Journal of Systems and Software

A Cybernetics Social Cloud

Victor Chang

School of Computing, Creative Technologies and Engineering, Leeds Beckett University, Leeds LS6 3QR, UK.
V.I.Chang@leedsbeckett.ac.uk

Abstract

This paper proposes a Social Cloud, which presents the system design, development and analysis. The technology is based on the BOINC open source software, our hybrid Cloud, Facebook Graph API and our development in a new Facebook API, SocialMedia. The creation of SocialMedia API with its four functions can ensure a smooth delivery of Big Data processing in the Social Cloud, with four selected examples provided. The proposed solution is focused on processing the contacts who click like or comment on the author's posts. Outputs result in visualization with their core syntax being demonstrated. Four functions in the SocialMedia API have evaluation test and each client-server API processing can be completed efficiently and effectively within 1.36 seconds. We demonstrate large scale simulations involved with 50,000 simulations and all the execution time can be completed within 70,000 seconds. Cybernetics functions are created to ensure that 100% job completion rate for Big Data processing. Results support our case for Big Data processing on Social Cloud with no costs involved. All the steps involved have closely followed system design, implementation, experiments and validation for Cybernetics to ensure a high quality of outputs and services at all times. This offers a unique contribution for Cybernetics to meet Big Data research challenges.

Key Words

SocialMedia API; data visualization; large scale simulations for APIs; Big Data Cybernetics; Cybernetics for Social Cloud.

1. Introduction

Social networks have been pervasive in our everyday part of many peoples' lives. There are social network sites such as Facebook, Twitter and LinkedIn who have huge user communities, and users are actively engaged with their social activities online. The social behaviors have been changed as a result of social networks due to the following reasons (Gross and Acquisti, 2005; Farkas, 2007; Glanz, Rimer and Viswanath, 2008). First, more online communications are available and interactive on social network sites. Features include live update, chats and videos allow contacts in the social network to communicate with each other directly or indirectly. Second, a significantly high volume of information can be shared, exchanged and read on daily basis. All the contacts in the network can know about the up-to-date news in a speedy fashion, which supports the Web 2.0 to allow individuals to broadcast about themselves and news centered around them. Third, an increasing number of people have used social network site in search of the information they pursuit, and find out what have happened in the news headline broadcasted by their contacts. For example, the news that the wedding of Prince and Princess of Cambridge and the birth of their son have created millions of twitter tweets and Facebook messages (British Council, 2013). In another example, when the wedding pictures from one of the author's friends were available on the social network sites, the bride received more than 200 congratulations from friends around the world within the first twenty four hours. In comparisons to pre-social network era of early 2000s, this could take months for the brides to receive the same volume of congratulations and best wishes due to the barriers of communications caused by long distance and mobility of people.

Social networks allow people to broadcast their headlines, share any information and interact with friends easily who can be geographically away (Chard et al., 2010). There are no or nearly low costs involved. The speed of interactions is almost instantaneous, and allows users to see pictures or watch videos of places that they have never been, or experienced the detailed scenes in important events such as wedding. Contacts in the network need not take part in those events, but they can find out details by being part of the network contacts and visiting photograph albums and video clicks. In contrast, there are downsides of this information sharing model. First, not every contact in the network is interested in anything posted to his or her account. When a particular event happened to the individual contacts that had an unpleasant experience, messages of sadness and disappointment can be frequently updated on the website. In another example, individual contacts may share multiple links to other news, which appear to be uninterested in the majority of their contacts. Second, some controversial topics such as inequalities in sex and religions, as well as social topics such as same-sex marriage and benefit reform can spark debates on the social networks. While negative comments are unavoidable due to conflicts of opinions, friendship can be damaged to a certain extent of debates becomes viral.

While social networks are influential to our everyday's lives, they generate billion of data including chats, posts, photographs, videos, clicks (such as likes), messages and forums. There are three groups demonstrating their innovative approaches for the social network data organization and management. Chard et al. (2010, 2012) demonstrate their Social Cloud by using Facebook APIs and their proposed architecture to effectively manage thousands of social network data. Facebook introduces their APIs for developers to organize thousands and millions of user data efficiently (Facebook, 2013). Suh et al (2010) demonstrate how to manage millions of twitter tweets in the use of Twitter Network. Consequently the amount of data they received fall into the category of Big Data Science, whereby examples demonstrated by Chard et al (2010, 2012), Suh et al (2010) can help scientists to manage Big Data for social networks (BDSN) and support the concept that social networks are part of the Big Data science. BDSN is an important topic as follows. First, BDSN can provide better recommendation to manage so much data generated on daily basis. It allows the researchers, developers and system managers classify the type of data and to design the right types of algorithms for different purposes. For example, if the focus of a research study is to investigate the relationship between different contacts, the system can query all the number of exchanged messages and replies in the selected contacts, rank them in the order. In another example, if the focus of another research study is to investigate the daily activities on social networks, archive all these information and present them in analytics form, the emphasis is on information gathering, retrieval and visualization. This requires multi-disciplinary approach to understand the complexity, implication and interpretations of Big Data science.

Software Cybernetics (SC) explores the interplay between software engineering theories and practices. Cai (2002) and Cai et al (2003) demonstrate the control theory and software engineering. They also define the SC concepts and definitions of SC. However, their definition is only on software engineering and control engineering. In the era of Cloud Computing and Big Data, newer definitions, scopes and demonstration should be provided. In this paper, we demonstrate that SC is an emerging area for processing large amounts of information and data in the Cloud and it involves integration of different technologies. For example, there are many people on social networks generating and disseminating a large amount of information. The relationships, discussion threads and extents of trust, collaboration and support between individuals on each person's social network account is different and varied from time to time. This requires intelligent systems such as BDSN that can process a vast amount of data and interpret the complex human relationships and the topics that people like and support. Fast and innovative methods are thus welcome. However, they can be expensive and difficult to use, which motivate us to develop an easy-to-use and cost-effective Social Cloud system.

This paper describes the Social Cloud, a platform based on the adapted BOINC open source project and our development work that use Cloud Computing and Big Data processing. We demonstrate how to use Software Cybernetics to govern the construction and running of the Social Cloud. The structure is as follows. Section 2 presents the BOINC project, its approaches and architecture. Section 3 demonstrates the development of a SocialMedia API, which ensure a smooth delivery of Big Data processing in the Social Cloud, with four examples provided to explain how to analyze and present Big Data analytics for social networks. Interpretations of outputs in visualization and their core syntax will be explained. Section 4 describes Social Cloud experiments involved with four API functions, including the single simulation and large scale simulations on three different types of Clouds. Results support good performance of our proposed solution for Big Data processing. Cybernetics with software testing steps and outputs will be presented at the end of Section 3 and Section 4. Section 5 presents four interesting topics of discussion and Section 6 sums up Conclusion and future work.

2. The Social Cloud based on BOINC project

This section is aimed at describing the Social Cloud based on BOINC (Berkely Open Infrastructure for Network Computing) project, including approaches, architecture and its relevance to Software Cybernetics. A Social Cloud is defined as a scalable computing platform which can be dynamically shared amongst a group of contacts (friends) in a social network, and resources can be heterogeneously by contacts. A Social Cloud can be benefits from trusts between contacts and the strengthening in friendships as a result of communications and sharing (Farkas, 2007). In contrast to Social Cloud, Virtual Organizations (VOs) have proposed a similar approach, since VOs have policies to define the type, membership and sharing permissions for the groups involved (Foster, Kesselman and Tuecke, 2001). However, the Social Cloud is different from VOs in the level of trusts and mechanism for social correction (identifying advantages and disadvantages for contacts to participate) between groups (Chard et al., 2010, 2012). Similarly, users can be members of multiple Social Clouds, and are not restricted to one group like VOs often do.

2.1 The BOINC project: Introduction and Motivation

According to BOINC (2013), there are at least 2.2 million BOINC participants, which are substantially available for undertaking the Social Cloud experiment. The BOINC project was first started as a generic volunteer computing middleware. It had over 50 supported projects, including a few internationally active ones (Anderson and Fedak, 2006, BOINC, 2013). The BOINC project had huge processing power of 8 petaflops, which included the super-computer of Tainhe-I of China (Costa, Silva and Dahlin, 2010). There are other active projects in collaboration with BOINC. GridRepublic is an account management system which can make multiple project management for volunteers much easier. BOINC has created a Facebook application called Progress Thru Processors (PTP) with Intel, and both organizations will demonstrate their prototypes in due course.

Social Cloud computing offers a novel approach for leveraging social network and distributed computing, and the successful adoption can motivate and facilitate volunteer based sharing. Motivation for using BOINC project for the Social Cloud can be available for two different groups, users and researchers, as follows. First, users need to find appropriate projects, decide which projects suit them the most, set up and maintain required software in the current model. This can be a barrier for some users. The Social Cloud approach can remove this barrier and allows anyone in the contact to join. Second, Social Cloud researchers can connect to the people that can be helpful or supportive to their research. This can reduce the amount of time for them to find partners.

2.2 Related background

This section describes the related work before introducing the approach and architecture adopted by BOINC project (2013). Users must download the BOINC client software, register themselves and install on the system before they can contribute. Users can choose to support more than one project by allocating resource shares for each project. Users can decide the extent of resource sharing and project selection before they start at their free wills. The BOINC client downloads work units periodically from their selected projects, and processes these units, and sends results back to the project servers and obtains credits for work completed. If the work unit returned is validated, the user receives credit, which is a win-win situation for both users and projects. Users will not receive credits if result is returned after deadline or result is inaccurate. The credit system is to discourage cheating and encourage users to donate more resources by having a sense of competitions around credits earned.

Users can manage multiple projects by using an account management system (AMS, such as GridRepublic), which allow users to setup a “meta-account” to manage all projects. Users can direct the BOINC client to connect to the AMS with their credentials, the account manager works as a proxy between users and projects. To help the process to go on smoothly, BOINC has published a set of WebRPCs to specify how account management systems and project servers should communicate. AMS has a limitation which does not have any social features to bring together with new and existing features. To offset this issue, current work-around is to introduce Progress Thru Processors (PTP) application to streamline with GridRepublic account from within the Facebook platform. However, additional work is still required to ensure a smooth delivery. This is where our research contributions for this area.

2.3 The Architecture

The Social Cloud is a hybrid cloud based on the integration of our private clouds in Southampton and London, community clouds adopted by BONIC projects and Facebook. The Social Cloud can be regarded as the AMS from our Cloud resources, volunteer PCs running BOINC clients and running of Facebook application. An important objective is to allow users to add and remove projects within Facebook application, which can set resource shares for projects of their choice. This information is used to communicate with project servers and control all BOINC clients. Each associated element in the architecture is presented in Figure 1, with the sequence of events and their explanations as follows.

1) Integrating into then regular Facebook experience.
2) The Social Cloud rendering as a Facebook application within the Facebook interface.
3) Using Facebook APIs to augment the Facebook experience for users and their contacts with Social Cloud APIs.
4) BOINC WebRPCs communicate with BOINC project servers to create account, query user credits and so forth.
5) BOINC Account Manager RPCs processes communication with the BOINC clients.
6) Communications between the BOINC servers and clients (not a function of this proposed Social Cloud)
7) The entire infrastructure and experiments to validate its performance, scalability and reliability will be presented in Section 4.

