UPPSALA
UNIVERSITET

IT Licentiate theses
2015-004

Parallelism and Efficiency in Discrete-Event Simulation

PAVOL BAUER

UPPSALA UNIVERSITY
Department of Information Technology

UPPSALA
UNIVERSITET

Parallelism and Efficiency in Discrete-Event Simulation

Pavol Bauer
pavol.bauer@it.uu.se

October 2015

Division of Scientific Computing
Department of Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

http://www.it.uu.se/

Dissertation for the degree of Licentiate of Technology in Scientific Computing

ISSN 1404-5117
Printed by the Department of Information Technology, Uppsala University, Sweden

Abstract

Discrete-event models depict systems where a discrete state is repeatedly altered by instantaneous changes in time, the *events* of the model. Such models have gained popularity in fields such as Computational Systems Biology or Computational Epidemiology due to the high modeling flexibility and the possibility to easily combine stochastic and deterministic dynamics. However, the system size of modern discrete-event models is growing and/or they need to be simulated at long time periods. Thus, efficient simulation algorithms are required, as well as the possibility to harness the compute potential of modern multicore computers. Due to the sequential design of simulators, parallelization of discrete event simulations is not trivial. This thesis discusses event-based modeling and sensitivity analysis and also examines ways to increase the efficiency of discrete-event simulations and to scale models involving deterministic and stochastic spatial dynamics on a large number of processor cores.

With love to my family,
Sanja and Lea.

List of Papers

This thesis is based on the following papers

I P. Bauer, S. Engblom. Sensitivity estimation and inverse problems in spatial stochastic models of chemical kinetics. In *Lecture Notes in Computational Science and Engineering*. Springer, 2015, 519-527

II P. Bauer, S. Engblom, S. Widgren. Fast event-based epidemiological simulations on national scales. *Submitted*. Preprint available under `http://arxiv.org/abs/1502.02908`.

III P. Bauer, J. Lindén, S. Engblom, B. Jonsson. Efficient Inter-Process Synchronization for Parallel Discrete Event Simulation on Multicores. In *ACM SIGSIM Principles of Advanced Discrete Simulation 2015*.

Contents

Chapter 1

Introduction

Discrete-event models depict a system as a sequence of discrete events evolving over time, each of them marking a change of the model state. In stochastic discrete-event models, the occurrence time of each event is of stochastic character, i.e. it is generated by a random variable or function. A typical example is the time series evolved by a continuous-time Markov Chain (CTMC) on a discrete state space. In the field of Computational Systems Biology, CTMCs are simulated with sampling methods such as the Gillespie Algorithm.

The main concern of this thesis is the efficiency of discrete-event simulation (DES). I discuss how to design efficient simulation algorithms as well as how to parallelize simulations of certain discrete-event models on modern shared-memory computers. In particular, I discuss two models where different parallelization techniques are required: the simulation of infectious disease spread on spatial networks, and the sampling of the Reaction and Diffusion Master Equation (RDME). In the first model, synchronization between parallel simulation processes is made at discrete steps of the model time that is specified in advance. For parallelization of such models, we discuss so-called *conservative* parallel DES methods. In the second model, synchronization between parallel simulation processes must be made at stochastic time steps, where the time increments are exponentially distributed and not bounded from below. For this class of models, we consider *optimistic* parallel DES techniques.

The thesis is structured as follows: In §2.1 I start with a brief introduction of DES and the application to the two specific models in §2.2. I continue in §2.3 by discussing parallel DES at deterministic and stochastic time steps. Finally, I briefly summarize the contributed papers in §3 and give a conclusion and outlook on future work in §4.

Chapter 2

Background

In the following chapter I will give a broad overview of the research areas this thesis covers. I start with a brief introduction of Discrete-Event simulation (DES) in §2.1, followed by an overview of two application areas; the simulation of systems governed by the reaction and diffusion master equation in §2.2.1 and a related framework for simulation of infectious disease spread on networks in §2.2.2. I conclude with the concrete topics of this thesis: a brief introduction to parallel Discrete-Event simulation (PDES) in §2.3, and with a discussion of PDES synchronization at deterministic or stochastic time steps in §2.3.1 and §2.3.2.

2.1 Discrete-Event Simulation

To discuss the area of DES, we first need to introduce the concept of a discrete-event system. According to Cassandras et al. [4], two characteristic properties describing a given system as a discrete-event system are;

1. The state space is a *discrete set*.

2. The state transition mechanisms are *event-driven*.

A system with a discrete state space is a system with a countable set of states. Typical examples of discrete state systems include finite-state automata and queues and more generally models of computers, communication or manufacturing processes.

The second property of a discrete-event system states that system transitions are driven by "events". Although a formal definition of an event is difficult to obtain, one can agree that an event is an instantaneous occurrence that changes the state of a system. In a system that is time-varying, such occurrences are assigned to points in time.

A concrete example of a discrete-event system is a *random walk* of a particle on a discrete plane in two dimensions. The state of the system can be described by the position of the particle $\mathbb{X} = \{(x_1, x_2) : x_{1,2} \in \mathbb{Z}\}$ and a natural set of events can be given by $E = \{N, W, S, E\}$, corresponding to the action of "taking a step to the north, west, south or east". Then, a possible sequence of events in the system starting at an initial state $(0, 0)$ at time $t = t_0$ can be $[W, W, N, N, S, E]$, occuring at some event times $[t_1, t_2, \ldots, t_6]$. Note that such a sequence of events can be determined stochastically but may also be defined by deterministic functions or logical rules.

