Parallel Computing: Technology Trends
I. Foster et al. (Eds.)

doi:10.3233/APC200026

Cloud vs On-Premise HPC: A Model for Comprehensive Cost Assessment

Marco FERRETTI and Luigi SANTANGELO
Department of Electrical, Computer and Biomedical Engineering
University of Pavia, Italy

Abstract. Cloud Computing has emerged as an interesting alternative for running business applications, but this might not be true for scientific applications. A comparison between HPC systems and cloud infrastructure not always sees the latter winning over the former, especially when only performance and economical aspects are taken into account. But if other factors, such as turnaround time and user preference, come into play, the landscape of the usage convenience changes. Choosing the right infrastructure, then, can be essentially seen as a multi-attribute decision-making problem. In this paper we introduce an evaluation model, based on a weighted geometric aggregation function, that takes into account a set of parameters, among which job geometry, cost, execution and turnaround time. The notion of user preference modulates the model, and allows to determine which platform, cloud or HPC, might be the best one. The model has then been used to evaluate the best architecture for several runs of two applications, based on two different communication models. Results show that the model is robust and there is a not negligible number of runs for which a cloud infrastructure seems to be the best place for running scientific jobs.

Keywords. cloud computing, HPC, workload, cost-benefit analysis, turnaround time

1. Introduction

Cloud vs on-premise HPC for scientific applications is a long-standing debate [1–5], that has been tackled from many viewpoints, including the cost-perspective [6–8]. Contrary to widespread belief, a cost-benefit analysis comparing cloud infrastructures and HPC systems from the economical point-of view not always sees the cloud as the winner, unless applications allow for preemptible virtual instances. We got this result porting on the cloud two real applications (Cross Motif Search and BloodFlow) that have completely different patterns in their usage of computation and communication resources [9–12]. The former shows a simple master/worker communication model and is therefore less prone to the cloud inefficiency in message routing; the latter instead depends heavily on efficient point-to-point and collective communication primitives. Results show that neither from the performance perspective nor for the economical point-of-view, cloud seems to be a convenient place for running scientific application. But such a comparison is not fair as it does not take into account any factor which make Cloud so appealing. Indeed, just taking into account the turnaround time, the landscape of the usage convenience of the cloud computing changes. Building an evaluation model might help researchers to understand which platform might be the best one depending on the user preference, the execution time, the cost for computing and the expected waiting time in the queue. To build such a model, a characterization of the workload of a real HPC system needs to be done in order to understand the job waiting time depending on the job geometry (job size, amount of memory, maximum runtime), job failure, setup time and maintenance time.

Many previous works [13–15] have already characterized the workload of the HPC systems, but most of them aimed at evaluating the resource utilization and improve the scheduling algorithms to get the highest system utilization possible. Many others instead tried to predict the waiting time using machine learning techniques [16–19]. In our work, instead, we want to characterize the workload of an HPC system, named Marconi, in order to assess the job waiting time. Such time is then introduced in a utility function which is used to evaluate the best infrastructure (between Marconi and Google Cloud) for running both target applications.

The paper is structured as follows: in Sec. 2, we describe Cross Motif Search and BloodFlow applications and their communication models. Section 3 summarizes the performance results we got running Cross Motif Search and BloodFlow on two similar architectures, Marconi, an HPC system, and Google Cloud Infrastructure, showing that both from the performance and from the economical perspective the cloud lags behind the HPC system, justifying the reason to introduce the turnaround time as a factor to make a fair comparison. In Sec. 4, we describe all the parameters, such as the job waiting time and the virtual instance startup time, that should be kept into account for making a better evaluation of both infrastructures. Section 5 shows a characterization of the jobs submitted on Marconi, with a focus on the job waiting time. Section 6 measures the virtual instance startup time on the Google Cloud Platform for different configurations. Section 7 puts the resulting job waiting time and virtual instance startup time into a decision-making model which uses the weighted geometric aggregation function to build a utility function. Such function also takes the elapsed time measured running Cross Motif Search and BloodFlow on the cloud and on Marconi, and makes an evaluation of both platform in order to understand which run is executed more conveniently on the HPC infrastructure and which one on the cloud, taking into account performance, cost and user preference. According to our utility function, the best infrastructure might not be the one which minimizes cost or maximizes performance, but that which optimizes the user expectation. Section 8 concludes the paper.

