

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

Kiran Mantripragada, Leonardo P. Tizzei,
Alecio P. D. Binotto, Marco A. S. Netto

IBM Research

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

Cloud computing was initially created to host web applications, but has since become an alternative platform for several complex applications, like in Big Data and High Performance Computing (HPC) areas. Typically, these applications execute on on-premise clusters due to their heavy infrastructure requirements; however, as virtualization and network technologies evolve, users can burst to the cloud part or all their workloads to reduce costs and response time.

For cloud environments, one of the key issues is deciding which jobs should be submitted to on-premise resources and which should be burst to the cloud. This decision depends on several variables, such as job execution time on both platforms (local and remote), financial costs, job queuing waiting time, provisioning time in the cloud, among others. End-users need to consider all these parameters to make proper decisions, which increase the complexity for (hybrid) cloud deployments and limit cloud applicability for HPC-based applications, especially when Service Level Agreements (SLAs) are in place. Apart from the

The final publication is available at link.springer.com - ICSOC'15

number of variables, end-users have to handle their volatility, which increases the chance of wrong decisions, wasting money, or missing important deadlines.

Current work on HPC cloud has mainly focused on understanding the cost-benefits of using cloud over on-premise clusters [8, 16, 17, 20]; and there is still a gap between understanding the cloud performance and costs, and helping users make decisions on bursting their applications to the cloud. While applications may have overhead in the cloud, mainly due to network performance, 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 computational jobs between on-premise and cloud resources. In summary, our main contributions are:

- An advisory service for bursting applications to the cloud, considering performance and cost difference between cloud and on-premise resources, as well as deadline, local job queue, and application characteristics (§ 2);
- A case study that shows the advisory service being used by a real world application—a seismic processing technique from the oil & gas industry. We employed a real test-bed with cloud and on-premise cluster resources, and measured the impact of unreliable execution time predictions on cloud bursting decisions (§ 3, §4).

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. A few of these variables, mainly related to performance and cloud resource setup times, can be specified in SLAs placed between the parties, including users, advisory service, and cloud provider.

The main input parameters to this advisor are the *application profiles* and the *cost models*. The *application profiler* generates a comprehensive application behavior considering infrastructure, financial costs, performance, and number of required processors. Several approaches exist to produce application profiles [3, 10, 18, 22, 24]—depending on the application and technique, different accuracy levels can be reached.

The estimation of a *cost model* for a cloud infrastructure is not a difficult task as it can be defined through simple simulations in the cloud provider. On the other hand, the *cost model* of on-premise resources is challenging since each owner operates financially different, and so we tackle this problem by assuming that *cost model* of on-premise infrastructure is proportional to the *cost model* of cloud infrastructure [14].

This section presents the architecture of the advisory service and two policies for HPC job mapping in hybrid clouds. In Section 3, we instantiate the advisor for a specific use case to show the practical applicability of the proposed service.

