Applying MDA to Generate Hadoop Based Scientific Computing Applications

Darkhan Akhmed-Zaki, Madina Mansurova, Bazargul Matkerim, Ekateryna Dadykina, and Bolatzhan Kumalakov

Faculty of mechanics and mathematics, al-Farabi Kazakh National University
al-Farabi 71, Almaty, Republic of Kazakhstan
mansurova01@mail.ru

Abstract. The paper presents an attempt to develop and deploy a functioning MDA (Model-Driven Architecture) model of a distributed scientific application. The main focus is a problem of modeling high performance computing processes in a visual notation and automatic generation of an executable code using the resulting diagrams. The article describes the efforts to create a platform independent model of process execution, transformation it into a platform specific model and, finally, automatic generation an application code. The research novelty includes a platform independent model of the classic hydrodynamics problem, equivalent Hadoop based platform specific model and the testing results that confirm feasibility of the research.

Keywords: Model-Driven Architecture, Hadoop, Scientific Computing.

1 Introduction

Model Driven Development (MDD) is a software engineering methodology that treats "formal specification of the function, structure and actions of the system" as the main object of development [1]. Other objects, such as program codes, are generated at a later stage from these specifications also called models. Models are usually developed using standard or extended Unified modeling language (UML) diagrams, which make one more level of abstraction in the object-oriented design paradigm. In this case, the increased level of abstraction facilitates development of platform independent application templates that are easily converted to an executable code for any platform. In other words, one may build a template application (set of diagrams) that may be used by an automated code generator to produce an executable code for any known software platform.

In the early 2000s, OMG consortium [2] defined the conceptual infrastructure of Model driven architecture (MDA) that serves as the basis for MDD methods and defines standard specification, model description and transformation languages. Figure 1 presents a generic MDA development cycle.

First, user requirements are processed and formalized in a form of Platform-independent-model (PIM). Despite being presented as a set of UML diagrams, PIM has to take into account how the automated interpreter would further

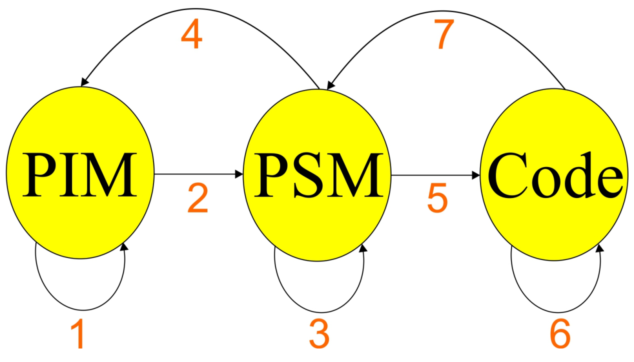

Fig. 1. An MDA application development cycle.

transform the model with the final goal of building a working solution. There is rich discussion on MDD modeling techniques in academic literature, which is out of the scope of this article. In this section, let us point out that an initial prototype of the model may contain inaccuracies or discrepancies, therefore, it goes through repetitive validation and introduction of changes before it proceeds to the second stage.

Platform-specific-model (PSM) is the same system specification as PIM, but with the elements of the target platform. For instance, if the system has to run under particular environment (target environment) such elements would include environment components, interfaces, data storage, etc. Thus, every time when it is necessary to adapt the system to new technology only PSM should be reconsidered.

Code stage in the figure represents automated executable generation. It involves special software that interprets PSM and produces final output. Such tools are available in different varieties and their properties vary depending on vendor and employment proposals. Academic literature provides some brief introduction to such tools, but this discussion is out of the scope of this article.

At this stage it should be noted that MDA and MDD have been successfully applied in several domains of software engineering including high performance computing (HPC). This article presents an attempt to construct and evaluate all levels of HPC MDA models, define their transformation rules and generate an executable code.

The remaining part of the article is structured as follows: Section 2.2 presents an extensive literature review and puts the article results on the body of knowledge landscape. Section 3 provides a detailed description of the solution including the main components overview (Subsection 3.1), PIM (Subsection 3.2) and PSM (Subsection 3.3) descriptions. Next, Section 4 describes the experiment design and evaluation results. Finally, the article is completed by a relevant discussion and future research directions.

