Performance Analysis of Shared Data Access Algorithms for Distributed Simulation of Multi-Agent Systems

Roland Ewald[1,3], Dan Chen[1], Georgios K. Theodoropoulos[1], Michael Lees[2],
Brian Logan[2], Ton Oguara[1], Adelinde M. Uhrmacher[3]

[1] School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK
[2] School of Computer Science and Information Technology,
University of Nottingham, Nottingham NG8 1BB, UK
[3] Institut für Informatik, Universität Rostock, 18059 Rostock, Germany

Abstract

Distributed simulation is an important instrument for studying multi-agent systems (MAS). Such large scale MAS simulations often have a large shared state space. Moreover, the shared state and the access pattern of agent simulations both are highly dynamic and unpredictable. Optimising access to the shared data is crucial for achieving efficient simulation executions. PDES-MAS is a framework for distributed simulation of MAS, which uses a hierarchical infrastructure to manage the shared data. In order to enable agent simulations to access distributed shared data effectively and efficiently, this paper proposes two routing algorithms, namely the address-based routing and the range-based routing. The paper introduces a meta-simulation approach to evaluate the characteristics of both solutions and provides a quantitative comparative analysis of the proposed algorithms.

1. Introduction

The Logical Process Paradigm seeks to divide the simulation model into a network of concurrent *Logical Processes* (*LPs*), each of which models some object(s) or process(es) in the simulated system. Each LP maintains and processes a portion of the state space of the system and state changes are modelled as timestamped events in the simulation. In conventional distributed simulations, the shared state is typically small and the processes interact with each other in a small number of well defined ways. The topology of the simulation is determined by the topology of the simulated system and its decomposition into processes, and is largely static.

However, in the case of systems, which operate in a complex environment and interact with it in complex and dynamic patterns (such as multi-agent systems, battlefield simulations, ecological systems, games etc), it is often difficult to determine an appropriate simulation topology a priori. In such systems there is a very large set of shared state variables which could, in principle, be accessed or updated by the processes in the model [20]. Encapsulating the shared state in a single process (e.g. via some centralised scheme) introduces a bottleneck, while distributing it all across the LPs (decentralised, event driven scheme) will typically result in frequent all-to-all communication and broadcasting. This problem has received considerable attention in the context of Interest and Data Distribution Management for large scale distributed simulations [5, 13].

In [6] we have proposed an approach to manage the shared data in distributed simulations of multi-agent systems (MAS). Shared data management in distributed simulations needs to address two problems: data distribution and data accessing. In [15] we have addressed the first problem and have described data distribution algorithms for the PDES-MAS framework[1]. In this paper we address the second problem of data access.

Data accessing targets both individual data items (ID query) and selected data items overlapping given query windows (Range query). Although in the area of distributed data bases, range query strategies are increasingly used, e.g. [1, 14], this problem has received little attention in distributed simulations [4]. The issue becomes much more complicated when the value and the physical distribution of data items both are dynamic. Further problems arise when positions of query sources are dynamic too, as in the context of location-dependent information services (LDIS) [7]. In turn, LDIS do not have to cope with highly dynamic data distributions, since data can be distributed according

[1] Synchronisation issues have been discussed in [8, 11].

to physical scope.

In this paper, we propose two candidate algorithms for data accessing in the context of the PDES-MAS framework, namely the address-based and the range-based routing.

The rest of this paper is organized as follows: Section 2 provides a brief overview of the PDES-MAS system and states the problem. The candidate routing solutions are briefed in section 3. Section 4 introduce the simulation model used for the performance evaluation of the algorithms. The experimental results and their analysis are presented in section 5 while section 6 epitomizes the conclusions and suggests future work.

2. Background and Problem Statement

PDES-MAS is a framework for the distributed simulation of agent-based systems (figure 1). Each agent in the framework is modelled as an Agent Logical Process (ALP). An ALP has both private, which is maintained within the ALP, and shared state which is accessible to other ALPs.

The shared state is modelled as a set of Shared State Variables (SSVs), each of which is a tuple of the form $< SSVID, attributetype, value, timestamp >$, where type represents an attribute of the object's class.

