Competitive P2P Scheduling of Users' Jobs in Cloud

Beniamino Di Martino, Rocco Aversa, Salvatore Venticinque, Luigi Buonanno
Department of Information Engineering
S.U.N. (Seconda Università di Napoli)
Aversa (Italy)
[Beniamino.DiMartino,Rocco.Aversa,Salvatore.Venticinque,Luigi.Buonanno]@unina2.it

***Abstract*— Existing distributed solutions for distributed computing (Grid, Cloud, etc.) pose a high threshold for potential customers. The reason deals with the technical background and effort that are usually required in order to successfully access the computing facilities, thus limiting their massive adoption. By exploiting the features offered by different distributed paradigms (P2P and Cloud), we propose here an approach that reverses the role of resource requestors and resource providers, allowing potential customers to access the distributed infrastructures in a user-friendly fashion. In the proposed scenario, the task of retrieving the user's submitted jobs and configure accordingly the necessary resources is in charge of the providers, thus lowering the threshold required to successfully exploit the computing facilities. The experimental activities, described in the paper, validate the hypothesis that a competitive approach, in distributed scheduling environments, can decrease the threshold required to access the facilities and lead, if properly set up, to substantial performance gains.**

***Keywords*- P2P; cloud; competitive scheduling.**

I. INTRODUCTION

Cloud computing [26] is a recent model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. On one hand, thanks to the virtualization technology, providers can rent their hardware resources in a very flexible way. On the other hand, users may have a dedicated data center as a service without the burden of buying and managing expensive hardware, but rather paying their utilization according to a pay-per-use business model.

Despite the benefits provided, many open issues have to be addressed with regard to this emerging computing paradigm. Some of them are portability of applications, lock-in proprietary solutions, negotiation and check of SLAs (Service Level Agreement) with Cloud providers. Among the others, an open issue has affected most of the distributed paradigms which have been spreading for the last few years: existing solutions require the customer to hold an advanced technical background in order to successfully exploit the computing facilities, thus limiting their massive adoption.

The list of issues a potential user has to deal with includes: the discovery of the architecture that is compliant with the application requirements, the setup of the execution environment, the research of the most convenient offer, the configuration of the acquired resources, the tuning of the applications, the uncertainty of execution time due to a best effort policy for resource sharing, etc.

Other distributed paradigms (e.g., inverted client-server systems [3]) do not pose such a high threshold to potential customers, but they do not encourage the intensive exploitation of resources.

P2P (Peer-to-Peer) [23] refers to logical organization of computing entities where each individual knows its neighbors and can behave both as a server and a client. There are some relevant examples of P2P systems, oriented to parallel and/or distributed computing, which have been successful in their exploitation.

In order to address the described issues, we propose a distributed paradigm that:

- Aims at implementing the same ease of use of P2P file sharing applications.
- Reverses the roles of requestors and providers, by charging the providers of all the overhead required to setup the execution environment, manage the job requirements, etc. In our model, clients just publish their jobs on the platform, specifying the software and hardware requirements, the application details, the deadline and the offered reward. Service providers, on the other hand, are in charge of discovering the published jobs and of addressing all the issues related to the jobs' requirements management;
- Adopts a competitive approach, where providers compete for satisfying the client's requests and are awarded with credits in case of successful elaborations, thus optimizing client's satisfaction and reducing the cost.

In the next section, we discuss related work. The third section introduces a comparison of policies for resource sharing in centralized and P2P networks. In the fourth section, we present our competitive approach for job scheduling in P2P. In the fifth section, we provide a description of a prototype implementation and we show experimental results aimed at evaluating the effectiveness of the proposed solution. Finally, we present the conclusion.

II. RELATED WORK

Cloud computing is an on-demand distributed paradigm that refers to providers offering a large pool of easily usable and accessible virtualized resources in a pay-per-use model [1]. The services can be delivered within different layers, that are usually classified as SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) [24]. This paper mainly refers to SaaS and IaaS clouds. Cloud computing allows data centers to transparently offer services through the Internet by exploiting their computing and storage fabric of resources. In SaaS and IaaS clouds, applications and nodes are virtualized and dynamically provisioned on-demand as a personalized resource collection to meet a specific service-level agreement, which is established through a negotiation. A market-oriented resource management is necessary to regulate the supply and demand of Cloud resources [9], providing feedback in terms of economic incentives for both Cloud consumers and providers, and promoting QoS-based (Quality of Service) resource allocation mechanisms that differentiate service requests based on their utility [2]. Many research contributions aim at supporting the user with negotiation services based on Service Level Agreement that delegate to agents the discovery and agreement of the best offer from multiple providers [7], [11]. The main aim of this paper is to bring this mechanism a step further, by delegating this task to providers.

