Implementação de escalonamento de buscas de banco de dados em grid computacional

Celso Henrique Poderoso de Oliveira[1], Prof. Dr. Maurício Almeida Amaral[2]

[1]FIAP – Faculdade de Informática e Administração Paulista
Av. Lins de Vasconcelos, 1264 – Aclimação – São Paulo – SP

[2]CEETEPS – Centro Paula Souza
Rua dos Bandeirantes, 169 – Bom Retiro – São Paulo – SP

cpoderoso@fiap.com.br, malmeida@inteligenciaartificial.eti.br

1. Introdução

Com o anúncio de banco de dados comerciais que utilizam a tecnologia de grid computacional para melhorar o desempenho de aplicações e aumentar a capacidade de armazenamento através da utilização de diversos computadores de baixa plataforma, as organizações empresariais começaram a se interessar pelo uso da tecnologia. Um dos grandes obstáculos para que isso se realize é a pouca aderência dos sistemas gerenciadores de bancos de dados a este ambiente. Empresas comerciais utilizam sistemas com bancos de dados relacionais e objeto-relacionais para manter e armazenar dados. Sistemas acadêmicos utilizam, em sua maioria, sistemas baseados em arquivos.

O grupo DAIS (*Data Access and Integration Services*) estabelece as diretrizes para a integração da arquitetura OGSA (*Open Grid Services Architecture*) e os sistemas gerenciadores de banco de dados. O grupo DAI (*Data Access and Integration*) implementa as sugestões do DAIS em um *middleware* (OGSA-DAI) com base nas definições de 2003. Mesmo que o *middleware* possibilite a integração de sistemas gerenciadores de banco de dados em uma grid computacional, ainda há diversas limitações. Uma das limitações é a possibilidade de escalonar as buscas.

Este artigo demonstra uma solução de escalonamento do comando de busca desenvolvida para funcionar como um serviço da OGSA-DAI. Para isso foi criada uma atividade no *middleware* que recebe as requisições do usuário, escalona e particiona a busca, submete aos bancos de dados disponíveis na grid que processa e retorna o resultado das buscas.

O artigo está dividido da seguinte forma: na seção 2 são discutidos os fundamentos da grid computacional. Na seção 3 são discutidas as formas de integração do banco de dados em uma grid computacional e os métodos de escalonamento. Na seção 4 estão listados os resultados alcançados com a implantação do algoritmo e na seção 5 são colocadas as considerações finais e conclusões.

2. Grid Computacional

Como os processadores têm evoluído em menor velocidade nas últimas décadas devido às limitações físicas dos componentes utilizados [4], o processamento distribuído procura contornar esta situação ao ampliar o uso dos equipamentos envolvidos no processamento dos dados.

Entende-se grid computacional como um conjunto de recursos heterogêneos, distribuídos e integrados que são compartilhados como se fossem um único e que utilizam redes de altíssima velocidade [5]. É um ambiente em que se permite o balanceamento de

carga, segurança de acesso e é tolerante a falhas. Dentre as principais características de uma grid destacam-se a descentralização, amplitude e heterogeneidade dos recursos e serviços que são utilizados para prover o processamento e armazenamento de dados em uma rede.

Baseado nas definições clássicas de uma grid computacional, um único computador pessoal é uma grid ou que, por estar conectado a uma rede local, esteja em uma grid de *clusters* ou ainda que, por permitir o armazenamento em diversos recursos, esteja em uma grid de armazenamento [6]. Como se nota, é possível obter diversas definições de grid computacional e todas têm um certo grau de coerência.

2.1 Organização Virtual

Entende-se Organização Virtual como um grupo de organizações ou indivíduos que compartilham recursos de forma controlada. Os participantes de uma comunidade podem utilizar recursos comuns entre si para atingir um objetivo [5].

As organizações virtuais são formadas por interesses comuns. Pode-se considerar as empresas, centros de pesquisa e universidades como exemplos de Organizações Virtuais [4]. Métricas de desempenho, prioridade, expectativas e limitações de cada recurso, escalonamento e balanceamento de tarefas são extremamente importantes quando se trabalha com organizações virtuais [5]. Uma grid computacional é criada para melhorar a utilização dos recursos de uma organização virtual.

2.2 *Middleware* para grid computacional

Entende-se *middleware* como um software que conecta duas ou mais aplicações. De acordo com Foster [5], um *middleware* para grid seria a união de protocolos, serviços, APIs (*Application Programming Interface*) e SDKs (*Software Development Kit*). É fundamental a existência de *middleware* para permitir a interoperabilidade entre as organizações virtuais e a padronização de protocolos de controle e acesso [5].

