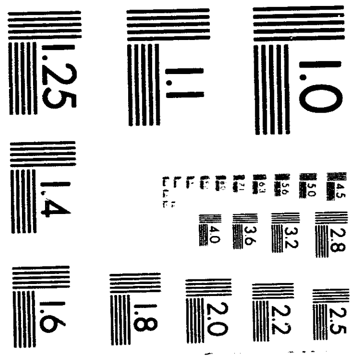

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

1 of 1

ORNL/TM-12506

Engineering Physics and Mathematics Division

Mathematical Sciences Section

ANALYZING PICL TRACE DATA WITH MEDEA

Alessandro P. Merlo [*]
Patrick H. Worley [†]

[*] Dipartimento di Informatica e Sistemistica
University of Pavia
Via Abbiategrasso, 209
I-27100 Pavia
Italy

[†] Oak Ridge National Laboratory
Mathematical Sciences Section
P. O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367

Date Published: November 1993

Research was supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy, by the Italian Research Council (C.N.R.) under Grant 92.01571.PF69 and by the Italian MURST under the 40% and 60% Projects.

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
managed by
Martin Marietta Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

[illegible]

Contents

ANALYZING PICL TRACE DATA WITH MEDEA

Alessandro P. Merlo
Patrick H. Worley

Abstract

Execution traces and performance statistics can be collected for parallel applications on a variety of multiprocessor platforms by using the Portable Instrumented Communication Library (PICL). The static and dynamic performance characteristics of performance data can be analyzed easily and effectively with the facilities provided within the MEasurements Description Evaluation and Analysis tool (MEDEA). This report describes the integration of the PICL trace file format into MEDEA. A case study is then outlined that uses PICL and MEDEA to characterize the performance of a parallel benchmark code executed on different hardware platforms and using different parallel algorithms and communication protocols.

1. Introduction

The demands for hardware and software resources of a computer system significantly influence its performance. Therefore, the quantitative description of resource consumption when running an application plays a fundamental role in every performance evaluation study [2]. The best way to obtain such a quantitative description for a system is to take measurements while the system is processing its real workload. However, the set of data collected by the monitoring tools represents a detailed "discrete" description of the behavior of the measured applications. While such a characterization is very useful when used as input to visualization tools, it is inappropriate when applied to system modeling, where a compact and manageable representation of the workload processed by the real system is needed.

The process of deriving a compact representation of the workload, *workload characterization*, can be subdivided into several phases [11]. The input to the process is the data collected by monitoring the execution of a given application over the system. Output includes both standard data analysis results, which provide useful insights into the behavior of the application, and workload models, which can be used as input to either simulation or analytic system models. How the data is analyzed and how the model is derived are functions of the type of questions being addressed about the performance of the computer system, the type of data collected, and the level of detail at which the analysis will be performed. For example, at some point in the process, the basic unit of work that is considered in a quantitative description of the workload, the *workload component*, must be specified.

While the type of analysis that is appropriate for a particular workload characterization will vary as different questions are asked or different computer systems evaluated, many mathematical techniques are common to a variety of analyses. To support this commonality, and to support the general data exploration process that is common to all workload characterization, researchers at the University of Pavia have developed the MEasurements Description Evaluation and Analysis tool (MEDEA) [15]. The basic aim of MEDEA is to define an integrated environment in which to perform workload modeling studies. The different operations required to fully examine the behavior of the applications submitted to a system have been logically subdivided into *modules*, each performing a specific manipulation over the performance data and the intermediate results produced at each step of the workload characterization process. Figure 1 shows the overall structure of MEDEA and its relationships with other tools used in the workload characterization process.

The collection of performance data is often a difficult task in itself, especially on systems without dependable operating system or hardware support for the collection of useful trace files. One portable option for the collection of performance data for message–passing computer systems is to use the Portable Instrumented Communication Library (PICL), developed at Oak Ridge National Laboratory, when implementing application codes [6]. PICL implements a generic message–passing interface able to support interprocessor communications on a variety of different hardware platforms.

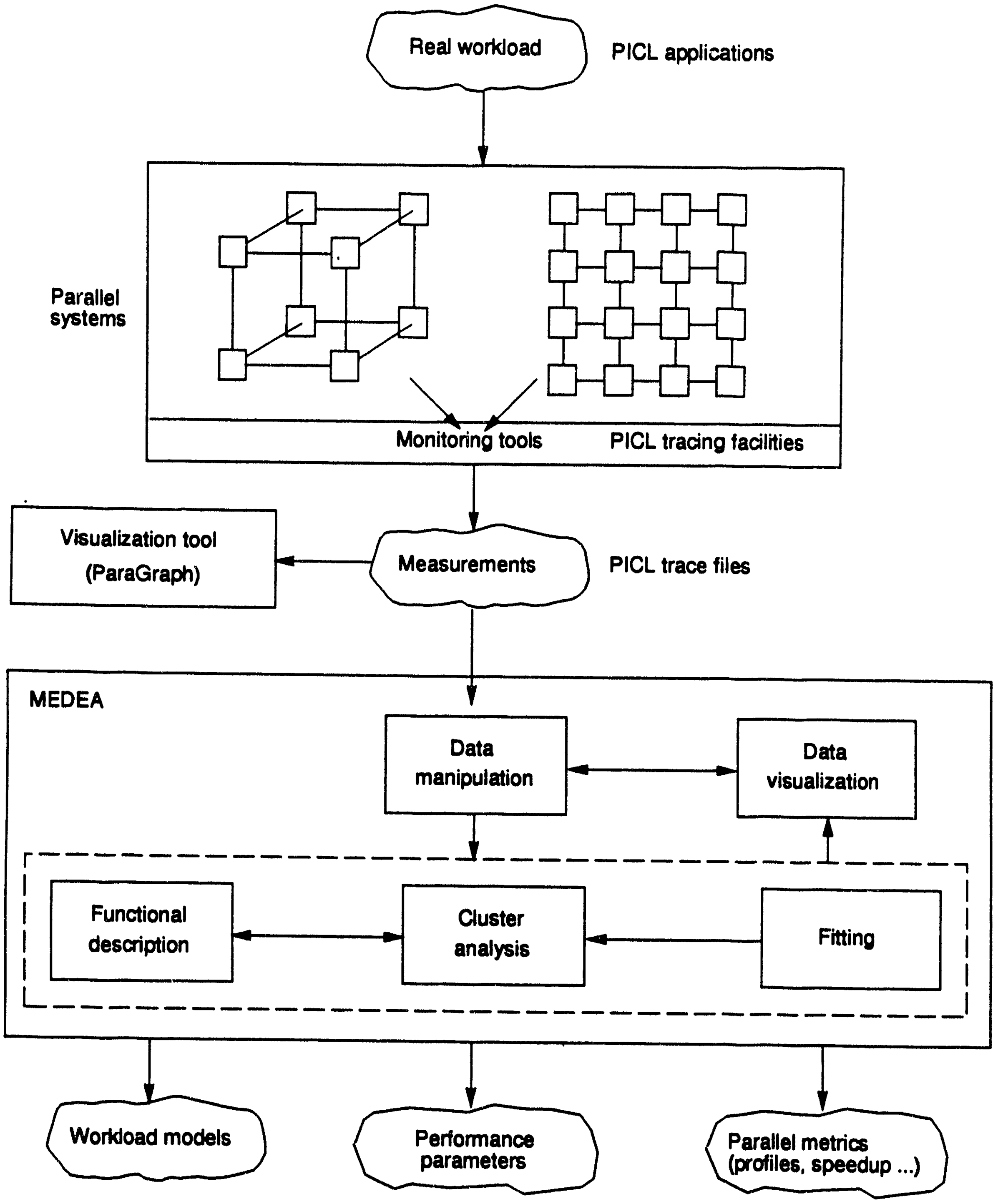


Figure 1: Overall structure of MEDEA and its relationships to other evaluation tools.

Furthermore, PICL tracing routines allow the user to collect detailed information on the behavior and performance of parallel programs. The trace files generated by PICL can be used as input to performance visualization tools, e.g. ParaGraph [8] [9], for performance tuning and debugging, as well as to performance evaluation tools like MEDEA.

This report describes the integration of PICL trace data into MEDEA, as illustrated in Fig. 1, indicating how the static and dynamic characteristics of the workload generated by PICL applications can be analyzed with the facilities provided within MEDEA. Sections 2 and 3 give a brief description of the main features provided within PICL and MEDEA, respectively. Section 4 deals with the integration of PICL and MEDEA: the selection of possible workload components and the specification of the corresponding performance parameters are outlined here. Section 5 outlines an experimental application. A few conclusions are summarized in §6.

2. The Portable Instrumented Communication Library

A detailed performance analysis of a computer system under its real workload can be achieved by means of event–driven monitors, i.e., tools that capture the events generated by a program and store them into trace files. However, the trace file formats adopted by different monitoring tools are, in general, quite different from one another (see, for example, [3], [6], [10], [12]), with each developer defining a specific record format able to address those events of interest for the particular system being evaluated. This lack of standardization makes it difficult to easily analyze trace files collected on different systems, but is a reflection of system differences that cannot simply be eliminated by a standardization process. Recently, there has been a movement toward establishing a standard metaformat in which to specify trace file formats [1]. If this approach is adopted, it will ease one aspect of integrating new types of trace data into tools like MEDEA, but it will not eliminate true semantic differences between the information collected on different systems or with different tools. The integration of new types of trace information will always require careful thought and design.

The PICL trace file format was chosen for integration with MEDEA because of the wide availability and utility of PICL trace files. The machine independent layer of PICL has proven to be a sufficient framework to support portability between different platforms, and the trace file format used by PICL is flexible enough to collect data for performance evaluation. Moreover, while many of the available traces are generated from PICL programs, a significant number are generated directly by other event-monitoring systems or via postprocessing, so that they can be visualized with ParaGraph.

