CRISTIANO MARCELO RESENDE

Ambiente Grid utilizando Software Livre

Monografia de Pós-Graduação “Lato Sensu” apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em “Administração em Redes Linux”.

Lavras
Minas Gerais - Brasil
2010

CRISTIANO MARCELO RESENDE

Ambiente Grid utilizando Software Livre

Monografia de Pós-Graduação “Lato Sensu” apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em “Administração em Redes Linux.

Área de Concentração:

Processamento Paralelo e Distribuído

Orientadora:

Profa. Dr. Marluce Rodrigues Pereira

Lavras
Minas Gerais - Brasil
2010

Dedico a minha esposa Ana Paula e a toda minha família.

Sumário

Resumo

A computação em *grid* é um modelo capaz de alcançar uma alta taxa de processamento dividindo as tarefas entre diversas máquinas, podendo ser em rede local ou rede de longa distância. É considerada uma boa solução para o problema enfrentado pelas organizações ou instituições de diversos fins, no processamento e armazenamento de uma quantidade cada vez maior de dados. Existem muitas soluções *open-source* em *middlewares* para construção de um ambiente *grid*, entretanto há alguns problemas, como a complexidade de implantação e a integração com projetos ou aplicativos. O *GridGain* é um projeto que busca sanar esses pontos, provendo uma solução de fácil integração com projetos Java, abstraindo ao máximo a complexidade de implantação. Porém há menos preocupação com a segurança o que é fortemente explorado em outras ferramentas como o *Globus Toolkit* [14]. Este trabalho aborda as principais características e componentes de um ambiente de *grid*, apresentando algumas ferramentas existentes e focando no *framework GridGain*, o qual foi instalado, configurado e testado em um pequeno ambiente de teste, demonstrando suas vantagens e desvantagens.

Palavras chave: *GridGain; grid; middleware*.

Abstract

The grid computing is a model capable of achieving a high rate of dividing the processing tasks between several machines may be on the LAN or WAN. It is considered a good solution to the problem faced by organizations or institutions for various purposes, processing and storing an increasing amount of data. There are many solutions on open-source middleware for building a Grid environment, however there are some problems, such as the complexity of deployment and integration projects or applications. The GridGain is a project that seeks to address these points, providing a solution for easy integration with Java projects, leaving aside the most of the complexity of deployment. But there is less concern for security that is heavily exploited in other tools like the Globus Toolkit [14]. The objective of this work is to discusses the key features and components of a Grid environment, presenting some existing tools and focusing on the GridGain framework, which was installed, configured and tested in a small test environment, showing its advantages and disadvantages.

Keywords: GridGain; Grid, middleware.

Capítulo 1

INTRODUÇÃO

Atualmente, tanto em áreas científicas quanto em ambientes corporativos, existe uma demanda crescente em processamento e armazenamento de dados, o que torna cada vez maior a necessidade de recursos computacionais que possam realizar tarefas de forma rápida e confiável.

Existem pesquisas envolvendo áreas como química, biologia, física e matemática, que demandam por recursos computacionais que geralmente podem ser fornecidos apenas por um supercomputador. No meio corporativo, a necessidade de mais recursos computacionais não é diferente. Novas ferramentas são criadas, a fim de diminuir o tempo com desenvolvimento e custos com os projetos de software sem que haja perda de qualidade. Entretanto, estas ferramentas exigem cada vez mais recursos computacionais, e neste caso, a aquisição de um supercomputador para cada membro de uma equipe, torna-se inviável pelo seu alto custo e complexidades relacionada à migração de sistemas locais para um sistema operacional que tire proveito do hardware do supercomputador. Neste contexto, a computação em *grid*[1] (do inglês *grid computing*) pode ser a solução ideal para estes problemas, pois através dela é possível utilizar a capacidade ociosa de vários computadores diferentes, separados fisicamente, ou até mesmo geograficamente, para executar tarefas ou armazenamento de dados.

[1] Modelo computacional capaz de alcançar uma alta taxa de execução de tarefas dividindo-as entre diversos computadores. [1]

Uma das principais estratégias da computação em *grid* é a utilização de um *middleware*[2], que atua como uma camada de *software* entre os computadores que compõem sua infra-estrutura, e a aplicação que irá utilizá-la. Seu principal objetivo é manter a transparência, de modo que um usuário possa utilizar uma aplicação que está sendo processada em vários computadores distintos, separados fisicamente, da mesma forma que utiliza em uma estação de trabalho comum.

Este trabalho tem como objetivo geral, descrever as características dos principais *middlewares open-souce* que são utilizados com mais frequência, e como objetivo especifico, descrever as principais características e dar ênfase a instalação e configuração de um *middleware* que possa ser integrado a aplicativos de forma simples, que seja altamente compatível com tecnologias emergentes no mercado e que tenha baixa complexidade em sua instalação, configuração, e utilização.

Os demais capítulos deste texto estão organizados da seguinte forma. O capítulo 2 apresenta o referencial teórico no qual este trabalho foi embasado. O capítulo 3 aborda a metodologia utilizada juntamente com os métodos utilizados e as instalações feitas no ambiente de *grid*. O capítulo 4 apresenta os resultados obtidos e os testes efetuados no ambiente. E o capítulo 5 traz a conclusão do trabalho juntamente com possíveis trabalhos futuros a serem realizados.

[2] Camada de software que possibilita comunicação entre aplicações distribuídas, tendo por objetivo diminuir a complexidade e heterogeneidade dos diversos sistemas existentes, provendo serviços que realizam comunicação entre as aplicações de forma transparente. [2]

Capítulo 2

REFERENCIAL TEÓRICO

Este capítulo apresenta alguns conceitos existentes na literatura sobre computação em *grid*, os serviços necessários para realizá-la e as principais ferramentas *open-source* existentes para prover tais serviços.

2.1 Conceitos

Na literatura, podem ser encontrados os termos *grid computing*, computação em *grid* ou computação em grade. Neste trabalho utiliza-se o termo computação em *grid*.

Computação em *grid* "é um modelo computacional capaz de alcançar uma alta taxa de processamento dividindo as tarefas entre diversas máquinas (...)." [1] Geralmente é utilizada com natureza científica, técnica ou para resolver problemas de negócios que requerem um grande número de ciclos de processamento ou a necessidade de processar grandes quantidades de dados.

"Em seu nível mais básico, a computação em grade é uma rede de computadores na qual os recursos do computador são compartilhados com todo e qualquer computador no sistema. Poder de processamento, memória e

armazenamento de dados são todos recursos comunitários que usuários autorizados podem explorar e aproveitar para determinadas tarefas.” [3]

Uma das principais estratégias da computação em *grid* está em utilizar um *software* para dividir e distribuir partes de um programa entre vários computadores, podendo chegar até milhares de máquinas realizando uma determinada tarefa. O tamanho de uma *grid* pode variar, sendo pequeno, limitado a uma rede de estações de trabalho em uma empresa, por exemplo, ou com o foco em um grande público, através da colaboração de muitas instituições e redes distintas.

Ian Foster [5] traduz os conceitos de *grid* de duas formas clássicas: “compartilhamento de recursos coordenados e resolução de problemas em organizações virtuais multi-institucionais dinâmicas” e “sistemas de suporte à execução de aplicações paralelas que acoplam recursos heterogêneos distribuídos, oferecendo acesso consistente e barato aos recursos, independente de sua posição física”. Além disso, a redução do custo de computadores e a melhoria do desempenho das redes nos últimos anos possibilitam agregar recursos computacionais variados e dispersos em um único “supercomputador virtual”, acelerando a execução de várias aplicações paralelas.

A principal vantagem desta arquitetura é que cada nó pode ser adquirido de um computador convencional, e quando combinados podem produzir recursos de computação semelhante a um supercomputador com múltiplos processadores, mas a um custo menor. A desvantagem principal é o desempenho na comunicação entre várias máquinas locais, onde geralmente não se tem conexões de alta velocidade. Este arranjo é, portanto, bem adequado para aplicações nas quais múltiplos processamentos paralelos podem ocorrer de forma independente, sem a necessidade de comunicação constante entre os nós. Dessa forma, a

escalabilidade de redes dispersas geograficamente é geralmente favorável.

Em analogia aos supercomputadores, há ainda algumas diferenças na programação e implantação. Pode ser caro e difícil escrever programas para serem executados em um ambiente de um supercomputador, que pode ter um sistema operacional personalizado, ou exigir que o programa resolva questões de concorrência. Se um problema pode ser adequadamente paralelizado, ou seja, dividido em partes menores a serem resolvidos ao mesmo tempo, uma pequena infra-estrutura de *grid* pode permitir que programas convencionais independentes, executem em vários nós. De acordo com [49] isto significa que uma tarefa é dividida entre o número de nós existentes, onde um nó pode ser inclusive uma sub-rede que pode ser composta de um ou mais computadores.

A figura 2.1[3] ilustra a arquitetura típica de um sistema *grid*.

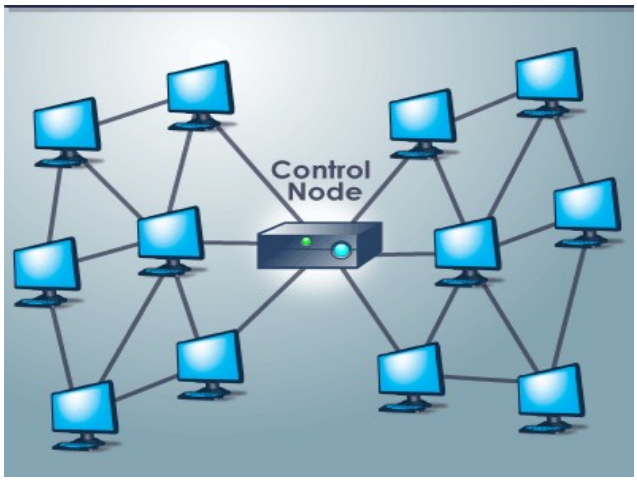

Figura 2.1: Arquitetura de um sistema *grid*.

[3] http://www.nusacm.org/newsletter/vol11.html

2.2 Histórico

O termo computação em *grid* surgiu na década de 1990. Segundo *Foster et al.* [5], a palavra *grid é* usada por analogia à rede elétrica ("*power grid*"), que provê acesso pervasivo à eletricidade. Este modelo de infra-estrutura apresenta ao usuário um computador virtual, mascarando toda a infra-estrutura distribuída, assim como a rede elétrica para uma pessoa que utiliza uma tomada sem saber como a energia chega a ela. Além disso, permite que tecnologias heterogêneas e geograficamente distantes componham um sistema robusto, dinâmico e escalável onde se possa compartilhar processamento, espaço de armazenamento, dados, aplicações, dispositivos, entre outros.

Em 1995, o *Information Wide Area Year*[4] (projeto experimental *I-WAY*) foi criado a partir da tecnologia *grid*. O projeto consistiu em criar uma rede de supercomputadores com tecnologia ATM[5], onde o poder computacional foi utilizado para testar aplicações de realidade virtual. Este projeto foi importante para a computação em *grid*, pois foi o precursor do *Globus*.[6] [6]

No final da década de 1990 a popularidade da Internet cresceu e já era utilizada por milhões de usuários diariamente. [51] isto fornecia escalabilidade aos projetos de *grid computing*. Neste contexto, dois projetos se destacaram: o *Distributed.net*[7] e o *SETI@home*[8].

