MONASH University

MONASH INFORMATION TECHNOLOGY

RECORD!

Introduction to Big Data

Module 3

TP3 2018

GROUP OF EIGHT AUSTRALIA

Module Intro

1. Introduction to Big Data
2. Hadoop Ecosystem and Applications
   a. introduction to Apache Hadoop framework/architecture
   b. Hadoop distributed File System (HDFS)
   c. MapReduce prog. Model

3. NoSQL 1: Big Data Processing Concepts and Technologies
   a. theory of distributed transactions & concurrency control.
   b. columnar type databases.
   c. Apache HBase architecture/operations

4. NoSQL 2: Spark & distributed graph processing
   a. intro to Apache Spark as a big data processing framework.
   b. Spark architecture/data abstractions/APIs/ETL
   c. intro to graph processing
   d. Spark GraphX
5. Data streams
6. Advanced Topics in Big Data

Module Intro: NoSQL

- is “Not Only SQL”
- isn’t “No SQL”
- makes it easy to deploy and store a wide range of data types, and they excel in performance

Module Intro: SQL vs. NoSQL

- **Storage:**
  - SQL:
  - NoSQL:
- **Flexibility:**
  - SQL:
  - NoSQL:
- **ACID Compliance:**
  - SQL:
  - NoSQL:

Module Intro: SQL vs. NoSQL

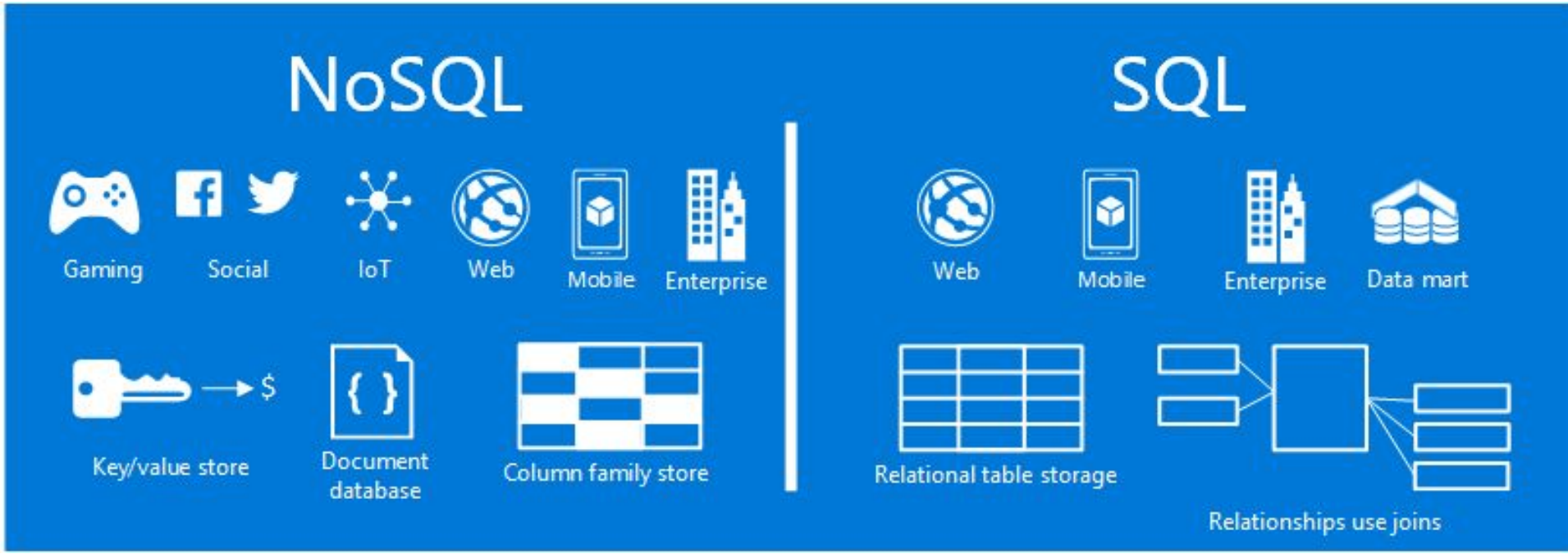
- **Storage:**
  - SQL: relational models (tables with links) where rows comprise of the information concerning a single entity.
  - NoSQL: database types ranging from graph and key-value to document and columnar.

