Proceedings of National Conference on Recent Advances in Computer Science & Engineering (NCRACSE-2017)

International Journal for Modern Trends in Science and Technology
Volume: 03, Special Issue No: 01, February 2017
ISSN: 2455-3778
http://www.ijmtst.com

A Study on Deployment of Web Applications Require Strong Consistency using Multiple Clouds

P.R.S.M.Lakshmi[1] | K.Santhi Sri[2] | Dr.N.Veeranjaneyulu[3]

[1]Assistant Professor, VFSTR University, Guntur, AP, India.
[2]Associate Professor, VFSTR University, Guntur, AP, India.
[3]Professor, VFSTR University, Guntur, AP, India.

To Cite this Article
P.R.S.M.Lakshmi, K.Santhi Sri and Dr.N. Veeranjaneyulu, "A Study on Deployment of Web Applications Require Strong Consistency using Multiple Clouds", *International Journal for Modern Trends in Science and Technology*, Vol. 03, Special Issue 01, 2017, pp. 14-17.

ABSTRACT

Web Applications require strong consistency to be deployed in multiple Clouds, various scalable database systems that can guarantee strong inter-data center consistency with reduced network. For applications using these database systems, it is essential to take both network latencies to end users and communication transparency caused by maintaining database consistency into account when selecting the hosting data centers. In this paper, we explain about to require strong inter-data center consistency under dynamic workloads.

KEYWORDS: *strong, consistency, deployment*

I. INTRODUCTION

Web applications are deployed in geographically dispersed Cloud data centers, application providers need to handle data consistency across data centers (Inter-data center consistency), which is challenging for some applications requesting strong consistency, e.g., e-commerce and banking. To make things worse, traditional. Inter-data center commit (two-phase commit) involves high network cost. Therefore, applications often have to adopt eventual consistency (asynchronous replication) to minimize user perceived latencies, which complicates application logic and forces application developers to handle the conflicts and errors caused by inconsistent data [1], even though such cases are rare in production as data synchronization usually completes in a short time in eventual-consistent databases [2]. After realizing that lack of strong consistency has impaired developing productivity, industry and academia shift to developing new databases that can guarantee strong interdata center consistency [4, 5, 7, 8, 9, 10, 1, 12] to help relieve the programmers' coding burden. Though inter-data center consistency protocols of these new databases are often optimized regarding network overhead, the resulted network delays are still significant and cannot be ignored. Thus, to minimize user perceived response time, it is essential to take the database network delay into account when selecting hosting data centers and when routing requests submitted by different users to the chosen data centers.

In this chapter, we aim to minimize the total excess response time users may perceive beyond the Service Level Objective (SLO) for applications with various inter-data center consistency requirements. The proposed approach benefits application providers so that they can ease their development by adopting these new databases, and in the meantime keep the performance penalties as low as possible.

The contributions of the paper are two folds. Firstly, a genetic algorithm that searches a deployment plan with a minimum amount of SLO violations when the application is initially migrated to the Cloud. After the initial deployment, the application performance may degrade as time passes due to changes in workload distribution. Secondly, to react to these changes, we propose a decision-making algorithm that continuously optimizes the deployment to balance application performance, redeployment cost, and operational cost. We exemplify how our approach can be applied to two widely used databases (Cassandra [4] and Galera Cluster [7]). To demonstrate the effectiveness of our approach, we conduct simulation studies using settings of two real applications (TPC-W [13], an e-commerce website, and Twissandra [14], a Twitter-like social network application). The existing protocols and databases with strong inter-data center consistency support.

II. Database Supporting Strong Inter-Data Center Consistency

Google's Systems

Google have been developing distributed databases that are both highly scalable and strongly consistent. Their first achievement is MegaStore [5]. It implements ACID semantics within each entity group (objects stored together) using synchronous replication based on optimized Paxos, and transaction across entity groups using two-phase commit. The second outcome is Spanner [8], which further supports external consistency (linearizability) with the help of physically synchronized clocks (GPS and atomic clock).Upon Spanner, Google built F1 [1], a distributed relational database system for their critical AdWords platform. It provides more enriched transaction semantics with high availability and scalability. All Google's systems remain proprietary.

