Parallel Performance Evaluation: The Medea Tool

M. Calzarossa, L. Massari, A. Merlo, D. Tessera

Dipartimento di Informatica e Sistemistica, Università di Pavia
Via Ferrata 1, I–27100 Pavia, Italia
e-mail: {mcc,massari,merlo,tessera}@gilda.unipv.it

Abstract. The performance of parallel programs is influenced by the multiplicity of hardware and software components involved in their executions. Experimental approaches, where trace files collected at run–time by monitors are the basis of the analyses, allow a detailed evaluation of the performance. Quantitative as well as qualitative information related to the behavior of the programs are required. Medea is a parallel performance evaluation tool which provides various types of statistical and numerical techniques integrated with visualization facilities such that both quantitative and qualitative descriptions of the programs are obtained. A large variety of studies dealing with tuning, performance debugging, and code optimization profitably benefits of Medea.

1 Introduction

Performance evaluation of parallel programs is typically based on experimental approaches (see e.g., [CM94], [CMM95a]) which require the analysis of large amounts of data collected at run–time by monitoring the executions of the programs. Many tools (see e.g., [HL91], [RAN+93], [RW93], [MBM94], [HMR95]) provide a few basic statistical information of the achieved performance together with diagrams which visually summarize the program behavior. More detailed data analysis techniques are required to improve the quantitative description which helps both system and program developers in testing alternative hardware and software configurations and in optimizing the code.

Medea is a software tool which meets such objectives. It allows the evaluation of the performance of parallel programs by automatically processing the trace files produced by monitors and by combining various types of statistical and numerical techniques together with visualization facilities. Synthetic and easy–to–interpret representations, very useful for studies dealing with tuning, performance debugging and diagnosis, and evaluation of system configurations, are provided (see e.g., [CMM+95b]). The integration of Medea within a parallel compilation system is currently under development in the framework of the ESPRIT IV Project HPF+ (see [HPF95]).

This paper presents the main features of Medea and is organized as follows. The design of the tool is presented in Section 2. Section 3 describes how Medea processes the trace files produced by monitoring tools and which metrics and

parameters can be obtained. In Section 4, the data analysis and visualization techniques implemented within our tool are described. Finally, conclusions with future extensions of the tool are drawn in Section 5.

2 Design of the tool

The main design feature of Medea has been the definition of a friendly environment where performance studies could be easily exploited. This goal was achieved by integrating the functionalities (modules) of the tool by means of a graphical interface built upon the OSF/Motif standard. Figure 1 shows the main window of the interface where the buttons corresponding to the various modules of Medea are listed in the menu bar.

Specific functionalities, used either in isolation or in combination with each other, are provided in order to obtain insights into the behavior of parallel programs. Performance parameters and metrics (e.g., communication times and speedup curves) are derived by means of the `Filtering` module, according to the level of detail of the monitoring activity and to the level at which the trace files are processed (see Sect. 3). The selected levels identify the components to be considered in the following analyses. Once parameters and metrics have been determined, the `Clustering` and `Fitting` modules can be activated. They implement detailed statistical and numerical techniques which derive a quantitative description of the behavior of the analyzed programs (see Sect. 4).

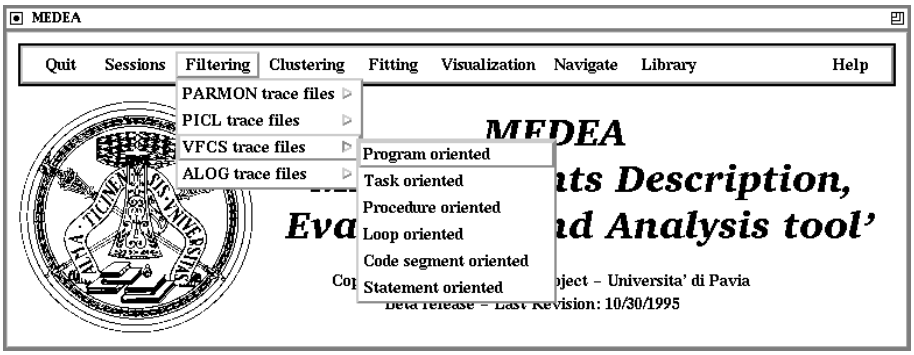

Fig. 1. Selection of VFCS trace files within the main window of Medea.

