Experiences in the Design and Implementation of a Social Cloud for Volunteer Computing

Ryan Chard
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Email: ryan@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*—Volunteer computing provides an alternative computing paradigm for establishing the resources required to support large scale scientific computing. The model is particularly well suited for projects that have high popularity and little available computing infrastructure. The premise of volunteer computing platforms is the contribution of computing resources by individuals for little to no gain. It is therefore difficult to attract and retain contributors to projects. The Social Cloud for Volunteer Computing aims to exploit social engineering principles and the ubiquity of social networks to increase the outreach of volunteer computing, by providing an integrated volunteer computing application and creating gamification algorithms based on social principles to encourage contribution. In this paper we present the development of a production SoCVC, detailing the architecture, implementation and performance of the SoCVC Facebook application and show that the approach proposed could have a high impact on volunteer computing projects.**

I. Introduction

Scientific computing projects are increasingly typified by a requirement for big data and high performance computation, the scale of which exceeds the resources available to many researchers. To perform large scale computations, researchers often purchase, assemble, or rent computation resources (clusters, grids, clouds, or supercomputers), for the duration of a particular project. However, this is potentially an expensive, and infrastructure intensive approach available only to large research groups. One alternative is the use of volunteer computing platforms, such as Berkley Open Infrastructure for Network Computing (BOINC) [1], to provide a low-cost model for resourcing large scale projects by leveraging resources contributed by the public. In this model the resources available to a given project is a function of the number of volunteers contributing at any point in time, it is therefore vital for the success of a project to attract and retain contributors.

In our prior work we proposed a novel resource sharing framework based on the relationships encoded in a social network – called a Social Cloud [2], [3]. A Social Cloud supports heterogeneous (virtualized) resource sharing between "friends" in a social network. The social network provides both an inferred network of trust (based on pre-existing relationships) and implicit social incentives for appropriate sharing. The Social Cloud model can also be leveraged to support a more immersive form of volunteer computing [4], incorporating attributes of social networks, such as the intuitive integrated application interface, potential peer-based advertising mechanisms, and gamification (using mechanisms found in games to promote desired behaviours in non-game contexts) opportunities amongst friends. This results in an increased contribution of resources to worthwhile research and/or public good projects.

In this paper we present and discuss our experiences implementing and deploying a 'first' production version of a Social Cloud for Volunteer Computing (SoCVC). We also highlight the benefits that can be obtained, both for users and projects, by exploiting a social fabric for volunteer computing. We believe this approach will increase the total capacity provided through volunteer computing platforms by using existing social networks as 1) a medium for interacting with volunteer computing projects, 2) a basis for leveraging social incentives to encourage contribution, and 3) an opportunity for projects to advertise, solicit and maintain contribution. The overarching goal of this work is to increase participation in volunteer computing projects by exploiting social incentives and the ubiquitous nature of social networks. Specifically, in this paper we describe the implementation of a production Facebook application that delivers a BOINC compliant account manager. This allows users to contribute resources to the BOINC platform while utilizing incentives based on relationships encoded in Facebook.

II. Volunteer Computing - Background

Since its introduction in 1996 volunteer computing has gathered a large number of users who have collectively contributed significant computing resources to a wide range of projects. Amongst the most well-established projects are SETI@Home [5] and Folding@Home [6] which search for extraterrestrial intelligence and simulate protein folding respectively. To reduce the overhead of project development, BOINC was developed as a generic volunteer computing platform. BOINC has since become the predominant volunteer platform on which many of the leading volunteer projects are built, including SETI@Home.

In the BOINC model users install and configure a BOINC client on their resources, they then create accounts with

978-1-4673-4466-1/12/$31.00 ©2012 IEEE

each project they wish to contribute to. To ease the task of contributing to multiple projects, users can use an Account Manager (AM) as a proxy to multiple projects. AMs simplify the perceived, and often real, difficulties of joining and contributing resources to projects by providing a single interface through which users can add, remove and modify resource shares across many projects. After selecting projects, users allocate *resource shares* (percentage of allocation) for each project. Using an AM, a user's BOINC client connects directly to an AM, rather than an individual project server, to retrieve project information and resource shares. The client then connects to each individual project server to request workloads and upload results.

III. Volunteer Computing - The Barriers

The major reasons for user contribution in volunteer computing systems can be loosely categorized into benefits related to the project (e.g. potential impact of the science, probability of success, and trust in the project) and personal benefits (e.g. a sense of community, competition, personal interest and even interesting visualizations)[1]. In general, users of volunteer computing platforms are thought to be motived by altruism, that is, they are willing to donate resources for little to no gain simply because they have spare resources or a project is seen to be a good cause. However, many are additionally compelled by a sense of competition between one another with respect to contribution levels. The majority of volunteer projects have attempted to exploit this notion of competition by utilizing gamification techniques such as the introduction of leaderboards.

