Uma abordagem *peer-to-peer* de grade computacional com balanceamento de carga

Érico Casella Tavares de Mattos[1]*, **Luis Carlos Trevelin**[1]

[1]Grupo de Sistemas Distribuídos e Redes – Departamento de Computação
Universidade Federal de São Carlos
Rodovia Washington Luis, Km 235 – Caixa Postal 676 – 13565-905 São Carlos, SP

erico@dc.ufscar.br, trevelin@dc.ufscar.br

Abstract. *Some characteristics contributed for the success of file sharing peer-to-peer systems, including scalability and use easiness. Trying to reach the same results, adding the processing capacity of some low cost computers, appears the idea to adopt this same model building computational grid environments, allowing the construction of high performance parallel computation environments with lower financial and administrative costs. This paper presents a software architecture composed by distributed resources management mechanisms, created motivated by the services offered by computational grid environment and the implantation and execution easiness of peer-to-peer systems.*

Resumo. *Algumas características contribuíram muito para o sucesso de sistemas peer-to-peer para compartilhamento de arquivos, incluindo escalabilidade e facilidade de uso. Objetivando alcançar os mesmos resultados, agregando a capacidade de processamento de vários computadores de pequeno porte, surge a idéia de adotar esse mesmo modelo para contrução de ambientes de grades computacionais, permitindo construir ambientes de computação paralela de alto desempenho com menores custos financeiros e administrativos. Este artigo apresenta uma arquitetura de* software *composta por mecanismos de gerência de recursos distribuídos, motivado pelos serviços oferecidos por um ambiente de grade computacional e pela facilidade de implantação e execução de sistemas* peer-to-peer.

1. Introdução

Aplicações com alto grau de complexidade exigem sistemas computacionais com grandes capacidades de processamento, memória e armazenamento. A necessidade desses sistemas levou à criação de diversos conceitos para solucionar problemas de grande escala. Um desses conceitos é a computação paralela e distribuída de alto desempenho.

Com a evolução das tecnologias para criação de redes de computadores, surgiu uma nova abordagem para a computação paralela e distribuída de alto desempenho. Essa abordagem oferece alto desempenho através da conexão de computadores, utilizando as capacidades computacionais de cada componente da rede. Exemplos de soluções que utilizam essa abordagem são os sistemas de computação paralela e distribuída baseados em aglomerados[Baker, 2000] e grades de computadores[Foster et al., 2001, Rajkumar, 2002].

Está sendo desenvolvido um trabalho focado nos conceitos envolvidos na computação de grades de computadores, cuja utilização não se restringe simplesmente

*Apoiado por CAPES.

à computação paralela e distribuída. Dentre as variações existentes, pode-se destacar os ambientes baseados em grades que oferecem serviços de dados e serviços para aplicações e informações específicas do domínio de um problema.

As grades destinadas à computação paralela e distribuída estão associadas ao compartilhamento da capacidade de processamento de recursos geograficamente espalhados e conectados por uma rede. Essas grades são chamadas de grades computacionais e oferecem serviços de processamento de tarefas, podendo ser aplicadas à solução de grandes computações, as quais englobam processos compostos por cálculos complexos, encontrados em aplicações das mais diversas áreas.

Com a motivação das altas capacidades e facilidades oferecidas por ambientes de grade computacional às aplicações que necessitam de alto desempenho e alta disponibilidade, este trabalho apresenta os resultados de pesquisa concentrada na apresentação dos principais conceitos desses sistemas distribuídos em grades computacionais, por meio de uma abordagem para construção de uma grade computacional cujas características diferem das observadas em trabalhos relacionados ao mesmo assunto, colocando em discussão e, em destaque, os pontos essenciais relativos à implementação e ao desenvolvimento desses sistemas.

2. Computação de Grade - *Grid Computing*

Com a evolução dos aglomerados de computadores e da Internet, surgiram os sistemas paralelos e distribuídos baseados em computação de grade[Foster et al., 2001, Rajkumar, 2002, Chetty and Buyya, 2002]. Esses sistemas utilizam a grande abrangência da Internet para ampliar a distribuição e permitir a conexão de um número maior de recursos computacionais heterogêneos. Tal heterogeneidade refere-se a localidade e aos tipos dos recursos. A localidade está associada ao fato de sistemas de grades de computadores englobar recursos pertencentes a diferentes redes e proprietários. A heterogeneidade de tipos de recursos pode ser considerada superior à dos aglomerados, pela possibilidade de existirem diferentes dispositivos, desde computadores pessoais e aglomerados, até pequenos dispositivos e sensores.

As grades de computadores têm a capacidade de agregar diferentes dispositivos com o mínimo de gasto em configurações. Isso é possível com a utilização de componentes intermediários (ou middleware) capazes de estabelecer ligações entre aplicações e diversos dispositivos e computadores.

O termo computação de grade é uma nova nomenclatura dada ao compartilhamento de recursos geograficamente distribuídos em redes interligadas[Rajkumar, 2002], podendo também ser chamado de metacomputação, computação em malha, scalable computing, computação global e Internet computing.

Uma plataforma de computação de grade pode ser usada com diferentes tipos de aplicações. Como foi reapresentado em [Rajkumar, 2002], aplicações em ambientes de grades podem ser categorizadas em cinco classes principais: supercomputação distribuída, de grande fluxo de dados, sob-demanda, de grande quantidade de dados e colaborativa.

