Architecture of a Cloud-Based Social Networking News Site

Jeff Luo, Jon Kivinen, Joshua Malo, Richard Khoury
Department of Software Engineering, Lakehead University, Thunder Bay, Canada
Email: {yluo, jlkivine, jmalo, rkhoury}@lakeheadu.ca

***Abstract*—Web 2.0 websites provide a set of tools for internet users to produce and consume news items. Prominent examples of news production sites include Issuu (issuu.com) and FlippingBook (page-flip.com) which allows users to upload publication files and transform them into flash-animated online publications with integrated social-media-sharing and statistics-tracking features. A prominent example of news consumption site is Google News (news.google.com), which allows users some degree of control over the layout of the presentation of news feeds, including trusted news sources and extra category keywords, but offers no real editing and social sharing components. This proposed project bridges the gap between news production sites and news consumption sites in order to offer to any user - including non-profit organizations, student or professional news media organizations, and the average Internet user - the ability to create, share, and consume social news publications in a way that gives users complete control of the layout and content of their paper, the facilities to share designs and article collections socially, as well as provide related article suggestions all in a single easy to use horizontally scaling system.**

***Index Terms*—Web 2.0, Web engineering, Cloud computing, News, Social networking, Recommendation systems**

I. Introduction

Web 2.0 applications, and in particular social networking sites, enjoy unprecedented popularity today. For example, there were over 900,000 registered users on Facebook at the end of March 2012[1], more than the population of any country on Earth save China and India. In a parallel development, a recent survey by the Pew Research Centre [1] discovered that an overwhelming 92% of Americans get news from multiple platforms, including in 61% of cases online news sources. Moreover, this survey showed that the news is no longer seen as a passive "they report, we read" activity but as an interactive activity, with 72% of Americans saying they follow the news explicitly because they enjoy talking about it with other people. The social aspect of news is dominant online: 75% of online news consumers receive news articles from friends, 52% retransmit those news articles, and 25% of people contribute to them by writing comments. There is also a clear intersection between social networks and news consumption: half of social-network-using news consumers use the social network to receive news items from their friends or from people they follow, and a third use their social networking site to actively share news. However, news portal sites such as GoogleNews remain the single most popular online news source.

There is thus a clear user need for a social networking news site. Such a site will combine the interactive social aspect of news that users enjoy with the diversity of news sources that portal sites offer.

The aim of this paper is to develop a new Web 2.0 site that offers any news publisher - including non-profit organizations, student or professional news media organizations, and the average Internet user - the ability to create and share news publications in a way that gives them complete control over the layout and content of their paper as well as the sharing and accessibility of their publications. The system will allow readers to discover new publications by offering helpful suggestions based on their current interests, their reading patterns, and their social network connections. It will also allow them to share news, comments, and interact generally with other readers and friends in their social network. Finally, the cloud platform will offer a horizontally-scaling and flexible platform for the system. The contribution of this paper is thus to present the design and development of a new Web 2.0 special-purpose social networking site.

From a web data mining point of view, implementing and controlling such a system opens up a lot of very interesting avenues of research. The news articles available on the system will create a growing text and multimedia corpus useful for a wide range of research projects, ranging from traditional text classification to specialized applications such as news topic detection and tracking [2]. Social networking platforms now supply data for a wide range of social relationship studies, such as exploring community structures and divides [3]. And feedback from the recommendation system will be useful to determine which variables are more or less influential in human decision-making [4].

The rest of this paper is organized as follows. The next section will present a brief overview of related projects. We will give an overview of the structure of our proposed system in Section III. In Section IV we will present in more details the functional requirements of each of the system's components. We will bring these components

[1] http://newsroom.fb.com/content/default.aspx?NewsAreaId=22

doi:10.4304/jetwi.4.3.227-233

together and present the overall architecture of the system in Section V. Next, Section VI will discuss implementation and testing considerations of the system. Finally, we will present some concluding thoughts in Section VII.

