

Performance Optimization for Multi-Agent Based Simulation in Grid Environments

Dawit Mengistu and Peter Tröger
Blekinge Institute of Technology
{dawit.mengistu, peter.troger} @bth.se

Abstract

Multi-agent based simulation (MABS) is a discrete event simulation technique used to study complex systems with entities having social and autonomous behavior. MABS applications are characterized by unpredictable execution behavior and high communication-to-computation ratio.

In this paper, we propose an adaptation strategy to support efficient execution of large-scale MABS applications on typical Grid infrastructures. To achieve this objective, the behavior of MABS applications and the execution environment is investigated, in order to constantly obtain performance prediction models. These models will then be used to realize dynamic load balancing and resource allocation schemes.

We discuss our basic approach, initial experimental results, the planned future research and an application of our research in the transportation and logistics simulation domain.

1. Background

The Grid provides the much needed computing resources for large-scale scientific problems. One area that can benefit from the Grid is simulation, in particular multi-agent based simulation (MABS).

MABS is a discrete event simulation (DES) technique used to study complex systems and phenomena consisting of entities having social and autonomous behavior. Researchers in the fields of economics, sociology, biology, and ecology use MABS as a replacement and/or complementary to macro-simulation [1].

The agents in a MABS form the entities of the simulated real world phenomena (such as human beings or other objects of interest). An agent is an autonomous computational entity existing in the form of programs, with sensory and perceptual capabilities, and can make decisions and acts on the environment it is embodied in through effectors [2]. Agents

communicate with each other, to influence one another or to pursue collective intentions using standard language constructs.

Developing MABS applications from scratch is a time-consuming task. There exist multi-agent platforms capable of handling the most generic features [3]. They provide a framework with standard templates for modeling generic agent behavior, to manage the launching of the simulation, delivery of messages, scheduling and timing of events, and provide mechanisms for life-cycle management. A number of global communities work in developing standards for multi-agent platforms. The “Foundation for Intelligent Physical Agents” (FIPA) is the prominent forum for agent technology standardization.

In certain social phenomena, partial simulations do not always produce meaningful results. It is thus desired to model and simulate the entire phenomena and implement a large-scale MABS application, which demands huge computational resources. The architectural behavior of MABS poses opportunities and challenges with regards to their implementation. MABS applications are built upon autonomous computational agents that can be conveniently deployed in a distributed execution environment. These agents have a highly unpredictable execution behavior attributed partly to their communication characteristics and inherent constraints of the MABS application domain. Inter-agent communication that takes place between agents is a source of latency and performance degradation. The time management aspect and the need to maintain global synchronization also severely affects application performance.

2. Motivation

The objective of our research is to improve the support for running MABS applications in cluster and Grid environments.

Usually, the agent implementations for the simulated entities are realized as set of threads executing asynchronously. A MABS application is therefore a parallel application that can be executed on

parallel architectures. The complexity of such simulation may be determined by:

- The number of agents / threads running;
- The intensity of inter-agent interaction given by the amount of messaging;
- The granularity and complexity of the agents' task implemented as its computational body

Running a simulation on a MABS platform involves the following steps:

1. Setup and initialization of the MABS platform in a hosting environment;
2. Creating simulation agent implementations with the platform supplied templates;
3. Implementing additional features of the agents and the communication behavior;
4. Launching the simulation application.

Existing multi agent platforms do not have built-in mechanisms to implement these steps in the Grid environment in a seamless manner. Some of these platforms also do not address the problems of large scale MABS. Furthermore, in their standard configuration, none of the platforms are optimized for execution in the Grid environment.

In order to achieve the best possible performance in Grids, applications need to be dynamic and adaptable to the changes in the execution environment. Due to this, specific middleware were introduced in the past for the various application domains, e.g. like the Cactus toolkit [10].

3. Research Plan

Large scale MABS are characterized by their high communication-to-computation ratio. In several MABS applications, there is also a need for maintaining global time synchronization across all execution nodes. Such applications are not generally suited for execution in the Grid environment with its resource heterogeneity, load fluctuations, constraints and policies at execution nodes, etc. However, by making necessary alterations on the run-time behavior of MABS applications, a considerable improvement in performance might be achievable. Possible solutions to this problem include:

- Use of appropriate deployment and load reallocation strategies such as agent migration to reduce cross-node inter-agent communication and to correct load imbalance;
- Use of aggregation techniques for delivery of cross-node inter-agent messages;
- Optimizing the application's threading level;

- Enforcement of an optimal time synchronization method at the global level.