2.1. Variações das grades

Como apresentado na grande maioria dos trabalhos relacionados com computação baseada em grade, e seguindo uma mesma linha conceitual [Rajkumar, 2002, Foster et al., 2001], as grades, pelo ponto de vista do usuário final, provêem um ou mais

tipos de serviços, os quais são divididos basicamente em quatro grupos: serviços computacionais, serviços de dados ou armazenamento, serviços de aplicação e serviços compostos, este último incluindo outros serviços como: repositório de códigos e catálogo de informações e conhecimentos.

2.2. Princípios e Características

Ao avaliar e desenvolver sistemas em grade, alguns princípios[Buyya et al., 2000, Rajkumar, 2002] diferem dos já observados ao construir aglomerados e outros sistemas distribuídos, outros são comuns às diferentes categorias. Podendo destacar os seguintes princípios: heterogeneidade de recursos, variabilidade de dimensão (scalability), dinamismo e adaptabilidade, existência de múltiplos domínios administrativos, exigência de alta disponibilidade, flexibilidade e extensibilidade, uso de nomeação única e global, facilidade no uso e acesso transparente, alto desempenho e segurança.

Para a efetivação dos serviços de processamento e armazenamento, anteriormente definidos, é necessário o desenvolvimento de alguns mecanismos que funcionam como suporte, incluindo segurança, diretório, alocação de recursos, sistema de informação, pagamento e serviços para desenvolvimento de aplicações, gerenciamento de execuções e agendamento de tarefas. Desses mecanismos, algumas características relevantes ao desenvolvimento de uma grade podem ser destacadas[Rajkumar, 2002, Foster et al., 2001]: hierarquia administrativa global, serviços de comunicação, informação e nomeação unificados, sistema de arquivos distribuídos e (Caching), segurança e autorização, monitoração, diagnóstico e tolerância a falhas, gerenciamento de recursos e agendamento e escalonamento de tarefas, economia computacional e negociação de recursos, diferentes paradigmas e ferramentas de programação, além de diversas interfaces gráficas administrativas e de uso.

3. Ambiente de grade computacional *peer-to-peer* com balanceamento de carga

Observando a possibilidade de convergência entre grades computacionais e sistemas *peer-to-peer* [Foster and Iamnitchi, 2003, Foster, , Foster et al., 2001], foi desenvolvido um trabalho, originado de uma extensão do TLBA[Mello et al., 2003], que visa o desenvolvimento de mecanismos e ferramentas para a criação de uma rede de recursos, que oferecem e utilizam serviços computacionais. Esses mecanismos são compostos por algoritmos e regras que definem o comportamento de serviços gerenciadores dos recursos, e as ferramentas implementam um conjunto de mecanismos e são utilizadas pelos gerenciadores para controlar os recursos de maneira eficiente e segura.

Os serviços computacionais permitem aos computadores participantes da rede executarem tarefas nos demais, e o ambiente capaz de gerenciar a oferta e procura desses serviços é chamado de grade computacional. Nesses ambientes, o estudo de técnicas de balanceamento de carga é essencial quando se deseja que os gerenciadores distribuam igualmente requisições de tarefas nos recursos que oferecem os serviços computacionais.

Um ambiente de grade computacional pode gerenciar recursos distribuídos e interligados por uma rede, seguindo diferentes modelos de comunicação. Um desses modelos é o *peer-to-peer*, que define uma rede composta por componentes com igual capacidade de oferecer e utilizar serviços. Em outras palavras, assemelha-se a uma rede seguindo o modelo cliente-servidor, cujos participantes são clientes e servidores dos demais.

O desenvolvimento de um ambiente de grade computacional com balanceamento de carga exige uma série de estudos. Dentre eles é possível destacar: modelos de

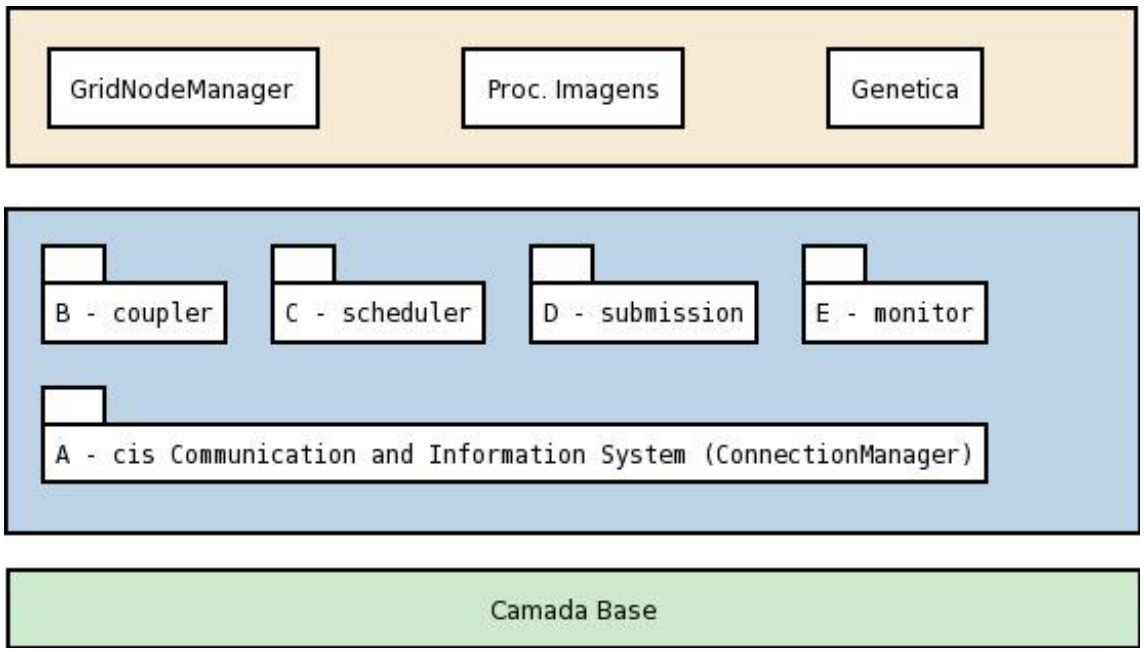


Figure 1: Arquitetura em camadas do sistema em desenvolvimento

comunicação e interconexão de recursos, política de troca de mensagens, escalonamento dinâmico de tarefas com balanceamento de carga, mecanismos de submissão de tarefas, carga dinâmica de códigos, segurança na comunicação, redirecionamento de primitivas de entrada e saída e monitoração de recursos.

