Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

MULTI-MODAL TRAFFIC SIMULATION PLATFORM ON PARALLEL AND DISTRIBUTED SYSTEMS

Toyotaro Suzumura

IBM Research
Smarter Cities Technology Center
Damastown Industrial Estate
Mulhuddart Dublin 15, IRELAND

Hiroki Kanezashi

Graduate School of Information Science and Engineering, Tokyo Institute of Technology
2-12-1 Oo-okayama
Meguro Tokyo, JAPAN

ABSTRACT

In this paper we describe a highly scalable multi-modal traffic simulation platform on parallel and distributed environments ranging from multi-core or many core machines to multi-node clusters or supercomputer environments. We evaluated our platform with multi-modal transportation networks from the capital city of Ireland, Dublin, and verified its high scalability and real-time performance on both a 1-node with 12 core machine for a multi-threading environment and a cluster of 12 nodes (with a total of 144 cores) for a distributed system. With this platform, municipal planners or traffic engineers at transportation companies can quickly conduct many series of simulations with various what-if scenarios in a parallel and distributed manner, allowing them to assess and select the best responses to sudden incidents, or plan for major events or disaster responses.

1 INTRODUCTION

In recent years, many researchers and practitioners from various backgrounds have focused on how to make cities smarter, more efficient, and more ecologically friendly, so as to become smarter cities. Many problems must be addressed, such as transportation, energy, public safety, and healthcare problems.

Better transportation is one of the classic and important problems in making cities smarter. Since more people tend to live in the center of a city, major traffic jams are caused by private cars and public transportation systems such as trains, subways, or buses, are increasingly overloaded.

We need more sophisticated transportation systems that can reduce traffic congestion by considering various constraints such as government policies, budgets, and transportation company revenues without sacrificing the convenience of individuals and wasting their money and time. Each of the components of a transportation system is not independent but sometimes involve trade-off relationships, so simulations have many non-deterministic and complex features.

At the same time, more and more data relevant to solving these transportation problems is becoming publicly available thanks to the trend of "open data" involving various cities. For instance, the city of Dublin in Ireland publishes a large amount of data via the website Dublinked.ie (Dublinked 2012). Private mobile carriers also have a rich data set of call-detail records that track human behaviors. MIT and some telecom companies sponsored a competition called the "D4D Challenge" in 2013 in which participants analyzed anonymized call-detail records for various purposes such as human behavior analysis that could improve transportation systems (Berlingerio et al. 2013).

At the same time, low-level simulation at the municipal or national level has become possible. In our earlier work (Suzumura and Kanezashi 2012, Suzumura and Kanezashi 2013), we described highly scalable and real-time traffic simulation for private cars. The next obvious step would be to incorporate various transportation systems beyond private cars, such as public buses, trains, and so forth. Although multi-modal transportation simulation platforms are available, such as MATSim (Balmer et al. 2009),

VISSIM (Fellendorf and Vortisch 2010), and SUMO (German Aerospace Center 2014), there are no published reports describing parallel and distributed low-level multi-agent-based simulation platforms that support multi-modal transportation systems at the municipal or national level.

In this paper we describe a highly scalable multi-modal traffic simulation platform for parallel and distributed environments ranging from multiple cores or many-core machines to multi-node clusters or supercomputer environments. We evaluated our platform with municipal data from Dublin, Ireland and demonstrated highly scalable and real-time performance. With such a platform, decision makers in municipal government or employees of public transportation companies can simulate various what-if scenarios in a parallel and distributed manner, allowing them to quickly assess and select the best responses for sudden incidents, perform city planning for major events such as the Olympics or the World Cup, or handle contingency planning for disasters such as tsunami or hurricanes.

This paper is organized in 6 sections. Section 2 is a system overview of our scalable multi-modal traffic simulation platform called M3, and also describes the agent models. Section 3 reviews the underlying agent simulation platform called XAXIS. Then Section 4 shows the performance characteristics on various environments with various multi-core and multi-node systems. Section 5 reviews relevant work, mainly focusing on multi-modal traffic simulations, and Section 6 describes possible future work and our conclusions.

2 M3: MULTI-MODAL TRAFFIC SIMULATION

This section describes the system design and implementation of our proposed scalable multi-modal traffic simulate platform called M3 and also the agent behavior models.

2.1 System Overview

We are working on a high-performance multi-modal traffic simulation platform called M3 for parallel and distributed systems. We can simulate a wide variety of transportation modes including public buses, trains, or private cars on a municipal or national scale.

The overall system architecture is depicted in Figure 1. The simulator is based on the IBM Mega-Traffic Simulator called *Megaffic* (Osogami et al. 2013) for private cars. As shown in the figure, the overall system is divided into 3 components, the simulation input data set, the simulation platform itself, and the simulation output. For the simulation input, a series of trips or individual travel trajectories are provided or generated by a modular external component called a journey planner. The input for the journey planner could be coarse-grained traffic flows from certain part of the city to other parts that are often provided by census data. If the OD (Origin-Destination) data includes home and work locations, then our journey planner computes the path or trajectory from the origin to the destination. Our current implementation uses basic functions to compute the routes that minimize the total distance or total time with specified transportation modes such as only using public transportation systems or only using private cars. For the input data, we need networks for multi-modal simulations, such as public buses, trains, and private cars. The timetables for public transportation are also crucial data. The simulation platform reads the input data including the routes, networks, and timetables for public transportation, and then moves all the agents representing the people, private cars, and public transportation vehicles according to their behavior models.

