IJCTA, 10(8), 2017, pp. 443-457 ISSN: 0974-5572
© International Science Press

Data Consistency Rationing Model: Survey and New Paradigm of Selective Consistency

Vijay Ghanekar* and Shraddha Phansalkar**

ABSTRACT

Data consistency is the most important part of CAP (consistency-availability-partition) theorem. Data consistency is rationed to the permissible levels to achieve higher availability and performance. This paper surveys the significant consistency rationing techniques presented in the latest research work. This paper also proposes a new concept of selective consistency to leverage the consistency of selective data/transactions with better performance. This technique applies the consistency level to the selected data objects and is further extensible to selective transactions in NoSQL data stores like Mongo DBthat have lowerassurance of consistencyto guarantee other performance metrics. In this paper, we conceptualize as well as present design of the selective consistency model on MongoDB data store to apply higher levels of consistency only on selected objects of transactional application to improve its performance.

Keywords: selective consistency, TPC-C, response time, MongoDB, read-write concern, replication.

1. INTRODUCTION

Relational Database Management System (RDBMS) guarantees higher levels of consistency that is strict consistency.Not Only Structured Query Language (No-SQL) and Big Data stores are now a days emerging data stores option used by customers and companies. The default consistency of NoSQL data stores is eventual consistency [4], [16] or weaker levels of consistency. No-SQL databases are increasingly used in big data and web based application for their configurable consistency property and easy methods of partitions. Also NoSQL supports variable data objects in a single collection.

CAP (Consistency-Availability-Partition Tolerance) theorem [11] statesthat only two properties from all three can be achieved at one time. The application having consistency and availability then need to compromise on partitioning and application having partitioning and availability sacrifices the consistency of data. And if consistency and partitioning tolerance is given by data object then availability of data object is not supported at that moment. However the distributed applications deployed on cloud [1] often compromise on consistency to achieve the availability.

Data consistency has different levels. Strong level or strict consistency [4, 16] returns most recent read operation from database, whereas sequential level of consistency is weaker than strict consistency and needsa read from a location to return the value of the last write to that location. Casual consistency is weaker model than sequential consistency and all read and causally related write operations appear in the same order. Eventual consistency model allows concurrent access to the replicas for updates and reads and this is weaker consistency model than casual consistency.

The No-SQL data stores are generally used to deploy the big data applications which demand availability and scalability. There is an exhaustive work done to survey how this can be achieved. All of these works [1, 2, 4, 5, 9, 10, 11, 12, 14, 15] require consistency to be lowered to achieve these new metrics.

* Symbiosis Institute of Technology, Symbiosis University, Pune, *Email: vijay.ghanekar@sitpune.edu.in*

** Symbiosis Institute of Technology, Symbiosis University, Pune, *Email: shraddhap@sitpune.edu.in*

This work also conceptualizes and designs selective consistency model for MongoDB and can be used for transactional application. Clients use replicas for read and updates. Every replica take some amount of time (T) to update the write of each data-object. “Selective data consistency” model applies strict consistency to a subset of data objects. Selective data consistency model promises better performance as against the consistency models that apply the same consistency constraints to all the data objects.In selective data consistency model,we select particular data object and apply configurable consistency as strong, sequential and weak consistency level as needed by user demand or transaction purpose.

An eventually consistent read might not reflect the results of a recently completed write and consistent read returns a result that reflects all writes that received a successful response from operation.

2. RELATED WORK

Researchers have proposed significant consistency rationing techniques and they are presented in the latest research work. The suggested and existing methodologies to achieve consistency in transactional application onbig data stores have some limitations. To overcome these,we refer related work in this paper and then present a new concept of *selective consistency* to improve performance. We also refer to some of the significant works in the literature which narrows down our discussion to the problem of consistency rationing.

Further study on measuring the performance and consistency levels elaborate the idea about selective consistency.

2.1. Review of Literature

2.1.1. A Survey of Large Scale Data Management Approaches in Cloud Environments

Author Sherif Sakr, Anna Liu, Daniel M. Batista, and Mohammad Alomaridescribeabout cloud technologies and cloud services models [10]. This model consists of three services “Infrastructure as a Service (IaaS)”, “Platform as a service (PaaS)” and “Software as a Service (SaaS)”. IaaS has hardware resources as servers ad virtual machine (VM) storage and network bandwidth. PaaS is platform as a service in which developers can write their applications. Maintenance, load-balancing, scaling operations are performed on PaaS. Software as aService provides applications to use through internet. Paper describes about cloud deployment models as private cloud and is strictly used by a single organization. Community cloud is a model in which its infrastructure is shared by many organization. Public cloud is model in which infrastructure is made open to all users e.g. Google, Amazon etc. Hybrid cloud is a combination of two or more clouds.Cloud methodologies as grid computing and virtualizationare briefed in this paper.

Goals and challenges of cloud data management systems are briefed and popular cloud data stores are elaborated in this paper. Google Big table is a distributed storage system and presents a data model based on key-value pairs with special mention of its replication system. Yahoo PNUTS supports Yahoo applications and is a scalable data-store which supports world-wide replication system to offer better availability. Amazon offers Dynamo DB, S3 and Simple DB in the public cloud services for databases and offers highest availability with reduced levels of consistency. Amazon also offers infrastructure where users can deploy their own Virtual Machine space on different nodes.

