Web 2.0-based SaaS for Community Resource Sharing

I-Ching Hsu

Abstract

Cloud computing is an emerging paradigm where computing resources are offered over the Internet. Web 2.0 is also the mainstream technology in the existing Internet. How to combine the two domains is the main purpose of this paper. Software as a Service (SaaS) is one of the important service models of cloud computing. This study presents a Web 2.0-based Software as a Service (Web 2.0-based SaaS) model based on Web 2.0 Mashups for integrating Web 2.0 technologies into SaaS. Based on the proposed Web-based SaaS model, a RSS-based Community Resource Sharing System, called RCRSS, is developed, and then is deployed in the Google App Engine platform. To illustrate the RCRSS applications, we demonstrate resource sharing of community can be acquired from heterogeneous and distributed resources, including Blog, Facebook, Social Bookmark, and Web Multimedia.

Keywords: Cloud Computing, SaaS, Web 2.0 Mashup, RSS

1. Introduction

Cloud computing has become one of the most promising IT solutions and business trends in recent years. The first to introduce the term cloud computing was Google's CEO Eric Schmidt [1]. The term refers to the important and long-term trend in computing over the Internet. Many institutions and companies provide definitions and solutions for cloud computing [2-6]. However, there is still no widely accepted definition of cloud computing. From our points of view, cloud computing is a conceptual architecture providing a set of distributed computing resources delivered over the Internet. It provides a service oriented architecture to developers for facilitating the rapid building information systems. There are three types of cloud service models that can be provided by the cloud computing: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). SaaS provides services to end users, while IaaS and PaaS provide services to cloud application developers.

As well known, Web 2.0 is recognized as the next generation of web applications proposed by T. O'Reilly [7]. The main feature of Web 2.0 applications is that they provide a medium for the sharing and exchange of resources [8-10]. These resources can be considered as a kind of cloud resource, such as Web feed and Web API, allow SaaS developers to take advantage of these resources to enrich their own applications or produce new integrated solutions by integrating resources, which they could not have provided on their own. This paper intends to advance in the direction. We present a Web 2.0-based Software as a Service (Web 2.0-based SaaS) model based on Web 2.0 Mashups for integrating Web 2.0 technologies into SaaS.

One major feature of Web 2.0 is to adopt Web feeds to build a more maintainable and cooperative Web. A Web feed contains a structured information source which is written in XML to facilitate the machine-readable. This means that Web feeds can be used to automatically transfer resource sharing information from one system to another, without any human intervention. Web feeds allow community members to publish frequently updated information such as work log, social bookmark, task presentation, report, and action items. RSS is currently the main format of Web feed.

Based on the proposed Web 2.0-based SaaS model, a RSS-based Community Resource Sharing System, called RCRSS, is developed, and then is deployed in the Google App Engine platform. The RCRSS is a Web 2.0-based community cloud that provides a cloud resource sharing service broker for a specific Web community, such as research, school, or institute project team. To illustrate the RCRSS applications, we demonstrate knowledge sharing of community can be acquired from heterogeneous and distributed knowledge resources, including Blog, Facebook, Social Bookmark, and Web multimedia. Furthermore, the proposed RCRSS provides sharing information on two different schemes,
including push-based scheme and pull-based scheme. The pull-based scheme allows RCRSS to pull the sharing information from the heterogeneous and distributed knowledge resources. The push-based scheme allows RCRSS to push the interested sharing information to project community member’s portal in real-time.

This paper is organized as follows. The next section briefly reviews some existing cloud computing applications. Section 3 discusses how Web 2.0 Mashups can be used in cloud computing. Section 4 describes RSS-based metadata for community resource sharing. In Section 5, we implemented the RCRSS to demonstrate the resource sharing application for a community. Section 6 presents a preliminary experimental study. Finally, summary and concluding remarks are included.

2. Related Works

Cloud computing is an emerging paradigm where computing resources are offered over the Internet. There are three types of cloud service models that can be provided by the cloud computing: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In [11], authors introduce new services in Cloud Computing environment. Cloud computing has several characteristics. First, it provides a service oriented architecture to help developers rapidly build distributed systems. Second, it supports consumers to access cloud resources on demand from another cloud application over the Internet. Third, it represents a new paradigm for deploying, managing and offering services through a shared infrastructure.

