INTRODUCTION TO GOOGLE BIGQUERY

Matthew Emery @lstmemery

matthew.emery44@gmail.com

OUTLINE

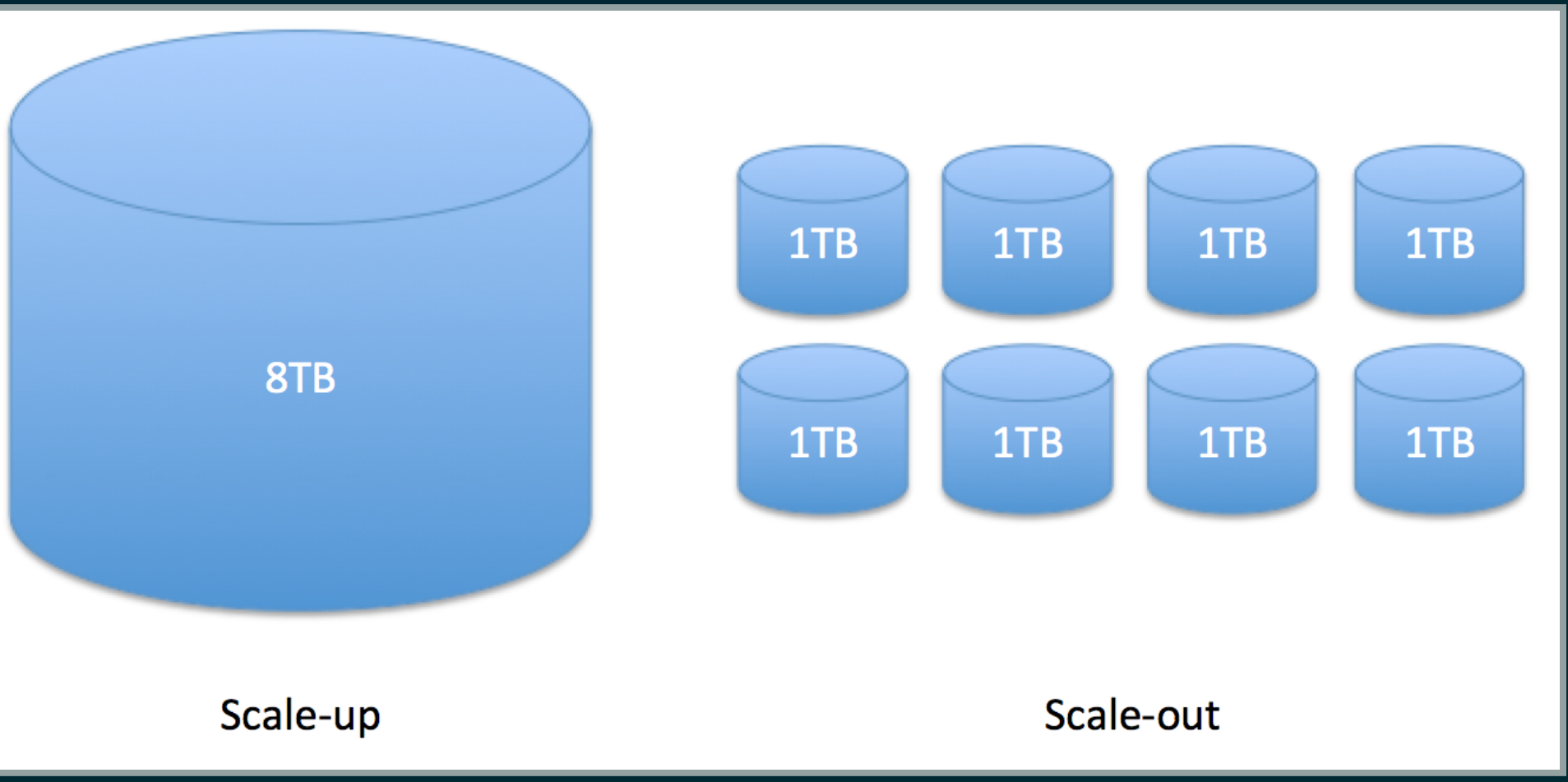
- Why do we need BigQuery?
- Setting up BigQuery
- A Strategy for Big Data
- BigQuery Best Practices
- Exercises

WHY DO WE NEED BIGQUERY?

BEFORE BIGQUERY

SQL was not distributed

- SQL Servers are expensive to scale up
- There are physical limits to scale up
- 1 Server = 1 point of failure


8TB
1TB
1TB
1TB
1TB
1TB
1TB
1TB
1TB
Scale-up
Scale-out

BIG DATA

- The rise of big data made it impossible for Google to continue using one system
- Google invested heavily in early Big Data technologies, such as MapReduce (Hadoop)
- MapReduce is hard to write

THE GOAL OF BIGQUERY

As easy as SQL. As scalable as MapReduce

HOW FAST IS BIGQUERY?

A typical hard drive reads data at 80-160 Megabytes/second = One terabyte in 1.75 to 3.5 hours

BigQuery aims to read data at a 1 Terabyte/second

HOW IS BIGQUERY DIFFERENT FROM REGULAR SQL IMPLEMENTATIONS?

