Parallel Discrete Event Simulation as a Paradigm for Large Scale Modeling Experiments

© Lev Shchur © Liudmila Shchur
Science Center in Chernogolovka,
142432 Russia
shchur@chg.ru , lvs@chg.ru

Abstract

Parallel Discrete Event Simulation (PDES) is a method introduced to conduct simulation of a complex system, which consists of a large number of objects. The PDES has been known for about 30 years and we discuss why further analysis of the method is important nowadays. We briefly introduce the main features of the method and discuss the current state of PDES. We present a survey of the method as well as promising applications.

1 Introduction

Methods of parallel algorithms developed in the middle of the last century are weakly applicable on the modern supercomputers. During the last two decades the high-performance computing evolved from the big mainframes with a complicated central processing unit (CPU) into the parallel processing of large numbers of CPUs and GPGPUs (general-purpose graphics processing units). Running hundreds of thousands or even millions of CPUs in a single task is one of the challenging problems of high-performance computing (HPC). We will focus our discussion on how the parallel discrete event simulation (PDES) method may be used in large-scale simulations.

There are two ways to parallelize a simulation model: time-parallel and space-parallel [1]. The time-parallel approach partitions the simulated time axis into intervals $[T_1,T_2]$, $[T_2,T_3]$, ..., $[T_i,T_{i+1}]$, ... and assigns each interval to a separate process. A process i computes a sample path for the interval $[T_i,T_{i+1}]$. The final state of the system in the sample path $[T_{i-1},T_i]$ must be the initial state in the sample path for $[T_i,T_{i+1}]$. It is clear that the time-parallel simulations are restricted to special models that fulfil that requirement.

The space-parallel approach partitions the system being modelled into a collection of subsystems, and assigns a logical process (LP) to each subsystem. Therefore, each LP is just a sequential event-driven simulator. Sending an event to a LP is equivalent to scheduling that event in the receiving list of LP pending events.

The PDES is a space-parallel approach with an execution of a single discrete event simulation program on a parallel computer or on a cluster of computers [1]. It is widely used in economics and logistics, and sometimes applied in physics and computer science. A system that should be simulated is divided into disjoint *subsystems*, which are mapped onto the programming *objects* [2]. Subsystems are not isolated during the simulations, and dependencies between objects should be resolved properly. There are three main essentials of PDES. First, it is assumed that changes of possible, or potential, dependencies occur at some particular moments of time. It is supposed that these moments of time are spread on a scale that is large enough compared to an elementary unit of simulation time. Therefore, changes are considered to be discrete (although random) in time, and are called *discrete events*. The objects evolve independently in time, as soon as there are no dependencies generated. We need to have some protocol in order to process dependencies correctly, in other words we would like to keep causality. Using general language we can say that we need some protocol for synchronization of discrete events.

The main idea of PDES is to use an asynchronous mechanism and to implement the concept of Virtual Time [2]. *A local virtual time* variable τ_i is associated with each object i. As soon as a new time event is generated by an object i, that event is *stamped* with the local virtual time (LVT) τ_i. A message containing information on the event is generated and is sent to other objects. A message sending mechanism is the second essential feature of PDES. The third essential feature of PDES - there is no information exchange between subsystems through common variables and there is no access of objects to a shared memory. Instead, all the information is distributed via messages. The ensemble of LVT τ_i generates a *profile* of local virtual times. A minimal value of the profile defines *time horizon,* and an evaluation of this value defines *Global Virtual Time* (GVT) of simulations. An evolution

Proceedings of the XVII International Conference «Data Analytics and Management in Data Intensive Domains» (DAMDID/RCDL'2015), Obninsk, Russia, October 13 - 16, 2015

of LVT profile and of GVT in time depends on a particular scheme of managing causality, and the analysis of time profile properties (which is related to the problem of synchronization!) is one of subjects of the present paper.

Originally, the PDES was designed to deal with complex and heavy problems. Suppose, one has to simulate a complex system with the condition that simulation is resource consuming (huge processor time, big RAM, and big data storage, large number of sensors) and could not be fit in a single computer. An appropriate simulation method should be able to handle any problem. The main requirement was not in the effective use of CPU time, or storage, or other hardware loading level. The main requirement was just to have *some* possibility to simulate. It is not a usual requirement of scalability, which is traditionally associated with the effective use of hardware. Contrary, scalability in the PDES is just a possibility to evolve a system regardless of how large and complex this system is. During the last years, the PDES method attracts interest of a computational physics community, which is looking for methods to simulate huge systems that can be simulated on computers with millions of nodes [4,5,6]. A discussion of that issue is another subject of the paper.

