Migration of tools and methodologies for performance prediction and efficient HPC on cloud environments: Results and conclusion *

Ronal Muresano, Alvaro Wong, Dolores Rexachs and Emilio Luque
Computer Architecture and Operating System Department (CAOS)
Universitat Autònoma de Barcelona, Barcelona, SPAIN
rmuresano@caos.uab.es, alvaro@caos.uab.es, dolores.rexachs@uab.es, emilio.luque@uab.es

Abstract— *Progress in the parallel programming field has allowed scientific applications to be developed with more complexity and accuracy. However, such precision requires greater computational power in order to be executed. However, updating the local systems could be considered an expensive decision. For this reason, cloud computing is emerging as a commercial infrastructure that allows us to eliminate maintaining the computing hardware. For this reason, cloud is promising to be a computing alternative to clusters, grids and supercomputing for executing these applications. In this sense, this work is focused on describing the manner of migrating our prediction tool PAS2P (parallel application signature for performance prediction), and how we have to analyze our method for executing SPMD applications efficiently on these cloud environments. In both cases, cloud could be considered a huge challenge due to the environment virtualization and the communication heterogeneities, which can seriously affect the application performance. However, our experimental evaluations make it clear that our prediction tool can predict with an error rate lower than 6,46%, considering that the signature for prediction represents a small portion of the execution time. On the other hand, analyzing the application parameters over the cloud computing allows us to find through an analytical model, which is the ideal number of virtual cores needed to obtain the maximum speedup under a defined efficiency. In this case the error rate was lower that 9% for the application tested.*

Keywords: Performance, PAS2P, Prediction, SPMD, Cloud.

1. Introduction

The constant evolution of the parallel computing field has permitted those scientific applications to be designed with more complexity and precision. However, these applications need to be executed with high computational power in order to obtain an improvement of parallel performances. One solution is to update our system by increasing the number of processing element but we have to consider that this is an expensive decision for both acquiring and maintaining the system. A second solution is to use a transparent architecture, which integrates the computational resources needed to execute the applications. In this sense, the cloud environment is an architecture which is promising to be a computing alternative to clusters, grids and supercomputing for executing these scientific applications. Initially, this emerging infrastructure-provider segment has been generally focused on business users and hosting web applications and services, but currently some researchers have begun to look at the cloud as a viable solution for scientific computing, especially in high performance computing [1] [2]. In this sense, cloud computing allows the user to define their computational resources according to the application characteristics.

However, executing a parallel application using these cloud environments could present a huge challenge, especially if you are trying to predict the execution time and also if you wish to execute faster and more efficiently [3]. These challenges can be addressed or caused by the communication heterogeneity or the different computational instances defined in each cloud such as: EC2 amazon, smart cloud IBM, bonfire, etc. Hence, this work is based on how to migrate our prediction tool PAS2P (parallel application signature for performance prediction), and how we have to analyze our method for executing efficiently SPMD applications on these cloud environments.

The first step is to evaluate the prediction quality obtained using PAS2P, which is a toolset to automatically extract the most significant behavior (phases) of parallel applications, into a parallel application signature. By its execution on different parallel computers, the performance of the application can be predicted. The accurate prediction of the performance of parallel applications is becoming increasingly complex and the time required to run it thoroughly is an onerous requirement; especially if we want to predict for different systems. Then, PAS2P is capable of instrumenting a binary and collects a set of phases by using interposition of functions. This process allows us to reduce the log trace size instrumenting the communication and computation events in order to be executed in a target machine, in this case the cloud architecture. Then, this signature can be executed on different target machines allowing for the signature to measure the execution time of each phase. Finally, the signature of the application allows us to predict the entire

* This research has been supported by the MEC-MICINN Spain under contract TIN2007-64974

application's run time (with an average accuracy above 98%) in each of those cluster tested by extrapolation of each phase's execution time using the obtained weights. The execution time of the application signature is a small fraction (less than 2%) of the whole applications runtime [4]. However, we have to evaluate if we obtain the signature in a based machine and we execute that signature in a cloud, we can predict with the same prediction relationship that we have obtained in a cluster.