The simulation platform outputs low-level data for each simulation step, which includes all of each agents' properties such as location, status, speed, and number of passengers for public transportation. By analyzing the low-level data, it is be possible to obtain higher-level results such as the length of each traffic jam, the average travel time, CO_2 emissions, and so forth. To support a wide variety of transportation modes with a special focus on public buses and trains, we initially considered how to represent networks that combines roads, bus routes, and trains. Then we designed behavior models for the passengers, buses, and trains. In the subsequent sections, we describe those various perspectives, beginning with the implementation of our first-stage journey planner.

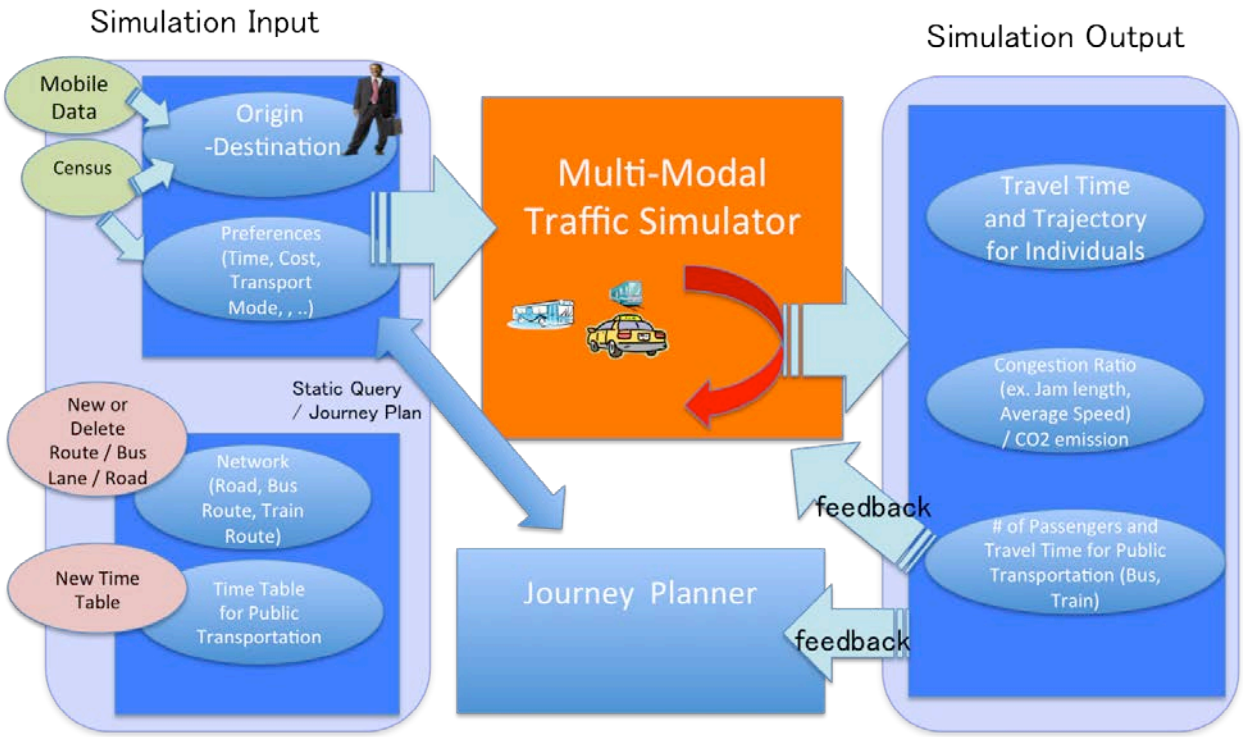

Figure 1: System Architecture of M3.

2.2 Network Representation

A network for a multi-modal traffic simulation is designed as a combination of a road network, bus network, and train network. In this research, we first extracted the road network for the city of Dublin in Ireland from the Open Street Map (OSM), and converted the OSM format to our road network representation in a CSV format.

Next, the bus routes were extracted from the city's open data website (http://dublinked.ie/), as published by the Dublin city council and the National University of Ireland, Maynooth. The data format was GTFS (General Transit Feed Specification), which contains various kinds of information such as the bus or train network and the fare data. The two files named "stops.txt" and "stop_times.txt" are needed to construct a bus network. The file "stops.txt" has information on each bus stop ID and its latitude and longitude. M3 has an integration tool to search for the nearest match with the road network, and then a new edge (representing the road) is added between that bus stop and nearest matching point.

A train network is added to M3's network structure in the same was as a bus network. In Dublin, the public train is a tram called "LUAS", and that data set is also available.

2.3 Agent Behavior Model

In the original Megaffic, a private car is a first-class citizen but in M3, "Person" is a first-class citizen. Each person agent is linked to certain transport agents, either a private vehicle, a public bus, a public train, a bicycle, or acts as a pedestrian. The bicycle vehicles are not yet supported. The current behavior model of all of the vehicles follows the car model and uses the lane-changing model used by cars.

