TESTING THE SPEED-UP OF PARALLEL DISCRETE EVENT SIMULATION IN HETEROGENEOUS EXECUTION ENVIRONMENTS

Gábor Lencse and István Derka
Department of Telecommunications
Széchenyi István University
Egyetem tér 1.
H-9026 Győr, Hungary
lencse@sze.hu

KEYWORDS

parallel discrete event simulation, heterogeneous execution environments, conservative synchronisation method, relative speed-up, MPI, OMNeT++, load balancing criterion, coupling factor criterion.

ABSTRACT

This paper deals with the experimental testing and verification of the earlier proposed load balancing and coupling factor criteria for the conservative parallel discrete event simulation in heterogeneous execution environments whether they can ensure a good speed-up. The definition of the relative speed-up is extended to the heterogeneous systems in a natural way. This definition is used to measure the efficiency of the parallel simulation executed by heterogeneous systems. A closed queueing network is used as the simulation model, and it is executed on different heterogeneous test systems. Among several scenarios, it is demonstrated on the example of a heterogeneous system containing 87 CPU cores of 5 different types that a good speed-up can be achieved using the load balancing and coupling factor criteria. It is shown that the extension of the definition of the relative speed-up to the heterogeneous systems made it easy to judge the speed-up of parallel discrete event simulation in heterogeneous execution environments.

INTRODUCTION

Event-driven ***discrete event simulation*** (DES) is a powerful method for the performance analysis of information and communication technology (ICT) systems. The detailed modelling and simulation of these systems often requires a huge amount of computing power and memory. Parallelisation can be a natural solution. Kunz (Kunz 2010) points out that as the ongoing development in the hardware sector favours an increasing number of processing units over an increasing speed of a single unit thus the parallel simulation will remain an important and active field of research.

However, because of the algorithm of the event-driven DES, ***parallel discrete event simulation*** (PDES) it is not an easy task and the achievable speed-up is often limited. When doing PDES, the model of the system is divided into partitions (called Logical Processes), and the partitions are assigned to processors that are executing them. To maintain causality, the virtual times of the partitions must be synchronised. There are different methods for synchronisation (Kunz 2010). The ***conservative method*** ensures that causality is never violated. An event can be executed only if we are certain that no events with smaller timestamp exist (and also will not be generated) anywhere in the model. Unless the simulated system has a special property that the so called *lookahead* is large enough, the processors executing the partitions need to wait for each other in the majority of time, so the achievable speed-up is poor.

In the paper (Varga et al. 2003), the authors proposed a method for assessing available parallelism in a simulation model for conservative synchronization. The method requires only a small number of parameters that can be easily measured on a sequential simulation. In our paper (Lencse and Varga 2010), we checked the results of the aforementioned work for homogeneous clusters up to 24 CPU cores and also examined how the different parameters of the model influence the achievable speed-up. In our next paper (Lencse et al. 2013) we examined the criteria for a good speed-up in a heterogeneous execution environment. Our criteria were justified by several measurements in a test system. However, we could not include all the planned experiments, due to space limitations. This paper presents our further results on the examinations of different factors that influence the achievable speed-up of parallel discrete event simulation in a heterogeneous execution environment. Moreover, the definition of the relative speed-up is extended to heterogeneous systems and this extension is used in the discussion of the results of our experiments to evaluate the efficiency of parallel simulation executed by heterogeneous systems.

The remainder of this paper is organised as follows: first, a brief summary of the method for assessing the available parallelism is given. Second, our concept of heterogeneous execution environment, our criteria for a good speed up and our previous results are summarized. Third, the definition of the relative speed-up is extended to the heterogeneous systems to be able to express the efficiency of parallel simulation executed by heterogeneous systems. Fourth, our heterogeneous test environment and simulation model are described. Fifth, our further experiments and result are presented and discussed. Finally, our paper is concluded.

This topic was identified as being of importance in the parallel simulation of large systems using heterogeneous execution environments.

