'Time' for Cloud?
Design and Implementation of a Time-Based Cloud Resource Management System

Ryan K. L. Ko, Alan Y. S. Tan, Grace P. Y. Ng
Cyber Security Lab, Dept. of Computer Science
University of Waikato
Hamilton, New Zealand
Email: ryan@waikato.ac.nz, {yst1, pygn1}@students.waikato.ac.nz

***Abstract*—The current *pay-per-use* model adopted by public cloud service providers has influenced the perception on how a cloud should provide its resources to end-users, i.e. *on-demand* and *access to an unlimited amount of resources*. However, not all clouds are equal. While such provisioning models work for well-endowed public clouds, they may not always work well in private clouds with limited budget and resources such as research and education clouds. Private clouds also stand to be impacted greatly by issues such as user resource hogging and the misuse of resources for nefarious activities. These problems are usually caused by challenges such as (1) limited physical servers/ budget, (2) growing number of users and (3) the inability to gracefully and automatically relinquish resources from inactive users.**

Currently, cloud resource management frameworks used for private cloud setups, such as OpenStack and CloudStack, only uses the *pay-per-use* model as the basis when provisioning resources to users. In this paper, we propose OpenStack Café, a novel methodology adopting the concepts of 'time' and booking systems' to manage resources of private clouds. By allowing users to book resources over specific time-slots, our proposed solution can efficiently and automatically help administrators manage users' access to resource; addressing the issue of resource hogging and gracefully relinquish resources back to the pool in resource-constrained private cloud setups. Work is currently in progress to adopt Café into OpenStack as a feature, and results of our prototype show promises. We also present some insights to lessons learnt during the design and implementation of our proposed methodology in this paper.

***Keywords*-Cloud computing; OpenStack Café; user-centric; time-based cloud resource management; cloud provisioning; cloud management systems; private clouds; resource hogging;**

I. Introduction

A. 'Time' for a Rethink

If one observes the spectrum of cloud computing [1] provisioning models today, there is a high chance that he or she will repeatedly come in touch with the traditional pay-per-use model [2]. This is especially so in public cloud service providers such as Amazon Web Services, Rackspace, GoGrid, etc, and it has undoubtedly influenced common perception on how a cloud should provide its resources to end-users: *on-demand* and *access to unlimited amount of resources* (if you can pay for the usage). However, this model hinges on a critical assumption: the assumption that the cloud service provider has the ability and sufficient underlying hardware resources to support a potentially ever-growing demand.

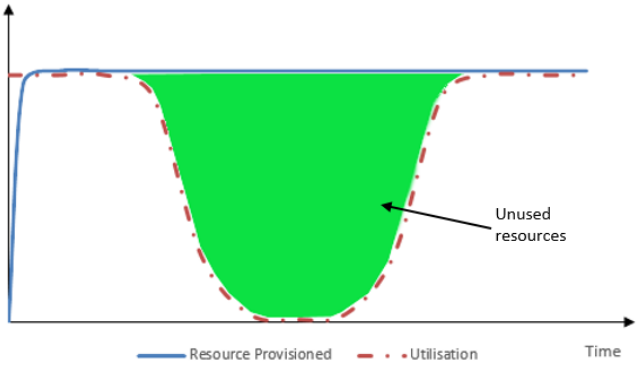

Figure 1. Illustration of potential resource wastage scenario

This assumption does not usually hold true for private clouds, which may be hosted by institutions whose core business is not in providing cloud services. Often, private clouds such as the ones used in research and education, face limitations such as budget and resources. With a limited physical resource pool, a growing number of users and an inability to gracefully and automatically relinquish resources from inactive users, private clouds stand to be impacted greatly by issues such as user resource hogging and the nefarious misuse of resources. Worse, current cloud resource management frameworks used in private cloud setups, such as OpenStack [3] and CloudStack [4], only uses the pay-per-use model as the basis when provisioning resources to users.

