EDPEPPS: An Integrated Graphical Toolset for the Design and Performance Evaluation of Portable Parallel Software*

T. Delaitre, P. Vekariya, R. Bigeard, G.R. Justo,
S.C. Winter and M.J. Zemerly

Centre for Parallel Computing, University of Westminster
115 New Cavendish Street, London W1M 8JS
Email: edpepps-all@cpc.wmin.ac.uk

Abstract. This paper describes an integrated graphical toolset for performance-oriented design of portable parallel software. The toolset consists of a graphical design tool based on the PVM communications library for building parallel algorithms, a simulation engine and a visualisation tool for animation of program execution and visualisation of platform and network performance measures and statistics. The toolset is used to model a virtual machine composed of a cluster of workstations interconnected by a local area network. The simulation model used is modular and its components are interchangeable which allows easy reconfiguration of the platform. The model is validated using experiments on the COMMS1 Benchmark from the Parkbench suite, and a standard image processing algorithm with accuracy to within 10%.

1 Introduction

A major obstacle to the widespread adoption of parallel computing in industry is the difficulty in program development due mainly to lack of parallel programming design tools. In particular, there is a need for performance-oriented tools, and especially for clusters of heterogeneous workstations, to allow the software designer to choose between design alternatives such as different parallelisation strategies or paradigms. A portable message-passing environment such as Parallel Virtual Machine (PVM) [4] permits a heterogeneous collection of networked computers to be viewed by an application as a single distributed-memory parallel machine. Traditionally, parallel program development methods start with parallelising and porting a sequential code on the target machine and running it to measure and analyse its performance. Re-designing the parallelisation strategy is required when the reached performance is not satisfactory. This is a time-consuming process and usually entails long hours of debugging before reaching an acceptable performance from the parallel program. Rapid prototyping is a useful approach to the design

* This project is funded by an EPSRC PSTPA programme, Grant No.: GR/K40468

of parallel software in that complete algorithms, outline designs, or even rough schemes can be evaluated at a relatively early stage in the program development life-cycle, with respect to possible platform configurations, and mapping strategies. Modifying the platform configurations and mappings will permit the prototype design to be refined, and this process may continue in an evolutionary fashion throughout the life-cycle before any parallel coding takes place.
The EDPEPPS toolset described here is based on a rapid prototyping philosophy and comprises three main tools: a graphical design tool (PVMGraph), a simulation utility based on SES/Workbench [14], and a visualisation tool (PVMVis). The advantage of the EDPEPPS toolset is that the cyclic process of design-simulate-visualise is executed within the same environment. The toolset is also modular and extensible to allow modifications and change of platforms and design as and when required.
In the next section we describe several modelling tools with similar aims to EDPEPPS. In section 3 we describe the different tools in the EDPEPPS toolset. In section 4 we present results obtained from the case studies. Finally, in section 5 we present conclusions and future work.

2 Parallel System Performance Modelling Tools

The current trend in parallel software modelling tools is to support all the software performance engineering activities in an integrated environment [12]. A typical toolset should be based on at least three main tools: a graphical design tool, a simulation facility and a visualisation tool [12]. The graphical design tool and the visualisation tool should coexist within the same environment to allow information about the program behaviour to be related to its design. Many existing toolsets consist of only a subset of these tools but visualisation is usually a separate tool. In addition, the modelling of the operating system is usually not addressed.
The Transim/Gecko [6] toolset is used to rapidly evaluate different designs of an Occam-like program running on a transputer-based multiprocessor by using the graphical tool (Gecko) to animate the traces generated by the simulator (Transim).
The HAMLET toolset [13] supports the development of real-time applications based on transputers and PowerPCs. HAMLET consists of a design entry system (DES), a specification simulator (HASTE), a debugger and monitor (INQUEST), and a trace analysis tool (TATOO).
HeNCE (Heterogeneous Network Computing Environment) [7] is an X-window based software environment designed to assist scientists in developing parallel programs that run on a network of computers. HeNCE provides the programmer with a high level abstraction for specifying par-

