ISSN 1547-4771, Physics of Particles and Nuclei Letters, 2017, Vol. 14, No. 7, pp. 985–992. © Pleiades Publishing, Ltd., 2017.

# Desktop Supercomputer: What Can It Do?[1]

**A. Bogdanov*, A. Degtyarev, and V. Korkhov**

*St. Petersburg State University, St. Petersburg, 199034 Russia*

**e-mail: a.v.bogdanov@spbu.ru*

Received December 9, 2016

**Abstract**—The paper addresses the issues of solving complex problems that require using supercomputers or multiprocessor clusters available for most researchers nowadays. Efficient distribution of high performance computing resources according to actual application needs has been a major research topic since high-performance computing (HPC) technologies became widely introduced. At the same time, comfortable and transparent access to these resources was a key user requirement. In this paper we discuss approaches to build a virtual private supercomputer available at user's desktop: a virtual computing environment tailored specifically for a target user with a particular target application. We describe and evaluate possibilities to create the virtual supercomputer based on light-weight virtualization technologies, and analyze the efficiency of our approach compared to traditional methods of HPC resource management.

**DOI:** 10.1134/S1547477117070032

## 1. INTRODUCTION

The notion of supercomputer as a computing facility became common after the list of so called 'grand challenges' has been published. This happened in the end of the 1980s in order to mark the necessity to fund the research in the area of high-performance computing and communications. As known, these years became critical for IT as first personal computers, global networks and WWW appeared and began conquering the market. The question about viability of development of computer technology in the same trend as in previous 30 years was posed. A set of fundamental and applied problems vital for humanity that could be solved efficiently only with the help of superpowerful computational resources (in contrast with personal computers and workstations) was formed.

Since then the list changed many times, and the notion of 'grand challenges' became common. The list contains complex problems of physics, nanotechnology, aeronautics, biology, national security, Earth sciences, power engineering, environment etc.

If we try to understand what computational means are required to solve the grand challenge problems, then we can come to the general conclusion about three conceptual configurations that target to solve [7]:

(1) Problems of processing large and very large volumes of data;

(2) Problems with large numbers of loosely coupled tasks;

(3) One tightly coupled tasks requiring large memory amounts and high performance computational resources.

In any case, especially in case 3, such tasks can be considered as complex, i.e. (a) processes of substantially different scale of variability are present in the considered task; (b) the nature of the processes is essentially nonlinear.

Calculation of complex problems, particularly non-linear ones, has always been a challenge for computer science. This challenge has always led to some remarkable ups as well as very disappointing downs. In our opinion, the downs are closely related to the degradation of computer system architectures equipped with multi-level data stream communication systems that are way too complex to remain effective. Currently, the appearance of multithreaded and graphic processors has led to similar problems.

We believe that the solution to these problems should be complex and consist of: (a) Preparation of specialized computer systems; (b) Development of algorithms that take into account the specifics of modern computer systems; (c) Use of more effective management of data flows in computing. To understand the emerging issues, consider one of the typical approaches to complex numerical problems. As an example, one can consider:

—The finite difference method.

—The solution of complex problems by iterations.

—Application of expansions (representation of the result via a combination of known solutions),etc.

[1] The article is published in the original.

