Using Experimental Data to Improve the Performance Modelling of Parallel Linear Algebra Routines

Luis-Pedro García[1], Javier Cuenca[2] and Domingo Giménez[3]

[1] Universidad Politécnica de Cartagena, Spain, luis.garcia@sait.upct.es
[2]Universidad de Murcia, Spain, javiercm@ditec.um.es
[3]Universidad de Murcia, Spain, domingo@dif.um.es

Summary

The performance of parallel linear algebra routines can be improved automatically using different methods. Our technique is based on the modellisation of the execution time of each routine, using information generated by routines from lower levels. However, sometimes the information generated at one level is not accurate enough to be used satisfactorily at higher levels. Therefore, a remodelling of the routines is performed by using (applied appropriately) polynomial regression. A remodelling phase is proposed, and analysed with a parallel matrix multiplication.

Introduction

In this work, a technique for redesigning the model for linear algebra programs from a series of sampling points by means of polynomial regression has been included. The basic idea is to exploit the natural hierarchy existing in linear algebra programs and to start from the hierarchical model using the information from lower level routines to model the higher level ones without experimenting with the latter. However, if for a concrete routine all this information is not useful enough, then its model, and/or the models of its set of lower level routines, would be rebuilt again from the beginning, using a series of executions and appropriately applied polynomial regression.

Self-Optimised Linear Algebra Routine ($SOLAR$)

The design, installation and execution of a $SOLAR$ is described step by step, following the scheme in figure 1.

Design

The tasks to be done by the designer of the Linear Algebra Routine (LAR) are:
Create the LAR: The LAR is designed and programmed if it is new. Otherwise, the code of the LAR does not have to be changed.
Model the LAR theoretically: The complexity of the LAR is studied, obtaining an analytical model of its execution time as a function of the problem size, n, the System Parameters (SP) and the Algorithmic Parameters (AP): $T_{exec} = f(SP, AP, n)$.
SP: system parameters (cost of arithmetic or communication operations). AP: algorithmic parameters (block size, number of processors, logical topology).
Select the parameters for testing the model: The most significant values of n and AP are written in the structure $Model\_Testing\_Values$ by the LAR designer.
Create the $SOLAR\_manager$: This subroutine is the engine of the $SOLAR$. It is in charge of managing all the information inside this software architecture.

Installation

The tasks performed by the $SOLAR\_manager$ are:
Execute the LAR: It executes the LAR using the different values of n and AP contained in the structure $Model\_Testing\_Values$.
Test the model: Compares execution times with the theoretical times provided by the model for the same values of n and AP. If the average distance between theoretical and experimental times exceeds an error quota, the $SOLAR\_manager$ would remodel the LAR and/or its set of lower level $LARs$.
The new model of a LAR is built from the original model by designing a polynomial scheme for the different combinations of n and AP in the structure $Model\_Testing\_Values$. The coefficients could be calculated using the following methods:
FI-ME: The experimental time function is approximated by a single polynomial. The coefficients are obtained with the minimum number of executions needed.
VA-ME: The experimental time function is approximated by a set of p polynomials corresponding to p intervals of combinations of (n, AP). For each of these intervals the above method is applied.
FI-LS: In this method the experimental time function is approximated by a single polynomial. Now a least squares method is applied to obtain the coefficients.
VA-LS: As in the VA-ME method, the execution time function is divided in p intervals, and the FI-LS method is applied in each. When one of them provides enough accuracy, then the other methods are discarded.

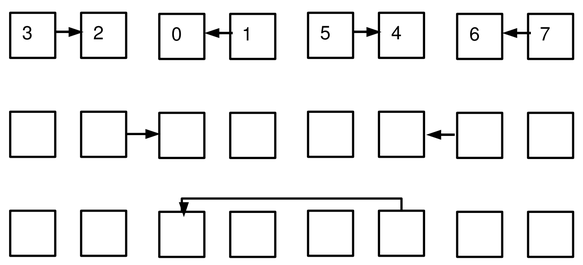

Figure 1: Life-cycle of a $SOLAR$.

