MEDEA

A Tool for Workload Characterization of Parallel Systems [1]

Maria Calzarossa, *Senior Member, IEEE*, Luisa Massari, Alessandro Merlo, Mario Pantano[2], Daniele Tessera

Dipartimento di Informatica e Sistemistica
Università di Pavia
Via Abbiategrasso 209
I-27100 PAVIA - Italy
E–mail: {mcc,massari,merlo,pantano,tessera}@gilda.unipv.it

ABSTRACT

Performance of parallel workload is heavily influenced by how its own characteristics match the systems. Experimental approaches are particularly suitable, although difficult, for studying the behavior of the programs. The raw data collected by monitoring tools on real systems need to be processed for obtaining meaningful and easy–to–interpret information usable in identifying sources of poor performance. MEDEA (MEasurements Description, Evaluation and Analysis) is a software tool which meets such objectives. It has been designed as the natural extension of monitors in that it is typically used in strict conjunction with them. Various statistical and numerical techniques are applied to the collected data for deriving compact descriptions of the load, i.e., models, able to capture and reproduce its behavior. Visualization facilities are an integral part of our tool. As applications of MEDEA, a kernel and two real life programs (i.e., climate modelling and turbulent flow) running in different environments are analyzed. Hints about possible code

[1] This work was supported in part by the the Italian Research Council (C.N.R.) under grants N. 93.01592.PF69 and N. 94.00409.CT12 and by the Italian M.U.R.S.T. under the 40% Project.

[2] Author's current affiliation: Institute for Software Technology and Parallel Systems, University of Vienna, Liechtensteinstr. 22, A-1090 Vienna, Austria.

optimizations, most suitable communication policies and data distributions are also discussed.

Keywords: Parallel Workload, Monitoring Tools, Measurements, Statistical Techniques, Performance Analysis, Tuning, Performance Debugging

EXECUTIVE SUMMARY

The performance that can be obtained on parallel systems is strictly dependent on the match between workload and system characteristics. Because of these dependencies, experimental approaches are required. Measurements collected at run–time by monitoring tools have to be processed for selecting the most significant information able to capture the behavior of the workload and to explain its performance. Systematic approaches for the analysis of this large amount of raw data have to be applied.

MEDEA (MEasurements Description, Evaluation and Analysis) is a software tool designed for deriving compact representations of the workload, i.e., models, able to capture and to reproduce its most relevant characteristics. These models are obtained by applying various types of statistical, numerical and visualization techniques provided in such an integrated and friendly environment. Facilities for the automatic pre–processing of the measurements collected on a variety of parallel environments are also part of the tool.

The outcomes of MEDEA represent an important support for studies dealing with tuning, performance debugging, benchmarking and testing of alternative hardware and software system configurations.

As application examples of our tool, a kernel, using the Jacobi relaxation method, and two real life programs, that is, a climate model and a turbulent flow model of stellar plasma, have been analyzed. Such studies have been carried out with different objectives. The influence on the performance of two different data distribution policies adopted by parallelizing compilers has been tested on the Jacobi kernel. The climate model was aimed at evaluating communication protocols as a function of the characteristics of individual parallel systems. The performance debugging studies carried out on the turbulent flow problem have outlined the portions of the code where tuning actions have to be focused.

1 Introduction

Parallel systems consist of many hardware and software components interacting together towards the execution of a workload. A workload, in turn, is constituted by one or more applications (or programs) typically subdivided into a set of tasks which are described by their precedence relationships, synchronization constraints and communication patterns. The performance that can be achieved on parallel systems is related to the match between program and system characteristics. The workload has to be analyzed in details with the aim of investigating and understanding its functional and dynamic behaviors such that bottlenecks and sources of poor performance can be easily identified.

Experimental approaches, based on the analysis of measurements collected on real systems, are well suited, although difficult, for these studies. No assumptions or abstractions about the characteristics of the system itself and of the workload are to be made, as in the case of analytical modelling and simulation. However, extensive monitoring, profiling and measurement activities are to be undertaken.

Many software tools for monitoring parallel programs running in shared memory as well as distributed memory systems have been developed (see e.g., [KS87], [AL90], [LS92], [Worl92]). The Vienna Fortran Compilation System (VFCS) [ZC93] and the Chameleon message–passing communication library [GS93] are also equipped with their own internal monitors. All these tools employ manual or automatic instrumentations of the source codes in order to trap the events of interest. Then, while the workload is executed on a real system, monitoring and profiling take place and the information related to such events is gathered and collected into trace files.

