Enhancing Energy sector efficiency: a study on supercomputer performance in optimizing energy systems

Mikhail Lavrentiev*, and Alexey Snytnikov

Institute of Automation and Electrometry SB RAS, 630090 Novosibirsk, Russia

Abstract. The paper discusses the question of how to evaluate actual performance of a supercomputer system, which is expected on solution to real scientific or engineering problem. In practice this actual performance is rather far from the rather well-known peak performance. In particular, the paper contains a review of the computational problems most commonly solved on petaflops supercomputers as well as the corresponding methods designed for exaflops supercomputers. A proposed technique for measuring the characteristics of the supercomputer using a Particle-In-Cell (PIC) code is described. The choice of this particular numerical method is based on its features, namely that PIC method involves a wide variety of numerical techniques, and thus it is very hard to optimize. The analysis of scalability, parallel efficiency and acceleration rate, which is possible at a particular supercomputer is presented, as well as the analysis of the performance of multi-architecture supercomputer nodes. The integrated criteria to evaluate the real performance of supercomputer system is proposed.

1 Introduction

Since last decades words High Performance Computing (HPC) have become quite common. The lists of Top 500 (see http://top500.org) of the most productive cluster systems and the lists of Green-500 (cf. http://green500.org) of the most energy-efficient computer systems are being published on a regular basis. A supercomputer (say, HPC system) is a computer with performance greater by an orders of magnitude compared to the performance of a general-purpose computer. The performance of a supercomputer is measured in Floating-Point Operations per Second (FLOPS). Supercomputer hardware consists of processor elements: multicore processors (like Intel Xeon and IBM Power, e.g.) or computation accelerators (like GPGPUs – General Purpose Graphic Processing Units – or MIC-based Intel Xeon Phi's). Second important part of the hardware is the interconnect – the internal network of the supercomputer that connects the processor elements, hard disks and input-output (IO) system. Modern interconnect is usually based on Infiniband or Gygabit Ethernet or some other systems. Most of the world's fastest supercomputers are using UNIX-based operating systems. Most powerful supercomputers today are of Petascale (1015 FLOPS) performance, and development of Exascale (1018 FLOPS) systems are under way.

* Corresponding author: mmlavrentiev@gmail.com

As a rule, HPC systems are compared it terms of the so-called Peak Performance. This is top performance of each CPU (Central Processing Unit) multiplied by the number of CPUs in the entire system, a performance almost infeasible in real computations. There is a sort of competition among world economy leaders for the rank in the list of the most powerful computation systems. There days creation of "exascale" class system is under discussion. At the same time, a very few practical problems require all the capabilities of super computers from the Top 500. Most applied and research challenges can be effectively solved with an application of modern personal computers with the use of proper software (algorithms) and hardware (architecture) acceleration.

Supercomputer performance is determined by running performance tests like HPL [1], HPCG [2], NAS Parallel benchmarks [3], and many other.

The most frequently used performance tests, like HPL, at the moment show quite high supercomputer performance of the modern HPC systems. At the same time, the real computational applications give out no more than 10% of the so obtained Peak Performance.

Based on a generalization of simulation experience in astrophysics and plasma physics, a method for determining the actual performance of supercomputers is here proposed. We understand actual performance as the one obtained solving numerically problems of mathematical physics. This method is based on measuring the operating time of Particle-In-Cell (PIC) numerical method, one of the most commonly used in numerical simulation. Generalization to other types of numerical methods could be also done.

First we need to say about why PIC [4] method was chosen for building benchmark and what makes parallel program implementing the PIC method in plasma physics an ideal test application for supercomputers. The Particle-In-Cell [4] method is a combination of a number of numerical methods, namely finite-difference methods for solving differential equations describing physical fields, various interpolation options for calculating the values of physical quantities (current and density in our case of plasma physics), and a wide range of methods for integrating the equations of motion of model particles. For the parallel implementation of the PIC method, both coarse-grained and fine-grained parallelism is used, thus the PIC method covers all issues of supercomputer implementation of numerical methods. Here, coarse-grain parallelism means parallelization of a computational algorithm by means of domain decomposition and processing different subdomains by different processes. Another main type of parallelism, the fine-grain parallelism, means that elementary objects like model particles, mesh nodes, finite elements are processed in parallel by different processes, threads etc.