ALPs interact with the shared state and other ALPs through read and write (update value) operations on SSVs. This operations can have the form of an ID query or Range query. To generate an ID query, an ALP needs to specify the SSV ID, and its new value in the case of update. A Range query requests a set of SSVs of a given type whose values match a designated range. Range queries are initiated to meet the need of an ALP to explore some portion of its environment, which it may not be aware of beforehand.

The SSVs are managed by a tree-shaped hierarchical structure of Communication Logical Processes (CLPs), which is dynamically reconfigured to reflect the shared data access patterns in the simulation [15]. In this process, SSVs are migrated towards the frequently accessing ALPs according to cost measures, thus the scalability of the framework is ensured. A CLP interacts with other LPs via *ports*. The queries from an ALP are modelled as timestamped messages, for which each CLP acts as a router responsible for forwarding them to the destination CLP(s). The ports are designed to maintain the distribution of the values of SSVs in the value space classified by the types of SSVs.

3. Two Routing Algorithms

In this section we propose two different algorithms to route ID and Range queries through the tree of CLPs. The two proposed algorithms dynamically adapt to different properties of the shared state and the system.

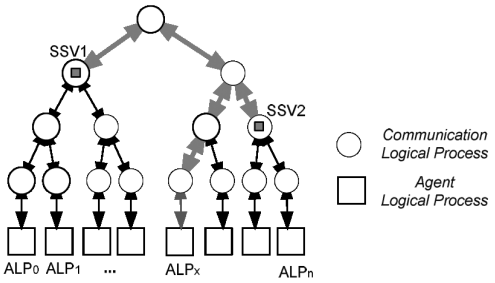

Figure 1. Illustration of PDES-MAS Framework

3.1. Address-Based Routing

The address-based routing scheme searches for SSVs according to their addresses, namely their exact location (host CLP) in the tree. Figure 2 illustrates an address-based routing scheme, which binds the ID of an SSV to its address. Each server CLP maintains a routing table which contains the addresses of SSVs. Furthermore, each CLP stores information about the values of SSVs that are hosted by its immediate neighbours. This information is obtained and refreshed when updates on those SSVs occur. To store the information efficiently, the overall value range of each SSV type is divided into a number of segments. Hence, only one bit per segment is needed to store information about the existence of SSVs with values covered by this segment. For example if a CLP contains a set of SSVs with values listed as: {20, 53, 56, 70, 80, 190, 310, 370}. Instead of using a simple range description like [Min(20), Max(370)], we segment the value space, such as Seg1: [0, 100], Seg2: [100, 200], Seg3: [200, 300], Seg4: [300, 400], Seg5: [400, 500], ... The approach logs the number of SSVs whose values fall onto each segment.

When an ALP issues a Range query (see figure 2), its server CLP propagates the request to all CLPs which host SSVs of that SSV type (in this example, CLP_1 and CLP_2). If the values of its SSVs are not within the segments covered by the Range query, the neighbours of a CLP can stop the query (unless there is another CLP that needs to be reached).

The algorithm for an ID query is straightforward. When an ALP issues an ID query, the server CLP determines the location of the SSV from its routing table and forwards the query to the corresponding host CLP.

3.2. Range-Based Routing

This approach uses information about the values of SSVs to locate them in the tree, in a fashion similar to associative memory. The algorithm matches the query window with