A) **Person**: Person agents are generated when the simulation reaches a starting time for each person. When a person is generated, the Journey planner starts to create a JourneyPlan for that person.
B) **Private Car**: When a Person starts driving a car, a Vehicle object will be created. It is deleted when it reaches its destination. The number of vehicles is recorded for the aggregated statistical analysis after the simulation.
C) **Pedestrian**: When a Person is walking, a Pedestrian object moves that Person object. The object is created when the Person starts walking, and is deleted when it reaches the destination.
D) **Public Train**: A public train line has a set of train stations, each of which has public timetables. In the normal mode, each train runs with a relatively unchanging pattern of speeds and movements to the next station by the time specified in the timetable. Unlike private cars, no interference with other trains is involved in normal conditions. In a special mode, due to some incident (e.g. a snowstorm, suicide, accident, etc), most of the trains are delayed or temporarily stopped. To understand the effects on the municipal transportation system, the timetables can be updated or some scheduled trains may be cancelled. During what-if scenario analysis, new train schedule and train route can be simulated. For planning purposes new train stations can be simulated. This allows testing new policies and their effects on the larger municipal transportation system.
E) **Public Bus:** A public bus behaves like a private car except that it runs in a bus dedicated lane

Public transportation (Bus and Train) objects are created when the simulation reaches the departure time of the objects based on the timetable. At each bus stop or train station, there is a passenger queue that holds a list of the people waiting for the arriving buses or trains. Each bus or train has a limit on how many people can board. This limit is set in the configuration file of each bus or train. In our prototype simulator, a public bus can hold 100 people, while a public train can hold 200 people. If the number of passengers is over the limit, then the extra people waiting in the queue cannot board that bus or train and must wait for the next one.

Speed model between adjacent stations or bus stops: currently the same as the "car model" used for private vehicles, but they start and stop depending on the given timetables. The road network for buses is currently treated as independent from the private cars network, even though they actually use the same roads. In contrast to cars and buses, the trains exist independently and they are not deleted until the entire simulation is finished.

2.4 Journey Planner

A journey planner computes a trajectory route (or trip) and transport modes for individual persons based on their origin and destination locations, preferred transport modes, and an optimization factor. The "JourneyPlanner" (JP) component is illustrated in Figure 1. JP first reads an input file that contains a set of OD pairs, departure times, preferred transportation modes, and optimization factors (to specify whether to minimize the total time or the total distance for each trip).

JP retrieves all of the road networks and public transportation networks with two weight values on each edge: its length and speed limit. JP uses either the length or estimated transit time for each road segment to compute the shortest paths using the Dijkstra Method. Currently, there are five types of preferences for transportation modes:

1. Mixed transportation minimizing the total distance. The weights of all of the roads including the bus lanes and trains are equal to the lengths of the segments.
2. Only public transportation minimizing the total distance: The weights of all of the roads for private cars are equal to their length, but the roads for public transport systems are very small (1e-20) so that people use public transport as much as they can.

3. Only private cars minimizing the total distance: The weights of all of the roads for public transport system are equal to their lengths, but the roads for private cars are very small. This causes the simulated people to use their cars or walk as pedestrians.
4. Mixed transportation minimizing the total time: (encoded as Preference 1). The weights of all roads including bus lanes and railroads are equal to their estimated transit time. The transit time of each road is calculated using the ratio of the length to the speed limit. The total transit time is the sum for al of the roads on a route, so waiting times for transfers are ignored.
5. Only public transportation minimizing the total time is preferred (encoded as Preference 2). The weights of all of the roads for private cars are equal to their transit times, but the roads used only for public transport are set very small (1e-20).

The input files for JP are (1) a design file that contains the initial OD data sets, departure times, preferred transportation modes, etc., (2) a multi-modal network with weights, and (3) the timetables for the public transportation networks, as shown in Figure 2.

Figure 2 includes the overall flow showing how a journey plan is generated. To generate the journey plans, JP runs several procedures: (1) Remove the OD pairs from a design file if intermediate checkpoints are specified, (2) Create the multi-modal network and timetable for the public transportation, (3) Calculate the shortest paths, and finally (4) Write the computed paths to the individual journey plan files.

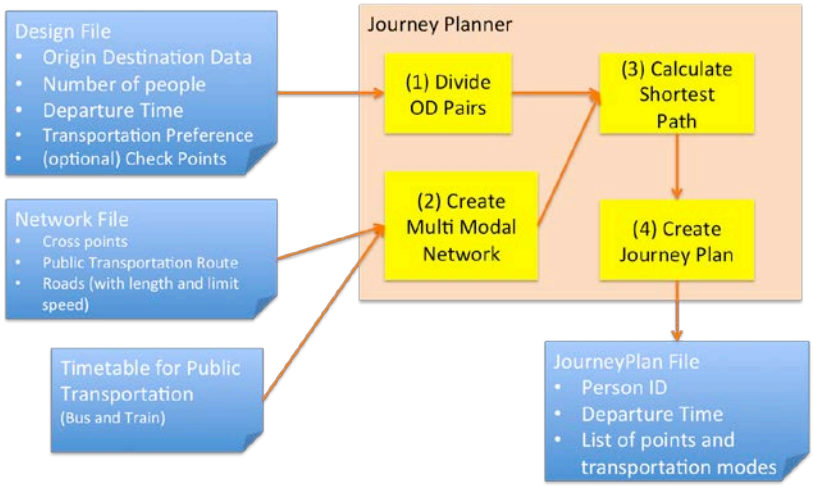

Figure 2: Data Flow of Journey Planner.