3.1. Visão geral

Para a construção do ambiente de grade computacional idealizado, foram implementados alguns componentes responsáveis por mecanismos que gerenciam operações em recursos da grade. Esses componentes são agrupados em uma camada intermediária de controle do ambiente, a qual utiliza serviços da camada inferior denominada base e oferece serviços à camada superior composta por aplicações dos usuários.

Na camada intermediária (ver figura 1) concentram-se os componentes responsáveis por todos serviços de controle necessários para manter o ambiente de grade computacional ativo. Um ambiente de grade está ativo quando seus recursos mantêm o controle das interligações existentes entre eles, estando aptos a atender requisições de serviços de usuários.

Os componentes da camada intermediária abstraem mecanismos gerenciadores específicos, incluindo acoplador, escalonador, disparador, executor e monitor. Há ainda os componentes que representam o sistema de comunicação e o sistema de submissão de trabalhos.

3.2. Sistema de comunicação

Para manter comunicação consistente entre os componentes heterogêneos da rede é importante o estudo de políticas de trocas de mensagens. Esse estudo envolve mecanismos, técnicas e protocolos para facilitar e garantir operações entre todos os recursos da rede, respeitando suas particularidades. A idéia é criar um sistema de comunicação capaz de tornar os serviços do ambiente de grade computacional independentes do meio de comunicação. Esse fato é semelhante às abstrações realizadas por cada camada de rede (OSI - The Open Systems Interconnection Reference Model), com o objetivo de permitir a comunicação entre camadas do mesmo nível, independentemente das demais.

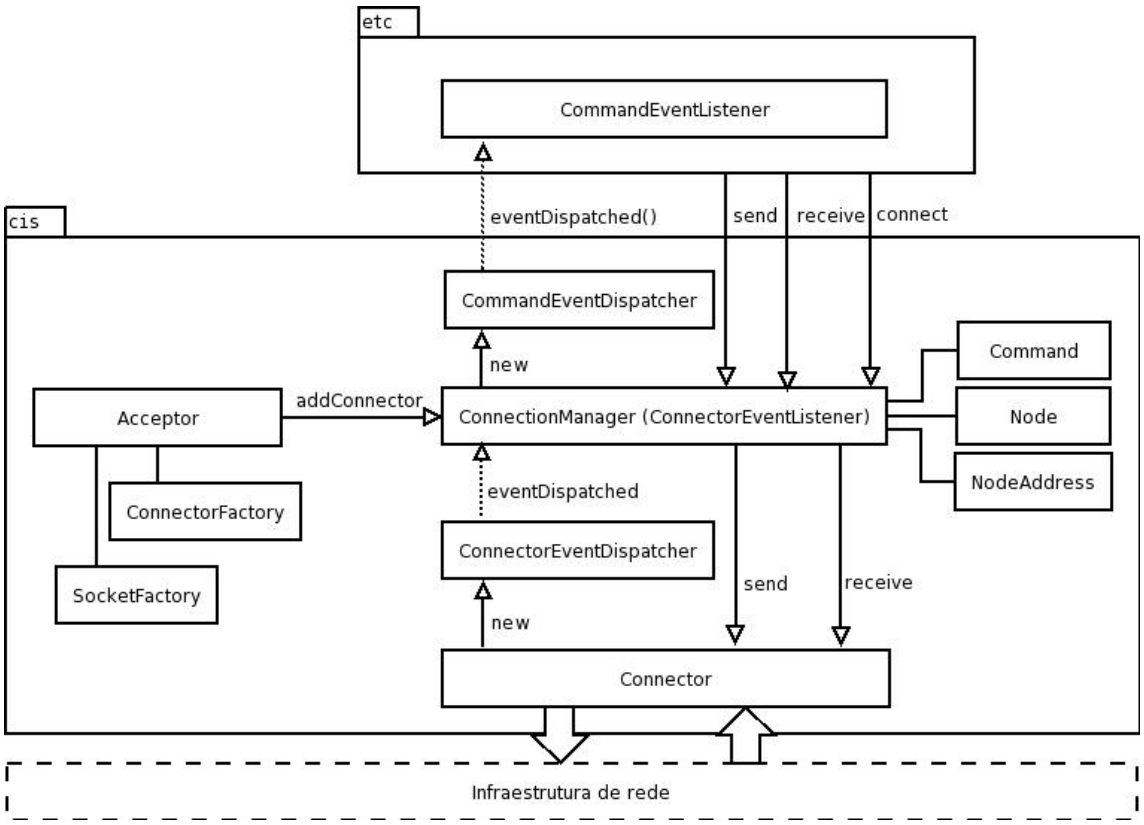


Figure 2: Componentes de implementação do sistema de comunicação e informação do P2PGrid

A arquitetura do sistema de comunicação proposto prevê a existência de alguns componentes que devem ser implementados conforme os padrões e as tecnologias de comunicação adotados, permitindo compatibilidade entre eles. O principal módulo é o gerenciador de conexões, o qual se responsabiliza pelo controle de todas conexões de um nó com seus adjacentes, além de fornecer aos serviços básicos de controle do middleware meios para receber e enviar comandos. A estrutura do sistema de comunicação, mostrada na figura 2, contém ainda o componente *Connector* para garantir a flexibilidade de implementação com diferentes protocolos, e os componentes *Dispatcher* baseado em um padrão de projeto de entrega de mensagens multi-tarefa.

