Performance Data Interface: Obtenção de Informações de Performance Estendendo a Implementação OGSI para Redes Grid

Matsui, Aurélio Akira M.
PCS - POLI - Universidade de São Paulo
aurelio.matsui@poli.usp.br

Sato, Líria Matsumoto
PCS - POLI - Universidade de São Paulo
liria.sato@poli.usp.br

Abstract

A escalabilidade de aplicações baseadas em serviços grid depende da capacidade de se prever a carga dos servidores. A Performance Data Interface, introduzida nesse artigo, foi proposta para prover informações para analisar condições anteriores de performance, enquanto especifica uma linguagem comum para troca de informações de performance entre elementos de um sistema grid. Além disso, o conceito de predição de performance em redes grid é demonstrado usando testes de execução paralela de tarefas.

1. Introdução

Sistemas baseados em computação grid [5] têm surgido nos últimos anos para interligar computadores geograficamente espalhados pelo mundo em organizações virtuais. Em cada instituição participante do grid, uma infra-estrutura de serviços grid encontra-se disponível para compartilhar recursos como processadores, bancos de dados ou unidades de armazenamento de arquivos. A função dessa infra-estrutura de serviços é tornar transparentes os recursos computacionais disponíveis, possibilitando o surgimento de aplicações distribuídas de larga escala como em [2].

Um dos benefícios provenientes da criação de um grid está em ser possível a colaboração mútua entre seus membros para somar as suas capacidades de computação. Ou seja, tarefas de computação demorada podem ser processadas mais rapidamente, através da sua divisão em sub-tarefas menores executadas em mais de um servidor do grid ao mesmo tempo.

Para que isso aconteça, a distribuição de tarefas deve ser feita com base no potencial de cada computador. Assim, a distribuição das tarefas requer que todo responsável pela distribuição de carga presente no grid seja capaz de conhecer a performance dos servidores que processam tarefas.

Esse trabalho não é trivial numa rede grid quando se leva em conta que os servidores nela presentes podem se encontrar em domínios administrativos distintos. Uma aplicação que pretende usar um servidor grid localizado em outro domínio administrativo deve se adaptar aos recursos e informações que o servidor oferece. Não é sempre possível alterar um servidor grid em outro domínio administrativo para oferecer às aplicações clientes toda informação de que elas precisam.

Além disso, é muito grande a quantidade de variáveis que influenciam na capacidade de execução de tarefas de um servidor grid. Até as condições de refrigeração e instalação elétrica, são fatores que podem influenciar na disponibilidade de um serviço.

É, então, necessária a existência de uma forma padronizada de divulgação dos parâmetros de performance de servidores grid.

Nesse artigo, apresenta-se a PDI (Performance Data Interface), que propõe uma solução para essa lacuna. A PDI, criada pelos autores deste artigo, torna possível a coleta de informação de performance de servidores grid de forma histórica, permitindo a criação de componentes para o balanceamento de carga especialmente desenhados para redes grid, como em [10].

O restante desse artigo está organizado da seguinte forma: as duas primeiras seções a seguir apresentam o contexto no qual a PDI está situada. A seção 2 introduz o conceito de sistemas grid, enquanto que a seção 3 discute o conceito de interface em serviços grid. A interface de performance PDI é apresentada em detalhes na seção 4. Cada grupo de problemas que ela resolve é apresentado em uma subseção.

Na seção 5, Antes da conclusão, são apresentados testes que mostram a PDI atuando para otimizar a utilização de recursos num grid.

2. Serviços Grid

Redes grid têm sua história iniciada a partir da criação da Internet. Logo após a interligação de redes locais de computadores espalhadas pelo mundo em uma rede global, pesquisadores vislumbraram nessa infra-estrutura uma possibilidade de distribuição da computação sem precedentes.

Agora, ao invés de recorrer apenas aos recursos presentes em redes locais, a computação distribuída poderia fazer uso do poder computacional localizado a grandes distâncias.

Para isso, foi criado o conceito de "rede grid", significando uma infra-estrutura de computação distribuída global, no qual os elementos que compõem o grid podem estar muito afastados e agrupados em domínios administrativos distintos.

Mesmo utilizando outros recursos computacionais para o processamento de suas tarefas, um elemento num grid é incapaz de conhecer ou controlar totalmente os elementos dos quais ele depende.

