Dynamic Cloud Resources Allocation on Multidomain/Multiphysics Problems

Niki Sfika, Aigli Korfiati, Christos Alexakos, Spiros Likothanassis
Department of Computer Engineering and Informatics,
University of Patras,
Patras, Greece
{sfika, korfiati, alexakos, likothan}@ceid.upatras.gr

Konstantis Daloukas, Panagiota Tsompanopoulou
Department of Electrical and Computer Engineering,
University of Thessaly,
Volos, Greece
{kodalouk, yota}@inf.uth.gr

***Abstract*— The solution of multidomain/multiphysics problems is a computationally and memory demanding process, especially for large-scale differential equations. In this paper, we propose a cloud application that provides users a solution environment for multiphysics/multidomain problems utilizing cloud technologies that manage pre-existing hardware, network, operating system and applications. Particularly, according to the problem and its computational demands, the user can have the results from any place and any device without any other concern. The user sets the problem's parameters, chooses the solution method that fits better to the specific problem and finally gets the problem solution. The application dynamically allocates the minimum possible resources automatically in the background without the user's interference.**

Keywords*— *cloud computing, multidomain/multiphysics problems, dynamic resources allocation, problem solving environment

I. INTRODUCTION

Cloud Computing is a new delivery model of computing services over the Internet. Cloud technologies can provide reliable solutions on complex and big data analysis problems [1]. In the area of processing data from real-life complex physical systems, such as gas turbine engine, air pollution or underwater acoustics, computationally demanding differential equation solving methods are required [2]. Utilization of cloud computing technologies for fast and reliable mathematical processing of continuously derived data from such physical systems can conclude to integrated systems that extract valuable information in each case.

Cloud computing is a new paradigm in the area of ICT. It refers to the consumption of all available computational resources (such as networks, applications, servers, storage and other computational resources) as a utility which can be accessed via web by anyone. What discriminates cloud computing from existing techniques is a series of characteristics, such as on demand provision of network access to the resources pool and flexible and adaptable service management by the cloud provider. In cloud computing, the user can increase or decrease the capacity and consume as many resources as he actually needs, depending on the computational requirements of a specific problem. Concerning adaptability, new applications and features can be deployed significantly faster than with traditional models. In reference to the resource benefits, information is not hosted on individual computers but on the cloud, so it can be accessed remotely as long as there is Internet connection and users and systems can collaborate simultaneously on this information [3].

From the software developers' point of view, cloud computing can be considered as an extension of distributed models of software development. More precisely, the software is composed by interconnected third party components, multiple partners (users or systems) can have the ownership, and the execution can take place in multiple computers in a distributed environment. Virtualization is a technology that enables cloud computing. It simulates network resources, storage devices, operating systems and hardware platforms. Actually, operating systems and applications are hardware-independent and virtual machines can be offered to any system.

The simulation and modeling of complex physical systems, such as the dissemination of primary brain tumors (gliomas) and the saltwater intrusion into freshwater aquifers due to overpumping, usually involve many components [4], [5]. A physical system in the real world normally consists of a large number of components that have different shapes, obey different physical laws and design constraints, and interact through geometric and physical interfaces. Mathematically, each component is modeled by a partial differential system with various formulations for the geometry, the partial differential equation (PDE) and the interface/boundary conditions.

The approaches for modeling and simulating complex physical systems can be divided to (1) domain decomposition, (2) Schwarz splitting and (3) interface relaxation techniques. The interface relaxation (IR) methods' main advantage is that they treat a multidomain and multiphysics (different PDE

The present research work has been partially supported by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program Education and Lifelong Learning of the National Strategic Reference Framework (NSRF) - Research Funding Program: THALIS. Investing in knowledge society through the European Social Fund (MIS 379416).

978-1-4673-8103-1/15 $31.00 © 2015 IEEE
DOI 10.1109/FiCloud.2015.59

IEEE computer society

operators are applied on different subdomains) problem as a loosely coupled system of subproblems consisting of much simpler (concerning both the geometry and the differential operator) PDE problems. More specifically, in the IR methods the PDE domain is decomposed into subdomains, derived by the physics or for parallelization purposes while initial guesses are set on the interfaces between the subdomains. The subproblems are solved independently and new values on the interfaces are computed by particular IR methods (forcing the correct conditions for the problem) iteratively until convergence is succeeded [2], [6], [7]. The inherent parallelism in IR methods makes them ideal for mapping onto parallel architectures and they are anticipated to perform really well when combined with cloud technologies.