A integração entre o gerenciador de conexões e os serviços da grade é garantida através de primitivas bloqueantes de envio e recebimento de comandos ou através da associação de listeners para cada tipo de comando. No caso dos listeners, as tarefas interessadas em determinados comandos se registram e são avisadas das chegadas dos objetos através de eventos. Os módulos Servidor e Cliente são utilizados para a criação de novas conexões com os nós, associando-as ao gerenciador de conexões correspondente.

3.3. Modelo de interconexão de recursos e acoplamento

O estudo de modelos de comunicação e interconexão de recursos permite conhecer e escolher melhores opções em relação à forma que cada participante da grade se comunica com os demais. Além disso, as diferentes ligações lógicas entre os recursos influenciam na visão que cada um tem dos outros. A união desses dois fatos afeta diretamente as características do ambiente de grade computacional, tais como, desempenho, escalabilidade, tolerância a falhas e alta disponibilidade.

Visando a criação de um modelo mais adequado a uma rede *peer-to-peer* destinada a serviços computacionais, cujas mensagens de controle focam na monitoração de

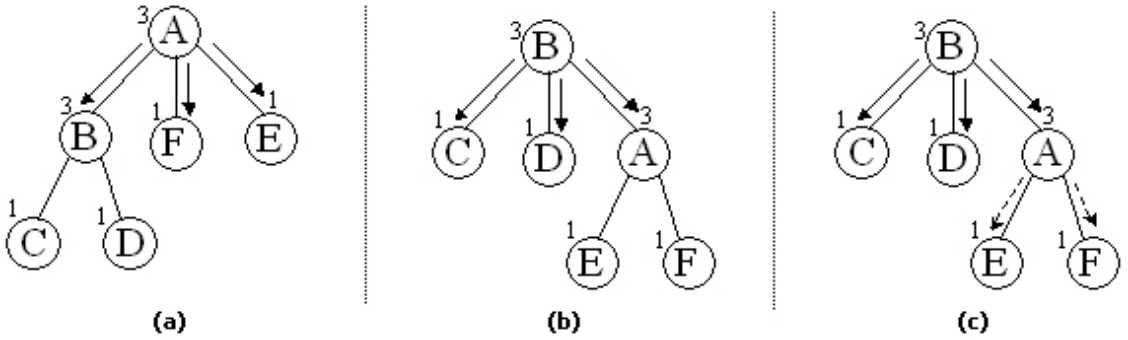


Figure 3: Organização dos recursos na topologia utilizada

recursos e no balanceamento de carga dos mesmos, chegou-se a uma rede cujos nós formam um grafo acíclico com comunicação bidirecional. Cada nó pertencente ao grafo é ligado com até N nós adjacentes, enviando e recebendo deles mensagens essenciais para o funcionamento do sistema. A existência de um limitante N, assim como a definição de seu valor, está ligada com o critério de balanceamento do grafo adotado. Na figura 3 é possível observar três situações de envio de mensagens em um mesmo grafo.

A utilização de uma estrutura *peer-to-peer* permite iniciar o processo de inclusão de um novo recurso à rede a partir de qualquer nó. Em outras palavras, quando um nó deseja fazer parte do grafo, ele requisita um processo de ingresso, também chamado de join, a qualquer outro integrante do grafo e este se responsabiliza por melhor posicioná-lo. Esse mecanismo está associado a diferentes políticas de balanceamento do grafo, o qual pode ser baseado na capacidade dos recursos ou no número de nós.

O modelo de comunicação proposto permite ainda desenvolver mecanismos que garantam a integridade do grafo mesmo havendo falhas de um ou mais nós. Esses mecanismos fazem parte do processo de reestruturação em casos de falhas e são implementados junto com o módulo acoplador.

3.4. Escalonamento de tarefas

Tendo uma rede de recursos interligados entre si e capazes de oferecer e utilizar serviços computacionais, é necessário utilizar mecanismos de escalonamento de tarefas. Esses mecanismos utilizam algoritmos e métodos para melhor alocar os recursos a cada processo. O escalonamento dinâmico de tarefas permite distribuir tarefas aos recursos obedecendo critérios que variam conforme a situação. O balanceamento de carga é uma técnica que determina esses critérios, com o objetivo de manter equilibrada a carga dos recursos.

O escalonador utilizado, iniciado de uma derivação do TLBA[Mello et al., 2004], está especialmente voltado à distribuição de tarefas entre os nós pertencentes à topologia lógica apresentada anteriormente. Em outras palavras, o escalonador utiliza informações de carga trocadas entre os recursos e permite selecionar nós trabalhadores sem a necessidade de consultar tabelas globais centralizadas.

Ao trabalhar com grade computacional, cuja dimensão tende a alcançar larga escala, torna-se inviável armazenar o índice de carga de todos componentes da rede, ou mesmo executar uma busca completa por todos os nós para encontrar o mais ocioso. A solução adotada é utilizar médias das cargas dos nós pertencentes a cada ramificação adjacente. Dessa forma, no momento de selecionar o nó trabalhador, cada um compara seu índice de carga local com as médias dos índices de cada ramificação adjacente, assumindo o trabalho ou repassando-no à ramificação com menor índice até um nó assumir o trabalho.

