2011 Seventh IEEE International Conference on eScience

# Collaborative eResearch in a Social Cloud

Ashfag M. Thaufeeg
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Email: Ashfag.Thaufeeg@ecs.vuw.ac.nz

Kris Bubendorfer
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Email: Kris.Bubendorfer@ecs.vuw.ac.nz

Kyle Chard
Computation Institute,
University of Chicago and
Argonne National Laboratory,
Chicago, IL, USA.
E-mail: kyle@ci.uchicago.edu

***Abstract*—Social networks provide a useful basis for enabling collaboration among groups of individuals. This is applicable not only to social communities but also to the scientific community. Already scientists are leveraging social networking concepts in projects to form groups, share information and communicate with their peers. For scientific projects which require large computing resources, one useful aspect of collaboration is the sharing of computing resources among project members. A social network provides an ideal platform to share these resources. This paper introduces a framework for Social Cloud computing with a view towards collaboration and resource sharing within a scientific community. The architecture of a Social Cloud, where individuals or institutions contribute the capacity of their computing resources by means of Virtual Machines leased through the social network, is outlined. Members of the Social Cloud can contribute, request, and use Virtual Machines from other members, as well as form Virtual Organizations among groups of members.**

***Index Terms*—Social Cloud, Social Networks, Cloud Computing, Service Computing**

## I. Introduction

Social networking has enabled new ways for individuals to communicate and share information. For scientists, multi-institute collaboration is common, however communication is often difficult with face to face meetings occurring only sporadically at conferences and workshops. Social networks can provide a greater level of scientific collaboration to increase communication and also to facilitate discovery of other scientists working on similar projects. However, scientific collaborations also often have resources they wish to share dynamically in groups for the duration of a project. Currently this is a difficult process requiring manual (reciprocal) user account registration, creation etc.

Our previous work introduced the concept of a Social Cloud [1], [2] as a means of facilitating multiparty resource sharing within the context of the relationships and policies represented in a social network. A Social Cloud is unique in that it is completely open and adhoc – it is created, controlled and managed by its users and therefore requires only a lightweight infrastructure. In this paper we propose extending the Social Cloud to support scientific collaboration through virtualised resource sharing. We believe that a cloud architecture integrated with a social network would benefit the scientific community in the following ways:

1) It allows scientists to share resources for the duration of collaboration. Therefore allowing scientists without sufficient computing resources to request them from other members of their Social Cloud. In addition other parties, even if they are not part of a specific research group, can also contribute computing resources via the trust enabled through the social network.
2) A social network allows more efficient use of available resources, as scientists with similar projects or interests would be encouraged to consolidate resources towards their common goals. This, in turn, contributes towards lowering the cost of scientific research.
3) Most importantly, it promotes greater collaboration among the scientific community by utilizing familiar tools, publicity and networking opportunities provided by the social network. In this way the social network is not just an add on, it is in fact a primary function of the Social Cloud.

To capture these benefits this paper presents an extended Social Cloud architecture and implementation tailored for the scientific community. The Social Cloud has been re-architected to provide a Platform as a Service (PaaS) model of computing by sharing virtualised physical resources represented as Virtual Machines. In this model scientists contribute resources and information (posts) to form virtual research environments for the duration of collaboration. We are calling this the *Social Collaborative Cloud* or SoCC. The prototype SoCC has been implemented as a Facebook application and includes VM scheduling and a VM image store.

By adopting the PaaS model in this first iteration we are targeting scientific researchers who would prefer access to low level computing resources to run computation intensive algorithms and process large data sets. In the future, we envisage the Social Cloud to be extended to support Software as a Service (SaaS) model as well. This would enable capabilities in the Social Cloud such as quickly deploying a web server to host a web site, or deploying a storage service to share files. A SaaS model would likely make the Social Cloud more attractive to researchers in non-scientific disciplines.

The remainder of this paper is organised as follows: section II outlines related work, section III describes the Social Cloud model, section IV defines the requirements of a collaborative social scientific computing environment, section V describes the architecture of the prototype SoCC, section VI presents the cost of provisioning VMs in this architecture,

978-0-7695-4597-4/11 $26.00 © 2011 IEEE
DOI 10.1109/eScience.2011.39



IEEE computer society

section VII outlines our planned future work and finally section VIII concludes this paper.