The remaining buttons of the menu bar of Fig. 1 are related to additional functionalities of Medea, dealing both with the management of the support environment and with the visualization of the obtained results. The `Sessions` module offers the possibility of specifying analysis sessions and, within them, different experiments. A "session" is the logical organization of the analyses performed on specific trace files. A session, in turn, consists of one or more

experiments. Each "experiment" corresponds to a particular set of input specifications within the analysis modules of Medea. For example, different runs of the `Clustering` module correspond to different experiments within the same session. The `Visualization` and `Navigate` modules represent a common framework for graphical representations, such as, pie charts and distributions, of the results produced by Medea and for browsing throughout text files, such as, input traces and logs of the analyses carried out within sessions. The `Library` module controls the definition and management of internal record structures to be used within the analyses performed by Medea. The on–line `Help` facility provides information at run–time about the general usage of the tool and of its different modules.

3 Trace file processing

Measurements produced by the monitoring tools and collected into trace files are typically related to the begin and the end of each event generated while executing a parallel program.
Accurate studies require the analysis of parameters and metrics, such as, execution times and speedup curves, which can be directly derived from the trace files by means of the `Filtering` module of Medea. This module a processes the measured data in order to extract the events of interest. Figure 1 shows a menu of the `Filtering` module where the trace file formats currently supported by Medea, namely, PARMON [PSV94], PICL [Wor92], VFCS [ZC93], ALOG [BL94], are listed. Note that, because of the modular structure of Medea, routines to process traces produced by different monitors can be easily integrated within the tool.
The figure also presents the submenu for the selection of the level of detail at which VFCS trace files can be processed. Possible levels are whole program, task, procedure, loop, code segment or single statement.

Once the trace files to be analyzed have been specified, parameters, such as, computation and communication times, number and length of exchanged messages, are derived. The number and the type of these parameters depend both on the information collected into the trace files and on the objectives of the analysis. For example, when scalability issues of a parallel program have to be addressed, one parameter of interest is the global execution time obtained varying the number of allocated processors.
Several parallel metrics can be obtained from the `Filtering` module. Examples are the profile curves, which represent the number of processors performing a specific activity, namely, execution, communication, computation, transmit, receive, and I/O, as a function of the execution time of the program. Speedup, efficiency, efficacy, and signatures, are also computed.
In what follows, we present a few results of the performance achieved by a hydrodynamic program executed on an IBM Sp2 at the Maui HPCC with a number of allocated processors ranging from 1 up to 64. These results are not to be intended as a case study, but as an aid to the description of the data analysis and visualization facilities of Medea.

4 Data analysis and visualization

As a result of the processing of the trace files, metrics and parameters are presented by means of the `Visualization` module of Medea. By looking at them, a first insight on the behavior of the program can be derived. The speedup curve and the signatures of the considered program are presented in Fig. 2(a) and Fig. 2(b). These signatures express the total execution, computation, and communication times as a function of the number of allocated processors.
Let us remark that depending on the events stored into trace files, the communication signature can account either for the time spent by the communication processors, or for the portion spent in communication activities by the computing processors only.

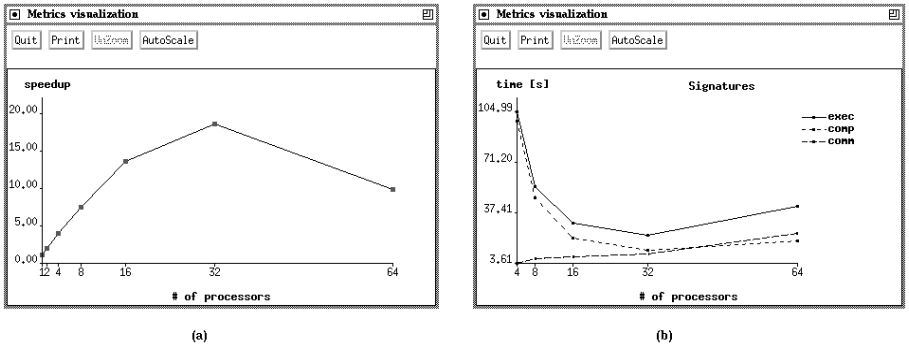

Fig. 2. Speedup (a) and signatures (b) curves.