2.1.2. Big Data: Review, classification and Analysis Survey

Author K.Arun and Dr. L. Jabasheela describeabout big data, its architecture and characteristics like volume, velocity and variety [9]. Volume of data means data is available in larger size of gigabyte or terabyte. Velocity of big data is a speed of data generation, processing and collection. Because of e-commerce applications, the speed of data generation is increased. Variety in big data can be any type of data like image, audio, video, text etc. and all types of data can be stored in one collection. Classification of big data for business application helps for decision making using different data mining techniques. The classification of big data is in such manner gives solutions to business problems. According to the problem solution for business, data mining algorithms are used to make the analysis and processing. In this paper, analysis methodology clarifies benefits, challenges and need of high performance data-stores of big data and NoSQL. Big data increases sales in ecommerce applications, improve services and risk management factors. The paper also briefs about the challenges faced during the use of big data management in structured and un-structured format.

For the analysis and decision making, mining algorithms were used which help system to find outlier data. Association rule techniques are used to find interrelation between data collected in stores. Clustering techniques are used to group the similar data that helps to improve searching of data. Classification and regression techniques analyse customer fulfilments that are needed for ecommerce application.

2.1.3. Perspectives on the CAP Theorem

Author Seth Gilbert and Nancy A. Lynch in*[11]* 2012, discussed about CAP (consistency-availability-partition) theorem. The paper explains theoretical context and practical implementation of CAP.

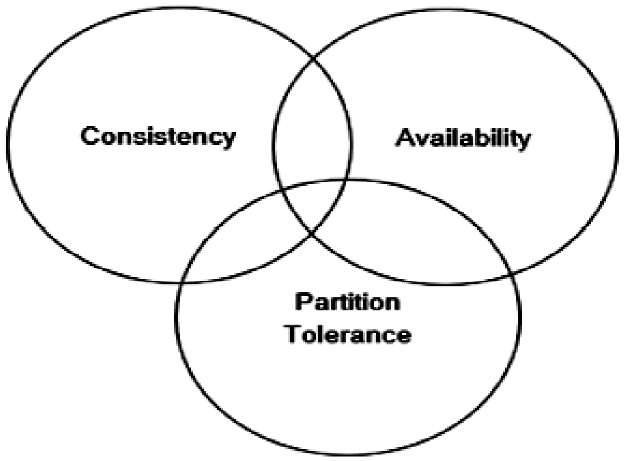


Figure 1: CAP Theorem

Theoretically only two from CAP guarantees can be achieved at one time shown in figure [1]. Consistency means the each replica returns correct response.Second aspect of CAP is availability which means server should always be available for read-write. And third aspect is partition tolerance that one server gets divided into chunks for better distribution of data, but at same time communication between partitions occurs synchronously. Practically, it is achieved by offering tolerance and choosing between data consistency and availability.

The future scope of CAP is maintaining the balance between safety and live-ness in system by achieving consistency, availability or partitioning. This paper puts forth a very interesting problem area of research.

2.1.4. The Development of a Benchmark Tool for NoSQL Databases

Author Ion Lungu, Bogdan George Tudorica develop a new tool [7]for RDBMS (MySQL)database as NoSQL database like TPC-C benchmark[17]. The OLTP (Online transaction processing) and OLAP (On-line Analytical Processing) are processing standards and need to be deployed on NoSQL databases to give performance.A model is to be evaluated using a benchmark. There are many benchmarks which tell us the performance of the applied models. Benchmarks define the workload which are used to test the model with the average load. TPC-E and TPC-H are other two benchmark with TPCC. TPC-H is an ad-hoc, decision support benchmark while TPC Benchmark™ E (TPC-E) is a new On-Line Transaction Processing (OLTP) workload developed by the TPC[17].

There exists a performance measurement application benchmark for NoSQL data stores but to compare this with RDBMS database there is no any existing benchmark. This work developed a new benchmark for RDBMS (MySQL).Proposed system was implemented on MongoDB and MySQL database with some web side scripting language to calculate results.

The study done on MySQL as well as NoSQL databases show that time required to give results is much higher in MySQL (RDBMS)whereas time taken to fire query on NoSQL is measurably less.

2.1.5. Trading replicated consistency for performance and availability: an adaptive approach

Author Chi Zhang and Zheng Zhang said that trade-off betweenperformance, availability and replication consistency is often necessary[14]. This workanalyses the problem of optimal performance/availability for a given consistency level. System proposes adaptive approach for consistency level maintenanceat replicated servers or distributes servers. The implementation model is designed with FRACS (Flexible Replication Architecture for a Consistency Spectrum) and built on application interface layer.FRACS is quantitative model bases on window strategies which defines consistency levels by measuring missed updates on replicas or replicated data objects. It measures inconsistency by obsoleteness in data or data objects with missed updates. Every replica has an update window with maximum number of updates and these updates can be buffered without consensus. In conclusion, the system gives balancing in consistency, availability and performance in replicated servers.

2.1.6. Adaptive Leases: A Strong Consistency Mechanism for the World Wide Web

Author VenkataDuvvuri, Prashant Shenoyand Renu Tewari [13]proposed weakerlevel of cache consistency mechanisms, which is supported by web-proxy and sacrifices stronger level consistency mechanisms to support the big data stores. In general, data is accessed by users by using browsers. As users are humans and can tolerate receiving false data and manually correct it using browser reloads, weaker cache consistency mechanisms were adequate for this purpose.

But internet has some delay in request and response. Due to thesedelays in the internet, the consistency technique cannot be stronglyconsistenteven when required in this idealized sense.

Duvvuri et al. present *lease* approach by referencing the cache consistency approach in distributed file system. System aims on searching optimal lease timing for data-objects. Lease time is strictly given to the owner of that lease time and no other user can do modifications to that hold data objects without passing messages or replay to that leaser. Success of lease time is determined by crucial parameters on which algorithms of lease is computed with different technologies used for different kind of data objects. This technique allows configurability of adopting different lease period to different objects.