On the other hand, the second step is how we have to analyze the application in order to predict the number of virtual core with the aim of obtaining the maximum speedup under a defined efficiency for a SPMD applications. In this case, we start with the method defined in [5] and [6], where we can predict the processing element using a communication heterogeneus environments such as multicore. The SPMD paradigm was selected due to its behavior, which is to execute the same program in all processes but with a different set of tiles. These tiles have to exchange their information in each iteration and these can become a huge issue when we use a heterogeneus communication, such as integrated, within the cloud environment. To solve these inefficiencies, we have developed a method that manages the communication latencies using some characteristics of each SPMD application (e.g. computation and communication tile ratio) and allows us to determine a relationship between scalability and efficiency. To achieve this performance relationship, our methodology is organized in four phases: characterization, tiles distribution model, mapping strategy, and scheduling policy, which allow us to distribute the tile inside the environment. The main idea is to evaluate the environment characteristics of the cloud, and how the application behavior is within these architectures.

Finally, this article shows how we have to evaluate both PAS2P and the method for efficient execution in order to achieve the migration steps to be applied in this new trend for high performance computing using cloud. In this sense, the experimental evaluation performed has illustrated how our methods can predict with a small error rate considering the variation of these cloud environments.

This paper is structured as follows: section 2 illustrates the impact of cloud over the HPC applications, then it is followed by section 3 where the PAS2P methodology is described. The section 4 explains the method for efficient execution of SPMD applications on cloud. Then the experimental evaluation is illustrated in section 5. Finally, the main conclusions are described in section 6.

2. High performance application on cloud environments

Currently, cloud computing is considered as an important paradigm for managing resources distribution and its infrastructure is now widely used in many domains, but one area where there has been more limited adoption is research computing, in particular for running scientific high-performance computing [1]. This is due to the uncertain scenarios in the communication and computation of the architecture. For this reason, it is very important to develop and to migrate tools and methods designed for HPC in computer clusters in order to understand the behavior of this complex architecture.

In this sense, the cloud computing has to deal with different challenge that administrators and users have to solve or manage with the aim of taking advantage of this virtualized architecture. Under this focus, we have to analyze system performance when an HPC application is used. One of the complex problems on cloud is the poor network performance, which can degrade the metrics such as efficiency and execution time of the scientific applications. These communication problems are increased by the virtualization overhead and the networking setup itself. Also, the problem increases when we execute scientific applications under the SPMD paradigm, which have to exchange information between neighbors. Another element is the real heterogeneity of the system because the cloud instances are set up using generic values for the architecture. This can result in instances reserve, which are not using identical configuration as can be presented in a homogeneous cluster. These issues can create imbalance problems which the users have to consider.

Moreover, we have to consider the noise of other process which can affect the performance of HPC applications. One example can be described, when an HPC application is executed over a real machine of 8 cores and we reserve instances with 4. The others cores can be accessed by other applications which can use resources like a communication network card, memory bandwidth, etc.

Despite all these challenges, cloud computing is certainly attractive to HPC users. Indeed, in many cases, users cannot get enough cycles on existing systems and Cloud HPC would be a viable economic alternative to purchasing more hardware in order to execute more complex scientific applications. Also, HPC facilities may not grow at the same pace as ever-growing computational demands, or they could be limited by local power supply. Instead of rejecting users' applications on their own private clusters, cloud is an alternative to execute application considering that this solution is not only economically feasible but it can also reduce the time to solution for scientific application programmers [7]. Moreover, cloud offers the benefits of virtualization, elasticity of resources and cluster setup for HPC

Therefore, cloud is an infrastructure, which has been designed under the concept of on paying for the resources used. This is an important advantage for the user and even more so when they can determine the ideal number of resources needed for executing the application. Under this focus, it is very important to migrate the prediction tools

in order to define the time for renting the resources. In this sense, our PAS2P tool extracts a signature of the real application and it allows us to predict the execution time with a very small error rate. One of the important aspects is that the signature can be extracted in a private machine and then it can be executed on the cloud architecture in order to predict the execution time of the scientific application for a determined number of MPI processes (message passing processes) and workload. This signature can predict the execution time of the application in order to considerably reduce the renting time.

On the other hand, one of the challenges is to take advantage of the resources on cloud. This allows us to evaluate our method for efficient execution of one paradigm with high communication volume how is presented on SPMD applications. This method permits us to determine the ideal workload and number of virtual cores needed to execute considering the cloud characteristics (computation and communication evaluation). This orientation will help us to migrate our method for a heterogeneous communication architecture where the communication links can present huge delays. The migration of both tools is a considerable advance for the trend of HPC on cloud.

