Developing PVM-Code on Various Hardware Platforms: Portability and Performance

Michael Resch, Alfred Geiger, Manuela Sang

RUS,
Allmandring 30, D-70550 Stuttgart,
e-mail: resch@rus.uni-stuttgart.de, geiger@rus.uni-stuttgart.de,
sang@rus.uni-stuttgart.de

Abstract. PVM has widespread use among the parallel computing community due to its advantage to allow for the development of portable code. This allows programmers to develop codes that may run on different hardware platforms without too much effort being invested into adaption to particular hardware. Originally mainly used in distributed computing in networks of workstations, PVM is meanwhile also available for massively parallel computer systems.
Some hardware vendors have already adapted PVM as a message-passing tool for their own machines. However, using standard PVM still shows some performance degradation on such systems compared to optimized PVM or native message-passing systems.
The paper presented is intended to show performance measurements of PVM on different hardware platforms and to give an insight into how message passing performance may affect real world application performance. Three applications will be presented that have been parallelized using PVM. They can be seen as typical problems of engineering making use of modern numerical techniques, and thus allow for an estimation of the behaviour of typical engineering science codes.

1 Introduction

PVM (Parallel Virtual Machine) from ORNL [1] has become a secret standard for message-passing systems. Since it is freely available, it has spread all over the academic community and beyond. PVM has been ported to a wide variety of currently available machines ranging from workstations to MPP systems. The highlight of PVM has so far been its usability in heterogeneous environments. However, its functionality is limited.

Some investigations have shown that the use of PVM on MPP systems may result in a decrease of performance when compared to native message-passing systems [2]. Depending on the type of application this difference in message-passing performance may result in an overall performance decrease.

However, the new release (PVM3.3.4) shows some promising results in performance on MPP machines. The impact of this improved performance on real world applications is investigated.

2 Message-Passing Systems

We will mainly present performance measurements of native PVM3.3.4. However, for comparison there will be results given for other message-passing systems that are used in the environment of the Computing Center of the University of Stuttgart (RUS).

- Due to its wide-spread use, hardware vendors of parallel machines have implemented specially designed versions of PVM intended to better exploit their hardware-architectures.
  PVM/6000 (PVM6K) is a message-passing system specially designed for IBM Power parallel systems and IBM RISC System/6000 clusters [3]. It is able to take full advantage of fast networks for workstation clusters. Furthermore, PVM6K does not use TCP/IP to exchange data like PVM does; rather, it goes directly to the lowest network layers.
- NX from Intel [4] is an example of a formerly native message-passing system specially designed for Intel's massively parallel systems.

3 Parallel platforms

Message-passing on two parallel hardware architectures was investigated.

- In a joint project of IBM, DLR, RUS and a Special Research Project about "High Temperature Problems of Re-Usable Space Transportation Systems" at the University of Stuttgart, a cluster of 6 IBM RS/6000-550 was installed called PARIS (PArallel RISc). Each of the machines has 128 MB main memory and 1 GB disk space. Only one of the machines allows user login while the others may be used only as additional nodes in a dedicated message-passing environment. All the machines in the cluster are connected via both Ethernet (10 Mb/s) and FDDI (100 Mb/s). Additionally three of them are connected via a special high speed point-to-point interconnect network SOCC (Serial Optical Channel Converter, 225 Mb/s) from IBM.
- The Paragon XP/S is a distributed-memory parallel system. The system installed at Stuttgart consists of 5 service nodes and 72 compute nodes (i860XP) connected via a scalable high-speed network. Each compute node has 32 MB of main memory. The interconnection-network has the form of a 2-D-mesh, the intersection-points of which are formed by mesh-routing-chips with switching times in the range of 10ns. Each router is connected to its four neighbors and one compute-node with a bidirectional link of 200 MByte/s in each direction.

4 Latency and Bandwidth

Basic measurements for latency and bandwidth were done for the architectures and message-passing systems described above to give an impression on the performance of message-passing for the user. Testing was done using a standard ping-pong example for varying sizes of messages.

The table below shows the impact of different networks in the PARIS-cluster. Performance figures are given for different message-passing systems. The comparison shows results for Ethernet, FDDI and SOCC.

Network	Ethernet	FDDI		SOCC	
MP System	PVM	PVM3	PVM6K	PVM3	PVM6K
latency (μsec)	1920	1850	960	1700	535
bandwidth (MB/sec)	1.0	3.2	5.9	3.6	9.3

The specially designed PVM6K is able to exploit the performance of FDDI and SOCC to an acceptable extent. PVM3 does not take care of the underlying hardware and thus, does not yield satisfying results for latency and bandwidth in the benchmark tests.

One interesting point is the performance of PVM compared to that of a native message-passing system on an MPP system. The next table summarizes such results for different hardware and message-passing systems.

