Parallel and Distributed Spatial Simulation of Chemical Reactions

Matthias Jeschke[1], Alfred Park[2], Roland Ewald[1],
Richard Fujimoto[2], Adelinde M. Uhrmacher[1]

[1]University of Rostock, 18059 Rostock, Germany,
{mj118, re027, au007}@uni.rostock.de
[2]Georgia Institute of Technology, Atlanta GA 30332-0765,
{park, fujimoto}@cc.gatech.edu

Abstract

The application of parallel and distributed simulation techniques is often limited by the amount of parallelism available in the model. This holds true for large-scale cell-biological simulations, a field that has emerged as data and knowledge concerning these systems increases and biologists call for tools to guide wet-lab experimentation. A promising approach to exploit parallelism in this domain is the integration of spatial aspects, which are often crucial to a model's validity. We describe an optimistic, parallel and distributed variant of the Next-Subvolume Method (NSM), a method that augments the well-known Gillespie Stochastic Simulation Algorithm (SSA) with spatial features. We discuss requirements imposed by this application on a parallel discrete event simulation engine to achieve efficient execution. First results of combining NSM and the grid-inspired simulation system AURORA *are shown.*

1. Introduction

An important application domain for advanced simulation techniques is the field of *systems biology* [14], which aims at integrating biology with system theory, mathematics, and computer science among others. While modeling and simulation has been applied to gain a better understanding of biological systems for well over four decades, broad interest in applying modeling and simulation in cell biology has been renewed by recent developments in experimental methods, e.g. high content screening and microscopy. Further, the importance of modeling and simulation in this domain is highlighted by the insight from the Human Genome Project that simply knowing an organism's DNA sequence is not going to answer many questions.

The development of valid, predictive models is now one of the major challenges for cell biologists. A variety of simulation methods are exploited in cell biology. Since the millennium, the significance of noise in cellular information processing has become widely accepted, so that stochastic discrete-event simulation has emerged as an established method to complement conventional ordinary differential equations as the norm in biochemical simulations. This, in turn, has lead to an increasing demand of efficient discrete-event simulation methods, which enable an in-depth analysis of such models [6]. In addition to noise, spatial aspects are receiving increased interest. In vivo experiments revealed that many intra-cellular effects depend on space, e.g. protein localization, cellular compartments and molecular crowding [13]. Approaches that support both stochasticity and space are therefore particularly promising [22, 5]. One of those is the Next-Subvolume Method (NSM) [7].

Simulations in system biology are often very time consuming, and it is almost always the case that many runs are required. Parallel and distributed simulation provides a means to help address this problem, however, a major impediment lies in gaining access to sufficient computational resources. Public-resource computing is a form of distributed computation where shared resources and idle machines are pooled together to form a larger, singular resource for computationally intensive programs. These infrastructures are usually managed by middleware. Such middleware is a natural fit for large-scale embarrassingly parallel codes with little or no data dependencies between portions of work. However, parallel discrete event simulations (PDES) exhibit data dependencies between partitions where data is exchanged via messages. Additional constraints are placed on these parallel programs such as strict ordering of events which must be processed in increasing time stamp order for correct execution and repeatability of results.

The AURORA [19] parallel and distributed simulation system allows PDES codes to run across loosely coupled machines, forming computational resources known as desk-

1087-4097/08 $25.00 © 2008 IEEE
DOI 10.1109/PADS.2008.20

IEEE computer society

Authorized licensed use limited to: McGill University. Downloaded on January 21, 2009 at 00:10 from IEEE Xplore. Restrictions apply.

top grids within organizations. Desktop grids are comprised of potentially idle desktops, laptops, and possibly even cluster machines. These resources can be a mixture of heterogeneous hardware architectures and operating systems with unpredictable uptimes and possibly intermittent network connectivity. AURORA provides resource matching of simulations to available workers while ensuring correct execution for discrete event simulation.

Despite the need for efficient execution mechanisms, the area of parallel and distributed discrete event simulation of cell biological systems is still in its infancy. Only a few approaches have been developed to date, most of which are spatio-temporal simulations. For example, [17, 4, 21] utilize the idea of local interactions. They focus on a time-stepped rather than on a discrete event approach. Therefore, experiences in exploiting a grid, e.g. [17], cannot easily be transferred to discrete event approaches.

