UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CENTRO TECNOLÓGICO

Grid Computacional
Uma Nova Abordagem para os Sistemas Distribuídos Tradicionais

FLORIANÓPOLIS, 2003.

ALEXANDRE PARRA CARNEIRO DA SILVA
KLAUS KRUG ROCHA

Grid Computacional
Uma Nova Abordagem para os Sistemas Distribuídos Tradicionais

Proposta de trabalho de conclusão do curso de Bacharelado em Ciências da Computação da Universidade Federal de Santa Catarina.

ORIENTADOR: MÁRIO RIBEIRO DANTAS

FLORIANÓPOLIS, 2003

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Ciências da Computação da Universidade Federal de Santa Catarina. Comissão Examinadora formada pelos professores:

Prof. Dr. Mário Antônio Ribeiro Dantas _____
INE / UFSC - Orientador

Prof. Dr. Roberto Willrich _____
INE / UFSC – Banca 1

Prof. Dr. Vitório Bruno Mazzola _____
INE / UFSC – Banca 2

Florianópolis, junho de 2003.

Àqueles (as) que encontramos na encruzilhada da vida e, mais que conhecidos, na convivência e no bem querer, nos tornamos amigos (as) .

Sumário

Sumário 5

Lista de Figuras e Tabelas 8

Lista de Acrônimos 9

Resumo 11

Abstract 12

1. Introdução 13
   1.1 Objetivos 14
   1.2 Justificativa 15
   1.3 Organização do Trabalho 15

2. Sistemas Distribuídos 18
   2.1 Introdução 18
      2.1.1 Compartilhamento de recursos 19
      2.1.2 Concorrência 19
      2.1.3 Transparência 20
      2.1.4 Escalabilidade 21
      2.1.5 Confiabilidade 21
      2.1.6 Desempenho 22
      2.1.7 Flexibilidade 22

3. Sistemas Paralelos 24
   3.1 Introdução 24
   3.2 MIMD com Memória Compartilhada 28
   3.3 MIMD com Memória Distribuída 29

4. Cluster 30

5. Grid Computacional 33
   5.1 O que Grid Computacional pode fazer 35
      5.1.1 Explorando recursos não aproveitados plenamente 35
      5.1.2 Capacidade de Paralelismo da CPU 37
      5.1.3 Aplicações 38
         5.1.3.1 Supercomputação Distribuída 38
         5.1.3.2 Computação de Alta Taxa de Rendimento 39
         5.1.3.3 Computação sob Demanda 39
         5.1.3.4 Computação Orientada a Dados 39
         5.1.3.5 Computação Colaborativa 39
      5.1.4 Colaboração de recursos virtuais e organizações virtuais 40

5.1.5 Acesso aos recursos adicionais 41
5.1.6 Balanceamento de recurso 42
5.1.7 Confiabilidade 44
5.1.8 Gerenciamento 46
5.2 Conceitos de grid e componentes 47
5.2.1 Tipos de Recursos 47
5.2.1.1 Computação 48
5.2.1.2 Armazenamento 48
5.2.1.3 Comunicações 50
5.2.1.4 Softwares e Licenças 51
5.2.1.5 Equipamentos especiais, capacidades, arquiteturas e políticas 51
5.2.1.6 Tarefas e aplicações 52
5.2.1.7 Escalonando, reservando e buscando recursos ociosos 53
5.3 Tipos de Grids 55
5.3.1 Intragrid, Intergrid e Extragrid 57
5.4 Arquitetura do Grid 60
5.5 Construção do grid 62
5.5.1 Desenvolvimento planejado 63
5.5.1.1 Segurança 63
5.5.1.2 Organização 63
5.5.2 Componentes de software do grid 64
5.5.2.1 Componentes de gerência 64
5.5.2.2 Software de Fornecimento 65
5.5.2.3 Software de Submissão 66
5.5.2.4 Gerenciamento no grid distribuído 66
5.5.2.5 Escalonadores 67
5.5.2.6 Comunicações 68
5.5.2.7 Observação, Gerenciamento e Medição 68
5.5.2.8 Autoridade Certificadora 69
6. Ferramenta Globus 71
6.1 Três Pirâmides 72
6.1.1 Gerenciamento de recursos 73
6.1.2 Serviços de Informação 73
6.1.3 Gerenciamento de Dados 73
6.2 Componentes do Globus Toolkit 73
6.2.1 Grid Security Infrastructure (GSI) 75
6.2.2 Grid Resource Allocation Manager (GRAM) 76
6.2.2.1 O comando globusrun 77
6.2.2.2 Resource Specification Language (RSL) 78
6.2.2.3 Gatekeeper 78
6.2.2.4 Gerenciador de Tarefas 78
6.2.2.5 Global Access to Secondary Storage (GASS) 79
6.2.2.6 Dynamically-Updated Request Online Coallocator (DUROC) 79
6.2.3 Monitoring and Discovery Service (MDS) 80
6.2.3.1 Informação do Recurso 81

6.2.3.2 Grid Resource Information Service (GRIS) 82
6.2.3.3 Grid Index Information Service (GIIS) 82
6.2.3.4 Provedores de Informação 82
6.2.3.5 Cliente MDS 82
6.2.3.6 MDS Hierárquico 83
6.2.4 GridFTP 83
6.2.4.1 Protocolo GridFTP 83
6.2.4.2 Cliente e Servidor GridFTP 84
6.2.4.3 Ferramentas do GridFTP 85
7. Ambiente de Desenvolvimento 86
8. Conclusão e Trabalhos Futuros 88
Referências 90
ANEXOS 92

Lista de Figuras

Figura 1 - Taxonomia de Flynn *25*

Figura 2 - Taxonomia de arquiteturas paralelas *26*

Figura 3 - Extensão da classe MIMD da taxonomia de Flynn *27*

Figura 4 – Virtualização dos recursos do grid *41*

Figura 5 – Balanceamento de carga *43*

Figura 6 – Confiabilidade do grid *45*

Figura 7 – Administradores podem ajustar políticas para uma melhor alocação de Recursos *47*

Figura 8 – Data Striping *49*

Figura 9 – Escalonamento de Tarefas *53*

Figura 10 – A Evolução do grid *56*

Figura 11 – Um grid simples *57*

Figura 12 – Uma intergrid mais complexa. *59*

Figura 13 – Arquitetura de um grid *60*

Figura 14 – As três pirâmides *72*

Figura 15 – Uma visão geral do sistema do Globus Toolkit. *74*

Figura 16 – Visão geral do GRAM *77*

Figura 17 – Visão geral do DUROC. *79*

Figura 18 – Visão geral do MDS *81*

Figura 19 – Transferência de arquivo padrão. *84*

Figura 20 - Transferência de arquivo “Third-party”. *84*

Lista de Acrônimos

AFS	Andrew File System
API	Application Program Interface
ATM	Asynchronous Transfer Mode
COTS	Common Off-The-Shelf
CPU	Central Processing Unit
DFS	Distributed File System
DUROC	Dynamically – Updated Request Online
EP	Elemento de Processamento
GASS	Global Access to Secondary Storage
GGF	Global Grid Forum
GPFS	General Parallel File System
GRAM	Grid Resource Allocation Manager
GRIIS	Grid Index Information Service
GSI	Grid Security Infrastructure
HA	High Availabity
HPC	High Performance Computation
IBM	International Business Machines
IETF	Internet Engineering Task Force
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MDS	Monitoring and Discovery Service
MIMD	Multiple Instruction, Multiple Data

MISD	Multiple Instruction, Single Data
NFS	Network File System
OGSA	Open Grid Services
OGSI – WG	Open Grid Service Infrastructure Working Group
PC	Personal Computer
QoS	Quality Of Service
RSL	Resource Specification Language
SDK	Software Development Kit
SDs	Sistemas Distribuídos
SIMD	Single Instruction, Multiple Data
SISD	Single Instruction, Single Data
SMP	Symmetric Multi Processor
SSL	Secure Socket Layer
TI	Tecnologia de Informação
TTL	Time To Live
WAN	Wide Area Network

Resumo

A abordagem de **grid computacional** (também, conhecida como *Grid Computing*) é uma nova forma de computação distribuída considerada uma evolução dos sistemas distribuídos tradicionais. O objetivo desse paradigma é criar a ilusão de um simples, todavia enorme e poderoso computador virtual, auto gerenciável. Ele permite uma enorme coleção de sistemas heterogêneos conectados compartilhando várias combinações de recursos. O *grid computacional* possui importantes capacidades como a exploração de recursos não utilizados plenamente, o paralelismo da CPU, a colaboração de recursos e organizações virtuais e balanceamento de recursos visando um aumento significativo da performance na execução de tarefas paralelas e distribuídas em vários *nós* que integram o ambiente. Devido a sua importância, neste trabalho serão apresentados os conceitos, os principais usos, e as arquiteturas conhecidas dos *grids computacionais*, incluindo a utilização de um ambiente de teste baseado em uma configuração de *grid*. Com este objetivo empregamos o ambiente *globus* porque o mesmo representa uma interessante plataforma aberta para o estudo ora proposto.

Abstract

The boarding of the grid computational (also, known as Grid Computing) is a new form of distributed computation considered an evolution of the traditional distributed systems. The objective of this paradigm is to create the simple illusion of an enormous and powerful virtual computer, itself managed. It allows an enormous collection of heterogeneous systems hardwired sharing some combinations of resources. Grid computational possesss important capacities as the exploration of resources not used fully, the parallelism of the CPU, the contribution of resources and organizations virtual and balancing of resources aiming at a significant increase of the performance in the execution of tasks parallel and distributed in several we who integrate the environment. Had its importance, in this work the concepts, the main uses, and computational the architectures known of grids will be presented, including the use of an environment of test based on a configuration of grid. With this objective we use the environment globus because the same it represents an interesting platform opened for the considered study.

1. Introdução

Desde os primórdios da computação o ser humano deparou com a necessidade de construir computadores com alto poder de processamento para atender as crescentes e variadas formas de tarefas, no intuito de reduzir o tempo de resposta a cada tarefa. A princípio, criaram-se computadores cada vez mais poderosos, ou seja, alto poder de processamento e memória. No entanto, a tecnologia de microeletrônica no passado não era tão avançada como a atual, resultando na construção de imensos computadores de alto desempenho como é o caso dos mainframes da IBM. Apesar de ter atualmente microcomputadores com poder de processamento muito maior do que os mainframes, paralelamente existem pesados aplicativos e tipos de dados a serem processados.

Como uma forma alternativa de acompanhar a crescente demanda por processamento, têm-se utilizado clusters inibindo assim a necessidade indefinida por supercomputadores. O que torna um cluster ser uma boa alternativa é a possibilidade de obter o mesmo poder de processamento de um supercomputador, porém com um custo relativamente muito baixo, ou seja, há uma melhor relação custo beneficio. Como exemplo, tem-se o Avalon que é um cluster de 140 máquinas Alpha 533 MHz, localizado no Laboratório Nacional de Los Alamos, nos Estados Unidos. Esse computador já foi o 113º no TOP500, a lista das 500 máquinas mais rápidas do mundo. O fato de o Avalon ter custado uma fração do preço de seus concorrentes próximos no ranking, mesmo usando máquinas de alta qualidade, é um testemunho da vantagem da solução de processamento pesado usando clusters.

A explosão da Intemet devido a padronização de comunicações entre sistemas heterogêneos propicia a utilização de sistemas distribuídos, podendo ser citados o CORBA, Web Servers, NFS, etc. No entanto, apesar da Internet ser um meio extremamente poderoso para interconexão de máquinas e dispositivos de arquiteturas heterogêneas, neste ambiente praticamente inexiste uma forma eficiente de controle desses recursos, tanto para quem provê os recursos como para quem os utiliza. Além disso, é evidente que os avanços tecnológicos ocorridos nos últimos anos têm tornado mais comum e necessária esta

interconexão, criando um grande potencial de poder computacional agregado. Atualmente, buscam-se formas de melhor aproveitar este poder computacional, tendo em vista que a utilização das redes de computadores nos campos científicos e comerciais é uma realidade, porém grande parte do poder de processamento é sub-utilizado.

Para organizar e controlar as máquinas ligadas à Internet, aproveitando o seu vasto poder computacional heterogêneo, nota-se não ser uma tarefa trivial. Neste contexto surge o paradigma do grid computacional, aproveitando o poder de processamento distribuído que a Internet possibilita.

A emergente padronização para recursos compartilhados junto com o aumento da disponibilidade de largura de banda está possibilitando uma etapa evolucionária possivelmente uniforme no grid computacional. Esta é uma tecnologia de computação distribuída com o objetivo de criar a ilusão de um simples, todavia enorme e poderoso computador virtual autogerenciável, com uma enorme coleção de sistemas heterogêneos conectados compartilhando várias combinações de recursos [1].

1.1 Objetivos

O trabalho consiste no estudo dos conceitos do Grid Computacional mostrando as diferenças e vantagens que essa nova tecnologia distribuída possui com relação aos sistemas distribuídos tradicionais, visando posteriormente, a construção de um grid utilizando software específico para desenvolvê-lo, o Globus Toolkit.

Haverá o estudo desta ferramenta para construção de um grid composto por recursos heterogêneos, objetivando um teste para avaliar o desempenho, tempo de execução. Este teste consiste em submeter uma determinada tarefa no grid e observar quanto tempo foi gasto para sua execução. Esta mesma tarefa será submetida a um PC dual, no intuito de mostrar que é possível um ambiente grid ser tão “fácil” e “popular” como um PC, tendo em vista que este conceito, no campo prático, é recente e está caminhando para provar que é possível concretizá-lo em sua totalidade.

1.2 Justificativa

A motivação para este trabalho é devido à necessidade de alto desempenho computacional e poder de processamento nas diversas áreas que demandem estas características. O mercado de hardware, para suprir essa necessidade, vem desenvolvendo computadores com alto poder de processamento e desempenho, porém com um custo extremamente elevado. Com o Grid Computacional, consegue-se o mesmo desempenho com um custo muito inferior.

Um outro ponto importante a ser considerado é relativo a Lei de Moore [2], que afirma que o número de transistores disponíveis para construir ou ocupar um circuito integrado baseado em silício dobrará a cada dois anos. Para alcançar esse crescimento exponencial em densidade de transistores é necessário uma constante miniaturização do tamanho do transistor. Até que ponto será possível construir ou ocupar um circuito integrado para se obter um bom desempenho dos dispositivos baseados em silício construídos com transistores menores?

Outra motivação vem do interesse pelas diversas áreas de conhecimento que o grid computacional agrega, tais como redes de alto desempenho, escalonamento de tarefas, processamento paralelo e sistemas distribuídos. O interesse é motivado pela inevitável utilização dessa tecnologia no futuro, o que vem ocorrendo atualmente de forma incipiente.

1.3 Organização do Trabalho

Este trabalho está organizado em sete capítulos principais:

- O primeiro engloba a introdução. Neste capítulo, dar-se-á uma visão geral do que vem a ser Grid Computacional, apresentando os motivos que levaram ao estudo desse ambiente bem como o seu objetivo;

- No segundo, terceiro e quarto capítulo é feita a revisão bibliográfica sobre Sistemas Distribuídos, Sistemas Paralelos e Cluster, respectivamente. O objetivo é fornecer embasamento teórico para auxiliar no entendimento dos conceitos do Grid Computacional;

- Os conceitos sobre o paradigma de Grid Computacional, quinto capítulo, enfatiza os seguintes tópicos:
    - Razões do uso do Grid Computacional, bem como suas limitações;
    - As classes de aplicações que se beneficiariam com o ambiente grid;
    - A evolução do Grid Computacional;
    - Conceitos e seus componentes principais;
    - Os tipos de grid;
    - A arquitetura básica do grid.

- No sexto, será abordada a estrutura básica do Globus Toolkit citando os principais componentes que o integram. Além disso, será mostrado o motivo levado a escolha desta ferramenta dentre várias outras para construir um grid;

- No sétimo capítulo será apresentado o resumo do desenvolvimento do trabalho proposto, verificando se os objetivos iniciais foram atingidos e, principalmente, relacionando as dificuldades encontradas.

Os anexos e as referências estão disponibilizados nos capítulos finais do trabalho. O primeiro anexo apresenta vários projetos de ferramentas de grid em andamento, fomecendo uma breve descrição de cada uma, além da classificação dos atributos do sistema de gerenciamento de recurso empregado, de acordo com a taxonomia em [3]. O segundo anexo demonstra as vantagens e desvantagens entre Grids e SDs tradicionais. O terceiro anexo descreve os passos para a criação de um grid simples desde a instalação do Globus Toolkit às principais configurações dos seus componentes básicos.

Por último, a referência bibliográfica cita obras que podem ser pesquisadas para um melhor detalhamento dos temas abordados em várias partes do trabalho.

2. Sistemas Distribuídos

2.1 Introdução

A princípio, o surgimento dos sistemas computacionais distribuídos deu-se, basicamente, devido à necessidade de compartilhamento de recursos, normalmente de alto custo e fisicamente separados. Além disso, notou-se que a difusão da Tecnologia da Informação (TI), na década de 80, como instrumento de transformação dos processos de negócios das empresas não culminou, como era esperado, em um aumento significativo nos níveis de produtividade com relação aos investimentos de TI. Nesta época atribuiu-se a baixa produtividade devido os profissionais trabalharem individualmente.

A tecnologia de sistemas distribuídos começou a ser implementada a partir do início da década de 90 devido à necessidade da interligação de recursos, se tornando uma alternativa para lidar com a baixa produtividade e com os altos custos dos recursos. Somando-se a essa necessidade houve na época o surgimento das redes Locais, que é a essência dos Sistemas Distribuídos (SDs) ou da Computação Distribuída, junto com a evolução da tecnologia de redes, como por exemplo, os repetidores, pontes, roteadores e gateways. Não há consenso na literatura com relação ao conceito de Sistemas Distribuídos [4]. Neste trabalho seguir-se-á o conceito apresentado em [4] que define sistema distribuído como "um conjunto de elementos, elementos de processamento (EP), que se comunicam através de uma rede de interconexão e que utilizam software de sistema distribuído". A seguir têm-se as principais características dos sistemas distribuídos:

- Compartilhamento de recursos
- Concorrência
- Transparência
- Escalabilidade

- Confiabilidade
- Desempenho
- Flexibilidade

2.1.1 Compartilhamento de recursos

Com o compartilhamento de recursos há possibilidade de diminuir os custos, principalmente em se tratando de recursos mais caros, e aumentar a segurança, como por exemplo o acesso a dados sigilosos que estão em um servidor de arquivos e que possui políticas de seguranças rígidas. Pode-se citar dois modelos de compartilhamento de recursos:

- *Modelo Cliente Servidor*: cada tipo de recurso é gerenciado por um servidor; clientes, podendo ser processos, enviam requisições a servidores, também podem ser processos, quando requerem um serviço. Alguns processos são clientes e servidores ao mesmo tempo. Este modelo é muito utilizado e adequado para a maioria das aplicações, por exemplo, em correio eletrônico, arquivos de disco, impressoras, impressoras laser, display, scanners, etc. Com isso, permite que usuários tenham acesso a recursos muito caros como, por exemplo, uma impressora laser;
- *Modelo baseado em objetos*: Cada recurso é visto como um objeto, cada objeto é unicamente identificado. Este modelo é mais geral do que o modelo cliente-servidor.

2.1.2 Concorrência

O compartilhamento de recursos pode ser feito em paralelo à medida que os servidores executam concorrentemente, cada um respondendo para diferentes clientes. O acesso concorrente aos recursos tem que ser feito cuidadosamente, pois senão pode haver problemas como, por exemplo, o de consistência de dados numa base de dados.

2.1.3 Transparência

Esta característica permite que os usuários ou programadores não precisem levar em conta a separação de componentes em um sistema distribuído, visto que a transparência traz a sensação de parecer não existir, contudo na verdade existe. Existem os seguintes tipos de transparências:

- *Transparência de Acesso*: Ocultação do uso de comunicação para acessar recursos remotos. Exemplo, clicar no ícone;
- *Transparência de Localização*: Usuários não sabem e nem precisam saber a localização dos recursos;
- *Transparência de Concorrência*: Usuários não estão cientes da existência de acesso simultâneo a recursos remotos, no entanto eles podem compartilhar um recurso automaticamente;
- *Transparência de Replicação*: Múltiplas instâncias de recursos são usadas, mas o SD oculta qualquer diferença entre recursos replicados, ou não replicados. Além disso, os usuários podem se mover sem alterar seus nomes;
- *Transparência de Migração*: Potencial para realocar recursos dinamicamente sem que os usuários estejam cientes do movimento de recursos. Além disso, os recursos podem se mover sem alterar seus nomes;
- *Transparência de Paralelismo*: Atividades podem ocorrer em paralelo sem que o usuário saiba;
- *Transparência de Falha*: Sistema Distribuído oculta os efeitos de falhas parciais;
- *Transparência de Performance*: SD é configurado para melhorar a performance na medida que a carga varia sem o usuário perceber a configuração;
- *Transparência de Escala*: Oculta a expansão sem mudar a estrutura do sistema.

