Journal of Physics: Conference Series **640** (2015) 012065 doi:10.1088/1742-6596/640/1/012065

Relation of Parallel Discrete Event Simulation algorithms with physical models

L.N. Shchur[1,2] and L.V. Shchur[1]

[1]Science Center in Chernogolovka, 142432 Chernogolovka, Russia
[2]National Research University Higher School of Economics, 101000 Moscow, Russia

E-mail: shchur@chg.ru

Abstract. We extend concept of local simulation times in parallel discrete event simulation (PDES) in order to take into account architecture of the current hardware and software in high-performance computing. We shortly review previous research on the mapping of PDES on physical problems, and emphasise how physical results may help to predict parallel algorithms behaviour.

1. Introduction

During the last decades high-performance computing evolved from big mainframes with a complicated central processing unit (CPU) into parallel processing of large numbers of CPUs and GPGPUs (general-purpose graphics processing units). Orchestrating thousands of them in a single task is one of the challenging problems of high-performance computing (HPC). We will focus our discussion on parallel discrete event simulations (PDES). PDES is the execution of a single discrete event simulation program on a parallel computer or on a cluster of computers [1]. It is widely used and is more and more applied in physics and computer science, as well as in economics and engineering. The system that should be simulated is divided into disjoint subsystems, which, nevertheless, are not isolated during the simulations, and dependencies between them should be resolved properly. There are three main essentials of PDES. Firstly, it is assumed that changes of possible, or potential, dependencies occur not at arbitrary, but at some particular moments of time. It is supposed that these changes' moments of time are spread on a scale which is large enough if compared to an elementary unit of time. Therefore, changes are considered as discrete (although random) in time, and are called *discrete events*.Subsystems 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.This is done with the concept of local virtual time [2] associated with each subsystem, and which is the second essential feature of PDES. As soon as a new time event occurs in subsystem, the message containing necessary information is generated and stamped with the local simulation time. This message is sent to all other subsystems. This is the third essential feature of PDES - there is no information exchange through common variables and no access to shared memory. Instead all the information is spread via messages. The ensemble of messages generates a profile of local simulation times. The evolution of the profile in time depends on the particular scheme of managing causality, and the analysis of profile properties is the main subject of the present paper.


Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

2. Extension of PDES concept and classification of PDES schemes

Originally [1], subsystems were associated with particular processing elements (PEs), which were assumed to be just one CPU. In paper [3] the concept was generalised taking into account that CPU may run a number of threads, and subsystems may be run in several parallel tasks. We generalise this problem further and analyse the current state of the typical HPC hardware and software.

The number of hybrid HPC systems built up with CPU and GPGPU, grows in Top500 list [1] during last years. Typically, each CPU can have several processing elements (cores) which can run concurrently two threads. We keep in mind the following mapping of a physical problem on a computing algorithm. Firstly, we associate each subsystem with a subtask. Secondly, we associate a processing element (PE) with each subtask. Thirdly, we assign local simulation time (LST) to each PE, $\tau_i(t)$, $i = 1, 2, ...N$, where t is the simulation time, and N is the total number of PEs. In the modern HPC system, we can think of PE as a core running by CPU or a kernel running by GPGPU. We do not discuss here optimisation and efficiency of simulations which depend strongly on a given and particular hardware architecture. We take into account only common and universal properties of parallel simulations on advanced hybrid HPC systems.

PEs within the PDES task can communicate conservatively or optimistically [1]. Under optimistic scheme all PEs run while assuming that all the causalities are fulfilled. Simulation is going on, and PE_i sends a message with the time stamp $T_i = \tau_i(t)$, i.e. at which event occurs. Suppose that at some moment of simulation it turns out that T_i is smaller than the current local simulation time τ_j of the processing element PE_j, that is $\tau_j > T_i$, and that PE_j depends on the corresponding event with the time stamp T_i. Therefore, causality gets broken, and the optimistic scheme provides a rollback mechanism to overcome this problem by sending anti-messages to recover causality [2]. It is clear that the recovery process typically is not local and can produce multiple anti-message processes, which can lead to an avalanche in LSTs. The time horizon will evolve back in time until no more anti-messages are generated. After that, simulations will resume and the system will proceed forward in time.

