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

TOWARDS BILLION-SCALE SOCIAL SIMULATIONS

Toyotaro Suzumura

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

Charuwat Houngkaew
Hiroki Kanezashi

Tokyo Institute of Technology
2-12-1 Oo-okayama
Meguro Tokyo JAPAN

ABSTRACT

Many social simulations can be represented using mobile-agent-based model in which agents moving around on a given space such as evacuations, traffic flow and epidemics. Whole planet simulation with billions of agents at microscopic level helps mitigate the global crisis. It introduces new technical challenges such as processing and migrating many agents and load balancing among hundreds of machines. To overcome these challenges, well-designed software architecture of a simulator is essential. In this research, we proposed agent-based complex cellular automata architecture (ABCCA) and studied the performance and scalability of two cell-based processing models, through simple traffic flow simulation on multi-core distributed system. The experiments show that the computation speedup can be achieved by reducing granularity of tasks and processing only active spaces. We achieved running the traffic flow simulation with one billion of agents in almost real time on 1,536 CPU cores of total 128 machines of TSUBAME supercomputer.

1 INTRODUCTION

Mobile-agent-based simulation is a simulation that simulates agents moving along their trajectories over a given space, which is appropriate for simulating entities that have moving behavior such as vehicles and people. Typical examples of this kind of simulation include traffic simulation, pedestrian simulation and epidemic simulation.

A mathematical model used in macroscopic simulation for simulating this kind of behavior have been studied for decades but this approach are incapable of capturing meticulous details as compared to cellular automa and agent-based approaches used in microscopic simulation. A perturbation of an agent's action can cause cascading effects that have a profound impact on the whole system. Nishi et al. report that a minute change in the configuration of vehicles at merging area on highway can greatly reduce congestion (Nishi et al. 2009). Such phenomena can be captured in microscopic simulation.

Understanding complex dynamics of our societies through planetary-scale simulation will provides the important clues for the solutions towards the contributions to global sustainability. The FuturICT project(Van den Hoven et al. 2012) is an ongoing project that aims at exploring techno-socio-economic-environmental systems in order to provide sustainable and resilience evolutionary guideline for our world. The project encompasses planetary-scale social simulation at microscopic level as a key tool to study interactions and cascading effects on the planet Earth to detect crises and find solutions for mitigating them. According to the fact that world population has already reached 7 billions, planetary-scale social simulation inevitably deals with billions of agents.

The advantage of agent-based approach over cellular automata approach is that the agent-based approach can represent more realistic, complex dynamics of the system and expose the effects of various parameters in different dimensions. The main downside of the agent-based approach is that the agent-based approach requires intensive computational resource, which leads to limitation of the number and the model complexity

of agents. The planetary-scale simulation is obviously unfeasible for a SMP machine and using distributed systems is a promising solution. However, the distributed system introduces new technical challenges such as migrating a large number of agents as well as load balancing among hundreds of machines. To overcome these challenges, well-designed software architecture of a simulator is essential.

In this research, we study the performance characteristics and scalability of two parallel,distributed processing models of agent-based complex cellular automata model for mobile-agent-based simulation using simple traffic flow simulation.

Our contribution are as follows:

- We propose the agent-based complex cellular automata (ABCCA) which is a variation of the agent-based cellular automata (ABCA)(Sudhira et al. 2005).
- We study performance characteristics and scalability of complex-cell-based processing model and sub-cell-based processing model for ABCCA using a simple traffic flow simulation.
- We demonstrate how to achieve simulating one billion of vehicles in simple traffic flow simulation in almost real time.

The rest of the paper is organized as follows. Section 2 describes background and related work. Section 3 describes ABCCA as well as complex-cell-based processing model and sub-cell-based processing model. Section 4 describes how we implement simple traffic simulation using ABCCA and also describe related software libraries. In section 5, we present evaluation results of our implementations. In section 6, we conclude.

2 RELATED WORK

Cellular Automata (CA)(WOLFRAM 1993), which was originally introduced by Stanislaw Ulam and John von Neumann, is a classic model for spatio-temporal modeling in which state, time and space are discrete. A cellular automaton consists of a regular, uniform grid of cells, each of which has a finite number of states. The next state of a cell changes according to transition rules as well as the current state of itself and its neighbors. CA have been used in many social simulations in which interactions are entirely local such as urban growth simulation(Batty, Xie, and Sun 1999) and epidemic simulation(White, del Rey, and Snchez 2007). Apart from simulation area, CA have been also applied in cryptography(Bouvry, Seredyki, and Zomaya 2004).

