Social Cloud: Cloud Computing in Social Networks

Kyle Chard,* Simon Caton,† Omer Rana,‡ and Kris Bubendorfer*
* School of Engineering and Computer Science, Victoria University of Wellington, New Zealand
{kyle,kris}@ecs.vuw.ac.nz
†Institute für Informationswirtschaft und Management (IISM), Karlsruhe Institute of Technology, Germany
simon.caton@kit.edu
‡School of Computer Science, Cardiff Univeristy, UK,
o.f.rana@cs.cardiff.ac.uk

***Abstract*—With the increasingly ubiquitous nature of Social networks and Cloud computing, users are starting to explore new ways to interact with, and exploit these developing paradigms. Social networks are used to reflect real world relationships that allow users to share information and form connections between one another, essentially creating dynamic Virtual Organizations. We propose leveraging the pre-established trust formed through friend relationships within a Social network to form a dynamic "Social Cloud", enabling friends to share resources within the context of a Social network. We believe that combining trust relationships with suitable incentive mechanisms (through financial payments or bartering) could provide much more sustainable resource sharing mechanisms. This paper outlines our vision of, and experiences with, creating a Social Storage Cloud, looking specifically at possible market mechanisms that could be used to create a dynamic Cloud infrastructure in a Social network environment.**

I. Introduction

Social networking has become an everyday part of many peoples' lives as evidenced by the huge user communities. Some communities even exceed the population of large countries, for example Facebook has over 400 million active users.[1] Social networks provide a platform to facilitate communication and sharing between users, therefore modelling real world relationships. Social networking has also extended beyond communication between friends, for instance, there are a multitude of integrated applications and some organizations even utilize a user's Facebook credentials for authentication rather than requiring their own credentials (for example the Calgary Airport authority in Canada uses Facebook Connect[2] to grant access to their WiFi network).

The structure of a Social Network is essentially a dynamic virtual organization with inherent trust relationships between friends. We propose using this trust as a foundation for resource (information, hardware, services) sharing in a Social Cloud. Cloud environments typically provide low level abstractions of computation or storage. Computation and Storage Clouds are complementary and act as building blocks from which high level service Clouds and mash-ups can be created. Storage Clouds are often used to extend the capabilities of storage-limited devices such as phones and desktops, and provide transparent access to data from anywhere. There are a large number of commercial Cloud providers such as Amazon EC2/S3, Google App Engine, Microsoft Azure and also many smaller scale open Clouds like Nimbus [1] and Eucalyptus [2]. These Clouds provide access to scalable virtualized resources (computation, storage, applications) through pre-dominantly posted price mechanisms. A Social Cloud, therefore, is a scalable computing model in which virtualized resources contributed by users are dynamically provisioned amongst a group of friends. Compensation for use is optional as users may wish to share resources without payment, and rather utilize a reciprocal credit (or barter) based model [3]. In both cases guarantees are offered through customized SLAs. In a sense, this model is similar to a Volunteer computing approach, in that friends share resources amongst each other for little to no gain. However, unlike Volunteer models there is inherent accountability through existing friend relationships. There are a number of advantages gained by leveraging Social networking platforms, in particular we gain access to huge user communities, can exploit existing user management functionality, and rely on pre-established trust formed through user relationships.

In this paper we outline our vision of a Social Cloud and describe our experiences with a prototype. The rest of the paper is organized as follows: section II outlines related research and some example applications that may use a Social Storage Cloud. Section III presents the design of a Social Cloud using Facebook as the Social network. A prototype Social Storage Cloud is described in section IV, before the evaluation of our prototype is presented in section V. Finally, section VI outlines future work and provides concluding remarks.

II. Related Work

There are multiple instances of Social network and Cloud computing integration. However, most examples use Cloud platforms to host Social networks or create scalable applications within the Social network. For example, Facebook users can build scalable Cloud based applications hosted by Amazon Web Services [4]. There is no literature related to creating a Cloud infrastructure leveraging Social networking as a means of dynamic user management, authentication, and user experience. Automated Service Provisioning ENvironment (ASPEN) [5] takes an enterprise approach to integrating Web 2.0, Social networking and Cloud Computing by exposing

[1] http://facebook.com/press/info.php?statistics – last accessed April. 2010
[2] http://developers.facebook.com/connect.php – last accessed April. 2010

978-0-7695-4130-3/10 $26.00 © 2010 IEEE
DOI 10.1109/CLOUD.2010.28

applications hosted by Cloud providers to user communities in Facebook. There are similar efforts in the Grid community to leverage Social networking concepts, communities, and mechanisms. PolarGrid [6] is one such example which extracts Social data using the OpenSocial [7] interface and relies on OpenID [8] for identification. Different Social networking functions are then incorporated in an application specific portal.

