American Journal of Networks and Communications
2013; 2(5): 112-124
Published online October 30, 2013 (http://www.sciencepublishinggroup.com/j/ajnc)
doi: 10.11648/j.ajnc.20130205.11

Performance modeling of parallel computers NOW and Grid

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. Performance Modeling of Parallel Computers NOW and Grid. *American Journal of Networks and Communications.* Vol. 2, No. 5, 2013, pp. 112-124. doi: 10.11648/j.ajnc.20130205.11

Abstract: The paper describes development, realization and verification of more precise analytical models for the study of the basic performance parameters of parallel computers based on connected parallel computers (Cluster, NOW, Grid). At first the paper describes very shortly the developing steps of parallel computer architecture and then he summarized the basic concepts for performance modeling of mentioned parallel computers. To illustrate theoretical evaluation concepts the paper considers in its experimental part the achieved results on concrete analyzed examples and their comparison. The suggested model considers for every node of the NOW or Grid networks one part for the own workstation's activities and another one for node's communication channel modeling of performed data communications. In case of using multiprocessor system, as modern node's communication processor, the suggested model considers for own node's activities M/D/m queuing theory system and for every node's communication channel M/D/1 system. Based on these more realistic assumptions we have been developed improved analytical models to account the real no exponential nature of the inputs to the modeling queuing systems. The achieved results of the developed models were compared with the results of the common used analytical and simulation model to estimate the magnitude of their improvement. The developed analytical models could be used under various ranges of input analytical parameters, which influence the architecture of NOW or Grid computer networks and which are interested from the sight of practical using. These consequences are in relation to the developed analytical models and their verifications through simulation model.

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

1. Developing Periods in 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. General purpose processors on a single chip, which had been invented in the early 1970's, were only mature enough to hit the HPC market by the end of the1980s, and it was not until the end of the 1990's that connected standard workstation or even personal computers (PC) had become competitive at least in terms of theoretical peak performance. 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) [1] so as on POWER PC as in connected network of workstations (NOW). Gained experience with the implementation of parallel principles and intensive extensions of computer networks, leads to the use of connected computers for parallel solution. These trends are to be characterized through downsizing of supercomputers as Cray/SGI, T3E and from other massive parallel systems [16] (number of used processor >100) to cheaper and more universal parallel systems in the form of a network of workstations (NOW). 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, 19]

- high performance processors and computers
- high speed interconnecting networks

- standardized tools for development of parallel algorithms (Shared memory, distributed memory).

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 metacomputer [31]. Metacomputer can be understood as the massive computer network of computing nodes built on the principle of the common use of existing processors, memories and other resources with the objective to create an illusion of one huge, powerful supercomputer. Such higher integrated forms of NOW's (Grid module) create various actually Grid systems or metacomputers we can define as the third period in developing trends of parallel computers.

2. Classification of Parallel Systems

It is very difficult to classify all existed parallel systems. But from the point of programmer-developer we can divide them [4, 10] to the two following different groups

- synchronous parallel architectures. These are used for performing the same or very similar computation on different sets of data. 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.). The typical examples of synchronous parallel computers illustrate Figure 1 on its left side. Some of used parallel principles in past time are step-by-step applied in actually modern personal computers (PC) for example in a form of SIMD (Single instruction multiple data) computer instructions within their computer set instruction (CSI)
- asynchronous parallel computers. They are composed of a number of fully independent computing nodes (processors, cores or computers. In programming parallel algorithms there are necessary to use inter process communications (IPC). To this group belong mainly various forms of computer networks (cluster), network of workstation (NOW) or more integrated Grid modules in the form as any networks of NOW networks (Grid). The typical examples of asynchronous parallel computers illustrate Figure 1 on its right side. According long-time trends asynchronous parallel computers based on PC computers (single, SMP) are dominant parallel computers [16, 27].

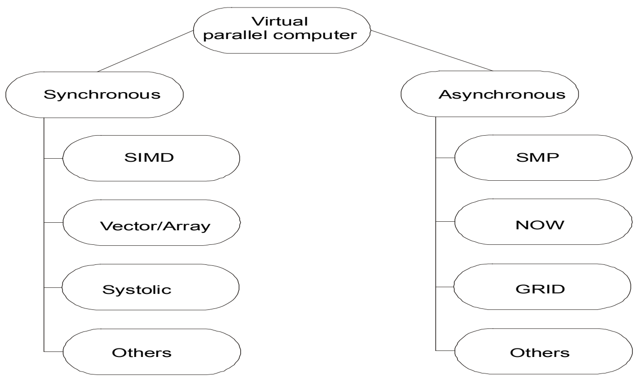


Figure 1. *Classification of parallel computers.*

3. Architectures of 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 such multiprocessor system illustrates Figure 2.

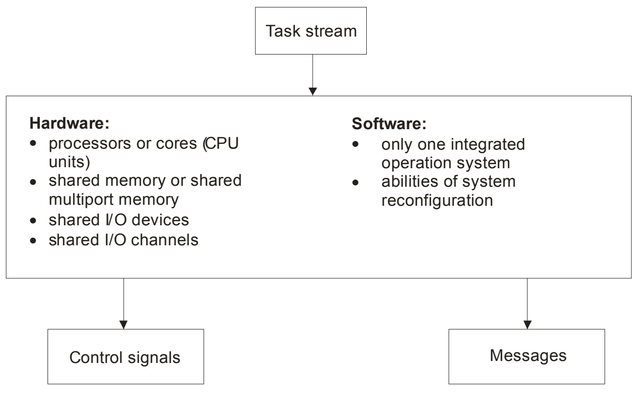
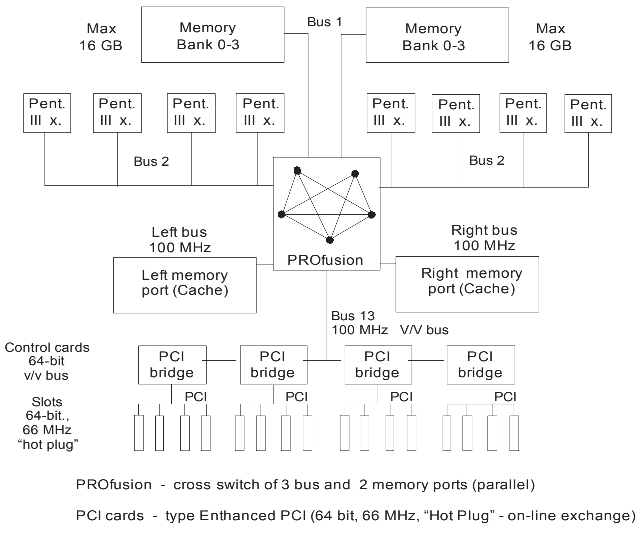