Agent-based model (ABM) is a new paradigm for simulating the actions and interactions of agents, which can expose their collective effects on the system as a whole. An agent is an autonomous, discrete entity which has its own rules controlling its action to the environment, and all agents function independently of the others. ABM is capable of representing more realistic, complex dynamics of the system than CA and revealing the relationships between parameters that is in different dimensions. ABM can be viewed as a generalization of cellular automata in which agents are able to move around in space, rather than being restricted to the cells(Macal and North 2008). Agent-based Cellular Automata (ABCA)(Sudhira et al. 2005) is an integration between CA and ABMs, which can represent spatial dynamics as well as agent dynamics.

Mobile-agent-based simulation that all interactions are entirely local coincides with ABCA. This kind of simulation is feasible to run on parallel, distributed computing systems for dealing with large-scale datasets. Simulating millions of agents in traffic flow simulation in super real time has been achieved in our prior work (Suzumura and Kanezashi 2012), and we also reported that global synchronization as well as workload imbalance are the critical problems that prevent them from achieving simulating billions of agents.

Many techniques have been introduced to mitigate such problems, for example, adapting granularity of synchronization and time-wrap technique(Jefferson et al. 1987). Adapting granularity of synchronization technique allows performing global synchronization to be skipped at some simulation steps in which a small number of data need to be synchronized, rather than performing synchronization at every step.

Suzumura and Kanezashi(Suzumura and Kanezashi 2013) enhance the traffic flow simulation using the adapting granularity of synchronization technique by reducing the frequency of synchronization when the time in the simulation is in the period from late night until dawn where the number of vehicles in the simulation is very low compared to in rush hours. They achieved significant speedup from using that technique with a trivial precision loss. Time Wrap is an optimistic synchronization protocol that allows simulation on different machine goes ahead of another. The integrity of the simulation are preserved by roll-back mechanism which is the heart of Time Warp. In this technique, each computation unit has it owns local time, i.e., its current simulation step, and each uses messages containing the time stamp of its local time to communicate with another. If a computation unit receive a message that has the time stamp before its local time, i.e., the message from the past, it will roll back its states as far as necessary and then continue its processing from that point onwards. Benchmarking the Time Warp performance on 1,966,080 cores of BlueGene/Q supercomputer which yields an event rate of 504 billion is presented in (Barnes et al. 2013). We are interested in enhancing the performance of simulation at fundamental level, i.e., the means of processing agents which pose the huge different in performance when dealing with billions of agents.

3 AGENT-BASED COMPLEX CELLULAR AUTOMATA

As the emergence of GIS data, many agent-based simulations have been carried out on the topic of integration of them, which leads to more complicated virtual space. For more realistic, a simulation might simulate the world city which consists of a variety of spatial objects such as road network, buildings. In this case, it is unfeasible to model spatial objects as a regular grid of cells.

In this section, we propose Agent-Based Complex Cellular Automata (ABCCA), which is derived from ABCA. In this model, a virtual space consists of an array of complex cells. A complex cell is an abstract of a complex spatial object that has agents move around in it. When an agent reaches the border of its current cell, it will be migrated to its next cell. The state of the complex changes according to its local agent. Basically, a complex cell is composed of a group of sub cells that shares some data. A sub cell is a portion of a complex cell that agents actually resides. In traffic flow simulation, for example, a cross point together with its incoming roads represents a complex cell and the roads represent sub cells, this approach is used in (Suzumura and Kanezashi 2012). Figure 1 illustrates the idea of ABCCA.

Agent-Based Complex Cellular Automata is able to run on distributed systems by partitioning a group of complex cells into separate partitions and putting them into respective machines. Each machine processes only its local complex cells and if agents happen to move to a new complex cell on remote machines, they will be migrated through ad-hoc communication or collective communication.

Basically a single machine in distributed systems is endowed with multi-core CPUs, which CPU core can process their own tasks in parallel with another. ABCCA architecture can harness all CPU cores by letting them process sub-cells in parallel. The rest of this section describes two processing models for ABCCA, (i) Complex-Cell-Based processing model and (ii) Sub-Cell-Based process model.

3.1 Complex-Cell-Based Processing Model

In this processing model, a group of complex-cells are divided into separate groups, the number of which is equal to the number of available processors in the systems. Each processor sequentially processes local agents in each complex-cells assigned to it. The mapping between physical computation and logical computation is shown in Figure 2a.

The major drawback of this model is that the computation is prone to delay owing to the workload imbalance when there is skewed distribution of agents (see Figure 2b).

