A Tool for Performance Analysis of Parallel Programs[1]

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

Dipartimento di Informatica e Sistemistica, Università di Pavia
Via Abbiategrasso, 209 - 27100 Pavia, Italia
e-mail: {mcc,massari,merlo,tessera,vidal}@gilda.unipv.it

Abstract: The analysis of the performance of parallel programs has to be carried out with the aid of appropriate tools able to provide compact and easy–to–interpret information about the behavior of the programs. MEDEA is a software tool which applies various types of visualization and workload characterization techniques to the raw measurements collected at run–time on parallel systems. Synthetic descriptions of the performance of parallel codes are obtained. Studies dealing with tuning, performance debugging and diagnosis largely benefit of these results.

1 Introduction

The analysis of the performance of parallel programs has to be approached with extreme care. The achieved performance is the result of the combination of the complex interactions of the hardware and software components involved in the execution of parallel programs. In order to describe and understand the behavior and the performance of these programs, measurements have to be collected by monitoring the code executions. Many tools for monitoring and profiling parallel programs have been developed (see e.g., [1], [2], [3], [4], [5]). Measurements hold all the information about the program performance. The program developers need this information for code optimizations and debugging. Hence, a post–processing of the raw data stored at run–time into trace files is required. The obtained results need to be presented in a compact and easy–to–interpret format such that they can be successfully used by the program developers for identifying the portions of the code responsible of performance losses. Nice diagrams (see e.g., [6], [7]), which represent the detailed behavior of the program, need to be coupled with some sort of more "condensed" results. This is particularly true in the case of massive parallel systems. Diagrams, such as communication matrices, Gantt charts, and pure animations of trace files, become useless when the amount of information to be represented is too large to be easily and intuitively presented and interpreted. Statistical and numerical techniques [8] interacting together in different ways have to be used for accurate and effective performance studies.

MEDEA (MEasurements Description, Evaluation and Analysis) is a software tool which combines visualization facilities together with various types of workload characterization techniques [9] for the analysis of parallel programs. The main goal of MEDEA is to provide synthetic descriptions of their behavior such that the most relevant information is presented in a "condensed" format. All the studies [10] aimed at the code optimization, e.g., tuning, performance debugging and diagnosis, will benefit of these analyses.

The paper is organized as follows. Section 2 describes the overall structure of the tool and its major functionalities. A closer look to the various modules it consists of is given in the following sections. Section 3 presents the pre–processing of the trace files. The statistical analysis and the fitting modules

[1] This work was supported in part by the the Italian Research Council (C.N.R.) and by the M.U.R.S.T. under the 40% and 60% Projects.

are described in Sections 4 and 5, respectively. Finally, a few conclusions are drawn in Section 6. Possible extensions of the tool are also pointed out.

2 The architecture of the tool

The starting point of performance evaluation studies is represented by the analysis of the measurements collected on real systems. All the monitoring tools employ manual or automatic instrumentation of the source code in order to trap the events generated during the program execution. Such measures are collected into trace files and they can be analyzed at various levels, according to the objectives of the analysis.

One of the main design features of MEDEA has been the definition of a friendly environment for the exploitation of performance studies. This goal has been achieved by structuring the tool in a modular way: each "module" addresses specific issues related to the analysis of parallel programs. Depending on the objectives of the study, a few of these modules are used in combination with each other in order to obtain the required information. Each module provides different results which give insights into particular aspects of the behavior of a parallel program and which represent, at the same time, the basis for the analyses applied within other modules. Parameters, such as, execution and communication times of the program, and metrics, such as, speedup and efficacy, are derived by means of *ad hoc* filtering routines (see Section 3). Once the values of these parameters have been determined, a few modules dealing with statistical and numerical analyses, such as, clustering [11] and fitting [12], can be applied (see Sections 4 and 5).

The integrated use of these modules is made easier by a graphical interface based upon the X Window and OSF Motif environments. Visualization facilities [13] represent a common framework for the presentation of the results provided by the various modules of MEDEA.

