Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

A SYSTEMS APPROACH TO SCALABLE TRANSPORTATION NETWORK MODELING

Kalyan S. Perumalla

Oak Ridge National Laboratory
One Bethel Valley Road
Oak Ridge, Tennessee 37831-6085, U.S.A.

ABSTRACT

Emerging needs in transportation network modeling and simulation are raising new challenges with respect to scalability of network size and vehicular traffic intensity, speed of simulation for simulation-based optimization, and fidelity of vehicular behavior for accurate capture of event phenomena. Parallel execution is warranted to sustain the required detail, size and speed. However, few parallel simulators exist for such applications, partly due to the challenges underlying their development. Moreover, many simulators are based on time-stepped models, which can be computationally inefficient for the purposes of modeling evacuation traffic. Here an approach is presented to designing a simulator with memory and speed efficiency as the goals from the outset, and, specifically, scalability via parallel execution. The design makes use of discrete event modeling techniques as well as parallel simulation methods. Our simulator, called SCATTER, is being developed, incorporating such design considerations. Preliminary performance results are presented on benchmark road networks, showing scalability to one million vehicles simulated on one processor.

1 INTRODUCTION

Accurate estimation of metrics related to transportation in planning under time deadlines (as in emergencies) requires fast evaluation of multiple candidate scenarios using high-fidelity simulations. Fast models optimized for such scenarios are required for best possible simulation speed that capture the right amount of fidelity and provide sufficient accuracy. The short amount of decision times can demand fast simulation turnaround times, while large geographical coverage for decisions requires evaluating large-scale scenarios. Quantitatively, these requirements translate to the ability to perform detailed simulation of traffic on the order of a few million vehicles traversing across a few million road intersections. Added to these temporal and spatial s-cales is the need to capture other related sociological and behavioral phenomena that affect the traffic patterns and behaviors. Together, the new requirements create the need for a simulation environment that can scale up to several cities worth of traffic and road networks, requiring the use of parallel simulation techniques. However, few traffic simulators have been built for scalable parallel execution. Based on the needs of meeting the simulation fidelity, speed and scale of anticipated scenarios, we are building a new simulation environment called SCATTER – Scalable Tool for Transportation and Emergency Research – which (a) employs an appropriately determined level of micro simulation at vehicular level to capture sociological behaviors pertinent to the scenarios (b) optimizes the data structures for modeled entity representations to minimize memory usage (c) casts the execution as a discrete event modeling formulation to skip over uninteresting aspects of vehicular movement (d) structures the models from the outset to enable efficient parallel execution.

1.1 Related Work

Tools such as TRANSIMS (Smith, et al. 1995), VISSIM (Innovative Transportation Concepts 2001), CORSIM (ITT Systems & Sciences Corporation 1998, Prevedouros and Wang 1999), MITSIM, PARAMICS (Cameron and Duncan 1996) and OREMS (Bhaduri, et al. 2006, Franzese and Han 2001) have been developed with varying goals and capabilities (Fujimoto and Leonard II 2002). Each of those tools has been designed for their respective target applications. The SCATTER approach is, at a fairly high-level, distinguishable by its modeling framework to accommodate new sociological phenomena, its discrete event modeling approach for speed, and, most importantly, the structure and design for scalable parallel execution.

The rest of the document is organized as follows. The modeling framework of SCATTER is described in Section 0. Details of the discrete event simulation algorithm that underlies the vehicular mobility models is presented in Section 3. A preliminary performance study of memory usage and execution speed on parallel computing platforms

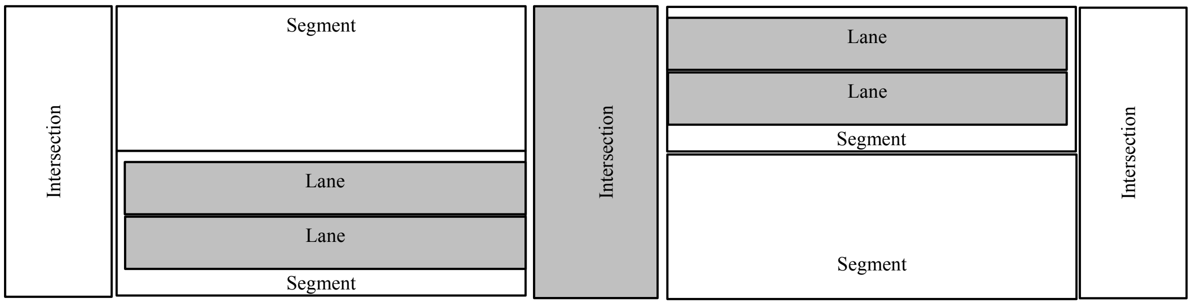

Figure 1: Illustration of the Basic Modeling Unit (Grayed Region), Which is the Intersection Plus Lanes of All Incoming Road Segments

is presented in Section 4. Summary and future work are outlined in Section 5.