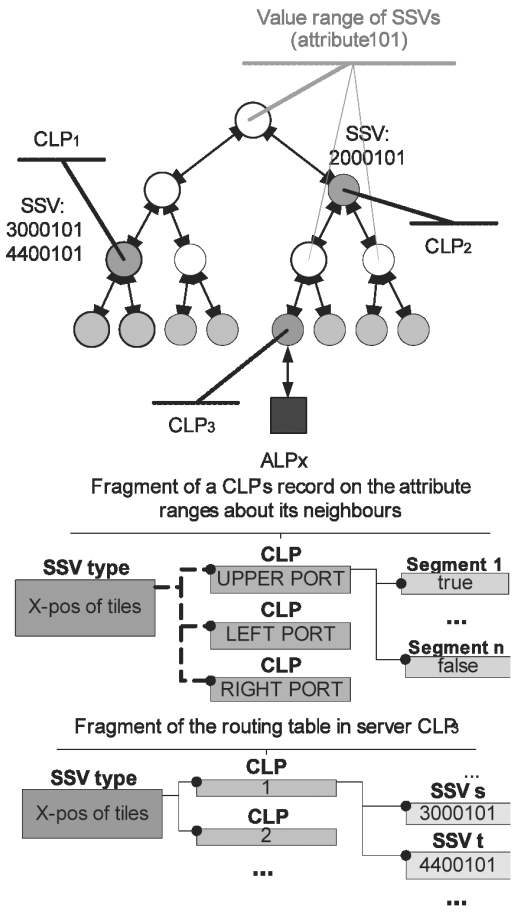

Figure 2. Address-based Routing

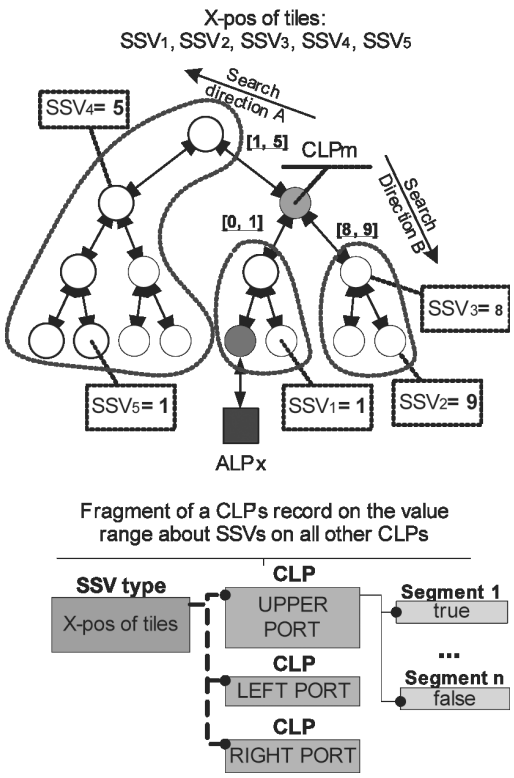

Figure 3. Range-based Routing

the value ranges along the searching paths to gradually approach the potential targets. Searching will stop at the directions where the query window and value ranges do not overlap. In the example shown in figure 3, CLP_m keeps a record of the SSV sets (denoted by the encirclings) behind its three ports.

When an ALP issues a Range query, this procedure starts from its server CLP. Suppose that a Range query for SSV type "X-pos" and range $[2, 6]$ reaches CLP_m: The CLP has two possible directions to propagate the query. Direction B will be omitted, as the value range $[8, 9]$ does not overlap the window $(2, 6)$. However, the query will be forwarded towards direction A, because the corresponding value range $[1, 5]$ matches the condition.

Like the address-based routing, the range-based routing stores the value range for each port by segmenting it. But instead of only considering the neighbour CLPs, each bit marks the existence of matching SSVs *somewhere* behind the port. The port information will be kept up to date according to returned messages from neighbours; if an empty message is returned, there are no matching SSVs behind the corresponding port and this information will be referred to in the future. Obviously, the port information may need to be updated when any SSV's value changes.

This approach can also be applied to access SSVs by ID. In this case, the ID number range is segmented as well, so that ID queries can be resolved in a similar way as Range queries.

4. Model of the Simulation System

A comparison of the two proposed algorithms is not trivial, as it involves the evaluation of efficiency in performing Range queries and ID queries, complexity for maintaining routing information, complexity for maintaining range information, design complexity etc. From the scale of CLP tree and number of SSVs, it is relatively straightforward to estimate the computational and communication complexity of the address-based routing solution using mathematical approaches. However, the evaluation of range-based routing needs to consider other complicated factors at both application level and simulation level.

For the sake of a quantitative analysis, one approach would be to directly implement and integrate the two so-

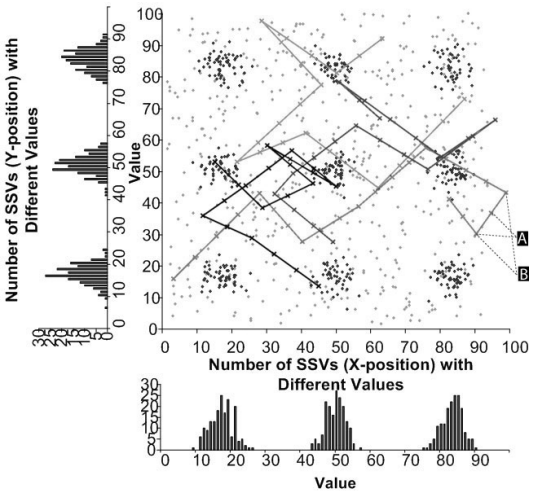

Figure 4. Environments and Agents' Behaviour Patterns

lutions into the PDES-MAS kernel. However, this would require considerable implementation efforts, and at least part of the implementation could be in vain, as the strategies may not meet the performance requirements. To avoid this, and to provide a generic framework for the study of dynamic data access in distributed systems in general, we have adopted a meta-simulation approach, as proposed for instance in [12, 18, 16].

