IRaaS: A Cloud Implementation of an Interface Relaxation Method for the Solution of PDEs

Aigli Korfiati, Niki Sfika, Konstantis Daloukas, Christos Alexakos, Panagiota Tsompanopoulou and Spiros Likothanassis

***Abstract*—The solution of composite Partial Differential Equations is an indispensable step in numerous scientific applications. However, this is a computationally and memory demanding process for large-scale differential equations. This is especially true for multidomain/multiphysics problems, which require application of an interface relaxation (IR) methodology on the common boundaries between domains. In this paper, we present IRaaS, a cloud-based environment for the solution of multidomain/multiphysics problems. IRaaS efficiently exploits the inherent parallelism found in the solution step for the individual subdomains, thus significantly reducing computational and memory requirements. At the same time, its efficient allocation and management mechanism allocates the optimal number of resources (virtual machines), based on the total number of resources available, as well as the size of the problems for solution.**

***Index Terms*—cloud computing applications, interface relaxation, multidomain/multiphysics problems, PDEs**

I. Introduction

The solution of large and composite Partial Differential Equations (PDE)s is a problem primarily faced with domain decomposition techniques [1], [2]. This approach involves decomposing at the linear algebra level after discretizing the domain and the equation with the desired method, i.e., Finite Differences (FD) or Finite Elements (FE). The main characteristic of these methods is the non-flexibility on the selection of different method for each subdomain of the initial problem. Interface Relaxation (IR) methodology is an interesting alternative [3]–[5]. Here, the PDE domain is decomposed into subdomains defined by the modelling of the underlying problem, while initial guesses are set on the interfaces between the subdomains. The subproblems are solved and new values on the interfaces are computed iteratively by particular IR methods (forcing the correct conditions for the problem), until convergence is succeeded.

Multidomain multiphysics problem solving environments (PSEs) implementing the interface relaxation methodology should be able to accommodate and incorporate a variety of existing PDE solvers and IR methods. These solvers should provide a minimum functionality including domain and PDE definition, mesh/grid generator, discretization scheme, evaluation of the solution and its derivatives at any point of the domain including the boundaries/interfaces. A complete list of existing software for the solution of differential equations can be found in [6].

A limited number of implementations of IR methods can be found in the literature. The first PSE implementing the IR methodology was the SciAgents Framework [7], [8]. This implementation exploits the parallelism inherent in IR methodology using the Agents computing paradigm over a network of heterogeneous workstations. A second approach was accomplished with BOND agent middleware [9]. Both SciAgents and BOND implementation used PELLPACK [10] for their PDE solvers. GasTurbnLab [11], [12] is the latest complete approach. It is a multidisciplinary PSE for the gas turbine engine design based on the Grasshopper agent middleware and FORTRAN and C libraries. Matlab has also been used for the implementation of IR methods [3], [13], not forming, though, a multidomain/multiphysics PSE. Last, a MATLAB toolbox that solves multidomain/multiphysics PDE problems is under construction, while a first stable version is presented in [14]. The main problems of the aforementioned PSEs are that they highly depend on the agent platforms and PELLPACK, revealing the need of a new implementation free of such constraints.

On the other hand, cloud computing introduces a set of technologies for delivering computational resources and services to the end-users according to their demands. The ability of flexibility and scalability of computational resources has set cloud computing an emerging technology for scientific applications that demand parallelization and high computational workload [15]. In this direction, many approaches turn to cloud computing for defining frameworks or architectures for parallel solution of scientific problems [16]. Another prominent paradigm is the SciCloud that studies the "unification" of already existing computational resources on research facilities under a cloud infrastructure for the execution of computational demanding

Manuscript received March 23, 2015; revised April 16, 2015. 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).

A. Korfiati is with Department of Computer Engineering and Informatics, University of Patras, Patras, Greece (phone: 00302610996985; e-mail: korfiati@ceid.upatras.gr).

