Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de computação

Programa de Pós-Graduação em Ciência da computação

"ProGrid: Uma infra-estrutura de suporte a programação paralela em grades computacionais"

Paulo Vicente Capellotto Costa

São Carlos

Maio de 2003

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

C837pg

Costa, Paulo Vicente Capellotto.
ProGrid: uma infra-estrutura de suporte a programação paralela em grades computacionais / Paulo Vicente Capellotto Costa. -- São Carlos : UFSCar, 2006.
88 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2003.

1. Processamento distribuído. 2. Grade computacional. 3. Sistemas distribuídos. 4. Redes de computação - segurança. I. Título.

CDD: 004.36 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

PROGRID: UMA INFRA-ESTRUTURA DE SUPORTE A PROGRAMAÇÃO PARALELA EM GRADES COMPUTACIONAIS

PAULO VICENTE CAPELLOTTO COSTA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Sérgio Donizetti Zorzo
(Orientador - DC/UFSCar)

Prof. Dr. Hélio Crestana Guardia
(Co-orientador - DC/UFSCar)

Prof. Dr. Ronaldo Augusto de Lara Gonçalves
(UEM/Maringá)

São Carlos
Maio/2003

Dedico este trabalho com muito amor
e carinho aos meus pais, Adilson e
Maria José pelo exemplo de vida

Agradecimentos

Primeiramente agradeço a Deus que me presenteou com uma vida maravilhosa, cercada de bons amigos e uma família amorosa, além de saúde e perseverança, que me possibilitaram chegar até aqui!

Agradeço meus familiares, em especial aos meus pais, Adilson Costa e Maria José Capellotto Costa, que me trouxeram ao mundo e me ensinaram a viver! Obrigado por estarem sempre presentes e por terem me apoiado em todos os momentos de minha vida, mesmo quando não concordavam plenamente com minha decisão. Obrigado pelo exemplo de vida, sem o qual não teria ido tão longe.

Agradeço ao Programa de Pós-Graduação da Universidade Federal de São Carlos e ao meu Orientador, Prof. Dr. Sérgio Donizetti Zorzo, por ter me oferecido a chance de cursar o programa e por me apoiar em todos os momentos. À Coordenação de Aperfeiçoamento Passoal de Nível Superior (CAPES), pelo apoio financeiro.

No decorrer deste Mestrado pude perceber o verdadeiro sentido das palavras: Amigo, Apoio e Comprometimento. Não teria chegado até aqui sem a amizade, o apoio e o comprometimento do professor Dr. Hélio Crestana Guardia, que me auxiliou durante todo o trabalho, nos bons e maus momentos, principalmente naqueles onde eu estava quase desistindo de tudo: ele me motivou e me incentivou a continuar. A você Hélio, um agradecimento especial!

Agradeço a amiga Lúcia Specia e ao amigo Eugeni Dodonov e demais colegas com quem convivi. Obrigado aos meus amigos de laboratório que sempre estavam dispostos a ajudar e me proporcionavam boas conversas e discussões (se bem que as vezes o barulho atrapalhava...). Sentirei falta do ambiente e dos laços de amizade que adquiri nesses 2 anos de convivência.

Ao meu ex-orientador Dr. Nilceu Marana, que me ensinou o prazer da pesquisa científica e ao ex-professor Dr. Marcos Antônio Cavenaghi, pela amizade e sugestões oferecidas. Ao ex-professor Dr. Eduardo Morgado, um agradecimento especial pelo conhecimento, pela amizade e pelas oportunidades que me proporcionou e continua proporcionando...

Não posso me esquecer daqueles amigos que indiretamente auxiliaram e estiveram presentes, compartilhando das minhas frustrações e alegrias nesta longa caminhada. Valeu

Juan Falguera, Waldemir, Samuel Charaba, Terence Bonato, Carlos, Marcelo, Rodrigo Lolato, Pascoal, Ellen, Welington e Gustavo Bogas (que apesar de estar sumido, sempre foi um grande amigo). Ahh! Tem o meu “brodinho” Rogério Cerri, minha amiga Midori e minha ex-companheira de iniciação científica Viviane, que sempre se lembram de mim apesar da distância! Também tenho que agradecer meu cunhado Valdir Michels que fez Mestrado na mesma época e compartilhou comigo sua experiência e amizade.

Finalizo com um agradecimento aos demais funcionários do departamento de computação, que muitas vezes são esquecidos, apesar de prestarem serviços que nos são essenciais! Obrigado!

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

O conceito de grade computacional permite o compartilhamento de recursos computacionais em larga escala. Este trabalho apresenta o sistema ProGrid, uma arquitetura para Grades Computacionais, na qual a infra-estrutura de comunicação e o gerenciamento de recursos são usados transparentemente pelas aplicações. Diferentemente de outras grades, este trabalho utilizou uma abordagem baseada em servidores Proxy para realizar os processos adicionais de comunicação e autenticação em nome da aplicação cliente. O propósito deste mecanismo é habilitar a execução de aplicações paralelas em ambientes geograficamente distribuídos interconectados por um canal de comunicação aberto, como a Internet, atendendo os requisitos de segurança desejáveis nas Grades Computacionais. Para alcançar tais objetivos, desenvolveu-se uma arquitetura genérica para o ProGrid , que é dividida em um conjunto de camadas de serviços. Este trabalho focou-se na implementação das camadas responsáveis pela comunicação segura e pelo compartilhamento controlado dos recursos disponíveis.

Palavras chaves: Grade computacional, proxy, comunicação segura, computação distribuída, MPI.

Abstract

The computational Grid concept allows resource sharing in large scale. This work introduces the ProGrid system, an architecture for computational Grids, whose communication and resource management infrastructure is used transparently by the applications. Unlike other grid approaches, this work relies on the use of proxy servers to perform additional communications and authentication procedures on behalf of client applications. The purpose of this mechanism is to enable parallel applications to be executed in geographically distributed environments interlinked by an open communication network, such as the Internet, meeting the security requisites desirable for computational grids. To reach such objectives, a generic architecture for ProGrid was developed, that is divided in a group services layers. This work was focused in the implementation of layers responsible by the secure communication and for the controlled sharing of available resources.

Keywords: Grid computing, proxy, secure communication, distributed computing, MPI.

Capítulo 1

INTRODUÇÃO

As pesquisas e estudos na área de computação distribuída apresentaram grande evolução nos últimos 10 anos, resultando em importantes avanços. As tecnologias conhecidas como grade computacional e computação na Internet prometem mudar a forma de lidar com problemas complexos. Estas tecnologias irão habilitar agregação em larga-escala e o compartilhamento de dados e recursos computacionais além das fronteiras institucionais. O poder destas novas tecnologias efetivamente transformará áreas científicas [1].

A evolução dos computadores aliada à pesquisa em sistemas distribuídos gerou tecnologias nunca antes imaginadas. Em 15 anos de desenvolvimento, a velocidade dos computadores foi incrementada por um fator de aproximadamente um milhão [1]. Apesar dos sistemas computacionais e dos recursos a eles associados terem evoluído rapidamente na última década, ainda estão longe do necessário para algumas classes de problemas. O Laboratório Europeu de Partículas Físicas (CERN) [2], por exemplo, deverá produzir em 2005 vários *petabytes* de informação por ano – cerca de um milhão de vezes a capacidade de armazenamento de um computador comum atual. A análise desses dados provavelmente necessitará de um poder computacional equivalente a 20 teraflops [1].

Uma forma de alcançar maior poder computacional é através de arquiteturas paralelas do tipo MIMD, como por exemplo, a utilização de máquinas paralelas e aglomerados. Este tipo de arquitetura paralela vem sendo largamente utilizada juntamente com interfaces de programação paralela. O alto custo de máquinas paralelas é um fator limitante para a utilização da computação distribuída em diversas instituições. Uma solução para suprir as necessidades computacionais de problemas complexos, como o citado anteriormente, é a utilização de aglomerados de computadores.

Um aglomerado de baixo custo pode ser obtido através do acoplamento de PC's comuns, caracterizando o que é comumente chamado de *Network of Workstations (NOW)*. Isso possibilita o reaproveitamento da infra-estrutura existente na construção de "máquinas virtuais" de grande capacidade. Porém, a desvantagem dessa abordagem está na restrição de compartilhamento, ou seja, na dificuldade de compartilhar recursos geograficamente distribuídos de maneira simples, segura e eficiente. Para conseguir melhores resultados é necessário um grande número de máquinas e isso nem sempre pode ser alcançado localmente. Para acoplar os recursos geograficamente distribuídos, utilizando canais de comunicação como a Internet, questões de segurança, controle e gerenciamento devem ser considerados.

Embora a utilização de aglomerado possa prover aumento significativo da capacidade computacional, as questões técnicas e financeiras associadas a esta arquitetura podem limitar sua utilização, já que o número de computadores locais disponíveis é restrito.

Percebe-se claramente a necessidade de criarem-se mecanismos para integrar os recursos disponíveis, para que estes possam ser utilizados de forma compartilhada, auxiliando na otimização das suas alocações, bem como na condução de trabalhos colaborativos entre duas ou mais instituições, geralmente geograficamente distribuídas. Os avanços realizados na área das comunicações, bem como a popularização da Internet são as bases para uma nova abordagem mais descentralizada na busca por maior poder computacional. Segundo Foster [1], o século 20 terminou com mais de 400 milhões de PC's espalhados pelo mundo, sendo que a maioria destes permanece ociosa grande parte do tempo. Grandes instituições têm centenas ou milhares de máquinas nessa situação.

A idéia de aproveitar esse potencial ocioso deu origem à computação na Internet, que resultou em sistemas como: Entropia [3], criado por Scott Kurwski com interesse científico e atingiu em apenas dois anos 30.000 computadores com um poder computacional agregado de mais de 1 Teraflop por segundo. Outro exemplo é o SETI@home [4], criado por David Anderson com o objetivo de analisar dados captados pelo telescópio Arecibo em busca de sinais que indiquem vida extraterrestre. O SETI@home conseguiu atingir a marca de 3 milhões de colaboradores, totalizando um potencial computacional equivalente a 25 Teraflop. Outros exemplos incluem o Parabon [5], que analisa a resposta de pacientes submetidos a quimioterapia e o projeto fightAIDS@Home [6] que procura a cura da Aids, entre outros.

Os exemplos citados deram origem a perspectivas muito mais abrangentes do que as propostas pelo conceito de aglomerado: a possibilidade de interligar e compartilhar recursos computacionais, dados e sensores introduziram o conceito de grades computacionais, que pode ser considerado uma estrutura de protocolos, serviços e aplicações que oferece o suporte necessário ao compartilhamento de recursos em larga escala. Por se tratar de um ambiente distribuído, compartilhado, flexível e dinâmico, os requisitos de segurança são de vital importância. O compartilhamento de informações e recursos de alto valor agregado entre diversas entidades distribuídas resulta na necessidade de mecanismos de autenticação de tais entidades, bem como de mecanismos para comunicação segura e confiável entre as partes envolvidas.

Os exemplos aqui descritos, somados à constatação da viabilidade e das vantagens dos sistemas paralelos distribuídos, motivaram o trabalho aqui apresentado. Nesse sentido, o presente trabalho objetivou a construção de uma infra-estrutura simples, transparente e segura que ofereçesse ao usuário o suporte necessário à programação paralela em um ambiente heterogêneo e distribuído, composto por diversos domínios. Para tanto, utilizou uma arquitetura baseada em servidores Proxy, distribuídos entre os domínios e atuando como um provedor de serviços e controlador dos recursos da grade.

A constituição e o enfoque aqui apresentados não foram verificados em nenhum outro trabalho, propondo-se uma abordagem diferenciada e inovadora, porém sem fugir dos padrões e descobertas já realizados por outros trabalhos da área. Denominado **ProGrid,** o sistema desenvolvido constitui uma infra-estrutura segura de suporte a programação paralela, que amplia o escopo dos aglomerados (descritos no Capítulo 2) utilizando para isso os conceitos da computação em grade (descritos no Capítulo 3), visando oferecer ao usuário uma *"máquina virtual"* segura e de fácil interação.

O **ProGrid** interconecta computadores criando um novo domínio de comunicação e segurança, permitindo que seus usuários utilizem os recursos disponíveis de maneira transparente. A criação dessa máquina virtual é facilitada pela abordagem utilizada, que emprega servidores Proxy que interconectam os domínios do sistema e oferecem os serviços necessários. Tal abordagem é semelhante à apresentada em [7], porém apresenta características diferenciadas e tem uma aplicabilidade mais especifica, valendo-se da simplicidade de instalação e utilização.

Para oferecer suporte à programação paralela utilizou-se o padrão MPI [8]. A escolha da interface de programação paralela MPI, como principal ferramenta suportada pelo sistema, é resultado da sua elevada aceitação na comunidade científica, sendo empregada em projetos reconhecidos, conforme será visto no Capítulo 4. Da mesma forma, optou-se por utilizar a biblioteca de segurança SSL (*Secure Socket Layer*) [9] por dois fatores: reconhecimento da eficiência do modelo pela comunidade científica e capacidade de prover autenticação e criptografia em módulos independentes.

A contribuição científica apresentada nesse trabalho é a introdução de uma abordagem diferenciada no suporte a interface MPI (utilizando servidores Proxy) além do foco na facilidade de criação e utilização da grade. Apesar do enfoque diferenciado, o **ProGrid** procurou manter a compatibilidade com projetos e tendências da área [7] [10]. Outra contribuição apresentada por este trabalho refere-se a politica de segurança desenvolvida, baseada em 3 níveis distintos, na qual cada nível pode utilizar uma tecnologia de segurança diferente. Neste caso, o servidor Proxy é responsável pela interoperabilidade dos sistemas, mapeando os requisitos entre os diferentes níveis.

Os próximos Capítulos estão estruturados da seguinte forma. No Capítulo 2, uma revisão dos paradigmas de computação paralela é apresentado, focando nas características mais relevantes ao presente trabalho. Ainda na parte da revisão, o Capítulo 3 apresenta o conceito de grade computacional, descrevendo sua origem, componentes, arquitetura e os desafios relacionados. Complementando o Capítulo 3 e fechando a revisão, dois projetos altamente reconhecidos são apresentados no Capítulo 4.

O Capítulo 5 constitui a descrição do **ProGrid**, iniciando com a sua contextualização, seguidos pela Modelagem e Estrutura da Arquitetura. Além disso, são apresentados também os detalhes referentes à implementação do sistema. Por fim, uma breve discussão é apresentada no Capítulo 6, avaliando o que foi feito no presente trabalho. Neste capítulo ainda são apresentadas e discutidas sugestões para trabalhos futuros que podem complementar e enriquecer o **ProGrid**.

Capítulo 2

Paradigma de Arquiteturas Paralelas

Esse capítulo faz uma breve revisão dos principais conceitos envolvendo as arquiteturas paralelas, abordando suscintamente desde a origem do computador na década de 1940, passando pela apresentação de algumas taxonomias para classificação de computadores até chegar nos conceitos mais recentes de computação distribuída, como a utilização de aglomerados.

2.1 Contextualização

O uso do computador na nossa sociedade iniciou-se com sua criação em 1945. Na primeira fase da computação, existiam apenas grandes máquinas que apresentavam alto custo, sendo privilégio de poucos. Esses computadores de grande porte ficaram conhecidos como *mainframes*. A criação dos chamados mini-computadores na década de 70 expandiu, ainda que de forma tímida, o uso dos computadores.

A grande revolução aconteceu na década de 1980 com o avanço da tecnologia dos processadores. O desenvolvimento de poderosos microprocessadores aliado à diminuição do seu preço tornou o computador acessível às pessoas, o que era inimaginável 15 anos antes. A nova classe de computadores conhecida como PC, rapidamente ultrapassou o poder dos poderosos mainframes que dominaram a primeira era da informática.

A pesquisa e o desenvolvimento dos sistemas computacionais sempre buscaram o aumento da capacidade computacional em todos os sentidos; velocidade, armazenamento, disponibilidade e eficiência. A demanda por computadores mais rápidos incentivou as pesquisas na área de computação paralela, visto que os avanços feitos na construção de microprocessadores

seguindo a *Lei de Moore* [11] não seriam suficientes para atender certas classes de aplicações como por exemplo, problemas de ordem científica que utilizam algoritmos complexos e grande quantidade de dados.

A *Lei de Moore* diz que o número de transistores dentro dos processadores aproximadamente dobra a cada dois anos, o que significa um aumento de desempenho anual na ordem de 50%. Porém, uma máquina paralela com centenas de processadores pode prover às aplicações o poder computacional que um único processador ainda levará 10 anos para alcançar. Dessa forma, a computação paralela cria uma nova regra de crescimento do poder computacional, relacionada com a capacidade de agregar elementos computacionais.

O processamento paralelo é um método que quebra grandes problemas em componentes menores e tarefas que podem ser executadas em paralelo. O desenvolvimento de duas tecnologias contribuiu para o desenvolvimento do processamento paralelo: *Massively Parallel Processors* (MPP's) e computação distribuída. O MPP consiste na utilização de várias unidades de processamento numa mesma máquina. Já a computação distribuída (Aglomerado) utiliza várias máquinas, interligadas por uma rede de alto desempenho, trabalhando conjuntamente.

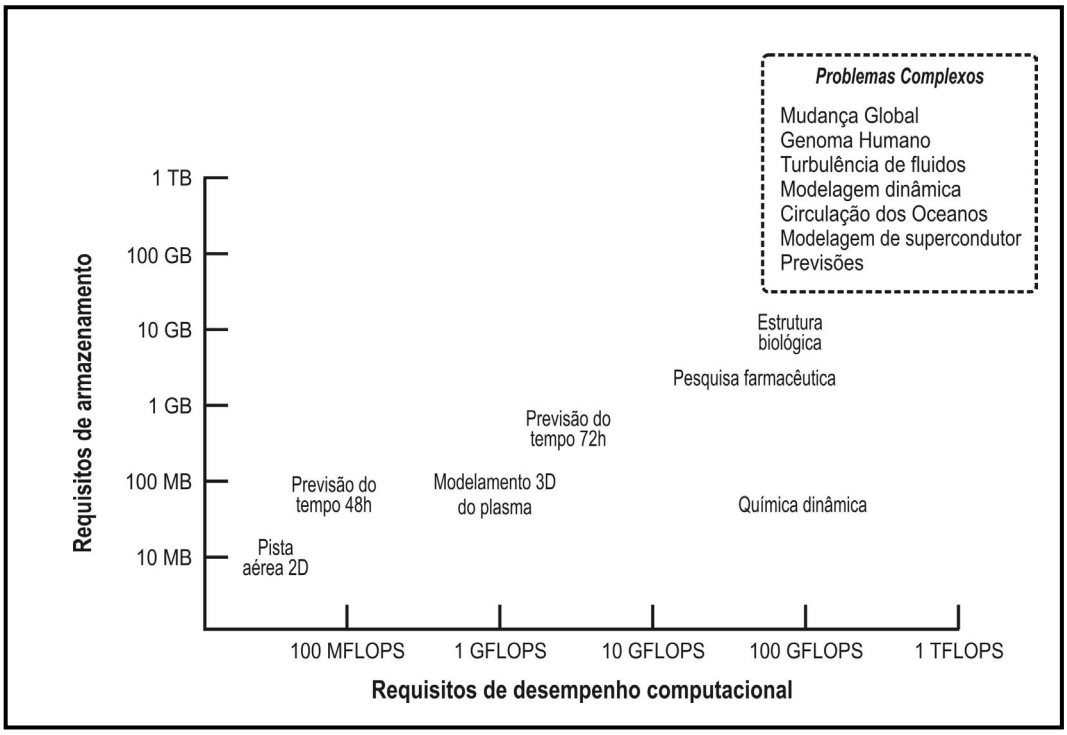

Figura 1: Requisitos de desempenho de problemas complexos [12]

A Figura 1 apresenta uma coleção importante de problemas científicos e de engenharia, posicionados no espaço e definidos pela capacidade de armazenamento e pelo desempenho computacional requerido. No canto superior direito são posicionados os problemas complexos que requerem alta capacidade de computação e armazenamento, exemplificando as aplicações que dependem da computação paralela.

A computação paralela desmembra grandes problemas em subconjuntos de tarefas ou operações que podem ser, de alguma maneira, executadas simultaneamente. Um computador paralelo consiste então de uma coleção de elementos de processamento que se comunicam e cooperam entre si objetivando a solução de problemas complexos mais rapidamente [13]**.** As aplicações que mais se beneficiam da computação paralela são as que estudam fenômenos em geral que não podem ser observados por modelos empíricos [14].

Na computação, o paralelismo pode ser encontrado em vários níveis, desde internamente no microprocessador na execução de instruções, como também agregando, de diversas formas, unidades de processamento. Essa diversidade encontrada na computação paralela levou à criação de diversas taxonomias que buscam classificar as arquiteturas existentes, bem como o grau de paralelismo das aplicações. As próximas seções irão descrever algumas taxonomias de sistemas paralelos, focando principalmente a categoria MIMD. Posteriormente será visto o mecanismo de sincronização de processos baseado no paradigma MTS.

Vale ressaltar que apesar da classificação ter o intuito de dividir as diversas estratégias de paralelismo, existem sistemas híbridos que utilizam uma combinação de duas ou mais estratégias. Conforme focado com mais detalhes no Capítulo 3, o avanço das redes de computadores locais, bem como a Internet, permitiram a criação de uma nova abordagem na construção de sistemas paralelos e na utilização de sistemas e recursos distribuídos.

