Adaptive consistency models in the cloud: review and comparative study

Abdennacer Khelaifa, Saber Benharzallah, Laid Kahloul

Abstract—

NoSQL storage systems are used extensively by Web applications and provide an attractive alternative to conventional databases due to their high security and availability with a low cost. High data availability is achieved by replicating data in different servers in order to reduce access time lag, network bandwidth consumption and system unreliability. In this context, one major challenge is data consistency. In one hand, strong consistency guarantees data freshness but affects directly the performance and availability of the system. In the other hand, weaker consistency enhances availability and performance but increases data staleness. Therefore, an adaptive consistency strategy is needed to tune, during runtime, the consistency level depending on the criticality of the requests or data items. Although there is a rich literature on adaptive consistency approaches in cloud storage, there is a need to classify as well as regroup the approaches based on their strategies. This paper will establish a set of comparative criteria and then make a comparative analysis of existing adaptive consistency approaches. A survey of this kind gives an insight to the user/researcher about not only a comparative analysis of performance of approaches but also suggests suitability of these approaches for candidate cloud systems.

Index Terms—Cloud storage, Big data, adaptive consistency, adaptive policy.

1 Introduction

With the fast development of processing and storage technologies and the success of the Internet, computing resources have become cheaper, more powerful and more available than ever before. This technological trend has enabled the realization of a new computing paradigm called cloud computing [1]. A client of the cloud can lease just the resources he need on in a Pay-as-You-Go manner [2] with very little knowledge of the physical resources. Nowadays cloud computing is the best alternative of grid and cluster computing because it performs well with data-intensive applications [3][4] companies like Google, Amazon, and Facebook deal with peta- and terabytes of data every day. In this context, storage management and performance within clouds is extremely important.

In cloud storage system, replica technology [5] is a key technology in enhancing performance of the system. With the assistance of replica, the distributed file system is able to reduce access time lag, network bandwidth consumption and system unreliability. However, this also leads to a problem in replica consistency management. Namely when data is accessed and read by multiple users, inconsistency occurs in replicas, and as a result the system's consistency and accuracy are directly influenced. The study on replica consistency aims to achieving synchronism among multiple replicas. The most popular strategies provided by storage systems are strong consistency and eventual consistency [6]. Strong consistency ensures all of the replicas to be updated immediately. There is no difference between replicas. Therefore, every access to replicas will get fresh data. But the maintaining cost increases significantly, which cuts down the availability of replicas and system's performance. For instance, Google BigTable [7], Microsoft Azure Storage [8] and Apache HBase [9] provide strong consistency. Eventual consistency doesn't update all the replicas immediately, so it tolerates replica divergence. But it promises all the replicas to be consistent at a specific time. For instance, Amazon Dynamo [10], Cassandra [11] and MongoDB [12] provide eventual consistency.

Providing one consistency strategy is only suitable for particular scenes because, the clients of cloud storage are multifarious and not all the applications need the same level of consistency. In addition, required consistency strategy of an application is variable at runtime. Take online conference as an example [13]. the data should be modified under strong consistency when an important speech is taking place. Otherwise, users accept eventual consistency for better performance. In this context, a self-adaptive approach is needed to dynamically adjust the consistency strategy according to the cloud system dynamicity and the application's demands. Therefore, adaptive replica consistency can satisfy the application requirements and minimize the transaction cost at the same time.

Many works in the literature addressed the adaptive consistency and the tradeoff between consistency and availability [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24]. The proposed approaches in these works used different protocols to provide adaptive consistency. Most of them defined a metric such as: read/write frequency [17], file heat [19], stale reads [14], etc. and used statistical or probabilistic model to calculate it. Then, the system changes the level of consistency when the calculated value exceeds the defined

