PAPER

An Improved Platform for Multi-Agent Based Stock Market Simulation in Distributed Environment

Ce YU[†a)], ***Nonmember***, **Xiang CHEN**[†], ***Member***, **Chunyu WANG**[†], **Hutong WU**[†], **Jizhou SUN**[†], **Yuelei LI**[††], ***and*** **Xiaotao ZHANG**[††], ***Nonmembers***

SUMMARY Multi-agent based simulation has been widely used in behavior finance, and several single-processed simulation platforms with Agent-Based Modeling (ABM) have been proposed. However, traditional simulations of stock markets on single processed computers are limited by the computing capability since financial researchers need larger and larger number of agents and more and more rounds to evolve agents' intelligence and get more efficient data. This paper introduces a distributed multi-agent simulation platform, named PSSPAM, for stock market simulation focusing on large scale of parallel agents, communication system and simulation scheduling. A logical architecture for distributed artificial stock market simulation is proposed, containing four loosely coupled modules: agent module, market module, communication system and user interface. With the customizable trading strategies inside, agents are deployed to multiple computing nodes. Agents exchange messages with each other and with the market based on a customizable network topology through a uniform communication system. With a large number of agent threads, the round scheduling strategy is used during the simulation, and a worker pool is applied in the market module. Financial researchers can design their own financial models and run the simulation through the user interface, without caring about the complexity of parallelization and related problems. Two groups of experiments are conducted, one with internal communication between agents and the other without communication between agents, to verify PSSPAM to be compatible with the data from Euronext-NYSE. And the platform shows fair scalability and performance under different parallelism configurations.

key words: *distributed stock market simulation, agent-based modeling, round scheduling, distributed*

1. Introduction

The multi-agent based simulation has become an important method in behavioral finance to study the micro structures and aggregate macro features of stock markets. Stock markets can be regarded as complex adaptive systems described by a large number of variables, which are in turn influenced by an even larger number of factors or investors [1]. And multi-agent simulation is an efficient method to simulate complex adaptive systems. In a multi-agent simulation system, agent is used to denote a hardware or (more usually) software-based computer system that enjoys the properties of autonomy, social ability, reactivity, pro-activeness and so on [2]. The potential system-level consequences of financial markets are reflected through the behaviors of sets of agents [3]. Agents in a Multi-Agent System (MAS) are not independent, they are social-able in that they interact with other agents via some kinds of agent communication languages, also they perceive and react to their environments [4]. The communication between agents is usually based on a social network, and there are already some artificial stock markets using social networks as the communication topology between agents, such as SimStockExchange [5]. Agent-Based Modeling (ABM) has been applied in some simulations of stock markets [6], [7], but the scheduling of agents in these simulations is sequential, which is not corresponding with the situation in the real financial markets where the agents (investors participating in stock market) think and behave concurrently. Besides, with the development of behavioral finance, it is needed to expand the number of agents in a stock market to achieve breakthrough research results. But when the number of agents is getting larger, these sequential simulation systems will cost much time for the experiments and the time grows in exponential speed with the number of agents. So we adapt traditional ABM method to parallel multi-agent simulation, which has been applied in some complex adaptive systems [8]. This paper represents an improved multi-agent simulation platform based on PSSPAM [9] (Platform for Stock market Simulation with Parallel Agent-based Modeling) to support the stock market simulation with large number of parallel agents. The platform is designed for distributed environments with multiple computing nodes, which can provide much greater computing capability than a single computer. With the increasing number of agents, the improved PSSPAM shows a good extensibility. Additionally, the improved PSSPAM maintains a social network which defines the topology of agent relationship, and it provides a communication system for agents to communicate with each other asynchronously based on the social network. Besides, a distributed scheduling strategy based on round is represented. Also the improved PSSPAM supports easy customization for new financial models provided by users, preventing them from being trapped in complex computer related stuff and parallel programming.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 introduces the architecture of the improved PSSPAM. Section 4 presents the experiments evaluating this platform. And in Sect. 5, we summarize the current work and suggest questions for the

Manuscript received February 12, 2015.
Manuscript revised May 22, 2015.
Manuscript publicized June 25, 2015.

[†]The authors are with School of Computer Science and Technology, Tianjin University, Tianjin, China.
[††]The authors are with College of Economics and Management, Tianjin University, Tianjin, China.
a) E-mail: yuce@tju.edu.cn

DOI: 10.1587/transinf.2015EDP7050

future.

2. Related Work

In the early 1990s, the Santa Fe Institution Artificial Stock Market (SFI-ASM), developed by LeBaron et al. [10], had largely promoted the development of Agent-Based Modeling and multi-agent simulation. With SFI-ASM, most of the artificial stock markets' features are malleable and can be changed to carry out different experiments. However, SFI-ASM is not of extensible structures and the financial researchers have to change a lot of the source code to achieve their own market model, since SFI-ASM is developed for general usage of all kinds of multi-agent systems. Additionally, running on a single processor restricts the number of agents in the artificial stock market. And all the agents in SFI-ASM are of sequential manner, which cannot reflect the concurrent behaviors of the investors in the real world. Up to now, there are many researches or improvements on SFI-ASM [11], [12], but these works are launched in financial perspective, and cannot solve the problems above. Many other MASs have been proposed after SFI-ASM, such as SWARM [13], Repast, MASON, Ascape, starLogo, Aspen and so on, all of which are single-processed, they share the same problem of non-concurrency and small number of agents.