Data reported by Chaisiri *et al.* [5] and Vogels [6] have shown that typical actual demands for computing resources do not sustain over long period of times, even in well established clouds or high performance servers such as those in Amazon and Google. Users often request for access to resources for a time period longer than the actual required amount of time needed for their experiments. One such scenario is when researchers request for compute resources to run their experiments. The normal behaviour is to run their experiments, obtain and analyse their results before running their experiments again. During the period of result

analysing, the resources are usually left idling. Figure 1 illustrates such typical resource utilisation versus resource provisioned behaviour. In private clouds where resources are limited, the unused resources during the analysis period can be put to better use: serving other users' demand for resources. Hence, a solution that can automatically relinquish unused resources gracefully is needed to tackle the issue of resource hogging.

B. Objectives and Contributions of Paper

In this paper, we propose a novel concept of 'time' in managing resources provisioned to users. We also demonstrate its feasibility and user-centricity through our prototype, Café, which has been acknowledged by the global community and will be integrated into the OpenStack open source project.

Just like an Internet café business model, users of OpenStack Café can request from cloud administrators resource usage for a predefined period of time. When the time is up, the resources booked will be relinquished and returned to the resource pool. Our proposed solution can efficiently and automatically help administrators manage users' access to resource by specifying fixed time-slots.

The main contribution of Café addresses the issue of resource hogging in resource-constrained private cloud setups. An overview of the list of contributions of our proposed methodology include:

- A time-slot approach to manage user access to resources.
- An automatic resource clearing mechanism for cloud resource management frameworks like OpenStack.
- A novel user access management tools for cloud services (which contrasts typical admin-centered management tools).
- User access based on a "request and use model" where users request/purchase timeslots for accessing resources.
- Eases administrators' need to keep track and plan the capacity of the cloud infrastructure [7].
- Flexibility on the users' part as they can schedule their usage and monitor their usage too.
- On the administrator part, Café is useful in keeping track of both current and future utilisation rate of the cloud.
- Helps in resource planning, in terms of expansion and maintenance.
- The system works automatically, hence reducing the effort required to monitor and maintain the infrastructure's capacity.

In short, Café ensures fair sharing of resources, and is very suitable for education and research environments. The implementation is also a contribution to the OpenStack global community. In the course of developing this, we also received feedback mentioning that this model would be relevant and useful for research clouds such as those in CERN and NECTaR. Insights to lessons learnt during the design and implementation of the Café prototype on OpenStack will be presented in Section III and IV.

II. Related Work

Management of user access to resources and resource of a computing system is a well-studied field in distributed systems [8] (e.g.grid systems). However, many focus on *resource allocation* and not *resource deallocation* (for higher utility of the cloud). Frameworks such as Globus toolkit [9] and Grid Cafe [10] help connect distributed resources from systems across the world and allowing users to access them through a single interface. While these systems use various established methods, including time-based methods, in managing resources and user access to the underlying resources, they are based on the concept of grid systems.

It is perhaps common knowledge that grid systems lack the flexibility and scalability found in cloud systems. As compared to grid users, cloud users are able to modify their setup (e.g. install new operating systems, modify applications and etc) without affecting other users and even easily scale up or down the number of 'machines' required so as to suit their needs, through virtualisation, the foundation of cloud systems. The flexibility and scalability offered by cloud systems are exactly the traits we required to cater to our needs for a system that caters to our user base: academic researchers and students in a educational institution.

Since cloud computing became popular from 2006 with the introduction of Amazon's Elastic Compute Cloud, EC2, many cloud resource management frameworks and services since been made available to users on the Internet, both commercially and non-commercially. On the commercial side: Google Cloud [11, 12], Amazon EC2 [13], Microsoft Azure [14], VMWare vSphere [15] and Citrix XenServer [16]; while on the non-commercial side: Open CIRRUS [17], OpenStack [3], Cloudstack [4], Eucalyptus [18] and OpenNebula [19], are some examples of cloud resource management frameworks.

