„Transportation and Telecommunication in the 3rd Millenium“
10th Anniversary of the Foundation of the Faculty Transportation Sciences
Prague, May 26 – 27, 2003

DESIGN AND PERFORMANCE MONITORING OF A HPC CLUSTER

Ondřej Jiroušek[1]

Abstract: *For high performance computing (HPC), clusters of workstations are becoming very popular platforms, but performance of these systems is harder to predict than on the traditional massively parallel machines. The process by which a low-cost, high performance Beowulf-style Linux cluster was built is discussed in the paper. The first tests of several parallel tasks and the estimation of the performance of the cluster is shortly described.*

Key words: parallel computing, cluster of PC, MPI, scalability, speed-up, linux

1 Introduction

Clusters of personal computers have become the fastest growing approach for building cost-effective high-performance parallel computing platforms. The rapid advancement of microprocessor technologies and high-speed interconnects have facilitated many successful deployments of this type of cluster. Clusters of workstations or PCs are now an alternative to massively parallel supercomputers. This can be demonstrated by the fact, that in the top500 list there are already 93 clusters, with 66,614 processors in total (and the number is still growing) [1]. As the operating system of these supercomputers, Linux is very often chosen. The cluster supercomputer Linux NetworX built for Lawrence Livermore National Laboratory, has been ranked as the fifth fastest supercomputer in the world on the top500 supercomputing list. The 2,304 -processor cluster can process 5.7 Tflop/s running the Linpack benchmark, and is the only Linux-based supercomputer to be ranked within the top five.

It is clear, that for such a great number of processors the cluster set up and management is extremely tedious and error-prone due to the inherent autonomy of the nodes in a cluster and the obtainable scale. For the same reasons, using a cluster is much more difficult than using a traditional supercomputer. It is hence better for the cluster to run an operating system that provides a single system image of the entire cluster. This contrasts with the traditional cluster architecture which is a loose coupling of many individual single user workstations. To provide our cluster with the single system image and thus making it easily manageable and extendable we use the Beowulf Distributed Process Space (BProc) from The Cluster Research Lab [2, 3].

[1]Ondřej Jiroušek, Department of Mechanics and Material, Faculty of Transportation Science, Czech Technical University in Prague, Na Florenci 25, 110 00 Praha 1, jirousek@fd.cvut.cz, Institute of Theoretical and Applied Mechanics, Prosecká 76, 190 00 Praha 9, jirousek@itam.cas.cz

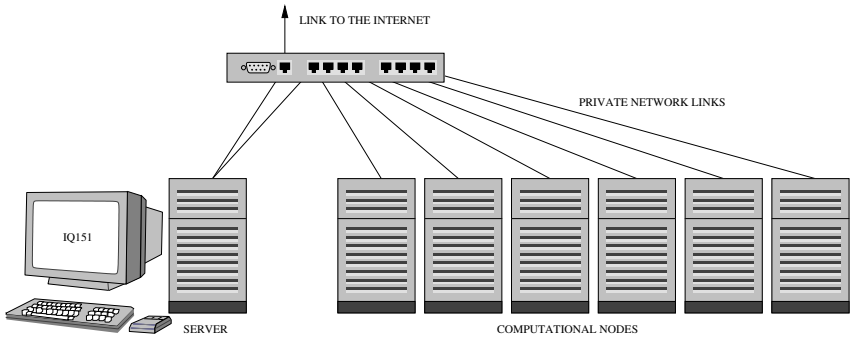

Figure 1: Scheme of the IQ151 cluster

The Beowulf Distributed Process Space provides a single system image of the entire cluster. BProc itself consists of a small set of kernel modifications, utilities and libraries which allow a user to start processes on other machines in a cluster (including reboot). Remote processes started with this mechanism appear in the process table of the front end. This allows remote process management using the normal UNIX process control facilities. Signals are transparently forwarded to remote processes and exit status is received using the usual `wait()` mechanisms.

2 System Building

The cluster consists of 6 computing nodes, equipped with Intel Pentium IV 2.4GHz, 1GB DDRAM, 120GB HDD and 100Mbps Fast Ethernet. The master node is of the same type, equipped with two network cards, one connected to the outside world. The access to the cluster is available only through the master node using the secure shell (`ssh`). To use the cluster, a user can login to the master node (`iq151`), writes a parallel program in fortran or C/C++ using the MPI library and submits the program to the cluster using `mpirun`.

The master node is also a file server for the cluster. The `/home` directory on the master is NFS mounted to the `/home` directories on all the slave nodes enabling users to easily submit their parallel jobs throughout the cluster. Similarly, the master `/usr/local` is also exported to all the nodes, making accessing the parallel applications easier for the users.

The cluster is designed for solving large systems of linear or nonlinear equations arising from the finite element method. For this purpose, we use PETSc [7] (Portable Extensible Toolkit for Scientific Computation) version 2.1.5. PETSc is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It employs the MPI standard for all message-passing communication. The most important features of PETSc include parallel operations with vectors and matrices, scalable parallel preconditioners, Krylov subspace methods and parallel Newton-based nonlinear solvers.

