Synchronised Range Queries in Distributed Simulations of Multi-Agent Systems

Vinoth Suryanarayanan, Bart G.W. Craenen, Georgios K. Theodoropoulos
School of Computer Science
University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
Email: {V.Suryanarayanan |B.G.W.Craenen |G.K.Theodoropoulos}@cs.bham.ac.uk

***Abstract*—Range-Query is an important associative form of data access in distributed simulations and Distributed Virtual Environments. This paper discusses the problem of Range-Queries in the context of distributed simulation of multi-agent systems. An algorithm is presented for performing instantaneous Queries within an optimistic synchronisation framework and in the presence of dynamic migration of the simulation state. A quantitative evaluation of the effectiveness of the algorithm under different conditions is also presented.**

***Keywords*-Range Queries, Distributed Simulation, Multi-Agent Systems, Load Balancing, Distributed Virtual Environments**

I. Introduction

Very large distributed data structures are used more than ever before in the deployment of distributed applications. As these applications become larger, more data-intensive and latency-sensitive, scalable data access becomes a crucial element for their successful deployment. An approach to address the scalability problem in this context is to build systems in a way that that the flow of data is optimised to reflect the "interests" of the user population - a paradigm generally referred to as Interest Management (IM). Data access comes in two different forms, namely access via *ID-Queries* and access via *Range-Queries*.

An ID-Query is taken to mean a read operation which obtains the current value of a data item given its identifier, assumed to be unique in the system.

A Range-Query is an operation obtaining a set of data items each of whose current value matches a given predicate, expressed as a contiguous range between two values. The semantics of a Range-Query are not formally defined and are different for different systems and applications but in general two different forms may be distinguished: (a) instantaneous **Queries** of the set of currently extant data items whose value matches some predicate, abd (b) persistent **Subscriptions** to all possible future values data items may take which match some predicate.

Range-Query, with different terminology and semantics, can be encountered in several areas and at different levels in computer systems research. In the area of Peer-to-Peer systems, a significant body of recent work has looked at ways to extend the fundamentally ID-Query oriented nature of DHTs to efficiently support Range-Query operations utilising hash-based and hash-free approaches. Most work in this area is geared towards Instantaneous Queries (e.g.[1]).

In traditional distributed simulations, which are based on message passing, the problem of ID-Queries has been addressed through the development of Distributed Shared Memory mechanisms (DSM) [2]. More recently, the application of the PDES paradigm to more non-traditional simulation tasks with Range-Query functional requirements, such as complex system or multi-agent system models has led to efforts to bridge the gap between traditional PDES and this, increasingly relevant, form of simulation model [3, 4]. Most of the relevant work in this direction has been conducted in the fields of large scale distributed simulation (HLA) and more generally Distributed Virtual Environments (DVEs). In HLA, Cell-based architectures map queries to multicast groups that correspond to discrete cells in a grid overlay [5–9] while Region-based approaches calculate explicitly the overlaps between regions themselves [10]. Similarly, several cell-based [11–13], zone-based [14] and aura-based [15, 16] IM approaches have been proposed to support Range-Queries in the context of DVEs and Massively-Multiplayer Online games.

In [17] we discussed an approach for implementing synchronised instantaneous Range-Queries in the context of PDES-MAS a distributed simulation infrastructure for multi-agent systems. In this paper we extend this approach to cater for the dynamic reconfiguration of PDES-MAS through state migration.

The rest of the paper is organised as follows: Section II provides a short description of the PDES-MAS kernel. Section III outlines the design of logical-time synchronised Range Queries in PDES-MAS and presents an algorithm for state migration to support Range Queries. Section IV provides an quantitative analysis of the proposed approach for different MAS simulation configurations. Finally, Section V summarises the main conclusions and outlines future work to be carried out in this area.

II. PDES-MAS

PDES-MAS is a framework for the distributed simulation of multi-agent systems [3]. It is based on a DSM structure which is used to represent the shared state (the public variables, including publicly accessible attributes of agents)

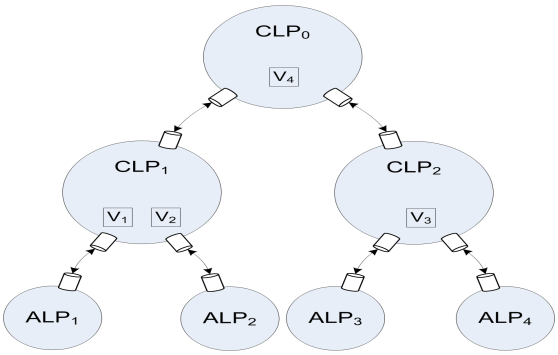


Figure 1: A PDES-MAS tree with 4 ALPs and 3 CLPs simulating a model with 4 Shared State Variables.

of the simulation. DSM variables are represented by Shared-State Variable (SSV) data-structures which store the history of values taken by a particular variable over time [4]. Agents in the MAS being modeled are assigned to Logical Processes, known as Agent-Logical-Processes (ALP).

