Parallelism vs Communication Overhead Trade-off in a JADE Multi-Agent Implementation of Cellular Automata

Hooman Amini*
*Department of Computer Science
University of York
ha502@york.ac.uk

Dimitar Kazakov†
†Department of Computer Science
University of York
Kazakov@cs.york.ac.uk

Enda Ridge‡
‡Department of Computer Science
University of York
ERidge@cs.york.ac.uk

Abstract

This paper investigates the implementation of a two-dimensional cellular automaton as a multi-agent system using the JADE agent middleware. The agents are distributed on separate computers in a local network, and their performance is measured to evaluate the net benefits of parallelising the algorithm, once the additional communication cost is taken into account. The results demonstrate that (1) the overall performance is a non-monotonic function of the number of agents employed; (2) a multi-agent platform may perform comparably to a single process implementation even for problems of relatively modest size; finally, (3), the trends observed in those cases indicate that the multi-agent platform may provide the best solution, but additional experiments would be needed for a definitive answer.

1 Introduction

This paper describes and studies the benefits of a multi-agent implementation of two-dimensional cellular automata using the JADE agent platform.

Multi-agent systems (MAS) can be used to decompose an algorithm and parallelise its execution. This implies a number of possible benefits. A multi-agent approach is usually characterised by the nature of information available to agents. Each agent only has access to local information, therefore, in the general case, it has no knowledge of the state of the system as a whole. If the desired system behaviour can be represented at agent level on the basis of this local information alone, then the use of agents is justified. The use of an agent-based platform is therefore linked to the possibility of decomposing the task at hand.

The suitability of an algorithm to be split into separate processes depends on two factors: (1) the amount of communication, and (2) the degree of indepence (or asynchronicity) between individual processes. The less communication is needed, the lower the computational overhead on top of the one required by each process. In addition, the less often one process needs to wait for another, the better.

Cellular automata (CA) are a modelling approach in which a system is represented as a lattice of finite state automata (FSA), where the next state of each FSA is only dependent on its own state and that of its neighbours. Therefore, as local information is sufficient to compute the next state of the system, CA are potentially suitable for a MAS implementation.

The basic idea is to partition the CA lattice and implement each partition as a separate agent. The only communication between these agents is limited to neighbours informing each other about the state of their immediately adjacent border areas (see Figure 1). The amount of information exchanged relative to the partition area depends on the shape of the partitions (e.g., for a square and rectangle with the same surface, the former will have a smaller circumference).

Despite the apparent asynchronicity of the communication involved, which is limited to messages between pairs of processes (neighbours) not using a global clock, a *de facto* global synchronisation among all processes emerges. This is due to the transitive nature of the information needed by each process, since a partition cannot be updated without information from all neighbours.

Here the benefits of employing a certain number of agents are a result of a trade-off. For a given CA lattice, allocating a part of it to a new agent increases the parallelism of the system, as this agent can run

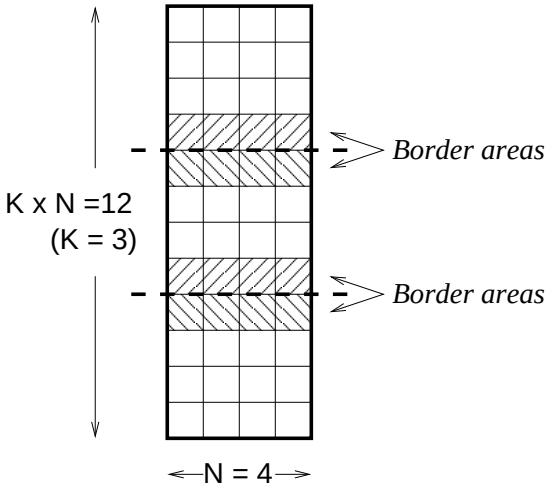

Figure 1: Partitioning of a CA.

on a separate computer/CPU, but creates additional communication overhead as new borders are created.

The aims of this work are to study the pros and cons of a multi-agent implementation of CA, and see if the additional benefits of parallelism can outweigh the additional communication overheads and result in a faster simulation. The factors studied here are (1) the number of agents/partitions employed, and (2) the size and shape of partitions.

2 Background

Cellular automata are dynamic systems consisting of a lattice of cells (in any number of dimensions) each of which has a number of associated discrete states. The state of these cells is updated at discrete time steps, the resultant state dependent upon local state rules Wolfram (1994). The specific set of rules is not important in our experiments. We have chosen a setup based on John Conway's well-known Game of Life, which uses a 2-D orthogonal lattice, and a cell's state rules only refer to the cell's 8 immediate neighbours.