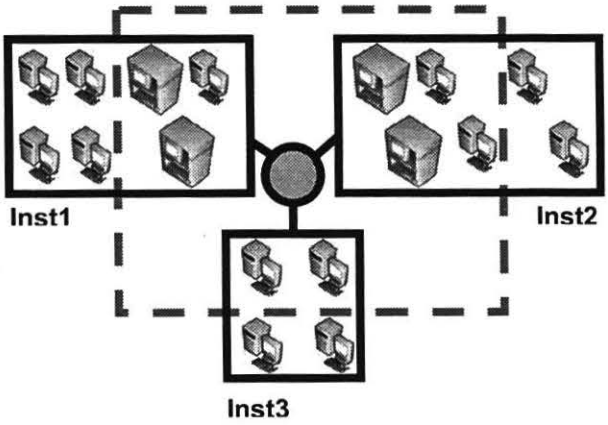

Figura 1 – Estrutura de uma organização virtual

Os elementos de um grid compõem uma organização virtual, conforme mostrado pela linha tracejada na Figura 1 e definido em [4]. Na figura, a organização virtual é composta de quatro servidores e cinco estações de trabalho. Os três retângulos em linhas sólidas são três instituições que encerram três redes locais. Cada rede local é entendida como um domínio administrativo distinto. Assim, por exemplo, as duas estações de trabalho da Inst3, apesar de terem acesso aos quatro servidores, não são administradas pela mesma equipe ou segundo a mesma política que os servidores que provêem os serviços grid.

Os participantes de uma organização virtual contam com mecanismos de autenticação e métodos de acesso unificados. Ao criar uma organização virtual, as instituições componentes estabelecem os padrões que todos os participantes da organização seguirão.

Os serviços grid, como especificados na OGSA (Open Grid Services Architecture) foram criados como uma extensão dos webservices. Esses serviços se apresentam como uma extensão dos serviços de requisições a grandes distâncias, adicionando mecanismos de meta dados, notificações e reservas como em [3]. A implementação mais importante de aplicação servidora para serviços grid é o Globus Toolkit, criado pela Globus Alliance [9].

Para definir "serviço grid", a OGSA contém uma especificação chamada OGSI (Open Grid Services Infrastructure) [8]. Esta é um conjunto de pré-requisitos que um webservice deve respeitar para ser considerado um serviço grid. A OGSI representa, então, um contrato bilateral entre clientes e servidores grid.

3. Interfaces e Serviços Grid

Além das assinaturas dos métodos que devem ser implementados, a especificação OGSI apresenta uma documentação a respeito de como eles devem se comportar internamente.

Então, a OGSI pode ser entendida como um classificador de webservices. Aqueles webservices classificáveis sob a especificação OGSI são os "serviços grid". Mas, uma vez que os próprios serviços podem ser também estendidos, os serviços grid também podem ser vistos como classes e suas instâncias nos servidores seriam os seus objetos.

Uma interface de serviço grid seria, consequentemente, um classificador de serviços grid. O programador que desenvolve um serviço grid pode usar uma interface grid para garantir a aderência do serviço grid à uma especificação da qual a interface faz parte. Praticamente, isso significa que um programador, ao implementar uma interface de serviço grid, é obrigado a programar uma implementação para cada operação declarada na interface.

Uma vez que uma interface é publicada junto à sua descrição, uma linguagem comum é criada entre clientes e servidores. Isso significa que um cliente que contacta um determinado serviço grid o qual implementa uma determinada interface, deve ser construído assumindo que os serviços grid seguem a descrição textual provida junto à interface.

De forma inversa, um servidor deve assumir que requisições a serviços sejam compatíveis com a mesma descrição textual embora seja fortemente recomendável que serviços sejam tolerantes a requisições inconsistentes.

4. Interface de Performance

A PDI foi desenvolvida fazendo uso desse mecanismo de interfaces. Nela, o contrato estabelecido garante que por um lado o servidor implementa um conjunto de operações de divulgação das medidas de performance e, de outro, que o cliente usa essas operações para obter os dados de performance de forma compatível.

