**Volunteer Cloud Computing (the name needs revising with Dr. Alaa, I think it must contain the term "Harvesting")**

**Abstract**

**Please read the abstract and suggest any changes**

The distributed Computation world is becoming complex and very large; Cloud Computing has emerged as a popular computing model to support large data processing using commodity hardware.

~~Most cloud computing infrastructures consist of services delivered through common centers and built on dedicated servers.~~ In this context, the number of physical units (dedicated servers) may suffer shortage problem at peak time due to the increasing needs of using resources. In this paper we introduce a solution for resolving this ~~increase~~ problem by adopting the use of idle general purpose machines exists within homes, universities or enterprises to be part of the cloud computing environment. ~~We implement this solution by creating~~ We propose to use additional computational resources with reliable computing power using CPU cycles denoted by public machines through using of OpenStack **Cloud computing software** platform for on-premise Infrastructure as a Service (IaaS) clouds. *The statement is too long and need to decompose into clear statements* ~~To supply reliable inexpensive public resources we propose to port on BONIC platform~~. The proposed solution was implemented and tested through the provision of our new IaaS model, evidencing high efficiency in the deployment of ~~virtual machines for different applications~~ Cloud Computing services on generic resources, showing that this mix of generic and non-generic resources can supply inexpensive resources with the guarantees of the quality of services. The test results must be stated in measurable units and compared with previous work.

**Introduction**

**We need here to add quick introduction to clouds, its concepts , problem that my suffer due to lack of resource, indicate the availability of huge number of unused resources, quick introduction to your system**

The rapid growth in the distributed computing environment technologies have led away from the in-house Public Resources Computing to Grid computing and now to Cloud Computing paradigm. In this paradigm, Cloud resources are delivered in a flexible and service-oriented manner to consumers on an on-demand pay-as-you-go manner. This differs from common practice in IT whereby resources are procured and used in a more coarse-grained and less flexible manner.

~~Most of the cloud computing implementation based on the Infrastructure as a Service (IaaS) model, due in part to the complexity associated with different cloud computing delivery models.~~

Cloud Computing is not a completely new concept for development and operations of the web application [*A taxonomy and survey* ] .In the cloud we have to address a set of Cloud Computing fundamentals like virtualization management ,scalability , interoperability, fail over mechanism and elasticity which represent one of the major attraction of the cloud. We have to address Cloud Computing delivery models (Publ ic, private, and Hybrid Clouds), focus on the cloud architecture, and services as well.

Cloud architectures are underlying on infrastructure (Infrastructure as a Service – IaaS) which is used only when it is needed to draw the necessary resources on-demand to perform a specific tasks, or simply we can say that IaaS is the delivery of computer infrastructure as a service aside from the higher flexibility. These architectures address the key difficulties surrounding large scale data processing .Cloud can basically defined in three deployment modes[ *Taxonomy*] ;1) **Private Cloud:** Data and processes are managed within an organization boundaries without any restrictions on the network bandwidth, security exposures , or legal requirement for using service like these on public services. 2) **Public Cloud:** it describes the cloud computing in the traditional mainstream sense, whereby resources are dynamically provisioned on a fine grained, self-serviced basis over web services from an off-site third parity provider. **3) Hybrid Cloud:** the environment ~~is~~ consists of ~~two or~~ private and public cloud computing providers.

Although private Cloud Computing is more secure than Public Cloud Computing, its elasticity is limited depending on the maximum hardware capacity and at the peak time it is bound to fall short of demand which known as Cloud burst. As the needs for computational and storage resources increase the technology needs to develop to fulfill this increasing needs of using resources. We propose a new architecture to ad-hoc additional computational resources with reliable computing power using CPU cycles denoted by public (Organization PC.s) through using of OpenStack Cloud computing software platform for on-premises Infrastructure as a Service clouds as computational and storage resources within organizations are often under-utilize for significant periods of time.

