

Scalability analysis of matrix-matrix multiplication on heterogeneous clusters*

Alexey Kalinov

Institute for System Programming of Russian Academy of Sciences
25, Bolshaya Kommunisticheskaya str., Moscow 1090045, Russia
ka@ispras.ru

Abstract

The paper is devoted to scalability analysis of a typical linear algebra algorithm on heterogeneous clusters. We prove that traditional scalability metrics proposed for analysis of linear algebra algorithms is applicable on heterogeneous platform and investigate influence of three heterogeneous strategies of computation distribution to Scalable Universal Matrix Multiplication Algorithm (SUMMA) scalability.

1. Introduction

To run efficiently on a homogeneous parallel platform high-quality parallel applications try to distribute computations between processors evenly. The wide class of application use homogeneous strategy of computation distribution, which can be referred as HoHo “homogeneous distribution of processes over processor – homogeneous distribution of data over processes”. According to HoHo strategy each physical process runs one process and data are evenly distributed between processes. When such applications run on a heterogeneous parallel platform the total time of computation is determined by time elapsed on the weakest processor [1]. Therefore to utilize full performance potential of heterogeneous parallel platform it is necessary to use heterogeneous strategies of computation distribution.

The main goal of heterogeneous strategies [1-8] is to improve efficiency of parallel algorithms. But there is another very important property of parallel algorithms, namely scalability. There is not generally accepted definition of scalability and metrics used for it estimation. In this paper we will follow scalability metrics proposed for linear algebra libraries [9] and used for scalability

*This research is partly supported by the program of the Presidium of the Russian Academy of Sciences “Mathematical modeling and intellectual systems”

analysis of the ScaLAPACK algorithms. We extend it for heterogeneous platform case and apply to investigation how heterogeneous strategies of computation distribution influence to typical linear algebra algorithm. As example was chosen the Scalable Universal Matrix Multiplication Algorithm (SUMMA) [10], which is scalable on homogeneous platform according to metrics proposed in [9]. We consider platforms with different processors performance and homogeneous communication equipment supporting parallel point-to-point communications. To the best of our knowledge such analysis on such platform is performed first time

The rest of the paper is organized as follows. Section 2 is devoted to the extension of scalability metrics to heterogeneous case. In section 3 we outline matrix-matrix multiplication algorithm SUMMA and proof of its scalability. In section 4 we analyze influence of heterogeneous strategies of computation distribution to scalability of SUMMA. Section 5 briefly describes related works.

2. Scalability on heterogeneous platform

In [9] the following scalability metrics is proposed and used for scalability analysis of some ScaLAPACK linear algebra parallel algorithms. *Concurrent efficiency* E , is defined as the concurrent speedup per processor

$$E(N, p) = \frac{T_{seq}(N)}{p \cdot T(N, p)},$$

where $T_{seq}(N)$ - the execution time for the best sequential algorithm running on a processor for solving problem of size N , $T(N, p)$ - the execution time for the parallel algorithm running on p processors. A parallel algorithm is called highly scalable (or simply scalable) if the concurrent efficiency depends on the problem size and the number of processors only through their ratio. For

majority of linear algebra tasks, the problem size is proportional to the matrix size squared.

First we extend the scalability metrics so that to take into account the performance of processors. Let the performance of each processor be characterized by positive real number. Thus we define concurrent efficiency E as follows

$$E(N, p, \mathfrak{R}) = \frac{T_{seq}(N, r_{seq})}{p \cdot T(N, p, \mathfrak{R})}, \quad (1)$$

where $T_{seq}(N, r_{seq})$ - the execution time for the best sequential algorithm running on a processor with performance r_{seq} for solving problem of size N , $T(N, p, \mathfrak{R})$ - the execution time for the parallel algorithm running on p processors performances of which are characterized by vector $\mathfrak{R} = \{r_l\}$, $l \in [0, p)$. For homogeneous parallel system $r_l = r_{seq}$, $l \in [0, p)$.

