Heterogeneous Distribution of Computations While Solving Linear Algebra Problems on Networks of Heterogeneous Computers

Alexey Kalinov and Alexey Lastovetsky

Institute for System Programming, Russian Academy of Sciences
25, Bolshaya Kommunisticheskaya str., Moscow 109004, Russia
{ka,lastov}@ispras.ru

Abstract. The paper presents a heterogeneous distribution of computations while solving dense linear algebra problems on heterogeneous networks of computers. The distribution is based on heterogeneous block cyclic distribution which is extension of the traditional homogeneous block cyclic distribution taking into account differences in the processor performances. The mpC language, specially designed for parallel programming heterogeneous networks is briefly introduced. An mpC aplication carring out Cholesky factorization on a heterogeneous network of workstations is used to demonstrate that the heterogeneous distribution have an essential advantage over the traditional homogeneous distribution.

1 Introduction

Progress in network technologies is making local and even global networks of computers (in particular, networks of PCs and workstations) more and more attractive for high-performance parallel computing. While developing applications for such networks it is necessary to take into account their heterogeneity being the main peculiarity of common networks differing them from supercomputers.

The heterogeneity is displayed at least in two forms. Firstly, in the form of heterogeneity of machine arithmetics of such parallel systems. Related challenges existing in writing reliable numerical library software for heterogeneous computing environments have been analyzed in [1].

Secondly, in the form of heterogeneity of both performances of individual processors and speeds of data transfer between the processors. As a rule, to solve linear algebra problems on a heterogeneous network of computers one uses numeric software originally developed for homogeneous distributed-memory machines and later on ported to the network. As a rule, while computing on such homogeneous computer systems, a strategy of homogeneous distribution of computations over processors is used. The strategy will be referred as the HoHo strategy - "Homogeneous distribution of processes over processors - Homogeneous distribution of data over the processes", with each physical processor running one process and data being evenly partitioned among the processes.

Let us see what happens when an application, that provides a good distribution of computations and communications (due to the HoHo strategy) while running in homogeneous environments, runs on the heterogeneous network of computers. Since volumes of computations executed by different processors are approximately equal to each other, more powerful processors will wait for the weakest one at synchronization points. Therefore, the total time of computations will be determined by the time elapsed on the weakest processor. Similarly, the total time of communications will be determined by communications via the slowest communication link. So, in general, the total running time on the heterogeneous network will be close to the total running time on a homogeneous network obtained from this heterogeneous one by means of replacement of both its processors with the weakest processor and its communication links with the slowest link. In the part concerning the heterogeneity of processor performances, the statement will be experimentally corroborated in section 4.

A natural solution of this problem is heterogeneous distribution of both processes over the processors and data among the processes, taking into account differences in performances of processors and speeds of communication links. As it has been demonstrated in [2], such a distribution allows to achieve much better distribution of computations over processors of a heterogeneous computing network and, hence, to utilize its performance potential more efficiently.

Such heterogeneous distribution is a complex problem whose solution needs adequate tools. Designed specially to write efficient and portable parallel application for heterogeneous networks of computers, the mpC language [3] is just such a tool. This language is an ANSI C superset allowing to write applications adapting to differences in performances of both processors and communication links of any particular executing network. The basic idea is that an mpC application explicitly builds in run time an abstract heterogeneous computing network and distributes data, computations and communications over the network. The mpC programming system uses this information in run time to map the abstract network to any real executing network of computers in such a way that ensures efficient running of the application on the real network. More about mpC as well as the mpC free software can be found at http://www.ispras.ru/~mpc.

In the paper, we consider only the heterogeneity of processor performances. We propose a heterogeneous distribution strategy based on homogeneous distribution of involved processes over processors with each process running on a separate processor and heterogeneous block cyclic distribution of processed matrix over the processes. We investigate the strategy, named the HoHe strategy, using a typical linear algebra problem - the Cholesky factorization of square dense matrices. The problem was chosen as a well-known example of the practically important problem, whose parallel solution needs careful balancing computations and communications. Our implementation of parallel Cholesky factorization is based on the algorithm implemented in ScaLAPACK [5]. Both distribution of the involved processes and the parallel Cholesky factorization proper are performed by an mpC program calling BLAS and LAPACK functions for local computations. Note, that in this case, the parallel algorithm implemented by

the mpC program for the HoHe strategy is, in general, the same as implemented by ScaLAPACK function PDPOTRF for the HoHo strategy.