II. Related Works

There has been some work done in developing alternative architectures for social networking sites. For example, the authors of [5] propose a mobile social network server divided into five components: an HTTP server that interacts with the web, standard profile repository database and privacy control components, a location database that allows the system to keep track of the user's location as he moves around with his mobile device, and the matching logic component that connects the other four components together. The authors of [6] take it one step further to create a theme-based mobile social network, which is aware not only of the user's location but also of activities related to his interests in his immediate surroundings, of their duration and of other participants. Our proposed system is also a theme-based social network, as it learns the users' interests both from what is stated explicitly in their profiles and implicitly from the material they read, and will propose new publications based on these themes. However, the architectures mentioned above were based on having a single central web server, in contrast to our cloud architecture.

Researchers have been aware for some time of the network congestion issues that comes with the traditional client-server architecture [7]. Cloud-based networking is a growingly popular solution to this problem. A relevant example of this solution is the cloud-based collaborative multimedia sharing system proposed in [8]. The building block of that system is a media server that allows users in a common session to collaborate on multimedia streams in real time. Media servers are created and destroyed according to user demand for the service by a group manager server, and users access the system through an access control server. The entire system is designed to interface with existing social networks (the prototype was integrated to Facebook). By comparison, our system is not a stand-alone component to integrate to an external social network, but an entire and complete social network.

There are many open research challenges related to online news data mining. Some examples surveyed in [9] include automated content extraction from the news websites, topic detection and tracking, clustering of related news items, and news summarization. All these challenges are further compounded when one considers that online news sources are multilingual, and therefore elements of automated translation and corpus alignment may be required. These individual challenges are all combined into one in the task of news aggregation [9], or automatically collecting articles from multiple sources and clustering the related ones under a single headline. News aggregate sites are of critical importance however; as the PEW survey noted, they are the single most popular source of online news [1]. Our news-themed social network site would serve as a new type of news aggregate site.

Researchers working on recommendation systems have shown that individuals trust people they are close to (family members, close friends) over more distant acquaintances or complete strangers. This connection between relationship degree and trust can be applied to social networks, to turn friend networks in to a Web-of-trust [10]. The trust a user feels for another can be further extrapolated from their joint history (such as the number of public and private messages exchanged), the overlap in their interests, or even simply whether the second user has completed their personal profile on the site they are both members of [4]. It is clear, then, that social networks are a ripe source of data for recommendation systems. Our proposed system is in line with this realization.

One of the key areas of applied research today in Cloud is on performance and scalability. The authors of [11] propose a dynamic scaling architecture via the use of "a front-end load balancer routing user requests to web applications deployed on virtual machines (VMs)" [11], aiming to maximizing resource utilization in the VMs while minimizing total number of VMs. The authors also used a scaling algorithm based on threshold number of active user sessions. Our proposed system adopts this approach, but considers the thresholds of both the virtual machines' hardware utilization as well as the number of active user-generated requests and events, instead of sessions. Further, our system adopts the performance architecture principles discussed in [12] to examine the practical considerations in the design and development of performance intelligence architectures. For performance metrics and measurements, our system adopts the resource, workload, and performance indicators discussed in [13] and the approach discussed in [14] to utilize the server-side monitoring data to determine the thresholds and when to trigger a reconfiguration of the cloud, and the client-side end-to-end monitoring data to evaluate the effectiveness of the performance architecture and implementation designed into the system, as it would be felt and perceived by the users of the system.

It thus appears that our proposed system stands at the intersection of several areas of research. Part of its appeal is that it would combine all these components into a single unified website, and serve as a research and development platform for researchers in all these areas. Likewise, the web data generated by the system would be valuable in several research projects. And all this would be done while answering a real user need.

III. System Overview

There are eight major components to our proposed system:

1. The cloud component serves the overall function of performing active systems monitoring, and providing a high performance, horizontally-scaling back-end infrastructure for the system.
2. A relational database system is essential to store and retrieve all the data that will be handled by the

system, including user information, social network information, news articles, and layout information.

3. The social network aspect of the system is crucial to turn the passive act of reading news into an active social activity. Social networking will comprise a large portion of the front-end functionalities available to users.
4. A natural language processing engine will be integrated into the system to analyze all the articles submitted. It will work both on individual articles, to detect its topic and classify it appropriately, and on sets of articles, to detect trends and discover related articles.
5. A suggestion engine will combine information from both the social network and the natural language processor in order to suggest new reading materials for each individual user.
6. The content layout system is central to the content producer's experience. It will provide the producer a simple and easy-to-use interface to control all aspects of a news article's layout (placement of text and multimedia, margins and spacing, etc.) and thus to create their own unique experience for the readers.
7. The user interface will give the reader access to the content and will display it in the way the producer designed it. The interface will also give the user access to and control over his involvement in the community through the social network.
8. The business logic component will facilitate user authentication and access control to ensure users are able to connect to the system and access their designs and article collections, and prevent them from accessing unauthorized content.

