Cloud-dew architecture: realizing the potential of distributed database systems in unreliable networks

Yingwei Wang[1] **and Yi Pan**[2]
[1]Department of Computer Science, University of Prince Edward Island,
Charlottetown, Prince Edward Island, Canada
[2]Department of Computer Science, Georgia State University,
Atlanta, Georgia, United States

Abstract - *Distributed database systems, which continue to inspire new architectures and new applications, have great potential in the modern computing world. In this paper, we show that the newly-proposed cloud-dew architecture realizes the potential of distributed database systems in the unreliable network environment, and provides the possibility of web-surfing without an Internet connection. Distributed database systems are generic and versatile; the proper applications of distributed database systems and their features will be beneficial to users and service providers.*

Keywords: distributed database system; cloud-dew architecture; peer-to-peer; super-peer; transparency

1 Introduction

A distributed database system is defined as a collection of multiple, logically interrelated databases distributed over a computer network [1]. Combined with other components, distributed database systems [1-3] play central roles in various applications. It is believed that the potential of distributed database systems has not been realized fully as yet [1]. The following paragraph describes one of the promising possibilities of distributed database systems:

"The failure of a single site, or the failure of a communication link which makes one or more sites unreachable, is not sufficient to bring down the entire system. In the case of a distributed database, this means that some of the data may be unreachable, but with proper care, users may be permitted to access other parts of the distributed database" [1].

This description suggests that an application may still work when a communication link fails. If the application is a web application and the communication link is an Internet connection, the possibility exists that the web application may still work when an Internet connection is not available. Today, web applications are daily essentials but an Internet connection is not always available. This potential is very attractive.

As indicated in the above paragraph, the great potential cannot be realized automatically, and "proper care" is necessary.

Is the great potential realizable? What is the proper care to realize this great potential? A newly-proposed architecture [4] shows that it is possible to do web-surfing without an Internet connection. In this case, the proper care is the architecture: cloud-dew architecture.

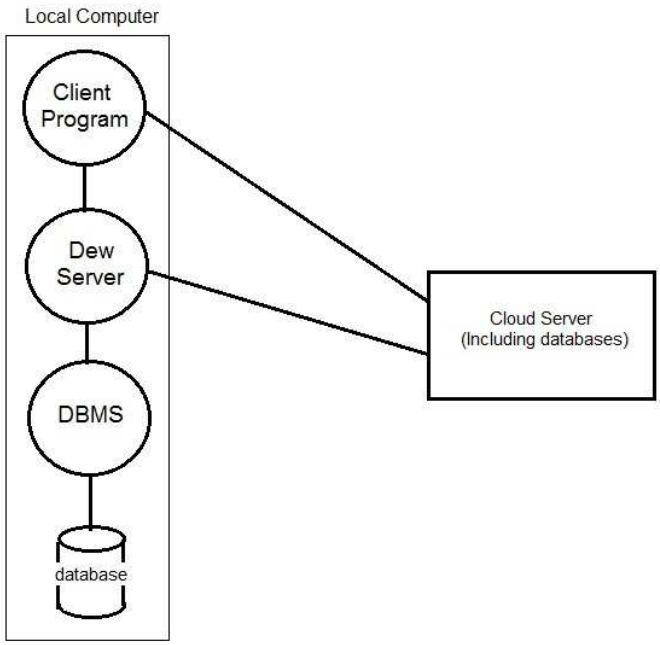

Figure 1: Cloud-dew architecture

Cloud-dew architecture is an extension of the client-server architecture [4]. This architecture is illustrated in Figure 1, and the client-server architecture is depicted in Figure 2 for comparison. 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. 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 synchronizes dew server databases with cloud server databases. A dew server has the following features:

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

(2) A dew server usually stores only the user's 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 for different reasons, for instance: hardware damage and failure or virus infections. Metaphorically, a dew server is as weak as a drop of dew.

(4) A vanished dew server can be recreated because all dew server data has a copy in the 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.

Figure 2: Client-server architecture

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, in this case 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 (called 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 synchronize 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. A local domain name system (LDNS) could be introduced so that the above-mentioned dewsite can be accessed using the URL http://mmm.facebook.com instead of http://localhost [4]. This URL makes web-surfing without an Internet connection more attractive.

At the beginning, the dewsite does not have the user's personal data. To let the dewsite synchronize 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 synchronize 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 synchronization will not occur immediately, but it will be performed automatically when an Internet connection is available later.

If many websites adopt cloud-dew architecture and many dewsites are available for a local computer to host, the potential experience of web-surfing without an Internet connection will become a reality.

In this paper, we analyze the cloud-dew architecture from the distributed database system viewpoint, and further explore the potential of the distributed database systems.

