Optimizing Monitorability of Multi-cloud Applications

Edoardo Fadda[1], Pierluigi Plebani[2], Monica Vitali[2]

[1] Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy
[2] Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy

Abstract. When adopting a multi-cloud strategy, the selection of cloud providers where to deploy VMs is a crucial task for ensuring a good behaviour for the developed application. This selection is usually focused on the general information about performances and capabilities offered by the cloud providers. Less attention has been paid to the monitoring services although, for the application developer, is fundamental to understand how the application behaves while it is running. In this paper we propose an approach based on a multi-objective mixed integer linear optimization problem for supporting the selection of the cloud providers able to satisfy constraints on monitoring dimensions associated to VMs. The balance between the quality of data monitored and the cost for obtaining these data is considered, as well as the possibility for the cloud provider to enrich the set of monitored metrics through data analysis.

Keywords: Optimized deployment, Monitoring requirements, Metric accuracy

1 Introduction

A multi-cloud application implies the availability of a set of cloud providers, not necessarily coordinated with each other, offering the capabilities to host and run resources and services that compose the application [12]. According to the Infrastructure as a Service (IaaS) provisioning model, these resources are Virtual Machines (VMs) and a multi-cloud application can rely on several VMs living on an infrastructure offered by several providers. In this context, for the application developer is important to figure out how to match VMs and cloud providers, ensuring an effective and efficient execution of the application.

In the recent years, several approaches have been proposed to find the optimal deployment of VMs among the different IaaS providers, mainly focusing on performance optimization [3] or energy consumption reduction [8]. This work integrates these important aspects with the perspective of application monitorability: the possibility to measure and assess the performances of the provided application. Instead of looking for a cloud provider able to sell VMs with some functional (e.g., size of VM) or non-functional (e.g., VM availability) characteristics, the application developer wants cloud providers able to measure those characteristics in order to know how the application using the VM behaves.

Goal of this paper is to propose a deployment optimization method based on the maximization of the quality of the monitoring system, with respect to the developer needs, and the cost of the monitoring. Here, we assume that a Cloud Broker receives the requests for deployment including the monitorability requirements. Exploiting a knowledge base, managed by the broker, the developer can easily express the monitorability requirements without entering into the technical details. The adopted multi-objective mixed integer linear optimization problem (MILP) can found the deployment solutions maximizing the quality of monitored data while minimizing the costs. To extend the possible matches, a Bayesian Network (BN) is adopted to make possible for a cloud provider to estimate the values for a dimension - that is not able to measure - based on the dependencies with other dimensions - which is actually able to measure.

The paper is structured as follows. Sect. 2 introduces the overall approach identifying the main stakeholders and the basic steps of the mechanism. Sect. 3 provides a formal definition of the optimization problem specifying the way in which the accuracy of monitored data is computed. Sect. 4 validates the approach discussing the performance and the limitations. Sect. 5 provides an overview of the current approaches related to the monitoring match-making in a cloud scenario. Finally, Sect. 6 concludes the work also outlining future extensions.

2 Overall approach

The stakeholders considered in this approach are a developer and a set of cloud providers. The developer is interested in finding out where to instantiate the VMs needed to run a cloud-based application. The cloud providers offer the facilities to host and manage VMs. The selection of the best site where to instantiate a VM is usually based on both the services offered by the cloud providers and the quality of these services. VM customization, VM migration, VM monitoring are possible services offered by cloud providers to the developers. At the same time, these services can be differentiated with respect to their quality of service (QoS): time required to instantiate a new VM, availability of the VM, availability of the entire site, and costs are example of QoS dimensions considered.

In this paper we focus on the monitoring capabilities. Cloud providers express their offerings, while the developer defines its requests according to the models discussed in the following paragraphs. The proposed match-maker is based on the implementation of a MILP model to offer to the developer a set of admissible VMs instantiation plans able to satisfy all the constraints while maximizing the quality of the monitoring data and minimizing the costs.

2.1 Cloud provider monitoring offering model

Cloud infrastructures are equipped with monitoring systems able to measure aspects like availability of VMs, CPU load, memory usage, and so on. Not all the cloud providers offer the same set of monitored properties with the same quality.

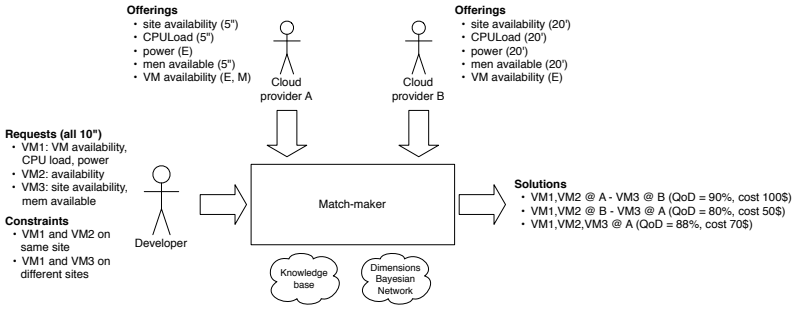


Fig. 1: Overall approach.