There is one more answer to the question: why do we propose PIC method. One must notice that PIC method is extremely hard for automated optimization either by compiler options (loop unrolling, constant propagation, vectorization etc.) or by processor-level operations like prefetching or cache utilization. Application of all these techniques will produce almost no result on performance of PIC implementation. It means that testing a supercomputer with PIC method will give results depending on the software/hardware configuration, but not on the compiler efficiency or on operating system performance.

Due to the fact that the PIC method creates a "bad" (very irregular) memory access pattern, it is almost impossible to optimize the code using cache, prefetching and other compiler-dependent optimization techniques. PIC method has no preferable architecture, that means no architecture suits it very well. In fact, it is good for a benchmark since it means that for every architecture the PIC-based benchmark will measure hardware performance and not the quality of the local compiler and other system software. Finally, we can state that the performance obtained using PIC method is actually a lower bound for the other computational methods.

Analysis of the papers devoted to Petaflops and Exaflops computing revealed the following main areas of interest. Most papers deal with the Computational Fluid Dynamics

targeting, for example, on simulation of the flow around ships and aircrafts [5], ocean currents computation [6], flood prediction in coastal cities of Northern Europe and other environmental issues. Computation of hydrodynamic turbulence [6,7], modeling a multiphase flows [8], the construction of grids for hydrodynamic modeling [9] also attract attention of researchers. The second most frequent area, is the modeling of various processes in nuclear reactors, e.g. [10,11]. As the third area, one must mention plasma simulation, like processes in controlled thermonuclear fusion facilities [12,13,14]. Finally, most computations with present Petascale computers were devoted for the development of new drugs and other biomedical applications [15,16].

Rest of the paper is arranged as follows. First, we describe the parallel implementation if PIC method solving plasma physics equations. Then results of numerical experiments at several cluster systems are presented. The integrated criteria to evaluate performance of supercomputer system is formulated. Finally, we compare several clusters in terms of some well-known benchmarks and the proposed criteria.

2 PIC method implementation and parallelization

PIC [4] method is used to solve Vlasov-Maxwell system of equations describing the dynamics of a collision-less plasma. The implementation is done basically as described in [4], well-known manual of a mathematical model of high-temperature collision-less plasma. The model consists of the Vlasov kinetic equation and system of Maxwell equations, which in dimensionless form have the following form:

$$\partial f_{i,e} / \partial t + \mathbf{v} \partial f_{i,e} / \partial \mathbf{r} + \mathbf{F}_{i,e} \partial f_{i,e} / \partial \mathbf{p} = 0 \quad (1)$$

$$Fi,e = qi,e(E+1/c[v,B]) \quad (2)$$

$$rotB = 4\pi j + 1/c \, \partial E / \partial t \quad (3)$$

$$rotE = -1/c \, \partial B / \partial t \quad (4)$$

$$divB = 0 \quad (5)$$

In the present work these equations are solved by the method described in [4], it includes Finite Difference Time Domain method [17] for Maxwell equations and PIC method for Vlasov equations. Some details of the implemented particle pusher are given below. In the sequel all the equations will be given in the non-dimensional form. The following basic quantities are used for the transition to the non-dimensional form:

- characteristic velocity is the velocity of light $v_{ch} = c = 3 \times 10^{10}$ cm/sec
- characteristic plasma density $n = 10^{14}$ cm^{-3}
- characteristic time t_{ch} is the plasma period (a value inverse to the electron plasma frequency) $t_{ch} = \omega_{pe}^{-1} = (4\pi n_0 e^2/m_e)^{-0.5} = 5.3 \times 10^{-12}$

The PIC method to solve Vlasov equations implies the solution of the equation of movement for all model particles. The movement equations of the particles are the equations of characteristics for the Vlasov equation.

