Creating scalable traffic simulation on clusters.

Agathocles Gourgoulis
Gabor Terstyansky
Peter Kacsuk
Stephen Winter

School of Informatics

Creating Scalable Traffic Simulation on Clusters

Agathocles Gourgoulis, Gabor Terstyansky, Peter Kacsuk, Stephen Winter
Centre for Parallel Computing (CPC),
University of Westminster, London, UK
E-mail: agourg@cpc.wmin.ac.uk

Abstract

This paper describes the implementation of a transport simulation in a parallel environment. The implementation is based on a graphical parallel programming environment called P-GRADE. The transport simulator, called MadCity, simulates a specific road network of a city and shows cars moving on the roads. To achieve scalability of the traffic simulation, the use of templates is necessary. This helps to control the number of participating processes required for the simulation without making modifications to the simulator's source code. Performance results are collected from four, eight and sixteen nodes of the Parsifal cluster and compared with the sequential execution results of the simulator. The implementation of the transport simulator is extended further to support the simulation of multiple cities within the same cluster and on the Grid.

1. Introduction

Computational simulations are becoming increasingly important because they are the only way how some physical processes can be studied and interpreted. These simulations may require computational power not available even on most powerful supercomputers. A solution is to use multiple computers connected through a computer network (cluster) to investigate complex simulations. The application area that will be investigated on a computer cluster is the urban traffic simulation.

A cluster is a collection of interconnected stand-alone computers connected by a high-speed local area network that work together as a single, integrated computing resource. It is a homogeneous entity from the software point of view and not necessarily in hardware [1]. A computer node in a cluster may consists of a single processor or multiprocessor system including memory, operating system and I/O facilities. The network interface is responsible for transmitting and receiving messages between nodes. The communication software is responsible for providing efficient and reliable data communication between the nodes and the outside world. The cluster at the University of Westminster (UoW), called Parsifal, is composed of 32 computers (nodes) plus a master node [5]. If more computational power is required that one cluster is able to offer, the use of multiple clusters (Grid) is an option, in other words a cluster of clusters.

A computational Grid [9] is a collection of distributed resources and infrastructure services that can be used as a single entity to execute large-scale applications. It enables the sharing of a wide variety of heterogeneous resources that are geographically distributed on the network and make them available to users.

2. Describing the traffic simulation problem

Traffic simulations may require computational power not available on today's most powerful supercomputers. Real-time traffic simulations must provide the drivers with information about the road network such as the travel time, delays or accidents, route guidance systems to divert drivers away from the congested areas, or any other kind of assistance for their travel planning. Thus, the time that will be spent to run a real-time simulation or to make a short-term prediction of the following couple of minutes should be at minimum, so that all information will be distributed to drivers early enough to help them decide the best route to follow.

The objectives of traffic simulation are as follows:

- To decrease the execution time by parallelising the simulation and distributing the computation on different nodes as shown in Figure 6.
- To test the parallelisation on the cluster and check the work load on the nodes (using monitoring).
- To test scalability, to keep adding more nodes and hence more resources as they are required for better performance of the program.
- To extend the current single city simulation to multiple city simulation using multiple clusters (Grid).

Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (EUROMICRO-PDP'04)
1066-6192/04 $20.00 © 2004 **IEEE**

3. Graphical development and execution environment / P-GRADE

We use a graphical development and execution environment that provides an integrated set of programming tools for development of general message-passing applications to be run in heterogeneous computing environments. P-GRADE offers a number of benefits such as a Graphical User Interface (GUI) where all parallel activities of applications are defined. Figure 2 shows the Graphical User Interface of P-GRADE.

The graphical environment used to implement the traffic simulator is P-GRADE. *P-GRADE* (Professional GRaphical Application Development Environment) [3] is an integrated graphical programming environment for development and execution of parallel programs based on the MPI/PVM [7] message-passing programming paradigm. It consists of several software tools, which assists the different steps of the development process. P-GRADE supports writing, editing, executing (debugging, monitoring, visualizing) parallel programs. P-GRADE based applications can be run among others on clusters.

