Performance versus Cost of a Parallel Conjugate Gradient Method in Cloud and Commodity Clusters

Leila Ismail

Computer Science and Engineering, College of Information Technology, UAE University

Summary
Cloud computing is an emerging technology to run HPC applications using computing resources on a pay per use basis. The CG method is a linear solver which is used in many engineering and scientific applications, and is computationally demanding. We implement different approaches of a parallel CG method and compare their performance on different types of platforms: an HPC-optimized cluster, a built heterogeneous cluster, and Amazon cloud. We evaluate the performance of two approaches: broadcast and a ring-based communication-computation overlap, and compare to that of National Aeronautics and Space Administration Advanced Supercomputing CG parallel benchmark. We present an evaluation of the performance vis-a-vis cost tradeoff. The results show that, cloud instances suffer from network performance issues, which is revealed by the low performance of the CG method for small problem sizes. An HPC cloud instance type performs the best with relatively less cost than HPC-optimized commodity cluster and other more virtualized cluster instance types, for big problem sizes, while scaling well with increasing problem size. It gives better performance for overlap-based CG method; the performance increases and the cost decreases. Given the emergence of Cloud Computing, the results in this paper analyze performance and cost issues when Clouds are used for CG-based scientific and engineering applications.

Keywords
Distributed Systems, High Performance Computing, Cloud Computing, Conjugate Gradient (CG) Method, Perfor- mance, Cost.

I. INTRODUCTION

The capacity of today's infrastructures and data centers, the ubiquity of network resources, and the low storage cost have led to the emergence of Cloud Computing [1] [2] [3]. The main objective is to draw benefits from the underlying infrastructure services to satisfy a Service Level Agreement [4] and a required performance to users.

There is a great interest in porting high performance applications on a Cloud ([5], [6], [7]) mainly to reduce the cost of running the applications on commodity clusters, and the dynamic elasticity of resources. However, running distributed applications in a Cloud of distributed resources face performance challenges. Though the computing resources are presented to users as a unified set, they bear overhead which is induced by virtualization and the sharing of physical resources [8]. The Conjugate Gradient (CG) method is one of the most popular mathematical tool which is used by many scientific and engineering applications, such as oil simulation [9], aerospace engineering [10], and finite methods [11] to solve a linear system of equations generated in those applications. The CG presents a high complexity; i.e., the number of operations generated is very high. Consequently, enhancing the performance of the CG contributes largely to the efficiency of those applications. An important speedup can be obtained by distributing CG [14]. In this work, we evaluate and compare, in cloud computing and other com- modity clusters, mainly 2 versions of parallel CG method that we implemented: one using the broadcast approach which is simple to design and implement, and another one using an overlap approach in which communication is overlapped with computation to decrease communication cost, the latter approach being difficult to design and implement, but showing an important speedup. We also analyze the performance of the National Aeronautics and Space Administration Advanced Supercomputing CG parallel benchmark which is widely used by scientific community.

We implement the parallel CG method by using the producer-consumer pattern [15], also called master-worker. A master is a core in the Cloud, which is responsible to divide the CG load to computing workers. The master collects partial results from the computing workers and combines them to constitute the final application's result. Every computing worker processes the load received and transmits the results back to the master worker, which in turn, in a next iteration, performs some data preparations and sends new loads until the application is completed. At every iteration of the parallel CG method, computing workers communicate to exchange data which is needed to continue further steps within the iteration.

In this study, we present a performance vis-a-vis cost tradeoff of the parallel CG using different strategies of commu- nication on a variety of platforms, such as an HPC-optimized cluster, a built heterogeneous cluster and different instances types of Amazon Elastic Cloud Computing (EC2). Our HPC- optimized cluster consists of a total of 16 Intel Xeon cores connected by InfiniBand switch. The heterogeneous cluster is made of heterogeneous machines and heterogeneous Ethernet connectivity. We used three types of Amazon Elastic

Manuscript received November 5, 2012
Manuscript revised November 20, 2012

Compute Cloud instances which are representative of infrastructures that can be used for HPC applications: Standard Large In- stance, High CPU Extra Large Instance, and Cluster Compute Quadruple Extra Large Instance. Several applications were evaluated using a cloud computing infrastructure ([8], [16], [17], [18], [19]), however, to our knowledge our work is the first attempt to evaluate the CG method in a cloud computing environment and to analyze and compare its performance and cost on different commodity clusters. Our study reveals that the cloud computing could be a better choice for high performance applications than commodity clusters when virtual instances specifications are chosen carefully according to the application characteristics; i.e., depending on whether the application to be deployed is communication-intensive and/or computing intensive and considering the ratio of communication to computation. The rest of the paper is structured as follows. Section II describes the CG method parallelization approach that we consider for the experiments. Section III overviews related works.

The computing platforms under experiments are described in section IV. In section V, we evaluate the performance of the CG parallel application running on different platforms and discuss the obtained results. Section VI analyzes the tradeoff between the performance and the price cost of running a parallel CG. Section VII concludes the work.

1) $x_0 = 0$
2) $r_0 := b - Ax_0$
3) $p_0 := r_0$
4) $k := 0$
5) $Kmax :=$ maximum number of iterations to be done
6) if $k < kmax$ then perform 8 to 16
7) if $k = kmax$ then exit
8) calculate $v = Ap_k$
9) $\alpha_k := \frac{r_k^T r_k}{p_k^T v}$
10) $x_{k+1} := x_k + \alpha_k p_k$
11) $r_{k+1} := r_k - \alpha_k v$
12) if r_{k+1} is sufficiently small then go to 16 end if
13) $\beta_k := \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
14) $p_{k+1} := r_{k+1} + \beta_k p_k$
15) $k := k + 1$
16) $result = x_{k+1}$

II. PARALLEL COMPUTATION

APPROACH Figure 1 shows a sequential execution of the CG method.

