Informática Pública ano 10 (2): 17-36, 2008

GerpavGrid – Utilizando Grades Computacionais no Aprimoramento de uma Ferramenta de Gestão Pública para a Manutenção de Pavimentos

César A. F. De Rose[1]
Tiago C. Ferreto[2]
Luiz Gustavo Fernandes[3]
Walfredo Cirne[4]
Milena Pessoa Micheli[5]
Vladimir Gonçalves Dias[6]
Marcelo Bukowski de Farias[7]
Felipe Afonso de Azevedo[8]

Palavras-Chave

Grades Computacionais, Escalonamento de Tarefas, Bag-of-Tasks, Gerência de Pavimentos, OurGrid, Gerpav

Resumo

O projeto GerpavGrid tem por objetivo o desenvolvimento de uma aplicação de grade para a gerência de pavimentos da cidade de Porto Alegre e a implantação de uma grade OurGrid para o suporte à execução desta aplicação. A aplicação GerpavGrid é baseada no Gerpav (sistema de gerência de pavimentos) que efetua cálculos sobre uma base de dados que contém informações sobre as condições dos logradouros da cidade. Dada a complexidade dos processos da aplicação, a execução de uma versão distribuída no OurGrid permite a realização dos modelos de cálculo já existentes de forma mais rápida e a inclusão de novos modelos bem mais complexos, que não seriam possíveis no sistema original, o que facilita o trabalho do engenheiro de trânsito.

[1] E-mail: cesar.derose@pucrs.br

[2] E-mail: tiago.ferreto@pucrs.br

[3] E-mail: gustavo@inf.pucrs.br

[4] E-mail: walfredo@dsc.ufcg.edu.br

[5] E-mail: milena@ourgrid.org

[6] E-mail: vladimird@dbserver.com.br

[7] E-mail: marcelof@dbserver.com.br

[8] E-mail: felipea@dbserver.com.br

1. Introdução

Computação em grade é a coordenação de grandes conjuntos de recursos para prover uma plataforma de execução para aplicações com uso intensivo de recursos, oferecendo uma solução para a crescente necessidade de capacidade computacional e transparência. Seu objetivo é prover uma infraestrutura que possibilite acesso a recursos distribuídos geograficamente de forma confiável, consistente e persistente, reunindo máquinas de diferentes redes, usando ciclos ociosos e produzindo poder computacional similar àqueles oferecidos por supercomputadores.

Uma funcionalidade importante desta tecnologia é o baixo custo, o que torna a computação em grade interessante no momento da escolha de uma plataforma para execução de aplicações paralelas. Apesar disso, de acordo com [Cir04], existem algumas funcionalidades como alta heterogeneidade, complexidade e ampla distribuição que criam vários desafios tecnológicos que devem ser considerados. O OurGrid é uma solução de grade aberta (*free-to-join*) e completa para computação em grade, oferecendo o *middleware*, as abstrações e os serviços necessários para a criação e uso de uma grade visando à execução de aplicações do tipo *Bag-of-Tasks* (BoT) [Cost04].

Aplicações BoT são aplicações paralelas cujas tarefas são independentes umas das outras. O OurGrid está em produção desde dezembro de 2004. Interessados podem aderir à grade no site http://www.ourgrid.org.

Apesar da sua simplicidade, as aplicações BoT são usadas em uma variedade de cenários, incluindo mineração de dados, pesquisa em massa (como quebra de chaves), troca de parâmetros, simulações, cálculo de fractais, biologia computacional e processamento de imagens. Além disso, devido à independência de suas tarefas, aplicações BoT podem ser executadas com sucesso sobre grades computacionais amplamente distribuídas.

Este trabalho apresenta o projeto GerpavGrid, fruto do Convênio FINEP nº 01.04.0893.00. Sua execução foi coordenada pela Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), por intermédio do Centro de Pesquisa em Alto Desempenho (CPAD) e do Centro de Pesquisa e Desenvolvimento de Aplicações Paralelas (CAP), e pela Universidade Federal de Campina Grande, pelo Laboratório de Sistemas Digitais (LSD). O projeto também contou com o apoio técnico da Hewlett-Packard do Brasil e de desenvolvimento realizado pela DBServer Assessoria em Sistemas de Informação.

O projeto consiste na adaptação de uma aplicação de gerência de pavimentos, denominada Gerpav, para uma grade computacional, utilizando o *middleware* OurGrid. A aplicação Gerpav realiza análises e projeções sobre as condições da malha viária municipal a fim de dar subsídios à tomada de decisão sobre aplicação de recursos. A aplicação foi originalmente desenvolvida pela Companhia de Processamento de Dados do Município de Porto Alegre (Procempa) para a Secretaria Municipal de Obras e Viação (SMOV) deste mesmo município. O *middleware* OurGrid é baseado em um modelo *peer-to-peer*, no qual os laboratórios oferecem seus recursos computacionais ociosos em troca do acesso a recursos computacionais ociosos de outros laboratórios, quando necessário.

2. Grades Computacionais

Grade Computacional é comumente tratada como uma infraestrutura que oferece um conjunto numeroso de recursos computacionais, de maneira transparente, às aplicações

distribuídas. A idéia (bem como o termo, "Grid") surgiu da rede de fornecimento de energia elétrica. Na computação, a "energia" foi representada inicialmente por ciclos ociosos de CPU. Isso aconteceu porque o foco da pesquisa em Grades era dar suporte às aplicações paralelas, fracamente acopladas, com demanda de grande poder computacional.

O conceito evoluiu para uma abordagem mais sofisticada na qual recurso pode se referir a dados e programas compartilhados pela Internet. O alvo passa a ser não só Grades para Processamento de Alto Desempenho, mas também Grades de Dados (Data Grids), que gerenciam e compartilham grandes quantidades de dados distribuídos, e as Grades de Serviços, para fornecer qualquer serviço computacional sob demanda.

Alguns projetos na área, como SETI@home [Seti06], Distributed.net [Dist06] e BOINC [Boin06], destacaram-se mundialmente, e muito investimento vem sendo feito em *middleware* para grades computacionais. De promessa a realidade, hoje podemos encontrar algumas grades em produção, por exemplo, TeraGrid (baseado no Globus Toolkit), LCG (LHC Computing Grid, CERN) e o próprio OurGrid.

Com a disponibilidade de recursos computacionais e a vasta quantidade de informação compartilhada entre cientistas, surge a e-Science, uma nova metodologia de pesquisa baseada em colaboração. Espera-se que as Grades computacionais tornem-se a infraestrutura para e-Science, e-Business, e-Government e e-Life.

3. GERPAV

Boas estradas são essenciais para a movimentação de pessoas e bens, entretanto envolvem custos substanciais em sua construção e manutenção. A cidade de Porto Alegre, no estado do Rio Grande do Sul, tem enfrentado um aumento no tráfego de veículos devido ao crescimento da cidade, o que requer ações rápidas visando à melhoria e manutenção de sua malha viária. A Secretaria Municipal de Obras e Viação (SMOV), responsável pela conservação e avaliação da pavimentação urbana, tem sido desafiada nos últimos anos a lidar com orçamentos limitados, requerendo ainda mais eficiência na aplicação dos recursos públicos.

Nesse contexto, a SMOV decidiu desenvolver um Sistema de Gerenciamento de Pavimentação, que inclui a construção de um sistema de informação para auxílio na tomada de decisão sobre a avaliação e manutenção da pavimentação. Esse sistema, chamado Gerpav, teve sua fase inicial concluída no final de 2004.

O Gerpav auxilia a SMOV na manutenção da informação sobre a malha viária da cidade, como tipos de pavimentos, defeitos, pistas, trechos e intervenções. O sistema possui um módulo de relatórios no qual o usuário pode simular a degradação dos pavimentos, de acordo com um conjunto de parâmetros.

