A Metascheduler Architecture to provide QoS on the Cloud Computing

Maycon L. M. Peixoto Marcos J. Santana Júlio C. Estrella Thiago C. Tavares Bruno T. Kuehne Regina H. C. Santana
Universidade de São Paulo – Instituto de Ciências Matemáticas e de Computação
São Carlos-SP, Brasil
{*maycon, mjs, jcezar, thiagocp, btkuehne, rcs*}*@icmc.usp.br*

Abstract— **In this paper, a Metascheduler architecture is proposed to be used on Cloud Computing environments. The aim of this Metascheduler architecture is to handle the submission of a service and to manage the infrastructure of the Grid resources belonging to the Cloud. Futhermore, this Metascheduler also aims of building the best pool of available resources for the implementation of the requests issued by a user. For this, an economy model based in approach of supply/demand is adopted. Additionally, it is proposed in this paper a novel approach for the communications between pairs of entities of the collaborative environment. This communications is implemented through the hierarchical P2P scheme, a model that extend the hierarchical model. The central idea on this model is the use of the upper level Metascheduler (named Superscheduler) to control partial overload and to impose performance under service demand.**

I. Introduction

Cloud computing is an emerging computing paradigm and its model emphasizes the ability to scale compute resources on demand. The Cloud computing can be split into three categories: (1) In the Software as a Service (SaaS): uses the complete applications hosted in the Cloud. (2) In the Plataform as a Service (PaaS): offers a software execution environment on the Cloud, such as an application server. (3) Infrastructure as a Service (IaaS): addresses the sharing of computational resources and its virtualization technology. By means of this, the resources can easily be adapted when demand increases, allowing many users to use existing resources [1] [2]. Normally, datacenters are used to perform the services in a Cloud. However, may be interesting to take advantage of available resources of a Grid infrastructure to execute the request services.

The enormous potential and clear advantages offered by Grid as infrastructure for Cloud computing motivated this work. In fact, Grid has emerged as a new infrastructure for distributed computing aiming at supporting the activities of scientific researchers. Recently, some researchers have adopted the concept of services at traditional Grid computing. Hence, addressing a novel concept, based in **S**ervice **O**riented **A**rchitecture (***SOA***) [3].

There are several advantages for Cloud computing users to adopt Grid as an infrastructure. The model discussed in this paper differs from others that use datacenter systems, such as clusters of servers, because there is no initial investment in infrastructure or staff. In addition, it keeps the same benefits of these approaches that use datacenters such as the total cost can be close to zero when resources are not in use. Therefore, a user will pay costs directly proportional to his needs [4].

However, it is not enough that communication occurs between geographically distributed applications from users of a Cloud. Furthermore, this communication should ack together with the features that cover ***QoS***[1] aspects. The users need for a mechanism to identify the best services that meet their needs with respect to ***QoS*** [5], for example, the metrics average response time and cost.

While several papers are concerned in showing the conceptual differences between Cloud computing, Grid computing and utility computing [6] [7] [8] [9]. This paper addresses the features of each approach and how they can be used together to build a complete and robust environment, by means of the extension of an environment proposed for Grids [10]. Hence, it is proposed an approach to use a MetaScheduler architecture for issues in a Cloud computing environment. Our approach is called ***MACC*** (**M**etaScheduler **A**rchitecture to provide QoS in the **C**loud **C**omputing). In ***MACC***, the main function is to handle submission for a service, and also to manage the infrastructure Grid resources, belonging to the Cloud, and to make the best available resources pool for implementation of the request sent by the user, by means of an economy model approach.

In order to provide ***QoS*** in Cloud computing, ***MACC*** monitors periodically the resources. It must use the Grid resources in an efficient way. ***MACC*** is a Grid component that deals with management, workflow, submission, scheduling, execution, monitoring, and finishing of the services sent to the Cloud.

It is ***MACC***'s function to receive notification messages of the local resources and the other services, such as: information services (monitoring), security, and execution. At moment, ***MACC*** receives a scheduling problem, it must solve the scheduling and return the order of the services to be performed. Sometimes, it is necessary to receive the scheduling order and to arrange it into a new decision from a feedback system which it is created based an economy models in ***MACC***'s layer.

The remainder of this paper is structured as follows. In Section II, the background and related works are addressed to

[1]Quality of Service.

explain how the work discussed in this paper is compared to the existing published works. A discussion of using ***MACC*** is made in Section III and its workflow is discussed in Section III-A. In Section IV it is presented the communication model adopted. The Section V presents the current economy model in ***MACC***. Finally, in Section VI, it is presented the final remarks and some directions for future works.

II. Background and Related Works

The use of the Grid as an infrastructure for services located in the Clouds came from the idea addressed in [11], whose understanding is given that Cloud and Grid are sharing systems that have the same basic idea: "to reduce the cost of compunting, increase reliability, and increase flexibility by transforming computers from something that we buy and operate ourselves to something that is operated by a third party.".

