*Transport and Telecommunication, 2010, Volume 11, No 4, 66–74*
*Transport and Telecommunication Institute, Lomonosova 1, Riga, LV-1019, Latvia*

# COMPUTING EFFICIENCY METRICS FOR SYNERGIC INTELLIGENT TRANSPORTATION SYSTEMS

***B. Tsilker[1], S. Orlov[2]***

*Transport and Telecommunication Institute*
*Faculty of Computer Science and Electronics*
*Lomonosova str. 1, Riga, LV-1019, Latvia*
*E-mail: tsilker@tsi.lv[1], sorlov@tsi.lv[2]*

Considerable progress in computing and telecommunications opened new approaches for solving of transportation problems, affected in the concept of intelligent transportation systems (ITS). Technically such systems represent a set of interacting computational nodes with various sensors, and can be considered as distributed computer systems. Distributed nature of the ITS implies parallelization of solvable transportation tasks in conjunction with their distributed realization.

Under efficiency of parallelized calculations we presuppose several aspects. Three of them are picked out in the work: calculation speed, efficiency of system scaling, and efficiency of parallel computations as compared to sequential ones. Typical metrics for numerical characterization of parallelized computations form three groups: index of parallelism and speedup (*PI* and *S*), efficiency and utilization (*E* and *U*), redundancy and compression (*R* and *C*).

The main peculiarity of synergic intelligent transportation systems shows up in bulk of intercommunications, substantially affecting the indexes of the system. Present work focuses on influence of these communication overheads on overall efficiency of the synergic ITS.

**Keywords:** intelligent transportation system, parallelization, distributed systems, efficiency, performance metrics

## 1. Introduction

Information technologies (IT) have transformed many industries. In respect to transportation systems such transformations started in the past decade and are now in the early stages. IT enables elements within the transportation system — vehicles, roads, traffic lights, message signs, etc. — to become intelligent by embedding them with microchips and sensors and empowering them to communicate with each other through wireless technologies. Transportation systems realizing this approach are known as intelligent transportation systems (ITS). According to definition of the Intelligent Transportation Systems Society ITS are those utilizing synergistic technologies and systems engineering concepts to develop and improve transportation systems of all kinds.

Through the use of advanced computing, control, and communication technologies, ITS promises to greatly improve the efficiency and safety of the existing surface transportation system and reduce the transportation-related energy consumption and negative environmental impact. ITS encompass a broad range of wireless and wire line communications-based information and electronics technologies. Depending on nature of the solvable transportation tasks, ITS is made up of 16 types of technology-based systems (Fig. 1).

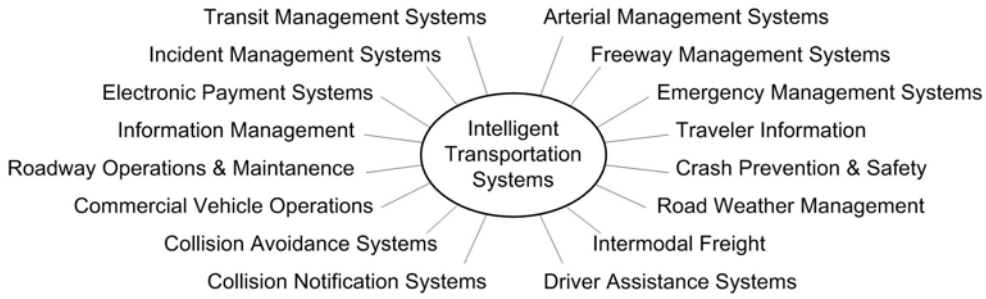


*Figure 1.* Classification of ITS applications

In less detail all ITS types are divided into intelligent infrastructure systems and intelligent vehicle systems.

## 2. Specificity of Parallel and Distributed Computing

The variety of transportation tasks stipulates a spectrum of ITS construction methods, and, of course, differing requirements to ITS hardware and software architecture. However in any case each system as much as possible must be economically and technically effective. It's clear, that for the estimation of technical-and-economic efficiency it is necessary to define the most suitable criteria and system of indexes, numerically characterizing the degree of ITS compliance with these criteria.

From the presented classification of transportation tasks (Fig. 1) follows that the majority of ITS applications on their synergic nature presume distributed solution, where outcome is reached due to cooperative processing of information, incoming from numerous sources, for example, from different sensors. Information processing, as a rule, is also decentralized. By other words, such ITS represent distributed computer systems. Only limited range of ITS is realizable by centralized computer systems, but even here, by reason of high performance requirements, parallel computer systems are usually used. Thus, at examining of ITS estimation, we must talk about parallel and distributed computer systems.