IV. System Requirements

Each of the eight components we listed in Section II is responsible for a set of functionalities in the overall system. The functional requirements these components must satisfy in order for the entire system to work properly are described here.

A. Cloud

The Cloud component responds to changes in processing demand by modifying the amount of available computing resources. It does this by monitoring and responding to traffic volume and resource usage, and creating or destroying virtual resources as needed. Additionally, the Cloud component is responsible for distributing the traffic load across the available resources. The Cloud must be designed to support the following functionalities:

- ability to interface with a Cloud Hypervisor [15] that virtualizes system resources to allow the system to control the operation of the Cloud;
- ability to perform real time monitoring of the web traffic and workloads in the system;
- ability to monitor the state and performance of the system, including its individual machines;
- ability for individual virtual servers within the system to communicate with each other;
- ability for the virtual servers to load share and load balance amongst each other;
- ability to distribute workloads evenly across the set of virtual servers within the system;
- ability to add or remove computing resources into the system based upon demand and load;
- ability to dynamically reconfigure the topology of virtual servers to optimally consume computing resources;
- ability to scale horizontally.

B. Database

The database component is required for persistent storage and optimized data retrieval. The database must be designed to support the following functionalities:

- represent and store subscribers and authors of each given publication;
- represent and store layouts of individual articles;
- represent and store sets of linked articles to form a publication;
- represent and store the social network.

C. Social Network

The social network component provides users with a richer content discovery experience by allowing users to obtain meaningful content suggestions. It must support the following functionalities:

- support user groups (aka friend lists);
- map social relationships;
- model user interactions with articles and publications;
- control sharing and privacy;
- comment on articles.

D. Natural Language Processor

The natural language processor allows the articles in the system to be analyzed and used for content suggestions and discovery. A basic version can simply build a word vector for each article, and computes the cosine similarity with the word vectors of other articles and of categories. The natural language processor must support the following features:

- ability to categorize the topics of an article or newspaper;
- ability to measure the similarity between different articles.

E. Suggestion Engine

The suggestion engine is a meaningful content discovery tool for users. One way to suggest new content to users is to display related articles alongside what they are currently viewing. The suggestion engine must support the following features:

- ability to draw conclusions on the interests of a reader given their activities and relationships on the social network;
- ability to draw conclusions on the interests of a reader based on the set of articles they have read;

- ability to provide relevant article suggestions based upon the conclusions reached about the user's interests.

F. *Content Layout System*

The layout system will provide content producers with an experience similar to that of a desktop editor application. Page design should be done in-browser with tools that the editor will find familiar. The layout system must support the following features:

- drag and drop content into place;
- adjust style of content;
- adjust layout of content including but not limited to adding, removing and adjusting columns; and adjusting size and position of areas for content;
- save and reload layouts or layout elements;
- aid positioning using grids or guidelines;
- save and reload default layouts for a publication.

G. *User Interface*

The user interface should render the content for the reader the way the content producer envisioned it to be. Navigation through content should be non-obtrusive. The user interface must support the following features:

- overall styling during reading experience set by content producer;
- user account management including but not limited to profile information, authentication, password recovery;
- content production including but not limited to viewing, adding, modifying, and removing content, issues, and layouts;
- unobtrusive navigation while immersed in the reading experience;
- options during the reading experience to comment on and/or share the article to the internal social networking, by email, and/or to the currently ubiquitous social networking sites.

H. *Business Logic*

User access to content should be controlled in order to differentiate between a user who may edit content of a given article or publication and a user who may only view the content. Additionally, in the case of paid subscriptions to publications, access control needs to differentiate between users who may or may not read certain articles. The business logics must support the following features:

- ability to authenticate users using unique login and passwords;
- ability to enforce access control of user's data based upon the user's privileges.