As our cloud testbed needed to cater to computing resource demands from researchers and students in an educational institution. Hence, in the context of our work, the evaluation of cloud resource management frameworks is done based on the following criteria:

- *Cost of using the framework* - it is to our benefit to reduce cost as much as possible as we work in the context of a non-profit educational institution.
- *Scalability of virtual machines (VMs)* - whether the framework can support a large number of concurrently running VMs.
- *Access to physical server* - this is a requirement as our research work may sometimes require deployment of softwares onto the physical host server.

Table I
COMPARISON OF CLOUD RESOURCE MANAGEMENT FRAMEWORKS

Framework	Cost Model	Highly Scalable (>1000 VMs)	Physical machine access	Customisable	Time-centric user access management features
Google Compute	Per unit	✓	✗	✗	✗
Amazon EC2	Per unit	✓	✗	✗	no full features
Microsoft Azure	Per unit	✓	✗	✗	✗
VMware vSphere	Annual Licence	✓	✓	✗	✗
Citrix XenServer	Annual Licence	✓	✓	✗	✗
Open CIRRUS	Free	✓	✗	✓	✗
OpenStack	Free	✓	✓	✓	✗
CloudStack	Free	✓	✓	✓	✗
Eucalyptus	Free	✓	✓	✓	no full features
OpenNebula	Free	✓	✓	✓	✗

- *Customisable* - in terms of source code. this is similar to having access to physical server, our work may require modification to the management framework.
- *A time-centric model for user access management* - as mentioned earlier, our user base consists of groups of users with different requirements. Researchers typically requires long term access to computing resources, for conducting their experiments. On the other hand, students only require short term access to computing resources. As our servers have finite amount of resources, it is crucial for us to limit access to users only to the time period in which they require the use of computing resources.

We compared the various cloud resource management frameworks in Table I based on the criteria laid out above. Our review of these frameworks revealed the **absence** of a time-centric user access control features in current day cloud resource management frameworks. Out of the ten cloud solutions surveyed, only Amazon EC2 and Eucalyptus provided some form of time-based resource access control but neither provided the full functionality required for this project due to not having support for administrative approval of user access requests [20, 21].

OpenStack is also currently developing a resource reservation feature called Climate which has a similar concept to Café. Climate has a different focus however as it focuses on creating the infrastructure upon which more specific capacity leasing services can be built and has resource-based access control [22].

III. REQUIREMENTS OF A TIME-BASED USER ACCESS MANAGEMENT

In this section, we first present the requirements and rationale for our system. As highlighted in Section I and II, unlike in public clouds, private clouds do not assume unlimited amount of resources. Hence, resource capacity management of cloud infrastructure is crucial [7], especially if there is a large user base. Inadequate capacity management can lead to failure in meeting users' expectation, in terms of performance, or in some cases, impact other applications and users in the same environment.

Our private cloud testbed is setup to cater primarily to research and education purposes. Hence it has to handle groups of users with different requirements, be it time of access or amount of resources required. For example, researchers require flexible on-demand access to their VMs, while students only access their VMs during their laboratory sessions or project sessions. While cloud frameworks such as OpenStack [3] are designed for virtual resource provisioning, these frameworks lack the ability to control user's access to resources automatically (i.e. grant access to users for a period of time only). Also, the system should be able to handle large number of users. With these considerations, we designed our proposed resource management system, to be usable with any cloud setup with finite resources, with the following requirements:

1) Ability to handle multiple groups of users, with each group having different access requirements. (e.g. time of access, period of access, resource quota)
2) Automatically manage user's access to resources.
3) Infrastructure resource capacity management.
4) Automatic management of unused resources.

Based on the above, we proposed a time-based resource management system. Much like booking a tennis court or renting a PC for web surfing in an Internet café, our proposed system uses a time-slot approach in managing user's access to resources. The basic idea is:

1) Users book a 'time-slot' through the management system to indicate the desired session and duration of access to virtual resource.
2) Administrator approves booking request and assigns a resource quota to user.
3) Upon the scheduled time, system automatically grants user access to the virtual resources. User can then freely provision VMs, within their assigned resource quota.
4) Once the user session ends, system automatically revokes user access rights to the virtual resources and frees up the resources used by the user.