In our research, we therefore study how a MABS platform can be adapted for an improved performance in the Grid environment. We investigate the behavior of the MABS application itself and the execution environment at runtime. The resource requirements of running MABS applications are recorded and analyzed in order to allow a performance prediction based on the analyzed data to devise a better allocation and scheduling strategy and accordingly initiate suspension, migration, and termination of agents.

We shall investigate two types of approaches on how to realize the proposed adaptation components for the MABS application:

- Centralized approach: The relevant performance data is collected from the execution nodes and stored in a central database for analysis purposes. This data includes cross node traffic data, agent task granularity, load information on execution node, etc. This data is then processed centrally to figure out the interaction topology and peer-list of agents to propose reallocation and bundling strategies using heuristics algorithms. The major limitation of this approach is that with growing simulation size and increasing number of Grid resources participating in executing the MABS application, the data collection and analysis tasks may be overwhelming. The resulting computational overhead can even make the viability of the solution for large MABS questionable.
- Self-configuration approach: an optimal (or sub optimal) distribution of agents with minimal cross node communication can be achieved using adaptive algorithms locally. It allows optimal threading level and load distribution at the nodes. The decisions for changes in the application configuration will be made at relevant local levels with minimal central interference. Though this approach may not always yield optimal solutions, it gives faster results with minimal computational and communication overheads. This approach requires implementing the data instrumentation and analysis components in a decentralized manner.

4. Initial Approach and Experiments

In order to study the general applicability of wide-area distributed processing in our problem domain, a generic MABS application was implemented for initial experiments in a Globus-based testbed. We used the WSRF programming model in order to ease up the development. Each Worker Grid service in the simulation hosts an equal number of agents, instantiated as user threads. The simulation is therefore run as a parallel application launched from a master (client) node where the workload is divided into balanced loads and distributed accordingly to the Grid services. At this stage, only simple agents with basic functionalities were used.

The experiment is designed in such a way that during one simulation cycle, each agent sends out a message after a fixed amount of computing. This cycle of computation-communication runs multiple times.

The following parameters are used as control variables in the initial study of the resource allocation problem:

- Number of agents, N
- Outbound communication rate (OB in %), the ratio of the number of messages to other nodes to the total number of messages sent out by an agent
- Granularity (G in ms), the amount of computational work the agent performs before sending a message
- Number of computational nodes (M)

Measurements were taken by varying the above parameters in a controlled environment. From the measured data, performance prediction models were built to compute the execution times T_{st} and T_{Grid} of simulation runs in stand-alone and Grid environments respectively, as functions of the control parameters:

$$T_{st} = T_{setup} + K h(N, G) \quad (1a)$$

$$T_{grid} = T_{setup} + K f(N, M, G, OB) \quad (1b)$$

h and f are functions that estimate the effects of the parameters in stand-alone (N, G) and Grid (N, M, G, OB) environments respectively. In a stand-alone simulation, $M=1$ and can be omitted; since all agents reside on one machine, all communications are inbound and OB is not a relevant variable.

T_{setup} is the time required to fully deploy and launch the simulation. This time depends on the number of agents involved in the simulation and should be studied separately, since it may even be longer than the useful simulation run-time.

K is a proportionality factor representing the length of the simulation if the execution events are assumed to be independent. Obviously, K has no effect on the set up time.

Since the simulation size is usually given in terms of the population size of the simulated entities, which is the number of agents, it is also possible to write the execution time as a function of N (the Grid version is shown here, the stand-alone has also a similar form):

$$T_{grid}(N) = T_{setup} + a_m N^m + a_{m-1} N^{m-1} + \dots + a_1 N + a_0 \quad (2)$$

Where a_0, a_1, \dots, a_m are coefficients dependent on the Grid simulation environment and the architecture of the application. They are given by:

$$a_i = f_i(M, G, OB) \quad (3)$$

f_i approximates the combined effect of the three parameters.

To determine the coefficients, data is collected from repeated runs of the simulation for different values of the input parameters. The measurement data were analyzed using the MATLAB software for the initial experiment, in order to build the required performance models. In future, this computation will be part of the performance modeler of the proposed solution.

