Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

LARGE-SCALE DISTRIBUTED AGENT-BASED SIMULATION FOR SHOPPING MALL AND PERFORMANCE IMPROVEMENT WITH SHADOW AGENT PROJECTION

Hideyuki Mizuta

IBM Research
19-21 Nihonbashi Hakozaki-cho, Chuo-ku
Tokyo 103-8510, JAPAN

ABSTRACT

In this paper, we introduce the agent-based simulation of a shopping mall with walking and purchasing behavior model and consider the performance of distributed parallel execution. To utilize the agent-based simulation for decision support, distributed parallel execution of large-scale agent-based social simulations is important for evaluating the complex behavior of a realistic number of people with acceptable performance. For this purpose, today's agent-based simulation frameworks often provide the functionality to transfer agents from one node to another. However, intelligent social agents tend to contain a large amount of data including demographics, preferences, and history. Hence, the transfer of such an agent incurs a heavy communication cost that has an adverse effect on performance. To improve the performance of distributed agent-based simulation, we introduce a shadow agent that is a lightweight entity projected among nodes with only required information such as the position and speed required to calculate interaction between agents.

1 INTRODUCTION

Recently, agent-based social simulations are utilized to support the decision making of city planner for various real social issues including evacuation (Yamashita, Matsushima, and Noda 2014), intelligent transportation (Chen and Cheng 2010) and financial market regulations (Mizuta et al. 2013). To evaluate complex behavior with interaction among heterogeneous peoples in a large city, such a social simulation need to manage millions of agents with various behavior models and preferences. Moreover, the computation speed is also required to analyze enormous combination of possible situations and strategies with repeated simulations (Reviews and roadmap are introduced in (Noda et al. 2015)). Therefore, the distributed parallel execution of large-scale agent-based social simulations is important for evaluating the complex social interaction of a realistic number of people and situation with acceptable performance.

For this purpose, many large-scale agent-based simulation software provide the functionality to communicate and transfer agents from one node to another (e.g. Repast HPC (Collier and North 2013) and Megaffic (Suzumura et al. 2012)). On the other hand, intelligent social agents tend to contain a large amount of data including demographics, preferences, and history, and the transfer of such an agent incurs a heavy communication cost that has an adverse effect on performance.

To improve the performance of such a distributed agent-based simulation, we introduce a shadow agent that is a lightweight entity projected among nodes with only required information such as the position and speed required to calculate interaction between agents. By utilizing this novel method for distributed simulation with a shadow agent, the number of message transactions is increased, but the communication data size is decreased enough to reduce the total transaction cost.

Such an approach to use the lightweight shadow agent for distributed simulation is a novel idea. In the context of distributed multi agent cooperation, (Bansal 2006) considered the copy of agents denoted as "shadow agent". However, their "shadow" agent contains all information of the original agent and works

on behalf of the original agent in the case of failure. Thus this is quite different from our shadow agent method that contains only small set of information for interaction.

As a concrete example of the social simulation, we develop a shopping mall simulation on the distributed agent framework and introduce the class structure and behavior models of the simulator. Then we introduce the shadow agent method to improve the performance of the distributed parallel execution. Finally, we evaluate the simulation execution time for the traditional method and our shadow agent method with this application.

2 AGENT-BASED SIMULATION OF SHOPPING MALL

In this section, we introduce an agent-based shopping mall simulator as an example of distributed social simulation. Our simulator is developed on the X10-based Agent Simulation on Distributed Infrastructure (XASDI) which uses the X10 programming language (X10) for distributed parallel execution. This XASDI framework is published as an open source software under the Eclipse Public License (EPL).

2.1 Agent Framework and Application

XASDI is a large-scale agent-based social simulation framework with enormous number (billions) of agents to represent citizens in cities or countries. In previous works, we utilized this framework to develop a large scale agent-based traffic simulator for the metropolitan traffic flow (Osogami et al. 2013, Mizuta 2015).

XASDI enables distributed simulations with the X10 programming language for post-Peta Scale machines. The X10 programming language is the APGAS (Asynchronous, Partitioned Global Address Space) language that provides highly parallel and distributed functionalities with Java-like syntax (X10). On the other hand, XASDI provides easy-to-use API with Java that is familiar to application programmer of social simulations and can be developed with powerful IDE functionalities (e.g. Eclipse refactoring and debugger).

