Modelling Parallel Linear Algebra Routines

Javier Cuenca[1], Luis-Pedro García[2], Domingo Giménez[3], José González[4], and Antonio Vidal[5]

[1] Departamento de Ingeniería y Tecnología de Computadores, Universidad de Murcia. Apartado 4021, 30001 Murcia, Spain
javiercm@ditec.um.es
[2] Servicio de Apoyo a la Investigación Tecnológica, Universidad Politécnica de Cartagena. Paseo Alfonso XIII, 30202 Cartagena, Murcia, Spain.
luisgarcia@sait.upct.es
[3] Departamento de Informática y Sistemas, Universidad de Murcia. Apartado 4021, 30001 Murcia, Spain
domingo@dif.um.es
WWW home page: dis.um.es/~domingo
[4] Centro de Investigación de Intel, Barcelona
javiercm@ditec.um.es
[5] Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia. Cno. Vera s/n, 46022 Valencia, Spain
avidal@dsic.upv.es

Abstract. This paper shows some ways to improve the modellization of parallel linear algebra routines and the use of the model to reduce the execution time. The proposed ideas include the consideration of algorithmic parameters in the model, the consideration of the parameters as a function of some system parameters, and the division of the theoretical execution time formula in different formulas for different problem sizes. The application of these ideas have been analyzed with several parallel linear algebra routines. In this paper, experiments with the QR and Cholesky factorizations are shown.

1 Introduction

The modellization of sequential or parallel linear algebra routines helps in the development of highly efficient code. This modellization can be done by simulations [14] or by analytic models, which can be used successfully for particular systems or routines when the problem size is big enough, but it is difficult to develop models useful for a wide range of systems, routines and problem sizes. There are some works in the development of accurate models of linear algebra routines [5, 8, 13, 16]. Using the models some decisions can be taken:

– In algorithms by blocks it is interesting to obtain a block size which provides execution times close to the optimum [4, 10, 16]. This decision could be included in the installation of the routines or libraries as LAPACK [2] or ScaLAPACK [3] to have paramaterized libraries [9].

– It is also interesting to select the number of processors and the topology to use in the solution of the problem [15]. The best number of processors and topology for a particular routine can vary from one system to other [6] or with the system workload [7].
– The solution of a linear algebra problem is carried out normally by calling to basic routines which can be taken from one library or another. The use of the best library in each one of the basic routines can also reduce the execution time [1].

In all the cases, to obtain a good selection it is necessary to have an accurate model of the execution time of the routine. The accurate model can be combined with a library development methodology to have adaptive libraries or a hierarchy of adaptive libraries [1, 6] which automatically adapt to a particular architecture or conditions in the system without intervention of the user.

In this paper we propose some modifications which can be included in the models to improve their accuracy, and show how the ideas are applied by using the QR and Cholesky factorizations. These ideas have been successfully applied to other routines [1].

2 Improvements in the model

The traditional basic model of a parallel linear algebra routine consists on the theoretical arithmetic cost plus the theoretical communication cost, with a parameter for the computation cost, other for the start-up time and another for the word-sending time. If the values of the parameters in the system are well estimated, the model predicts well the execution time when the problem size is big enough. In some cases it is necessary a more detailed formula. We propose some modifications in the traditional formulas and in the way they are used to predict the execution time and to obtain efficient routines:

– The value of the parameters in the formula can vary with the problem size. In that case it could be preferable to decompose the formula in different parts, and to obtain different values for the parameters in each part. For example, the arithmetic cost can be $\frac{2}{3}kn^3$, but it could be preferable to have the formula $\frac{2}{9}k_{\frac{n}{6}}n^3 + \frac{2}{9}k_{\frac{3n}{6}}n^3 + \frac{2}{9}k_{\frac{5n}{6}}n^3$, where $k_{\frac{in}{6}}$ represents the cost of an arithmetic operation whit problem size between $\frac{(i-1)n}{6}$ and $\frac{(i+1)n}{6}$.
– More than one algorithmic or communication parameter can be used:
  - When arithmetic operations of level two and three appears in the algorithm it is better to use different parameters for the operations of different levels.
  - It is also possible to use different parameters for different routines of the same level. For example, different cost can be obtained with `drot` than with `dscal`.
  - Also for the same routine with different data storage scheme the values can be different. The cost of multiplications and additions will be different when they are performed as part of a square matrix multiplication