Our proposed cloud application is a multiphysics/ multidomain problem solving environment implementing the Interface Relaxation methodology. The user provides a problem as input, chooses the IR method that fits better the specific problem and has access to the solution in a close-to-optimal execution time, from any place and any device without any other concern. At the same time, the application allocates the minimum possible resources automatically in the background without user's interference.

The proposed approach is presented in detail in the rest of the paper. In Section II some significant technologies and related work in the area of cloud computing and interface relaxation methods are presented. Section III describes the implementation of the proposed application and Section IV contains the results of the experiments. Finally, Section V concludes the paper.

II. Background and related work

A. Cloud Computing

Cloud Computing is an emerging model that provides access to all types of computational resources (consuming them as a utility) to end-users according to their demands. Cloud computing has such a tremendous growth due to the centralized storage, memory, processing and bandwidth which conclude to a more efficient cost model [8].

Cloud Computing is divided into three delivery models: (i) Infrastructure-as-a-Service (IaaS), where on demand virtual server machines can be offered with their own storage and network, (ii) Platform-as-a-Service (PaaS), where storage, operating system, hardware and network can be offered to the user's existing or new applications and (iii) Software-as-a-Service (SaaS) where an application can be offered as a service on demand. SaaS provides the user both the hardware and the appropriate software through a front-end portal.

Lately, a significant number of approaches that utilize cloud computing technologies for scientific data analysis have been proposed by both industry and academia. Specifically, in the area of scientific computing many approaches have been introduced for the solution of scientific problems [9], [10]. A benchmark evaluation has showed that cloud computing can provide the means for significant improvement of the performance of scientific analysis problems [11]. SciCloud [12] is an important step on how an application that takes advantage of cloud computing can be beneficial for solving computationally intensive scientific, mathematical and academic problems.

Cloud computing technologies have been used in many approaches that require dynamic allocation of resources. In most of these cases, the requirements refer to high capacity of memory and CPU. Cloud's scalability facilitates such applications in many use cases of scientific data analysis, permitting among others, the execution of modern high-performance parallel algorithms [13]. As an example, in bioinformatics, cloud-based solutions are anticipated to permit high performance analysis of large datasets of biological and genetic data [14]. In medicine, there are paradigms of tools which employ cloud technologies for delivering demanding medical image analysis results to the end-users directly to their working devices [15], [16]. Also, the analysis of geographic data requires a large number of resources, and cloud-based approaches are used leading to a significant performance improvement [17], [18].

The proposed application aims to a software tool delivered according to the SaaS model elaborating the scalability capability provided by the IaaS model. Thus, the software platform Cloudstack has been used for delivering IaaS services. CloudStack is open source software which provides reliable, flexible and scalable cloud orchestration [19]. It also comes with a management server for hypervisor hosts and, as a result, it manages the resource pool, the network and the storage as an IaaS cloud platform. In the proposed application the computational resources (memory, processor and network) are assigned to each problem according to its requirements.

B. Interface Relaxation methodology

Interface Relaxation methods serve for the efficient solution of multidomain PDEs through an iterative procedure [2], [16], [17]. Consider the composite differential problem defined by

$$Lu = f \; in \; \Omega \backslash \partial\Omega, \;\; u = u^b \; on \; \partial\Omega \tag{1}$$

where u^b is a prescribed function on the boundary $\partial\Omega$, $\Omega \equiv \bigcup_{i=1}^{p} \overline{\Omega_i}$ and $\Omega_i, i = 1, \dots, p$ are open sets such that, $\bigcap_{i=1}^{p} \overline{\Omega_i} = \emptyset$ while L is the differential operator which might be different in each subdomain Ω_i. With the IR methodology, the above problem can be replaced with the following loosely coupled system of differential problems.

$$\begin{aligned} &L_i u_i = f_i \; in \; \Omega_i \\ &G_{ij} u = 0 \; on \; \left(\partial\Omega_i \cap \partial\Omega_j\right) \backslash \partial\Omega, \;\; \forall j \neq i \\ &u = u_i^b \; on \; \partial\Omega_i \cap \partial\Omega \end{aligned} \tag{2}$$

where L_i, f_i and u_i^b, for $i = 1, \dots, p$ are the restrictions of L, f and u^b respectively on each subdomain Ω_i and G_{ij} is a condition on the interface between subdomains Ω_i and Ω_j which enforces proper coupling. This coupling is responsible for preserving the physical properties of the original problem (i.e., continuity, smoothness or jumping). The differential

operators and the coupling can be of any kind. However, this study is focused, but not limited, to the most common case of second order elliptic differential equations with smooth global solution. Thus, continuity of the solution and its first (normal) derivative should be imposed on the interfaces. As we can observe, the solution of (1) through (2) requires solution of each subdomain problem and combination of the computed solutions on the interfaces.