Section 2 introduces the heterogeneous block cyclic matrix distribution and describes the HoHe strategy. Section 3 shortly introduces the mpC language and describes the implementation of the HoHe strategy in mpC. Section 4 gives experimental results of Cholesky factorization on a network of heterogeneous workstations using the distribution strategies.

2 Homogeneous distribution of processes with heterogeneous data distribution

The traditional homogeneous block cyclic data distribution [4] is determinated by grid parameters P and Q and block parameters m and n. The distribution partitions the matrix into *generalized* blocks of the size $(m \cdot P)$x$(n \cdot Q)$, each of which in its turn partitioned into $(P \cdot Q)$ blocks of the same size, each going to a separate process.

Figure 1 shows an example of the homogeneous block cyclic distribution of a 12x12 matrix, block-partitioned with the block size 2x2 (m=2, n=2), over a 2x3 process grid (P=2, Q=3). In this case, generalized blocks are of size 4x6.

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	1	1	2	2	0	0	1	1	2	2
1	0	0	1	1	2	2	0	0	1	1	2	2
2	3	3	4	4	5	5	3	3	4	4	5	5
3	3	3	4	4	5	5	3	3	4	4	5	5
4	0	0	1	1	2	2	0	0	1	1	2	2
5	0	0	1	1	2	2	0	0	1	1	2	2
6	3	3	4	4	5	5	3	3	4	4	5	5
7	3	3	4	4	5	5	3	3	4	4	5	5
8	0	0	1	1	2	2	0	0	1	1	2	2
9	0	0	1	1	2	2	0	0	1	1	2	2
10	3	3	4	4	5	5	3	3	4	4	5	5
11	3	3	4	4	5	5	3	3	4	4	5	5

(a) matrix distribution

	0	1	6	7	2	3	8	9	4	5	10	11
0	0	0	0	0	1	1	1	1	2	2	2	2
1	0	0	0	0	1	1	1	1	2	2	2	2
4	0	0	0	0	1	1	1	1	2	2	2	2
5	0	0	0	0	1	1	1	1	2	2	2	2
8	0	0	0	0	1	1	1	1	2	2	2	2
9	0	0	0	0	1	1	1	1	2	2	2	2
2	3	3	3	3	4	4	4	4	5	5	5	5
3	3	3	3	3	4	4	4	4	5	5	5	5
6	3	3	3	3	4	4	4	4	5	5	5	5
7	3	3	3	3	4	4	4	4	5	5	5	5
10	3	3	3	3	4	4	4	4	5	5	5	5
11	3	3	3	3	4	4	4	4	5	5	5	5

(b) distribution from processor point-of-view

Fig. 1. Example of a homogeneous block cyclic distribution of a 12x12 matrix over 2x3 process grid with the block size 2x2.

Let an executing computer system consists of a set $\mathbf{L}$ of processors and $card(\mathbf{L}) \geq P \cdot Q$. Let $P \cdot Q$ processes be distributed over $P \cdot Q$ most powerful processors in such a way, that just one process goes to each of these processors, and matrix M be distributed over the processes in accordance with the heterogeneous block cyclic distribution presented below.

We also suppose that the positive real number r_{ij} is associated with each of the processors and characterizes its relative performance ($i \in [0, P-1]$, $j \in [0, Q-1]$). Then, in addition to four numbers P, Q, m and n, parametrizing the homogeneous block cyclic distribution, the heterogeneous one is parametrized by the PxQ matrix $\mathbf{R} = \{r_{ij}\}$, elements of which characterize performances of the corresponding processors. Its main difference from the homogeneous distribution lies in heterogeneous data distribution inside a generalized block. Like in case of the homogeneous distribution, a generalized $(m \cdot P)$x$(n \cdot Q)$ block is partitioned into $(P \cdot Q)$ blocks. But in case of the heterogeneous distribution, the blocks are not of the same size, but their sizes m_{ij}xn_{ij} depend on performances of processors. In the paper, we consider the simplest choice of m_{ij} and n_{ij} deduced from the assumption that part of matrix M processed by a separate processor is proportional to its performance. That is,

$$m_{ij} \cdot n_{ij} = \frac{m \cdot P \cdot n \cdot Q \cdot r_{ij}}{\sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} r_{ij}}.$$

In this case mxn is the average size of the uneven blocks.