Some of the most important tools [4] implemented into the P-GRADE environment are the following: GRAPNEL is a hybrid programming language in the sense that it uses both graphical and textual representations to describe the distributed application. GRP2C is a pre-compiler that produces the C code of the graphically defined program. DIWIDE is a distributed debugger with the ability to debug the processes running on heterogeneous machines at the same time. PROVE is the visualisation tool that shows the monitoring results, etc.

All message-passing library calls are automatically generated by the graphical environment. Thus, programmers are able to use predefined process communication templates (process farms, pipeline or ring, 2D mesh and tree). Templates are the major tools for creating scalable applications for clusters. A communication template defines a group of processes that have a pre-defined regular interconnection topology. The user only has to define the actual size and all processes and channels are created by the system automatically. The most relevant difference between a communication template and a simple process group is the ability to change the number of member processes without modifying the graphical code of the application. The user defines the name of the template. Processes in the template (i.e. members) are identified by indexes generated automatically by the system. Different types of templates use different index patterns. An index pattern always consists of one or more non-negative integer numbers. For instance, in case of a process farm or a pipeline, a simple integer as a rank number is enough to identify the members, but in case of a 2D Mesh a pair of integers is required to identify both the row and the column positions of processes.

4. Implementation of the simulation

4.1. MadCity simulator

MadCity consists of two tools, the GRaphical Visualiser (GRV) and the SIMulator (SIM). The GRaphical Visualiser helps to design a possible road network generating a network file that describes the road network to be investigated. The SIMulator of MadCity is implemented on P-GRADE. When the simulation starts, the network file is sent to different nodes on the cluster. After the end of the simulation, a trace file is created. This trace file is loaded on the GRV to display the behaviour of the cars on the roads and at junctions in a city.

The Parsifal cluster is used to run MadCity traffic simulator. The road network described by a network file may contain several thousand roads and hundreds of junctions. To simulate a city road network, a single processor is not able to perform such a simulation within a limited period of time because a real-time simulation and a short-term prediction of the traffic for the next 5, 10, or maximum 15 minutes of time are required. A solution is to use cluster computing to run a single massively parallel simulation to carry out traffic simulations. To achieve parallelisation of the simulation, the network file should be distributed to all participating nodes, and each node should work on a particular road area.

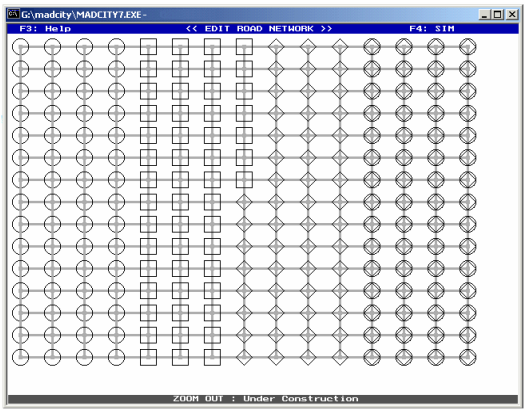

Figure 1. Manhattan road network on MadCity

Thus, it is possible to allocate traffic zones (Figure 1) on the road network in the way that each zone provides simulation locality and allows efficient parallelisation of the simulator. The simulation investigates a virtual road network called "Manhattan", and it consists of 225 junctions and 420 lanes. Performance results have been monitored from 4, 8 and 16 nodes of Parsifal cluster and compared with the sequential version of the simulator

based on a single processor. Figure 1 shows the MadCity GRV with the Manhattan network.

4.2. Template selection

P-GRADE provides three templates that offer different features in the cluster. The "farm" and the "2D mesh" are not suitable for the traffic simulation. The process farm template implements a communication pattern that is a group of processes without any interconnections. They can communicate only with processes outside the group but not with other processes within the same group. In the 2D mesh template, processes are arranged into a two dimensional Grid where each process (i.e. grid point) is interconnected with all of its neighbours. This approach could be considered as an appropriate but it is too complicated for the purposes of this simulation.