2. The target applications

An efficient interconnection network is of paramount importance for getting a high performance in many scientific applications. The communication model embedded in the application and the underlying network infrastructure, are two major factors that should not be ignored during transition toward the cloud. For this reason, to study whether or not cloud computing can be considered convenient for running scientific applications, from the performance perspective as well as the economical one, we selected two different applications, respectively Cross Motif Search and BloodFlow, which are based on two different communication models. The first application is based on a master/worker communication pattern and the time spent in communication is very small if compared with

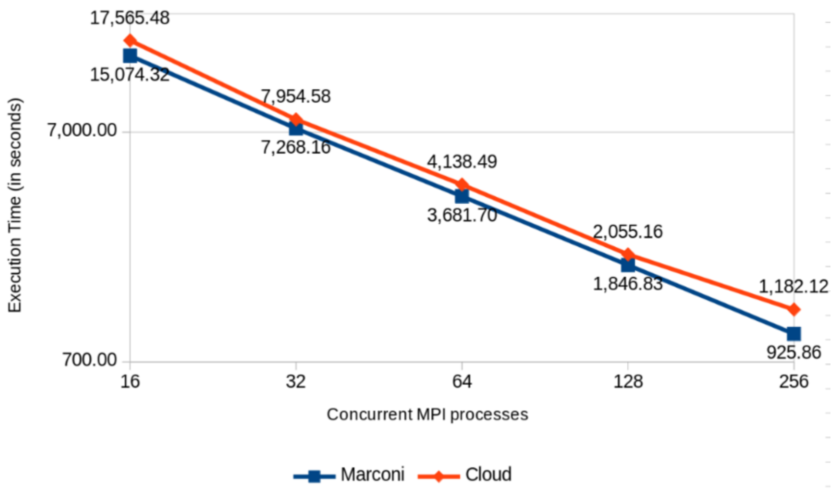


Figure 1. The CMS scalability on Marconi and on the cloud.

the whole elapsed time; the second one, instead, relies on a much more complex communication pattern, making extensive use of collective functions to scatter and gather data. Being based on two communication models which are opposite to each other, the two applications can be considered representative for a large subset of scientific applications. The following subsections describe both applications and their communication model.

2.1. *Cross Motif Search*

Cross Motif Search (CMS) [20] is a biological application which is able to look for recurring geometrical patterns in the secondary structures of proteins. The core algorithm relies on the generalized Hough transform [21] is used to find recurring geometrical patterns.

The last implementation of CMS [22] uses MPI standard to deliver messages across all processes. The communication model is very simple, as it is based on the traditional master/worker pattern. After starting the application, master and workers communicate to each other just using simple MPI primitives. Profiling activities [9] showed that the impact of the communication in the application performance is almost negligible if compared with the whole execution time. The last implementation of CMS [10, 23] was moved to the cloud in order to study its scalability and compare the cloud performance against the HPC performance. Figure 1 shows the application scalability for two similar architecture: an HPC system, named Marconi, and the cloud infrastructure provided by Google. The application showed a good scalability even on the cloud infrastructure as increasing the number of concurrent MPI processes, the time spent by the application is reduced, as on the HPC system. As the amount of messages exchanged among processes is not dependent on the number of concurrent processes, the scalability is good even increasing further the CPU number.

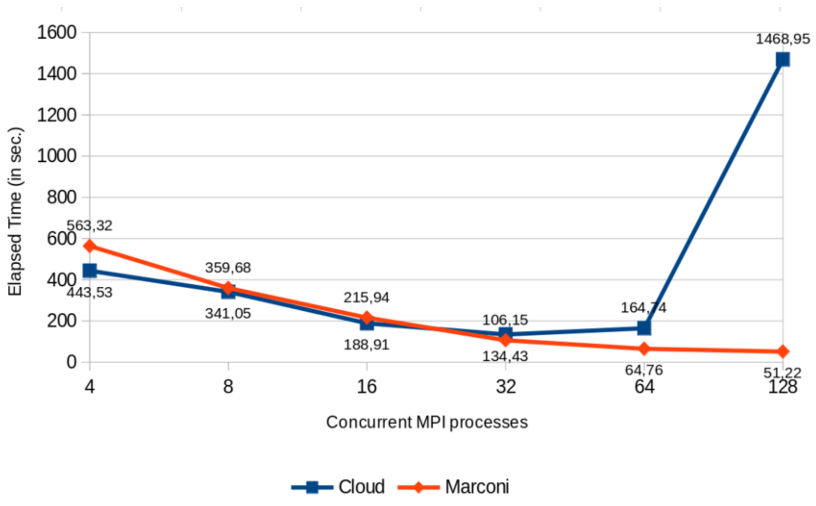


Figure 2. The BloodFlow scalability on Marconi and on the cloud.