The PICL trace file format was recently significantly modified, to better support the collection of information found to be most useful in visualization tools and in workload characterization, and to be more extensible [16]. The new trace file format has been incorporated into ParaGraph, and is being considered for use in other event monitoring systems. Because of the significant new capabilities of the new format, and because of its adoption outside of PICL, it is the new format that has been integrated with MEDEA.

The basic structure of PICL trace records is shown in Table 1.

Record type [int]	Event type [int]	Timestamp [double]	Processor ID [int]	Task ID [int]	Number of data fields [int]	Data descriptor [int or string]	Data

Table 1: Basic structure of PICL trace records.

Four different record types are currently supported by PICL: *user-defined*, *event*, *statistics*, and *subset-definition*.

- User–defined record types are used to specify the data associated with user–defined events.
- Event record types are used to collect detailed information needed for a visualization tool like ParaGraph or for the analysis of user events by means of MEDEA, as will be explained in §4.1.
- Statistics record types are used to collect profile data of system and user–defined events.
- Subset–definition record types are used to define subsets, e.g., of processors or processes, for which cumulative statistics are to be collected.

The process of workload characterization using PICL trace files is based primarily on the analysis of event and statistics record types. The tracing facilities provided within PICL allow the user to specify the amount and the type of data to store into trace files: if detailed data are needed, then for each event generated by the application, timestamped entry/exit records are stored for the processor and the process associated with the event; if only global information is needed (e.g., when it is not important to know the exact timing of the single events but when we are interested in the corresponding cumulative times), then the statistics records can be used to characterize the general behavior of an application at low overhead, since only these types of data will be collected during the tracing activity.

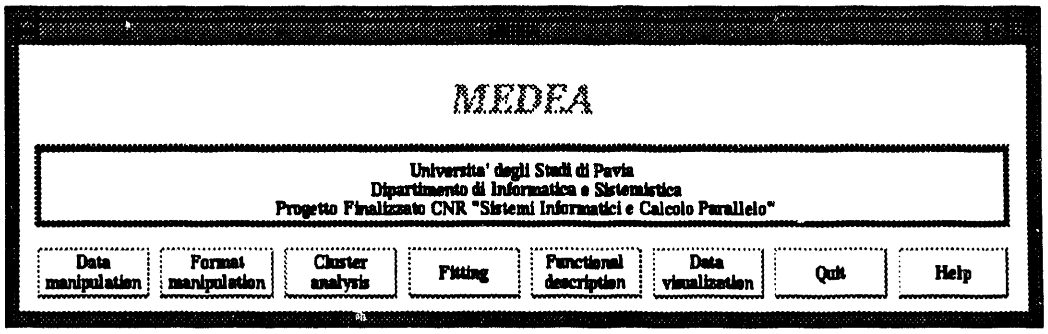
The event types currently supported by PICL cover most of the event data utilized in performance evaluation studies of message–passing parallel applications. The most important categories of these events are *user–defined*, *interprocessor communication*, *I/O*, *synchronization*, *resource allocation*, and *tracing*.

- User–defined events allow the user to specify that the execution of a subroutine or even arbitrary code segments be considered an event of a certain type, allowing the logical structure of the application to be represented during subsequent analysis;
- Interprocessor communication events represent PICL commands for enabling, disabling, or invoking interprocess communications, including, for example, send and receive;
- I/O events are used to collect performance data on (physical) I/O, which strongly influence the performance of most real parallel applications;

- Synchronization events currently supported include "clock normalization" and "barrier";
- Resource allocation events deal with the allocation/deallocation of processors to a given application;
- Tracing events are recorded with the dual goals of allowing a correct interpretation of the trace files and of providing a measure of the overhead implied by the tracing activity itself.

3. The MEasurements Description Evaluation and Analysis tool

The construction of accurate workload models requires the application of different types of statistical and numerical techniques interacting together to fully characterize the behavior of the applications submitted to a system. During the design phase of MEDEA [14], the need for integration between these different underlying techniques and the need for portability across a variety of computer platforms led to the choice of a standard development environment. As a consequence, MEDEA is currently implemented on UNIX systems running X Windows/Motif[1]. Figure 2 shows the main window of the graphical interface provided by MEDEA. Every module specified in the overall structure of the tool (see Fig. 1) can be identified in the active graphical elements of this window.



Universita' degli Studi di Pavia
Dipartimento di Informatica e Sistemistica
Progetto Finalizzato CNR "Sistemi Informatici e Calcolo Parallelo"

Figure 2: Main window of the graphical interface of MEDEA.

Data manipulation module. The *data manipulation* module performs a preliminary analysis of the trace data in order to correlate the events recorded during the execution of an application. Traditional performance indices, such as computation and communication times, and parallel metrics, such as speedup and efficiency, can then be derived by filtering the trace data. Initially, MEDEA required that all trace data be in the format generated by the PARallel MONitor (PARMON), a distributed event-driven monitoring tool [12] for transputer-based MIMD architectures. In order to parse other trace file formats, format-specific trace file analysis facilities must be added to the data manipulation module, as has been done for PICL.

[1]MEDEA requires at least X11R5 and Motif 1.1.4.

Format manipulation module. Within MEDEA, a *format* is a subset of the performance parameters that can be associated with the specific workload component under study. The *format manipulation* module of MEDEA allows user-defined subsets of parameters to be stored in an internal library. As a consequence, repeated workload analyses on different trace files can be performed with fewer interactions with the graphical interface.

Cluster analysis module. The *cluster analysis* module is used to examine the statistical properties of the measured data set. For example, it can be used to identify groups of workload components having homogeneous characteristics with respect to some predefined parameters. The multidimensional clustering algorithm implemented within MEDEA is the *k–means*, an iterative nonhierarchical method of partitioning data sets [7]. Each partition is derived by minimizing the distances between each workload component and the centroid of the cluster it belongs to. At the end of the analysis, the optimal partitions (if any) are derived according to the overall mean square ratios of the evaluated clusters.

Fitting module. As outlined in §1, workload models must be compact and easily manageable. The *fitting* module provided within MEDEA allows the user to derive analytic descriptions of the dynamic behavior of the workload from the measured data. The analytic models are described in terms of one or more of the collected parameters, and are able to represent the variations of the workload parameters with respect to any independent variables, including time.

Functional description module. The process of workload characterization can be approached from two different viewpoints. The *physical* viewpoint describes the behavior of the system and the applications by means of indices related to resource consumptions, such as computation and communication times. This quantitative approach is the one realized by the data manipulation and the cluster analysis modules of MEDEA. The *functional* viewpoint gives a logical description of the workload. In this case, the classification of workload components is based, for example, on the type of applications or on membership of particular components in a specific cluster The *functional description* module of MEDEA deals with the functional viewpoint.

Data visualization module. The graphical visualization of parameter values, derived directly from the trace data or from the results of analyses performed within MEDEA, is often an important tool in understanding the characteristics and the behavior of the workload. The *data visualization* module of MEDEA provides this facility.

4. PICL–MEDEA Integration

MEDEA uses the information stored in PICL trace files to derive models of the workload generated by the measured applications, and this is the only direct dependence MEDEA has on PICL. However, the selection of appropriate workload components and of the corresponding performance parameters is strongly dependent on the type of information collected into the trace files.

Since PICL tracing routines allow the user to specify the level of detail and the amount of data to collect during the execution of an application, the information that can be derived in the workload characterization process may be different from trace file to trace file. If detailed trace files are used as input to MEDEA, then the tool looks for each single event entry/exit pair and, according to the event record type, correlates this new information to the previous ones in order to accumulate statistics that refer to the performance parameters used to characterize the workload components. If trace files are used that contain only statistics records, then MEDEA parses only those records that contain global information. In the following sections, specifications for the possible workload components and the corresponding parameters are given.

4.1. Workload components

The workload submitted to a system may be analyzed at different levels of detail, according to the "granularity" of the components selected for the modeling activity. As mentioned in §1, a workload component is defined as the basic unit of work that is considered in a quantitative description of the workload. Three different approaches (or granularities) have been adopted in MEDEA for the analysis of PICL trace files: ***program–oriented***, ***processor–*** or ***task–oriented***, and ***user–event–oriented.***

In the program–oriented approach, a trace file is analyzed from a global viewpoint and information about the behavior of the application considered as a whole can be derived. The basic workload component is the program itself. The processor–oriented approach derives a more detailed analysis of a trace file, in which the tasks executed on each single processor are selected as representative workload components. (While the programming paradigm supported by PICL, and assumed by ParaGraph, only allows one process per processor, the trace file format can be used to record data from applications with more than one process per processor.) Finally, in the user–event–oriented approach the facility provided within PICL for defining arbitrary code segments to represent distinct workload components allows MEDEA to use the "logical" or "user" view of the application when analyzing its behavior.

4.2. Parallel metrics

Parallel profiles represent one of the best tools for analyzing the dynamic behavior of an application [13]. If detailed PICL trace files are used as input to MEDEA, then the number of processors in use as a function of the execution time can be evaluated with respect to the different types of operations performed by the processors. An example communication profile is shown in Fig. 3.