- **Flexibility:**
  - SQL: every record conforms to a predefined schema.
  - NoSQL: information can be updated on the fly (dynamic schema).

- **ACID Compliance:**
  - SQL: databases default to enabling ACID compliance.
  - NoSQL: emphasizes performance over data integrity.

Module Intro: SQL vs. NoSQL

.

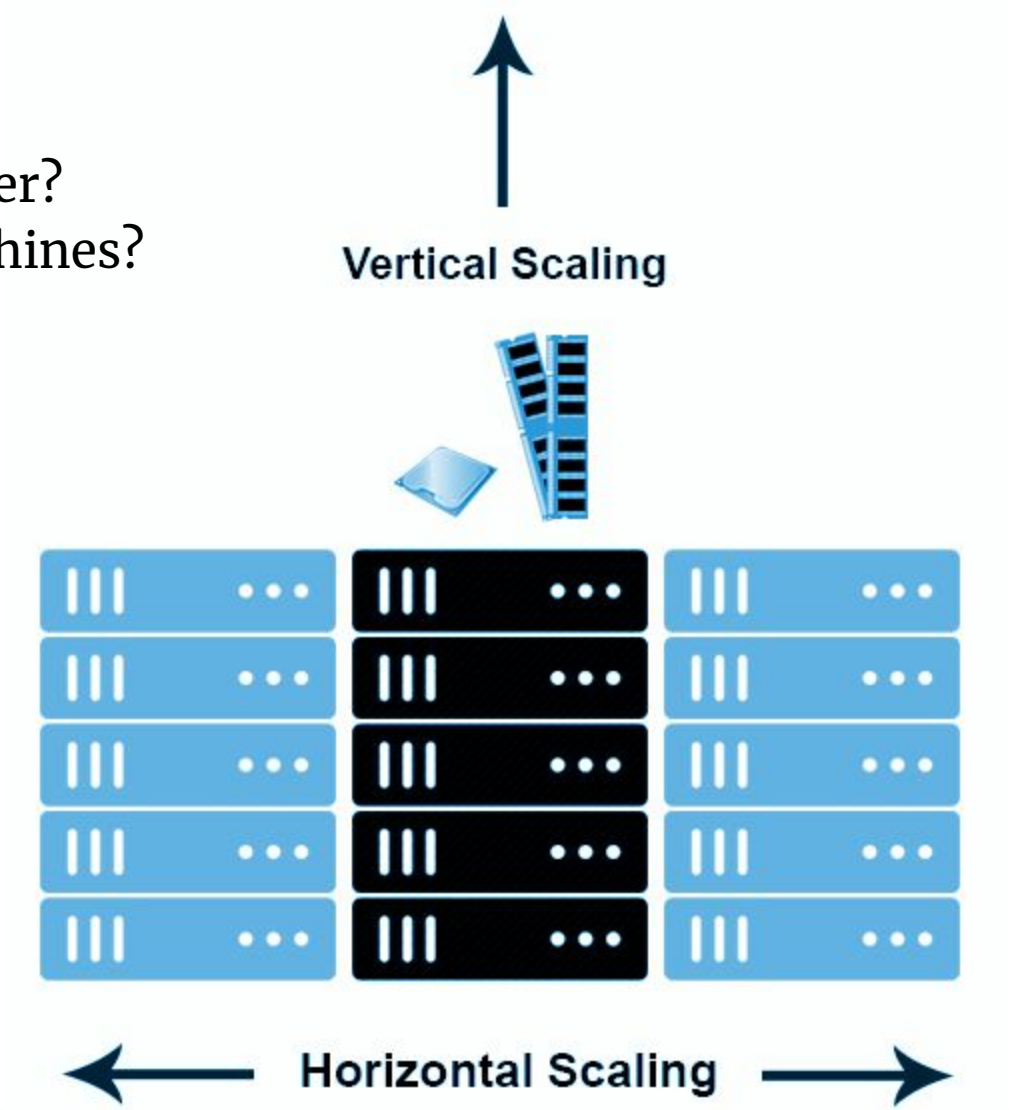


Scalability

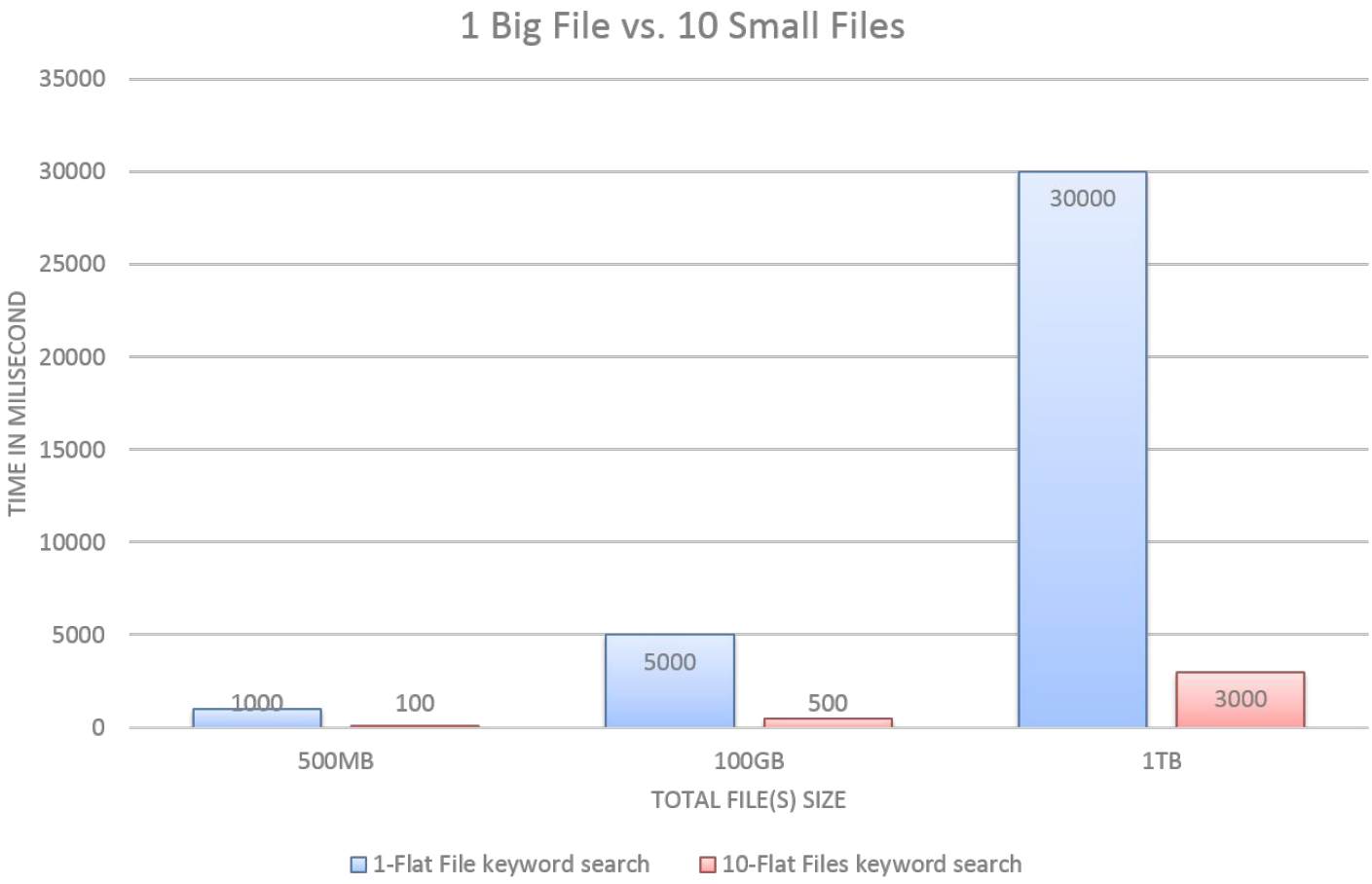
- It is not Big Data if doesn't scale!
- Vertical vs. Horizontal