As can be seen in Fig. 2(a), when more than 32 processors are used, the speedup decreases as a consequence of the communication time which exceeds its computation counterpart (see Fig. 2(b)). Such metrics can be used to determine the "optimal" number of processors, that is, the processor working set, to be assigned to a program in order to achieve a good balance between costs and performance.

Further insights into the program behavior can be obtained by means of various types of data analysis techniques. Such techniques, applied to the parameters derived by processing the trace files, can be as simple as basic statistics or as detailed as cluster analysis and numerical fitting techniques (see e.g., [Har75], [DS81]). The main goal of these analyses is the construction of a synthetic description, that is, a model, of the performance achieved by the program under study.
In the `Clustering` module, as a preliminary analysis, basic statistics, which help in evaluating the behavior of the parameters used to describe the components (e.g., programs, tasks, code segments), can be computed. For each parameter, indices, such as, mean and standard deviation, are obtained and displayed in a tabular form. Frequency and cumulative distributions are also presented, together with different percentile values.

The visualization window which allows the selection of the various types of results produced by Medea is presented in Fig. 3. The figure also shows five rows of the table containing the basic statistics of the times taken by SEND messages issued by each of the 64 processors allocated to the hydrodynamic program. The distribution of these times for processor p0, displayed in Fig. 4(a), shows that most of the messages are characterized by a time smaller than the mean value, that is, 0.065 ms.

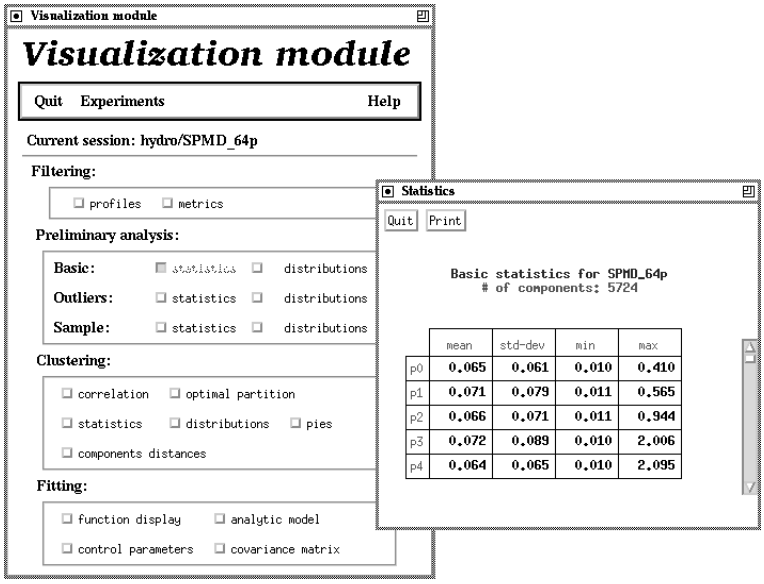

Fig. 3. Visualization window of Medea and basic statistics of the times (expressed in ms) taken by SEND messages on processors p0, . . . , p4.

These distributions and basic statistics, together with the percentiles, help in identifying the outliers, which is usually convenient to eliminate before clustering. They are those components having one or more parameters values very different from the corresponding values of the other components. Moreover, when the number of components becomes large, a sample may be taken before applying clustering.

Cluster analysis identifies homogeneous groups (classes) of components described by the parameters selected according to the objectives of the analysis. Note that this selection is also driven by looking at the correlation matrix which identifies highly positively correlated parameters. The components, considered as points in a multidimensional space, are grouped according to their similarities. The result of the cluster analysis is a partition into groups, characterized by their centroids, i.e., their geometric centers. For each of these classes, basic statistics and distributions of the parameters used for the analysis are provided. Figure 4(b)

refers to an example where three groups have been considered; in particular, the distribution of the times taken by SEND messages for processor p0 within the second group (`2.3`) is shown. This distribution is characterized by a mean value of 0.18 ms, with respect to the average value, equal to 0.065 ms, computed for all the components (see Fig. 3).