Data is stored in columns not in rows. This means:

- Google can compress data more easily
- Data is more easily distributed
- Indices are not necessary
- **Transactions are harder**

COSTS

Storage: 2 cents per gigabyte per month

Queries: $5 per terabyte per month.

1st terabyte is free

HOW TO USE BIGQUERY

BIGQUERY HAS MANY CONNECTORS

- Python (through pandas)
- R
- Tableau
- ODBC (other databases)
- Excel

R CODE

```
library(bigrquery)
project <- "{your project name}"
sql <- "{you SQL statement in a string}"
query_exec(sql, project = project)
```

PYTHON CODE

```
import pandas as pd
project = "{your project name}"
sql = "{you SQL statement in a string}"
pd.read_gbq(sql, project_id = project)
```

CAVEATS

- BigQuery is NOT strictly relational
- BigQuery is NOT open source
- Do NOT use BigQuery for online transactions

A GENERAL STRATEGY FOR BIG DATA

HOW DO YOU BREAK DOWN A BIG DATA?

The scientific method

1. Observe
2. Hypothesize
3. Experiment
4. Evaluate

OBSERVE

It's difficult to get a handle on

Take a *random* subset of the data and play around with it

I didn't do this in this project and wished I did

Analyze your sample with your tool of choice

```
SELECT
  *
FROM
  [bigquery-public-data:samples.shakespeare]
WHERE
  RAND() < 50/164656
```

HYPOTHESIZE

Try to list your assumptions about the data

What are the minimum/maximum values? Can data be missing?

How can this data add value to your company?

EXPERIMENT

Make a model on yourself subset. Is it predictive?

Plot the distributions of different columns against each other

EVALUATE

Scale your findings to the whole dataset

This strategy is much less expensive than running the very similar queries over and over again.

SETTING UP YOUR ACCOUNT

Go to the Google Developer API console

- Click Create Project
- Click Create
- Give your project a name and create
- Click Google APIs
- Click BigQuery API
- Click Enable
- Go to BigQuery

TEST OUT YOUR BIGQUERY

```
SELECT
  word,
  sum(word_count) AS total_word_count
FROM
  [bigquery-public-data:samples.shakespeare]
GROUP BY
  word
ORDER BY
  word_count DESC
LIMIT
  10
```

BIGQUERY BEST PRACTICES

DON'T USE SELECT * UNLESS YOU MEAN IT

The LIMIT statement does not affect the amount of data you read. This is because BigQuery is a columnar datastore.

Your queries will slower

PREVIEW ITEMS WITH THE PREVIEW TAB

DON'T STREAM IN DATA UNLESS YOU ABSOLUTELY HAVE TO

Batch inserts are free. Streaming inserts are not

DON'T SORT UNTIL YOU HAVE TO

Sorting is computationally expensive. Don't use ORDER BY until you have the subset of data you want

REDUCE AS MUCH DATA AS POSSIBLE BEFORE A JOIN

USE APPROXIMATE AGGREGATION FUNCTIONS WHENEVER POSSIBLE

Note that BigQuery does this automatically!

```
SELECT COUNT(DISTINCT word)
FROM [publicdata:samples.shakespeare]
```

Result: 31719

BigQuery sets a number of bins to include

```
SELECT COUNT(DISTINCT word, 50000)
FROM [publicdata:samples.shakespeare
```

Result: 32786 (Correct)

What if we don't want to guess the number of buckets?

```
SELECT COUNT(*)
FROM (
SELECT word
FROM [publicdata:samples.shakespeare]
GROUP BY word)
```

Result: 32786 (Also Correct)

This is a significantly slower query

EXERCISES

HOW HAS YOUR NAME CHANGED OVER TIME?

Use this table:

```
[bigquery-public-data:usa_names.usa_1910_current]
```

```
SELECT name, year, SUM(number) AS total_total
FROM
  [bigquery-public-data:usa_names.usa_1910_current]
WHERE
  name == "Matthew"
GROUP BY
  year, name
ORDER BY
  year
```

WHAT ARE ALL THE HACKER NEWS STORIES AND TOP COMMENTS ABOUT BIGQUERY?

You will need two tables:

```
[bigquery-public-data:hacker_news.stories]
```

```
[bigquery-public-data:hacker_news.comments]
```

```
SELECT stories.id, stories.title, stories.descendants, comments.t
FROM (
  SELECT id, title, descendants
  FROM
    [bigquery-public-data:hacker_news.stories]
  WHERE
    LOWER(title) CONTAINS "bigquery") AS stories
LEFT JOIN EACH (
  SELECT
    id, parent, text
  FROM
    [bigquery-public-data:hacker_news.comments]
  WHERE
    LOWER(text) CONTAINS "bigquery") AS comments
ON
  stories.id = comments.parent
```

REFERENCES

Tigani J, Naidu S. Google BigQuery Analytics. John Wiley & Sons; 2014. 528 p.