We use data parallelism to distribute the load generated by the following operations: matrix-vector multiplication (step
8), vector-dot product (step 9 and step 13), scalar-vector multiplication and vector addition (step 10 and step 14), and scalar-vector multiplication and vector subtraction (step 11). The number of non-zeros in the matrix A is distributed equally over the available number of computing workers for load balancing purpose. Vectors are distributed based on the number of rows of the local matrix on each computing worker. we use the master-worker model for the CG distribution. When the distributed CG runs, one of the cores of the Cloud instances is selected at random as the master and other computing cores are selected as associate workers to perform in parallel the CG's partitions, which are assigned by the master core. A core here is defined as the smallest processing unit that is allocated to an application. A core could be a context, a core or a processor of a Cloud instance which consists of multiple cores. Cloud cores are connected via a network link whose speed and load dictate the speed of the communication processing when
data is transmitted between nodes. Upon starting the parallel CG execution, the master divides the matrix into among the computing workers which perform local operations on local parts. The conditions of the communication link between the master and each computing worker, and among the computing workers and the existing computing power of each computing worker, all participate to the performance of the parallel CG. A communication channel depicts the time needed for data to travel over the connection connecting the master worker to a computing worker and the computing workers to each others. A computing power depicts the time needed by the computing worker to process the task assigned by the master. Each computing worker in the system has a computing capacity.

In order for each local computing worker to achieve its local computation of step 8, the whole vector p is needed by all the computing workers. However, each computing worker has only a local part of the vector p. Therefore a communication of the local vector p from each computing worker to all other computing workers is necessary in every iteration of the CG algorithm. We implemented 2 communications approaches, widely used by the parallel computing community:

• Broadcast-based strategy. In this approach, every computing
worker communicates a vector p of size l_i ($1 \le i \le N$) to the N computing workers participating in the parallel computation of the CG method. li is the size of the vector p whose number of elements is equal to the number of rows of the local matrix associated to the computing worker i, following a load-balanced distribution of the matrix A. The broadcast-approach is simple to design and implement for

the CG method. However, communication cost will increase drastically with the size of vector p and the number of computing workers, posing scalability issues. The total communication cost from a computing worker i to broadcast the vector p during the execution of the parallel CG method is as follows:

$$TComm_p = TBcast(l_i) * k_{max} \qquad i = 1, ..., N$$

Where TBcast(li) is the time needed to broadcast the vector p of size li from a computing worker to N – 1 computing workers, and kmax is the maximum number of iterations needed at the completion of the parallel execution of the CG.

• Overlap-based strategy. In this approach, the parallel CG method is designed in a way that the communication of the vector p is overlapped with the local matrix- vector multiplication of the step 8. We chose the ring- based approach to design and implement the overlap- based CG method. A complete description of the overlap- based approach and analysis of its performance vis-a-vis the broadcast-based approach can be found in [14]. The overlap-based approach aims to hide the communication by communicating asynchronously with other computing workers while the local computation is taking place. However, in case that the local computation of the matrix-vector multiplication takes less time than communicating the vector p, then a waiting time is generated on the computing worker whose local computation has com-pleted before receiving the chunk of p to work with. The communication cost of the parallel CG method is as follows:

$$TComm_p = T(Wait) * (N-1) * k_{max} \quad i = 1, ..., N \quad (2)$$

The best scenario is when $T(Wait) = 0$

III. RELATED WORKS

Reference [5] deployed two parallel applications on Amazon EC2: the classification of gene expression data and the functional magnetic resonance imaging (MRI) workflows used in brain analysis. Both applications involve computation and communication. Two types of Amazon instances were uses: the standard small instance type and the standard medium instance type. The first application performs better in a small type than in a medium type, in particular because the application is designed as single-threaded processes and consequently did not get benefit from the multi-core feature of the medium type. The paper concludes that the cost of running the MRI application when increasing the number of Cloud nodes is subdued by the significant reduction in application makespan. Reference [8] looks at the perfor- mance and the cost of applications in the areas of climate, material-science, fusion, accelerator modeling, astrophysics, and quantum chromodynamics. Amazon EC2 shows a lower performance than other two commodity clusters under study. Reference [16] examines the performance of running coupled atmosphere-ocean climate models on Amazon EC2, using small, medium and large standard and high-cpu instance types. Reference [18] compares the performance and the cost of the bioinformatics application (WCD) between Amazon EC2, using the large instance type, and two commodity clusters. WCD is characterized by generating more computation than communication as the data size grows. The performance and the cost of running the Montage workflow are examined by reference [19]. Other applications, such as NAMD, a molecular dynamics application, and NQueens, a backtracking search problem, were evaluated by reference [17].

IV. EXPERIMENTAL TESTBED

In this section, we describe the different platforms we use for our experiments. These platforms are representative of different categories of platforms available to High Performance Computing community. From each platform, we use 16 cores to deploy and run our applications: the broadcast-based par- allel CG method, the overlap-based parallel CG method and the NPB3.2.1 version of the NAS parallel benchmark.

A. Amazon Cloud Computing

Amazon Elastic Cloud Computing (EC2) [20] is probably one of the most well known cloud computing infrastructures which implement Infrastructure As A Service (IAAS) cloud [21]. EC2 exposes to users an Application Programming Interface which allows the deployment of cluster and Grid environments and the deployment of parallel applications. Amazon provides an amount of CPU per hour, no matter what the underlying hardware is. It is based on the utility computing model [22]. One EC2 Compute Unit

EXPERIMENTAL TESTBED: AMAZON ELASTIC CLOUD COMPUTING (EC2)

Instance Type	*Memory*	*Number*	*Number of EC2*	*Number*
Standard Large	7.5 GB	2	2	8
High-CPU Extra Large	7 GB	8	2.5	2
Cluster Compute Quadruple Extra Large	23 GB	2 x Intel Xeon X5570, quad- core Ne- halem architec- ture	4.19	2

provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or
2007 Xeon processor [23]. This is also the equivalent to an early-2006 1.7 GHz Xeon processor. We chose 3 different types of instances of the EC2 that are representative to run distributed applications. In amazon, a core is a virtual core. Table I shows the different types of instances which are used in our experiments. We use US East (Virginia) Amazon AWS region.
We use the Sun N1 Grid Engine as an engine to dispatch jobs to the virtual cores of the Amazon EC2. Our parallel CG method as well as NAS use Message Passing Interface (MPI) [24]. We use the StarCluster cluster-computing toolkit [25] which is developed by the Massachusetts Institute of Technology (MIT) group. StarCluster uses the API of the Amazon EC2 to build, configure and manage clusters of virtual cores. We use version 0.93.1 of the toolkit.