An alternative approach involves building a Social network around a specific application domain such as MyExperiment (www.myExperiment.org) for biologists and nanoHub (www.nanoHub.org) for the nanoscience community. MyExperiment provides a virtual research environment where collaborators can share research and execute scientific workflows remotely. nanoHub allows users to share data as well as transparently execute applications on distributed resource providers such as TeraGrid. These platforms highlight the types of collaborative scientific scenarios possible in Social networks, however they are not generic as they are focused on the communities they serve and lack the sizable user bases of Social networking platforms. Additionally, administrators need to create and manage proprietary social infrastructures and users require credentials for each network they participate in (unless they use OpenID). The same functionality can be realized using a Social Cloud deployed in an existing Social network. For example Social Storage Clouds can be used to store/share data and information (for example academic papers, scientific workflows, datasets, and analysis) within a community.

Volunteer computing is a distributed computing model in which users donate computing resources to a specific (academic) project. The first volunteer project was the Great Internet Mersenne Prime Search (www.mersenne.org) in 1996, however the term gained much exposure through the SETI@Home [9] and Folding@home [10] projects in the late 90's. These projects showed the enormous computing power available through collaborative systems. One of the most relevant Volunteer computing efforts is Storage@Home [11] which is used to back up and share huge data sets arising from scientific research. The focus of Volunteer computing has since shifted towards generic middleware providing a distributed infrastructure independent of the type of computation, for example the Berkeley Open Infrastructure for Network Computing (BOINC) [12]. Most Volunteer platforms do not define SLAs, typically users are anonymous and are not accountable for their actions (they are rewarded with different incentives however). In a Social Cloud context this does not suffice as users need to have some level of accountability. A more realistic model for this type of open sharing is a credit based system in which users earn credits by contributing resources and then spend these credits when using other resources. This type of policy is used in systems such as PlanetLab [13].

III. Social Cloud Architecture

The Social Cloud architecture presented is designed as a Facebook application, to make use of this widely used platform, development environment and API. In a Social Cloud, services can be mapped to particular users through Facebook identification, allowing for the definition of unique policies regarding the interactions between users. For example, a user could limit trading with close friends only, users in the same country/network/group, all friends, or even friends of friends. A specialized banking component manages the transfer of credits between users while also storing information relating to current reservations. A high level architecture of a Social Cloud is shown in Fig. 1.

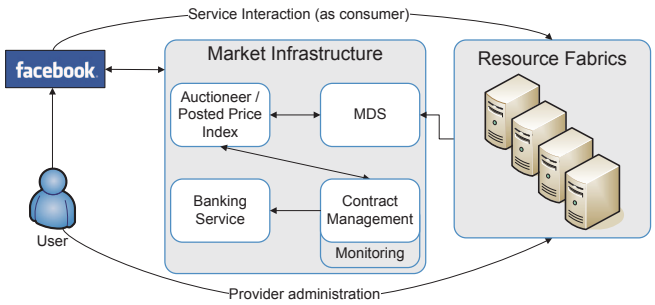

Fig. 1. Social Cloud Architecture. Users register Cloud services and friends are then able to provision and use these resources through the Social Cloud Facebook application. Allocation is conducted by the underlying market infrastructure(s).

A. Facebook Applications

Facebook exposes an application API through a REST-like interface which includes methods to get a range of data including friends, events, groups, application users, profile information, and photos. Facebook Markup Language (FBML) includes a subset of HTML with proprietary extensions that enables the creation of applications that integrate completely with the Facebook look and feel. Facebook JavaScript (FBJS) is Facebook's version of JavaScript – rather than sandboxing JavaScript, FBJS is parsed when a page is loaded to create a virtual application scope.

Facebook applications are hosted independently and are not hosted within the Facebook environment. A Facebook canvas URL is created for user access, this URL maps to a user defined callback URL which is hosted remotely. The process of rendering an application page is shown in Fig. 2. When a page is requested by the user through the Facebook Canvas URL (http://apps.facebook.com/socialcloud/) 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. At this point the page is parsed and Facebook specific content is added according to the FBML page instructions. The final page is then returned to the user. This routing structure presents an important design consideration in a Social Cloud context as access to the Cloud services would be expensive if routed through both the Facebook server and the callback application server in order to get data from the actual Cloud service. To reduce this effect FBJS can be used

to request data asynchronously from the specified service in a transparent manner without routing through the application server.

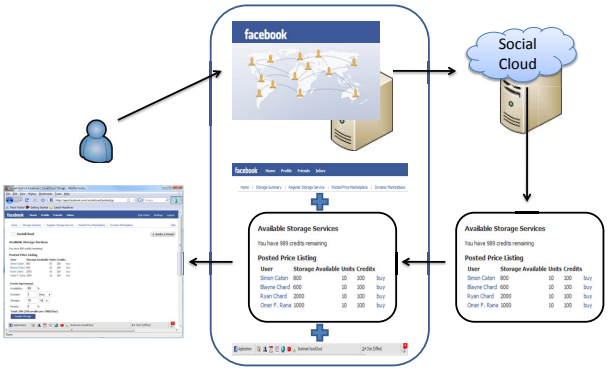

Fig. 2. Facebook application hosting environment. The Social Cloud web application generates page content which is parsed by Facebook to create the page delivered to the user.

