Parallel Debugging and Performance Analysis Tools

Prof. Dr. A. Bode, J. Krammer, Dr. T. Ludwig,
M. Oberhuber, Dr. R. Wismüller
Lehrstuhl für Rechnertechnik und Rechnerorganisation
Technische Universität München

Keywords: Parallel debugging, performance analysis, monitoring

Research Activities

A main research area of the Lehrstuhl für Rechnertechnik and Rechnerorganisation at Technische Universität München (LRR-TUM) are tools supporting the development of parallel programs. Within a couple of funded projects (German Science Foundation: SFB 342, Esprit: PREPARE and HAMLET) tools supporting parallel I/O, load balancing, resource management and checkpointing, performance analysis, debugging, deterministic execution, and visualization are being developed.

The debugging tool DETOP, the performance analysis tool PATOP, and the underlying monitoring system have recently been ported to the PowerXplorer and GC/PowerPlus parallel computers. This work has been carried out in cooperation with Parsytec, which distributes both tools since Dec. 1994. While LRR-TUM has ported and improved the debugger and the performance analyzer, Parsytec has implemented the necessary support in the operating system. A modified and enhanced PTrace interface is now provided by the nanokernel which is the basis of the PARIX library on the PowerPC based computers.

The Parallel Debugger DETOP

The parallel debugger DETOP allows to examine the run-time behavior of both data-parallel programs and programs based on functional parallelism. Programs may consist of multiple threads per node and even multiple contexts, i.e. executable files. The most common programming languages, C and Fortran 77, are supported.

DETOP provides a comfortable graphical interface that is based on a debugger window as shown in Fig. 1. The window can be associated with an arbitrary set of threads that is displayed in the list at the left side. All debugger commands, e.g. *single-step*, *set breakpoint* or *print*, are applied to all of these threads or a selectable subset. So data parallel programs where all threads execute the same code can be examined using a single window. Multiple windows can be used to debug groups of threads executing different codes.

DETOP offers special support to debug dynamically created threads. A special option requests the debugger to stop such treads immediately before they start executing. In order to ease identification of threads, additional information is displayed in the thread lists: a global identifier, the name of the thread's root function, and the name of its executable file.

The debugger is based on the event-action paradigm instead of a simple breakpoint scheme. Each event (e.g. 'thread reaches a source line', 'message is received' or 'new thread has been