Entre eles, destacam-se: *Globus Toolkit*, *Storage Resource Broker*, *Gridbus*, *Alchemi* e *Legion*. Entre estes, o *Globus* é o que tem maior aceitação. O *Globus* é um conjunto de ferramentas que realiza o trabalho de conectar e gerenciar recursos. É formado por um consórcio de várias empresas de tecnologia interessadas em disseminar a utilização das Grids Computacionais. Utiliza o GridFTP como mecanismo básico de acesso e transferência de dados. É integrado ao *Grid Security Infrastructure* (GSI) que provê os serviços de segurança da Grid, entre eles o controle de acesso [8]. O *Globus Resource Allocation Manager* (GRAM) é responsável pela interface do *middleware* com os recursos locais e o *Metacomputing Directory Services* (MDS) realiza a localização e informação dos recursos (serviço de diretório).

2.2.1 Arquitetura e Infra-estrutura de Grid

A infra-estrutura (OGSI - *Open Grid Services Infrastructure*) trabalha em conjunto com a arquitetura (OGSA - *Open Grid Services Architecture*) da grid para permitir a interação entre os aplicativos dos usuários e os serviços Web (*Web Services*).

Pode-se entender a OGSI como um conjunto de especificações WSDL (*Web Services Description Language*) que define padrões de interface e ações para a OGSA [4]. Os serviços de grid podem ser persistentes ou transientes, ou seja, de curta duração. Os serviços Web (*Web Services*) concentram-se no grupo de serviços persistentes, ou seja, de longa duração. A figura 1 mostra o modelo OGSA.

A OGSA fornece a descrição do ambiente através de especificações WSDL e uma coleção de interfaces de serviços através de um mecanismo de padronização para criação, identificação, comportamento e descoberta de instância de serviços. Por sua vez o WSDL utiliza o XML para descrever os *Web Services* [13]. Também são utilizados: protocolo *Simple Object Access Protocol* (SOAP) e *Web Services Inspections Language* (WSInspection).

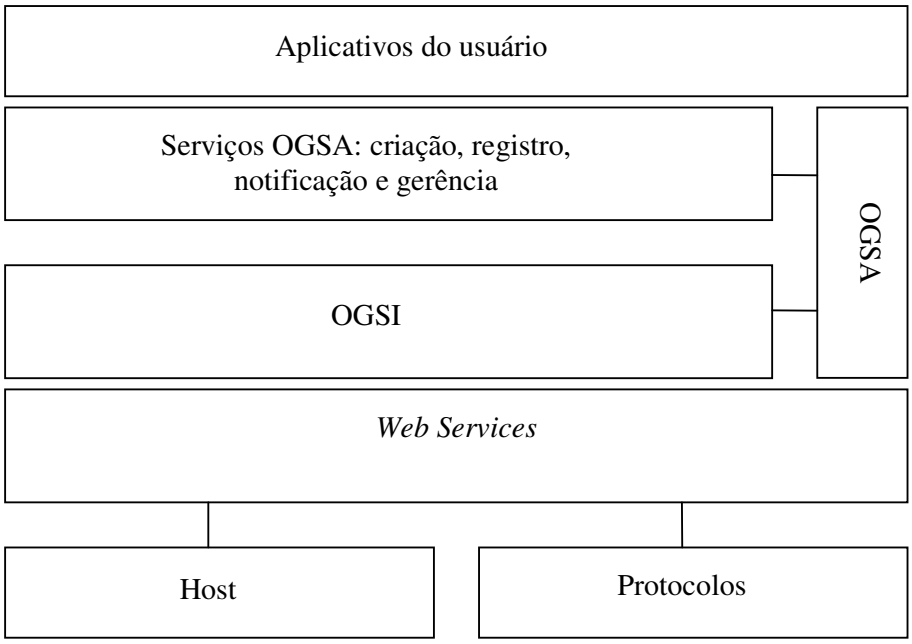

Figura 1: Modelo OGSA [4].

3. Banco de Dados em Grid Computacional

Os SGBDs (Sistemas Gerenciadores de Banco de Dados) possuem recursos muito mais avançados do que os sistemas baseados em arquivos de dados e são amplamente utilizados no ambiente empresarial. Os SGBDs possuem recursos que permitem realizar transações com os dados de forma segura e ágil. Por este motivo é importante que se crie condições de integração dos SGBDs comerciais à grid.