No uniquely defined methods or algorithms to simulate a DES exist, but one can agree on certain components which are contained in a typical discrete-event simulator;

1. *State:* a data-structure where the complete model state is stored

2. *Clock:* a variable where the current simulation time is stored

3. *Scheduled event list:* a data-structure where all scheduled events are stored in combination with their future occurrence time

4. *Initialization routine:* a routine which initializes all data structures (elements 1-3) at the beginning of the simulation run

5. *Time update routine:* a routine which identifies the next event to occur and advances the current simulation time to the occurrence of that event

6. *State update routine:* a routine which updates the state based on the event to occur

A typical simulator run consists of an initial call to the *initialization routine* which sets the simulation time $t = 0$. Then, the simulator calls the *time update routine* to obtain the next event and its occurrence time Δt from the *scheduled event list* and applies the event transition to the state using the *state update routine*. Next, the current system time is set to $t = t + \Delta t$. Afterwards the simulation continues with iterative execution of both of the before-mentioned routines until a stopping criterion such as $t > T_{end}$ is fulfilled and the simulation terminates.

As it is straightforward to introduce randomness in the time or state update routines of the algorithm, DES-algorithms can be easily adapted to Monte-Carlo samplers. Next, I will dicsuss how DES-algorithms can be used in different applications.

2.2 Applications

In this section I will discuss two specific applications of DES. The first application is the numerical simulation of trajectories governed by the reaction-diffusion master equation (RDME) which is an important model in the field of computational Systems Biology. The second application is a framework for modeling and simulation of infectious disease spread on networks that are created from epidemiological data. In both cases, DES is needed to generate trajectories of a continuous-time discrete-space Markov chain or to incorporate discrete state changes that are given by data.

2.2.1 Sampling the reaction-diffusion master equation

In order to discuss the RDME we need to introduce the chemical master equation (CME) first. The CME [16, 22] describes reactions kinetics of molecular species at the mesoscopic level. On this scale, the system is described by the discrete vector $\mathbb{X} = \mathbb{X}(t)$, where each entry is the copy number of a chemical species $j = 1 \dots D$. This species can take part in $r = 1 \dots R$ different reactions, which are defined with a stoichiometry matrix $\mathbb{N} \in \mathbf{Z}_+^{D \times R}$ and a set of propensity functions $\omega_r(x), r = 1 \dots R$. The transition between states caused by a reaction can be written as

$$\mathbb{X} \xrightarrow{w_r(\mathbb{X})} \mathbb{X} - \mathbb{N}_r. \tag{2.1}$$

The state is thus described until the next reaction happens, in all a continuous-time Markov Chain. As a consequence, the reaction time τ_r is an exponential random variable of the rate $1/\omega_r(\mathbb{X})$. It is possible to explicitly evolve the probability density function of such a system using the forward Kolmogorov equation or CME, which is given by

$$\frac{\partial p(\mathbb{X}, t)}{\partial t} = \sum_{r=1}^{R} w_r(\mathbb{X} + \mathbb{N}_r) p(\mathbb{X} + \mathbb{N}_r, t) - \sum_{r=1}^{R} w_r(\mathbb{X}) p(\mathbb{X}, t) \tag{2.2}$$

$$=: \mathcal{M} p, \tag{2.3}$$

where $p(\mathbb{X}, t) := P(\mathbb{X} = \mathbb{X}(t) | \mathbb{X}(0))$ for brevity.

Equation (2.2) can be solved analytically for simple models involving a small number of species. However, for a realistic set of species and reactions the *curse of dimensionality* prohibits an analytical solution and thus the study of such systems relies on approximations or sampling methods.

One such sampling method is the Gillespie's direct method [18], commonly known as the stochastic simulation algorithm (SSA) (Algorithm 1).

Algorithm 1 Gillespie's direct method (SSA)

1: Let $t = 0$ and set the state $\mathbb{X}$ to the initial number of molecules.
2: Compute the total reaction intensity $\lambda = \sum_r w_r(\mathbb{X})$. Generate the *time to the next reaction* $\tau = -\lambda^{-1} \log u_1$ where $u_1 \in (0, 1)$ is a uniform random number. Determine also the next reaction r by the requirement that

$$\sum_{s=1}^{r-1} w_s(\mathbb{X}) < \lambda u_2 \leq \sum_{s=1}^{r} w_s(\mathbb{X}),$$

where u_2 is again a uniform random deviate in $(0, 1)$.
3: Update the state of the system by setting $t = t + \tau$ and $\mathbb{X} = \mathbb{X} - \mathbb{N}_r$.
4: Repeat from step 1 until some final time T is reached.

The algorithm uses inverse transform sampling in order to generate exponential random variates and to determine the time τ until the next reaction fires.

Note that the algorithm generates a single realization or trajectory of the given stochastic system, but that the histogram of many such realizations converges to equation (2.2). It should also be clear, that this algorithm has a similar structure to the typical DES loop presented in §2.1.

A way of representing one realization of the probability density given by (2.2) is using the *random time change representation* introduced by Kurtz [13]. This representation describes the state $\mathbb{X}_t$ as a sum of R independent unit-rate Poisson processes Π_r, since $t = 0$. The representation is given by

$$\mathbb{X}_t = \mathbb{X}_0 - \sum_{r=1}^{R} \mathbb{N}_r \Pi_r \left(\int_0^t w_r(\mathbb{X}_{t-})\, dt \right), \tag{2.4}$$

where $\mathbb{X}_0$ is the initial state, and where $\mathbb{X}_{t-}$ denotes the state before any transitions at time t.

As shown in [11], an alternative construct to (2.4) is the random counting measure $\mu_r(dt) = \mu_r(w_r(\mathbb{X}_{t-}); dt)$. The measure is associated with the Poisson process for the rth reaction at rate $w_r(\mathbb{X}_{t-})$ for any time t. Writing the counting measures for each reaction as a vector $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_R]^T$ one can represent (2.4) as

$$d\mathbb{X}_t = -\mathbb{N}\boldsymbol{\mu}(dt), \tag{2.5}$$

which is a stochastic differential equation (SDE) with jumps.

The chemical kinetics discussed so far was assumed to be spatially homogeneous, which means that all molecules in the system are "well-stirred"

(uniformly distributed) in space. Clearly, this assumption can be restrictive if more complex structures, such as for example biological cells, are studied. Therefore it is meaningful to extend the mesoscopic description to the spatially heterogeneous case [27].

