Annales UMCS Informatica AI XII, 1 (2012) 57–66
DOI: 10.2478/v10065-012-0012-4

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

Influence of batch structure on cluster computing performance – complex systems approach

Paweł Dymora[1*], Mirosław Mazurek[1†], Dominik Strzałka[1‡], Marcin Piękoś[1§]

[1] *Rzeszów University of Technology, Faculty of Electrical and Computer Engineering, Department of Distributed Systems, ul. Wincentego Pola 2, 35-959 Rzeszów, Poland*

Abstract – This work presents the problem of clusters computing performance optimization. An assessment of the impact of computing architecture, applications, and communications equipment on the system performance was done. Processing bottlenecks were identified, showing that in order to obtain an optimum usage of computational structure the system should be treated as a complex one in which it is impossible to isolate and optimize each element independently.

1 Introduction

The development of parallel programming and the increasing demand for computing power forced progress in creating more efficient computer configurations. Such computing power is used to simulate many physical phenomena, chemical reactions or even the fluent management of the increasing databases. To enable processing of very large amounts of data, it is required to create faster processors, memory and I/O subsystems. Pre-viously, supercomputers were used more frequently for high performance computing (HPC), but with time and development of more powerful x86/x64 family processors, the systems using resources of each computer (node in the cluster) were introduced. Cluster solutions provide capabilities and modernization of adding more computational units. On the other hand, as a counter-example the supercomputer is given. It is created on one architecture, and in most cases there is no possibility of parts

*dymorap@prz.edu.pl
†mirekmaz@prz.edu.pl
‡strzalka@prz.edu.pl
§piekosm@prz.edu.pl

replacement for the better, which makes the costs much higher. Usage of supercomputers, in addition to all its advantages, also has some imperfections. These include, for example, the inability to perform hardware upgrades with technological progress. There are also significant costs of both purchase and operation. These are significant drawbacks, especially considering that the remarkable progress in recent years has been done in technology of processors, memory, bus, network, and also the software itself, which created the possibility of linking groups of inexpensive personal computers and workstations into clusters - systems that can successfully compete in their computing power with multiple supercomputers. Popularity of clusters caused an adap-tion of many popular operating systems to support them [**1, 2**].

2 Cluster as a complex system

Computing cluster is a set of computers built from components, interconnected by a fast communication network [**3, 4**]. Each has its own operating system and one (sometimes more) managing and control node. Clusters are used for mass data processing of one type (e.g, scien-tific data and processes visualization). They require specially prepared programs created by specialized software libraries such as Message Passing Interface (MPI) and the Parallel Virtual Machine (PVM) [**5, 6**].

Cluster architecture is critical of its performance, but does not affect the cluster-perception "outside". A parallel programming environment, such as PVM / MPI, and forming a single system of SSI (Single System Image) is responsible for the uniform perception of the cluster. The distribution of processes and optimal allocation of resources between nodes use specifically created implementations of operating system kernels. They are responsible for the transfer of processes from the more loaded nodes to others less busy. In the Mosix imple-mentation such an action does not require any user action, and all orders for resource alloca-tion are executed directly by the Linux kernel. This is done in a transparent manner to the user, as opposed to the static allocation of tasks through the library PVM / MPI [**7, 8**].

Considering the performance of cluster system it should be admitted that it is a typical complex system. Systems complexity may be a result of not only individual variation of basic system components but most of all from the relations occurring between them leading to such phenomena as: sensitivity to initial conditions, emergence, self-adaptation, long-term dependences, nonextensiveness, etc. Properties of individual elements or relations between elements in such a system vary nonlinearly. Complex system is characterized by one or more properties not nec-essarily inferred from the properties of the components, which means that complex systems are difficult to describe by means of conventional physical and mathematical methods, and often the only way to test them is a sort of computer simulation. The simulations show that the system performance is not only influenced by the number and type of processing elements, but mostly the batch which means program and data used in simulations.

Additionally, there will be demonstrated the influence of the communication limits on the overall computational power of the system [**9, 10**].