With the development of distributed computing, several distributed multi-agent systems have been proposed, like RepastHPC [3], ATOM [14] and D-MASON [15]. RepastHPC (Repast for High Performance Computing) is a toolkit for parallel agent-based modeling in distributed environments, which is improved on the base of Repast (Recursive Porous Agent Simulation Toolkit). RepastHPC is a useful and usable framework, a complete ABM simulation platform developed explicitly for large scale distributed computing systems that leverages modern C++ techniques and the ReLogo language [3]. D-MASON is a parallel version of MASON, a library for writing and running Agent-based simulations [15]. Besides RepastHPC and D-MASON, there are several other works [16], [17] presenting the toolkits for general parallel agent-based modeling. These platforms are not specialized in financial fields, and building an artificial stock market with these toolkits seems complex for financial experts who just want to focus on financial items.

Social networks are needed in an artificial stock market. N. Collier [3] have mentioned three kinds of relationship between agents in multi-agent systems: grid relationship, continuous space relationship and networks relationship. For ASMs, social network relationship is often used, since ASM is a particular simulation of sociology, and social networks can reflect the relationship of real stock investors better than generic networks or graphs. In recent social applications of multi-agent systems, agents have been organized into social networks [18]–[21], which is called multi-agent systems in social networks (MAS-SN) [22]. It is a necessity for agents to communicate with each other, and a social network is needed in an artificial stock market to define the communication topology of agents.

In distributed MASs, agents are concurrent and autonomous, with large number of agents, a scheduling algorithm is needed. In the real stock market, the natural time is used. However, in the simulation systems, it's hard to map the machine time to natural time. Lamport [23] proposed a strategy which uses a discrete event simulation [24] to simulate the continuous time of a simulation system. SWARM [13] is a single processed multi-agent simulation system, which uses the discrete event simulation strategy and presented a concept called "round". Round can be treated like a time clock. The behaviors of agents that take place in the same round can be regarded as that these agents and their behaviors are running at the same time clock. Other single processed simulation systems, like Aspen, Repast and so on, also use the round strategy to schedule the agents, which is called SWARM-like scheduling. And this leads to the result that most of the agent based financial models are based on round scheduling simulation systems. Artificial Open Market (ATOM) is a highly flexible agent-based model of distributed financial markets in an API form [14]. ATOM is also a round based distributed simulation system, but it stresses too much on the equity among all the agents. In ATOM, each agent sends at most one order during a "round table discussion" [14], which makes agents behave in a synchronized way. While in the real world, traders behave concurrently and independently, and the performance of ATOM is seriously affected by the "round table discussion" strategy.

In this paper, an improved PSSPAM based on [9] is represented. We introduce parallel agents into PSSPAM to mimic the concurrent features of real traders. PSSPAM builds the basic skeleton of a distributed stock market simulation. It can support large number of parallel agents, and at the same time, provides the interface for financial researchers to easily extend the simulation with their own financial models or algorithms. Also, PSSPAM introduces a communication system for agents to communicate with each other and with the market. Furthermore, it maintains a network topology for communication and schedules the agent threads based on the round strategy.

3. PSSPAM Platform for Distributed Environments

3.1 Logical Architecture of PSSPAM

There are four loosely coupled modules in the PSSPAM platform, as shown in Fig. 1, namely communication system, agent module, market module, and the user interface module. The communication system provides a stable channel for agents to communicate with each other and with the market. It is a basic layer for the platform, with its main functions of forwarding messages stably and asynchronously. The agent module contains a collection of all the agents. Agents reside on different nodes, and the number of agents on each node is determined by the control ar-

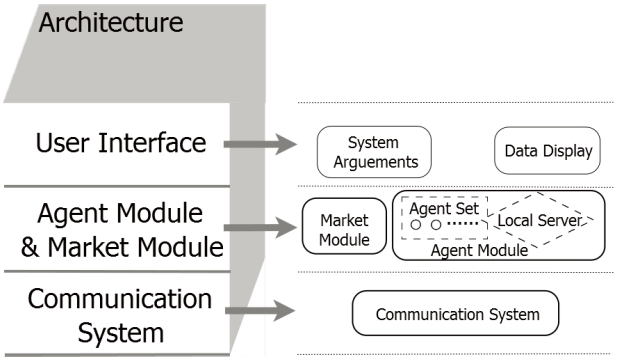


Fig. 1 Logical architecture of PSSPAM.

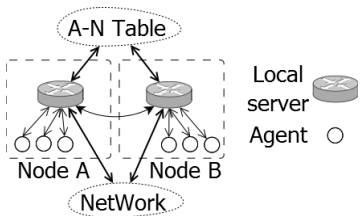


Fig. 2 Communication scheme.

guments from the user interface. The market module is a model of the stock market in the real world, such as Shanghai Stock Exchange. The platform is fairly extensible by supporting customization in each of the three modules. The user interface is the top layer to handle the interaction with users, such as configuration and execution of simulations.