Estruturalmente, a PDI é composta de duas partes: uma especificação de interface de serviço grid e uma API que torna simples à aplicação cliente o uso dos serviços que implementam essa interface.

A especificação é composta por uma lista de requerimentos que um serviço grid deve obedecer para que um cliente possa observar as suas características de performance historicamente. Por exemplo, a PDI requer que o serviço grid que a implementa contenha uma operação que informe quando a coleta de medidas foi iniciada.

A segunda parte, a API, tem como propósito usar um serviço grid que concorde com a PDI e, com base nele, proveja, tabelas de performance para uma aplicação cliente. Essa API já implementa o contrato do lado do cliente. Dessa forma, para se criar uma aplicação cliente compatível com a PDI é necessário apenas que a API seja usada pelas classes dessa aplicação.

Em suma, a PDI especifica a comunicação entre os softwares cliente e servidor grid. O programador que cria um serviço grid pode usar a PDI como uma lista de especificações das operações de performance. Já o programador que cria a aplicação cliente usa a API para obter os dados de performance de forma transparente.

A Figura 2 apresenta um serviço grid que implementa a PDI aos olhos de uma aplicação cliente. Essa aplicação pode ver as operações disponibilizadas pelo Globus Toolkit, pela PDI e as operações específicas do serviço grid (operações de interesse). Estas últimas são as operações que o cliente está interessado em usar.

As operações do Globus Toolkit comportam a funcionalidade básica dos serviços grid, como criação de serviços ou controle de acesso a service data. Já as operações da PDI, como *getVariablesCount* ou *getVariableValue*, controlam o acesso às variáveis de performance. Por fim, as operações de interesse são especificadas e desenvolvidas pelo programador.

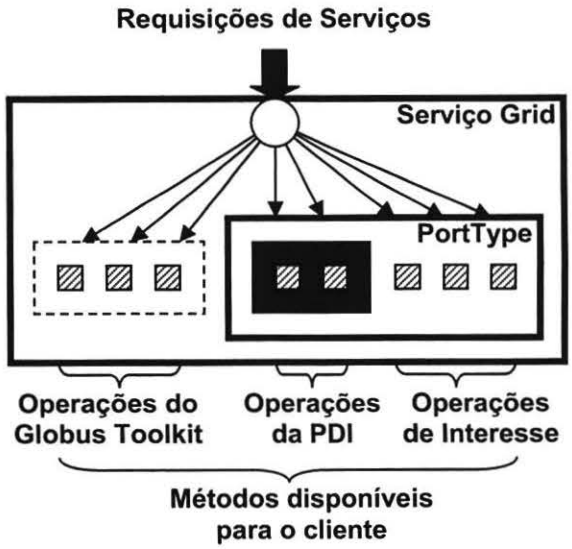

Figura 2 – Serviço grid implementando a PDI.

Tome-se, por exemplo, um serviço grid que recebe modelos tridimensionais e cria imagens bidimensionais a partir deles. Visto que o tempo de processamento desse serviço depende das características de performance do servidor, o programador que cria este serviço decide adicionar a ele o suporte a performance. Para isso, o serviço grid implementará a PDI. Então, o PortType desse serviço deve tanto fornecer as operações da PDI quanto as operações para processamento das imagens. Ou seja, as operações de interesse, nesse caso, são as de processamento de imagens.

Para declarar que o seu serviço implementa a PDI, o programador deve, no arquivo GWSDL que descreve o serviço, importar o arquivo GWSDL da PDI e fazer a declaração do serviço conforme destacado em negrito na Figura 3.

Após a alteração do arquivo GWSDL do serviço grid como na Figura 3, o compilador do Globus Toolkit passa a requerer que o serviço grid implemente todas as operações descritas no arquivo PDI.gwsdl. Para isso, o programador pode escrever esses métodos ou delegá-los para as classes da PDI criadas para esse propósito.

A Figura 4 mostra a estrutura de camadas da aplicação. Os métodos da PDI se comunicam com o sistema operacional para obter as medidas de performance, mantendo virtual a infra-estrutura necessária para a execução dos serviços.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name=" ServicoRenderizacao"

(...)

<import location="PDI.gwsdl"
namespace="http://www.usp.br/namespaces/2005/
07/lacad/Performance"/>

(...)

