Self-Adaptive Cost-Efficient Consistency Management in the Cloud

Houssem-Eddine Chihoub

Self-Adaptive Cost-Efficient Consistency Management in the Cloud

Houssem-Eddine Chihoub *
INRIA Rennes - Bretagne Atlantique
Rennes, France
houssem-eddine.chihoub@inria.fr

***Abstract*—Many data-intensive applications and services in the cloud are geo-distributed and rely on geo-replication. Traditional synchronous replication that ensures strong consistency exposes these systems to the bottleneck of wide areas network latencies that affect their performance, availability and the monetary cost of running in the cloud. In this context, several weaker consistency models were introduced to hide such effects. However, these solutions may tolerate far too much stale data to be read. In this PhD research, we focus on the investigation of better and efficient ways to manage consistency. We propose self-adaptive methods that tune consistency levels at runtime in order to achieve better performance, availability and reduce the monetary cost without violating the consistency requirements of the application. Furthermore, we introduce a behavior modeling method that automatically analyzes the application and learns its consistency requirements. The set of experimental evaluations on Grid'5000 and Amazon EC2 cloud platforms show the effectiveness of the proposed approaches.**

I. Introduction

Cloud computing evolution over the years made services deployment in the cloud very easy. Moreover, clients deployments can be at very large scales with no need for physical infrastructure management and maintenance. Such paradigm has attracted a variety of service providers and IT managers. Henceforth, they migrate their systems to the cloud to benefit from multiple features such as on demand scalability, geographical availability, and economical deployments.

Many of the applications that run in the cloud are data-intensive. These applications may be geo-distributed in order to provide geographically available services around the globe. In this context, cloud storage systems rely on geo-replication. Geo-replication allows storage systems designers to provide lower latency (fast access) through reading from replicas located in a close geographical area, geo-availability by replicating data in different geographical distant zones, fault-tolerance and disaster recovery from catastrophes that may hit one or more areas. However, geo-replication introduces the data consistency issue. Traditional strong consistency by the mean of synchronous replication may present a real bottleneck for system availability and performance due to the network latencies between wide areas. For instance, the cost of a single hour of downtime for a system doing credit card sales authorizations has been estimated to be between $2.2M-3.1$M [1]. As to overcome such limitations, many service providers rely on eventually consistent storage systems. Eventual consistency [2] may tolerate stale data to be read at some points in time, but ensures that all data will *eventually* become consistent. Eventually consistent systems such as Amazon Dynamo [3], and Cassandra [4], provide a massively scalable storage support while maintaining performance with very high availability for services like Amazon.com platform and Facebook social network. By avoiding synchronous replication, these systems tend to consume less resources in the cloud. Subsequently, they reduce the service monetary cost. However, this efficiency does come at the price of exposing stale data to the storage system users. In [5], it has been shown that under heavy data access, 66% of the read data may be stale. Many administrators as well as users would consider this rate as an unacceptable rate even if it provides better performance and costs less money.

In this PhD research, we focus on data consistency management in the cloud to provide systems that are:

- *Self-adaptive*: automatically manage consistency at runtime in order to provide better performance and availability without violating application requirements.
- *Cost-efficient*: Since cloud computing is an economy-driven model, the monetary cost should be considered when selecting the consistency level at runtime.
- *Application-specific*: Applications are different, thus mechanisms that apprehend their requirements are important to reach efficient consistency management.

II. Related Work

Consistency management and its impact on different storage system features, such as performance and availability, was widely studied. One particular solution that gained major popularity in the context of cloud computing is Eventual Consistency [2]. Systems such as Amazon Dynamo [3] and Cassandra [4] are respectively, the key for Amazon.com and Facebook large scale services' availability and fast accesses. A fair number of studies investigated the provided consistency in the cloud. *Wada et al.* [5] measure the actual consistency provided to consumers in cloud platforms. They investigate the consistency and performance properties as well as the monetary cost offered by various cloud providers.

* Advisers: Gabriel Antoniu, INRIA, Rennes, France, gabriel.antoniu@inria.fr; María S. Pérez, Universidad politécnica de Madrid, Madrid, Spain, mperez@fi.upm.es

In [6], the authors propose an offline consistency verification algorithm. Their algorithm checks three consistency semantics properties (safety, regularity, and atomicity) of data accesses in the key-value store.

