Performance Prediction of an NAS Benchmark Program with ChronosMix Environment

Julien Bourgeois and François Spies

LIFC,
Université de Franche-Comté,
16, Route de Gray,
25030 BESANCON Cedex,
FRANCE
{bourgeoi,spies}@lifc.univ-fcomte.fr,
http://lifc.univ-fcomte.fr/

Abstract. The Networks of Workstations (NoW) are becoming real distributed execution platforms for scientific applications. Nevertheless, the heterogeneity of these platforms makes complex the design and the optimization of distributed applications. To overcome this problem, we have developed a performance prediction tool called ChronosMix, which can predict the execution time of a distributed algorithm on parallel or distributed architecture. In this article we present the performance prediction of an NAS Benchmark program with the ChronosMix environment. This study aims at emphasizing the contribution of our environment in the performance prediction process.

1 Introduction

Usually scientific applications are intended to run only on dedicated multiprocessor machines. With the continual increase in workstation computing powers and especially the explosion of communication speed, the networks of workstations (NoW) became possible distributed platforms of execution and inexpensive for scientific applications. Its main problem lies in the heterogeneity of NoW compared to the homogeneity of the multiprocessor machines. In a NoW, it is difficult to allocate the entire work in an optimal manner, it is difficult to know the exact benefit or if there is any or simply to know which best algorithm to solve a specific problem is. Therefore, the optimization of the distributed application is a hard task to achieve.

A way to meet these objectives is to use performance prediction.

We identify three categories of tools that realize performance evaluation of a parallel program. Their objectives are quite different, because they use three different types of input: processor language, high-level language or modeling formalism.

The aim of tools based on a processor language, like Trojan [Par96] and SimOS [RBDH97], is to provide a very good accuracy. Thus, they avoid the introduction of compiler perturbations due to optimization. But, they work on

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 208–216, 2000.

unique architecture and cannot be extended to any other. Finally, this tool category implies an important slowdown.

The aim of tools based on high-level language is to allow adapted accuracy in designing parallel applications with a minimum slowdown. Mainly, this tool category has the ability to calculate application performance on various types of architecture from an execution trace. P3T [Fah96], Dimemas [GCL97], Patop [WOKH96] and ChronosMix [BST99] belong to this category. The P3T tool is part of the Vienna Fortran Compilation System and its aim is to evaluate and classify parallel strategies. P3T helps the compiler to find the appropriate automatic parallelization of the sequential Fortran program in homogeneous architecture. The Dimemas tool is part of the Dip environment [LGP+96]. It simulates distributed memory architecture. It exclusively uses trace information and does not realize any program model. So, Dimemas is able to evaluate binaries like commercial tools and various types of architecture and communication libraries like PVM and MPI. Patop is a performance prediction tool based on an on-line monitoring system. It is part of The Tool-set [WL96] and allow to analyze performances on homogeneous multi-computers.

The aim of a modeling formalism tool is to simplify the application description and to put it in a tuning up form. With this type of tools, the accuracy is heavily linked with the modeling quality. Pamela [Gem94] and BSP [Val90] are from this category. The Pamela formalism is strong enough to allow complex system modeling. Various case studies have been conducted with Pamela and many algorithm optimizations have been realized.

However, current tools and methods have disadvantages that do not allow an efficient use of performance prediction. The three main disadvantages of performance evaluation tools are:

- The slowdown is the ratio between the time to access simulation results and real execution time. It is calculated for one processor and the different times must come from the same architecture.
- The use constraints
- The lack of heterogeneous support

Our tool, ChronosMix, has been developed by taking account of these aspects. The slowdown has been minimized, the use has been improved thanks to automatic modeling and the heterogeneity has been integrated. These main aspects are emphasized in a case study.

This paper starts in section 2 with a description of the performance evaluation tool for a parallel program called ChronosMix. It ends in section 3 with a case study from the NAS Benchmark.

2 Presentation of the ChronosMix Environment

The purpose of ChronosMix [BST99] is to provide the most complete performance prediction environment as possible in order to help in the designing and the optimizing of distributed applications. To do so, ChronosMix environment comprises different modules:

– Parallel architecture modeling (MTC and CTC)
– Program modeling (PIC)
– Simulation engine
– Graphical interface (Predict and ParaGraph)
– Database web server

Figure 1 illustrates the relations between these different modules.

Parallel architecture modeling consists of two tools the MTC (Machine Time Characteriser) and the CTC (Communication Time Characteriser). The MTC is a micro-benchmarking tool which measures the execution time of instructions of which a machine instruction set is composed. A given local resource will just be assessed once by the MTC, meaning that the local resource model can be useful in the simulation of all program models. The CTC is an expansion of SKaMPI [RSPM98]. It measures MPI communication times by varying the different parameters. CTC measures are written in a file, making it necessary for a cluster to be measured just once.