2.2 Classificação de Arquiteturas de computadores

Segundo a Taxonomia de Flynn [15], apresentada em 1966, os computadores digitais podem ser classificados em quatro grupos que relacionam o fluxo de instruções e fluxo de dados para diferenciá-los. A Figura 2 ilustra a taxonomia proposta por Flynn, sendo que a descrição de suas categorias é apresentada a seguir:

1. **SISD *(Single Instruction / Single Data)*:** arquiteturas que realizam apenas operações seriais, onde uma única instrução é executada por vez, sobre um único dado. Tais arquiteturas não apresentam paralelismo;
2. **SIMD *(Single Instruction / Multiple Data)*:** arquiteturas que executam uma única instrução a cada intervalo de tempo, porém sobre um conjunto de dados. Esta arquitetura apresenta paralelismo síncrono;
3. **MISD *(Multiple Instruction / Single Data)*:** arquiteturas capazes de realizar múltiplas instruções a cada intervalo de tempo, porém sobre um único dado;
4. **MIMD *(Multiple Instruction / Multiple Data)*:** arquiteturas que operam várias instruções sobre múltiplos dados a cada intervalo de tempo. Esta arquitetura apresenta paralelismo assíncrono;

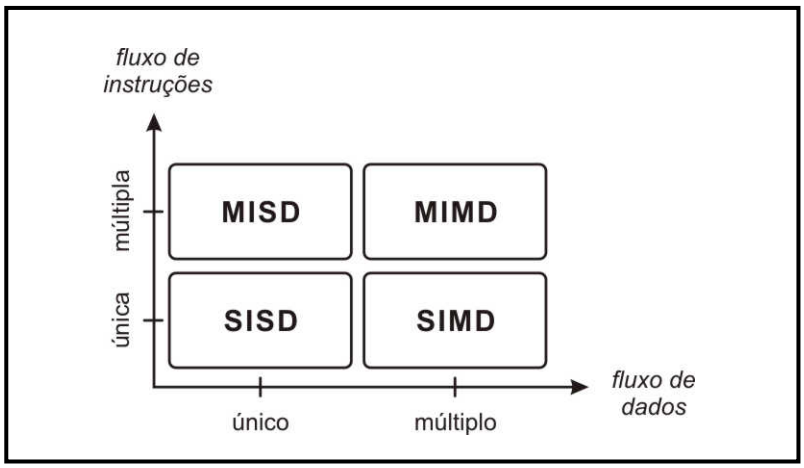

Figura 2: Classificação de sistemas de computadores de acordo com Flynn

O paralelismo síncrono é resultado da utilização de apenas um fluxo de controle *(thread)* de controle. Dessa forma, uma unidade de controle executa o programa enquanto outras estruturas mais simples são responsáveis por executar os comandos. O sincronismo neste caso é intrínseco, pois existe apenas um único fluxo de controle. Em contraste, temos o paralelismo assíncrono, onde diversos fluxos de controle são aceitas. Dessa forma, cada processador executa um programa individualmente, sendo que a troca de dados entre tais processadores deve ser feita explicitamente para garantir a sincronização. Seguindo este esquema de classificação, podemos dividir a classe MIMD em duas subclasses de acordo com o mecanismo de interconexão de seus processadores:

MIMD

1. **Fracamente Acoplados:** tais sistemas são constituídos por elementos de processamento com capacidade de E/S e uma grande memória local, onde as instruções e os dados são comumente acessados. Como tais elementos computacionais são interconectados por uma rede, as sincronizações entre os processos são feitas através do sistema de transferência de mensagens (MTS – *Message Transfer System*). O grau de acoplamento resultante é dado pela topologia de comunicação entre os nós. A largura de banda dos sistemas fracamente acoplados pode ser um limitante para sistemas de tempo real e sistemas que demandam elevado grau de troca de mensagens. Neste caso, sistemas fortemente acoplados devem ser empregados;

2. **Fortemente acoplados:** tais sistemas comunicam-se através de uma memória global compartilhada. Assim, a velocidade da comunicação entre os processos é dada pela largura de banda da memória. Sistemas de arquitetura fortemente acoplada podem tolerar alto grau de interação entre as tarefas sem comprometer significantemente o desempenho [16].

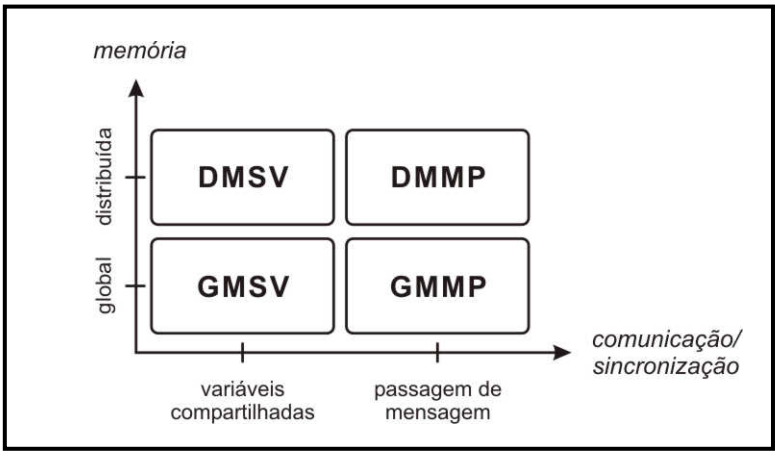

Figura 3: Classificação da arquitetura MIMD

Como a classe de sistemas MIMD apresenta grande diversidade, um novo tipo de classificação foi proposto [17] baseado na estrutura de memória (global / distribuída) e de sincronização (variáveis compartilhadas / passagem de mensagem). A Figura 3 apresenta uma representação dessa classificação listada a seguir:

- **GMSV** (Global Memory / Shared Variables);

- **GMMP** (Global Memory / Message Passing);
- **DMSV** (Distributed Memory / Shared Variables);
- **DMMP** (Distributed Memory / Message Passing);

Relacionando as classes apresentadas em função do acoplamento tem-se que GMSV e GMMP são fortemente acopladas e DMSV e DMMP são fracamente acopladas. Muitas outras classificações existem, porém as apresentadas nessa seção são as mais difundidas.

2.3 Grau de Paralelismo

O grau de paralelismo pode ser diferenciado pelo nível de abstração presente. Nos níveis inferiores o paralelismo é de granularidade fina, enquanto nos níveis superiores é de granularidade grossa. Cada nível está inserido em um diferente aspecto da programação paralela. No nível de procedimentos encontra-se o paralelismo assíncrono com granularidade grossa. Em contrapartida, no nível de expressão encontra-se o paralelismo síncrono com granularidade fina. A Tabela 1 apresenta a classificação do grau de paralelismo em níveis [18].

grossa ▲ granularidade ▼ fina

Níveis	***Objeto Executado***	***Arquitetura***
Nível de Programa	Programa, Tarefa	Sistema Operacional Multitarefa
Nível de Procedimento	Processo	MIMD
Nível de Expressão	Instrução	SIMD
Nível de bits	dentro da Instrução	Computador de von Neumann

Tabela 1: Níveis de Paralelismo

Nível de Programa: Neste grau de paralelismo, programas completos executam simultaneamente ou em intervalos de tempo *(time-sliced)*, conforme ilustrado na Figura 4. Cada programa recebe uma fatia de tempo do sistema operacional (escalonador) de acordo com sua prioridade. Dessa forma, os processos utilizam o processador por um período de tempo e retornam à fila de processos. Neste caso, o paralelismo no processamento não é real, mas sim uma alternativa que permite a execução de diversas operações (dando a impressão de

paralelismo). Esta arquitetura é encontrada em sistemas operacionais multitarefa e não requer *hardware paralelo*.

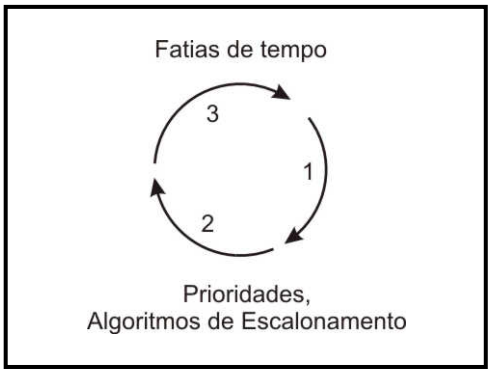

Figura 4: Paralelismo no nível de programa

Nível de Procedimento: Neste nível, diversas seções do mesmo programa são executadas em paralelo, utilizando arquiteturas que disponibilizam mais de uma unidade de processamento. Um exemplo desse nível de paralelismo são as arquiteturas do tipo MIMD. Assim, os problemas são divididos em partes (equivalentes a procedimentos) que podem ser tratadas de maneira independente ou com baixo nível de comunicação. Neste caso, o paralelismo é real, pois cada unidade de processamento recebe uma parte do problema (processo) e todas/várias são executadas simultaneamente, caso haja vários processadores. Este grau de paralelismo pode ser observado na Figura 5.

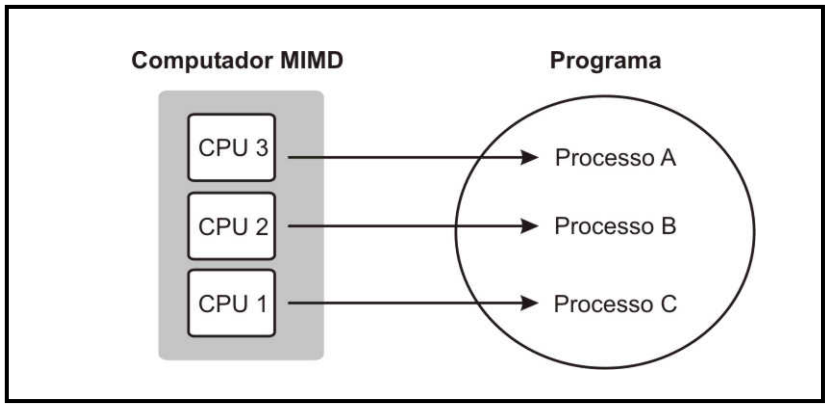

Figura 5: Paralelismo no nível de procedimento

Nível de expressão: Este tipo de paralelismo tira proveito do paralelismo inerente de operações matemáticas ou repetitivas. Tomando-se como exemplo a soma de duas matrizes n x n, tem-se que cada unidade de processamento pode representar um elemento da matriz

resultante (considerando-se que o número de unidades de processamento seja igual a n x n). Dessa forma, o tempo requerido para tal operação é equivalente ao de uma única soma.

Nível de bits: este nível de paralelismo ocorre dentro das instruções, diretamente nos bits, conforme representado na Figura 6. Neste caso, temos a execução paralela de operações binárias sobre uma palavra no nível de bit. Este tipo de paralelismo é largamente encontrado dentro dos microprocessadores.

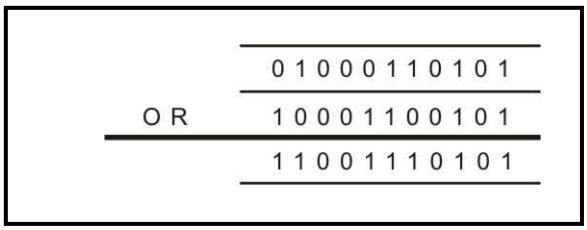

Figura 6: Paralelismo no nível de bits

2.4 Aglomerado

Dentre os vários sistemas possíveis dentro da categoria MIMD, será apresentado nesta seção o conceito de aglomerado, por se tratar de um tópico de relevância para o presente trabalho. As grades computacionais, que são abordadas no Capítulo 3, se originaram em experimentos que buscavam a interconexão de aglomerados geograficamente separados. Para o trabalho em questão, uma arquitetura composta de aglomerados e PC's comuns foi utilizada.

Aglomerado é uma abordagem utilizada para construir máquinas paralelas através da interconexão de vários computadores. A idéia é interligar vários computadores, utilizando uma rede de alta velocidade, para trabalhar cooperativamente no processamento de instruções. Foi explorada pela primeira vez no começo da década de 1980 e atualmente é utilizada na criação de supercomputadores nos laboratórios de pesquisa e indústrias.

Resumidamente, um aglomerado pode ser definido como sendo uma coleção de computadores completos (nós) que são fisicamente interconectados por uma rede de alto desempenho. Cada estação ou nó pode ser um servidor multiprocessado, uma estação de trabalho ou um computador pessoal. Todos os nós do aglomerado devem ser capazes de trabalhar

coletivamente como um único e integrado recurso computacional. Uma arquitetura conceitual de aglomerados é ilustrada na Figura 7 [19].

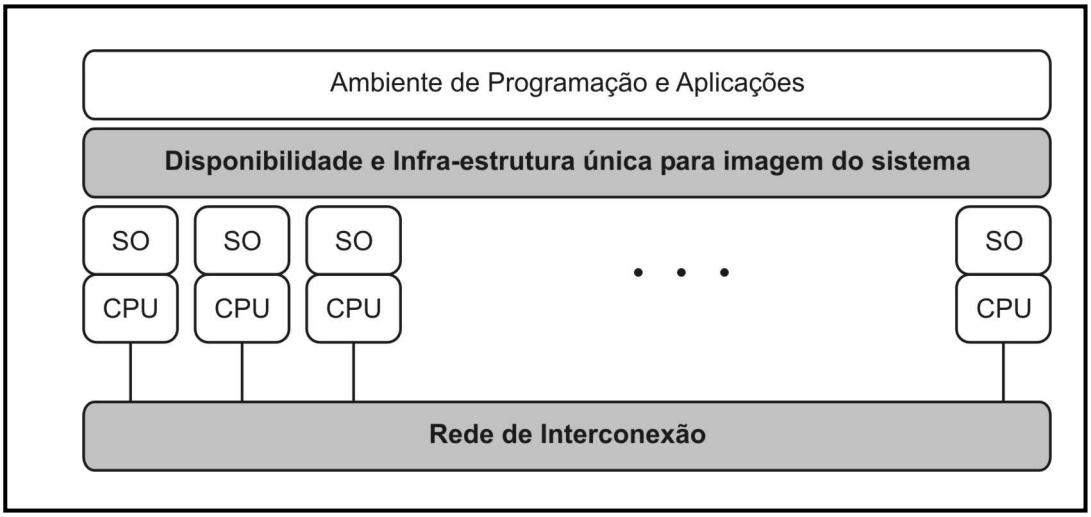

Figura 7: Arquitetura típica de um aglomerado

Foster e Kesselman [20] dizem que o aglomerado é uma coleção de computadores conectados por uma rede de alta velocidade e desenvolvido para ser usado como um computador integrado ou um recurso de processamento de dados. Os dois principais desafios na construção de aglomerados são:

- **Crescimento em escala:** um aglomerado pode envolver dezenas, centenas ou milhares de processadores. O resultado é a necessidade de algoritmos alternativos para gerenciamento de recursos e controle de funções;
- **Redução da integração:** O desejo de construir aglomerado a partir da junção de partes comuns (não-específicas para este propósito), com o intuito de otimizar a infra-estrutura e aproveitar as máquinas já existentes, pode resultar em uma arquitetura menos integrada. Uma implicação disto é a redução de desempenho para certas funções.

Uma classe de aglomerado que ganhou destaque foi a desenvolvida por Thomas Sterling e Don Becker [21] em 1994. O projeto resultou num aglomerado formado por 16 máquinas (Intel DX4) conectadas por uma rede Ethernet. O sucesso desta arquitetura baseia-se em sua concepção, que não necessita de nenhum componente especial, sendo construída utilizando-se o conceito COST *(Commodity off the shelf)*. Esta arquitetura ficou conhecida como *Beowulf*.

Outra característica que diferencia o *Beowulf* de outros aglomerados é que o *Beowulf* utiliza PC's sem teclado, mouse, monitor e placa de vídeo. Usualmente é dedicado a computação paralela e é otimizado para este propósito. Essas características dão ao *Beowulf* uma boa relação de custo/desempenho.

Outra classe de aglomerado muito utilizada é a *Network of Workstations* (NOW) e *Cluster of Workstations* (COW). A diferença destas para o *Beowulf* é que no conceito de NOW e COW, temos PC's completos interligados pela rede. Na maior parte dos casos, constitui a infra-estrutura de computadores de uma instituição e é utilizada para o propósito de aglomerado durante a noite ou nos finais de semana quando estão ociosas.

Podemos perceber que o objetivo do aglomerado é atingir alto poder de processamento através da interconexão de máquinas por uma rede de alta capacidade.

2.5 Bibliotecas de Comunicação Paralela

O Mecanismo de comunicação por passagem de mensagem (MTS) originou-se através da teoria criada no final dos anos 70 por Hoare, sendo colocada em prática pela implementação dos Transputers [22] e de sua linguagem de programação Occam [23] no meio da década de 1980. A idéia original era chamada de *Comunicação de Processos Seqüenciais* [24] e tinha por objetivo que processos independentes e seqüenciais, executando em paralelo, poderiam comunicar-se utilizando mensagens de maneira a resolver problemas paralelos.

Atualmente, o MTS é um paradigma largamente utilizado para comunicação em certas arquiteturas de máquinas paralelas, principalmente nas que utilizam memória distribuída. Um dos atrativos de tal paradigma é sua elevada portabilidade, podendo ser utilizado em diversas arquiteturas sem alteração do código da aplicação. Os aplicativos baseados no paradigma MTS podem ser executados em arquiteturas de multiprocessadores com memória distribuída, em redes de computadores e até em combinações deste. A intenção é que os mesmos códigos de passagem de mensagem possam ser executados numa variedade de máquinas, desde de que as mesmas disponham da biblioteca de passagem de mensagem. Nesse contexto, alguns ajustes são necessários para tirar o maior proveito das características de cada sistema [25].

Existem duas bibliotecas populares de passagem de mensagem que são freqüentemente utilizadas: PVM (*Parallel Virtual Machine*) [26] e MPI (*Message Passing Interface*) [8]. Ambas as bibliotecas provêem uma interface de programação (API) que suporta passagem de mensagem.

Do ponto de vista histórico, a biblioteca PVM apareceu primeiro e foi desenvolvida para operar em redes de estações (para criar uma Máquina Virtual Paralela). Desde então tem sido adaptada para muitos supercomputadores paralelos (distribuídos e de memória compartilhada).

O MPI é um padrão suportado por muitos fabricantes (IBM, HP, HITACHI, SUN e Cray). A biblioteca MPI apresenta mais funcionalidades que o PVM e, além disso, é resultado de um vasto estudo de pesquisa, que envolveu diversas organizações que tinham o intuito de criar um padrão para programação paralela distribuída.

2.5.1 PVM

A *Parallel Virtual Machine* (PVM) provê um *framework* no qual aplicativos paralelos podem ser desenvolvidos de maneira eficiente e direta. Utilizando o PVM, uma coleção de sistemas de computadores heterogêneos pode ser visto como uma única máquina virtual paralela. De maneira transparente, o PVM negocia rotas das mensagens, conversão de dados e sincronismo entre computadores de arquiteturas incompatíveis.

O modelo computacional do PVM é simples, porém muito geral e acomoda uma grande variedade de estruturas. A interface de programação é simples, permitindo que estruturas de programas simples sejam implementadas de maneira intuitiva. O usuário escreve a aplicação como uma coleção de tarefas cooperativas. As tarefas acessam os recursos do PVM utilizando uma biblioteca padrão de rotinas. Estas rotinas permitem a iniciação e o encerramento de tarefas ao longo das máquinas da rede, bem como a comunicação e a sincronização entre tarefas.

As tarefas podem ter controle arbitrário, ou seja, em aplicações concorrentes, uma tarefa tem a capacidade de iniciar ou parar outras tarefas e ainda adicionar ou retirar computadores da

"máquina virtual". Qualquer processo pode manter comunicação ou sincronização com qualquer outro processo.

O PVM contêm ainda uma característica de tolerância a falhas que se torna importante à medida que a máquina virtual cresce. A habilidade de construir aplicações que podem continuar funcionando mesmo quando uma máquina ou tarefa falha é desejável na computação distribuída heterogênea [25].

2.5.2 MPI

A Interface de Passagem de Mensagem (MPI - *Message Passing Interface*) foi desenvolvida para ser um padrão no desenvolvimento de aplicativos paralelos que utilizam o paradigma MTS, oferecendo os recursos necessários para implementação de programas paralelos. O MPI apresenta a vantagem de ter sido desenvolvido a partir do estudo de diversos sistemas de passagem de mensagem, refletindo as características mais atrativas de cada um deles [8].

O objetivo era estabelecer um padrão de passagem de mensagem prático, portável, eficiente e flexível. A versão 1.0 do padrão foi lançada em 5 de maio de 1994. Atualmente existem várias implementações do MPI, incluindo as que são de domínio público, como por exemplo o MPICH [27].

As principais características do MPI são:

- Apresenta um grande conjunto de rotinas de comunicação ponto-a-ponto para comunicação entre processos pertencentes ao mesmo grupo de processos;
- Possui um grande conjunto de rotinas de comunicação coletiva para comunicação entre grupos de processos;
- Utiliza o conceito de comunicador que garante uso seguro das bibliotecas de comunicação;
- Segue o conceito de topologias de comunicação, podendo ser portado para diversas topologias sem intervir na funcionalidade;

Existem várias pesquisas sobre comparações entre os dois modelos de passagem de mensagem [28]. Algumas vantagens do MPI frente ao PVM são listadas abaixo:

- O MPI tem mais de uma versão pública disponível e de qualidade: o MPI dispõe de versões públicas como LAM [29] e o MPICH, entre outras;
- O MPI tem comunicação assíncrona: a operação de envio e recepção imediata pode sobrepor-se à computação;
- O MPI gerencia eficientemente os *buffers* de mensagem: as mensagens são enviadas e recebidas de estruturas de dados dos usuários, e não dos *buffers* da biblioteca de comunicação, como no caso do PVM. Em certos casos, a buferização pode ser completamente evitada;
- O MPI é altamente portável: as aplicações podem ser compiladas e executadas em diversos sistemas.

A biblioteca MPI contém diversas funções de sincronização que auxiliam o programador a implementar aplicativos paralelos, além de oferecer uma camada de abstração que separa a aplicação paralela da arquitetura paralela utilizada. As principais funcionalidades alcançadas pela biblioteca MPI são:

- **Comunicação Ponto-a-Ponto:** define as funções básicas de envio e recebimento com suporte a operações bloqueante e não-bloqueante;
- **Comunicação Coletiva:** define operações de comunicação coletiva de grupos/processos, como *broadcast* e *gather*, entre outras;
- **Grupos, Contexto e Comunicadores:** define como os grupos de processos e comunicadores são criados e manipulados.

2.6 Considerações

Neste capítulo foram abordados os conceitos referentes à computação distribuída desde os aspectos e arquiteturas de hardwares paralelos, até técnicas e mecanismos utilizados para a programação paralela. Foi dada maior ênfase na classe de arquiteturas do tipo MIMD, pois a maioria dos sistemas paralelos desenvolvidos atualmente se encaixa nesta classe. O projeto

ProGrid oferece suporte a programação paralela utilizando uma implementação do padrão de comunicação MPI.

O aglomerado é uma técnica de arquitetura paralela de grande importância, pois ajudou a popularizar o conceito de computação distribuída, permitindo a montagem de sistemas paralelos com menor custo. Além disso, os esforços na criação de aglomerados com maior poder computacional fomentaram as pesquisas que tinham por objeto interconectar aglomerados, resultando em uma abordagem como a computação em grade. Esta abordagem é discutida no próximo capítulo e constitui a base para o desenvolvimento do **ProGrid**.

Capítulo 3

Computação em Grade

Esse Capítulo apresenta alguns conceitos fundamentais de computação em grade, discutindo os aspectos principais de sua implementação. Destaca também os aspectos e desafios relacionados a segurança de grades computacionais, que constitui uma característica importante em sistemas distribuídos de larga escala.

3.1 A Necessidade da Computação em Grade

Conforme foi apresentado no Capítulo 2, o paradigma da computação paralela vem sendo amplamente utilizado com o intuito de aumentar a capacidade computacional. Existem diversas classes de problema que requerem uma quantidade de recursos superior à oferecida por um único computador. Tais constatações alimentaram as pesquisas na área de computação distribuída nas áreas de software e hardware.

Na área de hardware, verifica-se a busca por máquinas de grande porte, que possam agregar vários processadores, memórias e discos de armazenamento visando o aumento da capacidade computacional oferecida. Já na área de software, os estudos concentram-se na criação de mecanismos para utilizar adequadamente os recursos paralelos disponibilizados seguindo diversas arquiteturas.

Várias tecnologias e arquiteturas de hardware e software foram desenvolvidas, (exemplos podem ser encontrados em [30]), criando máquinas de grande capacidade que reduziram o tempo de processamento de problemas que demorariam anos para serem resolvidos em uma máquina comum.

