Cloud-dew architecture

Yingwei Wang

Department of Computer Science and Information Technology,
University of Prince Edward Island,
550 University Avenue, Charlottetown,
Prince Edward Island C1A 4P3, Canada
Email: ywang@upei.ca

Abstract: Users derive many benefits by storing personal data in cloud computing services; however the drawback of storing data in these services is that the user cannot access his/her own data when an internet connection is not available. To solve this problem in an efficient and elegant way, we propose the cloud-dew architecture. Cloud-dew architecture is an extension of the client-server architecture. In the extension, servers are further classified into cloud servers and dew servers. The dew servers are web servers that reside on user's local computers and have a pluggable structure so that scripts and databases of websites can be installed easily. The cloud-dew architecture not only makes the personal data stored in the cloud continuously accessible by the user, but also enables a new application: web-surfing without an internet connection. An experimental system is presented to demonstrate the ideas of the cloud-dew architecture.

Keywords: cloud-dew architecture; cloud computing; client-server architecture; software architecture; internet connection; local domain name system; LDNS; cloud server; dew server; dewsite; dew server structure.

Reference to this paper should be made as follows: Wang, Y. (2015) 'Cloud-dew architecture', *Int. J. Cloud Computing*, Vol. 4, No. 3, pp.199–210.

Biographical notes: Yingwei Wang is a faculty member in the Department of Computer Science and Information Technology at the University of Prince Edward Island, Canada. His research interests include cloud computing and bioinformatics. He obtained his Bachelor's and Master's degree at Harbin Institute of Technology, China and his PhD degree at the University of Waterloo, Canada.

1 Introduction

The cloud computing paradigm (Sasikala, 2011; Rimal et al., 2009) is quickly accepted as an alternative to traditional information technology (IT) because of the significant benefits, such as on-demand resources and low maintenance costs. The architecture of cloud computing system has been discussed by many researchers (Goscinski and Brock, 2011; Rochwerger, 2009; Tsai et al., 2010; Xiong et al., 2011; Zhang and Zhou, 2009). In this paper, we discuss cloud computing from one special aspect: the accessibility of user data.

In cloud computing environment, users and organisations store their data in the cloud. If a user stores all of his/her data in the cloud, his/her own computer is merely used to access the cloud. The advantageous feature of this arrangement is data mobility: user data can be accessed from anywhere as long as an internet connection is available. The problem of this arrangement is that the user relies heavily on an internet connection and the servers. If any problem happens with the servers or an internet connection is not available, the user cannot access his/her data. According to a survey, 84% of the consumers are concerned about their data storage location (Habib et al., 2012), demonstrating that data accessibility is an important issue to users.

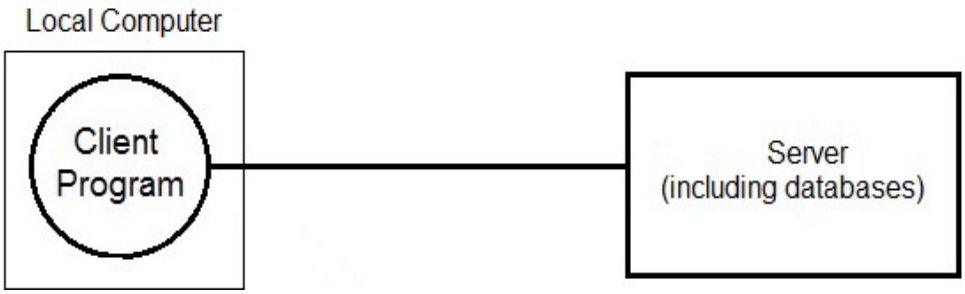
One apparent solution is to keep a copy of user data in a user's local computer. Two problems are associated with this solution: first, it is not easy to keep the data on the local computer always consistent with the data in the cloud. Second, the user may have to remember both local and cloud locations of his/her data. When the user's data is complex in nature and numerous, these tasks are not trivial. In this paper, we propose a better solution.

2 Cloud-dew architecture

2.1 Client-server architecture

The client-server architecture (Wikipedia, 2013a) is depicted in Figure 1. It is a distributed application structure in computing that partitions tasks or workloads between the providers of a resource or service, called servers and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. Client-server architecture is versatile and flexible in today's fast-changing IT landscape. It is modular in structure and is designed to improve flexibility, usability, scalability and interoperability. It is a realisation of increasingly distributed information networks accessible from anywhere at any time. Client-server architecture is the underlying technology of today's internet and cloud computing.

