INFORMATICA, 2015, Vol. 26, No. 1, 1–15

DOI: http://dx.doi.org/10.15388/Informatica.2015.35

A Distributed Cloud Brokering Service

Alba AMATO*, Beniamino DI MARTINO, Salvatore VENTICINQUE
Department of Industrial and Information Engineering
Second University of Naples, via Roma 29, 81031 Aversa (CE), Italy
e-mail: alba.amato@unina2.it, beniamino.dimartino@unina.it, salvatore.venticinque@unina2.it

Received: February 2014; accepted: August 2014

Abstract. The brokering of the best Cloud proposals that optimizes the application requirements allows to exploit the flexibility of the Cloud programming paradigm by a dynamically selection of the best SLA, which is available into the market. We present in this paper a scalable multi-users version of a Broker As A Service solution that uses the available resources of a distributed environment, and addresses related issues. The brokering problem is divided into simpler tasks, which are distributed among independent agents, whose population dynamically scales together the computing infrastructure, to support unforeseeable workloads produced by the interactions with large groups of users. The brokering model and its implementation, which adopts Cloud technologies itself, are described. Performance results and effectiveness of the first prototype implementation are discussed.
Key words: cloud brokering, service oriented architecture, cloud infrastructure, parallel computing.

1. Introduction

Cloud computing is already having a profound impact on the information technology. It represents a technology of enormous scope for innovation especially in a volatile and rapidly evolving market, in which the ability to exploit new technologies and remaining competitive is more important. In fact Cloud computing enables companies to adopt flexible and scalable solutions reducing infrastructure costs and giving the possibility to access resources and applications from remote. In this way it is possible for an enterprise to turn fixed costs into variable ones. However to optimize, enhance and simplify the management and use of information technology, adopting new technologies based on Cloud Computing, it is necessary the choice of Cloud providers, whose offers best fit the requirements of a particular application. This is a complex issue due to the variety of potential options and to the high number of criteria to consider, which are described using different metrics. Moreover service properties, general terms and conditions of service and service levels can change from one provider to another, as described in Amato and Venticinque (2013). In order to address the problem of the heterogeneity of Cloud services and technologies we have designed and developed Cloud Agency, a Multi-Agent System (MAS) that has the main task of dynamically selecting a set of Cloud resources, from different vendors, that

*Corresponding author.

best fits users' requirements. It also allows for vendor agnostic management and monitoring of Cloud infrastructures, where legacy or mOSAIC application are deployed as presented in Di Martino *et al*. (2011). It is compliant with the NIST definition of cloud broker as defined in NIST (2012), that is an entity that manages the use, performance and delivery of cloud services, and negotiates relationships between Cloud Providers and Cloud Consumers. Cloud Agency actively performs cloud broker functions at the platform level. In Amato *et al*. (2013) we demonstrated the feasibility of the specific approach in the case of a single user, considering the number of existing providers in the current Cloud market, in the case that all the providers have at least one proposal. However the centralized approach is not effective in the case of brokering delivered as a service to a huge number of users. To provide the brokering facility at Application as a Service level we have to overcome the performance limitations of the mOSAIC Cloud Agency solution, starting from the identification of new requirements. First of all it needs to identify the issues introduced by the new scenario, but also to take in consideration the available technologies for designing and implementing a new engineered solution. About the issues, we need to take into account the number of service users who can access contemporary the service by multiple requests. We have to consider that in a service oriented context, not only human users, but also applications and robots will be able to invoke the service producing different kind of workloads. The new workload can vary in dimension, but also it can dynamically change during the day, with regular or unforeseeable bursts on special periods. For this reason we have to share the workload over a distributed computing infrastructure, and we need to grant that both the infrastructure and the application will scale dynamically.

The Cloud technology is a promising solution to build a scalable computing infrastructure, but it needs to re-design the agents in order to let them exploit such an elastic computing model. In fact while the Cloud allows to scale the computing resources according to the measured workload improving also the utilization, the application must be able to reconfigure itself autonomically to keep the QoS level above the desired threshold. Hence we are going to build a distributed Cloud broker over a Cloud infrastructure. Not only performance issues should be addressed, but also reliability and availability when the service execute over such kind of distributed platform, due to lack of control both of the network and of the compute utility. In this paper we present a scalable distributed multi-users version of the brokering service provided by Cloud Agency. Such new Broker As A Service solution exploits the capability of a distributed environment and addresses related issues. The idea is to divide the brokering problem into simple tasks, which are distributed to independent and collaborative agents, whose number dynamically scales, together the computing infrastructure, to support unforeseeable workloads produced by the interactions with large groups of users. Besides we evaluate and discuss performance results and effectiveness of a first prototype implementation.

The paper is organized as follows. In Section 2 we present an overview of works related to multi-cloud resource brokering and negotiation. In Section 3 we introduce the Cloud Agency solution. The broker model is defined in Section 3. Architecture design and prototype implementation of the broker as a service are described in Section 4. In Section 5 we discuss experimental results. Finally conclusions are drawn.