Hardware	PARIS		PARAGON	
MP System	PVM3	PVM6K	PVM3	NX
latency (μsec)	1700	535	65	65
bandwidth (MB/sec)	3.6	9.3	52	95

5 Applications

5.1 Multigrid

This example is a model of an application kernel that's typical for modern multi-level algorithms: the solution to a 2-D Poisson-equation with a standard multi-grid method. It puts extreme stress to parallel systems for the following reasons:

- Only one and also a very simple differential equation leads to a very small amount of work per gridpoint and therefore a very bad communication to computation-ratio in domain-decomposition.
- Most work is done on the coarser grid-levels, decreasing vector lengths on the single PEs and also leading to a higher surface to volume ratio per domain.

The intention of this example is to find out whether workstation-clusters and parallel systems with low communication-performance are suited to modern algorithms with low numerical complexity.

As multigrid-methods are in the category of latency-bound algorithms, we did as much latency-hiding as was possible in the underlying message-passing system. With Intel's NX-model this was much easier than with PVM because of the possibility of non-blocking receives, which allow it to very easily overlap the communication with the computation on the inner region of each domain.

As long as we are on finer grid-levels this overlapping will be fine, whereas on the coarsest grid-levels all gridpoints will be situated on the domain-boundaries, thus there is no overlap as there is no more inner region. At any rate all communications can still happen in parallel if the interconnection-network allows this, so we expect a constant communication-overhead if we have at least three domains.

Since our intention is to solve larger problems in the same amount of time on a parallel system we present scaleup results on the Intel Paragon and the PARIS cluster.

5.2 NSFLEX

NSFLEX (Navier-Stokes-Solver using characteristic FLux EXtrapolation) is a solver to simulate compressible inviscid and viscous flow [5]. A finite volume code is used together with a multiblock strategy. An implicit solver runs independently on each block, thus leading to an explicit behaviour. An update of overlapping boundaries is done after each time step. The characteristics of the code with respect to a parellelization and performance analysis can be summarized as:

- A complex equation that has to be solved leads to a large amount of work per gridpoint.
- Since there is only communication involved in updating, a good computation to communication-ratio can be achieved.

The code involves only a very low amount of communication and is therefore very tolerant to the interconnection-network behavior. Thus, performance of the message-passing system used does not have an important influence on the overall performance.

5.3 URANUS

URANUS (Upwind Relaxation Algorithm for Nonequilibrium flows of the Universtity of Stuttgart) is a solver for inviscid or viscous flows based on a fully coupled, fully implicit Godunov-type finite volume algorithm. Parallelization of the code is part of the Special Research Project on "High Temperature Problems of Re-Usable Space Transportation Systems".

So far it has been done for the explicit parts of the code with PVM3 on the above-mentioned Intel Paragon and the PARIS-cluster. Parallelization is based on a domain-decomposition approach. Boundary exchange is done after each time step in order to obtain exactly the simulation results of the sequential version. Optimization and measurements are still in progress (as is the parallelization of the implicit parts), but some first results of performance checks are already available for the workstation cluster.

The goal of the parallelization is to be able to solve for larger grids in a reasonable amount of time. Increasing the amount of work will yield a balanced computation to communication-ratio. Apart from that the following optimization steps seem to be useful:

– Reducing the number of messages to reduce message-passing initialization overhead by collecting data in one message.
– Avoiding synchronous exchange of messages to overcome the network's reduced capabilities of simultaneous message handling on the PARIS-cluster.

6 Conclusions:

PVM has proven to be portable on workstations as well as on MPP systems. Codes written using PVM for message-passing thus can be run on a wide variety of machines without major changes. However, small changes sometimes may become necessary, thus making the handling of large portions of codes, including makefiles, a little bit unpleasant. Furthermore, we saw that usage of standard PVM results in performance degradation on systems were tuned message-passing libraries yield higher bandwidth and lower latency. Thus, if applications are performance-critical, PVM will not be the best choice.

References

1. Al Geist et. al., PVM 3 User's Guide and Reference Manual, ORNL/TM-12187, May, 1994.
2. M. Resch, A. Geiger, J. Zikeli, Message-Passing-Systems on Workstation Clusters and Parallel Computers – The Impact of Software- and Network-Architectures on Applications. In: High Performance Computing and Networking, W. Gentzsch, U. Harms (Eds), Vol. 2, Lecture Notes in Computer Science Nr. 797, 260 - 266, Springer 1994.
3. M. Bernaschi, G. Caliari, G. Richelli, PVM/6000 User's Guide and Subroutine Reference, Release 2.0, February 21, 1994.
4. Intel Paragon OSF/1, FORTRAN System-Calls Reference Manual.
5. A.Bode, M.Lenke, T.Michl, S.Wagner, Implicit Euler Solver on Alliant FX/2800 and Intel iPSC/860 Multiprocessors, in E.H.Hirschel, Flow Simulation with High-Performance Computers I, Notes on Numerical Fluid Mechanics, Vol. 38, Vieweg 1993.

This article was processed using the LaTeX macro package with LLNCS style