3. Parallel Application Signature for performance Prediction PAS2P

Applications typically possess highly repetitive behavior and parallel applications are no exception. PAS2P makes an analysis to characterize the computational and communications-related behavior of parallel applications by identifying these repetitive portions. It is important to notice that this is a methodology with two main steps. 1.

The first step is to analyze the application, build the application model to extract its phases and weights, and use that information to build the signature, which is an executable that contains the relevant phases with instrumentation, in order to have information about their behavior and their weights to predict the application performance on the target machines (Fig 1, Instrumentation, analyzer and signature generation modules).

The second step is to execute the signature in a target system, to measure the execution time of each phase and predict the execution time of the application (Fig 1, Performance prediction module).

3.1 Application analysis and signature construction

In order to obtain the behavior of computation and communication, the application is instrumented on a base machine in order to intercept and collect communication events of the parallel application. With this collected data an application trace log is generated. The communication events are ordered by means of a logical global clock according

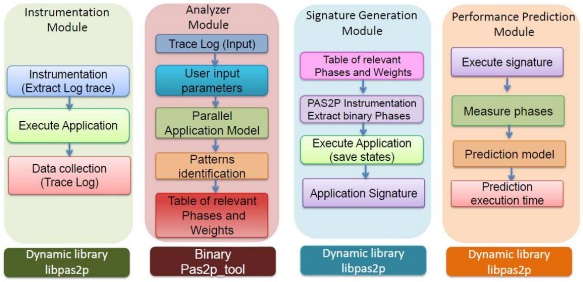

Fig. 1: Modules of PAS2P tool

to causality relations between communication events. The machine-independent application model can be obtained from this trace. Once we have the application model, the methodology strives to identify the application patterns in order to find a representative behavior of the application.

It is processed using a technique that searches for similarity to identify and extract the most relevant event sequences (phases) and assign them a weight based on the number of times the phases occur. Afterwards, in order to construct the signature, the last step is to re-run the application to create the coordinated checkpoints before each relevant phase happens. Therefore, the executable signature will be defined by a set of relevant phases and their weights.

3.2 Performance prediction model

Once we have constructed the application signature, we can run it on real target machines to analyze the application behavior and predict the application execution time. In order to execute the phases, we restart the checkpoints of the application before the phase begins and measure its execution time until the phase ends. To predict the application execution time equation 1 is used, where PET is the Predicted application Execution Time, n is the number of phases, TEPhasei is the Phase i Execution Time and Wi is the weight of the phase i.

Due to the complexity of the process and the huge quantity of information obtained during the analysis, we decided to automatize the methodology, allowing users to apply the whole methodology in an automatic and transparent way. The next section explains how the methodology was automated.

$$PET = \sum_{i=1}^{n} (TEPhasei)(Wi) \quad (1)$$

4. Methodology for efficient execution of SPMD applications

Our methodology is focused on managing the communications heterogeneities presented on hierarchical communication architecture, such as multicore clusters, multiclusters, cloud environments, etc. This process is realized through four phases which allow us to handle the latencies and the communication imbalances created due to the different