O Gerpav divide a malha viária em arcos, e cada arco representa um segmento de uma rua. Cada arco pode conter até quatros pistas. Cada pista, por sua vez, é dividida em um número de faixas; a faixa contém informações sobre o tipo de pavimento (cimento, asfalto, paralelepípedo etc.) e os levantamentos de campo contabilizam os defeitos encontrados para cada faixa. Baseado nos dados obtidos pelos levantamentos, o sistema calcula um valor para o ICP (Índice de Condição de Pavimento) para cada faixa e um ICP global para a pista. O valor do ICP varia entre 0 e 100, sendo que 100 representa uma condição perfeita.

Cada tipo de pavimento tem comportamento e desempenho diferentes. O Gerpav simula a degradação dos pavimentos, levando em conta o tipo do pavimento, conforme a Figura 1:

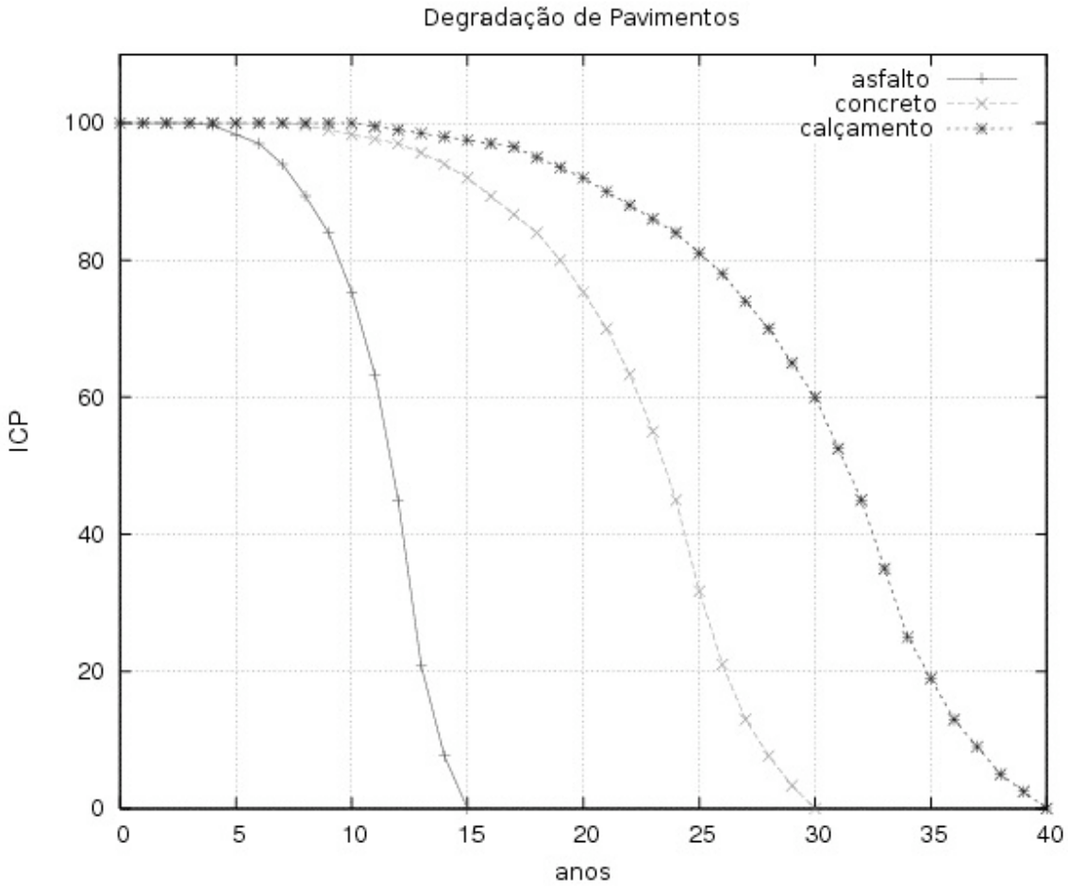

Figura 1 – Curvas de degradação dos pavimentos

A base de dados completa do Gerpav tem 26 tabelas e ocupa em torno de 200 MB de espaço em disco. As tabelas principais são aquelas listadas na Tabela 1:

Tabela 1 – Principais tabelas do Gerpav

Tabela	Número de registros
Arco	31.252
Pista	32.408
Faixa	217.141

3.1. Levantamento dos Subsistemas

A fim de avaliar a concreta possibilidade de paralelização do sistema Gerpav ou de parte dele, seus componentes foram agrupados e classificados em subsistemas conforme suas características de funcionalidade. Tendo essa classificação em mente, o sistema pode ser representado por seis subsistemas (Figura 2), conforme segue:

- Subsistema Genérico de Manutenção de Tabelas: responsável pela manutenção dos dados em tabelas genéricas do sistema, tais como tipos de pavimentos, tipos de defeitos, tráfego, etc.

- Subsistema de Manutenção da Malha Viária: responsável pela manutenção das tabelas relacionadas à malha viária, como ruas, pistas e faixas.
- Subsistema de Relatórios da Malha Viária: responsável pelos relatórios relativos às tabelas da malha viária.
- Subsistema de Cálculo de ICP: responsável pelo cálculo do ICP, de acordo com o tipo de pavimentação.
- Subsistema de Importação de Dados: responsável pela importação dos dados de arquivos externos gerados por aplicações externas.
- Subsistema de Simulação: responsável por simulações relacionadas com avaliação de pavimentos e planejamento de manutenção.

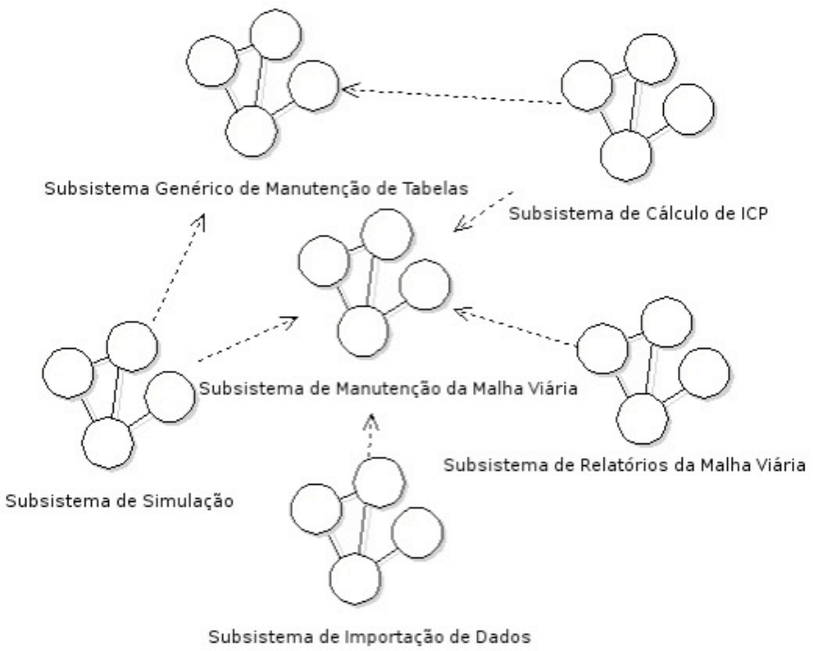

Figura 2 – Subsistemas do Gerpav

A partir deste levantamento, e tendo em vista que, para que se possa obter um melhor aproveitamento da computação em grade, os algoritmos e equações devem ser projetados de forma que possam ser decompostos e seus resultados agregados posteriormente, o Subsistema de Simulação do Gerpav foi identificado como o melhor candidato para a computação em grade. Apesar de os algoritmos deste subsistema necessitarem intensivamente de acesso a dados e memória, eles podem ser reprojetados de forma que se tenha pouca comunicação entre as tarefas e assim se tornarem altamente paralelizáveis.