Users can set not only each origin and destination (OD) pair, but also specify intermediate for an itinerary in the journey design file. Here is a more detailed explanation of each of the four procedures:

1. First, JP considers OD and checkpoints as sets of OD pairs without checkpoints, and calculates the shortest path for each pair.
2. The graph network is created from the network.csv file. Each edge (road) in this network has two weights and a type: the numbers are a length and an estimated passage time, and the type is for public or personal transport. The timetable file is also needed to specify which roads are used for public transport based on the timetable data.
3. JP calculates the shortest paths for each set of divided OD pairs using Dijkstra's algorithm so that people can use routes with the smallest total weight. The weights of the edges depend on the transport preferences. When no shortest paths are found for some of OD pairs, then there are no shortest paths of the original OD, the creation of the JP fails and it discards the rest of path. If all

of the calculations succeed, then each of the shortest paths is linked together to form the complete path and the forms of transportation are defined from the network.

4. After creation of the journey plan, it is written to a journey plan file.

The current prototype considers only the distance, but the journey planning component could be extended to consider such goals as minimizing the total cost, the total time, the total number of changes or other criteria, as well as considering users' preferences (such as for public transportation). Many prior work focuses on optimizing journey planning as a dynamic or static traffic assignment problem, so our future work may involve tests of various journey planning strategies by replacing this component in M3. We also want a capability to invoke the journey planning component dynamically during a simulation.

3 XAXIS: SCALABLE X10-BASED AGENT SIMULATION MIDDLEWARE

M3 is built on top of XAXIS, our X10-based distributed agent processing platform. XAXIS (X10-based Agent eXecutive Infrastructure for Simulation) (Suzumura and Kanezashi 2012, Suzumura and Kanezashi 2013) is a highly scalable multi-agent simulation middleware that allows application developers to run their applications productively on large-scale environments. The programming model of XAXIS was derived from the Java-based agent simulation middleware called IBM Zonal Agent-based Simulation Environment (ZASE). This section gives an overview of the agent model, its software stack, and the implementation model of XAXIS.

3.1 XAXIS Agent Model

The XAXIS stimulation model is illustrated in Figure 4. The details of XAXIS are described in our prior arts (Suzumura and Kanezashi 2012, Suzumura and Kanezashi 2013). In XAXIS each agent has internal states, handles messages, and updates its own states as needed. Messages are sent by a simulator or by other agents. When a message object is delivered to an agent, a callback method of the agent is called. A returned message object will be sent back to the sender. The XAXIS framework provides functions to create and to delete agents. Developers can add services to an agent runtime. Agents can search for the reference to a service and can directly invoke the methods of services. Each service object also has functions similar to those of agents. Messages are exchanged among agents, simulation runtimes, and agent runtimes. XAXIS provides point-to-point messaging and multicast messaging. XAXIS also provides message distribution and aggregation functions. When a simulator sends a multicast message to an agent runtime, the agent runtime will distribute the message to its agents. The agent runtime aggregates the reply messages from the agents into an aggregated message and sends that message to a simulator.

3.2 XAXIS Software Stack

The full software stack of XAXIS is illustrated in Figure 2 (Suzumura and Kanezashi 2012, Suzumura and Kanezashi 2013). An advantage of adopting X10 is that the XAXIS APIs can be implemented in a highly productive manner without implementing thread management or a messaging layer for the distributed environments, since the X10 language itself includes those functions at the language level. The software stack in Figure 2 includes two bridges, a Java bridge and an X10 bridge, depending on whether Java or X10 is being used. The Java-based bridge is for Java-based simulations on XAXIS. This is because Java is still more of a commodity language than X10 when developers implement simulations. However, we also provide an X10-based bridge so that the full software stack can be implemented entirely in X10 and compiled into an X10 C++ backend with high performance communication libraries such as MPI for developers who want to implement their simulations directly in X10.

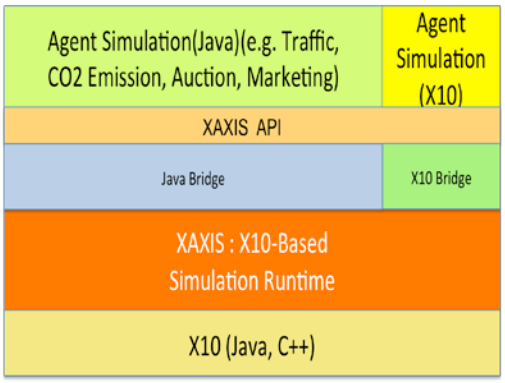

Figure 3: XAXIS Software Stack.