As can be seen in Figure 1, there is a relationship of overlaps between Cloud and other domains of distributed systems. Web 2.0 is targeted for service-oriented, where is also Cloud computing. Supercomputing and Cluster Computing are not concerned with service-oriented applications. Grid covers many fields and it is overlap with Cloud mainly in cases of lesser scale.

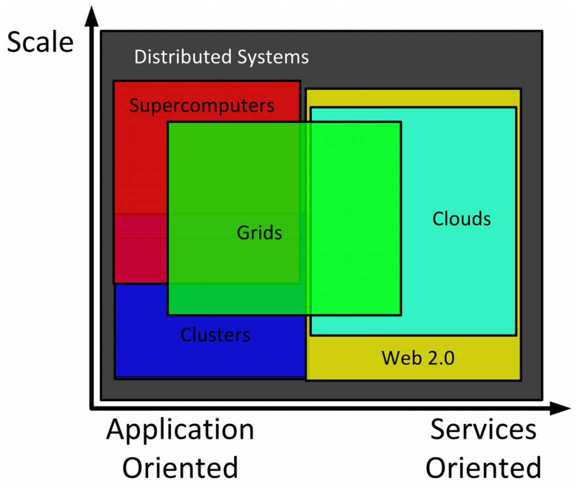

Fig. 1. Grids and Clouds Overview [12].

According to [13], computing infrastruture for e-Sciense is addressed to the Grid. Grid aims to "enable resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organizations" [3] [14]. In this vision, where the infrastructure working as a set of services that are providing by users or entities for consumptions by others, leads to the using of Grid Services [15].

This approach, to exploit an existing Grid infrastructure as basis for Cloud environment, can be adopted in place of big datacenters as it is done in a commercial world. The choice of this solution makes up new approaches for the Virtual Workspaces, like in the Nimbus project, and uses for cases of e-Science Clouds environments [16].

In this context, Grid computing can attend Cloud computing as infrastructure, since there is a control mechanism of the services available for execution. Normally, this mechanism is the Metascheduler or Broker. A Metascheduler can be defined as a scheduler that operates on top of the Grid. A Metascheduler not directly access the resources of an organization, it operates as a distributor of requests between the remote entities [10].

There are some Metaschedulers for general cases on Grid, as can be seen in Table I. Each one of these Metascheduler has one specification for use in certain cases. However, none of them is concerned with the additional issues that specifically involve the Cloud computing. It is necessary a Metascheduler that will take in account the features of Cloud, such as: where the client unaware of application deployment details, like the hardware that is actually operating the service, where it is located and its configuration issues. There should be a Metascheduler that allows to the client uses the services in a transparent way.

These issues are still in discussion. In current Cloud systems, the existing Metaschedulers are not prepared to find and to control the services in execution with the Grid as infrastructure. General Metaschedulers may not fit so well for the case of Cloud computing as happened with Grid computing.

III. MACC

By means of ***MACC*** is possible to express a business model built on a Cloud service. This business model deals with the management of services which can take into account issues related to the infrastructure of a Grid service, extending the domain and scope of applications of the participants. Another point that covers architecture, is the use of models based economy that helps in providing ***QoS*** for Cloud user's. To achieve this goal, the architecture deals with the priorities that are generated according to the contracted time, thus addressing the ***SLA***[2] stipulated. ***SLA*** contracts are constructed and dynamically adjusted. Finally, in the phase of experimental design, we will analyze how the proposed architecture can incorporate the requirements needed to provide QoS on a Cloud environment.

The jobs submition and handling the resources of the Grid infrastructure are the aims of ***MACC***, that invokes the service registry (**Index Service**) for discovery the resources. The addressing of each resource asking for both data on performance and notifications of resource changes is performed by the Index service. Just in case of resource has become unavailable or overloaded the **Trigger Service** component is operated.

Besides used for resources virtualization and provides other Grid features, the services also serve to specify in general terms how would be the operation of the service Metascheduler oriented. The service request is represented by a time (Ts) or cost (Cs) constraint that must be fulfilled. Therefore, to reache the meeting of the services in Cloud environment,

[2]**S**ervice **L**evel **A**greement.

TABLE I
METASCHEDULERS COMPARATION

Metascheduler	Middlewares	Scheduling Algorithms	License	Focus	Institution
CSF	Globus	Round Robin, Advanced Resource Reservation	Open Source	Provides a simple API for writing policies	University of Jilin, China
Gridbus Broker	Globus, Alchemi XGrid, Unicore	Deadline and Cost Otimization	LGPL	Otimizes utilization rate of resources	University of Melbourne, Australia
Gridway	Globus	Round Robin	GPL	Supports application self-adaptatives	University Complutense of Madrid, Spain
Nimrod-G	Globus, Legion Condor	Auction Based	Free, not commercial use	Focuses on computing, transparency	University of Monash, Australia
Condor-G	Globus, Unicore, NorduGrid	*none*	use, modify, redistribute	Fault tolerance	University of Wisconsin, Madison, USA

some concepts of the real-time systems (scheduling algorithms types, like ***EDF*** (Earliest Deadline First) and ***EBS*** (Earliest Deadline First) [17]) and economy models are applied.