O *Distributed.net* foi fundado em 1997 [7] e utiliza recursos de

4 http://www.nitrd.gov/pubs/bluebooks/1997/i-way.html

5 Tecnologia de comunicação de dados de alta velocidade usada para interligar redes locais, metropolitanas e de longa distância para aplicações de dados, voz, áudio, e vídeo. [8]

6 Conjunto de ferramentas ou serviços de código aberto para construção de *grids*.

7 http://www.distributed.net/

8 http://setiathome.berkeley.edu/

computadores convencionais para grandes projetos. Milhares de usuários doam seu poder de processamento baixando um programa que utilizam em seus computadores quando os mesmos estão ociosos. Neste mesmo ano, o projeto RC5-64 foi iniciado. O desafio do projeto era identificar uma chave secreta de 64 bits. Outro grande desafio era coordenar os servidores que distribuíam os blocos chave para as máquinas dos usuários. O tempo estimado para concluir a tarefa foi determinado em 1.235 anos no início do projeto. Em menos de um mês, a estimativa caiu para 104,5 anos devido à adesão de novos usuários doando seu poder de processamento e a revisão permanente da rede. [9]

Em julho de 2002, a chave vencedora foi encontrada, o que demorou em torno de apenas cinco anos. Isso mostra que a computação em *grid* na história aumentou exponencialmente o processamento de tarefas complexas em uma linha de tempo significativamente reduzida. [9]

Atualmente, um novo projeto está sendo formado, o RC5-72 para identificar uma chave secreta de 72 bits. *Distributed.net* hospeda também outros projetos que envolvem cálculos matemáticos intensos. [10]

Outro projeto histórico de computação em *grid* é o *SETI@home*. Milhões de usuários têm feito *download* do software desde o início do projeto em 1995. O SETI - *Search for Extraterrestrial Intelligence* (busca de inteligência extraterrestre) busca por sinais extraterrestres na galáxia. O processo de coletar e analisar os dados envolve o poder de processamento de milhões de computadores. Desde 1995, o projeto tem registrado 1.3 milhões de anos em tempo computacional. [11]

As ideias em computação em *grid* (inclusive sobre computação distribuída, programação orientada a objetos e *Web Services*) foram reunidas por *Carl Kesselman*, *Steve Tuecke e Ian Foster*, este último, considerado como o

“pai” da computação em *grid* [12]. Eles se uniram para criar o *Globus Toolkit* incorporando não apenas a gestão de processamento, mas gestão de armazenamento, segurança, transferência de dados e monitoramento. Além desses, foi incorporado também, um conjunto de ferramentas para desenvolvimento de serviços adicionais com base na mesma infra-estrutura, incluindo mecanismos de notificação, os serviços de *trigger*, e agregação de informações. Enquanto o *Globus Toolkit* continua a ser o padrão para a construção de soluções *grid*, uma série de outras ferramentas tem sido construída a fim de prover alguns subconjuntos de serviços necessários para implantação do ambiente, seja em redes locais ou globais.

2.3 Ferramentas livres de desenvolvimento

Atualmente existem várias ferramentas livres que facilitam a utilização da tecnologia *grid*. A seguir, tem-se a descrição de algumas implementações *open-source* existentes de *middlewares* de computação em *grid* que definem estaticamente o *grid* para um usuário.

2.3.1 *gLite*

Nascido a partir do trabalho de mais de 80 pessoas em 12 diferentes centros de pesquisas acadêmicas e industriais, como parte do projeto EGEE[9] (*Enabling Grids for E-sciencE*), o *gLite*[10] fornece um conjunto completo de serviços para a construção de aplicações *grid*. Os serviços do *gLite* são atualmente adotados por mais de 250 Centros de Informática e utilizados por mais de 15.000 pesquisadores na Europa e no mundo (Taiwan, América Latina, etc.) [13]

Segundo [1], a comunidade de usuários *gLite* está agrupada em organizações virtuais (*Virtual Organizations* - VOs). Um usuário deve participar de um VO suportado pela infra-estrutura em execução para ser autenticado e autorizado a utilizar os recursos de um *grid*. Sua infra-estrutura de segurança - *Grid Security Infrastructure* (GSI) permite a autenticação segura e comunicação através de uma rede aberta. O GSI é baseado em criptografia de chave pública, protocolo de comunicação certificados X.509, e *Secure Sockets Layer*[11] (SSL) [14]. Para se autenticar, o usuário precisa ter um certificado digital X.509 emitido por uma Autoridade Certificadora (CA) confiável para o ambiente em execução do *middleware*. A autorização de um usuário em um recurso *grid* específico pode ser feita de duas maneiras diferentes. A primeira é simples e baseia-se no mecanismo *grid-mapfile*. A segunda forma depende de *Virtual Organisation Membership Service* (VOMS) e mecanismos LCAS / LCMAPS, que permitirão uma definição mais detalhada dos privilégios do usuário. [16]

---

[9] http://*gLite*.web.cern.ch/*gLite*/

[10] http://*gLite*.web.cern.ch/*gLite*/

[11] Padrão global em tecnologia de segurança, onde é criado um canal criptografado para garantir que todos os dados transmitidos sejam sigilosos e seguros. [15]

2.3.2 *Globus Toolkit*

O *Globus Toolkit* (GT), atualmente na versão 4, é um conjunto de ferramentas ou serviços de código aberto para construção de *grids* desenvolvido e fornecido pela *Globus Alliance*[12].

Dentre os principais serviços oferecidos pelo *Globus*, pode-se destacar:

a) ***Globus Resource Allocation Manager* (GRAM):** "nível mais baixo da arquitetura de gerência de recursos do *Globus*. Este serviço permite executar *jobs* remotamente, controlar e monitorar cada *job*. O GRAM é encarregado de administrar um conjunto de máquinas, sejam elas máquinas paralelas, *cluster* ou estações de trabalho." [17]

b) **Mecanismos de Segurança:** "O principal desafio de um ambiente *grid* em relação à segurança é disponibilizar um sistema capaz de se adaptar bem ao ambiente dinâmico e à distribuição dos usuários por domínios administrativos diferentes. A infraestrutura de segurança do *Globus* (*Grid Security Infrastructure*) trata da autenticação de entidades no sistema. O objetivo do GSI é permitir que uma vez autenticado, o usuário receba uma credencial que o permita acessar os recursos sem a necessidade de se autenticar novamente. (...)" [17]

c) ***Grid Security Infrastructure* (GSI)**: "trata da autenticação de entidades no sistema. O objetivo do GSI é permitir que uma vez autenticado, o usuário receba uma credencial que o permita acessar os recursos sem a necessidade de se autenticar novamente. As credenciais são mapeadas para os mecanismos locais de autenticação de cada domínio

[12] http://www.globus.org/

administrativo, permitindo que diferentes ambientes participem da *grid* sem a necessidade de alterar suas políticas locais.” [17]

“Esses serviços permitem a submissão e controle de aplicações, descoberta de recursos, movimentação de dados e segurança no ambiente do *grid*. Tendo sido a solução de maior impacto na comunidade da computação de alto desempenho, o *Globus* e os protocolos definidos em sua arquitetura tornaram-se um padrão *de fato* como infra-estrutura para computação em *grid*.” [18]

Ainda de acordo com [18], apesar de resolver satisfatoriamente diversas questões do aspecto operacional, o *Globus* não ataca alguns problemas do *grid* do ponto de vista gerencial. Utilizando o mecanismo de acesso hoje disponível, são necessárias negociações com os donos de recursos para obter o acesso a estes e há necessidade do mapeamento dos clientes para usuários locais, fatos que limitam a escalabilidade deste mecanismo.

2.2.3 BOINC (*Berkeley Open Infrastructure for Network Computing)*

BOINC[13] é um *middleware* distribuído sob a licença GNU (*General Public License*) utilizado em *grid computing*. Foi desenvolvido no Laboratório de Ciências Espaciais, na Universidade da Califórnia, *Berkeley*, por uma equipe liderada por David Anderson. Seu propósito inicial era de dar suporte ao projeto *SETI@home*, mas acabou se tornando útil também como plataforma para outros aplicativos distribuídos em áreas tão diversas como a Matemática, Medicina,

[13] http://boinc.berkeley.edu/

Biologia Molecular, Climatologia e Astrofísica [19]. Seu objetivo é tornar possível explorar o enorme poder de processamento dos computadores pessoais em todo o mundo. Quando os colaboradores não estão usando toda a capacidade de seus computadores pessoais, um programa a utiliza (sempre em prioridade mais baixa para não atrapalhar outros programas ou tornar o uso do computador mais lento).

A participação no projeto é aberta para qualquer pessoa. Faz-se necessário apenas o *download* do *software* e cadastro em um dos projetos participantes. Todos os projetos que utilizam o BOINC são projetos sem fins lucrativos e não gerarão registro de patentes para empresas privadas.

De acordo com [20], suas principais características são:

a) *Framework* flexível para aplicações;

b) Tolerância a falhas e múltiplos servidores;

c) Possui ferramentas de monitoramento;

d) Código fonte disponível;

e) Suporte a um grande volume de dados e API para criação de *work unit* no Banco de Dados.

2.2.4 *XtremWeb*

O *XtremWeb*[14] é um projeto pesquisa de código aberto relacionado a sistemas de *grids* leves (*Light Weight Grid Systems*) mantido pelo *Laboratoire de Recherche en Informatique* (LRI) em Paris, França [21].

O projeto surgiu devido a uma solicitação realizada por físicos do *Pierre Auger Observatory*. A cada dia eles necessitavam executar milhares de vezes uma mesma aplicação com entradas diferentes [22].

Os principais elementos da arquitetura do *XtremWeb* são, o *client* que representa a máquina que está submetendo um *job*, o *worker* que representa a máquina que executará o *job* e o *server* que representa o elemento central da arquitetura e é responsável pelo gerenciamento dos *jobs* nele armazenados. O protocolo de comunicação é composto por quatro passos apresentados a seguir [22]:

a) *O worker* efetua registro no servidor através de uma requisição *hostRegister*.

b) Solicita um *job* ao servidor através de uma requisição *workRequest*. O servidor seleciona uma tarefa e entrega ao *worker* uma descrição da tarefa e os parâmetros de entrada. Se o mesmo ainda não tinha executado nenhuma tarefa da aplicação correspondente a tarefa que está sendo entregue, é feito o *download* do binário da aplicação. Caso contrário, o binário já está armazenado no *worker*, não necessitando efetuar o *download*.

c) Durante o processamento da tarefa, o *worker* informa ao *server* que está

[14] http://www.xtremweb.net/

ativo através de uma requisição *workAlive*. O *server* monitora estas requisições para implementar um protocolo de *timeout*, com o objetivo de escalonar novamente a mesma tarefa caso o *worker* permaneça muito tempo sem enviar uma requisição.

d) Ao terminar a execução da tarefa, o *worker* envia os resultados através de uma requisição *workResult* e o servidor atualiza a situação da tarefa, indicando que a mesma está concluída.

2.2.5 ARC (*Advanced Resource Connector*)

ARC é um *middleware open-source*, distribuído sob a licença Apache[15], introduzido pelo *NorduGrid.*[16] Surgiu (e ainda é muitas vezes referido), como o *NorduGrid middleware*, originalmente proposto como uma arquitetura acima do *Globus Toolkit* , otimizado para as necessidades de experimentos físicos como o processamento de dados do colisor de Hádrons LHC[17]. [23]

A primeira implantação do ARC no *NorduGrid* se deu no verão de 2002 e em 2003 foi utilizado para dar suporte a complexos desafios computacionais [24] .

A primeira versão estável do ARC (versão 0.4) foi lançada em abril de 2004 [25], sob a licença GNU (*General Public License*). O nome "*Advanced*

[15] http://www.apache.org/licenses/

[16] Projeto de pesquisa que visa o desenvolvimento, manutenção e suporte de um *middleware Grid open-source*, conhecido como o *Resource Advance Connector* (ARC).

