Iterative Mapreduce MPI Oil Reservoir Simulator

Madina Mansurova, Darkhan Akhmed-Zaki, Adai Shomanov,
Bazargul Matkerim and Ermek Alimzhanov
Department of Computer Science, al-Farabi Kazakh National University, Almaty, Kazakhstan

Keywords: MapReduce, MPI, Distributed Parallel Computing, Memory-mapped Files.

Abstract: Paper presents an advanced Iterative MapReduce MPI oil reservoir simulator. First we present an overview of working implementations that make use of the same technologies. Then we define an academic example of numeric problem with an emphasis on its computational features. We present a distributed parallel algorithm of hybrid solution of the problem using MapReduce Hadoop and MPI technologies and describe an improved variant of the algorithm using memory-mapped files.

1 INTRODUCTION

At present, effective solution of large-scale computational problems of oil-gas industry is related to the use of high-performance computational technologies. According to (Tokarev et al., 2012), "of the 500 most powerful supercomputers of the world, their use in geophysics by the companies of oil-gas service for searching, prospecting and mining of deposits holds the third place". Thus, the choice of the corresponding parallel software and parallel models of computations depending on the volumes of scientific calculations is an actual problem. In many cases, solution of problems of oil-gas industry comes to oil reservoir simulation.

This work presents the results of scientific investigations on creation of hybrid solutions for the problems of oil-gas industry. The earlier developed constructive approach of hybrid combination of MapReduce and MPI technologies (Mansurova et al., 2014) was used to solve the problem of calculating fluid pressure in an oil reservoir. In order to accomplish this task we first identify key features of the existing solutions in this domain in Section 2. Section 3.1 describes a mathematical model of the problem of fluid in elastic porous anisotropic medium. Section 3.2 presents a distributed algorithm of the problem solution using Mapreduce Hadoop technology. Section 3.3 presents a distributed parallel algorithm of hybrid solution of the problem using MapReduce Hadoop and MPI technologies. Section 3.4 describes an improved variant of the algorithm using memory-mapped files. Section 4 presents implementation and evaluation of algorithms performance. Finally, Section 5 concludes the paper.

2 RELATED WORK

The principles of organization of parallel and distributed computing have been known for a long time (Chen et al., 1984, Gropp et al., 1996, Sunderam et al., 1994). MPI and MapReduce can be referred to the most used technologies. MPI technology is the main instrument for parallel computing, when solving a wide spectrum of problems. However, with the increase in the volume of the data being processed there arise a question of reliability of MPI applications. In recent years the technologies of distributed computing MapReduce is being more widely recognized.

Creation of hybrid solutions allows using of the advantages of separate technologies. There exist a great variety of such solutions. The authors of (Hoefler et al., 2009) made the first attempt to write MapReduce with MPI. They implemented MapReduce using basic point-to-point and collective operations based on the original MapReduce model. The authors of the paper (Lu et al., 2011) compare MPI and MapReduce technologies from the point of view of the system failure. A numerical analysis is made to study the effect of different parameters on failure resistance. The authors believe that their research will be useful in answering the question: at what volumes of data it is necessary to decline MPI

Mansurova M., Akhmed-Zaki D., Shomanov A., Matkerim B. and Alimzhanov E..
Iterative Mapreduce MPI Oil Reservoir Simulator.
DOI: 10.5220/0005559204560461
In *Proceedings of the 10th International Conference on Software Engineering and Applications* (ICSOFT-EA-2015), pages 456-461
ISBN: 978-989-758-114-4
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

and use MapReduce in case of possible failures of the system. They implemented the original MapReduce model using MPI and obtained a speed improvement comparing to Hadoop based on MPI communications. The study of primitives of MPI and MapReduce communications allowed the authors (Mohamed et al., 2012) to assert the fact that MPI can give the rise in performance of MapReduce applications.