2 MODELING FRAMEWORK

We will describe the main features of SCATTER's modeling framework, its design elements and uses. Additional considerations on interfacing the traffic movement features with various data sources is discussed in (Perumalla and Bhaduri 2006), and hence not duplicated here.

2.1 Generalized Time Advancing Framework

The framework is designed for incorporating time-stepped (TS), discrete-event (DES) or any combinations of TS & DES across modeling units. For example, time-stepping can be employed for road segments in which accurate driver behavior needs to be captured (e.g., arbitrary number of dynamic lane switching decisions between entry and exit on a single, short/congested road segment), and DES-based time-stepping can be used on other road segments (e.g., relatively vacant highway).

2.2 Indivisible Modeling Unit

Following the overriding goal of parallel execution, the network model is first structured into multiple indivisible units whose states are by design decoupled from each other. All interaction between the units is via exchange of time-stamped events, with no shared memory/state across units. This permits maximum flexibility for arbitrary mapping of units to processors. Also, by mapping each unit to a logical process, it enables parallel execution via parallel/distributed discrete event simulation techniques.

There are multiple alternatives for the definition of such modeling units. For example, the network model could be divided by cutting each intersection in the intersection area. Such partitioning, however, suffers from potential violation of mutual exclusion conditions – the necessity for preventing more than one vehicle occupying the same physical position at an intersection. Another alternative is to cut a road segment in the middle. This suffers from zero lookahead(Fujimoto 1990) limitation across the cut road segments.

Our partitioning approach uses a novel combination of intersection and incoming slices of road segments, as shown in Figure 1.

The basic modeling unit consists of a slice of the road network that is strangely shaped at first sight. Given a set of roads and intersections, our modeling approach divides them into units. Each intersection is associated with one unit. This unit also then includes all the lanes of incoming roads incident onto that intersection. The benefit of this modeling approach is that all routing decisions and vehicle forwarding decisions can be performed without mutual exclusion violations at the intersection. Also, sophisticated traffic light controllers can be encoded without parallel/distributed memory considerations.

2.3 Vehicle Identity

Vehicles retain their identity during the entire duration of simulation. As in real life, all vehicles are instantiated at initialization, and persist throughout the period of simulation. Vehicles are not deleted during periods of inactivity – they remain "parked" at the nodes that are intermediate points in their trip (e.g., at work, home, etc.).

Vehicles are exchanged as events. When a vehicle leaves one intersection and enters another intersection, it is modeled as a time-stamped event sent by the source logical process to the destination logical process.

2.4 Basic Kinetics

The basic set of kinetics associated with typical vehicular traffic micro models are included. Each intersection unit is responsible for advancing the state of vehicles on all its incoming road segments. For each vehicles, the following kinetics are modeled.

- Car following: A simple car-following model is included in which vehicles obey the acceleration, deceleration and velocity constraints imposed by vehicles in front of them. Inter-vehicular gap is always maintained for greater realism.

- Lane changing: A simple variant of lane-changing behavior is modeled, which is slightly biased towards filling all lanes of road segment. This type of lane-changing is a specialization for emergency/evacuation scenarios; the lane choice is made by the vehicle upon entry onto a road segment from an intersection. This decision is performed at every intersection; hence, a vehicle can make multiple lane changes during its trip along a route.

2.5 Trip Modeling

Sophisticated trip behavior is incorporated via the ability to define trip plans (home, work, grocery, etc.) of arbitrary combinations, on a *per-vehicle basis*. Care has been taken to minimize the memory usage due to the power of such a feature, by encoding the trip in a tight manner. A trip plan consists of several trip steps. Each step is a triple: <destination type, destination identifier, time>. Destination type could be an ordinal value such as HOME, WORK, GAS, etc. Destination identifier could be the unique identifier of any intersection in the network, in which case the vehicle is routed towards that specific destination. Alternatively, it could be a specification of a node "around" which the destination type should be explored by the system, and the most preferred one chosen automatically (e.g., gas station closest to the desired node). Time could be the precise amount of time to elapse while parked at the destination, or it could be an absolute time for departure from that node to its next destination in the trip. Absolute times are useful to encode behaviors that are resilient to delays experienced by traffic due to network conditions.

