Distributed Parallel Algorithm for Numerical Solving of 3D Problem of Fluid Dynamics in Anisotropic Elastic Porous Medium Using MapReduce and MPI Technologies

Madina Mansurova, Darkhan Akhmed-Zaki, Matkerim Bazargul and Bolatzhan Kumalakov
Faculty of mechanics and mathematics, al-Farabi Kazakh National University, al-Farabi ave. 71, Almaty, Kazakhstan

Keywords: MapReduce, MPI, Distributed-parallel Computing.

Abstract: Paper presents an advanced iterative MapReduce solution that employs Hadoop and MPI technologies. 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. The named definition is used to justify the proposed solution design.

1 INTRODUCTION

Modern software engineering solutions provide wide range of tools for numeric modelling. They vary from commercial data centre platforms to open source cluster software, downloadable from internet. The aim of this paper is to design a solution that would utilize cluster resources by organizing MapReduce computation with the elements of MPI based parallel programming.

In order to accomplish this task we first identify key features of the existing solutions in this domain in Section 2. Then, in Section 3 we, first, construct an academic example of a numeric model that is used to place the platform into the constraints of a typical iterative algorithm; second, design solution framework; and finally, discuss technological considerations and ways to overcome them. Finally, Section 4 concludes the paper.

2 REVIEW OF HYBRID MAPREDUCE AND MPI SOLUTIONS

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 arises a question of reliability of MPI applications. In recent years the technologies of distributed computing MapReduce is being more widely recognized.

Most of the modern research in this field are directed to the search for new methods of organizations of effective parallel and distributed computing for large-scale problems and resources (Malyshkin, 2010), (Becker and Dagum, 1992), description of their adequate discrete models with the possibility to provide high reliability of the systems being developed. Of no less actuality are the works in the field of research on parallel computing on heterogeneous and hybrid systems, development of the systems for designing parallel programs - frame solutions (skeleton), systems of verification of parallel programming code and corresponding architectural solutions.

On the other hand, the problems of effective exploitation of the existing highly performance resources taking into account their heterogeneity have neither been completely solved (see (Fougère et al., 2005), (Díaz et al., 2012) and (Cappello et al., 2005)). One of the ways of solution is evidently the attraction of technologies for virtualization of computer systems and their integration with technologies of parallel computing. An even more complex problem arises, when considering the problems of organization of reliable systems realizing the distributed (on heterogeneous computing resources) high performance processing of large volume heterogeneous data (Wang

Mansurova M., Akhmed-Zaki D., Bazargul M. and Kumalakov B..
Distributed Parallel Algorithm for Numerical Solving of 3D Problem of Fluid Dynamics in Anisotropic Elastic Porous Medium Using MapReduce and MPI Technologies.
DOI: 10.5220/0005110605240528
In *Proceedings of the 9th International Conference on Software Engineering and Applications* (ICSOFT-EA-2014), pages 524-528
ISBN: 978-989-758-036-9
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

and Liu, 2008), (Pandey and Buyya, 2012).

It is to combination of the advantages of parallel, distributed and cloud computing technologies that the majority of works are devoted (see (Liu and Orban, 2008), (Valilai and Houshmand, 2013) and (Dean and Ghemawat, 2008)) in view of their special practical importance. That work describes a constructive approach of hybrid combination of MapReduce (Fagg and Dongarra, 2000) and MPI (Ng and Han, 1994) for organization of distributed parallel computing on heterogeneous systems. A general description of MapReduce technology is presented in many works (Cohen, 2009), (Jin and Sun, 2013) where the main accent is given to its use for distributed high performance processing of huge volume of data. The main problems in this field are provision of effective load balancing to the existing resources and high reliability of the carried out distributed processing of the data. The main advantage of MapReduce technology is a well-defined division of the roles of computing units - to ”mapper” and ”reducer” the determination of which does not depend on the structure and characteristics of the computing units. Its significant disadvantage is the complexity of organization of communications between units in the process of computation.

Creation of hybrid solutions allows using of the advantages of separate technologies. There exist a great variety of such solutions. 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 and use MapReduce in case of possible failures of the system. The study of primitives of MPI and MapReduce communications allowed the authors (Mohamed and Marchand-Maillet, 2012) to assert the fact that MPI can give the rise in performance of MapReduce applications.

The work (Slawinski and Sunderam, 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.

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 and Ludwig, 2009), (Ekanayake et al., 2010) and (Bu et al., 2012). 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 approaches to provide reliability of such systems.