2.2. *BloodFlow*

BloodFlow [24, 25] is a hemodynamics application which is able to run simulations of patient specific hemodynamics of an aorta through computational fluid dynamic analysis. The tool relies on a Navier-Stokes partial differential equation system, which is solved by using numerical approximations. A good description of BloodFlow can be found in [24, 26–28].

Profiling activities on BloodFlow [12] revealed that the application makes use of many MPI functions, point-to-point as well as collective, and that communication is a key factor which can affect application performance.

BloodFlow has been moved on the Cloud although the analysis we did on [10] revealed that BloodFlow might suffer if run on the cloud infrastructure, due to its huge amount of communication and the low network performance on the cloud system. Figure 2 compares the elapsed time measured running the application on both different architectures (Marconi and Cloud) and using different core numbers. The application seems to be able to perform well with a small number of concurrent processes, but when such number grows up the elapsed time becomes soon unmanageable and running the application on the cloud infrastructure is not convenient at all. A similar behaviour ensues even on Marconi, but at much higher core number (256-512).

3. Comparing performance and economical results

According to the results shown in figures 1 and 2, it is clear that CMS is able to scale very well even on the cloud but BloodFlow performs worse as it stops scaling at 32 cores, much before than on the HPC system. This comparison highlights that scientific applications based on a complex communication pattern, such as BloodFlow, might meet several troubles being run on the cloud while those applications based on simple communication model, like CMS, are good candidates to be executed on a cloud environment, because of the small impact of the interconnection network. In conclusion, cloud computing does not seem to be yet a convenient place for running scientific application at least from the performance perspective.

To understand if Cloud Computing can be convenient at least from the economical perspective, we estimated the cost for running a virtual instance on three different cloud platforms provided respectively by Google, Amazon and Microsoft, in order to compare it with the cost of using a similar configuration in the HPC environment. Each virtual instance in the cluster runs a Red Hat Enterprise Linux distribution and is equipped with 8 cores, 16 GB of memory RAM and 100 GB of Hard Disk. All virtual instances have also been created in a physical cluster based in London. Our analysis revealsed that Microsoft is slightly more expensive (0.53 dollars per hour) than the other two providers (0.41 dollars per hour) but Google wins the comparison as the Amazon billing policy is less convenient because, for example, the cost is computed by hours and not by seconds as in Google. All costs have been computed using the calculator tool made available from all three providers [29–31] and are valid as of March 2019. Even though Google is the cheapest solution, it is still more expensive than Marconi, where the cost per hour for 8 cores is two times lower than Google. And even using preemptible instances (that is, instances that can be stopped if other tasks require access to those resources) the cost, which is dropped by half, stays still higher than on Marconi. In conclusion, not even from the economical perspective cloud seems to be convenient, in the economical setting available at the moment of the experiments (cost estimation on Marconi was based on billing for commercial user).

4. The evaluation model

Looking at the results showed in the Section 3, it might sound that, for scientific researchers, cloud is a burden rather than an opportunity. This might be true if only performance and cost are taken into account. But if other factors [32–35] come in, comparison might yield different results. For example, as the jobs on HPC systems are not usually executed on-the-fly but put in a queue, they might experience a not negligible waiting time. Then, introducing the turnaround time (which is the sum of execution time and job queue delay) as further factor to compare HPC and Cloud systems, the landscape of the usage convenience changes. In our vision, a fair comparison between cloud and HPC infrastructures should take into account not just performance and economical aspects but also waiting time, job failure, job setup time, maintenance time as well as the user preferences. A time-sensible user might be willing to pay a bit more for getting the results sooner and then the chosen architecture will be different according to its preferences. Choosing the right infrastructure can be essentially seen as a multi-attribute decision-making problem. A proper model based on all these attributes might help researchers to understand which platform might be the best one depending on the user preference, the execution time, the cost for computing and the expected waiting time in the queue. The selected architecture might not be the highest performing one, nor the most affordable, but that one which maximizes the utility function describing the model.