Figure 1 Client-server architecture

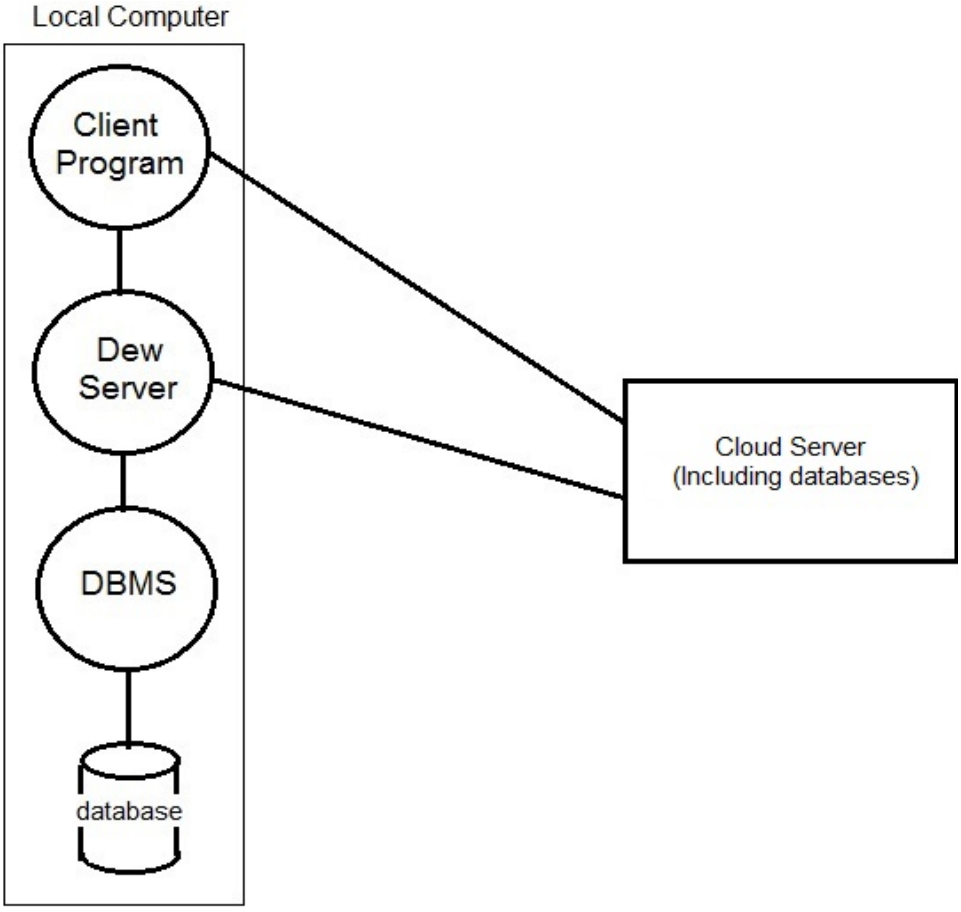


2.2 Cloud-dew architecture

The cloud-dew architecture is an extension of the client-server architecture. This architecture is illustrated in Figure 2. Comparing Figure 1 and Figure 2, the term server in

the client-server architecture has been replaced with the term *cloud server* in the cloud-dew architecture. The cloud-dew architecture was proposed to solve the data accessibility problem (described in Section 1) in cloud computing services and we want to be more specific that these servers are providing cloud services. When no confusion can occur, a cloud server can also be simply called a server. There is another difference between these two figures: a new kind of server, *dew server*, is introduced in the cloud-dew architecture. A dew server is a web server that resides on user's local computer. The dew server and its related databases have two functions: first, it provides the client with the same services as the cloud server provides; second, it synchronises dew server databases with cloud server databases.

Figure 2 Cloud-dew architecture



2.2.1 Dew servers

A dew server can be considered as a proxy server (Wikipedia, 2013d), although the purposes, usages and structure of a dew server are not the same with those of commonly-used proxy servers. A dew server has the following features:

1 A dew server is a lightweight web server. Usually, it only serves one user: the client.

2 A dew server usually only stores user's own data. The 'size' (i.e., data amount in related databases) of a dew server is much smaller than the 'size' of a cloud server. Metaphorically, a cloud server is as big as a cloud and a dew server is as small as a drop of dew.