3.2 Communication System

In distributed environments, it is difficult for agents to communicate with each other and with the market directly, since they are distributed on different nodes. An efficient communication system is introduced for communication without the knowledge of deployment and physical information of the message detination. On each node there is a local server responsible for forwarding the messages of local agents, and all the local servers communicate with each other in a point-to-point mode. Also, the market module contains a local server. The communication scheme is depicted in Fig. 2. All local servers maintain a network and an A-N table, where the network defines the communication topology of the agents and the A-N table tells which node each agent lies on. The network can be represented by an undirected graph and the scale of a network increases with a speed of n^2 as the number of agent denoted by n increases. Only these agents who have a link in the network model can communicate with each other. Also, the network can be customized by the financial researchers. All networks and A-N tables on different nodes are synchronized to stay the same.

During the delivery process of a message, the local server is responsible to parse the destination of the messages. As shown in Fig. 3, the destination of the message has two types. The first type is <*agentID*>, which is used

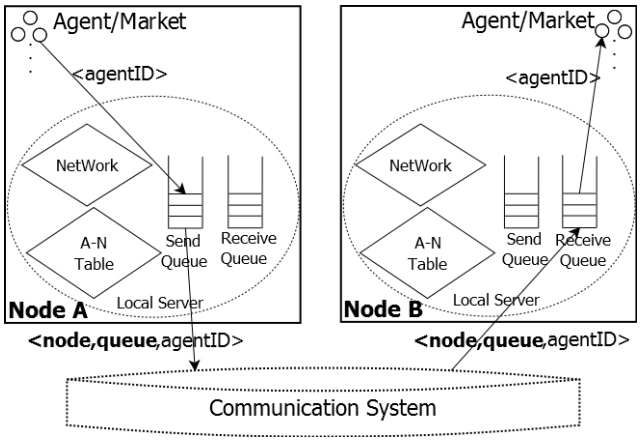


Fig. 3 Communication system.

in the agent module. As for the market, it has a unique ID to identify itself. When the messages come to the communication system, the local server will parse the destination to the second type of <*node*, *queue*, *AgentID*> according to the A-N table, where "node" is the name of the node that the destination agent locates on, and "queue" refers to the receive queue of the destination node. Also, a user can define customized types of messages. After the message arrives at the destination node, a reverse parsing process will be done to get the ID of the destination: an agent or the market, and then the destination gets the message. With this mechanism of delivery, the physical layer is transparent to the agent module and market module, thus the two modules are physical environment independent.

3.3 Agent Module

The agent module consists of a set of agents that reside on distributed computing nodes. In PSSPAM, we define the agents as investor agents (ItAgent), a simulation for real investors. A ItAgant contains its life kernel, decision algorithm, financial model and its internal state. The kernel part specifies the activity flow of an ItAgent during its lifetime. Generally, in the lifetime of an ItAgent, it continuously repeat the course: making decisions and behaving according to the result of the decisions to submit an order or get real time data of the market. Decision algorithm is a strategy that an agent takes to decide whether to trade or not, as well as the trading price and trading volume. The Decision algorithm can be a random decision algorithm, a genetic algorithm, a reinforce learning algorithm or other algorithms. Financial model is the micro models defined by financial researchers. The internal state is formed of its history trading data, assets and so on, and it evolves during the lifetime of ItAgent, which is a concrete manifestation of self-adaptation. Usually an agent achieves self-adaption through continuous learning, and the learning process embodies in the decision procedure.

3.4 Market Module

The market module is a simple model of the real exchange market. There are four separate areas in the market module, namely register area, agent response area, local server and trading area. The register area maintains a global ID table, and allocates a unique ID for each agent participated in the market. Agent response area is responsible for all the requests from agents. It calls the corresponding handler to handle the requests. Local server stores the social network and A-N table, also receives messages from agents and forwards messages back. Trading area matches orders submitted by agents. Trading area predefines two kinds of order auction mechanism, namely Call Auction mechanism and Continuous Auction mechanism, also the auction mechanism is under user customization. All these four areas are customizable for users.

Agent Response Area. Messages sent to the market are treated as requests, and the agent response area is to handle all the resquests. Due to the large number of parallel agents, there will be large number of requests queuing in the receive queue of the market. To deal with the large number of requests efficiently, we introduce a worker pool to agent response area, as is shown in Fig. 4. Each time a request is fetched from the receive queue, the area will take out a worker from the worker pool to deal with this request, and the worker invokes corresponding handler to handle the request. Predefined handlers in the system include register handler and order handler. Users can define their customized request handlers to coordinate with customized requests from the agent module.