From the point of view of the user, the technical limitations that make it difficult for the 'non expert general public' to contribute to projects are significant. For example, users have to discover and choose projects themselves, they are required to set up and manage client software to contribute to projects, and they have to create and manage accounts with multiple different projects, either through AMs or project sites themselves. From the view of the researchers, many small scale projects find it difficult to obtain access to sufficient resources, and while volunteer computing provides a cheap (or even free) alternative to harness large scale resources, it is still difficult to gain, and retain, a significant user base. In many ways this problem relates to issues advertising projects to appropriate (interested) users, simplifying the contribution model, and providing incentives to continue contribution. The difficulty motivating users can be seen in the current volunteer computing landscape as there are a few large, high profile projects and many less well known, small projects with only a few contributing users. The expense in engineering a BOINC application is considerable and it can be difficult to justify this cost for new, or less popular projects.

Lastly, volunteer computing is inherently a best effort computation paradigm that is, results are computed out of order as resources permit. In addition, volunteers are generally anonymous and unaccountable which makes the results untrustworthy – particularly when leaderboards and competition are introduced, which by its nature introduces cheating incentives. BOINC overcomes the problem of errors and cheating by adding redundancy to the results returned by contributors, however this adds significant overhead to the system and reduces effective computational power.

IV. A Social Cloud for Volunteer Computing The Benefits

There are three key ways in which the Social Cloud can benefit both users and researchers in volunteer computing platforms. Firstly, the integration of volunteer computing within the social network platform gives users a single place to install, configure and manage their contributions. The social network provides a single user identity (and password) and a web site that users frequently visit. It also provides a familiar web application in which users can discover and interact with the volunteer computing application. Secondly the pre-existing social framework provides a structured way to advertise projects, for example supporting implicit advertising channels such as social feeds. The social network also provides a huge user base through which the application will be exposed, this in turn will provide increased publicity for projects, better matching between projects and users (based on profiles and friend's preferences), and lower barriers for projects to increase their reach. Lastly, social engineering and gamification algorithms can be used, in particular, social engineering algorithms [4] can identify important nodes (people) in a social network and use these nodes to help increase and maintain participation through connected friends. Trust can be inferred from these social relationships, and gamification techniques will foster competition (and cooperation) between friends in the network that extend beyond the current model of global leaderboards.

V. Related Work

Social networking concepts have long been exploited in scientific projects. In many cases, web-based science portals leverage social networking principles to link researchers, developers and users in an effort to facilitate collaboration, MyExperiment.org [7] and nanoHub [8] are two such examples. Other scientific projects use social network pages, and sometimes even applications, as a means of publicizing and gathering support for their science, for example CERN's Facebook page[2] and PolarGrid's Facebook application [9]. These projects however, are generally aimed only at increasing publicity by trying to get people to "like" the research page, install the application, or contribute to the community, they do not utilize the network as a platform to harness resources nor do they exploit social engineering algorithms to motivate participation. More widely, social networks have also been used in crowdsourcing applications as a mechanism of distributing manual problem solving tasks to a wide group of users.

[1] http://www.boinc-wiki.info/How_to_decide_on_Resource_Share

[2] http://www.facebook.com/cern

There are several open source and commercial volunteer computing platforms and AMs available, for example Folding@Home [6], Distributed.net [10], Apple's XGrid [11], Univa's Grid MP [12], GridRepublic [13] and BOINCStats Account Manager (BAM!) [14]. While these platforms provide credits and leaderboards to incentivize contribution, none of them explicitly exploit social networking platforms or complicated social engineering algorithms.

The use of social network applications for volunteer computing has also been explored through applications such as Progress Thru Processors [15] and RenderWeb [16]. Progress Thru Processors utilizes Facebook to expand the reach of BOINC projects by advertising projects to users that may otherwise not be exposed to them. Although the application has had some success, we believe by fully utilizing the social aspects available within social networks we can further grow the number and contribution of users. RenderWeb sources volunteer resources from social media to create a volunteer rendering system, functioning within Facebook. Like RenderWeb, our Social Cloud for Volunteer Computing aims to gather donated resources from within a social network. However, our solution aims more broadly at the entire project set of BOINC and includes a strong focus on mechanisms to motivate users and drive the growth of the application.

