Automatic Generation of Purchasing Plans for Cloud Services*

Octavio Martín-Díaz, José María García, Pablo Fernandez, and Antonio Ruiz-Cortés

Dpto. Lenguajes y Sistemas Informáticos
ETS. Ingeniería Informática – Universidad de Sevilla
41012 Sevilla, España – Spain
omartindiaz@us.es, josemgarcia@us.es, pablofm@us.es, aruiz@us.es

Abstract. The myriad of cloud service providers, as well as their overwhelming variety of configuration and purchasing options, result in a highly complex purchasing scenario. Furthermore, users may specify their needs for cloud services provisioning with a certain scheduling restrictions. There is a need for an automatic support for obtaining an appropriate purchasing plan, which takes into account both service configurations and scheduling needs, while allowing the comparison among different providers and their various offerings. In this work, we present an automatic purchasing plan generator, which analyzes cloud service offerings from several providers to obtain an optimized purchasing plan according to user needs. From the obtained purchasing plan, our solution can provide the corresponding charge plan, possibly including discounts, which serves the purpose of comparing offerings to get the best option.

Keywords: Cloud Services, Purchasing Plan, Discounts

1 Introduction

Nowadays, IaaS purchasing aims to obtain virtual processing and storage resources from cloud providers, as a way to reduce cost if compared with the procurement of on-premise, private compute infrastructures. However, the myriad of cloud service providers, as well as their overwhelming variety of configuration and purchasing options [7], result in a highly complex purchasing scenario. In this setting, there are major heterogeneity issues that make the comparison among providers rather difficult, e.g. different variables for configurations, additional purchasing variants apart from the usual on-demand option, billing and charge processes, and particular discount rules, to name a few.

* This work has been partially supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes under grants TIN2015-70560-R, P12-TIC-1867, and P10-TIC-5906.

On the other hand, users may also specify their needs for cloud services provisioning including specific scheduling restrictions. These restrictions provide additional beforehand information concerning not only the number of instances of particular configurations that are needed at a certain time, but also the amount of time they are going to be used. Purchasing plans can be derived from the scheduling restrictions, which in turn may affect the charge plans proposed by cloud providers, since they usually offer discounts depending on purchase volume.

There are some tools which implement the search for an optimal configuration, such as [4] and [5], according to particular user needs. However, these tools do not take into account scheduling. As a consequence, their results, though very valuable in smaller cases, are but a simplified version of the overall purchasing scenario. Therefore, there is a need for an automatic support for obtaining an appropriate purchasing plan, which takes into account both service configurations and scheduling needs, while allowing the comparison among different providers and their various offerings.

In this work, we present an automatic purchasing plan generator, which analyzes cloud service offerings from several providers to obtain an optimized purchasing plan according to user needs. From the obtained purchasing plan, our solution can compute the corresponding charge plan. Our solution is able to search for optimal service configurations and apply discount rules from different providers, providing facilities to compare their offerings to select the best option. We have developed a prototype implementation that has been validated in a particular scenario with three different providers.

The rest of the article is structured as follows. Section 2 introduces a case study that further motivates our work. Next, Section 3 describes the conceptual model of the purchasing process. Then, Section 4 presents the architecture of our solution along with our validation results. Section 5 discusses the related work. Finally, Section 6 concludes the article and outline our future work.

2 Motivation

In order to further motivate our work, we present a case study about the *virtualization of laboratory classes* in the context of our Software Engineering courses. Laboratory classes may have a dynamic evolution from two viewpoints: (1) the software being used on those classes may evolve, usually requiring increasing computing resources, and thus possibly rendering the corresponding hardware obsolete at short notice; and (2) the demand, due to the number of students, may vary along the academic year. In order to increase flexibility and save overall costs, these classes can be virtualized by purchasing cloud infrastructure to support their dynamic environment.

In this context, we need to clearly state which particular computational requirements we have in order to obtain an appropriate purchasing plan that we have to follow with some cloud infrastructure services provider. As an example, let us consider that we need to provide infrastructure for the laboratory classes

of a four-month course beginning on Monday 3rd October 2016, which requires a very simple hardware configuration of single core CPU and 1GB memory (in short "`Cpu:1 Mem:1`").

