ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 4, Issue 1, January 2016

A Novel Approach for Transaction Management in Hetrogeneous Distributed Database Systems

Dheeraj Bhimrao Lokhande[1], Prof.Dr.R.C.Thool[2]

Research Scholar, Solapur University, Walchand Institute of Technology, Solapur, India

Shri Guru Govind Singhji, College of Engineering, Nanded, India

ABSTRACT: Database framework benchmarks like TPC-C and TPC-E concentrate on imitating database applications to think about various DBMS executions. These benchmarks utilize precisely built inquiries executed inside of the setting of exchanges to practice particular RDBMS components, and measure the throughput accomplished. Cloud administrations benchmark systems like YCSB, then again, are intended for execution assessment of disseminated NoSQL key-esteem stores, early illustrations of which did not bolster exchanges, thus the benchmarks use single operations that are not inside exchanges. Late executions of web-scale dispersed NoSQL frameworks like Spanner and Percolator, offer exchange components to oblige new web-scale applications. This has uncovered a hole in standard benchmarks. We distinguish the issues that should be tended to while assessing exchange support in NoSQL databases. We depict YCSB+T, an expansion of YCSB that wraps database operations inside of exchanges. In this system, we incorporate an approval stage to distinguish and evaluate database irregularities coming about because of any workload, and we accumulate measurements that measure value-based overhead. We have planned a particular workload called Closed Economy Workload (CEW), which can keep running inside of the YCSB+T structure. We impart our experience to utilizing CEW to assess some NoSQL frameworks.

KEYWORDS: YCSB, Elasticity, Availability, Scale.

I. INTRODUCTION

Distributed computing or utility processing [1] is bit by bit picking up prevalence as the arrangement design of decision for application in vast built up ventures and little new companies and organizations alike. The innate circulated nature of these designs combined with the utilization of ware equipment has offered ascend to the advancement of new information administration technologies extensively characterized under the name NoSQL databases which incorporate Amazon S3 [2], Google BigTable [3], Yahoo! PNUTS [4] and numerous others. These frameworks exploit the circulated way of the organization stage to bolster web-scale stockpiling and throughput while enduring disappointment of some part machines. As tradeoff, they commonly offer customers less in inquiry expressivity (for instance, they may not perform joins) and less value-based and consistency ensures. Early samples of NoSQL frameworks, for example, AWS' S3 [2] or Amazon's inside use Dymano [4] give little if any backing to exchanges and data consistency. Some permit a read operation to return to some degree stale information, and regularly just possible consistency [5] or course of events consistency [4] is offered individually. Later business offerings like Windows Azure Storage (WAS) [6] and Google Cloud Storage (GCS) [7] permit the exchange like gathering of numerous operations however limit this to include either a solitary thing, or an accumulation of things that are arranged. Another way to deal with better backing for application rationale lies in wealthier operations, for example, test-and-set or contingent put. A few plans have as of late seemed to defeat the constrained value-based semantics of these NoSQL databases. Cases incorporate Google Percolator [8], G-Store [9], Cloud-TPS [10], Deuteronomy [1], Megastore [2] and Google Spanner [3]. By what means can these plans be assessed? Conventional information administration stages were measured with industry standard benchmarks like TPC-C [4] and TPC-E [5]; these have focused on copying end-client application situations to assess the execution (particularly the throughput, and throughput in respect to framework expense) of the basic DBMS and

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 4, Issue 1, January 2016

application server stack. These benchmarks run a workload with questions and redesigns that are performed in the setting of transactivities, and the respectability of the information should be checked amid the procedure of the execution of the benchmark. In the event that the information is debased, the benchmark estimation is dismisses altogether.

For web-scale data management, especially the available-despite-failures key-value stores in the NoSQL category, a new benchmark YCSB [6] has become accepted. The focus of this benchmark is raw performance and scalability; correctness is not measured or validated as part of the benchmark, and the operations do not fall within transactions (since these systems may not support transactions nor guarantee data consistency). YCSB is actually a flexible framework within which the workload and the measurements can be extended.Our proposed benchmark (called YCSB+T) retains the flexibility of YCSB by allowing the user to implement the DBinterface to their database/data store; it allows for additional operations apart from the standard read, write, update, delete and scan; it enables defining workloads in terms of these operations; and most importantly it allows these operations to be wrapped into transactions. Further, there is a validation stage to specify consistency checks to be conducted on the database/data store after the completion of the workload in order to detect and quantify transaction anomalies.Our approach is intended to fill the gap between traditional TPC-C-style benchmarks that are designed for transactional RDBMSs and the non-transactional HTTP/web-service bench-marks which have no ability to define transactions.