The work (Slawinski et al., 2012) considerable differs from the above presented works in which MPI technology is built in the environment of MapReduce. The authors describe the reverse task – the start of MapReduce applications in MPI environment. It is pointed out that several additional MPI functions should be written for full support of MapReduce.

The authors (Mohamed et al., 2012) modified the MapReduce model to achieve speed up using MPI. In the model, the authors propose the idea of overlapping the map, the communication and the reduce phases with a more detailed policy for the communication and data exchange.

Nevertheless, many of these functions are, as a rule, recognized to be important and are developed in MPI to support other modern paradigms of programming and parallelization. In (Srirama et al., 2011) the essence of the approach is considered to be division of implementation of MPI applications to sequence of computation stages each of which is completed by the stage of communication. This approach is based on the conception of adapters distributed in conventional utilities for coordination of the requirements to applications and platform hardware.

Other applications for adaptation of MapReduce model to organization of parallel computing are given in (Matsunaga et al., 2008, Biardzki et al., 2009, Ekanayake et al., 2010, Bu et al., 2012, http://mapreduce.sandia.gov). As a whole, the problems of effective organization of iterative computing on MapReduce model remain, especially; the problems of scalability of such algorithms and their adaptation for a wide range of scientific problems, there are neither rigorous approach to provide reliability of such systems.

3 THE WORK OF ITERATIVE MAPREDUCE MPI OIL RESERVOIR SIMULATOR

3.1 A Mathematical Model of Fluid Pressure in the Oil Reservoir

Let us consider a hypercube in anisotropic elastic porous medium

$$\Omega = [0,T] \times K\{0 \le x \le 1,0 \le y \le 1,0 \le z \le 1\}.$$

Let equation (1) describes the fluid dynamics in hypercube Ω under initial conditions (2) and boundary conditions:

$$\frac{\partial P}{\partial t} = \frac{\partial}{\partial x}(\phi(x,y,z)\frac{\partial P}{\partial x}) + \frac{\partial}{\partial y}(\phi(x,y,z)\frac{\partial P}{\partial y}) + \frac{\partial}{\partial z}(\phi(x,y,z)\frac{\partial P}{\partial z}) + f(t,x,y,z) \quad (1)$$

$$P(0,x,y,z) = \varphi(0,x,y,z) \quad (2)$$

$$\left.\frac{\partial P}{\partial n}\right|_{\Gamma} = 0 \quad (3)$$

Here Γ under conditions (3) is the surface of cube Ω. In equation (1) the solution function $P(t, x, y, z)$ is seam pressure in point (x, y, z) at the moment t; $\phi(x, y, z)$ is diffusion coefficient in the reservoir; $f(x, y, z)$ is density of sources. To solve (1)-(3) Jacobi's numerical method was used (Imankulov et al., 2013). For problem (1)-(3) a parallel algorithm of solution was realized using MPI technology presented in (Matkerim et al., 2013).

3.2 Iterative MapReduce Architecture

According to MapReduce paradigm, realization of the algorithm for iteration processes is a series of MapReduce tasks (Ekanayake et al., 2010, Bu et al., 2012). Algorithm of numerical solution of the problem (1)-(3) with the help of MapReduce Hadoop technology consists of two stages: the stage of initialization at which MapReduce work of the first level is performed only once and the iteration stage at which a cycle of MapReduce works of the second level is performed. Mapper of the first level loads data from the file system HDFS. Then, Mapper distributes the data between Reducer processes on slabs, thus realizing 1D decomposition of the data. Reducer, in its turn, performs computations, duplications boundary slabs into the ghost slabs of the neighbors and stores the obtained

results. The data used by Reducer for computations are divided into two kinds: local data, i.e. the data which refer to the interior slab and shared boundary data (boundary slab). Reducer enters local computed data directly into a local file system and enter the shared boundary data into the output file of the distributed file system HDFS, which will be an input file for Mapper of the second level at the next iteration. At each iteration Mapper of the second level distributes the updated boundary data among Reducers, thus providing the exchange of boundary values between slabs. The fluid of data corresponding to the description is presented in Figure 1.