N. Sfika, C. Alexakos and S. Likothanassis are with Department of Computer Engineering and Informatics, University of Patras, Patras, Greece (e-mail: {sfika, alexakos, likothan}@ceid.upatras.gr).

K. Daloukas is with Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece and Helic Inc., 2880 Zanker Road, Suite 203, San Jose, CA, USA (e-mail: kodalouk@inf.uth.gr).

P. Tsompanopoulou is with Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece (e-mail: yota@inf.uth.gr).

ISBN: 978-988-19253-4-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

and parallel scientific solutions [17]. A platform that supports multiple problem solving environments and is able to execute code in parallel has been introduced in [18]. Although there is a lack of implementations of IR methods with cloud technologies, there are other cloud-based approaches with high demands on computational resources such as satellite image processing [19], medical image processing [20] and bioinformatics [21].

In this paper, IRaaS (Interface Relaxation as a Service), a cloud-based PSE implementing the interface relaxation methodology, is presented. The parallel IR implementation is based on the architecture described in [22], where a geometric (GEO) contraction based IR method [13] was implemented along with FEniCS [23] and RabbitMQ [24] (a message-oriented communication middleware). In that approach, the virtualization method (virtual machines (VMs) with predefined memory, network and processors) was used. In the proposed cloud application, the actual process for solving multidomain/multiphysics problems is conducted in the cloud infrastructure. In addition, in order to take into account the diversity of the subproblems for a given problem, the proposed application manages to calculate and provide optimal VMs according to the available resources, the size of the global problem and the size of its subproblems. Finally, the application can be accessed by its users following the SaaS paradigm and providing a user-friendly web-based graphical interface that can be used by most of today's devices.

The rest of the paper is organized as follows. We provide some background details regarding cloud computing, PSEs, and the interface relaxation methodology in Section 2, while Section 3 describes the proposed cloud implementation. Next, we present the evaluation results of the proposed methodology. Finally, Section 5 concludes our paper.

II. METHODS AND TECHNOLOGIES

A. Cloud Computing

Cloud computing is a new model in the area of Information and Communication Technologies. Its main purpose is to provide access to all computing resources (such as applications, networks, storage, servers, services, etc.) directly from the web. Cloud computing aims to share resources among the cloud service vendors, consumers and partners resulting to the provision of computational resources as a utility [25]. One important benefit of Cloud computing is elasticity, i.e., the capability to scale the computational resources depending on the computational needs. Cloud computing has been divided into three service models: infrastructure as a service (IaaS), platform as a service (PaaS) and software as service (SaaS). The IaaS model delivers basic storage and different computational resources utilizing virtual machines. The PaaS model gives programming APIs to users for coding their own applications. SaaS is a model that delivers software applications over the web enabling the user to use an application from anywhere without caring about the computational needs [26].

In the present work we provide a cloud application (SaaS) named IRaaS where the users can easily manage and solve multidomain/multiphysics problems based on the IR methodology from any place and device without being concerned about the computational needs. For the proposed cloud implementation a software platform called Cloudstack has been combined with the existing virtualization infrastructure. Cloudstack is a tool that controls pools of computational resources, manages the network resources and storage in order to build cloud infrastructures according to the IaaS model [27].

B. FEniCS

The FEniCS project is a collection of free, open source software components forming an environment for the automated solution of differential equations. FEniCS provides scientific computing tools to specify the domain's properties (i.e., domain's geometry, PDE operator and boundary/interface conditions), define different types of element in the FEM algorithm, and efficiently solve the corresponding PDE problems.

FEniCS employs the Sparse LU algorithm for the solution of the underlying linear systems, mainly due to its robustness. However, since it can become slow and memory demanding in large problems, FEniCS provides iterative methods such as preconditioned Krylov solvers, as well, which are faster and require much less memory. The actual solvers implementations that are brought into action depend on the choice of the corresponding linear algebra package. PETSc is the default choice and uBLAS, Epetra (Trilinos) and MTL4 are other supported backends [23].

