

Natural Block Data Decomposition for Heterogeneous Clusters

Egor Dovolnov, Alexey Kalinov and Sergey Klimov
Institute for System Programming of Russian Academy of Sciences
25, Bolshaya Kommunisticheskaya str., Moscow 1090045, Russia
{yegor,ka,sergey}@ispras.ru

Abstract

We propose general purposes natural heuristics for static block and block-cyclic heterogeneous data decomposition over processes of parallel program mapped into multidimensional grid. This heuristics is an extension of the intuitively clear heterogeneous data distribution for one-dimensional case. It is compared to advanced heuristics for heterogeneous data decomposition proposed for solving linear algebra problems on two-dimensional process grid. We experimentally show that for typical local network (12 Windows 2000 PCs interconnected via Fast Ethernet switch) and for typical linear algebra problems these two heuristics have almost the same efficiency. We demonstrate efficiency of the proposed natural decomposition for case of three-dimensional process grid on the example of 3D modeling of supernova explosion also.

1. Introduction

A lot of parallel algorithms are based on homogeneous multidimensional static block or block-cyclic data decomposition over processes mapped into multidimensional grid. Those algorithms provide perfect load balancing for homogeneous parallel systems. But the load balancing that can be achieved using those algorithms on a heterogeneous parallel system is not good enough.

Data decomposition implies performing the following two tasks:

- mapping array of processes into n-dimensional grid. In this paper we consider only the case when the number of processors equals the number of processes and thus we don't distinguish process and processor;
- distributing data over process grid.

For one-dimensional process grid there is intuitively clear heterogeneous data decomposition that doesn't

depend on mapping of processes into grid nodes. So, in this case data decomposition is data distribution. According to this data distribution the amount of data processed by some process is proportional to relative performance of this process.

For a lot of algorithms mapping processes into multidimensional grid is superior to one-dimensional grid. In these cases the situation with mapping processes into multidimensional grid and subsequent distributing data over the process grid is much more difficult. It is proved in [4] that in the case of two-dimensional process grid the optimal solution for the problem is NP-complete. So, for multidimensional process grid we are forced to use heuristic solution in any case.

In the paper we try to answer the question: if there is general heuristics which can be applied for wide range of parallel algorithms based on homogeneous multidimensional static block or block-cyclic data decomposition, or for any relatively small class of algorithms a special data decomposition have to be developed. We discuss obvious extension of one-dimensional data distribution for any number of dimensions of process grid. This heuristics requires minimal modification of the mentioned algorithms for good load balancing for heterogeneous clusters. We show that in spite of its simplicity the proposed heuristics has almost the same efficiency as advanced special heuristics for linear algebra problems and that it is efficient for solving partial differential equations using a hierarchy of nested grids for 3D modeling of supernova explosion.

The rest of the paper is organized as follows. In Section 2 we discuss statement of the problem and introduce natural heterogeneous static block-cyclic data decomposition. In Section 3 we compare the proposed decomposition with the advanced special purposes decompositions for linear algebra problems of Beaumont et al [4]. In Section 4 we examine the application modeling supernova explosion on three-dimensional process grid. We outline application and present results of experiments.

block 10x10 over four processes with relative performances {1, 2, 3, 4}.

2.2.2. Natural heterogeneous data decomposition (multi-dimensional process grid). The idea of the proposed heuristics is quite simple. The m -dimensional natural heterogeneous data distribution is regarded as combination of m natural heterogeneous data distributions $Nat(\hat{s}, \hat{e}, \hat{R})$ each of which distributes data along a dimension of process grid. For j -distribution $\hat{s} = s_j$, $\hat{e} = e_j$, and vector of generalized relative performances \hat{R}_j is computed in the following way:

$$\hat{R}_{j_l} = \sum_{i_0=0}^{e_0-1} \dots \sum_{i_{j-1}=0}^{e_{j-1}-1} \sum_{i_{j+1}=0}^{e_{j+1}-1} \dots \sum_{i_{m-1}=0}^{e_{m-1}-1} R_{i_0, \dots, i_{j-1}, l, i_{j+1}, \dots, i_{m-1}}, 0 \leq l < e_j,$$

here $R_{i_0, \dots, i_{m-1}}$ is relative performance of process with coordinates i_0, \dots, i_{m-1} in process grid. That is l -th element of vector \hat{R}_j is a sum of relative performances of all processes, which have j -th coordinate in process grid equal to l .

| | | |
|---|-----|-----|
| 3 | 0,0 | 1,0 |
| 7 | 0,1 | 1,1 |
| | 4 | 6 |

Figure 2. Natural decomposition of two-dimensional block 10x10 over four processes with relative performances {1, 2, 3, 4} mapped into two-dimensional grid.