The third subject of the paper is a discussion of the possibility to use the PDES method on a lower level, on a level of system software that will link together billions of nodes (computational units, CPUs, cores, sensors, etc.) within one task. This was mentioned in the White Paper "Performance Technologies for Peta-Scale Systems" prepared by a group of researchers from US National Laboratories and associated Universities [7].

The paper is organized as follows. In Section 2 we describe briefly the PDES method. In Section 3 we review the research on the evolution of LVT time profiles. In Section 4 we discuss an application of PDES for in physics in informatics. In Section 5 we discuss a possible application of the PDES method in system software. In Section 6 we summarize our discussion and point out possible future research.

2 Time evolutions in Parallel Discrete Event Simulation method

We discuss an evolution of the local time profile and of the GVT in PDES algorithms [8,9] using simplified models of that evolution. In a sense, it is a modelling of the possible PDES algorithms. As we already emphasized, the evolution of the LVT and the GVT is associated with synchronization problems. The importance of the model-of-model approach becomes clear if we take into account that using that approach we may prove, for example, that deadlocks are never happens in conservative algorithms. In terms of LVT profile time evolution, it means that the speed of the time profile is always positive because there exists at least one object with the LVT lower than LVTs of other objects, therefore that object may proceed in time. In the case of local algorithms it is possible to prove existence of a lower bound of that speed [10].

We use an extension [11] of Virtual Time concept on the current multi-core and multi-thread architecture of hardware and software. In our discussion, the LVT is associated with logical processes (LP) and not with processing elements, as it was the case in "before-core era" of hardware [9]. A processing element (PE) is associated with hardware, and a LP is more flexible because it is the virtual. Nowadays this extension is even more important with GPGPU in which threads may run LP in parallel and concurrently.

Let us define an abstract model of simulation in PDES. The whole physical system is decomposed into N subsystems. Each physical subsystem is associated with a *logical process*. Each logical process develops in time, changing its internal state at some moments of local simulation time. Time of simulation is measured in the Global Virtual Time [2], which is a *minimal* time of the LVT profile. We call a change in the internal state of simulated subsystem i as an *event* that happens at logical process i, it is denoted LP(i). We suppose that intervals of time between events are distributed according to a Poisson law. A logical process generates a message with information on the changes in the internal state, marked up (stamped) by the value of LVT. Sequence of LVT times τ_i is not decreasing. Each logical process LP(i) receives a message and may include the received information in simulation, or ignore information if it is not necessary for simulation. In the case when LPs perform simulations in parallel, the following problem may occur. It may happen that LVT τ_j of LP(j) is higher than τ_i in the received message from LP(i), and LP(j) does depend on the information contained in the message generated by LP(i). Therefore, causality is violated! Fujimoto classified PDES algorithms into two groups [1,9]. They are called optimistic algorithms and conservative algorithms reflecting the way in which synchronization is performed. FaS algorithm is introduced and described in the paper [10], it may be viewed as representatives of the third group of PDES.

2.1 Conservative PDES algorithm

In the case of a conservative algorithm, a subsystem waits for all events to happen in every subsystem from each it depends on, before proceed in time. Clearly it is possible only for the simulations in which we know in advance all dependencies between subsystems in the physical system (named as arcs on graphs of LPs). Therefore, in a conservative algorithm LVT τ_i of LP(i) should not be greater than LVT τ_k of all LPs from which LP(i) depends on [1].

In realizations, one has to define matrix of dependencies M(i,j) with elements equal to 0 or 1. If an element M(i,j) equals to 1, than LP(j) does may depend on events generated by LP(i). This matrix may be viewed as a matrix of sender-receiver for message sending. For each LP(i) the corresponding column of a matrix is defined, and each time event happens in

simulations of LP(i), it sends messages to all LP(j) for which element of matrix M(i,j) is equal to one. Accordingly, LP(j) will never go forward in simulation with LVT τ_j larger than minimal τ_k in its queue of messages stamped with the LVTs.