The primary philosophy of PDES-MAS is to provide multiple ALPs concurrent access to the set of SSVs in a scalable manner by a balanced distribution of SSVs around a tree-like network of servers known as Communication Logical Processes (CLPs) as depicted in Figure 1. Accesses to SSVs held by remote CLPs are forwarded until they reach their destination. The CLP tree is reconfigured dynamically and automatically to reflect the interaction patterns between the agents and their environment so that SSVs which are accessed most frequently by a given ALP are as close as possible to that ALP in the tree. The aim is to concurrently minimise the average hops an access would take to reach an SSV and the load imbalance between CLPs.

In response to the sensing and acting of agents in the simulation, ALPs perform ID-Query reads and writes on SSVs in the system. A CLP is responsible for synchronising the read/write events it receives from ALPs. Synchronisation algorithms that have developed for PDES-MAS are reported in [4, 18, 19]. In general, each SSV is associated with a list of Write Periods representing the values taken by the variable at different logical times through the simulation. If a write period is subsequently invalidated by a straggler write, any ALPs which read that period must by rolled back.

III. Range-Queries in PDES-MAS

In [17] we presented a basic algorithm for performing logical-time synchronised range queries within the PDES-MAS system.To summarise, the algorithm parses the tree selecting the CLPs that match the "query window" and stops at a CLP beyond which there are no SSVs of interest. Thus a Range-Query from an ALP creates an "horizon" (propagation extent) through the tree over the CLPs that have all the SSVs evaluated for the range query. Each CLP has four ports namely, 'H' (Here, for local information) and 'L', 'R' and 'U'(Left, Right and Up, external ports linking to neighbouring CLPs). These ports maintain range values (representing the (min, max)-values) of all SSVs within a CLP (H) as well as in CLPs beyond the port. A Range-Query is recorded at all ports of the CLPs inside the horizon. Any change to the value of an SSV inside the horizon could rollback a range query directly, while SSVs outside the horizon have to send range updates to rollback a range query indirectly. To enable synchronisation, the history of range values has to be maintained at each port.

The algorithm presented in [17] assumed fixed locations of SSVs in the tree and did not consider the dynamic reconfiguration of the tree. In this paper we focus on the latter. The hypothesis is that the reconfiguration of the tree results at the dynamic reduction of the horizon size of the range queries travelling over the CLP tree.

A. Range Queries in the presence of State Migration

There are different ways to reconfigure the CLP-tree, from splitting and merging CLPs, to migrating ALPs and/or SSVs through the tree. In this paper we assume a fixed tree of CLPs with ALPs as leaves in fixed locations and SSVs migrating through the tree closer to the ALPs that access them. SSV migration is achieved by means of a competitive optimisation algorithm reported at [20]. Following this approach, a CLP monitors the access patterns of its SSVs at each port maintaining access cost information for each SSV it hosts. The total access cost of a SSV is a function of the number of accesses and the number of hops for each access. SSVs are migrated through the port with the highest access cost over a period. Several threshold parameters are utilised to avoid thrashing and ensure a balanced load across CLPs. This basic algorithm is extended in this paper to support Range-Queries.

Migration of an SSV as achieved by deleting the SSV from one CLP and adding to another CLP together with its entire history, namely the list of *Write Periods* (A write period is the validity period of a value with a start and end time [19]). The ports are updated at both CLPs and does not affect prcessing any transient ID query/write operations.

To support Range-Queries however, migrating an SSV with its Write Period history also means changing the Range Period history at both CLPs (namely the validity periods of range information of SSVs within a CLP [17]). To achieve this, each Range Period has to be evaluated against the list of SSVs selected for migration, again with its entire history. This is because each write period of an SSV could contribute to one or more Range Periods (depending on the validity period of the value). The migration algorithm within each CLP is outlined in the following steps:

1) *Populate SSV List*: Generate lists of SSVs and the ports to which they have to be migrated.
2) *Initiate Global Virtual Time*: Whenever the migration algorithm selects an SSV, it first initiates Global

Virtual Time (GVT) to garbage collect the shared state from the simulation. The objective of this operation is to reduce the overhead of evaluating the Range Periods. Since every action is timestamped and recorded for rollbacks, it has to be ensured that the SSVs are no longer required for any LP in future. For the work presented in this paper, Mattern's GVT algorithm [21] is used in PDES-MAS to calculate the global minimum time of all send, receive and transient events over all LPs.

3) *Delete SSVs from the CLP*: Upon completion of the GVT operation, the SSVs selected for migration are deleted. Deletion of a SSV with a list of write periods from a CLP is implemented as a list of anti-writes/rollbacks. An anti-write/rollback operation also handles evaluating the Range Periods generating rollbacks and range updates to neighbouring CLPs.
4) *Add SSVs to the destination CLP*: The SSVs are migrated to the appropriate destination/neighbouring CLP. The receiving CLP accepts the SSV and adds it and its list of write periods. Adding an SSV to a CLP is implemented as a list of write operations. The write operation also handles evaluating the Range Periods and generating rollbacks and range updates to neighbouring CLPs.