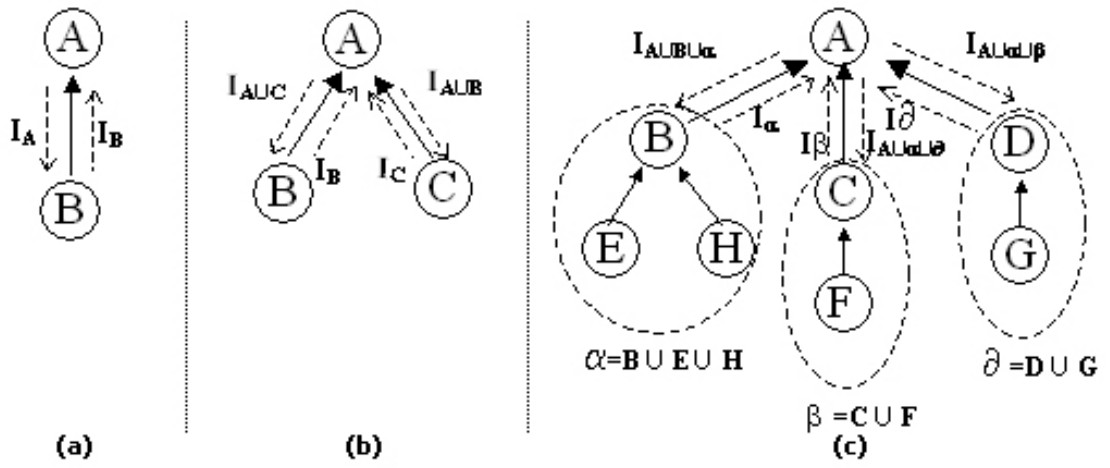
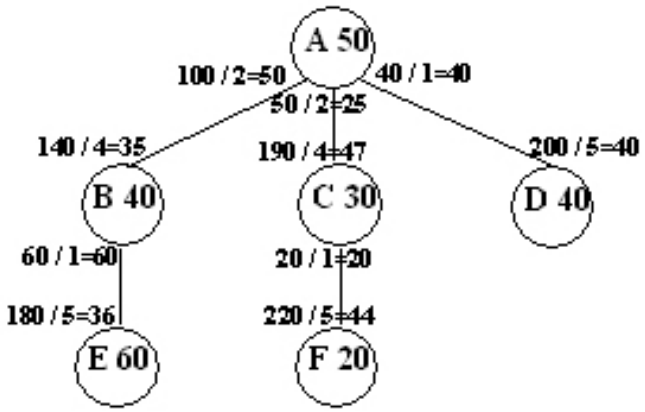


Figure 4: Mecanismo de troca de índices de carga utilizado no P2PGrid



DISPARA	EXECUTA
D	D
E	F
F	F
A	F
B	F
C	F

Figure 5: Efeito de alguns disparo de tarefas em um grafo da grade P2PGrid

A figura 4 apresenta o mecanismo de troca de índices de carga realizado entre os nós pertencentes à grade. Cada nó envia para seus adjacentes um índice de carga que representa a somatória de seu próprio índice e dos demais adjacentes, além da contagem total de nós. Dessa forma, cada integrante da grade terá a capacidade de calcular a média de carga de cada ramificação adjacente no momento de disparar tarefas.

É mostrado na figura 5 um quadro com situações de disparo de tarefas para a grade representada pelo grafo da própria figura. Esse quadro mostra a capacidade de encontrar um nó mais ocioso, mesmo trabalhando com médias. Essa postura garante que mesmo que a grade cresça para dimensões muito grandes de centenas de nós, a quantidade de mensagens de controle trocadas não crescerá proporcionalmente, e os nós conseguirão encontrar outros mais ociosos, partindo de uma busca pelos mais próximos.

3.5. Mecanismos de submissão de tarefas

Os mecanismos de submissão de tarefas são os componentes que mais influenciam na interação do usuário com o ambiente de grade computacional. Esses mecanismos definem os tipos e as formas que tarefas são submetidas ao ambiente. Além disso, define meios pelos quais as tarefas enviam e recebem, dos usuários e de dispositivos de entrada e saída, informações essenciais às suas execuções. Para garantir às tarefas todas informações necessárias, e aos usuários a resposta completa de seu processamento, os mecanismos de submissão utilizam outros serviços, incluindo, controle de carga dinâmica e redirecionamento de entrada e saída (E/S padrões, arquivos e chamadas ao sistema).

A primeira característica a ser avaliada ao distribuir trabalhos em um ambiente de grade é a capacidade do trabalho em cumprir sua função, dessa forma é possível categorizá-los basicamente em processos completos e pequenas tarefas.

Os processos completos compreendem àquelas aplicações que não dependem da execução de nenhum outro processo, sendo capazes de completar todo o processamento de um determinado problema. Normalmente a submissão desses processos assemelha-se com execuções de processos locais, fazendo apenas uma referência ao nome da aplicação a ser executada, a qual interage com dispositivos de entrada e saída locais ou remotos através de redirecionadores. Esse mecanismo permite executar um programa remotamente desenvolvido em Java, sem a necessidade de nenhuma mudança no mesmo, apenas na forma de executar o comando de início.

As tarefas normalmente são computações que, em conjunto com outros processamentos, completam a solução de algum problema específico. As tarefas podem ter diferentes características quanto à comunicação com outras tarefas, utilização de entrada e saída, uso de CPU e memória, entre outras, as quais justificam ou não a separação de uma aplicação em tarefas e a distribuição das mesmas.