- *A. Khelaifa was with the Department of Computer science, University of El-oued, 39000.*
  E-mail: abdennacer-khelaifa@univ-eloued.dz
- *S. Benharzallah was with the Department of Computer science, University of Batna 2, 05000.*
- *L. Kahloul was with the Department of Computer science, University of Biskra, 7000.*

threshold. Some approaches took monetary cost into consideration when they defined their metrics however, others used version vector for every copy of data to choose the suitable consistency for every object. To develop a robuste adaptive protocole it is necessary to make a comparative study of the proposed approaches by taking in consideration all the elements around consistency in the cloud. All the contributions that we have found described few proposed approaches as related works but no one, in our knowledge, established an exhaustive analysis for the existing adaptive consistency approaches. In our current work, we evaluate the most popular adaptive techniques in cloud system using a set of criteria proposed for this purpose. Thus, this paper provides a comparative study between the proposed approaches based on different issues: such as granularity of consistency strategy, conflict detection and resolution, monetary cost consideration, threshold definition and adaptive policy.

This paper is organized as follow. Section II briefly defines the principal concepts and types of consistency. Section III presents the comparison criteria and explains the utility of each one. Section IV gives the comparative analysis of the proposed approaches according to the above criteria. Section V discusses related works. Finally, section VII presents our conclusions and perspectives.

2 Consistency in the cloud

Consistency concepts and its relationship with different storage system features, such as performance and scalability, was widely addressed. We first review consistency definition in traditional database systems. Then, we review its definition in distributed systems including cloud systems. Finally, we review the consistency models and classifications.

2.1 Consistency in database systems

In traditional database systems, consistency defined as a property of transactions [25][26]. It builds with Atomicity, Isolation and Durability the well-known acronym: ACID properties. Consistency refers to the fact that the transaction takes the system from one consistent state to another. Note that a transaction may violate some of the integrity constraints during its execution. However, once it terminates, it must restore the system to a consistent state. When transactions are executed concurrently, the transaction processing system must ensure that the execution of a set of concurrent and correct transactions also maintains the consistency of the data. Atomicity, requires all the operations of a transaction to be treated as a single unit; hence, everything in a transaction succeeds or the entire transaction is rolled back. Isolation refers to the fact that transactions cannot interfere with each other, i.e., not to read the intermediate results of other transactions. Finally, durability requires the results of a committed transaction to be made permanent in the system.