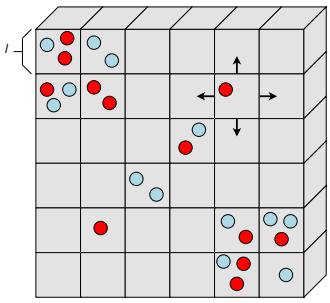

Figure 1. Discretization of a volume into smaller sub-volumes with side length l. The diffusion of molecules between neighboring sub-volumes is modeled as a Markov process with $d_i \Delta t = (D_i/l^2) \cdot \Delta t$ as the probability that a particular molecule of species i performs a diffusion during the infinitesimal small time step Δt (with D_i as the diffusion constant for species i).

2. Description of NSM

2.1. Spatial modeling and simulation algorithms

An overview of current approaches to modeling spatial features is given in [22]. Depending on the abstraction level a system is modeled, different formalisms and simulation algorithms can be used.

At the microscopic level, the fields of molecular dynamics [23] and Brownian dynamics permit the accurate simulation of individual interacting particles in continuous space but require high computational effort. Therefore, these approaches cannot be used for large scale models containing many particles. Examples for frameworks operating on this level are SmolDyn [1] and GFRD. Rather than considering individual particles and instead focusing on concentrations of species, the partial differential equations approach operates on a macroscopic level with continuous space and time.

Algorithm 1 Short version of the Next-Subvolume Method pseudo code description (for details, see [7])

Initialization

1. Distribute the initial particles over N sub-volumes
2. Calculate the sum of diffusion rates d_i and reaction rates r_i for all sub-volumes

$$s_i = r_i + d_i, 0 \leq i \leq N$$

3. Sample time of next event for all sub-volumes

$$\tau_i = \frac{-ln(u)}{s_i},$$

with u being a sample from the uniform distribution $U(0, 1)$ and enqueue sub-volumes according to their event times

Main loop

1. Take top sub-volume from event queue
2. Determine type of event w.r.t. to diffusion and reaction rates
   - Reaction: determine specific reaction, update the propensities of the current sub-volume and sample new event time (see [10] for details)
   - Diffusion: determine species type and diffusion target, update propensities and sample new event times for both sub-volumes

Here the system is modeled as a set of partial differential equations that can either be solved analytically or simulated by numerical integration algorithms. However, as the approach is deterministic, stochastic effects cannot easily be taken into account.

The basic algorithm for simulating reactions between chemical species on a mesoscopic level is the stochastic simulation algorithm (SSA) by Gillespie [10]. At this level, discrete numbers of particles are modeled. SSA is an exact algorithm, as it generates single trajectories of the underlying master equation. One key assumption is that the distribution of the species inside the volume is homogeneous. To simulate systems that do not adhere to this assumption, other approaches that introduce compartments and include the diffusion of species are necessary, e.g. [13].

A common way of introducing diffusion on mesoscopic level is the partition of space into sub-volumes and the extension of the master equation with a diffusion term, resulting in the reaction-diffusion master equation (RDME) [9]. The solution of the RDME is intractable for all but very simple systems, leading to the development of the Next-

Authorized licensed use limited to: McGill University. Downloaded on January 21, 2009 at 00:10 from IEEE Xplore. Restrictions apply.

Subvolume Method, an algorithm that, similar to SSA, generates trajectories of an underlying RDME.

2.2. NSM method

The Next-Subvolume Method is a discrete-event algorithm for simulating both reactions and the diffusion of species within a volume with an inhomogeneous distribution of particles. The volume is partitioned into cubical subvolumes each representing a well-stirred system. Events generated by the algorithm are either reactions inside or diffusions between these sub-volumes.

NSM uses a variant of Gillespie's Direct Method to calculate the first event times for all sub-volumes during initialization. Within the main loop, the sub-volume assigned to the smallest next event time is selected and the current event type according to the diffusion and reaction rates is determined. Finally, the event is executed and an update of the model state occurs. Note that the state update is performed only in a small region of the model volume, because either one sub-volume (in the case of a reaction) or two subvolumes (in the case of a diffusion) are affected, so that their propensities and next event times have to be updated.

Using an appropriate event queue structure such as a heap or binary tree, NSM has proven to be an efficient simulation algorithm even for larger systems consisting of many sub-volumes [7]. Given that a single run of the NSM algorithm generates only one possible realization of the underlying RDME, it is obvious that a number of replicated runs are necessary to gain insights into the modeled system in terms of mean values and standard deviations.