I. *Non-Functional Requirements*

The system's non-functional requirements are consistent with those of other web-based systems. They are as follow:

- the site must be accessible with all major web browsers, namely Internet Explorer, Firefox, Chrome, and Safari;
- system-generated data must be kept to a minimum and encoded so as to minimize the amount of bandwidth used;
- the user interfaces must be easy to understand and use, both for readers, producers, and administrators;
- the system must be quick to respond and error-free.

V. System Architecture

A. *Logical Architecture*

Figure 1 illustrates the logical connections between the components of our proposed system. The user interface is presented to the user through a browser, and it directly connects to the business and cloud logics module which will use the Suggestion Engine and Layout Engine as needed. It is also connected to the Social Networking module, which provides the social networking functionalities. The data is stored in a Database, which the Natural Language Processor periodically queries to analyze all available articles. The Social Networking module will maintain a Social Graph of all user relationships and interactions. Meanwhile, the Cloud Logics component will monitor the system's overall performance and interact with the Cloud Hypervisor as needed to adjust the system's physical structure to respond to levels of user demand.

B. *Physical Architecture*

All the end users of our system, be them readers or producers, will connect to the site via a web browser running on any device or platform. The browser connects to the cloud's software load balancer, which is hosted on a virtual machine. The load balancer forwards requests to corresponding identical web servers (also hosted as virtual machines) to service each request, and may load share amongst themselves. Furthermore, the web servers connect to a set of databases that will replicate between each other for both fail safe redundancy and throughput. The Web Servers also connect to the social network. This setup is illustrated in Figure 2.

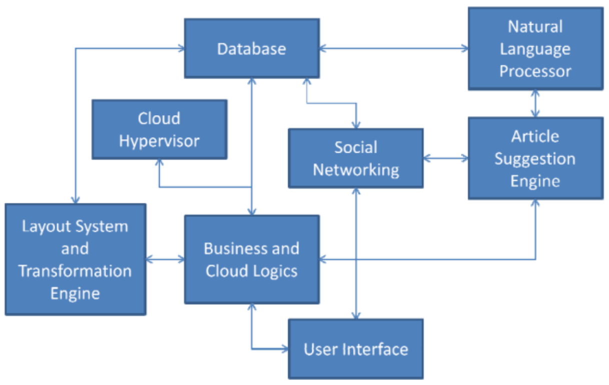

Database
Natural Language Processor
Cloud Hypervisor
Social Networking
Article Suggestion Engine
Layout System and Transformation Engine
Business and Cloud Logics
User Interface

Figure 1. Logical architecture of the system.

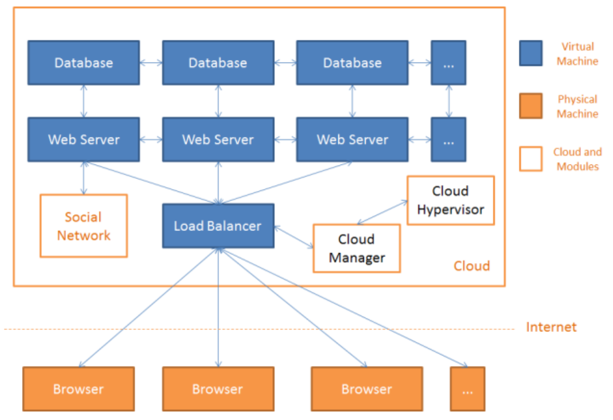


Figure 2. Physical architecture of the system.

C. *Architecture Details*

The software architecture for the client-side Web UI software consists mainly of JavaScript model classes used to represent the locations of elements on the page. Objects of these classes are saved to the database through requests to the web server or instantiated from saved object states retrieved from the web server.

The software architecture for the server-side software consists mainly of controllers and models. Ruby on Rails uses a Model, View, Controller architecture. Controllers use instances of Models and Views to render a page for the user. Requests are mapped to member functions of controllers based mainly on the request URI and HTTP method.

The natural language processor is implemented as a client-server system as well. The NLP server is responsible for the language processing functionalities of the system. It uses TCP functionalities to receive communications from the NLP client. It also interfaces directly with the database to fetch information independently of the rest of the system.

