TIME WINDOWS IN MULTI-AGENT DISTRIBUTED SIMULATION

Michael Lees[1], Brian Logan[1], Georgios Theodoropoulos[2]

[1]School of Computer Science and IT, University of Nottingham, Nottingham NG8 1BB, UK
{mhl,bsl}@cs.nott.ac.uk
[2]School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK
gtk@cs.bham.ac.uk

ABSTRACT

In this paper we describe a method for optimistic parallel discrete event simulation for multi-agent systems. We present an adaptive metric that calculates the appropriate degree of optimism for an agent process based on the reads and writes made to the shared state by that agent. We conclude by showing how results from our investigations into the application of moving time windows in agent simulation affects future development of the adaptive metric.

KEYWORDS

Distributed Simulation, Agent-based Systems, Synchronisation

1 INTRODUCTION

An autonomous *agent* can be viewed as a system which is situated in an environment and which interacts with its environment through sensing and actions. The agent's actions are performed in pursuit of its own agenda so as to affect what it senses in the future. The *environment* of an agent is that part of the world or computational system 'inhabited' by the agent. The environment may contain other agents whose environments are disjoint with or only partially overlap with the environment of a given agent. In addition to being an active research area, agent-based systems have been applied in a wide range of areas including telecommunications, business process modelling, computer games, control of mobile robots and military simulations.

Simulation is a key tool in the development of agent-based systems. They allow the agent designer to learn more about the behaviour of a system or to investigate the implications of alternative agent architectures, and the agent researcher to probe the relationships between agent architectures, environments and behaviour. Agent-based simulations typically take the form of a *test-bed* which consist of one or more simulated agent(s), their environment and a standardised task performed by the agent(s) in the environment. Such test-beds are often highly configurable, allowing the agent designer to test and tune the parameters of the agent over a large number of executions.

In [5] a parallel discrete event simulation framework for multi-agent systems is presented. Identifying the efficient distribution of the agents' environment (namely, the *shared state*) as a key problem in the simulation of agent-based systems, the framework models agents as Logical Processes and the environment as a tree-shaped network of processes (referred to as *Communication Logical Processes* or *CLPs*) which is dynamically reconfigured to reflect the changing interaction patterns between the agents and their environment in the simulation.

The central concept of the framework is the notion of the sphere of influence. The *sphere of influence* of an event is defined as the set of state variables read or written as a consequence of the event and depends on the type of event (e.g., sensor events or motion events), the state of the agent or environment logical process which generated the event and the state of the environment. The sphere of influence of an agent process p_i over the time interval $[t_1, t_2]$, $s(p_i)$, is defined as the union of the spheres of influence of the events generated by the process over the interval. In [5], the spheres of influence of the LPs are used to derive an idealised decomposition of the shared state into CLPs to facilitate dynamic load balancing and interest management, and in more recent work [4], we described how spheres of influence can be exploited in the design of an adaptive synchronisation mechanism.

The remainder of this paper is organised as follows: Section two gives a brief introduction to simulation and the problem of synchronisation. In section three we discuss the role of the shared state in agent simulation and show how aspects of the shared state and agent simulation in general can be exploited in optimistic synchronisation. Section three goes on to describe a metric based on spheres of influence which is used to define an appropriate degree of optimism for a process in a parallel simulation. In section four we present results which indicate a special relationship between agent based simulation and constrained optimism synchronisation schemes based on moving time window[9]. We conclude with a discussion of these results and indicate how they will affect the direction of our future research.

2 SYNCHRONISATION

An event-based simulation models a physical (real) system in terms of *events* and *states*. The execution of the simulation consists of processing events to modify the state. Each simulation also has an abstraction of the time at which events occur, known as *simulation time* or *virtual time*. Each event occurs at a particular instant in simulation time, known as the *timestamp* of the event. A single processor (sequential) discrete event based simulation therefore consists of the following,

- a set of *state variables* which collectively describe the state of the system;
- and *event list* containing those events yet to be processed; and
- a *global clock* denoting the current simulation time.

A sequential discrete event simulation can easily ensure that events are processed in time stamp order as it processes the event with the smallest time stamp in the event list. A parallel discrete event simulation (PDES) is composed of multiple sequential simulations or *Logical Processes* (LPs). Distributing the simulation over multiple processes requires multiple event lists, one for each LP. A consequence of this is that ensuring the events are processed in time stamp order is less straightforward. In asynchronous, event-driven distributed simulation, each LP maintains its own local clock with the current value of the simulated time, termed its *Local Virtual Time* (LVT). This value represents the process's local view of the global simulated time and denotes how far in simulated time the process has progressed. If each LP processes its event list independently and advances its LVT at its own rate, events may be processed out of time stamp order. Therefore a mechanism is required to ensure the parallel simulation faithfully implements the causal dependencies and partial ordering of events dictated by the causality principle in the modelled system.