3. From Sequential to Parallel Execution

3.1. Computation and Communication Granularity in NSM

The Next-Subvolume Method can be applied to PDES by discretizing space into sub-volumes and assigning sets of sub-volumes to logical processes (LP). Interactions between sub-volumes are represented by messages exchanged between LPs involving diffusion events sent between neighboring sub-volumes.

To evaluate the parallel distributed execution of the NSM algorithm, tests with a sequential version of NSM from JAMES II [12] have been performed, which supports virtual work unit assignment. As a test model a 20 x 20 grid with an initial distribution of 1000 molecules of type A and B per sub-volume is chosen. The experiments focus on retrieving statistics about diffusion events between work units, so no reactions are defined. Both the number of sub-volumes assigned to each work unit and the stochastic diffusion constants are varied to evaluate the dependence between these parameters and communication overhead.

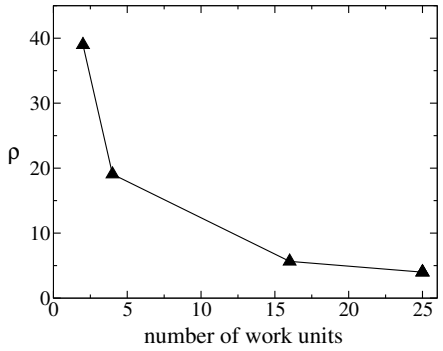

Figure 2. Dependence between ρ and the number of work units.

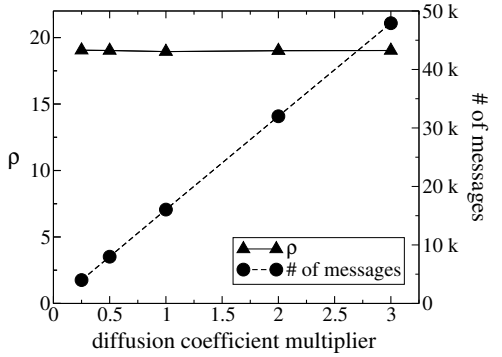

Figure 3. Dependence between ρ and the diffusion constant.

In figure 2, the change of the overall ratio

$$\rho = \frac{\#ev_{int}}{\#ev_{ext}}$$

between diffusion events exchanged between sub-volumes within the same work unit and diffusion messages sent between different work units is plotted as the number of work units was increased. The value of ρ is a measure for the computation granularity of the system, defined as the amount of computation that occurs between communications with another work unit. As the number of work units increases, the number of sub-volumes per work unit decreases, thus decreasing the computational load per work unit. Consequently, the overall concurrency of the system is reduced as the ratio of computation to communication is decreased. Figure 3 shows that granularity is unaffected by the diffusion activity of the species, although more messages are exchanged between work units as the diffusion constant is increased.

Parallel execution offers the potential to simulate larger models, to reduce the execution time of individual simulation runs, and to reduce the time required to complete multiple replicated runs. It is clear that large computation granularity is advantageous for maximizing concurrency in a parallel execution and potentially reducing the amount of time

Authorized licensed use limited to: McGill University. Downloaded on January 21, 2009 at 00:10 from IEEE Xplore. Restrictions apply.

required to complete a single run. These initial experiments indicate that the ratio of intra-work unit to inter-work unit communication decreases approximately in proportion with the number of work units.

Other observations concerning the NSM algorithm provide insight into the parallelization method that should be used. First, it is well known that the appropriate synchronization mechanism for parallel discrete event simulations depends on the amount of lookahead available for inter-work unit communications. In NSM, the inter-event times are sampled from an exponential distribution. This leads to a simulation with no lookahead, suggesting the use of optimistic synchronization. Alternate approaches such as pre-sampling the random number generator [18], artificially inserting lookahead into the computation, relaxing ordering constraint [20, 8, 2] or using alternate ordering rules [15] were also considered, but were rejected on the grounds that they would lead to unacceptable compromises concerning the accuracy of the simulation. Further examination of the algorithm revealed that the state saving requirements of the computation suggested the possibility of incremental state saving techniques due to successive states of the volume differing only in the number of particles in at most two sub-volumes. By only saving the state changes within each event, a rollback to a specific state can be accomplished by successively applying the inverse state change vector to the current state.

