PERFORMANCE EVALUATION OF SMP PARALLEL COMPUTERS

[a]PETER HANULIAK

Dubnica Technical Institute, Sladkovicova 533/20, 018 41 Dubnica nad Vahom, Slovakia
email: phanuliak@gmail.com

This work was done within the project Modelling, optimisation and prediction of parallel computers and algorithms at University of Žilina supervised by Prof. Ing. I. Hanuliak, CSc. The authors, as the co-workers of this project, gratefully acknowledge the help of all co-workers in this project.

Abstract: Recent trend in parallel computers is to use networks of workstations (NOW) as a cheaper alternative of parallel computer in comparison to in the world used massively parallel multiprocessors or supercomputers. As workstations are used mainly powerful personal computer (PC) or PC´s based symmetrical multiprocessors (SMP).
This paper is devoted to performance evaluation of parallel computers based on SMP, describes the typical parallel computers and analyses basic concepts of performance evaluation. Then it demonstrates how to apply queuing theory results to model computing nodes of parallel computer or parallel computers based on SMP systems.

Keywords: parallel computer, SMP, NOW, Grid, performance evaluation, queuing theory, SPEC tests.

1 Introduction

For the actual parallel computers there are dominating various forms of the parallel principles (Pipeline, super pipeline, cashes etc.). Recent trends in high performance computing (HPC) use network of workstations (SMP, NOW) as a cheaper alternative of parallel computer in comparison to used massively parallel multiprocessors [7,14]. A workstation in NOW can be also a parallel system based on symmetrical multiprocessors (SMP). In such parallel computer workstations are connected through widely used communication standard networks and co-operate to solve one large problem. All existed parallel computers build some form of virtual parallel computer according Fig. 1.

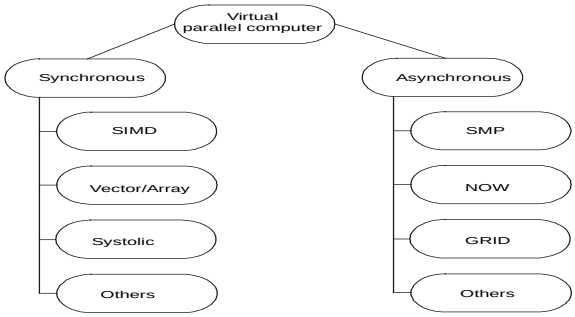

Fig. 1. System´s classification of parallel computers.

2 Classification of parallel systems

It is very difficult to classify the various existed forms of parallel computers. But from a system point we can divide parallel computers [1, 9] to the two following different groups

- synchronous parallel computers. The dominant system property is the concentration on massive data parallelism. The typical examples of synchronous parallel computers illustrate Fig. 1. on its left side. Some of used parallel principles are applied in actually modern parallel computers for example in a form of SIMD (Single instruction multiple data) instructions within their computing nodes [5, 18]
- asynchronous parallel computers. They are composed of a number of fully independent computing nodes (processors, cores or computers). To this group belong mainly various forms of computer networks (cluster), network of workstation (NOW|) or more integrated network of NOW networks (Grid). The typical examples of asynchronous parallel computers illustrate Fig. 1. on its right side. Typical computing node of actual parallel computer consists on SMP.

3 Architecture of modern parallel computers

3.1 Symmetrical multiprocessor system

Symmetrical multiprocessor system is a multiple using of the same processor or cores which are implemented on motherboard in order to increase whole performance of such parallel computer. Typical characteristics are following

- each computing node can access main shared memory
- I/O channels are allocated to computing nodes according their demands
- integrated operation system coordinates cooperation of whole multiprocessor.

Typical concept of such multiprocessor illustrates Fig. 2.

