Uma Proposta de Escalonamento Colaborativo de Aplicações em um Ambiente de Computação em Grade

Lucas Alberto Souza Santos
Instituto de Informática, UFRGS
Porto Alegre, RS, Brasil
lassantos@inf.ufrgs.br

Marcelo Trindade Rebonatto
Instituto de Ciências Exatas e Geociências, UPF
Passo Fundo, RS, Brasil
rebonatto@upf.br

Patrícia Kayser Vargas
COPPE/Sistemas, UFRJ
Rio de Janeiro, RJ, Brasil
UniLaSalle, Canoas, RS, Brasil
kayser@cos.ufrj.br

Cláudio F. R. Geyer
Instituto de Informática, UFRGS
geyer@inf.ufrgs.br

Resumo

A Computação em Grade é cada vez mais utilizada para a execução de aplicações que demandam elevados custos computacionais. A tendência atual é o aumento da utilização das grades computacionais. Neste artigo propomos uma organização de grade descentralizada e um modelo de escalonamento de aplicações. No modelo, as aplicações submetidas à grade deverão estar descritas na forma de grafo orientado acíclico. Este modelo segue uma estrutura descentralizada baseada na tecnologia das redes Par-a-Par (P2P). Alguns modelos de rede P2P possuem características de tolerância a falhas, escalabilidade e dinamicidade. Com o progressivo aumento do compartilhamento de recursos geograficamente distribuídos, estas características se tornarão fundamentais para o sucesso da Computação em Grade. O modelo proposto está sendo integrado ao projeto ISAM na forma de grid broker. O sistema ISAM é um ambiente de Computação Pervasiva que engloba características da Computação em Grade. O modelo será avaliado por simulação com o uso da ferramenta MONARC 2, um simulador de computação distribuída baseado em eventos discretos.

1. Introdução

A Computação em Grade é cada vez mais utilizada para a execução de aplicações que demandam elevados custos computacionais. Atualmente, os pesquisadores da área de Bio-Informática [13] e Física de Altas Energias (*High Energy Physics - HEP*) [3] são grandes usuários da tecnologia de grades computacionais. Há uma grande demanda nestas áreas por desempenho computacional e compartilhamento de dados.

O aumento do número de centros de computação (também denominados de domínios administrativos ou centros regionais) aumentar, exigirá soluções de sistemas de grade e escalonamento de aplicações cada vez mais sofisticados. Estas soluções devem promover uma maior escalabilidade do sistema, mantendo bom desempenho das aplicações que executam, além de utilizar mecanismos de tolerância a falhas, e suportar políticas locais de controle do uso dos recursos. Os sistemas de computação em grade mais referenciados na literatura não atendem completamente às necessidades das futuras grades.

Este artigo propõe uma organização de grade descentralizada e um modelo de escalonamento de aplicações descentralizado para a arquitetura proposta. Na seção 2 serão apresentados os sistemas de grade mais referenciados na literatura. Na seção 3 apresentamos a motivação para uma nova arquitetura de grade. Na seção 4, o modelo proposto é explicitado e sua metodologia de avaliação apresentada. Na seção 5 são apresentados outros projetos que propõem soluções descentralizadas para grades computacionais. Na seção 6 estão a conclusão e trabalhos futuros.

2. Escalonamento em Grades computacionais

Nota-se nos últimos anos um crescente desenvolvimento de grades computacionais e de *middlewares* para a sua implementação. Muitas das propostas de gerenciamento de recursos e escalonamento de tarefas em grades formadas por domínios administrativos autônomos seguem uma organização hierárquica ou centralizada. Alguns exemplos de *mid-*

dlewares são Globus [10], Condor/Condor-G [22], Legion [12], OurGrid [1] e ISAM [25]. Os dois primeiros são os mais referenciados na literatura.

O **Projeto Globus** [10] provê o *Globus Toolkit* (GT) para grades, uma infraestrutura que permite que usuários possam compartilhar recursos computacionais, bancos de dados e outras ferramentas *online* de forma segura, cruzando os limites institucionais, corporativos e geográficos sem sacrificar a autonomia local. O projeto Globus é um esforço de pesquisa multi-institucional que busca permitir a construção de grades computacionais. O Globus (GT versão 3) oferece serviços de informação através de uma rede hierárquica chamada **Metacomputing Directory Services (MDS)**. O espaço de nomes do MDS é organizado hierarquicamente em forma de árvore.

O modelo de escalonamento do sistema Globus é descentralizado. O Globus não fornece suporte nativo a políticas de escalonamento, mas ele permite que resource *brokers* externos adicionem esta capacidade. Os serviços Globus são utilizados em uma variedade de sistemas de gerenciamento de recursos e escalonamento: Nimrod/G [5], NetSolve, GrADS [2] e AppLeS [6].