Security is still a big concern in cloud frameworks. While in Grid [25] environments, indeed, both resources and users need to be registered and to get a digital certificate for authentication and authorization purposes, before they are allowed to start a session. This mechanism is feasible when the number of nodes are not many. Security in cloud computing infrastructures, instead, is still, mostly, a work in progress. Nevertheless, some analysis on this topic have been performed [17]. Furthermore, it must be said that the very subject of security is what is slowing down the adoption of cloud computing over other forms of distributed scheduling [18].

Current P2P systems have the perk of allowing a very high number of users (hundreds of thousands is a common figure). They offer few services, without doing assumptions on the reliability of the peers themselves [14]. However, it is very complicated to ensure a given QoS level [13] without any sort of distributed scheduling. From a security point of view, P2P systems are, by definition, environments where it is difficult to be aware of the identity and trustability of hosts: the chance of exploiting a malicious resource is intrinsically high. While this risk is largely accepted for file sharing systems, in order to make it acceptable for distributed computing many issues must be addressed to ensure the safety of both the code owner and the code executor.

In [11], an architecture for the resource sharing on large scale networks has been described (CompuP2P). CompuP2P uses a protocol based on Chord [16] and detects a set of ”dynamic markets”, each of them groups all the peers that are willing to buy or sell the same ”amount” of computing power. The main bottleneck is represented by a special peer (”Market Owner), that is responsible for the association between requests and offers of computing power. In [15], a solution for the scheduling of multiple applications, in a concurrent fashion, is proposed. Authors propose a decentralized scheduling pattern and do a comparative analysis of different heuristic logics. Many Grid solutions for task scheduling and workload distribution exist. For example, Condor [12] is a high-throughput distributed batch computing system that provides job management mechanisms, scheduling policies, resource monitoring, and resource management. However, it can hardly be defined as a P2P system, cause of the presence of a central manager that accepts job submissions. Conversely, the objective of our research is to design a P2P infrastructure that is not relying on any centralized element and that enables a huge numbers of machines, which connect/disconnect dynamically to the network without any guaranties on their reliability, to easily access the resources offered by cloud providers.

III. CRITICAL COMPARISON OF DIFFERENT APPROACHES FOR RESOURCES SHARING

In the computational grids model, providers offer their services with a best effort policy and a collaboration pattern is usually adopted among different parties, which share their resources belonging to a virtual organization, in order to optimize the global performances. Grid clients compete to use the resources: this model exploits the competition of clients and the collaboration of servers.

According to a common opinion, the business Grid model was unsuccessful because providers are business competitors and, usually, do not collaborate. However, even if theoretically the market should rely on the competition of providers, often, in real-world scenarios, sellers cooperate rather than competing while, at the same time, trying to create competition among buyers.

It is a model similar to the one that is currently adopted in the automotive, where different companies share engines and other components, or in the insurance field, where prices are fixed above a threshold using a behavior that is, at least, at the edge of the law. Great companies have much interests and resources to organize themselves for collaborating. Even if powerful ones should give they usually take, by choosing to collaborate, rather than fighting, when it means a bigger return. Collaboration of providers is exploited to take.

In volunteer computing, clients are asked to donate CPU cycles when their computers are idle.

Users’ resources are then managed and exploited by powerful big organizations. In fact, they have the capacity to exploit all the limited resources shared by a huge number of distributed users. The lack of this kind of organization ability, and, at the same time, the great capability of users in terms of availability is evident in real life and in distributed computing. In volunteer computing, collaboration among users is exploited *to give*. The most well-known case of volunteer computing is the SETI@Home project [3].

P2P is a successful example of decentralized resource sharing among clients. In P2P file sharing systems, users compete to download files from the available sources and are asked to share their data (“collaborate”) in change of credits that can be spent for acquiring download privileges. Competition is easier to be implemented, because organization is not required. Competition among users is exploited *to take,* while the collaboration is used *to give.*

Cloud computing is a new paradigm that was born in a business context. The business model is pay per use and it is not based on resource sharing. A limitation of this approach is that, if a user chooses a solution from a particular provider, he will be locked by that choice because of the lack of portability.

In this scenario, it would be useful to design a technological solution that implements a business model aimed at optimizing the QoS at user side, and to maximize the incomes at server side.

A free market model, that exploits the competition of sellers to give computing power and collaboration of users to take it, could be the best solution.

Our approach proposes the utilization of a P2P model that allows the users to collaborate by publishing their jobs as “calls for proposal” (cfp), in the same way as it occurs in file sharing systems. On the other hand, the business model is based on the competition among servers, which seek shared proposals and try to answer as soon as possible in order to obtain the offered reward.

IV. P2P COMPETITIVE SCHEDULING OF USERS’ JOBS

In this section, we propose a competitive approach for P2P distributed computing, whereas the roles of involved parties are inverted if compared to the Grid, Cloud computing or web services models: clients publish jobs on a P2P network overlay (“call for execution”) while the servers look for these and compete to deliver the results. Calls for execution describe the requirements of the application, the credits the user would pay and, optionally, a deadline before which the results should be available.