As one can see the results of m -dimensional distribution depend on mapping of processes into grid. We propose the simplest heuristics for such mapping. We just propose to arrange the set of processes in ascending order according to process relative performances. If L is a sorted set of processes then the process L_l will be assigned to the node with coordinates i_0, i_1, \dots, i_{m-1} where $I = (\dots(i_{m-1} \cdot e_{m-1} + i_{m-2}) \cdot e_{m-2} + \dots) \cdot e_0 + i_0$ (column wise style) or $I = (\dots(i_0 \cdot e_0 + i_1) \cdot e_1 + \dots) \cdot e_{m-2} + i_{m-1}$ (row wise style). Figure 2 presents natural heterogeneous

decomposition of two-dimensional block 10x10 over four processes with relative performances {1, 2, 3, 4}.

Both parts of the proposed heuristics - mapping processes into grid and computing blocks sizes - are very simple. And to achieve good load balancing with such heuristics on heterogeneous parallel systems, minimal changing in the algorithms, which were initially developed for homogeneous parallel system systems, is required.

2.2.3. Related works. One-dimensional static heterogeneous data distribution similar to natural one is used in different applications and computational kernels [8],[2]. Some specialized heterogeneous distributions for linear algebra problems were proposed in [15],[4].

Crandall and Quinn [9] and Kaddoura et al [13] investigated the general purposes static block data distribution for the case of two-dimensional process grid. The main disadvantage of the proposed distributions is that each process has more then four direct neighbors to communicate with. Over the last few years, a lot of attention was paid to static load balancing for linear algebra kernels. Kalinov and Lastovetsky [11],[12] proposed extension of two-dimensional homogeneous block-cyclic distribution to heterogeneous case that provide perfect load balancing. But this distribution has the same disadvantage as the general distributions of Crandall and Quinn and Kaddoura et al. Barbosa at al [1] used the distribution similar to the one proposed in this paper. Beaumont at al [3],[4] stressed attention on intercoupling of mapping of processes into 2D grid and data distribution. They proved that optimization problem is NP-hard for two-dimensional grids and provided a tuned heuristics intended for optimization of squire matrix multiplication.

To the best of our knowledge, there was only one attempt of Crandall and Quinn [10] to touch upon the three-dimensional process grid case. The paper deals with static distributions aimed at minimization of communication overheads. Several distributions were theoretically studied in the paper and only one of them was really three-dimensional.

3. Two-dimensional process grid (linear algebra problems)

To examine efficiency of the proposed general-purpose natural decomposition we compare it with specialized data decomposition for dense linear algebra kernels of Beaumont at al [3],[4]. First, we describe that decomposition. Then, we compare it with natural one by the simple example. And finally we compare these two decompositions using applications performing matrix multiplication and Cholesky factorization of squire

matrix. For the computations we used an ordinary network consisting of 12 diverse uniprocessor Windows 2000 PCs interconnected via Fast Ethernet switch. MPI Pro 1.6.3 was used as communication platform.

3.1. Specialized decomposition of Beaumont et al.

The objective of this decomposition is derived from the following question: What is the largest number of data elements that can be computed within one time unit? Assume that each processes P_{ij} of the $p \times q$ grid is assigned block of r_i rows and c_j columns of data elements. They need to have $r_i \cdot t_{ij} \cdot c_j \leq 1$ to ensure that P_{ij} can process its block within one cycle. Here t_{ij} is cycle-time – reciprocal of relative performance. Since the total number of data elements being processed is $\left(\sum_{i=1}^p r_i \cdot \sum_{j=1}^q c_j\right)$, they get the (equivalent) optimization problem:

$$\text{Objective: } \max_{r_i \cdot t_{ij} \cdot c_j \leq 1} \left\{ \left(\sum_i r_i \right) \cdot \left(\sum_j c_j \right) \right\}.$$

The rational values r_i and c_j returned by the solution of this optimization problem can be scaled and rounded to get the final solution.