Infrastructure as a Service (IaaS) provides the hardware and administrative services needed to store cloud applications and a physical platform for running cloud applications. These infrastructure resources, including computer server, storage system, network equipment, and data center, offer basic storage and computing capabilities as standardized services over the Internet. The cloud developers would typically deploy their applications on the infrastructure. Typical examples are Amazon EC2 (Elastic Cloud Computing) Service [12] and S3 (Simple Storage Service) [13] where compute and storage infrastructures are available to public access as a utility.

Platform as a Service (PaaS) provides an integrated environment or a middleware for cloud developers to implement and deploy cloud applications without bothering the cost and purchase of hardware or software. It is usually a set of development tools and services that are run on servers in the cloud. Practical examples include Google App Engine [14], Microsoft Windows Azure platform [15], Salesforce Force.com [16], and Aneka [17]. Google App Engine provides a python or java runtime environment and APIs for developers to build Web applications on the same scalable systems that power Google applications. Another example is Microsoft Windows Azure platform that provides an operating system and a set of developer services that can be used individually to build new applications to run on the cloud.

Software as a Service (SaaS) provides on-demand access to web-based applications that are maintained centrally by a provider. It is a software delivery model that allows an application is hosted as a service and provided to customers across the Internet. They can deploy their applications on existing clouds without installing and running the application on the customer’s own computer. Popular examples of SaaS include Google Mail, Google Docs, Google Blogger, Microsoft Hotmail, and so forth.

Social Web 2.0 technologies enable the development of cloud computing applications easy to integrate the various Web standards to facilitate the sharing and exchange of cloud resources. Therefore, Web 2.0 provides a new opportunity to share cloud resources among community members. Web-based Blog is a popular Web 2.0 application and is a personal online publishing platform, allowing project community members to quickly and simply write, publish and share their work experience via the internet. Blogs offer the community members, including storage service, easy access to online publishing of information and experience as no technical knowledge is required, which can then play as a platform for discussion as community members can write comments. Currently, many social websites including Windows Live Spaces, Google Blogger, Facebook, and My Yahoo! Blog provide Blog service as their most important service.

From the view of service boundary, there are four different deployment models of cloud computing: public cloud, private cloud, hybrid cloud, and community cloud [18]. Public cloud is a service provider
to make cloud resources, such as computing power, storage capacity, network management, service quantity, available to any customer via the Internet. Private cloud is also known as internal cloud, which is implemented to provide hosted services to a specific group of people behind a firewall. Hybrid cloud is a combination of a public and private cloud that interoperates and shares cloud resources by a secure network. Community cloud [19] is controlled and used by a specific community that has common interests or missions. The members of the community share access to the data and applications in the community cloud.

3. Web 2.0-based Software as a Service Model

The cloud computing framework is composed of three layers: hardware layer (i.e. IaaS), platform layer (i.e. PaaS), and application layer (i.e. SaaS). The Web 2.0-based SaaS is located in the application layer, as shown in figure 1. In the platform layer, this study adopts Google App Engine. The layers IaaS, PaaS, and SaaS represent current service models of cloud computing. The top layer is SaaS provides services to cloud users, while IaaS and PaaS provide services to PaaS and SaaS, respectively.

Google App Engine is a PaaS for developing web applications that are deployed on the data centers of Google. It provides Web APIs to retrieve the various services, including data store, Google Accounts, Google Map, Google Calendar and Google email. Google App Engine also supports a web-based administration console for the SaaS developers to easily build, maintain, and extend their web applications. Virtualization is one of key technology in Cloud computing. There are many levels of virtualization such as platform level, operating system level, network level, and hardware level. The proposed RCRSS is a cloud platform level of virtualization. Standing cloud computing virtualization features, each module can be a separate cloud service.

The core components of Web 2.0-based SaaS include Web Feed and Web API. Web Feed is a typical data resource, while Web API is a typical service resource. Major Internet companies, such as Google, Microsoft, Yahoo, Amazon, and eBay, have published APIs based on web standards that allow Web 2.0-based SaaS developers to access their services and data. The Web 2.0-based SaaS adopts Web feeds to support to publish dynamic information, including Blog, social bookmark, task presentation, report, and Web multimedia. The existing websites, such as Blog, Facebook, Twitter, YouTube, Delicious, and SlideShare, provide RSS as Web feed to deliver dynamic information. The Web 2.0-based SaaS developer adopts Web 2.0 Mashups to integrate Web Feed and Web API to create Web 2.0-based SaaS system on local computers. The source code of Web 2.0-based SaaS system can be uploaded to Google App Engine through HTTP proxy. End users can access the Web 2.0-based SaaS system with various clients, such as browser, RSS reader, and mobile phone.
4. RSS-based Metadata for Community Resource Sharing