The advantage of coinciding GVT calculation with SSV migration is that the migration process will not affect any transient messages in the simulation and that Range Queries over the CLPs and Range Updates are synchronised correctly. However, an increased overhead is generated in the form of additional rollbacks and range updates. The following section provide a quantitative analysis of the proposed approach.

IV. Experimental Investigation

The overall objective of state migration is to improve the performance of the system by moving SSVs closer to the accessing ALPs. Moving SSVs closer to the accessing ALPs will reduce the number of hops needed to access the SSVs resulting in a reduction in the average access cost of the SSVs. But state migration also comes with a cost; each state migration initiated also initiates a rollback and moving SSVs means an increase in the number of updates when they are moved from one CLP to another.

In this section we present an experimental investigation into both effects in order to evaluate whether the access cost reduction of state migration outweights its inherent cost. Two experiments will be presented:

1) Investigating the impact of range query propagation and State Migration on the access cost of the system; and
2) Investigating the impact of increased rollback volatility and State Migration cost on average simulation time.

Experimental results both with and without state migration will be presented to allow for a clear comparison between the two configurations. An effort has been made to keep the experimental setup similar to the one used in [17] so that the results of both papers remain comparable as well.

The experimental platform included an implementation of PDES-MAS in C++ being fed simulation traces that provide read, write, and range query operations issued by ALPs. These traces were generated using variable models implemented with the MWGrid MAS toolkit under development at the University of Birmingham[1].

A. Experimental Setup and Parameters

The experimental setup used in this paper is similar to that in [17]. It consists of a CLP-tree of 7 CLPs, the root CLP of which has an initial 7 clusters of 50 SSVs. All SSVs were placed at the root CLP to more clearly show the effect State Migration has on performance. SSV values ($\langle x, y \rangle$) lie within a 2-dimensional radius around a randomly selected central point. The radius is determined by the *SSV range* experimental parameter. 8 ALPs, two attached to each leaf CLP are used to issue 2-dimensional range queries, one per simulation tick. Range queries are issued with a static position and a radius corresponding to the *RQ size* experimental parameter. Simulations are run for 1000 ticks.

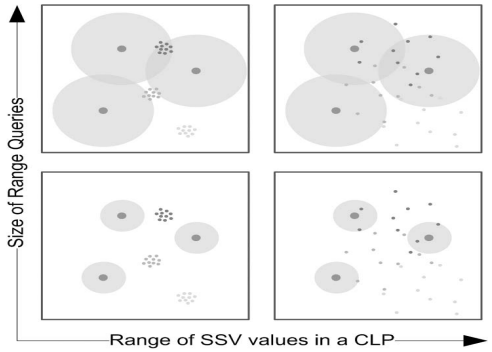


Figure 2: The logical view of the traces used to investigate the extent of Range Query propagation under different conditions.

The traces used were generated by varying the two experimental parameters: *SSV range*, and *RQ size*; both proportional to the size of 2-dimensional area covered by the simulation. Figure 2 shows a logical view of the simulation setup for different values of the experimental parameters. Varying the two experimental parameters will have an effect on the experimental behaviour of PDES-MAS.

The relationship between the *SSV range* and *RQ size* experimental parameters is determined by the area polled by the range queries and the amount of clustering of the SSVs in the 2-dimensional experimental area. When the area polled

[1] http://www.cs.bham.ac.uk/research/projects/mwgrid

by the range queries is small (*RQ size* small), and the SSVs are clustered tightly around the randomly chosen centre-points (*SSV range* small), there is a high probability of the range queries only accessing few if any SSVs everytime they are issued. When the *RQ size* parameter is increased while the *SSV range* remains small, the probability of accessing SSVs increases, and if one SSV is accessed, the probability of accessing the other SSVs in the cluster is also high. Given that the *RQ size* parameter adjusts the radius of the range query, the probability increase is likely to be quadratic, although diminished by overlapping areas. If the *SSV range* parameter is increased while the *RQ size* remain small, the probability of accessing the SSVs increases as well, as they are more spread out over the simulation area but the probability of accessing more SSVs in a cluster decreases proportionally. When both the *SSV range* and *RQ size* parameters are increased in unison a certain average behaviour can be observed. Note that for both parameters and phase transition can be observed when they are increased. For both parameters, each increase of the parameter value will also increase the probability of accessing more SSVs with the rate of increase flattening out and eventually levelling off. The rate of increase will be smaller for the *SSV range* parameter than it is for *RQ size* parameter.

