GerpavGrid: using the Grid to maintain the city road system *

César A. F. De Rose Tiago C. Ferreto
PUCRS[1]
{derose,ferreto}@inf.pucrs.br

Marcelo B. de Farias Vladimir G. Dias
DBServer[2]
{marcelof,vladimird}@dbserver.com.br

Walfredo Cirne Milena P. M. Oliveira
UFCG[3]
walfredo@dsc.ufcg.edu.br, milena@lsd.ufcg.edu.br

Katia Saikoski
HP Brazil[4]
katia.saikoski@hp.com

Maria Luiza Danieleski
SMOV[5]
danieleski@smov.prefpoa.com.br

Abstract

This paper presents and evaluates a governmental application that has been ported to run in a Grid. The Gerpav application is used in the city of Porto Alegre, located in the south of Brazil, to maintain and plan the investments in the city road system, and Grid technology brought substantial performance gains to its distributed version called GerpavGrid. We describe several optimizations strategies used in GerpavGrid to move the bottleneck of the sequential application from database to memory to facilitate distribution of tasks in the Grid. Results of its execution in a real Grid are also presented, and show that a Grid can be also an interesting execution platform for non-classical HPC applications that are database intensive.

1. Introduction

Grid computing is the coordination of large collections of resources to provide a platform for the execution of resource-intensive applications. Grid offers a solution to the growing need of computational capacity and transparency. Its goal is to provide an infrastructure that grants access to geographically distributed resources in a reliable, consistent and persistent way. The rationale is that congregating machines from different sites would provide computational power similar to those offered by supercomputers.

An important feature of this technology is its low cost, which makes grid computing an interesting choice when it comes to choosing a platform to run parallel applications. Nevertheless, there are some characteristics such as high heterogeneity, complexity and wide distribution that create many new technical challenges that must be considered [7]. Moreover, grid computing is a new technology and, as such, has not yet achieved the level of maturity and dissemination of more traditional technologies. As a result, there are yet few examples of successful grid use, especially outside the large and cutting-edge supercomputer centers.

This paper presents and evaluates a governmental application that has been successfully ported to the grid. The application is used by the city of Porto Alegre (located in south Brazil) to plan road maintenance. Grid technology brought substantial performance gains to the application, resulting in an augmented capacity of the city to plan road maintenance activities, finally leading to better results in road conservation spending less public money. Moreover, employing grid technology, such an advance was made possible requiring minimal additional investment in IT infrastructure.

We consider such a result a milestone in the dissemination of grid technology. To the best of our knowledge, this is the first application of grid technology running in production at the city government. This shows the increasing maturity of grid technology as (i) governments are not the original users of grid technologies (except for their research labs), and (ii) cities do not typically have the resources and expertise of state and federal governments.

The remainder of this paper is organized as follows. We begin in Section 2 describing some related work with focus

*This research was done in cooperation with HP-Brazil.

[1]Pontifícia Universidade Católica do Rio Grande do Sul, Faculdade de Informática, Porto Alegre, Brazil

[2]DBServer Assessoria em Sistemas de Informação Ltda, Porto Alegre, Brazil

[3]Universidade Federal de Campina Grande, Campina Grande, Brazil

[4]Hewlett Packard Computadores Ltda, Porto Alegre, Brazil

[5]Secretaria Municipal de Obras e Viação, Porto Alegre, Brazil

on e-government applications and grid computing. In Section 3 we present OurGrid, Gerpav, and the motivation to implement some of Gerpav modules in the Grid. In Section 4 we present the details about GerpavGrid development. In Section 5 we present a performance evaluation of GerpavGrid using the grid of the OurGrid community. Finally, we summarize our conclusions in Section 6.

2. Related Work

Grid computing is a very active area of research [5, 9]. Although it started within High Performance Computing, people have realized that Grid technology could be used to deliver computational services on demand. This observation has brought about convergence between Grid and Web Services technologies, as seen in standards like OGSA/OGSI [19] and its successor WSMF [8]. The vision is that grid technology will eventually turn any computing demand into the on-demand access of computation services.

