American Journal of Networks and Communications
2014; 3(5-1): 57-69
Published online July 30, 2014 (http://www.sciencepublishinggroup.com/j/ajnc)
doi: 10.11648/j.ajnc.s.2014030501.15
ISSN: 2326-893X (Print); ISSN: 2326-8964 (Online)

Modeling of single computing nodes of parallel computers

Peter Hanuliak, Michal Hanuliak

Dubnica Technical Institute, Sladkovicova 533/20, Dubnica nad Vahom, 018 41, Slovakia

Email address:
phanuliak@gmail.com (P. Hanuliak), michal.hanuliak@gmail.com (M. Hanuliak)

To cite this article:
Peter Hanuliak, Michal Hanuliak. Modeling of Single Computing Nodes of Parallel Computers. *American Journal of Networks and Communications.* Special Issue: Parallel Computer and Parallel Algorithms. Vol. 3, No. 5-1, 2014, pp. 57-69.
doi: 10.11648/j.ajnc.s.2014030501.15

Abstract: The paper describes analytical modeling for single computing nodes of parallel computers. At first the paper describes very shortly the developing steps of parallel computer architectures and then he summarized the basic concepts for performance evaluation. To illustrate theoretical evaluation concepts the paper considers in its experimental part the achieved results on concrete analyzed examples and their comparison. The suggested analytical models consider for single computing node based on processor or core and SMP modeling of own computer node´s activities and node´s communication channels of performed data communications within computing node queuing theory systems M/D/m or M/D/. In case of using SMP parallel system as node computer the suggested models consider for own node's activities M/M/m or M/D/m queuing theory systems. Although we are able to use other more complicated queuing theory systems we prefer modeling with mentioned models because achieved results for these models we can use in decomposed modeling of coupled computing nodes as network of workstations (NOW) or network of massive NOW modules (Grid). The achieved results of the developed analytical models we have compared with the results of tested computing nodes with other alternative evaluation method based on suitable benchmarks to verify developed analytical models. The developed analytical models could be used under various ranges of input analytical parameters, which influence the architecture of analyzed computing nodes which are interested for the praxis.

Keywords: Parallel Computer, Computing Node, Network of Workstation (NOW), Grid, Analytical Modeling, Queuing Theory, Performance Evaluation, Queuing Theory System, Benchmark

1. Developing Periods of Parallel Computers

In the first period of parallel computers between 1975 and 1995 dominated scientific supercomputers, which were specially designed for the high performance computing (HPC). These parallel computers have been mostly used computing models based on data parallelism. Those systems were way ahead of standard common computers in terms of their performance and price. Increased processor performance was caused through massive using of various parallel principles in all forms of produced processors. Parallel principles were used so in single PC's and workstations (scalar or super scalar pipeline, symmetrical multiprocessor systems SMP) [5] so as on extreme powerful PC as in various connected network of workstations (NOW, cluster). Gained experience with the implementation of parallel principles and intensive extensions of computer networks, leads to the use of connected computers for parallel solution. This period we can name as the second developing period. Their large growth since 1980 have been stimulated by the simultaneous influence of three basic factors [10, 29]

- high performance processors and computers
- high speed interconnecting networks
- standardized tools to development of parallel algorithms (OpenMP, MPI, Java).

Developing trends are actually going toward building of wide spread connected NOW networks with high computation and memory capacity (Grid). Conceptually Grid comes to the definition of meta computer [20], where meta computer could be understood as big computer network consisting on massive number of computing nodes, memories and other needed resources together creating an illusion of one single powerful supercomputer. These high integrated forms of NOW's create various Grid systems or meta computers we could define as the third period of

parallel computers.

2. Basic Modules of Parallel Computers

Basic technical components of parallel computers illustrate Fig. 1 as follows

- modules of processors, cores of mix of them
- modules of computers (Sequential, parallel)
- memory modules
- input/output (I/O) modules.

These modules are connected through intern high speed communication networks (within concrete module) and extern among used computing modules via high speed communication networks [25, 35].

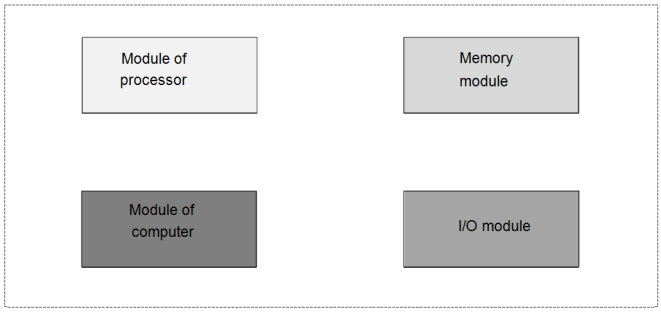

Figure 1. *Basic building modules of parallel computers.*

2.1. Classification of Parallel Computer

It is very difficult to classify all to this time realized parallel computers. The basic classification is from the point of realized memory as follows

- parallel computers with shared memory (multiprocessors, multicores)
- parallel computers with distributed memory (mainly based on computer networks)
- others.

2.1.1. Parallel Computers with Shared Memory

We can name realized parallel computers with shared memory as follows

- switched system
- multi bus system
- vector processor
- array processor
- associative processor
- transputer
- pipeline system
- systolic system
- wave front array system
- cellular system
- n-dimension cubes
- algorithm structured
- supercomputers
- connection machines
- super reliable
- neural networks.

2.1.2. Parallel Computers with Distributed Memory

To this group of parallel computers belong mainly parallel computers based on some form of network connection as follows

- localcomputer networks (LAN)
  - network of workstations (NOW)
  - PC farms, clusters
  - others
- wide area networks (WDN)
  - network of NOW networks (Grid)|
  - meta computers (Internet)
  - others.