Another important feature of our tool is represented by the possibility of specifying analysis sessions and, within them, different experiments. A "session" is the logical organization of the analyses performed on specific measurements. Facilities for sessions management are provided. For example, once a session is restored, all the results of previous analyses are available for further analyses. A session may consist of one or more experiments. Each "experiment" corresponds to a particular set of input specifications within one or a few modules. For example, different runs of the clustering module correspond to different experiments.

In the following sections, the characteristics of the main analysis modules of MEDEA are briefly outlined with the aim of pointing out their role and their possible usage.

3 The "filter" module

Measurements provided by the monitoring tool are typically related to the events generated during the execution of a parallel program. For example, when the performance study is focused on the communication activities of a program, a trace file contains the time stamps related to the beginning and the end of each communication event. Accurate studies require the analysis of parameters and metrics, such as, execution time and speedup, which can be directly derived from the trace files by means of the "filter" module of MEDEA. This module performs a pre–processing of the measured data in order to extract the events of interest. Currently, trace formats of PICL [2], PARMON [14], and p4 [15] environments can be automatically analyzed. However, because of the modular structure of MEDEA, routines to process traces produced by other monitors can be easily integrated within the tool.

Once the trace files to be analyzed have been specified, parameters, such as execution, computation, and communication times, are computed. The number and the type of these parameters depend both on the kind of 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.

Furthermore, several parallel metrics can be obtained from this module. Profile curves represent the number of processors performing a specific activity as a function of the execution time of the program. Various types of profiles (namely, execution, communication, computation, transmit, receive, and I/O) can be produced. As an example, Figure 1 shows the communication profile obtained from the analysis of traces collected during the execution of an hydrodynamics code. Visualization of these profiles reveals particularly useful when tuning actions related to the communication activities have to be approached.

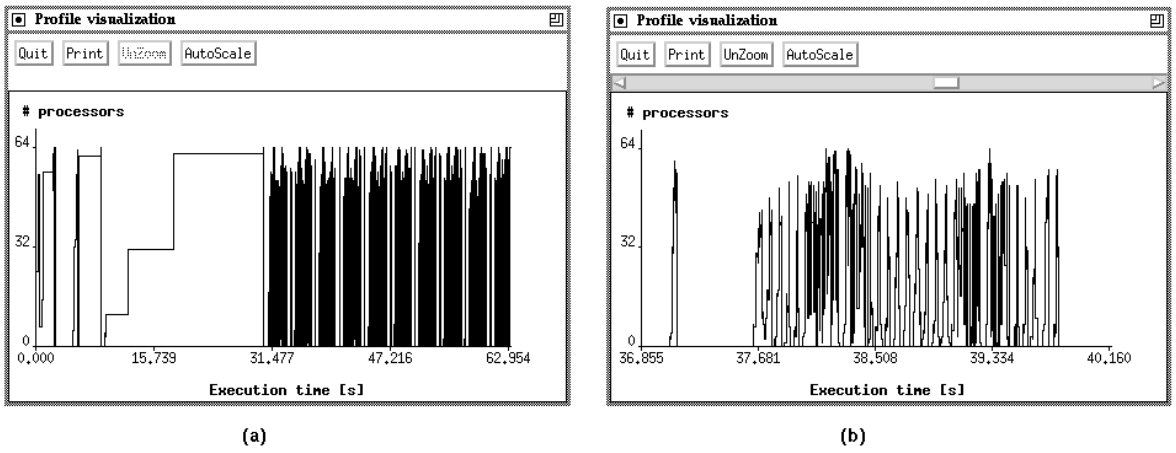


Figure 1: Communication profile (a) and zooming over one portion (b).

Other metrics, i.e., speedup, efficiency, efficacy, and execution signature, are also computed. Figure 2 shows the curves corresponding to the execution signature and the efficiency obtained from the analysis of a parallel FFT code. The execution signature expresses the total execution time of the program as a function of the number of processors. The efficiency is the ratio between the obtained speedup versus the number of allocated processors.