3 Program and the optimization environment

In simulation tests there were used the three research packets: ATLAS (*Automatically Tuned Linear Algebra*), ACML *(AMD Core Math Library*) and MKL (Intel® Math Kernel Library). ATLAS is both a research Project and the software package. It contains features that optimize multiple mathematical calculations in the field of linear algebra. In cooperates with such programs as Matlab, Octave or Mathematica and contains the module optimizing hardware resources to perform complex operations. It can use most of the processor instructions and allocate enough memory. Owing to such application, it can be used to test complex computing systems and works with such packages as OpenMPI, LAPACK and the R Project. The basic stable versions of ATLAS package are included in the repositories of popular Linux/Unix systems, such as Debian, FreeBSD, MAC OS 10, Scyld Beowolf and SuSe. Using standard processors, it is necessary to provide the adequate compilers Fortran, C/C++, perform the configuration file *./configure-prefix=/installation_ directory/*, and then compile the sources. The compilation process takes a very long time and depends on CPU speed because during the construction all of its parameters are tested and series of performance tests are made [**11**].

ACML is one of the commercial types of mathematical libraries containing ready-made optimization procedures and it is adapted for high performance computing using the AMD processors. It is characterized by full implementation of Basic Linear Algebra (BLAS), a full package of linear algebra procedures LAPACK, a comprehensive set of Fourier transforms(FFT – Fast Fourier Transform), or full scalability of vectors and mathematical arrays. The library can be used on Linux, Solaris, and Windows. It works with such compilers as Gfortran, Intel Fortran Compiler, Visual studio, the PGI compiler and Sun Studio. This means that it can be used for high-performance mathematical calculations and Linpack tests [**12**].

MKL differs from ACML because it has mathematical modules highly optimized for the Intel processors architecture . Unlike ATLAS it does not need to compile, because it has prepared an installer. This library can be used in programming such environments as Visual Studio or Eclipse. It is also fully compatible with OpenMP. Additionally, a big advantage because of the modular form, is possibility of implementing it on all popular operating systems: Linux, Windows and MacOS, Intel after the introduction of new processors to develop new modules that utilize all the instructions of their architecture. Therefore, when upgrading a cluster it is important also to update mathematical modules to use its complete potential. Selection of appropriate library has a significant impact on performance of the computing environment. ACML does not use Hyper Threading technology, while MKL does not support Hyper Transport. This is reflected in the results of performance tests such as the Linpack test. Since the

installation program installs the compiled module, the library can be used only once in compiling HPL, and then copy it to the appropriate folders on the other nodes. This does not affect the perfor-mance of the node – the library is a designer to work with the Intel processors and does not fully support the AMD architecture [**13**].

4 Simulation tests

The simulation studies used the Linpack test (High Performance Linpack), written by Jack Dongarral at the Argonne National Laboratory in the 80s of XX century. The aim of the test was collecting the results of system efficiency tests. The high performance of the LINPACK benchmark (HPL) is the most widely used method for measuring performance of computer systems [**14**]. The computational problem posed by the HPL benchmark is a solution of a system of linear equations, where the coefficient matrix is real, general and dense with random uniform distribution of its elements. Since performance gains can be achieved by sacrificing the correctness of the solution, as a guard against such practices, constraints are imposed on the numerical properties of the solution. In general terms, the answer is correct if it has the same relative accuracy simulation with partial pivoting used in the LINPACK package, when performed in double precision. To be more precise, the following scaled residuals are computed:

$$\begin{aligned} r_n &= \frac{\|Ax-b\|_\infty}{\|A\|_1 \cdot b \cdot e} \\ r_1 &= \frac{\|Ax-b\|_\infty}{\|A\|_1 \cdot \|x\|_1 \cdot e} \\ r_\infty &= \frac{\|Ax-b\|_\infty}{\|A\|_\infty \cdot \|x\|_\infty \cdot e}, \end{aligned} \tag{1}$$

where e is the relative machine precision. A solution is considered numerically correct when all of these quantities are of the order O(1). In calculating the floating-point execution rate, the formula $2n^3/3+2n^2$ is used for the number of operations, regardless of the actual number.