Since static predefined approaches deal poorly with cloud workloads dynamicity, dynamic adaptive consistency policies were introduced. *Kraska et al.* [7] propose a flexible consistency management that adaptively alternates between strong and weak consistency. In their model, inconsistencies are the results of update conflicts. Accordingly, they compute the probability of an update conflict. Based on this probability, and according to a threshold, they select either serializability as strong consistency or session consistency, which is a weaker consistency. However, their approach does not consider the staleness in the eventual consistency model. The staleness is due to the update propagation latency, while in their model, the inconsistency is due to the conflict of two or more updates on different replicas. Additionally, their threshold computation relies on the financial cost of pending update queues and not related to the cost of the storage backend itself nor the application explicit requirements. *Wang et al.* [8] propose an application-based adaptive mechanism of replicas consistency. Their approach was designed for their specific replication architecture. The architecture consists of multiple primary replicas and read-only secondary replicas. The proposed mechanism selects either strong or eventual consistency based on the comparison of the read rate and the write rate to a threshold. The main limitation of this work is the arbitrary choice of a static threshold. Moreover, the adaptive approach is restricted to their replication architecture, which is not commonly used. In contrast to the aforementioned work, our proposed approaches rely on the application consistency requirements and the storage system state in order to provide gradually stronger consistency when it is needed. Moreover, the monetary cost of the residing storage system in the cloud itself is considered in order to provide a cost-efficient model.

III. Efficient Consistency management

The core work of this PhD research consists of three main contributions.

A. Automated Self-Adaptive Consistency for cloud storage

We propose a novel approach [9], named *Harmony*, that automatically tunes the consistency level at runtime according to the application requirements. Harmony monitors the storage system and data accesses in order to estimate the stale reads rate in the system. Accordingly, it scales up/down the consistency level to preserve a stale rate tolerated by the application. Meanwhile, performance and availability are favored as long as the application requirements are not violated.

On a higher level, Harmony relies on a simple algorithm that compares the estimated stale reads rate in the system

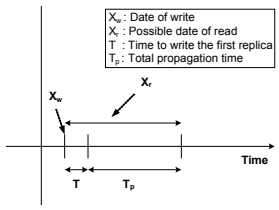

Figure 1. Situation that leads to a stale read: *The read may be stale if its starting time Xr is in the time interval between the starting time of the last write Xw and the end of data propagation time to the other replicas Tp.*

to the application tolerated stale reads rate. Accordingly, it chooses whether to select the basic consistency level ONE (involving only one replica) or else, computes the number of involved replicas necessary to maintain an acceptable stale reads rate while allowing better performance.

In order to estimate the stale reads rate in the system, Harmony embraces an estimation model based on probabilistic computations. The situation that leads to a stale read is defined as shown in Figure 1. A read may be stale if its starting time occurs when data is being propagated to one or more replicas. Based on this situation, we compute the probability of a stale read considering key information such as read and write mean arrivals rates and average data propagation time.

Harmony can be applied to different cloud storage systems that are featured with flexible consistency rules. In our current implementation Harmony operates on top of Apache Cassandra storage and consists of two modules. The monitoring module collects relevant metrics about data access in the storage system : read rates and write rates, as well as network latencies. These data are further fed to the adaptive consistency module. This module is the heart of the Harmony implementation where the estimation and the resulting consistency level computations are performed.

B. Cost-Efficient Consistency

While most optimization efforts focus on consistency - performance and/or availability tradeoffs, a little work investigated the impact of consistency on monetary cost in the economy-driven paradigm of cloud computing. In this work [10], we first investigate the bill details of running storage service in the cloud considering various consistency levels. We provide an in-depth understanding of the monetary cost of cloud services with respect to their adopted consistency models. We discuss the different resources contributed to the service and the cost of these resources. Therefore, we introduce an accurate decomposition of the total bill of the services into three parts for these resources: VM instances cost, storage cost and network cost.

In order to have a better understanding of the tight relation between monetary cost and consistency, we introduce a new metric, *consistency-cost efficiency*, to evaluate consistency in the cloud from an economical point of view. Based on our metric, we introduce a simple yet efficient approach

named *Bismar*, which adaptively tunes the consistency level at run-time to reduce the monetary cost while simultaneously maintaining a low fraction of stale reads. *Bismar* relies on a relative computation of the expected cost and probabilistic estimation of consistency in the cloud. At runtime, the consistency level with the highest consistency-cost efficiency value is always chosen.

