Rodriguez, G.; Badia, R.M.; Labarta, J. Generation of simple analytical models for message passing applications. A: International European Conference on Parallel Processing. "Euro-Par 2004: Parallel Processing: 10th International Euro-Par Conference: Pisa, Italy, August 31- September 3, 2004: proceedings". Springer, 2004, p. 183-188. ISBN 3-540-22924-8. The final authenticated version is available online at https://doi.org/10.1007/978-3-540-27866-5_23.

Generation of Simple Analytical Models for Message Passing Applications*

German Rodriguez, Rosa M. Badia, and Jesús Labarta

CEPBA-IBM Research Institute,
Technical University of Catalonia (UPC),
Campus Nord, Mòdul D6, Jordi Girona, 1-3, 08034 Barcelona, Spain

{grodrigu@ac.upc.es}, {rosab,jesus}@cepba.upc.es

Abstract. We present a methodology which allows to derive accurate and simple models which are able to describe the performance of parallel applications without looking at the source code. A trace is obtained and linear models are derived by fitting the outcome of a set of simulations varying the influential parameters, such as: processor speed, network latency or bandwidth.
The simplicity of the linear models allows for natural derivation of interpretations for the corresponding factors of the model, allowing for both prediction accuracy and interpretability to be maintained.
We explain how we plan to extend this approach to extrapolate from these models to be apply it to predict for processor counts different to the one of the given traces.

1 Motivation and Goal

Obtaining performance models of parallel applications is extremely useful for a broad range of purposes: determining if a platform achieves its expected performance, identification of the source of performance problems[16], scheduling, and many others.

Models based on simulations are used to explore the parameter space design, but are time consuming and lack the abstraction and possibility of interpretation that analytic models provide [8–10].

In this work we want present, extend and validate a methodology that we are developing [18] to extend the analysis capabilities of the tools DIMEMAS[1] and PARAVER[2] developed at CEPBA.

We want to derive simple analytical models including the number of processors as a parameter starting from post-mortem trace-files of the application under study. The simplest model presents non-linearities and we show how we treat them being methodological while gaining insight of how the application is affected by the architectural parameters.

* This work has been partially funded by the Ministry of Science and Technology of Spain under CICYT TIC2001-0995-CO2-01 and by a Ministry of Education of Spain *FPU* grant.

2 Methodology

We apply the following methodology: for each application that we want to analyze, we obtain traces from its real execution varying the number of processors. The dynamic instrumentation technology of MPIDtrace[11] (based on DPCL[12]) allows us to obtain traces of production binaries without needing access to its source code. This traces contain information of communication requests of all processes as well as the CPU demands between those requests. The usefulness of MPIDtrace has been shown in [13–15]. Other previously proposed methods needed the source code and knowledge of the program structure to be able to instrument the application [17].

This traces feed the DIMEMAS[1] simulator. DIMEMAS implements a simple abstract model of a parallel platform in which the MPI software layers are taken into account. It allows to simulate for varying Latencies, Bandwidths, and CPU ratios (that models the CPU performance ratio between the source and target machine). Given a trace for a particular number of processors, DIMEMAS is not able to perform a simulation of the trace under a different processor count.

STORM[3] performs DIMEMAS simulations of a particular trace file, randomly varying the influential architectural parameters.

From these results, we fit a linear model from these data of the elapsed time of the application against the architectural parameters. The coefficients characterize the application.

2.1 Brief Review of Previous Results and Methodology

We have already addressed the problem of finding an accurate model, simple and at the same time general enough, to explain and predict the behavior of the same set of programs [18], using a formula like: $T_{\mathrm{P}}(L, BW) = \alpha_{\mathrm{P}} + \beta_{\mathrm{P}} \cdot L + \frac{\gamma_{\mathrm{P}}}{BW}$[1].

We obtained the α_{P}, β_{P}, γ_{P} through linear fitting of DIMEMAS simulations varying L and BW. P was not a parameter.