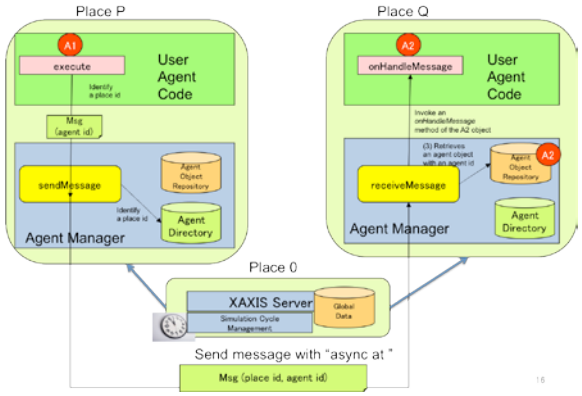

Figure 4: Simulation Model with X10.

4 PERFORMANCE EVALUATION

In this section we show the performance characteristics of our proposed platform on various environments with various numbers of multi-cores and multi-nodes.

4.1 Evaluation Environment

As the underlying execution platform, the parallel and distributed programming language called X10 (version 2.3.1) is used. We use Managed X10, which compiles X10 source code into Java source code, and then run Java bytecode on a Java virtual machine. For the Java virtual machine, we used Java 1.7.0_09 (HotSpot) with 48 GB as the maximum heap size, 24 GB as the minimum heap size, and 256 MB for the stack size. For the multi-node environment, we used from 1 node up to 12 nodes interconnected with Infiniband. The 1-node system has dual 6-core Intel Xeon CPUs for a total of 12 cores. To spawn off an equal number of CPU cores, the X10_NTHREADS environmental variable was set to 12.

4.2 Experimental Data Set

We selected the city of Dublin in Ireland for our experiments. Dublin has two main public transportation systems: public buses (primarily Dublin Bus) and a light-rail tram (LUAS). The maximum number of passengers for each double-decker bus is around 100. Each LUAS tram has a maximum of approximately 300 passengers. The Dublinked website (Callan Institute National University of Ireland Maynooth 2014) provides GTFS data for LUAS, including the tram network and timetables. The LUAS system has two lines: the Green Line and the Red Line, with a total of 54 stations as of January 2014.

According to the LUAS Wikipedia pages, the cost of building the original Red and Green Lines was €728 million. It was envisaged in the original plans that the Green Line would intersect the Red Line on O'Connell Street. However two unconnected lines were built, with a 15-minute walk at the closest point between the two lines. A new BX Line is now under construction to link the two lines. This kind of municipal planning is one of the what-if scenarios that M3 can support.

We extracted two road networks from OSM (Open Street Map), “Dublin All” and “Dublin Central”. “Dublin All” covers a wide area of Dublin with around 60,000 cross points. “Dublin Central” covers the city centre area of Dublin with around 17,000 cross points.

4.3 Generating the Trip Data Set

We used the trip generator function of the Journey Planner component described in Section 2.4 for testing a simulation assuming that only macro-level traffic flow data is available, such as census data. This generator can be used for predicting the approximate traffic conditions or used for performance

evaluations, as described in this section. The trip generator is a component that generates a trip or trajectory for an individual person. The system works best with fine-grained OD data for highly accurate simulations. The simulation results can also be validated with the observed sensor data, such as SCATS (Sydney Coordinated Adaptive Traffic System), which has been installed in Dublin. However it is easier for users or practitioners to use macro-level OD patterns in a grid-style as shown in Figure 5. Users can also specify an exact cross point identifier directly without starting from an "@" mark.

First, the entire road network is decomposed into a set of grids, and then prepared for a file that describes the total traffic volume between adjacent grid cells for the total number of simulation steps. The trip generator randomly picks points as origins or destinations, and then computes a shortest path for each pair. Each path can be computed in various ways such as minimizing the distance, minimizing the cost, minimizing the number of changes, and so on. Currently the trip generator picks the nearest stop of any public transportation (bus or train), and uses public transport as much as possible.

The network was equally divided into 0.05 x 0.05 degree squares for both longitude and latitude. Each cell of the network was numbered starting from 0 at the bottom-left (southwest) corner. Figure 6 is the visualization of a typical simulation result. We retrieved the base map picture from the Open Street Map website, and added several colors for the lines and dots on the map. White lines represent public road where private vehicles as well as buses operate, blue lines represent bus lanes, and green lines are the routes of LUAS. The violet dots represent bus stops or tram stations and red dots are people.

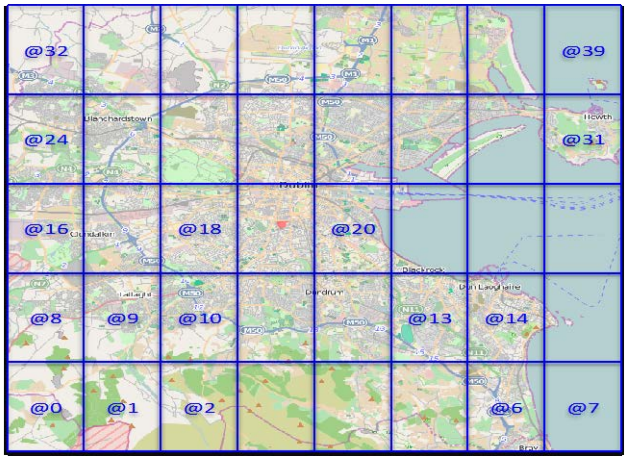

Figure 5: Cells and Their Numbers Divided from the "Dublin All" Network.