than when the multiplication is with rectangular matrices, or when the matrices are triangular.
   - The way in which the data are accessed in the algorithm also influences the value of the parameters. When the data are accessed in an increasing order of columns a value different to that obtained when they are accessed by blocks can be obtained.
   - When different types of communications are used in the program, the cost of the communication parameters varies according to the communication: the start-up and the word-sending times will be different when using point to point communications than when using broadcast.

  All this must be had in account to estimate the values of the parameters, estimating values for different routines or levels of routines, different data storage schemes and data access.
- Because the values of the parameters can depend on a set of other parameters, as can be the problem size, the block size in algorithms by blocks, and the number or processors and the topology, the parameters representing the costs in the system can be represented as a function of these algorithmic parameters. In that way a multidimensional function is obtained for each parameter. The function will be represented by a table with a number of dimensions equal to the number of algorithmic parameters which significatively influence the value of the system parameter.

In all the cases the value (or the table representing the value function) must be obtained by experiments with the basic operations in the algorithm, but also using the data storage and access scheme used in the algorithm. Experiments can be performed for each routine, but what we propose is to identify the basic routines more widely used, together with the size of the parameters and the data storage and access schemes normally used in linear algebra algorithms, and to develop the basic libraries including an engine to estimate these values when they are installed. Thus, using the model of the execution time of the algorithm and the values of the parameters estimated when the basic libraries were installed, the selection of satisfactory values of the parameters can be made automatically without important additional cost, and the user can be provided with intelligent routines which will execute efficiently independently of the system where they are executed and the user executing them.

3 Experimental results

Some of the possible improvements in the models and their applications are analyzed with the QR and Cholesky factorizations.

3.1 Division of the model in different formulas

Two non block versions of the Cholesky factorization are considered [11]. The block versions of the factorization work better than these non block versions, but it is more difficult to model low level routines, and what we want is to show how more accurate modelizations can be obtained.

In a matrix-vector product version the cost is $\frac{1}{3}k_2n^3$, with k_2 corresponding to the BLAS 2 routine `dgemv`. With only this parameter in the model the prediction of the execution time was not satisfactory. To improve the model the computation has been divided into four parts to include the cost of the parameters depending on the size and form of the submatrices in which operations are applied. The model is:

$$\frac{5}{96}n^3k_{gemv_{\frac{7n}{8}\times\frac{n}{8}}} + \frac{11}{96}n^3k_{gemv_{\frac{5n}{8}\times\frac{3n}{8}}} + \frac{11}{96}n^3k_{gemv_{\frac{3n}{8}\times\frac{5n}{8}}} + \frac{5}{96}n^3k_{gemv_{\frac{n}{8}\times\frac{7n}{8}}} \quad (1)$$

where the operations have been grouped for vector sizes between 1 and $\frac{n}{4}$, $\frac{n}{4}+1$ and $\frac{n}{2}$, $\frac{n}{2}+1$ and $\frac{3n}{4}$, $\frac{3n}{4}+1$ and $n-1$.

In an outer product version, the operations of order n^3 are performed with the `daxpy` operation. The cost is divided into four parts:

$$\frac{5}{96}n^3k_{axpy_{\frac{7n}{8}}} + \frac{11}{96}n^3k_{axpy_{\frac{5n}{8}}} + \frac{11}{96}n^3k_{axpy_{\frac{3n}{8}}} + \frac{5}{96}n^3k_{axpy_{\frac{n}{8}}} \quad (2)$$

The values of the parameters are obtained by performing a set of matrix updating on vectors in consecutive columns in the matrix, like in the algorithm. If the values are obtained with updating in non consecutive vectors, the theoretical prediction is much worse. Thus, we see the order in which the data are accessed is also important to obtain a good estimation.