Execution

When a $SOLAR$ is called to solve a problem of size n_R, the following tasks are carried out:
Collecting system information: The $SOLAR\_manager$ collects the information that the NWS (or any other similar tool installed on the platform) provides about the current state of the system.
Tuning the model: The $SOLAR\_manager$ tunes the $Tested\_Model$ according to the system conditions.
Selection of the $Optimum\_AP$: Using the $Tuned\_Model$, with $n = n_R$, the $SOLAR\_manager$ looks for the combination of AP values that provides the lowest execution time.
Execution of the LAR: Finally, the $SOLAR\_manager$ calls the LAR with the parameters $(n_R, Optimum\_AP)$.

Self-Optimised Parallel Linear Algebra Routine Samples

Parallel Matrix-Matrix Multiplication

In order to build the analytical model of the execution time, we identify different parts of the algorithm with different basic computation and communication routines:

- A routine that uses point-to-point communications distributes the blocks of matrix A onto $p-1$ processes.
- A broadcast communication routine is used to distribute the matrix B.
- The BLAS3 function **DGEMM** is used to compute the p matrix multiplication of size $\frac{n}{p} \times n \times n$.

Thus, the execution time for this algorithm can be modelled by the formula:

$$T(n,p) = t_{mult}\left(\frac{n^3}{p}\right) + (p-1)t_{Send}\left(\frac{n^2}{p}\right) + t_{Bcast}\left(n^2\right) \quad (1)$$

Experimental Results The experiments have been performed for two different systems: two Intel Itanium 2 systems connected by GigabitEthernet. where each node is equipped with four sets of dual-core 1.4 GHz Montecito processor (Rosebud); and two Intel Xeon systems connected by GigabitEthernet, each node is equipped with two sets of dual-core 3.0 GHz 5050 processors (Sol).
Modelling the computations. In the experiments for Rosebud and Sol we found that for the BLAS3 **DGEMM** routine the third order polynomial function was the best and good results were obtained using the FI-LS method. The values of the coefficients are obtained by performing a set of matrix multiplications of rectangular size, like in the algorithm. If the values are obtained with square matrices, the theoretical prediction is much worse. In addition, since Rosebud is a system with dual-core processors and shared memory access, it was necessary to perform several simultaneous executions of the routine and to use an average value for the coefficients of the polynomial that models **DGEMM**.
Modelling the communications. Initially the $SOLAR$ could use a linear model for the communications obtained from a ping-pong between two processes; but the start-up, t_s, and word-sending, t_w, times will be different when using a point to point communications than when using a broadcast. Since the values of t_s and t_w depend on the number of processes, a polynomial function was obtained for each case.

For the routine that distributes the blocks of matrix A, a model for the routine was obtained and not only for the MPI_Send. In Rosebud and for the broadcast routine MPI_Bcast, when the number of process p was greater than twelve, it was necessary to use the VA-LS method. The next table compares the execution time provide by the model (mod.) and the experimental time (exp.) for the MPI_Bcast routine when using FI-LS method and when using the VA-LS method, along with the deviation (dev.) with $p = 16$ processes.

n	512	1536	2048	3072	4096	5120	6144	6656
exp.	0,0528	0,4603	0,8157	2,0998	3,4852	5,4764	8,0366	9,0876
FI-LS	0,0495	0,4490	0,7986	1,7973	3,1956	4,9933	7,1905	8,4390
dev. (%)	6,13	2,46	2,10	14,40	8,31	8,82	10,53	7,14
VA-LS	0,0507	0,4605	0,8277	1,9823	3,5369	5,5356	7,9785	9,3666
dev. (%)	3,96	0,04	1,47	5,59	1,48	1,08	0,72	3,07

Discussion of experimental results.
The number in the next table represents the preferential order in which the values of the parameter p are selected by the model (mod.), and the order obtained from the experiments (exp.), on Rosebud. A value 0 in the deviation (dev.) means the value of p selected by our method coincide with the obtained experimentally. In the cases where the selection of the value of p does not coincide with the experimental one, the deviation in the execution time is very low and the mean of the deviations is 0.38%.