4. Towards optimistic execution in AURORA

4.1. Architecture Overview

AURORA is a scalable solution for delivering computational cycles for parallel discrete event simulation across desktop grid architectures. AURORA operates on the design of a strict master/worker paradigm where communication is allowed between the master services and the clients only. No peer-to-peer communication is assumed due to the potentially volatile uptimes of client machines. AURORA provides fault-tolerant and run replication support for both message-passing data-intensive PDES codes and task-parallel jobs.

The AURORA system is conceptually divided into three major parts: the back-end services, clients, and simulation packages as shown in figure 4. The back-end services consist of a broker along with any number of proxy, state and message servers for fault tolerance and scalability. Altogether, the back-end system provides the necessary support for correct PDES execution such as time management, persistent state vector storage, and collection of time-stamp ordered messages passed between logical processes. Computational cycles, whether idle or shared computing time, are provided by the clients. Clients register themselves with the back-end services which are then automatically managed

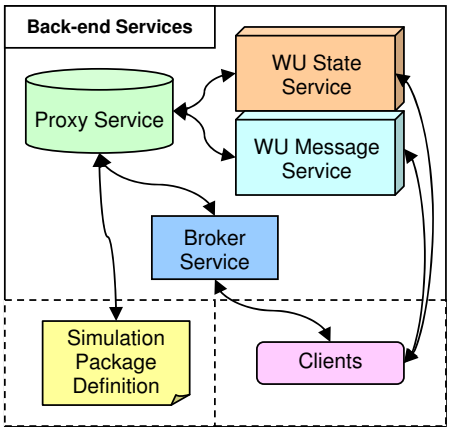

Figure 4. Aurora architecture overview

for work leases by the proxy service. The simulation package is a definition unique to the PDES execution specifying parameters such as the number of work units, connectivity between work units, simulation times, and other dependencies. Simulation packages are uploaded to the back-end services to allocate space and begin distribution of work to waiting clients.

4.2. New challenges for AURORA

The original AURORA system was designed to be conducive towards PDES codes that exhibited fairly large lookaheads with computationally intense work unit leases. Much like traditional PDES systems, a higher computation to communication ratio allows for potentially larger speedups. This property is amplified in the AURORA system due to messages being channeled through an intermediary. Therefore, AURORA was originally targeted towards simulations that performed reasonably well through conservative synchronization mechanisms.

The NSM model presents new challenges for the AURORA system due to the stochastic nature of diffusion messages; as mentioned earlier, lookaheads cannot be determined a priori. Thus, the parallel model can only be run using an optimistic time management approach, allowing clients to execute arbitrarily into the future, but providing mechanisms to rollback the simulation if messages are received in the past.

5. Distributed NSM with AURORA

5.1. AURORA and Optimism

To support simulations such as NSM, optimistic synchronization mechanisms were added to the AURORA system. Like traditional time warp systems, AURORA must provide means to rollback an execution if a message is delivered to the simulated past of a logical process. Here, in-

Authorized licensed use limited to: McGill University. Downloaded on January 21, 2009 at 00:10 from IEEE Xplore. Restrictions apply.

stead of using techniques such as anti-messages or direct cancellation, AURORA employs rollback detection and message cancellation entirely on the back-end system. Clients are notified of rollbacks through the proxy service. This design decision was deliberate, to reduce the potential large amount of network traffic that may be generated from anti-messages. When a client completes execution of a work unit and returns finalized data to the back-end system such as outgoing messages, the message server performs a virtual delivery of messages to the destination work unit input queues. Message delivery can cause a rollback if the message timestamp is less than the currently leased time window simulation end time. Messages sent after this time are canceled automatically by the message server and earliest rollback times are transmitted to the proxy. The proxy will then forward rollback notifications to any affected clients.

Certain issues must be addressed in conjunction with models such as NSM which exhibit fairly large state and fine event granularity. AURORA supports automatic tuning to the simulation as it performs computation, e.g. adaptively changing key parameters such as time window length and frequency of state saving. Both features are currently under development.

5.2. Self-Induced Rollbacks

There is a special case for rollbacks which must be considered in a master/worker PDES system. When a client finalizes a work unit, there is a possibility of rolling back its own execution. During the binning of messages from a work unit, a message may cause a rollback on another work unit which may cancel a message that was sent to the finalizing work unit which has been processed. This causes the returning work unit to perform a rollback on itself. If the client rolls back and returns messages that it already returned during the first finalization, there is a possibility of duplicate messages. The solution to this problem is to mask messages which have been already updated to the message server.

