A comparative analysis of adaptive consistency approaches in cloud storage

Abdennacer Khelaifa, Saber Benharzallah, Laid Kahloul, Reinhardt Euler, Abdelkader Laouid, Ahcène Bounceur

A comparative analysis of adaptive consistency approaches in cloud storage

Abdennacer Khelaifa[a,b,c,*], Saber Benharzallah[d,b], Laid Kahloul[a,b], Reinhardt Euler[e], Abdelkader Laouid[c,f], Ahcène Bounceur[e]

[a] *University Mohamed Khider of Biskra, 07000 Biskra, Algeria*
[b] *LINFI Laboratory of University Mohamed Khider of Biskra, 07000 Biskra, Algeria*
[c] *University Echahid Hamma Lakhdar of El Oued, 39000 El Oued, Algeria*
[d] *University of Batna 2, 05000 Batna, Algeria*
[e] *Lab-STICC UMR CNRS 6285, Université de Bretagne Occidentale, 20 Avenue Le Gorgeu, 29238 Brest, France*
[f] *LIMED Laboratory, University of Bejaia, 06000 Bejaia, Algeria*

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. Hence, the data consistency is a major challenge in distributed systems. In this context, strong consistency guarantees data freshness but affects directly the performance and availability of the system. In contrast, 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 not only provides the user/researcher with a comparative performance analysis of the approaches but also clarifies the suitability of these for candidate cloud systems.

Keywords: 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 needs in a Pay-as-You-Go manner [2] with very little knowledge of the

*Corresponding author
Email address: abdennacer-khelaifa@univ-eloued.dz (Abdennacer Khelaifa)

physical resources. Nowadays, cloud computing is the best alternative to grid and cluster computing because it performs well with data-intensive applications [3, 4], and 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 distributed systems, replica technology [5] is a key technology in enhancing performance of the system. With the assistance of replica and redundancy of data in several domains [6–11], the distributed file system is able to reduce access time lag, network bandwidth consumption and system unreliability. However, this also leads to the problem of 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 achieve synchronism among multiple replicas. The most popular strategies provided by storage systems are strong consistency and eventual consistency [12]. 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 cost of maintaining the freshness of replicas increases significantly, which cuts down the availability of replicas and the system's performance. For instance, Google BigTable [13], Microsoft Azure Storage [14] and Apache HBase [15] 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 [16], Cassandra [17] and MongoDB [18] provide eventual consistency.

The data consistency can be encountered in cloud systems more than in non-cloud systems [19]. V.Cheng and G.Wills define in [19] a model to compare cloud and non-cloud storage of big data. One of their goal was to show the consistency between the actual and expected execution time on cloud and non-cloud systems. The paper confirms that there is no comparable consistency on a non-cloud system. Therefore, in this comparative study we will only focus on data consistency in cloud storage systems.

Providing one consistency strategy is only suitable for particular scenes since the clients of cloud storage are multifarious and not all the applications need the same level of consistency. In addition, a required consistency strategy of an application is variable at runtime. Take an online conference as an example [20], in which 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's 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 trade-off between consistency and availability [21–39]. The proposed approaches have used different protocols to provide adaptive consistency. Most of them defined a metric such as: read/write frequency [23], file heat [29], stale reads [25], etc. and used statistical or probabilistic models to calculate it. Hereafter, the system changes the level of

consistency when the calculated value exceeds the defined threshold. Some approaches took monetary cost into consideration when they defined their metrics. However, others used version vectors [40] (informations about update history on the data item) for every copy of data to choose the suitable consistency for every object. To develop a robust adaptive protocol, it is necessary to make a comparative study of the proposed approaches by taking into consideration all the elements around consistency in the cloud. All contributions that we found described few proposed approaches as related works, but none, to 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 systems using a set of criteria proposed for this purpose. To achieve our goal, we first give the main concepts of consistency in distributed systems including adaptive consistency. Then, we define a set of criteria that describe the design or the behavior of such an approach like: architectural model, operation level, granularity of consistency strategy, adaptive policy, statistical based or predictive policy, conflict detection and resolution, monetary cost consideration, threshold definition, security and privacy and implementation tools. Thereafter, we review 19 proposed works by describing their main contributions and then resuming their behaviors towards the defined criteria in a table. The table-based analysis gives us a global view and allows us to discuss different aspects of adaptive consistency. We can resume the main results of our study in the following points:

- A middleware is the suitable architectural model to add the adaptive consistency functionality to an existing storage system and enhance its consistency management strategies.
- A statistical policy is very expensive to implement in big data due to large amount of heterogeneous data. So, using an intelligent technique is better in terms of performance and costs.
- Taking monetary costs into consideration is very important in such approaches because it is one the principal 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.

The remainder of this article is as follows. Section 2 describes the related literature. Section 3 briefly defines the principal concepts and types of consistency. Section 4 presents the comparison criteria and explains the utility of each one. Section 5 reviews the proposed approaches and describes its contributions. Section 6 presents a table-based analysis of the proposed approaches according to the above criteria. Section 7 discusses the different points of comparison. Section 8 sets out a number of challenges and future directions that will need to be addressed. Finally, Section 9 concludes the paper.

2. Related work

What we will discuss in this section are the main contributions of this paper compared with the most relevant existing comparative studies by noting all considered elements around consistency in the cloud. Almost all of the comparative contributions that we have found on this topic describe just some proposed approaches as related work. On the other hand, we note that, in the literature, there is a lack of deep comparative analysis establishing an exhaustive analysis of the principal existing adaptive consistency approaches. Hence, this section looks to evaluate the best known adaptive techniques in cloud systems by using a set of criteria and measures for this purpose. In each comparative study between the proposed approaches we will focus on these main issues: granularity of consistency strategy, conflict detection and resolution, monetary cost consideration, threshold definition and adaptive policy.

Prior surveys [41–43] have covered different aspects of the adaptive consistency approaches in cloud storage. In particular, surveys published by *H-E Chihoub* et al. [26, 44] over the last few years elaborated on various topics within the consistency scope such as the concept, benefits and models [26, 44], the architecture elements and the design of the security challenges in adaptive consistency [41]. During the reviewing of some comparative analysis topics in some specific sections, we have noted that none of these surveys has particularly focused on an exhaustive analysis of the existing adaptive consistency approaches. While the data duplication may be implemented using the existing distributed adaptive consistency approaches, their great number along with their particular pros and cons made the choice extremely difficult for those who attempted to adopt a distributed data duplication architecture in a large-scale context. In order to assist and promote recent initiatives to put into practice the adaptive consistency paradigm, this survey proposes original classifications that make comparisons between the broad range of proposed platform solutions with respect to various availability, reliability and performance criteria.

In this context and in line with the vision of Shin et al. [41], three main key reasons for the problems studied in the management can be summarized as follows:

1. For the goal of high availability of the stored data, data replication across storage entities in an efficient way is used. However, these schemes may mitigate data lock-in mainly in case of the appearance of new data stores. Hence, replication is a solution that offers availability. In fact, this solution prevents other functionalities such as data lock-in, data protection and data erasure coding.
2. In case that data-intensive applications are deployed within a single data store, the problems of data overflow and network bottlenecks have appeared. The survey of Shin et al. shows some proposed solutions to solve this problem.
3. Data security is one of the strongest factors in cloud storage and data duplication. It concerns the implemented hardware, users' access and the designed architecture. Many parameters may be changed when the data security plays its role such as data availability, data privacy, data location and data

storage size. To obtain secured distributed data in the cloud storage the authors have referred to many solutions.

