Google Cloud

High-Throughput BigQuery and Bigtable Streaming Features

Agenda

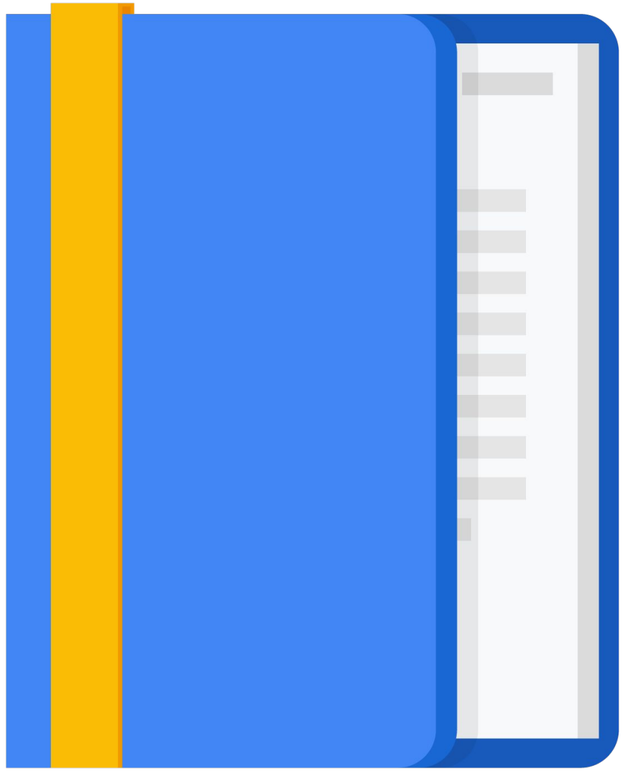
Processing Streaming Data

Cloud Pub/Sub

Cloud Dataflow Streaming Features

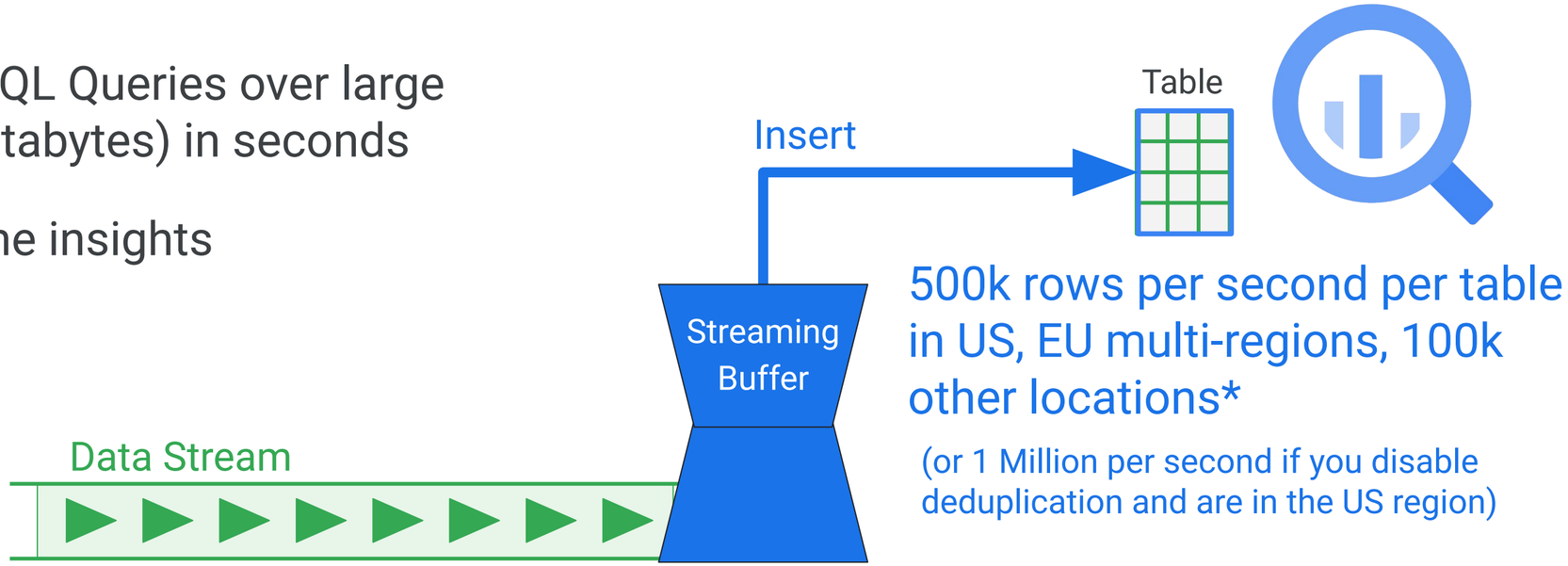
BigQuery and Bigtable Streaming Features

Advanced BigQuery Functionality



BigQuery allows you to stream records into a table; query results incorporate latest data

- Interactive SQL Queries over large datasets (petabytes) in seconds
- Near-real-time insights



500k rows per second per table in US, EU multi-regions, 100k other locations*

(or 1 Million per second if you disable deduplication and are in the US region)

Note: Unlike load jobs, there is a cost for streaming inserts (see quota and limits)

Insert streaming data into a BigQuery table

```
export GOOGLE_APPLICATION_CREDENTIALS="/home/user/Downloads/[FILE_NAME].json"
```

Credentials

The service must have Cloud IAM permissions set in the Web UI

```
pip install google-cloud-bigquery
```

Install API

Python

```
from google.cloud import bigquery
client = bigquery.Client(project='PROJECT_ID')

dataset_ref = bigquery_client.dataset('my_datasset_id')
table_ref = dataset_ref.table('my_table_id')
table = bigquery_client.get_table(table_ref)

# read data from Cloud Pub/Sub and place into row format
# static example customer orders in units:
rows_to_insert = [
    (u'customer 1', 5),
    (u'customer 2', 17),
    ]
errors = bigquery_client.insert_rows(table, rows_to_insert)
```