2. Related Work

The brokering of Cloud providers whose offers can meet the requirements of a particular application is a complex issue due to the different business models that are associated with such computing systems. According to the National Institute for Standards and Technology NIST (2012), a Cloud broker can provide services in three categories:

- Service Intermediation: A Cloud broker enhances a given service by improving some specific capability and providing value-added services to Cloud consumers. The improvement can be management access to Cloud services, identity management, performance reporting, enhanced security, etc.
- Service Aggregation: A Cloud broker combines and integrates multiple services into one or more new services. The broker provides data integration and ensures the secure data movement between the Cloud consumer and multiple Cloud providers.
- Service Arbitrage: Service arbitrage is similar to service aggregation except that the services being aggregated are not fixed. Service arbitrage means a broker has the flexibility to choose services from multiple agencies. The Cloud broker, for example, can use a credit-scoring service to measure and select an agency with the best score.

There were some efforts aimed at solving the problems of Service Intermediation, Service Aggregation and Service Arbitrage.

SLA@SOI, available at Sla@soi (2012), is the main project that aims at offering an open source based SLA management framework, which will provide benefits of predictability, transparency and automation in an arbitrary service-oriented infrastructure, being compliant with the OCCI standard. SLA@SOI offer solutions to design Cloud services with multi-level and multi-provider SLAs. It does not provide a proper brokerage service but it represents an important effort in standardization.

In Nair *et al*. (2010) it is presented an architectural design of a framework capable of powering the brokerage of Cloud services that is currently being developed in the scope of OPTIMIS, an EU FP7 project. In this work a broker is used to serve the needs of several different requirements. In particular it is used to ensure data confidentiality and integrity to service customers, to match the requirements of Cloud consumer with the service provided by the provider, to negotiate with service consumers over SLAs, to maintain performance check on these SLA's and take actions against SLA violation, to effectively deploy services provided by the Cloud provider to the customer, to manage the API so that provider does not learn anything about the identity of the service consumer, to securely transfer customer's data to the Cloud, to enforce access control decisions uniformly across multiple Clouds, to scale resources during load and provide effective staging and pooling services, to securely map identity and access management systems of the Cloud provider and consumer, to analyze and take appropriate actions against risks, to handle Cloud burst situations effectively. The cited paper introduces the problem and the architectural design, but it does not provide an implementation or algorithms to achieve the brokering.

In Buyya *et al*. (2010) an architecture is presented for a federated Cloud computing environment named InterCloud to support the scaling of applications across multiple Cloud

providers using a Cloud Broker for mediating between service consumers and Cloud coordinators for an allocation of resources that meets QoS needs of users.

The Vordel Cloud Service Broker, available in Vordel (2012), provides a mechanism for securely integrating local on-site applications with off-site cloud service, key value-added services and enabling monitoring, management and policy enforcement for all transactions. It allows organizations to apply a layer of trust onto their Cloud Computing applications. It brokers the connection to the Cloud infrastructure, applying governance controls for service usage and service uptime. The Vordel Cloud Service Broker sits between the Organization and the Cloud service provider. It can be deployed as software or an edge device, brokering connections to the Cloud. Additionally, it may be deployed "Cloud-Side", as an Amazon EC2 instance.

In Tordsson *et al*. (2002) the author explores the heterogeneity of Cloud providers, each one with a different infrastructure offer and pricing policy, in a Cloud brokering approach that optimizes placement of virtual infrastructures across multiple Clouds and also abstracts the deployment and management of infrastructure components in these Clouds. Besides he presents a scheduling algorithm for cross-site deployment of applications. However he presents a fine grained interoperability of Cloud services by means of a Cloud API that do not take into account the different implementation models for the virtual machine manager (VMM), which are at the base of each of the Cloud providers' infrastructure.

The JamCracker system, available in Jamcracker (2012), provides a platform where it aggregates and distributes on-demand services through a global ecosystem of Service Providers, Resellers, System Integrators, and ISVs but it does not provide a proper brokerage service whereby an entity looks at the actual QoS requirements of the service under question, the various IPs that could potentially meet them, rank them against parameters like cost, trust, eco-efficiency, risk etc. and provide functionality to on board these applications in to the various IPs finally selected as stated in Optimis-project (2011).

Within the mOSAIC project, we designed and implemented a reference set of APIs, which intend to be language independent and programing paradigm free. For this reason Cloud Agency implements a multi-agent brokering mechanism, described in Amato *et al*. (2012b), that is vendor agnostic and allows for the deployment of mOSAIC applications on any Cloud infrastructures.

In preliminary work, explained in Amato and Venticinque (2013), the authors present the architecture of the Broker Agent and its implementation in Cloud Agency for provisioning of brokering service at Cloud platform level. In this paper we discuss further research in this topic addressing the performance problems in the case of brokering delivered as a service and proposing a scalable, multi-user, distributed solution and showing the feasibility of such approach.