The problem of efficiency estimation for parallel systems is well studied [1-3]. By now exists universally recognized harmonious system of indexes, fully applicable to ITS of this type. On the other hand, universal evaluation metrics for distributed systems at the present time are unknown. Typical attempts usually come to separate estimation of distributed systems' components and getting some integral index based on these particular appraisals. Besides of complexity of such approach realization, it imperfectly takes into account influence of particular components interaction on the total system efficiency.

We offer another approach to efficiency estimation of ITS having distributed nature, based on considering of distributed calculations as a special case of parallel calculations. At distributed calculations each task is actually segmented, i.e. is partitioned on concurrent subtasks. The main property of distributed computing is in extra costs, arising due to interaction of system constituents. Because of ideological closeness of parallel and distributed calculations it is suggested to take the advantage of parallel system efficiency indexes by mapping them on the distributed systems.

## 3. Overheads in Parallel and Distributed Computations

Overheads in parallel and distributed computations fall in three groups:

- intercommunications (typically the most significant overhead in distributed computations);
- idling (processor may become idle because of load imbalance, synchronization, and presence of serial computation);
- excess computations (difference in computation fulfilled by the parallel program and the best sequential program is the excess computation overhead incurred by the parallel program).

Parallel overhead is encapsulated into a single expression referred to as the overhead function. Overhead function (or total overhead), $T_0$, of a parallel system is defined as the total time collectively spent by all $n$ processing elements over and above time $T(1)$, required by the fastest known serial algorithm for solving the same problem on a single processing element:

$$T_0 = n \times T(n) - T(1) .$$

## 4. Communication Overhead in Distributed Computing

For determining communication time between elements of distributed computations $t_{comm}$ three parameters usually are used:

- Start-up time ($t_s$): The time required to handle a message at the sending processor including the time to prepare the message, the time to execute the routing algorithm, and the time to establish an interface between the local processor and router.
- Per-hop time ($t_h$): The time, also known as node latency, for message header to travel between two directly connected processors.
- Per-word transfer time ($t_w$): The time for a word traverse a link. If the channel bandwidth is $r$ words per second, then per-word transfer time is $t_w = \frac{1}{r}$.

Thus, $t_{comm} = t_s + t_h + t_w$.

Consider two routing modes: store-and-forward routing (Fig. 2,*a*) and cut-through routing (Fig. 2,*b*).

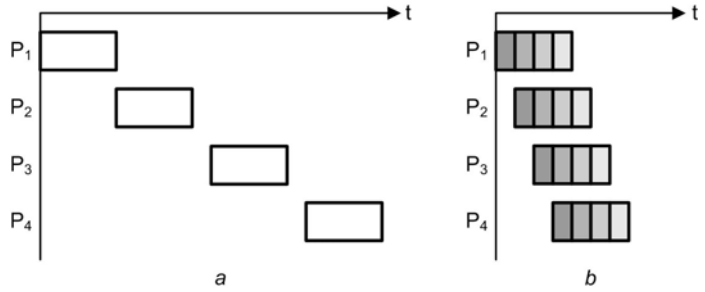


*Figure 2.* Routing modes: *a* – store-and-forward; *b* – cut-through

At store-and-forward routing a message is traversing a path with multiple links: each intermediate processor on the path forwards the message to the next processor after it has received and stored the entire message. In that case the communication overhead can be described by expression

$$t_{comm} = t_s + (mt_w + t_h)l ,$$

where $m$ — message size in words, $l$ — path length in channel number.

Usually $t_h << mt_w$, therefore the expression is simplified to

$$t_{comm} = t_s + mt_w l .$$

At cut-through routing a message is forwarded at intermediate node without waiting for entire message to arrive with no buffering in memory (busy link causes worm to stall; deadlock may ensure). Here the communication overheads are represented by expression

$$t_{comm} = t_s + mt_w + lt_h .$$

Again, considering $t_h$, to be small compared to $mt_w$, the communication overhead is

$$t_{comm} = t_s + mt_w .$$

## 5. Parallel Computing Metrics

To study the performance of systems or to compare different systems for a given purpose, we must first select some criteria. These criteria are often called metrics in performance evaluation. Different situations need different sets of metrics. Thus, selecting metrics are highly problem oriented. Different metrics may result in totally different values. Hence, selecting proper metrics to fairly evaluate the performance of a system is difficult. Another problem with selecting metrics is that in a real system different metrics may be relevant for different jobs. Good metrics are to be reliable, repeatable, consistent and independent. Additional, but not necessary requirements for good metrics are linearity and easiness of use.