3.5 Scheduling

When simulating, a scheduling algorithm is needed to arrange these agent threads and run the agents' behaviors concurrently. Most of the single processed simulation systems, like SWARM, Aspen, Repast and so on, use the round strategy to schedule the agents, which lead to the result that most of the agent based financial models are based on round scheduling. In order to support these previous ABM financial models, in PSSPAM, we as well use the round strategy to simulate the time clock of the real stock market. A round can be treated like a time clock. Agents' behaviors and decisions all run in rounds, and these behaviors that run in the same round can be treated as they are running at the same time clock. Orders submitted in the same round can be regarded as they are created and submitted at the same time clock. The time clock of PSSPAM is simulated as the round goes. According to the financial model, a round can be treated as a discrete timestamp in the real stock market, for example, several rounds can form a trading day. In fact, round can be regarded as a discrete simulation startegy to simulate the continuous time of multi-agent systems. Then, how to achieve a global round in the distributed environment? Every local server on each computing node maintains a local round, and the market maintains the center round. All these rounds should be synchronized at the same value, and the center round is responsible for the synchronization. As Fig. 5 shows, here we've got three computing nodes, one for the market module, two for the agents. On each computing node a round is maintained. The market maintains the center round, and other local servers maintain the local round, all these rounds are synchronized at the same value. When starting a round, the center round sends the round value to the local rounds, and each local server on agent computing nodes starts the agent threads concurrently and in a random order. The behaviors of agents that are scheduled at this round start to run. For the behaviors, agents can communicate with other agents, decide weather to buy or sell stocks, submit orders to the market through the communication module, or take other customized actions just according to the financial model. And there is no certain orders for agents' behaviors, since all these behaviors are customizable for users. During each round, the market module waits for submitted orders, matches them and sends the matching results back to agents through the communication system. Additionally, market-clearing can happen at the end of each round, or after the arrival of each order, or at other time clock, which just can be customized by users according to the financial models. When all agents end their behaviors, the local server send back the round message to tell the center local server this computing node has finished this round. When all local servers have finished this round, the next round starts. In this scheduling step, the simulation advances with the round goes.

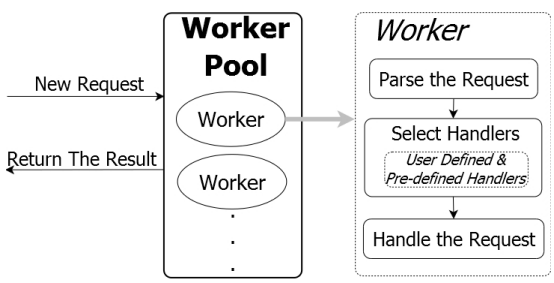


Fig. 4 Agent response area.

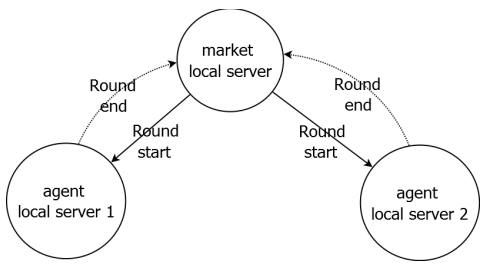


Fig. 5 Round scheduling.

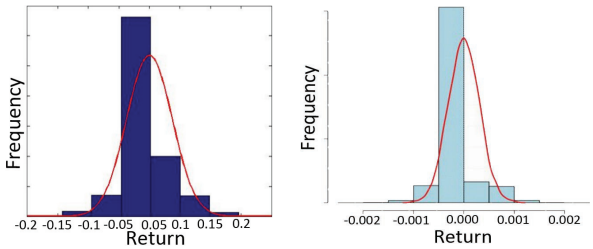
4. Experiments

Two groups of experiments are conducted with PSSPAM, one without internal communication between agents, and the other with internal communication between agents. The experiments without communication mainly have been shown in [9], but we have improved the original PSSPAM and conducted the experiments with internal communication between agents. In these experiments, the market is customized with Continuous Auction mechanism [25] and agents using a random decision-making strategy [25].

4.1 Experiments without Communication

The sample simulation of stock market with PSSPAM and without communication is executed on a Cluster with 5 computing nodes. Each node is a multicore server which has 4 800MHz AMD processors with 4 cores (Quad-Core AMD Opteron Processor 8374 HE), that is each computing nodes contains 16 cores. The operating system is CentOS release 5.8. The market is deployed on a single node, and the agents are deployed on other nodes, which is indicated by the graphic user interface. In this group of experiments, no communication between agents is assumed.

Validity Verification. First of all, we need to conduct experiments to verify the validity of the simulation platform, that is whether PSSPAM can generate the major stylized facts that are usually found in the real-world stock markets. These stylized facts are reposted in [25], and for the sake of simplicity we only present a form of the classical departure from Normality of asset returns. The distribution of asset returns does not follow the normal distribution, but appears the sharp peaked and heavy tailed property [25]. Sharp peaked indicates that the peak value (frequency near mean returns) is higher than the theoretical value estimated with normal distribution, while heavy tailed means that the frequency at the end is also higher than the theoretical value estimated with normal distribution, indicating that low probability events are more likely to happen in the real world. Figure 6 (b) is the distribution of asset returns of a specific stock on Euronext-NYSE. The curve is the fitted result of asset returns using normal distribution and the histogram is the exact frequency distribution of asset returns, which shows a typical feature of sharp peaked and heavy tailed. Figure 6 (a) depicts the departure from normality of asset returns when using random strategy in PSSPAM. It also shows the sharp peaked and heavy tailed property, suggesting that PSSPAM produces stylized facts in line with those observed for a specific stock on Euronext-NYSE. This experiment results prove that PSSPAM is a valid mimic of stock markets. To make the verification more credible, some basic statistics are computed. Here, P(t) is used to denote the time-series data of the trading price. Then the logarithm returns can be computed as $R(t) = \ln(P(t)/P(t-1))$. Eviews is used to compute some basic statistics of the time-series data of R(t), shown in Fig. 7. The kurtosis of R(t) is 4.855, larger than the kurtosis of normal distribution 3, which can also prove the fat-tail feature of PSSPAM with no internal communication between agents.