VI. Architecture

The Social Cloud for Volunteer Computing (SoCVC) leverages the social networking "App" paradigm to integrate volunteer computing management within a social network platform. The SoCVC uses published Social Network APIs to add socially-aware functionality to the app (e.g. gathering friends' information and publishing to feeds) and volunteer computing platform APIs to act as an intermediary between users and projects. The high level architecture of the platform is depicted in Figure 1.

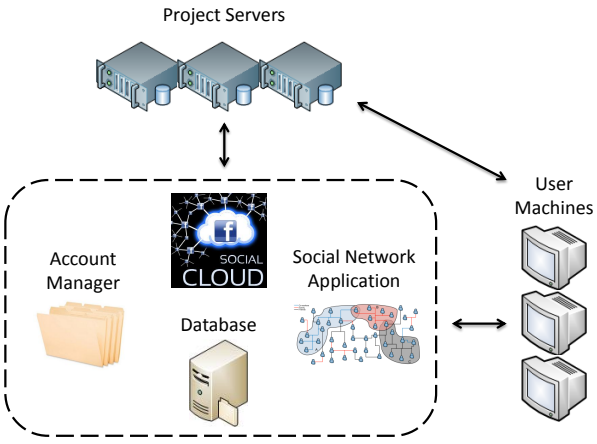

Fig. 1. The architecture for The Social Cloud for Volunteer Computing.

A. Architecture Components

The SoCVC is designed as a social network application to exploit the integrated look and feel of the social network, expose the standard app model for discovery and installation, provide a single location for users to manage and monitor their contributions, abstract authentication and authorization, and as a way of accessing social information (e.g. profile and connections). One major advantage of this approach is that users have a single identity and a single location to monitor their contribution over numerous projects. The SoCVC is also embedded within a web environment with which many users are familiar and visit often (e.g. over 50% of Facebook Users log in daily)[3], this will ease entry in to volunteer computing and encourage engagement.

Behind the application is a service-oriented architecture that manages users, their contributions to projects, and generates statistics and socially-oriented information (e.g. related to friends' contributions). Users, their preferences, and their contribution statistics are persisted by the SoCVC service. In order to interact with multiple projects for a single user, the SoCVC transparently manages user accounts with each project on behalf of a user. Periodically, project statistics are gathered for users and their projects. These statistics are stored and processed to produce statistics for the application on a per-user, per-friends, or global scale.

The SoCVC acts as an Account Manager and interacts directly with project servers on behalf of users through published APIs. The SoCVC also facilitates manipulation and creation of user accounts, monitoring of contributions, and retrieval of statistics from each project. The SoCVC service provides an interface to access individual and aggregated usage statistics for a particular user, which in turn allows the social network application to display statistics about users and their friends.

The volunteer computing client (installed on users' resources) communicates with the SoCVC through a standardized service interface. The protocol requires that the client and SoCVC authenticate one another before sending the list of projects, project credentials and resource shares to the client. The client uses this information to manage contributions and directly communicate with project servers. The SoCVC therefore acts only as a proxy to set up the contribution and is not involved in allocating workload and retrieving results.

B. Social Engineering

We have introduced several novel social algorithms that are designed to promote contribution and participation. The specific algorithms are described in detail in [4], and are summarized here for completeness.

Interest signatures have been developed to assist users when selecting projects. Interest signatures are generated on a per-user basis by allowing users to select high level topics that appeal to them. *Project signatures* are generated by calculating the average interest signature for a project from a group of users. These two signatures (along with the interest signatures of friends) are then used to match users with specific projects.

Identification of important nodes in a graph has long been a crucial part of social network analysis. In volunteer computing

[3]http://newsroom.fb.com/Key-Facts/Statistics-8b.aspx

projects, the largest contributors to a given project are clearly important nodes. In the SoCVC we term the largest contributor amongst a group of friends the *project champion*. The SoCVC can leverage project champions to motivate others to contribute more and to act as a contact point for technical assistance.

The connections between friends represent an important measure of a user's importance within a social network. To measure a user's social importance, the SoCVC calculates a *social score* for each user. The social score rewards users that promote the application and recruit friends to the platform. Within a group of friends, users with high social scores are termed *Social anchors* as they are directly responsible for the growth of the SoCVC.

Finally, *Compute magnates* are the users that provide, in total, the most computational value to the volunteer computing platform by encouraging others to participate. The purpose of this title is to acknowledge the efforts made to grow the contributions through the SoCVC by the top users, both within a group of friends and globally.