The scheme proposed by Langdon and Lasinski [18] is used to obtain the values of strength of electric and magnetic fields. The scheme employs the finite-difference form of the Faraday and Ampere laws:

$$(Bm+1/2 - Bm -1/2) / \tau = - rotkEm \quad (6)$$

$$(Em+1 - Em)/\tau = -4\pi jm+1/2 + rotkBm+1/2 \quad (7)$$

A detailed description of this scheme is given in [14]. The scheme has the second order of approximation both with respect to space and time.

Let us discuss briefly parallel implementation of the PIC method, graphically shown in figure 1.

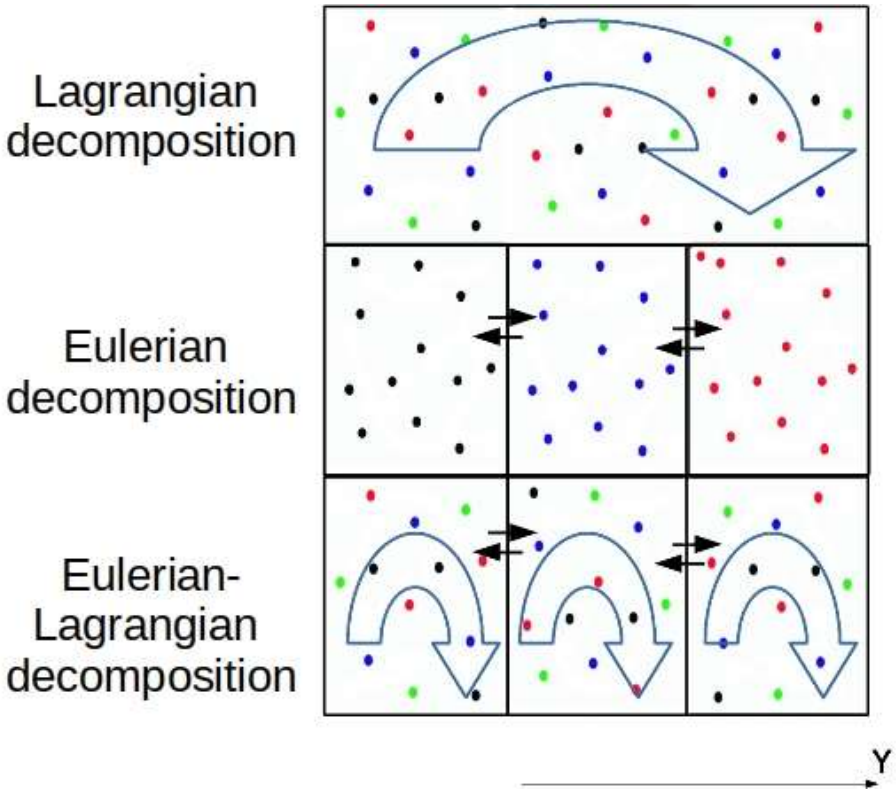

Fig. 1. Domain decomposition used in the code under discussion. Circles of different colours denote the model particles that belong to different processes. Small black arrows show peer-to-peer MPI operations (MPI_Send/MPI_Recv), bent hollow arrows show MPI Allreduce operation.

Three types of domain decomposition were implemented:

1) Eulerian decomposition: computation domain is divided between processes uniformly along one of the coordinate axis (ordinate Y in the present case), and the model particles are distributed among the processes according to their position in the domain. Since the computation domain is divided into subdomains, it is required to send and to receive operations for evaluation of fields and for currents at the subdomain boundaries. The massive send-receive operations cause a lot of traffic, but only between the adjacent processes in the parallelization algorithm. In such a way the Eulerian decomposition should be generally preferred in comparison to the Lagrangian decomposition (described below), which involves global information exchange.
2) Lagrangian decomposition: the whole computation domain is stored at all the processors, just the particles are distributed uniformly between the processors with no regard to their position in the domain. In this case parallelization is performed with CUDA or OpenMP (these are the technologies for fine-grain parallelism), depending on the hardware platform in use. Since the particle ensemble available for each process gives just a part of the current (or density) in the whole domain, this type of decomposition requires collective MPI operations like MPI_Allreduce to sum up the currents. Collective operations usually result in poor performance, because they need

synchronization of all the involved processes and nodes, and a lot of data being transferred among them.

