American Journal of Networks and Communications
2014; 3(5-1): 1-14
Published online July 30, 2014 (http://www.sciencepublishinggroup.com/j/ajnc)
doi: 10.11648/j.ajnc.s.2014030501.11
ISSN: 2326-893X (Print); ISSN: 2326-8964 (Online)

Complex modeling of matrix parallel algorithms

Peter Hanuliak

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

Email address:
phanuliak@gmail.com

To cite this article:
Peter Hanuliak. Complex Modeling of Matrix Parallel Algorithms. *American Journal of Networks and Communications.* Special Issue: Parallel Computer and Parallel Algorithms. Vol. 3, No. 5-1, 2014, pp. 1-14. doi: 10.11648/j.ajnc.s.2014030501.11

Abstract: Parallel principles are the most effective way how to increase performance in parallel computing (parallel computers and algorithms too). In this sense the paper is devoted to a complex performance evaluation of matrix parallel algorithms (MPA). At first the paper describes the typical matrix parallel algorithms and then it summarizes common properties of them to complex performance modeling of MPA. To complex performance analysis we are able to take into account all overheads influence performance of parallel algorithms (parallel computer architecture, parallel computation, communication etc.). To be le to analyze MPA in their abstract form we have defined needed decomposition models of MPA. For these decomposition strategies we derived analytical relation for defined complex performance criterions including isoefficiency functions, which allow us to predict performance although for hypothetical parallel computer. In its experimental part the paper considers the achieved results using defined complex performance criterions including issoefficiency function for performance prediction also for hypothetical future parallel computers. Such idea of common abstract analysis could be very useful in deriving complex performance criterions for groups of other similar parallel algorithms (PA) as for example numerical integration PA, optimization PA etc.

Keywords: Parallel Computer, NOW, Grid, Parallel Algorithm (PA), Matrix PA, Decomposition Model, Performance Modeling, Optimization, Overhead Function H (S, P), Inter Process Communication IPC, Performance Prediction, Issoeficiency Function

1. Trends in Parallel Computing

Basic common properties in parallel computing (parallel computers, parallel algorithms) computing, which are reaching continuous demands to performance acceleration, are as follows

- embedded parallel principles on various levels of technical (hardware) and program support means (software) [8]
- using of homogenous shared resources so in computing nodes of parallel computers (processors, cores, computers) as in parallel algorithms too [24]
- using of high speed communication networks reducing communication latency [39]
- increased client/server computing on symmetrical multiple processors or cores (SMP)
- trends to unified modeling of parallel computers (shared memory, distributed memory) and in parallel algorithms (shared memory, distributed memory, hybrid)
- continuous demands to increase mobility and data migration [23]
- the development of hardware neutral parallel programming language, such as Java, provides a virtual computational environment in which computing nodes of parallel computer appear to be homogenous
- continuous improvements in network technology and communication middleware in order to use shared parallel resources in unified manner (cloud computing, Internet computing).

Current trends also in high performance computing (HPC) are to use networks of workstations (NOW, SMP) as a cheaper alternative to traditionally used massively parallel multiprocessors or supercomputers and to profit from unifying of both mentioned disciplines [19]. The individual powerful computing nodes (workstations) could be so single personal computer (PC) as parallel computers based on modern SMP parallel computers implemented within computing node of parallel computer [13, 15]. Based on such modular NOW modules there were are realized high integrated massive parallel computers named as Grid

systems [38]. A member of NOW module or Grid could be any classic supercomputers [35].

2. Performance Evaluation in Parallel Computing

To performance evaluation of parallel computers and parallel algorithms we can use evaluation methods as follows

- analytical
  - application of queuing theory results [11, 21]
  - order (asymptotic) analyze [12, 20]
  - Petri nets [7]
- simulation methods [25]
- experimental
  - benchmarks [28]
  - modeling tools [32]
  - direct measuring [9, 30].

Analytical method 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.

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. They are still only applicable to a restricted class of models (though not as restricted as analytic approaches.) Many approaches increase rapidly their memory and time requirements as the size of the model increases.

Evaluating system performance via experimental measurements is a very useful alternative for computer systems. Measurements can be gathered on existing systems by means of benchmark applications that aim at stressing specific aspects of computers systems. 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.

3. Parallel Algorithms

In principal we can divide parallel algorithms (PA) to the following groups

- parallel algorithm using shared memory (PA_{sm}). These algorithms are developed for parallel computers with shared memory as actual modern symmetrical multiprocessors (SMP) or multicore systems on motherboard
- parallel algorithm using distributed memory (PA_{dm}). These algorithms are developed for parallel computers with distributed memory as actual NOW system and their higher integration forms named as Grid systems
- hybrid PA which combine using of both previous PA (PA_{hyb}). This trend support applied using of NOW consisted from computing nodes based on SMP parallel computers.

The main difference between PA_{sm} and PA_{dm} is in form of inter process communication (IPC) among created parallel processes [18, 33]. Generally we can say that IPC communication in parallel system with shared memory can use more communication possibilities (all the possibilities of communication in shared memory) than in distributed systems (only network communication).

2.1. Developing Steps of PA

The role of programmer is for the given parallel computer and for given application problem to develop the effective parallel algorithm. This task is more complicated in those cases, in which we have to create the conditions for any parallel activities in form of dividing the sequential algorithm to their mutual independent parts named parallel processes. Principally development of any parallel algorithms (shared memory, distributed memory, hybrid) includes performing of the following activities [29, 34].

- decomposition of a complex problem to a set of parallel processes including their data (decomposition model)
- mapping – distribution of decomposed parallel processes to computing nodes of used parallel computer
- inter process communication (IPC) to cooperation (data communications, synchronization, control) of performed parallel processes
- performance optimization (tuning) of developed parallel algorithm (effective PA).

The most important step is to choose for given complex problem optimal decomposition model. To do this there is necessary to understand given complex problem, shared data, applied sequential algorithms (SA) and the flow of SA control [4, 26].

3.1.1. Decomposition Models

Decomposition model defines distribution of given complex problem to its independent parts (parallel processes) in such a way, that they could be performed in a parallel way via computing nodes of used parallel computer. Optimal selection of decomposition model and degree of parallelism are critical conditions to develop effective parallel algorithm. Potential decomposition of given complex problem is crucial for effectiveness of parallel algorithm [16]. The chosen decomposition model then drives the rest of effective parallel program development. This is true is in case of developing new applied PA as in porting serial code. The decomposition model defines structure of PA codes and their data and estimate the optimal topology of needed communication network [27, 31]. The existed decomposition models we have been analyzed in [16].

3.1.2. Mapping

This step allocates created parallel processes to computing nodes of parallel computer for their parallel executions. There is necessary to achieve that every computing node should perform allocated parallel processes (one or more) with at least approximate input loads (load balancing) on real assumption of equal powerful computing nodes. Fulfillment of this condition contributes to optimal parallel execution time.

3.1.3. Inter process Communication

Inter process communication (IPC) represents a needed tool to cooperation of decomposed parallel processes. In general we can say that dominated parts of parallel algorithms are decomposed parallel processes (independent sequential parts) and inter process communication (IPC) among created parallel processes in performing of PA. We have been analyzed IPC communication in detail in [18].

3.1.4. Performance Optimization

After verifying developed parallel algorithm on used parallel computer the further step is performance modeling and its optimization in order to develop effective PA. This step contents analysis of previous steps in such a way to minimize whole execution time latency of parallel computing T(s, p). Performed optimization of T(s, p) for given parallel algorithm depends mainly from following factors

- allocation of balanced input load to used computing nodes of parallel computer (load balancing) [1, 36]
- minimization of accompanying overheads amounts (parallelization, IPC, synchronization control of PA) [14, 22].