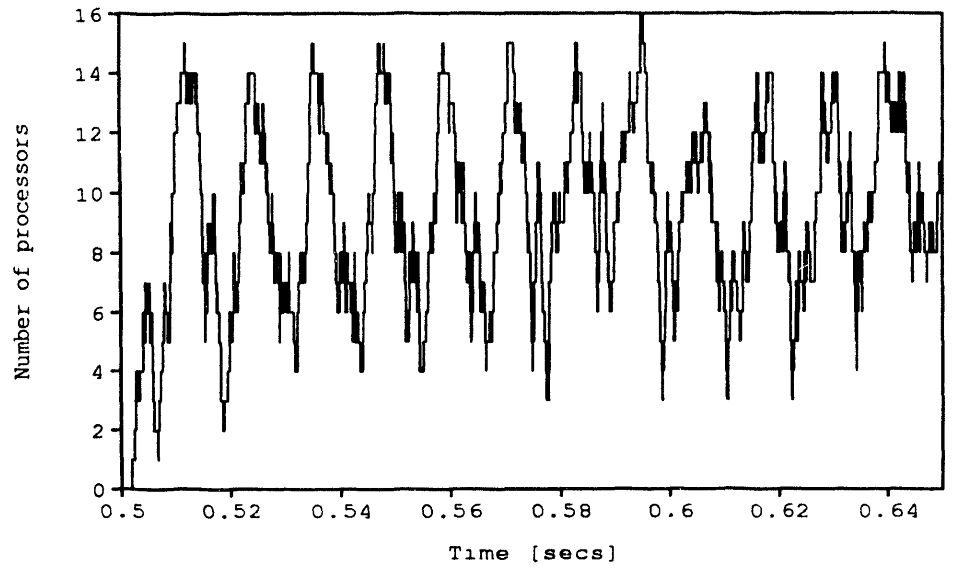


Figure 3: Example of communication profile.

When the performance of an application is measured for a varying number of processors, parallel metrics such as speedup, efficiency, efficacy, and execution signature can be use to characterize the behavior of the workload [5].

Table 2 lists the parallel metrics that can be evaluated by means of the data manipulation module of MEDEA.

Execution profile	I/O profile
Computation profile	Speedup
Communication profile	Efficiency
Receive profile	Efficacy
Transmit profile	Execution signature

Table 2: Parallel metrics evaluated by MEDEA.

4.3. Performance parameters

The selection of meaningful parameters to be considered in the workload characterization phase represents one of the most critical steps of this process. Table 3 lists the parameters that are currently used to characterize the program–oriented and the processor–oriented approaches.

Time parameters			
Execution time	(extime)	I/O time	(iotime)
Computation time	(cptime)	Communication enable/disable time	(iptime)
Communication time	(cmtime)	Synchronization time	(cktime)
Receive time	(rctime)	Resource allocation time	(rstime)
Transmit time	(trtime)	System time	(sytime)
Volume parameters			
Volume of data exchanged	(ttdata)	Volume of transmitted data	(trdata)
Volume of received data	(rcdata)	Volume of I/O data	(iodata)
Occurrence parameters			
Number of receive requests	(rcnum)	Number of I/O requests	(ionum)
Number of transmit requests	(rcnum)	Number of processors	(prnum)

Table 3: Parameters for the program–oriented and the processor-oriented approaches.

Note that these parameters are meaningful only with respect to a particular "granularity" of the workload components. For example, the parameter ***prnum***, representing the number of processors allocated to an application, is not meaningful when applied to a processor–oriented approach, where we are interested in the behavior of each single task. Furthermore, according to the workload component selected, we can have different interpretations for the same parameter. As an example, consider the case of the computation time (parameter *cptime*). If the single task has been selected to be the workload component, then the value of this parameter can be calculated as the sum of all the time intervals during which a computation is performed. This time represents the difference between the total time of the application on the specific processor and the total time spent by this processor while executing noncomputational instructions (e.g., send/receive requests, synchronization commands, etc.). Alternatively, in the case of the program–oriented approach, *cputime* can be defined only as a mean with respect to the number of processors.

Table 4 lists the parameters currently used within MEDEA to characterize the user–event–oriented approach.

Time parameters			
Total event time	(ctime)	User events time	(utime)
System events time	(stime)	Hidden system events time	(hstime)
Occurrence parameters			
Number of event occurrences	(cnum)	Number of hidden system events	(hsnum)
Number of system events	(snum)	Number of hidden user events	(hunum)
Number of user events	(unum)		

Table 4: Parameters for the user–event–oriented approach.

These parameters differ from those adopted for the other two approaches. In the following discussion, we use the trace records in Tab. 5 to explain the meaning and usage of the parameters. Here, the first field in each record denotes an event entry (-3) or an event exit (-4), the second field denotes the event type id, and the third field denotes the timestamp for the record. The other fields can be ignored for the following discussion. System events have types ids less than -10, and user events have nonnegative type ids. The indentation in Tab. 5 has been introduced to indicate nesting of events, and neither the indentation nor the timestamp labels, e.g., **(timestamp a)**, reflect what PICL would produce.

```
-3 0 0.000016 6 0 2 2 0 0                 (timestamp a)
   -3 -52 0.000128 6 0 1 2 0              (timestamp b)
   -4 -52 0.000516 6 0 3 2 8 0 0          (timestamp c)
   -3 1 0.000711 6 0 2 2 0 0              (timestamp d)
      -3 -52 0.000818 6 0 1 2 1           (timestamp e)
      -4 -52 0.001643 6 0 3 2 8 1 5       (timestamp f)
      -3 -21 0.001665 6 0 3 2 8 1 7       (timestamp g)
      -4 -21 0.001711 6 0 0               (timestamp h)
      -3 2 0.001982 6 0 0                 (timestamp i)
      -4 2 0.002005 6 0 0                 (timestamp j)
   -4 1 0.002013 6 0 0                    (timestamp k)
-4 0 0.002067 6 0 0                       (timestamp l)
```

Table 5: Example trace records.

In PICL applications, user–defined events can correspond to any arbitrary code segment. As a consequence, the presence of nested user events is very common, especially if the user events are associated with the execution of program subroutines. With respect to the example trace records in Table 5, two nested events (of types 1 and 2) can be recognized within the "main" event of type 0.

When these PICL trace records are analyzed according to the user–event–oriented approach, the following meanings and values are assigned to the identified parameters for user events of type 0.

- *total event time* is the elapsed time between the entry record for a type 0 event (**timestamp a**) and the corresponding exit record (**timestamp l**) if the event type occurs once, or is the sum of the elapsed times if it occurs multiple times:

$$ctime \;=\; 0.002067 - 0.00016 \;=\; 0.002051 \;\; \text{secs} \; .$$

- *system events time* is the sum of the execution times of any system events that are nested at the first level of type 0 events (a type -52 event starting at **timestamp b**):

$$stime \;=\; 0.000516 - 0.000128 \;=\; 0.000388 \;\; \text{secs} \; .$$

- *user events time* is the time spent executing user events nested at the first level of type `0` events (one type `1` event):

$$utime = 0.002013 - 0.000711 = 0.001302 \text{ secs} .$$

- *hidden system events time* is the time spent to execute system events that are detected in nested user events (type `-52` and type `-21` events nested in a type `1` event):

$$hstime = (0.001643 - 0.000818) + (0.001711 - 0.001665) = 0.000871 \text{ secs} .$$

- *number of event occurrences* is the number of times type `0` events have been executed on a given processor: $cnum = 1$.
- *number of system events* is the number of system events that are nested at the first level of type `0` events (a type `-52` event starting at **`timestamp`** `b`): $snum = 1$.
- *number of user events* is the number of user events that are nested at the first level of type `0` user events (one type `1` event): $unum = 1$.
- *number of hidden system events* is the number of system events occurring within nested user events (type `-52` and type `-21` events beginning at timestamps `e` and `g`, respectively, and nested within a type `1` event): $hsnum = 2$.
- *number of hidden user events* is the number of user events nested within user events at the first level (one type `2` event): $hunum = 1$.

5. A Case Study

This section outlines a workload characterization study that uses MEDEA to analyze PICL trace data. The study is presented to illustrate how MEDEA can be utilized to analyze PICL trace data, what types of analyses are possible, and, hopefully, how useful the insights available from the analysis are. In consequence, the emphasis in the exposition is on the experimental methodology. While preliminary results from the study are mentioned at the end of the section, the analysis of the data is ongoing. The complete analysis will be presented in a later report.

The application used for the study is PSTSWM, a message–passing benchmark code and parallel algorithm testbed that solves the nonlinear shallow water equations on a sphere [17]. This code models closely how CCM2, the Community Climate Model developed by the National Center for Atmospheric Research, handles the dynamical part of the primitive equations. PSTSWM was developed to compare parallel algorithms and to evaluate multiprocessor architectures for parallel implementations of CCM2.

PSTSWM uses the spectral transform method to solve the shallow water equations. During each timestep, the state variables of the problem are transformed between the physical domain, where most of the physical forces are calculated, and the spectral domain, where the terms of the differential equation are evaluated. The physical domain is a tensor product longitude-latitude grid. The spectral domain is the set of spectral coefficients in a spherical harmonic expansion of the state variables.

Transforming from physical coordinates to spectral coordinates involves first performing a fast Fourier transform (FFT) for each line of constant latitude, generating results on a wavenumber-latitude grid. This is followed by integration over latitude for each line of constant wavenumber, approximating the Legendre transform (LT). The inverse transformation involves evaluating sums of spectral harmonics and inverse FFTs, algorithmically analogous to the forward transform.

Parallel algorithms are used to compute the FFTs and to compute the vector sums used to approximate the forward Legendre transforms. Processors are treated as a two dimensional grid, with the longitude dimension mapped onto row processors and the latitude dimension mapped onto column processors. Thus, the specified aspect ratio determines how many processors are allocated to computing the FFTs and the LTs. Many different parallel algorithms are embedded in the code, and the choice of algorithms is determined via input parameters at runtime.