THE METHOD FOR ASSESSING AVAILABLE PARALLELISM

The available parallelism can be assessed using some quantities that can be measured during a sequential simulation of the model in question. The following description is taken from (Lencse and Varga 2010).

The paper (Varga et. al. 2003) uses the notations *ev* for events, *sec* for real world time in seconds and *simsec* for simulated time (model time) in seconds. The paper uses the following quantities for the assessing of available parallelism:

- ***P*** *performance* represents the number of events processed per second (*ev*/*sec*).
- ***E*** *event density* is the number of events that occur per simulated second (*ev*/*simsec*).
- ***L*** *lookahead* is measured in simulated seconds (*simsec*).
- ***τ*** *latency* (sec) is the latency of sending a message from one Logical Process (LP) to another.
- ***λ*** *coupling factor* can be calculated as the ratio of *LE* and *τP:*

$$\lambda = \frac{L \cdot E}{\tau \cdot P} \qquad (1)$$

In (Lencse and Varga 2010) we have shown that if λ is in the order of several hundreds or higher then we may expect a good speed-up. It may be nearly linear even for higher number of segments (*N*) if λ_N is also at least in the order of several hundreds, where:

$$\lambda_N = \frac{\lambda}{N} \qquad (2)$$

MODELLING AND SIMULATION IN HETEROGENEOUS EXECUTION ENVIRONMENTS

This chapter is a summary of (Lencse et al 2013).

Our Concept of Heterogeneous Execution Environments

We recommended a logical topology of two levels: a *star shaped network* of *homogeneous clusters*. This model is simple enough and can describe a typical *heterogeneous execution environment*. What is logically described as a homogeneous cluster, it can be physically, for example, a cluster of PCs with identical configuration interconnected by a switch or it can be a chassis based computer built up by several main boards, etc. The main point is that a homogeneous cluster is built up by identical configuration elements especially concerning CPU type and speed as well as memory size and speed. The homogeneous clusters are interconnected logically in a star shaped topology. The physical connection can be a switch or the topology may be different but our model considers it to be a star for simplicity.

Criteria for Achieving a Good Speed-up

We set up two criteria. The ***load balancing criterion*** requires that all the CPUs (or CPU cores) should get a *fair share* from the execution of the simulation. A fair share is proportional to the computing power of the CPU *concerning the execution of the given simulation model*. (This is very important, because, for example, using different benchmark programs for the same set of computers one can get seriously different performance results.) Thus, for the fair division of a given simulation model among the CPUs, the CPUs should be benchmarked by the same type of simulation model that is to be executed by them (but smaller in size, of course). The *lookahead* or ***coupling factor criterion*** is the same as presented and tested in (Lencse and Varga 2010) up to 24 CPU cores.

The Most Important Results

The load balancing criterion was justified by measuring the execution time of a model with different partitioning. The results of our experiments were quite close to the values computed according to the load balancing criterion. (See more details later.) The coupling factor criterion was justified by different scenarios including a simulation executed by 64 CPU cores of 4 types resulting in a good speed-up.

EFFICIENCY OF PARALLEL SIMULATION EXECUTED BY HETEROGENEOUS SYSTEMS

Relative Speed-up of Program Execution by Heterogeneous Systems

First, the definition of the relative speed-up of parallel execution of programs is extended for heterogeneous systems (in general, not only for simulation).

The conventional definition of the *speed-up* (s_n) of parallel execution is the ratio of the speed of the parallel execution by *n* CPUs and the sequential execution by 1 CPU that is equal with the ratio of the execution time of the sequential execution (T_1) and that of the parallel execution (T_n):

$$s_n = \frac{T_1}{T_n} \qquad (3)$$

The *relative speed-up* (r_n) can be calculated as the ratio of the speed-up and the number of the CPUs that produced the given speed-up:

$$r_n = \frac{s_n}{n} \qquad (4)$$

The relative speed-up measures the *efficiency* of parallel execution. A relative speed-up value of 1 means that the

speed-up is linear that is the computing power of *n* CPUs can be fully utilized.

