Future Generation Computer Systems 29 (2013) 2143–2156

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

eScience in the Social Cloud☆

Kris Bubendorfer [a,*], Kyle Chard [b], Koshy John [a], Ashfag M. Thaufeeg [a]

[a] School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand
[b] Computation Institute, University of Chicago and Argonne National Laboratory, Chicago, IL, USA

HIGHLIGHTS

- We explore two Social Cloud models designed to support collaborative eScience.
- Social Clouds rely on a collaborative overlay obtained from existing social networks.
- The SoCC supports VM based sharing of computational resources between peers.
- The SoCPR uses social incentives to increase contribution in volunteer computing.
- The prototype and simulations show efficient sharing and increased contribution.

ARTICLE INFO

Article history:
Received 6 March 2012
Received in revised form
29 January 2013
Accepted 6 April 2013
Available online 16 April 2013

Keywords:
Social Cloud
Distributed systems
Cloud computing
Volunteer computing
Collaboration
Resource sharing

ABSTRACT

Social networks offer great potential for fostering collaboration between individuals and amongst groups. This potential collaborative environment is not only applicable for recreation, but can also provide considerable value to diverse research communities. For this reason scientists are increasingly utilizing social networking concepts in projects to form groups, share information, publicize their work and communicate with their peers. This article describes two different approaches to supporting eScience, by providing scientific computing and collaboration within what we term the Social Cloud. In our first approach the social network is used as a collaborative overlay, in combination with the ad hoc creation of infrastructure composed of virtual machine clusters built from resources contributed, by the users, to the Social Cloud. Our second approach is based around the principle of volunteer computing, where the Social Cloud provides researchers with a platform to exploit social networks by reaching out to non technical users who would otherwise be unlikely to donate computational time for scientific and other research. In this article we specifically explore the motivations of users to contribute computational time and examine the various ways these motivations can be catered to through the use of incentives in existing social networks.

1. Introduction

The rapid growth of social networking has created new ways for individuals and companies to communicate and share basic information without geographic barriers. These digital relationships are as important to many individuals as their real world relationships and form a primary part of their daily social interactions. Many applications now use social networks as a platform for authentication, e.g., Facebook Connect and application Portals such as ASPEN [1] and PolarGrid [2]. However, Social Networks offer the potential for much more than simple user authentication or the provision of portals. Indeed, social networks provide a valuable general basis for enabling communication, coordination and discovery for individuals and groups of individuals. Scientists and researchers have been quick to adopt the principles and ideas of social networking, adding them to scientific platforms such as MyExperiment [3] and nanoHub [4] but the scope of these projects is limited by the size and disconnect between their respective communities.

The Social Cloud, first published in [5], takes a different approach by extending collaboration and computation functionality to existing social networks—rather than adding social networking to existing computation and collaboration tools. This has a number of advantages—we can utilize the existing connections between people, exploiting their preexisting and vital communities,

☆ This is a combined and extended version of the papers: K. John, K. Bubendorfer, and K. Chard. A Social Cloud for Public eResearch. In proceedings of the 7th IEEE International Conference on eScience, Stockholm, Sweden, December 2011. And A. Thaufeeg, K. Bubendorfer, and K. Chard. Collaborative eResearch in a Social Cloud. In proceedings of the 7th IEEE International Conference on eScience, Stockholm, Sweden, December 2011.

* Corresponding author. Tel.: +64 4 463 6484; fax: +64 4 463 5045.
E-mail addresses: kris@ecs.vuw.ac.nz (K. Bubendorfer), kyle@ci.uchicago.edu (K. Chard), koshyjohnuk@msn.com (K. John), thaufeashf@myvuw.ac.nz (A.M. Thaufeeg).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.04.003

and we can utilize the social network's familiar and comfortable user interfaces (including the use of groups, incentive mechanisms, feeds and walls) to reduce the learning curve and frustration with yet another system. From a pragmatic technical point of view, we benefit from well developed programming interfaces, reliable architecture, provision of infrastructure, plus authentication and authorization systems. Specifically a Social Cloud is defined in [6] as; *A Social Cloud is a resource and service sharing framework utilizing relationships established between members of a social network*, and relies on the principle that social connections represent contextual trust that can be applied in any scenario where sharing and exchange (essentially collaboration) takes place. A Social Cloud is unique in that it is completely open and ad hoc—it is created, controlled and managed by its users and therefore requires only a lightweight infrastructure external to that provided by the host social network.

Researchers faced with extremely large computations or the requirement of storing vast quantities of data have come to rely on distributed computational models, including clusters, grids and more recently clouds, to provide large scale science capacity. However, distributed computation is perceived as complex, expensive and often has little uptake outside of the eScience community—that is, by the wider group of eResearchers. This article explores two developments within the Social Cloud project in support of eScience and eResearch.

The first project is the Social Collaborative Cloud, outlined in Section 1.1, and the second project in this article is the Social Cloud for Public eScience, which is outlined in Section 1.2.

1.1. Social Collaborative Cloud

For scientists and other eResearchers, 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 be used to support collaboration by increasing communication channels and also facilitating the discovery of other scientists working on similar projects. Scientific collaborations also often have resources, such as compute, data and meta resources (e.g. workflows), that 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, authorization, multiple systems and credentials, multiple user interfaces and so on.

The Social Collaborative Cloud (SoCC), is a framework in which individuals or institutions contribute the capacity of their compute, data and other resources, which are leased through the social network. In the context of computation, members of the Social Cloud can contribute, request, and use Virtual Machines from other members, as well as from Virtual Organizations aligned with project goals by forming Social Network groups. The SoCC enables scientists to share resources for the duration of a collaboration, allows more efficient use of available resources, as scientists with similar projects or interests can naturally consolidate resources towards common goals, and most importantly, it promotes greater uptake beyond the scientific community by utilizing familiar tools, as well as publicity and networking opportunities provided by the host social network. In this way, the social network is not just an add-on, it is instead integral, to the Social Cloud.

1.2. Social Cloud for Public eResearch

Volunteer computing [7] is an alternative means of obtaining large computing resources for eScience and lately eResearch—by getting the public to support specific projects by donating their spare computational and storage resources. The amount of computational time available to researchers is a function of the number of volunteers contributing at any given point of time. While there are a sizable number of volunteers who participate in volunteer computing (e.g. 2.2 million BOINC participants [8]), this is insignificant when compared to the 845[1] million active Facebook users [9] and the potential compute power they could contribute for research—within their own individual limitations.

The Social Cloud for Public eResearch (SoCPR) aims to provide researchers with a platform to exploit social networks to reach out to users who would otherwise be unlikely to donate computational time for scientific and other research oriented projects. We explore the motivations of users to contribute computational time and examine the various ways these motivations can be catered to through established social networks. Specifically, we look at integrating Facebook and BOINC, and discuss the architecture of the functional system and the novel social engineering algorithms that power it.

1.3. The Social Cloud paradigm

What is different about the Social Cloud paradigm? The social network comes first: It is not a cloud or collaboration environment extended with a social network, it is a social network extended with cloud functionality. The people and their networks form the basic infrastructure, the services they access and share are formed around their unique social graph. Individuals contribute their competencies—services that encapsulate; data, storage, computation, algorithms, etc. The users choose how to delegate their contributions between the groups within which they collaborate.

Users fulfill all roles, from provision of resources, management of collaborations through to consumption of services and computation—access to which is performed with familiar social networking tools. In some ways this turns the provision of infrastructure on its head. Rather than fitting the user to the infrastructure, we build the infrastructure around the user.

A Social Cloud is not crowdsourcing as the relationships in the social cloud are essentially[2] symmetric, that is, participants are more-or-less equals who come together under some commonality of interest to benefit from sharing. There is explicitly a preexisting and external underlying set of relationships that connects people within a Social Cloud (no anonymity). Whereas crowdsourcing operates in the master–worker model where workload flows in one direction, while there may be monetary exchange or other compensation this does not in itself constitute sharing.

1.3.1. Advantages in usability

Using a Social Cloud lowers usability barriers: The interface and tools are already familiar to non expert (non computer science) users. Traditionally difficult authorization and access control take place transparently for the owner and users. There is no visible certificate management, the social network provides single sign on and applications can be seamlessly integrated with core functionality. Collaborations are light-weight and dynamic, services and resources can be delegated, removed and accessed using the social network group structure. Users share services for the duration of collaboration. A good role for a Social Cloud is in the early stages of a project, when the costs of dedicated or leased infrastructure would be prohibitive.

1.3.2. Groups in a Social Cloud

Groups in Social Networks, like virtual organizations [10], have an intent, membership and policies that define sharing in

[1] March 2012.

[2] The Social Cloud for Public eResearch is the exception.

social networks in relation to photos, media, etc. We exploit this analogy between social networking groups and dynamic virtual organizations by extending the social networking group structure to also represent Virtual Organizations. Social clouds are not mutually exclusive, that is, users may be simultaneously members of multiple Social Clouds. The analogy weakens a little when considering the permanence of a VO or group. While a VO is often associated with a particular application or activity, and is often disbanded once this activity completes, a group tends to be longer lasting and may be used in the context of multiple applications or activities.