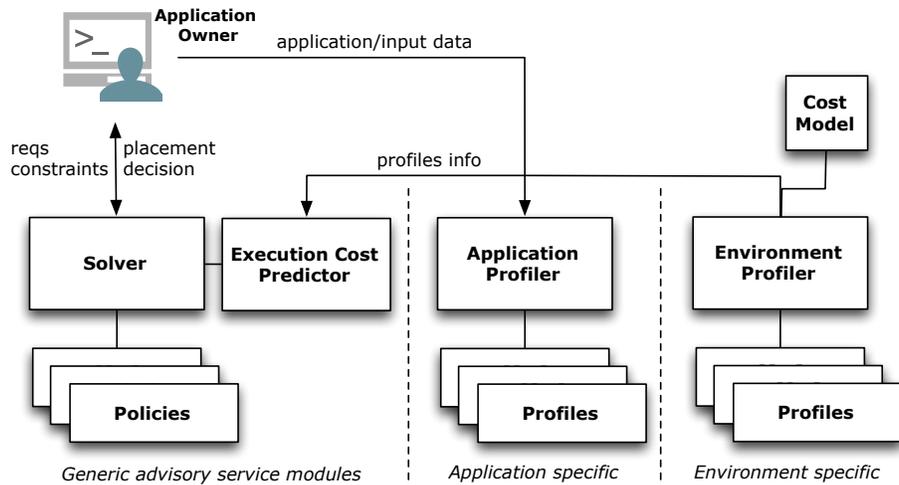


Fig. 1. Advisory service architecture

2.1 Architecture

The design of the advisory service is intended to be as generic as possible, relying on domain and application specific modules discussed in Section 3. Figure 1 presents the advisory service architecture containing three groups of modules: generic advisory service modules, application, and environment specific profilers.

Solver receives requirements and constraints to be fulfilled (*e.g.*, cost, execution time, budget, deadline). It calls the **Execution Cost Predictor**, which estimates the cost of the executions based on their application profile, on the environment cost model, and optionally on execution logs. The application profile describes the relation between computational resources and execution time, which is important to estimate the total execution cost based on a particular model. Whenever users run applications, details of their executions are stored in a database, similar to the approach described by Binotto *et al.* [2]. These logs refine future comparisons as they provide historical data that can be analyzed to improve accuracy of the estimations.

Particularly, the **Solver** module can use the results of the **Execution Cost Predictor** module to find the best configuration given user’s variables and constraints. It first classifies all configurations in two categories: those that fit and those that do not fit the constraints. The earlier are sorted based on the optimization criterion, *i.e.*, time or budget. The solver relies on placement policies—two examples of policies are described in the following section as they are generic enough to be used by several applications and environments.

Application Profiler characterizes an application considering computational resources (*e.g.*, processors, memory, network, I/O) and execution time. In order to compare cloud and on-premise cluster, users should generate application profiles for both environments, which are captured by the **Environment Profiler** containing the environment **Cost Model** (Section 3.2).

The proposed 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.

2.2 Budget-aware policy

For the Budget-aware policy, the user provides a budget restriction and the policy finds the best placement for the job, considering queue length for local cluster, a time for setup and provisioning in the cloud environment, and the price ratio between local and cluster. Here are the steps of this policy:

- (I) Get the **execution time** for given budget (inverse of Eq. 6, Section 3.3);
- (II) Find the **number of processors** to be allocated in a **cloud infrastructure** to meet the execution time requirement (Eq. 1, Section 3.1);
- (III) Define a vector with the **processors per node** distribution based on the available on-premise hardware and the cloud provider. In our experiments, we used the following configuration:
 $\mathbb{P}_{cloud} = \{1, 2, 4, 8, 12, 16\}$, $\mathbb{P}_{local} = \{1, 2, 3, \dots, 200\}$.
 Thus, if the advisor computes a number of processors $NProc_{cloud} = 45$, the distribution will be: $ProcPerNode_{cloud} = [16, 16, 13]$. Note that the last node would have 13 processors, but there may be no such configuration available in the cloud provider. It would be possible to define a distribution of 45 processors in the following way: $ProcPerNode_{cloud} = [16, 16, 12, 1]$, but it would require an additional processor, which would increase the costs. Instead, the distribution is defined as described in the next step.
- (IV) Adjust the number of processors based on the available configuration:
 New $ProcPerNode_{cloud} = [16, 16, 12]$.
 The last node was downsized to 12 because this is the next available value below 16 in \mathbb{P}_{cloud} . Since the user is looking for a turnaround time given the budget restrictions (budget-aware policy), the advisor must reduce the number processors trying to meet that budget baseline.
- (V) Compute the **new execution time** and **turnaround time** from number of processors, queue length, and setup time. The execution time is estimated from equations that define the *Application Profile* (Section 3.1).
- (VI) Compute the costs for the estimated infrastructure using the Cost Model (Section 3.2).

At the end, the advisor provides an achievable deadline based on budget restrictions. The decision would be to place the job in an infrastructure that delivers shorter turnaround time if they are still within the restricted budget.