A manual instrumentation of the source codes is required by PICL [Worl92],

a library of routines for message–passing systems which is able to capture interprocessor communications as well as user–defined events. PARMON [LS92] monitors distributed memory systems and includes the facilities for the automatic instrumentation of the programs and for the visualization of their performance.

Post–processing of the trace files produced by monitors is provided by many tools (see e.g., [Casa93]) which support various types of statistical techniques and visualization/animation facilities. These analyses aid in gaining preliminary insights into workload performance. *Ad hoc* routines, which extract information on specific aspects of the workload or of the system itself, are typically used. The lack of integration of such routines limits the scope and the usefulness of these analyses which have to be performed manually, hence representing a very complex job especially for application developers. More systematic approaches are required.

MEDEA (MEasurements Description, Evaluation and Analysis) is a software tool which tries to overcome these drawbacks by addressing the issues related to performance analyses of parallel workload. The tool has been designed with the objective of constructing workload models starting from measurements collected on real systems. MEDEA is seen as the natural extension of the monitors in that its typical use is in strict conjunction with them.

A complete methodology, which integrates a wide range of statistical, numerical and visualization techniques for the analysis of the collected data, is provided. The models obtained with our tool are compact representations of the behavior and of the most relevant characteristics of parallel workload. These models can be used in a variety of performance evaluation studies, e.g., program tuning and performance debugging, testing alternative hardware or software configurations of parallel systems.

The paper is organized as follows. Section 2 describes the basic structure of the tool and the interactions between the various modules it consists of. The analysis techniques employed by each of these modules are briefly presented. Section 3 illustrates application examples based on a kernel and on two real life programs (i.e., climate modelling and turbulent flow). Finally, conclusions and future developments are outlined in Section 4.

2 Structure and Functionalities of the Tool

MEDEA provides a systematic approach for the analysis of parallel workload starting from measurements collected on real systems. It combines, in a user–friendly environment with a graphical interface based on Motif [OSF90], the state of the art of the analysis techniques for workload characterization [CS93], [CM94]. In what follows, these techniques and the basic concepts of workload characterization are briefly described with the aim of pointing out their role and their possible usage within MEDEA. More details about the mathematical foundations can be found in [Hart75] and [FSZ83].

Depending on the objective of the study, the workload can be analyzed at various levels of details, e.g., full applications, tasks, routines, loops, arbitrary code segments. The selected level identifies the workload basic component, and drives the corresponding monitoring activity. For example, an analysis at the task level requires measurements of all the tasks executed within a given application. The instrumentation of the code with the insertion of "probes" related to the events of interest and the monitoring of the workload execution with the collection of such events into trace files have to take place.

Once the trace files have been produced, MEDEA automatically pre–processes them. Indeed, as already pointed out, our tool has been designed as the nat-

ural extension of the monitoring tools. The interactions of MEDEA with monitors and a few of its outcomes are shown in Figure 1.

Traces obtained in various environments, such as Chameleon, PARMON, PICL, VFCS, are currently supported. However, because of the modular structure of MEDEA, routines to process traces produced by other monitors can be easily accommodated within our tool. Furthermore, all the facilities for defining and saving *ad hoc* formats, i.e., record structures, to be used for measurements collected either in parallel, distributed or centralized systems, are provided.

The performance metrics and the workload parameters, selected according to the objective of the study or to some sort of knowledge of the behavior of the workload, are derived and analyzed by means of various types of techniques applied in isolation or in combination with each other. Visualization reveals particularly useful because it provides compact and easy–to–interpret representations of massive parallel workloads. Then, the use of statistical and numerical techniques is required for more detailed studies and for the construction of workload models.

For example, when the events of interest refer to the communication activities among the tasks of an application, the content of a trace file consists of the time stamps related to the beginning and the end of each communication statement, e.g., "send/receive", together with the size and the identifiers of the source/destination of the exchanged messages. The communication profile, the traffic flow matrix, the communication time and the message length are a few of the metrics and parameters derived by the pre–processing of the trace file.