The tests allow to gather a sufficiently large amount of data, depending on the number of nodes and their configuration. The main disadvantage of the Linpack test, for a single CPU and parallel system is the tendency to overestimate real performance for the scientific use and can only expect similar development in such systems. That is because the Linpack code operates on a large matrix, which has beneficial localization of the data. It is not uncommon for the scalable Linpack test, getting 30 percent or greater theoretical efficiency more than the peak performance. For highly optimized code for a given architecture, it rarely manages to achieve more than 10 percent of peak values for modern distributed systems such as cluster system.

The High Performance Linpack (HPL) benchmark is probably the single most studied and scrutinized benchmark in the High Performance Computing community. As the benchmark used for the TOP500 list, produced semi–annually by the University of

Mannheim, University of Tennessee, and NERSC/Lawrence Berkeley National Labs, HPL characterizes the perfor-mance of the world's fastest supercomputers and evaluates the performance of newly con-structed systems under controlled conditions.

5 Results – a single processor analysis

The studies were carried out in several stages. The first of these was to estimate the performance of a single processor, depending on the number of cores and optimization library. Research has shown that the MKL library has the best adjustment, which results in higher computing power of processing unit. Fig. 1. shows the command line options of selected tests for a single node. The parameters N, NB, P, Q adequately describe the size of matrix, the memory block size, number of cores in the processor and the number of nodes participating in the test.

MKL NODE0 1core1node

T/V	N	NB	P	Q	Time	Gflops
WR11C2R4	**10240**	**128**	**1**	**1**	**30.16**	**2.374e+01**

||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)= 0.0057614 ..... PASSED

ACML NODE0 1core1node

T/V	N	NB	P	Q	Time	Gflops
WR11C2R4	**10240**	**128**	**1**	**1**	**79.27**	**9.033e+00**

||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)= 0.0047150 ..... PASSED

ATLAS NODE0 1core1node

T/V	N	NB	P	Q	Time	Gflops
WR11C2R4	**10240**	**128**	**1**	**1**	**37.45**	**1.912e+01**

||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)= 0.0015889 ..... PASSED

ACML NODE0 8core1node

T/V	N	NB	P	Q	Time	Gflops
WR11C2R4	**10240**	**128**	**1**	**8**	**26.91**	**2.661e+01**

||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)= 0.0033508 ..... PASSED

ATLAS NODE0 8core1node

T/V	N	NB	P	Q	Time	Gflops
WR11C2R4	**10240**	**128**	**1**	**8**	**14.79**	**4.841e+01**

||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)= 0.0015357 ..... PASSED

MKL NODE0 8core1node

T/V	N	NB	P	Q	Time	Gflops
WR11C2R4	**10240**	**128**	**1**	**8**	**9.78**	**7.322e+01**

||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)= 0.0034292 ..... PASSED

Fig. 1. Some configurations of the computational test call.

To perform tests there were used computers with various architecture (AMD, Intel), the number of cores (1,2,3,4). As shown in the tests, the performance of a single processor for the same data varies depending on the library as well as processor type, reaching the outcome from 9 Gflop/s (ACML library) to 80,90 Gflop/s (MKL library). It can be noted that with each running process computing performance increases almost linearly. To get the best results the program uses four physical cores. Multithreading negatively affects performance of the processor. Hyper Threading used in the operating systems offloads CPU of excessive operations, which improves its performance. However, it slows down the work of the whole structure in high-performance calculations (Fig.2).