Open Source Databases

Cassandra [4] is a shared nothing NoSQL database using quorum-based protocol for its consistency model. It allows users to set individual read and write quorums at the granularity of query. It also provides limited transaction support (lightweight-transaction) starting from version 2.0 using a heavy-weight Paxos consensus protocol, which requires four round-trip messages to complete. Galera cluster [7] is an open source scalable synchronous replication solution developed and maintained by Codership for MySql. Galera's replication is based on certification based commit [6].

III. Application and Deployment Model

Target Applications

We target session-based web applications. We assume the delay of the application is dominated by the round-trip time (RTT) between different parties, as the processing time of the request can be considered constant provided that enough computing resources are provisioned. In this case, whether the SLOs can be met is largely determined by the involved network latencies. To benefit from our work, the application should also be deployed in geographically dispersed data centers, and some of its requests should require strong consistency, e.g., a group working application that always reflects the newest updates to its end-users, a social-network application that consistently and timely shows people's posts and comments, or a distributed banking application that needs to satisfy ACID semantics.

Deployment Model

We assume the whole software stack of the application (including application servers and underneath databases) is deployed in multiple geographically dispersed data centers and each application replica can autonomously scale up and down according to the changing workloads. We assume all chosen data centers have the full copy of data. Companies, like Face book [11], commonly adopt this approach. Furthermore, the databases studied in this chapter, Cassandra and Galera.

Cluster, support only full replication for multi-data center deployment within the same key space or namespace. The target applications should also use shared-nothing multimaster database clusters, which means all database queries originated from any server can be served by database nodes collocated in the same data center. Depending on the database and different queries' consistency requirements, we also consider the network delays caused by communications among database cluster nodes located in different data centers into the problem model. All inner-data center communications, otherwise, are omitted.

Table 1: Symbols of the Cassandra Model

Term	Meaning
R_l	Set of read queries in type l request
W_l	Set of write queries in type l request
$\mathbf{Qr^l_k}$	Read quorum of the kth read query in request type l
Qw^l_m	Write quorum of the mth write query in request type l
r	The replication factor of data centers
$a(j, k, \mathbf{X})$	The function finds the kth shortest RTT latency among all latencies between each data center within **X** and data center j

We classify users into groups according to their geographic locations. All requests from the same location are routed to the same data center using DNS routing services similar to Amazon Route 53's Geo Routing [3].

Initial Deployment: When the application is initially migrated to the Clouds, our approach aims to select the hosting data centers and route requests to chosen data centers with minimum amount of total estimated SLO violations according to the current geographical distribution of requests2.

Deployment Optimization: In the second step, our approach continuously attempts to maintain high performance of the application by contracting, optimizing, or expanding the deployment with acceptable migration3 efforts in response to changes in the requests distribution.

In this paper, we use the term **expand** and **contract** respectively for increasing and decreasing the number of chosen data centers. We focus on the geographical distribution of resources instead of the total resource amount, for which the term commonly used is scaling up and down.

The Cassandra Model: The widely-adopted replication strategy for Cassandra involving multiple data centers in production is symmetric replication, where each data center stores the same number of replicas [4]. Cassandra uses quorum-based protocol to implement consistent read/write operations across replicas, and it supports various consistency configurations at the granularity of query. Given the set of selected data centers (**X**), using the symbols in Table 1, its database network overhead dlt^l_j can be modeled

We model the delay of the read/write query as the slowest replica's response time in the quorum. For example, if the read quorum is 3 and each data center holds 1 data replica, Cassandra will wait to receive replies from the 2 replicas located in other data centers as the network delay to the local copy is orders of magnitude smaller. Hence, the resulted delay will normally be the second shortest RTT latency from the local data center j to the other selected data centers.

In Cassandra, the remote replica only replies digest of the objects. If the local copy is stale, it will send another request to fetch the complete data and update all the stale replicas. We ignore this overhead as such case is rare and, thus, only impose a little impact on the average delay of all the requests.

The administrator is responsible for deciding the quorum settings of each query, as, besides consistency and performance, other concerns in this process may complicate the decision, such as availability ($Qr = 1, Qw = H$ maximizes the performance for read intensive applications but is susceptible to failures). For query requiring strong consistency. Application administrator should specify its read/write quorum so that the object's Qr and Qw satisfy $Qr + Qw > H$. Certainly, it is also possible to set a weaker configuration [2] if strong consistency is unnecessary.