Another very active area has been e-government, which addresses how governments can make better use of IT technology to be more transparent, accessible, and efficient. We believe that grids and e-government have great synergy, as many e-government applications must deal with very large volumes of data and/or evaluate a multitude of scenarios for strategic planning. However, as both areas are relatively new, most efforts still focus on infrastructure, e.g. [17, 11]. Examples of grid applications in the government area are not yet common. We know of: The British project that aims to leverage from grid technology to create a better and richer learning experience for the country students [10]. Brazil's regional effort to enable meteorological and hydrological researchers to combine data, processing power, and expertise via grid technology as to create a tool for decision makers to explore the implications of the decisions over the water supply of the Brazilian Northeast Region, a semi-arid area [3]. None of these cases deal with the government at the city level, what imposes stringent requirements on the simplicity and maturity of the solutions employed.

3. Background

In this section, we present OurGrid, the grid solution used in this work, as well as the original Gerpav system, including the performance evaluation that guided its port to the grid.

3.1. OurGrid

There are a number of solutions for creating Computational Grids nowadays. For simplicity sake, we choose to use OurGrid [6]. OurGrid is an open, free-to-join, cooperative grid in which sites donate their idle computational resources in exchange for accessing other sites' idle resources when needed. For now, at least, OurGrid assumes applications to be Bag-of-Tasks (BoT), i.e. parallel applications whose tasks are independent. Despite their simplicity, BoT applications are used in a variety of scenarios, including data mining, massive searches (such as key breaking), parameter sweeps, simulations, fractal calculations, computational biology, and computer imaging. Moreover, due to the independence of their tasks, BoT applications can be successfully executed over widely distributed computational grids.

OurGrid strives to be non-intrusive, in the sense that a local user always has priority access to local resources. In fact, the submission of a local job kills any foreign jobs that are running locally. This rule assures that OurGrid cannot worsen local performance, a property that has long been identified as key for the success of resource-harvesting systems [18]. OurGrid is designed to be scalable, both in the sense that it can support thousands of sites, and that joining the system is straightforward. For scalability, OurGrid is based on a peer-to-peer network, with each site corresponding to a peer in the system. However, peer-to-peer systems may have their performance compromised by freeriding peers [13, 16]. A freerider is a peer that only consumes resources, never contributing back to the community. This behavior can be expected to have a very negative impact on OurGrid, because many users reportedly have an insatiable demand for computer resources, and thus we do not anticipate having a resource surplus to give to freeriders. We have dealt with this problem by creating the Network of Favors, a totally decentralized and autonomous allocation mechanism that marginalizes freeriders [1, 2].

A given site in OurGrid will commonly run tasks from other unknown sites that are also part of the community. This creates a very obvious security threat, especially in these days of so many software vulnerabilities. Therefore, we must provide a way to protect local resources from foreign unknown code. That is the job of SWAN (Sandboxing Without A Name), a solution based on the Xen virtual machine [4], which isolates the foreign code into a sandbox, where it can neither access local data nor use the network.

Users interact with OurGrid via MyGrid, a personal broker that performs application scheduling and provides a set of abstractions that hide the grid heterogeneity from the user. The great challenge in scheduling is how to assure good performance for the application in a system as large and loose-coupled as OurGrid. In particular, the scale of the system makes it hard to obtain reliable forecasts about the execution time of a given task on a given processor. To deal with this problem, we have devised schedulers that use task replication to achieve good performance in the absence

of performance forecasts [12, 15]. As for the abstractions, the goal is to balance between ease-of-use and performance, while considering the limitations of the current lack of general connectivity in the Internet [14].

In summary, OurGrid has three main components: the OurGrid peer, the MyGrid broker, and the SWAN security service. Figure 1 shows them all, depicting the OurGrid architecture.

3.2. Gerpav

Good roads are essential for the movement of people and goods, but there are substantial costs in construction and maintenance practices. The city of Porto Alegre, located in south Brazil, has been facing a swelling of its road traffic due to city growth, requiring rapid actions in order to improve and maintain its road system. Porto Alegre is considered in Brazil a middle-size city with approximately 500 km^2 and a population of 1 million and 400 thousand habitants (census of 2005). The city's department of transportation (SMOV), which is responsible for dealing with issues about pavement conservation and condition assessment, have been challenged in the last years to deal with limited budget, requiring even more efficiency in applying public resources.