2. Related work

Scientific applications are increasingly turning to social networks and social networking techniques as a means of organizing users and providing a collaborative computing environment. There are countless examples of applications that leverage existing social networks to provide user management, group management and authentication, for example ASPEN and PolarGrid both use social networks to manage users and facilitate resource sharing through integrated cloud and grid based applications. Other projects, such as MyExperiment, nanoHub and Globus Online [11], take the reverse approach by creating a proprietary social network within their application. MyExperiment provides a virtual research environment where collaborators can share and execute scientific workflows, while nanoHub allows users to share data and transparently execute applications on distributed resource providers such as TeraGrid. Globus Online is in the process of adding a group model (similar to a social network) to support scientific collaboration and sharing within the scope of their existing scientific tools.

In the commercial domain, application specific resource sharing networks such as FriendStore [12] and AmazingStore [13] allow users to share storage with their friends. Unlike the Social Cloud both FriendStore and AmazingStore offer a single resource (storage) and exist only within their P2P social network created by their members, they therefore do not leverage existing social networks nor facilitate more general sharing models. One of the most common socially based computation model is *Crowdsourcing*, which is used as a means of solving large scale problems by distributing tasks to a group of amateur members of the public [14]. The success of crowdsourcing projects demonstrates the potential for providing large scale computing resources through social computing, however it only provides a mechanism for asymmetric resourcing rather than the collaborative symmetric environment established in a Social Cloud.

In the eScience domain, volunteer computing such as SETI@Home [15] and Folding@Home [16] offer another asymmetric model for resource provisioning. SETI@Home was started in 1999 to analyze radio signals coming from outer space in the hope of detecting signatures indicative of intelligent life, while Folding@Home performs simulations of protein folding to provide a better understanding of the development of many diseases. Both projects have far exceeded researchers' expectations, gathering huge resource pools and generating worldwide publicity. clouds@Home [17] further extends the volunteer computing model by creating a reliable VM-based platform over unreliable contributed resources.

To simplify the process of creating volunteer computing platforms, the BOINC (Berkley Open Infrastructure for Network Computing) [18] platform was created as a generic volunteer computing architecture. BOINC has grown to support over 50 distinct projects, including Malariacontrol.net and Rosetta@Home. Malaria control simulates the spread of malaria to determine minimal efficacy and duration of effects needed for a trial vaccine and also to optimize deployment of established treatments, and Rosetta@Home discovers the shapes of new designs for three-dimensional proteins—in order to help find cures for intractable diseases such as cancer and HIV. The collective BOINC platform offers over 5.6 petaflops[3] of processing power through its army of volunteers. This represents a considerable computing resource, as the fastest supercomputer in the world (Japan's K Computer) has only recently broken the 10 petaflop barrier.[4] Several other volunteer computing middleware alternatives are also in public use, including Folding@Home and Distributed.net, which have fewer users than BOINC, and there have also been commercial volunteer computing systems such as XGrid [19] from Apple and Grid MP [20] from Univa.

Like any contribution or sharing framework, one of the major difficulties of the Social Cloud model is recruiting and retaining users, and motivating contribution. This task is even more difficult in volunteer computing projects, as there is typically no reward for user contribution. Moreover, there are perceived (and real) barriers for entry both in Social Clouds and volunteer computing, especially for less technical users. To alleviate some of the barriers for entry, BOINC has established a collaboration with consolidated account management systems, such as GridRepublic [21], to ease multi-project management for volunteers. Before the introduction of consolidated account management systems, users who supported multiple projects had to manually set up and manage their contribution for each project. In an attempt to leverage social networks to recruit users, BOINC has collaborated with Intel to create a Facebook application called Progress Thru Processors [22]. Using this Facebook application, users can contribute excess compute power to individually selected scientific projects and they can also publish statistics of their contributions in Facebook. Progress Thru Processors has seen modest success, however we believe there is much greater success possible if the social aspects of social networking are fully embraced, both for existing volunteer computing projects and also Social Clouds.

In general, the primary limitation with current scientific Social Networks is that they are proprietary and focus only on the specialized communities for which they serve, they therefore lack the sizable user bases of commercial social networking platforms. The overhead of creating and managing a proprietary social network is also considerable for both administrators and users. The same functionality can be realized using a generic scientific Social Cloud deployed in an existing social network. This approach has the advantage of bringing diverse scientific groups together, thereby consolidating scientific resources, as well as using a mature social network for a better social experience. The social overlay of a social network also provides opportunities to develop socially oriented incentives that will increase user recruitment and contribution.

3. The Social Collaborative Cloud

In the SoCC we apply a cloud-based usage model to the Social Cloud to enable virtualized (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. Social mechanisms (incentives, disincentives, peer pressure) inherent in a social network are used to enable a Cloud based framework for long term sharing.

The SoCC is built as an integrated social network application and it is designed to utilize core functionality and information provided

[3] http://boincstats.com/ —February 2012.
[4] http://top500.org —November 2011.

by the host social network, such as authentication processes and user's names, relationships and geographic locations.

The virtualized resources shared in the SoCC are exposed through Virtual Machines (VMs). Virtualization has become a common means of providing sand-boxed execution environments and providing a platform on which computational resources can be shared. 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 de facto standard computational "unit" of work. For example Amazon EC2, Nimbus [23] and Eucalyptus [24] all provision VMs. While there is some degree of overhead in virtualization, the benefits offered make it a sound choice for sharing computational resources in the SoCC.

As scientific computations are often large and complex, users may require large scale distributed resources to run computations. To provide a suitable scalable architecture the SoCC includes the ability to provision multiple VMs to create a virtualized cluster resource that is capable of executing parallel computations. The creation of a dedicated virtual network between VMs is used to simplify the coordination process for VMs working on the same dataset or project. Moreover, such a network model better imitates traditional clusters, thereby reducing the overhead of porting existing applications to the SoCC.

3.1. SoCC architecture

The SoCC prototype is designed and implemented as a Facebook application, due to Facebook's mature developer tools and extensive API. A similar SoCC architecture could also be deployed on other social networks such as LinkedIn, or it could possibly span multiple social networks.

The architecture presented in this section represents a flexible, generic, and mature model that incorporates many of the lessons learned from our previous work [6]. 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 renders directly inside the Facebook UI, thereby giving the impression of seamless integration to users. The application server also retrieves registered user data from Facebook through the Facebook Graph API, which is used to make scheduling decisions.
- *Image store*: contains the base set of system images for the VMs. These images are pre-packaged to provide quick instantiation for users who do not need to prepare their own custom images. The base images can be cached by each contributor to reduce instantiation times. Preparation of custom images requires considerable expertise due to the heterogeneous virtualization technologies.
- *Scheduler*: allocates requested VMs to one of the available clusters. The scheduler uses a set of customizable policies and constraints to ensure that VMs are distributed according to policy among individual resource contributors. In addition the scheduler considers user requirements in the VM specification, such as the need for a high speed backend network behind the virtual cluster.[5] The scheduler used in the SoCC is based on a fair share scheduling scheme [25], although in practice the scheduler should be selected to meet the needs of the specific collaborative group. The goal of the fair share scheduler is to maximize the load distribution and utilization of the cloud while prioritizing user requests according to their contribution. Each user is entitled to a proportional share of the resources at sign-up, and a user's share is increased or decreased depending on their 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, their remaining shares are translated to a job priority by the scheduler. Allocation is then based on the supply and demand at the time of the VM request, users with fewer redeemable shares (and therefore lower job priorities) can experience a longer delay before obtaining a VM lease from the Social Cloud. If contributed resources are idle, and no redeemable shares are presented, then the VMs will be provisioned on a First Come First Served (FCFS) basis to avoid lost cycles.
- *Context broker*: is used to contextualize the VM according to the submitted specifications, it is also used to create a virtual cluster. The SoCC implementation uses the Nimbus context broker and context agent components [26]. VM configuration uses shell scripts to contextualize 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 initialize 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.
- *User clusters*: are computational resources contributed by users. These resources allow the application scheduler to allocate user VMs and support communication between resources. We have adopted the Open Cloud Computing Interface (OCCI) API [27] to provide cloud based resource management, including VM instance creation, network configuration, storage creation and VM instance termination.

3.2. SoCC interactions

The following section describes the general functionality of the SoCC, specifically outlining interactions between various components of the SoCC, users, and external services.

3.2.1. User registration

When users install the SoCC Facebook application, they (and their contributed resources) are registered with the SoCC application. A user installs this application like any other Facebook application—using the standard Facebook UI. The registration process is shown in Fig. 2.

When installing the SoCC application the installation request is forwarded to the SoCC application to record the user's Facebook ID in the registered users table. The application then asks the user if they would like to contribute resources. This is an optional step and can be skipped if the user has no resources to contribute. The only requirements when contributing resources is that the resource exposes an OCCI interface and that the user registers the URL and associated username/password pair with the application server. In addition the user can specify optional access permissions for the resources, for example they can specify that the resource is to be used exclusively by a VO (see Section 3.2.4). To optimize the VM creation process the application server transfers pre-configured VM images to the user's resources or local storage. Users can also register additional resources or update information about registered resources at a later time.

[5] The heterogeneity of contributed compute resources 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.