Solution of this optimization problem is NP-Complete. Therefore a heuristic solution is proposed. This heuristic is based on the fact that if processes cycle-times can be arranged into rank-1 matrix, it is easy to compute the r_i and the c_j so that no idle time occurs. In general, it is not possible. The heuristic tries to find both a reasonable arrangement (close to rank-one matrix) and reasonable values of the r_i and the c_j (so that the idle time is low).

The heuristic works as follows: First, it determines a reasonable arrangement matrix for process cycle-times. Then, it computes approximate values for the r_i and the c_j corresponding to this arrangement matrix. Finally, an iterative refinement of the arrangement matrix is proposed which computes a new arrangement matrix which better fits with the values of the r_i and the c_j computed during the second step. Note that initial arrangement matrix is the same as arrangement matrix of natural decomposition and this heuristic cannot be extended for 3D process grid case.

Let us consider 2D heterogeneous decompositions of two-dimensional block 10x10 over four processes with relative performances $\{1, 2, 3, 4\}$, or cycle-times $\{1, 1/2, 1/3, 1/4\}$. According to Beaumont et al optimal decomposition is the following: $\mathbf{r}=\{1,3\}$, $\mathbf{c}=\{1,4/3\}$. After normalization we have $\mathbf{r}=\{2.5,7.5\}$, $\mathbf{c}=\{4.29,5.71\}$. After rounding we have $\mathbf{r}=\{3,7\}$, $\mathbf{c}=\{4,6\}$ which provide

the same solution as natural data decomposition (see 2.2.2).

3.2. Matrix-matrix multiplication

We have implemented SUMMA [16] matrix multiplication algorithm. The implementation of matrix multiplication itself is the same for the both decompositions. Initial mapping of processes into two-dimensional grid is also the same for the both decompositions. For serial matrix multiplication we use the free LAPACK library. The serial matrix multiplication is also used as benchmark for determination of relative performances of the computers. It appears that for a wide range of matrix dimensions relative performance is the same. They are presented in Table 1.

As a factor of comparison we use the ratio of the time of computations with the tuned specialized heterogeneous data decomposition to the time of computation with the natural data decomposition.

We have conducted two series of experiments. In the first series only the most powerful computers take part in the computations. That is computers 1-4 take part in computations for 2x2 grid, computers 1-6 take part in computations for 2x3 and 3x2 grids and so on. Table 2 presents results of the experiments. The grids, for which refinement procedure of specialized data decomposition changes the initial process mapping, are marked with star.

We have conducted two series of experiments. In the first series only the most powerful computers take part in the computations. That is computers 1-4 take part in computations for 2x2 grid, computers 1-6 take part in computations for 2x3 and 3x2 grids and so on. Table 2 presents results of the experiments. The grids, for which refinement procedure of specialized data decomposition changes the initial process mapping, are marked with star.

In the second series, to obtain different permutation of relative performances, we fixed 3x3 grid and changed set of the computers, which take part in computations. Table 3 presents results for the different permutations sorted in ascending order according the total sub network relative performances.

| | | | | | |
|----------------------|-----------------|--------------------|--------------------|--------------------|--------------------|
| Processor | Athlon 1700+ | Pentium III 933 | Pentium III 733 | Pentium III 533 | Pentium III 450 |
| Computer number | 1-3 | 4-8 | 9 | 10-11 | 12 |
| Relative performance | 1000 | 530 | 390 | 330 | 250 |

Table 1. Relative performance of computers demonstrated on serial matrix multiplication

| | Matrix dimensions | | | |
|---------------------------|-------------------|------|------|------|
| | 1000 | 2000 | 3000 | 4000 |
| Grid 2x2 computers 1-4 | 1,01 | 1,04 | 1,00 | 0,98 |
| Grid 2x3 computers 1-6 | 0,99 | 1,00 | 1,01 | 1,01 |
| Grid 3x2 computers 1-6 | 0,98 | 0,99 | 0,99 | 0,87 |
| Grid 3x3 computers 1-9 | 1,02 | 1,02 | 1,01 | 1,05 |
| Grid 3x4 computers 1-12 * | 0,95 | 0,95 | 0,88 | 0,90 |
| Grid 4x3 computers 1-12 | 0,94 | 1,05 | 1,04 | 1,03 |

