.com/techcenter/high-performance-computing/w/wiki/2329>. Acesso em: 11 out. 2014.

DEPOORTER, Win *et al*. **Scalability of Grid Simulators: An Evaluation**. 1. ed. Las Palmas de Gran Canaria. Springer Berlin Heidelberg, 2008.

DONGARRA, Jack; LASTOVETSKY, Alexey. **An Overview of Heterogeneous High Performance and Grid Computing**. 1. ed. Boston. Science Publishers, Inc, 2006.

DZERO (2014). The D0 Experiment. Fonte: d0. Disponível em: <http://www-d0.fnal.gov/>. Acesso em: 14 jul. 2014.

FARIA, Sérgio. **Grid Simulators**. FC U. Porto, p. 1-4, ago. 2012.

FOSTER, I. *et al*. **Grid services for distributed system integration**. IEEE Computer Society, p. 37-46, jun. 2002.

FOSTER, I. **What is the Grid? A three Point Checklist**. Argonne National Laboratory, Chicago, p. 1-4, jul. 2002.

FOSTER, I; KESSELMAN, C.. **The Grid 2:** blueprint for a new computing infrastructure. San Francisco: Morgan Kaufmann Publishers, 2004.

GOLDCHLEGER, A.. **InteGrade: um sistema de middleware para computação em grade oportunista**. Instituto de Matemática e Estatística da Universidade de São Paulo. São Paulo, 2004.

IOSUP, A; EPEMA, D.. **Grid Computing Workloads**. IEEE Internet Computing, p. 19-26, mar. 2011.

LEGRAND, Anaud (2006). **Models, Simulation, Emulation, Experimention for Grid Computing**. Mescal. Fonte: imag. Disponível em:

<http://mescal.imag.fr/membres/arnaud.legrand/articles/slides_g5k_simul.pdf>. Acesso em 14 jul. 2014.

LIU, Gang; SCHMIDER, Hartmut L.. **The Double-Layer Master-Slave Model: A Hybrid Approach to Parallel Programming for Multicore Cluster**. High Performance Computing Virtual Laboratory, p. 3-8, dec. 2011.

MOORE, Gordon E. (1965). **"Moore's Law" Predicts the Future of Integrated Circuits**. Computer History. Fonte: computerhistory. Disponível em: <http://www.computerhistory.org/semiconductor/timeline/1965-Moore.html>. Acesso em: 08 set. 2014.

MOORE, Gordon E., **Cramming More Components onto Integrated Circuits**. Electronics, pp. 114–117, April 19, 1965.

OPEN GRID FORUM (2014). Open Grid Forum. Fonte: An Open Global Forum for Advanced Distributed Computing. Disponível em: <http://www.ogf.org/dokuwiki/doku.php>. Acesso em: 01 set. 2014.

SABHARWAL, C. L.. **Java, Java, Java**. IEEE Potentials, p. 33-37, set. 1998.

SAMSUNG POWER SLEEP (2014). Samsung Project. Fonte: Samsung. Disponível em: <http://www.samsung.com/at/microsite/powersleep/app.html>. Acesso em: 01 set. 2014.

SÃO PUALO RESEARCH AND ANALYZES CENTER (2014). Sprace. Fonte: SPRACE Computing Center. Disponível em: <http://www.sprace.org.br/sprace-computing-center>. Acesso em: 14 jul. 2014.

SMITH,J. E.. **The architecture of virtual machine**. IEEE Computer Society, p. 32-38, mai. 2005.

VALKENBURG, Mac E. Van. **Network Analysis.** 3. Ed. New York. Prentice Hall College, 1974.

VIRTUALBOX (2014). Oracle. Fonte: VirtualBox. Disponível em: <https://www.virtualbox.org/>. Acesso em: 15 set. 2014.

XML(2014). XML. Fonte: xml. Disponível em: <www.xml.com>. Acesso em: 30 de set. 2014.

WALDROP, M. Mitchell (2002). **Grid Computing**. MIT Technology Review. Fonte: technologyreview. Disponível em: <http://www.technologyreview.com/featuredstory/401444/grid-computing/>. Acesso em: 01 set. 2014.

WORLD COMMUNITY GRID (2014). IBM Project. Fonte: worldcommunitygrid. Disponível em: <http://www.worldcommunitygrid.org/about_us/viewAboutUs.do>. Acesso em: 01 set. 2014.