EPJ Web of Conferences **108**, 01002 (2016)
DOI: 10.1051/epjconf/201610801002
© Owned by the authors, published by EDP Sciences, 2016

New Approach to the Simulation of Complex Systems

Alexander Bogdanov[1,a], Alexander Degtyarev[1,b], and Vladimir Korkhov[1,c]

[1] *St.Petersburg State University, St.Petersburg, 7-9, Universitetskaya nab, 199034, Russia*

Abstract. The paper analyzes the problems of scalability of modern computational systems, and offers a new paradigm for solving complex problems on them. It implies (1) Creating a virtual computing cluster with shared virtual memory, (2) Selecting a representation for the problem that minimizes the interaction between computing threads and (3) Configuring the virtual computer system for optimal mapping of the pertinent algorithm on it. Arguments for optimizing virtual clusters are given and test results on them are shown. We discuss the challenges that can be addressed most effectively within the framework of the proposed approach.

1 Introduction

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.

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

[a] e-mail: a.v.bogdanov@spbu.ru
[b] e-mail: a.degtyarev@spbu.ru
[c] e-mail: v.korkhov@spbu.ru

This is an Open Access article distributed under the terms of the Creative Commons Attribution License 4.0, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 non-linear systems is a very complex task.

There is an elegant approach, associated with the use of non-linear integral transformations [18], 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.

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.

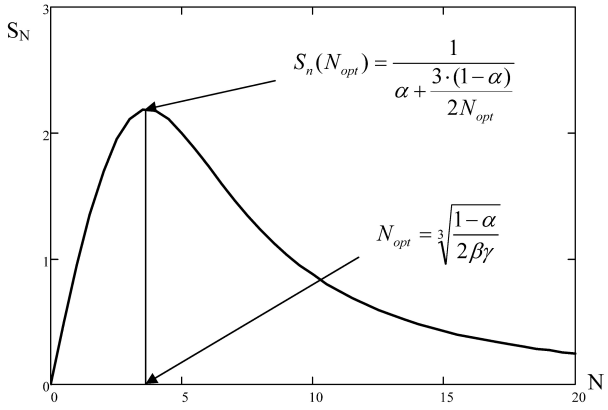

Figure 1. Speed up for Linux clusters

Acceleration of a cluster system may asymptotically be represented as follows [14]:

$$S_N = \frac{T_0}{T_N} = \frac{N}{1 - \alpha + \alpha N + \beta\gamma N^3}. \tag{1}$$

Here N denotes the number of nodes;

α is the coefficient representing the fraction of sequential calculations;

β is the coefficient presenting the architecture (diameter) of the system;

γ is the coefficient equal to the ratio of the rate of a node (processor) to that of a link (link speed).

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 can not 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 [17]. Therefore, the Linux-clusters can not 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 can not 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 can not 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 technology of

(a) metacomputing;

(b) service-oriented architecture and

(c) 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 virtual shared memory is the basis of the implementation of the paradigm of shared memory programming. This allows to overcome the problem of scalability of a computing system.

3 Implementation of a virtual supercomputer

The main principles of implementation of a virtual supercomputer are based on the following assumptions [5].

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. The 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) [9], 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, but we feel that the proper combination of Grid security tools with Cloud access technologies is possible.

4 Testing a virtual distributed computing system

Only lightweight virtualization technologies can be used to build efficient virtual clusters for large-scale problems [11, 12, 16]. 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.

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. Detailed hardware characteristics and software version numbers are listed in Table1.

Creating a virtual cluster in such an environment requires the following steps. First, a client submits a task requesting a particular number of cores. Then, according to the distribution of these cores among compute nodes, a container is started on each node from the list with Secure Shell (SSH) daemon as the only program running inside it. Here are two options: either start containers with

Table 1. Hardware and software components of the system

Component	Details	Component	Details
CPU model	Intel Xeon E5440	Operating system	CentOS 7
CPU clock rate (GHz)	2.83	Kernel version	3.10
No. of cores per CPU	4	LXC version	1.0.5
No. of CPUs per node	2	GlusterFS version	3.5.1
RAM size (GB)	4	TORQUE version	5.0.0
Disk model	ST3250310NS	OpenMPI version	1.6.4
Disk speed (rpm)	7200	IMB version	4.0
No. of nodes	12	OpenFOAM version	2.3.0
Interconnect speed (Gbps)	1		

network virtualization (using macvlan or veth LXC network type) and generate sufficient number of IP addresses for the cluster or use host network name space (none LXC network type) and generate only the port number to run SSH daemon on. The next step is to copy a (possibly amended) node file from the host into the first container and launch the submitted script inside it. When the script finishes its work, SSH daemon in every container gets killed and all containers are destroyed.