2.1.4 Escalabilidade

É a capacidade que um SD apresenta de poder adaptar-se facilmente a uma carga crescente de recursos homogêneos e serviços. Os sistemas distribuídos precisariam se adaptar a possibilidade de se ter ambientes com milhares de processadores. Para isso deve-se evitar:

1. Componentes centralizados
2. Tabelas centralizadas
3. Algoritmos centralizados

Para ambientes distribuídos deve-se usar algoritmos descentralizados que possuam as seguintes características:

- Nenhuma máquina deve centralizar informações sobre o estado do sistema;
- As máquinas devem tomar decisões baseadas apenas nas informações disponíveis localmente;
- Se uma das máquinas falhar não irá impedir o funcionamento do algoritmo;
- Não há suposições implícitas que exista um relógio global.

2.1.5 Confiabilidade

Os Sistemas Distribuídos devem possuir a capacidade, pelo menos em tese, de que ao parar de funcionar uma máquina outra deva assumir os serviços realizados pela mesma. Na prática, existem alguns servidores, em diferentes máquinas, que precisam estar no ar para que o sistema como um todo funcione.

Alguns aspectos relacionados à confiabilidade devem ser levados em consideração, os mesmos são mostrados a seguir:

- *Disponibilidade*: Refere-se a fração de tempo em que o sistema está funcionando, com isso, pode-se melhorar a disponibilidade através de um projeto que não exija o funcionamento simultâneo de um número substancial de componentes críticos; ou através de redundância, isto é, componentes chaves de hardware e software devem ser replicados, de modo que se um deles falhar, os outros estarão aptos a tomar conta da tarefa;

- *Tolerância a Falhas*: Os sistemas distribuídos são projetados com a capacidade de mascarar falhas, isto é, não permitem que o usuário perceba o que está acontecendo. Um exemplo clássico é o de serviço de arquivos, que pode ser construído com um grupo de servidores cooperantes, de tal forma que o usuário não perceba a falta de um ou mais servidores, apenas "estranhe" o desempenho.

2.1.6 Desempenho

Compartilhamento de recursos com uma melhor utilização da carga de processamento entre todas as máquinas.

2.1.7 Flexibilidade

O sistema deve ser flexível para permitir mudanças no sistema, caso necessite acomodar um novo projeto.

Vantagens

- *Econômica*: aproveita máquinas de baixo custo e poder de processamento; é mais barato ter vários processadores interconectados do que ter um supercomputador;

- *Velocidade*: a capacidade de interconectar processadores possibilita que se obtenha performances que apenas um sistema composto é capaz de atingir;

- *Distribuição inerente*: algumas distribuições são distribuídas por natureza. Ex: aplicação para uma cadeia de supermercados, alguns tipos de jogos, etc;

- *Tolerância a falhas*: quando uma máquina falha, o sistema como um todo pode continuar funcionando, apenas apresentando uma diminuição no seu desempenho;

- *Crescimento Incremental*: pode-se aumentar o poder computacional através da inclusão de novos equipamentos;

- *Flexibilidade*: os sistemas distribuídos são mais flexíveis do que máquinas isoladas, por essa razão são muitas vezes utilizados até mesmo sem a necessidade de maior desempenho. Esta flexibilidade permite que vários usuários compartilhem dados e periféricos [5] [6].

Desvantagens

- Poucos softwares de alto nível disponíveis para sistemas distribuídos;

- Dificuldades para evitar acesso indevido - questões de segurança;

- A rede de interconexão pode causar problemas ou não dar a vazão exigida pela demanda [5] [6].

3. Sistemas Paralelos

3.1 Introdução

O surgimento dos sistemas paralelos veio com o objetivo de reduzir o tempo de processamento de aplicações específicas que demandam altas taxas de processamento. Segundo [4], sistemas paralelos pode ser conceituado como uma forma restrita de sistemas distribuídos, onde o objetivo de todo o sistema é resolver um único problema no menor tempo possível, buscando, desta forma, otimizar a performance. Sistemas Distribuídos, então, são mais genéricos, admitindo várias outras formas de otimização.

As características principais dos sistemas paralelos são:

- Sistemas multiprocessados com um ou mais processadores em comunicação próxima;
- Sistemas fortemente acoplados (*Tightly coupled system*) – processadores compartilham memória e o clock; comunicação normalmente ocorre através de memória compartilhada;
- Sistemas que executam vários programas que compartilham os mesmos dados.

Com o avanço da computação paralela, foram propostas diferentes maneiras de se conectar os recursos computacionais, criando-se diferentes arquiteturas paralelas. Cada arquitetura apresenta determinadas características, sempre visando melhores desempenhos. Para acompanhar o desenvolvimento das arquiteturas paralelas e agrupar equipamentos com características comuns, foram propostas algumas taxonomias, dentre elas a de Flynn[7] - 1972. A taxonomia introduzida por Flynn é ainda a forma mais comum de se classificar sistemas com capacidade de processamento paralelo. Na classificação de Flynn são considerados os fluxos de instruções e de dados criando quatro classes de sistemas computacionais:

1) *Single Instruction, Single Data (SISD) Stream* - Um único processador que executa uma única seqüência de instruções que operam sobre dados armazenados numa única memória, por um computador seqüencial simples, por exemplo, ou monoprocessador;

2) *Single Instruction, Multiple Data (SIMD) Stream* - Uma única instrução do processador controla a execução simultânea de vários elementos processadores, de forma sincronizada denotada *lockstep*. Cada elemento processador tem uma memória de dados associada, tal que uma mesma operação seja executada sobre dados diferentes por processadores distintos. Os processadores matriciais "*array processors*" são exemplos dessa categoria;

3) *Multiple Instruction, Single Data (MISD) Stream* - Um único dado é transmitido a um conjunto de processadores, cada um dos quais executando diferentes instruções. Essa categoria se aproxima de uma arquitetura denominada de array sistólico;

4) *Multiple Instruction, Multiple Data (MIMD) Stream* - Um conjunto de processadores executa simultaneamente diferentes seqüências de instruções sobre diferentes dados, ou seja, cada processador enxerga apenas o seu espaço de memória. Os multiprocessadores podem ser inclusos nessa categoria.

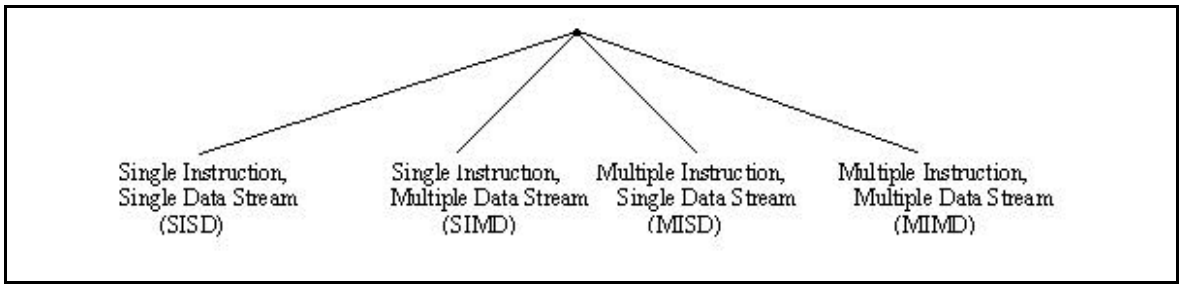

Figura 1 - Taxonomia de Flynn [7].

Esta classificação, apesar de muito adotada, não é muito específica, uma vez que classifica todos os sistemas distribuídos como MIMD. Em virtude disto, neste trabalho, será utilizada a proposta apresentada em [8] para estender a classe MIMD da taxonomia de Flynn. Na Figura 2 observa-se a taxonomia de arquiteturas paralelas a qual representa, de forma completa, a extensão da taxonomia de Flynn.

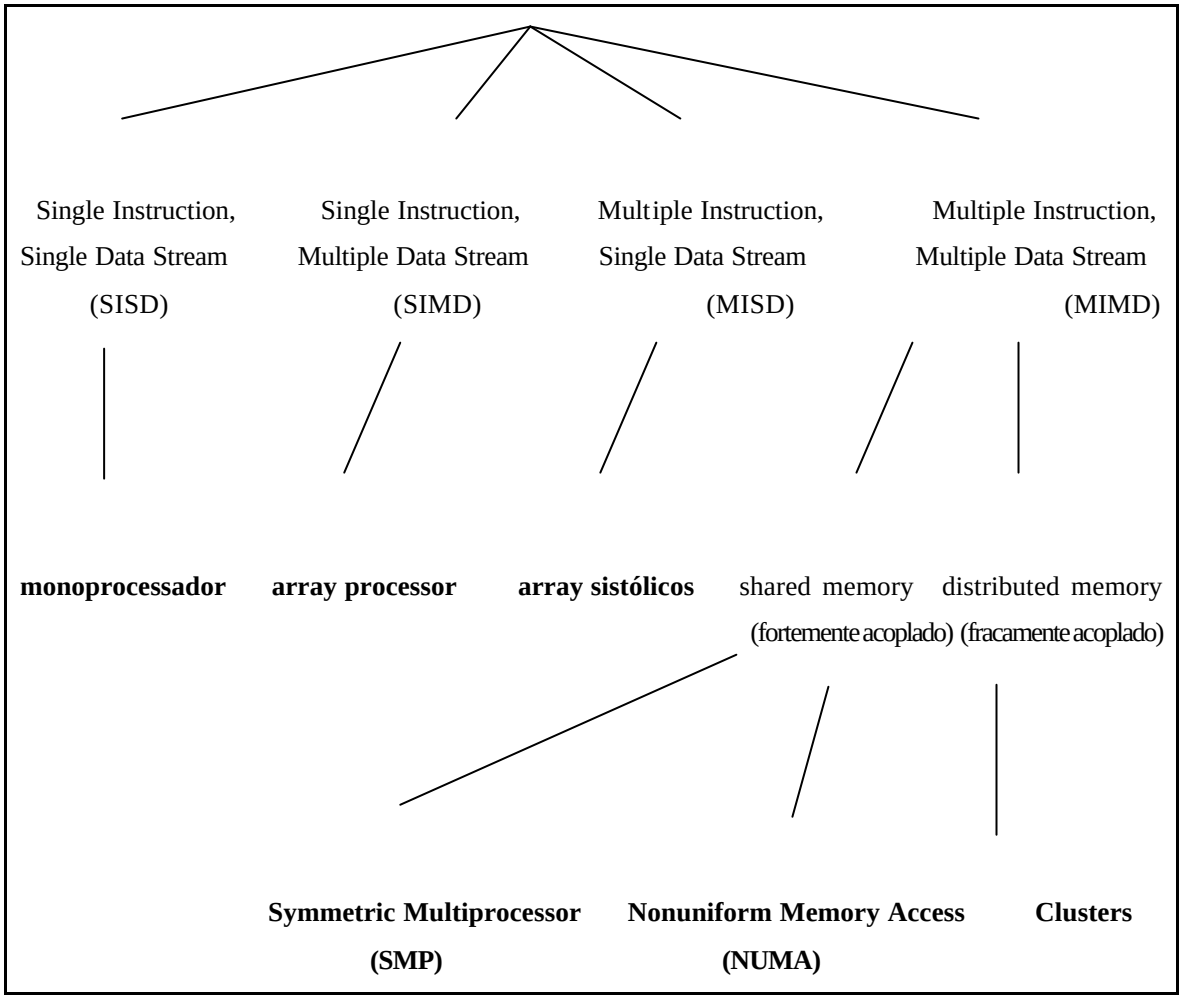

Figura 2 - Taxonomia de arquiteturas paralelas [8].

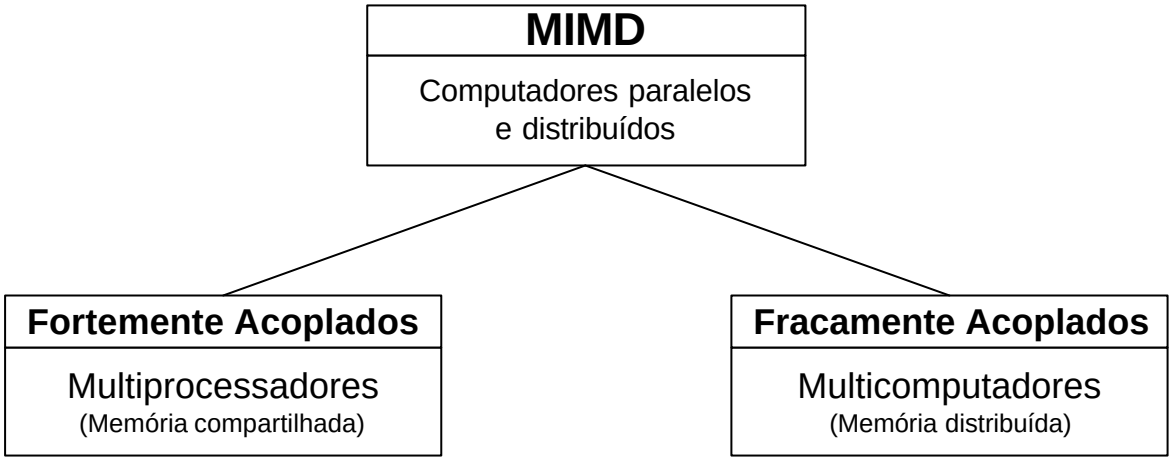

Figura 3 - Extensão da classe MIMD da taxonomia de Flynn [8].

Qualquer que seja a arquitetura MIMD empregada, uma necessidade comum é a de comunicação entre processos que executam em elementos de processamento diferentes. A forma pela qual esta comunicação pode ser realizada está diretamente relacionada à existência ou não de memória física compartilhada, principal característica utilizada na diferenciação entre arquiteturas fortemente e fracamente acopladas.

Com o avanço tecnológico foi possível a união da computação paralela e dos sistemas computacionais distribuídos, surgindo assim o que constitui a computação paralela distribuída. Nesse caso, tem-se a computação paralela sendo implementada sobre uma plataforma MIMD com memória distribuída, podendo explorar características atrativas de ambas as áreas. A seguir serão mencionadas as principais vantagens e desvantagens das duas categorias pertencentes a MIMD, Figura 3.

3.2 MIMD com Memória Compartilhada

Vantagens

- Não há necessidade de particionar o código ou dados, logo técnicas de programação para máquinas com processadores únicos podem ser facilmente adaptados para ambientes multiprocessadores;

- Não há necessidade de movimentar fisicamente os dados quando dois ou mais processadores se comunicam. Como resultado, a comunicação entre processos é bastante eficiente.

Desvantagens

- Há necessidade do uso de primitivas especiais de sincronização quando do acesso a regiões compartilhadas na memória;

- Falta de escalabilidade devido ao problema de contenção de memória. Depois de um determinado número de processadores, a adição de mais processadores não aumenta o desempenho.

3.3 MIMD com Memória Distribuída

Vantagens

- Altamente escalável e permitem a construção de processadores maciçamente paralelos;

- A troca de mensagens resolve tanto o problema da comunicação como o da sincronização.

Desvantagens

- Necessidade de fazer uma boa distribuição de carga entre os processadores quer seja automaticamente, quer seja manualmente;

- É necessário evitar as situações de "deadlock" no nível de aplicação;

- Modelo de programação menos natural.

A arquitetura do ambiente que será objeto de estudo neste trabalho é da classe MIMD, do tipo fracamente acoplado. Nesta categoria se enquadram todos os tipos de sistemas distribuídos, nos quais os EPs são ligados por redes locais (*Local Area Network*, LAN) ou por redes de longa distância (*Wide Area Network*, WAN). Dentro desta arquitetura, dois tipos de sistemas distribuídos se destacam: **cluster** e **grid**. Estes dois sistemas distribuídos serão apresentados de forma detalhada a seguir.

4. Cluster

Fisicamente um cluster nada mais é do que um agrupamento de computadores, ou seja, dois ou mais computadores interconectados por um meio. Geralmente esse meio é uma tecnologia de rede consagrada, como Ethernet (e suas variantes, Fast Ethernet, Giga Ethernet), ATM, Myrinet, Quadrics, Infinband, Rosettanet, Dolphin sci [9].

Conceitualmente, cluster é um grupo de máquinas físicas fracamente acopladas capazes de suportar o mesmo conjunto de máquinas virtuais [10]. Segundo a definição apresentada por [11], cluster é um tipo de sistema distribuído cujos EP são computadores "inteiros" (e.g., PC, estação de trabalho ou SMP) e que é visto e usado como se fosse um recurso computacional único, indivisível.

Um cluster típico apresenta as seguintes características [11] [12]:

- Pertence a um domínio administrativo único, e por isso é utilizado para resolver problemas do interesse de uma determinada empresa ou grupo de pesquisa;
- Externamente, é visto como um recurso computacional único. Seus EP não possuem identidade própria, sendo apenas engrenagens de uma entidade maior, o cluster;
- A latência nas comunicações é pequena (da ordem de centenas de µs, ou menor);
- A taxa de transmissão disponível é elevada (a partir de dezenas de Mbps até, atualmente, dezenas de Gbps);
- Pode se beneficiar de protocolos de comunicação mais eficientes entre seus EP, pois a rede de interconexão, pertencente ao mesmo domínio administrativo, é um recurso controlado;

- A segurança na comunicação entre EP diminui de importância, chegando mesmo a ser desnecessária, caso a rede de interconexão (intracluster) seja desacoplada da rede de acesso ao cluster;
- Pode-se ter homogeneidade de hardware e software nos EP, o que facilita a programação.

A forma como cada cluster é construído depende exclusivamente da função que ele irá exercer, ou seja, para cada aplicação temos um tipo diferente de cluster. Temos basicamente dois tipos de cluster que são os clusters de alta disponibilidade (HA – High Availability), e os clusters construídos para executar aplicações com alto desempenho (HPC – High Performance Computation).

Devemos usar um Cluster HPC quando temos tarefas que podem demorar muito tempo para serem resolvidas por microcomputadores convencionais. Um cluster HA deve ser usado quando um sistema computacional não puder apresentar falhas quanto a disponibilidade do sistema.

Vantagens

- *Escalabilidade*: pode crescer muito mais do que um único computador;
- *FailOver*: Pode tolerar defeitos em nodos e continuar a oferecer serviços;
- *Baixo Custo*: Pode ser construído a partir de componentes COTS – Common off-the-shelf;
- *Implementação*: por hardware é mais eficiente, porém pouco adaptável e por software há um menor custo.

Desvantagens

- Pertence a um único domínio administrativo;

- Buscando facilitar a programação das tarefas a serem executadas no cluster os hardwares e softwares empregados são homogêneos;

- Não é possível executar programas convencionais (seriais) com alto desempenho sem nenhuma modificação de código;

- O desempenho total do sistema é diretamente proporcional ao desempenho individual de cada estação, a qualidade do meio de comunicação e ao número de nodos que estarão conectados ao sistema;

- Maior desempenho desejado requer maior espaço físico, mais dinheiro a ser empregado, pessoas qualificadas para fazer manutenções constantes;

- O cluster necessita ficar alocado em lugar seguro, seco e refrigerado, portanto é imprescindível a utilização de equipamento de ar-condicionado dimensionado conforme o calor gerado pelos equipamentos do cluster.

5. Grid Computacional

A palavra *grid*, como verbete da língua inglesa, é utilizada em referência à rede de energia elétrica (geração, transmissão e distribuição). Esta rede é a infraestrutura que possibilita o uso da energia elétrica (o recurso, neste caso) de forma consistente, generalizada, barata e confiável.

Em computação, a palavra é usada para referir-se a sistemas distribuídos que apresentam características semelhantes à da rede de energia. O grid computacional pode ser definido então como uma infraestrutura de hardware e software que permite, de forma consistente, generalizada, barata e confiável, o acesso a recursos computacionais de alto desempenho. A fim de elucidar a definição, seus diferentes aspectos serão abordados.

O papel principal do grid é o de reunir recursos, em grande escala, para atingir determinado fim. Estes recursos, tipicamente, são da ordem de milhares ou mais (possivelmente milhões), dedicados ou não, pertencentes a vários domínios administrativos, distribuídos de forma global e ligados por uma rede de interconexão que apresentará características bem diversas das normalmente empregadas em clusters, por exemplo. Por isto, fala-se em infraestrutura de hardware e software. A primeira para se estabelecer as interconexões necessárias para acesso aos recursos e a segunda para monitorar e controlar os mesmos.