O **Condor** [22] é um sistema de gerenciamento de recursos, que tem o objetivo de prover uma plataforma computacional de alta-vazão, desenvolvido pela Universidade de Wisconsin em Madison nos EUA. O Condor realiza a descoberta de recursos ociosos em uma rede e a alocação destes recursos para execução de tarefas.

Um conjunto de estações de trabalho chamado de *Condor Pool* é gerenciado pelo sistema. Tarefas submetidas pelos usuários são colocadas em uma fila de espera, e escalonadas nas máquinas disponíveis de forma transparente ao usuário. O sistema Condor possui uma arquitetura de escalonamento centralizada. Uma máquina especial no sistema (Central Manager) é responsável pelo escalonamento. Cada estação de trabalho na rede Condor pode submeter tarefas ao Central Manager que é responsável por encontrar recursos disponíveis (ociosos) para a execução da tarefa.

O Condor possui um mecanismo chamado *flocking* [7] que permite o compartilhamento de recursos e migração de tarefas entre múltiplas *Condor Pools*.

3. Motivação

Um escalonador para grade depende diretamente do sistema de informação, módulo que possibilita a gerência, descoberta e monitoração de recursos. O Toolkit Globus e o sistema Condor, embora amplamente utilizados nas grades atuais, não apresentam a **totalidade** das seguintes características fundamentais para suportar o crescente uso da computação em grade:

- elevada escalabilidade;
- tolerância a falhas e ataques;
- auto-adaptação e dinamicidade;
- completa autonomia dos centros regionais sobre seus recursos.

O modelo estruturado em árvore do sistema MDS do Globus é escalável e possui boa eficiência, todavia os nós mais altos da hierarquia são pontos críticos de falha, tornando o sistema suscetível a falhas e ataques. Em caso de *crash* de uma entidade superior da hierarquia do MDS, os recursos indexados pela entidade afetada serão desconectados da rede principal. No sistema Globus, a re-integração dos recursos isolados à rede principal e a re-estruturação da hierarquia exige a intervenção manual dos gerentes responsáveis pelos recursos afetados.

A necessidade de intervenção humana após a falha, deve-se a incapacidade de re-estruturação autônoma da rede de informação dos recursos. O particionamento do sistema de informação diminui drasticamente o desempenho dos sistemas de descoberta de recursos e escalonamento da grade.

O mecanismo de *flocking* do sistema Condor é complexo para configurar e dar manutenção. Esta funcionalidade necessita de pré-configuração manual através de arquivos. Isto deve-se a característica estática do serviço de *flocking* do Condor, que não suporta recursos dinâmicos e necessita de conhecimento prévio de todas as *Condor Pools* remotas. Uma proposta de melhoria desta funcionalidade está analisada em [4], mas não ainda foi integrada ao sistema Condor.

O uso colaborativo de recursos computacionais distribuídos por centros de pesquisa, empresas, organizações governamentais e usuários domésticos, torna fundamental que sistemas de computação em grade sejam escaláveis e tolerantes a falhas, capazes de re-estruturação e auto-adaptação, necessitando o mínimo de manutenção humana.

A organização das redes P2P não-estruturadas mais conhecidas como o Kazaa [15], Gnutella [19] e os protocolos abertos do JXTA [11] permitem a re-estruturação da rede em caso de falhas, elevada escalabilidade e adaptação a mudanças no ambiente. Devido a essas características, existem autores que vêm considerando a possibilidade de aplicar técnicas de P2P em ambientes de grade [1, 8, 16, 21].

As alternativas propostas neste trabalho são uma arquitetura de grade computacional e um modelo de escalonamento de aplicações organizados de forma descentralizada. Nesta arquitetura, domínios administrativos autônomos compartilham recursos com outros domínios vizinhos, criando um sistema de compartilhamento colaborativo, não-hierárquico, dinâmico, organizado em rede **Par-a-Par** *(Peer-to-Peer)* **P2P**.

Esta proposta será integrada ao ambiente em grade do projeto ISAM. O sistema **ISAM (Infra-estrutura de Suporte às Aplicações Móveis Distribuídas)** [24], em de-

senvolvimento no Instituto de Informática da UFRGS, é um *middleware* direcionado ao gerenciamento de recursos em redes heterogêneas, suportando a execução de aplicações distribuídas baseadas em componentes. A arquitetura do ambiente ISAM é organizada na forma de células autônomas cooperativas, esta rede formada pelas células possui características de uma rede P2P não-estruturada. Cada célula possui um escalonador local e políticas próprias de uso de seus recursos. Desta forma, um modelo de escalonamento de tarefas descentralizado para a plataforma ISAM é fundamental para que o sistema alcance níveis elevados de escalabilidade, balanceamento de carga e tolerância a falhas.