(a) Departure from normality on PSSPAM (b) Departure from normality on Euronext-NYSE [25]

Fig. 6 PSSPAM produces stylized facts in line with those observed on Euronext-NYSE.

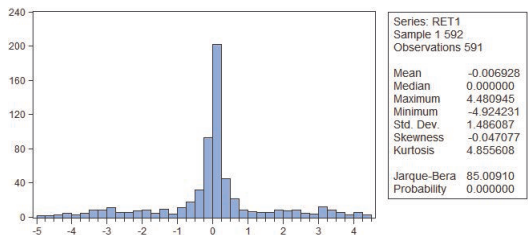


Fig. 7 The statistics of the data with no communication.

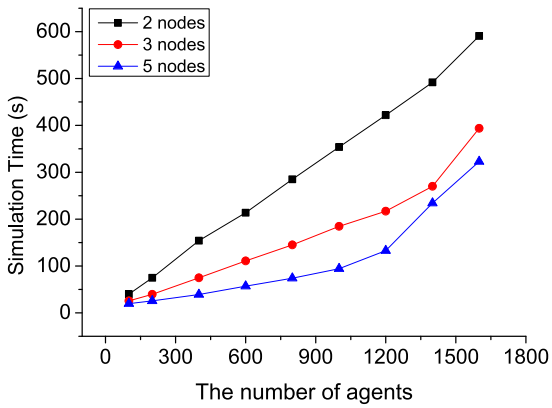


Fig. 8 Simulation time without thread pools, varying agent counts grouped by computing cores.

Scalability and Performance. A series of experiments are run to evaluate the performance and scalability of PSSPAM. In agent module, there are two methods to schedule the running of all the agent threads. The first one doesn't use thread pools, and the second one with thread pools. Without thread pools, each thread takes care of one agent and executing the kernel and behaviors of that agent. Then the threads are as many as the number of agents. When simulating in a round, all threads start to run at the same time. And when all agent threads finish, this round ends. To test the scalability of the simulation platform, we present the execution time for varying numbers of agents grouped by computing cores, depicted in Fig. 8. The abscissa is the total number of agents running in the simulation, representing the scale of the simu-

lation. For the same number of agents, the platform achieves a good time reduction with increasing computing nodes, and this trend keeps well as the number of agents increases. It also shows that PSSPAM scales weakly at a smaller number of agents but scales well at larger numbers. As with the increasing number of agents, the computing time instead of the cost of creating and scheduling threads conducts the execution time. This means that PSSPAM performs a good scalability in distributed environments. But when the number of agents exceeds 1600, there will be too much agents connecting the market at the same time, resulting in a bottleneck at the market module, since in old PSSPAM of [9] there is no worker pools in the market module. To overcome this problem, another method of using thread pool is proposed to coordinate all the agents.

Concerning the number of agents is large and the number of agent threads is also large in the first method, we can throw all agent threads into a thread pool in a random order. And this achieves the same results as the first method. With thread pools, the maximum number of agent threads is limited, which decreases the concurrency pressure of the market module and the time cost of thread creation and destruction. Each agent node mantains a thread pool. At the beginning of one round, all agent threads are thrown into a thread pool, and when all agent threads executed by the thread pools, this round ends. Figure 9 shows the comparison of the execution time at various numbers of agents respectively with the two methods. The size of the thread pool is set to 16 on each agent computing nodes, the same value as the number of cores on each node. When the number of agents is less than 1200, the execution time with each method is nearly the same and when the number is larger, the method using thread pools is more efficient than the first method. This results from that there is much less thread creation and scheduling consumption when using thread pools. And when the number of agents is small, the total simulation time is mainly donated by the computing cost and the thread consumptions can be neglected. But when the number of agents is large, these thread consumptions in the first method will be much greater than that in the method using thread pools. What's more, too much concurrent threads will bring heavy connection pressure to the market, which is more likely to make the market a bottleneck.

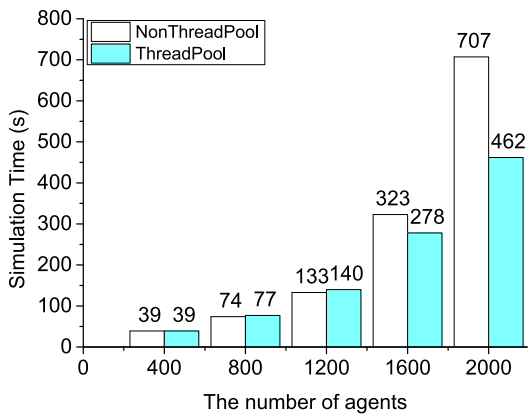


Fig. 9 Simulation time comparison of the two methods at various agent counts.