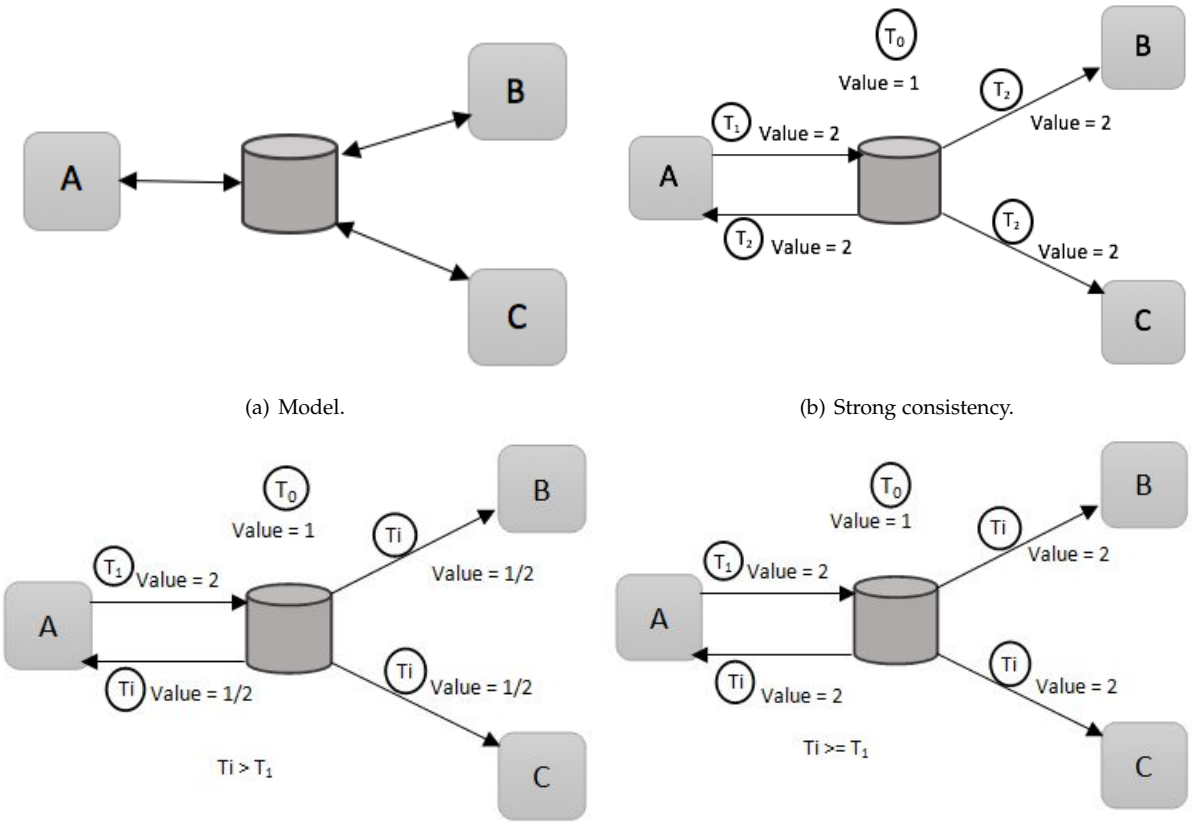
2.2 Consistency in distributed systems

In distributed systems, consistency was defined in a trade-off with availability and partition tolerance in the CAP theorem [27][28]. The theorem states that only two out of the three properties can be achieved simultaneously within a distributed system. In this context, consistency refers to that the clients should have a feeling of working on a single node regardless of the number of replicas. This is equivalent to requiring a total order on all operations and operations act as they are executing on a single node. Availability means that every request sent by a client to a non-failing node should obtain a successful response and Partition Tolerance: The system should continue delivering its services even if some part of the system loses many messages arbitrarily and only the total network failure is allowed to cause the system to respond incorrectly.

2.3 Consistency models

In this sub-section, we introduce the main consistency models adopted in earlier single- site storage systems and in current geo-replicated systems. Many works addressed this consistency models such as [29], [30] and [31]. They call the highest level: strong consistency, and the lowest level weak consistency. Between strong and weak consistency, they define other models that provide better performance than strong consistency and lower conflicts than weak consistency. In the following, we adopt the classification of Werner Vogels [29] due to its powerful discrimination and characterization of models. He states that there are two ways to view consistency. The first is from the client point of view: how the client observe writing operations. The second point of view is from the server side: how the system manages updates and which guarantees are provided with respect to updates.

1) *Client-Side Consistency*: this kind of consistency investigates in how the client observes the changes. For Instance, let's assume: There are three independent processes: A, B and C that they need to communicate to share information as shown in Figure 1(a). When the process A updates a data item, there are three possible cases:
   - *Strong consistency*: An access by any process, after the update completes, will return the recent data (Figure 1(b)). In other word, Strong consistency guarantees that all replicas are in a consistent state immediately after an update.
   - *Weak Consistency*: The system does not guarantee that subsequent accesses will return the recent data (Figure 1(c)). The period between the write operation and the moment when it is guaranteed that any process can see the recent data is called the inconsistency window.
   - *Eventual Consistency*: After a window time, the storage system guarantees that all processes will see the recent data if no new updates are made to the object (Figure 1(d)). This is a specific form of weak consistency and the size of the inconsistency window can be determined based on factors such as the load on the system, communication delays and the number of replicas in the system.



(a) Model. (b) Strong consistency.

(c) Weak consistency. (d) Eventual consistency.

