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

SYNCHRONIZATION METHODS FOR DISTRIBUTED AGENT BASED MODELS[*]

Christine Harvey
James E. Gentile, Ph.D.

The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102, USA

ABSTRACT

Distributed computing facilitates very large scale agent-based models. However, reliable and efficient communication strategies are needed to synchronize agent states across multiple processors. Traditional management methods are conservative in nature and perform a complete synchronization of all agent state information at every time step of the simulation. An alternative approach to traditional methods is proposed which uses an event-driven technique to synchronize agents. This procedure only synchronizes changes to pertinent information in the model at each time step. This technique requires less information to be broadcast which reduces the run time of the simulation while maintaining consistency in the model.

1 INTRODUCTION

Distributed computing addresses the traditional scaling limitations of Agent Based Models (ABMs) and allows for the development of massive-scale models. Synchronization of agent states between multiple processors is complex and needs to be reliable and efficient. The protocol used to synchronize agent states across processors has a significant impact on the efficiency of the tool. Repast HPC is one of the currently available tools for distributed ABMs (Collier and North 2013). This framework approaches the problem by performing a complete synchronization of all entities of interest at every time step.

This poster reviews an alternative approach to entity synchronization, a design which manages persistent, pertinent information. This protocol performs an initial synchronization between all entities with relationships to other agents and then only performs updates and synchronization following changes to relevant information. The pertinent data synchronization technique is an event-driven method to manage the communication and synchronization between the processors. The conservative and the event-driven approaches are both described and analyzed in the following sections.

2 METHODS

2.1 AGENT REQUESTS

The conservative approach to the problem performs a consistent synchronization of pertinent information at every time step of the simulation. Each processor cycles through their agents and compiles a list of non-local agents from which information is needed. The root processor aggregates this information and broadcasts the requests to all processors which return the requested state information to the root processor. Once the root processor distributes the information gathered, each processor creates local copies of their non-local agents of interest. These temporary copies only contain necessary state information on the requested entity. Therefore, each processor has information regarding the current state of their own agents as well as state information on all agents of interest. This entire process is repeated at the beginning of every time step.

[*]Approved for Public Release; Distribution Unlimited. Case Number 14-2788. ©2014 The MITRE Corporation. ALL RIGHTS RESERVED.

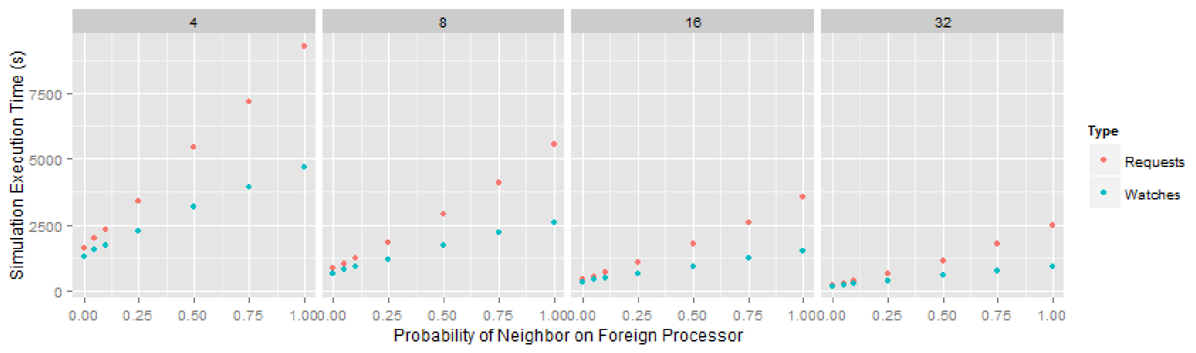

Figure 1: Timing Analysis of Requests and Watches

2.2 AGENT WATCHES

The alternate approach recognizes that not all pertinent information changes at every time step in the simulation. Complete synchronization can be achieved by only tracking and reporting the changes to relevant information. Agent watching only synchronizes information when an entity has experienced a change in state. After the creation of a relationship between two entities, if the agent of interest is not local, it is added to a global list of watched entities. During the synchronization process, the states of newly watched agents are communicated to any processor which has an interest in the agent. Basic, persistent local copies of these watched agents are made on the processors that require the state information of the non-local agent. The processor uses these local copies as a source of information for the updates on their local agents. When watched agents experience a change in state the processor sends the updated state information to the root node to be broadcast to all processors. Then, at the start of the next time step, remote copies of agents are updated to the correct, current state. With this method, fewer and smaller messages are sent at each time step than with the previous technique.

3 RESULTS

A complete timing analysis was performed on both techniques using Python's cProfile tool. The sample model used in this experiment was a basic rumor model, where each agent has exactly two assigned neighbors which were distributed according to the probability of having a neighbor on a foreign processor. Trials were run with 80 million agents total split across 4, 8, 16 and 32 processors. Three trials were completed at each design point and the average of these three trials can be seen in the charts above. The displayed timing is the actual simulation running time, which does not include the overhead to initialize the model.

4 CONCLUSIONS

The completed timing analysis shows that the watches technique is significantly faster than the requests method. The run times for the method are quicker overall than the requests technique. In addition to the overall speed increase, the watches technique also scales up in both number of processors and number of agents per processor more efficiently than the requests method. The use of the agent watches synchronization technique enhances the usability, scalability, and speed for anyone executing massive-scale agent based simulations.

REFERENCES

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