FEDERATING A PARALLEL TRAFFIC SIMULATION USING HLA

Narain Ramluchumun

Nasser Kalantery

Stephen Winter

School of Informatics

University of Westminster,

115 New Cavendish Street,

London, United Kingdom, W1W 6UW

Email: {n.ramluchumun, kalantn, wintersc }@wmin.ac.uk

KEYWORDS

performance, parallel traffic simulators, federates, HLA, RTI.

ABSTRACT

An outline of parallel road traffic simulation based on the High Level Architecture (HLA) is presented. The methodology used to achieve parallelisation includes domain decomposition where the road network model is partitioned into sub-models and these sub-models are simulated on different cluster nodes. The domain decomposition concept is used for the parallelisation strategy where the road network is partitioned into parts and these parts are simulated on different computers. All federates simulating the different parts of the network communicate through the Runtime Infrastructure (RTI). The RTI services are used to realise simulation time advance and event synchronisation, and to the transfer of data across the network partitions. This paper discusses the implementation issues and also reports elements of the performance analysis.

INTRODUCTION

This paper focuses on the implementation issues and performance analysis of parallel traffic simulation in HLA environment. The primary aim of this aspect of the research work was to develop an efficient implementation of parallel traffic simulation using a cluster computing technology. Until now most of the HLA applications have been generated in a sequential manner (Ramluchumun 2002; Schulze *et al.* 1999; Horst *et al.* 1998; Schulze *et al.* 2000) and very few have been parallelised in the HLA environment (Bodoh and Wieland 2003; Riley *et al.* 2004; Wu *et al.* 2001). The main objective of this analysis was to investigate the suitability and efficiency of HLA in cluster environment for parallel traffic simulation.

HLA offers several management services (Defence Modelling and Simulation Office 1998; Kuhl *et al.* 1999) but only a few of these services have been used in this current study to demonstrate parallel traffic simulation and to analyse its performance. The federation management services were used to initialise the execution of the traffic federation. The execution of such federation included reading the federation object description file, contacting other federates in the simulation and creating the basic two-way communications path between all the federates running on the different cluster nodes.

Declaration management services were used to define object and interaction classes and to set up communication between the federates using publish and subscribe paradigm. The federates were also allowed, using object management services, to declare object instances, update attributes, send interactions, receive updates to attributes, and receive interactions produced by other federates.

Finally, the time management services were used to control the advancement of the traffic simulation time within each federate and to prevent the federate from receiving messages in the past i.e. time stamp less than the federate's current simulation time.

PERFORMANCE ISSUES

The performance of the parallel traffic execution is determined by a complex interplay of several factors (Fujimoto 1998; Gourgoulis *et al*. 2004; Igbe *et al.* 2003). In this project, two factors are described which have considerable impact on performance. The first one is the Lookahead which is a concept used to improve performance in parallel and distributed simulations. A lookahead T for a federate means it will only generate messages at least T units of simulation time into the future (Fujimoto 1998). A larger lookahead allows more parallelism but can affect the overall performance of the simulation execution. In the traffic simulation application the lookahead value is chosen to be the size of one time step.

Another important issue is when federates need to be aware of shared information i.e. beyond their own sub-model. For instance, during the federation execution, when cars are at a junction some federates need to determine the next destination of the cars. The federates should therefore be aware of the entire topology of the road network and subsequently, the next destination is computed based on some pre-defined algorithm. In the traffic simulation application, the next destination is computed using a random function. Also, when cars have to cross partition boundaries they are moved to transient buffers commonly known as Lane Cut Points (LCPs) (Igbe et al. 2003). Eventually, at the end of the simulation step they are moved to the appropriate destination lane. The number of LCPs in a road network does affect the overall performance of the parallel simulation.

Execution time

The *execution time* in this context, is the wall clock time taken for the parallel traffic simulation to run to completion. During this time, different federates are initialised on different cluster nodes. The partitioned road network is mapped to the respective federates which is simulated for a number of timesteps before the simulation results from the different nodes are collected and merged. The execution is measured for a reasonable number of timesteps to minimise errors with regards to communication delays. It is computed by recording the start and end time before executing the main simulation loop using functions from the time library.