In this study, variants of two parallel algorithms to compute the forward Legendre transforms are compared. Both parallel algorithms are based on (1) computing local contributions to the vector of spectral coefficients, (2) summing the "local" vectors element-wise over a logical ring of processors, and (3) broadcasting the result to the members of the ring. Both algorithms send $P-1$ (equal-sized) messages per processor to compute the global sum and $P-1$ messages to implement the broadcast, where P is the number of processors in a processor column. Each message in the summation is sent to the logical right neighbor, while each message in the broadcast is sent to the logical left neighbor. The algorithms differ in when the three stages are executed. The first algorithm, ***ringsum***, first computes all local contributions, then computes the global sum, and finally broadcasts the results. The second algorithm, ***ringpipe***, interleaves the calculation of the local contribution with the global summation in a pipeline fashion, and interleaves the broadcast with the computation that uses the result, also in a pipeline fashion. Thus, the ringsum algorithm isolates the communication from the computation, preventing communication and computation from interfering with each other and (more) effectively synchronizing the processors in the interprocessor communication. The ringpipe algorithm allows the communication and computation to be overlapped, and requires less memory than ringsum. The question addressed by the study is whether attempting to overlap communication with computation is cost effective on a given architecture.

To address the question, PSTSWM was executed on four different platforms: the Intel iPSC/2, iPSC/860, Touchstone DELTA, and Paragon machines. The Intel iPSC/2 and iPSC/860 systems are distributed memory, hypercube–connected parallel architectures [4]. The processor elements are the Intel i80286/387 and the Intel i860, respectively. The communication hardware, based on bit–serial channels, is the same for both the systems. The Intel Touchstone DELTA and Paragon systems are distributed memory, wormhole–routed, mesh–connected parallel architectures. The processor elements are the Intel i860 and the Intel i860SP, respectively.

Table 6 summarizes the main features of the parallel systems we used for this study.

Hardware configurations				
	iPSC/2	iPSC/860	DELTA	Paragon
Interconnect	hypercube	hypercube	mesh	mesh
CPU type	80386/387	860	860	860SP
Clock rate	16MHz	40MHz	40MHz	50MHz
Memory/node	4MB	8MB	16MB	16MB

Table 6: Hardware characteristics of the architectures for the experimental study.

5.1. Measurements

On each architecture, PSTSWM was executed on a logical 1x16 mesh topology, calculating each FFT sequentially and each LT in parallel. Multiple runs were made using both ringsum and ringpipe algorithms, with varying implementations of the algorithms, underlying communication protocols, and number of communication buffers. The following naming convention identifies a given experiment:

```
<application_name>.<algorithm_type>.<protocol_option>.<buffering_option>
```

A guideline for the interpretation of trace file names is as follows:

Algorithm type. Each stage of both algorithms is characterized by sending data to one neighbor, receiving data from another, and using the data to update a running sum. The following options differ in the order of these operations.

- ringpipe:
  1) type 00: calculate local contribution (calc)/sum/send/receive
  2) type 01: calc/sum/send/receive or calc/sum/receive/send
  3) type 02: calc/receive/sum/send
- ringsum:
  1) type 10: send/receive/sum
  2) type 11: send/receive/sum or receive/send/sum
  3) type 12: same as 10, but posting receive requests early
  4) type 13: same as 11, but posting receive request early

Algorithms of type 01, 11, and 13 implement a conservative protocol, where the odd numbered processors in the logical ring send first and receive second, and the even numbered processors receive first and send second. This protocol works even when system buffer space is limited, and will also work on systems supporting only synchronous communication. Algorithms of type 12 and 13 use nonblocking receive requests to indicate where messages should be stored when they arrive.

Protocol option. On Intel multiprocessors, PICL supports both blocking and nonblocking communication requests and both regular and forcetype communication protocols. In blocking requests, control does not return to the calling process until the corresponding operation is complete. In nonblocking requests, control returns immediately, and further inquiries are required to determine when the corresponding operation is complete.[2] The forcetype protocol assumes that a receive request has been posted at the destination processor before a send request is made at the source, thus allowing the elimination of some handshaking overhead, but it requires that the user insure that this condition holds.

1) type 0: blocking send – blocking receive
2) type 1: nonblocking send – blocking receive
3) type 2: blocking send – nonblocking receive
4) type 3: nonblocking send – nonblocking receive
5) type 4: blocking send – nonblocking receive with forcetypes
6) type 5: nonblocking send – nonblocking receive with forcetypes
7) type 6: blocking synchronous send/receive (for algorithms of type 01, 11, and 13)

Protocol option type 6 uses extra handshaking messages to guarantee that messages are not sent until the corresponding receive requests have been posted. This simulates what occurs when using synchronous communication requests.

Buffering option. When nonblocking receives and/or sends are used and extra buffer space is available, some of the receive requests can be posted "early" and some sends completed "late", potentially eliminating system buffer copying overhead and allowing additional communication and computation to be overlapped.

1) type 0: use no extra communication buffers
2) type x: use the maximum number of extra communication buffers

Example. As an example, the trace file **pstswm.02.3.0** refers to the execution of PSTSWM using the ringpipe algorithm with the computational paradigm "calc/[send/receive | receive/send]", assuming the "nonblocking send – nonblocking receive" communication protocol and no extra communication buffers.

[2]Note that a send request on Intel multiprocessors is complete when the buffer containing the message being sent can be altered without altering the message, and does not indicate that the message has been received by the destination process.

5.2. Preliminary analysis: performance parameters and parallel metrics

We executed PSTSWM on the parallel systems described in §5, varying the algorithm type and the protocol and buffer options. From each execution, we collected a detailed trace file using the tracing facilities provided by PICL. As outlined in §4.1, these trace files can be analyzed at different levels of detail, according to the granularity of the workload components selected, which, in turn, is a consequence of the objectives of the analysis. We evaluated the usefulness of overlapping communications and computation using the program–oriented approach: each trace file was analyzed by MEDEA from a global viewpoint and a subset of the performance parameters described in Tab. 3 were used to characterize the behavior of the application. In our study, total execution, computation, communication, receive and transmit times were selected as representative parameters. Their values were used as input to MEDEA, and the parallel metrics described in §4.2 were used to obtain a first insight into the dynamic behavior of the application runs.

As an example of the differences between the experimental runs, Fig. 4 and 5 show the receive profiles derived from the execution on the Intel Paragon for `pstswm.02.4.x` and `pstswm.12.4.0`. The first is a ringpipe algorithm based on a "calc/receive/sum/send" execution paradigm. It uses the "blocking send – nonblocking receive with forcetypes" communication protocol options and the maximum number of extra communication buffers. This algorithm maximizes the opportunity for overlap of communication and computation phases. `pstswm.12.4.0` is a ringsum algorithm based on a "send/receive/sum" execution paradigm. It uses the same communication protocol as the ringpipe example, but without any extra communication buffers. The protocol option minimizes the overhead of interprocessor communication for any given send/receive pair, but the algorithmic options do not attempt to interleave the communication and computation or to eliminate all system buffer copying.

Figures 6 and 7 give a detailed view of the first communication phase shown in the receive profiles of these algorithms. Note that two subphases, summation and broadcast, can be easily identified for the ringpipe example (Fig. 6): each phase starts with a peak in the number of receiving processors, corresponding to the early posting of nonblocking receive requests by the single tasks, and then contains communications patterns involving a small number of processors at any one time as the explicit handshaking required when using forcetypes takes place. These patterns are separated by computation intervals during which no communication is performed at all, i.e. the sequential FFTs. In the ringsum example (Fig. 7), there is only one peak when the summation/broadcast has been started, and then an almost continuous communication pattern can be identified as the messages move around the logical ring.

5.3. Workload characterization

The behavior of each experimental run is represented by a single point in a five–dimensional space, as determined by the number of performance parameters selected in §5.2. In our study, the statistical properties of this data set have been examined by means of the cluster analysis and the functional description modules of MEDEA for each multiptocessor platform,

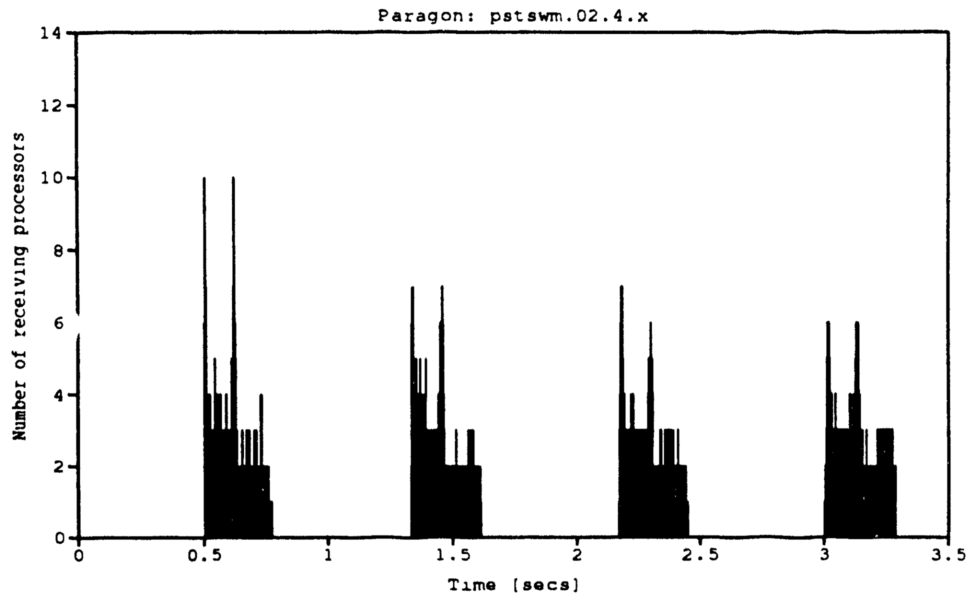


Figure 4: Receive profile for the **pstswm.02.4.x** ringpipe algorithm.