For the meta-simulation we follow a layered approach similar to [3]. At the top layer we find the application model, which is responsible for generating realistic query patterns. The next layer is the middleware layer, where the routing strategies are described and the PDES-MAS framework is represented. The third layer, which typically is reserved for the network model, is implicitly represented in the performance measurements which are integrated by calculating the costs of queries in the second layer. Thus, similar to many simulations of P2P systems, the characteristics of the underlying network are abstracted away by only counting hops and messages.

Application Model. The application model focuses on the simulation of *situated* agents, namely, an agent has a position that determines its region of interest: only objects situated in the region can be accessed by the agent. In addition, situated agents are usually able to change their own positions.

This behaviour was modelled for a two-dimensional environment, as shown in figure 4. An agent moves step-wise towards a pre-selected target along the shortest path, and it randomly chooses a new target on arrival. The distance an agent can move in each step is referred to as *step size* (mark "A"). The distance of the new target, the *target distance* (mark "B"), is defined by the number of steps it takes the agent to reach it. The *step size* and *target distance* determine the activity scope and movement speed of an agent. After each step of movement, an agent generates ID or Range queries concerning its actual region of interest.

We assume that all SSV types within the MAS model have a *spatial* meaning, i.e. the value ranges for Range queries reflect the actual positions of the agents. Each SSV type represents a certain dimension of the environment, such as 'X-Pos' or 'Y-Pos'.

SSVs may have a uniform value distribution or multiple normal distributions. These are illustrated in figure 4 as light and dark dots respectively.

To investigate the impact of different SSV distributions within the CLP tree, a custom parameter has been defined: The *fluctuation* constraints the maximum difference between the largest and the smallest value of equally typed SSVs that are hosted on one CLP. It is defined as a ratio of the overall value range for this SSV type. For example, suppose $fluctuation_X = 0.05$ and SSVs of type X can have a value from 0 to 100: A CLP may host two SSVs of type X with values 80 and 82, but another SSV of type X, with value 75, cannot be hosted by the same CLP, because $82 - 75 > 5$. Since SSVs are not moved but have dynamic values, this condition holds only for the initial state.

PDES-MAS Model The model of the PDES-MAS framework is formed by a set of SSVs. Each SSV consists of (unique) ID, type, value and position in the CLP tree. The modelled CLP tree is binary and complete and therefore its structure can be defined by its depth.

Another important parameter is the number of segments used by both routing algorithms, which determines the granularity of the description of the value distribution of SSVs.

To eliminate a possible source of bias, no load management mechanism has been modeled, i.e. the SSVs could not migrate (as proposed in [15]), although the SSVs' distribution pattern differs for different runs. Nevertheless, the mutual impact of routing and load management could be considerable and should be subject of future research. The model was simulated using discrete time steps.

5. Results and Analysis

In this section we present a quantitative comparative analysis of the two routing algorithms. Two main metrics are used to evaluate the algorithms:

- The *number of messages* is the number of all messages that are generated by the routing algorithm in order to

resolve a query. Hence, the number of messages is a measurement of the overall bandwidth consumption.

- The *number of hops* is the maximum number of messages that had to be sent sequentially until the request could be resolved. This means, that the *number of hops* corresponds to the maximal path length from the ALP generating the request to a CLP which had to be contacted, multiplied by 2 (for the query and the corresponding response). Hence, the number of messages is a measurement of the overall latency.

In the experiments, the application model simulates 64 agents for 300 time steps, with 31 CLPs (i.e., a binary tree with depth 4) and each server CLP linking to 4 ALPs. There exists 12,400 SSVs of 16 different SSV types (775 SSVs per type, 8 types per dimension), which were uniformly distributed over the CLP tree. The initial SSV values are chosen from a uniform distribution of real numbers in $[0, 100)$. All agents move on a 100 x 100 torus. In other words, the scenarios represent a MAS operating in an environment with 8 different objects types, each of them consisting of two attributes defining their X and Y position.