É consenso entre os estudiosos de banco de dados e de grid que não se deve criar um novo produto para ser utilizado na grid computacional [14]. O escopo da proposta segue alguns parâmetros aceitáveis tanto do ponto de vista da grid quanto dos SGBDs [13]:

- **Independência de Grid**: não se deve restringir a um *middleware* exclusivo.
- **Não criar um novo Banco de Dados**: estabelecer mecanismos de controle de transações, buscas e outros serviços que existem nos principais SGBDs.
- **Independência de Banco de Dados**: não se deve seguir um produto comercial único para estabelecer o modelo de dados.

A este mecanismo que dá-se o nome de Banco de Dados Federalizado ou Banco de Dados Virtual (figura 2).

A diferença entre uma integração realizada com bancos de dados distribuídos e bancos de dados virtuais é que o controle, no primeiro caso, está centralizado em um único servidor. Em uma grid, os SGBDs devem guardar autonomia sobre seus processos. Serão requisitados sempre que necessário, mas não estarão disponíveis apenas para uso na grid. Com isso, a distribuição das buscas é possibilitada.

Algumas das mais importantes questões levantadas por Watson [14] para adequar um SGBD a uma grid são: integração dos mecanismos de segurança, programação de banco de dados e integração com *Web Services*, escalonamento de utilização com programação de utilização futura, monitoramento do desempenho e publicação de metadados.

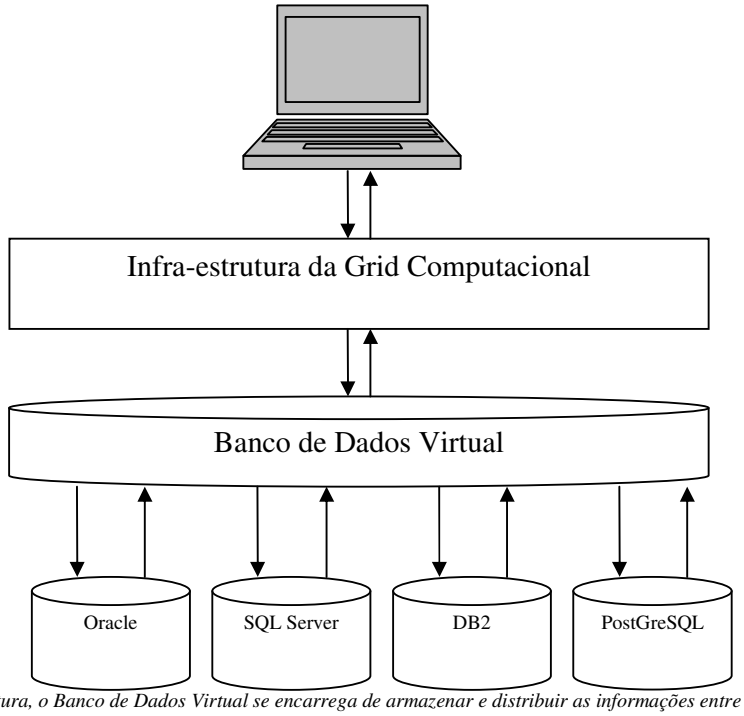

Figura 2 – Nesta estrutura, o Banco de Dados Virtual se encarrega de armazenar e distribuir as informações entre os bancos de dados que estiverem na Grid Computacional. As aplicações devem se comunicar com o Banco de Dados Virtual e ele resolve as demais questões. A comunicação do banco de dados virtual pode se dar com dados estruturados, semi-estruturados ou não estruturados

.

3.1 Serviço de Banco de Dados

Entende-se um Serviço de Banco de Dados como qualquer serviço que suporte uma interface de banco de dados que esteja integrado a um *middleware*. Os serviços de banco de dados não indicam "*como*" a informação será armazenada e também não especificam o formato ou modelo de armazenamento (relacional, objeto, XML, etc.).

As operações devem ser executadas completamente ou não devem realizar trabalho algum, exatamente como sugere o padrão SQL [11]. Em uma Grid Computacional as transações são grandes e há, portanto, necessidade de controlar interrupções e gerenciar falhas.

3.2 Serviços de Banco de Dados integrados à OGSA

Um serviço de dados (*Grid Data Service*) da OGSA é um serviço da Grid que implementa um ou mais interfaces para acesso ou gerenciamento de recursos distribuídos [7].