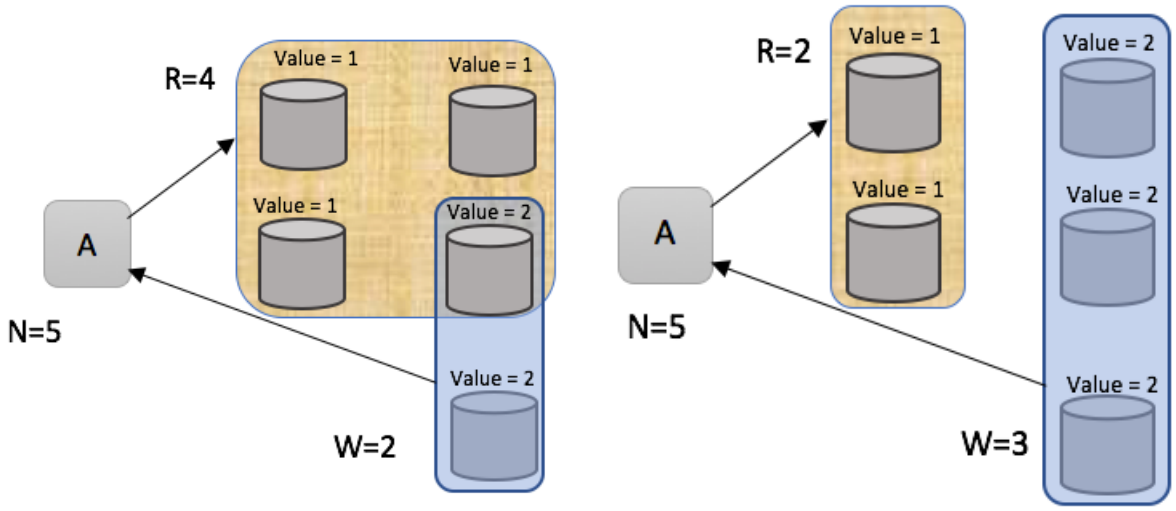
Fig. 1. Client-Side Consistency.

2) *Server-Side Consistency*: On the server-side, the consistency deals with how the updates flow through the system to differentiate the modes that can be experienced by application developers. For Instance, let's assume: N be the number of replicas in the system, W be the number of replicas involved in write/update operation and R be the number of replicas that are contacted when a data object is accessed through a read operation.

- *Strong consistency*: if $W + R > N$, the System will provide strong consistency and two cases are possible in this formula:
  - *Case 1*: If the number of replicas is 5 (N = 5) and the number of responses required to complete write queries (W = 5), it is sufficient to read from one replica (R=1) to get the most recent data $W(5) + R(1) > N(5)$.
  - *Case 2*: If the number of replicas is 5 (N = 5) and the client requires just two replicas to response to his write queries (W = 2). As shown in Figure 2(a) If he reads from more than three replicas during a read operation $(R > 3)$, then there will be at least one replica that will give the most recent data and the system will still provide strong consistency $W + R > N$.
- *Weak consistency*: If $W + R * *N$, the system provides Weak/eventual consistency. This means that there is small number of replicas that are guaranteed to have the latest write and during read operation, it is less likely to read from a replica that has the latest write (Figure 2(b)).

2.4 Adaptive consistency

In modern database systems, strong consistency generates always an additional cost on request latency, availability and scalability of the system. In order to find a tradeoff between consistency and both performance and availability of the system, most of the modern database systems guarantee eventual consistency by default and allow increasing the level of consistency according to client needs. increasing the level of consistency ensures better data consistency but deceases the system performance and availability. Therefore, instead of relying on a single consistency level, tuning the consistency level with other supplementary consistency options makes the system more proficient. This phenomenon of using the appropriate consistency option depending on the criticality of the requests or data items is known as Adaptive Consistency [31].



(a) Strong consistency.

(b) Weak consistency.

Fig. 2. Server-Side Consistency.

3 Comparison Criteria of Adaptive Consistency Approaches

When talking about adaptive consistency we distinguish numerous criteria have to be described, among these issues we are interested to the following ones: adaptive policy, conflicts consideration, granularity, operation level, prediction or statistic based, monetary costs and threshold. These criteria have a direct influence on the development of adaptive consistency approaches.