<gwsdl:portType name="ServicoRenderizacao"
extends="ogsi:GridService
performance:PerformanceTableSource">

<!-- declaração das operações de interesse -->
(...)

</gwsdl:portType>
</definitions>
```

Figura 3 – Detalhes das alterações no arquivo GWSDL para implementar a interface de performance

Apesar de executar no topo do Globus Toolkit, o serviço grid que implementa a PDI deve recorrer diretamente ao sistema operacional ou a outros softwares para determinar os valores das variáveis de performance. Do ponto de vista do cliente, esses recursos são apresentados virtualizados sob a forma de um conjunto de operações do serviço grid.

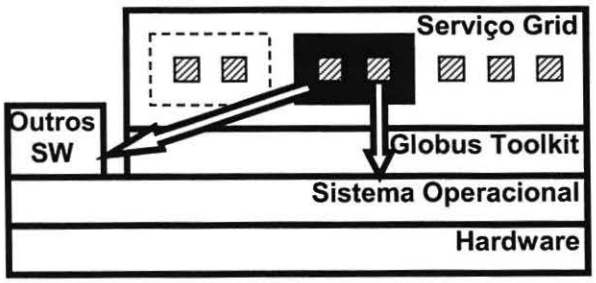

Figura 4 – Monitores de performance

4.1. Paralelos com o Gerenciamento de Rede

A PDI mantém alguns aspectos similares com a SNMP (Simple Network Management Protocol), um protocolo usado para administrar elementos de remotamente. Ao acessar métodos de um objeto que implementa a PDI, os clientes estão numa posição semelhante ao de um "application" ou "manager" segundo a nomenclatura do SNMP: uma aplicação que executa consultas buscando informações a respeito do status, configuração e performance de "network elements".

O equivalente aos agentes da SNMP na PDI são threads que fazem as medidas de performance e são executadas dentro do servidor. Os alertas providos pela PDI são um mecanismo equivalente ao "trap" da SNMP. Na PDI, esse mecanismo, construído usando o mecanismo de notificação da OGSI, avisa aos clientes cadastrados que novas medidas foram adicionadas às tabelas ou que alguma alteração foi feita nas suas estruturas.

Este mecanismo de notificação é importante para que um observador de performance possa tomar decisões a respeito de distribuição de carga assim que ocorrerem mudanças na capacidade dos servidores de processar essa carga.

O equivalente à MIB (Management Information Base) é a tabela de performance da PDI.

Assim como no SNMP, a PDI contém meta-informação administrativa. A informação administrativa está expressa sob a forma de algumas propriedades da tabela de performance, tais como nome do administrador, e-mail e telefone de contato.

4.2. Historicidade

Toda informação mantida usando a PDI é histórica, pois valores das variáveis de performance podem ser utilizados posteriormente para, por exemplo, identificar perfis de comportamento dos valores das variáveis como em [1] ou [6], calcular médias ou desvios padrão dessas medidas.

É comum que o servidor grid não esteja dedicado ao processamento de serviços grid, mas sim seja compartilhado também com tarefas da rede local na qual ele está inserido, como todos os servidores da Figura 1. Assim, os valores históricos das variáveis de performance podem permitir ao cliente concluir a respeito de quão repetível é o comportamento do servidor e estimar qual será a sua capacidade de processamento num futuro próximo.

4.3. Independência de algoritmo

Supõe-se que o algoritmo que lida com os dados de desempenho não seja conhecido. Portanto, é tarefa do programador desenvolver ou fornecer um algoritmo que trate os dados no cliente.

Assim, por exemplo, aquilo que se entende por "média" pode ser definido pelo cliente a qualquer momento. Além disso, não é papel do serviço grid implementar qualquer inteligência relacionada aos

dados de performance. Entender estes dados é papel da aplicação cliente que usa essas informações.

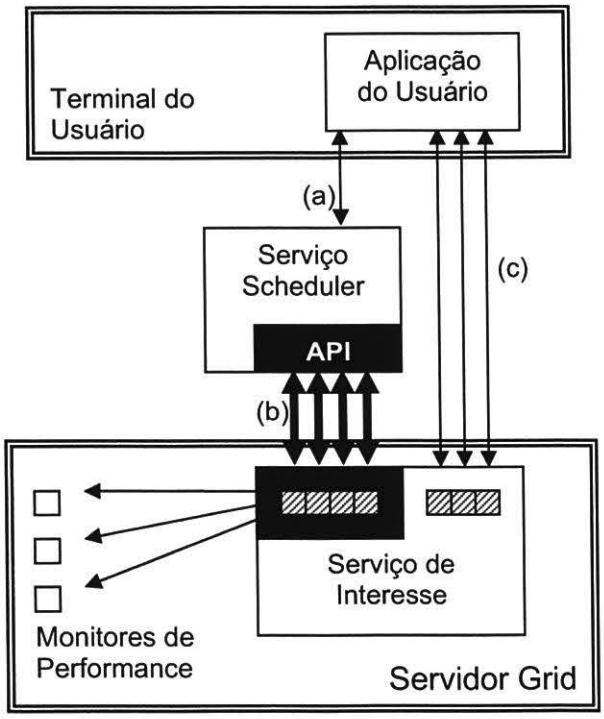

Figura 5 – Estrutura de requisições recomendada