2.3 Deadline-aware policy

This policy determines the required number of processors that meets the deadline in cloud and local environments. The policy steps are:

- (I) Get **execution time** for a deadline, setup time and queue length:
 $t = T - Q$, where t is the execution time, T is the total time (deadline), and Q is the queue length (local) or the setup time (cloud).
- (II) Find the **number of processors** that can fulfill the execution time requirement (Eq. 1 from the *Application profile*);
- (III) Define a vector with the **processors per node** distribution based on the available hardware (local) and the offers from the cloud provider. In our experiments, we used the following configuration:
 $procs_cloud = \{1, 2, 4, 8, 12, 16\}$, $procs_local = \{1, 2, 3, \dots, 200\}$
 Thus, if the advisor computes a number of processors $NProcs_{cloud} = 41$, the distribution will be:
 $ProcPerNode_{cloud} = [16, 16, 9]$. Note that, again, the last node would have 9 processors, but there is no such configuration in the cloud provider. It would be possible to use the following distribution: $ProcPerNode_{cloud} = [16, 16, 8, 1]$, but it would require an additional processor, which could increase the network communication and possibly the execution time. Instead, we define the distribution of processors as described in the next step.
- (IV) Adjust the number of processors based on the available configuration:
 New $ProcPerNode_{cloud} = [16, 16, 12]$, where the last node was upscaled to 12 since this is the next available value greater than 9 in \mathbb{P}_{cloud} . The same adjustment is done for the *procs per node* in the local HPC. Since users are looking for resources that meet their time constraints, we increase the number of processors in order to satisfy the restricted execution time.
- (V) Compute the new number of processors according the previous adjustments (new $NProcs = 16 + 16 + 12 = 44$);
- (VI) Compute the **new Execution Time** and **Turnaround Time** from the adjusted number of processors (Eq. 1), the queue length, and setup time;
- (VII) Compute the costs associated with the defined infrastructure (Eq. 3).

At the end of this policy, the advisor provides cost values for local and cloud infrastructures, trying to meet the deadline. If both environments can deliver the restricted deadline, the decision would be based on the Total Costs.

3 Application Case Study in Oil & Gas Industry

In this section, we present a case study based on Full Waveform Inversion (FWI) application [21]. To instantiate the advisory service for this application, we created the application profile and cost model components for SoftLayer cloud provider and an on-premise cluster. Rather than showing an overview of the advisor for several applications, we focused on a detailed explanation of a use case placed into practice. The FWI application contains components of several HPC applications such as communication among multiple processes, solvers for linear systems, matrix operations, among others. There are several ways to create the application profile, which can be implemented and plugged-in into the

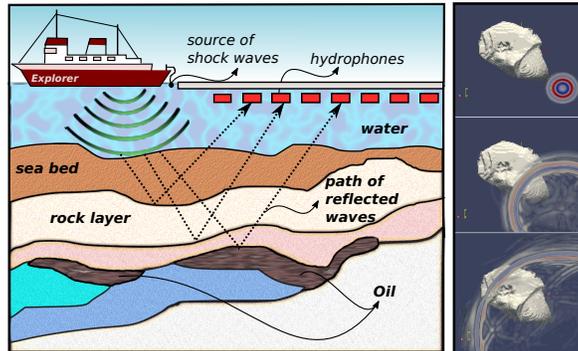


Fig. 2. FWI data collection and remote visualization

advisory service [3,10,18,22,24]. The more complex the application behavior, the less accuracy and more time will be required to create the profile. We evaluate the impact of accuracy in Section 4.3.

FWI is a numerically challenging technique based on full wavefield modeling of a geological domain that extracts relevant quantitative information from seismograms. When dealing with realistic physics-based elastic partial differential equations formulation and accurate discretization techniques, such as high order Spectral Element Method, the forward modeling becomes particularly challenging from the computational point of view. Millions of shots in different positions are calculated over the same data domain and each shot solution evolves along time steps. At the end, the generated images are composed to produce the final outcome (Figure 2).