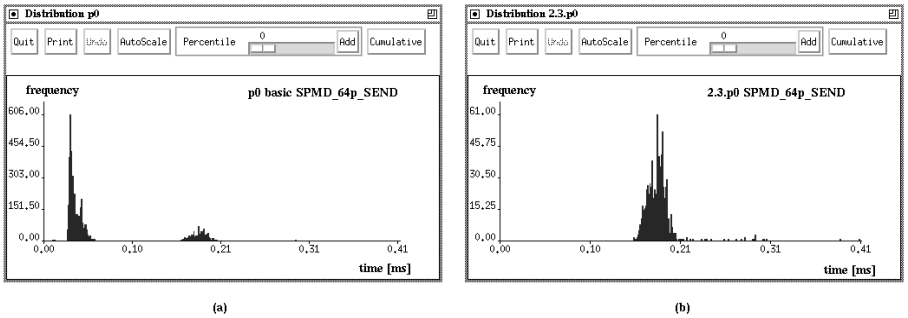

Fig. 4. Distributions of the times taken by SEND messages on processor p0 before (a) and after (b) cluster analysis.

Further analyses can be performed by means of numerical fitting techniques. The `Fitting` module has been designed to provide analytical representations of the dynamic behavior of parallel programs. Their internal activities, such as, communication, I/O, and synchronization, are captured by these representations. Examples include global and per–protocol communication profiles and metrics, like speedup and signatures. Such compact representations can be used to study various aspects of the program behavior, such as, scalability. The `Fitting` module summarizes the experimental measures by means of analytical expressions which can be simple functions like polynomials, trigonometrics, exponentials, or combinations of a few of them. Depending on the objective of the study, it is also possible to use customized expressions.

Once the fitted function has been computed, it is displayed by means of the `Visualization` module of Medea. The representation of the fitted function versus the experimental measures provides a rough evaluation of its ability to capture the characteristics of interest in the analyzed phenomenon. It may also suggest modifications to the fitting function in order to obtain a better match with the experimental measures. Figure 5 shows an example where a portion of a communication profile, consisting of 909 out of 18830 events expressing the number of processors involved in communication activities as a function of the execution time, has been fitted with a polynomial function of degree seven. As can be seen, the fitted polynomial is a good approximation which resumes the overall behavior of the profile.

Mathematical details of the analysis, such as, the analytical expression of the fitted function, together with its coefficients, their covariance matrix, and the

final residual, are produced in a tabular form.

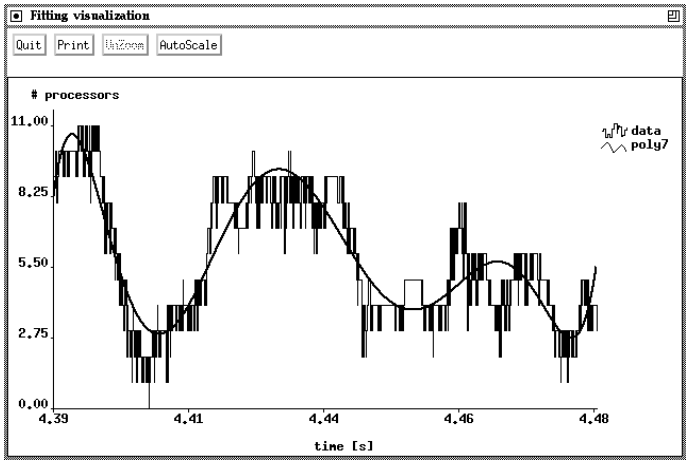

Fig. 5. Communication profile and fitted function.