Vertical vs. Horizontal Scaling

- More power?
- More machines?



Horizontal Scaling via Distributed Processing



We have HDFS, do we need more? Why?

HDFS is not Enough!

- HDFS limitations:
  - HDFS is a file system, not DBMS
  - HDFS is schemaless (treats all as unstructured files)
  - HDFS only supports sequential access in file chunks (what?!)
  - HDFS does not guarantee ACID

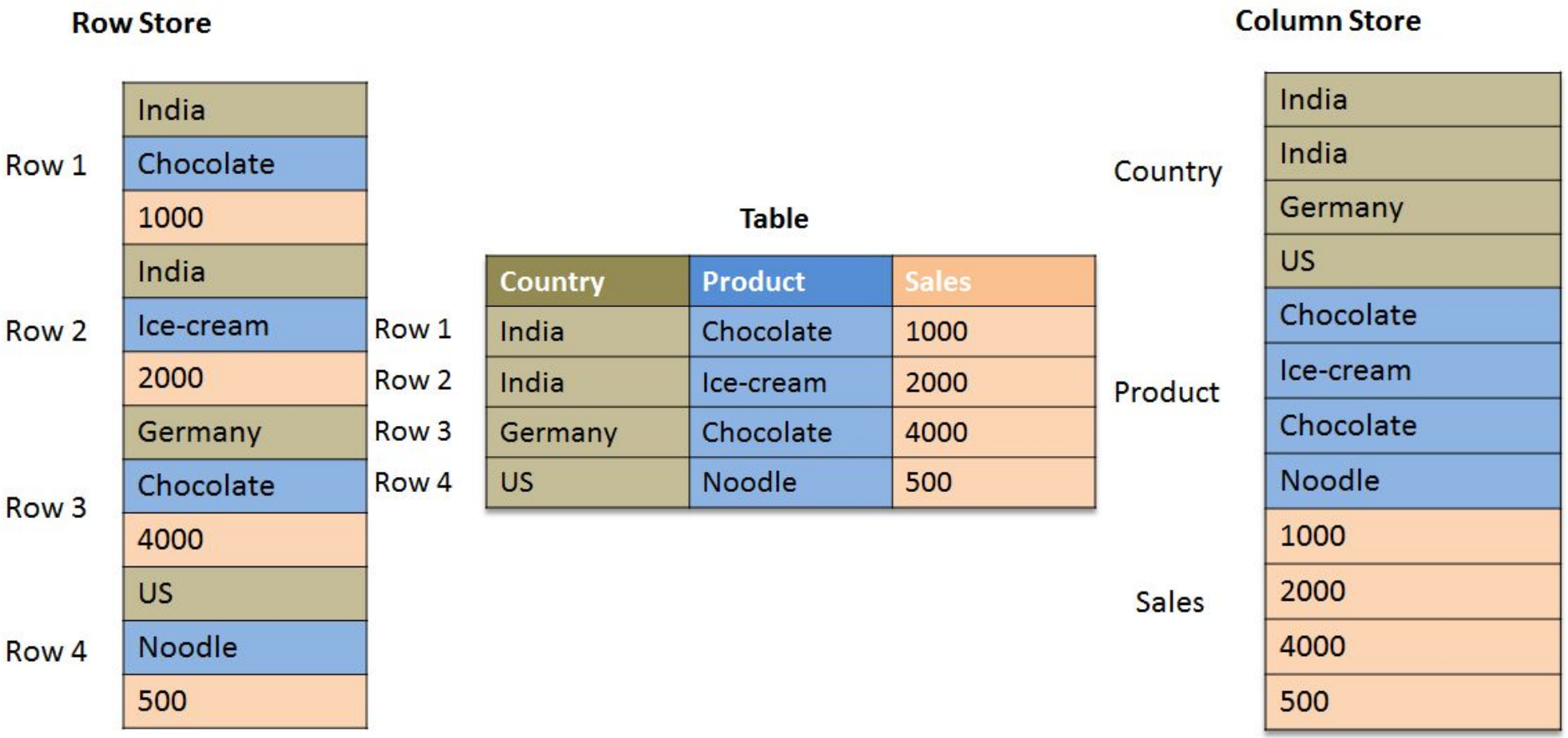
A distributed file system is not enough. We need distributed DBMSs