XASDI software stack contains the core runtime written in the X10 language for distributed agent and execution management and the API bridge to enable application programmers to utilize the familiar Java language (Figure 1). By utilizing XASDI framework users can easily develop their social simulator with Java on distributed parallel environment without studying an unfamiliar X10 language.

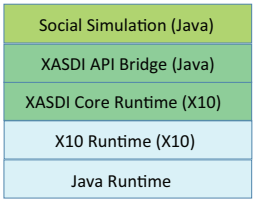

Figure 1: XASDI software stack on X10 and Java Runtimes.

The agent in XASDI is referred to as Citizen and Citizen has corresponding CitizenProxy that is managed in the simulation environment to exchange messages. To manage CitizenProxy, XASDI provides a hierarchical container structure called Place, Region and World (see Figure 2). CitizenProxies belong to a Place and Places belong to a Region. World can contain several Regions, but usually there is only one Region in a World.

Here, we need to note that the confusing terminology of the X10 language and the XASDI framework. The X10 language uses the term “Place”, too, but the meaning of the term is different. The Place of X10 is used to denote the distributed execution environment for multi-core or multi-node. For this meaning, we will use “X10 Place (node)” in distinction from the Place container of agents. Only one World instance exists in one X10 Place and manages lists of entities in the World including Region(s) and Citizens. The World can also contain IDs of Citizens in other nodes.

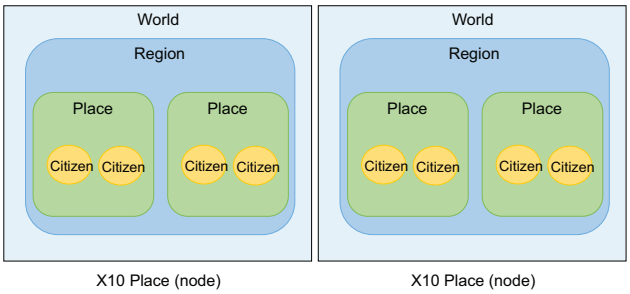

Figure 2: XASDI hierarchical structure to manage agents.

Other important classes in XASDI are Message, MessageRepository and Driver. MessageRepository manages message transaction among CitizenProxies and environment. This class also works as interface between the Java environment and X10 environment to exchange Messages in distributed X10 Places. Driver manages execution of the simulation with a corresponding thread. Each Driver is related to Places (and Citizens in the Places) where it has a responsibility for execution.

By extending the classes of this framework, we develop the shopping mall simulator with Java in which agents walk and shop around the mall. In our simulator, the *Mall* class extends the Region and the *Zone* (and Shop, which is a variation of Zone) extends the Place. The Zone represents a shop or aisle at which a *Consumer* agent is located.

Action classes are also implemented on the Driver class to perform an consumer behavior for each time step (typically 1 second) with different threads in parallel. For the shopping mall simulation, we define *Attraction*, *Move*, *Shop*, and *Purchase* actions. Each action is related with models possessed by a Consumer agent to allow different behavior for each person. We implemented Attraction, Move, Shop, and Purchase models based on Hui's model (Hui, Bradlow, and Fader 2009) and combined them with a *Walk* model.

We consider a three-layer representation of the mall. One map layer is a concrete geo spatial representation defined with 3D coordinate data and another is an abstract network representation defined with nodes and links. The Zone is a node and the adjacency relationship between two Zones is a link. On top of these two map layers, we also consider a third economic layer that includes marketing and purchasing activities.

Figure 3 shows the three layers of the shopping mall simulation.

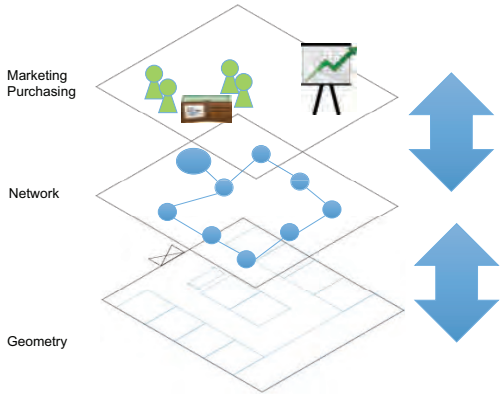

Figure 3: 3-layer structure of shopping mall simulation.

For economic activities including destination decision-making in the mall, we use the network representation. For pedestrian activities, we use the geo spatial representation. This integration of economic behavior and moving behavior is a key factor of this agent-based simulation.

At the beginning of the simulation, the simulator reads the Zone definition file and generates Zone objects (including *Entrance* and *Shop*) and related *GeoZones* that represent geo spatial locations simultaneously. Consumer agents enter the mall from the Entrance zone at uniform random times during the simulation.

