Proc. 2st Int. Conf. "Information Technologies and Management", April 15-16, 2004 (Information Systems Institute, Riga, Latvia, 2004) pp. 36-42.

LATVIAN SUPERCLUSTER - LASC: RECENT DEVELOPMENTS

ALEXEI KUZMIN

Institute of Solid State Physics, University of Latvia,Kengaraga Street 8, LV-1063 Riga, Latvia
e-mail: a.kuzmin@cfi.lu.lv , fax: +371 7132778, phone: +371 7251691

ABSTRACT

The solution of many problems in modern science and business requires the use of high performance computing (HPC). Traditionally such problems are addressed using supercomputers. However, recent tremendous increase in a speed of personal computers opens relatively cheap and scalable solution for HPC using cluster technologies. In this paper recent developments in the construction of the cluster system at the Institute of Solid State Physics (ISSP) of the Latvian University (LU) within the Latvian SuperCluster (LASC) project are reported. Two parallel programming technologies, installed on the LASC, are also discussed. The first one is the conventional message-passing interface (MPI), and the second one is the parametric execution system "Spider", developed at ISSP LU. [*Keywords: high performance computing, cluster computing, parallel computing*]

1. INTRODUCTION

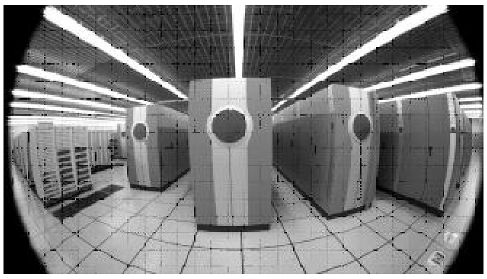
High Performance Computing (HPC) is the technology used to provide solutions to problems that require significant computational power, need to access or process very large amounts of data quickly or need to operate interactively across a geographically distributed network. A range of such problems covers scientific, industrial, business and military applications, and their solution is traditionally addressed using supercomputers.

Starting from 1960s, different supercomputer architectures were suggested. By the end of 2003 [1], the most used ones belong to clusters (41.6%), massive parallel processing (MPP) systems (33.0%) and constellation systems (25.4%). The most powerful today supercomputers are shown in Figure 1. They include Earth-Simulator (35.86 TFlops, 5120 processors), installed by NEC at Earth Simulator Center (Yokohama) in 2001; ASCI Q system (13.88 TFlops, 8192 processors), installed at Los Alamos National Laboratory in 2002 and based on the AlphaServer SC45 computer systems; Terascale Cluster (10.28 TFlops, 2200 processors), located at Virginia Tech and based on 1100 Apple G5 dual-processor systems; Tungsten cluster (9.82 TFlops, 2560 Xeon 3.06 GHz processors), installed at the National Center for Supercomputing Applications (NCSA) at Urbana-Champaign and based on Dell PowerEdge 1750

Proc. 2st Int. Conf. "Information Technologies and Management", April 15-16, 2004 (Information Systems Institute, Riga, Latvia, 2004) pp. 36-42.

servers. As one can see, the cluster technology gains leading positions in the supercomputers competition and becomes more and more popular due to very attractive price/performance ratio and good scalability.

Two types of cluster architecture can be distinguished: the first one is optimised for High Performance Computing, whereas the second one is exploited for High Availability Computing. The choice of the architecture is determined by the type of an application and available budget. A combination of both approaches is utilised in some cases, resulting in a highly reliable system, characterized by a very high performance. The principal difference between these two approaches consists of that in the HPC case, each node in the cluster executes a part of the common job, whereas in the second case, several nodes perform or are ready to perform the same job and, thus, are able to substitute each other in a case of failure. High availability clusters are used in mission critical applications to have constant availability of services to end-users through multiple instances of one or more applications on many computing nodes.