When dealing with heterogeneous systems, not only the number of the CPUs but also their performance is to be taken into consideration. We were looking for a definition of the relative speed-up of heterogeneous systems that can be used to measure the efficiency of program execution by the heterogeneous systems in the same way: its value of one should mean that the computing power of all the CPUs (from different types) can be fully utilized.

Let us denote the *number of the CPU types* in a heterogeneous system by *NT*, the *number* and the *performance* of CPUs available from *type i* by N_i and P_i, respectively. The *cumulative performance* of the heterogeneous system is:

$$P_c = \sum_{j=1}^{NT} P_i \cdot N_i \qquad (5)$$

Let us denote the *execution time* of a given program by a single CPU of *type i* by T_i and let T_h denote the *execution time* of the given program by the *heterogeneous system*. The speed-up of the heterogeneous system compared to the sequential execution by one CPU of type *i* is:

$$s_{h/i} = \frac{T_i}{T_h} \qquad (6)$$

The relative speed-up of the heterogeneous system against the sequential execution by one CPU of type *i* is now defined as:

$$r_{h/i} = \frac{T_i \cdot P_i}{T_h \cdot P_c} \qquad (7)$$

We believe that this definition can be used in general for measuring the efficiency of program execution by heterogeneous systems. Thus, for simplicity, we used the word "CPU" in this section, but the expression "CPU core" could be used instead, as it is used everywhere else in this paper.

Measuring the Efficiency of Parallel Simulation Executed by Heterogeneous Systems

If the above definition of relative speed-up of program execution by heterogeneous systems is used for measuring the efficiency of parallel simulation executed by heterogeneous systems and the performance values of the CPU cores from different types are measured by benchmarking them with the same simulation model (expressing its value in events per seconds and the value of execution time in seconds) then the numerator of expression (7) gives the same values for all the values of *i* (that is its value is independent of the CPU core types): it is equal with the total number of events in the sequential simulation.

Note that the number of events in the parallel version of the simulation may be higher (e.g. due to communication and synchronization overhead) but only the events of the original sequential simulation are the essential part of the operation of the original model. Thus efficiency should consider the events of the original sequential simulation only.

Denoting the total *number of events* in the sequential simulation by N_E, definition (7) can be rewritten as:

$$r_h = \frac{N_E}{T_h \cdot P_c} \qquad (8)$$

Thus, we have shown that the value of relative speed-up calculated by (8) does not depend on which CPU core it was calculated against, it characterises the parallel simulation itself. Note that this is still true if one uses definition (7), thus one can use any of them selecting on the basis of which values are easier to measure directly in the given simulation. We will do so when calculating the relative speed-up for measuring the efficiency of simulation in the different experiments presented in this paper.

HETEROGENEOUS TEST ENVIRONMENT

Available Hardware Base

The following servers, workstations and PCs were available for our experiments at the Info-communications Laboratory of the Department of Telecommunications, Széchenyi István University. Note that this hardware base is somewhat larger than that of our previous paper (Lencse et al. 2013).

One Sun Server SunFire X4150
Two Quad Core Intel Xeon 2.83GHz CPU, 8GB DDR2 800MHz RAM, 160GB HDD, Gigabit Ethernet NICs
Altogether it means a homogeneous cluster of 8 nodes.

Three LS120 Blades form an IBM BladeCenter
Two Dual Core Opteron 280 2.4GHz CPU, 4GB DDR2 667MHz RAM, 73GB HDD, Gigabit Ethernet NIC
Altogether it means a homogeneous cluster of 12 nodes.

One Itanium Server HP Server RX2600
Two Intel Mckinley IA-64 900MHz CPU, 8GB RAM DDR 266MHz RAM, 36.4GB HDD, Gigabit Ethernet NIC
Although it has no serious computing power but it was found interesting because of its non-x86 architecture CPU.