Strong level of data consistency can be achieved by *server-driven* or *client-driven* techniques. The ideal approach, referred to as *server-based invalidation*, which needs to notify proxy servers to inform when data getschanged. Currently available strong consistency mechanisms either demand a large state space overhead or a large control message overhead. Server allocate a lease to each request from a proxy. Lease has its time interval in which the server needs to notify to proxy server about modifications in original server (primary copy). This time interval can be used for performance checking.Analytical model forcalculatingthe lease duration is developed using duration based model with state space overhead and control message overhead although.

Adaptive policies used for calculation of lease duration are age based lease model in which server can cut down the invalidate message (lease time) which need to send by granting shorter lease to changing object(dynamic) while longer lease to long lasting objects (static).For designing the prototype, they used simulation environment which can be compared with other cache consistency models schema and also found the desired or intended results of model.

2.1.7. Design and Evaluation of a Continuous Consistency Model for Replicated Services

Author Haifeng Yu and Amin Vahdat in this paper [15] explores the definition of space between strong traditional and optimistic consistency models for replicated data models. Paper states that the relation between consistency, availability and performance is moving towards optimistic consistency from strong consistency.

They develop a *conit-based* continuous consistency model to capture the consistency spectrum of a replicated data-object using three application-independent metrics namely numerical error, order error, and staleness.

Consistency can be given as combination of numerical errors, order errors and staleness data. Numerical errors are variation between conit and final image value of data, where conit is a physical or logical unit of consistency. The numerical error limits the weight of the writes that can be applied across all replicas before being propagated to a given replica. Order error counts the difference between orders in which updates are made on local replicas.Order error limits the number of tentative writes that can be outstanding at any one replica. Staleness is the difference between current time and acceptance time of last or old write on conit transaction. Staleness placed a real time bound on the delay of write propagation among replicas

One of benefit of this model is consistency of conit can be achieved by per-replica set basis. Every individual replica has its own consistency level for a conit. Numerical errors can be made relaxed by pushing other replicas rather than replica which is used frequently. Since this error is bounded by pushing updates to replicas, if replicas are busy to manage communication then it may get delayed. On the same lines as numerical error, if order and staleness read relax, communication also may get delayed.

Current system is implemented using Java and RMI (Remote Method Invoke) for communication. Proposed system is applicable to applications like Airline Reservation system and QoS Load Distribution etc. In such applications, the continuous consistency techniques are applied and some better results are measured. They also present the design and implementation of a middleware layer that enforces arbitrary consistency bounds among replicas using these metrics.

2.1.8. IDEA: An Infrastructure for Detection-based Adaptive Consistency Control in Replicated Services

Author Yijun Lu, Hong Jiang said that[12] if consistency and scalability are at equal level then it is more important to achieve adaptability of consistency. Proposed paper gives adaptable consistency by adjusting level as per user preference on demand. E-business applications can lose some consistency for scaling their business.

Any system can run many applications with different consistency which can be configurable as per time requirements. IDEA is implemented on some applications like distributed white board system, airline ticket booking etc. The proposed design have two layer infrastructure (top/bottom layer)infrastructure to detect and resolve inconsistency foreach shared file or objectwhich gives accuracy in calculations. Top layer is much smaller in size so it is easy to detect and resolve the problem. Each file has its own consistency level so for each file’s top and bottom layer hastheir own consistency and different top layers do not interface with each other.

Paper proposed adaptive consistency control with “on-demand”, “hint-based” and “fully-automatic” methodologies for consistency. In on-demand consistency technique user explicitly requests consistency when existing consistency is not satisfying the requirements of users. In hint-based consistency method user is asked to give *hint* about consistency requirement for that data object.The system indicatesits tolerance level and IDEA will keep consistency level above this tolerance level. IDEA itself fires the new consistency level when it goes down from hint level defined by user for particular object. Fully-automatic methodology gives best-effort level by adjusting the consistency automatically.

In the application programming interface (API) of IDEA uses strategies like setting resolution strategy, setting hints for hint-based applications, setting frequency for background resolution and demand active inconsistency resolution.

IDEA is used on applications such as distributed white board system, airline ticket booking system with two layers and different consistency controls. Because of two layer structure, response time that is measured for application is very less and performance factor is much higher than using normal consistency level for all data objects.

In conclusion, paper validates the adaptive interface of IDEA and shows the performance of IDEA in terms of low resolution delay and lower communication cost it incurred.

2.1.9. Consistency Rationing in the Cloud: Pay only when it matters

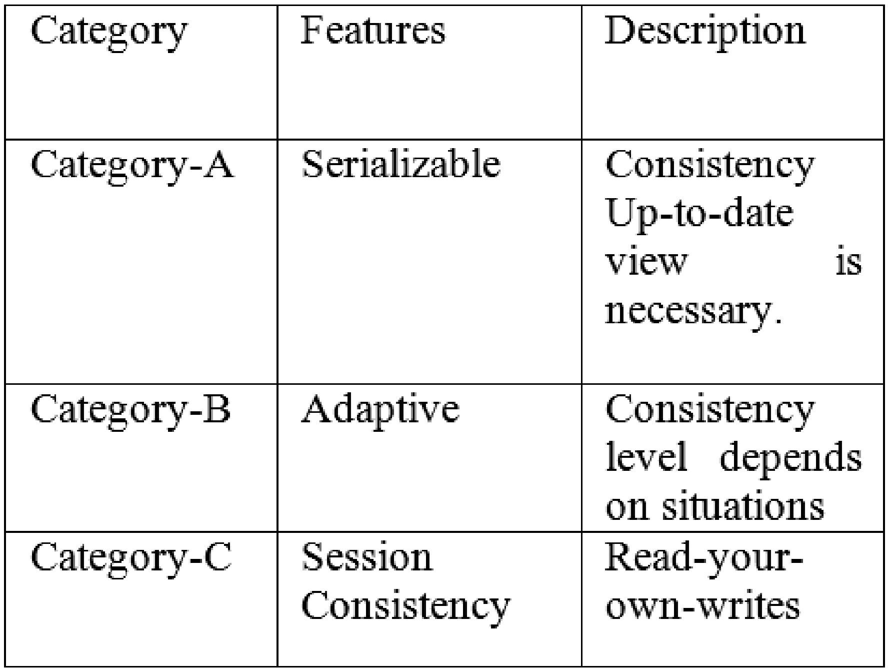
Author Tim Kraska, Martin Hentschel, Gustavo Alonso, Donald Kossmann proposed good terminology regarding consistency rationing[11]. This technique dynamically adapts the level of consistency by examining data. Consistency rationing allows system to get consistency at lower cost. The approach is collaborativeand hence many users can work on the same document at same time.