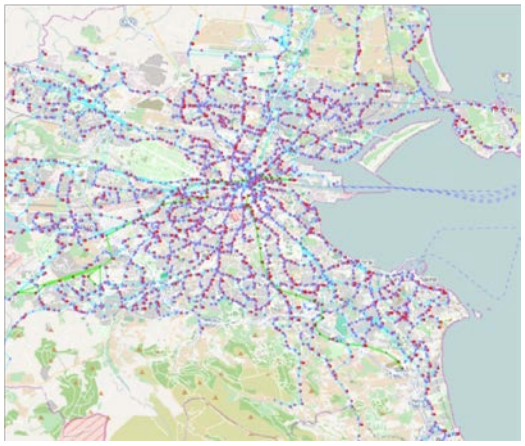

Figure 6: Visualizing Simulation Results for Dublin.

4.4 Performance with varying number of people

Our experiments show the differences in the performance of the transportation systems with different numbers of people using the 2 networks, the greater Dublin area labeled "ALL" in the graph, and the central area labeled "Central". From our experiment, it is found out that studying the dominant simulation performance factors, such as network size or the number of people in transit.

Our results show that the simulation performance is not bounded by the number of people but by the network size. For example, if the movements of 70,000 people are simulated in the two networks, for 100 simulation steps it takes around 140 seconds for the greater Dublin area and around 70 seconds for the Dublin center area. Each simulation step simulates approximately 100 seconds, so the 100 steps simulates approximately 10,000 seconds (approximately 2 hours and 50 minutes). However, if the number of people is set larger, the performance might be bounded by the number of people rather than the network size.

Right now 70,000 people is the largest number of people that the current prototype can handle before running out of memory. In these experiments, 1 node with 12 threads was used for the simulations.

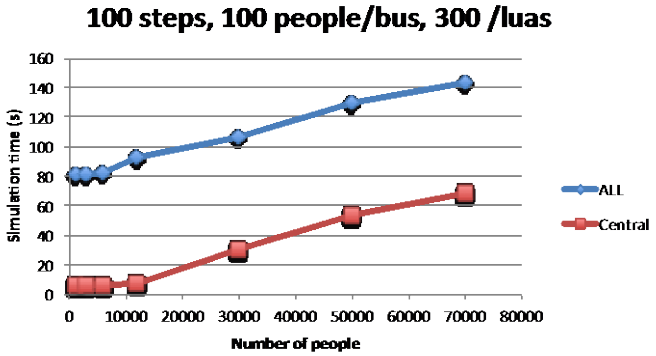

Figure 7: Simulation times as a function of the number of people.

4.5 Performance Characteristics on Multi-Core Environments

Hardware: 12 cores (Intel Xeon 2.93-GHz, 54 GB of RAM, SUSE Linux Enterprise Server) The number of people simulated was 50,000, all of the people are simulated and depart simultaneously at the first simulation step. The number of steps is 10,800 (4.63 trips per simulation step), corresponding to approximately 3 hours of time in the real world. The results of simulation speed was 0.1 seconds / step at maximum (Cf. 0.04 seconds / step for Hiroshima).

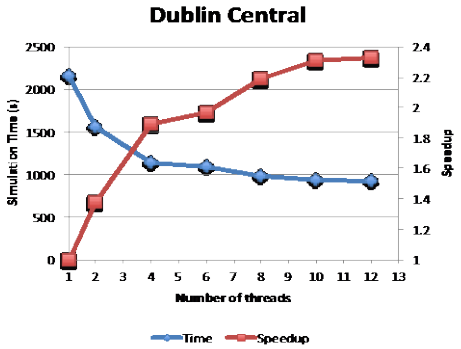

Figure 8: Results of simulations as a function of the number of cores for "Dublin Central".

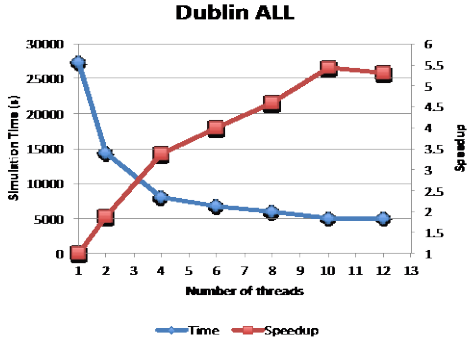

Figure 9: Results of simulations as a function of the number of cores for "Dublin All".

The workload is relatively light since around 5 people start at each step. With a small network in the central area of Dublin (Dublin Central), the performance acceleration relative to 1 core is 2.3 times at maximum. The performance improvement is saturated around 4 cores.

Based on our previous performance evaluation, the performance is bounded by the number of cross points rather than the number of people. For the small network data, Dublin (Central) with 17,000 cross points, 6 cores is not really a sufficient number. If 12 cores are used for this small network, the number of cross points per CPU core is around 1,400 cross points, which is too small a workload. With such small workloads other factors such as the synchronization become the bottlenecks at that scale.