Earth-Simulator
35.86 TFlops, 5120 processors

ASCI Q
AlphaServer SC45, 1.25 GHz
13.88 TFlops, 8192 processors

Terascale Cluster
1100 Dual 2.0 GHz Apple G5
10.28 TFlops, 2200 processors

Tungsten
Dell PowerEdge 1750
9.82 TFlops, 2500 Xeon 3.06 GHz processors

FIGURE 1. TOP SUPERCOMPUTER SYSTEMS IN 2003 [1].

An example of the HPC Beowulf-type cluster is Latvian SuperCluster – LASC (Figure 2 and 3), which was installed at the ISSP LU in 2002 [2,3]. The LASC was strongly optimized for price/performance and is used for scientific calculations in the field of materials science. Below we will provide with the present status of the LASC project and describe two parallel programming technologies available on LASC.

2. LATVIAN SUPERCLUSTER (LASC) PROJECT: PAST AND PRESENT

The main goal of the LASC project is to provide researchers and students at ISSP LU with an access to HPC system, able to help in a solution of physical problems by numerical simulations. The need for such installation is determined by a complexity of tasks appearing now within fundamental and applied research projects in materials science. As such, the main use of LASC is related to quantum chemistry calculations, Monte-Carlo modelling and x-ray absorption spectra analysis. The LASC project is supported by the Excellence Centre of Advanced Material Research and Technology (CAMART), the Institute of Solid State Physics and the Latvian University.

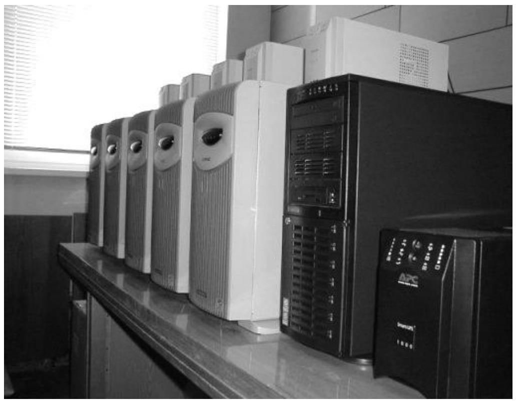


FIGURE 2. LATVIAN SUPERCLUSTER (LASC) AT THE INSTITUTE OF SOLID STATE PHYSICS.

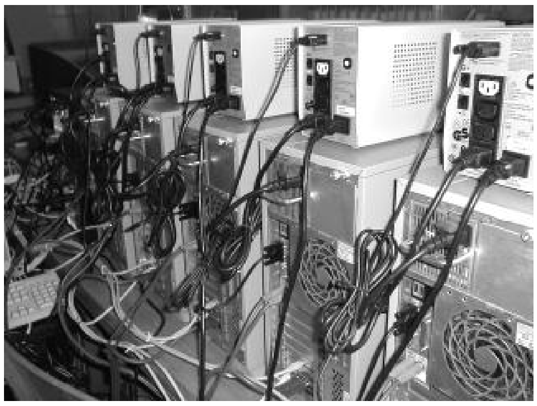


FIGURE 3. LATVIAN SUPERCLUSTER (LASC) FROM BACK-SIDE.