The *step size* and *target distance* of an agent follow normal distribution with $\mu = 2.0$ and $\sigma = 1.0$ and $\mu = 5.0$ and $\sigma = 2.0$ respectively.

Each agent generates 8 random requests per time step. An agent's region of interest has a diameter of 2.0 for each dimension. For example, an agent at the position $X = 50$ could query a range of $[49, 51)$ for all types associated with this dimension. In the default setup, the fluctuation is 1.0 and value range is defined using 100 segments.

5.1. Evaluation based on SSV Properties

We have carried out a set of experiments to examine the effect of SSVs' value distributions and location constraints on the proposed routing algorithms.

In the first experiment, the initial SSV values have been assigned in a round-robin manner by one of three normal distributions, instead of being uniformly distributed. The mean values of these normal distributions were $16\frac{2}{3}$, 50 and $81\frac{1}{3}$. The deviation is varied from 0 to 10, making the SSV value distribution changing from highly concentrated to highly scattered.

As can be seen in figures 5 and 6, the results concerning hops and messages have a similar characteristic. The results indicate that the concentration of the SSV values has a huge impact on the routing performance of both algorithms ($\approx 500\%$): The more concentrated, the faster are both solutions, although the range-based approach adapts better to this parameter.

Another interesting study is how the configuration of the simulation infrastructure affects the performance of routing,

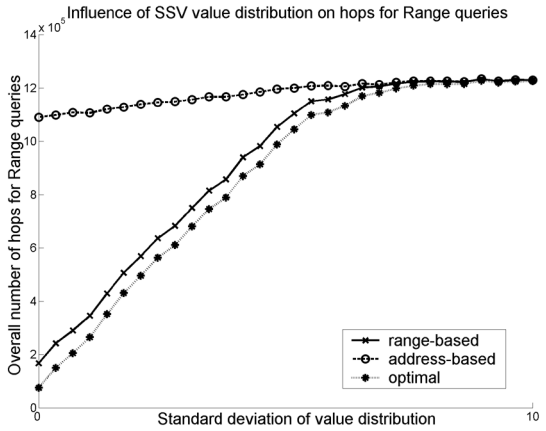

Figure 5. Hops traversed for Range queries, against different degrees of value concentration

such as a management policy regarding the SSVs' physical locations in the CLP tree. The policy is controlled by the $fluctuation$ parameter. Such a policy would, for instance, store SSVs with similar values on a single CLP. The effect can be reflected as in figure 7.

Given an arbitrary Range query, it is possible to calculate the probability of a single edge in the CLP tree to be used during a query. It assumes that the range-based approach has initialized all port information according to former queries, i.e. this is the best case. Let $ssvPerCLP_{type}$ be the number of SSVs each CLP is hosting per type.

Firstly, the edges from all server CLPs (except the one generating the query) to their parent CLPs in the tree are considered: These edges will only be used, if the CLP hosts an SSV of the specified type whose value is within the segments that cover the queried range. Let the probability that a single SSV is affected by a range query, given the number of segments (seg) and a query for the range (min, max), be $P^{type,seg}_{min,max}$. Hence, the probability that a link will *not* be used to resolve this query is:

$$P^{serverCLP}_{noFittingSSVs} = (1 - P^{type,seg}_{min,max})^{ssvPerCLP_{type}}$$

Clearly, this probability is very small if either $ssvPerCLP_{type}$ or $P^{type,seg}_{min,max}$ is significantly high. This equation holds true for both approaches.

