

01 BigTable -KirkYagami



Cloud Bigtable: An In-Depth Overview

Introduction to Cloud Bigtable

Cloud Bigtable is a fully managed, key-value, wide-column NoSQL database provided by Google Cloud Platform (GCP). It is designed to handle large-scale data workloads, offering high performance and low latency for both read and write operations. Bigtable is highly suitable for applications that require massive scalability and high throughput, making it a crucial part of Google's infrastructure for services like Gmail, YouTube, and Google Search.

- ◆ **Project**
 - ◆ **Instance**
 - ◆ **Cluster**
 - ◆ **Node**
 - ◆ **Table**
 - ◆ **Row**
 - ◆ **Column Family**
 - ◆ **Column Qualifier**
 - ◆ **Cell**
 - ◆ **Timestamp**

Key Features and Capabilities

1. Fully Managed Service:

- ◆ **No Operational Overhead:** Google manages all backend infrastructure, including scaling, replication, and maintenance, so users can focus on application development.
- ◆ **Automatic Replication:** Ensures data is automatically replicated across multiple zones or regions to provide high availability and fault tolerance.

2. Key-Value, Wide-Column Store:

- ◆ **Flexible Data Model:** Supports semi-structured and unstructured data, making it ideal for a wide range of applications, from time series data to machine learning datasets.
- ◆ **Wide-Column Format:** Data is stored in rows that can have a variable number of columns, each identified by a unique key. This allows efficient storage and retrieval of large amounts of data.

3. Optimized for High Throughput:

- ◆ **High Reads/Writes per Second:** Capable of handling millions of operations per second, making it suitable for applications with high transaction rates.

- ♦ **Low Latency:** Designed to provide sub-millisecond latency for both read and write operations, critical for real-time applications.

Bigtable in Real-World Applications

1. AdTech and Retail:

- ♦ **Google Ad Personalization:** Powers personalized ad recommendations by processing vast amounts of user data in real-time.
- ♦ **Home Depot:** Delivers personalized shopping experiences by leveraging Bigtable's ability to handle large-scale customer data.
- ♦ **OpenX:** Handles over 150 billion ad requests per day using Bigtable's high throughput capabilities.

2. Media and Entertainment:

- ♦ **YouTube:** Uses Bigtable to manage and analyze massive amounts of video metadata and user interaction data.
- ♦ **Twitter:** Analyzes ad engagement data in real-time to optimize ad placements and user experience.
- ♦ **Spotify:** Powers its music recommendation engine by processing and analyzing user listening data at scale.

3. Time Series and IoT:

- ♦ **Cognite:** Manages industrial time series data, allowing companies to monitor and optimize their operations in real-time.
- ♦ **Ecobee:** Improved the performance of its smart home data processing by 10x after migrating to Bigtable.

4. Machine Learning:

- ♦ **Tamr and Discord:** Utilize Bigtable to deliver machine learning-driven experiences by processing vast datasets quickly.
- ♦ **Credit Karma:** Makes over 60 billion predictions per day by leveraging Bigtable's scalability and low-latency data processing.

Ideal Use Cases for Cloud Bigtable

1. Fast Access to Structured, Semi-Structured, or Unstructured Data:

- ♦ Bigtable is ideal for applications that require quick access to various data types, whether structured, semi-structured, or unstructured.

2. IoT Applications:

- ♦ **Real-Time Processing and Analysis:** Bigtable excels in processing streaming data from IoT devices, enabling real-time decision-making and analytics.

3. Financial Trading Applications:

- ◆ **Real-Time Data Processing:** In financial trading, where decisions must be made in milliseconds, Bigtable's ability to process and retrieve large datasets instantly is invaluable.

4. High Performance Computing (HPC) Applications:

- ◆ **Batch Analytics and ML Training:** Bigtable is well-suited for large-scale batch analytics and training machine learning models, supporting high-performance computing needs.

Scalability and High Availability

1. Horizontal Scaling:

- ◆ **Seamless Scaling:** Bigtable supports horizontal scaling, allowing the addition of more nodes to handle increased load without disrupting service.
- ◆ **Balanced Load:** Each node can process reads and writes equally, ensuring even distribution of workload across the database.

2. Automatic Scaling:

- ◆ **Based on CPU and Storage Utilization:** Bigtable automatically adjusts its resources based on the current demands of the application, ensuring optimal performance and cost-efficiency.

3. Multi-Region Deployments:

- ◆ **Automatic Replication Across Regions:** Bigtable can be deployed across multiple regions, with automatic replication to ensure data availability and redundancy.
- ◆ **Protection Against Region Failures:** Even in the event of a regional outage, Bigtable ensures data remains accessible with a guaranteed availability of 99.999%.