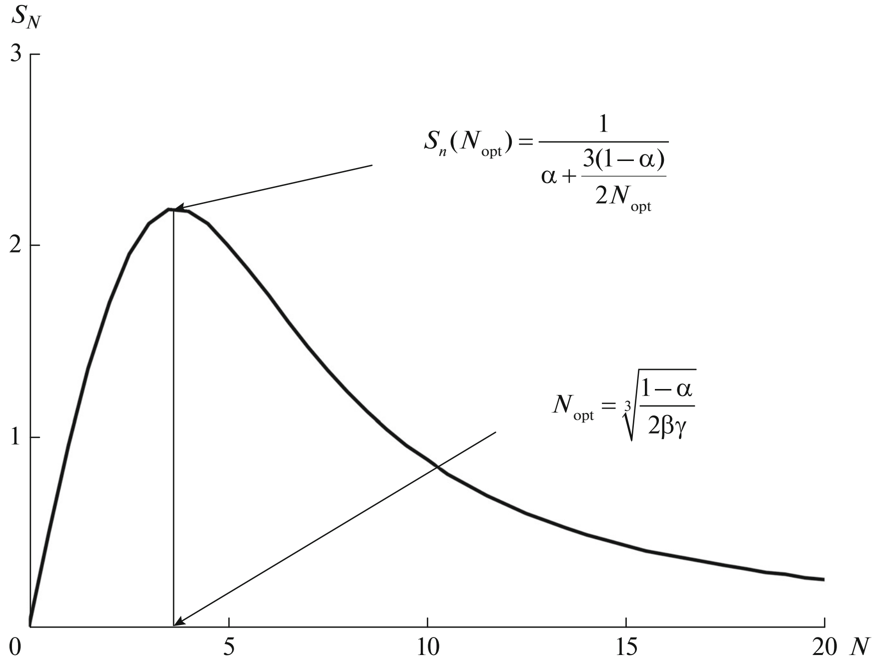

**Fig. 1.** Speedup of a cluster system.

All of the methods above lead to large dimension matrix systems (up to millions of rows and columns). In a parallel calculation of flows in the matrix, with big difference of scale of the processes, unnecessary computational burden occurs when calculating a slow process. If the scale relationship does not depend on the problem parameters, it is possible to simplify the task greatly by using asymptotic methods. However, in most cases it is not the case. Unfortunately, the most interesting effects arise where the characteristic scales of the described processes pass from small to large values.

From the point of view of dynamical systems theory [1], the appearance of large off-diagonal terms of the matrix is due to the wrong choice of the basis. However, choosing the right basis, especially for nonlinear systems is a very complex task.

There is an elegant approach, associated with the use of non-linear integral transformations [20], which for many cases allow appropriate diagonalization of a matrix. The central problem with this approach is that for such transformations the inverse transformations typically do not exist. In those cases where they can be found we obtain the known results. As a working hypothesis we can assume that the inverse transformation exists only for completely integrable systems [3].

Moreover, it is very important to study the cases where the desired result is obtained without applying the inverse transformation (for example, when we consider the mean values of any variables or integral characteristics that are directly used for other tasks). However, this idea can be applied as follows. As the core of decomposition one can use a completely integrable system, and a result of its quasidiagonalization can be substituted in the residue.

This general idea requires several steps for its implementation, including the use of peculiarities of modern computing devices. This article is devoted to analysis of this issue.

## 2. THE PROBLEMS OF MODERN COMPUTING PLATFORMS

Most modern calculations are performed on cluster systems, i.e. on systems with high-performance processors, connected by relatively slow routers and using message passing as a primary form of interprocess communication. In this case, even in the peer system, this scheme leads to long delays in the data transmission and problems with scalability.

Acceleration of a cluster system may asymptotically be represented as follows [11, 13]:

$$S_N = \frac{T_0}{T_N} = \frac{N}{1-\alpha+\alpha N+\beta\gamma N^3}, \quad (1)$$

where $N$ denotes the number of nodes, $\alpha$ is the coefficient representing the fraction of sequential calculations; $\beta$ is the coefficient presenting the architecture (diameter) of the system; $\gamma$ is the coefficient equal to the ratio of the rate of a node (processor) to that of a link (link speed).

The dependency (1) is shown in Fig. 1.

A distinctive feature of this dependency is the presence of optimal number of processors for a particular computing problem, which depends mainly on $\gamma$. Considering the overhead increasing with the number of processors, the speedup is not growing but falling down after reaching this optimal number of processors.

In practice, this leads to the consequence that for problems with large data transfer at each step (for example, engineering applications) the scalability is limited to a dozen or two dozen compute nodes. In reality, the situation is even worse. In modern processors, there is a multi-tiered storage system with a large number of cores and internal communication between them. The user often cannot control individual cores and implement load balancing between them. It also leads to a more complicated formula than (1) and a sharp decrease in scalability [20]. Therefore, the Linux-clusters cannot be general-purpose systems.

There exist, however, some very indicative examples where Linux-based clusters presented very effective specialized systems. This is due to the adaptation of the cluster architecture for computational algorithm and optimization of computing processes at all stages. Unfortunately, such adaptation/restructuring cannot be performed every time a new problem arises. Modern computing systems are very difficult to restructure and it is so expensive that in practical applications cannot be used. This is due to the famous problem of mapping of the algorithm onto computing architecture. Each computational algorithm has its own characteristics associated with the nature of exchanges between computing processes. Preparing a computing architecture that would be effective enough for every occasion is impossible.