3 SOLUTION DESIGN AND IMPLEMENTATION

3.1 Mathematical Model and Solution Algorithm

Let us consider a hypercube in anisotropic elastic porous medium $\Omega = [0,T] \times K\{0 \leq x \leq 1, 0 \leq y \leq 1, 0 \leq z \leq 1\}$. Let equation (1) describe the fluid dynamics in hypercube Ω under initial conditions (2) and boundary conditions (3).

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

$$P(0,x,y,z) = \phi(0,x,y,z) \qquad (2)$$

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

In equation (1) the solution function $P(t,x,y,z)$ is the seam pressure in point (x,y,z) at moment t; $\phi(x,y,z)$ is the diffusion coefficient; and $f(x,y,z)$ is density of sources. To solve the defined problem we employed Jacobs numerical method and domain decomposition method from (Malyshkin, 2010).

As a result, we have following solution algorithm: First, original domain is divided into sub-domains. Every sub-domain consists of three main parts: ghost slab, boundary slab and interior slab (see fig. 1).

Data transformations defined by (1)-(3) proceed independently only in the interior slab. Further computation requires boundary slab values that are stored

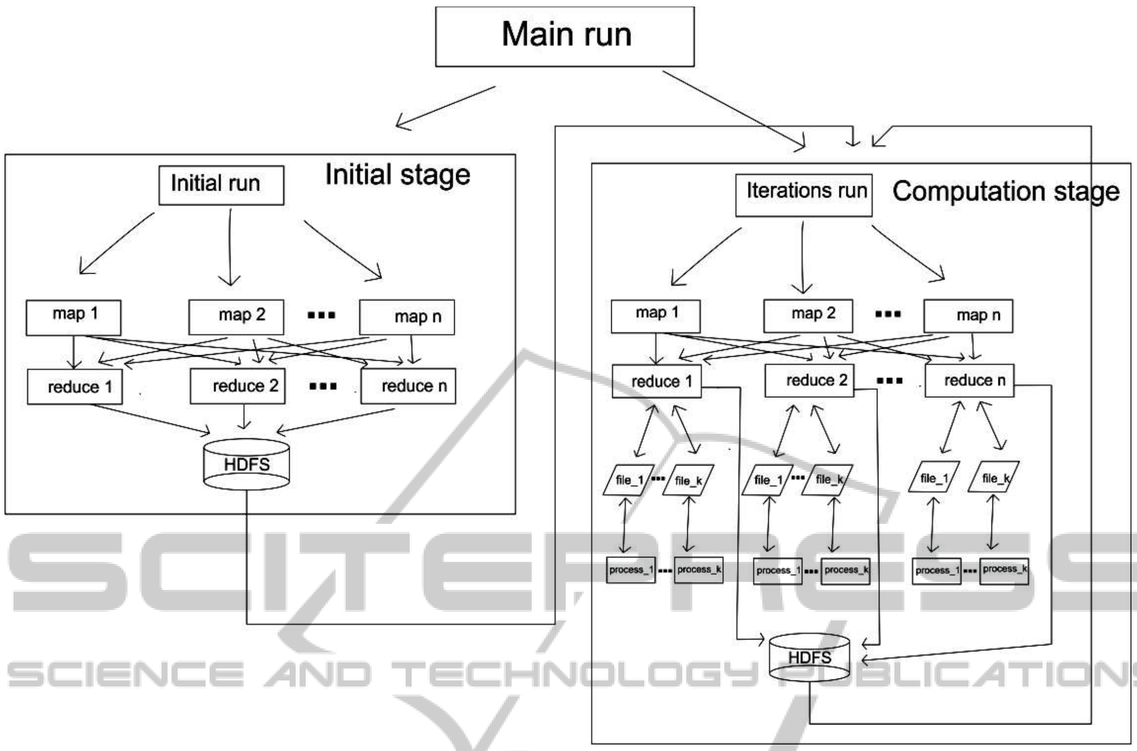

Figure 2: Figure visualizes stages of MapReduce job execution: initialization and computation. At initialization stage mappers divide data into sub-domains and distribute them between reducer nodes, then reducers compute initial values. Computation stage makes use of mappers to collect and re-distribute boundary values of each sub-domain at every iteration step. Reducers restructure the data received from mappers and launch MPI-code to process it.

