From Desktop Grid to Cloud Computing Based on BonjourGrid Middleware

Karim Hassan, Heithem Abbes, Mohamed Jemni
Research Laboratory LATICE
ESSTT/Université de Tunis,
5, Av. Taha Hussein, B.P. 56, Bab Mnara,Tunis,TUNISIA
karim.hassen@hotmail.fr, heithem.abbes@lipn.univ-paris13.fr, mohamed.jemni@fst.rnu.tn

***Abstract*—Desktop Grids provide computing and storage power, with a low economic cost, through the federation of free resources. Their performance relies strongly on the voluntary participation of users who make their machines available when these are unexploited. Several criteria such as number and volatility of resources make the execution of many applications in desktop grid, a great challenge. We attend today the emergence of a new concept: Cloud Computing. Similar to desktop grid, cloud computing provide resources for the execution of HPC (High Performance Computing) applications. In this context, we are interested in designing an approach, with an aim of having a hybrid execution's environment formed by BonjourGrid desktop grid middleware and a cloud computing, to overcome the constraint of lack of resources caused by the volatility of machines.**

Keywords—Desktop Grid; Cloud Computing; BonjourGrid

I. INTRODUCTION

Desktop grid is a hardware and software infrastructure aimed at exploiting the personal computers resources (processor, memory, disk space…). Several projects exploited the resources of desktop grid such as SETI@Home [1], the project of research of the extraterrestrial intelligence, and Climaprediction.net [2], a distributed computing project to produce predictions of the Earth's climate up to 2100 and to test the accuracy of climate models.

SungJin Choi et al. [3] present taxonomy of desktop grid: Desktop Grid is classified according to organization, platform, scale and resource type (see figure 1):

A. Organization

Desktop grid can be categorized in two categories according to the organization of the components of the grid: centralized and decentralized.

Centralized desktop grid such as BOINC [2] and XtremWeb [4], consists of three components: client, volunteer and server. A client is a parallel job submitter. A resource provider donates its computing resources during idle time. A server is a central manager that controls submitted jobs and volunteers and performs scheduling. Generally, a client submits a parallel job to a server. The job is divided into sub-jobs which have each own input data. The server allocates the sub-jobs to volunteers by using scheduling mechanisms. Each volunteer executes its task during idle time while continuously requesting data to the server. When each volunteer finishes the sub-job, it returns the result to the server. Finally, the server checks the correctness of the results and then returns the final result of the job back to the client.

Distributed desktop grid such as CCOF [5, 6, 7], Organic Grid [8, 9] and Messor [10, 11] consists of two components: client and volunteer. In contrast to centralized Desktop Grid, there is no server. Volunteers have the partial information of other volunteers. Volunteers are responsible for scheduling a job in a distributed way.

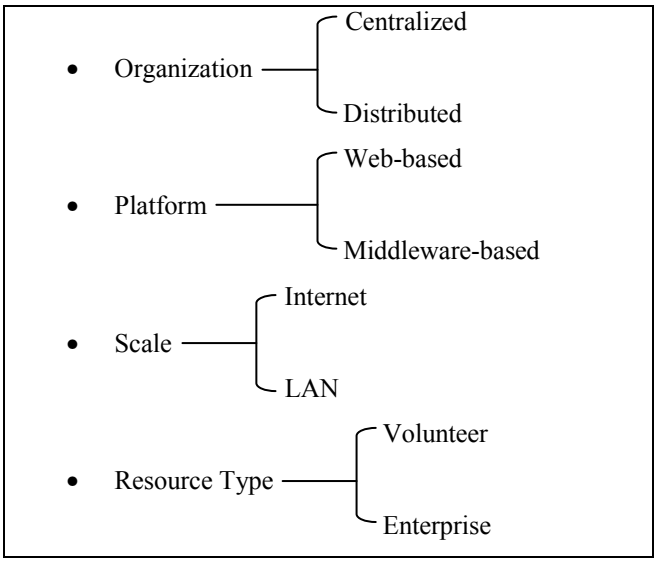

Figure 1. Taxonomy of Desktop Grid

B. Platform

Desktop Grid can be categorized in two categories according to platform running on volunteers: Web-based and Middleware-based.

In the Web-based desktop grid such as Charlotte [12], Bayanihan [13, 14, 15] and Javelin [16, 17], clients write their parallel applications in Java and post them as Applet on theWeb. Then, volunteers only join the web page with their browsers. The Applet are downloaded automatically and run on the resource provider's machine.