The pipeline template is suitable to implement the traffic simulator. A pipeline communication topology consists of a linearly ordered set of processes where each process is interconnected only with its neighbours. As the name suggests, this template can be used to implement pipeline parallelism where each member process has different task to do on data flowing through them. In case of the traffic simulator, all member processes perform the same task but on different data and communicate with each other exchanging information. Naturally, all processes in the pipeline may as well communicate with external processes. The user defines the code of the representative processes and using the template attribute settings (Figure 2) specifies the actual number of the processes that will participate in the program execution. The size of the template can be increased up to 100 processes, the communication channels between neighbour processes can be directed forward, backward or both (i.e. bi-directional channels). Also, the channel pattern can be cyclic if the last process is connected to the first one or acyclic otherwise.

5. Implementation and performance results of a single city simulation

MadCity is built on PVM (Parallel Virtual Machine) [6], the message-passing paradigm that P-GRADE also uses. We selected to examine the implementation of MadCity on four, eight and sixteen nodes. Figure 2 shows the simulation structure of MadCity (working on four nodes) and the graphical environment of P-GRADE where the simulation has taken place. It consists of the parent process and the "Children" pipeline template process. The user defines the code at the three representative processes of which number depends on the actual "template attribute" settings. The template attributes window of Figure 2 shows that four children processes (SIZE = 4) will participate in particular simulation. This number of processes can be easily increased or decreased according to the simulation needs by specifying the size at the template attributes window. The network file will be distributed to every child process to perform the simulation.

The simulation works as follows: the parent process sends the network file to every child process (within the template) together with the partition ID numbers. When we create the network file, we partition the road network (using GRV) according to the number of nodes we are going to use. Figure 1 shows the road network partitioned to work on four nodes, thus the generated network file contains four different ID numbers. Each group of shapes (circles, squares, etc.) corresponds to a particular node. Each child receives the same network file, but it works on the particular part of the network file according to the ID number it has been assigned. According to Figure 1, the first child works on the part of the network file described by the circles, the second child on the part described by the squares, the third child on the diamond, and so on.

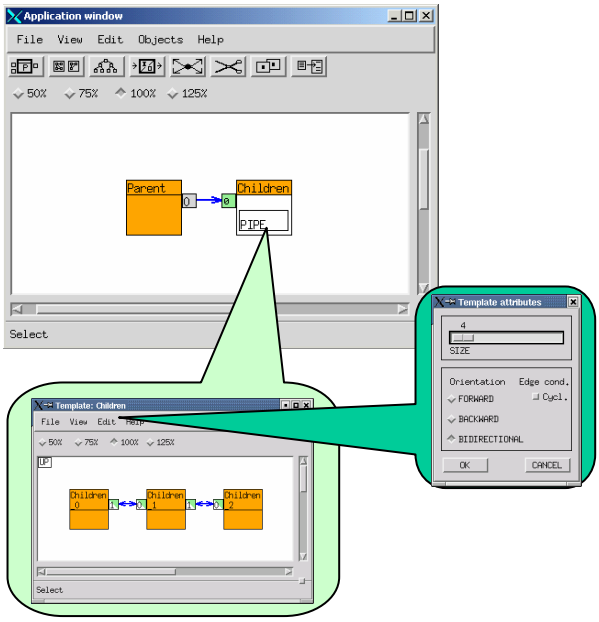

Figure 2. Single city simulation on P-GRADE

LCPs (Lane Cut Points) manage traffic if a junction resides on a partition boundary of a network segment. LCPs are the points where the partition boundaries cut the lanes. In other words, when a car is leaving a junction in a partition of the road network described by circles and moves to the neighbour junction of the partition described by the squares, LCPs are used. As shown in Figure 2, neighbour processes are communicating by exchanging the number of cars moving to the junction of the next partition. These cars are temporary stored in the LCP buffer and retrieved from the neighbour node by using synchronous communication. As a result, it simulates the continuous

movement of the cars on the road. The performance of the simulation depends on the number of the LCPs. The larger the number of LCPs leads to worse performance result of the simulator. This is due to the additional time required for communication between nodes. The number of LCPs used for this experiment (on four nodes) is 15 (between two nodes) or 60 in total for all four nodes.