C. Interactions

All usage of the SoCVC requires users to authenticate with the social network by logging in using their standard platform credentials. Once authenticated, users are able to discover the SoCVC application through advertisements, referrals from friends (through messages and feeds), or searching for the app in the app list. The user is then able to install the application within their environment and can set permissions describing what access rights the application has (e.g. access to profile information or the ability to publish to news feeds). As a separate, out of band process, the volunteer computing middleware must be configured on every contributed resource in order to execute the work supplied by the project.

After installing the application and the client, the user is able to select projects and configure their individual contribution through the SoCVC application. The user is given complete freedom to select projects themselves, however suggestions are also provided based on their friends' selections, information gathered from their profile, and answers to questions posed during application installation. After selecting appropriate projects, the user is able to allocate a percentage of their total resources towards each project (a resource share) which is enforced by the SoCVC service. Users can return to the application at any time to manage and monitor their contribution, and manage preferences, modify resource shares, delete projects and monitor their usage statistics.

Usage statistics are computed periodically by the SoCVC on a global, local and friend basis. These statistics are used to feed social engineering and gamification algorithms, such as identifying important nodes amongst a group of friends and computing friend-based leaderboards. Depending on user preferences statistics and milestones will also be published in news feeds, status updates, and through the application.

VII. Implementation

The production implementation of the SoCVC, as shown in Figure 2, relies on the leading social network and volunteer computing platforms: Facebook and BOINC respectively. The implementation is composed of three distinct modules; the core Social Cloud, the BOINC Account Manager, and the Facebook application. Each of these modules provides unique functionality which, when combined, creates an integrated socially-oriented volunteer computing platform that reduces the complexity for users to contribute to BOINC and leverages social engineering algorithms to encourage contribution.

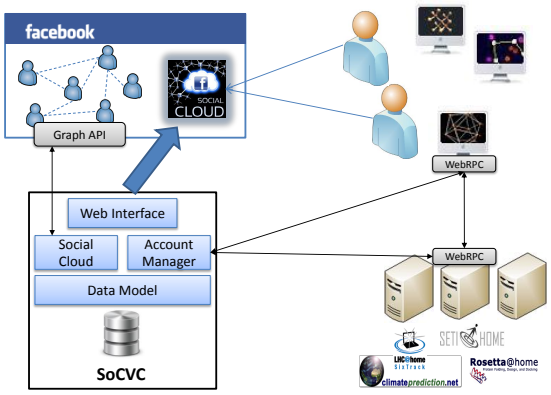

Fig. 2. SoCVC implementation. Users interact with the Facebook SoCVC App, social information is obtained through the Facebook Graph API. The SoCVC communicates with BOINC project servers using the WebRPC API and users' BOINC clients through the Account Manager API.

A. Facebook Application Paradigm

Facebook exposes access to its social graph through a simple RESTful interface called the Graph API. The Graph API includes methods to access objects in the graph and all the relationships between them, for example applications can access friends, events, groups, applications, profile information, and photos. In addition the same API allows information to be pushed to the social graph if user's permissions permit it, for example applications can write to news feeds and set statuses. Data sent and retrieved from the API is represented in a simple JSON format which is easily consumable in a variety of programming languages.

The Facebook Graph API uses OAuth 2.0 for authorization, thereby allowing users to set fine grained permissions over what access rights an application acting on behalf of the user has. Every Facebook application is identified by an Application ID and Key that is claimed when the application is first configured. These credentials are used to identify the application when accessing the Graph API. When a user installs the application, they are directed to the Facebook OAuth Dialog, the application includes its application ID to identify itself, a redirect URL to determine where to return to, and a list of permissions requested (e.g. access to streams or profile information). The user is then able to approve the request using the standard Facebook UI. If the user authorizes the application they will be returned to the redirect URL (in the application), Facebook also sends a signed request parameter that includes the user's id and an OAuth Token which must be used on all subsequent Graph API requests.

Facebook applications are hosted independently on external servers and are not hosted within the Facebook infrastructure. A canvas URL is created for the application within Facebook, this URL is mapped to the application's callback URL which is hosted remotely. When a page is requested by a user via the Facebook Canvas URL the Facebook server forwards the request to the defined callback URL. The application creates a page based on the request and returns it to Facebook.

B. Core Social Cloud

The core Social Cloud application is responsible for managing users' contributions to projects by interacting with the other components of the architecture. The core application is designed around a Model View Controller (MVC) architecture, where the model wraps a MySQL database, the controller is a PHP-based web application and the view is the HTML-based Facebook application.