Create a client

Access dataset and table

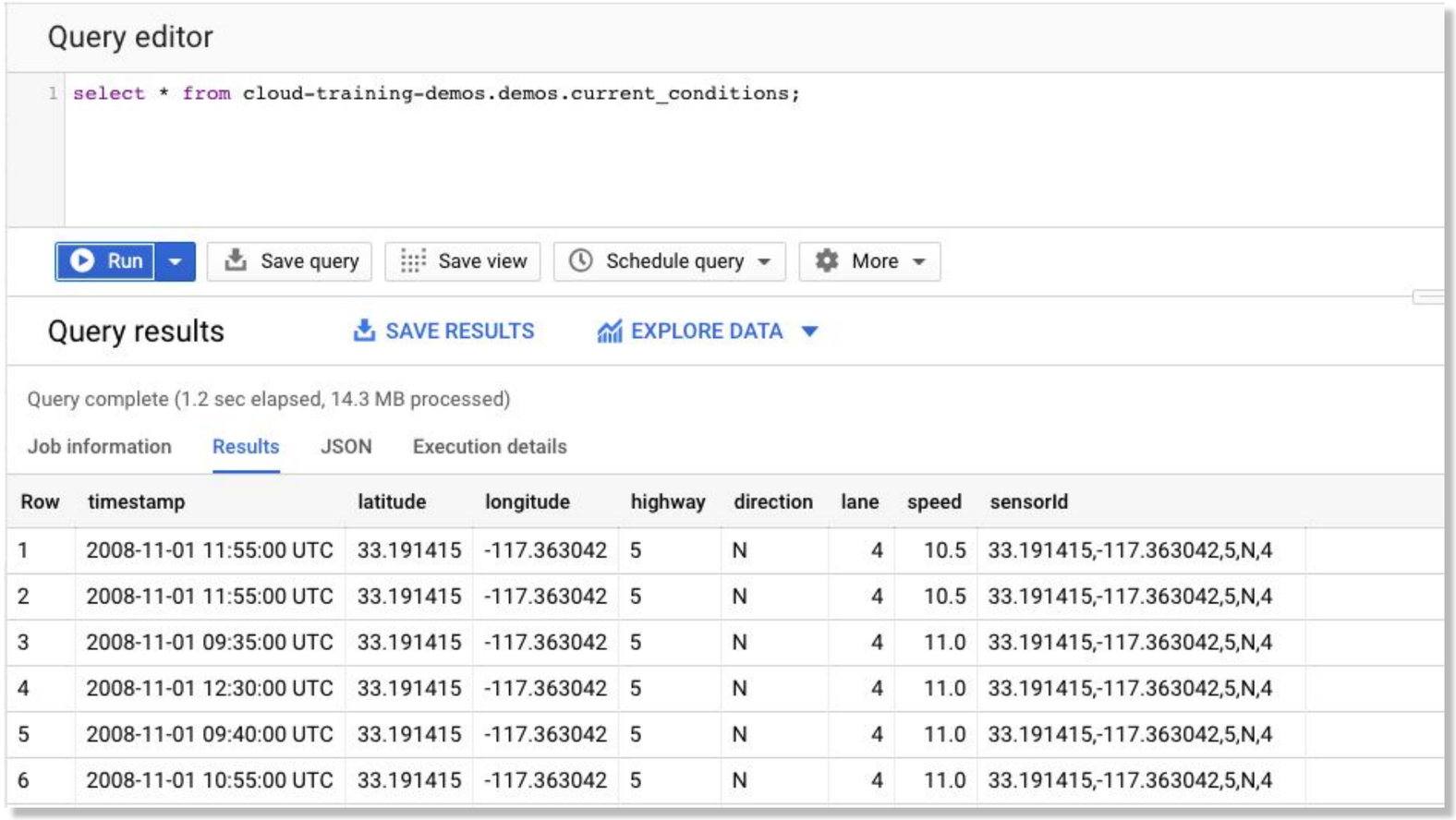
Get table access from API

Insert rows into table

Perform insert

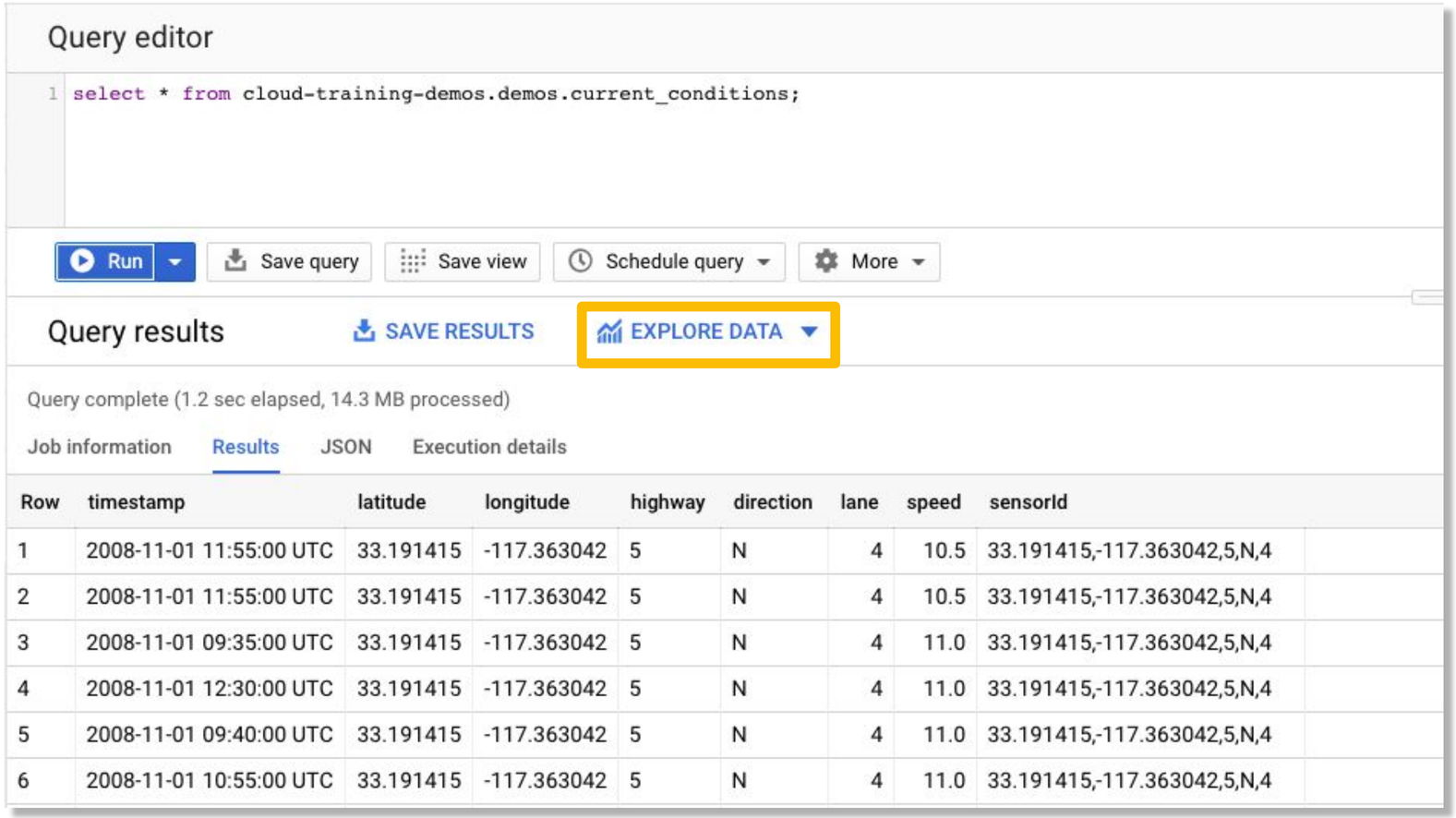
Google Cloud

Review streaming data in BigQuery



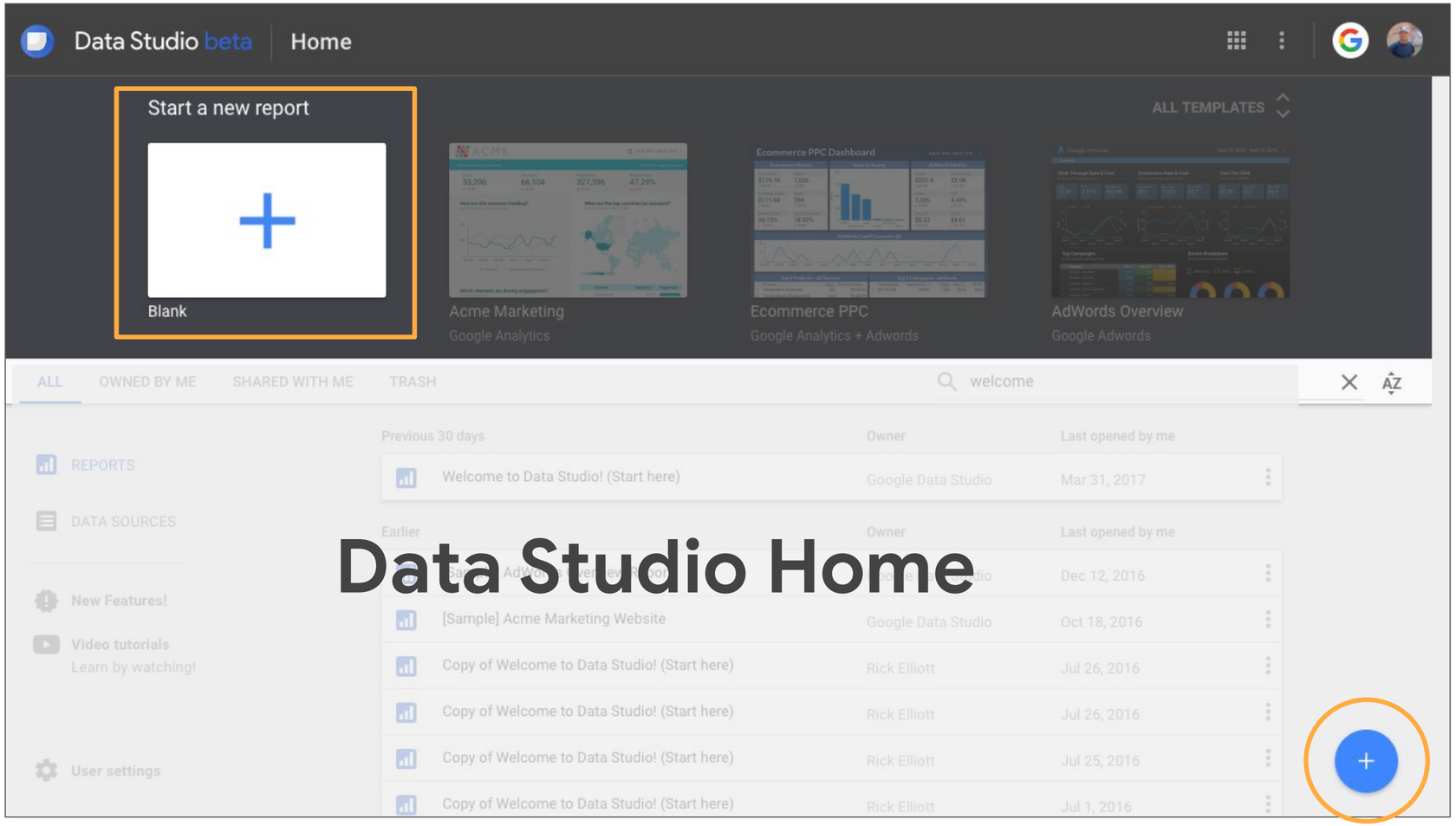
Row	timestamp	latitude	longitude	highway	direction	lane	speed	sensorId
1	2008-11-01 11:55:00 UTC	33.191415	-117.363042	5	N	4	10.5	33.191415,-117.363042,5,N,4
2	2008-11-01 11:55:00 UTC	33.191415	-117.363042	5	N	4	10.5	33.191415,-117.363042,5,N,4
3	2008-11-01 09:35:00 UTC	33.191415	-117.363042	5	N	4	11.0	33.191415,-117.363042,5,N,4
4	2008-11-01 12:30:00 UTC	33.191415	-117.363042	5	N	4	11.0	33.191415,-117.363042,5,N,4
5	2008-11-01 09:40:00 UTC	33.191415	-117.363042	5	N	4	11.0	33.191415,-117.363042,5,N,4
6	2008-11-01 10:55:00 UTC	33.191415	-117.363042	5	N	4	11.0	33.191415,-117.363042,5,N,4

Want to visualize insights? Explore Google Data Studio insights right from within BigQuery



Row	timestamp	latitude	longitude	highway	direction	lane	speed	sensorId
1	2008-11-01 11:55:00 UTC	33.191415	-117.363042	5	N	4	10.5	33.191415,-117.363042,5,N,4
2	2008-11-01 11:55:00 UTC	33.191415	-117.363042	5	N	4	10.5	33.191415,-117.363042,5,N,4
3	2008-11-01 09:35:00 UTC	33.191415	-117.363042	5	N	4	11.0	33.191415,-117.363042,5,N,4
4	2008-11-01 12:30:00 UTC	33.191415	-117.363042	5	N	4	11.0	33.191415,-117.363042,5,N,4
5	2008-11-01 09:40:00 UTC	33.191415	-117.363042	5	N	4	11.0	33.191415,-117.363042,5,N,4
6	2008-11-01 10:55:00 UTC	33.191415	-117.363042	5	N	4	11.0	33.191415,-117.363042,5,N,4

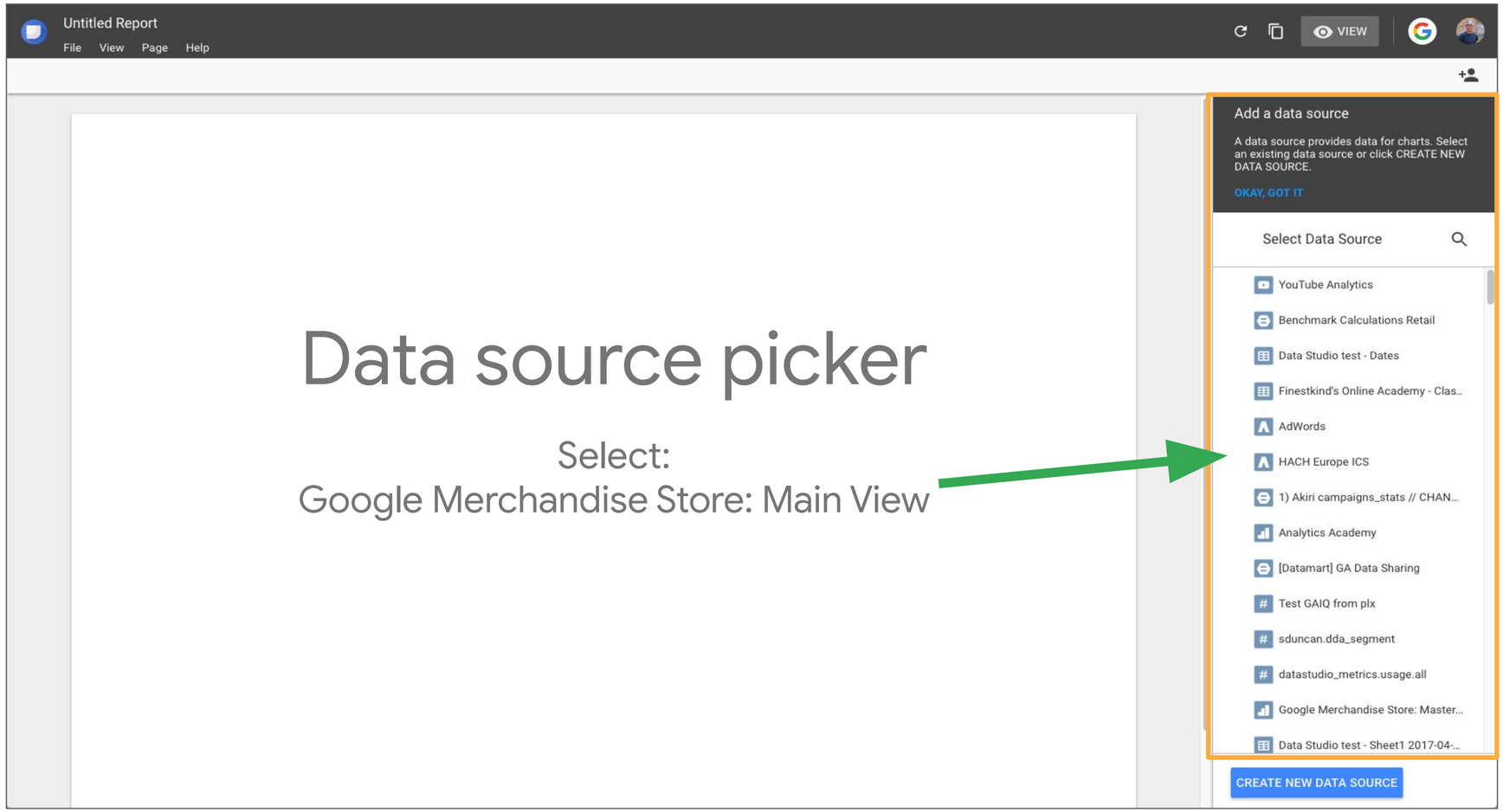
Create new reports in the Data Studio UI