The cloud is composed of several interconnected sub-components. There is a Cloud Controller, which is responsible for real time monitoring and workload management functions, dynamic cloud reconfiguration and server load balancing, and cloud hypervisor operations. This is the central component of the cloud, which manages the other servers and optimizes the entire system. To illustrate its function, its state chart is presented in Figure 3. The Controller accepts incoming TCP connections from the Load Balancer, and connects to each of the Server System Monitors that reside in the servers under its control via TCP as well. The Server System Monitors are a simple monitoring subcomponent, which periodically gathers system performance information and forwards it to the controller through a TCP connection. Both the Controller and the Monitors are written in C++ as Linux applications. The Load Balancer, by contrast, is implemented in PHP as a web application at the forefront of the Cloud. Its function is to accept incoming HTTP requests from users and forward them to designated web servers, and balance the workload so no single server is over- or under-utilized. It also forwards the responses from the servers back to the users. The Load Balancer communicates with the cloud over TCP to obtain the information for the web servers.

VI. Prototype Implementation and Testing

A working prototype of the entire system was implemented and run on a VMware 5.0 ESXi physical server, which contains a quad core CPU (2GHz per core) and 3 GB of RAM. Each of the virtual machines within runs a Ubuntu 11.10 Server operating system (32bit) with virtual hardware settings of 1 CPU Core, whereby VMware performs time-sharing between the 4 physical CPU cores, and 5 GB of HDD space. The setup also include 384 MB RAM for web and load balancer servers and 512 MB RAM for MySQL database and NLP servers, 1024 MB RAM for the Cloud Controller server. The server is connected to the internet using a router with the ability to set static IP addresses and port forwarding. A screenshot of the content layout interface is given in Figure 4. While the interface is simple, it gives content producers complete control to add, move and edit text, headings and multimedia items.

A set of test cases was developed and ran to verify the functionality of critical features of the system. The cloud controller was tested for its server manipulation features: the ability to create new servers, to configure them correctly, to reconnect to them to check on them, and to delete them when they are no longer needed. Each of these operations also tested the cloud controller's ability to update the network's topology. With these components validated, we then proceeded to test higher-level functionalities, such as the controller's ability to gather usage statistics from the servers, to balance the servers'

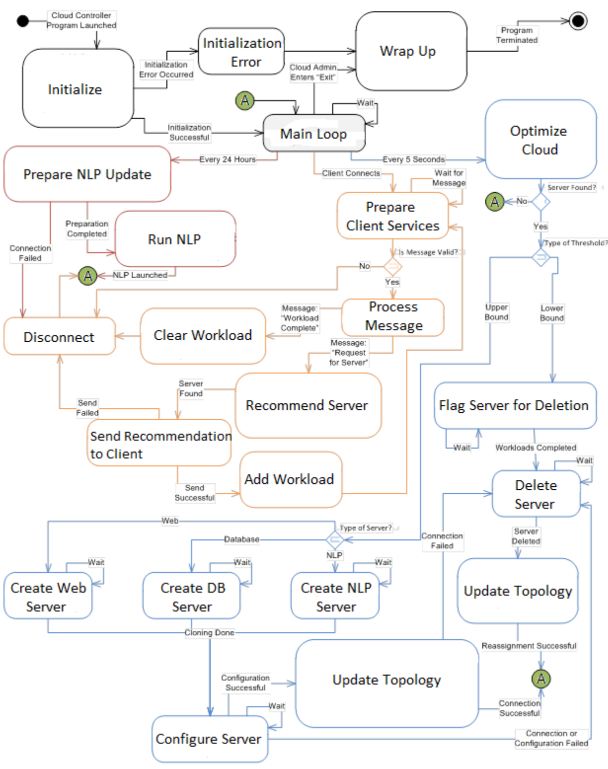
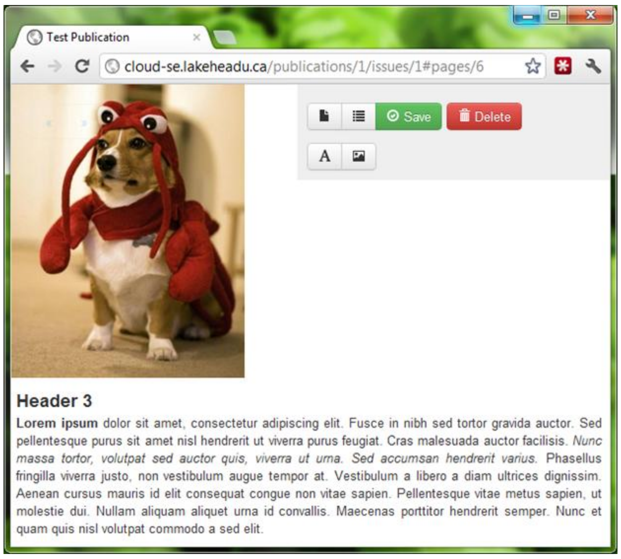


