An SLA-Based Advisor for Placement of HPC Jobs on Hybrid Clouds

Kiran Mantripragada, Leonardo P. Tizzei, Alecio P.D. Binotto, and Marco A.S. Netto(✉)

IBM Research, Sao paulo, Brazil
{kiran,ltizzei,abinotto,mstelmar}@br.ibm.com

Abstract. Several scientific and industry applications require High Performance Computing (HPC) resources to process and/or simulate complex models. Not long ago, companies, research institutes, and universities used to acquire and maintain on-premise computer clusters; but, recently, cloud computing has emerged as an alternative for a subset of HPC applications. This poses a challenge to end-users, who have to decide where to run their jobs: on local clusters or burst to a remote cloud service provider. While current research on HPC cloud has focused on comparing performance of on-premise clusters against cloud resources, we build on top of existing efforts and introduce an advisory service to help users make this decision considering the trade-offs of resource costs, performance, and availability on hybrid clouds. We evaluated our service using a real test-bed with a seismic processing application based on Full Waveform Inversion; a technique used by geophysicists in the oil & gas industry and earthquake prediction. We also discuss how the advisor can be used for other applications and highlight the main lessons learned constructing this service to reduce costs and turnaround times.

1 Introduction

Current work on HPC cloud has focused on understanding the cost-benefits of cloud over on-premise clusters [1,6–9,13,15–17,20,23]. However, there is still a gap between this understanding and helping users make decisions on bursting their jobs to the cloud. While applications may suffer network overhead, cloud is more effective when we consider resource availability—users do not have to wait long time periods in job queues of cluster management systems.

This paper introduces an advisory service to support users in deciding how to distribute computing jobs between on-premise and cloud resources. Our main contributions are: (i) an advisory service for bursting jobs to the cloud, considering performance and cost difference between cloud and on-premise resources, as well as deadline, local job queue, and application characteristics (Sect. 2); (ii) a case study that shows the advisory service being used by a seismic processing application from the oil & gas industry. We also measured the impact of unreliable execution time predictions on cloud bursting decisions (Sects. 3 and 4).

Extended version of this paper is available at arxiv.org.

A. Barros et al. (Eds.): ICSOC 2015, LNCS 9435, pp. 324–332, 2015.
DOI: 10.1007/978-3-662-48616-0_21

2 Advisory Service and Policies

The advisory service considers a user deadline, incurred costs, the on-premise job queue length (local), the provisioning time (cloud), the price ratio between local and cloud for the resource allocation, the type of available hardware, and the estimated execution time for both environments with different configurations.

The main input parameters to the advisor are the *application profiles* and the *cost models*. The *application profiler* generates profiles that describe the behavior of a given application considering infrastructure, financial costs, performance, and number of required processors. Several approaches exist to produce application profiles [2,3,10,18,22,24]. The *cost model* for a cloud infrastructure comes from price values offered by cloud providers, whereas the model for the on-premise cluster is a ratio based on the cloud costs [14].

The advisory service currently supports two policies: (i) the maximum *budget* for running jobs, when users are more concerned about costs; (ii) the maximum *execution time*, when users must meet a deadline to deliver results, and budget is a secondary concern. Both policies readjust the number of cores for the cloud environment due to restrictions in the number of cores per machine.

3 Application Case Study in Oil and Gas Industry

The case study of our advisory service relies on the application profile for the Full Waveform Inversion (FWI) [21] and cost models for the SoftLayer cloud provider and an on-premise cluster. FWI is a CPU-intensive application in the area of seismic analysis that has components present in other HPC applications, such as communication among multiple processes, solvers for linear systems, matrix operations, among others. We assume that the profile of such applications can be represented by a power-law function due to its inherent scale-invariance characteristics. Hence, similar applications can be scaled to a finer or coarser grid resolution and will behave similarly, by simple tuning the coefficients of the power-law function [11,12]:

$$t = a\,P^b \tag{1}$$

where t is the execution time, P is the number of processors, and the coefficients a and b are empirically determined. We can also invert Eq. 1 to solve the number of processors for a given time restriction t as the input parameter.