Row vs. Column-oriented DB

Table

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500

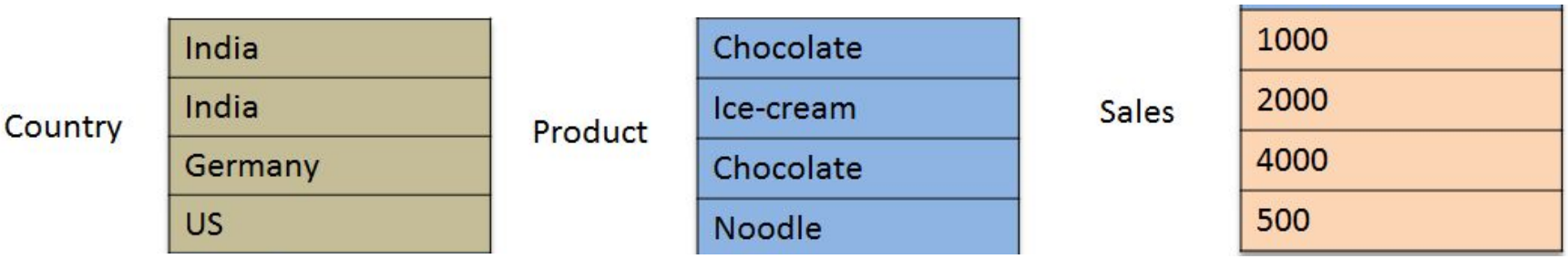
Row vs. Column-oriented DB



Which one is more scalable?

Columnar DB

- Pros:
  - Generally need less number of seeks for **OLAP** operations. *How?*
  - Usually no need for indexing. *Why is it good?*
  - Data can be compressed more efficiently. *Why?*
  - Ideal for evolving DBs. *Why?*
- Cons:
  - Not efficient for **OLTP** usage (*depends!*)



NoSQL DBMS Examples

- Introduce yourself!
- Share your experience using NoSQL database
- Do a quick research and discuss one of the followings
  - Group 1: MongoDB
  - Group 2: CouchDB
  - Group 3: Redis
  - Group 4: Neo4j
  - Group 5: Hive
- Nominate one representative to give a summary to the class