3.2 Subsistema de Simulação

Dentro do Subsistema de Simulação do Gerpav, existem alguns relatórios que o usuário pode executar; um deles é o "Relatório de Comportamento de Pavimentos" que simula a degradação do pavimento ao longo do tempo. Para criar a versão de grade, o "Relatório de Comportamento de Pavimentos" foi o primeiro eleito para uma análise mais detalhada, como veremos nas próximas seções.

3.3 Relatório de Comportamento de Pavimentos

No "Relatório de Comportamento de Pavimentos", o comportamento do pavimento é calculado para cada pista, de acordo com o tipo de pavimento predominante, seguindo a curva de degradação. O diagrama de atividades na Figura 3 mostra como a simulação funciona. Inicialmente são computados os trechos. Em seguida, as pistas são obtidas para cada trecho e depois as faixas são obtidas para cada pista. O algoritmo, então, computa o pavimento predominante baseado na informação das faixas e então calcula a degradação para a pista.

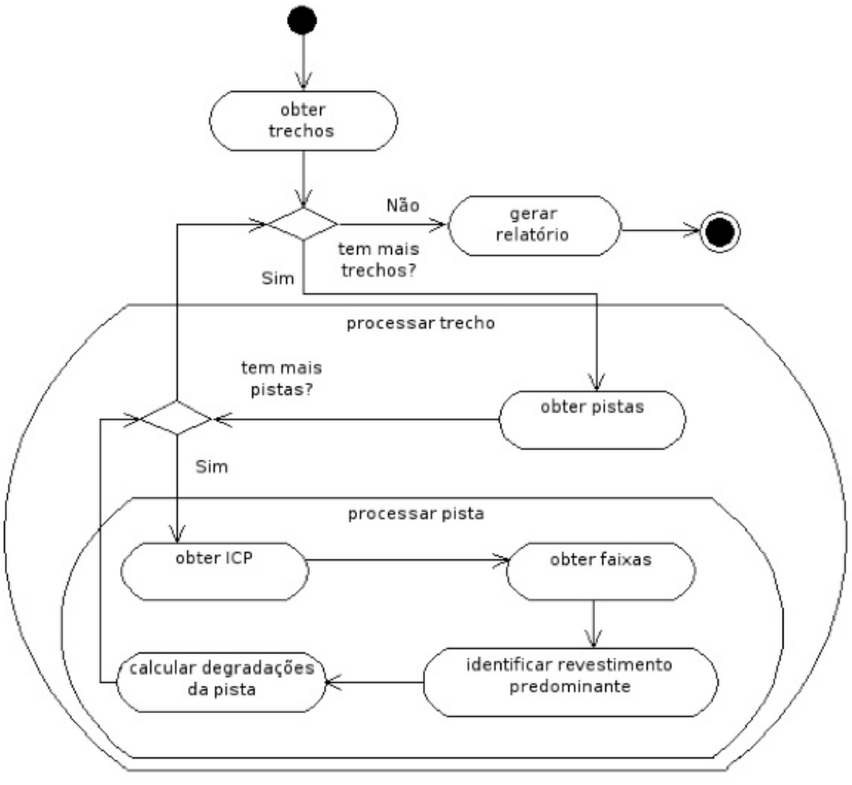

Figura 3 – Relatório de Comportamento dos Pavimentos

3.4 Análise do Relatório de Comportamento de Pavimentos

O "Relatório de Comportamento de Pavimentos" foi escolhido para ser testado detalhadamente. O cenário utilizado foi a simulação do comportamento dos pavimentos para os próximos cinco anos (2005-2010) de um bairro, Aberta dos Morros, um bairro de porte médio de Porto Alegre com 897 arcos de rodovia.

As ferramentas utilizadas pelo ambiente de testes incluem o Jakarta JMeter [Jaka05], usado para automatizar a geração de requisições para o servidor Web, e o JProbe Profiler [JPro06], usado para análise da aplicação e para localização de gargalos. Foi criado um script para JMeter para que este gerasse as requisições para execução do cenário desejado. O JProbe Profiler foi configurado para iniciar o servidor Web Jakarta Tomcat [Tomc06] e coletar as informações de desempenho a partir do código fonte do Relatório de Comportamento de Pavimentos em nível de métodos.

O Quadro 1 mostra a configuração das máquinas utilizadas para os testes.

Quadro 1 – Ambiente de testes do Gerpav

Máquinas	Zambinos	Saligna
Hardware	Intel Pentium 4 2.8GHz 512 MB de RAM	Intel Pentium 4 1.8GHz 384 MB de RAM
Software	Slackware Linux Apache JMeter 2.0.2 JProbe Profiler 5.2.1 Jakarta Tomcat 5.0.28 Sun J2SE 1.4.2_08	SuSE Linux 9.2 IBM DB2 UDB 8.2

Foi coletada a duração de tempo de execução por cada método do código-fonte do sistema. Para facilitar a análise, os métodos foram agrupados de acordo com suas principais funções. O Quadro 2 mostra os grupos de métodos e suas descrições.

Quadro 2 – Grupos de métodos do Gerpav

Grupo	Descrição
fetch	Métodos relacionados à recuperação de linhas de conjuntos de resultados do banco de dados.
getprop	Métodos relacionados à recuperação de propriedades a partir de atributos de linhas de conjuntos de resultados.
query	Métodos relacionados à consulta ao banco de dados.
dataproc	Métodos relacionados ao processamentos de dados.
report	Métodos relacionados à geração da saída.
system	Métodos de sistema.
iterate	Métodos iteradores de coleções.
logic	Métodos relacionados a testes condicionais.
framework	Métodos utilizados pelo framework de persistência.
updatedb	Métodos relacionados à atualização do banco de dados
other	Outros métodos.

A Tabela 2 mostra os tempos de execução acumulados, em milissegundos, e o número de chamadas dos métodos dentro de cada grupo. A tabela está ordenada em ordem decrescente pelo tempo de execução. Note que os métodos relacionados ao acesso (fetch, getprop e query) ao Sistema de Gerência de Banco de Dados (SGBD) tomam mais de 95% do tempo de processamento.

Tabela 2 – Tempo de execução e número de chamadas por grupo de métodos

Grupo	Chamadas	Tempo
fetch	18.160.106	816.936 ms
getprop	522.050.012	795.121 ms
query	8.778	722.889 ms
dataproc	1.091.806	8.235 ms
report	4	6.869 ms
system	596.255	1.955 ms
iterate	964.795	1.526 ms
logic	234.803	919 ms
framework	8.794	547 ms
updatedb	1	203 ms
other	332.192	3.755 ms
total	543.447.546	2.358.955 ms

Os resultados dos testes mostram que o desempenho do Gerpav é amarrado ao SGBD. Isso acontece devido à forma como a aplicação foi projetada, com alto acoplamento às tabelas do banco de dados. Assim, o tempo de execução da aplicação está diretamente ligado ao tempo de resposta do SGBD.

Ainda, o tempo de execução total de 2.358.955 ms (aproximadamente 39 minutos), para simular o comportamento dos pavimentos para um bairro com 897 arcos, mostra que seria praticamente impossível rodar uma simulação para toda a cidade (31.252 arcos) usando a abordagem atual. Em função dessa limitação, os usuários do Gerpav foram orientados a limitar ao máximo o volume de dados em suas simulações, o que, consequentemente, subutiliza o conceito original do Subsistema de Simulação, além de obviamente não suprir integralmente as expectativas criadas em relação ao sistema.

4 GerpavGrid

O GerpavGrid nasceu da idéia de utilizar a infraestrutura do OurGrid para resolver os problemas do Subsistema de Simulação do Gerpav. Agregar o OurGrid ao Gerpav pode tornar possível atingir não apenas a melhoria de desempenho, mas ainda agregar novas funções que não seriam possíveis no projeto anterior.

Dessa forma, a proposta do GerpavGrid é ser uma extensão do Gerpav, adicionando novas funcionalidades e reprojetando algumas das simulações no módulo de simulação para utilizar a infraestrutura do OurGrid, visando obter melhoria de desempenho.