For this algorithm to work as intended, the client's home directory should be bind-mounted inside the container before launching the script. Additionally, since some Message Passing Interface (MPI) programs require scratch directories on each node to work properly, container's root file system should be mounted in copy-on-write mode, so that all changes in files and all the new files are written to host's temporary directory and all unchanged data are read from read-only network mounted file system; this can be accomplished via Union or similar file system and that way application containers are left untouched by tasks running on the cluster. To summarize, only standard Linux tools are used to build the system: there are no opaque virtual machine's images, no sophisticated full virtualization appliances and no heavy-weight cloud computing stacks in this configuration.

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

The first experiment was to create a 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 (Figure 2).

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 Figure 3. The jump after the 8th core marks the bounds of a single machine which means using network for communication rather than shared memory.

The creation of containers is a fully parallel task and it 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 mpi run time. So, the use of Linux containers adds some constant overhead to the launching of parallel task depending on system's configuration which is split between the creation of containers and filling the file cache.

The second experiment was to measure the performance of different LXC network types using IMB suite, and it was found that the choice of the network virtualization greatly affects the perfor-

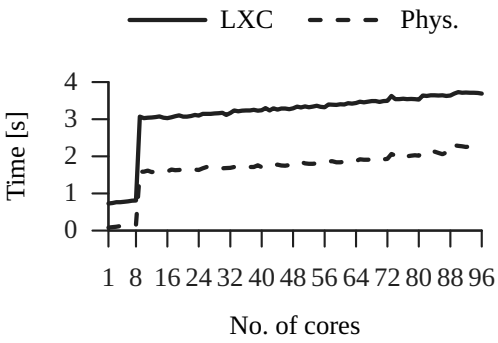

Figure 2. Comparison of LXC and physical cluster performance running an empty MPI program

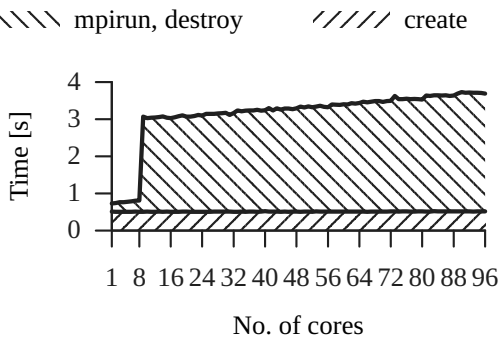

Figure 3. Breakdown of LXC empty MPI program run

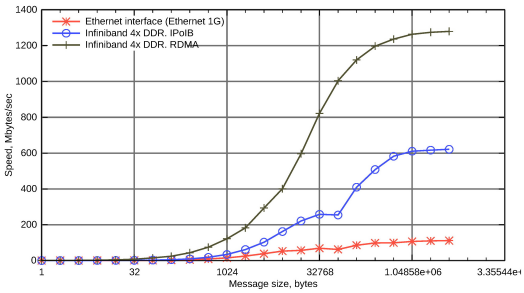

Figure 4. Network test 1 (cluster 1)

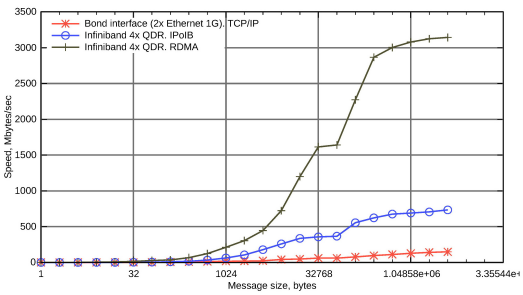

Figure 5. Network test 2 (hybrid cluster)

mance. Similar to the previous test, a container was set up with the same operating system and the same IMB executables as the host machine. Network throughput was measured with the exchange benchmark and displayed on the graph (Figure 4). From the graph it is evident, that the performance is approximately the same for all network types up to 214 bytes message size, however, after this mark there is a dip in the performance of the virtual ethernet. It is difficult to judge where this overhead comes from: some studies report that under high load the performance of bridged networking (veth is always connected to the bridge) gets decreased, but IMB does not put high load on the system. Additionally, the experiment showed that throughput decreases with the number of cores, as expected, due to synchronization overheads (Figure 5) [7].