In subsection 3.1 we will show that a conservative algorithm never leads to a deadlock and will demonstrate how simulation speed of GVT behaves for some simple cases

2.2 Optimistic PDES algorithm

In an optimistic algorithm, all LPs are evaluated in time with assumption that causality is fulfilled. Clearly, LVT of subsystems increased independently, and it may happen that LP(j) will receive message with time stamp τ_i smaller than τ_l in the last processed message. Causality is broken! LP(j) proceeds using the wrong information, while ignoring an event which happened earlier in simulation time, $\tau_i < \tau_l$. An optimistic algorithm introduces a protocol of *rollback* in time that resolves the problem of causality. This is done again using a message sending procedure. The most well known optimistic protocol is Time Warp paradigm [2]. The event causing rollback is called a *straggler*. A recovery is organized by undoing effects of all events that have been processed prematurely by the LP receiving the straggler.

In addition to the Input Message Queue and Output Message Queue, each LP keeps State Queue, which is a list of the recently processed events. It gives a possibility to restore an old state vector on rollback. LP sends anti-message which annihilates the original one when it reaches its destination LP. So, rollback generates an anti-message (negative), which annihilates while meeting the corresponding primary (positive) message. This happens like annihilation in particle physics, with an electron and a positron. If LP(i) receives an anti-message that corresponds to a positive message, which it has already processed, then the process must be rolled back itself to undo the effect of processing that positive message. In the case if both an anti-message and a positive message are in LP(i) queue they just annihilate within queue and do not cause any rollback by LP(i). Recursive repetition of this procedure allows cancelling all erroneous calculations. This recursive procedure may lead to the *avalanche* in the rollback, but the *length of the avalanche* is limited by GVT. No event with time-stamp smaller than GVT will ever be rolled back, so LPs with GVT may discard that event from the State Queue.

In subsection 3.2 we will describe a simple model of evolution of LVT profile, and demonstrate an analogy with a process known in physics as an unrestricted surface growth.

2.3 Freeze-and-Shift (FaS) algorithm

To make the classification of the PDES algorithms possible, two steps are performed. First, concept of Virtual Time [2] was generalized, as LVT is associated with the logical processes (LP) rather than with the processing elements (PE). Second, it is assumed that it is possible to handle several LPs within one PE, and that communication within PE may be efficient using conservative algorithm. In that case we can use correspondence of time profile evolution in the conservative PDES [10] with the evolution of surface growth on substrate [12], described by Kardar-Parisi-Zhang partial differential equation, named KPZ equation. There is one-to-one correspondence between classification of boundary conditions in KPZ equation and classification of PDES algorithms [11]. In KPZ equation, we may divide space into domains, and classify possible boundary conditions (BC) between domains. There are three possibilities of BC: continuous, free, and fixed. Mapping of KPZ onto PDES is as follows. Domain corresponds to PE, and BC between domains corresponds to the communication protocol. Therefore, with many LPs running within one PE and updated using conservative algorithm, there are three possible algorithms for communication protocol between PEs (associated with corresponding BC): conservative, optimistic, and FaS. FaS is abbreviation of the Freeze-and-Shift algorithm.

Let us consider the simplest parallel simulation task, in which LPs are placed along the horizontal lines (see Figure 1), and send time-stamped messages only to the left and right LP. Let us assume we distribute l logical processes on each of N processor elements.

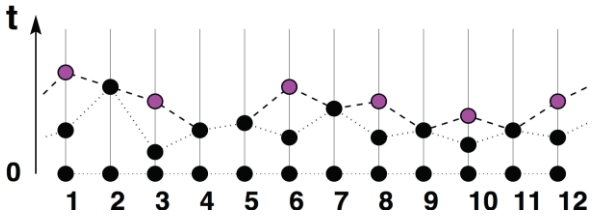

Fig. 1. Possible realization of the first two "steps" of the time profile evolution in the conservative algorithm. Number of LPs is equal to the number of PEs. Each LP sends message only to the neighboring LP at the left and right side.

Figure 1 shows time profile of simulation with 12 LPs on the 12 PEs. We start with all LVT $\tau_i = 0$. The system evaluates to the second raw of black LPs. At that "moment" GVT is defined by the process with the minimal LVT time, it is process number 3. At the "next" step of simulation only LPs with the minimal local virtual time (number 1, 3, 6, 8, 10, and 12) are allowed to proceed to the possible time marked by pink, according with the conservative algorithm. The third "step" consists in possible evaluation of those processes, which are sitting in local minima of LVT profile; it is processes with number 4, 7, 9, and 11.