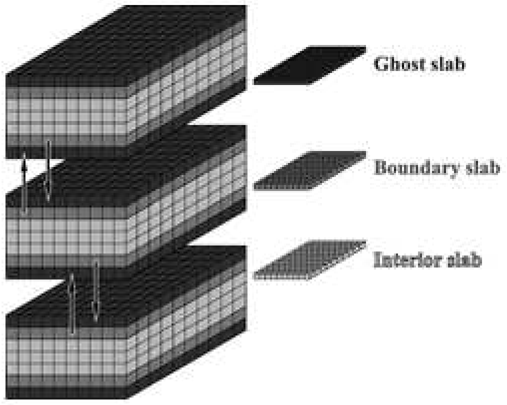

Figure 1: Figure visulizes how general compuing domain is divided into several sub-domains to be distributed between nodes for parallel execution.

in ghost slabs of neighbor sub-domains (i.e.ghost slabs store copies of neighbors boundary slab values). As a result, at each iteration sub-domains exchange their boundary slab values and recalculate their interior values. The iteration is carried out until user set termination conditions are met.

3.2 Technological Considerations and Algorithm Implementation

Review presented in 2 shows that use of MPI technology in MapReduce applications can give significant increase in performance. Nonethless, Jacobs algorithm (being used to solve the problem defined in 3.1) has to be modified to make the best use of proposed technological solution. It consists of several stages.

Initialization Stage. First, we call a MapReduce job that computes initial values for a three-dimensional field. The job, first, applies pre-determine method of field decomposition (set by software designer), then assigns every new sub-field a unique key, and passes them to reducer nodes for parallel execution. When reducers are finished they form an output file that stores string values in the following format: $(D, R, X_{coord}, Y_{coord}, Z_{coord})$. D here is the datum, R is the reducer id, and $X_{coord}, Y_{coord}, Z_{coord}$ represent coordinates on appropriate axes. It may be noted that there is no sub-field id, this is because every reducer is assigned its own sub-field during the first run and it does not change until the end of computation.

Iteration Stage. At this stage, the main iteration loop of problem solution takes place. Unlike at initiation step, mapper here is responsible for boundary values exchange between sub-domains. In particular after each iteration reducers pass mappers their boundary values for re-distribution while keeping their internal slab in memory. In this way we overcome network overloading and reduce the ammount of data transfered.

At the reducer level data obtained from mappers are written in different files depending on x coordinates. Then every MPI-process computes its own subset. Depending on the rank an MPI-process treates the data from the file entitled out + the rank of the process. In other words, data exchange between Hadoop and MPI involves following steps: data are stored locally on each of the nodes; then Hadoop reads from files in which data are stored, and writes into files dedicated for MPI-processes.

When input data for MPI-processes is distributed, platform starts the MPI-program itself. In our case it is developed using MPI-library for Java programming language. Mainly this decision is driven by the fact that Hadoop is written in Java and, moreover, it has a rich set of development and debugging tools.

Unfortunately, it is impossible to directly match the work of MPI library and Hadoop within one project medium. As a result, we call the MPI-program from reducer and it is started as a separated flow. First, MPI-program reads the data from the assigned file and then writes them in a three-dimensional file which will be needed to scale the values in the points form the subfield assigned for this process. The values in a three-dimensional file are converted according to the algorithm, and the new converted values are written in the corresponding file for the given MPI-process for the possibility of the their further processing on the side of reducer.

If the rank of the process is equal to 0 or the number of process -1, the top layer and the bottom additional layer which were distributed for processing remain unchanged. After this, the process of computing is transferred again to Reducer. Then, at stage Reduce, the data on all points computed, at stage MPI are grouped so that statical data are written (recorded) into local file system of the given node on which Reducer is fulfilled, and the boundary values subjected to exchange are reduced for further distribution of these values for the rest nodes which need these boundary values. Both stages of execution are vizualized in fig. 2.

4 CONCLUSIONS

In conclusion, main novelty of the designed solution is the organization of its iterative scheme with the elements of MPI programming. However, presented results lack testing data and, as a result, may raise questions that can not be answered at this stage. Thus, further actions primarily include testing the platform prototype implementation and adjusting further action in accordance with the actual results.

ACKNOWLEDGEMENTS

Research is funded under Kazakhstan government scientific grant "Developing models and applications of MapReduce-Hadoop based high-performance distributed data processing for oil extraction problems".

REFERENCES

Becker, J. C. and Dagum, L. (1992). Particle simulation on heterogeneous distributed supercomputers. In *HPDC*, pages 133–140.