Despite the complexity, the total vehicle byte size has been kept to under 150 bytes on a 64-bit machine.

2.6 Traffic Light Controllers

Simple traffic light controllers have been modeled that have synchronous light sets that change light colors in synchrony in round robin fashion. More sophisticated schedules can be easily added, without overhaul to the framework. Also, smart controllers such as those that operate based on detection of vehicular presence can also be easily added without violation of the discrete event modeling approach described next.

3 DISCRETE EVENT MODEL

The discrete event modeling approach is based on the simple insight that the most interesting events in the vehicular movement are easily captured by an intersection node process. At each intersection, the algorithm shown in Figure 2 is executed.

3.1 Algorithm

```
Intersection::execute( Event *event )

    switch( event->type )
        case LIGHT_CHANGE:
            Update_Lights();
            break;
        case VEHICLES_UPDATE:
            Reset_NERUTS();
            break;
        case VEHICLE_HANDOFF:
            AddArrival( event->vehicle );
            break;

    Update_InSegments();
    Update_ParkedVehicles();
```

Figure 2: Discrete Event Algorithm Executed by Intersection Node

The execute() method of an intersection node is executed by the discrete event simulation engine whenever there is an event destined for the intersection. Events are time-stamped and are executed in global time-stamp order.

As part of Update_InSegments(), each vehicle on all lanes of the intersection's incoming road segments are advanced by delta time equal to current time stamp minus the last time at which they were updated. In Update_ParkedVehicles(), vehicles parked at that node, if any, are checked for departure readiness.

If and when any vehicles are ready to depart, either from parked queue or from the front of incoming lane segments, then traffic control and congestion effects are incorporated for decision on whether the vehicle will be permitted to leave. For example, the traffic light corresponding to its next hop in its trip is consulted for green/yellow color. Also, availability of room in the lanes of outgoing road segment ahead is verified.

Both updates, to in segments and to parked vehicles, result in the scheduling of a new local VEHICLES_UPDATE event into the future for this intersection. Similarly, next time for light changes are scheduled via a local event of type LIGHT_CHANGE. As an optimization, only the earlier of the two event types VEHICLES_UPDATE and LIGHT_CHANGE is scheduled, since one automatically implies updates for the other.

When a vehicle is ready to depart this intersection, and has met all condition of departure (destination outgoing road segment is not full, light is green, etc.), then a VEHICLE_HANDOFF event is sent to the intersection that owns the outgoing road segment of that vehicle. Upon receiving a VEHICLE_HANDOFF event, the intersection adds the vehicle onto the incoming road segment corresponding to the outgoing road segment on which it was sent. At this point a lane changing algorithm is invoked on the vehicle which currently switches the vehicle to the least congested lane on its road segment (of course, while preserving an inter-vehicular gap).

The earliest departure time for vehicles is computed by simple Newtonian physics calculations which involve solving the quadratic equation on acceleration, velocity and position of the lead vehicle on each lane of each incoming road segment on all road segments incident on an intersection. Care is taken to handle all special cases arising out of dual and/or imaginary roots of the quadratic equation. It is this next earliest required update time that is used to schedule the next VEHICLE_UPDATE event.

3.2 Routing

Routing table computation is clearly a computationally intensive operation, which is incurred at simulation initialization. However, we are adapting the optimizations that have been discovered in the area of large-scale Internet (TCP/IP network) simulations, such as Nix-Vector routing. Moreover, we are using a "ghost-node" approach to representing the entire network at every processor for accurate routing table computation in the absence of shared memory.

3.3 Data Structures

The hierarchy realized in data structures for this is shown in Figure 3. Note that only in-segments are represented in full detail at each intersection (with lanes holding vehicles at different locations). Every in-segment of an intersection has a corresponding out-segment on another intersection. Another protocol is used to synchronize out-segment occupancy information with their origin intersections. This is necessary to accurately capture congestion spanning multiple intersections.