3. Cloud Agency Brokering

Cloud Agency, presented in details in Venticinque (2013), is a Multi-Agent System that complements the common management functionalities which are currently provided by

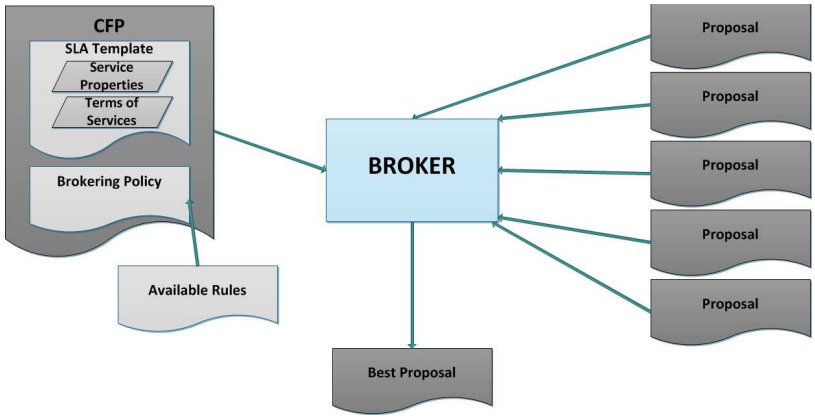

Fig. 1. Broker.

Private and Public Infrastructure as a Service (IAAS) with new advanced services, by implementing a Vendor Agnostic layer. The Provisioning service of Cloud Agency implements a minor modification of the original FIPA Contract-Net protocol described in Technical report (2002) that adds rejection and confirmation communicative acts. For a given task, one agent (the Initiator) takes the role of manager, asking for a service, and any number of Participants may respond with a proposal; the rest must refuse. Negotiations then continue with the Participants that returned valid proposals. For each received CFP Cloud Agency creates a broker that searches for vendors that can offer resources with the required QoS (Quality of services). SLA brokering is part of the agent based provisioning service of Cloud Agency. The broker collects a number of proposals described in an vendor agnostic way and chooses the best one(s) according to the brokering rules. The Call For Proposal (CFP) is the document to be prepared by the customer to specify his requirements in terms of the list of resources to be acquired and the rules/policies to be used for defining resource brokering strategies.

As shown in Fig.1, the CFP is composed of two sections. The first one is the *SLA Template* described according to the XML SLA@SOI schema described in Sla@soi (2012). The second section composing the CFP is the *Broker Policy*, containing a set of rules, to be enforced by the brokering algorithm, in order to choose among the different proposals offered by the Cloud market. In particular the SLA template, described in Amato *et al.* (2012a), is composed of *Service Properties*, that defines the technical requirements for user's applications; and the correspondent desired *Service Levels*, such as availability, reliability, performance; (*Terms of Service*) that include the contract duration, data location and billing frequency, etc.

Broker policy sets *constraints* and *objectives* on multiple parameters such as the best price per time unit, the greatest number of cores, the best accredited provider or the minimum accepted availability. As different proposals will come from Cloud Vendors, the broker have the main task to choose the best proposal according to the policies specified by the customer such as best price per time unit, maximum amount of memory, service availability and so on. In order to consistently develop a Cloud service broker, we propose

Table 1
Rules types.

Rule's name	Value type	Boolean expression
Exact match	Numerical & non-numerical	$t_i = s$
Value in a set	Numerical & non-numerical	$t_i \in S$
Greater then	Numerical	$t_i > s$
Less then	Numerical	$t_i < s$
Value in a range	Numerical	$t_i \in R$

a model to formulate the application requirements into constraints that can be architectural constraints and service level constraints and that can be divided into *hard* constraints and *soft* constraints. User selects properties, which characterize the specific class of chosen service; service levels in terms of performance, availability, etc.; the cost that he intends to pay for; the accreditation of the provider, which represents its reputation measured by the feedback of other users or by some rating agency. For each parameter the user eventually chooses some constraints, defines if they have to be hard or soft and specifies none or more objective functions to be optimized. The rules are chosen by selecting the SLA parameters and setting the required options using a friendly graphic interface. Simple constraint rules are in Table 1.

Of course not every constraint can be applied to any SLA parameters.

Given a set of constraints, it is possible that there are several contrasting objectives (e.g. the minimization of the cost, and maximization of the resources) so it is necessary a multi-objective approach to find the Pareto front (that is a set of all those solutions that are considered to be optimal in multi-criteria optimization). After that, a posteriori approach is used that deliver to the user the set of Pareto-optimal solutions among which the user will choose the preferred one. Nevertheless, in order to simplify the usage of the brokering service we allow for grouping multiple objectives according to the kind of SLA parameter *Service Properties*, *Terms of Services* or *Service Levels*. We also define the *Provider Reputation* as an additional brokering parameter, that is out of the SLA Template, but it is known to the broker. To compute the overall score we map the domain of each SLA parameter and we allow to assign a percentage relevance to each category.

In order to evaluate the best proposal the broker can use a set of rules $R = C \cup O$, which can be constraints rules $cr \in C$ or objectives rules $or \in O$ rules. Constraints rules are boolean expression that can represent soft or hard requirements:

$$cr : (t_i, c_i, m_i) \rightarrow [true, false], \tag{1}$$

t_i is an SLA term, c_i is a boolean expression, and m_i specifies that the constraints is hard (when 1) or soft (0). Objectives rules assign a score between 0 and 1 to the compliance of the SLA value of the term t_i with the correspondent user's requirements

$$or : (t_i, f_i, o_i) \rightarrow [0, 1]. \tag{2}$$

For each objective rule the user has to select a mapping function f_i between the t_i values and the correspondent score, and has to specify if that rule is an explicit ($o_i = true$) or

implicit objective ($o_i = false$). The mapping function change the way to evaluate that objective. For example logarithmic, linear, and exponential function can be used to define the relevance of that objective according to the value $v_{j,i}$ offered by the provider and the one desired by the customer.