Para que haja uma definição clara do que é um serviço de dados foram definidas quatro interfaces básicas para se implementar em diferentes comportamentos. Estas interfaces representam portas específicas do WSDL. Qualquer serviço que implemente uma destas interfaces é considerado um *Web service* compatível com a OGSI. As quatro interfaces são [7]: *DataDescription* (define um elemento do serviço de dados da OGSI), *DataAccess* (provê operações de acesso e modificação de conteúdo dos dados virtualizados), *DataFactory* (provê operações para criar um novo serviço de dados em uma nova virtualização) e *DataManagement* (operações para monitorar e gerenciar os serviços de dados da virtualização).

O grupo DAIS (*Data Access and Integration Services*) trabalha atualmente na definição dos serviços de bancos de dados. Para isso foram utilizadas as quatro interfaces de dados citadas anteriormente e que devem ser implementadas para atender aos comportamentos dos bancos de dados. Segundo Alpdemir [1], o principal objetivo do OGSA-DAI (*Data Access and Integration*) é criar uma infra-estrutura para entrega de serviços de alta

qualidade e que agreguem valor ao gerenciamento de dados da Grid, especificamente nos *Grid Database Services* (GDS).

3.3 Escalonamento

Para resolver problemas de planejamento e escalonamento de recursos é comum utilizar técnicas de Inteligência Artificial. De acordo com Nareyek [9], o problema de planejamento compreende uma descrição do estado inicial, uma descrição parcial dos objetivos e um conjunto de ações que mapeiam a transformação dos elementos de um estado para outro.

De acordo com Blythe [2], atribuir uma série de atividades em um fluxo de trabalho e alocar recursos a cada atividade é um problema de planejamento de Inteligência Artificial. Cada componente que faz parte do fluxo de trabalho é modelado como um operador de planejamento. Os efeitos e precondições dos operadores refletem a dependência de dados entre as entradas e saídas do programa e as necessidades e restrições dos recursos em termos de equipamento e software. O plano determinará a ordem de execução das atividades através das entradas e saídas esperadas.

De acordo com a abordagem proposta por Casavant [3], os métodos de escalonamento são divididos em local e global. No escalonamento de banco de dados em Grid, contudo, interessa o escalonamento global porque se podem aplicar seus métodos em sistemas distribuídos. O escalonamento global divide-se em: estático, quando a atribuição dos processos é feita no início do planejamento e não é possível alterá-lo durante o processamento; e dinâmico, quando os processos são alocados e acompanhados dinamicamente, ou seja, com possibilidade de alteração durante o processamento.

Os principais *frameworks* utilizados para escalonamento, tanto estático quanto dinâmico, com restrições são [9]: SAT (*Satisfiability*), IP (*Integer Programming*) e CP (*Constraint Programming*).

Devido às características deste trabalho, a utilização de *Constraint Programming* se mostrou mais adequada porque o escalonamento em grid computacional envolve o planejamento de recursos e tempo em ações concorrentes.

3.4 Fatores importantes para programação de recursos em Grid

De acordo com Ranganathan [12] alguns dos fatores mais importantes para programar recursos em grid computacional são: utilização do recurso, tempo de resposta, política global e local de acesso e escalabilidade. Forma identificados quatro parâmetros importantes que afetam o desempenho em uma Grid: banda de rede, padrões de acesso de usuários, qualidade da informação utilizada para realizar o trabalho e a relação entre o tempo de processamento e tamanho dos dados para determinar a importância da localização dos dados.

4. Solução e Resultados Alcançados

O objetivo foi criar um mecanismo que pudesse realizar buscas do usuário para execução nos diversos bancos de dados disponíveis na grid computacional, mais especificamente na OGSA-DAI. A proposta envolve a utilização de padrões da arquitetura de grid, com a disponibilização de um serviço de banco de dados em uma organização virtual. Para tanto a pesquisa envolveu as principais técnicas para extração e atribuição de heurísticas para solucionar problemas de escalonamento.

Procurou-se utilizar todos os recursos disponíveis na OGSA-DAI que, por sua vez, utiliza o *Globus Toolkit*. Para extração das métricas optou-se por não utilizar o MDS porque este se limita a identificar os recursos e apurar dados básicos dos equipamentos envolvidos. Os principais gerenciadores de banco de dados são capazes de extrair diversas outras informações relevantes sobre o ambiente no qual estão instalados. Optou-se por criar uma tabela com as informações mais relevantes para um gerenciador de banco de dados: uso de CPU, memória disponível, banda de rede disponível e volume de Entrada/Saída de dados.

