SPE 163090

Advances in Modeling of Giant Reservoirs

Vladislav I. Dzyuba, TNK-BP; Kirill Yu. Bogachev, Moscow State University; Anton S. Bogaty, Alexander R. Lyapin, Almaz R.Mirgasimov, Alexey E. Semenko, Rock Flow Dynamics

This paper was prepared for presentation at the Mathematical Methods in Fluid Dynamics and Simulation of Giant Oil and Gas Reservoirs held in Istanbul, Turkey, 3–5 September 2012.

Abstract

In order to create an efficient system for modeling giant reservoir models, there is a whole range of technical challenges which have to be addressed. In this paper we concentrate on the topic of parallel scalability of complex computing systems like multi-CPU clusters and workstations with GPU processing cards. For multi-CPU distributed memory computing systems, it is shown that about 10 times improvement in parallel performance can be achieved if a new so called “hybrid” approach is used. In this “hybrid” approach, the usual MPI synchronization between the cluster nodes is being interleaved with a shared memory system thread based synchronization at the node level. It is demonstrated, that for some “black oil” models of real oil and gas fields, the parallel acceleration factor can exceed 1300 times for 4096 CPU cores. Even for the extreme example of a giant full field model containing over 14,000 production and injection wells, it is shown that a parallel acceleration of over 350 times can be achieved. For CPU-GPU, and CPU-CPU based systems, we compare the parallel performance of simple iterative and realistic pre-conditioner based algorithms typically used in oil and gas simulations. Hardware systems equipped with AMD FirePro, nVidia TESLA and 16-core dual Intel Xeon E2580 systems are compared in this study.

1. Introduction

History matching of giant reservoir models is an important task necessary to improve the efficiency of production operations [1,2] and relies heavily on cluster HPC computing. All modern computing systems, even those called “shared memory”, are in fact highly heterogeneous especially when it comes to I/O performance of data transport between CPU cores and memory. The same CPU process allocating memory in its own memory bank versus the memory bank connected to another CPU would take significantly less processing time. Modern hardware chipsets and operational systems allow software applications to optimize physical memory allocation by providing NUMA (Non-Uniform Memory Access) technology. Dropping NUMA for two CPU 16-core workstation can degrade the overall parallel performance by 30 - 40% [3]. For cluster systems, the non-unformity becomes even more important since data now has to be transported between CPU cores of different nodes and the speed of the network becomes the major bottleneck.

All the factors mentioned above led to the creation of hierarchical software technology which uses different data transport and underlying synchronization mechanisms at different physical levels of the cluster systems. In the hybrid approach, each multi CPU multicore node becomes a single MPI process, while each CPU core runs separate system threads accessing the shared memory of the node [4, 5, 6, 7].

Performance and scalability differences as well as the memory footprint of the new Hybrid system as compared to a standard MPI based solution are discussed. An extreme example of a dynamic reservoir model with more than ten thousand wells is used to probe the limits of parallel scalability of the proposed Hybrid system. One should note that while hybrid technology has been proposed and implemented in other industries long ago, it is the first time it is applied in solving oil and gas reservoir simulation problems.

Many modern day clusters contain computing nodes that are equipped with a new generation of co-processor GPU boards. The raw floating point performance of these boards is usually added to the overall cluster number crunching power measured in TFLOPs. However, the necessity of using double floating point precision and complex data processing algorithms required in dynamic reservoir simulations puts severe limitations on the actual performance

value that these systems could bring. To compare parallel performance of CPU and GPU based systems, we manually port C++ parallel algorithms into CUDA and OpenCL languages and run them with realistic data samples on different hardware configurations. For fair testing, we use dual CPU and CPU-GPU based systems from approximately the same price range.

2. Understanding hybrid

One of the principle differences of the hybrid system versus MPI is in the partial shifting of the data transport load from the cluster network (e.g., Infiniband ~5GB/s) to the much faster data exchange channels (e.g., Intel QPI links with up to 64GB/s) available inside the shared memory nodes. For a cluster with dual CPU and 8 cores per node, the sharp divergence between MPI and Hybrid happens already at 16 nodes, as shown in the **Fig.1** below. The better data communications on the node level and higher the density of the cores effectively interconnected at each CPU node - the earlier MPI and Hybrid diverge. For example, for dual Intel Xeon E5 2680 Xeon based nodes with 16 cores per node, the point of Hybrid vs.MPI divergence would shift left from 16 to 8 nodes.