While in the Grid model, and in traditional architectures for distributed scheduling, the job owner is in charge to choose the execution node, to check its compliance with the application requirements and to ask for the execution, in the proposed model these issues must be managed at server side. We think that the proposed approach would be very effective in the Cloud market, where providers can set up virtual, specialized environments for the execution of different jobs and use the idle ones to satisfy the user’s request. Virtualization is commonly used by Cloud providers to improve the throughput of their hardware resources: thanks to the modern Cloud computing paradigms, the configuration of the task execution environment can be easily adapted to match the application requirements by exploiting the virtualization technology.

In our model, providers can exploit at the best their resources, and the Cloud IAAS, by managing both their overbooking and their smart scheduling. We try to design our model as much similarly as possible to current P2P systems for file sharing whose success in the Internet community has been bigger than the Grid.

It is evident, according to what has been discussed in the previous paragraphs, that many issues must be addressed in order to consider the P2P model as a viable relay for distributed computing at business level. This topic is out of the scope of this paper.

In the model, two kind of peers are defined: buyer and seller peers. User peers publish application descriptors, in the same way a file is commonly shared in P2P file sharing systems. The descriptor includes all the hardware and software requirements, as well as other constraints like the time within the results must be delivered and the offered reward. Clearly, it includes the info required for retrieving the task code and data. Seller peers crawl the network looking for published jobs, analyze the constraints, and choose to accept the proposals according to their own policy. For each retrieved request, the seller peer is able to evaluate its ability to fulfill the requirements and its convenience to accept the task. Multiple seller peers can accept the same task, and different patterns can be defined, e.g. the buyer could state that only the first business peer that delivers the results will be awarded, so that the seller peers will have to compete for being be the first one that completes the job. However, this is not the only possible pattern (e.g. in SETI@Home, multiple peers execute the same tasks and results are matched against each other).

Our model allows asynchronous mechanisms to be adopted: the user peer who published its job can disconnect, being aware that the results will be delivered according to what is specified in the job descriptor. This approach could be effective within today business scenario, where multiple providers exist and compete to promote their own services.. Furthermore, it allows for some flexibility (e.g. sellers could act as brokers that use resources provided by commercial Clouds providers).

Some keystones of the approach are:

- Client peers publish ”calls for execution”;
- Server peers discover and download calls for execution. Furthermore, they retrieve the code to be executed and the data;
- Server peers compete to complete as many as possible jobs to maximize their incomes;
- Clients can disconnect at any time: computation continues at server side;
- Workload balancing can be implemented.
- A business model is required to promote the execution of one’s own applications.
- It is effective in an industrial environment.
- Configurations of virtual machines, or general computing resources, are set up according to the application’s specific requirements.

V. IMPLEMENTATION

A prototype implementation of the above described model has been developed. In our implementation, the client and business peers are named, respectively, Buyers and Sellers, to highlight the market-like modeling of the system.

We have extended jKad [20], a publicly available open source implementation of the Kademlia [19] protocol released under GNU Lesser GPL. Each actor is composed by a set of different modules, each one performing a specific task. Figure 1 shows the modular architecture we have implemented.

The Buyer is composed of three modules:

- The *P2P GUI*, that implements a Graphical interface that allows the user to define the job properties according to the ontology.
- The *Job Sharing* is responsible for publishing the jobs submitted by the user into the P2P overlay.
- A *Network* module, that interacts with the Sellers, exchanging data files and results.

The Seller is composed of:

- A *Job Discovery* module, that is in charge of crawling the network in order to discover available CFEs (Call for Execution).
- A Parser, that analyzes the retrieved jobs. Furthermore, it interacts with the Buyer's Network module to retrieve the data required for the job's execution.
- A Job Queue Manager, that sequentially schedule the jobs.
- A Result manager, responsible for interacting with the Buyer's Network module in order to return the results.

A. *P2P technology*

The underlay system is a Kademlia-like P2P network. It has been chosen because of the major properties that DHT-based (Distributed Hash Tables) [27] P2P systems bring to applications (predictability of key research, robustness against node failures, etc.) and because of its simple protocol. In fact, only four messages are defined by the protocol:

- PING (node): to verify if a peer is still alive.
- STORE (key,value): to store a (key,value) pair in one or more nodes of the network.
- FIND NODE (node): to retrieve the k nodes that are closest (according to a XOR metric) to the node used as parameter.
- FIND VALUE (key): a node receiving this message returns the corresponding value if it has the requested key in his store. Otherwise, it will behave as upon receiving a FIND NODE.

As described in Figure 2, Buyers can publish jobs at any moment. Sellers look for shared CFEs and choose, for each job, whatever it is convenient or not to accept it. Once the job's descriptor is downloaded, the Seller can start the job execution. Results are then returned to the Buyer; consequently, the Seller can get its reward if the results and the timing are compliant to the job requirements.