Moreover, even the same cloud provider offer different levels of monitoring service based on different costs and the same property can be monitored differently in terms of sampling time, precision, or adopted unit of measure. For instance, Amazon CloudWatch[3] offers a basic monitoring service where pre-selected metrics are made available at five-minute frequency with no additional cost, and a detailed monitoring where the set of metrics is the same but at one-minute frequency and with an additional cost. Yet Paraleap CloudMonix (formerly known as AzureWatch)[4] offers the possibility to monitor an unlimited set of metrics but at ten-minutes frequency, with no additional cost, or at one-minute frequency with a fee. Based on this scenario, we can say that an offering of a cloud provider can include (see Fig. 1):

- Monitored dimensions directly usable by the provider specified by their sampling time and cost for usage (not reported in the figure).
- Monitored dimensions which are not directly measured but their trends are estimated exploiting the existing dependencies among metrics [18]. A Bayesian Network is adopted to express the likelihood of a metric to increase or decrease its value when the value of another metric increases or decreases. The mark *(E)*, i.e., estimate, is used to distinguish these dimensions. Sampling time and cost are provided, where the sampling time depend on the sampling time of the dimensions used to estimate the value, while the cost can be zero as the effort required by the cloud provider to estimate this value could be negligible. With this approach, each cloud provider can extend the set of monitored dimensions to be offered to the developer, declaring the reduced quality of the monitored data.
- Monitored dimensions which are not currently measured but the cloud provider is open to install probes able to measure them. The mark (M), i.e., make, is used to distinguish these dimensions. Cost in this case could be significantly higher than the estimate, as more effort is required to the cloud provider.

[3] https://aws.amazon.com/it/cloudwatch/
[4] http://cloudmonix.com

2.2 Developer monitoring request model

For each of the VMs composing the application, the developer specifies the desired monitoring features in terms of dimensions of interest and sampling time. In the example shown in Fig. 1 we assume that all the dimensions need to be sampled every 10 seconds. It is also possible to specify a different sampling time for each dimension. The developer can define a maximum cost admissible for the solution (not reported in the figure).

The request model might also include constraints about the structure of the application. The developer can impose that the final deployment plan places groups of VMs in the same site. This could be required as the communication among those VMs is frequent and putting them on the same site can improve the performances. Other constraints can be related to the data locality or legal issues that may impose that a VM must be located (or not located) in specific countries.

As request definition can become a complex task, especially if the developer is not aware of all the possible dimensions, our approach assumes the existence of a knowledge base. In the next section, relations between dimensions, metrics, and metric measurements are defined to allow the developers to derive low-level requirements (e.g., VM Mem free, VM availability) starting from high-level requirements (e.g., VM status or VM performance).

3 Problem statement

Before introducing a formalization of a cloud provider offering and a developer request, we formalize the common elements of our framework: i.e., *dimensions*, *metrics*, and *metric measurements*.

Definition 1. *A dimension is one of the perspectives of the application that the developer is willing to quantify (e.g. "performance", "sustainability"). It is usually an high level requirement that can not be directly measured. It is defined by its name and a set of metrics used to evaluate the dimension:*

$$d_i \in \mathcal{D} =< name, \{m_j\} >$$

Definition 2. *A metric defines how to assess a dimension by measuring some phenomenon. For instance, "response time" and "availability" are metrics related to the dimension "performance".*

$$m_j \in \mathcal{M} =< name, f(mm_k) >$$

where name is the name of the metric and $f(mm_k)$ is the function used to compute the metric based on some measurements of the environment. They correspond to low level requirements.

Definition 3. *A metric measurement is a measurement of the monitoring system used to compose the value of a metric. It is defined as:*

$$mm_k =< name, type, samplingTime >$$

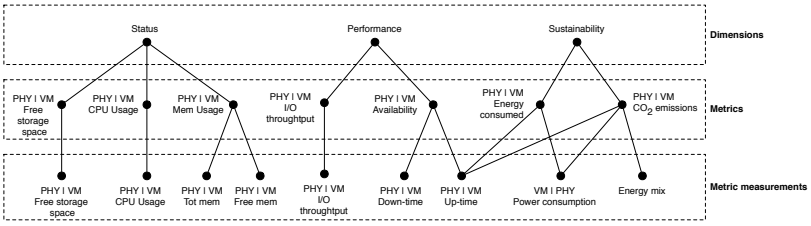


Fig. 2: Example of knowledge base.

As different cloud monitoring services can adopt different names to identify the same metric (e.g., as a basic example *CPUUtil* instead of *CPUUtilization*), in this paper we assume that both developers and cloud providers share the same vocabulary, thus no misunderstanding can occur in the match-making process. Anyway, the Cloud Broker can overcome such a limitation implementing existing techniques [14] for identifying similarities in names based on text analysis and domain-specific ontologies.