Consistency Rationing uses three categories of data as shown in figure 2.

“Category-A-Serializable” in which data should be kept in this category if consistency and up-to-date view is necessary.

“Category-B–Adaptive” this category defines that there is wide spectrum between data types and applications which require consistency level depends on situations.

“Category-C-Session Consistency” when user logs in to application, it has its session in which read can be done with read-your-own-writes monotonicity. But if session terminates, a new session may not immediately see write of last session.



Category	Features	Description
Category-A	Serializable	Consistency Up-to-date view is necessary.
Category-B	Adaptive	Consistency level depends on situations
Category-C	Session Consistency	Read-your-own-writes

Figure 2: Three categories of data consistencyproposed in [9]

In the implementation model, architecture and protocol implementation, logical logging, meta data and statistical components are considered. Protocol implementation uses additional buffer layer for pages in the queue on the top of amazon S3 dataset. It maintains session consistency which saves highest commit timestamp for each page. Protocol maintains serializability with two phase locking protocol (2PL). 2PL is widely used in heavy conflict rates and applied if there exists conflicts in system. Logical logging contains ID of changed record and the operations executed for that record. Every system maintains its own metadata that can be used to point the data object locations stored on replicas in S3.Static components are responsible for collecting information at runtime and which is collected locally.Experiments are made up with different policies, response time, cost per transaction and fixed threshold values.

2.1.10. Characterizing and Adapting the Consistency-Latency Trade-off inDistributed Key-value Stores

Authors Lewis Tseng, Son Nguyen and Nitin Vaidya [1] made a significant contribution trade-off consistency in distributed key-value stores. The paper extends the traditional CAP theorem to PCAP theorem where the first 'P' implies the probabilistic guarantees.

With consistency-latency trade-off, the paper defines new terminologies as *freshness* and *staleness* reads, with probability. The paper extends to derive the service level agreements (SLA) of consistency and latency with the contemporary key-value data stores. The paper also proposes controlling three parameters in the workload to observe positive and negative effects on consistency and its probability.The control factors are delays in reads, repairing the reads and consistency level. The paper further extends to define the SLA for consistency and latency to geo-distributed multiple data-centers where the replicas are now located on the distributed data-centres. The results are compared to the PBS systemand PCAP outperforms PBS in all the agreements.

2.1.11. AMulti-key Transactions Model for NoSQL Cloud Database Systems

Authors AdewoleOgunyadeka, Muhammad Younas, Hong Zhu, ArantzaAldea in [2]IEEE conference of 2016 proposed multi key transaction model for big data stores. This system presentsa system with standard transaction and high level of consistency. The working model have extra layer in architecture which manages all transactions. This is validated using MongoDB which results into strong consistency and good performance.

Proposed system have architecture with Transaction Processing Engine (TPE), Data Management Store (DMS) and Time Stamp Manage (TSM).

Transaction processing engine is responsible for executing multiple key transaction in system. It includes query processing from client side, managing data, providing schema level information and maintains relation between entities of data. Data management system actually implements the scalability property of big data on MongoDB. It also implements isolation protocol. Timestamp manager manages scheduling and flow of transaction in system. It communicates with TPE and DMS to schedule the operations of different executions.

The experimental model takes following elements into account for evaluation as "transaction overhead", "consistency" and "timestamp" which allows system to perform all transactions concurrently without conflicting with each other.

2.1.12. Tunable consistency guarantees of selective data consistency model

Author Shraddha P. Phansalkar and Ajay R. Dani proposed [4]tunableconsistency guarantees on selected data object from entire data sets.

The notion of consistency index (CI) was introduced to calculate ratio of correct read to total number of reads. The work proposes tunable consistency guarantees with a workload scheduler which introduces minimal latencies in the read operations (following updates) to assure convergence of the updates to all data replicas.Implementation of tunable consistency is evaluated on Amazon SimpleDB and TPCC transaction workload schema. System applies consistency to the selected data objects instead of applying to entire data set. This technique improves the performance and give better results in response time for all the transactions.

2.1.13. The many faces of consistency

Author Marcos K. Aguilera and Douglas B. Terry proposed [3]abstract model of consistency which has setting multiple clients performing operations. A transaction includes simple read-write, start and commit operations.

Every system has its state which contain data items and their values. Caches and replicas are also considered as state of program. On the other hand, operations are performed on execution of states.

In this paper, authors brief about two consistencies. The first one is “state consistency” which commit that the user gets correct state which user wants; and secondly “operation consistency” which gives properties that operation returns correct results or not.State consistency deals with transactional states which are requiredfor user applications. In operation consistency, system deals with transactional consistency levels like read-write, sequential or weak consistency and reference equivalence for serializability. The paper also explains the comparison and analysis of the different consistency levels.