Uma das metas do ambiente desenvolvido é abranger aplicações distribuídas compostas por apenas um processo remotamente executado, e por aplicações que disparam execuções de várias pequenas tarefas[Casanova et al., 2002]. Em ambos casos as interações com usuário pode ser realizada com o redirecionamento da entradas e das saídas padrão e de erro, e para o caso de tarefas remotas é possível ainda interagir por meio da transmissão de uma seqüências de dados (streams).

Para o execução transparente dos trabalhos remotos, o sistema de execução remota utiliza alguns mecanismos de suporte como o sistema de carga dinâmica de códigos e bibliotecas e o sistema de redirecionamento de entrada e saída padrões. O sistema de carga dinâmica de código é responsável por enviar o bytecode de cada classe ao nó trabalhador assim que ela seja requisitada. Com esse mecanismo os nós trabalhadores não precisam possuir em seus discos locais as cópias dos programas que serão executados pelos clientes. O sistema de redirecionamento de entrada padrão e saídas padrão e erro permite a interação com o processo remoto da mesma maneira que é feita em programas locais de linha de comando.

O mecanismo de submissão utilizado permite ainda criar mecanismos despachadores de tarefas, os quais flexibilizam as possíveis formas de interação com o usuário, por exemplo, pode ser criado um nó que liga o procedimento de disparo de tarefas com um portal WWW, pelo qual usuários enviam arquivos JAR contendo todas as classes necessárias para a execução da aplicação, arquivos de entrada, e recebem as saídas no próprio navegador ou por e-mail. Outras possíveis opções podem ser implementadas, como um sistema que possua um nó que permite aplicações que utilizam a plataforma Java/RMI ou Java/JAMP[Trevelin and Ferreira, 1998] executarem procedimentos remotos nos nós participantes da grade, e em um sistema no qual são enviadas requisições de processamento através de clientes Java por linha de comando ou com interfaces gráficas (GUI).

3.6. Resultados e Conclusões

O principal objetivo desse projeto é criar um ambiente de grade computacional de fácil utilização, tanto para montar uma grade, quanto para submeter tarefas. Esse objetivo é alcançado com uma ferramente multi-agente que permite facilmente criar um ambiente de grade computacional com máquinas tendo apenas uma máquina virtual Java (JVM) instalada, além de submeter tarefas sem a necessidade de muitas modificações no programa

Tam.	Seq.	1nó	2nós	3nós	4nós	5nós	6nós	7nós	8nós
		9nós	10nós	11nós	12nós	13nós	14nós	15nós	16nós
128	12.6	12.28	6.75	4.95	4.26	3.81	3.61	**3.41**	3.51
		3.58	3.54	3.70	3.84	3.87	4.12	4.36	4.48
256	101.3	98.56	51.96	35.13	26.80	22.12	18.95	16.46	14.98
		13.75	12.98	12.21	11.56	**11.03**	12.62	14.03	13.65
512	827	798	417	283	215	178	146	126	112
		101	92.0	87.1	78.7	74.2	**68.7**	83.8	89.1

Table 1: Tempos de execuções para o ambiente P2PGrid com 16 nós

a ser executado.

Para montar uma nova grade de computadores, é necessário apenas iniciar um processo gerente do ambiente em uma máquina e iniciar o mesmo gerente nas demais, passando o endereço de rede de alguma outra máquina já pertencente à grade. O gerente do ambiente é então responsável por organizar corretamente as conexões e as mensagens trocadas entre os recursos para que se mantenha uma grade consistente. Dentre as mensagens trocadas, é possível destacar as responsáveis pelo controle de acoplamento, contagem de máquinas, controle de desconexão, carga dos recursos e pedidos para trabalhos de processos e tarefas remotas.

Para a execução remota de um processo completo, o programa Java não deverá ser modificado, bastando no momento da execução chamar um gerente de submissão, passando o nome da classe e seus parâmetros. Para a execução de tarefas (threads) remotas, o programador deverá implementar a interface RemoteRunnable, equivalente à Runnable, e instanciar a RemoteThread, equivalente a Thread, recompilar e executar normalmente o programa. Essas duas formas permitem a interação com o usuário por meio da entrada padrão e saídas padrão e de erro, como é feito normalmente pelos programas de *console*.

Foi utilizada, como avaliação do funcionamento do ambiente, uma aplicação de processamento de imagem fruto de um outro projeto[Faria et al., 2003], composta pela reconstrução tridimensional de *loops* coronais solares, que consiste em gerar imagens intermediárias de interpolação, tendo duas imagens de raio-X de satélite como entrada.

A tabela 1 mostra a execução dessa aplicação utilizando pares de imagens de três tamanhos diferentes. Quanto maior o tamanho da imagem, mais processamento é realizado para calcular cada ponto da imagem de saída, além de precisar gerar um número maior de imagens intermediárias. Por se tratar de geração tridimensional, quando se aumenta em duas vezes o tamanho da imagem, a complexidade da solução aumenta em oito vezes.

Para essa aplicação, utilizando duas imagens de tamanho 512 de largura por 512 de altura, são geradas 402 imagens de mesmo tamanho, interpolando as duas imagens iniciais. Nessa situação a aplicação foi dividida em tarefas remotas, as quais foram distribuídas entre os nós de um ambiente contendo 16 recursos. Em diversas execuções foi possível notar um melhor desempenho com 14 nós trabalhando, o qual apresentou um ganho (*speedup*) de 11,6 vezes àquele executado em apenas um recurso.