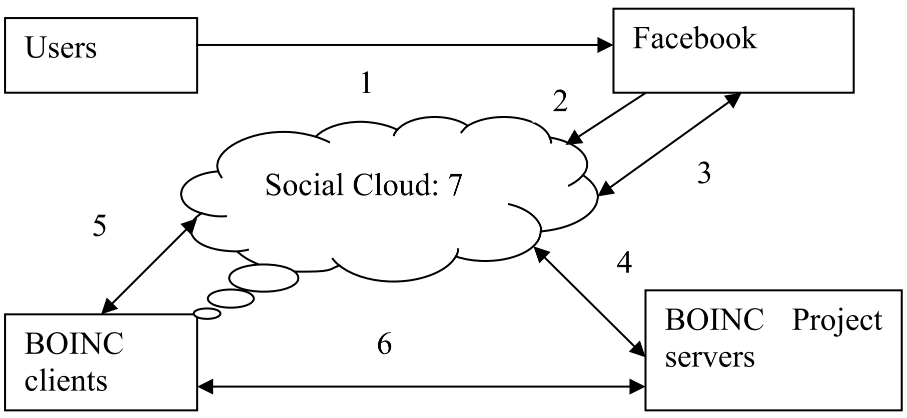

Figure 1: The architecture of deploying the Social Cloud

2.3.1 Additional work required for Facebook

This section describes the additional work required for Facebook, which is used as a platform to demonstrate the concept of the Social Cloud. Externally-hosted applications can run within the Facebook User Interface, and the Facebook Graph Application Program Interface (API) can retrieve and present the social information (Facebook, 2013). Both users and applications should be authenticated by using the OAuth protocol on the Facebook to access the Graph API (Facebook, 2013). This allows the Graph API to present an underlying social graph that contains users and their connections with other nodes in the graph, which mean the accessibility to the people, photographs, events, videos and pages can be presented in visual and graphical forms. The combined approach of using the Graph API and the vast number of users on the Facebook can help demonstrate the concept of the Social Cloud appropriately.

The design of API illustrated in this paper has followed the Software Cybernetics approach. This is an important step towards Big Data software engineering. The system design includes the following entities: User, Facebook, Social Cloud, Project Server and BOINC Client. All these entities have straight forward interpretations. For example, user is the person using this service. Facebook represents the Facebook API. Social Cloud is the platform provided by this project. Project server include both BOINC server and additional servers provided by our project. BOINC client is the client available from the BOINC project. Additional explanations for BOINC entities are presented as follows.

2.3.2 BOINC Project servers

The Social Cloud can handle the BOINC's published Web Remote Procedure Calls (WebRPCs) in order to support the BOINC project. The WebRPC model has the following assumption. First, every RPC has an HTTP GET transaction. Second, the input parameters are represented as a set of parameterized GET arguments. As a result, the output is an XML document which is parsed by the Social Cloud. The aim is to let users monitor their contributions and feed their defined social engineering algorithms, which will be presented in Section 3. Since the Social Cloud works as an AMS, its Social Cloud account manager can support the use of WebRPCs.

2.3.3 BOINC Clients

There are two ways for BOINC clients attach to an AMS. First, data on the account manager can bundle with the installer. Second, users can specify the AMS URL, which is the URL for the Social Cloud. In an either way, users should authenticate on their clients to obtain resource share preferences from the Social Cloud. Similarly, the BOINC clients use AMS RPC to communicate with the Social Cloud AMS. Once the clients have processed the data, it attaches itself to each of project servers directly and then pulls information from the server for processing. Results can be presented on the clients.

2.4 System Design

To demonmstrate cybernetics for testing, UML is a suitable method for system design. Figure 2 shows a UML diagram to help explain the relationship between each entity of the architecture, how the interactions between each entity takes place and the sequence of events happened in the architecture. In this example, a user can add Facebook application to the Social Cloud, permission for the required user data are requested through Facebook. Once this step is completed, the user can generate their interest signature, which is compared against project signatures for the user can be authenticated. He can be given suggested projects that he would like to join.

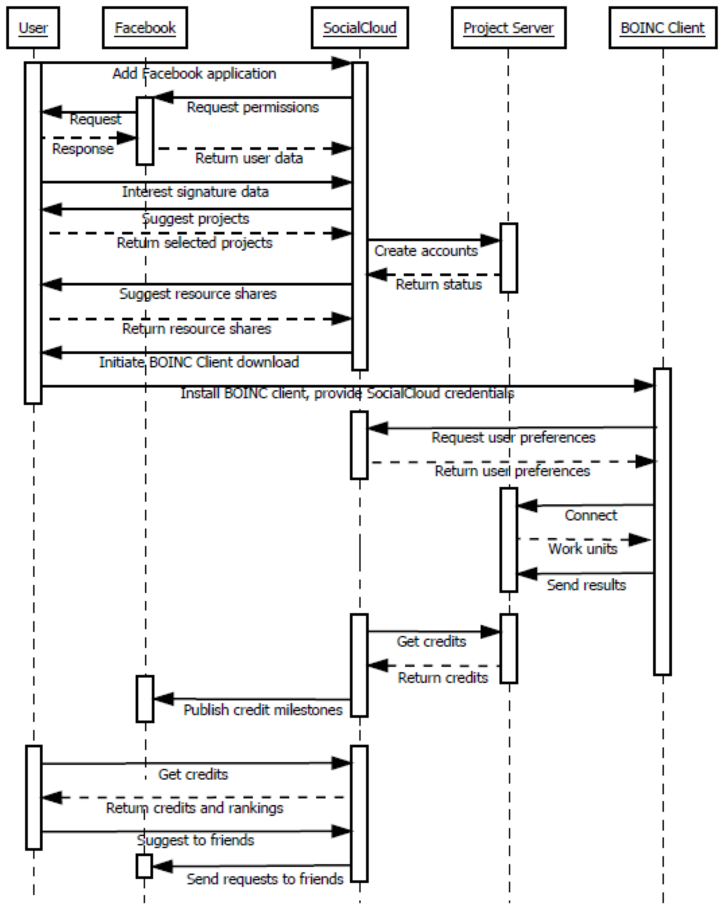

Figure 2: The UML diagram to explain the interactions in the architecture

The project server responds and grants for credential. The user can install a BOINC client, and then get an authorized respond to have access to all resources on the server. The user then gets to the server again to process the request he has sent, and then gets the results. The user's credits before approval are sent back to Facebook. The Social Cloud queries and checks the status, and the user gets the credits from the Social Cloud when job completion is confirmed. Results are then published on the Facebook. The user's status is archived in the ranking sites. As shown in Figure 2, the relationship between all entities (user, facebook, Social Cloud, project servers and clients) is presented. The request and response model between different entities is illustrated. Although system design by the UML diagram is common, it does not show the software cybernetics in system design and development. In order to demonstrate an improved prototype, a new cybernetics for Big Data is proposed for the Social Cloud starting from software design.

2.5 Cybernetics System Design for the Social Cloud

This section describes the system design with cybernetics approach. The use of Facebook API development can make software design and development relatively more convenient for developers, but it does not use any cybernetic approach to ensure that the system design and development being fit for purpose. Adoption of cybernetics can help process a large number and size of data. When data processing is not going on well by the traditional approach, problems such as downtime, delay and request failures can be minimized. This motivates us to develop SocialMedia API, a Facebook API with cybernetics approach. There are four functions in SocialMedia API that has been developed:

- FriendNetwork: It queries a list of friends, friend IDs and user data of the person interacting with the Facebook API.
- LikeNetwork: It queries all the data related to "FriendNetwork" and also post data. It then reads and post data of their friends and queries the number of likes that their friends have made altogether.
- LikeCommentNetwork: It queries all the data related to "FriendNetwork" and also post data. It then reads and post data of their friends and queries the number of comments and likes their friends have made altogether.
- Posts: It queries all the data related to "FriendNetwork" and also post data. The emphasis is to read all the text strings which display information such as IDs, types of posts and their date and time of posting.

The first three functions use Facebook Graph API to present data in the form of visualization. This can ensure that all results can be understood by the users more easily. While clicking each component in the visualization, the outputs can show the related text information. For example, if the "FriendNetwork" is used to processe data, the outputs will display a list of circles (individual friends) and how they are linked to each other. Each link represents the strength of the friendship. If a "dot" is selected and hovered, a small window on the top of the particular dot can display who this person is and his ID.

2.5.1 Cybernetic system design for FriendNetwork function

This section shows the system design for FriendNetwork fuction. Figure 3 shows the system design for FriendNetwork API which includes four stages. The user has to login the Facebook. The first stage is to query a list of friends and their IDs from the user who has approved that the SocialMedia API can query all these information. When the approval is completed, the FriendNetwork function proceeds with data processing to collect and analyze all the data related to thew user's friends. While all these information has been collected, they are all stored in the form of text. Text will need an available Facebook Graph API, which can transform all the text into visualization. During this stage, however, is not always successful depending on the number of friends and the quantity of the information the friends have. This also partly explains why Facebook has limited to 5,000 friends per account or their APIs will be unable to handle information. In most of circumstances (99% tested in our preliminary experimental conditions), all the text-based information can be transformed into the graphical presentation. When this step is successful, outputs can be in the form of visualization. In other words, the FriendNetwork function displays dots (individual friends) linking to different dots to represent the strength of human relationship based in the Social Cloud. The job is completed and results will be discussed in Section 3.2. If some steps are unsuccessful, they will be returned to Social Cloud reprocessing of SocialMedia API and start another new request for data processing and visualization. Cybernetic approach in system design ensures that Big Data processing can be handled in a more efficient way.

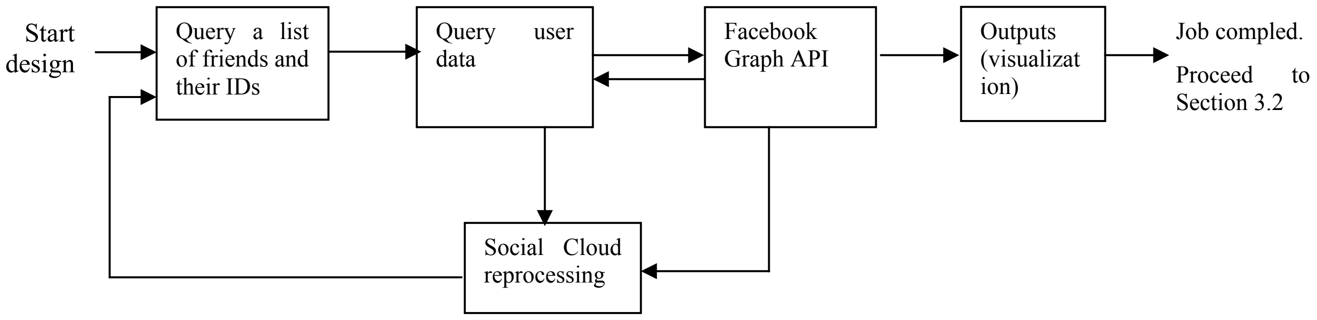

Figure 3: The system design for FriendNetwork function of SocialMedia API

2.5.2 Cybernetic system design for LikeNetwork function

The LikeNetwork function requires all the information collected by the FriendNetwork function plus collecting additional information that includes the number of likes and who click likes in all the phographs and status updates posted. shows the system design for LikeNetwork function. It starts with the same process of FriendNetwork function. If anything goes worng, it returns Social Cloud reprocessing of SocialMedia API to start all over again. After the completion of FriendNetwork processing, LikeNetwork function queries the number of likes and whom have cliked likes in all the photographs and status updates amongst the user's list of friends. It then sends all results to Facebook Graph API for transforming results into visualization. Similar to Section 2.5.1, if transformation is successful, outputs will be presented in visualization which will display a full relationship between dots and links and the text information about each dot and link. If unsuccessful, the request will be returned to Social Cloud reprocessing of SocialMedia API to start the job again.