In order to develop a *cost model*, we collected prices charged by cloud providers; in our case, SoftLayer[1] cloud infrastructure. Similar findings from our experiments could be obtained using other cloud providers as they rely on similar prices and charging models (hourly-based). Heterogeneity [5] will be explored as future work. We observed that the hourly-rate for provisioning nodes is a linear relationship to the number of processors P, which can be described by $(C_h = \alpha P + \beta)$. For simplification purposes, we assume the offset coefficient (β) can be neglected and the "price per hour":

[1] SoftLayer website: http://www.softlayer.com/.

$$\frac{\Delta C}{\Delta t} = \alpha P \tag{2}$$

where $\frac{\Delta C}{\Delta t}$ is the hourly-rate for nodes provisioning and α is a linear coefficient determined empirically.

We can integrate Eq. 2 over time to quantify the total cost for a given turnaround time (the number of processors P does not change with time), while we simplified the costs of on-premise HPC clusters by assuming it is proportional to cloud costs: [8,14]: $C_{cloud} = T(\alpha P)$ and $C_{local} = T(K\,\alpha\,P)$, where T is the turnaround time and C is the total cost for a given number of processors P and turnaround time T. By coupling the application and costs models (Eqs. 1 and 2), we can have one equation that provides C (total cost) for a given time T, a cost model coefficient (α), and the application profile (from a and b):

$$C = a\,\alpha \left[\frac{\left(\frac{T}{a}\right)^{(1+\frac{1}{b})}}{1+\frac{1}{b}} \right]. \tag{3}$$

4 Evaluation

The goals of the evaluation are to understand: (i) the financial and time savings of the advisor and (ii) how the advisor is dependent from the application profile accuracy. We compared the advisor against four policies:

- **Always-Local:** submits jobs to the on-premise environment—it represents users who do not want or cannot move their jobs to the cloud. We used this policy as baseline for comparison because it still represents the most conservative and traditional behavior of HPC users;
- **Always-Cloud:** submits jobs to the cloud—it represents users who do not have access to an on-premise cluster or are willing to test the cloud to avoid acquiring a new cluster in the future;
- **Random:** randomly decides between cloud and local environments—it is an attempt to represent users who do not have any supporting mechanism or intuition to know where to run their jobs;
- **Worst-Case:** chooses the opposite environment provided by the advisor—it represents hypothetical users who make extremely wrong decisions. This helps us understand how much a user can loose with such decisions.

Other policies [4,19] could be studied, however finding the optimal resource allocation policy is out of the scope of this paper.

The input data were: deadline ranges from 1 to 100 h; budget ranges from 10 to 100 USD; queue size time ranges from 1 % to 50 % of the deadline; setup time ranges from 1 % to 50 % of the deadline; price ratio between cloud and local environments, with the price of cloud environment ranging from 70 % to 340 % the price of the local environment; total of 28,000 executions per policy. The ranges for the budget, deadline, and setup time are based on our experience

with the FWI application. The price ratio is based on HPC cloud literature [8]. The FWI profile was generated using a single input data set, which described the size of the domain, the precision of the output image, and the varying number of processors from 10 to 40.

The advisor computes the costs and turnaround time for both environments for each set of input variables. For each result, the advisor calculates the relative difference between the costs of both environments in the following way:

$$\frac{min(Cost_{cloud}, Cost_{local}) - Cost_{local}}{Cost_{local}} \tag{4}$$

where $Cost_{local} > 0$. When the budget-aware policy is executed, the advisor calculates the relative difference of the turnaround time in a similar manner. The results of the other decision policies used for comparison are also relative to the always-local decision policy.

We selected two environments to compare the target application. Cloud: processor frequency = 2.60 GHz, cores per processor 4, memory per machine 64 GB, Ethernet network, CentOS operating system. Cluster: processor frequency = 2.80 GHz, cores per processor 10, memory per machine 132 GB, Ethernet/Infiniband network, RHEL operating system.

Application profiles describe resource consumption of the application. We derived the power-law scaling function (Eq. 1), in which the coefficients a and b were computed through non-linear least squares curve fitting:

$$t_{local} = 1013.50\, P_{local}^{-1.58} \text{ and } t_{cloud} = 7004.86\, P_{cloud}^{-2.06}.$$

4.1 Results: Costs and Time Savings

Figure 1 shows the results for the deadline-aware policy. When local environment is cheaper ($K < 1$) and even 80 % ($K = 1.8$) more expensive than cloud, it delivers the cheapest cost most of the time. When the price ratio is 1.8, local environment provides the cheapest cost around 71 % of the instances. The reason for this is that local environment showed the best computing performance for executing the target application. Thus, even when the local environment is around 80 % more expensive than cloud, its performance counterbalances its cost. When the price ratio reaches 2.2, the local environment is surpassed by cloud in terms of cost, *i.e.*, cloud is the cheapest environment around 56 % of the time and this percentage becomes greater as the price ratio grows, as expected.