B. Virtualized resources

Cloud computing relies on exposing virtualized resources as a service in a metered and elastic manner. A Social Cloud service could represent *any* resource that users may wish to share, ranging from low level computation or storage through to high level mash-ups such as photo storage. There are two generic requirements of this service: firstly, the interface needs to provide a mechanism to create a stateful instance for a reservation. In our model the Social Cloud application passes a SLA to the service which is parsed and used to instantiate the required state. Secondly, in order to be discovered the service needs to advertise capacity so that it can be included in the market. In our design this advertised capacity is XML based metadata which is periodically refreshed and stored in Globus Monitoring and Discovery System (MDS) [14].

C. Banking

The prototype Social Cloud includes a credit-based system that rewards users for contributing resources and charges users for consuming resources. The banking service registers every member of the cloud and stores their credit balance and all agreements they are participating (or have participated) in. Credits are exchanged between users when an agreement is made, prior to the service being used. To bootstrap participation in the Social Cloud, users are given an initial number of credits when joining the Cloud. While suitable for testing, this initial credit policy is susceptible to inflation and cheating (if fake users are created and the initial credits are transferred). Currently there is no mapping between Social Cloud credits and real currencies or Facebook credits.

D. Registration

Fig. 3 shows the registration process. – users first need to register themselves, and then specify the Cloud services they are willing to trade. As users are pre-authenticated through Facebook, user instances can be transparently created in the banking service using the users Facebook ID. Having registered, the user is presented with an MDS EndPoint Reference (EPR) and Cloud ID which they use to configure their service for registration (and refreshment) of resource capacity. Market services utilize the MDS XPath interface to discover suitable services based on user IDs and real time capacity.

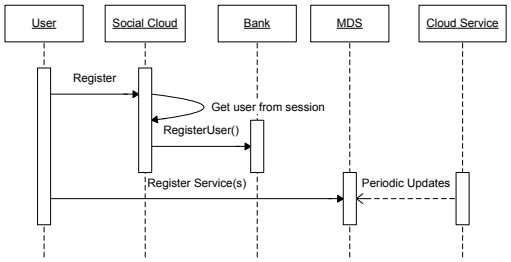

Fig. 3. Registration in a Social Cloud.

E. Service Marketplaces

Service usage is exchanged for credits within a marketplace. The Social Cloud marketplace is generic and not limited to a specific type of market, although two implementations are provided.

1) Posted Price: In a posted price model providers advertise offers relating to particular service levels for a predefined price or following a linear pricing function; consumers are then able to fine tune specific parameters to create a SLA. Creating such a market requires coordination between a number of the Social Cloud components to; discover Cloud services, create agreements, and transfer credits. Fig. 4 shows the flow of events for a posted price trade in a Social Cloud. When a user requests posted price offers the Cloud application uses the user ID (from the session) to check the user is registered in the bank and they have sufficient credits available. A list of all the users friends is generated using the Facebook REST API, this list is used to compose a query to discover particular Cloud services from MDS. The result of which populates the offer list that describes availability and pricing information. When the user selects a Cloud service, the Social Cloud application creates a SLA which it sends to the Cloud Service. Assuming both parties accept the agreement it is then passed to the Bank to transfer credits between users.

2) Auctions: In an auction-based market trades are established through a competitive bidding process between users or services. Like the posted price market, a list of friends is discovered and passed to a specialized auctioneer to create and run the auction. Fig. 5 details the auction process. In this example a reverse auction protocol is used, where Cloud

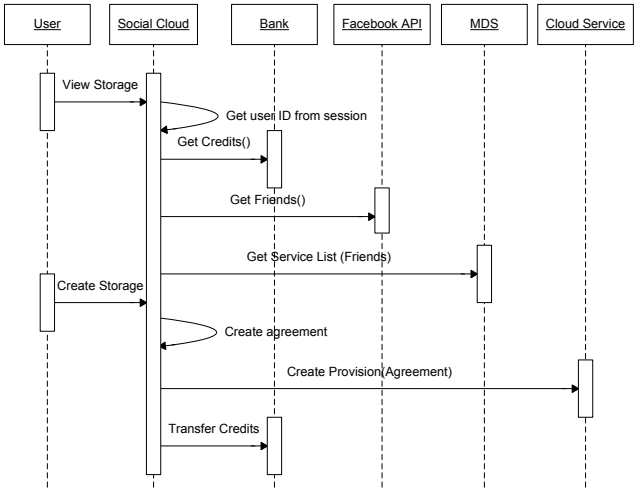

Fig. 4. Posted Price marketplace in a Social Cloud

services compete (bid) for the right to host the users task. The auctioneer uses the list of friends to locate a group of suitable Cloud services; these are termed the bidders in the auction. Each provider requires an agent to act on its behalf to value resource requests, determine a bid based on locally defined policies, and follow the auction protocol. The auctioneer determines the auction winner and creates a SLA between the auction initiator and the winning bidder. As in the posted price mechanism, the agreement is sent to the specified service for instantiation and the bank for credit transfer.