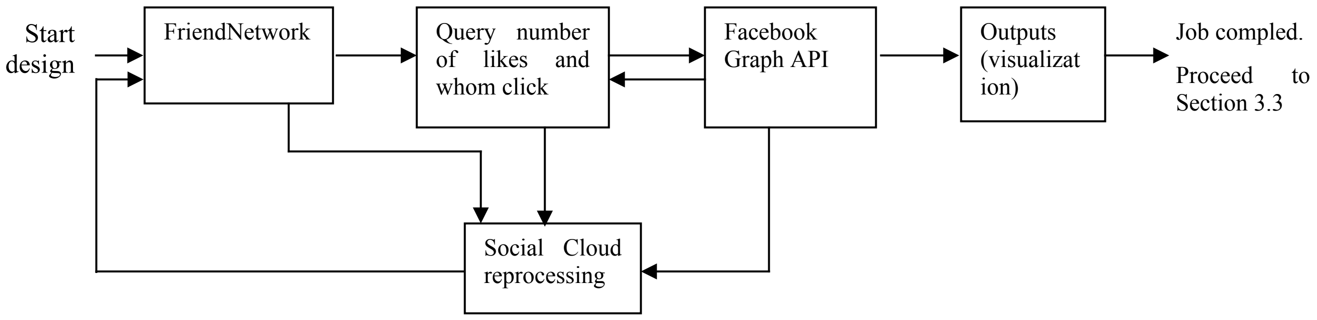

Figure 4: The system design for LikeNetwork function of SocialMedia API

2.5.3 Cybernetic system design for LikeCommentNetwork function

The LikeCommentNetwork function requires all the information collected by the FriendNetwork API plus number of likes, whom have clicked likes and whom have comments. It is the same as LikeCommentNetwork function except adding another query for comments. Figure 5 shows the system design for LikeCommentNetwork function with identical first two steps as in LikeNetwork function. Upon querying information for likes, the third step is to query number of comments and whom have entered their comments to the photographs and status update posted by the user. When all the information have been collected, Facebook Graph API will transform the text into visualization. Upon successful job completion, results will show dots, links and all the relationships between dots and links with textual explanations. If Facebook Graph does not generate successful outputs, then it sends back to Social Cloud reprocessing of SocialMedia API to start the job request again.

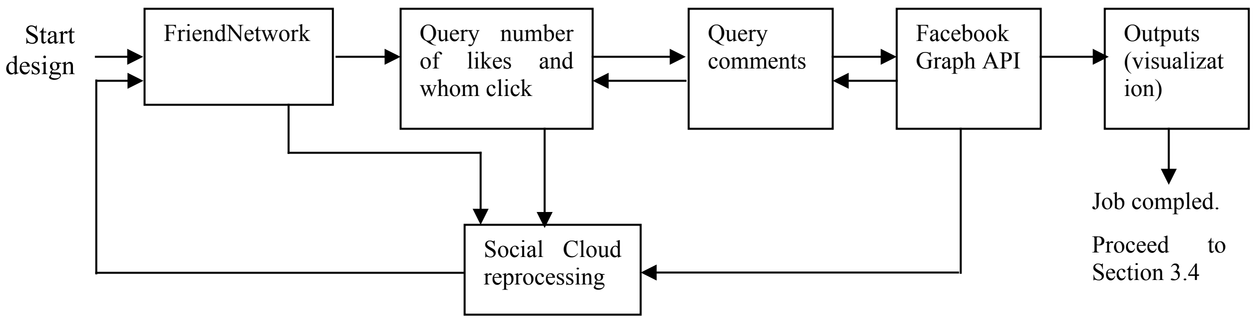

Figure 5: The system design for LikeCommentNetwork function of SocialMedia API

2.5.4 Cybernetic system design for Post function

The first three functions of the SocialMedia API are focused on querying the required information on the user's list of friends and their activities such as involved in likes and comments. Outputs are presented in the form of visualization upon successful job completion. The forth function, Post, is focused on querying on the number of people who have posted on the user's account and checks on whether the user has posted on his/her own blogs. This is important since the first three functions are focused on querying on others and Post function needs to query both on what others did and what the user himself/herself did. The query information will be focused on the names, links, dates, types of posts (for photos, comments and so on) and system information. The first step is identical to FriendNetwork function to query all the friends' data. The second step is the same as LikeNetwork to query number of likes and whom have clicked like. The third step is the same as LikeCommentNetwork which queries number of comments and whom have commented. Unsuccessful outcomes can return to social cloud reprocessing of SocialMedia API to start the job again. The only difference to the previous three functions is that no Facebook Graph API is involved. All the outputs upon successful job completion include strings of text and has an additional feature to summarize the shorter version of results. See Figure 6.

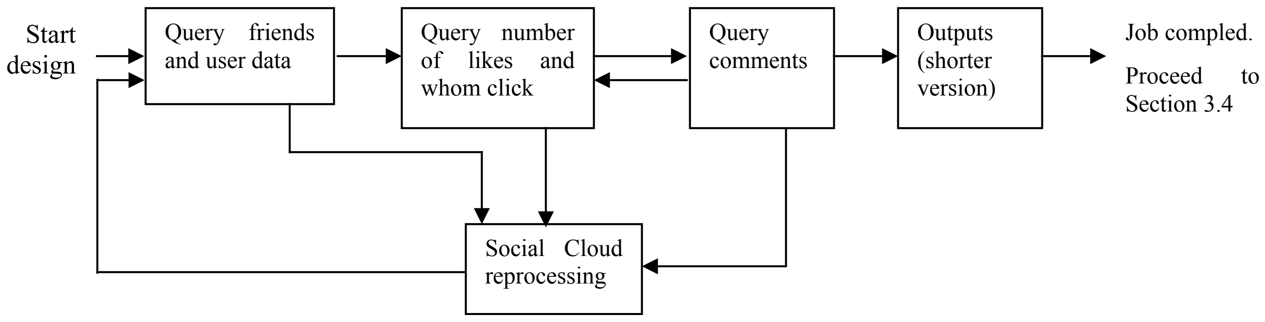
Start design
Query friends and user data
Query number of likes and whom click
Query comments
Outputs (shorter version)
Job compled.
Proceed to Section 3.4
Social Cloud reprocessing

Figure 6: The system design for Posts function of SocialMedia API

3. Our contribution to the Social Cloud: Development of SocialMedia API

Section 2 explains the background information in regard to the use of BOINC project for our Social Cloud project, and states that additional work is required for improving the existing Facebook API. Mislove et al (2009) provide an overview of social network websites and explain their measurement methodology. They assert that the web crawling technology to process a large number of data and present them in a form that people can understand, such as visualization, is a research challenge. They describe their method, experiments, results and their analyses. However, their work was developed five years ago. While acknowledging their work that visualization and analytics are challenging, we can develop based on the Facebook API to allow more software developers to offer services which are easy to use and redevelop. This motivates us to develop a SocialMedia API whereby the four major functions have been introduced with their cybernetics approach. SocialMedia API can read all the required information about the user from the Facebook, process all the information on the BOIC servers and send results back to the Facebook. This section aims to describe the SocialMedia API, including four inclusive functions and usage scenarios.

3.1 The usage of SocialMedia API

This section describes some of the functions offered by SocialMedia API.

- **SocialMedia["name"]** which gives information about the social media entity. Most of the cases, and “name” is usually Facebook by default.
- **SocialMedia["name", "property"]** which gives the value of the specified property for the social media entity . This is a commonly used command to retrieve the information for the user and process on the BOINC servers.

There are additional explanations for Facebook-related properties and they can be used for “property” in the command including:

- "Friends": list of friends
- "FriendIDs": list of friend IDs
- "UserData": user data
- "Posts": post data
- "Feeds": feed data

Advanced features are available for user-related networks, with vertices corresponding to users including:

- "FriendNetwork": x is connected to user y if x and y are friends
- "LikeCommentNetwork": x is connected to y if x and y like or comment on the same post
- "LikeNetwork": x is connected to y if x and y like the same post
- "CommentNetwork": x is connected to y if x and y comment on the same post

All these features can be incorporated into SocialMedia API to instruct the BOINC servers to retrieve the required information for the user, processes the information and presents results on the Facebook. Four examples of using SocialMedia API will be illustrated as follows.

3.2 The first example of using SocialMedia API

This section describes results generated by both the BOINC project and the SocialMedia API developed by us. The author undertook the evaluation process by demonstrating the results of social network analysis, followed by their explanations. The author is the user himself to query all the data based on his friends, friends’ IDs and related information for FriendNetwork function of SocialMedia API. The author has four groups of contacts based in Taiwan, Singapore, Australia and United Kingdom. This section describes the network analysis retrieved from the author’s contacts with their explanations. The code syntax will be explained in Section 4. The author followed the steps described in Section 2 to use BOINC project for the use of the Social Cloud and use the SocialMedia API to demonstrate the results. He typed in

SocialMedia [“Facebook”, “FriendNetwork”]

Results and their discussions are presented as follows.

3.2.1 Family and church

The author has the family members based in Taiwan and Singapore presented in Figure 3. The left half of Figure 3 shows 70% of his family members in Taiwan, and 30% of the rest in Singapore. The right half of Figure 7 shows his church members base din Singapore. The area of the circle shows how frequent the user (the author's contact) has used the Facebook application. The bigger the circle, the more frequent the user has used the Facebook. The links represent the number of contacts between different users in the public domains. The more links between the circles, the more frequent contacts between the users. However, this does not include the private messages exchanged between different users due to the restrictions of the Facebook privacy policy. There are also other contacts between the author and the family members outside the Facebook domain, and is not recorded here.

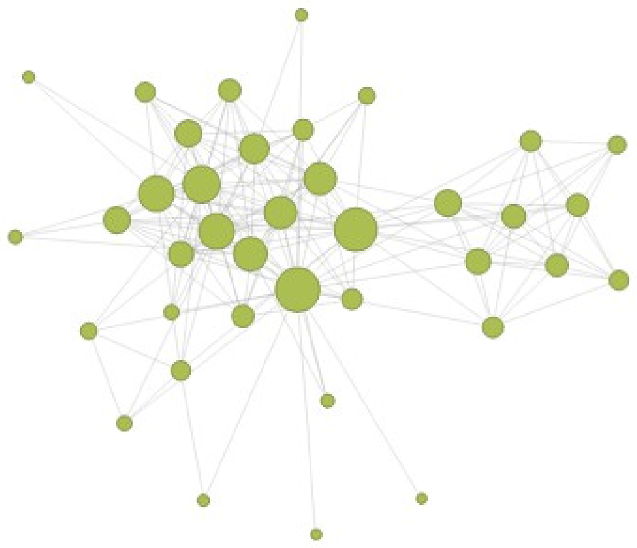

Figure 7: Representations of the Social Cloud based on the author's family and friends

3.2.2 Former classmates in Singapore

The majority of the author's former classmates are based in Singapore. Figure 8 shows the results, where the majority of the author's former classmates uses Facebook frequently. Despite of the size of Singapore and the ease of communications, most of them have interacted actively between one another on Facebook. A possible reason might be due to the availability of Facebook services on the mobile devices and their hectic work life, online interactions have become the main factor for communications between the former classmates.

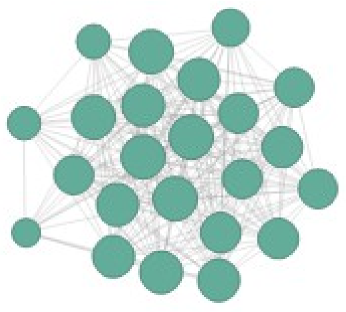

Figure 8: Representations of the Social Cloud based on the author's former classmates

3.2.3 Friends and church members in Australia

The third group of the author's contacts includes his friends and church members based in Australia. This group of users is the most frequent users of the Facebook as seen in Figure 9. Several circles are larger than the circles in other three groups. The number of online interactions has been so frequent that circles overlap with each other. The reasons are that first, friends and church members in Australia have created several online communities, and they have interacted actively on Facebook. Second, they have e-church services and activities, which appear successfully to get many of them together. When they posted missionary calls, it attracted many responses. When they posted the photographs and videos taken in the church camp, it attracted many members to visit several times, to circulate amongst their friends, to click 'likes' for support and to leave messages on their albums. The smaller dots mean they are the author's friends, but are non-Christians. Blue dots means they are the author's friends based in Singapore and UK, and only one of his friends in Australia know them.