4.1 Estratégias de Implementação

Enquanto o Gerpav é altamente acoplado ao SGBD e projetado para acessar uma base de dados sempre disponível, o GerpavGrid, para que possa obter as vantagens do processamento paralelo no OurGrid, não pode contar com a disponibilidade do SGBD da

mesma forma como no Gerpav. Além disso, esforço adicional é necessário para adaptar a aplicação para a abordagem de Bag-of-Tasks usada pelo OurGrid.

Consequentemente, algumas partes do código do Gerpav tiveram de ser reescritas para lidar com as restrições impostas. As próximas subseções irão explicar as estratégias adotadas pelo projeto GerpavGrid, incluindo a técnica de reestruturação de código conhecida como refactoring, o framework OJAL, desenvolvido para abstrair o MyGrid, as técnicas de filtragem em banco e em memória, além das técnicas de paralelização *Simple Slicing*, *Pipeline Dispatching* e *Distributed Database*.

O MyGrid [MyGr04] é o componente de escalonamento da solução OurGrid. É uma API (Application Programming Interface) que serve de camada de abstração do OurGrid, agindo como frontend para o usuário.

4.1.1 Técnica de Refatoração de Código – Refactoring

De acordo com [Fowl97], *refactoring* é uma técnica disciplinada para a reestruturação de um código existente, alterando sua estrutura interna sem mudar seu comportamento externo.

Visando mudar o algoritmo do Gerpav para ser paralelo, o Relatório de Comportamento de Pavimentos foi "refatorado" e dividido em três fases distintas (Figura 4):

- Preparação de dados: todos os dados necessários ao processamento são obtidos do banco de dados no início, isso torna possível a implementação dos próximos passos, independentemente do acesso aos dados;
- Processamento: tendo todos os dados em estruturas de memória, o processamento pode ser separado do algoritmo, esta é a parte que será paralela;
- Consolidação dos Resultados: depois que o processamento é feito, as estruturas de retorno são agregadas para gerar a saída para o usuário.

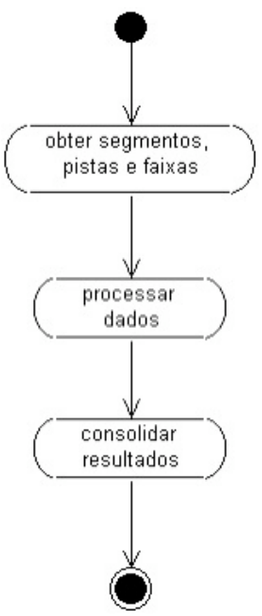

Figura 4 – Fluxo do Relatório de Comportamento de Pavimentos após o *Refactoring*

4.1.2 Camada de Abstração - OJAL (OurGrid Job Abstraction Layer)

Uma vez que utilizamos a infraestrutura do OurGrid, para submeter um job para execução, um arquivo de descrição do job (JDF – Job Description File) deve ser criado. Esse JDF contém detalhes sobre cada tarefa que irá executar uma parte do job. Se um determinado job tem centenas de tarefas, escrever um JDF pode se tornar entediante e consumir um tempo considerável. Por isso, alguns usuários preferem escrever um programa em Java ou um *script* customizado que submete o job para a execução, evitando a criação manual do arquivo. Assim, o MyGrid provê uma API que facilita esse processo [MyGr04].

Ainda assim, a API provida pelo MyGrid não é tão fácil de utilizar em um sistema baseado em web como o Gerpav. Dessa forma, para usar a API do MyGrid no GerpavGrid, uma camada extra de abstração foi criada. Esta camada foi chamada de "OurGrid Job Abstraction Layer" (OJAL). Ela auxilia na execução de operações relacionadas à serialização e desserialização de dados, divisão dos dados e submissão e espera dos jobs [FaDi05].

A Figura 5 representa uma visão de alto nível do OJAL; classes com o prefixo Concrete devem ser criadas pela aplicação. Um ConcreteClient pode instanciar um objeto JobDispatcher, passar alguns parâmetros, como tamanho de fatias, arquivos *jar* e classe remota e chamar um dos métodos de submissão, passando uma coleção com dados a ser processados na grade. Na parte remota, uma ConcreteTask deve estender RemoteTask e conter os métodos main e process.

O método de submissão do OJAL (dispatch) vai fatiar a coleção de dados e criar o JDF para o MyGrid para transferir a ConcreteTask e os arquivos relacionados para a infraestrutura da grade. Depois da execução do job, a aplicação irá obter a coleção de retorno e continuar seu processamento (Figura 6).