Apesar do sucesso da computação distribuída, o grande problema relacionado a esta tecnologia recaía sobre seu alto custo de aquisição. Dessa forma, os preços elevados dos computadores paralelos limitavam a sua utilização e apenas poucas empresas possuiam recursos suficientes para adquirir tal tecnologia.

Dessa forma, os pesquisadores empenharam-se para desenvolver computadores paralelos de baixo custo. Nesse contexto surgiu o conceito de aglomerado *Beowulf*, que auxiliado pelo crescente desempenho das redes locais, interconectou computadores comuns (PC's) para construir máquinas paralelas. Esta abordagem alcançou tanto sucesso que o conceito se expandiu e juntamente com o desenvolvimento das redes ultrapassou os limites locais e atingiu uma escala surpreendente.

A grande chama que incentivou as pesquisas de computação distribuída utilizando máquinas comuns foi a possibilidade de criar máquinas paralelas de baixo custo, aproveitando a infra-estrutura existente. Os conceitos aqui descritos uniram-se numa abordagem muito mais ampla e genérica que culminou na computação em grade *(Grid Computing)*.

3.2 Definição de grade computacional

Inicialmente, o conceito de grade computacional originou-se em um projeto que tinha por objetivo interligar supercomputadores geograficamente separados. Esse projeto expandiu-se além das expectativas iniciais, dando origem a perspectivas muito mais abrangentes: a possibilidade de interligar e compartilhar, além dos recursos computacionais, também dados e sensores, dando origem ao que hoje se conhece por grade computacional.

Os primeiros esforços na área de grade computacional começaram como projetos que interconectavam sítios de supercomputadores no final da década de 1980 e início dos anos 90; nesta época, essa abordagem era conhecida como metacomputing [31]. Tipicamente, o objetivo desses projetos iniciais era prover recursos computacionais abrangendo aplicações de alto desempenho [32]. Dois projetos representativos na vanguarda deste tipo de tecnologia foram FAFNER [33] e I-WAY [7].

Estes projetos diferem em muitos aspectos, mas ambos tiveram que superar dificuldades semelhantes, incluindo comunicação, gerenciamento de recursos e manipulação de dados

remotos, para permitir a utilização efetiva e eficaz do ambiente [32]. Um desses projetos será discutido em mais detalhes no capítulo 4.

Segundo Foster [20], o conceito de grade computacional pode ser comparado ao sistema de distribuição de energia elétrica. O sistema elétrico é composto por diversos recursos, interligados por uma rede de distribuição que disponibiliza o potencial elétrico aos consumidores. Da mesma forma, a grade computacional é uma infra-estrutura de hardware e software que interliga a capacidade computacional, disponibilizando seus recursos para serem utilizados de forma compartilhada. O uso cooperativo de redes de alto desempenho, somado à utilização de computadores e software avançados irá disponibilizar o acesso à capacidade avançada de computação, não importando a localização do usuário e do recurso.

Para Buyya [34], a grade computacional habilita o compartilhamento, a seleção e a agregação de uma grande variedade de recursos, incluindo supercomputadores, sistemas de armazenamento, coletores de dados, e dispositivos especiais, que estão geograficamente distribuídos e pertencem a diferentes instituições com o objetivo de tratar de problemas de grande-escala na ciência, indústria e comércio.

3.2.1 Principais Aspectos das grades computacionais

Existem 3 aspectos fundamentais que caracterizam as grades computacionais:

1. **Heterogeneidade *(heterogeneity)*:** uma grade envolve uma multiplicidade de recursos que são heterogêneos por natureza e que podem estar dispersos por numerosos domínios administrativos através de grandes distâncias geográficas;

2. **Escalabilidade *(scalability)*:** uma grade pode crescer de poucos recursos para milhões. Isto levanta o problema da potencial degradação do desempenho à medida que o tamanho da grade cresce. Conseqüentemente, aplicações que requerem um grande número de recursos dispersos geograficamente devem ser projetadas para serem extremamente tolerantes as latências de comunicação/distribuição;

3. **Dinamicidade ou adaptabilidade *(dynamicity or adaptability)*:** em uma grade a falha de um recurso é regra e não exceção. De fato, com tantos recursos, a probabilidade de

algum recurso falhar é naturalmente alta. Os gerenciadores de recursos ou aplicações devem adaptar o seu comportamento dinamicamente a fim de extrair o máximo de desempenho a partir dos recursos e serviços disponíveis.

A relação entre outras arquiteturas de computação distribuída, já apresentada e discutidas no Capítulo anterior e as grades computacionais pode ser vista na Tabela 2.

	Supercomputador	**Aglomerado**	**Grade Computacional**
Componentes	Microprocessadores	PC's, Estações	Supercomputadores, Aglomerados, PC's, sensores, etc...
Rede	Buses, switches	LAN	Internet
Sistema Operacional	Homogêneo	Usualmente Homogêneo	Heterogêneo
Segurança	-	Raramente requerida	Indispensável

Tabela 2: Comparação entre supercomputador, Aglomerado, NOW e grade

Através da tabela apresentada percebe-se o contexto na qual as grades computacionais estão inseridas e o relacionamento desta com outras tecnologias de computação distribuída. Apesar das grades poderem também ser utilizadas com o mesmo objetivo de outras arquiteturas como Supercomputador ou Aglomerados, existe uma grande diferença conceitual entre estas arquiteturas. A arquitetura da grade é mais heterogênea, complexa e composta além de computadores, por outros diferentes tipos de recursos.

3.2.2 Grade Computacional vs. Arquitetura Distribuída/Paralela

As principais diferenças da grade computacional frente às outras arquiteturas de computação distribuídas anteriormente apresentadas são:

- As grades são distribuídas geograficamente, podendo agregar recursos distribuídos em diversos sítios. Essa característica permite um melhor aproveitamento dos recursos disponíveis, bem como expande a aplicabilidade da grade;
- A aplicabilidade das grades computacionais visa além de aplicações de alto desempenho, aplicações colaborativas, sobre demanda, entre outras que são discutidas mais adiante;

- As grades são muito heterogêneas, incluindo uma diversidade de elementos. Os componentes no ambiente da grade englobam qualquer recurso computacional incluindo PC's, supercomputadores e até aglomerados;

Durante a Década de 1980, pesquisadores de diversas áreas se uniram na pesquisa e solução de problemas complexos *(Grand Challenge problems)* [35]. O problema vivenciado na condução de pesquisas em múltiplas áreas do conhecimento, a partir de colaborações dispersas e geograficamente distribuídas, mostrou aos pesquisadores os problemas de coordenação e distribuição neste tipo de esforço.

Enquanto a pesquisa em computação distribuída geralmente é focada na solução de problemas de separação geográfica, a pesquisa na grade computacional é focada na solução de problemas de integração e gerenciamento de software [36].

As principais motivações que levaram à criação da tecnologia grade foram os problemas complexos que necessitavam de alto poder computacional e a necessidade de compartilhamento de recursos na resolução de problemas guiados por trabalho colaborativo. Um exemplo do sucesso da computação distribuída foi o evento ocorrido em Janeiro de 1999, quando a *distributed.net* [37] utilizou a computação distribuída para quebrar um desafio da RSA [38] (um algoritmo de criptografia) em menos de 24 horas. Apesar da massiva utilização da grade com intuito de aproveitamento do poder computacional ocioso, esta não é a única classe de aplicação para esta tecnologia [20]. Outras classes de aplicação são descritas na próxima seção.

3.3 Aplicabilidade da grade

Existem vários tipos de aplicações que podem utilizar as características e vantagens da arquitetura de grade computacional para atingir alto desempenho e eficiência. No campo da pesquisa e desenvolvimento das ciências, a grade computacional tem um papel importante, pois permite o compartilhamento de equipamentos especializados de alto custo. Além disso, a capacidade computacional agregada permite aos pesquisadores executar aplicações complexas em um intervalo de tempo menor. No âmbito militar, o poder de processamento associado à grade pode ser utilizado em simulações complexas. No caso de várias empresas que trabalham

juntas num projeto, a criação de uma grade pode auxiliar no compartilhamento de arquivos, recursos e conhecimento. A grade computacional é a tecnologia que irá habilitar a criação de diversos aplicativos em diversas áreas: ciência, negócios, entretenimento, saúde, engenharia, entre outras. As grades computacionais podem ser divididas em 5 grandes classes, de acordo com o tipo de computação utilizada [20]:

- Supercomputação Distribuída;
- Computação de Alto Rendimento;
- Computação Sob Demanda;
- Computação com Elevado Volume de Dados;
- Computação Colaborativa;

Supercomputação Distribuída

Aplicações de supercomputação distribuída usam a grade para agregar recursos computacionais substanciais com a intenção de atacar problemas que não podem ser solucionados por um único sistema. Esta arquitetura pode ser formada por supercomputadores distribuídos ou simplesmente por todas as estações dentro de uma companhia.

Diversas ciências, como a bioinformática, o estudo do genoma, a biologia computacional entre outras podem utilizar grades computacionais para executar simulações em grande-escala e análises complexas que requerem alto poder computacional. Dois exemplos desse tipo de aplicação são descritos a seguir:

- Simulação interativa distribuída (*Distributed Interactive Simulation* – DIS) [39] é uma técnica usada para treinamento e planejamento na área militar. Cenários realísticos podem envolver centenas de milhares de entidades, cada uma delas utilizando um padrão de comportamento complexo;
- Simulação precisa de processos físicos complexos. Associar supercomputadores pode ser interessante nessas situações para superar barreiras e ainda obter de forma qualitativa novos resultados científicos.

Os desafios nesta perspectiva de arquitetura incluem: a necessidade de trabalhar com recursos escassos e caros, a escalabilidade de protocolos e algoritmos para milhares de nós, necessidade de algoritmos tolerantes a latência e a manutenção de altos níveis de desempenho num sistema heterogêneo.

Computação de Alto Rendimento

Na computação de alto rendimento, a grade é utilizada para agendar grandes quantidades de tarefas independentes, com o objetivo de aproveitar o poder computacional ocioso, ou seja, aproveitar os ciclos de CPU de estações que não estão sendo utilizadas naquele instante. Alguns exemplos de aplicações que utilizam a grade com esta finalidade são:

- O sistema Condor da Universidade de Wisconsin utilizado para gerenciar uma de centenas de computadores em universidades e laboratórios ao redor do mundo [40]. Estes recursos tem sido usados para estudos diversos, como simulação molecular de cristais líquidos e desenvolvimento de motores a diesel;
- Entropia [3] e SETI@Home [4] utilizam o potencial ocioso de computadores de usuários da Internet associados ao projeto. Tal arquitetura já permitiu reunir milhões de computadores para trabalhar num único objetivo, criando uma capacidade computacional enorme.

Computação Sob Demanda

Aplicações sob demanda usam a capacidade da grade para adequar recursos de curto prazo que nem sempre apresentam custo-benefício e conveniência para serem utilizados localmente. Esses recursos podem ser de computação, software, repositório de dados, sensores especializados, entre outros. A computação sob demanda difere da supercomputação distribuída no seu enfoque, pois se preocupa com a relação de custo-benefício e conveniência.

Ciências como Física enxergam na grade a possibilidade de compartilhar equipamentos sofisticados, possibilitando agregar recursos de diferentes laboratórios para trabalharem em conjunto. A comunidade de Astronomia vislumbra na grade a possibilidade de coletar,

compartilhar e explorar dados críticos sobre o universo. Exemplos desta classe de aplicação são:

- Virtual Observatory Project [41] realiza pesquisas utilizando a grade com o objetivo de compartilhar vários telescópios, procurando difundir e popularizar sua utilização;
- O NEOS [42] e a rede NetSolve [43] que permitem aos usuários a associação de software e recursos remotos em aplicações, despachando as mesmas para serem processadas em servidores remotos;
- O Sistema desenvolvido pela Corporação Aeroespacial para processamento de dados de satélites meteorológicos utiliza dinamicamente recursos adquiridos de supercomputadores para entregar os resultados do processamento para meteorologistas remotos numa razão muito próxima de tempo real [44].

Os desafios a serem enfrentados pelas aplicações sob demanda derivam da natureza dinâmica dos requerimentos de recursos. Além disso, têm-se as questões de alocação dos recursos, sincronização, gerenciamento de código, configuração, tolerância a falhas, segurança e mecanismos de tarifação.

Computação com grandes Volumes de Dados

Nas aplicações com grandes volumes de dados, o foco é sintetizar novas informações a partir de dados mantidos em repositórios distribuídos, bibliotecas digitais e bancos de dados. Essas aplicações trabalham com grandes bases de dados provenientes de diversas fontes e requerem grande capacidade computacional para o armazenamento, o processamento e a comunicação que são intensivas.

A computação de dados está emergindo como a aplicação primordial nas grades. Na próxima década, acredita-se que haverá diversas fontes de dados: instrumentos científicos, experimentos, sensores, entre outros dispositivos. Neste caso, a grade computacional será utilizada para coletar, armazenar e analisar as informações e dados, bem como sintetizar conhecimento a partir da mineração dos dados [36].

- Futuros experimentos em física energética irão gerar *terabytes* de dados por dia, sendo que no decorrer de um ano irá ultrapassar a fronteira dos *petabytes*. As complexas

buscas realizadas para detectar eventos "interessantes" precisam acessar grandes frações desses dados [45]. Os cientistas colaboradores que irão acessar esses dados encontram-se distribuídos em diversas instituições, assim como os sistemas que irão armazenar tais dados;

- Modernos sistemas de previsão meteorológica fazem uso extensivo de assimilação de dados para incorporar observação de satélites remotos. O processo completo envolve o movimento e processamento de muitos gigabytes de dados.

Os desafios relacionados com a computação com grandes volumes de dados recaem na sincronização e configuração de volumes de dados complexos e elevados através de múltiplos níveis de hierarquia.

Computação Colaborativa

Aplicações colaborativas concentram-se primariamente em habilitar e melhorar a interação entre pessoas. Tais aplicações são freqüentemente estruturadas em termos de espaços virtuais compartilhados. Muitas aplicações colaborativas são focadas no uso compartilhado de recursos computacionais como arquivos de dados, aplicativos e simulações. Em alguns casos englobam várias funcionalidades das classes de aplicações já descritas.

A utilização maciça da tecnologia de informação está trazendo mudanças na metodologia da pesquisa científica, o que vem sendo chamado de *e-Science*. Essa abordagem de ciência envolve a colaboração distribuída através da Internet, utilizando diversas bases de dados e recursos computacionais de grande porte. Dessa forma, *e-Science* pode ser traduzida como a junção entre as principais áreas das ciências e a próxima geração de infra-estrutura (grades computacionais) que irão torná-la possível [36]. Exemplos de aplicações colaborativas são:

- O sistema NICE desenvolvido na Universidade de Illinois de Chicago é voltado ao entretenimento e educação. O sistema permite que crianças participem na criação e gerenciamento de mundos virtuais realísticos [46];
- O projeto Groove [47] para suporte de ambiente colaborativo pela Internet.

Os desafios na criação de estruturas colaborativas recaem na necessidade de interação em tempo real, imposta pela percepção humana e na rica variedade de interações possíveis. À

medida que a infra-estrutura de software concretiza o potencial da grade e, da mesma maneira, a infra-estrutura da grade evolui na direção de prover uma plataforma estável e eficiente, a comunidade de usuários e aplicativos passará a crescer.

3.4 Composição da grade

Uma grade computacional é formada por recursos de hardware e software que buscam a criação de uma única "máquina virtual" que pode ser acessada de maneira uniforme em qualquer ponto da grade. O hardware é formado por diversos recursos computacionais, incluindo PC's, aglomerados, supercomputadores, servidores, sensores, coletores de dados, satélites e dispositivos de armazenamento, entre outros recursos. Esta camada de hardware fornece os recursos que são utilizados pelos usuários da grade de maneira transparente.

O software é responsável por gerenciar a interação entre os recursos e os usuários, formando a *middleware* do sistema. A *middleware* define as funções que habilitam a comunicação em sistemas distribuídos e as ferramentas que melhoram a usabilidade de uma arquitetura construída sobre múltiplas plataformas. Sendo assim, esta camada de software permite o compartilhamento de dados entre sistemas que não se comunicam de maneira simples ou convencional, ou seja, é responsável por intermediar diferentes aplicações. Na grade, a *middleware* é utilizada para esconder a natureza heterogênea e prover aos usuários e aplicações um ambiente homogêneo e simples a partir de um conjunto de interfaces padronizadas para uma variedade de serviços [32].

A *middleware* é dividida em diversas camadas, cada uma delas responsável por fornecer algum tipo de funcionalidade à grade. Algumas camadas constituem serviços que são diretamente acessados pelos usuários. Outras camadas são constituídas de tecnologias e protocolos já definidos e em sua maioria fornece serviços para as camadas adjacentes. Até o momento, não existe nenhuma padronização para o conjunto de camadas (*middleware)* que constitui uma grade, nem para o relacionamento interno desses módulos, porém há esforços na direção da sua criação.

O *Open Grid Service Architecture Working Group* (OGSA WG) [48] vem trabalhando na definição e esquematização do escopo e dos requisitos para os principais serviços das grades

computacionais. O OGSA representa as diversas camadas como serviços. Um serviço é uma entidade que provê alguma funcionalidade através da rede, como: recursos computacionais, recursos de armazenamento, redes, programas e base de dados, entre outros.

Dentro dessa perspectiva, um requisito fundamental em um ambiente heterogêneo como a grade é o mecanismo que habilita a interoperabilidade. Em uma visão orientada a serviços, é possível particionar o problema da interoperabilidade em 2 sub-problemas: definição das interfaces de serviço e a identificação dos protocolos que podem ser usados para invocar uma interface particular. A visão orientada a serviços também possibilita tratar os aspectos referentes a necessidade de mecanismos padronizados na definição de interfaces, na transparência local/remota, na adaptação para serviços do sistema operacional local e na obtenção de serviços semânticos uniformes.

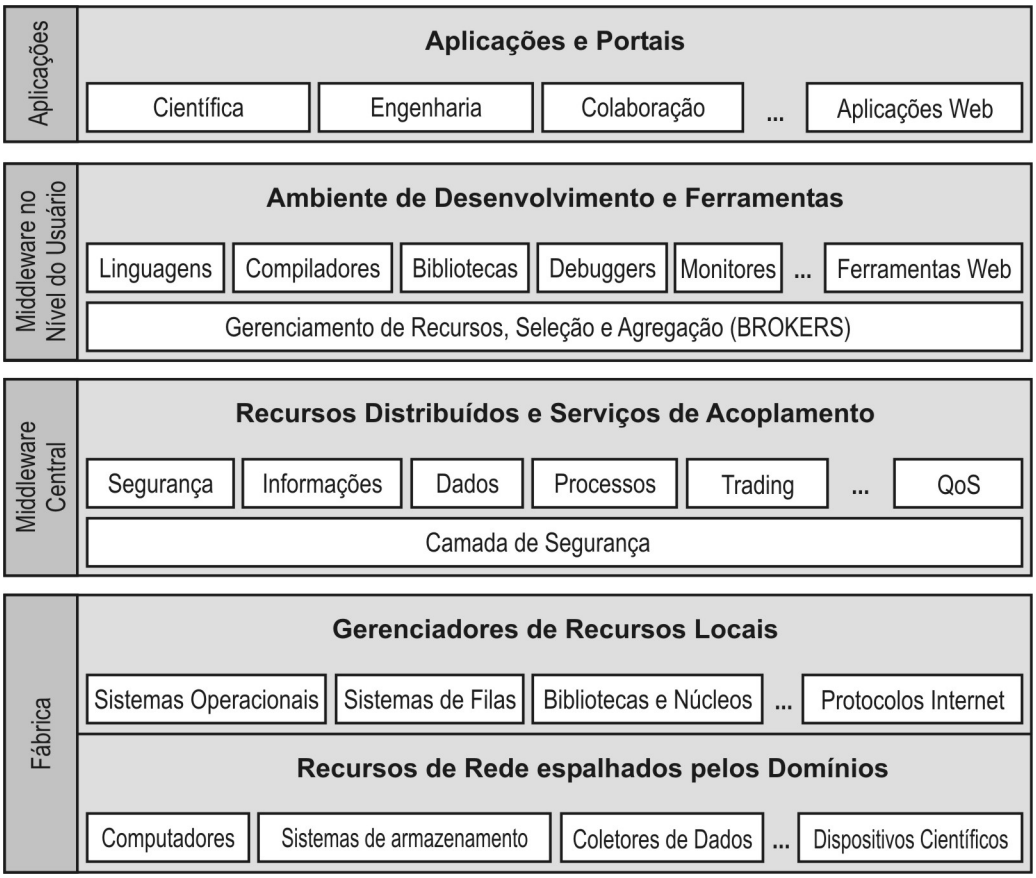

Figura 8: Representação da arquitetura de uma grade computacional Genérica

A Figura 8 ilustra uma arquitetura genérica para grades computacionais seguindo as taxonomias de Foster, Kesselman, et al [48] e também de Johnston [48]. Nela é possível verificar claramente a divisão em camadas resultante da arquitetura na visão orientada a

serviços. Nessa abordagem, a constituição das camadas utiliza o conceito da abstração crescente, que está diretamente ligado ao objetivo de fornecer transparência. No Apêndice A é apresentada uma abordagem mais completa da arquitetura na visão orientada a serviços segundo Johnston [49].