3.1 Application Profile

We considered the FWI application, which is based on a multi-resolution, Partial Differential Equation solver to run our experiments. We assume that the profile of such applications can be represented by a power-law function due to its inherent scale-invariance characteristics. For instance, 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], for arbitrary constants a and b . From experiments, we observed that the *Application Profile* function can be modeled as:

$$t = a P^b \quad (1)$$

where t is the execution time, P is the number of processors, and the coefficients a and b are empirically determined. More details on these coefficients can be found in Section 4.1, Figure 3(b), and Equation 8. In addition to Equation 1, we need to determine the number of processors for a time restriction t as the input parameter. This is solved by inverting Equation 1.

3.2 Environment Profile: Cost Model

In order to develop a *cost model*, we collected prices charged by cloud providers. We used the SoftLayer¹ cloud infrastructure—similar findings from our experiments could be obtained using other cloud providers as they mostly rely on similar prices and charging models (hourly-based). We consider here homogeneous resources and leave heterogeneity [5] as future work. Table 1 shows SoftLayer fees in US Dollars for different memory size configurations.

Table 1. Costs/hour over number of cores and three memory configurations

N Cores	1GB/proc	2GB/proc	4GB/proc
	Cost(\$/hour)	Cost(\$/hour)	Cost(\$/hour)
1	0.040	0.059	0.098
2	0.079	0.118	0.183
4	0.159	0.224	0.334
8	0.306	0.416	0.591
12	0.444	0.674	0.806
16	0.581	0.756	1.019

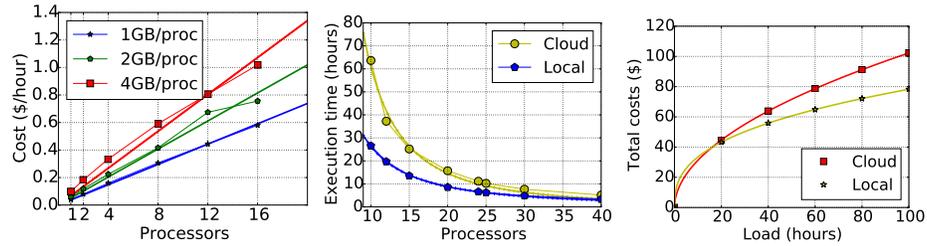


Fig. 3. Cost Model: (a) Listed price per node (cloud only); (b) Workload profile; (c) prices for 4GB RAM/proc and price ratio $k=1$

We observed that the hourly-rate for provisioning nodes is a linear relationship to the number of processors P . These functions are shown in Figure 3(a) and were fitted from real prices offered in SoftLayer (Table 1). The ratio “price per hour” can be described by a general linear equation ($C_h = \alpha P + \beta$). For simplification purposes, we assume the offset coefficient (β) can be neglected and the “price per hour” is directly proportional to the number of processors. We also claim that C_h can be written as dC/dT , by considering a continuous execution time:

$$\frac{dC}{dt} = \alpha P \quad \Rightarrow \quad dC = (\alpha P)dt \quad (2)$$

where dC/dt is the hourly-rate for nodes and α is a linear coefficient determined empirically.

¹ SoftLayer website: <http://www.softlayer.com/>

In order to quantify the total cost for a given turnaround time, we can integrate Equation 2 over time (remembering that the number of processors P does not change with time):

$$C = \int_0^T (\alpha P) dt \quad \Rightarrow \quad C = T (\alpha P) \quad (3)$$

where T is the turnaround time and C is the total cost for a given number of processors P and turnaround time T .

The integration constant is *zero*, since $C(0) = 0$, meaning there is no costs when execution time is also *zero*.