If $T_{seq}(N, r_{seq})/p$ is the ideal execution time of parallel computation, the concurrent efficiency can be formulated as the ratio of the ideal execution time of parallel computation and the real one

$$E_{he}(N, p, \mathfrak{R}) = \frac{T_{ideal}(N, r_{seq}, p, \mathfrak{R})}{T(N, p, \mathfrak{R})}.$$

That concurrent efficiency can be used for heterogeneous parallel systems as well. The ideal time of computation can be calculated as follows

$$T_{ideal}(N, r_{seq}, p, \mathfrak{R}) = \frac{T_{seq}(N, r_{seq})}{I_r},$$

where I_r characterizes the the increase of performance of the set of p processors compared to the performance of the processor used for sequential computation. I_r is calculated as follows

$$I_r = \frac{\sum_{j=0}^{p-1} r_j}{r_{seq}} = \frac{p \cdot r_{aver}}{r_{seq}},$$

where r_{aver} - the average performance of processors of the heterogeneous parallel system. Hence

$$T_{ideal}(N, r_{seq}, p, \mathfrak{R}) = \frac{r_{seq} T_{seq}(N, r_{seq})}{p \cdot r_{aver}}$$

and we obtain the following formula for *heterogeneous concurrent efficiency*:

$$E_{he}(N, p, \mathfrak{R}) = \frac{r_{seq} \cdot T_{seq}(N, r_{seq})}{p \cdot r_{aver} \cdot T(N, p, \mathfrak{R})} = \frac{r_{seq}}{r_{aver}} \cdot E(N, p, \mathfrak{R}) \quad (2)$$

A parallel algorithm is called scalable on the heterogeneous platform if its heterogeneous concurrent efficiency depends on the problem size and the number of processors only through their ratio. From formula (2) one can see that the heterogeneous concurrent efficiency is obtained by multiplication the usual concurrent efficiency by a factor, which does not depend on the problem size and the number of processors. So, we can use the same metrics for scalability analysis for both heterogeneous and homogeneous platform.

3. Scalable Universal Matrix Multiplication Algorithm (SUMMA)

3.1. Algorithm

Now we outline algorithm SUMMA [10]. Let us consider a problem of forming $C = AB$. Processes of parallel program are considered as logical $P \times Q$ mesh. They are indexed with of row and column numbers. We will denote (i, j) mesh node as P_{ij} . We will assume the following assignment of data to nodes: Given $m^X \times n^X$ matrix X , $X \in \{A, B, C\}$ and a $P \times Q$ logical mesh of nodes, we partition as follows:

$$X = \begin{pmatrix} X_{00} & \dots & X_{0(P-1)} \\ \dots & & \dots \\ X_{(Q-1)0} & \dots & X_{(Q-1)(P-1)} \end{pmatrix}$$

and assign X_{ij} to node P_{ij} . Submatrix X_{ij} has dimensions $m_i^X \times n_j^X$, with $\sum m_i^X = m$ and $\sum n_i^X = n$. For the operation to be well-defined we require $m^A = m$, $n^A = m^B = k$, and $n^B = n$.

If a_{ij} , b_{ij} , and c_{ij} denote (i, j) element of the matrices, respectively, then the elements of C are given by

$$c_{ij} = \sum_{l=0}^{k-1} a_{il} b_{lj}.$$

For homogeneous case data distribution is restricted so that rows of A and C are assigned to the same row of nodes and columns of B and C are assigned to the same column of nodes. Hence $m_i^C = m_i^A$ and $n_j^C = n_j^B$.