Esta infraestrutura deve permitir o acesso consistente aos recursos, através de serviços padronizados, com interfaces e parâmetros muito bem definidos. O maior desafio para isto é arrumar uma forma de encapsular a heterogeneidade dos recursos sem comprometer o alto desempenho. Sem esta padronização fica muito difícil o desenvolvimento de aplicações.

Generalizar o acesso não implica em dizer que o grid estará difundido por todos os lugares, e nem que os recursos disponíveis poderão ser acessados por todos. Mas significa que, a partir de um ponto qualquer do grid, deve ser possível ter acesso aos serviços disponibilizados, desde que se tenha a permissão adequada.

Permitir acesso barato a recursos de alto desempenho será possível de ser alcançado graças a um compartilhamento amplo e eficiente desses recursos (economia de escala), o que diminuirá as eventuais capacidades ociosas e **dividirá os custos** destes recursos entre os usuários dos mesmos, de forma proporcional ao uso.

O mercado para o grid computacional esta relativamente em seu estágio inicial, porém é necessário inicializar um desenvolvimento relacionado ao grid por várias razões, como a seguir:

- As aplicações emergentes são significativas, advindas do importante aumento do mercado, incluindo energia e óleo, serviços financeiros, governamentais, ciências da vida e manufatura;

- A infraestrutura para suportar estes tipos de aplicações atualmente é ínfima;

- O potencial do tamanho de mercado é substancial, previsto por diversas pesquisas empresariais;

- Compromisso de investimento e desenvolvimento observado nas maiores indústrias computacionais, incluindo Dell, HP, IBM, Microsoft e Sun, é um indicativo de um mercado em crescimento;

- Existe uma grande pressão para empresas do setor tecnológico cortar gastos e aumentar substancialmente a utilização de infraestrutura existente.

À medida que os grids evoluem dos clusters para virtualizar centros de dados empresariais para um campo distribuído geograficamente, a base da rede necessita crescer efetivamente, em escala e performance, para se ajustar às requisições demandadas pelo novo ambiente.

Por fim, ao se falar em acesso confiável faz-se referência à capacidade do grid em negociar e garantir qualidade de serviço (*Quality of Service*, ou QoS) com os diferentes componentes que fazem parte do grid. Entre os diversos parâmetros nos quais as aplicações necessitarão de garantias, podemos citar: taxa de transmissão, latência, jitter, capacidade de processamento e segurança. A seguir será apresentado, de forma mais detalhada, as capacidades do Grid Computacional.

5.1 O que Grid Computacional pode fazer

Quando você desenvolve um grid, ele reunirá uma coleção de requisitos do cliente. Para melhor comparar as capacidades do grid computacional de acordo com esses requisitos, é útil manter em mente as razões do uso do grid computacional. São elas:

5.1.1 Explorando recursos não aproveitados plenamente

O uso mais simples do grid computacional é executar uma aplicação existente em uma máquina diferente. A máquina na qual a aplicação é normalmente executada deve ser raramente ocupada após um pico incomum em atividade. A tarefa em questão pode ser executada em uma máquina ociosa que esteja em outro lugar do grid.

Há ao menos dois pré-requisitos para este cenário. Primeiramente, a aplicação deve ser executada remotamente e sem sobrecarga exagerada. Em segundo, a máquina remota deve encontrar-se em qualquer hardware especial, software, ou fonte de solicitação imposta pela aplicação.

Por exemplo, uma tarefa em lote que gaste uma quantidade significativa de tempo processando uma coleção de dados de entrada para produzir uma coleção de saída talvez seja o uso mais ideal e simples para um grid. Se as quantidades de entrada e saída são enormes, mais planejamento e pensamento devem ser exigidos para usar de modo eficiente o grid para tal tarefa. Geralmente não faria sentido usar remotamente um editor de texto no grid porque haveria provavelmente imensos atrasos e mais pontos potenciais de falhas.

Na maioria das organizações, há uma grande quantidade de recursos computacionais não aproveitados plenamente. A maioria dos desktops são ocupados menos do que 5% do tempo. Em várias organizações, até as máquinas servidoras podem estar relativamente ociosas. Grid Computacional fornece um framework (esqueleto) para a exploração destes recursos não aproveitados plenamente e dessa forma tem a possibilidade de aumentar substancialmente a eficiência do recurso usado.

O processamento dos recursos não é somente os que devem ser aproveitados plenamente. Muitas vezes, as máquinas têm enormes capacidades de disco não utilizado. Grid Computacional, mais especificamente, uma "data grid", pode ser usada para agregar este armazenamento não utilizado dentro de um depósito virtual de dados muito grande, e configurado possivelmente para alcançar aperfeiçoamentos na confiança e na execução de qualquer máquina simples.

Se um trabalho em lote necessitar ler uma grande quantidade de dados, estes dados podem ser automaticamente replicados para vários pontos estratégicos no grid. Desse modo, se o trabalho precisar ser executado em uma máquina remota, os dados já estão lá e não necessitam ser movidos para este ponto. Isto oferece benefícios de execução nítidos. Também, tais cópias dos dados podem ser usadas como backups quando as cópias primárias forem danificadas ou não estiverem disponíveis.

Uma outra função do grid é melhorar a utilização do recurso, equilibrando esta utilização. Uma organização pode ter picos ocasionais inesperados da atividade exigindo mais recursos. Se as aplicações forem permitidas no grid, elas podem ser movidas para máquinas subutilizadas durante tais picos. Na realidade, algumas implementações do grid podem migrar trabalhos parcialmente terminados. Em geral, ele pode fornecer uma maneira consistente de balancear as cargas na imensa união dos recursos.

Isto se aplica a CPU, ao armazenamento e a muitos outros tipos de recursos que podem estar disponíveis no grid. O gerenciamento pode usar um grid para melhorar a visão do uso de padrões em grandes organizações, permitindo um melhor planejamento quanto a atualização de sistemas, aumento da capacidade, ou reservando recursos computacionais não mais necessários.

5.1.2 Capacidade de Paralelismo da CPU

O potencial para a capacidade paralela maciça da CPU é uma das características mais atrativas de um grid. Em acréscimo às necessidades científicas puras, tal poder computacional é conduzido a uma nova evolução em indústrias, tais como o campo biomédico, modelagem financial, exploração de óleo, animação de filmes de cinema e muitos outros.

O atributo comum entre tais usos é que as aplicações têm sido escritas para usar algoritmos que possam ser particionados em partes independentemente executáveis. Uma aplicação do grid de forma a utilizar a CPU intensivamente pode ser concebida como várias tarefas, porém pequenas, cada uma executando em uma máquina diferente no grid. A medida que estas sub-tarefas não necessitem comunicar-se um com os outras, mais "escalável" a aplicação se tornará. Uma aplicação perfeitamente escalável, por exemplo, terminará dez vezes mais rápido se ela usar 10 vezes o número de processadores.

As barreiras existem para aperfeiçoar a escalabilidade. A primeira barreira depende dos algoritmos usados para dividir a aplicação entre muitas CPUs. Se o algoritmo puder ser dividido dentro de um número limitado de partes independentemente executáveis, então este limite tornar-se-á uma barreira de escalabilidade. A segunda barreira aparece se as partes não são completamente independentes; isto pode causar contenção, que pode limitar a escalabilidade. Por exemplo, se todas as sub-tarefas precisarem ler e escrever de um mesmo arquivo ou banco de dados, os limites de acesso do arquivo ou do banco de dados tornar-se-ão o fator limitante na escalabilidade da aplicação. Outras fontes de contenção intertarefa na aplicação paralela no grid incluem latências entre as mensagens de comunicações, capacidades de comunicação na rede, sincronização de protocolos, largura de banda de entrada e saída para dispositivos e dispositivos de armazenamento, e as latências que interferem nas exigências de aplicações em tempo real.

5.1.3 Aplicações

Há muitos fatores a serem considerados em uma aplicação que possa ser executada no grid. Deve-se compreender que nem todas as aplicações podem ser transformadas de tal forma que permitam ser executadas em paralelo no grid e que sejam escaláveis. Além do mais, não há nenhuma ferramenta prática para transformar aplicações arbitrárias fazendo com que elas explorem as potencialidades do paralelismo de um grid. Há várias ferramentas práticas que os programadores de aplicações mais hábeis podem usar para escrever uma aplicação paralela no grid. No entanto, a transformação automática de aplicações arbitrárias em paralelas é uma ciência nova.

Este pode ser um trabalho difícil e freqüentemente requer matemática avançada e talentos de programação, caso seja mesmo possível em uma dada situação. As aplicações intensivas atualmente escritas da nova computação estão sendo projetadas para a execução em paralelo e estas serão facilmente permitidas no grid, caso as novas aplicações não sigam os já emergentes protocolos no grid e padrões.

Nas referências [13] [14] são identificadas cinco classes de aplicações que mais se beneficiariam do ambiente Grid. Contudo, estas classes não são fixas, ou seja, há aplicações que, por suas características, se enquadram em mais de uma classe.

5.1.3.1 Supercomputação Distribuída

Nesta classe, encontram-se aquelas aplicações que, pelo volume de recursos de que necessitam (CPU, memória, disco, etc.), atualmente não podem ser executadas em um único sistema. Estas aplicações utilizariam o grid para agregar os recursos computacionais de que necessitam, tendo por objetivo maximizar sua performance absoluta (não se importando, a princípio, com os custos envolvidos).

Tipicamente, para aplicações desta classe, os recursos envolvidos são muito caros e/ou muito limitados (e.g., supercomputadores).

5.1.3.2 Computação de Alta Taxa de Rendimento

O objetivo das aplicações desta classe é executar o maior número possível de tarefas por período de tempo. Estas tarefas possuem a característica de serem independentes ou fracamente acopladas.

O ambiente de grid permite a estas aplicações a localização de recursos ociosos para a execução destas tarefas, maximizando, desta forma, o emprego dos recursos computacionais disponíveis.

5.1.3.3 Computação sob Demanda

Computação sob demanda permite o acesso a recursos especializados, cuja posse é economicamente não compensadora ou mesmo inviável, e cujo uso é feito de forma não freqüente, por períodos curtos de tempo.

As aplicações desta classe se preocupam em maximizar a razão performance/custo. O uso do grid permite a estas aplicações encontrar e obter acesso aos recursos necessários pelo menor custo.

5.1.3.4 Computação Orientada a Dados

Estas aplicações envolvem a síntese de informações a partir de um volume muito grande de dados que são mantidos em repositórios geograficamente distribuídos. Neste processo, normalmente se faz uso intensivo de processamento (para a síntese) e de comunicações (para a movimentação dos dados).

5.1.3.5 Computação Colaborativa

Tem por objetivo permitir, em níveis cada vez mais acentuados, a interação entre pessoas, e entre estas e recursos computacionais, que estejam geograficamente distribuídos. As aplicações desta classe normalmente fazem uso de espaços virtuais compartilhados para atingirem este fim.

5.1.4 Colaboração de recursos virtuais e organizações virtuais

Uma outra importante contribuição do grid computacional é permitir e simplificar a colaboração entre uma larga audiência. No passado, a computação distribuída prometeu essa colaboração e conseguiu de certa forma. O Grid Computacional utiliza estas potencialidades para um plano de larga audiência, ao oferecer padrões importantes que permitam que muitos sistemas heterogêneos trabalhem juntos para formar a imagem de um grande sistema computacional virtual oferecendo uma variedade de recursos virtuais, como ilustrado na Figura 4. Os usuários do grid podem ser organizados dinamicamente, cada qual com diferentes políticas de solicitações. Estas organizações virtuais podem compartilhar seus recursos coletivamente como um enorme grid.

O compartilhamento inicia com dados na forma de arquivos ou base de dados. Um grid de dados "data grid" pode expandir as potencialidades dos dados de várias formas. Primeiramente, arquivos ou base de dados podem perfeitamente atravessar vários sistemas e deste modo obter enormes capacidades do que qualquer sistema simples. Tal medida pode aperfeiçoar a taxa de transferência de dados através do uso de técnicas de "striping". Dados podem ser duplicados por todo o grid para servir como backup e podem ser guardados ou ficarem próximos às máquinas que mais os necessitem, em conjunção com técnicas avançadas de escalonamento.

O compartilhamento não é limitado a arquivos, mas também inclui vários outros recursos, tais como equipamentos, softwares, serviços, licenças, e outros. Estes recursos são virtualizados para dá-los mais uniformidade na interoperabilidade entre os participantes heterogêneos do grid.

Os participantes e os usuários do grid podem ser membros de várias organizações virtuais ou reais. O grid pode ajudar a reforçar as regras de segurança entre elas e a implementar as políticas, as quais podem resolver prioridades para ambos recursos e usuários.

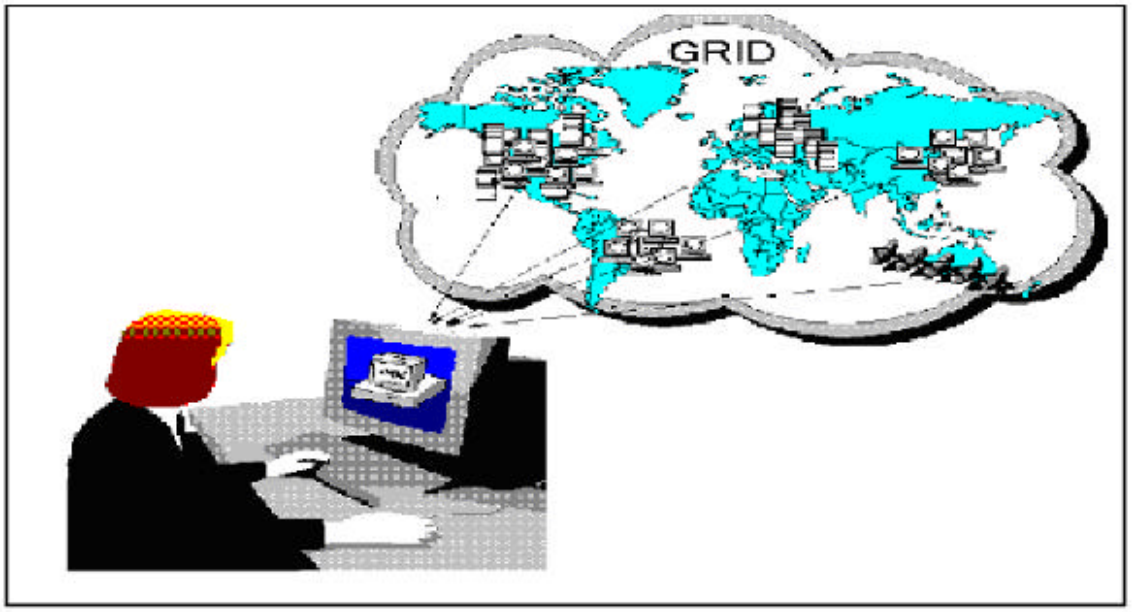

Figura 4 – Virtualização dos recursos do grid [1]

5.1.5 Acesso aos recursos adicionais

Além dos recursos de armazenamento e da CPU, um grid fornece acesso a quantidades elevadas de outros recursos e a equipamentos especiais, softwares, licenças, e outros serviços. Os recursos adicionais podem ser fornecidos em números adicionais e/ou em capacidade.

Por exemplo, se um usuário necessite aumentar o total da sua largura de banda à Internet para executar um mecanismo de busca para minerar dados, o trabalho pode ser dividido entre máquinas no grid que têm conexões independentes à Internet. Desta maneira a potencialidade total de procura é multiplicada, desde que cada máquina tenha uma conexão independente à Internet. Se as máquinas têm conexão compartilhada à Internet, não haveria um aumento eficaz na largura de banda.

Algumas máquinas podem ter um software cuja licença seja cara o qual o usuário requer. Suas tarefas podem ser emitidas a tais máquinas que exploram de forma mais completa as licenças do software.

Algumas máquinas no grid podem ter dispositivos especiais. A maioria dos nós tem usado impressoras remotas, possivelmente com capacidade de cor avançada ou velocidades

mais elevadas. Similarmente, um grid pode ser usado para tirar proveito de outros equipamentos especiais. Algumas máquinas do grid podem ser conectadas a microscópios elétricos de mapeamento para serem operadas remotamente. Nesse caso, o escalonamento e a reserva são importantes. Uma amostra pode ser enviada antecipadamente para facilitar a acomodação no microscópio. Então o usuário pode remotamente operar a máquina, trocando os pontos de visões até que a figura desejada seja capturada.

O grid pode permitir um acesso mais elaborado, potencialmente para diagnóstico médico remoto e ferramentas cirúrgicas robotizadas com interação bi-direcional à distância. As modificações são limitadas somente pela imaginação. Hoje, nós temos drivers de dispositivos remotos para impressoras. Eventualmente, nós veremos padrões de drivers de dispositivos disponíveis no grid para vários dispositivos e recursos incomuns. Tudo isso fará o grid parecer como uma enorme máquina virtual com um conjunto de recursos virtuais superior aos recursos que estão livres em apenas uma máquina convencional.

5.1.6 Balanceamento de recurso

Um grid congrega um grande número de recursos contribuídos por várias máquinas individuais para o interior de um imenso recurso virtual total. Para aplicações que estão ligadas no grid, este pode oferecer um efeito balanceado do recurso através do escalonamento de tarefas no grid em máquinas com baixa utilização, como ilustrado na Figura 5. Esta característica pode fornecer incalculável tratamento de picos ocasionais de cargas de atividade em várias partes de uma organização. Isto pode acontecer de duas maneiras:

- Um pico inesperado pode ser roteado às máquinas relativamente inativas no grid;

- Se o grid já estiver sendo utilizado plenamente, a tarefa de menor prioridade que está sendo executada no grid pode ser temporariamente suspensa ou mesmo cancelada e executada novamente mais tarde para dar espaço para a prioridade mais elevada trabalhar.

Sem uma infraestrutura no grid, tais decisões de balanceamento são difíceis de priorizar e executar.

Ocasionalmente, um projeto pode elevar-se em importância, aumentando o fim do prazo pré-estabelecido da execução da tarefa. Um grid não pode apresentar um milagre e conseguir a modificação do fim do prazo quando já é demasiado próximo. Entretanto, se o tamanho da tarefa é conhecida, se for um tipo de tarefa que pode ser suficientemente dividida em sub-tarefas, e se há recursos disponíveis suficientes após uma tarefa de baixa prioridade adquirir o direito de prioridade, um grid pode trazer uma quantidade muito grande de processamento para resolver o problema. Em tais situações ele pode, com pouco planejamento, substituir o fim do prazo.

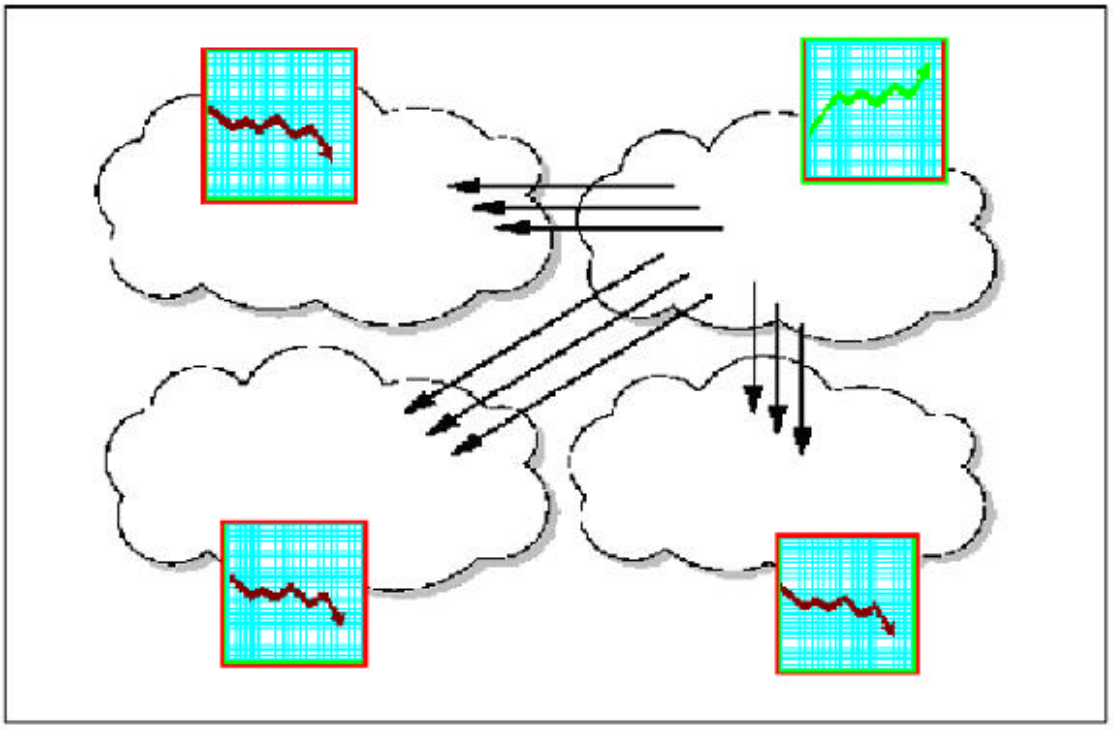

Figura 5 – Balanceamento de carga [1].