The availability of volunteer desktop machines as a part of the cloud leads to various benefits to individual organizations; it could reduce the numbers of dedicated machines needed to be purchased. 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 center costs [*Cost of the cloud paper*].Rather than ensuring resilience of a small number of physical buildings, the grain of resilience could be expanded by using more widely distributed machines and tolerating individual building failures [*ad-hoc cloud*]. As well as it Reduces overall power consumption through the reduction in the total number of machines required significant since the energy cost of manufacture for a computer has been estimated as four times that used during its lifetime [*Energy Intensity paper*]

The paper is organized as follows. ~~In section 2, related work is described~~  Section 2 presents the related works on Cloud Computing IaaS implementations, Desktop Grid and public resources computing. The proposed Architecture is given in section 3. The architecture implementation and results is presented in section 4. Finally, section 5 introduces the conclusion.

**Related work**

The Work proposed in [*UnaCloud pape*r] explain an opportunistic cloud computing Infrastructure as a Service (IaaS) model implementation, which provides at lower cost than dedicated cloud infrastructures based on basic computing resources (processing, storage and networking) through the opportunistic use of idle computing resources available in a university campus. UnaCloud uses a commodity and non-dedicated underlying infrastructure to implement the opportunistic design concepts which broadly effect on; 1) the cloud services availability as the computing resources is dependent on the behavioral pattern of their currently owners, as the availability of the service represent an effective term to ensure any type of QoS or SLA. 2) Service scalability will always depend on the available resources on the opportunistic idle desktop machine which will limit the service scalability .3)The proposed architecture doesn’t support integration with any of the currently used Cloud Computing platform APIs which limit their use at peak times or at cloud burst.

Private cloud would provide flexible service availability, scalability and addition of hybrid clouds will provide elasticity not only but also would provide a self-managing in terms of resilience, performance and balancing potentially.

Our approach focus on extending the available Cloud Computing platform infrastructure which ensure service availability with the opportunistic use of idle resources which will provide services elasticity and service scalability through the using of both of the dedicated and non-dedicated infrastructure.

In [integration of grid cloud] they proposed architecture to integrate grid and private cloud such that grid resource requirements are fulfilled by fetching resources from cloud when needed and vice versa with additional storage clusters to enhance the performance as Grid Computing resources are not utilized optimally most of the times.

In [*Ad-hoc cloud*] an outline of the major implementation challenges has been introduced on how underused computing resources within an enterprise may be harnessed to improve utilization and create an elastic computing infrastructure, In contrast to the data center cloud model. This model is analogous to volunteer computing as exemplified by Condor [*The Condor Experience*] and BOINC [*BONIC*] [*High-Performance Task Distribution for VC*], although it poses considerable additional implementation challenges.

Volunteer computing (VC), Public Resources Computing (PRC) sometimes described as a paradigm of a large number of computers, on which individual users’ machines are used to perform computationally tasks e.g. SETI@home, Climateprediction.net, and LHC@home are the most popular projects based on the BOINC (Berkeley Open Infrastructure for Network Computing) framework [*BONIC*] [*High-Performance Task Distribution for VC*] or based on Condor [*The Condor Experience*] .Volunteer computing is being used in high-energy physics, molecular biology, medicine, astrophysics, climate study, and other areas.

In the context of IaaS models, OpenStack Cloud Computing platform [*OpenStack*] is a collection of open source software projects that service providers can use to run their cloud compute and storage infrastructure.it is backed by Rackspace, NASA, Dell, Citrix, Cisco, and Canonical and over a hundred other organizations but it is uses only dedicated servers to setup such Cloud Computing platform like that platform which created using Eucalyptus [*Eucalyptus*]

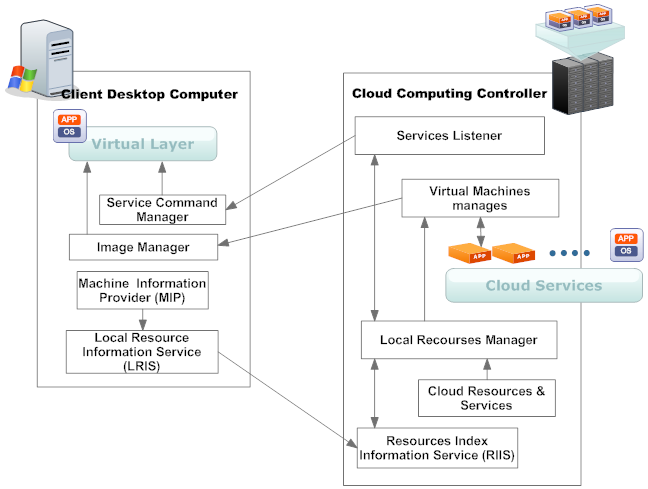
Our approach using the same idea behind the building methodology of OpenStack Cloud Computing platform extending the infrastructure used by harvesting the idle computing resources donated by the public (Organization PC.s).