In particular, the above condition can be satisfied by the following choice of m_{ij} and n_{ij}:

$$n_{ij} = n_j = \frac{\sum_{i=0}^{P-1} r_{ij} \cdot n \cdot Q}{\sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} r_{ij}}, \qquad m_{ij} = \frac{r_{ij} \cdot m \cdot P}{\sum_{i=0}^{P-1} r_{ij}}.$$

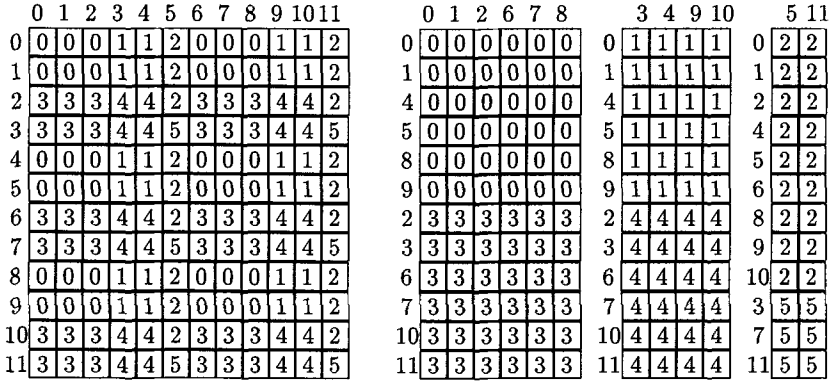
Figure 2 shows an example of the heterogeneous block cyclic distribution of a 12x12 matrix over a 2x3 processor grid (P=2, Q=3) with the average block size 2x2 (m=2, n=2), the generalized-block size 4x6 and the following matrix of processor performances

$$\mathbf{R} = \begin{pmatrix} 6 & 4 & 2 \\ 5 & 3 & 1 \end{pmatrix}.$$

With that choice of the m_{ij}, a row of the distributed matrix does not have to belong to the same row of the process grid, that can lead to additional communication overheads.

3 Implementation of heterogeneous distribution of computations in mpC

mpC [3] is a parallel language aimed at efficiently-portable modular programming heterogeneous networks of computers. It provides facilities for specification of requirements on resources, necessary for efficient execution of the parallel application, and the mpC programming system tries to satisfy the requirements taking into account peculiarities of any particular heterogeneous network of computers.



(a) matrix distribution (b) distribution from processor point-of-view

Fig. 2. Example of a heterogeneous block cyclic distribution of a 12x12 matrix over 2x3 processor grid with the average block size 2x2.

The main idea underlying mpC is that an mpC application explicitly defines a dynamic abstract heterogeneous computing network and distributes data, computations and communications over the network. The mpC programming system uses this information in run time to map the abstract computing network to any real executing network of computers in such a way that ensures efficient running of the application on this real network.

The mpC language is an ANSI C superset that introduces a new kind of managed resource, *computing space*, defined as a set of virtual processors of difference performances connected with links of different communication speeds. In run time, the virtual processors are represented by actual processes of the particular running parallel application. The programmer manages the computing space by means of creating and discarding regions of the computing space, named *network objects*, just like he manages storage creating and discarding data objects (regions of storage). At any moment of program execution, just a set of defined network objects represents the abstract computing network.

mpC application used in experiments, implements the HoHe strategy in three steps:

1. The information about performances of processors of the executing real network is updated in run time by means of execution of corresponding computations as a benchmark (namely, Cholesky factorization of a small matrix by means of the LAPACK routine `dpotf2`).
2. A network object, executing the corresponding computations, is defined in such a way that each of the $P \cdot Q$ most powerful processors of the executing network will execute only one process involved in the computations, and all the involved processes form a PxQ process grid.