2 Research Background and Related Work

2.1 HPC Model Development Process Overview

Figure 2 vizualizes the process of designing and implementing HPC application using UML 2.0 activity diagram. The process is divided into 5 stages. Each stage is performed by a specialist indicated on the left and its result is the corresponding MDA model shown on the right.

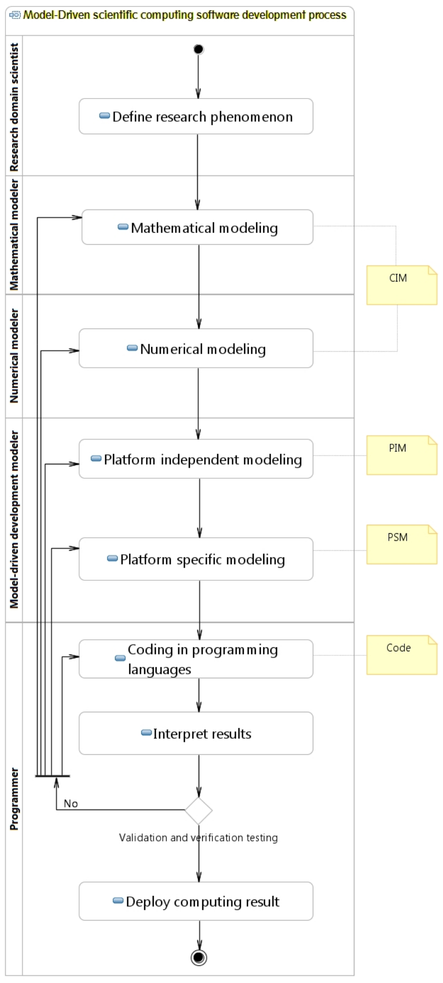
Model-Driven scientific computing software development process
Research domain scientist
Define research phenomenon
Mathematical modeler
Mathematical modeling
CIM
Numerical modeler
Numerical modeling
Model-driven development modeler
Platform independent modeling
PIM
Platform specific modeling
PSM
Programmer
Coding in programming languages
Code
Interpret results
No
Validation and verification testing
Deploy computing result

Fig. 2. MDD process of HPSC software.

HPC application design and development is carried out by a group of specialists as presented in Figure 2. A specialist in oil-gas industry defines the problem statement. Next, a mathematician creates a mathematical model that is passed on to a specialist in the field of the numerical methods. He/she finds a corresponding explicit or implicit numerical method and defines an application algorithm. Next, a software designer constructs HPC application architecture, and, finally, a software engineer who implements the solution.

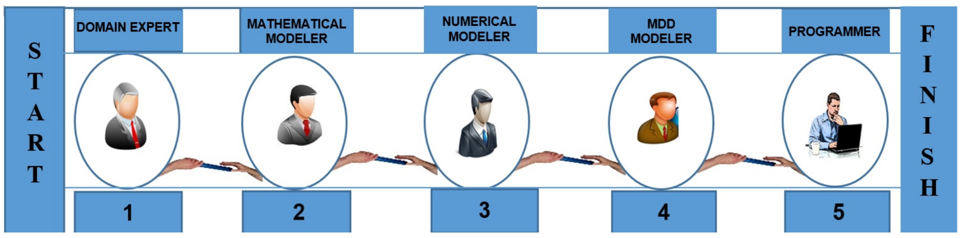

Fig. 3. Relay race of specialists in HPSC application development.

Throughout the complex application development errors may appear at different stages. Causes include ambiguity in interpretation of models, complexity of a program code, upgrade from one parallel architecture to another, modification of software, documentation updates, etc. Investigations show that MDD gives good results, when developing applications for scientific HPC.