Figure 2 shows the results for the budget-aware policy. Similarly to deadline-aware, always-cloud is comparable to worst-case and advisor is comparable to always-local when the local price is equal to or below the cloud price. However, for higher price ratios, always-cloud surpasses always-local faster than deadline-aware. The turning point occurs when price ratio is around 1.0: always-local is on average 30 % faster than always-cloud, but the median is –0.12; that is, always-cloud is half the time at least 12 % faster than always-local. Although the local environment has better computing performance results over cloud, when they have a similar price (*i.e.*, price ratio around 1.0), the decision on where to run for

a given budget is not obvious due to the other input parameters (queue length and setup time), affecting the turnaround time.

When the advisor calculates an execution time to meet the budget, the coupled model yields a solution that is near-optimal for execution time. Searching for the optimal execution time is not worthwhile since the adjustments over the infrastructure to meet the available configurations overpass intermediate values found in the optimal solution. For instance, an estimated number of processors $NProcs = 9.5$ must be adjusted to 10 or 9, according to the policies.

Results show the advisor selects the environment that best suits users' needs. The lack of such supporting tool might cause unnecessary costs or waste of time. Besides, some input variables (*e.g.*, queue size time) can change frequently, making impossible to manually calculate the best environment for execution.

4.2 Results: Accuracy of the Application Profile

We defined the range of inaccuracies from $-90\,\%$ (*i.e.*, -0.9) error to $100\,\%$ (*i.e.*, 1.0) error, aiming to cover a wide spectrum of such profiles. For each set of input data, the application profile specifies the infrastructure necessary to meet deadline or budget constraints depending on the policy. Let us say this infrastructure has 100 cores disregarding the policy; after injecting the error within the aforementioned range, it will have from 10 (i.e., $100 * (1.0 - 0.9)$) to 200 cores (i.e., $100 * (1.0 + 1.0)$).

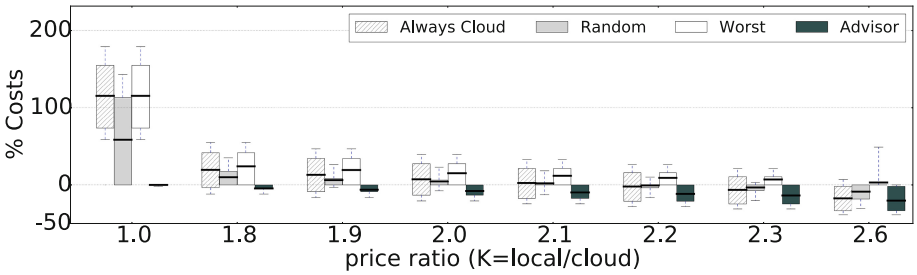

Fig. 1. Boxplots of the Deadline-aware policy: the lower the costs the better the policy

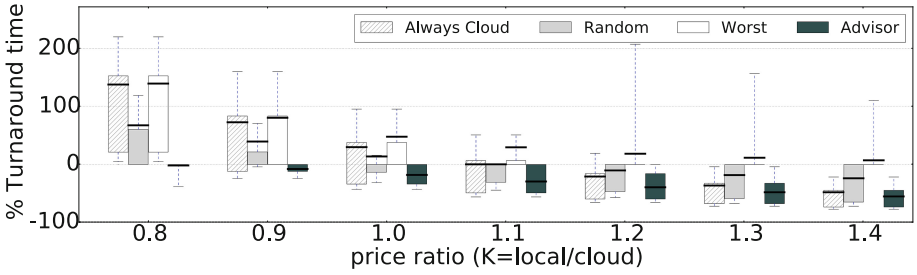

Fig. 2. Boxplots of the Budget-aware policy: the lower the time, the better the policy

For each estimation of the advisor, the input data also varied in the same way that was described in previous evaluation (Sect. 4.1), which means that each policy has been executed 28,000 times for each inaccurate profile. After the execution, the advisor provided either the same decision that was computed using the accurate profile or a different one. Even if the decision was the same, it is usually based on slightly different results. Thus, we measured whether the decision is the same or not, and the relative difference between results using an inaccurate and the accurate profile. These relative differences, when executing the deadline-aware policy, were calculated as follows:

$$\frac{Cost_{Inaccurate} - Cost_{Accurate}}{Cost_{Accurate}} \tag{5}$$

where $Cost_{Inaccurate}$ and $Cost_{Accurate}$ are the costs measured using inaccurate and accurate profiles, respectively. The relative differences were calculated similarly when executing the budget-aware policy. Table 1 shows a comparison of the results provided by the advisor using inaccurate and accurate profiles. The first column compares the decision of the advisor using both profiles: = means same decision and ≠ otherwise. For each policy, this table shows the average of the relative differences (avg), their standard deviation (std) and total number of decisions (size) for each inaccurate profile. So, for each inaccurate profile, the sum of equal and different decisions is 28,000.