[17] Acelerador circular de partículas, instalado a cem metros de profundidade no subsolo, na fronteira entre França e Suíça [27]

Resource Connector" foi introduzido nesta versão, a fim de distinguir o *middleware* da infra-estrutura. Em 2005, o *NorduGrid* foi formalmente criado com o objetivo de apoiar e coordenar o desenvolvimento do ARC [26] . Este fato abriu caminho para mais investimentos em desenvolvimento, e em 2006 dois projetos estreitamente relacionados foram lançados: o *Nordic Data Grid Facility*[18] e o *KnowARC*[19], este último centrado na transformação do ARC em uma próxima geração de *middleware grid*. A versão 0.6 foi lançada em maio 2007 [28], sendo esta, o segundo lançamento estável. Sua característica principal era a introdução de bibliotecas no cliente que permite o desenvolvimento fácil de aplicações em alto nível. Foi também a primeira versão fazendo uso de padrões abertos, como por exemplo, o apoio à JSDL (*Job Submission Description Language*)[20].

A maioria dos serviços do ARC são fornecidos através da camada de segurança GSI. O *middleware* padrão baseia-se em soluções *open source*, como o *OpenLDAP*[21] , *OpenSSL*[22] , *SASL (Simple Authentication and Security Layer)*[23] e bibliotecas *Globus Toolkit (GT)*. *NorduGrid* fornece soluções inovadoras, essencial para um *middleware* de qualidade em produção: o *Grid* Manager, o *gridftpd*, o modelo de informação e provedores (esquema *NorduGrid*), interface com o usuário, e o sistema de monitoramento. [29]

O ARC não utiliza a maioria dos serviços do *Globus Toolkit*, como o GRAM, comandos de submissão de *jobs*, servidor *GridFTP* baseado em *WUftp*,

18 http://www.ndgf.org/ndgfweb/home.html
19 http://www.knowarc.eu/
20 Linguagem utilizada para descrever as condições de computacionais [30]
21 Software que implementa o protocolo LDAP (protocolo para atualizar e pesquisar diretórios rodando sobre TCP/IP). [31]
22 Implementação de código aberto dos protocolos SSL e TLS [32]
23 Método para adicionar suporte à autenticação baseada em protocolos de conexão. [34]

scripts GRAM *job-manager*, esquemas ou provedores de informação MDS. Entretanto ele estende diversas funcionalidades do mesmo, sem alterar o código original, o que o torna uma poderosa solução *grid.* [29]

A documentação completa, incluindo descrições de componentes, detalhes de instalação, instruções de utilização, artigos e apresentações, podem ser encontrados na página de documentação[24]·.

O ARC é projetado para ser uma solução não intrusiva, escalável e portátil. O desenvolvimento é orientado a usuário e a aplicação, onde os requisitos principais são os de melhor desempenho, estabilidade, usabilidade e portabilidade. Como resultado desta abordagem, o cliente autônomo está disponível para uma dezena de plataformas e pode ser instalado em poucos minutos. A instalação do servidor não exige uma reconfiguração completa no site. O *middleware* pode ser construído em qualquer plataforma onde os pacotes de softwares externos (como bibliotecas *Globus Toolkit*) estão disponíveis. Ao ser implantado em uma grande produção *grid* e sendo usado por usuários reais, o *middleware* é naturalmente submetido a testes contínuos em tempo real. [29]

As ferramentas de cliente estão disponíveis em vários idiomas. O monitor *grid* é traduzido em mais de 10 idiomas, e a interface do usuário não está disponível apenas em Inglês, mas também em russo e sueco. [29]

[24] http://www.nordugrid.org/papers.html

2.4 Ferramenta Escolhida

Ao lidar com a diversidade de hardware e software inerentes a ambientes paralelos e, especialmente, *grids*, Java[25] ganhou muita popularidade devido a sua propriedade "escreva uma vez, rode em qualquer lugar", o que promove a independência de plataformas. [50] Neste contexto, como ferramenta escolhida, optou-se por um *framework* Java que promete ser o *middleware grid* mais simples de ser configurado que existe atualmente. Além disso, segundo [49], possui um *design* moderno com suporte a *cloud computing* (computação em nuvem). Com foco em redes locais e no processamento paralelo de aplicações novas ou já existentes, este *framework* consegue inclusive paralelizar blocos de códigos isolados, em um determinado programa. È possível também, colocar em um ambiente *grid*, aplicações que já estejam a muito tempo no mercado, sem que as mesmas tenham sido projetadas com a intenção de serem executadas sob um *grid* algum dia, sem mesmo alterar o código fonte destas aplicações. Este *framework* se chama *GridGain*, e as subseções a seguir descrevem seus conceitos e principais características.

[25] Linguagem de programação que permite o desenvolvimento de aplicações para uma série de plataformas. É possível ter software Java desde dispositivos pequenos, como telefones celulares, até computadores de grande porte, como os mainframes, por exemplo. [34]

2.4.1 *GridGain*

GridGain é um *framework open-source* utilizado na computação em *grid*, construído utilizando a linguagem de programação Java, que permite desenvolvedores desta mesma linguagem, melhorar o desempenho do processamento de aplicações dividindo e paralelizando a carga de trabalho. Com *GridGain*, os desenvolvedores podem obter melhor rendimento global, melhor escalabilidade e disponibilidade dos serviços [35].

Segundo [35], o projeto começou em *Pleasanton*, Califórnia, em 2005, por um grupo de pessoas que achavam as soluções tradicionais de computação em *grid*, ainda muito inadequadas para a maioria das empresas, devido à sua complexidade, alto custo e a falta de integração simples e com tecnologias utilizadas nas mesmas, como Java EE[26] (ou simplesmente JEE), por exemplo. A ideia era construir um *framework* de código aberto para computação em *grid* em ambientes corporativos, que fosse facilmente integrado a ambientes JEE, com uma plataforma simples de instalar, configurar e de fácil implementação.

2.4.2 Principais Características

A seguir são descritas algumas das principais características e recursos presentes no *GridGain*.

[26] O JEE (**J**ava ***E****nterprise* ***E****dition*) é a plataforma Java voltada para redes, internet, intranets e afins. Assim, ela contém bibliotecas especialmente desenvolvidas para o acesso a servidores, a sistemas de e-mail, a banco de dados, etc. [34].

2.4.2.1 Suporte a outros *Grids* através de POA e *Annotations*

Segundo o site do projeto, *GridGain* é a única infra estrutura de *grid computing* que permite ativar códigos de *grids* existentes, sem precisar de modificá-los. Isto é possível utilizando Java *annotations*[27] e o paradigma POA[28] (Programação Orientada a Aspectos). Em alguns casos, como por exemplo, com o *Jboss*[29], não é preciso nem mesmo ter o código fonte para habilitar o *grid*, pois anotações e POA podem ser aplicadas através de um arquivo externo. [36]

2.4.2.2 Integração e customização baseada em SPI (*Service Provider Interface)*

O centro de configuração e customização do *GridGain* é baseado em SPI (*Service Provider Interface)*. Com isso, todas as áreas funcionais mais importantes de sua infra-estrutura são completamente customizáveis, permitindo aos desenvolvedores personalizarem praticamente todos seus aspectos funcionais, como: comunicação entre nós, gerenciamento de *deploy*[30], de

---

27 Forma especial de sintaxe que pode ser adicionado ao código-fonte Java. È uma forma de meta-programação, que provê informações para o compilador ou um *framework* como o *Jboss*. [53]

28 Paradigma de programação que permite aos desenvolvedores de software separar e organizar o código de acordo com a sua importância para a aplicação, introduzindo o conceito de aspectos, os quais encapsulam comportamentos que afetam múltiplas classes. [37]

29 Servidor de aplicação de código fonte aberto baseado na plataforma J2EE implementada completamente na linguagem de programação Java. [38]

30 Em português significa “implantar”, é a instalação ou cópia da aplicação para um servidor de aplicações, de forma que a mesma seja disponibilizada para outros usuários. Muitas vezes este processo envolve a criação de *scripts* e outras tecnologias.

topologia, balanceamento de carga e resolução de colisão. As implementações de tarefas no *grid* (lógica de negócios reais que estão em execução) são mantidos inalteradas por diferentes SPIs executando em segundo plano. [36]

2.4.2.3 Balanceamento avançado de carga e agendamento

O balanceamento de carga do *GridGain* é definido pela resolução de colisão (agendamento) dos SPIs de forma eficaz, permitindo a customização completa de todo o processo. Este balanceamento permite adaptar a execução da tarefa *grid* para uma natureza não determinística de execução no próprio *grid*. Na verdade, este ambiente é geralmente heterogêneo e não estático, as tarefas podem alterar os seus perfis de complexidade dinamicamente em tempo de execução e os recursos externos podem afetar a execução da tarefa em qualquer ponto. Todos esses fatores reforçam a necessidade de um balanceamento de carga dinâmico durante a operação, bem como no nó de destino onde os *jobs* podem estar em filas de espera. [36]

2.4.2.4 Tolerância a falhas

Gerência de *Failover*[31] e tolerância a falhas são propriedades fundamentais de qualquer sistema *grid computing*. Baseado em sua arquitetura SPI, o *GridGain* fornece uma lógica *failover* totalmente compatível com várias implementações populares. Com uma tarefa *grid* sendo uma unidade atômica em execução com lógica *failover* totalmente customizável, é possível ao desenvolvedor escolher políticas específicas da mesma maneira como seria escolher a política de concorrência em transações em sistemas de banco de dados relacionais. [36] Isto permite ajustar como uma tarefa *grid* reage ao falhar, por exemplo:

a) Abortar a tarefa inteira logo após a falha de qualquer um dos seus *jobs*.

b) Repassar qualquer *job* que falha para outros nós até que todos os nós estejam ocupados.

2.4.2.5 Livre de *scripts* de *deploy*

Uma das características do *GridGain* que tem um efeito imediato e crucial na produtividade dos desenvolvedores é o seu modelo de *deploy*. Em *GridGain* pode-se começar com múltiplos nós no mesmo computador ou mesmo na mesma VM (*Virtual Machine*) tudo sem qualquer alteração na configuração.

[31] Processo no qual uma máquina assume os serviços de outra, quando esta última apresenta falha [39]

Além disso, através de sua tecnologia carregando classes *peer-to-peer* (P2P)[32] não é preciso executar quaisquer *scripts Ant*[33], copiar quaisquer arquivos, iniciar ou reiniciar nós, não é preciso fazer quaisquer medidas específicas, basta alterar o código. O *loading* de classes P2P irá cuidar de todo código alterado, repassando para os nós remotos.

O simples fato de que pode-se desenvolver, testar e depurar complexos algoritmos distribuídos em diversos nós, utilizando aplicativos em um único *laptop* sem sair de sua IDE e sem qualquer medida extra, resulta em economia de meses e meses em tempo de desenvolvimento. [36]

2.4.2.6 Gerenciamento e Monitoramento baseado em JMX

GridGain vem com uma extensa coleção de JMX[34] *Mbeans* que oferece um maior monitoramento das informações e estatísticas de todos os nós do *grid.* Estas informações estão disponíveis através de uma interface de programação, como qualquer compilador Web compatível com JMX ou visualizadores GUI

[32] Arquitetura de sistemas distribuídos caracterizada pela descentralização das funções na rede, onde cada nodo realiza tanto funções de servidor quanto de cliente. [40]

[33] Ferramenta escrita em Java e usada para a automatização de *builds* e tarefas. O uso do *Ant* é justificado devido à quantidade de tarefas que devem ser executadas antes que uma aplicação esteja pronta para instalação ou distribuição final. Entre estas tarefas estão a compilação de classes Java, a criação ou exclusão de diretórios, empacotamento de arquivos, execução de programas externos, etc. [41]

[34] Tecnologia Java que fornece ferramentas para gerenciamento de monitoramento de aplicações, objetos de sistema, dispositivos e redes orientadas a serviço. É uma Interface de Programação de Aplicativos (API), que usa o conceito de agentes, permite monitorar elementos da máquina virtual Java e dos aplicativos que estão sendo executados. [42]

autônomas, como o *Jconsole*[35] *que pode ser visto nas figuras*: 2.4.1 e 2.4.2. [36]