The history of computing, however, shows that all the shortcomings of computer systems at all times are compensated by effective software. We believe that at the present stage with the help of technologies of metacomputing, service-oriented architecture and virtualization it is possible to obtain such a solution. The main idea of our approach is the virtualization of all components of a computer system to create a virtual computer system, which is capable of displaying at best computational algorithm. The use of a shared filesystem and virtual shared memory is the basis of the implementation of the paradigm of shared memory programming. This allows overcoming the problem of scalability of computing systems.

## 3. PRINCIPLES OF A VIRTUAL SUPERCOMPUTER

The main principles of implementation of a virtual supercomputer are based on the following assumptions [4]:

(1) Cloud is determined completely by its Application Programming Interface (API). It is obvious from the user point of view, but the same is true from the point of view of different clouds interaction.

(2) Operational environment must be UNIX-like. One of the main problems of Computational Grids is load balancing and it is a very difficult task since the user is cut off from the resources. Partly this problem is solved by Problem Solving Environments (PSE). However, in order to make it really active, many standard UNIX tools must be introduced into the API.

(3) Cloud uses protocols, compatible with popular public clouds. Public clouds are not very useful for complex problems and the reason for this is clear – the more difficult problem you are solving, the more robust tools you must use. Universal tools cannot be used for complex problems. That is why specialized private clouds must be built for complex problems, but if their resources are not enough, some additional resources can be provided from a public cloud.

(4) Cloud processes the data on the base of distributed file systems. The main problem with the public cloud for data processing comes from the fact, that its own file system is used on each computer in the cloud. That prevents both processing large data sets and scaling out the problem solution. To overcome this, a distributed file system should be used in the private cloud, the type of which is determined by the nature of the problem to solve. If we add here three ways of data consistency providing (Brewer's theorem) [10], we can see that there are a lot of possibilities of data processing organization out of which only a few are in use.

(5) The consolidation of data is achieved by distributed Federal Data Base (Federal DB). There are three levels of consolidation – servers, data and resources. It is more or less clear how server consolidation is done. Consolidation of data is more difficult, and consolidation of resources is a real challenge to a cloud provider. We assume that the most natural way to do this is to use Federal DB tools. Up to now we managed to do this by utilizing IBM's DB2 tools, but we believe that possibilities of latest PostgreSQL release will make it possible to work out a freeware tool for such purpose.

(6) Load balancing is achieved by the use of virtual processors with controlled rate. New high throughput processors make it possible to organize virtual processors with different speed of computation. This opens the natural possibility of setting up a distributed virtual computational system with the architecture adapted to computational algorithm, and instead of mapping the algorithm onto the computer architecture we will match the architecture to the computational code.

(7) Processing of large data sets is done via shared virtual memory. Actually all previous experience shows that the only way to comfortably process large data sets is to use the Symmetric Multiprocessing (SMP) system. Now we can effectively use shared memory tools (e.g. OpenCL) in heterogeneous environment and so make virtual SMP. The same tool is used for parallelization. The possibilities of a single system image (SSI) operational environment are also very effective.

(8) Cloud uses complex grid-like security mechanisms. One of the cloud problems is security issues,

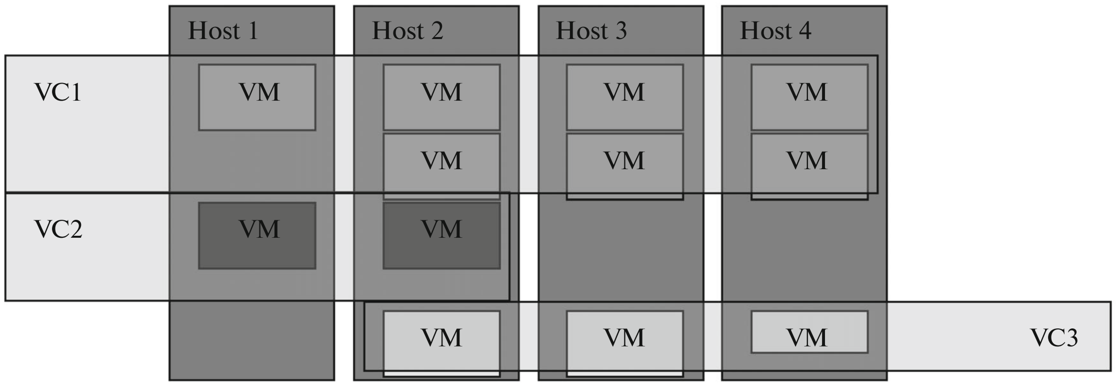

**Fig. 2.** A testbed example with a set of virtual clusters deployed over a set of physical resources.

but we feel that the proper combination of Grid security tools with Cloud access technologies is possible.