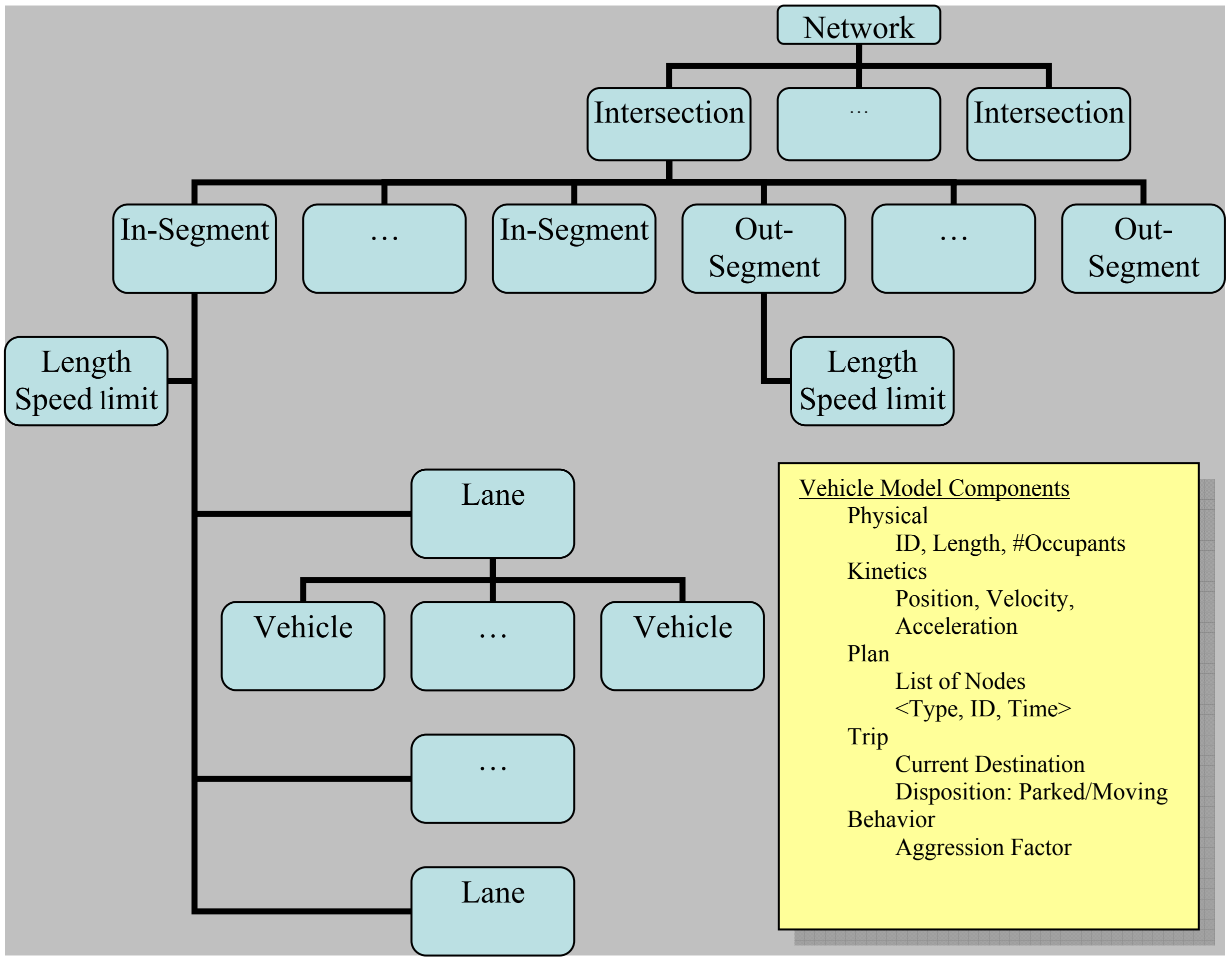

Figure 3: Model Data Structures

3.4 Configuration

Configuration of simulation is performed via a simple configuration file provided as input to SCATTER. The file consists of simple directives such as NODE, LINK and VEHICLES, which are straightforward to encode for any given network topology, including number of lanes, speed limits and geographical coordinates.

4 EXPERIMENTAL STUDY

We now turn to a preliminary performance study for the current implementation of SCATTER. In order to facilitate this study, we first define a simple synthetic benchmark. This is followed by test of simulation speed. Scalability of the simulator with increasing number of vehicles is examined, followed by initial report on parallel execution performance.