2.2. Classification from the Point of Programmer

But from the point of programmer we divide them to the two following different groups

- synchronous parallel architectures. These are used for performing the same or very similar process (independent part) on different sets of data (data parallelism) in active computing nodes of parallel system. They are often used under central control that means under the global clock synchronization (vector, array system etc.) or a distributed local control mechanism (systolic systems etc.). This group consists mainly of parallel computers (centralized supercomputers) with any form of shared memory. Shared memory defines typical system features and in some cases can in considerable measure reduces developing of some parallel algorithms. To this group belong actually dominated parallel computers based on multiply cores, processors or mix of them and most of realized massive parallel computers (classic supercomputers) [5, 31]. Basic common characteristics are as following
  - shared memory (at least a part of memory)
  - using shared memory for communication
  - supported developing standard OpenMP, OpenMP Threads, Java
- asynchronous parallel computers. They are composed of a number of fully independent computing nodes (processors, cores or computers) which are connected through some communication network. To this group belong mainly various forms of computer networks (cluster), network of powerful workstation (NOW|) or more integrated network of NOW networks (Grid). Any cooperation and control are performed through inter process communication mechanisms (IPC) per realized remote or local communication channels. According the latest trends asynchronous parallel computers based on PC computers (single, SMP) are dominant parallel computers. Basic common characteristics are as following [10, 29]
  - no shared memory (distributed memory)
  - computing node could have some form of local memory where this memory in use only by connected computing node
  - cooperation and control of parallel processes

only using asynchronous message communication
- supported developing standard
  - MPI (Message passing interface)
  - PVM (Parallel virtual machine)
  - Java.

3. Typical Architectures of Modern Parallel Computers

3.1. Symmetrical Multiprocessor System

Symmetrical multiprocessor system (SMP) is a multiple using of the same processors or cores which are implemented on motherboard in order to increase the whole performance of such system. Typical common characteristics are following

- each processor or core (computing node) of the multiprocessor system can access main memory (shared memory)
- I/O channels or I/O devices are allocated to individual computing nodes according their demands
- integrated operation system coordinates cooperation of whole multiprocessor resources (hardware, software etc.).

Concept of multiprocessor system illustrates Fig. 2.

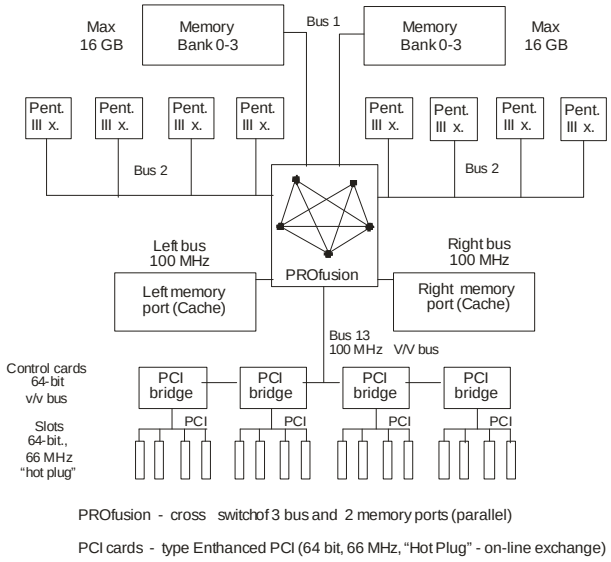

Figure 2. *Single computing node based on SMP (8-processors Intel Xeon).*

3.2. Network of Workstations

There has been an increasing interest in the use of networks of workstations (NOW) connected together by high speed networks for solving large computation intensive problems. This trend is mainly driven by the cost effectiveness of such systems as compared to massive multiprocessor systems with tightly coupled processors and memories (supercomputers). Parallel computing on a network of workstations connected by high speed networks has given rise to a range of hardware and network related issues on any given platform [3]. With the availability of cheap personal computers, workstations and networking devises, the recent trend is to connect a number of such workstations to solve computation intensive tasks in parallel on such clusters. Network of workstations [17, 19] 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. A way how to do it for an application problem builds its decomposition strategy. We will refer to it in [13].

Typical example of networks of workstations also for solving large computation intensive problems is at Fig. 3. The individual workstations are mainly extreme powerful personal workstations based on multiprocessor or multicore platform [1, 36]. Parallel computing on a cluster of workstations connected by high speed networks has given rise to a range of hardware and network related issues on any given platform.

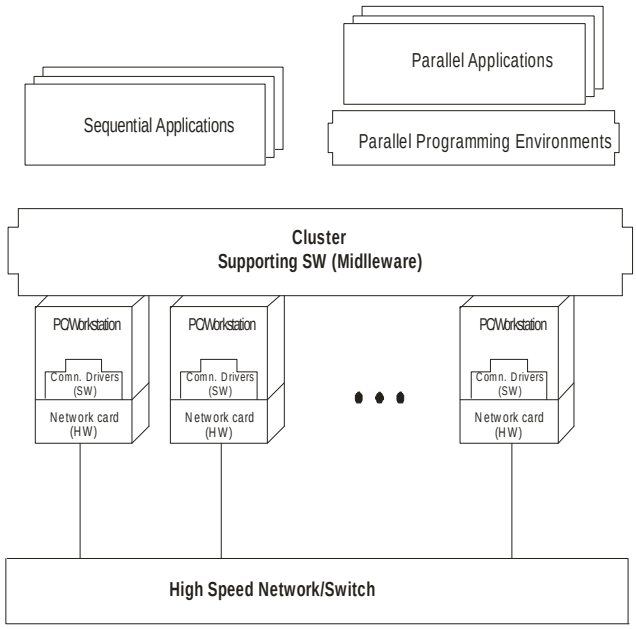

Figure 3. *Typical architecture of NOW.*

On such modular parallel computer we have been able to study basic problems in parallel computing (parallel and distributed computing) as load balancing, inter processor communication IPC [22, 32], modeling and optimization of parallel algorithms etc. [2, 14]. The coupled computing nodes PC_1, PC_2, ...,PC_i (workstations) could be single extreme powerful personal computers or SMP parallel computers. In this way parallel computing on networks of conventional PC workstations (single, multiprocessor, multicore) and Internet computing, suggest advantages of unifying parallel and distributed computing [30].

3.3. Grid Systems

Grid technologies have attracted a great deal of attention recently, and numerous infrastructure and software projects have been undertaken to realize various versions of Grids. In general Grids represent a new way of managing and organizing of computer networks and mainly of their deeper resource sharing [34]. 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. 4.). 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 [34]. It is obvious that existed HPC parallel computers (supercomputers etc.) could be a member of such Grid systems too. In general Grids represent a new way of managing and organizing of computer networks and mainly of their deeper resource sharing (Fig. 4).