4.2 Experiments with Improved PSSPAM

Based on the platform represented in [9], we improved the platform for better performance. First of all, round scheduling strategy is used to schedule the running of agent threads. Secondly, in [9], each agent maintains an adjacency list to store the agents that are linked with itself in the network model, but this kind of memory storing strategy produces much redundancy. However, in the new strategy, each local server stores a copy of the network, and these copies are synchronized to be the same. Additionally, in [9], on each computing node, there exists an implementation of the communication system for the sake of fault tolerance. And the messages of these communication systems need to be synchronized, which substantially affected the performance of PSSPAM. Here, we use only one implementation of the communication system, and all messages between different nodes are forwarded by this implementation through the high-speed network of the cluster. This strategy improves the performance particularly well if there are a lot of messages to be forwarded. Lastly, when there are many threads running at the same time, meaning many threads connecting the market at the same time, the market will become a bottleneck of the platform. To work out this problem, we started a worker pool in the market module. For each request or submit from the agents, a worker thread is fetched from the worker pool, and after the execution of the handler, the worker thread is put back into the worker pool. Also, the size of the worked pool is a changeable parameter. In this group of experiments, the communication of agents is based on a small word network. The small world network is created using the NW model proposed in [26]. The parameter of the initial fixed range K is set to 1, and the parameter of connection property p is set to 0.8. Both values of the two parameters are set just based on some results of experiments. Also, these parameters are changeable for users, and users can even customize the type of network. For the network, a vertex represent an agent. A link between two vertexes means the two agents share a connection, and only agents with links between them can communicate. During each round, each agent sends a communication message to all of its linked agents, the message is a fixed string. The communication rules and the content of messages can also be customized by users according to the financial models. All these comparison experiments are conducted on the same cluster environment with the privious experiments.

Validity Verification. With the improvements and internal communication between agents, we should first of all verify the validity of the new platform, also using the sharp peaked and heavy tailed property. In this experiment, the simulation runs 10 rounds, the agent module with thread pools, the market module with worker pools. Both pool size are set to 16. With the data averaged for 10 times running

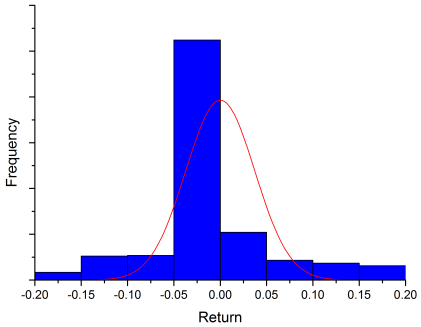


Fig. 10 Departure from normality on improved PSSPAM.

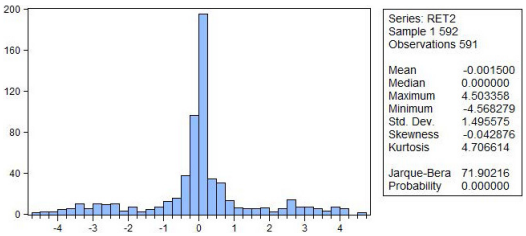


Fig. 11 The statistics of the data with communication.

of the simulation, we got Fig. 10, which indicates the validity of the new PSSPAM. Also, the kurtosis and other basic statistics of the logarithm returns R(t) with communication between agents are computed. As shown in Fig. 11. The kurtosis of R(t) is 4.7, larger than the kurtosis of normal distribution 3, which can also prove the fat-tail feature of PSSPAM with internal communication between agents.

Scalability and Performance. This set of experiments are conducted to explore the scalability and performance of the improved PSSPAM. The market is deployed on a single node, and agents are deployed on the other nodes. In agent module, the thread pool is used to run the agent threads, and in market module, worker pool is used as well. Both of these two pool sizes are set to 16, the same value as the number of cores on each computing nodes. PSSPAM is simulated for 10 rounds with various number of computing nodes and agents, and during each round, every agent sends a communication message to its linked agents. The simulation time against various numbers of computing nodes and agents is represented in Fig. 12. When the number of agents keeps unchanged, with the growing number of computing nodes, the simulation time decreases significantly, meaning improved PSSPAM has a good scalability for the distributed computing environment. With the number of computing nodes kept unchanged, when the number of agents grows, the size of the small world network grows. And with the same parameter of the initial fixed range K and the same parameter of connection property p for NW model, the total links between agents grows proportionally. And with the same communication rules for agents, the total messages communicated between agents increases linearly with the growing number of agents. Then the total computing time of agents' behaviors is growing and the total number of messages sent by agents

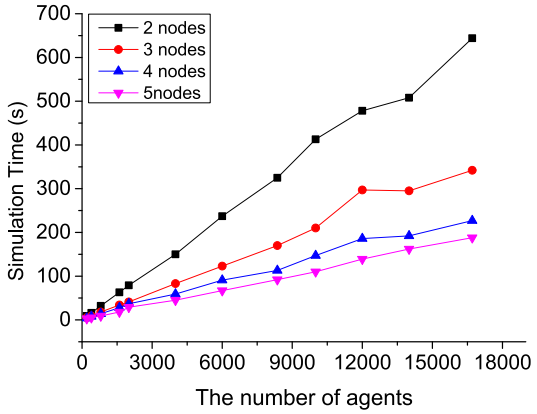


Fig. 12 Simulation time with varying agent counts grouped by computing cores.