3 A dew server disappears easily. The dew server's data could disappear due to different reasons: hardware damage and failure, virus infections and so on. Metaphorically, a dew server is as weak as a drop of dew.

4 A disappeared dew server can be recreated because all dew server data has a copy in cloud servers. Metaphorically, dew will come out again after it disappears as long as a cloud can provide all the necessities.

5 A dew server is accessible with or without an internet connection because it is running on the local computer. Metaphorically, a cloud could be far away, but the dew is close to you.

2.2.2 Cloud-dew applications

The last feature of a dew server suggests a new application: if a website adopts the cloud-dew architecture, it will be available all the time, with or without an internet connection, because a dew server runs on the user's local computer.

Suppose a user stores personal data, such as pictures and messages, on a website, say http://www.facebook.com. While the data is available publicly, the user cannot access his/her own data if an internet connection is not available. The user may decide to save a local copy of personal data in his/her own computer. However, saving pictures and messages in files may be awkward and difficult to manage.

Suppose a website, for instance http://www.facebook.com, adopts the cloud-dew architecture. The website will be duplicated onto a dew server running on a user's local computer. The duplication is not exactly copying. Generally speaking, the duplicated website in a dew server (we call it a *dewsite*) and the original website could be different in the following aspects:

1 the dewsite does not need to deal with a global heavy load so that it could be much simpler than the website

2 the dewsite will not include the proprietorial script that the website does not want to release. Instead, publicly-known technology will be used to implement similar functionalities

3 the content of a dewsite database could be limited

4 a new functionality, which will synchronise with the website, will be added to the dewsite.

Once a dewsite duplicating http://www.facebook.com is installed inside a dew server, the user may access the dewsite. At the beginning, the dewsite does not have the user's personal data. To let the dewsite synchronise with the website, the user needs to grant his/her http://www.facebook.com credentials to the dewsite. These credentials will be recorded by the dewsite and used in the future. The dewsite will be able to synchronise with the website http://www.facebook.com and the user's personal data and his/her

friends' related data will be transferred to the dewsite database. The dewsite will always be available even when an internet connection is not available.

If the user makes changes on the dewsite when there is no internet connection, the synchronisation will not be done immediately, but it can be performed automatically when an internet connection is available later.

2.2.3 The impacts of cloud-dew architecture

When the cloud-dew architecture is adopted, a cloud server and many dew servers cooperate as a distributed application to provide services. The cloud server provides global services; a dew server provides direct services to a user. The cloud-dew architecture has the following impacts:

1 A user not only can take advantage of cloud computing services to save his/her data in the cloud, but also can have immediate access and physical control over his/her own data at any time, even when there is no internet connection.

2 Because dew servers exist, a seemingly impossible capability, web surfing without an internet connection, becomes a reality. Although it is not feasible to access real-time information and perform communication tasks without an internet connection, it is still useful to surf through one's data and pre-downloaded data. Some communication tasks can also be prepared without internet connection, which will be done instantly and automatically when a connection is available later.

2.3 Local domain name system (LDNS)

In the example described in Section 2.2.2, a user puts a simplified version of website http://www.facebook.com in his/her dew server. Suppose another website, say http://www.hotmail.com, also adopts the cloud-dew architecture and the user also wants to put a simplified version of this website in the same dew server. A user can access the dew server by referring the URL http://localhost. But when there are two or more dewsites in the dew server, one URL cannot distinguish them. To accommodate two or more dewsites in the dew server and to make web surfing without internet connection more natural and more attractive, we introduce local domain name system (LDNS).

We suggest using *mmm* to replace *www* as an indicator that this URL is a local URL. For example, we use http://mmm.facebook.com to refer the dewsite of http://www.facebook.com and use http://mmm.hotmail.com to refer the dewsite of http://www.hotmail.com.