Outros testes mostraram que o ganho dessa implementação Java no ambiente desenvolvido foi superior a uma implementação equivalente em MPI rodando em um cluster com 9 nós do departamento, como pode ser visto na tabela 2.

A figura 6 apresenta os gráficos referentes aos tempos de execução em cada um dos ambientes para as imagens de raio-X de dimensões 128X128, 256X256 e 512X512.

Tam.	1nó 9nós	2nós 10nós	3nós 11nós	4nós 12nós	5nós 13nós	6nós 14nós	7nós 15nós	8nós 16nós
128	7.73 **1.11**	3.89 1.45	2.72 1.31	2.11 1.22	1.78 1.33	1.53 1.45	1.32 1.47	1.21 1.40
256	63.1 **9.46**	32.6 14.6	22.0 15.5	17.5 15.8	14.6 15.59	12.7 16.1	11.3 17.2	10.3 18.5
512	508.7 78.1	266.5 77.9	185.7 72.1	147.3 67.4	121.1 **64.2**	104.9 73.8	93.2 69.1	84.7 65.3

Table 2: Tempos de execuções para um cluster de 8 processadores rodando com MPI em vários segmentos (nós)

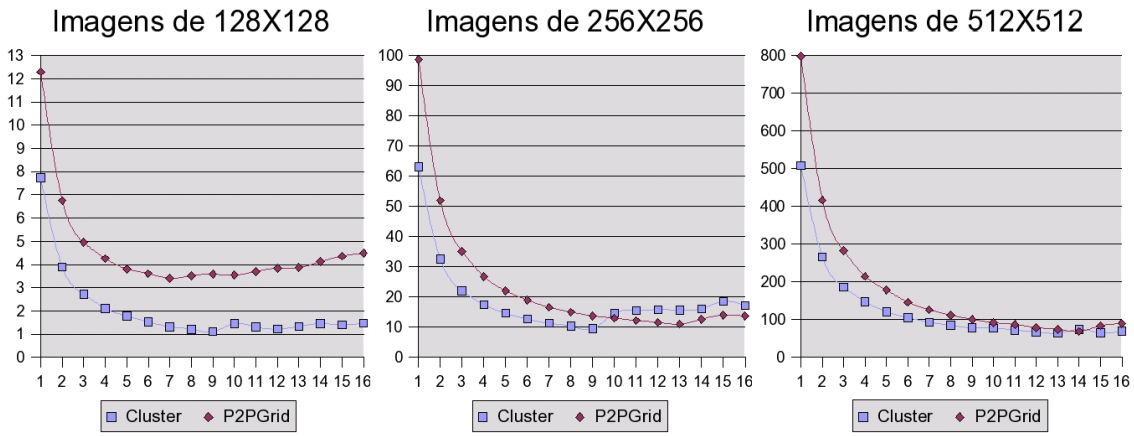


Figure 6: Gráficos dos tempos de execuções da aplicação paralela no cluster e no P2PGrid utilizando imagens de 128X128, 256X256 e 512X512

Esses gráficos mostram a desvantagem da implementação Java quando executada em apenas um nó, e a diminuição da vantagem quando se distribui em vários nós.

Ao fazer uma comparação direta do *speed-up* entre o ambiente P2PGrid e o desenvolvido em MPI rodando no cluster, apresentada na tabela 3, é possível notar que o cluster tem vantagem com a imagem menor de 128, mas conforme a imagem é aumentada, a solução desenvolvida para o P2PGrid aumenta o ganho computacional.

Os valores dos ganhos são também apresentados no gráfico da figura 7. Esse gráfico comprova a ineficiência de um ambiente de grade computacional para aplicações cuja quantidade de processamento é pequena em relação à quantidade de dados. Por outro lado, conforme essa relação se inverte, o ambiente de grade computacional permite ganhos competitivos e até melhores.

Os dados adquiridos até o momento comprovam o funcionamento do ambiente desenvolvido e de seu modelo de utilização, porém mais testes serão utilizados em comparação com outros ambientes semelhantes para que se prove não só o funcionamento do mesmo, mas de sua superioridade em situações equivalentes. A comparação da aplicação rodando nesse ambiente e no cluster, mostra que é possível obter ganhos na distribuição de tarefas sem a necessidade de alto custos com hardware e sem a necessidade de aprendizado de bibliotecas de programação paralela como a MPI. O ganho superior da aplicação rodando no ambiente de uma rede de *workstations* em comparação com a do cluster é dado pela diferença entre as programações utilizadas nas aplicações desenvolvidas com MPI e com Java, as quais possibilitam otimização, mas sendo estas mais complexas no primeiro caso.

Além da possibilidade de montar um ambiente de computação paralela e dis-

Situação	1nó	2nós	3nós	4nós	5nós	6nós	7nós	8nós
	9nós	10nós	11nós	12nós	13nós	14nós	15nós	16nós
Cluster 128	1.00	1.98	2.84	3.66	4.34	5.05	5.85	6.39
	6.96	5.33	5.90	6.33	5.81	5.33	5.52	5.26
Cluster 256	1.00	1.93	2.87	3.61	4.32	4.98	5.59	6.13
	6.67	4.32	4.06	3.99	4.05	3.91	3.40	3.67
Cluster 512	1.00	1.91	2.74	3.45	4.20	4.84	5.46	6.00
	6.51	6.53	7.05	7.54	**7.92**	6.89	7.79	7.36
P2PGrid 128	1.00	1.82	2.48	2.88	3.22	3.40	**3.61**	3.50
	3.43	3.46	3.32	3.20	3.17	2.98	2.81	2.74
P2PGrid 256	1.00	1.89	2.80	3.68	4.46	5.20	5.88	6.56
	7.17	7.59	8.07	8.53	**8.94**	7.80	7.02	7.22
P2PGrid 512	1.00	1.91	2.82	3.71	4.48	5.47	6.33	7.13
	7.90	8.67	9.16	10.14	10.75	**11.62**	9.52	8.96