Eleven Dell Precision 490 Workstations
Two Intel Xeon 5140 Dual Core 2.33GHz CPU, 4x1GB DDR2 533MHz RAM (quad channel), 80GB HDD, Gigabit Ethernet NICs
Altogether it means a homogeneous cluster of 44 nodes.

Eleven AMD PCs
AMD Athlon 64 X2 Dual Core 4200+ 2200MHz CPU, 2GB DDR2 667MHz RAM, 320 GB HDD, Gb. Eth. NIC
Altogether it means a homogeneous cluster of 22 nodes.

Four Old Intel PCs (P4)
Intel Pentium 4 HT 3GHz CPU, 512 DDR 400MHz RAM, 80 GB HDD, Fast Ethernet NIC.
Note that they use 32 bits CPUs.

Switches for Interconnection
- 3Com Baseline Switch 2948 SFP Plus (3CBLSG48)
- Cisco Intelligent Gigabit Ethernet Switch Module, 4 ports (Part Number 32R1894) in the BladeCenter
- D-Link EasySmart Switch DGS-1100-24

Software Environment

Operating Systems
Linux was used on all the computers. Sun Server and LS21 Blades: Ubuntu 12.04 LTS x86-64; Dell Precision 490 Workstations and AMD PCs: Debian Squeeze (x86_64); old Intel PC-s: Debian Squeeze (i386);

Cluster Software
OpenMPI 1.6.2 (x86_64 if not stated otherwise; i386 in some cases)

Discrete-Event Simulation Software
The widely used, open source OMNeT++ 4.2.2 discrete-event simulation environment (Varga and Hornig 2008) was chosen. It supports the conservative synchronization method (the Null Message Algorithm) since 2003 (Sekercioughlu et al. 2003). We also expect that because of the modularity, extensibility and clean internal architecture of the parallel simulation subsystem, the OMNeT++ framework has the potential to become a preferred platform for PDES research.

The Simulation Model

Bagrodia and Takai proposed the Closed Queueing Network for testing the performance of conservative parallel discrete-event simulation (Bagrodia and Takai 2000). OMNeT++ has a CQN implementation among its simulation sample programs. We have found this model perfect for our purposes thus it was used in our current paper as well as in our previous ones (Lencse and Varga 2010) and (Lencse et al. 2013). The below description of the model is taken from (Lencse and Varga 2010).

This model consists of M tandem queues where each tandem consists of a switch and k single-server queues with exponential service times (Figure 1, left).

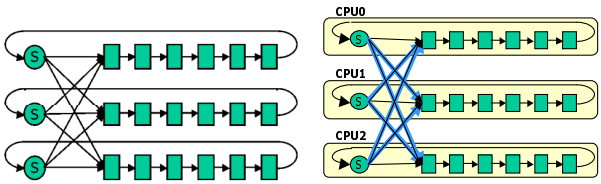

Figure 1. M=3 Tandem Queues with k=6 Single Server Queues in Each Tandem Queue – And its Partitioning

The last queues are looped back to their switches. Each switch randomly chooses the first queue of one of the tandems as destination, using uniform distribution. The queues and switches are connected with links that have nonzero propagation delays. The OMNeT++ model for CQN wraps tandems into compound modules.

To run the model in parallel, we assign tandems to different LPs (Figure 1, right). Lookahead is provided by delays on the marked links.

As for the parameters of the model, the preset values shipped with the model were used unless it is stated otherwise. Configuration B was chosen, the one that promised good speed-up. The main parameters of the CQN model were: M=24 tandem queues, k=50 queues in each tandem queue, exponential service time of the queues with expected value of 10 seconds, the delay between the tandem queues L=100 seconds and the length of the simulation was 10^6 seconds (in model time).

EXPERIMENTS AND RESULTS

Figure 2 shows the interconnection of the elements of our heterogeneous environment. Note that our experiments used different subsets of the elements.