At each time step, agents update their attraction for products and zones, as described later. The attraction is a numeric value associated with each agent (i.e., customer), where the value represents the level of interest of the customer to the zone, or products sold in the zone, at which the agent is located at a given time. Each agent may or may not have a destination. A destination is a zone in the network representation. If an agent does not have a destination, it selects a destination with the probability determined on the basis of zone attractions.

If an agent decides to stay in the current zone, or when it reaches the destination, it walks around the current zone using the pedestrian model described in the next section. After the agent decides to make one of the zones its destination, the shortest path on the network is chosen as a trip by Dijkstra's algorithm (Dijkstra 1959) and then the agent walks toward the next zone in the trip.

2.2 Walk Model

In this subsection, we introduce our walk model of agents with collision avoidance in a crowded mall.

For collision avoidance, we utilize a combined method of agents obstacle avoidance behavior: those of (Zanlungo, Ikeda, and Kanda 2011) and (Reynolds 1999). In a shopping mall, there are a variety of situations that influence shoppers' behaviors, such as passing through overcrowded places and changing routes to avoid collision with other customers.

Zanlungo utilizes a social force model that can estimate future collisions. This works even in crowded places, and as a result, agents can penetrate the gap between agents smoothly. Reynolds proposed an approach that is also applicable to situations where other agents are approaching from the front directly by using a steering force that negates the lateral side-up projection of the obstacle's center. As a result, agents can avoid direct approaches of other agents from the front.

We utilize these mixed models in the shopping mall simulation to handle both situations. The mixed social force based on two models are calculated by the closest opponent. We estimate the future position where a collision occurs at time t and calculate the force of circular specification (Zanlungo, Ikeda, and Kanda 2011) between agent i and agent j at that time.

The force $F1$ is calculated based on Zanlugo's model (Equation 1). A and B denote the strength and the range of interaction force, respectively. In our simulation, A and B is set to 2.0 and 1.5. The circular specification assumes the force to depend on the only distance $d_{ij}(t)$ between agent i and agent j at time t.

In addition, $F2$ shown in Equation 2 represents an evade force working only on the y axis (traverse direction) at the local space of agent i. We define the local space as a specific agent's local coordinate system that can describe the relative position and orientation of other agents from the viewpoint of the specific agent. C denotes the strength of the force that can be adjusted. In our simulation, the value of C is 2.0. Each r_j and y_j represents approaching agent j's radius and position at y axis at the local space of agent i based on the current time.

$$\mathbf{F1} = A\exp(-|d_{ij}(t)|/B)\frac{\mathbf{d}_{ij}(t)}{|d_{ij}(t)|}, \tag{1}$$

$$\mathbf{F2} = C(\mathbf{r}_j - \mathbf{y}_j). \tag{2}$$

The Consumer agents update their speed vector and position with the mixed force $F = F1 + F2$.

2.3 Purchase Model

For shopping behavior, we consider the agent behavior model that are developed by extending the Hui's model (Hui, Bradlow, and Fader 2009). The parameters for the original model by Hui (Hui, Bradlow, and

Fader 2009) is determined and verified with large amount of consumer tracking data at a super market. We use similar parameters for shopping mall with modification due to shop size and time scale, but currently do not verified with real data. The estimation and verification of the model parameters for a real shopping mall remains as future work. In general, it is difficult to obtain detailed tracking data at a large shopping mall. Hence we may need to simplify the model with reduced parameter set according to the available observation data at a mall.

We do not use the geometric distribution for sojourn time used in Hui's model, but rather check an exit action with a given probability at each time step so that the surrounding environment can affect the behavior of agents dynamically. This probability can be changed according to contact with the agents. Hui's model gives utilities and probabilities using attraction of products and zones in a supermarket based on detailed tracking data. With these attraction values and other factors (e.g., sojourn time), the model for moving to zones, the shop model at the destination zone, and the purchase model for products are defined.

At each time step t, attraction values a_{ikt} of agent i for product k and A_{ijt} for zone j are updated as

$$\begin{aligned} a_{ik(t+1)} &= a_{ikt} + \Delta_{bi} B_{ikt} + \Delta_{si} I\{k \in C(x_{it})\} \\ &\quad k \neq \text{checkout}, \end{aligned} \tag{3}$$

$$\begin{aligned} a_{ik(t+1)} &= a_{ikt} + \omega_{vi} S_{it} \\ &\quad k = \text{checkout}, \end{aligned} \tag{4}$$

$$A_{ijt} = \log\left(\sum_{k \in C(j)} \exp(a_{ikt})\right) \tag{5}$$

where S_{it} denotes the sojourn time after the agent has entered the mall. The parameter ω_{vi} denotes the weight for time pressure. All agents need to keep the attraction values for all products and zones in the mall that can be large amount of data to be transferred between distributed nodes.