In that context, SMOV decided to develop a Pavement Management System (PMS), which included the creation of an information system to aid in decision making about pavement assessment and maintenance. This system was named Gerpav and its first version was concluded at the end of 2004.

Gerpav helps SMOV maintain detailed information about the city road system, such as pavement, defect, track, lane and repair types. It has a report module where a user can simulate the degradation of pavements according to a set of parameters.

Gerpav divides internally the city road system in a number of segments, where a segment is a section of a road. Each segment can have up to four tracks. Each track is divided into a number of lanes, the lane has information about its type of pavement (cement, asphalt, stone, etc.) and field investigations count the defects by lane. Based on the data from field investigations, the system calculates the PCI value (Pavement Condition Index) for each lane and a global PCI for the track. The value for PCI ranges from 0 to 100, where 100 mean a perfect condition. Each type of pavement has different behavior and performance. Gerpav simulates pavement degradation of the pavements taking into account its type, according to Figure 2.

Gerpav manages all information using a database composed by 26 tables, which takes up around 200 Mb of disk space. The main tables used to represent the city road system are: Segment (31252 rows), Track (32408 rows), and

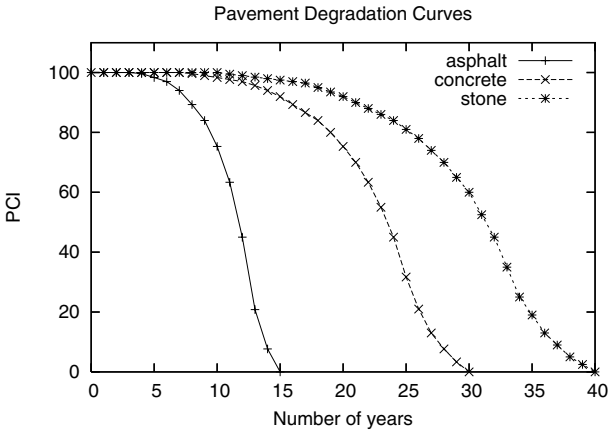

Figure 2. Pavement Degradation Curves

Lane (217141 rows). Functionality-wise, Gerpav is divided in 6 subsystems explained below:

Generic Tables Maintenance Subsystem: responsible for maintenance in generic system data tables, like pavement types, defect types, traffic.

Road System Maintenance Subsystem: responsible for maintenance of tables related to the road system, like roads, lanes and tracks.

Road System Report Subsystem: responsible for reports related to the road system tables.

PCI Calculation Subsystem: responsible for PCI calculation among different pavement types.

Data Importing Subsystem: responsible for importing data from external files generated by external applications.

Simulation Subsystem: responsible for simulations related to pavement assessment and maintenance planning.

In order to take advantage of the computing grid, we analyzed all subsystems to identify the one that has higher computation demands and could be decomposed in independent tasks to be executed by the resources provided by the grid. Gerpav Simulation Subsystem was identified as the best candidate for grid computing. Despite the algorithms in this subsystem are very memory and data intensive, it can be easily divided in several independent and coarse-grained tasks.

In Gerpav Simulation Subsystem there are several reports that can be generated. One of these is the "Pavement Behavior Report" that simulates pavement degradation over