In conservative scheme, alternatively, PEs do not run until they receive all the information from dependant PEs, and causality is never broken.

Let us start our analysis with a simplified case in which our system is one-dimensional and each subsystem is dependent on local (left and right) subsystems. Let us map our system on the simple HPC architecture with N nodes, each of these nodes has M CPUs, and each CPU has c cores. This gives us hierarchy of memory access times. The fastest communication is between cores within CPU which have access to the on-chip memory. The next level of communication speed is within a node, its CPUs share the same node memory, onboard memory. The slowest level of the communication speed is the communication between nodes which interact usually over Infiniband hardware. Let us divide the task on $c * M * N$ subsystems, from which $c * M$ subsystems will run on one node.

Figure 1 presents the simplified model of hardware. Processing elements 1,2,3, and 4 are mapped on cores 1,2,3, and 4 of CPU1. Processing elements 5,6,7, and 8 are mapped on cores 1,2,3, and 4 of CPU2, etc. Solid arrows correspond to the communication within CPU, and they can be efficiently realised with the conservative scheme.

PEs within nodes (shown in Figure 1 with dashed arrows) can communicate conservatively, and we suppose that this communication is much faster than the communication between nodes (shown in Figure 1 with dotted arrows), which is usually the case[2]. Sometimes it can happen

[1] The list which ranks HPCs is updated twice a year [4] and is announced at International Conferences on Supercomputing which are held in May and November.

[2] In this case we have a mechanism which allows to run a particular group of threads within one CPU. It is reasonable for them to communicate without sending messages, and rather use a mechanism for thread synchronisation, as shown in Figure 1 with solid arrows.

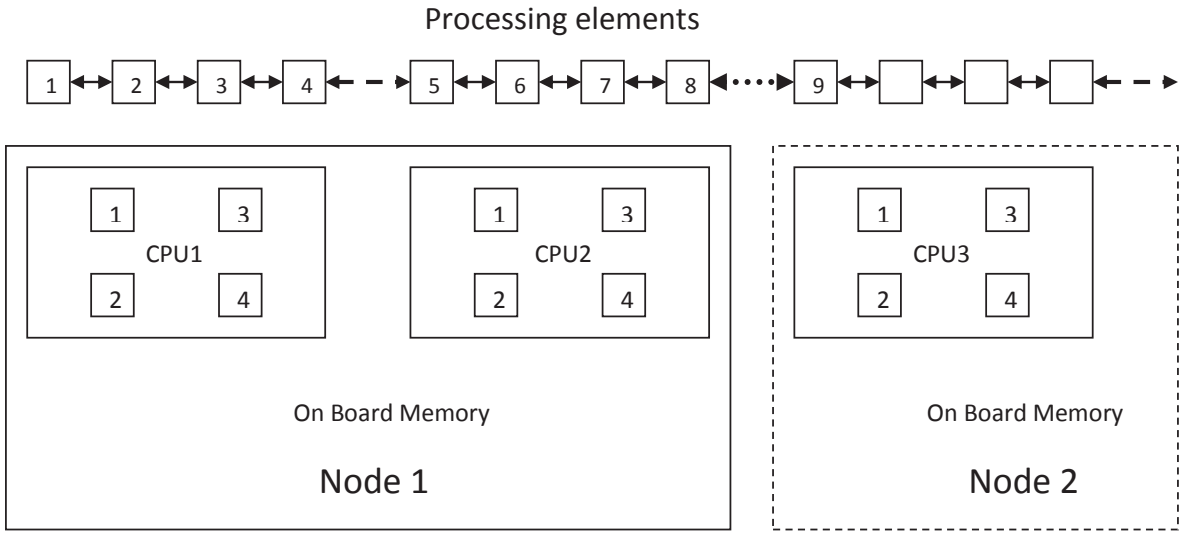

Figure 1. Mapping of processing elements on the cores, CPUs, and nodes.

that the average simulation time between events is much longer than sending messages via communication channels and analysing them. In this case the communication between nodes can be realised by any scheme, conservative or optimistic.