4.1 Benchmark

Our synthetic benchmark road network consists of road segments enclosing a grid of *n* x *n* blocks as shown in Figure 4. Each intersection has a single-lane road segment into the interior of the block, nominally representing lanes into parking lots in that grid element, as shown in close-up view in Figure 5. Each main road segment has two lanes, and there is a traffic light controller at every intersection, as illustrated in a zoom-in picture of an intersection in Figure 6. The perimeter road segments on the four boundaries have a slightly higher speed limit than the interior roads. For benchmarking a parallel execution, *m* instances of this grid are created and placed in a circular pattern, and the four corners of each instance are connected to adjacent (left and right) neighbor's corners. Vehicles are created, *v* per grid element (i.e., *v* vehicles parked in the parking lot of each grid location). Vehicles' destinations are picked randomly from all available blocks. The pictures of the benchmark network are generated as screenshots from the SCATTER visualization and animation tool.

Although we have access to more realistic road network topologies, we chose this benchmark as it represents a more controlled scenario whose properties are easier to understand. However, we also plan to perform benchmark studies on realistic networks shortly, after establishing the scalability of speed and size on synthetic benchmarks.

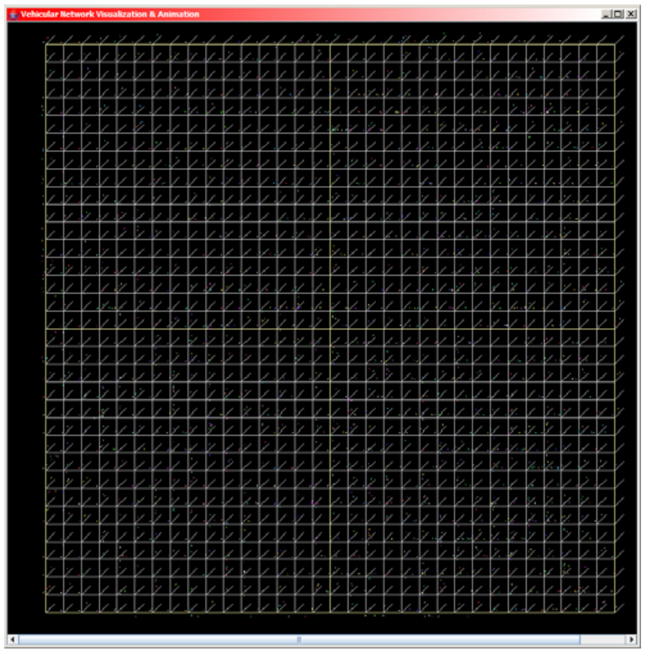

Figure 4: Benchmark Road Network with an *n* x *n* Grid

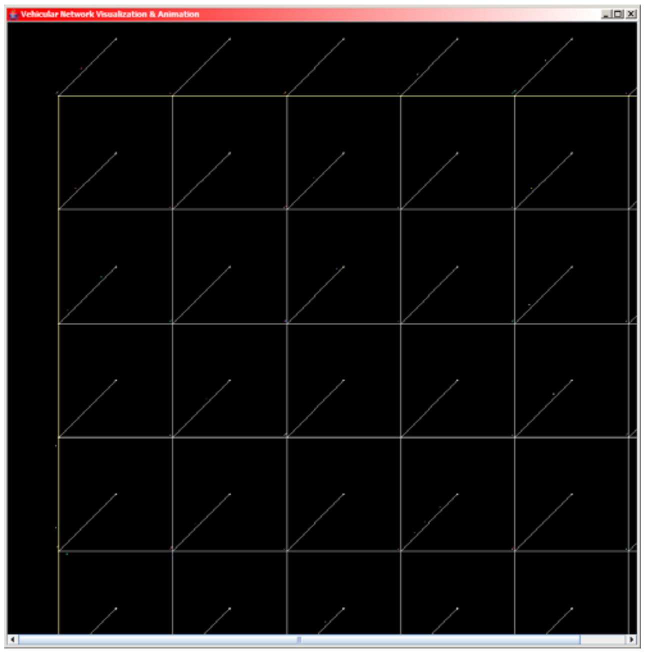

Figure 5: Close-Up View of a Few Grid Elements of the Benchmark Road Network