Figure 3. State chart of the cloud controller program.

workloads and optimize the topology, and to forward request and responses. The user interface and database were tested by registering both regular user accounts and publication editor accounts, and executing the legal and illegal functions of both classes of users. The editor could create new publications and new articles inside the publications, and edit these articles using the layout editor shown in Figure 4. The regular user could browse the publications, subscribe to those desired, and see the articles displayed in exactly the way the editor had laid them out. Finally, the natural language processing functionalities of the system (along with that section of the database) were tested by uploading a set of 10 news articles into the system and then having the processor parse them, build word vectors, and compare these to a set of predefined class vectors to classify the articles into their correct topics.

The last test we ran was a workload test, designed to verify the robustness and reliability of our cloud controller and architecture. For this test, we used a set of other computers to send HTTP requests. We used requests for the website's home page as well as multi-page requests. Given our system's dual-core hardware, we tested setups with one and two web servers. In each case, we measure the system's throughput, CPU usage, disk usage and memory usage. In both setups, we found similar disk, memory, and CPU usage. The throughput was different however, with the setup with two web servers consistently performing better than the setup with only one web server. On requests for the home page, the two-server setup yielded an average throughput of 258 kB/s against 182 kB/s for the setup with one web server. And for multi-page requests, which required database queries and more processing, the throughput of the two-server setup was 142 kB/s against 102 kB/s for the single-server setup. The 40% improvement observed in throughput demonstrates that our system can scale up quite efficiently. To further illustrate, the response times of both systems during our test are shown graphically in Figure 5. In that figure, the higher line is the response time of the single-server setup, the lower line is the response time of the two-server setup, and the individual points are HTTP requests. We can see that throughout the experiments, the single-server setup consistently requires more time to respond to the requests compared to the two-server setup. While these tests were conducted with static topologies, we expect the response times to "jump" from the single-server setup line to the double-server setup when the system adds a web server dynamically, and the throughput to increase in a similar fashion. Additional jumps are expected as the system adds additional servers.



Header 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce in nibh sed tortor gravida auctor. Sed pellentesque purus sit amet nisl hendrerit ut viverra purus feugiat. Cras malesuada auctor facilisis. *Nunc massa tortor, volutpat sed auctor quis, viverra ut urna. Sed accumsan hendrerit varius.* Phasellus fringilla viverra justo, non vestibulum augue tempor at. Vestibulum a libero a diam ultrices dignissim. Aenean cursus mauris id elit consequat congue non vitae sapien. Pellentesque vitae metus sapien, ut molestie dui. Nullam aliquam aliquet urna id convallis. Maecenas porttitor hendrerit semper. Nunc et quam quis nisl volutpat commodo a sed elit.

Figure 5. The prototype's content layout editor.

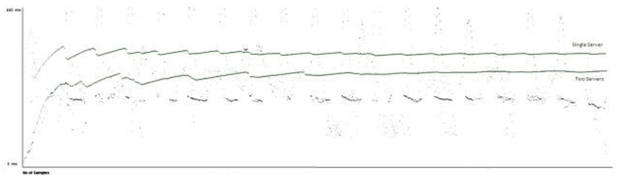


Figure 4. Response times of the two setups tested.

VII. Conclusion

Building a Web 2.0 social networking site is a very ambitious project. In this paper, we developed a web application with smart horizontal scaling using a cloud-based architecture, incorporating modern aspects of web technology as well as elements of natural language processing to help readers discover content and help publishers get discovered. The size and scope of this project make it a challenge for any developer. This paper aims to be a roadmap, to help others duplicate or improve upon our architecture.