5 Conclusions

Experimental approaches towards the evaluation of the performance of parallel programs are very appropriate, although difficult to apply. The Medea tool is particularly useful for these purposes. It starts from the analysis of trace files produced by monitors and it provides quantitative as well as qualitative evaluation of the performance.
In the framework of ESPRIT IV Project no. 21033 HPF+, a tighter integration of Medea with the Vienna Fortran Compilation System (VFCS) is in progress with the aim of evaluating the performance of HPF+ programs and of providing feedback to their developers. The tool will also be improved by enhancing the results currently provided with their functional description. This description will relate the performance achieved by a program to its source code, and will help the performance debugging and code optimization. These extensions will also be part of the HPF+ Project.

Acknowledgments

Authors thank Marco Vidal for his invaluable help in the implementation of the fitting module of Medea.
The research was supported in part by the the Italian C.N.R., by the M.U.R.S.T. under the 40% and 60% Projects and by the ESPRIT IV Project no. 21033 HPF+. The research was also sponsored by the Phillips Laboratory, Air Force Materiel Command,

USAF, under cooperative agreement number F29601-93-2-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Phillips Laboratory or the U.S. Government.

References

[BL94] R. Butler and W. Lusk. Monitors, Messages, and Clusters: The p4 Parallel Programming System. *Parallel Computing*, 20:547–564, 1994.

[CM94] M. Calzarossa and L. Massari. Measurement-based Approach to Workload Characterization. In G. Haring, R. Marie, and G. Kotsis, editors, *Performance and Reliability Evaluation*, pages 123–147. Oldenbourg Verlag, 1994.

[CMM95a] M. Calzarossa, L. Massari, and A. Merlo. Performance Analysis of Concurrent Software: issues, methodologies and tools. In G. Balbo and M. Vanneschi, editors, *General Purpose Parallel Computers: Architectures, Programming Environments, and Tools*, pages 287–298. ETS, 1995.

[CMM+95b] M. Calzarossa, L. Massari, A. Merlo, M. Pantano, and D. Tessera. MEDEA – A Tool for Workload Characterization of Parallel Systems. *IEEE Parallel and Distributed Technology*, 3(4):72–80, 1995.

[DS81] N. Draper and H. Smith. *Applied Regression Analysis – Second Edition.* J. Wiley, 1981.

[Har75] J.A. Hartigan. *Clustering Algorithms.* J. Wiley, 1975.

[HL91] V. Herrarte and E. Lusk. Studying Parallel Program Behavior with `upshot`. Technical Report ANL-91/15, Argonne National Laboratory, 1991.

[HMR95] M.T. Heath, A.D. Malony, and D.T. Rover. Parallel Performance Visualization: From Practice to Theory. *IEEE Parallel and Distributed Technology*, 3(4):44–60, 1995.

[HPF95] HPF+ Optimizing HPF for Advanced Applications, Technical Annex. Information Technologies Programme 1994–1998 (ESPRIT IV), Domain 4: Reactive Long Term Research, 1995.

[MBM94] B. Mohr, D. Brown, and A.D. Malony. TAU: A Portable Parallel Program Analysis Environment for pC++. In *Proc. of CONPAR 94 - VAPP VI*, LNCS 854, pages 29–40. Springer-Verlag, 1994.

[PSV94] E. Pozzetti, G. Serazzi, and V. Vetland. ParMon – A tool for monitoring parallel programs. Technical Report CNR "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo", R3/148, Rome, 1994.

[RAN+93] D.A. Reed, R.J. Aydt, R.J. Noe, K.A. Shields, B.W. Schwartz, and L.F. Tavera. Scalable Performance Analysis: The Pablo Performance Analysis Environment. In *Proceedings Scalable Parallel Libraries Conference*, pages 104–113. IEEE Computer Society, 1993.

[RW93] D.T. Rover and C.T. Wright. Visualizing the Performance of SPMD and Data-Parallel Programs. *Journal of Parallel and Distributed Computing*, 18:129–146, 1993.

[Wor92] P.H. Worley. A New PICL Trace File Format. Technical Report ORNL/TM–12125, Oak Ridge National Laboratory, 1992.

[ZC93] H.P. Zima and B. Chapman. Compiling for Distributed–Memory Systems. *Proc. of the IEEE*, 81(2):264–287, 1993.

This article was processed using the LaTeX macro package with LLNCS style