

Optimal mapping of a parallel application processes onto heterogeneous platform^{*}

Alexey Kalinov and Sergey Klimov

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

Abstract

The paper is devoted to analysis of a strategy of computation distribution on heterogeneous parallel systems. According to this strategy processes of parallel program are distributed over the processors according to their performances and data are distributed between processes evenly. The paper presents an algorithm that computes optimal number of the processes and their distribution over processors minimizing the execution time of an application. The processor performance is considered as a function of the number of processes running on the processor and the amount of the data processing by the processor.

1. Introduction

To run efficiently on a homogeneous parallel platform high-quality parallel applications try to distribute computations between the processors evenly. To achieve this wide class of application use homogeneous strategy of computation distribution, which can be referred as HoHo “homogeneous distribution of processes over processors with homogeneous distribution of data over processes”. According to HoHo strategy each physical processor runs one process and data are evenly distributed between processes. When application using HoHo strategy runs on a heterogeneous parallel platform the total time of computation is determined by time elapsed on the weakest processor because the more powerful processors wait it in communication or synchronization points.

To utilize full performance potential of heterogeneous parallel platform it is necessary to distribute computations between processors according to their performance. Two

main strategies of such computation distribution can be considered [2]:

- HeHo - heterogeneous distribution of processes of parallel program over processors with homogeneous distribution of data over processes;
- HoHe – homogeneous distribution of processes of parallel program over processors with heterogeneous distribution of data over processes.

The main attention of researchers was paid to the HoHe strategy, several different variants were proposed. They demonstrate good results, but implementation of those variants requires redesign of applications and for some variants resulting applications become much more complex than initial one.

In contrast, this HeHo strategy often does not require any changes of parallel application implemented according to HoHo strategy. In many cases, it is enough to just modify configuration file to obtain reasonable speedup compatible with that can be achieved using HoHe strategy.

The main contribution of the paper is investigation of HeHo strategy in case when the performance of the processor is given as function of the amount of data distributed to the processor and the number of processes running on it. The function implicitly depends of communications between processes and thus takes us out from divisible load paradigm [1]. To the best of our knowledge such analysis is performed for the first time.

The rest of the paper is organized as follows. Section 2 is devoted to the formal statement of the problem. In section 3 we consider a solution for the problem. In section 4 some experimental results for real-life application for 3D modeling of supernova explosion are presented. Section 5 briefly describes related works.

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

2. Problem of optimal mapping of a parallel application processes onto heterogeneous platform

We assume that there are a monotonic functional dependence between amount of data and computational load. Therefore, we choose an amount of data (all variables of problem) as a parameter of the application and denote this parameter as W . We divide the application into set of equal tasks, i.e. the tasks with equal computational loads and, hence, equal amounts of data. Each task is a process in our consideration. Very often, the processes are logically considered as a multidimensional grid. Looking for an optimal number of processes and its distribution over processors of the heterogeneous system we will relax from this and consider the set of processes as a one-dimensional grid. The optimal set of processes is configured as multidimensional process grid if possible.

Let us consider the heterogeneous system consisting of p processors. It is not easy to predict in advance how given application will be running on each particular processor of the heterogeneous system. Therefore, we divide all processors of the system into homogeneous groups and run some iteration of the application with different combinations of parameters on one processor of each group to obtain the functions of processor performance. We changed the number of processes (x) and the amount of data (y) and obtained, in general case, p functions $f_i(x, y)$, where $0 \leq i < p$. Example of the obtained surface for a complex real-life application is presented in figure 1. One can see the performance of processor decrease with increasing of the number of processes and the larger amount of data get, the faster this decreasing become.

Now we know all we need about the system and the application to find an optimal total number of processes, denoted by l , and optimal numbers of processes running on each processor of the system, denoted by l_i , $0 \leq i < p$ ($l = \sum_0^{p-1} l_i$). However, at first we will set a restriction on values of l_i .