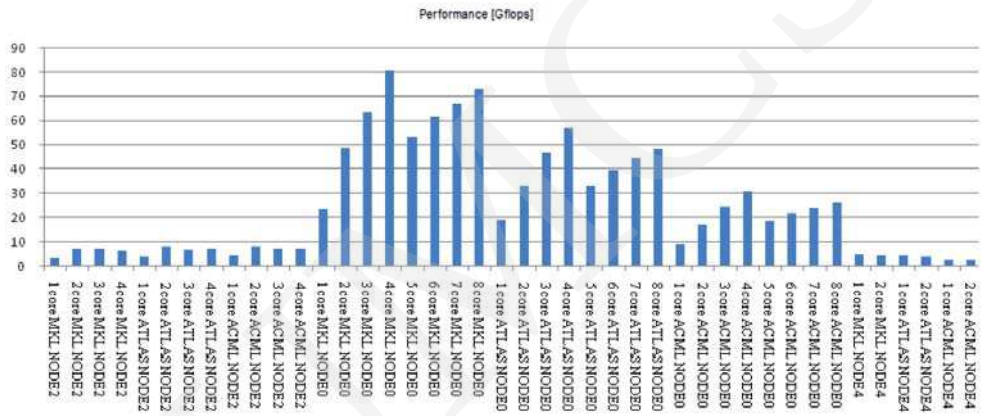

Fig. 2. Influence of optimization library on the CPU performance.

5.1 Performance evaluation of computing cluster

Performance evaluation of computing cluster was done for two the configurations: computing cluster with the nodes connected using the switch 100Mbit/s and computing cluster with the nodes connected using the switch 1 Gbit/s (Fig.3) to build a cluster there were used computers with the same architecture and hardware configuration (Intel i3 3 GHz, 4GB RAM).

As shown in Fig. 4 the computing performance of cluster at the network infrastructure with a speed of 100Mbit/s is significantly lower than using the switches 1Gbit/s.

Communication environment (switches, transmission medium) has a major impact on the obtained results. The study was stopped after connecting the sixth computer unit (six identical processors, using 1GB of RAM on each node), since 1Gbit switch did not allow to transfer data in real time, creating a processing bottleneck. The delays were unacceptable. For testing the dual-core CPUs rate reached 70000 kbit/s (Fig.5), while in the case of single-core pro-cessors for the network 100 Mbit/s the total capacity did not exceed 30 000 kbit/s.

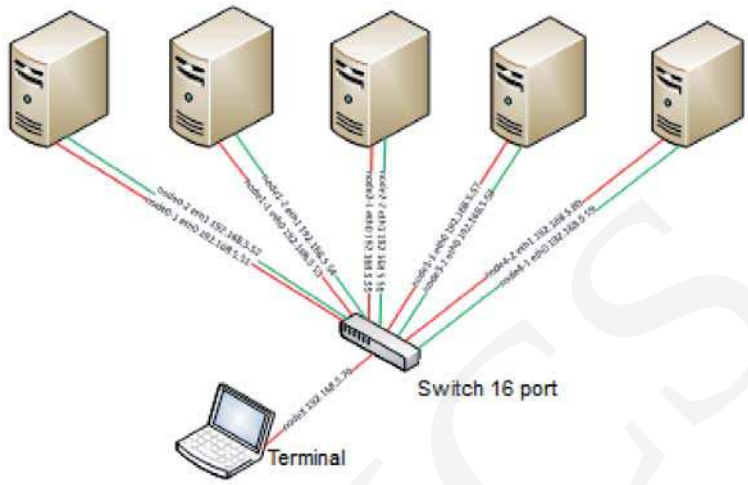

Fig. 3. Cluster computing architecture.

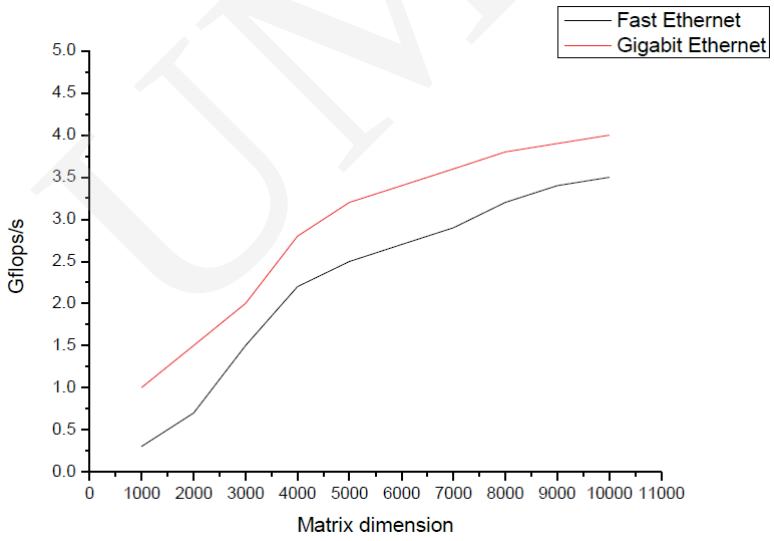

Fig. 4. Comparison of HPL performance with different interconnects: Fast Ethernet, and Gigabit Ethernet.