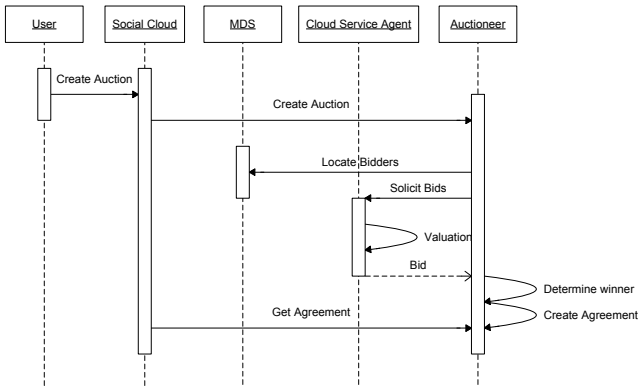

Fig. 5. Auction marketplace in a Social Cloud. This diagram excludes the actions taken to find the users' ID, retrieve the users' friends, instantiate the Cloud service, and transfer Credits these actions are shown in Fig 4.

F. Risk

Although there is a level of trust between participants in a Social network, this trust may not be sufficient in some situations. Take for example a storage service, where consumers are risking loss, compromise, or corruption of files while providers are risking their own environment by hosting unknown files. In a Social Cloud users may want to take into consideration this lack of control over corresponding users' actions and attempt to minimize risk. In the storage example providers can alleviate risk through service design and sandboxing, on the other hand consumers can avoid compromising file content through encryption and reduce the impact of file loss through replication. This raises the possibility of automatically managing such approaches, and offering premium differentiated services such as replication. For example Storage@home [11] includes a level of redundancy to minimize the risk of loss.

IV. Implementation

The Social Cloud prototype utilizes Web Services to create a scalable, distributed and decentralized infrastructure. All services use Web Service Resource Framework (WSRF) and run on Globus WS-core/Tomcat. The Facebook application is a JSP based web application. Two concurrent economic markets have been implemented to trade storage, both operate independently and are designed to work simultaneously. In a posted price market users select storage from a list of friends' service offers. In the auction market, consumers outline specific storage requirements and pass this description to the Social Cloud infrastructure, providers then bid to host the storage. Both markets result in the establishment of a SLA between users. The SLA is redeemed through the appropriate storage service to create the storage instance. In the market implementations participating users know the corresponding users identity, to provide accountability between friends. In traditional Cloud environments users are unaware of the location of their provision, the prototype Social Cloud could provide this transparency by removing user information from posted price listings, auction requests, and storage access.

A. Tools and Systems

Our implementation utilizes various tools and systems we have developed previously. In particular gRAVI (Grid Remote Application Virtualization Interface) [15] was used to create a base storage service, SORMA [16] is used to create WS-Agreements, and DRIVE (Distributed Resource Infrastructure for a Virtual Economy) [17] provides the auction framework.

B. Storage as a Service

A Social Storage Cloud is based on a generic Storage service which provides an interface for users to access virtualized storage. This service exposes a set of file manipulation operations to users and maps their actions to operations on the local file system. Users create storage by passing an agreement to the storage service, this creates a mapping between a user, agreement, and the storage instance. Instances are identified by user and agreement allowing individual users to have multiple storage instances in the same storage service. The storage service creates a representative resource and an associated working directory for each instance. The resource keeps track of service levels as outlined in the agreement such as the data

storage limit. Additionally the service has interfaces to list storage contents, retrieve the amount of storage used/available, upload, download, preview and delete files.

Each storage service relies on a Web application to deliver content to the Facebook application without routing data through the Social Cloud application. To do this the storage service has a collection of JSP pages that perform a specific action (corresponding to the service) and deliver a response in the form of JSON. This approach allows dynamic Ajax invocation of storage operations without requiring a callback or page reload of the Social Cloud Application.

C. Banking Service

The banking service manages user and agreement information. The service itself is composed of two associated context services each representing different instance data. The first context service records user resources while the second stores agreements. The user resource stores the user's Facebook ID, current credits, agreement IDs the user has participated in, and auction references (EPR/ID). The agreement resource contains any agreements created in the system which are used to manage provision information as well as acting as a receipt. Fig. 6 shows a summary page generated by querying the banking service, this page displays current and historical agreements with other users. It includes both storage provided and consumed, and information corresponding to each reservation.