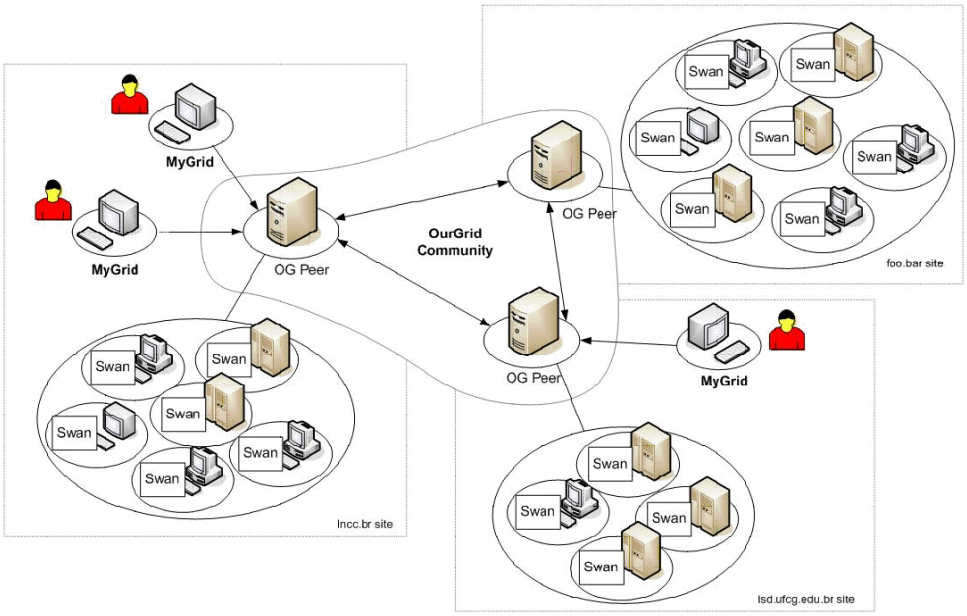

Figure 1. OurGrid Architecture

time. In order to create a grid version, the "Pavement Behavior Report" was the first selected for a closer analysis.

In the Pavement Behavior Report, the pavement behavior is calculated for each track, according to its predominant pavement type, following the degradation curve. Algorithm 1 shows how the simulation works. Note that there are three nested loops; the main loop computes the segments, tracks are retrieved for each segment and lanes are retrieved for each track. The algorithm computes predominant pavement based on lanes information and then calculates degradation for the track.

We performed some experiments generating the Pavement Behavior Report and measured the execution time of the main routines of Gerpav. Results showed that Gerpav performance is very tied to DBMS. This happens because of the way the application was designed with high coupling with database tables. Thus, application execution time is directly related to DBMS throughput.

The overall execution time to simulate pavement behavior for a neighborhood with 704 segments (almost 39 minutes), showed that it would be practically impossible to run a simulation for the entire city (31252 segments) using the original, non-grid application. Because of such limitation, Gerpav users were required to limit data for their simulations, and consequently they were not able to use Gerpav's report module in its utmost capacity.

Algorithm 1 Pavement Behavior Report Algorithm

```
1:  retrieve segments
2:  while there are segments to process do
3:      retrieve tracks
4:      while there are tracks to process do
5:          retrieve ICP
6:          retrieve lanes
7:          while there are lanes to process do
8:              identify predominant pavement
9:          end while
10:         calculate track degradation
11:     end while
12: end while
13: generate report
```

4. Gerpav in the Grid

The main motivation of GerpavGrid was to use the grid to solve the limitations in the generation of Pavement Behavior report. Our first goal was to port the generation of this report to the grid in order to get it for the whole city in much less time. We decided to use OurGrid to implement GerpavGrid due to it's simplicity to deploy grids.

GerpavGrid enabled to achieve not only performance en-

hancement, but also to aggregate new features that were not possible in the original design. GerpavGrid was implemented to be an extension of Gerpav, adding new functionalities and redesigning some of the simulations in the report module to use OurGrid infrastructure, in order to attain performance improvement.

While Gerpav is highly coupled with the Database Management System (DBMS), and designed to access an always-available database, GerpavGrid, in order to attain the advantages of parallel processing, cannot rely on DBMS availability as it was in Gerpav. In addition, additional effort is needed to adapt the application to the Bag of Tasks approach used by OurGrid.

Consequently, some parts of Gerpav were rewritten and several optimization strategies were tested to enable the application to take full advantage of the grid infrastructure.

In order to parallelize the generation of the Pavement Behavior Report, the generation algorithm was refactored and divided in three distinct phases: data preparation, processing and results consolidation. The processing phase was the one divided in several tasks to execute in the grid, while the other two phases are performed locally.

In order to facilitate the utilization of OurGrid and better adapt its utilization in the current web based implementation of Gerpav, we developed an utility framework to schedule tasks to grid resources. This framework is called OurGrid Job Abstraction Layer (OJAL) and is presented below.