3) Original mixed Eulerian-Lagrangian decomposition. First, we divide the computation domain into parts, second the particles of each subdomain are distributed between CUDA threads or OpenMP threads. Another option is to distribute the particles in subdomain among a group of MPI processes. Both peer-to-peer and collective communications are required.

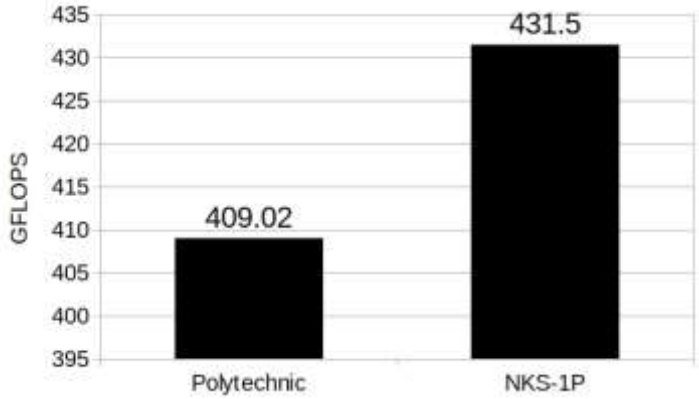

Fig. 2. Processor performance for two different clusters: Polytechnic (Intel 2697 v3 processor) and NKS-1P (Intel Xeon E5-2697A v4 processor).

Now let us discuss the performance evaluation within PIC-benchmark – how are we going to use the above PIC method implementation to measure the performance of a cluster. It is possible to measure time spent for particle push during a single timestep of the PIC method. About 440 floating point operations are performed in our code to move one model particle during a single timestep. This number is obtained by direct summation of the arithmetic operations in the code. Thus, provided that the number of model particles is given, it is possible to find out the total number of floating point operations performed during one timestep, and then, dividing this number by the timestep duration, we find out the computation performance in FLOPS. Figures 2 and 3 show single processor performance for several different clusters in Russia, see table 1 for details.

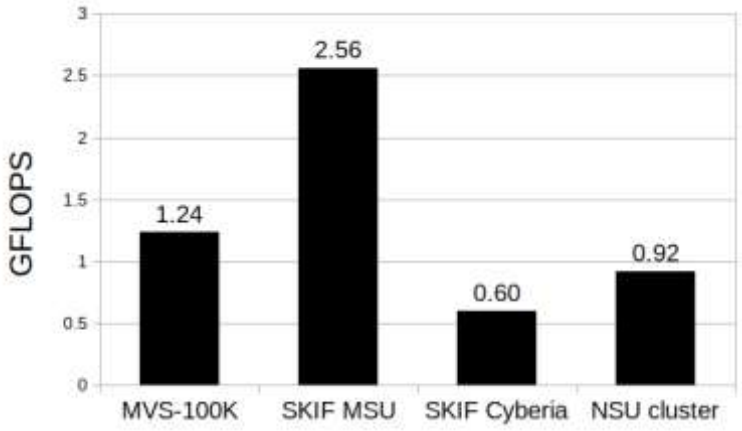

Fig. 3. Processor performance for some of the older cluster, details are given in Table 2.