Data Studio Home

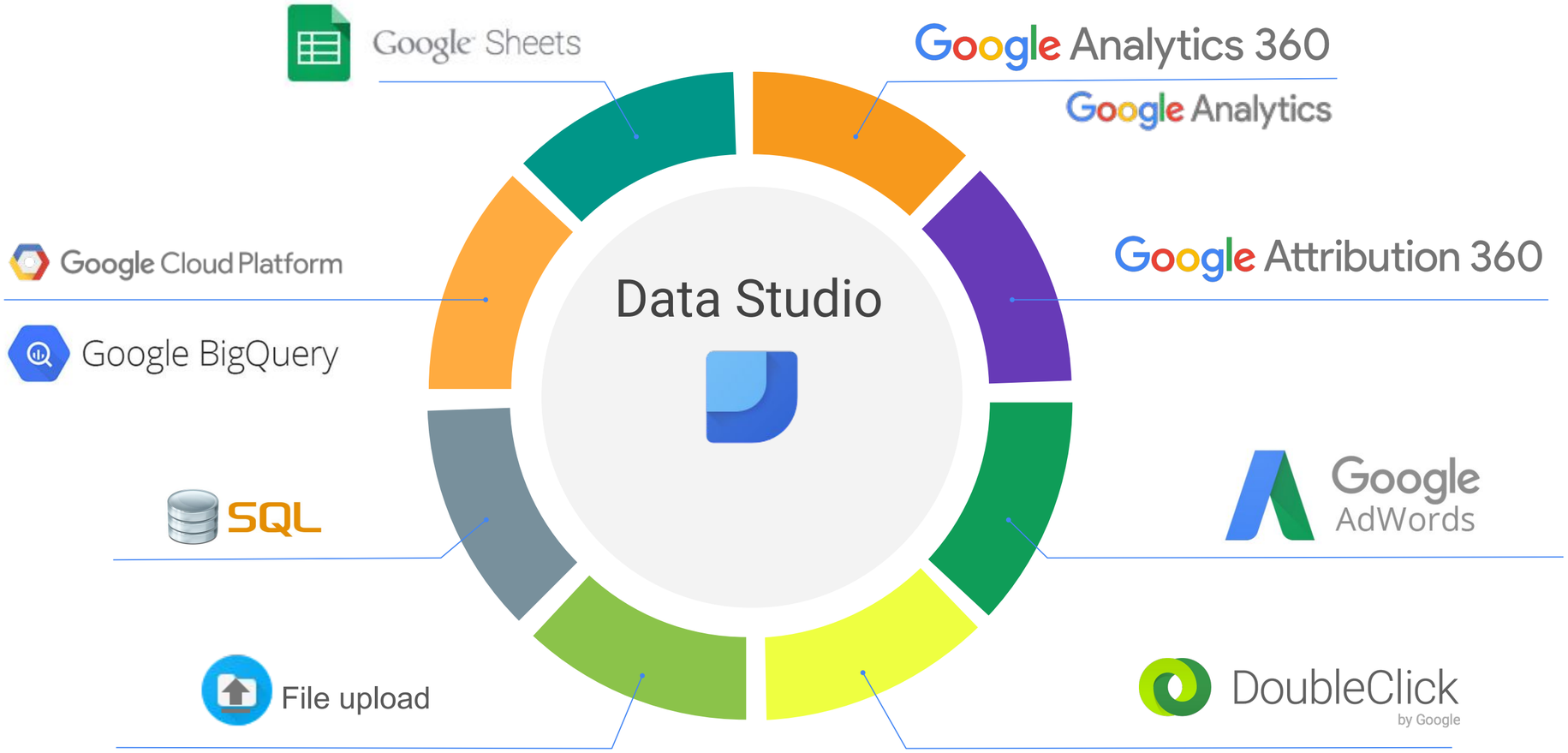
Google Cloud

Select data sources to build your visualizations



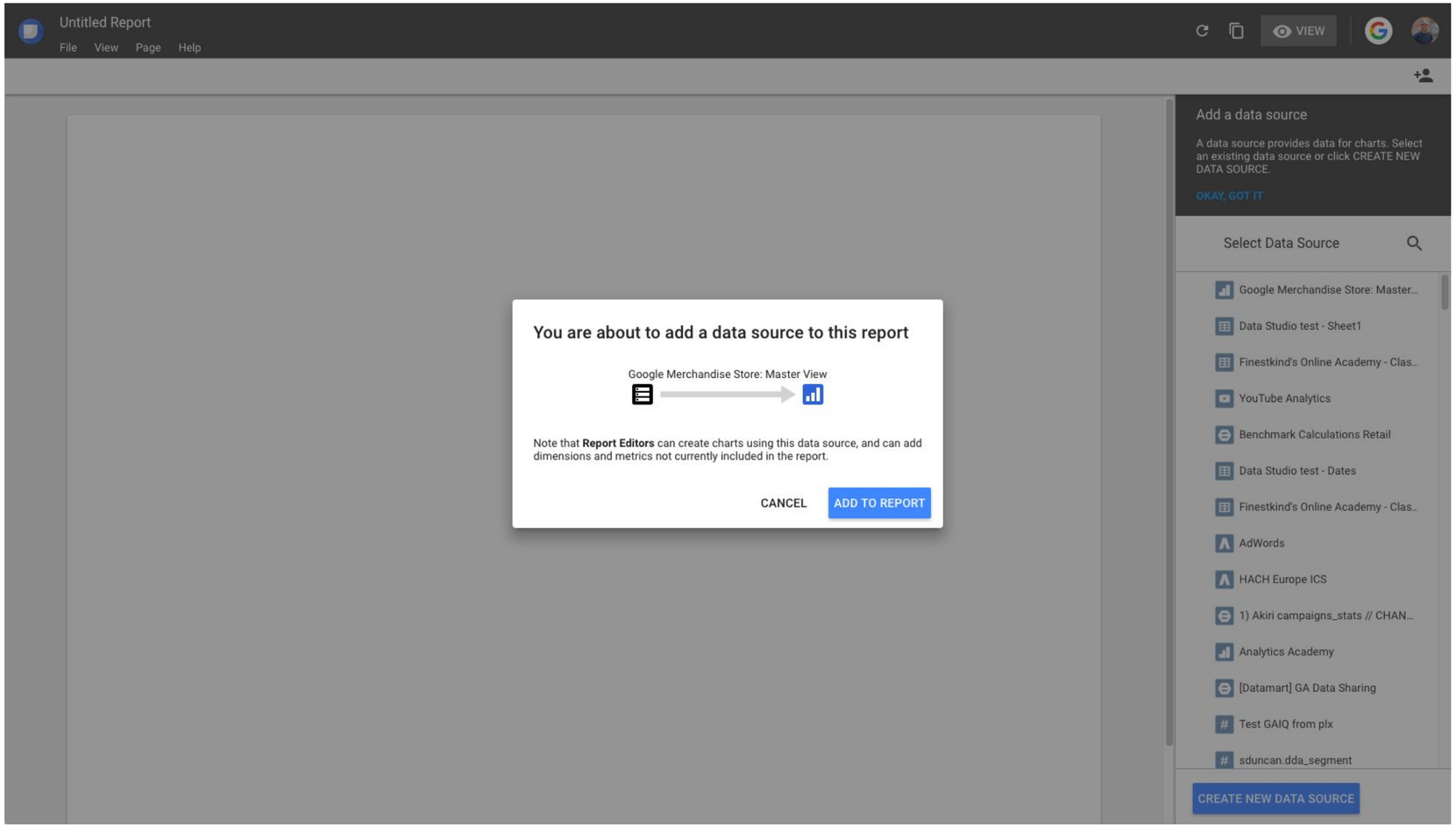
Google Cloud

Connect to multiple different types of data sources



Google Cloud

Add the data source to your report



Google Cloud

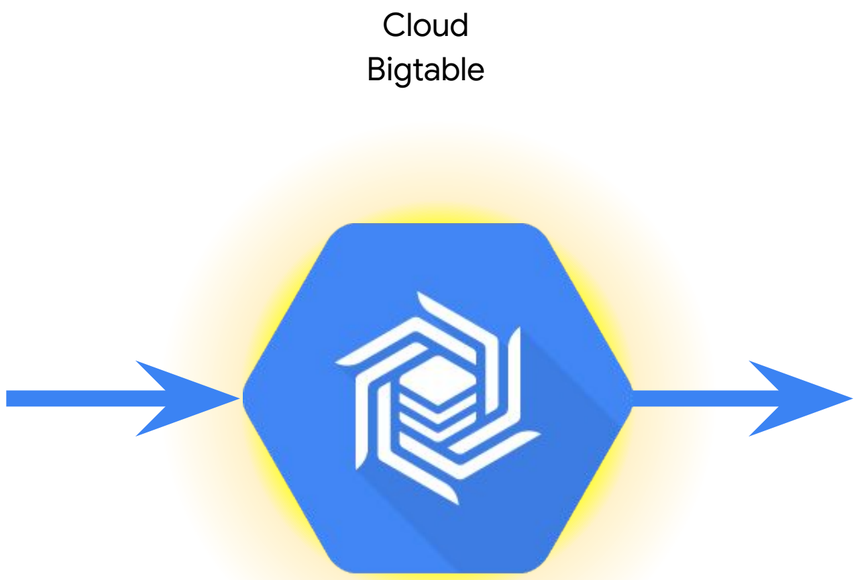
Lab

Streaming Analytics and Dashboards

Objectives

- Connect to a BigQuery data source from Google Data Studio
- Create reports and charts to visualize BigQuery data