5.3. Zero Lookahead and Dynamic Time Window Adaptation

AURORA operates on the premise of leasing portions of work to clients. However, AURORA does not allow clients to execute arbitrarily into the future, thus an upper bound for execution must be determined at work unit lease. This problem is simple to solve when simulations exhibit some form of consistent lookahead such as network simulations which often translate link latencies for message transmission as lookahead when partitioning some subset of the entire network model onto different processors. By utilizing lookahead values, AURORA can determine a suitable end time for each work unit lease.

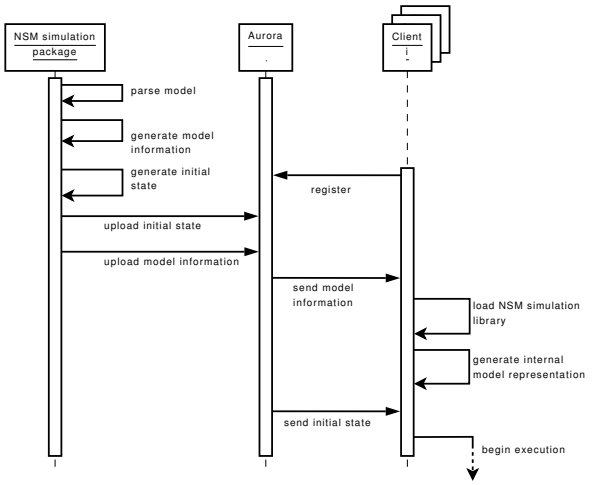

Figure 5. A simple sketch of the NSM integration in AURORA. The simulation package parses a model definition file and generates the initial state of the work units. Additionally, a file with general model information (species, reactions, connectivity graph) is created and sent, along with the initial state, to AURORA. Note that with this approach the model description file is only parsed once during the upload of the simulation package. The clients managed by AURORA can access the model information file and create an internal model representation. After the distribution of the work units, the clients load the library containing the NSM algorithm and begin execution.

Calculating a time window with zero lookahead poses a difficult problem. Although specifying a time window arbitrarily is easy based on the end time of the simulation, providing clients with a good time window to reduce the number of messages which may potentially cause rollbacks is non-trivial. A better approach is to heuristically determine time windows through collection of statistics over time. The AURORA proxy service keeps track of rollback histories for each work unit. These histories can be used to adaptively create time windows for work unit leases as the simulation progresses. AURORA approaches dynamic time window adaptation through two mechanisms.

The first implemented mechanism is to dynamically adapt time windows during the lease phase of the work unit. For zero lookahead simulations, the initial lease windows are based on the throttling parameter specified by the simulation package. After this initial lease window, the Aurora optimistic time management system performs dynamic window adaptation to tune the lease windows as the simulation progresses. Instead of using static time windows calculated from GVT and the simulation end time, the time management system takes into consideration the recent rollback histories, average past time window lease lengths, and standard deviations of rollbacks. Depending upon the throttling

Authorized licensed use limited to: McGill University. Downloaded on January 21, 2009 at 00:10 from IEEE Xplore. Restrictions apply.

aggressiveness chosen, the time management system will adaptively reduce the window if the standard deviation of the recent rollbacks exceeds a threshold value. Conversely, if there have been relatively no rollbacks from the histories, the system will adaptively grow the windows according to the throttling choice.

The second mechanism under development is to reactively change the length of the time window when a work unit completes and begins its finalization phase with the proxy. Instead of authorizing the work unit to immediately begin the update phase, the proxy will perform a rollback history lookup and calculate a rollback average for the returning work unit. If the work unit window exceeds recent time window lengths calculated from rollback averages, the proxy will send a prune message to both the client and message server that is hosting the client's messages. This prune message contains an earlier end time than what was leased to the client for that iteration. The client will prune any messages sent exceeding the new time window end time and restore the proper state for that time as well. The message server will adjust the current lease times for the work unit to prepare for message delivery. This active pruning mechanism can reduce the number of potential rollbacks if the proxy determines that the current optimism of a work unit exceeds that of the previous calculated time window.

5.4. Dynamic State Saving

In order to migrate state variables and messages across potentially heterogeneous machines, simulations for AURORA must provide application-dependent serialization and de-serialization routines. This mechanism is exploited for creating state histories as the work unit progresses forward in time for optimistically synchronized simulations. At every simulation time change, AURORA packs the simulation state into a buffer that can be later restored if a rollback is necessary.