It is not a trivial task to model costs of on-premise HPC clusters [14], as it is dependent on acquisition time, depreciation, occupancy, and maintenance costs. Therefore, in order to simplify the on-premise cost model, we assume it is proportional to cloud costs [8, 14]:

$$\left(\frac{dC}{dt}\right)_{local} = K * \left(\frac{dC}{dt}\right)_{cloud} \quad (4)$$

where $(dC/dt)_{local}$ and $(dC/dt)_{cloud}$ are hourly prices for local and cloud infrastructures respectively, and K is the local-cloud price ratio. All remaining equations can be derived from Equation 4 in a similar way done for cloud costs (Equations 2 to 3)

$$C = T (K \alpha P) \quad (5)$$

3.3 Integrating application profile and costs model

In order to find the total costs for a given turnaround time, we integrated this function over time. From Equations 2 and 1, the following can be written:

$$dC = \left[\alpha \left(\frac{t}{a}\right)^{\left(\frac{1}{b}\right)} \right] dt \quad \Rightarrow \quad \int dC = \int_0^T \left[\alpha \left(\frac{t}{a}\right)^{\left(\frac{1}{b}\right)} \right] dt$$

We have C , which is the total cost for a given time T , a cost model (from α), and the application profile (from a and b):

$$C = a \alpha \left[\frac{\left(\frac{T}{a}\right)^{\left(1+\frac{1}{b}\right)}}{1 + \frac{1}{b}} \right] \quad (6)$$

Equation 6 defines the coupled “*Profile-Cost model*”, and are used in the “Budget-aware policy”. Figure 3(c) shows the costs function $C(T)$ for a load (execution time X number of processors).

The inverse function $C^{-1} = T(C)$ is also useful when we have a budget restriction and want to look for a total execution time.

4 Evaluation

The goals of the evaluation are to understand: (i) the financial and time saving benefits of the advisor (Section 4.2); and (ii) how the advisor is dependent from the accuracy of the application profile (Section 4.3). We compared the advisor against four policies:

- **Always-local:** chooses to always submit 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:** chooses to always submit 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 lose with such decisions.

This is not an exhaustive list of policies that we could use for comparison. It represents a few extreme cases to understand how good or bad resource allocation decisions can impact costs and turnaround times if no advising mechanisms are provided to the user. Several other policies [4, 19] could be leveraged or further developed in comparison to those used by the advisor. Finding the optimal resource allocation is out of the scope of this paper.

To compare these decision policies with the advisor, we executed both budget-aware and deadline-aware policies varying the following input data, which can be part of SLAs: deadline ranges from 1 to 100 hours; 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.

All ranges have been defined to cover a wide spectrum of possible inputs. 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 case study 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, thus executing both deadline-aware and budget-aware policies. For instance, if the deadline-aware policy is executed, it computes the costs for both environments, say $Cost_{cloud}$ and $Cost_{local}$, in order to choose the cheapest one. For each result, the advisor calculates the relative difference between the costs of both environments in the following way:

$$\frac{\min(Cost_{cloud}, Cost_{local}) - Cost_{local}}{Cost_{local}} \quad (7)$$

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.

4.1 Environment Setup

We selected two representative environments, described in Table 2, to compare the target application on cloud and on-premise environments.

Table 2. Cluster and SoftLayer details

	Cluster	SoftLayer
processor (GHz)	2.80	2.60
cores per processor	10	4
cache size (KB)	25600	20480
total memory (MB)	132128	65712
network	ethernet/infiniband	ethernet
operating system	RHEL	CentOS

The FWI application contains a set of parameters such as number of degrees of freedom, polynomial order, data domain, among others, and it is a highly CPU and memory intensive. Details of the application parameters are not relevant to the experiments presented here (they are addressed elsewhere [21]).

Application profiles provide information about the behavior of the application in terms of metrics, such as total elapsed time, number of cores, memory, nodes, network, and costs. The selection of these metrics depends on the constraints and variables that end-users want to optimize. For instance, Figure 3(b) shows how the total elapsed time for executing application changes as the number of processors increases in the two distinct environments.