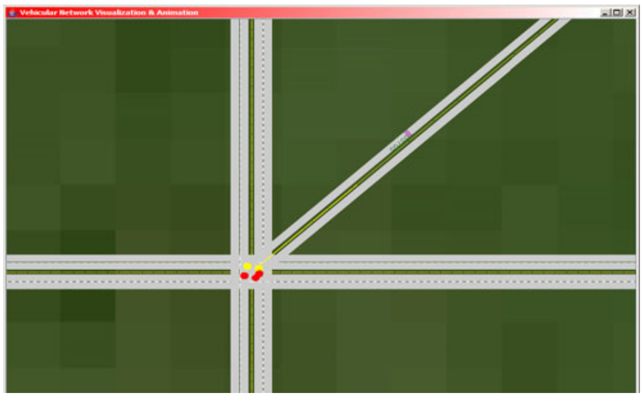

Figure 6: Close-up View of Streets at an Intersection of the Benchmark Road Network

4.2 Computation Platform

All experiments were executed on a cluster of nodes, each node containing dual Intel 3.4GHz Xeon EM64T processors, 4GB of memory and dual Gigabit Ethernet interconnects.

4.3 Performance

The number of microseconds taken by each event is tracked with varying number of vehicles introduced on two configurations of the benchmark network. One configuration uses *n*=3 (which gives 9 intersection nodes), and the other configuration uses *n*=33 (which gives 1089 intersection nodes). The number of vehicles *v* per node is varied from 1 to 1000, which gives up to 1 million vehicles total on the 1089 node configuration.

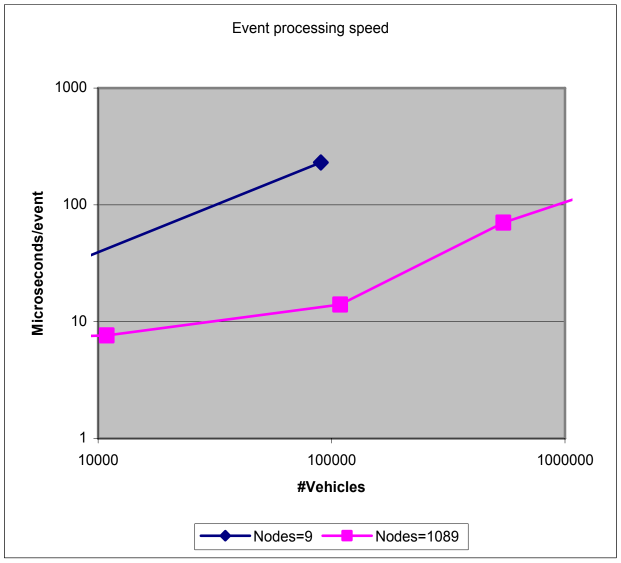

Figure 7: Event Processing Speed with Increasing Number of Vehicles on the Benchmark Road Networks

It can be seen from Figure 7 that event processing computation time is very low, and scales well even up to 1 million vehicles (on a single processor). The event processing stays in the 100 microsecond range even in the largest case.

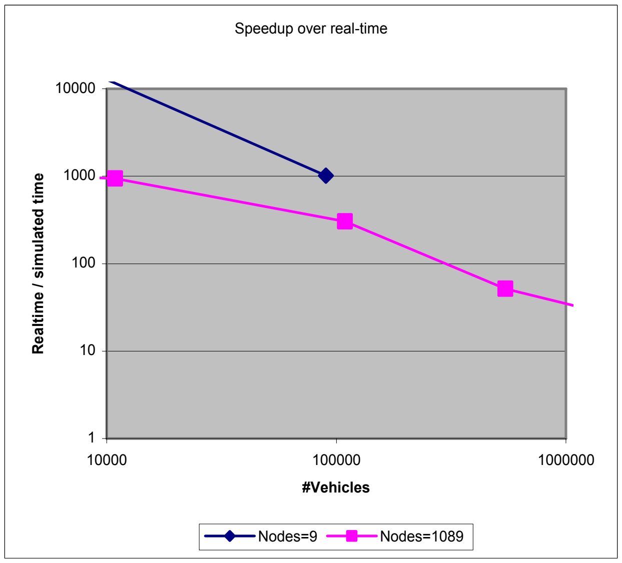

Figure 8: Speedup Over Real-Time as the Number of Vehicles on the Benchmark Road Networks is Increased

An important metric is the amount of speedup afforded by the simulation relative to real-time. A factor of $r > 1$ says r predictive simulation runs of 1-day can be explored in one day of wallclock time. Clearly, the higher the value of r, the more desirable it is, to be able to explore many alternative planning/emergency scenarios. This is tracked in Figure 8.