A further step was the unification of these formulas to a single one, including the processor count P as one additional parameter in the form of:

$$T(P, L, BW) = \alpha(P) + \beta(P) \cdot L + \frac{\gamma(P)}{BW} \tag{1}$$

$\alpha(P)$ represents the time of the critical path, $\beta(P)$ the number of non-overlapped latencies and $\gamma(P)$ the number of bytes exchanged between processors not overlapped with computation. By knowing these functions of P we would be able to predict for a different processor count of that of the original traces.

It would be simple to model $\alpha(P)$ as: $\alpha(P) = \alpha_0 + \frac{\alpha_1}{P}$ (following Amdahl's law). However, in a parallel program, this is not necessarily true. We observe a non-linearity on P for the "critical path"($\alpha(P)$) when the distribution of computation improves locality and therefore results in a better memory hierarchy performance.

[1] L stands for Latency, BW, Bandwidth, and P for the number of processors.

2.2 Decoupling Non-linearities

Such a non-linearity is found in the $\alpha(P)$ parameter for the NAS BT benchmark. Locality plays and important role, and executions for increasing processor counts account for smaller total amounts of computation[2], as would be expected following Amdahl's law.

We use the term *Locality Factor* to quantitatively express the difference between the expected amount of work using Amdahl's law, and the real one. This *Locality Factor* is computed as the relation between the total CPU time for all processors of the trace for the minimum processor count (*reference* P) and the *actual* amount of time accounted for a particular processor count P: $LocalityFactor(P) = \frac{TotalCPUtime(P)}{TotalCPUtime(P_{\text{reference}})}$.

The *Locality factors* of the NAS BT executions do not vary linearly with the number of processors. Amdahl's law would predict a constant behavior (Locality factor being always 1). However the BT total sequential execution times decrease as much as 55% from 4 to 49 processors (non-linearly).

In order to apply our methodology, this non-linear effect should be decoupled, and therefore we propose the following formula:

$$T(P, L, BW) = (\alpha'(P)) \cdot LocalityFactor(P) + \beta'(P) \cdot L + \frac{\gamma'(P)}{BW}, \qquad (2)$$

Where α', β' and γ' are obtained having previously corrected the effect of increased locality. We simply scale all the CPU bursts of the simulations by setting the DIMEMAS CPU efficiency scaling parameter, `CPU_ratio`, to the value of the *Locality factor*. The same methodology explained in the two first paragraphs of Subsect. 2.1 is applied to the trace of the execution of the application for *each* processor count P with the corresponding CPU efficiency scaling factor applied in the simulation to obtain the α'_{P}.

Given these α'_{P} we then apply a linear regression (this time with no non-linearities involved) to obtain an estimate of $\alpha'(P)$ of the form:

$$\alpha'(P) = \alpha'_0 + \frac{\alpha'_1}{P} \qquad (3)$$

We checked (Sect. 3) that β'_{P} and γ'_{P} are statistically identical to β_{P} and γ_{P} and independent of the *Locality factor* for a sensible range of correcting factors.

We expect to find linear behaviors in $\beta(P)$ and $\gamma(P)$, as they roughly represent the non–overlapped latencies and communication bandwidth (see [18]).

The model obtained predicts with very high accuracy the DIMEMAS simulations obtained with the Locality Factor correction. However, this model is not predicting the real application, but an ideal one where the memory effects are not taken into account. To model the real application, the corresponding $\alpha'(P)$ parameter should be again multiplied by the *Locality factor* corresponding to the processor count to obtain what would be the real $\alpha(P)$ (critical path's time). We checked this approach in Sect. 3 and we show that for the applications considered, this process does not change the communication pattern.

[2] in DIMEMAS terms, *Total CPU time*: sum of all CPU bursts of all processors.