## II. Related Work

The value of social networking has been observed in multiple scientific domains as a means of facilitating collaboration. Increasingly social networks are being used to coordinate research communities, two such examples are MyExperiment [3] for biologists and nanoHub [4] for the nanoscience community. MyExperiment provides a virtual research environment where collaborators can share and execute scientific workflows. nanoHub allows users to share data and transparently execute applications on distributed resource providers such as TeraGrid. While similar to a Social Cloud, MyExperiment and nanoHub each have specific sharing focuses and build their own proprietary social network stack. GlobusOnline [5] takes a different focus by aiming to abstract complex scientific processes through a Software as a Service (SaaS) model. At present it supports reliable file transfers, and work is underway to implement a group model (similar to a social network) to support collaboration and sharing.

Resource sharing in a social network context has also been examined with varying degrees of success. For example, Automated Service Provisioning ENvironment (ASPEN) [6] exposes applications hosted by Cloud providers to user communities in Facebook. The focus of ASPEN is exposing applications and sharing data within an enterprise through an intuitive and integrated environment, however this could also be applied to a scientific domain. Clusters are used to provide a scalable cloud infrastructure for their demonstrator application. Similar scientific projects have also attempted to combine Grid computing and social networking concepts, communities, and mechanisms. For example, PolarGrid [7] shares information and resources amongst members to study polar ice sheets. Social data is extracted using the OpenSocial interface to support global collaboration and multiple social networking functions are then incorporated in an application specific portal. *Crowdsourcing* is increasingly used as a means of solving large scale problems by distributing tasks to a group of amateur members of the public [8]. This approach is somewhat different to that of the SoCC, in particular the SoCC relies on symmetric resource sharing rather than asymmetric donation.

These diverse scientific platforms highlight the types of collaborative scientific scenarios possible in social networks. In general, the limitation with scientific social networks highlighted above is that they are proprietary and focuses only on the specialised communities they serve and lacks the sizable user bases of commercial social networking platforms (e.g Facebook). The overhead of such an approach is also considerable as administrators need to create and manage proprietary social infrastructures and users require credentials for each network. The same functionality can be realised using a generic scientific Social Cloud deployed in an existing social network. This approach would have the advantage of bringing diverse scientific groups together thereby consolidating scientific resources, as well as the utilisation of a mature social network for a better social experience.

## III. Social Cloud Computing

Social Cloud Computing is a resource sharing framework in which resources and services are shared amongst individuals on the premise of the relationships and policies encoded in a social network [2]. The basis of a Social Cloud is therefore the socio-digital relationships encoded in an online social network that identify implicit levels of trust that underpin and transcend the online community in which they exist. In this paper we apply cloud-based usage model to enable virtualised (elastic) resource sharing through service-based interfaces.

The resulting Cloud infrastructure is a scalable computing model in which heterogeneous resources contributed by users can be dynamically provisioned amongst a defined group of "friends" in a social network. The socially corrective mechanisms (incentives, disincentives, peer pressure) inherent in a social network are used to enable a Cloud based framework for long term sharing with lower privacy concerns and security overheads than are present in traditional Cloud environments.

The Social Cloud takes advantage of social network groups to provide collaboration support and resource sharing within dynamic Virtual Organizations (VOs) [9]. Social network groups have implicit understandings of the purpose of the group and who its members are. By augmenting these implicit rules with policies for collaboration and resource sharing, we can model VOs within the Social Cloud.

Due to the unique nature of the Social Cloud, a social market place has been proposed as a means of regulating sharing [2]. The social market is novel, as it uses both social and economic protocols to facilitate trading. The marketplace itself need not be purely economic and could in fact implement a range of unique protocols such as volunteer contribution, trophy systems, reciprocation or reputation economies.

## IV. Social Collaborative Cloud Requirements

Various requirements have shaped the development of SoCC, this section outlines specific properties required to create a Social Collaborative Cloud for the scientific community.