B. Heterogeneous Cluster

To analyze the performance and the cost of the parallel CG method on a heterogeneous cluster, we conduct the experiments on a set of heterogeneous machines made of 2 AMD Opteron processors and 7 Intel Xeon processors. The AMD Opteron Processor model is 252 with 2.59 GHz dual CPU single core. Each core has 1MB of cache, and 2 GB of memory. 5 out of the 7 Intel machines have Intel Xeon CPU of
3.06 GHz, dual CPU, dual core. Each core has 512KB of cache and 4GB of memory. Let us denote those Intel machines by Intel Type1. 1 out of 7 Intel machines has Intel Xeon CPU of
3.06 GHz with dual CPU, single core. Each core has 512 KB of cache and 4 GB of memory. Let us denote this Intel machine by Intel Type2. The AMD Opteron machines along with those
6 Intel machines described earlier are connected to each others via one hop 1Gb/second switch, those are connected to another Intel machine located in the shared LAN network via two-hop connectivity of 1 Gb/s. That Intel machine has Xeon CPU of
3.0 GHz and it has a single CPU, dual core. Let us denote that

TABLE II HE T E RO G E N E O U S CL U S T E R TE S T B E D

Computing Worker		Machine	
	CPU Type	Memory Size	Cache Size
CW0 and CW1	AMD Opteron Processor	2GB	1MB
CW2 and CW3	AMD Opteron Processor	2GB	1MB
CW4 and CW5	Intel Type 1	4GB	512KB
CW6 and CW7	Intel Type 1	4GB	512KB
CW8 and CW9	Intel Type 1	4GB	512KB
CW10	Intel Type 3	4GB	512KB
CW11 and CW12	Intel Type 1	4GB	512KB
CW13 and CW14	Intel Type 1	4GB	512KB
CW15	Intel Type 2	4GB	512KB

Intel machine by Intel Type3. Each core has 4MB of cache and 2GB of memory. The Linux Suse 3.1.0. was used. 16 computing workers (cores) from the heterogeneous platform were used to run the experiments. The workers are mapped to their corresponding machines as shown in Table II. Figure 2 shows the performance of the machines in terms of MFlops, measured by running the class C of the NAS benchmark on the platform.

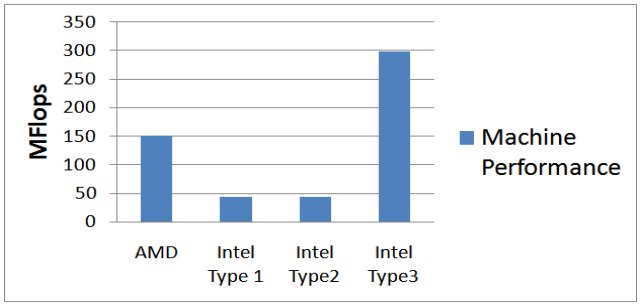

Fig. 2. Computing performance of our experimental heterogeneous platform.

C. HPC-Optimized Cluster

Our HPC-Optimized cluster consists of 2 Xeon Intel Quad Core 5355 machines with 2.66GHz CPUs. Each machine has a dual CPU. Each core has 4MB of cache, 1GB of memory,
2.66 x 4GFLOPS of peak performance. The machines are connected using Qlogic 9120 InfiniBand (IB) switch. The operating system used on the machines is Red Hat Enterprise Linux Server release 5.2. The experiments are done using the parallel capabilities of a single multicore machine, and using a Cloud of machines. Message Passing Interface (Open MPI version 1.3.2) library is used for implementing the parallel CG.

V. PERFORMANCE EVALUATION

In this section, we evaluate efficiency of the parallel CG method for broadcast-based implementation and overlap-based implementation when running on the different platforms under experiments. We also evaluate the performance of the NAS

A. Experimental Runs

The experiments use a set of 16 cores running 16 computing workers. The total execution time of the parallel CG is measured on each platform under experiments. The relative runtime of the parallel CG on the Amazon EC2 platforms and the heterogeneous platform compared to an HPC-optimized cluster is measured. In our experiments, one core acts as a master which distributes the tasks to the other cores that we call computing workers. The total execution time for the parallel CG is the elapsed time when master starts distributing the matrix till the final results are received by the master from the computing workers. The gettimeofday function is used to compute the elapsed time on the master. The average communication time involved in the parallel CG method is computed by computing the total communication time on each computing worker divided by the number of computing workers involved in the computation. In all our experiments, each experiment was run 100 times and the average was computed. The experiments use different matrix sizes of the matrix A. There is a total of 5 experiments or runs as shown in Table III to assess the impact of increasing matrix size to the performance of the parallel CG on the platforms under experiments. The matrix sizes are the same used by the NAS parallel CG benchmark. We also use the same sparsity patterns as implemented by NAS. In each run on a platform, the total execution time of the parallel CG is measured by vertically scaling the matrix size. On the heterogeneous platform, each run has executed using a load-balanced approach. The load- balanced approach takes into consideration the heterogeneity of the machines in the heterogeneous cluster. It computes dynamically the corresponding load; i.e., the number of non- zero elements of the matrix A to be allocated to each machine of the cluster based on its available computing power.

In order to know the relative speed of the CG method on each platform compared to the HPC-optimized cluster, we calculate the ratio of the total execution time of the CG method on the platform over its total execution time on the HPC- optimize cluster; with increasing problem size.