Before presenting the considered IR methods, let's introduce (only for explaining purposes) a type of geometry that clarifies the two methods in an easier way. Consider that the global domain Ω consists of subdomains which are adjacent rectangles along the x-axis. Then the interfaces are vertical lines and each subdomain has one common boundary, i.e., interface with its neighboring one or two (left and/or right) subdomain(s). GEO [7] is the first interface relaxation method implemented in our system and it calculates the values of the solution on the interfaces between different subdomains while it guarantees fast convergence. The new relaxed solution on an interface is obtained by adding to the old one a geometrically weighted average of the normal boundary derivatives of the adjacent subdomains. Specifically, the solution at iteration $k+1$, $u^{(k+1)}$, is given from the following equations:

$$u^{(k+1)} = u^{(k)} - \rho\left(\frac{\partial u_L^{(k)}}{\partial n} - \frac{\partial u^{(k)}}{\partial n}\right), on\ left\ interface \quad (3)$$

$$u^{(k+1)} = u^{(k)} - \rho\left(\frac{\partial u^{(k)}}{\partial n} - \frac{\partial u_R^{(k)}}{\partial n}\right), on\ right\ interface \quad (4)$$

$k = 0,1,2, \ldots$

where $u^{(k)}$ and $\frac{\partial u^{(k)}}{\partial n}$ is the computed solution and its normal derivative on the interface, while $\frac{\partial u_L^{(k)}}{\partial n}, -\frac{\partial u_R^{(k)}}{\partial n}$ are the values of the outward normal derivatives on the interface from the two adjacent subdomains, all in the previous iteration. ρ is a relaxation parameter used to accelerate convergence.

ROB [6] is another interface relaxation method based on Robin interface conditions to transmit information across boundaries. One solves the local PDE on the subdomains using Robin conditions on the interface lines by matching a combination of Dirichlet and Neumann data from neighboring subdomains. Specifically, the solution at iteration $k+1$, $u^{(k+1)}$, has to obey (5) on the subdomain's left interface and (6) on the right.

$$-\frac{\partial u^{(k+1)}}{\partial x} + \lambda u^{(k+1)} = -\frac{\partial u_L{}^{(k)}}{\partial x} + \lambda u_L{}^{(k)} \quad (5)$$

$$\frac{\partial u^{(k+1)}}{\partial x} + \lambda u^{(k+1)} = \frac{\partial u_R{}^{(k)}}{\partial x} + \lambda u_R{}^{(k)} \quad (6)$$

where $u_L^{(k)}$ and $\frac{\partial u_L^{(k)}}{\partial x}$ is the solution and its derivative on the left interface from the left adjacent subdomain, while $u_R^{(k)}$ and $\frac{\partial u_R^{(k)}}{\partial x}$ is the solution and its derivative on the right interface from the right adjacent subdomain. λ is a relaxation parameter to accelerate convergence.

According to our knowledge, a small number of implementations of IR methods can be found in the literature. They use Matlab [7], [20]-[22], the Agents computing paradigm over a network of heterogeneous workstations and PELLPACK [6], [21], [23], the BOND agent middleware and PELLPACK [24] and the Grasshopper agent middleware and FORTRAN and C [4], [5]. These PSEs highly depend on the agent platforms and PELLPACK [25]. A rather new implementation free of such constraints is presented in [26]. It uses the FEniCS [27] software for the solvers and RabbitMQ [28] for the necessary communication.

III. PROPOSED CLOUD IMPLEMENTATION

A. System Architecture

The proposed application is based on the management of the allocation of the cloud resources in order to create groups of virtual machines for parallel processing. Furthermore, it provides the appropriate interfaces to end-users and systems for accessing the platform.