In total $5 \cdot 5 = 25$ traces were generated for 5 values for both *SSV range* and *RQ size* (0.1, 0.2, 0.3, 0.4, and 0.5 proportions of the simulation area considered), and 3 experiments were done for each trace with different pseudo random number generated seed values for $3 \cdot 25 = 75$ experiments in total.

B. Range Query Propagation

In the first experiment we want to quantify the effect range query propagation and state migration has on the access cost of the SSVs. To isolate this effect, the traces generated for this experiment did not include any write-events. As such, any rollbacks initiated will be caused by state migration alone without further dilution from non-deterministically caused rollbacks by write-events in the traces themselves. With the simulations run for 1000 ticks, the ALPs will issue 1000 range queries as well.

Range Query propagation is measured by the number of hops it takes to fetch the SSVs in the CLP-tree. Without state migration the number of hops required would be 7; 3 hops from the leaf CLP to the root, 3 return hops, and an extra hop to the ALP. The same hop counting method is used for both experiments. The number of SSVs accesses, and thus the total number of hops for an experiment, is determined by the combination of the experimental parameters: *SSV range* and *RQ size*.

Figure 3a shows the average number of hops required to access SSVs and their deviation from the mean for varying *RQ size* values. The results were averaged over the various *SSV range* values without loss of accuracy. The results

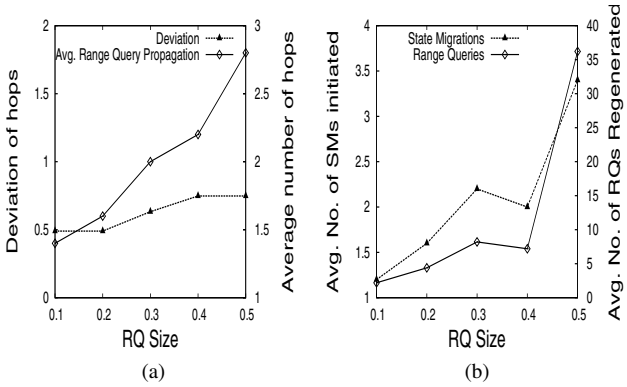


Figure 3: Average number of hops (with its deviation from the mean) and average number of State Migrations initiated and Range Queries regenerated for varying *RQ size*.

suggest a linear relationship between the average number of hops and the *RQ size* parameter, with the average number of hops ranging from 1.4 to almost 3 for *RQ size* from 0.1 to 0.5. In this range, the standard deviation from the average number of hops indicates a linear increase from 0.4 to 0.7. The results suggest that the increase in variance is caused by a higher probability for SSVs to be localised with minimum or no overlap with range queries issued from different ALPs for smaller *RQ size* values. With minimum or no overlap, SSVs will eventually be migrated to the leaf CLPs. When the *SSV range* and/or *RQ size* parameter is increased, a correspondingly higher likelihood of having range query overlapping each other means an increased likelihood of SSVs remaining at intermediate CLPs so as to be more efficiently available for several ALPs. This increases the spread of the average number of hops required to access these SSVs with the corresponding increase in the variance and standard deviation of this measure. Overall, the experimental results show a clear reduction of the average number of hops required to access the SSVs from the constant 7 hops required to access them in the root CLP. With an average of around 3 hops required in the worst case, this shows that state migration has a substantial effect on range query propagation.

Range Query propagation on its own only shows one aspect of overall picture though. State Migration's extra cost has to be taken into account. We express the extra cost incurred by state migration by measuring the number of state migrations initiated and the number of range queries regenerated with the underlying assumption that the number of State Migrations initiated is directly correlated with the number of range queries regenerated.

Figure 3b shows the average number of state migrations initiated and range queries regenerated for varying *RQ size* values. The figure supports our expectation that the average number of State Migrations initiated increases

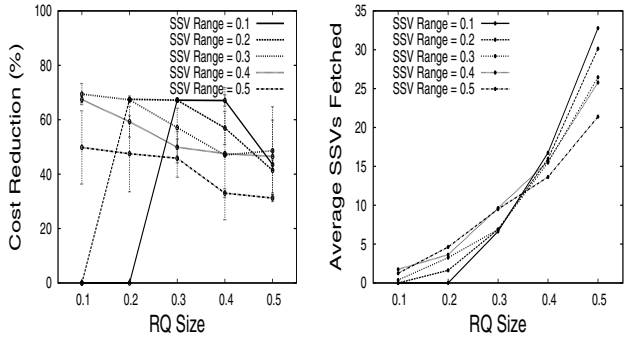
with increased *RQ size* values. The average number of range queries regenerated follows this trend closely with the maximum number of state migrations initiated reaching almost 4 and the number of range queries regenerated almost 35 for *RQ size* 0.5. Although the number of range queries regenerated is almost a factor of 8 times the number of state migrations initiated, compared to the total number range queries generated (1000 per ALP), this should still be considered to a low number.