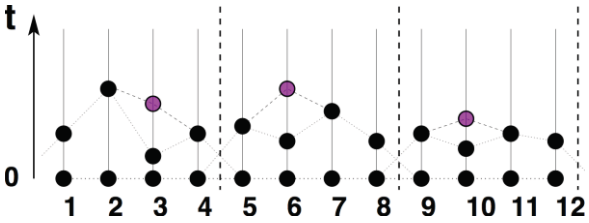

Fig. 2. Possible realization of the first two "steps" of the time profile evolution in the FaS algorithm. Number of LP=4 in each of PE=3. Each LP sends message only to the neighboring LPs at the left and right side: it is conservative algorithm between LP in each PE, and boundary LPs are frozen.

Figure 2 demonstrate FaS algorithm applied to the chain of LPs communicating only with the neighbors. 12 LP distributed over 3 PE, with 4 LP in each PE. Boundary LP (number 1,4,5,8,9, and 12) is not allowed to proceed after step 1 - they are fixed. On the "next step" only LP in the local minima and not on the border of PE are allowed to proceeds simulation, it is processes 3, 6, and 10, marked by pink. Compare Figure 2 and Figure 1 – the difference only on the border of PE. We have to note, that no LP could proceed after that step – there are no local minima of LVT time profile. The next step of FaS algorithm is to perform shift phase, i.e. perform exchange of LPs between PEs. For the case depicted in Figure 2, we have to "shift" LPs cyclically by two positions. So, PE number 1 will keep LPs 3,4,5, and 6; PE number 2 – LPs 7,8,9, and 10; PE number 3 – LPs 11,12,1, and 2. So, from that boundary LPs (the fixed ones) will be 3, 6, 7, 10, 11, and 2, this is the freeze phase of FaS algorithm. After that, processes 4, 8, 12, and 1 could proceed.

The forward process depends on the number l of logical processes on each of N processor elements, and time to stop can be estimated as $l^2/4$ for the linear chain of LPs discussed for simplicity.

FaS allows for an effective realization on the parallel computers, clusters of computers, and in grid computing. It effectively decouples the computation phase and communication phase for these parallel computations. This allows, for example, the programmer to utilize fast block-memory-transfer commands. Furthermore, it should allow the efficient simultaneous execution of both computation part and communication part when they are performed by independent hardware.

3 Evolution of time profile in PDES

In this section we will discuss evolution of LVT time profile in PDES. We do not discuss case studies in which particular realization of PDES are investigated empirically. Instead, we will discuss properties of models which mimics essential features of PDES and which can be investigated using methods of applied mathematics and mathematical physics. First, we discuss results of the research done in series of papers by group of Korniss, mainly on the conservative algorithm. Next, we discuss results of our research with Mark Novotny, mainly for the optimistic algorithm.

3.1 Conservative PDES and Kardar-Parisi-Zhang equation

Let us consider the simplest case of application of conservative algorithm to the chain of LPs communicating with neighbors only as already considered in Subsection 2.3 and depicted schematically in the Figure 1. LVT τ_i is associated with each LP(i). We start with all $\tau_i = 0$ (LVT time profile is flat at $t = 0$, where t is artificial time, and the discrete one for simplicity[1]). Evolution of LVT profile may be written as iterative process: $\tau_i(t+1) = \tau_i(t)+\eta_i(t)$ *if* $\tau_i(t) \le min\{\tau_{i-1}(t), \tau_{i+1}(t)\}$, and $\tau_i(t+1) = \tau_i(t)$, otherwise. Here $\eta_i(t)$ are random variables. When LP(i) advanced in time it sends messages to the right LP(i+1) and left LP(i-1) stamped with the LVT $\tau_i(t+1)$. In this process causality is never violated.

This algorithm is free of deadlock. This can be seen qualitatively from the fact that statistically there are four possible local configurations for LVT of LP(i), and only one of them with the local minima. Therefore, average speed of GVT (time horizon) should be 1/4. In fact, numerical simulation of Korniss et al model gives average speed of the time horizon is 0.246 410(7) [10] for the Poisson distribution of $\eta_i(t)$, 0.267(4) [11] for the uniformly distributed $\eta_i(t)$, and 0.258(5) [11] for the Gaussian noise. So, average utilization of CPU time in this simplified local version of conservative algorithm is close to 25 per cent.