Thus, to introduce diffusion, one can divide the domain V into K non-overlapping voxels $\zeta_1 \dots \zeta_K$, which can be ordered in a structured [9] or unstructured grid [12]. Then, the molecules are assumed to be well-stirred inside each single voxel ζ_n. As a consequence, the state space $\mathbb{X}$ will now be of size $D \times K$, and the new type of state transition occurring due to the diffusion of species i from voxel V_k to another voxel V_j is

$$\mathbb{X}_{ik} \xrightarrow{q_{kj}\mathbb{X}_{ik}} \mathbb{X}_{ij}, \tag{2.6}$$

where q_{kj} is a diffusion rate.

The diffusion master equation can be now written as

$$\begin{aligned}\frac{\partial p(\mathbb{X},t)}{\partial t} = &\sum_{i=1}^{D}\sum_{k=1}^{K}\sum_{j=1}^{K} q_{kj}(\mathbb{X}_{ik}+\mathbb{M}_{kj,k})p(\mathbb{X}_{1\cdot},\dots,\mathbb{X}_{i\cdot}+\mathbb{M}_{kj},\dots,\mathbb{X}_{D\cdot},t)\\ &-q_{kj}\mathbb{X}_{ik}p(\mathbb{X},t) =: \mathcal{D}p(\mathbb{X},t),\end{aligned} \tag{2.7}$$

where $\mathbb{M}_{kj}$ is a transition vector that is zero except for $\mathbb{M}_{kj,k} = -\mathbb{M}_{kj,j} = 1$.

For a system with reactions and diffusions, one can combine (2.2) and (2.7) and write the reaction-diffusion master equation

$$\frac{\partial p(\mathbb{X},t)}{\partial t} = (\mathcal{M}+\mathcal{D})p(\mathbb{X},t). \tag{2.8}$$

To simulate trajectories from (2.8), we can again apply the previously introduced Gillespie's Direct Method, although it is not a very efficient method if the model contains a larger number of cells K. Thus, further developed methods have been introduced, which improve the simulation efficiency with the implementation of a priority queue H and a hierarchical grouping of events [17]. Another factor for improving the simulation efficiency is the usage of a dependency graph G, which marks the rates that have to be recomputed at the occurence of a given event. This prevents the unnecessary evaluation of non-dependent events. The structure of one commonly used algorithm of this type, the Next Sub-volume Method (NSM) [9], is shown in Algorithm 2.

Further methods to simulate spatial models are spatial Tau-Leaping [19], Gillespie Multi-particle Method [29], and Diffusive Finite State Projection [8]. Note, that although some methods may be more efficient than the NSM, they take different assumptions that can influence the computational results. For example, solutions computed by the Gillespie Multi-particle Method have been reported to violate statistical properties, a consequence of the deterministic processing of diffusion events in the method [21].

Algorithm 2 The next subvolume method (NSM)

1: Let $t = 0$ and set the state $\mathbb{X}$ to the initial number of molecules. Generate the dependency graph G. Initialize priority queue H. For all $j = 1 \dots K$ voxels, compute the sum λ_j^r for all reaction propensities $\omega_r(\mathbb{X}_j)$ and the sum λ_j^d for all diffusion rates.
2: Generate the initial waiting times $\tau_j \simeq Exp(1/(\lambda_j^r + \lambda_j^d)$. Store the values in the priority queue H as ordered key-value pairs.
3: Remove the smallest time τ_j from H. Generate pseudo-random variable u_1.
4: if $u_1(\lambda_j^r + \lambda_j^d) < \lambda_j^r$ then a reaction occured in voxel j. Find out which one it was as in the Gillespie's Direct Method (Algorithm 1).
5: if $u_1(\lambda_j^r + \lambda_j^d) \geq \lambda_j^r$ then a molecule diffused from voxel j. Sample uniform random numbers to find out which species diffused to which voxel.
6: Update the state of the system by setting $t = t + \tau$ and $\mathbb{X} = \mathbb{X} - \mathbb{N}_r$.
7: Draw a new exponential random number τ_j for the currently occurred event.
8: Update all rates marked as dependent to the current event in G, and recompute the next waiting time as $\tau_j^{\text{new}} = t + \left(\tau_j^{\text{old}} - t\right) \frac{(\lambda_j^r + \lambda_j^d)^{old}}{(\lambda_j^r + \lambda_j^d)^{new}}$.
9: Update H and repeat from step 3 until the final time T is reached.

2.2.2 Modeling of infectious disease-spread on networks

Another application area of DES is in modeling and simulation of infectious disease spread on spatial networks. Here, the state $\mathbb{X} \in \mathbf{Z}_+^{D \times K}$ represents the count of some individuals contained in a compartment $c = 1 \dots D$ at some discrete node $i = 1 \dots K$. As an example, individuals could be grouped according to their health state into a susceptible, infected, or recovered group, as in the commonly used SIR-model [23]. The node index i represents some discrete location at which no finer spatial information of the individuals exist, or where it is not meaningful to consider one. Individuals are regarded as uniformly distributed at every node, similarly as in the spatial RDME setting discussed in §2.2.1.

The transitions between compartments are stochastic and described by the transition matrix $\mathbb{S} \in \mathbf{Z}^{D \times R}$ as well as the transition intensity $R : \mathbf{Z}_+^D \to \mathbf{R}_+^R$, assuming R different transitions.

Using the SDE representation of a Markov process from (2.5) we can define the change in the state $\mathbb{X}$ of all individuals contained in the ith node as

$$d\mathbb{X}_t^{(i)} = \mathbb{S}^{(i)} \boldsymbol{\mu}(dt), \tag{2.9}$$

where $\boldsymbol{\mu}(dt) = [\mu_1(dt), \ldots, \mu_R(dt)]^T$ is a vector of random counting measures for all R transitions, $\mu_k(dt) = \mu(R_k(X(t-)); dt)$.