Speedup

The *speedup* provides an indication of the effective number of processors utilised for executing a program (Buyya 1999). In this study, speedup was used to estimate the parallel traffic simulation performance with respect to the sequential version with increasing number of processors.

$$\boxed{\text{Speedup} = \frac{\text{sequential execution time}}{\text{execution time with N processors}}}$$

The speedup is defined here as the execution time taken by the sequential simulation divided by the execution time of the parallel simulation, which is executed on N processors. The value of N is varied in a set of experiment to determine the efficiency of the parallelisation approach and the system used.

Efficiency

In this parallel traffic application, *efficiency* is used as a measure of the effectiveness of the hardware and software being used (Buyya 1999).

$$\boxed{\text{Efficiency} = \frac{\text{Speedup}}{\text{Number of processors}}}$$

As shown above, the efficiency is defined as the speedup gained divided by the number of processors used in the parallel traffic simulation execution. A set of experiments with varying number of processors has been designed to determine the efficiency of the traffic system following the above-described approach.

THE PARALLEL TRAFFIC SIMULATION

A number of experiments were set up to evaluate the performance of the parallel traffic simulator prototype. The parallel simulation ran on a system of 32 Dell PowerEdge 1400SC computers which runs Linux RedHat 7.2 operating system (Gourgoulis *et al.* 2004). Each computer had a Pentium III 1Ghz CPU and 512MB of memory. The master node, on the other hand, was a Dell 1500SC with two 1133 MHz processors and 2GB of RAM. The 32 computers were connected to a Cisco switch at 100Mbps using Ethernet technology whereas the master node was connected at 1000Mbps to the Cisco switch.

At initialisation stage, each partition of the road network was mapped onto a Dell system and thus the entire road network was simulated as a single system. In a first set of experiment, a road network with four partitions was simulated. At this stage, only 4 nodes of the cluster were used as one partition was mapped to each node.

The same experiment was repeated for different number of nodes in order to determine the speedup as the number of nodes increases. These observations were used to compute the efficiency of the parallelisation approach used to execute the traffic simulation on a cluster of workstations. The figures should also give an indication of the suitability of the hardware used for this purpose.

A second set of experiments were designed to determine the effect of increasing the load on the execution time. The load on each processor was varied by varying the number of vehicles in the simulation execution. The execution time was measured for every set of experiments to determine any speedup in increasing the number of vehicles. The aim of this experiment was to investigate the suitability of HLA and cluster technology to simulate large urban areas where there were thousands of cars.

Further experiments had been carried out to determine the cause of poor performance of the parallel simulator executing in HLA environment. Two different HLA time management mechanisms (Fujimoto 1998; Kuhl *et al.* 1999), more specifically Time Advance Request (TAR) and Next Event Request (NER), were further investigated.

ANALYSIS OF EXPERIMENTAL RESULTS

Several sets of experiments were carried out on up to 16 nodes of the cluster and the results are discussed in the following sections.

Execution time against Number of Cars (Load)

This experiment was designed to measure the performance of the parallel simulator in terms of execution time.

