Performance Optimization of Parallel Programs - Tracing, Zooming, Understanding -

A. Arnold, *U. Detert*, and *W.E. Nagel*, Central Institute for Applied Mathematics, Research Centre Juelich (KFA), 52425 Juelich, Germany

ABSTRACT: *Performance optimization most often is based on the detailed knowledge of program behavior. One option to get this information is tracing. The visualization environment PARvis developed at KFA translates a given trace file generated on CRAY T3D into a variety of graphical views, e.g., state diagrams, activity charts, timeline displays, and statistics. PARvis supports an animation mode that can help to locate performance bottlenecks, and it provides flexible filter operations to reduce the amount of information displayed. Moreover, it has a powerful zooming feature that allows to identify problems at any level of detail.*

1 Introduction

On massively parallel computer systems, performance analysis and debugging can become an extremely complicated process. Over the years, experience has shown that user-friendly tools supporting this process are extremely helpful and can drastically shorten the time-to-solution for a given problem. At KFA Juelich, we have developed the X Window based *PARtools* environment (Fig. 1) which provides a general tool set for investigating problems in the area of parallel computing [NaAr93]. Currently, three components are provided: simulation of multiprocessor architectures (*PARsim*, [Nag93]), benchmark control (*PARbench*, [NaLi93]), and visualization (*PARvis*, [Arn93, NaAr93, NaAr94, Mue95]).

This paper describes the visualization component *PARvis*. Like most of the other performance analysis tools available for parallel computers (*Paragraph* [Int93] or *Pablo* [Ree92]), *PARvis* is used on a post-mortem basis, and it translates a given trace file into a variety of graphical system views which provide a reasonable basis for system understanding and program optimization.

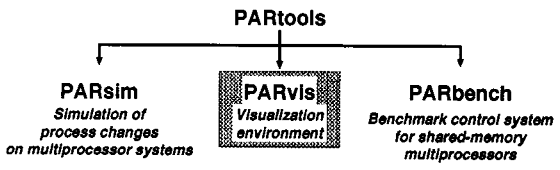

PARtools
PARsim
Simulation of process changes on multiprocessor systems
PARvis
Visualization environment
PARbench
Benchmark control system for shared-memory multiprocessors

Figure 1: The *PARtools* system structure

2 The *PARvis* Environment

Performance analysis and program optimization are often based on different categories of system views (Fig. 2), [Mue95]):

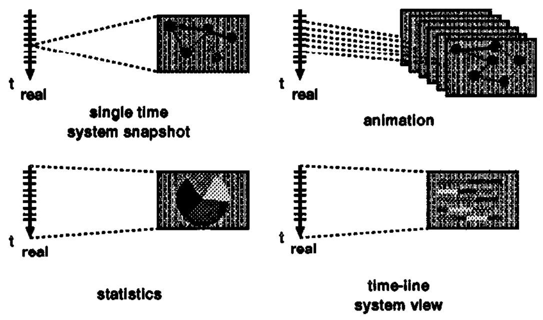


Figure 2: Visualization categories

- single time system snapshots: panels that show system activities at a particular point of time;
- animation: option to look at a sequence of system snapshots to investigate the dynamic behavior;
- statistics: the component that summarizes system behavior for the time under investigation;
- time-line system view: detailed view of system activities, which are visualized on a time axis.

Each of these categories is supported by the *PARvis* environment. Over the last years, *PARvis* has been extended to support performance visualization of parallel programs on Intel iPSC/860, Intel Paragon, and CRAY T3D systems. On Intel

Paragon, *PARvis* takes the *ipd*-generated trace file and extracts the graphical information automatically [NaAr94].On CRAY T3D, the user has to call a preprocessor which instruments the code before execution. Extensions have been made to show the flow of messages on different topologies as well as to display network activities. For user convenience, *PARvis* provides a configuration file *PARvis*.cnf where user preferences (color, layout, fonts, etc.) are stored from run to run. This file contains all settings made within *PARvis* and enables *PARvis* to come up with the same settings you had in the previous session. A detailed description of all *PARvis* features can be found in [Arn93, Arn95, Mue95].