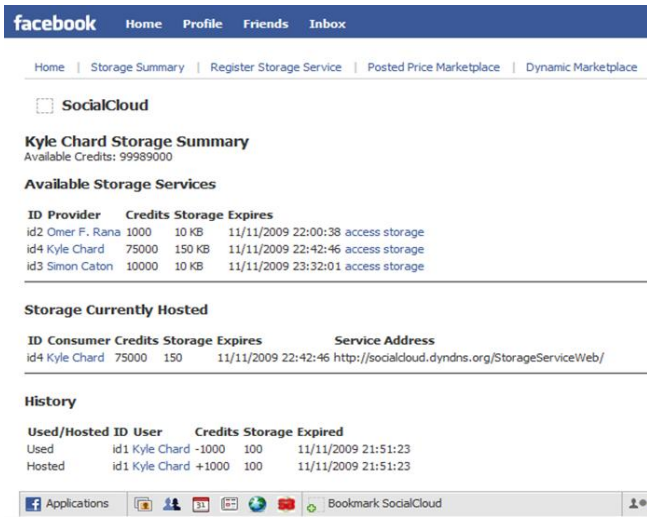

Fig. 6. Summary of services used and hosted

D. Posted Price Marketplace

In a posted price marketplace a user can select any advertised service and define specific requirements (storage amount, duration, availability, and penalties) of the provision. Fig. 7 shows the posted price marketplace page in the Social Cloud application. When the user selects a service and chooses required service levels, a SLA is created using the SLA creation component of SORMA. To do this the requirements are encoded into an EJSDL [18]; (JSDL [19] with economic extensions) document describing the storage request. This document is then converted into an agreement using SORMA SLA tools. The EJSDL document acts as the Service Description Term of the agreement and individual requirements are split into guarantee terms (as defined in [18]). EJSDL extends JSDL by adding additional economic information describing pricing and penalties which are mapped to their respective Business Value Lists. We have further extended this term language to include two additional Cloud specific QoS terms: Availability and Error Rate, which are defined as JSDL ranges and are used to describe and monitor the availability of the storage service. The SORMA reservation specification is used to capture the period of a service provision. Having created a SLA it is passed to the appropriate storage service to create a storage instance. The storage service determines if it will accept the agreement based on local policy and current resource capacity. Having instantiated the storage the agreement is then passed to the banking service to exchange credits and store a copy as a receipt. If either the banking service or storage service decline the agreement both entities remove the reservation.

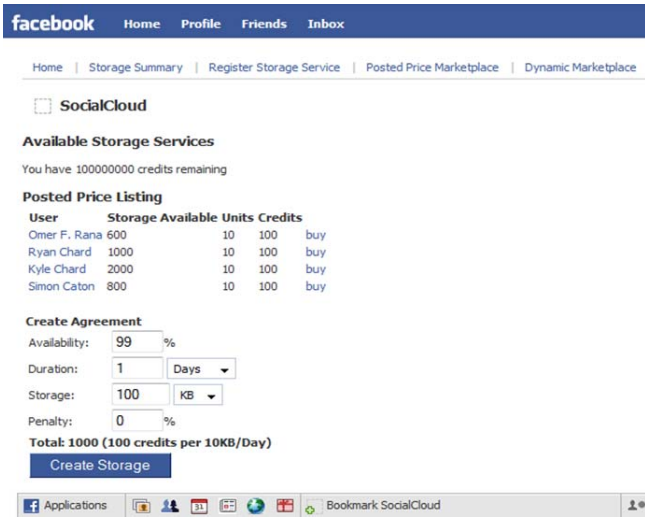

Fig. 7. Posted Price marketplace

E. Auction Marketplace

The prototype implementation uses of a subset of DRIVE services to provide an auction framework. A reverse Vickrey auction protocol is used due to the existing trust relationships within the Social network. Fig. 8 shows the dynamic auction marketplace, which lists currently running auctions. New auctions can be started by specifying required service levels, which are used as the basis for valuation and bid computation. When the auction completes a SLA is created between the user and the winning provider, the state of the SLA and the final price paid is displayed on the list of current auctions.

In DRIVE, an Auction Manager (AM) is responsible for creating the auction, soliciting bids, and determining a winner. Individual Bidding Agents (BA) act on behalf of a user to compute valuations according to local policy and valuation functions. The standard DRIVE BA has been modified to interact with the Storage Service (to check capacity). A Contract Manager (CM) is used to create a WS-Agreement based SLA between the user and auction winner.

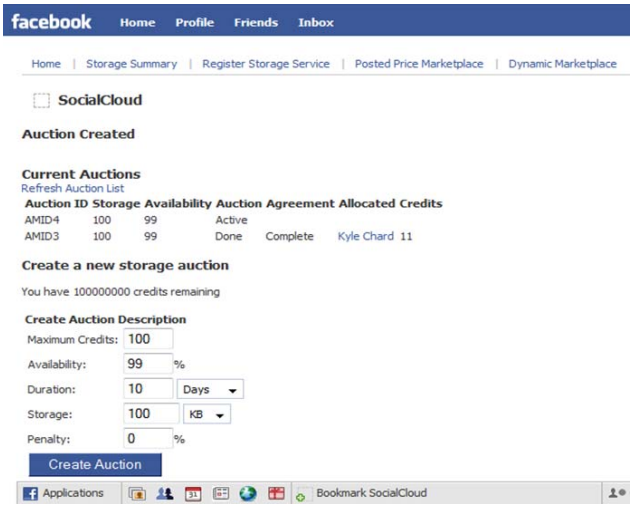

Fig. 8. Auction marketplace.

