QAComPS: A Quality-aware Federated Computational Semantic Web Service for Computational Modellers

Peter M. Dew (1) and Shahzad Nizamani (1, 2)
[1] School of Computing University of Leeds LS2 9JT, UK
[2] Mehran University of Engineering & Technology, Pakistan
p.m.dew@leeds.ac.uk, scsan@leeds.ac.uk

Abstract—*This research sets out to help computational modellers to select the most cost effective Cloud service provider. This is when they opt to use cloud computing in preference to in-house HPC facilities. Cloud computing is a pay-per-use model for accessing computing resources from a variety of service providers such as Amazon EC2. Increasingly cloud providers are offering the high performance computing options that are necessary for computational modellers.*

The paper is concerned with a quality-aware computational broker (QABroker). The QABroker service federates across specific service packages offered by a selected set of computational cloud providers that potentially meet the user's computational resource and QoS requirements. These vary during the various stages of the computational modelling cycle. The core of the QABroker is a novel Quality-aware federated computational semantic Web service (QAComPS). This includes an integrated ontology-based system that makes use of OWL2 features. This is used to filter the cloud providers' services into three groups. These are: High, Medium and Low quality of service. This classification is then used by a MatchMaker to automatically select the highest ranked service that meets the user requirements.

A SAWSDL interface was used to transfer semantic annotations to/from the QAComPS service and QABroker. Early evaluation of the QAComPS service was very promising and demonstrates its potential to make cloud computing more accessible and cost effective for computational modellers.

Keywords
Cloud computing, quality aware service, semantic Web, SAWSDL, computational modelling and broker mediation

I. Introduction

The motivation for this research was to help computational modellers to have cost-effective access to computational cloud services. Computational modellers address complex, real-world problems through building computerized models of physical phenomena. Their modelling often requires access to HPC facilities. Typically, their computational and quality of service needs vary during the modelling life cycle. This research therefore focuses on developing a quality-aware computational broker (QABroker) that automates for the user the process of selecting and running a cloud provider service that meets the user's computational and Quality of Service (QoS) requirements throughout the lifetime of the model and development.

Cloud computing [1] provides predictable and flexible on-demand pay-per-use access to a shared pool of computing resources (e.g. networks, servers, storage facilities, applications and services). In this research we were concerned solely with IaaS (Infrastructure as a Service) cloud service providers. Each cloud computing service provider offers the user a choice of different VMs (Virtual Machines). A VM emulates a physical machine.It is performed by hardware virtualization where a physical machine is used for creating VMs. Each VM has processor, memory, storage and other resources. Price of a VM depends on the allocated resources (e.g. the amount of run-time memory and the number of CPU cycles).

The user therefore has to choose between cloud computing service providers and also between the VMs they offer. This research considers the cloud providers: Amazon EC2 [2], Rackspace [3] and FlexiScale [4]. Amazon EC2 now offers a clustered HPC option so it is reasonable to assume other cloud providers will also offer comparable HPC services. An early comparative study of HPC cloud providers is given in [5] and [6] reports their experience of using EC2 for HPC.

This paper is concerned with a computational cloud broker service called the QABroker that mediates across a selected set of computational cloud providers. The broker service includes a cost structure that incorporates QoS metrics such as reliability, user satisfaction (or reputation), cost and security. The role of the QABroker is to automate the selection process of cloud providers and their associated VMs. The selection process takes account of the user's computational resource and QoS requirements at that point in time. The user receives the selected VMs without needing to know about the provider.

A key step in building the QABroker was to design and evaluate the *Quality-aware federated computational semantic Web service (QAComPS)*. The main features of QaComPS are:

1. A federated cloud provider's ontology service to integrate the information on the QoS and cloud provider's resources with associated costs;
2. An automatic (agent) selection process to discover the best VM that meets the computational modellers QoS and resources requirements;

3. A semantic annotation for web service description language (SAWSDL) interface between the Broker information and QAComPS service, for example to update the QoS metrics.

A primary evaluation study was performed to demonstrate the feasibly and value of the QAComPS service. The paper is structured in the following way: §2 provide the details on the proposed QABroker service architecture, in which the novel QAComPS service resides. The following section provides details of the QAComPS semantic Web service including the architecture and ontology. This is followed in §4 with the evaluation study and key details of the implementation. §5 covers the relevant background literature and the last section provides the conclusions and recommendations for future work.