The broker policy will be a subset of rules $R' \subset R$ that is defined for those terms of the SLA template, which are relevant for the user's requirements.

To solve the brokering problem we have to perform the following computation for each received proposal by replacing t_i with the correspondent value $v_{j,i}$:

- $M_j = \prod_{i=1}^{n} \neg(\neg c_i \wedge m_i)\ \forall j = 1, \ldots, m$
  that is used to check if the SLA proposal can be considered as a valid candidate for the SLA, in fact at least a false mandatory constraint invalidates that offer.
- $Opt_j = \sum_{i=1}^{n} (\neg m_i * (c_i = true))\ \forall j = 1, \ldots, m$
  evaluates how many soft constraints are met. It can contribute to the evaluation of the proposal.
- $V_j = \sum_{i=1}^{n} ((o_i = false) * f_i(v_{j,i}))\ \forall j = 1, \ldots, m$
  represents an overall evaluation for all those terms which have not to be negotiated independently.

All objectives rules which have $o_i = true$ will be considered independent objectives. In general the best proposals will be the ones which solve the following equations:

$$\max_{j=0}^{m}(or_i) : o_i = true \quad \text{and} \quad M_j = 1, \quad \forall i = 1, n. \tag{3}$$

An additional criteria will be:

$$\max_{j=0}^{m}(Opt_j) : M_j = 1. \tag{4}$$

All the defined criteria can be grouped if the user set $o_i = false$, $\forall i = 1, \ldots, n$, and $m_i = false$, $\forall i = 1, \ldots, n$. In this case the result of brokering will be:

$$\max_{j=0}^{m}(V_j) : M_j = 1, \tag{5}$$

that means the best proposal are the ones with the best overall score.

4. Architecture Design and Implementations

The main assumption we will take as strict requirement here will be the design of stateless and asynchronous scheduling of agents which will be able to run on every available computing resources. On the other hand the front-end will be implemented by a service interface that accepts synchronous requests. It stores the new pending tasks to be handled by pools of asynchronous working agents, and returns the current status of service

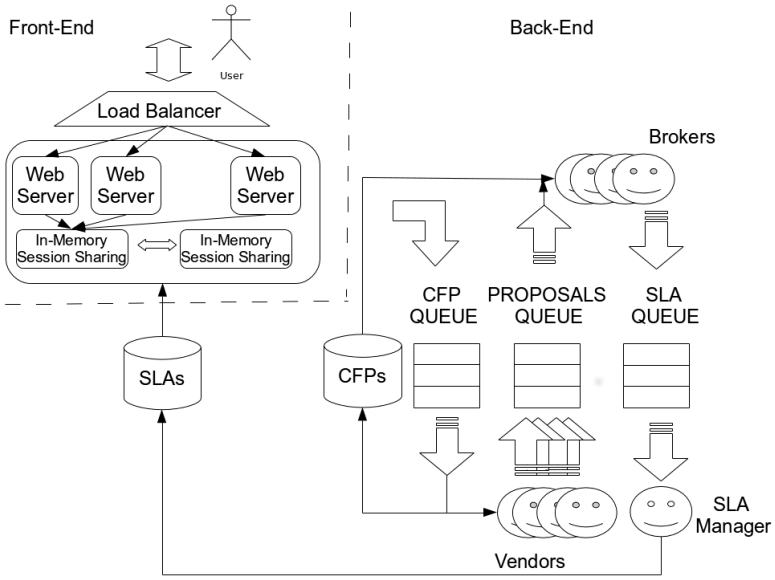

Fig. 2. Distributed architecture.

elaboration. The service architecture is shown in Fig. 2. In Fig. 2, in the upper left corner, we can see many service instances that receive requests from end users and access in-memory shared information. The in-memory session manager will allow for the exploitation of elastic capability of the Cloud infrastructure. The warm copy of the session manager allows for improving reliability.

As it is shown in Fig. 2 agents will implement the back-end of the new agency, that is completely relieved of handling interactions with clients. Cloud storages (database ad queues) keep persistent information of the distributed applications and implement communication channel between synchronous front-end handlers and back-end workers. This design choice allows for the easy distribution of the workload using a task parallel programming model. Stateless agents take new problems by a common bag of task, execute wherever there are available computing resources, and update the computing results if they complete successfully. Reliability is addressed by re-scheduling of unsolved problems which remain in the bag because of any failures or delay. The vendors agent sign up to the CFP_QUEUE to receive the cfps received from users. Each of them submits its proposal to the PROPOSALS_QUEUE. Idle brokers are waiting for proposals. A single proposal is dispatched to one broker that executes its matching with the correspondent CFP for evaluation purpose. The matching results is stored into the SLA_QUEUE if it belongs to the Pareto front of optimal solutions. ActiveMQ has been used as queue services for communication and synchronization. Brokering results are stored also into an in-memory session, together with information of each related user's request.

