



[Return to Table of Contents](#)

Cloud Bigtable Overview

[Previous](#)

Choose a Lesson

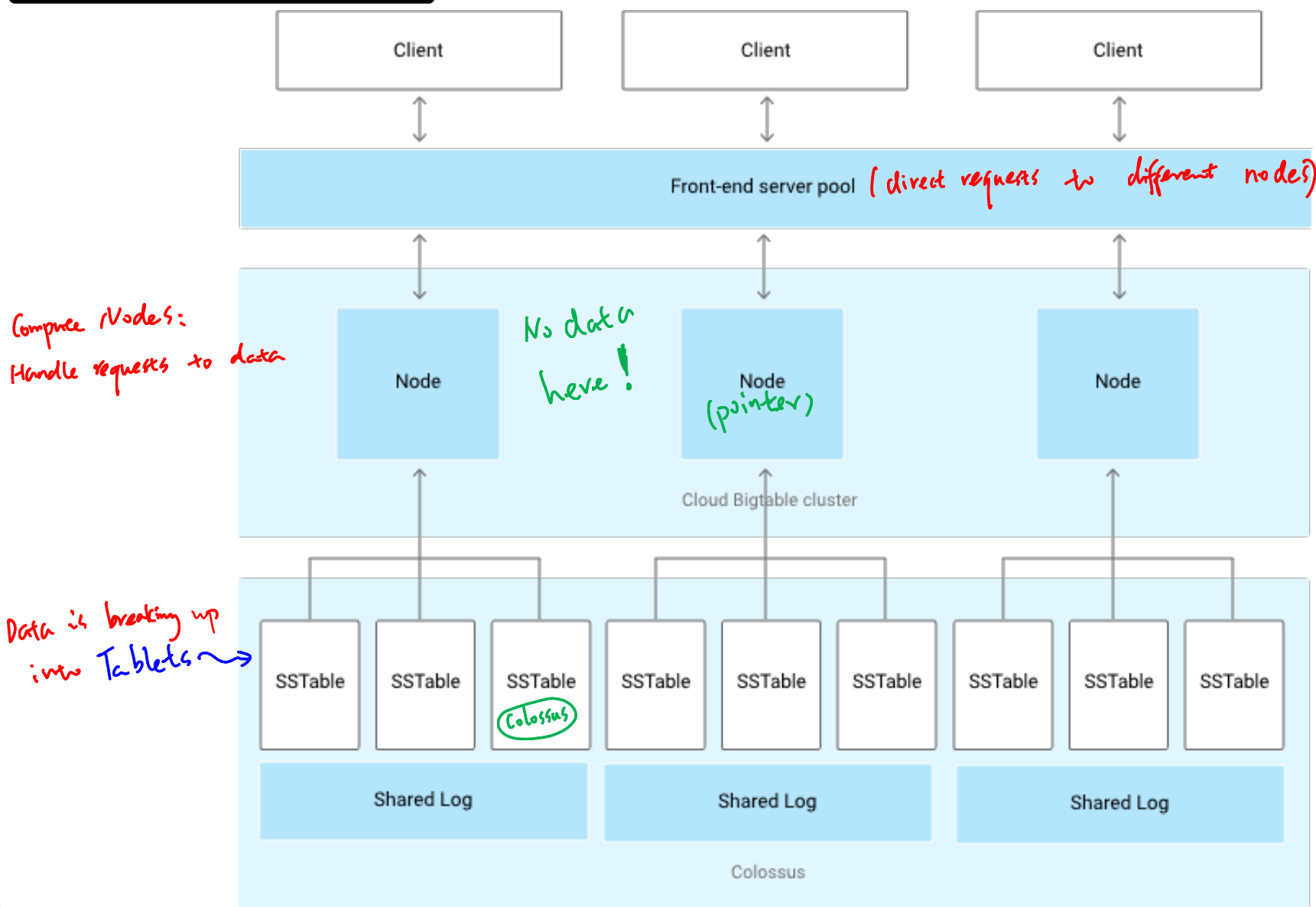
Cloud Bigtable Overview

Instance Configuration

Data Organization

Schema Design

Cloud Bigtable Infrastructure





[Return to Table of Contents](#)

Choose a Lesson

Cloud Bigtable Overview

Instance Configuration

Data Organization

Schema Design



Cloud Bigtable

expensive, not for small business

Cloud Bigtable Overview

[Next](#)

What is Cloud Bigtable?

- High performance, massively scalable **NoSQL** database
- Ideal for **large** analytical workloads

History of Bigtable

- Considered one of the originators for a NoSQL industry
- Developed by Google in 2004
 - Existing database solutions were too slow
 - Needed real-time access to **petabytes of data**
- Powers Gmail, YouTube, Google Maps, and others

What is it used for?