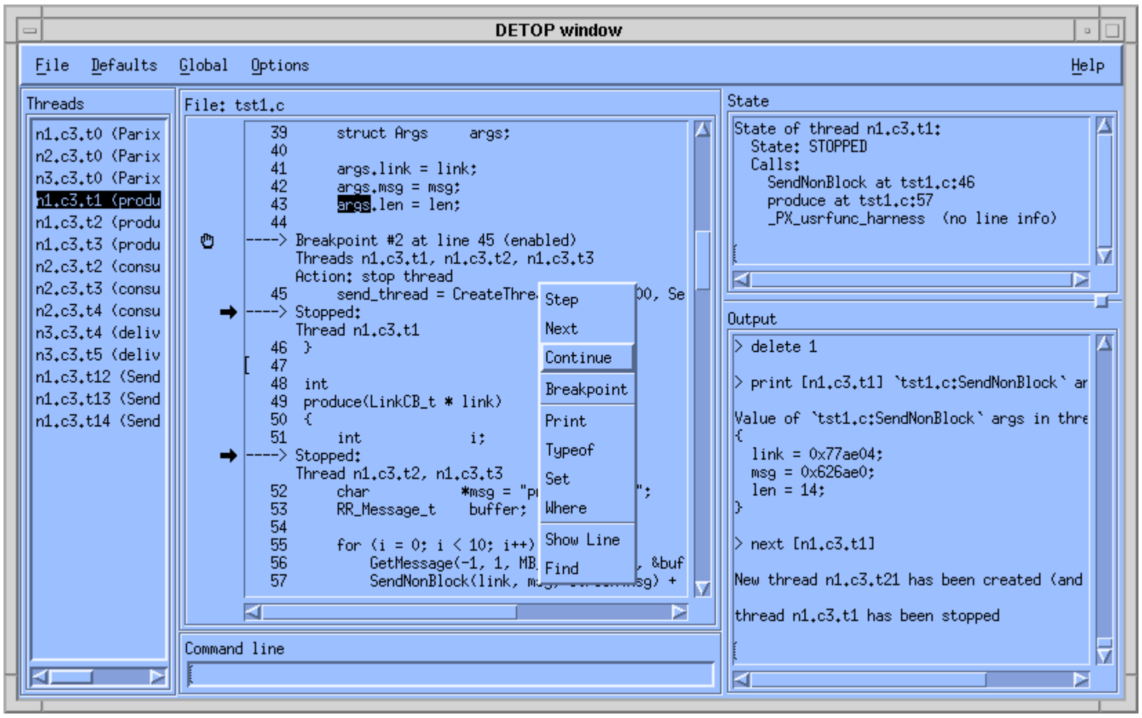

Figure 1: Sample window of DETOP

created') can be associated with a list of actions that are executed when the event occurs. Although only a limited set of actions (stop threads, retrieve a stack backtrace) is currently used by the debugger, we will support more complex actions (e.g. tracing, printing variables or defining new events and actions) in future. Since actions are evaluated autonomously by the distributed monitoring system without interaction with the debugger's front end or the user, intrusion is kept very low. Furthermore, actions may also be triggered by events on remote nodes, so global halting and distributed breakpoints are possible. We plan to provide special breakpoints allowing to follow message transfers: when a thread sends a message, the breakpoint will stop the receiver immediately after it has received that message, so data processing can be watched across thread boundaries.

An other issue for future research is the visualization of regular and irregular data structures and program states within the debugger. We will also extend the debugger in order to specially support the PVM and MPI parallel programming models running on top of PARIX.

The Performance Analyzer PATOP

The PATOP system is used to examine parallel PARIX C and Fortran 77 programs in order to find performance bottlenecks and potential optimization areas. It supports the on-line measurement of CPU utilization, communication delays, delays in external I/O, data transfer rates, and delays in parts of the program that are defined by statements inserted by the user. Measurements can be performed for the complete program and the whole computing system to get an overview on the system's performance and to detect the presence or absence of bottlenecks. In order to isolate the program parts that cause a bottleneck, measurements can also be restricted to individual procedures and sets of nodes or sets of processes.

An important research issue addressed in our performance analysis work is scalability. There are three aspects that must be considered: scalable data representation, a scalable method of tracking down performance bottlenecks, and a scalable data acquisition.