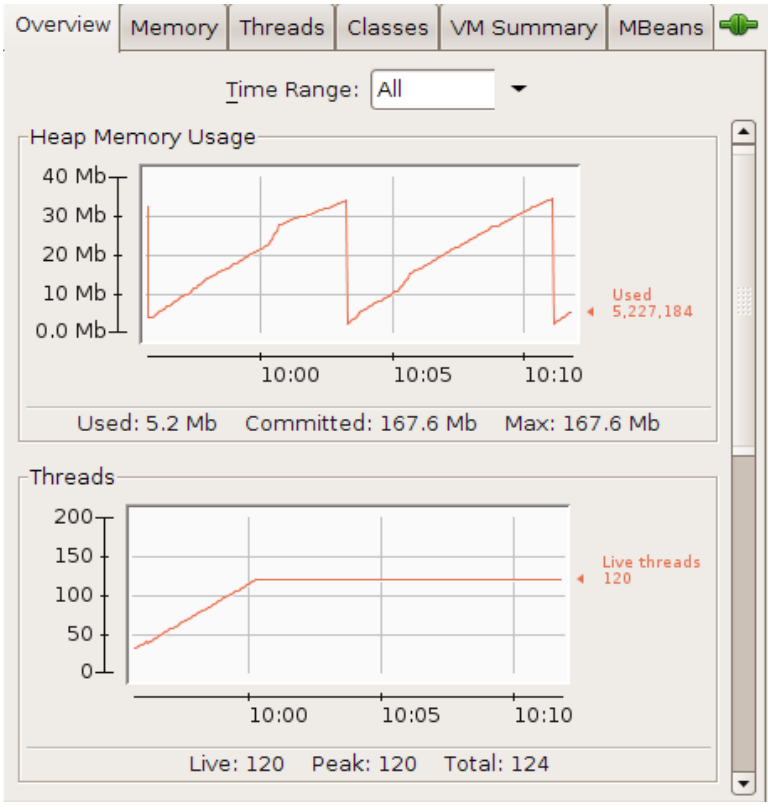

Figura 2.4.1: *jconsole* utilizado com *GridGain*

[35] Ferramenta de monitoramento compatível com JMX. [43]

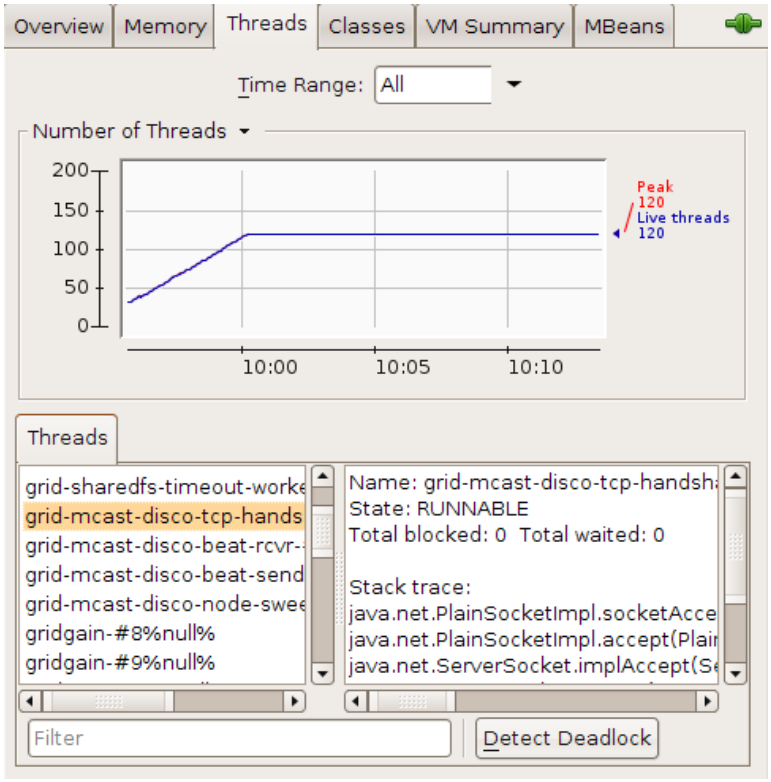

Figura 2.4.2: Outro exemplo do *jconsole* sendo utilizado com *GridGain*

2.5. Comparação entre as ferramentas

Esta seção apresenta uma comparação entre os diferentes *middlewares* para computação em *grid* abordados neste trabalho. A Tabela 2.5 mostra as principais características destas ferramentas de acordo com os dados informados em [13], [14], [16], [21], [26], [29], [36] e [48]:

Tabela 2.5: Comparação entre os diferentes *middlewares*

	Middleware					
Caracterís_ tica	***Globus Toolkit***	***gLite***	**BOINC**	***XtremWeb***	**ARC**	***Grid_ Gain***
Ano de criação	2004	2006	2002	2001	2002	2005
Gerência e monitora_ mento de recursos	Através do MDS[36]	Utiliza ferra_ mentas de terceiros.	BOINC *Manager*	Monitorame_ nto *Web* (PHP)	*Grid Monitor*	JMX *Mbeans*
Mecanismo de segurança	Certificados, autenticação, autorização e GSI	Chaves privadas, públicas e Certifica_ dos digitais.	Assina_ turas digitais baseadas em cripto_ grafia de chave pública.	Utiliza mecanismo *Sandboxing*[37]	Certifi_ cados, auten_ ticação e autori_ zação.	Supor_ ta[38]
Tolerância a falhas	Fraca [44]	SIM	SIM	SIM	SIM	SIM
Linguagem de programa_ ção utilizada	C	APIs disponíveis em C, C++, *Python* e Java	C	Java e acesso as rotinas críticas escritas em C	C++	Java

[36] *Monitoring and Discovering System.* Componente de serviços de informação do *Globus Toolkit*, que fornece informações sobre os recursos disponíveis no *grid* e seu *status*. [45]

[37] Técnica de segurança utilizada para criação de ambientes restritos que visa torná-los menos suscetíveis a ataques maliciosos. [46]

[38] Não possui uma implementação padrão, mas a arquitetura do *framework* permite que seja realizado utilizando SPIs.

Portal para autenticação e submissão de *jobs*	Suporta portais externos.	Possui portal próprio.	Possui interface GUI: BOINC *Manager*	Possui interface GUI.	NÃO	NÃO
Pré-requisitos para instalação	Java JDK 1.4.2 ou posterior, *Apache Ant* 1.5.1 ou posterior, gcc 3.2.1, *GNU tar*, *GNU sed*, *zlib* 1.1.4 ou posterior, *GNU Make*, *sudo*, *PostgreSQL* 7.1 ou posterior	Java JDK 1.4.2 ou posterior	*glibc* 2.3.2 ou posterior	JDK 1.4 ou posterior, *MYSQL* 4 ou posterior	*Grid Packa_ging Tools* (GPT) *Globus Toolkit* 4 (*pre-WS*)	JDK 1.5 ou supe_rior
***Hot Deploy*[39]**	NÃO	NÃO	NÃO	NÃO	NÃO	SIM
Suporta a vários nós na mesma máquina	NÃO	NÃO	NÃO	NÃO	NÃO	SIM

Dentre as ferramentas analisadas neste trabalho, o *GridGain* foi a que se mostrou mais propícia a integração com projetos de software existentes. Entre suas características podem se destacar o suporte a *annotations* e ao paradigma de programação POA, que permitem paralelizar apenas trechos de códigos de uma aplicação, ou uma aplicação inteira sem que seja necessário modificar o código fonte da mesma. Isso faz com que sua complexidade seja mínima, ao se tratar de integração com *software* existentes.

[39] Não é preciso reiniciar o *grid* e nem mesmo reinstalar a tarefa a cada mudança de código.

O fato de suportar vários nós em uma mesma máquina, permite um *debug* mais ágil, e menos trabalhoso, e seu poder de *hot deploy*, que permite que um trecho de código seja alterado em tempo de execução, sem que seja preciso reiniciar nenhum nó do ambiente, permite que manutenções no código de uma aplicação possam ser realizadas em tempo real, sem que seja necessário parar todo ambiente.

O *GridGain* entretanto, segundo [47], deixa a desejar nos quesitos de segurança, transferência e replicação de arquivos. Dessa forma, para aplicações de *Global computing* o *GridGain* não se apresenta como uma solução interessante quando comparado a *middlewares* com esse objetivo (por exemplo, *Globus Toolkit*, *Glite*, *BOINC*, entre outros). Este *framework* tem como objetivo ser um *Enterprise Grid level* e por isso seu principal diferencial é a fácil integração com *software* existentes, utilizando redes locais, onde existe mecanismos e políticas de segurança externas, sendo uma alternativa interessante para empresas ou outras instituições onde o tempo gasto na integração de um projeto de software a um ambiente *grid* deva ser mínimo.

Por estes motivos, o *GridGain* foi a ferramenta escolhida neste trabalho, onde criou-se um pequeno ambiente de testes afim de explorar sua configuração, desempenho, capacidade multiplataforma, outras vantagens e desvantagens.

Capítulo 3

Metodologia

No início do trabalho foi elaborada uma pesquisa sobre ambientes *grid*, para entender os detalhes do seu funcionamento e levantar em quais sistemas operacionais um ambiente *grid* poderia ser elaborado. A partir destes estudos optou-se por escolher o sistema operacional Linux na distribuição *Ubuntu* 9.10 onde será instalado e configurado o *GridGain* e a IDE Eclipse em uma das máquinas. Em uma segunda máquina será instalado apenas o *GridGain* com o sistema operacional *Windows* XP, para que se possa também avaliar a capacidade multiplataforma do ambiente.

Para efetuar as instalações necessárias foram utilizadas inicialmente duas máquinas, uma com processador Intel Core 2 Duo e outra com processador Intel Dual Core, nas quais foi instalado e configurado o ambiente *grid*. Como os processadores utilizados possuem dois núcleos, optou-se pelo uso de máquinas virtuais através da aplicação *Vmware*[40], a fim de aumentar o número de nós do ambiente. Entretanto, ao realizar os testes com duas máquinas virtuais, observou-se que o *Vmware* não foi capaz de isolar um núcleo do processador, para cada máquina virtual, pois o mesmo utiliza virtualização completa, o que resulta em um aproveitamento dos dois núcleos em todas as máquinas virtuais que sejam criadas. A partir deste problema e como virtualização foge ao escopo deste trabalho, realizaram-se os testes com mais dois computadores físicos, cada

[40] http://www.vmware.com/br/

um rodando o sistema operacional *Ubuntu* 9.10, totalizando quatro nós.

3.1 Pré-Requisitos para a Instalação do *GridGain*

Antes de iniciar a instalação e configuração do *GridGain* e de seus serviços é preciso instalar o Java JDK 1.5 ou posterior. Basicamente este é o único pré requisito para instalação do *GridGain*. Os passos para instalação e configuração do JDK no sistema operacional Linux são descritos na subseção 3.2.1.

3.2 Instalação e Configuração do *GridGain*

Esta seção apresenta um passo a passo, de como foi instalado e configurado o *GridGain*. Este pode ser instalado em diferentes tipos de sistemas operacionais, tais como Linux, *Mac* OS X e *Windows*. Vale ressaltar que não existem diferenças significativas na instalação de versões anteriores do *GridGain*, sendo recomendado instalar sempre a última versão disponível no site do projeto.