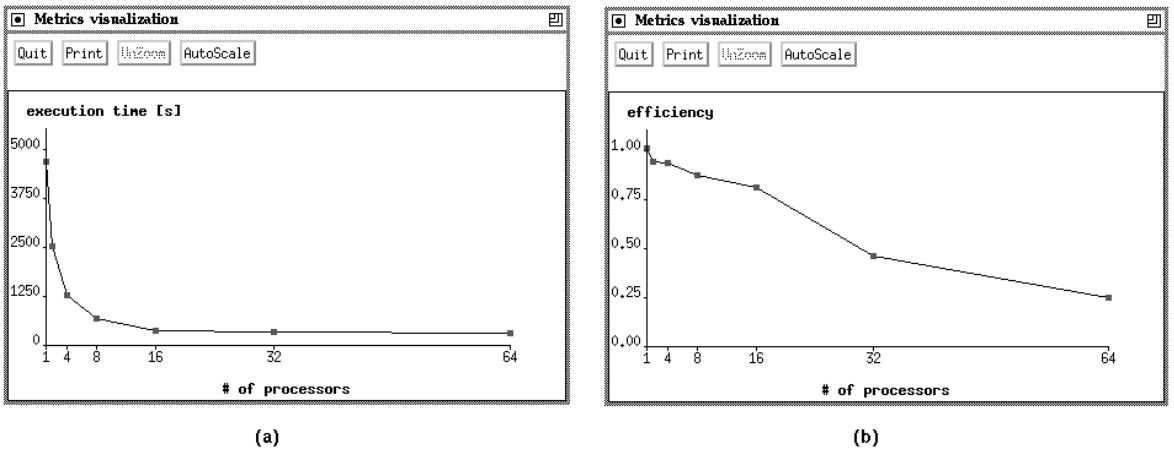


Figure 2: Execution signature (a) and efficiency (b) for a parallel FFT code.

These metrics allow the identification of possible performance losses that can be experienced by a parallel program when the number of allocated processors increases. In our example, the execution signature shows that the gain in the execution time is almost negligible when more than 16 processors are allocated to the program. The same conclusions can be drawn from Figure 2(b): there is a sharp degradation of the observed performance for executions with 32 and 64 processors. Such metrics can then be used to determine the "optimal" number of processors to be assigned to a program in order to achieve a good balance between costs and benefits (performance).

4 The "statistical" module

Once the pre-processing of the measured data has provided the parameters describing the behavior of the parallel programs, various types of statistical analysis have to be applied. They can be simply limited to the computation of basic statistics or they can be as detailed as multidimensional analysis techniques. The main goal of the statistical module is the construction of a synthetic description, that is, a model, of the performance of the programs under study, by means of clustering.

In what follows, we will consider a parallel program as the basic component of the analysis, i.e., the basic workload component. A few runs of the same program or of different programs constitute the workload that will be analyzed.

Figure 3 shows the input specification window for the statistical module. This window is logically subdivided according to the type of the analyses. In the left–hand side, specifications of the basic statistics in preparation to the clustering activity are presented; information related to the cluster analysis are given in the right–hand side. It is possible to require the visualization of the results (e.g., statistics, distributions, pie–charts) of each analysis. All these input specifications are saved into a file, to be selected for running an experiment.