It has been shown [2] that a distributed system consisting of asynchronous concurrent processes will not violate the causality principle if each process consumes and processes event messages in non-decreasing timestamp order (the *local causality constraint* (LCC)). There are two main approaches to ensuring that the local causality constraint is not violated: *conservative* and *optimistic*. Conservative mechanisms strictly avoid violation of the LCC while optimistic mechanisms provide a means to undo computation which results in a violation. In more recent years, hybrid mechanisms which take aspects of both have been developed, e.g., optimistic schemes with constrained optimism such as moving time window [8]. Other optimistic schemes have been developed so that the degree of optimism (how constrained they are) can be decided at run time. These are known as adaptive synchronisation mechanisms (e.g., [1]).

3 THE SHARED STATE

In the agent simulation framework presented in [4], the shared state of an agent simulation consists of a number of *state variables* whose values model the state of the agents' environment. Agents interact with their environment by reading and writing these state variables. We use the term *access* to refer to either read or a write. Table 1 shows part of the shared state of a typical agent simulation,.

Variable	Access patterns
v_1	$(A_1, R, t = 2), (A_2, R, t = 4), (A_1, W, t = 3) \ldots$
v_2	$(A_2, W, t = 2), (A_2, R, t = 3), (A_1, W, t = 5), (A_2, R, t = 4) \ldots$
.	
.	
.	
v_n	$\ldots$

Table 1. A global view of the shared state

In the example, two agents, A_1 and A_2, each access two state variables in the shared state, v_1 and v_2. Each access is represented by a triple where A_n indicates the agent performing the access, *R/W* indicates whether the access is a read or a write, and *t* is the timestamp of the access. The left to right ordering represents the order in which events are processed in real time, i.e., $(A_1, R, t = 2)$ was processed before $(A_2, R, t = 4)$.

The table shows two events being processed out of time stamp order. The write on state variable v_1 at virtual time 3 invalidated the read processed before it which was scheduled for virtual time 4. In this situation a rollback is necessary on agent A_2, it will then re-send the read and obtain the correct value for the state variable v_1. An important consequence for synchronisation is that rollback frequency is likely to increase when many different LPs read and write the same state variables.

However not all late messages result in a rollback. The late read on state variable v_2 need not result in a rollback as the late read can obtain the correct value for by checking the history of the variable which is stored in case of rollback. In this case the read event will return the value written at virtual time 2, not virtual time 5. Our framework exploits this property of read/write events and relaxes the causality constraint allowing certain events to be processed out of time stamp order [3]. This scheme is similar to the query event tagging proposed in [9] and should have similar advantages in reducing the frequency and depth of rollback and the state saving overhead. One important distinction is that we allow a write w_t with time stamp t to be processed after a write w_v, where $t < v$, if no read exists r_u such that $t < u < v$. With this relaxation it is only late writes which invalidate previous reads that cause a rollback.

To make this intuition precise, we now extend and clarify the definition of sphere of influence from [5] by splitting the sphere of influence into two distinct sets:

1. the *sphere of influence of writes* of a process p_i, $s_W(p_i)$, which contains the set of variables written to by p_i over the time period $[t_1, t_2]$; and

2. the *sphere of influence of reads* of a process p_i , $s_R(p_i)$,which contains the set of variables read by p_i over the time period $[t_1, t_2]$.

Considering the example given above we can now say:

1. any variable which appears only in $s_W(p_i)$ for all processes p_i over $[t_1, t_2]$ (i.e., no process reads the variable) is not important in terms of rollback;

2. any variable which appears only in $s_R(p_i)$ for all processes p_i over $[t_1, t_2]$ (i.e., no process writes the variable) is not important in terms of rollback either; and

3. a variable which is present in the $s_R(p_i)$ of one process p_i and in $s_W(p_j)$ of another, $i \neq j$, may cause a rollback.