II. QA Cloud Broker

a. QA-Aware Cloud Services

A number of papers around 2003/2004 discuss QoS issues in Web Services (e.g. see [7-9] and more recently [10]). Service ontology provides a consistent semantic data model for describing QoS metrics that are non-functional properties. The two types of QoS are "best–efforts" and "guaranteed service". Here we use "best-efforts" that is referred to as *Quality-awareness* where the service provider can: (1) just drop the service in the case of overload; and (2) provide no guarantees concerning the response time, job throughput etc.). Today most public cloud providers only offer quality aware services. This paper also provides a quality aware service.

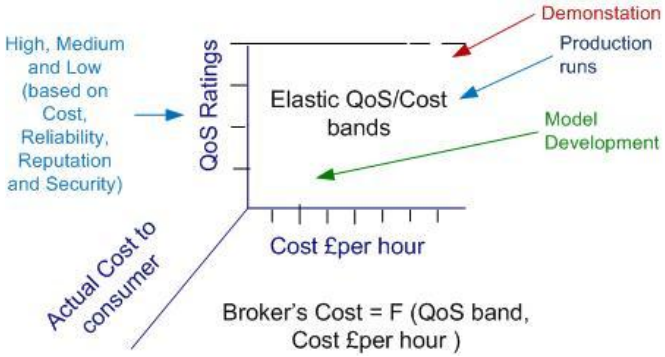

Figure 1 Cost Model

Figure 1 shows the use of QoS and the cloud cost. Four non-functional QoS metrics were considered:

Cost $C(p)$, Reputation $R_p(p)$, Reliability $R_l(p)$ and Security $S(p)$

where p is a provider and each rating is on a rating scale of 1 – 9 with 1equating to lowest quality and 9 to highest quality. Selection of these illustrative and other QoS metris can be added/deleted as required in the future. The cost QoS is computed from the cost model described below. The reputation and reliability ratings of the VM used can be updated with quantitative data after each computational modelling run. Security ratings are updated qualitatively. A MatchMaker matches the user's QoS requirements against all the VMs that the different providers offer and subsequently ranks the VMs from best to worst on the closeness of the match. This technique is widely used for selecting Web Services. In our case the goal is to automatically select the VM that meets the user's computational resource and QoS requirement at the lowest cost. For fuller details on the QoS metrics and the MatchMaker the reader is referred to paper [9, 10]. As it can be difficult for the user to specify the exact QoS they require they are asked to indicate whether they desire a high, medium or low QoS. They are also required to state the relative importance of the each QoS metric for a job. The QABroker can be adapted to incorporate additional levels. Three levels were used in our evaluation study.

b. Case Study

At the University of Leeds, computational modellers have been experiencing frustrations in two areas. Firstly, they feel disadvantaged by the way local HPC facilities schedule jobs with a run-time of twelve hours or more (a feature of much of their work). The turnaround time of these jobs can be unpredictable depending on the size of the HPC job queue and long job runs may be limited to weekends. Secondly, they find the price and reliability of the service. These are inflexible and do not cater very well for their computational service needs as they vary throughout the model development process. For example in the early stages of the modelling, when job runs tend to be short and each one has relatively little importance depending on the outcome, the developer may be happy to accept some reduction in service reliability in order to have lower run time costs. However, the longer the run time for a job, the more important it is that the run is successfully executed at the first attempt. In addition, at certain times, for example when demonstrating to project sponsors, the reliability of the service is crucial, and for these job runs, the modeller is likely to prefer to pay a premium rate for very high reliability.

Figure 1 illustrates the way cost and quality of service are related and indicates the likely preferences of the computational modeller during three phases (model development, production runs and demonstration). The QABroker would discover the user's computational and QoS preferences at each stage of the life cycle of the model. This information would enable the broker to make a set of VM selections. Each choice would meet the modeller's computational and QoS requirements at the lowest cost.

QABroker Service Architecture

Figure 2 shows our envisioned QABroker, the environment in which the QAComPS semantic Web service is used. The QABroker cloud service is managed by an external Broker. To illustrate the services figure 2 shows two Web services:

- Broker RUN Web Service provides the infrastructure to enable the computational modeller to run their job on the selected cloud;
- Broker information Web service manages the information for the QAComPS service.