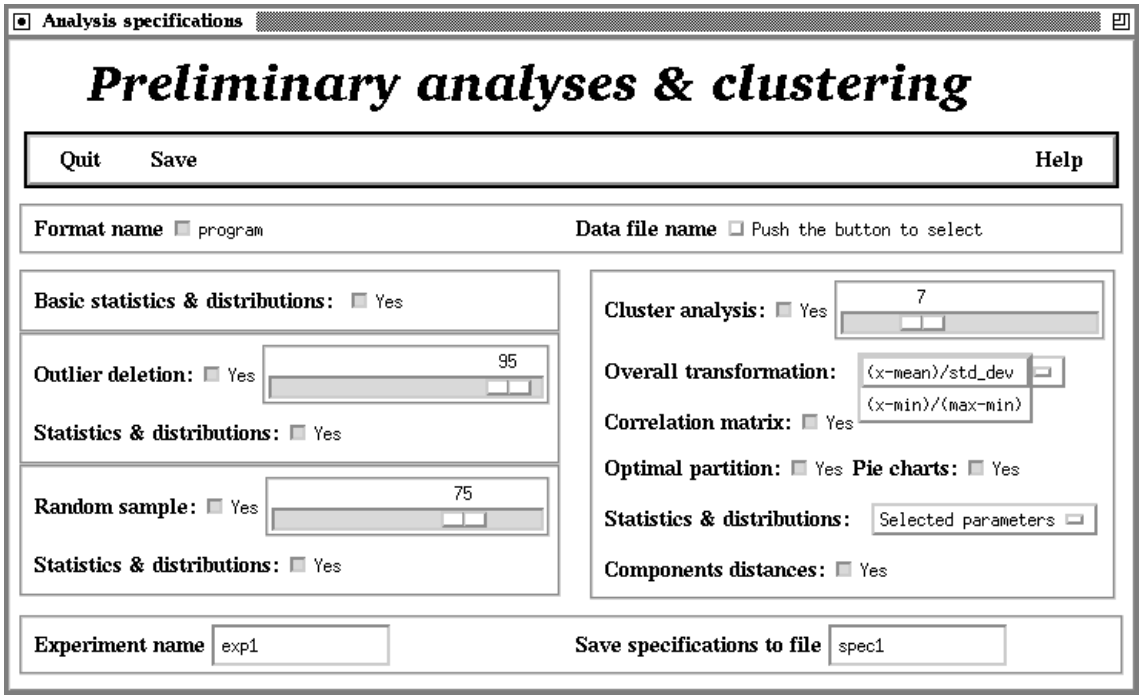


Figure 3: Input specification window for the statistical module.

At a first level, basic statistics, which help in evaluating the properties of the parameters used to define

the workload components, hence in understanding their overall behavior, can be computed. For each parameter, indices, that is, mean, variance, standard deviation, standard error, skewness, and kurtosis, are obtained and displayed in a tabular form. Frequency and cumulative distributions are also presented. Different percentile values can be visualized together with the distributions. All these results can be used as a preliminary step towards the construction of a synthetic model.

For example, the study of the distributions and percentiles and of the basic statistical indices helps in identifying the outliers, which is convenient and advisable to eliminate before constructing the model. Outliers represent those components having one or more parameters values very different from the corresponding parameters values of the other components.

Figure 4 shows the distribution of the message length, expressed in bytes, of an hydrodynamics code executed on 64 processors. As can be seen, the distribution is highly skewed. The median, i.e., the 50th percentile, is equal to 520 bytes. The 80th percentile is 2056 bytes. Both of them are lower than the mean value (equal to 3255 bytes). This leads to the identification of a few very long messages, which can be considered as outliers.

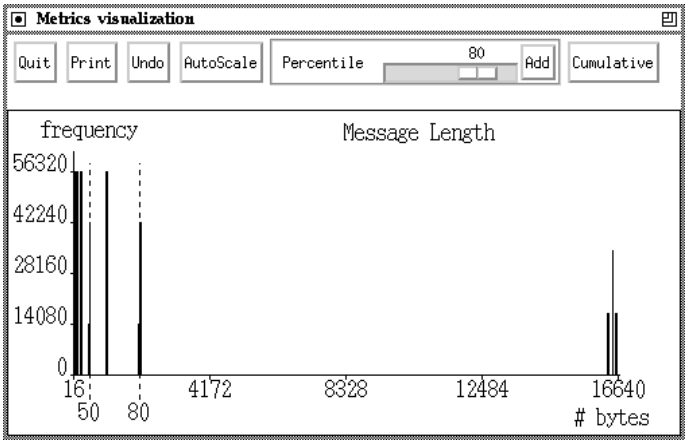


Figure 4: Message length distribution.

When the numbers of components and/or parameters chosen for their description become large, a sample of the workload components may be taken as input to the clustering. The representativeness of the obtained subset is guaranteed by the random sampling technique.