2.2 Related Work Review

In order to develop scientific computing applications using MDD methodologies, several contributions have been proposed. The paper of Lugato [3] is one of the early works that proposed MDE for high performance computing applications. Later in [4], he presented the idea in detail. The papers of [5]-[9] proposed MDE approach by creating a domain specific modeling language. M. Palyart et al. [7] proposed an approach called MDE4HPC using their own domain specific modeling language - High Performance Computing Modeling Language to describe abstract views of the software. Implementation of the approach using the tool ArchiMDE is integrated with Paprika studio and Arcane framework. Palyart et al. [6] introduced a DSVL to help in specifying and modeling HPC applications. They focus on specification of solution parallelism. However, they did not show how their language could be used to generate HPC code. Brual et al. [8] reported the needs for leveraging on knowledge and expertise by focusing on Domain-Specific Modeling Languages (DSML) application. Similarly, M. Almorsy et al. [10] proposed their prototype of scientific Domain Specific Visual Languages (DSVLs)-based toolset. However, they did not consider the distributed scientific computing. In [11], the authors developed an approach to interoperation of high performance, scientific computing applications based upon math-oriented data modeling principles. In [12], the authors define the architecture framework consisting of a coherent set of viewpoints to support the mapping of parallel algorithms to parallel computing platforms. The feature of the approach [12] is the particular focus on optimization at the design level using architecture viewpoints. This approach can be adopted for different parallel algorithms and can be used with different parallel technologies. Gamatie et al. [13] represent the Graphical Array Specification for Parallel and Distributed Computing (GASPARD) framework for massively parallel embedded systems on Multi-Processor System-on-Chips (MPSoCs) architectures to support optimization of the usage of hardware resources. GASPARD uses MARTE standard profile for modeling embedded systems at a high abstraction level. Based on the Model-Driven Engineering (MDE) paradigm, MARTE models are refined towards lower abstraction levels, this making automatic generation of the code possible. Danniluk [14] presented the problem of Molecular Beam Epitaxy and Reflection High-Energy Electron Diffraction with MDA approach. In the paper, he described a practical and pragmatic approach to MDA that had been used during the work at three scientific projects. The PIMs are described with UML and PSMs that specify implementation of PIMs with Object Pascal and C++. They applied MDD and visual-development tools to numerical simulation problems. Each of these projects has its own research interests and none of them considers Hadoop distributed computing platform as a specific platform. Similarly to the authors of this work, the works of [15], [16] proposed MDD approach in developing MapReduce applications. In [15], the authors apply Map/Reduce to EMF-based models to cope with complex data structures in the familiar an easy-to-use and type-safe EMF fashion. They store large EMF models in Hadoop's HBase and then use those models from within the Map/Reduce programming model using EMF's generated APIs. The authors of [16] developed an MDE-based cloud deployment framework that automates the deployment and execution of MapReduce applications. The model-driven approach is used to predict the performance of MapReduce application in the cloud environment. The features of our approach are: 1) We create scientific computing components for modeling scientific computing applications. Application modeling is achieved by using UML diagrams. 2) We presented the whole cycle of MDA process of development from modeling to code generation. 3) Our approach is oriented to the MapReduce application development for one of oil extraction problems. 4) The presented work is the continuation of earlier works [17], [18].

3 Solution Design

This section introduces generic components that form the building blocks of scientific HPC applications. Then it proceeds to present how template PIM is built and then transformed to PSM for Hadoop code generation.

3.1 Main Components of the System

In order to design models, we implement four generic components that will serve as basic building blocks for scientific HPC applications. The developed components are divided and named depending on the peculiarities of HPC applications.

At this stage, we assume that a scientific computing problem is of no importance for the solution, because the model can be constructed from these components in any case (for discussion on components functions see [19]).

For any application one has to determine input and output parameters, class of equations, explicit or inexplicit methods of computation and instruments (tools) for performing parallel computations. Therefore, we presented these 4 invariable independent parts in the form of 4 basic components. They are: an input-output component InOutPut, a component of equations of numerical methods NumerMethods and a component of organization of a high performance computing environment PEOrganize. Every component consists of several classes. Description of the components - as applied to MapReduce application - is presented below:

1. Component PEOrganize is used for creation of the topology on Hadoop platform.
2. Component NumerMethods is used for determination of a numerical model with different types of grid and numerical methods.
3. Component SciEquations is used for determination of a mathematical model with the number of final differential equations and conditions for these equations.
4. Component InOutPut consists of classes of reading from the file and writing into the file with the help of which input and output data of HPSC application are prescribed (set).

As is shown in Figure 4, in the process of designing and development of applications there takes place transformation of models starting from the upper level to the lower level.