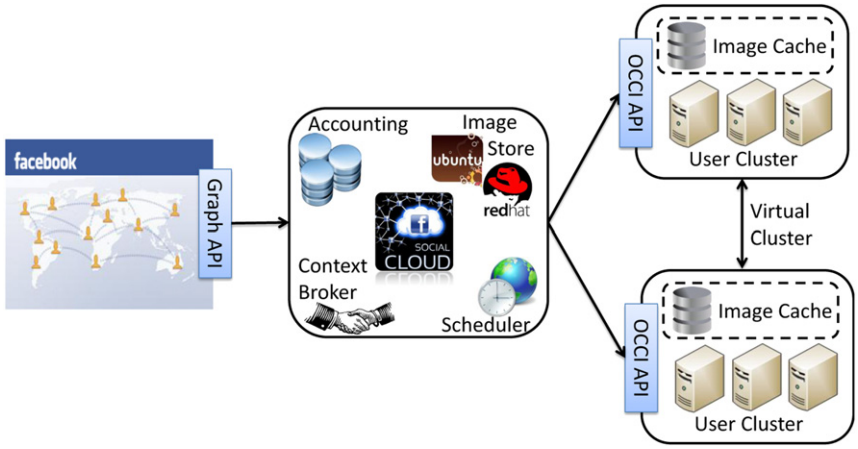


Fig. 1. Architecture of the Social Collaborative Cloud.

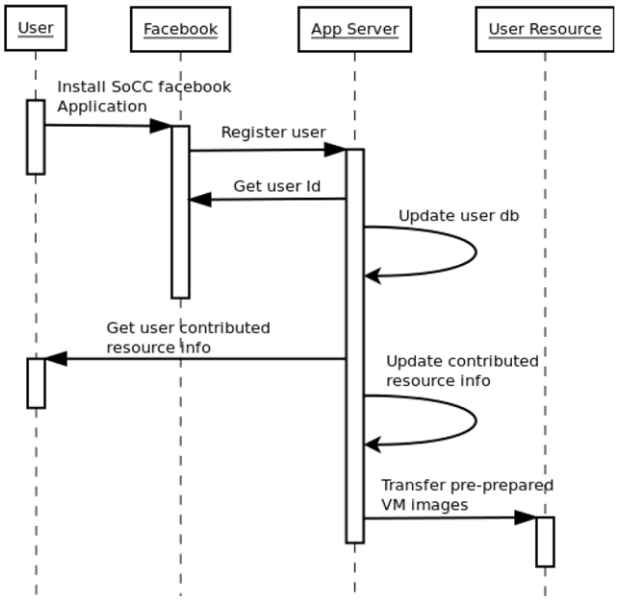


Fig. 2. User registration process.

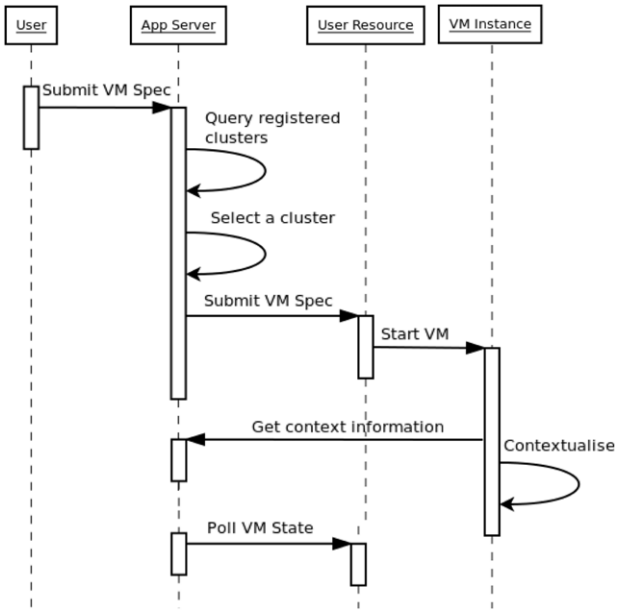


Fig. 3. Launching a VM from a pre-configured image.

3.2.2. *Image distribution*

The SoCC supports two models for distributing VM images to the host user: (1) the requesting user directly packages their own image and the location of the image is submitted along with the VM specification. When the VM is deployed, the host resource downloads the image. The main advantage of this approach is that it allows users a greater level of customization of the image, while the main disadvantage is that instantiation takes considerably longer. Alternatively, (2) the VM image is a standard pre-packaged SoCC image and is cached locally on the physical machines. With this approach users nominate the pre-packaged image they prefer during VM specification submission. The advantage of this approach is that the VM can be instantiated rapidly.

3.2.3. *Launching a Virtual Machine*

To launch a new VM a user submits a specification to the SoCC through the standard Facebook UI. The steps taken using the standard VM image approach are shown in Fig. 3. The specification consists of a VM name, Operating System and *type*. The type parameter defines the resources required, chosen from a set of predefined resource configurations (e.g. small, medium and large). For example, a VM instance of type *small* has one virtual CPU of 2.0 GHz, 1 GB RAM and 5 GB of storage. An instance of type *medium* has 2 virtual CPUs of 3.0 GHz, 4 GB RAM and 40 GB of storage. This approach is frequently used by cloud providers and will be familiar to users. From a technical view, it simplifies the planning and reporting of resource capacity.

After receiving the VM specification the application server queries the resources with sufficient capacity and the scheduling algorithm is applied to determine the most suitable match. The application server then forwards the VM specification to the resource and the VM is instantiated. Each VM image must 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 contextualization information, such as the VM name and user account information. After contextualization the host resource pushes the *RUNNING* VM state and the assigned IP to the application server. Users can view the runtime state of their VMs in the SoCC Facebook application, and connect to it directly via SSH.

3.2.4. *Creating Virtual Organizations*

Users can form VOs in the SoCC by first creating Facebook groups and defining a set of access policies. Depending on these policies other users can request to join a group or the group owner

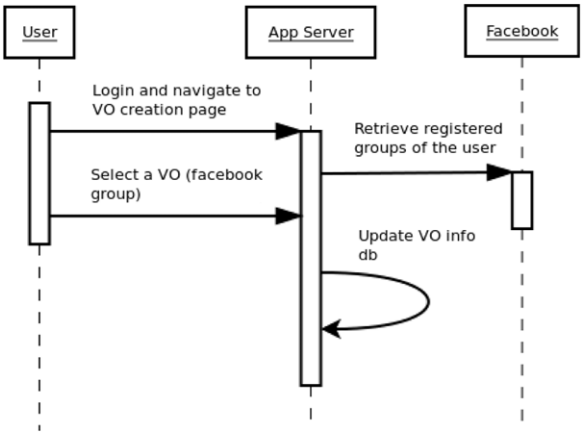


Fig. 4. Virtual Organization registration process.

can select users to join the group. To bind the social network group to a SoCC VO, one of the group members must register the group with the SoCC, as shown in Fig. 4.

VO members launch a VM within the VO context by including the VO identifier in the VM launch specification. In response, the scheduler will query resources that have been assigned to the specified VO. The application server finds the members of the VO dynamically by querying the Facebook Graph. When scheduling the VM, the application server gives preference to VO resources, but may include other resources as well. This approach allows users (non group members) to contribute to a particular scientific project simply by registering a resource to be used exclusively by the project's VO. There are potential privacy and data security issues in this open approach, and the VO policies may choose not to allow this. The SoCC includes a VO management interface that allows group administrators to manage VM instances, group members are able to view and utilize these resources through the same interface. Group membership management is conducted through the Facebook group interface.

3.2.5. Creating virtual clusters

Virtual clustering is crucial for supporting parallel computations, distributed workflows and providing a shared file system. To provide this functionality the SoCC uses the Nimbus context broker to create virtual clusters.

The process for creating a cluster follows the same sequence as instantiating a single VM, the only difference is that additional specifications are submitted to the application server. The sequence of steps for creating a virtual cluster is shown in Fig. 5.

When a VM boots, the context agent establishes a connection with the context broker to download the configuration parameters. The first VM instance that connects to the context broker is designated as the master node of the virtual cluster and contextualization information for the master node is passed to this VM instance. After the master node becomes stable, contextualization of the worker node VMs takes place.

4. The Social Cloud for Public eResearch

The Social Cloud for Public eResearch (SoCPR) presents a social network based volunteer computing approach to encourage users to contribute resources to public science projects. The SoCPR is designed to leverage the considerable resources available through everyday users of a social network by encouraging them to contribute to existing volunteer computing projects. It is hoped that this approach will expand the reach and visibility of volunteer computing from its current technical users to everyday social network users. This work also serves to highlight techniques for encouraging contribution in the more generic Social Cloud context.