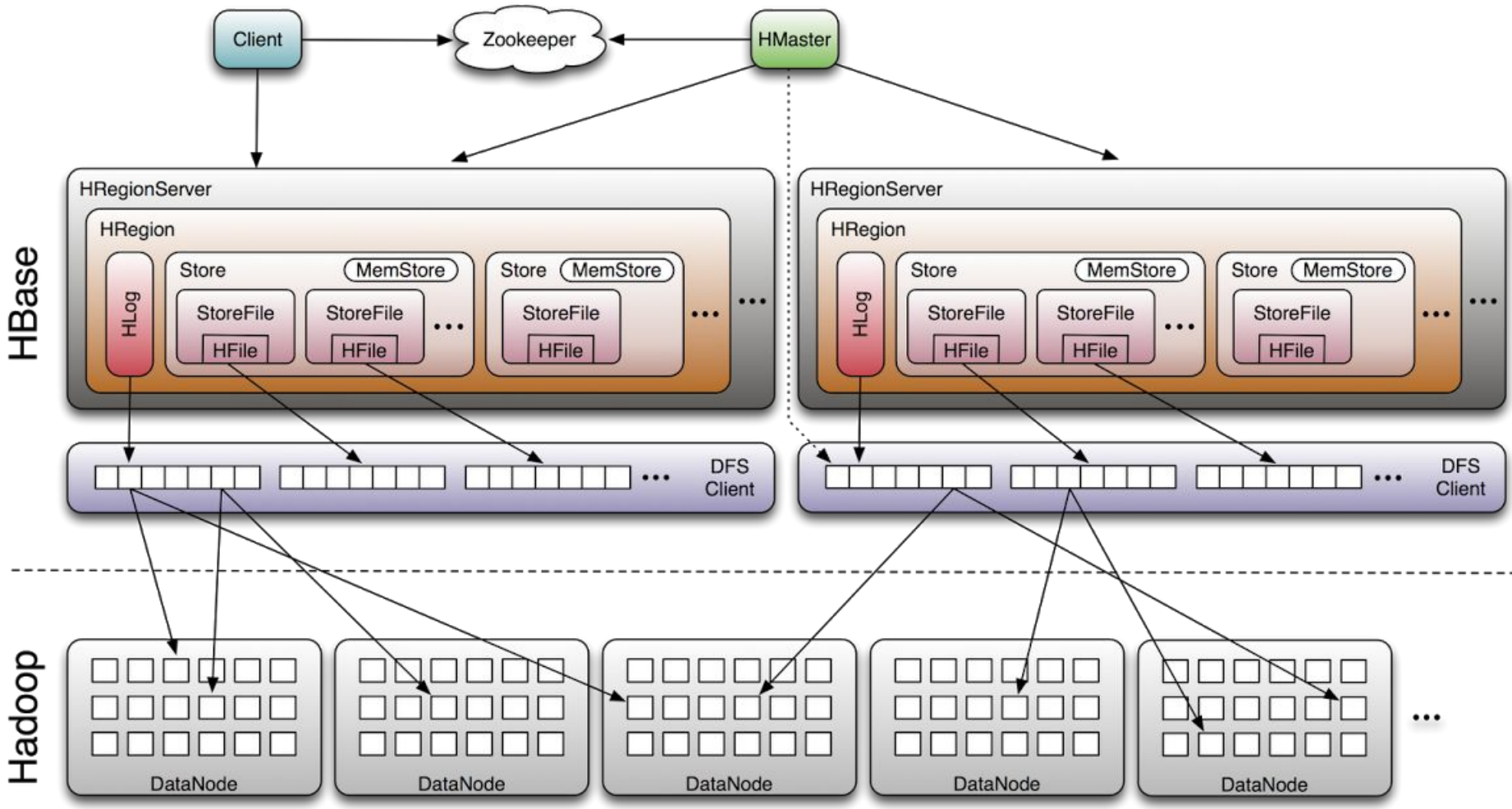
Apache HBase (Hadoop Database)

- A non-relational, distributed, and column-oriented database
- runs on top of **HDFS**.
- **real-time** read/write random access to datasets
- billions rows and millions columns.
- inspired by the famous **Google BigTable**
- does **not** support SQL-like queries

HBase vs. RDBM

Hbase	Relational Database
• It is schema-less • It is a column-oriented data store • It is used to store de-normalized data • It contains sparsely populated tables • Automated partitioning is done in Hbase	• It is a schema based database • It is a row-oriented data store • It is used to store normalized data • It contains thin tables • There is no such provision or built-in support for partitioning

HBase Architecture



HBase Architecture

1. **Zookeeper:**
   - a distributed resource management (centralized coordination service)
2. **Master Server (HMaster):**
   - runs on the *namenode*
   - monitors all instances of the region servers.
   - acts as the common interface for all metadata changes made within the cluster.
3. **Region Servers (HRegion):**
   - usually run on the *datanode*
   - are responsible for managing HBase regions (each of which comprises a subset of rows).
4. **Client (HTable):**
   - is responsible for finding region servers that are serving the particular row range of interest.
5. **Catalog Tables (HLog)**
   - stores the metadata and contains information on data distribution among the region servers.