4.1. OurGrid Job Abstraction Layer

Since we use the OurGrid infrastructure, we need to create a Job Description File (JDF) to submit a job for execution. This JDF contains details of each task that will execute a portion of a job. If a given job has hundreds of tasks, writing a JDF can be very tedious and time consuming. Therefore, some users prefer writing a Java program or customized script that submits a job for execution, bypassing the manual creation of the file.

MyGrid provides an API that facilitates this process, however its capabilities are not easy-to-use in a web based system as Gerpav. Therefore, to use MyGrid API in GerpavGrid, an extra abstraction layer was created. This layer was named OurGrid Job Abstraction Layer (OJAL) and it supports the automatic creation of OurGrid scripts to abstract details related to data serialization and deserialization, data slicing, job dispatching, and waiting.

Figure 3 shows the class diagram of OJAL. Classes with prefix Concrete must be provided by the application. A ConcreteClient can instantiate a JobDispatcher object, setting some parameters, such as slice size, jar files and remote class and call a dispatch method passing a collection with data to be processed by the grid. On the other hand, a ConcreteTask must extend RemoteTask and provide main and process methods.

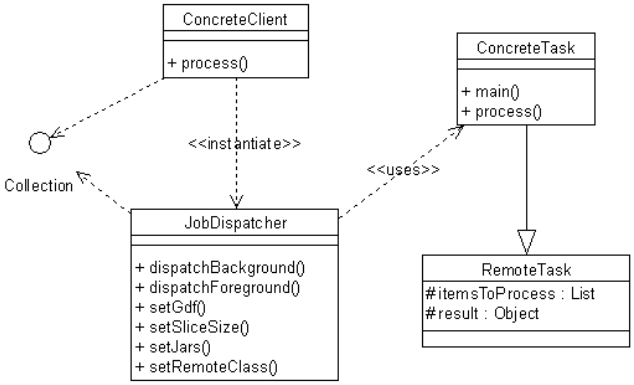

Figure 3. OJAL Class Diagram

The OJAL dispatch method will slice the collection and create the JDF (Job Description File) for MyGrid to transfer the ConcreteTask and related files to the Grid infrastructure. After job execution, application will get the return collection and continue its processing. Figure 4 presents the sequence diagram reproducing this process.

4.2. Optimization Strategies

One of the main issues in Gerpav is the highly dependency between the system and the DBMS. Besides storage, the DBMS is also used to perform filtering and basic calculation specified directly in the queries. In order to minimize the DBMS dependency and optimize the application using the Grid, we implemented and tested several versions using the optimization strategies explained below.

Database Filter versus Memory Filter: In the Database Filter strategy, the system retrieves tracks data from the database, using a complex SQL query to filter rows by the selection criteria and get only the needed tracks with attributes used for processing. The goal in this strategy is to reduce the size of data sent with each task to the grid. On the other hand, in the Memory Filter strategy, the system retrieves tracks data from the database, using a raw SQL query with very simple selection criteria, returning also an amount of tracks and attributes not used for processing. This data is sent with each task and the filtering of right tracks and attributes is performed by each task in the grid. The goal in this strategy is to reduce the amount of computation performed by the DBMS, resulting in faster queries.

Simple Slicing: In this strategy the system retrieves all necessary tracks data from the database using the Database Filter strategy. The tracks retrieved are divided in slices with a fixed number of elements. Each