While the system is completely functional in the state described in this paper, there is room to further develop and improve each one of its components. There is a wide range of NLP and recommendation algorithms in the literature, some of which could be adopted to improve the natural language processor and the suggestion engine respectively. New editing tools can be added to the content layout system to give more control to content producers. The design of a better user interface is an open challenge, not just in our system but in the entire software world. Gathering real workload usage will allow us to fine-tune the cloud's load balancing algorithms. And finally, the social networking component of the system could be both simplified and enhanced by linking our system to an existing social network such as Facebook, Google+, or Twitter. Each new feature and improvement we make in each component will of course require additional testing. And once a more complete and polished version of the site is ready, it should be deployed and used in practice, to gather both real-world usage information and user feedback that will help guide the next iteration of the system.

References

[1] Kristen Purcell, Lee Rainie, Amy Mitchell, Tom Rosenstiel, Kenny Olmstead, "Understanding the participatory news consumer: How internet and cell phone users have turned news into a social experience", Pew Research Center, March 2010. Available:

http://www.pewinternet.org/Reports/2010/Online-News.aspx?r=1, accessed April 2012.
[2] Xiangying Dai, Yunlian Sun, “Event identification within news topics”, International Conference on Intelligent Computing and Integrated Systems (ICISS), October 2010, pp. 498-502.
[3] Nam P. Nguyen, Thang N. Dinh, Ying Xuan, My T. Thai, “Adaptive algorithms for detecting community structure in dynamic social networks”, Proceedings of the IEEE INFOCOM, 2011, pp. 2282-2290.
[4] Chen Wei, Simon Fong, “Social Network Collaborative Filtering Framework and online Trust Factors: a Case Study on Facebook”, 5th International Conference on Digital Information Management, 2010.
[5] Yao-Jen Chang, Hung-Huan Liu, Li-Der Chou, Yen-Wen Chen, Haw-Yun Shin, “A General Architecture of Mobile Social Network Services”, International Conference on Convergence Information Technology, November 2007, pp. 151-156.
[6] Jiamei Tang, Sangwook Kim, “Theme-Based Mobile Social Network System”, IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC), 2011, pp. 1089-1095.
[7] Rabih Dagher, Cristian Gadea, Bogdan Ionescu, Dan Ionescu, Robin Tropper, “A SIP Based P2P Architecture for Social Networking Multimedia”, 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications, 2008, pp. 187-193.
[8] Cristian Gadea, Bogdan Solomon, Bogdan Ionescu, Dan Ionescu, “A Collaborative Cloud-Based Multimedia Sharing Platform for Social Networking Environments”, Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN), 2011, pp. 1-6.
[9] Wael M.S. Yafooz, Siti Z.Z. Abidin, Nasiroh Omar, “Challenges and issues on online news management”, IEEE International Conference on Control System, Computing and Engineering (ICCSCE), 2011, pp. 482-487.
[10] Paolo Massa, Paolo Avesani, “Trust-aware recommender systems”, Proceedings of the 2007 ACM conference on Recommender systems (RecSys '07), October 2007, Minneapolis, USA, pp.17-24.
[11] Trieu C. Chieu, Ajay Mohindra, Alexei A. Karve, “Scalability and Performance of Web Applications in a Compute Cloud”, *IEEE 8th International Conference on e-Business Engineering (ICEBE),* Oct. 2011, pp. 317-323.
[12] Prasad Calyam, Munkundan Sridharan, Yingxiao Xu, Kunpeng Zhu, Alex Berryman, Rohit Patali, Aishwarya Venkataraman, “Enabling performance intelligence for application adaptation in the Future Internet”, *Journal of Communications and Networks*, vol. 13, no. 6, Dec. 2011, pp.591-601.
[13] Jerry Gao, Pushkala Pattabhiraman, Xiaoying Bai W. T. Tsai, “SaaS performance and scalability evaluation in clouds”, *2011 IEEE 6th International Symposium on Service Oriented System Engineering (SOSE),* Dec. 2011, pp. 61-71.
[14] Niclas Snellman, Adnan Ashraf, Ivan Porres, “Towards Automatic Performance and Scalability Testing of Rich Internet Applications in the Cloud”, *37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA),* Aug. -Sept. 2011, pp. 161-169.
[15] Bhanu P Tholeti, “Hypervisors, virtualization, and the cloud: Learn about hypervisors, system virtualization, and how it works in a cloud environment”, IBM developerWorks, September 2011. Available: http://www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare, accessed April 2012.