Korniss et al argued that in the continuum limit (the increment of the artificial time t is infinitesimal) iteration process for the LVT time profile obeys the equation introduced by Kardar-Parisi-Zhang (KPZ) for the surface growth on the solid substrate [10,11]. This is very important observation because it provides mapping of conservative PDES algorithm with local message exchange onto KPZ problem. In turn, KPZ problem is well investigated [12], and solution demonstrates universal properties of the profile fluctuations depending on the time and system size.

First, from the fluctuations of surface profile it is possible to estimate the width growth of the profile. Mapping this result from KPZ onto the properties of LVT time profile in PDES shows us that width of LVT grows with time as $t^{3/2}$. This means that LPs are unsynchronized with simulation time growth. This is a bad news, but from KPZ analysis it follows, that width growth is saturated at some moment of time. Saturation level of LVT width is proportional to the square root from the number of LPs. This is a good news.

3.2 Optimistic PDES and unrestricted surface

[1] One should not confuse artificial time t we introduce for the sake of simplicity with the time at which events happens and time-stamped messages are generated.

growth

Model for the evolution of time profile, which mimics essential features of optimistic algorithm, was introduced first in [13]. Let us consider again for the sake of simplicity the linear chain of LPs. The first step is the optimistic unrestricted evolution of LVT at which simulation evolves *forward* in time and the second is the rollback (or *backward*) algorithm of sending anti-messages. For this purpose one may introduce two parameters, ***F*** and ***B*** associated with two steps of optimistic scheme. Namely, at the first step one evaluates the time horizon by updating time of the randomly chosen LPs with value ***F*** following the Poisson distribution, and every LP can be chosen with the non-zero probability. Then one have to *relax* (rollback!) LVT profile ***B*** times: for all LP(i) value of LVT time changed to the value of LVT time of the nearest left LP(i-1) or right LP(i+1). Value of ***B*** satisfies the Poisson distribution. The average value of the speed profile ***u(t)*** evolves proportional to $(q-q_c)^a$, with ***q=1/(1+B),*** with a value of ***a=1.74*** close to the critical exponent of directed percolation and describing *roughening transition* in one-dimensional unrestricted growth process [14]. The value of q_c***=0.23(3)*** corresponds to the value of the parameter ***B=3.3***. It should be mentioned that the width of the LVT profile acts better for values ***q*** far away from the critical value q_c. When close to q_c the system practically does not evolve in time. The growth of width provides an analogy with the roughening transition, which is in the same universality class as a directed percolation [14].
Practically this means that if the length of avalanche (see subsection 2.2) is larger than ***B=3.3*** in average, optimistic algorithm will not proceed in time but will be rather stock. It is quite important to perform case studies of optimistic algorithm in order to understand that prediction in details.

4 Using PDES in physics and informatics

The most important feature of PDES for using in physics is that PDES algorithms are scaled very naturally with the physical system size (number of objects, or logical processes) and with the hardware size (number of nodes, cores, threads) with the only requirement that the time for message distribution is smaller than computation time. This is valid for the simulation in which time to simulate object is much larger than time to send message. It is clear that more complex and hard simulations the more gain can be reached.

Last years a number of papers with the analysis of implementation of PDES ideas in the large-scale computing of models in physics have been published. Most of discussions were done for the analysis of application of PDES algorithm to the Ising model, which is the simplest model of ferromagnetism, and which is standard model to test algorithms in Monte Carlo simulations [15]. Here we review some of the papers that may be of general interest.

SPPARKS is the project running by Sandia National Laboratory and developing kinetic Monte Carlo simulator [16]. SPPARKS runs on single processors or in parallel using message-passing techniques and a spatial-decomposition of the simulation domain. The code is designed to be easy to modify or extend with new functionality. Main idea is in partitioning of simulation space into the computational domains, and performing simulations in parallel within some time window. In some sense, this idea is similar to FaS algorithm. It implements several KMC solvers whose serial computational complexity ranges from *O(N)* to *O(NlogN)* to *O(1)* in the number of events *N* owned by a processor. In a generic sense the solvers are catalog a list of "events", each with an associated probability, choose a single event to perform, and advance time by the correct amount. Events may be chosen individually at random, or by sweeping over sites in a more ordered fashion. Problems they addressed are magnetic spin models, surface growth on substrate, etc. One of the simulations is connected with the microstructural evolution during sintering [17]. The model is a grain growth with physical effects of particle diffusion, vacancy diffusion, and vacancy annihilation. This sintering model is able to capture all the necessary mechanism to simulate simple solid-state sintering correctly. It was demonstrated by comparing it to the three--dimensional, in-situ images taken in a high-energy synchrotron during the sintering of Cu particles.
Similar project is running by Lawrence Livermore National Laboratory, also for the parallel kinetic Monte Carlo [18,19]. It is based on ideas of virtual time and random order of simulating computational domains within time window. It is tested on the model of billion atoms. The approach was tested with the Ising model (the simplest ferromagnetic model) and demonstrated that scalability with the number of nodes is close to the ideal one.