The major functional components of the architecture are the following:

1. The **Graphical GUI** where each user has the ability to register for a new account or log in to an existing one. When logged in, users can define new problems by inserting the appropriate input and choosing the desired IR method for the problem solution. During the solution process, they can review their problems' solution status and details. Furthermore, third party systems can access the services provided by the application through a simple HTTP API that is loosely based in the exchange of JSON messages.
2. The **Job Schedule Module** which is responsible to orchestrate problem execution on the cloud infrastructure. Its duties consist of a) the resources allocation needed by the Virtual Solver Nodes for the problem execution, b) the deployment of the eligible VMs along with their information, c) the initialization of the problems and d) the deletion of the VMs after the execution is finished.
3. The **Job Monitor Module** is an Advanced Message Queue Server (AMQS) that handles the communication between the entities of the system. AMQS is based on the Advanced Message Queuing Protocol (AMQP) which connects systems and manages the information and messages exchange between them. A more extensive interpretation is that with AMQP programs and systems can produce and send messages, while other programs and systems can receive them and process them. In the present work RabbitMQ has been used. RabbitMQ is a messaging broker based on the AMQP and offering a common platform to send and receive messages while these messages stay safe until they reach their final destination.
4. The **DataStore Module** can be accessed from all the system components and stores the input data, the results and the intermediate data of the users' problems.

5. The **Virtual Solving Nodes** are created by the Job Schedule Module in order to start the problem execution. During the execution, they inform the Job Monitor Module about the state of the problem and once they finish the execution, they send their final results back to the Job Monitor Module. After job completion they are deleted.

Fig. 1 depicts an overview of the proposed system's architecture with all the system's modules and their interactions, where dotted lines present communication between modules. The AMQS infrastructure is utilized for the communication between the various modules. AMQS-based communication is used in the following interaction scenarios: a) inside Virtual Solving Nodes for the interface values and b) between the Virtual Solving Nodes and the Job Monitor Module for progress monitoring.

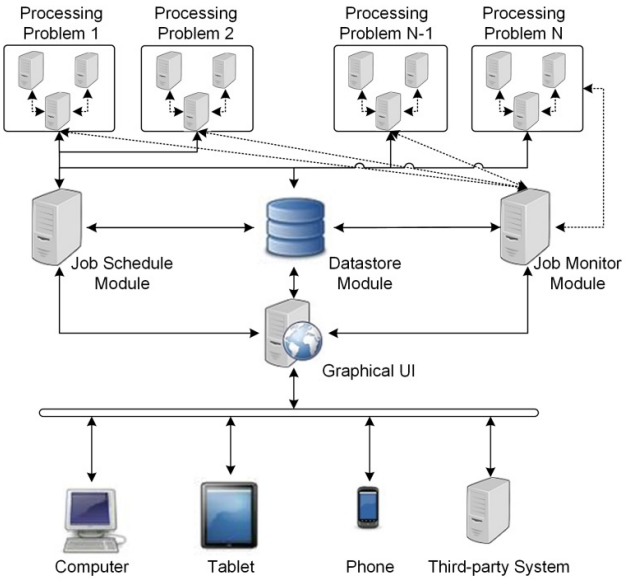

Fig. 1. System architecture

B. Managing process execution

During runtime, the Job Schedule Module and the Job Monitor Module are responsible for the resources allocation and the process orchestration. Their processes are depicted in Fig. 2 and Fig. 3, respectively. Both of the aforementioned systems are running in the same VM in Cloudstack and they perform a number of tasks.

At the beginning of its operation, the Job Schedule Module checks the DataStore in order to obtain the required information about the job queue. If the job queue is empty, it remains idle until a new job comes in the queue. In case there are more than one pending jobs, it selects the first job in chronological order from the queue and initiates the process for the problem solving. The first task in this process is the calculation of the appropriate computational resources (VMs for Virtual Solving Nodes) required for optimal performance depending on the size of the input data and the available resources in the cloud infrastructure. After this calculation, the creation of the VMs that will act as the Virtual Solving Nodes follows. These VMs are created based on pre-existing templates including the software required for the problem execution. When the problem has been executed, the Job Schedule Module destroys the created VMs and moves to the execution of a new job in the queue.