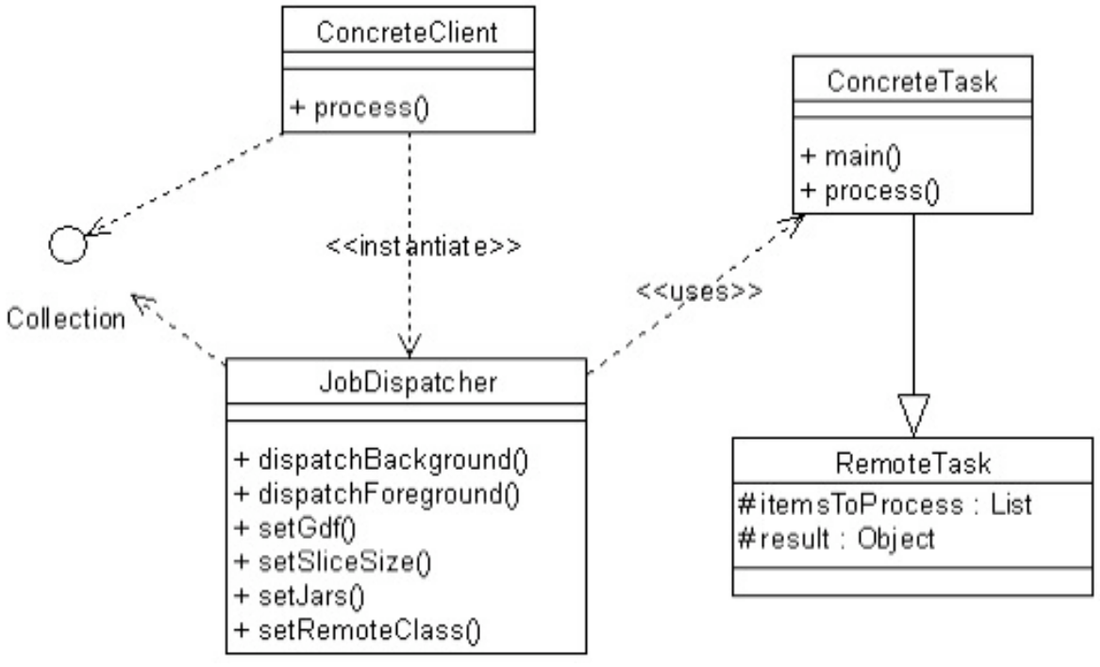

Figura 5 – Diagrama de classes do OJAL

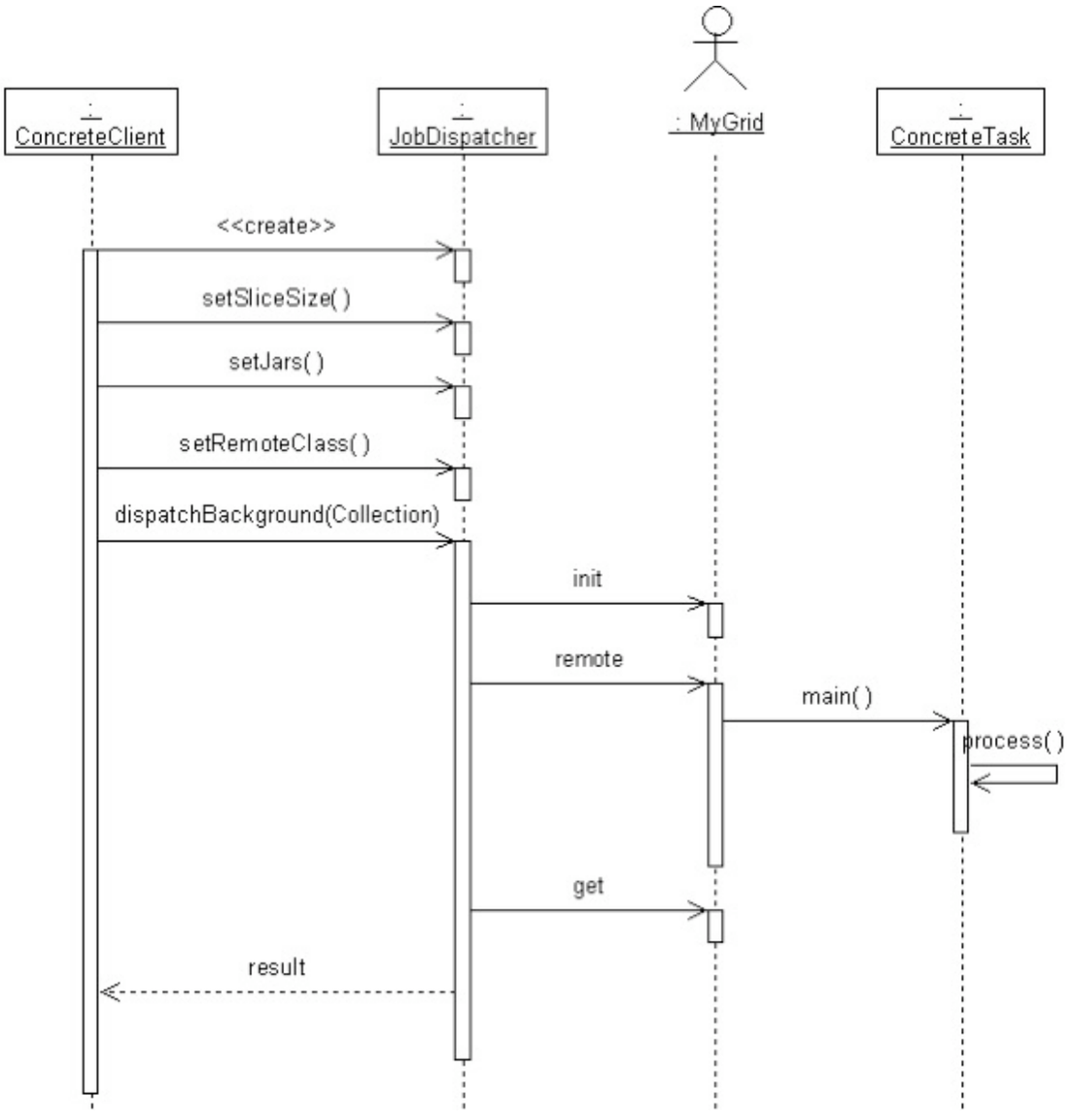

Figura 6 – Diagrama de sequência do OJAL

4.1.3 Técnica de Filtragem: Database Filter versus Memory Filter

Utilizando a estratégia Database Filter, o sistema obtém os dados das pistas do banco de dados, usando uma pesquisa SQL (*Structured Query Language*) complexa para filtrar as linhas com os critérios de seleção, visando obter somente os dados necessários. O objetivo dessa estratégia é reduzir o tamanho dos dados que serão enviados à grade.

Por outro lado, quando utilizando a estratégia de Memory Filter, o sistema obtém os dados das pistas do banco de dados, utilizando uma pesquisa crua, com critério de seleção o mais simples possível, deixando a seleção das pistas corretas para ser feita programaticamente; dessa forma, o trabalho pode ser feito também pelas tarefas remotas. Assim, o objetivo agora é reduzir o peso do trabalho feito pelo SGBD.

4.1.4 Técnica de Paralelização Simple Slicing

O sistema obtém os dados das pistas do banco de dados usando a estratégia Database Filter. As pistas obtidas são divididas em fatias com um número fixo de elementos. Cada fatia é serializada e anexada a uma tarefa remota; um job é submetido para execução remota das tarefas pelo OurGrid.

Depois da execução do job, a coleção de retorno é consolidada e enviada à saída. A Figura 7 ilustra este processo:

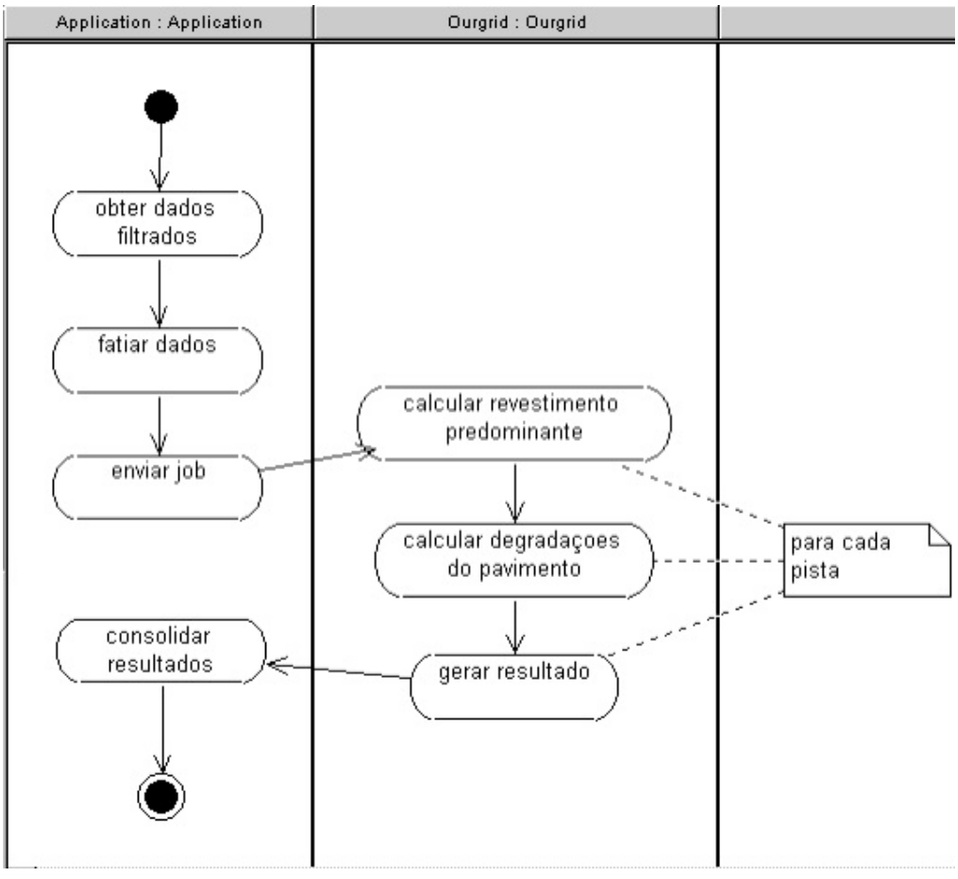

Figura 7 – Diagrama de estados da técnida de paralelização *Simple Slicing*

4.1.5 Técnica de Paralelização Pipeline Dispatching

O sistema obtém os dados das pistas do banco de dados, usando a estratégia Database Filter ou Memory Filter. As pistas obtidas para um bairro são divididas em fatias com um número fixo de elementos. Cada fatia é serializada e anexada a uma tarefa remota; um job é submetido para execução remota das tarefas pelo OurGrid. O sistema continua obtendo dados de pistas para o próximo bairro, sem esperar o final da execução do job anterior.

Depois da submissão do job para o último bairro, o sistema espera pelo término de todos os jobs. As coleções de resultados (uma para cada bairro) são então unidas, e o resultado é consolidado e enviado à saída. A Figura. 8 ilustra as atividades para o *Pipeline Dispatching* com *Memory Filter*.