3. A driver of the Cholesky factorization reads from a file problem parameters (matrix and block sizes), forms a distributed test matrix and performs its Cholesky factorization on the PxQ process grid. Each virtual processor of the network object computes a portion of the matrix, proportional to its performance in accordance with the heterogeneous block cyclic matrix distribution described in section 2.

The above 2-dimensional strategy is a generalization of the 1-dimensional strategy based on heterogeneous data distribution and presented in [6]. The latter describes in details an mpC application carring out Cholesky factorization for that strategy.

4 Experimental results

We compared two distribution strategies:

- The HoHo strategy - "Homogeneous distribution of processes over processors - Homogeneous distribution of data over the processes" - the traditional distribution strategy implemented in ScaLAPACK.
- The HoHe strategy - "Homogeneous distribution of processes over processors - Heterogeneous distribution of data over the processes" implemented in mpC.

The comparison was performed for the Cholesky factorization on a network of workstations. For our experiments, we used a part of a local network consisting of 6 uniprocessor Sun workstations of different performances interconnected via 10 Mbits Ethernet. MPICH 1.0.13 was used as a particular communication platform. All workstations executed the same copy of code. Performances of the workstations, obtained by means of execution of the LAPACK routine `dpotf2` performing serial Cholesky factorization, is shown in table 1.

1	2	3	4	5	6
190	190	190	190	100	280

Table 1. Performances of processors demonstrated on serial Cholesky factorization

In all our experiments we used *parallelization efficiency* as a factor characterizing how fully the parallel application utilizes the performance potential of the executing network of computers. Parallelization efficiency was defined as S_{real}/S_{ideal}, where S_{real} was the real speedup achieved by the parallel application on the parallel system, and S_{ideal} was the ideal speedup that could be achieved while parallelizing the problem. The latter was calculated as the sum of performances of processors, constituting the executing parallel system, divided by the performance of a base processor. All real speedups were calculated relative

to the sequential LAPACK routine `dpotf2` performing Cholesky factorization on the base processor.

Figure 3 presents the parallelization efficiency achieved by the compared distribution strategies while running on the homogeneous network `1-2-3-4` consisting of 4 workstations 1, 2, 3, and 4 with the optimal process grid (2x2) and block sizes. One can see that for large matrices the parallelization efficiency of the HoHo strategy is 8% higher than that of the HoHe strategy. Note, that the result is not due to the non-Cartesianity of the heterogeneous block cyclic matrix distribution (as shown in figure 2), since the same result was also obtained for a mpC application implementing the homogeneous block cyclic matrix distribution. So, it can be explained by more efficient implementation of Cholesky factorization in ScaLAPACK.

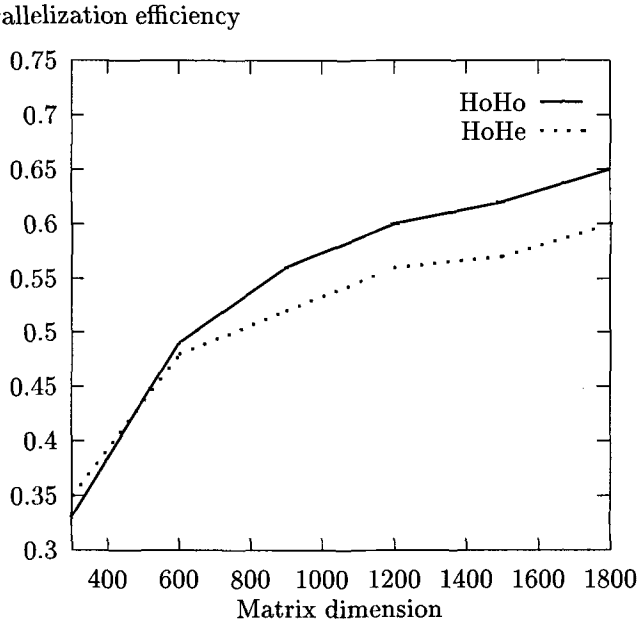


Fig. 3. Parallelization efficiencies achieved by the HoHo (ScaLAPACK) and HoHe (mpC) distribution strategies on the homogeneous network consisting of workstations 1, 2, 3, and 4.