Parallel computing metrics are a system of indexes, making possible quantitative estimation of advantages, got at parallel task solution on $n$ processors, as compared to the serial solution of the same task on single processor. On the other hand, they make it possible to judge about the legality of processor amount growth for solving of given task. Under *parallel computing* will understand the sequence of steps, where each step consists of $i$ operations, implemented simultaneously by a set of $i$ processors, working in parallel. The basis for definition of the mentioned metrics form followings characteristics:

- $n$ — number of processors, used for organization of parallel calculations;
- $O(n)$ — amount of calculations, expressed through the number of operations, executable by $n$ processors during the task solution;
- $T(n)$ — total time of calculations (task solution) with the use of $n$ processors.

We will arrange, that time changes discretely, and a processor executes any operation in time slice. The followings correlations are hereupon valid for time and volume of calculations: $T(1) = O(1)$, $T(n) \leq O(n)$. The last correlation formulates assertion: *time of calculations can be shortened due to distributing of calculation volume on a few processors*.

The number of processors used by a program at a particular point in time defines the *degree of parallelism P*(*t*). The plot of parameter *P* against time is called the *parallelism profile* for the program. Changes in the *P*(*t*) depend on many factors (algorithm, available resources, compiler optimisations, etc.). A typical parallelism profile for a *divide-and-conquer* algorithm is shown on Figure 3.

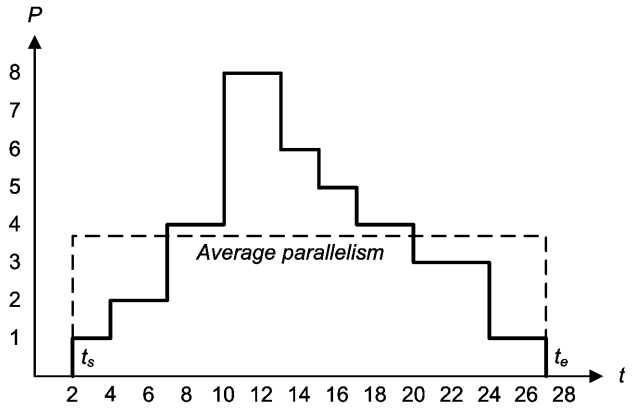


*Figure 3.* Parallelism profile for typical divide-and-conquer algorithm

Suppose that system consists of *n* of homogeneous processors. Let's express the computing capacity of a single processor *Δ.* as an amount of operations in a time unit, not taking into account overheads, related to memory access and data transmission. If for observed period of time (certain amount of time slices) are loaded *i* processors, then *P* = *i*. Thus, a program run from the start time $t_s$ to the moment of completion $t_e$ needs a total of *O*(*n*) operations. *O*(*n*) is proportional to area under to the curve of parallelism profile:

$$O(n) = \Delta \sum_{i=1}^{n} it_i,$$

where $t_i$ is a time interval (common amount of time slices), during which *P* = *i*, and $\sum_{i=1}^{n} t_i = t_e - t_s$ is common time of calculations.

*Average parallelism A* is defined as

$$A = \frac{\sum_{i=1}^{n} it_i}{\sum_{i=1}^{n} t_i}.$$

The parallelism profile on the Figure 3 in course of observation time ($t_s$, $t_e$) grows from 1 to peak value *n* = 8, then go down to 0. Average parallelism *A* = (1 × 5 + 2 × 3 + 3 × 4 + 4 × 6 + 5 × 2 + 6 × 2 + 8 × 3) / (5 + 3 + 4 + 6 + 2 + 2 + 0 + 3) = 93/25 = 3,72.

In essence it is possible to single out four groups of metrics.

The first group characterizes speed of calculations. This group is presented by a pair of metrics – *parallel index* and *speedup*.