To measure the effectiveness of each platform using several attributes, we devised a utility function based on the weighted geometric aggregation function. The attributes taken into account by the formula are user preference, execution time, core hour cost, expected waiting time in the queue for HPC system and virtual instance startup time for the cloud. Formulas 1 and 2 describe the utility function we adopt for measure the convenience to use the HPC infrastructure U_M or the Cloud system U_C for running any application.

$$U_M = \left(\frac{T_M + W_M}{\max\left(T_M + W_M, T_C + S_C\right)} \right)^{\lambda} * \left(\frac{C_M}{\max\left(C_M, C_C\right)} \right)^{(1-\lambda)} \tag{1}$$

$$U_C = \left(\frac{T_C + S_C}{\max\left(T_M + W_M, T_C + S_C\right)} \right)^{\lambda} * \left(\frac{C_C}{\max\left(C_M, C_C\right)} \right)^{(1-\lambda)} \tag{2}$$

In these formulas, T_M, W_M and C_M are respectively the elapsed time, the job waiting time and the cost for running the application on the HPC system, while T_C, S_C and C_C are the elapsed time, the virtual instance startup time and the cost for running the same application on the cloud. Parameter λ instead is the user preference. Its value ranges between 0 and 1. A value λ=0 means that the user is more sensible to the cost (and then the user would like to have the results at a lower cost, without being interested in the time to completion for getting such results); on the opposite side, λ=1 is the preference of a user who is mainly interested in minimizing the time to completion thus optimizing turnaround time. With this utility function, turnaround time comes in as a criterion for assessment. The model is validated by assessing all runs of CMS and BloodFlow applications on both Marconi and Cloud. If U_C is lower than U_M, users would choose Cloud as the preferred platform for running the applications, otherwise Marconi is the best choice.

As already mentioned, the utility function takes the time spent by the applications to be executed on the cloud and on Marconi and their relative costs. Furthermore, it requires the Waiting Time and the Virtual Instance Startup Time which need to be characterized. For this reason, in order to get both information, Sections 5 and 6 make a characterization of both times.

5. The workload analysis

The results described in this section refer to the jobs which have been successfully executed on Marconi A1 partition during eight months, from the 23rd of January up to the 26th of September 2018. During such observed period of time, the number of jobs submitted on Marconi A1 and successfully completed has been equal to 844,975. Half of all completed jobs terminated their execution in less than 43 seconds, while 80% stayed running for less than 1,400 seconds (almost 23 minutes). Only a negligible percentage of jobs (0.06%) took more than 24 hours to complete its execution, with a maximum elapsed time equals to 417,311.

5.1. Job Clusterization

Jobs on Marconi are submitted through Slurm [36]. Using Slurm, users can specify the task to run, the amount of required resources (number of cores and amount of memory), the wall time, which is the time the job might be left running the most, and finally the queue where the job has to be put on. A queue is not handled like a pure FIFO queue. Each job is assigned a priority index which is computed with a complex formula taking into account many factors such as the waiting time in the queue, the size of the job (core number and amount of memory), the required wall time and furthermore a fair share factor which slows down jobs submitted by users who have almost spent their monthly hours. Because of this scheduling policy, the waiting time spent by a job cannot be computed in advance and might be highly variable. Furthermore, as the queue is chosen by the user at submission time, each queue contains jobs having highly variable geometries, making the queue-oriented classification not proper to be used as a way to classify jobs according to their geometry and the time spent running. For this reason, in order to get sets of more homogeneous jobs, we decided to use k-means as partitioning method for job clusterization. Instead of fixing a priori the number of clusters, we iterated k-means method several times, until the covariance coefficient was lower than 1.1 for all clusters. The covariance coefficient *cc* for the *i-th* cluster is computed as in formula 3:

$$cc^i = \frac{sd(J^i_{ElapsedTime}) + sd(J^i_{CPUNumber})}{mean(J^i_{ElapsedTime}) + mean(J^i_{CPUNumber})} \tag{3}$$

where *sd* is the standard deviation function, *mean* instead is the mean function and J^i is the set of jobs belonging to the *i-th* cluster.

Our test showed that the ideal cluster number is 16, because using a smaller clusterization number makes the covariance coefficient higher the 1.1 at least for one cluster.

5.2. Job Waiting time

5.2.1. A global perspective

An overview on the waiting time of all jobs started on A1 partition shows that the time spent by each job is not negligible as it can last even several days. Indeed, 5.56% of the jobs had to wait at least 24 hours before being run. Only 19.21% instead has been executed almost immediately, while almost fifty percent of the jobs waited at least two minutes before being run.

5.2.2. Waiting time by clusters

We also did a characterization of the clusters we found using k-mean technique. The analysis revealed that the median waiting time for all clusters ranges between 3 seconds (cluster 7) and 92,094 seconds (cluster 1).