2 Single-super-peer hybrid P2P network

Cloud-dew architecture extends client-server architecture with dew servers. Such extension gives clients the power of servers. The new architecture is very similar to peer-to-peer networks [5-10], with the cloud server (web server) as the central node. Such a new structure can be classified as a hybrid P2P network, or a super-peer system. In this system, some nodes are given special tasks to perform. Apparently, the web server node is a super-peer. In a standard hybrid P2P network, there are two or more super-peers; if there is only one super-peer in the system, this reduces to the client-server architecture [1].

In the cloud-dew architecture, a single-super-peer P2P network may not reduce to client-server architecture because each peer (dew server) is not just a client, but also a server. The database is replicated between the super-peer and the other peers. If the communication link between the super-peer and one peer fails, the peer is partitioned from the whole

system. The dew server on this peer node will provide some basic web services and database services so that the goal of web-surfing without an Internet connection can be achieved.

A peer node not only deals with the super-peer node, but also can perform other actions with other peers. The real P2P features are reflected by these actions and new applications are made possible by these features.

Single-super-peer P2P is worth exploring not only as the underlying structure of cloud-dew architecture, but also as a promising extension of client-server architecture. Single-super-peer P2P is much simpler than multiple-super-peer P2P; therefore, it is easier to implement. However, it is more complicated than client-server architecture so that it can support various distributed database applications.

3 Transparencies

Transparency refers to separation of the higher-level semantics of a system from lower-level implementation issues [1, 11]. In other words, a transparent system "hides" the implementation details from users. There are different forms of transparency. In the context of this paper, the following forms of transparency are of our concern: replication transparency and distribution transparency [1].

Replication transparency refers to whether the users should be aware of the existence of copies or whether the system should handle the management of copies and the users should act as if there is a single copy of the data.

Distribution transparency, or network transparency, refers to that there would be no difference between database applications that would run on a centralized database and those that would run on a distributed database.

Although it is desirable that replication transparency and distribution transparency be provided as a standard feature of DBMSs, this is not always the case. The essences of transparency are: (1) to hide some details; (2) to create an illusion.

Suppose we are accessing a website, say http://www.facebook.com. If an Internet connection is not available, this website will not be accessible. We may replicate the website and the database in the local node so that this website will still be available even though there is no internet connection. If replication transparency and distribution transparency are both kept, we may need to modify the behavior of the browser so that when we want to access http://www.facebook.com, the browser will first try to connect to the website server; if the website server is available, everything is normal; if the website server is not available, the browser will try to connect to the local server and access the replicated website and database.

The transparency hides all the detailed behavior of the browser, hides the communication link failure, hides the existence and the operation of a local website and related database, and creates an illusion that there are no communication link failures and the website http://www.facebook.com is still available.

However, does this transparency arrangement give the user what he/she wants? The ability to still use a website when there is no Internet connection is convenient. Nevertheless, the offline website cannot provide exactly the same services as the online website. For example, the online web application changes the online database status as the user makes changes, but the offline web application will change the online database only once the user is online again. Additionally, the offline web application can only access a portion of the online database. For these reasons, the illusion created by the transparencies is, perhaps, too lofty and is not realistic. A more practical solution is not to keep the transparencies, and to tell the user exactly what is being provided.

In the cloud-dew architecture, a local domain name system is provided. Such a local domain name system is based on the fact that transparencies are not supported. Users know the cloud and the dew. Using the example mentioned above, a user knows the difference between http://www.facebook.com and http://mmm.facebook.com,, and has different expectations for these two related websites.

To summarize the above discussions, although transparency is generally considered a great feature, a concrete application may not want to support transparency for either of the following reasons:

(1) The illusion created by the transparency is not realistic in this application.

(2) The user needs to be involved in some details that otherwise would be covered by transparency.

4 Replication update strategies

Distributed database systems can increase system availability and remove single points of failure by replicating data [1]. In the case of cloud-dew architecture, database replication is the foundation of web-surfing without an Internet connection. Although data replication has clear benefits, it poses the considerable challenge of keeping different copies synchronized. In a cloud-dew architecture application, if there is no Internet connection between the cloud and the dew and the user has changed data at the dew level, the dew changes must be synchronized with the cloud server (the central node) when it is possible. In other words, mutual consistency, which refers to the replicas converging to the same value, is necessary [3].

In terms of replication update methods, two orthogonal dimensions can be used to classify [1]. One dimension is eager update and lazy update; eager update performs all of the updates within the context of the global transaction; lazy update propagates the updates sometime after the initiating transaction is committed. The other dimension is centralized update propagation and distributed update propagation; the centralized update requires that the updates are first applied at a master copy; the distributed update applies the update on the local copy at the site where the update transaction originates. Therefore, four combinations are possible: eager centralized, eager distributed, lazy centralized, and lazy distributed.

