Paramedir: A Tool for Programmable Performance Analysis

Gabriele Jost[1]*, Jesus Labarta[2], and Judit Gimenez[2]

[1]NAS Division, NASA Ames Research Center, Moffett Field, CA 94035-1000 USA
gjost@nas.nasa.gov
[2]European Center for Parallelism of Barcelona-Technical University of Catalonia (CEPBA-UPC), cr. Jordi Girona 1-3, Modul D6,08034 – Barcelona, Spain
{jesus,judit}@cepba.upc.es

Abstract. Performance analysis of parallel scientific applications is time consuming and requires great expertise in areas such as programming paradigms, system software, and computer hardware architectures. In this paper we describe an extension to the Paraver performance analysis system that facilitates the programmability of performance metric calculations thereby allowing the automation of the analysis and reducing the application development time.

1 Introduction

Successful performance analysis is one of the great challenges when developing efficient parallel applications. Meaningful interpretation of a large amount of performance data requires significant time and effort. A variety of software tools have been developed to assist the programmer in this task. An example of a commercial product is Vampir [9] which allows tracing and trace visualization of message passing and OpenMP [6] applications. In order to analyze the performance the user will typically inspect timeline views of processes and threads, calculate performance statistics for parts of the code, and try to identify the problem. There are several research efforts on the way with the goal to automate this process. We can only name a few. The URSA MINOR project [8] at the Purdue University uses program analysis information as well as performance trace data in order to guide the user through the program optimization process. The Paradyn Performance Consultant [4] automatically searches for a set of performance bottlenecks. The SUIF Explorer [3] Parallelization Guru developed at Stanford University uses profiling data to bring the user's attention to the most time consuming sections of the code. KOJAK [2] is a collaborative project of the University of Tennessee and the Research Centre Juelich for the development of a generic automatic performance analysis environment for parallel programs aiming at the automatic detection of performance bottlenecks.

* The author is an employee of Computer Sciences Corporation.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3036, pp. 466–469, 2004.

One crucial issue that has not been sufficiently addressed in the previous work, according to our opinion, is the flexibility in the calculation of performance metrics. The reasons for poor performance will usually be due to the interaction of a plethora of factors stemming from hardware, system software, and choice of the programming paradigm. Every new computer architecture or programming model will give rise to new metrics that need to be checked and compared during the performance analysis. The tool presented in this paper is based on the Paraver [7] performance analysis system. It allows the automatic calculation of complex performance metrics that have been predefined by expert users based on visual inspection of the trace data. We describe the extension of Paraver that provides for the programmability of the performance analysis process in Section 2 and draw our conclusions in Section 3.

2 Paramedir

Paraver is a performance analysis system that is being developed and maintained at the European Center for Parallelism of Barcelona-Technical University of Catalonia (CEPBA-UPC). It supports a variety of programming paradigms and enables the user to obtain a qualitative global perception of the application's behavior as well as a detailed quantitative analysis of the performance. The Paraver distribution includes a tracing package, OMPItrace [5], with a simple but very flexible format. The trace file contains a wealth of information, which must be filtered and interpreted in order to obtain meaningful statistics. Paraver has filter and semantic modules that provide a high degree of flexibility for the specification of time line views to be displayed by the Paraver graphical user interface (GUI). An analysis module allows the calculation of meaningful statistics. The specification of filters, semantics, and metric calculations can be saved to re-usable configuration files. This way know-how can be transferred from the experienced to the novice user. Nevertheless, the displayed information still needs to be visually inspected in order to draw conclusions. At this point we should mention that *ver* in the name Para**ver** is Spanish for *to see*.

We have extended the Paraver system by a non-graphical user interface to the Paraver analysis module. The new module, Paramedir (**Para**llel **Medir**, where *medir* is Spanish for *to measure*) is a command line tool that takes a performance trace file and a Paraver analysis configuration file as input. It generates an ASCII table containing the requested performance metrics, which can be used for further processing. Paramedir accepts the same trace and configuration files as Paraver. This way the same information can be captured in both systems. The internal structure of Paraver and Paramedir and their relationship is shown in Figure 1.

Paramedir supports the programmability of performance analysis in the sense that complex performance metrics, determined by an expert user, can be automatically computed and processed. An example for the usage of Paramedir is to automate the detection of reasons for poor performance. Paramdir was used within the prototype implementation of an expert system for automatic performance analysis [1]. A whole set of configuration files is applied to the same trace file and the performance metrics