allelism. HeNCE is composed of integrated graphical tools for creating, compiling, executing, and analyzing HeNCE programs. HeNCE relies on the PVM system for process initialization and communication. HeNCE displays an event-ordered animation of application execution.
The ALPSTONE project [8] comprises performance-oriented tools to guide a parallel programmer. The process starts with an abstract, BACS (Basel Algorithm Classification Scheme), description from which it is possible to generate a performance prediction time model of the algorithm on a particular system. This can be helped with a skeleton definition language (ALWAN or PEMPI- Programming Environment for MPI), and a portability platform (TIANA), which translates the program to C with code for a virtual machine such as PVM.
The VPE project [11] aims to design and monitor parallel programs in the same tool. The design is described as a graph where the nodes represent sequential computation or a reference to another VPE graph. Performance analysis and graph animation are not used here, but the design aspect of this work is elaborate.
The TOPSYS (TOols for Parallel SYStems) project [1] aims to develop a portable environment which integrates tools that help programmers cope with every step of the software development cycle of parallel applications. The TOPSYS environment contains tools which support specification and design, coding and debugging, and optimisation of multiprocessor programs. The tools are based on the MMK operating system but were later ported to PVM in [9]. A more detailed review of parallel programming design tools and environments can be found in [3].

3 Description of the EDPEPPS Toolset

The advantages of the EDPEPPS toolset over traditional parallel design methods are that it offers rapid prototyping approach to parallel software development, offers modularity and extensibility through layered partitioning of the model, and allows the software designer to perform the cycle of design-simulate-analysis in the same environment without having to leave the toolset.
Fig. 1 shows the components of the EDPEPPS toolset. The process starts with the graphical design tool (PVMGraph) by building a graph representing a parallel program design based on the PVM programming model. The software designer can then generate C/PVM code (.c files) for both simulation and real execution.

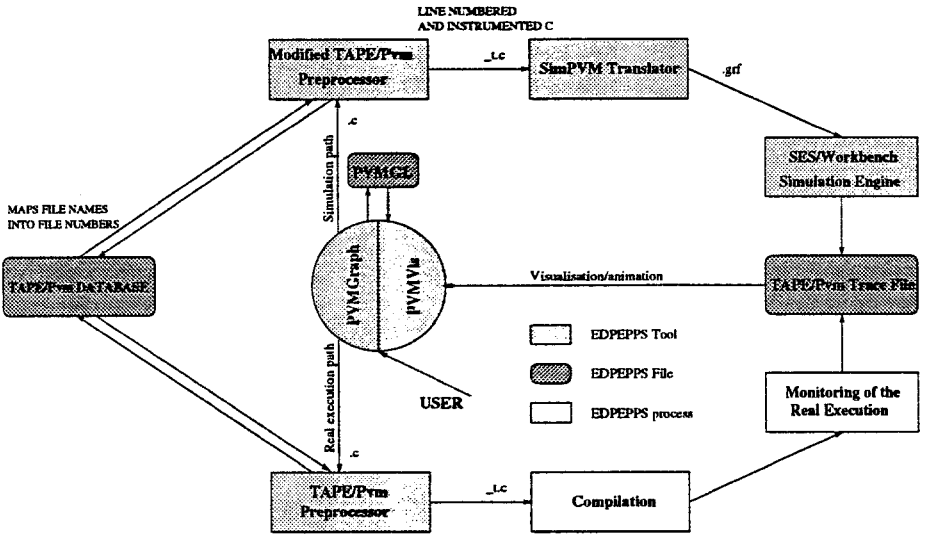

Fig. 1. The EDPEPPS Integrated Environment.

In the simulation path each C/PVM source code obtained from the PVM-Graph is processed using a slightly modified version of the Tape/PVM pre-processor [10]. The instrumented C source files are translated using the SimPVM Translator [3] into a queueing network representation suitable for Workbench graph (.grf file). SES/Workbench uses the graph file to generate an executable model using some SES/Workbench utilities, libraries, declarations and the PVM platform model. The simulation is based on discrete-event modelling.
The simulation output are the execution time, a Tape/PVM trace file and a statistics file about the virtual machine. These files are then used in the visualisation tool (PVMVis) to animate the design and visualise the performance of the system.
In the real execution path the Tape/PVM pre-processor is used to instrument the C source files and these are then compiled and executed to produce the Tape/PVM trace file required for the visualisation/animation process. This step can be used for validation of simulation results but only when the target machine is accessible. The following sections describe the main tools within EDPEPPS.