A. MACC Workflow

As described in Figure 2, ***MACC*** behavior follows a workflow that begins when some user sends a service request (job). This request is sent to the component **Admission Control**. It applies utilization based policy, that is addressed in ***CPU*** utilization rate, accepting or dropping the request to the system. Then, request is sent to the component of the ***Trade Server/Client*** if the system is not overload.

The **Trade Server/Client** component is responsible for defining value and deadline of the service provided. At moment, the parties involved, client and provider, try to maximize the performance according with the objectives of each one of them, so service value (Cs_i) and deadline (Ts_i) is negotiated.

MACC query the ***Cache:UDDI*** component if the service cannot be found locally, so, if this component has the desired service then it returns the **S**tandard **I**nterface **A**ccess ***(SIA)***[3] to the client. Only services ***SIA*** which are known (by means of its neighbors in P2P[4] network) are stored by the the ***Cache:UDDI*** component.

If the service is in the same virtual organization which was searched, all services are identified and described by component **Workload Engine**, shown in the Table II. The Δ is used for achievement relative time – Ts – of service i, in other words, it is possible to recognize how long the service i will take to execute towards as a function of the utilization rate. The Δ represents a deviation in relation to the average of the i services time. Then, the ***MACC*** uses this component for a better backfilling of the jobs sent to the resources. For example, if utilization rate is 30% and the service i is required, the variable Ts_i receives $yi + \Delta^*$. Where Δ^* is normal deviation time, Δ^+ is loose deviation time, and Δ^- is strict deviation time obtained. This is done to make a laxity in service time in relation to the estimated service time.

[3]In case of Web Services is known as WSDL - (Web Services Definition Language).

[4]Peer-to-Peer

TABLE II
WORKLOAD ENGINE - ESTIMATION (%)

Utilization Rate (%)	Service 1	Service 2	Service i
$0 \vdash 25$	$x1 + \Delta^*$	$x2 + \Delta^*$	$xi + \Delta^*$
$25 - 50$	$y1 + \Delta^*$	$y2 + \Delta^*$	$yi + \Delta^*$
$50 - 75$	$z1 + \Delta^*$	$z2 + \Delta^*$	$zi + \Delta^*$
$75 \dashv 100$	$w1 + \Delta^*$	$w2 + \Delta^*$	$wi + \Delta^*$

It is also duty of ***MACC*** to select the best set of available resources through the **Workload Engine** component, and forward its details for component ***LRAM*** – Local Resource Allocation Manager. ***LRAM*** handles the job submition service and publishes some information about the local scheduler, i.e., queue length, available CPU, jobs under execution, and some memory statistics.

MACC start up a request to execution of job, if the resources are free, otherwise, the job is sent to the queue and will be executed when the required resources become available. When the job is completed, ***MACC*** is notified by the report service, which in its turn notifies the user.

The policies which will be performed by ***MACC*** must use the information about resources state, ***MMDS*** – ***M****anager of* ***M****onitoring and* ***D****iscovery* ***S****ystem*. The **Index Service** component will monitor and discover the information about the Grid and publishes it in only one place. Whenever changes in the resources utilization rate occurs, ***MACC*** is notified by

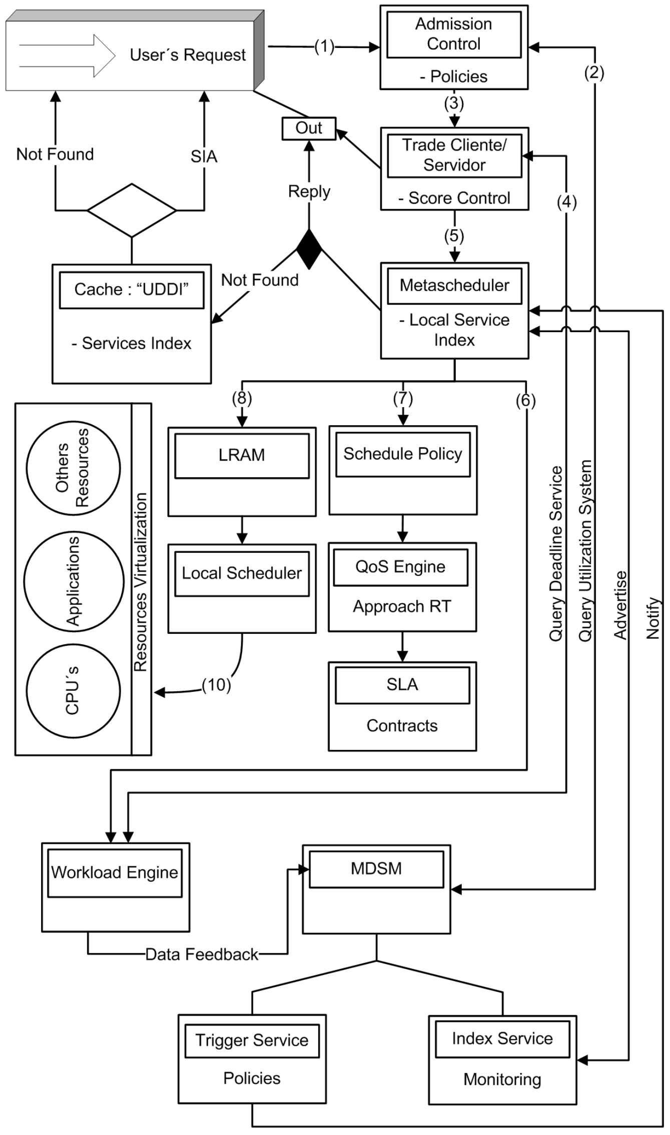

Fig. 2. ***MACC*** Architecture

the **Trigger Service**, remaining aware of all changes in the workload.