**Proposed Architecture**

The architecture has three major components; Cloud Computing platform, Harvesting Middleware, Resources allocation desktop agent.

**Cloud Computing platform:** The Cloud Computing environment consists of Cloud Controller which receive submitted user requests and passes to the underlying component as it represents the interaction between ~~compute~~ the computing nodes (processing), volumes (block storage), the networking controllers (software which controls network infrastructure), and API endpoints. The Cloud Controller control and schedule resources within the computing nodes, services Ips through the controlling of network node, and in turn control the storage node. The computing nodes act as an interface between the cloud controllers and Hypervisor, the virtualization model ~~used~~ uses a layer (Hypervisor) that mediates between the guest operating systems (Virtual machine or the Cloud Service) and the native hardware. The hypervisor is responsible for bring up virtualization by handling a set of certain protected instructions.

KVM is a full virtualization solution that is unique in that it turns a Linux kernel into a hypervisor using a kernel module. This module allows other guest operating systems to ~~then~~ run in user-space of the host Linux kernel.



Each time a request for a new cloud computing service is submitted, the cloud controller checkup the available resources e.g. cores, memory , storage to deploy the virtual machine 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 local resources manager to free up resources by moving one of the currently running cloud services to the most suitable registered desktop machine and suspend this cloud virtual machine after deploying it on the selected desktop machine.

**Harvesting Middleware:** The harvesting middleware consists of five daemons, each of these daemons play a vital role in running any desktop machine with the cloud computing. 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).

**Resources index information service (RIIS):** RIIS represents a lightweight directory of the attached desktop machines, containing the basic information about local desktop resources and which one is alive through 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. This along with catching information throughout the system makes it quite scalable. This information offered by the RIIS is about desktop cores, memory, load status, desk storage and cloud service (Virtual machine) running on it if any.

**Cloud Resources and Service (CRS):** it is contain a complete list of Cloud platform resources e.g. computing node, network nodes, and storage as well. Each computing node with a list of the running services associated with service information e.g. 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.

**Local Resources manager (LRM):** LRM is responsible for managing the execution of any cloud computing services on a desktop machine ,keeps track of all cloud computing services running on desktop machines , and it also periodically query the status desktop running tasks from RIIS.

If the Cloud Controller request the LRM to free up cloud resources ,the LRM will select the most suitable desktop machine from information provided by RIIS with respect to running cloud service information obtain from CRS, taking a snapshot of the running virtual machine to be deployed on the select desktop machine.

**Virtual machine manager (VMM):** VMM works in conjunction with the LRM to manage virtual machines in the cloud environment and on desktop machines .it is responsible for creating image snapshots , suspend and start running cloud services as well as on desktop machines it can issue a deployment commends for starting , stopping , and monitoring cloud computing virtual machines on desktop PCs.

**Services Listener (SL):** All users requests (Start, Terminate, Snapshot …etc.) are submitted to the cloud controller with the associated service identification. SL listen to all request passed to the cloud controller and redirect basic requests to LRM registered cloud service running on desktop machines instead of passing it to the compute nodes running these services or these that have a suspended image to guarantee service availability in case that desktop service fail.

**Resources allocation desktop agent (RAD)**: RAD consists of a simple for module, if we start from the bottom 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. Up on installing the RAD agent on any desktop machine the MIP start by 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) responsible for issue a simple virtualization commands for starting new virtual machine, terminating or monitoring it also manages desktop virtual machine networking. The last module is service command manager, the command manager used to execute basic requests redirect form SL Daemon on the running desktop cloud services.

**Architecture implementation and results**

Cloud computing environment is established with the OpenStack [*openStack*] [*Nova*] as a Cloud Computing middleware, KVM virtualization [*virtualization*] as a hypervisor which provide scalability and user isolation on OpenStack Cloud, and Libvirt [*libvirt*] for hypervisor communication with cloud middleware.

The desktop Cloud services deployment include the provisioning of the necessary data for its remote access using standard SSH [*SSHD*] mechanisms, QEMU as a desktop virtualization tool, and Libvirt for virtualization interfaces.

**Conclusion**