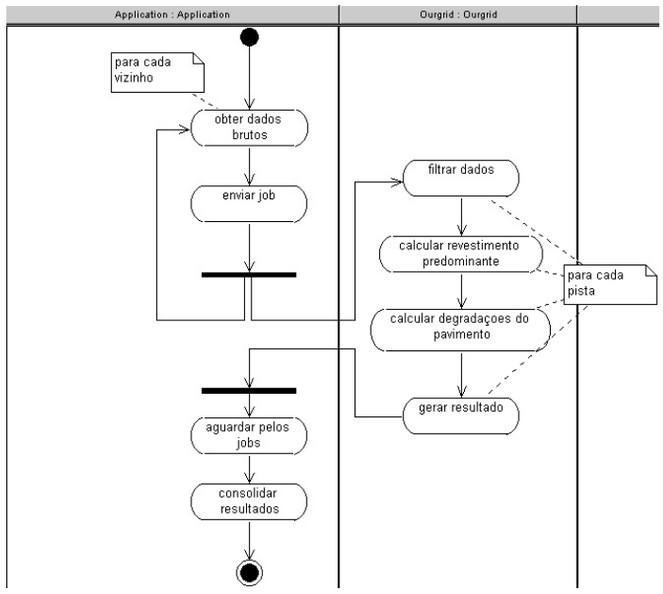

Figura 8 – Diagrama de estados da técnica de paralelização *Pipeline Dispatching*

4.1.6 Técnica de Paralelização Distributed Database

Os dados dos arcos, pistas e faixas são enviados previamente para bancos de dados hospedados pelos *peers* do OurGrid. O sistema cria uma tarefa para cada bairro; um job é submetido para execução remota das tarefas pelo OurGrid.

A tarefa remota obtém os dados das pistas acessando o banco de dados do seu peer e filtra os dados pelos critérios de seleção, utilizando a estratégia de *Memory Filter*. O sistema espera pela execução do job. Em seguida, obtém a coleção de saída para consolidar os resultados e enviá-los ao usuário. A Figura 9 ilustra este processo:

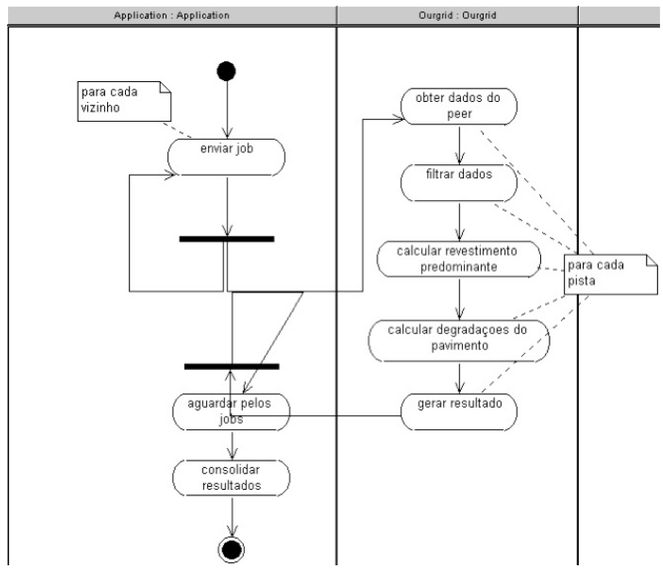

Figura 9 – Diagrama de estados da técnica de paralelização *Distributed Database*

4.2 Análise de Desempenho

Para os testes, um servidor chamado Pinus foi usado como servidor web e como escalonador do MyGrid. O Quadro 3 mostra as configurações do servidor pinus.

Quadro 3 – Configuração do Servidor Web

Máquina	Servidor Web
Hardware	Dual Xeon 3.0 GHz 2 GBytes de RAM
Software	SuSE Linux 9.2 Jakarta Tomcat 5.0.28 Sun J2SE 1.4.2_08 IBM DB2 UDB 8.2

Como dito anteriormente, o middleware para grade computacional utilizado no GerpavGrid foi o OurGrid. O OurGrid é uma solução *peer-to-peer*, de livre distribuição e de código aberto. Uma grade OurGrid é composta por três tipos de máquina: Grid Machine, Peer e Home Machine. A Grid Machine (GuM) é a máquina que executa uma tarefa submetida à grade, por meio do componente SWAN. O Peer, conhecido como GuM provider, é o elemento que gerencia os recursos ociosos (GuM disponíveis) em seu domínio administradtivo. A Home Machine é a máquina do usuário, onde o MyGrid é executado. O MyGrid é o módulo do sistema que solicita recursos (GuMs) ao Peer e faz o escalonamento dos recursos entre as tarefas da aplicação. Os Peers estão interconectados entre si (via Internet, por exemplo) e trocam recursos ociosos baseado num mecanismo de alocação chamado Network of Favors (NoF) [Cir05]. O componente SWAN é uma sandbox usada para garantir a segurança no acesso às GuMs. As tarefas remotas executam dentro de uma máquina virtual Xen (dentro da GuM), de maneira que apenas o acesso a essa máquina virtual é concedido pelo SWAN.

A Figura 10 mostra um exemplo de grade OurGrid.

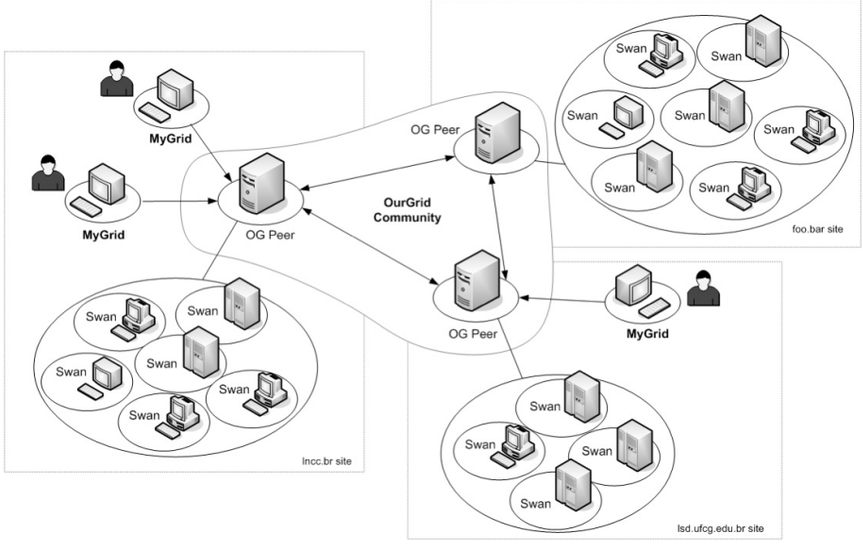

Figura 10 – A solução OurGrid [OuGr2004]

4.3 Implementações e Cenários

Para execução dos testes, algumas das estratégias foram combinadas em implementações e então alguns cenários de teste foram planejados para observar o comportamento das implementações.

A Tabela 3 descreve as implementações, e a Tabela 4 descreve os cenários criados para os testes.

Em todos os cenários, o Relatório de Comportamento de Pavimentos foi executado para calcular a degradação dos pavimentos para os próximos 15 anos.

Tabela 3 – Estratégias de implementação

Implementação	Estratégias	Tipo
original	Gerpav, original	Sequencial
I1	*Refactoring*	Sequencial
I2	*Refactoring* + OJAL + *Simple Slicing* + *Database Filter*	Paralela
I3	*Refactoring* + OJAL + *Pipeline Dispatching* + *Database Filter*	Paralela
I4	*Refactoring* + OJAL + *Pipeline Dispatching* + *Memory Filter*	Paralela
I5	*Refactoring* + OJAL + *Distributed Database* + *Memory Filter*	Paralela

Tabela 4 – Cenários de teste

Cenário	Nome	Arcos	Pistas	Faixas
S1	bairro aberta dos morros	897	929	2.094
S2	50 bairros	15.070	15.816	49.195
S3	toda a malha viária (93 bairros)	31.252	32.408	98.223

4.4 Resultados

A Tabela 5 mostra os tempos de execução coletados para cada implementação em cada cenário. Os dados de tempo são informados em milissegundos.