Table 2. Average ratio of the time of matrix multiplication with tuned specialized heterogeneous data decomposition to the time of matrix multiplication with natural data decomposition. The grids, for which the refinement procedure of specialized data decomposition changes initial process mapping, are marked with star. Block size is 20.

| Grid 3x3 relative performances of commuters | Matrix dimensions | | | |
|--|-------------------|------|------|------|
| | 1000 | 2000 | 3000 | 4000 |
| 3*1000+5*530+390 | 1,02 | 1,02 | 1,02 | 1,05 |
| 3*1000+4*530+390+330 | 0,96 | 0,94 | 0,92 | 1,03 |
| 3*1000+3*530+390+330+250 | 0,99 | 1,06 | 1,07 | 1,03 |
| 3*1000+2*530+390+2*330+250 | 0,95 | 1,01 | 1,03 | 1,02 |
| 2*1000+5*530+390+330 * | 0,95 | 0,92 | 0,90 | 0,85 |
| 2*1000+4*530+390+330+250 * | 0,93 | 0,94 | 0,91 | 0,89 |
| 2*1000+3*530+390+2*330+250 * | 0,97 | 0,98 | 0,96 | 0,93 |
| 1000+5*530+390+330+250 | 0,97 | 0,97 | 1,03 | 1,07 |
| 1000+4*530+390+2*330+250 | 0,95 | 0,98 | 0,96 | 1,03 |

Table 3. Average ratio of the time of matrix multiplication with tuned specialized heterogeneous data decomposition to the time of matrix multiplication with natural data decomposition. The grids, for which the refinement procedure of specialized data decomposition changes initial process mapping, are marked with star. Block size is 20.

| | Matrix dimensions | | | |
|---------------------------|-------------------|------|------|------|
| | 1000 | 2000 | 3000 | 4000 |
| Grid 2x2 computers 1-4 | 0,95 | 1,01 | 1,03 | 1,01 |
| Grid 2x3 computers 1-6 | 0,98 | 0,98 | 0,98 | 0,98 |
| Grid 3x2 computers 1-6 | 0,95 | 1,07 | 1,08 | 1,04 |
| Grid 3x3 computers 1-9 | 1,03 | 1,05 | 1,03 | 1,02 |
| Grid 3x4 computers 1-12 * | 1,01 | 1,00 | 1,00 | 1,00 |
| Grid 4x3 computers 1-12 | 0,99 | 1,01 | 1,03 | 1,04 |

Table 4. Average ratio of the time of Cholesky factorization with tuned specialized heterogeneous data decomposition to the time of Cholesky factorization with natural data decomposition. The grids, for which the refinement procedure of specialized data decomposition changes initial process mapping, are marked with star. Block size is 120.

| Grid 3x3 relative performances of commuturs | Matrix dimensions | | | |
|--|-------------------|------|------|------|
| | 1000 | 2000 | 3000 | 4000 |
| 3*1000+5*530+390 | 1,03 | 1,05 | 1,03 | 1,02 |
| 3*1000+4*530+390+330 | 1,03 | 1,02 | 0,99 | 0,97 |
| 3*1000+3*530+390+330+250 | 1,04 | 1,02 | 1,02 | 1,02 |
| 3*1000+2*530+390+2*330+250 | 1,01 | 1,02 | 1,02 | 1,02 |
| 2*1000+5*530+390+330 * | 0,97 | 0,99 | 1,01 | 0,98 |
| 2*1000+4*530+390+330+250 * | 0,98 | 0,99 | 1,01 | 0,98 |
| 2*1000+3*530+390+2*330+250 * | 0,95 | 1,01 | 1,00 | 0,99 |
| 1000+5*530+390+330+250 | 1,03 | 0,99 | 1,00 | 1,04 |
| 1000+4*530+390+2*330+250 | 0,97 | 1,01 | 1,00 | 0,99 |

Table 5. Average ratio of the time of Cholesky factorization with tuned specialized heterogeneous data decomposition to the time of Cholesky factorization with natural data decomposition. The grids, for which the refinement procedure of specialized data decomposition changes initial process mapping, are marked with star. Block size is 120.

