Parallel Solvers for Dense Linear Systems for Heterogeneous Computational clusters

Ravi Reddy
School of Computer Science and Informatics
University College Dublin
manumachu.reddy@ucd.ie

Alexey Lastovetsky
School of Computer Science and Informatics
University College Dublin
alexey.lastovetsky@ucd.ie

Pedro Alonso
Department of Information Systems and Computation
Polytechnic University of Valencia
palonso@dsic.upv.es

Abstract

This paper describes the design and the implementation of parallel routines in the Heterogeneous ScaLAPACK library that solve a dense system of linear equations. This library is written on top of HeteroMPI and ScaLAPACK whose building blocks, the de facto standard kernels for matrix and vector operations (BLAS and its parallel counterpart PBLAS) and message passing communication (BLACS), are optimized for heterogeneous computational clusters.

We show that the efficiency of these parallel routines is due to the most important feature of the library, which is the automation of the difficult optimization tasks of parallel programming on heterogeneous computing clusters. They are the determination of the accurate values of the platform parameters such as the speeds of the processors and the latencies and bandwidths of the communication links connecting different pairs of processors, the optimal values of the algorithmic parameters such as the total number of processes, the 2D process grid arrangement and the efficient mapping of the processes executing the parallel algorithm to the executing nodes of the heterogeneous computing cluster.

We describe this process of automation followed by presentation of experimental results on a local heterogeneous computing cluster demonstrating the efficiency of these solvers.

1. Introduction

This paper describes the design and the implementation of the parallel routines in the Heterogeneous ScaLAPACK library that solve a dense system of linear equations. These routines compute the solution to a (real, complex) system of linear equations $A \times X = B$, where A, X, and B are distributed matrices. We start with presentation of related works, which have produced solely heterogeneous parallel algorithms/strategies.

There are two strategies of distribution of computations that can be used to implement parallel solvers for dense linear systems for Heterogeneous Computing Clusters (HCCs) [1].

- HeHo - heterogeneous distribution of processes over processors and homogeneous block distribution of data over the processes. This is implemented in mpC language [2,3] with calls to ScaLAPACK [4];
- HoHe - homogeneous distribution of processes over processors with each process running on a separate processor and heterogeneous block cyclic distribution of data over the processes. This is implemented in mpC language with calls to BLAS [4] and LAPACK[4].

The HeHo strategy is a multiprocessing approach that is commonly used to accelerate ScaLAPACK programs on HCCs. The strategies are compared using Cholesky factorization on a HCC. The authors show that for HCCs, the strategies HeHo and HoHe are more efficient than the traditional homogeneous strategy HoHo (homogeneous distribution of processes over processors with each process running on a separate processor and homogeneous block cyclic distribution of data over the processes implemented in ScaLAPACK). The main disadvantage of the HoHe strategy is non-Cartesian nature of the data distribution, which is the distribution where each processor has to communicate with more than four direct neighbors. This leads to additional communications that can be crippling in the case of

large networks. Moreover, the non-Cartesian nature leads to nonscalability of the linear algebra algorithms that are proven to be scalable in the case of Cartesian data distribution and networks allowing parallel communications.

Beaumont et al. [5] present a proposal motivating provision of parallel solvers for dense linear systems in the form of a library for HCCs. The authors discuss HoHe strategies to implement matrix products and dense linear system solvers for HCCs and present them as a basis for a successful extension of the ScaLAPACK library to HCCs. They show that extending the standard ScaLAPACK block-cyclic distribution to heterogeneous 2D grids is a difficult task. However, providing this extension is only the first step. The ScaLAPACK software must then be completely redesigned and rewritten, which is not a trivial effort.

The multiprocessing strategy (HeHo), however, is easier to accomplish. It allows the complete reuse of high-quality software such as ScaLAPACK on HCCs with no redesign efforts and provides good speedups. It can be summarized as follows:

- The whole computation is partitioned into a large number of equal chunks;
- Each chunk is performed by a separate process;
- The number of processes run by each processor is proportional to its speed.

Thus, while distributed evenly across parallel processes, data and computations are distributed unevenly over processors of the HCC so that each processor performs the volume of computations proportional to its speed.