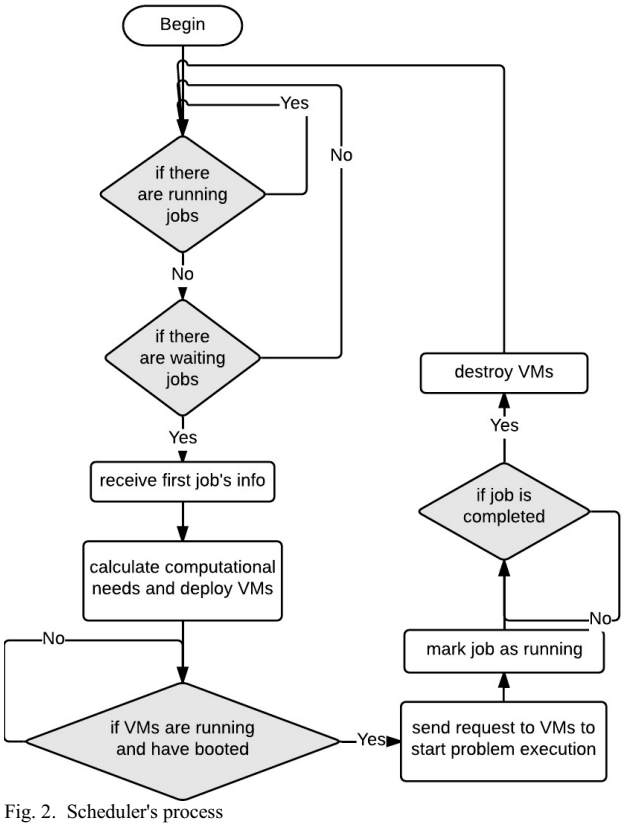

Fig. 2. Scheduler's process

After the Job Schedule Module initiates a job, the Virtual Solving Nodes execution progress is monitored by the Job Monitor Module. Through its connection with RabbitMQ, the Job Monitor Module receives the current iteration of the problem being solved and sends this information to the DataStore as well for publishing to the users/systems. When the job is finished, the Job Schedule Module marks the problem as completed and stores the solution in the DataStore. It remains idle until a new problem starts being solved.

C. Application Interfaces

The proposed platform is empowered with two interfaces in order to interoperate with the rest ecosystem. The first is an HTML graphical interface used by users for submitting processing requests, evaluating their progress and accessing results. A screen with a user's submitted problems is presented in Fig. 4. The second refers to third-party systems and it is an API that can be invoked by HTTP requests over Internet.

The graphical user interface is simple and follows responsive design guidelines in order to meet the desired level of user friendliness. The responsive design is a collection of techniques applied in the HTML code in order for the

application's screens to automatically be adapted to the screen resolution of the user's device, even in small devices such as tablets and smart phones. Thus, the proposed application can be easily used from anywhere with the condition that users have a mobile device with web browser and internet connection. Fig. 4 depicts a screenshot from the application where user can see the progress of the submitted jobs.

Third-party systems can invoke the HTTP-based API as web service. The API is consisted of three major methods:

- **setNewJob/{user-id}/{job-parameters}** which submits a new problem to the application's queue. The response is informing the invoker for the success or failure of the job submission.
- **getUserJobs/{user-id}** which returns the list of user's submitted problems and their progress.
- **getJobDetails/{user-id}/{job-id}** which returns the description of the submitted problem (input) and information about the progress. If the job is completed, a link to the final results is also provided.

For security reasons, additional parameters are sent by each request. These parameters contain the authentication credentials of the specific user account for the API and hash code for the evaluation of the integrity of the request.