The third and the last experiment dealt with a 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 Reynolds-Averaged Simulation (RAS) case was run with different numbers of cores (the total number of cores is the square of the number of cores per node) and different LXC network types and the average of multiple runs was displayed on the graph (Figure 6). Measurements for 4 and 9 cores were discarded because there was 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 the final performance of OpenFOAM by approximately 5–10% whereas *macvlan* and *none* performance are close to the performance of a physical cluster

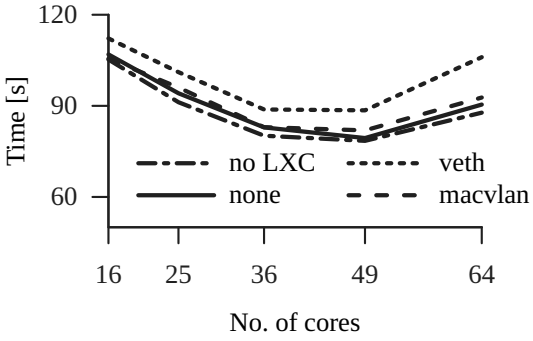

Figure 6. Average performance of OpenFOAM with different LXC network types

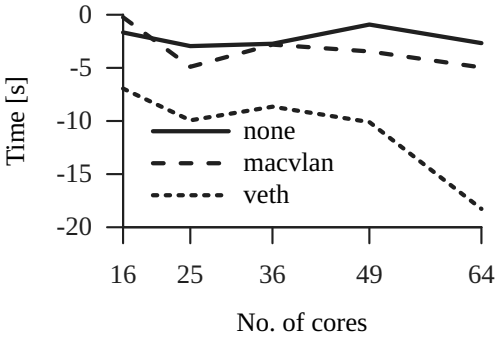

Figure 7. Difference of OpenFOAM performance on physical and virtual clusters. Negative numbers show slowdown of the virtual cluster

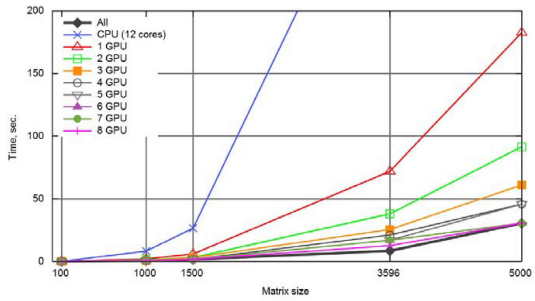

Figure 8. Trigonometric functions

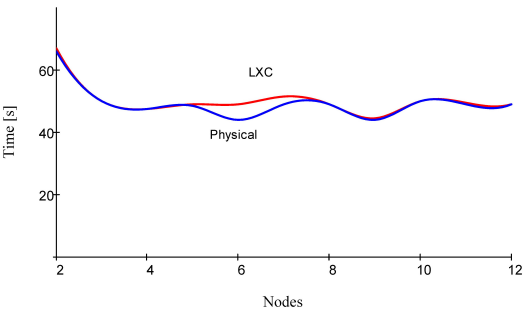

Figure 9. OpenFOAM run time form virtual and physical clusters

(Figure 7). We can thus conclude that the choice of the network type is the main factor affecting the performance of the 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.

To summarize, there are two main types of overheads when using virtual clusters: creation overhead which is constant and small compared to an average runtime of a typical parallel job, and network overhead which can be eliminated by avoiding the network virtualization at all.

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 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.

5 Examples of algorithms for virtual clusters implementation

1. As of late, the methods of functional integration are widely used in mathematical physics. The functional integral is a very effective tool, since it allows to represent the solution of a differential equation in partial derivatives in a closed and compact form. It can also serve as a basis for obtaining high-performance parallel algorithms.

   The main idea [2] is to make the transformation of variables in the functional or path representation of a quantum observable, so that

   – to reduce the Hamiltonian function of the problem to the one which is actually responsible for the described process;

   – to reduce the integration interval to a finite one, corresponding to the interaction region;

   – to transform asymptotical boundary conditions to standard ones.

   The method makes it possible to get the expression of the scattering amplitude as an average over Green's functions of the problem but in mixed representation and in interaction coordinates.

   As a result:

   – the phase of the path integral becomes a quadric functional;

   – the new representation of an observable is reduced to a standard multidimensional integral;

   – the solution, at every point in the coordinate space of that integral is reduced to a 1-st order partial differential equation system.