The problem with this mechanism is that simulations which are fine-grained such as the NSM model, may create a large state history reducing the effective memory available for the simulation and introducing increased undesired overhead spent in non-simulation related computation. An alternative approach is to adaptively save state using forward computation histories.

The client can use the frequency of rollbacks and number of coast forwards as a means to calculate how much computation is performed for each time window lease. Using these statistics, the client can then compute offsets within the leased time window to filter state saves that may potentially be unnecessary. During forward computation, state saves can be infrequent until the average rollback time is reached. As the simulation time approaches the average rollback time, state is saved more frequently as the probability of a rollback is higher. Once simulation time has passed this point, the client can then begin filtering state saves again until the leased window end time is reached.

The combination of dynamic state saving and adaptive time windows can potentially reduce the amount of overhead and unnecessary rollbacks in the system, providing more processor time and memory for the simulation computation.

6. Results of First Experiments

6.1. Application Models

The model used for evaluating this approach is a simple reversible enzyme inhibitor model on a 48 x 48 grid. An inhibitor molecule *I* can bind to an enzyme *E* and block the active site for a substrate *S*, decreasing the activity of the enzyme-substrate reaction with intermediate product *ES* and, after catalysis, with final product *P*. The reactions defined for this model are:

$$E + S \xrightarrow{100.0} ES \quad (1)$$

$$E + I \xrightarrow{0.5} IE \quad (2)$$

$$ES \xrightarrow{1.0} E + P \quad (3)$$

$$ES \xrightarrow{50.0} E + S \quad (4)$$

$$IE \xrightarrow{0.01} E + I \quad (5)$$

Due to high reaction constants, reactions (1) and (4) happen frequently and produce a high work load on the clients. The enzyme and substrate molecules are distributed among all sub-volumes homogeneously with every volume containing 1000 *E* and 1000 *S* molecules. Within this simple model, these types of molecules cannot move from one sub-volume to a neighbor. Only the inhibitor *I* and the inhibitor-enzyme-complex *IE* are allowed to diffuse between sub-volumes. The diffusion constants for these species were chosen to be small, to limit the number of messages that get sent between work units. An amount of 300 *I* molecules were added to specific sub-volumes each and the system was simulated in the interval $[0, 0.2]$ with different work unit setups.

6.2. Results

Our expectations in combining AURORA and NSM have been to exploit the parallelism inherent in a spatial simulation to the advantage of performance and the ability to simulate large models. However, as our experiments showed, the interplay between a simulation approach like NSM and the grid-inspired distributed simulation approach turned out to be far more complex than expected. First experiments were done without an adaptive time window. Large static windows can lead to costly rollbacks. Due to the independent nature of execution and message delivery, client

Authorized licensed use limited to: McGill University. Downloaded on January 21, 2009 at 00:10 from IEEE Xplore. Restrictions apply.

simulation of time windows can still complete in the presence of a straggler message to the back-end services. This differs from other optimistic approaches where a straggler event interrupts current computation. Thus, rollbacks may incur overhead penalties in coast forward re-computation of events from the last saved state to the rollback time. A window size that is too small, however, may decrease performance by reducing the potential concurrency of the system and increasing the amount of time spent in state saving. Alternate techniques such as incremental state saving can potentially lead to reduced overhead times.

A compromise must be established between the frequency of state saving and the length of the time windows. As the simulation progresses, an adaptive algorithm for choosing the window size appears desirable. Although a dynamic time windowing system is in place, acceptable speed-ups could not be achieved. Due to the Markovian process inherent in these simulations, lookahead is impossible to define a priori. Using metrics such as rollback averages and deviations may help with models that exhibit some form of predictability, but in parallel NSM simulations assumptions of previous message generation and rollbacks cannot be carried between time window allocations. An alternative method is to percolate application-specific details into the time management system. Allowing the use of metrics, such as likelihood of diffusion events in comparison to the reaction events, could result in improved time window leases leading to reduced rollbacks.

Figure 6 shows a sample run depicting time window size and a dramatic overhead increase after the first 100 iterations. The high correlation between time window sizes and overhead shows a paramount issue with state management and straggler events causing rollbacks which in turn reduce the execution window sizes. Further optimization to time window size selection, an incremental approach to state saving, and earlier straggler message notifications could help restore balance between computation and overheads.