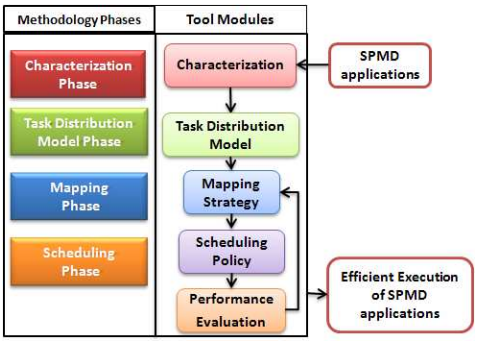

Fig. 2: Methodology phases and tool modules

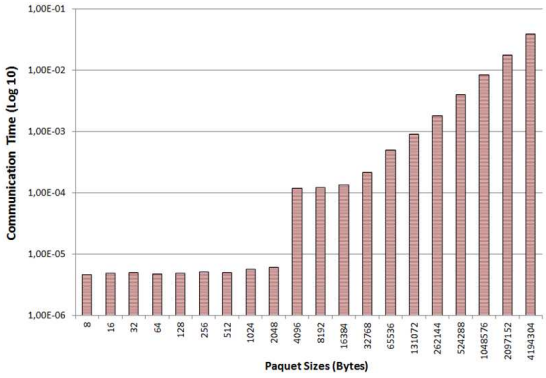

Fig. 3: Communication characterization on IBM smart cloud

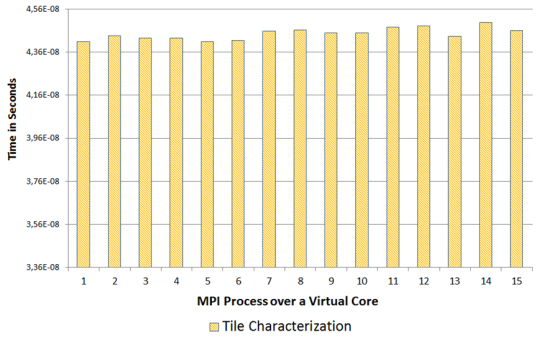

Fig. 4: Computation analysis on diverse virtual cores

communication paths. The phases defined in our methodology permit us to accomplish our objective of finding the maximum speedup while the efficiency is maintained over a defined threshold. These phases are integrated in five modules of a framework with the aim of improving the performance (Figure 2).

The latencies and the imbalance factors also have to be handled with the objective of removing the inefficiencies generated by communication links. These latencies generate idle time for different reasons, such as: tasks communication when processes are located in different processor chips or nodes, communication message size, bottlenecks in the communication paths, SPMD data synchronization, adaptation of an MPI application designed to be executed in single core nodes for multicore nodes, etc. The idle time generated decreases the performance, particularly efficiency and speedup. Our methodology, through its phases, can solve the inefficiencies generated in these communications links as was described in [5].

This method has been migrated to the cloud environment. To achieve this, we have considered two main aspects. The first one is that all instances have to be computationally equal in order to maintain homogeneity in the virtual cores and the second one is that we have to consider that communication cannot be controlled. These communication issues can be a part of the main problem of migrating our framework to cloud environment. Therefore, we will give a brief description of the method migrated to cloud environment by phases:

4.1 Characterization

This phase is focused on performing an application and environment analysis with the aim of obtaining the application parameters which are used to calculate the analytical model. The main idea is to find the nearest relationship between the cloud environment and the SPMD application. The parameters are classified in two groups: the application parameters and parallel environment.

The parameters determined allow us to establish the communication and computational ratio time of a tile inside of the hierarchical communication architecture. This relationship will be defined as $\lambda(p)(w)$, where p determines the link where the communication of one tile to another neighboring tile has been performed and w describes the direction of the communication processes (e.g. Up, right, left or down in a four communications pattern). This ratio is calculated with equation Eq. 2, where $Commt(p)(w)$ determines the time of communicating a tile for a specific p link and the Cpt is the value of computing one tile on a virtual core. This characterization process has to be done in a controlled and monitored manner.

$$\lambda_(p)(w) = CommT(p)(w)/Cpt \quad (2)$$

An example of this characterization can be found in figures 3 and 4. This characterization has been done using the IBM smart cloud using the silver instances [1]. As can be seen in figure 3, communications have a considerable increment where from 8 bytes to 2 KB in regular and then the time present a considerable increment around one order of magnitude in differences. These variances have to be considerered, when we analyse the tile size and its communication value.

On the other hand, the computation parameters illustrated in figure 4 allows us to conclude that if we choose the same instances, we can obtain a homogeneous environment although we are using an uncertain one. As can be detailed in figure 4, the computation times in different virtual cores are the same. Both results allow us to apply our analytical model.

[1]Silver instances are composed by 4 virtual core and 8 Gb virtual memory

4.2 Tile distribution Model

Once the parameters in the characterization are obtained, the next step is to calculate the ideal number of virtual cores and problem size in order to maintain the relationship between efficiency and speedup. To achieve this, we have introduced the concept of supertile (ST). An ST is a unit which integrates a set of tiles where these tiles are divided in two types; internal and edge. The problem of finding the optimal ST size is formulated as an analytical problem, in which the ratio between computation and communication of the tile has to be found with the objective of improving the relationship between efficiency and speedup.

The main idea of these STs is to create a structure which is assigned one per virtual core. These STs manage the communication heterogeneity of the cloud environment and also eliminate communication wasting time of parallel execution. This method takes advantage of the communication time assigning more computation tiles and hiding the communication effects of the cloud environment. The division of STs among internal and edge allow us to apply an overlapping technique, where the internal computation time is overlapped while the edge communication is performed. The ST size is calculated considering the slowest communication path, allowing us to manage the communication between all links in the hierarchical communication architecture.