Outros benefícios mais sutis podem ocorrer usando o balanceamento de carga. Quando as tarefas se comunicam uma com as outras na Internet, ou com recursos armazenados, um avançado escalonador poderia programá-las para minimizar o tráfego ou

a distância das comunicações. Isto pode reduzir potencialmente uma comunicação e outras formas de contenção no grid.

Finalmente, um grid fornece uma excelente infraestrutura para "brokering resources".

Recursos individuais podem ter os seus perfis traçados para determinar sua disponibilidade e sua potencialidade, e isto pode ser fatorado pelo escalonador do grid. Diferentes organizações participantes podem construir créditos no grid e usá-los quando necessitarem de recursos adicionais. Isto pode formar a base para a contabilidade do grid e a uma capacidade de distribuir a tarefa de modo mais correto.

5.1.7 Confiabilidade

Os sistemas computacionais convencionais de alta capacidade usam hardwares caros para aumentar a confiabilidade. Eles são construídos usando chips com circuitos redundantes que contém muita lógica para conseguir a recuperação de falhas do hardware. As máquinas também usam processadores duplicados. Caso eles falhem, um pode ser trocado sem desligar o outro. Os sistemas que fornecem eletricidade e resfriamento também são duplicados. Os sistemas são operados em fontes especiais de energia que podem iniciar geradores se a energia pública é interrompida. Todos estes sistemas construídos são confiáveis, mas com um custo alto, devido à duplicação de componentes de alta confiabilidade.

No futuro, nós teremos uma outra alternativa para chegar-se à confiabilidade, confiando mais na tecnologia do software do que no hardware caro. Um grid é apenas o princípio de tal tecnologia. Os sistemas no grid podem ser relativamente baratos e geograficamente dispersos. Assim, se houver uma falha de energia ou outro tipo de falha em um lugar específico, as outras partes do grid não serão provavelmente afetadas. O software de gerência do grid pode automaticamente submeter tarefas a outras máquinas quando uma falha é detectada. Em situações críticas, como em tempo real, cópias múltiplas de tarefas importantes podem ser executadas em diferentes máquinas por todo grid, como ilustrado na Figura 6. Seus resultados podem ser checados para verificar qualquer tipo de inconsistência, tal como falhas do computador, dados corrompidos, ou correção de dados.

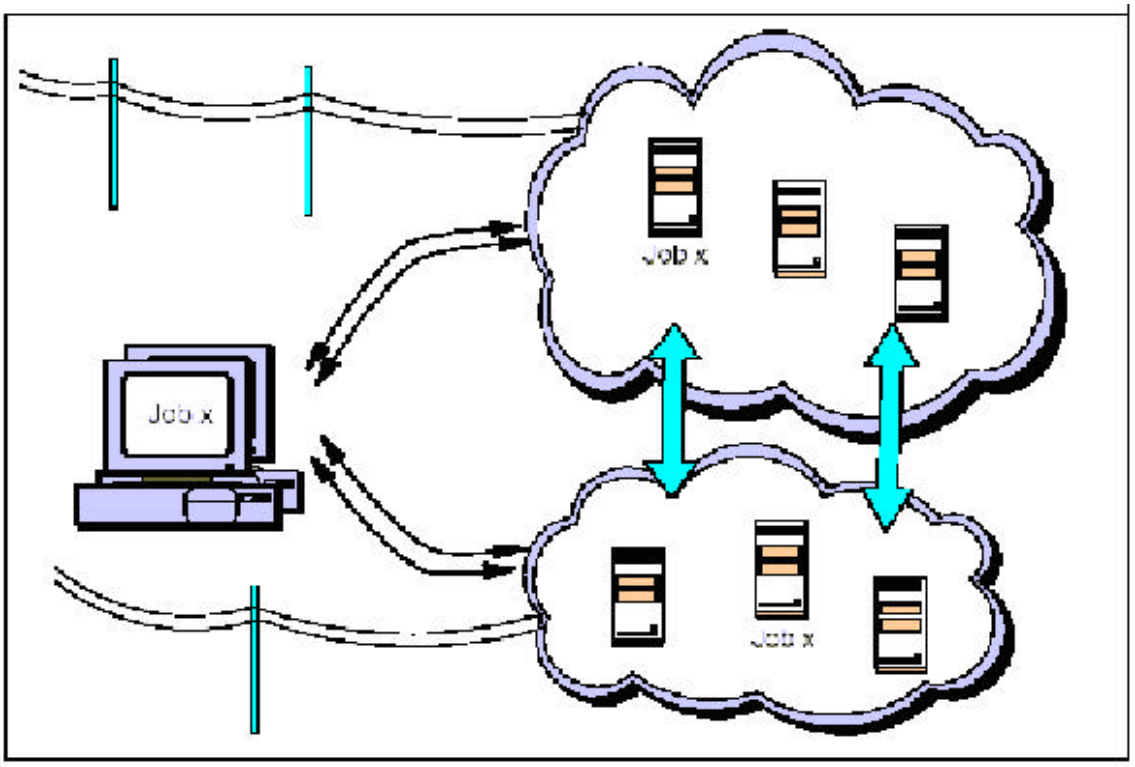

Figura 6 – Confiabilidade do grid [1].

Tais sistemas no grid utilizariam “autonomic computing”. Isto é um tipo de software que automaticamente trata de problemas no grid, talvez até antes mesmo que um operador ou gerente fique ciente deles. Em princípio, a maioria dos atributos de confiabilidade alcançados usando hardware em sistemas atuais de alta disponibilidade pode ser alcançado usando software do grid configurado no futuro.

5.1.8 Gerenciamento

O objetivo de virtualizar os recursos no grid e manipular uniformemente sistemas distribuídos criarão novas oportunidades para melhor gerenciar uma infraestrutura de TI ampla e dispersa. Será mais fácil visualizar a capacidade e utilização, facilitando o controle por parte dos departamentos de TI e de gastos de recursos computacionais em uma grande organização.

O grid oferece gerenciamento de prioridades entre diferentes projetos. No passado, cada projeto era responsável por seus próprios recursos de hardware e pelas despesas relativas a eles. Freqüentemente, este hardware poderia não estar sendo plenamente utilizado enquanto outro projeto poderia estar em dificuldades, necessitando de mais recursos devido à ocorrência de eventos inesperados que aumentem esta necessidade. Com a visão mais ampla que um grid pode oferecer, torna-se fácil controlar e gerenciar tais situações. Como ilustrado na Figura 7, os administradores podem alterar qualquer número de políticas que afetam a forma como as diferentes organizações podem compartilhar ou competir por recursos.

Agregar dados de utilização fornecidos por um grande número de projetos pode melhorar a habilidade das organizações de projetar futuras necessidades de atualização. Quando a manutenção é necessária, o trabalho do grid pode ser redistribuído a outras máquinas sem debilitar os projetos envolvidos.

O conceito de computação autonômica “autonomic computing” também pode estar envolvido neste item. As várias ferramentas podem ser capazes de identificar tendências importantes ao longo do grid, informando a gerência aquelas que requerem atenção.

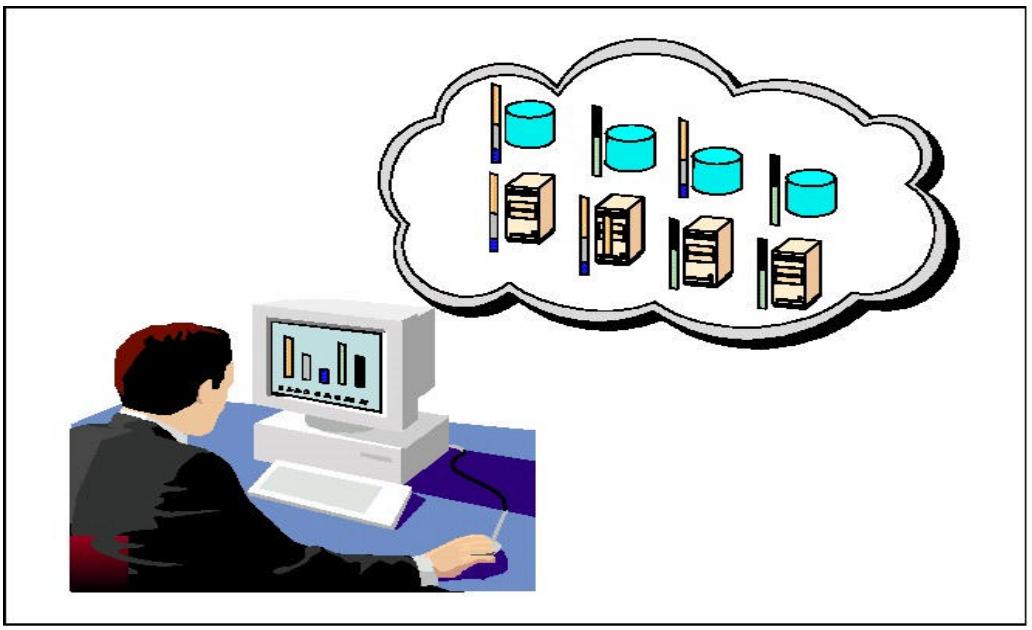

Figura 7 – Administradores podem ajustar políticas para uma melhor alocação de Recursos [1].

5.2 Conceitos de grid e componentes

Nesta seção, nós introduziremos com maiores detalhes os vários conceitos, componentes e termos relacionados as grids.

5.2.1 Tipos de Recursos

Um grid é uma coleção de máquinas, algumas vezes referenciadas como “nodos”, “recursos”, “membros”, “doadores”, “hosts”, “mecanismos” e muitos outros termos. Todos eles contribuem para a combinação de recursos no grid como um todo. Alguns recursos podem ser usados por todos os usuários do grid enquanto outros podem ter restrições específicas.

5.2.1.1 Computação

O recurso mais comum é o ciclo de computação proporcionado pelos processadores das máquinas do grid. Os processadores podem variar em velocidade, arquitetura, plataforma de software e outros fatores associados, como memória, armazenamento e conectividade. Existem três meios primários de explorar os recursos computacionais de um grid. O primeiro e mais simples é usá-los para executar uma aplicação existente em uma máquina existente no grid em vez de executá-la localmente. A segunda é utilizar uma aplicação desenvolvida para distribuir seu trabalho de tal forma que as diferentes partes podem ser executadas paralelamente em diferentes processadores. A terceira é rodar uma aplicação que deve ser executada muitas vezes em diferentes máquinas do grid. Escalabilidade é uma medida de quão eficientemente são usados os processadores do grid. Se o dobro de processadores faz com que uma aplicação seja completada na metade do tempo, então podemos dizer que ela é perfeitamente escalável. Porém, podem existir limites na escalabilidade quando as aplicações podem ser divididas em um número limitado de partes ou se estas partes proporcionam algum outro tipo de contenção de recursos.

5.2.1.2 Armazenamento

O segundo recurso mais comumente usado em um grid é o armazenamento de dados. Um grid, provendo uma visão integrada do armazenamento de dados, é muitas vezes chamado de grids de dados. Cada máquina usualmente provê alguma quantidade de armazenamento para o grid, mesmo que temporária. Armazenamento pode ser memória ligada ao processador ou armazenamento secundário, usando discos rígidos ou outro tipo de mídia de armazenamento. As memórias geralmente possuem um barramento de acesso mais rápido, porém volátil. Seria melhor utilizada para servir como um armazenamento temporário para aplicações em execução.

Armazenamento secundário em um grid pode ser usado de maneiras interessantes para aumentar a capacidade, desempenho, compartilhamento e confiança dos dados. Muitos sistemas do grid usam sistemas de arquivos em rede, como Andrew File System (ASF), Network File System (NFS), Distributed File System (DFS), ou General Parallel

File System (GPFS). Estes sistemas oferecem vários graus de desempenho, funções de segurança e de confiança.

A capacidade pode ser acrescida usando o armazenamento em múltiplas máquinas com um sistema de arquivos unificador. Um arquivo individual ou uma base de dados pode transpor vários dispositivos de armazenamento em máquinas diferentes eliminando as restrições de tamanho máximo, freqüentemente impostas pelos sistemas de arquivo embutidos nos sistemas operacionais tradicionais. Um sistema de arquivos unificador, neste caso, pode também prover um espaço de nomes simples para o armazenamento no grid. Isto torna mais fácil a referência por parte de um usuário a um arquivo, quando precisa acessá-lo, sem se preocupar com sua localização exata. De uma maneira similar, softwares de base de dados especiais podem "confederar" uma variedade de bases de dados e arquivos individuais para formar uma base de dados muito maior, mais abrangente, acessível por meio de funções de consulta à base de dados tradicionais.

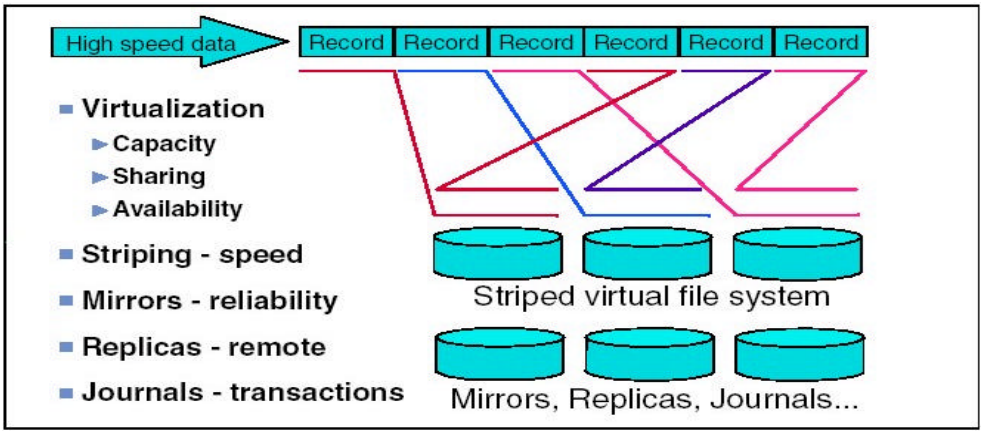

Figura 8 – Data Striping está escrevendo e lendo sucessivos registros dos/para os diferentes recursos físicos sobrepondo o acesso para proporcionar alta transferência; técnicas adicionais para aumentar a confiança[1].

Sistemas de arquivos mais avançados em um grid podem automaticamente duplicar conjuntos de dados, para prover redundância a fim de aumentar a confiança e desempenho. Um escalonador inteligente pode ajudar a selecionar os dispositivos de armazenamento apropriados para manter dados baseados no uso de padrões. Tarefas podem ser então

escalonadas para mais perto dos dados, preferencialmente nas máquinas que estejam diretamente conectadas aos dispositivos de armazenamento que contém os dados requeridos.

Data Striping pode ser implementada pelos sistemas de arquivos do grid, conforme ilustrado na Figura 8. Quando existem padrões de acesso seqüenciais e previsíveis aos dados, esta técnica pode criar um efeito virtual de que temos dispositivos de armazenamento que podem transferir dados em uma taxa mais rápida que qualquer dispositivo de disco tradicional. Isto pode ser importante para fluxos de dados de multimídia ou na coleta de uma enorme quantidade de dados em taxas extremamente altas como experimentos de física envolvendo partículas.

Um sistema de arquivo no grid pode também implementar **journaling,** de modo que os dados possam ser recuperados com maior confiança após determinados tipos de falhas. Além disso, alguns sistemas de arquivos implementam mecanismos de sincronização avançada para reduzir a concorrência quando dados são compartilhados e atualizados por vários usuários.

5.2.1.3 Comunicações

O rápido crescimento na capacidade de comunicação entre máquinas nos dias de hoje, torna o grid computacional útil e prático comparado com a largura de banda limitada, disponível quando a computação distribuída começou a surgir. Então, não é surpresa que outra importante função seja a capacidade de transmissão de dados. Isto inclui a comunicação dentro do grid e externo a ele. Tais comunicações são importantes para enviar tarefas e dados por ela requeridos aos pontos do grid. Algumas tarefas requerem uma grande quantidade de dados para serem processadas e estas informações podem nem sempre estar na máquina que está executando a tarefa. A largura de banda necessária para tal comunicação pode freqüentemente ser um recurso crítico que limita a utilização do grid.

Comunicação externa, como o acesso a Internet, pode ser valiosa quando for necessário construir mecanismos de busca. Máquinas no grid podem ter conexões para redes externas em conjunto com a conectividade entre elas mesmas. Quando estas conexões

não compartilham o mesmo caminho de comunicação com a Internet, elas somam sua banda ao total de banda disponível para acessar a Internet.

Caminhos de comunicação redundantes são necessários em algumas situações para tratar melhor possíveis falhas e tráfego excessivo na rede. Em alguns casos, redes de alta velocidade precisam ser redimensionadas para reunir as demandas de tarefas transferindo grandes quantidades de dados. Um sistema de gerenciamento de um grid pode exibir melhor a topologia da mesma e evidenciar os gargalos de comunicação. Esta informação pode ser usada com o objetivo de planejar novas atualizações.

5.2.1.4 Softwares e Licenças

O grid pode ter softwares instalados em algumas máquinas os quais seriam muito caros para serem instalados em todas as máquinas. Usando um grid, as tarefas que requerem este software são enviadas às máquinas nas quais ela está instalada. Quando as taxas com licenciamento são muito altas, esta abordagem pode evitar despesas significativas para uma organização.

Alguns artifícios permitem que o software seja instalado em todas as máquinas do grid, mas limitam o número de instalações que podem usá-lo de maneira simultânea em um determinado instante. O software de gerenciamento de licença mantém informações de quantas cópias simultâneas do software estão sendo usadas e previne que mais que o número limite de cópias sejam usadas em um determinado período. O escalonador de tarefas pode ser configurado para levar em conta as licenças de software, e opcionalmente realizar um balanceamento das mesmas em frente a outras prioridades ou políticas.

5.2.1.5 Equipamentos especiais, capacidades, arquiteturas e políticas

As plataformas no grid irão freqüentemente possuir arquiteturas, sistemas operacionais, capacidades de dispositivos e equipamentos diferentes. Cada um destes itens representa um tipo diferente de recurso que o grid pode usar como critério para atribuir tarefas às máquinas. Embora certos softwares possam estar disponíveis em várias arquiteturas, como PowerPC e x86, tais softwares são desenvolvidos para rodar apenas em

um tipo particular de hardware e sistema operacional. Tais atributos podem ser considerados quando as tarefas são atribuídas aos recursos no grid.

Em alguns casos, o administrador do grid pode criar um novo tipo artificial de recurso que é usado pelos escalonadores para atribuir trabalho de acordo com as regras de política e outras restrições. Por exemplo, algumas máquinas podem ser designadas para serem usadas somente por pesquisas médicas. Elas seriam identificadas através de um atributo de pesquisa médica e o escalonador poderia ser configurado para somente atribuir tarefas que requeiram máquinas destinadas à pesquisa médica. Outras máquinas poderiam estar participando do grid, somente se elas não forem usadas para propósitos militares. Nesta situação, tarefas requerendo um recurso militar não seriam atribuídas a tais máquinas. Naturalmente, os administradores precisariam impor uma classificação em cada tipo de tarefa através de algum procedimento de identificação para usar este tipo de abordagem.

5.2.1.6 Tarefas e aplicações

Embora vários tipos de recursos no grid possam ser compartilhados e usados, estes são geralmente acessados através de uma aplicação ou tarefa em execução. Usualmente nós usamos o termo “aplicação” como um nível mais alto para representar um pedaço de trabalho em um grid. Porém, algumas vezes o termo “tarefa” é usado de maneira equivalente. Aplicações podem ser quebradas em um número qualquer de tarefas individuais, como ilustrado na Figura 9. Estas, por sua vez, podem ser quebradas em sub-tarefas. A indústria do grid utiliza outros termos, como transações, unidades de trabalho ou submissão de maneira equivalente a tarefa.

Tarefas são programas que são executados em um ponto apropriado no grid. Eles podem computar alguma coisa, executar um ou mais comandos do sistema, mover ou coletar dados ou controlar algum dispositivo. Uma aplicação no grid, organizada como uma coleção de tarefas, é usualmente desenvolvida para comportar a execução paralela de tarefas em diferentes máquinas do grid.