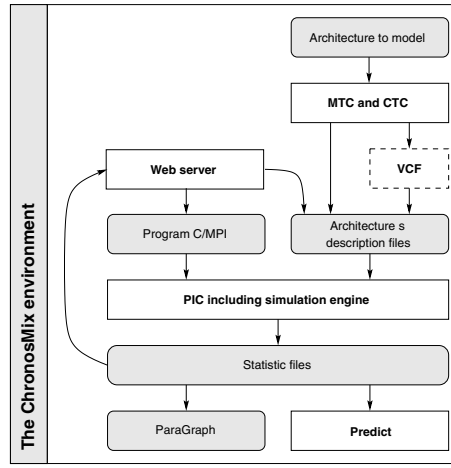


Fig. 1. The ChronosMix environment

The C/MPI program modeling and the simulation engine are both included in the PIC (Program Instruction Characteriser), which is a C++ 15,000-line program. Figure 2 shows how the PIC works. On the left side of figure 2 the method for analyzing the input C/MPI program will be entirely static. Indeed, the execution number of each block and the execution time prediction are calculated statically. On the right, you can see a program passing through the PIC in a semi-static way. The number of executions is attributed to each block by means of an execution and a trace. However, the execution time prediction phase is static, which explains why the program is analyzed through the PIC in a semi-static way.

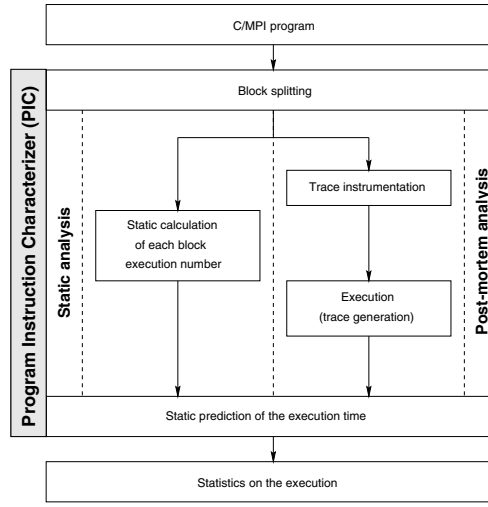


Fig. 2. PIC internal structure

ChronosMix environment currently comprises two graphical interfaces called ParaGraph and Predict. A third interface, the VCF (Virtual Computer Factory), is being developed which will allow new performance prediction architecture to be made.

3 Performance Prediction of the NAS Integer Sorting Benchmark

3.1 Presentation of the Integer Sorting Benchmark

The program used for testing the efficiency of our performance prediction tool is from the NAS (Numerical Aerodynamic Simulation) Parallel Benchmark [BBDS92] developed by the NASA. The NAS Parallel Benchmark (NPB) is a set of parallel programs designed to compare the parallel machine performances according to the criteria belonging to the aerodynamic problem simulation. These programs are widely used in numerical simulations. One of the 8 NPB programs is the parallel Integer Sorting (IS) [Dag91]. This program is based on the barrel-sort method. The parallel integer sorting is particularly used in the Monte-Carlo simulations integrated in the aerodynamic simulation programs.

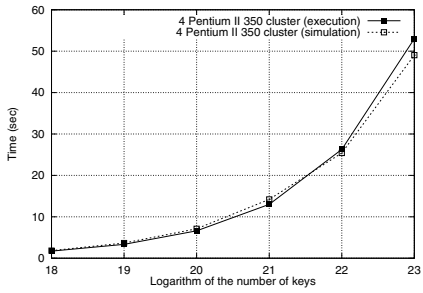
IS consists of various types of MPI communication. Indeed, it uses asynchronous point-to-point communications, broadcasting and gathering functions as well as all-to-all functions. IS therefore deals with network utilization in particular. For that matter, the better part of its execution time is spent in communication.

3.2 Comparison of IS on Various Types of Architecture

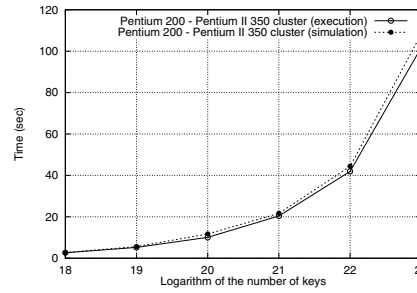
Four types of architecture have been used to test the ChronosMix accuracy :

- A cluster composed of 4 Pentium II 350Mhz with Linux 2.0.34, gcc 2.7.2.3 and a 100 Mbit/s Ethernet commuted network.
- A heterogeneous cluster composed of 2 Pentium MMX 200Mhz and 2 Pentium II 350Mhz with Linux 2.0.34, gcc 2.7.2.3 and a 100 Mbit/s Ethernet commuted network.
- A cluster composed of 4 DEC AlphaStation 433Mhz with Digital UNIX V4.0D, the DEC C V5.6-071 compiler and a 100 Mbit/s Ethernet commuted network.
- The same cluster but with 8 DEC AlphaStation 433Mhz