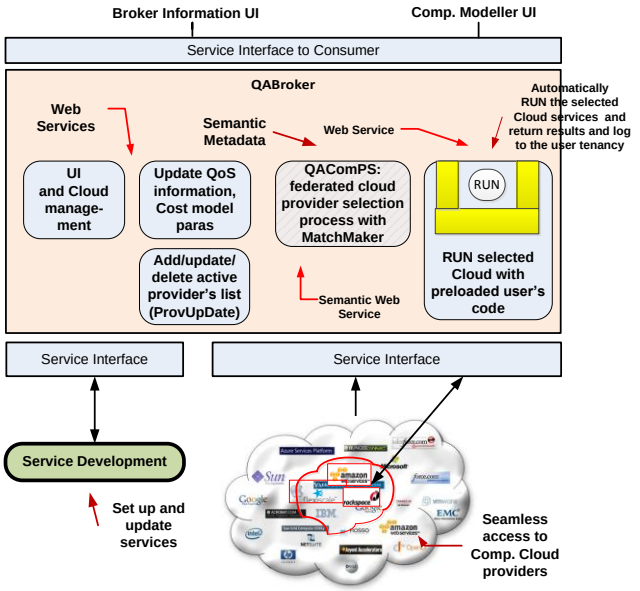

Figure 2 QA Cloud Services

The QAComPS semantic Web service is the main service discussed in this paper. Its purpose is to select the VM that best meets the user's QoS and computational requirements. This required enhancing the capability of the MatchMaker to automatically make VM selections. This would more precisely meet the user's requirements than had previously been possible. This QAComPS service is fully described below. It makes use of external Web services as shown in the diagram.

These Web services are used for two purposes: (1) To access the world of cloud providers, from which the Broker selects, a set of suitable computational cloud providers; (2) Where possible the Broker transparently accesses the provider's information from their WSDL document or API. The Broker service monitors each provider for any changes to the information it holds about each provider and updates its records at each Broker service break.

In addition to the QAComPS the QABroker internal services include: (1) the UI and management (not developed further in this paper); (2) a QABroker information service to dynamically manage updates about the QoS of each VM. The information in these updates is periodically passed across to the QAComPS service; (3) a RUN Service that provides the framework to run the selected cloud provider's VM; and (4) The ProvUpDate service maintains a list of available providers. It activates and deactivates providers depending on their availability and performance.

The RUN service has a SAWSDL interface that receives Resource Description Framework (RDF) information from the VM selected by the QAComPS. The RDF information is used to access the selected cloud service and run the user's computational model. The job log is by QABroker and is also uploaded to the user's tenancy.

III. QAComPS Service

This is a novel service that mediates across providers' various VMs. The service contains a logical model that integrates three areas of information: general information on the VM and cloud providers; details of the computational resources the VMs offer and information about their QoS. The model contains four QoS parameters: cost; reliability; reputation and security (each on a scale from one to nine). The precise meaning of a QoS rating is a business decision.

The cost element of the QoS model is derived from a separate Cost model. This maps cost against the computational resources purchased for example RAM (GB), virtual core (integer), Disk space (GB), price per hour, communication and storage services. The cost model can be expressed as:

$$Cost = (\alpha \times C_l + \beta \times Dt) / (\alpha + \beta)$$

where C_l is made up of memory (GB), processor (virtual cores), and non-persistent storage. D_t is the data transfer rate and α, β are constants chosen by the broker. The computational resource size is in four bands: Small, Medium, Large and Very Large. Table 1 shows the actual cost values used in our evaluation study. Here the model does not include long term storage costs that may also be important.

Table 1: *Broker Cost Model RUN Service*

	Cost/ hour	Memory GB	#Cores	Storage GB
Small	0.06	2.00	1	160
Medium	0.13	4.00	4	500
Large	0.22	8.00	8	800
Very Large	0.45	16.00	16	1700

The costs are based on an exchange rate of 0.6 £/$ and it assumes a Linux operating system.

Three cloud providers were considered: Amazon, Rackspace and FlexiScale as a representative sample of computational cloud services. These providers were used to specify a logical cost model and associated resources (see Table 1). This was used by the MatchMaker that forms part of the selection process. This is discussed in the next section.

The cost model was also used to define the cost QoS rating. In our evaluation study we used three cost bands: Low (1 – 4) equates to high cost VMs (the most expensive for the computational resources they provide); medium (5 – 7) represents intermediate value for money and high (8 – 9) represents excellent value for money (the lowest cost). Cost bands are the main linkage between the actual costs (what the user pays) and the Cost QoS ratings.