Now, let us consider the heterogeneous network `1-2-5-6`, consisting of workstations 1, 2, 5 and 6 and having the same total power of processors as the homogeneous network `1-2-3-4`. Parallelization efficiencies achieved by the different distribution strategies for this heterogeneous network are shown in figure 4 (as before, optimal sizes of process grid and block were used).

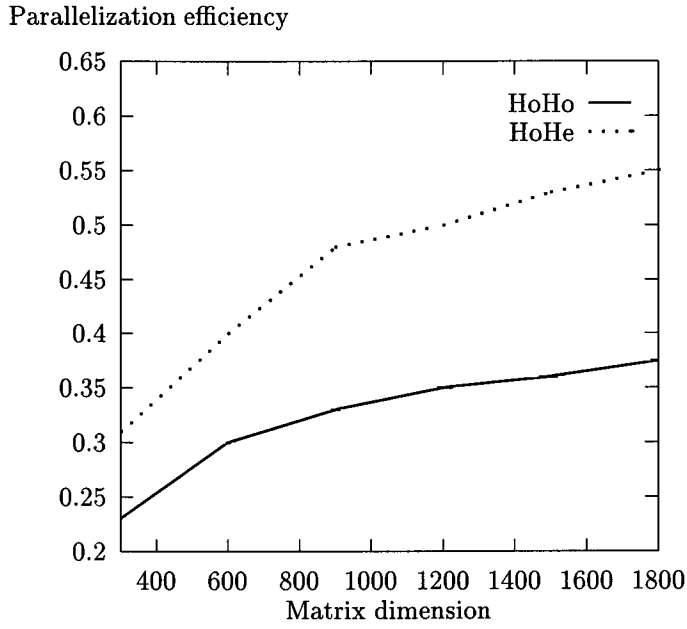


Fig. 4. Parallelization efficiencies achieved by the HoHo (ScaLAPACK) and HoHe (mpC) distribution strategies on the heterogeneous network consisting of workstations 1, 2, 5, and 6.

One can see, that in this case the HoHe strategy is much more efficient then the HoHo strategy. One can also see that its efficiency lowered insignificantly while transferring from the homogeneous network to the heterogeneous one.

The HoHo strategy demonstrates much worse parallelization efficiency on the heterogeneous network `1-2-5-6` than on the homogeneous network `1-2-3-4`, in spite of the two networks being characterized by the same total processor performance - 760 (the sum of performances of participating processors). It conforms to the statement formulated in Introduction that the total running time (and, hence, the parallelization efficiency) provided by the HoHo strategy on a heterogeneous network is approximately the same as on a homogeneous network obtained from this heterogeneous one by means of replacement of both its processors with the weakest processor and its communication links with the slowest link. Indeed, the parallelization efficiency of the corresponding application on the network `1-2-3-4` is 1.7 times higher than on the network `1-2-5-6`. At the same time, according to the above statement, this factor should be close to the ratio of performances of the weakest processors in networks `1-2-3-4` and `1-2-5-6`, that is, be close to 1.9. Note, that the faster are communication links and the less powerful are processors, the higher parallelization efficiency is achieved. Therefore, the factor should be a little bit less than 1.9, since the network `1-2-5-6` has better ratio of the communication speed and the weakest processor per-

formance than the network `1-2-3-4`. Thus, we can conclude that the obtained experimental results are in good conformance with the above statement.

Figure 5 shows the parallelization efficiency achieved by the two distribution strategies on the heterogeneous network `1-2-3-4-5-6` consisting of all 6 workstations. As before, optimal sizes of process grid and block were used for each of the strategies (3x2 process grid turned out optimal for the both strategies). In this case, as before, the HoHe strategy is much more efficient then the HoHo strategy.