3.1 Adaptive policy

A set of algorithms, models or techniques used to provide adaptive consistency. It captures the system need and give the best tradeoff between consistency and both performance and availability by tuning the consistency to the suitable level during the execution time. These policies are based on probabilistic models, artificial intelligence techniques or any other predictive algorithms. Choosing a good adaptive policy leads to a pertinent adaptive consistency approach.

3.2 Conflicts management

We put this criterion to ask the question: has the approach treated the point of conflict between transactions? And define which mechanism is used to detect and resolve the conflicts. Normally the conflicts between replica occur highly when approach choice an optimistic consistency (the consistency level is low) and vice versa, upgrading the level of consistency decreases the percentage of conflicts.

3.3 Granularity

It means that the consistency level is applied on the hole database or just a part. In fact, among the proposed approaches there are who work on the global database, others fragmentize the database onto categories and there exist who works on file. Applying strong consistency on only a part of the system or critical objects can give higher performance and optimize the cost of implementation.

3.4 Operation level

This criterion defines the side in which the approach is applied and divides the approaches into two categories: query side approaches and object side approaches. the approaches in the first category focus on the query or the transaction level this means that the consistency level in these approaches is tuned according to the application needs. However, the object side approaches define the consistency guarantees on the data by dividing them into categories and treat each category differently depending on the provided consistency level.

3.5 Prediction or statistic based

The adaptive policy may be based on prediction or statistic. Predictive policy uses probabilistic model, regression model or other intelligent techniques to predict the system state. This kind of models is very powerful in performance but it does not give exact results. However, the statistic model calculates the rate of given metric during runtime and compares it to a defined threshold to change the consistency level when it is achieved. the statistic model can give exact results but it produces additional and complicate calculs to the system.

3.6 Monetary costs

The cost of consistency level in the cloud describes the different resources contributed to obtain this level in geo-replicated storage systems. Different consistency levels result in different costs; high consistency implies high cost per transaction and reduced availability but avoids penalty costs. Low consistency leads to lower costs per operation but might result in higher penalty costs. The monetary cost is one of the most interest advantages of cloud storage. Hence, any consistency approach should take this criterion into consideration.

3.7 Threshold

When an adaptive consistency policy uses a metric, it defines a minimum or maximum value for this metric. Reaching the threshold triggers the system to change his behavior. defining a threshold in such approach means that it doesn't calculate the application needs and the tuning parameters are put manually.

4 Proposed approaches for adaptive consistency

In the literature, many approaches were proposed to adjust the consistency level in the cloud. In this section, we discuss the different approaches and their characteristics according to different criteria defined in the previous section.

4.1 Harmony: Automated Self-Adaptive Consistency

Harmony [14] is based on the estimation model of stale reads that will be adjusted to the application needs. In this approach, the application gives her appropriate stale read rate (app_stale_read) and the modules added by Harmony compute the stale read rate of the system (θ) using a probabilistic model and compare it with app_stale_read. If θ is greater, the algorithm calculates the number of replicas (N) needed to attenuate the stale read rate and modify the consistency level according to the obtained N.

4.2 Consistency Rationing

Consistency Rationing [15] divides the data into three consistency categories: A, B, and C. The A category ensures strong consistency guarantees and shows high cost per transaction. The C category ensures session consistency, shows low cost, but will result in inconsistencies. Data in the B category is handled with either strong or session consistency depending on the specified policy. In this paper the authors present and compare many policies to switch consistency guarantees (conflict probability, time policy, fixed threshold, Demarcation policy and Dynamic policy). The optimization in this paper is based on allowing the database to exhibit inconsistencies if it helps to reduce the cost of a transaction but does not cause higher penalty costs. So, try to find a tradeoff between transactions cost in the case of strong consistency and penalty costs (financial) in the case of weak consistency.

4.3 Consistency tuner

consistency tuner is a protocol based on consistency index [16] (number of the correct reads/Total number of the reads). To adjust the consistency index to a desired value the protocol predict the correctness of an incoming read request using logistic regression classifier and neural network classifier. These statistical predictive models use the number of replicas R_p and the time gap G_p between the read and the last update as an input parameters. The authors implement also the CICT (Consistency index based consistency tuner) in an architecture of Web based database application and demonstrate the relationship between the number of replicas and the threshold of time gap (minimum value of time gap between an update and a succeeding read request) using a statistical linear regression analysis.