Thusfar, the results show that including state migration decreases the number of hops, but also increases the number of range queries regenerated, in turn increasing the number of accesses. What remains is to see how this contributes to the overall access cost of the simulation. The access cost of a simulation is the sum of the cost of accessing SSVs in the CLP-tree. The cost of accessing a SSV in the CLP in turn is subject to the number of range queries (the number of accesses) and the number of hops needed to reach them in the CLP-tree.

The access cost of the simulation without and with state migration is compared by calculating the difference ratio in percentages called the *Cost Reduction*: C_r. *Cost Reduction* (C_r) is calculated as follows:

$$C_r = \frac{C_{-SM} - C_{+SM}}{C_{-SM}} \cdot 100 \quad (1)$$

With C_{-SM} the access cost of the simulation without state migration and C_{+SM} the total access cost of the simulation with state migration.



(a) (b)

Figure 4: Cost Reduction C_r (standard deviation) in percentages and average Number of SSVs fetched for different *SSV range* and *RQ size* combinations

Figure 4a shows the *Cost Reduction* for different *SSV range* and *RQ size* values. The C_r with state migration averaged over both *SSV range* and *RQ size* is 47.12 but the figure suggests a lot of variance for different *SSV range* and *RQ size* combinations with a decrease in C_r when *SSV range* and *RQ size* increase. The highest *Cost Reduction* was achieved with *SSV range* 0.3 and *RQ size* 0.1 for a reduction of almost 70%, while the lowest *Cost Reduction* was found to be with *SSV range* 0.5 and *RQ size* 0.5.

Figure 4a also shows the standard deviations from the averages *Cost Reduction*. The trend shown in the figure suggests that the variance decreases when the *SSV range* and *RQ size* values increase.

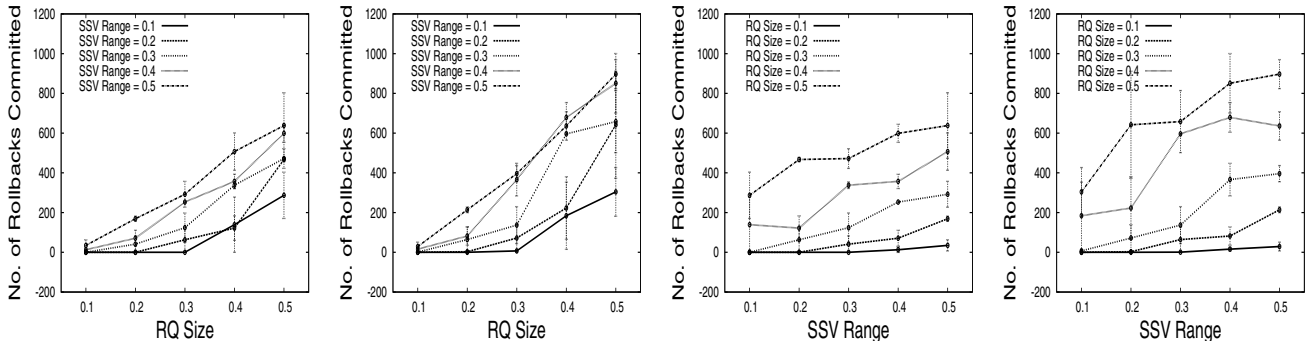
For a few *SSV range RQ size* combinations, no observable *Cost Reduction* was observed, meaning that the access cost both without and with state migration remained the same. To investigate this we present the average number of SSVs fetched by the range queries for different *SSV range* and *RQ size* combination in figure 4b.

The figure shows that when the *SSV range* and *RQ size* parameters are small, on average no SSVs were fetched. For these *SSV range* and *RQ size* values, the area covered by the range queries is so small, or, alternatively, the area in which the SSVs are concentrated is so small, that no SSVs matched the range queries issued by the ALPs. Since state migration only updates access cost with SSVs that match the range query predicate, in these instances, no state migrations were ever initiated, and without these no cost reduction was affected. In conclusion, the results show that for state migration to effect the greatest cost reduction, the *SSV range* and *RQ size* parameters should be large enough to at least fetch some SSVs while keeping them small enough so that SSVs access is localised enough for SSVs to migrate closest to the ALPs that will access them. In these circumstances, state migration will reduce range query propagation significantly with a enough positive effect on the access cost of the simulation to off-set the extra cost incurred by state migration.

C. Rollback volatility

Thusfar we have considered experimental traces without any write-events in order to clearly expose the effect state migration has on access cost. Experimental traces without write-events do no initiate any rollbacks from straggler writes, and do not expose the effects of the interactions between rollbacks initiated from different sources. In the following experiment we do include write-events in the experimental traces so that we can fully assess the impact of rollbacks on overall simulation time. The same experimental setup as used in the first experiment is used here with write-events increasing the value of the SSVs by 1. Write-events are issued by all ALPs to SSVs randomly selected from all SSVs. As before, all SSVs are placed at the root CLP in the CLP-tree, with the *SSV range* and *RQ size* parameters varied.