We considered 14 different network configurations. Specialized decomposition change initial mapping of processed into grid for four configurations. It provides better performance (ratio is less than 1) in 33 cases of 56 (that is in 59% of cases). The maximum advantage of specialized decomposition is 15%. The maximum advantage of natural decomposition is 7%. In 4 cases (7%) difference is more than 10% and in 17 cases (30%) difference is more than 5%. So, we can say that special heterogeneous data decomposition intended for matrix multiplication is better than natural one but the advantage is not dominant.

3.3. Cholesky factorization

Parallel matrix multiplication is an example of straightforward algorithm with simple communication pattern. We compare heterogeneous data decompositions on more complex algorithm namely Cholesky factorization as well. The problem was chosen as a well-known example of the practically important problem, whose parallel solution needs careful balancing computations and communications. For the both decompositions, the same algorithm of Cholesky factorization, namely, the one, implemented in ScaLAPACK[5], was used.

We have conducted the same two series of experiments as for matrix multiplication with the same relative performances of the processes. Results are presented in tables 4 and 5. Again, we considered 14 different network configurations. Specialized decomposition provides better performance (ratio is less than 1) in 20 cases of 56 (that is in 36% of cases). The maximum advantage of specialized decomposition is 5%. The maximum advantage of natural decomposition is 8%.

It provides better performance (ratio is greater than 1) in 30 cases of 56 (that is in 54% of cases). So, we can say that for algorithm with more complex communication and computational patterns special heterogeneous data decomposition is not better than natural one.

4. Three-dimensional process grid (partial differential equations)

We have rewritten sequential three-dimensional hydrodynamic application [7] modeling a type-II supernova. The application solves system of partial differential equations using three levels of increasingly finer nested grids [6,17]. All grids have the same number of dimensions and each nested grid is twice finer then the next coarser one. Grids are thickened near point of location of initial distribution of entropy (see Fig. 3). Boundary conditions are set on the largest grid and considered constant in time. Boundary conditions on nested grid are computed with linear interpolation from values from the next coarser grid.

One time step on coarser grid corresponds to two time steps on nested grid. Values on coarser grid points covered with nested grid are computed by averaging of values of nested grid. Flows through points of coarser grid are replaced by flows computed on nested grid. Five-point pattern is used for computation. That is, for computation on every point of grid it is necessary to know values of variables from two neighbors along all six directions. This is one of the main sources of the algorithm communication.

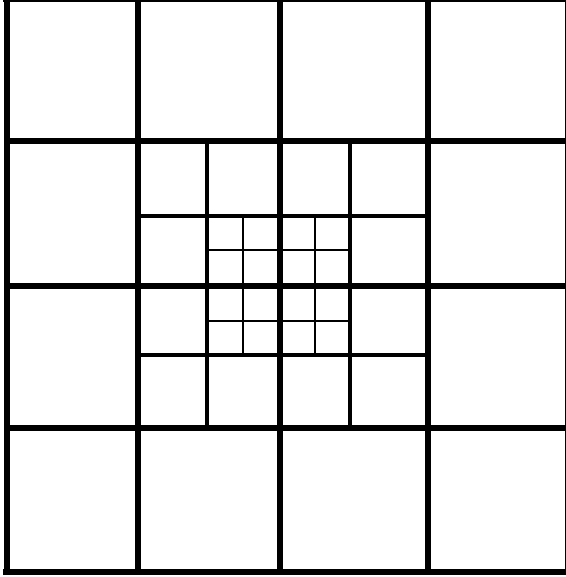


Figure 3. Two-dimensional projection of hierarchy of three nested 4x4 grids thickened near point of location of initial distribution of entropy.