3.2 Sub-Cell-Based Processing Model

The previous model has parallelism at course-grained level which leads to workload imbalance. We propose sub-cell-based processing model which is a result of optimizing the previous model. In this model, active

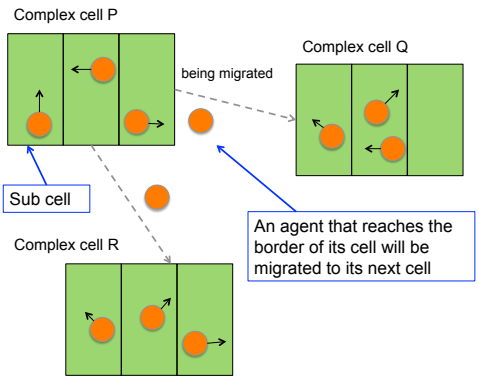
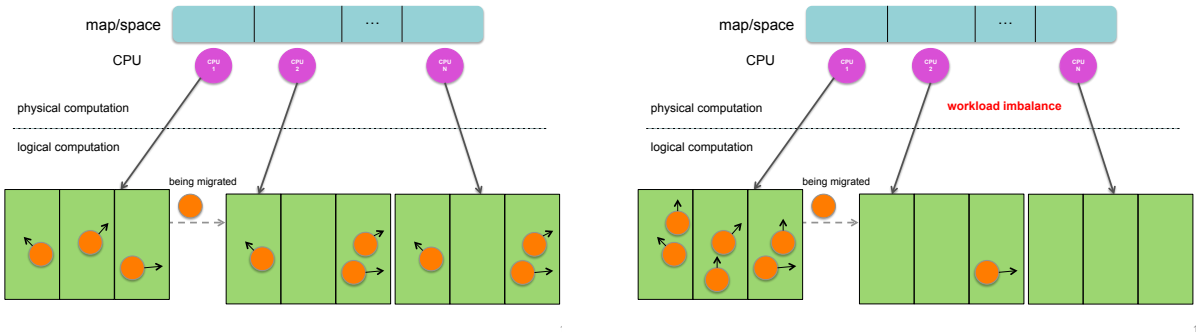


Figure 1: Agent-Based Complex Cellular Automata.



(a) (b)

Figure 2: A mapping between physical computation and logical computation of Complex-Cell-Based Processing Model, (a) normal distribution og agents, (b) skewed distribution of agents leads to the workload imbalance.

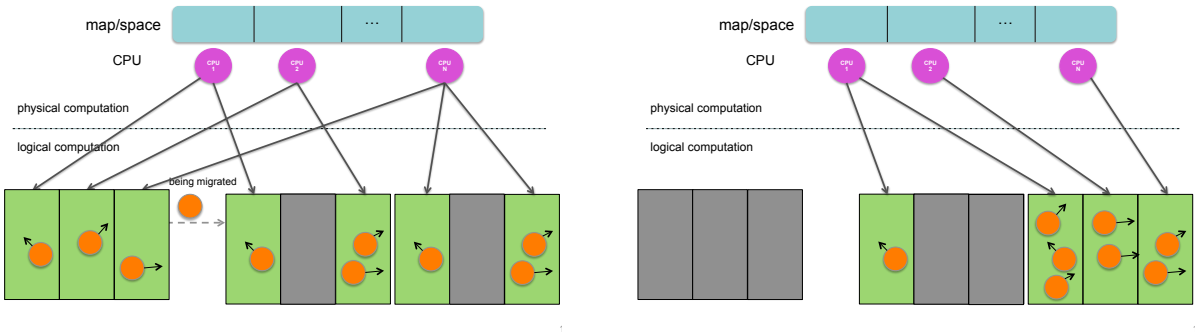
sub-cells, rather than complex cells will be processed in parallel by the number of active sub-cells evenly distributed to available machines. By this approach, the workload imbalance problem can be mitigated. The mapping between physical computation and logical computation is shown in Figure 3a. When there is skewed distribution of agents, the workload imbalance problem does not significantly affects the computation (see Figure 3b).

4 APPLICATION TO SIMPLE TRAFFIC FLOW SIMULATIOM

In this section, we describe the programming language, related software library and the implementation of a simple microscopic traffic flow simulation that is used for studying the performance and scalability of the two processing models.

4.1 X10 Programming Language Overview

X10(Saraswat et al. 2007) is a programming language which has been developed as open-source software by IBM research in collaboration with many academic institutions around the world. X10 is a type-safe,



(a) (b)

Figure 3: A mapping between physical computation and logical computation of Complex-Cell-Based Processing Model, (a) normal distribution og agents, (b) skewed distribution of agents leads to the workload imbalance.

imperative, concurrent, object-oriented programming language designed for multi-core and cluster systems, with the awareness of developer productivity as well as software performance.