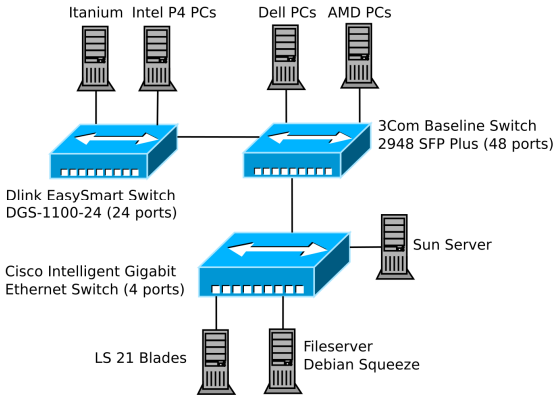

Figure 2. The elements of the Heterogeneous Execution Environment with their Interconnections

Further Validation of the Load Balancing Criterion

The load balancing criterion requires the benchmarking of the different CPU cores with the simulation model. The benchmarking was done using the CQN OMNeT++ sample model. All the experiments were performed 11 times and average and standard deviation was calculated. Unless stated otherwise, it was done so with all the following experiments, too. Table 1 shows the performance of the different CPU core types.

Table 1. The Performance of the Different 64-bit CPU Core Types (events/second)

Core Type	Sun	IBM	Dell	AMD	Itanium
Average	468 192	238 174	373 787	235 861	61 509
Std. Dev.	5 581	6 049	4 908	2 726	117

The load balancing was tested on the example of a minimal size heterogeneous system built up by one Sun core and one IBM core in (Lencse et al. 2013). Now it is validated on a similar system built up by one Dell core and one AMD core. The following series of experiments were performed: the CQN model built up by 24 tandem queues was cut into two segments: N and 24-N tandem queues were put into the segment executed by the AMD core and the Dell core, respectively, where N took its values form 1 to 23. The execution time was measured in all cases and the value of the relative speed up was calculated according to (8). Figure 3 shows the results. It can be seen that the partitioning was the best when 10 and 14 tandem queues were put into the segments executed by the AMD and the Dell computers, respectively. The exactly *performance proportional partitioning* would result in the assignment of 9.29 and 14.71 tandem queues. Thus the results of our experiments are in a good agreement with the computed "optimal partitioning".

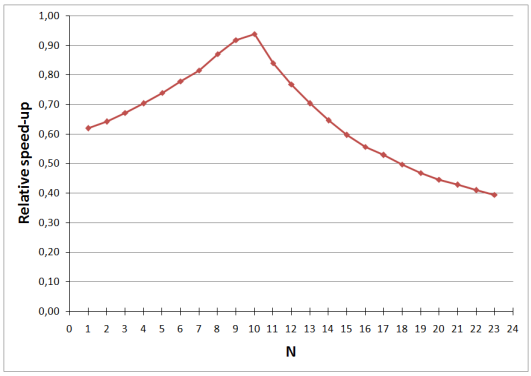

Figure 3. The Relative Speed-up of the Execution of the CQN Model in the Function of the Partitioning

A Test Including the Itanium Server

As the performance of the Itanium server is much less then that of all the others, it seems to be a reasonable question if it is worth using it at all? The following two experiments were conducted:

1. A cluster of one IBM and one AMD cores was used and 12 tandem queues were put on each of them.

2. The Itanium server was added to the above cluster. The number of the tandem queues was decreased by one both on the IBM and on the AMD cores and the two tandem queues were assigned to the Itanium server.

Note that the above partitioning in both cases was the best possible approximation of the performance proportional one.

Table 2 shows the parameters and the results of the experiments. The use of the Itanium server resulted in a 9.45% speed-up according to the conventional interpretation of speed-up. Is it a good result? To be able to judge the efficiency of the parallel simulation in the second experiment we calculated the relative speed-up according to (7) using the results of the first experiment as reference. We are really satisfied with the 0.9688 relative speed-up.