In the survey [43], data consistency was tackled, but not as a primordial factor. The authors deal with the data management and they tackle the data consistency only in one section by defining the three main aspects, i.e., level, metric, and model. In the data consistency section, the authors show the taxonomy of data consistency by focusing on these three main aspects in order to determine how much a consistency is stronger than another. The authors explain in detail the I/O-level consistency which requires conservative assumptions like concurrent write-write and read-write upon the same data. This survey focuses mainly on this level of consistency. Furthermore, the authors of [43] have drawn a weak consistency taxonomy together with the influenced factors to encounter the data conflicts which happen as operations to the same data in multi-master systems. In fact, this survey studies the data storage management in cloud environments. The consistency was discussed as a secondary issue. Hence, this work is a global survey which requires the design of a common data model and standard APIs for different cloud databases to help application providers. To reach this end, the authors assume that the ideal solutions for this challenge are to complete data transfer within a budget and deadline especially for OLTP applications that demand high response time.

As another work, Viotti and Marko [42] defined a structured and comprehensive overview of different consistency notions that appeared in distributed systems research. Their work scope was restricted to non-transactional semantics that apply to single storage object operations. The paper aims to complete the existing surveys done in the context of transactional database consistency semantics. The authors define their own non-transactional consistency model of a distributed system for reasoning about different consistency semantics. In fact, this work gives a deep knowledge about consistency for non-transactional systems, called linearizability. The designed authors's hierarchy of non-transactional consistency models shows six main component models: Fork based, session, causal, per-object, synchronized and staleness-based models. In all component models, the authors focus on the explanation of the model's use mode and a discussion of the principal existing work for these component models. We observe that this survey maps the consistency semantics to different practical systems and research prototypes. Therefore, the contribution of Viotti and Marko is to classify the existing work of consistency in non-transaction distributed systems depending on the proposed hierarchy. However, it does not include a comparative study of the existing work.

2.1. The main contributions of this comparative analysis

This paper is to fill this void by drawing a synthesized comparative study of 19 proposed consistency approaches for cloud storage. We investigate the consistency in the cloud and take a strong overview of each analyzed consistency approach. Hereafter, the comparison of the different approaches proposed for adaptive consistency in the cloud is shown and discussed. We note that, to our knowledge, there is no

work having established an exhaustive analysis for the existing adaptive consistency approaches. We may cite the two comparative studies in [32] and [44]. In [32], the authors presented a literature review that summarized the consistency management in distributed systems, grids and the cloud. For each proposed adaptive consistency approach in the cloud, they gave a short description and criticized it. In [44], the authors defined three comparison criteria: the level at which the consistency is specified, the cloud storage system in which the approach is implemented and the testbed which is used to evaluate the solution. Then, they compared the three approaches according to mentioned criteria.

3. Consistency in the cloud

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

3.1. Consistency in database systems

In traditional database systems, consistency is defined as a property of transactions [45, 46]. 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., transactions cannot read the intermediate results of other transactions. Finally, durability requires the results of a committed transaction to be made permanent in the system.

3.2. Consistency in distributed systems

In distributed systems, consistency is defined in a trade-off with availability and partition tolerance in the CAP theorem [47, 48]. The theorem states that only two of the three properties can be achieved simultaneously within a distributed system. In this context, consistency refers to the requirement that the clients should have the 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. Partition tolerance means that 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.

3.3. The consistency's use areas and benefits

Big data storage and processing are considered as one of the main applications for cloud distributed systems. Additionally, the concept of the Internet of Things (IoT) paradigm has advanced the research on Node to Node communications and enabled novel tele-monitoring architectures for several applications such as E-Health, Forest monitoring and others. However, there is a need for safe data storing in decentralized cloud systems and IoT systems. The purpose of this subsection is to show the gathered benefits of consistency by citing some existing work and some methods of big data processing within cloud distributed systems.

Nowadays, cloud computing, fog computing, big data and the Internet of Things are popular paradigms and their features can be combined for shaping the next era of data processing, storing and forwarding. Many applications need a data replication model, where the data consistency is a challenge [49–51]. We may find many techniques which are proposed for systems restoring processes. In this kind of systems, a restoring process fires a transaction consistency across consistency groups, e.g., disaster recovery systems. Several consistency groups are defined for replication. For instance, the author in [49] demonstrates a multi-purpose approach for disaster recovery. The consistency is involved to check whether any data has been lost, damaged or corrupted in the disaster recovery process.

We may cite several works in the literature where data consistency plays an important role. In order to show its benefits in different fields, the data consistency discussion will be divided into three main classes.

1. With the aim to provide an efficient disaster recovery system, the storage manager must generate multiple recovery copies of the client file and oversee the transmission of this recovery copies to remote sites. What is needed in this situation, is a way to ensure the data consistency of the generated multiple recovery copies. The author of [49] has proposed a big data system disaster recovery which focusses on multi-site and multi-technique approaches to ensure that if one method does not succeed, there are other methods that can retrieve and restore the data. The author considers the data consistency as an essential parameter in the experimentation and result. To reach this aim, the author has checked the data consistency of the proposed system during its experimentation with the aim to identify whether any data has been lost, damaged or corrupted in the disaster recovery process. The authors in [52] studied techniques that maintain stored data consistency when migrating virtual machines from a first site to a second site. The main objective aims to relocate the virtual machines when any migration is planned from the source site to the destination site during maintaining a consistent data. The data consistency challenge can be particularly observed while maintaining a consistent set of data with a pair of consistent data images between the protected source site and the destination recovery site. The authors use the disaster recovery as an exemplary scenario with the aim to avoid a disaster that is

predicted to occur in the vicinity of the source site. Another work can be cited [53], in which the data consistency plays an important role. The authors of [53] have tackled the problem of backing up an entire micro-service architecture in their work, where a running application was decomposed into multiple micro-services in such a way that it can be recovered after a disaster strike. The main objective of this work aims to ensure, in case of disaster, the impossibility to achieve a holistic recovery that brings back all micro-services in a globally consistent state.