All the execution times are an average of 20 executions. Figure 3 shows the IS execution and the IS prediction on the Pentium clusters. The error percentage between prediction and real execution is represented in figure 5. The cluster comprising only Pentium II is approximately twice as fast as the heterogeneous cluster. Replacing the two Pentium 200 with two Pentium II 350 is interesting - the error noticed between the execution time and the prediction time remains acceptable since it is not over 15% for the heterogeneous cluster and 10% for the homogeneous cluster, the average error amounting to 7% for the heterogeneous and the homogeneous cluster.



(a) IS on a 4 Pentium II cluster

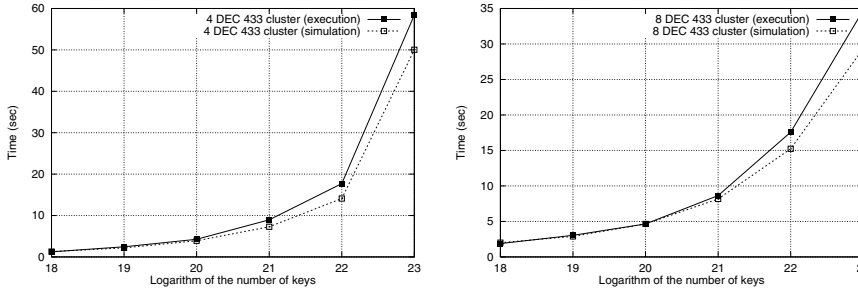


(b) IS on a 2 Pentium 200 - 2 Pentium II cluster

Fig. 3. Execution and prediction of IS on Pentium clusters

Figures 4(a) and 4(b) present IS execution and IS prediction respectively for the 4 DEC AlphaStation cluster and for the 8 DEC AlphaStation cluster. In parallel with figure 5 they show that the difference between prediction and execution remain acceptable. Indeed, concerning the 4 DEC AlphaStation cluster, the difference between execution and prediction is not over 20% and the average

is 12%. As for the 8 DEC AlphaStation cluster, the maximum error is 24% and the average is 9%.



(a) 4 DEC AlphaStation cluster (b) 8 DEC AlphaStation cluster

Fig. 4. IS on 4 and 8 DEC AlphaStation clusters

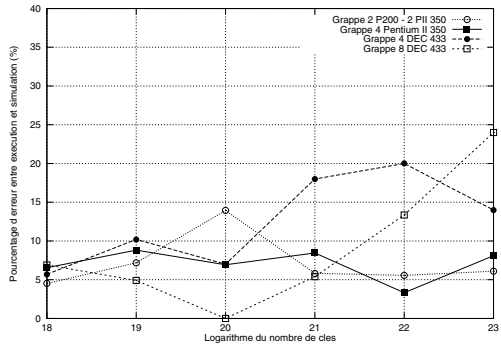
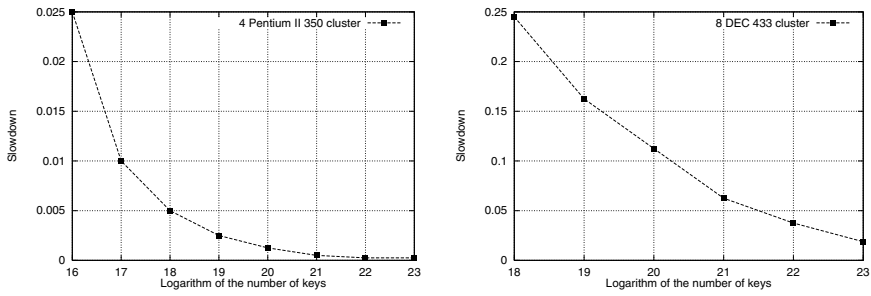


Fig. 5. Error percentage between prediction and execution on the three type of architecture

The integer sorting algorithm implemented by IS is frequently found in numerous programs and the executions and the simulations have dealt with real-size problems. Therefore, on a useful, real-size program, ChronosMix proved able to give relatively accurate results.

The other quality of ChronosMix lies in how quickly the results are obtained. Indeed, if a simulator is too long in giving the results, the latter can become quite useless. Even in semi-static mode, ChronosMix can prove to be faster than an execution.