## 4. VIRTUALIZATION FOR BUILDING VIRTUAL CLUSTERS

Virtualization refers to the act of creating a virtual version of an object, including but not limited to a virtual computer hardware platform, operating system (OS), storage device, or computer network resources. It can be divided onto several types, and each one has its own pros and cons. Generally, hardware virtualization refers to abstraction of functionalities from physical devices. Nowadays, on modern multicore systems with powerful hardware it is possible to run several virtual guest operating systems on a single physical node. In a usual computer system, a single operating system uses all available hardware resources (CPU, RAM, etc), whilst virtualized system can use a special layer that spreads low-level resources to several systems or applications; this layer looks like a real machine for launched applications. Virtualization technologies facilitate creation of a virtual supercomputer or virtual clusters that are adapted to problem being solved and help to manage processes running on these clusters (see Fig. 1). The work described in this paper continues and summarises our earlier research presented in [5, 13–18].

In our experience, the main benefit of virtualization for high-performance computing is structural decomposition of a distributed system into unified entities—virtual machines or application containers—which simplifies maintenance of the system. A new entity can be created for each new version of the application with optimal configuration and set of libraries, so that multiple versions of the same software may coexist and run on the same physical cluster. Entities can be copied or efficiently shared between different physical machines to create private cluster for each application run. Virtualization can sometimes even provide increase in application performance, however, it is not easy achievable. Allocating a separate container for each application allows compiling it for hybrid GPGPU systems which may or may not improve performance of an application. However, such optimisations are possible even without application containers. Full virtualization gives an option of choosing the right operating system for an application, but gives constant decrease in performance due to overheads, which is not tolerable for large-scale parallel applications.

Thus, for high-performance computing virtualization is a tool that helps manage parallel and distributed applications running on physical cluster. It allows different versions of the same libraries and operating systems to co-exist and to be used as environments for running applications that depend on them.

Earlier we evaluated the capabilities given by different approaches and virtualization technologies to build a computational environment with configurable computation (CPU, memory) and network (latency, band-width) characteristics, which we call Virtual Private Supercomputer (VPS) [15]. Such configuration enables flexible partitioning of available physical resources between a number of concurrent applications utilizing a single infrastructure. Depending on application requirements and priorities of execution each application can get a customized virtual environment with as much resources as it needs or is allowed to use.

Experiments show that using lightweight virtualization technologies (para-virtualization and application containers) instead of full virtualization is advantageous in terms of performance [2], hence virtual computing nodes should be created using lightweight virtualization technologies only. Thus, lightweight virtualization is inevitable in achieving balance between good performance and ease of system administration in distributed environment and as a consequence operating system should be UNIX-like for it to work. Load balance can be achieved using virtual processors with controlled clock rate and process migration. However, not every operating system supports these

**Table 1.** Hardware and software components of the prototype system

| Component | Details | Component | Details |
|---|---|---|---|
| CPU model | Intel Xeon E5440 | Operating system | CentOS 7 |
| CPU clock rate, GHz | 2.83 | Kernel version | 3.10 |
| Number of cores/CPU | 4 | LXC version | 1.0.5 |
| Number of CPUs/node | 2 | GlusterFS version | 3.5.1 |
| RAM (GB) | 4 | TORQUE version | 5.0.0 |
| Disk model | ST3250310NS | OpenMPI version | 1.6.4 |
| Disk speed, rpm | 7200 | IMB version | 4.0 |
| Number of nodes | 12 | OpenFOAM version | 2.3.0 |
| Interconnect speed, Gbps | 1 | | |

technologies and it should be possible to access virtual super-computer facilities through fully-virtualized hosts.