PARvis is implemented in ANSI C and uses the OSF/Motif libraries. The current implementation already supports a variety of different hardware platforms (IBM RS/6000, Sun Sparc, DEC MIPS computers (Ultrix), DEC Alpha, and Silicon Graphics). Figure 3 System activity snapshot at a single point of time

3 Program Instrumentation on CRAY T3D

Based on the preprocessor *Paff* [Ber89] developed at KFA Juelich, we have implemented an instrumentation tool called *PARvis.inst*. The command

PARvis.inst [options] file_name [file_name]

automatically instruments all Fortran 77 programs which are specified on the command line. There are flexible options to generate *wrapper*-routines e.g. for the message-passing routines, and the tracing for each routine can be switched off by just marking this routine as non-traceable. Control directives are supported to start and stop the tracing process. In addition, the output is limited to roughly 1 Mbyte per processor, nevertheless the user can enlarge this value as wanted. All supported directives can be found in the following list:

- CKFA$ TRACE STOP
- CKFA$ TRACE START
- CKFA$ TRACE LIMIT N
- CKFA$ TRACE NOREPLACE

The last directive identifies a *wrapper*-routine where the replacement of names is suppressed.

4 Visualization of System Activities

The main window of the *PARtools* environment (Fig. 3) is partitioned into several areas. Just below the menu line, the *PARvis* control buttons reside in the middle part of the window.The control functions and panels for the other components of *PARtools* (*PARsim* and *PARbench*) are placed on both sides. Under these sub-windows, the visualization area can be

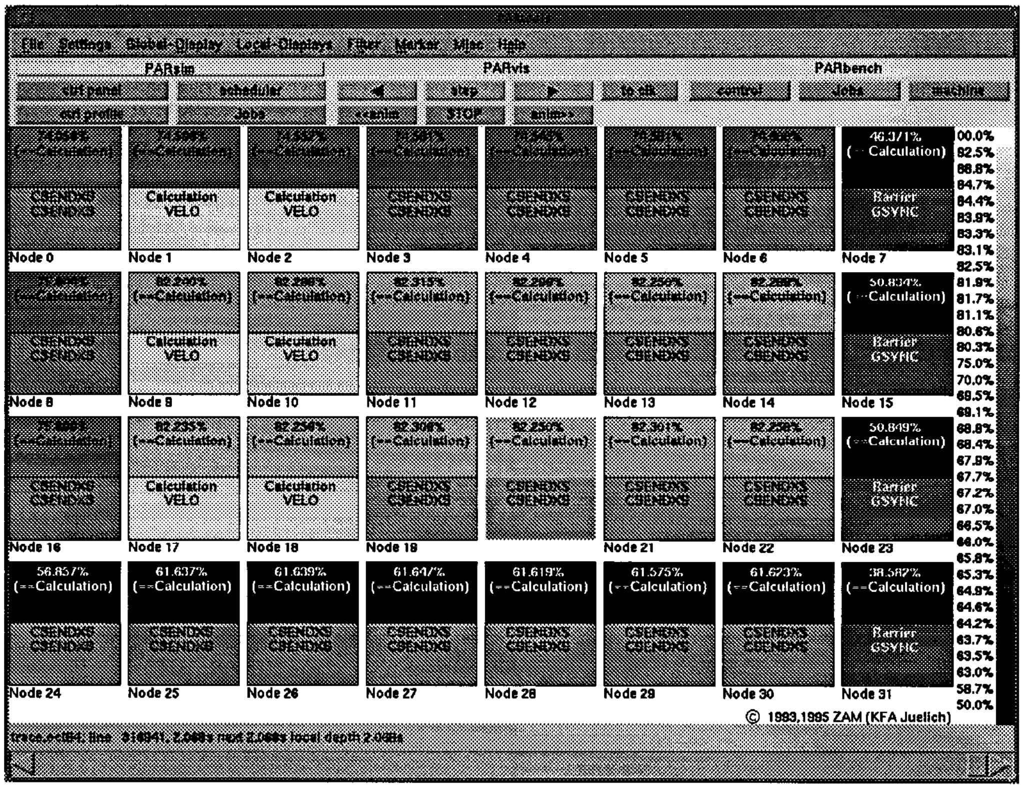


Figure 3: System activity snapshot at a single point of time

found, which is used to display the actual system activities executed on the nodes.