Figure 5 shows the rollbacks committed per ALP for varying *SSV range* and *RQ size* combinations. Four graphs are shown with the first two graphs show the results when the *RQ size* parameter was varied plotted for different *SSV range* parameter values (with and Without SM). The last two graphs show the results with the *SSV range* parameter was



(a) Avg. Rollbacks Committed without State Migration (b) Avg. Rollbacks Committed with State Migration (c) Avg. Rollbacks Committed without State Migration (d) Avg. Rollbacks Committed with State Migration

Figure 5: Rollbacks committed per ALP for varying *RQ size* and *SSV range* with their standard deviations.

varied plotted for different *RQ size* parameter values (with and Without SM). As the results were averaged over 3 runs with different psuedo random number generator seeds, the standard deviation for each result in the graphs is depicted using error-bars.

The order in which the events, both reads and writes, are processed by PDES-MAS is important in that an ALP commits a rollback if it arrives with a timestamp earlier that its local time (straggler-events), committing the event otherwise [3, 19]. A committed rollback also has the potential to regenerate range queries. As such, we expect an increase in the number of rollbacks committed by the ALPs for increasing *SSV range* and *RQ size* parameter values. In addition, with state migration moving SSVs around the CLP-tree, we also expect more rollbacks regenerating range queries.

The result presented in figure 5 bear out this expectation. The number of rollbacks increase almost linearly with larger *SSV range* and *RQ size* parameter values. With *RQ size* 0.1, the number of committed rollbacks is close to 0, both with and without state migration. Without state migration, the maximum number of rollbacks reaches 800, with state migration it reaches 1000, for both *SSV range* and *RQ size* 0.5. Although there is substantial variance in the standard deviation values, the overall trend is that it is relatively small upto *SSV range* and *RQ size* 0.3 suggesting few range queries regenerated. Beyond that there is a large amount of range query regeneration. In general, as expected, the number of rollbacks committed with State Migration is higher than with state migration.

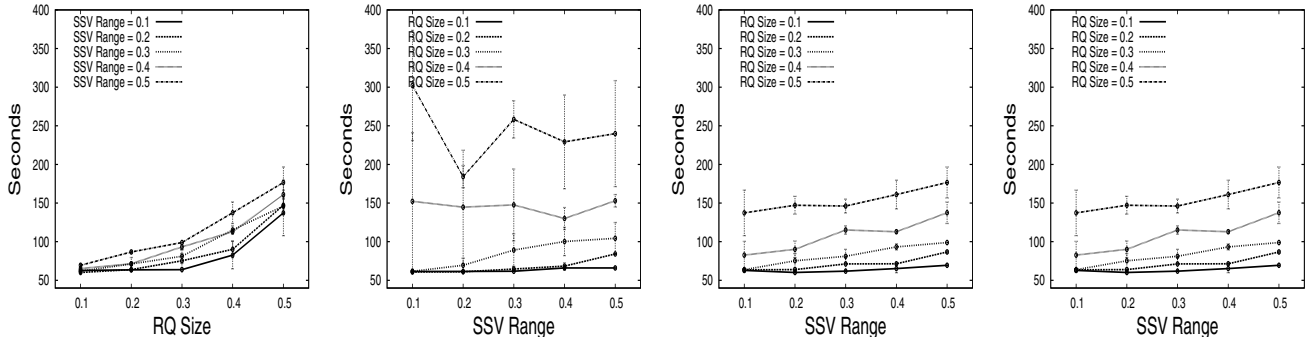
Based on these results we expect that as the *SSV range* and *RQ size* parameters increase, the simulation time required to finish the experimental traces will also increase. Figure 6 show the average simulation time for varying *SSV range* and *RQ size* parameter values with their standard deviations. The layout of the graphs is the same as described above for the number of rollbacks committed.

Figure 6 suggests that the average simulation time increases with increases in both the *SSV range* and *RQ size* parameters. Without state migration the average simulation time ranges from 60 seconds for *SSV range* 0.2 and *RQ size* 0.2 to 176 seconds for *SSV range* and *RQ size* 0.5. Again standard deviations over the averaged simulation time varies but suggest a minimum for *SSV range* and *RQ size* 0.3 beyond which the standard deviation reaches 20. This correlates strongly with the trend shown for the number of rollbacks committed as presented in figure 5.

With state migration the average simulation time increases linearly upto *RQ size* 0.3, increasing exponentially beyond that. Comparing the average simulation time without state migration with the average simulation time with state migration we see a small reduction of the average simulation time until *SSV range* and *RQ size* reach 0.3, after which the average simulation time with state migration exceeds that of the average simulation time without state migration.