is growing, so as to the total simulation time is growing. The simulation time is growing almost linearly with same computing nodes, as shown in Fig. 12, which denotes that the communication system is stable for the growing number of communication messages, the market with worker pool is no longer a bottleneck for large number of agents and PSSPAM has a stable and good scalability for the growing number of agents. In a word, PSSPAM presented a good and stable scalability for the distributed computing environment and the growing number of agents. Additionally, with the same number of agents, such as 1600, and run on the same number of computing nodes, such as 5 nodes, the simulation time of the improved PSSPAM is much less than the original PSSPAM, mostly because of the implementation of the communication system and the worker pool of the market module. With the original PSSPAM, on each computing node there exists an implementation of the communication system for the sake of fault tolerance, and the messages of all these communication systems need to be synchronized, which substantially affects the performance of PSSPAM espacially when the number of agents is large and the number of communictaion messages is large. However, only one implementation of the communication system is used in the improved PSSPAM, and all messages between different nodes are received and forwarded by this implementation through the high-speed network of the cluster, which highly improves the performance of PSSPAM. In the market module, RT is used to denotes the total time of receiving requests from agents, including the receiving time and the queueing time since the number of agents is very large and lots of requests arrive at the same time. HT is used to denote the total handling time of all requests, including the order-matching time, the data storing time, the forwarding time of result messages and so on. The total time cost of the market module can be roughly calculated as RT+HT. When many agents requesting from the market at the same time, the market module will be a bottleneck for the simulation system. With the worker pool, RT can not be reduced but HT can be cut down a lot. Then, the total time cost of the market module is reduced, and the performance of PSSPAM

is improved accordingly.

5. Summary and Future Work

This paper introduced the detailed architecture design of the improved PSSPAM (Platform of Stock market Simulation with Parallel Agent-based Modeling), which is based on agent-based modeling. The platform is designed for distributed environments which provide sufficient computing capability for the large number of parallel agents. A communication system is specially designed to support the interaction of agents and the market, and it is compatible with various communication network topologies that define the social relationship of all the agents. To arrange the running of the large number of agent threads, the round scheduling strategy is used. To support fair extensibility, PSSPAM allows users to customize the behaviors of agents, the network topology for agents, the message types of the communication system, the request handlers in the market module and so on. In addition, a series of experiments have been conducted to verify the correctness of the simulation and evaluate the performance and scalability of the platform.

Though with the correctness and efficiency, PSSPAM is still under improvement and more supplements are needed in the future. For instance, the agents are deployed on different computing nodes sequentially, load balancing has not been taken into consideration. And if the number of messages sent by agents are large enough, the performance of PSSPAM may be poor since there exists lots of inter-nodes communication messages. Fault tolerance hasn't been handled yet. The market module and the single implementation of the communication system may be the bottleneck for the fault tolerance. There is still much work to be done to enhance the system's robustness and performance. It is also hoped that more graphical tools can be provided to simplify users' customization.

Acknowledgments

The work is sponsored by the National Natural Science Foundation of China (71131007, 61303021).

References

[1] R. Lye, J.P.L. Tan, and S.A. Cheong, "Understanding agent-based models of financial markets: A bottom-up approach based on order parameters and phase diagrams," Physica A: Statistical Mechanics and Its Applications, vol.391, no.22, pp.5521–5531, Nov. 2012.

[2] M. Wooldridge and N.R. Jennings, "Intelligent agents: Theory and practice," The Knowledge Engineering Review, vol.10, no.2, pp.115–152, Jan. 1995.

[3] N. Collier and M. North, "Parallel agent-based simulation with repast for high performance computing," Simulation, vol.89, no.10, pp.1215–1235, Oct. 2013.

[4] M. Pipattanasomporn, H. Feroze, and S. Rahman, "Multi-agent systems in a distributed smart grid: Design and implementation," Proc. Power Systems Conference and Exposition (PSCE '09), IEEE/PES. pp.1–8, IEEE, March 2009.

[5] A.O.I. Hoffmann, W. Jager, and J.H. Von Eije, "Social simulation of stock markets: Taking it to the next level," Journal of Artificial Societies and Social Simulation, vol.10, no.2, March 2007.

[6] P.E. Johnson, "Agent-based modeling: What I learned from the artificial stock market," Social Science Computer Review, vol.20, no.2, pp.174–186, 2002.

[7] S.-H. Chen and C.-H. Yeh, "Evolving traders and the business school with genetic programming: A new architecture of the agent-based artificial stock market," Journal of Economic Dynamics and Control, vol.25, no.3, pp.363–393, March 2001.

[8] C. Deissenberg, S. Van Der Hoog, and H. Dawid, "EURACE: A massively parallel agent-based model of the European economy," Applied Mathematics and Computation, vol.204, no.2, pp.541–552, Oct. 2008.

[9] C. Wang, C. Yu, H. Wu, X. Chen Y. Li, and X. Zhang, "A platform for stock market simulation with distributed agent-based modeling," Algorithms and Architectures for Parallel Processing, Lecture Notes in Computer Science, vol.8631, pp.164–177, Springer International Publishing, 2014.