Users are provided with the web interface shown in Fig. 3. It allows for composing the CFP and listing, for each session, the best SLAs according to different brokering objectives. Users log into a web page, the web page makes a request to a REST service and

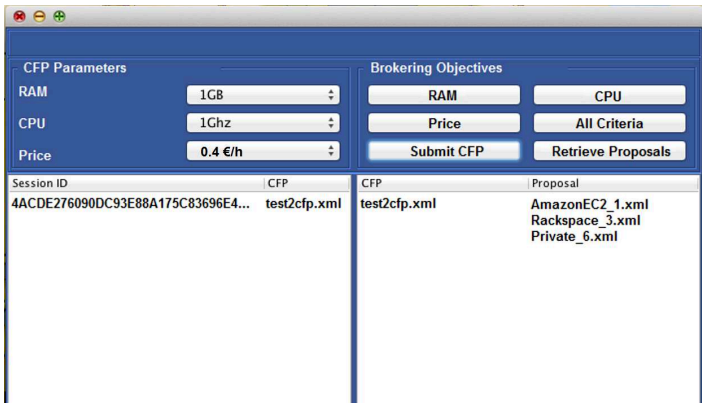

Fig. 3. Web GUI.

submits the call for proposals. The Call for Proposal are included, along with information of the session and the user, in CFP queue. The characteristic of the CFP queue is that all consumers that join the queue, receive all the submitted CFPs. Apache Tomcat has been used as web and application server to run the web service at front-end. It has been specifically configured for working with the Terracotta Framework for the transparent distribution and sharing of web sessions. Jersey API have been used to implement the RESTFull service that provide methods for authentication, CFP submission and SLA retrieval. Meanwhile their requests are pending, users can wait for the result of brokering or may poll periodically to get the status of their request. When one of the brokers has found a feasible proposal for that request the current results are updated. The completion of brokering is notified when there are no more proposal candidated to optimize the user's query to be evaluated. The RDBMS Mysql databases is used as persistent storage of CFPs and to SLAs.

5. Experimental Results

In order to evaluate performances of the proposed approach we set up the following testbed. A Linux physical machine hosts the ActiveMQ 5.6 service, with a Topic, named CFP_QUEUE, that receives CFPs from concurrent clients, which run on a different physical machine in the same 1 GB Ethernet local network. The server is 64 bit Intel Core^{TM}2 Quad Processor Q9300 (6M Cache, 2.50 GHz, 1333 MHz FSB) with 4 GB RAM. Oracle Java7 is the runtime environment. Concurrent clients send 10 CFPs, each one, according to a Poisson process with different mean time of arrivals. Vendors run on the server and wait for incoming CFPs. All vendors get the same CFP and generate their proposal, which is sent to the PROPOSALS_QUEUE. In a first scenario all brokers run on the server itself. They receive a different proposal from the QUEUE and evaluate the compliance with the correspondent CFP. The result is sent to an SLA_QUEUE from which only the best ones are notified to the clients. We evaluate the performance of such configuration changing

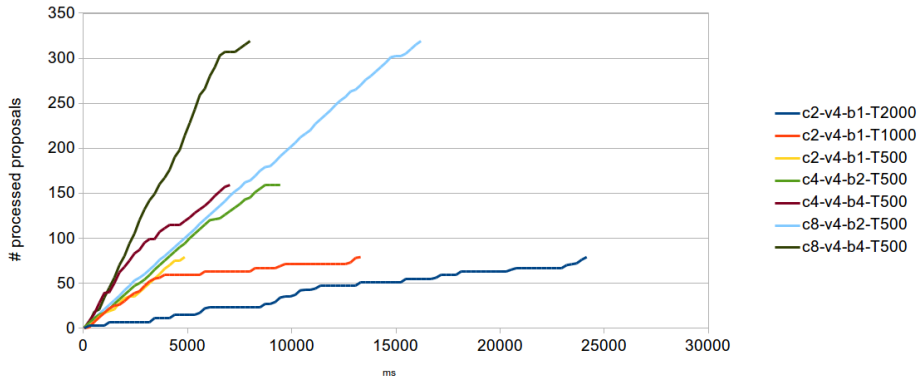

Fig. 4. Proposals processing.

the number of clients, the number of vendors and the number of brokers. All the measures are taken at server side. In the following we discuss the most significant figures. First of all in Fig. 4 we have, for each experiment the number of evaluated proposals per millisecond. The first series of Fig. 4 shows the case of 2 clients that send CFPs with a mean time of arrivals of 2 seconds to the server that hosts 4 vendors and only one broker. We can see that to evaluate 80 proposals the server takes about 24 seconds in this case. The second series represents the same scenario, but with an mean time of arrivals that is 1 second. In this case it is straightforward to observe that only one broker is able to process all the requests in about 13 seconds. The slope of the line shows that the proposals are processed faster as they arrive with greater frequency and the workload is below the capability of the server. The same happens in about 5 seconds when the mean time of arrivals is 0.5 seconds.