The calculation of this probability for edges that are higher up in the tree demonstrates the major difference between the range-based and the address-based approach. The former considers the SSV values of all CLPs within the subtree to which the edge leads. In contrast, the address-based approach will propagate a query to any CLP whose subtree contains a CLP that hosts an SSV of the speci-

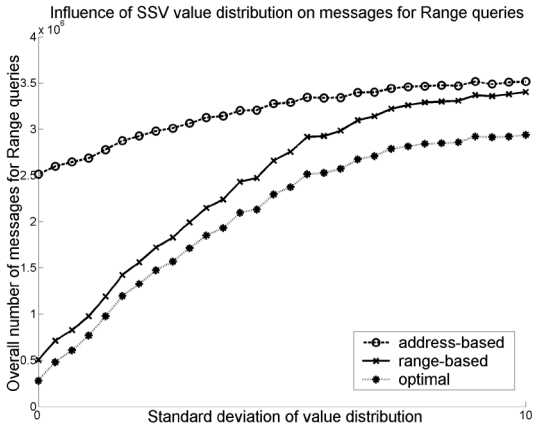

Figure 6. Messages generated for Range queries, against different degrees of value concentration

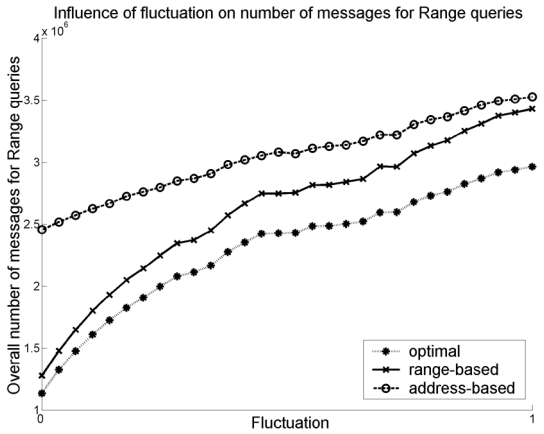

Figure 7. Messages generated for Range queries against different fluctuation values

fied type. For range-based routing, $P^{upperlevel}_{noFittingSSVs}$ depends on $P^{serverCLP}_{noFittingSSVs}$. For the address-based approach, $P^{upperlevel}_{noFittingSSVs}$ also depends on the probability of a CLP in the subtree hosting *any* SSV of this type.

The positive impact of concentrated SSV values (in figures 5 and 6) can be explained by the rather small $P^{type,seg}_{min,max}$ for all Range queries that did not cover high concentrations of SSV values. The same holds true for the experiment in figure 7, because introducing the fluctuation actually alters $P^{type,seg}_{min,max}$ on each individual CLP. For example, hosting only SSVs with higher values in the initial state would lead to a very small probability that this CLP is affected by Range queries covering lower values of this type.

The range-based approach benefits more from these situations, because its edge probabilities (between non-server CLPs) are more dependent on $P^{serverCLP}_{noFittingSSVs}$.

5.2. Evaluation based on Granularity

The experiments and results presented in this section mainly concern the granularity of segmenting value ranges. A larger number means a more precise attribute range description. The routing cost against segment number is reported in in figure 8. Increasing the number of segments leads to a reduction of generated messages. Although the performance of both routing algorithms are very similar for small segment numbers, range-based outperforms address-based routing when the number of segments is increased.

As mentioned above, both algorithms will show the same behaviour when accessing server CLPs for a Range query, but the range-based approach will save more communication between CLPs at higher levels. Of course, this is only possible if $P^{serverCLP}_{noFittingSSVs}$ is sufficiently high. One way of increasing this expression is to increase the number of segments, which explains the performance difference between both solutions (figure 8).

Consequently, it could be argued that the number of segments should be as high as possible, but a high number of segments also results in increasing storage and runtime requirements. Moreover, this way of optimization *only* increases the routing precision and can therefore only approximate an 'ideal' system, which would use a full list of SSVs instead of segments.

5.3. Evaluation based on Write Operations

In addition to routing Range queries efficiently, we have evaluated the performance of the routing algorithms for write operations (figure 9).

The number of update messages may strongly influence the speed of the PDES-MAS kernel, because the fast execution of write queries is crucial for optimistic simulations. If the write access to the shared state is too slow, this may provoke rollbacks [9].

Considering the nature of range-based routing, one could presume that the number of update messages it generates should be higher than that created by the address-based approach. However, this is not always the case, as illustrated in figure 9.