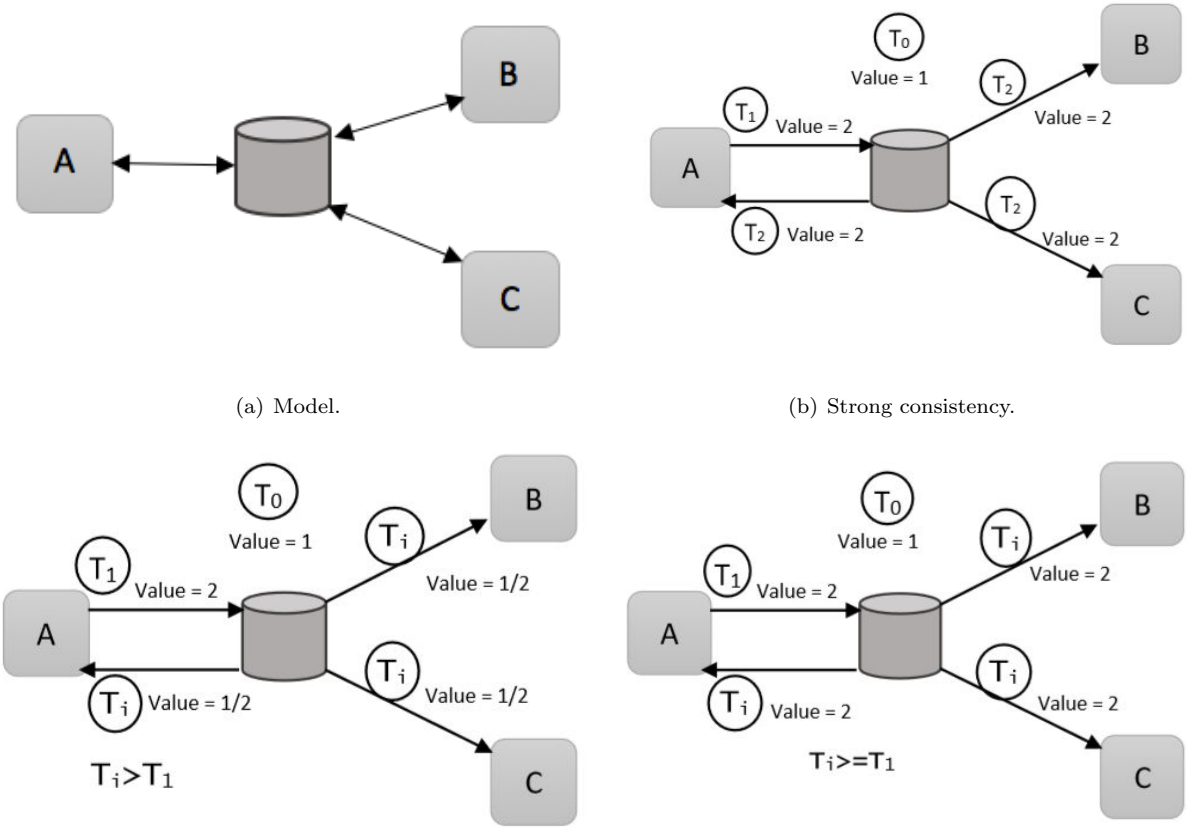
2. It is also necessary to know the data consistency advantages and uses in terms of security and privacy challenges. For example, in applications where different central entities collaborate with each other, they must be able to exchange trust information in order to fix inconsistencies in the reputation values [54]. For instance, any time instantiation of edge devices in fog computing can be used by multiple smart applications with set of users which raises the issue of edge device security and makes appear the consistency challenges [55]. The authors of [56] consider the consistency of data to be a primordial factor in several security aspects. The data integrity needs the action of maintaining consistency, where ethical values are important for cloud service providers to protect integrity of cloud user's data with honesty, truthfulness and accuracy at all time. The work of [57] has studied the eventual consistency as a relaxed trade-off model in order to link the strong consistency and availability. The authors of this work introduce *Byzec* which is a protocol that makes eventual consistency as secure as possible. The proposed protocol allows the service to run in an eventually consistent manner.
3. Software Defined Networking (SDN) is a promising network paradigm that aims to solve such problems and accelerate innovation. Its principal is to decouple the control plane and the data plane to achieve a logically centralized control architecture providing programmability to configure the network [58]. We may cite several works focussing on the design choice of distributed SDN controller with the aim to show that to build an efficient distributed controller it should consider the following aspects: scalability, robustness, consistency and security [59–62]. The papers of [63, 64] define the use of adaptive consistency models in the context of distributed SDN controllers. The authors of [63] aim to show how the typical SDN controller architecture can be extended in order to support adaptive consistency. The obtained results showed that adaptive controllers were more resilient to sudden changes in the network conditions than the non-adaptive consistency controllers. However, in [64], an adaptive consistency model for SDN controllers that employs concepts of eventual consistency models has been introduced by defining the concept of runtime adaptation of consistency levels in state synchronization for a distributed SDN control plane.

3.4. Consistency models

In this subsection, we introduce the main consistency models adopted in earlier single-site storage systems and in current geo-replicated systems. Many works addressed these consistency models such as [44], [65] and

[66]. 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 fewer conflicts than weak consistency. In the following, we adopt the classification of Werner Vogels [65] 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 observes 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 us assume that there are three independent processes: A, B and C which need to communicate to share information as shown in Figure 1(a). When process A updates a data item, there are three possible cases:
   - *Strong consistency*: An access by any process, after completeness of the update, will return the recent data (Figure 1(b)). In other words, 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.
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 us suppose that N is the number of replicas in the system, W the number of replicas involved in a write/update operation and R 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 is 5 too (W = 5), it is sufficient to read from one replica (R=1) to get the most recent data $W + R > N$.
     - *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), and if he reads from more than



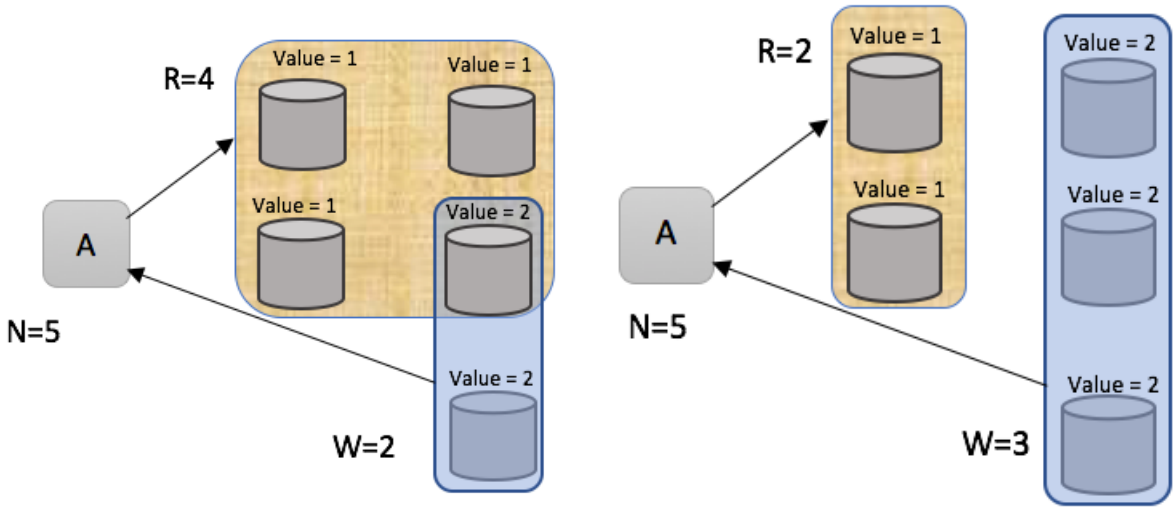
(a) Model.

(b) Strong consistency.

(c) Weak consistency.

(d) Eventual consistency.

Figure 1: Client-Side consistency.



(a) Strong consistency.

(b) Weak consistency.

Figure 2: Server-Side consistency.

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 $W + R \leqslant N$, 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 a read operation, it is less likely to read from a replica that has the latest write (Figure 2(b)).

3.5. 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 trade-off 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 [66]. Several kinds of applications need adaptive consistency. For example, Web shop applications store different kinds of data that need different consistency levels. The customer's credit card information and the price of the items must be handled under strong consistency. However, clients buying preferences and logging information could be handled under weak consistency. An online auction system is another example that needs adapative consistency but the selection of a consistency level in this case is based on time. At the

beginning, the data are not so important for the final deal, so the requirement of consistency is low. With the deadline approaching, auctioned items become very popular and the customers rely more and more on the latest data to make the next bid. Eventually, the data should be always up-to-date and modified under strong consistency.

4. Comparison criteria of adaptive consistency approaches

When talking about adaptive consistency we distinguish numerous criteria that are to be described. Among these issues we are particularly interested in the following ones: architectural model, operation level, granularity, adaptive policy, prediction or statistic based, conflicts consideration, threshold, monetary costs, security and privacy, provided consistency and implementation tools. These criteria have a direct influence on the development of adaptive consistency approaches.

4.1. Architectural model

Consistency management is a functionality of the storage system that is queried by cloud clients. Thus, each proposed consistency approach should take a position in the storage system architecture or his interaction with client applications. According to their architectures, the proposed adaptive consistency approaches come in two forms: either an additional module between the storage system and the application or an entire storage system. In the first case, the additional module can be on the top of the storage system, in bottom of client application or as a middleware.

4.2. Operation level

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

4.3. Granularity

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

4.4. Adaptive policy

An adaptive policy is a set of algorithms, models or techniques that are used to provide adaptive consistency. It captures the system's need and gives the best trade-off 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 other predictive algorithms. Choosing a good adaptive policy leads to a pertinent adaptive consistency approach.

4.5. Prediction or statistic based

The adaptive policy may be based on prediction or statistic. Predictive policy uses probabilistic models, regression models 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 statistical model calculates the rate of a given metric during runtime and compares it to a defined threshold to change the consistency level when it is achieved. The statistical model can give exact results but it produces additional and complicate calculations.