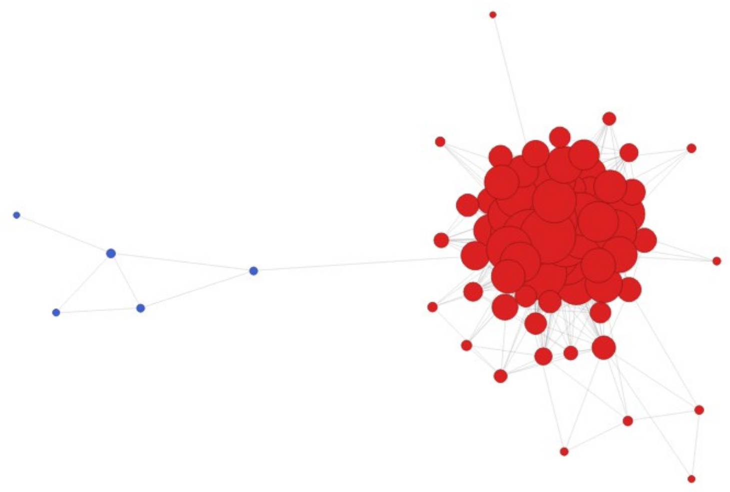

Figure 9: Representations of the Social Cloud based on the author's friends and church members in Australia

3.2.4 Colleagues and friends based in the United Kingdom

The last group of the author's contact includes his colleagues and friends based in the UK. The size of the circles and the intensity of the links vary. The most likely reason is that these contacts were met mainly in Cambridge, Southampton and London, and at different stages of the author's career development. The majority of the contacts in the left half of Figure 10 are contacts met in Southampton. Most of them have closer interactions on Facebook.

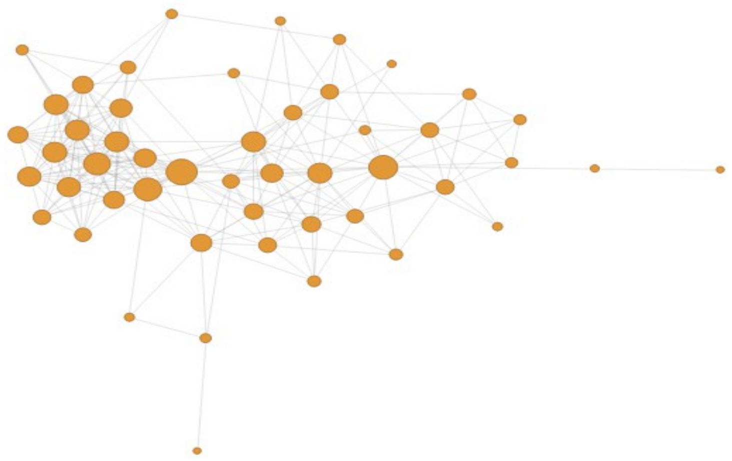

Figure 10: Representations of the Social Cloud based on the author's colleagues and friends in the UK

Some distant and smaller circles are contacts based in Cambridge, and the author met them between 6 and 12 years ago (and hence, there are fewer interactions between them). The right half of Figure 6 has the author's contacts met in Southampton and London, and some of them have interactions with each other. Additionally, some of them already left Southampton and London, and are categorized in this group because they are the contacts met in the UK. The FriendNetwork function of SocialMedia API can query all the four groups of friends and present them according to the locations, with all results represented by visualization as a result of cybernetics system design.

3.2.5 The core syntax that presents "FriendNetwork" in the SocialMedia API

This section shows the core syntax that presents the author's contact in the Social Cloud. Referring to Table 1, the syntax "author.contact" means all the friends in the author's Facebook account. The syntax "contact.groups" has four groups, which were presented in earlier section in that particular order. The syntax "contact.active" means these friends have public interactions with the author, which mean they have clicked "likes", or commented on the author's post. The syntax "contact.interact" means that all the friends who posted or commented on the author's post, they also interacted between themselves. In other words, they are the author's mutual friends. The query shows that both conditions of "contact.active" and "contact.interact" must be met to satisfy the conditions for SocialMedia API. The order statement means that all results should be presented in this order from Section 3.2.1 to 3.2.4. The use of Facebook Graphics API can then present all the results in the visualized form as presented in the earlier section.

Table 1: The core syntax for SocialMedia ["Facebook", "FriendNetwork"]

```
Select author.contact from contact.groups
where (contact.active) and (contact.interact)
order by contact.groups
```

3.3 The second example of using SocialMedia API

This section describes the second example to extract information from the Facebook by the use of the SocialMedia API developed by the author and the use of a BOINC project. The second example is to collect the information about the frequency and the people who click "likes" on Facebook, and present results in the visualized format. The command is as follows:

SocialMedia["Facebook", "LikeNetwork"]

3.3.1 The result of using "LikeNetwork"

Figure 11 shows the result of the author's contacts who click likes. This command does not categorize all contacts with different regions but collectively analyze all the contacts who have clicked likes. Each circle is the representation of the contacts who have clicked likes. The size of the circle does not matter in this case. The intensity of the links between the circles indicates the number of times that the contact click likes. Hence, if the author's contacts know more people amongst his contact, it can appear to have a high level of intensity for links. Different groups of contact get together and can link to other posts, photographs, videos and links. If the author's contacts are not mutual friends, they are not counted in this SocialMeida API.

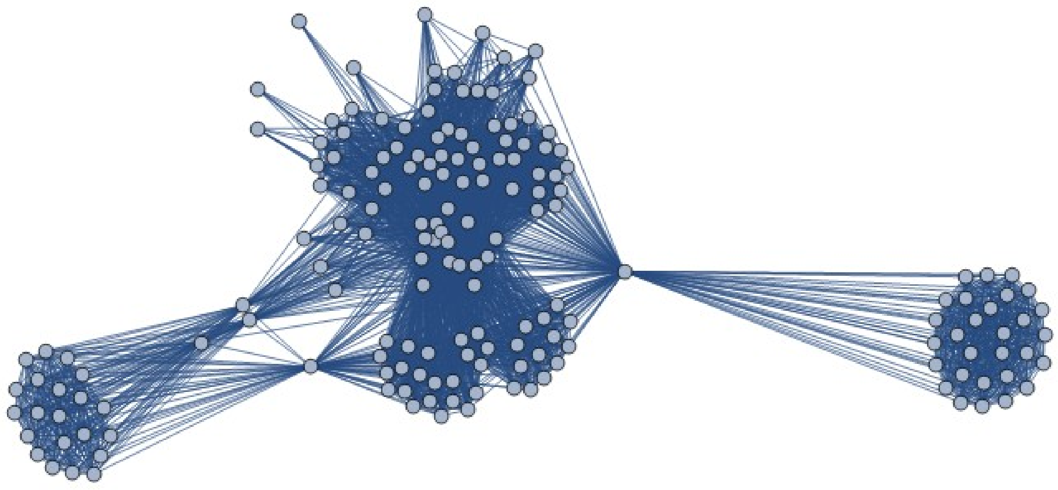

Figure 11: Representations of the Social Cloud based on the author's contacts who click likes.

3.3.2 The core syntax that presents "LikeNetwork" in the SocialMedia API

Table 2 shows the core syntax behind the "LikeNetwork". The queries select all the author's contacts, with four conditions to be satisfied. First, "contact.active" must be present. Second, all the contacts should be mutual friends. Third, the syntax "contact.likes" refers to the contact that clicked the authors' posts. Fourth, "count" refers to the number of times the contact have clicked likes, which is presented by the number of links between different circles. It only queries the head counts and the frequencies of clicking likes, which means anyone who clicked likes at anytime.

Table 2: The core syntax for SocialMedia ["Facebook", "LikeNetwork"]

```
Select author.contact from contact.groups
where (contact.active) and (contact.interact) and (contact.likes) and (count)
```

3.4 The third example of using SocialMedia API

This section describes the third example to extract information from the Facebook by the use of the SocialMedia API developed by the author. The third example is to collect the information about the frequency and the people who click "like" and also have commented on Facebook to present results in the visualized format. The command is as follows:

SocialMedia["Facebook", "LikeCommentNetwork"]

3.4.1 The result of using "LikeCommentNetwork"

This command is the same as in Section 3.3 except adding another condition, any contacts who have commented on the author's post. The same explanations in Section 3.3.1 applies. The only difference between Figure 11 and Figure 12 is that Figure 12 include some circles at a distance from the centre, they are the contacts who have commented. Some contacts have commented, but not necessarily clicked like, and vice versa. In the author's contacts, the majority have clicked like without leaving comments, who have much more than

- Contacts who commented without clicking like
- Contacts who commented and also clicked like

This explains how and why Figure 12 looks different than Figure 11 although the centre part of the Figure appears to be highly similar. Facebook has collected the information about whom and when click likes, and whom and when comment, and the use of this API simply extracts the information and presents them, and the next section explains the core syntax.

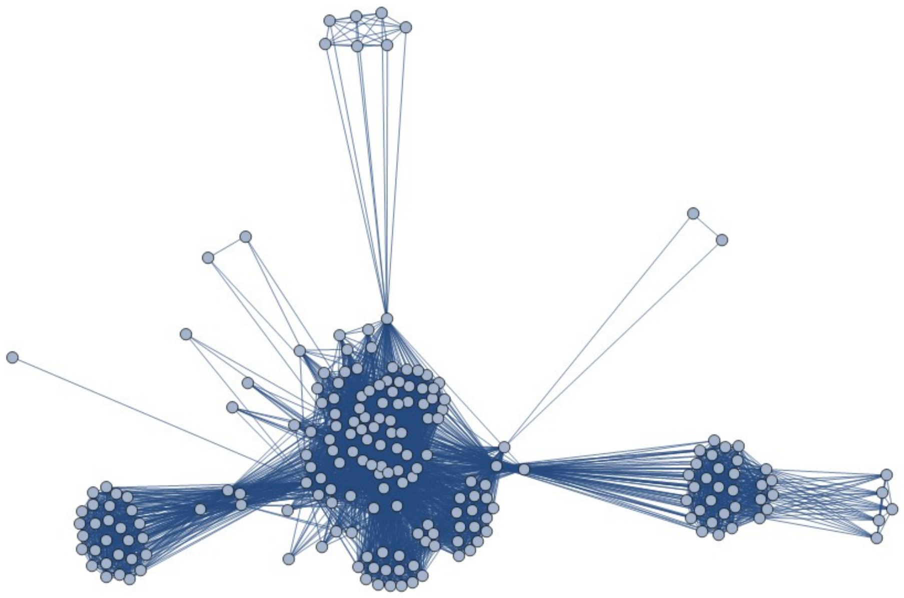

Figure 12: Representations of the Social Cloud based on the author's contacts who click like or comment.

3.4.2 The core syntax that presents "LikeCommentNetwork" in the SocialMedia API

Table 3 shows the core syntax behind the "LikeCommentNetwork". The queries select all the author's contacts, with four conditions to be satisfied. The first three conditions are identical to Section 3.3.2. The only difference is the fourth condition, and the queries can accept any contacts who have clicked like, or any contacts who have commented. Any contacts who fulfills either condition, is accepted by the queries.

Table 3: The core syntax for SocialMedia ["Facebook", "LikeCommentNetwork"]

```
Select author.contact from contact.groups
where (contact.active) and (contact.interact) and (count) and ((contact.likes) or (contact.comment))
```

3.5 The forth example of using SocialMedia API

The first three examples are focused on the representation of results by graphical visualization. The Social Cloud can also provide output as strings of text, as a result of queries from the author's Facebook account information. The fourth example is focused on the text retrieval. The command is **SocialMedia["Facebook", "Posts"] // short**

This command can query any of the author's contacts who posted on the author's Facebook main page by default. The command displays the author's basic information such as his IDs, time of post made, the type of posts and creation time. This is dependent on a few factors:

- Number of people who posted on the author's account
- Whether the author has posted on his account

The reason is that if there are more posts on the Facebook main page, the first post to be queries will be the one at the bottom of the main page, without clicking links to retrieve archived posts from Facebook and display on the author's account. For example, if there are as many posts as 300 and the main page can hold 100 posts (without clicking links to old posts) by default, and query only works for these 100 posts. The syntax "//short" presents the first query in the list. So if amongst 100 posts and the first one was posted in December 2012, the query can retrieve the one posted in December 2012.