C. Advanced Message Queuing Protocol

RabbitMQ is a lightweight, reliable, scalable and portable message broker that enables efficient communication between applications to send and receive messages. It is compatible with all major operating systems and easy to use. It supports several languages among which Python, which is also used by FEniCS. RabbitMQ is based on the Advanced Message Queuing Protocol (AMQP), a message protocol that deals with publishers and consumers. The publishers produce the messages; the consumers pick them up and process them. [24]

D. IR methodology and GEO

Interface Relaxation methods, such as GEO, provide an efficient methodology for the solution of multidomain PDEs through an iterative procedure [3]–[5]. Consider the composite differential problem defined by

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

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, $\bigcap_{i=1}^{p} \overline{\Omega_i} = \emptyset$ and 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

ISBN: 978-988-19253-4-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

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 interface. GEO [7] is an interface relaxation method that allows for efficient calculation of the values at the interface points between different subdomains and guarantees fast convergence. The new relaxed values on the interface points are obtained by adding to the old ones a geometrically weighted average of the normal boundary derivatives of the adjacent subdomains. Specifically, the solution at each iteration k is given from the following equation:

$$u^{(k+1)} = u^{(k)} - \rho\left(\frac{\partial u_L^{(k)}}{\partial n} - \frac{\partial u_R^{(k)}}{\partial n}\right), \quad (3)$$
$$k = 1,2, \ldots$$

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

III. PROPOSED CLOUD IMPLEMENTATION

A. System Architecture

The system architecture of the proposed approach is based on the IaaS and SaaS cloud models and is efficient in terms of performance, scalability, user experience and energy cost. It supports the multidomain/multiphysics problem solving environment with the provision of a user-friendly interface, the automated distribution of the problem to appropriate computational resources, the storage of the problems' parameters and results and the necessary communication. This architecture is depicted in Fig. 1.

The major functional components of the architecture are the following:

In the **graphical user interface** users can create a new account, log in, upload data for a new problem to be solved and have access to the progress and the results of their current or previous problems. This module is implemented utilizing the state-of-art technologies and techniques for front-end web applications in order to be easily adapted in any end-user's device (PC, tablet, smartphone). It is build according to W3C's HTML5 standard, JQuery JavaScript framework and the responsive design technique provided by Bootstrap framework.

The **Advanced Message Queue Server** (AMQS) handles the communication between the entities of the system. It is based on RabbitMQ and on the Advanced Message Queuing Protocol (AMQP).

The **task manager** is responsible for the whole procedure. It manages the virtual machines (VMs) in the cloud infrastructure, checks for new potential jobs/problems and according to the size of each problem, it creates the appropriate VMs, sends them the input data through a web service and oversees their functionality during the whole execution. Moreover, if there are no jobs or all jobs have finished, it is responsible for destroying the VMs and releasing their computational resources.

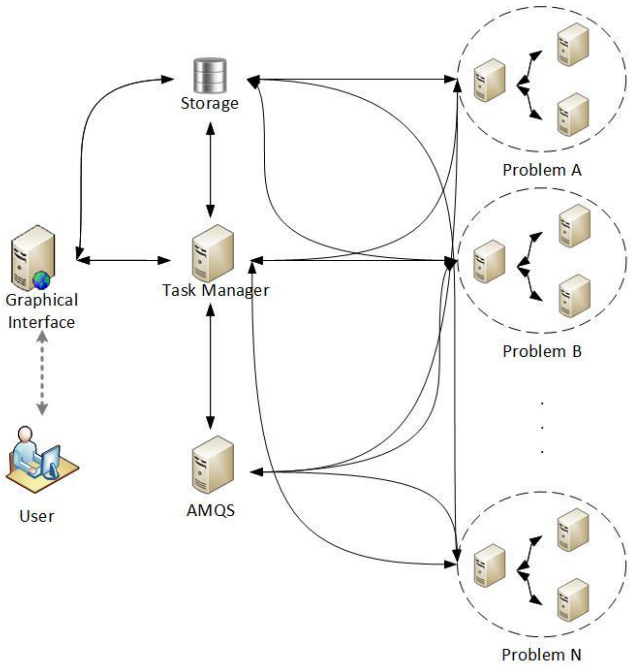

Fig. 1. IRaaS system architecture.