The main window can display the trace data in several selectable modes. After start-up, the *Node display* is set.This mode displays every processor as a box.The size and arrangement of the boxes depend on the number of processors and the geometry of the main window and are automatically calculated by *PARvis*. Each box is partitioned into a lower and an upper part.The lower part describes the current activity on the nodes, whereas the upper part (called *statistic field*) shows the time portion (in percent) spent on a particular activity for the period under investigation (here: *Calculation*). For monitoring reasons, the background color reflects the current value printed out, and the corresponding percentual values are listed on the right.

Fig. 3 represents one example of an actual system snapshot at a special point of time.The *step*-button in the *PARvis* control area can be used to show the system activity changes.Typically, the number of events to be displayed is rather large, so the animation mode can be used to animate the sequence of system snapshots. The step width for the animation mode can be either an event or a given time period; the time difference between two movements (i.e., the animation speed) and the number of movements after which the animation should stop can be adjusted in the panel *Settings/steppings*. This animation feature can be used to analyze the program behavior in time, to identify critical program sections, and to find the hot spots of the run.

5 Statistics

The Node display mode already contains a small statistics field, but due to its limited size only the time portion of *one* state can be monitored.Quite often, one would like to get a more detailed idea of how the time is spent on each of the nodes. To analyze the complete state distribution, it is possible to switch the display mode to the *statistics display*. Press F6 or select the menu option *Global_Display/Chart Style*, and the main window layout will switch to something like Fig. 4, which shows a statistics of the complete trace file in a pie chart style. The colors chosen for the individual states are just the same as those which are used as the background color for the state field in the CPU display. The most important activities can be identified for all nodes, and differences in the node behavior will be clear immediately (Fig. 4). As can be seen from that panel, most time is spent in *Calculation* on all nodes, and significant portions of time are also spent at a barrier.

When a lot of CPUs are involved in a parallel system, the individual statistics in this display can become very small and uninformative. To relieve this unfortunate situation, *PARvis* can open additional windows containing statistics for only *one* CPU. To select the CPUs you want statistics for, simply click at them with the left mouse button, and their frame color will be inverted.You can also drag Figure 4 Time distribution statistics for the program run over a couple of CPUs to select several CPUs with one action. In the example shown in Fig. 4, the