The simulation steps (STEPS) are another factor that affects the performance of the simulation. They show how long a car will move on the road. Each node does its computation for a defined number of STEPS, and the results of each node are sent back to the parent process. The parent process collects all information from children processes and creates a trace file. This trace file is loaded on the graphical visualiser (GRV) that shows the road with cars moving on it.

The performance results that follow are based on the "Manhattan" network that consists of 225 junctions, 420 lanes and a maximum of 300.000 cars for 500 simulation STEPS. The results of the sequential simulation are compared with the results of the same network type applied on four, eight and sixteen nodes of the Parsifal cluster.

The use of P-GRADE was necessary to generate the parallel version of the traffic simulation, to run the traffic simulation and to monitor the behaviour of the simulation execution time on the cluster. The trace files from both the sequential and parallel simulations were collected and loaded on the graphical visualiser (GRV) of MadCity.

The following figures summarise the performance measured on four, eight and sixteen nodes. The black colour represents the computation time spent on each node to complete its task. The grey colour represents the communication time (the time was spent for communication on each process).

Figures 3 and 4 show the execution statistics of the overall performance of the simulation on 4-nodes and 8-nodes respectively.

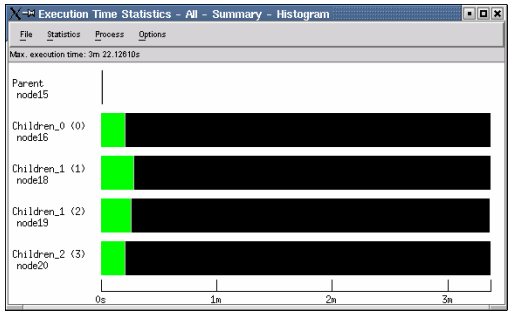

Figure 3. Performance results on 4 nodes

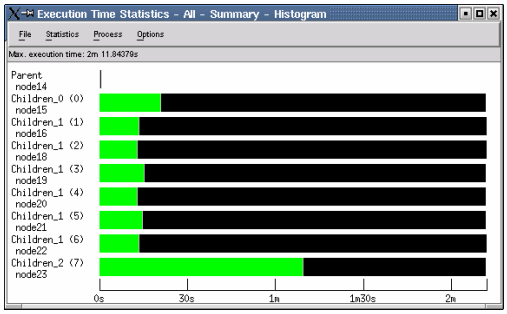

Figure 4. Performance results on 8 nodes

The execution performance on the 8-node simulation shows that the 8th child has smaller execution time than other children processes. That's because it is the only process that works on 15 junctions, the other processes work on 30. Figure 5 shows the total performance results and the execution time on 16 nodes. Similarly, the 15th and 16th child processes have less execution time than other children processes because fewer junctions have been assigned to these nodes.

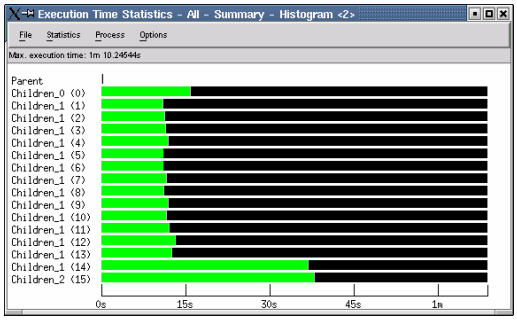

Figure 5. Performance results on 16 nodes

Figure 6 shows the difference in simulation time when the computation is executed on more than one node. As long as we increase the number of participating nodes, the execution time is decreasing.

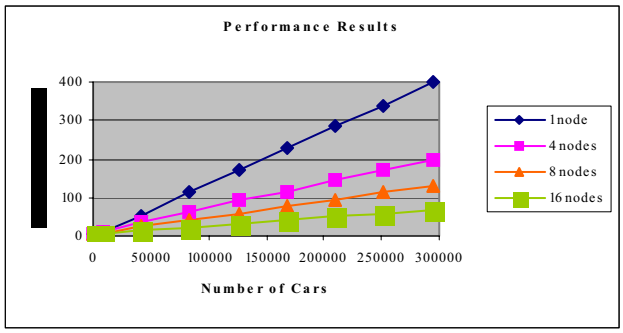

Figure 6. Simulation performance results

The parallel simulation has significant speedup over the sequential execution of the simulation. The following table

(Table 1) shows in detail the values of execution times (in seconds) of the above graph.