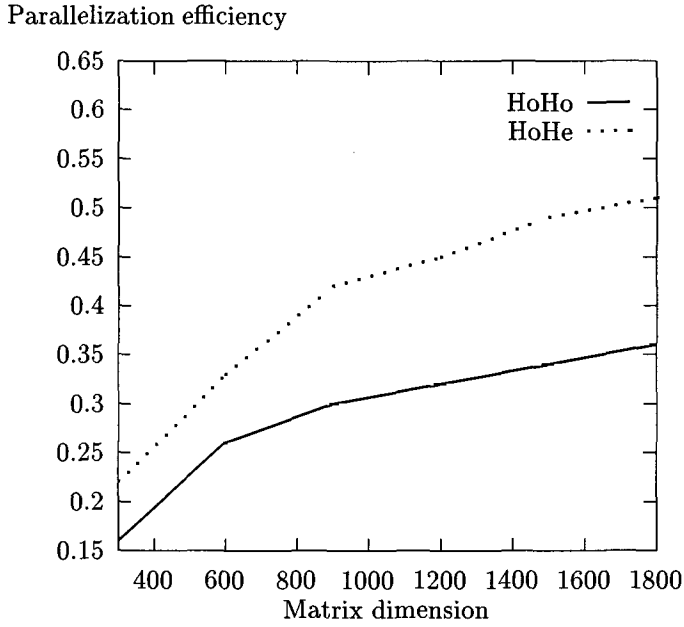


Fig. 5. Parallelization efficiencies achieved by the HoHo (ScaLAPACK) and HoHe (mpC) distribution strategies on the heterogeneous network consisting of workstations 1, 2, 3, 4, 5, and 6.

5 Conclusion

Numeric software developed for computations in homogeneous environments does not allow to utilize all performance potential of heterogeneous networks. It has been demonstrated that in this case a heterogeneous network is equal to some homogeneous network obtained from the heterogeneous one by means of replacement its processors with the weakest processor.

A natural way to answer this challenge is to develop dedicated numeric software aimed at heterogeneous environments. Such software should take into account the heterogeneity of processor performances and speeds of communication

links and support heterogeneous distribution of processes, involved in computations, over processors and/or data over the processes. The distribution strategy presented in the paper is based on heterogeneous block cyclic distribution of data. It has been shown that for heterogeneous parallel environments the heterogeneous strategy is much more efficient than the traditional homogeneous strategy.

Implementation of heterogeneous parallel algorithms needs appropriate porgramming languages and tools supporting and facilitating programming heterogeneous computations. In our research, we use the mpC parallel language and its free supportive programming environment developed in the Institute for System Programming of the Russian Academy of Sciences and just aimed at portable and efficient programming for heterogeneous environments.

The paper has investigated the heterogeneous distribution strategy taking into account only the heterogeneity of processor performances. Obviously, that the heterogeneity of link performances has no less an impact on parallelization efficiency. The mpC language has some means for taking into account that heterogeneity too, but this more complex problem still is waiting its solution.

6 Acknowledgments

We would like to thank Jack Dongarra and Antoine Petitet for their useful comments on preliminary versions of the paper.

References

1. L. S. Blackford, A. Cleary, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, A. Petitet, H. Ren, K. Stanley, and R. C. Whaley Practical Experience in the Dangers of Heterogeneous Computing UT, CS-96-330, July 1996.
2. D.Arapov, A.Kalinov, A.Lastovetsky, I.Ledovskih, and T.Lewis "A Programming Environment for Heterogeneous Distributed Memory Machines", *Proceedings of the Sixth Heterogeneous Computing Workshop (HCW'97)* , IEEE Computer Society Press, Geneva, Switzerland, April 1, 1997.
3. A.Lastovetsky, The mpC Programming Language Specification. Technical Report, ISPRAS, Moscow, December 1994.
4. B.Hendrickson and D.Womble,"The Torus–wrap Mapping for Dense Matrix Calculations on Massively Parallel Computers", SIAMSSC, 15(5), 1994.
5. J. Choi, J. J. Dongarra, S. Ostrouchov, A. P. Petitet, D. W. Walker, and R. C. Whaley "The Design and Implementation of the ScaLAPACK LU, QR, and Cholesky Factorization Routines" UT, CS-94-246, September, 1994.
6. D.Arapov, A.Kalinov, A.Lastovetsky and I.Ledovskih "Experiments with mpC: Efficient Solving Regular Problems on Heterogeneous Networks of Computers via Irregularization", *Proceedings of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel (IRREGULAR'98)* , Lecture Notes in Computer Science 1457, Berkley, CA, USA, August 9-11, 1998.