CitusDB Architecture for Real-Time Big Data

CitusDB Highlights
- Empowers real-time Big Data using PostgreSQL
- Scales out PostgreSQL to support up to hundreds of terabytes of data
- Fast parallel processing of queries across the nodes in the CitusDB cluster
- Reduces infrastructure cost and complexity when serving as both an operational and an analytics database

Introduction to CitusDB
CitusDB empowers real-time Big Data using PostgreSQL. CitusDB is a fully scalable, highly-available hybrid transactional and analytics database built on PostgreSQL which:

- Delivers real-time end user insights on very large data sets
- Reduces the cost and complexity of your database infrastructure
- Leverages your existing RDBMS knowledge and applications
- Provides the flexibility to use structured or unstructured data
- Elastically scales as your data grows

CitusDB remains in sync with the latest major PostgreSQL version and benefits from the most recent features and performance improvements driven by the large and dynamic PostgreSQL community. It requires no changes at the application layer when used as a drop in replacement for PostgreSQL. CitusDB extends PostgreSQL functionality and performance with features that include:

- A horizontally scalable architecture
- An advanced optimizer which pushes queries and aggregations to data for massive parallel processing
- An executor which intelligently recovers from mid-query failures, enabling high availability
- A columnar storage engine which allows superior compression and faster analytics

CitusDB serves as a hybrid transactional and analytics database. Terabytes of time series data from various sources can be ingested as the data is generated. Real-time interactive analytics applications for end users can be powered by the CitusDB query engine. By combining massive scalability with parallel query processing, CitusDB can bridge the gap between operational and analytics database, eliminating the need for complex multi-database architectures.

05/20/15
CitusDB Architecture

At a high level, CitusDB distributes the data across a cluster of commodity servers, then processes incoming analytic queries in parallel across these nodes. In the following, we will briefly explain the CitusDB architecture.

**Cluster Architecture: Master and Worker Nodes**

A CitusDB cluster is configured as one coordinator node called the “master node” and N data nodes called the “worker nodes”. The user interacts with the master node through standard PostgreSQL interfaces for data loading and querying. The master node distributes data and the queries across the worker nodes in the cluster. The master node keeps the table schemas and metadata which lets it distribute the queries intelligently and handle failures.

CitusDB pushes most of the computational workload to the cluster of worker nodes. It also minimizes the amount of data transferred on the network to prevent network I/O or the master node CPU from becoming bottlenecks. To avoid bottlenecks, CitusDB uses an advanced query planning logic which takes each user query and converts it to its commutative form. The commutative query can then be distributed across many nodes for parallel processing.

**Block Based Data Structure**

CitusDB utilizes a modular block architecture which is similar to Hadoop Distributed File System blocks but uses PostgreSQL tables on the worker nodes instead of files. Each of the PostgreSQL tables is a horizontal partition or a “shard”. The CitusDB master node maintains metadata tables which track all the cluster nodes, the locations of the shards on those nodes and statistics about the shards. The shard information is used by the CitusDB query planner to optimize the distribution of queries across the cluster.

Each shard is replicated on at least two of the cluster nodes (users can configure to a higher value). As a result, the loss of a single node does not impact data availability. The CitusDB block architecture also allows new nodes to be added to
increase the capacity and processing power of the cluster at any time without taking the cluster offline. Once new nodes are added, the system rebalances the data by moving shards into the new nodes. The system integrates the new nodes, adding to the capacity and processing power of the cluster.

Query Processing: Parallelization

When the user issues a query, the master node partitions the query into smaller queries such that each smaller query can run independently on a shard. This allows CitusDB to distribute each query across the cluster nodes, utilizing the processing power of all of the involved nodes and the multiple cores in the CPU. The master node then assigns these smaller queries to worker nodes, oversees their execution, merges their results, and returns the final result to the user. To ensure that all queries are executed in a scalable manner, the master node applies optimizations that minimize the amount of data transferred across the network.

CitusDB supports equi-JOINs between any number of tables, irrespective of their size and partitioning method. The query planner chooses the optimal join method and join order based on the statistics gathered from the partitioned tables. The system evaluates several possible join orders and creates a join plan which minimizes the amount of data to be transferred across network.

Failure Handling

Due to the CitusDB block based architecture, it can easily tolerate worker node failures. If a node fails in mid-query, CitusDB completes the query by re-routing the failed portions of the query to other nodes which have a copy of the shard. If the worker node is down permanently, users can also easily rebalance the shards from different nodes onto other nodes to maintain the same level of availability.

No worker node is an exact copy of another and replicas of the shards at a given node are distributed across multiple nodes. When a worker node is down, its workload is shared across many nodes in the cluster which minimizes the impact on overall query performance.

The master node keeps only metadata tables which are typically small (a few MBs in size). The metadata table can be replicated and quickly restored if the master node ever experiences a failure.

Data Loading and Real-time Writes

For fast data loading in batches, CitusDB provides the “\stage” command which is used to copy data from a file to a distributed table while handling replication and failures automatically. This command borrows its syntax from the client-side \copy command in PostgreSQL. The command opens a connection to the master node and fetches candidate worker nodes on which to create new shards. The command then connects to these worker nodes, creates at least one shard, and uploads the data to the shard(s). The system then replicates these shards on other worker nodes until the replication factor is satisfied and fetches statistics for the newly created shards. The command then updates the shard metadata with the master node. CitusDB also provides functionality to append the contents of a PostgreSQL table to an existing shard.