For instance, a user is attending a party where there is no internet connection. The user can open URL http://mmm.facebook.com in his/her browser and show favourite pictures and messages to friends, find phone numbers and other information in the website and put in new pictures and messages to his/her profile. He/she can also open URL http://mmm.hotmail.com to read and to reply to e-mails. Of course, at this moment, new e-mails cannot be received; the composed and 'sent' e-mails will not be transferred. The user can leave the computer in stand-by status. Later, when he/she travels to a place where an internet connection is available, the dew server will synchronise http://mmm.facebook.com with http://www.facebook.com automatically so that his/her http://www.facebook.com profile will be updated according to the changes; the dew

server will also synchronise http://mmm.hotmail.com with http://www.hotmail.com and the 'sent' e-mails will be transferred automatically.

Many offline e-mail clients provide similar functionalities. The novelty of this approach is that information organisation and user experience is almost the same with web surfing when actually the web is not available. Moreover, while an offline e-mail client can only provide e-mail functions, this approach opens the gate for various web resources to be simulated in a local computer.

While the task of the (global) domain name system is to map domain names to IP addresses, the tasks of the LDNS are different. Within the LDNS, IP address mapping is simple: to map all the local URLs to *localhost*. Besides IP address mapping, each local domain name should be redirected to the script of the corresponding dewsite. These two tasks are discussed in the following points:

2.3.1 Mapping local URLs to localhost

After LDNS has been introduced, URLs can be classified into regular URLs, such as http://www.example.com and *local URLs*, such as http://mmm.example.com. All local URLs should be mapped to *localhost* and taken over by the dew server instead of being sent to the internet. There are three methods to perform this mapping.

The first method is to configure the zone file (Wikipedia, 2013e) of the website's domain name. This is done by the owner of the domain through the domain's registrar. An *A* record could be added to map *mmm* to *127.0.0.1*. This method is very effective because the mapping will work on all dew servers. This method is suitable when a website has adopted the cloud-dew architecture. This method does not work without an internet connection when browser DNS cache expired.

The second method is to configure the *hosts* file. The *hosts* file (Wikipedia, 2013b) is a file that is used to map host names to IP addresses. This file exists in almost all operating systems. A user may put names of all the dewsites that he/she wants the dew server to take over into the *hosts* file. For example, a user may map http://mmm.site1.com, http://mmm.site2.com and so on to *localhost* or IP address *127.0.0.1*. This method only works on the computer where the *hosts* file has been changed and is suitable when the website has not adopted the cloud-dew architecture, but a dewsite is created by the user or a third party.

The third method is to change the behaviour of a browser so that local URLs are mapped to *localhost*.

2.3.2 Local domain name redirection

When a dew server receives a URL request to *localhost*, it needs to find out to which dewsite this URL request is and to redirect the request to the script of the corresponding dewsite. This task can be accomplished because the host name portion of the URL is captured by one of the environmental variables. The entry point of each dewsite script should be managed and searched in an efficient way.

For example, both dewsites http:/mmm.site1.com and http://mmm.site2.com will be mapped to *localhost* and taken over by the dew server. The dew server needs to find the environmental variable which contains the target host name of this URL request. The URL request will be redirected to the dewsite script for http://mmm.site1.com if the target host name is http://mmm.site1.com; the URL request will be redirected to the

dewsite script for http://mmm.site2.com if the target host name is http://mmm.site2.com. The details of this redirection are related to the dew server structure discussed in the next section (Section 2.4).

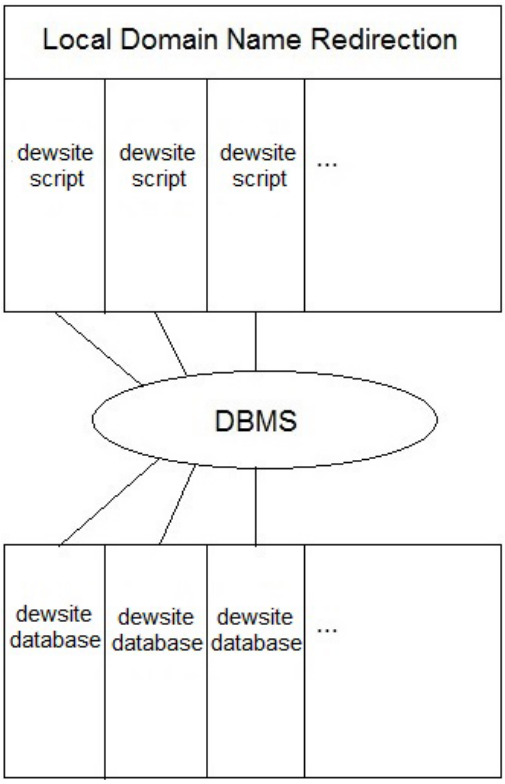
As a design alternative, we may not introduce LDNS at all. Instead, the behaviour of a browser could be modified so that it will always try to connect to the website in a cloud server. If a connection is not available, it will try to connect to the dew server and find the corresponding dewsite script.