Let us consider what computation is required to form C_{ij} :

$$C_{ij} = \left(\begin{array}{c|c|c|c} \tilde{A}_i & & & \\ \hline A_{i0} & A_{i1} & \dots & A_{i(Q-1)} \end{array} \right) \left(\begin{array}{c} \tilde{B}^j \\ \hline B_{0j} \\ B_{1j} \\ \dots \\ B_{(P-1)j} \end{array} \right).$$

Notice that \tilde{A}_i is entirely assigned to node row i , while \tilde{B}^j is entirely assigned to node column j . Letting

$$\tilde{A}_i = \left(\tilde{a}_i^0 \mid \tilde{a}_i^1 \mid \dots \mid \tilde{a}_i^{k-1} \right) \quad \tilde{B}^j = \left(\begin{array}{c} b_0^{jT} \\ \hline b_1^{jT} \\ \dots \\ b_{k-1}^{jT} \end{array} \right)$$

we see that

$$C_{ij} = \sum_{l=0}^{k-1} \tilde{a}_i^l \tilde{b}_l^{jT}.$$

Hence the matrix-matrix multiply can be formulated as sequence of rank-one updates. Pseudo-code for parallelizing each rank-one update is the following:

```

Cij = 0
for l=0, k-1
    broadcast  $\tilde{a}_i^l$  within my row
    broadcast  $\tilde{b}_l^{jT}$  within my column
    Cij = Cij +  $\tilde{a}_i^l \cdot \tilde{b}_l^{jT}$ 
endfor

```

SUMMA is based on implementation of broadcast as passing of a message around the logical ring that forms the row or column. This allows to pipeline computation and communication.

3.2. Scalability analysis

Let one process runs per processor. We assume simple model of time of sending message with length L between two processes $t = a + bL$. Parameters a and b represent the startup and cost per item transfer time, respectively. Let r is performance of the processor, demonstrated on sequential matrix-matrix multiplication. Then following [10] we can estimate time complexity as:

$$(Q-1)\left(a + \frac{m}{P}b\right) + (P-1)\left(a + \frac{n}{Q}b\right) + \quad (3)$$

$$k\left(\frac{mn}{PQr} + a + \frac{m}{P}b + a + \frac{n}{Q}b\right) + \quad (4)$$

$$(Q-2)\left(a + \frac{m}{P}b\right) + (P-2)\left(a + \frac{n}{Q}b\right) + \frac{mn}{PQr} = \quad (5)$$

$$\frac{mn(k+1)}{PQr} + (k+2Q-3)\left(a + \frac{m}{P}b\right) + (k+2P-3)\left(a + \frac{n}{Q}b\right) \quad (6)$$

Contribution (3) equals the time required for both the first column of $\tilde{A}_{(P-1)}$ and the first row of $\tilde{B}^{(Q-1)}$ to reach $P_{(P-1)(Q-1)}$ (filling the pipe); (4) equals the time for performing the local update and passing the messages; (5) equals the time for the final messages (initiated at $P_{(P-1)(Q-1)}$) to reach end of the pipe and the time for the final update at the end of the pipe ($P_{(P-1)(Q-2)}$ or $P_{(P-2)(Q-1)}$).

To establish the scalability of the pipelined approach, we will analyze the case where $m = n = k$, $P = Q = \sqrt{p}$. This changes the complexity in (6) to approximately

$$T(n, p) = \frac{n^3}{pr} + 2(n + 2\sqrt{p} - 3)\left(a + \frac{n}{\sqrt{p}}b\right) \approx \frac{n^3}{pr} + 2n\left(a + \frac{n}{\sqrt{p}}b\right). \quad (7)$$

Note, that in supposition that $n \gg \sqrt{p}$ it is possible to neglect contributions (3) and (5) and analyze contribution (4) only. Then the concurrent efficiency is

$$E(n, p) = \frac{n^3}{pr \left(\frac{n^3}{pr} + 2n \left(a + \frac{n}{\sqrt{p}} b \right) \right)} \approx \frac{1}{1 + O\left(\frac{p}{n^2}\right) + O\left(\frac{\sqrt{p}}{n}\right)}. \quad (8)$$

Formula (8) shows that the algorithm is scalable because concurrent efficiency depends on problem size n^2 and processor number p only through their ratio.