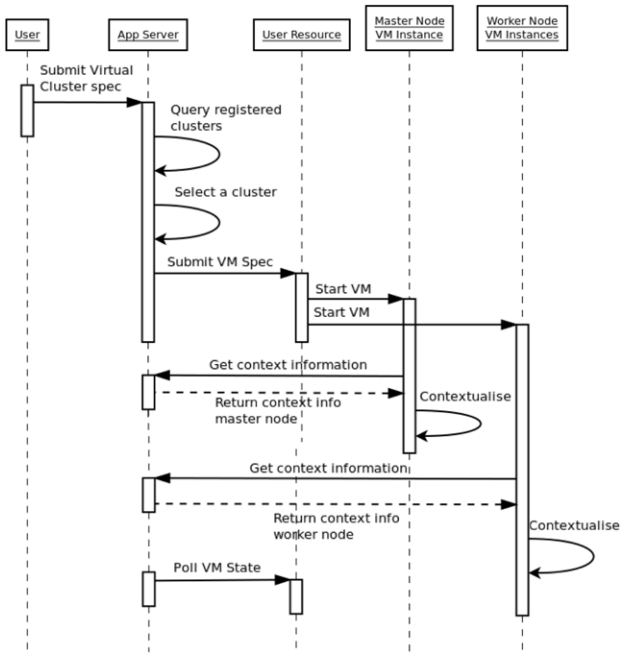


Fig. 5. Contextualization of virtual clusters.

4.1. Motivation

The adoption of Social Cloud computing provides a novel mechanism to leverage social engineering to motivate and facilitate volunteer based resource contribution. Specific motivations for users and researchers are detailed in the following sections and referenced throughout the remainder of this section.

4.1.1. Users

There are several reasons users currently participate in volunteer computing projects, in general users are motivated by either altruistic or self-interested reasons. For example, altruistic users may have a desire to make a difference or they may be strongly interested in a particular field of research, whereas self-interested users are typically motivated by competition with regards to their contribution size (leader-boards) or they might want to be publicly recognized. A 2007 user study by BOINC, referenced in [28], found that approximately 75% of contributors were motivated because the science is important and beneficial, while fewer than 16% were motivated by "Fair and quick granting of credit for work done" or "Getting more credit from this project than from others".

The motivation of volunteer computing users was studied in [29] and the following key factors were identified as strong motivation for volunteers: the potential impact of the science, the probability of success, the utility of the project, the safety of the project, political signals, democratizing science, and personal benefits such as a sense of community, competition, personal interest and visual pleasure. A more comprehensive study [28] of volunteers in `climateprediction.net` classifies several distinct classes of volunteers, and focuses on 3 classes in particular. Firstly, the *super-crunchers* (10% of active users, 60% of all credits) who offer considerable processing power to a small number of projects (between 1 and 3) and are motivated by their standing within a community. Retention of the super-crunchers requires credit systems, recognition and on-line forums. Secondly, the *lay public* (80% of active volunteers, less than 25% of all credits) who make smaller contributions to a small number of projects (again, between 1 and 3) and while they often state they are not motivated

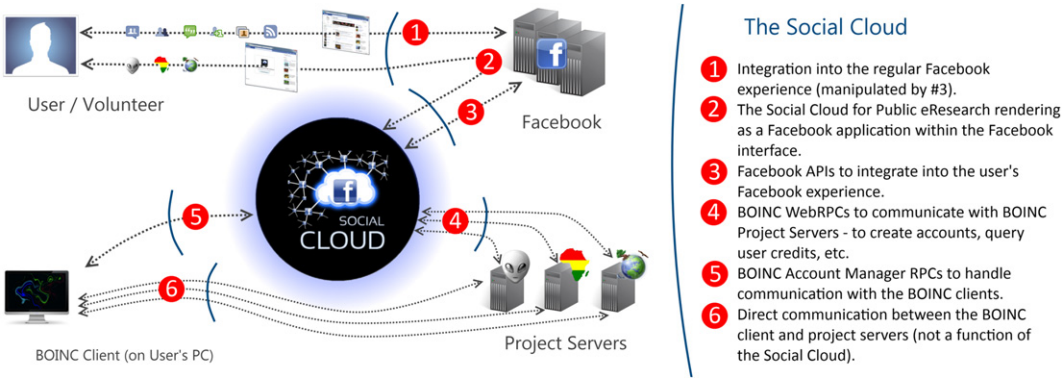


Fig. 6. Architecture of the Social Cloud for Public eResearch.

by earning credits, the credit and team systems play an important role in their retention [30]. Retention of the lay public also requires credit systems, but as they are motivated by science they also need to see the project progressing and their contributions making an impact on the overall project. Thirdly, the alpha testers are big processors that contribute to many projects (12 or more). They are often involved from recruitment during the alpha stage of development and retain some residual work units for the project. Typically they stay on because they have a unique and prestigious status as part of a small group of invited users. As this final group is more weakly defined, the strategies for retention are less clear, however credit systems play no role in retention of this group.

In a Social Cloud, relationships between users are used to share information and resources, these relationships can also be used to determine suitable projects for participation and reinforce retention. Targeted social algorithms and techniques can be used to both maximize the number of new volunteers, and to keep them engaged and involved in a project so that the computational time available to researchers grows quickly and sustainably. The SoCPR provides a single integrated management view (within the social network) of all projects that a user contributes to. Using this interface users can easily monitor current activity and explore new projects.

4.1.2. Researchers

Researchers face stiff challenges to obtain volunteer resources. They must advertise their projects and encourage user contribution. There are examples of under resourced projects where the resources gleaned by less popular projects have returned less than the cost of the software development. Indeed, volunteers may unintentionally favor high visibility projects and therefore discourage new or lesser known projects.

The SoCPR will enable researchers to connect naturally with the people (and groups) that support their research, and volunteering in new projects will be supported through incentive algorithms and policies.

4.2. SoCPR architecture

The architecture of the SoCPR is shown in Fig. 6. Like the SoCC it is a privately hosted Facebook application, but in this case it is designed to work with BOINC. From the perspective of project servers and volunteer computers running the BOINC client, the SoCPR acts as an account management system. From the perspective of users (volunteers), the SoCPR is a socially aware account management system that runs as a Facebook application. The existence of the Social Cloud is transparent to BOINC researchers and requires no additional effort on their part to support.

Users can add and remove supported projects from within the Facebook application, they can also set relative resource shares (percentages of the total computational time donated) for the various projects that they choose to support. This information is used to communicate with the various project servers and to control the BOINC client running on the user's PC.

Like any BOINC account manager, the SoCPR connects to BOINC project servers using BOINC's WebRPCs. The WebRPC model assumes every RPC to be an HTTP GET transaction, the input parameters are represented as a set of parameterized GET arguments. The resultant output is an XML document with well-defined fields that is parsed by the Social Cloud to let users monitor their contributions and also to feed our social engineering algorithms described in the following section.

BOINC clients can be attached to the SoCPR in various ways: information about the SoCPR can be bundled with the client installer, or the user can specify the SoCPR as their account management system. In either case, the user will provide authentication details for the BOINC client to obtain project and resource share preferences from the SoCPR. The BOINC client communicates with the SoCPR using the BOINC Account Manager RPCs. Once the client has processed data relating to the projects that the user supports, it attaches itself to each of the project servers directly and starts pulling information for processing.

4.3. Social engineering

In the SoCPR we attempt to motivate individuals by tapping into their sense of competition, by playing to their desire for public recognition and by relying on social pressure from their friends. This is also known as *gamification*—the process of using mechanisms typically found in games to induce desired behaviors in a non-game application.

Given the social context created by the Social Cloud, users are able to compete directly with their friends—people they have far stronger social relationships with. This mechanism acts both to encourage increased individual contribution and also to encourage new friends to join in.

In the following sections we describe the structured incentives we have created to work towards the various goals of the SoCPR. At this point, it is important to distinguish between *the Social Cloud*, which is the overarching reference to the application itself and the various actors that actively participate, and *a user's Social Cloud*, which is the set of their friends that are members of the Social Cloud.

4.3.1. Reducing barriers for contribution

When a potential user arrives at the SoCPR Facebook application, there are a four main steps that they need to follow before they become a contributing volunteer:

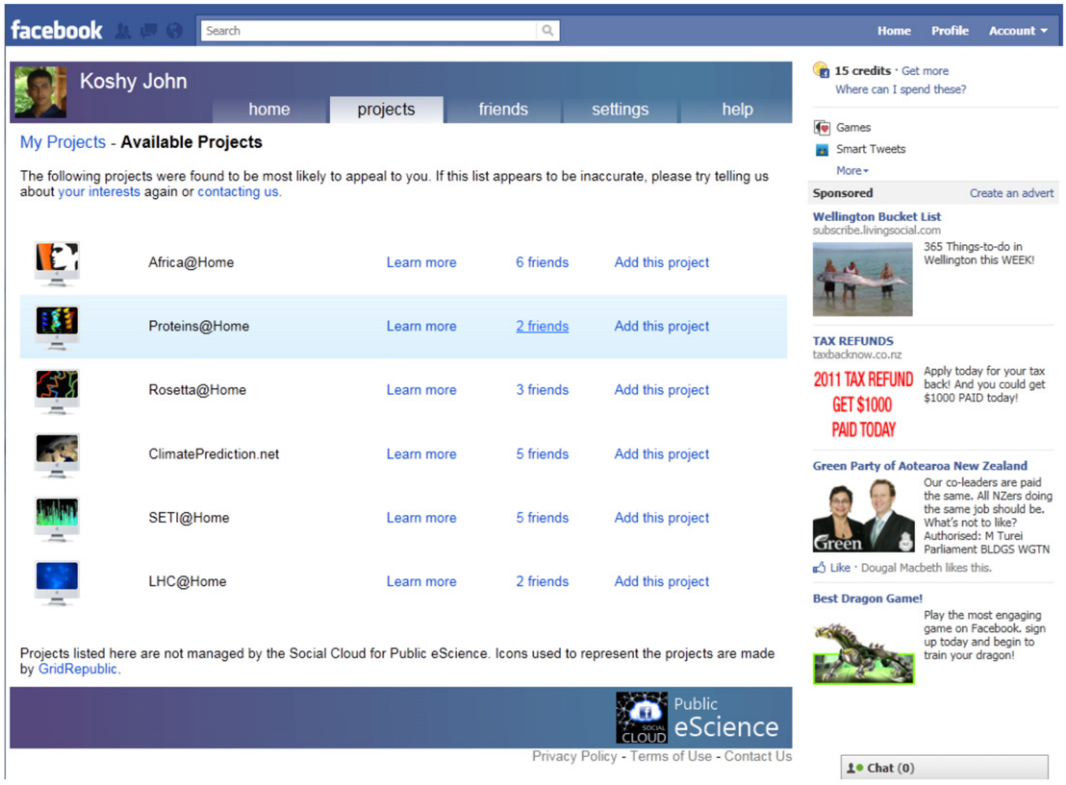


Fig. 7. A screen shot showing the project suggestions based on interest and project signature distances.