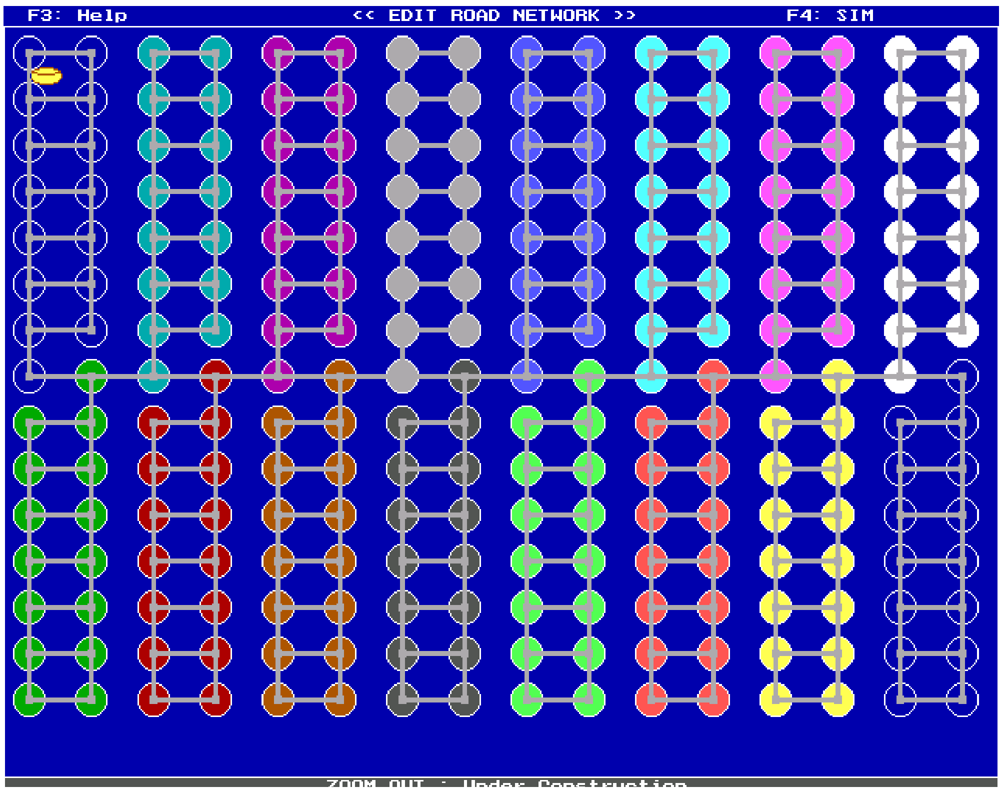

Figure 1: 15 x 16 road network with 16 partitions

The load (number of cars) on the network depicted in Figure 1 was varied and the run time on different number of cluster nodes were measured. The execution time against the load is illustrated in Figure 2 and the derived speedup is shown in Figure 3.

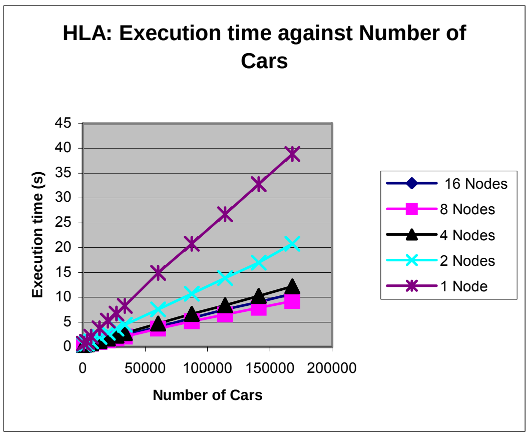

Figure 2: Execution time against Number of Cars using TAR in HLA

Figure 2 shows similar behaviour for the different number of nodes execution i.e. the execution time increased linearly as the number of cars was increased. Also, as expected, the execution time decreased as the number of nodes was increased. However, there was not much decline in the execution time for 8 and 16 nodes run. As the number of nodes was increased beyond 4, there seemed to be communication overhead resulting in smaller speedup.

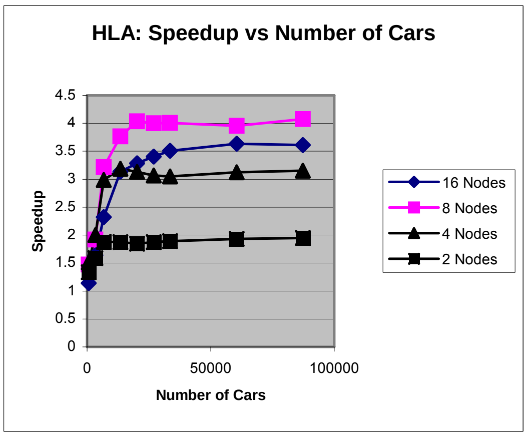

Figure 3: Speedup against Number of Cars in HLA

The speedup was also affected by the number of transient messages in the buffers, i.e., the cars moving across the boundary. This number increased as the number of cars were increased, i.e., more cars moved across the boundaries. The number of transient buffers increased as the number of network partitions was increased. As shown in Figure 3, different number of nodes execution reached the steady state at different point. This may be caused by the number of cars in the transient buffers which decreased with increasing number of nodes, therefore resulting in wider transitional states.

It is to be noted that in order to plot the *Speedup against Number of Cars* graph only values up to 90000 cars had be used. This allowed a better analysis of the part of the graph where the speedup was and also, after 90000 cars the speedup seemed to remain constant.