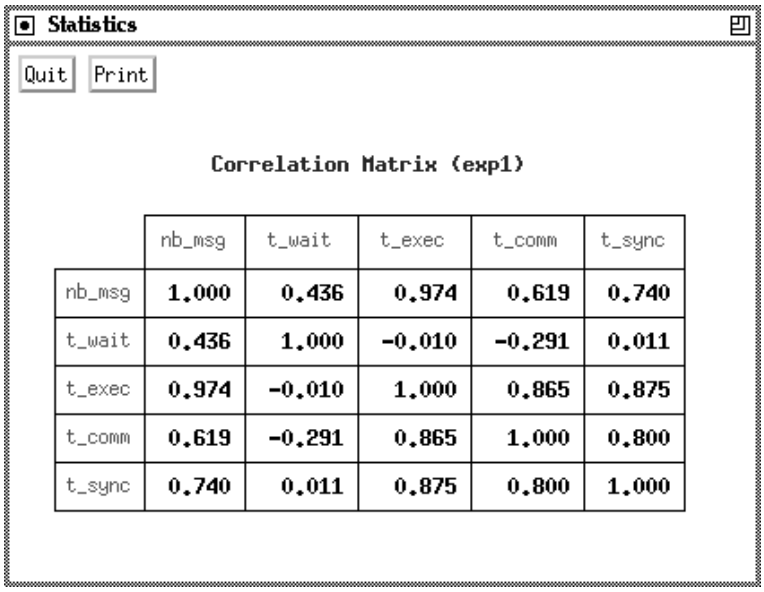
Clustering analysis is then applied to identify classes of components with similar characteristics. For obtaining meaningful comparisons among parameters that can be heterogeneous, it is necessary to scale them in a common range by using an overall transformation (see Fig. 3).

The result of the cluster analysis is a partition of the components into classes, characterized by their centroids, i.e., the geometric centers. The workload components, considered as points in a multidimensional space, are grouped into a given number of classes according to their similarities. The clustering algorithm implemented within MEDEA is the *k–means*, an iterative non–hierarchical method. The components are partitioned by minimizing their distances from the centroid of the class they belong to.

For example, similar patterns in the communication activities performed by the various processors allocated to a program are identified by applying clustering. Typical behavior can also be recognized in the executions of the same program with different number of processors.

As already pointed out in Section 2, various analyses applied to specific measurements are logically organized into a session, which in turn can be subdivided into different experiments. Each time statistical techniques are applied, a new experiment is defined and visualization of the obtained results can be performed on a per experiment basis.

Clustering can be applied by taking into consideration all the parameters which describe a workload component. The correlation matrix, computed within this module, helps in identifying and extracting a subset of the parameters to be used in the following studies. Highly–correlated parameters are usually discarded. Figure 5 shows an example where there is an high correlation between the execution time (`t_exec`) and the number of exchanged messages (`nb_msg`). A new experiment can then be defined, by applying clustering analysis to four out of the five parameters.



	nb_msg	t_wait	t_exec	t_comm	t_sync
nb_msg	1.000	0.436	0.974	0.619	0.740
t_wait	0.436	1.000	-0.010	-0.291	0.011
t_exec	0.974	-0.010	1.000	0.865	0.875
t_comm	0.619	-0.291	0.865	1.000	0.800
t_sync	0.740	0.011	0.875	0.800	1.000

Figure 5: Correlations among five parameters describing a Jacobi program.

5 The "fitting" module

The fitting module has been designed to provide analytical representations of the dynamic behavior of parallel programs. Their internal activities, such as, communications, I/O operations and synchronizations, are captured by these representations. A large variety of studies, dealing with the modelling of parallel program executions benefits of these representations. Examples include global and per-protocol communication profiles, where the number of processors involved in communications activities are expressed as a function of the execution time. Other examples are the communication patterns, e.g., the amount of data exchanged between the processors as a function of the execution time, and the speedup curves, which give information on the degree of parallelism achieved by a program. Metrics, like speedup and execution signatures, can also be fitted to study various aspects of the program behavior, such as, the program scalability.

The fitting module is used in strict conjunction with the filter module which, during the pre-processing phase, extracts from the trace files the measures related to the particular phenomenon under study. Both quantitative and qualitative descriptions are obtained.

The goal of the fitting module is to condense and summarize the experimental measures by deriving an analytical expression. It can be a simple function like a polynomial, a trigonometric, an exponential, or a combination of a few of them. Depending on the objective of the study, it is also possible to use a customized expression.

