A "No Data Center" Solution to Cloud Computing

Tessema Mengistu*, Abdulrahman Alahmadi*, Abdullah Albuali, Yousef Alsenani, and Dunren Che
Dept. of Computer Science, Southern Illinois University at Carbondale
Carbondale, Illinois USA
Email: tessema.mengistu, aalahmadi, aalbuali, yalsenani, dche@siu.edu

***Abstract*—Current Cloud Computing is primarily based on proprietary data centers, where hundreds of thousands of dedicated servers are setup to host the cloud services. In addition to the huge number of dedicated servers deployed in data centers, there are billions of underutilized Personal Computers (PCs), usually used only for a few hours per day, owned by individuals and organizations worldwide. The vast untapped compute and storage capacities of the underutilized PCs can be consolidated as alternative cloud fabrics to provision broad cloud services, primarily infrastructure as a service. This approach, thus referred to as "no data center" approach, complements the data center based cloud provision model. In this paper, we present our opportunistic Cloud Computing system, called cuCloud, that runs on scavenged resources of underutilized PCs within an organization/community. Our system demonstrates that the "no data center" solution indeed works. Besides proving our concept, model, and philosophy, our experimental results are highly encouraging.**

***Keywords*-Cloud Computing; IaaS; Volunteer Computing; No Data Center Solution; Credit Union Model; Credit Union Cloud.**

I. Introduction

The current Cloud Computing services are based on the "data center" approach, where hundreds of thousands of dedicated servers are setup to give the services. Setting up the data center for cloud is expensive and running the infrastructure needs expertise as well as a lot of resources such as high power for cooling, redundant power for assured availability, etc. For example, 45% of the data center cost goes to the acquisition of servers, 25% goes to specialized infrastructure for fault tolerance, redundant power, cooling systems, and backup batteries, while electrical cost consumed by the machines accounts for 15% of the amortized total cost [1]. In addition to the vast number of servers used in data centers, there are billions of Personal Computers (PCs) owned by individuals and organizations worldwide. These PCs are mostly underutilized, usually used only a few hours per day. Researches show that desktop computers owned by organizations are idle up to 97% of the time [2]. We had argued [3] that we shall treat the untapped CPU cycles and disk spaces of the great many underutilized PCs as precious assets, like monetary assets, to consolidate and reuse them for the good of the society and of the individuals just like the way that a credit union works. This argument had motivated an alternative Cloud Computing provision model, named "Credit Union Cloud Model" (or CUCM for short) [3]. Cloud services (mainly IaaS) built based on the CUCM are generally referred to as Credit Union Clouds (CU clouds for short). The key characteristic in CUCM is the "no data center" approach to provisioning Cloud Computing services for an institution, organization, or community. With the current public clouds, which are better called *vendor clouds* as they are all provided by vendors based on dedicated data centers, the concern for security/safety and loss-of-control is the primary obstacle keeping traditional IT from moving to clouds. It is understood that if the data of a business is highly confidential, the business owner is of course overly concerned about placing the data in the hands of another party. On-premise private cloud is plausibly a solution to mitigating this concern. However, the need of big upfront investment to setup the data center for the private cloud infrastructure can be prohibitively expensive. Among many other benefits of CU clouds, affordability (which means almost no additional cost for acquiring and running an on-premise cloud infrastructure) is particularly appealing. It can help an organization or business owner save up to the 45% of the cost of a data center by eliminating the upfront purchase for the cloud servers, which would otherwise be necessary. In addition, the credit union cloud infrastructure does not need additional cooling systems, which saves the additional 15% of data center's cost on cooling. In general, our credit union cloud management system provides a feasible on-premise solution to Cloud Computing for institutions and organizations that highly care about cost and security.

This paper is organized as follows. Section II reviews CUCM, our "no data center" cloud model; Section III provides an implementation overview of CUCM as demonstrated in cuCloud, and some empirical results and analysis; Section IV addresses related works; and finally, Section V concludes the paper and outlines the future work.

II. Credit Union Cloud Model

The Credit Union Cloud Model is such a cloud provision model that aims at tapping into the overabundant idle/underutilized computers for cloud service provisioning, while the standard practice is based on dedicated data centers. CU clouds run on existing infrastructures with excessive capacities, which are not specifically setup for supporting Cloud Computing. These PCs are not

2159-6190/17 $31.00 © 2017 IEEE
DOI 10.1109/CLOUD.2017.99

dedicated resources for the cloud infrastructure, instead they are still used by their intended users as usual, e.g., running a word processor or web browsing (referred to as *local/native* applications). CU cloud allows the PCs within an organization to join the "cloud credit union" and contribute their underutilized resources (CPU cycles or disk spaces) to the union's cloud resource pool to back up its cloud fabric [3]. Accordingly, these contributing PCs are called *member/volunteer* nodes.