In fact, there are third possibility introduced in paper [3], where the mapping of PDES conservative scheme on the physical problem of surface growth in molecular beam epitaxy [5] is used for the following classification. The surface growth may be described with a nonlinear differential equation, known as KPZ equation [6] (see for details in the next section). Boundary conditions of KPZ equation can be associated with the PDES schemes such that free boundary conditions can be associated with the optimistic algorithm, and periodic boundary conditions with the conservative algorithm, and fixed boundary conditions leads to the FaS algorithm proposal [3]. Schematically, it can be explained by Figure 1. In FaS algorithm there is no communication between PEs 8 and 9 (i.e., between those PEs which are on the boundary between two nodes) for some time interval T_F, and PEs communicate conservatively within each node, leading to the propagation of freezing from the boundary PEs to the intermediate PEs (which is PE_4 and PE_5 in Figure 1). After that, Shift process initiated in a way that PEs are cyclically shifted to the right between CPUs and nodes such that PEs 1, 2, 3 and 4 are mapped on the cores 1, 2, 3 and 4 of CPU2, PEs 5, 6, 7 and 8 will be mapped on the cores 1, 2, 3 and 4 of CPU3, and so on. Then Freeze step of FaS algorithm starts again. It is argued in [3] that such algorithm may be efficient for hardware architectures where memory exchange and computations can be done in parallel.

Generalisation of our extension for the hybrid architecture of supercomputers, built up with either GPGPU with CUDA or OpenCL, or Intel ϕ architecture, is straightforward.

3. Conservative algorithm are deadlock-free

Let us describe in more details the mapping of conservative algorithm on the problem of molecular epitaxy surface growth [5].

Let us associate the value of a local simulated time $\tau_i(t)$ with PE number i, which is the

function of global time t. For $t \geq 1$ the LST evolves iteratively as

$$\begin{aligned} \tau_i(t+1) &= \tau_i(t) + \eta_i(t) \quad \text{if} \quad \tau_i(t) \leq \min\left\{\tau_{i-1}(t), \tau_{i+1}(t)\right\}; \\ \tau_i(t+1) &= \tau_i(t) \quad \text{else} \quad , \end{aligned} \tag{1}$$

where η_i are random exponential variables. Each time the PE number i advances in time it sends messages to the right $(i+1)$ and left $(i-1)$ PEs with the time stamp of its LST $\tau_i(t)$.

The iterative process (1) can be rewritten as

$$\begin{aligned} \tau_i(t+1) = \quad & \tau_i(t) + \Theta\left(\tau_{i-1}(t) - \tau_i(t)\right) \\ & \times\Theta\left(\tau_{i+1}(t) - \tau_i(t)\right)\eta_i(t) \end{aligned} \tag{2}$$

using the Heaviside step function Θ.

It was argued by Korniss *et al* [5] that the coarse-grained time-horizon $\hat{\tau}$, $\left(\hat{\phi} = \partial\hat{t}/\partial x\right)$ obeys the Kardar-Parisi-Zhang (KPZ) equation

$$\frac{\partial\hat{\tau}}{\partial\hat{t}} = \frac{\partial^2\hat{\tau}}{\partial x^2} - \lambda\left(\frac{\partial\hat{\tau}}{\partial x}\right)^2 , \tag{3}$$

which should be extended with the noise to capture the fluctuations.