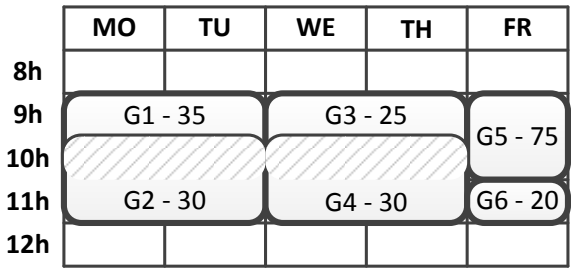

Fig. 1. Practices Scheduling

Figure 1 outlines the weekly sessions scheduling. There are 6 groups of students: G1 to G6. G1 to G4 have 4 hours-week (note that the sessions are overlapped between 10h and 11h), G5 has 2 hours-week, and G6 only has 1 hour-week. These groups have 35, 30, 25, 30, 75, and 20 students enrolled, respectively, which amount to 215 students. Additionally, there is a classroom open for students interested to further work outside regular sessions, which is planned to be open 6 hours-day from Monday to Friday for 10 students during 3 months, and 30 students during the 4^{th} month which is closer to the exams period.

Starting with the stated user needs including scheduling restrictions, we can derive a simple purchasing plan that lists the necessary purchasing actions that have to be carried out to fulfill those needs. Table 1 shows each action as a temporal event, enumerating the number of instances to be purchased, the term (or number of months during which the purchasing is valid), and the expected usage (expressed as hours per month). In this plan we are considering overlaps between the regular sessions as shown in Figure 1.

In order to actually provision the infrastructural needs for these laboratory classes, we need to carry out those particular purchasing actions against a cloud infrastructure service provider. Before selecting a particular provider, different offers should be compared, considering actual charges and possible discounts to be applied. Consequently, there is a need for computing expected charges with respect to user needs to enable comparison amongst providers. Our solution analyzes user needs, their associated purchasing plans, and available service offerings to obtain a corresponding charge plan, hence allowing the user to choose the best option in each case.

Table 1. Purchasing plan for our case study

Event time	**Instances** (number)	**Terms** (months)	**Usage** (hours/month)
Mon Oct 03 2016	20	4	5
"	75	4	10
"	55	4	10
"	25	4	10
"	30	4	20
"	65	4	10
"	35	4	10
"	10	3	160
Mon Jan 03 2016	30	1	120

3 Modeling the Purchasing Process

As motivated in Section 2, we need a plan generator to compute a purchasing plan from the user needs which, in turn, given the provider's offers, will determine the charge plan which, in the end, will make possible the comparison in order to select the most appropriate service. In order to automatically perform this process, we first need to model the relevant descriptions so that our solution can analyze and transform them into the resulting plan. In the following we present our modeling approach.

3.1 User Needs

User needs specifies the client's requirements on particular services (in our case study cloud infrastructure services, or IaaS in short). These requirements mainly state (1) the configurations which are needed to execute the client's software, and (2) the expected usage schedule .

Figure 2 shows our conceptual model representing user needs. User needs have a reference to the client's information, and a validity period which denotes the global time interval during which the IaaS is going to be used.

User needs are composed of several scheduling groups, which represent the particular usage schedule of certain service configuration [7].Essentially, each scheduling group is composed of scheduling items, which are "*temporal composites*" that detail the validity periods and number of instances of the same configuration that are needed. The validity period of a scheduling item is a time interval that may be periodic, and possibly disjoint or overlapped with others.

3.2 Purchasing and Charge Plans

A purchasing plan is composed of purchasing events. Each event comprises a series of purchasing actions that have to be performed at the same point of time. Conversely, each purchasing action is represented in Figure 3, including

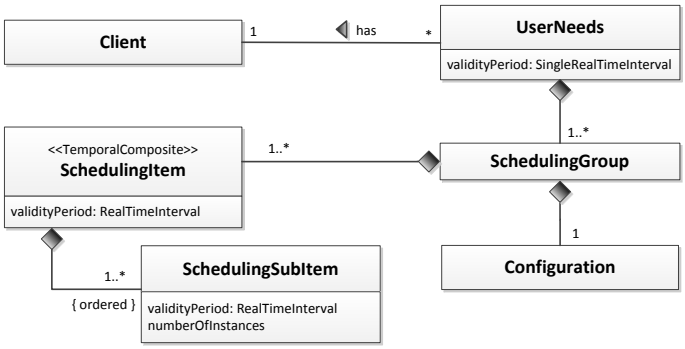

Fig. 2. Conceptual model for user needs

the required service configuration (i.e. memory and CPU in our case study), the purchasing type (prepayment, reservation, on-demand, or others), the expected usage (in hours/month), the number of instances to be purchased, and the term (number of months during which the purchase is going to be effective).