Tabela 5 – Tempos de execução (implementações x cenários)

Implementação	Cenário S1	Cenário S2	Cenário S3
I1	200.287 ms	3.551.650 ms	7.266.923 ms
I2	207.818 ms	3.370.382 ms	7.430.041 ms
I3	210.408 ms	3.489.814 ms	7.113.705 ms
I4	32.060 ms	275.864 ms	506.271 ms
I5	12.312 ms	50.327 ms	89.877 ms

Note que os tempos obtidos por I4 e I5 são significativamente menores do que os de I1, I2 e I3 (Figuras 11, 12 e 13). Claramente, as implementações que utilizam *Memory Filter* se comportam melhor do que aquelas que usam *Database Filter*.

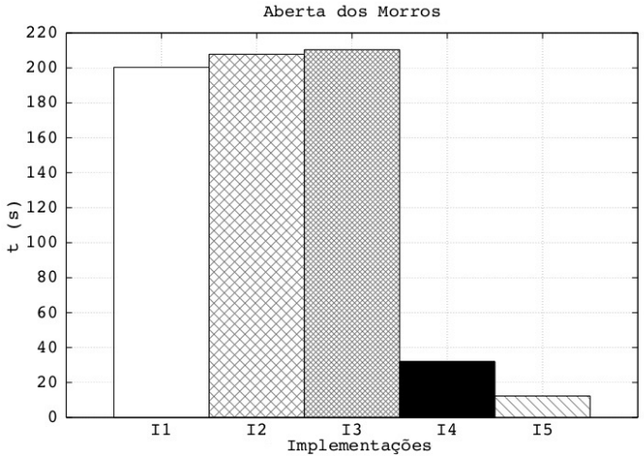

Figura 11 – Tempos de execução no cenário S1 (1 bairro)

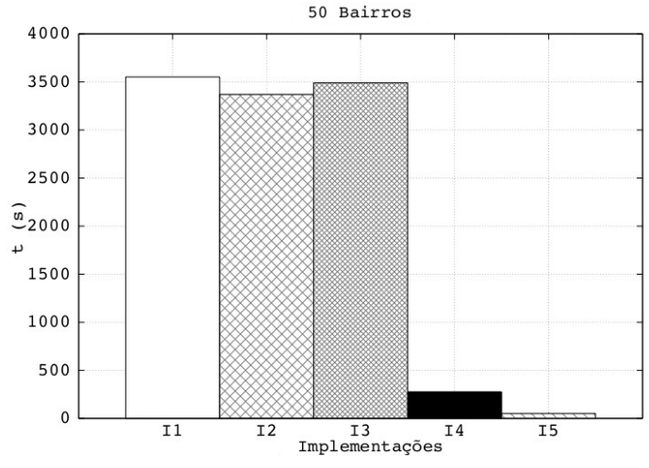

Figura 12 – Tempos de execução no cenário S2 (50 bairros)

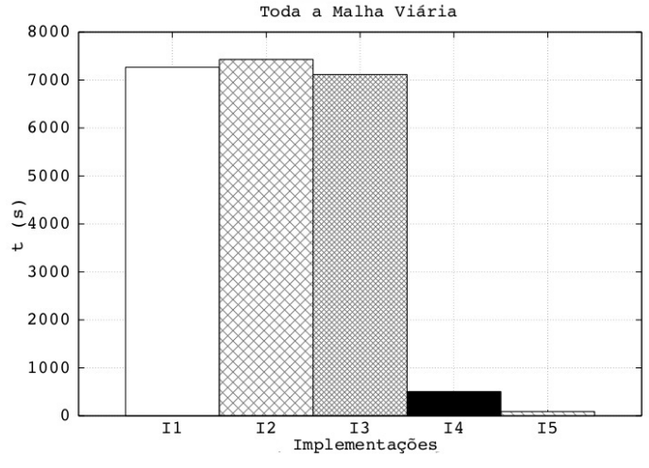

Figura 13 – Tempos de execução no cenário S3
(toda a malha viária - 93 bairros)

Na realidade, os tempos obtidos por I1, I2 e I3 são bastante similares (Figura 14). Por outro lado, I5 é melhor do que I4, e a diferença de desempenho também cresce de acordo com o número de arcos sendo processados (Figura 15).

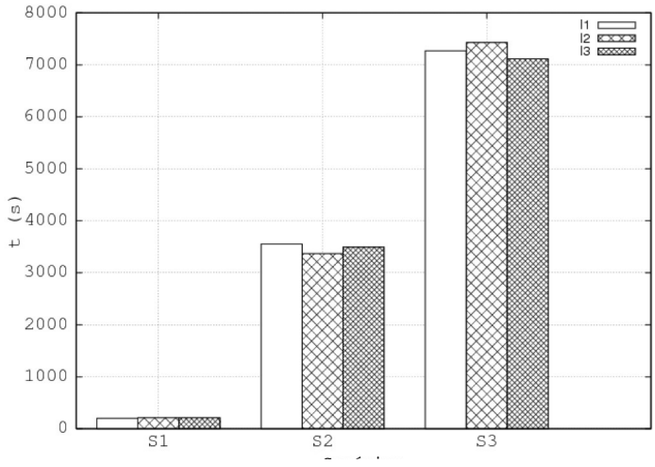

Figura 14 – Comparação das implementações I1, I2 e I3

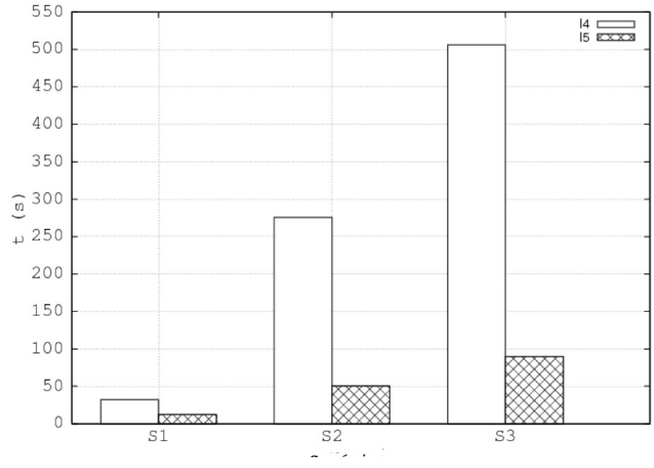

Figura 15 – Comparação das implementações I4 e I5

5 *Conclusão*

A aplicação Gerpav foi proposta para orientar os investimentos em conservação e manutenção dos pavimentos das vias públicas da cidade de Porto Alegre, visando à otimização de aplicação dos recursos disponíveis. Ela foi desenvolvida junto à Cia. de Processamento de Dados do Município de Porto Alegre (PROCEMPA), permitindo à Secretaria Municipal de Obras e Viação (SMOV) cadastrar e manter atualizadas as informações referentes às características das vias. Seu objetivo principal é fornecer as informações técnicas necessárias à tomada de decisão, permitindo uma inter-relação adequada e segura entre os aspectos técnicos e políticos, buscando garantir a manutenção do patrimônio público representado pelas vias.

O sistema GerpavGrid é uma adaptação da aplicação Gerpav para execução paralela dos módulos na grade, que consiste na realização de *refactoring* do código, criação de uma

grade OurGrid para o projeto (envolvendo o LSD/UFCG, o CPAD/PUCRS e a SMOV), elaboração de mecanismos de distribuição dos dados na grade e testes da aplicação.