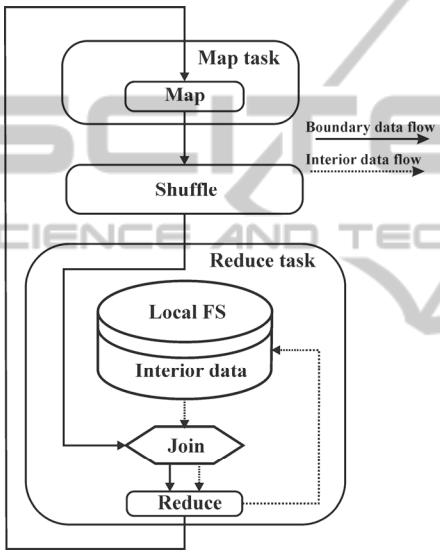

Figure 1: Iterative MapReduce framework scheme.

3.3 The Distributed Algorithm of the Fluid Dynamics in Oil Reservoir based on MapReduce Hadoop Technology

The distributed algorithm consists of two stages:

- The stage of initialization;
- The iteration stage.

The stage of initialization is a MapReduce task "Initial" in which there takes place initialization and writing of files necessary for computations in the process of iterations.

The iteration stage is a MapReduce task "Iterations". At each iteration in Mapper, points of the field with the same keys, i.e. numbers of subcubes, are grouped. The input data of Mapper are the output data of Reducer. In Reducer, the main computations are performed according to the formulae of the explicit method. Then, writing of the interior parts of files into the local file system and transfer of values of boundary slabs to the output of Reducer "Iterations" are performed.

3.4 A Distributed Parallel Algorithm of the Fluid Dynamics in an Oil Reservoir based on MapReduce-Hadoop and MPI Technologies

A distributed parallel algorithm of numerical solution of the problem (1)-(3) consists of the following main steps:

1. Initiation of the script on the main node of the cluster which copies the parameters of dimensions of computation region, the numbers of MPI processes, the number of nodes of Hadoop cluster necessary for computations onto all computational nodes of the cluster.
2. The stage of initialization. Initiation of MapReduce task which performs computations in every point of computation region at the zero moment of time with fulfillment of initial conditions.
3. The iteration stage. Initiation at i^{th} iteration of MapReduce task to the input of which enter all computed values in points of computation region from the (i-1) iteration. If the number of iteration is equal to zero, all data from the initial stage enter, in other cases, the input data are formed only from the boundary values of points which were obtained at the previous iteration.

On each of Reducers, non-boundary data are stored on the node in which the given Reducer operates and the boundary values Reduce into the distributed file system so that later they can be distributed onto other nodes which share the boundary values with the current node.

After reading out boundary and non-boundary values, each of the Reducers initiates the process of distribution of points to different files so that the number of MPI processes and each MPI process will perform computations in its subregion of data which it reads out from the file with the name corresponding to its rank (Fig. 2). After reading out the data from the file corresponding to its rank, each MPI process performs computations of the pressure equation in its subregion of data. After that, it performs writing of new values of points into the file from which it read out the data. After completing all MPI processes, Reducer gathers data and, depending on the fact whether the point is boundary or not, it performs operations of reducing or writing into the local memory of the node.