B. Experimental Results Analysis

When running our MPI-based implementation of the parallel CG method to Amazon cloud computing, we had the following requirements. These requirements have an impact on the usability of a cloud infrastructure to deploy and run distributed applications based on MPI.

a) Building a virtual cluster within the Cloud. To run a parallel CG on the allocated instances, it is needed to build a cluster out of the number of instances used to run the parallel CG. A cluster should be configured and consequently managed. Some instances can be configured as administration nodes while others are configured as computing nodes. Amazon provides API tools which serve as a client interface to launch and destroy instances, to read and write files within the instances, etc. We use the StarCluster [25] software which automatically configures a cluster out of Cloud instances. To configure and manage the cluster, Star Cluster uses an endpoint of communication defined by Amazon [27] depending on the region in which instances are running. In our experiments the endpoint of communication is ec2.us − east − 1.amazonaws.com for US East (Northern Virginia) Region.

b) Installation of the development tools in the Cloud. To run a CG in the cloud, it is needed to install development tools, such as MPI and gcc libraries for compiling and running CG. The StarCluster tools deploys an image which includes all the necessary tools to compile, dis- patch an run an MPI-based program.

c) Computing workers and user access to the virtual cluster file system. The cluster should have a shared file sys- tem between the cluster nodes to run our parallel CG implemented by using MPI [24] an to access files as a result of the parallel CG execution. We use commands of StarCluster tools to copy files, resultant from the CG parallel execution, from the virtual cluster to our local machines and analyze the results.

At the platform level, although it is obvious that the Amazon EC2 cluster compute quadruple extra large instance type will perform better than the Amazon standard, and the Amazon high-cpu extra large instance type, due to a higher network bandwidth assigned to the cluster compute [20] and more dedicated CPU cycles [28], as shown in Figures 3, 4, and

5. However, our experiments show the performance gain that can be obtained by the CG method when considering the performance of the platforms under experiments for the broadcast-based approach and the overlap-based approach. For the broadcast-based approach, the HPC-optimized cluster has the best performance, followed by the Amazon cluster com- pute for small problem sizes (Class S, Class W and Class A). This is due to the small percentage of computation compared to the communication that should be done for small problem sizes. For instance, Figure 6, for the class S, shows that more than 70% of the total execution time is spent in communication by the Amazon cluster compute. However, the Amazon cluster compute performs better than HPC-optimized cluster for big problem sizes (Class B and Class C), where the percentage of computation to

communication increases and thus the cluster compute computing capabilities are used, as shown in Figure 7 for the Class C problem size for example. Consequently a more computing power in the Amazon cluster compute does not give a boost to performance for small problem sizes, this is due to the Amazon cluster compute network being shared with other applications and users by virtualization. Except for the Amazon cluster compute, commodity clusters are performing better than the Amazon cloud instances under study for the broadcast and overlap as shown in Figures 3, 4, and 5. However, for NAS, the high-cpu extra large is performing better than the heterogeneous for large problem sizes, as the percentage of computation increases and thus NAS benefits from a larger computing power provided by the high-cpu extra large compared to our heterogeneous commodity cluster. For the overlap approach, the Amazon cluster compute shows the best performance as shown in Figures 4 and 8.

Figure 8 shows that the performance gap is bigger for small matrix sizes than larger matrix sizes, showing that platform types other than HPC-optimized clusters starts to perform for bigger matrix sizes than smaller matrix sizes. For instance, the HPC-optimized cluster is 195.3 times faster than the standard large instance type for NAS benchmark for class S matrix size. It is 131.4 times faster for the broadcast-based approach, and it is 55.7 times faster for the overlap-based approach. Our HPC-Optimized cluster performs 82.2 faster than the high-CPU extra large for the NAS parallel benchmark,

78.7 faster for the broadcast-based approach, and 62.7 faster for the overlap-based approach. Our HPC-optimized cluster performs 2 times better than the cluster compute quadru- ple extra large performs for the broadcast-based approach. However, the cluster compute quadruple extra large performs better for the overlap-based approach (2.5 faster) and for the NAS parallel benchmark (1.67 faster). The performance gap between our HPC-optimized cluster and the other Amazon instance types decrease with increasing matrix sizes, showing that the Amazon cloud computing becomes more interesting for running a parallel CG, when more computation is involved with larger matrix sizes. This is because for small matrix sizes, more communication is involved than computation when running a parallel CG, making cloud computing less effective as shown in Figures 9, 10, and 11, in particular for the broadcast-based approach.

VI. COST EVALUATION

Table IV shows the price that is charged per hour of execution. For our HPC-optimized cluster, we make an assumption of a charging rate of US$2.5 per core per hour [29]. For our built heterogeneous cluster, we make an assumption of US$0.6 per core per hour. The objective here is not to make a comparison of prices but to evaluate the tradeoff between the execution of running a parallel CG method and NAS benchmark and their corresponding costs on the platforms under study. This is with increasing matrix size. To compute the total runtime cost, TC ost , we use the following formula:

TC ost = TRun ∗ N (8) (price/(60 ∗ 60))

where, TRun is the total runtime of the parallel application, N is the number of cores used in the parallel computation, and price is the price of use per core per hour.

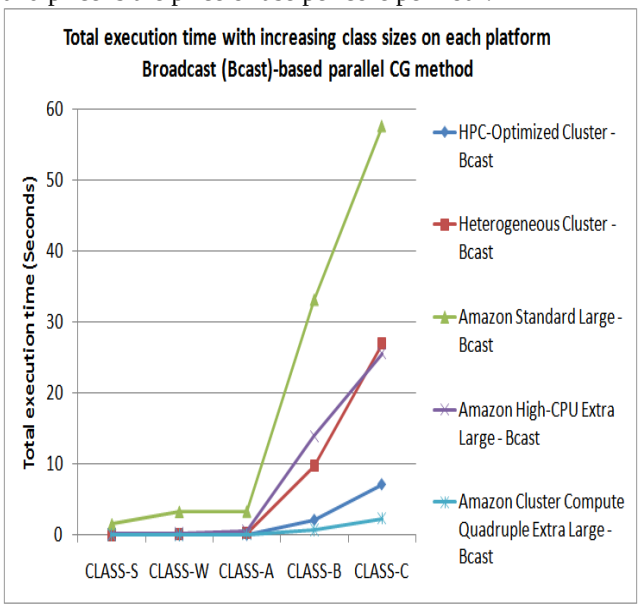

Fig. 3. Total makespan of the broadcast-based approach with increasing problem size running on different platforms.