The set of relations between dimensions, metrics, and metric measurements constitutes a knowledge base shared by the developers and the cloud providers (Fig. 2). It provides information on how dimensions are defined: i.e., status (of both VM and physical servers PHY) can be assessed by CPU Usage, Mem Usage, and so on, while sustainability by the power consumed or the CO_2 emissions. For some metrics the computation requires more than one measurements (e.g. CO_2 emissions depend on the energy mix[5]), whereas for other metrics a direct measurement is possible (e.g., CPU Usage). Both cloud providers and developers browse the knowledge base to understand what to offer and what to request, but these two actors differ in the way they use it to express the requests and the offerings:

- Cloud providers, evaluating the leaves of the knowledge base, realize the coverage of the metric measurements given the installed monitoring infrastructure. Moreover, for each dimension (the roots) the provider is able to know which are the covered metrics and so to define the offerings. The knowledge base also provides a tool for providers to know the gap of their offerings with respect to a complete monitoring support.
- Developers use the knowledge base to select the dimensions or the metrics to be monitored. By working on the higher levels of the tree, it is not needed for the developer to know the details of the monitoring systems (i.e., the monitoring measurements). In this way, a developer can simply express in the request the need for measuring, for instance, the status of a VM. This implies that all the metrics linked to these dimensions should be supported. Alternatively, the developer could select only a subset of these metrics.

[5] Energy mix is defined as the proportion of the different power generation technologies, including fossil fuels, nuclear power, and renewable sources. Variation on this proportion has impact on the CO_2 emissions.

The knowledge base represents a common knowledge among the several parties involved in the match-making. It can be really implemented if a more formal agreement is required, or it can be considered as a tacit knowledge.

Definition 4. *A cloud provider offering for a site PO^s is composed of a collection of probes $\{p_l\}$ supported by the monitoring infrastructure offered by a provider s. Each sensor is defined as:*

$$p_l = \{\langle name, type, samplingTime, cost\rangle\}$$

The type of the sensor is a set of one or more of the three values $[A|M|E]$ that indicates: (A) the availability of the metric on the monitoring system; (M) the possibility to modify the monitoring system to also support the measurement of the metric; (E); the possibility to estimate the trend without modifying the monitoring system. The sampling time provides information about the frequency at which the measurement is collected. The cost associated to the metric provisioning is also specified.

The cost for a metric depends on the business model adopted by the cloud provider. Some may put the cost as 0 for sensors made already available by the monitoring system as its cost is included in the overall subscription. To have a fair comparison among the different offers, we assume that the cost of monitoring a metric is explicitly stated in the offering. It is also reasonable to assume that the cost for modifying the monitoring system to offer a metric (option M) implies a higher cost than the evaluation (option E).

Definition 5. *A developer request*

$$DR^d = \langle \bigvee_r mc_r^d, \bigvee_s cc_s^d[, cost]\rangle$$

is defined by a set of metric requests, a list of constraints, and, optionally, a maximum budget. Metric requests and constraints are expressed using the Disjunctive Normal Form (DNF). Each minterm represents an alternative, so that the request includes R admissible configurations for metrics and S configurations for constraints.

A metric configuration $mc_r^d = \bigwedge(\langle VM_{id}, m_t, samplingTime\rangle)$ includes the set of T metrics requested. Each of them specifies the VM to which it refers and the sampling time.

A constraint configuration $cc_s^d = \bigwedge(\langle VM_{id}, P_{id}\rangle)$ specifies where the VMs can be deployed. If a configuration does not include a VM then no constraints are imposed.

Formalization of the request and constraints in the offering using the DNF make the identification of the different valid alternatives easier as, by construction, only one minterm is true at the same time. For instance, assuming to have two cloud providers, i.e., P1 and P2, the constraints in the offering in Fig. 1 can be expressed as:

$(\langle VM1, P1\rangle \wedge \langle VM2, P1\rangle \wedge \langle VM3, P2\rangle) \vee (\langle VM1, P2\rangle \wedge \langle VM2, P2\rangle \wedge \langle VM3, P1\rangle)$

3.1 Optimization problem formulation

The mathematical model of our problem is described using the following sets:

- $\mathcal{V}$ is the set of all VMs. The cardinality of this set is V.
- $\mathcal{S}$ is the set of all sites. The cardinality of this set is S.
- $\mathcal{M}$ is the set of all metric measurements. The cardinality of this set is M.
- $\mathcal{MP}$ is the partition[6] induced by the set of all metrics in M. The cardinality of this set is MP.
- $\mathcal{SS}$ is the set of couples of VMs (v_0, v_1) such that VM v_0 must be deployed in the same site of v_1. The cardinality of this set is SS.
- $\mathcal{DS}$ is the set of couples of activities (v_2, v_3) such that VM v_2 must be deployed in a different site of v_3. The cardinality of this set is DS.
- $\mathcal{SR} \subseteq \mathcal{V} \times 2^{\mathcal{S}}$ is the set of couples $(v, \{s_0, \ldots, s_n\})$ such that VM v must be deployed in one of the sites $s_0, \ldots, s_n$.
- $\mathcal{MR} \subseteq \mathcal{V} \times \mathcal{M}$ is the set of all couples (v, m) such that we want to measure the metric measurement m for VM v.
- $\mathcal{S}(k) \subseteq \mathcal{S}$ is the set of all sites s such that we have a measure of metric measurement m.