As already mentioned in the previous paragraphs, multiple patterns are possible: the reward could be awarded to the first Seller who executes the task, or to all those able do deliver the results before a given deadline, or to the first "*n*", etc. The analysis of this topic is beyond the scope of this paper.

B. *Prototype description*

The P2P GUI module allows users to specify the job requirements and to publish CFEs. The input form (Figure 3) is dynamically drawn by the application according to an OWL template. A hash of the data entered by the user is then calculated and published into the P2P overlay network.

Publishing is performed by the Job Sharing module and it is implemented as a simple STORE message on the Kademlia network, using a special label as a key that identifies the shared job descriptor.

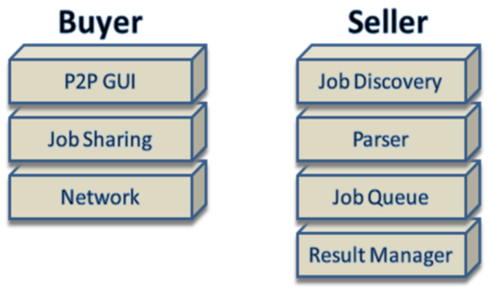
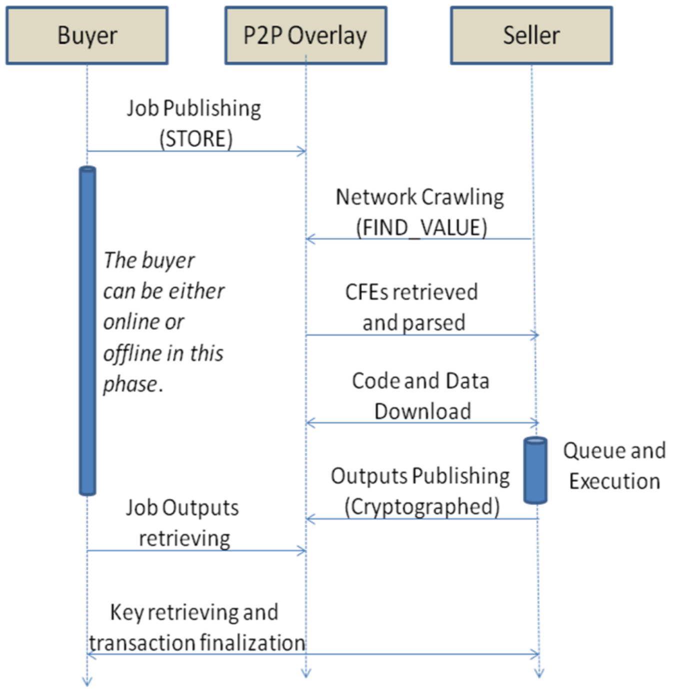
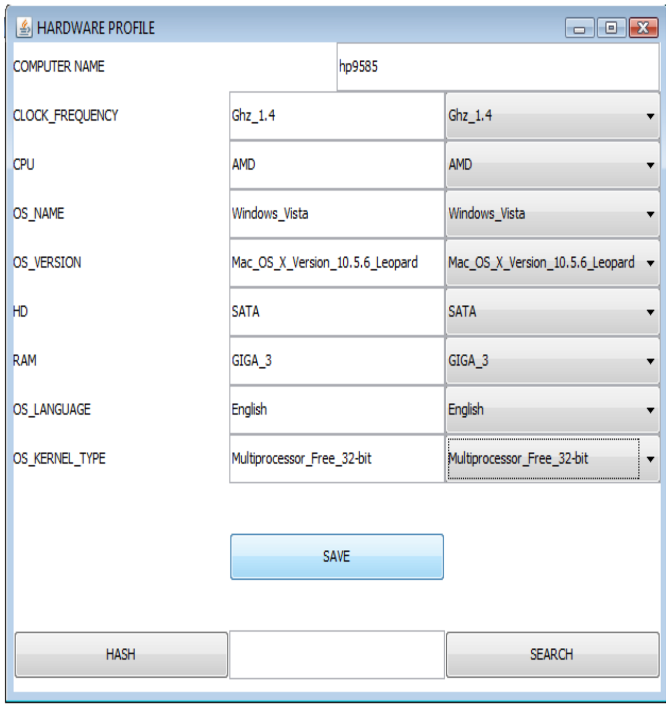


Figure 1. The architectural model



Buyer
P2P Overlay
Seller
Job Publishing (STORE)
Network Crawling (FIND_VALUE)
The buyer can be either online or offline in this phase.
CFEs retrieved and parsed
Code and Data Download
Queue and Execution
Outputs Publishing (Cryptographed)
Job Outputs retrieving
Key retrieving and transaction finalization

Figure 2. Sequence diagram describing the seller and buyer interactions.