II. RELATED WORK

In this area we give an outline of NoSQL database systems and afterward depict ways to deal with giving value-based access to them. We likewise condense the key elements of the YCSB benchmark for NoSQL frameworks.

A. Overview of NoSQL database systems

Distinctive disseminated key-esteem information stores or NoSQL information store display a blend of various execution, versatility, profit capacity attributes and structures. While these frameworks might vary in design and execution they endeavor to accomplish the accompanying attributes. This is examined in further detail in the first YCSB paper [6].Scale-out: Distributed NoSQL databases can bolster the vast information sizes and high demand rates since they can spread the solicitation burden and information to be put away over countless servers each facilitating a part of the information. A fruitful scale-out instrument can adequately spread the information and customer solicitations over these machines without uncovering any bottlenecks.

Flexibility: Elasticity empowers a framework to include limit by including new servers and spreading the heap adequately while the framework is as yet running. It supplements the scale-out abilities of a framework.
High accessibility: Commodity equipment is inclined to disappointment making the accessibility of the framework a vital prerequisite. This is especially imperative when these frameworks host information having a place with numerous occupants. It is difficult to give all the attractive elements into one framework and everyone makes distinctive configuration and architecture decisions. Case in point it is difficult to at the same time accomplish consistency, accessibility and parcel resistance [7]. In this manner, these frameworks need to make the accompanying tradeoffs to display the qualities said above: Read versus compose execution: High read execution is achievable with great arbitrary I/O throughput while higher compose execution can be accomplished utilizing attach just log-organized capacity organized. Inactivity versus strength: Persisting information to circle accomplishes sturdiness however increments compose inertness significantly. Not synching keeps in touch with the plate decreases inactivity and enhances throughput however lessens toughness guarantees. Synchronous versus offbeat replication: Replicating information enhances execution, framework accessibility and maintains a strategic distance from information misfortune. This should be possible either synchronously or no concurrently. Synchronous replication builds compose and redesign dormancy while offbeat replication reduces idleness additionally diminishes consistency ensures brought on by stale information.

Information apportioning: Data can be put away in succession arranged or segment situated capacity structure. Line

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 4, Issue 1, January 2016

arranged storage structures are suitable for applications that need to get to a huge extent of the fields in every information record amid an exchange while segment stockpiling structures are suited to applications that utilization a little number of the aggregate fields in every record and are suitable for circumstances when the records have an expansive number of fields. We are occupied with frameworks that pick consistency over accessibility and give a value-based interface to the application. Specifically, we might want to assess frameworks that backing multi-thing exchanges.

B. Transactions in NoSQL database systems

One way to address this is to implement a relational database engine to provide the query capabilities and Transaction support with the raw data stored in a distributed key-value store [3]. This is suitable for applications that require a complete SQL interface with full transaction support. The performance and transaction throughput of the system is limited only by the underlying data store and queue implementation.
The relative simplicity of the the data store API makes application development simple and robust. These applications most often use write-once-read-many (WORM) data access pattern and function well under eventual consistency guarantees. However, there are increasing demands for applications that are built to run on the same data that require better consistency guarantees across multiple records.One way to solve this is to implement transactional capabilities within the data store itself. The data store manages the storage as well as transaction management. The Spanner [3] system developed at Google is a distributed NoSQL data store that supports native transactions built on top of BigTable [3].COPS [1] and Granole [2] implement a distributed key-value store that provides a custom API to enable transactional access to the data store. HyperDex Warp [4] is another high-performance distributed key-value store that provides a client library that supports linearizable transactions and simplifies access to data items in the data store which maintains multiple versions of each data item.

III. ARCHITECTURE AND PROPOSED SYSTEM MECHANISM

The Yahoo! Cloud Services Benchmark (YCSB) is an extensible benchmarking structure that is generally used to quantify the execution and adaptability of expansive scale disseminated NoSQL frameworks like Yahoo! PNUTS [4] and customary database administration frameworks like MySQL. Figure 1 demonstrates the engineering of YCSB+T. The light shaded rectangles speak to YCSB segments and the dim hued rectangles speak to increases and upgrades actualized by YCSB+T.