Let us find the execution time of the i -th processor. As each process processes the same amount of data, we can get the amount of data processing by the i -th processor from formula

$$W_i = \frac{l_i}{l} W \quad (1)$$

Performance of the i -th processor is represented by function $f_i(l_i, W_i)$ hence we can find the time of computations on the processor as:

$$t_i(l_i, l, W) = \frac{W_i}{f_i(l_i, W_i)} = \frac{W}{l} \frac{l_i}{f_i(l_i, W_i)}.$$

Certainly, saying about the HeHo strategy we hope to decrease the execution time of application on heterogeneous platform. Therefore, it is natural to restrict the execution time of the application under the HeHo strategy by the execution time of the same application under the HoHo strategy on the same heterogeneous platform. This condition gives us restriction on value of l_i , because W is fixed. Thus, our restriction looks like

$$t_i \leq t_{HoHo}.$$

When we run an application under the HoHo strategy on a heterogeneous platform, the weakest processor of the system defines an execution time. Therefore, we can find this execution time by following formula:

$$t_{HoHo} = \frac{W}{p} \max_{0 \leq i < p} \frac{1}{f_i(l, \frac{W}{p})}.$$

Now, let us try to formulate an objective for our problem. The main qualitative characteristic of our algorithm will be an execution time of the application. Therefore, when we build our algorithm we have to minimize this time. Unfortunately, it is highly unlikely that t_i will be the same on all processors of the system because of coarse granularity in the HeHo strategy, therefore we consider an execution time as a maximum of t_i . Thus, for fixed total number of the processes l we will find such numbers of processes l_i running on each processor of the system that satisfy the following condition:

$$\min_{\{l_i\}: t_i \leq t_{HoHo}, \sum_i l_i = l} \left(\max_{0 \leq i < p} t_i \right). \quad (2)$$

The total number of the processes can take on any value if amount of data corresponding to one process is greater than unit of data in the application and if t_i less than t_{HoHo} for any i . Obviously, that the second condition will be satisfied earlier than the first one if we increase l , therefore we do not need any additional condition and can formulate final objective for our algorithm:

$$\min_{l: \exists \{l_i\} \forall l_i: t_i \leq t_{HoHo}, \sum_i l_i = l} \left(\max_{0 \leq i < p} t_i \right). \quad (3)$$

In the next section, we will try to build an algorithm for solving the problem (3).

3. A solution of the problem

In the algorithm, we will use the following assumption:

$$t_i(l_1, l, W) < t_i(l_2, l, W) \Leftrightarrow l_1 < l_2. \quad (4)$$

As we said above the processor performance decreases if the number of processes increases for fixed amount of data. In this assumption, the both the number of processes and the amount of data processing by the i -th processor are increased, and t_i is proportional to the number of processes. Thus, assumption (4) is almost obvious in our consideration.

The main idea of our algorithm is very simple. Let us assume there are l processes of the application and part of them we have already disseminated over p processors of the heterogeneous system. We have to find the processor for each other process from $l - \sum l_i$ residuary processes. As we know that t_i will be only increase with increasing of l_i (4), we can simple compute t_i for the number of processes l_i increasing by 1. It is clear that the best place for the next process will be processor for which computed t_i is minimal because in this case decreasing of the execution time is minimal. At the beginning of our algorithm, we fix value of the total number of processes l and set all l_i to zero, then we find the best place for each next process until all l processes are disseminated. This is a solution of the problem (2). For solving the problem (3), we look through all possible values of the total number of processes while we can find the set of l_i that is solution of problem (2). Computing the value of the objective (2) for each considering value of l , we compare these values and find the minimum one. Thus, we obtain a solution.

Formal notation of the proposed algorithm is following:

```

 $l^{opt} = 0, \forall i \Rightarrow l_i^{opt} = 0, l = 0$ 
while (true) do
     $l = l + 1$ 
     $l_i = 0, 0 \leq i < p$ 
for  $m=1$  to  $l$ 
if  $\exists k \in [0, p) : t_k(l_k + 1, l, W) \leq t_{HoHo}$  then
find  $k \in [0, p) \& t_k(l_k + 1, l, W) \leq t_{HoHo}$  such that
         $t_k(l_k + 1, l, W) = \min\{t_i(l_i + 1, l, W)\}$ 
         $l_k = l_k + 1$ 
else
    exit from while
end for

```

```

if  $\max\{t_i(l_i, l, W)\} < \max\{t_i(l_i^{opt}, l^{opt}, W)\}$  then
     $l^{opt} = l, \forall i \Rightarrow l_i^{opt} = l_i$ 

```

end while

3.1. Proof of the optimality

Let's set of $\{l_i\}$ is solution of the algorithm for fixed l ($\sum l_i = l$).

Suppose that this solution is not optimal. That is

$$\exists \{l'_i\}: \max_i t_i(l'_i) < \max_i t_i(l_i). \quad (5)$$

Let the maximums are achieved on k -th processor for the algorithm found set, that is $\max_i t_i(l_i) = t_k(l_k)$ and on k' -th processor for the alternative set, that is $\max_i t_i(l'_i) = t_{k'}(l'_{k'})$.