3.5.1 The result of using "Posts"

Table 4 shows the results of using "Posts". The results show the all the information retrieval from the author's first 100 posts and display the basic information about the status and type of his posts, the URI, creation time (for the first post) and update time (when the query was made). The syntax "author.ID" refers to the author's account ID on Facebook. Posts that have either like or comments are all queries. The application is a photograph album, which has an ID of 2305272732. All photograph albums (which have either likes or comments, coincidentally the entire author's photograph albums have either/both likes and comments) are queried. The creation time for the first post was in October 26, 2012 (US East Time), and the time for this query was made on December 26, 2014 (US East Time).

Table 4: Results of queries using SocialMedia ["Facebook", "Posts"] // short

```
short[{{Actions->{{Link-
>https://www.facebook.com/author.ID/posts/10152116781813979,Name->Comment},
{Link->https://www.facebook.com/author.ID/posts/10152116781813979,Name-
>Like}},Application->{ID->2305272732,Name->Photos},<<13>>,Type-
>photo,UpdatedTime->2014-12-26T15:58:50+0000},<<459>>,{CreatedTime->2012-10-
26T13:52:47+0000,From->{ID->author.ID,Name->Victor Chang},ID-
>author.ID_10151225731333979,<<3>>,Type->status,UpdatedTime->2012-10-
26T13:52:47+0000}}]
```

3.5.2 The core syntax that presents "Posts" in the SocialMedia API

Table 5 shows the core syntax behind the "Posts", and there are four conditions to be met. The use of the syntax "contact.active" and "((contact.likes) or (contact.comment))" is the same as Section 3.4.2. The difference is that there are two other syntax used. The syntax "contact.post" presents the information about which contacts posted on the author's account. The syntax "author.information" presents all the related information about the author, with regard to the posts.

Table 5: The core syntax for SocialMedia ["Facebook", "Posts"] // short

```
Select author.contact from contact.groups

where (contact.active) and (contact.post) and (author.information) and ((contact.likes) or
(contact.comment))
```

3.6 Validating four functions of SocialMedia API for Software Cybernetics

Software validation is an important process for Cybernetics, whereby four functions of the SocialMedia API require to pass validation. In other words, positive outputs can be produced by following instructions described between Section 2.4 and 3.5. Figure 13 shows the Cybernetics validations involved confirming the SocialMedia API can deliver four functions. How validations have been undertaken and passed for Cybernetics are as follows. First, the descriptions in Section 3.2 show that "FriendNetwork" of SocialMedia API can illustrate the friendship status and strength of connections in the author's networks, which have been mainly located in Taiwan, Singapore, Australia and the UK. Second, descriptions in Section 3.3 confirm that the function of "LikeNetwork" works well. Third, descriptions in Section 3.4 assert that "LikeCommentNetwork" illustrate the visualization of the networks who have commented with or without clicking likes. Last, the descriptions in Section 3.5 explain positive outputs generated by "Posts". If any function does not return positive outputs, then they were returned to Social Cloud validation of SocialMedia API, so that the entire process can be started again. All these examples can confirm that validations had been undertaken and positive outcome were demonstrated prior the client-server experiments. The four functions developed for SocialMedia API have passed the tests and results can be presented as social network visualization.

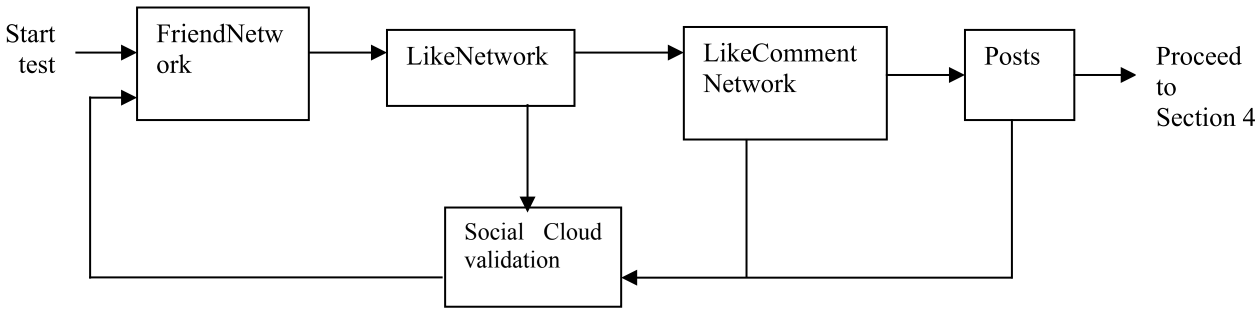

Figure 13: SocialMedia API testing for the software Cybernetics

4. The experiments used for the Social Cloud

This section describes the hardware setup to use the Social Cloud which uses the BOINC architecture and Facebook as described in Section 2 and 3. An objective is to test the performance of the SocialMedia API running on the Facebook and BOINC servers. Various simulations and experiments have been performed using a high specification desktop environment, private and public clouds. The desktop machine has 2.67 GHz Intel Xeon Quad Core and 4 GB of memory (800 MHz). The private cloud is used and it involves four sites in total; two in London and two in Southampton. The University of Southampton's resources are used for all 3D Visualization, and are also used to connect the author's home cluster, Greenwich and University London Computer Centre (ULCC), where hundred of servers have been hosted. There are reliable computational connections between internal networks.

4.1 Hardware set up and experiments

The ULCC has advanced Cloud and parallel computing infrastructure with network attached storage (NAS) service. In total it has CPUs totaling 30 GHz, 60 GB of RAM and 24 TB of disk space in place. Experiments performed in this environment can get the better sides of optical fiber network with 10 Gbps speed. There are two servers at London Greenwich, with a total of 9 GHz CPU and 20 GB RAM. The two servers at University of Southampton both have 6.0 GHz and 16 GB RAM. For the home cluster in Southampton, the total hardware capability is 24.2 GHz CPU and 32 GB RAM. Simulations and experiments on a desktop and two private clouds (one in Southampton and one in London) get the same results. Thus, the execution time to complete all simulations is the benchmark to differentiate their performance on different platforms.

4.1.1 The execution time for running the Social Cloud in the local environment

This section describes the execution time for using the Social Cloud, with the objective to demonstrate that the Social Cloud is efficient, quick and accurate to produce good-quality results. This is also part of cybernetic validations to ensure that all services can work well following strategies from the closed-loop feedback system. The first step is to test the execution time in each process of the SocialMedia API in the local environment. It can be done on either server 2 at the University of Southampton or 1 of HPC servers at ULCC. Results are running one hundred times to get the average execution time. The aim is to ensure the cybernetic validation is robust and not affected by the distance between client-server requests. The standard deviation is always 0.05 seconds and below and p value is less than 0.005 presented in Table 6. When p value is under 0.005, which then support the case that all software cybernetic tests are in the controlled state, whereby the expected outcomes match with the theories that all functions in SocialMedia API can be completed in a short period of time with a very small error percentage. Another important experiment is to verify that all results are repeatable. All experimental results get the same outputs as presented in Section 3.

Table 6: Execution time for each API to process in the local environment ($p < 0.005$)

Process within the SocialMedia API	Average execution time (sec)	Standard deviation (sec)	Same results?
FriendsNetwork	1.15	0.02	Yes. As in Section 3.2.
LikeNetwork	1.03	0.02	Yes. As in Section 3.3.
LikeCommentNetwork	1.10	0.02	Yes. As in Section 3.4.
Posts	1.05	0.02	Yes. As in Section 3.5.

4.1.2 The execution time for running the Social Cloud between Southampton clusters

There are two sites that can process the Social Cloud fully, and one site is located at the University of the Southampton (server 2) and one site is at ULCC (HPC servers). There are two additional experiments required. The first experiment is to make a request from server 1 to server 2 within the University of Southampton. The physical location between server 1 and 2 is about 100 meters and the network upload speed is 1 Gbps during the time experiments took place. The second experiment is to make a request in Southampton and process in ULCC in London and will be presented in the next section. The aim is to test the execution time while network speed becomes an influential factor. This is to ensure that when one factor may change, it will not affect the outcome of the closed-loop feedback system, since it has a self-rectifying system that can adjust to necessary changes to the system to obtain the expected results. Results are running one hundred times to get the average execution time. The standard deviation is always 0.02 seconds and p value is less than 0.005 presented in Table 7, which confirm that all cybernetic validations are in the controlled state. Another supporting evidence is that the results of standard deviation are identical.

Table 7: Execution time for each API to process between Southampton clusters ($p < 0.005$)

Process within the SocialMedia API	Average execution time (sec)	Standard deviation (sec)	Same results?
FreindsNetwork	1.17	0.02	Yes. As in Section 3.2.
LikeNetwork	1.05	0.02	Yes. As in Section 3.3.
LikeCommentNetwork	1.12	0.02	Yes. As in Section 3.4.
Posts	1.07	0.02	Yes. As in Section 3.5.

4.1.3 The execution time for running the Social Cloud between Southampton and ULCC London clusters

This experiment is to make a request in Southampton and process in ULCC in London. The physical location between servers in Southampton and ULCC is 100 miles and the network upload speed is 100 Mbps during the time experiments took place. Results are running one hundred times to get the average execution time. The standard deviation is always 0.03 and below and p value is less than 0.005. See Table 8.

Table 8: Execution time for each API to process between Southampton and ULCC London clusters ($p < 0.005$)

Process within the SocialMedia API	Average execution time (sec)	Standard deviation (sec)	Same results?
FreindsNetwork	1.26	0.03	Yes. As in Section 3.2.
LikeNetwork	1.24	0.03	Yes. As in Section 3.3.
LikeCommentNetwork	1.31	0.03	Yes. As in Section 3.4.
Posts	1.36	0.03	Yes. As in Section 3.5.

Results show that despite of the network speed and physical distance difference, the difference in execution time is still small comparing execution time in Table 8. Four functions of SocialMedia API are designed not entirely to rely on network speed for service delivery, since network speed is useful to send back results from the server to the client.

4.2 Large scale simulations

Experiments conducted between Section 4.1 and 4.3 represent the execution time taken per simulation. To demonstrate the capacity and capability to handle Big Data for Software Cybernetics (SC), large scale experiments involved with 50,000 simulations per attempt are necessary. The objective is to simulate whether our hybrid Social Cloud can handle large number of users simultaneously and can be scaled up easily. Four functions in SocialMedia API can accommodate large-scale simulations simultaneously by typing

SocialMedia [“Facebook”, “FriendNetwork”, “50000”]

SocialMedia["Facebook", "LikeNetwork", “50000”]

SocialMedia["Facebook", "LikeCommentNetwork", “50000”]

This allows our APIs to simulate Social network analysis 50,000 times as if there are 50,000 users in the Social Cloud. We can start off from 5,000 simulations. Each time we perform 50,000 simulations more for four API functions: FriendsNetwork, LikeNetwork, LikeCommentNetwork and Posts until we reach out 50,000 simulations.

4.2.1 Software Cybernetics validation for large scale simulations

This section presents the Cybernetics validation involved with large scale simulations. Examples described in Section 2 and 3 are focused on a single job request that can demonstrate four functions. In the real use case scenario, Facebook always processes a large scale processing at all times while dealing with client-server requests from billions of users in the world. The purpose of this validation is to ensure that SocialMedia API can handle a large number of simulations at all times. The aim is to identify the maximum capacity that the large scale simulations of the SocialMedia API can offer, which is useful for capacity testing to know the maximum number of users can the system handle.

Figure 14 shows Software Cybernetics validation, whereby each time an increase of 10,000 simulations are used for capacity testing. All the tests will be undertaken at three times to determine the maximum capacity. All the four functions of the SocialMedioa API can perform up to 50,000 simulations per attempt. The four functions of SocialMedia cannot guarantee the simulations can be successfully when over 50,000 simulations have been used. As a result, the maximum recommended capacity is 50,000 simulations. Detailed experiments will be presented and the execution time will be recorded in Section 4.2.3.