### A. Use Virtual Machines as the unit of resource sharing

The increase in computational power available on commodity hardware has lead to virtualisation, and more specifically VMs, becoming a common and mature technology. Infrastructure as a Service (IaaS) Cloud providers in particular have adopted VMs as both a means of providing a generic execution platform and also as the defacto standard computational "unit" of work. For example Amazon EC2, Nimbus [10] and Eucalyptus [11] all use VMs. While there is some degree of overhead in virtualisation, VMs offer a number of advantages over other models:

- Abstracts resource heterogeneity: VMs run on a hypervisor layer that can be applied to a range of commodity hardware platforms, this is especially important in

scientific domains in which users have a wide range of diverse hardware and tasks have very specific execution requirements.
- Emulates underlying raw hardware: VMs aim to expose a raw hardware layer that places no requirements on the operating systems or software deployed. This flexibility is crucial in a scientific domain for supporting scientific applications with a wide range of operating system and software requirements.
- Provides customization: VMs enable a high level of customization of resource usage, such as specifying the RAM and disk size, number of CPUs etc. This allows the Social Cloud to customise VMs to accommodate different types of workloads, as well as providing simpler resource management by individual resource contributors.
- Retains state: VMs are highly dynamic and can be paused, restarted, copied and even migrated without losing state. In addition checkpointing can provide some degree of fault tolerance in a highly dynamic Cloud environment. This allows scientists to create pre-packaged fully contained *virtual appliances* that can be shared with peers.
- Provides isolation from other users: A VM is typically isolated from the physical host it is running on through a sandboxing architecture, in addition the VM itself is typically secure and isolated from other VMs sharing the physical host.

### B. Provide seamless integration with social networks

The SoCC must integrate with the various functions of the social network utilizing standard operations as much as possible. Users should be able to share resources through the social network interface, for example instantiating VMs, viewing VM state and performing various management tasks through the same interface. The SoCC makes use of the information the user has already provided to the social network, such as their names and geographic location, and should respect the privacy settings the user has enabled in the social network.

### C. Facilitate formation of virtual clusters

Scientific computations are often large and complex, and therefore require many VMs to run parallel computations efficiently. This in turn requires coordination between VMs for example, to let other instances know which part of a data set is being processed. This suggests that a group of VMs deployed to work on the same data set or project would require a dedicated network amongst themselves. Such a network would imitate traditional HPC clusters [12], thereby providing a level of portability to existing HPC applications.

### D. Support open and standard APIs

The SoCC must communicate with resource providers in order to integrate their resources with the cloud. The use of open standards increases the likelihood that users will be able to use the cloud; either by using middleware that is readily available, or through existing support for the standard in their own systems. A proprietary or custom interface places an unnecessary burden on the users by either having to purchase licenses or implement support from scratch.

### E. Provide interoperability with other Cloud providers

Even though there would only be very limited opportunities for deep integration with commercial cloud providers, we believe that there are certain scenarios in which capacity could be augmented using the public APIs provided by commercial vendors. For instance, a scientist might already have data stored on Amazon S3, it is conceivable this same data could be required by VM instances running in the Social Cloud. In this scenario, the VM would need to establish a connection to Amazon S3 to download the data and store it locally, or the social cloud might provide a feature to automatically download the data as an initialization step before the VM is booted.

## V. Architecture of the Social Cloud

The current SoCC prototype is designed and implemented as a Facebook application, due to it's mature developer tools and extensive developer support. It's large user base will also help us in testing and evaluation. A final production SoCC would make sense deployed on other professionally oriented social networks such as LinkedIn, or spanning across multiple social networks.

In our previous work we implemented a proof of concept Social Storage Cloud [2] to demonstrate the feasibility of the Social Cloud vision. The architecture presented in this section represents a much more flexible, generic, and mature model that incorporates many of the lessons learned from our previous work. The major components of the SoCC are shown in Fig.1, and are detailed below:

- **Application Server:** hosts the Facebook application that integrates the SoCC with the social network. The application server hosts a web application built using the Facebook public API that renders directly inside the Facebook UI, thereby giving the impression of seamless integration to users. The application server is responsible for collecting registered user data from Facebook through the graph API and providing an interface to the SoCC.
- **Image store:** contains the base set of OS images for the VMs. These images are pre-packaged to provide quick instantiation for users who do not want to go to the trouble of preparing their own images. To improve the efficiency of the Social Cloud, individual resource contributors are able to set up caches of these base images. Users also have the option to use their own images, but due to the heterogeneous nature of available virtualisation technologies, packaging a VM image that would work on all platforms could prove challenging to some users. We therefore provide standard pre-packaged images for their convenience.
- **Scheduler:** schedules submitted VMs to one of the available clusters. The scheduler uses a set of customizable policies and constraints to ensure that VMs are distributed fairly and evenly among individual resource contributors.