According to assumptions (4)-(5) we can write

$$t_k(l'_k) \leq t_{k'}(l'_{k'}) < t_k(l_k) \Rightarrow l'_k < l_k.$$

Since $\sum l_i = \sum l'_i = l$ we have

$$\exists k'': l'_{k''} > l_{k''} \Rightarrow t_{k''}(l'_{k''}) \geq t_{k''}(l_{k''} + 1).$$

According to condition under which we increase l_k in the algorithm ($t_k(l_k + 1, l, W) = \min\{t_i(l_i + 1, l, W)\}$) we have that $t_{k''}(l_{k''} + 1) \geq t_k(l_k)$. So we obtain $t_{k''}(l'_{k''}) \geq t_k(l_k)$. On the other hand, since $t_{k'}(l'_{k'}) = \max_i t_i(l'_i)$ we have $t_{k''}(l'_{k''}) \leq t_{k'}(l'_{k'})$. Thus, we obtain that $t_{k'}(l'_{k'}) \geq t_k(l_k)$ that is contrary to assumption (5). So we obtain contradiction, hence assumption (5) does not correct.

We cannot proof the algorithm optimality for the problem (3). Certainly, it is possible that there are large values of l , which gives us a valid set of l_i . However, in this case, while the amount of data processing by the i -th processor of the system stays approximately the same, the value of l_i becomes very large. This leads to large number of communications within the i -th processor and it's performance decreases. Therefore, it is highly unlikely that this large value of l gives us an optimal set of l_i and solution of the problem (3). And even if it occurs we can notice that the deviation of execution time for solution of problem (2) became less with increasing of the total number of processes l . Thus, the probability of error in our algorithm is very low and the value of this error is not large.

4. Experimental results

We tested our algorithm on problem of 3D modeling of supernova explosion [3]. It is a real-life mpC [5]

application solving the system of partial differential equations. We ran this application on a small network of diverse uniprocessor PCs. All PCs are running MS Windows 2000 and interconnected with the Fast Ethernet switch. MPI Pro 1.6.5 was used as the underlying

communication platform. The table 1 present the relative performances of the PCs demonstrated on the modeling of supernova explosion. These results are obtained by running the application in sequential mode with data grid 60x60x60

Figure 1. Performance function of the application for 3D modeling of supernova explosion [3] as normalized function of the number of processes per processor and the dimension of cubic data grid. Function is normalized as $f(1,24^3) = 1$

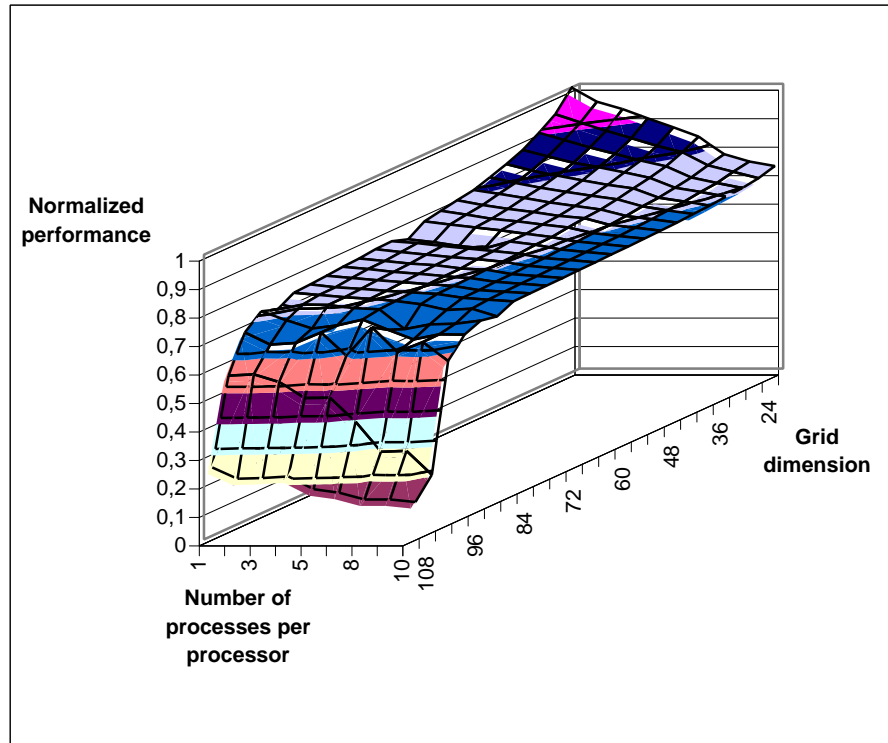


Table 1. The relative performances of the PCs demonstrated on the supernova explosion modeling with data grid 60x60x60