3.2.1 Instalação do JDK

Antes de instalar o *GridGain*, é preciso baixar e instalar o Java SE *Development Kit* (JDK), que está disponível no site da SUN[41]. De posse do JDK, segue alguns procedimentos para instalar e configurar o mesmo:

a)Adicionar permissão de execução no arquivo com o comando:

```
chmod + x jdk-x_x_x-linux-i586.bin
```

onde x_x_x é a versão do JDK.

b)Mover o arquivo para o diretório home:

```
mv jdk-x_x_x-linux-i586.bin ~/
```

c)Extrair o arquivo executando-o: .

```
./jdk-x_x_x-linux-i586.bin
```

d)Configurar as variáveis de ambiente:

```
sudo rm /etc/alternatives/java
sudo ln -s
/home/user/jdkx.x.x_xx/bin/java/etc/alternatives/java
```

e)Editar o arquivo "/etc/bashrc" e "/etc/profile" e acrescentar as seguintes linhas no final do arquivo:

```
export JAVA_HOME=/home/user/jdkx.x.x_xx
export PATH=:/home/user/jdkx.x.x_xx/bin:$PATH
```

[41] http://java.sun.com/javase/downloads/?intcmp=1281

3.2.2 Instalando o *GridGain* a partir de um arquivo binário

O *GridGain* pode ser obtido através do site do projeto[42]. De posse do arquivo de instalação, basta executá-lo com o comando:

```
./gridgain-unix-x.x.x.SH
```

e seguir o *wizard* do instalador como mostrado na figura 3.2:

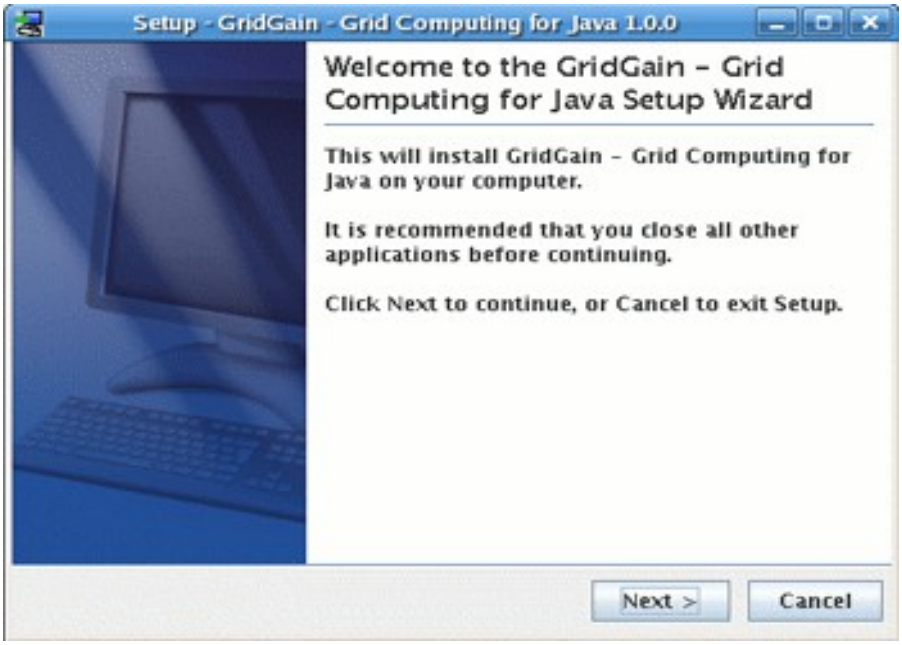

Figura 3.2: Instalação do *GridGain*

Se o instalador for chamado com o argumento "-c", a interação com o usuário é realizada no terminal onde o instalador foi chamado, isto é, em modo texto, sem a presença do *wizard* gráfico. Caso o instalador seja chamado com o argumento "-q", não há interação com o usuário, a instalação é feita

[42] http://www.*gridgain*.com/downloads.html

automaticamente com os valores padrões.

A instalação do *GridGain* pode ser feita ainda, através do arquivo "gridgain-unix-xxtar.gz", onde "xx" é a versão do mesmo. Neste caso, basta extrair o conteúdo do arquivo compactado com o comando:

```
tar xfz gridgain-unix-xxtar.gz
```

para que seja criado um diretório com todos os arquivos que são extraídos durante o *wizard*.

3.2.3 Configurando Variáveis de ambiente para o *GridGain*

Para que o *GridGain* funcione adequadamente é preciso configurar sua variável de ambiente. Para isso basta utilizar o comando:

```
export GRIDGAIN_HOME=/opt/gridgain
```

onde "/opt/gridgain" é o diretório padrão onde o *wizard* realiza a instalação. Assim como foi feito para o JDK, é preciso também configurar esta variável dentro dos arquivos: "bashrc" e "profile", presentes no diretório "/etc", adicionando esta linha no final dos arquivos, para que a variável esteja acessível por todo o sistema operacional.

3.3 Integrando aplicações com o *GridGain*

A construção de aplicações para ser usado sob o *GridGain*, pode ser feita através de qualquer IDE[43] Java. Neste trabalho optou-se pela IDE Eclipse[44] devido sua ampla popularidade e pela mesma ser *open-source*.

"A Eclipse *Foundation* é uma organização sem fins lucrativos, mantido por empresas que hospedam os projetos Eclipse e ajudam a cultivar tanto uma comunidade *open-source*, quanto um ecossistema de produtos e serviços complementares" [52].

É importante ressaltar que não existe diferenças significativas na criação de aplicativos utilizando outras IDEs como o *NetBeans*[45] por exemplo.

3.3.1 Baixando e instalando a IDE Eclipse

A IDE Eclipse está disponível no site Eclipse *Foundation*[46], e pode ser baixada sob a licença EPL (*Eclipse Public License*). O primeiro passo após baixar o Eclipse é descompactá-lo em um local de preferência e executar o arquivo "eclipse". Não é preciso compilar e nem mesmo executar nenhum *script* de configuração adicional. O único pré-requisito é que já esteja instalado o JDK e configuradas suas variáveis de ambiente como mostrado na subseção 3.2.1.

---

43 Ambiente de desenvolvimento integrado.
44 http://www.eclipse.org/
45 http://netbeans.org/
46 http://www.eclipse.org/org/

3.3.2 Criando um projeto e configurando o Eclipse para executar aplicações com o *GridGain*

A criação de um novo projeto no Eclipse é extremamente simples. Os passos a seguir descrevem como criar um novo projeto e adicionar as bibliotecas do *GridGain* para que o mesmo possa ser executado a partir do *grid*:

a) No eclipse, ir em File > *new Project* como mostra a figura 3.3 e 3.4.

b) Adicionar as bibliotecas "gridgain-x.x.x.jar" que se encontram no diretório "/opt/gridgain/", (onde "x.x.x" é a versão do *GridGain*) e todas outras, presentes em "/opt/gridgain/libs", ao projeto, clicando com botão direito do mouse sobre o projeto, indo em "propriedades" e acessando o menu "*Java Build Path*" como mostra figura 3.5.

c) Acrescentar ao eclipse o argumento:

```
-ea "-javaagent:/dados/gridgain/libs/aspectjweaver-
1.5.3.jar"
```

para que seja habilitada a programação orientada a aspectos na JVM (*Java Virtual Machine*). Para isso, deve-se selecionar o projeto e ir ao menu "*Run* > *Run Configurations*".

d) Acrescentar a pasta *aspectj* presente em "/opt/gridgain/config/aop" ao projeto, seguindo os mesmos passos do item b, e clicando no botão "*Add Class Folder*".