For the large network of the greater Dublin area, 50,000 trips for 600,000 cross points were simulated. For these workloads, the workload per CPU core is much sparser than for Dublin (Central). Since there are relatively few trips compared to the large number of cross points, some cross points do not

handle any vehicles. However, as previously mentioned, the performance is bounded by the number of cross points. This means the number of cross points per CPU core for 12 cores is 50,000 cross points per CPU core, which is a sufficient workload for one CPU core. This is the reason why the large network shows more than a fivefold speed-up compared to a single CPU core.

The results from this experiment show that if the network is small (roughly 5,000 cross points per CPU core), then the CPU utilization remains low and the multi-core environment can be exploited effectively. This means that if the target city network is small enough, the number of threads can be set to a smaller value, such as 2 threads, and multiple scenarios can be simulated in parallel.

4.6 Performance Results on Multi-Node Environment

For the performance evaluation on the multi-node environment, a road network for Dublin based on the Open Street Map data was used. The network originally has 604,599 cross points and 259,224 roads within the area whose longitude ranges from -6.45 to -6.05 and from 53.2 to 53.45 in north latitude. The network is partitioned into sub-networks, each of which is assigned to a compute node and processed in parallel. We are using a high performance graph partitioning tool named METIS (Karypis and Kumar 1998), a tool which assumes that the input network is a strongly connected component. Therefore we first implemented a Java program that converts the original network to a strongly connected network. In the converted network, the road network was comprised of 92,465 cross points and 191,210 networks, after removing the non-connected and isolated cross points. Each sub-network used by METIS has almost equal number of cross points.

For our evaluation, 1,000,000 trips were generated, where each trip is defined as a series of cross points and transport modes used by an individual person. We tested 100 time steps on different numbers of compute nodes up to 16 nodes while changing the ratio of people who only use public transportation, with percentages of 0%, 25%, 50%, 75% and 100%. The remaining users use only private cars.

Figure 10 shows the elapsed time on the left-side Y-axis and the accelerations relative to 1 node on the right-side Y-axis. The label of each line indicates the percentage of people using only public transportation and the percentage of people using only private vehicles. For example, the label "25_75 time" means that 25% of people use only public transportation, while 75% of the people use only private cars. As shown in the figure, there are performance improvements with up to 12 nodes, although the improvements differ depending on the ratios of transportation modes. The maximum improvements are around 3 times with 12 nodes compared to 1 node. There is still a room for improvement such as by balancing the workload and reducing the migration costs, but these problems will be studied in our future work.

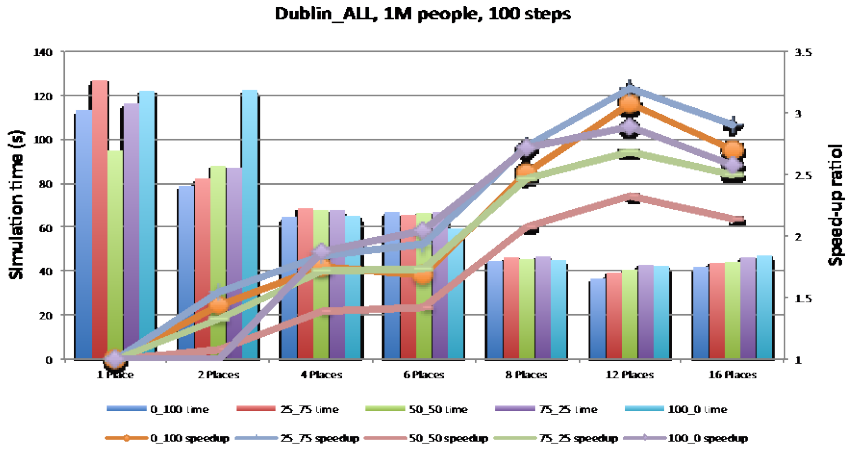

Figure 10: Result of Simulations Varying the Ratio of Vehicles and the Number of X10 Places.

5 RELATED WORK

Several projects have studied multi-modal traffic simulations (Morais and Digiampietri 2009; Naghawi and Wolshon 2012; Kretz et al. 2013). In the MATSim project, the authors of (Balmer et al. 2009) conducted a whole-city experiment for Zurich, Switzerland, where the network consists of 60,492 cross points and 24,180 networks, and 181,693 private vehicles were simulated. SUMO (Behrisch et al. 2011) also supports multi-modal traffic simulations and a distributed version of SUMO named *dSUMO* was recently proposed (Bragard et al. 2013). However as far as we know, none of the earlier work has focused on multi-modal traffic simulations on parallel and distributed execution environments with more than 100 CPU cores targeting millions of agents at the municipal or national scale.

6 CONCLUDING REMARKS AND FUTURE WORK

We are working on a highly scalable multi-modal traffic simulation platform for parallel and distributed environments. Our performance evaluation shows that our platforms scales well with an increasing number of CPU cores or with more compute nodes on distributed systems. With 8 nodes and 96 CPU cores, it takes around 150 seconds to conduct 3,600 steps simulation for the entire city of Dublin, and this performance is approximately 20 times faster than a single node 1 node.