Table 2: Symbols of the Galera Model

Term	**Meaning**
$b(j, \mathbf{X})$	The function finds the largest RTT latency among all latencies between each data center within X and data center j
V^l	Number of transactions that have write operations in request type l

The Galera Model:

In Galera cluster, all read-only transactions are executed locally while transactions with write operations synchronously replicated to all remote replicas using certification-based commit [6]. There is no further group communication involved in the protocol [6] after the transaction ID is determined; the network latency is dominated by the data center that has the largest RTT latency to the request originator. Based on symbols in Table 2, dlt^l_j, in this case, can be simply formulated as:

$$dlt^{l}_{j}= V^{l}\beta\ (j,\mathbf{X})\ j\ \text{€}\ \mathbf{X},\ l\ \text{€}\mathbf{I}$$

Where V^{l} is the number of database transactions that have write operations in type l request, and the function *b*(j,**X**) returns the largest RTT latency among all latencies between each data center within **X** and data center j.Galera nodes may queue the messages before delivering them due to the group communication overhead. We neglect this delay because we believe it is unpredictable, application specific, and also insignificant compared to the network transfer delay. To build a more precise model, application administrators can profile their applications.

IV. Conclusion

We proposed an approach to help web application providers deploy their applications with various inter-data center consistency requirements across multiple Cloud data centers.

V. Future Enhancement

After the application is deployed in the Clouds, it comes to the task to ensure just enough resources are provisioned to the application during its life cycle so that the QoS requirements can be met with the minimum cost incurred. We shift our focus to the provisioning aspect of web application management and propose an auto-scaler for web applications using heterogeneous spot instances.

References

[1] J. Shute, R. Vingralek, B. Samwel, B. Handy, C. Whipkey, E. Rollins, M. Oancea,K.Littlefield, D. Menestrina, S. Ellner, J. Cieslewicz, I. Rae, T. Stancescu, and H. Apte, “F1: A distributed sql database that scales,” Proc. VLDB Endow., vol. 6,no. 11, pp. 1068–1079, Aug. 2013.

[2] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica, “Quantifying eventual consistency with pbs,” The VLDB Journal, vol. 23, no. 2, pp. 279–302, 2014.

[3] “Route 53,” 2016. [Online]. Available: https://aws.amazon.com/route53/

[4] Apache, “Cassandra,” 2015. [Online]. Available: http://cassandra.apache.org/

[5] J. Baker, C. Bond, J. C. Corbett, J. Furman, A. Khorlin, J. Larson, J.-M. Leon, Y. Li,A. Lloyd, and V. Yushprakh, “Megastore: Providing scalable, highly available age for interactive services,” in Proceedings of the Conference on Innovative Data system Research (CIDR), 2011, pp. 223–234.

[6] Codership, “Certification-based commit,” 2014. [Online]. Available: http://galeracluster.com/documentation-webpages/certificationbasedreplication.html

[7] “Galera cluster,” 2015. [Online]. Available: http://galeracluster.com/products/

[8] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat,A.Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito,M. Szymaniak, C. Taylor, R. Wang, and D. Woodford, “Spanner: Google&rsquo;sglobally distributed database,” ACM Trans. Comput. Syst., vol. 31, no. 3, pp. :1–8:22, Aug. 2013.

[9] T. Kraska, G. Pang, M. J. Franklin, S. Madden, and A. Fekete, “MDCC: Multi-data center consistency,” in Proceedings of the 8th ACM European Conference on Computer Systems, ser. EuroSys ’13. New York, NY, USA: ACM, 2013, pp. 113–126.

[10] H. Mahmoud, F. Nawab, A. Pucher, D. Agrawal, and A. El Abbadi, “Low-latency multi-datacenter databases using replicated commit,” Proc. VLDB Endow., vol. 6,no. 9, pp. 661–672, Jul. 2013.

[11] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy,M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani, “Scaling memcache at facebook,” in Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). Lombard, IL: USENIX, 2013, pp.385–398.

[12] A. Thomson, T. Diamond, S.-C. Weng, K. Ren, P. Shao, and D. J. Abadi, “Calvin:Fast distributed transactions for partitioned database systems,” in Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD’12. New York, NY, USA: ACM, 2012, pp. 1–12.

[13] Transaction Processing Performance Council, “TPC-W Workload,” 2015. [Online]. Available: http://www.tpc.org/tpcw/

[14] Twissandra, “Twissandra.” [Online]. Available: https://github.com/twissandra/twissandra