4. Modelo Proposto

O modelo de grade proposto neste trabalho objetiva criar uma comunidade de centros regionais de computação como pares de uma rede lógica (*overlay network*) P2P . Os centros (domínios) que participam da rede P2P procuram compartilhar recursos computacionais, ou seja, utilizam recursos remotos e fornecem recursos computacionais para usuários de outros domínios. O sistema possui suporte para que os donos dos recursos possam estabelecer políticas de controle de acesso a seus recursos, assim eles têm controle total sobre sua participação na grade computacional. Cada domínio administrativo da grade P2P é formado por uma célula ISAM, que é composta de uma máquina base e outros nodos computacionais. A máquina base é a entidade responsável pelo gerenciamento de todos os recursos da célula, onde os serviços principais executam, e onde executará o serviço que manterá a rede de escalonamento P2P funcionando.

Desta forma, a arquitetura da rede P2P formada pelas células ISAM (Figura 1) se aproxima do modelo **Super-Peer** [26] de rede P2P. Este modelo de rede, é considerado um modelo híbrido de rede par-a-par, pois existem nodos da rede que se comportam como entidades centrais (base da célula) para alguns outros nodos (recursos da célula). Para facilitar a compreensão do modelo, podemos abstrair o modelo super-peer, encapsulando todos os recursos de uma célula ISAM, em uma única entidade nodo da rede P2P, sem perda informações.

Os usuários irão submeter aplicações organizadas em forma de DAG (*Directed Acyclic Graph*) a um dos pares da rede e o serviço de escalonamento P2P servirá de *Grid Broker* para comunicação com outros *grid brokers* de células vizinhas. O DAG da aplicação possui tarefas como nodos e dependências como arestas. Esse modelo de aplicações utilizado permite a expressão de aplicações *bag-of-task* através de um grafo com conjunto de arestas vazio. Os recursos de um nodo da rede (célula ISAM) poderão ser acessados por usuários locais da célula, ou por usuários de outras células, através da comunicação entre *Grid Brokers*. O serviço de *Grid Broker* (Figura 2) é composto por um escalonador de grade com uma fila de tarefas globais, um escalonador local com uma fila de tarefas locais, um repositório de políticas e uma lista de nodos vizinhos.

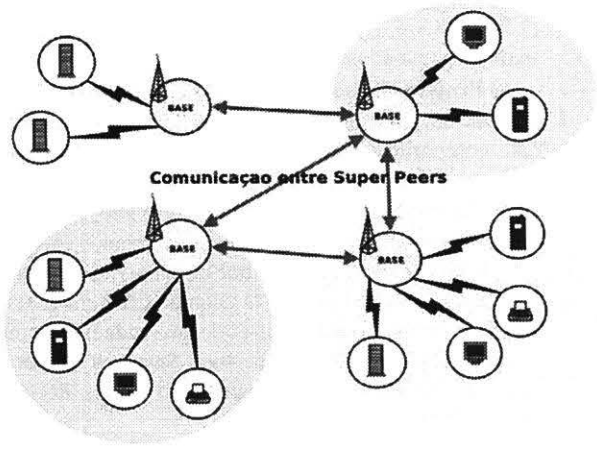

Figura 1. Arquitetura da rede super-peer formada pelas células ISAM

No modelo, cada centro regional da grade P2P possui um único *Grid Broker*. Este Grid Broker é responsável por escalonar as aplicações submetidas pelo usuário locais do centro e também as aplicações que tiveram origem em outro ponto da grade.

Os componentes do serviço de broker possuem as seguintes funções:

Escalonador da Grade Sua função é escalonar aplicações descritas na linguagem Grid-ADL (linguagem de descrição de aplicações para ambiente de grade) [23] nos recursos da grade. Uma aplicação DAG é formada por um conjunto de tarefas que pode possuir dependências de arquivos entre si. O usuário expressa em GRID-ADL apenas o conjunto de tarefas, e as dependências e o DAG são inferidos pelo sistema. Este componente, após inferir o DAG, particiona o grafo da aplicação através de um algoritmo de particionamento (*clustering*) e cada sub-grafo gerado é colocado na fila de tarefas globais.