Task stream
Hardware:
• processors or cores (CPU units)
• shared memory or shared multiport memory
• shared I/O devices
• shared I/O channels
Software:
• only one integrated operation system
• abilities of system reconfiguration
Control signals
Messages

Figure 2. *Typical characteristics of multiprocessor systems.*

Typical practical architecture example of eight multiprocessor systems (Intel Xeon) illustrates Figure 3.



Max 16 GB
Memory Bank 0-3
Bus 1
Memory Bank 0-3
Max 16 GB
Pent. III x.
Bus 2
Bus 2
Left bus 100 MHz
Right bus 100 MHz
PROfusion
Left memory port (Cache)
Right memory port (Cache)
Bus 13 100 MHz V/V bus
Control cards 64-bit v/v bus
PCI bridge
PCI
Slots 64-bit., 66 MHz "hot plug"
PROfusion - cross switch of 3 bus and 2 memory ports (parallel)
PCI cards - type Enthanced PCI (64 bit, 66 MHz, "Hot Plug" - on-line exchange)

Figure 3. *Architecture of multiprocessor (8-Intel processor).*

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 cluster of workstations connected by high speed networks has given rise to a range of hardware and network related issues on any given platform [6]. 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 [13, 28] 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. This step belongs to a most important step in developing effective parallel algorithm [13, 18].

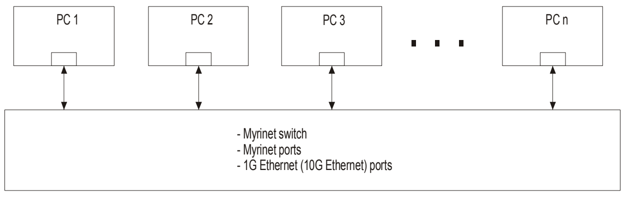


Figure 4. *Architecture of NOW.*

Principal example of networks of workstations is at Figure 4. The individual workstations are mainly powerful workstations based on multiprocessor or multicore platform.

3.3. Grid Systems

The In general Grids represent a new way of managing and organizing of computer networks and mainly of their deeper resource sharing (Figure 5.).

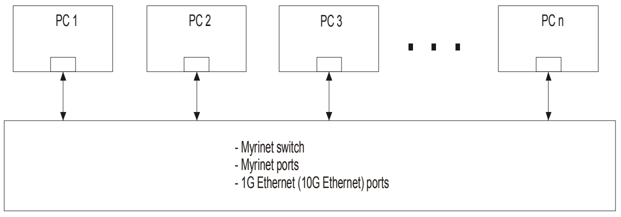


Figure 5. *Architecture of Grid node.*

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 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 user workstations.

3.3.1. Conventional HPC Environment Versus Grid Environments

In Grids, the virtual pool of resources is dynamic and diverse, since the resources can be added and withdrawn at any time according to their owner's discretion, and their performance or load can change frequently over the time. The typical number of resources in the pool is of the order of several thousand or even more. An application in a conventional parallel environment (HPC computing) typically assumes a pool of computational nodes from (a subset of) which a virtual concurrent machine is formed [4, 24]. The pool consists of PC's, workstations, and possibly supercomputers, provided that the user has access (valid login name and password) to all of them. Such virtual pool of nodes for a typical user can be considered as static and this set varies in practice in the order of 10 – 100 nodes. At table 1 we summarize mine analyzed differences between conventional distributed and Grid systems. We can also generally say that

- HPC environments are optimized to provide maximal performance
- Grids are optimized to provide maximum of existed resource capacities.

Table 1. *Comparison of environments in HPC and Grid computing*

	Conventional HPC environments	**Grid environments**
1.	A virtual pool of computational nodes	A virtual pool of resources
2.	A user has access (credential) to all nodes in the pool	A user has access to the pool but not to individual nodes
3.	Access to a node means access to all resources on the node	Access to a resource may be restricted
4.	The user is aware of the applications and features of the nodes	User has little or no knowledge about each resource
5.	Nodes belong to a single trust domain	Resources span multiple trust domains
6.	Elements in the pool 10 – 100, more or less static	Elements in the pool >>100, dynamic

3.4. Integration of Parallel Computers

With the availability of cheap personal computers, workstations and networking devises, the recent trends are to connect a number of such workstations to solve computation intensive tasks in parallel on various integrated forms of clusters based on computer networks. We illustrated at Figure 6 typical integrated complex consisted of NOW networks modules. It is clear that any classical parallel computers (massive multiprocessor, supercomputers etc.) in the word could be a member of such NOW [29].

For the support of reaching connectivity to any of existed integrated parallel computers in Europe (supercomputers, NOW, Grid) we can use the European classical massive parallel systems by means of scientific visits of project participants in the HPC centers of EU. These HPC centers are EPCC Edinburgh (UK), BSC (Barcelona, Spain), CINECA (Bologna, Italy), GENCI (Paris, France), SARA (Amsterdam, Netherland), HLRS (Stuttgart, Germany), CSC (Helsinki, Finland).