4.6. Conflicts management

We chose this criterion to investigate wether the proposed approach has considered or not conflicts between transactions . We define which mechanism was used to detect and resolve the conflicts. Normally, conflicts between replica occur highly when the storage system runs transactions optimistically i.e., provides a low level of consistency. Contrarily, raising the level of consistency decreases the percentage of conflicts.

4.7. 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 an approach means that the system does not calculate the application needs and that the tuning parameters are put manually.

4.8. Monetary costs

The cost of a consistency level in the cloud describes the different resources necessary 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 interesting advantages of cloud storage. Hence, any consistency approach should take this criterion into consideration.

4.9. Security and privacy

Security and privacy are always key concerns in information technology to which more challenges are brought about by cloud computing and big data. In fact, moving data and applications to a third party service provider and replicating objects makes the challenge more critical. Furthermore, the availability of cloud services is one of the most important security issues which directly affects the business of cloud service providers and also its customers. Thereby, any consistency approach should take into consideration these two factors.

4.10. Provided consistency

Data consistency is already provided by storage systems which can ensure many levels of consistency. Thereby, the application can choose the suitable level for its needs from the beginning. Adaptive consistency approaches were proposed to optimize consistency management by tuning the consistency to the suitable level during the run-time. To achieve this goal, existing adaptive approaches behave in three manners: switch between existing consistency levels to assign the nearest one to the application needs and the system state, moderate the weak level by an artificial delay to enhance consistency when it is needed, provide a new level according to the application needs.

4.11. Implementation tools

Evaluating a proposed approach in cloud and big data environments is quite difficult due to the high monetary charges and the large amount of data needed to complete the implementation. Nevertheless, using real implementation tools gives more credibility to the evaluation results. Since they are free of charge and easy to use, simulation tools are widely used to evaluate contributions and to emulate real environments. However, the obtained results are less credible.

5. 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.

5.1. IDEA

The infrastructure IDEA [21] (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).

5.2. Consistency rationing

Consistency rationing [22] 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 are handled with either strong or session consistency depending on the specified policy. In this paper, the authors present and compare many policies that switch between session and strong consistency: 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. The aim is to find a trade-off between transactions cost in the case of strong consistency, and penalty costs (financial) in the case of weak consistency.

5.3. An application-based adaptive replica consistency

The paper [23] proposed an adaptive mechanism for consistency management that allows the system to automatically switch between consistency strategies according to update frequency and read frequency at runtime. The authors also proposed a model structure that can manage the different consistency levels. 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 levels 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 evaluates 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 demonstrated, via simulations, that the adaptive approach reduces significantly the global throughput generated when applying strong consistency and that it provides better performance than weak consistency.

5.4. RedBlue

RedBlue consistency [24] was introduced to provide an adaptive consistency approach by decoupling operations into two types: red operations that require a strong consistency and blue operations that are executed under weaker consistency. To color an operation red or blue, the authors define the conditions that ensure the non-violation of application constraints and the convergence of all replicas to the same final state. Intuitively, commutative operations may be blue if they do not violate the application constraints. An extension of the proposed approach consists in dividing original application operations into two categories: a generator operation that has no side-effect and which is executed only at the primary site, and a shadow operation, which is replicated to all sites. Only shadow operations are colored red or blue.

5.5. Harmony

Harmony [25] is based on the estimation model of stale reads that will be adjusted to the application needs. In this approach, the application provides the 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 the greater value, the algorithm calculates the number of replicas (N) needed to attenuate the stale read rate and modifies the consistency level according to the obtained N.

5.6. Bismar

Bismar [26] takes the monetary cost into consideration, both for the evaluation and selection of consistency levels in the cloud. Accordingly, the authors define a new metric called consistency-cost efficiency. Based on this metric, they propose an economic 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 probabilistic consistency model to estimate the stale reads and the relative costs of the application according to the current read/write rate and the network latency. Then the algorithm selects the consistency level that offers the most equitable consistency or cost trade-off (given by the maximum consistency-cost efficiency value: max[consistency(cl)/cost(cl)]).

5.7. Pileus

Pileus [27] is a key-value storage system that supports consistency-based service level agreements (SLAs). With SLAs, applications can declare their consistency and latency priorities. With the offered consistency choices, applications can indicate a decreasing series of desired consistency/latency trade-offs (SubSLAs). Furthermore, applications that share the same data are allowed to obtain different consistency guarantees. Pileus tries to satisfy the best desired service due to the configuration of replicas and the network conditions. It may not always succeed in satisfying the first subSLA. Therefore, Pileus tries out lower subSLAs that are acceptable but less desirable. Finally, the system returns an error code and no data if none of the subSLAs can be satisfied.

5.8. DepSky

DepSky [28] is a virtual storage cloud, a cloud of clouds, which is accessed by its users to manage data items stored in a group of individual clouds. It targets four limitations of individual clouds: availability, consistency, privacy and monetary costs. The DepSky system enhances data availability and consistency by exploiting data replication on several clouds and by introducing proportional consistency. Thus, the system allows access to the data whereas a subset of them is reachable and provides the same level of consistency of the subordinate clouds. To minimize the monetary cost and improve the privacy, DepSky uses erasure codes [67] to store only a fraction (typically half) of the total amount of data in each cloud and to avoid storing

clear data. Furthermore, stored data are encrypted and the encryption key is shared by the underlying clouds so that individual faulty clouds cannot reconstruct it to disclose the data.

5.9. File heat-based self-adaptive replica consistency

This work proposed the algorithm MRFU [29] (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 selects 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 also proposed a replica management model that divides the system into three levels: one storage control node that controls many main replicas, each of which is connected to many other subreplicas.

5.10. Consistency tuner

Consistency tuner is a protocol based on the consistency index [30] (number of correct reads/total number of reads). To adjust the consistency index to a desired value the protocol predicts the correctness of an incoming read request using a logistic regression classifier and a 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 input parameters. The authors also implemented the CICT (consistency index based consistency tuner) in an architecture of a web based database application and demonstrate the relationship between the number of replicas and the threshold of a time gap (minimum value of time gap between an update and a succeeding read request) using a statistical linear regression analysis.

5.11. Fine-tuning the consistency-latency trade-off

The trade-off consistency-latency is addressed by proposing and evaluating two techniques [31]. The first is 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 artificially 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.

5.12. A self-adaptive conflict resolution with flexible consistency guarantee

The paper [32] presents an adaptive and hierarchical model using version vectors of replicas to ensure consistency management. The proposed approach divides the consistency management into two levels:

global management between data centers in the cloud and local management in the data center. 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 also propose the integration of a conflict detection and resolution mechanism between the replicas based on version vectors. This mechanism resolves the conflict in the local data center and then the conflicts between different data centers in the cloud.

5.13. Incremental consistency guarantees

ICG [33] targeted the consistency/performance trade-off by developing an abstraction interface, called *Correctables*, between applications and replicated objects. Therefore, rather than struggling with the complexity of distributed storage protocols, developers will just focus on tuning the consistency level of the ongoing operation. After invoking an operation on a replicated object, the abstraction interface provides a view on the operation result for each specific consistency level. First obtained views reflect operation results under weak consistency while stronger consistency will be guaranteed within later views.

5.14. Safe serializable secure scheduling

The paper [34] addresses the trade-off between strong consistency guarantees and strong security properties in decentralized systems. The authors state that distributed transaction scheduling mechanisms cannot prevent unauthorized access to confidential information. Security risks are potential due to the aborting messages of failed transaction. Thus, they proposed the staged commit protocol that can secure the transaction scheduling using relaxed monotonicity. The defined protocol divides a transaction into stages, each of which can be securely committed using a traditional protocol.

5.15. OptCon

OptCon [35] 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 provides the following dynamic parameters as input variables to the learning algorithms: the read proportion in the operation, the number of user threads spawned by the client, and 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.