The model can be further extended with interactions over a network, where each node is a vertex of an undirected graph $\mathcal{G}$. Then, each node i may affect the state of the nodes in the connected components $C(i)$ of i, and may also be affected by other nodes j, where $i \in C(j)$. This may for example model a transfer or movement process of individuals between the nodes. If each such connection is described by the counting measures $\boldsymbol{\nu}^{(i,j)}$ and $\boldsymbol{\nu}^{(j,i)}$, the overall network dynamics is given by

$$d\mathbb{X}_t^{(i)} = -\sum_{j \in C(i)} \mathbb{C}\boldsymbol{\nu}^{(i,j)}(dt) + \sum_{j;\, i \in C(j)} \mathbb{C}\boldsymbol{\nu}^{(j,i)}(dt). \tag{2.10}$$

Combining (2.9) and (2.10) the overall dynamics of the framework is

$$d\mathbb{X}_t^{(i)} = \mathbb{S}\boldsymbol{\mu}^{(i)}(dt) - \sum_{j \in C(i)} \mathbb{C}\boldsymbol{\nu}^{(i,j)}(dt) + \sum_{j;\, i \in C(j)} \mathbb{C}\boldsymbol{\nu}^{(j,i)}(dt). \tag{2.11}$$

Equation (2.11) can be further extended with other terms, one may for example add additional discrete or continuous state variables that are needed in a particular model. Furthermore, as it is discussed in Paper II, it is possible to extend the model with deterministic dynamics that can be combined with (2.11).

2.3 Parallel Discrete Event Simulation

Parallel discrete-event simulation (PDES) is a collection of techniques used to simulate discrete-event models on parallel computers. In general, the goal is to divide an entire simulation run into a set of smaller sub-tasks that are executed concurrently. As discussed by Liu [24], the simulation can be decomposed to parallel work on several levels;

- *Replicated Trials:* The simplest form of PDES, independently processing multiple instances of a sequential simulation on parallel processors. An example from Computational Systems Biology is the generation of multiple trajectories of a stochastic model. Such computations can be run independently in parallel, for example on multiple nodes in a cloud infrastructure [1].

- *Functional decomposition:* Different functions of the sequential simulator, such as the random number generation [26] or the state update routine are processed in parallel by separate processors, but the main simulation loop is executed in a serial fashion.

- *Time-parallel decomposition:* A division of the time axis into smaller time intervals, where each of them is simulated in parallel and then re-assembled at synchronization steps. In the context of Computational Systems Biology, an example of such a decomposition is given in [10].
- *Space-parallel decomposition* The spatial domain is divided into non-overlapping sub-domains. Each sub-domain is assigned to a processor. Events affecting several sub-domains have to be communicated between processors. This is the commonly used decomposition in PDES. In this thesis I will focus exclusively on this approach.

When space-parallel decomposition is used, the simulation task is distributed onto a group of so-called *logical processes* (LPs). Each LP is mapped to a processor core or virtual thread, where it runs a self-contained discrete-event simulator with its own list of scheduled events and an own simulation clock. Typically, each LP has also it's own local state (the state of the sub-domain) that is not shared with other LPs [15].

Hence, LPs are required to communicate with each other in order to synchronize events that affect the state of two or more sub-domains (residing on two or more LPs). In order to do this, the LPs exchange so-called *time-stamped messages*. A message contains the specification of the event and the event occurrence time. The LP receiving the message enters the event into its *input queue* (an event list dedicated for received events) and processes it interleaved with the locally scheduled events.

In general, the design and implementation of PDES must be sensitive to the computing environment. In the distributed environment, messages are usually communicated via a network protocol, whereas on multi-cores they can be written to variables in the shared memory. One implication is that the issue of transient messages (that are sent by one LP, but not yet received by the other LP) does not have to be handled in this case. Furthermore, PDES operated in a cloud environment have to additionally compensate for the unbalanced workload distribution and communication delays due to the virtualization layer between the LP and the underlying hardware [25].

A significant challenge in PDES is to reproduce exactly the same end state as in sequential DES. This is a non-trivial task, as messages could arrive *out-of-order* due to the asynchronous processing of events on LPs. As the event contained in the message should have been executed at an earlier local simulation time the current state has to be consequently invalidated. This violation is called a *causality error*, and it is defined as the violation of the Local Causality Constraint (LCC), a term coined by Fujimoto [15]:

A discrete-event simulation, consisting of LPs that interact exclusively by exchanging messages obeys the local causality constraint if and only if each LP processes events in non-decreasing timestamp order.

To satisfy the Local Causality Constraint, different PDES synchronization methods have been proposed which can be generally categorized into two major classes,

- *Conservative synchronization,* where causality errors are strictly avoided by blocking every local or global execution which could lead to such a violation.

- *Optimistic synchronization,* where causality errors are initially allowed, but the system implements some sort of recovery which restores the valid state once the error is detected.

It is difficult to state which method is the best choice for a computational problem in general. While a simulation relying on optimistic synchronization may seem to achieve a high processor utilization, the overhead caused by the state recovery, or roll-backs, may be vast. Conservative synchronization on the other hand may create a significant amount of blocking time while deciding if an event can be processed safely without inducing causality errors on neighboring domains. As we will discuss later, one may also consider a class of *hybrid-methods* which make use of a combination of both approaches.

Next, we will discuss which type of PDES is more suitable for two specific types of simulations. From the perspective of scientific computing, the key issue is whether the simulation is carried out at deterministic or stochastic time steps.

2.3.1 PDES at deterministic time steps

In this section I will briefly review PDES at deterministic time steps. In this type of simulation, knowledge exists about future events and their exact occurrence times. An LP can use this knowledge to synchronize with dependent neighbors when needed. As reviewed by Jaffer et al. [20], the literature distinguishes between two types of approaches for simulation of such models: *synchronous* or *asynchronous simulation.*

In *synchronous simulation*, the local simulation time is identical on each LP and evolves on a sequence of time steps $(0, \Delta t, 2\Delta t, \ldots, i\Delta t)$. This approach is suitable for models where events occur exactly at the ith time step, or models where several continuous time updates occur in the time interval $[i\Delta t, (i+1)\Delta t)$, but only the final state requires synchronization with other LPs. Hence, if such a method is used for solutions of the RDME (as in [8, 2]), it implies a numerical error that occurs due to the disregarded synchronization of the continuous time state updates.