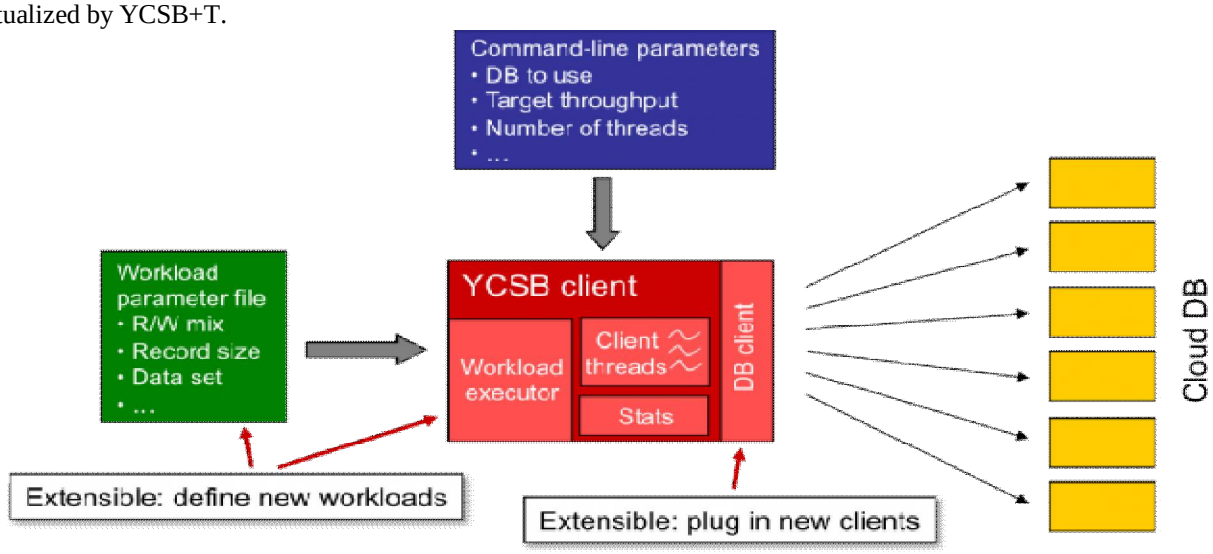

Fig No 01. YCSB+ T Architecture

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 4, Issue 1, January 2016

Explanation-
The workload executor instantiates and initializes the workload then loads the data into the database to be tested or executes the workload on the database using the DB client abstraction. The CoreWorkload workload defined by default with the YCSB framework defines different mixes of the simple database operations read, update, delete and scan operations on the database. The doTransactionRead() operation reads a single record while the doTransactionScan() operation is used to fetch more than one record identified by a range of keys. Data records are inserted using the doTransactionInsert() method and doTransactionUpdate() is issued to perform record updates in the database. The doTransactionReadModifyWrite() reads a record determined by the key sequence generator, assigns a new value and writes it back to the data store.
Operations on records in the database are defined by a distribution that is specified in the workload parameter file. The YCSB distribution defines six workloads called workloads A, B, C, D, and E that generate different read-intensive, update-intensive and scan-intensive loads on the database respectively.The distribution also defines DB client abstractions for various NoSQL and other data stores and databases including Cassandra, HBase, MongoDB and MySQL.The YCSB client starts the specified number of client threads each of which use the workload to perform the specified database operations using the DB client abstraction. The client can be passed command-line parameters to alter the behavior of the client.Addition, they discuss two additional tiers to address availability and replication.We propose two additional tiers for evaluating the transactional overhead on throughput of the data store, and transactional consistency of the database as a result of concurrent execution of the benchmark workload.Tier 1 and 2 are described in detail in the YCSB paper [16] and tiers 3 and 4 are discussed as future work. Here we discuss our two new tiers in further detail.

A. Tier 5 - Transactional Overhead

The Transactional Overhead tier of the benchmark focuses on measuring the overhead of the individual database operations (exposed via the DB client class) when they are executed in a transactional context. The latency of database CRUD and scan operations are measured in both transactional and non-transactional modes. In addition, the latency for DB start(), abort() and commit() method calls are also gathered in both modes.The Transactional Overhead tier of the YCSB+T framework aims to characterize the overhead of transactions and identify the impact of each database operation that executes within and outside a transactional context.This tier of the benchmarking framework requires the workload executor to capture the metrics associated with the database transactional operations start(), commit() and abort() and the workload methods to capture the latencies measured for the CRUD and scan database operations.The latency of each raw database CRUD and scan operation along with the start, commit and abort operations captured makes it possible to measure the overhead of transactional access to the data store.