As future work, we want to measure the accuracy of the simulation results with real-world sensor data, such as the SCATS data available for Dublin. To do this, we must obtain the currently missing individual-level OD data and feed it into the simulation. This paper is focused on the highly scalable distributed multi-modal transport simulation platform, so the natural next step is to enhance the agent models and journey planning components.

REFERENCES

Balmer, M., M. Rieser, K. Meister, D. Charypar, N. Lefebvre, K. Nagel, and K. W. Axhausen. 2009. "MATSim-T: architecture and simulation times" *Multi-agent Systems for Traffic and Transportation Engineering*, edited by Bazzan, A.L.C., and F. Klügl, Information Science Reference, Hershey, 57-78.

Behrisch, M., L. Bieker, J. Erdmann, and D. Krajzewicz. 2011. "SUMO – Simulation of Urban Mobility, An Overview", *SIMUL 2011, The Third International Conference on Advances in System Simulation*, 55-60.

Berlingerio, M., F. Calabrese, G. D. Lorenzo, R. Nair, F. Pinelli, and M. L. Sbodio. 2013. "AllAboard: a System for Exploring Urban Mobility and Optimizing Public Transport Using Cellphone Data", In *Proceeding of the European Conference, ECML PKDD 2013*, 663-666.

Bragard, Q., A. Ventresque, and L. Murphy. 2013. "dSUMO: Towards a Distributed SUMO." In *The first SUMO User Conference (SUMO2013)*. Berlin, Germany.

Callan Institute National University of Ireland Maynooth. 2014. "Dublinked: Open Data Ireland A Briefing Paper." Accessed April 15, 2014. http://www.dublinked.ie/

Fellendorf, M., and P. Vortisch. 2010. "Fundamentals of Traffic Simulation", Volume 145 of International Series in Operations Research & Management Science, chapter Microscopic Traffic Flow Simulator VISSIM, 63-93. Springer, 2010

German Aerospace Center, Institute of Transportation Systems. 2014. "SUMO - Simulation of Urban Mobility". Accessed April 15. http://sumo-sim.org

Karypis, G., and V. Kumar. 1998. "Multilevel k-way Partitioning Scheme for Irregular Graphs." *Journal of Parallel and Distributed Computing*. 48 (1998) 96-129.

Kretz, T., R. Frédéric, and S. Florian. 2013. "Multimodal Simulation-Based Planning for Pedestrians", 2013, In *Transportation Research Board Annual Meeting 2013 Paper*. 13-1943.

Morais, D. M. G., and L. A. Digiampietri. 2012. "A review about multimodal traffic simulation techniques", In *The Revista de Sistemas de Informação da FSMA*, 10: 2-9.

Naghawi, H., and B. Wolshon. 2012. ”Performance of Traffic Networks during Multimodal Evacuations: Simulation-Based Assessment.” *Nat. Hazards Rev.*, 13(3), 196–204. Technical Papers

OpenStreetMap. 2014. OpenStreetMap Japan. Accessed April 14, 2014. http://openstreetmap.jp/

Osogami, T., T. Imamichi, H. Mizuta, T. Suzumura, and T. Ide. 2013. “Toward simulating entire cities with behavioral models of traffic”, IBM Journal of Research and Development, Vol. 57, Issue 5, 6:1-6:10, 2013.

Osogami, T., T. Imamichi, H. Mizuta, T. Morimura, R. Raymond, T. Suzumura, R. Takahashi, and T. Ide. 2013. “*IBM Mega Traffic Simulator*.” Technical Report, IBM Research - Tokyo

Suzumura, T. and H. Kanezashi. 2012. “Highly Scalable X10-based Agent Simulation Platform and its Application to Large-scale Traffic Simulation” In *IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2012)*, 243-250, Dublin, Ireland. October 25-27.

Suzumura, T. and H. Kanezashi. 2013. "A Holistic Architecture for Super Real-Time Multiagent Simulation Platform", In *Winter Simulation Conference 2013*, 1604-1612

AUTHORS BIOGRAPHIES

TOYOTARO SUZUMURA is a research scientist of IBM Research as well as a visiting associate professor of University College Dublin. He received his Ph.D. in Computer Science from Tokyo Institute of Technology in 2004. He joined IBM Research - Tokyo in 2004 and had involved with several projects such as high performance XML processing, the PHP scripting language, large-scale stream computing, the X10 programming language and so forth. Since 2010, he has started to develop an X10-based agent simulation platform and its application to large-scale traffic simulation. He had served as a visiting associate professor at the Graduate School of Information Science and Engineering of Tokyo Institute of Technology since April 2009 until October, 2013 and explores research on high performance computing and large-scale graph analytics. Since April, 2013, he has joined the Smarter Cities Technology Center of IBM Research located in Dublin, Ireland. His e-mail address is toyo@jp.ibm.com.

HIROKI KANEZASHI is a student at the Graduate School of Information Science and Engineering of the Tokyo Institute of Technology. His email address is kanezashi.h.aa@m.titech.ac.jp.