The implementation of a synchronous simulation engine is rather straightforward. Typically, a single LP evolves the local time until a global barrier

and broadcasts the new state to the neighboring LPs. As the computations at each time step are independent of each other, massively parallel architectures as GPUs can be targeted for efficient implementation, see for example [28] in the context of the RDME.

In *asynchronous simulation*, the local simulation time differs on each LP and the time propagation is event-driven; Events are lined up in input queues and processed in non-decreasing timestamp order. The processing of each event increases the local time on the LP. If the event generates a new event on an other LP, the event is being sent as a message and enqueued in the destination's input queue. As some knowledge about future events exists, conservative simulation algorithms such as the Null-message protocol [5, 3] (Algorithm 3), can be used to process events safely, without validating the LCC.

Algorithm 3 The Null-message protocol

1: Initialize N input queues for events received from N neighboring LPs
2: **while** $t < T_{end}$ **do**
3: **if** Some queue is empty **then**
4: Propagate t to the time contained in the Null-Message for the empty queue.
5: **end if**
6: Remove the event with the smallest time from all input queues.
7: Process that event and increment t. If the event generates another event on a different LP, send a message.
8: Communicate a Null-message containing the lower bound of future event times to all neighboring LPs.
9: **end while**

The lower bound of future event times, also termed *lookahead*, is the earliest time when the LP communicates new events to a given neighbor. The lookahead must exist in order to ensure the progress of the simulation, and needs to be communicated via Null-messages.

Given an input queue is empty, the LP can not continue with the processing of other messages, as causality errors may occur due to straggling messages ariving in the empty queue at a later simulation time. When the Null-message for the empty queue is available, the LP can propagate the local time to the lookahead time contained in the Null-message and safely process other messages whose time is smaller than the new local time.

In general, asynchronous simulation can also be seen as a scheduling problem. As shown by Xiao et al. [32], a parallel scheduler can be used to distribute the processing of events onto several parallel processors while maintaining the LCC centrally. In such cases, messages do not have to be

sent explicitely but can, for example, be stored in a memory location shared between all parallel processors.

2.3.2 PDES at stochastic time steps

From the perspective of computer science, the design and implementation of PDES at stochastic time steps is a much more challenging task than the implementation of PDES at deterministic time steps. Nonetheless, this approach is required if the time stepping is either given by a stochastic function or by a deterministic function that is not accurately predictable at previous time steps (e.g. chaotic or given by a complex set of rules), and thus no lower bound of future event times is available.

As discussed in §2.3.1, if no such lookahead is available, PDES using conservative methods should be avoided. As noted by Dematté and Mazza [7], this is clearly the case for the exact numerical simulation of RDME models, where the time increments between events are exponentially distributed and hence unbounded from below.

Thus, *optimistic simulation* needs to be applied to this class of problems, where future events are executed speculatively, and causality errors are resolved using roll-backs. As shown by Wang et. al [31], optimistic simulation of spatial stochastic systems governed by the RDME is scalable. On the other hand such approaches are prone to be "over-optimistic", in the sense that an overly large number of local events are processed speculatively. This may hinder the efficiency of the parallel simulation due to two main reasons;

- When a causality error occurs, the local state must be roll-backed to the timestamp of the message that caused the error. Clearly, if the amount of speculation is beyond some limit, the amount of roll-backed events increases. Hence, the larger amount of unnecessary forward computations and roll-backs will impose a greater overhead on the processing LP.

- The speculative processing of local events may generate events that affect the state of neighboring LPs and therefore have to be communicated via messages. If a causality error is detected, the changes in the neighbors state caused by the messages sent during the speculative period thus have to be reverted, too. Moreover, the events received at the neighboring LP could have generated new events that were communicated to other, remote LPs. Hence, the state of the remote LPs has to be included in the roll-back as well. It should be clear that this can lead to a long cascade of roll-backs spreading over several LPs, which are costly to resolve.

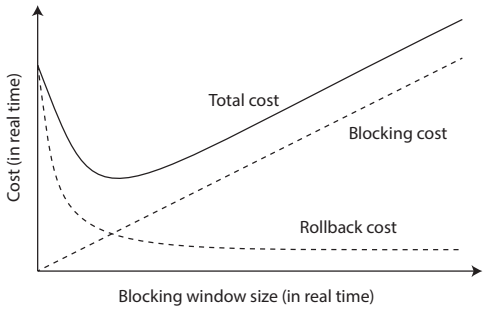

Figure 2.1: The impact of a static blocking window size on the simulation cost of an optimistic simulator (freely adapted from [6]).

Therefore, *adaptive protocols*, which aim to limit the optimistic processing of the simulator, can be beneficial for the performance. In general, limitations can be imposed on all simulating LPs, or rather on LPs that have a stronger contribution to over-optimism, e.g. by processing events faster then their neighbors. For example, the limitations can be implemented by a reduction of the event processing rate due to a propagated *blocking window*, during which the local simulation is suspended for a certain amount of real time $[rt(\mathbf{t})\ rt(\mathbf{t}) + s]$, where $rt()$ is the real time as a function of the simulation time, $\mathbf{t}$ is the local simulation time and s is the size of the blocking window.

As shown in Figure 2.1, it is a challenge to set a window size that has the optimal impact on the simulation performance. Assuming the window is of fixed size (defined in real time units, e.g. micro-seconds) it is optimal when both, the cost of roll-backs and the overhead due to the additional blocking time are minimized. In addition, the blocking time should be chosen for each LP in a way, so that all LPs propagate at the same average event rate throughout the simulation. Under this condition, the simulation time is expected to be lower, as causality errors due to straggling messages are less probable to occur.

Since it is difficult to define an optimal static window size, different methods to set an adaptive blocking window have been proposed, see for example [6] for a review by Das. Typically, the adaptivity is based on a measure of potential over-optimism that is estiamted during the simulation run. One possibility is to compute the measure globally, e.g. by monitoring the behaviour of simulating LPs and inferring on the individual progress.

