Cloud-Scale Transaction Processing with ParaDB System: A Demonstration

Xiaoyan Guo, Yu Cao, Baoyao Zhou, Dong Xiang, and Liyuan Zhao

EMC Labs China
{xiaoyan.guo,yu.cao,baoyao.zhou,dong.xiang,liyuan.zhao}@emc.com

Abstract. Scalability, flexibility, fault-tolerance and self-manageability are desirable features for data management in the cloud. This paper demonstrates ParaDB, a cloud-scale parallel relational database system optimized for intensive transaction processing. ParaDB satisfies the aforementioned four features without sacrificing the ACID transactional requirements. ParaDB is designed to break the petabyte or exabyte barrier and scale out to many thousands of servers while providing transactional support with strong consistency.

1 Introduction

Relational database management systems (RDBMS) have been extremely successful in traditional enterprise environments for more than three decades. However, RDBMSs are no longer competitive choices for cloud-scale applications, since data management in the cloud desires scalability, flexibility, fault-tolerance and self-manageability, which cannot be completely well supported by existing RDBMSs.

Recently, numerous NoSQL systems have been proposed for scalable data management in the cloud, such as Amazon Dynamo, Google BigTable, Yahoo PNUTS, Facebook Cassandra. These systems do not (fully) support SQL and usually build atop key-value data storage, where data are partitioned, replicated and then distributed over multiple nodes to achieve high performance, scalability and availability. However, all these systems guarantee only eventual consistency or other weak consistency variants.

Although a small number of large-scale Web applications can tolerate weak consistency, almost all enterprise applications demand ACID-compliant transaction processing, so as to guarantee the application correctness and simplify the application logic design. As such, Google developed Megastore and Percolator on top of BigTable for more general support of transaction processing. Other recent research works towards the similar direction include ElasTraS [1] and CloudTPS [2]. However, these systems still cannot perfectly support cloud-scale enterprise transactional applications.

We thereby present ParaDB, a scalable, flexible, fault-tolerant and self-manageable parallel database system supporting ACID transaction consistency. Unlike the aforementioned systems, ParaDB facilitates the data management in the cloud by scaling out a traditional centralized RDBMS. Moreover, since current NoSQL databases usually expose a subset of functionalities of RDBMS, a straightforward encapsulation of ParaDB can easily enable existing NoSQL applications to run with ParaDB. ParaDB is designed to break the petabyte or exabyte barrier and scale out to many thousands of servers while providing transactional support with strong consistency.

S.S. Bhowmick et al. (Eds.): DASFAA 2014, Part II, LNCS 8422, pp. 535–538, 2014.

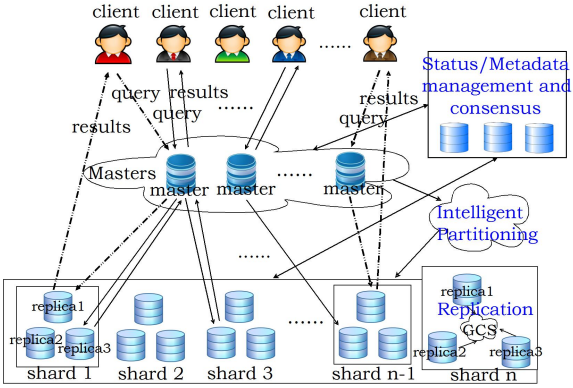


Fig. 1. ParaDB System Architecture

The major contributions of ParaDB are summarized as follows:

(1) ParaDB employs an intelligent iterative hyper-graph based database partitioning engine, which minimizes the number of distributed transactions, the major performance bottleneck. ParaDB also conducts intelligent data re-partitioning and live database migration without losing load balance and transparency.

(2) ParaDB implements an efficient eager active-active replication mechanism, which ensures strong ACID consistency in the multi-master configuration, and improves system availability and fault-tolerance at the cost of only small performance reduction.

(3) ParaDB deploys multiple identical masters to avoid the single node bottleneck and improve system scalability, with the help of an efficient and elastic multi-master metadata synchronization mechanism.

2 System Overview

Figure 1 depicts the high-level system architecture of ParaDB. There are two types of system nodes: *shard server* node and *master server* node. The shard server handles table storage and query processing. The tables in the database are partitioned, replicated and then stored at different shard server nodes. The master server serves as a query router and a distributed transaction coordinator. It only stores the metadata and system status information. There could be multiple master servers, which can concurrently accept and process clients' query requests. ParaDB supports concurrency control, transaction recovery and consistency management. ParaDB utilizes two-phase commit protocol to guarantee the ACID properties.

As shown in Figure 1, ParaDB consists of three major functional components,i.e. *intelligent data partitioning*, *eager and active-active data replication* and *metadata management and synchronization*, which are described as follows. Due to the space limitation, we ignore some technical details and the comprehensive performance study, both of which can be found in the technical report [5].

Intelligent Partitioning. ParaDB considers both the database schema and workloads to derive an intelligent scheme for data partitioning and query routing, so as to minimize the number of distributed transactions and thus optimize the system performance.

In addition, in case of dramatic system configuration changes, database upgrades or workload shifts, ParaDB also applies intelligent database re-partitioning and live data migration without losing load balance and transparency. We realize the above functionalities by constructing an intelligent iterative hyper-graph based database partitioning engine, which first analyzes the database and workload and constructs a weighted hyper-graph, then conducts iterative hyper-graph partitioning to obtain the optimal partitioning scheme. More details about the partitioning engine can be found in the paper [3].