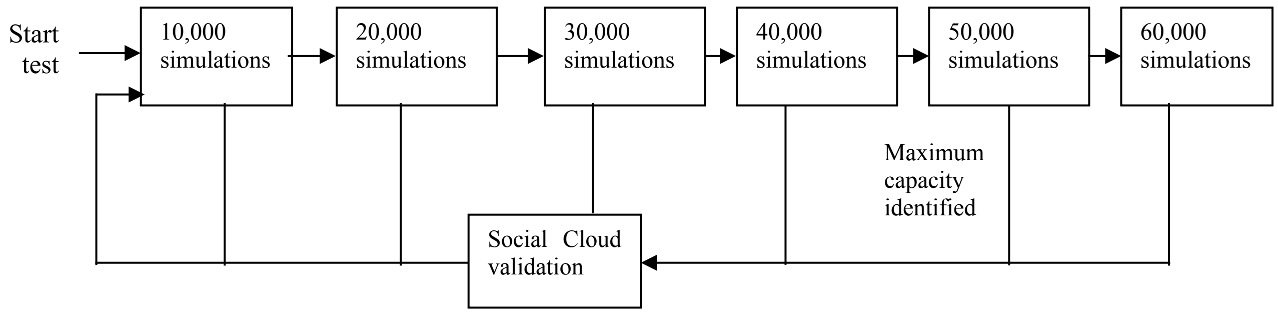

Figure 14: The Software Cybernetics validation for large scale simulations

4.2.2 The MapReduce framework

This section describes the MapReduce framework used to optimize the performance of running 50,000 simulations. As presented in Section 3, each API requires SQL queries to make FriendsNetwork, LikeNetwork and LikeCommentNetwork functional. In other words, SQL queries need to perform 50,000 times. If there are 50,000 operations without proper structuring, this will make processing and networking speed slow to respond. MapReduce framework itself has also adopted cybernetic approach. It splits the Big Data processing into Map and Reduce. In the Map step, it collects all the data and categorizes all of them together based on the common features. In the Reduce step, it processes the data from the Map step and presents the final output. See Figure 15 for the illustration of MapReduce approach.

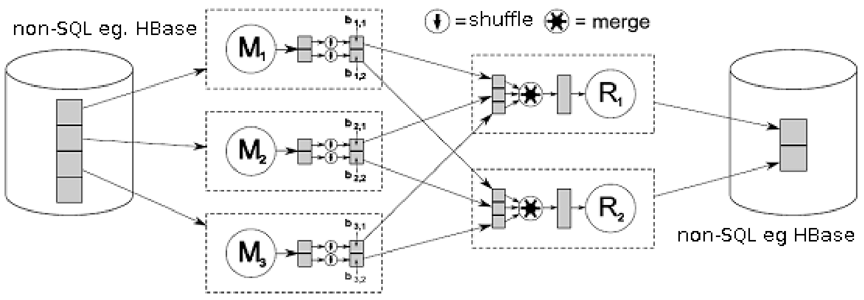

Figure 15: A MapReduce illustration for Social Cloud Big Data processing

To optimize the performance, MapReduce framework is used. Each query has a job ID and each job ID is equally distributed to each node of the hybrid cloud, being processed and results returned to the central node. The algorithm for our MapReduce is divided into “Map” and “Reduce” function. By having two main processes, it speeds up the distribution of resources and computation of Big Data processing. All the Map and Reduce steps have been written, created and presented as map( ) and reduce( ). This allows the ease of use since it can save architect a significant amount of time writing code, defining libraries and compiling all software resources each time. In summary, the Map function can read the datasets from the social data through Facebook. Additionally, it can calculate the nearest class center to the input data point. The output is presented by <key, value>, which includes <cluster category ID, record at attribute vector>. We assign each query as a cluster ID. The reasons to do so is to support the handling of data processing which is involved with thousands of records <key j, value j> produced in Map process. The Reduce function is to calculate the new clusterID associated with the Map function and is useful for the next round of MapReduce job. The form of the input data <key, value> is <cluster category ID, {record}>. However, the existing problem is that there are no considerations for software cybernetics while implementing MapReduce framework. It is important to do so, so that errors can be rectified at the earlier stage. Additionally, when tasks at one stage are completed, it can send back to the required stage to check and feedback any useful information. For example, if stage 3 of the tasks are completed and required to informed stage 4 and even stage 2 about completion of tasks, such as ‘job’ in HPC or Cloud Computing, software cybernetics approach will have the edge over traditional MapReduce, since there is no need to perform jobs again and find out where the problems are. The cybernetics approach can intelligently go to the job that has failed and start again. If this rectification process is successful, there is no need to run the entire jobs or run batch of jobs to complete the required task. In order to demonstrate this, two functions, cybernetics-1( ) and cybernetics-2( ) are developed. The function for cybernetics-1( ) is similar to map( ), with the difference in that it can do self-rectification when spotting errors or informing to the next stage when the jobs at the current stage are completed. The function for cybernetics-2( ) is similar to reduce( ), with the similar aims like cybernetics-2( ). Table 9 show the algorithm for both cybernetics-1 and cybernetics-2.

Table 9: The algorithm for cybernetics-1 and cybernetics-2 function

```
void cybernetics-1 {
for(i = 0;i < k;i++){
if (dis(point, cluster [i]) >= whereareyou){
whereareyou = dis(point, cluster[i]);
current cluster ID = i;}}
output (current clusterID, point);}
end;
void cybernetics-2 {
while (points. Has Next()){
Point Writable current point = points. next();
Num + =current point. get num();
for(i = 0;i < reducer;i++){
sum[i] + =current point. point [i];}
for(i = 0;i < reducer;i++)
mean[i] = sum[i]/num;
out(key, mean);}
end;
```

MapReduce is focused on the handling of jobs and is not designed to check and rerun any incomplete or failed jobs. Thus, additional work is required to achieve it. This motivates us to design a new function called “cybernetics”, which checks all incomplete and failed job and ensure they can be processed by MapReduce. The cyberbetic step can combine the advantages from both the Map and Reduce steps. In Table 11, the variable “whereareyou” is to check that all job IDs have been looked after and then processed by Map step. What would then happen if the value is equal or larger than “whereareyou”, then a rectification process, cybernetics-1, is required to ensure all incomplete or failed jobs are processed. Table 10 demonstrate Reduce function that processes the outputs from Table 9. The main code can be reused and is called cybernetics-2, since it only deals with the incomplete and failed jobs. To perform job submission and completion in the cybernetics social Cloud, Table 10 shows the core algorithms, which reduce the length of code and any unnecessary complexity involved.

The main advantage includes the improvement of execution time to complete all the jobs since less time is required to run the code before starting any job submission till the job completion. Two additional functions, cybernetics-1 and cybernetics-2 can help processing Big Data for social network applications. To demonstrate the performance, additional experiments have been designed. Results will be presented in Section 4.2.

Table 10: Job submission and completing while using cybernetics approach

```
void job {
cybernetics-1( )
cybernetics-2( )
}
end;
```

4.2.3 Experiments on four API functions and MapReduce for running up to 50,000 simulations

This section is focused on the performance of the four API functions between Section 3.2 and 3.5 and the MapReduce framework in Section 4.2.2. The execution time was taken three times to get the mean values and standard deviations. This includes performing simulations on the local environment (either one Private Cloud in Southampton, one Private Cloud in London), between Southampton Private Cloud clusters and between Southampton and London Private Cloud clusters in Section 4.1. We present results as follows.

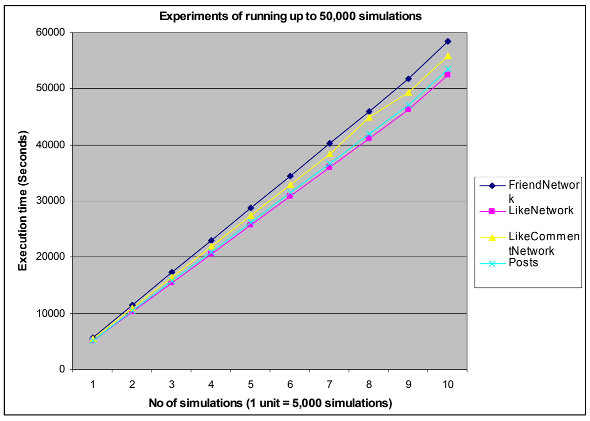

Figure 16: Experiments of four API functions in the local environment

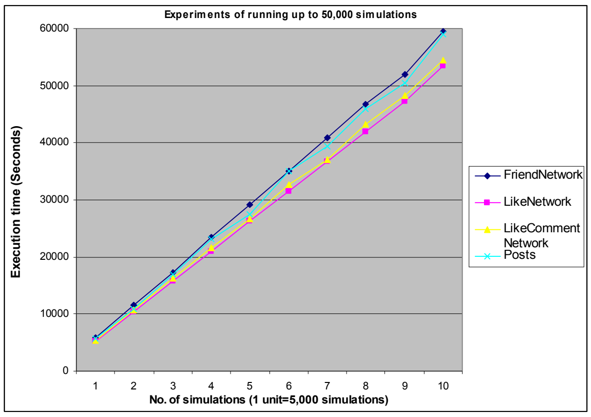

Figure 17: Experiments of four API functions between Southampton clusters

Figure 16 and Figure 17 show experiment results of four API functions in the local environment and Southampton cluster respectively. Each unit in the figure represents 50,000 simulations. All the execution time were completed under 60,000 seconds, or 16 hours and 40 minutes, for 50,000 simulations. "FriendNetwork" function took the longest and "LikeNetwork" function took the shortest execution time in both experiments, although their differences were within 5%. Our setup can cope with a large number of queries and data processing.

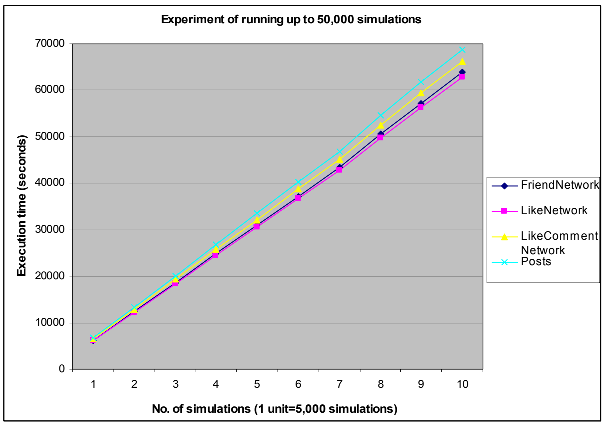

Figure 18: Experiments of four API functions between Southampton and ULCC London clusters

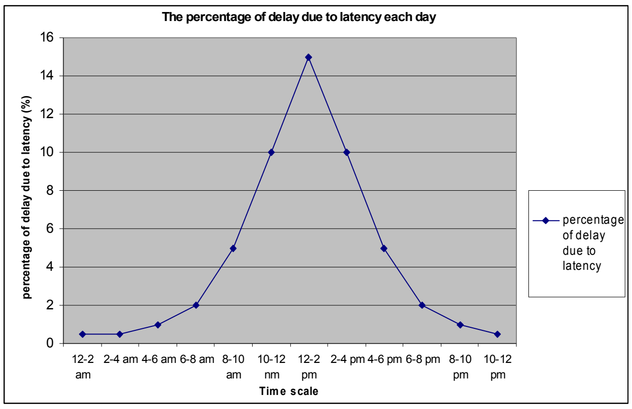

Figure 19: The percentage of delay due to latency each day

Figure 18 shows experiment results of processing four API functions between Southampton and ULCC London clusters. Due to the distance involved, the network latency was expected resulting in longer execution time, all of which were completed within 70,000 seconds, or within 19 hours and 26 minutes. The network latency experiment were conducted and tested while in the process of completing our previous project in Chang (2014). Network latency depends on the demands on the network resources. It has the lowest points in off-peak hours and can reach as high as 15% in peak hours as shown in Figure 19. During the peak hours, the demands on the network requirement remain high and may affect the overall processing speed, which explain the additional 10,000 seconds in the execution time. Additionally, all of 50,000 simulations can be processed successfully. All results have been produced and experiments demonstrate a high level of reliability.