In the third series we doubled the number of clients and the number of brokers. We have 160 CFPs, which arrive with a doubled rate, but the throughput of the server does not change as we doubled the number of broker threads. Just in the end we can observe a slightly shift of the line because the clients stopped and the queue are going to be empty. In the fourth series we doubled the number of brokers and get a faster throughput at the beginning, if compared with the previous case, but after that the slope of the line follows again the trend of the previous case. Finally we show the case of 8 clients with 2 and 4 brokers. We observe that the slope of the line depends on the number of brokers again and it is constant till when the queue has proposals ready to be dispatched or the overhead makes the machine slow. We can conclude that in all the case the workload does not exceed the server capability the application scales well.

We observed a loss of performance either when the number brokers is greater than 4, as they exceed the number of cores of our server, and when the number of clients double, as the workload overcome the capability of our machine. As we can see in Fig. 5, we have the maximum throughput in the case of 8 clients and 4 brokers.

The effect of the overhead, due both to the need of handling the high rate of incoming CFPs and to the schedule of a number of broker greater than the number of cores can be observed in Fig. 6. The x-axis shows the different proposals in the order of arrival. The

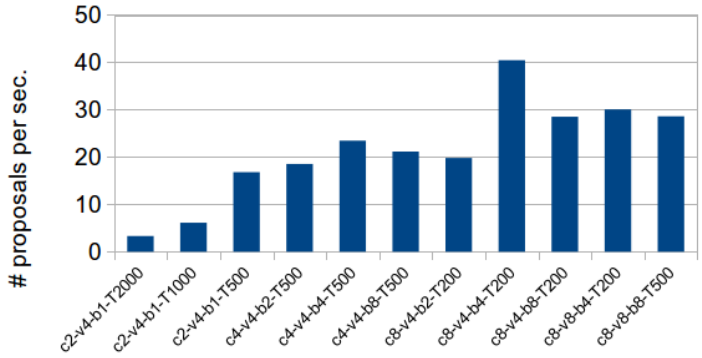

Fig. 5. Broker throughput.

blue series shows the elapsed time between the arrival of a proposals and the previous one. The red series shows the time between a proposals has been received by the broker and the next arrival into the SLA_QUEUE. We can see in Fig. 6(a), (c), (e) that when the rate of incoming proposals is low there is no overhead and the broker takes a constant time for processing the proposal. In particular when we have 2 clients we never overload the system. In Fig. 6(a), (d), (f) the workload increase both because the high number of concurrent clients and because of the scheduling overhead. For this reasons we have a random delay for both the traced events starting from when the server reaches the overload condition.

A noticeable effect of this behavior is also the average waiting time of a proposal into the queue, that affects directly the response time of the system and service level perceived by the client.

In Fig. 7 we can see the average enqueued time of a proposals in the case of the server. It is much more then the mean time needed by a broker to process the proposal that is about 55 ms.

In order to improve this parameter we investigated the possibility to offload part of the workload by a Cloud Infrastructure. We used an OpenStack installation in the same local network. This private cloud provided a Linux virtual machines. The Linux OS sees just one processor with a 64 bit virtual Intel Core 2 Duo P9xxx (Penryn Class Core 2) 2.5 GHz with 2 K L1 cache and 2 GB RAM. In order to estimate the processing capability of such computing resource we tested the scenario defined above with no brokers running on the server and some brokers executing only on the virtual machine. We estimated a processing time for a proposal evaluation equals to 100 ms when only one broker is running. In Fig. 8 we can see that the average enqueued time for a proposals improves when two brokers are used and goes worse when 4 threads run concurrently. We tested this Cloud support to evaluate the improvement of the best working configuration presented before, that has 8 clients, 4 vendors and 4 brokers on the server. In Fig. 9 we can see that we are able to improve the performances when 2 broker threads run on the VM. About the distribution of workload between machines, it has been delegated to the ActiveMQ service that distributes the messages to the consumers according to their capability to consume. It has not been studied further.