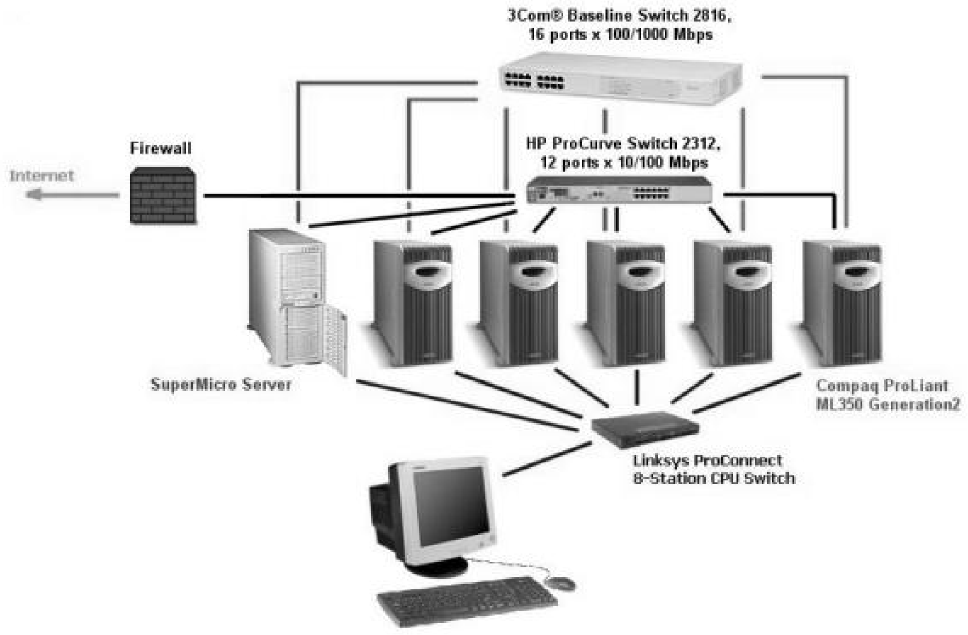


FIGURE 4. LATVIAN SUPERCLUSTER (LASC) ARCHITECTURE IN 2004.

The first turn of LASC, installed in 2002, consisted of five nodes: one front-end node plus four computational nodes. All nodes were dual-processor Compaq ProLiant ML350 G2 servers, connected through the FastEthernet network. The total resources available to the users were 10 CPUs, having a theoretical peak power about 13 GFlops, 20 GB of physical memory (RAM) and 336 GB of storage space on IDE/SCSI hard disks. By the end of 2003, the cluster was significantly upgraded (Figure 4). The new front-end node, having two Pentium 4 XEON 3.066GHz CPUs, 5 GB RAM and the RAID-disk subsystem was installed. The second network, based on Gigabit Ethernet technology, was added for parallel computations tasks, and a new firewall system was installed for cluster protection. As a result, the total computation resources available to the users were significantly improved. At present, the theoretical peak power of the cluster is about 19 GFlops, the total physical memory is 25 GB, and the total disk storage is 1.8 TB. The detailed information on the cluster configuration, its resources, and different useful documentation/links are available at the LASC website [3]. The website provides also with real-time status monitoring of the LASC activity. The access to the cluster is maintained through the simultaneous use of secure shell interface (SSH2) and client's authorisation by the IP address.

The software available to users consists of the GNU set of compilers (C/C++/Fortran77/Pascal) and the Intel's C/C++/Fortran77/Fortran90 compilers. The Intel's compilers have much better degree of optimization for Pentium processors and are able to automatically parallelise the user's code on SMP systems, thus providing with up to 10-50% improvement in the calculation speed compared to GNU compilers. To run programs in parallel, i.e. to use the entire cluster computing resources for a single task, a special software technology is required.

The most known approach is based on the use of special libraries, utilizing so-called "message-passing" ideology. Examples of such libraries include the Message-Passing Interface (MPI) [4] and Parallel Virtual Machines (PVM) [5]. On LASC, the MPICH realization of the MPI, developed at Argonne National Laboratory, is installed and widely used by quantum chemistry applications as Crystal [6] and VASP [7]. The use of MPI technology has strong advantage in that the program can be easily transferred from one supercomputer to another since MPI-specification is standard. However, the use of both MPI and PVM libraries requires strong modification of the program source code. Therefore, if the source code is not available for some reason, such technology cannot be used.

Another approach to the parallel computing is based on a concept of parametric execution system. The idea is to execute the same application many times with different starting parameters on different processors, thus reducing significantly the total execution time. Parametric executions are common in computational modelling, simulations and analysis, and many practical tasks can be reduced to parametric executions, for example, Monte Carlo analysis, design optimization and verification, computational experiments, data mining, searching, combinatorial optimization, what-if scenarios and many others tasks of this nature. An example of commercial parametric execution system is EnFusion, made by Axceleon Inc. [8]. On LASC, the original parametric execution system, called "Spider", was developed at ISSP and can be easily used for a number of different applications.