Amazon is a cloud provider that explicitly offers HPC VMs. These are clustered and support applications using MPI (the message passing interface) and running very computationally intense jobs. As these services are ideal for running HPC applications the broker RUN service can be restricted to considering only computational clouds that offer this type of service.

a. Architecture

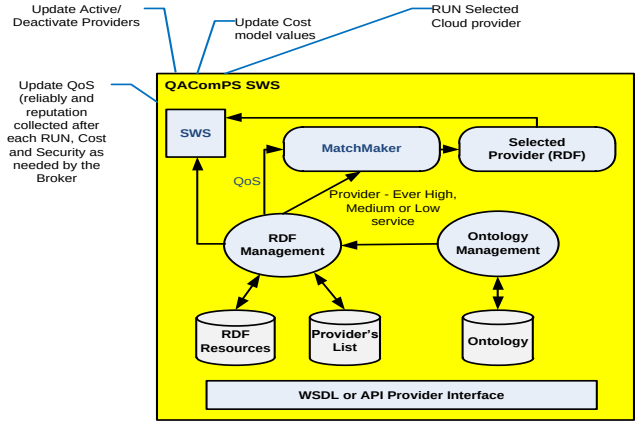

Figure 3 QAComPS Architecture

The QAComPS architecture has three main functions: (1) The management of the selected providers; (2) The matchmaker; and (3) The selection of the best cloud provider. The service ontology federates across the active cloud providers and logically filters the providers' VMs into High, Medium or Low quality services. This filtration improves the performance of the selection process.

The first stage of the selection process is to apply the widely used Euclidean distance algorithm [4] to rank the active provider's VMs based on the four QoS ratings. The second phase is to take the top *five* ranked VMs and use the *Analytic Hierarchy Process (AHP*) [11]. It is a widely used MCDA (Multi Criteria Decision Analysis) based method that uses a hierarchical approach to decision making. The AHP process matches the VMs against the user requirements using the QoS levels (High, Medium and Low) together with weights. A weight represents the importance of each QoS parameter for the particular job. For example at certain times the cost parameter may be much more important to the user than the parameter 'provider reliability'. QoS levels and weights express the relative priority of each QoS parameter. The performance evaluation results are given below along with the justification for our approach (see §4).

b. Provider's Ontology

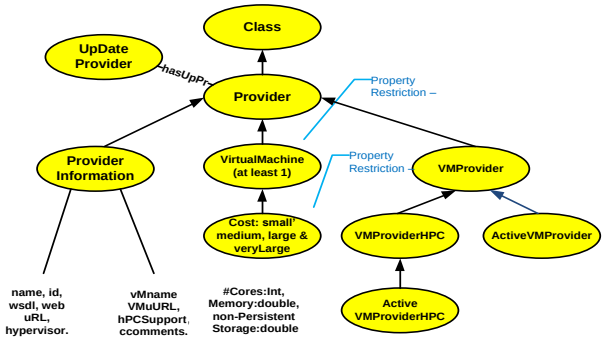

Figure 4 Provider's ontology

The Provider's ontology is shown in Figure 4. The top class is the Provider and a sub-class models the number of Virtual Machines (VM) offered by the provider. An OWL2 property restriction is used to ensure that there is at least one VM. As discussed above the cost model provides four levels varied by VM size. An information class is provided that handles information associated with a provider. The VMProvider class models the type of VM and whether it's active or inactive. A UpDateProvider class enables the information on the selected Provider's VM to be transferred to RUN service using SAWSDL annotations.

c. QoS ontology

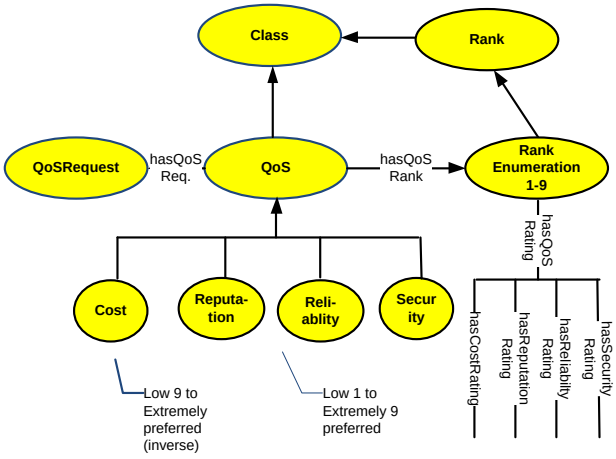

Figure 5 QoS Ontology

The QoS ontology is based on the QoS model as discussed above. It uses an OWL2 Rank Enumeration giving the meaning to the ranking. The data properties enable the user to access each QoS data property. A QoSRequest class enables communication of information between the QAComPS and the broker information service using SAWSDL annotations.