More precisely If the sphere of influence of two processes p_i and p_j overlap then we can say the probability of p_i causing rollback on p_j is affected by the number of *critical accesses* made by p_i and p_j, CA_{ij}, defined as:

$$CA_{ij} = |s_R(p_i) \cap s_W(p_j)| + |s_W(p_i) \cap s_R(p_j)| \tag{1}$$

We can then say for any process p_i, the likelihood of rollback is dependent on the critical accesses made by p_i, CA_i, and each of the other $n-1$ processes

$$CA_i = \sum_{j=1}^{n-1} (CA_{ij}) j \neq i \tag{2}$$

Using critical accesses it is now possible to propose the form of an equation which can be used to determine the degree of optimism appropriate for a process p_i as:

$$O(p_i) = \frac{a}{k_1 CA_i + \sum_{j=1}^{n-1} (k_2 CA_{ij} \times k_3 \Delta LVT_{ij})} \tag{3}$$

where the total number of agents in the simulation is n and a, k_1, k_2, k_3 are appropriate constants. The degree of optimism given by the equation may then be used in some form of throttling mechanism such as moving time window [9].

4 RESULTS

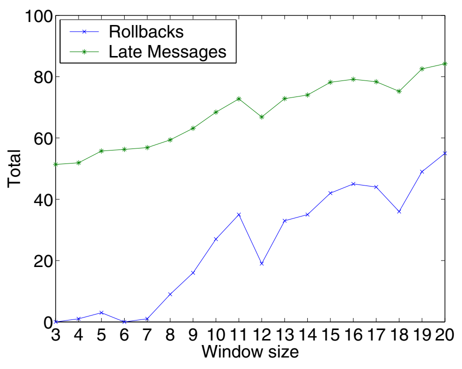
To test this hypothesis, we performed a number of experiments using ASSK, a parallel simulation kernel we have developed to investigate synchronisation mechanism for agent simulation. ASSK is a library of C++ classes which use MPI to aid communication over a network of machines. ASSK does not interface directly with any agent toolkit, rather it uses traces of simulation events produced by a toolkit. An ASSK simulation consists of one or more agent LPs and a single shared state LP. Each agent LP processes a trace from the agent toolkit and sends events from the trace via MPI to the shared state LP. The shared state LP is responsible for maintaining the state variables in the simulation. Upon receiving an event, the shared state LP reads or writes the relevant state variable and generates any necessary responses (ie., rollback). ASSK implements a static window mechanism and the experiments reported here show how window size influences the number of late messages and the number of rollbacks in the simulation.

For our experiments, we used traces from SIM_BOIDS, an implementations of boids [6] developed using the SIM_AGENT toolkit. A boids simulation consists of a number of agents which navigate using a set of simple behavioural rules which result in a bird like flocking behaviour. The shared state of a boids simulation is small; it consists only of the agent's position in 2D space. The experiments used traces from two SIM_BOIDS agents which sense one another (access the other agents position variables) for 20 simulation cycles. Each trace contains 120 events (6 events per cycle). Runs were performed on a homogeneous mini-cluster at Nottingham University, consisting of four Pentium 2.4GHz redhat 7.2 machines with 1GB or RAM connected via gigabit Ethernet.

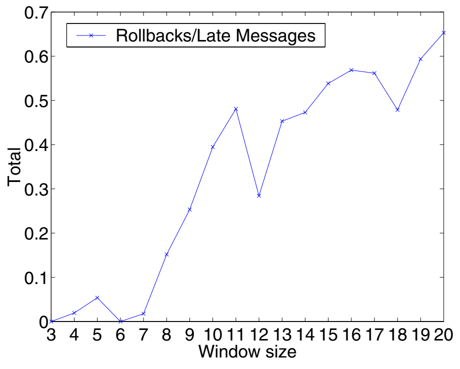
To illustrate the relationship between window size and the number of rollbacks, experiments were performed using twenty different static window sizes within the ASSK simulation. To simulate differing processor loads, each event was delayed by between 100 and 200 milliseconds.[1] The results shown represent the average of fifty runs at each window size. Figure 1(a) shows how the number of late

[1]These values were obtained experimentally, and are chosen to be large enough to dominate the noise within the system due to thread context switching and MPI interaction.

messages and the number of rollbacks in the system is affected by window size. Late messages are defined as late read and writes received by the shared state LP. Rollbacks are all the rollback messages received by all the LPs plus the number of late writes received by the shared state LP which cause rollback.



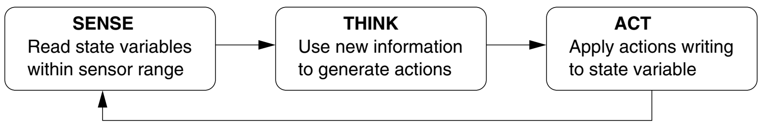
(a) Rollbacks and late messages with different window sizes



(b) Proportion of late messages that are rollbacks.