In the middleware-based desktop grid such as BOINC, XtremWeb, Korea@Home [18, 19, 20] and Entropia, volunteers need to install and run a specific middleware on the resource provider's machine.

C. Scale

Desktop grid is categorized into Internet-based and LAN-based ones according to scale. Internet-based desktop grid is based on anonymous resource providers. On the other hand, LAN-based desktop grid is based on resource providers within a corporation, institution, and university.

D. Resource type

Resource type specifies how resources are provided to the system. There are two main trends: volunteer and enterprise. Volunteer desktop grid is based on voluntary participants, while enterprise desktop grid is based on non-voluntary participants usually within a corporation, research lab or university. Mostly, volunteer desktop grid relies on Internet, while enterprise desktop grid is LAN-based. Volunteer desktop grid is more volatile, malicious, and faulty, whereas enterprise desktop grid is more controllable because its resource providers are located in the same administrative domain.

Due their inherent resource volatility it is difficult to exploit desktop grid for applications that require rapid turnaround.

We attend today the emergence of a new concept to execute HPC applications: Cloud Computing.
Today there is not yet a consensus for what exactly this term means. Examining some of the existing definitions helps to clarify the term and what it involves. Here we quote four definitions for cloud computing:

- "Cloud computing is a 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." - U.S. National Institute of Standards and Technology (NIST) [21].
- "A pool of abstracted, highly scalable, and managed compute infrastructure capable of hosting end customer applications and billed by consumption" - Forrester Research, Inc. [22].
- "A style of computing where massively scalable IT-enabled capabilities are delivered as a service to external customers using Internet technologies." - Gartner, Inc. [23].
- "A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers" - R. Buyya, C.S Yeo, and S.Venugopal [24]

Cloud computing can be classified by the model of service it offers into one of three different groups. These will be described using the XaaS taxonomy, first used by Scott Maxwell in 2006, where "X" is Software, Platform, or Infrastructure, and the final "S" is for Service (see figure 2).

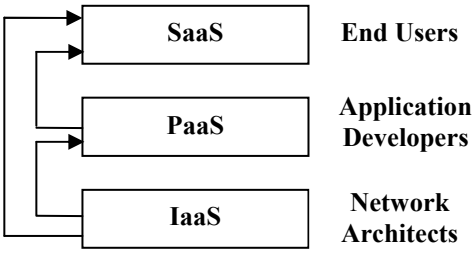

Figure 2. Cloud computing service models

It is important to note, as shown in Figure 2, that SaaS is built on PaaS, and the latter on IaaS. Hence, this is not an excluding approach to classification, but rather it concerns the level of the service provided. Each of these service models is described as follows:

- IaaS (Infrastructure as a Service): The capability provided to the customer of IaaS is raw storage space, computing, or network resources with which the customer can run and execute an operating system, applications, or any software that they choose. The cloud customer is not able to control the distribution of the software to a specific hardware platform or change parameters of the underlying infrastructure, but the customer can manage the software deployed.
- PaaS (Platform as a Service): In the case of PaaS, the cloud provider not only provides the hardware, but they also provide a toolkit and a number of supported programming languages to build higher level services (i.e. software applications that are made available as part of a specific platform). The users of PaaS are typically software developers who host their applications on the platform and provide these applications to the end-users.

- SaaS (Software as a Service): The SaaS customer is an end-user of complete applications running on a cloud infrastructure and offered on a platform on-demand. The applications are typically accessible through a thin client interface, such as a web browser. The customer does not control either the underlying infrastructure or platform, other than application parameters for specific user settings.

Clouds can also be classified based upon the underlying infrastructure deployment model as Public, Private or Hybrid clouds.

A public cloud's physical infrastructure is owned by a cloud service provider. Such a cloud runs applications from different customers who share this infrastructure and pay for their resource utilization on a utility computing basis.

A pure private cloud is built for the exclusive use of one customer, who owns and fully controls this cloud.

Finally, any composition of clouds, be they private or public, could form a hybrid cloud.

In this paper, we propose an approach to overcome the constraint of lack of desktop grid resources caused by the volatility of machines. This approach aims to make BonjourGrid desktop grid able to supply resources from a public cloud.

The rest of the paper is structured as follows. Section 2 illustrates an overview of some related works. Section 3 presents BonjourGrid desktop grid middleware. Section 4 presents our approach. Section 5 concludes the paper.