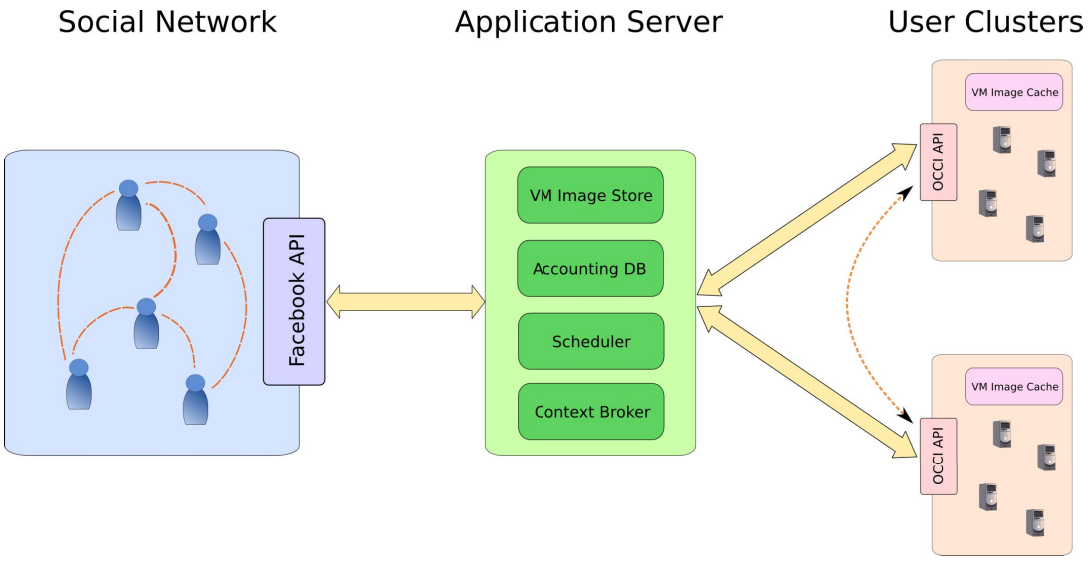


Fig. 1. Architecture of the Social Collaborative Cloud

In addition the scheduler considers user requirements in the VM specification. For example, if the user specifies that a high speed backend network is required for the virtual cluster, then the VM would be submitted to a resource contributor that meets those requirements, or fail if the requirements could not be met.

The compute resources contributed by individual users are heterogeneous with different architectures, network bandwidth, storage capacity etc. This introduces a number of issues when scheduling VMs. For instance, if a user submits a group of VMs as part of a workflow that requires a high bandwidth network to move data, then all the VMs would have to be scheduled to the same resource contributor. Scheduling them to separate contributors, even if the individual contributors have a high speed backend network, is not preferable as the data would then have to be moved through the Internet which may cause a bottleneck.

The scheduler is also responsible for provisioning the current capacity of the Social Cloud fairly among users. In essence, the scheduler should maximise the load distribution and utilisation of the cloud as much as possible while prioritising users according to their contribution, as in the fair share scheduling scheme [13]. Each user is entitled to a proportional share of the resources at sign-up time, and a user's share is increased or decreased depending on contribution or consumption. A decay factor is used to give more weight to more recent usage patterns. When a user submits a request for a VM lease, her remaining shares are translated to a job priority by the scheduler. Depending on the supply and demand at the time of the VM request, users with fewer redeemable shares (and therefore lower job priorities) will potentially experience a longer delay before obtaining a VM lease from the Social Cloud.

- **Context Broker:** is used to contextualise the VM according to the submitted specifications, it is also used to create a virtual cluster. The current implementation uses the Nimbus context broker and context agent components [14]. Configuration relies on shell scripts to customise images, the advantage of this approach is that only minimal dependencies are imposed. After the VM boots the context agent script connects with the context broker to initialise the VM and to set up network interfaces necessary to prepare the virtual cluster. For example, initialization may consist of setting up ssh keys and user accounts so that members belonging to the same VO (social network group) can connect to the VM.
- **Open Cloud Computing Interface (OCCI) API:** is an open standard [15] for resource management in cloud environments. Currently it is in the draft stage but already specifies many functions such as VM instance creation, network configuration and storage creation. We have adopted the OCCI draft specification in the SoCC for communication between user resources and the application server, as well as for communication between user resources.

### A. *Social Collaborative Cloud interactions*

The following section outlines the general functionality of the SoCC, specifically interactions between various components of the SoCC, users, and external services are described.