Figure 1. How window size relates to the number of rollbacks

As can be seen, there is a clear increase in the average number of late messages as the window size increases, with around 50 late messages with a window size of 3, increasing to around 85 late messages at a window size of 20. The same graph shows how rollbacks increase as the window size and hence the degree of optimism of the simulation increases. This is as one would expect: a larger window (higher degree of optimism) increases the likelihood of late events and hence rollback. The results also show quite distinct drops in the number of rollbacks at window sizes of 6, 12 and 18. We believe this is an artifact of the SIM_AGENT traces used in these experiments. Like many other toolkits [7, 10], SIM_AGENT agents work on the basis of a sense-think-act cycle (see figure 2)



SENSE
Read state variables within sensor range
THINK
Use new information to generate actions
ACT
Apply actions writing to state variable

Figure 2. The sense-think-act cycle of a typical agent

In the traces used for these experiments, the events occur in cycles. Each cycle consists of four reads made by the boid followed by two writes. Each boid *senses* the environment around it by reading the other boid's x and y position and its own x and y position. The boid then uses this information to *think* about its next x and y position. Finally the boid *acts* and updates its position by writing its own x and y position. The cycle then repeats with the boids sensing the new positions. Because in this experiment each boid is always within the sensor range of the other, all events occur in the same 6 event cycle pattern.

5 DISCUSSION AND FUTURE WORK

In previous work [4] we proposed an equation which could be used to dynamically adapt the window size of an agent LP depending on the number of critical accesses made by the agent. This equation relied upon the assumption that smaller window sizes constrained the system and resulted in fewer rollbacks. We had expected, as seen in previous studies [9], that window size was more or less proportional to the number of rollbacks in the system. Here we have shown that for agent based simulation this is not the case. The cyclic nature of agent simulation combined with the read write optimisation of ASSK creates complex interactions with window size and rollback.

We now believe that equation (3) from [4] should be interpreted as indicating the appropriate level of optimism rather than window size directly. The results shown here are for a particular agent simulation and use small numbers of agents. While we feel the relationship between cycles and window size is valid, with simulations where the cycle size isn't constant the affect will be less pronounced. With agent simulations in which the agents don't all read the same variables at each cycle, e.g., when some parts of the environment are beyond the range of the agent's sensors, the number of events per cycle will vary greatly during the simulation and we therefore feel confident that time windows can still be applied to agent simulations. In the future we plan to carry out further investigations into the relationships between window size, rollback and agent simulation. We plan to generate traces using large number of agents in different SIM_AGENT simulations including SIM_TILEWORLD.

REFERENCES

[1] A. Ferscha. Probabilistic adaptive direct optimism control in time warp. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS '95)*, pages 120–129, 1995.

[2] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, pages 558–564, July 1978.

[3] M. Lees, B. Logan, R. Minson, T. Oguara, and G. Theodoropoulos. Distributed simulation of MAS. In *In Proceedings of the Joint Workshop on Multi-Agent and Multi-Agent-Based Simulation*, pages 21–30, 2004. (to appear).

[4] M. Lees, B. Logan, and G. Theodoropoulos. Adaptive optimistic synchronisation for multi-agent simulation. In D. Al-Dabass, editor, *Proceedings of the 17th European Simulation Multiconference (ESM 2003)*, pages 77–82, Delft, 2003. Society for Modelling and Simulation International and Arbeitsgemeinschaft Simulation, Society for Modelling and Simulation International.

[5] B. Logan and G. Theodoropoulos. The distributed simulation of multi-agent systems. In *Proceedings of the IEEE*, pages 174–185, 2001.

[6] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34. ACM Press, 1987.

[7] P. Riley. MPADES: Middleware for parallel agent discrete event simulation. In G. A. Kaminka, P. U. Lima, and R. Rojas, editors, *RoboCup-2002: The Fifth RoboCup Competitions and Conferences*, number 2752 in Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin, 2003. *RoboCup Engineering Award* (to appear).

[8] L. Sokol and B. Stucky. Mtw: Experimental results for a constrained optimistic scheduling paradigm. In *Proc. SCS Multiconf. Distributed Simulation*, volume 22, pages 169–173, Jan 1990.

[9] L. Sokol, J. Weissman, and P. Mutchler. Mtw: An empirical performance study. In *Proceedings of the 1991 Winter Simulation Conference*, pages 557–563, December 1991.

[10] A. Uhrmacher and K. Gugler. Distributed, parallel simulation of multiple, deliberative agents. In *Proceedings of Parallel and Distributed Simulation Conference (PADS'2000)*, pages 101–110, May 2000.