Eager and Active-Active Replication. In ParaDB, data are replicated within the boundaries of *shard groups*. Each shard group consists of three shard server nodes, which store identical data. Eager replication means that modifications by a transaction to one replica of a data item will be applied to other replicas before the transaction commits, which guarantees strong consistency at all times. Active-active replication means that every data replica can accept read/write requests of transactions, as well as coordinate the data synchronization with other replicas. We implement an active-active eager replication protocol based on Postgres-R(SI) [4] with the help of an open-source group communication system called Spread[1], which guarantees that messages will be sent to all members of a group following a strict and user-specified order. With the atomicity and total order guaranteed by Spread, our replication protocol can ensure the transactions to be executed (or the changes to be applied) at different nodes in the same order, therefore enforcing the transaction consistency across different shard server nodes.

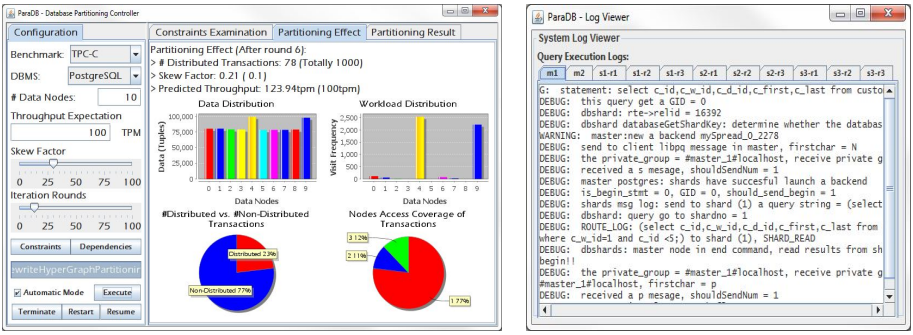
Metadata Management and Synchronization. ParaDB deploys multiple master servers to avoid single node bottleneck. Efficient metadata synchronization mechanisms are required among these masters. The metadata management component in ParaDB is responsible for synchronizing the system metadata and the snapshots (e.g. name and status) for both master nodes and shard nodes. All these distributed coordination functions are realized with the aid of Zookeeper[2], an open-source coordination service for distributed applications, which simplifies the consensus protocol design, and are elastic when facing new or unpredictable synchronization and management expectations.

3 Demonstration Scenarios

Our demonstration will illustrate both the general properties of ParaDB as a parallel database system, as well as its unique and novel features like intelligent partitioning and eager replication. For the purpose of system demonstration, we will install and run ParaDB on three physical machines, which are virtualized into two master server nodes and nine shard server nodes, managing sample databases generated by the TPC-C[3] benchmark. In this demonstration, we design four demonstration scenarios.

System Introduction. In this scenario, we will first introduce the motivation of ParaDB and explain its overall system design and novel features. Then we will dive deeply into the technical details of the data partitioning, active-active eager replication and metadata synchronization in ParaDB.

[1] `http://www.spread.org/`
[2] `http://zookeeper.apache.org/`
[3] `http://www.tpc.org/tpcc/`



(a) ParaDB Data Partitioning Controller (b) ParaDB Log Viewer

Fig. 2. ParaDB Demonstration System

Intelligent Partitioning. In this scenario, we will demonstrate how the data partitioning controller of ParaDB (shown in Figure 2a) semi-automatically and intelligently partitions and replicates database tables. We will show the audience the visualized partitioning results for the TPC-C database. Finally, we will illustrate how ParaDB conducts data re-partitioning and live migration by removing one shard server node.

Eager and Active-Active Replication. In this scenario, we will demonstrate how the eager active-active replication protocol works, by analyzing the query execution logs output by the log viewer of ParaDB (shown in Figure 2b). We will verify the correctness of our protocol by conducting a set of conflicting transactions to be executed concurrently at two shard server nodes. We will also demonstrate that shard servers of a shard group can simultaneously accept and process read/write requests of transactions.

Performance Study. In this scenario, we will demonstrate the performance and scalability of ParaDB, with the log viewer in Figure 2b. We will first conduct experiments to study the performance impact of our replication protocol. After that, we will sequentially run the TPC-C benchmark queries against three TPC-C databases containing 1000 warehouses, 2000 warehouses and 3000 warehouses respectively, in order to prove that ParaDB can scale linearly along with the database size.

References

1. Das, S., Agrawal, D., El Abbadi, A.: ElasTraS: An Elastic Transactional Data Store in the Cloud. In: USENIX HotCloud (2009)
2. Wei, Z., Pierre, G., Chi, C.-H.: CloudTPS: Scalable Transactions for Web Applications in the Cloud. In: IEEE Transactions on Services Computing (2011)
3. Cao, Y., Guo, X., Zhou, B., Todd, S.: HOPE: Iterative and Interactive Database Partitioning for OLTP Workloads. In: ICDE (2014)
4. Wu, S., Bettina, K.: Postgres-R(SI): Combining Replica Control with Concurrency Control Based on Snapshot Isolation. In: ICDE (2005)
5. Guo, X., Cao, Y., Zhou, B., Xiang, D., Zhao, L.: ParaDB: A Cloud-Scale Parallel Database System for Intensive Transaction Processing. Technical Report (2013), `https://tinyurl.com/paraDB-techreport`