3 Example of Methodology use

To validate this methodology we have used IBM SP2 traces, from 4 to 49 processors, of the following applications: NAS BT[4], Sweep3D[6], RNAfold[5] and POP[7]. We show how we applied the process and show the relative errors obtained.

The last column of Table 1 shows the α_{P} factors obtained through regression of the set of simulations with no Locality Factor correction; the second column shows the $\alpha'(P)$ parameters[3] obtained using (4), in which its coefficients where obtained by fitting the data obtained of a set of simulations using the corresponding *Locality Factor* (shown in the third column), and therefore decoupling the non-linearity. Finally, the correctness of the model is validated by reconstruction of the α_{P} parameter as the product $\alpha'(P) \cdot LocalityFactor(P)$, shown in the fourth column. In this table, the $\alpha'(P)$ factors are calculated as:

$$\alpha'(P) = \alpha'_0 + \frac{\alpha'_1}{P} \tag{4}$$

where α'_0 and α'_1 were calculated by doing a linear regression on the α'_{P} obtained having previously applied the *Locality Factor* correction. The α'_{P} followed a linear behavior on the inverse of the number of processors, whereas the original α_{P}, with non-corrected Locality Factors, did not.

Table 1. Going back to the α_{P} factors from the linear model built using the α'_{P} parameters.

NAS BT				
Procs.	$\alpha'(P)$	$Loc.Factor(P)$	$\alpha'(P) \cdot Loc.Factor(P)$	α_{P}
4	14.748	1.000	14.748	14.718
9	6.589	0.986	6.496	6.585
16	3.734	0.900	3.361	3.351
25	2.412	0.672	1.620	1.598
36	1.694	0.575	0.974	0.964
49	1.261	0.553	0.697	0.699

In Table 1 we show that we could eliminate the non-linear behavior of the α_{P} parameter through a simple transformation, and then go back to the original one using a function $LocalityFactor(P)$ for which we understand its meaning and could derive a model. But we also have to show that this transformation does not affect the other factors: β_{P} and γ_{P}.

Table 2 shows that the β_{P} parameters are independent from the scaling of the CPU bursts according to the $LocalityFactor(P)$.

Given these facts, it was possible to use the $\alpha'(P)$, $\beta'(P)$ and $\gamma'(P)$ to go back to (2), recovering what the $\alpha(P)$, $\beta(P)$ and $\gamma(P)$ parameters would be and validate the model and the explained transformations. Table 3 shows the values obtained for the linear models used to fit the three factors.

[3] i.e., the $\hat{\alpha}'(P)$ parameters.

Table 2. β_P and γ_P factors against β'_P and γ'_P factors

NAS BT				
# P	β_P	β'_P	γ_P	γ'_P
9	320.37	321.96	0.46	0.44
25	527.92	521.80	0.50	0.55
36	675.39	677.55	0.27	0.23

Table 3. Parameters: N.S. means "Non Significant" (regression analysis)

Benchmark	α'_0	α'_1	β'_0	β'_1	γ'_0	γ'_1
NAS	0.062	58.743	200.814	12.240	0.453	N.S.
POP	1.366	337.444	14478.395	N.S.	2.739	N.S.
SWEEP	2.258	105.007	1850.302	21.488	N.S.	0.569
RNAfold	0.411	91.778	3528.737	N.S.	0.202	N.S.

We validated our results with more than 200 randomly selected bandwidths and latencies for each of the traces for varying P. In Table 4 we show the Maximum Relative Errors using (2) (where the parameters were obtained simulating with the corrected *Locality Factors*). These relative errors compare the prediction of model (2) and the actual times calculated by DIMEMAS.

Table 4. Maximum Relative Errors of validation against equation 2.

