cuCloud: Volunteer Computing as a Service (VCaaS) System

Tessema M. Mengistu(✉), Abdulrahman M. Alahmadi, Yousef Alsenani, Abdullah Albuali, and Dunren Che

Department of Computer Science, Southern Illinois University at Carbondale, Carbondale, USA
{tessema.mengistu,aalahmadi,yalsenani,aalbuali,dche}@siu.edu

Abstract. Emerging cloud systems, such as volunteer clouds and mobile clouds, are getting momentum among the current topics that dominate the research landscape of Cloud Computing. Volunteer cloud computing is an economical, secure, and greener alternative solution to the current Cloud Computing model that is based on data centers, where tens of thousands of dedicated servers are setup to back the cloud services. This paper presents cuCloud, a Volunteer Computing as a Service (VCaaS) system that is based on the spare resources of personal computers owned by individuals and/or organizations. The paper addresses the design and implementation issues of cuCloud, including the technical details of its integration with the well-known open source IaaS cloud management system, CloudStack. The paper also presents the empirical performance evidence of cuCloud in comparison with Amazon EC2 using a big-data application based on Hadoop.

1 Introduction

Cloud Computing, the still fast growing and evolving technology, has kept drawing significant attention from both the computing industry and academia for nearly a decade. Extensive researches in Cloud Computing have resulted in the fulfillment of many of the promised characteristics, such as utility computing, pay-as-you-go, flexible (unlimited) capacity, etc. Emerging cloud systems such as ad hoc and mobile clouds, volunteer clouds, federation of clouds, and hybrid clouds are among the research topics that may jointly help to reshape the landscape of future Cloud Computing – to better satisfy computing needs.

Currently, more than 4 billion users use the Internet[1]. Supercomputing sites and large cloud data centers provide the infrastructures that host applications like Facebook and Twitter, which are accessed by millions[2] of users concurrently. The current cloud infrastructures that are based on tens (if not hundreds) of thousands of dedicated servers are expensive to setup; running the infrastructure needs expertise, a lot of electrical power for cooling the facilities, and redundant

[1] http://www.internetworldstats.com/.
[2] http://www.internetlivestats.com/.

M. Luo and L.-J. Zhang (Eds.): CLOUD 2018, LNCS 10967, pp. 251–264, 2018.
https://doi.org/10.1007/978-3-319-94295-7_17

supply of everything in a data center to provide the desired resilience. Cloud servers, specialized infrastructure for fault tolerance, and the electricity consumption account for 45%, 25%, and 15% of the total amortized cost of a cloud data center, respectively [1]. The servers and the cooling system make up about 80% of all the electricity consumption in a typical data center [2]. The high energy consumption in data centers contributes to the environmental impact of global warming. Currently, the ICT industry contributes 2% of the total greenhouse gases emission and the contribution of data centers to this emission is expected to increase to 18% by 2020 [3]. On top of the costs and environmental factors, the current "datacenter based" cloud infrastructures, even though well-provisioned and well-managed, are not suitable for some application scenarios. For instance, the inevitable concerns of cloud clients for "no control" on their confidential data and business logic can prohibit them from migrating their applications and datasets to the clouds that are in the hands of third-parties. Self-provisioned, on-premise, private cloud can be a suitable solution to these users. However, the initial investment for setting up such a private cloud infrastructure based on a dedicated data center can be prohibitively expensive.

In addition to the vast number of dedicated servers setup in cloud data centers, there are billions of Personal Computers (PCs) owned by individuals and organizations worldwide. These PCs are underutilized, usually used only for a few hours per day [4]. The computing power of these computers can be consolidated as a huge cloud fabric and utilized as an alternative Cloud Computing solution, i.e., volunteer cloud computing. Volunteer cloud computing based on consolidating the spare capacities of edge computing resources (within an organization or community) and delivering on-premise private clouds is a rather economical and greener alternative to the conventional "data center based" Cloud Computing solutions.