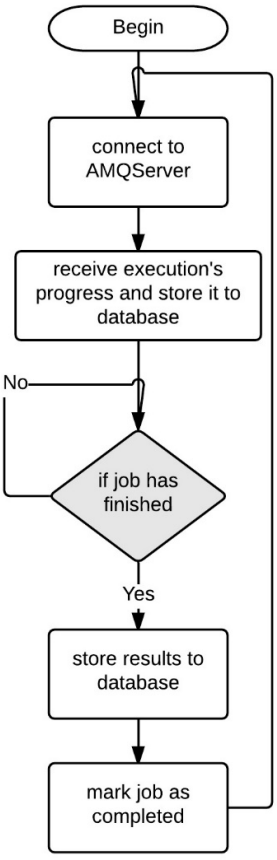

Fig. 3. Monitor's process

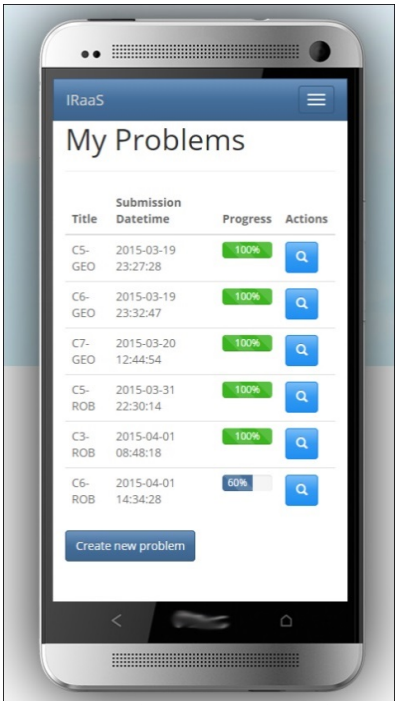

Fig. 4. Application's screen with user's submitted problems

D. Interface Relaxation implementation

The main characteristic of the interface relaxation techniques is the abundant level of parallelism in the solution process. In order to take advantage of such inherent parallelism, we assign to each Virtual Solving Node the solution of one subdomain and the subsequent update of the values in the interface points. Considering the geometry discussed in Section II.B for a 3 domains case, Fig. 5 describes the interaction of the solving nodes per iteration. Specifically, Virtual Solving Node A handles the left subdomain and the left-middle interface. Thus, we can achieve full utilization of the computational resources found in a cloud infrastructure.

Referring again to Fig. 5, the user defines the input parameters through a form in the graphical interface. A Python script in each VM generates the subdomain's mesh (triangular elements), applies the boundary conditions and the initial guesses on the interfaces and defines the variational formulation of the PDE problem. Then, the local solution and its gradient are computed. Their values on the interfaces points are sent to the VMs that handle the adjacent subdomains. These VMs compute the new relaxed interface point values based on selected IR method (GEO or ROB), which serve as input for the subsequent solution of the corresponding subdomains. A new iteration begins once the VMs that handle neighboring subdomains have finished the communication step regarding the computed solution and gradients.

An additional benefit of the proposed scheme is the minimization of the communication overhead. The updated

interface values are computed on one of its adjacent subdomains' node and therefore the communication is reduced by half.

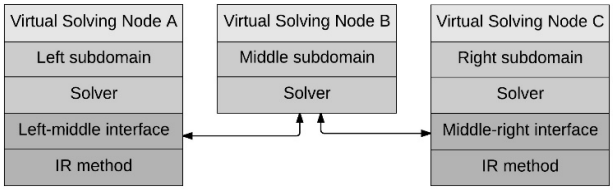

Fig. 5. Interface relaxation methodology: solving process and topology

IV. Experimental Results

To evaluate the performance of the proposed methodology, we employed the following elliptic PDE problem:

$$Lu \equiv -\nabla^2 u(x,y) + \gamma^2 u(x,y) = f(x,y), \ \ (x,y) \in \Omega \qquad (7)$$
$$u(x,y) = u^b(x,y), \ \ (x,y) \in \partial\Omega$$

with $f(x,y)$ and $u^b(x,y)$ selected such that the true solution is:

$$u(x,y) = e^{y(x+4)} x(x-1)(x-0.7)y(y-0.5) \qquad (8)$$

The problem consists of three subdomains. The interface points are at $x_1 = \frac{1}{3}$ and $x_2 = \frac{2}{3}$ while $\gamma^2 = 2$. Seven grid sizes are examined, according to seven different values of the discretization parameter h, which is considered equal in both x and y direction. The number of interface points in each case increases from 6 to 321 points. The left subdomain is approximately four times larger than the middle, while the right one is approximately two times larger than the middle subdomain. The interfaces have the same number of points and therefore are of equal workload. The considered problems along with details, such as their discretization step and the grid sizes for the left, middle and right subdomains are described in Table I.

TABLE I. Considered Problems

Problem	Problems' Details				
	Discretization Parameter	*Left Domain Size*	*Middle Domain Size*	*Right Domain Size*	*Number of Interfaces Points*
P1	0.1	84	24	44	6
P2	0.05	328	88	168	11
P3	0.025	1134	294	574	21
P4	0.0125	4508	1148	2268	41
P5	0.00625	17655	4455	8855	81
P6	0.003125	69228	17388	34668	161
P7	0.0015625	274134	68694	137174	321

For the evaluation of the application, we performed a number of experiments for each test case. Experiments were executed with a variety of given computational resources. According to the results, in order to balance the tradeoff between the minimum execution time and resources allocation, it was decided that the best resource allocation depends exclusively on the input datasets. Thus, the proposed application was parameterized in order to adopt the following rule in its decision for the resources allocation for each subdomain:

if subdomain size <= 75000:
virtual solving node RAM = 1GB
virtual solving node processor = 1core, 2GHz
else if 75000 < subdomain size <= 150000:
virtual solving node RAM = 2GB
virtual solving node processor = 1core, 2GHz
else if subdomain size > 150000:
virtual solving node RAM = 4GB
virtual solving node processor = 1core, 2GHz

Table II presents the execution time in the proposed application with both GEO and ROB IR methods implemented for the test cases of Table I, using the aforementioned rule.