As the error gets close to zero, the percentage of same decisions increases. Even when the error is 0.9, the advisor computed the same decision for deadline-aware policy around 93 % of the times and for budget-aware policy around 92 % of times. The number of different decisions for the budget-aware is greater than the number of same decisions (53 % against 47 %, respectively) only when the

Table 1. Results from the advisor using inaccurate and accurate profiles

Decision	Error	Deadline-aware			Budget-aware		
		Avg	Std	Size	Avg	Std	Size
=	–0.9	–0.8	0.3	17293 (62 %)	–0.2	0.7	13068 (47 %)
≠		–1.0	0.0	10707 (38 %)	–0.4	0.5	14932 (53 %)
=	–0.5	–0.4	0.3	25356 (91 %)	0.3	0.6	23236 (83 %)
≠		–0.6	0.0	2644 (9 %)	0.0	0.4	4764 (17 %)
=	–0.1	–0.1	0.1	26004 (93 %)	0.1	0.4	27272 (96 %)
≠		0.1	0.3	1996 (7 %)	0.2	0.3	728 (4 %)
=	0.1	0.1	0.1	27115 (97 %)	0.1	0.1	27829 (99 %)
≠		0.1	0.1	885 (3 %)	0.1	0.0	171 (1 %)
=	0.5	0.4	0.4	26597 (95 %)	0.1	0.3	26685 (95 %)
≠		0.8	0.2	1403 (5 %)	0.2	0.2	1315 (5 %)
=	0.9	0.9	0.8	25982 (93 %)	0.0	0.3	25807 (92 %)
≠		1.5	0.4	2018 (7 %)	0.3	0.3	2193 (8 %)

error is –0.9. For this error, the advisor proposed the same decision for the deadline-aware policy around 62 % of the time.

For most of the inaccuracy ranges, the number of equal decisions is far greater than the number of different decisions. That is, even if the application profile is inaccurate, it has a minor decision impact. Therefore, the advisor provides evidence that it is resilient to inaccuracies in the application profile. One reason for this is the wide spectrum of data that has been exercised. In some situations, the difference between choosing cloud or local is so great that the inaccuracy has little to no impact on job(s) placement decision. These results also show that as inaccuracy gets close to zero, the number of correct decisions increases, as expected. When the inaccuracy is close to zero, the infrastructure calculated by the application profile has low chance of being relevant to the final result.

5 Conclusions

The advisory service is composed of modules that can be extended/plugged-in to have more refined *Application Profiles* and to suit other applications. Further investigations will be required to collect data from a wide range of applications. The main lessons from our study are: (i) in HPC cloud, apart from resource performance, it is important to consider the time a user has to wait in a job queue of the on-premise environment compared to the overhead of cloud resources—more relevant is the total turnaround time, as also pointed by Marathe *et al.* [14]; (ii) it is possible to consider an advisory service for HPC hybrid clouds even without having highly precise application/job profiles—however, very inaccurate profiles may generate negative impact on costs and turnaround delays; (iii) the higher the cost differences between cloud and on-premise resources the higher the savings brought by an advisory service for resource selection on hybrid clouds.

Acknowledgment. We thank Eduardo Rodrigues and Nicole Sultanum for their comments on this paper. This work has been partially supported by FINEP/MCTI under grant no. 03.14.0062.00.