Esta representação genérica da grade computacional inclui diversos tipos de hardware, serviços e tecnologias de interconexão. No entanto, existem grades computacionais mais específicas que foram desenvolvidas focando uma determinada classe de problema. Isto pode ser verificado em projetos como Condor [40] e DataGrid [50] que são discutidos mais adiante. A seguir, a função de cada camada representada na Figura 8 é descrita com maiores detalhes:

- **Aplicações:** aplicações para a grade são desenvolvidas tipicamente usando linguagens e utilitários que facilitam o desenvolvimento de programas paralelos, como MPI ou Nimrod [51]. Portais para a grade oferecem uma interface de interação, onde os usuários podem submeter tarefas e coletar resultados através de recursos remotos via WEB. As interfaces gráficas facilitam a interação do usuário com a grade e são um importante instrumento nesse ambiente distribuído e altamente complexo;

- **Middleware no Nível do Usuário:** nesta camada localizam-se os serviços de acesso aos recursos da grade. Para aproveitar tais recursos, o usuário tem à sua disposição, diversas camadas de serviços, que em conjunto com a Middleware Central são responsáveis por intermediar as suas requisições. Nesta camada encontram-se: ambientes de desenvolvimento de aplicações, ferramentas de programação, recursos de *broker* para gerenciamento dos recursos e agendamento das tarefas para execução em recursos globais;

- **Middleware Central:** esta camada disponibiliza os serviços que são responsáveis por controlar a grade. É responsável por abstrair a complexidade na utilização dos recursos, oferecendo regras e padrões para encontrá-los e acessá-los. Oferece serviços como: gerenciamento remoto de processos, realocação de recursos, controle de armazenamento, segurança, qualidade de serviço e reserva de recursos, entre outros. Esta camada é considerada o coração da grade, pois contém os principais serviços necessários para o funcionamento da grade;

- **Fábrica:** é formada por todos os recursos distribuídos que são acessíveis de qualquer parte da grade. Estes recursos podem ser computadores (como PC's, SMP's, aglomerados) executando uma variedade de sistemas operacionais (como Linux ou Microsoft® Windows) bem como recursos de gerenciamento de sistemas como *Load Sharing Facility* (LSF) [52], dispositivos de armazenamento, banco de dados e equipamentos científicos especiais como telescópios ou sensores;

A visão da grade voltada para serviços divididos em camadas facilita a visualização dos componentes que formam tal ambiente. Do ponto de vista do usuário, a abstração fornecida pelas diversas camadas de serviço simplifica a interação com o ambiente, além de proporcionar transparência para portar os serviços para outras plataformas. Dentre os esforços propostos pelo OGSA, está o mapeamento dos recursos mínimos necessários às grades computacionais, sendo eles:

- **Descoberta de Recursos:** como a grade é um ambiente altamente dinâmico e heterogêneo, seu estado muda com certa freqüência. É então necessário ter um serviço na grade que forneça as informações de todos os recursos disponíveis (incluindo serviços), de forma padronizada. Essas informações auxiliam o usuário a encontrar os recursos necessários. Dada a complexidade da arquitetura e a diversidade de recursos, a camada responsável pela descoberta de recursos é de vital importância à grade, pois fornece a transparência na localização dos recursos aos usuários e aplicativos;

- **Gerenciamento e Agendamento de Recursos:** o gerenciamento de informações como disponibilidade de processador, memória, largura de banda, armazenamento e outros componentes da grade é muito importante. Isto possibilita o agendamento das aplicações que precisam dos recursos disponíveis no ambiente distribuído. Do ponto de vista do usuário, o gerenciamento e agendamento de recursos deve ser transparente. Além disso, é necessário que os serviços de gerenciamento e agendamento possam interagir com os serviços locais, possibilitando a integração com escalonadores de recursos locais;

- **Acesso Uniforme a computação:** para a grade ser considerada uniforme, é necessário haver meios para compor uma estrutura comum para requisições de recursos de computação que seja independente das diferentes plataformas, políticas e

convenções de nomes nos sítios e nas máquinas individuais. Uma requisição de computação também envolve implicitamente a preparação de áreas para arquivos, área de armazenamento dos resultados e área de memória temporária enquanto a requisição está sendo executada. Dessa forma, a grade deve fornecer mecanismos para descrever todas as características de requisição de recursos (ou tarefas) incluindo as características essenciais do ambiente de forma independente das políticas locais a cada sítio. Esta característica facilita o trabalho de migração de processos;

- **Serviços de Informação:** a grade é um ambiente altamente dinâmico, onde a localização e os tipos de serviços disponíveis mudam constantemente. Dessa forma, é necessário prover mecanismos para permitir que as informações a respeito da grade estejam disponíveis e possam ser facilmente obtidas pelos recursos e usuários. Tais mecanismos devem registrar e também obter informações de estruturas, recursos, serviços, status e natureza do ambiente;

- **Segurança e Autenticação:** qualquer sistema distribuído envolve os seguintes aspectos de segurança: confidencialidade, integridade, autenticação e direito de acesso. A segurança dentro da grade é uma questão complexa que requer a interação de diversos recursos, muitas vezes em domínios de segurança distintos, com diferentes políticas de acesso. Dessa forma, mecanismos de segurança devem ser inseridos cuidadosamente para não impactar na usabilidade dos recursos e também não introduzir falhas e brechas no ambiente. Como a segurança é uma questão imperativa em sistemas distribuídos em geral, assim como na grade, ela será tratada em mais detalhes na seção 3.5.

3.5 Requisitos de Segurança

O ambiente da grade computacional objetiva alto grau de compartilhamento e agregação dos recursos que o constituem, formando um único ambiente virtual. A interconexão de tal ambiente muitas vezes utiliza uma rede pública como a Internet. Além disso, a possibilidade de executar aplicativos em outras máquinas, de requisitar dados, adicionar ou remover

recursos, entre outras operações disponíveis nesse ambiente, inserem vulnerabilidades que podem ser exploradas com intenções maléficas. A preocupação aumenta quando as informações e recursos compartilhados no ambiente apresentam alto valor agregado.

Torna-se evidente a necessidade de mecanismos de segurança que suportem as operações realizadas na grade oferecendo a segurança desejada, sem a perda da transparência e do desempenho.

Os desafios de segurança encontrados no ambiente da grade podem ser agrupados em 3 categorias [53]:

- Integração com tecnologias e sistemas existentes;
- Interoperabilidade com diferentes domínios e plataformas;
- Relações de confiança entre as diferentes plataformas e domínios;

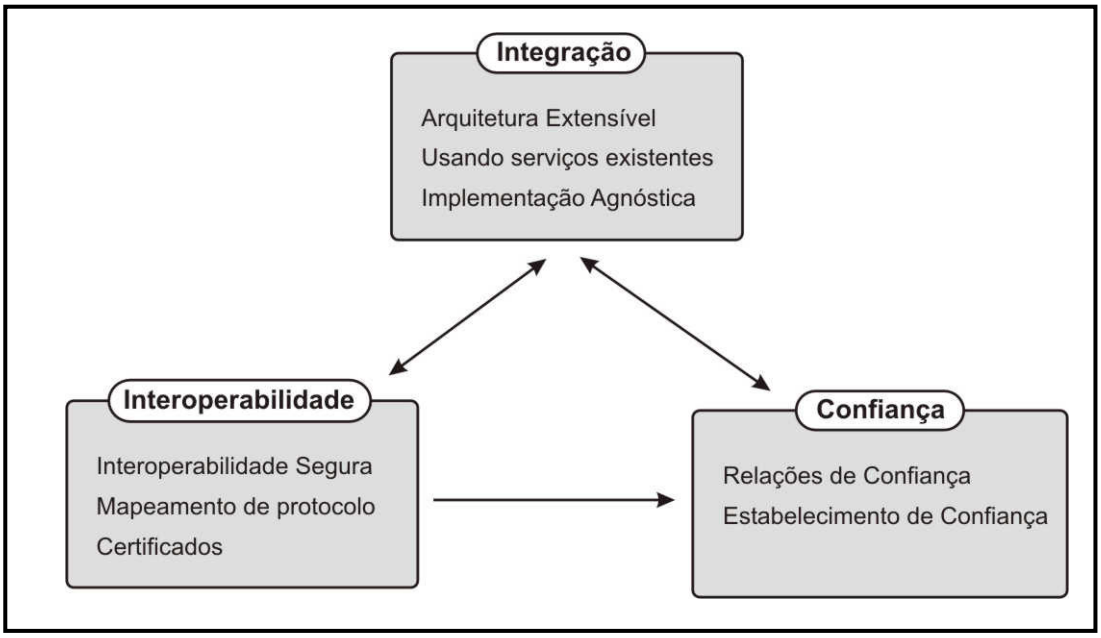

Figura 9: Desafios de segurança das grades computacionais

A relação entre as três categorias referentes aos desafios de segurança é ilustrada na Figura 9. Para citar um exemplo referente à dependência ilustrada pela figura, considere a emissão de credenciais para alcançar interoperabilidade. A criação de tais certificados depende do modelo de confiança estabelecido entre as entidades participantes.

Integração

A solução ideal para alcançar integração dos sistemas seria conseguir estabelecer uma tecnologia de segurança única, que fosse adotada por todas as entidades da grade e que seja responsável por suprir as todas as necessidades de segurança. No entanto, isso não pode ser alcançado facilmente.

Cada domínio dentro da grade possui sistemas próprios de segurança, o que dificulta a adoção de um sistema de segurança único. Uma imposição desse tipo desencorajaria o ingresso e a utilização da grade por parte do usuário. Dessa forma não é aceitável a utilização de um sistema de segurança que venha substituir os sistemas em uso.

Assim, o sistema de segurança da grade deve ser capaz de interagir com os diversos modelos de segurança existentes nos diversos domínios e plataformas. A integração entre os sistemas deve ser capaz de garantir a segurança da grade sem interferir na segurança local já existente.

Interoperabilidade

Os serviços que atravessam diversos domínios precisam estar aptos a interagir uns com os outros. Esta necessidade introduz vários níveis de interoperabilidade:

- **Protocolo:** são necessários mecanismos para permitir a troca de mensagens entre domínios diferentes;
- **Políticas de Segurança:** a interoperabilidade requer que as partes envolvidas sejam capazes de especificar as políticas desejadas para entrar numa comunicação segura. Seguindo essas políticas, os participantes podem estabelecer um contexto de segurança através de autenticação mútua e relações de confiança;
- **Identidade:** são necessários mecanismos para identificar um usuário nos vários domínios. No entanto, este requerimento vai além do estabelecimento de relações de onfiança e utilização de mecanismos como autenticação (*Kerberos* [54], *X.509 Certificates* [55], entre outros). O ponto mais importante a ser considerado são os mecanismos responsáveis por mapear as identidades e credenciais entre os múltiplos domínios da grade;

Confiança

Os serviços utilizados na grade podem estar dispersos por múltiplos domínios. Para garantir a transparência e a segurança na utilização de tais recursos, relações de confiança entre os domínios que compõem a grade são indispensáveis.

Os desafios referentes ao estabelecimento de relações de confiança são agravados no ambiente da grade devido à sua natureza dinâmica. Isso requer o suporte a ambientes dinâmicos e a serviços transitórios criados e gerenciados por usuários [53]. Tais serviços são criados por usuários que desejam executar uma tarefa referente a uma requisição específica que pode envolver diversos serviços e recursos localizados em diversos domínios. Os desafios associados à criação de serviços transitórios pelos usuários incluem:

- **Identidade e Autorização:** deve ser possível controlar a identidade sobre a qual o serviço transitório é executado;
- **Políticas de utilização:** os usuários devem estabelecer as políticas dos serviços transitórios, estabelecendo as ações que estes podem executar a partir das próprias permissões;
- **Delegação:** serviços transitórios podem ter a necessidade de executar ações em nome do usuário sem a intervenção direta do mesmo.

Para melhor entender os serviços de segurança que podem estar disponíveis nas grades, os tópicos de segurança discutidos serão traduzidos em requisitos de segurança específicos para a grade [53] [56] [57]:

- **Isolamento:** os componentes que compõem o sistema devem ser capazes de se isolar de outras partes que apresentarem brechas de segurança. Esta característica é importante em sistemas de larga-escala, pois quanto maior o número de componentes no sistema, mais complexa e difícil é a tarefa de controle;
- **Autenticação única:** a grade não deve exigir que o usuário tenha que se autenticar a cada serviço solicitado, ou seja, uma única autenticação deve ser requerida para que o usuário tenha acesso a todos os recursos da grade. Isso não é uma tarefa simples, pois a grade é um ambiente heterogêneo que pode englobar diversos domínios com políticas de acesso diferenciadas. Além disso, cada recurso pode requerer um

processo de autenticação diferente. Dessa forma, para atender a característica de autenticação única, a grade deve ser capaz de mapear a autenticação do usuário (em sua maioria através de certificados) nos diversos domínios e recursos;

- **Delegação:** a execução de aplicações distribuídas na grade pode requerer o acesso a base de dados ou outros recursos. Dada a complexidade das tarefas encontradas na grade, pode se tornar impossível para o usuário interagir diretamente com todos os sistemas possivelmente envolvidos. Dessa forma, os processos devem ser capazes de acessar os recursos da grade usando os direitos de acesso do usuário a que pertencem. A identidade do usuário é transferida ao processo/sistema remoto sem a intervenção direta do usuário: isso garante a execução completa das tarefas, complementando a característica de autenticação única. Para atender a essa característica, um "*proxy certificate*" é normalmente criado e armazenado num local acessível aos processos iniciados pelo usuário. O "*proxy certificate*" contém a chave privada do usuário, juntamente com seu certificado, possibilitando a autenticação e autorização;

- **Integração com a segurança local:** o sistema de segurança da grade não deve substituir o sistema de segurança local. Ao invés disso, o sistema de segurança da grade deve ser capaz de interagir com as diversas soluções de segurança empregadas nos diversos domínios;

- **Privacidade e Integridade:** como os recursos que compõem o sistema muitas vezes são interconectados por uma rede pública, métodos de confidencialidade como criptografia, devem ser utilizados para garantir a comunicação segura entre as partes. Além disso, o ambiente deve garantir a integridade das mensagens;

- **Controle de Acesso:** o sistema de segurança deve dispor de um eficiente sistema que engloba políticas de autenticação e autorização. O ambiente da grade requer um sistema de controle de acesso mais complexo que os utilizados nos sistemas tradicionais. Na grade, um usuário deve ser capaz de repassar seus direitos de acesso a processos, que devem ser capazes de acessar outros recursos sem a intervenção do usuário;

- **Identidade:** o ambiente deve ser capaz de confirmar a identidade dos componentes que interagem na grade;
- **Detecção e Recuperação:** o ambiente deve suportar mecanismos para detectar intrusão e abuso de recursos, sendo capaz de recuperar-se rapidamente no caso de falhas;
- **Portabilidade:** a grade é composta por uma diversidade de sistemas. Assim, os mecanismos de segurança devem ser construídos sobre padrões comuns, abertos e modulares para que possam ser facilmente integrados e portados entre os componentes que compõe a grade.
- **Renovação de Credenciais:** em alguns cenários, uma tarefa iniciada por um usuário pode ter um ciclo de vida maior que a prevista pelo usuário inicialmente. Neste caso, o usuário deve ser notificado, sendo capaz de renovar a credencial permitindo que a tarefa seja completada;
- **Autorização:** permite o acesso aos serviços baseados numa política de autorização (quem pode acessar o serviço / sobre quais condições) vinculada a cada serviço;
- **Troca de políticas de segurança:** permite que usuários e provedores de recursos possam dinamicamente trocar informações sobre políticas de segurança para negociar e estabelecer um contexto seguro entre eles;
- **Passagem por Firewall:** a principal barreira para a grade computacional é a existência de *firewalls*. Dessa forma, o modelo de segurança deve considerar a passagem por *firewalls* sem o comprometimento das políticas de segurança locais já estabelecidas;

Os requisitos descritos acima são gerais, sendo que aplicações específicas podem necessitar de outros serviços de segurança não relacionados aqui.

3.6 Considerações

Neste capítulo o conceito de grade computacional foi apresentado, mostrando desde as motivações que culminaram na sua criação, até os aspectos teóricos referentes a este conceito.

Foram apresentadas as características mais comuns das grades, seguindo as constatações realizadas por diversos trabalhos da área, como os serviços e desafios encontrados, divisão dos serviços em camadas, questões relacionadas com a segurança e categorização dos diversos tipos de grade segundo sua aplicabilidade. Os conceitos aqui apresentados serviram de base para a modelagem da arquitetura do **ProGrid**, que será apresentada no capítulo 5.

No próximo capítulo, serão apresentados dois projetos de referência na área de grades computacionais. Eles foram escolhidos pela sua relevância no mundo científico e pelo alto grau de relacionamento com a abordagem apresentada pelo **ProGrid**, salvo as devidas proporções.

Capítulo 4

Implementações de grades

Este capítulo apresenta a revisão de dois projetos relacionados a grades computacionais. O primeiro deles é o I-Way, pioneiro na construção de grades e referência como o primeiro esforço na construção de tal sistema. O segundo deles é o Globus, que baseado nos resultados do I-Way, expandiu-se, tornando-se a base no esforço de padronização da arquitetura das grades computacionais.

4.1 Introdução

O conceito de grades computacionais ganhou força e reconhecimento a partir da implementação de um projeto de teste que demonstrou na prática as questões, os serviços, as vantagens, as desvantagens e as dificuldades na criação de grades. Desde então, diversos projetos foram criados tanto com intuito de pesquisar os conceitos e de construir sistemas de infra-estrutura e suporte, como projetos visando uma classe específica de aplicação, alguns até com fins comerciais.

A pesquisa na área da computação em grade continua crescendo e como foi apresentado no Capítulo anterior, aumenta-se também o esforço na criação de padrões que facilitem a integração e comunicação entre as diversas implementações. A Tabela 3 ilustra alguns exemplos de grades computacionais.

Este Capítulo tem por objetivo descrever alguns projetos de referência na área de grades computacionais, possibilitando um melhor entendimento dos conceitos apresentados no Capítulo anterior, bem como a sua demonstração prática de aplicação. A seção 4.2 descreve em detalhes o *Information Wide Área Year* (I-Way), projeto pioneiro na construção de grades computacionais, servindo de base e impulsionando a criação de diversos outros projetos. Um desses projetos é o Globus [58], apresentado na seção 4.3. Este último é tido como referência

na pesquisa e estudo de grades computacionais, sendo que o esforço de padronização proposto pelo OGSA [48] é baseado nos resultados alcançados pelo Globus.

Categoria	Projeto	Organização	Descrição
Sistemas de grade Integrados	NetSolve	U. Tenneesse	Sistema de programação e execução para acesso a bibliotecas de alto desempenho e recursos transparentemente
	Unicore	Alemanha	Ambiente baseado em Java para acesso a supercomputadores
Core Middleware	Globus	ANL e ISI	Prove um ambiente uniforme e seguro para acessar recursos computacionais remotos.
	Legion	U. of Virginia	Sistema operacional de grade que prove acesso transparente aos recursos distribuídos.
Nível do usuário (escalonadores)	Condor-G	U. of Wisconsin	Amplo sistema de processamento de tarefas.
Nível do usuário (Ambiente de Programação)	MPICH-G	Northern Ilinois U.	Uma implementação do MPI para o Globus
Aplicabilidade e Aplicativos que são guiados pela grade	European Data Grid	CERN	Física energética, biologia
	Earth Grid System	LLNL, ANL & NCAR	Modelagem do Clima

Tabela 3: Outros projetos de na área de grades computacionais

4.2 I-Way

Em poucas palavras, o I-Way era uma rede ATM que interconectava supercomputadores, sistemas massivos de armazenamento e dispositivos avançados de visualização.

O projeto I-Way é considerado como o primeiro esforço na construção de grades computacionais. Desenvolvido como projeto experimental para demonstração na Conferência de Supercomputação (SC95), o I-Way conseguiu agregar o poder computacional de 17 sítios. Entre as aplicações desenvolvidas para o I-Way, encontra-se uma infra-estrutura para prover acesso, garantir a segurança e coordenar os recursos, entre outras tarefas, que conjuntamente constituíram uma grade computacional. Os resultados propiciados por este projeto, motivaram

a pesquisa na construção de infra-estruturas para suportar ambientes semelhantes. Desde então, diversos projetos surgiram com o intuito de explorar e construir infra-estruturas para grades computacionais.

Segundo os pesquisadores responsáveis pelo desenvolvimento do I-Way, os esforços anteriores produziram apenas partes da solução [7]. Soluções como, CORBA [59], Condor [40], Nimrod [51] e Prospero [60] endereçavam problemas de localizar e/ou acessar recursos distribuídos; sistemas como AFS [61] e Truffles [62] endereçava problemas de compartilhamento de dados distribuídos; ferramentas como Nexus [63], MPI e PVM endereçavam problemas de acoplamento de recursos computacionais distribuídos; e tecnologias de rede de baixo nível como ATM (Asynchronous Transfer Mode) prometiam comunicação da ordem de gigabit/sec. Entretanto, pouco trabalho havia sido feito para integrar estas soluções de maneira a satisfazer os requisitos de escalabilidade, desempenho, funcionalidade, confiança e segurança de aplicações reais, distribuídas e de alto desempenho entre redes de grande escala.

O objetivo do projeto I-Way foi prover um *testbed (protótipo)* em grande escala, onde aplicações de alto desempenho e geograficamente distribuídas pudessem ser desenvolvidas.

A parte central do I-Way era o software de gerenciamento e o ambiente de programação, chamado I-Soft. A arquitetura do sistema utilizava máquinas dedicadas no qual o I-Soft era executado. Tais máquinas tinham que estar presentes em cada sítio e eram conhecidas como pontos de acesso (I-POP – Point of Presence). O I-POP era responsável por prover autenticação uniforme, reserva de recursos, criação de processos, e funções de comunicação ao longo dos recursos do I-Way.

Diversas aplicações foram selecionadas para executar no ambiente provido pelo I-Way. Tais aplicações dividiam-se em 3 grupos:

1. Grupo de aplicações que criavam ambientes virtuais acoplando diversos supercomputadores, sistemas de dados e instrumentos científicos remotos;
2. Aplicações que acoplavam vários supercomputadores geograficamente distribuídos para atender problemas complexos que são muito grandes e demandariam muito tempo se executados isoladamente em um único supercomputador;

3. Aplicações que acoplavam diferentes ambientes virtuais permitindo a usuários distribuídos, interagir entre si e com simulações de supercomputadores.

4.2.1 Máquinas de Ponto de Acesso

O I-POP era uma estação dedicada, acessada através da Internet e que operava dentro do domínio do sítio. Esta característica fazia com que o I-POP fosse localizado atrás do Firewall, dentro do domínio de segurança do sítio. Por esta razão e também por manter comunicação com diversos sítios, a segurança era uma questão imperativa. Um conjunto padrão de softwares era executado no I-POP:

- Uma interface ATM permitia o monitoramento e o gerenciamento de switches de rede ATM;
- Um sistema de gerenciamento permitia a comunicação com outras máquinas do sítio para alocação de recursos, inicialização de processos, entre outras funcionalidades;
- O sistema de arquivos distribuídos AFS era utilizado como repositório de software e informações de status;

4.2.2 Escalonador

Para ser capaz de localizar recursos computacionais em um ambiente heterogêneo e distribuído e de acordo com critérios pré-estabelecidos, o I-Way utilizava um sistema de escalonamento de recursos. Este sistema expressava as requisições primeiramente de forma abstrata, que eram mapeadas e negociadas com os escalonadores locais, sendo um por cada I-POP (para cada sítio). Dessa forma havia um escalonador global e diversos escalonadores locais. O escalonador global armazenava as diversas filas e tabelas representando o estado das diferentes máquinas virtuais, e era responsável por alocar tempo nessas máquinas de forma abstrata. O scheduler global comunicava-se com o scheduler local para requisitar as operações de:

- **Alocação de Recursos:** este tipo de requisição permitia ao escalonador local a execução de operações para tornar o recurso requerido disponível para o usuário;
- **Criação de Processos:** esta requisição pedia ao escalonador local para criar processos em um processador específico, de acordo com a requisição do usuário;
- **Desalocação de Recursos:** esta requisição permitia ao escalonador local realizar operações necessárias para terminar o acesso do usuário ao recurso, desabilitando seu acesso, matando processos ou apagando arquivos temporários;

4.2.3 Segurança

A segurança do I-Way era focada em prover um ambiente uniforme de autenticação. Problemas relacionados com autorização, contas, privacidade e integridade dos dados do usuário não foram considerados.

Dessa forma, a abordagem de segurança utilizada no I-Way foi dividida em duas partes: autenticação do usuário com o I-POP e autenticação com os recursos.