Relative rate R_{jt} to select destination zone j is the exponential of the move utility $\exp(u^{\text{move}})$, which is given by

$$u^{\text{move}} = Z_j + \gamma_{vi} \rho_{jt} + \kappa_i G_{ijt}, \tag{6}$$

$$G_{ijt} = A_{ijt} + \sum_{j' \neq j} \frac{A_{ij't}}{(1 + d_{jj'})^{\lambda}_i} \tag{7}$$

where Z_j denotes a baseline utility of zone j and ρ_{jt} denotes the congestion of zone j. The parameter γ_{vi} denotes the weight for social influence. G_{ijt} represents the attraction of the destination zone j under the influence from nearby zones j' with distance $d_{jj'}$. The destination is determined by the probabilities proportional to the relative rate. Though only the adjoining zones are considered in the Hui's model, we consider all zones in the mall as the candidates for the destination so that agent can travel to a far-flung zone with the appropriate (shortest) path.

At the destination zone, transition into shop mode (consideration for purchase) occurs with the following probability:

$$P^{\text{shop}} = \exp(u^{\text{shop}})/(1 + \exp(u^{\text{shop}})), \tag{8}$$

$$u^{\text{shop}} = \alpha_{si} + \beta_{si} A_{ijt} + \omega_{si} S_{it} + \gamma_{si} \rho_{jt} + \eta_j \tag{9}$$

where α_{si} and η_j denote the baseline utility of agent i and zone j.

Similarly, the probability $P^{\text{buy}} = e^u/(1 + e^u)$ to purchase a product is given by

$$u^{\text{buy}} = \alpha_{bi} + \beta_{bi} a_{ikt} + \omega_{bi} T_{it} + \gamma_{bi} \rho_{jt}, \; k \in C(x_{it}) \tag{10}$$

under the condition that the agent is in the shop mode and product category k exists in the current zone j. T_{it} denotes the duration after the agent has entered the current shop. We can see similar parameters for time pressure and social influence in shop behavior and purchase behavior. However, the social influence has a positive value for shop model $\gamma_{si} > 0$ and a negative value for purchase model $\gamma_{bi} < 0$ according to the estimation results by (Hui, Bradlow, and Fader 2009).

Though the model seems like a system dynamic model rather than an autonomous agent model, but the microscopic behavior models and attraction values are assigned to each agent and can represent heterogeneous preferences and perception as usual agent-based model.

2.4 Integration of Walk Model and Purchase Model

We now describe the integration of the geo-spatial walking behavior model and the purchasing behavior model in a zone (shop).

Although the trip path between zones is defined on the network layer, consumer agents move around a zone or toward the border with the next zone on the geospatial layer using the walk model described before. With the walk model, agents avoid each other by the combined force model. However, the personal space of a consumer agent may come into collision with another agent if there are many agents in a zone. We consider this invasion count of the personal space (*CPS*) to modify purchase behavior with two steps. First, we dynamically decrease the purchase probability by changing social pressure with accumulated *CPS* since the agent stays in the current zone. In addition, we increase the probability for a consumer to depart from a shop with a *CPS* logarithm.

The probability $P^{\text{buy}} = e^u/(1+e^u)$ to purchase a product is given by the modified utility

$$u^{\text{buy}} = \alpha_{bi} + \beta_{bi} a_{ikt} + \omega_{bi} T_{it} + \gamma_{bi}(\rho_{jt} + w_{PS} \sum CPS) \tag{11}$$

under the condition that the agent is in the shop mode and product category k exists in the current zone j. T_{it} denotes the duration after the agent has entered the current shop. The departure probability is given by

$$P^{\text{move}} = P^{\text{base}} + w_d \log(1 + CPS). \tag{12}$$

With these behavior model, consumer agents interact with each other. In the next section, we describe a new method for interacting agents in the distributed environment.

3 MULTI NODE EXECUTION OF AGENT-BASED SIMULATION

In this section, we describe the multi-node execution of an agent-based simulation and introduce our proposed method with shadow agent projection.