Isso não significa que a aplicação com a qual o usuário final interage deva ser sempre responsável por tomar decisões sobre a distribuição de carga. Pelo contrário, recomenda-se que exista um serviço grid intermediário que cuide dessa distribuição. Este serviço, o scheduler, seria a aplicação cliente para a PDI, conforme o arranjo mostrado na Figura 5.

Na figura, a aplicação do usuário final pede uma operação de análise de performance (a). O scheduler, então, usando a API provida pela PDI, requisita os dados de performance (b) de um determinado servidor monitorado. Após essa análise de performance, a aplicação do usuário sabe para qual servidor devem ser feitas as requisições das operações de interesse (c).

A PDI, contribuindo com a porção destacada do diagrama da figura, permite, então, a criação de schedulers especializados, como as ferramentas estudadas em [11].

4.4. Acoplamento de variáveis

As medidas são agrupadas em variáveis, que são qualitativas ou quantitativas, dependendo da sua natureza.

Variáveis qualitativas representam características fixas e próprias do servidor, como o sistema operacional, o hardware, a quantidade de memória ou meta-informações administrativas.

Por sua vez, variáveis quantitativas, como a porcentagem de utilização de CPU, têm o seu valor variado ao longo do tempo e são expressas em uma determinada unidade de medida.

A unidade da variável utilização de CPU poderia ser chamada de "porcentagem de utilização", por exemplo. A classe que representa essa unidade é responsável por cuidar que o valor da variável seja sempre um número de ponto flutuante entre zero e um.

Tanto as variáveis de medida de performance quanto as unidades são acopláveis. Ou seja, novas variáveis de medida de desempenho podem ser criadas para suportar novas formas de avaliar a performance dos servidores.

Novas unidades de medida para as variáveis podem ser criadas implementando-se a interface *PerformanceUnit* fornecida como parte da API. Tipos primitivos da linguagem Java e alguns outros tipos básicos já estão disponíveis na API.

A Figura 6 mostra uma ferramenta integrante da PDI que possibilita a navegação pelas variáveis de performance de um servidor grid. Ao lado do nome de cada variável está o nome da unidade de medida disponível. As variáveis qualitativas estão indicadas com o tipo *Tag*, que representa uma string. A lista de variáveis aumenta ou diminui conforme variáveis são criadas e destruídas no servidor grid.

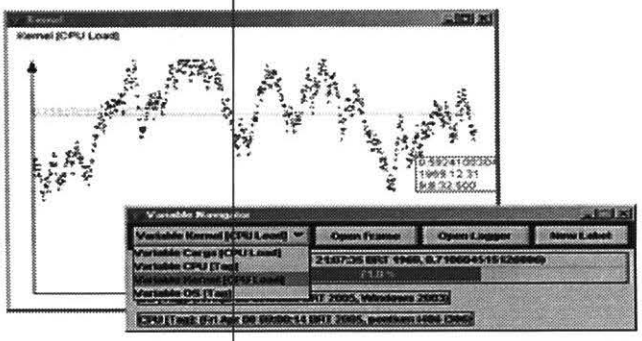

Figura 6 – Ferramenta para visualização remota de variáveis de performance

Cada servidor pode conter um conjunto diferente de variáveis disponíveis, equivalente ao conjunto de monitores de performance disponíveis. Assim, soluções de performance podem ser desenvolvidas ad hoc explorando peculiaridades de uma organização virtual.

4.5. Distribuição do Processamento

Cabe ao cliente fazer uso dessa infra-estrutura de medidas, analisar os dados, eventualmente buscando modelar o seu desempenho como em [7] ou [1], e tomar decisões sobre o destino da carga. Apesar disso, é interessante que o próprio servidor seja capaz de adiantar parte da tarefa de análise aproveitando o fato de que todos os dados estão dentro de si.

Certamente, a maior contribuição que o servidor pode fazer para acelerar a análise que os clientes executam é fornecer medidas sobre a amostra, como médias e desvio padrão. Procedendo-se dessa forma, podem ser reduzidos a quantidade de bytes trafegados pela rede e o tempo que uma aplicação cliente leva para analisar os dados.