The controller responds to events from the view and performs actions on behalf of users. It also exposes an Account Manager interface to allow BOINC clients to connect directly to the SoCVC. The application exposes a PHP-based service interface through which the Facebook application communicates with the controller. All information regarding users, projects, and statistics is stored in a database which is accessed by the controller through a data access model. When Facebook users install the application, the controller stores information about the user, their preferences (e.g. publication options) and the social network. The controller then acts as a gateway for coordinating contribution, retrieving statistics, and calculating the social algorithms.

The main role of the Social Cloud application is to manage users' contributions to projects. To do this the application communicates directly with BOINC project servers via the published BOINC WebRPC APIs. These APIs describe essential functionality and standardized interfaces that project servers must maintain in order for third parties to manage accounts and to retrieve statistics. The API includes functionality such as the ability to create user accounts, manage user preferences and retrieve contribution statistics. Through these APIs, the application is able to significantly reduce the complexity of users interacting with multiple BOINC projects. When a user adds a new project to their resource share, the Social Cloud application transparently creates an account on the appropriate project server. It also manages all existing accounts for users at each of their desired projects while providing a single sign on and a single unified view to end users.

The Social Cloud application is also responsible for monitoring and gathering information regarding user's contributions to each of the registered BOINC project servers. As there is no efficient subscription or notification interface exposed by project servers, the application relies on asynchronous Cron jobs to periodically contact each BOINC project, establish whether the project is currently running, and request contribution statistics for each user using the project. These statistics are then stored in the database for each user and are returned to the interface when requested. In order to provide a scalable and responsive architecture, calculations relating to these statistics, such as determining average contribution and running social engineering algorithms, are also executed at predetermined, low usage times, rather than on a per-request basis. The subsequent results are cached in the database and are returned to the interface when requested.

The controller also exposes a plug-in API through which various social engineering algorithms, such as those discussed in Section VI-B, can be implemented. This API provides the ability for algorithms to access information related to a user, their friends, and their contributions, and to store the results (e.g. metrics and rankings) through the extensible data model. The stored results are made accessible to the UI through the core application service. The algorithms run asynchronously within the same application scope, so as not to release potentially sensitive information.

C. Data Model

The SoCVC is responsible for storing a large amount of state related to users, projects, contributions, statistics, and the results of social engineering algorithms. We have designed a set of data models to store and access both information and policies. The data model layer abstracts all interactions with the respective data representation (i.e. database tables), establishing a fully encapsulated implementation which can quickly be developed upon while also allowing for trivial change of data storage mechanism. The data access layer is a PHP based model that acts as a proxy to abstract interactions with the MySQL database. The model also provides an auditable interface to monitor and log all communications with the database. The application relies on the MySQL database to store all information regarding users, friends, projects and their associated accounts, preferences, and contribution statistics.

D. Account Manager

The AM module of the SoCVC exposes an interface that conforms to the BOINC Account Manager API. This interface enables users' BOINC Clients, running on remote heterogeneous resources, to connect, authenticate and retrieve the user's selected projects, their delegated user credentials, and their preferences that have been assigned through the Facebook application interface, such as their resource share.

A user's client application must first authenticate with the SoCVC AM before retrieving project specific information. The protocol begins with the client requesting a description of the Account Manager. The response includes the name of the AM, a minimum password length, and an empty *<account_manager>* tag that signifies that the project operates as an AM rather than a BOINC project. After validating the response, the client submits a HTTP POST request which includes authentication details and any locally configured preferences, which are subsequently stored by the AM.

Once the client application is authenticated and the AM has matched the request to a given user account, the AM responds with its own public key and a set of BOINC projects associated with the user. The response may also include preferences,

regarding the AM and BOINC projects, which are then set by the client before execution of a given task. Each project element that is returned to the client is required to specify an authenticator which is used by the project servers to identify the user following a delegated authentication model. Authenticators are generated when creating a user account through a BOINC project's WebRPC. In order to mitigate the chance that a client connects to the wrong project server or is manipulated into performing work for a malicious project, the URL of the project server is signed by the AM. Rather than sign URLs when requested, the AM pre-signs URLs for every project server it interacts with as this significantly reduces overhead. Preferences defining project execution, such as the resource share or when to collect more work, are also included in the response.

In order to sign a project server's URL, the *crypt_prog* application has been compiled from the BOINC source code and is used directly in the SoCVC to generate RSA keys. Because the BOINC client is given the AM's public key during the authentication process, clients are able to evaluate these signed project URLs using the AM's public key and a local crypt_prog instance.