   The advantages of such operations are obvious:

   – the problem, although still difficult, can be very effectively parallelized;

   – the use of the proposed approach shows substantial speed-up over the standard algorithms.

   The approach above seems to become more effective with the increase of the size of a problem.

2. There is a remarkable application of the idea of the nonlinear integral transformation, outlined in the introduction, in physical and chemical kinetics. The fact is that the kinetic equations with a nonlinear transformation are reduced to quasidiagonal Riccati equations. Since the physical quantities of interest can also be effectively represented in the transformed coordinates, as in the initial populations, the problem of reverse transformation is not important.

   In realistic cases, the system of kinetic equations describing the relaxation in molecular systems consists of thousands of equations of the type

   $$\frac{Dc(\boldsymbol{j},t)}{Dt} = I(\boldsymbol{j}-\mathbf{1},t) - I(\boldsymbol{j},t) \tag{2}$$

   with $\boldsymbol{j}$ being the multi-index, describing molecular states and I being the molecular current in index space of the form $I(\boldsymbol{j},\boldsymbol{j}+\mathbf{1})\,c(\boldsymbol{j},t) - K(\boldsymbol{j}+\mathbf{1},\boldsymbol{j})\,c(\boldsymbol{j}+\mathbf{1},t)$, where K is a rate constant for molecule-central particle collision.

   The problems with (2) actually originate from two factors: there are large nondiagonal members, corresponding to important physical transitions, and values of $I(\boldsymbol{j},t)$ which are very large while the difference between them in the r.h.s. of (2) is relatively small. The situation becomes dramatic if the integration starts at thermal equilibrium, where all I's are equal.

To overcome these difficulties, it is useful to introduce new variables $f(\boldsymbol{j},t) = c(\boldsymbol{j}+\mathbf{1},t)/c(\boldsymbol{j},t)\,a(\boldsymbol{j},t)$ with $a(\boldsymbol{j},t)$ being the ratio of two rate constants $K(\boldsymbol{j},\boldsymbol{j}+\mathbf{1})$ and $K(\boldsymbol{j}+\mathbf{1},\boldsymbol{j})$. f's are so called slow variables, which become constants at equilibrium conditions. The equations for f's are

$$\frac{df(\boldsymbol{j},t)}{dt} = \tilde{R}(f(\boldsymbol{j},t)) + H(\boldsymbol{j},t)f(\boldsymbol{j},t) + S(\boldsymbol{j})f(\boldsymbol{j},t), \tag{3}$$

with $\tilde{R}$ being the quadric relaxational term that is diagonal in $\boldsymbol{j}$, H is the source term proportional to hydrodynamic gradients, and S is the source of the population change, the only term that is nondiagonal in $f(\boldsymbol{j},t)$. The main advantage of (3) is that not only the sum of three terms in r.h.s. (3) is small, but they are small separately and it is easy to determine their relative values beforehand. Moreover, it is important that the major contribution in r.h.s. of (3) is diagonal. This opens interesting opportunities for a parallel algorithm.

Such notation gives an opportunity to rewrite numerical schemes in tensor form. Tensor mathematics is naturally embedded in the finite- operation in the construction of numerical schemes.

3. The theory of nonlinear equations gives us an important example of the proposed approach. These problems are well understood, and many classes of completely integrable systems are well described [19]. In practice, however, one have to deal with systems that are not completely integrable. In the spirit of the approach set out in the introduction, we will consider them as perturbations of completely integrable systems. One implementation of this approach may be associated with the decomposition of the desired solution over the solutions of completely integrable equations. Then all nonlinearity is covered by these solutions, and for the residue we get simple differential equations, as in the conventional perturbation theory. As the realization of this idea let us look at an example of a generalized KDVB equation.

$$v_t + vv_x + \alpha v_{xx} + \beta v_{xxx} = \gamma I(v). \tag{4}$$

Here α is the measure of dissipation effects, β is the measure of dispersion and expressed via transport coefficients and relaxation times, γ is the measure of interphase interactions and is expressed via integral brackets and relaxational times, and $I(v)$ is the integral operator, that to a first approximation is linear, and can be expressed as

$$I(v) = -\int_0^t G(t,\tau)v_\tau\, d\tau$$

with G for different models of interaction being an exponential or inverse power function.