The visualization of common performance metrics, such as profiles (see Fig. 4), speedup (see Fig. 5), and execution signatures, that is, the global execution time of an application as a function of the number of allocated

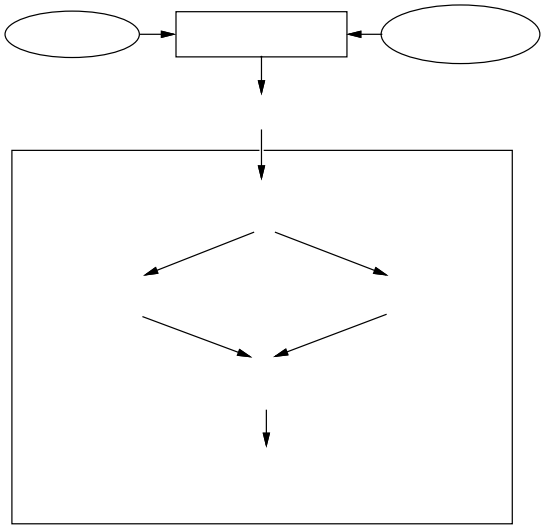


Figure 1: Interactions of MEDEA with monitoring tools.

processors, is also part of the tool. Note that for the visualization of a few of these metrics (e.g., speedup), MEDEA manages multiple trace files obtained varying the number of processors.

When a workload basic component is executed more than once within a single run or on different runs, the set of these observations has to be analyzed. For example, at the task level, the executions of the various tasks spawned by an application are considered. At the application level, the workload consists of multiple independent runs of the application itself. In such a case, the non–determinism inherent in parallel systems is also taken into account. Summaries of statistical indices, such as mean, variance, coefficient of variation, percentiles, concerning the parameters selected for the analysis are computed in order to give preliminary insights on the workload composition. Such values are presented in tabular form. Charts with the density and cumulative distributions of the parameters are also produced for a more intuitive comprehension of the analyzed data (see Fig. 6). Various levels of

zooming over the distributions are also possible to highlight specific portions of the diagrams.

The workload characterization process then requires various phases easily followed because of the modular structure, of the graphical interface and of the on–line help facility provided within MEDEA. Different paths are selected depending on the types of parameters considered and on the objectives of the study. A possible sequence is shown in Figure 2.

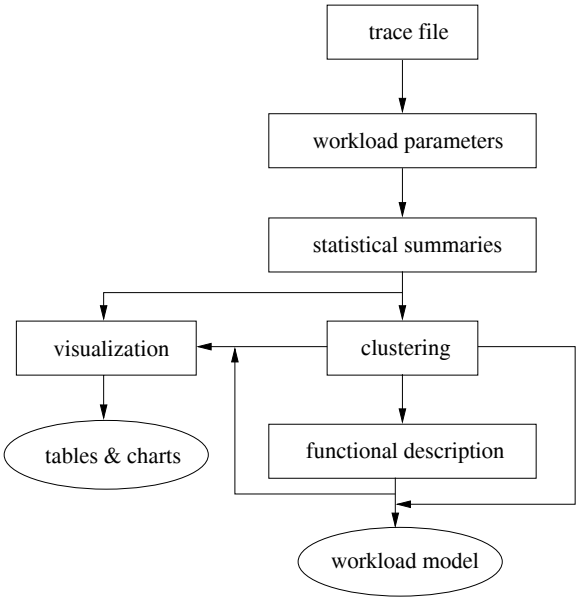


Figure 2: Possible phases followed in a workload characterization process.

A clustering–based approach is adopted when the workload components (e.g., tasks, loops) need to be classified. This multi–dimensional technique groups together "objects" according to some predefined criterion. The *k-means* clustering algorithm [Hart75] adopted within MEDEA partitions the workload components, considered as points in n–dimensional space, according to the Euclidean distance; n denotes the number of parameters selected

for the description of each component. The algorithm minimizes the distances among components belonging to the same group (cluster) and maximizes the distances among components belonging to different groups. Hence, the application of clustering yields data partitions such that components with similar characteristics belong to the same group.

For example, the subdivision of the tasks of an application with respect to communication parameters helps in discovering the existence of bottlenecks and "irregular" communication patterns. "Synthetic" and "artificial" benchmarks benefit of the clustering results for the construction of their workload components.

As part of the clustering module, MEDEA computes the correlation matrix which helps in selecting the most appropriate parameters for the analysis. Indeed, the complexity of these algorithms is related to the number of workload components considered and of parameters used for their description. A coefficient of correlation between two parameters close to 1 suggests to discard one of them since they exhibit similar behavior.

The output of the cluster analysis is presented by the visualization module of MEDEA. Summaries of the basic statistics of the centroids, that is, the geometric centers of the clusters, are provided in tabular forms (see Fig. 3). Moreover, pie charts showing the distribution of the number of components among the groups are displayed.