References

1. Belgacem, M.B., Chopard, B.: A hybrid HPC/cloud distributed infrastructure: coupling EC2 cloud resources with HPC clusters to run large tightly coupled multiscale applications. Future Gener. Comput. Syst. **42**, 11–21 (2015)
2. Binotto, A.P.D., Wehrmeister, M.A., Kuijper, A., Pereira, C.E.: Sm@rtConfig: a context-aware runtime and tuning system using an aspect-oriented approach for data intensive engineering applications. Control Eng. Prac. **21**(2), 204–217 (2013)
3. Calheiros, R.N., Netto, M.A.S., Rose, C.A.F.D., Buyya, R.: EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications. Prac. Experience, Softw. **43**(5), 595–612 (2013)

4. De Assunção, M.D., Di Costanzo, A., Buyya, R.: Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In: Proceedings of the ACM International Symposium on High Performance Distributed Computing (2009)
5. Delimitrou, C., Kozyrakis, C.: QoS-aware scheduling in heterogeneous datacenters with paragon. ACM Trans. Comput. Syst. **31**(4), 12 (2013)
6. Gentzsch, W., Yenier, B.: The UberCloud HPC experiment: compendium of case studies. Technical report, Tabor Communications, Inc. (2013)
7. Gentzsch, W., Yenier, B.: The UberCloud experiment: technical computing in the cloud - 2nd compendium of case studies. Technical report, Tabor Communications, Inc. (2014)
8. Gupta, A., Kale, L.V., Gioachin, F., March, V., Suen, C.H., Lee, B.S., Faraboschi, P., Kaufmann, R., Milojicic, D.: The who, what, why and how of high performance computing applications in the cloud. In: Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (2013)
9. Gupta, A., Milojicic, D.: Evaluation of HPC applications on cloud. In: Open Cirrus Summit (2011)
10. Jarvis, S.A., Spooner, D.P., Keung, H.N.L.C., Cao, J., Saini, S., Nudd, G.R.: Performance prediction and its use in parallel and distributed computing systems. Future Gener. Comput. Syst. **22**(7), 745–754 (2006)
11. Li, H.: Workload dynamics on clusters and grids. J. Supercomput. **47**(1), 1–20 (2009)
12. Lowen, S.B., Teich, M.C.: Fractal-Based Point Processes. Wiley, New York (2005)
13. Mantripragada, K., Binotto, A., Tizzei, L.P.: A self-adaptive auto-scaling method for scientific applications on HPC environments and clouds. In: Proceedings of the International Workshop on Adaptive Self-tuning Computing Systems (2015)
14. Marathe, A., Harris, R., Lowenthal, D.K., de Supinski, B.R., Rountree, B., Schulz, M., Yuan, X.: A comparative study of high-performance computing on the cloud. In: Proceedings of the International Symposium on High-performance Parallel and Distributed Computing (2013)
15. Mateescu, G., Gentzsch, W., Ribbens, C.J.: Hybrid computing-where HPC meets grid and cloud computing. Future Gener. Comput. Syst. **27**(5), 440–453 (2011)
16. Napper, J., Bientinesi, P.: Can cloud computing reach the top500? In: Proceedings of the Combined Workshops on UnConventional High Performance Computing Workshop Plus Memory Access Workshop (2009)
17. Ostermann, S., Iosup, A., Yigitbasi, N., Prodan, R., Fahringer, T., Epema, D.: A performance analysis of EC2 cloud computing services for scientific computing. In: Avresky, D.R., Diaz, M., Bode, A., Ciciani, B., Dekel, E. (eds.) Proceedings of Cloud Computing. LNICST, vol. 34, pp. 115–131. Springer, Heidelberg (2010)
18. Sadjadi, S.M., Shimizu, S., Figueroa, J., Rangaswami, R., Delgado, J., Duran, H., Collazo-Mojica, X.J.: A modeling approach for estimating execution time of long-running scientific applications. In: Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (2008)
19. Unuvar, M., Steinder, M., Tantawi, A.N.: Hybrid cloud placement algorithm. In: Proceedings of the IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (2014)
20. Vecchiola, C., Pandey, S., Buyya, R.: High-performance cloud computing: a view of scientific applications. In: Proceedings of the International Symposium on Pervasive Systems, Algorithms, and Networks (2009)
21. Virieux, J., Operto, S.: An overview of full-waveform inversion in exploration geophysics. Geophysics **74**(6), WCC1–WCC26 (2009)

22. Yang, L.T., Ma, X., Mueller, F.: Cross-platform performance prediction of parallel applications using partial execution. In: ACM/IEEE Supercomputing (2005)
23. Zaspel, P., Griebel, M.: Massively parallel fluid simulations on amazon's HPC cloud. In: Proceedings of the International Symposium on Network Cloud Computing and Applications (2011)
24. Zheng, G., Wilmarth, T., Jagadishprasad, P., Kalé, L.V.: Simulation-based performance prediction for large parallel machines. Int. J. Parallel Program. **33**(2), 183–207 (2005)