5.2.3. Relative Waiting Time by Clusters

As the main aim of this work is to characterize the waiting time the jobs might experience on an HPC system like Marconi, we studied how the relative waiting time changes inside each cluster. We define the relative waiting time *RWT* as follows:

$$RWT = \frac{WaitingTime}{ElapsedTime} \tag{4}$$

Table 1. Relative Waiting Time for all clusters

cluster	CPU Number interval	Elapsed Time interval (sec.)	median	mean	3rd quartile	max
cluster 1	1 - 4752	34387 - 62258	2.17	5.50	8.15	42.96
cluster 2	1 - 4176	1864 - 8176	0.07	6.21	2.54	248.46
cluster 3	1 - 1872	735 - 1234	1.48	15.84	6.91	1,015.34
cluster 4	1 - 512	282 - 534	2.51	29.57	9.74	1,461.78
cluster 5	1 - 216	200 - 335	1.88	25.82	12.24	2,528.10
cluster 6	2048 - 7488	1 - 6643	708.14	2,341.14	3,387.19	69,066.80
cluster 7	1 - 34	1 - 46	0.50	409.09	11.00	313,432.00
cluster 8	1 - 5760	6960 - 19590	0.23	3.38	3.51	133.06
cluster 9	1 - 5760	62227 - 417311	0.06	1.33	1.28	20.95
cluster 10	1 - 5760	18892 - 34880	1.12	5.27	5.84	64.79
cluster 11	1 - 180	103 - 208	0.76	28.31	5.45	4,120.61
cluster 12	1 - 1872	1035 - 2593	0.47	14.39	2.51	507.77
cluster 13	162 - 2088	1 - 920	0.80	381.70	4.60	37,7678.50
cluster 14	1 - 1024	473 - 803	3.40	37.16	27.68	1,809.86
cluster 15	1 - 72	39 - 117	1.80	99.38	12.53	20,702.60
cluster 16	30 - 162	1 - 111	1.50	284.07	13.87	166,114.00

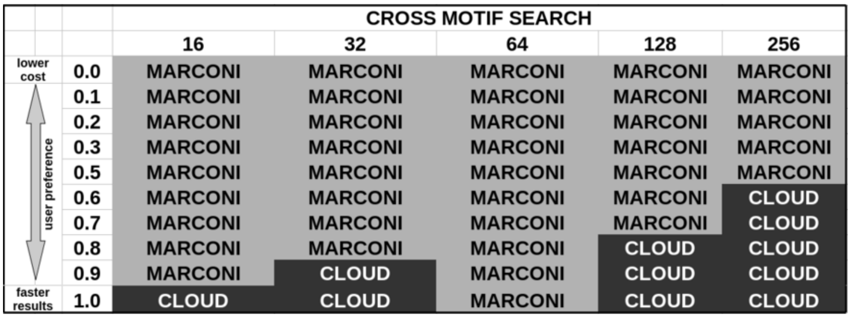
This value is always greater than or equal to 0. Better values are close to 0. The higher the relative waiting time, the greater the impact of the waiting time on the elapsed time. Table 1 shows the numerical values of such distribution. The mean value is almost always greater than the third quartile. This highlights that the distribution is badly affected by some outlier making the mean value and the maximum value much higher than the median value. For this reason, median value can better describe the waiting time because it is insensitive to the presence of outliers.

6. Virtual Instance Startup

Formulas 1 and 2 take into account not only the waiting time, which has been characterized in the previous section, but also the virtual instance startup time. Many works [37–39] have already studied Virtual Instance Startup Time and its relations with other factors such as the time of the day, operating system image size, instance type, data center location and number of instances requested at the same time. Neththeless, we checked virtual instance startup time for a number of cluster configurations typically used for the benchmark suite of applications we are interested in. The analysis covers startup time measured when activating a cluster of three virtual instances on a physical infrastructure hosted in the West US (us-west2-a). Each virtual instance is equipped with CentOS 7, 50GB of virtual disk and using different virtual hardware configurations (from 1 to 8 cores, from 3.75 to 30 GB of RAM). Times have been measured starting virtual instance from a custom tool written using the Google SDK. Our tests revealed that the median startup time is about 10 seconds.

Table 2. Cluster distribution for CMS and BloodFlow runs

CPU Number	**4 CPU**	**8 CPU**	**16 CPU**	**32 CPU**	**64 CPU**	**128 CPU**	**256 CPU**
CMS			8	8	2	12	3
BloodFlow	14	4	5	11	15	16	



		CROSS MOTIF SEARCH				
		16	32	64	128	256
lower cost	0.0	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI
user preference	0.1	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI
	0.2	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI
	0.3	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI
	0.5	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI
	0.6	MARCONI	MARCONI	MARCONI	MARCONI	CLOUD
	0.7	MARCONI	MARCONI	MARCONI	MARCONI	CLOUD
	0.8	MARCONI	MARCONI	MARCONI	CLOUD	CLOUD
	0.9	MARCONI	CLOUD	MARCONI	CLOUD	CLOUD
faster results	1.0	CLOUD	CLOUD	MARCONI	CLOUD	CLOUD

Figure 3. Preferred architecture for running CMS.