## 5. IMPLEMENTATION OF THE TESTBED

Only lightweight virtualization technologies can be used to build efficient virtual clusters for large-scale problems [15, 16, 18]. This stems from the fact that on the large scale no service overhead is acceptable if it scales with the number of nodes. In the case of the virtual clusters, a scalable overhead comes from processor virtualization which means that neither para- nor fully-virtualized machines are suitable for large virtual clusters. This leaves only application container technologies for investigation. The other challenge is to make dynamic creation and deletion of virtual clusters in real time.

The test system comprises many standard components which are common in high performance computing: distributed parallel file system which stores home directories with experiment's input and output data; cluster resource scheduler which allocates resources for jobs and client programs to pre- and post-process data; the non-standard component is network-attached storage exporting container's root files systems as directories. Linux Container technology (LXC) is used to provide containerisation, GlusterFS is used to provide parallel file system and TORQUE to provide task scheduling. The most recent CentOS Linux 7 is chosen to provide stable version of LXC (>1.0) and version of kernel which supports all containers' features. Due to the limited number of nodes each of them is chosen to be both compute and storage node and every file in parallel file system is stored on exactly two nodes.

To summarise, only standard Linux tools are used to build the system: there are no opaque virtual machines images, no sophisticated full virtualization appliances and no heavy-weight cloud computing stacks in this configuration.

## 6. EVALUATION OF THE TESTBED

To test the resulting configuration OpenMPI and Intel MPI Benchmarks (IMB) were used to measure network throughput and OpenFOAM was used to measure overall performance on a real-world application.

The first experiment was to create virtual cluster, launch an empty (with /bin/true as an executable file) MPI program in it and compare execution time to ordinary physical cluster. To set this experiment up in the container the same operating system and version of OpenMPI as in the host machine was installed. No network virtualization was used, each run was repeated several times and the average was displayed on the graph (Fig. 3). The results show that a constant overhead of 1.5 second is added to every LXC run after the 8th core: one second is attributed to the absence of cache inside container with SSH configuration files, key files and libraries in it and other half of the second is attributed to the creation of containers as shown in Fig. 4. The jump after the 8th core marks bounds of a single machine which means using network for communication rather than shared memory. The creation of containers is fully parallel task and takes approximately the same time to complete for different number of nodes. Overhead of destroying containers was found to be negligible and was combined with mpirun time. So, usage of Linux containers adds some constant over-

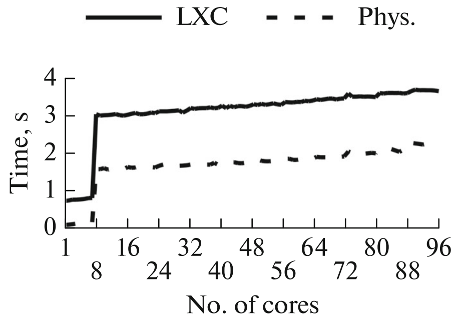

**Fig. 3.** Comparison of LXC and physical cluster performance running empty MPI program.

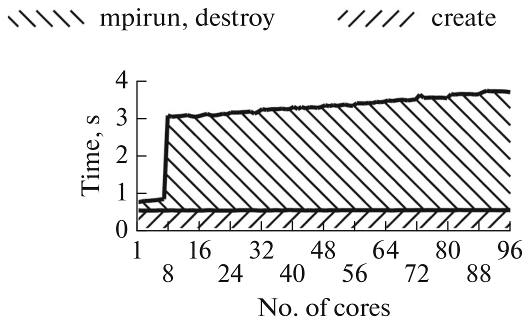

**Fig. 4.** Breakdown of LXC empty MPI program run.

head to the launching of parallel task depending on system's configuration which is split between creation of containers and filling the file cache.

Another experiment dealt with real-world application performance and for this role the OpenFOAM was chosen as the complex parallel task involving large amount of network communication, disk I/O and high CPU load. The dam break RAS case was run with different number of cores (total number of cores is the square of number of cores per node) and different LXC network types and the average of multiple runs was displayed on the graph (Fig. 5). Measurements for 4 and 9 cores were discarded because there is a considerable variation of execution time for these numbers on physical machines. From the graph it can be seen that low performance of virtual ethernet decreased final performance of OpenFOAM by approximately 5–10% whereas *macvlan* and *none* performance is close to the performance of physical cluster (Fig. 6). Thus, the choice of network type is the main factor affecting performance of parallel applications running on virtual clusters and its overhead can be eliminated by using *macvlan* network type or by not using network virtualization at all. More experimental results are presented in [14].