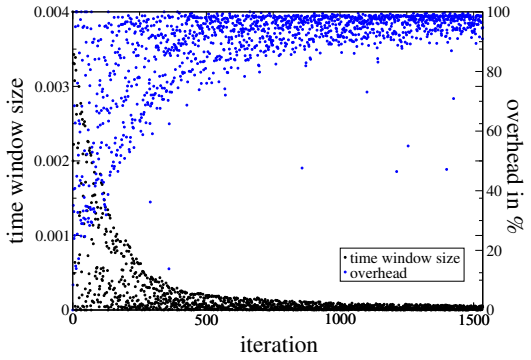

Figure 6. The size of the time window (black, bottom) and overhead (blue, top) is plotted for different iterations. The current adaptive algorithm reduces time window lengths in the presence of increasing messages and straggler events leading to more time spent in overhead such as state serialization.

7. Related work

The Gillespie Multi Particle Method (GMP) [24] is another approach for simulating reaction systems with an inhomogeneous distribution of particles using sub-volumes, but, in contrast to NSM, diffusion events take place at predetermined times, lifting the diffusion to the macroscopic level. During initialization, the first diffusion event time is calculated for each species. In each iteration, the event with the smallest time stamp is selected and Gillespie's Direct Method is used to simulate reactions between the last and the next event time. The execution of the next diffusion event is then performed locally in each sub-volume, distributing all entities of the species assigned to the event randomly among its neighbors. Therefore, in contrast to NSM, the GMP method abandons the idea of single entity diffusion events between two sub-volumes and performs bulk diffusions of species entities in all sub-volumes simultaneously.

Bernstein [3] applies the basic Gillespie algorithm to simulate a non-homogeneous system by introducing new reactions that correspond to diffusion events between sub-volumes. With this approach, diffusion coefficients that are inhomogeneous in space can be introduced. Although Bernstein uses Gillespie's Direct Method, other simulation algorithms to speed up computation (as, for example, Tau-leaping) can be applied as well. Despite these advantages, a parallel distributed execution of this method seems quite difficult, at least a partition of space is not possible.

GROMACS is a simulation system for parallel molecular dynamics simulation [16]. Many problems of simulating particle-based systems are also eminent in so-called n-body simulations, i.e., in simulations where all bodies are dependent of each other (e.g., because of gravity etc.). Although these systems are often used to simulate astrophysical phenomena, research results, like communication and partitioning schemes (e.g., [11]), could in principle be applied to molecular simulation as well.

8. Summary and Outlook

Biologists assume that localization has a central impact on cell biological dynamics. Therefore, spatial simulations of cell biological systems will gain increasing attention in the future. The combination of parallel, distributed execution and spatial simulation of cell biological systems promises one means for managing large scale realistic models. By realizing an optimistic version of NSM, a spatial simulation algorithm, in AURORA, two approaches were combined, namely optimistic execution and an approach for

Authorized licensed use limited to: McGill University. Downloaded on January 21, 2009 at 00:10 from IEEE Xplore. Restrictions apply.

assigning work units to processor in desktop grid computing environments. With our first experimental results problems have been identified. The NSM algorithm presents significant challenges because it contains zero lookahead events and relatively fine grained computation. These properties would slow down any parallel, distributed execution. This is particularly the case in combination with a grid-inspired approach, where notifications are delayed and work-units have to be packed, distributed and unpacked, overhead masked any potential speed-up. Different avenues for further research suggest themselves. In the current setting, one is to explore the effect of incremental state saving and a model-based dynamic time window adaptation heuristic. Another possibility is to evaluate the optimistic NSM without AURORA. As a third option, AURORA's strengths in conservative parallel execution could be exploited by realizing approximative approaches to spatial cell biological simulation, like GMP, in a grid-inspired setting.

9. Acknowledgments

This research is partly supported by the DFG (German Research Foundation) and the DAAD.

References

[1] S. S. Andrews and D. Bray. Stochastic simulation of chemical reactions with spatial resolution and single molecule detail. *Phys Biol*, 1(3-4):137–151, 2004.

[2] R. Beraldi and L. Nigro. Exploiting temporal uncertainty in time warp simulations. *Distributed Simulation and Real-Time Applications, 2000. (DS-RT 2000). Proceedings. Fourth IEEE International Workshop on*, pages 39–46, 2000.

[3] D. Bernstein. Simulating mesoscopic reaction-diffusion systems using the gillespie algorithm. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 71(4):041103, 2005.