Let us consider how to configure n processes of parallel program to accelerate computations on one grid with homogeneous distribution of data. The total time of computations T is sum of time of sequential computations T_{comp} and time of communications T_{comm} ,

$$T = T_{comp} + T_{comm}.$$

Time of sequential computations is proportional to volume of data distributed to process. In case of cubic data grid it can be expressed by the following formula:

$$T_{comp} = k \cdot \frac{N^3}{n},$$

where N – data grid dimension, k – coefficient of proportionality. It is approximately independent from process grid configuration.

To estimate time of communication we assume that communication between processes are parallel (for example for hardware, like switched Ethernet). In addition, we approximate the time to transfer a message between two processes by a linear function $\alpha + \beta \cdot L$, where α is time to prepare the message for transfer (communication latency) and $\beta \cdot L$ is the time to transfer L doubles. Let n is such that processes can be configured in two and tree dimensional process grids. For the sake of simplicity we assume grids of processes with the same lengths of edges. On every step on computations the volume of data to transfer is proportional to lateral area of block and because of five-point computational pattern to two. In each direction communications are

performed practically simultaneously on all processors. In different directions communications are implemented sequentially (in X-direction, in Y-direction if any, and in Z-direction if any). So time of communications for one, two, and tree dimensional grids T_{comm}^1 , T_{comm}^2 , and T_{comm}^3 we get from the formulas:

$$T_{comm}^1 = a + 2b \cdot N^2,$$

$$T_{comm}^2 = 2a + 2b \cdot N,$$

$$T_{comm}^3 = 3a + 2b \cdot \frac{N}{n^{1/3}}.$$

These formulas imply that with reasonable N the time of communication and correspondingly total time of computations decreases with increase of the number of grid dimensions. So, it is necessary to arrange processes to tree-dimensional grid, if it is possible.

For implementation of this parallel application we use mpC parallel programming language [14]. It is a high level programming language intended for programming for heterogeneous clusters. More information about mpC can be found on <http://www.ispras.ru/~mpc>. As an underlying communication platform for mpC we use MPI implementation MPIPro 1.6.3. Processes are mapped to the three-dimensional grid, which is as close to cube as possible. So, if program consists of 4 processes, then the processes will be mapped into 2x2x1 grid. Similarly, if the program consists of 8 processes they will be mapped into 2x2x2 grid, and so on. The application makes several iterations of algorithm with 3D homogeneous distribution of data. The relative performances of the processes are estimated using the time elapsed on every process. After estimation of relative performances the computing continues with 3D natural heterogeneous decomposition.

| | | | |
|------|-----|-----|-------|
| 1-3 | 4-8 | 9 | 10-12 |
| 1000 | 525 | 415 | 315 |

Table 6. Relative processors performance demonstrated on solving problem with homogeneous data distribution

For experiments we use application modeling the system with three levels of 60x60x60 grids on the network of 12 diverse PCs described in the section 3. Relative performances of computers for homogeneous data distribution are presented in Table 3. To compare heterogeneous and homogeneous data decompositions we have conducted two series of experiments. In the first series only the most powerful computers take part in the computations. That is the computers 1-4 take part in the computations for 2x2x1 grid, the computers 1-8 take part in the computations for 2x2x2 grid and so on. Figure 4

presents the ratio of the time of computations with the homogeneous data distribution to the time of computation with the natural data decomposition.

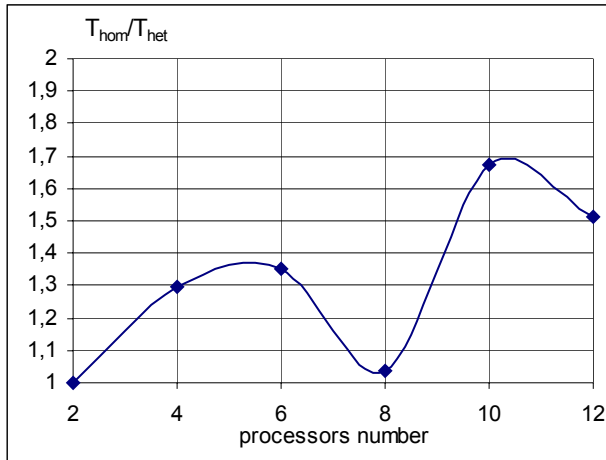


Figure 4. Average speedup achieved by application with natural heterogeneous decomposition relative to homogeneous one. Application models system with three levels of 60x60x60 grids. Increasing of number of processes is done due to addition of weaker computers.