For the parameters we will use the following notation:

- F_s cost of measuring from site s,
- $CI_{m,s}$ is the cost for implementing a probe for metric measurement m in site s,
- $CE_{m,s}$ is the cost for estimating a probe for metric measurement m in site s,
- $a^{(A)}_{m,s,v} \in [0, 1]$ is the accuracy of metric measurement m for VM v in site s,
- $\Delta a^{(E)}_{m,s,v} \in [-1, 1]$ is the variation of accuracy for VM v if we decide to evaluate a metric measurement m in site s,
- $\Delta a^{(MP)}_{m,s} \in [-1, 1]$ is the variation of accuracy for VM v if we decide to implement a probe related to metric measurement m in site s,
- β is the budget, i.e., maximum amount of money that we want to pay.
- α is the minimum accuracy that we ask.

We will use the following variables:

- w_s, binary variable, true if site s is used;
- x_{vs}, binary variable, true if VM v is deployed in site s;
- y_{ms}, binary variable, true if metric measurement m in site s is made because not available (option M);
- z_{ms}, binary variable, true if metric measurement m in site s is estimated because not available (option E);
- l, measuring how much we violate the budget constraint that we fix.

[6] This may not be trivial, e.g. in Fig. 2 we have the metric measurement up-time that contributes to two metrics. In this case, we consider *uptime* to be in the set defined by the metric with more value for the user.

Our problem [7] is then:

$$\text{maximize} \sum_{m \in MP_1} (\sum_{s=1}^{S} (\sum_{v=1}^{V} (a_{m,s,v}^{(A)} x_{v,s}) + \sum_{v=1}^{V} (\Delta a_{m,s,v}^{(E)}) z_{m,s} + \sum_{v=1}^{V} (\Delta a_{m,s,v}^{(M)}) y_{m,s})), \cdots$$

$$\text{maximize} \sum_{m \in MP_M} (\sum_{s=1}^{S} (\sum_{v=1}^{V} (a_{m,s,v}^{(A)} x_{v,s}) + \sum_{v=1}^{V} (\Delta a_{m,s,v}^{(E)}) z_{m,s} + \sum_{v=1}^{V} (\Delta a_{m,s,v}^{(M)}) y_{m,s})),$$

$$\text{minimize } l$$

subject to:

$$w_s \geq x_{v,s} \quad \forall \ \ v \in \mathcal{V}, \ s \in \mathcal{S} \qquad (1)$$

$$\sum_{s=1}^{S} x_{v,s} = 1 \quad \forall \ \ v \in \mathcal{V} \qquad (2)$$

$$y_{m,s} \leq \sum_{v=1}^{V} x_{v,s} \quad \forall \ \ s \in \mathcal{S}, \ m \in \mathcal{M} \qquad (3)$$

$$a_{m,s,v} \leq 1 - z_{m,s} \quad \forall \ v \in \mathcal{V}, \ s \in \mathcal{S}, \ m \in \mathcal{M} \qquad (4)$$

$$z_{m,s} \leq \sum_{v=1}^{V} x_{v,s} \quad \forall \ \ s \in \mathcal{S}, \ m \in \mathcal{S} \qquad (5)$$

$$x_{v_0,s} = x_{v_1,s} \quad \forall \ \ s, (v_0, v_1) \in \mathcal{SS} \qquad (6)$$

$$x_{v_0,s} + x_{v_1,s} \leq 1 \ \forall \ s, (v_0, v_1) \in \mathcal{DS} \qquad (7)$$

$$\sum_{s \in (v,\{s\}) \in \mathcal{SR}} x_{v,s} = 1 \ \forall \ v, (v, \{s\}) \in \mathcal{SR} \qquad (8)$$

$$\sum_{s=1}^{S} F_s w_s + \sum_{m=1}^{M} \sum_{s=1}^{S} CE_{m,s} z_{m,s} + \sum_{m=1}^{M} \sum_{s=1}^{S} CI_{m,s} y_{m,s} = \beta + l \qquad (9)$$

$$\max_s \left[a_{m,s,v}^{(A)} x_{v,s} + \Delta a_{m,s,v}^{(E)} z_{m,s} + \Delta a_{m,s,v}^{(M)} y_{m,s} \right] \geq \alpha \quad \forall \ \ (v, m) \in \mathcal{MR} \qquad (10)$$

$$w_s \in \{0,1\} \quad \forall \ \ s; \quad x_{v,s} \in \{0,1\} \quad \forall \ \ v, s; \quad y_{m,s} \in \{0,1\} \quad \forall \ \ m, s$$

$$z_{m,s} \in \{0,1\} \quad \forall \ \ m, s; \quad l \in \mathbb{R}$$

The constraints have the following meaning:

1. If we deploy VM v in site s then we use site s.
2. All VMs must be deployed.
3. We can implement a probe relative to metric measurement m in site s only if we have a VM in that site.
4. We can estimate a metric measurements only if we don't have the measure.
5. We can ask for an evaluation of the metric measurement m in site s only if we have a VM in that site.
6. Some VMs must be deployed on the same site.
7. Some VMs must be deployed on different sites.
8. Some VMs must be deployed on a fixed set of sites.
9. We don't want to spend too much money.
10. We must measure some metric measurements for some VMs.

[7] The proposed problem formulation assumes that the utility of the decision maker can be well approximated by a linear function as it is reasonable to think that the second order iterations between the accuracy of different metric measurements is negligible.