Volunteer clouds come with multi-folds of benefits: no upfront investment for procuring a large number of servers that are otherwise inevitable for data center hosting, no maintenance costs such as electricity consumption for cooling and running the servers in a conventional cloud data center, and boosting the utilization of edge computing resources (such as individually owned PCs). In the meantime, volunteer cloud computing introduces its own technical challenges that are centred on the high dynamics and high heterogeneity of the volunteer computers that are shared not only among the cloud users but also between the cloud users and the local users of the machines. The key issues in Cloud Computing such as availability, reliability, security, and privacy all need to be readdressed in a more serious and critical way in volunteer cloud computing. Nevertheless, exploitation of the untapped excessive capacity of the numerous edge computers owned by individuals and/or organizations for volunteer cloud computing is a worthwhile endeavour [5]. We believe "no data center based" volunteer cloud computing is one of the most promising future research directions of Cloud Computing [6,7].

This paper discusses cuCloud, a Volunteer Computing as a Service (VCaaS) system that is based on the spare resources of personal computers owned by

individuals and/or organizations. The paper addresses the design and implementation issues of cuCloud, including the technical details of its integration with the well-known open source IaaS cloud management system, CloudStack. The paper also presents the empirical performance evidence of cuCloud in comparison with Amazon EC2 using a big-data application based on Hadoop. To the best of our knowledge, cuCloud is the first volunteer cloud system that is implemented using an existing IaaS system with empirical evidence on its performance in comparison with data center based cloud infrastructure.

The rest of the paper is organized as follows. Section 2 discusses cuCloud and its client/server architecture in detail. It also elaborates on the internal details of CloudStack together with the implementation of cuCloud using CloudStack as a base support. Section 3 presents empirical performance comparison between cuCloud and Amazon EC2 using a big-data application. Section 4 reviews related works in regard to volunteer cloud computing and Sect. 5 concludes the paper.

2 Design and Implmentation of cuCloud

cuCloud is a Volunteer Computing as a Service (VCaaS) system that runs over an existing computing infrastructure, which is not specifically setup for cloud purposes. The system thus can be considered as an opportunistic private/public cloud system that executes over scavenged resources of member PCs (also referred to as member nodes or volunteer hosts) within an organization (or community). Being non-dedicated, the PCs have other primary purposes, such as word processing, data entry, web browsing, etc. Only the residual resource capacity that is not used by the local processes is going to be extracted and utilized by cuCloud. This setting decides numerous appealing advantages of cuCloud: affordability, on-premise, self-provision, and greener computing. On the other hand, full-fledged implementation of cuCloud raises unique technical challenges: efficient management of highly dynamic and heterogeneous compute resources, QoS assurance, and security/trust, which are all made more difficult due to the high dynamic availability and heterogeneity of the non-dedicated volunteer hosts. The following subsections discuss the design and implementation issues of cuCloud in detail.

2.1 Design of cuCloud

The cuCloud adopts a client/server architecture with the volunteer hosts being the clients and the dedicated management machine(s) being the server(s). The server has various components as depicted in Fig. 1. The following are descriptions for each of the server side components.

– **Interface:** is the first port of communication between users and cuCloud. cuCloud has different types of users: *admin users*, *cloud users*, and *local users*. *Admin users* use the *Interface* to manage the cuCloud system. The *cloud users* (a.k.a clients or customers) specify their resource requirements and manage

the allocated resources via the *Interface*. *Local users* (native users) are the ones who contribute their spare computing resources to the cuCloud system and use the *Interface* to check their resource contribution details, credits earned, etc.