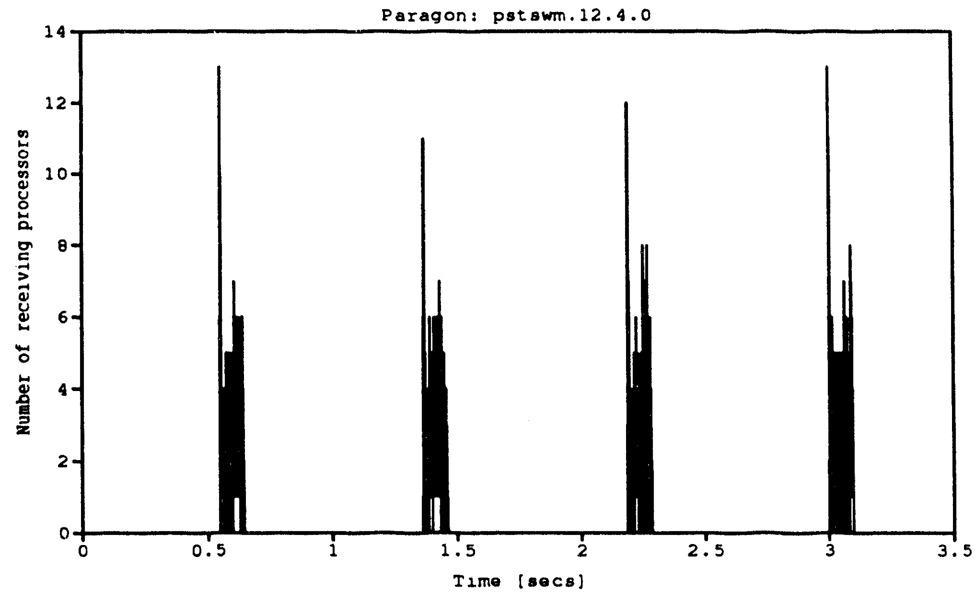


Figure 5: Receive profile for the **pstswm.12.4.0** ringsum algorithm.

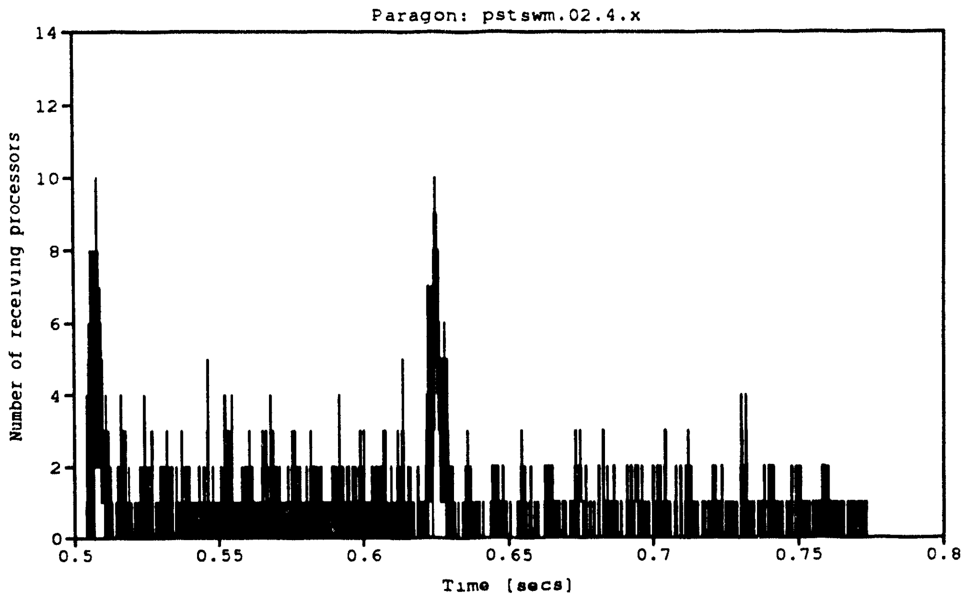


Figure 6: Detailed view of the first communication phase for the **pstswm.02.4.x** ringpipe algorithm.

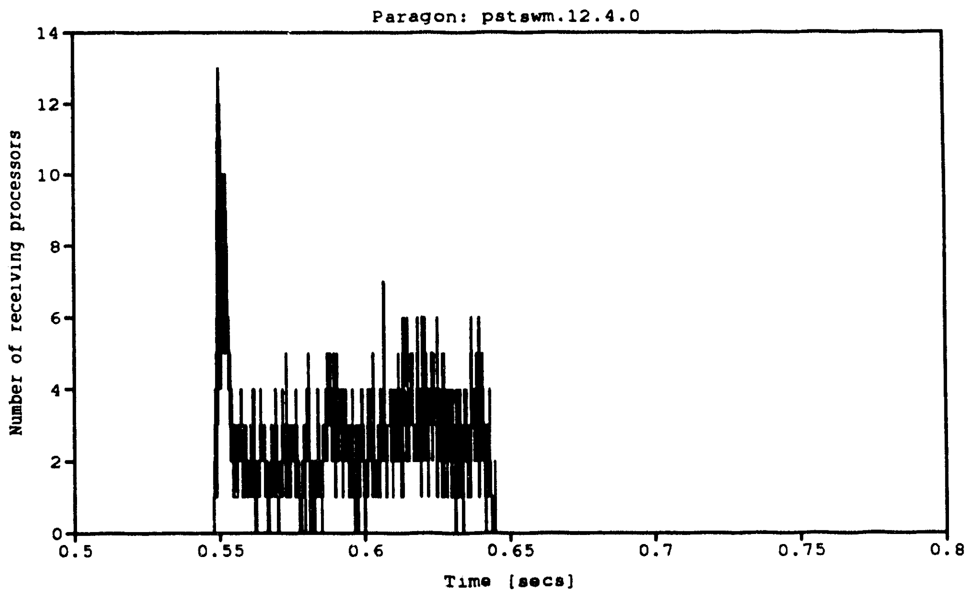


Figure 7: Detailed view of the first communication phase for the **pstswm.12.4.0** ringsum algorithm.

5.3.1. Workload models

As mentioned previously, the behavior of a real workload is very complex and difficult to reproduce, and the amount of information collected into trace files is, in general, difficult to manage. In consequence, system studies usually require that a model, or simplified characterization, of the workload be constructed. Even though the execution of a workload is usually a deterministic phenomenon, it is often modeled as a nondeterministic one, with statistical techniques being applied. As outlined in §3, MEDEA classifies processes in preparation for construction of workload models by means of the *k–means* clustering algorithm. In order to make meaningful comparisons between the performance parameters selected for our analysis, the values of these parameters are first scaled so that they lie in a common interval. Then the partitioning of workload components is derived.

In our study we analyzed 56 trace files (corresponding to the execution of PSTSWM for the different implementations of the ringsum and the ringpipe algorithms and varying the underlying communication protocol and the number of communication buffers) for each architecture. The following tables summarize the optimal partitions of the workload components with respect to the overall mean square ratios of the evaluated clusters.

The means of the execution, computation and communications times represent the values for the centroid of the corresponding cluster. They can be used, together with the standard deviations, as input to either analytic or simulation system models to reproduce the behavior of real workload.

Note that these experiments are part of a larger exercise in determining optimal algorithm parameters for problems that will be used on the largest configurations of each multiprocessor. To capture the right granularity when running on only 16 processors, the problem sizes were scaled. Thus, there is some difference between the different sets of experiments, and raw timings cannot be compared between the multiprocessors. The timings for the DELTA and the Paragon do represent the same problem though, and can be compared.

5.3.2. Functional description

The composition of each cluster has also been investigated from a functional viewpoint.

We constructed a preliminary characterization by projecting the experimental runs onto a subspace identified by two of the selected parameters. Figure 8 shows the projections of the ringsum and the ringpipe algorithms within the *extime–cptime* subspace for experiments run on the Paragon (see Table 10). While the first and second cluster can be easily identified, the remaining partitions do not have well defined shapes. This indicates that the extime and cptime parameters are insufficient to characterize the workload generated by algorithms belonging to the third and to the fourth cluster. The relationships that characterize the last two partitions involve the whole subset of performance parameters the cluster analysis was based on. For example, if we consider the projections within the *rctime–trtime* subspace (see Fig. 9), the third and fourth clusters are well shaped too.

		Extime		Cptime		Cmtime	
Cluster	Percentage	mean	std dev	mean	std dev	mean	std dev
Cluster 1	17.0%	139.577	0.561	137.659	0.225	1.712	0.797
Cluster 2	47.2%	144.483	0.350	141.450	0.169	3.013	0.334
Cluster 3	35.8%	142.571	0.247	137.376	0.167	5.514	0.303

Table 7: Workload model for the trace files collected on iPSC/2.

		Extime		Cptime		Cmtime	
Cluster	Percentage	mean	std dev	mean	std dev	mean	std dev
Cluster 1	1.8%	16.158	0.000	12.836	0.000	3.321	0.000
Cluster 2	28.6%	19.788	0.355	13.089	0.218	6.690	0.380
Cluster 3	19.6%	17.849	0.840	13.307	0.324	4.525	0.706
Cluster 4	3.6%	15.463	0.686	13.869	0.159	1.585	0.846
Cluster 5	46.4%	19.802	0.481	12.904	0.099	6.891	0.467

Table 8: Workload model for the trace files collected on iPSC/860.

		Extime		Cptime		Cmtime	
Cluster	Percentage	mean	std dev	mean	std dev	mean	std dev
Cluster 1	20.2%	5.227	0.147	3.947	0.069	1.278	0.139
Cluster 2	34.8%	5.849	0.169	4.185	0.140	1.663	0.136
Cluster 3	15.7%	5.835	0.126	4.756	0.181	1.074	0.191
Cluster 4	20.2%	5.679	0.139	4.205	0.152	1.471	0.140
Cluster 5	9.0%	5.152	0.159	3.947	0.042	1.204	0.129

Table 9: Workload model for the trace files collected on DELTA.