3.1 PVMGraph

PVMGraph is a graphical programming environment to support the design and implementation of parallel applications. PVMGraph offers a

simple but yet expressive graphical representation and manipulation for the components of a parallel applications. The main function of PVM-Graph is to allow the parallel software designer or programmer to develop PVM applications using a combination of graphical objects and text. Graphical objects are composed of boxes which represent tasks (which may include computation) and arrows which represent communications. The communication actions are divided into two groups: input and output. The PVM actions (calls) are numbered to represent the link between the graph and text in the parallel program. Also different types and shapes of arrows are used to represent different types of PVM communication calls. Parallel programs (PVM/C) can be automatically generated after the completion of the design. Additionally, the designer may enter PVM/C code directly into the objects. The graphical objects and textual files are stored separately to enable the designer to re-use parts of existing applications [3].

3.2 PVMVis

The main objective of this tool is to offer the designer graphical views and animation representing the execution and performance of the designed parallel application from the point of view of the hardware, the design and the network. The animation is an event-based process and is used to locate an undesirable behaviour such as deadlocks or bottlenecks. The animation view in PVMVis is similar to the design view in PVM-Graph except that the pallet is not shown and two extra components for performance analysis are added: barchart view and platform view. The barchart view shows historical states for the simulation and the platform view shows some statistics for selected performance measures at three levels: the message passing layer, the operating system layer and the hardware layer [3].

3.3 SimPVM Translator

From PVMGraph graphical and textual objects, executable and "simulatable" PVM programs can be generated. The "simulatable" code generated by PVMGraph is written in a special intermediary language called SimPVM, which defines an interface between PVMGraph and SES/Workbench [3].
To simulate the application, a model of the intended platform must also be available. Thus, the simulation model is partitioned into two submodels: a dynamic model described in SimPVM, which consists of the

application software description and some aspects of the platform (e.g. number of hardware nodes) and a static model which represents the underlying parallel platform.
The SimPVM language contains C instructions, PVM functions, and constructs such as computation delay and probabilistic functions.

3.4 The EDPEPPS Simulation Model

The EDPEPPS simulation model consists of the PVM platform model library and the PVM programs for simulation. The PVM platform model is partitioned into three layers (Fig. 2): the *message passing layer*, the *operating system layer* and the *hardware layer*. Modularity and extensibility are two key criteria in simulation modelling, therefore layers are decomposed into modules which permit a re-configuration of the entire PVM platform model. The modelled configuration consists of a PVM environment which uses the TCP/IP protocol, and a cluster of heterogeneous workstations connected to a 10 Mbit/s Ethernet network.

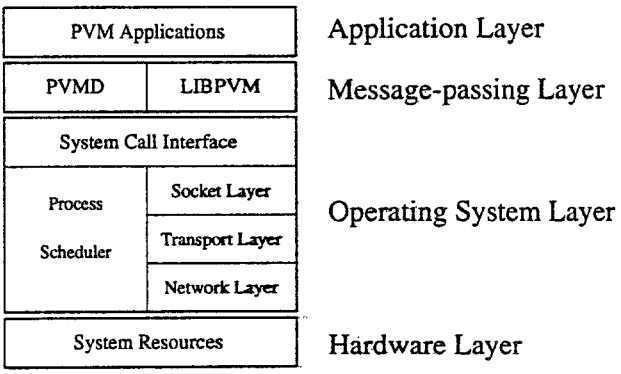

Fig. 2. Simulation model architecture.