- **Authentication and Authorization:** component handles the user level authentication, authorization, and security issues of cuCloud. *Admin users* and *cloud users* should be preregistered and should be authenticated and authorized before start using the system.
- **Resource Manager and Allocator (RMA):** component is the one that manages all the volunteered computing resources possessed by cuCloud. It communicates with the cuCloud member nodes and periodically updates the availability of the resources from each member node. The component keeps the reliability and availability profiles of all volunteer hosts. The profiles together with the Service Level Agreement (SLA) are the basis of selecting volunteer hosts for deploying Virtual Machines (VMs) to best satisfy *cloud users'* resource requirements and cuCloud's system scheduling criteria.
- **Scheduler:** component receives *cloud users'* requests and decides whether to admit or deny the requests in consultation with the RMA and the Virtual Machine Manager components. The Scheduler makes the admission decision and allocation of VM(s) to host(s) based on the dynamically updated resource profiles maintained by the RMA. The Scheduler also handles the (re)allocation of VMs in cooperation with the Virtual Machine Manager.
- **Virtual Machine Manager (VMM):** component handles the deployment of VMs on volunteer hosts. More specifically, it performs the following tasks regarding VM management: create a VM, migrate (live and cold) a VM, suspend and resume a VM, kill and restart a VM, etc. The component also monitors all deployed VMs and keeps track of their progress information.
- **Security Module (SM):** component handles the security of the VMs deployed in cuCloud. There is no assumption of trust relationship between the volunteer hosts and the VMs in cuCloud. Therefore, the SM secures VMs from malicious volunteer hosts and vice versa.
- **Monitoring and Management:** component provides fine-grained information about the aggregate computing resources of the cuCloud. This includes, but not limited to, details of donated resource from each individual volunteer host, e.g., the total number of CPU cores, aggregated memory size, total storage capacity, used/available CPU cores, etc.

In the system architecture (Fig. 1) of the cuCloud, member nodes are subsidiary to the server. There is a critical piece of software, called Membership Controller (MC), that resides on each member node that contributes resources to the cuCloud system. Membership Controller monitors the resource utilization of processes at each volunteer node and decides the node's membership status, which is highly dynamic. It also decides the availability and reliability of the node based on the member's historical data and the current resource availability. The MC collects and sends information to the server about the types and quantities of the available resources (CPU, RAM, Hard Disk) for contribution

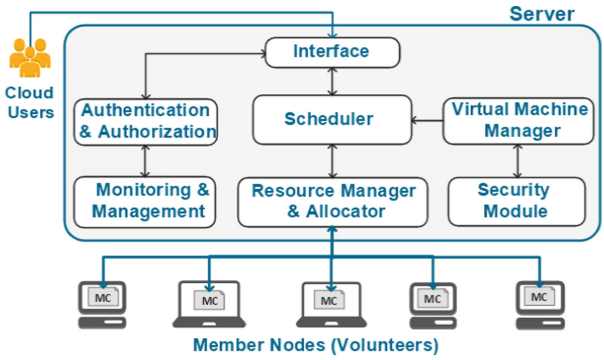

Fig. 1. System architecture of cuCloud

to the resource pool of cuCloud. It also allows the owner of a volunteer host to adjust the proportions of the node's resources for contribution to the cuCloud system. The MC has the following components (Fig. 2):

- **Sensor:** component periodically monitors the resource utilization of processes (both local and guest) on a volunteer host and sends that information to the *Node Profiler*. The total resource capacity of the host such as the number of CPU cores, RAM, and Hard Disk capacity as well as the used and idle resources are monitored periodically. The *Sensor* also collects other pieces of data that are used by the *Node Profiler*.
- **Node Profiler:** is the heart of the Membership Controller. It periodically accepts the resource utilization information from the *Sensor* and calculates the residual computing resource availability information. It also predicts the future availability and reliability of the volunteer host using the historical data it receives from the *Sensor*. The historical data includes time to failure, mean time between failures, mean time to repair, etc. This reliability and availability profile will be used by the RMA for further processing that may result in actions taken by the VMM, for example migration of VM(s).
- **Reporter:** The availability and reliability profile of a volunteer host is reported to the *Resource Manager and Allocator* of the server of cuCloud via the *Reporter*. The *Reporter* periodically accepts the profile information from the *Node Profiler* and pushes the information to the server of cuCloud.
- **Virtual Environment Monitor:** is the component that manages the VMs that are deployed on a volunteer node. It monitors the deployed VMs on a host and passes information regarding their progress and status to the *Node Profiler* for building the availability and reliability profile of the host.
- **Policy:** component stores the *native user's* preferences regarding donated resources, such as the time the resource should not be used, the number of cores or the RAM capacity donated, etc. This policy information is used by the *Node Profiler* for building the profile of the volunteer host.

The cuCloud, a "No Data Center" cloud system [8], is expected to fulfil all the characteristics of Cloud Computing systems such as elasticity, metered services, resource pooling, networked access, and automated on-demand service provisioning. Besides, service provisioning using non-dedicated, highly heterogeneous, and dynamically available volunteer hosts are idiosyncratic to cuCloud. The following are the design goals of the cuCloud.