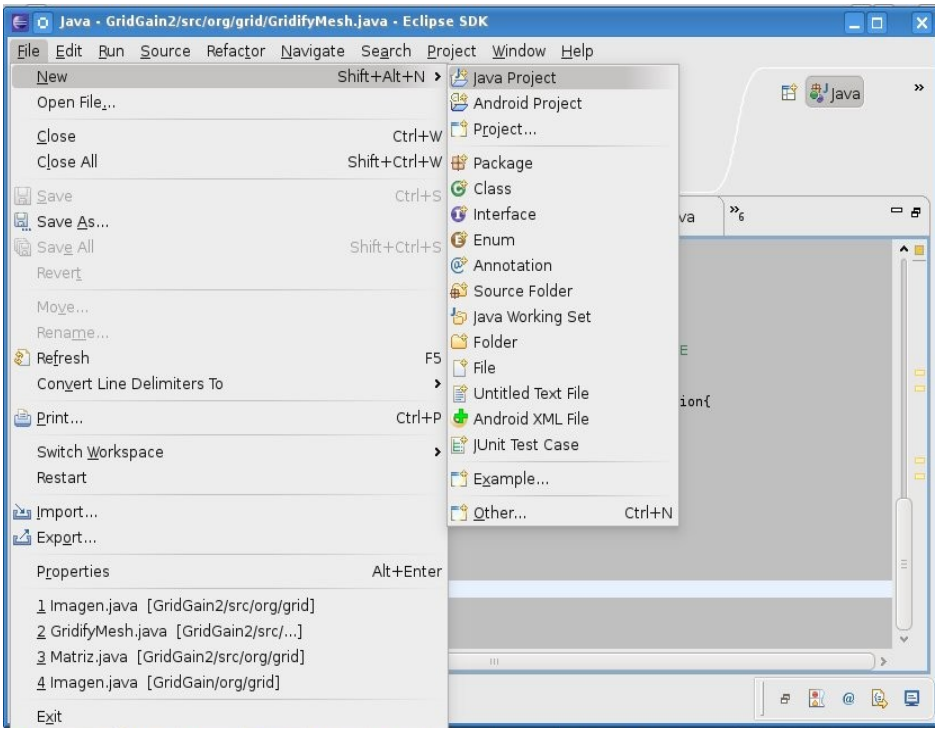

Figura 3.3: Criação de um novo projeto no Eclipse

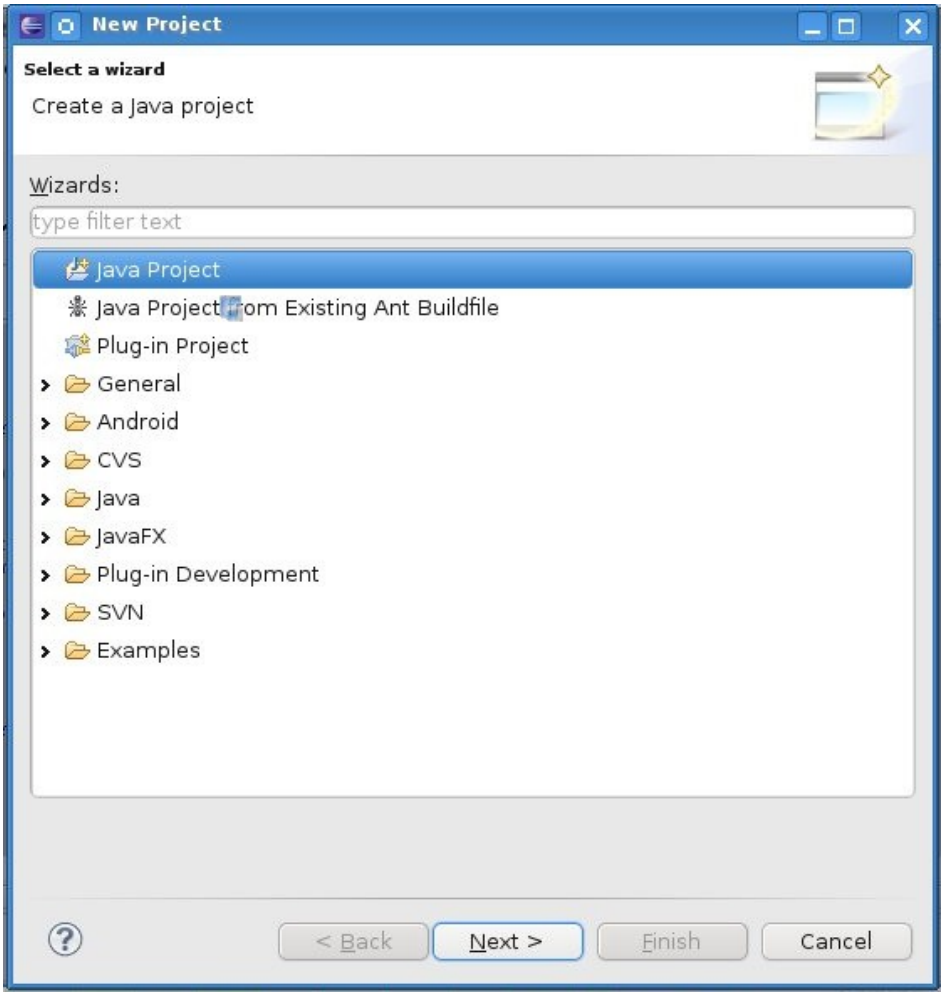

Figura 3.4: Escolhendo um novo projeto pelo *Wizard* do Eclipse

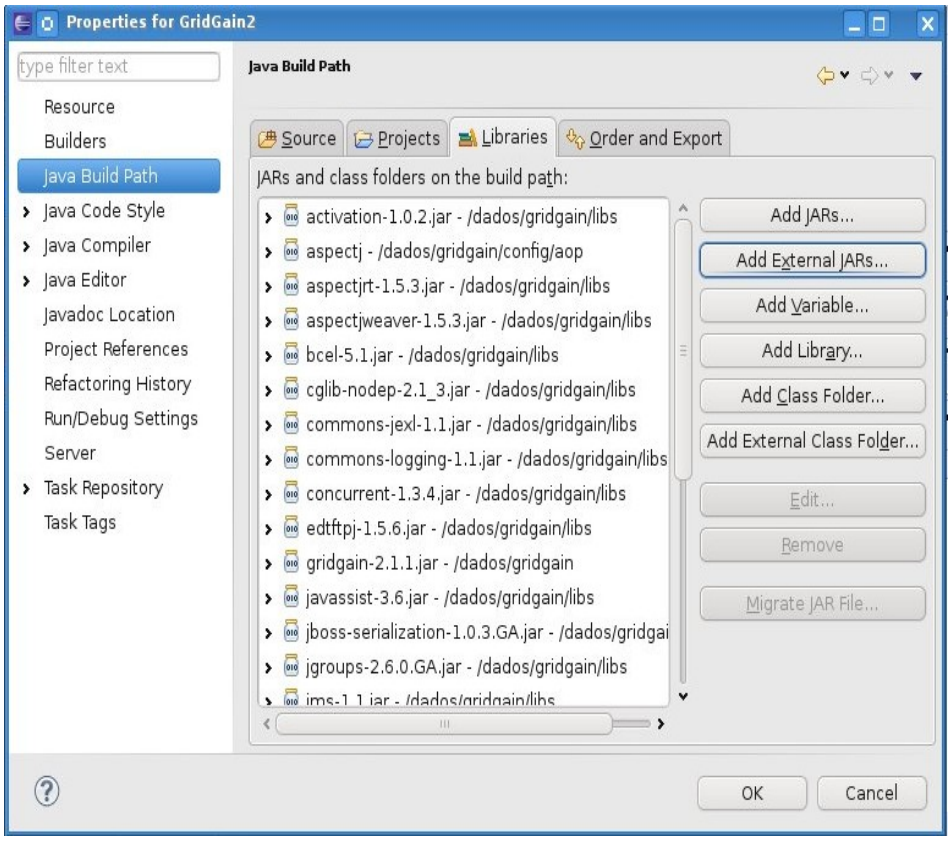

Figura 3.5: Adição das bibliotecas do *GridGain* ao projeto no Eclipse.

Capítulo 4

Resultados

4.1 Criando um ambiente de testes

Para realizar os testes com o *GridGain* foi configurado um ambiente utilizando quatro computadores e uma aplicação que realiza a multiplicação de duas matrizes, uma contendo cinquenta linhas por cem colunas e outra contendo cem linhas por cinquenta colunas.

A figura 4.1 mostra como ficou configurado o ambiente de testes. O computador onde se encontra o aplicativo de testes criado no Eclipse, está utilizando o sistema operacional *Ubuntu* 9.10. Dos três computadores restantes, que apenas executam o *script* do *GridGain*, um utiliza o sistema operacional *Windows* XP, para que se possa avaliar a capacidade do *GridGain* de manter a transparência entre sistemas operacionais diferentes, e os outros dois utilizam a distribuição Linux *Ubuntu* 9.10.

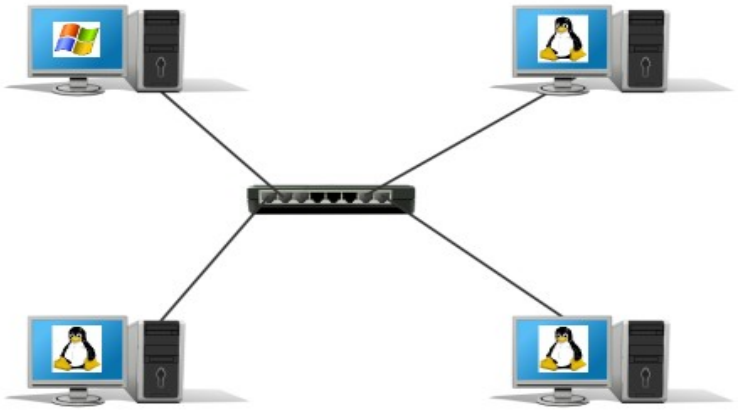

Figura 4.1: Configuração do ambiente de testes.

O *GridGain* trabalha com nós (*nodes*) independentes e já configurados. Desta forma, é possível ter mais de um nó em uma única máquina. Cada nó representa um ponto de processamento paralelo, onde é executado um número determinado de *jobs*, gerenciado pelo *GridGain*. Para criar um nó, basta executar o *script* "gridgain.sh" presente em "/opt/gridgain/bin", como mostra a figura 4.2.

```
GridGain Command Line Loader, ver. 2.1.1
Copyright (C) 2005-2009 GridGain Systems.

[15:56:42,181][INFO ][main][GridKernal%null]
  _____       _    _ _____       _
 / ____|     (_)  | |/ ____|     (_)
| |  __ _ __ _  __| | |  __  __ _ _ _ __
| | |_ | '__| |/ _` | | |_ |/ _` | | '_ \
| |__| | |  | | (_| | |__| | (_| | | | | |
 \_____|_|  |_|\__,_|\_____|\__,_|_|_| |_|

>>> GridGain ver. 2.1.1-26022009
>>> Copyright (C) 2005-2009 GridGain Systems.

[15:56:42,200][INFO ][main][GridKernal%null] GRIDGAIN_HOME=/dados/gridgain
[15:56:42,207][INFO ][main][GridKernal%null] VM arguments: [-ea,
-XX:MaxPermSize=128m, -XX:+UseParNewGC, -XX:MaxNewSize=32m, -XX:NewSize=32m,
-Xms256m, -Xmx256m, -XX:SurvivorRatio=128, -XX:MaxTenuringThreshold=0, -XX:+UseTLAB,
-XX:+UseConcMarkSweepGC, -XX:+CMSClassUnloadingEnabled,
-Dcom.sun.management.jmxremote, -DGRIDGAIN_HOME=/dados/gridgain,
-DGRIDGAIN_PROG_NAME=./gridgain.sh, -DGRIDGAIN_HOME=/dados/gridgain]
[15:56:42,306][INFO ][main][GridKernal%null] License file can be found at:
/dados/gridgain/license.txt
[15:56:42,307][INFO ][main][GridKernal%null] 3-rd party licenses can be found at:
/dados/gridgain/libs/licenses
[15:56:42,376][INFO ][main][XmlBeanDefinitionReader] Loading XML bean definitions
from resource loaded from byte array
[15:56:42,597][INFO ][main][GridKernal%null] User class loader version: 0
[15:56:42,598][INFO ][main][GridKernal%null] Local node ID: 358591fb-acc9-4bba-823d-
464510c6e1dd
[15:56:42,598][INFO ][main][GridKernal%null] P2P exclude path: []
[15:56:42,598][INFO ][main][GridKernal%null] Peer class loading enabled: true
```

Figura 4.2: *Script* do *GridGain* sendo iniciado

Toda configuração de *hardware*, *software*, autenticação, segurança e comunicação entre os recursos computacionais está contida neste *script*. Outras configurações adicionais podem ser realizadas na aplicação Java que está sendo processada no ambiente *grid*, através do uso de *annotations* e POA. Ao executar o *script*, o mesmo gera informações úteis como número de nós, número de

CPUs, endereço IP, sistema operacional (inclusive versão do *Kernel*) e versão da JVM. Estas informações locais e remotas são ilustradas na figura 4.3.

```
>>> -------------------
>>> Discovery Snapshot.
>>> -------------------
>>> Number of nodes: 3
>>> Topology hash: 0xFC2E4917
>>> Local: 358591FB-ACC9-4BBA-823D-464510C6E1DD, 192.168.254.2, Linux i386 2.6.31-
20-generic, cristiano, Java(TM) SE Runtime Environment 1.6.0_17-b04
>>> Remote: 114518A7-6324-4E77-9576-CED4D2054111, 192.168.254.3, Linux i386 2.6.22-
14-generic, ubuntu, Java(TM) SE Runtime Environment 1.6.0-b105
>>> Remote: A943321A-BC70-4FC3-9299-04F2E260544D, 192.168.254.4, Windows XP x86 5.1,
Cristiano, Java(TM) SE Runtime Environment 1.6.0_18-b07
>>> Total number of CPUs: 4

[15:56:47,172][INFO ][main][GridKernal%null]

>>> ---------------------------------------------------
>>> GridGain ver. 2.1.1-26022009 STARTED OK in 5015ms.
>>> ---------------------------------------------------
>>> OS name: Linux 2.6.31-20-generic i386, 2 CPU(s)
>>> OS user: cristiano
>>> VM information: Java(TM) SE Runtime Environment 1.6.0_17-b04 Sun Microsystems
Inc. Java HotSpot(TM) Client VM 14.3-b01
>>> VM name: 4017@cmr
>>> Optional grid name: null
>>> Local node ID: 358591FB-ACC9-4BBA-823D-464510C6E1DD
>>> Local node physical address: 192.168.254.2, eth0
>>> GridGain documentation: http://www.gridgain.org/product.html

[16:07:05,862][INFO ][Thread-2][GridFactory] Invoking shutdown hook...
[16:07:06,068][INFO ][Thread-2][GridMulticastDiscoverySpi] SPI stopped ok.
[16:07:06,069][INFO ][Thread-2][GridBasicTopologySpi] SPI stopped ok.
[16:07:06,069][INFO ][Thread-2][GridFifoQueueCollisionSpi] SPI stopped ok.
[16:07:06,070][INFO ][Thread-2][GridAlwaysFailoverSpi] SPI stopped ok.
[16:07:06,070][INFO ][Thread-2][GridRoundRobinLoadBalancingSpi] SPI stopped ok.
[16:07:06,071][INFO ][Thread-2][GridLocalDeploymentSpi] SPI stopped ok.
[16:07:06,071][INFO ][Thread-2][GridMemoryEventStorageSpi] SPI stopped ok.
[16:07:06,071][INFO ][Thread-2][GridSharedFsCheckpointSpi] SPI stopped ok.
[16:07:06,088][INFO ][Thread-2][GridTcpCommunicationSpi] SPI stopped ok.
[16:07:06,089][INFO ][Thread-2][GridJdkLocalMetricsSpi] SPI stopped ok.
[16:07:06,089][INFO ][Thread-2][GridKernal%null]

>>> ---------------------------------------------
>>> GridGain ver. 2.1.1-26022009 STOPPED OK in 53ms.
>>> ---------------------------------------------
>>> Optional instance name: null
>>> Grid up time: 00:10:18:918
```