A autenticação junto à entidade I-POP era feita utilizando-se um cliente *Telnet* modificado para utilizar a autenticação e criptografia baseada no Kerberos. A autenticação junto aos recursos era feita pelo escalonador que guardava as informações de qual *userID* devia ser utilizado em cada sítio para cada usuário do I-Way. Dessa formal, o escalonador atuava também como um *"authentication proxy"*, pois realiza autenticações futuras em nome do usuário. O usuário realizava uma única autenticação no sistema, sendo que as autenticações posteriores eram feitas pelo escalonador em nome do usuário.

4.2.4 Ferramentas de Programação Paralela

O I-Way utilizava diversas bibliotecas para programação paralela, procurando oferecer ao usuário, ferramentas para facilitar a utilização dos recursos distribuídos, através de uma camada de abstração que escondia a complexidade do sistema. Entre as bibliotecas utilizadas,

encontra-se a CAVEcomm [64] e a MPICH [27], uma implementação do padrão MPI que foi estendida para utilizar os mecanismos da biblioteca Nexus.

4.2.5 Sistema de Arquivos

Em um ambiente heterogêneo e distribuído como o I-Way, os usuários necessitavam de mecanismos para facilitar a troca de dados e informações entre os diversos sítios. Para facilitar a troca de dados e informações entre as entidades do I-Way, o sistema de arquivos AFS foi utilizado como um repositório para o software I-Way, e também para a manutenção das informações de status do escalonador. A célula AFS podia ser acessada somente através do I-POP, pois nem todas as estações suportavam o AFS e nas estações que suportavam, problemas de autenticação dificultavam o acesso.

4.3 Globus

O Globus é um esforço que reúne múltiplas instituições na pesquisa e construção de grades computacionais. Constituiu-se a partir do I-Way, onde vários pesquisadores que participaram na criação deste último continuaram seus esforços na construção do Globus. Além da área de pesquisa, o Globus disponibiliza as ferramentas e serviços desenvolvidos. Assim, a parte central do projeto Globus é uma "Toolkit" que define e disponibiliza os serviços e as capacidades básicas requeridas nas grades.

Assim como o I-Way, o Globus também tornou-se referência na pesquisa e construção de grades. O I-Way por ser um precursor no desenvolvimento de tal arquitetura e o Globus por ter mapeado os requerimentos e iniciado o desenvolvimento de diversos protocolos e serviços da grade, sendo a base para os esforços de padronização. Os principais serviços disponibilizados pela Toolkit Globus são apresentados na Tabela 4 [65].

Serviço	Nome	Descrição
Gerenciamento de recursos	GRAM	Alocação de recursos e gerenciamento de processos
Comunicação	Nexus	Serviços de comunicação *unicast* e *multicast*
Segurança	GSI	Autenticação e serviços de segurança relacionados
Informação	MDS	Acesso distribuído a informações de estrutura e estado
Status	HBM	Monitora a "saúde" e o status dos componentes do sistema
Acesso Remoto aos Dados	GASS	Acesso remoto aos dados através interfaces paralelas e seqüenciais
Gerenciamento de executáveis	GEM	Construção, armazenamento e localização de executáveis

Tabela 4: Principais serviços disponibilizados pelo Globus

4.3.1 Gerenciamento de Recursos

O *Globus Resource Allocation Manager* (GRAM) é um dos principais serviços da grade, sendo responsável pela alocação, monitoramento e controle dos recursos computacionais. Este serviço foi implementado seguindo uma arquitetura hierárquica. Existem diversos GRAM, sendo cada um deles responsáveis por um determinado número de recursos computacionais (computadores, aglomerados, bases de dados, entre outros). As requisições são expressas através de uma linguagem de especificação de recursos criada para este fim. A *Resource Specification Language* (RSL) transforma as requisições abstratas progressivamente em requerimentos mais específicos até um conjunto de recursos ser identificado.

Os diversos GRAM são controlados por co-alocadores de recursos, responsáveis por dividir os GRAM em grupos. No topo da hierarquia, encontram-se os gerenciadores de recursos que são específicos para cada aplicação. Este nível interage com os co-alocadores, que devem repassar as informações progressivamente até serem totalmente traduzidas chegando ao conjunto de recursos desejado.

4.3.2 Comunicação

Os serviços de comunicação no Globus utilizam a biblioteca Nexus [63]. A camada Nexus define uma API de comunicação de baixo nível que opera sobre diversos protocolos de rede e oferece suporte a diversas bibliotecas e linguagens de comunicação de alto nível, como Passagem de Mensagem (MPI), chamada remota de procedimentos (CC++), entre outras [65]. A comunicação entre dois pontos ocorre em 2 fases: na primeira delas, um canal de comunicação é criado entre as entidades que desejam realizar a comunicação; na outra fase, a comunicação é iniciada com um pedido de serviço remoto (remote service request – RSR) feito pelo ponto de origem. Então, o ponto de origem envia os dados ao ponto destino e associa-os aos processos que requisitaram a comunicação.

A grande vantagem é a possibilidade de se estabelecer diversos canais de comunicação, sendo que cada um deles apresenta uma configuração diferenciada. As várias maneiras nas quais os pontos de origem e destino podem ser interconectados, possibilita a criação de redes complexas.

4.3.3 Segurança

O *Globus Security Infrastructure* (GSI) é responsável pela autenticação das entidades ao sistema. A infra-estrutura de segurança inicialmente desenvolvida é focada na autenticação, pois esta funcionalidade é a base para os outros serviços de segurança.

Um dos grandes desafios na construção de sistemas de segurança para grades é referente à heterogeneidade do ambiente da grade, resultante da necessidade de integração de diversos domínios com políticas de segurança distintas. Além disso, deseja-se obter um sistema onde o usuário deve-se autenticar uma única vez, recebendo uma credencial que permite o acesso aos recursos da grade. Dessa forma, o Globus definiu uma política de segurança para ser aplicada na grade. Esta política [56] identifica quais componentes devem ser protegidos e define as operações de segurança em termos de algoritmos abstratos.

O sistema de segurança desenvolvido apresenta serviços globais e locais. Para lidar com a heterogeneidade da grade e possibilitar a autenticação única por parte do usuário, a Identidade global do usuário é mapeada em identidades de usuários locais em cada recurso. Dessa forma,

utilizando este tipo de mapeamento, não é necessária a alteração das políticas locais de segurança.

4.3.4 Informação

O Metacomputing Directory Service (MDS) é o serviço responsável por centralizar e prover um ambiente de informações rico e dinâmico, onde as informações sobre o sistema estão sempre disponíveis.

O MDS armazena e torna disponíveis informações referentes ao tipo de arquitetura, versão do sistema operacional, quantidade de memória do recurso, largura de banda, latência da rede e protocolos de comunicação disponíveis. Para isso, provê um conjunto de ferramentas e APIs para descobrimento, publicação e acesso a informação sobre a estrutura e estado da grade. A representação das informações é feita utilizando o padrão LDAP, ou seja, em cada domínio existe um servidor responsável por armazenar as informações locais. O MDS é então o agrupamento das informações coletadas nos servidores LDAP locais.

4.3.5 Status

O *Heartbeat Monitor* (HBM) é um serviço que oferece mecanismos simples para monitoramento do status de processos distribuídos. O sistema é composto pela interface do cliente e pela API coletora de dados. Através da interface, o cliente pode registrar um processo com o serviço HBM, que então espera receber *heartbeats* (sinais) provindos do processo. A API coletora de dados permite a outros processos obterem informações referentes ao status dos processos registrados. Tais informações podem ser utilizadas na implementação de mecanismos de detecção e recuperação de falhas. O HBM é utilizado para monitorar os serviços do próprio Globus, como o GRAM e o MDS.

4.3.6 Acesso remoto aos dados

O Global Access to Secondary Storage (GASS) é o sistema utilizado para permitir a programas que utilizam a biblioteca de E/S padrão do C, abrir, ler e escrever em arquivos localizados em computadores remotos, sem a necessidade de alteração das diretivas de leitura e escrita originalmente utilizadas.

4.4 Considerações

Foram descritos neste capítulo dois projetos de referência no âmbito de grades computacionais. Os projetos aqui apresentados oferecem uma idéia da complexidade e da quantidade de serviços e recursos necessários na construção de grades computacionais genéricas. A construção de grades envolve o esforço em diferentes áreas da computação. Alguns dos serviços aqui apresentados, bem como alguns modelos e entidades, serviram de base para a definição do **ProGrid** que será detalhado no Capítulo 5.

Capítulo 5

ProGrid – Implementação de uma infra-estrutura de grade computacional

> *A visão de grades computacionais é absolutamente crítica para avanços futuros na ciência a na sociedade, mas a visão sozinha não criará a grade. A promissora capacidade da grade deve ser guiada por pesquisas, desenvolvimento e utilização durante a próxima década [36].*

Este capítulo apresenta o sistema **ProGrid**: uma infra-estrutura de suporte a programação paralela em grades computacionais, sendo centrada no padrão MPI. O sistema **ProGrid** não está restrito a este trabalho, que apesar de ter sido o passo inicial, é apenas uma parte na construção de uma grade computacional robusta e completa. Dentre os serviços propostos na modelagem, foram implementadas algumas camadas objetivando o suporte à programação paralela e à segurança da grade. Este Capítulo inicia-se com o Projeto do Sistema, passa pela Modelagem e Estrutura da Arquitetura e termina na descrição dos detalhes referentes a implementação.

5.1 Projeto do Sistema

O presente trabalho foi desenvolvido a partir da análise do ambiente de grades computacionais e dos conceitos de computação distribuída, em especial da programação paralela. A análise, a pesquisa e o estudo desses dois tópicos conduziram à constatação da

necessidade de uma infra-estrutura simples e apropriada que fosse capaz de aproveitar o poder e a facilidade de ferramentas de programação paralela juntamente com as vantagens disponibilizadas por um ambiente heterogêneo, dinâmico e multi-institucional como o ambiente das grades computacionais.

Os conceitos abordados nos capítulos 2 e 3 somados às conclusões iniciais obtidas através da análise de outros trabalhos, conduziram a uma delimitação inicial do problema.

Inicialmente foram levantadas todas as funcionalidades desejadas no sistema e a relação entre as mesmas. As principais dificuldades encontradas nesta fase foram delimitar o problema e as funcionalidades presentes no sistema. A diversidade e a abrangência do conceito de grades somadas à vontade de construir um sistema robusto, levou a planejamentos que fugiram do escopo e do tempo disponíveis para a realização deste trabalho. Dessa forma, entre as diversas idéias e possibilidades, optou-se por focar no suporte à interface MPI, buscando simplicidade, transparência e oferecendo também mecanismos de segurança. Além disso, outras funcionalidades e serviços foram previstos para serem incorporados em trabalhos futuros, garantindo a continuidade e a expansão do sistema, que foi batizado de **ProGrid**.

Este esforço inicial culminou em uma arquitetura em camadas utilizada como base para o desenvolvimento. Essa arquitetura é mostrada e discutida nas próximas seções.

A fase principal e a responsável por apresentar o maior nível de dificuldade, correspondeu à implementação do **ProGrid**. A transposição da arquitetura proposta juntamente com suas delimitações para a realidade, expôs alguns detalhes que não foram tratados de forma apropriada, requerendo a modificação/adaptação de partes do projeto.

- Estudo dos mecanismos utilizados na programação paralela para ambientes distribuídos;
- Levantamento das funcionalidades previstas numa grade;
- Elaboração de uma arquitetura de software para atender aos requisitos;
- Implementação dos serviços e módulos previstos;

5.2 Modelagem da Arquitetura

A modelagem da infra-estrutura desejada baseou-se em alguns conceitos e padrões que começam a ser definidos [48], porém com um certo grau de inovação. Como resultado, atingiu-se uma abordagem diferenciada para atender aos requisitos e objetivos propostos, sem distanciar-se dos conceitos atualmente utilizados. A definição da arquitetura proposta neste trabalho remete-se à composição e a arquitetura das redes de computadores para delimitar o ambiente da grade computacional e seus componentes.

As redes de computadores são interligadas por *gateways/roteadores* que atuam como pontos de interconexão. Estes pontos de acesso fornecem diversos serviços que permitem a comunicação de computadores localizados em redes distintas, não importando a arquitetura ou o protocolo de rede local utilizado. Seguindo este raciocínio e considerando-se que a grade é uma delimitação virtual de um conjunto de computadores, percebe-se que existe uma estreita relação entre a estrutura de ambos. Dessa forma, a partir da observação da estruturação das redes de computadores, este trabalho propõe a utilização de uma arquitetura baseada em servidor Proxy. Tal entidade atua de forma semelhante ao *gateway*, servindo de ponto de interconexão entre os sítios que compõem a grade computacional.

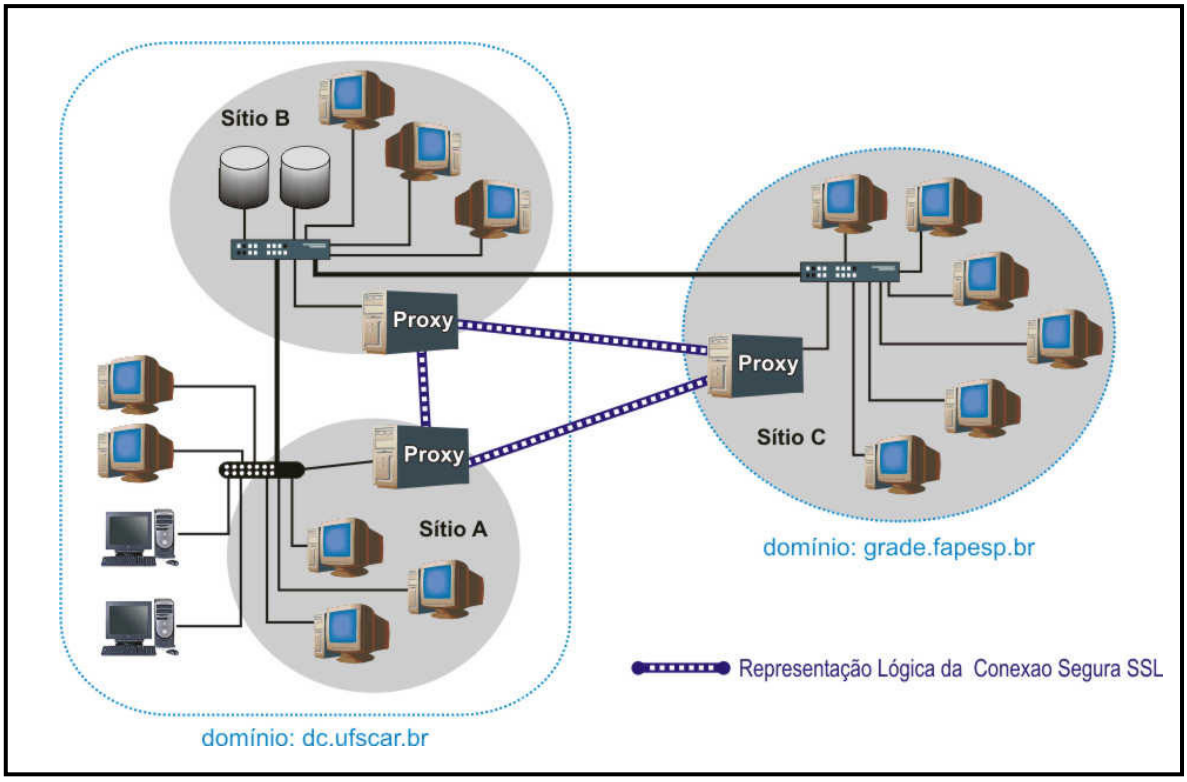

*Figura 10: Visão geral da arquitetura do **ProGrid***

Utilizando-se essa abordagem, buscou-se uma arquitetura simples, transparente para o usuário e que seja facilmente incorporada na rede, minimizando a necessidade de alteração da infra-estrutura. O controle e as funcionalidades da grade são inseridos na borda dos sítios e não individualmente em cada nó. A Figura 10 ilustra a relação entre os componentes que compõem a arquitetura.

Analisando-se a Figura 10, percebe-se que a presente arquitetura não estabelece uma relação com os domínios existentes, sendo que o Proxy estabelece uma nova delimitação virtual. Cada Proxy estabelece um sítio de administração dentro da grade, sendo responsável pelos usuários e recursos inseridos nesse sítio. A junção de vários sítios forma a grade computacional que para o usuário corresponde a uma única máquina virtual.

O servidor Proxy é responsável por aglutinar os serviços fornecidos na grade computacional visando uma arquitetura simples, porém eficiente. Diferente de outras abordagens [58] e [66] que objetivam grades computacionais de grande porte, voltadas para ambientes heterogêneos com diferentes classes de aplicação, a abordagem aqui apresentada limita-se a oferecer um conjunto restrito de funcionalidades para uma classe específica de problema.

Pode-se estabelecer uma relação do presente trabalho com o projeto I-WAY [7] que é referenciado como um marco na pesquisa e desenvolvimento de grades computacionais e foi resumidamente descrito na seção 4.2. O projeto I-WAY era baseado na utilização de servidores I-POP, responsáveis por interconectar os sítios participantes, fornecendo aos usuários suporte a interface de programação paralela MPI, além de um escalonador de processos e um sistema de segurança.

O **ProGrid** irá oferecer serviços de gerenciamento da grade, recursos de segurança, como autenticação e criptografia do tráfego e suporte a interface de programação MPI. Através da análise do **ProGrid**, verifica-se que a entidade Proxy apresenta certa semelhança com o I-POP utilizado no I-WAY. A principal diferença está na maneira pela qual os resultados foram alcançados.

A forma mais utilizada para tornar a implementação do MPI compatível com uma grade é estendendo suas funcionalidades e, se necessário, reimplementando a camada de funções de comunicação para torná-la compatível com outros serviços da grade, como de segurança e de escalonador de processos, entre outros. Tal abordagem pode ser encontrada em projetos

como: PACX-MPI [67] e MPICH-G2 [68]. Esta última faz parte da Toolkit do Globus e oferece suporte a segurança. A MPICH-G2 é uma extensão da implementação MPICH, largamente conhecida e utilizada, desenvolvida pelo Argonne National Laboratory and Mississippi State Univerty.

Neste trabalho optou-se por utilizar a implementação MPICH original, utilizando o Proxy como um ponto de controle. Essa abordagem, utilizando o Proxy como ponto de acesso intermediário apresenta algumas vantagens como:

- **Maior Segurança:** Toda a comunicação entre os servidores Proxies utiliza uma única porta, sendo todo o tráfego multiplexado neste canal de comunicação seguro. Originalmente, o tráfego MPI utiliza diversos números de portas, sendo que para cada nova aplicação, diferentes portas são utilizadas. Caso este tráfego passe por um *Firewall* é necessário que o mesmo tenha um conjunto grande de portas desbloqueadas (referentes as aplicações MPI). Com a utilização do Proxy, todo o tráfego das aplicações MPI é multiplexixado num único canal de comunicação, autenticado e seguro;

- **Transparência:** os usuários podem aproveitar os recursos disponíveis no **ProGrid** de forma transparente, como os recursos de segurança e localização de recursos;

- **Confiabilidade:** a interferência no código da biblioteca de programação paralela MPICH foi altamente minimizada, sendo praticamente nula. Esta característica permite que alterações na implementação do MPICH (salvo alterações no mecanismo de inicialização dos processos), não a torne incompatível com o sistema **ProGrid**. Dessa forma, futuras versões apresentam maior probabilidade de continuarem sendo compatíveis com o **ProGrid**.

Dois tipos de aplicações paralelas foram focados neste trabalho: no primeiro deles, uma aplicação já paralelizada, para um conjunto heterogêneo de máquinas, num ambiente de redes não local passa a utilizar os serviços do Proxy para aproveitar as funcionalidades de autenticação e segurança do **ProGrid** de maneira transparente; outra forma de desenvolvimento de aplicações para a grade utiliza funções de uma nova API desenvolvida para usar explicitamente os recursos autenticação, segurança e distribuição de carga disponibilizada pela grade.

5.3 Estrutura da Arquitetura

A arquitetura do sistema foi modelada em camadas de serviços, baseando-se no conceito e estrutura apresentado na seção 3.4. Assim, foram modelados os principais serviços necessários a uma grade computacional. Para o escopo deste trabalho entretanto, apenas alguns deles foram realmente implementados. O restante, apesar de igualmente importante será abordado em trabalhos futuros e anexados ao **ProGrid** gradativamente.

A Figura 11 ilustra a composição geral do **ProGrid**, esquematizada em camadas de abstração, onde as camadas inferiores disponibilizam serviços para as camadas superiores. Neste trabalho estão sendo apresentadas as camadas 1, 2, 3 e 4, que foram implementadas objetivando o suporte a comunicação, segurança, controle da grade e suporte a programação paralela utilizando o padrão MPI, respectivamente.

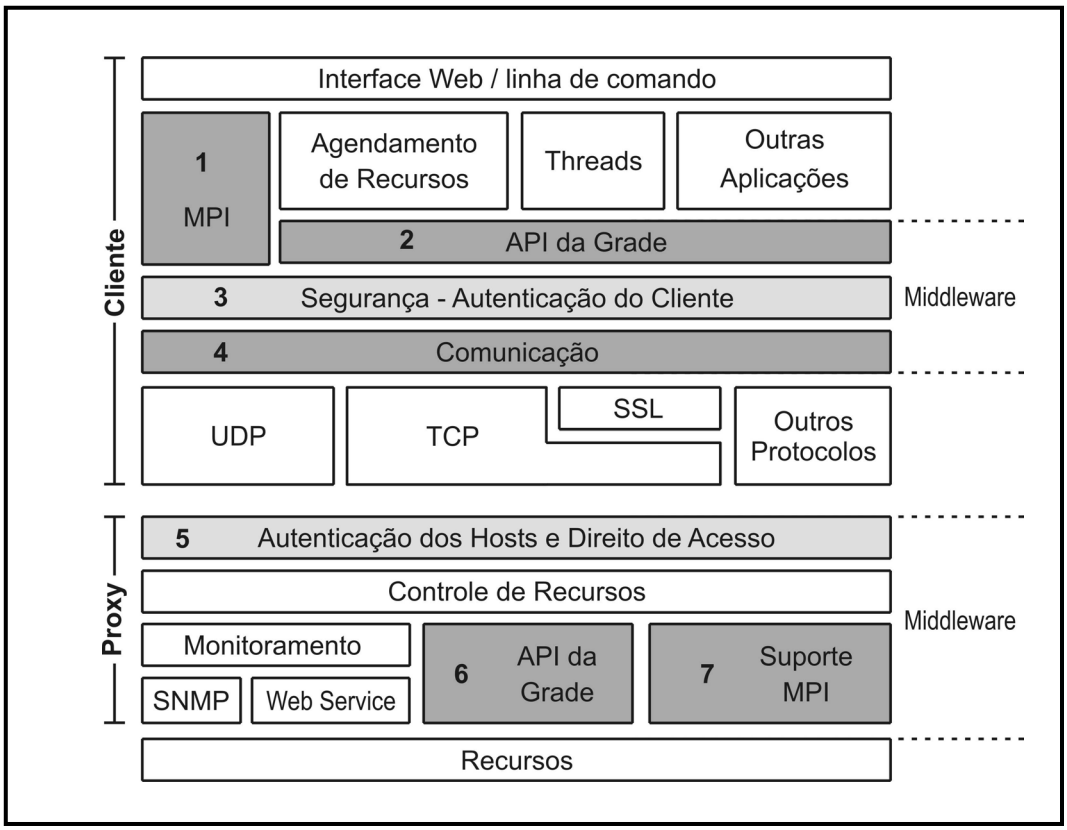

*Figura 11: Arquitetura em camadas do **ProGrid***

Resumidamente, as camadas 2, 3 e 4, formam a midleware presente nos clientes e responsável por oferecer aos usuários e aplicativos o acesso aos recursos do **ProGrid**. As camadas 5, 6 e 7 constituem a Middleware do servidor Proxy responsável por interligar e mediar o acesso aos recursos da grade, oferecendo os serviços de suporte e controle necessários.