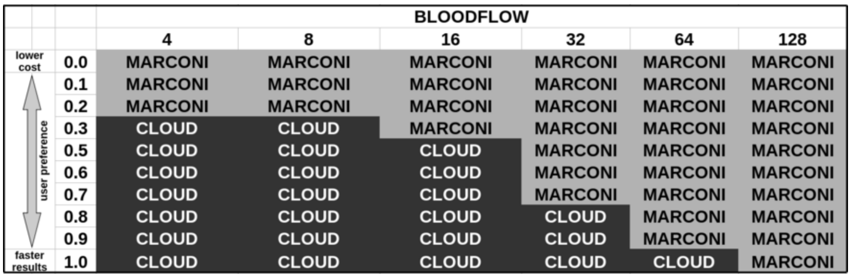
7. Applying the evaluation model on both applications

As described in Sec. 4, the utility function takes the time spent by the applications to be executed on the cloud and on Marconi and their relative costs. Furthermore, it requires the Virtual Instance Startup Time S_C and the Waiting Time W_M. The previous section gave us a measure of the time spent by a virtual instance to get ready to start the application, which can be fixed to 10. For the parameter W_M instead of using a single value on all runs, we decided to identify the cluster which each run might belong to, according to the job geometry. Then, the median relative waiting time *RWM* for the selected cluster is chosen as a factor to determine the waiting time used in the utility function. The job waiting time *WM* then can be easily computed as follows:

$$WM = RWM * TM \tag{5}$$

where *TM* is the job elapsed time. To be clearer, lets consider the first run of CMS. The application took 15,074.32 seconds using 16 cores. According to the job clusterization defined in table 1, the run might belong to the cluster number 8, where the median relative waiting time for all job in the cluster is 0.23. Then for this run, the estimated waiting time *WM* is equal to 15,074.32 * 0.23 = 3,467.09 seconds. The last run of BloodFlow, instead, took 51.22 seconds using 128 cores. Then this run belongs to the cluster number 16, having a relative waiting time equals to 1.50. Then for such run, the waiting time is equal to 76.83 seconds. Table 2 shows the cluster where each run for both applications belongs to.

Now, we have got all data needed to apply the utility function and assess the best platform. Figures 3 and 4 show which architecture might be the preferred for each run depending on the user preference. Light gray areas represent runs for which the cloud infrastructure is better. Looking at the table describing CMS, for 22% of all runs, the cloud Infrastructure seems to be the best architecture for running the application. For BloodFlow, instead, this percentage is higher, namely 48%.



		BLOODFLOW					
		4	8	16	32	64	128
lower cost	0.0	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI
user preference	0.1	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI
	0.2	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI	MARCONI
	0.3	CLOUD	CLOUD	MARCONI	MARCONI	MARCONI	MARCONI
	0.5	CLOUD	CLOUD	CLOUD	MARCONI	MARCONI	MARCONI
	0.6	CLOUD	CLOUD	CLOUD	MARCONI	MARCONI	MARCONI
	0.7	CLOUD	CLOUD	CLOUD	MARCONI	MARCONI	MARCONI
	0.8	CLOUD	CLOUD	CLOUD	CLOUD	MARCONI	MARCONI
	0.9	CLOUD	CLOUD	CLOUD	CLOUD	MARCONI	MARCONI
faster results	1.0	CLOUD	CLOUD	CLOUD	CLOUD	CLOUD	MARCONI

Figure 4. Preferred architecture for running BloodFlow.

The results described so far are heavily dependent on the relative waiting time introduced in our model. Indeed, if we suppose to rise the relative waiting time of cluster 2 (which is the cluster where the 64-core CMS runs belong in) from 0.07 to 0.40, the cloud preference for CMS rises from 22% to 28%. It is worth noting that the increase we introduced changing the relative waiting time from 0.07 to 0.40, is not negligible. In fact, supposing to have an elapsed time of 3,681.70 seconds (which is the real elapsed time CMS took being run on Marconi using 64 cores), changing the relative waiting time from 0.07 to 0.40, the waiting time goes from 258 seconds to 1,473. According to this observation, we can state that our model is robust, since a high perturbation of the relative waiting time brings a small variation in the convenience to use the cloud infrastructure rather than the HPC system. The results presented above also show that although cloud computing might be more expensive and less powerful than the HPC system, when turnaround time becomes important, cloud computing can be a convenient alternative for running scientific applications.