- **Volunteer based:** Participation in cuCloud is purely voluntary. No host will be forced to join the virtual infrastructure base of cuCloud.
- **Unobtrusive:** The local processes of a volunteer host always have higher priority than the guest processes (cuCloud's VMs).
- **Scalable:** It should be scalable to tens of thousands of volunteer hosts.
- **Easy to use and deploy:** Usage and deployment of cuCloud should be as simple as possible.
- **Security:** The volunteer hosts should securely execute the deployed VMs without tampering them and vice versa.

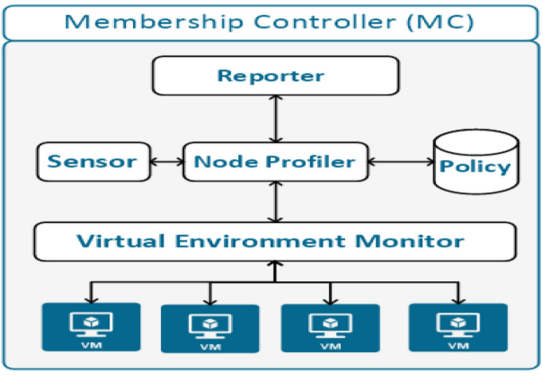

Fig. 2. Membership Controller of cuCloud

2.2 CloudStack Based Implementation of cuCloud

We chose CloudStack[3], an open-source Infrastructure-as-a-Service (IaaS) management platform, to implement cuCloud. Apache CloudStack manages and orchestrates pools of storage, network, and compute resources to build a public or private IaaS cloud. It is a cloud system that supports multiple hypervisors, high availability, complex networking, and VPN configurations in a multitenant environment. According to a survey [9], CloudStack has a 13% adoption rate as a private cloud solution for enterprises. CloudStack is chosen for the implementation of cuCloud because of its modular design, rich documentation, active online development community, and highly extensible architecture. By using

[3] http://cloudstack.apache.org/.

CloudStack for the implementation of cuCloud, we can make use of its rich set of functionalities such as Account Management, Virtual Machine Management, etc., with little or no modification. This will increase the adoption of cuCloud and avoid reinventing the wheel.

CloudStack follows a distributed client-server architecture where the Management Server(s) acts as a server and compute nodes act as clients. CloudStack Management Server can manage tens of thousands of physical servers (compute nodes) installed in geographically distributed datacenters. The Management Server (MS) communicates with the compute nodes through the hypervisors on the machines. Type I hypervisors such as Xen, Hyper-V, and VMWare are supported in CloudStack.

As depicted in the basic layered architecture of CloudStack Management Server (Fig. 3), the business logic is the one that accepts the user/admin requests and applies the necessary checks such as authentication or resource availability before sending it to the next layer. The Orchestration layer is responsible for configuring, provisioning, and scheduling any operations such as VM creation or storage allocation. Controllers are the one that directly communicate with the underlining computing resources such as computing units and networking devices, for the actual provisioning of resources. CloudStack is designed to be extensible in such a way that plugin APIs are provided for extending the functionality or modifying its behaviour [10]. Thus, a new plugin can be defined and integrated with CloudStack as needed.

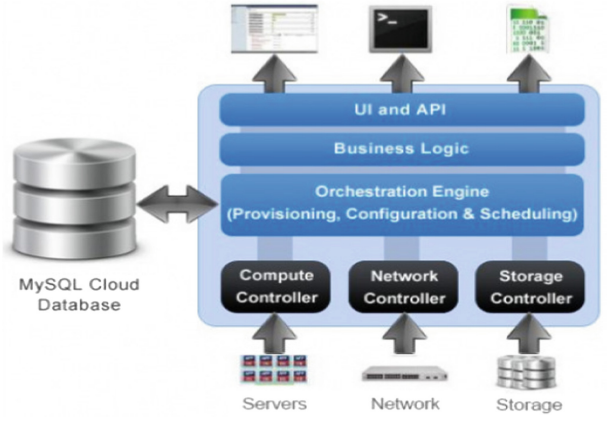

Fig. 3. Layered architecture of CloudStack [10]