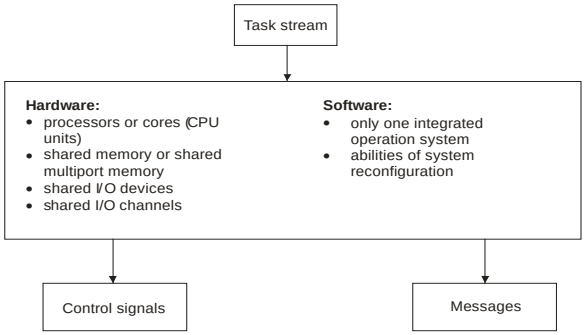

Fig.2. Typical characteristics of SMP multiprocessor systems.

Typical practical example of eight multiprocessor systems Intel Xeon illustrates Fig. 3.

3.2 Network of workstations

There has been an increasing interest in the use of networks of workstations, which are connected together by high speed networks [17, 19] for solving large computation intensive problems. This trend is mainly driven by the cost effectiveness of such systems as compared to massive multiprocessor systems (Supercomputers). Network of workstations (NOW) [11] has become a widely accepted form of high performance computing (HPC). Each workstation in a NOW is treated similarly to a processing element in a multiprocessor system. However, workstations are far more powerful and flexible than processing elements in conventional multiprocessors (Supercomputers). Network of workstations (NOW) [11] has become a widely accepted form of high performance computing (HPC). Each workstation in a NOW is treated similarly to a processing element in a multiprocessor system. However, workstations are far more powerful and flexible than processing elements in conventional multiprocessors (Supercomputers). To exploit the parallel processing capability of a NOW, an application algorithm must be paralleled [21]. A way how to do it for an application problem builds its decomposition strategy.

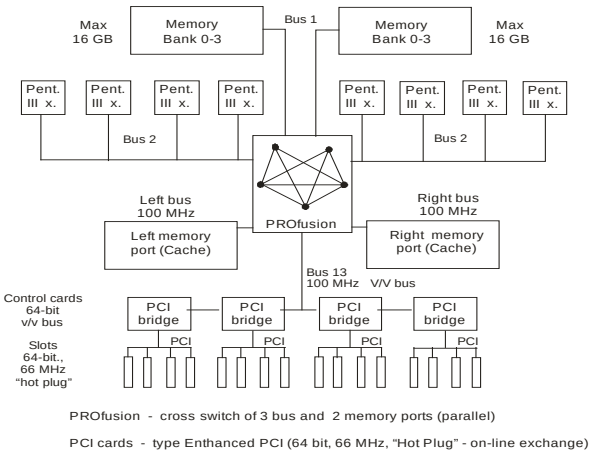

Fig. 3. Architecture of 8-Intel multiprocessor.

3.3 Grid

In early years of twenty-first century, high speed, highly reliable Internet connectivity is as commonplace as electricity in commercial, governmental, and research/educational institutions, and individual consumers are not far behind. This observations led researchers, in the mid – 1990's, to propose the notion of computational Grids [22], where computing resources would be available as universally and easily as for example electric power enabled the utilization of resources on external computing devices over commodity networks.

Grid systems are expected to operate on a wider range of other resources as processors (CPU), like storages, data modules, network components, software (typical resources) and atypical resources like graphical and audio input/output devices, sensors and so one (Fig. 5.). All these resources typically exist within nodes that are geographically distributed, and span multiple administrative domains. The virtual machine is constituted of a set of resources taken from a resource pool. It is obvious that existed HPC parallel computers (supercomputers etc.) could be a member of such Grid systems too [20].

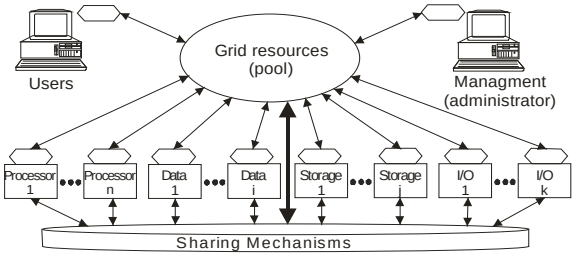

Fig. 4. Architecture of Grid node.