1	2-3	4	5	6
Athlon 1700+ 512 Kb	Pentium III 933 MHz 256 Kb	Pentium III 933 MHz 256 Kb	Pentium III 533 MHz 128 Kb	Pentium III 533 MHz 128 Kb
1	0,54	0,5	0,32	0,3

Table 2. The execution time of the application of supernova explosion implemented under different strategies of load distribution for different one-dimensional process grid

Number of processes per processor for the HeHo strategy						Time for different strategies, sec		
1	2	3	4	5	6	HoHo	HoHe	HeHo
2	1	1	-	-	-	95	73	80,7
2	1	1	1	-	-	70,2	62	61,2
4	2	2	2	1	-	96,3	60,5	62,8
4	2	2	2	1	1	84,1	54	58,6
3	-	-	-	-	1	192,6	107,8	106,7
3	-	-	-	1	1	151	96,7	95,3

The surfaces of the performance function for all processors are similar to one presented in figure 1, which correspond to the processor 1.

The table 2 presents the results of implementation of all three strategies. In addition, we can see distribution of the processes obtained from our algorithm for the HeHo strategy in this table. For the other strategies, distribution of the processes is one-by-one. These values of time were obtained by timing of the five steps of the application with the data grid 60x60x60. In implementation of the HoHe strategy, we use the multidimensional static block data decomposition for heterogeneous clusters [4]. The processes are configured as a one-dimensional process grid.

One can see that the HeHo strategy provides the reasonable speedup. Often it is a bit lower than one provided by the HoHe strategy but sometimes it is even a bit higher.

The application models a 3D phenomenon. For such kind of application, it is useful to configure a set of the processes as a 3D grid because it usually minimizes time of communications (it depends on communication platform). Earlier we relax from this. But it is interesting how topology of the process grid influences on the execution time for selected numbers of processes. For the network 1-6, the total number of processes is equaled 12. The processes can be configured as a 1D process grid 12x1x1, as a 2D process grid 4x3x1 or as a 3D process grid 3x2x2. Table 3 presents the execution time of five steps of the application for the data grid 60x60x60.

The results presented in the table show that the execution time depends on the topology of the process grid. More correct is to consider this topology, when we formulate the objective (2), but this is a more complex problem. It is out of scope of the paper but it is in focus of our close research.

One more interesting question is how to map the processes onto the process grid. Intuitively it is natural to say that all processes running on a processor have to be close to each other in the process grid and arranged in such way that a number of external links would be minimal but a number of internal links would be maximal.

We have not tested this assertion for the application of supernova explosion yet. However, the results obtained from the application of matrix multiplication show that the arrangement of processes in the multidimensional process grid does not influence on the execution time. Free parameter of these experiments was a sum of the numbers of internal links for all processors of the system. Table 4 presents the execution time of matrix multiplication for the one-dimensional process grid 1x10. Table 5 presents the results for the two-dimensional process grid 3x4. In the both cases, we considered matrix 1000x1000.

Table 3. Time of the five steps of application algorithm for different process grid topologies

Process grid topologies	Time, sec
12x1x1	58,6
4x3x1	53
3x2x2	50,3

Table 4. Execution time of matrix multiplication for different number of internal links in case of the one-dimensional process grid 1x10

Number of internal links	Time, sec
0	12.11867
1	11.75328
2	11.23894
3	10.7648
4	10.33172
5	9.628072
6	8.939332
7	8.353306

Note, that in case of the HeHo strategy taking heterogeneity of the links into account by the arrangement of processes in the process grid request more thorough analysis, but we think it is necessary to take heterogeneity of the links between processes into account by correction of the performance functions. The weaker links connect

this process to network, the less performance of the process should be.

Table 5. Execution time of matrix multiplication for different number of internal links in case of the two-dimensional process grid 3x4

Number of internal links	Time, sec
2	8.901026
3	8.859049
4	8.915698
5	8.881971
6	8.881678
7	8.925185
8	8.891172
9	8.901823
10	8.824514
11	8.842702
12	9.106328
13	8.712188
15	9.280016

5. Related work

Task of optimal mapping of fixed numbers of processes with different weights is solved in mpC programming system [5]. It was used for acceleration of different applications, for example of the ScaLAPACK solvers [6]. The system uses a single positive number to represent the speed of processor.