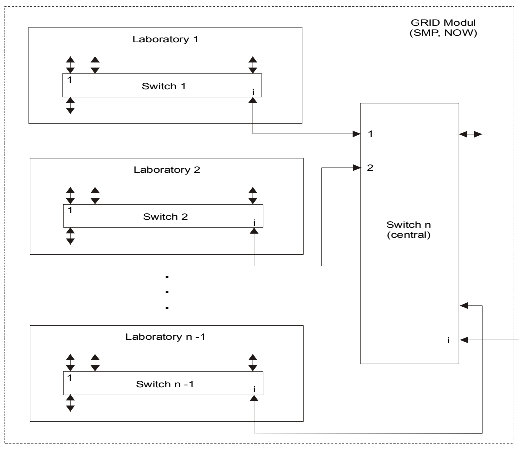


Figure 6. *Integration of NOW networks.*

4. 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. 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.

The individual workstations are mainly powerful workstations based on multiprocessor or multicore platform.

4.1. 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 [7, 8]
  - asymptotic (order) analysis [11, 13]
- simulation [20]
- experimental measurement
  - benchmarks [17, 23]
  - modeling tools [30, 22]
  - direct parameter measuring [11, 13]

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.

4.1.1. 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 [3, 25]. 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.

4.1.2. 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.

4.1.3. 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 [11, 13].

4.1.4. 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.

5. 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.

5.2. Queuing Networks

Continuing the examination of analytically tractable models, we look for useful results for networks of queues. These can be divided into two main groups, known as product form and non-product form. Product form networks have the property that they can be regarded as independently operating queues, where steady state can be expressed as both a set of global balance equations on customer flow in the whole network and a set of local balance equations on each queue. Local flow balance says that the mean number of customers entering any queue from all others must equal the number leaving it to go to all others, including customers which leave and rejoin the same queue immediately.

As an example of simplest queuing networks is serial connection of two queuing theory systems according Figure 7. (Tandem network), for which we can get following final solution, that the probability of k_1 demands at first node and k_2 demands at second node is

$$p\,(k_1,k_2) = p_1(k_1)\cdot p_2(k_2) = (1-\rho_1)\rho^{k_1}.(1-\rho_2)\rho^{k_2}$$

where we assumed that

$$\rho_1 = \frac{\lambda}{\mu_1} < 1,\ \ \rho_2 = \frac{\lambda}{\mu_2} < 1$$

and

- p (k_1, k_2) is the probability of k_1 demands in the first queue and k2 demands in the second queue
- p (k_1) is the probability of k_1 demands in the first queue
- p (k_2) is the probability of k_2 demands in the second queue.

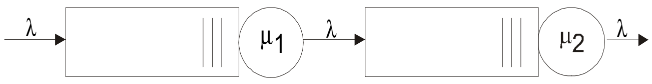


***Figure** 7. Tandem network of two M/M/1 queuing systems.*

The final expression proves by evidence independence of both M/M/1 queuing theory systems. Generalization to the U queuing theory systems of M/M/1 or M/M/m made Jackson (Jackson theorem) [5, 9]. Several different ways of identifying this sort of behavior have been proposed, but the name product form comes from Jackson`s theorem, which expresses the joint probability of the numbers of customers at each queue being a particular combination is the product of their individual probabilities of having that number. We begin by considering Jackson networks and then look at the extension to a more general queuing theory systems.

Based on Jackson result any network of queuing theory systems without feedback loop (they are not allowed to return any serviced demands again) with exponential service distribution, which are servicing independent Poisson input streams, generate also for a next node independent Poisson input stream. The whole demand probabilities for all nodes are given through multiplying of individual independent queuing theory systems and that in general as M/M/m systems. The outgoing stream will be exactly Poisson on the assumption of unlimited size of queue.

5.3. Jackson Theorem

Consider the case of a network of U queue/server nodes (Workstations). Customers enter the network at node j in a Poisson stream with rate γ_j. Each node has a multiple servers m (workstations based on multiprocessor, for m = 1 workstations with single server) and service times are distributed exponentially, with mean $1/\mu_j$, (j = l,…, U). When a customer leaves node i it goes to node j with probability r_{ij}. Customers from i leave the network with probability

$$1-\sum_{j=1}^{U} r_{ij}$$

Now let λ_i be the average total arrivals at node i, including those from outside (external input) and those from other nodes (Internal inputs). If the network is in steady state, λ_i is also the rate of customers leaving i node (including intern output). Overall we can formulate a set of „flow balance equations" which express these flows.

$$\lambda_i = \gamma_i + \sum_{i=1}^{U} \lambda_i\, r_{ij} \qquad j=1,2,\ldots, U$$

As long as the network is open, i.e. at least one γ_i is non-zero, this represents a set of linear simultaneous equations with an obvious solution. Let traffic intensity at i be

λ_i / m_i . μ_j < 1 for every i. The joint distribution of the number of customers p (k_1, k_2, ... k_U) at each of the U nodes, $p_1(k_1)$, $p_2(k_2)$, … p_U (k_U), can be expressed as

$$p\,(k_1,k_2,\ldots,k_U) = p_1(k_1)\cdot p_2(k_2)\cdot\ldots\cdot p_U(k_U) = \prod_{i=1}^{U} p_i.k_i$$

This is Jackson theorem for M/M/m system. The individual probabilities p_i (k_i) are given as

$$p_i(k_i) = \begin{cases} p_0\,\dfrac{(m\ \rho)^i}{i!}, & \text{pre}\quad 1 \le i \le m \\[2ex] p_0\,\dfrac{\rho^k m^m}{m!}, & \text{pre}\quad i > m \end{cases}$$

, where

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

Jackson's theorem describes each node as an independent single server system with Poisson arrivals and exponential service times. The total average number of customers in the whole NOW module is

$$\text{E (q)}_{\text{now}} = \sum_{i=1}^{U} E(q)_i$$

where $E(q)_i$ is given as

$$E(q)_i = \frac{(\rho\, m)^{m+1}}{(m-1)!\left[\sum_{i=0}^{m} \frac{(m\,\rho)^i}{i\,!}\left[(m-i)^2 - i\right]\right]}$$

Then from Little's law, total time spent by customers in the network $E(t)_q$ is

$$\mathrm{E}[t_q]_{now} = \sum_{i=1}^{U} \frac{E(q)_i}{\lambda_i}$$

Jackson theorem assumes for its applying verification of assumed independence of individual network nodes. Every element on its right side is a solution of independent M/M/m geeing system with their average input value λ_i. We can get the intensities of this individual inputs λ_i with solving a system of linear differential equations for concrete values of extern inputs λ_i and for given transition matrix r_{ij}.