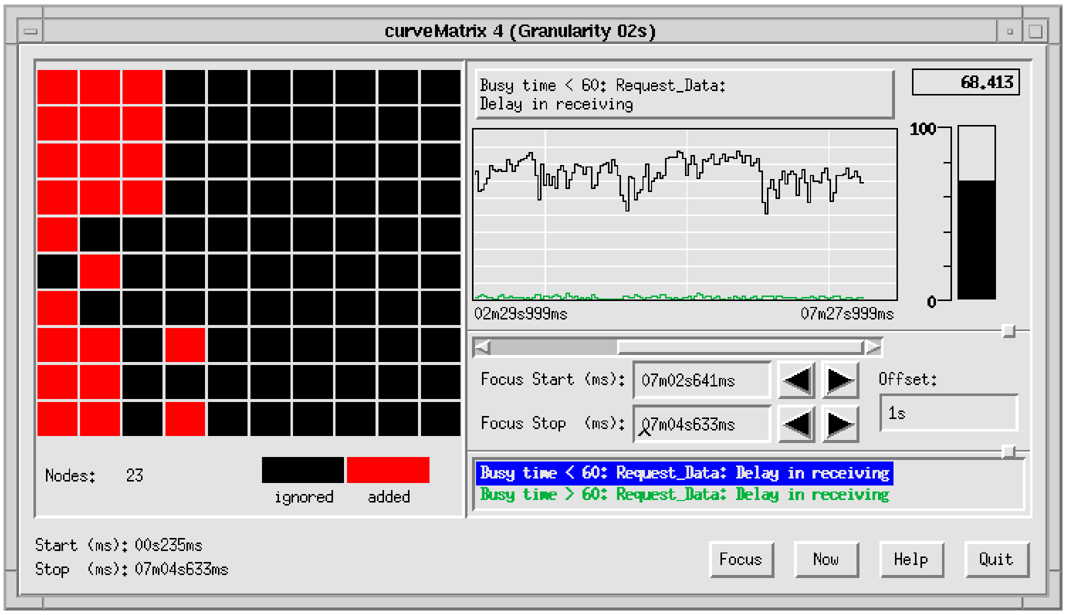

Figure 2: Sample window of PATOP

A scalable data representation requires the size and clarity of the displays to be independent of the number of processing nodes or threads used by the observed program. In PATOP, this is achieved by displaying global performance figures summed up over all nodes, resulting in a single value for the whole system, and by using distribution diagrams, showing the number of nodes that have performance values in predefined ranges. In addition, there are also conventional displays, such as color-coded matrix diagrams or curve diagrams, plotting a value against time.

The task of tracking down performance bottlenecks is supported independently of the system size by using attributed measurements, that close the gap between measurements performed on the whole system and those done on only one node. In an attributed measurement, the nodes to be observed need not be enumerated explicitly, but are specified by their performance behavior. On all nodes, the attribute is then permanently monitored. If the condition is fulfilled, the actual measurement is performed and reckoned in the resulting value. A possible measurement of this class is shown in Fig. 2: "The average delay in receiving messages in function `Request_Data` on all nodes that are less than 60% busy". The matrix at the display's left side shows the nodes included in the last measurement, while the curve shows the resulting value over a time axis.

Scalable data acquisition for global measurements, distribution values, and attributed measurements is achieved by computing measurement values in a data parallel fashion. The monitor on each node gets a position in a spanning tree. It receives values from its predecessors, reads and evaluates its local values, combines them with the received ones and sends the result to its successor. In this way, no further post-processing has to be done by the performance analyzer's front end and still the additional overhead in the monitoring system is very small.

Future work will address a more automated guidance in the search of performance bottlenecks and an adaption of PATOP for Parsytec's industrial systems.

References

[1] T. Bemmerl and R. Wismüller. On-line Distributed Debugging on Scalable Multiprocessor Architectures. In W. Gentzsch and U. Harms, editors, *High-Performance Computing and Networking, Volume II: Networking and Tools*, volume 797 of *Lecture Notes in Computer Science*, pages 394–400, München, April 1994. Springer-Verlag.

[2] A. Bode. Parallel Program Performance Analysis and Visualization. In J. Dongarra and B. Tourancheau, editors, *Environments and Tools for Parallel Scientific Computing*, pages 246–253, Townsend, Tennessee, May 1994. SIAM.

[3] O. Hansen. A tool for optimizing programs on massively parallel computer architectures. In W. Gentzsch and U. Harms, editors, *High-Performance Computing and Networking, Volume II: Networking and Tools*, volume 797 of *Lecture Notes in Computer Science*, pages 350 – 356, Munich, April 1994. Springer-Verlag.

[4] O. Hansen and J. Krammer. A Scalable Performance Analysis Tool for PowerPC Based MPP Systems. In Mirenkov et al., editors, *The First Aizu International Symposium on Parallel Algorithms / Architecture Synthesis*, pages 78–84, Aizu, Japan, March 1995. IEEE Comp. Soc. Press.

[5] M. Oberhuber and R. Wismüller. DETOP – An Interactive Debugger for PowerPC Based Multicomputers. In *Proc. ZEUS'95 Workshop on Parallel Programming and Computation*, Linkoping, Sweden, May 1995.

Contact address:

Contact:	Prof. Dr. A. Bode
Company:	Lehrstuhl für Rechnertechnik und Rechnerorganisation
	Technische Universität München (TUM)
Mail:	Prof. Dr. A. Bode
	Institut für Informatik, SAB
	Technische Universität München
	D-80290 München
Phone:	+49-89-2105-8240
Fax:	+49-89-2105-8232
Email:	bode@informatik.tu-muenchen.de
WWW:	http://wwwbode.informatik.tu-muenchen.de/