To summarise, there are two main types of overheads when using virtual cluster: creation overhead which is constant and small compared to average time of a typical parallel job and network overhead which can be eliminated by not using network virtualization at all.

## 7. MANAGEMENT OF CONTAINER-BASED VIRTUAL INFRASTRUCTURE

The next step in using containers for building virtual cluster is applying various automation and management tools that help to ease deployment and handling of virtual clusters. We investigated capabilities provided by several modern tools (Docker, Mesos, Mininet) to model and build virtualised computational infrastructure; investigated configuration management in the integrated environment and evaluated performance of the infrastructure tuned to a particular test application. Docker—a lightweight and powerful open source container virtualization technology which we use to manage containers—has a resource management system available so it is possible to test different configurations: from "slow network and slow CPUs" to "fast network and fast CPUs."

Figure 7 shows the experimental results for execution of NAS Parallel Benchmarks (NPB) suite on different configurations of virtual testbed (Class W: workstation size; Classes A, B: standard test problems, 4X size increase going from one class to the next). With NPB results are also very different, everything depends on benchmark type. For example, for SP test smaller size system of nonlinear PDEs had better Mop/s than for bigger size. However, for lower matrices sizes in LU test results are worse than for bigger matrices.

The presented approach for virtual clusters creation from Linux containers was found to be efficient and its performance was proven comparable to that of an ordinary physical cluster: not only the use of containers does not incur processor virtualization overheads, but also the network virtualization overheads can be totally removed if the host's network name space is used and the network bandwidth is saved by automatically transferring only those files that are needed through network-mounted file system rather

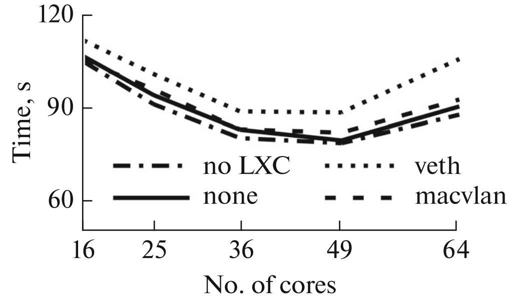

**Fig. 5.** Average performance of OpenFOAM with different LXC network types.

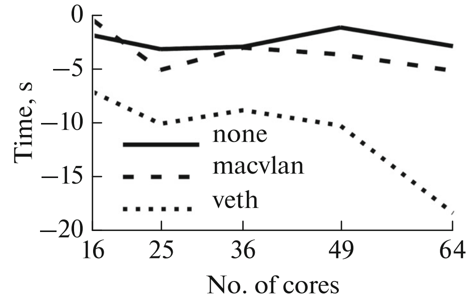

**Fig. 6.** Difference of OpenFOAM performance on physical and virtual clusters. Negative numbers show slowdown of virtual cluster.

