Chapter 20

A Key-Based Database Sharding Implementation for Big Data Analytics

Sikha Bagui
University of West Florida, USA

Loi Nguyen
Development and Technology Center (NETPDTC), USA

ABSTRACT

In this chapter, we use MySQL Database Cluster to demonstrate and discover the capabilities of key based database sharding and provide the implementation details to build a key based sharded database system. After the implementation section, we present some examples of datasets that were sharded using our implementation. The sharded data is then used for data mining, specifically association rule mining. We present the results (association rules) for the sharded data as well as the non-sharded data.

1 INTRODUCTION

The dawn of Big Data entails the re-evaluation of data storage and analytical analysis. Traditional data storage techniques stored records as data, inserted via INSERT statements into database tables. Today, with the digital revolution, as tables have enormously grown in size storing petabytes of data, sifting through data for particular instances can consume a vast amount of resources (i.e. memory_target, sort_area_size). Data analytics is also facing new challenges. The idea of a centralized data storage has to be modified to accommodate availability, scalability, reliability and manageability. Higher performance and lower cost become a challenge. In this chapter we show how this is accommodated by database sharding.

The architecture of a sharded database is comprised of multiple computed nodes deployed across commodity servers to provide continuous uptime operation in the event of hardware or network failure. The commonly available sharding schemes are: vertical partitioning, range-based partitioning, key or hash based partitioning, and directory based partitioning. By leveraging on database sharding or horizontal partitioning, the database is divided into smaller chunks or shards across multiple data nodes in the
cluster. Each node contains and is responsible for its own subset of the data to create a shared-nothing environment.

Unlike the shared-disk clustered database, where data can be accessed from all cluster nodes and thus can cause contention during simultaneous reads and writes, shared-nothing clustered database’s nodes operate on their own subset of data. Data nodes are replicated to provide redundancy, and thus provide high availability and scalability. In a worst-case scenario, if a node, containing a subset data of a table, and the replicated node become unavailable, the other nodes, with a different subset of data, will remain online and available. The business application continues to perform transactions and access data from the remaining available data nodes. These database operations will execute successfully and transparently to the application if the application does not access the unavailable data node. The failed node can be examined, recovered, and rejoin the cluster when it is ready, without affecting any other nodes.

Due to economies of scale, the incurred cost of operating the database needs to be substantially reduced. Database sharding alleviates cost in two ways (1) cost of operation and maintenance - Total Cost of Ownership (TCO) and (2) cost of accessing and processing data. Instead of a centralized storage-area network (SAN), which introduces a single point of failure, database clustering addresses cost (1) by providing a shared-nothing environment that enables the database to scale horizontally on low cost commodity hardware. By wisely partitioning data rows horizontally in a cluster using range or hash partitioning, subsets of data can be accessed separately on each individual data node, which significantly influences the performance capacity while reducing the cost of accessing and processing the data.

Database sharding solves the issue of manageability by partitioning tables and indexes into more manageable units. Database administrators can use database sharding’s Separation of Concerns design concept to pursue the management of data. Since each node is a representation of a portion of the data, maintaining the entire data table can be accomplished in succession for each individual node without affecting any other data nodes. For example, a typical usage for a database administrator is to archive or back up outdated data to save on storage. Rather than archiving the entire data table, a database administrator could archive a single data node containing the data for a particular partition, i.e. year. The archived data can be compressed and transferred to a less-expensive storage tier at a lower TCO cost.

For many years, the perception of gaining more database throughput was by building bigger and better database servers with supplementary multi-cores CPU, faster disk drives, and higher bandwidth and low latency memory. However, according to the law of diminishing returns, all three factors, CPU, disk drive, and memory, have to be added proportionally to provide maximum performance. If only one of the three factors is increased while the others remain constant, then the system will yield lower marginal returns. As business applications grow and are stored on the cloud, databases need to provide high performance to handle the increasing workloads. Not only does database sharding alleviate the TCO cost, but it is also a factor for gaining high performance. Since sharding divides the table into smaller partitions and distributes it across multiple data nodes, the number of records in each node is smaller than whole table. This technique improves searching performance by reducing the index size and limiting the number of records to be traversed.

In this chapter, we use MySQL Database Cluster to demonstrate and discover the capabilities of key based database sharding. We will provide the implementation details to build a key based sharded database system. After the implementation section, we present some examples of datasets that were sharded using our implementation. Then we used the sharded data for data mining, specifically association rule mining. We present the association rules for the sharded data as well as the non-sharded data.
23 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the product's webpage:
www.igi-global.com/chapter/a-key-based-database-sharding-implementation-for-big-data-analytics/138512?camid=4v1

www.igi-global.com/e-resources/library-recommendation/?id=107

Related Content

Security and Privacy Issues, Solutions, and Tools for MCC
www.igi-global.com/chapter/security-and-privacy-issues-solutions-and-tools-for-mcc/162012?camid=4v1a

Overview of Big Data-Intensive Storage and its Technologies for Cloud and Fog Computing
www.igi-global.com/article/overview-of-big-data-intensive-storage-and-its-technologies-for-cloud-and-fog-computing/219362?camid=4v1a

Fog Computing Qos Review and Open Challenges
www.igi-global.com/article/fog-computing-qos-review-and-open-challenges/210568?camid=4v1a

Text, Images, and Video Analytics for Fog Computing
www.igi-global.com/chapter/text-images-and-video-analytics-for-fog-computing/204279?camid=4v1a