The Journal of Systems and Software 101 (2015) 43–58

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A scalable generic transaction model scenario for distributed NoSQL databases

Ramesh Dharavath*, Chiranjeev Kumar

Department of Computer Science and Engineering, Indian School of Mines, Dhanbad 826004, Jharkhand, India

ARTICLE INFO

Article history:
Received 21 February 2014
Revised 22 September 2014
Accepted 20 November 2014
Available online 28 November 2014

Keywords:
Atomic transaction
Column-oriented distributed databases
NoSQL databases

ABSTRACT

With the development of cloud computing and internet; e-Commerce, e-Business and corporate world revenue are increasing with high rate. These areas not only require scalable and consistent databases but also require inter database transaction support. In this paper, we present, a scalable three-tier architecture along with a distributed middle-ware protocol to support atomic transactions across heterogeneous NoSQL databases. Our methodology does not compromise on any assumption on the accuracy of failure modalities. Hence, it is suitable for a class of heterogeneous distributed systems. To achieve such a target, our architectural model exploits an innovative methodology to achieve distributed atomic transactions. We simulate this architectural setup with different latency tests under different environments to produce reliable impact and correctness.

1. Introduction

Distributed computing infrastructures were emerged as the deployment platforms for a variety of applications. Increasingly, desktop and mobile applications are using distributed infrastructure to take advantage of high scalability characteristics. Massive usage of internet produces huge amount of data. These vast volumes of data are also called Big Data. This information can be used in many ways. This Big Data is beyond the capability of traditional databases. New type of databases have been introduced to deal with this Big Data, these are called column oriented databases and also known as NoSQL which are scalable, eventual consistent, and reliable. Due to these features NoSQL databases like Big-Table (Chang et al., 2006, 2008), Cassandra (Lakshman and Malik, 2010), HBase (Vora, 2011), MongoDB (Membrey et al., 2010), and Amazon S3 (DeCandia et al., 2007) are successfully used by almost all internet giants. NoSQL databases offer great scalability and availability for applications.

Relational database systems (RDBMSs) also dominate the environment by providing set of services which refer variety of requirements. But, these require analytical processing and decision support. However, some trepidation has recently materialized towards RDBMSs. First, it has been argued that, there are cases where RDBMSs performances are not passable. Second, the structure of the model is considered to be too rigid and not useful in some instances with the arguments that call for the semi structured data (Abiteboul et al., 1999). At the same time, the full power of relational models with multifaceted transactions and queries would not require in some perspectives, where simple operations are quite enough (Stonebraker and Cattel, 2011). Also, in some instances, ACID consistency and complete consistency assured by relational model is not essential. It has been observed that many internet application areas related to networking domain need both scalability and flexibility in construction, while being fulfilled with operations and weak forms of consistency. With these drives, a number of new systems are not following the RDBMS models. Their futures are limited and support simple operations with limited flexibility in the structure of data. According to McKinsey Global Institute (Manyika et al., 2011), the data volume is growing at a rate of 40% per year. From this, we conclude that NoSQL is better for large data volume applications, where the date rate increases rapidly. There is wide variety of systems with NoSQL (Cattel, 2010; Chang et al., 2008) exposes a different interface models and APIs. Certainly, it has been pointed that, the lack of standard is a great trepidation for organizations in adopting any of these systems (Stonebraker, 2011), applications, and data are not portable and reusable with others. Also, each of these systems has specific objectives, and it is customized on few specific prototypes. As an outcome, it might be possible that complicated applications could benefit from the use of NoSQL systems. Traditional databases are useful for those applications, where consistency and transaction support is needed in more suitable fashion. But, the current trend applications such as corporate world, e-business and e-commerce needs database support which is not only scalable as well as consistent too. With the massive success of NoSQL databases and growth of e-commerce, e-banking and corporate data, we assume

* Corresponding author. Tel.: +91 326 2235795; fax: +91 326 2296563/2296613.
E-mail addresses: ramesh.d.in@ieee.org (D. Dharavath), k_chiranjeev@yahoo.co.uk (C. Kumar).

http://dx.doi.org/10.1016/j.jss.2014.11.037
0164-1212/

that, in future, NoSQL databases can be used in e-commerce and e-banking areas. In order to preserve atomicity and isolation related instances, in a previous work (Ramesh et al., 2012a,b, 2013), we have implemented an architectural methodology for multi row transactions on column oriented distributed databases using RDBMS. We have also proposed a suitable methodology that preserves accuracy of atomic transactions in heterogeneous distributed column oriented database environment (Dharavath et al., 2014). The observations above have motivated us to look for tools and methods that can improve the consequences of the heterogeneity in distributed NoSQL database systems and can also enable interoperability between them.

As a first step in this direction, in this paper, we present here a *"distributed three-tier architecture"* as well as *"distributed middleware protocol"* to support atomic transaction across heterogeneous distributed NoSQL databases. Proposed framework is entirely distributed; hence it is infinitely scalable, highly available, very flexible and cost effective. We also exemplify the correctness and effectiveness of our framework in different aspects.