HBase Architecture

1. **Zookeeper**
2. **Master Server (HMaster)**
3. **Region Servers (HRegion)**
    - 3.1. **Block Cache:** used for fast in-memory access to more frequently read data items
    - 3.2. **MemStore:** a fast in-memory storage for the transactions that have not committed yet.
    - 3.3. **Write-Ahead-Log (WAL):** Region server updates are written to WAL first, and then to MemStore to ensure a durable write (like MySQL BIN log).
4. **Client (HTable)**
5. **Catalog Tables (HLog)**
    - 5.1. **Root Table:** It keeps track of the location of Meta table.
    - 5.2. **Meta Table:** It stores a list of all regions in the system.

HBase Challenge 1

Create a large table in HBase

1. Download the datasets from:
   http://samplecsvs.s3.amazonaws.com/SalesJan2009.csv
   http://spatialkeydocs.s3.amazonaws.com/FL_insurance_sample.csv.zip
1. Transfer them to HDFS
2. Create a table for each dataset using HBase Shell (call them *sales* and *insurance*)
3. Import the files to populate the tables using *importtsv* utility
4. Count the number of rows

How many rows do we have?

HBase Challenge 2

Modify table in HBase

1. Add one row to a table,
2. Scan the table to find it,
3. Change one or more values of the added row,
4. Drop the row

HBase Challenge 3

Analyse data in HBase

1. Create some questions such as:
   a. What is the number of k==v
   b. What is the largest/smallest/sum/mean value in k
   c. What is the largest/smallest/sum/mean value in k1 when k2==v2
   d. What is the largest/smallest/sum/mean value in k1 when k2==v2 and k3>=v3
2. Answer the above questions!
3. Share your code and results in the Forum

Note that: scanning in HBase Shell (specially with *ValueFilter*) is not the best practice when it comes to working with massive tables (it doesn't use MapReduce). If you are interested, develop some MapReduce programs to query HBase tables to receive the best performance.

Assignment 2 Preparation

- You need to able t upload files **<50 MB** to your (local or cloud) BigVM
- You need to be able to perform file operations HDFS
- You need to be able to prepopulate an HBase table using *importtsv*
- You need to use *put*, *get*, *delete*, *scan* (with *FILTER*), *drop*, *list*,... commands in HBase.
- You got only **ONE** attempt and only **ONE** hour
- It is an open-book assignment
- You need to write .sh file that runs HDFS and HBase commands

Assignment 2 Preparation

```
#!/bin/bash
echo "Student Full Name"
echo "Student Number"

echo "1. Create a folder on the HDFS and name it after your student number: /tmp/<student_number>"

# write your code here (uncomment the line!). For example:
Hadoop fs -mkdir -p /tmp/12345678

# add comments using echo command:
echo "An example of irrelevant comment!"
```

Assignment 2 Preparation

```
#!/bin/bash
echo "Student Full Name"
echo "Student Number"

echo "# 2. Complete this to create a table called <student_number> using HBase Shell commands.
# write your code here:
create '12345678',...
# add comments when necessary (in HBase start comment line using hash sign)
" | hbase shell
```

Assignment 2 Preparation

To run the codes:

```
# we make your codes executable:
chmod +x <student_number>_q?.sh

# then, we run your answer to q1:
./<student_number>_q1.sh

# finally, we try your answer to q2:
./<student_number>_q2.sh
```

MONASH
University

Any (simple) Question?


MONASH
University
GROUP
OF EIGHT
AUSTRALIA