The next section is to present the percentage of incomplete and failed jobs with and without the use of cybernetics-1 and cybernetics-2 functions. Incomplete or failed jobs means they are required to be run again, resulting in a longer execution time. Results in Figure 20 show the percentage of jobs completed for the first time with the cybernetics-1 and cybernetics-2, With cybernetics functions, all the job stayed with 100% all the ways to 50,000 simulations. On the other hand, the percentage of job completed for the first time dropped while the number of simulations increased and the rate of drop was more than a linear regression. In the experiment, 96.4% of all jobs were completed successfully for the first time, meaning that an additional 3.6% of time was required, if the modified MapReduce could already identify where the failed or incomplete jobs were. Assuming this was the case and the longest time to complete 50,000 simulation was 68,801 seconds in our experimental results under repeated tests. It means that at least 68801 x 0.0036 = 233.9 seconds would be require to run failed or incomplete jobs should the job IDs and locations of the failed/incomplete jobs were found.

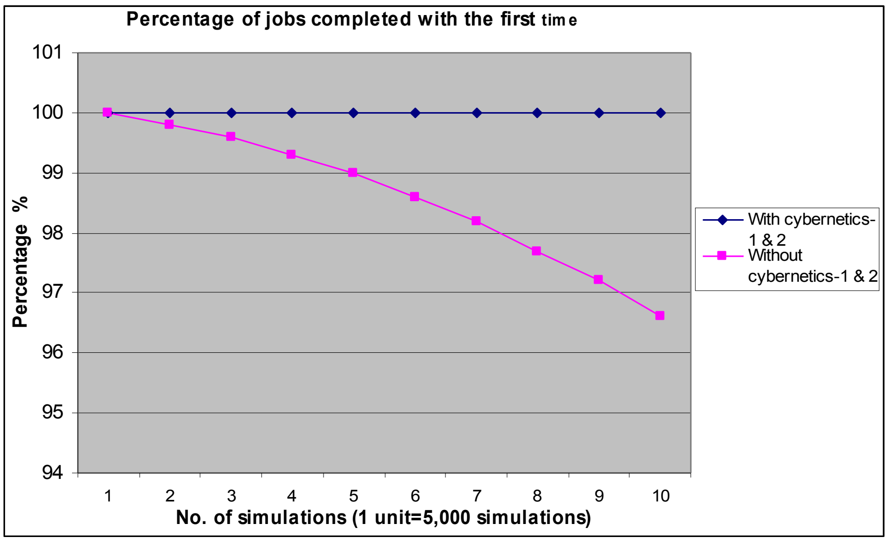

Figure 20: Percentage of jobs completed for the first time.

4.3 Client-server experiments for Software Cybernetics

Referring to Section 2.5 and Figure 13 in Section 3.6, this section discusses whether four main functions of SocialMedia API have passed the client-server tests. Figure 21 shows Software Cybernetics tests inclue the single and large-scale client-server tests for all the functions. Single client-server tests were involved with tests in the localhost, between Southampton clusters and between London and Southampton cluster, where all the results could confirm the positive outcome. The large-scale simulations could use MapReduce framework, in which the API could accommodate 50,000 simulations for all the four functions of the SocialMedia API. Results could assert all the experiments could pass the tests and could successfully complete Cybernetics validation for social network analyses.

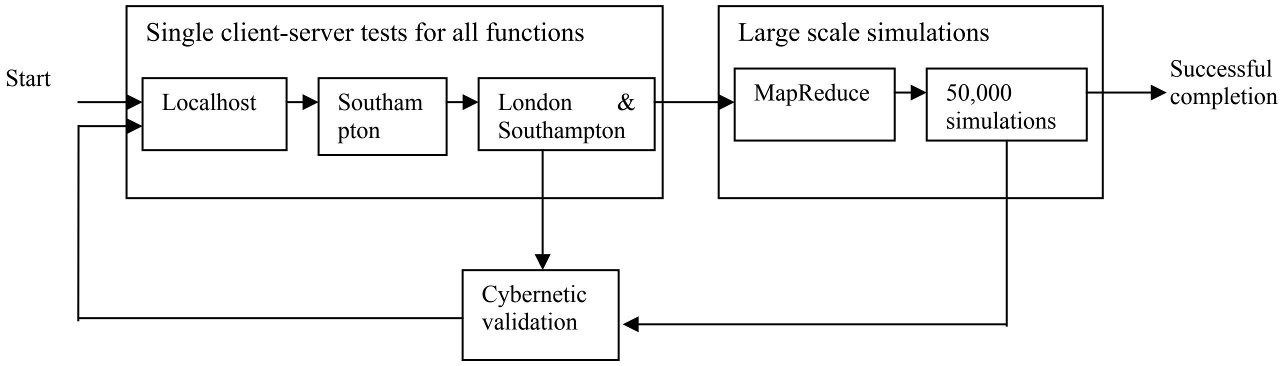

Figure 21: Client-server experiments of SocialMedia API testing for Software Cybernetics

5. Discussion

The Social Cloud presented in this paper can deal with the Big Data associated with social network and also presents a way to process and present all the data. There are four topics of discussion to support the validity of our Social Cloud.

5.1 Summary of the Social Cloud and key lessons learned

While following the steps described in Section 2 and 3, the Big Data can be managed more easily and presented in a way that can be more easily understood by the public, without the need to go through threads of text and a huge number of test queries undertaken by the

BOINC servers. The use of visualization can ease the complexity to understand the relationship and the extent of interactions between different contacts in the social network.

Section 2 describes the architecture and a solution for a low-cost yet effective way of demonstrating the effective use of the Social Network. Our contribution is demonstrated in Section 3 and 4 in the use of the SocialMedia API, which can present the massive and complex data collected on the Facebook. Section 3 describes the use of the core syntax, and there are four examples using the author's account to illustrate. The first three examples can query the required information and present the outputs as visualization, which also represent the number of contacts, the frequencies they click like or comment on the author's posts, and their relationship between one another. The last example can query and analyze the text-based information about the posts. All their core syntax was explained. Section 4 demonstrates the experiments focusing on the performance of running the Social Cloud, and all API syntax can complete the processing within 1.36 seconds.

Section 4 also present experiments of running up to 50,000 simulations in which each simulation included queries and data processing on the APIs. Four API functions: FriendsNetwork, LikeNetwork, LikeCommentNetwork and Posts were thoroughly tested in the local environment, between Southampton clusters and between Southampton and London ULCC clusters. Results were recorded and showed that all the execution time was completed in less than 70,000 seconds. "LikeNetwork" took the shortest execution time and "FriendsNetwork" took the longest execution time in the first two large scale simulations and "LikeNetwork" took the shortest execution time and "Posts" took the longest execution time in the final large scale simulations. Impacts due to network latency have been explained.

The same principle can be applied to other individuals who wish to know their status and strength of their relationship between their peers, colleagues, collaborators and suppliers. Additionally, companies can get potential benefits from further redevelopment of this service. Companies can learn how people like, react and comment on their latest products and services. They can write APIs to understand the number of views, hours and other implicit information (such as whether their allies recommend to their networks and frequency of doing so). This may offer incentives to firms that try to apply the benefits of market research.

5.2 Cybernetics for Social Cloud and Big Data

Section 2.5 described the system design in SocialMedia API with its four functions and explained each function, steps involved and system design diagrams. Section 3 demonstrated how to use these four API functions with outputs in visualization and the code syntax presented. Results support the Social Cloud and Big Data processing since thousands of information can be processed and presented in ways that users could understood easily without the need for further programming. Visualization was also focused on the relationship between people with regard to the strength of friendship, likes, comments and posts given. It showed the dynamic interactions between people without revealing the underlying complex information but pinpointing the facts between the relationship with other networks. Experiments in Section 4 were used to support the effectiveness of the Social Cloud. All the execution time for single client-server requests at the local environment, between Southampton clusters and between Southampton and ULCC clusters took 1.36 seconds and below. Large scale simulations had been performed. To demonstrate that, Cybernetics validation system diagram was presented to demonstrate all verification was done step by step. MapReduce framework was the one that can process large amount of data an optimize performance for large scale simulations upon receiving jobs to process data. But the current MapReduce framework did no specifically deal with incomplete or failed jobs. Hence, two additional functions, cybernetics-1 and cybernetics-2 were designed for this purpose. The experimental results showed that both cybernetics functions could support Big Data processing with a low execution time and a good performance. The maximum capacity of 50,000 simulations can be identified. In other words, our API can handle 50,000 client-server requests in real time. Experimental results also confirmed that there was an excellent performance (low execution time) for all client-server tests.

Figure 22 shows the Software Cybernetics for validating our proposed Social Cloud and the sequence of presenting the work in this paper. All the work presented in this paper can support Cybernetics for Big Data as follows. Section 2 discusses the Social cloud design illustrated by the BIONIC architecture and UML. The working design will lead to the development of APIs. Each of the API will be tested to show whether there is a consistency for each function such as retrieving and interpreting the data in the social network, whereby Section 3 will present functions of each developed API and results of the API outputs. Testing will be required for this stage and if APIs do not produce the expected output, redesign of the Social Cloud prototype will be required. Since APIs require the client-server experiments to validate satisfactory outputs, experiments with single client-server request and large scale client-server requests will be used to demonstrate which will be presented in Section 4. At this stage, external disturbances may happen, which include the interruptions of network or failures of large scale simulations which may interrupt the entire experiments. Vigorous tests will be required to ensure that all client-server experiments provide satisfactory outputs. If there are, then the Social Cloud can be successfully validated. If not, the Social Cloud will be re-investigated in the previous stages to find out the sources of problems or reasons that can cause failures.

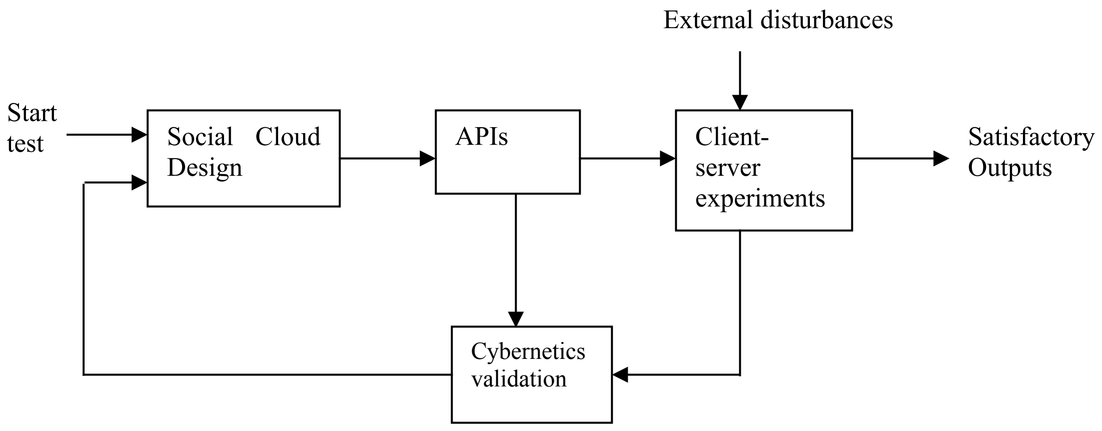

Figure 22: The Cybernetics for the Social Cloud

5.3 Comparison with other approaches

Comparison with related work is relevant for research and development. With regard to Mislove et al (2009)'s view that visualization offers a research challenge, there are recent work based on the API development to process a large number of data and present analyses in visualization like our paper does. The aim is to simplify the process of software development since the emphasis is on the API in the Cloud rather than software development on the desktop connecting to the Cloud. Khalid et al (2014) propose their Mobile Social Network systems and explain their architecture. They describe the major components and propose their recommended framework. They introduce the rank system, explain the related algorithms and illustrate an example. They explain their strategies and have their performance evaluation. However, their work is on Mobile Social Networks and is not a generic model. They do not mention any types of Mobile platforms and networks to evaluate their work, assuming their work is relevant to the Mobile networks in the US. Canton et al (2014) explain sharing their infrastructure resources via Social Networks. They describe their model, architecture, evaluation and their solution to resolve existing challenges. Experiments are based on the number of ranked users and event number. However, they do not describe whether these users are simulated agents or real users, although they imply they are agents. There are no enough technical details to determine the novelty of the research, if this work is based on the continuation of their previous research (Chard et al., 2010, 2012). Klein et al (2013) do not work on Social Cloud but they present an interesting approach relevant to this. They develop five algorithms known as “simulateExecution”, “initialOpProbs”, “distributeTotalProbs”, “distributeTypeProbs” and “distributeProbs”. However, such an approach regards each query as first-come-first-serve basis. The Social Cloud has the freedom to allow different people to simultaneously post comments in different blogs. Although first-come-first-serve basis is a popular option, another algorithm can be developed to determine the sequence and status of posting, which may include rank in the community, priority and strength of friendship. All the literatures above do not use comprehensive Cybernetics for software design, implementation, Social Cloud Big Data processing and experiments. Results from the experiments support that Cybernetics can be used for Social cloud and Big Data processing. The adoption of Software Cybernetics approach illustrated in this paper can help improve the quality of outputs since all the work recommend to use Cybernetics approach with step-by-step approaches.