Initially, a purchasing plan is not associated with a particular provider. However, when searching for the most appropriate configuration from provider offerings, there could be specific information (e.g. base price) that would be stated by corresponding providers. Consequently, the generation of the specific purchasing plan and subsequent charge plan needs to be particularized for each provider.

Similarly, a charge plan is composed of charge events. In this case, our model specifies the point of time at which the actual charge is carried out, the type of charge (e.g. monthly payment or one-time payment), the monetary quantity, and a reference to the purchasing information that the charge is related to. Though it is not represented in the figure, purchasing and billing policies, including discount rules, are taken into account when generating charge plans from the purchasing plan, as discussed in the following section.

4 Prototype Implementation

In order to validate our approach, we developed a prototype solution that is based on the models described in Section 3. This solution is able to automatically derive purchasing plans and corresponding charge plans from different providers with regards to stated user needs. Figure 4 sketches the overall architecture of our solution.

On the one hand, service offerings are automatically imported in our system using a JSON-LD [10] parser that takes JSON files published by service

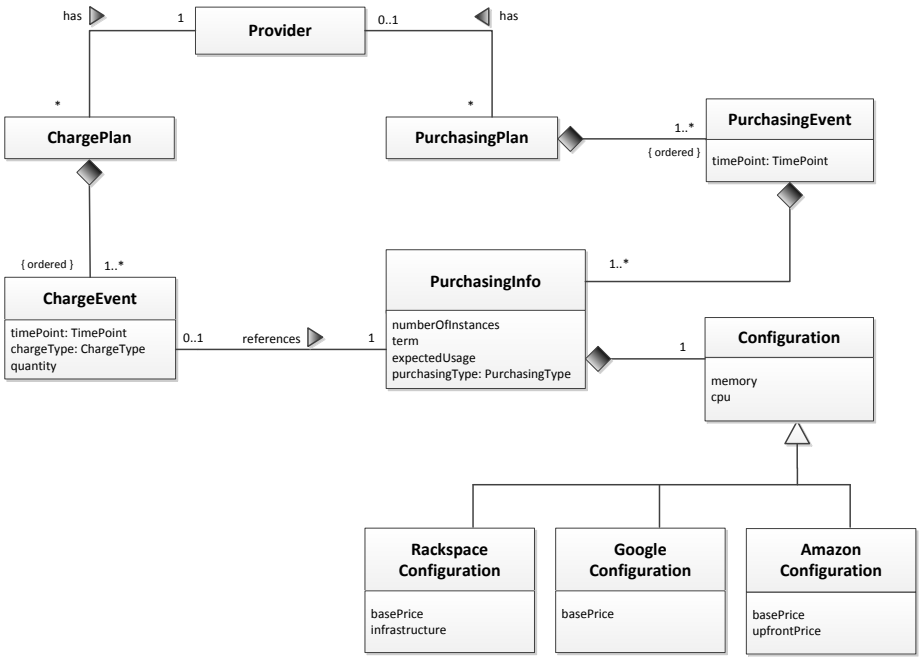

Fig. 3. Conceptual model for purchasing and charge plans

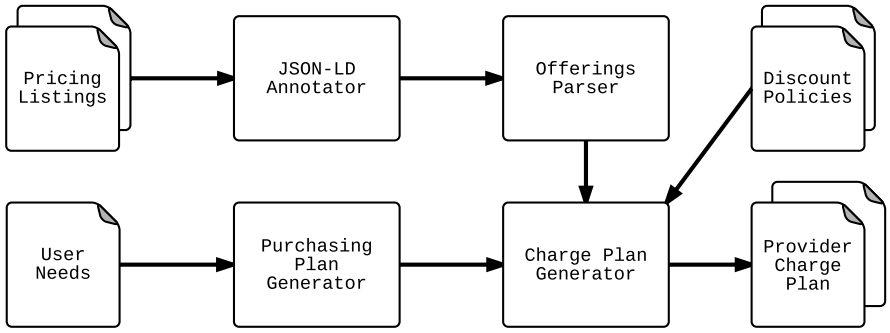

Fig. 4. Architecture of our automatic purchasing plan generator

providers, such as Amazon[1] or Google[2], and annotate some properties to identify common properties (such as base price, CPU, or memory) using JSON-LD facilities. We use the conceptual model sketched in Figure 3 as the fundamental schema to annotate pricing information from these providers, enabling interoperability of their original JSON schemas. Then, annotated JSON pricing listings are parsed in order to populate the catalog of service offerings from different providers.