5.16. SPECSHIFT

SPECSHIFT [36] is a tuning framework that uses artificial delays to adjust the consistency level according to network conditions and workload characteristics. The delay is calculated from a combination of empirical measurement and probabilistic analysis and injected at the beginning of each read and at the end of each write. Developers of this framework also presented a probabilistic model of consistency under latency optimized settings that precisely captures the relationship between consistency, system workload and network latency.

5.17. Selective data consistency

The paper [37] proposes a selective model for transactional application data consistency. The model is built on the replicated data store MongoDB. To boost application performance, strong consistency is only applied to selected critical data objects. However, less critical objects can have lower consistency levels. The second contribution of this paper is the mathematical function that analyses data criticality using the number of reads and the number of writes of data items. This function is designed for a shopping application and considers three indices for determining total criticality: popularity index, stock sales index and stock purchase quantity index.

5.18. Adaptive consistency policy for kafka

Kafka [68] is a distributed streaming platform that allows to build real-time data pipelines and streaming applications. The work [38] proposes a replica adaptive synchronization strategy for Kafka based on the message heat and replica update frequency. For every period of time T_i, the heat and the update frequency of a partition (unit of data replication) are calculated following particular formulas. If the partition heat is greater than the update frequency, synchronous replication (high consistency level) should be granted for this partition. Otherwise, the partition will receive updates asynchronously.

5.19. FogStore

FogStore [39] is an extension designed on top of existing distributed storage systems to allow their seamless integration into a Fog Computing environment. To provide best consistency management in Fog nodes, Fogstore ensures two additional functionalities: Fog-aware replica placement and context-aware differential consistency. A replica placement strategy should optimize the placement to achieve minimal response time between the copy of a data record, the data sources (devices) and the clients. Context-aware differential consistency matches the consistency level of the client's query to the client's context. In fact, clients who query the Fog data store often have an individual context, the position in particular, which can influence their requirements on consistency.

6. A comparison of the different approaches proposed for adaptive consistency in the cloud

Table 1 summarizes the evaluation of adaptive consistency approaches according to the criteria.

	The comparison criteria										
adaptive consistency approaches	Architecture	Operation level	Granularity	Adaptive policies	Prediction	Conflicts management	Metric in which the threshold is defined	Monetary cost	Security and privacy	Provided consistency	Implementation tools
IDEA [21]	Middleware	Object	Object	Statistical model(Version vector)	×	Uses version vector to detect conflicts	Implicit on calculated consistency value	×	×	New level	Planet-Lab
Consistency rationing [22]	Middleware	Object	Category	Probability, Time policy, Fixed threshold, Demarcation, Dynamic policy	×	Uses conflict probability as a metric	Conflict probability, fixed threshold, dynamic policy	✓	×	Switch between strong and session consistency	Amazon EC2, S3, TPC-W
ABARC [23]	Entire CMS	Query	Global	Statistical model(Read/write frequency)	×	×	Time between two writes	×	×	Moderated level	OptorSim
RedBlue [24]	CMS + MySQL as Backend	Query	Global	Coexistence of multiple consistency levels	×	×	×	×	×	Switch between many consistency levels	Gemini, MySQL
Harmony [25]	Middleware + ACSM	Query	Global	Probability	✓	×	×	×	×	New level	EC2/Grid'5000, Cassandra, Ycsb
Bismar [26]	Middleware	Query	Global	Probability, Consistency-cost efficiency	✓	×	×	✓	×	New level	EC2/Grid'5000, Cassandra, Ycsb
Pileus [27]	Entire CMS	Query	Global	Coexistence of multiple consistency levels	×	×	×	×	×	Switch between many consistency levels	Pileus, Ycsb
DepSky [28]	Middleware	Query	Global	Proportional consistency	×	×	×	✓	✓	Proportional consistency	DepSky, Amazon S3, Windows Azure, RackSpace, Nirvanix
FHBC [29]	Entire CMS	Object	File	Statistical model(file heat)	×	×	File heat	×	×	Switch between strong and eventual consistency	OptorSim
Consistency Tuner [30]	Middleware	Query	Global	RBFNN, Logistic regression, Linear regression	✓	×	Time gap	×	×	New level	Amazon SimpleDB, TPC-C
CPQ [31]	ACSM	Query	Global	CPQ + Probability	✓	×	×	×	×	Choosing randomly between weak and strong consistency	Amazon EC2, Cassandra, Ycsb
Artifical delay [31]	ACSM	Query	Global	Weak consistency level + Tunable Artifical delay	×	Uses Artifical delay to decrease conflicts rate	×	×	×	Moderated level	Amazon EC2, Cassandra, Ycsb
SACRFCG [32]	Entire CMS	Query	Global	Statistical model(write rate)	×	Uses version vector to detect conflicts	Write rate	×	×	Switch between weak and strong consistency	CloudSim (v3)

ICG [33]	Middleware	Query	Global	Coexistence of multiple consistency levels	×	×	×	×	×	Switch between strong and eventual consistency	Amazon EC2, Cassandra/ZooKeeper, Ycsb
4S [34]	Entire CMS	Query	global	Staged commit protocol	×	√	×	×	√	New level	Fabric system
OptCon [35]	Middleware + ACSM	Query	Global	Logistic regression, Bayesian learning, Decision Tree, Random Forest and ANN	√	×	Latency, staleness	×	×	New level	Amazon EC2, Cassandra, Ycsb
SPECSHIFT [36]	Middleware	Query	Global	Probability	√	Uses artificial delay to decrease conflicts rate	×	×	×	Moderated level	Amazon EC2, Cassandra, Ycsb
SCM [37]	Middleware	Object	Critical data	Criticality analyser	×	×	×	×	×	Switch between strong and eventual consistency	Amazon cloud, MongoDB, TPC-C
ACPK [37]	Middleware	Object	Partition	Statistical model(topic heat and update frequency)	√	×	Topic heat and update frequency	×	×	Switch between strong and eventual consistency	Kafka cluster
FogStore [39]	Middleware	Query	Regions	Client context-awareness	√	×	×	×	×	Switch between strong and eventual consistency	MaxiNet, OpenStack

ABARC:An Application-Based Adaptive Replica Consistency.
FHBC: File heat-based self-adaptive replica consistency.
SACRFCG:Self-adaptive conflict resolution with flexible consistency guarantee.
ICG:Incremental Consistency Guarantees.
SCM: Selective Consistency Model.
ACPK: Adaptive consistency policy for kafka.
4S:Safe Serializable Secure Scheduling.
√ : the approach used the criterion.
× : the approach didn't use the criterion.
CMS : Consistency Management System.
ACSM : Additional Client-Side Module.

Table 1: Summary table of evaluation criteria.

7. Discussion

This paper has given a review on the works that propose adaptive approaches to manage the consistency in the cloud. The main goal of these approaches is to optimize the performance of cloud systems and to provide a suitable consistency level to client applications.

According to the architectural model of the proposed adaptive consistency approaches, we state that most of the approaches provide adaptive consistency as an additional functionality to the existing consistency policy. Since data consistency is already managed by storage systems, this functionality is injected in the middle of a server and client application in the form of middleware. Some approaches add a module to the client application to communicate with the middleware like in *Harmony* and *OptCon*. However, author approaches implement necessary adaptive functionalities in the client side module like in *DepSky, CPQ* and *Artifical Delay*. On the other hand, few propositions have designed an entire consistency management system such as *ABARC, Pileu, FHBC, SACRFCG and 4S*. Approaches that add adaptive consistency functionality to the existing storage system are more effective, either as a middleware or on the client side. In fact, rather than facing all challenges of designing an entire consistency management system, these approaches focus only on the adaptive consistency challenge and avail of the existing consistency management. Moreover, the additional module can be tested in the real environment contrarily to the consistency management system which is usually tested by simulation tools.