6. Modeling of the NOW and Grid

NOW is a basic module of any Grid system (Network of NOW networks as for example Internet). Structure of essential parts in any workstation (i-th node) of NOW based on single processor (m=1) or multiprocessor system (m - processors or cores) is illustrated at Figure 8. Inter process communication (IPC) represents all needed communication in NOW as

- communication among parallel processes
- control communication.

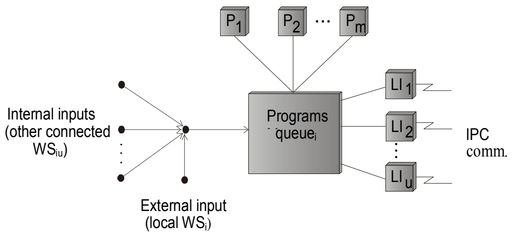

P1
P2
...
Pm
Internal inputs (other connected WSiu)
Programs queuei
LI1
LI2
LIu
IPC comm.
External input (local WSi)

Figure 8. *Structure of i – th computing node (WSi).*

In principle we are assumed any constraints on structure of communication system architecture. Then we are modeling one workstation as a system with two dominant overheads

- computation overheads (processor's latency)
- communication latency [21, 26].

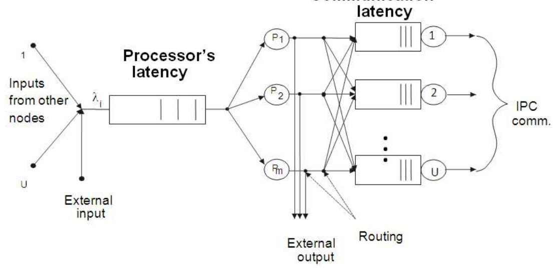


Figure 9. *Mathematical model of i – th node of NOW.*

To model these overheads through applying queuing theory we created mathematical model of one i-th computing node according Figure 9, which models

- computation overheads (processor's latency) [2] as queuing theory system
- every communication channel of i-th node LI_i i=1,2,…U (Link interface) as next queuing theory systems (communication system).

Such communication network in NOW module we can represent by a weighted graph where their nodes are individual workstations. IPC data arrive at random at a source node and follow a specific route in the networks towards their destination node. Data lengths of communicated parallel processes in data units (for example in words) are considered to be random variables following distributions according Jackson theorem. Those data units are then sent independently through the communication network nodes towards the destination node. At each node a queue of incoming data units is served according to a first-come first-served (FCFS) discipline. The defined communication network generally creates oriented graph (communication network) with U-nodes according to the Figure 10, where

- $\gamma_1, \gamma_2, ..., \gamma_U$ represent total extern intensities of input data stream to the given WS_i
- r_{ij} is a relation probabilities from node i to the neighbouring connected nodes j (WS_{ij}) for (i= 1,...,U, j = 1,...U)
- u_i – the number of communication channels at i-th node
- U – number of computing nodes (workstations)
- $\beta_1, \beta_2, ..., \beta_U$ are the individual total external output streams of data units from WS_i.

Such a model corresponds in queuing theory to the model of open servicing network. Adjective "open" characterize the extern input and output data stream to the servicing transport network [14, 26]. In common they are the open Markov servicing networks, in which the demand are mixed together at their output from one queuing theory system to another connected queuing theory system in a random way to that time as they are leaving the network. To the given i-th node the demand stream enter extern (from the network side), with the independent Poisson arrival distribution and the total intensity γ_i demands in seconds. After servicing at i-th node the demand goes to the next j-th node with the probability r_{ij} in such a way that the demand walks to the j-th node intern (from the sight network). At this time the demand departures from i-th node to the other nodes are defined with probability

$$1 - \sum_{j=1}^{U} r_{ij}$$

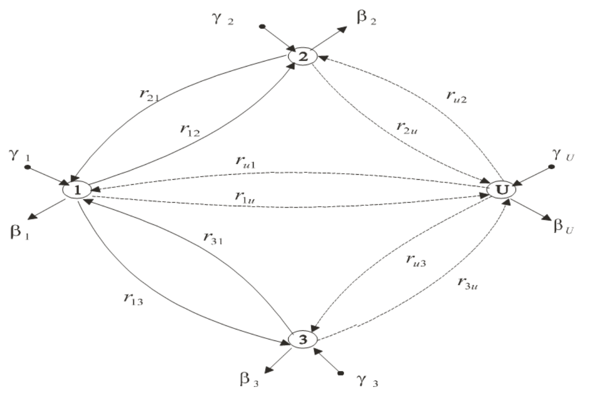


Figure 10. Model of IPC communication system (U=4).

6.1. Analytical Model of Workstations as M/M/m System

Let U be a node number of the whole transport system. For every node of NOW (i-th node according Figure 11.) we define the following parameters

- λ_i - the whole number of incoming demands to the i-th node, that is the sum both of external and internal inputs to the i-th node
- $\gamma = \sum_{i=1}^{U} \gamma_i$ represent the sum of individual total extern intensities in the NOW
- λ_{ij} - the whole input flow to the j-th communication channel at i-th node
- $E(t_q)_i$ - the average servicing time in the program queue (the waiting in a queue and servicing time) in the i-th node
- $E(t_q)_{ij}$ - the average servicing time of the j-th queue of the communication channel (the queue waiting time and servicing time) at i-th node.

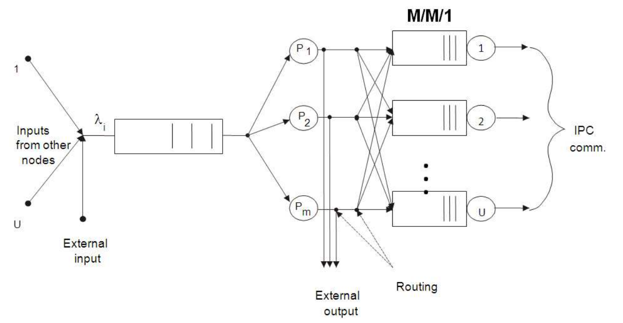


Figure 11. Mathematical model of i-th node.