3.3. Blocking

The algorithm can be improved if reformulated it in terms of matrix-matrix multiplication instead of rank-one updates. In that case parallel algorithm is parameterized with block size nb and each column \tilde{a}_i^l becomes a panel of columns and row \tilde{b}_i^j a corresponding panel of rows.

Let n is divide to \sqrt{p} without reminder and sizes of block distributed to every process n/\sqrt{p} are divided to nb without reminder too. In this case \tilde{a}_i^l and \tilde{b}_i^j are matrices with one dimension equals to nb and number of algorithm step is n/nb .

Then time complexity can be estimated as follows

$$T(n, p) = \frac{n}{nb} \left(\frac{n^2 nb}{pr} + 2 \left(a + nb \frac{n}{\sqrt{p}} b \right) \right) = \frac{n^3}{pr} + 2n \left(\frac{a}{nb} + \frac{n}{\sqrt{p}} b \right)$$

and concurrent efficiency becomes

$$E(n, p) = \frac{n^3}{pr \left(\frac{n^3}{pr} + 2n \left(\frac{a}{nb} + \frac{n}{\sqrt{p}} b \right) \right)} \approx \frac{1}{1 + O\left(\frac{p}{n^2}\right) + O\left(\frac{\sqrt{p}}{n}\right)}.$$

That is blocked algorithm is scalable as well.

4. Influence of heterogeneous strategies to scalability of SUMMA

We consider the following heterogeneous strategies of computation distribution:

- HeHo - heterogeneous distribution of processes of parallel program over processors with homogeneous distribution of data over processes [1,2];
- HoHe - homogeneous distribution of processes of parallel program over processors with heterogeneous distribution of data over processes:
 - HoHe1 - columnwise data distribution [3-5];
 - HoHe2 - data distribution providing true grid [6-8].

In this paper we consider unblocked version of the algorithm only. It is possible to proof that blocked version with different heterogeneous strategies has the same scalability as unblocked version.

4.1. HeHo strategy

According to HeHo strategy number of processes g_l involved in matrix multiplication on processor $l, l \in [0, p)$ is proportional to its performance r_l and data are distributed between processes evenly. In that case upper estimation of time of passing the messages is $g_l(\mathbf{a} + \mathbf{Lb})$, because the process may wait other processes running on the processor to perform communication. If *level of network heterogeneity* (ratio of maximal and minimal processors performance) equals to H then $g_l \leq H$ and upper estimation of time of passing the message $H(\mathbf{a} + \mathbf{Lb})$. Let again $m = n = k$ and $P = Q = \sqrt{p}$. Let performance of a process running on a processor l equals to r_l / g_l . According to [1] time of local updates can be estimated as n^3 / pr'_{\min} , where r'_{\min} is performance of the weakest process performing matrix multiplication. In this case time complexity becomes

$$T(n, p, \mathfrak{R}) = \frac{n^3}{pr'_{\min}} + 2Hn \left(a + \frac{n}{\sqrt{p}} b \right)$$

and concurrent efficiency becomes

$$E(n, p, \mathfrak{R}) = \frac{n^3}{pr_{seq} \left(\frac{n^3}{pr'_{\min}} + 2Hn \left(a + \frac{n}{\sqrt{p}} b \right) \right)} \approx \frac{1}{\frac{r_{seq}}{r'_{\min}} + O\left(\frac{p}{n^2}\right) + O\left(\frac{\sqrt{p}}{n}\right)}. \quad (9)$$

The formula (9) shows that on application of HeHo strategy algorithm SUMMA is scalable.

4.2. HoHe1 strategy

According to HoHe1 strategy one process runs per processor and data are distributed between processes according to their performances.

Let $m = n = k$, $P = Q = \sqrt{p}$ then time of local updates equals to n^3 / pr_{aver} .