Experiments have been performed in a Pentium III and a Pentium 4, and with reference BLAS (BLASR) and an optimized BLAS for Pentium III (BLASIII) and Pentium 4 (BLAS4) [12].

The modelled and experimental times are compared in table 1. The way in which the values of the parameters in the model are obtained for each problem size produces a good estimation of the execution time and consequently a good selection of the best library and algorithm to use. The optimal decision is not always the obvious, but it is well taken using the model.

3.2 Selection of parameters

In an algorithm by blocks the block size and the library to be used must be decided, but the optimized library will be clearly the best. A "dot product" version by blocks of the Cholesky factorization has been studied [11]. If A and G are divided in blocks A_{11}, A_{21}, A_{21}^T, A_{22}, G_{11}, G_{21} and G_{22}; with block A_{11} of size $b \times b$, A_{21} of $(n-b)\times b$, and A_{22} of size $(n-b)\times(n-b)$, and G_{11} and G_{22} lower triangular, the computations are:

– Compute the Cholesky factorization in the diagonal blocks, using the level 2 routine of LAPACK **dpotf2**.
– Compute the factor for blocks in the lower triangular part, $G_{21} = A_{21}(G_{11}^T)^{-1}$, using the level 3 routine of BLAS **dtrsm**.

Table 1. Non block Cholesky factorization, in Pentium III and 4, and using different libraries and algorithms

	matrix-vector product				outer product			
	BLASR		BLASIII		BLASR		BLASIII	
n	mod.	expe.	mod.	expe.	mod.	expe.	mod.	expe.
PentiumIII								
256	0,117	0,058	**0,081**	**0,041**	0,141	0,061	0,129	0,058
512	0,826	0,702	**0,350**	**0,335**	1,099	1,021	1,135	1,041
1024	5,982	5,734	**2,660**	**2,615**	7,889	8,749	7,952	8,872
2048	46,135	45,450	**18,931**	**19,199**	61,817	71,544	64,784	76,184
Pentium4								
256	0,032	0,020	**0,026**	**0,014**	0,192	0,024	0,188	0,021
512	**0,179**	**0,154**	0,476	0,416	0,446	0,218	0,441	0,209
1024	**1,350**	**1,118**	4,690	3,960	2,244	1,646	2,167	1,596
2048	**9,551**	**8,586**	29,667	28,792	13,316	13,048	12,550	11,934

– Update blocks in the lower triangular part and in the diagonal, using the level 3 routines of BLAS **dgemm** and **dsyrk**, $\tilde{A}_{22} = A_{22} - G_{21}G_{21}^T = G_{22}G_{22}^T$.

The execution time is modelled:

$$2k_{3,gemm}\left(\frac{n^3}{6} - \frac{n^2 b}{2} + \frac{nb^2}{3}\right) + k_{3,trsm}\left(\frac{n^2 b}{2} - \frac{nb^2}{2}\right) + k_{3,syrk}\left(\frac{n^2}{2} - \frac{nb}{2}\right)(b+1) + k_{2,potf2}\frac{nb^2}{3} \qquad (3)$$

where the k_3 values depend on that of b, and they are practically constant when the matrix size varies.

Table 2 show the execution time predicted by the model (model), the lowest experimental time obtained by performing experiments with different block sizes (exp. low.) and the time when the value of b provided by the model is used (exp. mod.). The lowest experimental time and the time with the parameter provided by the model usually coincide. The model gives a good prediction for the execution time, leading to a good library and parameter selection.

3.3 Selection of parameters in parallel algorithms

For the parallel block QR factorization the costs are modelled by:

$$\frac{4}{3}k_{3,gemm}\frac{n^3}{p} + \frac{1}{4c}bk_{3,trmm}n^2 + \frac{1}{r}bk_{2,geqr2}n^2 + \frac{1}{2r}nk_{2,larft}n^2 \qquad (4)$$

and:

Table 2. Comparison of the execution time (in seconds) and the block size of the block Cholesky factorization using reference and optimized BLAS. In Pentium III and Pentium 4