By requiring the administrator to approve booking requests,

it enables the administrator to keep track of the utilisation rate of the cloud infrastructure.

Using a time-slot approach in managing user access helps to ease user access management. Firstly, it helps predefine the duration in which users have access to the virtual resources. This help ensure fair-share usage of resources across multiple groups of users. While users can book consecutive time-slots, bookings are subject to administrator approvals. Secondly, by having a fix predefined time period, it helps ease management of user sessions from the system perspective. (e.g. A fixed duration for time slots allow regular polling to be implemented easily. Allowing users to specify their own time period is more complex, from an implementation perspective.) Thirdly, confining user access to fixed time-slots help prevent resource hogging in private cloud. By having a clear and predefined end to user access duration, the system can automatically 'terminate' users' access to the virtual resources. Lastly, a time-slot approach naturally involves resource reservations. This helps administrators in resource utilisation monitoring and planning. It also gives the administration an idea of the current and future utilisation rate of the infrastructure.

IV. Design and Implementation of OpenStack Café

In this section, we present a conceptual model of our proposed time-based user access management system in Section IV-A, followed by our implementation of the model on OpenStack [3], an open source cloud resource management framework in Section IV-B.

A. Conceptual Model

Our proposed conceptual model is illustrated in Figure 2. An additional layer for managing users' access to virtual resources in a time-based manner is built on top of a cloud resource management frameworks. The user interface acts as the portal between users and the management system. Users perform tasks such as booking time-slots to indicate the period of access desired, view or edit booking details and delete bookings through the user interface. The interface also acts as a portal where administrators can monitor the current and future utilisation rate of the cloud infrastructure; by looking at the amount of time-slots and resources being reserved. Administrators also approve and reject user requests, set user resource quota limits and perform other administrative functions here.

The booking manager illustrated in Figure 2 serves as the main component in the model. Its main role is to coordinate users' access to the underlying cloud resource management framework. The booking manager retrieves detail of booking requests made by users and ensures that users can gain access only at the stipulated approved time-slots. This includes granting and removing permission to access the underlying virtual resources from the users.

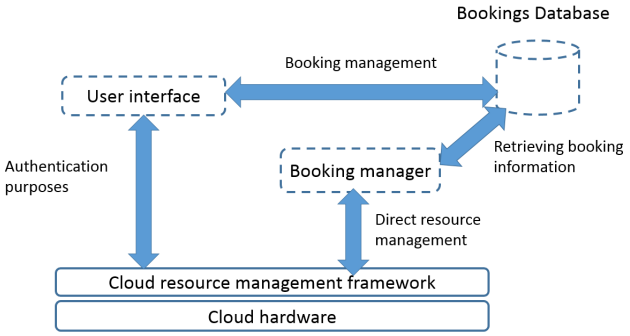

Figure 2. Illustration of the conceptual model

The novelty of our proposed system is not in the model itself, but rather the concept of a time-based approach to managing cloud resources in clouds. This was explained in Section III.

B. Prototype

We implemented a prototype of our time-based resource management system on OpenStack. The OpenStack version used was *Grizzly*. Figure 3 below illustrates how our prototype system, Café, interacts with the underlying OpenStack, which acts as the cloud resource management framework. Café components are denoted by a solid border while original OpenStack components are denoted with a dash border.

In line with the service oriented architecture of OpenStack, we implemented another additional module, the resource API, to allow service orchestration. (e.g. other services or modules of OpenStack can also access modules in Café through the APIs.)

Users first book through *Waiter*, Café's web interface, the time-slots that they desire for accessing virtual resources. Administrators are then notified by *Waiter* and the request is either approved or rejected. In our prototype, we managed to link *Waiter*'s interface to *Horizon*, OpenStack's web interface, so that users and administrators can switch between the two interfaces seamlessly.