Under the provider's view, ***MACC*** must fulfills the constraints of the contract addressed by ***SLA*** component. On the other hand, under the client's view, ***MACC*** must accomplishes the choice of the best ***SIA*** among all available. ***SLA*** component covers the aggregate terms of the contract in use by defining the contract between the workload and resources manager. To each service running, ***MACC*** must keeps the real average response time (Ω) lower than the agreed average response time $(\Omega^{'})$. If $(\Omega \leq (\Omega^{'} + \Delta))$ can say that the contract was fulfilled and the difference between $(\Omega^{'} + \Delta)$ and (Ω) is the deadline (D). These parameter is controlled and used by the **QoS Engine** component to appoint how many and what resources will be used by the service.

The service differentiation for the clients is proposed in an individual way. Every virtual organization (named node) stores a score (Credit – $) which means the amount of services provided. This score is proportional to the time spent on activities of processing services, as was said earlier; this value is certain both by Trade Server and Trade Client components. Therefore, if the virtual organization provides many services during a lot of time then is possible that it will has a greater range of scores. There are an authentication and authorization module in the **Trade Client/Server** component, which become this score well identified.

MACC's proposed in this paper has the following characteristics:

- It works as a broker, does not directly control the Cloud resources. It only choose the features in accordance with the requirements imposed.
- It need not necessarily stay at same administrative domain.
- It figure out meet the deadlines according to the contract related.
- It minimizes impact that new services accepted in the environment could make about services that are already running
- It examines where the services need more resources to meet the associated metrics.

IV. Communication Model

Communication model is an important issue when the subject is scalability, because it affects on project decisions with regard to concepts about topological order and in case of Cloud the network latency. There are already three main communication model for Metascheduler that is **centralized model**, **distributed model** and **hirerchical model**.

In the **centralized model**, Figure 3, there is a main Metascheduler that maintain information of workload about all organizations participants of the Grid. All jobs that are submitted for this Grid are received by this Metascheduler that will take the decision of what organization will execute the job. This model has some disadvantages caused by the centralized Metascheduler as hard scalability and problems with fault tolerance [18].

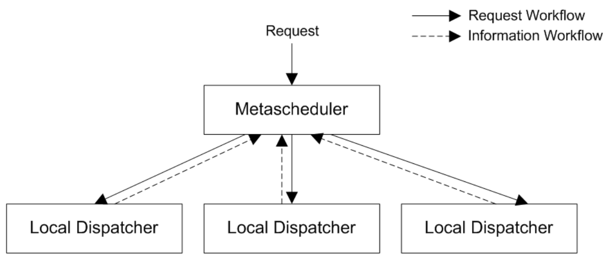

Fig. 3. Metascheduler centralized model. Adapted [18]

In the **distributed model**, Figure 4, each virtual organization has its own Metascheduler that periodically consults other Metascheduler to obtain instantaneous workload information of the Grid as a whole. The jobs are submitted for the local Metascheduler that will decide if the local organization will execute it or the job must be migrate to other organization with lower load [18].

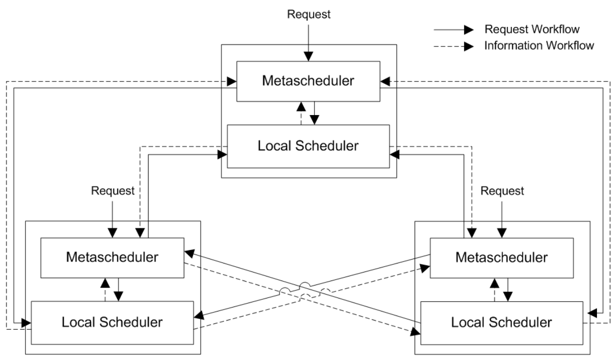

Fig. 4. Metascheduler distributed model. ADAPTED [18]

In the **hierarchical model**, Figure 5, the workflow is shared between the Metascheduler and virtual organizations, in this model the jobs are submitted for the Metascheduler and it does not maintain a queue of jobs, the Metascheduler sends the job to the virtual organization which the earliest time is expected for it. Each virtual organization can use different scheduling policies due to the fact that it has its local scheduler. When the job is submitted to the local scheduler in the virtual organization, its whole execution has to be made in there because the metascheduler does not influence in the tasks order [18].