Most of the studied approaches apply the chosen consistency level to the overall system with the exception of six approaches: *Consistency Rationing* divides the system into categories, *SCM* applies strong consistency on critical data, *ACPK* divides the system into partitions, *ForStore* divides the system into regions and *FHBC* and *IDEA* focus on the object. These works tune the consistency level according to data importance unlike the other works that focus on application needs. Only the *FogStore* approach considers client and data contexts at the same time. Categorizing data objects according to their criticality allows the system to apply lower consistency levels to lower critical categories, and therefore, enhance performance especially in the case of high volume data with small percentage of critical data. On the other hand, data categorization may adversely affect system performance if it reposes on a statistical model or other complex functions.

To optimize the consistency/performance trade-off, the proposed approaches use intelligent techniques that predict the suitable consistency level like *Harmony, Bismar, Consistency Rationing, CPQ, OptCon, SPACHIFT and ACPK*. However, other approaches use statistics like *Consistency Tuner, ABARC and SACRFCG*. Unlike these approaches, *FHBC, IDEA, 4S, FogStore and ICG* use particular protocols to enhance consistency management. Due to the large volume and variety of data, applying statistical models obligates the system to collect statistics for each operation or data object which will generate huge additional charges in processing and storage. Therefore, approaches that use machine learning techniques overcome the statistics inconvenience by treating only a subset of data (training data) to generate the predictive model.

Thus, intelligent techniques enhance performance but provide less accurate prediction.

When talking about conflicts, two approaches use version vectors to detect conflicts between different replicas of the same object: *SACRFCG and IDEA*. *AD and SPECSHIFT* use artificial delays to allow more time for update propagation through the system, which decreases the likelihood of conflicts. *Consistency rationing* uses the probability of conflicts as a metric that is used to tune the consistency level when it is compared with defined thresholds. *Staged protocol* proposed in *4S* aims to reduce aborting messages and thereby it reduces the conflict rate. Traditional conflict management mechanisms become more and more complex when moving to the cloud and big data environments as they save all historical events for each object. Artificial delay and probability represent an alternative solution of traditional mechanisms as they have an optimistic behavior to predict and as they reduce most conflicts without affecting performance.

The threshold is used by several approaches as a trigger to change the consistency level when it is achieved by the value of the defined metric. Among these approaches we find: *consistency rationing, consistency tuner, ABARC, FHBC, SACRFCG, OptCon and ACPK.*

Only three works take into consideration the monetary costs of different consistency levels. *Consistency rationing* optimizes the runtime cost and makes a trade-off between consistency cost and penalty cost of inconsistencies. *Bismar* proposes a method to calculate the cost of consistency and combines it with the rate of stale reads to choose the best level of consistency. *DepSky* uses erasure codes to store only a part of the total amount of data in each cloud, and therefore minimizes the monetary cost. We state that the best way of considering monetary cost in an adaptive consistency approach is to combine it with other input parameters. In this case, the monetary cost can have a direct effect on the adopted level of consistency.

Despite its importance in cloud storage, security and privacy was not considered by most of the proposed approaches. Only two works raised security issues with adaptive consistency. *DepSky* used an erasure code and an encryption with a shared key. *4S* proposed the staged commit protocol which can prevent unauthorized access to confidential information.

The proposed approaches adapt the consistency level during runtime by giving a suitable level according to application needs and the system state. Most approaches switch between existing levels and give the nearest one. However, a second type of approaches gives a new calculated level from its defined policy like *IDEA, Harmony, Bismar, Consistency Tuner, 4S and OptCon.* A third type of approaches moderates the existing level to enhance consistency like *ABARC, AD an SPECSHIFT.* We state that delivering a new calculated level should be the best solution especially when the existing storage system provides few and divergent consistency levels. Thus, calculating the desired level gives the most appropriate consistency guarantees.

Only three works used simulation in their implementation: *ABARC, FHBC and SACRFCG.* The remaining works used real implementations which adds more credibility to their results.

8. Challenges and future research directions

It can be deduced, from this review, that there are still a lot of issues to be addressed either in the cloud or in author computing environments. Therefore, some open challenges and research directions are discussed in this section.

8.1. Consistency in emerging computing architectures

The fast evolution of Internet of Things (IoT) applications has brought about more challenges to classical centralized cloud computing such as network failure and high response time. Fog and mobile edge computing target these challenges by bringing the cloud closer to IoT devices which provides low latency and secure services. An adaptive consistency approach in this environment should consider more criteria specifically replica and edge servers placement [69, 70], client context-awareness [39], security and privacy [55]:

- *Replica and Edge servers placement*: Bringing computation power to the edge of network reduces latency but obstructs many services that need to access centrally stored data. Thus, a replica placement strategy is recommended to place data originating from a central server towards the end devices in the cloud network in order to decrease data access response time and enhance the benefits of edge computing. Moreover, the locations of edge servers are critical to the access delays of mobile users and the resource utilization of edge servers, which makes edge servers placement highly crucial. Inefficient edge server placement will result in long access delays and heavily unbalanced workload among edge servers.

- *Client context-awareness*: Contrary to classical cloud computing, different clients accessing the distributed Fog computing data store often have an individual context. This context can influence their requirements on consistency. For instance, a situation aware application [71] should consider conditions and things that happen at a particular time and place.

- *Security and privacy*: A hacker, for example, could deploy malicious applications on an edge node by exploiting vulnerability. If a fog node is hacked, it can get false input and formulate bad results which can affect the performance of the whole application. Therefore, the development of methods to characterize and detect malwares at large scale is recommended. Moreover, since user specific data need to be temporarily stored on multiple edge locations, privacy issues will need to be targeted along with security challenges.

Consistency is also needed in another new paradigm: Software-Defined Networking [72]. SDN is an emerging network architecture that separates the control plane from the data plane in order to provide programming network configuration. To avoid single point of failure and reduce response latency, multiple controllers are needed. Therefore, consistency requires that every controller has an identical global view of the network

state. The challenging task in distributed SDN controllers is to maintain a consistent and up-to-date global network view for SDN applications while preserving good performance.

8.2. Data clustering to categories

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. However, one difficulty that arises is the categorization of data objects due to large volumes of data and various data types. The related challenge in this situation is to provide an automatic mechanism that allows splitting data into different consistency categories by applying clustering techniques [73]. In this way, according to its enclosing data, every category should be given the most appropriate consistency level.

8.3. Intelligent techniques for adaptation and other input parameters

A statistical policy is very expensive to implement in big data due to the large amount of heterogeneous data. So, using an intelligent technique is better in terms of performance and costs. Traditional machine learning algorithms were restricted to execution on large clusters given the large computational requirements. Thus, recent researches in applying machine learning classification schemes are directed to the cloud [74, 75] to exploit its availability and obtain cheaper data storage. These researches are branded under the name of 'Deep Learning'. Deep learning frameworks, like Google TensorFlow[1] and Nervana Cloud[2], provide APIs and software libraries to perform complex learning tasks in a reasonable time and to give more accurate results.

Adaptive consistency approaches studied in this paper have considered many parameters in the input set of learning algorithms such as: number of replicas, time gaps, network packet count, average throughput, etc. We propose to consider other parameters to tune the consistency level, as for example, access permissions. A role based access control is an access control policy that classifies clients according to their roles in the system. Hence, clients that have the same role should get the same consistency level. The client context in fog computing environments can be also used to predict the adequate consistency level. Thus, a client who changes his place or access period should not maintain the same consistency guarantees.

8.4. Minimizing monetary cost

Taking monetary costs into consideration is very important in such approaches because it is among the principle goals of using the cloud. Stronger consistency causes higher monetary cost by means of synchronous replications that introduce high latencies due to cross-site communication. On the other hand, lower levels of consistency may reduce the monetary cost by favoring performance to latency, but increase the rate of

[1] https://www.tensorflow.org/.