The idea of the numerical fitting is to minimize, according to the least–squares criterion, the distance between the experimental measures and the analytical expression. Once the fitted function has been com-

puted, it is graphically presented by means of the visualization module of MEDEA. Mathematical details of the analysis, such as the analytical expression of the fitted function, the final residual, the sensitivity and the covariance matrix of the obtained coefficients, are produced in a tabular form. The graphical 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 suggests modifications on the fitting function in order to obtain a better match with the experimental measures. Figure 6 shows an example where both the fitted function and the experimental measures are presented.

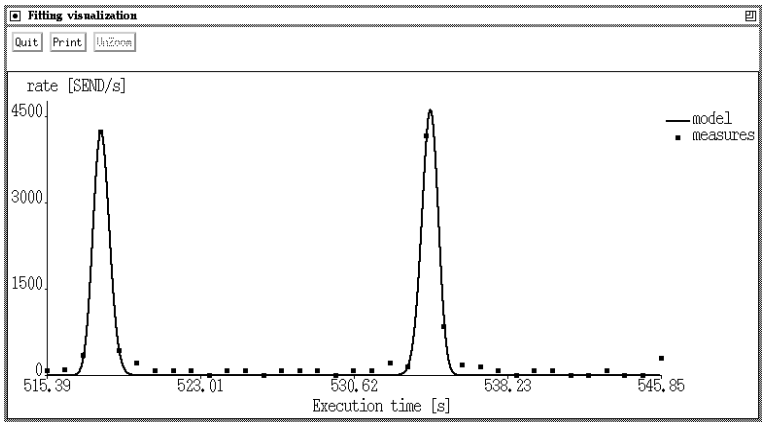


Figure 6: Fitted function (solid line) and experimental measures (dotted curve).

The example is related to the rate of SEND operations issued per second by the allocated processors within an interval of about 31 seconds. The fitted function is a combination of an exponential and a polynomial of degree equal to four. Such a simple function is able to capture the two peaks which characterize the behavior of such communication activities.

6 Conclusions

Appropriate tools for the analysis of the performance of parallel programs are required in order to identify the influence of all the hardware and software components involved in their execution. MEDEA is a tool which studies the behavior of parallel programs by applying a combination of visualization facilities with various types of statistical and numerical techniques. Synthetic and easy–to–interpret information usable in studies dealing with tuning, performance debugging and diagnosis, are obtained. Program developers benefit of this tool for performance analysis of their codes.
Future extensions will be dedicated to an integration of MEDEA within compilation environments. Information about the achieved (or achievable) performance are fundamental for the design of efficient parallel compilers.

References

[1] T. Kerola and H. Schwetman. Monit: A Performance Monitoring Tool for Parallel and Pseudo–Parallel Programs. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 163–172, 1987.

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

[3] B. Ries, R. Anderson, W. Auld, D. Breazeal, K. Callaghan, E. Richards, and W. Smith. The Paragon performance monitor environment. In *Proceedings Supercomputing '93*, pages 850–859, 1993.

[4] D.A. Reed. Experimental Analysis of Parallel Systems: Techniques and Open Problems. In G. Haring and G. Kotsis, editors, *Proc. 7th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 25–51. Springer-Verlag, 1994.

[5] J.C. Yan. Performance Tuning with AIMS - An Automated Instrumentation and Monitoring System for Multicomputers. In *Proceedings of the 27th Hawaii International Conference on System Sciences*, volume II, pages 625–633, 1994.

[6] M.T. Heath and J.A. Etheridge. Visualizing the Performance of Parallel Programs. *IEEE Software*, 8:29–39, 1991.

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

[8] R. Jain. *The Art of Computer System Performance Analysis.* John Wiley & Sons, 1991.

[9] M. Calzarossa and G. Serazzi. Workload Characterization: a Survey. *Proc. of the IEEE*, 81(8):1136–1150, 1993.

[10] 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(3), 1995.

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

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

[13] D. Tessera. The Data Visualization Module of MEDEA - User's Guide. Technical Report CNR "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo" R3/138, Rome, 1994.

[14] 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.

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