| Grid 2x2x2 relative performances of commuters | T_{ho}/T_{he} |
|---|-----------------|
| 5*525+3*325 | 1,24 |
| 5*525+415+2*325 | 1,28 |
| 1000+4*525+3*325 | 1,36 |
| 2*1000+4*525+2*325 | 1,71 |
| 3*1000+525+415+3*325 | 1,49 |
| 3*1000+2*525+3*325 | 1,60 |
| 2*1000+5*525+1*325 | 1,66 |
| 3*1000+3*525+415+325 | 1,67 |
| 3*1000+4*525+415 | 1,25 |
| 3*1000+5*525 | 1,04 |

Table 7. Average speedup achieved by application with natural heterogeneous decomposition relative to homogeneous one on different sets of eight processors configured as 2x2x2 grid. Application models system with three levels of 60x60x60 grids. Processor sets are sorted in ascending order according to the sums of processor performances.

As a factor characterizing how fully the parallel application utilizes the performance potential of the executing network of computers we use *parallelization*

efficiency E. Parallelization efficiency is defined as $E=S_{real}/S_{ideal}$, where S_{real} is the real speedup achieved by the parallel application on the parallel system, and S_{ideal} is the ideal speedup that could be achieved while parallelizing the problem. The latter is calculated as a sum of performances of processors, consisting the parallel system, divided by the performance of the base processor. Figure 5 presents parallelization efficiency of the application with natural heterogeneous data decomposition. One can see that despite the fact that every time we increase number of computers we add weaker computers parallelization efficiency is still increasing.

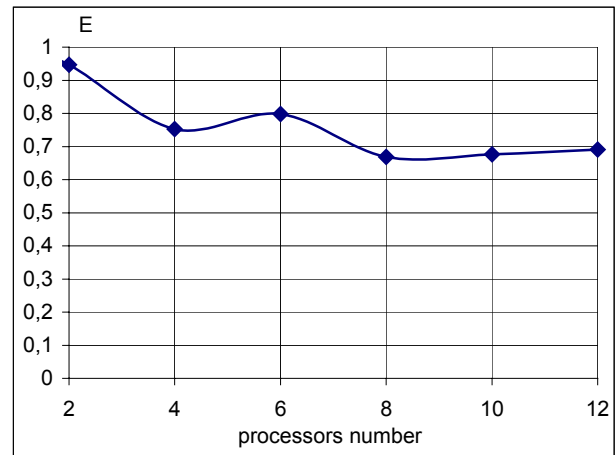


Figure 5. Average parallelization efficiency of the application with natural heterogeneous data decomposition. Application models system with three levels of 60x60x60 grids. Increasing of number of processes is done due to addition of weaker computers.

The presented results show that 3D application with natural 3D heterogeneous data decomposition provides reasonable efficiency. They also show that the speedup achieved owing to the use of heterogeneous data decomposition is worth the relatively small efforts required to implement this data decomposition. Note that adapting the application to heterogeneous decomposition has taken less than 5% of full efforts of algorithm implementation.

5. Conclusion

In this paper we have considered the general-purpose static block data decomposition for parallel computing on heterogeneous networks. This data decomposition provides good load balancing for the cases of both two- and three-dimensional process grids. We have experimentally shown that the considered data

decomposition has almost the same efficiency as the specialized data decomposition for linear algebra problems.

The natural data decomposition is very simple and allows reworking of parallel algorithms intended to homogeneous parallel systems with minimal modification. We show that in spite of simplicity the proposed data decomposition allows writing applications, which run efficiently on multidimensional process grids.