d. Filter Ontology

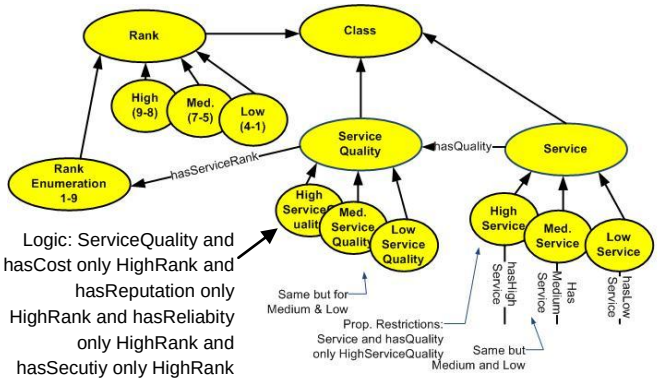

Figure 6 Filter Ontology

The purpose of the filter ontology is to formally classify the providers' VMs into three groups. These are Classes named High, Medium and Low Service. They are used by the MatchMaker. Our evaluation results in §4 show the effectiveness of the filter and this simple classification for

the selection of the best VM. The ontology is shown in figure 6. The two main Classes are ServiceQuality and Service. There is an object property linking them. The QoS ratings (1-9) are again used for each QoS metric. The meaning given to the three service quality levels are: High (8-9), Medium (7-5) and Low (4-1). The meaning of each level is assigned by using the OWL2 union property restriction. The logical expression to do this is:

$$UQoS = C \ \& \ R_p \ \& \ R_l \ \& \ S$$

where C, R_p , R_l & S are given in §2.1. The meaning assigned to each service quality class is as follows: HighServiceQuality is UQoS ≥ 8; MediumServiceQuality is UQoS ≥ 5 and < 8; and LowServiceQuality = UQoS < 5. The semantic Web statements are shown in figure 6.

Logic for filtering providers for high QoS is given below:

Logic: ServiceQuality and hasCost only HighRank and hasReputation only HighRank and hasReliabity only HighRank and hasSecutiy only HighRank.

The logic for medium and low can be given in the same manner.

IV. Evaluation

The evaluation identified whether the QAComPS selected the "best" provider while fulfilling all the QoS constraints. It also pointed to any performance lags and other issues with the proposed solution. The proposed solution evaluation was mainly for the selection process against AHP and Quality Matchmaking Process (QMP) [9] that used a combination of Euclidean Distance and AHP algorithms. QAComPS was implemented as a service. In order to undertake this experiment AHP and QMP services were needed to be created. These services were created and tested before undertaking the evaluation. This section is subdivided into a description of the experiment and its results.

a. Experiment

The experiment consisted of the development of two services: QAComPS (responsible for processing user requests) and QABroker (used for updating the provider information).

QABroker provided a light weight service used for creating and updating provider information. Inputs to the service were provider and VM information shown as data properties in Figure 4. This information was processed by creating a model of the ontology with the inputted data values. This was than inferred and reasoned for any errors. If there were no errors than the information was added to a newly created RDF record and an entry was made in the index file. For this experiment public providers such as Amazon were considered but were not used as it would have been too expensive. The results from this research will provide assurance before investing further with this research. Instead twenty five simulated providers were created each offered a different set of QoS metrics while all of them offered the same computational resources. The simulated providers offered small, medium, large VMs while public providers also offered extra large VMs that were not part of this experiment. The resource information associated with small, medium or large came from public cloud providers and is shown in Table 1. QAComPS as described in section 2.3 that used SAWSDL to communicate with other services. SAWSDL annotations included a model reference, and lifting and lowering schema mapping data. The model references represented entities that form part of the ontology while the lifting and lowering schema mapping formed the communication channel between QAComPS and other web services. The lifting schema mapping was used for transferring data from a non-semantic source, such as XML to QAComPS. For the lowering schema mapping QAComPS used a SPARQL query to extract information from RDF and pass it to a non-semantic web service.