Algorithm 1 Accuracy Computation (A)

Input: mm_m: the metric measurement to evaluate
Input: PO^s: the monitoring infrastructure offered in site s
Output: $a^{(A)}_{m,s,v}$: the accuracy of the measurement of mm_m in the site s

```
a^(A)_{m,s,v} = 0
for p_l ∈ PO^s do                                   ▷ Find the probe in PO^s measuring mm_m
    if mm_m.name == p_l.name & (A) ∈ p_l.type then ▷ the sensor provides a measured value
        a^(A)_{m,s,v} = min(1, p_l.samplingTime / mm_m.samplingTime)
    end if
end for
```

In order to have a MILP, we have to change (10) with other linear constraints. For the discussion we consider m, v fixed. Constraint (10) is equivalent to:

$$a^{(A)}_{m,1,v}x_{v,1}+\Delta a^{(E)}_{m,1,v}z_{m,1}+\Delta a^{(M)}_{m,1,v}y_{m,1} \geq \alpha \ \vee \ a^{(A)}_{m,2,v}x_{v,2}+\Delta a^{(E)}_{m,2,v}z_{m,2}+\Delta a^{(M)}_{m,2,v}y_{m,2} \geq \alpha \ \vee \cdots$$

$$a^{(A)}_{m,S,v}x_{v,S} + \Delta a^{(E)}_{m,S,v}z_{m,S} + \Delta a^{(M)}_{m,S,v}y_{m,S} \geq \alpha$$

this can be translate using linear expressions by introducing:

$$u_s = \begin{cases} 1, & \text{if } a^{(A)}_{m,s,v}x_{v,s} + \Delta a^{(E)}_{m,s,v}z_{m,s} + \Delta a(M)_{m,s,v}y_{m,s} - \alpha \geq 0 \\ 0, & \text{otherwise} \end{cases}.$$

that must satisfy $\sum_{s=1}^{S} u_s \geq 1$. Hence in order to fix the conditions about the behaviour of u_s we have to add the following constraints:

$$a^{(A)}_{m,s,v}x_{v,s} + \Delta a^{(E)}_{m,s,v}z_{m,s} + \Delta a^{(M)}_{m,s,v}y_{m,s} - \alpha \leq u_s$$

$$\alpha - a^{(A)}_{m,s,v}x_{v,s} - \Delta a^{(M)}_{m,s,v}z_{m,s} - \Delta a^{(M)}_{m,s,v}y_{m,s} \leq 1 - u_s$$

In the following paragraph we describe how the metrics accuracy and their variations needed for running the optimization algorithm are estimated using a probabilistic approach supported by a Bayesian network.

3.2 Quality of Data computation

The optimization is based on the accuracy of the measurability of each of the selected metrics $a^{(A)}_{m,s,v}$ that depends from several factors. Each metric is derived from the composition of several metric measurements, which can be available in the considered site, estimated, or implementable at a given cost. In case all the metric measurements related to a metric are available, the accuracy of the metric depends from the discrepancy between the required quality in terms of sampling time of the metric and the one required. In this case, the lower sampling time between all of the metric measurements is considered in the evaluation (worst case), as described in Algorithm 1.

The same approach can be used in case the implementation of a metric measurement is required (M), since the cloud provider declares the accuracy that will be provided. In this case, the gain in implementing a new sensor is

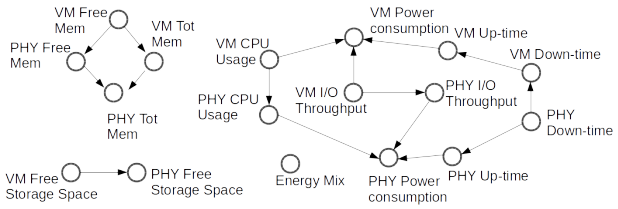


Fig. 3: Bayesian Network for metric measurement estimation

$\Delta a^{(M)}_{m,s,v} = a^{(M)}_{m,s,v} - a^{(A)}_{m,s,v}$, which is convenient only if a physical sensor is not yet implemented.

An additional level of abstraction is needed for the computation of the accuracy of an estimated metric measurement ($a^{(E)}_{m,s,v}$). In some condition, the collection of the data required by the developer can be not available without an additional cost that could bring the total amount of cost higher than the specified budget. However, in some cases, the cost of the implementation does not worth the benefit obtained, since for some metric the developer can be interested in trends more than in precise values. In such cases, an estimation is possible by modelling correlations through a Bayesian Network. The set of data used for its generation is obtained from the monitoring of all the sites in the multi-cloud environment, in order to derive general relations between metric measurements. The BN is composed of: (i) nodes, each one represents a metric measurement; (ii) edges, directed links that connect two nodes expressing a dependency between a parent and a child; (iii) Conditional Probability Tables (CPTs) associated to each node in the BN, quantifying the influence of the parents on the node. The CPT usually expresses conditional probabilities between each of the possible states of the child variable knowing the values of the parents. In this case we consider binary values, expressing the likelihood that a given metric measurement will increase given the trends of its parent set. An example of BN structure is shown in Fig. 3, using the metric measurements illustrated in Fig. 1. The BN is computed using the techniques described in [18]. It is created from the analysis of the correlation values between the metric measurements collected in all the sites and refined using the Max-Min Hill Climbing Algorithm [17] for links orientation. This inter-site BN enables making predictions about the trends even in sites where the specific measurement is not provided.