3.4 Metacomputing

This term define massive computational Grid with following basic characteristics

- wide area network integrated free computing resources. It is a massive number of interconnected networks, which are connected through high speed connected networks during which time whole massive system is controlled with network operation system, which makes an illusion of powerful computer system (virtual supercomputer)
- grants a function of metacomputing that means computing environment, which enables to individual applications a functionality of all system resources
- system combines distributed parallel computation with remote computing from simple user workstation.

4 The role of performance

Quantitative evaluation and modelling of hardware and software components of parallel systems are critical for the delivery of high performance. Performance studies apply to initial design phases as well as to procurement, tuning and capacity planning analysis. As performance cannot be expressed by quantities independent of the system workload, the quantitative characterization of resource demands of application and of their behaviour is an important part of any performance evaluation study. Among the goals of parallel systems performance analysis are to assess the performance of a system or a system component or an application, to investigate the match between requirements and system architecture characteristics, to identify the features that have a significant impact on the application execution time, to predict the performance of a particular application on a given parallel system, to evaluate different structures of parallel applications.

4.1 Performance evaluation of parallel computers

The study of the performance of computers attempts to understand and predict the time dependent behaviour of parallel computers. It can be broadly divided into two areas – modelling and measurement. These can be further divided by objective and by technique. These two apparently disjoint approaches are in fact mutually dependent and are both required in any practical study of the performance of a real or planned system. The overall process of estimating or predicting the performance of a computer system is sometimes referred to as performance analysis or performance evaluation.

5 Performance evaluation methods

Several fundamental concepts have been developed for evaluating parallel computers. Tradeoffs among these performance factors are often encountered in real-life applications. To the performance evaluation we can use following methods

- analytical methods
  - application of queuing theory results [3, 6, 12]
  - asymptotic (order) analyze [2, 8]
  - Petri nets [4]
- simulation methods [15]
- experimental measurement
  - benchmarks [13]
  - direct parameter measuring [16].

When we solve a model we can obtain an estimate for a set of values of interest within the system being modelled, for a given set of conditions which we set for that execution. These conditions may be fixed permanently in the model or left as free variables or parameters of the model, and set at runtime.

5.1 Analytic techniques

There is a very well developed set of techniques which can provide exact solutions very quickly, but only for a very restricted class of models. For more general models it is often possible to obtain approximate results significantly more quickly than when using simulation, although the accuracy of these results may be difficult to determine. The techniques in question belong to an area of applied mathematics known as queuing theory, which is a branch of stochastic modelling. Like simulation, queuing theory depends on the use of computers to solve its models quickly.

5.2 Petri nets

A Petri net is essentially an extension of a finite state automaton, to allow by means of tokens several concurrent threads of activity to be described in one representation. It is essentially a graphical description, being a directed graph with its edges defining paths for the evolution of a system's behaviour and its nodes or vertices being of two sorts, places and transitions. An example of simple Petri net illustrates Fig. 8. Places S11, S12,

S13, S21 and S22 represent states, transition t represents event and existed vertigos are connected through oriented edges. Points in places define tokens. All incoming edges to a place must come from a transition and vice versa. Tokens are held in places and when all the input places to a transition are marked, i.e. have at least one token, that transition is enabled and fires, depositing a token in each of its output places.

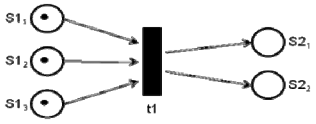

Fig. 5. Simple Petri network.

There are a number of extensions to these simple place/transition nets, mostly to increase the ease of describing complex systems. The most widely used is to define multiplicities for the edges, which define how many tokens flow down an edge simultaneously.