V. Evaluation

This section outlines measurements obtained from a deployed Social Storage Cloud focusing on the economic marketplaces used and the Facebook application load time. For the following experiments we assume an average Facebook user has 130 friends.[3] Unless otherwise stated these experiments are run on a single server running Windows Vista with a 2.2 GHz Dual Core processor and 2 GB memory.

A. Posted Price Allocation

Posted price trading requires several steps: identification of storage requirements, generation of a SLA, instantiation of a storage service, and registering the transaction with the banking service. The time taken to perform these operations is generally small compared to the time taken to locate applicable storage offers, which is dependent on the registration service. The prototype uses MDS to register and discover XML-based metadata describing the real time capabilities of a storage service. Fig. 9 shows the time taken to query MDS for an increasing number of registered entries. The time includes the cost of converting the XML result into a Java Object. MDS performance is dependent on the amount of memory given to the container and the number of registered entries. With 1GB of memory over 2000 entries can be retrieved in less than 2 seconds. Therefore, MDS can be run even on a low specification server yet still support a small Social Cloud and its market.

[3]http://facebook.com/press/info.php?statistics – last accessed April. 2010

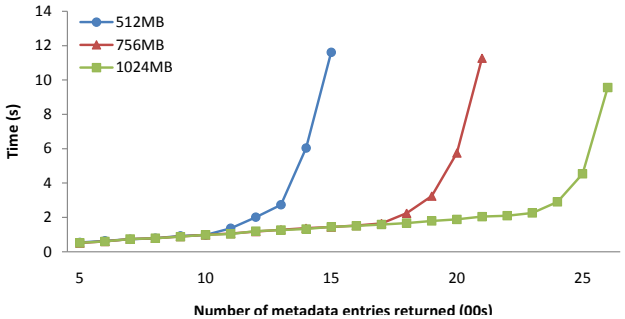

Fig. 9. Time taken to retrieve service metadata from MDS with different amounts of container memory.

In a Social Cloud environment policies dictate the services with which a user is willing to interact with (e.g. friends, friends of friends). Such services can be identified by querying for registered services matching particular user IDs (i.e all friends' IDs). Fig. 10 reflects this situation by loading an increasing number of services in MDS and querying for a subset of registered services (friend's services). The query result ranges between 20 and 200 services, while the number of registered services is increased from 200 to 2000. The container is running with 1GB of memory. The time taken to retrieve entries is proportional to the number of registered services and also the number of services returned in the query. Assuming on average 130 friends per user, and the fact not all of these friends would be involved in a Social Cloud, this performance is acceptable - selecting 100 of 2000 registered entries takes approximately 2s.

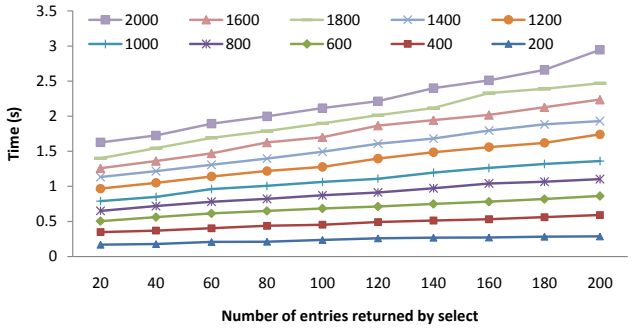

Fig. 10. Time to select a subset of the registered service metadata from MDS with increasing number of total registrations

B. Auction Allocation

The auction mechanism relies on a collection of Web services representing the parties involved in the marketplace. The prototype uses a single Auction Manager (AM) to conduct

the auction and a single Contract Manager to create SLAs as a result of the auction. Each storage service is represented by a Bidding Agent which consults local policies to determine a price based on pre-defined metrics. The major point of stress in this system is the AM as it is responsible for creating an auction, advertising the auction to suitable bidders, soliciting bids, and determining the result of the auction. Contract creation is much simpler as it only involves creation of a WS-Agreement and one call to the winning bidder to verify the agreement. The performance of the AM is dependent on the number of concurrent auctions and the number of bids being placed in each auction. Fig. 11 shows the system throughput with an increasing number of bidders in each auction. The number of auctions per minute is calculated based on the time taken for 500 auctions to complete, this time is measured on the client side. The time starts when the client submits the first task (of 500) through to the creation of the final agreement. Bidders are hosted in a virtualized environment containing 5, 3.0 GHz Core 2 Duo machines with 4GB RAM. Fig. 11 represents the worst case situation when all auctions are started immediately; auctions close as soon as all bidders have bid. In a typical scenario auctions are created with a predefined deadline and users expect some latency between submission and agreement creation. Additionally in a storage context one would expect relatively long term stable reservations which implies users would not conduct auctions frequently. These results show that even with 50 bidders a small scale AM can complete 65 auctions per minute which, under our assumptions, would be capable of supporting a large scale Social Storage cloud. This number could easily be increased by adding additional AMs to the system, which would be run independently on dedicated hosts.