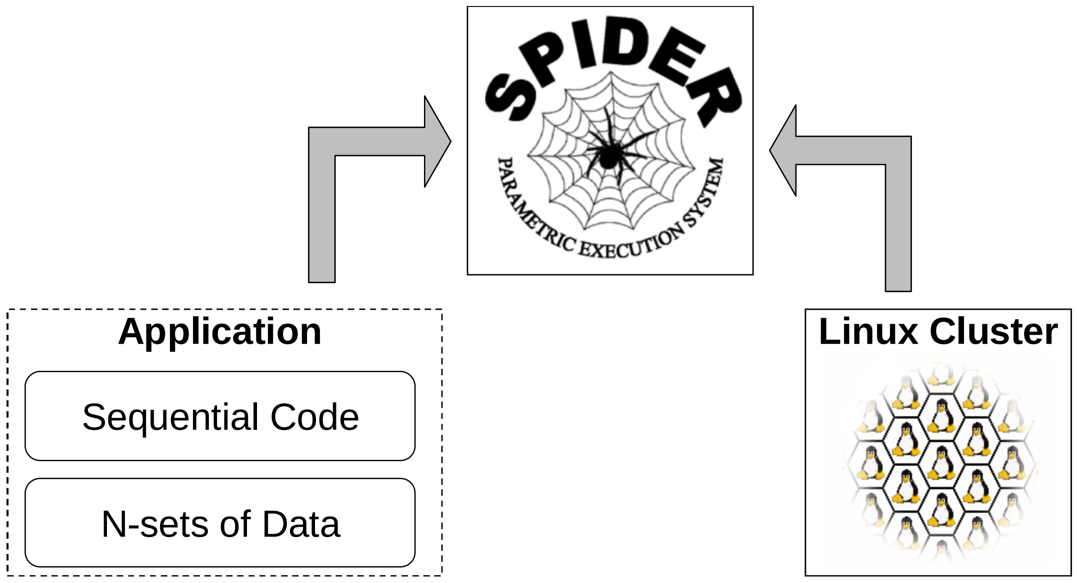


FIGURE 5. BASIC CONCEPT OF THE PARAMETRIC EXECUTION SYSTEM "SPIDER" IS TO MAKE IT EASY TO EXECUTE A LARGE NUMBER OF SEQUENTIAL JOBS OVER A LARGE NUMBER OF COMPUTERS.

Proc. 2st Int. Conf. "Information Technologies and Management", April 15-16, 2004 (Information Systems Institute, Riga, Latvia, 2004) pp. 36-42.

3. PARAMETRIC EXECUTION SYSTEM SPIDER

Basic concept of the parametric execution system “SPIDER” is shown in Figure 5. “SPIDER” is based on the SPMD (Single Program Multiple Data) programming paradigm, and its main goal is to give users the ability to execute a large number of sequential jobs over a large number of computers, in particular organized in a cluster. “SPIDER” can be installed on any homogeneous or heterogeneous cluster, running Linux RedHat 7.x operating system and interconnected via TCP/IP based network with SSH support. The use of secure shell interface for communications between nodes makes it possible to use “SPIDER” not only in a cluster environment but also over public networks such as Internet.

The “SPIDER” system consists of five files: a shell script, a scheduler and three control files (Figure 6). The shell script describes main steps of the calculation, including pre- and post-processing of data, which can be required before and after the parallel task is performed. In particular, the code for input files construction is task dependent and as such should be provided by the user. Additionally, three control files, spider.inp, spider.list and spider.node, should be created as well. These three files provide the “SPIDER” scheduler with the information on (Figure 7):

(1) the name of user’s application to execute and its input and output file names (file spider.inp);

(2) a number of jobs to execute and appropriate filenames for input files, which should be prepared by the shell script before running scheduler (file spider.list);

(3) a list of cluster nodes, which are available for calculations, specified by names or IP addresses (file spider.nodes).