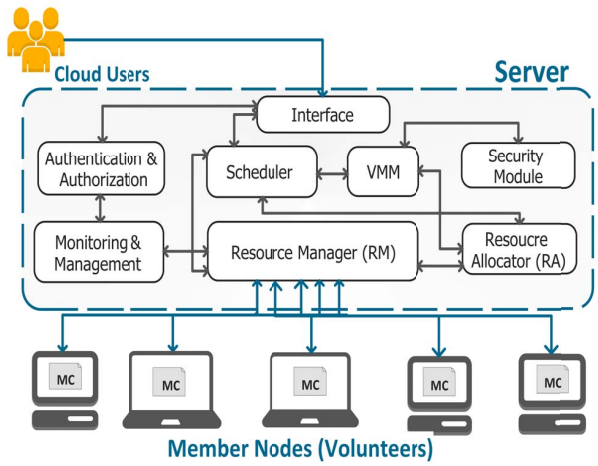

Fig. 1: Architecture of CUCM/cuCloud

CUCM assumes a client/server architecture with member nodes as clients and dedicated management machine(s) as server(s). The server has different components as depicted in Fig.1. These components are briefly explained as follows. The *Interface* is the first port of communication between CU Cloud and its users/clients, whose access will be authenticated and authorized by the *Authentication and Authorization* module. The *Resource Manager (RM)* has the global picture of the resources that the cloud infrastructure has as a whole. The *Resource Allocator (RA)* component selects a list of suitable member nodes for the deployment of Virtual Machines (VMs) according to the resource requirements of the cloud customer, the Service Level Agreement (SLA), and the availability as well as reliability profiles of the member nodes. The *Scheduler* module accepts user requests and allocates or denies the requested resources in consultation with the *Virtual Machine Manager (VMM)* and the Resource Allocator. The VMM component handles the deployment of VMs on member nodes. The *Security* Module handles the security of the Virtual Machines. The *Monitoring and Management* module gives fine-grained resource information about the resources of the CU Cloud system.

There is a critical piece of software, called the Membership Controller(MC), that resides on each member node that contributes resources to the CU Cloud system. MC monitors the resource usage at a member node and decides the node's membership status: *active* status indicates there are enough resources to meet the need of a minimum VM, while *inactive* status indicates unavailability of such resources. The Membership Controller collects and sends information to the server about the types and quantities of the available resources (CPU, RAM, Hard Disk) for contribution to the resource pool of CU Cloud periodically. Membership Controller has the following components. The *Sensor* component monitors the resource usage of processes on a member node and gives that information to the Reporter component. The *Reporter* component decides the status of membership of the node in the cloud infrastructure based on the sensed information. If the resource (RAM, CPU, Hard Disk) availability is above/below some threshold value, it will send a message to the Resource Manager on the server indicating that it is either an active or inactive member. The *Virtual Environment Monitor* component manages the VMs that are deployed on the member node.

III. Implementation and Experimentation

A. Implementation

As a cloud management system and platform for delivering IaaS Services, CU Cloud is expected to fulfill all the characteristics of Cloud Computing such as elasticity, metering service, multitenancy, etc. As the first step of our project, we have implemented a preliminary version of CUCM (called cuCloud) using Apache CloudStack. Apache CloudStack is an open source IaaS platform that manages and orchestrates various resources including storage, network, and computers to build a public or private IaaS compute cloud [4]. CloudStack has management server(s) that can manage tens of thousands of physical servers installed in geographically distributed data centers. The Management Server of CloudStack communicates with the compute nodes (physical servers) through the hypervisors (Xen, KVM, Hyper-V, etc) installed on the machines. Since CloudStack is an IaaS system that is developed to manage dedicated data centers with only dedicated hosts, we need to modify the management server of CloudStack in such a way that it can handle the non-dedicated member nodes that physically back up the cuCloud system. We developed a special component (called AdHoc component) and integrated it into the CloudStack management server to form cuCloud management server. On the other side, we used SIGAR (System Information Gatherer and Reporter)[5] for the Sensor component of Member Controllers that reside on member nodes. The number of CPU cores, the RAM capacity, idle CPU percentage, free memory percentage, and available free hard disk space are sensed/gathered and passed to the Reporter module of MC. Each instance of Membership Controller running on a member node continuously senses the resource usage of the processes on the member host. If the resource utilization at the member node is below a

certain threshold, the MC instance sends an "active" message to the AdHoc component on cuCloud management server, and an "inactive" message otherwise. The AdHoc component will update the resource base of Cloudstack based on the message it accepts from the MCs. The other components of CloudStack remain unchanged in this preliminary version of cuCloud. Due to the full self-autonomy of member nodes, we cannot use type I hypervisors like Xen or Hyper-V. Instead, we need a virtualization solution that runs along with other applications on the member nodes. Therefore, we chose to use KVM (Kernel-based Virtual Machine), which introduces virtualization by augmenting the traditional kernel and user modes of Linux with a new process mode called *guest* that has its own kernel and user modes and answers for code execution from guest operating systems [6]. This characteristic of KVM allows us to run VMs along with local applications on member nodes at the same time.