Table 1. Performance results values

Cars	1 node	4 nodes	8 nodes	16 nodes
294000	398	198	130	69
252000	340	172	113	58.8
210000	284	144	94	50.4
168000	229	116	77	42
126000	172	91	59.6	33
84000	114	62	40.3	23
42000	54	34.2	24.4	13.1
8400	11	10.3	7.1	5.5
840	5	4.8	3.6	2.8

According to the results from the graph above (Figure 6), if we want to simulate the "Manhattan" city and provide the drivers with a short-term prediction of traffic for the next 5 minutes, the sequential approach is not appropriate because the time spent on simulation (6min 40sec) is greater than the required prediction time (5 min). Running the simulation on 4 nodes, the time spent on simulation (3min 10sec) is less than the required prediction time, but not less enough because information should be passed to drivers on time to make their decisions. Thus, the appropriate simulation performance is achieved when the simulation is running on more than 8 nodes (e.g. on 16 nodes lasts 1min).

To summarise, the performance of the simulation depends on the following factors:

- the number of LCPs used. The larger the number of LCPs used the worst the performance is, since there is extra time spent for communication.
- the number of cars applied on the road network increases the computation time.
- the number of time STEPS used is responsible for the overall simulation time.
- the available nodes in the cluster ready to work on assigned simulation tasks.

6. Simulation of multiple cities within the same cluster

The simulator is extended to simulate two different cities within the same cluster. The difference between the two different types of traffic simulation is that using the multiple city simulation, we should be able to investigate not only the behaviour of cars within a single city, but also the traffic between two cities.

Figure 7 shows how two cities are implemented on P-GRADE. Once again the parent process is responsible for distributing the network file to the templates and all information necessary for the simulations. The pipeline templates called "City1" and "City2" define the children processes. On each city we can define how many children processes will participate for the simulation of the particular city that they belong to. Between the two cities there is a router process that dispatches the messages that are exchanged. These messages carry information about the cars that travel to the neighbour city.

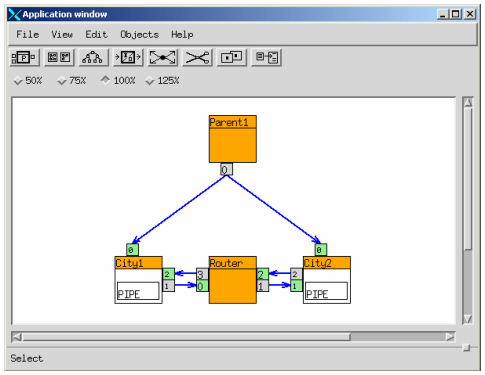

Figure 7. Simulation of two cities

Figure 8 shows the performance results collected from the simulation of the two cities. There are four processes that work on each city. The fourth process on the second city (City2_2) has half execution time compared with the other execution times. It is because half the amount of junctions has been assigned to work on that process.

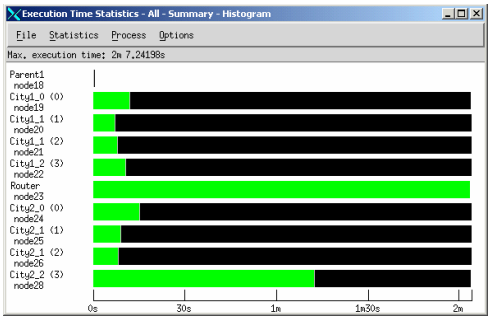

Figure 8. Performance results on two cities

7. Further work (Extending the simulation of multiple cities towards the Grid)

The MadCity simulator (SIM) is extended one step further to simulate more than one city on different clusters (University of Westminster and MTA SZTAKI). The "multicity" simulation is implemented on P-GRADE (Figure 9) where two cities are simulated at the same time.