4.4 An Application-Based Adaptive Replica Consistency

This paper [17] proposed an adaptive mechanism for consistency management that allows system to automatically switch consistency strategies according to update frequency and read frequency at runtime. the authors proposed also a model structure that can manage the different consistency level. The nodes in the proposed structure are put in the following order: one master node, three deputy nodes and many child nodes. The adaptive consistency mechanism divides the consistency level into four categories according to statistical read frequency P_r and update frequency P_w(a combination of high and low values for every one). for each checkpoint interval τ, the system judges the consistency category that the application needs according to P_r and P_w in the interval. If the category is not the same as the current one, it will be changed in the next interval. They calculated the amount of operations in the system's process and demonstrate that it is less than strong consistency cost.

4.5 IDEA

The infrastructure IDEA [18] (an Infrastructure for DEtection-based Adaptive consistency guarantees) was presented to guarantee the adaptive consistency of replicated services. IDEA enables many functions including quick inconsistency detection and resolution, consistency adaptation and quantified consistency level guarantees. For each object of the system, IDEA divides the nodes into two layers where the top layer includes those nodes that frequently update this object and the bottom layer consists of the remaining nodes. The inconsistency detection and resolution policies are based on a new extended version vector where a triple of information is attached for every object (numerical error, order error, staleness).

4.6 File heat-based self-adaptive replica consistency

This work proposed the algorithm MRFU [19] (Most Recent and Frequently Used) to calculate the file heat during an interval of time I. the file heat in this algorithm is a combination of access time and access frequency. The self-adaptive consistency strategy proposed in this paper switches the consistency level of a file between strong and eventual consistency according to the file heat. The value of file heat is calculated during a time interval I and compared with a threshold. the strong consistency strategy is adopted when file heat exceeds the predefined threshold; and the eventual consistency strategy is adopted when file heat is under the threshold value to cut network bandwidth consumption. The authors proposed also a replica management model that divides the system into three levels: one storage control node that controls many main replicas and every main replica is connected to many other sub replicas.

4.7 Fine-tuning the Consistency-Latency Trade-off

The trade-off consistency-latency is addressed by proposing and evaluating two techniques [20] The first, a novel technique called continuous partial quorums (CPQ) that assigns the consistency level on a per- operation basis by choosing randomly between multiple discrete consistency levels with

a tunable probability. The second technique, called artificial delays (AD) uses a weak client-side consistency level and injects an artificial tunable delay into each storage operation. This technique boosts consistency by allowing more time for updates to propagate through the system, which decreases the likelihood of consistency anomalies at the cost of increasing latency.

4.8 Bismar

Bismar [21] take the monetary cost into consideration in the evaluation and the selection of consistency level in the cloud. Accordingly, they define a new metric called consistency-cost efficiency. Based on this metric, they propose an economical consistency model, called Bismar, that adaptively tunes the consistency level at run-time in order to reduce the monetary cost while simultaneously maintaining a low fraction of stale reads. The proposed approach uses a consistency probabilistic model to estimate the stale reads and the relative costs of the application according to the current read/write rate and network latency. Then, the algorithm selects the consistency level that offers the most equitable consistency, cost trade-off (the maximum consistency-cost efficiency value: Max[Consistency(cl)/Cost(cl)]).

4.9 A self-adaptive conflict resolution with flexible consistency guarantee

This paper [22] presents an adaptive and hierarchical model using the versions of vectors among the replicas for consistency management replicas. The proposed approach divides the consistency management into two levels: global management between datacenters in the cloud and local management in the datacenter. For the local management, the multi-agent technology is used to modulate the different parts of the system which tunes the consistency between three levels (Optimistic, Hybrid and Pessimistic) according to the rate of write operations. The authors propose also the integration of a conflict detection and resolution mechanisms between the replicas based on version vectors. These mechanisms resolve the conflict in the local datacenter then resolve the conflicts between different datacenters in the cloud.