B. Experimentation

To test the feasibility of CUCM, we conducted two sets of experiments using one server and four client machines. We used the modified CloudStack version 4.9.0 as discussed in the implementation subsection. The Management Server of CloudStack is installed on a machine with 8 GB of RAM, Intel 8 Core i7 2.4 GHz CPU, and a hard disk of 250GB. Each computing nodes has 8 GB of RAM, intel 4 cores i3 3.1 GHz CPU, and 250GB hard disk capacity. All the machines run Ubunu 14.04, are connected to a 16Gbps switch, and supports intel hardware virtualization (VT-x). For the first set of experiments, we setup a dedicated CloudStack infrastructure using one Management Server and four compute nodes. The performance and usage of CPU and RAM are measured using well-known benchmarks, LINPACK [7] and STREAM [8] respectively. The second set of experiments was done on a modified CloudStack, renamed as cuCloud as it implements our CUCM model. cuCloud also assumes one Management Server node and four member nodes, with which the benchmarks were run to gather the same set of measurements for comparison. A member node will join cuCloud infrastructure if its CPU idle percentage is greater than or equal to 70%. We used five scenarios to compare the performance of CloudStack assuming dedicated machines (or data center) vis–a-vis our cuCloud relying on contributing member compute nodes. The experiments with cuCloud were conducted while the local users were still using the machines. The 5 scenarios are as follows:

Scenario 1: On dedicated CloudStack infrastructure, one small VM instance (1 vCPU, 512MB RAM, 20GB HD) is deployed on one of the computing nodes.
Scenario 2: On dedicated CloudStack infrastructure one medium (2 vCPU, 1GB RAM, 20GB HD) and two small VM instances are deployed on one of the compute nodes, and on one of the instances the performance measurement tasks are carried out while the other two instances are set busy with 40% and 60%(CPU usage).
Scenario 3: Same as Scenario 1 except it is on cuCloud non-dedicated infrastructure.
Scenario 4: Same as Scenario 2 but on cuCloud. Here, we purposefully make the member node that hosts the benchmark tasks busy to cause live migration of the VM with the performance measurement tasks.
Scenario 5: Same as Scenario 4, but we purposefully induced two live migrations of the VM that hosts the performance tasks.

Targeted at CPU-intensive computations involving large linear equations, the LINPACK benchmark was set up with matrix dimensions of 5000x5000. We gather the average of 10 executions of the LINPACK benchmark. The following diagrams (Fig.2 and Fig.3) depict the experimental results. As shown in Fig.2, there is almost no difference between running a task on a CloudStack dedicated compute node and on a cuCloud shared member node, as long as there is enough resource to execute the task. However, as shown in Fig.3, when there are one or more migrations, the tasks running on cuCloud might take longer time. This is obviously a consequence of the induced migration of the VMs. From our experimental result, the performance gap between one migration and two migration does not seem to be obvious (12.64 vs 12.71 seconds), which however may not be generalized.

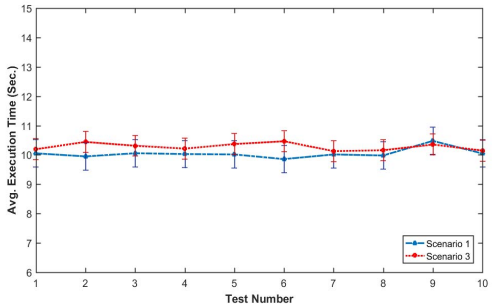

Fig. 2: LINPACK: Dedicated vs. cuCloud (no migration)

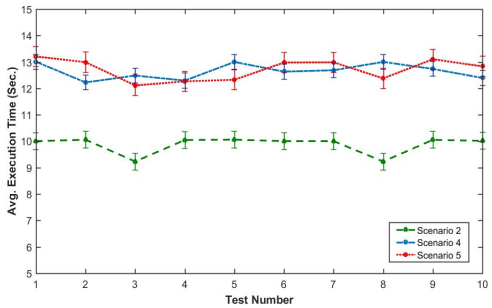

Fig. 3: LINPACK: Dedicated vs. cuCloud (with migration)

STREAM is designed mainly to measure the memory bandwidth using four operations, Add, Copy, Scale and Triad, and was set up with an array size of 2,000,000. Fig.4 depicts the average bandwidth usage of 10 trials of running STREAM with the 5 scenarios. As Fig.4 shows, the bandwidth usage of the four operations in the first 3 scenarios is almost the same; and VM migration noticeably causes increase in bandwidth usage.