To do load balancing we need in case of obvious using of equally powerful computing nodes of PC results of load allocation for given developed PA. In dominated parallel computers (NOW, Grid) there are necessary to reduce (optimize) mainly number of inter process communications IPC (communication complexity) for example by considering of alternative existing decomposition model.

3.2. Complex Performance Evaluation Metrics

To evaluating parallel algorithms we have been defined in [14] complex performance criterions of PA. Tradeoffs among these performance factors are often encountered in real applied PA. We summarize these criterions as follows

- complex parallel execution time T(s, p) including overhead function h(s, p)
- complex speed up S(s, p)
- complex efficiency E(s, p)
- issoeficiency w(s).

4. Typical Matrix Parallel Algorithms

Some of the typical matrix parallel algorithms we have been yet analyzed as follows

- parallel matrix multiplication [16]
- parallel fast discrete Fourier Transform (DFFT) in [14].

We will short describe further typical MPA.

4.1. System of Linear Equations

System of n linear equations (SLE) with n variables x_1, x_2, x_3, ..., x_n, in matrix form is defined as follows [3, 10]

$$A \,.\, X = B$$

where the matrix A is a square matrix of coefficients, B is the vector of the right side and X is a vector of searching unknown as follows

$$A = \begin{pmatrix} a_{11}, a_{12}, a_{13}, \ldots\ a_{1n} \\ a_{21}, a_{22}, a_{23}, \ldots\ a_{2n} \\ \cdots \\ \cdots \\ \cdots \\ a_{n1}, a_{n2}, a_{n3}, \ldots\ a_{nn} \end{pmatrix} \quad B = \begin{pmatrix} a_{1,n+1} \\ a_{2,n+1} \\ \cdot\cdot \\ \cdot\cdot \\ a_{n,n+1} \end{pmatrix} \quad X = \begin{pmatrix} X_{1n} \\ X_2 \\ \cdot\cdot \\ \cdot\cdot \\ X_n \end{pmatrix}$$

4.1.1. Methods of SLE Solving

There is no known universal optimal method of solving systems of linear equations. There are several different ways of solving SLR whereby each of them at fulfillment of defined assumption implies the option of the solution method. In principle, we divide the available methods for exact (finite) and iterative. There exist many various ways how to solve system of linear equations. But there does not exist any optimal way of solving it. The existed methods can be divided into

- exact
  - Cramer rule
  - Gaussian elimination methods (GEM)
  - GEM alternatives
- iterative

4.1.2. Typical Decomposition Models

To parallel solution of SLE by preferred Gauss eliminated method (GEM) the decomposition models are as follows [16]

- allocation of block strips
  - gradually allocation of strips.

In the first allocation method strips are divided to set of strips and to every computing node is assigned one block. Illustration of these allocation methods is at Fig. 1.

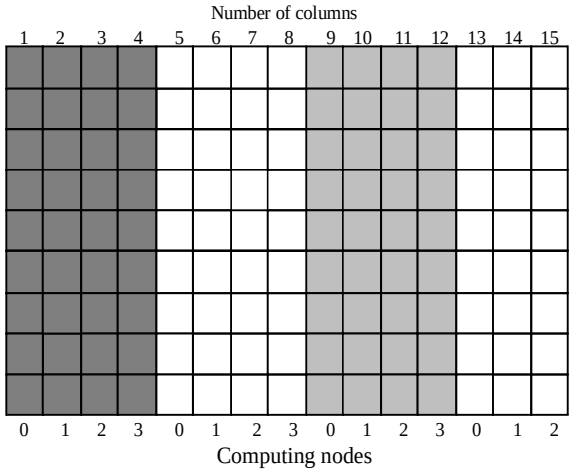


Figure 1. *Allocation of matrix data blocks.*

Another alternative decomposition model with gradual allocation of columns they are allocated columns to individual computing nodes like the card are gradually passing out at games to game participants. Illustration of gradual assignment of columns is at Fig. 2.

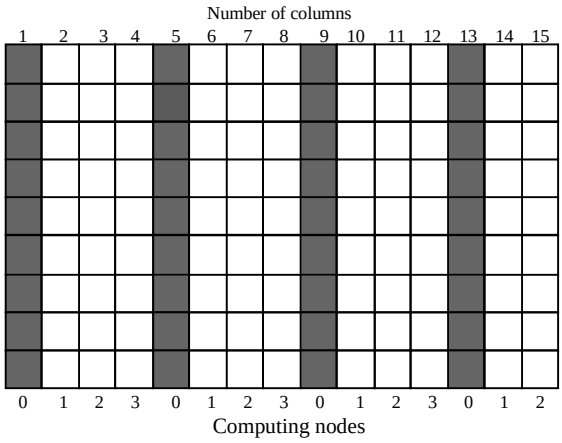


Figure 2. *Allocation of matrix columns.*

4.2. Partial Differential Equations

Partial differential equations (PDE) are the equation involving partial derivates of an unknown function with respect to more than one independent variable. PDEs are of fundamental importance in modeling all types of continuous phenomena in nature. Typical examples are weather forecasting, optimization of aerodynamic shapes, fluid flow, and the like. Simple PDE can be solved directly, but in general it is necessary to approximate the solution on the extensive network of final points by iterative numerical methods [18]. We will confine our attention to PDE with two space independent variables x, y. The needed function we denote as u (x, y). The considered partial derivations we denote as u_{xx}, u_{xy}, u_{yy} etc. For practical use the most important PDE are two ordered equations as follows

- heat equation, $u_t = u_{xx}$
- wave equation, $u_{tt} = u_{xx}$
- Laplace equation $u_{xx} + u_{yy} = 0$.

These three types are the basic types of general linear second order PDR as in follows

$u_{xx} + b\ u_{xy} + c\ u_{yy} + d\ u_x + e\ u_y + f\ u + g = 0$. This equation could be transformed by changing the variables to one of three basic equations, including the members of the lower rows, provided that the coefficients a, b, c are not all equal to zero. Variable $b^2 - 4\ ac$ is refer to as discriminant whereby its value determines the following basic groups PDR of second order

- $b^2 - 4\ ac > 0$, hyperbolic (typical equation for waves).
- $b^2 - 4\ ac = 0$, parabolic (typical of heat transfer)
- $b^2 - 4\ ac < 0$, elliptical (typical is the Laplace equation) [6, 37].

Classification of more general types of PDE is not so clear. When the coefficients are variable, then the type of equation can be modified by changes in the analyzed area and if it is intended at the same time with several equations, each equation can generally be of a different type. Simultaneously analyzed problem may be nonlinear or equation requires more than second order [2, 23]. Nevertheless, the basic used classification of PDE is also used when determining if it is not accurate. Specifically the following types of PDEs are as follows

- hyperbolic. This group is characterized by time dependent processes that are not stabilized at some steady state
- parabolic. Group characterized by a time dependent processes, which tend to the stabilization
- elliptical. Group describes the processes that have reached steady state and are therefore time independent. A typical example is the Laplace equation.

Here we show how to solve in parallel way specific PDE – Laplace equation in two dimensions – by means of a grid computation method that employs finite difference method. Although we focus on this specific problem, the same techniques are used for solving other PDE (Laplace - three dimensional, Poisson equation etc.), extensive approximations calculations on various parallel computers (supercomputers, massive, SMP, NOW, Grid) eventually solving another similar complex problems.