In one node, there is one Mall instance that manages the execution of the simulator. The core framework written using X10 manages the Regions and *MessageRepositories* of distributed nodes. The MessageRepository supports several kinds of messages such as individual, broadcast, and move. An individual message is a standard message from one agent to another. A broadcast message is sent to all nodes and received by the Mall or agents corresponding to the type of message. A move message is a control message to trigger the transfer of an agent from one node to another. The data for a shadow agent is aggregated by Zones and distributed by the Mall to other nodes with a broadcast message.

Multi-node execution of the simulation is possible by using the XASDI framework with X10-based activity and message management. The actions defined for simulation described in the previous section were executed and synchronized in parallel by using X10 activity for a multi-node environment. A message can be sent during the execution of actions from agents or a Region to others even if they are located in different nodes.

The framework also supports the traditional agent transfer between nodes using a control message including the necessary data of the agent as a deep copied field to restore the agent in the destination node.

The arrangement of agents with the traditional method is processed as follows. Each X10 Place (node) has one region instance and each region generates an agent at the same Place with a predefined schedule or distribution in the simulation duration. For the shopping mall simulation, entrance zones manage the schedule of entering consumer agents and a region where an entrance is located generates an agent at the entering time given in the schedule. During the simulation, agents move among zones. If an agent moving between zones belongs to different regions (X10 Places or nodes), the transfer of the agent occurs. The source region sends a control message to move the agent through the framework and delete the agent from the region. By using the serialized data of the agent in the control message, the framework restores the agent at the destination node with the same data as before. In the simulation, each agent determines an action based on a behavior model (e.g. walk and purchase models) and interacts with other agents in the same zone. In the shopping mall simulation, the main interactions of agents are avoidance of the walk model and probability changes in the purchase model that depend on the relative position and speed of other agents.

In the traditional method, the transfer of agents among X10 Places requires message transaction with deep copying of large data.

We can classify the following three types of data in an agent:

1. data for determining agent behavior (personality)
2. data for agent interaction (relationship)
3. data for logging and statistics (record)

With advances in the intelligent agent models for reproducing human society, the size of such data becomes larger to maintain complex preferences, demographic information, and historical records required for heterogeneous, intelligent, and learning-behavior models.

To decrease the size of the data transferred among distributed nodes and improve simulation performance, our proposed method involves a shadow agent that has only information required to compute an interaction (for example, position and speed). With our method, all regions have all zones. There is no transfer of an agent but only projection of the shadow agent to other X10 Places using broadcast messages. The relationship data in the shadow agent is much less than the total data of the substantial agent including all personality, relationship, and record data. We project agents' shadows into other X10 Places where relationship data are required to calculate interactions.

The range of shadow data projection can be reduced if there is no agent to interact with in the corresponding Zones of other nodes. Of course, if there is no agent in the current zone, no message is needed for sending or receiving from or to the current zone.

When receiving the package of shadow information, the Mall at a node delivers the information to corresponding zones in this node, and the agents in the zones use the information of shadow agents together with real agents in the node to compute behavior such as avoidance or reaction regarding invasion of personal space.

The Figure 4 and 5 show the comparison between the traditional agent transfer and a shadow agent. There are two nodes with different colors for two zones. With the agent transfer method, each zone belongs to only one node. When an agent changes its location from one zone to another, the transfer between two nodes also occurs.

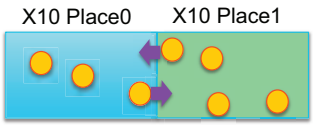

Figure 4: Agent transfer method with two zones.

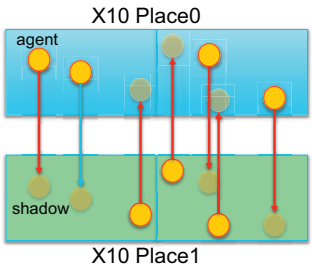

Figure 5: Shadow agent method with two zones.

On the other hand, both zones belong to both nodes with our shadow agent method. In both nodes, the agent can change its location between zones without agent transfer (stay in each node). However, the shadow information should be updated to other nodes at each timestep of the simulation. With the shadow agents, it looks like all agents exist in each node from the viewpoint of an agent. However, actions adapt for behavior models to only the real agents in the zone. The shadow agent only has shadow information to be used by the real agent for decision and does not execute actions with thread for computation. Hence, the calculation time is not affected by the node distribution.

Both methods use message transaction to bring the information of agents to other nodes. However, the amount of data and frequency of transactions are different and there is a trade-off. We discuss this trade off with a simplified setting in the next section.