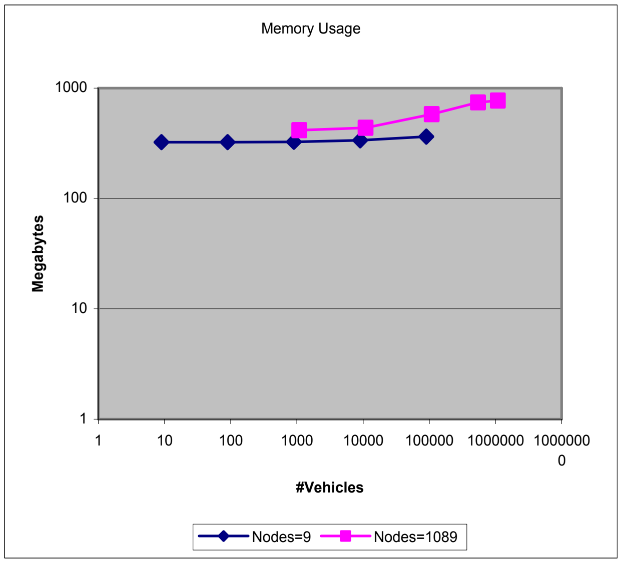

Figure 9: Memory Usage as The Number of Vehicles On the Benchmark Road Networks is Increased

Memory usage is kept low by design, as shown in Figure 9. Even with 64-bit computing overheads, each vehicle is represented in 144 bytes, each intersection is represented in 630 bytes and in/out segments consume less than 100 bytes each. The memory for each intersection is relatively high because of the overhead of approximately 500 bytes imposed by the underlying parallel simulation engine for each logical process (recall that each intersection is realized as one logical process).

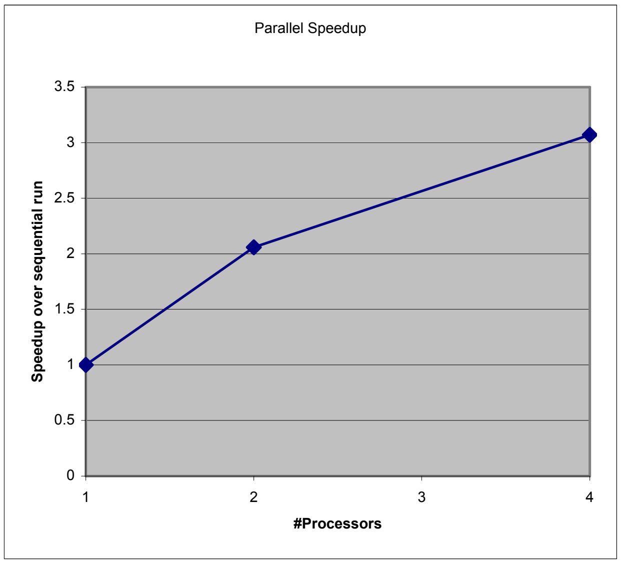

Figure 10: Speedup Due to Parallel Execution

Parallel execution is enabled via simple directives in the configuration file, to specify a mapping from intersection identifier to a processor identifier (integer to integer). The performance improvement from parallel execution is shown in Figure 10 for a benchmark test case of n=15 and m=4.

5 CONCLUSIONS

Efficient simulation of vehicular traffic on large spatial scenarios is an emerging need. Parallel simulation is needed to best meet the geographical scales and real-time speeds warranted by related applications. To the best of our knowledge, SCATTER is one of the few transportation simulators that is discrete-event based and also enabled for parallel execution. The SCATTER simulator currently runs on cluster of workstations platform connected by network, and does not rely on availability of shared-memory across processors.

5.1 Future Work

We are incorporating efficient routing algorithms for dynamic traffic behavior (change of individual routes based on congestion levels and/or events). While full or incremental re-computation of shortest path routes seem to be applicable, that is not necessarily an accurate representation of real life operation, since not every driver is aware of entire network state instantaneously. We are exploring more representative models such as localized self-rerouting. Also, efficient geographical abstract routing is being explored, based on the observation that road networks are clustered in terms of towns and cities, with sparse connectivity across the clusters. We are exploiting this structure by computing routes only on demand, and cache the computed routes. This approach avoids computing shortest paths between all pairs of intersections, and instead computes only those routes that are actually invoked in the particular simulated scenario.