Table 3: Ganhos obtidos com a paralelização no cluster e no P2PGrid com 16 computadores

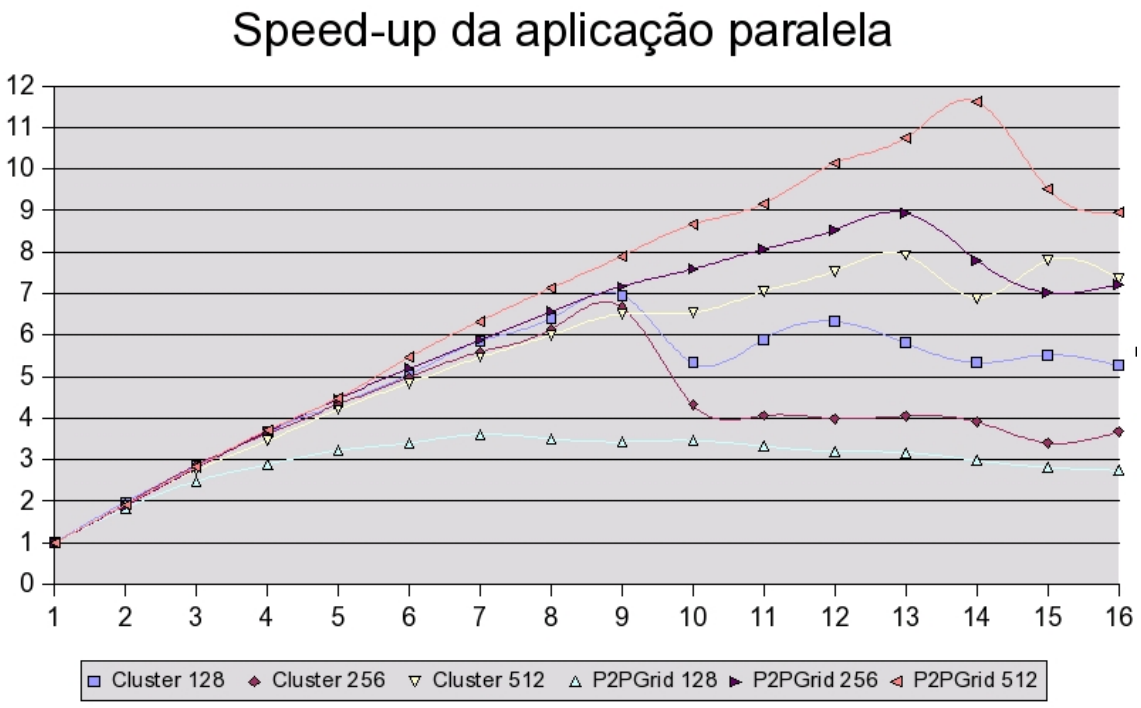


Figure 7: Gráfico dos ganhos (speed-up) da aplicação paralela no cluster e no P2PGrid utilizando imagens de 128X128, 256X256 e 512X512

tribuída de alto desempenho apenas com uso de computadores comuns sem muitas configurações, esse ambiente facilita ao usuário programador de aplicações paralelas, pois nÃo é necessário utilizar bibliotecas complexas para manipular a paralelização, basta utilizar classes semelhantes às utilizadas para desenvolver aplicações concorrentes em Java. Isso torna possível e acessível ambientes de alto desempenho a uma vasta quantidade de pessoas.

References

Baker, M. (2000). Cluster computing white paper.

Buyya, R., Abramson, D., and Giddy, J. (2000). Economy driven resource management architecture for computational power grids.

Casanova, H., Hayes, J., and Yang, Y. (2002). Algorithms and software to schedule and deploy independent tasks in grid environments. In *Grid 2002*.

Chetty, M. and Buyya, R. (2002). Weaving Computational Grids: How analogous are they with electrical grids? *Journal of Computing in Science and Engineering (CiSE)*, pages 61–71.

Faria, L. N., Mascarenhas, N., Morón, C., Rosa, R., and Sawant, H. (2003). Image morphing applied to 3d reconstruction of coronal loops. In *XVI Brazilian Symposium on Computer Graphics and Image Processing*, pages 207–213, São Carlos, Brasil.

Foster, I. Internet computing and the emerging grid.

Foster, I. and Iamnitchi, A. (2003). On death, taxes, and the convergence of peer-to-peer and grid computing. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Berkeley, CA, USA.

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1–24.

Mello, R. F., de Mattos, E. C. T., and Trevelin, L. C. (2003). Proposal of a tree load balancing algorithm to grid computing environments. In *2nd I2TS'2003 - International Information and Telecommunication Technologies Symposium*, Florianopolis, SC, Brasil.

Mello, R. F., de Mattos, E. C. T., Trevelin, L. C., de Paiva, M. S. V., and Yang, L. T. (2004). Proposal of a tree load balancing algorithm to grid computing environments. *IEICE Transactions on Information and Systems*, E87-D(7):1729–1736.

Rajkumar, M. B. (2002). Grids and grid technologies for wide-area distributed computing. *Journal of Software: Practice and Experience (SPE)*, 32(15):1437–1466.

Trevelin, L. C. and Ferreira, M. M. (1998). The java broker system: Concepts & java programming guide. Technical report, DC, UFSCar.