RSS and Atom are currently the two main formats of Web feed. RSS is a family of Web feed formats specified in XML standard. There are three different version of RSS, namely Rich Site Summary, RDF Site Summary and Really Simple Syndication. The Really Simple Syndication (RSS 2.0) is the most widely used. Unlike RSS, Atom is proposed RFC 4287 and is defined with XML schema. This study adopts RSS 2.0 to present the metadata of knowledge and experience sharing. The detail comparisons of RSS and Atom are shown in Table 1.

Table 1. The comparison of RSS and Atom

<table>
<thead>
<tr>
<th></th>
<th>RSS</th>
<th>Atom</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Different names for different versions</td>
<td>Atom 1.0</td>
</tr>
<tr>
<td></td>
<td>RDF Site Summary (RSS 0.9 and 1.0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rich Site Summary (RSS 0.91, RSS 1.0)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Really Simple Syndication (RSS 2.0)</td>
<td></td>
</tr>
<tr>
<td>publisher</td>
<td>Harvard University</td>
<td>IETF community</td>
</tr>
<tr>
<td>namespaces</td>
<td>no namespaces</td>
<td>RFC 4287</td>
</tr>
<tr>
<td>schema</td>
<td>no XML schema</td>
<td><a href="http://www.w3.org/2005/Atom">http://www.w3.org/2005/Atom</a></td>
</tr>
<tr>
<td>XML standard</td>
<td>Yes, support Well-formed XML document</td>
<td>Yes, support Validating XML document</td>
</tr>
<tr>
<td>mime type</td>
<td>no mime type</td>
<td>application:atom+xml</td>
</tr>
</tbody>
</table>

Using RSS feeds to present the metadata of knowledge sharing have a few advantages. (1) Community members can be notified of new knowledge sharing information without needing to visit the websites. (2) RSS feeds provide metadata of knowledge sharing information together with hyperlinks to the full versions of the Web-based information content. (3) RSS feeds allow community members to pull the knowledge sharing information they are interested in rather than data being pushed to the team members.

This study develops a RSS-based Community Resource Sharing System (RCRSS) which is based on RSS feeds to support community members to publish heuristic documents, including work log, social bookmark, task presentation, report, and action items. Knowledge sharing information of community can be acquired from heterogeneous and distributed Web 2.0 application platforms, including Blog, Social Bookmark, and Web Multimedia. The Web 2.0 application platforms adopted by RCRSS are described in Table 2.

Table 2. The Web 2.0 platforms employed in RCRSS

<table>
<thead>
<tr>
<th>Web 2.0 platform</th>
<th>Web Site</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Blog</td>
<td>Blogger</td>
<td>Web-based Blogs are a personal online publishing tool, allowing community members to quickly and simply write, publish and share their own work experience via the internet. Blogs offer the community members, including storage service, easy access to online publishing of information and experience as no technical knowledge is required, which can then play as a platform for discussion as team members can write comments.</td>
</tr>
<tr>
<td>Social Bookmark</td>
<td>Delicious</td>
<td>A social bookmarking website can provide the community members to store and share their bookmarks on the web instead of inside their browser. Bookmarks are annotated by RSS feed which are searchable keywords assigned by community members.</td>
</tr>
<tr>
<td>Web Multimedia</td>
<td>SlideShare</td>
<td>The Web multimedia is widely used in web environment, such as PowerPoint, picture, video, audio. Slideshare is a great Web 2.0 platform that let community members upload their presentations and share them online via a Web interface.</td>
</tr>
</tbody>
</table>
5. RSS-based Community Resource Sharing System

The RSS-based Community Resource Sharing System (RCRSS) serves as a resource sharing service broker for the heterogeneous and distributed knowledge resources, such as Blogger, Facebook, Delicious bookmark, and SlideShare. It uses RSS feeds to facilitate resources reusability. The main modules of our developed RCRSS include: Member Query Module, RSS Crawler Module, and Database Module. In the RCRSS, knowledge sharing information is described with RSS feeds and collected through the RSS Crawler Module, transferred and stored into the Database Module.

