Simulating artificial stock markets with efficiency

Philippe MATHIEU and Yann SECQ

Introduction

Stock markets have been studied for years in economy and finance academic departments by relying on the idea of a general homo economicus that makes rational choices. Classical approaches use equational representation to enable a global markets characterization, but they fail to explain the link between individual behaviours and the global market dynamic and trends that emerge.

Schelling seminal work on segregation models [12] has initiated a novel approach relying on an individual behaviour characterization that leads to global observations. This approach enables a finer grain behaviour modelling that allows more detailed simulations. Within this context, it is possible to have several kind of agents, with heterogenous cooperative or concurrent behaviours, and to study their actions aggregation and feedback loops which produces emergent macroscopic behaviours. The trade-off between the expressivity available with an agent-based approach and its execution speed relies on the complexity and information volume of involved processes.

Our study is focused on order driven markets and describes why scalability issues appear when using an agent-based approach. After introducing our existing simulation platform ATOM, we study the impact of the number of orders and transactions and the cost of notification on execution time. These aspects lead us to search how we could optimize our platform to be able to handle higher volumes that are close to those generated by High Frequency Trading (HFT) algorithms.

First section studies problematics linked to order-driven market simulators in a centralized setting. Second section identifies problematics that have to be tackled to scale agent-based simulations. Third section presents the two main approaches to efficiently distribute stock market simulations: network-based or GPGPU-based. Last section concludes and identifies future works.

P. Mathieu and Y. Secq - Université Lille 1, Computer Science Dept. LIFL (UMR CNRS 8022)
e-mail: firstname.surname@univ-lille1.fr

1 Agent-based stock market simulation with ATOM framework

This section first introduce general notions on order-driven markets. Then, principles and dynamic of agent-based market simulators are described. These elements show why scaling can be a crucial issue with this type of approach and within this application domain. Then, section 2 study which solutions can be proposed to ease scaling issues in agent-based simulations.

In order-driven markets, price formation is based on the confrontation of a formalized offer and demand. This formalization is done through the emission of orders by traders that contain an asset name, a direction (bidding or selling), a quantity (number of shares) and a price. All orders for a given asset are gathered in an order book (generally a double auction order book) that is organised in two sets sorted on price: sell orders and buy orders. Price formation is realized by confronting the best ask order price with the best sell order price. Thus, in a stock market driven by orders, actors involved in price creation are traders that send ask or bid orders, order books that stores and generate prices (one for each negotiated share on the market) and the market that can be seen as the container for all these information exchanges (orders and quotes).

In an agent-based approach of stock markets simulation, traders are represented as agents, there are several order books (as much as the number of assets) and there is the market that establish the link between traders and order books. The order processing dynamic has two stages: in the first stage, traders send orders to the market, that forward them to the concerned order book, the second stage happens if a price is fixed. When a new price is created, it is sent to the market (to store price history) and traders whose orders where involved in the last quote have their portfolio and their balance updated accordingly to the exchanged shares and fixed prices. Traders behaviours can then react to these changes in the market leading to interesting feedback loops between agents strategies. This approach allows simulations with the finest possible grain, because all entities in the system are observable. Nevertheless, this granularity which is an interesting aspect that enables experimentations and observations close to real stock markets is also an important challenge to be able to scale these simulations.

The ATOM (ArTificial Open Market, `atom.univ-lille1.fr`) platform has been developed to easily build fine-grain stocks markets simulations. This platform is made of several layers that enable the definition of markets, simulations and to allow their execution. If needed, visualization and live interaction with a running simulation can be done. The main component of ATOM is a Java API that define all entities introduced in section **??**. This component includes detailed predefined object for orders (`Limit`, `Market`, `Cancel`, `Update`, `Stop`) and trading strategies (`ZIT`, `Chartist`, ...). Thus, a simulation can easily be built as shown in figure 1 (a more detailed description of the ATOM platformis available in [6]).

```
Simulation sim = new Simulation();
for (int i=0; i<10; i++) { // Adding 10 order books
  sim.addNewOrderBook("IBM_"+i);
}
for (int i=1;i<=1000;i++) { // Adding 1000 ZIT agents
  Agent trader = new ZIT("zit_"+i);
  sim.addNewAgent(trader);
  trader.account.credit(10000);
}
// 1000 days with 100 ticks (opening), 800 (continuous) and 100 (closing)
sim.run(1000,100,800,100);
```

Fig. 1 A simple intra/extra day simulation with ATOM

2 Scaling issues in agent-based simulations

The main principle of agent-based simulation (ABS) is its focus on individual behaviours. Thus, simulation involving a high number of agents or messages implies scalability issues. These issues can be categorized in two classes: volume handling (number of messages, information to be stored for post-analysis) and information exchange efficiency (latency between an update within the system and its notification to agents).