Cloud Bigtable

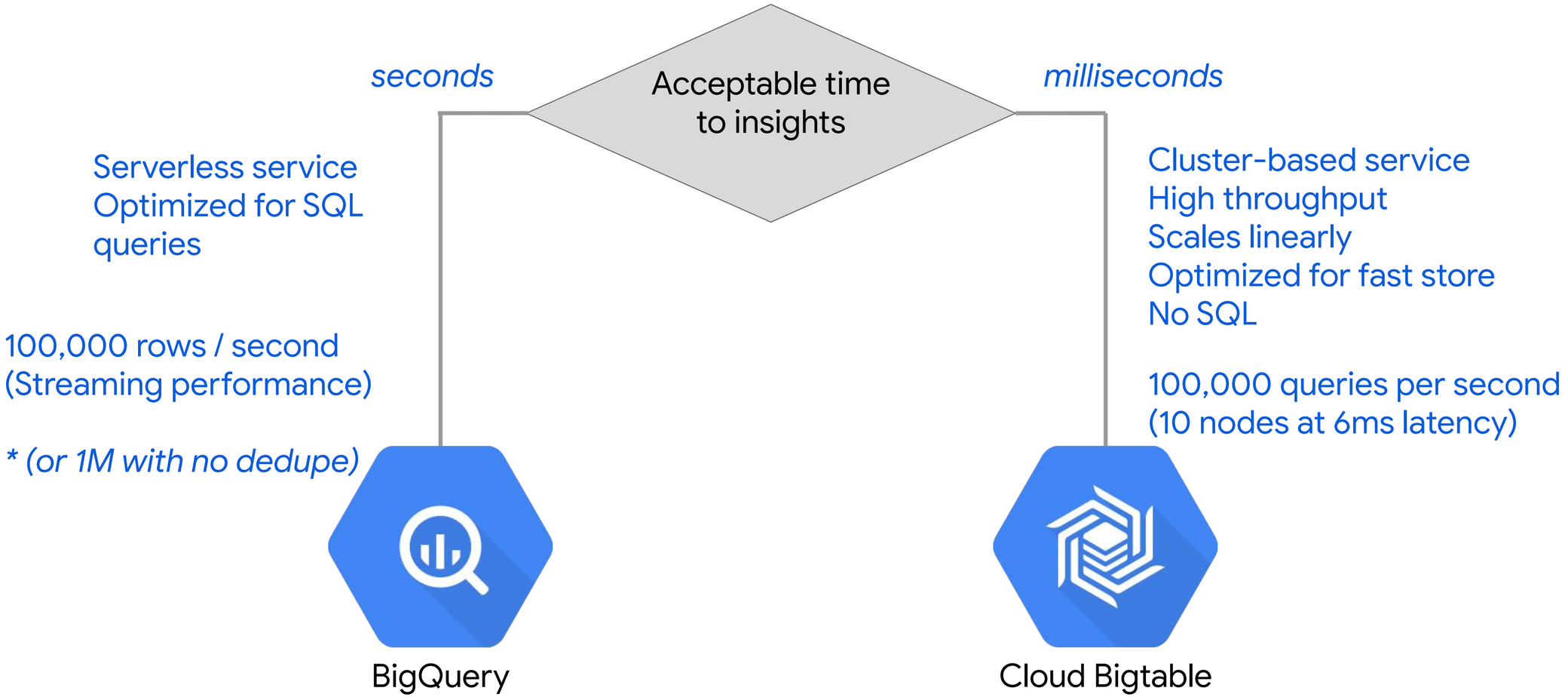


Qualities that Cloud Bigtable contributes to Data Engineering solutions:

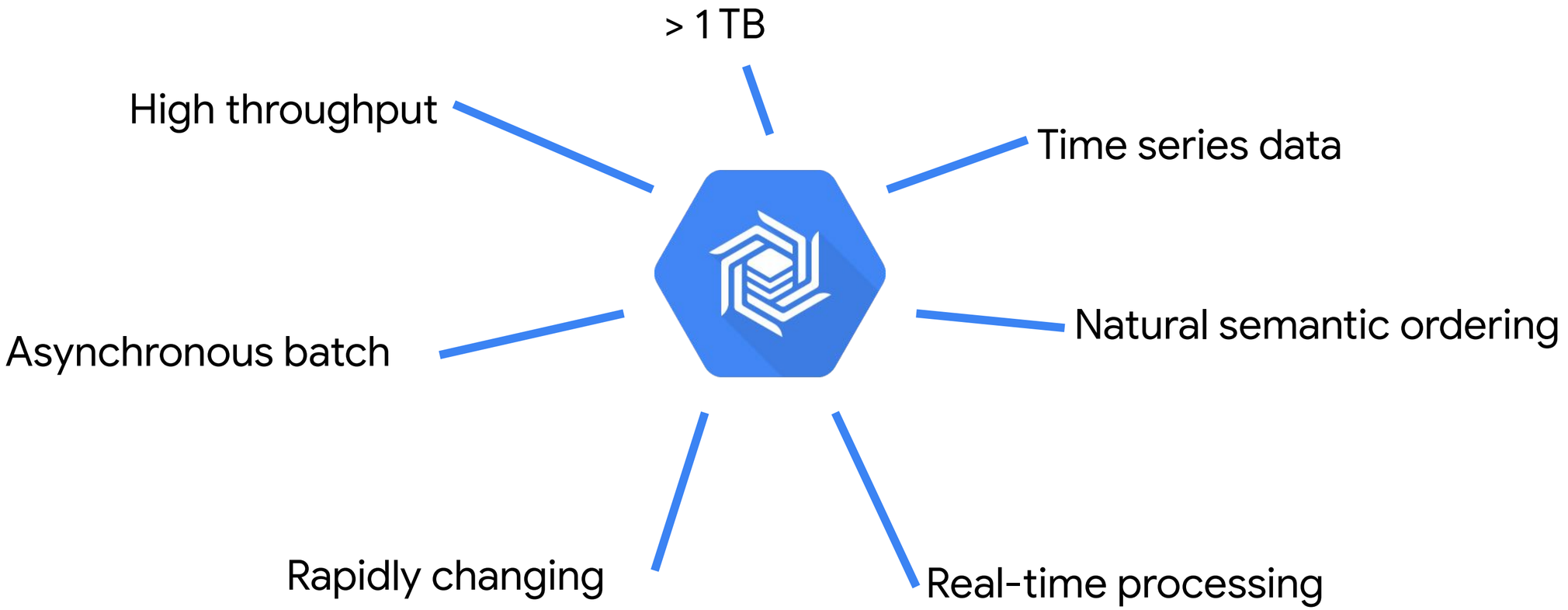
NoSQL Queries over large datasets (petabytes) in milliseconds

Very fast for specific cases

How to choose between Cloud Bigtable and BigQuery



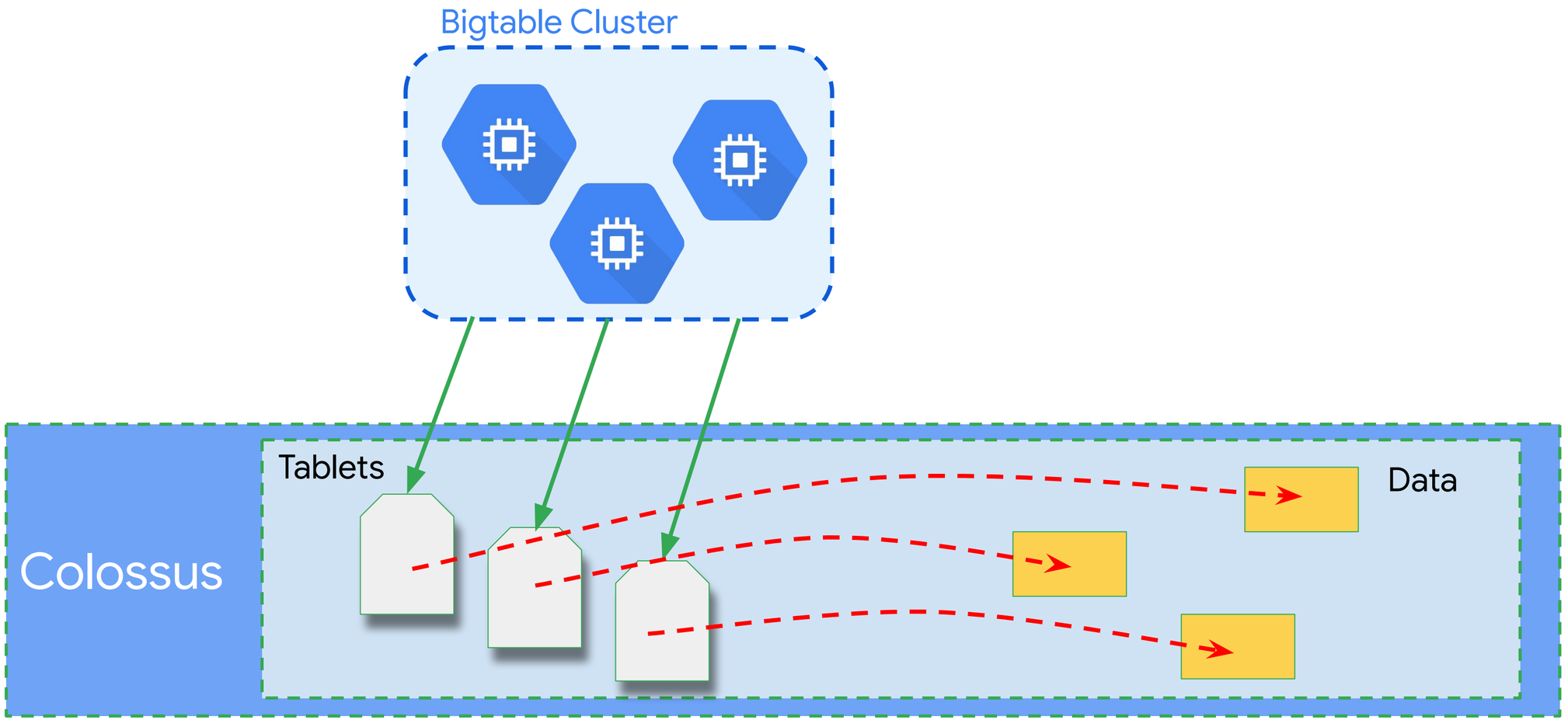
Consider Cloud Bigtable for these requirements