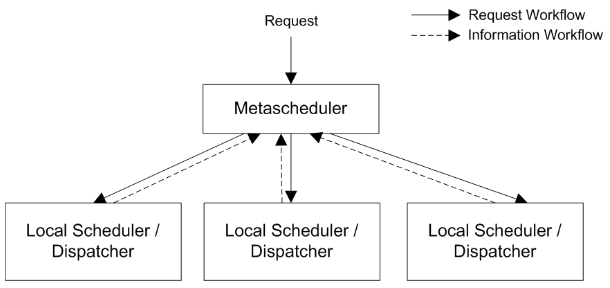
Request
Request Workflow
Information Workflow
Metascheduler
Local Scheduler / Dispatcher
Local Scheduler / Dispatcher
Local Scheduler / Dispatcher

Fig. 5. Metascheduler hierarchical model. ADAPTED [18]

In this paper it is proposed the **hierarchical P2P scheme**, an extending model of the hierarchical model, as shown in the Figure 6. This scheme is differentiated of the bare hierarchical because the elected Metascheduler working in the upper lever can performs managing of the others who are in lower levels. The idea about this model is to use the upper level Metascheduler for basically performs two functions: 1) to control partial overload in the service network and 2) to improve performance in query for services in the connectivity graph minimizing the congestion of the tasks in the Metascheduler.

An election is performed to choose what Metascheduler become in Superscheduler at the level $k+1$ of the graph

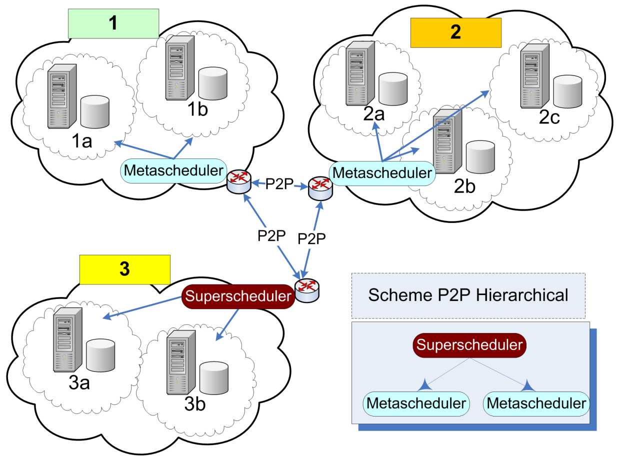

Fig. 6. Metascheduler hierarchical P2P scheme.

network. It happens when it has an estimation of capacity degree greater than their neighbors. This capacity degree, given by Θ, is calculated through the Equation 1, whose S_i denotes the amount of services, ΔS_i denotes the average execution time of service i, and Pc_n denotes the power computational of resource n. f returns an integer that represents Θ of virtual organization. It is possible to determine the weight to each one of these parameters, such as: $\propto$ - related to the amount of services and ϖ - related to the power computational. This election happens both inside (become Metascheduler) in ***VO*** and outside (become Superscheduler) among ***VO's***, both using the Equation 1.

$$\Theta = \left(\frac{\sum S_i}{\sum(\Delta S_i)} * \propto \right) + \sum (Pc_n * \varpi) \qquad (1)$$

Inside the virtual organization the election should address what Metascheduler into each site will be responsible for management of Virtual Organization. Each site has one Metascheduler, but just one between them will be the representative of Virtual Organization. Outside the virtual organization the Superscheduler happens for all virtual organizations on the P2P layer, which is used for the management of other virtual organizations. The goal here is to use this approach that will receive one requisition and will find out whom will execute it, concerning the hierarchical P2P model.

The metric used to determine the Metascheduler level (k) was formed to improve the fairness at query moment of the system, because the virtual organizations that has the greater function value of amount of services in relation to the execution time of them, and greater power computational will be in the same logical cluster. The dynamics about the building of graph and the election of Metascheduler focuses on how this structure is based. Therefore, it is adopted a data structure based on ***P2P***, because this structure type performs few changes on the graph and minimizes the overhead of maintenance. In this approach, each virtual organization is a node in the graph and its index is represented by Θ, as shown in Figure 7.

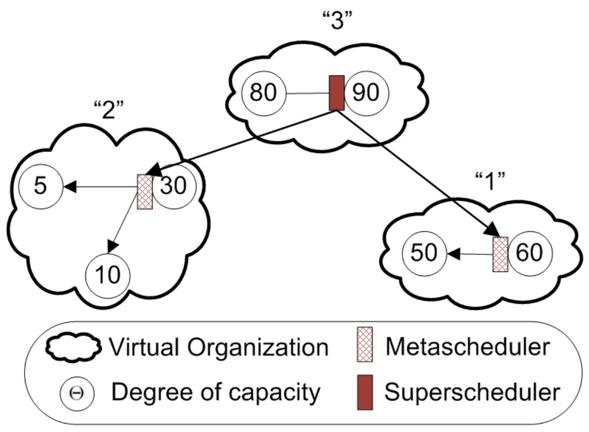

Fig. 7. MACC's Applied.