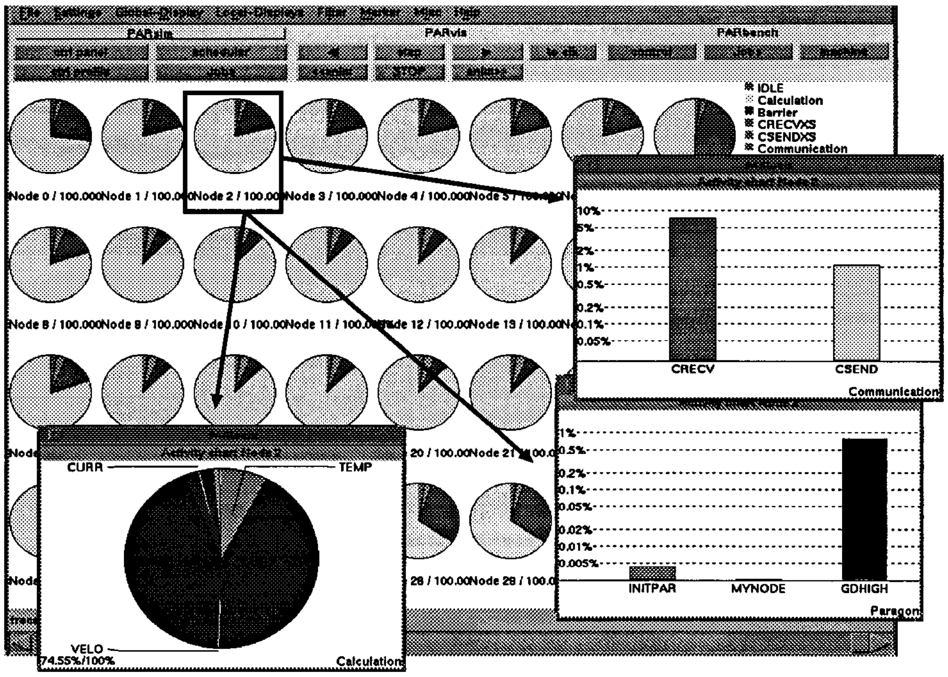


Figure 4: Time distribution statistics for the program run

actual time distribution spent in user subroutines (*Calculation*, pie chart in the left sub-window: most time is spent in subroutine VELO) as well as for node communication (*Communication*, histogram in the upper right sub-window), and *Paragon* emulation (Paragon, histogram in the lower right sub-window) is shown for node 2. The user can toggle between table, pie chart, and histogram in all chart windows.The histograms may be linear or logarithmic, and zooming is supported.

6 System Activity Profile

The statistics windows only show the accumulated breakdown of system activities over the whole program run. Nevertheless, the program may have different phases where the timing behavior of system activities is quite different. These phases can be identified either by looking for differences during an animation run or by viewing the window *system activity profile* (Fig. 5).

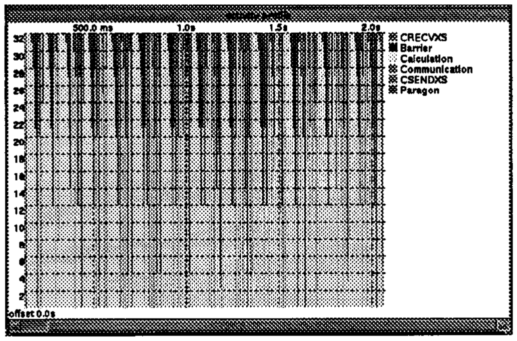


Figure 5: System activity profile

This window shows the actual time distribution for all system activities in each time step accumulated over all the nodes; and problem areas where, for example, *calculation* has low values, can be seen immediately. Figure 7 *PARvis* realization: Make zooming as easy as possible

7 Time-line Displays

Based on the data visualization options presented above, we now concentrate on the interaction of parallel activities and possible bottlenecks. At this point, the user is interested in seeing a sequence of activities on all nodes, and the interdependences between these different program parts.

The problem with most other visualization tools like *Paragraph* [Int93] or *Pablo* [Ree92] is that these tools are based on the *Replay Technique*: Whenever the user wants to have just another information about a special part of the program, the whole trace file is analyzed once again, even if the file contains several hundreds of Mbytes (see Fig. 6).The magnification glass has to scan the whole trace file several times whenever the user would like to see a different information or just another time frame.

This is different in the *PARvis*-environment:here, the user can specify the size of the magnification glass, and all details

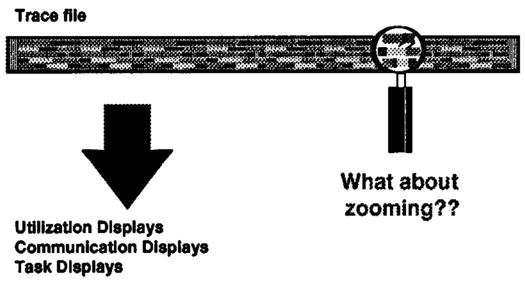


Figure 6: Zooming and the *Replay Technique*

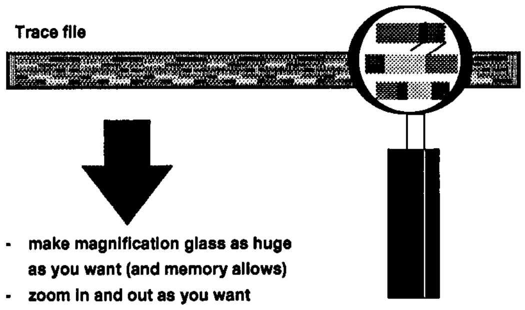

Trace file
make magnification glass as huge as you want (and memory allows)
zoom in and out as you want