2.1 Handling the volume

The first problem that arise is information volume that is generated and that has to be processed to compute one simulation time step. In this specific application domain, information volume is function of the number of order books, of agents, of prices fixed and thus, of agent wealth updates. Each time that an agent send an order, it has to be processed by the market (stored and forwarded to the specified order book), then by the order book (insertion and matching), and if a price is fixed, the market must be informed (to keep price history) and then involved traders on this transaction have to be updated.

When one thousand agents and a dozen of order books are created (as in figure 1), it involves a very large number of orders, prices and portfolios to be managed. The cost is high memory but also in computation. In the above example, the number of orders is $1000*(100+800+100)*1000 = 10^9$, prices generated are in the order of 10^6 and the total time taken for this simulation is roughly one hour on a 2.66Ghz i7 computer. It is difficult to know the number of orders sent in a day on Euronext, but the Global Average Daily Volume is in the order of 8 to 10 millions prices fixed in a single day (Source) with an average order matching speed below 1 millisecond (Source).

2.2 Information exchange efficiency

The second important aspect is linked to the information exchange cost between traders and order books. There are mainly two main information flows: the incoming orders flowing from traders through the market to be handled by order books and an outgoing flow starting with a price fixed within an order book that has to be transmitted to the market, then to traders involved in the transaction (and more globally to all traders). A last information exchange that can happen between traders and the market is concerned with information on volumes and order books (bid-ask spread and first rows of best bid-ask orders waiting).

Traders in agent-based simulations use several information to compute their next action (do nothing or send an order): market (quotes, orders waiting in orderbooks, bid-ask spread) and social (news or messages from others agents). In function of their trading strategy (chartist, arbitragist ...), agents need to access more or less information and thus, the mechanism used to transmit them to traders becomes a bottleneck for the simulator. When the number of orders increases, the delay between an order emission and its execution notification (if another order can match it) increases also.

For real markets, we have not been able to find quantified information on the duration of such round trip message. But, with a matching process that is below 1 millisecond and a publicized data message latency given under 5 milliseconds (Source). we can infer that the whole process should not take more than 10-100 milliseconds. But in an artificial markets, this information exchange of a new quote generates more computation costs because agents portfolio have to be updated.

3 Agent-based simulator distribution

To handle scalability issues two main approaches can be used: distribute the load among a computer network or on a General Purpose Graphical Processor Unit (GPGPU). The first approach enables a scalability linked to the number of available computers while the second is highly dependant on the GPGPU design to be able to chain several GPGPU on a given host. A third hybrid approach using a network of computers using GPGPU is possible but raises others issues (see [13] for internet scale and [4] for local scale networks).

3.1 Distributing an artificial stock market

To be able to scale agent-based stock market simulations, we have seen that two main issues are involved: data volume storage and querying and information exchange speed between traders and order books. Visualisation has also to be taken

into account because it involves a process that gather an important information set. Even if visualisation can be deported, it involves a high cost in information transfer.

Several strategies can be used to distribute and parallelize market computations but one has to understand that there are three distinct problematics: computation costs involved by trading strategies, computation costs implied by order/price handling (emission, logging, insertion and matching, portfolio updating), communication latency occurring because of information exchange between traders and the market.

These problematics correspond to different kind of computations: heterogenous behaviours for trading strategies, homogenous computations for orders matching process and network infrastructure and libraries for communications. In the following sections, we explain how network-based distribution should be used to enable concurrent execution of heterogenous computations (ie. traders strategies) while GPGPU are fitted to intense data parallelism, and thus interesting for homogenous computations (ie. price fixing mechanism).

3.2 Should quotes be pushed or pulled?

The second problematic related to information exchange becomes critical in a distributed setting. Two main schemes can be used to transmit information between traders and the market: pulling or pushing. In a pulling scheme, traders initiate a request to the market to retrieve some information, while in a pushing scheme, the market sends the information to all traders. Pulling is efficient when trading strategies do not use a lot of information from the market, but when strategies needs information on multiple assets, with an important level of details (bid-ask spread, orders within order books ...), pushing the information is more efficient.