Esta tabela pode ser alimentada automaticamente pelo próprio gerenciador de banco de dados, pela simples criação de procedimentos que atualizem periodicamente os valores disponíveis no servidor.

A proposta permite que se estabeleçam pesos específicos para cada um dos elementos analisados e possa, inclusive, acrescentar outros que sejam relevantes para a organização virtual. Isso se dá pela adição de outros elementos (linhas na tabela de métrica) que se julgar importante, atribuição de peso específico e apuração da métrica. Com isso a heurística passa a refletir o peso que se atribui a cada um dos recursos disponíveis e permite ao planejador escolher, dentre os bancos de dados disponíveis, os que poderão fornecer a resposta com maior rapidez. Esta técnica é adequada para se utilizar os conceitos apresentados em *Constraint Programming*.

É importante notar, contudo, que o foco deste artigo não está no estabelecimento e atualização das métricas e sim na utilização destas para selecionar os gerenciadores de banco de dados que possam executar as buscas. Também não é foco deste trabalho a separação dos elementos que compõem o comando SELECT. Com isso abre-se espaço para que outras pesquisas possam melhorar o que foi realizado e se implemente soluções de melhor desempenho.

O objetivo deste planejador é utilizar a heurística definida no ambiente da organização virtual e as sub-buscas do comando SELECT para selecionar e submeter aos gerenciadores de banco de dados.

Sabe-se que o planejador deve se basear em um *middleware* que já exista e tenha suporte às principais atividades de um banco de dados além de não ter vínculo exclusivo com uma única implementação comercial de banco de dados. Isso faz com que haja necessidade de estabelecer uma interface única para qualquer tipo de banco de dados. Sabe-se que a melhor forma de realizar trabalhos em uma grid computacional é estabelecer o desenvolvimento baseado em serviços. A integração com o *middleware* é importante para que não se limite e se permita a evolução da solução apresentada.

Foram utilizadas cinco tabelas replicadas em até cinco sistemas gerenciadores de banco de dados. Para realizar os testes, utilizou-se um conjunto crescente de gerenciadores disponíveis na grid.

4.1 Fases do Escalonador

O sistema planejador será dividido em quatro partes principais, cada uma com sua função específica:

- Interceptar as requisições do usuário que são enviadas para o *middleware*
- Solicitar ao *middleware* as informações dos recursos disponíveis na grid computacional
- Separar a busca submetida pelo usuário em sub-buscas que permitam o escalonamento da requisição
- Programar os recursos e distribuir as solicitações para que o *middleware* execute as buscas em cada um dos SGBDs.

A figura 3 demonstra as entradas e saídas esperadas pelo planejador. Deve haver um processo que identifique o estado atual dos recursos e outro para definir as necessidades do usuário. Ambos servem para iniciar o processo de planejamento. A identificação dos recursos e a forma de comunicação com os SGBDs serão delegadas ao OGSA-DAI. As necessidades do usuário deverão estar especificadas dentro do formato utilizado na OGSA-DAI.

O usuário informa em um arquivo XML sua necessidade de busca dentro da estrutura proposta na OGSA-DAI. O planejador solicita ao *middleware* informações dos recursos disponíveis e verifica quais recursos possuem as tabelas necessárias para execução do comando. O planejador particiona a busca em comandos que possam ser executados paralelamente em diversos SGBD e vincula aos recursos disponíveis. Após o estabelecimento

do plano de execução, o serviço será enviado à Grid Computacional para ser processado nos gerenciadores de bancos de dados. Terminado o processo, o planejador une as linhas retornadas e devolve ao usuário que solicitou a busca.