*Parallel Index* characterizes average speed of parallel calculations through the amount of the executed operations:

$$PI(n) = \frac{O(n)}{T(n)}.$$

*Speedup* due to program concurrent execution serves as the index of effective speed of calculations. It expresses how much the performance improves when compared to the sequential execution (relative benefit). Speedup is calculated as the ratio of runtime for solving a problem on a

single processor (using the best sequential algorithm) to the time taken for the same problem by a parallel system of *n* processors (at using the best parallel algorithm).

$$S(n) = \frac{T(1)}{T(n)}.$$

Remark in relation to the algorithms of task solution algorithms underline the fact that different algorithms can in the best case appear for serial and parallel realization, and at the estimation of the speedup it is necessary to come exactly from the best algorithms. If the fastest sequential algorithm is not known, the fastest practical approach usually is used.

The second group is formed by *efficiency* and *utilization* metrics, making possible to judge about the efficiency of bringing to the task solution of additional processors.

*Efficiency* characterizes the reasonability of processor number increasing through the fraction of speedup, attained due to parallel calculations, which falls on one processor:

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{nT(n)}.$$

Efficiency measures the fraction of time the processor is usefully employed. It indicates the actual degree of speedup achieved in a system as compared with the maximum possible speedup.

*Utilization* takes into account contribution of every processor at a parallel computing, expressed as amount of operations, executed by a processor in time unit. It indicates the degree to which the system resources were kept busy during execution of the program.

$$U(n) = \frac{O(n)}{nT(n)}.$$

The third group of metrics, — *redundancy* and *compression*, — characterizes efficiency of parallel computing by comparison of volume of calculations, executed at the parallel and serial task solution.

*Redundancy* is a ratio of parallel calculations volume and equivalent successive calculations volume:

$$R(n) = \frac{O(n)}{O(1)}.$$

It shows the extent of the workload increase for going from serial to parallel execution. Importance of this metrics is in that it will proceed not from the relative speedup and efficiency indexes, got from calculation time, but from the absolute indexes, being based on the volume of the executed computational work. The $R$ ($n$) figure indicates to what extent the software parallelism is carried over to the hardware implementation without having extra operations performed.

Note that utilization can be expressed through the redundancy and efficiency metrics:

$$U(n) = \frac{O(n)}{nT(n)} = R(n)E(n).$$

Correlation $T(1) = O(1)$ is taken into account here.

*Compression* is calculated as a reciprocal value of redundancy:

$$C(n) = \frac{O(1)}{O(n)}.$$

Finally, the fourth group is formed by the only metrics – *quality*, joining three considered groups of metrics.

*Quality* is defined as:

$$Q(n) = \frac{T^3(1)}{nT^2(n)O(n)} = S(n)E(n)C(n).$$

This measure is directly related to speedup and efficiency, and inversely related to redundancy R. As far as this metrics ties up speedup, efficiency and compression metrics, it is more objective index of performance improvement due to parallel calculations, and can be considered as an common measure (integral index) defining the whole system performance.

For an example will define the numerical values of metrics in respect to the task, used for parallelism profile concept illustration (Fig. 1). Supposing that the best algorithm for a successive and parallel calculation match, have: $n = 8$; $T(1) = O(1) = O(8) = 93$; $T(8) = 25$. Then:

$$PI(8) = \frac{O(8)}{T(8)} = \frac{93}{25} = 3,72; \qquad S(8) = \frac{T(1)}{T(8)} = \frac{93}{25} = 3,72;$$

$$E(8) = \frac{T(1)}{8T(8)} = \frac{93}{8 \times 25} = 0,465; \quad U(8) = \frac{O(8)}{8T(8)} = \frac{93}{8 \times 25} = 0,465;$$

$$R(8) = \frac{O(8)}{O(1)} = \frac{93}{93} = 1; \qquad C(8) = \frac{O(1)}{O(8)} = \frac{93}{93} = 1;$$

$$Q(8) == S(8)E(8)C(8) = 3,72 \times 0,465 \times 1 = 1,73.$$

At completion note that for the considered metrics the following correlations are true:

$$1 \le S(n) \le PI(n) \le n; \quad 1 \le R(n) \le \frac{1}{E(n)} \le n; \quad \frac{1}{n} \le E(n) \le C(n) \le 1;$$

$$\frac{1}{n} \le E(n) \le U(n) \le 1; \quad Q(n) \le S(n) \le PI(n) \le n.$$

## 6. Speedup Models

There are several laws that have been observed in parallel computing. Three well-known speedup models – fixed-size, fixed-time, and memory-bounded – define the upper limit of speedup achievable by a parallel system. All laws assume that workload in parallel computing comprises of two parts, a sequential part, and a perfectly parallel part. Let $f$ be the percentage of the sequential portion.