The results of the clustering can also be analyzed from a functional viewpoint, namely, according to the operations or functions performed by each workload component. In such a case, the identifier of the components (e.g., send, receive, name of the routine) represents their characterizing parameter. The functional description helps in discovering how the various components are subdivided between the groups, that is, which component belongs to which group. The composition of the clusters, expressed either in terms of

the type, i.e., name, and of the relative weight of the components, is presented by means of multiple pie charts (see Fig. 7).

For example, in a cluster characterized by a very large communication time, it is important to precisely identify the tasks exhibiting such a behavior where tuning actions have to be focused. Moreover, the functional description drives the choice of the components for benchmark experiments based on real workloads. A few representatives are easily selected from each group according to their type.

When dynamic aspects of the workload have to be considered, approximation techniques based on numerical fitting are adopted. The behavior of the measured data is captured and reproduced by means of analytic functions which provide compact descriptions of the analyzed phenomena. Various types of functions (e.g., polynomial, exponential, trigonometric, Gaussian) are provided as default by the fitting module of MEDEA. Arbitrary compositions of such functions are also allowed. The measured data together with the fitted function (see Fig. 8) are plotted by the visualization module. The analytic expression of this function as well as a few statistics, such as covariance matrix, residuals, are also given. As an example, the measured data can represent a computation profile, that is, the number of processors simultaneously computing as a function of the execution time of the application. In such a case, the analytic functions produced by the fitting module can be used for studies aimed at predicting the performance of applications under development. Testing alternative processor allocation strategies can also benefit of these models.

3 Application Examples

As applications of our tool, we present a few examples where MEDEA has been employed in studies dealing with analyses of data distributions, evalua-

tion of different communication policies, and performance debugging. A Jacobi kernel (see Sect. 3.1) and two real life programs, namely, a climate model (see Sect. 3.2) and a turbulent flow problem (see Sect. 3.3), have been considered. These programs, which employ different programming paradigms (i.e., data and functional parallelisms), have been tested on various parallel systems with different support environments. Note that although these experimental examples focus on analyses of applications executed on distributed memory systems, MEDEA can also process measurements collected on shared memory and virtual shared memory systems.

3.1 Jacobi kernel

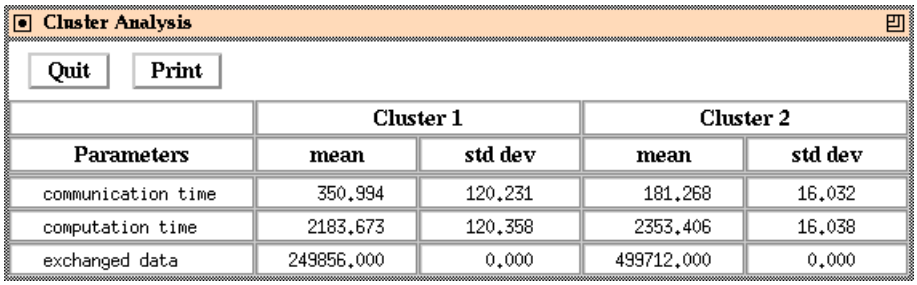
As a first example, a kernel using the Jacobi relaxation method has been analyzed for addressing the problem of automatic data distribution, a hot topic in the development of parallelizing compilers for distributed–memory systems. The aim of this study was to evaluate the impact on program performance of different data distribution strategies.

The kernel, written in Vienna Fortran, has been analyzed for arrays of size 512×512 physically distributed across the allocated processors according to two different policies: a 2D block and a column distribution, respectively. The parallelization and monitoring activities were performed using the facilities provided within the VFCS. During a run of this kernel on 16 processors of an Intel iPSC/860, the activities of each individual processor have been monitored. Eight parameters, namely, the execution, communication, transmit, receive, and computation times and the total volume of exchanged, transmitted, and received data, have been considered.

Statistical summaries and distributions of these parameters provide a first insight into the performance of the kernel. By looking only at the global execution times obtained with the two policies, we have observed that the

2D block distribution exhibits a better performance than the column counterpart. Their execution times are equal to 1707 μs and to 2535 μs, respectively.

In order to refine this result and to assess whether the computation and communication activities are balanced across processors, clustering has been applied. The identification of highly correlated parameters has provided a subset of the previous parameters, namely, the communication and computation times, and the volume of exchanged data to be used in the following analyses. Figure 3 shows the values of the centroids of the groups obtained for the column distribution strategy.



	Cluster 1		Cluster 2	
Parameters	**mean**	**std dev**	**mean**	**std dev**
communication time	350.994	120.231	181.268	16.032
computation time	2183.673	120.358	2353.406	16.038
exchanged data	249856.000	0.000	499712.000	0.000

Figure 3: Centroids of the clusters obtained for the column distribution strategy. Times are expressed in μs and exchanged data in bytes.