II. RELATED WORK

Buyya et al. [25] present integration between Aneka cloud platform and Amazon Elastic Compute Cloud (EC2) [26]. When the Aneka scheduling engine detects that the current capacity in terms of resources is not enough to satisfy the user's QoS requirement and to complete the application on time, an additional resources must be provisioned. It is the responsibility of the Aneka Resource Provisioning service to acquire these resources from the Amazon public cloud.

Yi et al. [27] illustrate integration between Aneka cloud platform and Windows Azure [28], which enables the usage of Windows Azure Compute Service as a resource provider of Aneka PaaS. The integration of Aneka with Windows Azure includes two different types. The first type is to deploy Aneka Worker Containers on Windows Azure while the Aneka Master Container is run on local. The second type is to deploy the entire Aneka PaaS including Aneka Master Container and Aneka Worker Containers on Windows Azure.

The third integration of Aneka is with GoGrid [29], it's similar to the first case (Aneka/Amazon EC2).

To make possible to write portable and interoperable code that works with multiple cloud providers, Apache develop Libcloud API [30]. Libcloud is a python library which allows users to manage their cloud resources (servers, storage, load-balancers) using a unified and easy to use interface. Actually Libcloud supports 16 providers such as: CloudSigma, GoGrid, OpenNebula, Openstack, OpSource and Eucalyptus.

Libcloud provides several functionalities:

- List_nodes(): returns a list of the currently active nodes
- Reboot_node(): allows user to restart a node.
- Create_node(): allows user to create a new node.
- Destroy_node() : allows user to destroy an existing node.
- List_images(): returns a list of available node images. An image represents an operating system which is installed on the server.

Libcloud supports these functionalities for all its providers, but Deploy_node() which allows user to run shell script on the node after it has been provisioned, is not supported for many cloud providers such as: Bluebox, Brightbox, CloudSigma, Dreamhost, VCL Cloud, CloudStack and OpenNebula.

Similar to Libcloud, Jclouds [31] is an open source java library that helps user to start in the cloud and reuse its java and clojure development skills. Jclouds supports many clouds including Amazon EC2, GoGrid, Windows Azure and Rachspace. Jclouds simplifies the tasks of managing machines in the cloud. For example, using Jclouds, the computeService can be used to both start a pool of machines in any of supported cloud and install software on them. Jclouds supports also the parallel execution of server commands such as create and execute scripts. For example, Neotys use this to launch hundreds of servers simultaneously.

These integrations are between clouds. Moreover, the approach we propose in this paper is between desktop grid and public cloud.

III. BONJOURGRID DESKTOP GRID MIDDLEWARE

BonjourGrid is an approach for the decentralization and the self organization of resources in desktop grid systems [32, 33].

The key idea is to exploit existing desktop grid middleware (Boinc, Condor, XtremWeb) and concurrently to manage multiple instances of desktop grid middlewares (Figure.3) [34, 35]. The notion of meta desktop grid middleware has been introduced with BonjourGrid and the Pub-Sub (Publish/Subscribe) paradigm is used intensively for the coordination of the different desktop grid middlewares [36].

Each user, behind a desktop machine in his office, can submit an application. BonjourGrid deploys a master (coordinator), locally on the user machine, and requests for participants (workers). Negotiations to select them should now take place. Using a publish/subscribe infrastructure, each machine publishes its state (idle, worker or master) when changes occur as well as information about its local load or its

use cost, in order to provide useful metrics for the selection of participants. These informations about the machine are interpreted as a resource which is published as a Web service. Under these assumptions, the master node can select a subset of worker nodes according to selection criteria. That aspect is managed as a Web service discovery.

The master and the set of selected workers build the Computing Element (CE) which will execute and manage the user application. That aspect is managed as a Web services composition. When the execution of an application of a CE terminates, its master becomes free, returns to the idle state and it releases all workers before returning to the idle state too. Then, the nodes can participate to others projects.

To implement this approach, BonjourGrid has been decomposed into three fundamental parts: a) A fully decentralized resources discovery layer, based on Bonjour protocol [37]; b) A CE, using a Desktop Grid middleware such as XtremWeb, Condor or Boinc, which executes and manages the various tasks of applications; c) A fully decentralized protocol of coordination between a) and b) to manage and control all resources, services and CEs.