Finally, after the client application has authenticated with the appropriate project server, it is able to directly receive work units following the standard BOINC protocol. Specific configuration options defining project execution are passed by the AM and are stored by the client for the duration of execution. These preferences include information such as the specified resource share and when to collect more work.

E. Facebook Application

The final module of the implementation provides a Facebook Application Interface for users to interact with the SoCVC. The interface is written in HTML and relies on JQuery to perform active manipulation of the page. Communication with the SoCVC core uses AJAX to perform asynchronous requests for structured information that is then rendered in HTML.

The user interface, as shown in Figure 3 is separated into five distinct tabs. The *home* tab provides an overview of a user's contribution by showing the projects they currently contribute to, along with their respective contribution statistics. The *projects* tab provides an interface for managing contributions to BOINC projects, this interface allows discovery and selection of projects as well as configuration of project shares. The *friends* section allows the user to review the performance and relative standing of their friends with respect to one or more projects. *Settings* enables the user to review their preferences for using the SoCVC. Finally, the *help* tab is a source of information to assist users through the process of installing and getting started with volunteer computing.

F. Production Deployment

The SoCVC has been designed to be hosted on a scalable virtualized Cloud platform such as Amazon Web Services. The core components of the architecture are PHP-based and can be

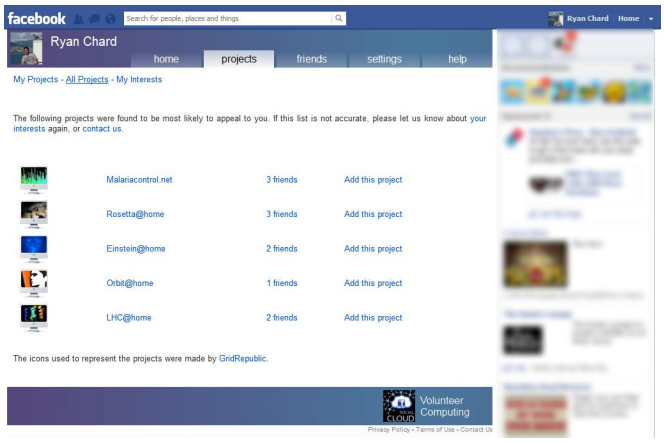

Fig. 3. The interface to The Social Cloud for Volunteer Computing's Facebook App. This screen shot shows the projects suggested, based on interest and project signatures.

hosted in any PHP compliant web server, we use the Apache HTTP server. Following the MVC paradigm, the controller and view are stateless and can therefore be replicated across multiple Cloud instances, if required, load balancers (such as Amazon's ELBs) will be used to allocate workload across this distributed pool of instances. The underlying database maintains all state and can be hosted on independent resources. The MySQL-based database model can also be deployed to Cloud-based elastic database services such as Amazon's RDS. During the initial phase of deployment we have adapted this deployment architecture using dedicated resources hosted within our institution. We have deployed a single instance of the Facebook application, core and database on a single host and will expand the deployment environment to utilize Cloud based resources as, and when, required.

VIII. Results

A. Database Performance

The database schema provides a fine grained approach to maintain information about users, projects, statistics and social networks. To evaluate the design of the data model Figure 4 shows the average time taken to create, load and remove the various data models used in the SoCVC. The figure shows a small constant deletion time, this is because the SoCVC marks accounts as inactive, rather than deleting them, as we assume that users may leave and later re-join projects. While this approach could create potential confusion for users we anticipate using garbage collection techniques to remove artefacts after a period of time. The figure also depicts a large discrepancy in the relative time to create a model and store it. The reason for the additional overhead associated with Project models is due to an initial validity check of the project URL.

B. Project Delegation

In order to measure the approximate overhead of communication between the SoCVC and individual projects, we evaluate the time taken to retrieve a response from each of

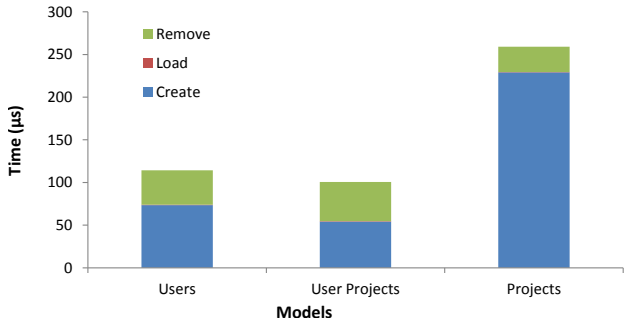

Fig. 4. The average time required to create, load and remove User, User_Project and Project database models. The time taken to load information is insignificant compared to the overhead of creating and removing data models.

the BOINC WebRPC functions used. Figure 5 shows the time taken for Rosetta@home[4] to respond to each WebRPC call. The figure shows a significant difference in query time for different requests. While the time taken for individual requests is small, when subject to scale it will be difficult to retrieve all information in real time. Instead we have used this information to calculate the approximate cost of retrieving different types of information and altered the SoCVC application to cache frequent and longer running queries. Overhead time is also used in determining the time required to execute Cron jobs to update user statistics.