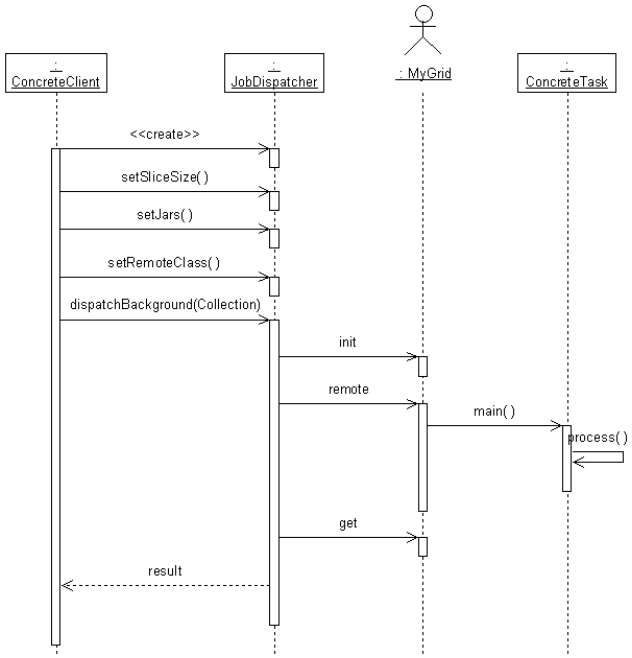

Figure 4. OJAL Sequence Diagram

slice is serialized and processed by a grid task. A simulation subsystem run is converted into the submission of a job (containing the grid tasks) to OurGrid. After job execution, the result collection is consolidated and sent to output.

Pipeline Dispatching: In this strategy the system retrieves tracks data from the database using the Database Filter or Memory Filter strategies. The tracks retrieved from the neighborhood are divided in slices with a fixed number of elements. Each slice is serialized and attached to a remote task; a job is dispatched to be executed by MyGrid containing the remote task. The system continues retrieving tracks data for the next neighborhood, without waiting for previous job completion. After the submission of the last neighborhood data, system waits for all jobs to complete, then the result collections (one for each neighborhood) are joined and the result is consolidated and sent to output.

Distributed Database: The data from segments, tracks and lanes is sent previously to databases hosted by OurGrid peers. The system creates a task for each neighborhood; a job is dispatched to remote execution of the tasks by OurGrid. The remote task retrieves data from the tracks accessing its peer database and filter data by selection criteria using Memory Filter Strategy. The system waits for job completion. After, it gets the result collection to consolidate results and send to output.

4.3. Additional Functionalities

Besides the implementation of the Pavement Behavior Report generation in the grid, we discussed with the Engineering group at SMOV that uses Gerpav in order to develop more simulation scenarios that could be useful to improve their workflow. This interaction resulted in two more simulations executed in the grid that couldn't be performed in the original version due to limitation in resources capacity. The new simulation scenarios are:

Budget Distribution: In this scenario the roads maintenance budget is distributed in a repairing schedule for a specified number of years using as parameters a priority model applied to roads characteristics, and a minimal PCI desired for each pavement type and road priority. It uses an intervention decision table which defines for each pavement type the available interventions to be made in relation to current PCI and its respective cost.

Initially, all possible intervention schedules for each road are generated for the time frame given (number of years). Schedules that don't satisfy the minimal PCI desired are excluded. This process is performed in the grid by each task for a set of roads, which returns a set of possible intervention schedules with their corresponding costs. After that, the budget is distributed using only the required interventions to obtain the minimal desired road system condition, and the remaining budget is distributed according to roads priority.

Intervention Optimization: This simulation is used to return an optimal schedule for a time period (in years) that provides the lowest cost given and average PCI for each pavement type and road priority. The priority model and intervention decision table used in the Budget Distribution simulation are also used in this simulation.

As performed in the Budget Distribution simulation, all possible intervention schedules for each road are generated for the time frame given (number of years), and schedules that don't satisfy the minimal PCI desired are excluded. After that, the average PCI for each pavement type in each schedule is calculated and schedules with an average PCI below the average PCI defined as input are discarded. The remaining schedules are sorted in relation to their costs and the cheapest one is selected. This process is performed in the grid by each task for a set of roads, which returns the cheapest schedule for each road. In the end, the system ends up with a schedule with minimal cost in order to have all roads in a condition equal or greater than the average PCI during all the time frame specified.

5. Evaluation

In this section we present the performance evaluation of GerpavGrid to show the benefits of porting the Gerpav application to a grid infrastructure. All tests were performed using the Pavement Behavior Report generation module to calculate pavement degradation for the next 15 years in three different scenarios: one middle-size neighborhood (Aberta dos Morros), 50 neighborhoods of different sizes, and for the complete road system of Porto Alegre. Details about each scenario considering the amount of segments, tracks and lanes are presented in Table 1.