More details about the composition of these clusters in terms of the behavior of each processor have been obtained by applying the functional description. The analysis has shown that the first and last processors of the logical one–dimensional array considered belong to Cluster 1 (see Fig. 3). Their activities are characterized by a small volume of exchanged data and by long communication times. This is due to the synchronization delays (included in the communication times) of the data parallel programming model adopted. All the remaining processors exhibit similar behavior between each other.

The same type of analysis performed on the 2D block distribution strategy shows that, although the processors are characterized by a different amount of data to exchange, the performance achieved by each of them is very similar. In this case, the arrays are distributed on a 4×4 mesh of processors. Since, the angular, border, and central processors have to exchange different volumes of data, three groups could have been expected by the clustering. However, such an intuitive result has not been confirmed. The overhead in the communication times due to network contentions and to the routing algorithm of the system alters the intrinsic behavior of the processors. Angular as well as border processors are characterized by a loss of performance. Then, we can conclude that, for the analyzed problem, the 2D block distribution strategy globally achieves better performance than the column distribution counterpart.

3.2 Climate Modelling

The case study[3] described in this section is aimed at the evaluation of a few communication protocols and to assess whether, on different parallel systems, attempts to overlap communications with computations are costly effective. Indeed, parallel programs are very sensitive to changes in the inter–processors communication policies.

A parallel message–passing code that solves the nonlinear shallow water equations on a sphere using the spectral transform method has been used as a test application. Two versions of this code, both exploiting a functional parallelism, are considered. The ringpipe version (algorithm) allows computations and communications to be logically overlapped. The ringsum version isolates the computations from the communications, and is characterized by a sharp logical synchronization between processors. On each target system,

[3] A complete description can be found in [MW94].

namely, the Intel iPSC/860, Touchstone DELTA and Paragon, the code, instrumented by means of PICL, was executed on a logical one–dimensional array of 16 processors. Multiple runs of both algorithms, considered as workload components, have been performed with different communication protocols and number of communication buffers. Each run has been described by five parameters, i.e., execution, computation, communication, transmit, and receive times.

From the automatic processing of PICL traces, MEDEA provides the basic statistics and the distributions of these parameters. Parallel metrics, such as profiles, have also been evaluated and used to obtain a first insight on the behavior of the workload components. Figure 4 shows two phases, extracted from the communication profiles obtained on the Paragon, for a ringpipe and a ringsum algorithm, respectively. The number of simultaneously receiving processors is plotted as a function of the execution time.

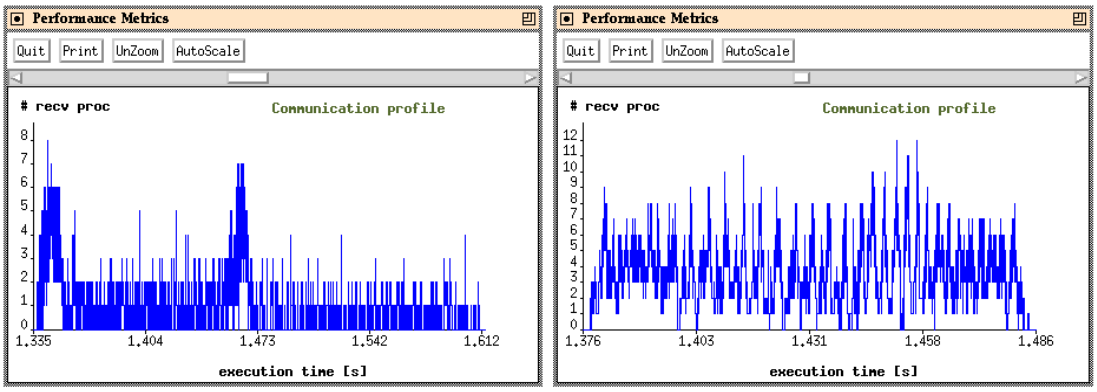


Figure 4: Zooming of one phase of the communication profiles for a ringpipe (a) and a ringsum (b) algorithm executed on sixteen processors of the Paragon.

As can be seen, although both algorithms use the same communication pro-

tocol, the resulting performance is very different. Two subphases are easily identified in the ringpipe case (see Fig. 4 (a)), each starting with a peak in the number of receiving processors. Then, computation intervals are recognized between communication patterns involving only few processors. On the other hand, the ringsum algorithm exhibits a lower level of overlap between communication and computation activities and a larger number of simultaneously receiving processors (see Fig. 4 (b)).