Figura 4.3: Informações geradas pelo *Script* do *GridGain*

O código Java utilizado para realizar o teste encontra-se no Apêndice A. Ao executar o *script* “gridgain.sh” em um terminal, este passa ser um nó do ambiente *grid*. Ao executar o aplicativo de testes, o processamento da

multiplicação das duas matrizes é feito em cada um dos nós que estiverem ativos, ou seja, em cada máquina que estiver executando o *script* do *GridGain.*

```
[22:36:47,641][INFO ][main][GridKernal%null]
  _____      _     _ _____       _
 / ____|    (_)   | |/ ____|     (_)
| |  __ _ __ _  __| | |  __ __ _ _ _ __
| | |_ | '__| |/ _` | | |_ |/ _` | | '_ \
| |__| | |  | | (_| | |__| | (_| | | | | |
 \_____|_|  |_|\__,_|\_____|\__,_|_|_| |_|

>>> GridGain ver. 2.1.1-26022009
>>> Copyright (C) 2005-2009 GridGain Systems.

[22:36:47,642][INFO ][main][GridKernal%null] GRIDGAIN_HOME=/opt/gridgain
[22:36:47,642][INFO ][main][GridKernal%null] VM arguments: [-
DGRIDGAIN_HOME=/opt/gridgain, -ea, -javaagent:/opt/gridgain/libs/aspectjweaver-
1.5.3.jar, -Dfile.encoding=UTF-8]
[22:36:47,735][INFO ][main][GridKernal%null] License file can be found at:
/opt/gridgain/license.txt
>>> ---------------------------------------------------
>>> GridGain ver. 2.1.1-26022009 STARTED OK in 6963ms.
>>> ---------------------------------------------------
                                        .......
Número de lihas: 100
Número de Colunas: 50
|
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,
32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50|
                                        .......
Número de lihas: 50
Número de Colunas: 100
|
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,
32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,
60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,
88,89,90,91,92,93,94,95,96,97,98,99,100|
                                       ..........
[18:49:17,651][INFO ][main][GridDeploymentLocalStore] Class locally deployed:
LocalDeploymentClass [undeployed=false, usage=0, super=GridDeploymentClass
[cls=class org.exemplo.GridMatrixMultiplier, depMode=ISOLATED,
clsLdr=sun.misc.Launcher$AppClassLoader@17590db, clsLdrId=7ea9776b-33cf-46f4-b0ef-
5be3a300246c, userVer=0, seqNum=1, alias=org.exemplo.GridMatrixMultiplier,
local=true]]

Número de lihas: 100
Número de Colunas: 100
|
4166275,4167550,4168825,4170100,4171375,4172650,4173925,4175200,4176475,4177750,4179
025,4180300,4181575,4182850,4184125,4185400,4186675,4187950,4189225,4190500,4191775,
4193050,4194325,4195600,4196875,4198150,4199425,4200700,4201975,4203250,4204525,4205
800,4207075,4208350,4209625,4210900,4212175,4213450,4214725,4216000,4217275,4218550,
4219825,4221100,4222375,4223650,4224925,4226200,4227475,4228750,4230025,4231300,4232
575,4233850,4235125,4236400,4237675,4238950,4240225,4241500,4242775,4244050,4245325,
4246600,4247875,4249150,4250425,4251700,4252975,4254250,4255525,4256800,4258075,4259
350,4260625,4261900,4263175,4264450,4265725,4267000,4268275,4269550,4270825,4272100,
4273375,4274650,4275925,4277200,4278475,4279750,4281025,4282300,4283575,4284850,4286
125,4287400,4288675,4289950,4291225,4292500|
.........
==========================
Tempo da tarefa: 9 Minutos
==========================
```

Figura 4.4: Saída resumida no console do Eclipse

A figura 4.4 mostra de forma resumida, como fica a saída no console do Eclipse após a execução da multiplicação de matrizes.

Enquanto os cálculos da matriz são processados em cada nó, o Eclipse fica “aguardando” o término dos mesmos, para apresentar os resultados no console.

A configuração de um nó remoto com *GridGain* em uma rede é extremante simples, bastando para isso instalar o JDK, o *GridGain* e executar seu *script* “gridgain.sh” ou “gridgain.bat”, onde este último é feito para sistemas *Windows*. O aplicativo de testes utilizado pode ser executado em qualquer nó do *grid*. Entretanto, nesta configuração não houve diferenças significativas no tempo de execução, ao rodar o mesmo em nós diferentes.

O tempo obtido em cada nó pode ser visto na tabela 4.1.

Tabela 4.1: Tempos obtidos para o cálculo da multiplicação de matrizes

Número de nós	Tempo gasto em minutos
2	28
3	15
4	9

Executando a mesma aplicação com 1 nó remoto e 1 nó local, o cálculo foi processado em 28 minutos. Incluindo um terceiro nó, o tempo caiu para 15 minutos e com 4 nós o tempo total para o cálculo do processamento ficou em 9 minutos.

É importante ressaltar que o tempo de processamento pode variar bastante, devido a muitos fatores como tráfego na rede, utilização dos processadores de cada computador (já que o *GridGain* utiliza apenas a parte

ociosa), configuração na aplicação que será executada sobre o *grid*, entre outros. O objetivo principal do *GridGain* não é obter alto desempenho, mas utilizar ciclos ociosos de CPU para executar aplicações grandes.

Estes resultados mostram que mesmo em aplicativos simples, executando sobre um *grid*, é possível obter uma redução no tempo de execução ao aumentar o número de processadores utilizados. Pode-se notar também, que o *GridGain* é independente de plataforma, o que resulta em uma alta escalabilidade, pois vários nós podem ser adicionados sem que seja necessário realizar configuração adicional em algum nó que já esteja dentro do ambiente.

Capítulo 5

Conclusão

A computação em *grid* é uma abordagem inovadora que aproveita a infra-estrutura existente para otimizar recursos computacionais e gerenciar dados ou processamentos “pesados”.

Atualmente existem várias ferramentas que buscam prover uma estrutura completa de computação em *grid* como o *Globus Toolkit* (amplamente utilizado), o *gLite*, ARC da *NorduGrid*, entre outros. Neste trabalho, foram apresentadas as principais características e vantagens que a utilização da tecnologia *grid* pode trazer tanto aos usuários comuns, como as grandes organizações. Entretanto, muitas destas soluções ainda apresentam grande complexidade e falta de integração simples com tecnologias amplamente utilizadas, como o JEE, por exemplo.

Visando apresentar a implantação de uma tecnologia *grid* que seja de fácil instalação configuração e integração, este trabalho teve como principal objetivo descrever algumas ferramentas utilizadas e analisar uma destas de forma detalhada. Neste contexto, foi apresentado como é feita a implementação do *framework GridGain* com foco em sua instalação e integração com uma aplicação Java SE, devido ao mesmo ter demonstrado que pode ser integrado a aplicativos de forma simples, e com baixa complexidade em sua instalação, configuração, e utilização.

Os resultados obtidos através dos testes demonstram que mesmo em

aplicativos simples, executando sobre um *grid*, é possível obter um ganho de desempenho. Pode-se notar também, que o *GridGain* é independente de plataforma, o que resulta em uma alta escalabilidade, pois vários nós podem ser adicionados ou retirados sem que seja necessário realizar nenhuma configuração adicional em algum nó que já esteja dentro do ambiente.

Como trabalhos futuros pode-se realizar a integração do *GridGain* com projetos corporativos existentes, a fim de demonstrar o ganho de desempenho com números de nós variáveis. Pode se ainda, utilizando a tecnologia JEE, desenvolver um portal que oferecerá informações sobre os nós, melhorando a interface com o usuário.

Referências Bibliográficas

[1] FOSTER, Ian. KESSELMAN Carl. TUECKE, *Steven. The Anatomy of the Grid: Enabling Scalable Virtual Organizations, Intl J. Supercomputer Applications*, 2001. Documento pdf. Disponível em: http://www.globus.org/alliance/publications/papers/anatomy.pdf Acesso em: setembro 2009.

[2] PITANGUEIRA, S. Rita. ASSIS, M. R. *Semírames*. *Middleware*: Uma solução para o desenvolvimento de aplicações distribuídas. Documento em pdf. Disponível em: http://www.frb.br/ciente/Impressa/Info/I.8.Semiramis.Middleware.pdf. Acesso em: setembro 2009.

[3] STRICKLAND, Jonathan. Como funciona a computação em grade. Disponível em: http://informatica.hsw.uol.com.br/computacao-em-grade.htm. Acesso em: maio 2009.

[4] MORIMOTO, E. Carlos. Computação em Grade - Uma Visão Introdutória. Disponível em: http://www.clubedohardware.com.br/artigos/124/2. Acesso em: junho 2009.

[5] FOSTER, Ian. KESSELMAN, Carl. *The Grid: Blueprint for a new computing infrastructure*. *Elsevier*. 1999. Acesso em: junho 2009.

[6] INRIA M. Christine, ATLANTIQUE B. Rennes: *Service Oriented Infrastructures: from Grids to Clouds & XtreemOS.* Documento em pdf. Disponível em: http://www.xtreemos.eu/xtreemos-events/xtreemos-summer-school-2009/slides/christinemorin-2009-09-07xtreemos-oxford-public.pdf. Acesso em: junho 2009.

[7] *distributed.net History & Timeline.* Disponível em: http://www.distributed.net/history.php. Acesso em: julho 2009.

[8] Tutoriais Banda Larga: ATM: O que é? Disponível em: http://www.teleco.com.br/tutoriais/tutorialatm/pagina_1.asp. Acesso em: julho 2009.

[9] *distributed.net completes* rc5-64 *project*. Disponível em: http://www.distributed.net/pressroom/news-20020926.txt. Acesso em: julho 2009.

[10] *distributed.net*. Disponível em: http://www.distributed.net/. Acesso em: julho 2009.

[11] *Distributed Computing – the SETI@home project.* Disponível em: http://www.ariadne.ac.uk/issue27/seti/ Acesso em: outubro 2009.
[12] BRAVERMAN, M. *Amy. Father of the Grid*. Disponível em: http://magazine.uchicago.edu/0404/features/index.shtml. Acesso em: outubro 2009.

[13] *glite middleware*. Disponível em: http://glite.web.cern.ch/glite/. Acesso em: outubro 2009.

[14] *The Globus Toolkit* 4.0, *Overview of the grid Security Infrastructure*. Disponível em: http://www-unix.globus.org/security/overview.html. Acesso em: outubro 2009.

[15] "O que é SSL?" Disponível em: http://www.comodobr.com/ssl_o_que_e.php. Acesso em: janeiro 2010.

[16] FARGETTA, Marco: *gLite Security.* Disponível em: http://www.euasiagrid.org/training/gLitesecurity.pdf. Acesso em: janeiro 2010.

[17] PEIXOTO, Deyse M. Barros, CARVALHO Leonardo. ALBUQUERQUE, Portes Marcelo. ALBUQUERQUE , Portes Márcio. Instalação e Configuração do *Globus Toolkit* para Computação em *Grid*. Documento pdf. Disponível em: ftp://ftp2.biblioteca.cbpf.br/pub/apub/2003/nt/nt_zip/nt01203.pdf. Acesso em: janeiro 2010.

[18] ANDRADE, Nazareno. Acesso em *grids* computacionais: estado da arte e perspectiva. Documento pdf. Disponível em: http://www2.lsd.ufcg.edu.br/~nazareno/documentos/acessoEmGrids.pdf. Acesso em: março 2010.

[19] Dr. David Anderson describes SETI@home *& BOINC* . Disponível em: http://www.unitedboinc.com/en/video/41-video/135-dr-david-anderson-describes-setihome-a-boinc. Acesso em: março 2010.

[20] CRUZ, Almir. Comparação de ferramentas *Grid* tipo *Desktop Computing*: *Boinc*, *XtremWeb*, e *Condor*. Documento pdf. Disponível em: https://saloon.inf.ufrgs.br/twiki-data/Disciplinas/CMP157/TF08Almir/Boinc_XTremWeb_Condor.pdf Acesso em: março 2010.

[21] *XremWeb : A Generic Global Computing System.* Documento em pdf. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.5844&rep=rep1&type=pdf. Acesso em março 2010.

[22] GEYER, Cláudio. XtremWeb. Disponível em: https://saloon.inf.ufrgs.br/twiki/view/Projetos/Grade/XtremWeb Acesso em: março 2010.