Porém, para que seja vantajoso adiantar esse processamento, essa carga extra no servidor deve consumir o mínimo de recurso computacional. Como exemplo, as médias aritmética e geométrica poderiam ser calculadas da seguinte forma:

$$\overline{x}_{n+1} = \frac{\overline{x}_n \times n + x_{n+1}}{n+1} \quad (1)$$

$$\overline{x}_{n+1} = \sqrt[n+1]{\overline{x}_n^n \times x_{n+1}} \quad (2)$$

O desvio padrão poderia ser calculado aproximadamente da seguinte forma:

$$s_{n+1} = \sqrt{\frac{s_n^2 \times (n-1) + (x_{n+1} - \overline{x}_{n+1})^2}{n}} \quad (3)$$

Esta última fórmula é aproximada, pois recorre à média anterior. A diferença entre as duas médias origina o erro.

Nestes três exemplos, os parâmetros (as médias e desvio padrão) podem ser recalculados rapidamente. Recorre-se apenas à última medida feita, ao valor anterior de cada parâmetro e ao número de medidas. Parâmetros assim calculados estão sempre disponíveis para serem consultados pelos clientes.

O mesmo princípio pode ser usado para a identificação de perfis de comportamento das variáveis. Durante o tempo ocioso, a thread de medidas de performance pode fazer estes cálculos simplificados, não onerando significativamente o servidor.

4.6. Erros nas Medidas

Erros imprevisíveis podem ocorrer nas medidas. Por exemplo, a medida pode ser feita através de um deamon sendo executado no sistema operacional de forma independente do Globus. Este deamon pode ter conter um falha e, após alguns dias ligado, causar um vazamento de memória, parando de funcionar a qualquer momento.

Os erros devem ser reportados para os clientes, pois mesmo os erros nas medidas devem ser considerados para avaliar a confiabilidade da medida de desempenho.

5. Testes

Testes foram executados para demonstrar a capacidade da PDI de auxiliar serviços grid que tratam de distribuição de tarefas. Para fazer uma simulação em tempo real de um grid e, ao mesmo tempo, controlar essa simulação precisamente, decidiu-se criar um pequeno grid numa rede local composto por quatro computadores idênticos em software e hardware. Uma carga de teste foi aplicada a esse grid durante vinte e quatro horas.

Efeitos de atraso na comunicação entre computadores não foram considerados por serem secundários se comparados com o longo tempo que as tarefas em si levam. A variável usada para o teste foi a porcentagem de CPU livre, embora outras variáveis possam ser utilizadas de forma acessória.

Todos os quatro computadores compuseram uma organização virtual que proveu, entre outros, serviços de cálculos complexos. Três dos quatro computadores fizeram o papel de servidores em países diferentes. Cada servidor, de forma conveniente para o teste, estaria localizado em fusos horários distantes oito horas entre si. O quarto computador fez o papel de uma estação de trabalho cliente.

Além disso, presumiu-se que das oito da manhã às quatro da tarde todos os três servidores seriam requisitados em toda a sua capacidade de processamento para processar serviços dos clientes nas suas redes locais, como mostrado na Figura 7. Esta simulação procura reproduzir um dia de trabalho de oito horas em três países em fusos horários diferentes e complementares, escolhidos convenientemente.

Para o teste, as tarefas requisitadas por clientes nas redes locais sempre têm prioridade maior do que as requisições grid. Então, durante o período no qual usuários da rede local estão usando cada servidor, os processos grid simplesmente aguardariam oito horas para continuarem a sua execução.