Although this alternative is a viable solution, LDNS seems to be a better one. A dewsite could be a personalised copy of the website. Therefore, it is possible that a user may want to go to the dewsite even though the website is available. Once created, dewsites may gain new roles, thereby requiring proper local URLs to reference them. Local URLs might be referenced, linked and weaved with other URLs in the internet.

2.4 Dew server structure

Figure 3 shows the structure of a dew server.

Figure 3 Dew server structure



To make web surfing without an internet connection possible, the scripts and databases of all the dewsites, in which the user is interested, must present in the user's local computer. Instead of running one server for each dewsite so that many servers are running on one

local computer, a reasonable design is to run only one dew server on the local computer and the dew server will provide all services of these dewsites.

The script and database of each dewsite could be provided by the corresponding website, by the user, or by a third party. Wherever the dewsite script and database come from, they should be integrated in the dew server. For this reason, a dew server should adopt a plug-in structure. In Figure 3, we can see that dewsite script and dewsite database can be installed (plugged-in) on the dew server in a standardised manner. Once the script and database of a dewsite are plugged in, the dew server should be able to redirect those local URLs pointing to the dewsite correctly to the plugged-in script.

It is a challenge for many script packages and databases provided by different companies/organisations to run simultaneously in the same computer, under the control of the same dew server. The designers and programmers of a dew server should ensure that these components will not interfere with each other. Local domain isolation is a requirement of dew servers.

In the world of web development, many platforms and many database management systems are available. For web development platforms, there are PHP, Perl, Python, Java Servlet, JSP, ASP.NET and so on. For DBMSs, there are MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP HANA, dBASE, FoxPro, IBM DB2, among others. Apparently, it is impossible for a dew server to support all platforms and DBMSs. The following aspects will be helpful in the design of dew servers:

1 a website and its corresponding dewsite do not have to use the same platform and DBMS

2 existing open source server packages, such as UwAmp and XAMPP, show that it is feasible to support multiple platforms and multiple DBMSs on desktop and laptop computers

3 dew server development standards or guidelines are necessary to coordinate dew server developers and dewsite developers.

2.5 Dew server's potential applications

Although dew servers are introduced for a specific goal, which is to keep data available when there is no internet connection, many more new applications could be developed when powerful dew servers and related DBMSs are in place. One possibility of such applications is a personal information centre.

One of the basic functions of a dewsite is to synchronise with the website to keep dewsite data consistent with website data. To go one step further, the dewsite may actively search and collect website data according to the user's pre-determined rules.

For example, the user may specify a rule to the dewsite that at a specific time. The dewsite will collect a specific category of news items from a website such as http://www.cnn.com. Therefore, the user will still be able to browse CNN news from the dewsite http://mmm.cnn.com on his/her local computer even when the internet connection is lost.

Readers may notice that similar functionalities to distribute news have been widely used in RSS technology. However, using a stand-alone or web-based RSS reader is different from web surfing. For a web-based RSS reader, the only web page still visible when an internet connection is lost is the last page retrieved; any further surfing will lead

to a no-response page. The focus of our discussion is not how to obtain news, but how to surf the web without an internet connection.

The main idea of cloud-dew architecture is aimed at the applications on laptop computers and some powerful tablet computers. With the rapid development of mobile devices, some smart phones have much more computing resources than before. It is quite possible that cloud-dew architecture could be used on smart phones in the near future.

As an emerging area, mobile cloud computing (Fernando et al., 2013; Chun et al., 2011) is very promising. One approach of mobile cloud computing is to consider other mobile devices in the local vicinity as a peep-to-peer network. The collective resources of various mobile devices in this network will be utilised by the user in different ways. Of course, some incentive should be involved for these mobile devices to share their resources.