Easy Migrations and Maintenance

1. Migrations from Other NoSQL Databases:

- ◆ **Live Migrations:** Bigtable supports live migrations, enabling seamless transition from other NoSQL databases with minimal downtime.
- ◆ **HBase Bigtable Replication Library:** Facilitates live migrations from HBase to Bigtable without affecting ongoing operations.
- ◆ **Dataflow Templates:** Simplify migrations from Cassandra to Bigtable, reducing the complexity and effort involved.

2. Automatic Maintenance:

- ◆ **Zero Downtime:** Maintenance operations are handled automatically by Google, ensuring that your application experiences no downtime during updates or upgrades.
- ◆ **Automatic Replication:** Ensures data is consistently replicated without any manual intervention.

Rich Application and Tool Support

- ◆ **Seamless Integration:** Bigtable integrates easily with a wide range of Google Cloud services and open-source tools, making it easier to build data-driven applications.
 - ◆ **Apache Spark and Hadoop:** Bigtable can be used as a backend for big data processing frameworks like Spark and Hadoop.
 - ◆ **Google Kubernetes Engine (GKE):** Enables containerized applications to interact with Bigtable, providing a flexible and scalable infrastructure.
 - ◆ **Dataflow and Dataproc:** Supports data processing pipelines and big data analytics with minimal setup and configuration.
 - ◆ **BigQuery:** Allows for powerful SQL-based analytics on data stored in Bigtable, combining the strengths of both platforms.

Conclusion

Cloud Bigtable is a powerful, fully managed NoSQL database that offers unparalleled scalability, low latency, and high throughput for a wide variety of applications. Whether it's powering real-time analytics, managing IoT data, or supporting machine learning workloads, Bigtable provides the necessary infrastructure to handle the demands of modern, data-intensive applications. Its integration with other Google Cloud services and ease of migration from other NoSQL databases make it a versatile choice for enterprises looking to scale their data operations effortlessly.



Each row represents a financial instrument (e.g., a stock), and columns store various data points over time.

Row Key	Date	Open Price	High Price	Low Price	Close Price	Volume	Moving Avg (50-day)	Moving Avg (200-day)
AAPL-2024-09-01	2024-09-01	150.00	155.00	149.50	154.00	25,000,000	148.00	145.00
AAPL-2024-09-02	2024-09-02	154.00	156.00	153.00	155.50	22,000,000	149.00	146.00
GOOGL-2024-09-01	2024-09-01	2750.00	2800.00	2740.00	2785.00	18,000,000	2700.00	2650.00
GOOGL-2024-09-02	2024-09-02	2785.00	2820.00	2770.00	2810.00	15,000,000	2720.00	2670.00
MSFT-2024-09-01	2024-09-01	300.00	305.00	299.00	304.00	20,000,000	298.00	295.00

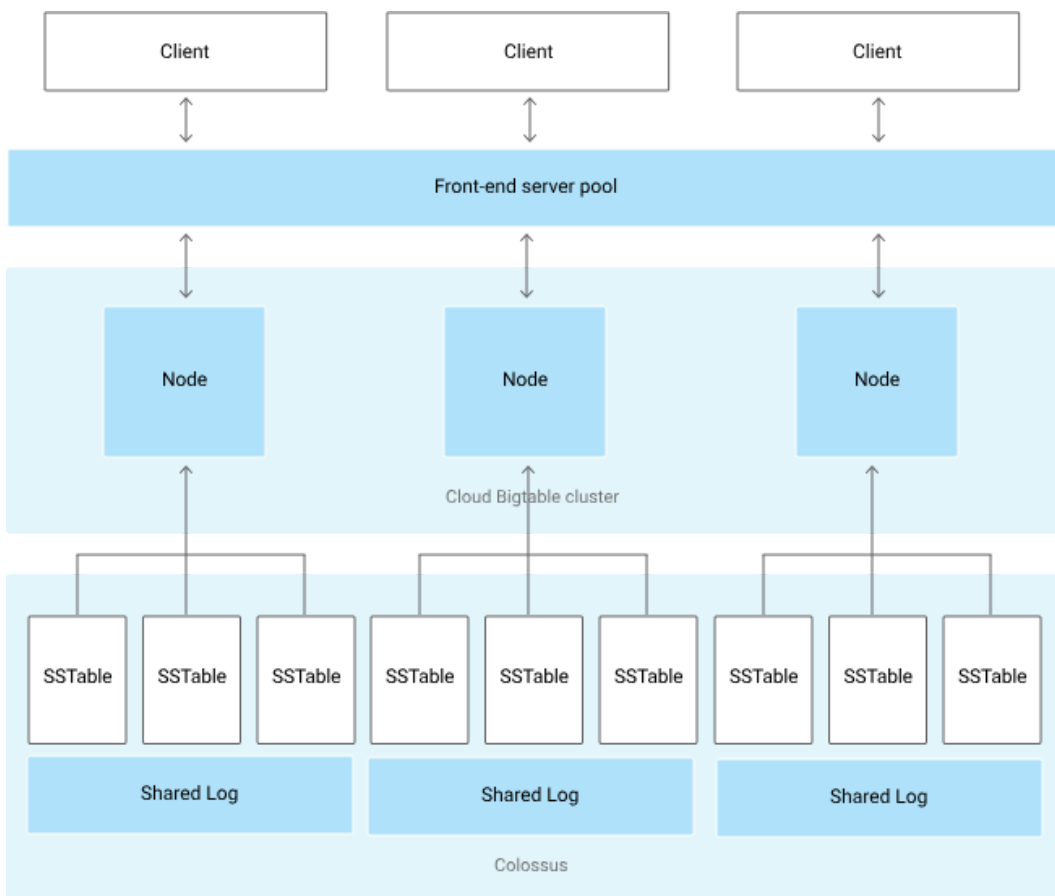
Row Key	Date	Open Price	High Price	Low Price	Close Price	Volume	Moving Avg (50-day)	Moving Avg (200-day)
09-01								
MSFT-2024-09-02	2024-09-02	304.00	310.00	303.50	309.00	19,000,000	300.00	296.00