CloudStack has *Resource Manager* component that is responsible for managing the compute, storage, and network resources. The addition of a new resource (compute node or storage) in CloudStack is manual, i.e., the admin should give the resource's attributes such as IP address, root password, hypervisor type, etc., via a web-based form. The resource information will then be stored in a MySQL based database. Since CloudStack is an IaaS system that is developed

to manage dedicated servers in data centers, we need to modify the Management Server of CloudStack in such a way that it can handle the non-dedicated volunteer nodes that we use as cloud infrastructure in cuCloud. This needs a fundamental change of the *Resource Manager* component of the MS for two reasons. First, the non-dedicated and high churn rate of the volunteer hosts coupled with the unobtrusive nature of cuCloud require an automatic addition of resources instead of the usual manual addition. Second, once a compute node is added manually to the CloudStack database, the MS pulls servers to check their availability and updates the status of the node in the database. This pull mechanism is not scalable for a highly dynamic resource base of cuCloud. Therefore, a push mechanism should be deployed in cuCloud. Volunteer hosts push availability and reliability information periodically to the MS and if the MS doesn't receive this information for a certain duration, it will mark the volunteer host as unavailable. A new plugin, called *AdHoc component*, is developed to handle the volunteer hosts in coordination with the *Resource Manager* of CloudStack.

The Membership Controller (MC) (see Fig. 2) on the volunteer hosts continuously monitors the resource utilization of processes on the host. This resource utilization information will be used, together with the policy information, by the *Node Profiler* to decide the availability and reliability of the volunteer host [16]. Based on this information, the MC sends "active" availability message to the *AdHoc component* of the CloudStack Management Server, if the volunteer node is available to host Virtual Machine(s), "inactive" otherwise. Algorithm 1 describes the actions the *AdHoc component* takes after receiving the message from the volunteer node.

The host (compute node) is the smallest organizational unit within CloudStack deployment. Every host should have a hypervisor such as Xen or KVM, which is abstracted as *ServerResource* in CloudStack. The Management Server cannot directly communicate with the hypervisor of a compute node, instead it communicates with *ServerResource* through the Agent. *ServerResource* is a translation layer between the hypervisor on a compute node and the commands of the Management Server. The *ServerResources* makes the inclusion of a new hypervisor support in CloudStack simple.

Due to the full self-autonomy of volunteer hosts and the non-intrusiveness of cuCloud, we cannot use type I hypervisors like Xen or Hyper-V that takes full control of the underline hardware of a volunteer host. Instead, we need a virtualization solution that runs along with local processes on member nodes. Therefore, for the virtualization environment at member nodes, we can use Type II hypervisors like VirtualBox or KVM. The current version of CloudStack (4.11) supports KVM but not VirtualBox. Adding a support for VirtualBox in CloudStack is one of the entries in the to-do list of cuCloud. With the inclusion of *AdHoc component* at the Management Sever and based on our client architecture, we extend CloudStack in such a way that hosts (compute nodes) can be added and removed from the resource pool dynamically, which is in high contrast with the dedicated nature of the compute nodes in the original CloudStack. Moreover, the VMs are now able to run along with non-cloud (local) tasks/processes on the volunteer hosts.

Algorithm 1. Member Node Addition Pseudo Code

```
1: message ← null;
   status ← null;
   message ← reciveFromHost();
2: if message is "active" then
3:    if host in DB of CloudStack then
4:       checkStatus();
5:       if status is "maintenance"  then
6:          status ← enabled;
7:       end if
8:    else
9:       addNewHost();
10:   end if
11: else
12:   if host in DB of CloudStack then
13:      checkStatus();
14:      if status is "enabled"  then
15:         status ← maintenance;
16:      end if
17:   end if
18: end if
```