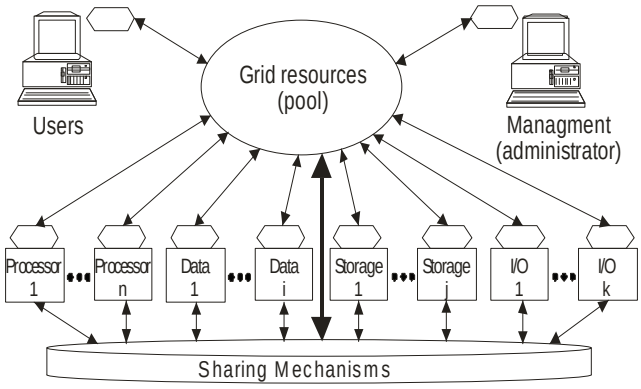

Figure 4. *Architecture of Grid node.*

Conceptually they go out from a structure of virtual parallel computer based on computer networks. In general Grids represent a new way of managing and organizing of resources like network of NOW networks. This term define massive computational Grid with following basic characteristics

- wide area network of integrated free computing resources. It is a massive number of inter connected 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 meta computing that means computing environment, which enables to individual applications a functionality of all system resources
- Grid system combines distributed parallel computation with remote computing from user workstations.

3.4. Meta Computing

This term define massive parallel computer (supercomputer, Grid).

The best example of existingmeta computer is Internet as massive international network of various computer networks. Fig. 5 illustrates Internet as virtual parallel computer from sight of common Internet user.

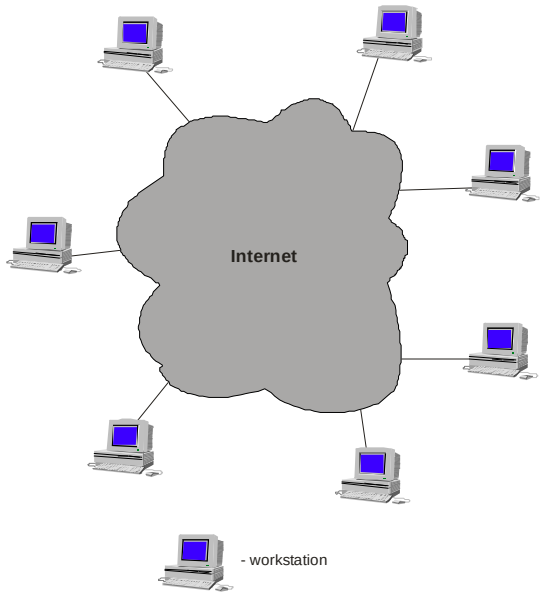

Figure 5. *Internet as virtual parallel computer.*

Another sight to Internet as network of connected individual computer networks is at Fig. 6. The typical networking switches are bridges, routers, gateways etc. which we denote with common term as network processors [27].

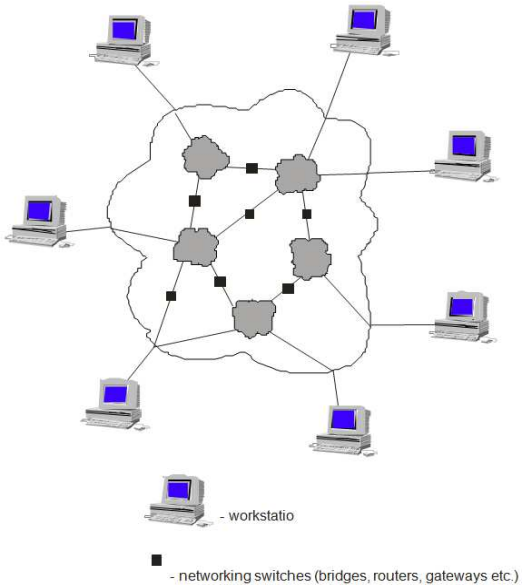

Figure 6. *Internet as network of connected networks.*

4. Modeling in Parallel Computing

Generally model is the abstraction of the system according Fig. 7 [15]. The functionality of the model represents the level of the abstraction applied. That means, if we know all there is about the system and are willing to pay for the complexity of building a true model, the role of abstraction is near nil. In practical cases we wish to abstract the view we take of a system to simplify the complexity of the real system. We wish to build a model that focuses on some basic elements of our interest and leave the rest of real system as only an interface with no details beyond proper inputs and outputs. A real system could be any parallel process or parallel computer that we are going to model [15]. In our cases they should be applied parallel algorithms (PA) or concrete parallel computers (SMP, NOW, Grid etc.).

The basic conclusion is that a model is a subjective view of modeler's subjective insight into modeled real system. This personal view defines what is important, what the purposes are, details, boundaries, and so one. Therefore the modeler must understand the system in order to guarantee useful features of the created model.

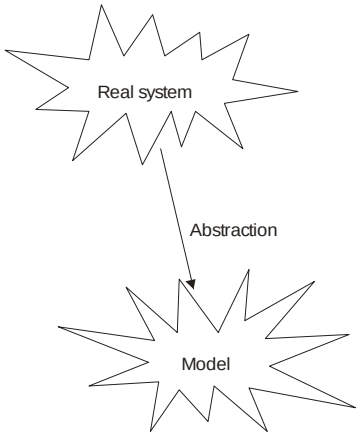

Figure 7. *Modeling process.*

4.1. Model Construction

Modeling is high creative process which incorporates following basic assumptions

- high ability of abstract thinking
- brain storming (creativity)
- alternating behavior and strategy
- logical hierarchical approaches to differ primary and secondary facts.

In general the development of model in any scientific area include the collection of following steps

- define the problem to be studied as well the criteria for analysis
- define and/or refine the model of the system. This include development of abstractions into mathematical, logical or procedural relationships
- collect data input to the model. Define the outside world and what must be fed to or taken from the model to "simulate" that world
- select a modeling tool and prepare and augment the model for tool implementation
- verify that the tool implementation is an accurate reflection of the model
- validate that the tool implementation provides the desired accuracy or correspondence with the real world system being modeled
- experiment with the model to obtain performance measures
- analyze the tool results
- use findings to derive designs and improvements for the real world system.

Corresponding flow diagram of model development represents Fig. 8.