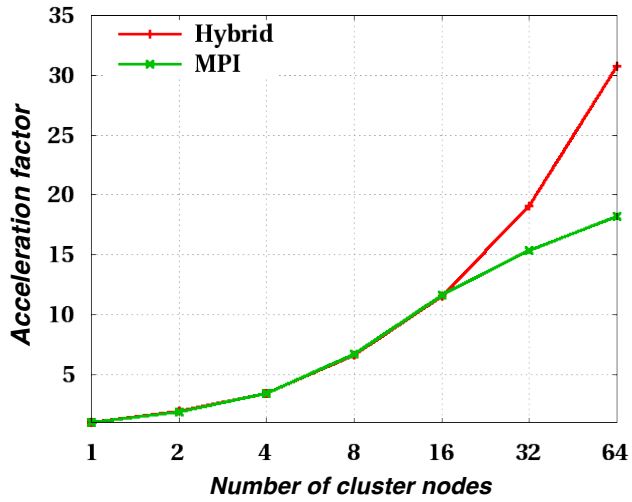

Figure 1. *Parallel performance acceleration for Hybrid (red) and MPI (green) as a function of the cluster nodes used. Cluster nodes used for this study have dual CPUs with 8 cores per node.*

Another less obvious but important benefit of Hybrid over MPI is that it requires a much smaller memory footprint as a function of number of cluster nodes used. The difference comes from the fact that inside the shared memory nodes there is only a small overhead related to the exchange of boundary conditions between parallel domains (e.g. matrix overlaps), since most manipulations are performed at the logical level inside the common memory address space. In the **Fig.2** below, for dual CPU cluster nodes with 8 cores per node, one can see the difference in the memory footprint for Hybrid and MPI as a function of the number of nodes used.

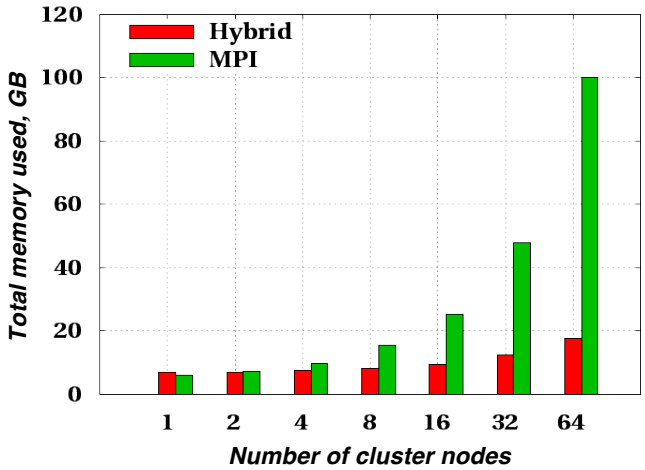

Figure 2. *Example of memory footprint for Hybrid (red) and MPI (green) as a function of the cluster nodes used.*

From **Fig.2,** one can see that for the same 64-node cluster, Hybrid users are able to run up to 5 times larger hydrodynamic models than users relying on pure MPI technology.

3. Extreme testing of hybrid performance

Using various full-field models of real oil and gas fields of different grid sizes, number of wells, and physics, for clusters from 240 to 4096 cores of Intel Xeon X5570 and 1333MHz DDR3 memory, there has been no indication of saturation effects in parallel performance. For example, while comparing runs with 1 and 4096 CPU cores used to simulate a black oil model with 39 wells and 21.8 million active grid blocks, the total elapsed simulation time (not just the linear solver part) was reduced by a factor of 1328, as can be seen in [8,9,10]. It is remarkable, that for this model, significant acceleration was still observed when the core count went well beyond 2048. In fact, when the number of CPU cores doubled from 2048 to 4096, the overall simulation performance increased 1.4 times. This behavior is very different from all known commercial simulators, where the benefit of adding extra nodes typically stops at 16 nodes.

To push the new "hybrid" algorithm to a new extreme, we select a giant "black oil" three-phase model with 43.5 million active grid blocks and more than 14,000 thousand production and injection wells. This massive waterflood model contains several reservoirs with a significant gas cap, multi-branch PVT correlations, multiple PVT and equilibrium regions, horizontal, vertical, and deviated wells. Small fragments of the model are shown below (**Fig3, Fig.4**) as 2D and vertical profile projections of water and water-oil-gas saturations, respectively.