C. Customized Consistency by means of Application Behavior Modeling

Applications consistency requirements are different. A webshop application for instance requires a stronger consistency as reading stale data could, in many cases, lead to serious consequences and a probable loss of client trust and/or money. A social network application on the other hand, requires a less strict consistency as reading stale data has less disastrous consequences. Understanding such requirements only at the level of the storage system is not possible. In this work, and in contrast to related work, we focus on the application level. We argue that in order to fully understand the application and its consistency requirements, such a step is necessary. Therefore, we introduce an approach to provide a customized consistency specific to the application. In order to build a customized consistency, we introduce a modeling process for application data access behavior. This is an offline process that consists of several steps. First, several predefined metrics are collected based on application data access past traces. These metrics are collected per time period in order to build the application *timeline*. This timeline is further processed by machine learning techniques in order to identify the different states and states evolvements of the application during its lifetime. Each state is then automatically associated with a consistency policy (policies include geographical policies, Harmony, and static eventual and strong policies) based on a set of both generic predefined rules and customized rules (integrated by application' administrator) specific for the application. At runtime, the application state is identified by the application classifier and accordingly, it chooses the consistency policy associated with that state. The experimental evaluations of this approach are part of future plans.

IV. RESULTS

In order to validate our approaches, we conducted a set of experimental evaluations on two main platforms: Amazon Elastic Cloud Compute (EC2) [11], and the french cloud and grid testbed Grid'5000 [12] that consists of 10 sites in France and Luxembourg. For all experiments we ran Apache Cassandra as an underlying storage system, and used the Yahoo Cloud Serving Benchmark! (YCSB) [13].

A. Performance/Staleness evaluation in Harmony

We deployed Cassandra on 20 VMs on Amazon EC2, and 84 nodes on two different clusters in Grid'5000. The goal of these experiments is to evaluate system performance and measure staleness rate. We used a heavy read-update workload from YCSB! with two data sets of size 23.85 GB for EC2, and 14.3 GB for Grid'5000. The workload consisted of 5 million operations for Amazon EC2 and 3 million operations for Grid'5000. We compare Harmony with two different tolerable stale read rates (20% and 40% for Grid'5000, and 40% and 60% for EC2) with static strong and eventual consistency approaches on our both platforms (Grid5000 and Amazon EC2). Results show that Harmony reduces the read stale data when compared to weak consistency by almost 80% while adding minimal latency. Meanwhile, it improves the throughput of the system by up to 45% while maintaining the desired consistency requirements of the applications when compared to the strong consistency model in Cassandra.

B. Consistency-Cost Efficiency

As to investigate the consistency cost in the cloud, we conducted two separate sets of experiments. First, we wanted to build a deeper understanding picture of the consistency impact on monetary cost in the cloud. After that, experiments to evaluate our cost-efficient consistency model Bismar were conducted. In all experiments we used a heavy read-update YCSB workload that consists of 10 million operations and a total data size of 23.84 GB. Apache Cassandra was deployed with a replication factor of 5 on two availability zones (datacenters) in the *us-east-1* region in Amazon EC2 with a total of 18 VMS for the first set of experiments, and two sites in the east and the south of France in Grid'5000 with a total of 50 nodes for both sets of experiments.

- **Consistency impact on monetary cost**. Results show the total monetary cost decreases when degrading the consistency level. We observed down to 48% of cost reduction with weaker consistency. This was an expected result as lower consistency levels use less resources and lower the operations latencies. This cost reduction, however, is associated with a significant increase in the stale reads rate. For instance we observed that only 21% of reads are estimated to be up-to-date when the consistency level is the lowest (level ONE). Obviously, one of the most efficient consistency levels is the level Quorum. This level returns always an up-to-date replica (which is always included in the quorum) but reduces the cost of the strong consistency level by 13%.
- **Bismar evaluation**. In order to validate our efficiency metric, we collect samples when running the same workload with different access patterns and different consistency levels. Results show that the most efficient consistency levels are the ones that provide a staleness rate smaller than 20%. This demonstrates the effectiveness of our metric where lower levels are efficient only when they provide an acceptable consistency. Experiments on Bismar show that only the consistency level

ONE costs less. This level (ONE) however, tolerates up to 61% of stale reads. Our approach Bismar achieves up to 31% of cost reduction compared to the static level Quorum which is one of the most efficient approaches. Meanwhile, it only tolerates 3.5% of stale reads which is acceptable for a large class of applications that do not require strictly strong consistency.