The accuracy of the estimated metric measurement can be obtained by combining the accuracy of the metric measurements from which it is derived and a likelihood value expressing the reliability of the dependency obtained by the CPT of the node. The accuracy computation of estimated metric measurements is described in Algorithm 2.

Algorithm 2 Accuracy Estimation (E)

Input: mm_m: the metric measurement to estimate
Input: PO^s: the monitoring infrastructure offered in site s
Input: BN: the correlation existing between the metric measurements
Output: $\Delta a^{(E)}_{m,s,v}$: the accuracy of the estimation of mm_m in the site s

$a^{(E)}_{m,s,v} = 0$
for $p_l \in PO^s$ **do** ▷ *Find the probe in* PO^s *measuring* mm_m
  if $mm_m.name == p_l.name$ & $(E) \in p_l.type$ **then** ▷ *the probe provides an estimated value*
    $ST = 0$
    for $mm_x \in BN.Parents(mm_m)$ **do**
      $ST = max(ST, mm_x.samplingTime)$
    end for
    $rel = p(mm_m | BN.Parents(mm_m))$
    $a^{(E)}_{m,s,v} = min(1, rel \cdot \frac{min(ST}{mm_m.samplingTime})$
  end if
end for
$\Delta a^{(E)}_{m,s,v} = a^{(E)}_{m,s,v} - a^{(A)}_{m,s,v}$

4 Validation

The optimization problem formulated in Section 3 has been implemented in C++ using the commercial solver Gurobi[8] for the solution of the MILP [9]. The main limitation coming from this choice is that we cannot modify the algorithm in order to use characteristic of the problem that can improve the speed. Nevertheless, this software is good enough to deal with the real instances and, for this reason, we do not implement our own algorithm. The optimizer has been executed on an Intel® Core™i7-5500U CPU @2.40 Ghz with 8 GB RAM and Microsoft® Windows™10 Home installed. To obtain reliable results, all the tests described in the following have been conducted 30 times.

As the optimization problem is NP-hard (complexity $\mathcal{O}(2^{\max[VS,MS]})$) the goal of this validation is to figure out how much the response time of the optimizer increases when varying the number of VMs (i.e., V) and the number of metrics (i.e., M). Due to the nature of the problem, a reasonable variation of the number of sites, i.e., S, does not affect strongly the response time as adding more sites means not only to add constraints and variables but also to increase the feasible space (the more sites available the more possible solutions). The case in which we have a high variation of the number of sites will be considered in future work as a more efficient algorithm is required (e.g. imposing a very good initial solution derived from the use of a proper heuristic).

Fig. 4 reports the response time of the optimizer varying the number of metric measurements included in the requests (from a min of 1 to a max of 17 according to the knowledge base presented in Fig. 2). Each curve corresponds to the response time required to obtain a solution with different set of VMs, each of them specifying a set of metric measurements in their request.

[8] http://www.gurobi.com

[9] Due to page restrictions we are not able to specify all the elements of the problem in this article. If interested, the reader can find all the problem specifications and the code for running the application at https://github.com/monicavit164/requirementMeasurementMILP

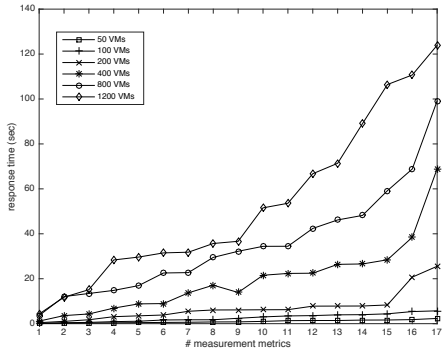


Fig. 4: Optimizer response time.

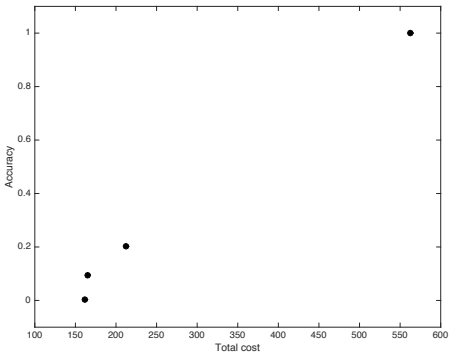
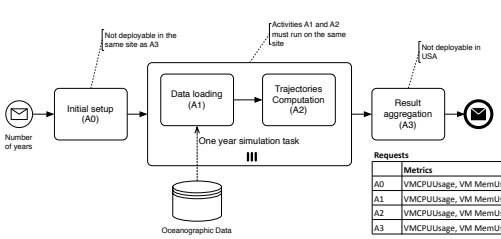


Fig. 5: Pareto front of Fig. 6 app.