[23] MATTIAS,Ellert. KONSTANTINOV, Aleksandr. KÓNYA, Balázs. SMIRNOVA, Oxana. WAANANEN, Anders. (2003). *"The NorduGrid project: using Globus toolkit for building GRID infrastructure"*. Documento em pdf. Disponível em: http://www.nordugrid.org/slides.html. Acesso em: março 2010.

[24] EEROLA, Paula; et al. (2003). *"Atlas Data-Challenge 1 on NorduGrid". Proceedings of 2003 Conference for Computing in High Energy and Nuclear Physics.* Disponível em: http://arxiv.org/abs/physics/0306013. Acesso em: março 2010.

[25] *ARC 0.4 Release Notes.* Disponível em: http://www.nordugrid.org/arc/release_notes_0_4_3.html. Acesso em: março 2010.

[26] *NorduGrid Web site.* Disponível em: http://www.nordugrid.org/ Acesso em: março 2010.

[27] MORAES M. Arthur. Explorando os mistérios do Universo com o LHC. Documento em pdf: Disponível em: http://www.visaoportal.com.br/arquivos/palestras/seminario_lhc.pdf. Acesso em: abril 2010.

[28] *ARC* 0.6 *Release Notes.* Disponível em: http://www.nordugrid.org/arc/release_notes_0_6.html. Acesso em: abril 2010.

[29] *NorduGrid middleware, the Advanced Resource Connector.* Disponível em: http://www.nordugrid.org/middleware/. Acesso em: abril 2010.

[30] ANJOMSHOAA, Ali; BRISARD, Fred; DRESCHER, Michel; FELLOWS, Donal; LY, An; MCGOUGH, Stephen; PULSIPHER, Darren; SAVVA, Andreas. *Job Submission Description Language (JSDL) Specification, Version* 1.0. Documento em pdf: Disponível em: http://www.gridforum.org/documents/GFD.56.pdf. Acesso em: abril 2010.

[31] *OpenLdap*. Disponível em: http://www.openldap.org/. Acesso em: abril 2010.

[32] *OpenSSL*. Disponível em: www.openssl.org/. Acesso em: abril 2010.

[33] *SASL*. Disponível em: http://asg.andrew.cmu.edu/sasl/. Acesso em: abril 2010.

[34] JSE, JEE e JME: uma breve explicação. Disponível em: http://www.infowester.com/versoesjava.php. Acesso em: abril 2010.

[35] *GigaSpaces Partners with GridGain to Provide Robust grid Computing Solution.* Disponível em: http://www.brm.com/Article.aspx?id=264&p=13&y=2007. Acesso em: abril 2010.

[36] *GridGain Open Cloud Platform.* Disponível em: http://www.gridgain.com/product_features.html. Acesso em: abril 2010.

[37] JUNIOR, S. G. Vicente; WINK, V. Diogo; MACHADO, P. A. Caio. Programação Orientada a Aspectos: Abordando Java e *aspectJ*. Disponível em: http://inf.unisul.br/~ines/workcomp/cd/pdfs/2337.pdf. Acesso em: abril 2010.

[38] *JBoss Community*. Disponível em: http://www.jboss.org/. Acesso em: abril 2010.

[39] Guia do Servidor Conectiva Linux: Alta Disponibilidade. Disponível em: http://www.conectiva.com/doc/livros/online/9.0/servidor/ha.html. Acesso em: abril 2010.

[40] DUARTE, B. M. C. Otto; DURANTE, B. Gabriel. Redes *Peer-to-Peer*. Disponível em: http://www.gta.ufrj.br/grad/04_1/p2p/. Acesso em: abril 2010.

[41] SILVA, J. Osmar. Tutorial *Apache ANT*. Disponível em: http://www.arquivodecodigos.net/arquivo/tutoriais/ant/automacao.php. Acesso em: abril 2010.

[42] Java *Management Extensions* (JMX). Disponível em: http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/. Acesso em: abril 2010.

[43] *Using jconsole.* Disponível em: http://java.sun.com/j2se/1.5.0/docs/guide/management/jconsole.html. Acesso em: abril 2010.

[44] JIN, Hai; DEQING, Zou; HANHUA, Chen; SUN, JianHua; WU, Song. *Fault-tolerant grid architecture and practice.* Documento em pdf. Disponível em: http://portal.acm.org/citation.cfm?id=944148. Acesso em: abril 2010.

[45] GT *Information Services: Monitoring & Discovery System* (MDS). Disponível em: http://www.globus.org/toolkit/mds/. Acesso em: abril 2010.

[46] *A Taste of Computer Security*. Disponível em: http://www.kernelthread.com/publications/security/sandboxing.html. Acesso em: abril 2010.

[47] RUBBO, B. Fernando. *Um estudo do framework de grid GridGain.* Disponível em: https://saloon.inf.ufrgs.br/twiki-data/Disciplinas/CMP157/TF07FernandoRubbo/TF07FernandoRubboArtigo.pdf Acesso em abril 2010.

[48] FILHO, D. F. NEMÉSIO. PEREIRA, R. Marluce. Implantação do middleware *Globus Toolkit* 4 para aplicações em ambientes de Grid. Documento em pdf. Disponível em: http://www.bcc.ufla.br/monografias/2008/Implantacao_do_middleware_Globus_Toolkit_4_para_aplicacoes_em_ambientes_de_grid.pdf. Acesso em maio 2010.

[49] ALEXANDRE, Ricardo; PRATA, Paula; GOMES, Abel. *A Grid Infrastructure for Online Games.* Documento em pdf. Disponível em: http://www.sciencedirect.com/. Acesso em maio 2010.

[50] MATEOS, Cristian; ZUNINO, Alejandro; CAMPO, Marcelo. *An approach for non-intrusively adding malleable fork/join parallelism into ordinary JavaBean compliant applications.* Documento em pdf. Disponível em: http://www.sciencedirect.com/. Acesso em maio 2010.

[51] LEMOS, André; RIBEIRO, S. L. Danilo; GUERREIRO, P. Danilo; PIRES A. Gabriel; PINTO, C. Moisós; BATISTA, L. Vander. A Internet e suas especificidades: Uma ferramenta de segmentação de público. Documento em pdf. Diponível em: http://www.slideshare.net/monzacosta/ainternetesuasespecificidadesumaferramentadesegmentacao. Acesso em Maio 2010.

[52] Eclipse. Disponível em: http://www.eclipse.org. Acesso em: maio de 2010.

[53] *Annotations*. Disponível em: http://java.sun.com/j2se/1.5.0/docs/guide/language/annotations.html. Acesso em: maio de 2010.

Apêndice A

Código Java utilizada no aplicativo de testes

```
package org.exemplo;


import org.gridgain.grid.GridException;
import org.gridgain.grid.GridFactory;
import org.gridgain.grid.gridify.Gridify;

public class GridMatrixMultiplier{


	public static void main(String args[]) throws
GridException{
		GridMatrixMultiplier mm= new GridMatrixMultiplier();
		GridFactory.start();
		long t1 = System.currentTimeMillis();
		try{
			mm.runExample();
			long t2 = System.currentTimeMillis();

	System.out.println("==========================");
	System.out.println("Tempo da tarefa: "+new Date(t2-
t1).getMinutes() + " Minutos");

	System.out.println("==========================");
		}catch(Exception e){
			e.printStackTrace();
		}finally{
			GridFactory.stop(true);
		}
	}



	/*
	 * Preenche as matrizes, realiza a multiplicação e  envia o
resultado para tela
	 */
	public void runExample()
	{
```

```
            int mat1[][] =  populateMatrix(10,5,1);
            int mat2[][] =  populateMatrix(5,10,1);

            excreveMatriz(mat1);
            System.out.println("");
            System.out.println("           X");
            System.out.println("");
            excreveMatriz(mat2);
            System.out.println("");
            System.out.println("           =");
            System.out.println("");
            excreveMatriz(multiplicaMatrizes(mat1, mat2));

      }


      /**
       * Preenche a Matriz
       */
      public int[][] populateMatrix(int rows, int cols, int
starting_counter){
            int mat [][] = new int [rows][cols];
            for ( int i =0 ; i< rows; i++)
            {
                  for(int j=0; j < cols; j++)
                  {
                        mat[i][j]= starting_counter++;
                  }
            }
            return mat;
      }



      /**
       * Mostra as matrizes na tela
       */
      public void excreveMatriz(int mat[][]){
            System.out.println("NÃºmero de lihas: "+mat.length);
            System.out.println("NÃºmero de Colunas:
"+mat[0].length);
            for ( int i =0; i < mat.length; i++)
            {
                  System.out.print("|");
                  for(int j=0; j < mat[0].length; j++)
                  {
                        System.out.print((j < mat[0].length -1)
? mat[i][j]+"," : mat[i][j]+"|");
                  }
                  System.out.println();
            }
      }
```

```
	public int [] getLinData(int mat[][], int row_num){
		int [] rowData = new int [mat[0].length];
		for(int i = 0; i < mat[0].length;  i++)
			rowData[i] = mat[row_num][i];
		return rowData;
	}


	public int[] getColData(int mat[][], int col_num){
		int [] colData = new int[mat.length];
		for(int i=0; i < mat.length ; i++)
			colData[i] = mat[i][col_num];
		return colData;

	}


	public int[][] multiplicaMatrizes(int mat1[][], int mat2[]
[]) {

		if(mat1[0].length != mat2.length){
			//throw new Exception("Number of rows of the
first matrix must match the number of columns of second
matrix.");
		}
		int result[][]= new int [mat1.length]
[mat2[0].length];
		int mat1_row_data [] =mat1[0];
		int mat2_col_data []= new int[mat2.length];
		for(int i=0; i < mat1.length; i++){
			mat1_row_data = getLinData(mat1, i);
			for(int j=0; j < mat2[0].length; j++){
				mat2_col_data = getColData(mat2,j);
				result[i][j] = compute(mat1_row_data,
mat2_col_data);
			}
		}
		return result;
	}


	@Gridify
	public int compute(int arr1[], int arr2[]){

		int result =0;
		for(int i=0; i < arr1.length ; i++){
			result += arr1[i] * arr2[i];
		}
		System.out.println("Calculando:"+result);
		return result;
	}
```

```
	public void printArray(int arr[]){
		System.out.print("[");
		for(int i=0; i < arr.length; i++)
			System.out.print((i < arr.length -1) ? arr[i]
+"," : arr[i]+"]");
		System.out.println();
	}

}
```

Apêndice B

Principais erros que ocorreram durante a implantação do *GridGain*

Ao tentar executar o projeto no eclipse, o seguinte erro é obtido:

```
WARN: Default Spring XML configuration file not found relative to
GRIDGAIN_HOME, will use system defaults (is your GRIDGAIN_HOME
set?): config/default-spring.xml

Exception:
----------
>>> Type: org.gridgain.grid.GridException
>>> Message: Failed to detect GridGain installation home. It was
neither provided in GridConfiguration nor it could be detected
from GRIDGAIN_HOME system property or environmental variable.
>>> Documentation: http://wiki.gridgain.org
>>> Stack trace:
>>>     at
org.gridgain.grid.GridFactory$GridNamedInstance.start(GridFactory
.java:1222)
>>>     at
org.gridgain.grid.GridFactory.start0(GridFactory.java:924)
>>>     at
org.gridgain.grid.GridFactory.start(GridFactory.java:661)
>>>     at
org.gridgain.grid.GridFactory.start(GridFactory.java:638)
>>>     at org.grid.br.HelloWord.main(HelloWord.java:17)
WARN: Ignoring attempt to stop grid instance that was either
already stopped or never started: null
```

Este erro acontece porque o *GridGain* não consegue identificar a variável de ambiente “GRIDGAIN_HOME” e com isso, as configurações do *framework*

spring também não são encontradas. Para resolver, basta adicionar o argumento: “-DGRIDGAIN_HOME=/dados/gridgain” na configuração de compilação do Eclipse.

Outros erros comuns estão relacionados a não configuração da variável de ambiente JDK, bem como a não instalação do mesmo.