Explanation:

- ◆ **Row Key:** Each row is uniquely identified by a combination of the financial instrument's ticker symbol (e.g., AAPL for Apple, GOOGL for Google, MSFT for Microsoft) and the date. This allows the database to efficiently store and retrieve time-series data for each financial instrument.
- ◆ **Date:** Represents the specific date for the financial data entry.
- ◆ **Open Price:** The price at which the financial instrument started trading on the given date.
- ◆ **High Price:** The highest price reached by the financial instrument during the trading session on that date.
- ◆ **Low Price:** The lowest price reached by the financial instrument during the trading session on that date.
- ◆ **Close Price:** The price at which the financial instrument finished trading on the given date.
- ◆ **Volume:** The number of shares or contracts traded during the day.
- ◆ **Moving Avg (50-day):** The 50-day moving average price, which is commonly used in technical analysis to identify trends.
- ◆ **Moving Avg (200-day):** The 200-day moving average price, another commonly used indicator to identify long-term trends.

This table illustrates how financial data can be stored efficiently in a wide-column database, enabling quick access to historical and real-time data for analysis.






Each node in the cluster handles a subset of the requests to the cluster. By adding nodes to a cluster, you can increase the number of simultaneous requests that the cluster can handle. Adding nodes also increases the maximum throughput for the cluster. If you enable replication by adding additional clusters, you can also send different types of traffic to different clusters. Then if one cluster becomes unavailable, you can fail over to another cluster.

A Bigtable table is sharded into blocks of contiguous rows, called *tablets*, to help balance the workload of queries. (Tablets are similar to HBase regions.) Tablets are stored on Colossus, Google's file system, in SSTable format. An SSTable provides a persistent, ordered immutable map from keys to values, where both keys and values are arbitrary byte strings. Each tablet is associated with a specific Bigtable node. In addition to the SSTable files, all writes are stored in Colossus's shared log as soon as they are acknowledged by Bigtable, providing increased durability.

Importantly, data is never stored in Bigtable nodes themselves; each node has pointers to a set of tablets that are stored on Colossus. As a result:

- ◆ Rebalancing tablets from one node to another happens quickly, because the actual data is not copied. Bigtable updates the pointers for each node.
- ◆ Recovery from the failure of a Bigtable node is fast, because only metadata must be migrated to the replacement node.
- ◆ When a Bigtable node fails, no data is lost.

See [Instances, Clusters, and Nodes](#)  for more information about how to work with these fundamental building blocks.



In Google Cloud Bigtable, the hierarchy of components is structured to manage and optimize the storage and retrieval of large amounts of data.

1. Project:

- ◆ **Description:** A GCP Project is the top-level container in Google Cloud that contains all the resources. Each Bigtable instance is associated with a project.
- ◆ **Purpose:** Projects allow you to manage access, billing, and the lifecycle of resources.