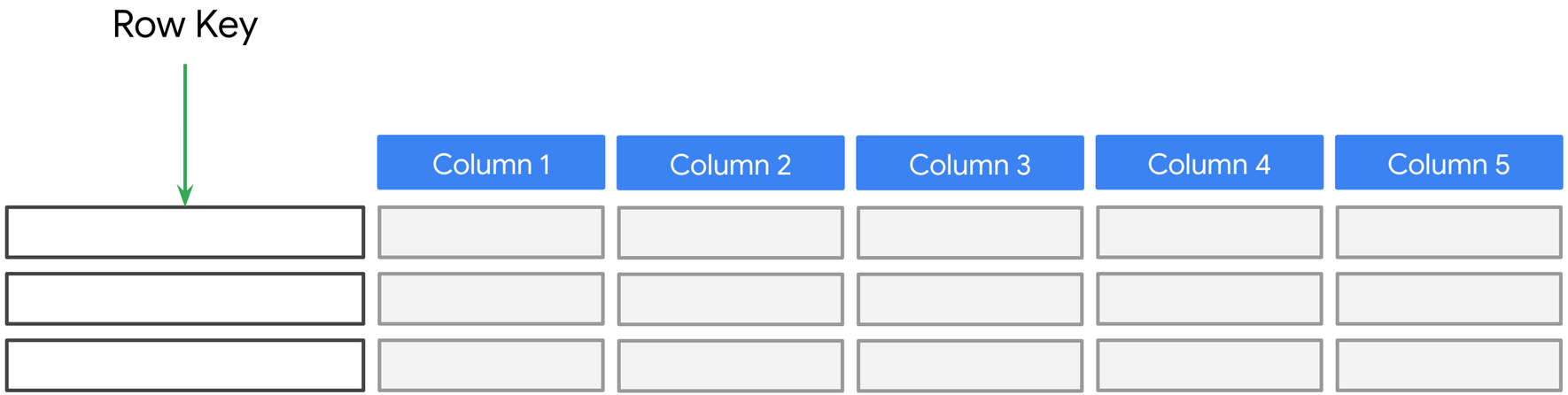
The most common use of Cloud Bigtable is...

Productionize a real-time lookup as part of an application, where speed and efficiency are desired beyond that of other databases.

How does Cloud Bigtable work?



Cloud Bigtable design idea is "simplify for speed"



The Row Key is the index.

And you get only one.

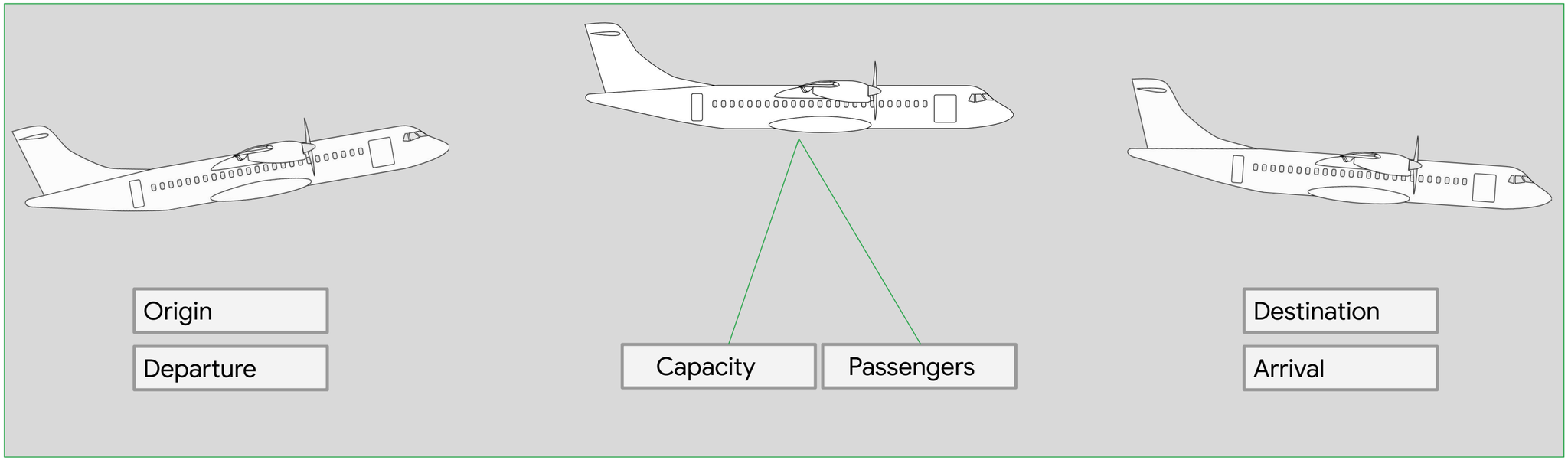
But speed depends on your data and Row Key

Flights of the world: Reviewing the data

Make

Model

Age



Origin

Departure

Capacity

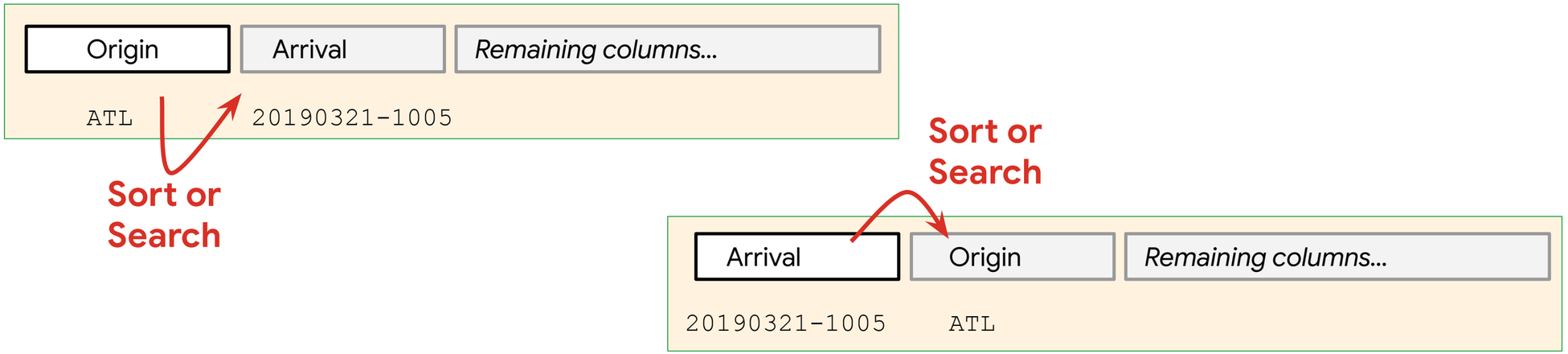
Passengers

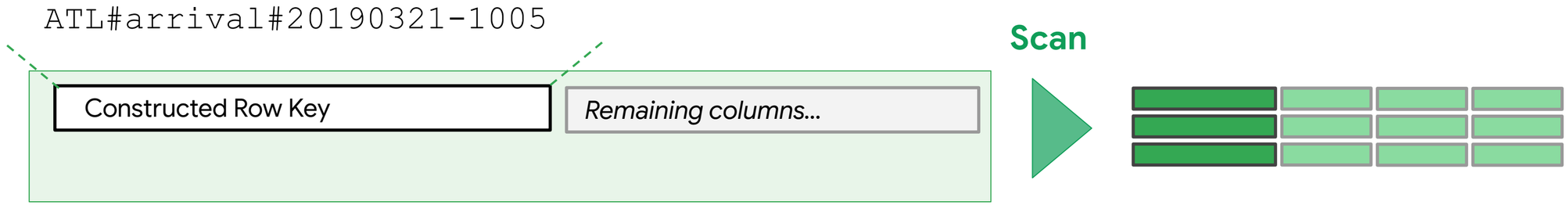
Destination

Arrival

What is the best Row Key?

Query: All flights originating in Atlanta and arriving between March 21st and 29th





Cloud Bigtable schema organization

Column Families

Row Key	Flight_Information					Aircraft_Information			
	Origin	Destination	Departure	Arrival	Passengers	Capacity	Make	Model	Age
ATL#arrival#20190321-1121	ATL	LON	20190321-0311	20190321-1121	158	162	B	737	18
ATL#arrival#20190321-1201	ATL	MEX	20190321-0821	20190321-1201	187	189	B	737	8
ATL#arrival#20190321-1716	ATL	YVR	20190321-1014	20190321-1716	201	259	B	757	23

Queries that use the row key, a row prefix, or a row range are the most efficient

Query: Current arrival delay for flights from Atlanta

1

ROW KEY BASED ON ATLANTA ARRIVALS
E.G. ORIGIN#arrival
(ATL#arrival#20190321-1005)

Puts latest flights at bottom of table

2

REVERSE TIMESTAMP TO THE ROWKEY
E.G. ORIGIN#arrival#RTS
(ATL#arrival#12345678)

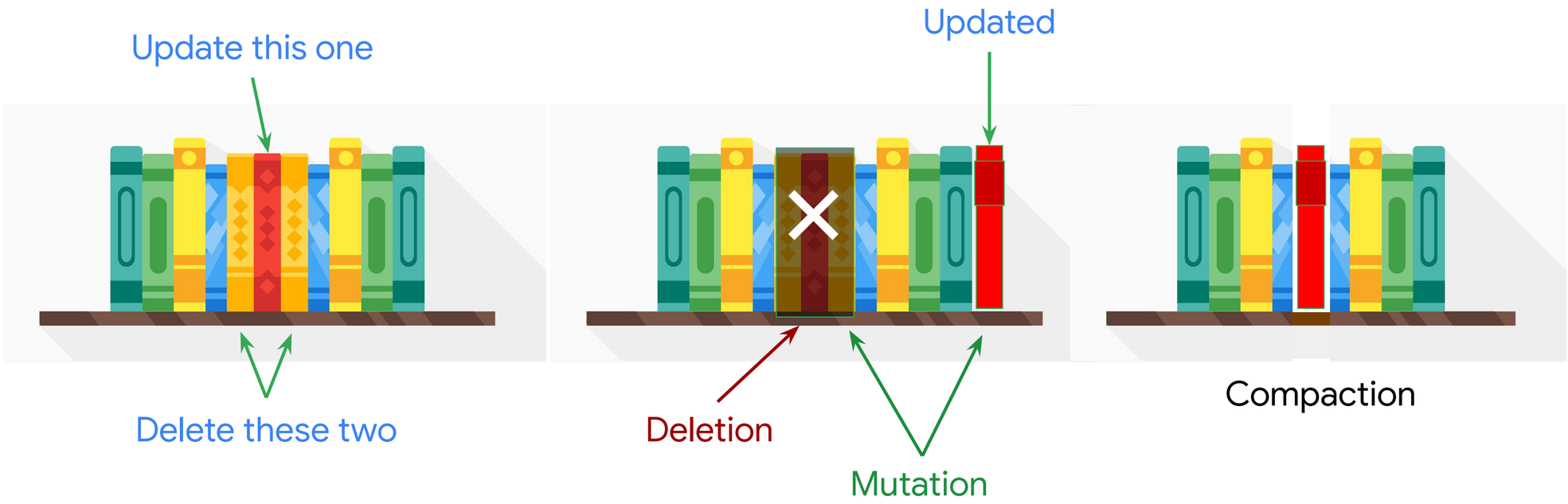
Puts latest flights at top of table

Use reverse timestamps when your most common query is for the latest values.