The **storage** is a universal repository accessible from all the system components. The graphical interface stores the users' input data. The VMs that are computing the problem solution get the users' input data from the storage and store their results. The task manager communicates with the storage in order to calculate the available and appropriate recourses for each problem.

The **execution VMs** are responsible for executing processing tasks. They are deployed and destroyed by the task manager and they receive input data from it, as well. Their life cycle begins with their initialization by the task manager when a new problem appears and the available resources are enough for its execution. They last for the time it takes to solve a problem and they die when they send the results to the storage in order to be accessed from the user interface.

B. IR Distributed Solution

The main characteristic of the GEO interface relaxation technique is the abundant level of parallelism in the solution of the interface points. In order to take advantage of such inherent parallelism, we assign each available VM the solution of one subdomain and the subsequent update of the values in the interface points. Thus, we can achieve full utilization of the computational resources found in a cloud infrastructure.

Referring to Fig. 2, the user defines the input parameters through a form in the graphical interface and then a Python script in each VM generates the subdomain's mesh

ISBN: 978-988-19253-4-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

(triangular elements), applies the boundary conditions and the initial guesses on the interfaces and expresses and defines the PDE problem as a variational problem. Then, the local solution and gradient are computed. Then, the computed solutions and gradients on the interfaces points are sent to the VMs that handle the adjacent subdomains. These VMs compute the new relaxed interface point values as in (3), 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. Owing to the independence of the solution for each subdomain, there is no need for a separate VM that will handle the communication step between VMs that solve neighboring subdomains. As a result, the communication overhead is greatly reduced.

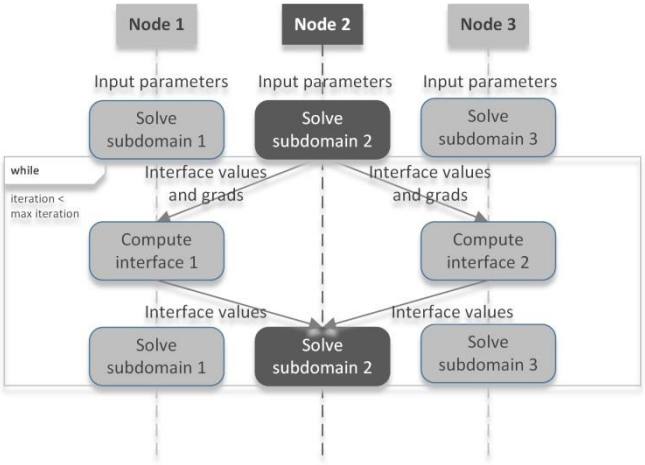

Fig. 2. Interface Relaxation distributed solution: an example with 3 subdomains and 2 interfaces.

C. Runtime Environment

The aforementioned system architecture is responsible for supporting the problem solving, from the provision of a user-friendly interface for defining problem parameters to the managing of the distributed process to the nodes.

The task manager is the main service that controls and orchestrates the whole procedure from the time when the user requests a solution for a problem, until the time when the VMs finish their executions and produce their results. It is also executed on a VM of the cloud infrastructure, communicates with every functional entity in the system and provides a series of functionalities. The most significant modules of the task manager are the following:

1) The job initiator, which iteratively reviews if there are any pending jobs in the queue. It chooses the oldest job request, reads the problem properties and calculates the optimal computational resources according to the available resources, the size of the global problem and the size of its subproblems. Then, it performs a request to the cloud platform in order to deploy the appropriate VMs (one for each subproblem) based on a template that contains all the software needed for the solution of each subproblem (such as FEniCS). Finally, using a web service, it passes to each VM the problem parameters. When the VMs have booted and received their input data successfully, they send a message back to the task manager. Fig. 3 depicts the job initiator's process.