Este componente utiliza protocolos P2P para manter a rede em funcionamento e gerenciar a lista de células vizinhas na rede lógica par-a-par. É responsável por verificar o estado das tarefas que compõem as aplicações, e tomar as decisões adequadas para que as aplicações terminem com sucesso, mesmo em caso de falhas.

Para escalonar, este componente analisa o estado

dos recursos da célula (consultando o escalonador local) e as *políticas de escalonamento*. Através de um *algoritmo de escalonamento global*, decide quais sub-grafos (conjunto de tarefas) da fila de tarefas globais podem executar nos recursos de sua célula local, e quais sub-grafos serão enviados à outros centros administrativos da rede lógica da grade P2P.

- Os sub-grafos escalonados para execução na célula local são transferidos para a fila de tarefas locais.
- Os sub-grafos escalonados para execução distribuída são enviados ao *Grid Broker* do centro administrativo vizinho, que é escolhido pelo *algoritmo de escalonamento global*. A heurística deste algoritmo de escalonamento global envolve uma busca por recursos ociosos na rede lógica da grade P2P.

Fila Global Nesta fila estão os conjuntos de tarefas globais da grade, estes conjuntos de tarefas são sub-grafos que fazem parte de aplicações (grafos DAG) disparadas em algum ponto da grade. O conjunto de tarefas (sub-grafo) pode migrar de um centro administrativo a outro, de célula em célula, até que seja colocado em uma fila de escalonamento local.

Escalonador Local Sua função é escalonar as tarefas da fila local nos recursos disponíveis da célula. Este componente realiza a monitoração do estado dos recursos e a contabiliza a utilização destes, estas informações serão utilizadas pelo *algoritmo de escalonamento global* do *Escalonador de Grade*.

Este escalonador utiliza um *algoritmo de escalonamento local* que seleciona as tarefas a serem executadas em um dado momento, e decide a quais recursos serão destinadas as tarefas. As decisões deste escalonamento são diretamente influenciadas pelas *políticas de escalonamento* definidas pelos gerentes dos recursos da célula. O usuário com privilégio de acesso local à célula tem a opção de disparar uma aplicação Grid-ADL diretamente neste escalonador local, então as tarefas do grafo desta aplicação DAG passarão diretamente à fila de tarefas locais.

Ao receber um sub-grafo do *Escalonador da Grade*, este componente desfaz o sub-grafo e colocas as tarefas que o compõem na fila local.

Fila Local Nesta fila estão as tarefas que serão executadas na célula local. Estas tarefas fazem parte de aplicações DAG disparadas inicialmente em algum ponto da grade, seja ele local ou remoto. Este tipo de tarefa não pode migrar a outro centro administrativo, será executada nos recursos da célula local.

Políticas de Escalonamento As políticas de escalonamento são propriedades que controlam quais recursos são escalonados dentre os recursos que atendem as requisições das tarefas. Estas propriedades definem permissões de **quem** pode acessar, **quando** acessar e **como** podem ser acessados os recursos conectados ao domínio administrativo.

As políticas de escalonamento são divididas em dois grupos:

- *Políticas Globais* são definidas pelo gerente do domínio administrativo, estas políticas são respectivas ao conjunto formado por todos os recursos da célula.
- *Políticas Locais* estão relacionadas a recursos individuais da célula, como um computador, banco de dados ou impressora. Estas últimas são definidas pelo dono do recurso.

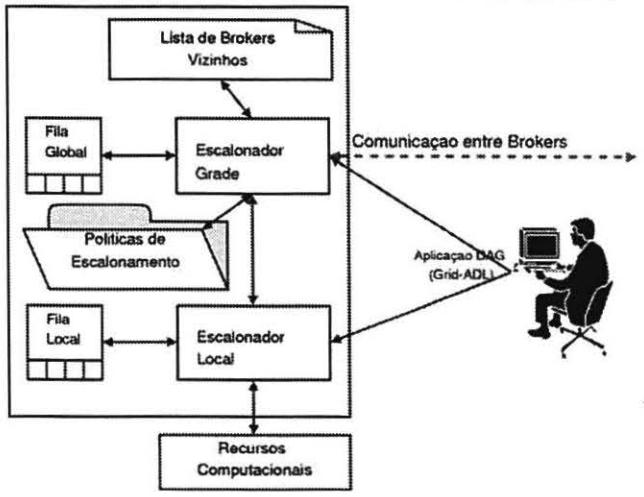

Figura 2. Arquitetura do componente Grid Broker

4.1. O Simulador MONARC 2

A validação do modelo de escalonamento proposto neste trabalho será feita por simulação usando a ferramenta MONARC 2 [18].