Query: Current arrival delay for flights from Atlanta

```
// key is ORIGIN#arrival#REVTS
String key = info.getORIGIN() //
   + "#arrival"  //
   + "#" + (Long.MAX_VALUE - ts.getMillis()); // reverse timestamp
```

What happens when data in Cloud Bigtable is changed?



Optimizing data organization for performance

Group related data for more efficient reads

Example row key:

```
DehliIndia#2019031411841
```

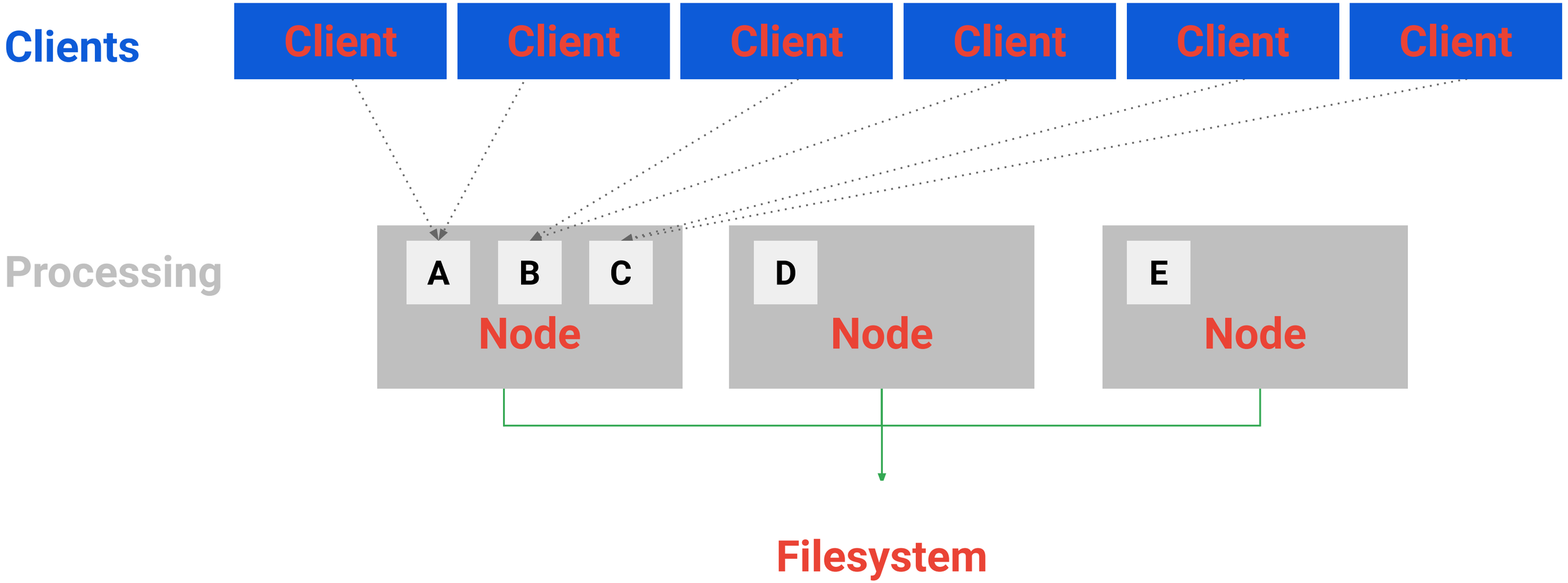
Use column families

Distribute data evenly for more efficient writes

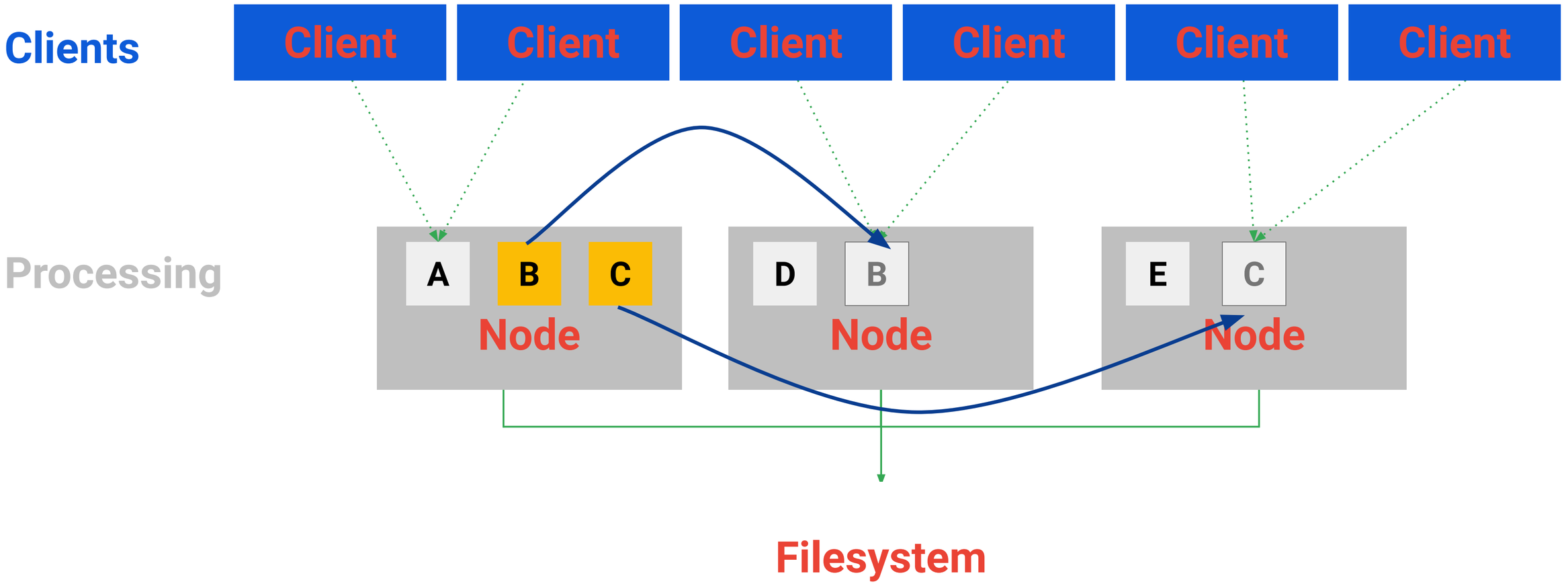
Place identical values in the same row or adjoining rows for more efficient compression

Use row keys to organize identical data

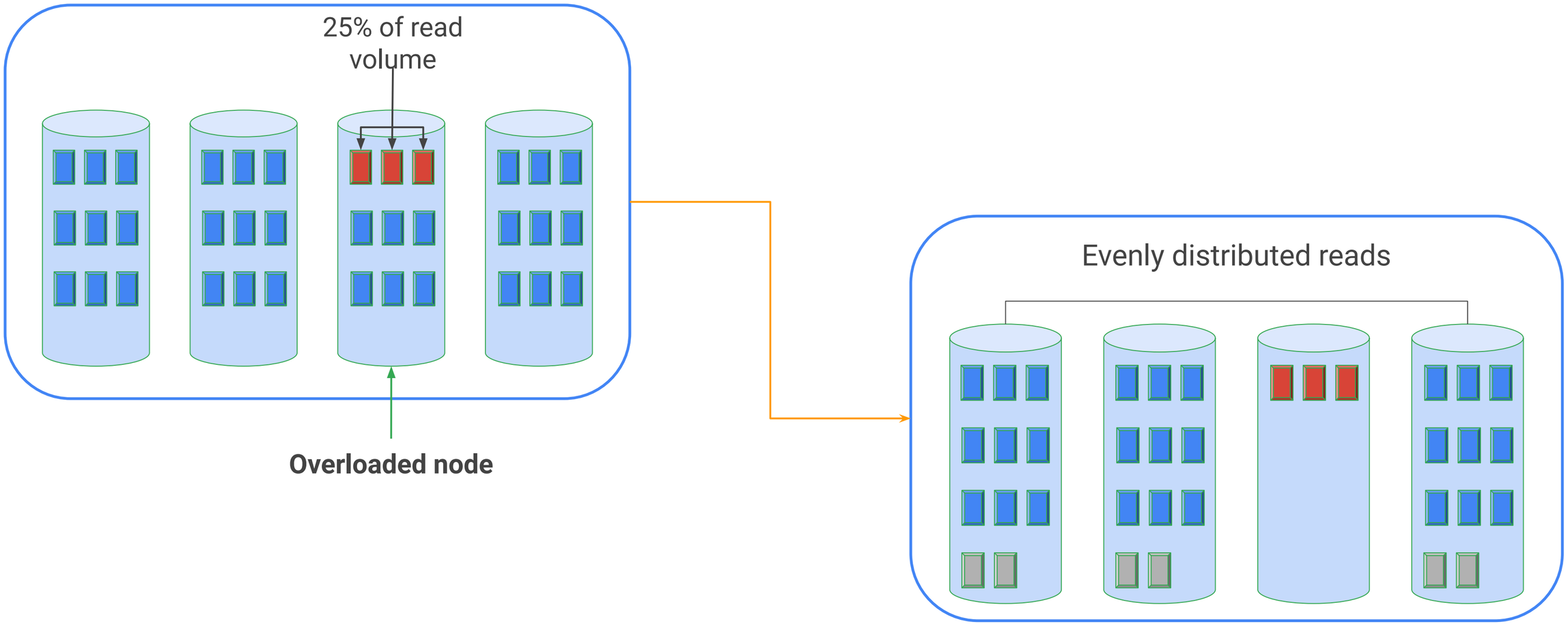
Cloud Bigtable self-improves by learning access patterns...