The concept of an *agent* is a general modelling abstraction representing a system through its *behaviour*, mapping inputs from the environment onto system outputs. In this context, a single agent is virtually synonymous with the concept of a system Bertalanfi (1968). Using a multi-agent model of a system introduces the possibility of agent to agent interactions, and highlights issues, such as communication and coordination.

¿From the narrower software engineering perspective, agents are a "*design metaphor for ... applications structured around autonomous, communicative elements*" involving software tools supporting the approach. At the same time, agents are "*a source of techniques and algorithms for dealing with interactions in dynamic and open environments*" Luck et al. (2002).

While there are no general purpose agent orientated languages, certain standards have emerged in the area. FIPA is one such standard for agent management and communication Dale (2005). Here we use JADE, a Java agent development framework that is compliant with the FIPA specifications Bellifemine et al. (2001). Of particular interest here are JADE's asynchronous Peer-to-Peer messaging and its scheduling of multiple agent behaviours.

3 Design

The experimental set-up consists of three types of agent. These agents and their functionality are as follows:

System Pilot Agent: is responsible for overall simulation coordination and lattice partitioning. Coordination includes, signalling the start of generations and distributing new partitions to Partition Simulator Agents.

Partition Simulator Agent: is in charge of each partition's simulation. It communicates with other Partition Simulator agents to negotiate and update border values. Partition Simulators subscribe to the System Pilot agent to receive simulation trigger commands. They report their results to the System Pilot at the end of each generation.

Simulation Inspector Agent: analyses the output and generates diagrams.

An experiment consists of the following five stages: (1) lattice initialisation, (2) partition creation, (3) lattice distribution, (4) partition simulation, and, (5) simulation termination. In stage 1, a lattice of a given size is created and its initial state set. In stage 2, a number of Partition Simulator Agents are created, each running on a separate, identical PCs in a local network. In stage 3, the each of the partition agents is initialised with the corresponding part of the lattice. Stage 4 is the central part of the simulation, in which each lattice partition is simulated on a separate agent. In this phase, all simulation agent interactions are pairwise (peer to peer), and there is no synchronisation (or global clock) provided by the Pilot Agent.

It is only at the end of the whole run that the partitions are assembled again to reconstruct the state of the complete lattice.

In all experiments, the lattice is of width N (where $N = 50$ or 100) and height $K \times N, K \in [1, 30]$. The lattice is divided into identical partitions as shown in Figure 1. The figure illustrates the case when K is the same as the number of partitions (the borders of which are marked with a dashed line), but this is generally not the case. This study describes experiments with 1, 3 and 5 partitions for K taking values in the above range. The dependent variable studied is the overall simulation time. All experiments were averaged over 3 runs for each set of parameters, and the CA was made to go through 10 consecutive states in all cases.

4 Results and Evaluation

The first set of experiments study the simulation time as a function of K for 2 different values of N (Figure 2). Despite the fact that the lattice surface (number of cells) is linear w.r.t. K, the simulation time trend is non-linear, and increases more rapidly, which is clear in the case of $N = 100$.

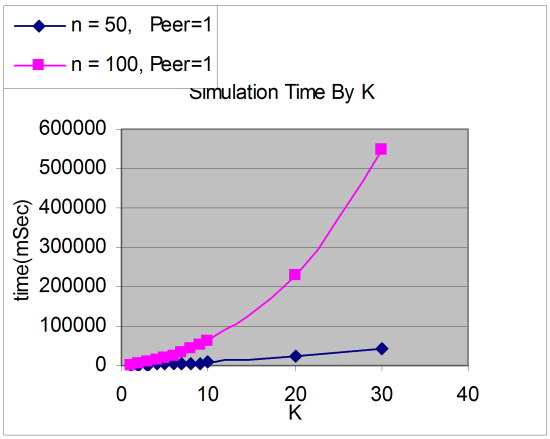

Figure 2: Simulation time as a function of lattice size.