B. Tier 6 - Consistency

The primary reason to execute database operations within the context of a transaction is to ensure that the ACID transaction properties and consistency in particular are preserved. The isolation level used to execute transactions has a significant impact on the performance of the operations.The Consistency tier of the YCSB+T framework is designed to detect consistency anomalies in the data introduced during the execution of the workload and quantify the amount of anomalies detected.To achieve this, a database validation phase is added to the workload executor that is implemented as a method in the workload class. The validation stage goes over all the records in the database and applies an application-defined check to the content of the database. The validation stage also calculates an application-specific anomaly score to quantify the degree of inconsistency detected. It is intended that a score of zero should mean that the data is entirely consistent (as from a serialisable execution).

IV. FUTURE WORK

We are taking a shot at extra workloads that will target particular abnormalities that are seen at different exchange separation levels [26] and create measures to evaluate these. We will run these against our customer composed exchange library and conveyed key-esteem store and also freely accessible cloud administrations like Google Cloud

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Vol. 4, Issue 1, January 2016

Storage (GCS) and Windows Azure Storage (WAS).We mean to discharge the source code for these workloads and the upgrades made to the YCSB+T structure. We will investigate the likelihood of consolidating them into the primary YCSB source tree so that the more noteworthy group can advantage.

V. CONCLUSIONS

We have displayed YCSB+T, an expansion of the Yahoo! Cloud Services Benchmark (YCSB), with the capacity to wrap numerous database operations into exchanges that presents an approval stage that executes in the wake of running the workload. Further, we depicted a workload called the Closed Economy Workload (CEW) that is utilized to assess the execution of an information store utilizing that comprises of read and read-adjust compose operations that reenact an application situation and later accepts the consistency of the information and evaluates the oddities if recognized. Our benchmark is suitable to execution test frameworks giving exchanges in cloud-based NoSQL frameworks. We utilize it to gauge the execution and approve the rightness of our own key-esteem store and our customer facilitated exchange convention and library that gives value-based access to it. YCSB+T can be utilized to perform logical comparison between contending information stockpiling arrangements and empowers the application designer to characterize a workload that recreates the application nearly. The workload approval stage can be utilized to accept the consistency assurances of the framework and evaluate the oddities identified.

REFERENCES

[1] B. Hayes, "Cloud computing," Commun. ACM, vol. 51, pp. 9–11, July 2008. [Online]. Available: http://doi.acm.org/10.1145/1364782.1364786
[2] "Amazon S3 API Reference," 2011. [Online]. Available: http://docs.amazonwebservices.com/AmazonS3/latest/API/
[3] F. Chang et al., "Bigtable: A Distributed Storage System for Structured Data," ACM Trans. Comput. Syst., vol. 26, no. 2, pp. 1–26, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1145/1365815.1365816
[4] B. F. Cooper, R. Ramakrishnan et al., "PNUTS: Yahoo!'s hosted data serving platform," Proc. VLDB Endow., vol. 1, pp. 1277–1288, August 2008. [Online]. Available: http://dx.doi.org/10.1145/1454159.1454167
[5] W. Vogels, "Eventually consistent," Queue, vol. 6, pp. 14–19, October 2008. [Online]. Available: http://doi.acm.org/10.1145/1466443.1466448
[6] B. Calder et al., "Windows Azure Storage: a highly available cloud storage service with strong consistency," in SOSP'11, 2011, pp. 143– 157. [Online]. Available: http://doi.acm.org/10.1145/2043556.2043571
[7] "Google Cloud Storage," 2013. [Online]. Available: https://developers.google.com/storage/docs/overview
[8] D. Peng and F. Dabek, "Large-scale incremental processing using distributed transactions and notifications," in OSDI'10, 2010, pp. 1–15. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924961
[9] S. Das et al., "G-Store: a scalable data store for transactional multi key access in the cloud," in SoCC '10, 2010, pp. 163–174. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807157
[10] W. Zhou et al., "CloudTPS: Scalable Transactions for Web Applications in the Cloud," IEEE Transactions on Services Computing, 2011.
[11] J. J. Levandoski et al., "Deuteronomy: Transaction support for cloud data," in CIDR '11.
[12] J. Baker et al., "Megastore: Providing Scalable, Highly Available Storage for Interactive Services," in CIDR, Jan. 2011, pp. 223–234.
[13] J. C. Corbett, J. Dean et al., "Spanner: Google's globally-distributed database," in OSDI '12, 2012, pp. 251–264. [Online]. Available: http://dl.acm.org/citation.cfm?id=2387880.2387905