Using the API module, *Barista*, *Waiter* updates user's booking details into the database. From this database, Café's main management engine, *Manager*, will retrieve booking details at regular intervals. The time between intervals is synchronised with the time period allocated for each time slot (i.e. If one time slot is one hour, *Manager* will retrieve booking details from the database at an hourly interval). Based on booking details, *Manager* will revoke or grant users' rights to access the virtual resources managed by OpenStack. This is achieved though user access-list managed by OpenStack's authentication service, *Keystone*. *Manager* is also in-charge of suspending all running VMs belonging to user whose session has ended. This prevents users from accessing their resources even without using the web interface (e.g. secure-shell access to VMs). Suspending running

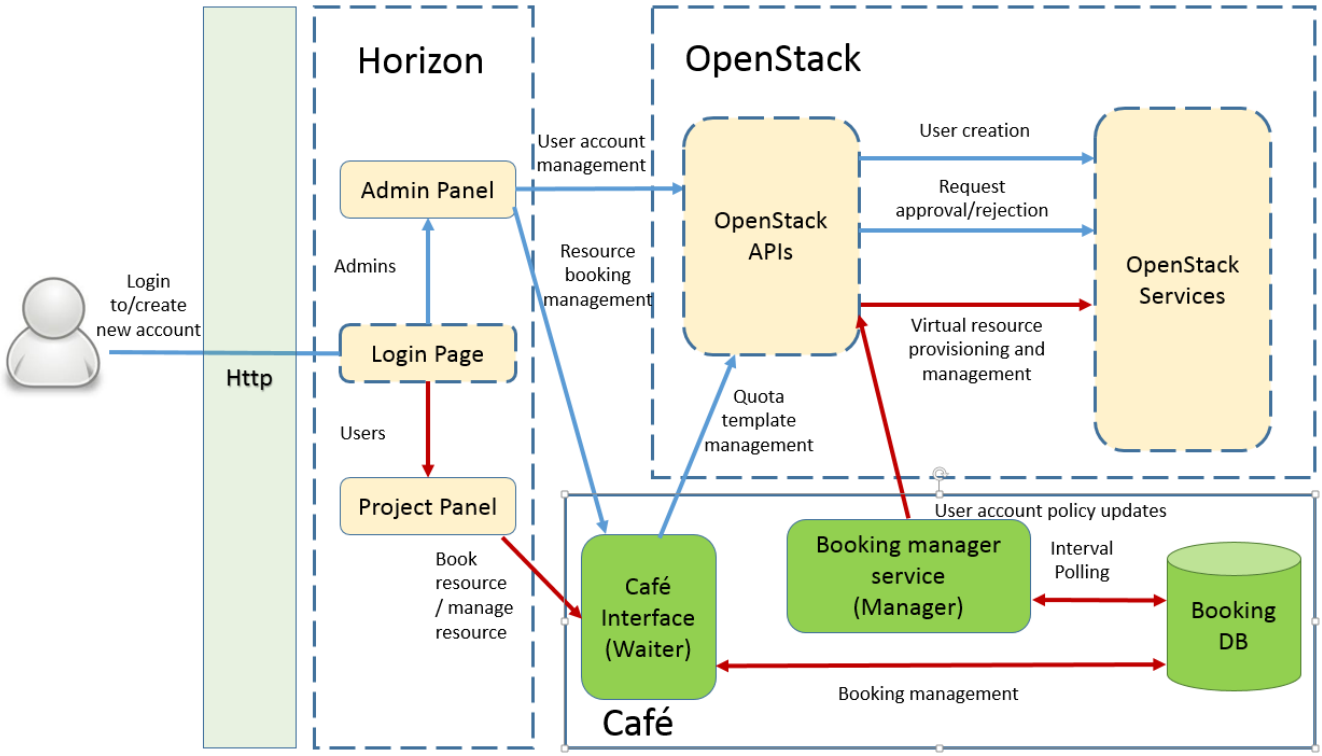

Figure 3. Interaction between OpenStack modules and our time-based resource management system - Café

VMs also frees up resources for the next batch of users and prevents hogging of unused resources.

For more details on Café's implementation, we invite readers to visit Café's wiki page at https://wiki.openstack.org/wiki/Café [23].