4.2.1. Parallel Application of Iterative Algorithms

Here we show how to solve in parallel way specific PDE – Laplace equation in two dimensions – by means of a grid computation method that employs finite difference method. Although we focus on this specific problem, the same techniques are used for solving other PDE (Laplace - three dimensional, Poisson equation etc.), extensive approximations calculations on various parallel computers (supercomputers, massive, SMP, NOW, Grid) eventually solving another similar complex problems. Laplace equation is a practical example of using iterative methods to its solution. The equation for two dimensions is following

$$\frac{\delta^2\Phi}{\delta x^2}+\frac{\delta^2\Phi}{\delta y^2}=0$$

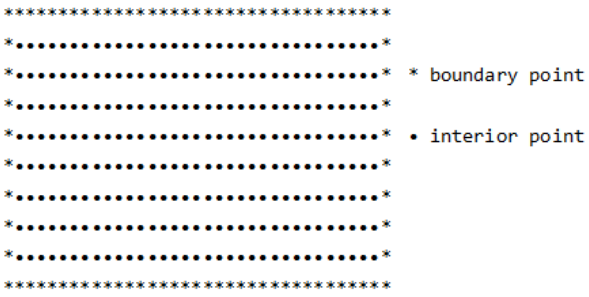


Figure 3. *Grid approximation of Laplace equation.*

Function Φ(x, y) could represent some unknown potential, such as heat, stress etc. Given a two-dimensional region and values for points of the region boundaries, the goal is to approximate the steady-state solution Φ(x, y) for points in the interior by the function u(x, y). We can do this by covering the region with a grid of points (Fig. 1) and to obtain the values of $u(x_i, y_j) = u_{i,j}$.

Let us consider square region (a, b) x (a, b). For coordinates of grid points is valid $x_i = i*h$, $y_j = j*h$, $h = (b-a) / N$ for i,j = 0, 1, ..., N. We replace partial derivations of Φ ~ u(x, y) by the differences of $u_{i,j}$. After substituting we obtain final iteration formulae as

$$X_{i,j}^{(t+1)} = (X_{i-1,j}^{(t)} + X_{i+1,j}^{(t)} + X_{i,j-1}^{(t)} + X_{i,j+1}^{(t)}) / 4$$

orits alternative version

$$X_{i,j}^{(t+1)} = (4\,X_{i,j}^{(t)} + X_{i-1,j}^{(t)} + X_{i+1,j}^{(t)} + X_{i,j-1}^{(t)} + X_{i,j+1}^{(t)}) / 8$$

Each interior point is initialized to some value. The steady-state values of the interior points are then computed by repeated iterations. In each iteration the new point value is set to a combination of the previous values of neighboring points. The computation terminates either after a given number of iterations or when every new value is within some acceptable difference Epsilon > 0 of the previous value.

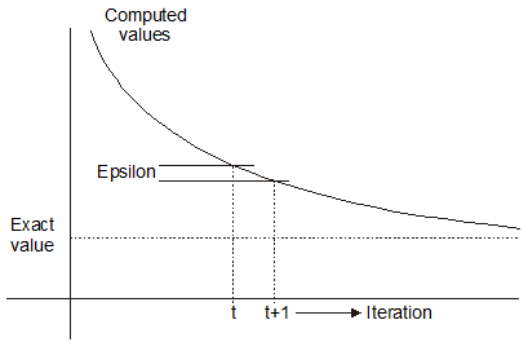


Figure 4. *Convergence rate.*

Illustration of convergence rate of iterative parallel algorithms is at Fig. 4. For the convergence of Gauss-Seidel iterative method is valid the same conditions as for Jacobi iterative method whereby the Gauss-Seidel method converges faster. Given condition is not only necessary but only a sufficient one for the convergence of both methods. In practice, we use both iterative methods also in case of not satisfying of this condition based on them that convergence is influenced also by selection of initial vector.

4.2.1.1. Communication Model

For Jacobi finite difference method a two-dimensional grid is repeatedly updated by replacing the value at each point with some function of the values at a small fixed number of neighboring points. The common approximation structure uses a four-point stencil to update each element $X_{i,j}$ (Fig. 5.).

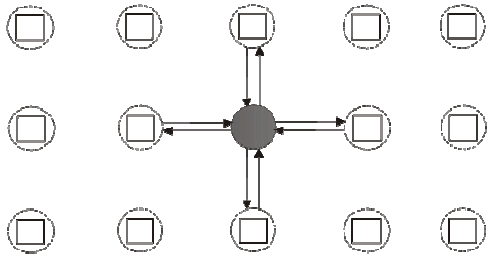


Figure 5. *Communication model for 4 - points approximation*

Similar for more accurate value of any point we can also used more precise multipoint approximation relation, and that for example through approximation of nine points according the pencil at Fig. 6 with following relation

$$X_{i,j}^{(t+1)} = (16\,X_{i-1,j}^{(t)} + 16\,X_{i+1,j}^{(t)} + 16\,X_{i,j-1}^{(t)} + 16\,X_{i,j+1}^{(t)} - X_{i-2,j}^{(t)} - X_{i+2,j}^{(t)} - X_{i,j-2}^{(t)} - X_{i,j+2}^{(t)}) / 60$$

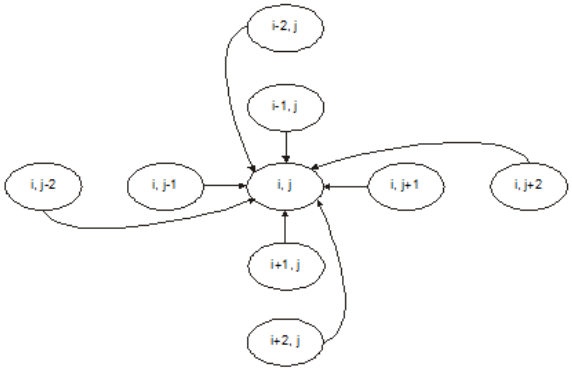


Figure 6. *Stencil with nine points.*

5. Complex analytical performance modeling of MPA

For a complex performance analysis (including overhead function) of matrix parallel algorithms (MPA) we will be consider general shape of a square n x n matrix at Fig. 7.

$$A = \begin{pmatrix} a_{11}, a_{12}, & \dots & , a_{1n} \\ a_{21}, a_{22}, & \dots & , a_{2n} \\ .. & & . \\ .. & & . \\ .. & & . \\ a_{n1}, a_{n2}, & \dots & , a_{nn} \end{pmatrix}$$

Figure 7. *Square matrix n x n.*

The reason for preferred square matrix reducing of parameter number (n = m) in derivation process of complex analytical performance relations (execution time, speed up, efficiency, isoefficiency etc.). Such more transparent approach is supported with following additional reasons

- any rectangular matrix n x m could be transformed into a square matrix n x n either by extending the number of rows (where m < n) or column (if m > n)
- derivation process of performance relations will be the same except for the fact that when considering the complexity of the matrix instead of n^2 (square matrix) we have to consider product n. m .

5.1. Basic Common Characteristics of MPA

Typical characteristics of matrix parallel algorithms are regularities both in the program (matrix computational activities) and also in data structure matrix (matrix elements). Such regularity we refer to as a domain. Matrix computational activities we will represent as $T(s, p)_{comp}$ latency. Considering square matrix n x n sequential computational complexity is given as n^2. Common characteristics of matrix parallel algorithms (MPA) are as follows

- parallelization - matrix itself is well parallelized theoretically up to level of its single data element. But applying such a maximal degree of parallelization could not be effective because of low computation complexity for one matrix element. Therefore we will consider basic matrix decomposition models in group of matrix data elements
- using of domain decomposition models in which domain is represented by data matrix elements (data domain)
- applied matrix date domain decomposition models define that for parallel computation on decomposed parts of matrix data elements there is necessary to perform in a parallel way the whole computation as in sequential matrix algorithms
- to do any computational operation on matrix there is necessary to do this operation on every matrix element or group of them. From performed analysis comes out that at solving typical MPA parallel matrix computations are performed as follow
  - allocated matrix data part of given computing node are repeatedly evaluated according used PA (iteration PA). After every iterations step there is necessary to perform IPC communication to neighboring computing nodes of shared matrix data elements
  - allocated matrix data part of given computing node in one computing step are reduced according used PA to simpler matrix data part (for example GEM PA). After every reduction step there is necessary to perform IPC communication to all other used computing nodes.

Based on these conclusions to modeling of MPA there is necessary to derive needed computational and communication complexity.