2.1.14. Cloud Auditing and Consistency Service

AuthorQin Liu, Guojun Wang, and Jie Wupresented asurvey [6]of the existing cloud auditing system and try to overcome drawbacks of loose consistency by proposing new system with heuristic auditing strategy (HAS) which give correct reads to particular data store.

They proposed Consistency as Service (CAAS), two-level auditing structure needs loosely synchronized clock and put forth local cloud auditing, global cloud auditing and data upload module auditing modules.

The system allows user to select best cloud service provider among available and also count the consistency that the service provider is guaranteeing to provide. Most of the applications expect that the cloud should provide casual consistency service where the data gets updated on replicas depending on the causal dependencies. Existing system cloud provides eventual consistency with which user is not able to read updated data from any replica in the replica set. So proposal is that the user update should be reflected on all the servers or replicas so receiver user can read correct read and not steal read.

This paper uses local and global cloud auditing technique. Implementation of data on cloud cannot be monitored by all users due to virtualization technique. Thus user can’t check whether each replica in data cloud has new versioned or old versioned data. This system however permits the user in audit cloud to check cloud consistency.Proposed system allows user to maintain user operation table (UOT) which keeps track of logical and physical vectors as well as operations inserted in UOT. A write operation has either zero or more reads. From the value of a read, we know the logical and physical vectors of write.

Thus in conclusion, system allows user to maintain UOT, so the consistency is provided by CSP (Cloud Service Provider) can be checked and modified as user perspective.The local and global cloud use techniques as consistency as a service (CAAS) model, user operation table (UOT), and read-write consistency for data objects.

Implemented system gives better results with effectiveness of providing consistency on local consistency model. In UOT, the update read and writes need to be saved to avoid garbage collection. The future work of this paper is to study theoretical model which improves consistency of cloud.

2.1.15. A Survey on Improving Cloud Consistency Using Audit Cloud

Author Vasanti Kulkarni, MeghaBansodegivestheoretical understanding[5] of novel Consistency as a Service (CaaS) model provided by the cloud service provider.

In the implementation, system considers assumptions, dependencies and constraints designs andproposes the architecture to improve cloud consistency.

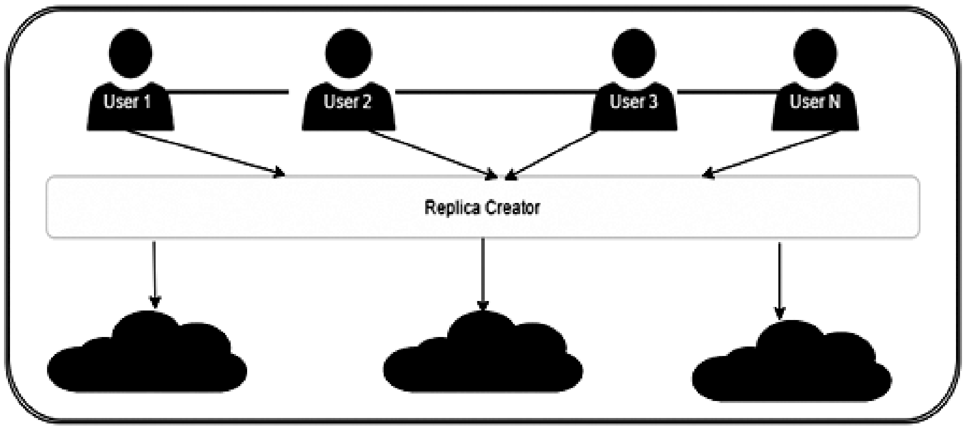


Figure 3: Architecture for improving consistency proposed in [5]

This architecture shown in figure [3] consistency improvement techniques that how userscan create replicas on clouds. Using these replicas, middleware between cloud and user increases the performance and consistency as the replicas maintain their own consistency on top of the cloud using consistency as service audit cloud.

We now summarize the highlights of these significant consistency rationing techniques with special remark on their limitations.

Summary Table of Literature Survey

We hereby present the summary of the significant contribution and analyse the methods they used to ration consistency to guarantee better performance.

Table 1
Comparison of papers with highlights and limitations.

Sr. No	*Paper Title*	*Highlights*	*Limitations*
1	A Survey of Large Scale Data Management Approaches in Cloud Environment	1. Introduces cloud service models: Infrastructure as a service, platform as a service, software as a service. 2. Cloud deployment on public, private, hybrid cloud. 3. Briefs about Cloud service provider as Google HBase, Yahoo PNUT, Amazon S3/Simple DB.	1. Less emphasis on CA conflict, scalability, consistency and low cost processing of data on cloud. 2. Less emphasis on NoSQL limitations such as programming model, support and expertise.
2	Big Data: Review, classification and Analysis Survey	1. Emphasizes on Big data characteristics like Volume, Velocity and Variety. 2. How to execute decision making using data mining techniques	1. It does not give analysis of data emphasizing on data mining techniques.
3	Perspectives on the CAP Theorem	1. Only two from consistency (C), Availability (A), Partition Tolerance (P) can be achieved at one time	1. In network CAP not able to handle tolerating attacks. 2. Not applicable to wireless communication systems.