- *Understanding volunteer computing*
  The first step of understanding the volunteer computing model is critical for capturing the user's interest and keeping them motivated through the subsequent stages. From a social engineering perspective, this can include the use of compelling hooks from friends and providing various *calls to action* that motivate users to learn about the framework and its projects.
- *Selecting a project*
  To ensure that new users find it easy to select projects that match their own interests, we generate an *interest signature* for them based on their selections in a simple form. An interest signature defines a point in n dimensional space describing an individual user's specific areas of interest, each dimension represents a well-defined field of interest.
  The interest signatures that are obtained from users are normalized to enable reliable comparison. We calculate the interest signature distance, D_{uf}, as defined in Eq. (1), between a user, u, and all their friends, U_f, based on their interest signatures, I_u and I_f, to identify friends with the most similar interests to that particular user. The shorter the Euclidean distance between two interest signature points, the more similar the interests of the two users.

$$\forall f \in U_f, \quad D_{uf} = \sqrt{\sum_{i=0}^{n-1} (I_u[i] - I_f[i])^2}. \tag{1}$$

  We also introduce a *project signature*, which is the combination of all the interest signatures of the contributors to a given project. By calculating *signature distances*, we are able to highlight projects that were chosen by users, either globally or within the user's set of friends, that have interest signatures similar to the new user.
  We calculate the signature distance, D_{up}, between a user's interest signature, I_u, and a project signature, I_p, as follows:

$$D_{up} = \sqrt{\sum_{i=0}^{n-1} (I_u[i] - I_p[i])^2}. \tag{2}$$

  The calculation of project signatures allows the SoCPR to help users select projects that appeal to them.
  Another option for selection is allowing new users, who may not be sure of which projects to choose, to align their choices with those of their friends. Users that want a reliable personal opinion on a project are directed to *project champions*, large project contributors, within their group of friends.
  Fig. 7 shows a screen shot of the prototype SoCPR; in this figure the user is shown a list of BOINC projects that are most suited to their interests. This list of projects is selected based on the distance between the project signature and the user's own interest signature. Users are also shown the number of their friends that support each project, and may select projects (and resource shares) from this view.
- *Selecting resource shares*
  Once users select projects, they are required to select individual resource shares for each of the projects. A resource share is a reflection of the percentage of the computational time that a project gets on a user's machine. Users can select their own shares but we also provide recommendations based on normalized averages of the shares of their friends. This helps both to reduce technical barriers to entry and also to catalyze engagement by encouraging competition.
- *Installing the client*
  The final step of the process is to install and configure the client. In the SoCPR users are given instructions on installing the pre-configured BOINC client. Another advantage of the social network is that interested friends should be able to give technical advice in the event that a user needs assistance.

4.3.2. Incentivizing involvement, contribution and growth

We have developed a set of incentive mechanisms to encourage users to join and contribute to the SoCPR and therefore increase the computational time available to BOINC projects. We have introduced three high scoring sets of users—*project champions, social anchors* and *compute magnates* based off *project credits, social scores* and *compute scores* respectively. These labels are local to the view of each user and are reserved for friends that fall into the top bracket in a particular score. Project credits are routinely queried from project servers while social scores and compute scores are periodically recalculated using algorithms detailed in the following sections.

To motivate users and to enable them to measure their progress, we utilize leader-boards specific to each user's personal Social Cloud. Significant changes in positions on the leader-board are published to the user's Facebook wall—this in turn is expected to create interest in the application in addition to giving the user a sense of recognition.

There will be situations where there might not be enough data to work with, such as when a user with no friends joins the Social Cloud. In those situations, we fall back to values based off the Social Cloud as a whole. Likewise it is possible to compute the *best of the best* globally, although this is computationally expensive.

- *Project champions*: are the "biggest" contributors to a given project and therefore represent the best go-to person for a user from within their friends list if they need to know more about a project. Due to their high contributions they are expected to champion the cause of the project. Project champions are identified based on the total credits that they have earned contributing to a given project, these credits are termed *project credits*.
  It is easier to become a project champion of a less popular project, we therefore expect the desire to become a project champion will also increase the computational time that smaller projects receive. This is an important goal of the SoCPR.
- *Social anchors*: Given the underlying social nature of the Social Cloud, it is expected that the majority of users will be introduced to it by their friends, either explicitly (through an invitation or status message on their Facebook wall) or serendipitously, where a potential volunteer may chance upon a report of the contributions of their friends in their Facebook news feed.
  Existing Social Cloud users are incentivized to bring their friends into the Social Cloud via a *social score*. The users with the highest social scores are identified as social anchors.
  A user's social score represents a measure of their continuing contributions to the growth of the Social Cloud. Social scores are used to motivate existing users to encourage their friends to join the Social Cloud. New users with the least number of existing friends in the Social Cloud are of high value because they have more potential to bring in new users from new social clusters, hence the users responsible for recruiting them are rewarded with a comparatively higher boost to their social score. New users with a lot of existing friends in the cloud are of comparatively less value.
  To this end, every user has an associated social value, Sv_u, as defined in Eq. (3). The higher the number of friends, n_f, they have in the Social Cloud the lower their social value. Social values of a user's friends (in the Social Cloud), Sv_{uf}, add up to give the user their social score, Ss_u, as defined in Eq. (4). For a user to maintain a high social score, they would need to keep recruiting users who are less likely to join.
  Social Scores

$$\forall u \in U, \quad Sv_u = \frac{1}{n_f + 1} \tag{3}$$

$$\forall u \in U, \quad Ss_u = \sum Sv_{uf}. \tag{4}$$

  The social value of users who have increased the number of friend connections in the cloud since they joined will decay by virtue of that fact. This serves the dual purpose of disincentivizing users adding existing Social Cloud users as friends on Facebook to boost their social score and to ensure that users do not rest on their social score achievements.
  Social anchors are the key to growing the pool of users, and the title is recognition for their continuing contributions in this direction.
- *Compute magnates*: represent the top bracket of users generating computational time for the Social Cloud through their *own* Social Clouds. The computational time is based on the calculation of individual compute scores. This title was shaped to serve the dual purpose of incentivizing higher computational time contributions and application of social pressure on friends to maintain or improve on their contributions.
  Compute scores are a reflection of how much credit a user and their set of friends generate in a rolling 30 day window, C_{u30}. They are used to encourage users to ensure that their friends are contributing computational time with regularity. Like social scores, users are incentivized to focus on people with fewer friends in the Social Cloud, and disincentivized from adding friends simply to boost their compute scores.
  Compute Scores

$$\forall u \in U, \quad Cr_u = \sum C_{u30} \tag{5}$$

$$\forall u \in U, \quad Cv_u = \frac{Cr_u}{n_f + 1} \tag{6}$$

$$\forall u \in U, \quad Cs_u = \sum Cv_{uf}. \tag{7}$$

  The rolling credit value of a user, Cr_u (Eq. (5)), is used to generate their compute value, Cv_u, by dividing it by the number of friends, n_f, in their social cloud, as shown in Eq. (6). Compute scores, Cs, for every user are then generated by summing up the compute values of all their friends, as shown in Eq. (7).
  Because of the method of calculation, users who have friends that contribute less than what they (the users) stand to gain in terms of their own compute value may feel incentivized to remove those friend connections. But this would work only through breaking the Facebook friend relationship itself, and we feel that most relationships are strong enough for the user to work on getting their friend to contribute more rather than breaking the relationship.

4.4. SoCPR interactions

To help clarify the interplay between the architecture and social engineering we describe a basic scenario for the SoCPR through a sequence diagram in Fig. 8.

The process starts when a Facebook user discovers the SoCPR application. If they choose to add the application, permissions for the required user data are requested through Facebook and stored in the SoCPR application.

Once the application has been added, the user is presented a form to determine their interests in various research areas—this is used to generate their interest signature. The interest signature generated is then compared against the project signatures of all the available projects to determine appropriate projects ordered (by projected interest) for this user. The user can then select any projects that appeal to them and the Social Cloud proceeds to create accounts for them at each of the various BOINC project servers on their behalf.