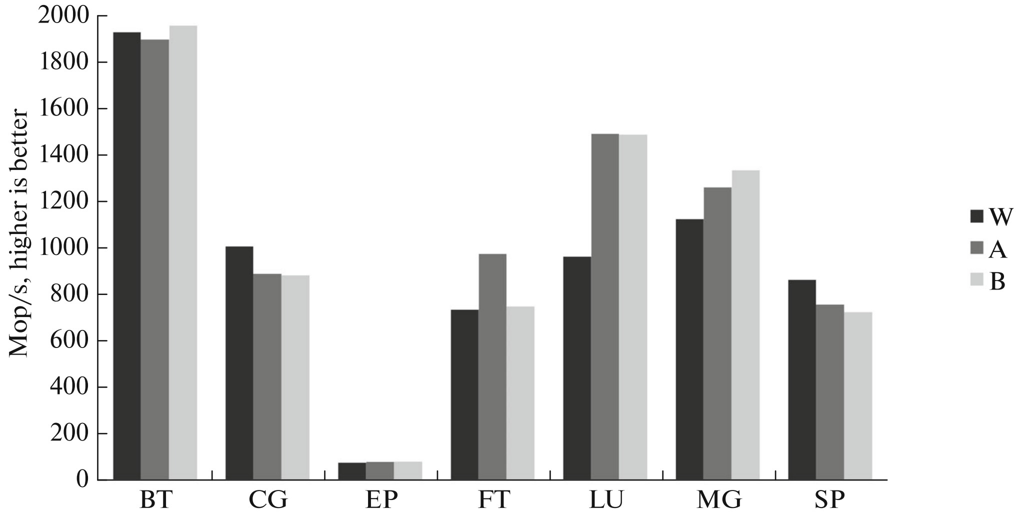

**Fig. 7.** Performance of different tests from NAS Parallel Benchmarks suite on different configurations.

than the whole images. From the point of view of a system administrator, storing each HPC application in its own container makes versioning and dependencies control easily manageable and their configuration does not interfere with the configuration of the host machines and other containers. In general, the testing above demonstrated sufficient efficiency of the virtual cluster, and in the future we can move on to the implementation of the program, described in the introduction.

## 8. EXAMPLES OF PROBLEMS SOLVED WITH THE PROPOSED APPROACH

Herein we only mention some of the important problems that are solved in this way:

(1) Numerical investigation of the classical trajectory problem.

The quantum approach is constructed on the generating trajectory tubes which allow taking into account the influence of classical nonintegrability of the dynamical quantum system [8]. When the volume of classical chaos in a phase space is larger than the quantum cell in the corresponding quantum system, quantum chaos is generated. Success in numerical investigation in this case was reached just only owing to the proposed approach.

(2) Direct numerical simulation in fluid dynamics.

Direct numerical experiments in continuum mechanics using digital discrete computers are based on a limited set of numeric objects which interpolate parameters of the state of the physical fields in time [12]. The proposed approach makes it possible to exclude from consideration the mathematical models of fluid mechanics in the form of differential equations in partial derivatives. Computational experiment is carried out on the basis of fundamental conservation laws. The dualism of corpuscular and continual models of continuous medium allowed to present computing procedure in the form of three serial stages combining approaches of Euler and Lagrange. Such division is aimed at providing efficient computing procedure especially in the conditions of the multiprocessor computer environment [7]. As the basis of computational efficiency the use of explicit numerical schemes can be considered.

(3) Virtual testbed.

Under the virtual testbed we mean hardware-software system that provides comprehensive modeling of dynamic objects interacting with each other and with environment. Since the use of a universal model in this case is not possible, harmonization of certain models of objects and their mapping to distributed computer architecture is necessary [6, 17].

(4) Physical and chemical processes in PECVD reactors.

A realization of virtual testbed principles is demonstrated in such an important application as the Virtual Reactor for Plasma Enhanced Chemical Vapor Deposition (PECVD) [19].

(5) Modeling political behavior psychology.

Traditionally social science research methods are focused mainly on the use of statistical methods and game theory. The use of such instruments is due mainly to complex character and difficult formalization of the problems. In [9] an attempt to extend the described approach to the solution of this problem is carried out.

## 9. SUMMARY

The analysis showed that the new computing resources can solve complex nonlinear problems. However, this requires great efforts. The most difficult part is to prepare a virtual distributed computing system. Unfortunately, this problem cannot be formalized. It requires highly skilled both system programmers and users. With regard to the development of algorithms, with the appearance of virtual clusters, this problem is likely to be simpler in view of the mathematical formulation of problems. It seems to us that we should expect significant progress in both these areas.

## ACKNOWLEDGMENTS

The research was carried out using computational resources of Resource Centre "Computational Centre of Saint Petersburg State University" (T-EDGE96 HPC-0011828-001) and supported by Russian Foundation for Basic Research (projects nos. 16-07-01111, 16-07-00886, 16-07-01113) and St. Petersburg State University (project no. 0.37.155.2014).