[10] B. LeBaron, "Building the Santa Fe artificial stock market," Physica A, June 2002.

[11] B. LeBaron, "Evolution and time horizons in an agent based stock market," Macroeconomic Dynamics, vol.5, no.2, pp.225–254, April 2001.

[12] N. Ehrentreich, "The Santa Fe artificial stock market re-examined — Suggested corrections," Martin-Luther-University Halle-Wittenberg, Wirtschaftswiss. Fak., 2002.

[13] N. Minar and R. Burkhart, ed., "The Swarm simulation system: A toolkit for building multi-agent simulations," Santa Fe Institute, Santa Fe, June 1996.

[14] P. Mathieu and O. Brandouy, "A generic architecture for realistic simulations of complex financial dynamics," Advances in Practical Applications of Agents and Multiagent Systems, Advances in Intelligent and Soft Computing, vol.70, pp.185–197, Springer, Berlin, Heidelberg, 2010.

[15] G. Cordasco, R. De Chiara, A. Mancuso, D. Mazzeo, V. Scarano, and C. Spagnuolo, "A framework for distributing agent-based simulations," Euro-Par 2011: Parallel Processing Workshops, Lecture Notes in Computer Science, vol.7155, pp.460–470, Springer, Berlin, Heidelberg, 2012.

[16] M. Kiran and P. Richmond, ed., "FLAME: Simulating large populations of agents on parallel hardware architectures," Proc. 9th International Conference on Autonomous Agents and Multiagent Systems, vol.1, pp.1633–1636, May 2010.

[17] M. Scheutz, P. Schermerhorn, R. Connaughaton, and A. Dingler, "SWAGES: An extendable distributed experimentation system for large-scale agent-based ALife simulations," Proc. Artificial Life X, pp.412–419, 2006.

[18] J. Sabater and C. Sierra, "Reputation and social network analysis in multi-agent systems," Proc. First International Joint Conference on Autonomous Agents and Multiagent Systems: part 1, pp.475–482, ACM, 2002.

[19] J.M. Pujol, R. Sangüesa, and J. Delgado, "Extracting reputation in multi agent systems by means of social network topology," Proc. First International Joint Conference on Autonomous Agents and Multiagent Systems: part 1, pp.467–474, ACM, 2002.

[20] M.M. de Weerdt, Y. Zhang, and T. Klos, "Multiagent task allocation in social networks," Autonomous Agents and Multi-Agent Systems, vol.25, no.1, pp.46–86, Feb. 2012.

[21] Y. Jiang, J. Hu, and D. Lin, "Decision making of networked multiagent systems for interaction structures," IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol.41, no.6, pp.1107–1121, March 2011.

[22] Y. Jiang, Y. Zhou, and W. Wang, "Task allocation for undependable multiagent systems in social networks," IEEE Trans. Parallel Distrib. Syst., vol.24, no.8, pp.1671–1681, Aug. 2013.

[23] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," Commun. ACM, vol.21, no.7, pp.558–565, July

1978.
[24] J. Misra, "Distributed discrete-event simulation," ACM Computing Surveys, vol.18, no.1, pp.39–65, March 1986.
[25] R. Cont, "Empirical properties of asset returns: Stylized facts and statistical issues," Quantitative Finance, vol.1, no.2, pp.223–236, March 2002.
[26] M.E.J. Newman and D.J. Watts, "Renormalization group analysis of the small-world network model," Phys. Lett. A, vol.263, no.4-6, pp.341–346, Dec. 1999.

Ce Yu received B.S., M.S. and Ph.D. degree in Computer Science and Technology from Tianjin University, Tianjin, China, in 2002, 2005 and 2009, respectively. He worked as a visiting scholar in Illinois Institute of Technology, IL USA, from Sept. 2011 to Feb. 2012. He is currently an associate professor in School of Computer Science and Technology, Tianjin University. His research interests include high performance computing, computational finance and massive data processing.

Xiang Chen received B.S. in School of Computer Software from Tianjin University, Tianjin, China, in 2013, and he is now a master student in School of Computer Software, Tianjin University. His research focus on multi-agent simulation, artificial stock market simulation and Behavioral Finance.

Chunyu Wang received B.S. and M.S. in Computer Science and Technology from Tianjin University, Tianjin, China, in 2012 and 2015. Her research focus on multi-agent simulation, artificial stock market simulation and Behavioral Finance.

Hutong Wu received the master degree in computer science from Tianjin University, China, in 2004. He works at the School of Computer Science and Technology, Tianjin University, China. His research interests include high performance computing, data visualization, and visual analytics.

Jizhou Sun received the master degree in computer science from Tianjin University, China, in 1982, and the Ph.D. degree in electrical engineering and computer science from Sussex University, United Kingdom, in 1995. Currently, he is working as a professor in computer science and technology, Tianjin University. His main research interests include parallel algorithms and architectures, high-performance computing, computer graphics, scientific visualization.

Yuelei Li is an assistant professor of finance in Tianjin University. His research focus on Agent-based Computational Finance, Behavioral Finance, and Asset Pricing.

Xiaotao Zhang is an associate professor of finance in Tianjin University. His research focus on Computational Finance and Behavioral Finance.