2. Instance:

- ◆ **Description:** An instance is a container for a Bigtable cluster and holds tables. An instance can span across multiple clusters.
- ◆ **Purpose:** It is the primary management unit for Bigtable, determining the configurations like replication, encryption, and instance type (production or development).

3. Cluster:

- ◆ **Description:** A cluster is a collection of nodes within an instance and represents a specific Bigtable service deployment in a single zone.
- ◆ **Purpose:** Clusters define the physical location of your Bigtable data and processing. You can have multiple clusters for replication and high availability.

4. Node:

- ◆ **Description:** Nodes are the processing units in a cluster. Each node has its own CPU, memory, and storage resources.
- ◆ **Purpose:** Nodes handle the read and write operations. The number of nodes directly affects the performance and throughput of Bigtable.

5. Table:

- ◆ **Description:** A table is a logical collection of data organized by rows and columns. Each table is defined within an instance and spans across all clusters in that instance.
- ◆ **Purpose:** Tables store your data and are the main unit for organizing and accessing data in Bigtable.

6. Row:

- ◆ **Description:** Rows are the fundamental data unit in a table. Each row is identified by a unique row key.
- ◆ **Purpose:** Rows store your actual data in Bigtable. All data associated with a specific row key is stored together, making row-oriented operations efficient.

7. Column Family:

- ◆ **Description:** A column family is a group of columns that are stored together in a table. It acts as a logical container for columns.
- ◆ **Purpose:** Column families help to organize columns and optimize data storage. All data in a column family is typically compressed and stored together.

8. Column Qualifier:

- ◆ **Description:** Column qualifiers are individual columns within a column family.
- ◆ **Purpose:** They store the actual data in key-value pairs under a specific row key. Column qualifiers allow you to segment different types of data within a row.

9. Cell:

- ◆ **Description:** A cell is the intersection of a row key, column family, and column qualifier. Each cell can contain multiple versions of data.
- ◆ **Purpose:** Cells store individual data values. The versioning allows you to keep track of historical data.

10. Timestamp:

- ◆ **Description:** Each cell in Bigtable can have multiple versions, each identified by a timestamp.
- ◆ **Purpose:** Timestamps allow you to store and retrieve different versions of data, enabling version control within Bigtable cells.



In Google Cloud Bigtable, the term "key-value" refers to how data is stored and accessed within the database.

Key-Value Structure in Bigtable

1. Key:

- ◆ **Description:** In Bigtable, the key is primarily the **row key**. This is a unique identifier for each row in a table. The row key is used to quickly locate and retrieve data within Bigtable.
- ◆ **Role:** The row key is crucial because Bigtable is optimized for fast lookups based on these keys. The data is sorted lexicographically by row key, making range queries efficient.

2. Value:

- ◆ **Description:** The value in Bigtable is the data associated with a row key. However, unlike simpler key-value stores, the "value" in Bigtable is more complex:
 - ◆ The value is stored in **cells**, which are defined by a combination of a row key, a **column family**, and a **column qualifier**.
 - ◆ Each cell can store multiple versions of data, with each version identified by a **timestamp**.

- ◆ **Role:** The value represents the actual data you are storing, which can include different versions of the same data (e.g., different timestamps for the same cell).

Example

Consider a table storing user data where the row key is the user ID:

- ◆ **Row Key:** `user123`
- ◆ **Column Family:** `profile`
 - ◆ **Column Qualifier:** `name`
 - ◆ **Cell:** Stores the user's name (e.g., "Alice")
 - ◆ **Timestamp:** Each update to the name creates a new version.
 - ◆ **Column Qualifier:** `email`
 - ◆ **Cell:** Stores the user's email (e.g., "alice@example.com🔗")
 - ◆ **Timestamp:** Tracks changes to the email over time.

Key-Value in Bigtable vs. Traditional Key-Value Stores

- ◆ **Traditional Key-Value Stores:** These typically have a single key pointing to a single value. For example, the key "user123" might directly map to a JSON object with all user data.
- ◆ **Bigtable's Key-Value Approach:** Bigtable enhances the basic key-value concept by allowing each key (row key) to map to multiple column families, each with its own set of columns (column qualifiers). Each of these can store multiple versions of data, making it much more flexible and suitable for complex datasets.

Conclusion

In summary, the "key-value" aspect of Cloud Bigtable emphasizes its ability to efficiently store and retrieve data based on keys (row keys). However, unlike traditional key-value stores, Bigtable provides a more sophisticated and flexible data model, allowing for wide-column storage with multiple versions of data per key. This makes Bigtable particularly powerful for large-scale, time-series data, user profiles, and other use cases requiring high read/write throughput and low-latency access.