5.3 Simulation

Simulation is the most general and versatile means of modelling systems for performance estimation. It has many uses, but its results are usually only approximations to the exact answer and the price of increased accuracy is much longer execution times. Numerical techniques vary in their efficiency and their accuracy. They are still only applicable to a restricted class of models. Many approaches increase rapidly in their memory and time requirements as the size of the model increases. To reduce the cost of a simulation we may resort to simplification of the model which avoids explicit modelling of many features, but this increases the level of error in the results. If we need to resort to simplification of our models, it would be desirable to achieve exact results even though the model might not fully represent the system. At least then one source of inaccuracy would be removed. At the same time it would be useful if the method could produce its results more quickly than even the simplified simulation. Thus it is important to consider the use of analytic and numerical techniques before resorting to simulation.

5.4 Experimental modelling

Benchmark
We divide used performance tests as following
- classical
  - Peak performance
  - Dhrystone
  - Whetstone
  - LINPAC
  - Khornestone
- problem oriented tests (Benchmarks)
  - specialised tests
  - SPEC tests.

SPEC ratio
SPEC (Standard Performance Evaluation Corporation - www.spec.org) defined one number to summarise all needed tests for integer number. Execution times are at first normalised through dividing execution time by value of reference processor (chosen by SPEC) with execution time on measured computer (user application program). The achieved ratio is labelled as SPEC ratio, which has such advantage that higher numerical numbers represent higher performance, that means that SPEC ratio is an inversion of execution time. INT 20xx (xx means year of latest version) or CFP 20xx result value is produced as geometric average value of all SPEC ratios.

5.5 Experimental measurement

Evaluating system performance via experimental measurements is a very useful alternative for parallel systems and algorithms. Measurements can be gathered on existing systems by means of benchmark applications that aim at stressing specific aspects of the parallel systems and algorithms. Even though benchmarks can be used in all types of performance studies, their main field of application is competitive procurement and performance assessment of existing systems and algorithms. Parallel benchmarks extend the traditional sequential ones by providing a wider a wider set of suites that exercise each system component targeted workload.

6 Application of queuing theory systems

The basic premise behind the use of queuing models for computer systems analysis is that the components of a computer system can be represented by a network of servers (or resources) and waiting lines (queues). A server is defined as an entity that can affect, or even stop, the flow of jobs through the system. In a computer system, a server may be the CPU, I/O channel, memory, or a communication port. Awaiting line is just that: a place where jobs queue for service. To make a queuing model work, jobs are inserted into the network. A simple example, the single server model, is shown in Fig. 9. In that system, jobs arrive at some rate, queue for service on a first-come first-served basis, receive service, and exit the system. This kind of model, with jobs entering and leaving the system, is called an open queuing system model.

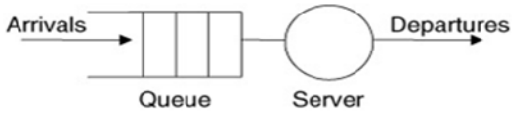

Fig. 6. Queuing theory based model.

6.1 Kendall classification

In addition to the notation for the quantities associated with queuing systems, it is also useful to introduce a notation for the parameters of a queuing system. The notation we will use here is known as the Kendall notation in its extended form as A/B/m/K/L/Z [1, 2], where
- A means arrival process definition
- B means service time distributions
- m is number of identical servers
- K means maximum number of customers allowed in the system (default = ∞)
- L is number of customers allowed to arrive (default = ∞)
- Z means discipline used to order customers in the queue (default = FIFO).

Three symbols used in a Kendall notation description also have some standard definitions. The more common designators for the A and B fields are as following
- M means Markovian (exponential) service time or arrival rate
- D defines deterministic (constant) service time or arrival rate
- G means general service time or arrival rate.

The service discipline used to order customers in the queue can be any of a variety of types, such as first-in first-out (FIFO), last in first out (LIFO), priority ordered, random ordered and others.