Suggestions are made to the user on the resource shares that they should allocate to the various projects that they have selected. These suggestions are based on the normalized resource share

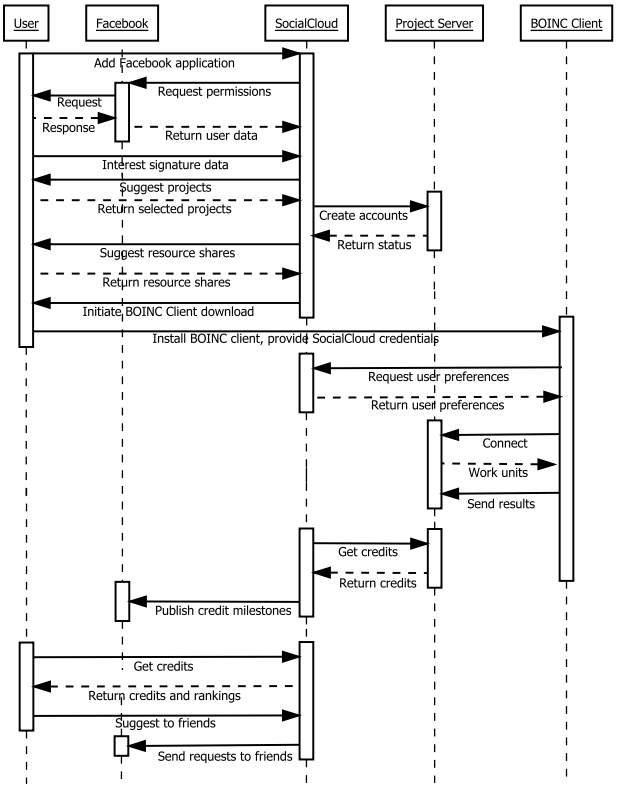


Fig. 8. A simple example of interactions between all the actors associated with the Social Cloud for Public eResearch.

values of their friends. This also allows for meaningful competition in the future.

The user is then prompted to install the BOINC client and provide credentials to connect to the Social Cloud. The BOINC client pulls information regarding the projects that the user has selected along with their resource shares from the Social Cloud. It connects to each of the project servers and downloads work units for processing. In due course, the results of the processing are sent back to the project servers. Each project server verifies the results obtained and grants credits as appropriate.

The Social Cloud routinely queries credits for every user from individual project servers. If a user is found to have achieved a credit milestone in a project, it is published to their Facebook wall. This is visible to friends and acts as a form of project advertising and reward. The user can also view the application at their convenience to check on their progress and that of their friends. They may also suggest the SoCPR application to their friends to help drive its growth (and increase their social score).

In addition, the Social Cloud periodically processes data available to it to establish rankings and achievements.

5. Evaluation

In this section we present performance measurements taken from a deployed SoCC prototype and analyze the effectiveness of the social incentive mechanisms in a simulated SoCPR environment.

5.1. SoCC

The prototype SoCC has been deployed as a Facebook application to experiment with the performance of creating and using VMs. In this section we focus on the performance of the two image distribution methods and the overhead of creating multi-VM clusters dynamically.

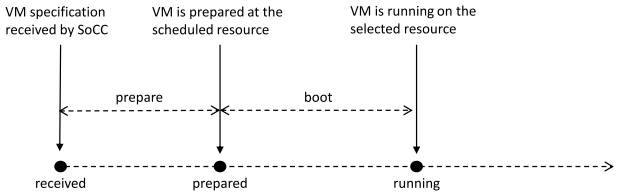


Fig. 9. VM instantiation states in the SoCC.

Table 1
VM instantiation timings (in seconds) for the two image distribution options.

	Custom image		Standard image	
	Prepare	Boot	Prepare	Boot
ttylinux	10	15	2	10
ubuntu 10.04	820	126	2	125

5.1.1. Image distribution

To evaluate the performance of VM image distribution we used two different sized Linux distributions—*ttylinux* and *ubuntu 10.04 × 86_64 server*. The ttylinux is a 32 bit distribution with an image size of approximately 40 MB. The ubuntu image is a 64 bit server distribution that is approximately 5 GB. The two image distribution methods (see Section 3.2.2) in the SoCC are:

1. *Custom image*: The user provides the URL of the custom image in the VM specification when requesting a new VM. The allocated resource then downloads the image from the specified URL before instantiating the VM.
2. *Standard image*: The user selects a standard pre-packaged image when submitting the VM specification. These images are cached locally to the physical machines.

The testbed for the experiment consists of a cluster made up of 3 machines (Core2Duo 3.0 GHz, 4G RAM, 250 GB hard-disk each) connected by a switched Gigabit Ethernet LAN. The SoCC application server is also located on a machine of the same configuration on the same LAN.

In this experiment we register a single compute resource to host the chosen VM, we then submit a single VM specification through the SoCC Facebook application and measure the time taken between various stages of preparation (schedule and load image) and booting the image (see Fig. 9). To measure the time taken to distribute the image (for option 1) we staged the user uploaded images to another machine, on the same LAN, before submitting the VM specification. The physical machine hosting the VM downloads the image from this machine via an HTTP connection before creating the VM. This base experiment was then repeated multiple times.

The measurements were taken at the SoCC application server by logging the times at which VM status updates were reported from the host resource. For each result, the average of five runs for each image distribution option is taken. The results are presented in Table 1 and Fig. 10.

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 LAN, the difference between the two methods can still be seen. The cost of transferring an image represents 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 the different preloaded packages included in the distributions.

Option 1 provides complete customization, however it is only suitable for small image sizes (due to the considerable overhead of image transfer). Option 2 is able to rapidly deploy a VM (as the image is already cached on the physical machine), however this comes at the expense of customizability and flexibility. One

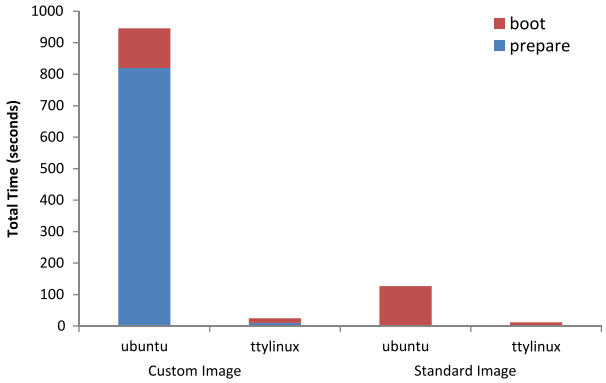


Fig. 10. Average times required to instantiate and run VMs within the SoCC.

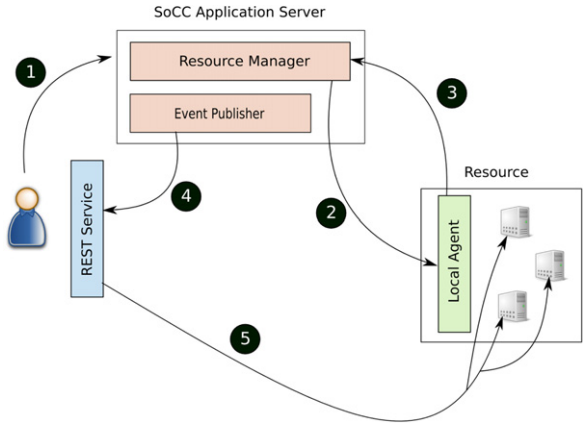


Fig. 11. Experimental architecture to create a multi-node virtual cluster in the SoCC.

potential solution is to cache a collection of common base images (potentially uploaded by users) and utilize customization scripts or a configuration management system such as Chef or Puppet to completely customize the deployed VM.

5.1.2. Virtual cluster creation

To evaluate the overhead of creating a virtual cluster in the SoCC we use the topology illustrated in Fig. 11. We use a predefined VM image that includes the Nimbus contextualization scripts required to configure the virtual network settings and interact with the other nodes in the cluster. The application includes an event publisher to create notifications and the user exposes a REST service to listen for notifications in the system.

The general flow of events used in this experiment to create a virtual cluster is:

1. User submits a request for *x* compute nodes in a virtual cluster.
2. The SoCC application schedules appropriate resources and forwards the request to the scheduled resource.
3. The Local Agent of the resource instantiates the VM and pushes status updates of the nodes to the SoCC application.
4. The application notifies the user, through their registered REST service, of status updates.
5. When the user has received the *RUNNING* state for all nodes, the contextualization scripts are executed on all nodes to configure the virtual network.