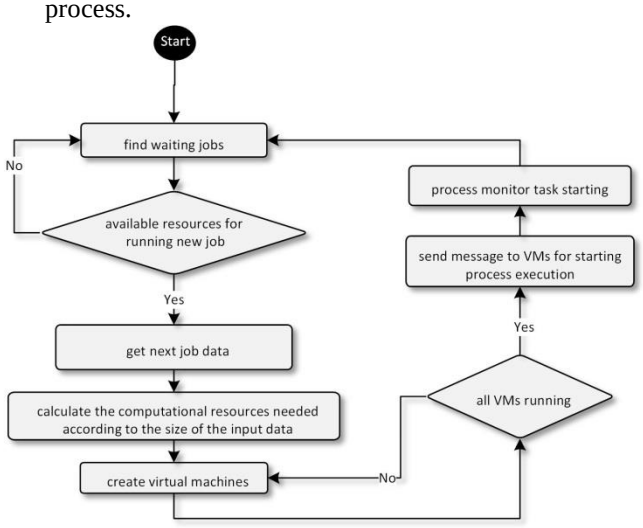

Fig. 3. Job initiator's process.

2) The job monitor, which checks the status of each running job. During the execution, each VM sends to the job monitor a RabbitMQ message declaring the iteration it has completed. Upon completion of the execution an end message is sent to the job monitor and this is the time when the task manager automatically destroys the VMs so that their resources are released and continues with the next problem in the queue, if any.

IV. EXPERIMENTS AND EVALUATION

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

$$Lu \equiv -\nabla^2 u(x,y) + \gamma^2 u(x,y) = f(x,y),\ (x,y) \in \Omega \quad (4)$$
$$u(x,y) = u^b(x,y), (x,y) \in \partial\Omega$$

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

$$u(x,y) = e^{y(x+4)}x(x-1)(x-0.7)y(y-0.5) \quad (5)$$

The problem consists of three subdomains which are depicted in Fig. 4. The interface points are at $x_1 = \frac{1}{3}$ and $x_2 = \frac{2}{3}$ and $\gamma^2 = 2$. 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 is equal to the number of points in the y direction of the middle subdomain, i.e., increases from 6 to 321 points. The left subdomain is approximately four times larger than the middle, while the right subdomain is approximately two times larger than the middle. The interfaces have the same number of points and therefore are of equal workload. The considered test cases along with their discretization step and the grid sizes for the left, middle and right subdomains are described in Table I.

ISBN: 978-988-19253-4-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

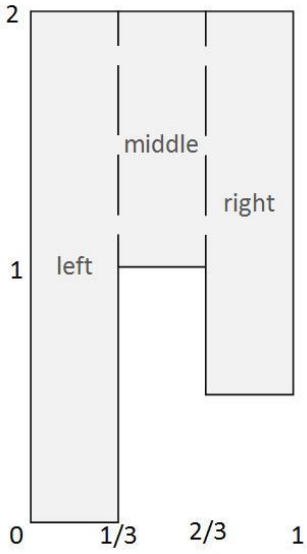

Fig. 4. Subdomains.

IRaaS has been implemented and installed in a small private cloud infrastructure of Cloud@CEID in the facilities of the Pattern Recognition Laboratory of the Dept. of Computer Engineering and Informatics of University of Patras, Greece. The test infrastructure comprises a server with an Intel(R) Xeon(R) CPU E3-1220@3.10GHz with 4 cores and 16GB RAM. As a result, our experiments are restricted to a small range of input datasets.