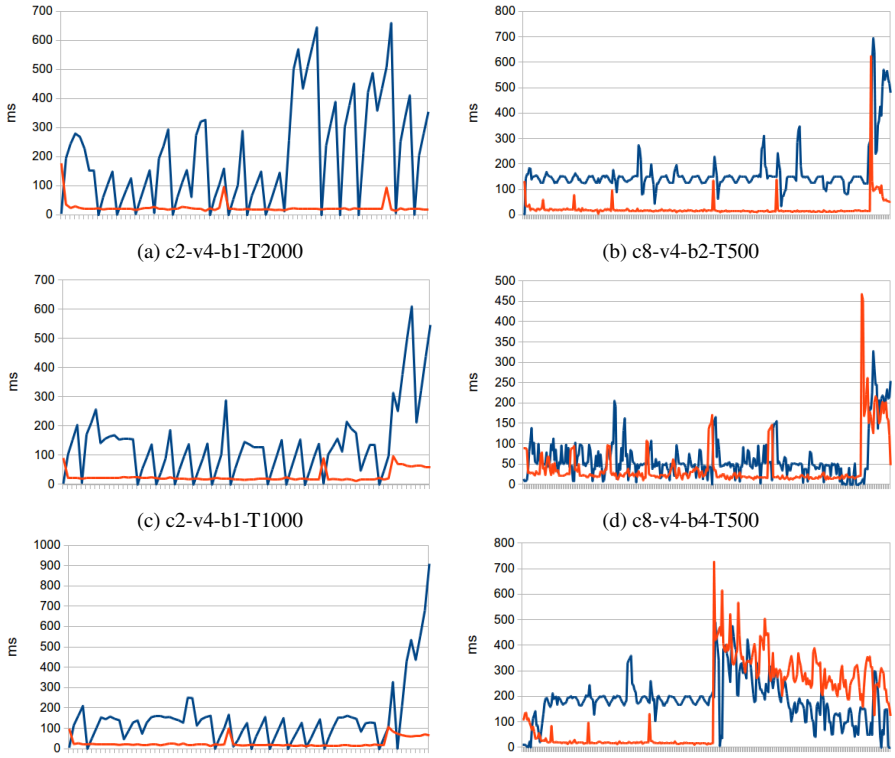

Fig. 6. Mean time of arrivals vs service time of proposals.

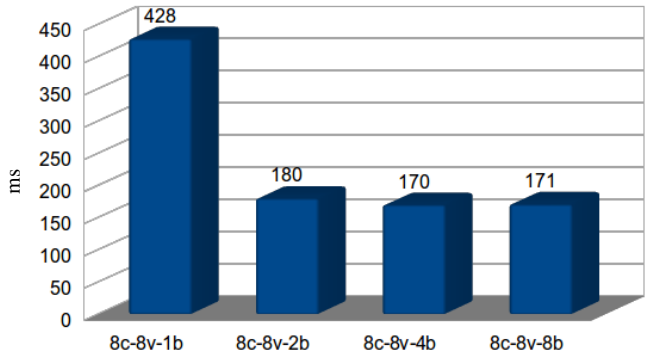

Fig. 7. Average enqueued time on VM.

6. Conclusion

We discussed about the decision problems occurring during the choice of the cloud provider, which are non-trivial due to the increasing number of Cloud providers and to the

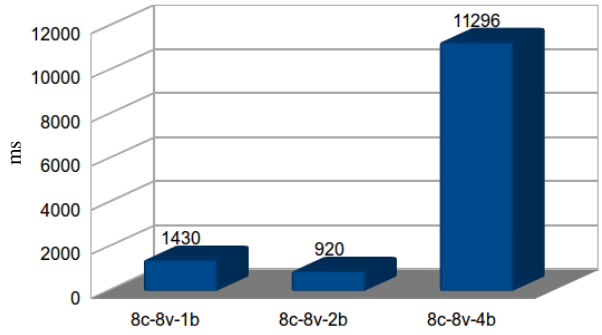

Fig. 8. Average enqueued time on VM.

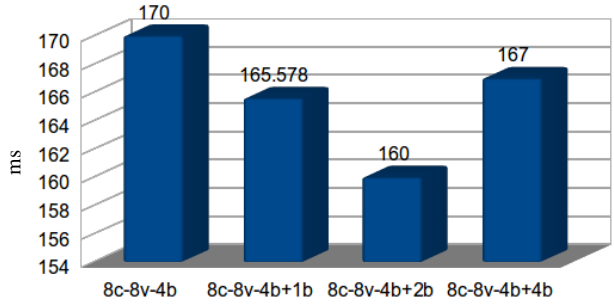

Fig. 9. Average enqueued time on hybrid computing infrastructure.

lack of common formal models for describing service properties and service levels. Moreover the heterogeneity of services provided by current public Cloud technology providers contributes to increase the complexity of comparison between different solutions. In this paper, we introduced the broker module, whose aim is to acquire automatically resources from providers on the basis of SLA evaluation rules, finding the most suitable Cloud provider that satisfy users' requirements. We presented architecture and implementation details of a scalable broker as a service solution. We presented a prototype implementation and provided preliminary performance figures. We discussed scalability of such prototype, over Cloud infrastructure too, showing positive results for further improvements. Future works will focus on improvement of absolute performance by the optimization of the matching between proposal and CFP, by the indexation and caching of document and results. Moreover autonomic strategy for the dynamic reconfiguration of the computing infrastructure and the workload distribution will be investigating. Finally we are investigating the possibility to use heuristic techniques for brokering composite services, using a set of proposals from different providers, according a defined work-flow.

Acknowledgement. This work has been supported by PRIST 2009, Fruizione assistita e context aware di siti archelogici complessi mediante terminali mobili, founded by Second University of Naples; by the mOSAIC project (EU FP7-ICT) and by CoSSMic (EU FP7-ICT-608806).

References

Amato, A., Venticinque, S. (2013). Multi-objective decision support for brokering of cloud SLA. In: *The 27th IEEE International Conference on Advanced Information Networking and Applications, AINA-2013, Barcelona, Spain, March 25–28, 2013*. IEEE Computer Society. ISBN 978-0-7695-4952-1/13.