Efficiency of the Parallel Simulation

The following experimental data was derived to investigate the efficiency of increasing the number of cluster nodes to reduce execution time. Based on the experiments carried out in the previous section, only three set of data were used to plot a graph of *Efficiency against Number of Nodes*. Six thousand seven hundred and twenty, 67200 and 134400 cars were investigated to determine how the simulation performs with small and large number of cars.

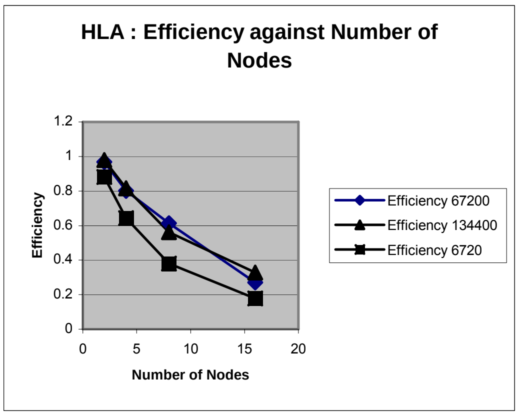

Figure 4: Efficiency against Number of Nodes

As shown in Figure 4, there is a considerable decline in the efficiency as the number of federates was increased up to 16. For instance, the observed efficiency was approximately 99% for 2 federates execution, 80% for 4 federates execution, 53% for 8 federates execution and then decreased further as the number of cluster nodes were increased.

Also, better results were observed for a larger load, i.e., road networks with greater number of cars. This decline in performance may be due to communication overhead. It was found that as the number of cluster nodes increased, the computation time decreased and at the same time, the communication and synchronisation overhead increased. Further experiments that were carried out to investigate the communication overhead are described in the following sections.

Idle time against Number of Nodes

This experiment was carried out to determine the federates' idle times with increasing number of nodes. During the simulation time, regulating federates alternated state between the granted state and the advancing state.

The *granted state* is the state during which the simulation is performing computations at some logical times.

The *advancing state* is when the federate has issued a time advance request to the RTI and is waiting for a grant.

When the federate is granted permission to advance its logical time, it resumes the granted state. During the time it is waiting for the grant, the federate may receive and process any time-stepped interactions delivered by the RTI from other federates.

Figure 5 depicts the advancing state for different number of federate execution with varying load.

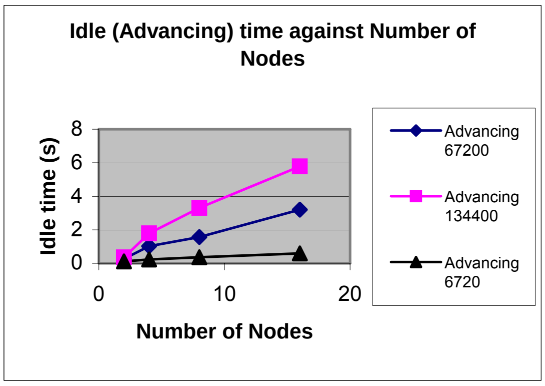

Figure 5: Federate idle time against Number of Nodes

The idle time is a measure of the time a federate has to wait until it is granted permission to move to the next time step. As the number of federates and load increased, federates were spending more time in advancing state, i.e., waiting for permission to be granted to advance their logical time to the next step than, in the granted state where they performed the real simulation computation.

One reason for increasing advancing state may be because of the simulation was load imbalanced which, therefore, meant that federates finishing their simulation earlier would have to wait until all federates were done. This situation arose as the vehicles were moving from one region to another and after a period of time, some regions might have had more vehicles compared to others. If, for a major part of the simulation time, the maximum workload was significantly greater than the average workload then, the simulation was considered to be load imbalanced. Further experiments will be carried out in the near future to test this hypothesis.

CONCLUSION AND FUTURE WORK

This paper focuses on the feasibility of parallel traffic simulation in cluster environment. The implementation issues have been discussed and a performance analysis of the results presented. The performance graphs show that reasonable speedup has been achieved but, as the number of cluster nodes was increased, the gain in speedup decreased. A breaking point seems to be reached after eight nodes execution since the communication overhead increased considerably thereafter. Further experiments need to be carried out to investigate the barrier synchronisation and global check methods adapted in communication algorithm for the parallel simulators.