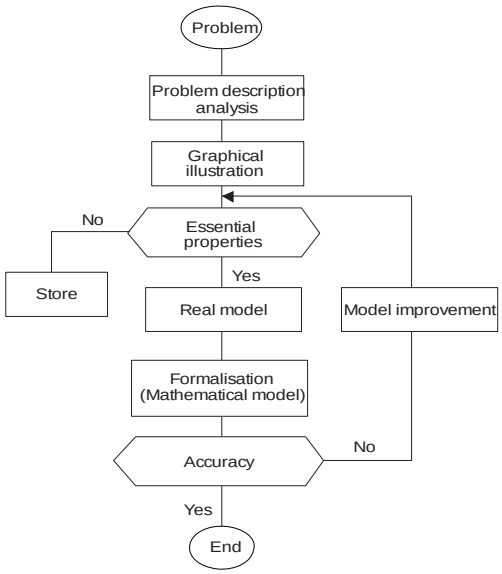

Figure 8. *Flow diagram of model development.*

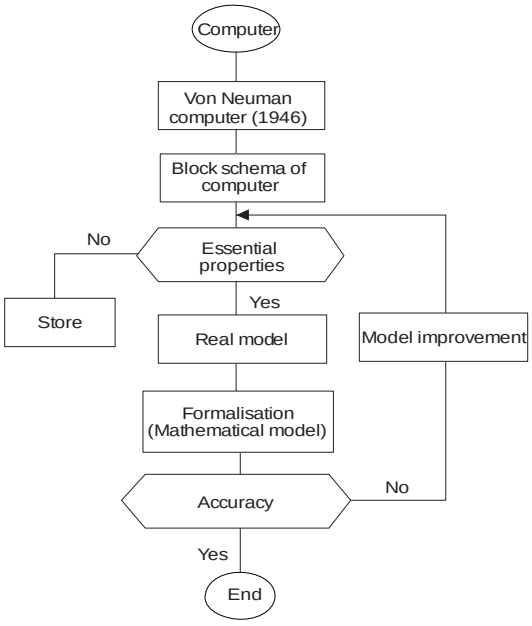

Figure 9. *Applied computer modeling.*

To practical illustration we have chosen applied modeling of classical sequential von Neumann computer according Fig. 9.

5. Abstract Models of Computing Nodes

5.1. Abstract model of SMP computing node with shared memory

Basic abstract model of parallel computer with shared memory is at Fig. 10.

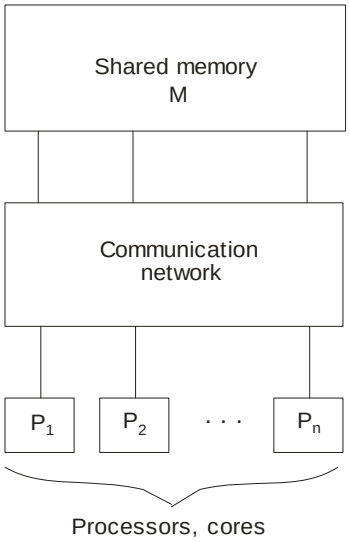

Figure 10. *Abstract model of SMP.*

5.2. Abstract Model of SMP with Distributed Memory

Basic abstract model of parallel computer with distributed memory is at Fig. 11.

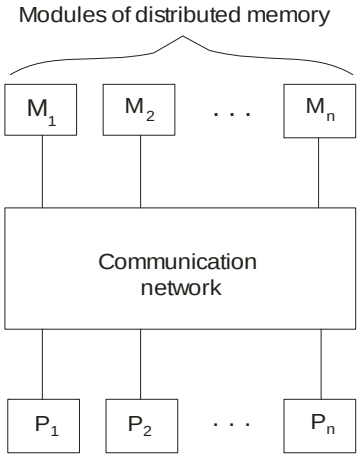

Figure 11. *Abstract model of NOW.*

6. The Role of Performance

Quantitative evaluation and modeling 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 behavior is an important part of any performance evaluation study [24, 33]. 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 [26].

6.1. Performance Evaluation Methods

The fundamental concepts have been developed for evaluating parallel computers. Trade-offs 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 [4, 8,16]
  - asymptotic (order) analysis [13, 14]
- simulation [21]
- experimental measurement
  - benchmarks [18]
  - modeling tools [23]
  - direct parameter measuring [6].

When we solve a model we can obtain an estimate for a set of values of interest within the system being modeled, 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. Each set of m input parameters constitutes a single point in m-dimensional input space. Each solution of the model produces one set of observations. Such a set of n values constitutes a single point in the corresponding n-dimensional observation space. By varying the input conditions we hope to explore how the outputs vary with changes to the inputs.

6.1.1. 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 modeling [7, 9]. Like simulation, queuing theory depends on the use of powerful computers in order to solve its models quickly. We would like to prefer techniques which yield analytic solutions.

6.1.2. The Simulation Method

Simulation is the most general and versatile means of modeling 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. To reduce the cost of a simulation we may resort to simplification of the model which avoids explicit modeling 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. This method is based on the simulation of the basic characteristics that are the input data stream and their servicing according the measured and analyzed probability values simulate the behavior model of the analyzed parallel system. Its part is therefore the time registration of the wanted interested discrete values. The result values of simulation model have always their discrete character, which do not have the universal form of mathematical formulas to which we can set when we need the variables of the used distributions as in the case of analytical models. The accuracy of simulation model depends therefore on the accuracy measure of the used simulation model for the given task.

6.1.3. Asymptotic (Order) Analysis

In the analysis of algorithms, it is often cumbersome or impossible to derive exact expressions for parameters such as run time, speedup, efficiency, issoefficiency etc. In many cases, an approximation of the exact expression is adequate. The approximation may indeed be more illustrative of the behavior of the function because it focuses on the critical factors influencing the parameter. We have used an extension of this method to evaluate parallel computers and algorithms in [13, 14].

6.1.4. Experimental Measurement

Evaluating system performance via experimental measurements is a very useful alternative for parallel systems and parallel 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.

6.1.4.1. Benchmark

We divide used performance tests as following

- classical
  - Peak performance
  - Dhrystone
  - Whetstone
  - LINPAC
  - Khornestone
- problem oriented tests (Benchmarks)
  - SPEC tests [37]
  - PRISM [23].

6.1.4.2. SPEC Ratio

SPEC (Standard Performance Evaluation Corporation) defined one number to summarize all needed tests for integer number. Execution times are at first normalized 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 labeled 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. The relation for geometric average value is given as

$$\sqrt[n]{\prod_{i=1}^{n} normalised \quad execution \quad time_{\,i}}$$

, where normalized execution time is the execution time normalized by reference computer for i – th tested program from whole tested group n (all tests) and

$$\prod_{i=1}^{n} a_i \quad - product\ of\ individual\ \ a_i .$$

7. 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 (or customers or message packets or anything else that requires the sort of processing provided by the server) are inserted into the network. A simple example, the single server model, is shown in Fig. 12. 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.