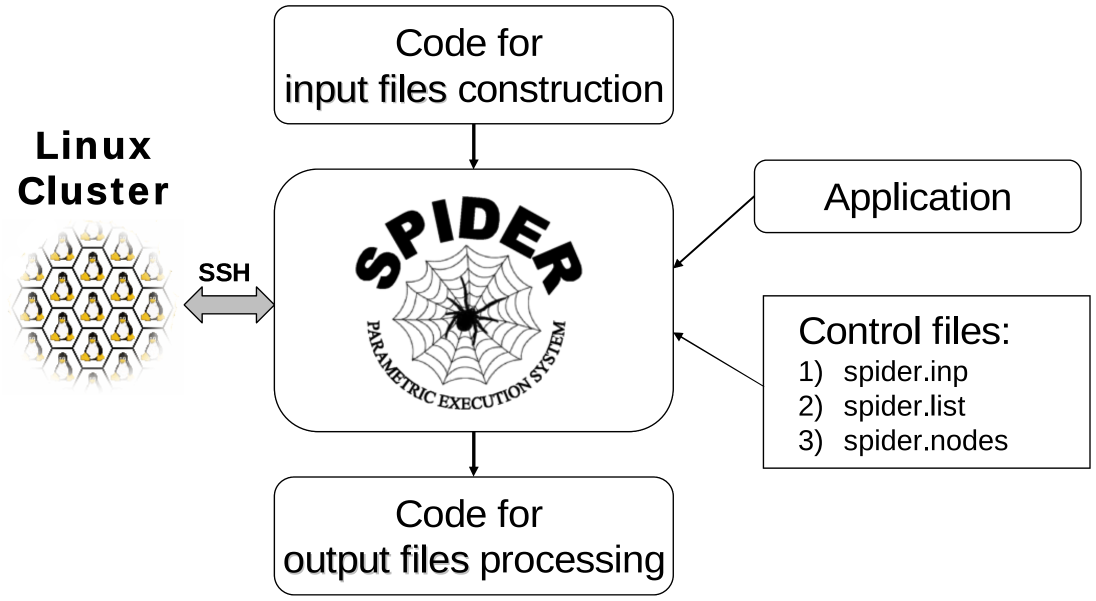


FIGURE 6. “SPIDER” EXECUTION DIAGRAM.

The “SPIDER” scheduler operation occurs in the following way. Let’s imagine an application (called program_name), which requires *n* input files (input_1 ... input_n) to run and produces *m* output files (output_1 ... output_m) after execution (Figure 7). First, the scheduler reads three control files. Next, the scheduler constructs a script to run the user’s application, specified in the file spider.inp, using the first series of the input files (input_1_1 ... input_n_1), taken from the file spider.list. The input files from spider.list are automatically renamed to that required by the application (input_1 ... input_n) and the process is started at the first node (node_1), given in the file spider.nodes. The next copy of the application is started at node_2 with input files: input_1_2 ... input_n_2. When application is finished, the scheduler renames output files output_1 ... output_m into i.output_1 ... i.output_m (i is the number of the finished task, i=1..N) for further processing by the user's specified code in the shell script. If the number of tasks (N) is larger than the number of nodes (L), then the scheduler will start L tasks and will wait for the first free node, before starting next task. Such strategy guaranties that the nodes will not be overloaded and, in heterogeneous cluster environment, the fastest node will be used most often. When all N tasks are finished, the shell script continues for output files processing (Figure 6).