HoHe strategy with more realistic performance model was considered by Lastovetsky and Reddy [7]. They consider the processor performance as a function of data distributed to the processor.

Several issues of HeHo strategies were investigated in the papers presented at HCW'04 [8], [9] and HeteroPar'04 [10].

6. Conclusion

In this paper, we considered a class of application that cannot be divided into divisible tasks and proposed implementation of the HeHo strategy for such applications. We described the pair of heterogeneous platform – the application by set of the performance functions of a number of processes per processor and a amount of data processing by processor. In this paper, we propose a simple algorithm for solving of complex scientific problems on the heterogeneous platforms. It can be useful because when scientists work at scientific problem they are anxious for the scientific results. Very often, it is a big problem for them to implement even the

simplest HoHo strategy. And they do not ready to use very difficult algorithms for the heterogeneous platforms.

Certainly, in this paper, we waste long time for obtaining surfaces of the performance functions (about 8 hours), but it can be noticed that these functions can be approximated by function

$$f(x, y) = \frac{c}{\frac{x^2}{a^2} + \frac{y^2}{b^2}}$$

or something of this kind. It is most probably that this approximation will be work very good in working space of surface. If it is true, it allows us to decrease tuning time very much.

In addition, the questions of the process grid topology and indemnification for heterogeneity of the links between processes are not considered when we build our algorithm, thus, the algorithm require modification.

However, the algorithm demonstrates very good results for real-life application. Moreover, our tests show that implementation of the HeHo strategy may occasionally give a better speedup than more usual HoHe strategy.

7. References

1. Thomas G. Robertrazzi, "Ten Reasons to Use Divisible Load Theory", *Computer*, May 2003, pp. 63-68
2. 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
3. A. Ya. Kalinov, S. A. Klimov, M. A. Posypkin, G. I. Savin, S.D. Ustyugov, V. M. Chechetkin, and B. M. Shabanov, "Mathematical Modeling a Supernova Explosion on a Parallel Computer", *Computational Mathematics and Mathematical Physics*, 44, 5, 2004, pp. 903-909
4. A. Kalinov and S. Klimov, Multidimensional Static Block Data Decomposition for Heterogeneous Clusters, in *Proceedings of 5th PPAM*, Chenstohova, Poland, September 2003, LNCS 3019 , Springer, pp.907-914
5. A.Lastovetsky, D.Arapov, A.Kalinov, and I.Ledovskih, "A Parallel Language and Its Programming System for Heterogeneous Networks", *Concurrency: Practice and Experience*, 12(13), 2000, pp.1317-1343.
6. A.Kalinov, and A.Lastovetsky, "mpC + ScaLAPACK = Efficient Solving Linear Algebra Problems on Heterogeneous Networks", *Proceedings of the 5th International Euro-Par Conference*, Lecture Notes in Computer Science 1685, Toulouse, France, August 31 - September 3, 1999, pp.1024-1031

7. A.Lastovetsky and R. Reddy, "Data Partitioning with a Realistic Performance Model of Networks of Heterogeneous Computers", *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, 26-30 April 2004, Santa Fe, New Mexico, USA, CD-ROM/Abstracts Proceedings, IEEE Computer Society 2004.
8. Y. Kishimoto and S. Ichikawa, "An Execution-Time Estimation Model for Heterogeneous Clusters", *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, 26-30 April 2004, Santa Fe, New Mexico, USA, CD-ROM/Abstracts Proceedings, IEEE Computer Society 2004.
9. Y. Ohtaki, D. Takahashi, T. Boku and M. Sato, "Parallel Implementation of Strassen's Matrix Multiplication Algorithm for Heterogeneous Clusters", *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, 26-30 April 2004, Santa Fe, New Mexico, USA, CD-ROM/Abstracts Proceedings, IEEE Computer Society 2004.
10. J. Cuenca, D. Gimenez, and J. Martinez, "Heuristics for Work Distribution of a Homogeneous Parallel Dynamic Programming Scheme on Heterogeneous Systems", *Proceedings of the 3rd International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (HeteroPar'04)*, July 5-8, 2004 Cork, Ireland, IEEE CS Press.

Biographies

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 graduated from Moscow Engineering Physics Institute, chair of Nuclear Theoretical Physics, in 2004. Now he is a Ph.D. student at Institute for System Programming of Russian Academy of Sciences. His research interests include computational modeling of complex physics phenomena on the heterogeneous parallel platform.