We utilize the same 3 node test bed, and set up a user REST service running on a similar machine within the same LAN. The VM image used in this experiment is the 5 GB ubuntu image. Fig. 12 shows the results of this experiment for both an uploaded customized image and a cached standard image. The difference in

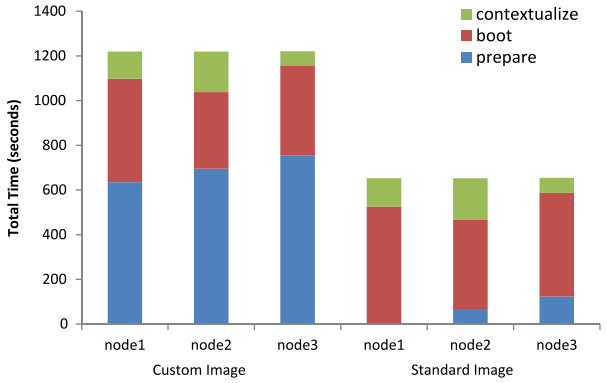


Fig. 12. Virtual cluster creation of 3 nodes using custom and standard images.

contextualization time is due to the order in which the nodes are added and depending on which node is the deemed the master node. The actual time to contextualize each individual VM is the same, the differences in the times evident in the graph are due to the customization scripts that must wait for all other nodes in the cluster to be configured—which is of course the reason all nodes experience the same total instantiation time. In addition, the base preparation times are also the same, but the differences experienced by the nodes are due to the sequential setup and submission of each VM specification by the scheduler. The overall timings in these results are greater than those shown in the earlier experiments in Fig. 10 due to the additional contextualization steps.

5.2. SoCPR

The SoCPR utilizes several novel incentive mechanisms based on the definition of different user categories (project champions, social anchors, and compute magnates). In this section we evaluate the effect of these incentive mechanisms on participation and resource contributions of users in the Social Cloud.

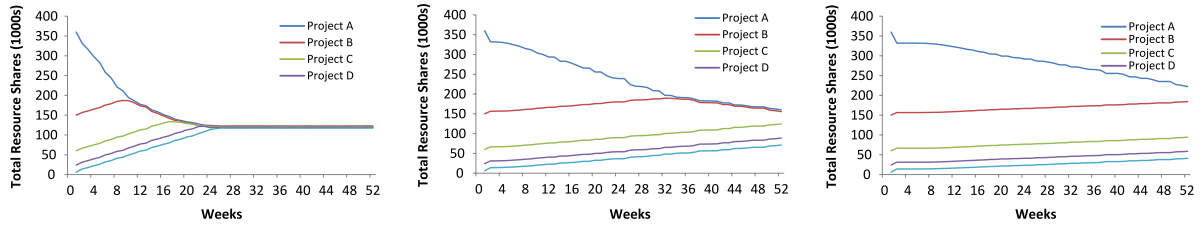
In order to study the effects of these incentives we utilize a standard social network dataset to perform simulations. The dataset is part of the Social Network and Geospatial benchmark[6] created at the University of Maryland. It has been used for the IEEE Visual Analytics Science and Technology (VAST) Challenge (2009). The dataset represents entities (people, cities, and countries) and relationships between the entities. The entire dataset includes 6000 individuals with 29,888 relationships. The average number of relationships for any person is 4, with a maximum of 449. The distribution of relationships is low, with over 50% of individuals having between 5 and 7 relationships. While this is a small scale model compared to the 800 million users in Facebook and the average social network size of 130, it provides a suitable model on which to base our analysis.

5.2.1. Project champions

Project Champions (PC) are defined to help smaller projects gain additional resources. When the PC incentive is enabled, users will try to become PC of as many projects as possible within their Social Cloud. Without the PC incentive resource shares would remain relatively constant, as is historically the case in BOINC.

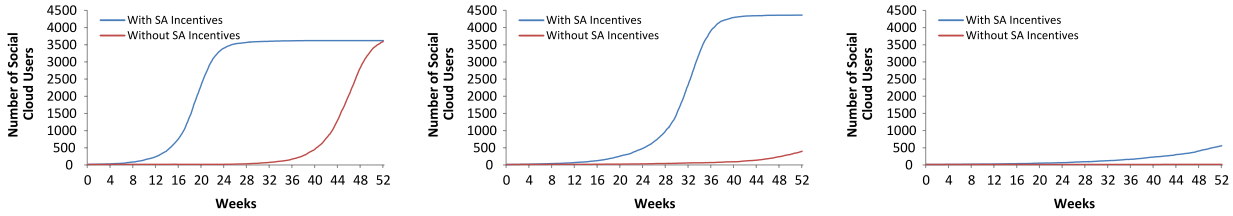
To evaluate the effect of the PC incentive, we have created a group of 5 different projects, with each having 60%, 25%, 10%, 4% and 1% of the total computational time available. Every user in the simulated dataset is randomly initialized with different resource

[6] http://hcil.cs.umd.edu/localphp/hcil/vast/archive/task.php?ts_id=119.



(a) $\lambda = 1$. (b) $\lambda = 5$. (c) $\lambda = 10$.

Fig. 13. Project champion simulations showing the project resource share distributions over a simulated duration of 1 year. The disinterest factor of each user is obtained from a Poisson distribution with different λ values.



(a) $\lambda = 3$. (b) $\lambda = 5$. (c) $\lambda = 10$.

Fig. 14. Social anchor simulations showing the growth of the Social Cloud over a simulated duration of 1 year. The number of requests each member of the social network has to receive before they join is obtained from a Poisson distribution with different λ values.

shares, such that the overall project allocations are maintained. To model a changing contribution environment we allow users to change their project shares once a week to increase their chances of becoming PC within their network—we model the probability of participation in a week as inversely proportional to a *disinterest factor* that is assigned following a Poisson distribution. The experiments presented in Fig. 13 show the total resource allocation to each project with disinterest factors obtained from Poisson distributions with λ values of 1, 5, and 10 respectively.

The results show that the PC incentive has the desired effect of evening out resource allocation across projects. Contributors change their contribution levels from larger projects to smaller projects in an attempt to become a PC. The results also show that the PC incentive is sensitive to user participation—low participation levels (higher disinterest factors) significantly increase the time taken for project allocations to converge.

5.2.2. Social anchors

Social Anchors (SA) are defined as users in a social network who recruit the most friends to the Social Cloud. To simulate the effect of the SA incentive we randomly define a set of users in the Social Cloud (leaving a set of users who are not part of the Social Cloud) and monitor the change of membership over time. Each experiment is run with and without the use of the SA incentive.

All experiments start with an initial state of 15 members, who are randomly chosen from the simulation dataset. The selection of the initial 15 was found to have no statistical significance on the final outcome of the experiments. The initial number of members was chosen as it represents the same percentage (0.25%) of the total simulation dataset (6000) as is the current number of BOINC users (2.2 million) to the current size of the Facebook user base (800 million).

To model different user behavior with respect to joining the Social Cloud we assign users a *coercion factor*, which represents a user's willingness to join the Social Cloud when asked. The coercion factor is simply defined as the number of requests before a user joins the Social Cloud. Again to model a range of values, we obtain the coercion factor from a Poisson distribution. Every time a user receives a request to join, the user's request count is incremented by 1. When the request count exceeds the coercion factor the user will join the Social Cloud. Every user that joins sends out a one time request to join to all their friends. Following that, once a week, every user in the Social Cloud requests one of their friends (that is not in the Social Cloud) to join the Social Cloud. When SA incentives are not applied, the user selects an eligible friend at random. When SA incentives are applied, the user selects a friend with the highest social value among all their friends who is not yet part of the Social Cloud.

Fig. 14 shows the Social Cloud membership over time, with and without SA incentives. In each of the graphs the SA incentives are shown to dramatically improve the rate at which the Social Cloud grows. While an increase in coercion level required to join slows the growth of the Social Cloud it impacts both situations similarly.

5.2.3. Compute magnates

The goal of the Compute Magnate (CM) incentive is to raise the combined contributions of all users in the Social Cloud by motivating large users to encourage their friends to increase their contribution.

To simulate the CM incentive we assign a contribution level to every individual in the simulation dataset. The contribution level is defined as a percentage of a user's available resources. When users are defined as active (they contribute more than 25%[7]) they approach between 1 and 3 friends in the Social Cloud every week and request that they contribute more (to raise their own compute scores).

To model active contribution and willingness to increase contribution we again use a *disinterest factor* that decays over time. A user's propensity to encourage their friends is related to their current level of contribution and their disinterest factor. A large contributor at a given time is considered to be more likely to encourage their friends to contribute, whereas uninterested users are unlikely to encourage their friends to increase their contributions. The selection of which friends to approach is based on the calculation of compute values as described in Section 4.3.2. Any increase in contribution is inversely proportional to the user's disinterest factor.

The experiments shown in Fig. 15 show the total user contribution when disinterest factors are drawn from a Poisson

[7] In BOINC only 12% of volunteers are actively contributing at any one time.

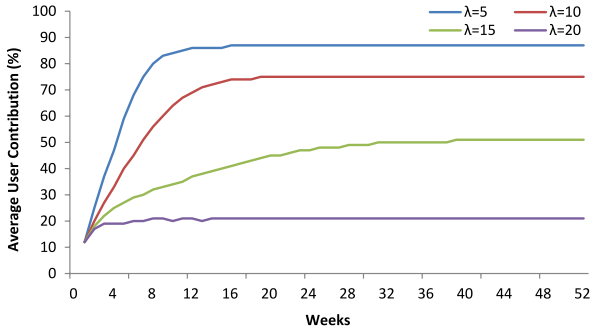


Fig. 15. Compute magnate simulations showing total user contributions over a 1 year period. The disinterest factor for an individual is drawn from a Poisson distribution with different λ values.

distribution with λ set to 5, 10, 15 and 20. The initial combined contribution level is set to the current active contribution rate in BOINC (12%). The graphs show that user contribution can be significantly increased through the use of appropriate social motivation.