CitusDB provides an extension which automatically redirects low-latency INSERTs to appropriate shards for use cases requiring real-time writes into distributed tables. This extension also enables UPDATEs and DELETEs on distributed tables.
Columnar Storage

CitusDB provides a columnar store for PostgreSQL data, which lets users keep both row based (standard PostgreSQL) and column-based tables in the same database. The CitusDB columnar store feature is modelled after the Optimized Row Columnar (ORC) format. ORC improves upon the RCFile format developed at Facebook and delivers:

- Compression – reduces in-memory and on-disk data size by 4-6x and can be extended to support different codecs
- Column Projections – only columns which are relevant to the query are read which improves performance for I/O bound queries. In cases where tables have many columns but individual queries only involve a few columns, the performance increases can be more than 2x.
- Skip Indexes – the system stores min/max statistics for grow groups and uses them to skip over unrelated rows

The CitusDB columnar store feature is implemented using PostgreSQL’s APIs. It therefore provides support for 40+ PostgreSQL data types and allows users to create and use new types.

PostgreSQL Integration

CitusDB is not a fork of PostgreSQL. It is built using the “hooks” and API frameworks within PostgreSQL, which enable CitusDB to utilize powerful PostgreSQL internals and remain in sync with the latest version of PostgreSQL. This allows CitusDB to leverage constantly advancing PostgreSQL features and functionality while still providing powerful scalability, high availability, parallelization, and columnar storage.

The CitusDB integration with PostgreSQL covers its data types and extensions. For example, CitusDB 4.0 allows users to leverage the JSONB datatype that was introduced in PostgreSQL 9.4 to efficiently store and analyze semi-structured data. Similarly, popular PostgreSQL extensions such as PostGIS and HLL can be scaled out using CitusDB.

Applications

CitusDB is used across many industry sectors and differing company sizes. Some of the most popular uses cases are briefly described here.

Scaling out PostgreSQL

CitusDB enables current PostgreSQL users to cost effectively leverage their PostgreSQL knowledge and scale out their database with no impact on the application layer. PostgreSQL is single threaded so increasing the computing power of the database server can only support growth of the dataset to a point. If you already use the most powerful server possible, CitusDB offers a solution for continuing to grow and manage your PostgreSQL database far beyond its current limits.

CitusDB allows you to easily scale PostgreSQL horizontally by adding additional nodes to your database cluster. Increasing the capacity of your database is as easy as adding additional nodes and instructing CitusDB to reconfigure the shards across the cluster.

CitusDB can dramatically decrease query times because it parses each query and sends the query to each of the involved nodes. Because the nodes run the query in parallel, you leverage the computing power of all the nodes in the cluster, achieving query speeds that are orders of magnitude greater than is possible with a single PostgreSQL database server.
Real-time Analytics

Companies that provide customer facing big data analytics portals or business units that maintain analytics dashboards for internal use can dramatically improve their end user experience with CitusDB. CitusDB can also simplify the database infrastructure and reduce upfront and operational costs. An example is a portal for viewing mobile phone location information or online advertising effectiveness to evaluate advertising opportunities. Another example is presenting performance information for an online website management solution. However, unlike data warehousing approaches, CitusDB can manage real-time inserts so analytics can be run on current data, not on a dated snapshot of the data.

Companies that use CitusDB to provide customer facing big data analytics portals create a clustered PostgreSQL environment. Parallel processing of the queries leverages the computing power of the cluster and can deliver 100x or more improvements in query response times versus running queries on a single PostgreSQL node.

CitusDB is built on PostgreSQL 9.4 so it can manage structured or unstructured data. It can handle short requests so it can potentially replace more complex systems. In some cases, CitusDB users have found they can run CitusDB on top of their Hadoop cluster to directly generate real-time analytics without pre-processing the data before analysis.

Hybrid Transactions and Analytics Processing (HTAP)

Many companies maintain a data warehouse to collect a high volume of events and a SQL database to query post-processed data from the data warehouse. In such a situation, CitusDB can allow them to:

- Leverage their existing knowledge of SQL databases
- Reduce the complexity of their database environment
- Reduce the cost to deploy and maintain their infrastructure

CitusDB is a hybrid transaction/analytical processing (HTAP) database. It can support high volume, real-time inserts and also provide very fast queries on the real-time data. Instead of building and maintaining a data warehouse and a separate analytics database, CitusDB can replace both with a single platform. This reduces cost and complexity while simultaneously improving overall system performance.

High Availability PostgreSQL

CitusDB can provide high availability out of the box across a scalable cluster of commodity nodes for existing PostgreSQL users without the use of complex replication and failover mechanisms. CitusDB shards your database and replicates multiple copies of each shard across a cluster of commodity nodes automatically. If any node in the cluster becomes unavailable, CitusDB transparently redirects any writes or queries to one of the other nodes which houses a copy of the impacted shard.

About Citus Data

Citus Data brings the advantages of relational databases to big data while also decreasing costs and complexity. Citus Data extends PostgreSQL to seamlessly address big data challenges with familiar technology. The result is an extensible distributed SQL database with an open ecosystem that tackles heavy workloads orders of magnitude faster than what has been possible with PostgreSQL before. We make it simple for organizations to ingest, explore and aggregate their large datasets in real time. To learn more about Citus Data solutions, please call us at (415) 688-4279 x1 or email us at sales@citusdata.com now.