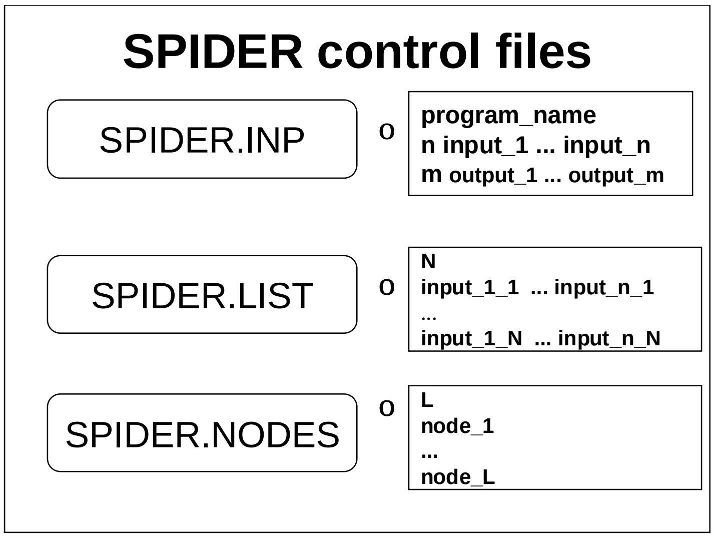


FIGURE 7. "SPIDER" CONTROL FILES AND THEIR STRUCTURE.

At present, the "SPIDER" system is successfully used on LASC for two types of problems. The first one was previously described in [9] and is related to ab initio calculations of x-ray absorption spectra (XAS) from nanoparticles. In this case, the shell script runs first a special code for nanoparticle construction, then the XAS for each atom in the nanoparticle is independently calculated by the FEFF code [10] in parallel way using the "SPIDER" scheduler, and finally another special code calculates the total XAS for the whole nanoparticle as an average of signals for each atom [9].

Another problem addressed on LASC using "SPIDER" is related to a visualisation of simulation results using the rendering software POV-Ray™ [11], a tool for producing high-quality static and dynamic computer graphics. POV-Ray™ [11] is copyrighted freeware and exists for many operating systems as Linux/UNIX, Windows and Macintosh. It is well suitable for artistic, scientific and business purposes, allowing to create stunning three-dimensional graphics. The main problem with using POV-Ray™ on standard single-CPU computers is a long time required by rendering algorithm already for the single scene. Obviously, the time will be even longer for movies creation. However, rendering operation can be easily parallelised. In the case of single static pictures, the picture can be divided into non-overlapping parts, which can be rendered in parallel. In the case of movies, each frame can be rendered independently. The speed of rendering process scales nearly linear with the number of processors, that makes the use of cluster very efficient. The "SPIDER" system allows currently to run sequential version of POV-Ray™ to render in parallel both static and dynamic scenes. The future use of this approach will include direct visualisation of (Reverse-)Monte Carlo or Molecular Dynamics simulations results.

ACKNOWLEDGEMENT

This work was supported in part by the Latvian Government Research Grants No. 01.0807 and 01.0821. The author wish to thank all colleagues from the ISSP LU for support of the LASC project.

Proc. 2st Int. Conf. "Information Technologies and Management", April 15-16, 2004 (Information Systems Institute, Riga, Latvia, 2004) pp. 36-42.

REFERENCES

[1] http://www.top500.org/

[2] Kuzmin A. (2003) Cluster approach to high performance computing. *Computer Modelling and New Technologies* **7,** 7-15.

[3] http://www.cfi.lu.lv/lasc

[4] http://www-unix.mcs.anl.gov/mpi/

[5] http://www.epm.ornl.gov/pvm/pvm_home.html

[6] http://www.crystal.unito.it/

[7] http://cms.mpi.univie.ac.at/vasp/

[8] http://www.axceleon.com/

[9] Kuzmin A. (2003) High-performance computing: application to ab-initio simulations of x-ray absorption spectra from nanoparticles. *In Proc. 1st Int. Conf. "Information Technologies and Management"* (Information System Institute, Riga, Latvia) pp. 44-52.

[10] Ankudinov A.L., Ravel B., Rehr J.J., and Conradson S.D. (1998) Real-space multiple-scattering calculation and interpretation of x-ray-absorption near-edge structure. *Phys. Rev. B* **58**, 7565-7576.

[11] http://www.povray.org/