3 Experimentation

In order to test the suitability of cuCloud for real applications, we run a big-data workload using Hadoop over the system. To gain a more detailed comparative performance, the same workload was run on Amazon EC2 with similar VMs specifications. We used BigDataBench[4] software for the benchmarking. The BigDataBench, which is a multi-discipline research and engineering effort from both industry and academia, is an open-source big data and AI benchmark [11]. The current version of BigDataBench provides 13 representative real-world data sets and 47 benchmarks. We used an offline analytics workload of PageRank calculation using Hadoop MapReduce. Three VMs were deployed on cuCloud for the experimentation (one master and 2 slaves). The master is deployed on a machine with 8 GB of RAM, Intel 8 Core i7 2.4 GHz CPU, and a hard disk of 250 GB. Each slave node has 8 GB of RAM, intel 4 Core i3 3.1 GHz CPU, and 250 GB hard disk capacity. All the machines run Ubunu 14.04, are connected to a 16 Gbps switch, and support intel hardware virtualization (VT-x). The experimentation for cuCloud was done in the computer labs at Southern Illinois University Carbondale. The PCs were being used by local users while the PageRank calculation was computed, i.e., they were not dedicated only for the cloud task. General-purpose instances with the same specifications as cuCloud VMs were configured on Amazon EC2. The availability zone of all the EC2 instances was us-east-2b of Ohio region. The details of the VMs for both the cuCloud and EC2 is given in Table 1.

[4] http://prof.ict.ac.cn/.

Table 1. VMs specification

	cuCloud			Amazon EC2		
Master	medium			t2.medium		
	CPU	RAM	HD	CPU	RAM	HD
	2	4	75	2	4	75
Slaves	small			t2.small		
	CPU	RAM	HD	CPU	RAM	HD
	1	2	50	1	2	50

The PageRank calculation was performed for different data sizes (numbers of nodes with the corresponding number of edges). Table 2 gives the dataset used in our experimentations. We run the experimentation for each data set 10 times at different time of the day and week (morning vs. evening and weekend vs. weekdays). Figure 4 depicts the average time required to run the application in both infrastructures.

Table 2. Experimental data size

	Nodes	Edges
Run1	16	21
Run2	256	462
Run3	65,536	214,270
Run4	1,048,576	4,610,034

The experimental results showed that cuCloud outperforms Amazon EC2 on the Hadoop MapReduce based PageRank calculation for all runs. This is mainly due to the comparative advantage of cuCloud over EC2 with regard to the network latency, as cuCloud runs over a LAN. The average round trip time (rtt) between the master and slaves for 10 packets is 10.5 ms for EC2 vs. 0.5 ms for cuCloud. Moreover, the runs on the general-purpose instances of EC2 showed a very large processing time variation. For instance, calculating the PageRank for Run3 took 352 min (during the afternoon) vs. 16 min (during early in the morning). On the other hand the processing times in the cuCloud were more or less uniform. Figure 5 depicts the variability of processing times of the 10 experiments performed on both systems for Run3 dataset. In general, the workloads deployed on EC2 run faster during night times and were slower during afternoons. From the preliminary experimental results we can conclude that for applications with a heavy inter-task network communications requirement like MapReduce, cuCloud performs well. This indicates that volunteer clouds have comparative advantages over public clouds, typically backed by remote data centers, in reducing the round trip time latency of applications. Therefore, volunteer

clouds are a perfect fit to the concept of edge computing since most applications naturally happen at the edge (of the Internet) and volunteer clouds can be most conveniently deployed to directly serve these applications – edge applications.

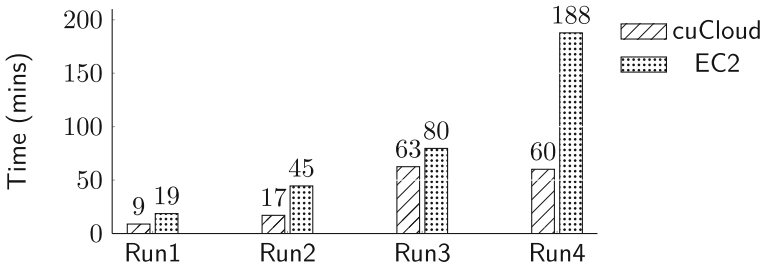

Fig. 4. Average processing time of workloads cuCloud vs. EC2

4 Related Works

The next generation of Cloud Computing needs fundamental extension approaches, to go beyond the capabilities of data centers. This means to include resources at the edge of the network or voluntarily donated computing resources, which are not considered in the existed conventional Cloud Computing model [6]. The concept of using edge resources for computing has been investigated in Cloud Computing as AdHoc Cloud [12], Volunteer Cloud Computing [13–15], and Opportunistic Cloud Computing [17].