5.1.1. Computational Matrix Complexity

5.1.1.1. Sequential

Sequential computational matrix complexity for considered square matrix n x n is given as n^2 (computation on each matrix element). Then the used asymptotic complexity is given as $O(n^2)$.

5.1.1.2. Parallel

Computational parallel matrix complexity Z(s, k) is given as computational complexity of one decomposed parallel process in k computing steps where parameter s we have been defined [14] as working load of given problem. For square matrix $s = n^2$ (sequential matrix complexity). Through matrix decomposition models we are creating p parallel processes (matrix decomposition to p matrix parts) the Z(s, k) will be given as a quotient of computational sequential matrix complexity n^2 and number of decomposed parallel processes p as follows

$$Z(s,k) = Z(s,1)\frac{n^2}{p}$$

where Z(s, 1) represent computational complexity in one computation step. From derived relation the parallel computation time complexity $T(s, p)_{comp}$ is given through quotient of parallel computing time running time of one parallel process (product of its complexity $Z(s, 1)_{comp}$ and a constant t_{c1} as an average value of performed computation operations) through number of decomposed parallel processes p as follows

$$T(s, p)_{comp} = \frac{Z(s,1)_{comp} \,.\, n^2 \,.\, t_{c1}}{p}$$

In MPA we are oft using mapping under the condition n = p. Then we get for $T(s, p)_{comp}$ following simpler relation as

$$T(s, p)_{comp} = Z(s,1)_{comp} \,.\, n \,.\, t_{c1}$$

For this simpler relation asymptotic complexity is given as O (n). At the same time in relation to ideally parallelized MPA and under assumptions of theoretical unlimited

number of computation nodes p mathematical limit of T(s, p)$_{comp}$ is given as

$$T(s,\mathrm{p})_{comp} = \lim_{p\to\infty} \frac{Z(s,1)_{comp} \cdot n^2 \cdot t_{c1}}{p} = 0$$

From this result we can see that a MPA the dominant influence will have mainly communication complexity. Therefore we will exanimate basic matrix decomposition models and their consequences to defined complex performance criterion.

5.2. Basic Matrix Decomposition Models

Supposed efficiency of parallel matrix algorithms (shared memory, distributed memory, hybrid) required to allocate a parallel process to more than one internal element of the square matrix (data elements). Then for decomposition of matrix elements to some groups of matrix data elements we have in principal two basic decomposition models as follows

- decomposition model of n x n matrix to square blocks of matrix elements (parallel process). Illustration example of matrix decomposition to p blocks (B_1, B_2, …, B_p) is at Fig. 8 a. In this case the decomposed blocks consist from at least four matrix data elements.
- decomposition model of n x n to continual matrix strips of matrix elements. Continual strips consist of at least one matrix row or one matrix column. Illustration example of matrix decomposition to p strips (S_1, S_2, …, S_p) is at Fig. 8 b. In this case the decomposed strips consist from at least one matrix row.

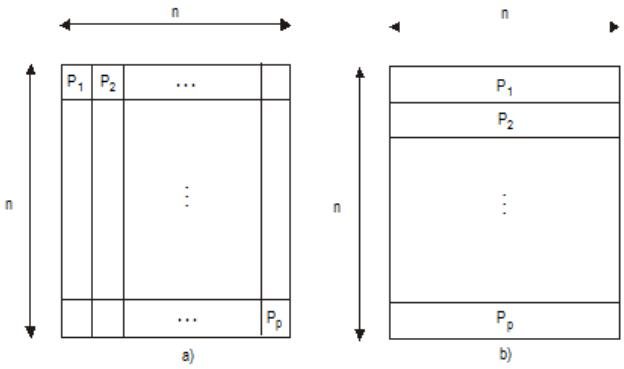


Figure 8. *Matrix decomposition models a) blocks b) strips.*

5.2.1. Decomposition Model to Blocks

For mapping matrix elements in blocks a inter process communication is performed on the four neighboring edges of blocks, which it is necessary in computation flow to exchange. Every parallel process therefore sends four messages and in the same way they receive four messages at the end of every calculation step (Fig. 9. a) supposing that all needed data at every edge are sent as a part of any message).

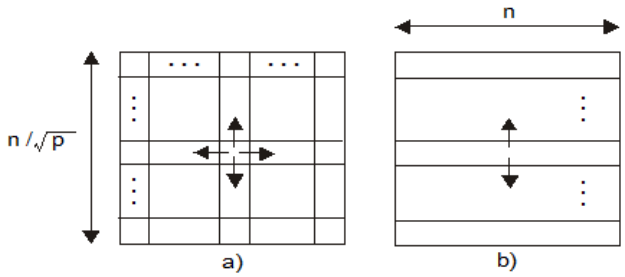


Figure 9. *Communication decomposition models a) blocks b) strips.*

Then the requested communication time for this decomposition model is given as

$$T(s,\ p)_{commb} = 8\,(t_s + \frac{n}{\sqrt{p}}\,t_w)$$

using defined technical communication parameters [18] as follows

- t_s is defined parameter for communication initialization
- t_w is defined parameter for data unit latency.

This equation is correct for p ≥ 9, because only under this assumption it is possible to build at least one square because only then is possible to build one square block with for communication edges. Using these variables for the communication overheads in decomposition method to blocks is correct

$$T(s,\mathrm{p})_{comm} = T(s,\ p)_{commb} = h(s,p) = 8\,(t_s + \frac{n}{\sqrt{p}}\,t_w)$$

Then the requested communication time for this decomposition method is given as

$$T_{comb} = 8\,(t_s + \frac{n}{\sqrt{p}}\,t_w)$$

5.2.2. Matrix Decomposition to Strips

Decomposition method to rows or columns (strips) are algorithmic the same and for their practical using is critical the way how are the matrix elements putting down to matrix. For example in C language are array elements put down from right to left and from bottom to top (step by step building of matrix rows).

In this way it is possible send very simple through specification of the beginning address for a given row and through a number of elements in row (addressing with indexes). Let for every parallel process (strips) two messages are send to neighboring processors and in the same way two messages are received from neighboring processors (Fig. 8 b) supposing that it is possible to transmit for example one row to one message. Communication time for a calculation step T(s, p)$_{comms}$ is then given as

$$T(s,\ p)_{comms} = 4\,(t_s + n\ t_w)$$

Using these variables for the communication overheads in decomposition method to strips is correct

$$T(s,p)_{comm} = T(s,p)_{comms} = h(s,p) = 4(t_s + n\,t_w)$$

The whole time to execute parallel algorithm T(s, p) for decomposition to strips is then given in general as

$$T(s,p) = \frac{n^2.t_c}{p} + 4(t_s + n\,t_w)$$

In this case a communication time for one calculation step does not depend on the number of used calculation processors.