V. Future Plans

As part of a future plan we intend to investigate three directions as to provide better consistency management for cloud storage clients. The primary direction targets the investigation of power consumption behavior of different consistency approaches. We plan to conduct an in-depth study that analyzes power consumption and resources usage (Cpu, Memory, Disk access ...) of the whole storage system considering different consistency levels and various cpu frequencies and governors. As a result, we intend to build a new approach to improve the power-efficiency of storage systems while maintaining the application consistency requirements.

In a different direction, we intend to provide a cost-efficient storage provisioning in the cloud under consistency, performance and failures constraints. One of the main advantages of cloud computing is its flexible resource leasing in a pay as you use way. In this work, we plan to provide an efficient mechanism, that considers application and environment constraints such as the level of consistency or the presence of failing nodes. Accordingly, the quantity of additional storage nodes that reduce the bill is computed.

Current eventually consistent systems do not give any guarantees on when all replicas in the storage system would converge to a consistent state. This could be a real problem for many applications as there is no provided certainty about read data. Moreover, if the read data is stale, the question is how stale it could be. In this future work, we plan to design and build an eventually consistent system prototype that provides guarantees on the freshness of data read and ensures that data is consistent after a set of defined deadlines. Furthermore, the proposed system will introduce different levels of guarantees considering the network performance and topology in addition to data location.

VI. Conclusion

Nowadays, it is easy to rely on multiple distant datacenters in the cloud to provide geographically available services by means of geo-replication. However, synchronous geo-replication that ensures strong consistency can impose a real bottleneck. Storage systems' performance and availability, in addition to monetary cost, may suffer from the high wide area network latencies. On the other hand static weaker consistency models such as eventual consistency, may result in a very high stale data being read. In this work, we introduced an efficient consistency management in the cloud. The first step towards this goal, was to introduce a self-adaptive approach, called Harmony, that automatically tunes the consistency level at runtime according to the storage system state and the application specified requirements. Harmony provides a better performance without exceeding the application tolerated rate of staleness. Moreover, we proposed the Bismar approach that considers the monetary cost when selecting consistency in the cloud. Bismar investigates the efficiency of all consistency levels and always selects the level with the highest cost-consistency efficiency at runtime. To complete our consistency management proposal, we presented an application modeling approach that studies the data access behavior and learns the consistency requirements in the aim of providing an application specific consistency.

References

[1] R. Peglar, "Eliminating planned downtime: the real impact and how to avoid it," May 2012. [Online]. Available: http://findarticles.com/p/articles/mi_m0BRZ/is_5_24/ai_n6095515/

[2] W. Vogels, "Eventually consistent," *Commun. ACM*, pp. 40–44, 2009.

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," in *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, ser. SOSP '07. New York, NY, USA: ACM, 2007, pp. 205–220.

[4] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, pp. 35–40, April 2010.

[5] H. Wada, A. Fekete, L. Zhao, K. Lee, and A. Liu, "Data consistency properties and the trade-offs in commercial cloud storage: the consumers' perspective," in *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings*, 2011, pp. 134–143.

[6] E. Anderson, X. Li, M. A. Shah, J. Tucek, and J. J. Wylie, "What consistency does your key-value store actually provide?" in *Proceedings of the Sixth international conference on Hot topics in system dependability*, ser. HotDep'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–16.

[7] T. Kraska, M. Hentschel, G. Alonso, and D. Kossmann, "Consistency rationing in the cloud: pay only when it matters," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 253–264, Aug. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1687627.1687657

[8] X. Wang, S. Yang, S. Wang, X. Niu, and J. Xu, "An application-based adaptive replica consistency for cloud storage," in *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, nov. 2010, pp. 13 –17.

[9] H.-E. Chihoub, S. Ibrahim, G. Antoniu, and M. S. Pérez-Hernández, "Harmony: Towards automated self-adaptive consistency in cloud storage," in *2012 IEEE International Conference on Cluster Computing (CLUSTER'12)*, Beijing, China, 2012, pp. 293–301.

[10] H.-E. Chihoub, S. Ibrahim, G. Antoniu, and M. S. Pérez, "Consistency in the cloud: When money does matter!" in *Technical Report*. [Online]. Available: http://hal.inria.fr/hal-00756314/

[11] "Amazon Elastic Compute Cloud (Amazon EC2)," November 2012. [Online]. Available: http://aws.amazon.com/ec2/

[12] Y. Jégou, S. Lantéri, J. Leduc *et al.*, "Grid'5000: a large scale and highly reconfigurable experimental grid testbed." *Intl. Journal of High Performance Comp. Applications*, vol. 20, no. 4, pp. 481–494, 2006.

[13] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM symposium on Cloud computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 143–154.