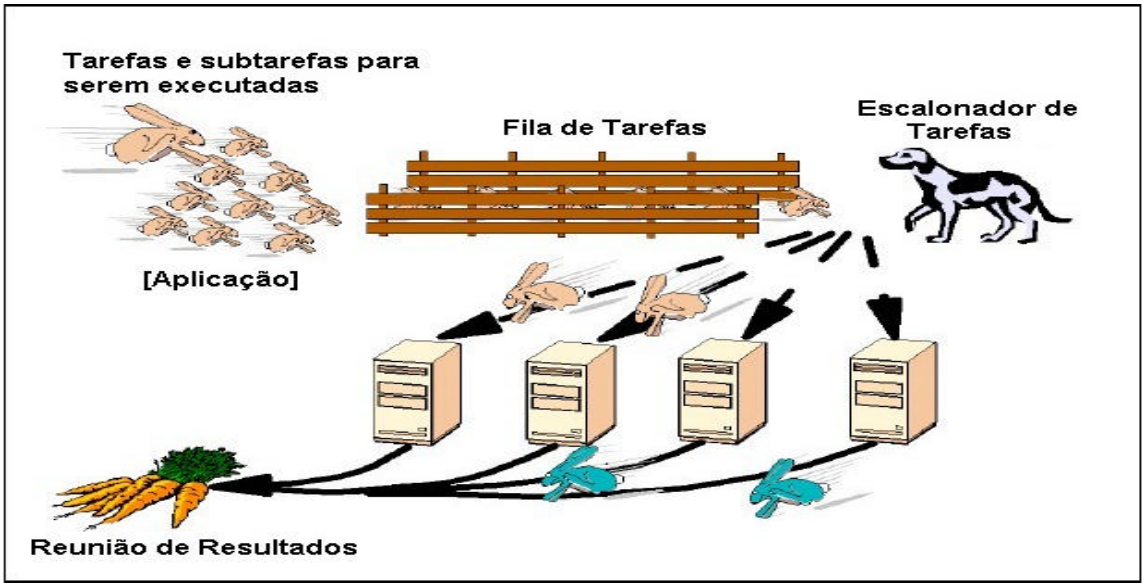

Figura 9 – Uma aplicação é composta por uma ou mais tarefas que são escalonadas para ser executadas em máquinas do grid; os resultados são coletados e reunidos para produzir a resposta [1].

As tarefas podem ter dependências específicas que podem impedir sua execução paralela em qualquer caso. Por exemplo, elas podem requerer dados que precisam ser copiados para a máquina na qual a tarefa está sendo executada. Algumas tarefas podem necessitar da saída produzida por outras e não pode ser executada até que estas, que são pré-requisitos, tenham sua execução concluída. Tarefas podem gerar sub-tarefas adicionais, dependendo dos dados que elas estiverem processando. Este fluxo de trabalho pode criar uma hierarquia de tarefas e sub-tarefas. Finalmente, os resultados de todas estas podem ser coletados e apropriadamente reunidos para produzir a resposta final da aplicação.

5.2.1.7 Escalonando, reservando e buscando recursos ociosos

O sistema de grid é responsável por enviar uma tarefa para uma determinada máquina para ser executada. No mais simples dos sistemas de grid, o usuário pode selecionar uma máquina adequada para rodar sua tarefa e então executar um comando que a envia para a máquina selecionada. Sistemas de grid mais avançados incluiriam algum tipo de escalonador de tarefas que automaticamente procura a máquina mais apropriada para

executar determinada tarefa que está aguardando para ser executada. Escalonadores reagem à disponibilidade atual de recursos no grid. O termo “escalonamento” não pode ser confundido com reserva de recursos na tentativa de promover qualidade de serviço. Algumas vezes o termo “resource broker” é usado no lugar de escalonador, mas este termo implica que alguma espécie de troca de capacidade seja fatorada no escalonador.

Em um sistema de grid que busca recursos ociosos, qualquer máquina que fique ociosa tipicamente reportaria seu estado de ociosidade para o nó de gerenciamento do grid. Este nó de gerenciamento poderia atribuir a esta máquina a próxima tarefa que será executada usando os recursos da máquina. A busca por recursos ociosos é geralmente implementada de uma maneira não intrusiva ao usuário normal da máquina. Se a máquina se tornar ocupada com uma tarefa local não pertencente a grid, esta é normalmente suspendida e deve esperar para retomar sua execução. Isto dá origem a uma situação onde nunca temos uma previsão exata do tempo necessário para que uma tarefa seja completada, embora não chegue a ser um problema o fato das máquinas cederem recursos a grid quando estes estiverem disponíveis.

Para criar comportamentos mais previsíveis, as máquinas do grid estão freqüentemente dedicadas e não são tomadas por trabalhos externos. Isto possibilita que os escalonadores calculem aproximadamente o término de um conjunto de tarefas, quando as suas características de execução são conhecidas.

Como um passo futuro, os recursos do grid podem ser reservados antecipadamente para executar um conjunto de tarefas. Tais reservas operam como um sistema de agenda usado para reservar salas de conferência para reuniões. Isto é feito para reunir prazos e garantias de qualidade de serviço. Quando as políticas permitem, os recursos antecipadamente reservados poderiam também ser garimpados para executar tarefas com baixa prioridade quando eles não estão sendo ocupados durante um período de reserva, respeitando as tarefas que lhe foram atribuídas. Desta forma, várias combinações de escalonamento, reserva e busca por recursos ociosos podem ser usadas para utilizar o grid de forma mais completa.

Escalonamento e reserva são mais claros quando somente um tipo de recurso está envolvido, normalmente CPU. Porém, otimizações adicionais podem ser atingidas no grid

considerando mais recursos no escalonamento e no processo de reserva. Por exemplo, seria desejável atribuir tarefas para máquinas mais próximas dos dados que estas tarefas requerem. Isto reduziria o tráfego na rede e possivelmente reduziria os limites de escalabilidade. O escalonamento perfeito, considerando múltiplos recursos, é um problema matemático muito difícil de ser resolvido. Então, tais escalonadores podem fazer uso de heurísticas. Estas heurísticas são regras atribuídas com o intuito de melhorar a probabilidade de encontrar a melhor combinação de escalonadores e reservas para otimizar a transferência ou qualquer outra métrica.

5.3 Tipos de Grids

De uma perspectiva de uma aplicação, existem dois tipos de grids: grids computacionais e grids de dados. Da perspectiva da topologia, existem tipos adicionais, incluindo clusters, intragrids, extragrids e intergrids. Na realidade estes tipos são melhores definidos como estágios de evolução.Em cada um desses estágios, é possível suportar grids computacionais, grids de dados ou até mesmo a combinação de ambos. A maioria dos desenvolvimentos do grid foram focados no aumento computacional, mas como os grids de dados provem um acesso simples a dados compartilhados, eles também se tornaram importantes.

Um grid computacional é essencialmente uma coleção de recursos computacionais distribuídos, ao longo de locações, que são agregadas para agir como um recurso de processo unificado ou um supercomputador virtual. Estes recursos computacionais podem estar tanto dentro ou entre domínios administrativos. Coletar estes recursos em um conjunto unificado envolve políticas de utilização coordenadas, programação de tarefas, características de enfileiramento, segurança e autenticação de usuário perante o sistema. Seus benefícios se resumem em maior velocidade, maior eficiência para o processamento de tarefas computacionais intensivas, enquanto utiliza recursos existentes. Grids computacionais eliminam também o defeito de amarrar fortemente máquinas específicas para tarefas específicas, permitindo disponibilizar serviços seqüenciais ou tarefas paralelas com boa granulação nos atributos de usuário.

Um grid de dados provê uma ampla área de acesso seguro para os dados correntes. Permite aos usuários e aplicações gerenciar e eficientemente utilizar informações da base de dados das locações distribuídas. Assim como grids computacionais, os grids de dados também contam com um software para políticas de utilização e acesso seguro. Podem ser desenvolvidos dentro de um domínio administrativo ou através de múltiplos domínios. São nesses casos onde os softwares do grid e as políticas de gerenciamento se tornam críticas. Os grids de dados eliminam a necessidade de mover dados sem necessidade, ou replicá-los ou também centralizá-los.

A evolução dos grids computacionais para os grids de dados é um fator importante na reposição de aplicações do grid para área educacional e empresarial. Esta transição é uma indicação que o mercado, assim como a tecnologia, está evoluindo. Da perspectiva de uma rede, o impacto dos grids de dados incluirá uma forte integração de protocolos de armazenamento e uma alta performance na rede.

Na figura 10, observa-se os passos e as características de cada etapa da evolução da grid, que serão explicadas na próxima seção:

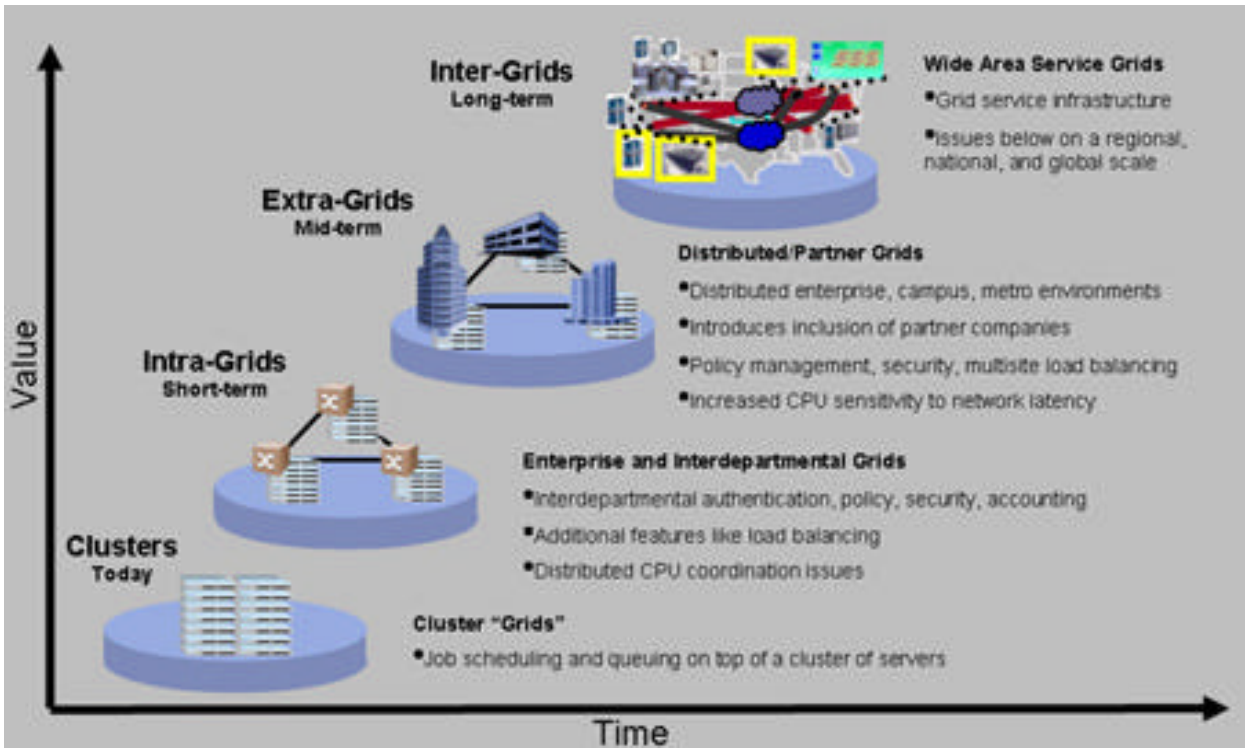

Figura 10 – A Evolução do grid [21].

Atualmente as companhias de empreendimento utilizam esta tecnologia, com o intuito de reduzir custos e aumentar a vantagem competitiva, utilizando recursos disponíveis ao invés de investir pesado em equipamentos de alta performance de última geração. O amadurecimento do middleware do grid para grids computacionais e de dados, em adição dos recentes avanços nos equipamentos de rede de dados, irão prover o momento necessário para as grids evoluírem do seu cluster local para o distribuído, constituindo os sistemas de área aberta "wide-area systems".

5.3.1 Intragrid, Intergrid e Extragrid

Houve algumas tentativas de formular uma definição precisa sobre o que significa grid. De fato, o conceito de grid computacional está evoluindo e a maioria das tentativas de definí-lo precisamente excluem implementações que muitos considerariam ser grids.

Os grids podem ser construídos em todos os tamanhos, variando apenas de algumas máquinas em um departamento a uns grupos de máquinas organizadas como uma hierarquia.

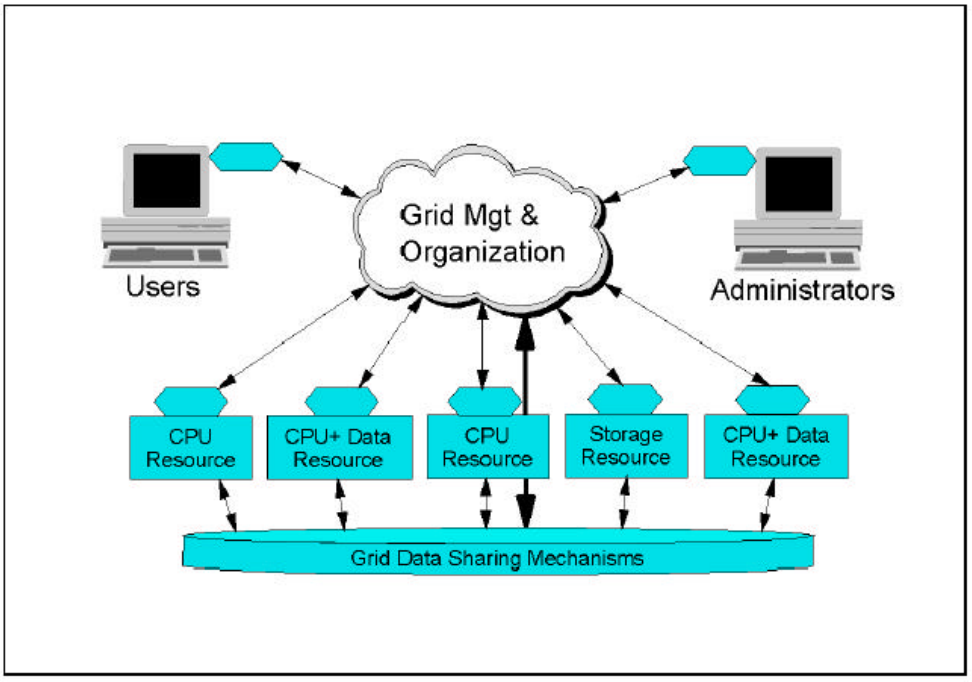

Figura 11 – Um grid simples [1].

Como apresentado na figura 11, o grid mais simples consiste apenas em algumas máquinas, todas de mesma arquitetura e de mesmo sistema operacional, conectada em uma rede local. Este tipo de sistemas homogêneos de uso do grid contém poucas considerações e pode ser usado como experimento de um software de grid. As máquinas estão geralmente dispostas em um departamento de uma organização, e as utilizando como membros de um grid não deve requerer nenhuma política ou interesse especial de segurança. Devido às máquinas terem a mesma arquitetura e sistema operacional, o software de aplicação escolhido para estas máquinas é geralmente simples. Alguns chamariam isto de uma implementação em cluster.

A progressão seguinte deve incluir máquinas heterogêneas. Nesta configuração, mais tipos de recursos estão disponíveis. É provável que o sistema de grid inclua alguns componentes de programação. O compartilhamento de arquivos pode ainda ser realizado usando sistemas de arquivos de rede. As máquinas que participam no grid podem incluir departamentos múltiplos, mas dentro da mesma organização. Tal grid é conhecida como "Intragrid".

Quando o grid atravessa os limites dos departamentos de uma corporação, políticas podem ser requeridas para definir como este deve ser usado. Como exemplo, podem existir políticas definindo quais tipos de tarefas são permitidos no grid em determinados períodos. É possível haver uma priorização por departamento ou por tipos de aplicações que devem ter acesso aos recursos do grid. Também, a segurança torna-se mais importante à medida que mais organizações sejam envolvidas. Os dados sensíveis em um departamento podem necessitar serem protegidos do acesso por tarefas que funcionam para outros departamentos. Máquinas dedicadas podem ser adicionadas para aumentar a qualidade do serviço no grid computacional, ao invés de depender inteiramente dos recursos ociosos.

O grid pode crescer geograficamente em uma organização que tenha recursos em diferentes cidades. As conexões de comunicações dedicadas podem ser usadas entre estes recursos e o grid. Em alguns casos, um tunelamento VPN ou outras tecnologias pode ser usado sobre a internet para conectar as diferentes partes da organização. A segurança aumenta em importância uma vez que os limites de determinados recursos sejam

atravessados. O grid pode vir a crescer de forma hierarquicamente organizada para reduzir a disputa pelo controle central, aumentando sua escalabilidade.

Os desafios adicionais nas fases posteriores nos traz o conceito de extragrids. Extragrids são essencialmente clusters de grid e/ou intragrids que são conectados entre lugares geograficamente distribuídos dentro ou entre organizações empresariais. As principais distinções incluem a distribuição geográfica e a relação inter empresa. Devido a estas relações, extragrids são algumas vezes referenciados como grids sócias. Portanto, pode-se ter processamento e/ou compartilhamento dos dados entre duas organizações diferentes. A autenticação, o gerenciamento de políticas e a segurança tornaram-se requisitos críticos que o middleware precisa trabalhar.

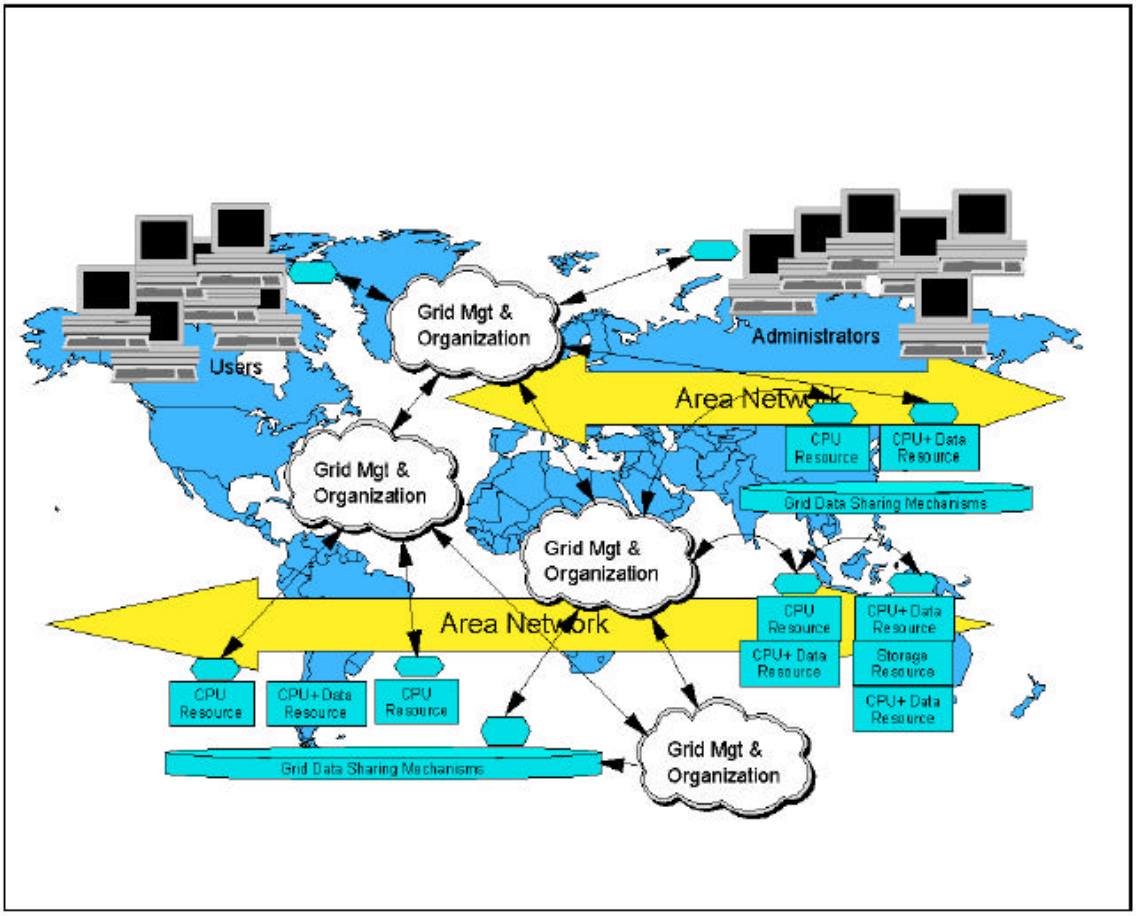

Figura 12 – Uma intergrid mais complexa[1].

Além do mais, como ilustrado na Figura 12, um grid pode vir a crescer de maneira a cruzar os limites da organização, e pode ser usado para colaborar em projetos de interesse comum. Isto é conhecido como "Intergrid". Os níveis mais elevados de segurança são geralmente requeridos nesta configuração para impedir possíveis ataques e espionagem. O Intragrid oferece a capacidade de negociar recursos sobre uma audiência muito mais larga. Os recursos podem ser adquiridos como uma funcionalidade dos fornecedores confiáveis.