In order to make use of and derive the benefit from the preexisting computing resources within an institution/organization scope, Rosales *et al.* built an opportunistic IaaS model through exploiting idle computing resources available in a university campus called UnaCloud [17]. The UnaCloud follows a client/server model, where the server hosts a web application that works as the main interface for all the UnaCloud services and the client is a lightweight application, based on the design concept of SETI@Home agent, installed on each node in the system. All the resources are homogeneous where each resource can handle one VM at a time in order to avoid resources competition. The UnaCloud doesn't provide any explanation on how to handle the associated challenges of non-dedicated computing resources such high churn rate, unreliability, and heterogeneity. On the other hand, the model lacks Cloud Computing basic features such as scalability, interoperability, and QoS. Another research work similar to cuCloud is the AdHoc Cloud Computing system proposed in [12]. The idea of AdHoc Cloud is to transform spare resource capacity from an infrastructure owned locally, but non-exclusive and unreliable, into an overlay cloud platform. The implemented client/server AdHoc cloud system is evaluated for its reliability and performance. In comparison with Amazon EC2, the author claimed

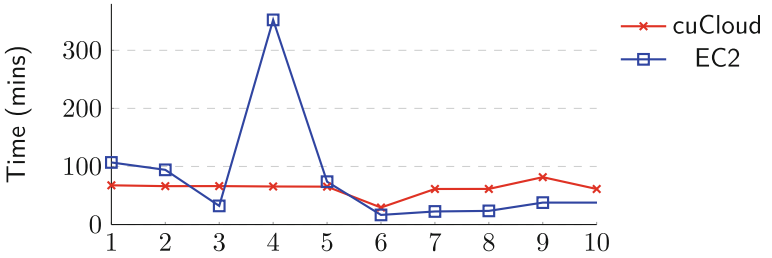

Fig. 5. Workload processing time variability cuCloud vs. EC2

a good comparable performance and concluded that AdHoc cloud is not only feasible, but also a viable alternative to the current data center based Cloud Computing systems. However, the authors mentioned nothing about the elasticity, multitenancy, etc. characteristics of the system. Nebula, proposes to use a widely distributed edge resources over the Internet [14]. It is a location and context-aware distributed cloud infrastructure that provides data storage and task computation. Nebula consists of volunteer nodes that donate their computation and storage resources, along with a set of global and application-specific services that are hosted on dedicated, stable nodes [13]. The authors used Planetlab based simulation to test their system. One typical characteristic of cuCloud that differentiates it from all the above-mentioned works is that by extending the already existing IaaS cloud system, it full-fills all the basic characteristics of a cloud system such as elasticity, multitenancy, etc.

Cloud@home is a volunteer cloud system that provides similar or higher computing capabilities than commercial providers' data centers, by grouping small computing resources from many single contributors [18]. In order to tackle the node churn problem that occurs due to the random and unpredictable join and leave of contributors, an OpenStack[5] based Cloud@home architecture is proposed [15]. Every contributing node has a *virtualization layer* and a *Node Manager* that handle the allocation, migration, and destruction of a virtual resource on the node. The authors detailed the mapping of the reference architecture of Cloud@home to Openstack as well as a possible technical implementation. Compare to our work, the proposed prototype is only a blueprint of implementing Cloud@Home where the authors gave no such evaluation or validation of the performance of the proposed system.

5 Conclusion

The cuCloud, an opportunistic private/public cloud system that executes over scavenged resources of member PCs within an organization (or community), is a cheaper and greener alternative Cloud Computing solution for organizations/communities. In this paper, we discussed cuCloud together with its architecture and its implementation based on the open-source IaaS CloudStack. We

[5] https://www.openstack.org/.

also described the architecture and relevant components of CloudStack with regard to cuCloud's implementation. The paper presented the empirical performance evidence of cuCloud in comparison with Amazon EC2 using a big-data application based on Hadoop. From the results of the experimentation, we concluded that cuCloud can handle big-data applications that need heavy communications well. However, we plan to do an experimentation at a larger scale and with different workloads as well as applications in order to give general conclusions. Moreover, the cuCloud system is still under development. In general, the cuCloud is a system that can be called a genuine volunteer cloud computing system, which manifests the concept of "Volunteer Computing as a Service" (VCaaS) that finds particular significance in edge computing and related applications. In a highly dynamic and heterogeneous resource environment assumed by cuCloud, research and re-investigation on scheduling algorithms, VM migration, QoS, Security, unobtrusiveness, resource management, etc. immediately find new meanings and momentum.