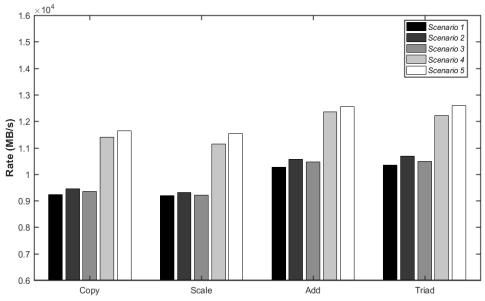

Fig. 4: STREAM Bandwidth Rate

Our preliminary implementation of CUCM and experiments with both cuCloud and CloudStack demonstrate that not only the concept of CUCM (i.e., no data center cloud solution) works but also it presents a promising alternative approach to Cloud Computing with many facets of advantages.

IV. Related Works

There are very few works that have been done in the research direction that we embark on: the development of Cloud Computing model based on spare resources of PCs as its resource base. One noticeable work is the so-called *ad hoc* cloud reported in [9] wherein various research issues related to cloud provisioning using general-purpose computers were explored. The proposed cloud infrastructure architecture consists of several components: one creates/destroys cloud elements; one monitors the effects of created cloud elements; one handles the QoS issues; and one executes allocated tasks. The authors gave no actual implementation of their *ad hoc* cloud system. Another work in the area of non-dedicated data center based Cloud Computing is [10] that tried to investigate the feasibility, reliability and performance of *ad hoc* Cloud Computing infrastructures. The *ad hoc* cloud system, which is a client/ server system, is based on the well-known VC system BOINC with a virtualization support called V-BOINC. The server component has three subcomponents: *Cloud Interface, VM Service,* and *Job Service.* The client is the one that accepts the jobs and executes them reliably. The research concluded that *ad hoc* cloud is not only feasible, but also a viable alternative to the current data center based Cloud Computing systems. The authors mentioned nothing about the elasticity, multitenancy, etc. characteristics of the system.

V. Conclusion and Future Work

Credit Union Cloud Model, which aims at tapping into the underutilized computing resources available within an organization/community rather than dedicated servers, provides a promising alternative Cloud Computing solution for organizations and communities. Our work demonstrates that the "no data center" solution indeed works. Besides proving the concept, model, and philosophy of CUCM, our experimental study turned out to be highly encouraging – the "no data center" solution can gain highly competitive performance compared to its counterpart that depends on dedicated cloud servers.

The most significant aspect of our work so far is that by cuCloud we have setup a platform and have a door widely open for many exciting new research issues for the future. The resource pool over which VMs run in cuCloud is not dedicated but shared with native users/tasks. We need to devise a mechanism to provide cloud services reliably and efficiently, while keeping the services from interfering with the native users/tasks at member nodes. Another requirement of CUCM is to have a robust, dynamic and efficient resource management and provisioning mechanism. The resource management and provisioning module should consider the dynamic and unreliable nature of the member hosts that contribute resources to the resource pool of cuCloud. We also need to investigate novel and efficient scheduling algorithms that consider the availability, location, and reliability of the member nodes used by the system to deploy VMs. In addition, cuCloud requires strong security measures in order to insure the security of member nodes from malicious cloud client processes and client VMs from malicious native users at member nodes.

References

[1] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008.

[2] P. Domingues, P. Marques, and L. Silva, "Resource usage of windows computer laboratories," in *Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on.* IEEE, 2005, pp. 469–476.

[3] D. Che and W. C. Hou, "A novel "credit union" model of cloud computing," in *International Conference on Digital Information and Communication Technology and Its Applications.* Springer, 2011, pp. 714–727.

[4] Apache cloudstack. [Online]. Available: http://cloudstack.apache.org

[5] System information gatherer and reporter. [Online]. Available: https://support.hyperic.com/display/SIGAR/Home

[6] J. Che, Q. He, Q. Gao, and D. Huang, "Performance measuring and comparing of virtual machine monitors," in *Embedded and Ubiquitous Computing, 2008. EUC'08. IEEE/IFIP International Conference on*, vol. 2. IEEE, 2008, pp. 381–386.

[7] J. J. Dongarra, P. Luszczek, and A. Petitet, "The linpack benchmark: past, present and future," *Concurrency and Computation: practice and experience*, vol. 15, no. 9, pp. 803–820, 2003.

[8] J. D. McCalpin, "Stream: Sustainable memory bandwidth in high performance computers," University of Virginia, Charlottesville, Virginia, Tech. Rep., 1991-2007.

[9] G. Kirby, A. Dearle, A. Macdonald, and A. Fernandes, "An approach to ad hoc cloud computing," *arXiv preprint arXiv:1002.4738*, 2010.

[10] G. McGilvary, "Ad hoc cloud computing," PhD dissertation, The University of Edinburgh, 2014.