As camadas que foram modeladas, e ainda não implementadas no presente trabalho, são também abordadas e resumidamente descritas para melhor entendimento do contexto no qual o **ProGrid** está inserido e também como referência para a expansão almejada em trabalhos futuros. A seguir é apresentado a descrição de cada camada:

- **Interface de Web / Linha de Comando:** esta camada disponibiliza para o usuário uma interface de acesso, através do qual o mesmo interage direta ou indiretamente às funcionalidades do **ProGrid**. Para tanto, o usuário terá a sua disposição além da linha de comando, uma interface Web, que facilitará o acesso às informações e serviços. A interface Web também será responsável por permitir a utilização dos recursos da grade por usuários remotos ou por máquinas que não fazem parte da grade. Dessa forma, as máquinas que não possuírem a Middleware que disponibiliza os recursos da grade, poderão fazer uso dos recursos utilizando uma interface Web pela rede, sem ter a necessidade de instalar softwares adicionais. Tal interface está em fase de desenvolvimento;

- **MPI (1):** esta camada representa a biblioteca de programação paralela MPICH. Ela fornece ao usuário funções que facilitam a criação de programas paralelos, conforme descrito na seção 2.5.2. Para utilizar os recursos providos pela grade, esta camada interage com a camada 4 (Suporte MPI) localizada no Proxy e será descrita mais adiante;

- **Agendamento de Recursos:** para garantir a confiabilidade e a prioridade na utilização dos recursos, a grade deve ter mecanismos que permitam ao usuário especificar e alocar os recursos necessários para realizar uma determinada tarefa. Esta camada trabalha em conjunto com a camada de Controle de Recursos localizada no Proxy e que efetivamente realiza o agendamento dos recursos;

- **Threads Distribuídas:** oferece suporte para o gerenciamento de *threads*, independente da biblioteca utilizada. Este serviço não faz parte do escopo deste projeto, mas vislumbra-se a possibilidade de incorporá-lo ao **ProGrid**;

- **Outras Aplicações:** aqui se encontram diversas aplicações que podem utilizar alguns recursos do **ProGrid** explicitamente. Outras aplicações podem ser diretamente suportadas, sendo desmembradas em outras camadas, para garantir a transparência e a total compatibilidade de tais aplicações com o **ProGrid**;

- **API da grade (2):** esta camada disponibiliza funções de manipulação da grade a outras camadas, aplicações ou diretamente para os usuários. Tais chamadas são executadas pelos servidores Proxies. Mais detalhes sobre as funções disponíveis será descrita logo abaixo na camada API da grade (6);

- **Autenticação de usuários (3):** esta camada é responsável por prover a autenticação dos usuários que desejam utilizar os recursos do **ProGrid**. Dessa forma deve impedir os acessos de usuários não cadastrados na grade e daqueles que apresentarem alguma irregularidade, garantindo a confiabilidade e segurança do sistema. A segurança foi dividida em 2 camadas: uma presente na *middleware* dos clientes e outra presente no servidor Proxy. As questões referentes às políticas de segurança e implementação das camadas aqui representadas são discutidas em detalhes na seção 5.5.3;

- **Comunicação (4):** nesta camada estão localizados os recursos que controlam e possibilitam a comunicação entre os sítios que compõem a grade. Até o momento, o **ProGrid** oferece 3 tipos de canais de comunicação entre os sítios através do servidor Proxy: canais de comunicação sem conexão, através do protocolo UDP, com conexão utilizando o TCP e também um canal de comunicação seguro e autenticado utilizando o SSL sobre TCP;

- **UDP/TCP:** esta camada constitui os protocolos de rede sobre o qual os serviços de comunicação foram montados;

- **SSL:** protocolo de segurança que fornece as funções para a comunicação segura e para a autenticação. Os serviços de autenticação e comunicação segura utilizam as funções disponibilizadas por esta camada. Novos mecanismos de segurança podem ser inseridos neste nível da arquitetura com o objetivo de aumentar as funcionalidades e a compatibilidade do **ProGrid** com outras tecnologias e padrões de segurança;

- **Outros Protocolos:** dado que a grade é formada por um ambiente heterogêneo, o **ProGrid** deve suportar mais protocolos de forma transparente. Outros protocolos

inseridos nessa camada serão acessados da mesma forma através da camada de comunicação, garantido a transparência para as camadas superiores e para os usuários;

- **Autenticação de Hosts e Direito de Acesso (5):** esta camada complementa os requisitos de segurança, sendo responsável por checar os direitos dos usuários e também por autenticar os diversos sítios que compõem a grade. Como canais de comunicação não-confiáveis (como a Internet) podem ser utilizados para interconectar os diversos sítios que compõem a grade, canais de comunicação seguros são necessários. Nesse sentido, é responsável por controlar a segurança (autenticação e criptografia) na comunicação entre os servidores Proxy. Além disso, determina os direitos de acesso e do *logon* destinados aos recursos requisitados. Os detalhes referentes a implementação desta camada são abordados na seção 5.5.3;

- **Controle de Recursos:** esta camada é responsável por controlar os recursos presentes na grade, podendo ser desmembrada em diversos tipos de controle, como, por exemplo, controle de balanceamento de carga;

- **Monitoramento:** esta camada constitui um serviço de grande importância para a grade que é constituída de um ambiente heterogêneo e dinâmico, ou seja, que sofre alterações na sua composição a todo instante. Dessa forma, é necessário dispor de mecanismos para monitorar o estado da grade e também para obter informações específicas dos recursos que a constituem. Isso deve ser feito utilizando padrões como o SNMP [69] para coleta de dados e Web Services [70] como mecanismo para disponibilizá-las;

- **SNMP:** esta camada é constituída pelo Simple Network Management Protocol (SNMP), protocolo responsável pelo gerenciamento e monitoramento de dispositivos da rede e suas respectivas funções;

- **Web Service:** a utilização de *web services* tem por objetivo facilitar a disponibilização das informações da camada de monitoramento para os usuários, aplicativos e outras camadas de serviço da grade através de uma interface de acesso padronizada;

- **API da grade (6):** esta camada disponibiliza funções de manipulação da grade. Entre as funções que podem ser utilizadas por outras aplicações, ou diretamente pelos usuários, destaca-se: função *(host_status)* para verificar o estado de um recurso dentro da grade (ligado, desligado), função *(get_host_domain)* que retorna o domínio no qual o host se

encontra, função *(host_number)* que captura o número de hosts ativos na grade e função *(proxy_number)* que indica o número de servidores Proxies ativos;

- **Suporte ao MPI (7):** esta camada constitui a parte central do trabalho, sendo responsável por oferecer suporte a programação paralela através do padrão MPI. Dessa forma essa camada fornece interage com a biblioteca MPI, compatibilizando tal aplicação com o ambiente e os recursos da grade oferecido pelo **ProGrid**. As aplicações escritas no padrão MPI podem ser executadas aproveitando os recursos disponíveis dentro da grade de forma transparente, ou seja, sem que seja necessária a alteração de código. Os detalhes de funcionamento e implementação desta camada são discutidos na seção 0;
- **Recursos:** na base da arquitetura encontra-se a camada de recursos. Esta camada é formada por todos os recursos que compõem a grade, incluindo: computadores de mesa, aglomerados, supercomputadores, sistemas de armazenamento, entre outros. O **ProGrid** focou-se na utilização de computadores de mesa e aglomerados.

5.4 Classes de Aplicações Suportadas

Esse trabalho visa principalmente oferecer suporte a aplicações escritas no padrão MPI. Os usuários do **ProGrid** também terão a possibilidade de executar aplicações que não utilizam o padrão MPI, mas neste caso, terão que paralelizá-las e iniciá-las explicitamente. Porém, neste último caso, os usuários terão à disposição as funcionalidades da API da grade, bem como os mecanismos de segurança e controle.

Conforme apresentado no Capítulo 2, existem diversos níveis de aplicações paralelas, o que reflete diretamente nos requisitos necessários para executá-las. Dessa forma, temos que o desempenho das aplicações é guiado pelo ambiente no qual estão inseridas. Para executar aplicações MPI que apresentam granularidade fina, o sistema deve apresentar uma arquitetura interconectada por uma rede de alta velocidade, pois irá existir muita troca de mensagens. Já no caso de aplicações com granularidade grossa, redes compartilhadas com desempenho menor são aceitas.

Nesse sentido, é de vital importância que o ambiente forneça os requisitos necessários de acordo com o nível de paralelismo das aplicações. Na grade, isso é dificultado devido à

dinâmica e a heterogeneidade do ambiente, que engloba diversos tipos de recursos com desempenhos diferentes. O **ProGrid** é interconectado por diversas redes, incluindo desde redes locais de alto desempenho até redes de longa distância como a Internet. Para executar aplicações de granularidade fina, seriam necessários mecanismos que permitissem a analise do ambiente selecionando células interconectadas por redes de alta velocidade para executar aplicações desta classe. Esta característica é proposta como trabalho futuro e é resumidamente discutida na seção 6.1.1.

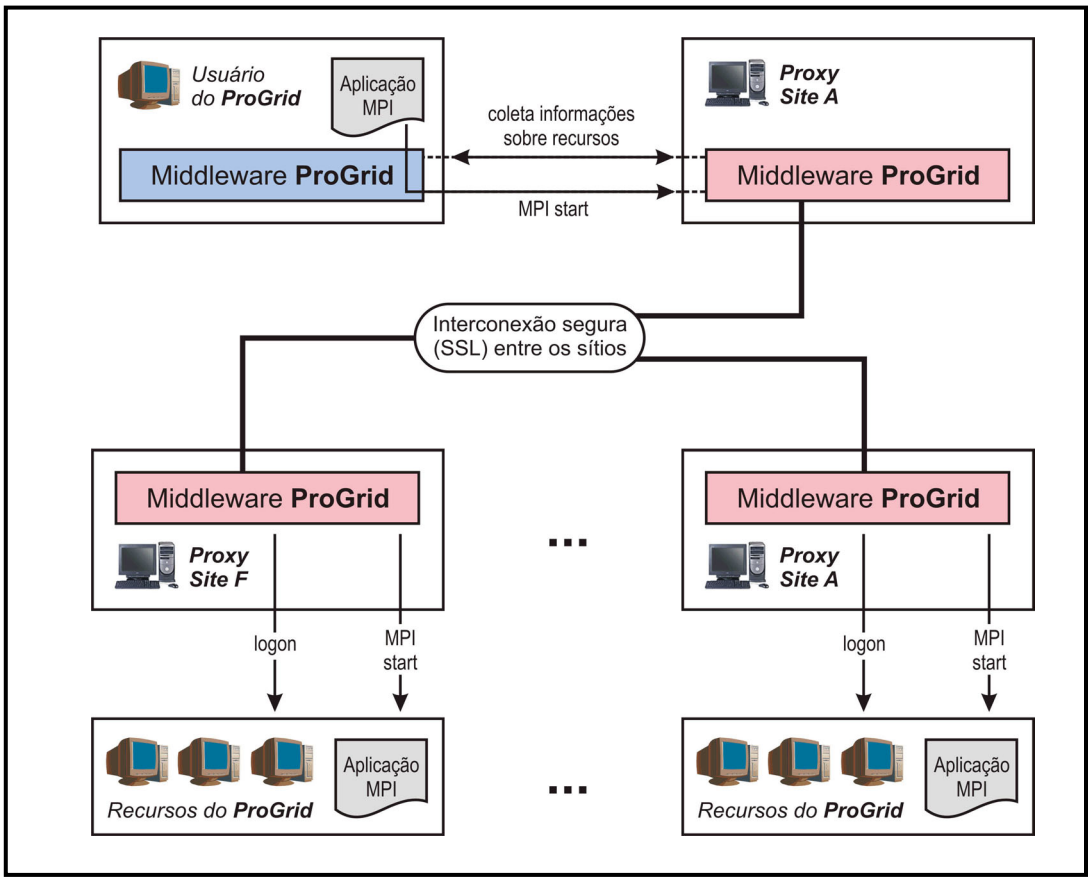

*Figura 12: Ilustração da aplicabilidade do **ProGrid***

No entanto, este trabalho manteve o foco em aplicações de granularidade grossa que não exigem uma rede de alta velocidade entre estações nas quais a aplicação paralela está sendo executada. A Figura 12 ilustra uma abordagem para utilização do **ProGrid**, guiado pela necessidade de executar uma aplicação escrita no padrão MPI, em diversos recursos.

5.5 Implementação

5.5.1 Plataforma de Hardware e Software;

Todos os módulos do sistema **ProGrid** foram implementados na linguagem C e compilados na plataforma *Linux Suse 8.0*. A opção pela plataforma *Linux* é resultado do reconhecimento desta na comunidade científica que defende a utilização de código aberto. A construção do sistema utilizou algumas bibliotecas de domínio público para implementar as seguintes funcionalidades básicas:

- **Segurança:** o conjunto de bibliotecas *OpenSSL* [71], uma implementação de código aberto do protocolo SSL, foram utilizadas para prover as funções de autenticação e criptografia do tráfego. Tais mecanismos são utilizados entre os servidores Proxy para prover um canal seguro e confiável;
- **Comunicação:** o conjunto de bibliotecas de *"Berkeley Sockets"* [72] foi utilizado para prover as funções básicas de comunicação ponto-a-ponto;

Para auxiliar no processo de desenvolvimento do código, foi utilizada a interface de programação *Anjuta*, assim como o *vi*, aplicativos integrantes da distribuição *Linux Suse 8.0*.

Como um dos objetivos do presente trabalho é fornecer suporte a interface de programação paralela MPI, utilizou-se uma implementação de código aberto que foi incorporada ao sistema final. A referente biblioteca é a MPICH compatível com padrão MPI e amplamente utilizada em diveros projetos de grades computacionais, como [7] e [58].

5.5.2 Módulo de suporte a interface de programação MPI

Diferentemente de outras abordagens, como Globus [58], o suporte a interface de programação MPI é feito externamente, não havendo intervenção direta no código fonte do MPI. As vantagens dessa abordagem são:

- **Reutilização:** utilizando as ferramentas criadas, futuras aplicações podem ser mais facilmente incorporadas, sem a necessidade de intervenção no código fonte. Além disso, nem sempre é possível ter acesso ao código fonte;

- **Confiabilidade:** a inserção de código internamente na aplicação aumenta a probabilidade de falhas disparadas pela aplicação. No caso da abordagem externa, a identificação das falhas e sua repercussão sobre a arquitetura podem ser reduzidas de maneira mais eficiente;
- **Diminuição da sobrecarga:** a sobrecarga gerada pelas informações de controle e segurança, como criptografia, é introduzida apenas em alguns nós e não em todos.

Como exemplo, tem-se a interligação de aglomerados através da grade. Nas abordagens tradicionais, como a segurança incide dentro da aplicação MPI, todos os nós do aglomerado refletem a sobrecarga gerada pela comunicação segura e de controle da grade. No caso da abordagem proposta, haverá um tunelamento das informações apenas entre os aglomerados e não dentro deles;

Para dar suporte a aplicações MPI e permitir as suas execuções em toda a grade, o Proxy atua como um multiplexador da comunicação entre o processo *master* e seus respectivos *escravos*. Tal abordagem fornece ao MPI a ilusão de um único aglomerado virtual. Para alcançar tais resultados, optou-se por não interferir internamente no MPI, utilizando-se o Proxy como entidade responsável por fornecer a abstração necessária ao MPI. Isso foi feito criando-se *escravos virtuais* no Proxy que mantêm comunicação direta com o processo *master* do MPI. Os *escravos virtuais* repassam as informações através de canais seguros para os respectivos nós reais responsáveis pela execução do processo. Os *escravos virtuais* constituem então a abstração responsável por fornecer a ilusão do cluster virtual.

Para cada aplicação MPI iniciada na grade, é criado no Proxy um novo espaço de endereçamento associado à aplicação. A partir dos parâmetros fornecidos pelo usuário, ou através de uma configuração padrão, o Proxy distribui os processos ao longo da grade, cria os *escravos virtuais* associando-os com os nós reais. Este mapeamento realizado pelo Proxy é transparente para a aplicação e pode ser visto como uma multiplexação da comunicação entre a origem e o destino. É importante ressaltar que não é necessária a instalação de nenhum módulo adicional no cliente, além do MPI e da introdução do servidor proxy nos sítios. Isso facilita a sua aplicabilidade em ambientes já estruturados. A Figura 13.a ilustra a comunicação de uma aplicação MPI numa rede local. A Figura 13.b exibe a multiplexação da comunicação feita pelo Proxy para permitir o tunelamento seguro do tráfego.

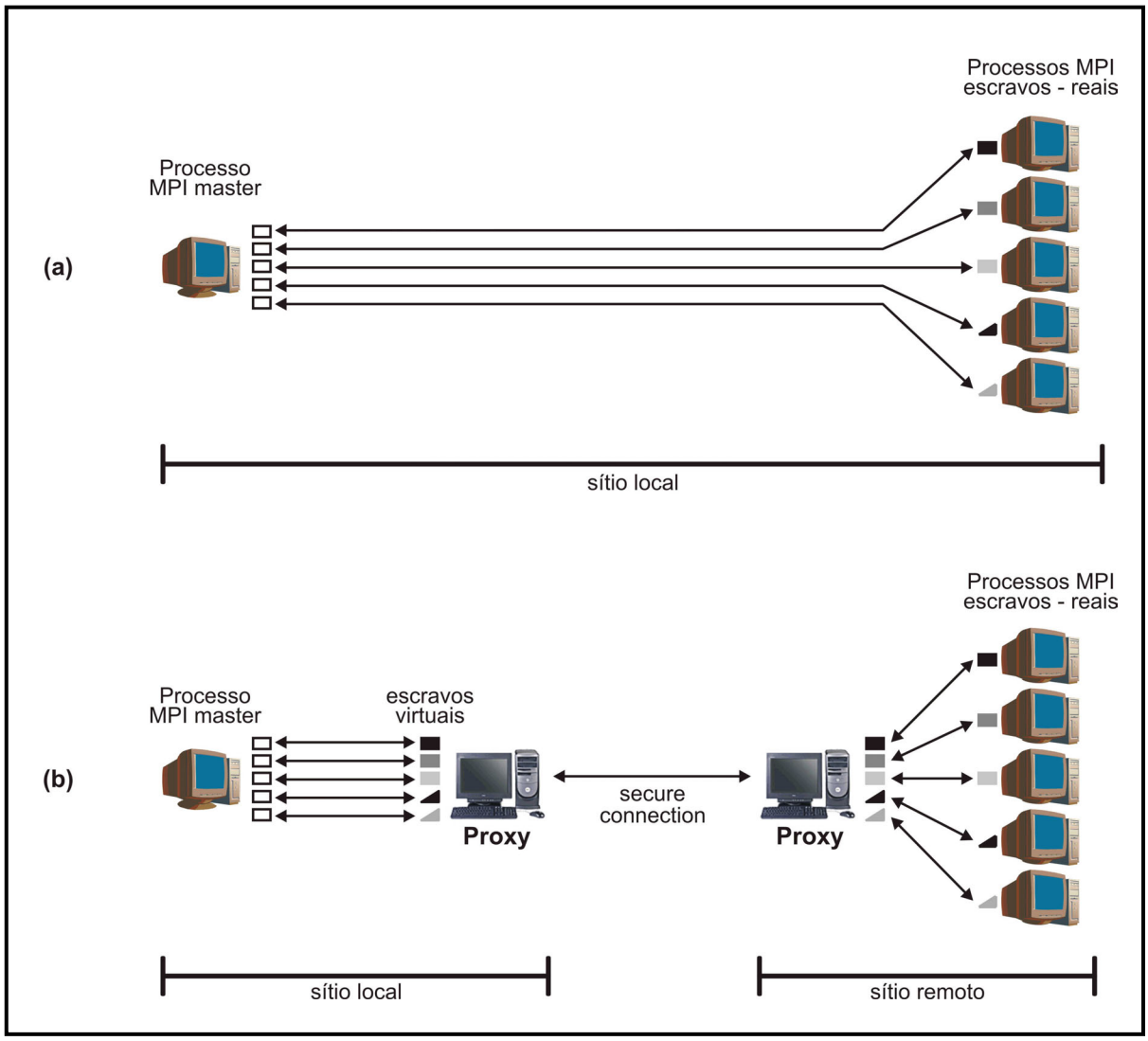

Figura 13: (a) comunicação local – (b) multiplexação realizada pelo proxy

A Figura 14 ilustra a representação lógica da implementação do módulo de suporte à biblioteca MPI. Nela, é possível verificar como o suporte a biblioteca MPI foi implementado. O comando de inicialização da aplicação MPI é passado ao Proxy, que verifica os recursos destinos e encaminha tais requisições aos Proxies remotos dos quais os recursos fazem parte. Cada Proxy mantém uma lista de sockets (escravos virtuais) que controlam a comunicação entre os sítios.