In a similar way, provided that the information about how much data is sent to other MPI processes, and the time required for the send/receive operations, it is possible to compute the

network performance. At each timestep some model particles leave the subdomain attached to the present MPI process, and move to some other subdomain (and, therefore, to other MPI process). Each model particle takes 48 bytes, and on average 5% of them move to other subdomain at every timestep. Some data transfer occurs, of course, during Maxwell equation solution, but it is not discussed here, since the amount of these data is much less compared to particle transfer. The network performance obtained in such a way is shown in figure 3 for several cluster systems in Russia, see Table 2.

3 Comparative characteristics of clusters for numerical experiments

The code under discussion was used as a performance test for several HPC systems located in Russia. The present section gives their parameters and performance results. These systems were selected for performance testing for two reasons: first, they are the systems used for our PIC plasma simulations, and second, they have similar hardware but very different performance. Moreover, we have two groups of systems in this section, modern HPC systems, and older ones built with the same type of architecture.

Table 1. Main parameters of Polytechnic cluster and NKS-1P cluster.

	"Polytechnic" cluster	**NKS-1P cluster**
URL	https://research.spbstu.ru/skc/	http://www.sscc.icmmg.nsc.ru/main.html
Location (the institution)	Saint-Petersburg State Polytechnic University, SPb, Russia	Siberian Supercomputer Center, Novosibirsk, Russia
Main Processor	Intel Xeon E5 2697 v3	Intel XeonE5-2697Av4
Position in Russian Top50 list	5	49
Number of processors	1528	70
LinPack perf., TFLOPS	801.63	56.95

The first cluster is referred to as Polytechnic (Intel Xeon 2697 v3 processors). It is installed in Saint-Petersburg State Polytechnic University (Saint-Petersburg, Russia). The second cluster NKS-1P (Intel Xeon E5-2697A v4) is from the Siberian Supercomputer Center (Institute of Computational Mathematics and Mathematical Geophysics Siberian Branch RAS, Novosibirsk, Russia).

Table 2. Parameters of the clusters under consideration.

Cluster system	**Main processor**	**Interconnect**	**Location**	**LinPack Perf., TFLOPS**	**Russian Top 50 (2010)**
MVS-100K	Intel Xeon E5450	Infiniband 4x DDR	Joint Supercomputing Center, Moscow	95.04	1
SKIF MSU	Intel Xeon E5472	InfiniBand	Moscow State University, Moscow	60	2
SKIF Cyberia	Intel Xeon 5150	QLogic InfiniPath	Tomsk State University,Tomsk	12.002	14
NSU cluster	Intel xXeon 5355	Infiniband 4x DDR	Novosibirsk State University, Novosibirsk	5.4	20

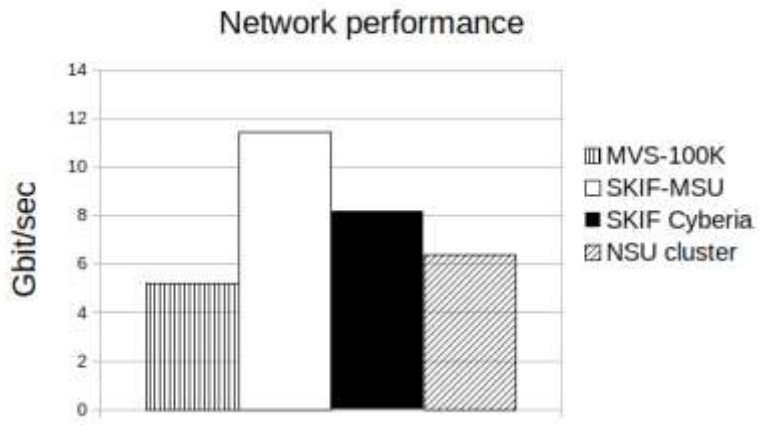

Fig. 4. Supercomputer interconnect network performance for older clusters (Table 2) measured on the basis of Particle-In-Cell code.