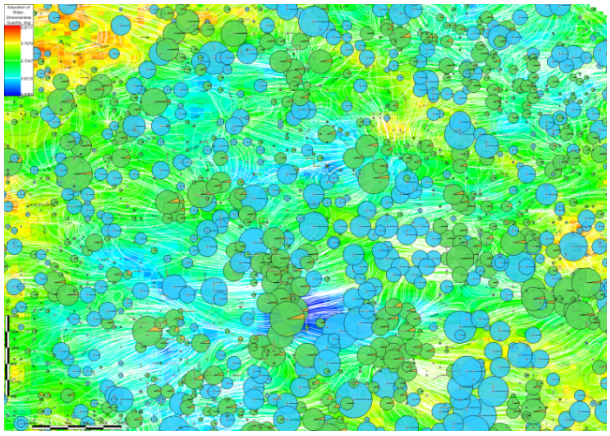

Figure 3. *Map of water saturation averaged over vertical axis for a small segment of massive waterflood reservoir model with more than 14,000 wells. Injection wells are shown as blue bubble diagrams, production wells are shown as green-orange bubble diagrams representing oil-water rate fractions. Finite difference streamlines are shown as white lines, contours are shown as black lines.*

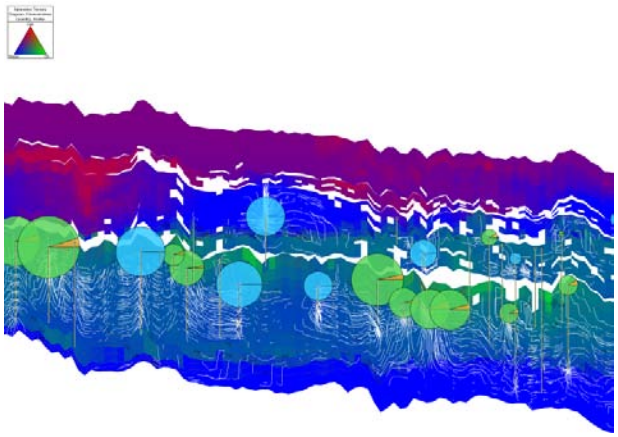

Figure 4. *Ternary distribution of oil-water-gas saturations projected on vertical profile. Gas distribution is shown in red, water in blue, and oil in green. Injection wells are shown as blue bubble diagrams, production wells are shown as green-orange bubble diagrams. Streamlines are shown as white lines, contours are shown as black lines.*

It is known that presence of wells with multiple perforations and the possibility of crossflow and long trajectories can cause significant numerical problems even for serial reservoir simulations. Trying to run such models over multiple cluster nodes, CPUs and cores is a much harder problem. Each well introduces a separate well equation, which has to

be solved with the rest of the differential equations representing dynamic properties of the grid. Models with a couple of thousand wells are typically considered to be a challenge, so our hydrodynamic model of a giant oil field with over 14,000 wells definitely can be considered as an extreme test for Hybrid parallel simulation technology.

When it comes to the logistics of handling giant models of such scale, one has to solve many additional problems related to the shear amount of data. It was found that the most popular commercial geological packages start to experience problems with handling large grids and well data. Even such things as 3D grid visualization and limited I/O performance of disk systems can become a challenge for reading and writing data of such scale in a reasonable amount of time.

Another problem is related to the fact that sometimes it is necessary to run proprietary reservoir models on some kind of public HPC cluster system. In our case we used one of the world's TOP20 systems called Lomonosov hosted at Moscow State University [11] and shown in the **Fig. 5** below.

Figure 5. *Lomonosov cluster at Moscow State University high performance computing center.*

The approach that was taken included two additional steps. All the input data files describing the hydrodynamic model were compressed using the publicly available "gzip" library. The resulting compression factor turned out to be close to 10 times and helped to reduce file sizes and related I/O times. The compressed data files were then encrypted using one of the 128-bit public key algorithms. All the decryption and uncompressing algorithms were integrated into the commercial simulator tNavigator that allowed performing these actions "on the fly" in the operational system memory of individual cluster nodes. Decryption and uncompressing is done on the level of the individual cluster nodes, which allows running both algorithms in parallel. Also, on the data security side, at no time was an entire model assembled unprotected as one piece in memory or on disk.

The implemented compression/encryption technology described in this paper can be used for public cloud environments effectively combining massive and inexpensive computing power and ensuring a high level of data protection.