References

1. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud research problems in data center networks. SIGCOMM Comput. Commun. Rev. **39**(1), 68–73 (2009)
2. Brown, R.: Report to congress on server and data center energy efficiency: public law 109–431. Lawrence Berkeley National Laboratory (2008)
3. The Climate Group: SMART 2020: enabling the low carbon economy in the information age. The Climate Group on behalf of the Global eSustainability Initiative (2008)
4. Domingues, P., Marques, P., Silva, L.: Resource usage of windows computer laboratories. In: IEEE, pp. 469–476 (2005)
5. Che, D., Hou, W.-C.: A novel "Credit Union" model of cloud computing. In: Cherifi, H., Zain, J.M., El-Qawasmeh, E. (eds.) DICTAP 2011. CCIS, vol. 166, pp. 714–727. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21984-9_59
6. Varghese, B., Buyya, R.: Next generation cloud computing: new trends and research directions. Future Gener. Comput. Syst. **79**, 849–861 (2018). https://doi.org/10.1016/j.future.2017.09.020
7. Petcu, D., Fazio, M., Prodan, R., Zhao, Z., Rak, M.: On the next generations of infrastructure-as-a-services, pp. 320–326 (2016)
8. Mengistu, T., Alahmadi, A., Albuali, A., Alsenani, Y., Che, D.: A "No Data Center" solution to cloud computing. In: 10th IEEE International Conference on Cloud Computing, pp. 714–717 (2017)
9. RightScale: State of the Cloud Report (2016). https://www.rightscale.com/lp/2016-state-of-the-cloud-report
10. Sabharwal, N.: Apache Cloudstack Cloud Computing. Packt Publishing Ltd, Birmingham (2013)
11. Gao, W., Wang, L., Zhan, J., Luo, C., Zheng, D., Jia, Z., Xie, B., Zheng, C., Yang, Q., Wang, H.: A Dwarf-based Scalable Big Data Benchmarking Methodology. CoRR (2017)

12. McGilvary, G.A., Barker, A., Atkinson, M.: Ad hoc cloud computing. In: 2015 IEEE 8th International Conference on Cloud Computing (CLOUD), pp. 1063–1068 (2015)
13. Ryden, M., Chandra, A., Weissman, J.: Nebula: data intensive computing over widely distributed voluntary resources. Technical report (2013)
14. Chandra, A., Weissman, J.: Nebulas: using distributed voluntary resources to build clouds. In: Proceedings of the 2009 Conference on Hot Topics in Cloud Computing series, HotCloud 2009 (2009)
15. Distefano, S., Merlino, G., Puliafito, A.: An openstack-based implementation of a volunteer cloud. In: Celesti, A., Leitner, P. (eds.) ESOCC Workshops 2015. CCIS, vol. 567, pp. 389–403. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33313-7_30
16. Mengistu, T.M., Che, D., Alahmadi, A., Lu, S.: Semi-markov process based reliability and availability prediction for volunteer cloud systems. In: 11th IEEE International Conference on Cloud Computing (IEEE CLOUD 2018) (2018)
17. Rosales, E., Castro, H., Villamizar, M.: Unacloud: opportunistic cloud computing infrastructure as a service. In: Cloud Computing 2011: The Second International Conference on Cloud Computing, GRIDs, and Virtualization. IARIA, pp. 187–194 (2011)
18. Cunsolo, V.D., Distefano, S., Puliafito, A., Scarpa, M.: Cloud@Home: bridging the gap between volunteer and cloud computing. In: Huang, D.-S., Jo, K.-H., Lee, H.-H., Kang, H.-J., Bevilacqua, V. (eds.) ICIC 2009. LNCS, vol. 5754, pp. 423–432. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04070-2_48