…and rebalances data accordingly



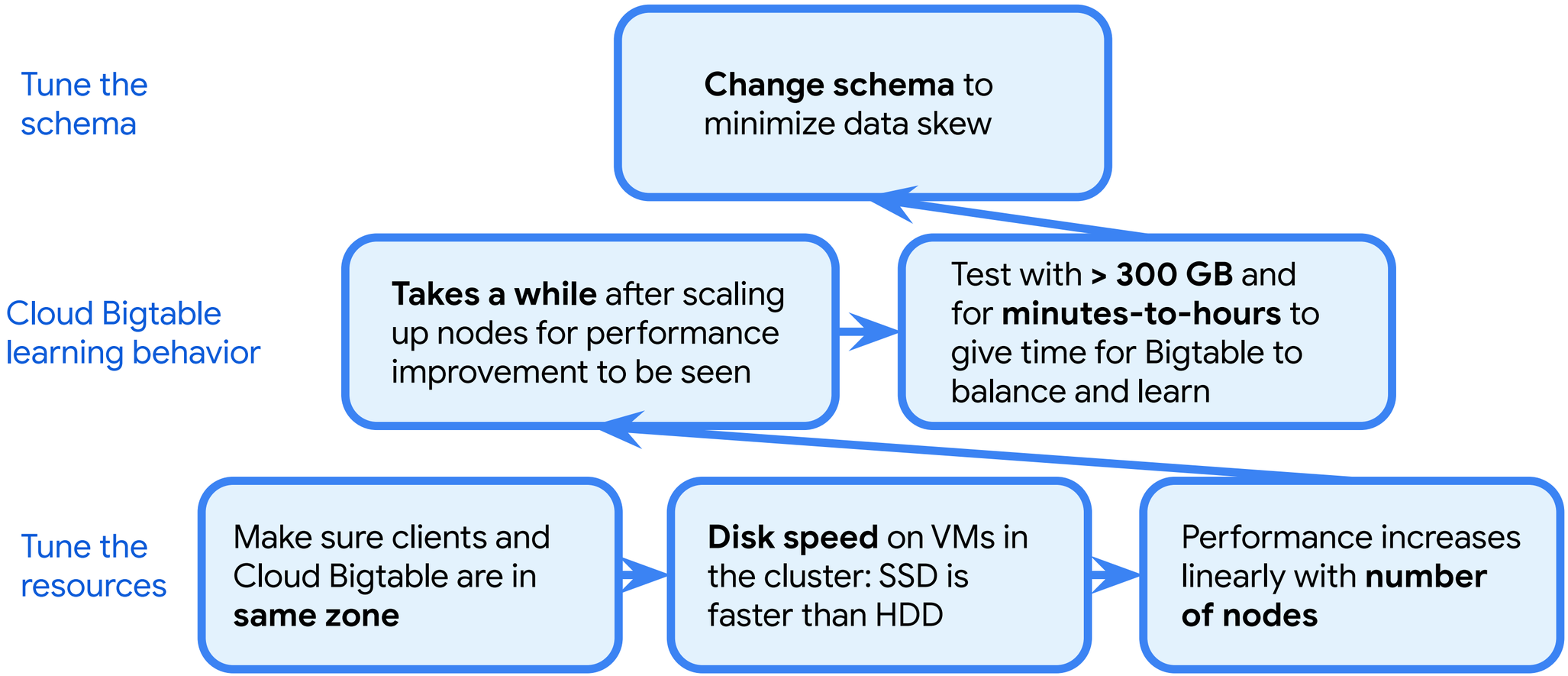
Rebalance strategy: distribute reads



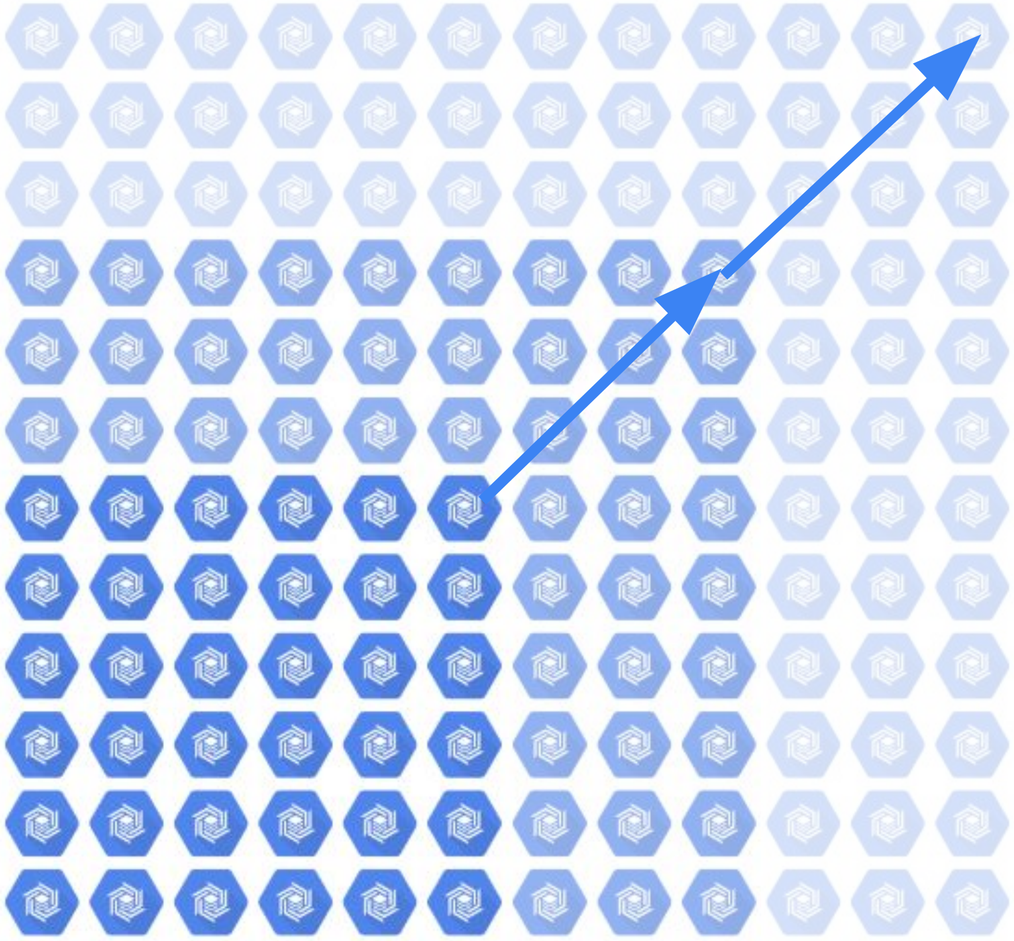
Google Cloud

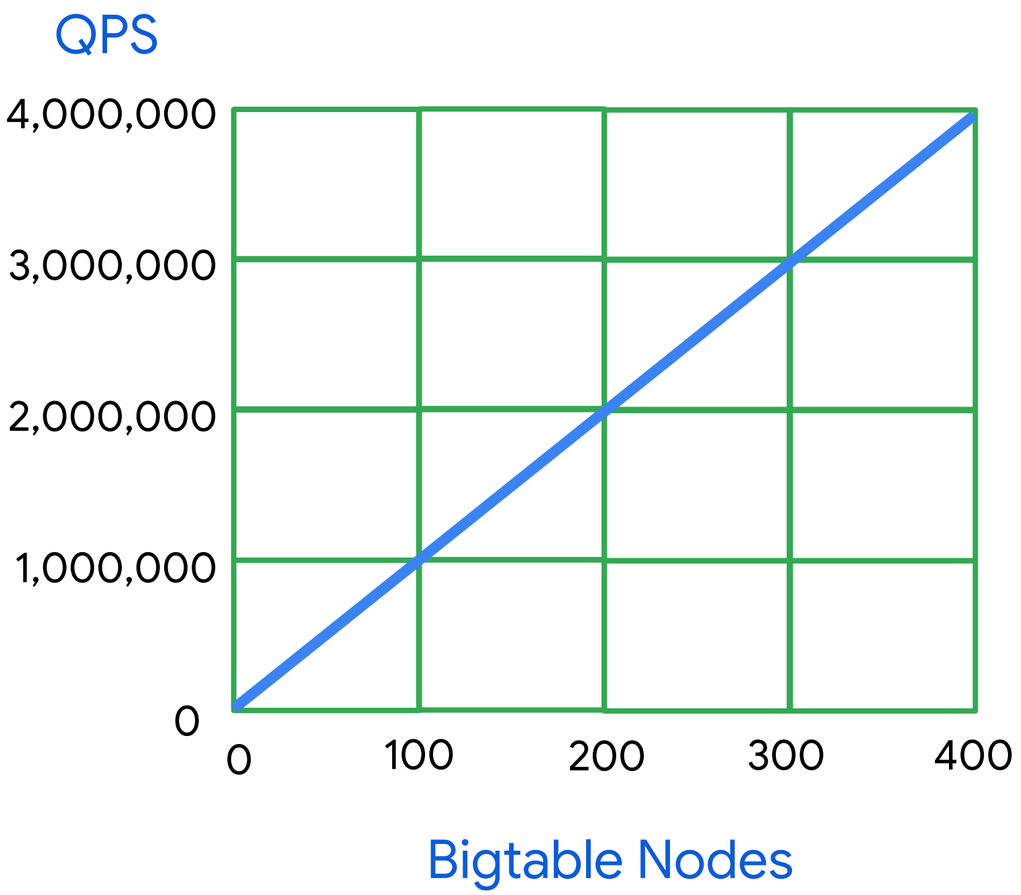
Optimizing Cloud Bigtable Performance

Optimizing Cloud Bigtable Performance



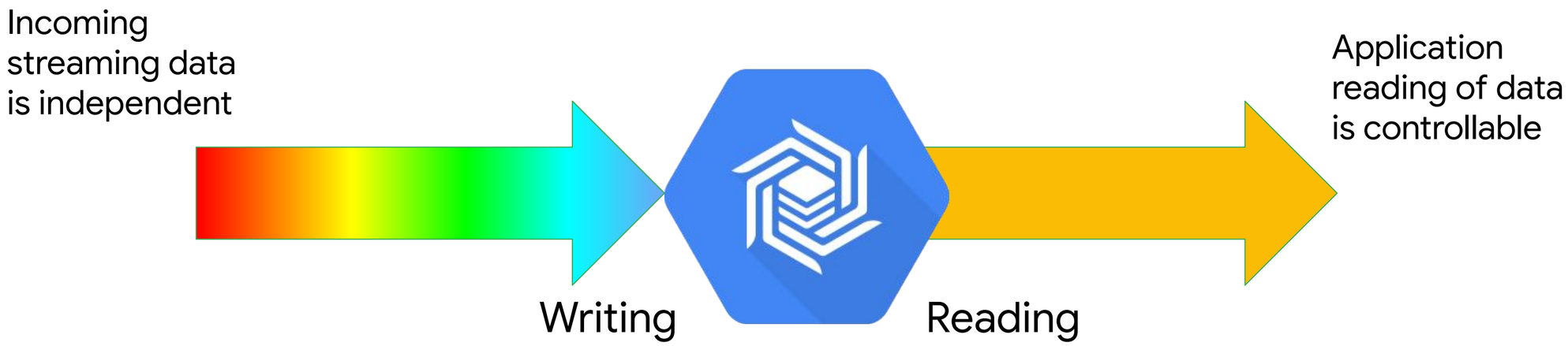
Throughput can be controlled by node count