If cloud-dew architecture is used in mobile cloud computing, a user may allow the dew server on his/her smart phone to be accessed by other users in the vicinity. This leads to the following amazing application: suppose a group of people are in an area without internet connection. Although each smart phone is not capable of carrying out many functions and having enough information, the users can share their functions and information by accessing each other's dew servers. Thus, they may be able to accomplish more in this limited environment than without the shared dew servers. This application also shows that LDNS described in Section 2.3 is necessary.

3 An experimental cloud-dew system

3.1 Basic functionalities

An experimental system has been built to test the ideas of the cloud-dew architecture. This system includes a website http://www.clouddew.com and a dewsite http://mmm.clouddew.com. The dewsite was built on our local computer. To demonstrate this experimental system, we needed to install a dew server and the dewsite http://mmm.clouddew.com on the tester's local computer. To make the procedure easier, we used a website http://mmm.clouddew.com to simulate the dewsite. Without going through the installation process, a tester can have the same experience as if a dew server and this dewsite have been installed in the tester's local computer.

The basic functionalities of website http://www.clouddew.com are very simple: a user can create, edit, save and delete HTML files. This website can serve many users. A user does not need to register to use this website. This website accepts *openid* login (Wikipedia, 2013c).

The dewsite http://mmm.clouddew.com has the same functionalities as the website: a user can create, edit, save and delete HTML files. The dewsite does not accept *openid* login, because that the *openid* login needs an internet connection. To keep the dewsite usable without an internet connection, *openid* login cannot be used. Users have to go through a simple registration process to use the dewsite.

3.2 User identity mapping

Some websites need to verify user identities. A login process is a typical way to verify user identities. For a cloud-dew implementation of such website, two kinds of identities

are involved: website user identities and dewsite user identities. We are familiar with website user identities: we need to go through a registration process to obtain credentials, such as a username and a password and use these credentials to prove our identity. But do we need dewsite user identities? If the local computer is only used by one user, the dew user identity may be not necessary. Generally, it is possible for multiple users to access one dew server and it is also possible for one user to have multiple identities for different purposes. Thus, it is necessary to keep dewsite user identities. Importantly, website user identities and dewsite user identities are completely different. Even though they may have the same username/password, the website user identities and dewsite user identities are still different because they are verified by different servers and managed in different databases.

This leaves a unique problem to be dealt with in a cloud-dew system: how to associate a website user identity with a dewsite user identity.

In this experimental system, suppose the user John Doe uses his *openid johndoe.myopenid.com* to access the website http://www.clouddew.com and he also registers on the dewsite http://mmm.clouddew.com as user *johndoe*. He creates a file *FA* on the dewsite; *FA*'s owner is *johndoe*. After a synchronisation process, file *FA* is transferred to the website. However, one thing must be changed: the owner of *FA* should not be *johndoe*, because on the website, there is no user *johndoe*. John Doe logs on the website using *openid johndoe.myopenid.com*. Thus *FA*'s owner should be converted to *johndoe.myopenid.com* when it is transferred to the website. In this way, John Doe will be able to access file *FA* again when he logs onto the website using his *openid*.

How does the synchronisation mechanism know the user *johndoe* in the dewsite is equivalent to the user *johndoe.myopenid.com* on the website? John Doe must specify that. In another words, John Doe will create a link between user *johndoe* on the dewsite and user *johndoe.myopenid.com* on the website. In this experimental system, a link-creating mechanism was designed for this task.

On the top left corner of the dewsite page http://mmm.clouddew.com, is the button 'Open http://www.clouddew.com'. This button can open the website page inside the dewsite page as a frame. Another button will close this frame. When the frame is open, both the website page and the dewsite page appear inside one single web document. The user can login to the dewsite and the website in this single web document. When the user has logged into both sites, a button 'Create a link to http://www.clouddew.com' will be enabled. When the button is pressed, a link is created.

Internally, the *userid* of the dewsite user and the *userid* of the website user will be put into a mapping table; access keys will be issued so that the dewsite can access the linked website user's data and the website can access the linked dewsite user's data.

3.3 Synchronisation

A button 'Sync with http://www.clouddew.com' will be enabled once a dewsite user identity has been linked to a website user identity. The synchronisation procedure will upload and/or download those files that have not been synchronised. The procedure is based on the timestamps of the files.