In addition to the communication time between nodes for such a large amount of data, copying and restoring the whole agent instance incurs cost.

Figure 6 shows screen shot of shopping mall simulation with agent transfer method. The eight colors of dots indicate the X10 Place where agents belongs to.

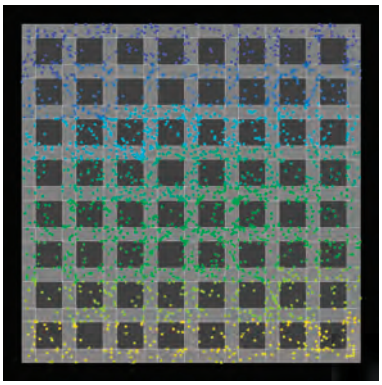

Figure 6: Screen shot of shopping mall simulation with agent transfer method.

Also, Figure 7 shows screen shot of shopping mall simulation with shadow agent method. We can see different color of dots (agents in different X10 Places) exists in the same zone.

4 SIMPLE ESTIMATION OF TRADE-OFF

We now evaluate the trade-off of the two methods described in the previous section with a simplified setting. For simplicity, we consider only the data size and number of message transactions with two nodes (not considering computation time).

As the common definition for the number of agents, we denote the number of X10 Places (nodes) as N_X and the number of agents per X10 Place as N_P. Therefore, the total number of agents $N_A = N_X N_P$ in the distributed nodes.

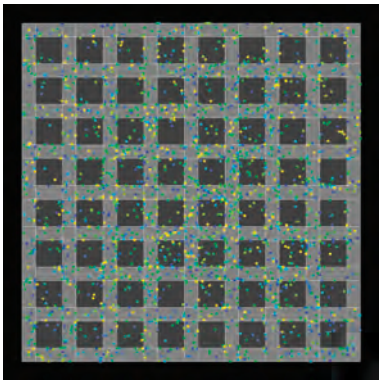

Figure 7: Screen shot of shopping mall simulation with shadow agent method.

We now estimate the amount of data communications with our shadow agent method.

The number of shadow agents projected from each X10 Place is $N_S = N_P = N_A/N_X$. When we set the data size of one shadow agent to D_S, the amount of data transactions for our shadow agent method at each timestep is as follows:

$$\begin{aligned} T_S &= D_S N_S N_X (N_X - 1) \\ &= D_S N_A (N_X - 1). \end{aligned} \tag{13}$$

Next, we estimate the amount of data transactions with the traditional agent transfer method.

As discussed in the previous section, the data size of one agent $D_T \gg D_S$

The number of transferred agents at each timestep is $N_T = \alpha N_X N_A$, where we define the ratio of the boundary area for moving into another node compared to whole geographical map used for the simulation as α.

Agents located in the boundary area are candidates for moving agents in the next step. The total boundary area for all nodes is smaller than the total area $\alpha N_X < 1$. This αN_X depends on the scale of the geographical map (for example, nation-wide area or one building) and the speed of agents (vehicle or pedestrian).

Hence, the amount of data transferred with this traditional method T_T is as follows:

$$T_T = \alpha D_T N_X N_A. \tag{14}$$

Finally, the comparison of the amount of data transactions with the two methods is shown as

$$T_S : T_T = D_S(N_X - 1) : \alpha D_T N_X. \tag{15}$$

This describes the trade-off between data size for one agent and frequency. Our method is effective when $D_S \ll D_T$ and α is not too small.

With this simple estimation, we can assume that the number of X10 Places has little effect and the data size, and the ratio of the moving boundary to whole map determines the effectiveness of our shadow agent method compared with the traditional agent transfer method. In the next section, we examine the performance of the simulation by changing the data size. For the nation-wide traffic simulation, the data size of each vehicle is small and the moving ratio (typically roads crossing the boundary of states or prefectures) is very small. In this case, the traditional agent transfer method will be more effective than our shadow agent method. For the crowded shopping mall simulation, the data size of a shopping agent with preferences and carts is large and the moving ratio (typically the boundary area of shopping areas) is relatively large. In this case, we can expect that our shadow agent method is more effective than the traditional agent transfer method.