Each virtual organization has one elected Metaescalonador for ***VO***. In the case of Figure 7, the Superescalonador is the node "3", because it has the largest value between Metaescalonadores. Therefore, the search operations and submission service of Metaescalonadores are managed by Superescalonador. Thus, it figure out a virtual organization that contains the desired service through P2P infrastructure.

An environment that represents both the infrastructure and the links between resource managers can be seen in Figure 8.

Under the client's view, the request for a service occurs in transparent way (addressing the Cloud view). The request is forwarded to the ***VO*** that the user belongs, addressing in Figure 8 as can be seen on step (1). If the service not available inside the ***VO***, the Metaescheduler forwards the request service for the Superscheduler (step 2) within the P2P network of interconnections, which in its turn makes the search for the requested service in other ***VO's*** that are connected the P2P overlay layer. Next, the Superscheduler sends the ***SIA*** for client (step 3), and it invokes the service in ***VO*** which has it (step 4) and continues on (step 5). Finally, ***VO*** performs the service and returns the results to the client (step 6).

V. Economy Grid Model in MACC

This work uses an economy model based on supply/demand. In this type of model, the providers publish their services and the clients realize a choice by means of metrics that demand a better service selection. Thus, an environment for sharing resources is enabled. In these environments, the providers with greater contribution have more probability to be selected when needed. This approach makes possible to regulate the priority from scheduling. The benefits on environments with these characteristics are to provide flexibility and breadth of performance for all interested towards the changes. Besides, it does not need of a coordinator in the trading.

The aforementioned approach is used in real time applications. A support for deadline constraint can be added with cost constraint. Deadline constraints means to limit the time that the providers have to execute the service. This deadline is assigned by trade client and trade server components. The approach supplies an incentive for the providers to share their resources on the Grid take into account cost and deadline. Deadline is represented with (D_n) and it is obtained from TS_i variable. A component that provides a workload engine is responsible to bring forth this variable.

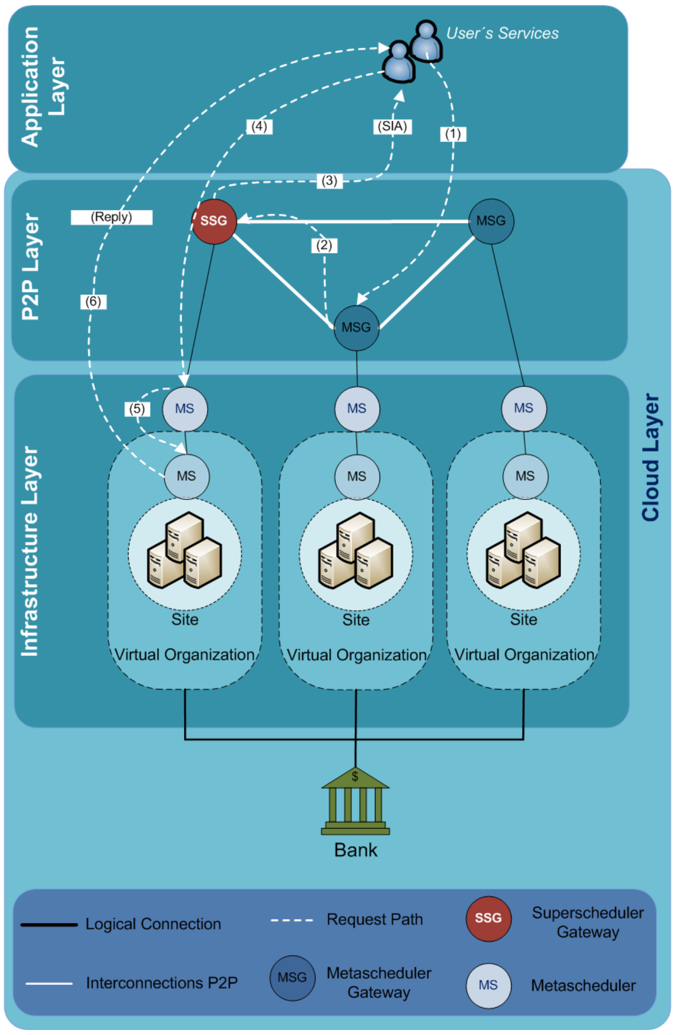

Fig. 8. Peering Arrangement MACC.