5.4 Security

We adopt the Cloud Computing Adoption Framework (CCAF) multi-layered security service, developed in another research project, which include the combined uses of three major security solutions: (1) access control and firewall; (2) an identity management and (3) encryption (Chang and Ramachandran, 2016). The use of CCAF multi-layered security services can ensure users in a secure and protected environment supported by a large number of experiments, penetration and user testing. Additionally, all the queries and temporary files are deleted at the end of each service to ensure the user security and prevent any data leak. There is also a surveillance and tracking system to monitor any unusual activities and report to the user instantly while detecting any viruses, trojans and risks that can post a threat to the users. The use of CCAF multi-layered security to ensure all the data are protected against the threats and alerted to the system manager for enforced protection when detecting unauthorized attempts to gain access.

5.5 Contributions to the Big Data processing for Software Cybernetics

Social network sites generate a massive amount of data and the Big Data processing becomes challenging as follows. First, the amount of the data collected can be huge and requires petabytes (Bryant, Katz and Lazowska, 2010). Second, the quality of the data is important because only useful data should be processed and analyzed (Quackenbush, 2002; Chang et al., 2011; Chang, 2014). Third, data processing should be fast and the steps involved should be efficient, effective and easy to handle (Han, Kamber and Pei, 2006; Cohen et al., 2009). Fourth, the costs of processing Big Data should be as low as possible (Jacobs, 2009; BOINC, 2013). All these contribute to the requirements of modern Software Cybernetics. This paper can demonstrate meeting four core requirements confirmed as follows.

1. The proposed solution uses the BOINC servers and Facebook (particularly the latter) to process the Big Data, so that scientists need not manage petabytes of data directly.
2. The proposed solution is focused on processing the contacts who click like or comment on the author’s posts. Results in visualization and core syntax have been presented.

3. All data processing of a single client-server request can be completed within 1.36 seconds.
4. Large scale simulations can be tested thoroughly to ensure the Social Cloud service can accommodate a large number of requests and data processing of the user's own networks.
5. There is no cost involved due to the benefits of volunteer computing and Facebook.

Big Data has five characteristics: volume, velocity, variety, veracity and value (Chen et al., 2012). Our work has clearly demonstrates velocity and variety as follows. First, the data generates by the researcher can be in different forms and can grow significantly over a period of time. For example, when the researcher publicized his first international workshop on Emerging Software as a Service and Analytics 2014 (ESaaSA 2014), it generated thousands of viewings on social network. These includes 20 clicked likes, 7 left comments, 150 views from the network and 2,070 views from friend's networks or strangers who have seen the posts advertised by the researcher. Second, there are a different variety of data being posted. This includes pictures taken during the leisure, visits and work; discussion about work, interests, plans, opinions, religious belief, events and news; the author's network news such as wedding, passing PhDs, getting funding, plans, personal views and holidays. A variety of data in different file formats has been read, processed and presented in visualization as discussed in Section 3 and 4. The APIs in the Social Cloud should be intelligent enough to understand the differences and make the best sense from the data, so that anyone without technical knowledge can understand.

6. Conclusion and Future Work

This paper presents our solution for the Social Cloud. We present the combined approach of using BOINC and Facebook, where all data processing takes place. The architecture and the related information of the BOINC project have been discussed. The creation of SocialMedia API can ensure a smooth delivery of Big Data processing in the Social Cloud. Four examples based on our experience are given to support the validity of Big Data processing. The first example is focused on presenting the demographics of the author's contacts, of who have also interacted with one another as the author's mutual friends. The second example is focused on retrieving and displaying the contacts who have clicked likes on the authors' posts and they are mutual friends. The third example is focused on retrieving and displaying on the contacts who have either clicked like or have left comments on the author's posts. The fourth example is focused on the retrieving the text-based information for those who have either clicked like or commented. These examples are supported by the use of the core syntax presented for each case. These four functions of the SocialMedia API have undertaken experiments to test on its performance. Three environments were set up: the local environment, between the Southampton clusters and between ULCC London and Southampton clusters. All processes in SocialMedia API can be completed very efficiently within 1.36 seconds. Up to large scale of 50,000 simulations were undertaken for four API functions. "LikeNetwork" took the shortest execution time in three large scale simulations. "FriendsNetwork" took the longest execution time in the first two large scale simulations and "Posts" took the longest execution time in the final large scale simulations. Our APIs have been thoroughly tested to ensure that large scale data processing can be completed smoothly. The proposed solution presented in this paper can also meet the four challenges for Big Data research as presented in the Discussion section. Our proposed solution is easy to use, being able to handle large scale simulations and cost-free.

A comprehensive Cybernetics for Social Cloud and Big Data processing has been fully illustrated. The four functions in SocialMedia API had followed Cybernetics for system design and implementation. Following MapReduce framework with the improved "cybernetics" functions, single client-server requests and large scale simulations had low execution time. Results demonstrated an excellent performance and 100% job completion rate for 50,000 simulations. The benefits of doing so can ensure a high quality of outputs and standards to be maintained, which are better off than the existing literature that do not employ Cybernetics approach for Social Cloud and Big Data processing.

Future work may contain two streams of concurrent development. The first stream is focused on the development of more processes offered by SocialMedia, so that it can take on more types of data processing. Advanced mathematical models will be investigated to study how to process more complex data, and the possibility of introducing neutral network or business intelligence systems for our proposed Social Cloud. The second stream is focused on the development of analyzing other social network websites such as Twitter and LinkedIn, so that our future work can handle Big Data processing on the major social network websites. The four functions of SocialMedia API will increase up to 200,000 simulations per attempt to improve on the capacity management. More research and development will be continued to ensure better algorithms can be fully incorporated with visualization and data processing techniques.

References

Anderson, D. P., Fedak, G., 2006. The Computational and Storage Potential of Volunteer Computing, Cluster, Cloud and Grid Computing (CCGrid), Vol. 1, May, pp 73-80.

Bai, X., Chen, Y., & Shao, Z., 2007. Adaptive web services testing. In IEEE 31st Annual International Conference on Computer Software and Applications, COMPSAC 2007, July, Vol. 2, pp. 233-236.

British Council, 2013. Welcome to the World: Royal birth at the age of social media, article, Turkey, July 26.

BOINC project, 2013. Online doumentation and statistics, http://boinc.berkeley.edu/, based on the latest update on Dec. 28.

Bryant, R., Katz, R. H., Lazowska, E. D., 2010. Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science and Society, technical paper, pp 1-15.

Cai, K. Y., 2002. Optimal software testing and adaptive software testing in the context of software cybernetics. Information and Software Technology, 44(14), 841-855.

Cai, K. Y., Cangussu, J. W., DeCarlo, R. A., & Mathur, A. P., 2003. An overview of software cybernetics. In the Eleventh Annual IEEE International Workshop on Software Technology and Engineering Practice, September, pp. 77-86.

Canton, S., Haas, C., Chard, K., Bubendorfer, K., & Rana, O., 2014. A Social Compute Cloud: Allocating and Sharing Infrastructure Resources via Social Networks. IEEE Transactions on Services Computing, 7(3).

Chang, V., De Roure, D., Wills, G., John Walters, R., Barry, T., 2011. Organisational Sustainability Modelling for Return on Investment (ROI): Case Studies Presented by a National Health Service (NHS) Trust UK. Journal of Computing and Information Technology, 19(3), 177-192.

Chang, V., 2014. A proposed model to analyse risk and return for Cloud adoption, ISBNs: 9783659587696 (print), Lambert

Chang, V. and Ramachandran, M. 2016. Towards achieving Big Data Security with the Cloud Computing Adoption Framework, IEEE Transactions of Services Computing, forthcoming.

Chard, K., Caton, S., Rana, O., Bubendorfer, K., 2010. Social Cloud: Cloud Computing in Social Networks, The IEEE 3rd International Conference on Cloud Computing, Miami, USA, 5-10 July.

Chard, K., Bubendorfer, K., Caton, S., Rana, O. F., 2012. Social cloud computing: A vision for socially motivated resource sharing. IEEE Transactions on Services Computing 5(4), pp. 551-563.

Chen, H., Chiang, R. H., & Storey, V. C., 2012. Business Intelligence and Analytics: From Big Data to Big Impact. MIS quarterly, 36(4), 1165-1188.

Chen, J., Zhang, Q., & Bruda, S. D., 2009. Cybernetics in Software System Verification. In IEEE International Conference on Intelligent Human-Machine Systems and Cybernetics, 2009. IHMSC'09, Aug. Vol. 2, pp. 274-277.

Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J. M., Welton, C., 2009. MAD skills: new analysis practices for big data. Proceedings of the VLDB Endowment, 2(2), 1481-1492.

Costa, F., Silva, L., Dahlin, M., 2011. Volunteer Cloud Computing: MapReduce over the Internet, 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), Shanghai, China, 16-20 May.

Facebook, 2013. Graph API for Developers, https://developers.facebook.com/docs/graph-api/ and statistics, http://newsroom.fb.com/, based on the latest update on Dec. 20.

Farkas, M. G., 2007. Social Software in Libraries: Building Collaboration and Communication Online, Information Today Inc, ISBN 978-1-57387-275-1.

Foster, I., Kesselman, C., and Tuecke, S., 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International Journal of High Performance Computing Applications Fall 2001 15(3), pp 200-222.

Glanz, K., Rimer, B. K., Viswanath, K., 2008. Health Behavior and Health Education: Theory, Research, and Practice, Wiley Publishers, ISBN 978-0-7879-9614-7.

Gross, R., Acquisti, A, 2005. Information Revelation and Privacy in Online Social Networks (The Facebook case), ACM Workshop on Privacy in the Electronic Society (WPES), Alexandria, VA, USA, Nov 7.

Han, J., Kamber, M., Pei, J., 2006. Data mining: concepts and techniques, Morgan kaufmann, ISBN 978-1-55860-901-3.

Jacobs, A, 2009. The pathologies of big data. Communications of the ACM, 52(8), 36-44.

Khalid, O., Khan, M., Khan, S., & Zomaya, A., 2014. Omnisuggest: a ubiquitous cloud based context aware recommendation system for mobile social networks, IEEE Transactions on Services Computing, 7(3).

Klein, A., Fuyuki, I., & Honiden, S., 2013. SanGA: A self-adaptive network-aware approach to service composition. Services Computing, IEEE Transactions on, vol. PP, (99), 1-1.

Mislove, A., Marcon, M., Gummadi, K. P., Druschel, P., & Bhattacharjee, B., 2007. Measurement and analysis of online social networks. In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, October, pp. 29-42.

Quackenbush, J., 2002. Microarray data normalization and transformation. Nature genetics, 32, 496-501.

Suh, B., Hong, L.C, Pirolli, P., Chi, E. H., 2010. Want to be Retweeted? Large Scale Analytics on Factors Impacting Retweet in Twitter Network, The IEEE Second International Conference on Social Computing (SocialCom), Minneapolis, MN, USA, 20-22 Aug.