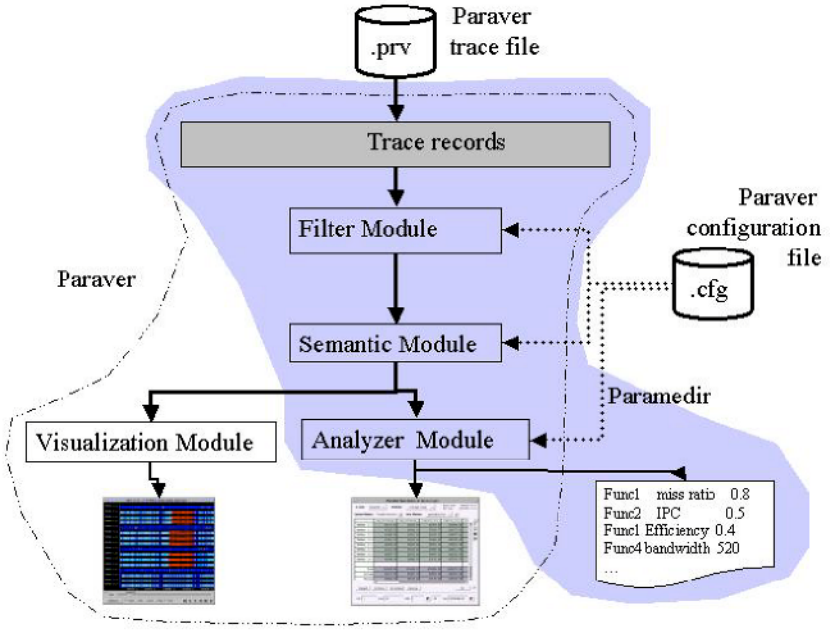

Fig. 1. Internal structure of Paraver and Paramedir. Paramedir uses the shaded components. Paramedir is a command line tool that takes a performance trace file and a Paraver analysis configuration file as input. It generates an ASCII table containing the requested performance metrics

are compared against empirically determined thresholds. Performance problems are determined by applying a set of rules to the outcome of the threshold tests. A high degree of flexibility in the calculation of performance metrics is essential in this scenario. For example, in order to determine why there is an imbalance in the computation time among the threads of a parallel program, many metrics need to be checked. Large sequential sections or an imbalance in the computational workload within the parallel sections are potential reasons. On NUMA architectures, the experienced user may also want to check the cost of L2 cache misses, in order to detect problems related to memory placement. If we denote the number of instructions by *Instr*, the number of L2 misses by *L2misses* and the ideal number of instructions per seconds for the system by *idealMPIS* , then we can estimate the cost for an L2 miss as:

$$Estimated\ L2cost = (Elapsed\ Time - Instr/(idealMIPS))/(L2misses)$$

The Paraver configuration files provide sufficient flexibility to specify this metric and Paramedir allows its calculation in batch mode. The automatic checking of performance metrics saves time for the experienced user and points the novice user, who may often not know what to look for, to potential performance problems.

3 Conclusions

We have extended the Paraver performance analysis system by a tool that facilitates the programmability of performance metric calculations and discussed a usage scenario within an expert system for performance analysis. The first conclusion we draw is, that the great challenge in automatic performance analysis is to de-mangle the factors that influence the performance of the program in the right way. The extensive tracing and analysis capabilities of Paraver are crucial to meet this challenge. Secondly, we found it to be very important that the GUI based Paraver system and the command line based Paramedir module share the same configuration files. This makes it possible to switch from one tool to the other at any point during the analysis process. The automated analysis using Paramedir rapidly guides the user to code segments that require further detailed analysis with Paraver. The detailed analysis will often lead to the design of new analysis configuration files which can then, in turn be included in the automated process.

Acknowledgements. This work was supported by NASA contract DTTS59-99-D-00437/A61812D with Computer Sciences Corporation/AMTI, by the Spanish Ministry of Science and Technology, by the European Union FEDER program under contract TIC2001-0995-C02-01, and by the European Center for Parallelism of Barcelona (CEPBA).

References

1. G. Jost, R. Chun, H. Jin, J. Labarta, and J. Gimenez, *"An Expert System for the Development of Efficient Parallel Code"*, to be presented at PARA'04, Kopenhagen, Denmark, June 2004.
2. KOJAK Kit for Objective Judgment and Knowledge based Detection of Performance Bottlenecks, `http://www.fz-juelich.de/zam/kojak/`.
3. S. Liao, A. Diwan, R. P. Bosch, A. Ghuloum, M. Lam, "*SUIF Explorer: An interactive and Interprocedural Parallelizer*", 7th ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming, Atlanta, Georgia, (1999), 37-48.
4. B.P. Miller, M.D. Callaghan, J. M. Cargille, J. K. Hollingsworth, R. B. Irvin, K.L. Karavanic, K. Kunchithhapdam and T. Newhall, "*The Paradyn Parallel Performance Measurement Tools",* IEEE Computer 28, 11, pp.37-47 (1995).
5. OMPItrace User's Guide, https://www.cepba.upc.es/paraver/manual_i.htm
6. OpenMP Fortran/C Application Program Interface, `http://www.openmp.org/`.
7. Paraver, `http://www.cepba.upc.es/paraver/`.
8. I. Park, M. J. Voss, B. Armstrong, R. Eigenmann, *"Supporting Users' Reasoning in Performance Evaluation and Tuning of Parallel Applications",* Proceedings of PDCS'2000, Las Vegas, NV, 2000.
9. VAMPIR User's Guide, Pallas GmbH, `http://www.pallas.de`.