Figure 9 shows two different processes (orange colour boxes) that will simulate three different cities (Liverpool and Manchester) using hypothetical road networks and

different number of cars for each city. Each process communicates with a pipeline template (orange and white colour boxes) that helps to increase or reduce the number of participating nodes necessary for the simulation within a particular cluster. Templates offer the ability to change the number of member processes without modifying the general code of the application. So, in case that the need of more processes is required (e.g. 32 or more), the only thing that should be changed is the number of member processes inside the template instead of drawing 32 or more different processes.

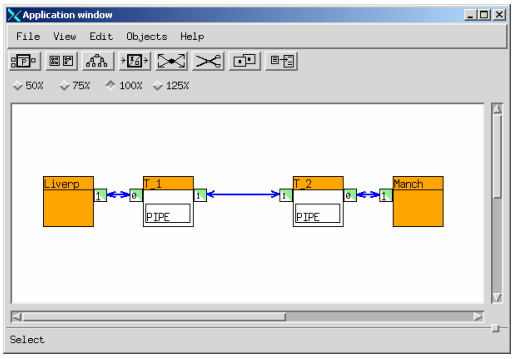

Figure 9. MultiCity on P-Grade

Communication messages are exchanged between templates and between templates and processes that describe the behaviour of cars that are moving between the cities. To minimise the communication time, the use of local buffers is essential. All car information that is about to change a city is stored in a local buffer and sent at the end of the simulation as one message.

The same high-level graphical environment of P-GRADE used to develop parallel programs for clusters is now extended towards the Grid. This environment enables the Grid application programmer to develop a parallel program that can be executed as a Grid job on any parallel site of a Grid in a transparent way. P-GRADE supports the Condor mode for job submission to the Grid. In Condor [10] mode, P-GRADE constructs the necessary description file containing the resource requirements of the parallel job, and it submits the Condor job to the Condor pool. The restriction of this mode is that it can only be used if the submitting machine is part of a condor pool.

To execute the multicity simulation of MadCity, the program should start as a Condor job under P-GRADE. Using the mapping option, two clusters can be selected to participate for the job execution of the simulation. The monitor system can also be used for on-line monitor and visualisation of the job status, processes and their interaction on all clusters simultaneously.

8. Conclusion

The implementation of a scalable traffic simulator has been presented. The time of the sequential simulation was compared with the parallel execution of the simulation on the cluster. Parallelising the simulation reduced the execution time, so that short-term predictions of traffic can be achieved. Templates offer the ability to create scalable solutions, such as the traffic simulator, where the number of participating nodes can be easily increased or reduced without modifying the code of the simulator. P-GRADE was a successful tool that helped to implement the simulator, make the parallelisation and monitor the performance of the simulation on the cluster. The simulation of a single city was extended to the simulation of multiple cities within the same cluster and towards the Grid to take advantage of more computational power that one cluster can offer.

9. References

[1] Buyya Rajkumar, *High Performance Cluster Computing Volume 1: Architectures and Systems,* London, Prentice-Hall International (UK) 1999.

[2] A. Apon, R. Buyya, H. Jin, J. Mache, "Cluster Computing in the Classroom: Topics Guidelines, and Experiences", 2001

[3] P. Kacsuk, "Visual Parallel Programming on SGI Machines". Invited paper, Proc. of the SGI Users. Conference, Krakow, Poland, pp. 37-56, 2000.

[4] P-GRADE User's Manual: http://www.lpds.sztaki.hu/projects/p_grade/manual/manual_frame.html

[5] The Centre for Parallel Computing Computational Cluster: http://parsifal.cpc.wmin.ac.uk

[6] PVM: Parallel Virtual Machine: http://www.csm.ornl.gov/pvm/pvm_home.html

[7] MPI – The Message Passing Interface Standard: http://www-unix.mcs.anl.gov/mpi

[8] SZTAKI cluster: http://www.lpds.sztaki.hu/cluster_computing/klaszterjell/index.htm

[9] Baker M., Buyya R., Laforenza D. "Grids and Grid Technologies for the Wide-Area Distributed Computing", 2002.

[10] J. Frey, et al, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", Proc. of the10th IEEE Symp. on High Performance Distributed Computing (HPDC10), 2001.