Amato, A., Liccardo, L., Rak, M., Venticinque, S. (2012a). Sla negotiation and brokering for sky computing. In: *The 2nd International Conference on Cloud Computing and Services Science, CLOSER*, pp. 611–620.

Amato, A., Di Martino, B., Venticinque, S. (2012b) Evaluation and brokering of service level agreements for negotiation of cloud infrastructures. In: *7th International Conference for Internet Technology and Secured Transactions, ICITST*, pp. 144–149.

Amato, A., Di Martino, B., Venticinque, S. (2013). Cloud brokering as a service. In: *The 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC, Compiegne, France, December 28–30, 2013*. IEEE Computer Society, pp. 9–16.

Buyya, R., Ranjan, R., Calheiros, R.N. (2010). Intercloud: utility-oriented federation of cloud computing environments for scaling of application services. In: *The 10th International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP*, pp. 13–31.

Di Martino, B., Petcu, D., Cossu, R., Goncalves, P., Máhr, T., Loichate, M. (2011). Building a mosaic of clouds. In: *Proceedings of the 2010 Conference on Parallel Processing, Euro-Par 2010, Berlin, Heidelberg, 2011*. Springer-Verlag. ISBN 978-3-642-21877-4.

Foundation Intelligent Physical Agents Technical Report (2002). *Fipa contract net interaction protocol*. http://www.fipa.org. Online: accessed 31 January 2014.

Jamcracker (2012). http://www.jamcracker.com/. Online: accessed 31 January 2014.

Nair, S.K., Porwal, S., Dimitrakos, T., Juan Ferrer, A., Tordsson, J., Sharif, T., Sheridan, C., Rajarajan, M., Khan, A. U. (2010). Towards secure cloud bursting, brokerage and aggregation. In: *Proceedings of the 2010 Eighth IEEE European Conference on Web Services, ECOWS '10, Washington, DC, USA, 2010*. IEEE Computer Society, pp. 189–196. ISBN 978-0-7695-4310-9.

NIST (2012). Nist cloud computing reference architecture. http://www.nist.gov/itl/cloud/upload/SP_500_293_volumeI-2.pdf. Online: accessed 31 January 2014.

Optimis-project (2011). http://www.optimis-project.eu/. Online: accessed 31 January 2014.

Sla@soi (2012). http://sla-at-soi.eu/. Online: accessed 31 January 2014.

Tordsson, J., Montero, R.S., Moreno-Vozmediano, R., Llorente, I.M. (2012. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Generation Computer Systems*, 28(2), 358–367. ISSN 0167-739X.

Venticinque, S. (2013). User-centric infrastructure as a service by cloud agency. *Multiagent and Grid Systems*, 9, 157–159. ISSN 1574-1702. doi:10.3233/MGS-130204.

Vordel (2012). http://www.vordel.com/solutions/cloud-service-broker.html. Online: accessed 31 January 2014.

A. Amato is Research Assistant at the Department of Industrial and Information Engineering, Second University of Naples, Aversa, Italy. She received her PhD in Electronic Engineering in 2013. Her research interests focus on intelligent systems, agent architectures, multiagent systems and their applications in cloud computing.

B. Di Martino is Full Professor of Information Systems at the Second University of Naples (Italy) since 2005. He participated to various research projects supported by national and international organizations. He is vice Chair of the Executive Board of the IEEE CS Technical Committee on Scalable Computing. His research interests include: Knowledge Discovery and Management, Semantic Web and Semantic Web Services, Semantic based Information Retrieval, Cloud Computing, High Performance Computing and Architectures, Mobile and Intelligent Agents and Mobile Computing, Reverse Engineering, Automated Program Analysis and Transformation, Algorithmic Patterns Recognition and Program Comprehension.

S. Venticinque is an Assistant Professor at Department of Information Engineering of the Second University of Naples. He received his PhD in Electronic Engineering in 2003. He coauthored more than 100 scientific papers published in international conferences and journals. He is involved in research activities dealing with parallel and grid computing and mobile agents programming for distributed systems.

Paskirstyta debesų tarpininkavimo paslauga

Alba AMATO, Beniamino DI MARTINO, Salvatore VENTICINQUE

Debesų kompiuterijos pasiūlymų tarpininkavimas, optimizuojantis taikomosios programos reikalavimus, įgalina išnaudoti debesų kompiuterijos paradigmos lankstumą dinamiškai parenkant geriausią rinkoje esantį paslaugos lygmens susitarimą. Šiame straipsnyje pristatome išplečiamą daugelio vartotojų tarpininkavimo kaip paslaugos versiją, kuri naudoja prieinamus paskirstytosios aplinkos išteklius ir sprendžia susijusius klausimus. Tarpininkavimo paslauga yra sudalinta į paprastesnes užduotis, kurios yra paskirstytos nepriklausantiems agentams, o jų populiacija dinamiškai išplečiama kartu su debesų infrastruktūra siekiant palaikyti iš anksto neįvertinamą darbo krūvį, sukeliamą didelės vartotojų grupės. Tarpininkavimo modelis ir jo realizavimas, pritaikantis debesų technologijas, yra aprašytas. Pirmojo prototipinio realizavimo rezultatai ir veiksmingumas yra aptarti.