Influence of batch matching for the cluster architecture shown in Tab. 1. depending on the defined parameters for the number of nodes and processor cores used for a wide range of ob-tained results.

The presented results show that the application to perform the tasks required full hardware optimization or adjustment of the code to the cluster structure, because in

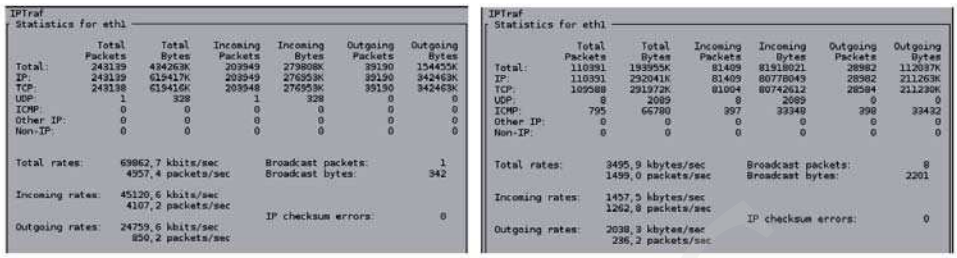

Fig. 5. Analysis of the transmission parameters using the IPTRAF package for transmission of 1 Gbit/s and 100 Mbit/s.

Table 1. Processing time and performance numbers of various nodes and processes.

Nodes/processes	Performance [Gflop/s]	Processing Time [s]
1/1	8.83	81.07
1/2	13.85	51.71
1/4	13.34	53.67
2/2	19.23	37.43
2/4	17.48	40.95
4/2	15.35	46.72
4/4	24.84	28.82
5/1	15.45	46.35
5/2	16.47	43.48

most cases there will not be used full computing power that comes from a cluster. The use of the second CPU core shortened by 30 seconds task realization, and the efficiency increased to 13.85 Gflop/s (at the time of 51.71 sec) from 8.83 Gflop/s (at the time of 81.07 sec) as shown in Tab. 1. This result is slightly worse than that obtained during the implementation of tasks across the cluster (15.45 Gflop/s in time 46.35 s), therefore any slowdown in the transmission of network packets between nodes was omitted. The program executed 10 processes, which is 2 tasks per core. Four cores made the task 14 seconds faster (37.43 sec at 19.23 Gflop/s performance). Simultaneously very visible is increase in the performance of 5.5 Gflop/s (i.e. 60% of single-configuration). Using four nodes shortened the duration of the task by only 4 seconds (15.33 Gflop/s performance, this at the time of 46.72 s), and five by 7 seconds (16.47 Gflop/s per-formance, at the time of 43.48 sec). Hardware limitations are not allowed in further tests with a larger number of nodes, because the communication delays that appeared on the switch became unacceptable.

The presented simulations have shown that the selection of batch determines the optimal use of cluster. As demonstrated by the tests, the optimal configuration of batch depends on the number of its processing units, in our case it was the division

of tasks into 4 nodes with 4 processes. Configuration of four identical copies of the program on each node, did not require full verification of the data between them, which allowed to obtain computing power equal to 24.84 Gflop/s at 28.82 sec. The program IPTRAF showed utilization of network interfaces up to 70%. This caused significant network congestion and the opportunity to obtain quite high results. Division of tasks into four nodes with 4 processes allows to get computing performance nearly 100% higher than in the case of the single-node configuration.