Another example is given by the Elastic Time protocol [30], where the measure is computed locally based on the simulation time of an LP and the simulation time of its neighbors. Assuming transient messages do not exist,

the measure of over-optimism M computes as

$$M = max(\mathbf{t} - \alpha, 0), \tag{2.12}$$

where $\mathbf{t}$ is the local simulation time of an LP and α is the minimum local simulation time of its neighbors. The measure M is computed at each iteration of the simulation loop by every LP and translated into applicable blocking time. Evidently, the computation of α requires ongoing exchange of time information between LPs.

A different approach to adaptive PDES was proposed by Ferscha with the Probabilistic Adaptive Direct Optimism Control (PADOC) [14]. In a nutshell, the algorithm applies to an LP that is subject to the incoming messages m_i from its neighbors at simulation time $t(m_{(-2)}), t(m_{(-1)}), \mathbf{t}, t(m_1), t(m_2)$, where $\mathbf{t}$ is the local simulation time on the LP. Then, the method first computes an estimate of the future event arrival time $\hat{t} = t(m_{(-1)}) + \hat{\Delta}$, where $t(m_{(-1)})$ is the arrival time of the last message and $\hat{\Delta}$ the estimated inter-event time. With this, the LP computes the probability to block the local simulation at the current simulation time $\mathbf{t}$ as

$$P(\text{block}|\mathbf{t}) = 1 + e^{\frac{-\zeta(\mathbf{t}-\hat{t})}{\hat{t}}}, \tag{2.13}$$

where ζ is the scaling of a confidence interval in the range [0 1].

The inter-arrival time estimator can be computed via different statistical methods, as for example by

$$\hat{\Delta} = \frac{1}{n}\sum_{j=1}^{n} t(m_j + 1) - t(m_j), \tag{2.14}$$

and iteratively propagated as a moving observation window.

In summary, the main goal of both algorithms is to adapt the progress of LPs to the same average rate. PADOC compares to the Elastic Time protocol in that it uses statistical estimation of arrived messages in order to infer on an LPs progress rate in relation to its neighbors. This can be unsatisfactory if the event dynamics change frequently throughout the simulation, and the prediction of future arrival times fails due to a large statistical error. In contract, the Elastic Time protocol is based on the observation of the actual model state, but it requires additional information to be communication between LPs, which introduces further simulation overhead.

Chapter 3

Summary of papers

3.1 Paper I

This paper addresses parameter sensitivity estimation in spatial stochastic models of chemical kinetics. It proposes a new algorithm which can propagate perturbations effectively through simulations. Furthermore, the algorithm can be used to solve problems of inverse character, for example in combination with numerical optimization.

In the stochastic setting, the goal of sensitivity estimation is to characterize the mean effect of a function of interest due to some perturbation $c \to c + \delta$. One possibility would be to determine

$$E[f(\mathbb{X}(t, c + \delta))] - E[f(\mathbb{X}(t, c))], \tag{3.1}$$

for example, by computing a sample average. In this case c could be a reaction rate constant and f a function of the species copy number $\mathbb{X}_t$.

If one solves a parameter estimation task using Monte-Carlo methods, such as the Gillespie's method (Algorithm 1), a trivial approach would be to generate N independent trajectories each of $f(\mathbb{X}(t, c + \delta))$ and $f(\mathbb{X}(t, c))$ using independent random numbers and then observing the difference of their average. This approach can lead to unsatisfactory results as the variance obtained in both simulations can be large in comparison to the average difference $f(\mathbb{X}(t, c + \delta)) - f(\mathbb{X}(t, c))$.

A solution of the variance reduction problem is given by a strong coupling of the two processes $\mathbb{X}(t, c + \delta)$ and $\mathbb{X}(t, c)$. As discussed in this paper, the previously introduced Next Subvolume Method (Algorithm 2) can not guarantee such consistency, as the coupling between $\mathbb{X}(t, c+\delta)$ and $\mathbb{X}(t, c)$ is simply not the intended one. This fact was the inspiration for the creation of the All Events Method (AEM), which is demonstrated to achieve a better coupling between processes and therefore reduces the variance significantly compared to the NSM.

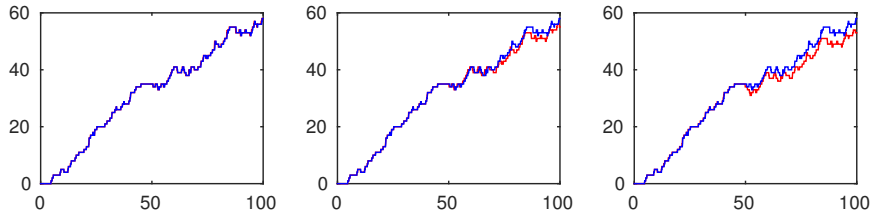

Figure 3.1: AEM solution of an unperturbed (blue) and increasingly perturbed (red, from left $\delta = [0.01, 0.5, 0.1]$) trajectory.

As an example, take the perturbation of a simple birth-death process of a species A,

$$\emptyset \xrightarrow{1} A, \qquad A \xrightarrow{A(1-\delta)} \emptyset. \tag{3.2}$$

The difference in perturbed and unperturbed trajectories computed with the AEM is shown in Figure 3.1.

We evaluated the performance of the AEM and NSM in terms of variance reduction. For this purpose, we constructed a spatial model with strongly nonlinear reaction dynamics and perturbed a sensitive parameter. We concluded that using the AEM, significantly less trajectories are required in order to minimize the error in (3.1) at a given tolerance level.

We furthermore used the method in combination with a numerical optimization routine to solve for an inverse formulation. In particular, we found the optimal amount of a reactant required to obtain a certain quantity of a product in a chemical reaction network.