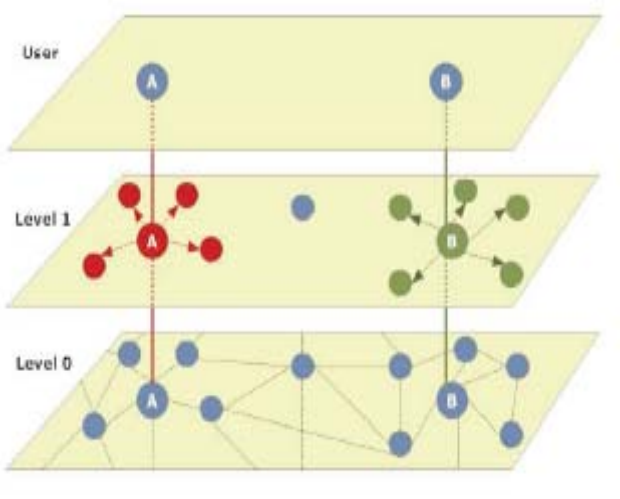

Figure 3: BonjourGrid abstract layers

As shown on Figure 3, in the user level, user A (resp. B) deploys his application on his machine and the execution seems to be local. Level 1 (middleware layer) shows that, actually, a CE with 4 (resp. 5) workers has been dynamically created, specifically for user A (resp. B). Level 0 shows that all machines are interconnected and under the availability of any user.

IV. CONTRIBUTION

BonjourGrid use participant's resources to execute user's applications. To overcome the constraint of lack of BonjourGrid resources caused by the volatility of machines, we propose a designing of an approach aims to provision resource from public cloud such as OpenNebula [38] and OpenStack [39] to satisfy the user's QoS requirement and to complete the application on time (Figure 4).

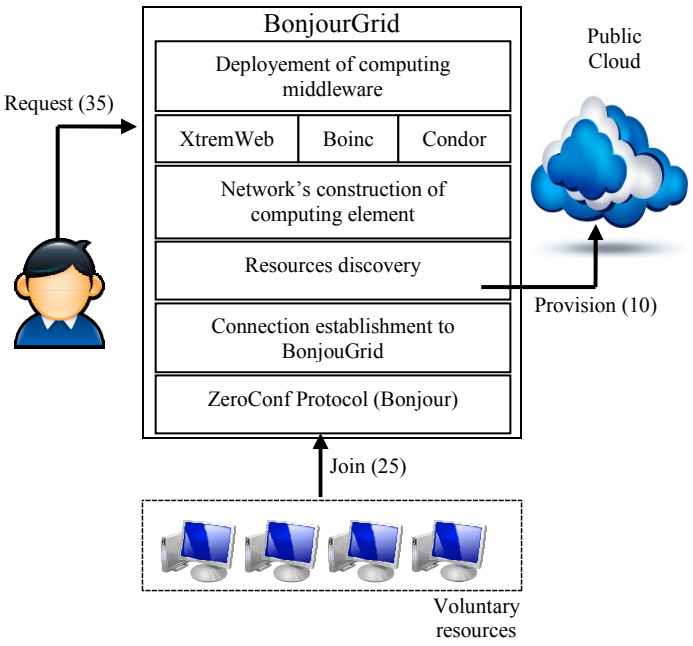

Figure 4. Use case of resources provisioning under BonjourGrid

As shown in Figure 4, once the client has submitted the application, the BonjourGrid middleware detects that the current capacity in terms of resources (25 nodes) is not enough to satisfy the user's QoS requirement and to complete the application on time. An additional 10 resources must be provisioned.

Our approach consists of two components: 1) a standard API that abstract away differences among multiple public cloud provider and BonjourGrid APIs. The use of this API, make the user's application, portable, which executes on BonjourGrid or any public cloud without any modification of the code, 2) a software layer in BonjourGrid which implements a resource provisioning service from a public cloud to overcome the lack of resources.

To make BonjourGrid able to provision resources from public cloud, the first constraint is to modify the discovery protocol. BonjourGrid is established on Pub/Sub paradigm which is an asynchronous mode for communicating between entities. Some users, namely subscribers or clients, express and record their interest under the form of subscriptions, and are notified later by another event produced by other users, namely producers. It is known that this asynchronous communicating mode allows spatial decoupling (the interacting entities do not know each other), and time decoupling (the interacting entities do not need to participate at the same time). This total decoupling between the production and the consumption of services increases the

scalability by eliminating many sorts of explicit dependencies between participating entities. Eliminating dependencies reduces the coordination needs and consequently the synchronizations between entities. These advantages make the communicating infrastructure well suited to the management of distributed systems and simplify the development of a middleware for the coordination of desktop grids.