O simulador MONARC foi desenvolvido como parte do projeto "The MONARC Project - Models of Networked Analysis at Regional Centres for LHC Experiments"[17]. Este ambiente de simulação para computação distribuída, é constituído de uma ferramenta de simulação orientada

a processos, baseada em eventos discretos concorrentes usando a tecnologia Java. O objetivo era fornecer uma ferramenta que provesse simulações o mais reais possível, de sistemas de computação distribuída, otimizada para o processamento de dados da área de Física de Altas Energias (HEP) e flexível para fornecer avaliação de desempenho de uma variedade de arquiteturas de processamento.

Os principais componentes da segunda versão do *framework* (MONARC 2) são centros regionais (cpus, clusters), redes (LAN, WAN), bancos de dados, tarefas (*jobs*) e um escalonador de tarefas (local, distribuído). A ferramenta MONARC 2 também pode simular falhas dos componentes do centro regional, possibilitando assim sua utilização como ferramenta de avaliação de técnicas de tolerância a falhas em sistemas distribuídos.

Para a validação das propostas de arquitetura de grade e escalonamento deste trabalho, o *framework* do MONARC 2 foi modificado. A característica de herança permitiu que as classes mais importantes do sistema fossem estendidas sem que seus códigos originais fossem modificados, facilitando assim o processo de implementação do compartilhamento de recursos em rede P2P.

A simulação é bastante útil pois permitirá a definição da melhor forma de organização da rede lógica P2P, já que esta tem impacto direto no tempo total de execução das aplicações DAG e na quantidade de recursos consumidos na transferência de arquivos.

4.2. Simulando Escalonamento P2P

As extensões criadas neste trabalho para o simulador MONARC 2 permitem que o modelo de escalonamento apresentado neste artigo seja avaliado por simulação. Abaixo descrevemos um experimento realizado para averiguar a arquitetura de grade P2P criada através do compartilhamento de recursos entre centros administrativos.

O cenário simulado apresenta uma grade computacional formada por 25 centros de computação. Estes centros computacionais compartilham recursos entre si através de uma rede lógica par-a-par. Cada centro regional possui um conjunto de centros vizinhos aos quais pode enviar ou receber tarefas para execução.

A grade deste teste foi construída por um algoritmo gerador de topologia de rede par-a-par. Através do ajuste dos parâmetros deste algoritmo é possível modelar uma variedade de arquiteturas de grades computacionais em rede P2P. Este algoritmo possui como parâmetros:

1. Os tipos de centros regionais (***T***) (descrição dos recursos computacionais e redes LAN/WAN);
2. Número (***N***) de centros administrativos (células ISAM) de cada tipo;
3. Número (***c***) mínimo de conexões entre centros regionais;
4. Número (***C***) de conexões médias entre centros regionais;
5. Desvio padrão (***dC***) das conexões entre centros regionais;

No experimento são no total 5 centros regionais do *Tipo 1* e 20 centros regionais do *Tipo 2*:

- Os centros do *Tipo 1* possuem uma arquitetura bastante heterogênea: 2 computadores Pentium 4 (2 GHz, 512 MB), 2 computadores Pentium 3 (600 MHz, 256 MB), 1 computador AMD (1.8 GHz, 512 MB) e 1 computador Sun Sparc Ultra 3 (128 MB). A rede local LAN é de 100 Mb/s e a rede WAN possui 10 Mb/s.
- Os centros do *Tipo 2* possuem 6 computadores homogêneos Pentium 4 (1.8 GHz, 512 MB). A rede local LAN é de 100 Mb/s e a rede WAN possui 10 Mb/s.

Neste experimento, o número mínimo de conexões entre os centros regionais é $c = 1$, o número de conexões médias é $C = 4$ e o desvio padrão das conexões é igual a $dC = 1.0$.

Os centros do *Tipo 1* submetem à grade uma mesma aplicação 10 vezes seguidas. Esta aplicação é do *tipo fase*, muito comum em de programação paralela. Neste tipo de aplicação, a fase inicial é composta por várias tarefas que processam dados. Logo após este processamento, a fase final é responsável por processar os resultados da fase anterior e gerar os arquivos de saída da aplicação. Na simulação foi usado uma aplicação DAG composta de 6 tarefas (Figura 3): 5 tarefas na fase inicial e 1 tarefa na fase final. As tarefas iniciais geram arquivos de saída de 10 MB, estes arquivos são entradas da tarefa final coletora.