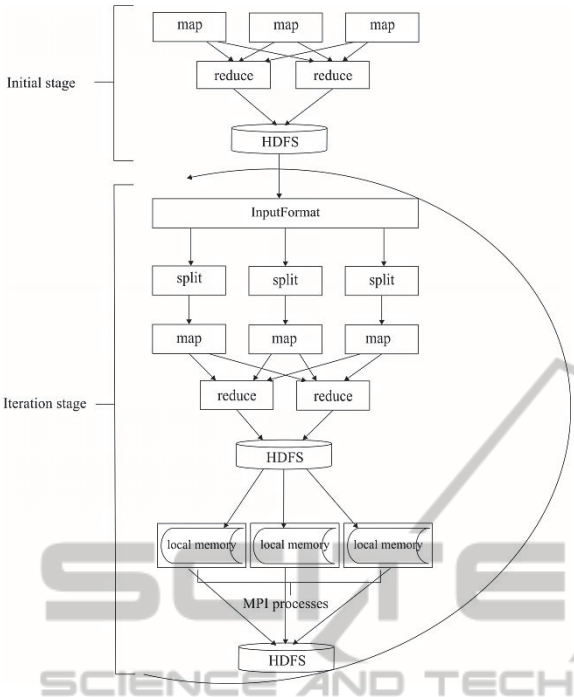

Figure 2: Scheme of MapReduce MPI algorithm.

3.5 A Distributed Parallel Algorithm of the Fluid Dynamics in Oil Reservoir using Memory-Mapped Files

Realization of the distributed parallel algorithm with writing of data into files for their subsequent processing by MPI processes showed that reading / writing operations cause significant delays of performance. In this regard, in this section we propose to use the kind of files called memory-mapped files (MMF).

The peculiarity of these files is that, when working with them, the whole file or a definite continuous part of this file is corresponded by a definite site of memory (range of addresses of operative storage), that allowing to considerably accelerate realization of reading/ writing operations (http://en.wikipedia.org/Memory-mapped-file).

To realize the algorithm of MapReduce MPI, we used the method of data exchange between Hadoop and MPI which uses the library Java Chronicle. Java Chronicle is a data base which is stored in operative memory and possesses the following properties:

1) A low level of delays;
2) High capacity;
3) Persistence (i. e. after completion of the program the data are stored in memory)

This library allows to work with memory-mapped files writing and reading out large volumes of information which will be physically stored in the disk but, if necessary, loaded into operative memory. Files of this type perform reading/ writing operations of much quicker compared to reading/ writing operations of usual file. A memory-mapped file is a segment of virtual memory which is in a direct bite correspondence with a definite part of file or other file-like resource. The main advantage of these files is the increase in the rate of operations of reading / writing, especially, on files of large volumes. It is for this reason that we have chosen this method of data exchange between Hadoop application and MPI application. The main idea of the problem solution is the same as in the method with the use of exchange with usual files but the general rate of the program operation increases.

Static data on coordinates of points and their corresponding values which are stored in each of reducers locally as well as boundary values or initial values and coordinates of points which were distributed to the given Reducer from the Map-stage are read and recorded into Chronicle so that MPI-processes can read out the points of computation region corresponding to their ranks for their parallel processing.

The library Chronicle allows to create a definite indexing of objects, which are contained in a definite copy of the database Chronicle. This property of indexing was used in order that each MPI process can determine, according to the correspondence of its rank and index of the object (a three-dimensional file), a definite copy of the data base Chronicle which was assigned to this process for performing computations.

After the branch process completes its operation it is necessary to read out the result of MPI-processes using the methods of reading given by the library Chronicle and, depending on the fact whether these points are boundary or not, the read out data will be reduced or stored locally on this node.

4 IMPLEMENTATION AND EVALUATION

Numerical solution of problem (1)-(3) was carried out for a particular case with functions:

$$f(x,y,z) = e^{-bt}\sin ax \sin ay \sin az\,\phi(x,y,z) \quad (4)$$

$$\phi(x,y,z) = x^2 + y^2 + z^2 \quad (5)$$

$$\varphi(0,x,y,z) = \sin ax \sin ay \sin az \quad (6)$$

$$a = \pi, b = 3\pi^2$$

For problem (1)-(3) with the pre-determined functions (4)-(6) there is an analytical solution equal to $P(t,x,y,z) = e^{-bt}\sin\pi x \sin\pi y \sin\pi z$ which was used for comparison of the exact and numerical solutions of the problem.

Figures 3-5 present the results of testing of algorithms: a distributed algorithm of the problem of fluid in an oil reservoir based on MapReduce Hadoop technology, a distributed parallel algorithm based on MapReduce and MPI, a distributed parallel algorithm based on MapReduce Hadoop and MPI with memory-mapped files.