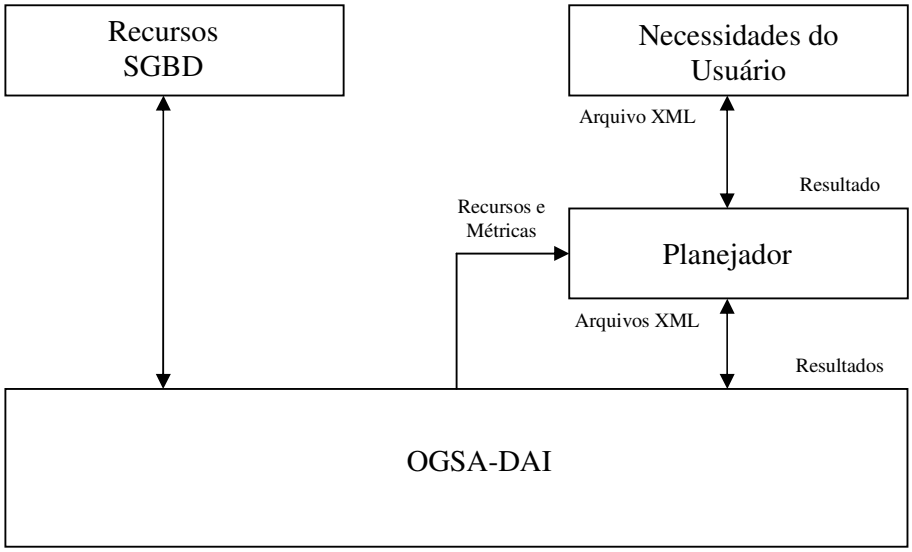

Figura 3 – Entradas e Saídas do sistema planejador

4.2 Algoritmo desenvolvido

O escalonamento se dá através da paralelização de uma única consulta submetida à grid. Esta consulta, contudo, possui diversas tabelas cujas subconsultas podem ser submetidas aos diversos bancos de dados disponíveis na grid. Com isso obtém-se ganho no desempenho global da consulta. Este trabalho, da mesma forma que o trabalho relacionado (OGSA-DQP), não realiza a busca em tabelas particionadas em diversos bancos de dados. Utiliza-se, portanto, uma subconsulta para cada recurso disponível. O particionamento da busca ocorrerá sempre que a métrica for inferior ao limite máximo estabelecido por parâmetro.

Os passos utilizados pelo algoritmo de escalonamento estão demonstrados na figura 4.

```
float a = getThreshold()
Resource rsc = getResources( handle )
Query qry = splitQuery( str )
DistQuery dstqry = checkDataResources( rsc, qry)
repeat
   Metric mtr = getMetrics( rsc )
   repeat
      if CheckQuery( dstqry, mtr ) < a
         fnl = addQueryResource( dstqry )
   until dstqry
until rsc
Result rs = submitQuery( fnl )
```

Figura 4 – Algoritmo desenvolvido

Um banco de dados relacional depende basicamente de quatro elementos da arquitetura: processador, rede, memória e meio de armazenamento físico (E/S). A heurística recupera informações relacionadas estes elementos em cada implementação de banco de dados.

Inicialmente é feita a identificação de cada um dos recursos disponíveis. Em seguida separa-se os argumentos do comando SELECT. Faz-se a vinculação de cada elemento do comando de busca com os possíveis recursos. O algoritmo contém dois laços: o primeiro é executado em cada um dos recursos disponíveis para extrair as métricas e o segundo, após o vínculo do recurso com a tabela, identifica se o limite operacional foi ou não ultrapassado para que se particione a busca.

Como parâmetro de entrada, é interceptado o arquivo XML utilizado para processamento da consulta no *middleware* (OGSA-DAI). Foi criado um parâmetro no arquivo XML para especificar o limite aceitável para escalonamento da busca. São solicitadas ao *middleware* informações sobre quais são os recursos disponíveis na grid.

Com base nestas três informações, são lidos os recursos para os objetos e importados os objetos de consulta. Um método irá separar cada uma das tabelas e respectivas colunas de busca. Deste método retornam, além das tabelas e colunas, as uniões entre as tabelas e as eventuais funções a serem utilizadas no banco de dados.

Para cada item da consulta, são identificados quais bancos de dados estão disponíveis para busca. Ao mesmo tempo será possível extrair métricas de desempenho do banco de dados. Em seguida para cada tabela necessária para realizar a consulta são identificados os recursos disponíveis. Isso é feito através de uma busca no dicionário de dados do banco de dados.

Para cada uma das tabelas já vinculadas ao recurso, é verificado se o limite operacional estabelecido como parâmetro é superior à métrica extraída do banco de dados. Caso seja, cria-se uma base de consulta XML dentro do padrão do *middleware*.

No final do processo há um ou mais arquivos XML com a lógica necessária para que sejam processados no *middleware*.

O algoritmo prevê uma única modificação no padrão que já existe e está em uso pela comunidade: o limite operacional para aceitação do escalonamento. Pode-se, contudo, estabelecer um limite razoável que atenda a maior parte das solicitações, sem prejudicar o desempenho do planejamento. O usuário continuará criando seu arquivo XML para processamento no *middleware* e receberá o resultado sem a necessidade de modificar o que é feito atualmente.

Não se corre o risco de estourar o tempo de escalonamento, visto que o limite do algoritmo está no número de tabelas da busca e no número de recursos disponíveis. A menos que haja um volume extremamente grande de recursos, o tempo de processamento deve ser coerente com o ganho na distribuição do processamento.