A number of projects connected with simulation of communication networks are using PDES. One of such projects is running by INRIA, and it is connected with the simulation of Border Gateway Protocol using optimistic PDES [20] with number of nodes from 10 to 100 thousands. In the last case simulation lasts up to 36 hours on 4-core Intel Xeon W5580 3.2Ghz with 64GB of RAM running 64-bit Fedora 12 on a Linux kernel 2.6.32.

5 PDES and system software

The goal of high end computing (HEC) initiative in US and Europe is to deploy large-scale computing platforms with hundreds thousands of nodes. In the White Paper “Performance Technologies for Peta-Scale Systems” the group of US researchers emphasized importance of the research on the program tools and software facilities, and in particular in system simulation [7]. They see that as “an open-source architectural simulation framework and API that enables

plug-and-play between separately-developed simulators for different architectural features… and would also enable zoom-out and zoom-in between statistically-based and cycle-accurate simulation techniques". Solution seen by them as "the simulations will be decomposed into logical processes, and will be synchronized by either conservative or optimistic methods … as developed in PDES community". The reason is that by such approach the high degree of computational parallelism in the simulation will perfectly match the high degree of real parallelism of HEC systems.

There are several simulators available as open source software.

The most famous realization of Time Warp synchronization algorithm is Rensselaer's Optimistic Simulation System, named as ROSS simulator [21]. It is public domain software, which can be installed on the cluster. Recently group of researchers from Lawrence Livermore National Laboratory, Rensselaer Polytechnic Institute and University of Illinois at Urbana-Champaign tested ROSS on the Blue Gene architecture with successful use of almost 2 billions of cores [22] yielding speed of 504 billion events per second.

Modification of optimistic algorithm named shock-resistant Time WARP (SRTW) designed by group in Westminster University and analysis was performed using the Grid'5000 platform [23].

Group from Bologna University developed PDES software named ErlangTW based on Time Warp algorithm. Work was presented at 1st ACM SIGPLAN workshop on Functional high-performance computing (FHPC '12) [24], where some preliminary performance results on multicore and distributed architectures using the PHOLD benchmark.

There are at least two attempts to implement PDES in Cloud architecture. Fujimoto group introduced TW-SMIP (the Time Warp Straggler Message Identification Protocol) protocol resigned for environment with shared hardware resources for which it is known that traditional Time Warp (TW) algorithm shows poor performance [25]. TW-SMIP protocol defines dynamic synchronization points for individual LPs based on straggler messages (a new event with time stamp smaller than other events it has already processed), which improves efficiency of simulation.

Second example is realization of PDES simulator on Cloud/Virtual machine platforms developed in Oak Ridge National Laboratory [26]. The scalability of virtual time scheduler has been tested on 128 virtual machines multiplexed on 32 cores, showing improvement in the runtime relative to the default Cloud/VM scheduler. Authors believe that algorithmic design, observations, and results of their work are timely for emerging cloud/VM-based installations, highlighting the need for PDES-specific support in high performance PDES on the cloud/VM platforms.

6 Summary and discussion

Parallel Discrete Event Simulation paradigm is one of the paradigms for the large-scale high performance computing. It is under extensive development by many groups of researches. PDES was successfully tested on the new architectures, from Blue Gene and conventional clusters and CPUs to the Virtual Machine/Cloud approach. Efficiency of PDES does depend on the problem and it is promising to investigate properties of synchronization, scalability and efficiency.

7 Acknowledgments

Russian Science Foundation supported this work under the grant number 14-21-00158.

References

[1] R.M. Fujimoto. Parallel Discrete Event Simulation. Comm. of the ACM, 33(10), p. 30-53, 1990.

[2] A.M. Law and W.D. Kelton. Simulation modeling and analysis. McGraw-Hill. Third edition. 2000.