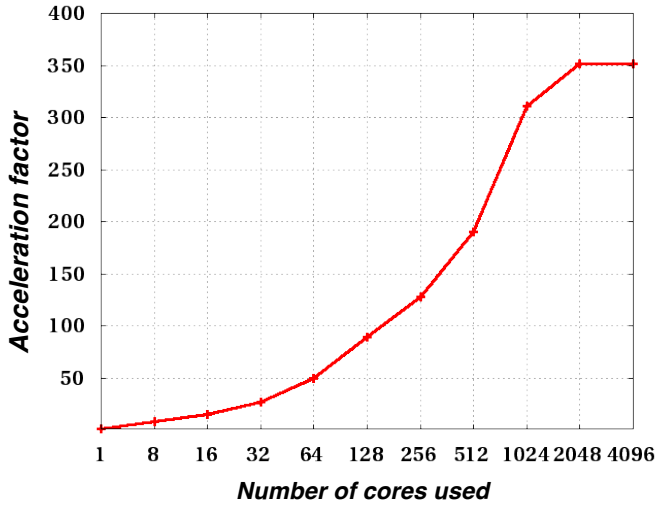

Figure 6. *Parallel acceleration factor as a function of number of cores used measured for 43.5 million active grid block "black oil" model with over 14,000 wells.*

Once the "black oil" model with 43.5 active grid blocks and over 14,000 wells was successfully encrypted, compressed, transferred over the network and securely loaded into the cluster nodes, a set of parallel runs with a different number of nodes was taken. The dependence of the parallel acceleration factor on the number of CPU cores used is shown in **Fig. 6**. The maximum acceleration rate was found to be 352 times and appears to stay unchanged between 2048 and 4096 cores.

To understand the origins of the observed saturation, a detailed profiling of the parallel performance has been done. It was no surprise to find that when all 4096 cores were used, the amount of time spent on resolving all the well equations alone actually exceeded the CPU time spent on the linear solver.

4. Performance of CPU-GPU systems

To ensure a comprehensive and fair performance comparison of parallel reservoir simulations run on GPU and CPU based systems the following three workstation configurations have been selected [12, 13, 14]:

- nVidia Tesla C2075 and Core i7 950, DDR3 1600MHz

- ATI FirePro v9800 and Core i7 950, DDR3 1600MHz

- Dual Xeon E5-2680 (16-core 2.8GHz), DDR3 1600MHz

All three workstations have approximately the same price range and represent top of the line single and dual socket workstations widely available from many vendors worldwide.

Data samples used for tests are randomly selected sparse matrices exported from production models of real oil and gas fields with different numbers of active grid blocks, wells, and underlying physics. To compare parallel performance, we choose two types of parallel algorithms. The first type is called iterative [15] and believed to be the most suitable for GPU based computing, since it contains only the most trivial vector-matrix arithmetic operations like adding and multiplication. The standard BiCGStab algorithm is used as an example of an iterative algorithm. Even though the iterative algorithms are ideally suited for GPU-type processing, they are not well suited for asymmetric, poorly defined sparse matrices, which are extremely common in oil and gas dynamic reservoir simulations. As a result, only a fraction (7) of initially selected matrices (46) allowed BiCGStab to converge.

The second type of numerical algorithm is adopted directly from parallel tNavigator, which is believed to be very well understood and tested with thousands of real reservoir models in shared and distributed memory environments on multicore workstations and clusters. This algorithm is based on the parallel block ILU(0) pre-conditioner and overlap method for effective parallelization. This is the algorithm which demonstrated one of the world's highest parallel acceleration factors of 1328 times for 4096 cluster cores.

To run the pre-conditioner based algorithm implemented in C++ for GPU boards, a semi-manual conversion of C++ code to CUDA and OpenCL languages, designed for massive parallel systems, was done. After conversion, there are still a number of additional steps, mostly in memory and kernel threads management, which have to be taken to maximize GPU parallel performance. Both CUDA and OpenCL are C-like languages with their program functions executed on one or several GPU kernels by using threads combined into groups. This is a reflection of the actual architecture of GPU chips, where kernels are grouped by blocks. The synchronization of each core is available only for threads contained in one group. Only a limited number of groups can be run simultaneously: 1024 for C2075 and 256 for FirePro v9800. The size of the groups has to be modified for the maximum performance.