		Extime		Cptime		Cmtime	
Cluster	Percentage	mean	std dev	mean	std dev	mean	std dev
Cluster 1	49.1%	3.372	0.053	3.028	0.029	0.344	0.344
Cluster 2	33.3%	3.866	0.039	3.033	0.033	0.833	0.049
Cluster 3	8.8%	3.920	0.128	3.035	0.028	0.885	0.110
Cluster 4	8.8%	3.345	0.011	3.167	0.042	0.178	0.045

Table 10: Workload model for the trace files collected on Paragon.

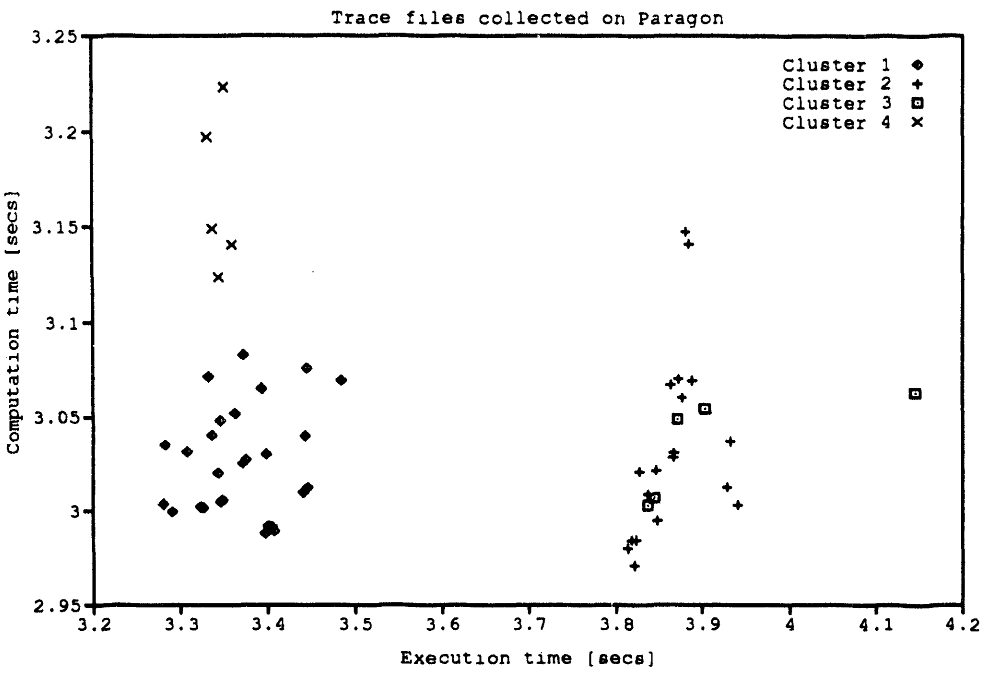


Figure 8: Projections within the *extime–cptime* subspace.

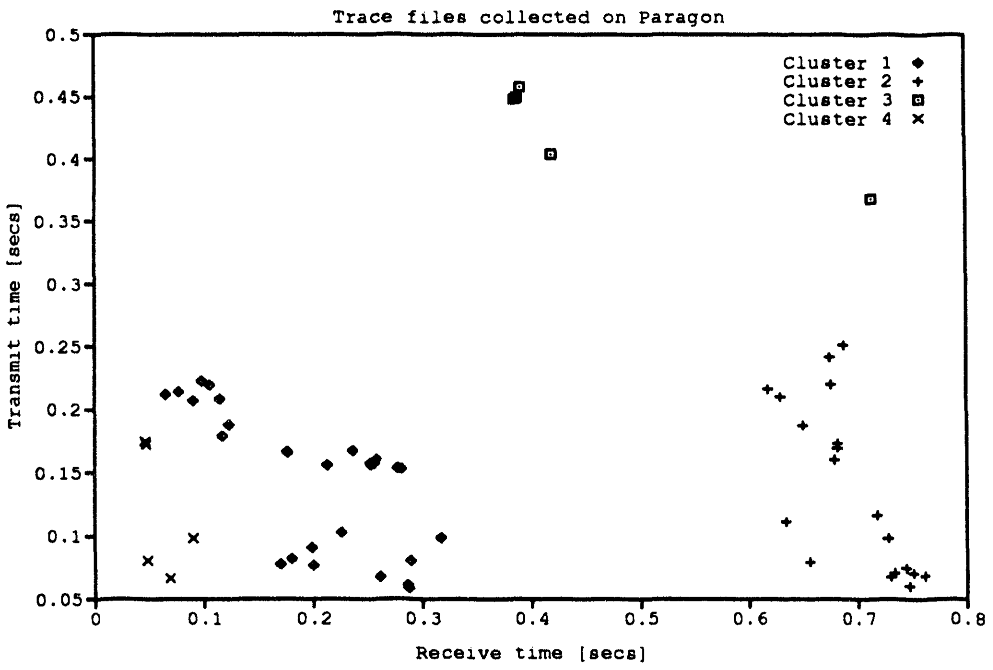


Figure 9: Projections within the *rctime–trtime* subspace.

As a second step in our functional characterization process, the components belonging to a specific cluster can be listed in order to obtain better insights into the model of the workload being evaluated. As an example, Table 11 lists the applications grouped into the fourth cluster of the workload model for the Paragon.

Paragon: cluster 4	
`pstswm.02.2.x`	`pstswm.02.5.0`
`pstswm.02.3.x`	`pstswm.02.5.x`
`pstswm.02.4.x`	

Table 11: Composition of the fourth cluster of the workload model for the Paragon.

Note that all the components belonging to this cluster correspond to trace files derived from ringpipe algorithms based on the "send/calc/receive" execution paradigm. Furthermore, this cluster groups together those PSTSWM runs utilizing nonblocking receive communication protocols and extra communication buffers. The cluster also includes the experiment utilizing nonblocking send – nonblocking receive communications with forcetypes and no extra communication buffers (`pstswm.02.5.0`). These results imply that the forcetype protocol does not change the fundamental behavior of this algorithm when using extra communication buffers, but that extra buffers are unnecessary (on the Paragon, using this algorithm) when forcetypes are used with nonblocking sends and receives.

5.3.3. Results

This case study has important implications on how these multiprocessors should be used. The preliminary results confirm that the utility of overlap varies across the platforms. Moreover, the techniques required to productively exploit overlapping communication with computation also vary between the architectures, even though their programming models are identical. For example, overlap is useful, and simple to characterize and exploit, on the iPSC/2. It is even more important for efficiency on the iPSC/860, but is more difficult to utilize effectively. Techniques maximizing the possibility of overlap have a marginal utility on the Touchstone DELTA, and it is doubtful whether overlap is the reason for the efficiency. The performance analysis on the Paragon currently changes with every operating system upgrade, but its performance characteristics, with regard to exploiting overlap, seem to lie between the those of the Touchstone DELTA and the iPSC/860. We are currently quantifying these observations with further experiments and analysis, and will report on the results in a future report.

6. Conclusions

In this report we described the integration of the Portable Instrumented Communication Library (PICL) trace file format into the MEasurements Description Evaluation and Analysis tool

(MEDEA). This integration was motivated by the wide availability and utility of PICL trace files, and by the capabilities in MEDEA for easily analyzing the static and dynamic characteristics of parallel workloads from trace data. We also described a workload characterization study, to indicate exactly how PICL data can be analyzed using MEDEA. In our initial experiences in using MEDEA to analyze PICL trace files, we have found the combination of these tools to be effective and powerful in workload characterization studies.

Our experiments on the Paragon also point out the utility of having portable tools like PICL and MEDEA. While the Paragon will have a full suite of performance monitors and tools in future releases of the system software, they were not available for these experiments. This is typical in the analysis of early or experimental systems. It is important to understand the performance of early systems quickly, and PICL and MEDEA allow us to measure and analyze performance on these systems without depending on the availability of vendor–supplied tools.

7. Acknowledgments

In addition to the research programs directly funding this work, we acknowledge the Computer Hardware, Advanced Mathematics, and Model Physics (CHAMMP) program of the U.S. Department of Energy for supporting the development of the benchmark code PSTSWM and the collection of the performance data used in this research.

This research was performed in part using Intel iPSC/2, iPSC/860, and Paragon multiprocessor systems at Oak Ridge National Laboratory (ORNL). The iPSC/2 is owned by the University of Tennessee, and both it and the IPSC/860 are managed by the Mathematical Sciences Section of ORNL. The Paragon is managed by the Center for Computational Sciences at ORNL.

The research was also performed using the Intel Touchstone Delta System operated by Caltech on behalf of the Concurrent Supercomputing Consortium. Access to this facility was provided by Argonne National Laboratory (ANL). We gratefully acknowledge the help of Brian Toonen at ANL in collecting the data on the Delta.

8. References

[1] R. A. AYDT, *The Pablo self-defining data format*, tech. report, Department of Computer Science, University of Illinois at Urbana-Champaign, March 1992.

[2] M. CALZAROSSA AND G. SERAZZI, *Workload Characterization: A Survey*, Proceedings of the IEEE, 81 (1993).

[3] T. H. DUNIGAN, *A Message-passing Multiprocessor Simulator*, Technical Report ORNL/TM-9966, Oak Ridge National Laboratory, Oak Ridge, TN, 1986.

[4] ——, *Performance of the Intel iPSC/860 and nCUBE 6400 Hypercubes*, Technical Report ORNL/TM-11790, Oak Ridge National Laboratory, Oak Ridge, TN, April 1991.