4.10 OptCon

OptCon [23] is a machine learning-based framework that can automatically predict a matching consistency level that satisfies the latency and staleness thresholds specified in a given Service Level Agreement (SLA). For this reason, OptCon put the following dynamic parameters as an input variables to the learning algorithms: the read proportion in the operation, the number of user threads spawned by the client, the number of network packets transmitted during the operation in addition to the client-centric consistency level. Many machine learning techniques were implemented in OptCon: To visualize the significance of the model parameters and the dependency relations among these parameters, it used Logistic regression and Bayesian learning. Furthermore, for more accurate predictions computed directly from the data, OptCon implemented Decision Tree, Random Forest, and Artificial Neural Networks (ANN). The framework provides to users and developers the choice of a suitable learning technique that best suits the respective application domain and use case.

4.11 SPECSHIFT

SPECSHIFT [24] is a tuning framework that uses artificial delays to adjust the consistency level according to network conditions and workload characteristics changing. The delay is calculated over a combination between empirical measurement and probabilistic analysis and injected at the beginning of each read and at the end of each write. Developers of this framework presented also a probabilistic model of consistency under latency optimized settings that captures precisely the relationship between consistency, system workload and network latency.

5 Related works

In our knowledge, there is no work that have established an exhaustive analysis for the existing adaptive consistency approaches. Most contributions described few proposed approaches as related works. Only two works made a comparative study [22] and [30].In [22], they presented a literature review that summarized the consistency management in distributed systems, grids and cloud. For each proposed adaptive consistency approaches in the cloud, they gave a little description and criticized it. In [30], they put just three criteria: The level at which the consistency is specified, Cloud Storage system: implemented within and Tested for evaluating the solution. Then, they compare between three works according to mentioned criteria. In contrast to the aforementioned work, our contribution outlines most characteristics for an adaptive consistency approach and then describes briefly and compares the most popular approaches in the cloud.

6 Conclusion

In this paper, we have presented a comparative study between adaptive consistency approaches in terms of conflicts, granularity, adaptive policy, operation level, threshold, and monetary costs. From this comparison, we deduce the following points:

- A statistic policy is very expensive to implement in Big data due to large amount of heterogeneous data. So,using an intelligent technique is better in term of performance and costs.
- Taking monetary costs into consideration is very important in such approaches because it is among the principle goals of using the cloud.
- Applying the same level of consistency for the overall system is not always practical when data in the cloud have not the same importance or the same access frequency.
- It is necessary that the adaptive policy makes a combination between transaction level and object statistics (operation side and data side) to define the consistency guarantees.

Building an adaptive consistency model should use artificial intelligence techniques. We believe that these techniques with their predicting capacity could help to build more optimal consistency approach. Furthermore, dividing the system to categories could ameliorate the system performance by affecting the strong consistency to limited part

of data.

The tradeoff between consistency and performance remains a big challenge in cloud storage. another important challenge is the optimisation of monetary costs by combining between strong consistency costs and penalty costs of weaker consistency level. many researchers tried to resolve these problems by managing the consistency adaptively according to application needs and/or data values.

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing," *Communications of the ACM*, vol. 53, no. 6, p. 50, 2010.

[2] H. Jin, S. Ibrahim, T. Bell, L. Qi, H. Cao, S. Wu, and X. Shi, "Tools and technologies for building clouds," in *Cloud Computing*. Springer, 2010, pp. 3–20.

[3] S. Zhang, X. Chen, S. Zhang, and X. Huo, "The comparison between cloud computing and grid computing," in *2010 International Conference on Computer Application and System Modeling (ICCASM)*, vol. 11. IEEE, 2010, pp. V11–72.

[4] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of several cloud computing platforms," in *2009 Second International Symposium on Information Science and Engineering (ISISE)*. IEEE, 2009, pp. 23–27.