COMPUTER NAME	hp9585	
CLOCK_FREQUENCY	Ghz_1.4	Ghz_1.4
CPU	AMD	AMD
OS_NAME	Windows_Vista	Windows_Vista
OS_VERSION	Mac_OS_X_Version_10.5.6_Leopard	Mac_OS_X_Version_10.5.6_Leopard
HD	SATA	SATA
RAM	GIGA_3	GIGA_3
OS_LANGUAGE	English	English
OS_KERNEL_TYPE	Multiprocessor_Free_32-bit	Multiprocessor_Free_32-bit

Figure 3. User interface for job publishing and discovery

Sellers, through the Job Discovery module, crawl the network looking for CFEs, using the FIND VALUE message. Once a job is retrieved, its descriptor is analyzed by the Parser module. The descriptor contains information about the type of executable, application requirements, the time constraints, the reward and further details that are summarized in Table I. The Parser, then, will interact with the Buyer's Network module to download code and data required by the job. Notice that, for simplicity, no negotiation mechanism has been implemented: once a job descriptor is retrieved, the Seller checks if the requirements can be fulfilled, then decides if it is worth to accept the task.

The decision-making mechanism does not take in consideration whatever any other Seller could be already on the same CFE. Accepted jobs are managed by the Job Queue module, that will sequentially schedule them. Finally, the Result Manager will interact with the Buyer's Network module in order to return the outputs.

An ontology has been created, that allows to define the application details and the application specific hardware/software requirements in a not ambiguous way. Some of the concepts are listed in Table II.

VI. EXPERIMENTAL RESULTS

In order to evaluate the behavior of the described prototypal implementation, under both functional and performance points of view, a testing environment has been set up. The developed software platform emulates the proposed approach, enabling the analysis of the system dynamics, including the overhead introduced by the adoption of the Kademlia protocol. In Figure 4, a communication diagram of the software platform is showed.

TABLE I. APPLICATION DETAILS

Variable	**Meaning**
Universe	Specific the kind of application that is been submitted (Exe file, java executable, etc.)
Unique ID	Identifier used to retrieve the code and data inputs on the overlay network.
Executable file	The name of the main executable file
Input	The url where the package can be retrieved
Output	The url where the results can be sent
Contract owner	A unique indentifier of the job submitter
Budget	The reward offered for the job execution
Deadline	The date by which the task must be completed
Owner email	Email contact of the owner

TABLE II. APPLICATION SPECIFIC HARDWARE AND SOFTWARE REQUIREMENTS

Variable	**Meaning**
CPU Architecture	Possible constraints on the CPU type
N.of CPUs	Number of required CPUs
RAM	Minimun amount of available ram required
Libraries	Possible required libraries – Optional
OS	Possible required OS – Optional
Storage	Minimum amount of free storage required

Different test cases have been defined. Each of them is characterized by a set of meaningful parameters, whose combination leads to a different statistical behavior of the system. The most relevant parameters that can be set for each test case are:

- Job Arrival rate.
- Number of peers in the system.
- Mean and standard deviation. It depend both on the computational requirement of the task and on the computing power of the seller peer. Times are modeled as Gaussian distributions.
- Cool down (time between subsequent network scans).
- Maximum allowed concurrency level (MAC). It is the number of peers that can simultaneously compete on a single task.

The test analysis has allowed us to detect interesting system dynamics. In particular, we evaluated the mean queue time that a job has been waiting inside a buyer's queue, the mean execution time and the mean time of permanence within the overlay network.