6.2 Little's law

One of the most important results in queuing theory applications is Little's law. This was a long standing rule of thumb in analysing queuing systems, but gets its name from the author of the first paper which proves the relationship formally. It is applicable to the behaviour of almost any system of queues, as

long as they exhibit steady state behaviour. It relates the mean number of customers in the system to mean time spent in the system by each customer, for a given arrival rate. Little's law says

$$E(q) = \lambda \cdot E(t_q)$$

The main parameters in queuing theory application are as following

- λ - arrival rate at entrance to a queue
- m - number of identical servers in the queuing system
- ρ - traffic intensity (dimensionless coefficient of utilisation)
- q - random variable for the number of customers in a system at steady state
- w - random variable for the number of customers in a queue at steady state
- E (t_s) - the expected (mean) service time of a server
- E (q) - the expected (mean) number of customers in a system at steady state
- E (w) - the expected (mean) number of customers in a queue at steady state
- E (t_q) - the expected (mean) time spent in system (queue + servicing) at steady state
- E (t_w) - the expected (mean) time spent in the queue at steady state.

6.3Poisson distribution

The Poisson distribution models a set of totally independent events as a process, where each event is independent of all others. Knowledge of past events does not allow us to predict anything about future ones, except that we know the overall average, the Poisson distribution represents the likelihood of one of a given range of numbers of events occurring within the next time interval. The definition of Poisson distribution probabilities p_i is as

$$p_i = \frac{\lambda^{\,i}}{i!} e^{-\lambda}$$

, where the parameter λ is defines as the average number of successes during the interval.

6.4Exponential distribution

If the Poisson distribution represents the likely number of independent events to occur in the next time period, the exponential distribution is its converse. It represents the distribution of inter - arrival times for the same arrival process. Its mean is inter - event time, but it is often expressed in terms of the arrival rate, which is 1/inter - arrival time. Exponential distribution function p_i and its mean value E (t_s) are

$$p_i = \mu \; e^{-\mu t} \qquad E(t_s) = 1/\mu$$

The Poisson distribution models a set of totally independent events as a process, where each event is independent of all others. It is not the same as a uniform distribution. Where knowledge of past events does not allow us to predict anything about future ones, except that we know the overall average, the Poisson distribution represents the likelihood of one of a given range of numbers of events occurring within the next time interval.

6.5M/M/m queue model

The basic needed derived relations for M/M/m queue model (Fig.7) are following.

Average number of customer in the queue

$$\rho = \frac{\lambda}{\mu \,.\, m} < 1 \qquad E(w) = \frac{\rho\,(\rho\, m)^m}{m!\,(1-\rho)^2}\, p_0$$

where the probability

$$p_0 = \left[\sum_{i=0}^{m-1} \frac{(m\,\rho)^i}{i!} + \frac{(m\,\rho)^m}{m!} \frac{1}{1-\rho} \right]^{-1}$$

The further parameters E(t_q) and E(t_w) we can derive using the Little´s law.

Fig. 7. M/M/m (m=3) model of multiprocessor or multicore systems.

6.6 M / D / m queue model

In this queue model traffic intensity ρ and the service time are as

$$\rho = \frac{\lambda}{\mu \,.\, m} < 1 \qquad E(t_s) = \frac{1}{\mu} = constant.$$

For the mean number of customers in the queue we have chosen approximate relation [11]

$$E(t_w)[M/D/m] \doteq \left[1 + (1-\rho_i)\cdot(m-1)\frac{\sqrt{45\,m}-2}{16\,\rho_i\, m} \cdot \frac{E(t_w)[M/D/1]}{E(t_w)[M/M/1]} E(t_w)[M/M/m] \right]$$

Average number of customer in the system is given as

$$E(q) = E(w) + m\,\rho$$

The further parameters E (t_q) and E (t_w) we can derive using the Little´s law.

7 Results

7.1 Application of THO models

We have modelled multiprocessor system as M/M/m and M/D/m queuing models, where we were supposed parallel activity of used processors or cores. The differences between multiprocessor or multicore are in their performance (input parameters).