2. Related literature

Frolund and Guerraoui (2001, 2002), have recently proposed a reliability framework called e-transaction (exactly once transaction), that provide atomicity for distributed transaction. An e- transaction framework is based on three tier architecture and protocol implementation for each tier is clearly explained. In this framework, there is a chance of single point failure and scalability issue. There have been several works that actively replicates database systems in this way. Pacitti et al. (2003), Whitney et al. (1997), Stonebraker et al. (2007), and Jones et al. (2010) proposed about performance of transaction processing in a distributed database without concurrency control mechanism. Moreover, these methodologies execute the transactions serially which is equivalent to a serial order i.e. where a node can be a single CPU core in a multi-core server (Stonebraker et al., 2007). By executing the transactions serially, no determinism due to thread scheduling of concurrent transactions is eliminated, and active replication is easier to achieve. However, serializing transactions can limit transactional throughput, since if a transaction stalls (e.g. for a network read), other transactions are unable to take over.

Other researchers have proposed techniques for providing multi-row transactions (Peng and Dabek, 2010), (Zhang and Sterck, 2010). The Percolator system (Peng and Dabek, 2010) addresses the problem of providing SI-based transaction for Big- Table (Chang et al., 2008). However, it does not address the problem of ensuring serializability of transactions. Moreover, it is intended for offline processing of data. The work presented in (Zhang and Sterck, 2010) does not adequately address issues related to recovery and robustness when some transaction fails. Kallman et al. (2008) proposed a technique which is related to main memory transaction processing. This technique has not explored in scaling the performance of transactions which were situated in main memory. Dadiomov et al. (2003), patents a system for distributed transaction processing with asynchronous message delivery. According to this, a transaction is combination of many database operations. These operations spans over many distributed databases. All operations in a transaction must be performed atomically. To implement this system authors use message queue (MQ) concept. This message queue (MQ) guarantee exactly-once in-order message delivery. For each transaction there is a coordinator. With the help of message queue (MQ) message is passed between coordinator and participating databases. In this proposed system, atomicity can be achieved using two-phase commit (2PC) protocol.

The problems of transaction serializability have been extensively studied in the form of different models (Fekete et al., 2005; Bornea et al., 2011; Cahill et al., 2009; Revilak et al., 2011; Jung et al., 2011). The work presented by Fekete et al. (2005) exemplifies about necessary conditions for non-serializable transaction executions. Based on this, many approaches have been suggested to avoid serialization anomalies. These approaches include static analysis (Jorwekar et al., 2007) as well as runtime detection of anomalies (Bornea et al., 2011; Cahill et al., 2009; Revilak et al., 2011). Methodology presented by (Bornea et al., 2011; Cahill et al., 2009; Jung et al. 2011) tend to be distrustful and can lead to unnecessary aborts. The PSSI approach (Revilak et al., 2011) avoids such problems and aborts only the transactions that lead to serialization anomalies. However, these approaches were developed in the context of traditional relational databases and (except in Jung et al., 2011) provided solutions only for centralized databases. Thomson et al. (2012) proposed a Calvin layer to support fast and scalable transactions. Calvin layer is concerned about replication and scheduling. Calvin has no single point failure. This transaction model is useful to get high throughput but replication relaxed ACID is an issue with this layer. Another method called MAV (Monotonic Atomic View) proposed by Peter et al. (2013), which requires disallowing reading intermediate writes of a transaction. But it incurs two writes for every client side write and achieves 75% of the throughput only. Han et al. (2011), made a survey on NoSQL database that explains distinct column-oriented databases like MongoDB, Cassandra, Hyper table etc. This paper also describes about data model used by NoSQL and explores features of NoSQL. In this paper, we describe a framework to provide atomicity for distributed transactions. Proposed framework is scalable and flexible in nature and recovers from failure instances. In novelty, this framework fulfils present as well as future requirements related to real world applications.

3. Proposed methodology

In this work, we describe and present a distributed architecture (depicted in Fig. 1) to provide atomic transaction instance across heterogeneous distributed column-oriented NoSQL databases. This architecture is scalable and flexible in transactional issues. This architecture is mainly inspired by Apache HBase data-store architecture (Lars, 2011). In HBase (Vora, 2011), master nodes are managed by zookeeper. Master is used for load balancing and managing region nodes. In the same way in our architecture, manager nodes are managed by zookeeper and manager is used for balancing the load among labour nodes. HBase is having the control to store and retrieve Big-Data, in other hand our architecture is used to provide atomicity for transaction across heterogeneous distributed NoSQL databases. As name implies, three-tier architecture is divided physical entities in three layers: (i) client (ii) middle-tier, and (iii) distributed database (HBase).

3.1. Client

Architecture supports multiple client nodes denoted by C_1, C_2 … C_p. User interacts with the entire architecture by utilizing the

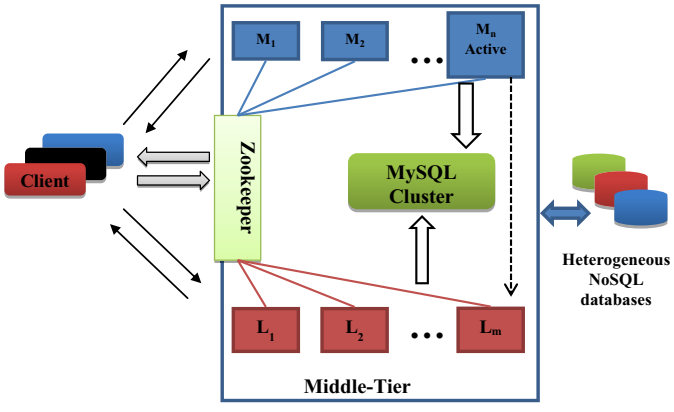

Fig. 1. Schematized three-tier architecture.