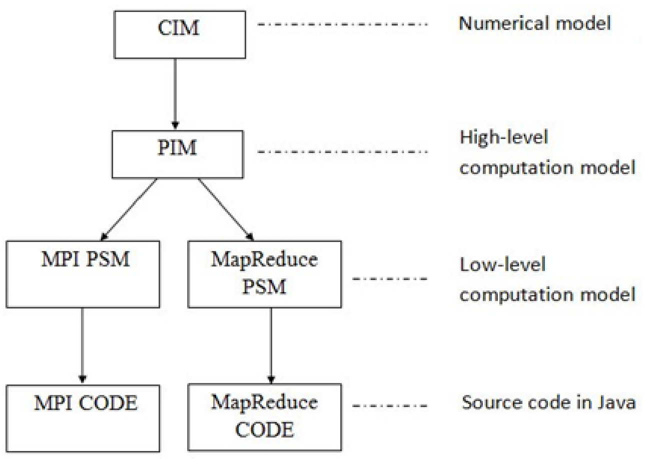

Fig. 4. MDA modeling.

As it was mentioned above, MDD specialist receivers a computationally independent model CIM from the specialist in numerical methods. In case of solution of the problem, CIM contains the algorithm of a numerical solution of the problem by the explicit method. In his turn, MDD-specialist creates an independent on the platform and the programming language PIM model for the given numerical CIM model using HPSC components. Model CIM can be described by the components of input-output-InOutPut, the component of equation of a scientific computing problem SciEquations, the component of numerical methods NumerMethods.

But in CIM model there is no information for the component of organization of high performance computational environment PEOrganize as the environment of development in the computational models is not considered.

The work resulted in the development of the MDA model and realization using the Hadoop technology.

3.2 PIM Model

In our case, computations are performed in MapReduce Hadoop environment. The algorithm with the use of MapReduce consists of the stage of initialization and iteration stage, a separate MapReduce work being fulfilled at each iteration.

Computations are performed on Hadoop platform, a MapReduce problem receives a cube of data, Mappers perform 1D decomposition, each Reducer receives its block of data and performs computations. After computations are completed, boundary data are entered into a distributed file system HDFS, the values of inside points are written into a local file system. Then a new cycle begins. The process continues until the condition is satisfied.

Thus, we have developed a PIM model for HPSC applications for the problem with the help of UML diagram of classes (Figure 5) indicating relations between the classes. On account of retrieving calling methods of each other, the classes of components are in associative relations.

3.3 PSM Model

Models of transformation of PIM to PSM can be classified by several categories: improving the quality of transformation, with perfection of the development, with refining, with specialization, translation, abstraction, generalization and forms of designing. In our case, transition from PIM model to MapReduce Java PSM model refers to the category with refining. Refining means redetermination in the course of transition from CIM to PIM, transition from PIM to PSM. Refining can be added at one level of abstraction. Transformation of PIM model to MapReduce Java PSM model is transformation of UML-diagram of classes Java to the diagram of classes Java with addition of MapReduce specification to PSM. When transforming PIM model to MapReduce Java PSM model, multiple succession, associations of classes and qualified associations must be removed. In PSM model, the relations between components shown in Figure 6 are preserved, but specification of the programming language Java is added. The Hadoop PSM

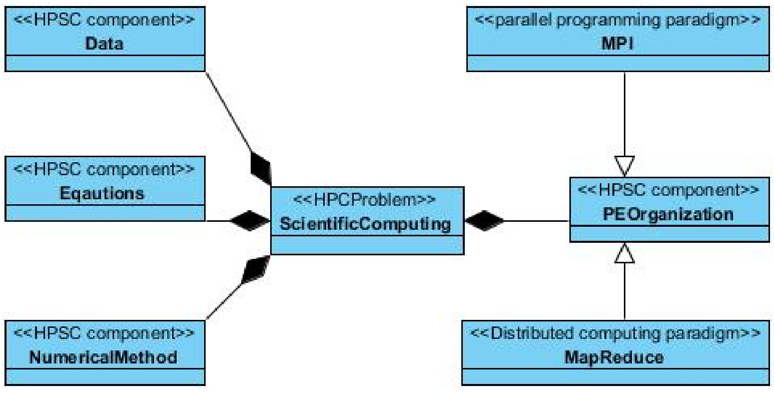
<<HPSC component>> Data
<<parallel programming paradigm>> MPI
<<HPSC component>> Eqautions
<<HPCProblem>> ScientificComputing
<<HPSC component>> PEOrganization
<<HPSC component>> NumericalMethod
<<Distributed computing paradigm>> MapReduce

Fig. 5. PIM model.

model in Figure 6 shows specification of the HPSC component - PEOrganization. The MapReduce distributed programming paradigm which consists of initialization and iteration stages of computation is modeled with the help of Map class and Reduce class.