## REFERENCES

1. V. I. Arnold, *Ordinary Differential Equations* (Springer, Berlin, Heidelberg, 1992).
2. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," ACM SIGOPS Operat. Syst. Rev. **37**, 164–177 (2003).
3. A. V. Bogdanov, "Calculation of the quantum-mechanical inelastic scattering amplitude via solution of the classical dynamical problems," Sov. Phys. Tech. Phys. **31**, 833–835 (1986).
4. A. Bogdanov, "Private cloud vs personal supercomputer," in *Proceedings of the 5th International Conference on Distributed Computing and Grid Technologies in Science and Education* (JINR, Dubna, 2012), pp. 57–59.
5. A. V. Bogdanov, A. B. Degtyarev, I. G. Gankevich, V. Yu. Gayduchok, and V. I. Zolotarev, "Virtual workspace as a basis of supercomputer center," in *Proceedings of the 5th International Conference on Distributed Computing and Grid Technologies in Science and Education* (JINR, Dubna, 2012), pp. 60–66.
6. A. Bogdanov, A. Degtyarev, and Yu. Nechaev, "Parallel algorithms for virtual testbed," in *Proceedings of the 5th International Conference on Computer Science and Information Technologies CSIT'2005, Yerevan, Armenia, 2005,* pp. 393–398.
7. A. V. Bogdanov, A. B. Degtyarev, and V. N. Khramushin, "High performance computations on hybrid systems: will "grand challenges" be solved?," Comput. Res. Model. **7**, 429–438 (2015).
8. A. Bogdanov, A. Gevorkyan, and G. Nyman, "Regular and chaotic quantum dynamics in atom-diatom reactive collisions," Phys. At. Nucl. **71**, 876–883 (2008).
9. A. Bogdanov et al., "Mathematical model of psychology-political classification of political parties," in *Proceedings of the International Conference on Mathematical Modeling and Computational Physics, Stará Lesná, Slovakia, 2015.*
10. E. A. Brewer, "A certain freedom: thoughts on the CAP theorem," in *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing* (ACM, New York, 2010), No. 1, pp. 335–336.
11. A. Degtyarev, "High performance computer technologies in shipbuilding," in *Optimistic. Optimization in Marine Design,* Ed. by L. Birk and S. Harries (Mensch & Buch, Berlin, 2003), pp. 245–258.
12. A. B. Degtyarev and V. N. Khramushin, "Design and construction of computer experiment in hydrodynamics using explicit numerical schemes and tensor mathematics algorithms," Mat. Model. **26** (11), 4–17 (2014).
13. I. Gankevich, Yu. Tipikin, A. Degtyarev, and V. Korkhov, "Novel approaches for distributing workload on commodity computer systems," Lect. Notes Comput. Sci. **9158**, 259–271 (2015).
14. I. Gankevich, S. Balyan, S. Abrahamyan, and V. Korkhov, "Applications of on-demand virtual clusters to high performance computing," Comput. Res. Model. **7**, 504–509 (2015).
15. I. Gankevich, V. Gaiduchok, D. Gushchanskiy, Y. Tipikin, V. Korkhov, A. Degtyarev, A. Bogdanov, anbd V. Zolotarev, "Virtual private supercomputer: design and evaluation," in *Proceedings of the 9th International Conference on Computer Science and Information Technologies CSIT (2013).* doi 10.1109/CSITechnol.2013.671035810.1109/CSITechnol.2013.6710358
16. I. Gankevich, V. Korkhov, S. Balyan, V. Gaiduchok, D. Gushchanskiy, Y. Tipikin, A. Degtyarev, and A. Bogdanov, "Constructing virtual private supercomputer using virtualization and cloud technologies," in *Proceedings of the International Conference on Computational Science and Its Applications ICCSA'2014,* Lect. Notes Comput. Sci. **8584**, 341–354 (2014).
17. I. Gankevich and A. Degtyarev, "Model of distributed computations in virtual testbed," in *Proceedings of the 9th International Conference on Computer Science and Information Tecnologies CSIT'2013, Yerevan, Armenia, 2013,* pp. 240–244.
18. V. Korkhov, S. Kobyshev, and A. Krosheninnikov, "Flexible configuration of application-centric virtualized computing infrastructure," in *Proceedings of the International Conference on Computational Science and Its Applications ICCSA'2015,* Lect. Notes Comput. Sci. **9158**, 342–353 (2015).
19. V. Korkhov, V. Krzhizhanovskaya, and P. Sloot, "A grid based virtual reactor: parallel performance and adaptive load balancing," J. Parallel Distrib. Comput. **68**, 596–608 (2008).
20. I. Shoshmina and A. Bogdanov, "Using GRID technologies for computations," Vestn. SPb. Univ., Fiz. Khim., No. 3, 130–137 (2007).
21. M. Yu. Sumetsky and M. L. Fel'shtyn, *Resonant Tunneling of Electrons through the Two- and Three-Dimensional Nanostructures* (NTO Akad. Nauk SSSR, Leningrad, 1990).