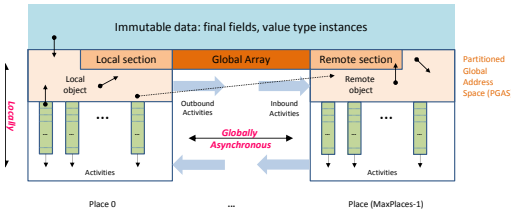
X10 implements the APGAS (asynchronous partitioned global address space) model which merges the merits of an SPMD programming style for distributed memory systems and the data referencing semantics of shared memory systems as well as permits local and remote asynchronous task creation. X10 introduces the concepts of *place* and *activity*. *Place* might be loosely viewed as a process; it has its own local *data* and has *activities* running within it. *Activity*, which is similar to a light-weight thread, is a statement that is being executed; it has its own local variables but shares *Place's local data* with one another. *Activity* can access *remote data* (i.e., *data* located on another place) via collective communication routines of *X10.util.Team* class or *at* statement. An *activity* can also migrate with all referenced objects from one place to another place and then continues its execution seamlessly on the new place. X10 programming model is shown in Figure 4a.

X10 source code can be compiled to either native (C++) program or managed (Java) program (see Figure 4b). The key advantage of a native program is performance gain, while that of managed program is portability. X10 also provides the notion of *native* which permits C++ or Java code to be integrated into X10 code. This allows developer to use existing third-party libraries written in C/C++/Java with X10 program, or even tune the performance at the back-end level.

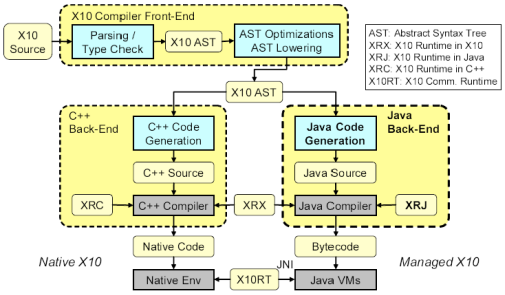
Currently, X10 supports many programming styles used in HPC world, e.g., fine-grained concurrency, Cilk-style fork-join programming, GPU programming, SPMD computations, phased computations, active messaging, MPI-style communicators, and cluster programming, which gives developers various choices for dealing with their problems. X10 offers various transportation ranging from basic transportation to the transportation that is widely used in high performance computing systems, such as TCP/IP, PAMI, DCMF and MPI.For more information regarding to X10, we refer the reader to X10 language specification(Saraswat et al. 2012).

4.2 ScaleGraph library

ScaleGraph(Dayarathna et al. 2012) is a library designed for large-scale graph analytics using state-of-the-art highly-productive X10 programming language. ScaleGraph provides rich algorithms that are capable of running on parallel and distributed systems. The algorithms cover degree distribution, distributed betweenness centrality, PageRank and spectral clustering, HyperANF as well as Pregel-based(Malewicz et al. 2010) highly-efficient distributed computing framework. Apart from the algorithms, ScaleGraph also offers many features that significantly enhance the performance of X10 program.



(a) X10 programming model.



(b) X10 compiler architecture.

Figure 4: X10 programming language.

MPI is a well-known library used for developing software that run distributed memory systems. ScaleGraph uses MPI as a mean of communication among places and also optimizes the interface layer between X10 runtime and MPI. The performance gain of the collective communication in X10 from the optimization is significantly improved(Houngkaew and Suzumura 2013).

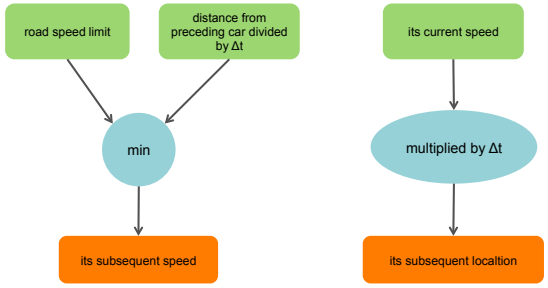
4.3 Simple Traffic Flow Simulation

The simple microscopic traffic flow simulation used throughout all experiments is a simplified version of IBM Mega traffic or Megaffic(Osogami et al. 2012). In Megaffic, a trajectory of a vehicle is predetermined according to the model of origin and destination as well as the model of route selection, which is different from the typical traffic flow simulation where trajectory of a vehicle is dynamically determined at each simulation step according certain factors such as traffic congestion, distance and speed limit. Because the model of origin and destination as well as the model of route selection actually consider the landmarks in the map, probe-car data, and census data, this makes Megaffic capable of simulating the traffic flow of any place in the scale of an arbitrary city or even the whole country. Megaffic can be used to provide the information to authorities for designing a better layout of a city.

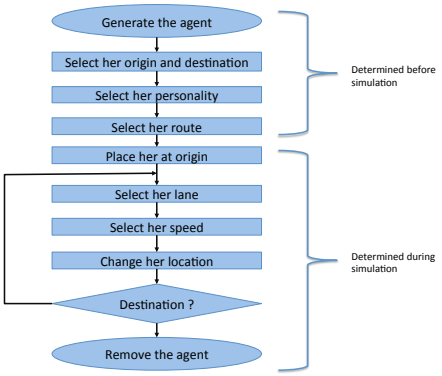
The simple traffic flow simulation uses simple car-following model in which the location and speed of a vehicle are simply determined from the speed of the road and the location of the preceding vehicle, rather than Gipps' car-following model as in Megaffic. In the simulation, a vehicle is represented by an agent which moves on the roads along its predefined route starting from the origin to the destination. An agent is created at the time it departs and deleted when it reaches its destination. A group of a cross point and its associated incoming roads is represented by a complex cell and the roads are the sub cells. The whole road network is represented by a group of all complex cells. An agent keeps four pieces of information: (i) trip data, i.e., a predefined route from origin to destination, which is a series of cross points, where the first cross point is the origin and the last cross point is the destination, (ii) departure time which is the time that an agent start traveling, (iii) current speed which is the safety speed that an agent uses to move itself, and (iv) current location which is the length from the origin of the road to its current position on the road.

Trip data of each agent is predetermined before the simulation while its speed and position are iteratively determined during simulation. The speed are adjusted according to road speed limit and the distance from the preceding vehicle's location by selecting the maximum speed that the vehicle will not crash the preceding vehicle and the speed is less than road speed limit. The location is considered from its current speed and the time length of simulation step. Figure 5a shows how the speed and location of an agent are determined and Figure 5b shows the overview of agent life in the simulation.

We chose to implement our simulator on top of ScaleGraph library according to the performance and productivity. Road network is evenly divided into separate partitions, the number of which is equal to the number of available places. Each partition is put into its respective place. The agent data are distributed according to the place of its first cross point. Each place processes its complex cells for complex-cell-based processiong model or sub cells for sub-cell-based processing model in parallel in parallel by creating a given number of X10 *activities*, e.g., the number of available hardware threads, and assigning them the tasks evenly. If an agent happens to change its complex cell, i.e, cross point, that is in a remote place, the agent will be migrated to the remote place. The migration are not performed immediately when an agent is found to move to a remote place, but each place maintains a list of agents to be migrated and after all places finish processing all agents, they exchanges those agents in the lists with each other using all-to-all collective communication.

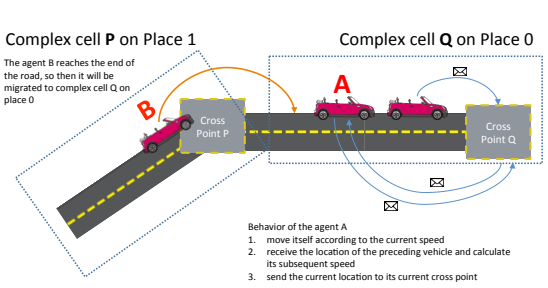


(a) Speed and location determination at each simulation step.

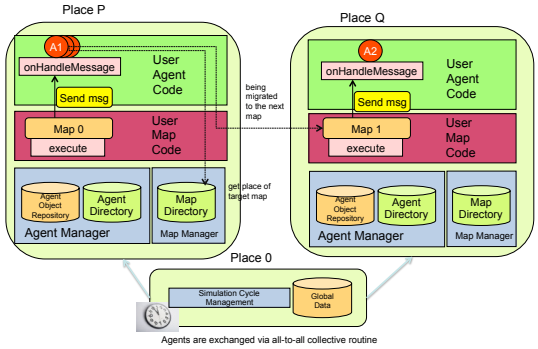


(b) Agent life.

Figure 5: A model of an agent.



(a) Agent behavior.



(b) Simulation flow.

Figure 6: Simple traffic flow simulation using ABCCA.

5 EVALUATION

We conducted the evaluation of performance and scalability on Tsubame 2.5 supercomputer. Tsubame is a production supercomputer ranked as the fourth of TOP500(Top500.org 2014) in November 2010 and the fifth in June 2011 with peak performance of 2,288 teraFLOPS (FLoating-point Operations Per Second). Each compute node used in all experiments is equipped with two Intel® Xeon® X5760 2.93 GHz CPU cores by each having 6 cores and 12 hardware threads (12 cores and 24 hardware threads in total), 54GB of memory and 128GB SSD. All compute nodes run on SUSE Linux Enterprise 11 SP1 and each is connected with InifinitBand device 'Grid Director 4700' with non-blocking and full-bisection bandwidth. We used ScaleGraph 2.1 release, X10 2.3.1 release and MVAPICH2 1.9 release. We set X10_NTHREADS (the number of worker threads), GC_NPROCS (the number of garbage-collector threads) environment variables to 16 and 8 respectively. All experiments are configured to run one place per compute node. We measured wall-clock time of simulating from step zero until the given last step excluding the time for loading agent data and road network as well as printing the result.

5.1 Dataset

5.1.1 Road Networks

Open Street Map (OpenStreetMap 2014) is an open crowd-sourced map data that allows users to share geography data including data from their GPS devices and aerial photographs. The road networks used in our experiments are real Tokyo road networks which were extracted from OSM. We renumbered the IDs of cross points and roads to contiguous range numbers starting from zero for the convenience and efficiency of storing data in an array. The Tokyo road networks consist of 770,192 of cross points and 2,089,374 of roads.

5.1.2 Trip Data

In our experiments, we did not use probe-car data, census data, nor the model of origin-destination as in IBM Mega traffic. All trip data we used is synthetic data generated by each pair of origin and destination is arbitrarily selected. The route from the origin and the destination is the shortest path between them. The depart time of a trip is uniformly random between 0 and 10,000. Because computing shortest paths for every pair of origin and destination is time-consuming process, the trip data that have the number of trips more one than million are generated by replicating the one million data set with new random departure time. The departure time of a trip in billion-scale data set is uniformly random between 0 and 2,000.

5.2 Strong-Scaling Performance Evaluation

We ran the simulation by varying the number of machines with fixed problem size, i.e., the number of agents remaining the same in all cases, to study how well the implementations scale when more machines are provided. We ran the implementations simulating 3,600 steps with one million of agents on Tokyo road network on one, two, four, eight and sixteen machines, respectively. The wall-clock time of the simulation of each implementation is shown in Figure 7. In the case of single machine, the sub-cell-based processing model obviously outperforms the complex-cell-based processing model with 2.37 of speedup. In the case of multiple machines, the complex-cell-based processing model and the sub-cell-based processing model have the performance saturated from eight machines and four machines, respectively. According to the evaluation result, communication cost is a critical factor affecting the performance of both processing models.

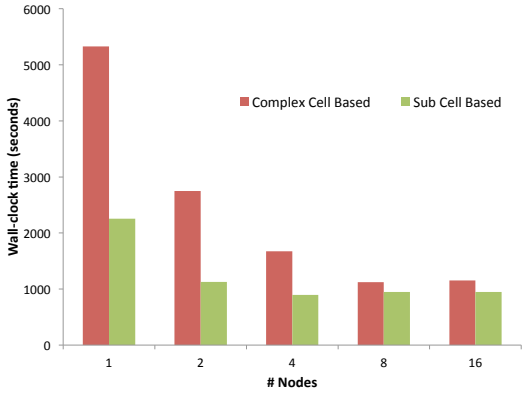


Figure 7: Wall-clock time for simulating 3,600 steps using one-million data set.

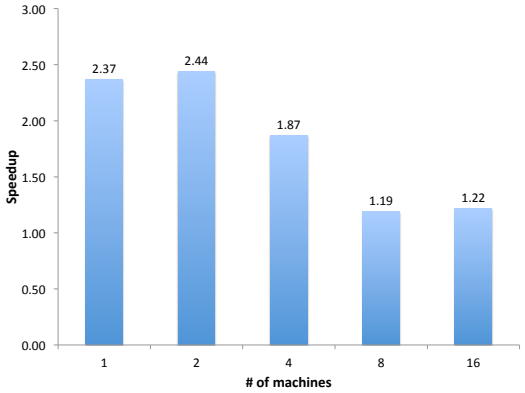


Figure 8: Speedup of sub-cell-based processing model against complex-cell-based processing model for simulating 3,600 steps using one-million data set.

5.3 Weak-Scaling Performance Evaluation

We ran the simulation by varying the number of machines and problem size to study how well the implementations scale when the problem size increases as the number of machines increase by the problem size per machines being fixed. We ran the implementations simulating 3,600 steps using one-million data set, two-million data set, four-million data set and eight-million data set on one machine, two machines, four machines and eight machines, respectively, on Tokyo road network. The wall-clock time of the simulation of each implementation is shown in Figure 9. From the evaluation result, the sub-cell-based processing model obviously outperforms the complex-cell-based processing model for one-machine case to four-machine case. In the case of eight machines, both wall-clock time is significantly different. As the problem size of one million agents is not large enough for a single machine, doubling the size of the problem as well as the number of machines leads to shorter wall-clock time rather than constant.

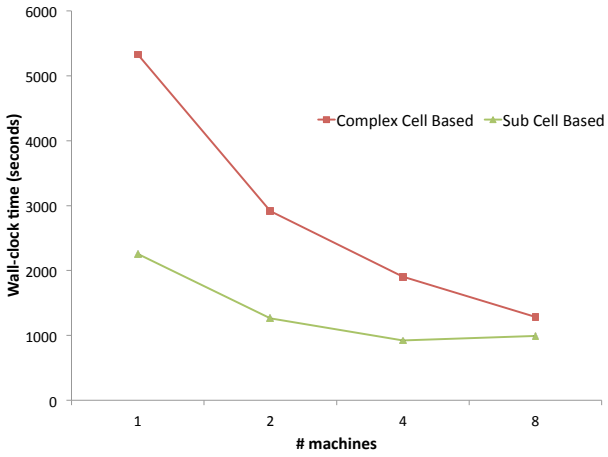


Figure 9: Wall-clock time for simulating 3,600 steps using one-million data set, two-million data set, four-million data set and eight-million data set on one machine, two machines, four machines and eight machines, respectively.

5.4 Experiment Towards Billion-Scale Evaluation

In this section, we evaluated only the implementation of the simple traffic flow simulation with sub-cell-based processing model with billion-scale dataset, because it shows promising performance and scalability for large-scale simulation as presented in weak-scaling performance analysis and strong-scaling performance analysis sections.

We ran the implementation simulating 3,600 steps with one billion of vehicles on Tokyo road network on 128 machines, which utilizes 1,536 CPU cores in total. The Tokyo road networks are considered small compared to one billion of agents. Using this networks imposes very high density of agents per road segment, which might affect the performance of the simulator. Since generating a massive number of agents on a very large map which involves determining a billion shortest paths, is a time consuming process, we instead use the Tokyo road networks and replicate the agents from one-million data set. Simulating on a larger graph will be our future work. Because the departure time of all trip in billion-scale data set is randomly selected between 0 to 2,000, which means all vehicles has departed after the step 2,000 onwards. The wall-clock time is around 114 minutes with an average of 1.9 seconds per simulation step. Figure 10 shows the wall-clock time of each simulation step and Figure 11 shows the average wall-clock time at each simulation step. As the simulation progresses, the wall-clock time increases owing to more agents added into the simulation.

We observed the glitches in the wall-clock time graph, and according to our preliminary investigation, it is related to memory management of garbage collector. The implementation used many large chunks of memory and during simulation they were repeatedly allocated and released. These also affect the performance of the garbage collector as well as the simulator.

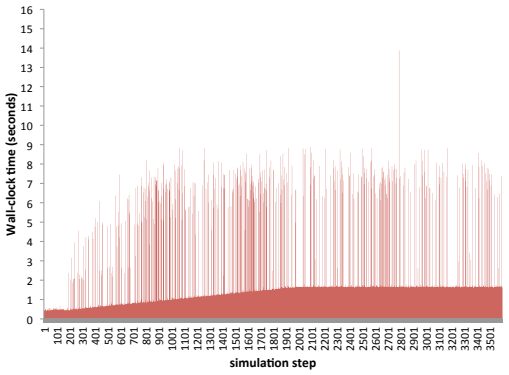


Figure 10: Wall-clock time of each simulation step in billion-scale evaluation.

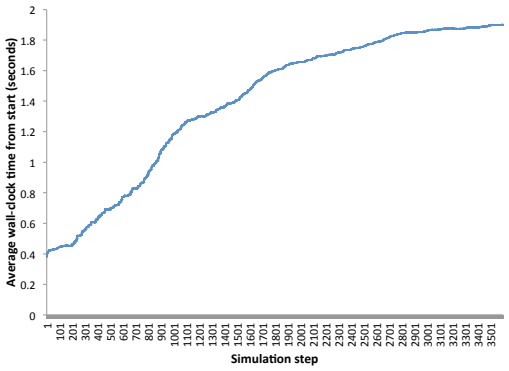


Figure 11: Average wall-clock time at each simulation step in billion-scale evaluation.

6 CONCLUSION

We studied the performance and scalability of the two processing models, complex-cell-based processing model and sub-cell-based processing model, for mobile-agent-based simulation through simple traffic flow simulation. According to the evaluation result, sub-cell-based processing model significantly outperformed complex-cell-based processing model. We achieved running the traffic flow simulation with a billion of agents in almost real time on 1,536 cores of total 128 machines of Tsubame supercomputer using sub-cell-based processing model.

As for future work, we are interested in improving our simulation to be more realistic by such as generating trip data from probe-car data and census data, implementing more sophisticated model of speed selection as well as using country-scale road networks. The frequency of global synchronization has

directly affects the performance of the simulator, we are also interested in another approximation method such as adaptive granularity of synchronization, to cope with this problem. Furthermore, conducting the performance and scalability evaluation on various environments such as commodity PC clusters and Amazon EC2 is also interesting.

ACKNOWLEDGEMENTS

This research was partly supported by the Japan Science and Technology Agency's CREST project.

REFERENCES

Barnes, Jr., P. D., C. D. Carothers, D. R. Jefferson, and J. M. LaPre. 2013. "Warp Speed: Executing Time Warp on 1,966,080 Cores". In *Proceedings of the 2013 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '13, 327–336. New York, NY, USA: ACM.

Batty, M., Y. Xie, and Z. Sun. 1999. "Modeling Urban Dynamics Through GIS-based Cellular Automata". *Computers, Environment and Urban Systems* 23 (3): 205 – 233.

Bouvry, P., F. Seredyki, and A. Zomaya. 2004. "Application of Cellular Automata for Cryptography". In *Parallel Processing and Applied Mathematics*, edited by R. Wyrzykowski, J. Dongarra, M. Paprzycki, and J. Waiewski, Volume 3019 of *Lecture Notes in Computer Science*, 447–454. Springer Berlin Heidelberg.

Dayarathna, M., C. Houngkaew, H. Ogata, and T. Suzumura. 2012. "Scalable Performance of ScaleGraph for Large Scale Graph Analysis". In *2012 19th International Conference on High Performance Computing (HiPC)*, 1–9.

Houngkaew, C., and T. Suzumura. 2013. "X10-Based Distributed and Parallel Betweenness Centrality and Its Application to Social Analytics". In *2013 International Conference on High Performance Computing (HiPC)*, 109–118.

Jefferson, D., B. Beckman, F. Wieland, L. Blume, and M. Diloreto. 1987. "Time Warp Operating System". In *Proceedings of the Eleventh ACM Symposium on Operating Systems Principles*, SOSP '87, 77–93. New York, NY, USA: ACM.

Macal, C., and M. North. 2008, Dec. "Agent-Based Modeling and Simulation: ABMS Examples". In *Simulation Conference, 2008. WSC 2008. Winter*, 101–112.

Malewicz, G., M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. 2010. "Pregel: A System for Large-scale Graph Processing". In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, 135–146. New York, NY, USA: ACM.

Nishi, R., H. Miki, A. Tomoeda, and K. Nishinari. 2009, Jun. "Achievement of alternative configurations of vehicles on multiple lanes". *Phys. Rev. E* 79:066119.

OpenStreetMap 2014. "OpenStreetMap". http://openstreetmap.jp/.

Osogami, T., T. Imamichi, H. Mizuta, T. Morimura, R. Raymond, T. Suzumura, R. Takahashi, and T. Ide. 2012. "IBM Mega Traffic Simulator". Technical report, IBM Research Report, RT0896.

Saraswat, V., B. Bloom, I. Peshansky, O. Tardieu, and D. Grove. 2012, Feb. "X10 Language Specification (version 2.3)". http://x10-lang.org/.

Saraswat, V. A., V. Sarkar, and C. von Praun. 2007. "X10: Concurrent Programming for Modern Architectures". In *Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '07, 271–271. New York, NY, USA: ACM.

Sudhira, H., T. Ramachandra, A. Wytzisk, and C. Jeganathan. 2005. "Framework for Integration of Agent-Based and Cellular Automata Models for Dynamic Geospatial Simulations". *Centre for Ecological Sciences, Indian Institute of Science) Bangalore*.

Suzumura, T., and H. Kanezashi. 2012. "Highly Scalable X10-Based Agent Simulation Platform and Its Application to Large-Scale Traffic Simulation". In *2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 243–250.

Suzumura, T., and H. Kanezashi. 2013. "Accelerating Large-Scale Distributed Traffic Simulation with Adaptive Synchronization Method". In *20th ITS World Congress 2013*.

Top500.org 1993-2014. "TOP500". http://www.top500.org.

Van den Hoven, J., D. Helbing, D. Pedreschi, J. Domingo-Ferrer, F. Gianotti, and M. Christen. 2012. "FuturICT-The road towards ethical ICT". *The European Physical Journal Special Topics* 214 (1): 153–181.

White, S. H., A. M. del Rey, and G. R. Snchez. 2007. "Modeling Epidemics using Cellular Automata". *Applied Mathematics and Computation* 186 (1): 193 – 202.

WOLFRAM, S. 1993. "Statistical Mechanics of Cellular Automata". *Reviews of Modern Physics* 55 (3): 601–644.

AUTHOR 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.

CHARUWAT HOUNGKAEW is a student at the Graduate School of Information Science and Engineering of Tokyo Institute of Technology. He enrolled there in April 2012. His e-mail address is swagen.xivth@gmail.com.

HIROKI KANEZASHI is a student at the Graduate School of Information Science and Engineering of Tokyo Institute of Technology. He enrolled there in April 2013. His e-mail address is kanezashi.h.aa@m.titech.ac.jp.