4 Experiment Design and Evaluation

4.1 Hydrodynamics Problem Definition

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:

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

$$P(0,x,y,z) = \varphi(0,x,y,z). \tag{2}$$

$$\frac{\partial P}{\partial n}|_{\Gamma} = 0. \tag{3}$$

Here, (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; *ϕ(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. In order to implement a test solution, we employ a mathematical problem for a particular case with functions from [17].

First, the original domain is divided into sub-domains (Figure 7). Every sub-domain consists of three main parts: ghost slab, boundary slab and interior slab.

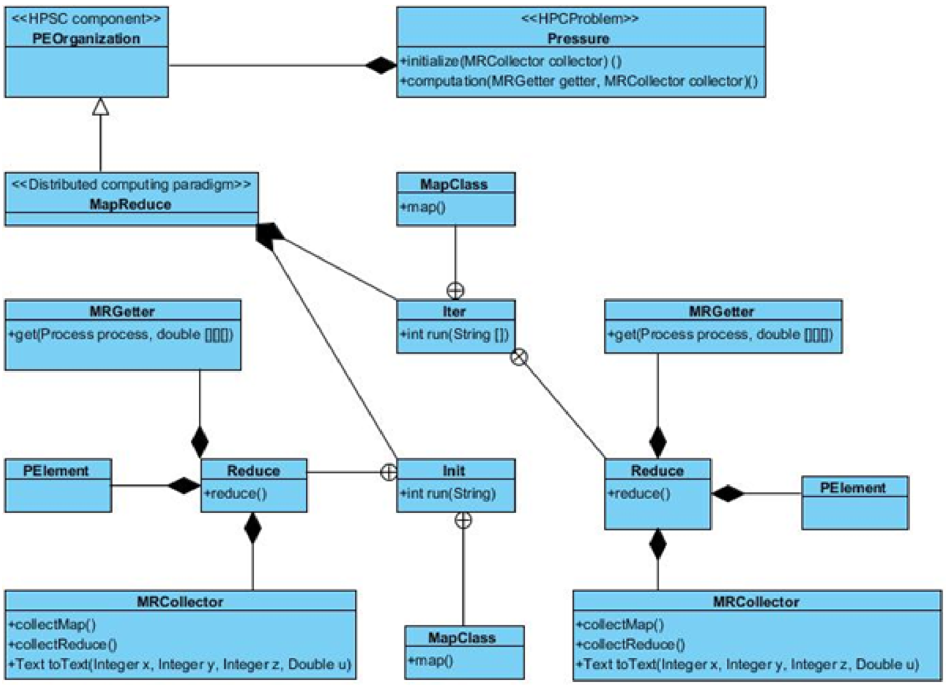
<<HPSC component>>
PEOrganization
<<HPCProblem>>
Pressure
+initialize(MRCollector collector) ()
+computation(MRGetter getter, MRCollector collector)()
<<Distributed computing paradigm>>
MapReduce
MapClass
+map()
MRGetter
+get(Process process, double [][][])
Iter
+int run(String [])
MRGetter
+get(Process process, double [][][])
PElement
Reduce
+reduce()
Init
+int run(String)
Reduce
+reduce()
PElement
MRCollector
+collectMap()
+collectReduce()
+Text toText(Integer x, Integer y, Integer z, Double u)
MapClass
+map()
MRCollector
+collectMap()
+collectReduce()
+Text toText(Integer x, Integer y, Integer z, Double u)

Fig. 6. Hadoop based PSM model.

Data transformations (defined by the algorithm) can proceed independently only in the interior slab. The boundary slab of the sub-domain, when being computed, requires boundary slab values of its neighbors and those are stored in the ghost slab. In other words, the ghost slab stores copies of neighbors boundary slab values.