Offerings

	Cost	Capabilities	Sampling time	Location
Site0	50	PHY Layer monitoring, no energy	60"-90"	UK
Site1	100	PHY+VM monitoring, no energy	30"-60"	USA
Site2	300	PHY Layer monitoring, yes energy	30"-60"	UK
Site3	400	PHY+VM monitoring, yes energy	30"-60"	UK
Site4	400	a livello PHY+VM e con energia	60"	USA
Site5	300	all metrics	120"	Italy
Site6	150	all metrics	600"	Italy

Requests

	Metrics	Sampling time
A0	VMCPUUsage, VM MemUsage, VM Availability, VM EnergyConsumed	120"
A1	VMCPUUsage, VM MemUsage, VM Availability, VM EnergyConsumed, VM I/O Throughput	120"
A2	VMCPUUsage, VM MemUsage, VM Availability, VM EnergyConsumed	60"
A3	VMCPUUsage, VM MemUsage, VM Availability, VM EnergyConsumed, VM Storagespace	60"

Fig. 6: Running example.

It is worth noting that the optimizer will be used at design time, when the developer wants to deploy the application. For this reason, a result is not necessarily required in seconds. Some minutes is also an acceptable response time. Anyway, the chart shows that the optmizer returns a solution in around 5 secs in case of 100 VMs with 17 metric measurements. In the worst case, two minutes are enough to compute a solution with 1200 VMs with the same number of metric measurements.

We also considered a real HPC application in the ecology domain [11] shown with a BPMN in Fig. 6. Without entering into the details, the application starts with an initial setup (activity A0). The work is then split into several instances composed of two activities: data loading (A1) and computation (A2). Once all the instances terminate, the partial results are aggregated (A3) to provide the result to the final user. We assume that one VM is required for A0 and A3, while for A1 and A2 the number of VMs may change according to the number of iteration required. Based on this example, Fig. 6 also includes the requests and offerings tables. The former reflects the requirements in terms of monitoring for each of the VMs composing the application. The latter reports the capabilities of the 7 cloud providers where the VMs can be deployed. For the sake of simplicity, the offering table does not report the detailed list of the supported metric measurements,

but a high level description (e.g., PHY layer monitoring means all the metric measurements at PHY level). For the sampling time we assumed that for the majority of the metric measurements the sampling time is equal.

Fig. 5 shows the Pareto front calculated with 4 VMs, 7 sites, and 7 metric measurements which took 19.2 secs. The curve has very few points as we have to fulfil some requests related to the metric measurements that we have to register hence we cannot go below some given price. Furthermore once that we have implemented all the probes we cannot improve more our solution. The discontinuous front derives from the adoption of a MILP. In order to compute the response time in Fig. 4 we have considered, for each point, ten random points in the Pareto Front by using several weighted sums of the M+1 objectives. The front in Fig. 5 considered 2 objectives and we compute point solution for a uniform grid of 10^4 points in the square $[0, 10]^2$. The choice of the square is related to the characteristic of the values of the two objective functions.

5 Related work

Current approaches for monitoring applications distributed in different cloud infrastructures are usually provider-centric and focus on solutions to hide the heterogeneity of the adopted monitoring platforms [2, 19] through a common interface. As in this paper we follow a multi-cloud approach, the perspective is client-centric and, in particular, the end-user is the application that coordinates the access and utilization of different cloud providers to meet the application requirements [16]. Here, the role of the Cloud Broker, as seen by the NIST [10], can provide intermediation services to facilitate the relationships between the cloud providers and the application (that holds the role of cloud consumer). In our case, the cloud broker enhances the deployment strategy of the cloud consumer making easier to find the cloud providers able to support the monitoring capabilities as needed by the application.

Some work in the state of the art has investigated the issue of modelling and in some cases discovering the relations between different metrics that can give a hint about the value of a missing metric, allowing the owner of the application to reason about the metric even if the real value is not directly provided by the monitoring system. The framework proposed in [9] looks for influential factors between metrics, represented in a dependency tree learned using machine learning techniques. The influential factors existing among indicators are statically and manually defined by the user. A study conducted by Google [5] employs a neural network framework that learns from monitored data to model and predict the outcome of some modifications over the monitored variables. A more complex and comprehensive approach has been proposed in [18]. Here, relations between the information collected at several levels of abstraction (monitored information and complex metrics) is represented in a Bayesian Network built automatically from the analysis of historical data, and kept updated through a continuous refinement. Even if in [18] the modelled relations are about satisfaction and dissatisfaction of constraints among metric values, this can be adapted to model relations about trends observed in the collected data provided by the

monitoring system. In this work we applied a modification of this approach to provide a prediction about missing metrics to the user.

The optimized deployment of VMs in a cloud environment can depend on several factors. In [6] a multi-objective algorithm is employed for VM placement in a cloud system. The algorithm minimize total resource wastage and power consumption providing a Pareto set of solutions. In [7], a greedy allocation algorithm is used to optimize the cloud provider's profit, considering energy efficiency, virtualization overheads, and SLA violation penalties as decision variables. In these approaches a single cloud provider is considered, thus measurability is not a relevant issue for the authors.

The relevance of the problem addressed in this paper is witnessed by the existence of several cloud platforms which differ in terms of set of metrics, sampling times, costs, and flexibility. About the possibility to extend, on user demand, the monitored metrics, in addition to the already mentioned Amazon CloudWatch and Paraleap CloudMonix, different monitoring solutions like Nagios, PCMONS, and Sensus, support the extensibility of the monitoring metrics[10] [1].

Moving to the knowledge base, semantic technologies are gaining more and more attention also in the cloud computing [15]. Focusing on the monitoring system, in [13] linked data are used to handle the heterogeneity of the collected data, whereas [4] provides a semantic meta-model for classifying dimensions and metrics.