The runs, obtained on each parallel system, have then been analyzed separately using clustering as a first step towards their functional description. For example, four groups have been obtained for the Paragon. Looking at the centroids of these groups, we have noticed that all the components with the shortest communication time (0.178 s) and the longest computation time (3.167 s) belong to one cluster. Functional description has shown that this group contains all the ringpipe algorithms which use the nonblocking receive protocol and extra communication buffers. It also includes the ringpipe implementation based on the nonblocking send – nonblocking receive with forcetype[4] paradigm and no extra communication buffers. As a consequence, we can state that, on the Paragon, the forcetype protocol does not alter the fundamental behavior of the ringpipe algorithm with extra communication buffers. Moreover, extra buffers are not useful for this algorithm when forcetypes are adopted in conjunction with nonblocking sends and receives.

3.3 Turbulent Flow Problem

As a third application example of MEDEA, a real life program which solves the turbulent flow problem of stellar interior plasma[5] has been considered.

[4]The forcetype protocol assumes that a receive has been posted before a send request is made, thus allowing the elimination of some handshaking overhead.

[5]More details on this case study can be found in [CMMT94].

The objective of this study was a detailed analysis of the code aimed at tuning and performance debugging. Even small savings in its execution time are important especially in the case of "Grand Challenge" three–dimensional problems.

The analyzed program uses domain decomposition techniques for the solution of hydrodynamic and of temperature dependent thermal conduction equations. The code, instrumented by means of the Chameleon library, has been tested on two parallel systems, namely, an Intel Paragon and an IBM Sp1. Automatic pre–processing of the trace files collected by Chameleon and stored according to the ALOG format is performed within MEDEA.

In order to have a preliminary idea of the program performance, speedup curves have been derived. In Figure 5, the speedup obtained on the IBM Sp1 for a problem size of 512×256 is shown. The curve exhibits a "knee", cor-

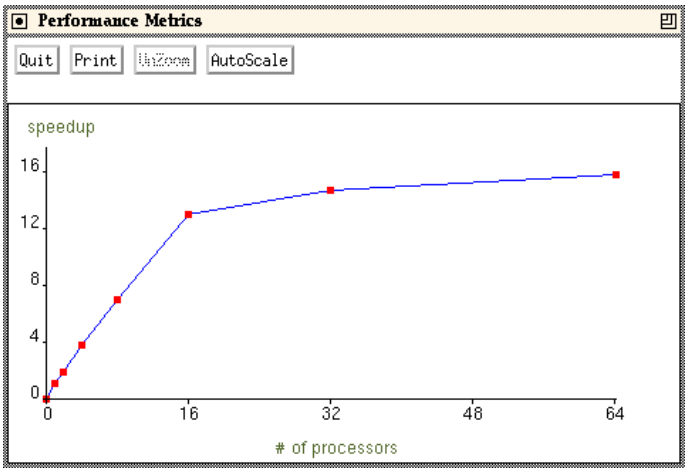


Figure 5: Speedup curve obtainted on the IBM Sp1.

responding to an execution with 16 processors. This is due to a larger

increase of the communication time with respect to its computation counterpart. Note that such phenomenon could not have been explained from a direct analysis of the source code. On the Paragon, due to its architectural characteristics, this behavior and the corresponding knee have been observed for a lower number of processors, namely, eight.

These preliminary results suggest more accurate analyses. Because of the data parallel programming paradigm adopted, the amount of computation of each allocated processor does not change significantly. Hence, the focus has been shifted to the communication activities.

The distributions of the length of the messages exchanged by each processor have been computed. Figure 6 shows an example of such a distribution for a problem size of 512×256 solved on eight processors of the Paragon.

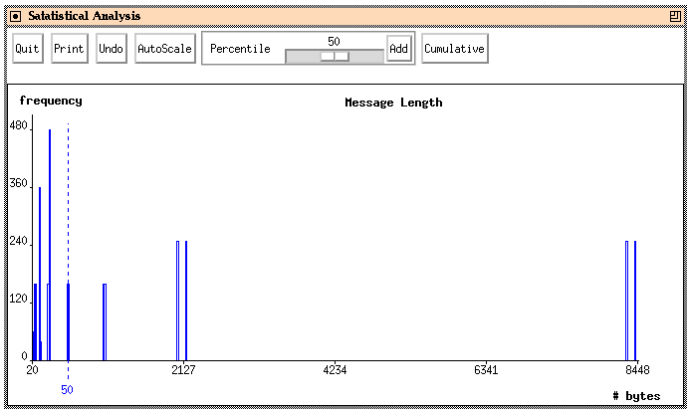


Figure 6: Message length distribution. The dashed line represents the 50th percentile.