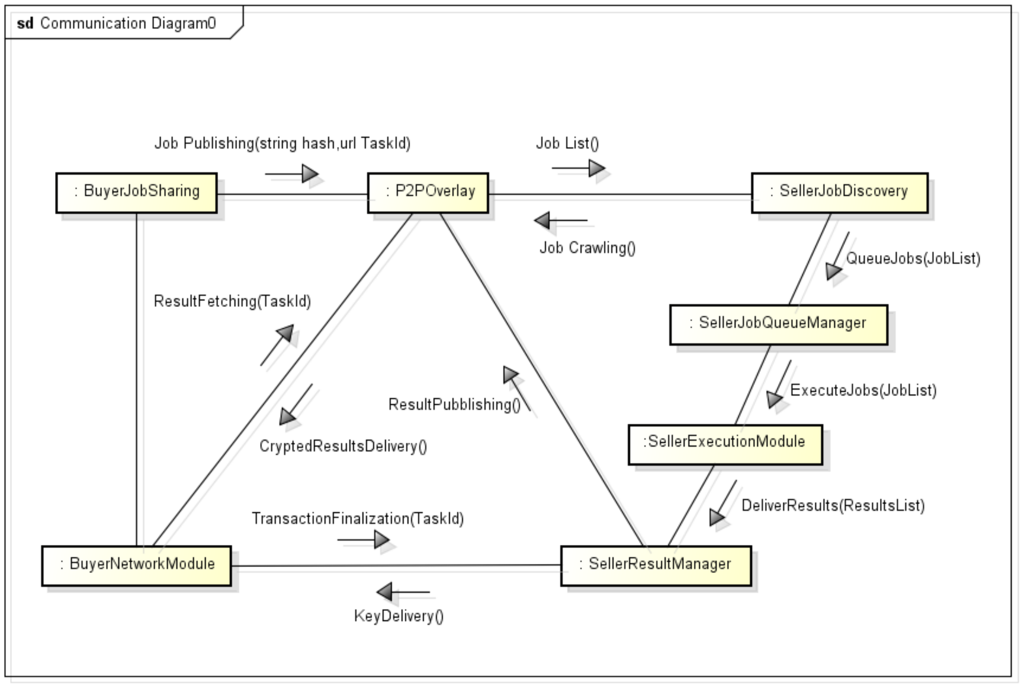


Figure 4. Testing platform: communication diagram

The results are compared with the values obtainable by running the reference job on a single dedicated server: we can model it as a scenario with reference mean execution time, and mean queue time equal to zero. Considering that a traditional PaaS environment allows the customer to autonomously manage the obtained server instance, the reference scenario correspond to what a user would expect by executing the same kind of tasks on a commercial PaaS cloud service (e.g., Amazon EC2 [21]).

For testing purposes, specific assumptions have been made: the first one is that the service providers have, globally, enough available resources to manage the overhead introduced by the competitive scheduling layer.

In other words, it means that, for the set job arrival rate, the global permanence time converges to a finite value. As long as this assumption is proved true, our tests show an improvement in mean system permanence times. Test set 3 describes a scenario where this assumption becomes untrue for one of the tested MAC values. The second assumption is that service providers consider cost-effective to commit resources to compete for job executions rather than keeping them idle.

A. Test set 1

The test has been performed with the following parameter settings:

- Job Arrival rate: 5 jobs per minute.
- Mean: 300 s.
- Standard deviation: 40 s.
- Number of peers: 300.
- Cool down: 120 s.

The test has been performed with two distinct MAC level values. As it is evident in Figure 5 and in Table III, in both cases, the mean permanence time of jobs in the system is lower than the mean execution time of the single job. This result is due to multiple peers competing to execute the job and deliver the results, so the actual execution time is definite by the peer that is the quickest one to perform the execution. Notice how an increase of the MAC value has lead to get a better mean of the system permanence times, despite the increasing queue time.

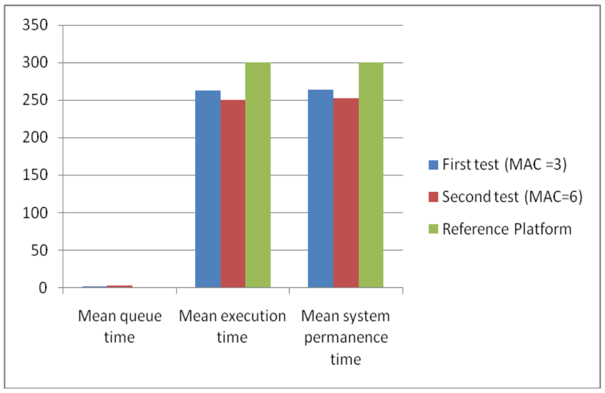


Figure 5. Performance results of test set 1

TABLE III. PERFORMANCE RESULTS OF TEST SET 1

	First test (MAC =3)	Second test (MAC=6)	Reference Platform
Mean queue time	1,339689655	2,348795559	0
Mean execution time	262,7362816	250,6696798	300
Mean system permanence time	264,2982312	253,0184754	300

B. *Test set 2*

The test has been performed with the following parameter settings:

- Job Arrival rate: 6 jobs per minute.
- Mean: 400 s.
- Standard deviation: 40 s.
- Number of peers: 2000.
- Cool down: 120 s.

As it has already been done in the previous scenario, this test has been performed with two distinct MAC level values. Figure 6 and Table IV summarize the results.

This test shows again, for both test cases, an improvement of the system permanence time compared to the baseline execution time. Notice, however, how the increased MAC value (test case 2) does not lead to better overall system permanence times: the increased queue time, due to having too many business peers fighting over each CFE and so less frequent network crawling.