The important difference between CUDA and OpenCL is that while CUDA can be run only on nVidia GPUs, OpenCL can be run on both AMD and nVidia GPU cards. In CUDA and OpenCL there are three layers of memory: shared memory, memory shared by the group of threads and local thread variables. The shared memory is the biggest (4GB in FirePro, 6GB in Tesla) and the slowest of three. To compensate for high latency, one needs to allocate the number of threads exceeding the number of physical cores (1600 cores for FirePro, 448 cores for Tesla). Memory shared by the group of threads is faster than the shared memory by an order of magnitude, but much smaller in size: 32KB in FirePro, 64KB in Tesla.

Overall, the migration of non-trivial C++ code into CUDA and OpenCL with the goal of reaching maximum performance requires a fundamental rewrite of the original code and is far from being a trivial or automatic process.

The performance of the iterative BiCGStab algorithm, as well as CUSP library from nVidia, which successfully completed only for 7 matrices, is shown below in **Fig.7.**

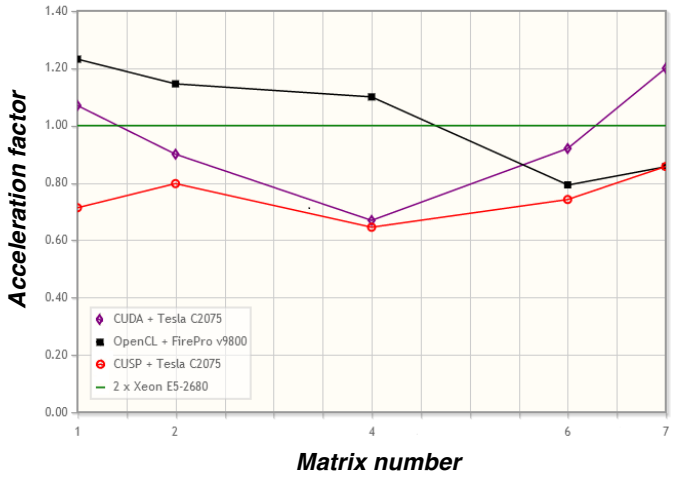

Figure 7. *Parallel acceleration factor for different matrices suitable for iterative algorithm. Black line is OpenCL run on FirePro v9800, green is C++ on 16-core dual Xeon E5-2680 workstation, violet is CUDA on Tesla C2075, red is CUSP from nVidia on Tesla C2075.*

As one can see from **Fig. 7**, even for fully iteractive algorithms best suited for the GPU environment, only in two cases did the CUDA version run on Tesla C2075 GPU outrun the C++ version of the same algorithm run on 16-core Xeon.

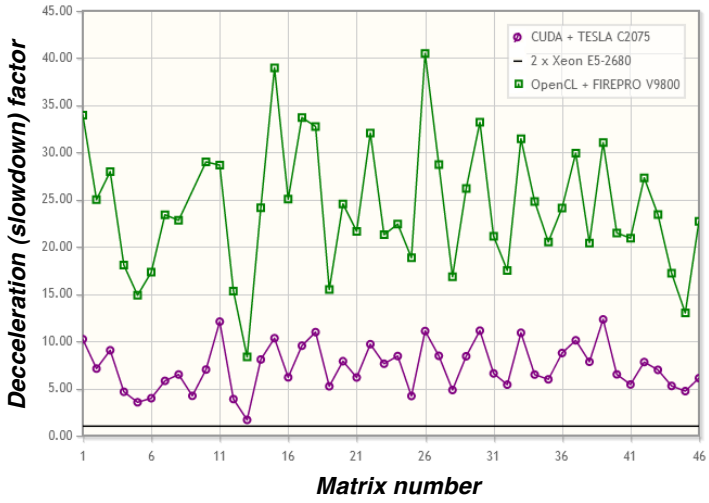

Figure 8. *Parallel deceleration (slowdown) factor for different matrices solved with pre-conditioner based algorithm. Green line is OpenCL run on FirePro v9800, black is C++ on 16-core dual Xeon E5-2680 workstation, violet is CUDA on Tesla C2075*

After switching from pure iterative algorithms to the more stable and useful for realistic dynamic reservoir simulations pre-conditioner based block ILU(0) algorithms, the situation changes considerably. As one can see from **Fig. 8**, only in one case out of 46 did the CUDA version of the algorithm perform comparable with C++ version run on a 16-core dual Xeon E5-2680 workstation. For the rest of the 45 matrices, GPU performance is from 5 to 40 times slower than dual CPU 16-core workstation performance. It is important to point out, that for tests performed with GPU cards the ideal conditions are assumed. All the I/O overheads related to having physically different GPU and CPU memory cards and limitation caused by the limited sizes of GPU cards are not included into the elapsed time measurements.