Beside the computational load, there is also a relationship of simulation performance to the amount of communication between agents. In parallel applications, one technique to reduce the cost of communications is to minimize outbound messaging. MABS, however, have inherent limitations to employ this technique directly. It is assumed that the list of peers an agent communicates with does not change in time. However, any two peer agents may not necessarily have identical sets of peers between themselves. Because of this, it is not possible to eliminate outbound communication completely.

Our proposed solution tackles this issue by rearranging highly interacting agents on the same node. After an initial random placement, messaging patterns are observed through the monitoring services. Based on this data, a heuristic clustering algorithm identifies groups of peer agents that can be co-allocated in the further execution:

1. Starting with any arbitrary Grid node q , find the number of agents n_q , running on it.
2. For each agent a_i running on node q , create a set S_i of peer agents it communicates with
3. From S_i , identify agents $(a_0, a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_k)$ executed on the same node as a_i and repeat step 1 for each, create $S_0, S_1, \dots, S_{i-1}, S_{i+1}, \dots, S_k$.

4. From the sets $S_0, S_1, \dots, S_{i-1}, S_i, S_{i+1}, \dots, S_k$, find a common set of peer agents, S_p . Agents belonging to S_i and S_p are peers and can be located on the same node. If the number of peers obtained in this way exceeds the capacity of the node n_q , the excess can form another peer set to be located in the remaining nodes in orderly fashion.
5. Repeat this process until all agents are grouped as above.
6. Reallocate the agents in accordance with the new grouping such that as many peers as possible are deployed on the same node.

As a further effort to relieve the network congestion, messages destined to a node are aggregated together for dispatching. Similarly, messages leaving a node are likewise collected together and dispatched. The messaging agent usually delivers or receives several bundles of messages when it takes control of the CPU.

For time-driven MABS, all agents advance execution at a synchronized pace of simulation time steps to imitate real time. Agents that have completed a one time-step task remain in a waiting state until all agents get to the same level.

5. Initial Results

The experiment was run with the following range of input variables:

$$\begin{aligned}
 M &= 1, 2, 3, 4, 5, 6 \\
 OB &= 0, 10, 20, \dots, 90, 100 \\
 G &= 0.2, 1, 2, 10, 20, 100, 200 \text{ ms} \\
 N &= 50, 100, 150, \dots, 3000
 \end{aligned}$$

The resulting performance model in the initial experiment for a granularity $G=10\text{ms}$ is shown in equation 4.

$$\begin{aligned}
 T(N) = & ((-0.0024*M) + (0.0001*OB) + 0.0113) * N^2 + \\
 & ((-0.2748*M) + (0.020 *OB) + 1.1535)*N + \\
 & ((75.2585*M) + (2.7441 * OB) - 113.757) \quad (4)
 \end{aligned}$$

The thread setup time (in seconds) is given by the following model, valid for $100 < N < 1000$:

$$T_{\text{setup}} = (0.000476 * N^2) - (0.03752 * N) + 8.2625 \quad (5)$$

The coefficients of OB are small indicating that the effect of outbound communication is minimal for large N (larger simulations).

The models are validated with measurements as can be seen in the data shown in table 1. The chosen

values of the inputs are selected from the model's valid ranges.

G (ms)	M	OB (%)	Response Time (s)		
			Predict.	Measured	Dev. (%)
2	2	10	616	483	22%
2	4	10	660	751	14%
2	5	10	682	790	16%
5	2	20	373	320	14%
5	4	20	353	358	1%
5	5	20	405	403	0%
10	2	30	330	368	12%
10	4	30	256	245	4%
10	5	30	292	281	4%

Table 1. Performance prediction using N=180

If there is large outbound communication, scalability obviously cannot be achieved, since raising the number of worker nodes only increases the communication cost much more than the saving in computation time.