5.3. Complex Analytical Performance Modeling

To complex MPA performance modeling we have been defined as deriving of evaluation criterions of IPA including considering overhead function h(s, p). We summarized derived analytical results as following

Shared results for both decomposition models (blocks, strips)

- execution time of sequential square matrix algorithm T(s, 1)

$$T(s,1)_{comp} = n^2\,t_{c1}$$

- execution time for own parallel computation time of IPA parallel algorithms $T(s,p)_{comp}$

$$T(s,p)_{calc} = \frac{n^2.t_{c1}}{p}$$

- optimal conditions to selection of matrix decomposition model for t_s ,and for t_w respectively

$$t_s > n\,(1 - \frac{2}{\sqrt{p}})\,t_w \quad t_w > n\,(1 - \frac{2}{\sqrt{p}})\,t_s .$$

Different results for basic matrix decomposition models (blocks, strips)

- overhead function for blocks $h(s,p)_b$ and for strips $h(s,p)_s$ as follows

$$h(s,p)_b = 8\,(t_s + \frac{n}{\sqrt{p}}\,t_w)$$

$$h(s,p)_s = 4\,(t_s + n\,t_w)$$

- complex parallel execution time for blocks T(s, $p)_{compb}$, and for strips $T(s,p)_{comps}$

$$T(s,p)_{calcb} = T(s,p)_{calc} + T(s,p)_{ipcb} = \frac{n^2.t_{c1}}{p} + 8(t_s + \frac{n}{\sqrt{p}}\,t_w)$$

$$T(s,p)_{calcs} = T(s,p)_{calc} + T(s,p)_{ipcs} = \frac{n^2.t_{c1}}{p} + 4(t_s + n\,t_w)$$

- parallel speed up for blocks $S(s,p)_b$, and for strips $S(s,p)_s$

$$S(s,p)_b = \frac{T(s,1)}{T(s,p)_{calcb}} = \frac{n^2\,p\,t_{c1}}{n^2 t_{c1} + 8\,(p\,t_s + \sqrt{p}\,n\,t_w)}$$

- efficiency for blocks $E(s,p)_b$, and for strips $E(s,p)_s$

$$E(s,p)_b = \frac{S(s,p)_b}{p} = \frac{n^2\,t_{c1}}{n^2 t_{c1} + 8\,(p\,t_s + \sqrt{p}\,n\,t_w)}$$

$$E(s,p)_s = \frac{S(s,p)_s}{p} = \frac{n^2\,t_{c1}}{n^2 t_{c1} + 4\,p\,(t_s + n\,t_w)}$$

- constant C (needed constant in deriving an isoefficiency function w (s)) and that for blocks as C_b, and for strips as C_s in issoeficiency function

$$C_b = \frac{E(s,p)}{1 - E(s,p)} = \frac{n^2 t_{c1}}{8\,(p\,t_s + \sqrt{p}\,n\,t_w)}$$

$$C_s = \frac{E(s,p)}{1 - E(s,p)} = \frac{n^2 t_{c1}}{4\,p\,(t_s + n\,t_w)}$$

Fig. 10 illustrates growth dependencies of parallel computing time $T(s,p)_{comp}$, communication time $T(s,p)_{comm}$ and complex parallel execution time $T(s,p)_{complex}$ from input load growth n (Square matrix dimension) at constant number of computing node p = 256

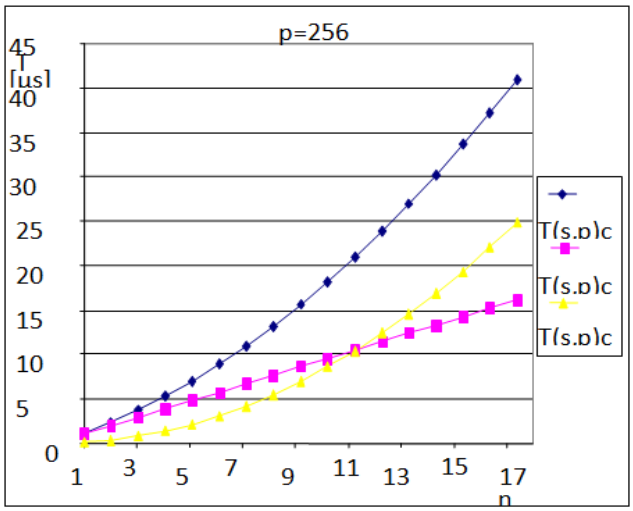


Figure 10. *Dependencies of $T(s,p)_{complex}$,$T(s,p)_{comm}$ and $T(s,p)_{comp}$ from n (p=256).*

Fig. 11 illustrates growth dependencies of parallel computing time $T(s,p)_{comp}$, communication time $T(s,p)_{comm.}$ and complex parallel execution time $T(s,p)_{complex}$ from increasing number of computing node p at constant input load n (Matrix dimension n = 512)

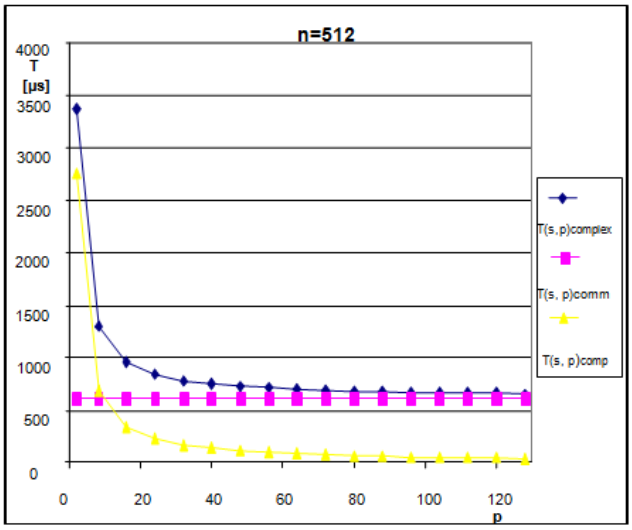


Figure 11. *Dependencies of $T(s, p)_{complex}$, $T(s, p)_{comm}$ and $T(s, p)_{comp}$ from p (n=512).*

5.3.1. Issoeficiency Functions

Issoeficiency function w(s) is very important for performance prediction of parallel algorithms PA. For modeling of performance prediction in PA we are going to derive for defined basic matrix decomposition models (blocks, strips) corresponding analytical issoeficiency functions $w(s)_b$ (Decomposition to blocks) and $w(s)_s$ (decomposition to strips). For asymptotic complexity of w(s) is valid following derived relation as

$$w(s) = \max\left[T(s, p)_{calc},\ h(s, p)\right]$$

where defined workload s is a function of input load n. For IPA it is given as $s = n^2$. We have defined that for given value efficiency E(s, p) following quotient of efficiencies E(s, p) is constant

$$\frac{E}{1-E} = C$$

5.3.1.1. Canonical Matrix Decomposition Models

As basic matrix decomposition models we have defined such matrix decomposition models to which there is possible to reduce all other known matrix decomposition models. Then the canonical matrix decomposition models are as follows

- matrix decomposition model to blocks
- matrix decomposition model to strips.

For defined constants C_b (blocks) and C_s (strips) which are integral parts of issoeficiency functions $w(s)_b$ (blocks) and w(s) (strips) we have derived following relations

$$C_b = \frac{E(s,p)}{1-E(s,p)} = \frac{n^2 t_{c1}}{8\ (p\ t_s + \sqrt{p}\ n\ t_w)}$$

$$C_s = \frac{E(s,p)}{1-E(s,p)} = \frac{n^2 t_{c1}}{4\ p\ (t_s + n\ t_w)}$$

To win a closed form of issoeficiency function $w(s)_b$, $w(s)_s$ we have used an approach in which we performed at first the analysis of increasing input load influenced the analyzed expression contained t_s in relation to p so to keep this growth constant (we supposed that $t_w = 0$). Then for constants C_b, C_s we get following expressions

$$C_b = \frac{n^2 t_{c1}}{8\ p\ t_s} \quad C_s = \frac{n^2 t_{c1}}{4\ p\ t_s}$$

From these expressions we can derive for searched functions $w(s)_b = w(s)_s = n^2$ from relations for C_b, C_s following relations

$$w(s)_b = n^2 = 8\ C_b\ p\ \frac{t_s}{t_{c1}} \quad w(s)_s = n^2 = \frac{4\ C_s\ p\ t_s}{t_{c1}}$$

With a similar approach we can analyze the influence growth of input load caused another part of expression from t_s in relation to p so to keep this growth constant (we supposed that $t_s = 0$). Then after setting and performed needed adjustments we get for searched functions $w(s)_b$ and $w(s)_s$ following relations

$$w(s)_b = n^2 = 8\ C_b\ \sqrt{p}\ n\ \frac{t_w}{t_{c1}} \quad w(s)_s = 4\ C_s\ n\ p\ \frac{t_w}{t_{c1}}$$

Final derived analytical functions $w(s)_b$and $w(s)_s$ are as follows

$$w(s)_b = \max\left[\frac{n^2 t_{c1}}{p},\ 8\ C_b\ p\ \frac{t_s}{t_{c1}},\ 8\ C_b\ n\sqrt{p}\ \frac{t_w}{t_{c1}}\right]$$

$$w(s)_s = \max\left[\frac{n^2 t_{c1}}{p},\ 4\ C_s\ p\ \frac{t_s}{t_{c1}},\ 4\ C_s\ n\ p\ \frac{t_w}{t_{c1}}\right]$$

5.3.1.2. Optimization of Issoefficiency Functions

Optimization of derived issoefficiency functions require to search for dominant expressions in derived final relations for $w(s)_b$ a $w(s)_s$. For this purpose we have been done comparison of individual expressions of $w(s)_b$ and $w(s)_s$ with following conclusions

- the first expressions of $w(s)_b$ and $w(s)_s$ are the same and therefore this expression will be the component of final optimized $w(s)_{opt}$. At the same time for this expression at performed asymptotical analysis in relation to parameter p the following limit is valid

$$\lim_{p\to\infty} \frac{n^2 . t_{c1}}{p} = 0$$

  and therefore the similar first expressions of issoeficiency functions $w(s)_b$ and $w(s)_s$ we can omit from searched $w(s)_{opt}$
- in relation to the similarity of actually first expressions of $w(s)_b$ and $w(s)_s$ (After omitting expressions according previous conclusion) as

follows

$$8\,C_b\;p\,\frac{t_s}{t_{c1}} \geq 4\,C_s\;p\,\frac{t_s}{t_{c1}}$$

This condition after reducing of shared expression parts lead to inequality $2\,C_b \geq C_s$. After setting and following adjustments we get final condition as $p \geq 1$, which is valid on the whole range of spotted values of parameter p. It means the with this performed expression comparison we have got more dominated expression which we let to the next three comparisons (Two from $w(s)_b$ and one from $w(s)_s$)

- in an analogous way we do comparison of third expressions from original $w(s)_b$ and $w(s)_s$ issoeficiency functions and that as follows

$$4\,C_s\;n\;p\,\frac{t_w}{t_{c1}} \geq 8\,C_b n\,\sqrt{p}\,\frac{t_w}{t_{c1}}$$

These conditions after reducing of shared expression parts lead to following inequality $C_s\sqrt{p} \geq 2\,C_b$. After setting and performed adjustments we get final condition $p \geq 1$, which is fulfilled on the whole range of parameter p. With performed comparison we have ignored further less expression and final relation $w(s)_{opt}$ is actually as follows

$$w(s)_{opt} = \max\left[4\,C_s\;n\;p\,\frac{t_w}{t_{c1}}, 8\,C_b\;p\,\frac{t_s}{t_{c1}}\right]$$

- final comparison of remaining expression comes to following expression comparison

$$4\,C_s\;n\;p\,\frac{t_w}{t_{c1}} \geq 8\,C_b\;p\,\frac{t_s}{t_{c1}}$$

This condition after reducing of shared expression parts leads to following inequality $C_s\,n\,t_w \geq 2\,C_b t_s$. After setting and performed adjustments we come to following inequality $n^2.\,t_w{}^2 \geq \sqrt{p}\,t_s{}^2$. This inequality we are able to solve only for concrete values of parameters n, p, t_s, t_w. For example using following values of parameters t_s = 35 µs, t_w = 0,23 µs and under assumption of in praxis frequent case of choosing n = p we get simpler expression to condition validity as $n.\,\sqrt{n} \geq 152{,}17^2$ or $p.\,\sqrt{p} \geq 152{,}17^2$. The smallest integer number which satisfies given condition is p = n = 813. Satisfying this condition for n or p, the final issoefficiency function $w(s)_{opt}$ given with first expression and in opposite is given with second expression of following final optimized issoeficiency function $w(s)_{opt}$

$$w(s)_{opt} = \max\left[4\,C_s\;n\;p\,\frac{t_w}{t_{c1}}, 8\,C_b\;p\,\frac{t_s}{t_{c1}}\right].$$

5.3.1.3. Conclusions of Issoeficiency Functions

Then for the given concrete value of E(s, p) and for given values of parameters p, n we can in analytical way the thresholds, for which growth of isoefficiency function means decreasing of efficiency of given parallel algorithm with assumed typical decomposition strategies. This means the minor scalability of the assumed algorithm. In case of decomposition strategy the approach is similar to analyzed practical used decomposition matrix strategies.

Based on analysis of computer technical parameters t_s, t_w, t_c for some parallel computers in the world they are valid following inequalities $t_s >> t_w > t_c$. Alike is valid that $p \leq n$. Using these inequalities it is necessary to analyze dominancy influence of the all derived expressions.

Then the asymptotic issoefficiency function is limited through dominancy conditions of second and third expressions. From their comparison comes out

- based on real condition $t_w \geq t_s$ a third expression is bigger or equal than a second expression and an issoefficiency function is limited through the first expression of $w(s)_{opt}$. If we used following technical parameters t_s = 35 µs, tw= 0,23 µs this is true for n ≥ 813
- for n < 813 and for the same technical constants t_s = 35 µs, t_w= 0,23 µs issoefficiency function is limited through a second expression of $w(s)_{opt}$.

6. Results

We illustrate some of chosen performed tested results. To practical illustrations we have used MPA algorithms for iterative solving of Laplace PDE equation defined as follows

- four point iteration relation in which in one iteration are performed five arithmertic operations ($t_{c1} = 5\,t_c$)
- communication model according Fig. 5.

For experimental testing we have used workstations of NOW parallel computer (workstations WS 1 – WS 5) and supercomputer as follows

- WS 1 – Pentium IV (f = 2,26 G Hz)
- WS 2 - Pentium IV Xeon (2 proc., f = 2,2 G Hz)
- WS 3 - Intel Core 2 Duo T 7400 (2 cores, f=2,16 GHz)
- WS 4 - Intel Core 2 Quad (4 cores, 2.5 GHz)
- WS 5 - Intel SandyBridge i5 2500S (4 cores, f=2.7 GHz)
- supercomputer Cray T3E in remote computing node.

Comparison of decomposition model influence (D1 - blocks, D2 - strips) is at Fig. 11. For comparison were measured values recomputed to one iteration step. Performed measurements have proved higher efficiency of decomposition model to blocks for tested parallel computer Cray T3E. Technical parameters of parallel computer Cray T3E (t_s = 3 µs, t_w = 0,063 µs, t_c = 0,011 µs).

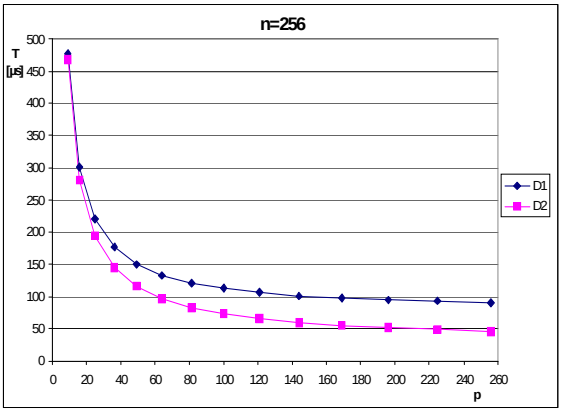


Figure 12. *Comparison of $T(s, p)_{complex}$ for decomposition models (n=256).*

Fig. 13 illustrate dependencies to optimal selection of decomposition strategy for technical parameter t_{si} (t_{s1}, t_{s2}) using verified technical parameters of supercomputer Cray T3E for t_w = 0,063 μs and n = 128, 256.

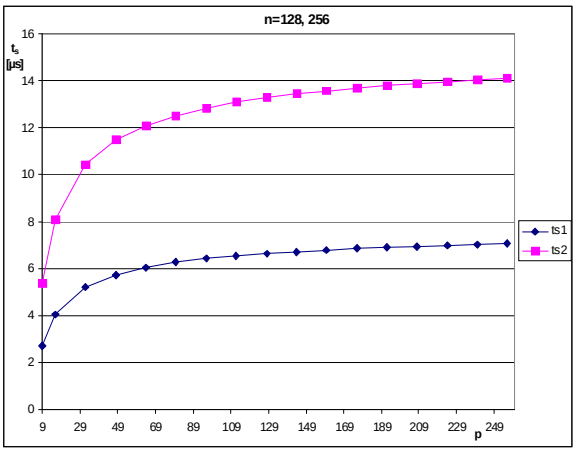


Figure 13. *Influences t_s for n = 128, 256*

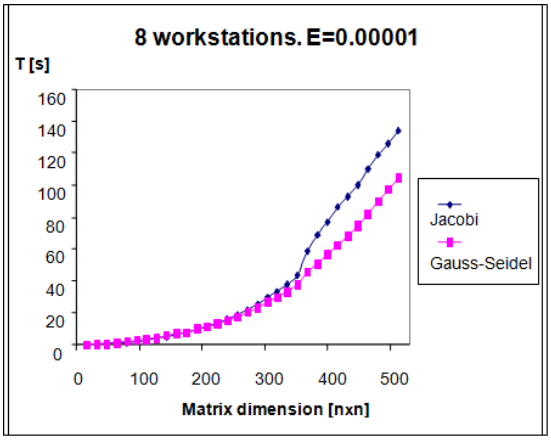


Figure 14. *Comparison of $T(s, p)_{complex}$ for Jacobi and Gauss-Seidel IPA for $E=10^{-5}$.*

At Fig. 14 we have presented measurement results of the whole solving time for both developed parallel algorithms (Jacobi, Gauss-Seidel) with the number of processors p = 8 and for various values of input workload n (Matrix dimensions) for $E=10^{-5}$. From comparisons of these measurements come out that for great number of workstations (p=8) are the whole solving times approximately the same. The reasons are that lower computation complexity at Gauss-Seidel method (Computation) is eliminated through greater communication complexity in its parallel algorithm practically double higher than Jacobi IPA.

This figure illustrates continually percent spreading of the individual overheads (Initialization, computation, communication, gathering) for Jacobi parallel algorithm with the given number of workstations p = 4 for various values of workload n (matrix dimensions) and for accuracy E=0,001. From comparisons we can see raising trend of computation in dependence of accuracy E.

Generally for the problems with increasing communication complexity through using great number of processors p based on Ethernet NOW we come to the point (Threshold that parallel computing is no more effective, that means we are without any speed-up. It is evident that for the given problem, given parallel algorithms and given parallel computer to find such a threshold (no speed-up) is very important.

The individual parts of the whole execution parallel time are illustrated at Fig. 15 for Jacobi iterative parallel algorithm for 4 workstations and for E = 0,001.

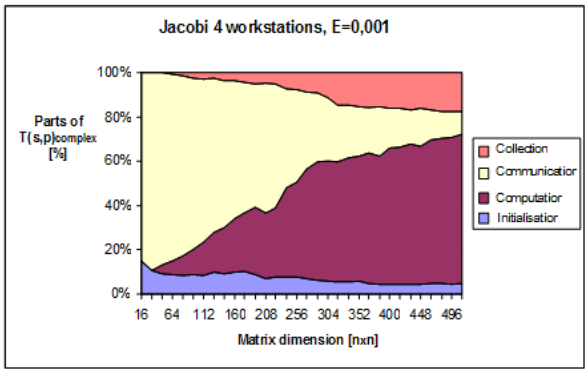
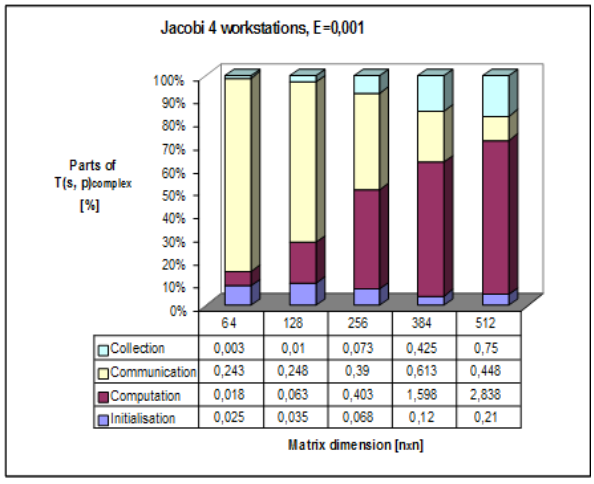


Figure 15. *Percentage comparison of $T(s, p)_{complex}$ for its components (E=0,001).*



	64	128	256	384	512
Collection	0,003	0,01	0,073	0,425	0,75
Communication	0,243	0,248	0,39	0,613	0,448
Computation	0,018	0,063	0,403	1,598	2,838
Initialisation	0,025	0,035	0,068	0,12	0,21

Figure 16. *Comparison of $T(s, p)_{complex}$ parts for p=4 (E=0,001).*

The influence of number of workstations at given accuracy E=0,001 to the individual parts of the whole solving time for pre Jacobi iterative parallel algorithm for various sizes of input workload (matrix dimensions from 64 x 64 to the size 512 x 512) illustrates Fig. 16 for the number of workstations p = 4. From the comparisons come out percent sinking of computations at the bigger number of workstations (parallel speed-up though the higher number of workstations) at the moderate percent raising of network communication overheads.

Fig. 17 illustrates the times of individual parts of the whole solving time as a function of input workload n (Square matrix dimensions) for number of workstations p=4 and for accuracy E=0,001. From comparison of these both graphs comes out higher contribution through the number of working stations than the raising overheads of the network communication.

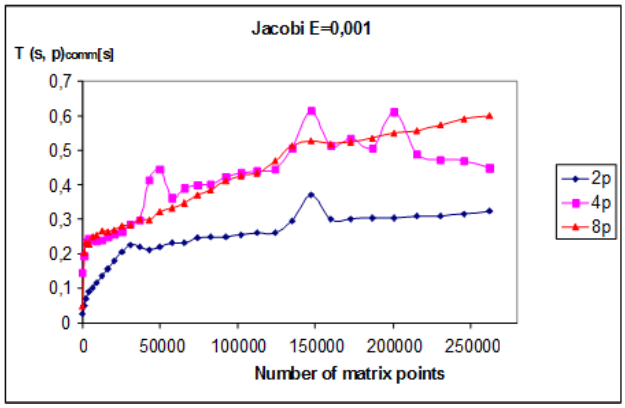


Figure 17. *Influence of computing nodes to $T(s, p)_{comm}$, (E=0,001).*

Fig. 18 illustrate influence of number of workstations NOW to quicker solutions of both distributed parallel algorithms (Gauss-Seidel parallel algorithm) for matrix dimensions 512x512 and various analyzed accuracies of Epsilon.

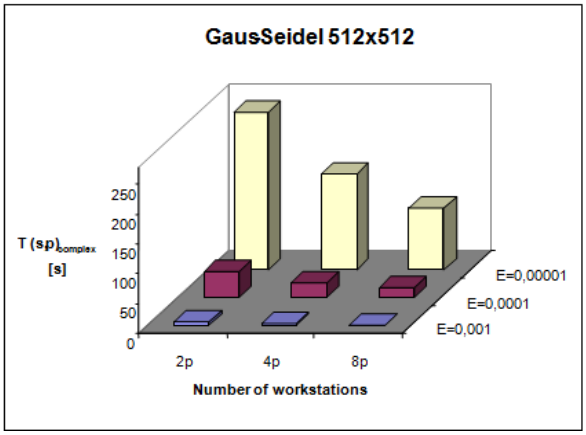


Figure 18. *Influence of workstation number for Gauss-Seidel IPA.*

Derived analytical issoeficiency functions allow us to predict parallel computer performance also for theoretical not existed ones. We have illustrated at Fig. 19 isoefficiency functions for individual constant values of efficiency (E = 0,1 to 0,9) for n < 152 using the published technical parameters t_c, t_s, t_w communication constants of used NOW (t_c= 0,021 µs, t_s = 35 µs, t_w= 0,23 µs,).

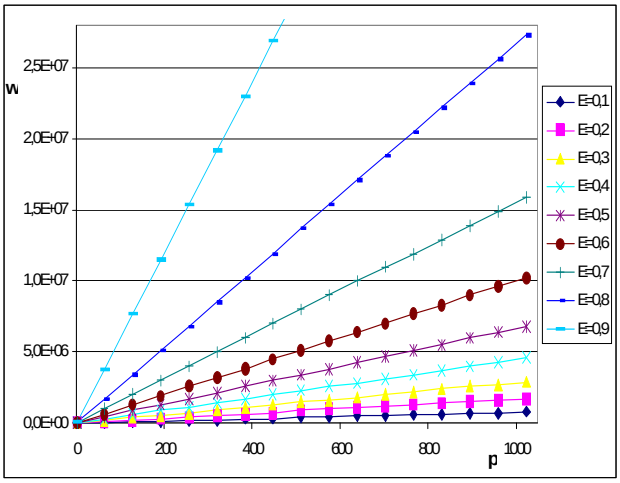


Figure 19. *Isoefficiency functions w(s) for n < 152*

Fig. 20 illustrates isoefficiency functions for individual constant values of efficiency (E = 0,1 to 0,9) for n = 1024 and for communication parameters of parallel computer Cray T3E (t_c= 0,011µs, t_s = 3 µs, t_w= 0,063 µs,).

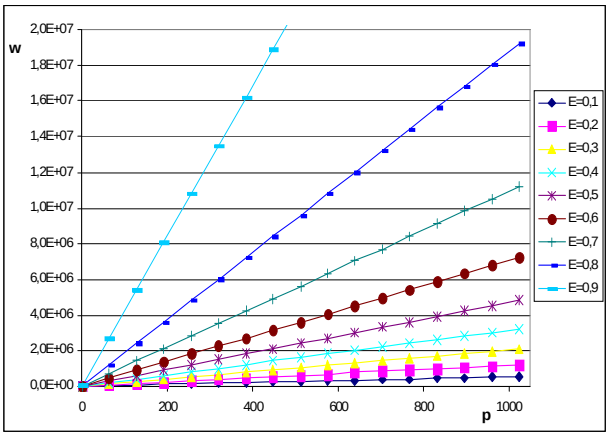


Figure 20. *Isoefficiency functions w(s) (n = 1024).*

From both pictures (Fig. 19 and Fig. 20) we can see that to keep a given value of efficiency we need step by step increasing number of computing processors and higher value of workload (useful computation) to balance higher communication overheads.

7. Conclusions

Performance evaluation as a discipline has repeatedly proved to be critical for design and successful use in parallel computing. At the early stage of design, performance models can be used to project the system scalability and evaluate alternative solutions. At the production stage, performance evaluation methodologies

can be used to detect bottlenecks and subsequently suggests ways to alleviate them. Queuing networks has been established in modeling of parallel computers [5, 17]. Extensions of complexity theory to parallel computing have been successfully used for the evaluation of parallel algorithms and communication complexity too. Via extended form of isoefficiency concept for parallel algorithms we have demonstrated its applied using to performance prediction in typical matrix parallel algorithms (MPA).

To derive isoefficiency function in analytical way it is necessary to derive al typical used criterion for performance evaluation of parallel algorithms including their overhead function (parallel execution time, speed up, efficiency). Based on this knowledge we are able to derive issoefficiency function as real criterion to evaluate and predict performance of parallel algorithms also for future hypothetical parallel computers. So in this way we can say that this process includes complex performance evaluation including performance prediction.

Due to the dominant using of parallel computers based on NOW modules and their high integration named as Grid there has been great interest in performance prediction of parallel algorithms in order to achieve effective parallel algorithms (optimized). Therefore this paper summarizes the used methods for complexity analysis which can be applicable to all types of parallel computers (supercomputer, NOW, Grid).

This paper finalizes applying of complex analytical modeling to the whole group of matrix parallel algorithms which are characterized by domain decomposition models. To present this group of MPA we have modeled abstract matrix using basic decomposition models with supposed intensive communication complexity. In such a way performed complex modeling could be inspiring also to other PA or even a group of PA. The complex analyzed examples we have been evaluated so on classic massive supercomputers (hypercube, mesh) as on dominant parallel computers represented by NOW module.

Acknowledgements

This work was done within the project “Complex performance 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] Arora S., Barak B., Computational complexity - A modern Approach, Cambridge University Press, pp. 573, 2009

[2] Bahi J. H., Contasst-Vivier S., Couturier R., Parallel Iterative algorithms: From Sequential to Grid Computing, CRC Press, USA, 2007

[3] Bronson R., Costa G. B., Saccoman J. T., Linear Algebra - Algorithms, Applications, and Techniques, 3rd Edition, Elsevier Science & Technology, Netherland, pp. 536, 2014

[4] Casanova H., Legrand A., Robert Y., Parallel algorithms, CRC Press, USA, 2008

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

[6] Davis T. A., Direct methods for sparse Linear Systems, Cambridge University Press, United Kingdom, pp. 184, 2006

[7] Desel J., Esperza J., Free Choise Petri Nets, Cambridge University Press, United Kingdom, pp. 256, 2005

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

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

[10] Edmonds J., How to think about algorithms, Cambridge University Press, United Kingdom, pp. 472, 2010

[11] Gelenbe E., Analysis and synthesis of computer systems, Imperial College Press, pp. 324,2010

[12] Goldreich O., P, NP, and NP - Completeness, Cambridge University Press, United Kingdom, pp. 214, 2010

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

[14] Hanuliak P., Hanuliak J., Complex performance modeling of parallel algorithms , American J. of Networks and Communication, Science PG, Vol. 3, USA, 2014

[15] Hanuliak M., Modeling of parallel computers based on network of computing nodes, American J. of Networks and Communication, Science PG, Vol. 3, USA, 2014

[16] Hanuliak M., Hanuliak J., Decomposition models of parallel algorithms, American J. of Networks and Communication, Science PG, Vol. 3, USA, 2014

[17] Hanuliak M., Hanuliak I., To the correction of analytical models for computer based communication systems, Kybernetes, Vol. 35, No. 9, UK, pp. 1492-1504, 2006

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

[19] Hanuliak M., Unified analytical models in parallel and distributed computing, AJNC (Am. J. of Networks and Comm.), SciencePG, Vol. 3, No. 1, USA, pp. 1-12, 2014

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

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

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

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

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

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

[26] Kshemkalyani A. D., Singhal M., Distributed Computing, University of Illinois, Cambridge University Press, UK, pp. 756, 2011

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

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

[29] Levesque John, High Performance Computing: Programming and applications, CRC Press, USA, pp. 244, 2010

[30] Lilja D. J., Measuring Computer Performance, University of Minnesota, Cambridge University Press, United Kingdom, pp. 280, 2005

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

[32] Meerschaert M., Mathematical modeling (4-th edition), Elsevier, pp. 384, 2013

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

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

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

[36] Riano l., McGinity T.M., Quantifying the role of complexity in a system´s performance, Evolving Systems, Springer Verlag, Germany, pp. 189 – 198, 2011

[37] Shapira Y., Solving PDEs in C++ - Numerical Methods in a Unified Object-Oriented Approach (2nd edition), Cambridge University Press, United Kingdom, pp. 800, 2012

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

[39] www.top500.org.