**RSS Crawler Module** consists of Crawler Engine and RSS Parser. The Crawler Engine is a program which collects the RSS feeds from distributed Web 2.0 application websites, including Google Blogger, Delicious bookmark, and SlideShare. The RSS Parser reads the RSS feed and converts it into Resource Database.

**Member Query Module** comprises Manager Interface and Resource Filter. The Manager Interface listens to the member’s request and interacts with other components of the RCRSS. The Resource Filter receives and filters the information from the Manager Interface and Resource Database.

**Database Module** is composed of Resource Datastore and Member Datastore. The profile of project community member is stored in Member Datastore. The RSS feed of community resource is stored in Resource Datastore.

![Figure 2. The flow-oriented RCRSS architecture](image)

The following steps explain the message flow illustrated in Figure 2:

1. The project community members can create project documents for sharing at any time, from anywhere on the Web. The metadata of a project document is described in a RSS feed, shown as Figure 3.
   1.1 Blogger provides community members to write work log for knowledge and experience sharing, including work report, problem statement, problem solution, Q&A interaction, and information announcement.
   1.2 Delicious bookmark offers community members to store, annotate, search and manage bookmarks of important websites that they might want to retrieve or share with others.
   1.3 SlideShare allows community members to upload, tag, embed, comment on, and share...
Web 2.0-based SaaS for Community Resource Sharing
I-Ching Hsu

PowerPoint, OpenOffice, Keynote, and PDF presentations.

2. The Manager Interface is a tool designed to help community members create and maintain their profiles. The following steps are to register:
2.1. Manager Interface provides an interface to members to input personal information (as shown in Figure 4 [A]), including topics of interest, URL of Blogger, Delicious, and SlideShare.
2.2 The Manager Interface saves the personal information into the Member Datastore.

3. The step is a pull-based interaction scheme that accomplishes the following tasks:
3.1 The Crawler Engine queries Member Datastore to retrieve member profiles.
3.2 The Crawler Engine parses the member profile to extract the URL of sharing resources, including Blogger, Delicious bookmark, and SlideShare, and then pulls the RSS feeds based on those URLs.
3.3 The Crawler Engine sends the RSS feeds to RSS Parser.
3.4 RSS Parser parses the RSS feeds to filter the available information, and then saves into the Resource Datastore.

4. The step is a push-based approach to query database that accomplishes the following tasks:
4.1 Member logins via the Manager Interface, and then set the query condition (as shown in Figure 4 [B]).
4.2 The Manager Interface invokes Resource Filter with member ID.
4.3 The Resource Filter query Member Datastore to retrieve member’s topics of interest.
4.4 The Resource Filter query Resource Datastore with member’s topics of interest to acquire the sharing resources, including Blogger, Delicious bookmark, and SlideShare.
4.5 The Resource Filter responses the Web feeds to Manager Interface.
4.6 The Manager Interface converts the Web feeds into an HTML page to response to community member (as shown in Figure 4 [C]).

<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0">
  <channel>
    <title>Work Log - Project#21312</title>
    <link>http://140.130.35.139/?source=rss</link>
    <description>Work Log - Project Community member: 231</description>
    <language>English</language>
    <pubDate>Sat, 01 May 2010 11:12:24 GMT</pubDate>
    <copyright>Copyright 1995 - 2010 China Times Inc.</copyright>
    <category>work log</category>
    <generator>Community #21312-21-89</generator>
    <ttl>30</ttl>
  </channel>
  <item>
    <title>Member: Allan Wang - Work Log</title>
    <link>http://140.130.35.139/2009HTC/20091015.jsp</link>
    <pubDate>Sat, 01 May 2010 11:12:54 GMT</pubDate>
    <guid>JTO:YTU879:PQ:221</guid>
    <subject>Web services application in CALS</subject>
    <description><![CDATA[The Work Log -
      <br>standard: SOAP<br><b>32</b><br><br>time: <b>Sat, 01 May 2010 12:10:22 GMT</b><br><br>Server Platform: <b>ASUS TS300E5</b><br><br>Project Team: <b>CALS</b>]]></description>
  </item>...
</rss>

Figure 3. Partial code of RSS feed
This study implements a RSS-based Community Resource Sharing Platform (RCRSS) that is a Web 2.0-based SaaS application to provide a resource sharing service broker for a specific Web community, such as project team. The member queries RCRSS with interest topics to acquire the sharing resources, as shown in Figure 4.