In this cloud-dew experimental system, what the user created will be saved in the cloud and available anywhere. The user can access his/her files on his/her local computer at any time at any location even there is no internet connection. If the user is away from his/her computer, he/she can still access his/her files on the website from another

computer that is connected to the internet; all changes will be reflected on his/her computer after synchronisation at a later time.

4 Conclusions

The cloud-dew architecture is an extension of the client-server architecture. A new kind of server, dew server, is introduced in this architecture. A dew server is a web server that resides on a user's local computer, which performs various functions. With the help of the dew server plug-in structure and the LDNS, the cloud-dew architecture enables a new experience: web-surfing without an internet connection. The key contribution of the cloud-dew architecture is that it provides a systematic scheme to organise user local data/programmes, to integrate the usage of such local data/programmes with web-surfing activities seamlessly and to maximise the synchronisation between local data/programmes and the cloud. An experimental system was built to test and demonstrate the ideas of the cloud-dew architecture.

References

Chun, B., Ihm, S., Maniatis, P., Naik, M. and Patti, A. (2011) 'CloneCloud: elastic execution between mobile device and cloud', in *Sixth Conference on Computer Systems (EuroSys'11)*, Salzburg, Austria, 10–13 April, New York, NY, USA, ACM, pp.301–314.

Fernando, N., Loke, S.W. and Rahayu, W. (2013) 'Mobile cloud computing: a survey', *Future Generation Computer Systems*, Vol. 29, No. 1, pp.84–106.

Goscinski, A. and Brock, M. (2011) 'Toward higher level abstractions for cloud computing', *International Journal of Cloud Computing*, Vol. 1, No. 1, pp.37–57.

Habib, S.M., Hauke, S., Ries, S. and Muhlhauser, M. (2012) 'Trust as a facilitator in cloud computing: a survey', *Journal of Cloud Computing: Advances, Systems and Applications*, [online] http://www.journalofcloudcomputing.com/content/1/1/19 (accessed 27 July 2013).

Rimal, B.P., Choi, E. and Lumb, I. (2009) 'A taxonomy and survey of cloud computing systems', in *NCM '09: Proceedings of Fifth International Joint Conference on NC, IMS and IDC*, Seoul, Korea, pp.44–51.

Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K. Llorente, I.M., Montero, R., Wolfsthal, Y., Elmroth, E., Cáceres, J., Ben-Yehuda, M., Emmerich, W. and Galán, F. (2009) 'The reservoir model and architecture for open federated cloud computing', *IBM Journal of Research and Development*, Vol. 53, No. 4, pp.535–545.

Sasikala, P. (2011) 'Cloud computing: present status and future implications', *International Journal of Cloud Computing*, Vol. 1, No. 1, pp.23–36.

Tsai, W., Sun, X. and Balasooriya, J. (2010) 'Service-oriented cloud computing architecture', in *ITNG2010: Seventh International Conference on Information Technology: New Generations*, Las Vegas, NV, pp.684–689.

Wikipedia (2013a) *Client-Server Model – Wikipedia, the Free Encyclopedia* [online] http://en.wikipedia.org/wiki/Client-server_architecture (accessed 27 July 2013).

Wikipedia (2013b) *Hosts (File) – Wikipedia, The Free Encyclopedia* [online] http://en.wikipedia.org/wiki/Hosts_(file) (accessed 27 July 2013).

Wikipedia (2013c) *OpenID – Wikipedia, The Free Encyclopedia* [online] http://en.wikipedia.org/wiki/OpenID (accessed 27 July 2013).

Wikipedia (2013d) *Proxy Server – Wikipedia, The Free Encyclopedia* [online] http://en.wikipedia.org/wiki/Proxy_server (accessed 27 July 2013).

Wikipedia (2013e) *Zone File – Wikipedia, The Free Encyclopedia* [online] http://en.wikipedia.org/wiki/Zone_file (accessed 27 July 2013).

Xiong, N., Rindos, A., Russell, M.L., Robinson, K.P., Vandenberg, A. and Pan, Y. (2011) 'Sharing computing resources to satisfy multi-cloud user requirements', *International Journal of Cloud Computing*, Vol. 1, No. 1, pp.81–100.

Zhang, L. and Zhou, Q. (2009) 'CCOA: cloud computing open architecture', in *ICWS2009: IEEE International Conference on Web Services*, Los Angeles, CA, pp.607–616.