8. Conclusion

Studying the convenience to use Cloud Infrastructures as alternative to HPC systems for running scientific application is not easy as it should take into account many factors, not only related to the performance and economical aspect. Even the user preference plays an important role as some users might prefer to have results as fast as possible, others instead might wish spending less. In this paper we introduced a new model for the cloud convenience evaluation which takes into account performance, cost, user preference and waiting time. The model has then been applied to study the best architecture to run two different applications, based on two different communication models. Results show that our model is robust as high perturbations in the relative waiting time bring small variation in the results. Furthermore, there is a not negligible number of runs of both applications for which Cloud seems to be the better place, according to our evaluation mode.

References

[1] Rashid Hassani, Md Aiatullah, Peter Luksch, Improving HPC Application Performance in Public Cloud, In IERI Procedia, Volume 10, 2014, Pages 169-176, ISSN 2212-6678.

[2] Napper, Jeffrey, and Paolo Bientinesi. ”Can cloud computing reach the top500?.” Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop. ACM, 2009.

[3] Ramakrishnan, L., Canon, R. S., Muriki, K., Sakrejda, I., & Wright, N. J. (2012). Evaluating interconnect and virtualization performance for high performance computing. ACM SIGMETRICS Performance Evaluation Review, 40(2), 55-60.

[4] Jackson, K. R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., ... & Wright, N. J. (2010, November). 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 (pp. 159-168). IEEE.

[5] Zhai, Y., Liu, M., Zhai, J., Ma, X., & Chen, W. (2011, November). Cloud versus in-house cluster: evaluating Amazon cluster compute instances for running MPI applications. In State of the Practice Reports (p. 11). ACM.

[6] A. G. Carlyle, S. L. Harrell and P. M. Smith, "Cost-Effective HPC: The Community or the Cloud?," 2010 IEEE Second International Conference on Cloud Computing Technology and Science, Indianapolis, IN, 2010, pp. 169-176.

[7] T. Passerini, J. Slawinski, U. Villa and V. Sunderam, "Experiences with Cost and Utility Trade-offs on IaaS Clouds, Grids, and On-Premise Resources," 2014 IEEE International Conference on Cloud Engineering, Boston, MA, 2014, pp. 391-396.

[8] Nanath, K., & Pillai, R. (2013). A model for cost-benefit analysis of cloud computing. Journal of International Technology and Information Management, 22(3), 6.

[9] Ferretti, M., & Santangelo, L. (2018, September). Protein secondary structure analysis in the cloud. In Proceedings of the 6th International Workshop on Parallelism in Bioinformatics (pp. 63-70). ACM.

[10] Ferretti, M., & Santangelo, L. (2019, February). Profiling hemodynamic application for parallel computing in the cloud. In 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP) (pp. 41-50). IEEE.

[11] Ferretti, M., & Santangelo, L. (2018, March). Hybrid OpenMP-MPI parallelism: porting experiments from small to large clusters. In 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP) (pp. 297-301). IEEE.

[12] Auricchio, F., Fedele, M., Ferretti, M., Lefieux, A., Romarowski, R., Santangelo, L., & d Veneziani, A. (2018). Benchmarking a hemodynamics application on Intel based HPC systems. Paral Comput Everywhere, 32, 57.

[13] Dror G. Feitelson, Dan Tsafrir, David Krakov, Experience with using the Parallel Workloads Archive, Journal of Parallel and Distributed Computing, Volume 74, Issue 10, 2014, Pages 2967-2982, ISSN 0743-7315,

[14] Gonzalo P. Rodrigo, P.-O. stberg, Erik Elmroth, Katie Antypas, Richard Gerber, Lavanya Ramakrishnan, Towards understanding HPC users and systems: A NERSC case study, Journal of Parallel and Distributed Computing, Volume 111, 2018, Pages 206-221, SSN 0743-7315,

[15] S. Di, D. Kondo and W. Cirne, "Characterization and Comparison of Cloud versus Grid Workloads," 2012 IEEE International Conference on Cluster Computing, Beijing, 2012, pp. 230-238.

[16] Kianpisheh, Somayeh & Jalili, Saeed & Charkari, Nasrolah. (2012). Predicting Job Wait Time in Grid Environment by Applying Machine Learning Methods on Historical Information.