This does not correlate exactly with the trend shown in figure 5. With the query propagation trend with state migration as shown in figure 3a we would expect more localised access patterns to effect a reduction of the number of rollbacks and consequently the simulation time. What should be noted though is that the writes in the simulation time are issued to random SSVs in the CLP-tree without consideration of where these SSVs are localised in relation to the range queries. This has two effects on the simulation based on the location of the SSV in the CLP-tree:

1) As the *RQ size* and *SSV range* increase, the overlaps of the range queries between different ALPS also increase. Though query propagation is reduced significantly by state migration (see figure 3a), writes to random SSVs in this scenario also increase the overlaps between ALPs. The reasoning behind this is that as the random writes change the value of a SSV, the possibility of the SSV overlapping range queries of different ALPS also increase as the *SSV range* and *RQ size* parameters increase. To further quantify this effect we measured the *Query Response Time* over a



(a) Average Simulation Time without State Migration (b) Average Simulation Time with State Migration (c) Average Simulation Time without State Migration (d) Average Simulation Time with State Migration

Figure 6: Average Simulation Time for varying *RQ size* and *SSV range* with their standard deviations.

simulation run. With state migration, changes in the *Query Response Time* should be minimal but with state migration the *Query Response Time* depends on range query propagation in the CLP-tree, and variances could be large.

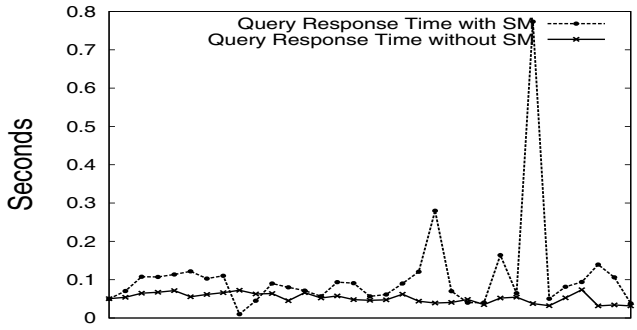


Figure 7: Query Response Time with and without State Migration for a typical simulation run.

Figure 7 illustrates this pattern by showing *Query Response Time* during a typical simulation run both with and without state migration. As expected, figure 7 shows that *Query Response Time* for simulation without state migration hardly alters during the run. *Query Response Time* with state migration however shows an initial increase but a subsequent reduction, suggesting more localised access patterns. But as the simulation progresses we see peeks of dramatic increases in *Query Response Time* followed by equally dramatic decreases. This suggests an interaction between the random writes and the State Migration with a dramatic decrease in efficiency by a confluence of random writes followed by state migration reacting on this. State migration is always reacting on these instances and in the end, averaged over the run *Query Response Time* is negatively affected.

2) Rollback depth is increased with localised access patterns. Rollback depth is measured as the difference between the local time of a rollback and the ALP committing the rollback. All range queries and writes generated in this time period will be rolled back and regenerated. When *RQ size* is small, the SSVs are moved much closer to the ALPs accessing them (see figure 3a) and as such we should observe a significant reduction in simulation time. However, as the SSVs move closer to the leaf CLPs, a random write from a remote ALP will need more hops to update the SSV while having a higher probability of triggering a rollback because of the increased time needed to traverse the CLP-tree as well as increasing the *Rollback Depth*.

RQ size	***SSV range***				
	0.1	0.2	0.3	0.4	0.5
0.1	0	0	14.5	19.52	15.91
0.2	0	13.85	1.91	12.23	6.10
0.3	3.42	9.09	9.24	4.68	4.16
0.4	7.19	8.57	4.35	3.34	2.94
0.5	9.93	2.48	2.46	3.34	3.37

Table I: Average Rollback Depth for different *SSV range* and *RQ size* combinations

This effect is illustrated by the average rollback depths found for different *SSV range* and *RQ size* parameter values as shown in table I. Although no linear trend can be distinguished for all *SSV range* and *RQ size* combinations, we note that for *RQ size* 0.1 this depth is almost 10 whereas for *RQ size* 0.5 it is just 4.3. This suggests that as the *RQ size* increases, the rollback depth decreases with more SSVs remaining at intermediate nodes where rollbacks are reached faster. Overall this suggests that although the number of rollbacks committed are fewer (see figure 5), the depth of these rollbacks is higher, making them more expensive to perform and thus negatively affecting simulation time.

In summary we conclude that the inclusion of state migration has a mixed effect on simulation time. For small *SSV range* and *RQ size* inclusion of state migration results to a clear decrease in the simulation time. In this range state migration moves the SSVs close enough to the accessing ALPs for the decreased access cost of these SSVs to outweight the extra costs incurred through the need for more rollbacks and updates necessitated by state migration. Beyond these *SSV range* and *RQ size* values the inclusion of state migration yields an increased overhead. This is caused primarily by the extra number of rollbacks needed for state migration while when the number of rollbacks needed is not increased, performing these rollbacks is more expensive.