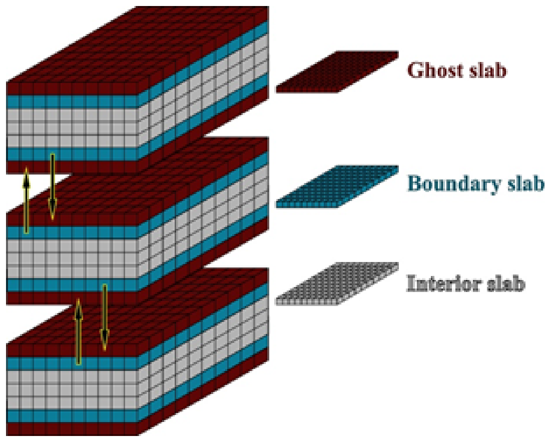

Fig. 7. The hypercube is divided into a number of computation sub-domains. Every sub-domain is assigned as a reducer.

The algorithm of numerical solution of the problem 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 of 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 enters 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 flow of data corresponding to the description is presented in Figure 8.

The distributed algorithm consists of two stages:

1. The stage of initialization;

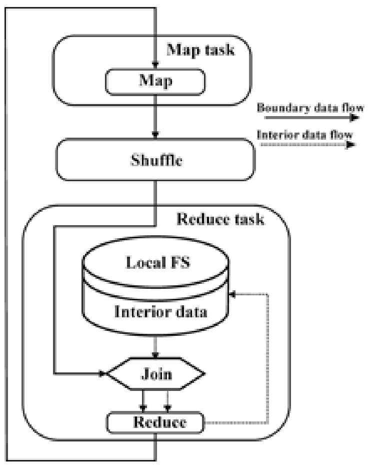

Fig. 8. Iterative MapReduce framework scheme.

2. 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.

4.2 Computational Experiment Results

An automatic transition from MapReduce Java PSM to Java code is realized with the help of generator Acceleo. Acceleo is a pragmatic realization of Object Management Group (OMG) of MOF model. Acceleo UML2 for Java is a code generator based on Acceleo 3.2. This generator supports creation of the initial code Java for classes and interfaces.

After automatic generation we have a program code which contains description of classes and methods as well as relations between methods corresponding to PSM model (Figure 6). Then, the methods of each class of HPSC components are written down according to their functionality or called from class libraries. According to Figure 2 the process of code writing and interpretation of the results goes on till the program gives the results corresponding to the results of a sequential program. The computing experiments on the generated MapReduce program and the sequential program must be performed at equal pre-determined parameters of the computational problem.

We have performed all the stages of the process of MapReduce application development for problem (1)-(3) according to MDD methodology and have obtained the same computing results for the problem solution of the sequential program code and generated MapReduce code. The experimental design and the results of the generated MapReduce program execution on a special deployed Apache Hadoop Mini-Cluster of Laboratory of Computer Science of al-Farabi Kazakh National University are presented below.

Apache Hadoop 2.6.0 Mini-Cluster consists of 1 master node and 7 slaves. All slave nodes have Ubuntu 14.04 on board, master node has Ubuntu Server 14.04. Master node hardware characteristics: Hardware: HP ProLiant-BL460c-Gen8, Architecture: x86-64, CPU(s): 4, Model name: Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz. Slaves hardware characteristics: Architecture: x86-64, CPU(s): 4, Model name: Intel(R) Core(TM) i5-2500 CPU @ 3.30GHz. NFS server is configured on master node. Slaves have the same folder mounted with read/write access rights. Nodes are connected using Ethernet devices, this providing up to 1000 Mbps, Intel(R) PRO/1000 Network Connection. 3D problem sizes are chosen by the possibility of the memory of computing nodes: $256 \times 128 \times 128$, $1024 \times 128 \times 128$, $4096 \times 128 \times 128$, $8192 \times 128 \times 128$.

The results obtained for the dependence of the speedup and efficiency are presented in Figure 9 and 10. We can see that only the problem size of $8192 \times 128 \times 128$ has got the four times of speedup, it means that, according to the features of Hadoop platform, with the increase of MapReduce application data size the speedup increases. Figure 10 also shows that when we choose a big problem size the computing nodes are more effectively used.

To test the fault tolerance appearance of IOException with the probability of 33 % is added to the Reducer code. In the experiment, 3 jobs have taken part each of which initiates 8 Reducers. The number of broken Reducers is 25 out of 97. The average time of the problem performance without IOException and with IOException was the same. Thus, it can be concluded that the time of performance does not depend on failures of Reducers and their restarting. The computations obtained in both cases have equal values.