Biardzki, C. and Ludwig, T. (2009). Analyzing metadata performance in distributed file systems. In Malyshkin, V., editor, *Parallel Computing Technologies (10th PaCT'09)*, volume 5698 of *Lecture Notes in Computer Science (LNCS)*, pages 8–18. Springer-Verlag (New York), Novosibirsk, Russia.

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

Cappello, F., Djilali, S., Fedak, G., Hérault, T., Magniette, F., Néri, V., and Lodygensky, O. (2005). Computing on large-scale distributed systems: Xtremweb architecture, programming models, security, tests and convergence with grid. *Future Generation Comp. Syst*, 21(3):417–437.

Chen, S. S., Dongarra, J. J., and 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.

Cohen, J. (2009). Graph twiddling in a mapreduce world. *Computing in Science and Engineering*, 11(4):29–41.

Dean, J. and Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *CACM*, 51(1):107–113.

Díaz, J., Muñoz-Caro, C., and Niño, A. (2012). A survey of parallel programming models and tools in the multi and many-core era. *IEEE Trans. Parallel Distrib. Syst*, 23(8):1369–1386.

Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., and Fox, G. (2010). Twister: a runtime for iterative mapreduce. In Hariri, S. and Keahey, K., editors, *HPDC*, pages 810–818. ACM.

Fagg, G. E. and Dongarra, J. (2000). FT-MPI: Fault tolerant MPI, supporting dynamic applications in a dynamic world. In Dongarra, J., Kacsuk, P., and Podhorszki, N., editors, *PVM/MPI*, volume 1908 of *Lecture Notes in Computer Science*, pages 346–353. Springer.

Fougère, D., Gorodnichev, M., Malyshkin, N., Malyshkin, V. E., Merkulov, A. I., and Roux, B. (2005). Numgrid middleware: MPI support for computational grids. In Malyshkin, V. E., editor, *PaCT*, volume 3606 of *Lecture Notes in Computer Science*, pages 313–320. Springer.

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

Jin, H. and Sun, X.-H. (2013). Performance comparison under failures of MPI and mapreduce: An analytical approach. *Future Generation Comp. Syst*, 29(7):1808–1815.

Liu, H. and Orban, D. (2008). Gridbatch: Cloud computing for large-scale data-intensive batch applications. In *CCGRID*, pages 295–305. IEEE Computer Society.

Lu, X., Wang, B., Zha, L., and Xu, Z. (2011). Can MPI benefit hadoop and mapreduce applications? In Sheu, J.-P. and Wang, C.-L., editors, *ICCP Workshops*, pages 371–379. IEEE.

Malyshkin, V. (2010). *Assembling of Parallel Programs for Large Scale Numerical Modelling*. IGI Global, Chicago, USA.

Matsunaga, A. M., Tsugawa, M. O., and Fortes, J. A. B. (2008). CloudBLAST: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. In *eScience*, pages 222–229. IEEE Computer Society.

Mohamed, H. and Marchand-Maillet, S. (2012). Enhancing mapreduce using MPI and an optimized data exchange policy. In *ICPP Workshops*, pages 11–18. IEEE Computer Society.

Ng, R. T. and Han, J. (1994). Efficient and effective clustering methods for spatial data mining. Technical Report TR-94-13, Department of Computer Science, University of British Columbia. Tue, 22 Jul 1997 22:21:44 GMT.

Pandey, S. and Buyya, R. (2012). Scheduling workflow applications based on multi-source parallel data retrieval in distributed computing networks. *Comput. J*, 55(11):1288–1308.

Slawinski, J. and Sunderam, V. S. (2012). Adapting MPI to mapreduce paaS clouds: An experiment in cross-paradigm execution. In *UCC*, pages 199–203. IEEE.

Srirama, S. N., Batrashev, O., Jakovits, P., and Vainikko, E. (2011). Scalability of parallel scientific applications on the cloud. *Scientific Programming*, 19(2-3):91–105.

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

Valilai, O. and Houshmand, M. (2013). A collaborative and integrated platform to support distributed manufacturing system using a service-oriented approach based on cloud computing paradigm. *Robotics and computer-integrated manufacturing*, 1(29):110–127.

Wang, J. and Liu, Z. (2008). Parallel data mining optimal algorithm of virtual cluster. In Ma, J., Yin, Y., Yu, J., and Zhou, S., editors, *FSKD (5)*, pages 358–362. IEEE Computer Society.