(*contd...Table 1*)

Sr. No	*Paper Title*	*Highlights*	*Limitations*
4	The Development of a Benchmark Tool for NoSQL Databases	1. Comparing all RDBMS and NoSQL data stores to develop new benchmark tool as TPC-C 2. Uses OLTP and OLAP transactions 3. Uses different programming paradigms to develop a benchmark.	1. Not yet implemented on Oracle and MS SQL.
5	Trading replicated consistency for performance and availability: an adaptive approach	1. Designed FRACS (Flexible Replication Architecture for a Consistency Spectrum) model to measure consistency. 2. Inconsistency is measured in terms of obsoleteness. 3. Replica is assigned an update window which is maximum number of updates that are executed without consensus.	1. Does not guarantees strong level of consistency.
6	Adaptive Leases: A Strong Consist-ency Mechanism for the World Wide Web.	1. Guarantees and shifts from Weak level to strong level of consistency in cloud. 2. Lease time is introduced to achieve higher level of consistency. 3. Lease has its time interval in which server informs proxy about updates. 4. Achieved by *server-driven* or *client-driven* techniques	1. Proxy cache may store steal data so if user read data from cache then in may be wrong or not updated data.
7	Design and Evalua-tion of a Continuous Consistency Model for Replicated Services	1. *Conit-based* continuous consistency model 2. Consistency can be given as combination of numerical errors, order errors and staleness data 3. Numerical errors- limits the weight of the writes that can be applied across all replicas 4. Order error- limits the number of tentative writes 5. Staleness- placed a real time bound on the delay of write propagation among replicas	1. Application consistency level in response of wide area network is not implemented.
8	IDEA: An Infrast-ructure for Detect-ion-based Adaptive Consistency Control in Repli-cated Services	1. Two layer infrastructure (top/bottom layer) detect and resolve inconsistency in different levels a) On-demand- users explicitly request consistency resolution b) Hint-based- system asks users to give hints about their approximate consistency requirements c) Fully-automatic- improves consistency with best effort	1. Implemented on selected distributed systems. 2. Two layer structure could not detect inconsistency if implemented on different distributed systems.
9	Consistency Rationing in the Cloud: Pay only when it matters	1. Dynamically adapt the level of consistency, three categories of data based on consistency a) Category-A – serializable b) Category-B – Adaptive c) Category-C-Session Consistency	1. Only probabilistic consistency guarantees in proposed system. 2. Automatic optimization for all categories.
10	Characterizing and Adapting the Con-sistency-Latency	1. Freshness and staleness reads 2. Probabilistic consistency, latency, probabilistic latency	1. Only probabilistic consistency

(*contd...Table 1*)

Sr. No	*Paper Title*	*Highlights*	*Limitations*
	Trade-off in Distributed Key-value Stores	3. Different algorithms introduced to measure latency and trade-off in the same	
11	AMulti-key Transactions Model for NoSQL Cloud Database Systems	1. Transaction model of the application on MongoDb is presented. The architecture has three major components: 2. Transaction Processing Engine (TPE) 3. Data Management Store (DMS) 4. Time Stamp Manager (TSM).	1. Only proposed system not implemented.
12	Tunable consistency guarantees of selective data consistency model	1. Adaptive consistency guarantees 2. No-SQL data stores used 3. Leverages consistency of selected data items by scheduling the workload i.e. increasing read latency just enough for an N-replicated system.	1. Proposed system implemented only on Amazon Simple DB. 2. Selective data consistency level on different big data stores not discussed.
13	The many faces of consistency	1. Consistency levels as strict, weak, sequential explained 2. State consistency - commit the state which user wants 3. operation consistency - gives properties that operation returns correct	1. Proposed new levels of consistency but implementation and affect to performance is not discussed.
14	Cloud Auditing and Consistency Service	1. With heuristic auditing strategy (HAS) which gives correct reads to particular data store 2. local cloud auditing 3. global cloud auditing 4. Maintains user operation table (UOT) which keeps track of logical and physical vectors	1. Theoretical models of cloud consistency. 2. Cost model for local and global cloud auditing
15	A Survey on Improving Cloud Consistency Using Audit Cloud	1. Theoretical understanding of novel consistency as a service	1. Not implemented in open source language as Java.

Proposed Model: Selective consistency level on MongoDB

I. Selective Consistency: Concept

The figure [4] shows that different big data stores with different guarantees of consistency (C), availability (A) and partition tolerance (P). For proposed system implementation, we select No-SQL data stores where consistency is lowered to weaker levels. We choose:

a. Amazon Simple DB (A-P) and
b. MongoDB (C-P)

They both have partition tolerance as common service and differ in availability and consistency guarantees of CAP theorem.

MongoDB guarantees higher consistency at the document level with different levels of read and write concern, but does not support joins or consistency in the multi-key transactions.

SimpleDB guarantees consistency at the eventual level that is the replicas may not agree with the values at all point of time.

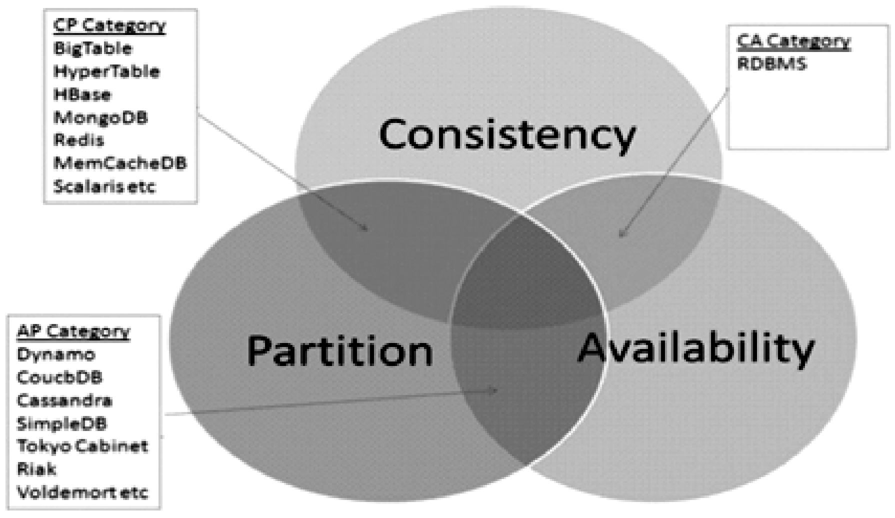


Figure 4: Existence of MongoDB and Simple DB in CAP

MongoDB offers more flexibility with combinations read and write consistency. Hence we present the prototype on MongoDB.