5.4 Arquitetura do Grid

Projetar um grid viável inclui muito mais do que a agregação da força de processamento de máquina ociosa via software. Existem muitas camadas, cada qual representando seus pontos críticos. Em geral, as camadas mais altas são mais centradas em software, enquanto as mais baixas em hardware. O diagrama abaixo fornece uma visão geográfica das camadas que definem uma grid computacional e a função que cada uma provê.

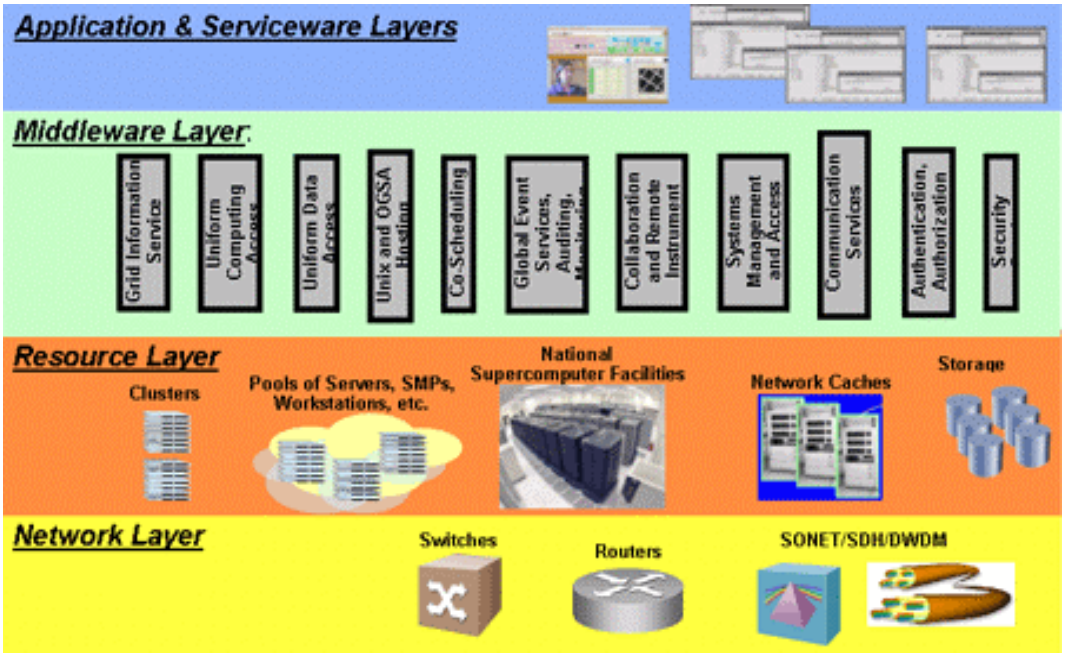

Figura 13 – Arquitetura de um grid [21].

O software de aplicação e Serviceware define as camadas mais altas de uma arquitetura do grid. Isto inclui também kits de ferramentas de desenvolvimento. As aplicações variam de indústria para indústria, dependendo do problema de negócio. O Serviceware provê diversas funções de gerenciamento, incluindo faturamento, contabilidade e medidas de métricas utilizáveis – todos pontos importantes para a virtualização dos recursos compartilhados entre diferentes usuários, departamentos e companhias.

A camada de middleware fornece protocolos permitindo múltiplos elementos (servidores, armazenadores, redes, etc.) participem em um ambiente de grid unificado. Esta camada pode ser referenciada pela sua inteligência, trazida aos vários elementos unidos através de software e domínio. Existem diversos protocolos e funções diferentes que a camada de middleware suporta. A camada de recursos é constituída por recursos que fazem parte do grid, incluindo servidores primários e dispositivos de armazenamento. A camada de rede é a base da conectividade para os recursos do grid.

Arquiteturas de múltiplas camadas como os grids demandam uma forte padronização, pois equipamentos de diferentes fornecedores serão membros do desenvolvimento distribuído. Existem vários padrões de especificação de grids preliminares. Abaixo segue uma lista das atividades de padronização mais significativas para o grid computacional:

- ***Fórum Global do Grid*** (Global Grid Fórum - GGF): é responsável por promover e dar suporte ao desenvolvimento de padrões de grids. As recomendações processadas dentro do modelo GGF pertencentes ao IETF (Internet Engineering Task Force), seguem um modelo comprovado;

- Patrocinadores do GGF incluem Hewlett-Packard Co., IBM Corp., Intel Corp., Microsoft Corp., e a Sun Microsystems Inc., entre diversos outros;

- ***Globus***: Um projeto de pesquisa e desenvolvimento focado na capacitação de aplicações do grid fornecendo ferramentas que facilitem o desenvolvimento de

grids computacionais. Globus auxilia o desenvolvimento do Kit de Ferramentas Globus - Globus Toolkit. A organização do projeto é dividida em quatro atividades principais: aplicações, pesquisas, ferramentas de software e bateria de testes;

- ***Kit de Ferramentas Globus*** (Globus Toolkit): Uma arquitetura aberta, um kit de ferramentas de software livre permitindo que usuários construam mais facilmente grids computacionais. Fornece uma coleção de protocolos e definições de serviços para desenvolvimento do grid;

- ***Arquitetura de serviços da grade aberta*** (Open Grid Services Architecture - OGSA): A proposta de evolução do kit atual Globus é dirigida a uma integração do grid com conceitos e tecnologias dos serviços Web;

- ***Outros***: Existem diversas outras iniciativas, algumas incluindo o OGSI – WG (Open Grid Service Infrastructure Working Group), Access Grid e Condor.

Muitas dessas iniciativas existem há anos, mas somente agora estão começando a alcançar um estágio significante, a partir do momento que as grandes corporações empresariais começaram a testar e desenvolver redes em grid. As indústrias mais significativas nos setores de servidores, dispositivos de armazenamento e rede passaram a incentivar essas iniciativas de padronização.

5.5 Construção do grid

Um grid temporário pode ser instalado por alguns programadores em seu tempo livre, mas à medida que o grid cresce, e enquanto os usuários se tornam mais dependentes dela para um trabalho de missão crítica, um grau de planejamento é essencial. É melhor compreender as exigências das organizações e escolher tecnologias do grid que melhor

ajuste a estas exigências. Esta seção discutirá algumas das considerações de planejamento e os componentes do grid que se dirigem às exigências.

5.5.1 Desenvolvimento planejado

O uso de um grid advém freqüentemente de uma necessidade de obter mais recursos sobre algum assunto. Um olha freqüentemente a seu vizinho que pode ter a capacidade adicional no recurso particular. Uma das primeiras considerações é o hardware disponível e como é conectado através de uma LAN ou de uma WAN. Após isso, uma organização pode querer adicionar hardware adicional para aumentar as potencialidades do grid. É importante compreender as aplicações a serem usadas no grid. Suas características podem afetar as decisões de como melhor escolher e configurar o hardware e sua conectividade aos dados.

5.5.1.1 Segurança

A segurança é um fator muito mais importante no planejamento e na manutenção de um grid do que em uma computação distribuída convencional, onde os compartilhamentos dos dados compreendem o volume da atividade. No grid, as máquinas que o compõe são configuradas para executar programas que preferencialmente apenas movam dados. Isto faz com que um grid inseguro se torne uma região potencialmente produtora de programas de vírus e cavalos de tróia. Por esta razão, é importante compreender exatamente quais componentes do grid devem rigorosamente ser fixados para deter qualquer tipo de ataque. Além disso, é importante compreender as ações envolvidas na autenticação de usuários e na execução correta das responsabilidades de uma autoridade certificadora.

5.5.1.2 Organização

As considerações da tecnologia são importantes no desenvolvimento do grid. Entretanto, as ações organizacionais e de negócios podem ser igualmente importantes. É importante compreender como os departamentos em uma organização interagem, se operam, e se contribuem ao todo. Freqüentemente, há barreiras construídas entre

departamentos e projetos para proteger seus recursos em um esforço para aumentar a probabilidade do sucesso oportuno. Entretanto, repensando alguns destes relacionamentos, pode-se dizer que compartilhar os recursos pode às vezes beneficiar a organização inteira da melhor maneira. Por exemplo, um projeto que se encontre atrasado e com orçamento alto pode não proporcionar recursos requeridos para resolver o problema. Um grid daria a tais projetos uma medida adicional de segurança, fornecendo uma margem extra da capacidade de recurso necessário para terminar o projeto. Similarmente, um projeto em seus estágios adiantados, quando os recursos computacionais não estão sendo utilizados inteiramente, pode-se doá-los a outros projetos quando preciso. Um grid oferece também a habilidade na gerência das organizações.

5.5.2 Componentes de software do grid

Esta seção apresenta alguns dos componentes chave que devem ser discutidos antes de projetar uma arquitetura de um grid computacional.

5.5.2.1 Componentes de gerência

Qualquer sistema no grid contém alguns componentes de gerência. Primeiramente, há um componente que mantém a par os recursos disponíveis no grid e quais usuários são membros. Esta informação é usada primeiramente ao se decidir onde as tarefas do grid devem ser atribuídas. Em segundo, existem componentes de medida que determinam as capacidades dos nós no grid e sua taxa atual de utilização em qualquer tempo dado. Esta informação é usada para programar tarefas no grid. Tal informação é usada também para determinar o estado do grid, alertando eventuais problemas tais como interrupção, congestionamentos, ou o excesso de comprometimentos. Esta informação é usada também para determinar testes padrões e estatísticas totais de uso, assim como para registrar e esclarecer o uso dos recursos do grid. Em terceiro lugar, o software avançado de gerência do grid pode automaticamente controlar muitos aspectos. Isto é conhecido como computação autônoma ou computação orientada a recuperação “autonomic computing”. Este software recupera automaticamente vários tipos de falhas e de interrupções do grid, encontrando maneiras alternativas para começar a carga de trabalho processada.

5.5.2.2 Software de Fornecimento

Cada máquina contribui com recursos tipicamente necessários para registrar-se como um membro do grid e instalar algum software que controle o uso dos seus recursos. Geralmente, algumas identificações e procedimentos de autenticação devem ser executados antes que uma máquina possa se juntar a grid. Uma autoridade certificadora pode ser usada para estabelecer a identidade da máquina fornecedora, dos usuários e do próprio grid.

Alguns sistemas em grid fornecem seu próprio login enquanto outros dependem de sistemas operacionais nativos para a autenticação de usuário. Em último caso, um sistema mapeado de ID de usuário pode ser útil para combinar corretamente os direitos de usuários em máquinas diferentes. Isto é mantido por um administrador do grid. Ele determina qual ID de usuário um dado usuário pode possuir em cada máquina do grid e incorporar estes IDs a uma base de dados ou a um registro protegido. Desta maneira, quando as tarefas do grid são submetidas a diferentes máquinas para um usuário, o ID do usuário apropriado da máquina local é usado determinando os direitos dos usuários.

Em alguns sistemas em grid, é possível se unir ao grid sem nenhuma autenticação especial. E em outros, é possível que qualquer usuário submeta tarefas ao grid. Tais sistemas podem ser convenientes ajustados, mas devem ser desencorajados nas distribuições maiores devido aos sérios problemas de segurança.

O sistema em grid fornece informações sobre os recursos disponíveis recentemente adicionados por todo o grid. A máquina fornecedora terá geralmente algum tipo de monitor que determina ou mede quão ocupada a máquina está e a taxa ou a quantidade de recursos utilizados. Esta informação é repassada ao software de gerência do grid e usada para escalonar o uso destes recursos conforme for. Em um sistema de recursos ociosos, esta informação diz ao software de gerência do grid quando a máquina está inativa e disponível para a tarefa.

Um ponto importante, é que o software instalado em uma dada máquina possa aceitar uma tarefa executável do sistema de gerência do grid e executá-la. Um usuário em algum lugar do grid submete uma tarefa para execução. O software de gerência do grid deve comunicar-se com o software fornecedor do grid para emitir a tarefa. Ele deve ser

capaz de receber o arquivo executável ou selecionar de forma apropriada uma das cópias pré-instaladas na máquina fornecedora. O software é executado e a saída é enviada de volta para o requisitante. Implementações mais avançadas podem dinamicamente ajustar a prioridade de uma tarefa em andamento, suspendê-lo e recomeça-lo mais tarde, ou verificar o ponto que ela tenha a possibilidade de recomeçar sua execução em uma máquina diferente. Estes tipos de ações podem ser necessários para responder aos problemas de balanceamento de carga ou prioridade ou mudanças de política no grid.

5.5.2.3 Software de Submissão

Geralmente qualquer máquina que compõe um grid pode ser usada para submeter tarefas para iniciar consultas no grid. Entretanto, em alguns sistemas, esta função é executada como um componente separado instalado em "nós de submissão" ou "clientes de submissão". Quando um grid é construído usando recursos dedicados especialmente os que são ociosos, o software de submissão é instalado geralmente de forma distinta no desktop ou em uma estação de trabalho dos usuários.

5.5.2.4 Gerenciamento no grid distribuído

Os grids maiores podem ter uma hierarquia ou outro tipo de topologia organizacional que combina geralmente com a topologia de conectividade. Isto é, máquinas localmente conectadas junto a uma LAN formam um "cluster" de máquinas. O grid pode ser organizado em hierarquia de clusters. O trabalho envolvido no gerenciamento do grid é distribuído para aumentar a escalabilidade. A coleção, a operação e os dados de recurso do grid bem como a tarefa escalonada é distribuída para unir-se a topologia do grid. Por exemplo, um escalonador de tarefa central não programará uma tarefa submetida diretamente para a máquina a qual está para executá-la. Ao invés disso, a tarefa é emitida a um escalonador de um nível mais baixo que assegure um conjunto de máquinas (ou clusters mais distantes). O escalonador de nível mais baixo assegura a missão à máquina específica. Similarmente, a coleção de informação estatística é distribuída. Os clusters de níveis mais baixos recebem a informação de atividade das máquinas individuais, agregam-na, e emitem-na aos nós de gerência de um nível mais elevado na hierarquia.

5.5.2.5 Escalonadores

A maioria dos sistemas em grid inclui algum tipo de software de escalonamento de tarefa. Este software localiza uma máquina na qual rode uma tarefa que seja submetida por um usuário. Nos casos mais simples, pode-se apenas ser atribuídas tarefas em uma forma de "round-robin fashion" à máquina seguinte que combina as exigências de recurso. Entretanto, há algumas vantagens em usar um escalonador mais avançado.

Alguns escalonadores executam um sistema de prioridade de tarefas. Isto às vezes é feito usando diversas filas de tarefas, cada uma com uma prioridade diferente. Enquanto as máquinas do grid se tornam disponíveis para executar tarefas, estas são selecionadas a partir da fila de maior prioridade. Políticas podem incluir vários tipos de restrições sobre as tarefas, usuários, e recursos. Por exemplo, pode haver uma política restringindo a execução das tarefas no grid por um determinado intervalo de tempo.

Os escalonadores reagem geralmente à carga imediata do grid. Usam a informação da medida sobre a utilização atual das máquinas para determinar quais não estão ocupadas antes de submeter uma tarefa. Os escalonadores podem ser organizados em uma hierarquia. Por exemplo, um meta-escalonador pode submeter uma tarefa a um escalonador do cluster ou a outro escalonador de nível mais baixo de preferência a de uma máquina individual.

Escalonadores mais avançados monitorarão o progresso das tarefas programadas que controlam o fluxo total de trabalho. Se as tarefas forem perdidas devido a interrupções do sistema ou da rede, um bom escalonador submeterá novamente a tarefa de maneira automática em outra parte do grid. Entretanto, se uma tarefa parecer estar em um laço infinito e alcançar um intervalo de interrupção máxima, então tais tarefas não devem ser escaladas novamente. Tipicamente, as tarefas têm tipos diferentes de códigos de conclusão, alguns são apropriados para uma re-submissão e alguns não são.

A reserva de recursos no grid é realizada através de um sistema de reserva. É mais do que um escalonador. É primeiramente, um sistema baseado em um calendário com o objetivo de reservar recursos por períodos de tempo específicos e impedir a reserva do mesmo recurso ao mesmo tempo. Também deve remover ou suspender as tarefas que funcionam em todas as máquinas ou recursos quando o período de reserva é alcançado.

5.5.2.6 Comunicações

Um sistema em grid pode incluir um software para ajudar as tarefas a comunicarem entre si. Por exemplo, uma aplicação pode se dividir em um grande número de sub-tarefas. Cada uma destas sub-tarefas é uma tarefa separada no grid. Entretanto, a aplicação pode executar um algoritmo que requeira que as sub-tarefas se comuniquem e troquem informações entre elas. As sub-tarefas precisam ser capazes de localizar outras sub-tarefas específicas, estabelecer uma conexão de comunicações com elas, e enviar os dados apropriados. O padrão aberto MPI e qualquer uma de suas variações são freqüentemente incluídos como parte do grid apenas neste tipo de comunicação.

5.5.2.7 Observação, Gerenciamento e Medição

Nós mencionamos acima a reação dos escalonadores às cargas atuais no grid. Geralmente, o software fornecedor incluirá algumas ferramentas que medem a carga e a atividade atual em uma dada máquina usando um dos dois, as facilidades do sistema operacional ou pela medida direta. Este software é consultado as vezes como um sensor de carga. Alguns sistemas em grid fornecem meios de implementação adaptados a sensores de carga para outras CPU ou recursos armazenados.

Tal informação de medida é útil não somente para o escalonamento, mas também para descobrir padrões totais do uso no grid. As estatísticas podem mostrar as tendências a qual pode sinalizar a necessidade de um hardware adicional. Além disso, as informações da medida sobre as tarefas específicas podem ser coletadas e usadas para predizer melhor as exigências do recurso dessa tarefa para próxima vez que for solicitada. Quanto melhor a predição, mais eficiente é controlada a carga de trabalho dos grids.

A informação da medida pode também ser conservada para finalidades de contabilidade, para formar a base para o fatiamento do recurso do grid, ou controlar mais razoavelmente as prioridades. A informação pode também ser indicada em vários formulários para melhor visualizar a atividade e a utilização do grid.

5.5.2.8 Autoridade Certificadora

É critico assegurar altos níveis de segurança num grid, pois ele é projetado para executar código e não apenas para compartilhar dados. Dessa forma, ele pode servir de base para produção de vírus, cavalos de tróia, e outros ataques se o grid ceder de alguma forma. A autoridade certificadora é um dos aspectos mais importantes de suporte forte para a segurança do grid. Uma organização deve escolher uma autoridade certificadora externa ou operar a sua própria autoridade. Você deve ser capaz de ter confiança na autoridade certificadora para atribuir estritamente as suas responsabilidades.

As responsabilidades primárias de uma autoridade certificadora são:

- Identificar entradas de pedidos de certificados de forma positiva;
- Distribuir, remover, e arquivar certificados;
- Proteger servidor de autoridade certificadora;
- Manter um espaço de nomes de nomes únicos para os donos de certificado;
- Entregar certificados assinados àqueles que precisam por entradas autenticadas.

Resumidamente, uma autoridade certificadora é baseada num sistema de cifragem por chave pública. Neste sistema, as chaves são geradas aos pares, uma chave pública e uma chave privada. Um dos dois podem ser usados para cifrar alguns dados do mesmo modo que o outro é usado para decifrá-los. A chave privada é guardada pelo dono e nunca revelada a ninguém. A chave pública é entregue a alguém que precise. Uma autoridade certificadora é usada para guardar chaves públicas e para garantir que elas pertençam a uma chave privada correspondente. Quando um usuário usa sua chave privada para cifrar algo, o receptor usa a chave publica correspondente para decifrá-lo. O receptor sabe que somente a chave pública do usuário pode decifrar a mensagem corretamente. No entanto, alguém poderia interceptar essa mensagem e decifrar ela porque qualquer um pode ter acesso a chave pública criada. Se a criada ao invés de duas vezes cifrar a mensagem com sua chave

privada e cuja intenção da chave pública do receptor, uma conexão de comunicação segura é formada. O receptor usa sua chave privada para decifrar a mensagem e por isso usa a chave pública despachada para uma segunda decifração. Visto que o receptor sabe que se a mensagem decifrada corretamente, então somente o transmissor poderia ter enviado-o e além disto, o transmissor sabe que somente o receptor alvo pode decifrá-lo. A vantagem de tudo isto é que ninguém tem seguramente alcance a uma chave de cifragem desde o transmissor ao receptor, à medida que deva ser feito por sistemas de cifragem convencional, e qualquer ocupação na comunicação é revelada. Uma troca similar é usada para obter uma chave pública de qualquer um a partir de uma autoridade certificadora, de modo que o usuário saiba que ele tenha recebido uma chave pública inalterada do usuário desejado.