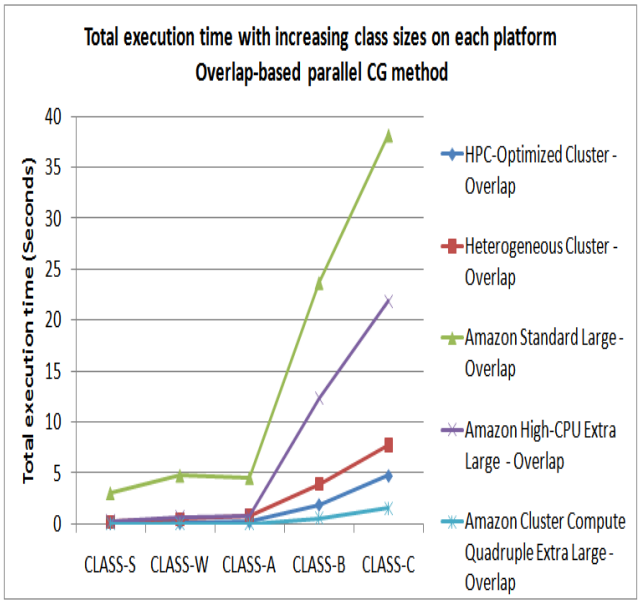

Fig. 4. Total makespan of the overlap-based approach with increasing problem size.

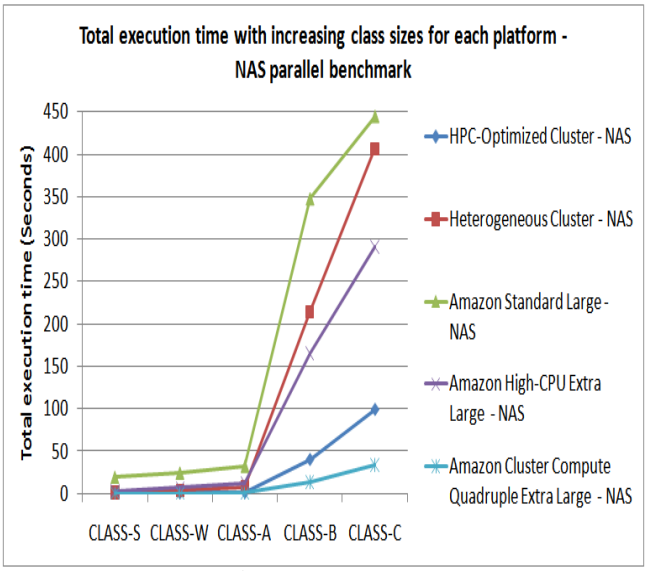

Fig. 5. Total makespan of the NAS parallel benchmark with increasing problem size running on different platforms.

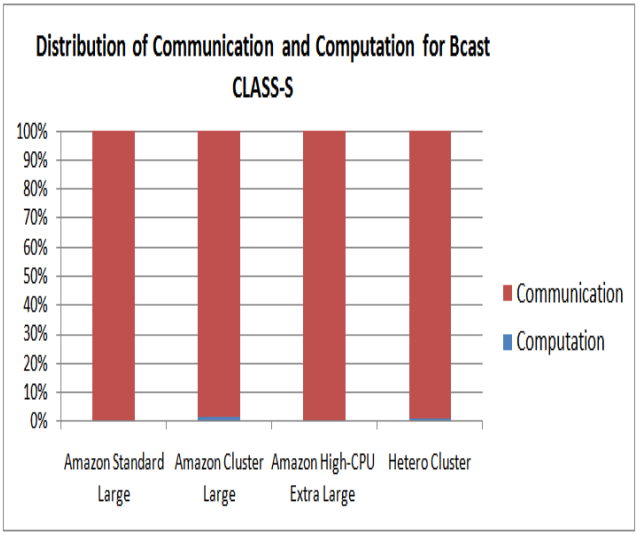

Fig. 6. Distribution of communication and computation for the Class S problem size running on different platforms.

Figures 13, 14, and 15 show that the total runtime cost as a function of total execution time and increasing problem size. For all cases, the cost increases as the problem size increases, due to the linear relationship between problem size and total execution time within the same platform. For all cases, the Amazon cluster compute instance shows the best cost to performance relationship for all problem sizes, except for the Class S problem size and for the broadcast-based approach, where the use of an HPC-optimized cluster is less costly than Amazon cluster compute. These results show it is more cost- effective to run a parallel CG on an HPC-Cloud instance type. The performance of a parallel CG method increases and the cost decreases by using a communication-computation overlap strategy. For instance, the total execution time of the overlap-based parallel CG is 1,52 versus 2.33 seconds of the broadcast-based approach for the Class C matrix size, while the corresponding cost is 0.017 versus 0.011.

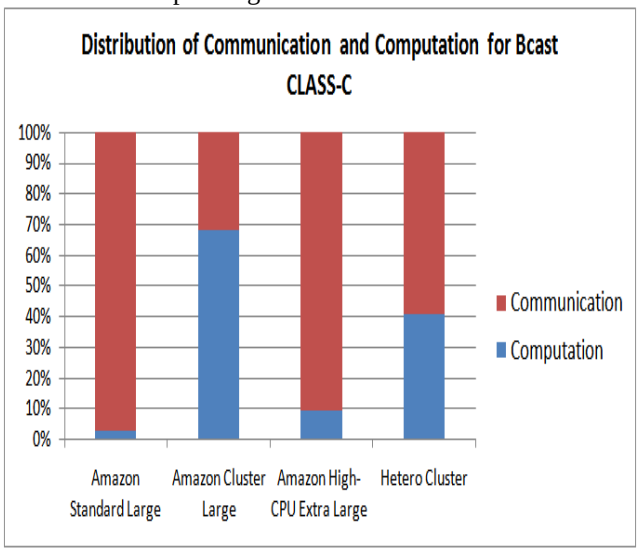

Fig. 7. Distribution of communication and computation for the Class C problem size running on different platforms.