[2] https://www.nervanasys.com/cloud/.

stale reads which raises costs if the application imposes penalties to incorrect operations. Partial replication [76] can give better performance with low costs for some applications. In fact, placing replicas at only a subset of data centers significantly reduces the number of messages sent with each write operation. Thus, partial replication reduces both storage and communication costs.

9. Conclusion

In this paper, we have presented a comparative study between adaptive consistency approaches. A taxonomy of the adaptive consistency approaches has been introduced by defining several comparison criteria such as architectural model, conflicts, granularity, adaptive policy, operation level, threshold, monetary costs, security and implementation tools. Furthermore, the paper reviews an interesting number of studies on adaptive consistency in cloud computing which allowed us to build a table-based analysis that summarizes our comparative study. A detailed discussion of the behavior of each approach is also given for each criterion. According to the results of the studied approaches, we have deduced for each criterion the suitable choice for each situation. Based on the analysis, we have identified several research challenges that need further investigation. These challenges include consistency in emerging computing architectures, intelligent learning techniques, monetary cost and data clustering.

References

[1] Peter Mell and Tim Grance. The nist definition of cloud computing. *Communications of the ACM*, 53(6):50, 2010.

[2] Hai Jin, Shadi Ibrahim, Tim Bell, Li Qi, Haijun Cao, Song Wu, and Xuanhua Shi. Tools and technologies for building clouds. In *Cloud Computing*, pages 3–20. Springer, 2010.

[3] Shuai Zhang, Xuebin Chen, Shufen Zhang, and Xiuzhen Huo. The comparison between cloud computing and grid computing. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, volume 11, pages V11–72. IEEE, 2010.

[4] Junjie Peng, Xuejun Zhang, Zhou Lei, Bofeng Zhang, Wu Zhang, and Qing Li. Comparison of several cloud computing platforms. In *Information Science and Engineering (ISISE), 2009 Second International Symposium on*, pages 23–27. IEEE, 2009.

[5] Matthias Wiesmann, Fernando Pedone, Andre Schiper, Bettina Kemme, and Gustavo Alonso. Understanding replication in databases and distributed systems. In *Distributed Computing Systems, 2000. Proceedings. 20th International Conference on*, pages 464–474. IEEE, 2000.

[6] Vlad Serbanescu, Florin Pop, Valentin Cristea, and Gabriel Antoniu. Architecture of distributed data aggregation service. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications (AINA)*, pages 727–734. IEEE, 2014.

[7] Vlad Serbanescu, Florin Pop, Valentin Cristea, and Gabriel Antoniu. A formal method for rule analysis and validation in distributed data aggregation service. *World Wide Web*, 18(6):1717–1736, 2015.

[8] John W Bates and Mark Aldred. System and method for secure and reliable multi-cloud data replication, June 24 2014. US Patent 8,762,642.

[9] Dejene Boru, Dzmitry Kliazovich, Fabrizio Granelli, Pascal Bouvry, and Albert Y Zomaya. Energy-efficient data replication in cloud computing datacenters. *Cluster computing*, 18(1):385–402, 2015.

[10] Shrikant V Karve, Janmejay S Kulkarni, Sarvesh S Patel, Ashish R Pathak, and Sandeep R Patil. Cloud based data migration and replication, July 7 2015. US Patent 9,075,529.

[11] Ismail Kertiou, Saber Benharzallah, Laid Kahloul, Mounir Beggas, Reinhardt Euler, Abdelkader Laouid, and Ahcène Bounceur. A dynamic skyline technique for a context-aware selection of the best sensors in an iot architecture. *Ad Hoc Networks*, 81:183–196, 2018.

[12] Heinz Stockinger, Asad Samar, Koen Holtman, Bill Allcock, Ian Foster, and Brian Tierney. File and object replication in data grids. *Cluster Computing*, 5(3):305–314, 2002.

[13] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[14] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin 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*, pages 143–157. ACM, 2011.

[15] Apache hbase. `http://hbase.apache.org/`, February 2018.

[16] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, 41(6):205–220, 2007.

[17] Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[18] mongodb. `http://www.mongodb.org/`, February 2018.

[19] Victor Chang and Gary Wills. A model to compare cloud and non-cloud storage of big data. *Future Generation Computer Systems*, 57:56–76, 2016.

[20] Tianying Chang, George Popescu, and Chris Codella. Scalable and efficient update dissemination for distributed interactive applications. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, pages 143–150. IEEE, 2002.

[21] Yijun Lu, Ying Lu, and Hong Jiang. Adaptive consistency guarantees for large-scale replicated services. In *Networking, Architecture, and Storage, 2008. NAS'08. International Conference on*, pages 89–96. IEEE, 2008.

[22] Tim Kraska, Martin Hentschel, Gustavo Alonso, and Donald Kossmann. Consistency rationing in the cloud: pay only when it matters. *Proceedings of the VLDB Endowment*, 2(1):253–264, 2009.

[23] Ximei Wang, Shoubao Yang, Shuling Wang, Xianlong Niu, and Jing Xu. An application-based adaptive replica consistency for cloud storage. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pages 13–17. IEEE, 2010.

[24] Cheng Li, Daniel Porto, Allen Clement, Johannes Gehrke, Nuno M Preguica, and Rodrigo Rodrigues. Making geo-replicated systems fast as possible, consistent when necessary. In *OSDI*, volume 12, pages 265–278, 2012.

[25] Houssem-Eddine Chihoub, Shadi Ibrahim, Gabriel Antoniu, and Maria S Perez. Harmony: Towards automated self-adaptive consistency in cloud storage. In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pages 293–301. IEEE, 2012.

[26] Houssem-Eddine Chihoub, Shadi Ibrahim, Gabriel Antoniu, and Maria S Perez. Consistency in the cloud: When money does matter! In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 352–359. IEEE, 2013.

[27] Douglas B Terry, Vijayan Prabhakaran, Ramakrishna Kotla, Mahesh Balakrishnan, Marcos K Aguilera, and Hussam Abu-Libdeh. Consistency-based service level agreements for cloud storage. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 309–324. ACM, 2013.

[28] Alysson Bessani, Miguel Correia, Bruno Quaresma, Fernando André, and Paulo Sousa. Depsky: dependable and secure storage in a cloud-of-clouds. *ACM Transactions on Storage (TOS)*, 9(4):12, 2013.

[29] Zhen Zhou, Shuyu Chen, Tao Ren, and Tianshu Wu. File heat-based self-adaptive replica consistency strategy for cloud storage'. *J Computers*, 9(8):1928–1933, 2014.

[30] Shraddha P Phansalkar and Ajay R Dani. Tunable consistency guarantees of selective data consistency model. *Journal of Cloud Computing*, 4(1):13, 2015.

[31] Marlon McKenzie, Hua Fan, and Wojciech Golab. Fine-tuning the consistency-latency trade-off in quorum-replicated distributed storage systems. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 1708–1717. IEEE, 2015.

[32] Said Limam and Ghalem Belalem. A self-adaptive conflict resolution with flexible consistency guarantee in the cloud computing. *Multiagent and Grid Systems*, 12(3):217–238, 2016.

[33] Rachid Guerraoui, Matej Pavlovic, and Dragos-Adrian Seredinschi. Incremental consistency guarantees for replicated objects. In *OSDI*, pages 169–184, 2016.

[34] Isaac Sheff, Tom Magrino, Jed Liu, Andrew C Myers, and Robbert van Renesse. Safe serializable secure scheduling: Transactions and the trade-off between security and consistency. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 229–241. ACM, 2016.

[35] Subhajit Sidhanta, Wojciech Golab, Supratik Mukhopadhyay, and Saikat Basu. Adaptable sla-aware consistency tuning for quorum-replicated datastores. *IEEE Transactions on Big Data*, 3(3):248–261, 2017.