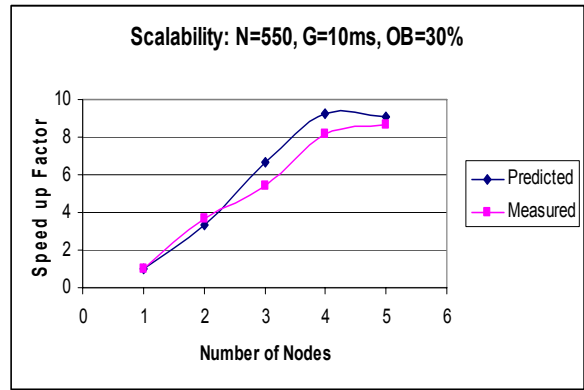


Figure 1: Scalability for increasing node count

If the simulation involves N agents randomly distributed on M machines, the probability that an agent finds its peers on its own node is $1/M$. The number of agents launched per node will be N/M and the rate of outbound communication will then be:

$$OB = (1 - 1/M) * 100\% \quad (6)$$

Using the earlier mentioned algorithm, the OB can be reduced significantly depending on the number of peers an agent has. Table 2 shows the reduction in the number of outbound messages as a result of this intervention. The experiment was run with 500 agents and different inter-agent communication intensities (different number of peers per agent) on 5 participating machines.

Those runs with larger reduction percentage correspond to cases where agents have relatively fewer peers, so that after reallocation, they find most of their peers on the same node and the communication becomes essentially inbound. If, however, agents interact with too many peers, it may only be possible to move a limited number of these peers to the same node and a good portion of the communication remains outbound.

Run No.	No. of OB Messages		Reduction
	Without support	With support	
1	1564	458	70.7%
2	5716	3840	32.8%
3	15825	4721	70.2%
4	28691	13393	53.3%
5	32552	8496	73.9%

Table 2: Comparison of outbound messages with and without agent reallocation support

6. Related Work and Use Case

The Caribbean system [6] is a massively multi-agent platform for use in large-scale navigation system of humans. As most agent simulations, this tool is developed as a standalone threaded application, which does not scale up to a distributed parallel simulation environment. Another example is from Shibuya et al. [7], who propose a framework for multi-agent based computational modeling and quantification of social behavior in mobile systems.

Gunderson and Petersen [8] propose a multi-agent based problem solving approach in mobile environments. It mainly focuses on dynamic planning issues of applications running on mobile devices. The work is at a prototype stage and no empirical validations have been produced.

Poggi et al. [9] discuss the difference between agents as enablers or customers of grid capabilities. Like in our approach, they propose the extension of JADE for direct usage in Grid environments, but mainly for rule-based creation and composition of tasks. There is no specific relation to application scenarios or particular grid environments.

Ludwig et al. [11] propose the usage of agents for implementing SLA negotiation facilities in Service Grid environments. Their work mainly introduces a negotiation protocol based on the FIPA standards.

Newman et al. [12] describe agent-based distribution services for data grids in high-energy physics. Their work is based on the JINI platform,

using a self-organizing scheduling approach. Even though the application domain differs, these scheduling approaches might be interesting for further investigation.

Moreau [13] describes the usage of agent technology for building grid infrastructures. He describes grid environments in terms of a service market place, where agents could act on behalf of both service consumers and resource providers. In contrast to this work, our scenarios focus on agents as grid application.

Transportation Scenario

We plan to use the “Java Agent DEvelopment Framework” (JADE) toolkit as our MABS platform, a widely used multi agent platform based on FIPA specifications. JADE simplifies the implementation of MABS applications through a layer that complies with FIPA agent specifications and services such as white-page services, yellow-page services, message transport and parsing services. The JADE platform can be used across Java-enabled heterogeneous machines. It is based on an own execution container, which provides a complete run-time and communication environment for agent execution.

The MABS application employed as a use case focuses on effects of control policies on freight transport chains. It is known that besides the positive impact freight transport has on the economy, it has also several drawbacks, such as emission of dioxides, traffic congestion, or accidents [4]. The objective is to study the impact of introducing new government control policies on freight transport chains through a micro-level simulation technique. The simulation model can also be used as a decision support system for the various stakeholders of a transport chain, such as customers, buyers, suppliers, production, or transport coordinators. It captures important information such as production capacity, storage, vehicle loading and unloading time, vehicle capacity, speed, environmental performance, and others. The model also incorporates the interaction between all the entities, which are rightly modeled as agents.

An available version of the simulator, known as TAPAS, is implemented using the JADE platform for stand-alone machines [5]. It was observed that for performance reasons, it is not possible to carry out a full-scale simulation. This observation provides the foundation for the overall research topic. Since the existing simulation is based on JADE, we extend it in order to rely on our improved MABS support for agent placement and reallocation decisions.