There have been several research contributions illuminating the merits of this strategy. Kishimoto and Ichikawa [6] use it to estimate the best processing element (PE) configuration and process allocation based on an execution-time model of the application. Kalinov and Klimov [7] investigate the HeHo strategy where the performance of the processor is given as a function of the number of processes running on the processor and the amount of data distributed to the processor. They present an algorithm that computes optimal number of processes and their distribution over processors minimizing the execution time of the application. Cuenca et al. [8] analyze automatic optimization techniques in the design of parallel dynamic programming algorithms on heterogeneous platforms. The main idea is to automatically determine the optimal values of a number of algorithmic parameters such as number of processes, number of processors, and number of processes per processor.

So there has been a proliferation of research efforts presenting approaches to implement parallel solvers for HCCs but scientific software based on these approaches is virtually nonexistent. However, from these research efforts, one can conclude that any software aspiring to provide automatically tuned parallel linear algebra programs for HCCs must automate the following essential and complicated tasks, which are also described in [9]:

- Determination of the accurate values of platform parameters such as speeds of the processors, latencies and bandwidths of the communication links connecting different pairs of processors. These parameters compose the performance model of the executing heterogeneous platform;
- Determination of the optimal values of algorithmic parameters such as data distribution blocking factor and 2D process grid arrangement for linear algebra kernel, and
- Efficient mapping of the processes executing the parallel algorithm to the executing nodes of the HCC.

The Heterogeneous ScaLAPACK library [10] is one such software package, currently under development, providing automatically tuned parallel linear algebra programs for HCCs. It performs all the aforementioned automations. It uses the multprocessing strategy (HeHo) and is built on top of HeteroMPI [11] and ScaLAPACK.

In this paper, we describe the efforts made to implement parallel routines solving dense linear systems in this library. We present details of the implementation of one of the solvers, PDGESV, which computes the solution to a real system of linear equations. At the same time, we also explain how the optimal values of the algorithmic parameters such as the data distribution blocking factor and 2D process grid arrangement are determined and how efficient mapping of the processes to the executing nodes of the HCC is accomplished.

The rest of the paper is organized as follows. We start with an overview of the software in the Heterogeneous ScaLAPACK package. In section 3, we explain how the optimal values of the algorithmic parameters are determined. In section 4, we present the experimental results of execution of a couple of Heterogeneous ScaLAPACK programs on a local HCC. We conclude the paper by outlining our future research goals.

2. Determination of the Optimal Algorithmic Parameters

The principal routines in Heterogeneous ScaLAPACK package are the context creation

functions for the ScaLAPACK routines. There is a context creation function for each and every ScaLAPACK routine. It provides a context for the execution of the ScaLAPACK routine, but most importantly performs the critical work of automating the difficult optimization tasks. All the context creation routines have names of the form **hscal_pxyyzzz_ctxt**. The context creation/destruction routines call interface functions of HeteroMPI runtime system (the main routines being **HMPI_Recon**, **HMPI_Timeof**, **HMPI_Group_auto_create**).

The first step in the implementation of the context creation routine for a ScaLAPACK routine is the writing of its performance model using the HeteroMPI's Performance Model Definition Language (PMDL), which is a subset of mpC.

The writing of performance models of all the ScaLAPACK routines solving a dense system of linear equations (PxGESV, PxPOSV) has been the most laborious and intricate effort of this project. The key design issues were (a) *accuracy*, to facilitate accurate prediction of the execution time of the ScaLAPACK routine, (b) *efficiency*, to execute the performance model in reasonable execution time, (c) *reusability*, as these performance models are to be used as building blocks for the performance models of ScaLAPACK routines, and (d) *preservability*, to preserve the key design features of underlying ScaLAPACK package.

The reader is referred to the Heterogeneous ScaLAPACK programmer's manual for more details of the user interface. The complete performance model descriptions of the routines PxGESV and PxPOSV can be studied from the files `pm_pxgesv.mpc` and `pm_pxposv.mpc` in the directory `/SRC` of the Heterogeneous ScaLAPACK package available at the URL [10].

The PDGESV context constructor routine **hscal_pdgesv_ctxt** is the main function automating the difficult optimization tasks of parallel programming on HCCs. They are the determination of the accurate values of the platform parameters such as the speeds of the processors and the latencies and bandwidths of the communication links connecting different pairs of processors, the optimal values of the algorithmic parameters such as the 2D process grid arrangement and efficient mapping of the processes executing the parallel algorithm to the executing nodes of the HCC.

3.1 Data Distribution Blocking Factor

The context constructor routine **hscal_pdgesv_ctxt** uses the HeteroMPI function **HMPI_Timeof**, as shown below, to determine the optimal value of the data distribution blocking factor.

```
int b, opt_b;
double time, min_time = DBL_MAX; void *modelp;
for (b = 1; b <= N; b++) {
   // Fill the performance model parameters
   modelp[0] = N; modelp[1] = NRHS;
   …
   modelp[8] = b; // DESCA[MB_]
   modelp[9] = b; // DESCA[NB_]
   …
   HMPI_Recon(&hscal_dgemm_benchmark, modelp);
   if (HMPI_Is_host())
      time = HMPI_Timeof(&hscal_model_pdgesv,
                         modelp);
   if (time < min_time)
      opt_b = b; min_time = time;
}
```

The function **HMPI_Timeof** is used to estimate the execution time of the algorithm on the underlying hardware without its real execution. This is a local operation that can be called by any process, which is a member of the group associated with the predefined communication universe of HeteroMPI. The parameters to this function are the handle, **hscal_model_pdgesv**, generated from the compilation of the performance model of the PDGESV routine, and the parameters to this performance model. As one can see from the code snippet, this estimation is performed for each possible set of values to the parameters to the performance model. Using the execution time predicted for each set, the optimal value can be determined, which would be the one resulting in minimum estimated execution time.

The estimation is based on the performance model of the PDGESV routine and the performance model of the executing network of computers, which reflects the state of this network just before the execution of the PDGESV routine. The function **HMPI_Recon** is used to dynamically update the estimation of processor speeds at runtime. This is a collective operation and must be called by all the processes running in the Heterogeneous ScaLAPACK application. The performance model of the executing network of computers is summarized as follows:

- The performance of each processor is characterized by the execution time of the same serial code (takes place during the execution of **HMPI_Recon**)
  - The serial code is provided by the application programmer;
  - It is supposed that the code is representative for the computations performed during the execution of the application;
  - The code is performed at runtime in the points of the application specified by the application

programmer. Thus, the performance model of the processors provides current estimation of their speed demonstrated on the code representative for the particular application.

In this case, the serial code, *hscal_dgemm_benchmark*, performs a local DGEMM update of (**N**/**bp**)×**b** and **b**×(**N**/**bq**) matrices where **b** is the data distribution blocking factor. The value of the parameter **bp** is the square root of the total number of processes available for computation. The value of parameter **bq** is equal to total number of processes divided by **bp**;

- The communication model [3] is seen as a hierarchy of homogeneous communication layers. Each is characterized by the latency and bandwidth. Unlike the performance model of processors, the communication model is static, a shortcoming that would be addressed in our future work.

The estimation procedure is explained in detail in [3]. This procedure to determine the optimal value of the data distribution blocking factor is now performed during the installation of the Heterogeneous ScaLAPACK package rather than for every call of the context constructor routine for performance reasons.

3.2 Two-dimensional Process Grid Arrangement

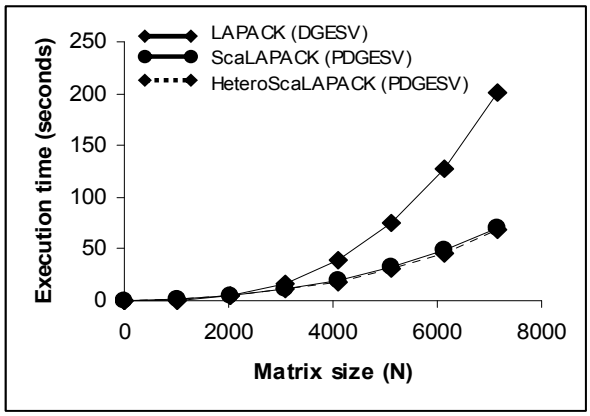
The context constructor routine **hscal_pdgesv_ctxt** calls the HeteroMPI function **HMPI_Group_pauto_create** to determine the optimal values for the total number of processes, the number of process rows, the number of process columns, and efficient mapping of the processes to the executing computers of the HCC. Its operation is shown below using HeteroMPI calls.

```
int i, k, pa, p, opt_p, q, opt_q, *a,
terminate = 0;
double t, mint=DBL_MAX; void *modelp = NULL;
// The host process
if (HMPI_Is_host()) {
    int np =
    HMPI_Group_size(HMPI_COMM_WORLD_GROUP);
    for (k=np; k>=1; k--) {
    HMPI_Get_2D_process_arrangements(
        k, &pa, &a);
    for (i=0; i<pa; i++) {
    // Estimate the execution time for each
    // process arrangement (p,q)
    t = HMPI_Timeof(&hscal_model_pdgesv,
                    modelp);
    if (t < mint) {
       terminate = 0; mint = t; opt_p = p;
       opt_q = q;
    }
    if (terminate) { break; }
    terminate=1;
}
modelp[8] = opt_b; // DESCA[MB_]
modelp[9] = opt_b; // DESCA[NB_]
…
modelp[1+1+1+1+DLEN1_+1+1+DLEN_] = opt_p;
modelp[1+1+1+1+DLEN1_+1+1+DLEN_+1] = opt_q;
}
// Create a HeteroMPI group of processes
// with the optimal algorithmic param values
HMPI_Group_create(gid, modelp);
return HMPI_SUCCESS;
```

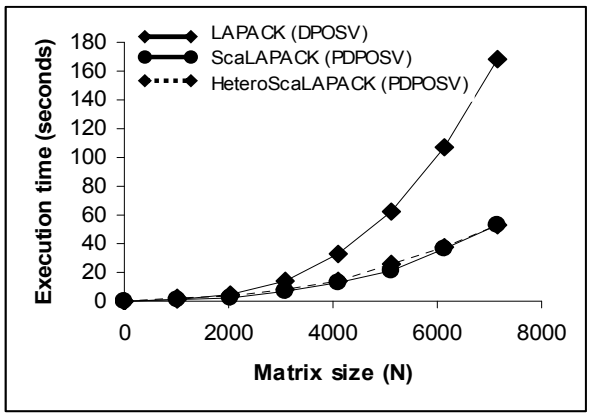
The algorithm can be summarized as follows: The number of steps of the algorithm is represented by the variable, **k**. For **k**=1, the total number of processes available for computation, **np**, is determined. All the possible two-dimensional process arrangements, (**p**,**q**), whose product is **np**, are obtained using the function, **HMPI_Get_2D_process_arrangements**. For example, if the total number of processes available is 25, then the possible two dimensional process arrangements are {(1,25), (5,5), (25,1)}.

Each such process arrangement, (**p**,**q**), is filled into the array of parameters to the performance model of the PDGESV routine. The function call **HMPI_Timeof** is then invoked to estimate the execution time of the algorithm. One of the inputs to this function call is the handle, **hscal_model_pdgesv**, which encapsulates all the features of the performance model in the form of a set of functions generated by the compiler from the description of the performance model of the PDGESV routine. The function call **HMPI_Timeof** invokes the mapping algorithms of the Heterogeneous ScaLAPACK runtime system to select such a mapping that is estimated to ensure the fastest execution of the parallel algorithm for that process arrangement. The selection process is described in detail in [3,11]. It is based on the performance model of the PDGESV routine and the performance model of the executing network of computers, which reflects the state of the network just before the execution of the PDGESV routine. During the selection process, the Heterogeneous ScaLAPACK runtime system searches for some approximate solution that can be found in some reasonable interval of time by probation of a subset of all possible mappings. From the execution times predicted for all the possible process arrangements, the process arrangement, (**opt_p**, **opt_q**), that results in the least estimated time of execution of the algorithm is determined.

For the next step, the total number of processes is decremented by one. The possible two-dimensional process grid arrangements are again obtained and evaluated using the function **HMPI_Timeof**. The



(a)



(b)

Figure 1. Execution times of LAPACK, ScaLAPACK, and Heterogeneous ScaLAPACK programs. (a) Solving the same real system of linear equations involving a general dense matrix *A* and (b) Solving the same real system of linear equations involving a Hermitian positive definite matrix *A*.

algorithm continues if a process arrangement is found for which the estimated execution time is less than the estimated execution time of the process arrangement (`opt_p`, `opt_q`) determined in the previous step. Otherwise the algorithm terminates.

The optimal values of the blocking factor and the 2D process grid arrangemnt (`opt_p`, `opt_q`) are then passed as performance model parameters to the function call, `HMPI_Group_create`, which creates a HeteroMPI group of MPI processes that participate in the execution of the parallel application. These processes become members of the heterogeneous PDGESV context. This function call is a collective operation and must be called by all the processes available for computation in the predefined communication universe of HeteroMPI. Heuristics are used to reduce the number of process arrangements evaluated.

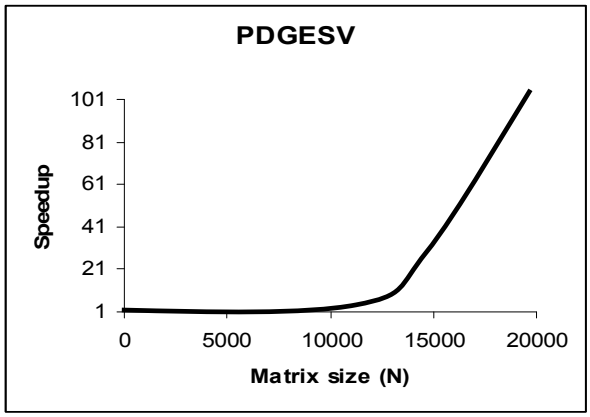
3. Experimental Results

We use a small heterogeneous local network of 16 different Linux processors (hcl01-hcl16) for the experiments. The specifications of the network are available at the URL http://hcl.ucd.ie/Hardware/Cluster+Specifications. The network is based on 2 Gbit Ethernet with a switch enabling parallel communications between the computers. The software used is MPICH-1.2.5, ScaLAPACK-1.8.0 and ATLAS, which provides an optimized BLAS library.

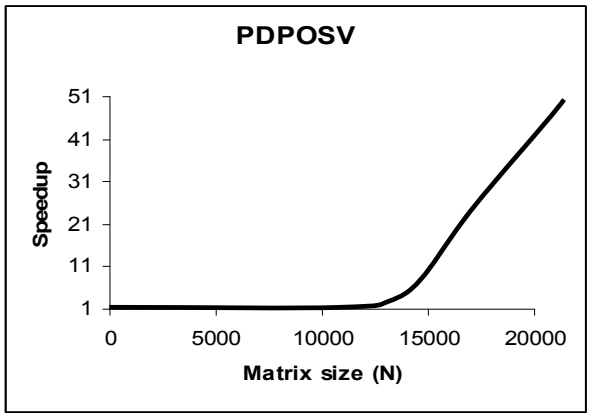
We use a couple of Heterogeneous ScaLAPACK programs for the experimental results. They employ PDGESV, which computes the solution to a real system of linear equations employing LU decomposition of a general distributed matrix, and PDPOSV, which computes the solution to a real system of linear equations using Cholesky decomposition of a Hermitian positive definite distributed matrix.

The Heterogeneous ScaLAPACK program uses the multiprocessing approach, where more than one process is run on each processor. The number of processes to run on each processor during the program startup is determined automatically by the Heterogeneous ScaLAPACK command-line interface tools. A simple heuristic used is the heterogeneity of the network. The heterogeneity of the network due to the heterogeneity of the processors is calculated as the ratio of the absolute speed of the fastest processor to the absolute speed of the slowest processor. For example, consider the benchmark code of a local DGEMM update of two matrices 2560×128 and 128×2560, the absolute speeds of the processors hcl01-hcl16 in million flop/s performing this update are {7696, 5196, 7852, 14418, 8000, 8173, 7288, 7396, 9037, 8987, 13661, 14194, 11182, 14410, 12008, 15257}. As one can see, hcl16 is the fastest processor and hcl02 is the slowest processor. The heterogeneity in this case ≈ 3. The command-line tools, therefore, would run three processes on each processor during the program startup.

The speedup, which is shown in the figures, is calculated as the ratio of the execution time of the homogeneous ScaLAPACK program and the execution time of the Heterogeneus ScaLAPACK program. Dense square matrices of size **N**×**N** were used in the experiments and **NRHS** is equal to **N**. The homogeneous ScaLAPACK programs use the default parameters recommended by the ScaLAPACK user's guide which are to (a) use the best BLAS and BLACS libraries available, (b) use a data distribution block size of 64, (c) use a square processor grid and (d) execute



(a)



(b)

Figure 2. Speedup of Heterogeneous ScaLAPACK over ScaLAPACK.

no more than one process per processor. The ScaLAPACK programs, therefore, use a 4×4 grid of processes (using one process per node configuration).

For equitable comparision, the ScaLAPACK programs must be optimized for the HCC. However one of the goals of this paper is to show how the Heterogeneous ScaLAPACK package automates this procedure of optimization and therefore the comparision is done using unoptimized ScaLAPACK available online. However, the process of optimization is not trivial as demonstrated in this work.

Figure 1(a) shows the execution times of a sequential application employing LAPACK, parallel ScaLAPACK and Heterogeneous ScaLAPACK programs solving the same real system of linear equations. The LAPACK program employs DGESV routine and uses optimized BLAS library (ATLAS). For Figure 1(b), the LAPACK program employs DPOSV routine whereas the ScaLAPACK and Heterogeneous ScaLAPACK programs employ PDPOSV routine. The LAPACK programs are executed on the fastest processor hcl16. They fail for problem sizes, (**N**>7168), due to the memory limitations of the processor. The ScaLAPACK and Heterogeneous ScaLAPACK programs perform better than the sequential LAPACK programs.

Figure 2 shows the experimental results from the execution of the ScaLAPACK and Heterogeneous ScaLAPACK programs employing the routines PDGESV and PDPOSV. There is not much difference in execution times between ScaLAPACK and Heterogeneous ScaLAPACK programs executing PDGESV for the range of problem sizes ($0 \leq \mathbf{N} \leq 11264$) and PDPOSV for the range of ($0 \leq \mathbf{N} \leq 13312$). But beyond these problem sizes, the ScaLAPACK programs perform very poorly due to severe paging of one/more of the processors. This is due to poor load balancing caused by the HoHo strategy used by ScaLAPACK.

There are a few reasons for the good speedups delivered by the Heterogeneous ScaLAPACK programs on HCCs for all problem sizes. The first reason is the better load balance achieved through proper allocation of processes involved in the execution of the algorithm to the processors. During the creation of a HeteroMPI group of processes in the context creation routine, the mapping of the parallel processes in the group is performed such that the number of processes running on each processor is proportional to its speed. In other words, while distributed evenly across parallel processes, data and computations are distributed unevenly over processors of the heterogeneous network, and this way each processor performs the volume of computations as proportional to its speed as possible.

Secondly, the optimal values of the algorithmic parameters such as the blocking factor that are automatically determined by the Heterogeneous ScaLAPACK library. In the case of LU decomposition (employed in PDGESV) or Cholesky Factorization (employed in PDPOSV), since the largest fraction of the work takes place in the update of the trailing matrix A_{22}, all processes should, therefore, participate in its update to obtain maximum parallelism. Since A_{22} reduces in size as the computation progresses, a block cyclic data distribution is used to ensure that at any stage A_{22} is evenly distributed over all processes, thus obtaining their balanced load. Since the distribution of work becomes uneven as the computation progresses, a larger block size results in greater load imbalance, but reduces the frequency of communication between processes. There is, therefore, a tradeoff between load imbalance and communication startup cost which can be controlled by varying the blocking factor, **b**. The optimal values of the block size determined by the Heterogeneous ScaLAPACK library are in the range ($128 \leq \mathbf{b} \leq 256$). It uses the value of 128. The procedure

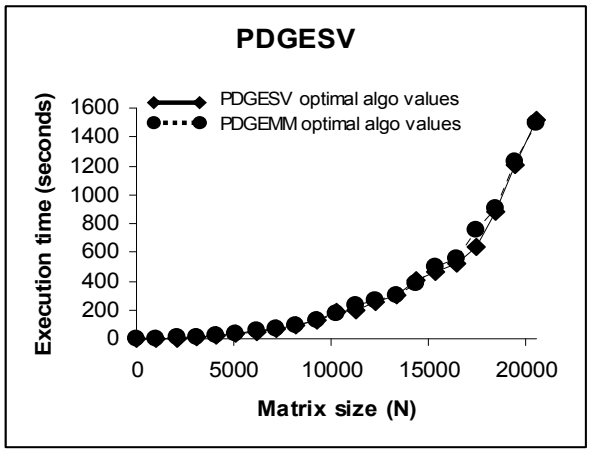
Size of the matrix (N)	Optimal 2D process grid arrangement (p,q)		Size of the matrix (N)	Optimal 2D process grid arrangement (p,q)	
	Heterogeneous ScaLAPACK (PDGESV)	Heterogeneous ScaLAPACK (PDPOSV)		Heterogeneous ScaLAPACK (PDGESV)	Heterogeneous ScaLAPACK (PDPOSV)
1024	(2,8)	(2,8)	11264	(2,12)	(2,8)
2048	(2,7)	(2,8)	12288	(3,8)	(2,7)
3072	(4,4)	(3,5)	13312	(4,6)	(2,7)
4096	(3,5)	(2,8)	14336	(2,12)	(2,6)
5120	(3,5)	(2,8)	15360	(3,7)	(3,4)
6144	(2,8)	(2,7)	16384	(3,7)	(2,11)
7168	(2,8)	(3,5)	17408	(3,7)	(3,7)
8192	(2,8)	(2,7)	18432	(3,7)	(3,7)
9216	(2,8)	(3,5)	19456	(3,7)	(3,7)
10240	(2,8)	(2,8)	20480	(2,12)	(3,7)

Table 1. Optimal 2D process grid arrangements (p,q) determined during the execution of the context creation routines.

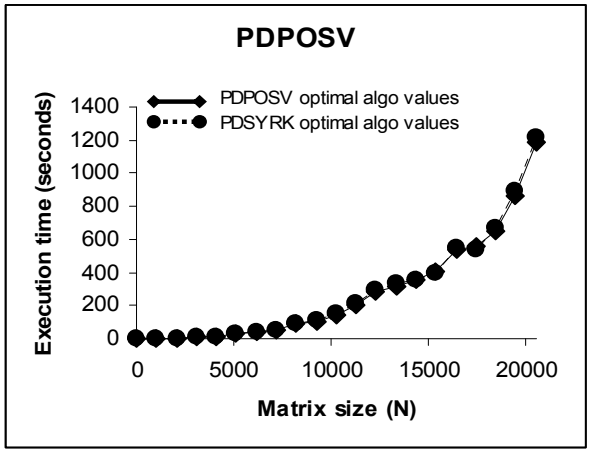
to determine the optimal block size is performed during the installation of the software.

Finally, the optimal values of the 2D grid arrangement of processes (**p**,**q**) address the load imbalance caused by the computational "hot spots" where certain processes have more work to do between synchronization points than others. This is the case in the LU decomposition routine due to partial pivoting being performed over rows in a single column of the process grid while the other processes are idle. Similarly, the evaluation of each block row of the U matrix requires the solution of a lower triangular system across processes in a single row of the process grid. There is thus an optimal 2D process grid arrangement, (**p**,**q**), which depends on the communications characteristics of the network and determines the overlap of the communication with the computation. During the creation of a HeteroMPI group of processes in the context creation routine, the function **`HMPI_Group_pauto_create`** estimates the time of execution of the algorithm for each process arrangement evaluated. For each such estimation, it invokes the runtime mapping algorithm, which tries to arrange the processes along a 2D grid so as to optimally load balance the work of the processors. It returns the process arrangement that results in the least estimated time of execution of the algorithm. Table 1 shows the optimal 2D process grid arrangements determined by the context creation routines during the execution of the Heterogeneous ScaLAPACK programs.

In Figure 3(a), the optimal values of the algorithmic parameters (blocking factor, 2D process arrangement, mapping of the processes to the computers of the heterogeneous network) determined for the execution of the PDGEMM routine are used in the execution of the PDGESV routine. To do this, we use the heterogeneous context created for the PDGEMM routine in the execution of the PDGESV routine. We have chosen the PDGEMM routine (used for the update of the trailing sub-matrix) because it is the most expensive routine in the execution of PDGESV. What we are trying to determine here is the impact of using the optimal mapping of processes for the PDGEMM routine during the execution of the PDGESV routine. Figure 3(b) shows the results for PDPOSV. From the figures, it can be concluded that the heterogeneous context created for the most expensive routine, PDGEMM, could be used in the execution of dense linear system solver, PDGESV, without causing any performance degradation. That is the optimal values determined for the algorithmic parameters to be used in the execution of one of the most expensive routines, PDGEMM, could be used in the execution of dense linear system solver, PDGESV. The conclusions apply to PDPOSV. This approach is used in ATLAS where the optimal values of the algorithmic parameters



(a)



(b)

Figure 3. (a) Optimal algorithmic parameter values for PDGESV and PDGEMM routines used in the execution of Heterogeneous ScaLAPACK program calling PDGESV routine. (b) Optimal algorithmic parameter values for PDPOSV and PDSYRK routines used in the execution of Heterogeneous ScaLAPACK program calling PDPOSV routine.

determined for matrix-matrix multiplication (GEMM) are used for the other level-3 routines as well.

4. Conclusions and Future Work

In this paper, we have presented the details of implementation of the parallel routines in the Heterogeneous ScaLAPACK library that solve a dense system of linear equations.

We show that the efficiency of these parallel routines is due to the most important feature of the library, which is the automation of the difficult optimization tasks of parallel programming on heterogeneous computing clusters. They are the determination of the accurate values of the platform parameters such as the speeds of the processors and the latencies and bandwidths of the communication links connecting different pairs of processors, the optimal values of the algorithmic parameters such as the 2D process grid arrangement and efficient mapping of the processes executing the parallel algorithm to the executing nodes of the heterogeneous computing cluster.

Our future work would address the least squares and eigenvalue problems. We would also present results on multicore platforms.

Acknowledgement

The research was supported by Science Foundation of Ireland (SFI). Pedro Alonso wishes to acknowledge the support provided by Vicerrectorado de Investigación, Desarrollo e Innovación de la Universidad Politécnica de Valencia, and Generalitat Valenciana.

References

[1] A. Kalinov and A. Lastovetsky, "Heterogeneous Distribution of Computations Solving Linear Algebra Problems on Networks of Heterogeneous Computers," Journal of Parallel and Distributed Computing, 61(4), pp.520-535, April 2001.

[2] 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), pp.1317-1343, November 2000.

[3] A. Lastovetsky, "Adaptive Parallel Computing on Heterogeneous Networks with mpC," Parallel Computing, 28(10), pp.1369-1407, October 2002.

[4] Netlib repository. http://www.netlib.org.

[5] O. Beaumont, V. Boudet, A. Petitet, F. Rastello, and Y. Robert, "A Proposal for a Heterogeneous Cluster ScaLAPACK (Dense Linear S8lvers)," IEEE Transactions on Computers, 50(10), pp.1052-1070, October 2001.

[6] Y. Kishimoto and S. Ichikawa, "An Execution-Time Estimation Model for Heterogeneous Clusters," In Proceedings of 18th International Parallel and Distributed Processing Symposium, IEEE Computer Society (2004).

[7] A. Kalinov and S. Klimov, "Optimal mapping of a parallel application processes onto heterogeneous platform," In Proceedings of 19th International Parallel and Distributed Processing Symposium, IEEE Computer Society (2005).

[8] J. Cuenca, D. Giménez, and J-P. Martinez, "Heuristics for work distribution of a homogeneous parallel dynamic programming scheme on heterogeneous systems," Parallel Computing, 31(7), pp.711-735, Elsevier, 2006.

[9] A. Lastovetsky, "Scientific Programming for Heterogeneous Systems - Bridging the Gap between Algorithms and Applications," Proceedings of the 5th International Symposium on Parallel Computing in Electrical Engineering, IEEE Computer Society (2006).

[10] Heterogeneous ScaLAPACK. http://hcl.ucd.ie/project/HeteroScaLAPACK/.

[11] A. Lastovetsky and R. Reddy, "HeteroMPI: Towards a Message-Passing Library for Heterogeneous Networks of Computers," Journal of Parallel and Distributed Computing (JPDC), 66(2), pp.197-220, Elsevier, 2006.