6. Ferramenta Globus

Existem atualmente vários softwares disponíveis para o desenvolvimento de um grid computacional, como se pode verificar no anexo A.

O projeto *Globus*, feito em conjunto pelo *Argonne National Laboratory* e a *University of Southern California's Information Sciences Institute*, desenvolve uma tecnologia necessária à construção de sistemas grid. O núcleo deste projeto é o desenvolvimento da infraestrutura básica para aplicações que integram sistemas geograficamente distribuídos. Um protótipo de *Grid* computacional, o *GUSTO*, vem sendo desenvolvido, bem como aplicações de software para o *Grid* que servem para testar esta infraestrutura.

O *Globus* é um trabalho em progresso, pois a medida que novos problemas são descobertos, novas técnicas são propostas e avaliadas para saná-las. Dessa maneira, avanços nesta infraestrutura facilitam a utilização da computação distribuída pelo aumento do desempenho e elimina a necessidade do usuário conhecer extensivamente o sistema utilizado. A abordagem do *Globus* difere-se de outras arquiteturas de *Grid* em três pontos: pelo seu modelo "*bag of services*" que permite que aplicações usem os serviços do *Grid* sem ter que adotar um modelo particular de programação; pelos mecanismos especializados que podem coexistir e até mesmo substituir mecanismos já pertencentes ao kit; e pelo seu suporte a abordagens baseadas em informação para determinar requisitos de performance das aplicações.

Esta seção apresenta alguns dos principais componentes do Globus Toolkit 2.2, o qual fornece:

- Assinatura única, autorização, e autenticação;
- Submissão de tarefas;
- Monitoramento, busca e alocação de recursos;
- Movimentação de dados.

Além disso, o Globus Toolkit fornece uma coleção de ferramentas para programação de aplicações (APIs) e Kits de desenvolvimento de sistemas (SDKs)

Informação extensiva sobre o Globus Toolkit e o Projeto Globus podem ser encontrados em: http://www.globus.org/

6.1 Três Pirâmides

Globus Toolkit é formado por três pirâmides de suporte à construção sobre a infraestrutura de segurança, como ilustrado na Figura 14. Eles são:

- Gerenciamento de recurso;
- Gerenciamento de Dados;
- Serviços de Informação.

Todas essas pirâmides são construídas sobre a Grid Security Infrastructure (GSI).

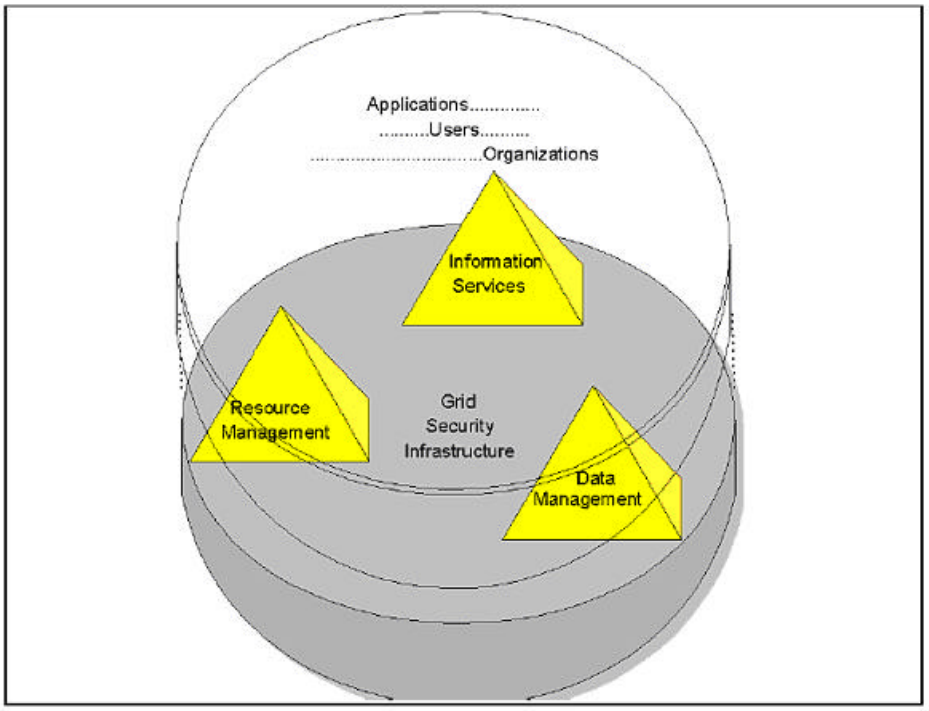

Figura 14 – As três pirâmides [22].

6.1.1 Gerenciamento de recursos

A pirâmide de gerenciamento de recursos fornece suporte a:

- Alocação de recurso;
- Submissão de tarefas: remotamente arquivos executáveis executam e recebem resultados;
- Gerenciamento do status e do progresso da tarefa.

Globus Toolkit não tem seu próprio escalonador de tarefa para encontrar recursos disponíveis e automaticamente enviá-los para máquinas apropriadas. Ao invés disso, ele fornece as ferramentas e interfaces necessárias para implementar escalonadores.

6.1.2 Serviços de Informação

A pirâmide de serviços de informação suporta coletar informações na grid e consultar esta informação, baseada no Lightweight Directory Acess Protocol (LDAP).

6.1.3 Gerenciamento de Dados

A pirâmide de gerenciamento de dados fornece suporte na transferência de arquivos entre máquinas no grid e no gerenciamento dessas transferências.

6.2 Componentes do Globus Toolkit

Para cada pirâmide previamente apresentada, Globus fornece um componente para implementar gerenciamento de recursos, gerenciamento de dados, e serviços de informação, como ilustrado na Figura 15.

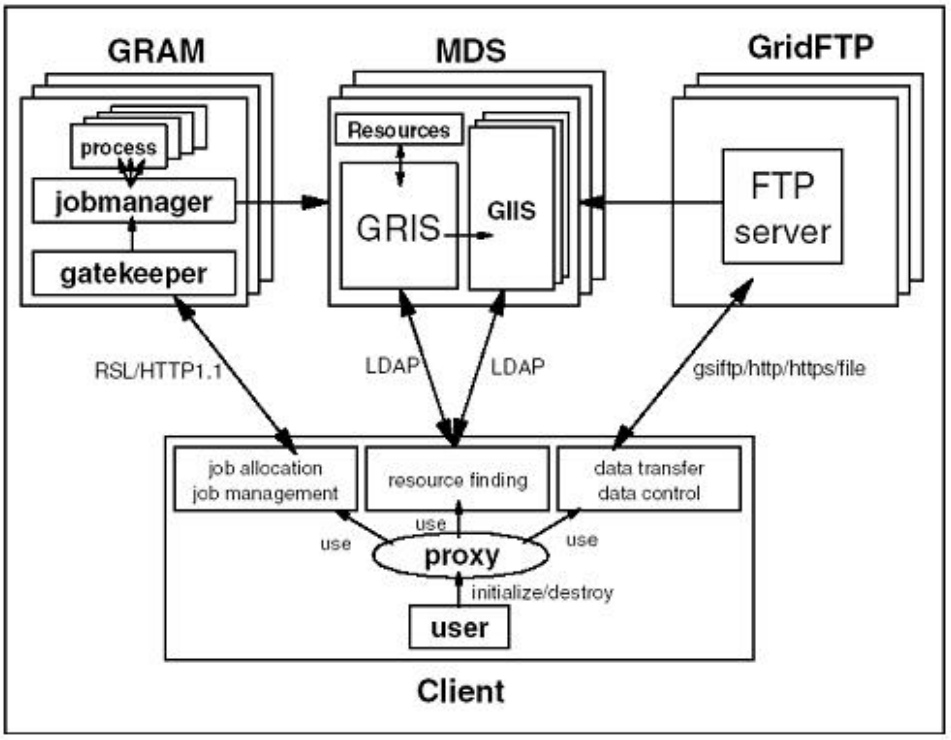

Figura 15 – Uma visão geral do sistema do Globus Toolkit [22].

GRAM/GASS

Os componentes primários desta pirâmide são o Grid Resource Allocation Manager (*GRAM*) e o Global Access to Secondary Storage (*GASS*).

MDS (GRIS/GIIS)

Baseado nos componentes Lightweight Directory Access Protocol (*LDAP*), o Grid Resource Information Service (*GRIS*) e Grid Index Information Service (*GIIS*) pode ser configurado em uma hierarquia para coletar informação e distribuí-la. Estes dois serviços são chamados Monitoring and Discovery Service (*MDS*). A informação coletada pode ser informação estática sobre as máquinas bem como informação dinâmica mostrando a atividade corrente da CPU e do disco. Um rico conjunto de provedores de informação é

incluído com o Toolkit e os usuários Globus podem acrescentar o seu próprio. A informação fornece uma interface com o *GRIS*, o qual relata esta informação para a hierarquia dos servidores GIIS na grid. A linguagem de consulta LDAP é usada para alcançar a informação desejada.

GridFTP

GridFTP é um componente primordial para a segurança e alta transferência de dados. O Globus Replica Catalog and Management é usado para registrar e gerenciar tanto cópias parciais quanto completas de coleções de dados. Estas três pirâmides são modularizadas e podem funcionar isoladas; no entanto, juntas, completam-se.

GSI

Todos os componentes acima são construídos sobre o Grid Security Infrastructure (*GSI*). Esta fornece funções de segurança incluindo autenticação única/mútua, comunicação confidencial, autorização e delegação.

6.2.1 Grid Security Infrastructure (GSI)

GSI fornece elementos para autenticação e comunicação segura na grid. A infraestrutura é baseada no protocolo SSL (Secure Socket Layer), cifração por chave pública, e certificados x.509.

Para uma assinatura única, Globus acrescenta extensões no GSI. Ela é baseada na API Generic Security Service, sendo esta uma API padrão elaborada pela Internet Engineering Task Force (IETF).

Estas são as principais funcionalidades implementadas pelo GSI:

- Autenticação única/mútua;
- Comunicação confidencial;
- Autorização;
- Delegação.

Maiores informações a respeito do mecanismo do GSI pode ser visto em detalhes no site oficial do GSI: http://www.globus.org/security/

6.2.2 Grid Resource Allocation Manager (GRAM)

GRAM é o módulo que fornece a execução remota e o gerenciamento do status da execução. Quando uma tarefa é submetida por um cliente, o pedido é enviado para o host remoto e tratado pelo daemon gatekeeper localizado no host remoto. Então o gatekeeper cria um gerenciador de tarefas para inicializá-la e monitora-la. Quando a tarefa é finalizada, o gerenciador envia a informação de status de volta para o cliente e finaliza-se.

O site oficial para o GRAM é: http://www.globus.org/gram

A Figura 16 descreve a visão conceitual sobre o GRAM. Ele contém os seguintes elementos:

- O comando **globusrun;**
- Resource Specification Language (*RSL*);
- O daemon gatekeeper;
- O gerenciador de tarefas;
- O processo bifurcado;
- Global Access to Secondary Storage (*GASS*);
- Dynamically-Updated Request Online Coallocator (*DUROC*).

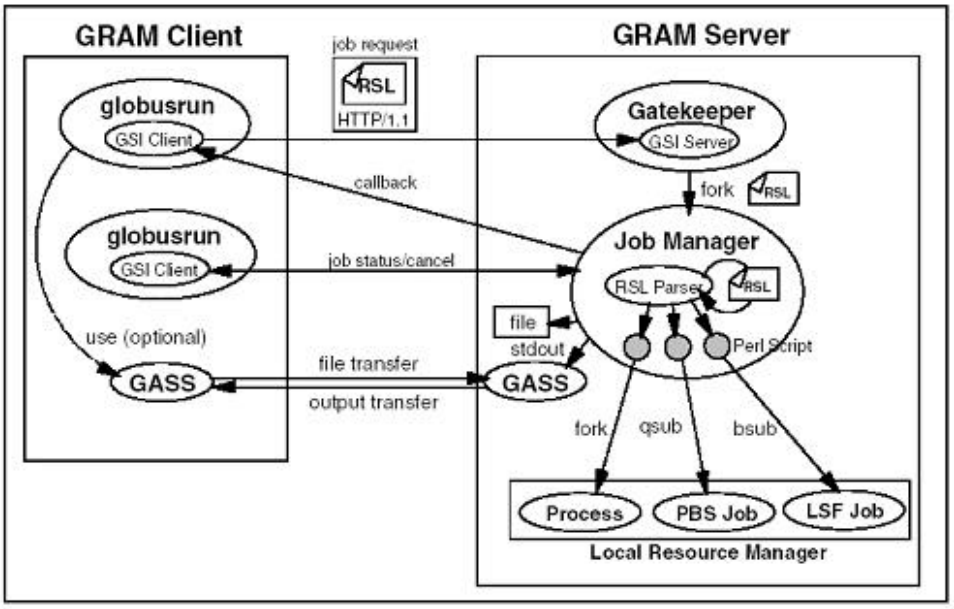

Figura 16 – Visão geral do GRAM [22].

6.2.2.1 O comando globusrun

O comando **globusrun** submete e gerencia tarefas remotas e utiliza quase todas as ferramentas cliente do GRAM. Este comando fornece as seguintes funções:

- Pedido de submissão de tarefa para máquinas remotas

  Submissão de tarefa usa funções de segurança (tais como GSS-API) para checar autenticação mútua entre clientes e servidores, e também verifica os direitos para submeter a tarefa.

- Transferir os arquivos executáveis e o resultado da submissão da tarefa da saída de arquivos

  O comando **globusrun** tem a capacidade de obter os resultados das tarefas submetidas às máquinas remotas. Ele usa *GASS* para fornecer segurança na transferência de arquivo entre máquinas do grid.

6.2.2.2 Resource Specification Language (RSL)

RSL é uma linguagem usada pelos clientes para submeter uma tarefa. Todos os pedidos de submissão de tarefa são descritos no *RSL*, incluindo o arquivo executável e a condição na qual deve ser executado. Você pode especificar, por exemplo, a quantidade de memória necessária para executar uma tarefa na máquina remota.

6.2.2.3 Gatekeeper

O daemon gatekeeper constrói a comunicação segura entre clientes e servidores. Ele é similar ao daemon *inetd* ou *xinetd* em termos de funcionalidade. No entanto, gatekeeper fornece uma comunicação segura. Ele comunica-se com o cliente *GRAM* (globusrun) e autentica o direito para submeter tarefas. Após a autenticação, o gatekeeper bifurca-se e cria um gerenciador de tarefas que delega a autoridade para comunicar-se com os clientes.

6.2.2.4 Gerenciador de Tarefas

Gerenciador de tarefas é criado por um daemon gatekeeper como parte do processo de requisição da tarefa. Ele fornece as interfaces que controlam a alocação de cada gerenciador de recurso local, tais como um escalonador de tarefa como *PBS*, *LSF*, ou *LoadLeveler*.

As funções do gerenciador de tarefas são:

- Análise gramatical da linguagem do recurso

  Decompor os scripts do *RSL*.

- Alocar pedidos de tarefas para os gerenciadores de recurso local

  O gerenciador de recurso local é geralmente um escalonador de tarefas. A interface do gerenciador de recurso é escrita na linguagem Perl, a qual facilmente permite que você crie um novo gerenciador de tarefa para o gerenciador de recurso local, se necessário.

- Envia chamadas de volta "**callbacks**" aos clientes, se necessário

- Recebe os pedidos de status e cancelamento dos clientes

- Envia resultados de saída aos clientes usando *GASS*, se requisitado

6.2.2.5 Global Access to Secondary Storage (GASS)

GRAM usa *GASS* para fornecer o mecanismo de transferência do arquivo de saída dos servidores aos clientes. Algumas APIs são fornecidas sob o protocolo *GSI* para fornecer transferências seguras. Este mecanismo é usado pelo comando **globusrun, gatekeeper,** e o gerenciador de tarefas.

6.2.2.6 Dynamically-Updated Request Online Coallocator (DUROC)

Usando o mecanismo DUROC, os usuários são capazes de submeter tarefas para diferentes gerenciadores de tarefas, diferentes hosts ou diferentes gerenciadores de tarefas do mesmo host (veja Figura 17 abaixo).

O script RSL contém a sintaxe *DUROC* que é analisada gramaticalmente pelo cliente *GRAM* e alocado para diferentes gerenciadores de tarefas.

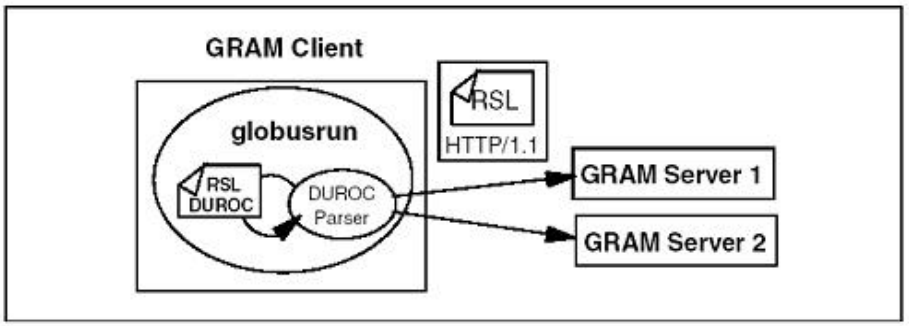

Figura 17 – Visão geral do DUROC [22].

6.2.3 Monitoring and Discovery Service (MDS)

MDS fornece acesso a informação estática e dinâmica de recursos. Basicamente, ele contém os seguintes componentes:

- Grid Resource Information Service (*GRIS*)
- Grid Index Information Service (*GIIS*)
- Provedor de Informação
- Cliente *MDS*

O site oficial do *MDS* é: http://www.globus.org/mds/.

A Figura 18 representa a visão conceitual da interconexão dos componentes do MDS. Como ilustrado, a informação do recurso é obtida pelo provedor de informação e é passada à *GRIS*. Esta registra sua informação local com o *GIIS*, o qual também registra com um outro *GIIS*, e assim por diante. Clientes *MDS* podem pegar informação do recurso diretamente do *GRIS* (para recursos locais) e/ou de um *GIIS* (para recursos em uma ampla grid).

O *MDS* usa *LDAP*, o qual fornece a manutenção descentralizada da informação do recurso.

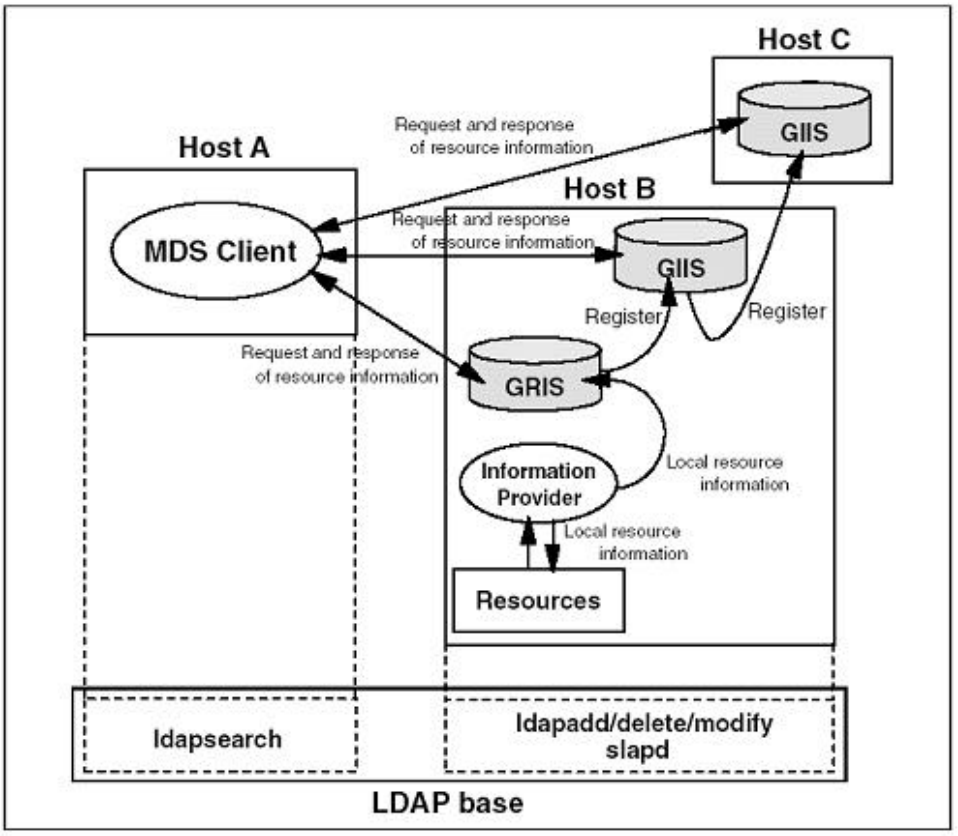

Figura 18 – Visão geral do MDS [22].