On the other hand, user needs are instantiated according to the model discussed in Section 3. The purchasing plan generator component analyzes user needs and scheduling restrictions in order to obtain a specific purchasing plan, such as the one shown in Table 1. Generic purchasing plans need to be instantiated with respect to a specific service provider, so that a subsequent charge plan is generated. The charge plan generator component of our system perform this process, taking into account discount policies of each provider to finally obtain a specific provider charge plan that completely fulfill user needs.

In order to validate our solution, we carried out the case study described in Section 2 using the implemented prototype. We considered service offerings from Amazon, Google, and Rackspace. As stated by our user needs, our tool searched for the closer configuration to "`Cpu:1 Mem:1`" in their catalogs, using the approach presented in [7]. According to the *on demand* purchasing type, Table 2 summarizes the results of the search[3]:

Table 2. Selected configurations for our case study

Provider	Configuration	Base price	Infrastructure Price
Amazon	t2.micro	0.013$/hour	
Google	g1-small	0.027$/hour	
Rackspace	General1-1	0.032$/hour	0.005$/hour (Min 50$/month)

After applying each provider's policies regarding the purchasing process, including discount rules, Table 3 shows the corresponding charge plans:

In this case, Amazon is the best option. However if we elevate the requirements, for instance increasing the usage to 15 or 24 hours-day, requiring a more powerful configuration "`Cpu:16 Mem:120`" or "`Cpu:32 Mem:240`", or increasing the number of students, then it would be possible that the application of discount rules results in other providers becoming better options. Table 4 shows the resulting totals.

In this situation the best option depends on the particular case. Amazon is cheaper with smaller configuration, but it is the most expensive when strong requirements are needed, with Google offerings being the best positioned.

[1] `http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/price-changes.html`

[2] `https://cloudpricingcalculator.appspot.com/static/data/pricelist.json`

[3] Note that all prices considered were obtained on April, 20^{th} 2016.

Table 3. Charge plans for our case study

Amazon		Google		Rackspace	
Date	**Quantity**	**Date**	**Quantity**	**Date**	**Quantity**
Nov 01 2016	56.74$	Nov 02 2016	117.85$	Nov 03 2016	202.70$
Dec 01 2016	63.05$	Dec 02 2016	130.95$	Dec 03 2016	202.70$
Jan 01 2017	63.05$	Jan 02 2017	130.95$	Jan 03 2017	202.70$
Feb 01 2017	86.45$	Feb 02 2017	179.55$	Feb 03 2017	276.70$
Mar 01 2017	8.91$	Mar 02 2017	18.50$		
Total	**278.20$**	**Total**	**577.80$**	**Total**	**884.80$**

Table 4. Comparison between different variants of our case study

Case study variant	**Amazon**	**Google**	**Rackspace** (Prepayment)
4 hours-day (initial case study)	278$	577$	884$
9 hours-day	947$	1,958$	2,697$
15 hours-day ...	1,579$	3,218$	4,495$
24 hours-day ...	2,527$	4,889$	7,192$
... and Cpu:16 Mem:120	536,544$	185,528$	238,157$
... and Cpu:32 Mem:240	1,073,088$	365,057$	449,254$
... and Cpu:32 Mem:240 / x10 users	10,730,880$	3,650,573$	4,262,025$

As a conclusion, obtaining the best provider for certain user needs and scheduling restrictions depends on the actual configurations needed as well as the discount policies that providers offer. Our solution supports this process, facilitating the computation of purchasing and charge plans that can be compared among different providers.

5 Related Work

As far as we know, there is no approach to the purchasing problem which takes into account both heterogeneity of configuration, purchasing options, and discount rules, together with a required usage scheduling. However, there are related works which partially solve the purchasing and related problems.

First, previously cited tools such as [4, 5] compare providers given a configuration and some purchasing options, but discount rules and scheduling are not taken into account. From a workflow perspective, there are approaches to optimize the IaaS purchasing process supporting a workflow under scheduling restrictions [1, 6, 12, 13]. From an economic perspective, some studies on the cost of cloud services have been carried out [2, 3, 15], focussed on the analysis of investments to support the decision making during the purchasing process, but they do not address how the different purchasing options are formally expressed. From a quality perspective, in [8] authors provide a comprehensive model to

automate the ranking of different providers, but define a simplified version of cost and do not introduce a real purchasing option variability in their scenario.

Concerning purchasing models, similar limitations can be found in other related works such as [9] where authors present an initial review of different pricing models for cloud services that shows simple cost model scenarios, but elements such as purchasing processes variability or discounts are not discussed. For instance, in [16], authors present a formal language for cloud services.