[36] Shankha Chatterjee and Wojciech Golab. Self-tuning eventually-consistent data stores. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 78–92. Springer, 2017.

[37] VIJAY GHANEKAR and DR SHRADDHA PHANSALKAR. Selective consistency model on mongodb big data store. *Journal of Theoretical and Applied Information Technology*, 96(5), 2018.

[38] Zonghuai Guo and Shiwang Ding. Adaptive replica consistency policy for kafka. In *MATEC Web of Conferences*, volume 173, page 01019. EDP Sciences, 2018.

[39] Harshit Gupta and Umakishore Ramachandran. Fogstore: A geo-distributed key-value store guaranteeing low latency for strongly consistent access. In *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems*, pages 148–159. ACM, 2018.

[40] Haifeng Yu and Amin Vahdat. Design and evaluation of a continuous consistency model for replicated services. In *Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation-Volume 4*, page 21. USENIX Association, 2000.

[41] Youngjoo Shin, Dongyoung Koo, and Junbeom Hur. A survey of secure data deduplication schemes for cloud storage systems. *ACM Computing Surveys (CSUR)*, 49(4):74, 2017.

[42] Paolo Viotti and Marko Vukolić. Consistency in non-transactional distributed storage systems. *ACM Computing Surveys (CSUR)*, 49(1):19, 2016.

[43] Yaser Mansouri, Adel Nadjaran Toosi, and Rajkumar Buyya. Data storage management in cloud environments: Taxonomy, survey, and future directions. *ACM Computing Surveys (CSUR)*, 50(6):91, 2017.

[44] Houssem-Eddine Chihoub, Shadi Ibrahim, Gabriel Antoniu, and María S Perez. Consistency management in cloud storage systems. In *Advances in data processing techniques in the era of Big Data*. CRC PRESS, 2014.

[45] Philip A Bernstein, Vassos Hadzilacos, and Nathan Goodman. *CONCURRENCY CONTROL AND RECOVERY IN DATABASE SYSTEMS*. Addison- Wesley, 1987.

[46] Ahmed K Elmagarmid. *Database transaction models for advanced applications*. Morgan Kaufmann Publishers Inc., 1992.

[47] Eric A Brewer. Towards robust distributed systems. In *PODC*, volume 7, 2000.

[48] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM Sigact News*, 33(2):51–59, 2002.

[49] Victor Chang. Towards a big data system disaster recovery in a private cloud. *Ad Hoc Networks*, 35:65–82, 2015.

[50] Gabriel Tarasuk-levin, Patrick William Penzias Dirks, Ilia Langouev, and Curt Kolovson. Maintaining consistency using reverse replication during live migration, June 6 2017. US Patent 9,672,120.

[51] Madhumita Bharde, Suparna Bhattacharya, and Dileep Deepa Shree. Store edge ripplestream: Versatile infrastructure for iot data transfer. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[52] Erin N Bournival, David L Black, Saar Cohen, Assaf Natanzon, and Mark J Halstead. Maintaining stored data consistency of a plurality of related virtual machines across a plurality of sites during migration, May 16 2017. US Patent 9,652,333.

[53] Guy Pardon and Cesare Pautasso. Consistent disaster recovery for microservices: the cab theorem. *IEEE Cloud Computing*, 2017.

[54] Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10):2266–2279, 2013.

[55] Amandeep Singh Sohal, Rajinder Sandhu, Sandeep K Sood, and Victor Chang. A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments. *Computers & Security*, 74:340–354, 2018.

[56] Victor Chang and Muthu Ramachandran. Towards achieving data security with the cloud computing adoption framework. *IEEE Trans. Services Computing*, 9(1):138–151, 2016.

[57] Ali Shoker, Houssam Yactine, and Carlos Baquero. As secure as possible eventual consistency. In *Proceedings of PaPoC'17*. ACM, 2017.

[58] Felipe A Lopes, Marcelo Santos, Robson Fidalgo, and Stenio Fernandes. A software engineering perspective on sdn programmability. *IEEE Communications Surveys & Tutorials*, 18(2):1255–1272, 2016.

[59] Yustus Eko Oktian, SangGon Lee, HoonJae Lee, and JunHuy Lam. Distributed sdn controller system: A survey on design choice. *Computer Networks*, 121:100–111, 2017.

[60] Ivan Farris, Tarik Taleb, Yacine Khettab, and Jae Seung Song. A survey on emerging sdn and nfv security mechanisms for iot systems. *IEEE Communications Surveys & Tutorials*, pages 1–1, 2018.

[61] Van-Giang Nguyen, Truong-Xuan Do, and YoungHan Kim. Sdn and virtualization-based lte mobile network architectures: A comprehensive survey. *Wireless Personal Communications*, 86(3):1401–1438, 2016.

[62] Fábio Botelho, Tulio A Ribeiro, Paulo Ferreira, Fernando MV Ramos, and Alysson Bessani. Design and implementation of a consistent data store for a distributed sdn control plane. In *2016 12th European Dependable Computing Conference (EDCC)*, pages 169–180. IEEE, 2016.

[63] Mohamed Aslan and Ashraf Matrawy. Adaptive consistency for distributed sdn controllers. In *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, pages 150–157. IEEE, 2016.

[64] Ermin Sakic, Fragkiskos Sardis, Jochen W Guck, and Wolfgang Kellerer. Towards adaptive state consistency in distributed sdn control plane. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2017.

[65] Werner Vogels. Eventually consistent. *Communications of the ACM*, 52(1):40–44, 2009.

[66] Sathiya Prabhu Kumar. *Adaptive Consistency Protocols for Replicated Data in Modern Storage Systems with a High Degree of Elasticity*. PhD thesis, Conservatoire national des arts et metiers-CNAM, 2016.

[67] Alexandros G Dimakis and Kannan Ramchandran. Network coding for distributed storage in wireless networks. In *Networked Sensing Information and Control*, pages 115–134. Springer, 2008.

[68] Jay Kreps, Neha Narkhede, Jun Rao, et al. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, pages 1–7, 2011.

[69] Shangguang Wang, Yali Zhao, Jinlinag Xu, Jie Yuan, and Ching-Hsien Hsu. Edge server placement in mobile edge computing. *Journal of Parallel and Distributed Computing*, 2018(in press).

[70] Atakan Aral and Tolga Ovatman. A decentralized replica placement algorithm for edge computing. *IEEE Transactions on Network and Service Management*, 15(2):516–529, 2018.

[71] SK Alamgir Hossain, Md Anisur Rahman, and M Anwar Hossain. Edge computing framework for enabling situation awareness in iot based smart city. *Journal of Parallel and Distributed Computing*, 122:226–237, 2018.

[72] Yuan Zhang, Lin Cui, Wei Wang, and Yuxiang Zhang. A survey on software defined networking with multiple controllers. *Journal of Network and Computer Applications*, 103:101–118, 2017.

[73] Chun-Wei Tsai, Shi-Jui Liu, and Yi-Chung Wang. A parallel metaheuristic data clustering framework for cloud. *Journal of Parallel and Distributed Computing*, 116:39–49, 2018.

[74] Surat Teerapittayanon, Bradley McDanel, and HT Kung. Distributed deep neural networks over the cloud, the edge and end devices. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 328–339. IEEE, 2017.

[75] Srinivas Sridharan, Karthikeyan Vaidyanathan, Dhiraj Kalamkar, Dipankar Das, Mikhail E Smorkalov, Mikhail Shiryaev, Dheevatsa Mudigere, Naveen Mellempudi, Sasikanth Avancha, Bharat Kaul, et al. On scale-out deep learning training for cloud and hpc. *arXiv preprint arXiv:1801.08030*, 2018.

[76] Ta-Yuan Hsu, Ajay D Kshemkalyani, and Min Shen. Causal consistency algorithms for partially replicated and fully replicated systems. *Future Generation Computer Systems*, 86:1118–1133, 2018.