7. Conclusion and Next Steps

An important issue in parallel simulation is time management, preserving temporal characteristics of the simulated phenomena. The ordering and causality of events has to be correctly captured and enforced during the execution of the simulation to reproduce the behavior of the real world problem and to ensure repeatability of the experiment. Preserving temporal consistency of MABS applications ported to the Grid environment needs to be explored in detail and appropriate measures are required to adapt existing methods for management and synchronization of simulation time.

In the first experiments, we used a synthetic workload with generic features to represent a basic MABS application. This workload differs from an actual MABS workload because its model contains several simplifying assumptions. Prominent features of MABS applications used in social simulation modeling were not considered until now. A further simplification in the workload was the avoidance of any I/O related activities.

Although it was attempted to make the compute resources heterogeneous by exposing them to different load conditions, the initial study would have been more comprehensive in an environment with heterogeneity in hardware and software. So far, the experiment environment was only a local cluster system of homogeneous nodes. To address this problem, we are currently expanding our initial experiment setup. Future experiments will be performed on an already established Globus 4 testbed with around 40 machines over 3 sites in Germany and Sweden.

We will also replace the initial agent simulation by an existing MABS application from practice in the transport and logistics domain.

We believe that the outcome of this research will be useful for development of large-scale MABS applications on the Grid. Although the work is domain specific, some of the important findings can also be used in other types of parallel distribution simulation applications.

8. References

[1] Sansores, C. and Pavon, J. "A framework for agent based social simulation". *The Second European Workshop on Multi-Agent Systems*. Barcelona, Spain. 2006

- [2] Conte, R., Gilbert, N., and Sichman, S. "MAS and social simulation: A suitable commitment." *LNCS V1532, Springer Berlin/Heidelberg*. 1998. pp. 1-9
- [3] Kota, R., Bansal, V., and Karlapalem K. "System issues in crowd simulation using massively multi-agent systems" *Proceedings of International Student Workshop on Agents*. Kyoto, Japan. 2006.
- [4] Ramstedt, L. "Analyzing the effects of government control policies in transport chains using micro-level simulation", *Licentiate Thesis, Blekinge Institute of Technology*. Ronneby, Sweden. 2005.
- [5] Holmgren, J. et. al. "An Agent Based Simulator for Production and Transportation of Products", *11th World Conference on Transport Research*. Berkeley, USA. 2007
- [6] Nakajima, Y. et.al. "Caribbean/Q: A Massively Multi-Agent Platform with Scenario Description Language". *Second International Conference on Semantics, Knowledge, and Grid*. 2006.
- [7] Shibuya, K. "A framework of multi-agent-based modeling, simulation, and computational assistance in an ubiquitous environment" *SIMULATION, Vol. 80*. No.7-8. The Society for Modeling and Simulation International 2004, pp 367-80
- [8] Gunderson, O.E., and Petersen A.K. "Multiagent Based Problem-solving in a Mobile Environment" *Norsk Informatikkonferanse*. Bergen, Norway. 2005.
- [9] Poggi, A.; Tomaiuolo, M.; Turci, P., "Extending JADE for agent grid applications," *WET ICE*. 2004. pp. 352-357.
- [10] Allen, G.; Bengler, W.; Dramlitsch, T.; Goodale, T.; Hege, H.-C.; Lanfermann, G.; Merzky, A.; Radke, T.; Seidel, E.; Shalf, J.: "Cactus Tools for Grid Applications". *Cluster Computing 4 (2001), Nr. 3*. pp. 179-188
- [11] Ludwig A.; Braun P.; Kowalczyk R.; Franczyk B. (2005). "A Framework for Automated Negotiation of Service Level Agreements in Service Grids". *Proc. 3rd International Conference on Business Process Management (BPM 2004)*. Nancy, France. September 5th, 2005.
- [12] Newman H.; Legrand I.; Bunn J.; "A Distributed Agent-based Architecture for Dynamic Services". *Computing in High Energy and Nuclear Physics*. Beijing, China. 2001.
- [13] Moreau, L.; "Agents for the Grid: A Comparison with Web Services" *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*. 2002. p. 220