- High throughput analytics
- Huge datasets

> 1TB datasets

Use Cases

- Financial data – stock prices
- IoT data
- Marketing data – purchase histories

Access Control *(IAM access)*

- Project wide or instance level
- Read/Write/Manage

[Return to Table of Contents](#)

Choose a Lesson

[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)

Cloud Bigtable

Instance Configuration

[Next](#)

Instance basics

- Not no-ops
 - Must configure nodes
- Entire **Bigtable project** called **'instance'**
 - All nodes and clusters
- Nodes grouped into clusters
 - 1 or more clusters per instance
- Auto-scaling storage
- Instance types
 - **Development** - low cost, **single node** in a single cluster
 - No replication (for testing)
 - **Production** - 3+ nodes per cluster, also can create other clusters too
 - Replication available, throughput guarantee

Replication and Changes

- Synchronize data between clusters
 - One additional cluster, total
 - (Beta) available cross-region (may change later)
- Resizing
 - Add and remove nodes and clusters with no downtime
- Changing disk type (e.g. **HDD to SSD**) requires new instance

Interacting with Bigtable

- Command line - ^①**cbt tool** or ^②**HBase shell**
 - **cbt tool** is simpler and preferred option

1. export data to cloud storage
 2. create new instance
 3. re-import data again

① How many nodes in your cluster?

② What type of disk you use?

[Return to Table of Contents](#)

Choose a Lesson

[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)

Cloud Bigtable

*cbt:
we're interacting
with bigtable*

Instance Configuration

[Previous](#)

Bigtable interaction using **cbt**

- Install the cbt command in Google SDK
 - `sudo gcloud components update`
 - `gcloud components install cbt`
- Configure cbt to use your project and instance via `.cbtrc` file'
 - `echo -e "project = [PROJECT_ID]\ninstance = [INSTANCE_ID]" > ~/.cbtrc`
- Create table
 - `cbt createtable my-table`
- List table
 - `cbt ls`
- Add column family
 - `cbt createfamily my-table cf1` (family name is cf1)
- List column family
 - `cbt ls my-table`
- Add value to row 1, using column family cf1 and column qualifier c1
 - `cbt set my-table r1 cf1:c1=test-value`
- Delete table (if not deleting instance)
 - `cbt deletetable my-table`
- Read the contents of your table
 - `cbt read my-table`

Get help with cbt command using 'cbt --help'

Memorize the page ↗



[Return to Table of Contents](#)

Choose a Lesson

Cloud Bigtable Overview

Instance Configuration

Data Organization

Schema Design

Data Organization

How data is organized in big table

Data Organization

- **One** big table (hence the name Bigtable)
- Table can be thousands of columns/billions of rows
- **Table is sharded across tablets**

Table components

- Row Key
 - First column
- Columns grouped into column families

	Column-Family-1		Column-Family-2	
Row Key	Column-Qualifier-1	Column-Qualifier-2	Column-Qualifier-1	Column-Qualifier-2
r1	r1, cf1:cq1	r1, cf1:cq2	r1, cf1:cq1	r1, cf1:cq2
r2	r2, cf1:cq1	r2, cf1:cq2	r2, cf1:cq1	r2, cf1:cq2

Indexing and Queries

- **Only the row key is indexed**
- Schema design is necessary for efficient queries!
- Field promotion - move fields from column data to row key

Row key	Column data
BATTERY#Corrie#20150301124501001	METRIC:PERCENTAGE:98
BATTERY#Corrie#20150301124501003	METRIC:PERCENTAGE:96
BATTERY#Jo#20150301124501002	METRIC:PERCENTAGE:54
BATTERY#Sam#20150301124501004	METRIC:PERCENTAGE:43
BATTERY#Sam#20150301124501005	METRIC:PERCENTAGE:38

[Return to Table of Contents](#)

Choose a Lesson

Cloud Bigtable Overview

Instance Configuration

Data Organization

Schema Design

Row Key

memusage+user+timestamp

20-mattu-201805082048

good schema for row key

Schema Design (exam topic)

choose the best schema for the row key for max out of efficiency.

Schema Design

- 1 Per table – Row key is the **only indexed** item
- 2 Keep all entity info in a single row
- 3 Related entities should be in adjacent rows
 - More efficient reads
- 4 Tables are sparse – empty columns take no space

Schema Efficiency

- Well-defined row keys = less work
 - Multiple values in row key
- Row key (or prefix) should be sufficient for a search
- **Goal** = spread loads over multiple nodes
 - All on one node = **hotspotting**

Row Key Best Practices

- **Good** row keys = distributed load
 - 1 Reverse domain names (com.linuxacademy.support)
 - 2 String identifiers (mattu)
 - 3 Timestamps (reverse, NOT at front/or only identifier) Y-m-d
- **Poor** row keys = hotspotting
 - Domain names (support.linuxacademy.com)
 - Sequential ID's
 - Timestamps alone/at front

Table Design - Time Series Data

tall and narrow
short and wide

- For time series data, use tall and narrow tables (one event per row)
 - Easier to run queries against data