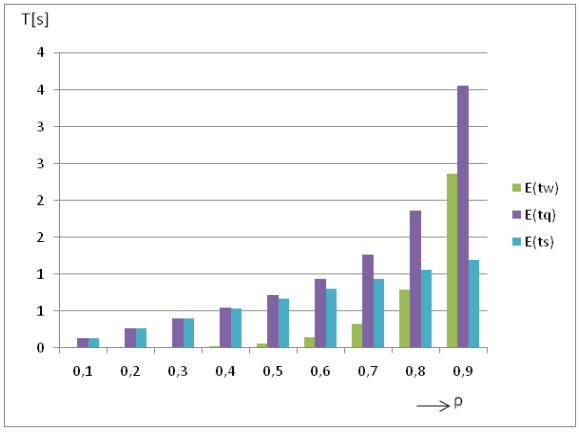

Fig.8 Mean values E(t_w),E(t_q), E(t_s) for M/M/4 model (λ=3).

Graphics illustration at Fig. 9 compare queuing models M/M/4 and M/D/4 (λ=3, ρ = λ . E (t_s) / 4) for average time in system (queue + servicing).

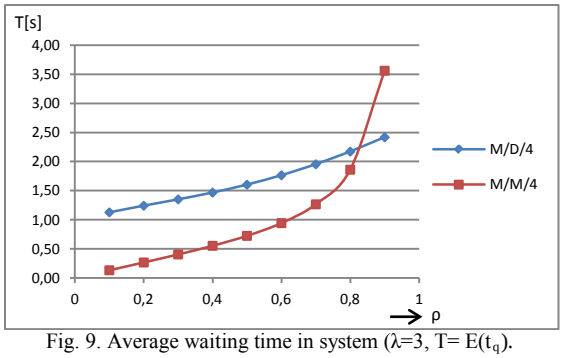

Fig. 9. Average waiting time in system (λ=3, T= $E(t_q)$.

7.2 Spec test ratio

We have been performed various tests (benchmarks) to verify analytical results. We illustrate some achieved results using Spec test ratio to compare performance of following processors

- AMD Athlon X2 6000+
- Intel Core2Duo E7300
- Intel i7-950.

At Fig. 10. we illustrated tested results for mentioned processors using SPEC tests. We have chosen SPEC tests because these tests are from various really applications in order to come to more universal tested results. To compare any computers using SPEC ratios test we preferred to use geometric mean value therefore, it defines the same relative value regardless of used normalised reference computer. If we were evaluating normalised values using arithmetic mean value results would be depended from the type of used normalised computer. According our expectations processor Intel i7-950 achieved the highest SPEC ratio value.

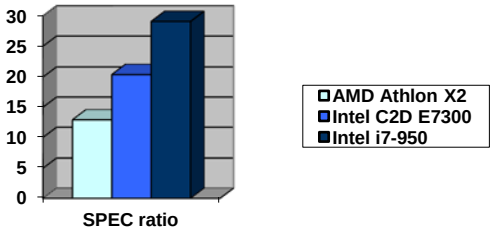

Fig. 10. Comparison of tested processors

8 Conclusions

Performance evaluation of computers generally used to be a very hard problem from birthday of computers. It was very hard to apply any analytical methods to performance evaluation of sequential computers because of their high number of not predictable parameters. Any analytical method is to be preferred in comparison with other possible methods, because of transparent using of achieved analytical results.

Actually dominant using of multiprocessor and multicore computers opens more possibilities to apply a queuing theory results to analyse their performance. This implies the knowledge that outputs from more than single processor approximate closer assumed Poisson distribution. Second the outputs from one multiprocessor system (workstation) are going to another multiprocessor system (neighbouring workstation) in dominant parallel computers (SMP, NOW, Grid).