According to HoHe1 strategy columns of matrices are assigned to the same column of nodes but rows of matrices can be assigned to different rows of nodes. To estimate communication time let us consider an example of HoHe1 distribution of 6x6 matrix over six processes mapped onto 3x2 process grid presented on figure 1. On the figure vectors \tilde{a}_0^2 , \tilde{a}_1^2 and \tilde{a}_2^2 are highlighted. One can see that in the worse case of communication $P_{00} \Rightarrow P_{01}$, $P_{00} \Rightarrow P_{11}$, $P_{00} \Rightarrow P_{21}$, $P_{10} \Rightarrow P_{21}$ and $P_{20} \Rightarrow P_{21}$, time of vector \tilde{a}_2^2 transfer is equal to time of entire column 2 transfer and requires 5 communications.

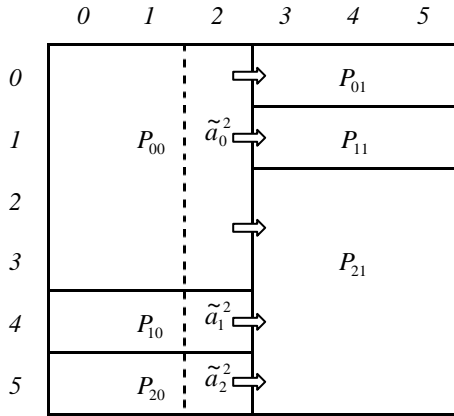


Figure 1. An example of HeHo1 decomposition of matrix 6x6 over six processes mapped into two-dimensional grid 3x2.

In that case for communication overheads estimation we can proceed from the following two obvious statements.

Statement 1. If rows of matrix do not belong to the one row of nodes then every step of cyclic broadcast of a vector \tilde{a}_i^l requires sending of not more than $2P - 1$ messages.

It is necessary to stress that if a row of matrix does not belong to a row of nodes then communication between nodes are losing their independence and in that case may be no difference between communication platform ensuring parallel point-to-point communication and not ensuring.

Statement 2. If rows of matrix do not belong to the one row of nodes then time of every step of cyclic broadcast of a vector \tilde{a}_i^l may be equal to time of all column l transfer.

Then the time of a step of vector \tilde{a}_i^l broadcast is equal to $(2P - 1)a + mb$.

According to [3] sizes of submatrices m_i^X and n_i^X can be bounded with the following value ($m = n = k$, $P = Q = \sqrt{p}$)

$$\frac{\sum_i r_{ij}}{\sum_i \sum_j r_{ij}} n < \frac{\sqrt{p} H r_{\min}}{pr_{aver}} n < \frac{\sqrt{p} H r_{aver}}{pr_{aver}} n = \frac{Hn}{\sqrt{p}}.$$

Time of message sending on the step of algorithm with HoHe1 strategy can be estimated as follows

$$(2P - 1)a + mb + a + \frac{Hn}{Q} b.$$

Then in case $m = n = k$, $P = Q = \sqrt{p}$ time complexity becomes

$$T(n, p, \mathfrak{R}) = \frac{n^3}{pr_{aver}} + 2n\sqrt{p}a + n^2 \left(1 + \frac{H}{\sqrt{p}} \right) b,$$

and concurrent efficiency becomes

$$E(n, p, \mathfrak{R}) = \frac{n^3}{pr_{seq} \left(\frac{n^3}{pr_{aver}} + 2n\sqrt{p}a + n^2 \left(1 + \frac{H}{\sqrt{p}} \right) b \right)} \approx \frac{1}{\frac{r_{seq}}{r_{aver}} + \sqrt{p}O\left(\frac{p}{n^2}\right) + (\sqrt{p} + 1)O\left(\frac{\sqrt{p}}{n}\right)}. \quad (10)$$

Formula (10) shows that with HeHo1 strategy algorithm SUMMA is not scalable. This non-scalability is caused by bad communication pattern.

The HoHe1 strategy is good example demonstrating importance of scalability analysis. On small network with essential heterogeneity of processors performance HoHe1 strategy is more effective then the HoHo strategy. But on large network the HoHo strategy is more effective.