For comparison purposes, we also present the respective execution times when the same implementations were executed in 3 different predefined virtual machines, as presented for the GEO method in [26]. The virtual machines used for the experiments have 4 virtual cores and 2GB RAM and are running in a XenServer virtualization environment installed on a server with 2 x Intel(R) Xeon(R) CPU E5-2620, 2.00GHz.

TABLE II. Execution Times of ROB and GEO

Problem	Execution Times	
	ROB	*GEO*
P1	15.348	15.041
P2	15.373	15.411
P3	16.022	15.724
P4	19.223	19.169
P5	27.313	28.548
P6	65.386	71.416
P7	265.415	297.122

TABLE III. Comparison Analysis to Virtual Implementations

Problem	Comparison Analysis			
	Virtual Implementation ROB	*Proposed Cloud Implementation ROB*	*Virtual Implementation GEO*	*Proposed Cloud Implementation GEO*
P1	16.004	15.348	15.495	15.041
P2	16.281	15.373	16.068	15.411
P3	17.392	16.022	18.140	15.724
P4	23.634	19.223	25.595	19.169
P5	45.456	27.313	51.808	28.548
P6	132.179	65.386	166.633	71.416
P7	1099.05	265.415	735.731	297.122

As we can observe from Table III, the proposed implementation provides a significant reduction in execution time compared to the previous virtual implementation. The reduction factor increases further with the increase of the problem size, which is a testament of the efficiency of the proposed methodology for the solution of large-scale problems. In addition, the application provides an important

gain in resource utilization cost for every test case compared to the virtual implementation with the predefined VMs. As a result, both time and resource usage reduction has been achieved with the proposed cloud application.

V. CONCLUSION

We have presented a cloud-driven application for the solution of multidomain/multiphysics problems based on the Interface Relaxation methodology. In this way a large problem is split in smaller sub-problems, which are solved independently exchanging information about their solution iteratively until the initial large problem is solved. Through the proposed application the users can define complex multiphysics problems, select the appropriate PDE solvers for the smaller sub-problems and IR method (ROB or GEO) for the interfaces and get the computed solution of the global problem.

The resources' allocation is dynamic and is based on the computational needs of each problem. The main benefits when using the proposed application are that it allocates the minimum possible resources, solves the problem in a close to minimum execution time, and the resources allocation is performed automatically in the background without the user's interference. In the near future, we plan to implement a larger number of PDE solvers and IR methods, as well as to provide support for more complex geometries. These modifications will enable us experiment with model problems representing the dissemination of primary brain tumors (gliomas) and the saltwater intrusion into freshwater aquifers due to overpumping.

REFERENCES

[1] Agrawal, D., Das, S., & El Abbadi, A, "Big data and cloud computing: current state and future opportunities", In Proceedings of the 14th International Conference on Extending Database Technology, pp. 530-533, ACM (2011)

[2] Tsompanopoulou, P., "Collaborative PDEs: theory and practice", Ph.D. thesis, Mathematics Department, University of Crete, Greece (2000)

[3] George C. Kagadis, Christos Kloukinas, Kevin Moore, Jim Philbin, Panagiotis Papadimitroulas, Christos Alexakos, Paul G. Nagy, Dimitris Visvikis, William R. Hendee, "Cloud computing in medical imaging", Vision 20/20 paper, Medical Physics, Vol. 40, No. 7, AAPM, 070901 (2013)

[4] Markus, S., Houstis, E., Catlin, A., Rice, J., Tsompanopoulou, P., Vavalis, E., Gottfried, D., Su K., Balakrishnan, G., "An Agent-Based Netcentric Framework for Multidisciplinary Problem Solving Environments", International Journal of Computational Engineering Science, 1, 33–60 (2000)

[5] Houstis, E.N., Catlin, A.C., Tsompanopoulou, P., Gottfried, D., Balakrishnan, G., Su, K., Rice, J.R., "GASTURBNLAB: A Multidisciplinary Problem Solving Environment for Gas Turbine Engine Design on a Network of Non-Homogeneous Machines", J. of Comp. and Applied Mathematics, 149(1), 83-100 (2002)

[6] Tsompanopoulou, P., Vavalis, E., "An Experimental Study of Interface Relaxation Methods for Composite Elliptic Differential Equations", Applied Mathematical Modelling, 32 1620–1641 (2008)