Using the decomposition over solutions of KdV equations, we get

$$\frac{da}{d\tau} = -\Phi(\kappa)\, a$$

with a being the first moment of v and

$$\Phi(\kappa) = 2\int_0^\infty \exp(-\kappa\theta)\int_0^\infty \cosh^{-2}(\theta+\gamma\theta')\times\cosh^{-3}\theta\sinh\theta'\, d\theta\, d\theta'.$$

In addition to simplifying the computing procedure the analysis of this equation allows qualitative conclusions, for example, predict the emergence of a pilot wave.

6 Examples of problems solved within the framework of 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 [4]. 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 [10]. 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 [6]. 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 [3, 13].

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) [15].

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 [8] an attempt to extend the described approach to the solution of this problem is carried out.

7 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 can not 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.

Acknowledgements

The research was carried out using computational resources of Resource Center Computer Center of Saint-Petersburg State University and supported by Russian Foundation for Basic Research (project N 13-07-00747) and St. Petersburg State University (projects N 9.38.674.2013, 0.37.155.2014).

References

[1] Arnold V.I. *Ordinary Differential Equations* (Springer-Verlag Berlin Heidelberg, 1992) 338 p.
[2] Bogdanov A.V., Sov. Phys. Tech. Phys. **31** (7), 833–835 (1986)
[3] Bogdanov A., Degtyarev A., Nechaev Yu., Proc. of V Int. Conf. on Computer Science and Information Technologies (CSIT'2005), Yerevan, Armenia, 393–398 (2005)
[4] Bogdanov A., Gevorkyan A., Nyman G., Physics of Atomic Nuclei, N 5, 876–883 (2008)
[5] Bogdanov A., Proc. of the 5th Int. Conf. "Distributed Computing and Grid Technologies in Science and Education," Dubna: JINR, 57–59 (2012)
[6] Bogdanov A.V., Degtyarev A.B., Khramushin V.N., Computer Research and Modeling **7**, i.3, 429–438 (2015)
[7] Bogdanov A., Gaiduchok V., Ahmed N., Cubahiro A., Gankevich I., in: Gervasi O., et al.(eds.) ICCSA 2015. LNCS **9158**, 299–310 (2015)
[8] Bogdanov A., et al., Mathematical Model of Psychology-Political Classification of Political Parties, Int. Conf. Mathematical Modeling and Computational Physics, Stará Lesná, Slovakia, (2015)
[9] Brewer E.A., Proc.of the XXIX ACM SIGACT-SIGOPS symposium on Principles of distributed computing, N.Y.: ACM **29**, N 1, 335–336 (2010)
[10] Degtyarev A.B., Khramushin V.N., Mathematical Modeling **26**, i.11, 4–17 (2014)
[11] Gankevich I., Korkhov V., Balyan S., Gaiduchok V., Gushchanskiy D., Tipikin Y., Degtyarev A., Bogdanov A., Proceedings of International Conference on Computational Science and Its Applications (ICCSA 2014), Lecture Notes in Computer Science Volume 8584, 341–354 (2014)
[12] Gankevich I., Gaiduchok V., Gushchanskiy D., Tipikin Y., Korkhov V., Degtyarev A., Bogdanov A., Zolotarev V., CSIT 2013 – 9th International Conference on Computer Science and Information Technologies (CSIT), Revised Selected Papers. DOI: 10.1109/CSITechnol.2013.6710358 (2013)
[13] Gankevich I., Degtyarev A., Proc. of IX Int. Conf. on Computer Science and Information Technologies (CSIT'2013), Yerevan, Armenia, pp. 240–244 (2013)
[14] Gankevich I., Tipikin Yu., Degtyarev A., Korkhov V., in: Gervasi O., et al.(eds.) ICCSA 2015. LNCS **9158**, 259–271 (2015)
[15] Korkhov V., Krzhizhanovskaya V., Sloot P., Journal of Parallel and Distributed Computing **68** (5), 596–608 (2008)
[16] Korkhov V., Kobyshev S., Krosheninnikov A., Computational Science and Its Applications – ICCSA 2015, Lecture Notes in Computer Science, vol. 9158, 2015, 342–353 (2015)
[17] Shoshmina I., Bogdanov A., Saint Petersburg State University Bulletin (Physics and Chemistry) i.3, 130–137 (2007)
[18] Sumetsky M.Yu. *Resonant tunneling of electrons through the two- and three-dimensional nanostructures* (Leningrad, STS USSR Academy of Sciences, 2 91-8/1283, 1990)
[19] Whitham G.B., *Linear and Nonlinear Waves* (John Wiley & Sons Inc., N.Y., 1974) 636 p.