Table 2. The Parameters and the Results of the Experiments for the Itanium Server

Elements of the System	IBM+AMD	IBM+AMD+Itanium
Cumulated Performance (ev/sec)	474 035	535 544
Average Execution Time (sec)	335.97	306.96
Std. Dev. of Exec. Time	2.43	3.17
Speed-Up (Conventional)	(reference)	1.0945
Relative Speed-up	(reference)	0.9688

A Test Including the Old 32-bit Intel PC-s

Unfortunately, it is a feature of the OpenMPI, that if 32-bit libraries are used in one of the computers then they should be used on all the other ones so that the computers can communicate with each other. Thus we had to recompile everything in 32-bit mode for the 64 bit computers and benchmark them again in 32-bit mode. Table 2 shows the performance of the different CPU core types in 32-bit modes. The configuration script of OMNeT++ did not find the 32-bit MPI libraries under Debian, thus we could not test the Dell and AMD platforms. As we could not fix the problem due to lack of time, they were omitted from the 32-bit experiments.

Table 3. The Performance of the Different CPU Core Types in 32-bit mode (events/second)

Core Type	Sun	IBM	*P4*
Average	593 340	334 516	218 014
Std. Dev.	9 989	7 467	1 362

Note that the performance results of the Sun and IBM servers are significantly higher in 32-bit mode than in 64-bit mode. It is probably due to the fact that the integers and the pointers use twice as much memory space in 64-bit mode than in 32-bit mode and the longer programs can be less effectively cached.

The following two experiments were conducted:

1. The 8 cores of the Sun server and the 4 cores of one IBM Blade were used.

2. The four old Intel P4 computers were added to the above system.

The number of the tandem queues was increased in the CQN model from 24 to 240 to be able to utilize all the CPU cores. The value of the lookahead was increased from 100s to 1000s see its justification in (Lencse et al. 2013). The tandem queues were divided as close to the performance proportional partitioning as it was possible.

According to (Lencse et al. 2013), if the number of the CPU core types is denoted by NCT, the number and the performance of the CPU cores available from core type *i*

are denoted by N_i and P_i, respectively then the n_i number of the queues to be put into a segment executed by a core from type i should be:

$$n_i = \frac{240 \cdot P_i}{\sum_{j=1}^{NCT} P_i \cdot N_i} \qquad (9)$$

However, the number of the tandem queues per segments must be an integer, thus the division of the tandems could not be fully precise, some "roundings" and adjustments were done manually and there were differences made even between the load of the cores from the same core type so that the number all the tandems be exactly 240. Table 4 and 5 show the division of the tandems among the cores for the first and the second experiments.

Table 4. The Division of the 240 Tandem Queues among the Sun and IBM cores

Core type	P_i (ev/sec)	N_i	n_i	no. of cores	tandems /core	cumulated tandems
Sun	593 340	8	23.40	4	23	92
				4	24	96
IBM	334 516	4	13.19	4	13	52
No. of all the cores:		12	Number of all the tandems:			240

Table 5. The Division of the 240 Tandem Queues among the Sun, IBM and P4 cores

Core type	P_i (ev/sec)	N_i	n_i	no. of cores	tandems /core	cumulated tandems
Sun	593 340	8	20.47	8	20	160
IBM	334 516	4	11.54	4	12	48
P4	218 014	4	7.52	4	8	32
No. of all the cores:		16	Number of all the tandems:			240

Table 6 shows the parameters and the results of the experiments. The use of the four old Intel P4 PCs resulted in a 9.47% speed-up according to the conventional interpretation of speed-up. Again, as with the Itanium server, we calculated the relative speed-up according to (7) using the results of the first experiment as reference to be able to judge if the efficiency of the parallel simulation in the second experiment compared to the first one. We can be satisfied with the 0.9575 relative speed-up and mention just for comparison that even the speed-up caused be the P4 PCs was a little bit higher than the speed-up caused be the Itanium server, this a is bit less good result, as the efficiency is somewhat smaller now.