We derived the power-law scaling function, as described in Section 3.1, from the experiments (Equation 1). The coefficients a and b were computed through non-linear least squares curve fitting. Thus, we have:

$$t_{local} = 1013.50 P_{local}^{-1.58} \quad \text{and} \quad t_{cloud} = 7004.86 P_{cloud}^{-2.06} \quad (8)$$

4.2 Results: Costs and Time Savings

The results of the comparison are presented in Figures 4 and 5. The y axis of Figure 4 represents the relative cost as calculated by Equation 7, while the y axis of Figure 5 describes the relative turnaround time. In both figures, y axes are in a symmetrical log scale and the x axes depict the price ratio. The higher the price ratio, the higher the costs for local environment compared to cloud. In

both figures, results were stabilized outside the plotted ratios, and therefore we omitted them.

Figure 4 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 environment, it delivers the cheapest cost most of the time. For instance, 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, as depicted in Figure 3. 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 the 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 5 depicts the results for the budget-aware policy. Similarly to the deadline-aware policy, the always-cloud policy 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 policy surpasses always-local faster than deadline-aware policy. 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

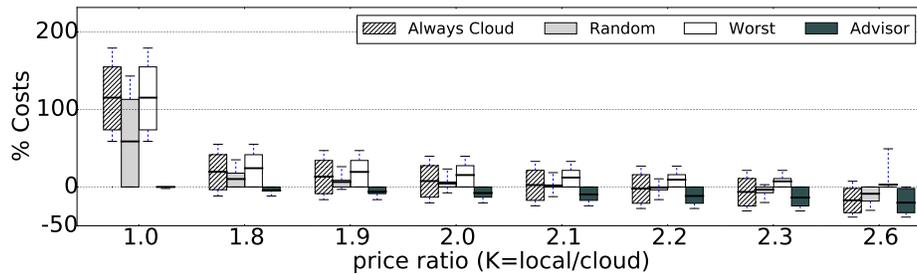


Fig. 4. Deadline-aware policy: the lower the costs the better the policy

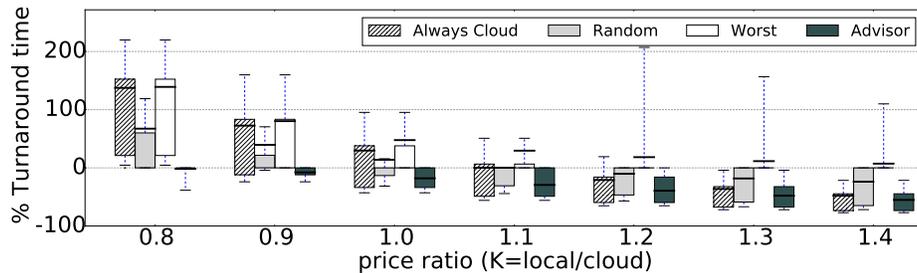


Fig. 5. Budget-aware policy: the lower the time, the better the policy

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 in terms of execution time. We observed that 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.

Both figures show that the advisor supports the selection of the environment that best suits users' needs. The lack of such supporting tool might cause unnecessary costs or waste of time as it is illustrated by the worst-case decision policy. Furthermore, some of the input variables (*e.g.*, queue size time) can change very frequently thus making impossible to manually calculate the best environment to run an application.

4.3 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 amount of 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.*, $100 * (1.0 + 1.0)$).

For each estimation of the advisor, the input data also varied in the same way that was described in previous evaluation (Section 4.2), 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:

$$\frac{Cost_{Inaccurate} - Cost_{Accurate}}{Cost_{Accurate}} \quad (9)$$

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 3 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 \neq 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.

Table 3. 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%)

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 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. Thus, 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, there is a higher chance that the infrastructure calculated by the application profile is not relevant for the final result. For example, an inaccuracy of -10% of 100 cores would calculate 90 cores; instead of running in 1 hour and 30 minutes, the job(s) would run 1 hour and 50 minutes. For a cloud environment that rents VMs per hour, the result would be 2 hours for both options.