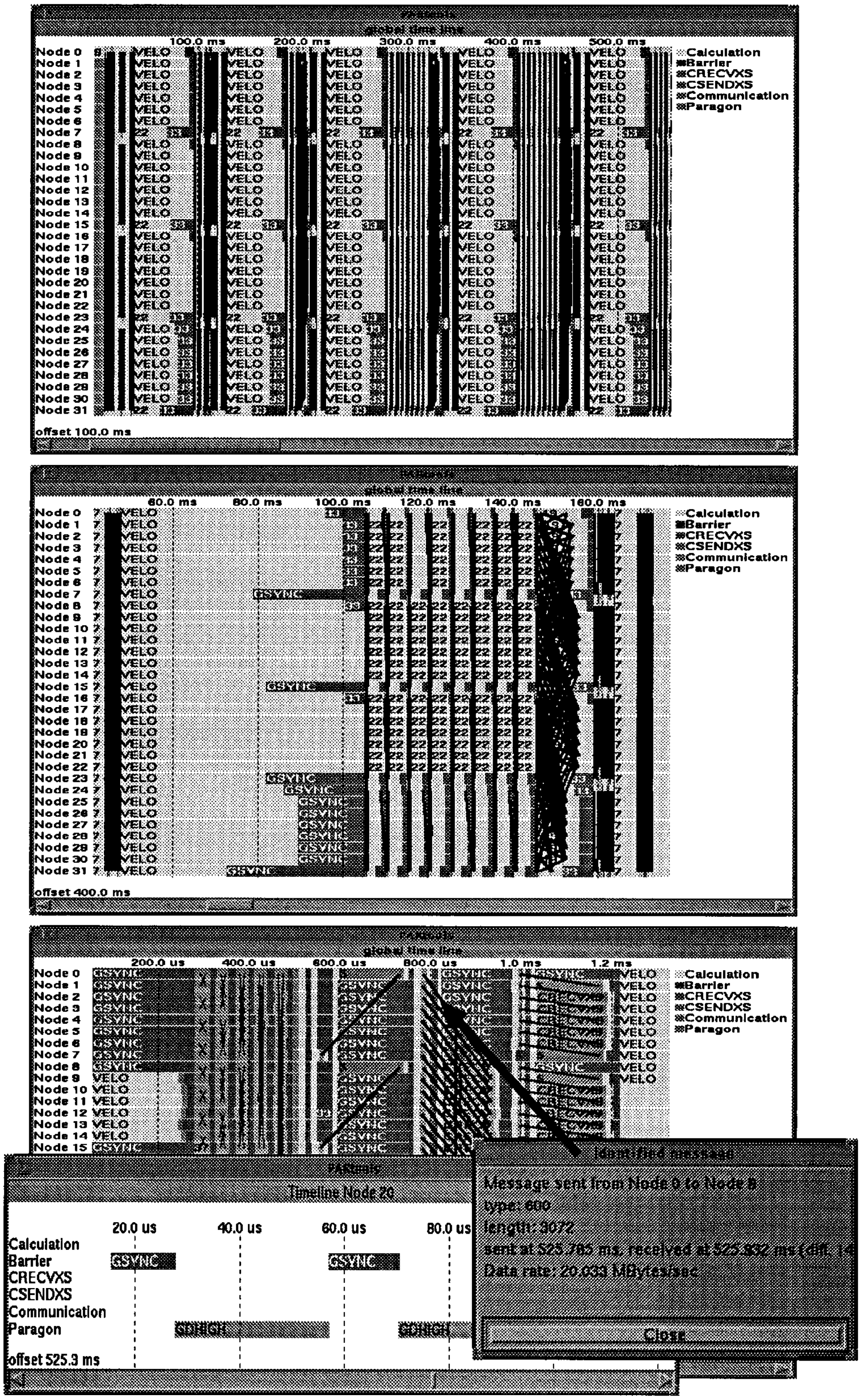
Figure 7: *PARvis* realization: Make zooming as easy as possible