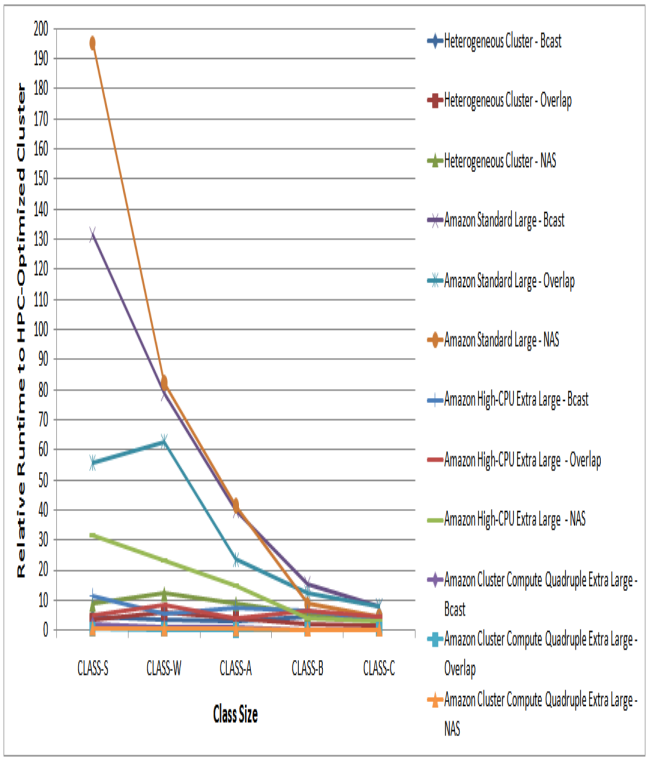

Fig. 8. Relative Runtime of the parallel CG method and NAS on different platforms to the runtime on HPC-optimized cluster

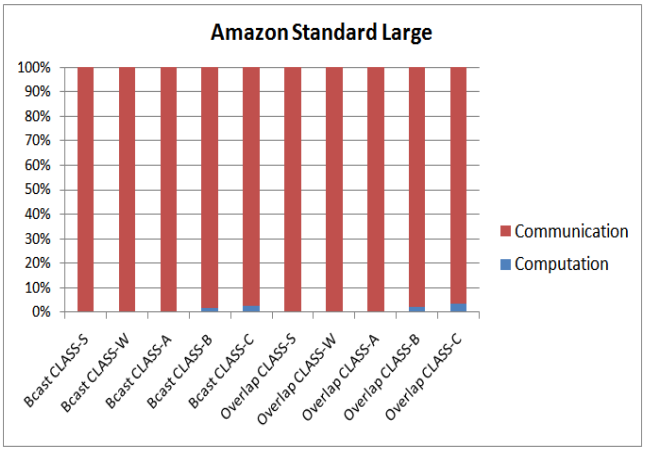

Fig. 9. Percentage of Communication and computation time with increasing matrix size for the broadcast-based approach and the overlap-based approach

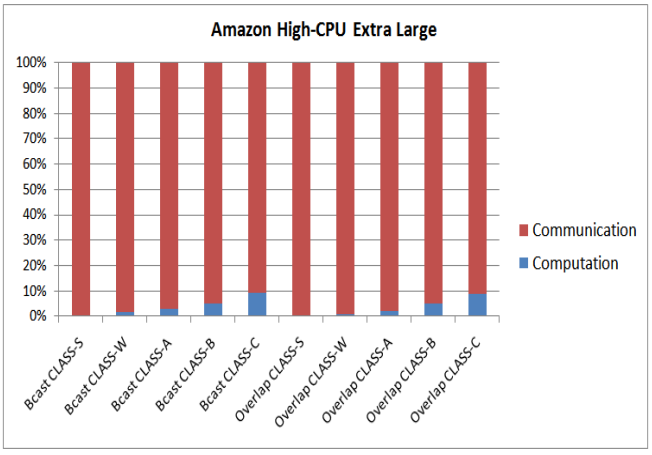

Fig. 10. Percentage of communication and computation time with increasing matrix size for the broadcast-based approach and the overlap-based approach.

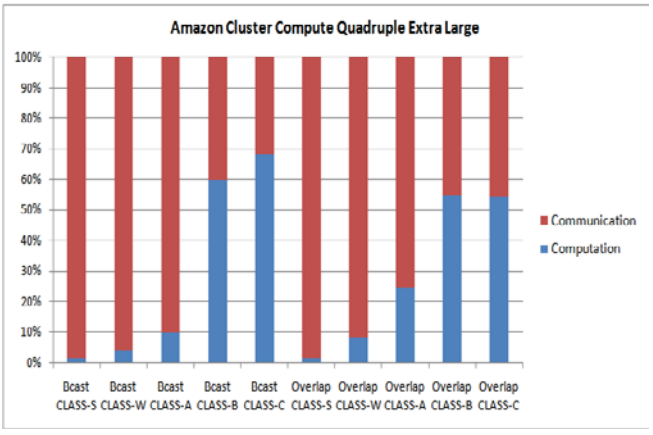

Fig. 11. Percentage of communication and computation time with increasing matrix size for the broadcast-based approach and the overlap-based approach

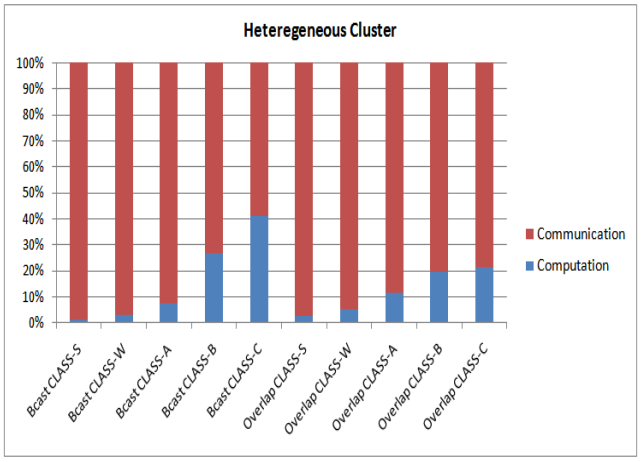

Fig. 12. Percentage of communication and computation time with increasing matrix size for the broadcast-based approach and the overlap-based approach.