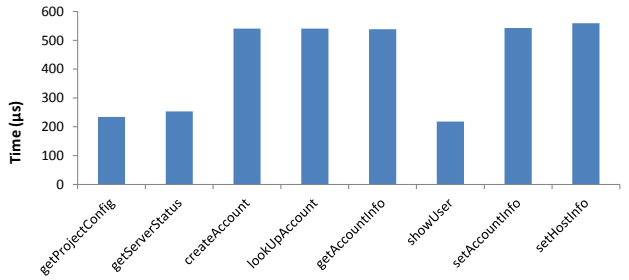

Fig. 5. The average time from Rosetta@home to respond to each WebRPC function call.

C. Account Manager Performance

The AM performs as an intermediary between the BOINC clients, running on a user's machine, and the BOINC project servers. The key role of the AM is to authenticate a user and form responses consisting of projects and preferences. As discussed in Section VII-D, the AM signs project URLs so that requesting clients can trust the URL is correct. Due to the substantial overhead associated with encryption, the SoCVC application pre-computes signed URLs for each project to optimize response time. Figure 6 shows the average time taken, with respect to the number of project preferences being delivered, for the AM to form a response to a client request. The figure shows that pre-signing URLs offers a considerable improvement in the time taken to deliver content when compared with real-time signing.

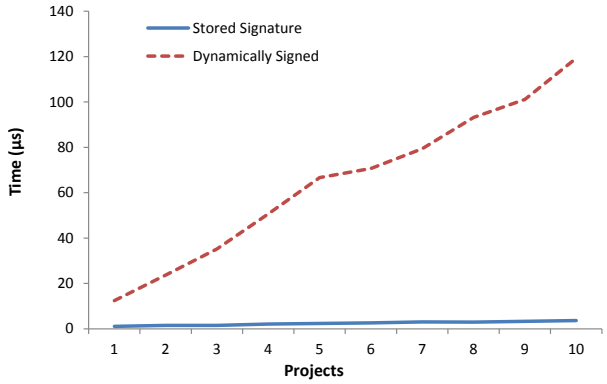

Fig. 6. The time for the Account Manager to create a response for a BOINC client, with either dynamically signed project server URLs or retrieving stored signatures.

D. Social Engineering

In order evaluate the effectiveness of the proposed Social Engineering algorithms discussed in Section VI-B, we have simulated each of the algorithms using a standard social network dataset. In this section we focus only on the Compute Magnate algorithm. The dataset used is part of the social network and Geospatial benchmark[5] created at the University of Maryland. It was also used in the IEEE Visual Analytics Science and Technology (VAST) Challenge (2009). The dataset includes 6000 individuals and 29,888 relationships between individuals.

Compute Magnates were proposed to motivate existing contributors to raise the contributions of their friends. In this simulation we assign a random contribution level to each user which represents the percentage of their resources that they contribute to the SoCVC. The initial combined contribution level is set to the current active contribution rate in BOINC (12%). In the simulation, active users (those that contribute more than 25% of their resources) will contact a random number of their friends in the network to encourage them to raise their contribution levels. We model contribution and willingness to increase one's contribution through 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, and a user's probability to increase their contribution when asked is also related to their disinterest factor. That is, a large contributor at a given time is considered to be more likely to encourage their friends to contribute to the SoCVC, and interested users are more likely to increase their contribution when asked. The magnitude of increased contribution is inversely proportional to their disinterest factor and directly proportional to the difference between their current contribution and their maximum contribution (100%)

[4]http://boinc.bakerlab.org/

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

Figure 7 shows the result of simulating a social network with Compute Magnates over a 1 year period. The figure depicts the growth of contributions with regard to varying degrees of *disinterest*, as modelled by a Poisson distribution with different λ values (5, 10, 15, 20). The figure shows that with lower disinterest factors (higher interest) that contribution can be significantly improved, it also highlights the potential for growth of the SoCVC when social engineering algorithms are used. Further analysis of the social engineering algorithms can be found in [17].