In the figure, the graphs of update messages and hops have very different characteristics, caused by the inherent properties of the solutions: An update message for the address-based approach will not be propagated further, since only the direct neighbours of a CLP need an update. On the other hand, *all* neighbours will receive a message in this case. Hence, given that the average degree of a node in

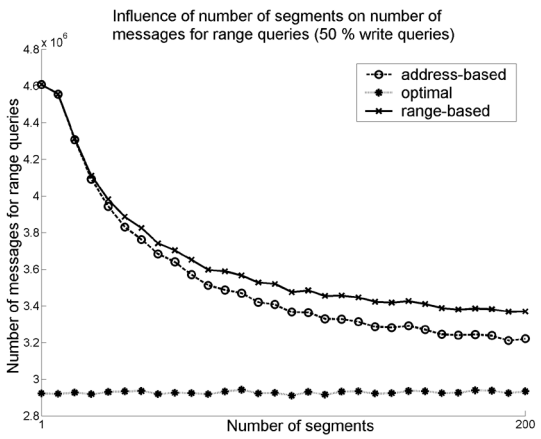

Figure 8. Number of messages generated for Range queries using different segment numbers

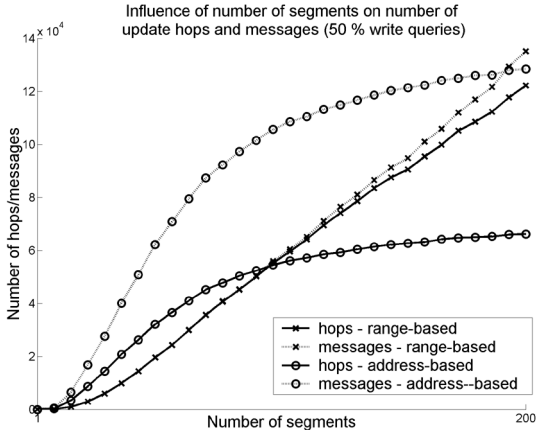

Figure 9. Number of update messages generated for write queries using different segment numbers

the CLP tree is ≈ 2, the message graph can be interpreted as the hop graph multiplied by the average degree of the CLP tree.

This is in contrast to the range-based approach, where the number of hops is very similar to the number of messages. This means that the message propagation is usually stopped after reaching the first neighbour, and often only one neighbour (instead of all) needs to be updated. This situation changes slowly when the number of segments is increased.

5.4. A Combined Comparison

Figure 10 illustrates a comparison of the two algorithms in terms of the difference in the total number of messages they generate (range-based minus address-based) and for different parameters.

Obviously, either algorithm may outperform the other one within certain regions of the parameter space. To facilitate an accurate comparison, we have also calculated the minimal number of hops and messages, as the optimal cost of each scenario. In this experiment, the probability for the occurrence of a Range query varies between 50% to 99% (with the rest being write queries) and the number of segments varies from 1 to 200. Each point is the result of a single simulation. A negative value means a better overall result for the range-based approach, whereas positive values show situations in which the address-based approach performed better.

6. Conclusions and Future Work

In this paper, we have identified efficient data accessing as a key issue to optimising the execution of MAS-based distributed simulations and we have described two different routing algorithms to achieve that in the context of the PDES-MAS framework.

The main conclusions drawn for the evaluation of the algorithms can be summarized as follows: (a) A highly ordered (low-fluctuation) system can significantly reduce (approx. 50%) the number of messages for Range queries. This fact needs to be considered in designing future data distribution mechanisms. (b) The address-based solution is superior to the range-based one when segments are very precise. When range-based routing is adopted, precise segmentation implies accurate routing, but this also leads to a large overhead in dealing with update queries. (c) The range-based solution with proper configuration can provide very efficient Range queries, whereas the address-based solution has an excellent performance for ID queries. A combination of both solutions would be desirable.

Future work will integrate the two proposed routing algorithms in the PDES-MAS kernel and evaluate their runtime performance. Another important issue is to analyse the impact of alternative load management mechanisms on the performance of the routing algorithms.

Acknowledgement

This work is part of the PDES-MAS project[2] and is supported by EPSRC research grant No. GR/R45338/01.

[2] http://www.cs.bham.ac.uk/research/pdesmas

Figure 10. Overall message difference between range-based and address-based solution

References

[1] An, N., J. Jin, A. Sivasubramaniam. 2003. Toward an Accurate Analysis of Range Queries on Spatial Data. IEEE Transactions on Knowledge and Data Engineering 15(2):305-323.

[2] Chawathe, Y., S. Ramabhadran, S. Ratnasamy, A. LaMarca, J. Hellerstein, S. Shenker. 2005. A Case Study in Building Layered DHT Applications, ACM SIGCOMM Computer Communication Review, volume 35, Issue 4: 97-108.