TABLE IV CO S T P E R US E

Platform Type	Price per Hour (US$)	Number of Cores or Instances Used
HPC-Optimized cluster	2.5 per core	2
Built heterogeneous cluster	0.6 per core	8
Amazon EC2 Standard Large instance type	0.34 per instance	8
Amazon EC2 High-CPU Extra Large instance type	0.68 per instance	2
Amazon EC2 Cluster Compute Quadruple Extra Large instance type	1.3 per instance	2

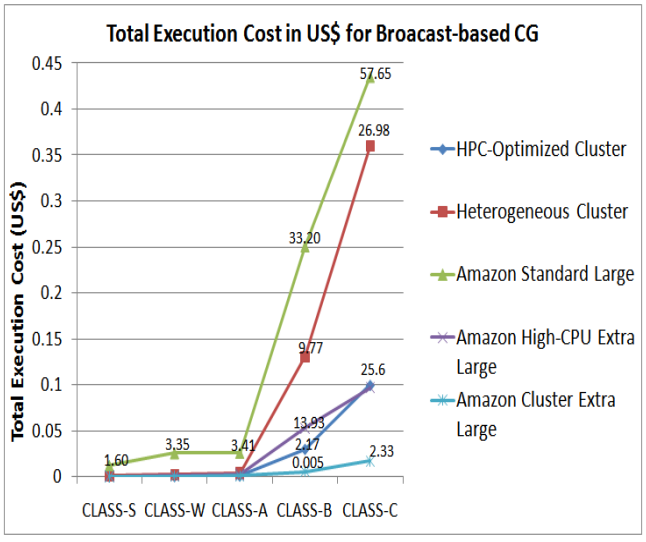

Fig. 13. Total execution cost of the broadcast-based approach for the parallel CG on different platforms with increasing matrix size.

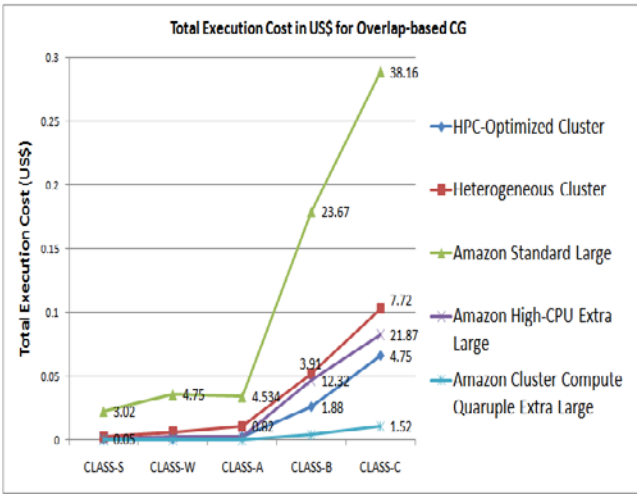

Fig. 14. Total execution cost of the overlap-based approach for the parallel CG on different platforms with increasing matrix size.

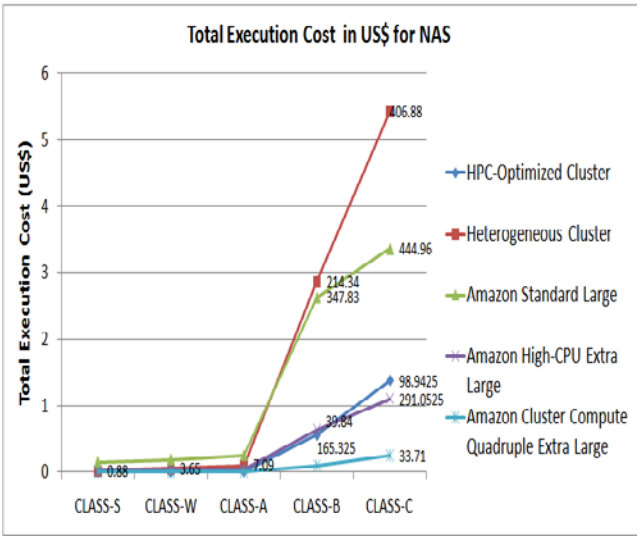

Fig. 15. Total execution cost of NAS parallel benchmark on different platforms with increasing matrix size

VII. CONCLUSION

In this work, we study the tradeoffs between the performance and the cost of a parallel CG application on different cluster types, ranging from a commodity heterogeneous clus- ter, to an HPC-optimized cluster to different cloud instance types including HPC-cloud instance type. We used the Ama- zon Cloud as a Cloud reference platform. We experimented with two versions of a parallel CG, one based on a broadcast approach strategy for the inter-processors communications and one based on a communication-computation overlap strategy using the ring approach. We also experimented with the NAS CG benchmark as it is widely used as a reference by the scientific community. Our results show that the HPC-Cloud instance type performs the best results with increasing matrix size. In addition, the performance of the parallel CG increases as the cost decreases compared to the other platforms under study. Other cloud instance types result in a bigger execution time and a bigger cost compared to the HPC-Cloud instance type. Our results show that, on those cloud types, a big percentage of the CG method total execution time was spent in communication. This is due to the virtualization mechanisms and the cloud network being shared by different applications.

ACKNOWLEDGEMENTS

This work is supported by the UAE University research grant. The research is part of a project that is ranked Highly Competitive by international peer reviewers following a UAE national competition organized by the UAE National Research Foundation and won by the author.