To overcome the lack of resources from public clouds, our approach should provide a service for accounting. The large-scale, heterogeneous, decentralized and distributed nature of desktop grid environments places special requirements on the accounting service such as:

- Scalability: The accounting service must scale with an increasing number of users and resources.
- User Transparency: An ideal accounting service is completely transparent to users. I.e., the end user should neither notice the presence of the accounting service, nor need to be aware of its existence.
- Fault Tolerance: An accounting service should run smoothly even in the event of component failures.
- Trust and Security: Accounting information must only be exchanged between trusted entities. Thus, the accounting system needs to maintain some notion of trusted users, privileged users and their associated access rights.

V. CONCLUSION

In this paper, we have presented the characteristics of desktop grid systems and proposed an approach for provisioning resources from a public cloud to overcome the constraint of lack of workers in BonjourGrid desktop grid caused by the volatility of machines. This approach aims also to make hybrid execution environment formed by BonjourGrid desktop grid middleware and public cloud such as OpenNebula and OpenStack.

In the future we aim to implements the two components of our framework, providing advanced scheduling techniques for heterogeneous and we would like to extends the number of resource provider.

REFERENCES

[1] P. A. David, C. Jeff, K. Eric, L. Matt, and W. Dan. Seti@home :an experiment in public-resource computing. Communications of the ACM, 45(11) :56–61, 2002.

[2] P. A. David. Boinc : A system for public-resource computing and storage. In 5th IEEE/ACM International Workshop on Grid Computing, pages 4–10, 2004.

[3] C. SungJin, K. HongSoo, B. EunJoung, B. MaengSoon, K. SungSuk, P. ChanYeol, and H. ChongSun. Characterizing and classifying desktop grid. In CCGRID '07 : Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, pages 743–748, Washington, DC, USA, 2007. IEEE Computer Society.

[4] C. Franck, D. Samir, F. Gilles, H. Thomas, M. Frédéric, N. Vincent, and L. Oleg. Computing on large scale distributed systems : Xtremweb architecture, programming models, security, tests and convergence with grid. In Future Generation Computer Systems, volume 21 of issue 3, pages 417–437, 2005.

[5] V. Lo, D. Zhou, D. Zappala, Y. Liu, and S. Zhao, "Cluster Computingon the Fly: P2P Scheduling of Idle Cycles in the Internet," The 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04), LNCS 3279, pp.227-236, Feb. 2004

[6] S. Zhao and V. Lo, "Result Verification and Trust-based Scheduling in Open Peer-to-Peer Cycle Sharing Systems," Fifth IEEE International Conference on Peer-to-Peer Computing (P2P 2005), IEEE CS Press, pp. 31-38, Sept. 2005

[7] D. Zhou and V. Lo, "Wave Scheduler: Scheduling for Faster Turnaround Time in Peer-to-peer Desktop Grid Systems," 11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'05), LNCS 3834, pp. 194-218, Jun. 2005

[8] A.J. Chakravarti, G. Baumgartner, M. Lauria., "The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network," IEEE Transactions on Systems, Man, and Cybernetics, vol. 35, no. 3, pp. 1-12, May 2005.

[9] A.J. Chakravarti, G. Baumgartner, M. Lauria, "The Organic Grid: Self-Organizing Computational Biology on Desktop Grids," Chapter 27 in Parallel Computing for Bioinformatics and Computational Biology: Models, Enabling Technologies, and Case Studies, Wiley, 2006

[10] O. Babaoglu, H. Meling, A. Montresor, "Anthill: a framework for the development of agent-based peer-to-peer systems," 22nd International Conference on Distributed Computing Systems, pp. 15-22, Jul. 2002

[11] A. Montresor, H. Meling and O. Babaoglu, "Messor: Load-Balancing through a Swarm of Autonomous Agents," International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2002), LNAI 2530, pp. 125-137, Jul. 2003

[12] A. Baratloo, M. Karaul, Z. M. Kedem and P. Wijckoff, "Charlotte: Metacomputing on the Web," Future Generation Computer Systems, vol. 15, issues 5-6, pp. 559-570, Oct. 1999

[13] L. F. G. Sarmenta, S. Hirano. "Bayanihan: Building and Studying Volunteer Computing Systems Using Java", Future Generation Computer Systems, Special Issue on Metacomputing, vol. 15, no. 5/6., 1999

[14] L. F. G. Sarmenta, "Sabotage-Tolerance Mechanisms for Volunteer Computing Systems", Future Generation Computer Systems, vol. 18, issue 4, 2002.