4.3 Mapping phase

The main purpose of this phase is to apply a distribution of ST in the execution virtual core. In cloud the ST assignations are made applying a cartesian map of processes with the aim of minimizing the communications latencies. This map will determine where the processes has to be allocated and how the ST have to be assigned to each virtual core. However, the ST assignations should maintain the initial considered allocation used in the characterization phase.

This phase is divided in three key points. The first point performs a logical processes distribution of the MPI processes. The second function is to apply the core assignation, and the last one is the division and distribution of the STs. The mapping has to divide the tiles in order to create the ST considering the value of K obtained by the analytical model. It's important to understand that an incorrect distribution of the tiles can generate different application behaviors.

4.4 Scheduling phase

The main function is to assign an execution priority assignment to each tile with the aim of applying the overlapping strategy. This process establishes the highest priorities for tiles which have communications through slower paths and slower priority to internal tiles. This phase performs an overlapping strategy, which allows us to hide the communications effects as can be detailed in figure 5.

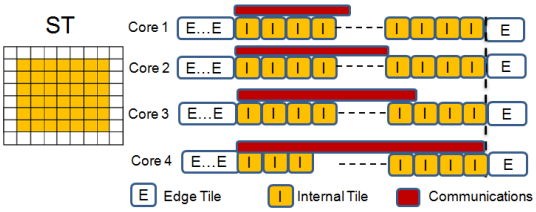

Fig. 5: Scheduling for hiding the communication effects

5. Experimental Validation

In order to test the migration of the PAS2P tool and the efficient framework for executing SPMD applications, we have tested using two cloud environments; Amazon EC2 [8] and IBM smart cloud [9]. The signatures of PAS2P have been extracted using the NOVA cluster with 4 Intel Xeon quad-core E7350 2.66Ghz, Tigerton Processors L2 cache 2x4 MB, 48 GB DDR2 SDRAM, ConnectX IB Mellanoxcard, and the execution was in the Amazon EC2 cloud with 8 instances EC2 of 4 virtual cores with 15 Gb of memory and 690 GB of storage. The network also was 1000 Mbps. For the framework we have used the IBM smart cloud using 4 instances silver with a 4 virtual core and 8 Gb of RAM memory. The applications used were the NAS parallel benchmark suite and the heat transfer application.

5.1 PAS2P validation

This section validates that the prediction methodology using the signature works in system like cloud. We can obtain the application prediction at a high level of precision in a short time (Signature Execution Time). We show the signature execution for each application on Nova cluster and Amazon cloud cluster. We predict their execution times and demonstrate the prediction quality of each signature.

The methodology used to obtain the results involves executing the applications on Nova cluster in order to analyze and extract the phases of the application. With the phases, we construct the signature in order to predict the AET (Application Execution Time) using the Nova cluster as the target machine. We applied the PAS2P methodology to the above applications to extract phases and obtain the application signatures. After running the signatures from all applications, we now know the execution time for each phase and the Signature Execution Time (SET), which is the sum of the execution times of all constituent phases. However, to obtain the Predicted Execution Time (PET), we multiply the execution time of each phase by the weight vector given by the PAS2P and add the times obtained.

In Table 1 shows the results from Nova cluster. We execute CG, LU, SP from NAS Parallel Benchmarks with a different number of processes. As this table indicates, when we compare columns 2 (SET) and 4 (AET), it can be seen that the SET is notably shortened compared with the AET. Column 3 shows the Predicted Execution Time (PET) given by the signature being executed. Column 5 presents the Prediction Execution Time Error (PETE) lower than 4.4%.

Table 1: Predictions on cluster NOVA

Program	SET (Sec.)	PET (Sec.)	AET (Sec.)	PETE (%)
CG.B.8	0.43	37.71	38.99	3.27
LU.B.8	3.00	71.01	67.96	4.49
BT.B.9	4.73	95.31	95.87	0.58
SP.B.9	5.70	235.40	234.39	0.43
BT.C.64	3.59	72.32	72.68	0.49
SP.C.64	3.09	121.58	118.06	2.97
CG.C.8	2.04	242.40	245.59	1.30
BT.C.9	23.87	492.49	494.76	0.46
SP.C.9	25.81	1065.18	1067.8	0.25