TABLE I
CONSIDERED CASES

Case	h	Left	Middle	Right
C1	0.1	4x21	4x6	4x11
C2	0.05	8x41	8x11	8x21
C3	0.025	14x81	14x21	14x41
C4	0.0125	28x161	28x41	28x81
C5	0.00625	55x321	55x81	55x161
C6	0.003125	108x641	108x161	108x321
C7	0.0015625	214x1281	214x321	214x641

The combinations of the resources used in order to produce the rule that decides the resources' allocation are presented in Table II. R1 is the minimum case where each subdomain is solved in a VM with a processor with 1 core, 2 GHz and 1 GB RAM. In R2 each subdomain is solved in a VM with a processor with 1 core, 2 GHz and 2 GB RAM. R3 is a more complex scheme where the left subdomain is solved in a VM with a processor with 1 core, 2 GHz and 4 GB RAM, the middle subdomain in a VM with a processor with 1 core, 2 GHz and 1 GB RAM and the right subdomain in a VM with a processor with 1 core, 2 GHz and 2 GB RAM. R4 is almost identical to R3, except that the middle subdomain is solved in in a VM with a processor with 1 core, 1.5 GHz and 2 GB RAM.

In Table III the execution times for all possible combinations for the aforementioned cases and resources are presented. Each value includes the time for the solution of the three subdomains along with the interface computations' time and the necessary communication for 15 iterations. It also includes the time it takes to inform the system for the progress of the computations, which takes approximately 12 seconds.

TABLE II
RESOURCES

Resources	Left domain			Middle domain			Right domain		
	cores	GHz	GB	cores	GHz	GB	cores	GHz	GB
R1	1	2	1	1	2	1	1	2	1
R2	1	2	2	1	2	2	1	2	2
R3	1	2	4	1	2	1	1	2	2
R4	1	2	4	1	1.5	2	1	2	2

In the first six cases it can be observed that the execution times are almost equal disregarding the computational resources used and as a result R1, i.e., the minimum resources allocation, would be sufficient. On the other hand in case C7 the R3 allocation of the resources is more appropriate because it handles better the tradeoff between time and computational resources minimization. This is due to the large size of the subproblems addressed in C7. The left subdomain comprises 274134 points, the middle 68694 points and the left 137174 points.

TABLE III
EXECUTION TIMES

Resources/Case	R1	R2	R3	R4
C1	**15.041**	15.082	15.016	15.042
C2	**15.411**	15.457	15.320	15.345
C3	**15.724**	15.894	15.736	15.735
C4	**19.169**	21.442	19.234	19.334
C5	**28.548**	37.271	28.669	28.470
C6	**71.416**	81.252	70.972	71.499
C7	500.146	307.241	**297.122**	298.870

Based on the above observations, we have employed the following rule for the allocation of computational resources:

IF subdomain size < =75000:
VM: 1 core, 2 GHz and 1 GB RAM.
IF 75000 < subdomain size < =150000:
VM: 1 core, 2 GHz and 2 GB RAM.
IF subdomain size >150000:
VM: 1 core, 2 GHz and 4 GB RAM.

This rule, when used from the task manager, reaches three goals simultaneously. It allocates the minimum possible resources, it 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.

Referring back to Table III, the execution time achieved and the resources allocated in the actual implementation of IRaaS, which is based on the aforementioned rule, are presented in bold letters. The comparison with the parallel implementation presented in [22] is shown in Fig. 5. Taking into account the additional 12 seconds that IRaaS needs to inform the system for the progress of the computations (which is not needed in the previous parallel implementation), the time minimization can be easily observed. This is especially evident with the increase of the problem size. The previous parallel implementation was performed in 3 nodes with 4 cores and 2GB RAM each,

ISBN: 978-988-19253-4-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

running as VMs on a 2 x Intel(R) Xeon(R) CPU E5-2620, 2.00GHz server with XenServer. Thus, apart from the reduction in the execution time, IRaaS is able to achieve better resource utilization, as well.