As can be seen, the distribution is highly skewed. Indeed, the dashed line corresponding to the 50th percentile (i.e., the median) shows that 50% of the messages are shorter than 520 bytes. Moreover, four or five groups can

be visually identified. To assess the validity of this intuitive result, based on a one–parameter characterization, the analysis has been refined by applying clustering to the times spent by each of the eight processors in the communication events issued during the execution of the application. From the 12606 8–tuple of communication times, the clustering has produced two groups, even though we would have expected to find at least four. Cluster 1 is a "blob" containing almost all the events (99.7%) characterized by a mean communication time of about 1 ms with respect to the tenths of a second of cluster 2. Cluster 2, in turn, consists of the remaining 0.3% workload components, namely, 33, which account for about the 53.5% of the program communication time.

The communication events collected by the Chameleon library have also been analyzed according to their type (e.g., SYNC, NSEND, NRECV) in order to find out the composition of each cluster from a functional viewpoint. Figure 7 shows the weights of the communication events either with respect to the communication time (`t_comm`) they account for, and to their number of occurrences (`# calls`). We have observed that the second cluster (`2/2`) is mainly composed of the events related to the dumping of the program results which involve physical I/Os and require global operations taken over all the processors, such as broadcast communications or synchronizations for the computation of a global minimum (GMIN).

Communication activities have also been studied by looking at their profiles. A compact and more manageable model has been obtained by applying fitting techniques to the 6029 communication events of one of the phases visually identifiable in the communication profile and corresponding to one solution step of the program. The collected measures have been initially smoothed in order to reduce their statistical fluctuations. The analytic model provided by the fitting is a combination of exponential and trigono-

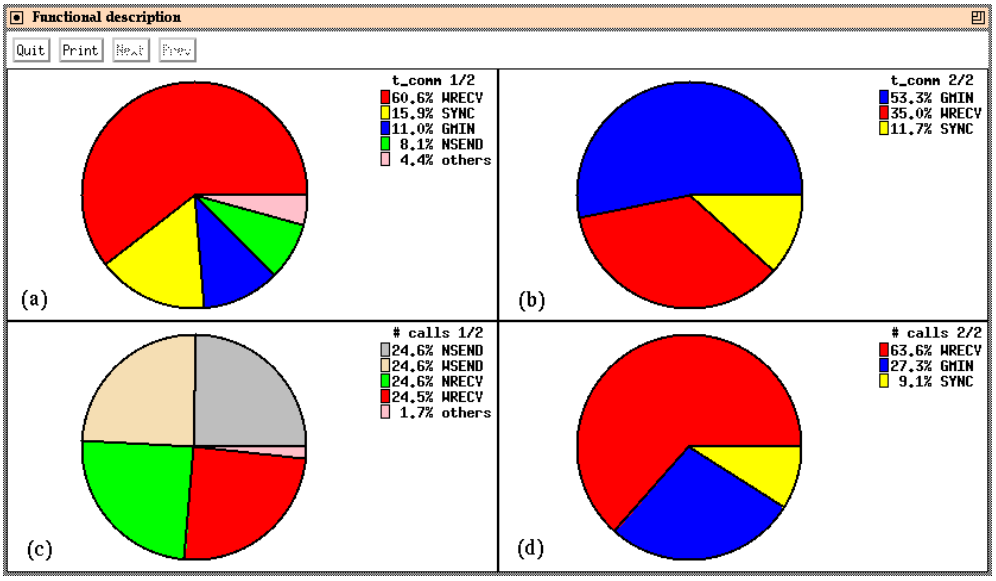


Figure 7: Compositions of the two obtained clusters (`1/2` and `2/2`) in terms of percentages of communication times (upper pies) and of number of calls (lower pies) of each type of communication event.

metric functions. Figure 8 displays the smoothed profile (thin line) and the fitting model (thick line). Such a result can be used, for example, for predicting the communication patterns of this application in the following phases.