In the fixed-size model, known as Amdahl's law [4], it is assumed that the problem size, or workload, is fixed (Fig. 4), and the respective expression for speedup is

$$S(n) = \frac{T(1)}{T(n)} = \frac{1}{f + \frac{1-f)}{n}}.$$

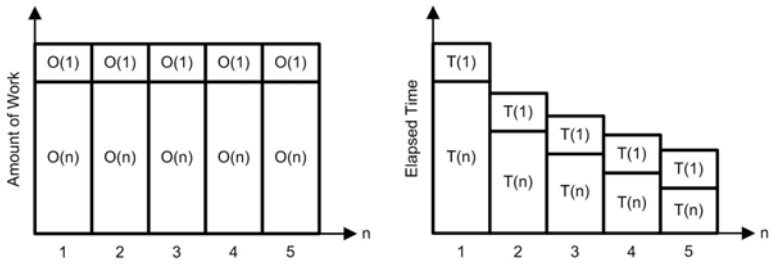


*Figure 4.* Illustration for Amdahl's law statement

John Gustafson [5] proposed a fixed law using time concept to scale the speed up model and remove the fixed load restriction. This law states that problem size scales with the number of processors and idea is to keep all processors busy by increasing the problem size. When the system size is increased and more computing power obtained, we may increase the problem size and perform more operations,

thus obtaining more accurate solution, yet keeping the turnaround time unchanged (Fig.5). This observation is in a basis of Gustafson's law:

$$S(n) = \frac{O(n)}{O(1)} = f + (1-f)n$$

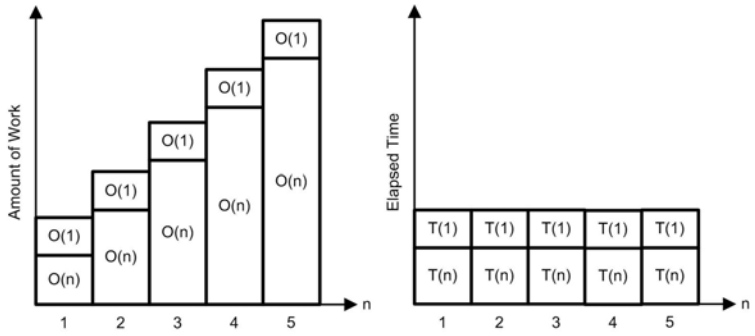


*Figure 5.* Illustration for Gustafson's law statement

Sun and Ni [6] notice that in parallel systems the scaled problem size is limited by memory space. Under the assumption (Fig.6) they developed the memory-bounded speedup law:

$$S(n) = \frac{O(n) / T(n)}{O(1) / T(1)} = \frac{f + (1-f)G(n)}{f + (1-f)\dfrac{G(n)}{n}},$$

where $G(n)$ is the increase of parallel workload as the memory capacity increases $n$ times.

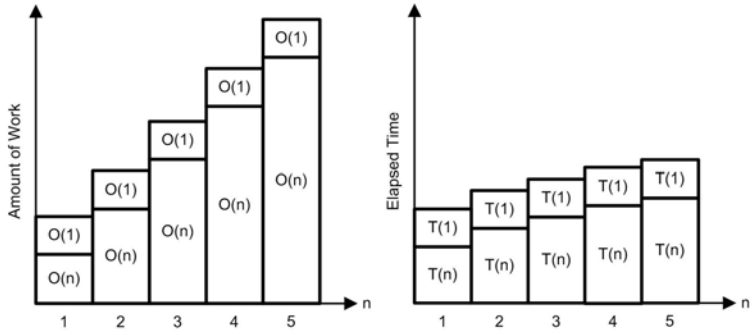


*Figure 6.* Illustration for Sun's and Ni's law statement

At $G(n)=1$ Sun' and Ni's speedup transforms into fixed-sized form (Amdahl), but at $G(n)=n$ – into fixed-time version (Gustafson). $G(n) > n$ represents parallel computing with overheads.

## 7. Metrics for Distributed Computing

For obtaining of similar system of metrics for distributed computing we propose mapping of main parallel computing metrics on the case of distributed computing by taking into account communications overheads. Such mapping results in following "distributed" metrics.

In particular, the speedup metrics goes over:

$$S_d(n) = \frac{T(1)}{T(n) + \sum_{i=1}^{n} t_{comm}(i)},$$

where $t_{comm}(i)$ is communications overhead of *i*-th processor, $\sum_{i=1}^{n} t_{comm}(i)$ is total communication overhead of all *n* processors of the distributed system.