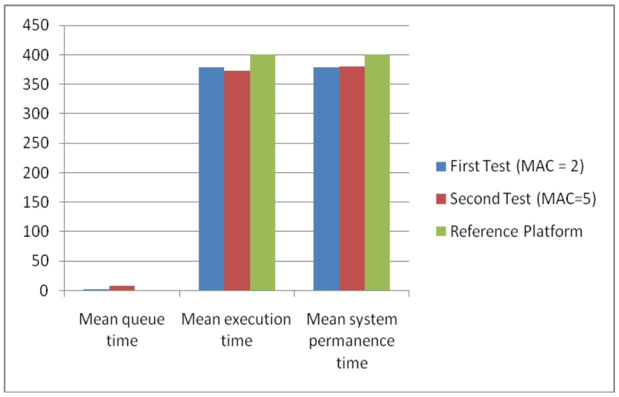


Figure 6. Performance results of test set 2

TABLE IV. PERFORMANCE RESULTS OF TEST SET 2

	First Test (MAC = 2)	Second Test (MAC=5)	Reference Platform
Mean queue time	0,878025466	7,532815631	0
Mean execution time	378,506824	372,9120812	400
Mean system permanence time	379,3848495	380,4448968	400

C. *Test set 3*

The test has been performed with the following parameter settings:

- Job Arrival rate: 6 jobs per minute.
- Mean: 600 s.
- Standard deviation: 40 s.
- Number of peers: 300.
- Cool down: 120 s.

Once again, this test has been performed with two different MAC level values. Results are summarized by Figure 7 and Table V. It is evident that, for a MAC = 6, the assumption of mean permanence time converging to a finite value is not proved: in this case, the reference platform would perform better than the system with the added competitive scheduling overlay.

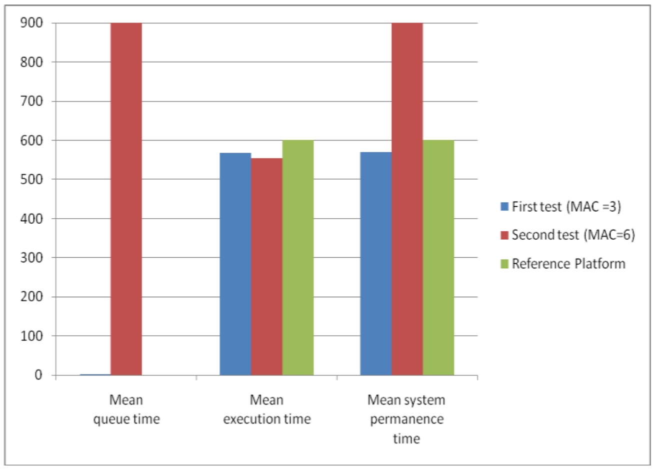


Figure 7. Performance results of test set 3

TABLE V. PERFORMANCE RESULTS OF TEST SET 3

	First test (MAC =3)	Second test (MAC=6)	Reference Platform
Mean queue time	2,2465506542	$\rightarrow +\infty$	0
Mean execution time	567,44207082	554,254621	600
Mean system permanence time	569,68862147	$\rightarrow +\infty$	600

VII. CONCLUSION

We presented a competitive approach for job scheduling in a P2P overlay of Cloud providers. Cloud technology is used for effective set-up of virtual resources which are compliant with application's requirements. The P2P overlay is used to publish and discover jobs' "calls for execution" and to overcome the complexity of negotiation mechanisms. Competition of providers is investigated to implement a business model where the cost is fixed by the users and providers try to respond and adapt.

We investigated the effectiveness of the proposed approach by implementing a framework that emulates the network protocols and the peers behaviors. The experimental activities validate our hypothesis that a competitive approach, in distributed scheduling environments, does not only decreases the threshold required to access the facilities, but it can also lead, if properly set up, to substantial performance gains. More specifically, this objective can be achieved by setting an appropriate level of competition between the infrastructure managers. A fine balancing must be pursued: too many competitors increase the concurrence over each submitted job. As a result, we notice a degradation of the system performances due to longer queue times.

VIII. ACKNOWLEDGMENT

This research is supported by the grant FP7-ICT- 2009-5-256910 (mOSAIC [22]).

IX. REFERENCES

[1] e-IRG, "White paper 2009," June 2009, http://www.e-irg.eu/images/stories/publ/white-papers/e-irg_white_paper_2009_final.pdf, (last accessed 18/8/2011).

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, n. 6, June 2009, pp. 599-616, doi:10.1.1.144.8397.

[3] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky, "Seti@home: massively distributed computing for Seti," Computing in Science and Engg., vol. 3, n. 1, 2001, pp. 78-83, doi:10.1109/5992.895191.

[4] A. Forestiero, C. Mastroianni, and M. Meo, "Self-Chord: a Bio-Inspired Algorithm for Structured P2P Systems," 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009), Shanghai, May 2009, pp. 44-51, doi:10.1109/CCGRID.2009.39.

[5] A. Forestiero, E. Leonardi, C. Mastroianni, and M. Meo, "Self-Chord: a Bio-Inspired P2P Framework for Self-Organizing Distributed Systems," IEEE/ACM Transactions on Networking, vol. 18, n. 5, October 2010, pp. 1651-1664, doi: 10.1109/TNET.2010.2046745.