*1) User registration:* The registration process starts when the user installs the SoCC Facebook application hosted on the SoCC application server. The user installs this application as any other Facebook app using the standard Facebook UI. The process is shown in Fig.2.

Facebook forwards the request to the SoCC application

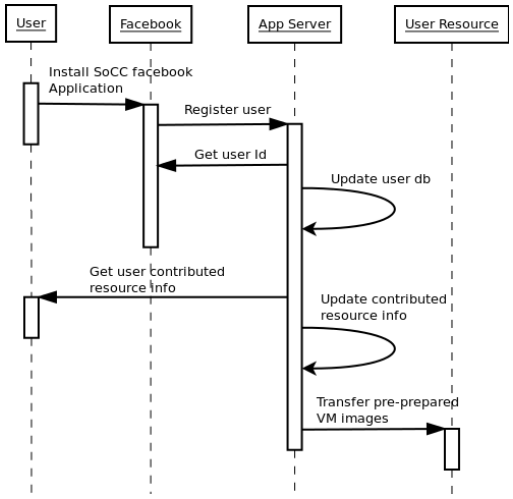


Fig. 2. User registration process

server where the users' Facebook id is recorded in the registered users table. This is necessary to better integrate the SoCC into the social network. The application server then asks if the user would like to contribute resources. This is an optional step and can be skipped if the user has no resources to contribute. The only requirement when contributing resources is that the user exposes an OCCI interface to the resource in the form of a URL and associated username/password pair. If the user is contributing a resource then this information along with specification of whether the resource is to be used exclusively by a VO (see section V-A4) is supplied by the user. The application server then transfers the pre-configured VM images to the users local storage. Users can register additional resources or update information about registered resources at a later time.

*2) Image distribution:* The SoCC supports two models for distribution of VM images to the host user:

- **Option 1** The user directly packages their own image and uploads it to a pre-defined staging area. This might be a storage space leased from either the SoCC itself or from a third party vendor such as Amazon S3. The path to the image is then submitted along with the VM specification. When the VM is deployed, the host resource downloads the image from the given path. The main advantage of this approach is that it allows users a greater level of customization of the image. The main disadvantage is that instantiation is more time consuming, given that the image has to be downloaded to the selected resource.
- **Option 2:** The VM image is pre-packaged by the SoCC and distributed to contributing users in advance. The distribution can be optimised by using peer-to-peer methods such as BitTorrent. With this approach users select the pre-packaged image they prefer during VM specification submission. The advantage of this approach is that the VM can be instantiated quickly, given that the image is already present in the host users resource.

*3) Launching a Virtual Machine:* Launching a VM in the SoCC is subtly different depending on the image distribution method used. The basic steps taken using the second approach (pre-packaged VM) are shown in Fig. 3.

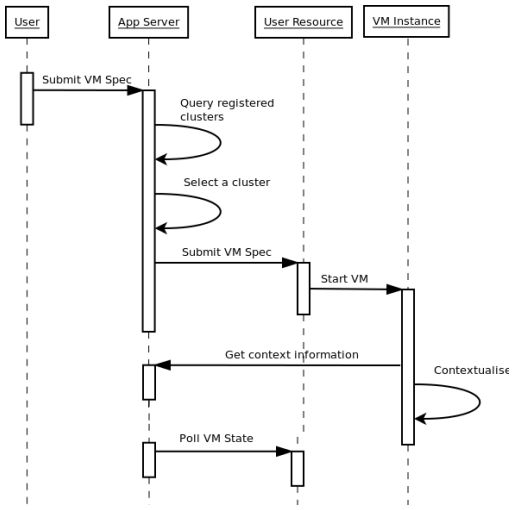


Fig. 3. Launching a VM from a pre-configured image

The process starts when the user submits a VM specification to the SoCC through the standard Facebook UI. The basic specification consists of a VM name, Operating System and type. The type parameter specifies the required resources and is selected from a pre-populated list. For example, a VM instance of type *small* might have a specification of one virtual CPU of 2.0 GHz, 1 GB RAM and 5GB of storage. An instance of type *medium* might have 2 virtual CPUs of 3.0 GHz, 4 GB RAM and 40GB of storage. This approach of specifying the size of a VM is commonly used by other major cloud providers and is likely to be a familiar system for most users. It also simplifies the planning and reporting of resource capacity and utilization by individual resource providers.