II. Consistency level: MongoDB

Our proposed model of "selective consistency" explains the consistency level configuration for critical data object. We deployed TPC-C Entity-Relation diagram on the replicated servers and clients of MongoDB.

There are multiple OLTP benchmark but we select TPC-C [7, 17]as it has standard weightage for each transaction and can give better performance as compared to other benchmarks.

In MongoDB, each write operation is executed from Master node and read can be through either master or slave.

MongoDB uses a readConcern of "local" which does not guarantee that the read data would not be rolled back and a readConcern of "majority" to read data that has been written to a majority of replica set members and thus cannot be rolled back.

For Write Concern if value of "w" is 1 then itRequests acknowledgement that the write operation has propagated to the standalone mongod or the primary in a replica set. W: 1 is the default write concern for MongoDB.

For Write Concern if value of "w" is 0 then Requests no acknowledgment of the write operation. However, w: 0 may return information about socket exceptions and networking errors to the application.

Table 2
Read-Write Concern combinations in MongoDB

Read Write	*Local*	*Majority*
1	Consistent	Consistent
0	Consistent	Inconsistent

Table 2 shows the possible consistency levels with read and write concerns. Combinations of writes "0" and "1" with read "local" "majority" is shown in this table.

Figure 5 shows the data structure of primary and secondary replicas with read-write concerns. It also shows the response of read and request to write to primary as well as secondary nodes.

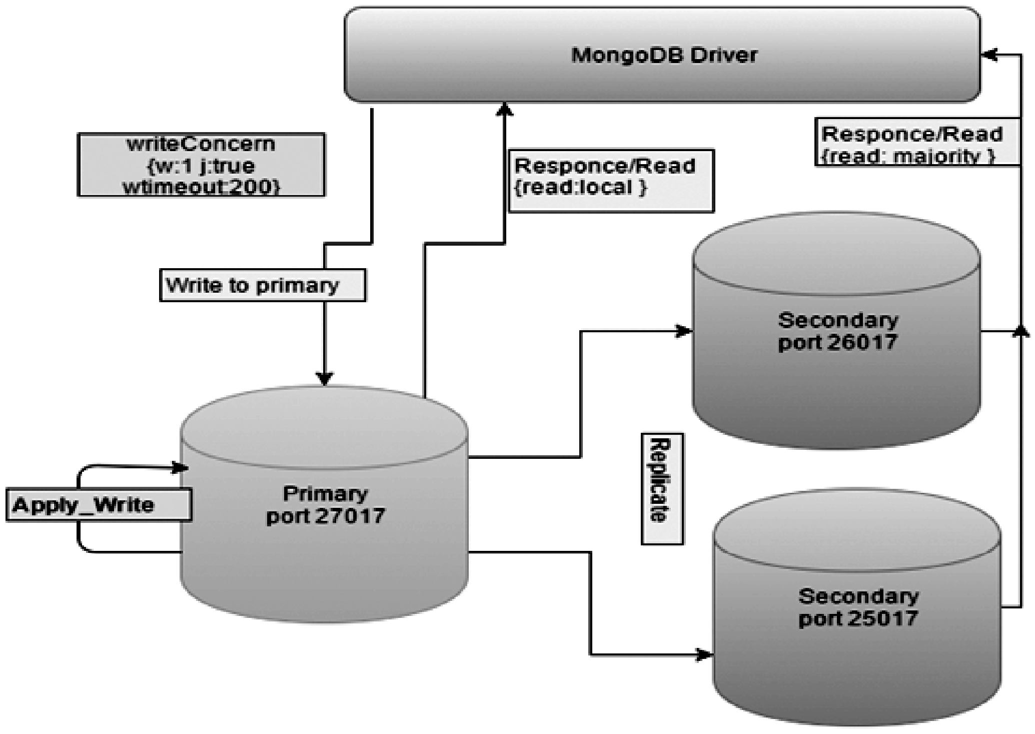


Figure 5: Read-Write Concerns of MongoDB replica set with 2 copies

3. ARCHITECTURE OF SELECTIVE CONSISTENCY MODEL ON MONGODB

Architecture shown in Figure 6 gives overview of implementation design with TPC-C transactions and Timestamp manager to avoid conflicts of transaction accesses on one data store and sequentially execute the transactions.