[5] D. EAGER, J. ZAHORJAN, AND E. LAZOWSKA, *Speedup Versus Efficiency in Parallel Systems*, IEEE Transactions on Computers, 38 (1989), pp. 408–423.

[6] G. A. GEIST, M. T. HEATH, B. W. PEYTON, AND P. H. WORLEY, *A User's Guide to PICL: A Portable Instrumented Communication Library*, Technical Report ORNL/TM-11616, Oak Ridge National Laboratory, Oak Ridge, TN, August 1990.

[7] J. A. HARTIGAN, *Clustering Algorithms*, John Wiley, 1975.

[8] M. T. HEATH AND J. A. ETHERIDGE, *Visualizing the performance of parallel programs*, Technical Report ORNL/TM-11831, Oak Ridge National Laboratory, Oak Ridge, TN, May 1991.

[9] M. T. HEATH AND J. A. ETHERIDGE, *Visualizing the performance of parallel programs*, IEEE Software, 8 (1991), pp. 29–39.

[10] V. HERRARTE AND E. LUSK, *Studying parallel program behavior with upshot*, Technical Report ANL/TM-91/15, Argonne National Laboratory, Argonne, IL, August 1991.

[11] R. JAIN, *The Art of Computer System Performance Analysis*, John Wiley & Sons, New York, 1991.

[12] P. LENZI AND G. SERAZZI, *PARMON: PARallel MONitor - User's Guide Release 1.0*, Technical Report R3/95, University of Milan, October 1992.

[13] M. R. LEUZE, L. W. DOWDY, AND K. H. PARK, *Multiprogramming a distributed-memory multiprocessor*, Concurrency: Practice and Experience, 1 (1989), pp. 19–33.

[14] A. MERLO, *MEDEA: MEasurements Description Evaluation and Analysis tool - User's Guide Release 1.0*, Technical Report R3/117, Progetto Finalizzato C.N.R. "Sistemi Informatici e Calcolo Parallelo", Roma, Aprile 1993.

[15] A. MERLO AND P. ROSSARO, *MEDEA: Design Document*, Technical Report R3/92, Progetto Finalizzato C.N.R. "Sistemi Informatici e Calcolo Parallelo", Roma, Settembre 1992.

[16] P. H. WORLEY, *A New PICL Trace File Format*, Technical Report ORNL/TM–12125, Oak Ridge National Laboratory, Oak Ridge, TN, October 1992.

[17] P. H. WORLEY AND I. T. FOSTER, *PSTSWM: A Parallel Algorithm Testbed and Benchmark Code for Spectral General Circulation Models*, Technical Report ORNL/TM–12393, Oak Ridge National Laboratory, Oak Ridge, TN. (in preparation).

INTERNAL DISTRIBUTION

1.	B. R. Appleton	17.	T. H. Rowan
2.	T. S. Darland	18–22.	R. F. Sincovec
3.	J. J. Dongarra	23–27.	R. C. Ward
4.	J. B. Drake	28–32.	P. H. Worley
5.	T. H. Dunigan	33.	Central Research Library
6.	G. A. Geist	34.	ORNL Patent Office
7.	J. A. Kohl	35.	K-25 Applied Technology Library
8.	M. R. Leuze	36.	Y-12 Technical Library
9.	D. R. Mackay	37.	Laboratory Records - RC
10.	C. E. Oliver	38–39.	Laboratory Records Department
11.	G. Ostrouchov		
12–16.	S. A. Raby		

EXTERNAL DISTRIBUTION

40. Donald M. Austin, 6196 EECS Building, University of Minnesota, 200 Union Street, S.E., Minneapolis, MN 55455

41. Robert G. Babb, Oregon Graduate Center, CSE Department, 19600 N.W. Walker Road, Beaverton, OR 97006

42. David H. Bailey, NASA Ames, Mail Stop 258-5, NASA Ames Research Center, Moffet Field, CA 94035

43. Henri E. Bal, Vrije Universiteit, Department of Mathematics and Computer Science, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

44. Gianfranco Balbo, Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, I-10149 Torino, Italy

45. Edward H. Barsis, Computer Science and Mathematics, P. O. Box 5800, Sandia National Laboratory, Albuquerque, NM 87185

46. Adam Beguelin, Carnegie Mellon University, School of Computer Science, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890

47. Robert E. Benner, Parallel Processing Division 1413, Sandia National Laboratories, P. O. Box 5800, Albuquerque, NM 87185

48. Philippe Berger, Institut National Polytechnique, ENSEEIHT, 2 rue Charles Camichel-F, 31071 Toulouse Cedex, France

49. Donna Bergmark, 745 E & TC Building, Hoy Road, Cornell University, Ithaca, NY 14853

50. Maurelio Boari, Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy

51. Ansgar Boehm, Department of Computer Science III, University of Erlangen/ Nüremberg, Martensstrasse. 3, D-8520 Erlangen, Germany

52. Don Breazeal, Intel Supercomputer Systems Division, 15201 N.W. Greenbrier Pkwy, CO1-01, Beaverton, OR 97006

53. Roger W. Brockett, Harvard University, Pierce Hall, 29 Oxford Street Cambridge, MA 02138

54. James C. Browne, Department of Computer Sciences, University of Texas, Austin, TX 78712

55. Greg Burns, Trollius Project Leader, Academic Computing, The Ohio State University, 1224 Kinnear Rd., Columbus, OH 43212

56. Bill L. Buzbee, Scientific Computing Division, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307

57. Thomas A. Callcott, Director, The Science Alliance Program, 53 Turner House, University of Tennessee, Knoxville, TN 37996

58. Maria Calzarossa, Dipartimento di Informatica e Sistemistica, Università Degli Studi di Pavia, Via Abbiategrasso 209, I-27100 Pavia, Italy

59. Brian M. Carlson, Computer Systems Research Institute, University of Toronto, Toronto, Ontario M5S 1A1, Canada

60. Ron Casselman, Systems and Computer Engineering, Carleton University, Ottawa, Ontario K1S 5B6, Canada

61. Jagdish Chandra, Army Research Office, P. O. Box 12211, Research Triangle Park, NC 27709

62. Doreen Y. Cheng, Principal Engineer, Computer Science Corporation, NASA Ames Research Center, MS 258-6, Moffett Field, CA 94035

63. Melvyn Ciment, National Science Foundation, 1800 G Street N.W., Washington, DC 20550

64. Dennis Cottel, Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA 92152-5000

65. Alva Couch, Department of Computer Science, Tufts University, Medford, MA 02155

66. Tom Crockett, ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, VA 23665-5225

67. Ron Daniel Jr., Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, United Kingdom

68. Michel Dayde, Institut National Polytechnique, ENSEEIHT, 2 rue Charles Camichel-F, 31071 Toulouse Cedex, France

69. Craig Douglas, IBM T. J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598-0218

70. Larry Dowdy, Computer Science Department, Vanderbilt University, Nashville, TN 37235

71. Donald J. Dudziak, Department of Nuclear Engineering, 110B Burlington Engineering Labs, North Carolina State University, Raleigh, NC 27695-7909

72. Dannie Durand, Bellcore 2D-335 , 445 South Street, Morristown, NJ 07962

73. Derek Eager, Department of Computer Science and Engineering, Sieg Hall, FR-35, University of Washington, Seattle, WA 98195

74. Stanley Eisenstat, Department of Computer Science, Yale University, P. O. Box 2158 Yale Station, New Haven, CT 06520

75. Jean-Marc Fellous, Center for Neural Engineering, University of Southern California, Los Angeles, CA

76. Edward Felten, Department of Computer Science, University of Washington, Seattle, WA 98195

77. Charles Fineman, Ames Research Center, Mail Stop 269/3, Moffet Field, CA 94035

78. Jon Flower, Parasoft Corporation, 2500 E. Foothill Boulevard, Suite 205, Pasadena, CA 91107

79. Anthony Ford, Meiko Limited, 650 Aztec West, Bristol BS12 4SD, United Kingdom

80. Ian Foster, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

81. Geoffrey C. Fox, NPAC, 111 College Place, Syracuse University, Syracuse, NY 13244-4100

82. Chris Fraley, Statistical Sciences, Inc., 1700 Westlake Avenue N, Suite 500, Seattle, WA 98119

83. Joan M. Francioni, Computer Science Department, University of Southwestern Louisiana, Lafayette, LA 70504

84. Offir Frieder, George Mason University, Science and Technology Building, Computer Science Department, 4400 University Drive, Fairfax, Va 22030-4444

85. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650

86. Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47401

87. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

88. Ian Glendinning, University of Southampton, Department of Electronics and Computer Science, Southampton, SO9 5NH United Kingdom

89. Gene Golub, Computer Science Department, Stanford University, Stanford, CA 94305

90. Andy Grant, Computer Graphics Unit, Manchester Computing Centre, University of Manchester, Oxford Rd, Manchester M13 9PL, United Kingdom

91. William D. Gropp, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

92. Eric Grosse, AT&T Bell Labs 2T-504, Murray Hill, NJ 07974

93. John L. Gustafson, Ames Laboratory, 236 Wilhelm Hall, Iowa State University, Ames, IA 50011-3020

94. Robert M. Haralick, Department of Electrical Engineering, Director, Intelligent Systems Lab, University of Washington, 402 Electrical Engineering Building, FT-10, Seattle, WA 98195

95. Gunter Haring, Department of Applied Computer Science, University of Vienna, Lenaugasse 2/8, A-1080 Vienna, Austria

96. Ann H. Hayes, Computing and Communications Division, Los Alamos National Laboratory, Los Alamos, NM 87545

97. Michael T. Heath, National Center for Supercomputing Applications, 4157 Beckman Institute University of Illinois, 405 North Mathews Avenue, Urbana, IL 61801-2300