Then the whole extern input flow to the transport network is given as

$$\gamma = \sum_{i=1}^{U} \gamma_i \quad \text{and} \quad \lambda_i = \sum_{i=1}^{u} \lambda_{ij} + \beta_i$$

where β_i represents the intern output from i-th node (finished parallel programs in this node) which is not further transmitted and is therefore not entering to the $(LO)_i$. Then the whole delay we can modeled as

$$E(t_q)_{now} = \frac{1}{\gamma}\left[\sum_{i=1}^{U}\left(\lambda_i \cdot E(t_q)_i + \sum_{j=1}^{u_i} \lambda_{ij} \cdot E(t_q)_{ij}\right)\right]$$

, where $\dfrac{\lambda_i \cdot E(t_q)_i}{\gamma}$ and $\dfrac{\lambda_{ij} \cdot E(t_q)_{ij}}{\gamma}$

define individual contribution of computation queue delay (M/M/m) and communication channel delay (M/M/1) of every node to the whole delay. For establishing $E(t_q)_i$ for computation queue delay it is necessary to know λ_i as the whole intensity of the input flow to the message queue where $\lambda_i = \gamma_i +$ all intern inputs flow to i-th node.

The intern input flow to i-th node is defined as the input from other connected nodes. We can express it in two ways

- through solving a system of linear equations in matrix form as $\bar{\lambda} = \gamma + \bar{\lambda} \cdot \bar{R}$
- using of two data structures in form of tables and that is the routing table (RT) and destination probability tables (DPT).

In related model the routing table creates deterministic logical way from source i to the destination j. Concretely RT(i,j) has index (1,...,N) of the next node on the route from i to j. This assumption of the fixed routing is not rare. We have proved also experimental, that the fix routing produces good analytical results in comparison to the alternate adaptive routing in a concrete communication network. The destination probability table destiny for each i,j pair the probability, that the message which outstands in node i is destined for node j. This table with n x n dimension and elements DPT (i,j) terminates which fraction of the whole extern input γ_i has the destination j, that is $\gamma_i \cdot DTP(i,j)$. A path through the transport network we can define as the sequence ($x_1, x_2, \dots, x_m$) in which

1) exist physical communication channel, which connects x_k a x_{k1}, k = 1,2, ..., m - 1
2) x_j a $x_k, \forall$j,k j ≠ k (they do not exist loops).

We can define path with record "path (j→k,i)" as expression of the ordered sequence nodes, which are on the route from node j to the node k and they pass step by step through nodes i. That is x_i=j, x_m=k, x_p=i and $1 < p \le m$.

We define then $\sum_{k \in path(j \to k, i)}^{U}$

as the summation over the set of all destination nodes k so, that node i lies on the route from the source node j. Then we get the following relation

The intern input flow to i-th node = $\sum_{j=1}^{U}\sum_{k=1}^{U} \gamma_j \cdot DTP\ (j,k)$, for $j \ne i$, k ∈ path (j → k, i)

and whole input flow to node i as

$$\lambda_i = \gamma_i + \sum_{j=1}^{U}\sum_{k=1}^{U} \gamma_i \cdot DTP\ (j,k),$$

for $j \neq i$, $k \in$ path $(j \to k, i)$

We supposed also that the incoming demands are exponential distributed and that queue servicing algorithm is FIFO (First In – First Out). The program queue PQ_i is servicing through one or more the same computation processors, which performed incoming demands (parallel processes). In demand servicing in a given node could be two possibilities

- demand will be routed to another node of the transport networks by their placing to the one of the used communication channel (IPC communication)
- demand is in the addressed node and she will leave communication network (finishing of parallel algorithms in this node) of the given node).

To every communication channel is set the queue of the given communication lines (LQ), which stores the demands (their pointers) who are awaiting the communication through this communication channel. Also in this case we supposed its unlimited capacity, exponential interarrival time distribution of input messages and the servicing algorithm FIFO. Every communication line queue has its communication capacity S_{ij} (in data units per second). Because we supposed the exponential demand length distribution the servicing time is exponential distributed too with average servicing time $1/\mu S_{ij}$, where μ is the average message length and S_{ij} is the communication capacity of node i and of communication channel j. For simplicity we will assume, as it is obvious, that S_{ij} is a part of μ. To find the average waiting time in the queue of the communication system we consider the model of one communication queue part node as M/M/1 queuing theory system according Figure 12.

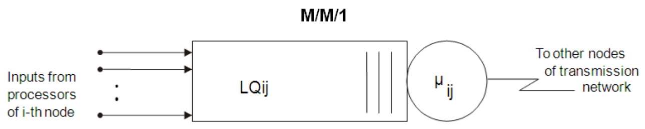

M/M/1
Inputs from processors of i-th node
LQij
μ ij
To other nodes of transmission network

Figure 12. *Model of one M/M/1 communication channel of the i-th node.*

The total incoming flow to the communication channel j at node i which is given through the value λ_{ij} and we can determine it with using of routing table and destination probability table in the same way as for the value λ_i. Then ρ_{ij} as the utilization of the communication channel j at the node i is given as

$$\rho_{ij} = \frac{\lambda_{ij}}{\mu\, S_{ij}}$$

The total average delay time in the queue $E(t_q)_i$ is

$$E(t_q)_{ij} = \frac{1}{\mu_{ij} - \lambda_{ij}}$$

If we now substitute the values for T_i and T_{ij} to the relation for T we can get finally the relation for the total average delay time of whole transport system as

$$E(t_q)_{now} = \frac{1}{\gamma}\left[\sum_{i=1}^{U}\left(\lambda_i \cdot \frac{1}{\mu_i - \lambda_i} + \sum_{j=1}^{u_i} \lambda_{ij} \cdot \frac{1}{\mu_{ij} - \lambda_{ij}}\right)\right]$$

6.2. Suggestion and Derivation of more Precise Models

6.2.1. Model with M/D/m and M/D/1 Systems

The used model were build on assumptions of modeling incoming demands to program queue as Poisson input stream and of the exponential interarrival time between communication inputs to the communication channels. The idea of the previous models were the presumption of decomposition to the individual independent channels together with the independence presumption of the demand length, that is demand lengths are derived on the basis of the probability density function $p_i = \mu\, e^{-\mu t}$ for $t > 0$ and $f(t)=0$ for $t \leq 0$ always at its input to the node. On this basis it was possible to model every used communication channel as the queuing theory system M/M/1 and to derive the average value of delay individually for every channel too. The whole end-to-end delay was then simply the sum of the individual delays of the every used communication channel.