Table 6. The Parameters and the Results of the Experiments for the Old Intel P4 PCs

Elements	Sun+IBM	Sun+IBM+4xP4
Cumulated Performance (ev/sec)	6 084 784	6 956 480
Average Execution Time (sec)	1 327.31	1 212.51
Std. Dev. of Exec. Time	34.91	47.15
Speed-Up (Conventional)	(reference)	1.0947
Relative Speed-up	(reference)	0.9575

Note that we calculated the *relative efficiency* in both cases because we tested if it was worth using the Itanium server or the old Intel P4 PCs. It will be done differently in the test of the largest available system.

A Test of the Largest Available System

The maximum possible number of 64-bit cores were included into the heterogeneous system. The number of the tandem queues and the lookahead were increased to 480 and 2000s, respectively. First, the partitioning was done according to the old performance values that were measured during the simulation of the system built up by 24 tandem queues and using 100s lookahead. Table 7. shows the partitioning.

Table 7. The Division of the 480 Tandem Queues Among the 64-bit Cores Using the Old P_i Values from Table 1.

Core type	P_i (ev/sec)	N_i	n_i	no. of cores	tandems /core	cumulated tandems
Sun	468 192	8	7.94	8	8	64
IBM	238 174	12	4.04	12	4	48
Dell	373 787	44	6.34	29	6	174
				15	7	105
AMD	235 861	22	4.00	22	4	88
Itanium	61 509	1	1.04	1	1	1
No. of all the cores:		87	Number of all the tandems:			480

The execution time of the parallel simulation was 159.80 seconds with 2.84 standard deviation. To be able to determine its efficiency, we needed either a sequential execution time produced by any of the CPU cores or the total number of events in the sequential simulation. As the model was large we executed the sequential simulation only for a 10^5 seconds long model time interval that is only one tenth of the 10^6 model time simulated by the parallel execution. (The execution time and event number values for 10^6 seconds model time sequential simulation can be easily extrapolated the by multiplying the results with 10.) We executed the sequential simulation on all the CPU types 11 times except for Itanium, where it was executed only once. The number of all the events was approximately 170.34 million in all cases (that means 1,703.4 million events for 10^6 model time seconds). The execution time values are shown in table 8. This table also shows the speed-up values (calculated by multiplying the sequential execution time values by 10) against the sequential execution by the given CPU core types. The next line of the table shows the recalculated P_i performance values. They are much smaller than those in table 1 due to the effect called "vacationing jobs" in (Lencse and Varga 2010). Shortly summarized the effect: the long delay lines are "storing" some jobs (events) that are missing from the elementary queues of the tandem queues thus the number of events per real-time seconds is significantly decreased. The new P_i values are not even proportional with the old ones. The next line of the table shows the quotient of the new and the old P_i values for the different CPU core types: they are really different. (For this reason, we tested also another

partitioning using the new P_i values.) For computing the relative speed-up, the cumulative performance was computed from the new P_i values and we got: P_c=17 929 579 ev/sec.

The relative speed-up was calculated according to (8) as:

$$r_h = \frac{1.7034 \cdot 10^9 ev}{159.8 sec \cdot 17929579 ev / sec} \approx 0.5945 \quad (10)$$

This value is not at all bad for such a big and inhomogeneous cluster.

Table 8. Execution Time of the 10^5s (model time) Long Sequential Simulation and Other Derived Values

Core Type	Sun	IBM	Dell	AMD	Itanium
Avg. Ex. time (s)	593.20	960.03	778.65	972.69	6837.44
Std. Dev. of E. T.	5.38	31.40	3.87	7.49	-
Speed-up	37.1	60.1	38.7	60.9	427.9
new P_i (ev/sec)	287 154	177 432	218 763	175 122	24 907
new P_i / old P_i	0.6133	0.7450	0.5853	0.7425	0.4049

The system was repartitioned according to the new P_i values, see table 9. Note that even though the number of available cores was 87, only 86 of them were actually used as no tandem queues were put to the Itanium server.