6. References

- [1] J.Barbosa, J.Tavares and A.J.Padilha, "Linear Algebra Algorithms in a Heterogeneous Cluster of Personal Computers", *Proceedings of 9th Heterogeneous Computing Workshop (HCW 2000)*, IEEE Computer Society, Cancun, Mexico, May 1, 2000, pp.147-159
- [2] J.Barbosa, J.Tavares and A.J.Padilha, "A group block distribution strategy for a heterogeneous machine", *Proceedings of the IASTED International Conference NPDPA 2002*, 2002, pp.378-383
- [3] Olivier Beaumont, Vincent Boudet, Fabrice Rastello, and Yves Robert, "Load balancing strategies for dense linear algebra kernels on heterogeneous two-dimensional grids", In *14th International Parallel and Distributed Processing Symposium (IPDPS'2000)*, pages 783-792, 2000
- [4] Olivier Beaumont, Vincent Boudet, Antoine Petit, 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
- [5] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petit, K. Stanley, D. Walker, R. C. Whaley, "ScaLAPACK Users' Guide", SIAM, 1997
- [6] Berger M.J., Philip P. Collela, JCP, v.82, N 1, p. 64-84, 1989
- [7] V. M. Chechetkin, S. D. Ustyugov, A. A. Gorbunov, and V. I. Polezhaev, "On the Neutrino Mechanism of Supernova Explosions", *Astronomy Letters*, Vol.23, No.1, 1997, pp. 30-37
- [8] B.Chetverushkin, N.Churbanova, A.Lastovetsky, and M.Trapeznikova, "Parallel Simulation of Oil Extraction on Heterogeneous Networks of Computers", *Proceedings of the 1998 Conference on Simulation Methods and Applications (CSMA'98)*, the Society for Computer Simulation, November 1-3, 1998, Orlando, Florida, USA, pp.53-59.
- [9] Crandall, P. E., and Quinn, M. J., "Block Data Decomposition for Data-Parallel Programming on a Heterogeneous Workstation Network," *Proceedings of the Second International Symposium on High Performance Distributed Computing* (July 1993), pp. 42-49.
- [10] Crandall, P. E., and Quinn, M. J., "Three-Dimensional Grid Partitioning for Network Parallel Processing," *Proceedings of the ACM 1994 Computer Science Conference* (March 1994), pp. 210-217.
- [11] A.Kalinov "Heterogeneous Two-Dimension Block-Cyclic Data Distribution for Solving Linear Algebra Problems on Heterogeneous Networks", *Programming and Computer Software*, 2, 1999, pp. 3-11
- [12] A.Kalinov, and A.Lastovetsky, "Heterogeneous Distribution of Computations Solving Linear Algebra Problems on Networks of Heterogeneous Computers", *Journal of Parallel and Distributed Computing*, Vol.61, 2001, pp.520-535
- [13] Maher Kaddoura, Sanjay Ranka, and Albert Wang. "Array Decomposition for Nonuniform Computational Environment" *Journal of Parallel and Distributed Computing*, 36(2): 91-105,1996
- [14] A.Lastovetsky, D.Arapov, A.Kalinov, and I.Ledovskih "A Parallel Language and Its Programming System for Heterogeneous Networks", *Concurrency: Practice and Experience*, 12, 2000, pp. 1317--1343
- [15] Sean P. Peisert and Scott B. Baden, "A Programming Model for Automated Decomposition on Heterogeneous Clusters of Multiprocessors", UCSD CSE Technical Report, 2001.
- [16] Robert van de Geijn and Jerrell Watts. SUMMA: Scalable universal matrix multiplication algorithm. *Concurrency: Practice and Experience*, 9(4):255--274, April 1997.
- [17] Ziegler, U., *Comp. Phys. Comm.*, v.109, p.111, 1998

Biographies

Egor Dovolnov is PhD student in the Department of Elasticity (Faculty of Mechanics and Mathematics) at Moscow State University and Research Assistant at Institute for System Programming of Russian Academy of Sciences. He received his diploma in mathematics from Moscow State University in 2001. His research interests include parallel algorithms and heterogeneous computing.

Alexey Kalinov is a senior researcher at the Institute for System Programming of Russian Academy of Sciences. He received his MS in mathematics and engineering from the Moscow Aviation Institute in 1980 and his PhD in engineering from Heavy-Machinery research Institute in 1990. His research interests include

different aspects of parallel computing in heterogeneous environment.

Sergey Klimov is student at Moscow Engineering Physics Institute, chair of Theoretical Physics and Research Assistant at Institute for System Programming of Russian Academy of Sciences. His research interests include modeling of complex physics phenomena.