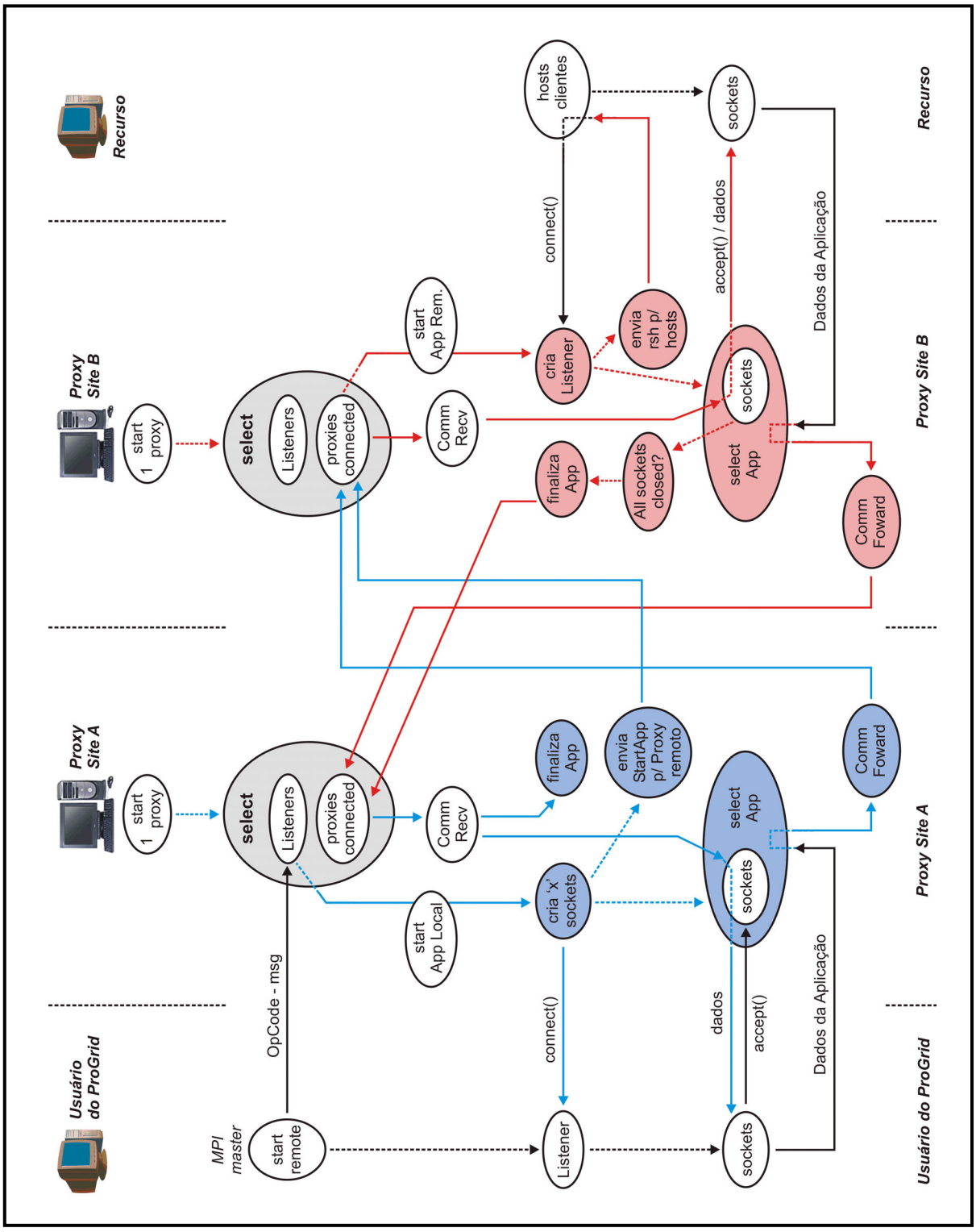

Figura 14: Representação lógica do modulo de suporte da biblioteca MPI

Outra característica a ser incorporada no **ProGrid** é o serviço de balanceamento de carga. Este serviço visa garantir um melhor aproveitamento e otimização dos recursos disponíveis. Como o controle e o suporte as aplicações MPI é feito através do Proxy, a inclusão desse serviço, irá permitir também, o balanceamento de carga das aplicações MPI executadas no

ProGrid. Este serviço é sugerido como trabalho futuro, sendo resumidamente descrito na seção 6.1.2.

5.5.3 Módulo de Segurança

O módulo de segurança tem por objetivo disponibilizar mecanismos de segurança capazes de suportar os desafios inseridos na condução de trabalhos no ambiente criado pelo **ProGrid**. Seguindo os requisitos de segurança desejados nas grades computacionais, que foram apresentados na seção 3.5, e valendo-se da delimitação do escopo realizada, os desafios de segurança atingidos com o módulo de segurança do **ProGrid** são:

- **Isolamento:** caso um determinado sítio da grade, ou algum componente dentro deste apresente alguma falha de segurança, todo o sítio pode ser isolado, bastando para isso à exclusão temporária (ou permanente) do Proxy responsável pelo sítio, da relação de sítios confiáveis;
- **Autenticação única:** o usuário do **ProGrid** não precisa realizar a autenticação cada vez que um novo serviço é solicitado. Apenas uma única autenticação é solicitada ao usuário, sendo que as autenticações futuras são realizadas pelo Proxy em seu nome;
- **Integração com a segurança local:** apesar de não suportar diversas tecnologias de segurança, como Kerberos e X.509, o **ProGrid** buscou prover um sistema de segunça semelhante ao utilizado pelo MPI, através de *rsh* e *ssh*;
- **Privacidade e Integridade entre os sítios:** a comunicação feita entre os servidores Proxy (que delimitam e fornecem as funcionalidades aos sítios) deve ser estritamente segura. Para atingir esse objetivo, utilizou-se um canal seguro, criptografado que garante a confidencialidade das informações trafegadas entre os sítios. Para atingir tais fins, o protocolo SSL foi utilizado através da invocação das funções da biblioteca de código aberto, OpenSSL. A utilização de um canal de comunicação seguro apenas entre os sítios e não fim-a-fim (entre usuário e o recurso) reflete a concepção dada ao sistema.

Na definição do sistema verificou-se que a maioria dos sítios já utiliza algum sistema de segurança. A utilização de um canal seguro entre o usuário e o recurso poderia sobrepor mais de um sistema de segurança, gerando uma sobrecarga desnecessária. A integração dos sistemas de segurança é feita através dos pontos de interconexão dos sítios, ou seja, através dos servidores Proxy. Qualquer comunicação proveniente de outras fontes estará sujeita às políticas de segurança locais já existentes, enquanto as comunicações provindas de entidades da grade estarão sujeitas às políticas de segurança definidas para a grade;

- **Passagem por Firewall:** a utilização do servidor Proxy multiplexando diversas conexões locais sobre uma única conexão externa (tunelamento), contribui para minimizar a necessidade de mudanças na política de segurança, como por exemplo, alteração das regras utilizadas no *firewall*;
- **Autenticação:** a autenticação é um ponto muito importante e indispensável para praticamente todo sistema distribuído. Os mecanismos de autenticação garantem a origem das ações e definem as permissões de acesso dado aos usuários de acordo com a política adotada. No **ProGrid** a autenticação não é fim-a-fim, mas sim dividida em três níveis. A Figura 15 ilustra os níveis de segurança que são descritos em mais detalhes posteriormente.

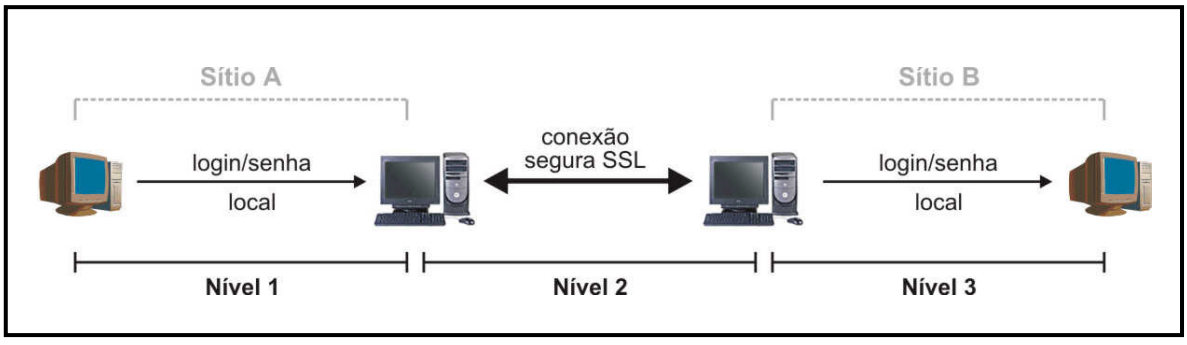

Figura 15: Níveis de Segurança

5.5.3.1 Nível 1 (Usuário → Proxy)

A autenticação dos usuários do **ProGrid** é controlada localmente. Cada servidor Proxy é responsável por controlar a base de usuários locais, pertencentes ao seu sítio de atuação. Para ingressar na grade, tanto como usuários de serviço ou servidores de recurso, as entidades

devem explicitamente executar um aplicativo *ProGrid_init*, que irá autenticar a entidade junto ao Proxy. Além disso, essa função muda o *status* da entidade para ativo, alterando a composição da grade. A autenticação entre os usuários e o Proxy é feita via login/senha. Dessa forma, a função *ProGrid_init* recebe o *login* e a *senha do usuário* como parâmetro. Após realizar a autenticação, os usuários estão aptos a utilizar os recursos providos pelo **ProGrid** de forma transparente, levando em consideração as permissões de acesso.

5.5.3.2 Nível 2 (Proxy→ Proxy)

Neste nível, é necessário que todos os servidores Proxy dentro da grade, sejam capazes de realizar autenticação mútua. Neste caso, também foi utilizado o protocolo OpenSSL, pois este provê um sistema de autenticação que pode ser utilizado de forma separada aos de comunicação segura dos dados (Criptografia). Para permitir a comunicação entre entidades localizadas em sítios diferentes, é necessária a autenticação mútua dos servidores Proxies responsáveis pelo sítio.

Toda a comunicação/requisição proveniente de sítios que não estejam cadastrados na grade, não será aceita. Esta verificação é feita através de certificados emitidos por uma Autoridade de Certificação (CA) eleita pelos integrantes da grade. É aconselhável ter uma CA específica para cada grade, pois isto facilita o gerenciamento e restringe a utilização dos certificados, aumentando a segurança. No estabelecimento de um canal de comunicação entre dois sítios da grade, é necessária a realização da autenticação mútua desses sítios através do Proxy. Se o certificado não for válido, a autenticação irá falhar e a comunicação entre tais entidades será bloqueada até que seja apresentado um certificado válido.

5.5.3.3 Nível 3 (Proxy → Usuário)

Este nível de autenticação corresponde a terceira e última fase, onde é verificada a permissão de acesso do usuário que fez a requisição. O Proxy do sítio onde o recurso está inserido é responsável por esta tarefa. Para tanto, no momento em que o recurso é cadastrado na grade através da função *ProGrid_init*, o arquivo "*~/.rhosts"* dessa estação é enviado ao Proxy. As permissões de acesso são avaliadas segundo as entradas contidas neste arquivo. Assim,

quando um usuário remoto tentar acessar o recurso, a permissão de acesso será verificada no Proxy do sítio no qual o recurso faz parte. Essa abordagem facilita o mapeamento de um conjunto de usuários. Para dar permissão de acesso a todos os usuários de um determinado sítio, basta o usuário colocar uma entrada no seu arquivo *"~/.rhost"*, colocando o sítio autorizado. Dessa forma, todos os usuários contidos no domínio do sítio autorizado, poderão acessar o recurso usando o mesmo *login*, mas ainda sim, terão que ser autenticados pelo Proxy local referente ao sítio ao qual estão inseridos.

Quando um usuário desse sítio fizer uma requisição, o Proxy do sítio no qual o recurso faz parte autorizará o acesso de todas as requisições provindas dos sítios autorizados, utilizando o *login* oferecido pelo recurso ao sítio em questão. Além disso, este mecanismo facilita o gerenciamento, que pode ser feito pelo próprio usuário detentor do recurso. Essa configuração é semelhante à necessária para a execução de aplicativos da interface MPI em sítios locais, onde o usuário (ou no caso do **ProGrid**, o sítio no qual o usuário está inserido) deve ter conta em todos os recursos que deseja utilizar.

Para carregar as permissões de acesso dos recursos que compõem o **ProGrid**, o arquivo *"~/.rhost"* do recurso é lido no momento em que a função *ProGrid_init* é executada. Neste instante, as permissões de acesso são transferidas para o servidor Proxy. Tais pemissões podem ser direcionadas diretamente a um único usuário, ou a um determinado sítio. No caso da permissão dada a um determinado sítio, todos os usuários do sítio em questão, poderão acessar o recurso. No entanto, esse acesso poderá apenas ser realizado utilizando o servidor Proxy como intermediário, para garantir a segurança e evitar que usuários não cadastrados na grade tenham ao recurso.

5.5.4 Estrutura de Dados e Algoritmo Utilizados

As configurações referentes aos usuários, recursos e Proxies que compõem a grade são armazenadas em arquivos texto. A utilização de banco de dados para esse fim é interessante, porém não foi abordada no presente trabalho.

Assim, quando o Proxy está ativo, uma lista encadeada dos servidores Proxies ativos é mantida em memória. Outras duas filas encadeadas são mantidas na memória pelo Proxy: uma para armazenar informações referentes às aplicações em execução na grade e outra para armazenar informações dos recursos que estão executando alguma computação. Para facilitar a programação, foram criadas algumas funções para manipulação das filas, como inserção, busca e exclusão. A seguir as estruturas e o algoritmo utilizados são descritos de forma genérica:

Estruturas de Dados

1. **CA** (Autoridade de Certificação)
    a. ip;
    b. phe;
    c. certificado;
    d. proxies Cadastrados;

2. **Proxy** (lista de proxies cadastrados na grade)
    a. hostname;
    b. ip;
    c. phe: struct hostent *phe;
    d. porta (15.000 - fixa para Proxy);
    e. socket (manter conexão enquanto proxy ativo);
    f. ssl (certificado para cada Proxy cadastrado);
    g. status (flag de controle ativado/desativado);

3. **Hosts** (lista de hosts cadastrados no site)
    a. hostname;
    b. ip;
    c. phe: struct hostent *phe;
    d. status: (flag de controle ativado/desativado);
    e. acesso:
        i. login name no host local
        ii. login name no host remoto

4. **Users** (lista de usuários)
    a. login;
    b. hosts (lista de hosts que inseriu no grid: permite retirar entradas da lista de aceites nas entradas de hosts);
    c. certificado;

5. **Slaves** (lista de "virtual slaves": pontos de acesso locais das aplicações iniciadas remotamente)
    a. hostname;
    b. ip;
    c. porta;
    d. socket local (conectado à aplicação);
    e. id (identificação da aplicação: permite identificar o outro socket no Proxy)

6. **App** (lista de aplicações)
    a. slaves (lista de "virtual slaves");

Estruturas de comunicação

1. **UDP** (atende requisições de serviço)
    a. register, unregister;
    b. startapp (local);
    c. snmpget, snmpgetnext, snmpset;

2. TCP/SLL
    a. um socket para cada proxy conhecido
    b. um socket para cada processo (slave) remoto iniciado

Algotitmo

fim=0;
do {
espera_dados();
se UDP:
se register:
- insere dados do usuario e das maquinas de quem aceita rsh (.rhosts) e do host local

se unregister:
- retira dados do host local, do usuario

se startapp:
- verifica usuario e host;
- insere entrada na lista de aplicações
- identifica proxy associado ao destino
- encapsula dados
- envia para o proxy apropriado
- criar socket local associado a aplicacao, gerando entrada na lista de virtual slaves

se snmpget:
se snmpgetnext:
se snmpset:

se TCP:
se local:
- identifica destino, encapsula e envia para proxy apropriado

se remoto (veio de proxy):

- se handshake_proxy:
    - cria entrada para o proxy
    - inicia sessao SSL
    - identifica hosts remotos a ele associados

se startapp: // veio de socket/proxy remoto
- identifica destino, verifica direito de acesso
- desencapsula e envia para cliente local apropriado

se resposta startapp:
- localiza aplicacao
- cria socket local: ja' foi criado em startapp?
- conecta socket local ao socket da origem
- envia dados que chegaram do socket remoto para o socket local

se dados app:
- identifica aplicacao e socket local
- envia dados que chegaram do socket remoto para o socket local

} while (!fim);

5.6 Considerações

Este Capítulo abordou os detalhes referentes ao projeto, modelagem, estrutura e implementação do sistema **ProGrid**. A abordagem para suporte à interface MPI e para o módulo de segurança foram descritas, mostrando como os requisitos foram alcançados. Alguns dos módulos que ficaram de fora do escopo de implementação deste trabalho já estão sendo abordados por outros trabalhos e serão incorporados ao sistema **ProGrid**.

Capítulo 6

Discussão e Conclusões

Como pode ser visto no decorrer deste trabalho, o conceito de grades computacionais está se expandindo, ganhando adeptos em diversas áreas, principalmente no meio acadêmico e de pesquisa. No entanto a infra-estrutura que viabiliza a criação dessa estrutura é complexa e irá exigir muito esforço no sentido da padronização das arquiteturas e dos protocolos para permitir a integração de domínios em larga escala. A heterogeneidade do ambiente da grade de ser superada através de Middlewares que acoplem os diversos recursos e tecnologias, oferecendo um ambiente transparente para os usuários. A segurança é outro desafio em ambientes distribuídos, sendo de vital importância para a grade. A interconexão de diversos sítios com políticas de segurança distintas somada ao elevado grau de compartilhamento dos recursos que compõem a grade resultam na necessidade de um sistema de segurança complexo e padronizado.

O trabalho aqui apresentado compreende o início do esforço na criação de uma grade computacional que servirá de base para a criação de outros serviços e recursos. O trabalho apresenta uma arquitetura própria, porém sem distanciar-se dos padrões que começam a ser estabelecidos. Dessa maneira, atinge-se como resultado desse trabalho uma infra-estrutura de suporte a programação paralela em grades computacionais, nomeada **ProGrid**.

A abordagem utilizando servidores Proxy como entidades intermediárias na comunicação entre sítios tem como vantagens:

- **Facilidade de instalação:** a configuração e instalação de uma grade ou de apenas um único sítio é simples. As maiorias dos recursos são instaladas e providas pelo Proxy, minimizando a alteração da infra-estrutura existente;

- **Simplicidade da arquitetura:** o **ProGrid** apresenta uma arquitetura simples, com controle centralizado nos servidores Proxies. Outras arquiteturas apresentam diversos serviços que são instalados em diversas máquinas o que dificulta a instalação, o controle e a manutenção da estrutura;
- **Segurança:** o tráfego de dados entre os sítios são multiplexadas através de um canal de comunicação seguro (autenticado e criptografado). Dessa forma, o Proxy pode operar dentro do domínio do sítio, atrás da barreira de segurança (Firewall), exigindo que apenas poucas portas sejam abertas para permitir a comunicação.

Em contrapartida, a principal desvantagem introduzida por essa abordagem, concentra-se no gargalo criado pelo Proxy. Se o Proxy falhar, um sítio pode ficar isolado até que o problema seja sanado. Além disso, o Proxy pode limitar o crescimento, pois pode tornar-se um gargalo no tratamento/encaminhamento das requisições dos recursos e usuários. Isso pode acarretar em uma concorrência no acesso ao servidor Proxy. Tais desvantagens podem ser minimizadas através da exploração da paralelização dos servidores Proxy e com a utilização de servidores Proxy redundantes.

Apesar das desvantagens citadas, a abordagem apresentada visa ambientes de pequeno e médio porte, onde cada sítio é constituído de algumas poucas centenas de estações. Dessa forma, não espera-se que o Proxy se torne um gargalo na comunicação, sendo que testes devem ser realizados para verificar-se o seu limite.

6.1 Trabalhos Futuros

Dentre os diversos serviços necessários em uma grade computacional, apenas uma parte deles foi desenvolvida no **ProGrid**. Dessa forma, existe um vasto campo na criação de novos recursos e serviços para o ambiente aqui proposto, sendo que algumas expansões já foram propostas e já começam a serem desenvolvidas.

6.1.1 Serviço de Monitoramento

As grades computacionais são inerentemente dinâmicas, pois sua composição é alterada a todo instante e são também heterogêneas, sendo formadas em sua maioria por diferentes plataformas e tecnologias. Assim, o serviço de Monitoramento é de grande importância para uma grade computacional, pois ele irá coletar as diversas informações que servirão de parâmetro para outros serviços na manipulação da heterogeneidade e dinâmica da grade.

Para permitir a interoperabilidade, este serviço deve ser também, quando possível, baseado em padrões e tecnologias comuns. Neste caso, o protocolo SNMP constitui uma importante tecnologia, amplamente difundida e utilizada na coleta de informações em diversos recursos computacionais.

Vários outros serviços e funcionalidades podem ser adicionados na grade aproveitando-se das informações fornecidas por esta camada. A utilização desse serviço no **ProGrid**, possibilitaria ao mesmo oferecer suporte a aplicações de granularidade mais fina. Isto seria alcançado pela posssibilidade de coletar informação sobre a largura de banda entre os recursos e para aplicações desta classe (granularidade fina) apenas recursos interligados por uma largura de banda mínima (a ser definida) seriam utilizados.

6.1.2 Serviço de Balanceamento de Carga

O serviço de balanceamento de carga é de grande valia na computação distribuída em geral, sendo da mesma forma, importante para a grade computacional. Este serviço garante um melhor aproveitamento dos recursos, distribuindo de maneira adequada os recursos disponíveis, impedindo que recursos sejam sobrecarregados enquanto que outros estejam ociosos. O balanceamento de carga pode ocorrer apenas na inicialização da computação ou também durante a sua execução. Esta última é muito mais complexa e exige o suporte à migração de processos. O desafio na migração de processos consiste em manter a consistência entre o ambiente de origem e de destino.

O sistema **ProGrid** se beneficiaria em muito com a utilização de um serviço de balanceamento de carga, dado que o padrão MPI não oferece este tipo de suporte. O MPI utiliza um mecanismo fixo para determinar os recursos que serão utilizados, ou seja, a

alocação é feita de forma seqüencial e cíclica *(round-robin)* entre os recursos disponíveis no arquivo de configuração *(machines.LINUX)*.

A implementação do serviço de balanceamento de carga depende do serviço de monitoramento descrito na seção 6.1.1, pois as informações referentes ao estado da grade, bem como dos recursos que a compõem a grade são de responsabilidade do serviço de monitoramento. Tais informações servem de parâmetro na definição dos recursos que serão utilizados numa determinada computação.

6.2 Considerações Finais

Este trabalho apresentou o sistema **ProGrid**, que utiliza uma abordagem baseada em servidores Proxy para oferecer uma infra-estrutura de grade computacional que oferece suporte à programação e execução de aplicativos paralelos, além do compartilhamento seguro de recursos. Apesar de terem sido coletados poucos resultados de testes para comprovar as expectativas previstas com a abordagem apresentada, o conceito apresentado, juntamente com o refinamento e a inclusão de novos serviços pode resultar numa grade computacional robusta.

As contribuições alcançadas [73] justificam a realização do trabalho, destacando-se:

- A abordagem diferenciada no suporte a interface de programação MPI;
- A modelagem da arquitetura em camadas, prevendo os principais serviços da grade;
- O desenvolvimento do módulo de segurança em 3 níveis.

Uma nova fase de testes (estudo de caso) torna-se necessária para verificar se os requisitos de segurança, transparência e simplicidade no suporte a aplicações MPI e no controle do ambiente formado pelo **ProGrid**, foram totalmente alcançados. Isso comprovaria com dados a viavilidade da arquitetura e da interface propostas.

Para consolidar e validar a utilização do **ProGrid** de maneira unânime, também são necessários testes para verificar a capacidade de manipulação de eventos por parte do servidor Proxy, estabelecendo-se uma configuração mínima para o mesmo, dado um conjunto de

requisitos. Ainda nesse sentido, podem ser estudados os gargalos presentes no sistema e as situações que levam a esse estado.