There are also general comparisons among cloud providers, such as [14] whose author presents a comparison framework where different characteristics of the purchasing process, including the price model, are evaluated in order to obtain a classification of service providers.

6 Conclusions and Future Work

Purchasing plans are of utmost importance when trying to optimize computational resources required to fulfill some user needs during specific time periods. This article presents a solution to automatically derive purchasing plans from user needs specification including scheduling restrictions. After modeling user needs for a particular scenario, our prototype implementation searches for appropriate service configurations from different providers and generates the corresponding charge plan related to the purchasing actions needed to fulfill user requirements. Consequently, users can choose the best provider with respect to their needs.

As future work, we plan to further validate our approach, automatically crawling pricing and service configuration options from other providers, such as Microsoft of Heroku. Moreover, we will integrate our current solution with previous work in order to optimize purchasing plans and hence obtain more competitive charge plans [11]. Finally, we are evaluating rule-based approaches to generalize the definition of discount rules from service providers.

Acknowledgements

Authors would like to thank Manuel Arenillas for his support on the prototype implementation.

References

[1] Anbazhagi, Latha Tamilselvan, and Shakkeera. QoS based Dynamic Task Scheduling in IaaS Cloud. In *2014 IEEE International Conference on Recent Trends in Information Technology (ICRTIT)*, 2014.

[2] Slaven Brumec and Neven Vrček. Cost effectiveness of commercial computing clouds. *Information Systems*, 38(4):495–508, 2013.

[3] Se-Hak Chun and Byong-Sam Choi. Service models and pricing schemes for cloud computing. *Cluster Computing*, 17(2):529–535, 2014.

[4] Cloudorado.com. Cloud Computing Price Comparison Engine. `http://www.cloudorado.com/`.

[5] CloudScreener.com. Cloud Computing Comparison and Evaluation. `http://www.cloudscreener.com/`.

[6] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads. In *2010 IEEE 3rd International Conference on Cloud Computing*, 2010.

[7] Jesús García-Galán, Pablo Trinidad, Omer F. Rana, and Antonio Ruiz-Cortés. Automated Configuration Support for Infrastructure Migration to the Cloud. *Future Generation Computer Systems*, 2015.

[8] Saurabh Kumar Garg, Steve Versteeg, and Rajkumar Buyya. A framework for ranking of cloud computing services. *Future Generation Computer Systems*, 29(4):1012–1023, 2013.

[9] Sahil Kansal, Gurjeet Singh, Harish Kumar, and Sakshi Kaushal. Pricing models in cloud computing. In *ACM International Conference on Information and Communication Technology for Competitive Strategies (ICTCS)*, page 33, 2014.

[10] Markus Lanthaler and Christian Gütl. On using JSON-LD to create evolvable restful services. In Rosa Alarcón, Cesare Pautasso, and Erik Wilde, editors, *Third International Workshop on RESTful Design, WS-REST '12, Lyon, France, April 16, 2012*, pages 25–32. ACM, 2012.

[11] O. Martín-Díaz, P. Fernandez, P. Trinidad, and A. Ruiz-Cortés. Apoyo a la toma de decisiones en la compra de iaas. In *10th Jornadas de Ciencia e Ingeniería de Servicios (JCIS'14)*, pages 179–188, Cádiz, Spain, 2014.

[12] Nuttapong Netjinda, Booncharoen Sirinaovakul, and Tiranee Achalakul. Cost Optimal Scheduling in IaaS for Dependent Workload with Particle Swarm Optimization. *The Journal of Supercomputing*, 68(3):1579–1603, 2014.

[13] M. K. Nivodhini, K. Kousalya, and S. Malliga. Algorithms to Improve Scheduling Techniques in IaaS Clouds. In *2013 IEEE International Conference on Information Communication and Embedded Systems (ICICES)*, 2013.

[14] Thoran Rodrigues. 11 Cloud IaaS Providers Compared. `http://www.techrepublic.com/blog/the-enterprise-cloud/11-cloud-iaas-providers-compared/`.

[15] Bhanu Sharma, Ruppa K Thulasiram, Parimala Thulasiraman, Saurabh K Garg, and Rajkumar Buyya. Pricing cloud compute commodities: a novel financial economic model. In *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 451–457, 2012.

[16] Rafael B. Uriarte, Francesco Tiezzi, and Rocco De Nicola. SLAC: A formal Service-Level-Agreement Language for cloud computing. In *IEEE/ACM th Int. Conf. on Utility and Cloud Computing (UCC)*, pages 419–426, 2014.