Introducing the local slopes $\phi_i = \tau_i - \tau_{i-1}$, the density of local minima can be written as

$$u(t) = \frac{1}{L}\sum_{i=1}^{L}\Theta\left(-\phi_i(t)\right)\Theta\left(-\phi_{i+1}(t)\right) \tag{4}$$

and its average

$$\langle u(t)\rangle = \langle\Theta\left(-\phi_i(t)\right)\Theta\left(\phi_{i+1}(t)\right)\rangle \tag{5}$$

is the mean velocity of the time horizon, equal to 0.246410(7). Hence, the efficiency of the algorithm (in this worst-case scenario) is about 25 per cent. The algorithm is free of deadlock, since at least the PE with the absolute minimum LST can proceed.

It is argued in [7] that the width of the LST profile growth as $\sqrt{N}$ with the number N of processing elements, thus the synchronisation in conservative algorithm becomes a problem.

4. Optimistic algorithm and directed percolation problem

The model of time evolution of LST profile in optimistic algorithm can be described as simulation process consisting of two steps [8]. The first step is the optimistic unrestricted growth at which a system evolves forward in time and the second is the backward algorithm of sending anti-messages. For this purpose one may introduce two parameters, J and K associated with two steps of optimistic scheme. Namely, at the first step one evaluates the time horizon by updating all the randomly chosen LST $\tau_i(t+1)$ with J following Poisson distribution, and every PE can be chosen with the non-zero probability. Then we have to relax K times the LST of all PE_i to LST of the nearest left (right) PE_{i-1} (PE_{i+1}), with K following Poisson distribution. It was found in [8] that the average of time profile $u(t)$ evolves proportionally to $(q-q_c)^\nu$, with $q = J/(J+K)$, with a value of $\nu = 1.74$ close to the critical exponent of directed percolation [9]. The value of $q_c \approx 0.23$ corresponds to the relation between the parameters of the model $K \approx 2.39J$. It should be mentioned that the width of the LST profile acts better for values of q faraway 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 [9].

5. Nonlocal message exchange and stabilisation of the width growth

What happens to the evolution of LSTs profile in the case of long-range interactions? This question was answered in relation to the conservative algorithm in paper [10] through the introduction of random interactions of PEs throughout the whole system. This construction corresponds to the famous small-world network problem [11]. It was shown in [10] that an addition of only 10 per cents of random and non-local interactions between processing elements i and j, chosen randomly from $\{1, 2, \ldots, N-1, N\}$ processing elements, does stabilise the width of the LSTs profile. This scheme keeps property of finite evolution speed, typical for conservative scheme, and therefore, it is dead-lock free.

6. Discussion

In this paper we extend the concept of processing elements and local simulation times in order to take into account the current state of high performance hardware and software computing systems. We review the results of the association of PDES evolution with the physical processes, and emphasise that the results received from physical problems may be useful for the analysis of some classes of parallel algorithms. It seems that PDES algorithms may be extended to the simulation of systems with non-local interactions, and this investigation is in progress.

7. Acknowledgments

This work was supported by the grant of the Russian Science Foundation 14-21-00158.

References

[1] Fujimoto R M 1990 *Commun. ACM.* **33** 31
[2] Jefferson D R 1985 *Assoc. Comput. Mach. Trans. Programming Languages and Systems* **7** 404
[3] Shchur LN and Novotny M A 2004 *Phys. Rev. E* **70** 026703
[4] Top500 List `http:\\top500.org`
[5] Korniss G, Toroczkai Z , Novotny M A, and P.A. Rikvold P A 2000 *Phys. Rev. Lett.* **84** 1351
[6] Kardar M, Parisi G, and Zhang Y C 1986 *Phys. Rev. Lett* **56** 889
[7] Guclu H , Korniss G, Novotny M A, Toroczkai Z, and Rácz Z 2006 *Phys. Rev. E* **73** 066115
[8] Shchur L N and M. A. Novotny M A *unpublished.*
[9] Alon U, Evans M R, Hinrinchsen H, and Mukamel D 1996 *Phys. Rev. Lett.* **76** 2746
[10] Korniss G, Novotny M A, Guclu H, Toroczkai Z, Rikvold P A 2003 *Science* **299** 677
[11] Watts D J and Strogatz S H 1998 *Nature* **393** 440