QAComPS MatchMaker consisted of a ranking and selection step. It started off by receiving a user request that consisted of resource and QoS requirements. The resource requirements included memory, storage and CPU requirements. The QoS requirements were low, medium and high. The request was passed on to the Euclidean Distance based ranker that ranked the list of available providers. The top five providers were passed on to the AHP-based matchmaker that selected the "best" provider. At the top of the AHP hierarchy the goal was setup to identify the best provider. This was followed by the QoS criteria parameters and their associated weights. These were inputted by the user to reflect their relative priorities. At the bottom of the hierarchy there were alternatives that represented available VM options.

b. Results

The experiment was performed by creating twenty five simulated providers and twenty four user requests. There were eight user requests each for low, medium and high QoS. They were controlled requests whose output was previously calculated beforehand to identify the progress of each service. Each user request was passed to AHP, QMP, QAComPS and QAComPS (with filter).

Figures 5, 6 and 7 showed the results. The horizontal axis shows the user request whilst the vertical axis shows the logical cost. The logical cost model is given in (section III). The logical cost is measured on a scale of one to nine (with nine showing the best option).

The selection was dependent on the user requests and provider QoS. While providers with higher QoS service and lower costs were selected more than once there were other providers which were not selected at all as they offered higher costs and lower QoS. The results shown in figures 5, 6 and 7 show the average cost of the selected providers. At the start of the experiment user requests for high QoS were made and then for medium and lower QoS.

Figure 5 shows the results for high QoS. This means that each of the QoS rating (Cost, Reliability, Reputation and Security), for the selected providers, were high. It can be

observed that AHP, QAComPS with and without the filter produced good results. However QAComPS had some inconsistencies and QMP was not very effective.

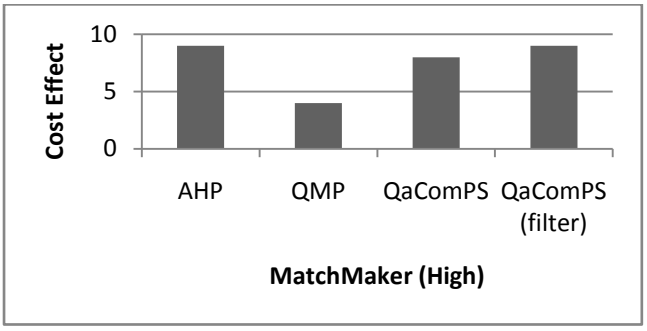

Figure 5 MatchMaker Comparisons (High QoS)

Figure 6 shows the results for medium QoS where the selected providers had a medium level of QoS for all four ratings. It can be observed that QAComPS and QAComPS (with filter) were very effective.

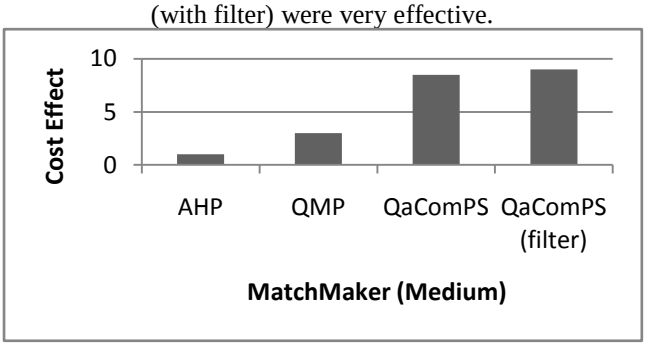

Figure 6 MatchMaker Comparisons (Medium QoS)

Figure 7 shows the results for low QoS, for the selected providers, all have low QoS ratings. It can be observed that QMP was very effective with low QoS while QAComPS (with filter) was consistent for all the user requests.

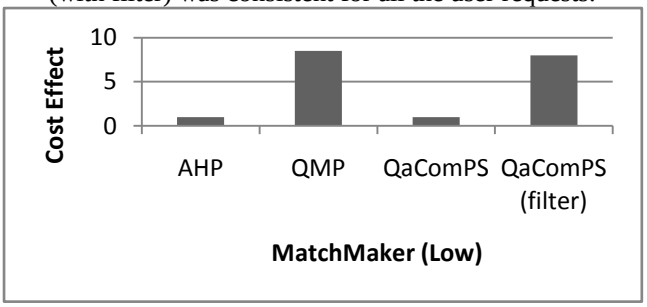

Figure 7 MatchMaker Comparisons (Low QoS)Overall AHP performed well for user requests for higher QoS and did not perform well for medium and low QoS. This may be due to the way AHP selects the best provider as it prioritize high. It behaved in a different way to a fixed set of simulated providers and prioritized selecting the same provider more than once.