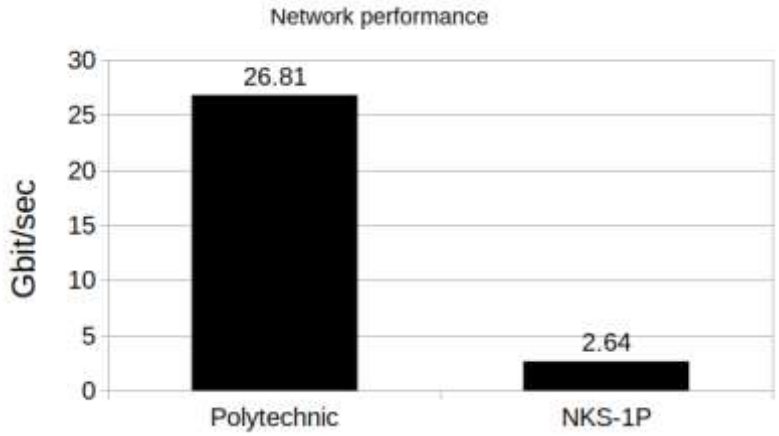

Fig. 5. Supercomputer interconnect network performance for modern clusters measured on the basis of Particle-In-Cell code.

As is clear from Figures 4 and 5, for the rather older systems (Figure 4) variety of network performance is comparatively small, nearly two times between the fastest for SKIF-MSU and the slowest of NSU cluster. On the contrary, network performance for Polytechnic cluster is 10 times faster compared to the NKS-1P, see Figure 5.

As a conclusion to the present section, we should say that the clusters listed in Table 2 have pretty similar processors, yet quite different performance. Moreover, Linpack performance data say little or even nothing about the network performance of the cluster. The benchmark proposed below in the paper has better sensitivity for both issues: we show different CPU performance when it's really different and we analyze network performance at the same time.

One more reason why it is not possible to be content with universal tests like LinPack only is the following. For example, if you use one fourth of the processors cluster with a peak performance of 5.4 TFLOPS (1 TFLOPS is 1000 GFLOPS), you may think that the peak performance of the used part of the cluster should be on level 1 TFLOPS. In fact, in the problem under consideration, 0.18 TFLOPS is obtained, which is caused not only by user program flaws, but also by the fact that the cluster is not suitable for this type of task.

4 The integrated balance criteria to evaluate the real performance of supercomputer system

It is obvious that for the optimal (most complete) use of the potential of all components of a supercomputer system in the case when the computational algorithm requires often random access to external memory, the overall performance of computing nodes should be comparable to the speed of memory access and, at the same time, with the performance of the data exchange network. Of course, for certain classes of problems, the use of all the capabilities of a supercomputer system is possible even if these parameters differ significantly. For example, if you only need to perform massive operations of integer comparison of sets of "samples" with the original DNA sequence (the so-called search for motifs in the genomic sequence), access to external memory is practically not required. In such occasion the system can be fully loaded in the case of orders of magnitude lower memory access speed compared to the total performance of computing cores. Accordingly, you can come up with tasks where you need an increased speed of memory access. In case of insufficient speed of such access, some of the computing cores will simply stand idle.

For overall evaluation of a supercomputer system solving the problems of computational physics the following quantities should be accounted:

- TPP – Total processors performance, evaluated in GFLOPS (for a single CPU),
- RAM – Random-access memory throughput, which is measured in gigabits per second,
- NP – Network performance, also measured in gigabits per second.

It is natural to suppose that all the three values above must be in balance, say, they must be equal, or at least should have the same order of magnitude. Obviously, high performance in FLOPS is useless if the necessary data are delivered from other nodes of supercomputer much slower than they are processed.

If we talk about a balanced supercomputer system that allows almost full use of its capabilities on a wide class of computational problems in mathematical physics, the ratio of the total performance of computing nodes and the speed of memory access should be close to 1.

Here below the following Balance Criteria (BC), characterizing mutual compliance among the above three basic subsystems of a supercomputer is proposed:

$$BC = 100\% \min(C1, C2), \tag{8}$$

where $C_1 = \min(TPP/RAM, RAM/TPP)$, $C_2 = \min(TPP/NP, NP/TPP)$.