Processors	NAS	POP	SWEEP	RNAfold
4	0.24%	0.35%	—	—
8	—	—	0.47%	0.05%
9	1.39%	0.93%	—	—
12	—	—	—	0.43%
16	0.27%	2.48%	1.15%	—
25	1.37%	2.83%	—	—
28	—	—	—	0.88%
32	—	—	3.50%	—
36	1.13%	9.09%	—	—
49	0.28%	—	5.16%	—

4 Conclusions and Future Work

We have analyzed the possibility of easily deriving simple models that accurately characterize the behavior of a set of representative parallel benchmarks, from which one of them is a real application: POP[7].

The methodology under development allows to understand the underlying factors that influence the performance of a program. We analyzed the meaning of the parameters in a previous work [18] and are extending that analysis.

We have overcome the difficulties imposed by non-linearities and we are now testing this methodology to extrapolate the results to any number of processors.

We plan to extrapolate the $\alpha(P)$ model using similar techniques to the ones in [15] (PMACS), as well as determining the regions of linearity in a more or less automatic way.

References

1. DIMEMAS: http://www.cepba.upc.es/dimemas/
2. PARAVER: http://www.cepba.upc.es/paraver/
3. STORM: a tool for stochastic analysis, http://www.easi.de/storm
4. Bailey, D., Harris, T., Saphir, W., van der Wijngaart, R., Woo, A., Yarrow, M.: "The NAS Parallel Benchmarks 2.0". The International Journal of Supercomputer Applications, 1995.
5. Hofacker, I.L., Fontana, W., Bonhoeffer, L. S., Tacker, M., Schuster, P.: "Vienna RNA Package", http://www.tbi.univie.ac.at/ ivo/RNA, October 2002.
6. "The ASCI sweep3d Benchmark Code", http://www.llnl.gov/asci_benchmarks/asci/limited/sweep3d/asci_sweep3d.html
7. "Parallel Ocean Program": http://climate.lanl.gov/Models/POP/
8. Culler, D., Karp, R., Patterson, D., Sahay, A., Schauser, K. E., Santos, E., von Eicken, T.: "LogP: Towards a Realistic Model of Parallel Computation". Proc. of the 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, May 1993.
9. Mathis, M. M., Kerbyson, D. J., Hoisie, A.: "A Performance Model of non-Deterministic Particle Transport on Large-Scale Systems". Proc. Int. Conf. on Computational Science (ICCS), Melbourne, Australia, Jun 2003.
10. Jacquet, A., Janot, V., Leung, C., Gao, G. R., Govindarajan, R., Sterling, T. L.: "An Executable Analytical Performance Evaluation Approach for Early Performance Prediction". Proc. of IPDPS 2003.
11. MPIDtrace manual, http://www.cepba.upc.es/dimemas/manual_i.htm
12. DeRose, L.: "The dynamic probe class library: an infrastructure for developing instrumentation for performance tools". International Parallel and Distributed Processing Symposium, April 2001.
13. Girona, S., Labarta, J.: "Sensitivity of Performance Prediction of Message Passing Programs". International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), Monte Carlo Resort, Las Vegas, Nevada, USA, July 1999.
14. Girona, S., Labarta, J., Badia, R. M.: "Validation of Dimemas communication model for MPI collective operations". EuroPVM/MPI'2000, Balatonfüred, Lake Balaton,Hungary, September 2000.
15. Snavely, A., Carrington, L., Wolter, N., Labarta, J., Badia, R. M., Purkayastha, A.: "A framework for performance modeling and prediction". SC 2002.
16. Crovella, M. E., LeBlanc, J. L.: "The Search for Lost Cycles: A New Approach to Parallel Program Performance Evaluation (1993)". Tech. Rep. 479, Computer Science Department, University of Rochester, Dec., 1993.
17. Mehra, P., Schulbach, C., Yan, J.C.: "A comparison of two model-based performance-prediction techniques for message-passing parallel programs". Sigmetrics'94, pgs. 181-190, May 1994.
18. Badia, R. M., Rodriguez G., Labarta, J.: "Deriving analytical models from a limited number of runs". ParCo 2003 Proceedings