The last set of measurements is done to compare performances of different GPU cards using CUDA and OpenCL languages using the block parallel ILU(0) algorithm. As can be seen in **Fig. 9,** the clear winner is CUDA run on TESLA C2075.

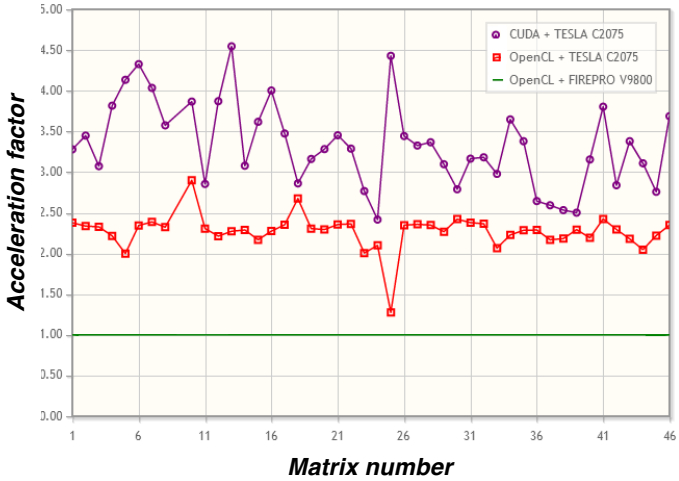

Figure 9. *Parallel acceleration factor for different matrices solved with pre-conditioner based algorithm. Green line is OpenCL run on FirePro v9800, red line is OpenCL on Tesla C2075 , violet is CUDA on Tesla C2075*

5. Conclusions

A parallel Hybrid algorithm mixing MPI and system threads at the node level was tested for various full-field models of real oil and gas fields and demonstrated a record parallel acceleration exceeding 1300 times on a 4096 node cluster. Even for the extreme case scenario, where a "black oil" model with 43.5 active grid blocks and over 14,000 wells that was run on a cluster with up to 4096 cores - an acceleration factor of 352 times was reported. The saturation in parallel acceleration between 2048 and 4096 cores was caused solely by the presence of over 14,000 production and injection wells.

It was demonstrated that the Hybrid parallel algorithm outperforms classical MPI parallel scaling and memory footprint already at 16 nodes (for 8 cores/node cluster).

Using iterative and pre-conditioner based algorithms the performance of AMD and nVidia GPU-CPU and 16-core CPU-CPU Intel Xeon E5-2680 workstation were compared. Based on 46 matrices sample, the performance of parallel block ILU(0) coded in C++ run on a 16-core workstation was found to be superior to GPU performance with CUDA and OpenCL from 5 to 40 times.

It appears that with parallel Hybrid technology, history matching of giant reservoirs gets a significant boost when a higher number of CPU cores/node is used. Upgrading to 16-core Intel Xeon E5-2600 based nodes from 8 core/node configurations will greatly boost cluster performance, and will continue to do so, if densities grow even higher to 20, 24 core/node in 2013 - 2014.

With the current state of GPU development, the benefits from using CPU-GPU configurations remain unclear. It may take several more generations of GPU architecture to become competitive with dual CPU Intel Xeon based performance.

References

1. F. Briens, Lundin Oil; H.T. Chua, Lundin Malaysia Ltd., «Reservoir Simulation Applied to Production Operations», SPE 66367-MS, SPE Reservoir Simulation Symposium, 11-14 February 2001, Houston, Texas.
2. M.A. Christie, SPE, BP Exploration, Upscaling for Reservoir Simulation, SPE 37324-MS, JPT, Vol. 48, Num. 11.
3. Bogachev K.Yu., Mirgasimov A.R. On optimization of computing applications for multiprocessor systems with nonuniform memory access. Numerical methods and programming, 2010, vol. 11, pp. 198-209.
4. Chapman B. Parallel Application Development with the Hybrid MPI + OpenMP Programming Model. Lecture Notes in Computer Science. 2002. Vol. 2474. Pp. 13-27.
5. Rabenseifner R., Hager G., Jost G. Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes // Proceedings of the 17th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2009, Weimar, Germany, 18-20 February 2009. Pp. 427-436.
6. Jost G., Robins B. Experiences using hybrid MPI/OpenMP in the real world: Parallelization of a 3D CFD solver for multi-core node clusters. Scientific Programming. 2010. Vol. 18, no. 3-4. Pp. 127-138.
7. Jin H., Jespersen D., Mehrotra P., Biswas R. High Performance Computing Using MPI and OpenMP on Multi-core Parallel Systems Parallel Computing. 2011 Feb.