[7] Tsompanopoulou, P., Vavalis, E., "Analysis of an interface relaxation method for composite elliptic differential equations", Journal of Computational and Applied Mathematics 226 2, 370– 387 (2009)

[8] Iosup, A., Ostermann, S., Yigitbasi, M. N., Prodan, R., Fahringer, T., & Epema, D. H., "Performance analysis of cloud computing services for many-tasks scientific computing", Parallel and Distributed Systems, IEEE Transactions on, 22(6), 931-945 (2011)

[9] Wang, L., Tao, J., Kunze, M., Castellanos, A. C., Kramer, D., & Karl, W., "Scientific Cloud Computing: Early Definition and Experience", In HPCC Vol. 8, pp. 825-830 (2008)

[10] Wang, L., Kunze, M., Tao, J., & von Laszewski, G., "Towards building a cloud for scientific applications", Advances in Engineering software, 42(9), 714-722 (2011)

[11] Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., & Epema, D., "A performance analysis of EC2 cloud computing services for scientific computing", In Cloud computing, pp. 115-131, Springer Berlin Heidelberg (2010)

[12] Srirama, S., Batrashev, O., & Vainikko, E., "SciCloud: scientific computing on the cloud", In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 579-580, IEEE Computer Society (2010)

[13] Srirama, S. N., Batrashev, O., Jakovits, P., & Vainikko, E., "Scalability of parallel scientific applications on the cloud", Scientific Programming, 19(2-3), 91-105 (2011)

[14] Andreas, Holzinger, Matthias Dehmer, and Igor Jurisica, "Knowledge discovery and interactive data mining in bioinformatics-state-of-the-art, future challenges and research directions", BMC bioinformatics 15.Suppl 6 (2014): I1.

[15] Rittner, Leticia, et al., "Web-based platform for collaborative medical imaging research", SPIE Medical Imaging. International Society for Optics and Photonics (2015)

[16] G. Kagadis, C. Alexakos, P. Papadimitroulas, N. Papanikolaou, V. Megalooikonomou, D. Karnabatidis, "Cloud Computing Application for Brain Tumor Detection", European Congress of Radiology – ECR 2015, European Society of Radiology (ESR), Vienna, March 4–8 (2015)

[17] Wang, L., Kunze, M., Tao, J., & von Laszewski, G., "Towards building a cloud for scientific applications", Advances in Engineering software, 42(9), 714-722 (2011)

[18] van Lew, Baldur, et al., "Interactive analysis of geographically distributed population imaging data collections over light-path data networks", SPIE Medical Imaging. International Society for Optics and Photonics, (2015)

[19] Cloudstack, 2015. Available: http://cloudstack.apache.org

[20] Rice, J. R., Tsompanopoulou, P., Vavalis, E., "Interface relaxation methods for elliptic differential equations", Applied Numerical Mathematics 32 2, 219–245 (2000)

[21] Rice, J.R., Tsompanopoulou, P. Vavalis, E.A., "Fine Tunning Interface Relaxation Methods for Elliptic Differential Equations", Applied Numerical Mathematics, 43(4), 459–481 (2002)

[22] Chalkias, C., "Implementation of a Distributed System for the Solution of MultiDomain / MultiPhysics Problems", Diploma Thesis, Dep. of Electrical and Computer Eng., Univ. of Thessaly, (2013)

[23] Drashansky T., "An Agent-Based Approach to Building Multidisciplinary Problem Solving Environments", PhD Thesis, Purdue University, Computer Science Department, (1996)

[24] Bölöni, L., Marinescu, D.C., Rice, J.R., Tsompanopoulou, P., Vavalis, E.A., "Agent Based Scientific Simulation and Modelling", Concurancy: Practice and Experience, 12, 845–861 (2000)

[25] Houstis, E. N., Rice, J. R., Weerawarana, S., Catlin, A. C., Papachiou, P., Wang, K.-Y., Gaitatzes, M., "PELLPACK: A Problem Solving Environment for PDE Based Applications on Multicomputer Platforms", ACM Transactions on Mathematical Software, 24, 30-73, (1998)

[26] Korfiati A., Tsompanopoulou P. and Likothanassis S., "Serial and Parallel Implementation of an Interface Relaxation Method", in Proceedings of the 6th International Conference on Numerical Analysis, pp 167-173 (2014)

[27] Logg , A., Mardal, K. A., Wells, G. N., et al., "Automated Solution of Differential Equations by the Finite Element Method", Springer, 2012

[28] Rabbitmq, 2015. Available: http://www.rabbitmq.com/documentation.html