These conditions are not fulfilled for every input load, for all architectures of node and for the real character of processor service time distributions. These changes could cause imprecise results. To improve the mentioned problems we suggested the behavior analysis of the modeled NOW module improved analytical model (Figure 13), which will be extend the used analytical model to more precise analytical model supposing that

- we consider to model computation activities in every node of NOW network as M/D/m system
- we consider an individual communication channels in i-th node as M/D/1 systems. In this way we can take into account also the influence of real non exponential nature of the interarrival time of inputs to the communication channels.

These corrections may to contribute to precise behavior analysis of the NOW network for the typical communication activities and for the variable input loads. According defined assumption to modeling of the computation processors we use the M/D/m queuing theory systems according Figure 13. To find the average program queue delay we used the approximation formula for M/D/m queuing theory system [14, 15] according as

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

, in which

- ρ_i - is the processor utilization at i-th node for all used processors
- m_i - is the number of used processors at i-th node
- $E(t_w)(M/D/1)$, $E(t_w)$ (M/M/1) and $E(t_w)$ (M/M/m) are the average queue delay values for the queuing theory

systems M/D/1, M/M/1 and M/M/m respectively

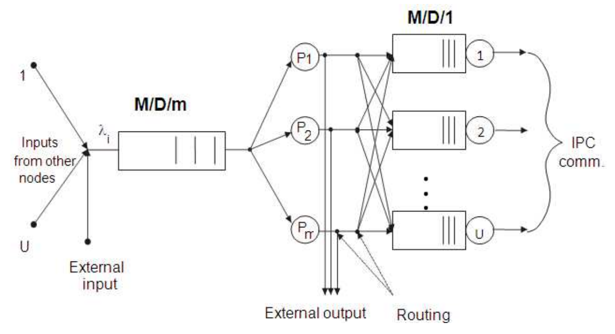


Figure 13. *Precise mathematical model of i-th node.*

The chosen approximation formulae we selected from two following points

- for his simply calculation
- if the number of used processors equals one the used relation gives the exact solution, that is W(M/D/1) system. Such number of processors is often used in praxis
- if the number of processors greater than one (m_i>1) the used relation generate a relative error, which is not greater as 1%. This fact we verified and confirmed through simulation experiments.

Let $\overline{x_i}$ define the fixed processing time of the i-th node processors and $E(t_w)_i$ (PQ) the average program queue delay in the i-th node. Then ρ_i, as the utilization of the i-th node, is given as

$$\rho_i = \frac{\lambda_i . \overline{x_i}}{m_i}$$

Then the average waiting time in PQ queue $E(t_w)_i(M/D/m_i)$ is given trough the following relations

$$E(t_w)_i(M/D/1) = \frac{\rho_i \cdot \overline{x_i}}{2(1-\rho_i)}$$

$$E(t_w)_i(M/M/1) = \frac{\rho_i \cdot \overline{x_i}}{1-\rho_i}$$

$$E(t_w)_i(M/M/m_i) = \frac{\dfrac{(m_i \cdot \rho_i)^{m_i}}{m_i!(1-\rho_i)}}{\sum\limits_{j=0}^{m_i-1}\left[\dfrac{(m_i \cdot \rho_i)^j}{j!} + \dfrac{(m_i \cdot \rho_i)^{m_i}}{m_i!(''1-\rho_i)}\right] \cdot \dfrac{\dfrac{\overline{x_i}}{m_i}}{(1-\rho_i)}}$$

By substituting relations for ρ_i, $E(t_w)_i(M/D/1)$, $E(t_w)_i(M/M/1)$ and $E(t_w)_i(M/M/mi)$ in the relation for $E(t_w)_i(M/D/m_i)$ we can determine $E(t_w)_i(PQ)$. Then the total average delay for the communication activities in i-th node is simply the sum of average message queue delay (MQ) plus the fixed processing time

$$E(t_w)_i = E(t_w)_i(PQ) + \overline{x_i}$$

To find the average waiting time in the queue of the communication system we consider the model of one communication queue part node as M/M/1 queuing theory system according Figure 12. Let $\overline{x_{ij}}$ determine the average servicing time for channel j at the node i. Then ρ_{ij} as the utilization of the communication channel j at the node i is given as

$$\rho_{ij} = \frac{\lambda_{ij}\,\overline{x_{ij}}}{S_{ij}}$$

where S_{ij} is the communication channel speed of j-th node. For simplicity we will assume that S_{ij} =1. The total incoming flow to the communication channel j at node i which is given through the value λ_{ij} and we can determine it with using of routing table and destination probability table in the same way as for a value λ_i. Let $E(t_w)_{ij}(LQ)$ be the average waiting queue time for communication channel j at the node i. Then

$$E(t_w)_{ij}(LQ) = \frac{\rho_{ij} \cdot \overline{x_{ij}}}{(1-\rho_{ij})}$$

The total average delay value is the queue $E(t_w)_{ij}$ is given then as

$$E(t_w)_{ij} = E(t_w)_{ij}(PQ) + \overline{x_{ij}} = \frac{\rho_{ij} \cdot \overline{x_{ij}}}{(1-\rho_{ij})} + \overline{x_{ij}}$$