Assim, a partir desse trabalho, almeja-se alcançar uma infra-estrutura de sítios conectados e controlados através de servidores Proxy. Sobre os recursos computacionais disponíveis, aplicações paralelas poderão ser desenvolvidas, utilizando computadores e dispositivos de armazenamento de maneira transparente, com balanceamento de carga, serviços de monitoramento, tolerância a falhas e mecanismos de segurança.

Referências Bibliográficas

[1] Foster, I., "Internet Computing and the Emerging Grid", Dezembro de 2000. Publicado em: Nature Web Matters. URL: http://www.nature.com/nature/webmatters/grid/grid.html - Visualizado em 20/04/2003.

[2] CERN - European Organization for Nuclear Research. URL: http://www.cern.ch - Visualizado em 20/04/2003.

[3] Entropia – PC Grid Computing. URL: http://www.entropia.com - Visualizado em 20/04/2003.

[4] SETI@home - The Search for Extraterrestrial Intelligence (SETI). URL: http://www.setiathome.ssl.berkeley.edu - Visualizado em 20/04/2003.

[5] Parabon Computation, Inc. URL: http://www.parabon.com - Visualizado em 20/04/2003.

[6] Fight Aids at Home – Computing toward a cure. URL: http://www.fightaidsathome.org - Visualizado em 20/04/2003.

[7] Foster, I. et al., "Software Infrastructure for the I-WAY High Performance Distributed Computing Experiment" in Proc. 5th IEEE Symposium on High Performance Distributed Computing. pp. 562-571, 1997.

[8] MPI – The Message Passing Interface Standard – URL: http://www-unix.mcs.anl.gov/mpi/index.html - Visualizado em 20/04/2003.

[9] SSL 3.0 Specification – URL: http://wp.netscape.com/eng/ssl3 - Visualizado em 20/04/2003.

[10] Johnston, W. and Brooke, J., "Core Grid Functions: A minimal Architecture for Grids", Working Draft, version 3.1, Julho de 2002. Disponível em: http://www.ggf.org/ogsa-wg - Visualizado em 20/04/2003.

[11] Moore, G. E., "Cramming More Components Onto Integrated Circuits", *Electronics*, Volume 38, Número 8, April 19, 1965. – URL: ftp://download.intel.com/research/silicon/moorespaper.pdf - Visualizado em 20/04/2003.

[12] Parhami, B., "*Introduction to Parallel Processing – Algorithms and Architectures"*, Fisrt Edition, Plenun Press, 1999.

[13] Almasi, G. S. and Gottlieb, A., *"Highly Parallel Computing"*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1989.

[14] Culler, D. E., Singh, J. P. and Gupta, A., *"Parallel Computer Architecture – A hardware/software approach"*, San Francisco, CA, Morgan Kaufmann, 1999.

[15] Flynn, M., *"Some computer organizations and their effectiveness"*, IEEE Transactions in Computing, C-21, p. 94, 1972.

[16] Hwang, K. and Briggs, F. A., *"Computer Architecture and Parallel Processing"*, McGraw Hill, International Edition, 1985.

[17] Parhami, B., *"Introduction to Parallel Processing – Algorithms and Architectures"*, Fisrt Edition, Plenun Press, 1999.

[18] Bräunl, T., *"Parallel Programming: an Introduction"*, Prentice Hall, 1993.

[19] Hwang, K., Xu, Z., *"Scalable Parallel Computing: Technology, Architecture, Programming"*, Ed. MacGraw-Hill, 1998.

[20] I. Foster and C. Kesselman, *"The Grid: Blueprint for a New Computing Infrastructure"*, Ed. Academic Press, 1998.

[21] Beowulf – The Beowulf Cluster Site. URL: www.beowulf.org - Visualizado em 20/04/2003.

[22] May, M. D., Thompson, P. W. and Welch, P. H., *"Networks, Routers and Transputers: Function, Performance, and Application"*, IOS Press, pp. 210, 1994.

[23] Inmos Ltd, *occam 2 Reference Manual*, Prentice Hall, pp. 133, 1988.

[24] Hoare, C. A. R., "Communicating Sequential Processes", Prentice Hall, pp. 256, 1985.

[25] Morrison, R. S., *"Cluster Computing: Architectures, Operating Systems, Parallel Processing & Programing Languages"*, Janeiro de 2002. Disponível em: http://www.netspace.net.au/~morri/cct.html - Visualizado em 20/04/2003.

[26] PVM – Parallel Virtual Machine. URL: http://www.csm.ornl.gov/pvm/pvm_home.html - Visualizado em 20/04/2003.

[27] MPICH – A Portable MPI Implementation. URL: http://www-unix.mcs.anl.gov/mpi/mpich - Visualizado em 20/04/2003.

[28] G. A. Geist, J. A. Kohl, P. M. Papadopoulos, *"A Comparison of Features"*, Calculateurs Paralleles, Vol. 8, No. 2, Maio de 1996. Disponível em: http://www.epm.ornl.gov/pvm/PVMvsMPI.ps - Visualizado em 20/04/2003.

[29] LAM / MPI Parallel Computing. URL: http://www.lam-mpi.org - Visualizado em 20/04/2003.

[30] Top500 Supercomputer Sites. URL: http://www.top500.org/list/2002/11 - Visualizado em 20/04/2003.

[31] Kacsuk, P. and Vajda, F., *"Network-based Distributed Computing (Metacomputing)"*, Prospective Report, ERCIM 1999. Disponível em: http://www.ercim.org/publication/prosp/NdBC.pdf - Visualizado em 20/04/2003.

[32] Baker, M. A., et al. *"The Evolution of the Grid"*, Universities of Portsmouth and Southampton, UK, Publicado em: Grid Computing: Making the Global Infrastructure a Reality, Publisher: Wiley, John & Sons, Incorporated. Disponível em: http://www.grid2002.org - Visualizado em 20/04/2003.

[33] FAFNER - Welcome to the RSA Factoring-By-Web project. URL: http://www.npac.syr.edu/factoring.html - Visualizado em 20/04/2003.

[34] Buya, R., *"Economic-based Distributed Resource Management and Scheduling for Grid Computing"*, Melbourne, Austrália, Abril de 2002, 180p., Dissertação de Doutorado, School of Computer Science and Software Engineering Monash University.

[35] National Grand Challenge Applications – URL: http://www.itrd.gov/pubs/blue02/national-grand.html - Visualizado em 20/04/2003.

[36] Berman, F., Hey, T. and Fox, G., *"The Grid: Past, Present, Future"*, Publicado em: Grid Computing: Making the Global Infrastructure a Reality, Publisher: Wiley, John & Sons, Incorporated. Disponível em: http://www.grid2002.org - Visualizado em 20/04/2003.

[37] Distributed.net. URL: http://distributed.net - Visualizado em 20/04/2003.

[38] RSA Security – Web, Managemente and Internet E-Security. URL: http://www.rsasecurity.com - Visualizado em 20/04/2003.

[39] Hardt, J. and White K., "Distributed Interactive Simulation (DIS)", EEL 4781 Computer Networks, University of Central Florida, Fall 1998.

[40] M. J. Litzkow, L. Bergman and P. Li and W. Mutka, *"Condor – a hunter of idle workstations"*, 8ª Conferencia Internacional de Sistemas de Computação Distribuída, páginas 104-111, Junho de 1988.

[41] NVO - National Virtual Observatory. URL: http://www.us-vo.org - Visualizado em 20/04/2003.

[42] J. Czyzyk, M. P. Mesnier and J. J. Moré, *"The Network-Enabled Optimization System (NEOS) Server"*, Argone National Laboratory, Argonne, Illinois, 1996.

[43] H. Casanova and J. Dongarra, *"Netsolve: A network Server for solving computational science problems"*, Technical Report CS-95-313, Universidade do Tennessee, Novembro de 1995.

[44] C. Lee, C. Kesselman and S. Schwab, *"Near-realtime satellite image processing: Metacomputing in CC++"*, IEEE Computer Graphics and Applications, Vol 16, No.4, pag. 79-84, 1996.

[45] Marzullo, K., et al., *"NILE: Wide-area computing for high energy physics"*, ACM SIGOPS European Workshop, Connemara, Ireland, 1996.

[46] Roussos, M., et al., *"NICE: Combining constructionism, narrative, and collaboration in a virtual learning environment"*, ACM SIGGRAPH Computer Graphics, Vol 31, Issue 3, Agosto de 1997.

[47] Groove Network, Inc., Desktop Collaboration Software. URL: http://www.groove.net - Visualizado em 20/04/2003.

[48] Open Grid Services Architecture Working Group (OGSA-WG). URL: www.ggf.org/ogsa-wg - Visualizado em 20/04/2003.

[49] Johnston, W. and Brooke, J., *"Core Grid Functions: A minimal Architecture for Grids"*, Open Grid Service Architecture Working Group, Working Draft, version 3.1, Julho de 2002. Disponível em: www.ggf.org/ogsa-wg - Visualizado em 20/04/2003.

[50] The DataGrid Project. URL: http://eu-datagrid.web.cern.ch/eu-datagrid - Visualizado em 20/04/2003.

[51] D. Abramson, et al., "Nimrod: A tool for performing parameterised simulations using distributed workstations", In *Proc. 4th IEEE Symp. on High Performance Distributed Computing,* IEEE Press, 1995.

[52] Load Sharing Facility (LSF). URL: http://accl.grc.nasa.gov/lsf - Visualizado em 20/04/2003.

[53] Nagaratnam N., et al., *"The Security Architecture for Open Grid Services"*, Open Grid Service Architecture Security Working Group, version 1, Julho de 2002. Disponível em: http://www.cs.virginia.edu/~humphrey/ogsa-sec-wg - Visualizado em 20/04/2003.

[54] Kohl, J. and Neuman, C., "*The Kerberos network authentication service (v5)*", Internet RFC 1510, Setembro de 1993.

[55] Carvalho, D. B., *"Segurança de Dados com Criptografia: Métodos e Algoritmos"*, Ed. Book Express, 2000.

[56] Foster, I., et al., "A Security Architecture for Computational Grids", Proc. 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998.

Disponível em: ftp://ftp.globus.org/pub/globus/papers/security.pdf - Visualizado em 20/04/2003.

[57] Wulf, W. A., Wang, C., Kienzle, D., "A New Model of Security for Distributed Systems", Computer Science Technical Report CS-95-34, University of Virginia Agosto de 1995.

[58] The Globus Project. URL: http://www.globus.org - Visualizado em 20/04/2003.

[59] CORBA, Welcome to the OMG's CORBA Website. URL: http://www.corba.org - Visualizado em 20/04/2003.

[60] Neumann, B. C. and Rao, S., "The Prospero resource manager: A scalable framework for processor allocation in distributed systems", *Concurrency: Practice and Experience*, June 1994.

[61] Morris, J.H., et al., "Andrew: A distributed personal computing environment", *Communications of the ACM*, Vol. 29, No. 3, pp 184- 201, Março de 1986.

[62] Cook, J., et al., "Truffles: Secure file sharing with minimal system administrator intervention", Proceedings of the 1993 World Conference on Tools and Techniques for System Administration, April 1993.

[63] Foster, I., Kesselman, C. and Tuecke, S., "The Nexus approach to integrating multithreading and communication", *Journal of Parallel and Distributed Computing*, 1996.

[64] Cruz-Neira, C., et al., "The CAVE: Audio visual experience automatic virtual environment", *Communications of the ACM*, Vol. 35, Issue 6, 1992.

[65] Foster, I. and Kesselman, C., "The Globus Project: A Status Report" *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp. 4-18, 1998.

[66] Legion: A Worldwide Virtual Computer. URL: http://legion.virginia.edu - Visualizado em 20/04/2003.

[67] Gabriel, E., Resch, M., Beisel, T. and Keller, R., "Distributed computing in a heterogenous computing environment", *EuroPVMMPI'98 Liverpool/UK*, 1998.

Disponível em: http://www.hlrs.de/organization/pds/projects/pacx-mpi/papers/liverpool98.ps.gz - Visualizado em 20/04/2003.

[68] Karonis, N., Toonen, B. and Foster, I., "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface", *Journal of Parallel and Distributed Computing*, a ser publicado em 2003. Disponível em: http://www.globus.org/research/papers/mpich-gei03.pdf - Visualizado em 20/04/2003.

[69] SNMP – Research. URL: http://www.snmp.org - Visualizado em 20/04/2003.

[70] WebServices.ORG – The Web Services Industry Portal. URL: http://www.webservices.org - Visualizado em 20/04/2003.

[71] OpenSSL: The open Source Toolkit for SSL/TSL. URL: http://www.openssl.org - Visualizado em 20/04/2003.

[72] Stevens, W. R., *"Unix Network Programming"*, Ed. Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[73] Costa, P. V. C., Zorzo, S. D., Guardia, H. C., "An Architecture for Computational Grids Based on Proxy Servers", a ser publicado em Junho de 2003 no 1st International Workshop on Middleware for Grid Computing, Rio de Janeiro, Brasil. Disponível em: http://virtual01.lncc.br/mgc2003 - Visualizado em 02/05/2003.

APÊNDICE A

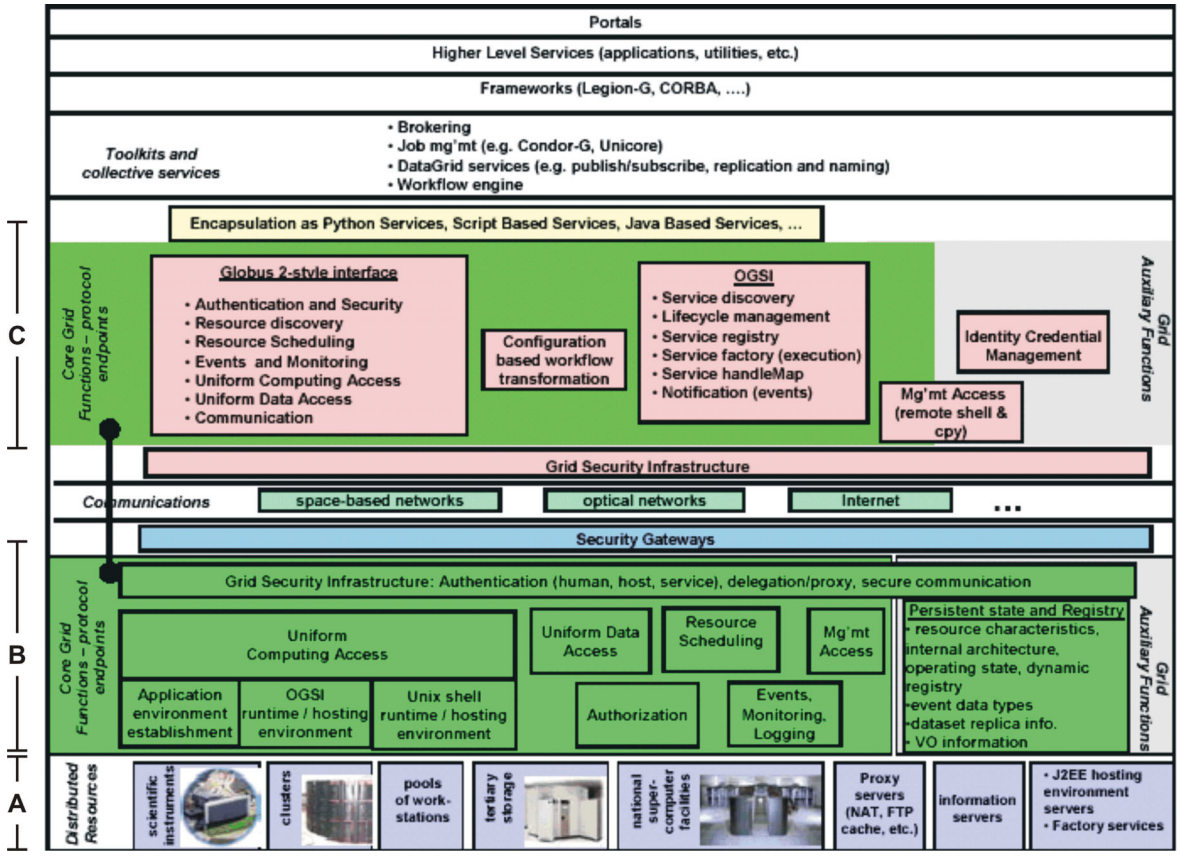

Figura 16: Ilustração da Arquitetura da grade na visão de serviços

As seções A, B e C da Figura 16 ilustram a composição da grade computacional e a interação entre seus componentes.

A seção da Figura 16 demarcada pela letra "A", representa os diversos recursos de hardware que compõem a grade e localizam-se na parte mais inferior da arquitetura. A seção B representa a *middleware* do sistema responsável por manipular e coordenar os diversos recursos de hardware. Esta *middleware* cria uma abstração entre os recursos de hardware e o restante da grade. As seções A e B são interconectadas à seção C através de diferentes tecnologias de rede. Como a grade pode ter diversas topologias de interconexão, esta seção é responsável por esconder tais características das aplicações, possibilitando a comunicação entre os diversos dispositivos e processos.

A seção C corresponde à outra *middleware* de funcionalidades. Neste caso, ela atua como porta de entrada para a grade, oferecendo funções como: agendamento de tarefas, localizador de recursos, autenticação, entre outros. Por fim, temos a seção D que corresponde às aplicações, portais e usuários que utilizam os recursos fornecidos pela grade de maneira transparente. As seções A, B e C da Figura 16 interagem de modo a proporcionar a visão de uma única máquina virtual para a camada superior. Este é o objetivo das grades computacionais: aglomerar diversos recursos heterogêneos disponibilizando-os para as aplicações de maneira uniforme.

As camadas apresentadas na Figura 16, constituem uma representação da proposta de padronização apresentada pelo OGSI. É possível notar neste esquema, a elevada semalhança com o projeto Globus, que serviu de base para a definição inicial de tal arquitetura.

APÊNDICE B

Secure Socket Layer (SSL)

A necessidade de proteger as informações que trafegam em canais de comunicação públicos e também de prevenir a falsificação de identidade com intuito de roubar dados conduziu ao desenvolvimento de padrão aberto para transmissão segura de dados através da Internet.

Desenvolvido pela *Netscape Communications* em 1994, o SSL foi o primeiro protocolo para proteger transações realizadas pela Web. O protocolo SSL descreve um método, composto por um conjunto de regras responsável por autenticar e criptografar a comunicação entre clientes e servidores. A criptografia de chave pública (Public Key Infrastructure – PKI) constitui a tecnologia base sobre a qual o protocolo SSL foi desenvolvido. O PKI é uma infra-estrutura que utiliza a criptografia de chave pública (assimétrica) para verificar os certificados, possibilitando a seus usuários o estabelecimento de uma conexão segura mediados por uma autoridade confiável.

Vantagens:

As principais características do protocolo SSL, que o levaram a ser um padrão amplamente difundido, sendo o protocolo mais utilizado para a transmissão segura de dados na Internet, são:

- **Solução Stand-Alone:** como o SSL não foi construído dentro de uma aplicação, mais sim como uma camada de acesso padrão, ele pode suportar outros protocolos de aplicação;
- **Protocolo Flexível:** o SSL está apto a suportar um amplo conjunto de protocolos de aplicação, incluindo http, FTP a NNTP. Além disso, o comportamento interno dos protocolos de aplicação não é alterado quando a segurança é adicionada;
- **Padrão da Indústria:** com a vantagem de ser o primeiro do mercado, o SSL tornou-se um padrão universal para segurança de informações e de transações Web. Suportado por todos os principais navegadores e servidores, o SSL é um protocolo confiável.

As Funções do Protocolo SSL

O objetivo do SSL é verificar a identidade das entidades envolvidas em uma transação segura e garantir que a transmissão é protegida contra interceptação (ou seja, caso interceptado por entidades não autorizadas, não deve ser legível). As principais funções providas pelo protocolo SSL são:

- **Autenticação do Servidor:** permite ao usuário confirmar a identidade do servidor envolvido na transação. Isto é alcançado utilizando-se técnicas de chave pública que verifica se o certificado do servidor é válido e assinado por uma autoridade de certificação confiável;
- **Autenticação do Cliente:** permite ao servidor confirmar a identidade do cliente do mesmo modo descrito para a autenticação do servidor. Este tipo de autenticação pode ser usado por bancos para garantir a autenticidade da identidade do cliente;
- **Conexão Criptografada:** cria um canal de comunicação seguro entre duas entidades. O SSL coordena o processo de criptografia e descriptografia de todas as informações transmitidas por este canal, bem como o estabelecimento do mesmo.

Arquitetura do Protocolo SSL:

O protocolo SSL foi desenvolvido para ser usado sobre a camada TCP provendo um serviço de comunicação seguro e confiável fim-a-fim. O SSL não é um simples protocolo, mas sim 2 camadas de protocolo, conforme ilustrado na Figura 17.

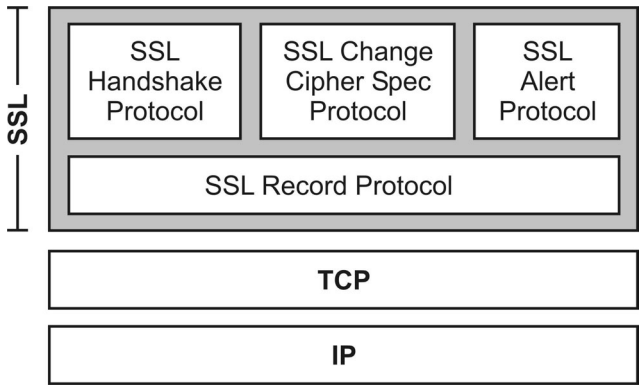

Figura 17: Divisão em camadas do Protocolo SSL

O SSL Record Protocol fornece os serviços básicos de segurança para os vários protocolos da camada superior. Os outros três protocolos definidos como partes do SSL são: o *Handshake Protocol*, o *Change Cipher Spec Protocol* e o *Alert Protocol*. Esses protocolos são utilizados no gerenciamento do SSL.

Dois conceitos importantes a serem observados no SSL são: sessão SSL e conexão SSL, definidos como:

- **Sessão:** a sessão SSL é a criação de um canal virtual entre um cliente e um servidor. As sessões são criadas pelo *Handshake Protocol* e definem um conjunto de parâmetros de criptografia, que podem ser compartilhados entre múltiplas conexões. As sessões são utilizadas para evitar a sobrecarga na negociação de novos parâmetros de criptografia para cada conexão.
- **Conexão:** a conexão é utilizada para criar diversos sub-canais dentro de uma sessão, aproveitando os parâmentros de segurança definidos nesta sessão;

Entre um par de entidades pode haver múltiplas conexões seguras. Na teoria, múltiplas sessões entre um par de entidades também podem ser utilizadas, mas essa é uma característica que não é usada na prática.

Durante o ciclo de vida de uma sessão, dois estados podem ser associados à mesma. Uma vez estabelecida a sessão, existe um estado de operação *corrente* para ambos leitura e escrita (enviar e receber). Durante o estabelecimento da sessão (Handshake Protocol) o estado *pendente* para leitura e escrita é criado. Se a conexão for estabelecida com sucesso, o estado *pendente* passa para *corrente*.