[5] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso, "Understanding replication in databases and distributed systems," in *Proceedings of the 20th International Conference on Distributed Computing Systems, 2000*. IEEE, 2000, pp. 464–474.

[6] H. Stockinger, A. Samar, K. Holtman, B. Allcock, I. Foster, and B. Tierney, "File and object replication in data grids," *Cluster Computing*, vol. 5, no. 3, pp. 305–314, 2002.

[7] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, p. 4, 2008.

[8] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci *et al.*, "Windows azure storage: a highly available cloud storage service with strong consistency," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 143–157.

[9] "Apache hbase," http://hbase.apache.org/, February 2018.

[10] 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," *ACM SIGOPS operating systems review*, vol. 41, no. 6, pp. 205–220, 2007.

[11] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[12] "mongodb," http://www.mongodb.org/, February 2018.

[13] T. Chang, G. Popescu, and C. Codella, "Scalable and efficient update dissemination for distributed interactive applications," in *Proceedings of the 22nd International Conference on Distributed Computing Systems, 2002.* IEEE, 2002, pp. 143–150.

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

[15] T. Kraska, M. Hentschel, G. Alonso, and D. Kossmann, "Consistency rationing in the cloud: pay only when it matters," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 253–264, 2009.

[16] S. P. Phansalkar and A. R. Dani, "Tunable consistency guarantees of selective data consistency model," *Journal of Cloud Computing*, vol. 4, no. 1, p. 13, 2015.

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

[18] Y. Lu, Y. Lu, and H. Jiang, "Adaptive consistency guarantees for large-scale replicated services," in *International Conference on Networking, Architecture, and Storage, 2008. NAS'08.* IEEE, 2008, pp. 89–96.

[19] Z. Zhou, S. Chen, T. Ren, and T. Wu, "File heat-based self-adaptive replica consistency strategy for cloud storage," *J Computers*, vol. 9, no. 8, pp. 1928–1933, 2014.

[20] M. McKenzie, H. Fan, and W. Golab, "Fine-tuning the consistency-latency trade-off in quorum-replicated distributed storage systems," in *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015, pp. 1708–1717.

[21] H.-E. Chihoub, S. Ibrahim, G. Antoniu, and M. S. Perez, "Consistency in the cloud: When money does matter!" in *13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2013*. IEEE, 2013, pp. 352–359.

[22] S. Limam and G. Belalem, "A self-adaptive conflict resolution with flexible consistency guarantee in the cloud computing," *Multiagent and Grid Systems*, vol. 12, no. 3, pp. 217–238, 2016.

[23] S. Sidhanta, W. Golab, S. Mukhopadhyay, and S. Basu, "Adaptable sla-aware consistency tuning for quorum-replicated datastores," *IEEE Transactions on Big Data*, vol. 3, no. 3, pp. 248–261, 2017.

[24] S. Chatterjee and W. Golab, "Self-tuning eventually-consistent data stores," in *International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Springer, 2017, pp. 78–92.

[25] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *CONCURRENCY CONTROL AND RECOVERY IN DATABASE SYSTEMS*. Addison- Wesley, 1987.

[26] A. Elmagarmid, "Database transaction models for advanced applications morgan kaufmann," *San Mateo*, vol. 184, 1995.

[27] E. A. Brewer, "Towards robust distributed systems," in *PODC*, vol. 7, 2000.

[28] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *Acm Sigact News*, vol. 33, no. 2, pp. 51–59, 2002.

[29] W. Vogels, "Eventually consistent," *Communications of the ACM*, vol. 52, no. 1, pp. 40–44, 2009.

[30] H.-E. Chihoub, S. Ibrahim, G. Antoniu, and M. S. Pérez, "Consistency management in cloud storage systems." in *Advances in data processing techniques in the era of Big Data*. CRC PRESS, 2014.

[31] S. P. Kumar, "Adaptive consistency protocols for replicated data in modern storage systems with a high degree of elasticity," Ph.D. dissertation, Conservatoire national des arts et metiers-CNAM, 2016.