Scenario	Segments	Tracks	Lanes
Aberta dos Morros	897	929	2094
50 neighborhoods	15070	15816	49195
Porto Alegre road system	31252	32408	98223

Table 1. Scenarios

GerpavGrid was deployed at the CPAD Research Center (*http://www.cpad.pucrs.br*). CPAD hosts one of the peers that compose the OurGrid Community grid, which is currently composed by approximately 20 peers with more than 350 machines (*http://status.ourgrid.org/*). For these measurements, GerpavGrid was installed in a local web server - Dual Xeon 3.0 GHz with 2 GBytes of RAM using SuSE Linux 9.2. This machine was also used to execute the MyGrid Scheduler to dispatch tasks in resources obtained from the OurGrid Community grid. In order to execute GerpavGrid, the following software packages were previously installed and configured in the server: Sun J2SE 1.4.2_08, IBM DB2 UDB 8.2, and Jakarta Tomcat 5.0.28.

In each measurement, some of the optimization strategies presented in the previous section were combined in different implementations and tested to analyze its performance (Table 2).

Version	Optimization Strategies	Type
I1	none	Sequential
I2	Database Filter and Simple Slicing	Parallel
I3	Database Filter and Pipeline Dispatching	Parallel
I4	Memory Filter and Pipeline Dispatching	Parallel
I5	Distributed Database and Memory Filter	Parallel

Table 2. Implementations

Figure 5 and Table 3 show the execution time in seconds of each implementation in each scenario.

We can observe that implementations using the Memory Filter (I4 and I5) provided much better results than the Database Filter strategy (I2 and I3) in all cases. This occurred due to the high penalty introduced by the complex query executed in the DBMS. We can also observe that in all the cases using the Database Filter, the results were very similar to the sequential version, being even worst in some cases. The best results were obtained using the Distributed Database optimization strategy, which reduces considerably the overhead imposed on a centralized DBMS enabling the trivial parallelization of queries in each peer. We can observe that using the grid, the necessary time to generate the Pavement Behavior Report reduced considerably, approximately 80 times faster than the sequential version for the larger scenario (Porto Alegre road system).

	Aberta dos Morros	50 neighborhoods	Porto Alegre road system
I1	200.287	3551.650	7266.923
I2	207.818	3370.382	7430.041
I3	210.408	3489.814	7113.705
I4	32.060	275.864	506.271
I5	12.312	50.327	89.877

Table 3. Execution time measurements

We already started experiments with the two new simulation scenarios: Budget Distribution and Intervention Optimization. In our first experiments we execute both simulations for a time frame of 7 years. The time to execute the simulations in the grid are approximately 56 minutes with the Budget Distribution simulation, and approximately 40 minutes with the Intervention Optimization simulation. Since these scenarios were not implemented in the sequential version, we couldn't perform the direct comparison between sequential versus grid version. However, analyzing the problem and based on some performance tests executed in Gerpav's server, we could optimistically predict that both simulations would take at least 15 hours to execute in a sequential version resulting in a speed up of at least 15 times.

6. Conclusions

This paper presents GerpavGrid, a distributed version of an application to maintain and plan the investments in the road system of the city of Porto Alegre. Since the sequential application was database intensive, we applied several optimizations strategies to move the bottleneck from database to memory to facilitate distribution of tasks in the Grid.

Grid technology brought substantial performance gains to the application, resulting in an augmented capacity of the city to plan road maintenance, dramatically reducing the workflow of the traffic engineer and leading to better results in road conservation with the available public money. Moreover, using a grid of idle resources the city administration had access to a high performance execution environment with a minimal additional investment in IT infrastructure.

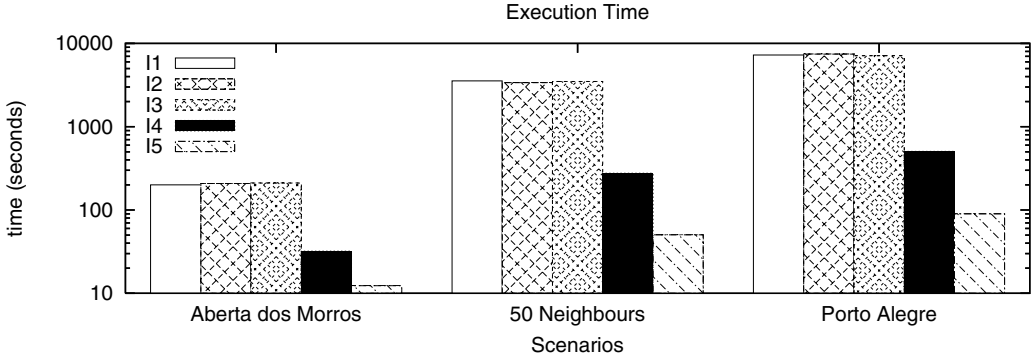

Figure 5. Execution time measurements

We believe that the results achieved with the GerpavGrid application could be reproduced in other similar applications and that a Grid can be also an interesting execution platform for non-classical HPC applications that are database intensive.