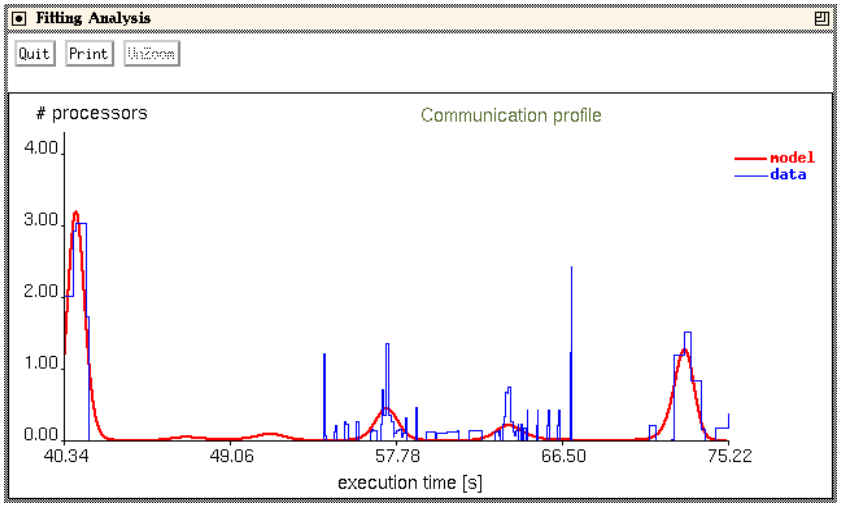


Figure 8: Fitting of one phase of a communication profile.

As a conclusion, these performance analysis and debugging studies have pointed out that the major responsible of the "knee" identified in Figure 5 is the loss of synchronization between processors, which produces large delays. Some processors have to perform extra computations aimed at coordinating the activities of a few others. Hence, the flow of messages is unevenly distributed among processors. Since the program adopts the data parallel paradigm, during the synchronization events, each processor has to wait for the slowest one and a degradation of the overall performance is experimented by the program. As a result, the portions of the code to be optimized in order to avoid such phenomena have been identified.

4 Conclusions

Accurate and detailed analyses of parallel workload are based on experimental approaches. The most relevant information able to capture and to explain its performance has to be selected among the large amount of measurements collected at run–time. Hence, it becomes necessary to have tools which help for the purpose.

MEDEA provides in a user–friendly environment the basic techniques for workload characterization such that a systematic approach can be applied to the raw data produced by monitoring tools. The models obtained with MEDEA are an important support for studies dealing with benchmarking and testing alternative hardware and software system configurations. Performance analysis, debugging and tuning can also largely benefit of these models. Indeed, even small savings in the execution times of parallel programs are very significant, although a few runs, aimed at collecting measurements, may be required to meet such performance goals.

Future work will be dedicated to a tighter integration of MEDEA with parallel compilation systems. Compilers have to choose the "best" strategy for parallelizing the code in order to balance the work distributions among the processors. Automatic analyses of the performance of a few application runs could be used as a basis for the development of self–learning systems embedded within these environments.

Acknowledgement

Authors would like to thank Marco Vidal for his precious help in the implementation of the fitting module of MEDEA.

References

[AL90] T.E. Anderson and E.D. Lazowska. Quartz: A Tool for Tun-

ing Parallel Program Performance. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 115–125, 1990.

[Casa93] T.L. Casavant (Guest Editor). Special Issue on Tools and Methods for Visualization of Parallel Systems and Computations. *Journal of Parallel and Distributed Computing*, 18(2), 1993.

[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.

[CMMT94] M. Calzarossa, L. Massari, A. Merlo, D. Tessera, and A. Malagoli. Performance Debugging of Parallel Programs. In *Proc. AICA Conference*, pages 541–556, Palermo, Italy, 1994.

[CS93] M. Calzarossa and G. Serazzi. Workload Characterization: a Survey. *Proc. of the IEEE*, 81(8):1136–1150, 1993.

[FSZ83] D. Ferrari, G. Serazzi, and A. Zeigner. *Measurement and Tuning of Computer Systems*. Prentice–Hall, 1983.

[GS93] W. Gropp and B. Smith. Users Manual for the Chameleon Parallel Programming Tools. Technical Report ANL-93/23, Argonne National Laboratory, 1993.

[Hart75] J.A. Hartigan. *Clustering Algorithms*. J. Wiley, 1975.

[KS87] 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.

[LS92] P. Lenzi and G. Serazzi. PARMON: PARallel MONitor. Technical Report CNR Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo" n. 3/95, Rome, 1992.

[MW94] A.P. Merlo and P.H. Worley. Analyzing PICL Trace Data with MEDEA. In G. Haring and G. Kotsis, editors, *Proc. 7th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 445–464. Springer-Verlag, 1994.

[OSF90] OSF, editor. *OSF/Motif User's Guide, 5th. Edition*. Prentice-Hall, 1990.

[Worl92] 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.