Accordingly the efficiency and utilization metrics are written down in a form of:

$$E_d(n) = \frac{T(1)}{n \times T(n) + \sum_{i=1}^{n} t_{comm}(i)},$$

$$U_d(n) = \frac{O(n)}{n \times T(n) + \sum_{i=1}^{n} t_{comm}(i)}.$$

Formula of integral index *of quality* is modified as follows:

$$Q_d(n) = \frac{T^3(1)}{(T(n) + \sum_{i=1}^{n} t_{comm}(i)) \times (n \times T(n) + \sum_{i=1}^{n} t_{comm}(i)) \times O(n)}.$$

Expressions for the rest metrics remain unchanged.

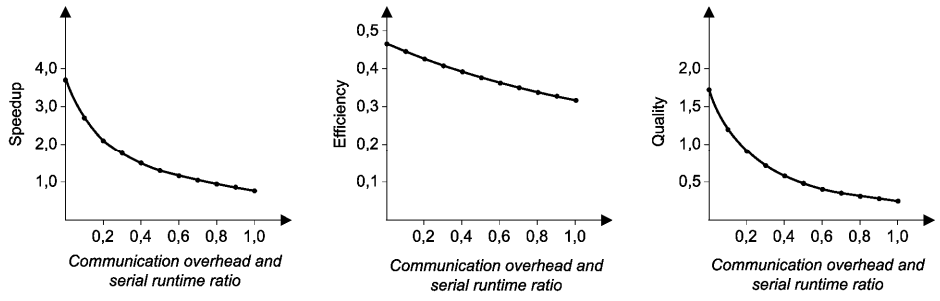


*Figure 7.* Efficiency indexes of a distributed system, realizing the divide-and-conquer algorithm

Figure 7 illustrates influence of communication overhead on the efficiency indexes of a computer system at distributed solution of the before considered divide and concur algorithm. Value on a vertical axis corresponds to absence of communication overhead, i.e. to the parallel system.

## 8. Speedup Models for Distributed Computing

Speedup laws for parallel computing also can be mapped on the distributed computation case by taking into account communication overheads typical for last ones.

Fixed-size speedup low (Amdahl) with overheads transforms in the following expression ($\sum_{i=1}^{n} t_{comm}(i)$ is denoted as $T_{comm}$).

$$S_d(n) = \frac{T(1)}{T(n)} = \frac{1}{f + \frac{1-f}{n} + \frac{T_{comm}}{T(1)}}.$$

Fixed-time Speedup (Gustafson) with overheads will be

$$S_d(n) = \frac{O(n)}{O(1)} = f + (1 - f - \frac{T_{comm}}{T(1)})n.$$

As follows from the expressions the upper speedup bound is not only limited by the sequential bottleneck but also by the average overheads.

## 9. Conclusions

The issue of the day in intelligent transportation systems development still is a task of effective selection and design of the used software and hardware tools. Within the framework of decision of this task, authors had analysed characteristics of given problem domain, singularities of structural organization and functioning for this family of computer systems, had explored their specificity, had offered a set of metrics for the estimation of efficiency of parallel and distributed modes of operation. These metrics permit estimation of integral efficiency of intelligent transportation systems, and take into account basic features of both parallel and distributed computing.

## References

1. Voevodin, V.V., Voevodin, Vl. B. *Parallel computing*. SPb.: BHV-Petersburg, 2002. 608 p. (In Russian)
2. Orlov, S.A., Tsilker, B.J. *Computer organization and systems. 2nd ed.* SPb.: Peter, 2010. 688 p. (In Russian)
3. Zomaya, Albert Y.H. *Parallel and Distributed Computing Handbook: McGraw-Hill Series on Computing Engineering.* New York, 1996.
4. Amdahl, G. M. Validity of the single processor approach to achieving large scale computing capabilities. In: *Proc. AFIPS Conference*, Reston, VA. April 1967. Vol. 30, pp. 483-485.
5. Gustafson,, J. L. Re-evaluating Amdahl's law, *Communications of the ACM*, Vol. 31, No 5, May 1988, pp. 532-533.
6. Sun, X. -H. and Ni, L. M. Scalable problems and memory-bounded speedup, *Journal of Parallel and Distributed Computing*, Vol. 19, 1993, pp. 27-37.