References

[1] N. Andrade, F. Brasileiro, W. Cirne, and M. Mowbray. Discouraging free-riding in a peer-to-peer cpu-sharing grid. In *Proceedings of 13th IEEE International Symposium on High-Performance Distributed Computing (HPDC13)*, Honolulu, Hawaii, June 2004.

[2] N. Andrade, M. Mowbray, W. Cirne, and F. Brasileiro. When can an autonomous reputation scheme discourage free-riding in a peer-to-peer system? In *Proceedings of 4th Workshop on Global and Peer-to-Peer Computing (GP2PC)*, Chicago, USA, April 2004.

[3] E. Araújo, W. Cirne, G. Wagner, N. Abílio, E. Souza, C. Galvão, and E. Martins. The seghidro experience: Using the grid to empower a hydro-meteorological scientific network. In *Proceedings of the eScience 2005: 1st IEEE International Conference on eScience and Grid Computing*, December 2005.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of SOSP 2003*, 2003.

[5] F. Berman, G. Fox, and T. Hey, editors. *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley & Sons, 2003.

[6] W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes, and M. Mowbray. Labs of the world, unite!!! *Accepted for publication in Journal of Grid Computing*, 2006.

[7] W. Cirne, F. Brasileiro, N. Andrade, L. Costa, D. Paranhos, E. Santos-Neto, C. D. Rose, T. Ferreto, M. Mowbray, R. Scheer, and J. Jornada. Scheduling in bag-of-task grids: The pauÁ case. In *SBAC-PAD 2004 16th Symposium on Computer Architecture and High Performance Computing*, 2004.

[8] K. Czajkowski et al. From open grid services infrastructure to ws-resource framework: Refactoring & evolution. http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf, Version 1.1, 3/05/2004. Cited 14 March 2006 (2006).

[9] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure, 2nd Edition*. Morgan Kaufmann, 2004.

[10] J. Furlong, R. Furlong, K. Facer, and R. Sutherland. The national grid for learning: a curriculum without walls? *Cambridge Journal of Education*, 30(1):91–110, March 1, 2000.

[11] Y. Li, M. Li, and Y. Chen. Towards building e-government on the grid. In *E-Government International Conference, LNCS 3416*, 2005.

[12] D. Paranhos, W. Cirne, and F. Brasileiro. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In *Proceedings of Europar 2003*, Austria, 2003.

[13] M. Ripeanu and I. Foster. Mapping the gnutella network: Macroscopic properties of large-scale peer-to-peer systems. In *Proceedings of First International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[14] S. S. and M. Livny. Recovering internet symmetry in distributed computing. In *Proceedings of GAN 03 Workshop on Grids and Advanced Networks*, Tokyo, Japan, May 2003.

[15] E. Santos-Neto, W. Cirne, F. Brasileiro, and A. Lima. Exploiting replication and data reuse to efficiently schedule data-intensive applications on grids. In *Proceedings of 10th Workshop on Job Scheduling Strategies for Parallel Processing*, 2004.

[16] S. Saroiu, P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN)*, San Jose, CA, USA, Jan 2002.

[17] F. Silva and H. Senger. The grid: An enabling infrastructure for future e-business, e-commerce and e-government applications. In *The Third IFIP Conference on E-Commerce, E-Business, E-Government (I3E 2003)*, pages 253–265, 2003.

[18] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: The condor experience. *Concurrency and Computation: Practice and Experience*, 17(2-4):23–356, 2005.

[19] S. Tuecke et al. Open grid services infrastructure (ogsi) version 1.0. global grid forum draft recommendation. http://www.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf, 6/27/2003.