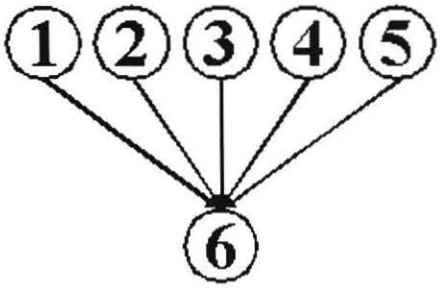

Figura 3. Aplicação DAG do tipo fase utilizada na simulação

O algoritmo de escalonamento deste experimento é simples, com o objetivo de simplificar o teste:

1. Na submissão de aplicações DAG em um centro regional, o grafo DAG é particionado aleatóriamente em N partes. Os sub-grafos que compõem o DAG são colocadas na *Fila de Escalonamento Global*.
2. No escalonamento de aplicações, o centro regional só pode mover as tarefas da *Fila de Escalonamento Global* à sua *Fila de Escalonamento Local* quando sua carga de média de CPU esteja abaixo do limiar 70%.
3. Quando a carga de CPU estiver acima deste limiar, as tarefas são enviadas ao centro administrativo vizinho que esteja menos carregado. Desta forma as tarefas vão percorrer a rede P2P até que sejam colocadas em uma *Fila de Escalonamento Local* de um centro da grade.
4. Para evitar loops, as tarefas possuem um valor de TTL (tempo de vida) igual a 5, ou seja, só poderão migrar no máximo 5 vezes. Quando este valor de TTL é atingido, as tarefas são colocadas na *Fila de Escalonamento Local* do centro regional corrente, mesmo que a carga média de CPU do centro esteja acima do limite.

Os resultados seguintes são as medidas de Carga média de CPU/Memória de alguns centros administrativos dos *Tipo 1 e 2* (Figuras 4 e 5).

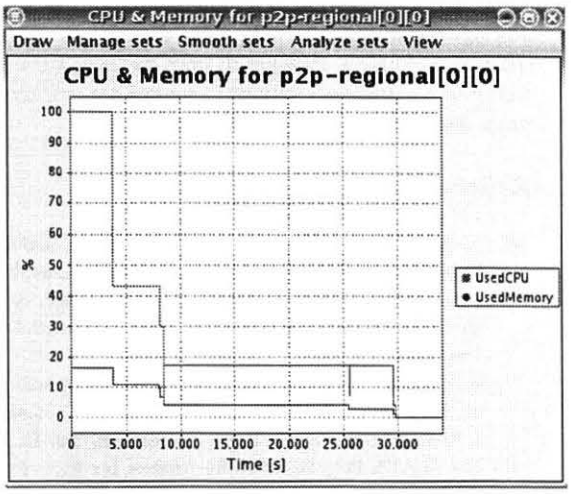

Figura 4. Carga de CPU e Memória de um centro administrativo do *Tipo 1*

Os resultados abaixo são as medidas da execução em alguns centros administrativos dos *Tipo 1 e 2* (Figuras 6 e 7).

Notar nos gráficos do resultado do experimento que os centros do *Tipo 1* sofrem uma grande utilização de CPU e memória, em virtude da estratégia de ocupação máxima de sua capacidade de processamento. Quando a média de utilização de CPU atinge o limite de 70%, os centros do *Tipo*

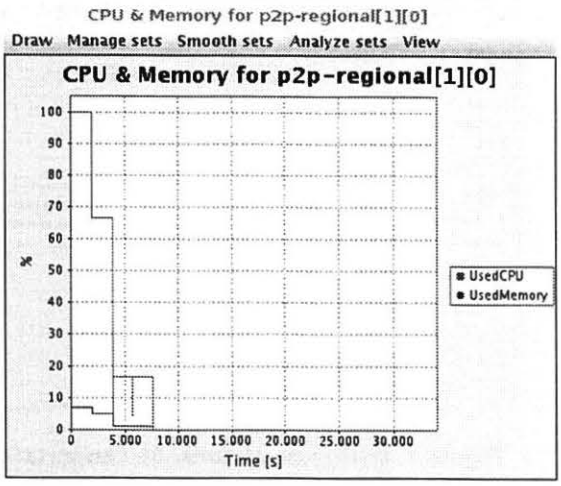

Figura 5. Carga de CPU e Memória de um centro administrativo do *Tipo 2*

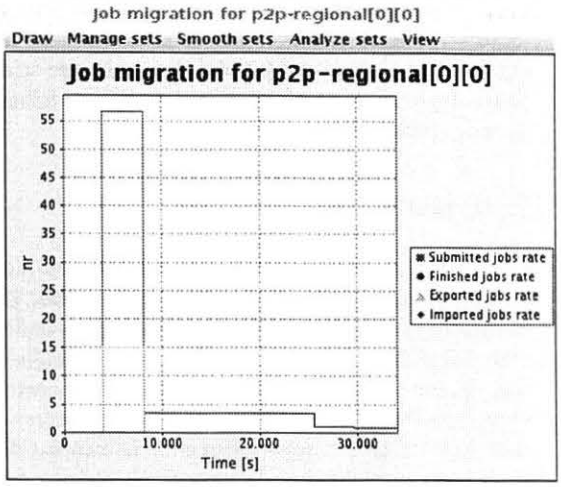

Figura 6. Dados da submissão / importação / exportação de tarefas de um centro administrativo do *Tipo 1*

