THE TOOL-SET — An Integrated Tool Environment for PVM*

Thomas Ludwig, Roland Wismüller, Rolf Borgeest, Stefan Lamberts,
Christian Röder, Georg Stellner, Arndt Bode
Lehrstuhl für Rechnertechnik und Rechnerorganisation (LRR-TUM)
Technische Universität München, D-80290 München, Germany
Tel.: +49-89-2105-2042 or -8164 or -8240, Fax: +49-89-2105-8232
E-mail: {ludwig,wismuell,bode}@informatik.tu-muenchen.de

Abstract

THE TOOL-SET *for PVM will comprise a set of integrated tools which can either be used individually or in concert.* THE TOOL-SET *will be composed of a debugger, a performance analyzer, a visualizer, a deterministic execution controller, a load balancer including a checkpoint generator, and a parallel file system. All tools will be available under the GNU General Public License Agreement. First versions of the tool environment will be released in spring 1996.*

1 Introduction

PVM has become a de-facto standard for writing parallel applications based on the message passing paradigm. It is also being used as a platform for developing tools for parallel programming. Although some of them are rather sophisticated, currently only few are really usable for application programmers. One reason for this fact is that many tools are pure research prototypes providing only rudimentary, clumsy interfaces. A second reason is that they cover only a single aspect of parallel program development and don't support an integrated usage, i.e. they can not be used in combination and don't provide support for each other. Currently, tools often require a special compilation or instrumented libraries not compatible with each other.

A new project of LRR-TUM will change this situation by providing an integrated tool-set for PVM including both run-time support and programming tools. The project is based on the previous work of our group that is performing research in this field for more than eight years. It will integrate and adapt our existing tools and experience to form a PVM tool environment supporting parallel I/O [6], load balancing [3], resource management and checkpointing [7], performance analysis [2], debugging [5], deterministic execution [4], and visualization [1].

*This work is partly funded by the German Science Foundation, Contract: SFB 342, TP A1

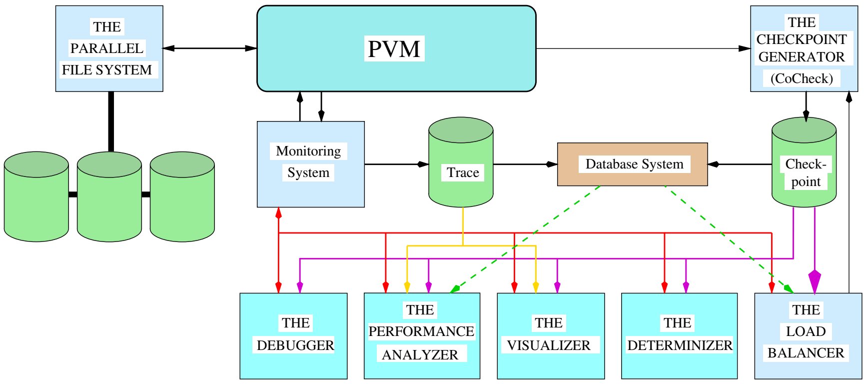

Figure 1: Module structure of the complete tool environment

2 THE TOOL-SET

The tools can be divided into classes according to the following criteria. First, we will offer interactive tools and automatic tools. Interactive tools (e.g. THE DEBUGGER) will support implementation and maintenance phases whereas automatic tools (e.g. THE LOAD BALANCER) mainly concentrate on the production phase of the software. Tools will use different sources of information. A monitoring system will give direct access to the running program on the workstation cluster, thus supporting online debugging and performance measurement. All measured characteristics can be recorded in traces which describe the individual behavior of a single program execution. Traces can be used e.g. for statistical program analysis. In addition, checkpoints will be generated which represent the state of a program at a given point of time. Checkpoint data is mainly used for load balancing purposes. Traces and checkpoints can be entered in a database system for comparison, version management etc. All tools are based upon these few information types, however, none of them uses the complete set. Interactive tools will be integrated considering their graphical user interface. The GUIs will be based on OSF/Motif and therefore give the users a standardized look and feel.