We will now turn our attention to some suitable queuing systems, the notation used to represent them, the performance quantities of interest, and the methods for calculating them. We have already introduced many notations for the quantities of interest for random variables

and stochastic processes.

7.1. Kendall Classification

Queuing theory systems are classified according to various characteristics, which are often summarized using Kendall`s notation [16, 28]. In addition to the notation described previously 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 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. Next, we will apply several suitable queuing systems to model computer systems or workstations and give expressions for the more important performance quantities. We will suppose in Kendall notation default values that means we will use typical short Kendall notation.

7.2. Little's Laws

One of the most important results in queuing theory is Little's law. This was a long standing rule of thumb in analyzing queuing systems, but gets its name from the author of the first paper which proves the relationship formally. It is applicable to the behavior of almost any system of queues, as long as they exhibit steady state behavior. It relates a system oriented measure - the mean number of customers in the system - to a customer oriented measure - the mean time spent in the system by each customer (the mean end-to-end time), for a given arrival rate. Little's law says

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

or it's following alternatives

- $E(w) = \lambda . E(t_w)$
- $E(w) = E(q) - \rho$ (single service where m=1)
- $E(w) = E(q) - m . \rho$ (m – services).

We can use also following valid equation

$E(t_q) = E(t_w) + E(t_s)$.

where the named parameters are as

- λ - arrival rate at entrance to a queue
- m - number of identical servers in the queuing system
- ρ - traffic intensity (dimensionless coefficient of utilization)
- 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.

7.3. The M/M/1 Queue Model

To model a single workstation as single PC computer we give results needed results for M/M/1. There are many other kinds of queues, including those where FIFO strategy (First In First Out) is not assumed, but few yield easily usable analytic results. The M/M/1 queuing system is characterized by a Poisson arrival process and exponential service time distributions, with one server, and a FIFO queue ordering discipline. The system at Fig. 12 represents an input buffer holding incoming data bytes, with an I/O processor as the server. A few of the quantities that we will be interested in for this type of queuing system are the average queue length, the wait time for a customer in the queue, the total time a customer spends in the system, and the server utilization.

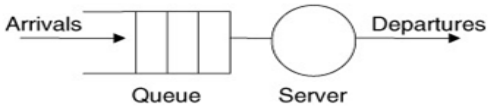

Figure 12. *Queuing theory based model.*

7.3.1. Poisson Distribution

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. The definition of Poisson distribution is according following relation

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

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

7.3.2. Exponential 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.

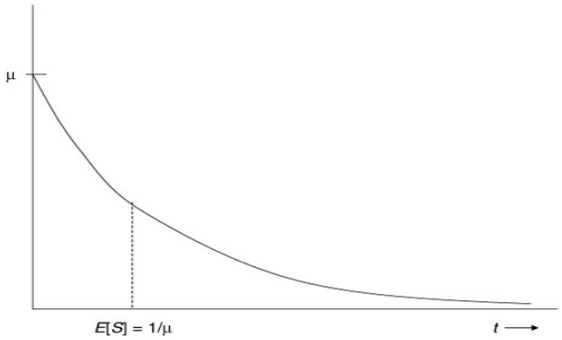

Figure 13. *Graphic illustration of exponential distribution.*

Exponential distribution function (Fig. 13) is defined as $p_i = \mu\ e^{-\mu t}$ for t > 0 and f (t) = 0 for t ≤ 0.

and its mean value as

E (S) = E (t_s) = 1/μ.

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.

7.3.3. The Derived Relation of M/M/1 Queue Model

We consider at first the M/M/1queue model. This represents a Poisson stream of independent arrivals into a queue whose single server has exponentially distributed service times. The queue is assumed to be unbounded and the population of potential customers to be infinite. Let λ be the (mean) rate of arrivals and μ be the (mean) rate of service (Fig. 14.).

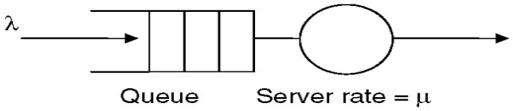

Figure 14. *M/M/1 queuing system model.*

We derive a measure called traffic intensity, ρ as

$$\rho = \frac{\lambda}{\mu}$$

From this we can see that the mean and variance of the number of customers in the queue are

$$E(w) = \frac{\rho}{(1-\rho)}$$

It also turns out that the end- to end delay (waiting time plus time being served) for each customer is exponentially distributed with parameter μ - λ. Thus the mean end to end delay is

$$E(q) = Var(q) = \frac{1}{(\mu - \lambda)}$$

Using Little's law we can get the end to end delay (waiting time plus time being served) for each

customer which is exponentially distributed with parameter μ - λ. Thus the mean end- to end delay is as

$$E(t_w) = \frac{\rho}{\mu - \lambda}$$

and waiting time in the queue as

$$E(t_q) = \frac{1}{\mu - \lambda}$$

7.3.4. M/M/m Queue Model

The illustration of M/M/m model for m=3 is at Fig. 15.

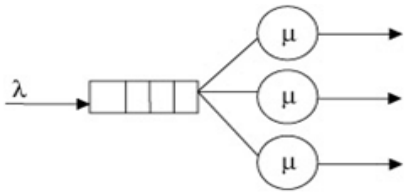

Figure 15. *M/M/m (m=3) model of multiprocessor or multicore systems.*

The basic needed derived relations for M/M/m queue model are following

$$\rho = \frac{\lambda}{\mu \,.\, m} < 1$$

Average number of customer in the queue

$$E(w) = \frac{\rho\,(\rho\, m)^m}{m\,!\,(1-\rho)^2}\ p_0$$

where the probability

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

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.

7.4. Non Markovian models

7.4.1. M/D/1 Queue Model

In this queue model traffic intensity ρ is as

$$\rho = \frac{\lambda}{\mu} < 1$$

The service time is constant and is given as

$$E(t_s) = \frac{1}{\mu}$$

Then we can derive that the mean number of customers in the queue are

$$E(w) = \frac{\rho^2}{2(1-\rho)}$$

and the mean number of customers in the system is

$$E(q) = \frac{\rho(2-\rho)}{2(1-\rho)}$$

The waiting time in the queue for each customer is

$$E(t_w) = \frac{3\rho - 2}{2\mu(1-\rho)}$$

and the end- to end delay (waiting time plus time being served) for each customer is

$$E(t_q) = \frac{\rho}{2\mu(1-\rho)}$$

7.4.2. M/D/m queue model

In this queue model traffic intensity ρ is as

$$\rho = \frac{\lambda}{\mu \,.\, m} < 1$$

The service time is constant and is given as

$$E(t_s) = \frac{1}{\mu}$$

Then we can use for the mean number of customers in the queue following approximate relation

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

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.

8. Results

8.1. Application of THO Models

We have modeled single processor system as M/M/1 and multiprocessor system as M/M/m and M/D/m queuing models (two processor system as M/M/2 resp. M/D/2, four processor system as M/M/4 resp. M/D/4 etc.), where we were supposed parallel activity of used independent processors or cores. The differences between multiprocessor or multicore are in their performance (input parameters). Therefore we can model booth system with the same queuing theory system with appropriate input parameters. The individual result parameters are as follows

Input parameters

- λ - arrival rate at entrance to a queue
- ρ - traffic intensity
- m- number of identical servers in the queuing system.

Output parameters

- E(q) is the expected (mean) number of entities in a system
- E(w) is the expected (mean) number of entities in a queue
- $E(t_q)$ is the mean time spent in system (queue + servicing)
- $E(t_w)$ is the mean time spent in the queue
- $E(t_s)$ is the mean time of servicing.

Table 1 contains all the needed mean values of M/M/4 queuing system. The input parameter is ρ as input load. To compute the results we used concrete value of input intensity λ=3 and ρ = λ . $E(t_s)$ / 4 as input load intensity of four service equipment.

Table 1. *Results for modeled 4 – multiprocessor system (λ=3)*

P	E(w) [MIPS]	E(q) [MIPS]	$E(t_w)$ [s]	$E(t_q)$ [s]	$E(t_s)$ [s]
0,1	0,000	0,400	0,000	0,133	0,133
0,2	0,002	0,802	0,001	0,267	0,267
0,3	0,016	1,216	0,005	0,405	0,400
0,4	0,060	1,660	0,020	0,553	0,533
0,5	0,174	2,174	0,058	0,725	0,667
0,6	0,431	2,831	0,144	0,944	0,800
0,7	1,000	3,800	0,333	1,267	0,933
0,8	2,386	5,586	0,795	1,862	1,067
0,9	7,090	10,690	2,363	3,563	1,200

Graphics illustration of results from Tab. 1 for modeled 4 – multiprocessor system (λ=3, ρ = λ . $E(t_s)$ / 4) are at Fig. 16, where x - axis contains values of parameter ρ (range of input load) and y - axis individual processing times per second.

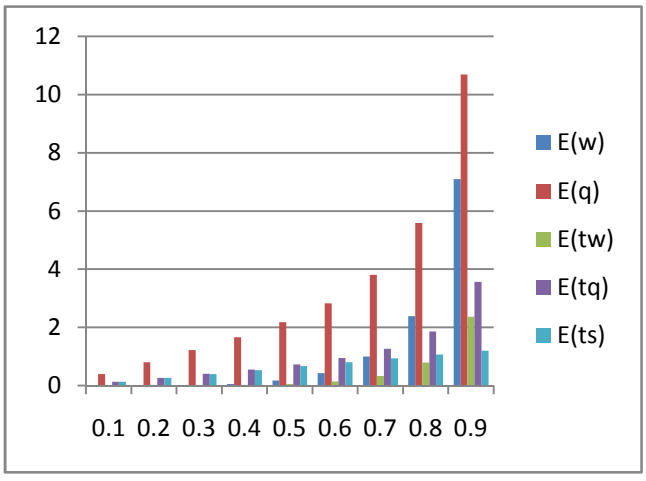

Figure 16. *Illustration of modeling 4 – multiprocessor system (λ=3).*

Graphic illustration at Fig. 17 illustrates average waiting times spent in system (queue + servicing) $E(t_q)$ of M/M/m queuing model (M/M/1, M/M/2, M/M/4) (λ=3, ρ = λ. $E(t_s)$ / m) for various number of services m, where x - axis

contains values of parameter ρ (traffic intensity) and y - axis individual processing times per second.

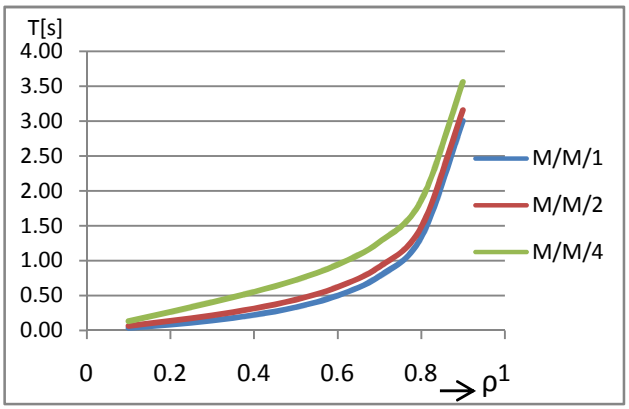

Figure 17. *Mean time in system T= E(t_q).*

Table 2. *Results for M/D/4 (λ=3, 4 – multiprocessor system)*

ρ	E(w) [MIPS]	E(q) [MIPS]	E(t_w) [s]	E(t_q) [s]	E(t_s) [s]
0,1	2,99	3,39	1,00	1,13	0,13
0,2	2,93	3,73	0,98	1,24	0,27
0,3	2,85	4,05	0,95	1,35	0,40
0,4	2,81	4,41	0,94	1,47	0,53
0,5	2,81	4,81	0,94	1,60	0,67
0,6	2,90	5,30	0,97	1,77	0,80
0,7	3,07	5,87	1,02	1,96	0,93
0,8	3,32	6,52	1,11	2,17	1,07
0,9	3,66	7,26	1,22	2,42	1,20

Graphics illustration of some results from Tab. 2 for modeled 4 – multiprocessor system (λ=3, ρ = λ .E(t_s) / 4) are at Fig. 18., where x - axis contain values of parameter ρ (range of traffic intensity) and y - axis contain individual processing and waiting times per second.

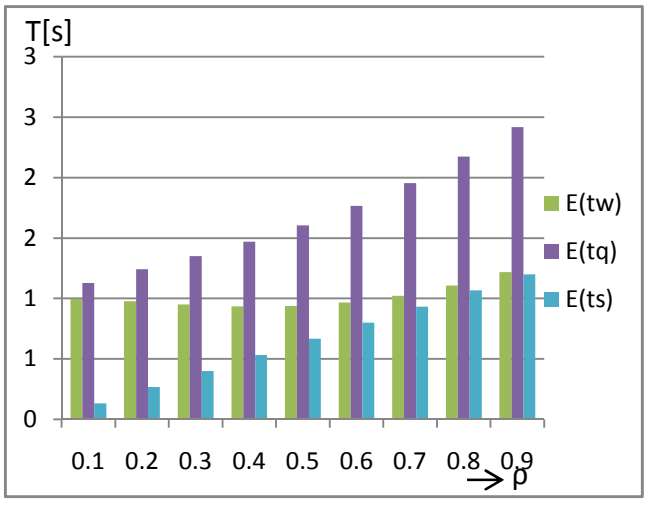

Figure 18. *Illustration of some values E(t_w), E(t_q), E(t_s) for M/D/4 system.*

Graphics illustration at Fig. 19 compare various queuing models (M/D/4, M/M/4) (λ=3, ρ = λ .E(t_s) / 4) for average time in system (queue + servicing), where x - axis contains values of parameter ρ (input load) and y - axis individual processing times per second. From this comparison we can better results namely for higher values of traffic intensity parameter ρ.

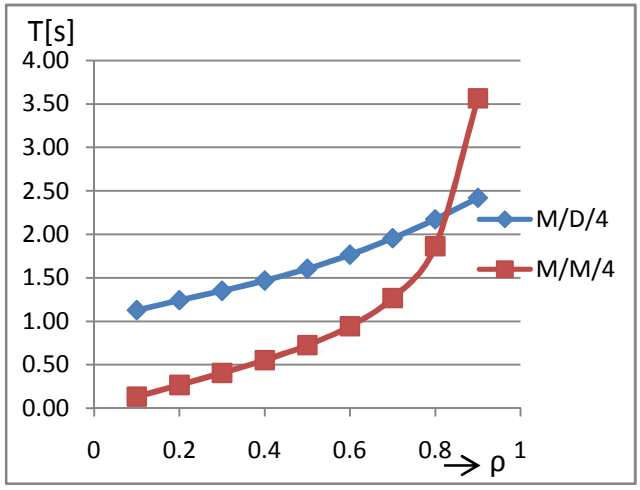

Figure 19. *Average waiting time in system (λ=3, T= E(t_q).*

8.2. Spec test ratio

We have been performed various tests (benchmarks) to verify derived 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.

Tab. 3 illustrates tested results for processor Intel i7-950 and at the same time description of used SPEC tests to evaluating performance. As we can see the used tests are really from various applications in order to come to more universal tested results.

Table 3. *Illustration of tested results for processor Intel i7-950*

Description	Name	Execution time [s]	SPEC ratio
String processing	Perl	445	21,9
Compression	bzip2	554	17,4
GNU C compiler	Gcc	321	25,1
Combinatorial optimization	Mcf	202	45,1
Artificial Intelligence	Go	460	22,8
Search gene sequence	Hmmer	516	18,1
Chess game (AI)	Sjeng	507	29,3
Quantum computer simulation	Libquantum	97,7	212
Video compression	h264avc	605	36,6
Discrete event simulation library	Omnetpp	269	23,3
Games/path finding	Astar	414	16,9
XML Processing	Xalancbmk	240	28,7
Geometric mean			29,1

To compare any computers using SPEC ratios test we prefer to use geometric mean value therefore it defines the same relative value regardless of used normalized reference computer. If we were evaluating normalized values using arithmetic mean value results would be depended from the

type of used normalized computer. Graphical illustration of our tested computers is at Fig. 20 using standardized performance tests of SPEC consortium. According our expectations processor Intel i7-950 achieved the highest SPEC ratio value.

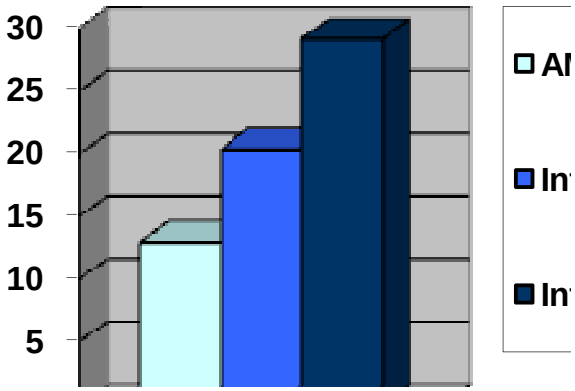

Figure 20. *Comparison of tested processors.*

9. Conclusions

Performance evaluation of computers generally used to be a very hard problem from birthday of computers. It was very hard to apply analytical methods based on queuing theory results to performance evaluation of sequential computers because of their high number of not predictable parameters. Secondly increasing of computer performance was done at first through technology improvements and processor's architecture changes.

From a point of user application of any analytical method (queuing theory, order analyze) is to be preferred in comparison with other possible methods because of transparent using of achieved results. Application of simulation method or Petri nets does not have such universal character as discussed analytical methods. Actually dominant using of multiprocessor or multicore computers opens more possibilities to apply a queuing theory results to their performance evaluation. This is based on a knowledge that outputs from more than one processor better approximate assumed Poisson distribution. Further the outputs from any computing node are going to another computing node in dominant parallel computers (NOW, Grid, meta computer). In relation to it we began to apply results of queuing theory to at first single computing node of parallel computer and then to dominant parallel computers based on NOW and their high integrated Grid (virtual parallel computer). To solve such coupled network of computing nodes appoints to couple network of queuing theory systems. We refer to it in another paper of this Special Issue [11]. The achieved results we can apply to performance modeling of multiprocessors or multicores) using as input parameter $\rho = \lambda . E(t_s) / m$ (for m=1 we can model computing node with one processor) as follows

- 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 model). In case of balanced parallel processes we could use the results as upper limits of exanimate parameters
- 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 model)
- 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.