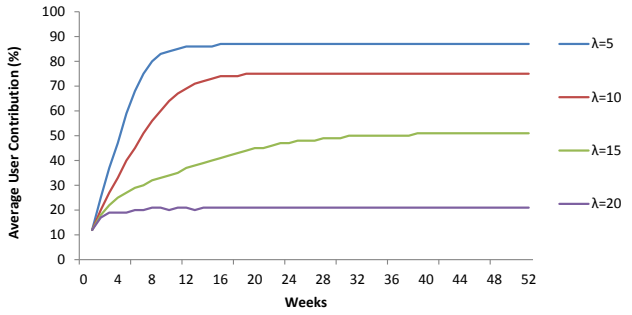

Fig. 7. Compute magnate simulations showing user contributions with a disinterest factor modelled with a Poisson distribution using different λ values.

IX. Conclusion

Volunteer computing provides an unparalleled scientific resource for many researchers, that is only limited by the number of volunteers contributing at any point in time. While there are a large number of active volunteer participants (e.g. 2.5 million BOINC users) this number pales in comparison to the number of users of social networks (e.g. 550 Million daily Facebook users). Moreover, attracting participants is just one of the problems facing volunteer computing projects as they must also compete to maintain participation and increase contributions from a static pool of expert contributors.

In this paper we have presented three key ways in which the Social Cloud can benefit both contributors and researchers in volunteer computing platforms. The integration of volunteer computing within the social network gives users a single user identity, familiar user interface, and a single application with which to configure and manage their contributions. In addition, the existing social network provides a structured way to advertise projects via channels such as feeds, tweets and so on. In our model, this infrastructure is also leveraged, using social engineering and gamification to build community and competition between users. Finally, existing social networks provide a huge user base from which the application and associated projects can recruit.

The Social Cloud for volunteer computing application has been implemented as an integrated Facebook application to leverage existing Facebook identities and provide a single location to interact with one's contributions. The backend architecture relies on a distributed service based model which interacts with Facebook directly to gather social information and also acts as a BOINC Account Manager to coordinate project accounts and resource contribution. The results included in this paper show the basic performance attributes of the Social Cloud application, and simulation results outlining the potential for growth of volunteer computing resources with the adoption of social algorithms.

Our future work includes the continued development of the application, and once released publically, an analysis of usage and further refinement of the social algorithms. In particular, we hope to use the platform as a unique source of data to analyze usage and contribution with respect to social constructs, we will use this information to develop and study new social engineering algorithms. In terms of the application itself, our focus will be on dissemination to broaden our user base. We will then concentrate on optimizing performance in light of scale.

References

[1] D. Anderson, "BOINC: A system for public-resource computing and storage," in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Nov 2004, pp. 4 – 10.

[2] 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.

[3] K. Chard, K. Bubendorfer, S. Caton, and O. Rana, "Social cloud computing: A vision for socially motivated resource sharing," *IEEE Transactions on Services Computing (Preprint)*, vol. PP, 2012.

[4] 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, Dec 2011, pp. 363–370.

[5] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky, "SETI@home-massively distributed computing for seti," *Computing in Science Engineering*, vol. 3, no. 1, pp. 78 –83, jan/feb 2001.

[6] A. Beberg, D. Ensign, G. Jayachandran, S. Khaliq, and V. Pande, "Folding@home: Lessons from eight years of volunteer distributed computing," in *IEEE International Symposium on Parallel Distributed Processing*, May 2009, pp. 1 –8.

[7] 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.

[8] 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, Sept 2008.

[9] 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)*, Nov 2009.

[10] "Distributed.net," http://www.distributed.net, 2012.

[11] D. Kramer and M. Maclnnis, "Utilization of a local grid of mac os x-based computers using xgrid," in *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, june 2004, pp. 264 – 265.

[12] J. Venkat, "Grid computing in the enterprise with the ud metaprocessor," in *Proceedings of the 2nd International Conference on Peer-to-Peer Computing*, 2002.

[13] "Grid republic volunteer computing," http://www.gridrepublic.org/, 2012.

[14] "Boincstats account manager (bam!)," http://boincstats.com/en/bam/, 2012.

[15] Intel, "Progress thru processors," http://www.facebook.com/progressthruprocessors, June 2011.

[16] A. Mcmahon and V. Milenkovic, "Social volunteer computing," *Jorunal of Systemics, Cybernetics and Informatics*, vol. 9, no. 4, pp. 34–38, 2011.

[17] K. John, "The social cloud for public eresearch," Master's thesis, Victoria University of Wellington, 2012.