Finally everything is grouped around PVM version 3.3.x. We expect THE TOOL-SET to be easily adaptable to future versions of PVM as only the monitoring system is closely interconnected with internal PVM mechanisms. However, any change of programming model would require an adaption of those interactive tools which should be able to handle the modified constructs.

Figure 1 shows the module structure of THE TOOL-SET. Due to the limited space of this paper we can only give a list of some selected highlights of the tools:

- THE DEBUGGER can use checkpoint information to resume execution of a program starting from a specific point during the program run. Debugging cycles will become shorter as the program has not to be restarted from the beginning.
- THE PERFORMANCE ANALYZER will try to automatically detect a focus of interest, e.g. workstations with high idle-time or communication frequency. It will also offer statistical data and comparison of different program runs by using traces.
- THE VISUALIZER will show the behavior of PVM programs at the level of tasks and communications thus unveiling e.g. deadlock situations.

- THE DETERMINIZER supports deterministic runtime behavior of a parallel program thus ensuring reproducible program runs and results during test phases.
- THE LOAD BALANCER can migrate running processes from loaded to free workstations, using the checkpoint generator CoCheck. The heuristics that control migration decisions can be improved by learning from traces of previous program runs.
- THE PARALLEL FILE SYSTEM will support various modes for accessing data files. All activities can be monitored and evaluated by THE PERFORMANCE ANALYZER.

3 Project Status and Availability

Currently we are implementing THE VISUALIZER, THE PARALLEL FILE SYSTEM, and CoCheck. As the monitoring system is not yet implemented, THE VISUALIZER will preliminaryly be based upon XPVM-traces. We have not yet decided whether we will also provide a version of THE PERFORMANCE ANALYZER for this trace type.

The next step will be the implementation of the monitoring system which is a prerequisite to all interactive tools and THE LOAD BALANCER. With the monitor being finished we will adapt our existing tools to the instrumented PVM environment. First products will be available for the users in spring 1996. They will be fully functional but will not yet have enhanced features which are current research topics of our group.

References

[1] A. Bode and P. Braun. Monitoring and Visualization in TOPSYS. In *Proc. Workshop on Monitoring and Visualization of Parallel Processing Systems*, pages 97 – 118, Moravany nad Váhom, CSFR, Oct. 1992.

[2] O. Hansen. A Tool for Optimizing Programs on Massively Parallel Computer Architectures. In *High-Performance Computing and Networking, Volume II*, volume 797 of *Lecture Notes in Computer Science*, pages 350 – 356, München, Apr. 1994. Springer Verlag.

[3] T. Ludwig. Aspects of Load Management on Parallel Computers. In *Proc. International Conference Parallel Computing Technologies, PaCT-93, Obninsk, Russia*, pages 301–313, Vol. II, Moscow, Russia, Sept. 1993. Recursive Super Computers (ReSCo).

[4] M. Oberhuber. Elimination of Nondeterminacy for Testing and Debugging Parallel Programs. In *Proc. of the 2nd Int. Workshop on Automated and Algorithmic Debugging (AADEBUG'95)*, St. Malo, France, May 1995.

[5] M. Oberhuber and R. Wismüller. DETOP - An Interactive Debugger for PowerPC Based Multicomputers. In P. Fritzson and L. Finmo, editors, *Parallel Programming and Applications*, pages 170–183. IOS Press, May 1995.

[6] C. Röder, S. Lamberts, and T. Ludwig. PFSLib - An I/O Interface for Parallel Programming Environments on Coupled Workstations. In *2nd European PVM Users' Group Meeting*, Lyon, France, Sept. 1995.

[7] G. Stellner and J. Pruyne. Resource Management and Checkpointing for PVM. In *2nd European PVM Users' Group Meeting*, Lyon, France, Sept. 1995.