Multithreading processors used in the cluster enabled the task accomplishment in time not much longer than that achieved in the previous tests. Distribution of tasks in 8 processes on 2 cores with Hyper Threading enabled 13.34 Gflop/s performance within 53.67 sec. Multithreading processors enabled the implementation of the task in time not much longer than in the case of four processes running on two cores (17.48 Gflop/s performance within the time of 40.95 sec). Also the tested conditions were characterized by not optimal distribution of load on computational elements. There was simulated implementation of sixteen processes on twelve processors (6 cores with Hyper Threading). The result decreased in comparison to the execution on one processor achieving 11.10 Gflop/s performance within 64.53 sec. The low performance was influenced by the transmission network, which has become the bottleneck of computing structure.

6 Summary

The simulation studies performed on the cluster showed that considering performance of the cluster system, it must be assumed that this is a typical complex system. Its complexity results not only from individual variation of the system components, but primarily from the relations between the elements, leading to such phenomena such sensitivity to initial conditions, emergency, self-adaption, long-range dependences, nonextencivity, etc.. Complex system is characterized by one or more properties not necessarily inferred from giving the properties of components, which makes the complex system hard to describe with the methods of classical physics and mathematics, and often the only way to test them is a sort of computer simu-lation. The simulations have shown that the system performance is not only affected by the number and type of processing elements, but mostly the same "load" which means program and data used in the simulation.

In order to increase the computing performance of cluster there are used multi-core processors with the division of threads technology, such as Hyper Transport and Hyper Threading. Hyper Transport helps avoid the problem of common bus, and enables better utilization of memory. However, Hyper Threading allows the decrease in productivity loss during the launch of more processes. In the case of node communication, the amount and the quality of connections that have a major impact on the communication delay are important. In the case of assembling permanent, excessive traffic on the switch, an extra node or route traffic to another, less loaded one should be used. To

improve communications reliability and performance there can be used more and faster network adapters or increase of their number. To use the full possibilities of multi-core processors, it becomes necessary to change the overall network model, technology by Myrinet or Infiniband.

Acknowledgements

Equipment purchased in the project No POPW.01.03.00-18-012/09 from the Structural Funds, The Develop-ment of Eastern Poland Operational Programme co-financed by the European Union, the European Regional Development Fund.

References

[1] Sterling T., Beowulf Cluster Computing with Linux, Massachusetts Institute of Technology (2002).
[2] Wilkinson B., Allen M., Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers (1999).
[3] Yang C. T., Tseng S. S., Fan Y. W., Tsai T. K., Hsieh M. H., Wu C. T., Using Knowledge-based Systems for research on portable parallelizing compilers 13 (2001).
[4] Kopper K., The Linux Enterprise Cluster - Build a Highly Available Cluster with Commodity Hardware and Free Software (2005).
[5] Lai C. L., Yang C. T., Construct a Grid Computing Environment on Multiple Linux PC Clusters, International Conference on Open Source (2003).
[6] Yang C. T., Tseng S. S., Hsiao M. C., A Portable paralleling compiler with loop partitioning (1999).
[7] Morrison R. S., Cluster Computing, Architectures, Operating Systems, Parallel Processing & Programming Languages (2003).
[8] Gropp W., Lusk E., Doss N., Skjellum A., A high-performance, portable implementation of the MPI Message-Passing Interface standard (1996).
[9] Dymora P., Mazurek M., Strzałka D., Statistical mechanics of memory pages reads during man–computer system interaction, Metody Informatyki Stosowanej 1(26) (2011):15.
[10] Strzałka D., Grabowski F., Non-Extensive Thermodynamics of Algorithmic Processing - the Case of Insertion Sort Algorithm, in Thermodynamics, ed. Tadashi Mizutani, InTech (2011).
[11] ATLAS (Automatically Tuned Linear Algebra); http://math-atlas.sourceforge.net/.
[12] ACML; http://developer.amd.com/libraries/acml/pages/default.aspx.
[13] MKL (Intel®Math Kernel Library); http://software.intel.com/en-us/articles/intel-mkl/.
[14] Kurzak J., Dongarra J., Implementation of the Mixed-Precision High Performance LINPACK Benchmark on the CELL Processor (2006).