5 Conclusion and Further Research

The aim of this direction of investigations is the use of MDD methods for development of applications for HPC in the field of oil and gas production. For the work, MDA standard is chosen as MDD methodology. The process of designing and developing a high performance application is described on the example of MDA modeling. The method of passing on a baton between different specialists of oil and gas industry, the close interaction of which can facilitate the work on creation of complex applications for oil and gas industry, is shown. The investigation results show the prospects of using MDD methodology for solution of complex resource intensive problems.

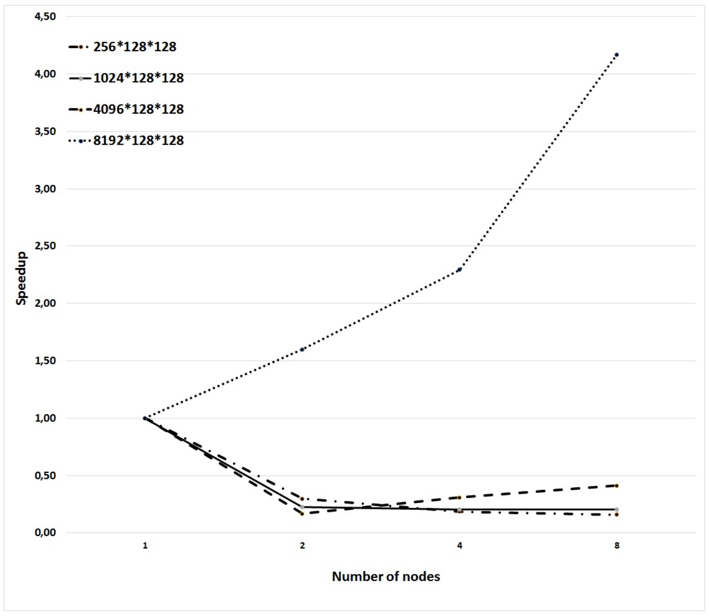

Fig. 9. The speedup versus the number of nodes for different meshes.

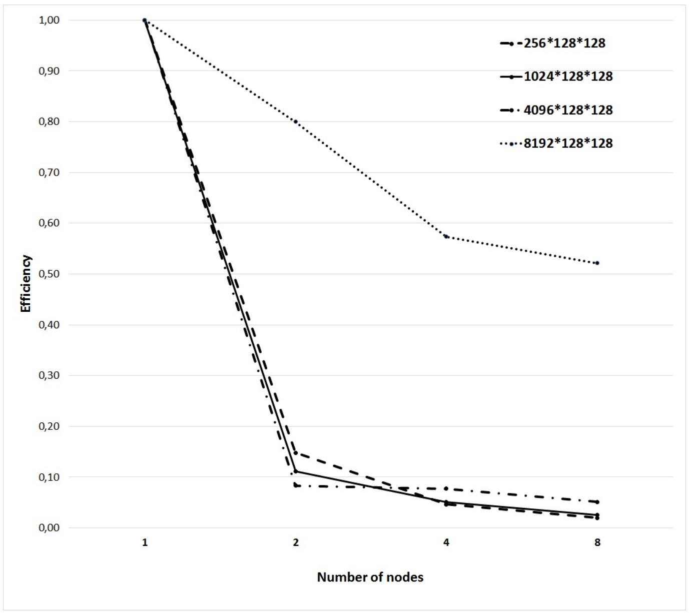

Fig. 10. The efficiency versus the number of nodes for different meshes.

The experimental results allow to conclude that the distributed application works well and with the increase in the volume of the data being processed the performance of Hadoop implementation increases. HPSC applications can be designed and developed with the help of the proposed MDA model and its basic components. This approach will possible become one of the ways to perform distributed scientific computing on high performance heterogeneous systems.

Acknowledgments. The presented research was funded under Kazakhstan government research grant "Development of models and applications for high performance distributed processing based on MapReduce-Hadoop technology for oil extraction problems".