4.3. HoHe2 strategy

The HoHe2 strategy keeps restriction that columns of matrices are assigned to the same column of nodes and rows of matrices are assigned to the same row of nodes. Figure 2 presents an example of HoHe2 distribution of 6x6 matrix over six processes mapped onto 3x2 process grid.

For HoHe2 strategy we can estimate time of local update as for HoHe1 strategy and according to natural data distribution [7] sizes of submatrices m_i^X and n_i^X can be upper bounded with the same value as in case of HoHe1 strategy.

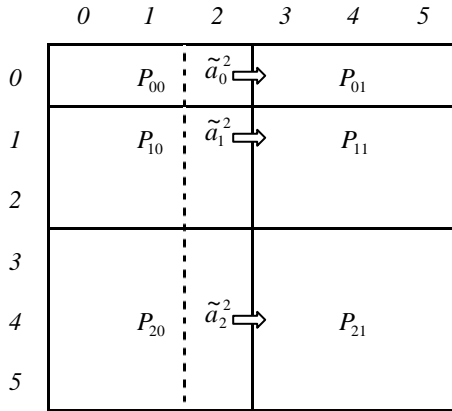


Figure 2. An example of HeHo2 decomposition of matrix 6x6 over six processes mapped into two-dimensional grid 3x2.

Then time complexity becomes

$$T(n, p, \mathfrak{R}) = \frac{n^3}{pr_{aver}} + 2n \left(a + \frac{Hn}{\sqrt{p}} b \right),$$

and concurrent efficiency becomes

$$E(n, p, \mathfrak{R}) = \frac{n^3}{pr_{seq} \left(\frac{2n^3}{pr_{aver}} + 2n \left(a + \frac{Hn}{\sqrt{p}} b \right) \right)} \approx \frac{1}{\frac{r_{seq}}{r_{aver}} + O\left(\frac{p}{n^2}\right) + O\left(\frac{\sqrt{p}}{n}\right)}. \quad (11)$$

That is HoHe2 strategy keeps scalability of SUMMA.

5. Related works

Scalability of algorithms on heterogeneous clusters is not a hot topic of interest. I can point to only a few papers.

X.Zhang and Y.Yan studied scalability of dedicated and no dedicated heterogeneous network [11]. They use definition of scalability as a property, which exhibits performance linearly proportional to the computing power of employed heterogeneous network system. The research deals mainly with influence of hardware heterogeneity and owner load to scalability.

L.Pastor and J.L.Bosque extended the isoefficiency model of scalability proposed in [12] to heterogeneous clusters [13]. The extension is based on definition of concurrent efficiency equals to one used in this paper that is the ratio of the ideal time of parallel computation to the real one. The authors of the paper claims that such definition of concurrent efficiency is new. But it was, for example, used in [6] for estimation of influence heterogeneous data distribution to algorithm efficiency.

In [1] scalability of Cholesky factorization with HeHo and HoHe1 strategies on shared Ethernet was estimated. As scalability metrics was used (1) without proof of its applicability to heterogeneous networks. Was shown that for such network equipment Cholesky factorization is not scalable even with homogeneous data distribution.

6. Conclusion

In the paper was discussed applicability of scalability metrics used for linear algebra algorithms analysis on homogeneous distributed memory systems on heterogeneous platform. It is shown that the scalability

metrics proposed for linear algebra problems can be used for case of heterogeneous cluster as well.

We investigate effect of different strategies of heterogeneous computation distribution to scalability of matrix-matrix multiplication algorithm SUMMA, which is proven to be scalable with homogeneous strategy of computation distribution. We show that HeHo and HoHe2 strategies keep scalability of SUMMA algorithm but HoHe1 does not keep it. Non-scalability of SUMMA with HoHe1 strategy is determined with violation of restriction to allocation of matrix row to one row on process grid.