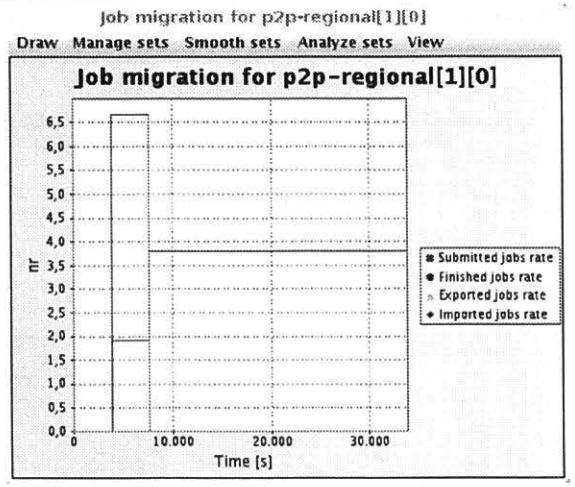

Figura 7. Dados da submissão / importação / exportação de tarefas de um centro administrativo do *Tipo 2*

1 iniciam a migração de sub-grafos de sua *Fila Global* para os escalonadores de outros centros ociosos da grade.

Os centros do *Tipo 2* que estavam ociosos, importam os sub-grafos de aplicações submetidas em outros centros da grade.

Estes resultados mostram que a simulação de uma organização de grade P2P é viável e abre perspectivas para estudos posteriores de algoritmos de escalonamento nesta arquitetura. O simulador MONARC permite que sejam avaliados aspectos de escalabilidade, tolerância a falhas, carga da rede LAN/WAN e carga de recursos.

5. Trabalhos Relacionados

Existem algumas propostas de modelo para escalonamento descentralizado não-hierárquico em grades. Em [20], os autores afirmam que uma arquitetura descentralizada em P2P possibilitou melhor escalabilidade e tolerância a falhas, quando comparada com uma arquitetura centralizada. O projeto **OurGrid** [1] utiliza uma rede de favores estruturada em P2P para compartilhamento de recursos de forma justa entre participantes de uma grade, todavia o modelo de aplicações do sistema OurGrid é limitado ao tipo *bag-of-task*. As aplicações do tipo *bag-of-task* representam uma pequena parcela dos tipos de aplicações para grade. O OurGrid possui um protótipo implementado com a tecnologia JXTA [11]. O sistema **Triana Grid** [21] utiliza um modelo de grade descentralizado em P2P. O Triana Grid utiliza as tecnologia JXTA e OGSA [9]. O sistema **LEAF** (*Learning Agent based FIPA toolkit*) [16] é um sistema de grade descentralizado que utiliza explicitamente o modelo de agentes como pares de uma rede P2P.

6. Conclusão e Trabalhos Futuros

O modelo atende os requisitos de um escalonador para grades. O *Grid Broker* é a entidade responsável por manter a rede P2P, executar as políticas definidas pelos donos dos recursos e escalonar as aplicações de forma distribuída na grade. A proposta está sendo avaliada através de simulação com o simulador MONARC 2 [18] [14].

O modelo de aplicação utilizado no *framework* será baseado no *framework* GRAND [23], que define a Grid-ADL, uma linguagem de descrição de aplicações do tipo DAG, constituídas por tarefas com ou sem dependências de arquivo entre si. O protocolo P2P utilizado no *framework* JXTA [11] está sendo analisado e possivelmente será adotada uma extensão deste protocolo para atender os requisitos do modelo proposto.

Os algoritmos de escalonamento de aplicações DAG e de mapeamento de tarefas nos recursos locais, respectivos aos módulos *Escalonador Grade* e *Escalonador Local* do modelo, estão sendo pesquisados. O estudo destes algoritmos renderá trabalhos futuros.

A validação do modelo será feita através da análise do desempenho de algumas aplicações descritas em DAG, escolhidas de forma a abranger os tipos de carga (leve, baixa, alta) e (computacional, armazenamento) a que estão sujeitos os ambientes de computação em grade.