3.2 Paper II

This paper contains two contributions. The first contribution is a mathematical framework for modeling of infectious disease spread on spatial networks. The framework is briefly reviewed in §2.2.2 of this thesis. The second contribution is a sequential and parallel DES algorithm for models formulated within the framework. We show how the parallel algorithm can be used to solve a computationally intensive problem: the fitting of model parameters to given reference data.

Because the spatial dynamics of the model is assumed to be deterministic and given by data, the PDES algorithm makes use of synchronization approaches as discussed in §2.3.1 of this thesis. In particular, we view the parallelization as a scheduling problem and make use of a dependency-aware task-computing library to evolve events concurrently while maintaining the causality of the model.

The parallel simulation algorithm using the `SuperGlue` task-library is demonstrated to efficiently scale real-world problems on multi-socket shared-memory computers with up to 32 computing cores. The performance of the task-based approach is significantly better than traditional parallelization using `OpenMP`, provided that the average task size does not drop under a critical limit.

3.3 Paper III

This paper addresses the parallel numerical solution of spatial stochastic systems governed by the RDME by sampling statistically equivalent trajectories from a continuous-time Markov chain, as introduced in §2.2.1 of this thesis. Paper III comes to Paper I in that the parallel algorithm is based on the All Events Method (AEM), the numerical algorithm presented and evaluated in Paper I.

As discussed in §2.3.2, the parallel simulation of spatial stochastic systems requires synchronization at stochastic time steps of the model time. For this purpose, we propose a hybrid synchronization protocol for simulation on shared-memory multicores, which uses a mixture of optimistic and conservative PDES techniques.

The reason why the algorithm is based on the AEM is because it enables the extraction of future diffusion event times stored in an LPs event list. The times are communicated as a "probabilistic lookahead' to neighboring LPs, where the estimate is used to control the optimism by introduction of an adaptive blocking window. This process is demonstrated to significantly reduce the probability of roll-backs and thus to improve the overall efficiency.

Further contributions include a *selective roll-back* function, which additionally increases the parallel performance by lowering the overall cost of roll-backs and therefore enabling a more speculative simulation regime.

Overall, we show that the presented simulator is able to achieve a parallel efficiency of up to 80% at the simulation of certain RDME models on 64 parallel processes. Furthermore, the method achieves a significantly better speedup in comparison to previously published parallel simulation algorithms based on the Gillespie method, tested on the same benchmark. We also analyze the influencing factors of the performance on a wide selection of differently constructed spatial models and present a statistical evaluation of so-called *performance indicators*.

Chapter 4

Conclusions

In this thesis I described the applications and concepts that have been considered in our research so far. I introduced two application areas and several sequential and parallel discrete-event simulation methods.

I conclude that discrete-event modelling is a useful and flexible concept applicable in many areas of computational science. The main advantages of DES are that models may consist of discrete and continuous state variables and that the dynamics can be given by both, stochastic and deterministic terms.

I have discussed several practical issues regarding the simulation of discrete-event models. For example, if the aim is to estimate parameter sensitivity of stochastic models, caution has to be taken in the choice of the simulation method, as the effect of parameter perturbations may not be observable when a less suitable method is employed.

Regarding the efficiency of DES, I conclude that the parallelization of such algorithms is not trivial, as they typically consist of a sequential simulator loop where the state updates are causally dependent on each other. Hence, PDES techniques have to be used, where decomposed parts of a model are simulated concurrently and some kind of synchronization exists for events that affect several processors.

PDES methods can be categorized into optimistic and conservative approaches. Conservative methods can be used if a model is simulated at deterministic time steps. As discussed in this thesis, such simulators can also be implemented using task-based processing, where causality of state updates is maintained by a dependency-aware scheduler.

Optimistic PDES is necessary if a model is simulated at stochastic time steps. In such case it is important that the simulation regime does not become "over-optimistic" in the sense that roll-backs dominate the total simulation cost. Adaptive methods have been proposed that attempt to find a

trade-off between optimistic and conservative execution. Results shown in this thesis suggest that such methods are very suitable for parallel simulation of spatial stochastic systems, as for example systems governed by the reaction-diffusion master equation.

4.1 Outlook

I am looking forward to further extend my research in the areas discussed in this thesis, and here I present a short outlook on future work.

A challenging task is the parallelization of the Next-subvolume method (NSM), which is a different task than the parallelization of the AEM as there is no explicit information of future diffusion events available. However, I believe that this is an important step in the current research, as the algorithm finds frequent use in the Computational Systems Biology community and efficient multi-core implementations are therefore demanded.

Another challenging aspect is the consideration of an approximative parallel simulation of systems governed by the RDME, which allows for some *controlled error* taken in the simulation of a trajectory. Such a method would likely improve the efficiency of parallel simulations, as small deviations from temporal causality would not necessarily lead to roll-backs in affected sub-domains.

Lastly, a natural progression of the work presented in Paper II is to analyse the scaling of the task-based approach in a distributed memory environment. This task is motivated by ongoing extensions of the underlying modeling framework which substantially increase the computational intensity of simulations.

Bibliography

[1] M. Aldinucci, M. Torquati, C. Spampinato, M. Drocco, C. Misale, C. Calcagno, and M. Coppo. Parallel stochastic systems biology in the cloud. *Briefings in Bioinformatics*, 15(5):798–813, 2014.

[2] G. Arampatzis, M. A. Katsoulakis, P. Plechac, M. Taufer, and L. Xu. Hierarchical fractional-step approximations and parallel kinetic Monte Carlo algorithms. *Journal of Computational Physics*, 231(23):7795–7814, 2012.

[3] R. E. Bryant. Simulation of packet communication architecture computer systems. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1977.

[4] C. G. Cassandras and S. Lafortune. Systems and models. In *Introduction to Discrete Event Systems*, pages 1–51. Springer US, 2008.

[5] K. M. Chandy and J. Misra. Asynchronous Distributed Simulation via a Sequence of Parallel Computations. *Commun. ACM*, 24(4):198–206, Apr. 1981.

[6] S. R. Das. Adaptive protocols for parallel discrete event simulation. *The Journal of the Operational Research Society*, 51(4):385–394, 2000.