A PVM program generated by the PVMGraph tool is translated into the SES/Workbench simulation model language and passed to the SES/Workbench simulation engine, where it is integrated with the platform model for simulation. The message-passing layer models a single (parallel) virtual machine dedicated to a user. It is composed of a daemon which resides on each host making up the virtual machine and the library which provides an interface to PVM services. The daemon acts primarily as a message router. It is modelled as an automaton which is

a common construct for handling events. The LIBPVM library allows a task to interact with the daemon and other tasks.
The major components in the operating system layer are the System Call Interface, the Process Scheduler, and the Communication Module. The Communication Module is structured into 3 sub-layers: the Socket Layer, the Transport Layer and the Network Layer. The Socket Layer provides a communications endpoint within a domain. The Transport Layer defines the communication protocol (either TCP or UDP). The Network Layer implements the Internet Protocol (IP).
The Hardware Layer is comprised of hosts and the communications subnet (Ethernet). Each host is modelled as a single server queue with a time-sliced round-robin scheduling policy.

4 Case Studies

4.1 COMMS1 Benchmark

The COMMS1 benchmark is taken from the Parkbench [5] suite (version 3.0). COMMS1 is designed to measure the communication performance of a parallel system by exchanging of messages of various sizes between two processors. COMMS1 is selected here to highlight the accuracy of the communication model used. Fig. 3 shows the execution time results for COMMS1 benchmark between the predictions and the measurements (averages of 1000 iterations). The figure shows good match between the two curves. The step like features are caused by fragmentation of the message into 1500-Byte segments at the IP level.

4.2 CCITT H.261 Decoder

The application chosen here is the pipeline processor farm (PPF) model of a standard image processing algorithm, the H.261 decoder proposed by Downton et al. [2]. The H.261 algorithm decomposes into a three-stage parallel pipeline: frame initialisation (T1); frame decoder loop (T2); and frame output (T3). The first and last stages are inherently sequential, whereas the middle stage contains considerable data parallelism.
Two experiments for 1 and 5 images were carried out. The number of processors in Stage T2 is varied from 1 to 5. In every case, the load is evenly balanced between processors.
The target platform is a heterogeneous network of up to 6 workstations (SUN4's. SuperSparcs and PC's). Timings for the three algorithm stages were extracted from [2] and inserted as time delays. Fig. 4 shows the simulated and real experimental results for speed-up.