[6] J. Linnolahti, "QoS routing for P2P networking," Helsinki University of Technology, Department of Computer Science, 2004, doi:10.1.1.58.7192, http://www.tml.tkk.fi/Studies/T-110.551/2004/papers/Linnolahti.pdf, (last accessed 18/8/2011).

[7] Y. Wang, L. Wang, and C. Hu, "A QoS Negotiation Protocol for Grid Workflow," Grid and Cooperative Computing (GCC 2006), Fifth International Conference, Dec. 2006, pp. 195-198, doi:10.1109/GCC.2006.14.

[8] G. Antoniu, M. Jan, and D. Noblet, "A practical example of convergence of P2P and grid computing: an evaluation of JXTAs communication performance on grid networking infrastructures," Proc. IEEE Symp. Parallel and Distributed Processing (IPDPS 2008), June 2008, pp. 1-8, doi:10.1109/IPDPS.2008.4536338.

[9] R. Buyya , D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in Grid computing," Concurrency Computat.: Pract. Exper., 2002, vol. 14, pp. 1507–1542, doi:10.1002/cpe.690.

[10] B. Cao, B. Li, and Q. Xia, "A Service-Oriented Qos-Assured and Multi-Agent Cloud Computing Architecture," CloudCom'09, LNCS, vol. 5931, Springer, 2009, pp. 644-649, doi:10.1007/978-3-642-10665-1_66.

[11] S. Venticinque, R. Aversa, B. Di Martino, and D. Petcu, "Agent based cloud provisioning and management: design and protoypal implementation," Proc. of Cloud Computing and Services Science (CLOSER), SciTePress, 2011, pp. 184-191.

[12] D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience," Concurrency and Computation: Practice and Experience, vol. 17, 2005, pp. 2-4, doi:10.1.1.6.3035.

[13] N. Drost, R. V. van Nieuwpoort, and H. Bal, "Simple Locality-Aware Co-allocation in Peer-to-Peer Supercomputing," Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRIDW'06), 2006, p. 14, doi:10.1.1.78.1535.

[14] I. Foster and A. Iamnitchi, "On death, taxes, and the convergence of peer-to-peer and grid computing," In 2nd International Workshop on Peer-to-Peer Systems (IPTPS03), 2003, pp. 118-128, doi:10.1.1.104.7210.

[15] A. Ghatpande, H. Nakazato, O. Beaumont, and H. Watanabe, "Analysis of divisible load scheduling with result collection on heterogeneous systems," IEICE Transactions, vol. 91-B, n. 7, 2008, pp. 2234-2243, doi: 10.1093/ietcom/e91-b.7.2234.

[16] I. Stoica, "Chord: a scalable peer-to-peer lookup protocol for internet applications," IEEE/ACM Transactions on Networking (TON), vol. 11, n. 1, Feb. 2003, pp. 17-32, doi:10.1109/TNET.2002.808407.

[17] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards Trusted Cloud Computing," USENIX, Proceedings of the 2009 conference on Hot topics in cloud computing (HotCloud'09), San Diego, CA, USA, 2009, doi:10.1.1.149.2162.

[18] Survey: "Cloud Computing 'No Hype', But Fear of Security and Control Slowing Adoption," July 2011, http://www.circleid.com/posts/20090226_cloud_computing_hype_security/, (last accessed 18/8/2011).

[19] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric," Revised Papers from the First International Workshop on Peer-to-Peer Systems (IPTPS'01), 2002, pp. 53-65, doi:10.1.1.18.6160.

[20] B. Penteado, "JKad: Java implementantion of the Kademlia Network," http://code.google.com/p/jkad/, (last accessed 18/8/2011).

[21] Amazon Elastic Compute Cloud (Amazon EC2), July 2011, http://aws.amazon.com/ec2/, (last accessed 18/8/2011).

[22] mOSAIC, July 2011, http://mosaic-cloud.eu/, (last accessed 18/8/2011).

[23] C. Gonzalo, "Peer-to-Peer (P2P) architecture: Definition, taxonomies, examples, and applicability," Internet Requests for Comment, RFC Editor, Fremont, CA, USA, Tech. Rep. 5694, Nov. 2009, http://www.rfc-editor.org/rfc/rfc5694.txt, (last accessed 18/8/2011).

[24] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, vol. 53, n. 6, 2009, p. 50.

[25] C. Kesselman and I. Foster, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann Publishers, Nov. 1998.

[26] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, and R. Katz, "Above the Clouds: A Berkeley View of Cloud Computing," Electrical Engineering and Computer Sciences University of California at Berkeley, Technical Report No. UCB/EECS-2009-28, Feb. 2009, doi:10.1.1.149.7163, http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf, (last accessed 18/8/2011).

[27] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking up data in P2P systems," Communications of the ACM, 2003, vol. 46, no. 2, pp. 43-48, doi:10.1145/606272.606299.