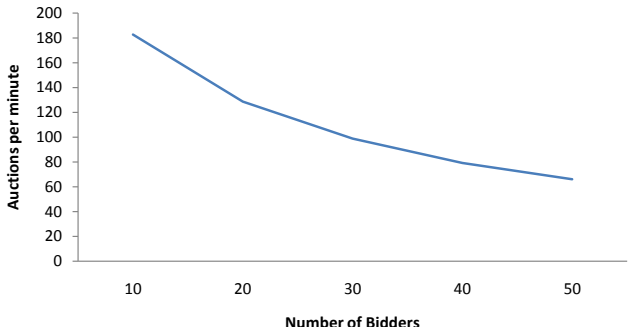

Fig. 11. Auction throughput. Number of auctions completed per minute for an increasing number of bidders.

C. Application Page Load Time

The Social Cloud Facebook application is made up of several different web pages, each page makes multiple Web services requests to generate page content. To measure the cost of this infrastructure we can compare average page loading times between the Social Cloud pages and simple HTML pages without Social Cloud Web service calls. In our testbed the application is hosted in a Tomcat 6 container running on a single server (Windows Vista, 2.2 GHz Dual Core processor, 2 GB memory) Each request is submitted from a separate machine (with the same specifications) also located locally and page load time is measured using Mozilla Firebug.[4]

Fig. 12 shows the time taken to load various pages in the application. Each page queries the appropriate Social Cloud services and renders a page displaying the results. Two additional sample pages (Control JSP/HTML and simple JSP/FBML) are included to compare load times with the Social Cloud application pages. The bars show the time taken to load the main page content and the additional overhead of other requests for Facebook related content. The main request includes the application page content as well as some Facebook content such as the navigation bar, header, and advertising. All results are measured using caching so as to replicate normal user experience

Fig. 12 shows there is considerable time taken to load a simple page in a Facebook application hosted in our environment. Over 3 seconds is required to load a "hello world" HTML/JSP page containing only a few lines of code that display one line if the user is logged in (a single session variable check). The Simple FBML page is designed to analyze the cost of FBML parsing, in this page a single line of FBML is rendered if the user is logged in. This page is only slightly slower than the plain HTML page due to the additional Facebook parsing. The time taken to render these pages is based on the cost of routing requests through the Facebook server, and latency between the client, Facebook server, and application server.

The time taken to load the Social Cloud pages is a little more than the plain HTML pages due to the extra time taken to make Web Service requests for information. Posted Price (PP) Listing and Storage Summary are the least expensive operations as they only make a single Web service call to retrieve a list of registered providers and a list of used storage services respectively. The Posted Price (PP) creation page takes over 4 seconds on average to load, due to the time taken to create a WS-Agreement, register it with the banking service, and then create the storage instance. Listing auctions and creating auctions take the longest time as they involve multiple interactions with Social Cloud services. The auction marketplace pages are slow due to the complexity of generating the pages.

The substantial page load time in the initial prototype greatly effects user experience. However, most of this overhead is due to the routing and parsing process required to generate a Facebook page in our environment. The obvious solution to this problem is to reduce the frequency of page loading and make asynchronous direct calls for content using Ajax. Fig. 13 shows the loading times for each of the main pages when converted to Ajax calls. The initial page load is also shown as it has increased from the previous results due to the additional time taken to parse and load Javascript. This graph

[4]http://getfirebug.com/ – last accessed April. 2010

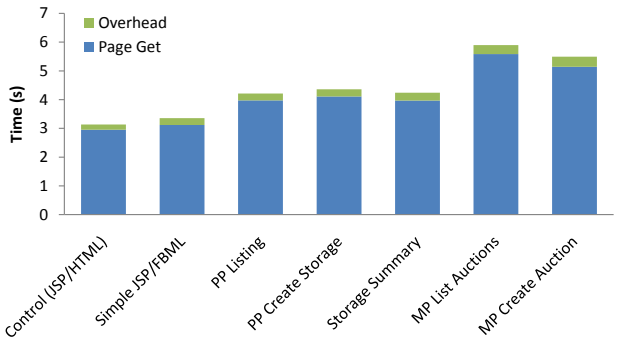

Fig. 12. Page load time for various Social Cloud pages showing both the main page Get and also the overhead of additional requests.

shows that the time taken to perform operations is greatly decreased, for example listing the posted price offerings takes approximately 1 second using Ajax whereas previously it took 4 seconds to load a new page without Ajax.