6 Conclusion

In this paper, we have proposed an approach for supporting the deployment of multi-cloud applications where monitoring capabilities are taken into account. With a MILP problem, a cloud broker can figure out which is the best association among VMs composing the application and can make a request for some monitoring features, and for a cloud infrastructures providing some monitoring capabilities. A peculiar aspect of our approach relies on the possibility to extend the measurable metrics or to estimate the trends of metrics that are not supported by relying on other metrics. Estimation is based on a Bayesian Network able to infer how a metric changes with respect to other metrics. The deployment strategy proposed in this work balances between the cost for monitoring the application and the quality of the monitored data. The cost usually increases when the site offers a complete set of measurable metrics, thus with high quality of measured data. Conversely, the cost decreases for sites with limited set of measurable metrics that require an estimation of monitoring data, affecting the quality. The conducted experiments demonstrated the feasibility of the approach and, given the low response time, our optimizer can be adopted to facilitate the deployment of multi-cloud applications also composed of hundreds of VMs.

At this stage, the work focused on the IaaS multi-cloud provisioning model. Metrics considered in this work mainly refer to the physical and the virtualization layers. A complete set of metrics covering also the PaaS and SaaS provisioning models needs to be addressed in the future.

[10] https://www.nagios.org; https://code.google.com/p/pcmons; https://sensuapp.org

References

1. Aceto, G., Botta, A., de Donato, W., Pescap, A.: Cloud monitoring: A survey. Computer Networks 57(9), 2093 – 2115 (2013)
2. Alcaraz Calero, J.M., Knig, B., Kirschnick, J.: Using Cross-Layer Techniques for Communication Systems, chap. Cross-Layer Monitoring in Cloud Computing. Hershey: IGI Global (2012)
3. Dai, W., Chen, H., Wang, W., Chen, X.: RMORM: A framework of Multi-objective Optimization Resource Management in Clouds. In: Proc. IEEE SERVICES 2013. pp. 488–494 (June 2013)
4. Funika, W., Godowski, P., Pegiel, P., Król, D.: Semantic-Oriented Performance Monitoring of Distributed Applications. Computing and Informatics 31(2), 427–446 (2012)
5. Gao, J.: Machine Learning Applications for Data Center Optimization. Tech. rep., Google (2014)
6. Gao, Y., Guan, H., Qi, Z., Hou, Y., Liu, L.: A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences 79(8), 1230–1242 (2013)
7. Goiri, Í., Berral, J.L., Fitó, J.O., Julià, F., Nou, R., Guitart, J., Gavaldà, R., Torres, J.: Energy-efficient and multifaceted resource management for profit-driven virtualized data centers. Future Generation Computer Systems 28(5), 718–731 (2012)
8. Kaur, T., Chana, I.: Energy efficiency techniques in cloud computing: A survey and taxonomy. ACM Comput. Surv. 48(2), 22:1–22:46 (Oct 2015)
9. Kazhamiakin, R., et al.: Adaptation of Service-Based Applications Based on Process Quality Factor Analysis. In: ICSOC/ServiceWave 2009 Workshops (2010)
10. Liu, F., et al.: NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology (Special Publication 500-292). CreateSpace Independent Publishing Platform, USA (2012)
11. Melià, P., Schiavina, M., Gatto, M., Bonaventura, L., Masina, S., Casagrande, R.: Integrating field data into individual-based models of the migration of european eel larvae. Marine Ecology Progress Series 487, 135–149 (2013)
12. Petcu, D.: Multi-cloud: Expectations and current approaches. In: Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds. pp. 1–6. MultiCloud '13, ACM, New York, NY, USA (2013)
13. Portosa, A., Rafique, M., Kotoulas, S., Foschini, L., Corradi, A.: Heterogeneous cloud systems monitoring using semantic and linked data technologies. In: IFIP/IEEE Int'l Symp. on Integrated Network Mgmt. pp. 497–503 (May 2015)
14. Seco, N., Veale, T., Hayes, J.: An intrinsic information content metric for semantic similarity in Wordnet. In: Proc. Eureopean Conf. on Artificial Intelligence (ECAI'04), Valencia, Spain, August 22-27. pp. 1089–1090. IOS Press (2004)
15. Sheth, A., Ranabahu, A.: Semantic modeling for cloud computing, part 1. Internet Computing, IEEE 14(3), 81–83 (May 2010)
16. Toosi, A.N., Calheiros, R.N., Buyya, R.: Interconnected cloud computing environments: Challenges, taxonomy, and survey. ACM Comp. Surv. 47(1), 1–47 (2014)
17. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The Max-min Hill-climbing Bayesian network structure learning algorithm. Machine learning 65(1), 31–78 (2006)
18. Vitali, M., Pernici, B., O'Reilly, U.M.: Learning a goal-oriented model for energy efficient adaptive applications in data centers. Inf. Sciences 319, 152–170 (2015)
19. Zeginis, C., et al.: Towards cross-layer monitoring of multi-cloud service-based applications. In: ESOCC 2013, Malaga, Spain. Proceedings (LNCS 8135). Springer (2013)