V. Discussions and Lessons Learned

During Café's design and implementation, a few lessons and issues surfaced:

- *Not all Clouds have infinite resources* - While public clouds are perceived as having 'infinite' resources (i.e. one can keep requesting for resources from service providers such as Dropbox or Amazon's EC2), private clouds are often constrained by the amount of investments from their owners (e.g. the amount of money invested by companies into purchasing hardware for their cloud infrastructure). This issue was the main problem that motivated this project. Resource management between different groups of user with different needs: students that require short term and at different time period of access and researchers whom requires long term access to resources.
- *Automation of resource allocation and deallocation is required for a large number of users* - In our case, the user base for our testbed was expected to be large (i.e. greater than 1000). This makes managing users a laborious task for the administrators. Imagine having to grant and remove permissions from 40-50 students every hour! We designed the system to automatically manage user access to virtual resources. This frees up the manual labour time required from the administrator for other tasks, such as monitoring and maintaining the health of the infrastructure.

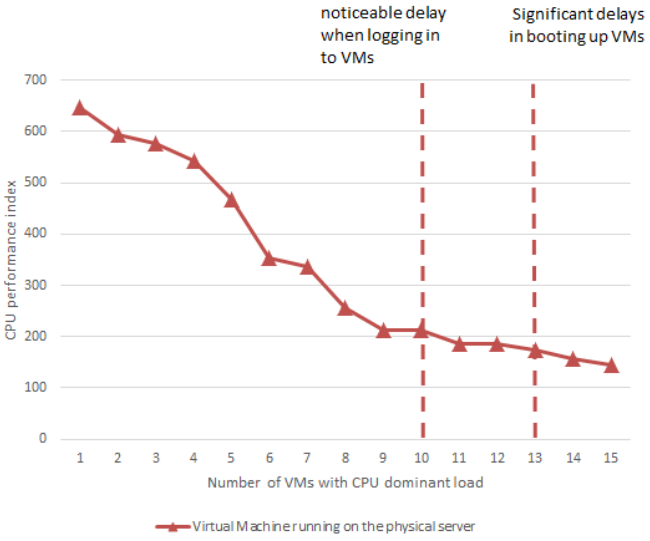

Figure 4. Benchmark results on VMs and physical host server when number of CPU intensive VMs running on the host server is increased

Our initial testing of Café also showed that the man-

agement of user access, on top of managing users of such magnitude is an extremely time consuming task. (e.g. account creation for a whole batch of new students and staff) This was simplified later on by getting Café to leverage on OpenStack's authentication service, *Keystone*, for account creation. When it comes to designing usable systems, one should also consider the size of the system's user base.

- *Many institutions are facing similar issues* - During the development of Café, we were in contact with fellow OpenStack developers. They have also brought to our notice that institutions such as NECTaR [24] and CERN [25] might also benefit from such a time-based resource management system. This spurred us to put developing Café as an independent resource management system for cloud resource management frameworks in our future work pipeline.
- *Workload on VMs are also important* - We are currently testing and overloading our cloud testbed with different kinds of workload on VMs. The aim is to understand how our testbed would perform under stress from different types of workload (e.g. CPU intensive or disk I/O intensive workloads.). Our initial experimental results showed that intensive workloads running on VMs can also affect user experience and performance on other VMs that are hosted on the same cloud infrastructure. Figure 4 shows one of the results from our load testing experiment. In that experiment, we increased the number of VMs running CPU intensive workloads and benchmark the CPU performance on both VM running on the physical server hosting the VMs and on the physical server. The performance of the virtual CPU on VMs can be seen to decrease rapidly as the number of VMs with CPU intensive workload increases.

VI. Future work

While Café addresses several issues, it still has room for further work.

Firstly, we can further expand the range/ type of resources managed by Café. Currently, Café manages only CPU and RAM resources. Control of other resources such as storage, are not implemented yet. The end result of Café should be a system that can fully automate the management of user access to all underlying virtual resources made accessible to user by the deployed cloud resource management frameworks (e.g. OpenStack).