Having established this trend, the next set of experiments studied the main research question addressed here, namely the net effect of the number of agents (partitions) on the overall simulation time (see Figure 3). The observed trends are revealing: simulating the whole CA as a single process, on the same PC, is fastest. The slowest runs are in the case of 3 agents: it appears that the benefits of decomposing and parallelising the task are outweighed by the additional overhead that communication between agents

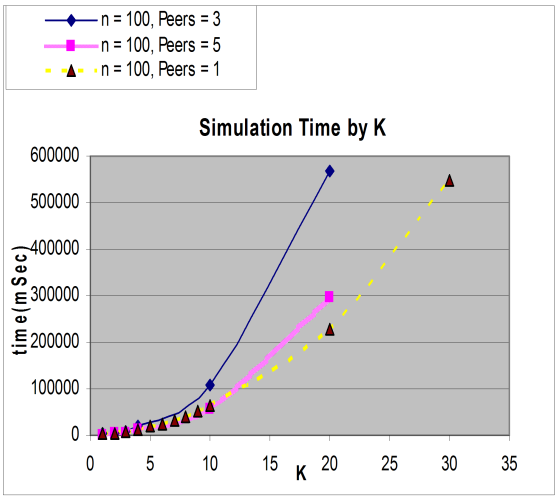

Figure 3: Agent simulation time by K for 1, 3 and 5 agents (peers).

adds. However, this trend is clearly reversed when the number of agents is increased to 5. One is tempted to predict that a MAS CA with an even larger number of agents would outperform the single agent, but no more data is available at the moment.

The simulation times in general appear quite high, which was confirmed when we compared them to a simple object-orientated, non-agent, single thread Java implementation of the CA (data not reported here). Another interesting observation was that the time to create and initialise partitions was considerable (Figure 4), and in fact, comparable or longer than the simulation (which only run the CA for 10 cycles).

While considerable, this overhead is constant, as it does not depend on the number of CA cycles, and therefore, it does not influence the time complexity order of growth.

To have a rough comparison between agent-based and non-agent-based implementations of our cellular automaton, we ran the same simulation on a non-JADE non-MAS (single machine, single thread) platform (Figure 5). The non-JADE framework exploits the same model structure as the other one. The representation of the lattice cells is copied and the execution routine mirrored. Obviously, the issue of segmentation, and related issues of communication and synchronisation, are not present in the non-JADE algorithm. Whilst performance is significantly higher in the non-MAS scenario for CA size up to the hardware limitation of a single machine, extendibility and reusability issues are still more satisfactory on the MAS platform.

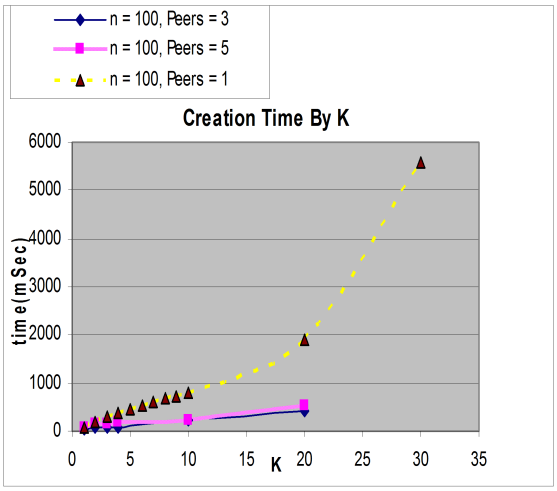

Figure 4: Creation time by K for different number of peers.

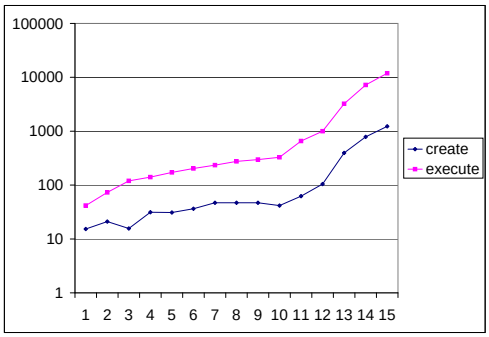

Figure 5: Creation and simulation times (log scale, in *ms*) for different values of K (X axis).

5 Conclusions

The conclusion to be drawn from the results is that the simulation time is a non-monotonic function of the number of agents employed. The observed trend suggests that a MAS CA gives a performance comparable to that of a single process despite the added communication overhead, even for relatively small lattices that do not impose memory problems. Furthermore, the trend observed clearly suggest that a larger number of agents may be able to outperform a single process/agent. It is also clear that using a MAS implementation and/or standard middleware (JADE) may promise scalability, but also comes at a price, as suggested by the creation time data and comparison with a non-agent implementation.

References

Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE: a FIPA2000 compliant agent development environment. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 216–217. ACM Press, New York, NY, USA, 2001.

L. von Bertalanfi. *General System Theory: Foundations, Development, Applications*. George Brazier Inc., NY, 1968.

Jonathan Dale. The Foundation for Intelligent Physical Agents, 2005. URL `www.fipa.org/`.

M. Luck, P. McBurney, O. Shehory, and S. Willmott. Agent technology roadmap, 2002.

S. Wolfram. *Cellular Automata and Complexity*. Perseus Books Group, 1994.