To consider incoming instructions at using M/M/m system it would be necessary to recalculate at entrance incoming parallel processes to instructions. To verify and to precise used analytical models we have been used SPEC ratio tests results (mean values of execution times for applied tests). To get comparable results we used the relation between throughput and execution time (latency) as

$$Throughput = \frac{1}{Latency}.$$

To model single computing nodes we can also use other more complicated single queuing theory systems than the analyzed ones (M/M/1, M/M/m, M/D/1, and M/D/m). We have choose the analyzed models from these causes

- to finish performance analysis of networks of queuing theory system we need results of chosen single queuing theory systems M/M/m and M/D/m
- we need their results to compute approximation relation for M/D/m
- M/M/1 and M/M/m models could be used to compare their results with other models M/D/1 and M/D/m respectively
- results of analyzed models M/M/m and M/D/m are necessary to finish coupled network of computing nodes [12].

Now according current trends in virtual parallel computers (SMP, NOW, Grid), based of powerful personal computers, we are looking for unified flexible models of any parallel computer that will be incorporated influences of

- other queue ordering discipline than FIFO
- various routing strategies
- various decomposition models etc.

In such flexible models we would like to study load balancing, inter process communication (IPC), transport protocols, performance prediction etc. We would refer to achieved results later.

Acknowledgements

This work was done within the project "Modeling, optimization and prediction of parallel computers and algorithms" at University of Zilina, Slovakia. The author

gratefully acknowledges help of project supervisor Prof. Ing. Ivan Hanuliak, PhD.

References

[1] Abderazek A. B., Multicore systems on chip - Practical Software/Hardware design, Imperial college press, pp. 200, 2010

[2] Arora S., Barak B., Computational complexity - A modern Approach, Cambridge University Press, pp. 573, 2009

[3] Coulouris G., Dollimore J., Kindberg T., Distributed Systems – Concepts and Design (5 - th Edition), Addison Wesley, United Kingdom, pp. 800, 2011

[4] Dattatreya G. R., Performance analysis of queuing and computer network, University of Texas, Dallas, USA, pp. 472, 2008

[5] Dubois M., Annavaram M., Stenstrom P., Parallel Computer Organization and Design, Cambridge university press, United Kingdom, pp. 560, 2012

[6] Dubhash D.P., Panconesi A., Concentration of measure for the analysis of randomized algorithms, Cambridge University Press, United Kingdom, 2009

[7] Gautam Natarajan, Analysis of Queues: Methods and Applications, CRC Press, USA, pp. 802, 2012

[8] Gelenbe E., Analysis and synthesis of computer systems, Imperial College Press, United Kingdom, pp. 324, April 2010

[9] Giambene G., Queuing theory and telecommunications, Springer, Germany, pp. 585, 2005

[10] Hager G., Wellein G., Introduction to High Performance Computing for Scientists and Engineers, CRC Press, USA, pp. 356, 2010

[11] Hanuliak M., Modeling of dominant parallel computers based on NOW, American J. of Networks and Communication, Science PG, Vol. 3, USA, 2014

[12] Hanuliak M., Hanuliak P., Performance modeling of parallel computers NOW and Grid , AJNC (Am. J. of Networks and Communication), Science PG, USA, pp. 112-124, 2013

[13] Hanuliak J., Modeling of communication complexity in parallel computing, American J. of Networks and Communication, Science PG, Vol. 3, USA, 2014

[14] Hanuliak J., Hanuliak I., To performance evaluation of distributed parallel algorithms, Kybernetes, Volume 34, No. 9/10, United Kingdom, pp. 1633-1650, 2005

[15] HarcholBalterMor, Performance modeling and design of computer systems, Cambridge University Press, United Kingdom, pp. 576, 2013

[16] Hillston J., A Compositional Approach to Performance Modeling, University of Edinburg, Cambridge University Press, United Kingdom, pp. 172, 2005

[17] Hwang K. and coll., Distributed and Parallel Computing, Morgan Kaufmann, pp. 472, 2011

[18] John L. K., Eeckhout L., Performance evaluation and benchmarking, CRC Press, USA, 2005

[19] Kshemkalyani A. D., Singhal M., Distributed Computing, University of Illinois, Cambridge University Press, United Kingdom, pp. 756, 2011

[20] Kirk D. B., Hwu W. W., Programming massively parallel processors, Morgan Kaufmann, USA, pp. 280, 2010

[21] Kostin A., Ilushechkina L., Modeling and simulation of distributed systems, Imperial College Press,United Kingdom, pp. 440, 2010

[22] Kushilevitz E., Nissan N., Communication Complexity,Cambridge University Press, United Kingdom, pp. 208, 2006

[23] Kwiatkowska M., Norman G., and Parker D., PRISM 4.0: Verification of Probabilistic Real-time Systems, In Proc. 23rd Int. Conf. on CAV'11, Vol. 6806 of LNCS, Springer, Germany, pp. 585-591, 2011

[24] Le Boudec Jean-Yves, Performance evaluation of computer and communication systems, CRC Press, USA, pp. 300, 2011

[25] McCabe J., D., Network analysis, architecture, and design (3rd edition), Elsevier/Morgan Kaufmann, USA, pp. 496, 2010

[26] Miller S., Probability and Random Processes, 2nd edition, Academic Press, Elsevier Science, Netherland, pp. 552, 2012

[27] Misra Ch. S., Woungang I., Selected topics in communication network and distributed systems, Imperial college press, United Kingdom, pp. 808, April 2010

[28] Natarajan G., Analysis of Queues - Methods and Applications, CRC Press, USA, pp. 802, 2012

[29] Patterson D. A., Hennessy J. L., Computer Organization and Design (4th edition), Morgan Kaufmann, USA, pp. 914, 2011

[30] Peterson L. L., Davie B. C., Computer networks – a system approach, Morgan Kaufmann, USA, pp. 920, 2011

[31] Resch M. M., Supercomputers in Grids, Int. J. of Grid and HPC, No.1, Germany, pp. 1 - 9, 2009

[32] Riano l., McGinity T.M., Quantifying the role of complexity in a system's performance, Evolving Systems, Springer Verlag, Germany, pp. 189 – 198, 2011

[33] Ross S. M., Introduction to Probability Models, 10th edition, Academic Press, Elsevier Science, Netherland, pp. 800, 2010

[34] Wang L., Jie Wei., Chen J., Grid Computing: Infrastructure, Service, and Application, CRC Press, USA, 2009
www pages

[35] www.top500.org

[36] www. intel.com

[37] www.spec.org.