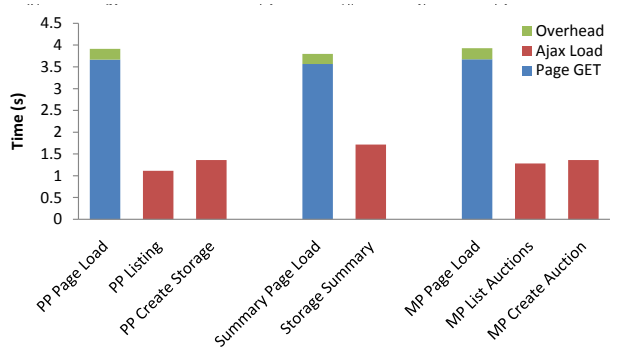

Fig. 13. Page load time for pages in the Social Cloud Facebook Application showing the initial page load and the time taken to update the page using Ajax.

VI. Conclusions & Future Work

This paper has presented the architecture and implementation of a Social Cloud; an amalgamation of Cloud Computing, Volunteer Computing and Social networking. In our system Facebook users can discover and trade storage services contributed by their friends, taking advantage of pre-existing trust relationships. In order to discourage free loading we have adopted a credit-based trading approach. Users may trade with a specific member of their Social network using a posted price market, or participate in an auction-based market.

We have shown empirically that the marketplaces used for trading and/or reciprocation of services could be hosted using small scale resources, based upon the observation that individual social networks are small in size (averaging 130 individuals). In addition, we have shown that even under load, our system can perform multiple concurrent auctions that would satisfy the requirements for a moderately sized social network. Note that this was even the case without deploying the system upon a dedicated resource infrastructure. Our future work aims to generalize our approach so that we can capture additional marketplaces – e.g. Amazon S3 storage could be included in our open storage market.

References

[1] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the grid. *Scientific Programming Journal: Special Issue: Dynamic Grids and Worldwide Computing*, 13(4):265–276, 2005.

[2] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The eucalyptus open-source cloud-computing system. In *Proceedings of 9th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 09)*, Shanghai, China., 2009.

[3] Nazareno Andrade, Francisco Brasileiro, Miranda Mowbray, and Walfredo Cirne. A reciprocation-based economy for multiple services in a computational grid. In R. Buyya and K. Bubendorfer, editors, *Market Oriented Grid and Utility Computing*, pages 357–370. Wiley Press, 2009.

[4] Amazon. building facebook applications on aws website. http://aws.amazon.com/solutions/global-solution-providers/facebook/.

[5] *Facebook Meets the Virtualized Enterprise*, Washington, DC, USA, 2008. IEEE Computer Society.

[6] Zhenhua Guo, Raminderjeet Singh, and Marlon Pierce. Building the polargrid portal using web 2.0 and opensocial. In *GCE '09: Proceedings of the 5th Grid Computing Environments Workshop*, pages 1–8, New York, NY, USA, 2009. ACM.

[7] OpenSocial and Gadgets Specification Group. Opensocial specification v0.9. http://www.opensocial.org/Technical-Resources/opensocial-spec-v09/OpenSocial-Specification.html, April 2009.

[8] David Recordon and Drummond Reed. Openid 2.0: a platform for user-centric identity management. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 11–16, New York, NY, USA, 2006. ACM.

[9] Dan Werthimer, Jeff Cobb, Matt Lebofsky, David Anderson, and Eric Korpela. Seti@home—massively distributed computing for seti. *Computing in Science and Engineering*, 3(1):78–83, 2001.

[10] M. R. Shirts and V. S. Pande. Screensavers of the world unite! *Science*, 290:1903–1904, 2000.

[11] A. L. Beberg and V. S. Pande. Storage@home: Petascale distributed storage. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–6, 2007.

[12] David P. Anderson. Boinc: A system for public-resource computing and storage. In *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10, Washington, DC, USA, 2004. IEEE Computer Society.

[13] Larry Peterson and Timothy Roscoe. The design principles of planetlab. *SIGOPS Oper. Syst. Rev.*, 40(1):11–16, 2006.

[14] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *the 10th IEEE Symposium on High Performance Distributed Computing (HPDC)*, 2001.

[15] K. Chard, W. Tan, J. Boverhof, R. Madduri, and I. Foster. Wrap Scientific Applications as WSRF Grid Services using gRAVI. In *IEEE 7th International Conference on Web Services (ICWS*, 2009.

[16] D. Neumann, J. Ster, A. Anandasivam, and N. Borissov. SORMA - Building an Open Grid Market for Grid Resource Allocation. In *Lecture Notes in Computer Science: The 4th International Workshop on Grid Economics and Business Models (GECON 2007)*, pages 194–200, Rennes, France, 2007.

[17] K. Chard and K. Bubendorfer. Using secure auctions to build a distributed meta-scheduler for the grid. In R. Buyya and K. Bubendorfer, editors, *Market Oriented Grid and Utility Computing*, pages 569–588. Wiley Press, New York, USA, 2009.

[18] N. Borissov, S. Caton, O. Rana, and A. Levine. Message Protocols for Provisioning and Usage of Computing Services. In *6th International Workshop on Grid Economics and Business Models*, pages 160–170, 2009.

[19] Anjomshoaa et al. Job Submission Description Language (JSDL) Specification, Version 1.0. 2005.