[3] D.R. Jefferson. Virtual Time. ACM Trans. Programming Languages and Systems, 7(3), p. 404-425, 1985.

[4] S. Miller and S. Luding. Event-Driven Molecular Dynamics in Parallel. J. Comp. Phys. 193(1), p. 306-316, 2004.

[5] D.C. Richardson, K.J. Walsh, N. Murdoch, and P. Michel P. Numerical simulations of granular dynamics: I. Hard-sphere discrete element method and tests. Icarus 212, p. 427-437, 2011.

[6] J.P. Nilmeier, J. Marian J. A Rigorous Sequential Update Strategy for Parallel Kinetic Monte Carlo Simulation. (accepted to Communications in Computer Physics. 2014) arXiv link: http://arxiv.org/pdf/1403.4674.pdf

[7] D.H. Bailey et al. Performance Technologies for Peta-Scale Systems: A White Paper Prepared by the Performance Evaluation Research Center and Collaborators. May 14 2003.

[8] R. Fujimoto. Parallel and Distributed Simulation Systems. Wiley Interscience, 2000.

[9] G. Korniss, Z. Toroczkai, M.A. Novotny, and P.A. Rikvold. From Massively Parallel Algorithms and Fluctuating Time Horizons to Nonequilibrium Surface Growth. Phys. Rev. Lett., 84, p. 1351-1354, 2000.

[10] L.N. Shchur and M.A. Novotny. Evolution of time horizons in parallel and grid simulations. Phys. Rev. E, 70, 026703, 2004.

[11] M. Kardar, G. Parisi, and Y.C. Zhang. Dynamic Scaling of Growing Interfaces. Phys. Rev. Lett., 56, p. 889-892, 1986.

[12] L.N. Shchur, M.A. Novotny. Evolution of time horizon in parallel discrete event simulations.

Proceedings of Conference “Scientific services in Internet: multicore computer world”, Abrau-Dyurso, p. 197-200, 2007 (in Russian).

[13] U. Alon, M.R. Evans, H. Hinrinchsen, and D. Mukamel. Roughening Transition in a One-Dimensional Growth Process. Phys. Rev. Lett., 76, p. 2746-2749, 1996.

[14] D.P. Landau and K. Binder. A Guide to Monte Carlo Simulations in Statistical Physics. 3d edition. Cambridge: Cambridge University Press. 2013.

[15] SPPARKS Kinetic Monte Carlo Simulator

[16] . Sandia National Laboratory. http://spparks.sandia.gov/index.html

[17] C. G. Cardona, V. Tikare, S. J. Plimpton. Parallel Simulation of 3D Sintering. Int J Comp Materials Science and Surface Engineering, 4, p. 37-54, 2011.

[18] J.P. Nimeier and J. Marian. A rigorous sequential update strategy for parallel kinetic Monte Carlo simulation. Computer Physics Communications, 185, p. 2479-2486, 2014.

[19] D. Coudert, L. Hogie, A. Lancin, D. Papadimitrou, S. Perennes, and Isaam Tahiri. Feasibility study on distributed simulation of BGP. Preprint arxiv: 1304.4750.

[20] ROSS - Rensselaer's Optimistic Simulation System. http://www.cs.rpi.edu/~chrisc/ross.html

[21] P.D. Barnes Jr, D.R. Jefferson, M. Schordan, D. Quinlan, C.D. Carothers, L.V. Kale. Extreme scale optimistic parallel discrete event simulation with dynamic load balancing. Proceedings of the 2014 Winter Simulation Conference. 2014.

[22] G. Krafft and V. Getov. Transaction-oriented simulation in Ad Hoc Grids: design and experience. Procedings of HPCS, pp. 38-44, 2008.

[23] L. Toscano, G. D’Angelo, and M. Marzolla. Parallel discrete event simulation with Erlang. FHPC '12 Proceedings of the 1st ACM SIGPLAN workshop on Functional high-performance computing. p. 83-92, 2012.

[24] A.W. Malik, A.J. Park, R.M. Fujimoto. An Optimistic Parallel Simulation Protocoal for Cloud Computing Environments. SCS M&S Magazin, 4, p. 1-9, 2010.

[25] S.B. Yoginati and K.S. Permulla. Efficient Parallel Discrete Event Simulation on Cloud/Virtual Machine platform. ACM Transactions on Modeling and Computer Simulation. 2014.