Benchmarking on more realistic scenarios based on actual highway topologies is underway. SCATTER is also being ported to supercomputing platforms. Being built on the scalable μsik(Perumalla 2005) system is making this port easier. μsik supports a micro-kernel framework that scales well the with number of logical processes (LPs), the number events exchanged by the LPs and with the number of processors. Virtual time synchronization is realized using a scalable asynchronous algorithm(Perumalla and Fujimoto 2001) that provides rapid time advances. In some of the largest configurations, up to one million LPs are tested to exchange a 1-billion event population. Since intersections are mapped to LPs and vehicle transfers are mapped to events, μsik's scalability could potentially be exploited by SCATTER.

ACKNOWLEDGEMENTS

This paper has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. Accordingly, the United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

The author gratefully acknowledges insights from Dr. Budhendra Bhaduri (ORNL) into the needs of emergency scenarios.

REFERENCES

Bhaduri, B., C. Liu and O. Franzese, "Oak Ridge evacuation modeling system (OREMS): A PC-based computer tool for emergency evacuation planning," *Symposium on GIS for Transportation*, 2006.

Cameron, G. D. B. and G. I. D. Duncan, "PARAMICS, parallel microscopic simulation of road traffic," *Journal of Supercomputing*, vol. 10(1), pp. 25-53, 1996.

Franzese, O. and L. Han, "A methodology for the assessment of traffic management strategies for large-scale emergency evacuations," 11th Annual Meeting of ITS America, 2001.

Fujimoto, R. M., "Parallel discrete event simulation," *Communications of the ACM*, vol. 33(10), pp. 30-53, 1990.

Fujimoto, R. M. and J. Leonard II, "Grand challenges in modeling and simulating urban transportation systems," in *Proceedings of the First International Conference on Grand Challenges for Modeling and Simulation*, January ed. January, 2002, 2002.

VISSIM Simulation Tool, Innovative Transportation Concepts, I., 2001, <http://www.itc-world.com/VISSIMinfo.htm>.

ITT Systems & Sciences Corporation, "CORSIM user's manual, version 1.04," Federal Highway Administration, U.S. Department of Transportation 1998.

Perumalla, K. and R. Fujimoto, "Virtual time synchronization over unreliable network transport," Workshop on Parallel and Distributed Simulation, 2001.

Perumalla, K. S., "µsik - A micro-kernel for parallel/distributed simulation systems," Workshop on Principles of Advanced and Distributed Simulation, 2005.

Perumalla, K. S. and B. Bhaduri, "On accounting for the interplay of kinetic and non-kinetic aspects in population mobility models," European Modeling and Simulation Symposium, 2006.

Prevedouros, P. D. and Y. Wang, "Simulation of large G=freeway and arterial network with CORSIM, INTEGRATION, and WATSIM," *Transportation Research Record*(1678), pp. 197-207, 1999.

Smith, L., R. Beckman, D. Anson, K. Nagel, and M. E. Williams, "TRANSIMS: Transportation analysis and simulation system," in *Proceedings of the Fifth National Conference on Transportation Planning Methods*. Seattle, Washington: Transportation Research Board, 1995.

AUTHOR BIOGRAPHIES

KALYAN S. PERUMALLA is a senior researcher in the Computational Sciences and Engineering Division at the Oak Ridge National Laboratory. He also holds an adjunct faculty appointment with the College of Computing, Georgia Tech. He received a Ph.D. in Computer Science from Georgia Tech (1999). Dr. Perumalla has over 10 years of research and development experience in the area of parallel and distributed simulation systems, including high-performance runtime infrastructures and large-scale simulation, and has published widely on these topics. He co-developed the Federated Simulations Development Kit (FDK), a widely-disseminated high-performance runtime infrastructure for HLA-like distributed simulator federations. He has also built several additional research prototype systems and tools (e.g., for distributed debugging, network modeling, interoperable simulations and parallel optimization), most of which are in use by researchers worldwide. He has served as co-PI on multiple federally-funded projects on scalable parallel/distributed discrete event simulation systems.. He can be reached via email at <perumallaks@ornl.gov>, and his homepage is at <www.ornl.gov/~2ip>.