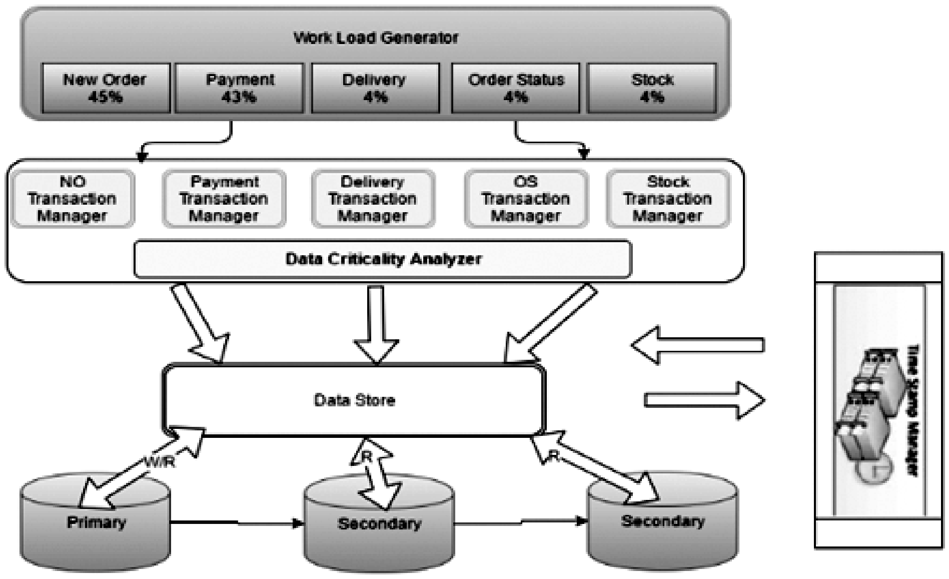


Figure 6: Architecture of selective consistency model on MongoDB

Architecture having three tire:

1. Transaction manager: Each transaction has its own transaction manager and for monitoring we build transaction monitoring manager (TMM) on top of it.

   This tire contain Data criticality analyser which is used to analyse the criticality of the data in the business. The analyser uses statistical model to generate an index depending on the sage and access pattern of the given data.

2. Random load generator: It distributor manager create a random load on server with standard transaction and distribute the transactions to respective managers.
3. Time stamp manager: Uses time-stamp to achieve sequential concurrency and conflict handling.

The query analyser is also proposed where the critical transactions will have their data buffered in the caching of the application server instead of fetching it from the data store.

We then prove hypothesis is that selective consistency will lead to better performance in respect of response time in MongoDB as well as Amazon Simple DB andsupport configurable consistency by applying it to selective data andtransactions critical to the application.

3. CONCLUSION

The purpose of this paper is to review the trends in studies of consistency within the past few decades and existing work in data consistency. This review highlights the significant approaches in consistency models used in big data and NoSQL. This review helps us to propose selective data consistency for data objects, implement and also measure the performance of the application in response time. We propose consistency model to leverage guarantees of only few data itemsand hence thisapproach is considered as “Selective data consistency”. We implement the prototype of this methodology on two big data stores as Amazon Simple DB and further extend it to MongoDB and comparethe performance results. The prototype shows promising results with respect to responsiveness. The prototype needs to further enhance with the critical data analyser and query analyser which is part of the future work.

REFERENCE

[1] Sakr, Sherif, Anna Liu, Daniel M. Batista, and Mohammad Alomari. “A survey of large scale data management approaches in cloud environments.” IEEE Communications Surveys & Tutorials 13, no. 3 (2011): 311-336.

[2] Arun, K., and Dr L. Jabasheela. “Big Data: Review, Classification and Analysis Survey.” International Journal of Innovative Research in Information Security (IJIRIS) 1, no. 3 (2014): 17-23.

[3] Gilbert, Seth, and Nancy Ann Lynch. “Perspectives on the CAP Theorem.” Institute of Electrical and Electronics Engineers, 2012.

[4] Lungu, Ion, and Bogdan George Tudorica. “The development of a benchmark tool for nosql databases.” Database Systems Journal BOARD 13 (2013).

[5] Zhang, Chi, and Zheng Zhang. “Trading replication consistency for performance and availability: an adaptive approach.” In Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on, pp. 687-695. IEEE, 2003.

[6] Duvvuri, Venkata, Prashant Shenoy, and RenuTewari. “Adaptive leases: A strong consistency mechanism for the world wide web.” In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 2, pp. 834-843. IEEE, 2000.

[7] Yu, Haifeng, 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, p. 21. USENIX Association, 2000.

[8] Lu, Yijun, Ying Lu, and Hong Jiang. “IDEA:: an infrastructure for detection-based adaptive consistency control in replicated services.” In Proceedings of the 16th international symposium on High performance distributed computing, pp. 223-224. ACM, 2007.

[9] Kraska, Tim, Martin Hentschel, Gustavo Alonso, and Donald Kossmann. “Consistency rationing in the cloud: pay only when it matters.” Proceedings of the VLDB Endowment 2, no. 1 (2009): 253-264.

[10] Rahman, M.R., Tseng, L., Nguyen, S., Gupta, I. and Vaidya, N., 2015. Characterizing and adapting the consistency-latency tradeoff in distributed key-value stores. arXiv preprint arXiv:1509.02464.

[11] Ogunyadeka, Adewole, Muhammad Younas, Hong Zhu, and ArantzaAldea. “A Multi-key Transactions Model for NoSQL

Cloud Database Systems.” In 2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService), pp. 24-27. IEEE, 2016.

[12] Phansalkar, Shraddha P., and Ajay R. Dani. “Tunable consistency guarantees of selective data consistency model.” Journal of Cloud Computing 4, no. 1 (2015): 1.

[13] Aguilera, Marcos K., and Douglas B. Terry. “The many faces of consistency.” Data Engineering (2016): 3.

[14] NAGALEELA, ANKALA, and T. ANUSHA. “Cloud Auditing and Consistency Service.” (2015).

[15] Kulkarni, Vasanti, MeghaBansode, Gaurav Mirje, SanketSolanke, and Spurti S. Shinde. “A Survey on Improving Cloud Consistency Using Audit Cloud.” computing 4, no. 1 (2015).

[16] http://delab.csd.auth.gr/~dimitris/courses/mpc_fall05/papers/invalidation/ieee_tkde99_data_consistency_pitoura.pdf

[17] http://www.tpc.org/