After receiving the VM specification the application server queries for available resources with sufficient capacity to host the specified VM. A scheduling algorithm is applied to determine the "best" match, the scheduling algorithm can be based on various factors, for example geographic location of the resource. At present a simple round robin algorithm is used.

Having selected a suitable target, the application server forwards the VM specification to the selected resource which instantiates the VM. This specification includes the identifier of the VM image to use.

The pre-configured VM images include the Nimbus context agent which is started when the VM boots. The context agent connects to the Nimbus context broker (on the application server) to retrieve the contextualisation information, such as the VM name and user account information. After contextualisation the host resource pushes the VM state update to the application server. Users can view the state of their VMs

through the SoCC Facebook application, and connect to it directly via SSH.

*4) Creating Virtual Organizations:* Users can form SoCC VOs by creating Facebook groups. Other users can then join the VO by joining the relevant Facebook group. To make the SoCC aware of a VO, one of the group members can register it with the SoCC as shown in Fig.4.

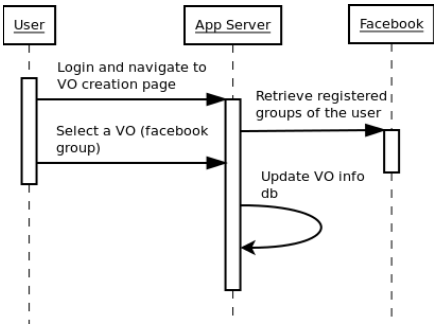


Fig. 4. Virtual Organisation registration process

Members can launch a VM within the context of a VO by selecting it as part of the VM specification. When a VM is submitted in this way, the scheduler will also query resources contributed by VO members that are tagged to be used exclusively by the VO (resources tagged in this manner are only available to members of the VO). The application server finds the members of the VO dynamically by querying the Facebook social graph. When scheduling the VM, the application server gives preference to VO resources. This approach allows users to contribute to a particular scientific project simply by registering a resource to be used exclusively in that projects VO.

*5) Creating Virtual clusters:* A virtual cluster is formed by networking two or more VMs together with one of the nodes acting as a master. Virtual clustering is crucial for supporting some types of computations and workflows. For instance, when deploying a set of VMs with a shared file system.

The Nimbus context broker provides support for virtual clusters in the SoCC. When a VM boots, the context agent establishes a connection with the context broker to download the configuration parameters. The context broker communicates with the context agent inside each VM to configure and manage the cluster. The sequence of steps for creating a virtual cluster is shown in Fig. 5.

The process for creating a multi instance cluster follows the same process as instantiating a single VM, the only difference is additional information is included in the specification sent to the application server. The application server queries the user resources and submits the VM specification to the selected resource as before. Note that the master node VM and the worker node VMs are shown as different entities in the figure only to illustrate their different contextualizations. When they boot, the first VM instance that connects to the context

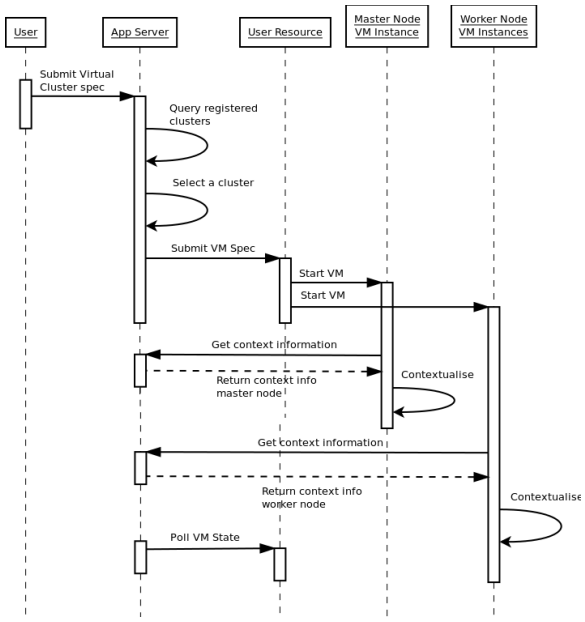


Fig. 5. Contextualisation of virtual clusters

broker is treated as the master node of the virtual cluster and contextualisation information for the master node is passed to this VM instance. After the master node becomes stable, contextualisation of the worker node VMs takes place.