Table 2: Predictions on Amazon EC2 cloud

Program	SET (Sec.)	PET (Sec.)	AET (Sec.)	PETE (%)
CG.B.8	1.0233	81.79	84.49	3.20
LU.B.8	4.4379	97.08	98.52	1.45
BT.B.9	10.2806	150.53	155.24	3.03
SP.B.9	10.4252	419.83	419.86	0.01
BT.C.64	6.4208	151.94	156.34	2.80
SP.C.64	5.9997	287.18	277.40	3.52
CG.C.8	2.6945	262.49	260.39	0.80
BT.C.9	39.1058	792.68	791.77	0.11
SP.C.9	42.0293	1706.69	1701.40	0.31

Therefore, we can transfer the signatures from the cluster Nova to the cloud cluster. Table 2 shows the execution of the signature in the cloud. We execute the signature in order to get the PET. In order to validate the PET in the cloud, we execute the whole application to compare with the PET and discover the PETE where the maximum error is 3.5%. We can notice that the signature constructed in a base machine (Nova cluster) can be used to predict the performance in cloud systems.

In Figures 6 and 7, we show a comparison between the Signature Execution Time vs. the Application Execution Time, where we demonstrate that signature represents a small fraction of the whole application execution.

Fig. 6: SET vs. AET on Nova cluster.

Finally in Table 3 we show that the signature can predict the performance using different mapping policies. In this case we execute the signatures created with 64 processes with 16, 32 and 64 cores. The column SET shows how the signature execution increased when the mapping police

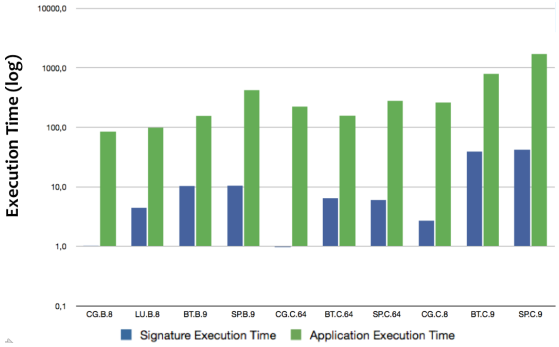

Fig. 7: SET vs AET on Cloud.

changes and AET increased due to the process of mapping the signature when the number of cores is reduced. Scientists can use the signature to provided performance estimations for their applications on cloud systems, they could more efficiently choose resources for their applications.

Table 3: Signature exec. with different mapping on Cloud

Program	Cores	SET (Sec.)	PET (Sec.)	AET (Sec.)	PETE (%)
BT.C.64	16	25.0469	472.26	492.63	4.14
	32	10.8577	236.51	252.84	6.46
	64	6.4208	151.94	156,34	2.80
SP.C.64	16	18.7724	890.63	883.48	0.80
	32	9.8615	461.43	460.35	0.23
	64	5.9997	287.18	277.40	3.52
FT.C.64	16	70.9388	239.48	235.83	1.54
	32	46.3466	159.51	157.34	1.37
	64	29.9754	101.09	111.88	9.65

5.2 Methodology for efficient execution validation

In order to probe the effectiveness of our method predicting the ideal ST which maintains the relationship of efficiency and speedup, we have used the heat transfer application. This is a finite difference problem that is solved using the SPMD paradigm. As a first experiment, we have defined the problem size as 5160 using 10000 iterations and we have tested in two different instances time (Scenario A and B Table (4)). As can be seen in table 4, the characterization values obtained are different. These results could be due the system in some moments being shared with another process. However, for our method it is transparent. An important aspect is to characterize every time before executing your SPMD applications. In this case, a summary of the data obtained in the characterization step for both scenarios A and B are illustrated in table 4.

Table 4: Heat transfer application analysis

Scenario &hline A	Problem 5160	Effic 100%	TileComp 4,10E-08	TileComm 5.27E-05	λ 1285
B	5160	100%	2,6E-08	5.27E-05	2026

Once obtained these values, we have to find the ideal values applying the model defined in [6]. Our model allows us to find the ideal ST size which maintains the overlapping strategy between internal computation and edge communications. Table 5 illustrates the theoretical values obtained using our model and also it defines the ST size and the ideal number of virtual cores needed. The model also give us the approximate execution time, which in cloud is a very important key for defining the time to rent the computing instances.