V. Conclusions

In this paper we have extended our previous work on logical-time synchronised Range Queries in a parallel simulation kernel for multi-agent systems to take into account dynamic migration of state variables. The state migration reduces the propagation extent of the Queries but introduces extra overhead in the system. We have evaluated the proposed approach using two experiments, with and without rollback scenarios for different parameters: Range Query Size, SSV Range Value and write delta. In the first experiment, we have shown a significant reduction in the access latency and cost of the simulation for varying Range Query Sizes and SSV Range Values. The second experiment showed that the volatility of rollbacks has a direct relation with varying Range Query sizes and SSV Range values. The volatility of rollbacks has an adverse effect with State Migration with minimum gain. We have also shown that the extra cost of migration process does not necessarily compromise the performance of the system. In the future we will focus on testing different threshold values and SSVs distributions and analyse the scalability of the overall system.

References

[1] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 353–366, 2004.

[2] H. Mehl and S. Hammes, "Shared variables in distributed simulation," in *PADS '93: Proceedings of the seventh workshop on Parallel and distributed simulation*. New York, NY, USA: ACM, 1993, pp. 68–75.

[3] B. Logan and G. Theodoropoulos, "The distributed simulation of multi-agent systems," *Proceedings of the IEEE*, vol. 89, no. 2, pp. 174–186, Feb 2001.

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

[5] D. V. Hook, S. J. Rak, and J. O. Calvin, "Approaches to relevance filtering," in *In Eleventh Workshop on Standards for the Interoperability of Distributed Simulations*, 1994, pp. 26–30.

[6] A. Boukerche and A. Roy, "Dynamic grid-based approach to data distribution management," *J. Parallel Distrib. Comput.*, vol. 62, no. 3, pp. 366–392, 2002.

[7] M. R. Macedonia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, and P. T. Barham, "Exploiting reality with multicast groups," *IEEE Comput. Graph. Appl.*, vol. 15, no. 5, pp. 38–45, 1995.

[8] K. L. Morse, "An adaptive, distributed algorithm for interest management," Ph.D. dissertation, Information and Computer Science, 2000, chair-Bic, Lubomir and Chair-Dillencourt, Michael.

[9] A. Berrached, M. Beheshti, O. Sirisaengtaksin, and A. deKorvin, "Approaches to multicast group allocation in hla data distribution management," in *In Proceedings of the 1998 Spring Simulation Interoperability Workshop*, 1998.

[10] K. L. Morse and M. Zyda, "Multicast grouping for data distribution management," *Simul. Pr. Theory*, vol. 9, no. 3-5, pp. 121–141, 2002.

[11] B. Knutsson, M. M. Games, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," in *In Proceedings of INFOCOMM 2004*, March 2004.

[12] A. P. Yu and S. T. Vuong, "Mopar: a mobile peer-to-peer overlay rrchitecture for interest management of massively multiplayer online games," in *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*. New York, NY, USA: ACM, 2005, pp. 99–104.

[13] R. Minson and G. Theodoropoulos, "Adaptive support of range queries via push-pull algorithms," in *PADS '07: Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 53–60.

[14] J. W. Barrus, R. C. Waters, and D. B. Anderson, "Locales: Supporting large multiuser virtual environments," *IEEE Computer Graphics and Applications*, vol. 16, pp. 50–57, 1996.

[15] G. Morgan, K. Storey, and F. Lu, "Expanding spheres: A collision detection algorithm for interest management in networked games," in *In Proceedings of the Entertainment Computing ICEC 2004: Third International Conference*, 2004, p. 435.

[16] S.-Y. Hu and G.-M. Liao, "Scalable peer-to-peer networked virtual environment," in *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*. New York, NY, USA: ACM, 2004, pp. 129–133.

[17] V. Suryanarayanan, R. Minson, and G. Theodoropoulos, "Synchronised range queries," in *Proceedings of the 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2009)*, S. J. Turner, D. Roberts, W. Cai, and A. El-Saddik, Eds. Singapore: IEEE Press, Oct 2009, pp. 41–47.

[18] M. Lees, B. Logan, and G. Theodoropoulos, "Analysing probabilistically constrained optimism," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 11, pp. 1467–1482, August 2009.

[19] M. Lees, B. Logan, and G. Theodoropoulos, "Using access patterns to analyze the performance of optimistic synchronization algorithms in simulations of mas," *Simulation*, vol. 84, no. 10/11, pp. 481–492, October 2008.

[20] T. Oguara, D. Chen, G. Theodoropoulos, B. Logan, and M. Lees, "An adaptive load management mechanism for distributed simulation of multi-agent systems," in *Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2005)*, A. Boukerche, S. J. Turner, D. Roberts, and G. Theodoropoulos, Eds. Montreal, Quebec, Canada: IEEE Press, Oct 2005, pp. 179–186.

[21] F. Mattern, "Efficient algorithms for distributed snapshots and global virtual time approximation," *Journal of Parallel Distributed Computing*, vol. 18, no. 4, pp. 423–434, 1993.