## VI. Evaluation

In this section we present performance measurements taken from the deployed SoCC prototype implementation. In particular this evaluation focuses on the performance of the two image distribution methods. The prototype currently implements the following features:

- User registration using the SoCC Facebook application.
- User resource registration.
- VM submission and scheduling via a round robin algorithm.
- Image distribution via the two distribution methods detailed in section V-A2.
- VM monitoring.

To evaluate the performance of image distribution we used two different sized Linux distributions – *ttylinux* and *ubuntu 10.04 x86_64 server*. The ttylinux is a 32 bit distribution with an image size of approximately 40MB. The ubuntu image is approximately 5GB. The ttylinux is suitable for a user that would like a distribution with only minimal packages such as a C compiler and bash interpreter. Ubuntu might be preferable for a user that wants to quickly deploy a web server with minimal effort. These two distributions therefore cater to different types of usage scenarios.

Recall, the two image distribution methods are:

- **Option 1:** The user uploads the image to a staging area. The user submits a URL pointing to the image


Fig. 6. Screenshot of the SoCC prototype running inside the Facebook UI

location with the VM specification to the SoCC. The scheduled resource then downloads the VM image before instantiating it.

- **Option 2:** The user selects a pre-packaged image when submitting the VM specification to the SoCC. These images are provided by the SoCC and are cached by the registered user resources, so there will not be any need to download the image before instantiating the VM.

The VMs from ttylinux were deployed with specifications of 64MB RAM and 0.5 units of virtual CPUs. Those from ubuntu were deployed with specifications of 1024MB RAM and 1 (one) unit of virtual CPU. The testbed we used for the experiment consists of a cluster made up of 3 machines (Core2Duo 3.0 Ghz, 4G RAM, 250GB hard-disk each) located in our lab and is connected by a LAN. The SoCC application server is also located on a machine of similar configuration in the same LAN. For this experiment we registered a single compute resource, we then submitted a single VM specification through the SoCC Facebook application and measured the time taken between various stages of the VM lifecycle (Fig. 7). To measure the time taken to distribute the image (for option 1) we staged the user uploaded images to another machine in the same lab before submitting the VM specification. The user resource hosting the VM downloads the image from this machine via a HTTP connection before creating the VM.

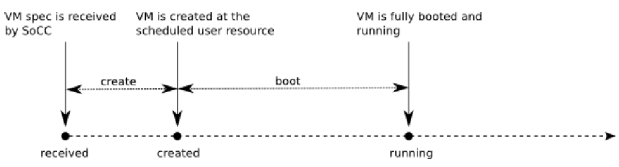


Fig. 7. VM lifecycle states in the SoCC

The measurements were taken at the SoCC application server by logging the times at which VM status updates were reported from the host resource. As detailed in section V-A3, individual user resources "push" VM status updates to the application server. For the two image types, the average of five runs for each image distribution option was taken. The results are presented in the following table and graph.

| | create (sec) | boot (sec) |
|---|---|---|
| ttylinux option 1 | 10 | 15 |
| ubuntu 10.04 option 1 | 820 | 126 |
| ttylinux option 2 | 2 | 10 |
| ubuntu 10.04 option 2 | 2 | 125 |

TABLE I
VM LIFECYCLE TIMINGS FOR THE TWO IMAGE DISTRIBUTION OPTIONS

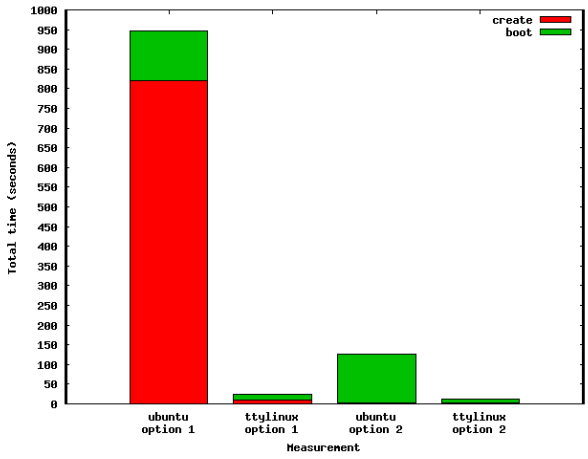


Fig. 8. Average times required to create and run VMs within the SoCC.

These results show the relative performance between the two image distribution methods. Although all the machines used in this experiment were located in the same lab, the impact of the two methods can still be compared. Clearly the cost of transferring an image (e.g ubuntu for option 1) is a considerable overhead when compared to the time taken to boot the image. The difference in boot times for ttylinux and ubuntu are due to their image sizes and preloaded packages in these two distributions.