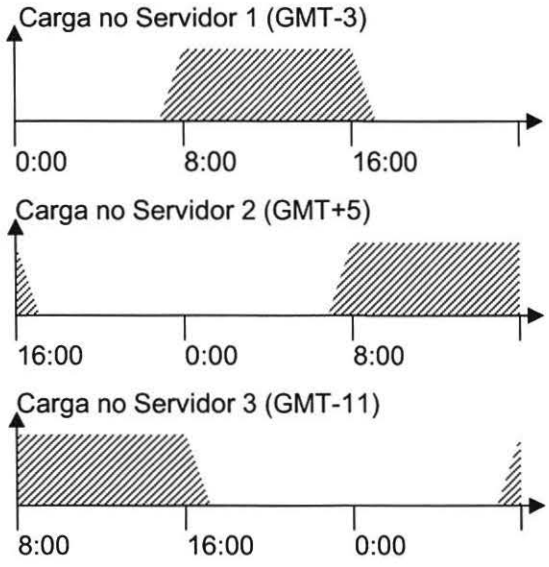

Figura 7 – Perfis de carga diária dos três servidores utilizados

Os testes foram executados em três diferentes condições de execução: (a) sequencialmente, usando apenas um dos servidores, (b) usando os três servidores em paralelo ignorando todo dado histórico, e (c) usando os três servidores em paralelo e considerando os dados históricos de carga. O primeiro teste foi usado como base de comparação para permitir o julgamento da performance dos outros dois.

Ao ignorar a informação histórica de carga, o algoritmo simplesmente distribuía tarefas para o servidor ocioso no momento, ignorando as prováveis concorrências que essa atribuição de carga causaria entre a execução de sub-tarefas do grid e as tarefas normais requeridas pelos clientes locais.

Nos testes usando dados históricos, o algoritmo calculou a duração para cada tempo esperado para a duração do processamento. Então, o algoritmo verificava se o envio de carga para um determinado servidor causaria concorrência entre tarefas, baseado no que já se conhecia historicamente sobre o perfil de carga. O algoritmo apenas atribuía carga a um servidor se nenhuma concorrência acontecesse. Isso evita que uma sub-tarefa seja interrompida e aguarde até oito horas para continuar a processar.

A granulosidade da divisão das tarefas em sub-tarefas também foi variada tal que o efeito da predição da carga pudesse ser observado. Para este propósito, uma tarefa que levaria vinte e quatro horas de processamento foi dividida em sub-tarefas de durações de 1, 3 e 6 horas. Os tamanhos dessas sub-tarefas foram chamados respectivamente de granulosidade "baixa", "média" e "alta". O grau de granulosidade foi dado com base na comparação com o período de trabalho de oito horas.

Para este primeiro teste, a tarefa foi um cálculo algébrico sem interdependências. Então, dividir a tarefa em sub-tarefas foi bastante simples.

Tabela 1 – Resultados das medidas experimentais

	Granulação Alta (6 horas)	Granulação Média (3 horas)	Granulação Baixa (1 hora)
a) Seqüencial	24 h	24 h	24 h
b) Paralelo sem Histórico	20 h (60%)	17 h (70,59%)	12 h (100%)
c) Paralelo com Histórico	14 h (71,43%)	14 h (85,71%)	12 h (100%)
Ganho pelo uso do Histórico	25%	12,5%	0%

A Tabela 1 contém os resultados dos testes executados. Os números entre parêntesis se referem à porcentagem da capacidade de processamento utilizada. A última linha se refere ao ganho de tempo pelo uso dos dados históricos diante do tempo total seqüencial para cada granulação.

Esses resultados demonstraram que tarefas de maior granulação tomaram mais proveito da existência de dados históricos. No caso de a granulação ser baixa, a capacidade de predição não oferece vantagens consideráveis diante da simples medida instantânea de performance. Isso se explica pelo fato de que quanto menor a granulação de uma tarefa, tanto menos o comportamento do servidor varia durante a execução da tarefa, e tanto mais a medida da carga instantânea é suficiente para o balanceamento de carga.

Pode-se observar pela Tabela 1 que durante os testes feitos, apenas dois servidores estavam disponíveis efetivamente a cada instante. Por isso a performance máxima obtida tende a ser aquela que se observaria ao utilizar dois servidores em paralelo.

Um segundo teste foi executado, no qual oito sub-tarefas foram executadas pelo grid, seguindo as relações de precedência representadas na Figura 8.