Table 9. The Division of the 480 Tandem Queues Among the 64-bit Cores Using the New P_i Values from Table 8.

Core type	P_i (ev/sec)	N_i	n_i	no. of cores	tandems /core	cumulated tandems
Sun	287 154	8	7.69	8	8	64
IBM	177 432	12	4.75	12	5	60
Dell	218 763	44	5.85	44	6	264
AMD	175 122	22	4.68	18	4	72
				4	5	20
Itanium	24 907	1	0.67	1	0	0
No. of all the cores:		87	Number of all the tandems:			480

The execution of the simulation gave excellent results. The average execution time was 108.12 seconds with 2.34 standard deviation. The speed-up values calculated against the four actually used CPU core types are shown in table 10.

Table 10. Speed-up of the Second Heterogeneous System

Core Type	Sun	IBM	Dell	AMD
Speed-up	54.9	88.8	72.0	90.0

As the Itanium server was not used, its performance was left out from the cumulative performance: P_c=17 904 672 ev/sec. For relative speed-up, we got:

$$r_h = \frac{1.7034 \cdot 10^9 ev}{108.12 sec \cdot 17904672 ev / sec} \approx 0.8799 \quad (11)$$

This is excellent for such a big and inhomogeneous cluster.

CONCLUSIONS

Some of our earlier defined criteria (load balancing and coupling factor) for the possible good speed-up of parallel discrete event simulation in heterogeneous execution environments were summarized. A novel extension of the relative speed-up for heterogeneous systems was introduced.

The operation of the criteria was justified by several experiments with different size and type of heterogeneous systems.

The extension of the definition of the relative speed-up to heterogeneous systems proved to be an appropriate tool for the evaluation of the speed-up of parallel discrete event simulation in heterogeneous execution environments.

We conclude that the recommended methods are worth using and further studying.

ACKNOWLEDGEMENT

This research was supported by the TÁMOP-4.2.2/B-10/1-2010-0010 project and by the Széchenyi István University (15-3202-08).

REFERENCES

Bagrodia, R. L. and M. Takai. 2000. "Performance evaluation of conservative algorithms in parallel simulation languages" *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11. No 4. 395-411.

Kunz, G. 2010. "Parallel Discrete Event Simulation" In *Modeling and Tools for Network Simulation* K. Wehrle, M. Günes and J. Gross (Eds.). Springer-Verlag, Berlin 2010. ISBN 978-3-642-12330-6

Lencse, G. and A. Varga. 2010. "Performance Prediction of Conservative Parallel Discrete Event Simulation". *Proceedings of the 2010 Industrial Simulation Conference (ISC'2010)* (Budapest, Hungary, 2010. June 7-9.) EUROSIS-ETI, 214-219.

Lencse, G., I. Derka and L. Muka. 2013. "Towards the efficient simulation of telecommunication systems in heterogeneous execution environments", accepted for the *36th International Conference on Telecommunications and Signal Processing (TSP), July 2-4, 2013, Rome, Italy*.

Sekercioglu, Y. A., Varga, A. and Egan, G. K. Egan. 2003. "Parallel Simulation Made Easy with OMNeT++". *Proceedings of the European Simulation Symposium (ESS 2003)*, Oct. 26-29, 2003, Delft, The Netherlands.

Varga, A., Y. A. Sekercioglu and G. K. Egan. 2003. "A practical efficiency criterion for the null message algorithm". *Proceedings of the European Simulation Symposium (ESS 2003)*, (Oct. 26-29, 2003, Delft, The Netherlands.) SCS International, 81-92.

Varga, A. and Hornig, R. 2008. "An overview of the OMNeT++ simulation environment", *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. March 7, 2008, Marseille, France.