There If we now substitute the values for $E(t_q)_i$ and $E(t_q)_{ij}$ to the relation for $E(t_q)_{now}$ we can get finally the relation for the total average delay time of whole NOW model is given as

$$E(t_q)_{now} = \frac{1}{\gamma}\left[\sum_{i=1}^{U}\left(E(t_w)_i(PQ) + \overline{x_i}\right) + \sum_{j=1}^{u_i}\left(E(t_w)_{ij}(LQ) + \overline{x_{ij}}\right)\right]$$

6.2.2. Other Real Analytical Models

6.2.2.1. Analytical Model with M/M/m and M/D/1 Systems

This model is mixture of analyzed model. The first part of final total average time $E(t_q)_i$ we get from chapter 6.1 and second part from 6.2.1 one. Then for $E(t_q)_{now}$ we can get finally

$$E(t_q)_{now} = \frac{1}{\gamma}\left[\sum_{i=1}^{U}\left(\lambda_i \cdot \frac{1}{\mu_i - \lambda_i} + \sum_{j=1}^{u_i}\left(E(t_w)_{ij}(LQ) + \overline{x_{ij}}\right)\right)\right]$$

6.2.2.2. Model with M/D/m and M/M/1 Systems

In this model the first part of final total average time $E(t_q)_i$ we can also get from chapter 6.2.1 and second part from 6.1 respectively. Then for $E(t_q)_{now}$ we get for this model finally

$$E(t_q)_{now} = \frac{1}{\gamma}\left[\sum_{i=1}^{U}\left(\left(E(t_w)_i(PQ) + \overline{x_i}\right) + \sum_{j=1}^{u_i}\lambda_{ij} \cdot \frac{1}{\mu_{ij} - \lambda_{ij}}\right)\right]$$

7. Analytical Model of Real Grid Systems

We have defined Grid system as network of NOW network modules. Let N is the number of individual NOW networks or similar clusters. Then final total average time $E(t_q)_{grid}$

$$E(t_q)_{grid} = \frac{1}{\alpha}\left[\sum_{i=1}^{N} E(t_q)_{i\ now}\right]$$

where

- $\alpha = \sum_{i=1}^{N} \gamma_i$ represent the sum of individual total extern intensities to the i-th NOW module in the Grid
- $E(t_q)_{i\ now}$ correspondent to individual average times in i- th NOW module (i=1, 2, … N).

8. Results

Figure 14 and Figure 15 represent results and relative errors for the average value of the total message delay in the 5-noded communication network so for classical analytical model (M/M/m + M/M/1) as for developed more precise analytical model (M/D/m + M/D/1) in which for multiprocessor's node activities we consider very real fixed latency. The same fixed delay was included to the average communication delay at each node and in simulation model too. These assumptions correspondence to the same communication speeds in each node's communication channel. If used communication channels do not have the same communication speeds then communication latencies are different constants. In both considered analytical models (M/M/m + M/M/1, M/D/m + M/D/1) performed experiments have proved that decreasing of processor utilization ρ cause decreasing of total average delay in NOW module $E(t_q)_{now}$. Therefore parallel processes are waiting in parallel processes queues shorter time. In contrary decreasing of node's communication channel speed increase communication channel utilization and then data of parallel processes have to wait longer in communication channel queues and increase the total node's latency. Tested results have also proved the influence of real non exponential nature of the input inter-arrival time to node's communication channels. In relation to it the analytical model M/D/m + M/D/1 provides best results and the analytical model M/M/m + M/M/1 the worst ones. The results for other possible mixed analytical models (M/M/m + M/D/1, M/D/m + M/M/1) provide results between the best and worst solutions. For simplicity deterministic time to perform parallel processes at node's multiprocessor activities (the servicing time of PQ queue) was settled to 8μs and the extern input flow for each node was the same constant too.

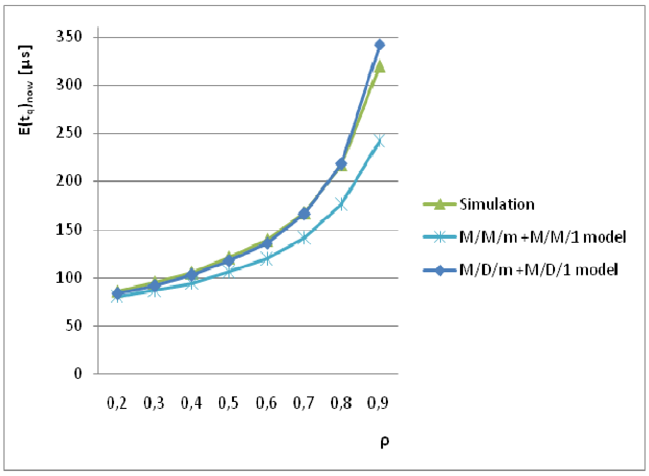


Figure 14. *Comparison of analyzed models.*

To vary node's processor utilization we modified the extern input flow in the same manner for each node of NOW module. For both analytical models (the best and the worst cases) are at Figure 15 the relative errors in relation to simulation results. The best analytical model (M/D/m + M/D/1) provides very precision results in the whole range of input workload of multiprocessors and every communication channel's utilization with relative error, which does not exceed 6.2% and in most cases are in the range up to 5%. This is very important to project heavily loaded NOW network module (from about 80 to 90%), where the accurate results are to be in bad need of to avoid any bottleneck congestions or some other system instabilities. The performed comparison of this best analytical model to analytical model (M/M/m + M/M/1) according Figure 12 show improvements in all range of input node's multiprocessor loads (from 20 to 90%).