within the magnification glass can be seen without any further I/O-activity (Fig. 7). For example, statistics for all activities inside the chosen time window can be generated within milliseconds.Moreover, the user can use a powerful zooming feature to analyze the program behavior on any level of detail; each zoom-operation also takes only a few milliseconds, even if several Mbytes of tracing information are under investigation.Of course, a hierarchical *unzoom*-operation is provided for user convenience.

In *PARvis*, the *Global_Display/Timeline* panel is used to display this type of information.As can be seen from the upper part of Fig. 8, colors are used to represent different kinds of activities, and it is possible to show system activities over time on each of the nodes. In this example, the program is running in phases where the subroutine VELO is executed several times. The black parts are hundreds of messages (represented by one line each) which are sent between the nodes. Based on the information displayed in this window, it is quite easy to identify critical program sections where problems may have occurred.

The powerful zooming feature can now be used to go into detail. As shown in the middle part of Fig. 8, the period of interest[1] (400 - 560 ms) was zoomed-in by just specifying the time frame with the mouse. Here, one of the time-step iterations can be seen, and the load imbalance causes long synchronization times at the barrier called GSYNC.

1. The time offset is specified in the lower left corner of the panel.



Message sent from Node 0 to Node 8
type: 600
length: 3072
sent at 525.785 ms, received at 525.932 ms (diff. 14
Data rate: 20.033 MBytes/sec

Figure 8: Time-line zooming and message identification

The zooming feature also can be used to get deeper and deeper into the analysis process, to understand program behavior, and finally to identify problems. The lower part of Fig. 8 shows a data communication exchange part of the program (at about 525 ms) where different communication patterns inform the user about his communication activities. In the message passing programming model, communication and data exchange are solely based upon the sending and receiving of messages.Regardless of the network's topology (which is hidden to the application programmer in most cases), it is obvious that the visualization of message transfers and patterns plays an important role in the performance analysis and debugging of parallel programs. Therefore, *PARvis* includes means to display and inquire information about message-passing transfers.These tools are not isolated from the other part of *PARvis*: message events are read through the same trace file interface into *PARvis*, and the message visualization tools work hand-in-hand with the features described so far. It is possible to mouse-click a message that pops up another panel showing all information related to that message, including the transfer rate in MByte/s (i.e.about 20 MByte/s).The information for this message is coming out of the wrapper of the *shmem_get* communication routine, and the overhead involved is quite low. Moreover, detailed information about the activities on one node or a selection of nodes can be obtained.The lower left part of Fig. 8 documents that even calls to *gdhigh* (a few microseconds inside the communication library routine) easily can be identified. A case study on Intel Paragon [WiNa94] describes a situation where the *PARvis* environment was extremely helpful in identifying performance bottlenecks in the communication library; based on the optimization process, the output performance (*hippi-output*) was increased by a factor of more than five within a few hours.

In addition, the zooming operation can be used to identify typical communication patterns. It is obvious that the visualization of such communication patterns gives knowledge about implementation aspects of the system and of your own program, and it is very helpful to understand synchronization delays and related side effects which sometimes significantly influence the performance of real applications.

8 Additional Features

PARvis accesses several external tools to perform some of its tasks. These tools must be located in a directory included in your PATH environment variable:

- *lpr*, the standard UNIX printing facility, to print lists and window snapshots.
- *import*, a screen snapshot utility from the *ImageMagick* package to export or print window contents. *ImageMagick* can be downloaded from *ftp.zam.kfa-juelich.de*, directory *pub/graphics/ImageMagick*.
- If you have trace files compressed with *gzip* or *compress*, *PARvis* can extract them automatically if their counterparts *gunzip* or *uncompress*, respectively, are available.