A trace is generated for a given size of cluster and for a given number of keys. This trace, generated on any computer capable of running an MPI program. Concretely, traces have been generated on a bi-Pentium II 450 and were then casually used in the IS prediction on the 4 DEC AlphaStation cluster and on the 4 Pentium II cluster. This shows it is difficult to count the trace generating time in the total performance prediction time. Figures 6(a) and 6(b) show the IS performance prediction slowdown. Normally, the slowdown of a performance prediction tool is greater than 1, meaning that the performance prediction process is longer than the execution. For this example, the ChronosMix slowdown is strictly smaller than 1, meaning that by simulated processor, performance prediction is faster than the execution. This result shows that slowdown is an inadequate notion for ChronosMix. Performance prediction of the Pentium II cluster is at least 10 times faster than the execution. With the maximum number of keys, ChronosMix is 1000 times faster in giving the program performances than the real execution. Concerning the 8 DEC AlphaStation cluster, the ChronosMix slowdown is always below 0.25 and falls to 0.02 for a number of keys of 2^{23}. If the slowdown is reduced when the number of keys rises, it is simply because the performance prediction time is constant whereas the execution time rises sharply according to the number of keys to sort.



(a) 4 Pentium II cluster (b) 8 DEC AlphaStation cluster

Fig. 6. Slowdown for the IS performance prediction

4 Conclusion

The IS performance prediction has revealed three of the main ChronosMix qualities:

Its speed. On average, ChronosMix is much faster in giving the application performances than an execution on the real architecture.

Its accuracy. On average, the difference between the ChronosMix prediction and the execution on the different types of architecture is 10%.

Its adaptability ChronosMix proved it could adapt to 2 types of parallel architecture and to several sizes of cluster.

The ability to model distributed system architecture with a set of micro-benchmarks allows ChronosMix to take heterogeneous architecture completely into account. Indeed, in a sense, modeling is automatic, because simulation parameters are assigned by the MTC execution to the local resources and by the CTC execution between workstations. The distributed architecture is simply and rapidly modeled, which allows to follow the processor evolution, but also to adapt our tool to a wide range of architecture.

It is possible to build target architecture from existing one by extending the distributed architecture, e.g. a set of one thousand workstations. It is also possible to modify all the parameters of the modeled architecture, e.g. to stretch the network bandwidth or to increase the floating-point unit power four-fold.

References

[BBDS92] D.H. Bailey, E. Barszcz, L. Dagum, and H. Simon. The NAS parallel benchmarks results. In *Supercomputing'92*, Minneapolis, November 16–20 1992.

[BST99] J. Bourgeois, F. Spies, and M. Trhel. Performance prediction of distributed applications running on network of workstations. In H.R. Arabnia, editor, *Proc. of PDPTA'99*, volume 2, pages 672–678. CSREA Press, June 1999.

[Dag91] L. Dagum. Parallel integer sorting with medium and fine-scale parallelism. Technical Report RNR-91-013, NASA Ames Research Center, Moffett Field, CA 94035, 1991.

[Fah96] T. Fahringer. *Automatic Performance Prediction of Parallel Programs.* ISBN 0-7923-9708-8. Kluwer Academic Publishers, Boston, USA, March 1996.

[GCL97] S. Girona, T. Cortes, and J. Labarta. Analyzing scheduling policies using DIMEMAS. *Parallel Computing*, 23(1-2):23–24, April 1997.

[Gem94] A.J.C. van Gemund. The PAMELA approach to performance modeling of parallel and distributed systems. In editors G.R. Joubert et al., editor, *Parallel Computing: Trends and Applications*, pages 421–428. 1994.

[LGP+96] J. Labarta, S. Girona, V. Pillet, T. Cortes, and L. Gregoris. DiP: A parallel program development environment. In *Proc. of Euro-Par'96*, number 19 in 2, pages 665–674, Lyon, France, August 1996.

[Par96] D. Park. *Adaptive Execution: Improving performance through the runtime adaptation of performance parameters.* PhD thesis, University of Southern California, May 1996.

[RBDH97] M. Rosenblum, E. Bugnion, S. Devine, and S.A. Herrod. Using the SimOS machine simulator to study complex computer systems. *ACM TOMACS Special Issue on Computer Simulation*, 1997.

[RSPM98] R. Reussner, P. Sanders, L. Prechelt, and M. Müller. SKaMPI: A detailed, accurate MPI benchmark. *LNCS*, 1497:52–59, 1998.

[Val90] L.G. Valiant. A bridging model for parallel computation. *Communications of the ACM (CACM)*, 33(8):103–111, August 1990.

[WL96] R. Wismüller and T. Ludwig. The Tool-set – An Integrated Tool Environment for PVM. In H. Lidell and al., editors, *Proc. HPCN*, pages 1029–1030. Springer Verlag, April 1996.

[WOKH96] R. Wismüller, M. Oberhuber, J. Krammer, and O. Hansen. Interactive debugging and performance analysis of massively parallel applications. *Parallel Computing*, 22(3):415–442, March 1996.