The conclusion from these results is that option 1 is more suitable for small image sizes, which would likely be the case if a user wishes to customise the image heavily. To aid the user in this scenario, we could make our pre-packaged images downloadable so that they can customise it by adding or removing the packages they want, while still likely maintaining compatibility with the individual resource providers. Option 2 is suitable for deploying a VM quickly as the images are already cached in the hosting user resource.

## VII. Future Work

We are actively working on improving the SoCC prototype. The major focus at this point is providing transparent VO management through integration with the Facebook group feature. In addition we are also implementing contextualisation of VMs using the Nimbus context broker. An area for future research is creating socially oriented scheduling algorithms that determine appropriate resource allocation based on VO membership and the relationships represented in the social network. We are also, concurrently, exploring incentive mechanisms for contribution and diverse economic allocation models such as reputation and reciprocation based economies that are designed to favor users who contribute heavily.

## VIII. Conclusion

The scientific community represents a unique use case in which individuals form highly dynamic, potentially short duration, collaborations that involve sharing a wide variety of information and resources. Increasingly scientists are turning to social networks as a tool to improve collaboration and share information instantly.

In this paper we introduced the concept of a Social Collaborative Cloud (SoCC) designed to take the idea of sharing in a social network one step further. In the SoCC users are able to form dynamic VOs and share computational resources with one another using VMs. With our initial prototype implementation we have investigated the viability of VM image distribution within the SoCC network. Results from our experiment have given us sufficient motivation to continue our work towards implementing a full prototype that would be usable in the real world.

## References


[1] K. Chard, S. Caton, O. F. Rana, and K. Bubendorfer, "Social cloud: Cloud computing in social networks," in *proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD)*, Miami, Florida, July 2010.

[2] K. Chard, K. Bubendorfer, S. Caton, and O. Rana, "Social cloud computing: A vision for socially motivated resource sharing," *To appear in IEEE Transactions on Services Computing*, vol. 4, no. 4, 2011.

[3] D. D. Roure, C. Goble, and R. Stevens, "The design and realisation of the myexperiment virtual research environment for social sharing of workflows," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 561–567, 2009.

[4] G. Klimeck, M. McLennan, S. P. Brophy, G. B. Adams III, and M. S. Lundstrom, "nanohub.org: Advancing education and research in nanotechnology," *Computing in Science and Engineering*, vol. 10, pp. 17–23, September 2008.

[5] I. Foster, "Globus online: Accelerating and democratizing science through cloud-based services," *IEEE Internet Computing*, vol. 15, pp. 70–73, 2011.

[6] R. Curry, C. Kiddle, N. Markatchev, R. Simmonds, T. Tan, M. Arlitt, and B. Walker, "Facebook meets the virtualized enterprise," in *Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC '08)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 286–292.

[7] Z. Guo, R. Singh, and M. Pierce, "Building the polargrid portal using web 2.0 and opensocial," in *Proceedings of the 5th Grid Computing Environments Workshop (GCE '09)*. New York, NY, USA: ACM, 2009, pp. 1–8.

[8] J. Howe, "The rise of crowdsourcing," Wired Magazine http://www.wired.com/wired/archive/14.06/crowds.html, June 2006.

[9] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International Journal of Supercomputer Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[10] K. Keahey, I. Foster, T. Freeman, and X. Zhang, "Virtual workspaces: Achieving quality of service and quality of life in the grid," *Scientific Programming Journal: Special Issue: Dynamic Grids and Worldwide Computing*, vol. 13, no. 4, pp. 265–276, 2005.

[11] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Proceedings of 9th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 09)*, Shanghai, China., 2009.

[12] W. Emeneker and D. Stanzione, "Cluster spanning with virtual environments." in *CLUSTER'05*, 2005, p. 1.

[13] J. Kay and P. Lauder, "A fair share scheduler," *Commun. ACM*, vol. 31, pp. 44–55, January 1988. [Online]. Available: http://doi.acm.org/10.1145/35043.35047

[14] K. Keahey and T. Freeman, "Contextualization: Providing one-click virtual clusters," in *Procedings of the 4th IEEE International Conference on eScience (eScience '08)*, dec. 2008, pp. 301 –308.

[15] OCCI, "Open cloud computing interface," http://occi-wg.org/.