The resource price has effects in the service utilization. The consumers tend to use those services with lower price. In [19] is presented a Trade Server component that deals with a mechanism for automatic prices. The authors show a mechanism of prices adjusts aiming to maximize the resources utilization rate. The parameters were adapted and adjusted in this work for consideration of deadline constraint. These parameters are highlighted:

Demand (d_n): This parameter represents the price to execute one service (understand as resource). It is proportional to its demand. Equation 2.

$$d_n = \frac{(d_n - d_{n-1})}{d_{n-1}} * \alpha \tag{2}$$

Supply (O_n): This parameter represents the price to execute one service. It is inversely proportional to its offering. Equation 3.

$$O_n = \frac{(O_n - O_{n-1})}{O_{n-1}} * \beta \quad (3)$$

Supply/Demand (dO_n): The relationship between the amount of requests and the amount offered to one service is represent by this parameter. If the offering is higher than the demand, the price tends to decrease. In contrast, if the demand is higher than offering, the price tends to increase. In the cases that there is equality between demand and offering, the Grid has reached the equilibrium. Equation 4.

$$dO_n = \frac{(d_n - O_n)}{min(d_n, O_n)} * \varrho \quad (4)$$

Utilization Rate (U_n): This parameter can be increased if the utilization rate is high. The price may be high and must be decreased if the utilization rate is high. When the system utilization rate reaches 80% is additioned a new value in the equation, as can be seen in Equation 5.

$$U_n = \left(\left(1 - \frac{U_{n-1}}{100}\right) * \delta\right) + \left\lfloor \frac{U_{n-1}}{80} \right\rfloor * \gamma \quad (5)$$

Deadline (D_n): This parameter represents the delivery time of a service. The price becomes lower if this time is loose. Otherwise, if this time is strict, the price becomes higher. Equation 6.

$$D_n = \frac{(D_n - D_{n-1})}{D_{n-1}} * \epsilon \quad (6)$$

These parameters can be determined by the provider administrator that can define the weight to each one of them. The parameters, such as: demand (α), supply (β), supply/demand (ϱ), utilization rate (δ) and (γ), and deadline (ϵ) ranging from 0 to 1. The new base prices of these parameters are obtained following the schema presented in equation 7. This equation connects the new prices in relation to these parameters that change over time.

$$P_n = P_{n-1}\left(1 + (d_n - O_n) + dO_n - D_n - U_n\right) \quad (7)$$

Additionally to the establishment of base price, there is the workflow trading based on the studies presented in [20] [21]. The Figure 9 shows the sequence of steps to define this workflow. A client produces a requisition (***TB WSLA***[5]) containing all information about desired QoS. Then, the connection is established. Multiple providers can be accessed for a client. Therefore, this protocol has a timeout to determine how long the information is valid in the proposal. Moreover, the client and the provider perform the confirmation of the ***QoS***. In the last step, the client closes the connection.

[5]Template Basic on ***WSLA***.

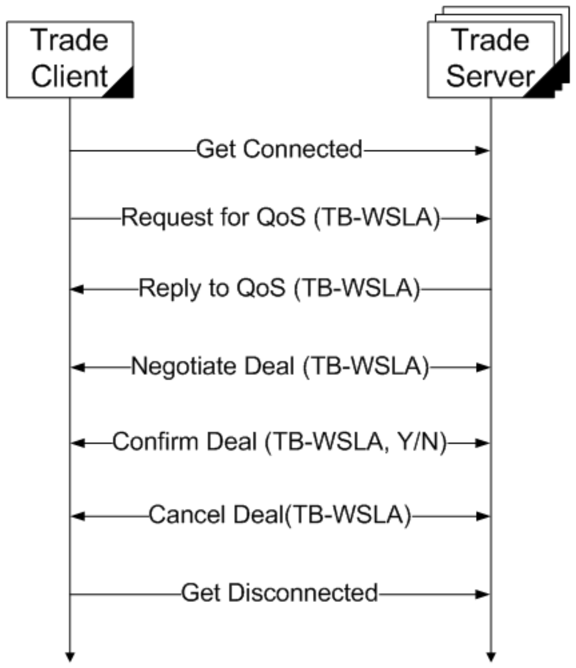

Fig. 9. Trading Protocol - Adapted [21].

VI. Final Remarks

This paper proposes the use of a metascheduler architecture to provide ***QoS*** in Cloud computing, called ***MACC***. ***MACC*** is used for handling computing resources in a Cloud computing environment. It was presented the main functions of ***MACC*** and its application in a real environment through the use of techniques to search of data using P2P, and an economic model that become possible the provision of resources to be requested by users of the Cloud and also a communication model to ensure the minimum scale considering aspects of quality of service.

By means ***MACC*** is enabling expresses a business model based to the Cloud environment, this framework offers a business model for management of the services which can takes into account the issues related to the Grid infrastructure, extending the domain and scope of the user's applications.

Another goal addressed in this paper was the using of economy-based models that help in offering ***QoS*** in the Cloud. To achieve this goal, ***MACC***'s deals with the priorities that are generated according to the contracted time, addressing a ***(SLA)*** stipulated. The contracts in the ***SLA*** are dynamically adjusted. Thus, the user that providing more cycles of processing time will be one that holds the highest priority when make requisition of a service.

Summarizing, the ***P2P*** Hierarchical Metascheduler in ***MACC*** can offer QoS for a Cloud environment and it is becoming an increasingly prominent element in this architecture. The Metascheduler is responsible for ensuring that clients have their services meeting under the contract stipulated in ***SLA***. It works on the Middleware layer. Besides of these Metascheduler, can be added other initiatives to provide ***QoS***, scalability and interoperability. These initiatives relate to the adoption of Web services as mechanisms of communication and economic models for handling value and deadline of involved services.