This choice has also important implications on reliability. Indeed when a pulling mechanism is used, denial of service attacks can be generated by malicious traders by overloading the market with information requests. Even with fair traders, if their trading strategy use an important information volume, it generates loads on the market to prepare and transfer all these information. It should be noted that in a distributed setting efficient group communication can be done thanks to publish/subscribe architectures, linked to networks broadcasting capabilities. For all these reason, we believe that pushing information is clearly more fitted in a network-based distribution. For GPGPU, this choice can be queried because memory latencies are really low.

3.3 Network-based distribution

Distributing computation on a computer network allow to harness power of commoditized computers network. The main idea is to enable concurrent computations

on each node and to use some message passing scheme to coordinate tasks. This approach is particularly fitted to heterogenous computations that can be expressed through task parallelism or embarrassingly parallel computations (as in [2]). The main limiting factor is communication latency cost. To bypass this limitation, distributed systems try to cover communication costs with a coarser grain of computation chunk. These costs have been clearly evaluated in [8], where large scale traffic simulations lead to a factor of 10 between a simulation on computers on a LAN and on parallel machines. An interesting comparison of some of the well known platforms (Repast, Cougaar, Aglets and AAA) can be found in [3].

Because heterogenous computations can be distributed, it is efficient to distribute traders among the network. Even if some agents share the same trading strategy, there are several strategies used during a simulation. This approach enables a concurrent computation of traders decisions. Difficulties that arise with this approach is the guaranty of equity between agents. In a centralized setting, equity can be easily enforced with a turn of speak. In a distributed setting it becomes harder, but it can be enforced by waiting that all traders have transmitted their action before integrating orders in the market. This synchronisation barrier decreases performances because some computers are idle while the market is waiting for all traders answers. An interesting use case that can leverage embarrassingly parallel computation is the execution of several simulations. Experimentations done in [7] to compute social welfare in function of orders permutations requires $orders!$ simulations that are all independent from each other. This is typically a context where network-based distribution is clearly fitted.

3.4 GPGPU-based distribution

To understand General Purpose Graphical Processing Unit (GPGPU) principles, concepts of SISD, SIMD and streams have to be detailed. A Single Instruction Single Data (SISD) can be seen has the default computation model, with one instruction applied to one data in a time step. Early in computer hardware history, the concept of Single Instruction Multiple Data (SIMD) has been developed in order to gain computation power through a parallel execution of the same instruction on multiple data.

Recently, these graphical processors that were tailored to graphical computations have been redesigned to allow also some more general computation scheme by relying on a stream processing approach. GPGPU processors allow scalability through the use of a large number of SIMD unit that can handle several parallel computation chunks (called *kernels*).

GPGPU computation distribution is fitted to data intensive processing, but on homogenous computations. The same algorithm can be applied in parallel to a large number of data chunks. The main difficulty is then to redefine computation in a stream oriented design to reach interesting performance gains. Works done on agent-based simulation on GPU (as [10] and [9]) demonstrate the speedup that can

be achieved but also illustrate difficulties and challenges to transform agent-based computations in a stream parallelism model.

In stock markets, we can identify several mechanisms that could benefit a data parallelism model: market trends characterization (as done in [5]) and price fixing mechanism. The first computation type, for example assets mean value on a given period or time series analysis, the benefit would come from being computed once and shared by all trading strategies. The same process being applied to several assets fit nicely with the stream processing model. The second computation type is harder to parallelize. Indeed, price are fixed by heterogenous information chunks (orders which varies in quantities, prices and assets). Nevertheless, an important time slice is taken by orders sorting in ask and bid data structures. This point could benefit from parallelized sorting algorithms.

A last important issue implied by GPGPU distribution is reproducibility. As demonstrated in [11], if computations are not rigorously checked when using floating point arithmetics, errors can grow rapidly and results become totally biased. This could be crucial if algorithms linked to market data aggregation are parallelized in order to provide on the shelves results to trading strategies.