8. V. Dzuba, et al., SPE 162090
9. Bogachev K.Yu., Klimovsky A.A., Mirgasimov A.R., Semenko A.E. Load balancing of nodes for cluster systems with distributed memory is considered for modeling of filtration of viscous compressible fluids. Numerical methods and programming, 2011, vol. 12, pp. 70-73..
10. Bogachev K.Yu., Zhabitskiy Ya.V., Klimovsky A.A., Mirgasimov A.R., Semenko A.E. Different methods for solving sparse systems of filtration equations for distributed memory cluster systems. Numerical methods and programming, 2011, vol. 12, pp. 74-76.
11. Lomonsov cluster at Moscow State University: http://parallel.ru/cluster/lomonosov.html
12. Fermi Architecture Whitepaper, http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf
13. ATI FirePro v9800, http://www.amd.com/us/Documents/ATI_FirePro_V9800_DataSheet.pdf
14. Intel Xeon E5, http://www.intel.com/content/www/us/en/processors/xeon/xeon-processor-5000-sequence.html
15. Saad Y. Iterative methods for sparse linear systems. Second edition. Philadelphia, PA, USA: SIAM Press, 2003. Pp. XVIII + 528.
16. X.-H. Wen, ChevronTexaco Energy Technology Co.; L.J. Durlofsky, ChevronTexaco Energy Technology Co., Stanford U.; Y. Chen, Stanford U., Efficient Three-Dimensional Implementation of Local-Global Upscaling for Reservoir Simulation, SPE 92965-MS, SPE Reservoir Simulation Symposium, 31 January-2 February 2005, The Woodlands, Texas.
17. Culler D. E., Singh J. P., Gupta A. Parallel computer architecture: a hardware/software approach. San Francisco, CA, USA: Gulf Professional Publishing, 1999. 1025 pp.
18. Gropp W., Lusk E., Skjellum A. Using MPI, Portable Parallel Programming with the Message Passing Interface. 2nd edition. Cambridge, MA, USA: MIT Press, 1999. 395 pp.
19. Bogachev K.Yu. The basics of parallel programming. “Binom”, Moscow, 2003. 342 pp.
20. Chapman B., Jost G., Van der Pas R. Using OpenMP, Portable Shared Memory Parallel Programming. Cambridge, MA, USA: MIT Press, 2007. 353 pp.
21. D. DeBaun, T. Byer, ChevronTexaco; P. Childs, J. Chen, F. Saaf, M. Wells, J. Liu, H. Cao, L. Pianelo, V. Tilakraj, P. Crumpton, D. Walsh, Schlumberger; H. Yardumian, R. Zorzynski, K.-T. Lim, M. Schrader, V. Zapata, ChevronTexaco; J. Nolen, Schlumberger; H. Tchelepi, ChevronTexaco, An Extensible Architecture for Next Generation Scalable Parallel Reservoir Simulation, SPE 93274-MS, SPE Reservoir Simulation Symposium, 31 January-2 February 2005, The Woodlands, Texas.
22. P.A. Fjerstad, A.S. Sikandar, H. Cao, and J. Liu, Schlumberger, and W. Da Sie, Chevron, Next Generation Parallel Computing for Large-Scale Reservoir Simulation, SPE 97358-MS, SPE International Improved Oil Recovery Conference in Asia Pacific, 5-6 December 2005, Kuala Lumpur, Malaysia.
23. M EhteshamHayder and MajdiBaddourah, Saudi Aramco, Challenges in High Performance Computing for Reservoir Simulation, SPE 152414-MS, SPE Europec/EAGE Annual Conference, 4-7 June 2012, Copenhagen, Denmark.
24. Preconditioners in problems of multicomponent fluid flow filtration in porous medium. MSU Vestnik, Ser. 1, Mathematics, Mechanics, 2010, N. 1, pp. 46-52.
25. Rock Flow Dynamics, 2012. tNavigator User Manual .