3 Parallel performance measuring methods

To evaluate the parallel performance of the cluster, we first checked one of the PETSc example problems for scalability. The first example we tried is an iterative solution of the Poisson's problem on a uniform mesh in two dimensions. The example can be found in `src/sles/`

Table 1: Speedup of the PC cluster in user times

# processors	small problem		medium problem		big problem	
	time [s]	speedup	time [s]	speedup	time [s]	speedup
1	12.670	1	191.560	1	7236.57	1
2	6.820	1.858	97.020	1.974	3659.24	1.978
3	4.840	2.618	67.150	2.853	2459.68	2.942
4	3.730	3.397	50.710	3.778	1865.24	3.880
5	2.890	4.384	42.840	4.472	1525.81	4.743
6	2.368	5.351	35.435	5.406	1273.82	5.681

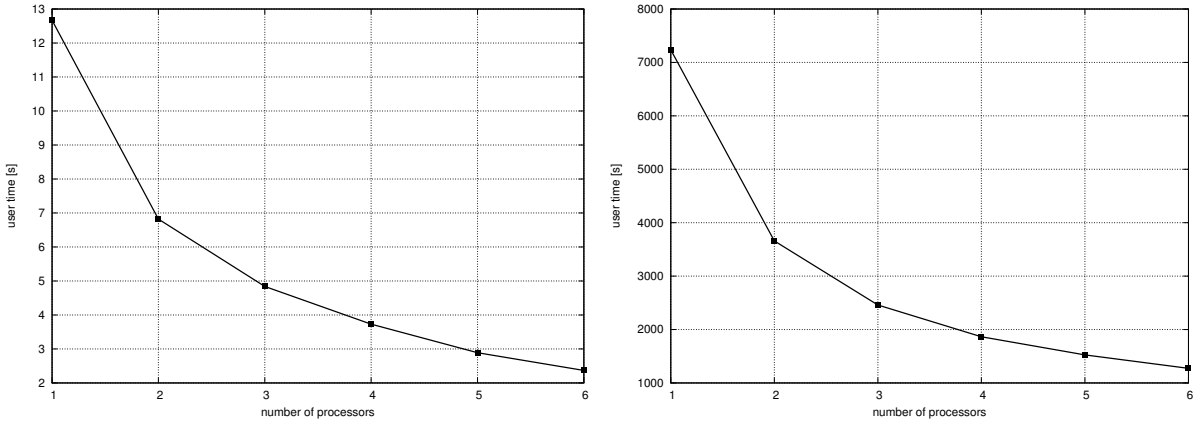

Figure 2: Speed-ups for the small and big problem

`examples/tutorials/ex3.c` in the PETSc version 2.1.5 distribution tree. The PDE is discretized using the standard five point cell-centered finite difference formulation, which produces a linear system of rank $(m)(n)$ to be solved, where m and n are the dimensions of the discretization mesh. The stiffness matrix of the whole problem as well as the right-hand-side vector are assembled in parallel using the `MatCreate`, `MatAssemblyBegin/End`, `VecCreate`, `VecAssemblyBegin/End` functions. The stiffness matrix and the right-hand-side (RHS) vector define the linear system $Au = b$. After the matrix and RHS vector modification for Dirichlet boundary conditions, the system is solved using GMRES (Generalized Minimum RESidual) method.

The problem was solved using one to six nodes of the cluster and the problem rank was varied across $n = m = 100$ (small), $n = m = 200$ (medium) and $n = m = 500$ (big). The performance and the speed-ups were calculated using the total execution (user) time. From these numbers, we computed the MFLOPs rate per processor for each of the test cases and plotted the speed-up curves (Fig. 2).

4 Parallel Benchmarks

The next step in the process of evaluation of the performance was to benchmark the cluster. As the most appropriate we chose the NAS Parallel Benchmarks 2.4 developed by Numerical Aerospace Simulation Systems Division (NAS) of NASA [8]. The goal of these benchmarks is to measure the performance of parallel architecture by running real scientific calculations. These include five kernels (random number generator, integer sort, conjugate gradient, multigrid method for Poisson's equation and FFT for Laplace equation) and three pseudo applications (implicit CFD code).

The benchmarks come in 6 sizes (classes): A, B, C, D, W(orkstation) and S(ample). For classes C and D the disk space required reaches 3 GB and 135 GB respectively. Due to the the huge disk space requirements we were not able to conduct benchmarks of class D. The NAS parallel benchmarks show that the peak performance of the cluster can reach 1.8 Gflops.

Another benchmark we conducted was the Linpack Parallel Benchmark. The benchmark was run on 6 processors with a total memory requirement of 1800 MBytes. The results from the Linpack benchmark showed similar performance as the NAS parallel benchmarks.

5 Discussion and Conclusions

The first experiences with the cluster are very positive. The speed-up for our problems is almost linear and the scalability is also unique. The cluster will be used for finite element calculations of various problems including large nonlinearities, both geometrical and material, large-scale calculations in biomechanics, where geometrically complex models are constructed from computer tomography scans making the resulting mesh often very dense and the number of elements large. It is common for these models to have hundred of thousands nodes.

The cluster was designed as a typical Beowulf [5, 6], making the administration and software maintenance easy. Very positive is the fact, that the cluster is designed to be easily extensible; to plug-in another computational node is virtually a question of seconds.

Acknowledgment: The research has been sponsored by the research fund MSM 210000024 of the Ministry of Education, Youth and Sports of the Czech Republic.

References

[1] Top 500 Supercomputer sites `http://www.top500.org/`

[2] The Cluster Research Lab `http://public.lanl.gov/cluster`

[3] Clustermatic - A complete cluster solution `http://www.clustermatic.org`

[4] Scyld Computing Corporation `http://www.scyld.com`

[5] The Beowulf Cluster Site `http://www.beowulf.org`

[6] The Beowulf Underground `http://www.beowulf-underground.org`

[7] Portable, Extensible Toolkit for Scientific Computation `http://www.mcs.anl.gov/petsc`

[8] NAS Parallel Benchmarks `http://www.nas.nasa.gov/Software/NPB`