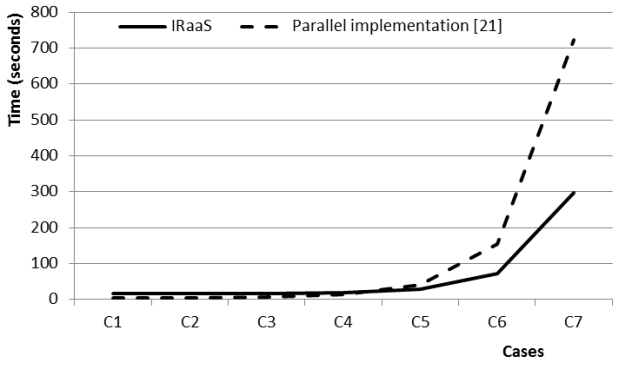

Fig. 5. Comparison to the parallel implementation in [22].

V. Conclusion

We have presented IRaaS, a cloud-based environment for the solution of multidomain/multiphysics problems based on the GEO interface relaxation methodology. The users: (i) define complex multiphysics PDEs, (ii) select the appropriate PDE solvers for the domains and IR methods for the interfaces and (iii) get the computed solution of the global problem. This SaaS cloud application is based on an IaaS model where the subproblems are automatically assigned to (not preexisting) VMs according to their computational needs. The advantages of the proposed environment are threefold. It provides significant reduction in the execution time by taking advantage of the inherent parallelism in the solution process of a multidomain problem, while at the same time provides increased resource utilization efficiency and allows for seamless integration, with limited user intervention

As future work, we plan to implement a larger number of PDE solvers and IR methods are going to be implemented and provided to the users soon, as well as provide support for more complex domains.

References

[1] Chan, T.F., Mathew, T.P., "Domain decomposition algorithms", in: Acta Numerica 1994, Cambridge University Press, Cambridge, 61-143 (1994)

[2] Keyes, D., Gropp, W., "A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation", SIAM J. Sci. Statist. Comput. 8 s166-s202 (1987)

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

[4] 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)

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

[6] Young, R., MacPhedran, I. (2006): Internet Finite Element Resources. [Online]: Available: http://homepage.usask.ca/~ijm451/finite/fe_resources/ (accessed March 17, 2015)

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

[8] Rice, J.R., Tsompanopoulou, P., Vavalis, E.A., "SciAgents Tool: User's Guide", Tech. Rpt. TR- 98-043, Dept. Computer Sciences, Purdue Univ., (1998)

[9] 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)

[10] 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)

[11] 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)

[12] 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)

[13] 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)

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

[15] Dikaiakos, M. D., Katsaros, D., Mehra, P., Pallis, G., & Vakali, A. (2009), "Cloud computing: Distributed internet computing for IT and scientific research". Internet Computing, IEEE, 13(5), 10-13.

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

[17] Srirama, S., Batrashev, O., & Vainikko, E. (2010, May), "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.

[18] Ludescher, Thomas, Thomas Feilhauer, and Peter Brezany, "Cloud-Based Code Execution Framework for scientific problem solving environments", Journal of Cloud Computing 2.1 (2013): 1-16.

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

[20] 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

[21] Krampis, K., Booth, T., Chapman, B., Tiwari, B., Bicak, M., Field, D., & Nelson, K. E. (2012), "Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community", BMC bioinformatics, 13(1), 42.

[22] 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)

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

[24] Rabbitmq, 2015. [Online]: Available: http://www.rabbitmq.com/documentation.html (accessed March 17, 2015).

[25] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010), "A view of cloud computing", Communications of the ACM, 53(4), 50-58.

[26] 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)

[27] Cloudstack, 2015. [Online]: Available: http://cloudstack.apache.org (accessed March 17, 2015).

ISBN: 978-988-19253-4-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)