98. Stephen Hellberg, Department of Electronics and Computer Science, University of Southampton, S09 5NH, United Kingdom

99. John L. Hennessy, CIS 208, Stanford University, Stanford, CA 94305

100. N. J. Higham, Department of Mathematics, University of Manchester, Gtr Manchester, M13 9PL, England

101. Dan Hitchcock, Office of Scientific Computing, ER-7, Applied Mathematical Sciences, Office of Energy Research, U.S. Department of Energy, Washington, DC 20585

102. Charles J. Holland, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332

103. Leah H. Jamieson, School of Electrical Engineering, Purdue University, West Lafayette, IN 47907

104. Gary Johnson, Office of Scientific Computing, ER-7, Applied Mathematical Sciences, Office of Energy Research, U.S. Department of Energy, Washington, DC 20585

105. Lennart Johnsson, Thinking Machines Corporation, 245 First Street, Cambridge, MA 02142-1214

106. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309

107. Malvyn Kalos, Cornell Theory Center, Engineering and Theory Center Building, Cornell University, Ithaca, NY 14853-3901

108. Kenneth Kennedy, Department of Computer Science, Rice University, P.O. Box 1892, Houston, TX 77001

109. Tom Kitchens, Office of Scientific Computing, ER-7, Applied Mathematical Sciences, Office of Energy Research, U.S. Department of Energy, Washington, DC 20585

110. Peter A. Krauss, Lehrstuhl fuer Rechnergestuetztes Entwerfen, Technische Universitaet, Muenchen, Germany

111. Domenico Laforenza, Parallel Processing Group, CNUCE - Istituto del CNA, Via Santa Maria 36, 56100 Pisa, Italy

112. Michael Langston, Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301

113. Richard Lau, Office of Naval Research, Code 111MA 800 Quincy Street, Boston Tower 1, Arlington, VA 22217-5000

114. Robert L. Launer, Army Rese^rch Office, P. O. Box 12211, Research Triangle Park, NC 27709

115. E. D. Lazowska, Department of Computer Science and Engineering, Sieg Hall, FR-35, University of Washington, Seattle, WA 98195

116. James E. Leiss, Rt. 2, Box 142C, Broadway, VA 22815

117. Eric Leu, Swiss Federal Institute of Technology, Department of Computer Science, Operating System Laboratory, IN - Ecublens, CH-1015 Lausanne, Switzerland

118. Ted Lewis, Research Director, Oregon Advanced Computing Inst., 19500 NW Gibbs Boulevard #101, Beaverton, OR. 97006

119. Heather M. Liddell, Center for Parallel Computing, Department of Computer Science and Statistics, Queen Mary College, University of London, Mile End Road, London E1 4NS, England

120. Rik Littlefield, Pacific Northwest Laboratory, MS K1-87, P.O.Box 999, Richland, WA 99352

121. Ivo de Lotto, Dipartimento di Informatica e Sistemistica, Università Degli Studi di Pavia, Via Abbiategrasso 209, I-27100 Pavia, Italy

122. Ewing Lusk, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, MCS 221 Argonne, IL 60439-4844

123. Allen D. Malony, Department of Computer and Information Science, University of Oregon, Eugene, OR 97403

124. James McGraw, Lawrence Livermore National Laboratory, L-306, P. O. Box 808, Livermore, CA 94550

125–129. Alessandro Merlo, University of Pavia, Dip. di Informatica e Sistemistica, Via Abbiategrasso, 209, I-27100 PAVIA, Italy

130. Paul C. Messina, Mail Code 158-79, California Institute of Technology, 1201 E. California Boulevard, Pasadena, CA 91125

131. Neville Moray, Department of Mechanical and Industrial Engineering, University of Illinois, 1206 West Green Street, Urbana, IL 61801

132. Richard Muntz, Computer Science Department, University of California at Los Angeles, Los Angeles, CA 90024

133. David Nelson, Director, Office of Scientific Computing, ER-7, Applied Mathematical Sciences, Office of Energy Research, U.S. Department of Energy, Washington, DC 20585

134. Randolph Nelson, IBM, P.O. Box 704, Room H2-D26, Yorktown Heights, NY 10598

135. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742

136. Joseph Oliger, Computer Science Department, Stanford University, Stanford, CA 94305

137. James M. Ortega, Department of Applied Mathematics, Thornton Hall, University of Virginia, Charlottesville, VA 22901

138. Steve Otto, Oregon Graduate Institute, Department of Computer Science and Engineering, 19600 NW von Neumann Drive, Beaverton, OR 97006-1999

139. Cherri Pancake, Department of Computer Science, Oregon State University, Corvallis, OR 97331-3202

140. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706

141. James C. T. Pool, Deputy Director, Caltech Concurrent Supercomputing Facility, California Institute of Technology, MS 158-79, Pasadena, CA 91125

142. David A. Poplawski, Department of Computer Science, Michigan Technological University, Houghton, MI 49931

143. Thierry Priol, IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France

144. Daniel A. Reed, Computer Science Department, University of Illinois, Urbana, IL 61801

145. Bernhard Ries, Intel Corporation, European Supercomputer Development Center, Dornacher Str. 1, W-8016 Feldkirchen b. M., Germany

146. Emilia Rosti, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Via Comelico 39, 20135 Milano, Italy

147. Diane T. Rover, 155 Engineering Building, Department of Electrical Engineering, Michigan State University, East Lansing MI 48824

148. Ahmed H. Sameh, Department of Computer Science, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455

149. Joel Saltz, ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, VA 23665-522

150. Jorge Sanz, IBM Almaden Research Center, Department K53/802, 650 Harry Road, San Jose, CA 95120

151. Carlo Savy, Dipartimento di Informatica e Sistemistica, Università di Napoli, Via Claudio 21, I-80125 Napoli, Italy

152. Robert B. Schnabel, Department of Computer Science, University of Colorado at Boulder, ECOT 7-7 Engineering Center, Campus Box 430, Boulder, CO 80309-0430

153. Robert Schreiber, RIACS, MS 230-5, NASA Ames Research Center, Moffet Field, CA 94035

154. James L. Schwarzmeier, Cray Research, Inc., 900 Lowater Road, Chippewa Falls, WI 54729

155. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006

156. The Secretary, Department of Computer Science and Statistics, The University of Rhode Island, Kingston, RI 02881

157. Giuseppe Serazzi, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133 Milano, Italy

158. Kenneth C. Sevcik, Computer Systems Research Institute, 10 King's College Road, University of Toronto, Toronto, Ontario M5S 1A1, Canada

159. Margaret L. Simmons, Computing and Communications Division, Los Alamos National Laboratory, Los Alamos, NM 87545

160. Horst D. Simon, NASA Ames Research Center, Mail Stop T045-1, Moffett Field, CA 94035

161. Burton Smith, Tera Computer Company, 400 North 34th Street, Suite 300, Seattle, WA 98103

162. Marc Snir, IBM T.J. Watson Research Center, Department 420/36-241. P. O. Box 218, Yorktown Heights, NY 10598

163. Danny C. Sorensen, Department of Mathematical Sciences, Rice University, P. O. Box 1892, Houston, TX 77251

164. Giandomenico Spezzano, Consorzio per la Ricerca e le Applicazioni di Informatica (CRAI), Località S. Stefano, I-87036 Rende (CS), Italy

165. Rick Stevens, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

166. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742

167. Paul N. Swarztrauber, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307

168. Wei Pai Tang, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2l 3G1

169. Bernard Tourancheau, LIP ENS-Lyon 69364, Lyon cedex 07, France

170. Anne Trefethen, Engineering & Theory Center, Cornell University, Ithaca, NY 14853

171. Cecile Tron, IMAG, INPG (LMC), 46, av Felix Viallet, 38031 Grenoble Cedex, France

172. Sue Utter Honig, Computer Science Department, 4105C Upson Hall, Cornell University, Ithaca, NY 14853

173. Roberto Vaccaro, Istituto per la Ricerca sui Sistemi Informatici Paralleli (IRSIP), CNR, I-80125 Napoli, Italy

174. Marco Vanneschi, Dipartimento di Informatica, Università di Pisa, Corso Italia 40, I-56124 Pisa, Italy

175. Mary Vernon, Computer Sciences Department, University of Wisconsin, 1210 W. Dayton Street, Madison, WI 53706

176. Robert G. Voigt, National Science Foundation, Room 417, 1800 G Street N.W., Washington, DC 20550

177. Michael D. Vose, 107 Ayres Hall, Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301

178. Tammy Welcome, Lawrence Livermore National Lab, Massively Parallel Computing Initiative, L-416, P. O. Box 808, Livermore, CA 94550

179. Steve Weston, Scientific Computing Associates, Inc., One Century Tower, 265 Church St., New Haven, CT 06510

180. Mary F. Wheeler, Department of Mathematical Sciences, Rice University, P. O. Box 1892, Houston, TX 77251

181. Andrew B. White, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545

182. John Zahorjan, Department of Computer Science and Engineering, Sieg Hall, FR-35, University of Washington, Seattle, WA 98195

183. Hans Zima, Department of Statistics and Computer Science, University of Vienna, Brunner Strasse 72, A-1210 Vienna, Austria

184. Roland Zink, Institute of Parallel and Distributed Supercomputers, University of Stuttgart, Univ. Stuttgart, IPVR, Breitwiesenstrasse 20, W-7000 Stuttgart 80, Germany

185. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P. O. Box 2001, Oak Ridge, TN 37831-8600

186–187. Office of Scientific & Technical Information, P. O. Box 62, Oak Ridge, TN 37831

DATE
FILMED
4 / 4 / 94

END