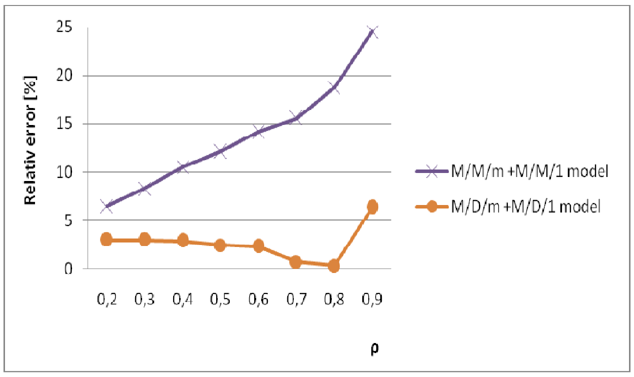


Figure 15. *Relative errors of analyzed models.*

The relative errors of worst analytical model are from 7 to 25%. This is due influences of processes queues delays, the nature of interarrival input to the communication channel in the case of high processor utilization. All developed analytical models could be applied also for large NOW networks practically without any increasing of the computation time in comparison to simulation method because of their explained module's structure based on NOW module. Simulation models require oft three orders of magnitude more computation time for testing such a massive metacomputer. Therefore limiting factor of the developed analytical models will not be computation time, but space complexity of memories for needed RT and DPT tables. These needed RT and DPT tables require $O(n^2)$ memory cells, thus limiting the network analysis to the number of N nodes about 100-200 for the common SMP multiprocessor. In case of possible solving system of linear equations (SLE) to find in analytical way node's λ_i and λ_{ij} respectively input intensities, most parallel algorithms use to its solution Gauss elimination method (GEM). These GEM parallel algorithms have computation complexity given as $O(n^3)$ floating point multiplications and a similar number of additions [2, 9]. These values are however adequate to handle most existing communication network of based NOW module. In addition to it also for any future massive metacomputers we would be always used hierarchically modular architecture, which consist on such simpler NOW modules.

9. Conclusion and Perspectives

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 (for example known results of queuing theory) to performance evaluation of sequential computers because of their high number of not predictable parameters. Secondly endless user demands to increase computer performance were to be done more quickly through continues technology improvements and computer architecture changes. Application incorporation of various forms of parallel principles for a long time create more stabile conditions to apply performance evaluation methods mainly for parallel computers (actually dominant using of SMP multiprocessors and multicores, NOW and Grid systems) open more possibilities to apply mainly ~~a~~ queuing theory results to analyze performance of parallel structured computers. This implies one of known queuing theory results that many inputs to queuing theory system, which create shared stream and which are generating at various independent resources by chance, could be a very good approximation of Poisson distribution as a basic assumption to solve such systems in analytical way. Therefore we are able to model parallel computing nodes (multiprocessor, multicores, workstations etc.) of any actually dominant or perspective parallel computer (SMP, NOW, Grid, metacomputer) as M/D/m queuing theory systems and computing node's communication channels as M/D/1 queuing theory systems respectively.

Then such very flexible modeling tool (queuing theory), based on preferred analytical solutions show real paths to a very effective and practical performance analysis tool including massive NOW networks or another types of massive computer networks (metacomputer, Grid).

In summary developed more precise analytical models could be applied to performance modeling of dominant parallel computers and that in following typical cases

- single computing node based on SMP parallel computer (multiprocessors or multicores)
- NOW based on workstations (multiprocessors or multicores)
- Grid (Network of NOW network modules)
- mixed parallel computers (SMP, NOW, Grid)
- metacomputers (massive Grid).

For our further research work in relation to dominant trends in parallel computers (SMP, NOW, Grid), based of powerful workstations, we will be looking for preferred analytical models in which could be to study load balancing, inter-process communication (IPC) in both parallel and distributed computing, effective transport protocols, influence of various parallel computer architectures, performance prediction etc. We would be also like to analyze

- role of adaptive routing in considered analytical models [5]
- to prove, or to indicate experimentally, the role of the independence assumption, if we are looking for higher moments of overhead latencies (IPX communication – parallel and distributed computing, synchronization, parallelization, architecture etc.)
- to verify analytical models also for node's limited resources capacities – buffers, communication channels etc., and for other existing queue servicing algorithms than standard assumed FIFO (First in First out) [12, 15].

Acknowledgements

This work was done within the project "Modeling, optimization and prediction of parallel computers and algorithms" at University of Zilina. The authors gratefully acknowledge crucial 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, 200 pp., August 2010

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

[3] Cepciansky G., Schwartz L., Stochastic processes with discrete states, LAP Lambert, Germany, 109 pp., 2013

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

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

[6] Dubois M., Annavaram M., Stenstrom P., Parallel Computer Organisation and Design, 560 pages, 2012

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

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

[9] Giambene G., Queueing theory and telecommunications, Springer, 585 pages, 2005

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

[11] Hanuliak P., Analytical method of performance prediction in parallel algorithms, The Open Cybernetics and Systemics Journal, United Kingdom, pp. 38-47, 2012

[12] Hanuliak P., Hanuliak M., Performance modeling of SMP parallel computers, pp. 1-18, Int. Journal of Science, Commerce and Humanities (IJSCH), , United Kingdom, Vol. 1, No 5, pp. 243/261, July 2013

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

[14] Hanuliak M., Hanuliak I., To the correction of analytical models for computer based communication systems, Kybernetes, Volume 35, No. 9, 1492-1504, United Kingdom, 2006

[15] Harchol-Balter Mor, Performance modeling and design of computer systems, Cambridge University Press, 576 pages, 2013

[16] Hwang K. and coll., Distributed and Parallel Computing, Morgan Kaufmann, 472 pages, 2011

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

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

[19] Kirk D. B., Hwu W. W., Programming massively parallel processors, Morgan Kaufmann, 280 pages, 2010

[20] Kostin A., Ilushechkina L., Modelling and simulation of distributed systems, Imperial College Press, 440 pages, Jun 2010.

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

[22] 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, pp. 585-591, 2011

[23] Le Boudec Jean-Yves, Performance evaluation of computer and communication systems, CRC Press, 300 pages, 2011

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

[25] Miller S., Probability and Random Processes, 2nd edition, Academic Press, Elsevier Science, 552 pages, 2012

[26] Misra Ch. S., Woungang I., Selected topics in communication network and distributed systems, Imperial college press, 808 pages, April 2010,

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

[28] Peterson L. L., Davie B. C., Computer networks – a system approach, Morgan Kaufmann, 920 pages, 2011

[29] Resch M. M., Supercomputers in Grids, Int. Journal of Grid and HPC, No.1, p 1 - 9, January- March 2009

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

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