[7] L. Dematté and T. Mazza. On parallel stochastic simulation of diffusive systems. In *Computational Methods in Systems Biology*, number 5307 in Lecture Notes in Computer Science, pages 191–210. Springer Berlin Heidelberg, 2008.

[8] B. Drawert, M. J. Lawson, L. Petzold, and M. Khammash. The diffusive finite state projection algorithm for efficient simulation of the stochastic reaction-diffusion master equation. *The Journal of Chemical Physics*, 132(7):074101, 2010.

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

[10] S. Engblom. Parallel in Time Simulation of Multiscale Stochastic Chemical Kinetics. *Multiscale Modeling & Simulation*, 8(1):46–68, 2009.

[11] S. Engblom. On the stability of stochastic jump kinetics. *Applied Mathematics*, 51:3217–3239, 2014.

[12] S. Engblom, L. Ferm, A. Hellander, and P. Lötstedt. Simulation of stochastic reaction-diffusion processes on unstructured meshes. *SIAM Journal on Scientific Computing*, 31(3):1774–1797, 2009.

[13] S. N. Ethier and T. G. Kurtz. *Markov Processes: Characterization and Convergence*. Wiley series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 1986.

[14] A. Ferscha. Probabilistic adaptive direct optimism control in time warp. In *Proceedings of the Ninth Workshop on Parallel and Distributed Simulation*, PADS '95, pages 120–129. IEEE Computer Society, 1995.

[15] R. M. Fujimoto. *Parallel and Distribution Simulation Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1999.

[16] C. W. Gardiner. *Handbook of stochastic methods for physics, chemistry, and the natural sciences*. Springer, 1985.

[17] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, 2000.

[18] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[19] K. A. Iyengar, L. A. Harris, and P. Clancy. Accurate implementation of leaping in space: The spatial partitioned-leaping algorithm. *The Journal of Chemical Physics*, 132(9):094101, 2010.

[20] S. Jafer, Q. Liu, and G. Wainer. Synchronization methods in parallel and distributed discrete-event simulation. *Simulation Modelling Practice and Theory*, 30:54–73, Jan. 2013.

[21] M. Jeschke, R. Ewald, and A. M. Uhrmacher. Exploring the performance of spatial stochastic simulation algorithms. *Journal of Computational Physics*, 230(7):2562–2574, 2011.

[22] N. G. V. Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier, Aug. 2004.

[23] W. O. Kermack and A. G. McKendrick. A Contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 115(772):700–721, 1927.

[24] J. Liu. *Parallel Discrete-Event Simulation.* John Wiley & Sons, Inc., 2010.

[25] A. Malik, A. Park, and R. Fujimoto. Optimistic Synchronization of Parallel Simulations in Cloud Computing Environments. In *IEEE International Conference on Cloud Computing, 2009. CLOUD '09*, pages 49–56, Sept. 2009.

[26] M. Mascagni and A. Srinivasan. SPRNG: A Scalable Library for Pseudorandom Number Generation. *ACM Transactions on Mathematical Software*, 26(3):436–461, 2000.

[27] R. Metzler. The Future is Noisy: The Role of Spatial Fluctuations in Genetic Switching. *Physical Review Letters*, 87(6):068103, July 2001.

[28] M. S. Nobile, P. Cazzaniga, D. Besozzi, D. Pescini, and G. Mauri. cuTauLeaping: A GPU-Powered Tau-Leaping Stochastic Simulator for Massive Parallel Analyses of Biological Systems. *PLoS ONE*, 9(3):e91963, 2014.

[29] J. V. Rodríguez, J. A. Kaandorp, M. Dobrzyński, and J. G. Blom. Spatial stochastic modelling of the phosphoenolpyruvate-dependent phosphotransferase (PTS) pathway in escherichia coli. *Bioinformatics*, 22(15):1895–1901, 2006.

[30] S. Srinivasan and P. F. Reynolds, Jr. Elastic Time. *ACM Trans. Model. Comput. Simul.*, 8(2):103–139, Apr. 1998.

[31] B. Wang, B. Hou, F. Xing, and Y. Yao. Abstract next subvolume method: A logical process-based approach for spatial stochastic simulation of chemical reactions. *Computational Biology and Chemistry*, 35(3):193–198, 2011.

[32] Z. Xiao, B. Unger, R. Simmonds, and J. Cleary. Scheduling Critical Channels in Conservative Parallel Discrete Event Simulation. In *In Proceedings of the 13th Workshop on Parallel and Distributed Simulation*, pages 20–28, 1999.

Recent licentiate theses from the Department of Information Technology

2015-003 Fredrik Hellman: *Multiscale and Multilevel Methods for Porous Media Flow Problems*

2015-002 Ali Dorostkar: *Developments in preconditioned iterative methods with application to glacial isostatic adjustment models*

2015-001 Karl Ljungkvist: *Techniques for Finite Element Methods on Modern Processors*

2014-007 Ramūnas Gutkovas: *Advancing Concurrent System Verification: Type based approach and tools*

2014-006 Per Mattsson: *Pulse-modulated Feedback in Mathematical Modeling and Estimation of Endocrine Systems*

2014-005 Thomas Lind: *Change and Resistance to Change in Health Care: Inertia in Sociotechnical Systems*

2014-004 Anne-Kathrin Peters: *The Role of Students' Identity Development in Higher Education in Computing*

2014-003 Liang Dai: *On Several Sparsity Related Problems and the Randomized Kaczmarz Algorithm*

2014-002 Johannes Nygren: *Output Feedback Control - Some Methods and Applications*

2014-001 Daniel Jansson: *Mathematical Modeling of the Human Smooth Pursuit System*

2013-007 Hjalmar Wennerström: *Meteorological Impact and Transmission Errors in Outdoor Wireless Sensor Networks*

2013-006 Kristoffer Virta: *Difference Methods with Boundary and Interface Treatment for Wave Equations*

UPPSALA
UNIVERSITET

Department of Information Technology, Uppsala University, Sweden