VII. Acknowledgments

The authors thank the financial support from FAPESP, CNPq and CAPES for the projects under development at the Distributed System and Concurrent Program, group of the Computes Systems Department at ICMC-USP, as well the valious contributions from Ph.D. students at ICMC-USP, in special, Douglas by his corrections.

References

[1] B. Ohlman, A. Eriksson, and R. Rembarz, "What networking of information can do for cloud computing," in *WETICE '09: Proceedings of the 2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 78–83.

[2] M. A. Vouk, "Cloud computing: Issues, research and implementations," in *Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on*, 2008, pp. 31–40. [Online]. Available: http://dx.doi.org/10.1109/ITI.2008.4588381

[3] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," *Lecture Notes in Computer Science*, vol. Vol: 2150, 2001. [Online]. Available: citeseer.ist.psu.edu/foster01anatomy.html

[4] J. Napper and P. Bientinesi, "Can cloud computing reach the top500?" in *UCHPC-MAW '09: Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*. New York, NY, USA: ACM, 2009, pp. 17–20.

[5] A. Erradi and P. Maheshwari, "A broker-based approach for improving web services reliability," in *ICWS '05: Proceedings of the IEEE International Conference on Web Services (ICWS'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 355–362.

[6] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *HPCC '08: Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 5–13.

[7] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson, "Cost-benefit analysis of cloud computing versus desktop grids," in *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–12.

[8] G. V. Mc Evoy and B. Schulze, "Using clouds to address grid limitations," in *MGC '08: Proceedings of the 6th international workshop on Middleware for grid computing*. New York, NY, USA: ACM, 2008, pp. 1–6.

[9] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 2009.

[10] M. L. Peixoto, M. J. Santana, and R. H. Santana, "A p2p hierarchical metascheduler to obtain qos in a grid economy services," *Computational Science and Engineering, IEEE International Conference on*, vol. 1, pp. 292–297, 2009.

[11] I. Foster, "There's grid in them thar clouds - weblog," 2009, available in: http://ianfoster.typepad.com/blog/2008/01/theres-grid-in.html. Last access: 25/10/2009.

[12] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE '08*, 2008, pp. 1–10. [Online]. Available: http://dx.doi.org/10.1109/GCE.2008.4738445

[13] I. Foster and C. Kesselman, *The Grid. Blueprint for a New Computing Infrastructure.: Blueprint for a New Computing Infrastructure (Elsevier Series in Grid Computing)*, 2nd ed. Morgan Kaufmann, December 2003. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1558609334

[14] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," 2002. [Online]. Available: citeseer.ist.psu.edu/foster02physiology.html

[15] D. De Roure, N. R. Jennings, and N. R. Shadbolt, "Research agenda for the semantic grid: A future e-science infrastructure," National e-Science Centre, Tech. Rep. UKeS-2002-02, December 2001.

[16] E. P. Mancini, M. Rak, and U. Villano, "Perfcloud: Grid services for performance-oriented development of cloud computing applications," in *WETICE '09: Proceedings of the 2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 201–206.

[17] L. S. Casagrande, F. J. Monaco, R. F. de Mello, R. Bertagna, and J. A. A. Filho, "Exigency-based real-time scheduling policy to provide absolute qos for web services," in *SBAC-PAD '07: Proceedings of the 19th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'07)*. Gramado, RS, Brazil: IEEE Computer Society, 2007.

[18] V. Subramani, R. Kettimuthu, S. Srinivasan, and P. Sadayappan, "Distributed job scheduling on computational grids using multiple simultaneous requests," in *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 2002, p. 359.

[19] F. Teixeira and B. Toledo, "Computational grid architecture based in economy models," in *5th International Information and Telecomunication Technologies Symposium, Cuiaba, MG - Brazil*, 2006.

[20] S. Benkner and G. Engelbrecht, "A generic qos infrastructure for grid web services," in *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*. Washington, DC, USA: IEEE Computer Society, 2006, p. 141.

[21] R. Buyya, D. Abramson, and J. Giddy, "A case for economy grid architecture for service oriented grid computing," in *IPDPS '01: Proceedings of the 10th Heterogeneous Computing Workshop â HCW 2001 (Workshop 1)*. Washington, DC, USA: IEEE Computer Society, 2001, p. 20083.1.

[22] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy, "A distributed resource management architecture that supports advance reservations and co-allocation," *Quality of Service, 1999. IWQoS '99. 1999 Seventh International Workshop on*, pp. 27–36, 1999.

[23] I. Foster, K. Czajkowski, D. Ferguson, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke, "Modeling and managing state in distributed systems: the role of ogsi and wsrf," *Proceedings of the IEEE*, vol. 93, no. 3, pp. 604–612, March 2005.

[24] D. C. Vanderster, N. J. Dimopoulos, and R. J. Sobie, "Improved grid metascheduler design using the plackett-burman methodology," in *HPCS '07: Proceedings of the 21st International Symposium on High Performance Computing Systems and Applications*. Washington, DC, USA: IEEE Computer Society, 2007, p. 9.