In the application context where the cloud-dew architecture is proposed, Internet connections may get lost constantly for an extended period of time. In such a situation, eager replication update is not possible and not necessary. Thus, lazy update is the choice. Between lazy centralized and lazy distributed, the central super-peer node is often not available when the Internet connection is lost. This leaves only one applicable replication update method: lazy distributed update. In this method, the propagation to other copies is done asynchronously from the original transaction, by means of refresh transactions that are sent to the replica sites some time after the update transaction commits. Lazy distributed replication protocols are the most complex ones owing to the fact that updates can occur on any replica and that they are propagated to the other replicas lazily [1, 12-14].

Normally, the dew database is a partial replica of the central database. The central database contains data of all users, but the dew database can only contain data of the current user. Therefore, the dew database is a subset of the central database. In special cases, should the application require, it is possible for the dew database to contain data beyond the central database. There are two situations in which part of the dew database is not replicated to the central database. The first situation is that this portion of data is trivial and only related to detailed execution of the dew operations. The second situation is that this portion of data is too important and the user does not want to take any risk in sending this portion of data to the Internet. In either case, the extra dew data will make more varieties of web application possible.

5 Conclusions

From a distributed database systems viewpoint, cloud-dew architecture's ability to provide a web-surfing experience without an Internet connection is the realization of the distributed database systems' potential. The organization of cloud-dew architecture can be considered as a single-super-peer P2P network. Although multiple-super-peer P2P networks are popular, the single-super-peer P2P network may be a promising extension of the client-server architecture. Transparencies are generally desirable features in the design of distributed database systems, but they may not be always desirable. In cloud-dew architecture, non-transparent solutions are more suitable because users need to be aware of the communications link failure and to expect a realistic replacement. The local replica of a database not only is a subset of the database, but also could have extra local data. The local data is not replicated to the super-peer central database because either the data is too trivial to be replicated or is too important to be replicated. The extra local data could lead to new applications. A distributed database system is a generic, versatile structure; the proper use of its features may bring great inspiration to the computing world.

Acknowledgement

YW would like to gratefully and sincerely thank Prof. Tamer Ozsu at the University of Waterloo. He has discussed with YW about the direction of the cloud-dew architecture in the early stage. His vision and guidance played an important role in YW's research.

References

[1] Ozsu and Valduriez. "Principles of Distributed Database Systems". Spinger Science, 2011.

[2] Marius Cristian MAZILU. "Database Replication"; Database Systems Journal, Vol. I, No. 2, 33—38, 2010.

[3] Davidson, S. B., Garcia-Nilina, H., and Skeen, D. "Consistency in partitioned networks"; ACM Comput. Surv., 17(3):341-370, 1985.

[4] Yingwei Wang. "Cloud-Dew Architecture"; International Journal of Cloud Computing, OPEN ACCESS, http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijcc, 2014

[5] Beverly Yang, Hector Garcia-Molina. "Designing a Super-peer Network"; in Proceedings of the 19th International Conference on Data Engineering (ICDE), Bangalore, India, 2003.

[6] Beverly Yang, Hector Garcia-Molina. "Comparing Hybrid Peer-to-Peer Systems"; in Proceedings of the 27th International Conference on Very Large Databases (VLDB), Roma, Italy, 2001.

[7] Ulusoy, O. "Research Issues in peer-to-peer data management"; in Proc. 22nd Int. Symp. On Computer and Information Science, 1--8, 2007,

[8] Bernstein, P. A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., and Zaihrayeu, I. "Data management for peer-to-peer computing: A vision"; in Proc. 5th Int. Workshop on the World Wide Web and Databases, 89--94, 2002.

[9] Daswani, N., Garcia-Molina, H., and Yang, B. “Open problems in data-sharing peer-to-peer systems”; in Proc. 9th Int. Conf. on Database Theory, 1--15, 2003.

[10] Valduriez, P. and Pacitti, E. “Data management in large-scale p2p systems”; in Proc. 6th Int. Conf. High Performance Comp. for Computational Sci., 104--118, 2004.

[11] Umar Farooq Minhas, Shriram Rajagopalan, Brendan Cully, Ashraf Aboulnaga, Kenneth Salem, Andrew Warfield. “RemusDB: Transparent High-Availability for Database Systems”; in Proc. of the VLDB Endowment, 4(11), 2011.

[12] Khuzaima Daudjee, Kenneth Salem. “Lazy Database Replication with Snapshot Isolation”; in Proc. International Conference on Very Large Data Bases (VLDB'06), 715--726, 2006.

[13] Khuzaima Daudjee, Kenneth Salem. “A Pure Lazy Technique for Scalable Transaction Processing in Replicated Databases”; in International Conference on Parallel and Distributed Systems (ICPADS'05), 802--808, 2005.

[14] Khuzaima Daudjee, Kenneth Salem. “Lazy Database Replication with Ordering Guarantees”; in Proc. International Conference on Data Engineering (ICDE'04), 424--435, 2004.