Overall the AHP performed well for higher level of QoS while it did not perform well for others. AHP was effective for major changes to user requests however and it was less effective to smaller changes. This may be in part due to the way the algorithm worked as we were using the same data set for every request while it is still being investigated. QAComPS without the filter works well for high and medium with some discrepancies but was not very effective for low while the introduction of filter resolves produces effective results for any QoS. There were no performance issues as the processing time for the twenty five user requests was always under two seconds.

Overall QMP performed well for low but not so well for medium and high. This is due to the way QMP operates as it may prioritize for providers offering lower QoS
Overall QaCompS performed well for high and medium QoS while for low it was not good. This was due to the way QaComPS operates as it always prioritises requests for higher QoS and prefers selecting providers with higher QoS.

Overall the results showed that the QAComPS (with filter) performs best for all the three levels of QoS. Another advantage of the filter was that the performance was enhanced as it reduces the set of relevant providers which results in less processing.

V. Related Research

For the background Semantic Web Ontology the reader is referred to [12]. There are two approaches to creating semantic web services. One is a top down approach using OWL-S the other is a bottom up solution using either SAWSDL [13] or WSMO lite [14]. SAWSDL annotations were used in this paper. This avoids the complexity of using OWL-S [15] while retaining the benefits of a semantic user interface.

There are a number of papers on semantic matchmaking arising from e-commence. The most notable is [16] that presents a prototype matchmaking service using description logic and DAML-S (this was a forerunner to OWL-S). However for performance concerns a non-semantic matchmaker was used in this paper.

There are also papers on QoS Web services that use ontologies. In [17] they provide a novel, rich and extensible ontology for the selection of the requested QoS. Also see [18, 19]. However these ontologies were too comprehensive for this research.

The Grid community [20] has been actively involved in QoS Web services federated by a broker. This brokerage involves managing the negotiation between the service provider and the service consumer and recording any service level agreement (SLA) that is reached. Such brokers often use the standard on-line WS-Agreement SLA document [21] for enforcement (including QoS guarantees). For example, see the ASSESS project that considers risks to fulfilling the SLA (e.g. because of reliability failures) [22]. Buyya and Ranjan discuss a federated resource manager for both the grid and cloud providers [23]. Rochwerger et.al. [24] argue that cloud providers have only recently begun to address the requirements of enterprise solutions, such as support for infrastructure service-level agreements. Their Reservoir project aims to enable providers of cloud infrastructure to

dynamically partner with each other to create a seemingly infinite pool of IT resources. A new EU cloud project called OPTIMIS [25] is also aimed at enterprise cloud computing. Its goal is to enable organizations to automatically externalize services to trustworthy and auditable cloud providers in a hybrid cloud model. WS-Agreement are being used in this project.

VI. Conclusion

This paper has presented a novel quality-aware federated computational semantic Web service (QAComPS). This service enabled automatic selection of cloud providers. QaComPS is the key service for the envisioned cloud broker. The evaluation results have potentially shown that QaComPS service can successfully select the best computational cloud provider that meets the user's resource and QoS requirements. The paper has also shown the benefits of using semantic annotations to communicate with external services.

For the future this research needs to ground the simulated results to actual computational cloud providers. The next stage is to fully integrate QaComPS service into to QaBroker service.

QaComPS selects the best cloud provider and passes the RDF file of the selected provider onto the cloud run service. This service invokes the required VM and deploys a VM image on the selected VM. Public providers such as Amazon offer a public DNS key which is used to access VMs remotely. The run service shares this key with the user which grants the user access to the VM. The user can now run his code on the VM and can also install new software. Once the user has completed the job he notifies QaBroker which invokes the cloud run service to stop VM.

This research also has the potential for helping brokers provide a service within a specified timeframe. This service assumes that a user or broker has inserted a number of checkpoints. This would enable the performance of the selected VM to be monitored during a job run. These jobs typically take many hours it would be possible to switch to another VM if the job run fell behind schedule. This would help the broker fulfill service level agreements irrespective of difficulties encountered with a particular VM. This moves us closer to providing a guaranteed service. This assumes that the modeling process is particularly regular to predict the end performance.

VII. Acknowledgments

The authors are extremely grateful to Dr. Benadon Bennett for writing the Filter ontology. Thanks to Professor Peter Jimmack for posing the computational modelling scenario. Thanks also to Dr K Djemame his support for this research.

VIII. References

[1] Armbrust, M., Fox, A., Griffith, R., and Joseph, A.D.: 'A view of cloud computing', Communication of the ACM, 2010, 2010, 53, (4), pp. 50-58.