[3] He, Q., M. H. Ammar, G. Riley, H. Raj, R. Fujimoto. 2003. Mapping Peer Behavior to Packet-level Details: A Framework for Packet-level Simulation of Peer-to-Peer Systems. 11th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2003

[4] Jang, M., G. Agha. 2005. Agent framework services to reduce agent communication overhead in large-scale agent-based simulations, G. K. Theodoropoulos, H. Karatza, Eds, Simulation Practice and Theory Journal, Special Issue on Distributed Systems Simulation.

[5] Kuhl, F., R. Weatherly, J. Dahmann. 1999. Creating Computer Simulation Systems: An Introduction to HLA. ISBN 13-022511-8, Prentice Hall, USA.

[6] Logan, B., G. K. Theodoropoulos. 2001. The distributed simulation of multi-agent systems. Proceedings of the IEEE 89(2): 174-186.

[7] Lee, D.L., Xu, J., Zheng, B., Lee, W. 2002. Data Management in Location-Dependent Information Services, IEEE Pervasive Computing: 65-72.

[8] Lees, M., B. Logan, and G. Theodoropoulos. Adaptive optimistic synchronisation for multi-agent simulation. 2003. In D. Al- Dabass, editor, Proceedings of the 17th European Simulation Multiconference (ESM 2003):77¨C82.

[9] Lees, M., B. Logan, R. Minson, T. Oguara, G. K. Theodoropoulos. 2004. Modelling Environments for Distributed Simulation, 1st International Workshop on Environments for Multi-Agent Systems (E4MAS), in conjunction with the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS04).

[10] Lees, M., B. Logan, R. Minson, T. Oguara, G. K. Theodoropoulos. 2004. Distributed Simulation of MAS. Multi-Agent and Multi-Agent-Based Simulation: Joint Workshop MABS 2004: 25-36.

[11] Lees, M., B. Logan, D. Chen, T. Oguara, G. K. Theodoropoulos. 2005. Decision-Theoretic Throttling for Optimistic Simulations of Multi-agent Systems. Proceedings of the Ninth IEEE International Workshop on Distributed Simulation and Real-Time Applications: 179-186.

[12] Liu, J., D. Nicol, B. Premore, A. Poplawski. 1999. Performance Prediction of a Parallel Simulator, Proceedings of the 13th Workshop on Parallel and Distributed Simulation (PADS'99):156-164.

[13] Morse, K. L., M. D. Petty. 2001. Data Distribution Management Migration from DoD 1.3 to IEEE 1516, Proceedings of the Fifth IEEE International Workshop on Distributed Simulation and Real-Time Applications. August.

[14] Ndiaye, S., M. Tsangou, M. Seck, W. Litwin. 2003. Range Queries to Scalable Distributed Data Structure RP*. Proceedings of the Fifth Workshop on Distributed Data and Structures, WDAS 2003.

[15] Oguara, T., D. Chen, G. K. Theodoropoulos, B. Logan, M. Lees. 2005. An Adaptive Load Management Mechanism for Distributed Simulation of Multi-agent Systems. Proceedings of the Ninth IEEE International Workshop on Distributed Simulation and Real-Time Applications: 179-186.

[16] Perumalla, K. S. , R. M. Fujimoto, P. J. Thakare, S. Pande, H. Karimabadi, Y. Omelchenko, J. Driscoll. Performance Prediction of Large-Scale Parallel Discrete Event Models of Physical Systems. Winter Simulation Conference 2005

[17] Pollack, M. E., M. Ringuette. 1990. Introducing the Tileworld: Experimentally Evaluating Agent Architectures. AAAI

[18] Rajive, B., E. Deelman, T. Phan. 2001. Parallel Simulaiton of Lage-Scale Parallel Applications, The International Journal of High Performance Computing Applications, Volume15, No.1:3-12.

[19] Sycara, K. P. 1998. Multiagent Systems, AI Magazine: The American Association for Artificial Intelligence Volume 19, Issue 2: 79-92.

[20] Wyens, D., H. Parunak, F. Michel, T. Holvoet, J. Ferber. 2005. Environment for Multi-agent Systems, State-of-art and Research Challenges, Volume 3374, Lecture Notes in Computer Science: 1-48.