RE F E RE N CE S

[1] R. Buyya, C.S. Yeo, and S. Venugopal, Market-Oriented Cloud Comput- ing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, Keynote Paper, in Proc. 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008), IEEE CS Press, Sept. 2527, 2008, Dalian, China

[2] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems. Volume 25, Issue 6, June 2009

[3] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G.Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: A Berkeley View of Cloud computing. Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, February 10, 2009

[4] Service Level Agreement Zone, “The Service Level Agreement”, @Copy- right 2007, http://www.sla-zone.co.uk/index.htm

[5] Christian Vecchiola1, Suraj Pandey1, and Rajkumar Buyya, ”High- Performance Cloud Computing: A View of Scientific Applications”, Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN 2009, IEEE CS Press, USA), Kaoh-siung, Taiwan, December 14-16, 2009

[6] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, M. Tsugawa, ”Science Clouds: Early Experiences in Cloud Computing for Scientific Applica- tions”, in the Proceedings of the Cloud Computing and its Applications 2008 (CCA-08), October 2008

[7] K. Keahey, ”Cloud Computing for Science”, 21st International Confer- ence on Scientific and Statistical Database Management, Springer-Verlag, p. 478, 2009

[8] Keith R. Jackson, Krishna Muriki, Shane Canon, Shreyas Cholia, and Jon Shalf, ”Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud”, 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 159-168, November 30 2010-December 3 2010,

[9] Dogru, Ali h., From Mega-Cell to Giga-Cell Reservoir Simulation, Saudi Aramco Journal of Technology, Spring 2008

[10] J. W. Manke, ”Parallel Computing in Aerospace”, Parallel Computing, Vol. 27, Issue 4, March 2001, pp. 329-336

[11] Haee-Dae Kwon, ”Efficient Parallel Implementations of Finite Element Methods Based on the Conjugate Gradient Method”, Applied Mathemat- ics and Computation, Vol. 145, Issue 2-3, 25 December 2003, pp. 869 -880

[12] Leila Ismail, Khaled Shuaib, ”Empirical Study for Communication Cost of Parallel Conjugate Gradient on a Star-Based Network,” ams, pp.498-503, 2010 Fourth Asia International Conference on Mathemati- cal/Analytical Modeling and Computer Simulation, 2010

[13] Leila Ismail, ”Communication Issues in Parallel Conjugate Gradient Method using a Star-Based Network”. 2010 International Conference on Computer Applications and Industrial Electronics (ICCAIE 2010), 05-07December 2010, Kuala Lampur, Malaysia

[14] Leila Ismail, Rajeev Baru, ”Implementation and Performance Evaluation of a Distributed Conjugate Gradient in a Cloud Computing Environment”, Software: Practice and Experience journal, 2012, Wiley InterScience. DOI:10.1002/spe.2112

[15] Ian Foster, ”Designing and Building Parallel Programs”, Addison-Wesley (ISBN 9780201575941), 1995

[16] Constantinos Evangelinos and Chris N. Hill, ”Cloud Computing for Parallel Scientific HPC Applications: Feasibility of Running Cou- pled Atmosphere-Ocean Climate Models on Amazon’s EC2”, CCA-08, Chicago, 2008

[17] Abhishek Gupta an Dejan Milojicic, ”Evaluation of HPC Applications on Cloud”, Hewlett-Packard Development Company, L.P., 2011.

[18] Scott Hazelhurst, ”Scientific computing using virtual high-performance computing: a case study using the Amazon Elastic Computing Cloud”, the Proceedings of the South African Institute of Computer Scientists and Information Technologists (SAICSIT) Conference, 978-1-60558- 286-3, 2008

[19] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good, ”The cost of doing science on the cloud: the Montage Example”, In Proc. of the ACM/IEEE conference on Supercomputing (SC’08), 2008

[20] http://aws.amazon.com/ec2/

[21] Mell P, Grance T. A NIST Definiton of Cloud Computing. Na- tional Institute of Standards and Technology, Special Q12 Publication 800145, September 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[22] Amazon, ”What is a EC2 Compute Unit and why did you introduce it?”, http://aws.amazon.com/ec2/faqs/#What is an EC2 Compute Unit and why did

[23] LegitReviews.com, ”Intel’s Mighty Dual Xeon Beast - V8, Platform Preview”, http://www.legitreviews.com/article/527/1/, Copyright 2002-2012, last retreived on 15 April 2012

[24] William Gropp, Steven Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, Bill Nitzberg, William Saphir and Marc Snir, ”MPI: The Complete Reference”, Vol. 2, ISBN-10:0-262-57123-4, ISBN-13:978-0-262-57123-4, September 1998

[25] StarCluster by MIT, http://web.mit.edu/star/cluster/docs/latest/overview.html

[26] Bailey D, Barszcz E, Barton J, Browning D, Carter R, Dagum L, Fatoohi R, Fineberg S, Frederickson P, Lasinski T, Schreiber R, Simon H, Venkatakrishnan V,Weeratunga S. The NAS Parallel Benchmarks. RNR Technical Report RNR-94-007, March 1994

[27] Amazon Web Services Glossary (Ver- sion 1.0), AmazonElastic Cloud (EC2), http://docs.amazonwebservices.com/general/latest/gr/rande.html#ec2 region

[28] Amazon EC2 Instance Types, http://aws.amazon.com/ec2/instance-types/, last retrieved on 12 April 2012

[29] insidehpc.com, ”ISC launches hosted HPC service”, 2008, http://insidehpc.com/2008/04/01/isc-launches-hosted-hpc-service/

Leila Ismail is currently an Assistant Professor of Computer Science and Engineering at the College of Information Technology, UAE University. She served as an Assistant Professor at the American University of Beirut during 2004-2005. Before that, she worked for Sun Microsystems Research and Development Center in Grenoble, France, where she participated to the deposit of a patent in the domain of high available network. Dr. Ismail received her PhD in Computer Science and Engineering from the National Polytechnic Institute of Grenoble (INPG), France, in September 2000 with very honorable degree. She conducted her research work in Distributed Systems at the French National Institute for Research in Computer Science and Control (INRIA, Grenoble). She completed her higher studies of DEA at the Joseph Fourier University/ENSIMAG Engineering School in France. Dr. Ismail has an active commitment to research in High Performance Computing, Distributed Systems, Cluster, Grid and Cloud computing, Autonomous Computing, Parallel Processing Programming Models and Middleware for Mobile Agents. She is the founder and the director of the High Performance and Grid and Cloud Computing Research Laboratory at the UAE University. Dr. Ismail is the winner of several research awards including the IBM Shared University Research Award (SURA) in 2007 and the IBM Faculty Award in 2008, very competitive awards world-wide. She is also the winner of a UAE national competition condducted by the National Research Foundation and her project is ranked Highly Competitive by anonymous peer international external reviewers. She is an active referee for several international journals and actively participating in conferences programs committees and organizations. She was General Chair for the IEEE- DEST 2010.