There are quite a few other enhanced features that cannot be described in detail in this paper; the most important ones are mentioned below:

- filter functions: *PARvis* allows to simultaneously display up to 512 nodes. Typically, this number is much too large to be handled meaningfully; therefore, powerful filter functions are available to reduce the number of nodes, either automatically or manually by the user.
- network activity: communication messages are sometimes moving over the same hardware connections, leading to hot spots on the network. This component is able to display communication patterns on the underlying network.
- extensions for *shared virtual memory* (*SVM*): *PARvis* has been enhanced recently to include a number of new windows that help to understand memory access patterns in systems with *shared virtual memory* (*SVM*). These extensions are not discussed in this paper; details can be found in [Mue95].
- movie support: after each animation step, control is optionally given back to a user-command (i.e., a shell script). This allows to generate movies unattended by the user, just by specifying a single command in a sub-panel.

9 Summary and Conclusions

This paper describes the *PARvis*-environment which provides some powerful features to discover parallel program behavior on several parallel systems like Intel Paragon and CRAY T3D. Experience has shown, that for debugging, as well as for performance optimization purposes, the supported time-line displays in combination with the statistics features are the strength of the system. With the extremely flexible zooming function in the time-line displays, analysis operations are supported which can drastically improve the understanding of observed performance problems.

References

[Arn93] A. Arnold, *PARvis*: *Eine X-basierte Umgebung zur Visualisierung von parallelen Programmen in Multiprozessorsystemen*, Juel-2848, Forschungszentrum Juelich (KFA), 1993.

[Arn95] A. Arnold, *PARvis - an X-based visualization environment for parallel programs* (*User's guide*), Forschungszentrum Juelich (KFA), to be published.

[Ber89] R. Berrendorf, *Der FORTRAN-Parser PAFF als wiederverwendbares Modul fuer Programmier-Tools*, Juel-Spez-537, Forschungszentrum Juelich (KFA), 1989.

[Int93] *Paragon application tools user's guide*, Intel Corporation, 1993.

[Mue95] Ch. Muellender, *Visualisierung der Speicheraktivitaeten von parallelen Programmen in Systemen mit virtuell gemeinsamem Speicher*, Juel-2911, Forschungszentrum Juelich (KFA), 1994.

[Nag93] W. E. Nagel, *Ein verteiltes Scheduler-System fuer Mehrprozessorrechner mit gemeinsamem Speicher: Untersuchungen zur Ablaufplanung von parallelen Programmen*, Juel-2850, Forschungszentrum Juelich (KFA), 1993.

[NaAr93] W.E. Nagel und A. Arnold, *PARvis: Ein Werkzeug zur Visualisierung von parallelen Prozessen auf Mehrprozessorsystemen*, Proc.7.ITG/GI Fachtagung MMB'93 (Kurzberichte und Werkzeugvorstellung) pp. 178-187, 1993.

[NaAr94] W.E. Nagel und A. Arnold, *Performance visualization of parallel programs: The PARvis environment*, In: Proc. 1994 Intel Supercomputer Users Group Conference (ISUG'94), pp. 24-31.

[NaLi93] W.E. Nagel and M.A. Linn, *Benchmarking parallel programs in a multiprogramming environment: The PAR-Bench System*, in Dongarra/Gentzsch, Eds.:Advances in parallel computing, Vol. 8:Computer benchmarks, Elsevier Science Publishers B.V., pp. 303-322, 1993

[Ree92] D.A. Reed, R.A. Aydt, T.M. Madhyastha, R.J. Noe, K.A. Shields, and B.W. Schwartz, *An overview of the Pablo performance analysis environment*, Technical Report, Dept. of Computer Science, University of Illinois, Urbana-Champaign, 1992.

[WiNa94] R. Williams and W. E. Nagel, *Optimization of output bandwidth from a Paragon*, Technical Report CCSF-44, Caltech Concurrent Supercomputing Facilities, Pasadena, CA, 1994 (also on WWW: http://www.ccsf.caltech.edu/"roy/hippipap/paper.html).