Table 3 presents the values of the proposed BC for the 6 clusters under consideration.

Table 3. Balance Criteria for supercomputers under consideration.

No	Cluster	TPP	RAM	NP	C1	C2	BC
1	MVS-100K	1,24	6.02	4.7	0.2	0.26	**20**
2	SKIF MSU	2,56	12.47	11.4	0.2	0.2	**20**
3	SKIF Cyberia	0,6	1.45	8.17	0.4	0.4	**40**
4	NSU cluster	0.92	4.47	6.35	0.2	0.14	**14**
5	Polytechnic	409	471.2	26.81	0.86	0.06	**6**
6	NKS-1P	431.5	497	2.64	0.87	0.006	**0.6**

The cluster systems under consideration have been tested against the PIC benchmark, described in the previous sections. Remind that PIC is characterized by quite random memory access, inefficient use of the cache and by very small efficiency of compiler-based optimizations like loop unrolling, constant propagation, etc. Comparative performance results are given in Figure 6 below.

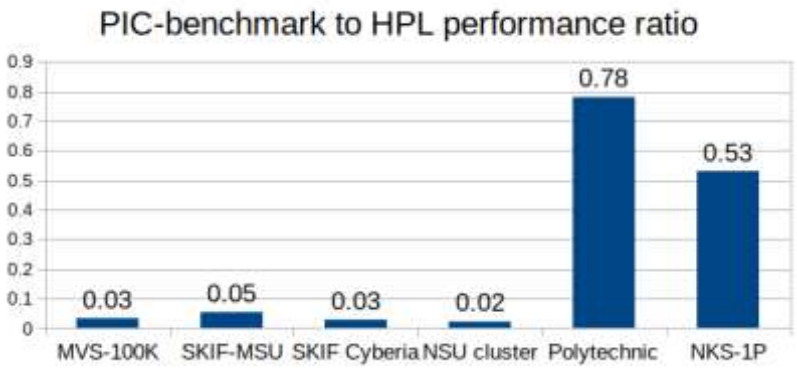

Fig. 6. Comparison of the performance evaluated by the PIC-benchmark to HPL performance.

Now let us compare the values of TPP, relative performance at PIC benchmark, and the obtained values of BC, proposed in the paper. Results are given in Table 4.

Table 4. Influence of BC on PIC benchmark against TPPs.

No	Cluster	TPP	PIC test	BC	TPP/PIC	10^3 * PIC/TPP
	2	3	4	5	6	7
1	MVS-100K	1.24	0.03	20	41,3	**24.2**
2	SKIF MSU	2.56	0.05	20	51,2	**19.5**
3	SKIF Cyberia	0.6	0.03	40	20	**50**
4	NSU cluster	0.92	0.02	14	46	**21.7**
5	Polytechnic	409	0.78	6	524	**1.9**

Table 4 shows the influence of the value of the proposed balance criterion BC (1) of the cluster system on the ratio of its performance on the PIC benchmark to the total processors performance TTP. The column 7 of Table 4 presents the values of this relationship, for convenience multiplied by 1000. It can be seen that the lower the value of the BC criterion, the smaller is the specified relationship. The MVS-100K and SKIF MSU clusters have the same values of the BC, so, almost coincide are the ratio values in the column 7. The SKIF Cyberia system has a two times as great value of BC and, accordingly, doubled the value of the ratio in the column 7. Figures on the NSU Cluster coincidence is not so good, but comparison to SKIF Cyberia makes clearly visible the influence of the BC to the values in the column 7. This trend is perfectly transferred to more modern Polytechnic and NKS-1P systems. Thus, the values of the BC criterion in these clusters are significantly lower than that of clusters in lines 1-4. Accordingly, the performance ratio on the PIC test to the overall performance of processors (column 7 in Table 4) is significantly less. The BC at the NKS-1P system is less than that of Polytechnic, which is reflected in the corresponding digits of column 7.

Thus, the better balance of the system according to the proposed criterion BC indicates more good performance results (relative to the total performance of computational nodes TTP) when solving practical problems compared to the worse balanced system, albeit with a greater total productivity TTP.

5 Conclusion

Since the peak performance of cluster computing systems is usually very far from actually observed in the numerical solution of the real problems of mathematical physics, a PIC test

is proposed for obtaining such a "real" performance evaluation. It is characterized by a non-deterministic high load on RAM, which allows you to accurately measure the real performance of the memory system, as well as a large number of interprocessor communications, which makes the system performance dependent on the parameters of the interconnect. The known number of operations per model particle is used to calculate computing performance as a separate processor and the system as a whole.

Depending on the features of the problem being solved, the bottleneck for performance may be both the total processors performance TPP and the speed of access to RAM or the performance of the interconnect. A simple criterion BC (1) of the system balance is proposed, measured in percent. It shows how much the speed of memory access and network performance differ from the TPP. The examples in Table 4 show that the better the system is balanced (the value of the proposed criterion BC above), the smaller the performance on the PIC test differs from the TPP. As a consequence, it is possible to formulate a recommendation that if the designed supercomputing system is focused on solving a wide class of problems of mathematical physics, the maximum possible value of the BC criterion should be achieved.

References

1. R. van de Geijn, *Massively Parallel LINPACK Benchmark on the Intel Touchstone DELTA and iPSC/860 Systems* (1991 Annual Users Conference Proceedings. Intel Supercomputer Users Group, Dallas, TX, 1991)
2. J. Dongarra, M. Heroux, *Toward a New Metric for Ranking High Performance Computing Systems* "(Sandia National Laboratory, 2013)
3. D. Baily, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, S. Weeratunga, *The NAS Parallel Benchmarks* (NAS Technical Report RNR-94-007, NASA Ames Research Center, Moffett Field, CA, 1994)
4. Yu. Grigoriev, V. Vshivkov, M. Fedoruk, *Numerical «particle-in-cell» methods: theory and applications* (De Gruyter, 2002)
5. V. L. Peterson, Grand Chall. to Comp. Sci. **5**, 2-3 (1989)
6. F. Stern, Z. Wang, J. Yang, H. Sadat-Hosseini, M. Mousaviraad, S. Bhushan, M. Diez, S.-H. Yoon, P. Wu, S. M. Yeon, T. Dogan, D.-H. Kim, S. Volpi, M. Conger, T. MichaelI, T. Xing, R. S. Thodal, J. L. Grenestedt, J. of Hydrodynamics, B. **27**, 1 (2015)
7. D. Kotov, J. Comp. Phys. **307** (2016)
8. P. G. Tucker, Num. Fluid (2014)
9. A. Safi, S. Turek GPGPU-based rising bubble simulations using a MRT lattice Boltzmann method coupled with level set interface capturing (Tech. Rep. 500, Inst. Appl. Math., TU Dortmund, 2014)
10. E. Yilmaz, S. Aliabadi, Computers & Fluids **80** (2013)
11. A. Siegel, K. Smith, P. Fischer, V. Mahadevan, J. Comp. Phys. **231**, 8 (2012)
12. E. Bauge et al., Nuclear Data Sheets **118**, 32—37 (2014)
13. V. K. Decyk, T. V. Singh, Comp. Phys. Comm. **185**, 3 (2014)
14. T. Minoshima, J. Comp. Phys. **236** (2013)
15. J. A. Acebrón, J. Comp. Phys. **250** (2013)
16. H. Markram et al., Procedia Computer Science **7** (2011)
17. K. Yee, IEEE Trans. Antennas Prop. **14,** 3 (1966)
18. H.Joshi et al. Progress in Biophys. Mol. Biol. **107**, 1 (2011)

19. A. Langdon, B. Lasinski, Methods in Comp. Phys. **16**, 327 (1976)
20. J. Dongarra, M. Heroux, *Toward a New Metric for Ranking High Performance Computing Systems* (Sandia National Laboratory, 2013)