To estimate the execution time of three algorithms experimental computations were executed with this number of points:

1) 120×120×120 = 1,728,000;
2) 240×240×240 = 13,824,000;
3) 360×360×360 = 46,656,000;
4) 480×480×480 = 110,592,000.

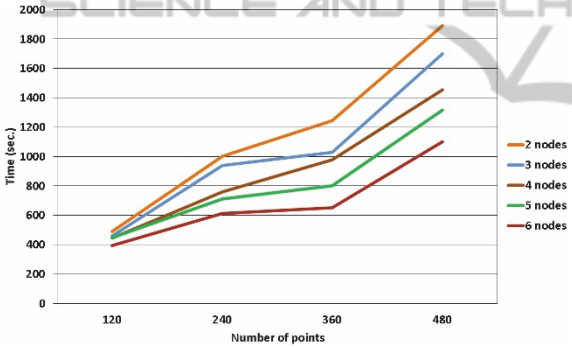

Figure 3: Running time of MR algorithm.

The obtained results demonstrate the achievement of a significant gain of time, when using the library Java Chronicle. In the course of experiments it was noted that in case when the dimensionality of the transferred object (a three-dimensional file) makes up more than 20 million of values of the type with a double floating points the library Chronicle does not allow to transfer the whole array of data in one operation of reading/ writing. To solve this problem, a three-dimensional array of data was divided by the first measuring x into numerous parts so that each part was smaller or equal to the permissible limit of reading/writing in one operation. Consequently, in this case there takes place the change of indexation, i.e. for each MPI-process a range of indexes from which it will read out the data for calculations is formed. Thus, the problem of solving for large dimensionalities of data arrays was solved.

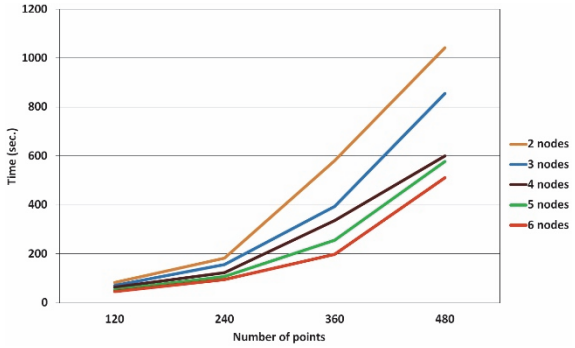

Figure 4: Running time of MR+MPI algorithm.

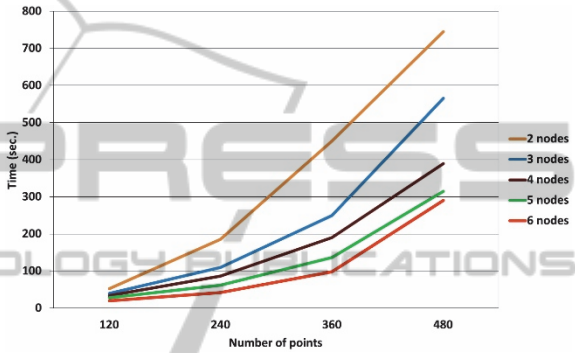

Figure 5: Running time of MR+MPI algorithm with MMF.

5 CONCLUSIONS

Iterative MapReduce MPI oil reservoir simulator developed within the framework of scientific investigations allows to organize distributed parallel computations on heterogeneous systems for solution of oil production tasks. The earlier developed constructive approach of hybrid combination of MapReduce and MPI technologies was used to solve the problem of calculating fluid pressure in an oil reservoir. The novelty of the research includes the use of the library Chronicle with the aim of a more effective realization of reading/writing operations. The comparison of testing results of oil-gas industry problem confirms feasibility of the research, and further actions primarily include adjusting further action in accordance with the actual results.