5 COMPARISON OF SIMULATION PERFORMANCE

To compare the performance of the traditional agent transfer method and our shadow agent method, we evaluated the execution time (ms) of the simulation for 3,600 timesteps with 10,000 agents on 8 distributed nodes (8 X10 Places) by changing the conditions of data size. The compute nodes are NeXtScale nx360 M4 machines connected with 10 GbE and Infiniband. The data size of agents that include preference for products and shops and historical records can vary widely. For simple evaluation, we put dummy data array on the agents with double values and changes the size of array up to 10,000. In the case of the shopping mall simulation, this data size mainly corresponds to the attraction values for products (e.g. 100 products at 100 shops requires 10,000 attraction values).

The results are shown in Figure 8.

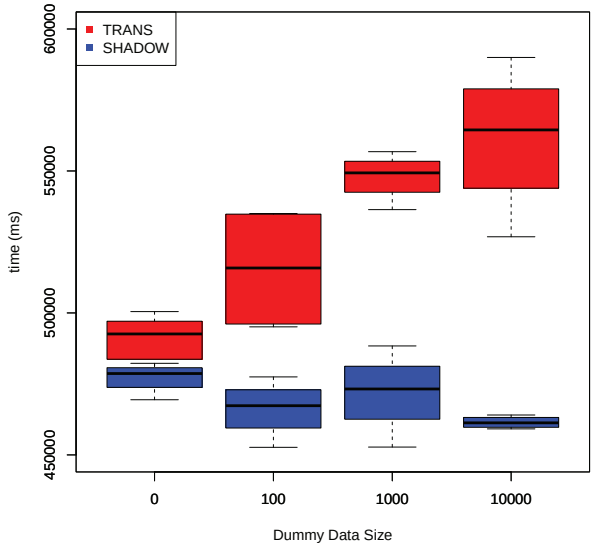

Figure 8: Execution time (ms) comparison between shadow agent (SHADOW) and agent transfer method (TRANS) with dummy data.

We can see the effect of data size as estimated in the previous simple calculation. Our shadow agent method performed better for almost all conditions. While the execution time under the traditional approach consistently increases with data size, the execution time for our shadow agent method has only small fluctuation due to computing environment and does not increase with data size. This is because our method only broadcast the fixed set of data (position and speed) required for interaction and the total size of data including history or preference possessed by agent does not affect the execution time. Though we only executed the simulation for 3,600 timesteps for the evaluation, it is required to simulate whole day or week activities with various combinations of situations. Hence the difference of the performance is essential for real use cases.

In addition to the data size of agents, we can consider that the balanced number of agents in each node is also bring this advantage of the shadow agent method. It is difficult to balance the number of agents among nodes with agent transfer method.

We also evaluate the scalability by using different number of nodes without dummy data (Figure 9). Both methods show improvement up to 8 nodes, but saturate with 16 nodes. In Figure 9, the performance of two method are almost same at each node size and the traditional method can show better performance

than our method. The reason is that the agent does not have large data which is given at the previous comparison (Figure 8) to concentrate the discussion on the scalability.

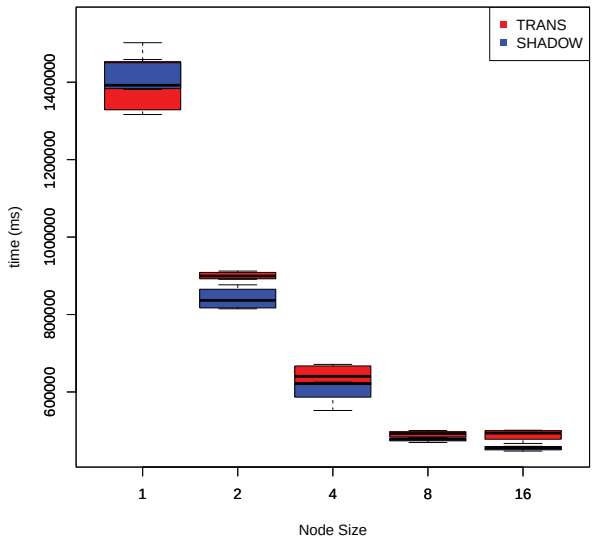

Figure 9: Execution time (ms) comparison between shadow agent (SHADOW) and agent transfer method (TRANS) with different node sizes.

6 CONCLUSIONS

In this paper, we introduced a shopping mall simulation with walking and purchasing behavior of consumer agents on the distributed agent framework XASDI for an example of social simulation and implemented the shadow agent method to improve the performance in multi nodes (X10 Place) environment. The distributed parallel execution of large-scale agent-based social simulations is very important for evaluating the complex social interaction of an enormous number of people and situation with acceptable performance to support the decision making of city planners.

To evaluate complex behavior with interaction among heterogeneous peoples in a large city, such a social simulation need to manage millions of agents with various behavior models and preferences, and the computation speed is also required to analyze enormous combination of possible situations and strategies with repeated simulations. But for the realistic behavior based on real data, social agents tend to contain a large amount of data including demographics, preferences, and history, and the transfer of such an agent incurs a heavy communication cost that has an adverse effect on performance. To improve the performance of such a distributed large-scale social simulation with complex agents, we introduced a shadow agent that is a lightweight entity projected among nodes with only required information such as the position and speed required to calculate congestion and avoidance interaction between agents.

In this paper, we evaluated the execution time for the traditional agent transfer method and our shadow agent method with changing conditions by using the shopping mall simulator. As we estimated on the trade off of data size and transaction frequency, the advantage of the shadow agent method increased significantly when the data size of each agent glowed. Though we used the dummy data for simplicity, it is natural for consumer agents to have such data for heterogeneous preferences of shops and goods in a mall and shopping record during the stay.

ACKNOWLEDGEMENTS

This work was supported by CREST, JST.

REFERENCES

Bansal, A. K. 2006. "Incorporating Fault Tolerance in Distributed Agent Based Systems by Simulating Bio-Computing Model of Stress Pathways". In *Proc. of SPIE Vol*, Volume 6201, 620108–1.

Chen, B., and H. H. Cheng. 2010, June. "A Review of the Applications of Agent Technology in Traffic and Transportation Systems". *IEEE Transactions on Intelligent Transportation Systems* 11 (2): 485–497.

Collier, N., and M. North. 2013. "Parallel agent-based simulation with Repast for High Performance Computing". *Simulation* 89 (10): 1215–1235.

Dijkstra, E. W. 1959. "A Note on Two Problems in Connexion with Graphs". *Numerische Mathematik* 1 (1): 269–271.

Hui, S. K., E. T. Bradlow, and P. S. Fader. 2009. "Testing Behavioral Hypotheses Using an Integrated Model of Grocery Store Shopping Path and Purchase Behavior". *Journal of consumer research* 36 (3): 478–493.

Mizuta, H. 2015. "Evaluation of Metropolitan Traffic Flow with Agent-based Traffic Simulator and Approximated Vehicle Behavior Model Near Intersections". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 3925–3936. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Mizuta, T., K. Izumi, I. Yagi, and S. Yoshimura. 2013. "Design of Financial Market Regulations against Large Price Fluctuations Using by Artificial Market Simulations". *Journal of Mathematical Finance* 3:15–22.

Noda, I., N. Ito, K. Izumi, T. Yamashita, H. Mizuta, T. Kamada, Y. Murase, S. Yoshihama, and H. Hattori. 2015. "Roadmap for Multiagent Social Simulation on HPC". In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, Volume 3, 22–25. IEEE.

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* 57 (5): 6–1.

Reynolds, C. W. 1999. "Steering Behaviors for Autonomous Characters". In *Game Developers Conference*, Volume 1999, 763–782.

Suzumura, T., S. Kato, T. Imamichi, M. Takeuchi, H. Kanezashi, T. Ide, and T. Onodera. 2012. "X10-based Massive Parallel Large-Scale Traffic Flow Simulation". In *Proceedings of the 2012 ACM SIGPLAN X10 Workshop*, 3. ACM.

X10. "The X10 Parallel Programming Language". http://x10-lang.org/. Accessed: 2017-04-27.

XASDI. "X10-based Agent Simulation on Distributed Infrastructure (XASDI)". https://github.com/x10-lang/xasdi. Accessed: 2017-04-27.

Yamashita, T., H. Matsushima, and I. Noda. 2014. "Exhaustive Analysis with a Pedestrian Simulation Environment for Assistant of Evacuation Planning". *Transportation Research Procedia* 2:264 – 272.

Zanlungo, F., T. Ikeda, and T. Kanda. 2011. "Social Force Model with Explicit Collision Prediction". *EPL (Europhysics Letters)* 93 (6): 68005.

AUTHOR BIOGRAPHIES

HIDEYUKI MIZUTA is a Research Staff Member of IBM Research Tokyo. He received B.S., M.S., and Ph.D. degrees in Physics from the University of Tokyo. He is a member of IPSJ and ACM SIGSIM. His research interests include dynamic economic-social systems with heterogeneous agents, SSME (Services Science, Management and Engineering) and smarter cities. His email address is e28193@jp.ibm.com.