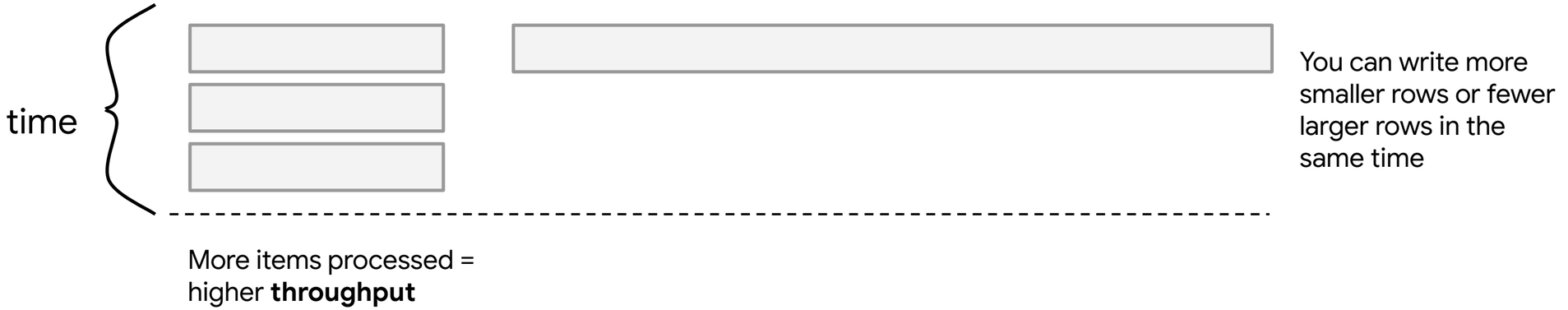


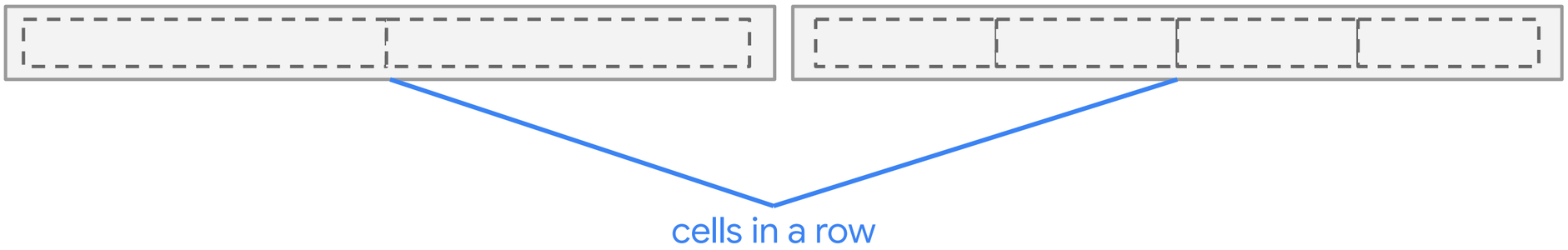
Google Cloud

Features for Cloud Bigtable streaming



Schema design is the primary control for streaming



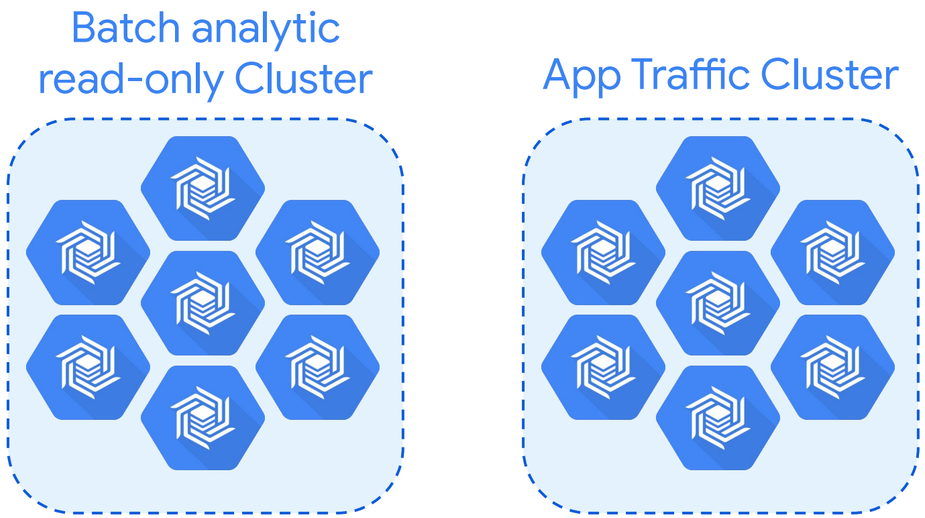


Use Cloud Bigtable replications to improve availability

Why perform replication?

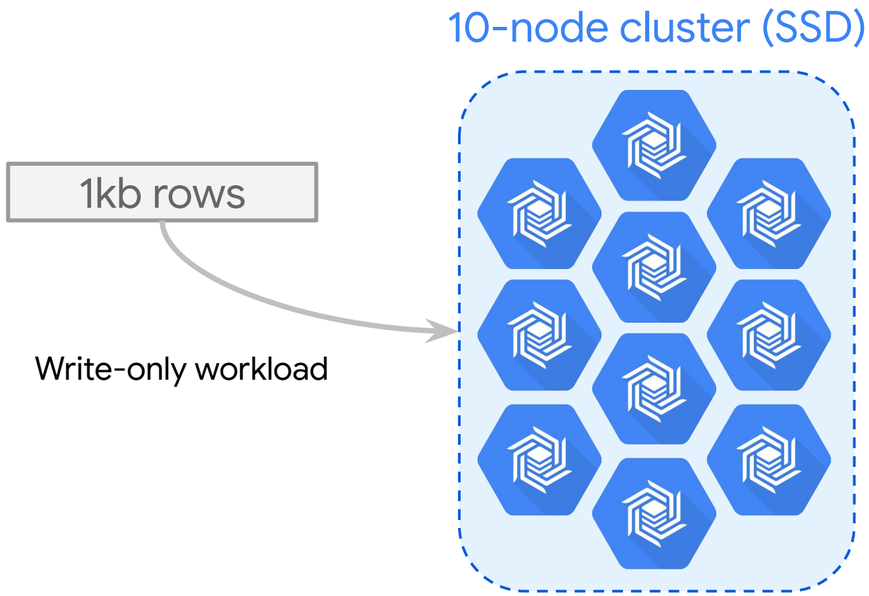
- Isolate serving applications from batch reads
- Improve availability
- Provide near-real-time backup
- Ensure your data has a global presence

```
gcloud bigtable clusters create CLUSTER_ID \
    --instance=INSTANCE_ID \
    --zone=ZONE \
    [--num-nodes=NUM_NODES] \
    [--storage-type=STORAGE_TYPE]
```



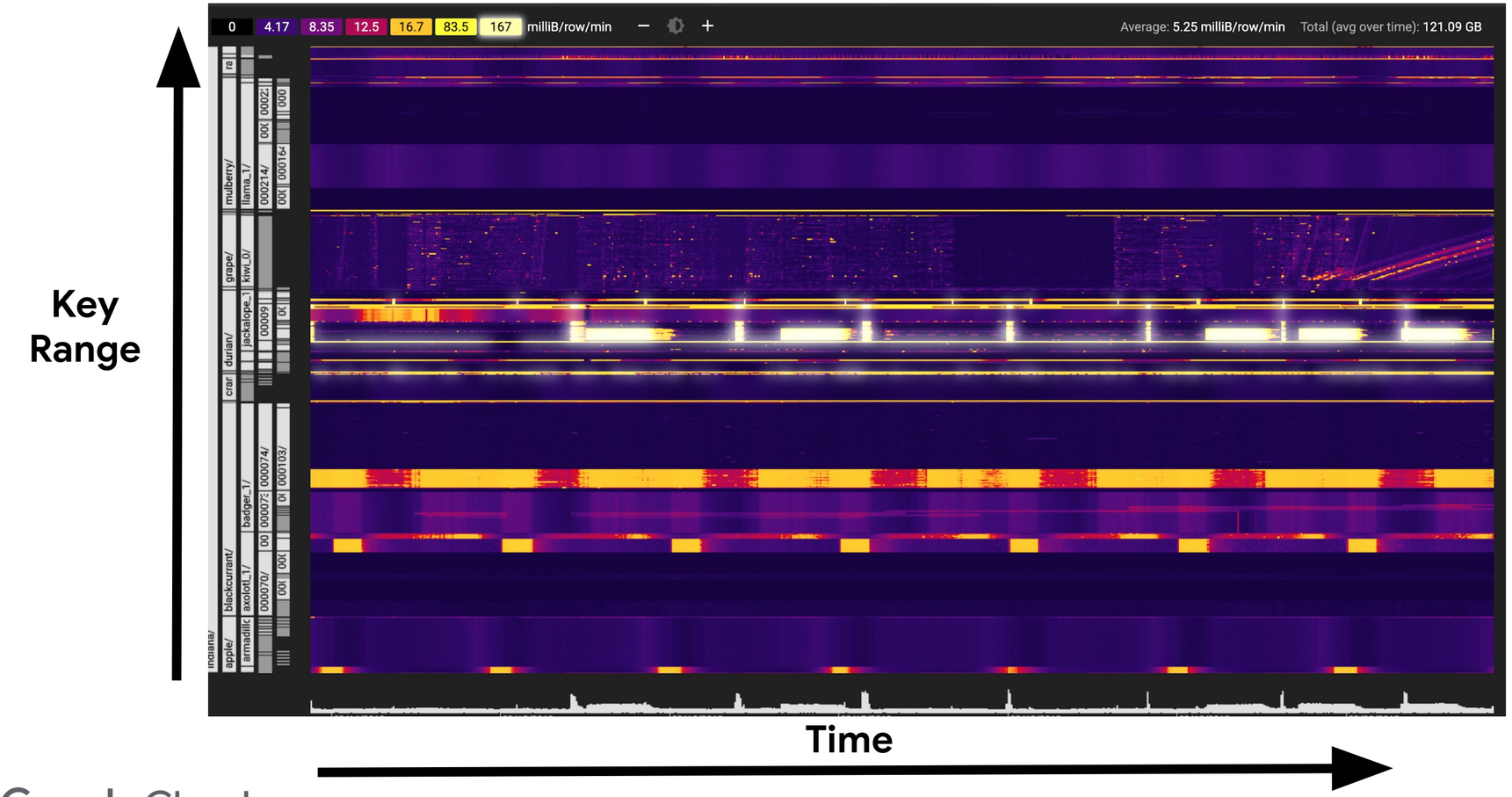
Run performance tests carefully for Cloud Bigtable streaming

Ideal



Real

Key Visualizer exposes read/write access patterns over time and key space



- find/prevent hotspots
- find rows with too much data
- see if your key schema is balanced

Google Cloud

Lab

Streaming Data Pipelines into Bigtable

Objectives

- Launch a Dataflow pipeline to read from PubSub and write into Bigtable
- Open an HBase shell to query the Bigtable database