Figure 4. The query operation based on personal information in the RCRSS

6. Preliminary Experiment

The proposed RCRSS has been implemented and described. This section presents a preliminary experiment to evaluate the performance of the RCRSS. This study uses three indexes, Precision, Recall and F1 Measure, to evaluate the performance of the proposed RCRSS. They are formally defined as follows:
NRWI = the number of relevant Web feed items retrieved by RCRSS
TNRW = the total number of existing relevant Web feed items in the dataset
TNWI = total number of Web feed items retrieved by a query
Precision = NRWI / TNWI
Recall = NRWI / TNRW
F1 measure = (2 * Precision * Recall) / (Precision + Recall)

The test dataset contained 1200 Web feed items. Each Web feed item contains at least one category element to refer to a specific keyword. The content of category element is regarded as a keyword of the Web feed item. The experiment is performed by 5 volunteers having different preferences, shown as in Table 3. This experiment has implemented based on keyword matching. The test results are summarized in Table 4. Figure 5 illustrate the results in diagram.

Table 3. The interest topics of users

<table>
<thead>
<tr>
<th>User ID</th>
<th>Interest Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0122101</td>
<td>Java, C++, C#</td>
</tr>
<tr>
<td>W8921012</td>
<td>Report</td>
</tr>
<tr>
<td>P1201121</td>
<td>XML, HTML</td>
</tr>
<tr>
<td>R5421892</td>
<td>Javascript</td>
</tr>
<tr>
<td>T3201128</td>
<td>JSP, PHP, ASP</td>
</tr>
</tbody>
</table>

Table 4. Test results

<table>
<thead>
<tr>
<th>User ID</th>
<th>Keyword Matching</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1 Measure</td>
</tr>
<tr>
<td>F0122101</td>
<td>0.544</td>
<td>0.516</td>
<td>0.53</td>
</tr>
<tr>
<td>W8921012</td>
<td>0.638</td>
<td>0.662</td>
<td>0.65</td>
</tr>
<tr>
<td>P1201121</td>
<td>0.589</td>
<td>0.544</td>
<td>0.566</td>
</tr>
<tr>
<td>R5421892</td>
<td>0.636</td>
<td>0.609</td>
<td>0.622</td>
</tr>
<tr>
<td>T3201128</td>
<td>0.674</td>
<td>0.633</td>
<td>0.653</td>
</tr>
</tbody>
</table>

Figure 5. The test results compared with different users

The experimental results are reported in Figure 5. It can be observed that the Precision has better performance than the Recall. However, Precision value and Recall value are still low. This indicates that the Web feed personalized by keyword matching are much relevant because these Web feed items have similar concepts. This shows that using keyword matching to improve the efficiency of Web feeds is limit. In future studies, we will use ontology-based reasoning to improve the limitation of keyword matching.
7. Conclusion

Cloud Computing has emerged as a new paradigm to facilitate for deploying, managing and offering services over the Internet. It provides the advantages of low cost, high performance, and scalability. However, there is still no widely accepted definition and uniform standards for cloud computing. This study has proposed a Web 2.0-based Software as a Service model, called Web 2.0-based SaaS. The Web 2.0-based SaaS model adopts Web 2.0 Mashups to integrating Web 2.0 technologies, such as Web Feed and Web API, into SaaS of cloud computing. To demonstrate the feasibility of Web 2.0-based SaaS model, a RSS-based Community Resource Sharing System (RCRSS), is implemented to provide community members to retrieve the heterogeneous and distributed resources. The RCRSS is composed of Member Query Module, RSS Clawer Module, and Database Module. The main vision of our RCRSS is to support a community in which members with similar work backgrounds use Web 2.0 application platforms for resource sharing.

One future work is to investigate how to integrate Semantic Web [20-22] technologies into RCRSS to facilitate the implementation of intelligent resource sharing application. This approach supports ontology-based reasoning for finding dynamic correlations of a certain web feed. Such dynamic customized Web feed is desirable for personalized needs. Firstly, it is customized for each individual user, based on what metadata and ontology the user profile has shown so far. Secondly, because the content or category of a user profile may keep changing, dynamic customization provides more up-to-date web feeds than a static design. It can be expected that the Precision, Recall and F1 Measure of ontology-based reasoning have better performance than the keyword matching.

8. References