Note, that in case of HoHe1 strategy there is a conflict between efficiency and scalability. The violation of restriction to allocation of matrix row to one row on process grid provides ideal distribution of computations between processors at the expense of worse communication pattern. This allows to non-scalable HoHe1 strategy be faster than scalable HoHo strategy on relative small network with essential processors heterogeneity. But on large network efficiency of the strategy will be determined with its scalability.

References

- [1] A. Kalinov, and A. Lastovetsky, "Heterogeneous Distribution of Computations While Solving Linear Algebra Problems on Networks of Heterogeneous Computers", *Journal of Parallel and Distributed Computing*, 61, 4, 2001, pp.520-535.
- [2] A. Kalinov, and A. Lastovetsky, mpC + ScaLAPACK = Efficient Solving Linear Algebra Problems on Heterogeneous Networks, in "*Proceedings of the 5th International Euro-Par Conference*", pp.1024-1029, Lecture Notes in Computer Science 1685, Springer, Toulouse, France, August/September 1999.
- [3] A. Kalinov. Heterogeneous two-dimensional block-cyclic data distribution for solving linear algebra problems on heterogeneous networks of computers. *Programming and Computer Software*, Vol. 25, No. 2, 1999, pp. 3-11
- [4] A. Kalinov, and A. Lastovetsky, Heterogeneous Distribution of Computations While Solving Linear Algebra Problems on Networks of Heterogeneous Computers, "*Proceedings of the 7th International Conference on High Performance Computing and Networking Europe (HPCN Europe'99)*", pp.191-200, Lecture Notes in Computer Science 1593, Amsterdam, the Netherlands, April 12-14, 1999.
- [5] Olivier Beaumont, Vincent Boudet, Fabrice Rastello, and Yves Robert, "Matrix Multiplication on Heterogeneous Platforms", *IEEE Trans. On Parallel and Distributed Systems*, 2001, Vol.12, No.10, pp.1033-1051
- [6] Olivier Beaumont, Vincent Boudet, Antoine Petitet, Fabrice Rastello, and Yves Robert. A Proposal for a Heterogeneous Cluster ScaLAPACK (Dense Linear Solvers). *IEEE Trans. Computers*, 2001, Vol.50, No.10, pp.1052-1070
- [7] E. Dovolnov, A. Kalinov, and S. Klimov, Natural Block Data Decomposition for Heterogeneous Clusters, *Proceedings of 13th Heterogenous Computing Workshop (HCW'03)*, IEEE Computer Society, Nice, France, April 2003, pp.
- [8] A. Kalinov and S. Klimov, Multidimensional Static Block Data Decomposition for Heterogeneous Clusters, *to appear in Proceedings of 5th PPAM*, Lecture Notes in Computer Science , Springer, Chenstohova, Poland, September 2003.
- [9] Jack J. Dongarra, Robert A. van de Geun, and David W. Walker. Scalability Issues Affecting the Design of a Dense Linear Algebra Library, *Journal of Parallel and Distributed Computing*, 22, 523-537, 1994
- [10] Robert van de Geijn and Jerrell Watts. SUMMA: Scalable universal matrix multiplication algorithm. *Concurrency: Practice and Experience*, 9(4):255-274, April 1997.
- [11] X.Zhang and Y.Yan, Modeling and characterizing parallel computing performance on heterogeneous networks of workstations, *7th IEEE Symposium on Parallel and Distributed Processing*, October 25-28, 1995, San Antonio, Texas, pp.25-35
- [12] A. Grama, A. Gupta, V. Kumar, Isoefficiency: Measuring the Scalability of Parallel Algorithms and Architectures, *IEEE Parallel & Distributed Technology*, Aug. 1993, pp.12-21
- [13] Luis Pastor and Jose L. Bosque, An efficiency and scalability model for heterogeneous clusters. *Proceedings of Cluster 2001*, 8-11 October 2001, Newport Beach, CA, USA. IEEE Computer Society pp.427-434