Table 5: Execution Model Heat transfer app.(Time in Sec)

Scen.	ST	Comp(Iter)	Comm(Iter)	Exec T	Cores
A	1289	0,067	0,067	681	16
B	2030	0,106	0,106	1071	8

Then, the next step is to execute the application using the number of virtual cores obtained in table 5. As can be shown in figure 8, we can obtain an execution with an efficiency of around 100% for the scenario A. In this case, the error rate obtained is lower than 9%. Similarly, figure 9 for scenario B shows that the ideal value for efficiency is around the ideal value obtained using the model. The error rate in this case is lower than 8,5 % for the ideal case and then the efficiency is going down considerably.

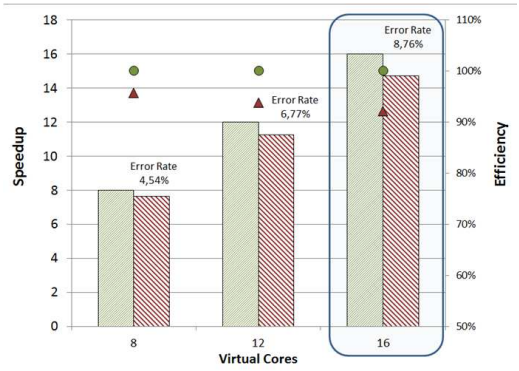

Fig. 8: Heat transfer application (Scenario A)

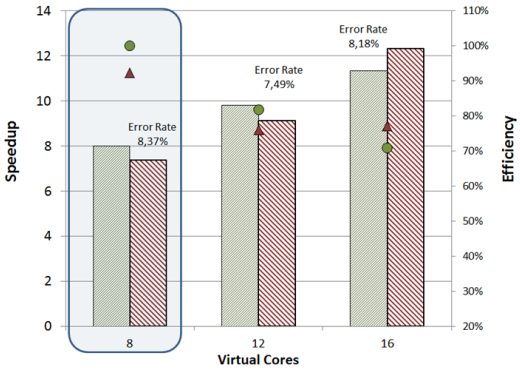

Fig. 9: Heat transfer application (Scenario B)

As has been demonstrated, our method can be migrated to the cloud. Only, we have to consider that characterization must be done every time that we turn off the instances. The issue is that cloud cannot guarantee the same machine. They try to maintain some default characteristics in their virtual environments.

6. Conclusions and future works

This work addresses how we can migrate our tools and frameworks developed to be used in a cluster to cloud enviroments. In this sense, we have started with PAS2P, where we have observed how the prediction maintains the same quality level when these cloud environments are used to execute parallel applications. Experimental validation has also shown that PAS2P can predict with an error rate below 6,5% using these virtualized environments. Similarly, our framework for executing SPMD applications has been adapted to predict and execute efficiently on a cloud environment. In this case, the error rate is below 9% without making any change to the original method developed for multicore clusters.

As was observed, both methodologies can be used to predict and execute efficiently on cloud environments. The migration of our tools to cloud will allow us to use these virtualized environments using HPC in an efficient manner and with high presicion, when we will rent the resources for executing parallel applications.

References

[1] Z. Hill and M. Humphrey, "A quantitative analysis of high performance computing with amazon's ec2 infrastructure: The death of the local cluster?" in *Grid Computing, 2009 10th IEEE/ACM International Conference on*, 2009, pp. 26–33.

[2] R. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "An early performance analysis of cloud computing services for scientific computing," Tech. Rep., 2008.

[3] K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, 2010, pp. 159–168.

[4] A. Wong, D. Rexachs, and E. Luque, "Pas2p tool, parallel application signature for performance prediction," in *Proceedings of the 10th international conference on Applied Parallel and Scientific Computing - Volume Part I*, ser. PARA'10. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 293–302.

[5] R. Muresano, D. Rexachs, and E. Luque, "Methodology for efficient execution of spmd applications on multicore environments," *10th IEEE/ACM Int Conf on Cluster, Cloud and Grid Comp, CCGrid 2010, Australia*, pp. 185–195, 2010.

[6] ——, "A method for scaling spmd applications on multicore clusters," in *In proceeding of: 2012 International Conference on Parallel and Distributed Processing Techniques and Applications. PDPTA, Las Vegas*, 2012.

[7] A. Gupta and D. Milojicic, "Evaluation of hpc applications on cloud," in *Proceedings of the 2011 Sixth Open Cirrus Summit*, ser. OCS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 22–26.

[8] A. EC2. (2013, May) Amazon ec2 instances. [Online]. Available: http://aws.amazon.com/es/ec2/instance-types

[9] IBM. (2013, April) Ibm smart cloud. [Online]. Available: http://www.ibm.com/cloud-computing/us/en/