Secondly, we can further optimise user bookings and resource capacity management with automated scheduling and planning. Current implementation of Café requires administrator to manually determine whether the amount of resources user requests for, can be met based on other users' bookings (capacity level of resources) at the stated time slot. However, this can be a taxing task as the number of request increases. Hence, we plan to investigate on the use of schedulers for automated resource planning and scheduling, based on knowledge of the resource capacity of the infrastructure and the bookings that have been made. Resource planning and scheduling is a well-studied field in both distributed systems [8] and cloud computing [26]. We believe that by automating resource planning and allocation, we can reduce the workload required to manage user requests.

Thirdly, a mechanism that allows administrators to monitor the utilisation rate of booked resources, is required to better address the issue of resource hogging. While the time-slot approach adopted by Café prevents user from hogging resources for an infinite amount of time, it does not prevent users from booking resources but not use them. Mechanisms, such as Ganglia [27] and Mochi [28], that monitor and report the utilisation of resources in cloud infrastructures, is required to help administrators identify users who booked but not utilise cloud resources. Once identified, administrators can choose to either revoke the resources assigned to those users or to approach them to discuss alternate booking plans.

Lastly, we believe that real-time workload and capacity monitoring is required [7]. We are beginning work on proposing a utility meter, which takes into consideration the different types of workload running on VMs, the capacity of each resource type on the cloud infrastructure and etc information, and reports an estimate of how many VMs can the infrastructure support before user experience can be affected.

VII. Concluding Remarks

In this paper, we proposed a novel concept of using 'time' in managing resources provisioned to users. Just like an Internet café business model, users of OpenStack Café can request from cloud administrators resource usage for a predefined period of time. When the time is up, the resources booked will be relinquished and returned to the resource pool. Our proposed solution can efficiently and automatically help administrators manage users' access to resource by specifying fixed time-slots. Such elegant resource deallocation reduces wastage/ idling of resources. This is a situation which would benefit most private clouds. We also compared the various cloud resource management frameworks and revealed their current inabilities to meet the actual needs of private cloud resource management. Our review of these frameworks revealed the absence of a time-centric user access control features in current day cloud resource management frameworks. Café was designed to address these gaps.

The main contribution of Café addresses the issue of resource hogging and potential 'unlimited' nefarious usage of allocated cloud resources in typical resource-constrained private cloud setups. We demonstrated our proposed concept's feasibility and user-centricity through our prototype,

Café, which has been acknowledged by the OpenStack community and is on track to being integrated into the OpenStack Project. Details of Café can be found at the OpenStack wiki pages: https://wiki.openstack.org/wiki/Café [23].

ACKNOWLEDGMENT

The authors would like to acknowledge the University of Waikato's Faculty of Computing and Mathematical Sciences CAPEX 2013 Fund for supporting our large-scale cloud test bed: Cloud8 (https://www.crow.org.nz/projects_cloud8.html) and this project. We would also like to thank Brad Cowie, Clint Dilks, Mike Vallabh, Raja Naeem Akram and Geoff Holmes for their advice during the setup of Cloud8.

REFERENCES

[1] R. K. L. Ko, "Cloud computing in plain English," *ACM Crossroads*, vol. 16, no. 3, pp. 5–6, 2010.

[2] C. Verstraete, "What really is pay-per-use in cloud?" urlhttp://h30499.www3.hp.com/t5/Grounded-in-the-Cloud/What-really-is-pay-per-use-in-cloud/ba-p/2407093#.UsykmfQW0vk (Accessed: 8/01/14).

[3] OpenStack, "Openstack - Open source software for building private and public clouds," http://www.openstack.org/ (Accessed: 17/12/12013).

[4] Apache CloudStack Community, "Apache CloudStack," http://cloudstack.apache.org/ (Accessed: 19/12/2013).

[5] S. Chaisiri, R. Kaewpuang, B.-S. Lee, and D. Niyato, "Cost Minimization for Provisioning Virtual Servers in Amazon Elastic Compute Cloud," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*, 2011, pp. 85–95.