ACKNOWLEDGEMENTS

Presented research is funded under Kazakhstan government research grant.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

REFERENCES

Biardzki, C., Ludwig, T. 2009. *Analyzing Metadata Performance in Distributed File Systems. In: Malyshkin, V. (ed.)*: PACT 2009. LNCS. Vol. 5698: 8–18.

Bu, Y., Howe, B., Balazinska, M., Ernst, M.D., 2012. *The HaLoop approach to large-scale iterative data analysis*. *VLDB Journal* 21(2): 169–190.

Chen, S. S., Dongarra, J. J., Hsiung, C. C., 1984. Multiprocessing linear algebra algorithms on the CRAY X-MP-2 - experiences with small granularity. *Journal of Parallel and Distributed Computing,* 1(1): 22–31.

Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., and Fox, G., 2010. Twister: a runtime for iterative MapReduce. In: HPDC, pages 810–818. ACM.

Gropp, W., Lusk, E., and Doss, N., 1996. *A high-performance, portable implementation of the MPI message passing interface standard. Parallel Computing 22(6)*: 789-828.

Hoefler, T., Lumsdaine, A., and Dongarra, J., 2009. Towards efficient MapReduce using MPI. In: Ropo, M., Westerholm, J., Dongarra, J. PVM/MPI, *Lecture notes in computer science,* Springer, 5759: 240–249.

http://en.wikipedia.org/Memory-mapped-file.

http://mapreduce.sandia.gov.

Imankulov, T. S., Mukhambetzhanov, S. T., and Ahmed-Zaki, D.Zh. 2013. Simulation of generalized plane fluid filtration in a deformable environment. *Bulletin of the D.Serikbaev East Kazakhstan State Technical University. Computational technologies.* Part 1, № 3, pages 183–191.

Lu, X., Wang, B., Zha, L., and Xu, Z., 2011. Can MPI Benefit Hadoop and MapReduce Applications? In *Proceeding ICPPW '11 Proceedings of the 2011 40th International Conference on Parallel Processing Workshops,* pages 371–379.

Mansurova, M., Akhmed-Zaki, D., Kumalakov, B., and Matkerim, B., 2014. Distributed parallel algorithm for numerical solving of 3D problem of fluid dynamics in anisotropic elastic porous medium using MapReduce and MPI technologies. In *Proceedings of 9th International Joint Conference on Software Technologies,* Vienna, Austria, pages 525–528.

Matkerim, B., Akhmed-Zaki, D., and Barata, M. 2013. Development High Performance Scientific Computing Application Using Model-Driven Architecture. Applied Mathematical Sciences. Vol. 7, №. 100, pages 4961–4974.

Mohamed, H., and Marchand-Maillet S., 2012. Enhancing MapReduce Using MPI and an Optimized Data Exchange Policy In *Proceeding ICPPW '12 Proceedings of the 2012 41st International Conference on Parallel Processing Workshops,* pages 11–18.

Matsunaga, A., Tsugawa, M., Fortes, J. 2008. CloudBLAST: Combining MapReduce and virtualization on distributed resources for bioinformatics applications in eScience'08. In *Proceeding IEEE 4th Int. Conference*. IEEE.

Mohamed, H., and Marchand-Maillet S., 2012. Distributed media indexing based on MPI and MapReduce. Springer, Science+Business Media.

Slawinski, J., and Sunderam, V., 2012.Adapting MPI to MapReduce PaaS Clouds: An Experiment in Cross-Paradigm Execution. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing,* pages 199–203.

Srirama, S. N., Batrashev, O., Jakovits, P., et al. 2011. Scalability of parallel scientific applications on the cloud. Scientific programming, 19(2-3): 91–105.

Sunderam, V. S., Geist, G. A., and Dongarra, J., 1994. The PVM concurrent computing system – evolution, experiences, and trends. Parallel Computing 20(4): 531–545.

Tokarev, M. Y., Tyurin, E. A., and Sinitsyn, M. N. 2012. Supercomputer technologies in science, education and industry. Moscow State University.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS