A New Approach to Manage and Utilize Cloud Computing Underused Resources

Alaa Eldeen S Ahmed
Computer Engineering Dept
Benha University
Shoubra Cairo-Egypt

Abdulwahab K. Alsammak
Computer Engineering Dept
Benha University
Shoubra Cairo-Egypt

Essam Algizawy
Computer Engineering Dept
Benha University
Shoubra Cairo-Egypt

ABSTRACT

In this paper a new approach for driving a better cloud computing IaaS Services is presented. This approach focuses on extending the available cloud computing platform infrastructure by harvesting underused generic computing resources that are widely available within public domains such as universities and organizations. Most of the current cloud computing platforms are mainly based on using a dedicating infrastructure to achieve the requested services. The proposed approach aims to improve the computing power for the cloud computing platform without charging any extra cost since generic machines are owned by the enterprise. A new Resources management mechanism is introduced to manage the combination of the dedicated and generic machines. In order to implement and achieve the goals of the proposed approach several challenges should be conquered. These challenges are coming from the inherently stochastic characteristics of the harvested unused computing such as reliability and hardware compatibility. In the implementation, Openstack cloud computing platform is used and extended in such a way that guarantees QoS and an opportunistic use of the idle or underused generic or public computing resources.

General Terms

Cloud Computing, Resource Allocation, Open Stack Platform

Keywords

Cloud Computing, IaaS, Cycle-Harvesting, Resource Management, Generic Computing.

1. INTRODUCTION

Cloud computing changes our mind and the way of thinking about what is considered to be ”Our Data”or ”Our System”. They are no longer physically stored on a set of computers and disks, both have been shifted to the cloud which is diffused and distributed [1]. In this paradigm, cloud computing services are delivered in an elastic, service-oriented and on-demand pay-as-you-go manner to consumers. According to McKinsey, with cloud computing commercial enterprises can achieve saving potential of 30% to 40% with needs orientated [2].

Although the cloud computing has many deployment models, the private cloud is the most secure one. In private clouds data and processes are managed within organization boundaries without any restriction on the network bandwidth, security exposures, or legal requirement [3]. Most of these clouds are built on a well provisioning and well managed infrastructure like OpenStack [4], Nimbus [5], Eucalyptus [6], OpenNebula [7], and other open source IaaS/PaaS cloud computing software platforms. Private cloud’s elasticity is limited depending on the subjected organization’s maximum dedicated hardware capacity.

Also it is bounded to fall short of demand, and adopting all of infrastructure demands on a private cloud computing will waste organization’s infrastructure investments. Running idle generic machines as a part of the cloud offer an opportunity to resolve such increasing need for cloud computational resources, enable us to avoid the cost impact of the over-provisioning or under-provisioning of the private cloud dedicated resources. Also it reduces the need for specialized infrastructure for resilience, such as redundant power and cooling systems, battery backup, etc, which represents 25% of data centre costs [8]. As well as it reduces overall power consumption through the reduction in the total number of machines since the energy cost of manufacture for a computer has been estimated as four times that used during its lifetime.

This paper is organized as follows; section II presents the volunteer cloud computing related works. The proposed Architecture is described in section III. The extended OpenStack cloud computing software platform is presented in section IV. Section VI describes the architecture implementation results. Finally, section VII presents the conclusion.

2. RELATED WORK

Volunteer computing provides processing power up to teraflops scale for scientific and technical projects. So there are many projects and methodologies that aim to obtain the benefits of cloud computing using volunteer machines. These projects started with ad-hoc cloud that outlines the major implementation challenges, and how under-use computing resources within enterprises may be harnessed [9]. A typical example is the ongoing project Cloud@Home [10] aims to provide a cloud infrastructure that is capable of providing and sharing resources and services for the scientific purposes. The volunteer computing is behind the “@Home” philosophy of sharing / donating resources [11][12]. The basic Cloud@Home architecture is organized in three hierarchical layers: frontend, virtual, and physical layer. The physical layer is composed of cloud generic nodes geographically distributed across the internet. Cloud@Home negotiate with users that want to join the cloud about their contributions, and monitoring such constrains, requirements, and policies in order not to be violated [13].

Another example BoincVM aims to provide cross-platform distributed computing platform. The key development in BoincVM is to adapt BOINC [14][15] to support virtualization which has been successfully pioneered at CERN. In 2008 CERN R&D fire a project called CernVM, it offered a general solution to the problem of VM images management. CernVM offers a minimum VM appliance - Less than 250 MB in size- then that image download all of its libraries by itself. In 2011, the first BOINC project based on VM was porn in the LHC@home Test4Theory project. This is done based on CernVM and job management framework (Co-pilot). Co-Pilot is CernVM's cloud interface [16].

In [17] B. Sodhi proposed an architecture which serves a user of the real-time system-state information from a decentralized cluster of nodes. The most important components of this architecture are the cloud controller, cloud agent, and VM foundry. Cloud controller represents the gateway between the cloud and the client. Its job is to determine which suitable node can run the required VM to satisfy the client requests. The cloud agent responses to the cloud controller's queries about the worker nodes, and configure VM to run within the cloud on the selected nodes. Finally the VM foundry which acts as images repository.

The work presented in [18], provides grid/cloud services at low cost through the opportunistic use of idle computing resources available in a university campus. UnaCloud architecture is divided into two main components: UnaCloud server and UnaCloud client. The main function of UnaCloud server is to provide an entry to all services like service deployment, access, management and monitoring. The UnaCloud client installed and run directly on the underlying opportunistic infrastructure. It is responsible for performing all functions to provide a dynamic and on demand IaaS model which makes it complex.

Sheng Di, 2011 introduces a design for a Self-organized Cloud (SoC). SoC tackles two main issues; first makes a full use of widely dispersed idle resources to construct a win- win situation. Second guarantee the overall performance of the system [19].

Table I shows a comparison between the well-known @home cloud methodologies. There already exist many other projects such as EduCloud@Home [20], MOON cloud [21], Wuala cloud [22], Nebulas [23], symbiotic computing [24] etc. All of these research projects generalize the concept of volunteer computing to provide cloud services over it with different ways. The main drawbacks of using distributed volunteer resources for building the @Home cloud computing platform results from the stochastic nature of the public resources and limitation on the capability of the individual underutilized resources.

3. PROPOSED ARCHITECTURE

The IaaS provisioning involves dynamics creation of an infrastructure consisting of different types of computing re-sources with necessary control and management planes. In the proposed approach we modify and extend this model which is built using the OpenStack cloud computing platform. OpenStack seems to be on its way to be the open-source software cloud stack of choice. Openstack is extend to allow the ability of harvesting idle computing resources donated by the public to join and leave the cloud and hence improve the provided IaaS services [25].

Figure 1 shows the abstract view of the proposed model. This model is considered as a modified version for the standard IaaS/PaaS cloud computing platform. It consists of three functional layers which are defined in the following subsections:

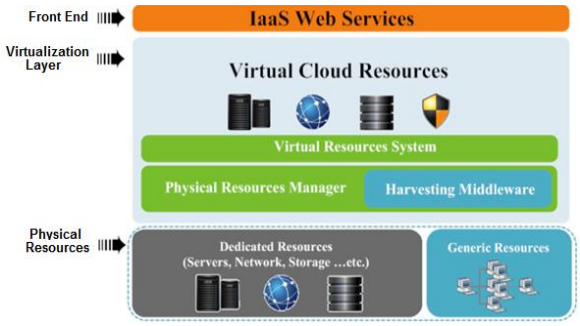

Fig. 1: The Proposed Approach Abstracted Architecture

Elements of each layer communicate and cooperate with each other to perform the required tasks. Each of these layers provides services for the above layer, each one knows only the layer that communicates with and knows nothing about other layers. Figure 1 illustrates the separation of concerns of the architecture layers stated above that can be described as the following:

- Front-end Layer: This layer represents IaaS services, where the major goal is to facilitate service related data handling and user interaction. It is also responsible for managing all resources and services. It makes commands and controls the virtualization hypervisor. Implementing this layer also provides QoS, facilitates business models management, and provides SLA polices as well.
- Virtualization Layer: This layer is the key element of our approach. It provides homogenous view of heterogeneous environment provided by the physical infrastructure layer. This homogeneity is absorbed by the higher front-end layer in the form of a set of services. Each service is provisioned and delivered using virtual machines.

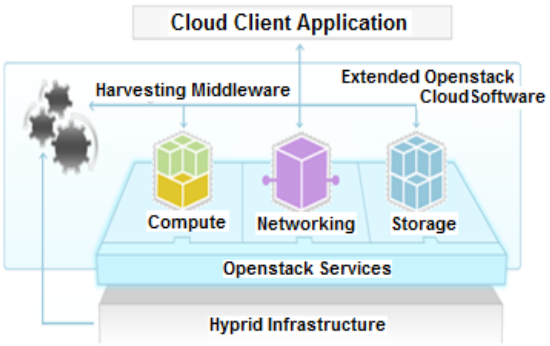

Fig. 2: The Extended OpenStack Cloud Software

- Physical Infrastructure Layer: This layer provides a possibility to handle all incoming requests and locally running software. This layer combines two types of re-sources; the cloud computing dedicated resources which ensure QoS, and the volunteer computing resources de-livered by the harvesting middleware. This mix of generic and dedicated resources is managed and organized using the higher virtualization layer physical resources manager module.
- Each time a request for a new cloud computing service is submitted, the cloud controller checks-up the available resources such as cores, memory, and storage to deploy

such VM. If there is a provisioning for deploying such virtual machine, the cloud controller will process the request by deploying the requested service. But if the virtual machine can't be deployed on the current cloud infrastructure due to lack of resources, the cloud controller will ask the harvesting middleware to free up cloud dedicated resources as shown in figure 2.

3.1 Harvesting Middleware Architecture

Harvesting middleware is the central part of the proposed approach. It takes care of volunteer machines allocation and its management. Also it is responsible for managing these machines that join the cloud and resume cloud services of the leaving volunteer machines. It consists of two main parts:

- Server-side harvesting middleware which takes care of managing resources and resolving problems arise from using volunteer machines.
- Light client side Resources Allocation Agent which provides a tool for services management on volunteer machines, and cloud interaction.

The harvesting middleware consists of five daemons. Each one of these daemons plays a vital role in running any cloud service on a desktop machine to be a part of the cloud computing infrastructure. The five daemons are Resources Index Information Service (RIIS), Cloud Resources and Service, Local Resources Manager (LRM), Virtual Machine Manager (VMM), and Services Listener (SL). Figure3 shows the harvesting middleware architecture components:

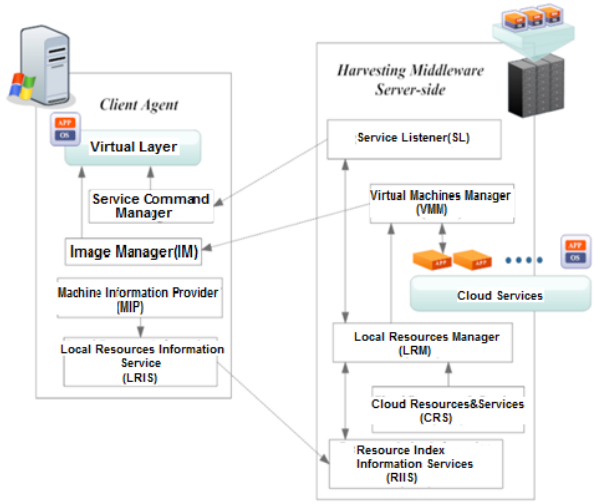

Fig. 3: Harvesting Middleware Architecture

1. Resources index information service (RIIS): RIIS represents a lightweight directory of the attached desktop machines, containing the basic information about local desktop resources and its aliveness status. This is done through a periodically heart beat message received from each registered desktop machine. RIIS collects information from several LRIS to allow searching the information to find the most suitable resources. The desktop machine local resources information offered by the RIIS includes cores, memory, load status, desk storage and cloud service (Virtual machine) running on it if any.
2. Cloud Resources and Service (CRS): CRS contains a complete list of Cloud platform resources e.g. computing node, network nodes, and storage. For each computing node there is a list of different running services associated with service information. This list may include, instance name, memory, virtualization type, kernel, desk source, interface type, service IP, DHCP server and all other information contained with a libvirt XML generated files.
3. Local Resources manager (LRM): LRM is responsible for managing the execution of any cloud computing services on a desktop machine. It keeps track of all cloud computing services running on desktop machines, it also periodically query the status desktop running tasks from RIIS. If the Cloud Controller requests the LRM to free up certain cloud resources, the LRM will select the most suitable desktop machine from information provided by RIIS and CRS. Finally, it takes a snapshot of the running virtual machine to be deployed on the selected desktop machine.
4. Virtual machine manager (VMM): VMM works in conjunction with the LRM to manage virtual machines in both the cloud environment and the desktop machines. It is responsible for creating image snapshots, suspending and starting running cloud services. Also, it can issue a deployment commands for starting , stopping , and monitoring cloud computing virtual machines on desktop PCs
5. Services Listener (SL): All users requests (Start, Terminate, Snapshot etc.) are submitted to the cloud controller with the associated service identification. SL listens to all requests passed to the cloud controller and redirects basic requests to (LRM registered / Migrated) cloud service running on desktop machines instead of passing it to either the compute nodes running those services or these that have a suspended image to guarantee service availability in case of desktop service failure.

3.2 Light Client Side Resources Allocation Agent

Light Client Side Resources Allocation Agent is a lightweight, highly portable and easy to install which is installed and run directly on a volunteer computer. It consists of four modules:

- Both of the Machine Information Providers (MIP) and Local Resources Information Service (LRIS) represent the resources discovery system which is responsible for publishing and queuing the state of resources and their configurations. Upon installing the RAA agent on any desktop machine the MIP starts capturing information about the local resources which in turn passed to LRIS which represents the interface for RIIS on each desktop machine.
- Image Manager (IM) is responsible for issuing simple virtualization commands for starting, terminating or monitoring new virtual machine. Also, it manages desktop virtual machine networking.
- The last module is the service command manager. It is used to execute basic requests redirect from SL daemon on the running desktop cloud services

4. EXTENDED OPENSTACK CLOUD PLATFORM

Extended OpenStack IaaS Lifecycle is shown in figure 4.This figure presents different steps occurred through the IaaS lifecycle if there are no spare dedicated cloud resources for provisioning a new cloud service:

The proposed OpenStack extension to allow private cloud's elasticity not to be limited to the cloud dedicated machines. This extension reflects the architecture explained in the previous section that allow cloud computing platforms to add

the idle desktop machines when there is a need to free up some of the dedicated cloud components to improve the cloud platform performance which allow new services provisioning. The OpenStack's nova implementation has two essential components; Messaging Queue and Database. These components facilitate the asynchronous system orchestration of complex tasks through message passing and information sharing.

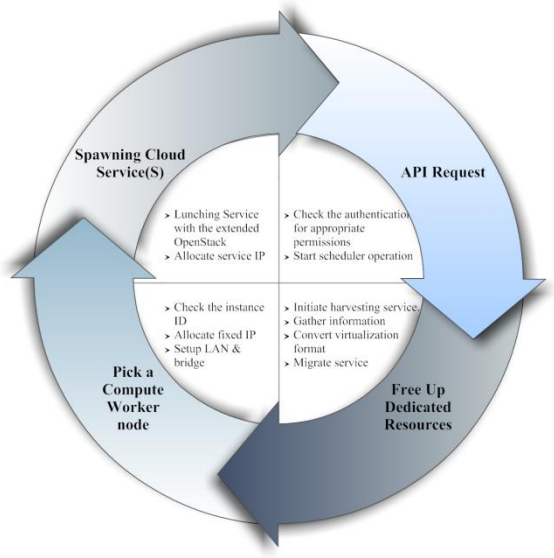

Fig. 4: Extended OpenStack IaaS Lifecycle

4.1 Message passing Extension

The complete message passing to free up a worker node through the using of the extended OpenStack cloud computing platform is shown in figure 5.

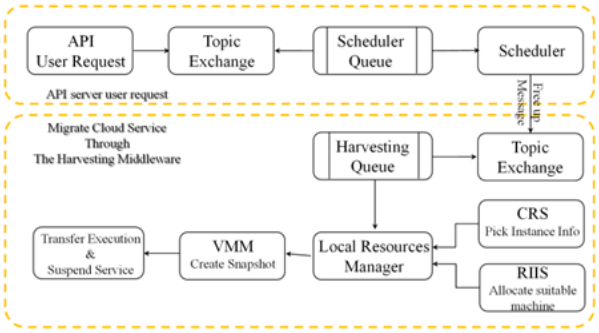

Fig. 5: Message passing for free up worker node task

It starts with a message from the API publisher to the scheduler topic exchange. The worker scheduler retrieves the message with a free up arguments from the scheduler queue. The scheduler sends rpc.cast message to the harvest topic to free up a picked worker node. This message causes the harvesting middleware LRM to initiate a service migration task. The LRM pick a random instance metadata from CRS, and allocate a suitable desktop machine from this information provided by RIIS and issue a VMM create-snapshot task with metadata about the pick instance. The information about volunteer resources passed on by the agent MIP to the LRIS interface. The RIIS collects information from multiple LRIS, to enable resources searching to through this information to find a suitable desktop machine for service migration. VMM create-snapshot removes the loop mapping by creating and compiling a new boot.cmd file to remove the cloud cloud-init information. Also it adds a multi-boot compliant floppy image, and combines cloud client applications and the base system into a single VM after converting it to RAW format. Finally transfer the execution of the created image to the allocated volunteer machine and suspend the cloud service. After migrating the cloud's service to the allocated volunteer machine, The LRM sends a message for the scheduler with the compute worker host information where a new service can be provisioned. To provision a new service, a fixed IP is needed so the computer worker sends a message to network controller to allocate such IP from IPTable and continues with spawning of this instance.

Allocating fixed IP address or service address accomplish by standard RPC call to the network controller. This call targeted a specific host using a topic host exchange and expecting a response from the called machine. The instance spawning process is performed by the libvirt virtualization interface driver. Libvirt stores all information about the spawned instance in a libvirt.xml file using to-xml to store and retrieve metadata. The libvirt.xml file placed within the instance folder. Figure 6 shows the processes of allocating a fixed address for a new instance.

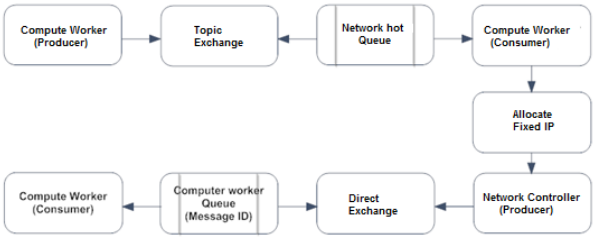

Fig. 6: New service platform service spawning

4.2 Database Modifications

The SQL database stores most of the build-time and run- time state for OpenStack's nova cloud computing infrastructure. But it does not support the bridging to the volunteer computers, so a few modifications and additions are needed. The first thing needed is a way to record registered volunteer computers information. In standard OpenStack case only the dedicated resources compute nodes are stored. So an extra field added to indicate the generic desktop machines. An additional networking table is needed referencing the registered generic machine host information. The networking table used for migrated services requests routing; besides services fixed IPs. Regular cloud computing service cannot be run on generic desktop machines, so a Volunteer-VMs table is added. It contains the picked up services snapshots and their corresponding path to compute nodes. This table has been used to simplify the process of service migration to volunteer desktop machines. It contains a path for converted cloud service to suitable desktop VM format and all other information about these VMs. So to pick these completed snapshots, both of the instances table, and the instances metadata table should be viewed. Finally keep track the migrated instances to be able to record the source compute nodes information, the destination generic desktop machine host, the service status, and the snapshot running version. The source compute node is used to ensure the service availability, in case that any service status becomes down. Once the service becomes down VMM will be ordered to resume the latest running snapshot version stored.

5. CHALLENGES

Driving this approach for better cloud computing IaaS services based on the opportunistic use of generic resources is a significant challenge. This challenge due to the characteristics of the harvested resources are inherently stochastic, as well as cloud computing client always expects to obtain optimized services against QoS definitions provided based on SLA. The main challenges are discussed here

- Cloud computing and generic resources computing are a very different approach in distributed computing. Using generic resources as a part of a cloud computing platform require familiarizing ourselves with the concepts of the cloud computing and generic resources in particular.
- Cloud computing resources are generally reliable whereas the generic resources are not. So it is important to figure out how to deal with a running service meant on reliable resources on unreliable resources.
- Resources and service management mechanism - a mechanism for managing dedicated and generic resources are needed. This mechanism must provide a high level of abstraction and service oriented for different users through a unified interface. This interface should provide a unique and uniform access point to our system, which overcomes the problem of hardware compatibility. It must allow users to submit functional and nonfunctional requests without knowing the system resources.
- QoS - The QoS can be defined in the term of service availability based on the provider Service Level Agreement (SLA). A problem to face at this level is the reliability of services as the volunteer resources are stochastic in nature.
- Security - An effective mechanism is required to provide identity management and data protection.
- Clouds Interoperability - Our solution must be interoperating with other cloud computing provider's platforms.

To successfully run generic resources as a part of a cloud computing platform, The OpenStack cloud computing platform is adapted to take advantage of the harvested idle generic resources. The adaptation of the OpenStack platform will ensure the availability of cloud computing client's services and the QoS of these services.

6. ARCHITECTURE IMPLEMENTATION RESULTS

Evaluating the proposed approach is performed on two levels. The first level is monitoring the cloud computing loads before and after adding the harvesting middleware. These loads are generated using the Stress Work load generator bench mark [26]. The generated loads are gradually increased until reaching the full utilization of the migrated services. The performance of these services is monitored with each load. Not only cloud service loads that affect the cloud QoS but also new services creation can affect it. The second level is monitoring the migrated cloud service QoS on volunteer desktop machines with different loads. These loads enable investigating their effects on cloud computing nodes. It also monitors the effects of using non-dedicated infrastructure as a part of the cloud computing platform on the migrated service QoS. By applying this scenario, The following effects are recorded:

6.1 Effect on Cloud compute Nodes

1- Cloud Computing Services Creation Effects: From the client point of view, providing IaaS is no more than creating a set of VMs by selecting the appropriate OS system required for the cloud client from those offered by the cloud provider. Upon creating service, the client will have a VM with specific hardware specifications running the chosen OS like any physical machine. IaaS Services creation affects the performance of other services running on this host. These effects shown in figure 7:

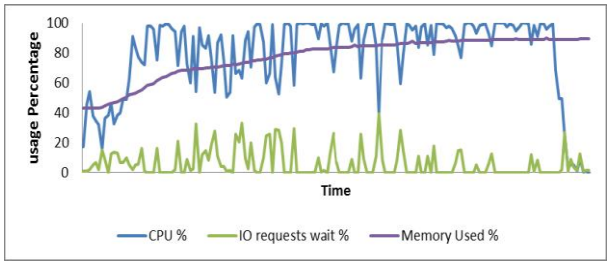

Fig. 7: Cloud services creation usage

2- Compute Node Performance without Loading Services: Upon monitoring different system parameters after IaaS ser- vices creation, about 90% of the computing node memory still used or reserved even though these services didn't use these resources in any useful work. The reason behind is that each cloud service locked its own memory, which made it so difficult to create another service on the same machine as shown in figure 8.

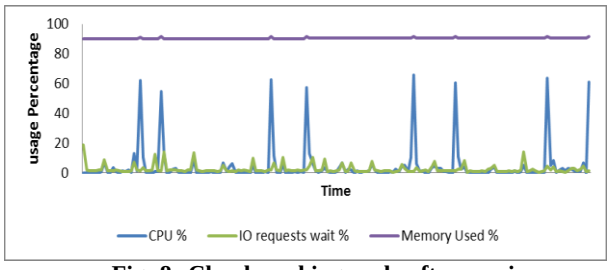

Fig. 8: Cloud working node after service creation without loading a cloud service

3- Half Loading of a Compute Node Services: Upon putting the services under half load stress, the memory reach its maximum values and the CPU reach 90% at some points. However the system is fully utilized, the full load services are not reached yet as shown in figure 9.

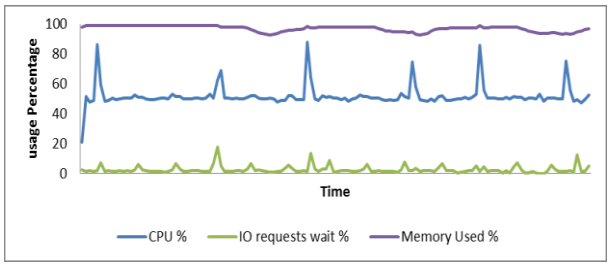

Fig. 9: Loading half of a compute node running services

4- Full Loading of a Compute Node: Now, on full loading the cloud services with a cloud client demand pattern is

shown in figure 10, which defines the actual user CPU, Memory and IO requests to a Cloud provider. This causes the computing node memory and CPU reaching its maximum values which enforces the system to reset a lot of tasks as shown in figure11. Also IO requests will wait for a significant amount of time. All these bad effects degrade the system throughput for the requested services.

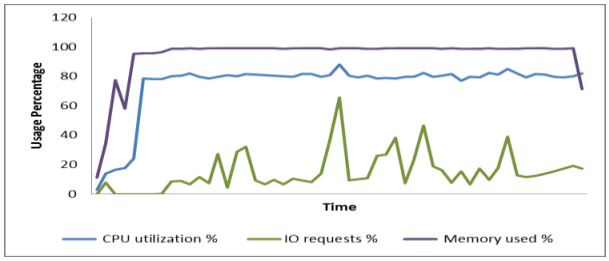

Fig. 10: Cloud client demand pattern

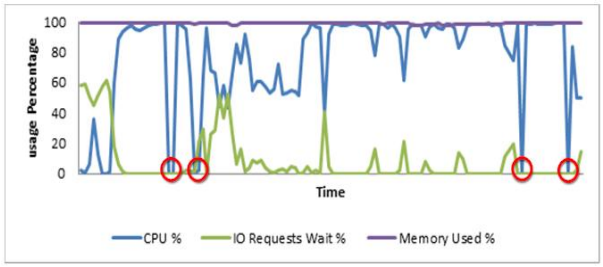

Fig. 11: Loading all of the cloud services

5- Moving Part of a Compute Node Services to Desktop Machines: on applying our proposed solution and moving 40% of a cloud computes services to suitable desktop (Windows / Linux) machines with the same demand pattern shown in figure 10. This movement is partially improving the performance of a compute node due to decreasing its load which enables it to perform properly with the remaining part of the cloud services.

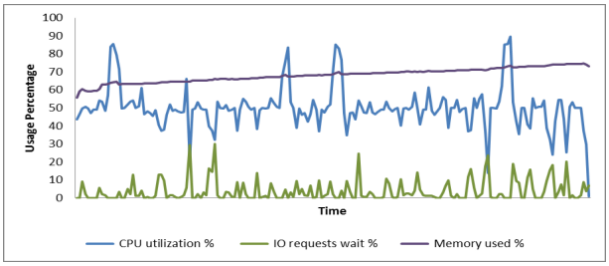

Fig. 12: Moving 40% of a compute node services

6- Moving all Services to Desktop Machines: Moving all services of a compute node to suitable desktop (Windows / Linux) machines with the same load described in the demand pattern figure 10. The compute node performance is significantly enhanced as shown figure 13. Displacement of these services to desktop machines makes the compute nodes ready to receive any further cloud services or offloading any other compute node in the Cloud Computing platform.

6.2 Effect on Volunteer machines (Windows/Linux)

1- Desktop Machine Regular Usage: In normal cases there is a consumption of a small part of the desktop machine computing power capabilities, as shown in the following figure 14 and figure 15. As shown in these figures, over than 50% of machine memory is free to be used. Also, 80 % of CPU utilization is free to be used. This indicates the good performance achieved when harvesting any available non-dedicated machines.

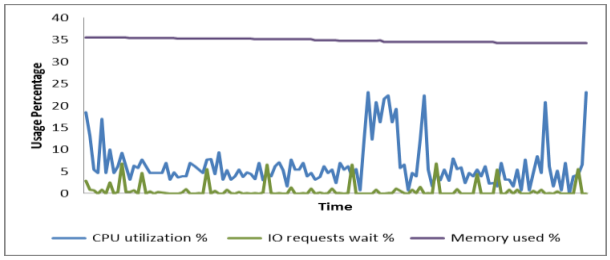

Fig. 14: Regular windows machine performance

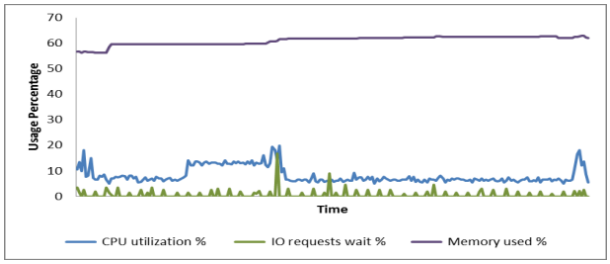

Fig. 15: Regular Linux machine performance

2- Loading Cloud Service on a Desktop Machine: To measure the performance effect on loading the desktop machine, IaaS service is run on the desktop (windows / Linux) machine. Both of the desktop and cloud service are not loaded yet with any cloud client requests. So the performance is still good in both memory and the processor (memory is about 55% underutilized) as shown in figure 16 and figure 17.

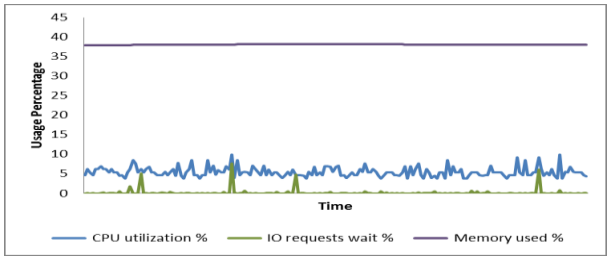

Fig. 16: Run Cloud Service on windows machine (Both of the desktop and service are not loaded)

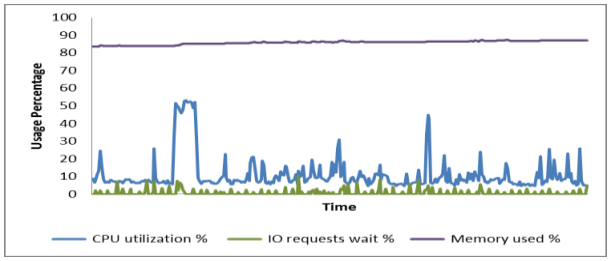

Fig. 17: Run Cloud Service on Linux machine (Both of the Client and service not loaded)

Secondly, loading a migrated cloud service with the same load shown in figure 10, the system is still working properly as shown in figure 18 and figure 19

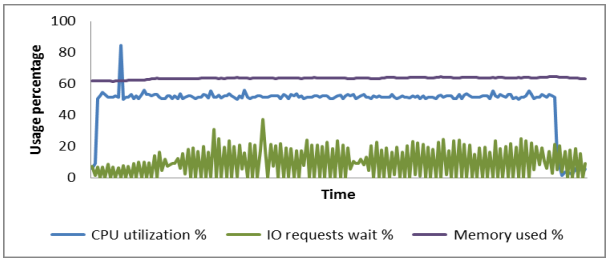

Fig. 18: Load Cloud service on windows machine

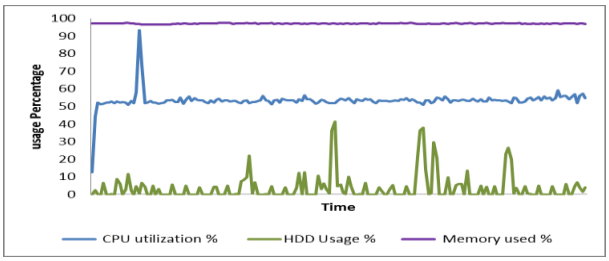

Fig. 19: Load Cloud service on Linux machine

7. CONCLUSION

In this paper a new approach for better cloud computing IaaS Services is presented. It is based on allocating idle generic resources to maximize the cloud computing QoS at low cost. This is accomplished through the implementation of the harvesting middleware that represents a bridge from OpenStack cloud computing platform to generic resources donated by the on-premises desktop machines. These resources run a set of different operating systems platforms. This bridge enables us (with simple modifications) to work with a number of cloud computing platforms like Amazon Ec2, Eucalyptus, and other that use almost the same virtualization methodology. According to our implementation results, extending the available cloud computing platform guarantees cloud service availability and an opportunistic use of idle resources. This will provide a better service elasticity and scalability through using both of the dedicated and non-dedicated infrastructures. The action of moving cloud computing services to generic resources, such as public desktop machines available in homes, universities or enterprises, will not affect the provided cloud services. But it proves that using this mix of generic resources and dedicated cloud computing infrastructure supplies inexpensive resources with guarantee of QoS.

8. REFERENCES

[1] V. Delgado, "Exploring the limits of cloud computing," master of science thesis, Kungliga Tekniska Hgskolan (KTH),Stockholm, Sweden, October 2010.

[2] S. Wind, "Open source cloud computing management platforms: Introduction, comparison, and recommendations for implementation," in 2011 IEEE Conference on Open Systems (ICOS), (Langkawi), pp. 175 – 179, Business Informatics and Systems Engineering,University of Augsburg, Augsburg, Germany, IEEE, Sept 2011.

[3] I. T. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," CoRR, vol. abs/0901.0131, 2009.

[4] D. Parrilla, "Stackops documentation," May 2011.

http:// docs.stackops.org/ display/ doc03/ Home.

[5] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of several cloud computing platforms," 2009 Second International Symposium on Information Science and Engineering, pp. 23–27, 2009.

[6] I. Eucalyptus Systems, "Cloud roles - developers."

http:// www.eucalyptus.com.

[7] I. OpenNebula.org, "Opennebula," 2012.http:// opennebula.org/ .

[8] E. Williams, "Energy intensity of computer manufacturing:hybrid assessment combining process and economic input/output methods," Environ- mental Science & Technology, vol. 38, no. 22, pp. 6166–6174, 2004.

[9] G. Kirby, A. Dearle, A. Macdonald, and A. Fernandes, "An Approach to Ad hoc Cloud Computing," ArXiv e-prints, Feb. 2010.

[10] A. Cuomo, G. D. Modica, S. Distefano, M. Rak, and A. Vecchio, "The cloud@home architecture - building a cloud infrastructure from volunteered resources," in 2011 - Proceedings of the 1st International Conference on Cloud Computing and Services Science, Noordwijker- hout, Netherlands, 7-9 May, 2011 (F. Leymann, I. Ivanov, M. van Sinderen, and B. Shishkov, eds.), pp. 424–430, SciTePress, 2011.

[11] S. Distefano, M. Fazio, and A. Puliafito, "The cloud@home resource management system," in IEEE 4th International Conference on Utility and Cloud Computing UCC, pp. 122–129, IEEE Computer Society, 2011.

[12] V. D. Cunsolo, S. Distefano, A. Puliafito, and M. Scarpa, "Volunteer computing and desktop cloud: The cloud@home paradigm," in Proceedings of The Eighth IEEE International Symposium on Networking Computing and Applications NCA, pp. 134–139, IEEE Computer Society, 2009.

[13] S. Distefano, A. Puliafito, M. Rak, and S. Venticinque, "Qos management in cloud@home infrastructures," in CyberC, (Beijing,China), pp. 190–197, International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, IEEE, Oct 2011.

[14] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, GRID '04, (Washington, DC, USA), pp. 4–10, IEEE Computer Society, 2004.

[15] U. of California, "Computing with boinc," 2012.

http:// boinc.berkeley.edu/ .

[16] M. G. . A. G. P. Buncic, "Boinc service for volunteer cloud computing," in Computing in High Energy and Nuclear Physics (CHEP) 2012, Distributed Processing and Analysis on Grids and Clouds, (New York City, NY,

USA), Citizen Cyberscience Centre, CERN, CH-1211 Switzerland, May 2012.

[17] B. Sodhi and T. Prabhakar, "A cloud architecture using smart nodes," Asia-Pacific Conference on Services Computing. 2006 IEEE, vol. 0, pp. 116–123, 2011.

[18] E. Rosales, H. Castro, and M. Villamizar, "Unacloud : Opportunistic cloud computing infrastructure as a service," CLOUD COMPUTING 2011 The Second International Conference on Cloud Computing GRIDs and Virtualization, no. c, pp. 187–194, 2011.

[19] S. Di, C.-L. Wang, L. Cheng, and L. Chen, "Social-optimized win- win resource allocation for self-organizing cloud," Cloud and Service Computing, International Conference on, vol. 0, pp. 251–258, 2011

[20] G. C. Guang Li, "A novel enhanced education application of cloud computing," in 2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), Unit 61906, PLA, Langfang, China, IEEE, Sept. 2011.

[21] H. Lin, X. Ma, J. Archuleta, W.-c. Feng, M. Gardner, and Z. Zhang, "Moon: Mapreduce on opportunistic environments," in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC '10, (New York, NY, USA), pp. 95–106, ACM, 2010.

[22] L. AG, "Wuala storage cloud," 2012.

http:// www.wuala.com/ en.

[23] A. Chandra and J. Weissman, "Nebulas: using distributed voluntary resources to build clouds," in Proceedings of the 2009 conference on Hot topics in cloud computing, HotCloud'09, (Berkeley, CA, USA), USENIX Association, 2009.

[24] R. C. M. Akshay K. Singh, "Symbiotic computing," in 2010 IEEE 4th International Symposium on Advanced Networks and Telecommunication Systems, Yahoo! Software Development Center, India, IEEE, 2010.

[25] E. Algizawy, A.E.S Ahmed, and A. Alsammak, "A generic resources allocation approach for better cloud computing iaas services," in PDPTA'12, WORLDCOMP'12 USA, 2012.

[26] A. Waterland, "Stress workload generator," Junuary 2009. http://weathe.ou.edu