[15] L. F. G. Sarmenta, "Volunteer computing," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Jun. 2001.

[16] M. O. Neary, S. P. Brydon, P. Kmiec, S. Rollins, and P. Cappello, "Javelin++: Scalability Issues in Global Computing", Concurrency: Parctice and Experience, pp. 727-735, Dec. 2000.

[17] M. O. Neary, P. Cappello, "Advanced eager scheduling for Javabased adaptive parallel computing," Concurrency and Computation: Practice and Experience, vol. 17, issue 7-8, pp. 797-819, Jun. 2005

[18] Korea@Home, http://www.koreaathome.org/eng/

[19] S.J. Choi, M.S. Baik, J.M. Gil, S.Y. Jung, and C.S. Hwang, "Adaptive Group Scheduling Mechanim using Mobile Agents in Peer-to-Peer Grid Computing Environment", Applied Intelligence, Special Issue on Agent-based Grid Computing , vol. 25, no. 2, pp. 199-221, Oct. 2006

[20] S.J. Choi, M.S. Baik, J.M. Gil, C.Y. Park, S.Y. Jung, and C.S. Hwang, "Group-based Dynamic Computational Replication Mechanism in Peer to Peer Grid Computing", IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2006), Sixth International Workshop on Global and Peer to Peer Computing (GP2P), IEEE CS Press, May 2006.

[21] P. Mell and T. Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 2009.

[22] J. Staten. Is cloud computing ready for the enterprise? *Forrester Research, March*, 7, 2008.

[23] D. C. Plummer, T. J. Bittman, T. Austin, D. Clearley, and D. M. Smith. Cloud computing: Defining and describing and emerging phenomenon, Gartner, Inc. *Retrieved September*, 25:2008, 2008.

[24] R. Buyya, C.S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as

computing utilities. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 5–13. IEEE, 2008.

[25] C. Vecchiola, X. Chu, M. Mattess, and R. Buyya, Aneka - Integration of Private and Public Clouds, Cloud Computing: Principles and Paradigms, 249-274pp, R. Buyya, J. Broberg, A.Goscinski (eds), ISBN-13: 978-0470887998, Wiley Press, New York, USA, February 2011.

[26] Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/

[27] W. Yi, S. Karthik, V. Christian, K. Dileban and R. Buyya, Aneka Cloud Application Platform and Its Integration with Windows Azure, Cloud Computing: Methodology, Systems, and Applications, L. Wang, R. Ranjan, J. Chen, and B. Benatallah (eds), ISBN: 9781439856413, CRC Press, Boca Raton, FL, USA, October 2011.

[28] L. Henry, *Introducing Windows Azure*, ISBN: 978-1-4302-2469-3, Apress, 2009.

[29] GoGrid, http://www.gogrid.com

[30] Libcloud, http://libcloud.apache.org/

[31] Jclouds, http://www.jclouds.org/

[32] A.Heithem, C. Christophe, and J. Mohamed. Bonjourgrid as a decentralised job scheduler. In APSCC'08: Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference, pages 89–94,Washington, DC, USA, 2008. IEEE Computer Society.

[33] A. Heithem, C. Christophe, J. Mohamed: A decentralized and fault-tolerant Desktop Grid system for distributed applications. Concurrency and Computation: Practice and Experience (2010), 22(3):261-277

[34] A. Heithem, C. Christophe, J. Mohamed, Bonjourgrid: Orchestration of multi-instances of grid middlewares on institutional desktop grids. Parallel and Distributed Processing Symposium, International, 0:1–8, 2009.

[35] S. Walid, A. Heithem, C. Christophe, J. Mohamed., A Self-Configurable Desktop Grid System On-Demand, 3PGCIC 2012, November 12-14 2012, Victori, Canada.

[36] A. Heithem, C. Christophe, J. Mohamed, and S. Walid, Toward a meta-grid middleware. Journal of Internet Technology, Volume 11 No1, 2010.

[37] H .S. Daniel, C. Stuart, Zero Configuration Networking: The Definitive Guide, Publisher: O'Reilly Media, Released: December (2005) 256

[38] OpenNebula http://www.opennebula.org

[39] OpenStack http://www.openstack.org