Uma outra vantagem do algoritmo é que, caso não haja recurso disponível para tratar a solicitação, este será imediatamente interrompido.

4.3 Resultados

Os testes foram realizados com a OGSA-DAI WSRF versão 2.2 que, por sua vez, utiliza o *Globus Toolkit* 4.0.2. Ambos foram instalados no Apache Tomcat 5.5, de acordo com a especificação mínima estabelecida nas respectivas documentações.

O escalonador se comportou como esperado: a busca realizada dentro da linguagem SQL e com a utilização do padrão da OGSA-DAI fez com que a atividade desenvolvida fosse capaz de escalonar e paralelizar a busca nos bancos de dados disponíveis.

Nos testes foram utilizados três computadores sendo um Pentium IV, 3.2 Ghz, 1GB de memória RAM e 80 GB de disco com banco de dados Oracle 10gR2 e DB2 e com o *middleware* OGSA-DAI WSRF versão 2.2; um Pentium IV, 1.4 Ghz, 750MB de memória RAM com banco de dados Oracle 10gR1 e PostGreSQL e 40 GB de disco; e um Pentium III, 750 Mhz, 750 MB RAM e 20 GB de disco com banco de dados PostGreSQL. Este ambiente procura simular a situação real de uma organização virtual, visto que há recursos finitos, com diferentes gerenciadores disponíveis e com tabelas replicadas. A submissão pode ser realizada

em qualquer computador da rede, sempre com a utilização do servidor de aplicação disponível.

Ao enviar o comando de busca, em todos os casos em que havia uma mesma tabela disponível em mais de um banco de dados o algoritmo dividiu a busca entre os recursos. Somente quando não havia duas tabelas em servidores diferentes não houve o particionamento. Houve uma nítida opção por utilizar o Oracle 10gR2 e DB2 sempre que possível, pois que estavam no mesmo computador que o *middleware*. A última opção sempre ficou com o Pentium III, por ser o processador de pior desempenho. Foram utilizadas variáveis para limite de escalonamento altas, visto que o objetivo era testar a distribuição e paralelização da busca. Devem-se realizar novos testes para atestar a eficácia do algoritmo sob este aspecto.

Os resultados obtidos podem ser vistos na figura 5. O gráfico mostra a quantidade de tabelas utilizadas na cláusula FROM do comando SELECT e a atribuição do escalonador de partes da busca em cada um dos bancos de dados disponíveis na grid. No eixo ***x*** têm-se os recursos disponíveis na grid (banco de dados) e no eixo ***y*** tem-se a quantidade de tabelas escalonadas para cada um dos bancos de dados em função das tabelas utilizadas no comando SELECT.

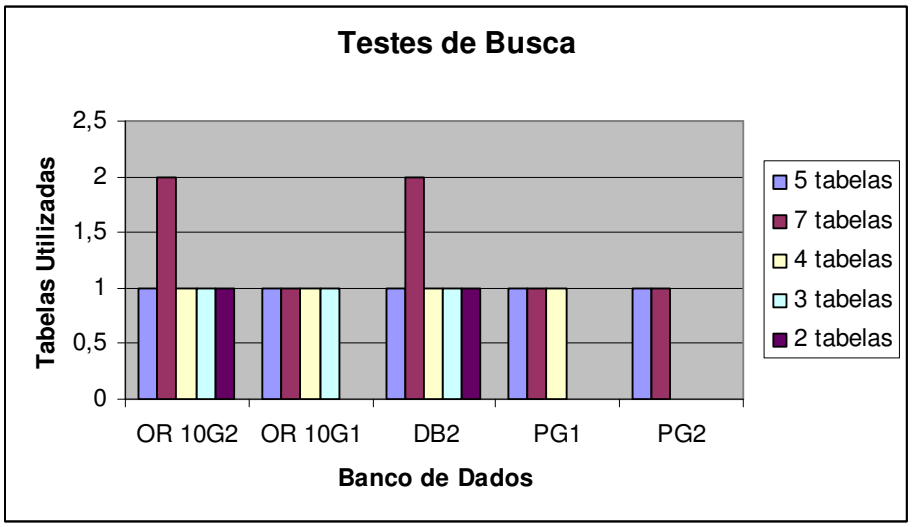

Figura 5 – Resultados do teste de busca com a quantidade de tabelas especificadas no quadro na cláusula FROM do comando SELECT.