The achieved results we can apply to performance modelling of multiprocessors or multicores) in following cases (input parameter $\rho = \lambda . E(t_s) / m$, for m=1 we can model performance of single processor)

- running of unbalanced parallel processes where λ is a parameter for incoming parallel processes with their exponential service time distribution as $E(t_s) = 1/\mu$ (M/M/m)
- running of parallel processes (λ parameter for incoming parallel processes with their deterministic service time $E(t_s) = 1/\mu$ = constant). The same deterministic servicing time is a very good approximation for all optimal balanced parallel processes (M/D/m)
- in case of using M/D/m model we can consider λ parameter also for incoming computer instructions with their average service time for instruction t_i, where $E(t_s) = 1/\mu = t_i$=constant.

We have choose the analysed models M/M/m, M/D/1, and M/D/m from this causes

- to finish performance analysis of networks of queuing theory system [10] we need results of chosen queuing theory systems
- we need their results to compute approximation relation for M/D/m
- based on models in this article together with application of Jackson theorem [10] we are able to analyse also more complicated network of NOW networks (Grid etc.).

Literature:

1. Abderazek A. B., *Multicore systems on-chip – Practical Software/Hardware design*, 200 pp., August 2010, Imperial college press
2. Arora S., Barak B., *Computational complexity - A modern Approach*, Cambridge University Press, 573 pp., 2009
3. Dattatreya G. R., *Performance analysis of queuing and computer network*, 472 pp., University of Texas, Dallas, USA, 2008
4. Desel J., Esparza J., *Free Choice Petri Nets*, 256 pages, 2005, Cambridge University Press, United Kingdom
5. Dubois M., Annavaram M., Stenstrom P., *Parallel Computer Organisation and Design*, 560 pages, 2012
6. Gelenbe E., *Analysis and synthesis of computer systems*, 324 pages, April 2010, Imperial College Press
7. Hager G., Wellein G., *Introduction to High Performance Computing for Scientists and Engineers*, 356 pages. July 2010
8. Hanuliak P., *Analytical method of performance prediction in parallel algorithms*, The Open Cybernetics and Systemics Journal, July 2012
9. Hanuliak J., Hanuliak M., *Analytical modelling of distributed computer systems*, NOW, In Proc.: TRANSCOM 2005, pp. 103-110, Žilina, Slovakia
10. Hanuliak M., *Performance evaluation of coupled parallel computers*, AD ALTA, 2013, submitted
11. Hanuliak M., Hanuliak I., *To the correction of analytical models for computer based communication systems*, Kybernetes, Vol. 35, No. 9, pp. 1492-1504, 2006, UK
12. Hillston J., *A Compositional Approach to Performance Modelling*, *University of Edinburg*, 172 pages, 2005, Cambridge University Press, United Kingdom
13. John L. K., Eeckhout L., *Performance evaluation and benchmarking*, CRC Press, 2005
14. Kirk D. B., Hwu W. W., *Programming massively parallel processors*, Morgan Kaufmann, 280 pages, 2010
15. Kostin A., Ilushechkina L., *Modelling and simulation of distributed systems*, 440 pages, Jun 2010, Imperial College Press
16. Lilja D. J., *Measuring Computer Performance*, 280 pages, 2005, University of Minnesota, Cambridge University Press, United Kingdom
17. Kushilevitz E., Nissan N., *Communication Complexity*, 208 pages, 2006, Cambridge University Press, United Kingdom
18. Patterson D. A., Hennessy J. L., *Computer Organization and Design (4th edition)*, Morgan Kaufmann, 914 pages, 2011
19. Peterson L. L., Davie B. C., *Computer networks – a system approach*, Morgan Kaufmann, 920 pages, 2011
20. Resch M. M., *Supercomputers in Grids*, Int. Journal of Grid and HPC, No.1, p 1 - 9, January- March 2009.

21. Vaníček, J., Papík, M., Pergl, R. a Vaníček T., *Theoretical principles of informatics* (In Czech), 431 pages, Kernberg Publishing, 2007, Prag , Czech republic
22. Wang L., Jie Wei., Chen J., *Grid Computing: Infrastructure*, Service, and Application, 2009, CRC Press.

Primary Paper Section: I

Secondary Paper Section: IN, JD, BA