6. Conclusion

Scientific collaborations are formed to share complementary resources to achieve a given goal. In many collaborations there is a requirement for computational resource usage that exceeds the available capacity for any individual user, therefore requiring the ability to share resources. In formal collaborations this process of sharing is well defined and formalized, however in ad hoc or dynamic collaborations no such agreements exist. 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 our previous work we proposed a unique resource sharing model called Social Cloud Computing, in which users, represented in a social network, are able to share resources with one another. This model builds upon a unique environment in which individuals are motivated by social incentives and trust is established through the external real world basis of relationships. The concept of a Social Cloud provides untapped potential to create a collaborative resource sharing framework for general scientific computing. In this article we presented two approaches to create a dynamic collaborative scientific computing environments.

The first approach uses a Social Collaborative Cloud (SoCC) to allow users to form dynamic VOs and share computational resources with one another through VMs (and clusters of VMs). The architecture and implementation presented demonstrates the viability of the approach and shows with the recent advances in virtualization that specialized collaborative computational environments can be created with minimal overhead. The experiments presented show that both user-defined customized images and standard cached images can be used, and that the prototype system is able to create larger scale multi-VM clusters that can meet the increasing computational requirements of ad hoc scientific collaborations.

The second approach, termed the Social Cloud for eResearch, aims to increase the uptake of public eResearch, or volunteer computing, through a set of social incentives. To increase participation and contribution we identified three different roles that incentivize users by rewarding contributors. The Social Anchor role rewards active recruitment of peers, Compute Magnates reward both an individual and their friend's contributions, and Project Champions represent users who contribute strongly to a specific project. In addition to these roles, we introduced the concept of an interest signature, which makes it easier for users to choose projects based on the interests of their friends. This will also contribute to the formation of communities embedded within the Social Cloud. The results of applying these incentive models in a simulated social network highlights the potential for increased participation and contribution using novel social-based incentives.

We are currently in the process of creating a production version of the SoCPR that acts as a BOINC Project Manager and integrates Facebook, BOINC project servers and clients. We envision that with the unique strengths of this approach to volunteer computing we will see significant increases in the processing power available to BOINC based projects. If we are able to recruit 1% of the current user base of Facebook to become BOINC volunteers, we will effectively double, or even triple, the present number of BOINC volunteers.

References

[1] R. Curry, C. Kiddle, N. Markatchev, R. Simmonds, T. Tan, M. Arlitt, B. Walker, Facebook meets the virtualized enterprise, in: Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference, EDOC'08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 286–292.
[2] Z. Guo, R. Singh, M. Pierce, Building the polargrid portal using web 2.0 and opensocial, in: Proceedings of the 5th Grid Computing Environments Workshop, GCE'09, ACM, New York, NY, USA, 2009, pp. 1–8.
[3] D.D. Roure, C. Goble, R. Stevens, The design and realisation of the myexperiment virtual research environment for social sharing of workflows, Future Generation Computer Systems 25 (2009) 561–567.
[4] G. Klimeck, M. McLennan, S.P. Brophy, G.B. Adams III, M.S. Lundstrom, nanohub.org: advancing education and research in nanotechnology, Computing in Science and Engineering 10 (2008) 17–23.
[5] K. Chard, S. Caton, O.F. Rana, K. Bubendorfer, Social cloud: cloud computing in social networks, in: Proceedings of the 3rd IEEE International Conference on Cloud Computing, CLOUD, Miami, Florida.
[6] K. Chard, K. Bubendorfer, S. Caton, O. Rana, Social cloud computing: a vision for socially motivated resource sharing, IEEE Transactions on Services Computing 5 (2012).
[7] D.P. Anderson, G. Fedak, The computational and storage potential of volunteer computing, in: Sixth IEEE International Symposium on Cluster Computing and the Grid, 2006, CCGRID 06, vol. 1, pp. 73–80.
[8] BOINCstats, BOINCstats | BOINC combined—Credits Overview, 2011. http://boincstats.com/stats/project_graph.php?pr=bo.
[9] Facebook, Statistics | Facebook, 2011. http://www.facebook.com/press/info.php?statistics.
[10] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: enabling scalable virtual organizations, International Journal of Supercomputer Applications 15 (2001) 200–222.
[11] I. Foster, Globus online: accelerating and democratizing science through cloud-based services, IEEE Internet Computing 15 (2011) 70–73.
[12] D.N. Tran, F. Chiang, J. Li, Friendstore: cooperative online backup using trusted nodes, in: Proceedings of the 1st International Workshop on Social Network Systems, SocialNet'08, Glasgow, Scotland.
[13] Z. Yang, B.Y. Zhaoy, Y. Xing, S. Ding, F. Xiao, Y. Dai, Amazingstore: available, low-cost online storage service using cloudlets, in: Proceedings of the 9th International Conference on Peer-to-Peer Systems, IPTPS'10, USENIX Association, Berkeley, CA, USA, 2010.
[14] J. Howe, The rise of crowdsourcing, WIRED Magazine 14 (2006).
[15] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, M. Leboisky, SETI@home-massively distributed computing for SETI, Computing in Science and Engineering 3 (2001) 78–83.
[16] A. Beberg, D. Ensign, G. Jayachandran, S. Khaliq, V. Pande, Folding@home: lessons from eight years of volunteer distributed computing, in: IEEE International Symposium on Parallel Distributed Processing, 2009, IPDPS 2009, pp. 1–8.
[17] A. Andrzejak, D. Kondo, D.P. Anderson, Exploiting non-dedicated resources for cloud computing, in: 12th IEEE/IFIP Network Operations and Management Symposium, NOMS 2010, Osaka, Japan.
[18] D. Anderson, BOINC: a system for public-resource computing and storage, in: Proceedings, Fifth IEEE/ACM International Workshop on Grid Computing, 2004, pp. 4–10.
[19] D. Kramer, M. MacInnis, Utilization of a local grid of Mac OS X-based computers using Xgrid, in: Proceedings, 13th IEEE International Symposium on High performance Distributed Computing, 2004, pp. 264–265.
[20] J. Venkat, Grid computing in the enterprise with the UD MetaProcessor, in: Proceedings, International Conference on Second Peer-to-Peer Computing, 2002, P2P 2002, p. 4.
[21] GridRepublic, Gridrepublic | gridrepublic | boinc volunteer distributed grid computing, 2011. http://www.gridrepublic.org/index.php?page=about.
[22] Intel, Progress Thru Processors | Facebook, 2011. http://www.facebook.com/progressthruprocessors.

[23] K. Keahey, I. Foster, T. Freeman, X. Zhang, Virtual workspaces: achieving quality of service and quality of life in the grid, Scientific Programming: Dynamic Grids and Worldwide Computing 13 (2005) 265–276.
[24] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, 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.
[25] J. Kay, P. Lauder, A fair share scheduler, Communications of the ACM 31 (1988) 44–55.
[26] K. Keahey, T. Freeman, Contextualization: providing one-click virtual clusters, in: Procedings of the 4th IEEE International Conference on eScience, eScience'08, pp. 301–308.
[27] A. Edmonds, T. Metsch, A. Papaspyrou, Open cloud computing interface in data management-related setups, in: S. Fiore, G. Aloisio (Eds.), Grid and Cloud Database Management, Springer, 2011, pp. 23–50.
[28] P. Darch, A. Carusi, Retaining volunteers in volunteer computing projects, Philosophical Transactions of the Royal Society A 368 (2010) 4177–4192.
[29] Unofficial BOINC Wiki, How to decide on Resource Share—Unofficial BOINC Wiki, 2011. http://www.boinc-wiki.info/How_to_decide_on_Resource_Share.
[30] C. Christensen, T. Aina, D. Stainforth, The challenge of volunteer computing with lengthy climate model simulations, in: Proceedings of the First International Conference on e-Science and Grid Computing, E-SCIENCE'05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 8–15.

Kris Bubendorfer is the program director for Networking Engineering and Senior Lecturer in the School of Engineering and Computer Science at Victoria University of Wellington. He received his Ph.D. degree, on mobile agent middleware, in Computer Science from the Victoria University of Wellington in 2002. His research interests include market oriented utility computing, social computing, digital provenance, and reputation.

Kyle Chard is a Senior Researcher at the Computation Institute, University of Chicago and Argonne National Laboratory. He received the Ph.D. degree in Computer Science from Victoria University of Wellington in 2011, having previously received the B.Sc. (Hons) degree in Computer Science and the B.Sc. degree in Mathematics and Electronics. His research interests include distributed meta-scheduling, Grid and Cloud computing, economic resource allocation, social computing, services computing, and medical natural language processing.

Koshy John completed his M.E. from Victoria University of Wellington in 2011 with his thesis on 'The Social Cloud for Public eResearch'. He is presently with the Cloud and Datacenter Management team at Microsoft in Redmond.

Ashfag M. Thaufeeg completed his M.E. from Victoria University of Wellington in 2012 with his thesis on 'Towards a Social Cloud Framework for Collaborative eResearch'. He is currently the lead developer in the Corporate Management Services, Information Systems Unit for the President's Office, Maldives.