When the error is -0.9, the number of different decisions achieved its peak for both policies. On the other hand, when the error is 0.9, the number of different decisions is small compared to the number of equal decisions. Despite the absolute values of both errors being the same, the number of different decisions provided by the advisor using these inaccurate profiles is distinct. The results provided by the advisor are based on application profile and the integrated profile-cost model, which are not linear, as depicted in Figures 3(b) and (c). This explains why these inaccurate profiles provided distinct rates.

5 Related Work

Scientific and engineering applications usually require HPC platforms and are currently being tested on modern cloud platforms that are now starting to incorporate HPC features. For instance, UberCloud experiment reports [6, 7] contain a set of use cases and challenges when moving HPC applications to the cloud. These challenges include data transfer from and to the cloud, unpredictable costs when using cloud resources which depend on workload demands and algorithm complexity, lack of easy and intuitive self-service administration, long waiting times to start-up VMs (at unacceptable level for some end-users), among others.

Ostermann *et al.* [17] evaluated whether performance of cloud environments are sufficient for scientific computing. Their results showed that cloud environments were insufficient for scientific computing at the time of their publication. At the same time, Napper and Bientinesi [16] used Linpack benchmarks to evaluate whether cloud could potentially be included in the top 500 list of supercomputing. Their results showed that the performance of single cloud nodes was as good as nodes in HPC systems, however, memory and network were not sufficient to scale the application. Vecchiola *et al.* [20] also evaluated the use of clouds for scientific applications. They concluded that clouds are effective for conducting scientific experiments, but the trade-off between costs and performance has to be evaluated case by case.

Following a new phase of HPC cloud studies, Mateescu *et al.* [15] evaluated a set of benchmarks and complex HPC applications on a range of platforms, both in-house and in the cloud. The studies showed cloud effectiveness for such applications mainly in the case of complementing supercomputers. Gupta and Milojevic [9] highlighted that cloud can be suitable for some HPC applications, but not all. The same group of authors [8] performed a set of experiments using benchmarks and complex HPC applications on various platforms, including supercomputers and clouds, to answer the question “why and who should choose cloud for HPC, for what applications, and how should cloud be used for HPC?”.

More recently, Belgacem and Chopard [1] evaluated a computational fluid dynamics application over a heterogeneous environment of a cluster and cloud. They ratified a previous work [23] that demonstrated the potential of clouds to fluid dynamics. Mantripragada *et al.* [13] proposed a method to use a hybrid approach using a local cluster plus the cloud to dynamically boost computing intensive applications dealing with data partitioning respecting performance deadlines. The results using a seismic application showed that the approach is feasible in the case of a seamless connection between both environments despite of synchronization concerns. Outside the HPC efforts, Unuvar *et al.* [19] introduced a hybrid cloud placement algorithm, which focus on application structure to allocate multiple VMs.

Although these findings do not propose advisory tools, their empirical studies open an opportunity for proposing new tools focusing on a heterogeneous approach over local and cloud, like ours.

6 Conclusions

We introduced an advisory service based on two policies to help users decide where to run their HPC jobs: on-premise *versus* cloud. It considers the application profile, job waiting queue, costs, budget, and deadline constraints. We used a real testbed and seismic application in the area of oil & gas industry to perform a comprehensive set of experiments.

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 Generation Computer Systems* (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 Engineering Practice* (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. *Software: Practice and Experience* (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 Transactions on Computer Systems* 31(4), 12 (2013)
6. Gentzsch, W., Yenier, B.: The UberCloud HPC Experiment: Compendium of Case Studies. Tech. rep., Tabor Communications, Inc. (2013)
7. Gentzsch, W., Yenier, B.: The UberCloud Experiment: Tech. Comp. in the Cloud - 2nd Compendium of Case Studies. Tech. rep., 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 Generation Computer Systems* (2006)
11. Li, H.: Workload dynamics on clusters and grids. *Journal of Supercomputing* (2009)
12. Lowen, S.B., Teich, M.C.: *Fractal-based point processes*. John Wiley & Sons (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 Generation Computer Systems* (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: Proceedings of Cloud Computing (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: Proceeding 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 & 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* (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. *International Journal of Parallel Programming* (2005)