Partitioning strategies and load balancing algorithms will also be investigated to optimise the performance of the parallel traffic simulator. Since the overall performance of the federation is determined by the slowest federate, the next phase of the research work will therefore focus on how redistributing the loads of heavily loaded federates affects the performance of the federation. These results will then be compared to traffic simulators based on other parallel mechanisms like Parallel Virtual Machines (PVM) or Message Passing Interface (MPI) for communication on clusters.

REFERENCES

Bodoh David J., Wieland F. 2003. *"Performance Experiments with the High Level Architecture and the Total Airport and Airspace Model (TAAM)."* Seventeenth Workshop on Parallel and Distributed Simulation (PADS'03), June 10 - 13, 2003 San Diego, California.

Buyya R., 1999. "High Performance Cluster Computing: Architectures and Systems Vol 1." Prentice Hall 1999, ISBN0-13-013-784-7

Defence Modelling and Simulation Office (DMSO). 1998. *"The High Level Architecture (HLA) Homepage.":* https://www.dmso.mil/public/transition/hla 1998.*(Feb 2002)*

Fujimoto R. M. 1998. *"Time management in the high level architecture."* Simulation, 1998,71(6):388-400

Gourgoulis A., Terstyansky G., Kacsuk P., Winter S. C. 2004. *"Creating Scalable Traffic Simulation on Clusters,"* pdp, Vol. 00, no. , p. 60, 12th 2004.

Horst M., Roberts D. W., Esslinger D. I., Marks J. R., Johnson, T. B. 1998. *"Building an HLA Radar Federation from Legacy Simulations."* 1998 Spring Simulation Interoperability Workshop, 98S-SIW-213

Igbe D., Kalantery N., Ijaha S., Winter S. C. 2003. *"An Open Interface for Parallelization of Traffic Simulation.". DS-RT, Vol. 00, no. , p. 158, Seventh 2003.*

Kuhl F., Weatherly R., Dahman J. 1999. *"Creating Computer Simulation Systems. An Introduction to High Level Architecture"* Prentice Hall 1999, ISBN 0-13-022511-8.

Ramluchumun N., S. Ijaha, S.C. Winter, N. Kalantery. 2002. *"An Open Framework for Traffic Simulation Tools using the High Level Architecture (HLA)"* 2002 Conference AI, Simulation and Planning In High Autonomy Systems- AIS Lisbon 2002. SCS ISBN: 1-56555-242-3.

Riley G., Ammar M., Fujimoto R., Park A., Perumalla K. and Xu D. 2004. *"A Federated Approach to Distributed Network Simulation"* ACM Transactions on Modelling and Computer Simulation (TOMACS), Vol. 14(2), April 2004.

Schulze T., Straßburger S., Klein U. 1999. *"Migration of HLA into Civil Domains: Solutions and Prototypes for Transportation Applications."* SIMULATION, Vol. 73, No. 5, pp 296-303, November 1999.

Schulze T., Straßburger S., Klein U. 2000. *"HLA-Federate Reproduction Procedures in Public Transportation Federations."* In Proceedings of the 2000 Summer Computer Simulation Conference, Vancouver, Canada.

Wu H., R. Fujimoto, and M. Ammar. 2001. *"Experiences Parallelizing a Commercial Network Simulator."* Proceedings of Winter Simulation Conference, December 2001 p1353-1360.

AUTHOR BIOGRAPHIES

Narain RAMLUCHUMUN is a PhD student in the Centre for Parallel Computing (CPC) at the University of Westminster, UK. His work is focussed on Parallel and Distributed Software Engineering using the High Level Architecture and Cluster Technology.

Dr Nasser KALANTERY has recently passed away and this work is dedicated to him. Dr Kalantery was a Reader in Parallel and Distributed Simulation at the School of Informatics, University of Westminster, UK. His research work was on High Performance Networked Computation via Logical Time and Structured Shared Memory, and Parallel Discrete Event Simulation.

Professor Stephen WINTER is the Dean of School of Informatics at the University of Westminster, London, UK. His research interests are in the technology and applications of parallel and distributed computing.