6.2.3.1 Informação do Recurso

Informação do recurso contem os objetos gerenciados pelo *MDS*, que representa os recursos dos componentes (estáticos e dinâmicos) como segue:

- Infraestrutura dos componentes

  Por exemplo, o nome do gerenciador de tarefa ou nome da tarefa em execução.

- Recursos do Computador

  Por exemplo, interface de rede, endereço IP, ou tamanho de memória.

6.2.3.2 Grid Resource Information Service (GRIS)

GRIS é o repositório de informação de recurso local, derivado dos provedores de informação. Ele pode registrar sua informação com um *GIIS*, mas o próprio GRIS não recebe pedidos de registro. A informação local mantida pelo *GRIS* é atualizada quando pedida, e guardada por um período de tempo conhecido como tempo de vida "time-to-live" (TTL). Se nenhum pedido de informação é recebido pelo *GRIS*, o tempo de vida da informação terminará e será deletada. Se mais tarde um pedido for requerido, *GRIS* chamará(ão) o provedor(es) da informação relevante para recuperar a informação mais atrasada.

6.2.3.3 Grid Index Information Service (GIIS)

GIIS é o repositório que contém índices de informações de recursos registrados pelo GRIS e outros GIISs. Ele pode ser visto como um servidor de informações de um grid amplo. Ele tem um mecanismo hierárquico, como o DNS, e cada um tem o seu próprio nome. Isto significa que usuários clientes podem especificar o nome do nodo GIIS para procurar pela informação.

6.2.3.4 Provedores de Informação

Os provedores de informação traduzem as propriedades e o status dos recursos locais para o formato definido nos arquivos de configuração e esquema. A medida que é acrescentado seu próprio recurso para ser usado pelo MDS, você necessita criar provedores de informação específicos para traduzir as propriedades e estados para o GRIS.

6.2.3.5 Cliente MDS

O cliente MDS é baseado no comando LDAP cliente, **ldapsearch**. Uma busca pela informação do recurso que você queira no seu ambiente grid é desempenhada pelo seu cliente MDS.

6.2.3.6 MDS Hierárquico

O mecanismo hierárquico MDS é similar ao usado no MDS. GRIS E GIIS, pertencentes às camadas inferiores da hierarquia, registra-se com o GIIS nas camadas superiores. Clientes podem consultar o GIIS para buscar qualquer informação sobre recursos que foram construídos no ambiente grid.

6.2.4 GridFTP

GridFTP provê uma transferência de dados segura e confiável entre nodos do grid. A palavra GridFTP pode ser referido a um protocolo, um servidor ou uma coleção de ferramentas.

6.2.4.1 Protocolo GridFTP

GridFTP é um protocolo cujo objetivo é ser usado em todas as transferências de dados na grid. Ele é baseado no FTP, mas estende o protocolo padrão com capacidades tais como transferência de múltiplos fluxos, alto direcionamento e segurança baseada no Globus. Este protocolo ainda está em nível de testes. Para maiores informações consultar o site: http://www-fp-mcs-anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf

Como o protocolo GridFTP ainda não está completamente definido, Globus Toolkit não suporta toda a coleção de características apresentadas acima. Uma coleção de ferramentas do GridFTP é distribuída pelo Globus como pacotes adicionais. O projeto Globus tem selecionado algumas características e extensões já definidas nos RFCs do IETF e acrescenta poucas características adicionais reunindo requisitos de projetos atuais do grid de dados.

6.2.4.2 Cliente e Servidor GridFTP

Globus Tookit fornece o servidor e o cliente GridFTP, que são implementados pelo **daemon in.ftpd** e pelo comando globus-url-copy, respectivamente. Eles suportam a maioria das características definidas no protocolo GridFTP.

O cliente servidor GridFTP suporta dois tipos de tranferência de arquivos: padrão e "third-party". A transferência de arquivos padrão é quando o cliente envia o arquivo local para uma máquina remota, a qual executa o servidor FTP. Uma visão geral é mostrada na Figura 19.

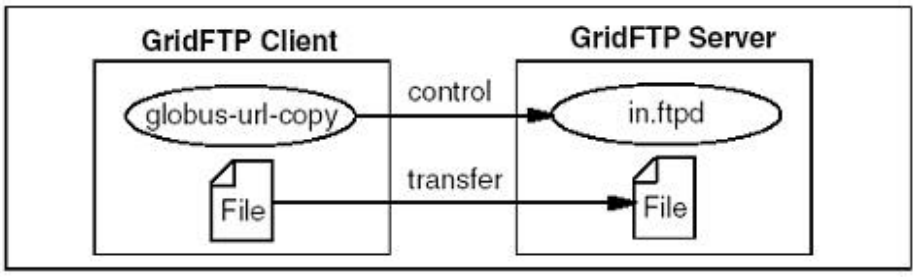

Figura 19 – Transferência de arquivo padrão [22].

Transferência de arquivo "Third-party" acontece quando há um arquivo armazenado remotamente e o cliente deseja copiá-lo para outro servidor remoto, como ilustrado na Figura 20.

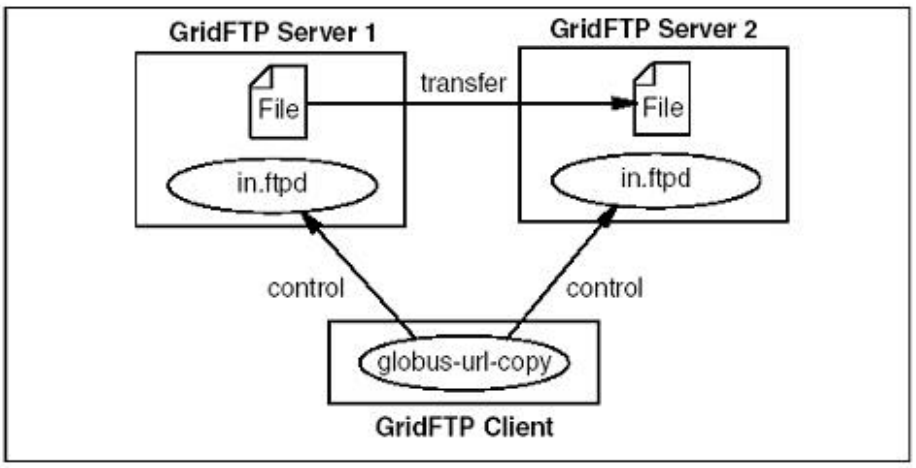

Figura 20 - Transferência de arquivo "Third-party" [22].

6.2.4.3 Ferramentas do GridFTP

Globus Toolkit fornece um conjunto de ferramentas para suportar tipo de transferência de dados do GridFTP. O pacote gsi-ncftp é uma das ferramentas usadas para comunicar com o servidor GridFTP. Este pacote está disponível no seguinte site:

http://www-unix.globus.org/ftppub/gt2/2.0/contrib/globus_gsincftp-0.2.tar.gz

O pacote GASS API faz parte também das ferramentas do GridFTP. Ele é usado pelo GRAM para transferir a saída do arquivo dos servidores aos clientes.

7. Ambiente de Desenvolvimento

Para o desenvolvimento do trabalho foi pesquisada qual ferramenta de construção de grid iria ser utilizada e após esse estudo decidiu-se utilizar o Globus Toolkit, devido as características abaixo:

- Devido a abordagem do Globus diferir das outras arquiteturas nos seguintes aspectos:
    - Pelo seu modelo "bag of services";
    - Pela modularidade dos mecanismos integrantes do kit, permitindo até mesmo a coexistência com outros mecanismos especializados que não fazem parte dele;
    - Suporta mecanismos que coletam informações dos diversos recursos do grid permitindo um maior desempenho no escalonamento das tarefas;
- Disponibiliza uma lista de discussão que integram os diversos pesquisadores e desenvolvedores na área de grid computacional;
- A IBM, como parceiro do projeto Globus, oferece uma variedade de artigos revisados sobre vários temas da área do grid computacional. Permite aos iniciantes adquirir os conceitos essenciais sobre grid e até mesmo construir um simples grid computacional. Além disso, traz a possibilidade dos experts no assunto ampliar seus conhecimentos construindo grids mais complexos, e até mesmo desenvolver suas próprias aplicações no ambiente construído.

Após a revisão bibliográfica sobre grid computacional, incluindo suas várias sub-áreas - Sistemas Distribuídos, Sistemas Paralelos, Clusters, Balanceamento de Carga, Escalonamento de tarefas, etc. - buscou-se a construção efetiva do grid utilizando a ferramenta Globus, baseando-se no RedBook da IBM, "***Introduction to Grid Computing with Globus***" [14]. Para possibilitar essa construção foi necessário a instalação do sistema

operacional Linux RedHat 7.3, porém, não havia recursos estruturais disponíveis. Somente após a disponibilização das máquinas localizadas no LRG – Laboratório de Redes e Gerência – INE/UFSC, foi possível iniciar o experimento.

As maiores dificuldades encontradas foram devidas as configurações de componentes integrantes do kit Globus, tais como: GRAM e GridFTP. Houve também um problema na configuração do componente de segurança do grid, sendo este, primordial para o correto funcionamento do ambiente.

Devido a esses contratempos não foi concretizado na prática a conceitualização do grid computacional, ocasionando a impossibilidade da inclusão de recursos heterogêneos neste ambiente e a não conseqüente realização do teste, o qual objetiva demonstrar a possibilidade de um grid computacional ser tão “fácil” e “popular“ como um PC.

8. Conclusão e Trabalhos Futuros

O conceito de Grid é vantajoso em relação aos Sistemas Distribuídos tradicionais, ver anexo A. De fato aparenta mesmo ser o futuro da computação distribuída. Grandes empresas como Sun e IBM já deram os primeiros passos, lançando produtos comerciais para configuração de ambientes de Grid Computacional. No entanto, ainda há muito a ser feito na área, pois a configuração do grid pode afetar muito o desempenho, confiabilidade, e a segurança da infraestrutura computacional de uma organização.

Ao estudar os conceitos básicos de Grid Computacional, surpreendeu quão extenso e abrangente esse novo paradigma computacional agrega. As áreas computacionais que o Grid abrange são muito vastas e altamente promissoras não tanto quanto se imaginou. Como um dos motivos que impulsionaram este estudo foi a diversidade de áreas computacionais abrangidas, pode-se dizer que neste ponto estamos realizados.

A ferramenta utilizada, o Globus, é uma coleção de ferramentas úteis para a construção de um grid. Suas vantagens se resumem em disponibilizar um bom modelo de segurança, com uma provisão hierárquica de coleta de dados sobre a grid, tão bem quanto às facilidades básicas para executar uma simples grid. Globus se desenvolverá com o tempo, através do trabalho de muitas organizações que estão expandindo suas capacidades.

Devido as dificuldades na construção de um simples grid utilizando o Globus, citadas anteriormente, e tendo este como uma ferramenta largamente utilizada em vários projetos no mundo, constatou-se que a sua construção não é tão trivial quanto a de um simples PC. No entanto, era esperado a princípio não ser tão trivial, pois a arquitetura do grid é muito complexa. Atualmente há um crescente aumento nas pesquisas a nível mundial para diminuir essa complexidade, trazendo-a a níveis que possibilitem compará-la com a de um PC dual.

Como proposta de trabalho futuro, poderia–se pensar em projetar e construir aplicações que, instalado nos equipamentos que integram a grid, permita ao usuário obter informações sobre os recursos de todos os equipamentos pertencentes ao grid, possibilitando-no utilizar esses recursos de forma otimizada para uma aplicação específica.

A obtenção dessas informações disponível no grid poderia ser feita através de agentes JOTA [15].

Referências

[1] BERSTIS, Viktors; RedBooks Paper -IBM – *Fundamentals of Grid Computing*, "www.redbooks.ibm.com/redbooks"

[2] DONGARRA, Jack. High Performance Computing Trends, the Grid, and Numerical Algorithms, HPCS 2003, Sherbrooke, Canada, May 2003.

[3] K. Krautr R. Buyya, and M. Maheswaran, A Taxonomy and Survey of Grid Resource Managment Systems for Distributed Computing, International Journal of Software: Practice and Experience (SPE), Wiley Press, New York, USA, May 2002. (to appear).

[4] ALVARENGA, Anderson T. H. , Uma Tese de Dissertação – Um ambiente para processamento paralelo oportunístico na Internet, Universidade de Brasília – Instituto de Ciências Exatas – Departamento de Ciência da Computação, Brasília, 2003.

[5] SILBERCHATZ, Abraham. *Operation System Concepts.* 5 eth Addison-Wesley, November 1998.

[6] TANEMBAUM, Andrew S.. *Sistemas Operacionais Modernos.* Editora Prentice Hall do Brasil, 1995.

[7] FLYNN, Michael J. *Some Computer Organization and their Effectiveness*. IEEE Transactions on Computers, v. 21, p. 948-960, Sep. 1972.

[8] TANEMBAUM, Andrew S. *Sistemas Operacionais Modernos.* Tradução por Nery Machado Filho. Rio de Janeiro: Livros Técnicos e Científicos Editora, 1999. 493p. ISBN 85-216-1165-X. Tradução de: Modern Operating Systems.

[9] DANTAS, Mario; *Tecnologias de Redes de Comunicação e Computadores*, Axcel Books, 2003.

[10] HUGHES-FENCHEL. *A Flexible Clustered Approach to High Availability,* IEEE 1997.

[11] PFISTER, Gregory F. *In Search of Clusters*. 2. ed., Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN 0-13-899709-8.

[12] SHIRAZI, Belrooz A.; HURSON, Ali R.; KAVI, Krishma M. (Eds.) *Scheduling and Load Balancing in Parallel and Distributed Systems.* Los Alamitos, CA, USA: IEEE Computer Society, 1995. 503p. ISBN 0-8186-6587-4

[13] FOSTER, Ian; KESSELMAN, Carl (Eds.). *The Grid: Blueprint for a New Computing Infrastructure.* San Francisco, CA, USA: Morgan Kaufmann Publishers, 1999. 677p. ISBN 1-55860-475-8.

[14] FERREIRA, L.; BERSTIS, V.; ARMSTRONG, J.; KENDZIERSKi, M.; NEUKOETTER, A.; TAKAGI, Masanobu; BING-WO, Richard; A.; MURAKAWA, R.; HERNANDEZ, O.; MAGOWAN, J.; BIEBERSTEIN, N.; – ***Introduction to Grid Computing with Globus,*** RedBooks Paper –IBM www.redbooks.ibm.com/redbooks

[15] LOPES, José Geraldo Rodrigues Campos; RAMOS, Tânia Gomes; Trabalho de Conclusão de Curso – Escalonamento Híbrido de Tarefas Paralelas para Ambientes Heterogêneos com o Uso de Agentes Móveis, Universidade de Brasília – Instituto de Ciências Exatas – Departamento de Ciência da Computação, Brasília, 2002.

[16] LEINBERGER, William; KUMAR, Vipin. *Information Power Grid: the new frontier in parallel computing?* IEEE Concurrency. v. 7, n. 4, p. 75-84, Oct./Dec. 1999.

[17] Rajkumar Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, School of Computer Science and Software Engineering Monash, University, Melbourne, Australia, April 2002, www.redbooks.ibm.com/redbooks

[18] MELO, Alba C. M. A. de. *Sistemas Operacionais Distribuídos*. Brasília: 1999. Notas da Cadeira de Sistemas Operacionais Distribuídos, Departamento de Ciência da Computação, Instituto de Ciências Exatas, Universidade de Brasília.

[19] EL-REWINI, Hesham; LEWIS, Ted G. *Distributed and Parallel Computing*. Greenwich, CT, USA: Manning Publications, 1998. 447p. ISBN 0-13-795592-8.

[20] CULLER, David; SINGH, Jaswinder P.;GUPTA, Anoop. *Parallel Computer Architecture*: a hardware/software approach. San Francisco, CA, USA:Morgan Kaufmann Publishers, 1997. 1100p. ISBN 1-55860-343-3.

[21] THOMPSON, Rick; CLAVENNA, Scott; Light Reading Paper - *Grid Networking*, "www.lightreading.com "

[22] FERREIRA, L.; BERSTIS, V.; ARMSTRONG, J.; KENDZIERSKI, M.; EUKOETTER, A.; MASANOBUTAKAGI; BING-WO, R.; AMIR, A.; MURAKAWA, R.; HERNANDEZ, O.; MAGOWAN, J.; BIEBERSTEIN, N.; RedBooks Paper -IBM – *Introduction to Grid Computing with Globus*, "www.redbooks.ibm.com/redbooks"

ANEXOS

Anexo A – Softwares disponíveis atualmente para o desenvolvimento de grids computacionais

Sistema	Tipo de grid	Organização	Recurso[1]	Escalonamento[2]
AppLes	Grid Computacional (Escalonamento)	Hierárquica	Usa o modelo dos serviços de middleware fornecido pelo Globus, Legion ou NetSolve	Escalonador descentralizado, estimação heurística preditiva, reescalamento online, política centrada no sistema
DataGrid	Data Grid, Grid Computacional	Hierárquica	Modelo de esquema extensível, nomes únicos hierárquicos, sem QoS, rede LDAP de armazenamento, descoberta baseada em consulta distribuída, disseminação periódica.	Escalonadores hierárquico, estimação heurística preditiva, reescalamento online, política centrada no usuário
Condor	Grid Computacional	Plana	Modelo de esquema extensível, nomes únicos híbridos, sem QoS, outra rede de armazenamento, descoberta baseada em consulta centralizada, disseminação periódica.	Escalonador cooperativo ou centralizado
Globus	Grid ToolKit	Células Hierárquicas	Modelo de esquema extensível, nomes únicos hierárquicos, QoS flexível, rede LDAP de armazenamento, descoberta baseada em consulta distribuída, disseminação periódica.	Escalonador hierárquico, política extensível ad-hoc
Javelin	Grid Computacional	Hierárquica	Modelo de esquema extensível, nomes únicos hierárquicos, QoS flexível, outra rede de armazenamento, descoberta baseada em consulta centralizada, disseminação periódica	Escalonador descentralizado, política centrada no sistema
Legion	Grid Computacional	Plana, Hierárquica	Modelo Objeto extensível, nomes únicos gráficos, QoS flexível, modelo de armazenamento a objeto, descoberta baseada em consulta distribuída, disseminação periódica	Escalonador hierárquico, políticas extensíveis de escalonamento ad-hoc

MOL	Grid Computacional	Células Hierárquicas	Modelo de esquema extensível, nomes únicos hierárquicos, sem QoS, modelo de armaz. a objeto, descoberta baseada em consulta distribuída, disseminação periódica	Escalonador descentralizado, políticas extensíveis de escalonamento ad-hoc
NetSolve	Grid Computacional & de Serviços	Hierárquica	Modelo de esquema extensível, nomes únicos hierárquicos, QoS flexível, busca baseada em consulta centralizada, disseminação periódica	Escalonador descentralizado, política centrada no sistema
Nimrod-G	Grid Computacional & de Serviços	Células Hierárquica	Usa o modelo dos serviços de middleware fornecido pelo Globus, Legion ou NetSolve e estende uma aproximação econômica computacional	Escalonador descentralizado, modelos de avaliação preditiva, reescalonamento dirigido a evento, política centrada no sistema
Ninf	Grid Computacional & de Serviços	Hierárquica	Modelo de esquema extensível, nomes únicos relacionais, sem QoS, descoberta de recurso baseada em consulta centralizada, disseminação periódica	Escalonador descentralizado
PUNCH	Grid Computacional & de Serviços	Hierárquica	Modelo de esquema extensível, nomes únicos híbridos, QoS flexível, descoberta em consulta distribuída, disseminação periódica	Escalonador descentralizado, aprendizado da máquina, política orientada em sistema fixo

1 - Recurso: modelo, namespace, QoS, informação, descoberta, disseminação**.**

2 - Escalonamento: organização, estimação de estado, re-escalonamento e política**.**

Anexo B – Vantagens e Desvantagens entre Grids e SDs Tradicionais

Sistemas Distribuídos Tradicionais	Grids
- Baixo Custo - Alta Velocidade - Tolerância a falhas - Gerenciamento centralizado - Único domínio administrativo - Alta Escalabilidade - Potenciais problemas de segurança - Alguns tipos de aplicações simples não podem ser paralelizáveis - Único ponto de processamento	- Baixo Custo - Alta Velocidade - Tolerância a falhas - Gerenciamento descentralizado - Vários domínios administrativos - Alta Escalabilidade - Potenciais problemas de segurança - Alguns tipos de aplicações simples não podem ser paralelizáveis - Vários pontos de processamento compartilhados por diversos

Anexo C – Passos para a construção de um simples grid

Encontra-se no CD disponibilizado no final do trabalho.