[4] G. Broderick, M. Ru'aini, E. Chan, and M.J. Ellison. A life-like virtual cell membrane using discrete automata. *In Silico Biol.*, 16(5), 2004.

[5] G. Broderick and E. Rubin. The realistic modeling of biological systems: A workshop synopsis. *ComPlexUs Modelin in Systems Biology, Social Cognitive and Information Science*, 3(4):217–230, 2006.

[6] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *The Journal of Chemical Physics*, 121(9):4059–4067, 2004.

[7] J. Elf and M. Ehrenberg. Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. *Systems Biology (IEE)*, 1(2):230–236, 2004.

[8] R. M. Fujimoto. Exploiting temporal uncertainty in parallel and distributed simulations. In *PADS '99: Proceedings of the thirteenth workshop on Parallel and distributed simulation*, pages 46–53, Washington, DC, USA, 1999. IEEE Computer Society.

[9] C. W. Gardiner. *Handbook of Stochastic Methods: For Physics, Chemistry and the Natural Sciences (Springer Series in Synergetics)*. Springer, November 1996.

[10] D.T. Gillespie. Exact Stochastic Simulation of Coupled Chemical Reactions. *Journal of Physical Chemistry*, 81(25), 1977.

[11] A. Y. Grama, V. Kumar, and A. Sameh. Scalable parallel formulations of the barnes-hut method for n-body simulations. In *Supercomputing '94. Proceedings*, pages 439–448, 1994.

[12] J. Himmelspach and A.M. Uhrmacher. Plug'n simulate. In *Proceedings of the 40th Annual Simulation Symposium*, pages 137–143. IEEE, 2007.

[13] B.N. Kholodenko. Cell-signalling dynamics in time and space. *Nature Reviews Molecular Cell Biology*, 7(3):165–176, 2006.

[14] H. Kitano. Systems biology: a brief overview. *Science*, 295(5560):1662–1664, March 2002.

[15] B-S Lee, W Cai, and J Zhou. A causality based time management mechanism for federated simulation. *Parallel and Distributed Simulation, 2001. Proceedings. 15th Workship on*, pages 83–90, 2001.

[16] E. Lindahl, B. Hess, and D. van der Spoel. Gromacs 3.0: a package for molecular simulation and trajectory analysis. *Journal of Molecular Modeling*, 7(8):306–317, 2001.

[17] P.J. Love, M. Nekovee, P.V. Coveney, J. Chin, N. Gonzalez-Segredo, and J.M.R. Martin. Simulations of amphiphilic fluids using mesoscale lattice-boltzmann and lattice-gas methods. *Comput. Phys. Commun.*, 153(3):340–358, 2003.

[18] D. M. Nicol. Parallel discrete-event simulation of fcfs stochastic queueing networks. *SIGPLAN Not.*, 23(9):124–137, 1988.

Authorized licensed use limited to: McGill University. Downloaded on January 21, 2009 at 00:10 from IEEE Xplore. Restrictions apply.

[19] A. Park and R. Fujimoto. A scalable framework for parallel discrete event simulations on desktop grids. In *Grid Computing, 2007 8th IEEE/ACM International Conference on*, pages 185–192, 2007.

[20] D. Rao, N. Thondugulam, R. Radhakrishnan, and P. Wilsey. Unsynchronized parallel discrete event simulation. In *Proceedings of the 1998 Winter Simulation Conference*, 1998.

[21] D. Ridgway, G. Broderick, and MJ. Ellison. Accommodating space, time and randomness in network simulation. *Curr Opin Biotechnol*, 17:1–6, 2006.

[22] K. Takahashi, S. Nanda, V. Arjunan, and M. Tomita. Space in systems biology of signaling pathways : towards intracellular molecular crowding in silico. *FEBS letters*, 579(8):1783–1788, 2005.

[23] W.F. van Gunsteren and H.J. Berendsen. Computer simulation of molecular dynamics: Methodology, applications, and perspectives in chemistry. *Angewandte Chemie International Edition in English*, 29(9):992–1023, 1990.

[24] J. Vidal Rodriguez, J.A. Kaandorp, M. Dobrzynski, and J.G. Blom. Spatial stochastic modelling of the phosphoenolpyruvate-dependent phosphotransferase (PTS) pathway in Escherichia coli. *Bioinformatics*, page btl271, 2006.

Authorized licensed use limited to: McGill University. Downloaded on January 21, 2009 at 00:10 from IEEE Xplore. Restrictions apply.