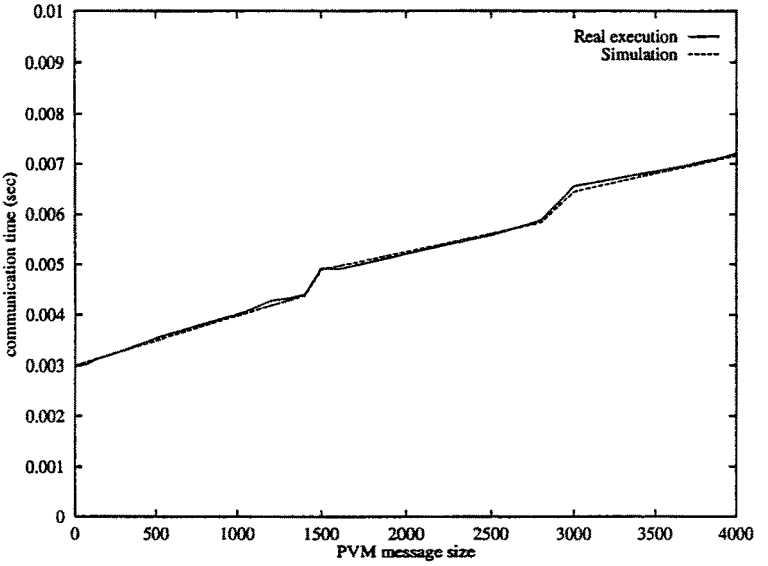

Fig. 3. Comparison between predictions and measurements for COMMS1

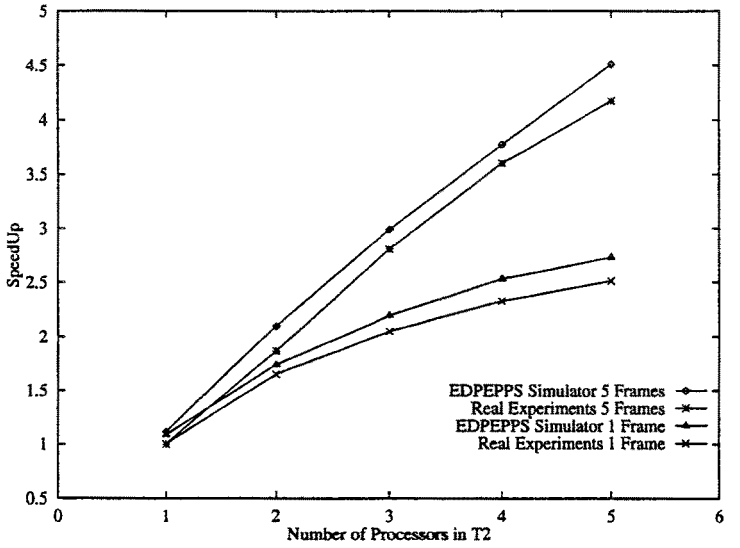

Fig. 4. Comparison between predictions and real experiments for PPF

As expected, the figure shows that the 5-frame scenario performs better than the 1-frame scenario, since the pipeline is fuller in the former case. The difference between simulated and real speed-ups is below 10% even though the PPF simulation results do not include packing costs.

5 Conclusion

This paper has described the EDPEPPS toolset which is based on a performance-oriented parallel program design method. The toolset supports graphical design, performance prediction through modelling and simulation, and visualisation of predicted program behaviour. The designer is not required to leave the graphical design environment to view the program's behaviour, since the visualisation is an animation of the graphical program description. It is intended that this environment will encourage a philosophy of program design, based on a rapid synthesis-evaluation design cycle, in the emerging breed of parallel programmers. Success of the environment depends critically on the accuracy of the underlying simulation system. Preliminary validation experiments showed an error for the PPF model of less than 10% between the simulation and the real execution. CPU modelling, PVM group functions and the simulator speed will be addressed in future work.

Acknowledgments

The authors wish to acknowledge the early contribution of Dr F. Spies. Also T. Delaitre wishes to acknowledge the contributions of his supervisor, Dr. S. Poslad, in the simulation aspects of this work.

References

1. T. Bemmerl. The TOPSYS architecture. In H. Burkhart, editor, *CONPAR90-VAPPIV Conf., Zurich, Switzerland*, Springer, September 1995. Lecture Notes in Computer Science, 457, 732–743.
2. A.C. Downton, R.W.S. Tregidgo and A. Cuhadar, Top-down structured parallelisation of embedded image processing applications, in: *IEE Proc.-Vis. Image Signal Process.* 141(6) (1994) 431-437.
3. EDPEPPS Web Site, http://www.cpc.wmin.ac.uk/~edpepps.
4. A. Geist, et al. *PVM: Parallel Virtual Machine* (MIT Press, 1994).
5. R. Hockney and M. Berry, Public International benchmarks for parallel computers report-1. Tech. Rep., Parkbench Committee, 1994.
6. E. Hart and S. Flavell, Prototyping transputer applications, in: H. Zedan, ed., *Real-time systems with transputers* (IOS Press, 1990).
7. HeNCE Web Site, http://netlib2.cs.utk.edu/hence.
8. W. Kuhn and H. Burkhart. The ALPSTONE project: An overview of a performance modelling environment. In *2nd Int. Conf. on HiPC'96*, (McGraw Hill 1996) 491–496.

9. T. Ludwig, et al. The Tool-Set - an integrated tool environment for PVM. In *EuroPVM'95, Lyon, France*, September 1995. Tech. Rep. 95-02, Ecole Normale Superieure de Lyon.
10. E. Maillet, TAPE/PVM an efficient performance monitor for PVM applications - User guide. LMC-IMAG, ftp://ftp.imag.fr/ in pub/APACHE/TAPE, March 1995.
11. P. Newton, J. Dongara, Overview of VPE: A visual environment for message-passing, Heterogeneous Computing Workshop, 1995.
12. C. Pancake, M. Simmons and J. Yan, Performance evaluation tools for parallel and distributed systems, *Computer* 28 (1995) 16–19.
13. P. Pouzet, J. Paris and V. Jorrand, Parallel application design: The simulation approach with HASTE, in: W. Gentzsch and U. Harms, ed., *HPCN* 2 (1994) 379–393.
14. Scientific and Engineering Software Inc. SES/workbench Reference Manual, Release 3.1, Scientific Engineering Software Inc., 1996.