[2] Amazon Web Services. http://aws.amazon.com/ec2 (09/03/2011).

[3] Rack Space Hosting. www.rackspace.co.uk/cloud-hostin (09/03/2011).

[4] FlexiScale Public Cloud. http://flexiscale.com (09/03/2011).

[5] He, Q., Zhou, S., Kobler, B., Duffy, D., and McGlynn, T.: 'Case Study for Running HPC Applications in Public Clouds'2009.

[6] Evangelinos, C., and Hill, C.N.: 'Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2'. Proc. CCA-08, Chicago 2008.

[7] Menasce, D.: 'Composing Web Services: A QoS Veiw', IEEE Internet Computing, 2004

[8] Menasce, D.: 'QoS Issues in Web Services', IEEE Internet Computing, 2002.

[9] Eleyan, A., Mikhailov, L., and ., L.Z.: 'Quality-of-Services in Web Services Architecture', Ingnierie des Systmes d'Information, Special Issue on Information Systems Quality, 2004, 9, (5-6), pp. 185-203.

[10] Tran, V., Tsuji, H., and Masuda, R.: 'A new QoS ontology and its QoS-based ranking algorithm for Web services'. Proc. Simulation Modelling Practice and Theory, 2009.

[11] Saaty, T.L.: 'How to make a decision: the analytic hierarchy process', European Journal of Operational Resea rch, 1990, 48, (1), pp. 9-26.

[12] Allemang, D., and Hendler, J.: 'Semantic Web for the Working Ontologist' (Morgan Kaufmann, 2008.

[13] Kopecky, J., Vitvar, T., Bournez, C., and Farrell, J.: 'SAWSDL: Semantic annotations for WSDL and XML schema', IEEE Internet Computing, 2007, 11, (6),

[14] Brooke, J., Fellows, D., Garwood, K., and Goble, C.: 'Semantic matching of Grid resource descriptions', in Editor 'Book Semantic matching of Grid resource descriptions' (Springer-Verlag, 2004,), pp.60-67.

[15] Vitvar, T., Kopecký, J., Viskov, J., and Fensel, D.: 'Wsmo-lite annotations for web services'. Proc. Proceeding ESWC'08 Proceedings of the 5th European semantic web conference, 2008

[16] Martin, D., Burstein, M., Mcdermott, D., and Mcilraith, S.: 'Bringing semantics to web services with owl-s'. Proc. World Wide Web 2007.

[17] Li, I., and I Horrocks: 'A software framework for matchmaking based on semantic web technology - International Journal of Electronic Commerce, Proc. WWW2003, May 20-24 2003.

[18] Wang, F., Vitvar, T., Kerrigan, M., and Toma, I.: 'A QoS-aware selection model for semantic web services', IEEE Transaction Software Engineer, 2006, 30, (5), pp. 311-327.

[19] Kritikos, K., and Plexousakis, D.: 'Semantic QoS metric matching'. Proc. European Conference on Web Services (ECOWS'06) 2006.

[20] Ian Foster, Carl Kesselman, and Steven Tuecke "The Anatomy of the Grid: Enabling Scalable Virtual Organizations" International Journal of High Performance Computing Applications Fall 2001 15

[21] Andrieux, A., Czajkowski, K., Dan, A., K. Keahey, Ludwig, H., Pruyne, J., Tuecke, S., and XU, M.: ' Web services agreement specification (WS-Agreement)', in Editor 'Book Web services agreement specification (WS-Agreement)' (Gridforum.org), 2006.

[22] K. Djemame,J. Padgett, I. Gourlay, and D. Armstrong. 'Brokering of Risk-Aware Service Level Agreements in Grids'. Concurrency and Computation: Practice and Experience, Vol. 23, No. 7, May 2011

[23] Buyya, R., and Ranjan, R.: 'Special section: Federated resource management in grid and cloud computing systems', Future Generation Computer Systems, Elsevier B.V., 2010.

[24] Rochwerger, B.B., D. Levy, E. et al: 'The Reservoir model and architecture for open federated cloud computing', IBM Journal of Research and Development, 2010, 53, (4), pp 1-11.

[25] A.Juan Ferrer, F.Hernandez, J.Tordsson, E.Elmroth, R.M.Badia, K.Djemame 'OPTIMIS: a Holistic Approach to Cloud Service Provisioning'. In Proceedings of the 1st International Conference on Utility and Cloud Computing (UCC 2010), Chennai, India, Dec 2010.