4.4 Trabalho relacionado

Um trabalho equivalente tem sido desenvolvido para criar um serviço de buscas distribuídas. Denominado OGSA-DQP (*Distributed Query Processing*), o objetivo da proposta é prover uma maneira de implementar linguagens declarativas e de alto nível para acesso e análise de dados além de permitir a facilidade de gerenciamento de recursos que uma grid computacional possui.

O maior problema, contudo, é não utilizar todos os recursos da OGSA-DAI, o que limita a utilização ao MySQL, e realizar o particionamento da busca no projeto Polar* [10]. Outro problema é a utilização do OQL (*Object Query Language*) ao invés do SQL, padrão dos gerenciadores de bancos relacionais.

5. Conclusão

Com o desenvolvimento do algoritmo foi possível criar um mecanismo de escalonamento que está completamente integrado à Grid Computacional. Ao utilizar o principal *middleware* para acesso e manutenção de dados, o OGSA-DAI, garante-se que a solução estará disponível para qualquer novo banco de dados e poderá seguir o mesmo padrão

especificado pela equipe que trabalha na definição do que será feito nas próximas versões do produto.

Algumas das principais limitações de utilização de sistemas gerenciadores de banco de dados no *middleware*, como utilização de um único banco de dados, limitação da localização do banco de dados, restrição do tipo de dado de busca e não localização de recursos através do OGSA-DAI foram vencidas com esta aplicação.

A principal contribuição deste artigo foi demonstrar que é possível utilizar as extensões do padrão OGSA-DAI para realizar o escalonamento e paralelização de buscas em sistemas gerenciadores de banco de dados. Abre também espaço para que novas pesquisas sejam direcionadas para aprimorar o algoritmo com a utilização de técnicas de Inteligência Artificial, estabelecer mecanismos de replicação de dados na grid e armazenar as heurísticas para que o planejador seja capaz de tomar decisões de planejamento com base na utilização dos recursos.

Por fim, com a criação de mecanismos que facilitam a utilização de sistemas gerenciadores de banco de dados em grid, abre-se a possibilidade para que mais Instituições de pesquisa e organizações comerciais utilizem esta tecnologia para otimizar o uso de seus recursos computacionais.

6. Referências bibliográficas

[1] Alpdemir, M. N.; Mukherjee, A.; Paton, N. W.; Watson, P.; Fernandes, A. A. A.; Gounaris, A.; Smith, J. **Service-Based Distributed Querying on the Grid.** : 2003.

[2] Blythe, J.; Deelman, E. e Gil, Y. **Planning and Metadata on the Computational Grid**. Em: AAAI Spring Symposium on Semantic Web Services: 2004.

[3] Casavant, T. L. e Kuhl, J. G. **A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems** Em: IEEE Transactions on Software Engineering, v.14, n. 2, p. 141-154: 1988.

[4] Dantas, M. **Computação Distribuída de Alto Desempenho**. Ed. Axcel. Rio de Janeiro: 2005.

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

[6] Foster, I. **What is the Grid? A three point checklist**. GRIDToday, Daily News And Information For the Global Grid Community: 2002.

[7] Foster, I.; Tuecke, S. e Unger, J. **OGSA Data Services**. Em: http://www.ggf.org: 2003. Acessado em julho de 2005.

[8] Globus Toolkit. http://www.globus.org. Acessado em fevereiro de 2005.

[9] Nareyek, A. N.; Freuder, E. C.; Fourer, R.; Giunchiglia, E.; Goldman, R. P.; Kautz, H.; Rintanen, J. e Tate, A. **Constraints and AI Planning**. IEEE Intelligent Systems 20(2): 62-72: 2005.

[10] OGSA-DAI. http://www.ogsadai.org.uk/about/ogsa-dqp/. Acessado em fevereiro de 2006.

[11] Oliveira, Celso H. P. **SQL Curso Prático**. Ed. Novatec. São Paulo: 2002.

[12] Ranganathan, Kavitha e Foster, Ian **Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids**. In: Journal of Grid Computing, Volume 1 p. 53-62: 2003.

[13] Watson, P.; Paton, N.; Atkinson, M.; Dialani, V.; Pearson, D. e Storey, T. **Database Access and Integration Services on the Grid**. Em: Fourth Global Grid Forum (GGF-4): 2002. http://www.nesc.ac.uk/technical_papers/dbtf.pdf. Acessado em julho de 2005.

[14] Watson, P. **Database and the Grid**. In: Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons Inc: 2003.