Essa adaptação do sistema original, além do visível ganho de desempenho – de aproximadamente 39 minutos para 12 segundos na simulação de um bairro –, viabiliza a implementação de novos módulos, os quais já estão em fase final de construção. Esses módulos compreendem simulações que permitem estimar o investimento anual necessário para a manutenção da malha viária durante um período de até dez anos, ou ainda avaliar a melhor maneira de distribuir o orçamento disponível ao longo dos anos, visando à otimização da aplicação dos recursos públicos nas obras de manutenção da malha viária. Tais simulações demandam a avaliação de todas as combinações de intervenções possíveis em cada pista, o que representa um esforço computacional de uma grandeza inviável para implementações tradicionais (não-distribuídas); por outro lado, caracterizam um típico problema cuja abordagem de processamento distribuído torna a implementação possível, além de trazer significativos ganhos de desempenho.

Portanto, o fruto dos esforços envolvidos no decorrer das diversas etapas deste projeto é visível e extremamente animador, especialmente por mostrar a viabilidade da utilização de Grades Computacionais em sistemas de gestão pública, a fim de proporcionar não só significativos ganhos de desempenho, mas também novas possibilidades de sistemas antes inviáveis devido à escassez de recursos computacionais nos ambientes tradicionais (não-distribuídos). Adicionalmente, o framework OJAL, disponibilizado livremente como subproduto deste projeto, mostra grande potencial de utilização em diversos cenários de sistemas passíveis de distribuição, devido às características de adaptabilidade com as quais foi concebido, caracterizando uma contribuição imediata à comunidade de desenvolvedores de aplicações distribuídas que optarem pela arquitetura do OurGrid.

Keywords

Grid Computing, Task Scheduling, Bag-of-Tasks, Road Pavement Management, OurGrid, Gerpav

Abstract

The GerpavGrid project investigates the utilization of computational grids in the management of road pavements of the city of Porto Alegre. After the implementation of a distributed version of the management application, an OurGrid federation was formed to support the execution of this application. The distributed GerpavGrid application is based on Gerpav (road pavements management system) that makes calculations over a database containing information on the conditions of the public streets of the city. Given the complexity of the centralized application, running a distributed version on OurGrid allows the accomplishment of the existing models of calculation in a faster way and the addition of new complex models that could not be added in the original system, making the transit engineer's job easier.

Referências Bibliográficas

[BeFH03] BERMAN, F., FOX, G., HEY, T. (Ed.). Grid Computing: making The Global Infrastructure a reality. Hoboken: John Wiley & Sons, 2003.

[Cirn04] CIRNE, W. et al. Scheduling in Bag-of-Task Grids: The PAUÁ Case. In: Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'2004).

[Cost04] COSTA, L. B. et al. MyGrid: A complete solution for running Bag-of-Tasks Applications. In: Proceedings of the SBRC 2004 - Salão de Ferramentas (22nd Brazilian Symposium on Computer Networks - III Special Tools Session).

[Napa04] NAPA, 2004. Disponível em: http://www.hotmix.org/allaboutasphalt.php. Acesso em: 24/02/2006.

[BoGH02] BODOFF, S., GREEN, D., HAASE, K., JENDROCK, E., PAWLAN, M., STEARNS, B. The J2EE Tutorial. Copyright 2002, Addison-Wesley. Also available online at: http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html. Acesso em: 24/02/2006.

[AlCM01] ALUR, D., CRUPI, J., MALKS, D. Core J2EE Patterns: best practices and design strategies. Upper Saddle River: Prentice-Hall PTR, 2001.

[Fowl97] FOWLER, M. UML distilled: applying the standard object modeling language. Reading, MA: Indianapolis: Addison-Wesley, c1997. 183 p.

[Cirn05] CIRNE, W. et al. Labs of the world, unite!!! Disponível em: http://www.ourgrid.org/twiki-public/pub/OG/OurPublications/Labs_of_the_World_Unite_v12.pdf. 2005. Acesso em: 24/02/2006.

[OuGr04] OurGrid. Disponível em http://www.ourgrid.org. Acesso em 24/02/2006.

[MyGr04] MyGrid API. Disponível em: http://www.ourgrid.org/twiki-public/bin/view/OG/ManualDesenvolvedor#High_Level_Architecture. Acesso em: 24/02/2006.

[FaDi05] FARIAS, M., DIAS, V. OurGrid Job Abstraction Layer. Disponível em: http://www.ourgrid.org/twiki-public/pub/OG/OurPortalsAndApplications/cpad-mygrid.zip. Acesso em: 24/02/2006.

[Jaka05] JAKARTA Jmeter. Disponível em: http://jakarta.apache.org/jmeter. Acesso em: 02/03/2006.

[Jpro06] JPROBE PROFILER. Disponível em: http://www.quest.com/jprobe. Acesso em: 02/03/2006.

[Tomc06] JAKARTA TOMCAT. Disponível em: http://tomcat.apache.org. Acesso em: 02/03/2006.

[Seti06] SETI@home web site. Disponível em: http://setiathome.ssl.berkeley.edu/. Acesso em: 02/03/2006.

[Dist06] DISTRIBUTED.net: Node Zero. Disponível em: http://www.distributed.net. Acesso em: 02/03/2006.

[Boin06] BOINC. Disponível em: http://boinc.berkeley.edu/. Acesso em: 02/03/2006.

Sobre os Autores

César A. F. De Rose

Doutor em Ciência da Computação pela Universidade de Karlsruhe, Alemanha, nas áreas de Sistemas Operacionais e Processamento Paralelo e Distribuído; Professor da Faculdade de Informática da PUCRS e do Programa de Pós-Graduação em Ciência da Computação da PUCRS; areas de interesse: processamento paralelo e distribuído, grades computacionais.

Tiago C. Ferreto

Mestre em Ciência da Computação pela PUCRS; Professor da Faculdade de Informática da PUCRS; areas de interesse: processamento paralelo e distribuído, grades computacionais.

Luiz Gustavo Fernandes

Doutor em Informática pelo Institut National Polytechnique de Grenoble (INPG), França, 2002, nas áreas de Sistemas Distribuídos e Modelagem de Aplicações de Alto Desempenho; Professor da Faculdade de Informática da PUCRS e do Programa de Pós-Graduação em Ciência da Computação da PUCRS; áreas de interesse: aplicações paralelas, algoritmos paralelos, avaliação de desempenho, computação de alto desempenho, processamento de imagens.

Walfredo Cirne

Ph.D. em Ciência da Computação pela University of California San Diego (UCSD), nos Estados Unidos; Professor junto ao Departamento de Sistemas e Computação (DSC) da Universidade Federal da Campina Grande (UFCG); áreas de interesse: processamento paralelo e distribuído, grades computacionais, processamento de imagens.

Milena Pessoa Micheli

Mestre em Informática pela Universidade Federal da Campina Grande (UFCG); Assistente de pesquisa do Laboratório de Sistemas Distribuídos (LSD) do Departamento de Sistemas e Computação (DSC) da UFCG; áreas de interesse: processamento paralelo e distribuído, grades computacionais.

Vladimir Gonçalves Dias

Bacharel em Informática pela PUCRS; Arquiteto de Software da DBServer Assessoria em Sistemas de Informação; áreas de interesse: computação distribuída, computação gráfica, arquitetura de sistemas, sistemas web, Java/J2EE.

Marcelo Bukowski de Farias

Graduando em Ciência da Computação pela UFRGS; Arquiteto de Software da DBServer Assessoria em Sistemas de Informação; áreas de interesse: computação distribuída, engenharia de software, arquitetura de sistemas, sistemas web, Java/J2EE.

Felipe Afonso de Azevedo

Bacharel em Ciência da Computação pela PUCRS; Arquiteto de Software da DBServer Assessoria em Sistemas de Informação; áreas de interesse: desenvolvimento de jogos, computação gráfica, realidade virtual, arquitetura de sistemas, sistemas web, Java/J2EE, C++.