	BLASR					BLASIII				
	model		exp. (low./mod.)		dev.	model		exp. (low./mod.)		dev.
n	time	blo.	time	blo.	(%)	time	blo.	time	blo.	(%)
PentiumIII										
256	0,04	32	0,03/0,03	64/32	60/20	0,02	64	**0,01**	64	30
512	0,12	32	0,11	32	0,2	0,12	64	**0,11**/0,12	32/64	24/8
1024	0,91	32	2,20	32	0,1	0,85	128	**0,82**	128	3,5
2048	17,39	32	17,32	32	0,4	5,97	128	**5,81**	128	2,7
Pentium4										
256	0,02	32	0,01/0,01	64/32	42/25	0,01	64	**0,01**	64	15
512	0,12	32	0,11	32	4	0,05	64	**0,05**	64	2
1024	0,91	32	0,92	32	2	0,31	128	**0,31**	128	0,4
2048	7,17	32	8,10	32	12	1,99	128	**2,01**	128	2

$$t_s \frac{n}{b}((2+3b)\log r + 2\log c) + t_w \left(\frac{n^2}{2} \left(\frac{2(r-1)}{r^2} + \frac{\log r}{c} + \frac{\log r}{r} \right) + nb \log p \right) \quad (5)$$

The matrix is distributed in a logical two-dimensional mesh (in ScaLAPACK style [3]) of $p = r \times c$ processors, with block size b.

Experiments have been carried out in a network of Pentium III with Fast-Ethernet. Three libraries have been used: ATLAS, a BLAS for Pentium II Xeon (BLASII), and a BLAS for Pentium III (BLASIII). The arithmetic parameters have been taken as function of the block size. In table 3 the execution time with the parameters provided by the model (mod.) and the lowest execution time (low.) are shown together with their parameters. The times with the model are always close to the lowest times, because the model makes a good selection of the parameters.

The values of the parameters predicted varies for different systems and problem sizes. The values predicted in an IBM-SP2, an Origin 2000, a network of SUN Ultra 1 with Ethernet (nSUN) and a network of Pentium III with Fast-Ethernet (nPIII) are shown in table 4. In all the cases the selection is satisfactory.

4 Conclusions and future works

We have shown some modifications to the traditional modellization of parallel linear algebra routines. The combination of the model with the experimental estimation of some systems parameters allows a better selection of algorithmic parameters, and the possibility of the designing of routines adaptable to systems with different characteristics. Some experiments with Cholesky and QR factorizations have been shown. Experiments with other routines confirm those here

Table 3. Comparison of the execution time of the QR factorization. In a network of 8 Pentium III with Fast-Ethernet

	size:	128	256	384	1024	2048	3072
	time	0.025	0.087	0.178	1.92	9.90	28.00
mod.	processors	1×2	1×8	1×8	1×8	1×8	1×8
	block size	8/16	8	8	16	16	16
	library	BLASII	BLASII	BLASII	BLASII	BLASII	BLASII
	time	0.025	0.086	0.176	1.92	9.40	25.11
low.	processors	1×2	1×8	1×8	1×8	1×8	1×8
	block size	16	8	8	16	16	16
	library	BLASII	BLASIII	BLASIII	BLASII	BLASII	BLASII

Table 4. Parameters selection for the QR algorithm, in different systems

	IBM SP2		Origin 2000		nSUN	nPIII	
	p=4	p=8	p=4	p=8	p=4	p=4	p=8
size	b r c	b r c	b r c	b r c	b r c	b r c	b r c
1024	16 1 4	16 1 8	32 4 1	32 4 2	16 1 4	16 1 4	16 1 8
2048	32 1 4	16 1 8	64 4 1	32 4 2	16 1 4	16 1 4	16 1 8
3072	32 1 4	32 2 4		32 4 2		32 1 4	16 1 8
4096	32 1 4	32 2 4		64 4 2		32 1 4	16 1 8

shown [1]. It is also possible to use the models in systems with variable workload, influencing the parameters with some measure of the system load taken at execution time [7].

At present we are working in the inclusion in parallel linear algebra libraries of an engine for the automatic obtention of the parameters. This engine would use the model of the routine to obtain appropriate values of the parameters. The main idea is to obtain libraries automatically adaptable to different systems, and which could be used in a blackbox way.