[6] W. Vogels, "Beyond Server Consolidation," *Queue*, vol. 6, no. 1, pp. 20–26, 2008.

[7] B. Semple, "Five Capacity Management Challenges for Private Clouds," http://datacenterpost.com/2011/07/five-capacity-management-challenges-for.html(Accessed: 16/12/2013), July 2011.

[8] B. Neuman and S. Rao, "Resource Management for Distributed Parallel Systems," in *Proceedings of 2nd International Symposium on High Performance Distributed Computing*, 1993.

[9] I. Foster and C. Kesselman, "Globus: a Metacomputing Infrastructure Toolkit," *International Journal of High Performance Computing Applications*, vol. 11, pp. 115–128, 1997.

[10] "Grid Cafe," http://www.gridcafe.org/EN/globus-toolkit.html (Accessed: 19/12/2013).

[11] "Google Cloud Storage Access Control," https://developers.google.com/storage/docs/accesscontrol (Accessed: 19/12/2013).

[12] "Google Compute Engine," https://cloud.google.com/products/compute-engine (Accessed: 19/12/2013).

[13] "AWS Cloud Formation," http://aws.amazon.com/cloudformation/(Accessed: 19/12/2013).

[14] "Windows Azure," http://www.windowsazure.com/en-us/ (Accessed: 19/12/2013).

[15] "VMware vSphere," http://www.vmware.com/products/vsphere/(Accessed:19/12/2013).

[16] "Citrix XenServer," http://www.citrix.com/products/xenserver/overview.html(Accessed: 19/12/2013).

[17] A. Avetisyan, R. Campbell, I. Gupta, M. Heath, S. Ko, G. Ganger, M. Kozuch, D. O'Hallaron, M. Kunze, T. Kwan, K. Lai, M. Lyons, D. Milojicic, H. Y. Lee, Y. C. Soh, N. K. Ming, J.-Y. Luke, and H. Namgoong, "Open Cirrus: A Global Cloud Computing Testbed," *Computer*, vol. 43, no. 4, pp. 35–43, 2010.

[18] "Eucalyptus: Open Source Private Cloud Software," http://www.eucalyptus.com/eucalyptus-cloud/iaas (Accessed: 19/12/2013).

[19] "OpenNebula," http://opennebula.org/about:keyfeatures (Accessed: 19/12/2013).

[20] "AWS Identity and Access Management," http://docs.aws.amazon.com/IAM/latest/UserGuide/AccessPolicyLanguage_EvaluationLogic.html\#policy-eval-reqcontext (Accessed: 19/12/2013).

[21] "Eucalyptus - Sample policies," http://www.eucalyptus.com/docs/eucalyptus/3.2/ag/policies_samples.html\#policies_samples (Accessed: 19/12/2013).

[22] J. Danjou, "Announcing Climate, the Openstack capacity leasing project," http://julien.danjou.info/blog/2013/openstack-climate-capacity-leasing (Accessed: 19/12/2013).

[23] "Cafe - a time-based cloud resource management system for Openstack," https://wiki.openstack.org/wiki/Cafe (Accessed: 18/12/2013).

[24] "Known issues at NECTAR research cloud," https://support.rc.nectar.org.au/support/known_issues.html (Accessed:18/12/2013).

[25] A. Tselishchev, P. Tedesco, E. Ormancey, and C. Isnard, "CERN Computing Resources Lifecycle Management," *Journal of Physics:Conference Series*, vol. 331, 2011.

[26] R. Pal and P. Hui, "Economic models for cloud service markets: Pricing and capacity planning," *Theoretical Computer Science, Journal, Elsevier*, vol. 496, pp. 113–124, 2013.

[27] B. University of California, "Ganglia Monitoring System," http://ganglia.info/ (Accessed: 12/4/2014).

[28] J. Tan, X. Pan, S. Kavulya, R. Gandhi, and P. Narasimhan, "Mochi: Visual Log-analysis Based Tools for Debugging Hadoop," in *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, ser. HotCloud'09. Berkeley, CA, USA: USENIX Association, 2009.