Referências

[1] N. Andrade, W. Cirne, F. V. BRASILEIRO, and P. ROISENBERG. OurGrid: An approach to easily assemble grids with equitable resource sharing. In *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, June 2003.

[2] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski. The GrADS Project: Software support for high-level Grid application development. *The International Journal of High Performance Computing Applications*, 15(4):327–344, 2001.

[3] J. J. Bunn and H. B. Newman. Data intensive grids for high energy physics. In F. BERMAN, G. FOX, and T. HEY, editors, *Grid Computing: Making the Global Infrastructure a Reality*, pages 859–906. John Wiley & Sons, 2003.

[4] A. R. Butt, R. Zhang, and Y. C. Hu. A self-organizing flock of condors. In *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, page 42, Washington, DC, USA, 2003. IEEE Computer Society.

[5] R. Buyya, D. Abramson, and J. Giddy. Nimrod/G: An architecture of a resource management and scheduling system in a global computational grid. In *HPC Asia 2000*, pages 283–289, Beijing, China, May 2000.

[6] H. Casanova, G. Obertelli, F. Berman, and R. Wolski. The apples parameter sweep template: User-level middleware for the grid. In *Proceedings of the Supercomputing 2000*, 2000.

[7] D. H. J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A worldwide flock of condors: load sharing among workstation clusters. *Future Gener. Comput. Syst.*, 12(1):53–65, 1996.

[8] I. Foster and A. Iamnitch. 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, February 2003.

[9] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers/ogsa.pdf, January 2002.

[10] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[11] L. Gong. Jxta: A network programming environment. *IEEE Internet Computing*, 5(3):88–95, 2001.

[12] A. S. Grimshaw, A. Ferrari, F. Knabe, and M. Humphrey. Wide-area computing: Resource sharing on a large scale. *IEEE Computer*, 32(5):29–36, May 1999.

[13] K.-i. Kurata, C. Saguez, G. Dine, H. Nakamura, and V. Breton. Evaluation of unique sequences on the European data grid. In *Proceedings of the First Asia-Pacific conference on Bioinformatics 2003*, pages 43–52. Australian Computer Society, Inc., 2003.

[14] I. Legrand and H. B. Newman. The MONARC toolset for simulating large network-distributed processing systems. In *Winter Simulation Conference 2000*, pages 1794–1801, 2000.

[15] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the KaZaA network. In *3rd IEEE Workshop on Internet Applications (WIAPP'03)*, San Jose, CA, USA, June 2003.

[16] S. Lynden and O. F. Rana. Coordinated learning to support resource management in computational grids. In *P2P '02: Proceedings of the Second International Conference on Peer-to-Peer Computing*, page 81, Washington, DC, USA, 2002. IEEE Computer Society.

[17] Models of networked analysis at regional centres for large hadron collider experiments, 1999. `http://monarc.web.cern.ch/MONARC/`.

[18] Models of networked analysis at regional centres for lhc experiments, 2005. `http://monarc.cacr.caltech.edu:8081/`.

[19] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1):50–57, January/February 2002.

[20] H. Shan, L. Oliker, and R. Biswas. Job superscheduler architecture and performance in computational grid environments. In *Proceedings of the Supercomputing 2003*, 2003.

[21] I. Taylor, M. Shields, I. Wang, and R. Philp. Distributed p2p computing within triana: A galaxy visualization test case. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 16.1, Washington, DC, USA, 2003. IEEE Computer Society.

[22] D. Thain, T. Tannenbaum, and M. Livny. Condor and the Grid. In F. BERMAN, G. FOX, and T. HEY, editors, *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley, 2003.

[23] P. K. Vargas, I. d. C. Dutra, and C. F. Geyer. Gerenciamento hierárquico de aplicações em ambientes de computação em grade. In *Escola Regional de Alto Desempenho (ERAD 2004)*, Pelotas, RS, 13 a 17 de janeiro 2004.

[24] A. Yamin, I. Augustin, J. Barbosa, and C. F. Geyer. ISAM: a pervasive view in distributed mobile computing. In *Proceedings of the IFIP TC6 / WG6.2 & WG6.7 Conference on Network Control and Engineering for QoS, Security and Mobility (NET-CON 2002)*, pages 431–436, October 23–25 2002.

[25] A. Yamin, I. Augustin, J. Barbosa, L. C. d. Silva, R. A. Real, G. Cavalheiro, and C. F. Geyer. Towards merging context-aware, mobile and grid computing. *International Journal of High Performance Computing Applications*, 17(2):191–203, June 2003.

[26] B. Yang and H. Garcia-Molina. Designing a super-peer network. Technical Report 2002-13, Stanford University, 2002.