[17] Kumar R., Vadhiyar S. (2015) Prediction of Queue Waiting Times for Metascheduling on Parallel Batch Systems. In: Cirne W., Desai N. (eds) Job Scheduling Strategies for Parallel Processing. JSSPP 2014. Lecture Notes in Computer Science, vol 8828. Springer, Cham

[18] Andresen, D., Hsu, W., Yang, H., & Okanlawon, A. (2018). Machine Learning for Predictive Analytics of Compute Cluster Jobs. arXiv preprint arXiv:1806.01116.

[19] A. Matsunaga and J. A. B. Fortes, "On the Use of Machine Learning to Predict the Time and Resources Consumed by Applications," 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, VIC, 2010, pp. 495-504.

[20] Ferretti, M., Musci, M., & Santangelo, L. (2015). MPICMS: a hybrid parallel approach to geometrical motif search in proteins. Concurrency and Computation: Practice and Experience, 27(18), 5500-5516

[21] Ballard DH. Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognition 1981; 13(2): 111-122

[22] Ferretti, M., Musci, M., & Santangelo, L. (2014, September). A hybrid OpenMP and OpenMPI approach to geometrical motif search in proteins. In 2014 IEEE International Conference on Cluster Computing (CLUSTER) (pp. 298-304). IEEE.

[23] Ferretti, M., & Santangelo, L. (2019). Optimized cloud-based scheduling for protein secondary structure analysis. The Journal of Supercomputing, 1-22.

[24] Quarteroni, A., & Valli, A. (2008). Numerical approximation of partial differential equations (Vol. 23). Springer Science & Business Media.

[25] Formaggia, L., Quarteroni, A., & Veneziani, A. (Eds.). (2010). Cardiovascular Mathematics: Modeling and simulation of the circulatory system (Vol. 1). Springer Science & Business Media.

[26] Bertagna, Luca & Deparis, Simone & Formaggia, Luca & Forti, Davide & Veneziani, Alessandro. (2017). The LifeV library: engineering mathematics beyond the proof of concept.

[27] M. A. Heroux et al., An overview of the trilinos project, ACM Trans. Math. Softw., vol. 31, no. 3, pp. 397-423, 2005

[28] Bertagna, L., Deparis, S., Forti, D., Formaggia, L., & Veneziani, A. (2016), The LifeV library: engineering mathematics beyond the proof of concept, Tech Report Dept. Math & CS, Emory University, TR2016-008, www.mathcs.emory.edu

[29] Amazon Calculator. Retrieved July 5, 2019, from https://calculator.s3.amazonaws.com/index.html

[30] Azure Pricing Calculator. Retrieved July 5, 2019, from https://azure.microsoft.com/it-it/pricing/calculator/

[31] Google Cloud Platform Princing Calculator. Retrieved July 5, 2019, from https://cloud.google.com/products/calculator/

[32] Costa, Pedro & Santos, Joao & Mira da Silva, Miguel. (2013). Evaluation Criteria for Cloud Services. IEEE International Conference on Cloud Computing, CLOUD. 598-605. 10.1109/CLOUD.2013.70.

[33] Geeta, Prakash S. (2018) A Review on Quality of Service in Cloud Computing. In: Aggarwal V., Bhatnagar V., Mishra D. (eds) Big Data Analytics. Advances in Intelligent Systems and Computing, vol 654. Springer, Singapore

[34] J. Singh, S. Agarwal, J. Mishra, A review: Towards quality of service in cloud computing, International Journal of Science and Research

[35] Kumar, Rakesh & Kumar, Chiranjeev. (2018). A Multi Criteria Decision Making Method for Cloud Service Selection and Ranking. International Journal of Ambient Computing and Intelligence. 9. 1-14. 10.4018/IJACI.2018070101.

[36] Overview Slurm Workload Manager. Retrieved July 5, 2019, from https://slurm.schedmd.com/overview.html

[37] M. Mao, H. Humphrey, A performance study on the VM startup time in the cloud, IEEE 5th International Conference on Cloud Computing, June, IEEE, 2012, pp. 423-430 (2012)

[38] Razavi K., Razorea L.M., Kielmann T. (2014) Reducing VM Startup Time and Storage Costs by VM Image Content Consolidation. In: an Mey D. et al. (eds) Euro-Par 2013: Parallel Processing Workshops. Euro-Par 2013. Lecture Notes in Computer Science, vol 8374. Springer, Berlin, Heidelberg

[39] Marathe, Aniruddha & Harris, Rachel & K. Lowenthal, David & R. de Supinski, Bronis & Rountree, Barry & Schulz, Martin & Yuan, Xin. (2013). A comparative study of high-performance computing on the cloud. HPDC 2013 - Proceedings of the 22nd ACM International Symposium on High-Performance Parallel and Distributed Computing.