Cada uma das sub-tarefas T1, T2,... , T8 tinha duração de 3 horas. Ao final do teste, o ganho pelo uso do histórico foi de 25% em relação ao teste sem histórico. Comparando-se este resultado com os da Tabela 1, vemos que o ganho pelo uso do histórico é maior nos casos em que há dependências entre as tarefas, para uma mesma granulação.

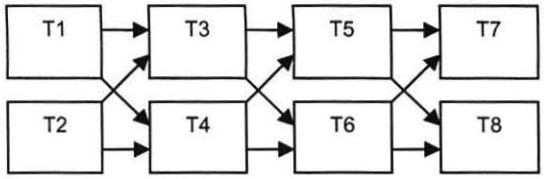

Figura 8 – Relações de precedência entre as sub-tarefas para o segundo teste

Ou seja, quanto maior é a complexidade da distribuição de carga, tanto mais o escalonamento usando predição de desempenho oferece vantagens em comparação ao escalonamento sem predição.

6. Conclusões e Trabalhos Futuros

Neste artigo foi apresentada uma proposta de extensão da OGSI para novos tipos de serviços acessórios aos serviços Grid.

Apesar de se basear na OGSI, a abordagem de predição de desempenho para melhoria da distribuição de carga apresentada neste artigo pode ser usada para qualquer tipo de grid, pois todo grid tal como descrito em [4] e [5] lida com o problema da concorrência dos serviços do grid com os serviços das redes locais das instituições participantes da organização virtual.

Este trabalho abre espaço para implementações de sistemas auto-adaptáveis nos quais a performance global é otimizada em relação à capacidade de computação dos elementos que o compõem.

Foram feitas diversas simplificações para os testes. Supuseram-se os componentes da organização virtual homogêneos, e a subdivisão de tarefas foi bastante simples. Além disso, o algoritmo usado foi simplista ao usar somente o critério de não concorrência para atribuição de carga. Apesar disso, em geral, os resultados demonstraram que o conhecimento do histórico de performance dos servidores de uma rede grid traz grandes benefícios para a otimização do uso dos recursos computacionais.

Para os testes feitos, as medidas de performance se comportaram corretamente, mas prevê-se uma degradação do serviço quando a monitoração acumular muitos dados. Assim, um trabalho futuro deve discutir um modelo para ao mesmo tempo descartar medidas, e permitir a mesma qualidade de análise de desempenho dos servidores grid.

7. Referências

[1] B. Lee, J. Schopf, "Run-time Prediction of Parallel Applications on Shared Environments", Poster Paper, Proceedings of Cluster2003, Dezembro de 2003

[2] I. Foster, "Service-Oriented Science", Science, vol. 308, 06/05/2005

[3] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy, "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation", International Workshop on Quality of Service, 1999

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

[5] I. Foster, C. Kesselman, J. Nick and S. Tuecke, "The Phisiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", International J. Supercomputer Applications, 2001

[6] L. Yang, J.M. Schopf, I. Foster, "Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments" Supercomputing 2003, Novembro de 2003

[7] M. Ripeanu, A. Iamnitchi, I. Foster, "Cactus Application: Performance Predictions in Grid Environments", EuroPar 2001, Manchester, UK, Agosto de 2001

[8] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt and D. Snelling, "Open Grid Services Infrastructure (OGSI) Version 1.0.", Global Grid Forum Draft Recommendation, 27 de Junho de 2003

[9] The Globus Alliance: http://www.globus.org

[10] W. Smith, I. Foster, V. Taylor, "Scheduling with Advanced Reservations", Proceedings of the IPDPS Conference, Maio de 2000

[11] X. Zhang, J. Freschl, J. Schopf, "A Performance Study of Monitoring and Information Services for Distributed Systems", Proceedings of HPDC, Agosto de 2003