Acknowledgements

This work has been funded in part by Comisión Interministerial de Ciencia y Tecnología, project number TIC2000/1683-C03, and Fundación Séneca, project number PI-34/00788/FS/01.

This work has been developed using the systems of the CEPBA (European Center for Parallelism of Barcelona), the CESCA (Centre de Supercomputació de Catalunya) and the Innovative Computing Laboratory at the University of Tennessee.

References

1. Alberti, A., Alonso, A., Vidal, A., Cuenca, J., García. L. P., Giménez, D.: Designing polylibraries to speed up linear algebra computations, TR LSI 1-2003, Departamento de Lenguajes y Sistemas Informáticos, Universidad de Murcia, 2003. `dis.um.es/~domingo/investigacion.html`.
2. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, SIAM (1999).
3. Blackford, L. S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R. C.: ScaLAPACK Users' Guide, SIAM (1997).
4. Jaeyoung Choi: PoLAPACK: Parallel Factorization Routines with Automatic Blocking. Concurrency Comput. Pract. Exp., Vol 13, n 12 (2001) 1033-47.
5. Čiegis, R., Starikovičius, V., Waśniewski, J.: On the Efficiency of Scheduling Algorithms for Parallel Gaussian Elimination with Communication Delays. In Applied Parallel Computing, New Paradigms for HPC in Industry and Academia, 5th International Workshop, PARA 2000, Eds. Sørevik, T., Manne, F., Moe, R., Gebremedhin, A. H. Lecture Notes in Computer Science, Vol 1947 (2000).
6. Cuenca, J., Giménez, D. González, J.: Architecture of an Automatic Tuned Linear Algebra Library. Parallel Matrix Algorithms and Applications (1997).
7. Cuenca, J., Giménez, D., González, J., Dongarra, J., Roche, K.: Automatic Optimisation of Parallel Linear Algebra Routines in Systems with Variable Load. In 11th EUROMICRO Workshop on Parallel, Distributed and Networked Processing PDP 2003, IEEE (2003) 409-416.
8. Dackland, K., Kågström, B.: A Hierarchical Approach for Performance Analysis of ScaLAPACK-based Routines Using the Distributed Linear Algebra Machine, 3th International Workshop, PARA 1996. In Applied Parallel Computing, Industrial Computation and Optimization, Eds. Wasnieski, J., Dongarra, J., Madsen, K., Olesen, D. Lecture Notes in Computer Science, Vol 1184 (1996).
9. Dongarra, J.: Self Adapting Numerical Software (SANS) for Grid Computing. Innovative Computing Laboratory, University of Tennessee (2002).
10. Gansterer, W. N., Kvasnicka, D. F., Ueberhuber, C. W.: Blocking Techniques in Numerical Software. In Parallel Computation, 4th International ACPC Conference, Eds. Zinterhof, P., Vajteršic, M., Uhl, A. Lecture Notes in Computer Science, Vol 1557 (1999) 127-139.
11. Golub, G. H., Van Loan, C. F.: Matrix Computations, The Johns Hopkins University Press (1989).
12. Intel Math Kernel Library, `http://developer.intel.com/software/products/mkl/mkl52/index.htm`.
13. Kothari, S. C., Mitra, S.: Analysis of Computational Balance of Parallel Numerical Algorithms. Proceedings of the 8th SIAM Parallel Processing for Sientific Computing Conference (1997).
14. Kvasnicka, D. F., Ueberhuber, C. W.: Developing Architecture Adaptive Algorithms using Simulation with MISS-PVM for Performance Prediction. 11th ACM Conference on Supercomputing (1997).
15. Luecke, G. R., Coyle, J. J.: High Performance Fortran versus explicit message passing on the IBM SP-2 for the parallel LU, QR, and Cholesky factorizations. Supercomputer, **68** XIII-2 (1997) 4-14.
16. Yunquan Zhang: Block Size Selection of Parallel LU Factorization. Proceedings HPC-ASIA2000, IEEE Computer Society, Vol. 1 (2000) 247-249.