Conclusion

This paper has described the main challenges that appear while simulating large order-driven stock markets with an agent-based approach. These problems have been identified through the use of our ATOM platform whose core components are executed on a single computer. We have identified two fundamental problematics that have to be tackled in order to scale simulations to a large number of orders: information volume and information exchange. These problematics are particularly critical to study the impact of High Frequency Trading algorithms that generate a high number of orders in short time slice. The first approach, distributing computa-

Network	GPGPU
heterogeneous computation	homogeneous computation
actor/concurrent model	stream processing model
network communication cost (high latency)	host/card communication costs (low latency)
high scalability (number of machines)	scalability limited to GPGPU chaining
language agnostic	C-99 (and OpenCL / CUDA)

Fig. 2 Network vs. GPGPU distribution

tion on a computer network; is fitted to scaling heterogeneous computations. These computations are typically trading strategies used by artificial agents. The main limiting criteria of this kind of distribution is network communication costs that have to be compensated by a high computation cost. The second approach relies on GPGPU optimization and thus is particularly fitted for homogeneous computations. We have

seen that data intensive computation that can be parallelized can leverage GPGPU computation power, but as with the network-based approach, the cost of information transfer between host and graphic card memory is the limiting factor.

We have begun experimentations to implement an hybrid approach that tries to leverage network and GPGPU approaches (in the same spirit as [1] but not focused on spatial based issues). We focus on distributing only trading strategies on a computer network and to using GPGPU cards to manage the market. The main difficulties are the network latency hiding, the parallelization of price fixing mechanism and determining an efficient price notification mechanism.

References

1. Aaby, B., Perumalla, K., Seal, S.: Efficient simulation of agent-based models on multi-gpu and multi-core clusters. In: 3rd International ICST Conference on Simulation Tools and Techniques, SIMUTools '10, pp. 29:1–29:10 (2010)
2. Chiara, R.D., Mancuso, A., Mazzeo, D., Scarano, V., Spagnuolo, C.: A framework for distributing agent-based simulations. In: HeteroPar'2011 jointly published with Euro-Par 2009. Springer Verlag (2011)
3. Gorton, I., Haack, J., McGee, D., Cowell, A., Kuchar, O., Thomson, J.: Evaluating agent architectures: Cougaar, aglets and aaa. In: Software Engineering for Multi-Agent Systems II, *LNCS*, vol. 2940, pp. 33–35. Springer Berlin / Heidelberg (2004)
4. Hamada, T., Narumi, T., Yokota, R., Yasuoka, K., Nitadori, K., Taiji, M.: 42 tflops hierarchical n-body simulations on gpus with applications in both astrophysics and turbulence. In: Conf. on High Performance Computing Networking, Storage and Analysis, pp. 62:1–62:12 (2009)
5. Lee, M., hong Jeon, J., Kim, J., Song, J.: Scalable and parallel implementation of a financial application on a gpu: With focus on out-of-core case. In: 2010 IEEE 10th Int. Conf. on Computer and Information Technology, pp. 1323–1327 (2010)
6. Mathieu, P., Brandouy, O.: A generic architecture for realistic simulations of complex financial dynamics. In: 8th International conference on Practical Applications of Agents and Multi-Agents Systems (PAAMS'2010), vol. 70, pp. 185–197. Springer (2010)
7. Mathieu, P., Brandouy, O.: Efficient monitoring of financial orders with agent-based technologies. In: Practical Applications of Agents and Multi-Agents Systems, vol. 88, pp. 277–286. Springer (2011)
8. Nagel, K., Rickert, M., Barrett, C.: Large scale traffic simulations. In: J. Palma, J. Dongarra (eds.) Vector and Parallel Processing VECPAR'96, *LNCS*, vol. 1215, pp. 380–402. Springer Berlin / Heidelberg (1997)
9. Perumalla, K., Aaby, B.: Data parallel execution challenges and runtime performance of agent simulations on gpus. In: Spring simulation multiconference, pp. 116–123. Society for Computer Simulation International (2008)
10. Richmond P. Coakley S., R.D.: Cellular level agent based modelling on the gpu. In: High Performance Computational Systems Biology (2009)
11. Saponaro, P., Taufer, M.: Improving numerical reproducibility and stability in large-scale simulations on gpu. Master's thesis, University of Delaware (2010)
12. Schelling, T.: Models of segregation. American Economic Review **59**(2) (1969)
13. Spurzem, R., Berczik, P., Berentzen, I., Nitadori, K., Hamada, T., Marcus, G., Kugel, A., Männer, R., Fiestas, J., Banerjee, R., Klessen, R.: Astrophysical particle simulations with large custom gpu clusters on three continents. Comput. Sci. **26**, 145–151 (2011)