References

1. Frankel, D.S.: Model Driven Architecture: Applying MDA to Enterprise Computing. Wiley, New York (2003)
2. OMG. Unified Modeling Language, Version 2.2. Superstructure (2009)
3. Lugato, D.: Model-driven engineering for high-performance computing applications. In: Proceedings of the 19th IASTED International Conference on Modeling and Simulations, Quebec City, Quebec, Canada (May 2008)
4. Lugato, D., Bruel, J.M., Ober, I., Venelle, B.: Model-driven Engineering for High-Performance Computing Applications, Modeling Simulation and Optimization - Focus on Applications, Shkelzen Cakaj (Ed.), (2010)
5. Palyart, M., Lugato, D., Ober, I., Bruel, J.M. MDE4HPC: An Approach for Using Model-Driven Engineering in High-Performance Computing. SDL 2011: Integrating System and Software Modeling. Lecture Notes in Computer Science. Volume 7083, pp. 247-261. (2012)
6. Palyart, M., Lugato, D., Ober, I., Bruel, J.M. HPCML: A Modeling Language Dedicated to High-Performance Scientific Computing. In: Proceedings MDHPCL '12 Proceedings of the 1st International Workshop on Model-Driven Engineering for High Performance and CLoud computing. Article No. 6. (2012)
7. Palyart, M., Lugato, D., Ober, I., Bruel, J.M.: Improving Scalability and Maintenance of Software for High-Performance Scientific Computing by Combining MDE and Frameworks. Model Driven Engineering Languages and Systems. Lecture Notes in Computer Science Volume 6981, pp. 213-227. (2011)
8. Bruel, J.M., Combemale, B., Ober, I., Raynal., H.: MDE in Practice for Computational Science. ICCS 2015: 660-669. (2015)
9. Arkin, E., Tekinerdogan. B.: Domain Specific Language for Deployment of Parallel Applications on Parallel Computing Platforms. In: Proceeding ECSAW '14. Article No. 16. (2014)
10. Almorsy., M., Grundy, J., Sadus, R.J., van Straten, W., Barnes, D.G., Kaluza., O.: A suite of domain-specific visual languages for scientific software application modelling. VL/HCC 2013: 91-94. (2013)
11. Miller, M.C., Reus, J.F., Matzke, R.P., Arrighi, W.J., Schoof, L.A., Hitt, R.T., Espen, P.K.: Enabling Interoperation of High Performance, Scientific Computing Applications: Modeling Scientific Data with the Sets and Fields (SAF) Modeling System. International Conference on Computational Science (2) 2001: 158-170. (2001)

12. Tekinerdogan, B., Arkin. E.: Architecture Framework for Mapping Parallel Algorithms to Parallel Computing Platforms. In: Proceedings MDHPCL '13 Proceedings of the 2nd International Workshop on Model-Driven Engineering for High Performance and CLoud computing. pp. 53-63. (2013)
13. Gamatie, A., Le Beux, S., Piel, E., Ben Atitallah, R., Etien, A., Marquet, Ph., Dekeyser, J.-L.: A Model-Driven Design Framework for Massively Parallel Embedded Systems. ACM Trans. Embedded Comput. Syst. 10(4): 39. (2011)
14. Daniluk, A.: Visual modeling for scientific software architecture design. A practical approach. Computer Physics Communications. 183(2012): 213. (2012)
15. Scheidgen, M., Zubow., A.: Map/reduce on EMF models In: Proceedings MDHPCL '12 Proceedings of the 1st International Workshop on Model-Driven Engineering for High Performance and CLoud computing. Article No. 7. (2012)
16. Shekhar, Sh., Caglar, F., An, K., Kuroda, T., Gokhale, A., Gokhale, Sh.: A Model-driven Approach for Price/Performance Tradeoffs in Cloud-based MapReduce Application Deployment. In: Proceedings MDHPCL '13 Proceedings of the 2nd International Workshop on Model-Driven Engineering for High Performance and CLoud computing. pp. 37-43. (2013)
17. Mansurova, M., Akhmed-Zaki, D., Matkerim, B., 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. In: Proceed-ings of 9th International Joint Conference on Software Technologies ICSOFT 2014, pp. 525-528. Vienna, Austria (2014)
18. Matkerim B., Akhmed-Zaki D., Barata M.: Development High Performance Scientific Computing Application Using Model-Driven Architecture, Applied Mathematical Sciences, Vol. 7, no. 100. pp. 4961-4974. (2013)
19. Bezivin,J.: Object to Model Paradigm Change with the OMG/MDA Initiative, presentation of Summer School on MDA for Embedded System Development, pp. 16-20. Leon, France (2002)