	$p=4$			$p=8$			$p=12$			$p=16$		
n	exp.	mod.	dev. (%)	exp.	mod.	dev. (%)	exp.	mod.	dev. (%)	exp.	mod.	dev. (%)
1024	2sd	1st	2.49	1st	2nd	2.49	3rd	3rd	0	4th	4th	0
2048	2sd	2nd	0	1st	1st	0	3rd	3rd	0	4th	4th	0
2560	2sd	2nd	0	1st	1st	0	3rd	3rd	0	4th	4th	0
4096	4th	4th	0	1st	1st	0	2nd	3rd	0.37	3rd	2nd	0.37
5120	4th	4th	0	1st	1st	0	3rd	3rd	0	2nd	2nd	0
5632	4th	4th	0	1st	1st	0	3rd	3rd	0	2nd	2nd	0
6144	4th	4th	0	1st	1st	0	3rd	3rd	0	2nd	2nd	0
6656	4th	4th	0	1st	1st	0	3rd	3rd	0	2nd	2nd	0

The next table shows the experimental execution time, the theoretical execution time, and the deviation between both for different problem sizes and number of processes. From the table it is seen that the model successfully predicts the number of processes with which the optimum execution times are obtained, except for matrix size 1024. In this case the execution time obtained with the values provided by the model is about 5.41 % higher than the optimum experimental time.

n	1024	2048	2560	3072	3584	4096
$p=2$						
exp.	0.413	2.695	4.913	8.254	12.694	18.826
the.	0.444	2.760	5.081	8.424	12.972	18.911
dev. (%)	7.61	2.40	3.42	2.05	2.19	0.45
$p=4$						
exp.	**0.351**	1.945	3.359	5.449	8.136	11.783
the.	0.338	1.847	3.270	5.263	7.917	11.325
dev. (%)	3.58	5.04	2.63	3.41	2.68	3.89
$p=6$						
exp.	0.367	1.870	2.997	5.037	6.822	9.488
the.	0.362	1.791	3.055	4.775	7.000	9.796
dev. (%)	1.44	4.24	1.94	5.22	2.61	3.25
$p=8$						
exp.	0.370	**1.740**	**2.750**	**4.436**	**6.166**	**9.033**
the.	**0.312**	**1.507**	**2.549**	**3.949**	**5.753**	**8.008**
dev. (%)	15.68	13.38	7.32	10.97	6.70	11.36

Remodelling parallel matrix multiply. In the methodology proposed, the $SOLAR\_manager$ could build a new model for a LAR. The method consists of defining a new polynomial function for the different combinations of n and AP. For this algorithm the polynomial function used is:

$$T(n,p) = a_{3,1}n^3p + a_{3,0}n^3 + a_{3,-1}\frac{n^3}{p} + a_{2,1}n^2p + a_{2,0}n^2 + a_{2,-1}\frac{n^2}{p} + a_{1,1}np + a_{1,0}n + a_{1,-1}\frac{n}{p} + a_{0,1}p + a_{0,0} + a_{0,-1}\frac{1}{p} \quad (2)$$

and twelve coefficients must be calculated with any of the methods indicated, good results were obtained with the FI-LS method.
The next table shows the execution time provided by the model (mod.) and the experimental time (exp.) for different problem sizes and number of processes, along with the deviation (dev.). The relative error ranges from 0.08 % to 10.30 %. This means that the new $Theoretical\_Model$ could be the $Tested\_Model$.

n	1024	2048	3584	4608	5632	6656
$p=2$						
exp.	0.238	1.794	9.012	18.824	34.014	55.781
the.	0.235	1.752	8.955	18.695	33.710	55.133
dev. (%)	0.91	2.35	0.64	0.68	0.89	1.16
$p=4$						
exp.	0.167	1.147	5.181	10.497	18.640	30.135
the.	0.164	1.118	5.237	10.490	18.316	29.205
dev. (%)	1.98	2.55	1.09	0.08	1.74	3.09
$p=6$						
exp.	**0.159**	1.109	4.227	8.202	14.145	22.453
the.	**0.153**	0.978	4.117	7.935	13.497	21.117
dev. (%)	4.19	11.86	2.60	3.26	4.58	5.95
$p=8$						
exp.	0.165	**1.066**	**4.029**	**7.559**	**12.517**	**19.499**
the.	0.157	**0.962**	**3.647**	**6.793**	**11.322**	**17.490**
dev. (%)	5.00	9.71	9.48	10.14	9.54	10.30

Conclusions

The values of the algorithmic parameters vary for different systems and problem sizes, but with the model and with the inclusion of the possibility of remodelling, a satisfactory selection of the parameters is made in all the cases, enabling us to take the appropriate decisions about their values prior to the execution. Thus, we can conclude that the methodology proposed can be used to obtain execution times close to the optimum without user intervention.

Acknowledgments. This work partially has been supported by the Consejería de Educación de la Región de Murcia, Fundación Séneca 02973/PI/05.