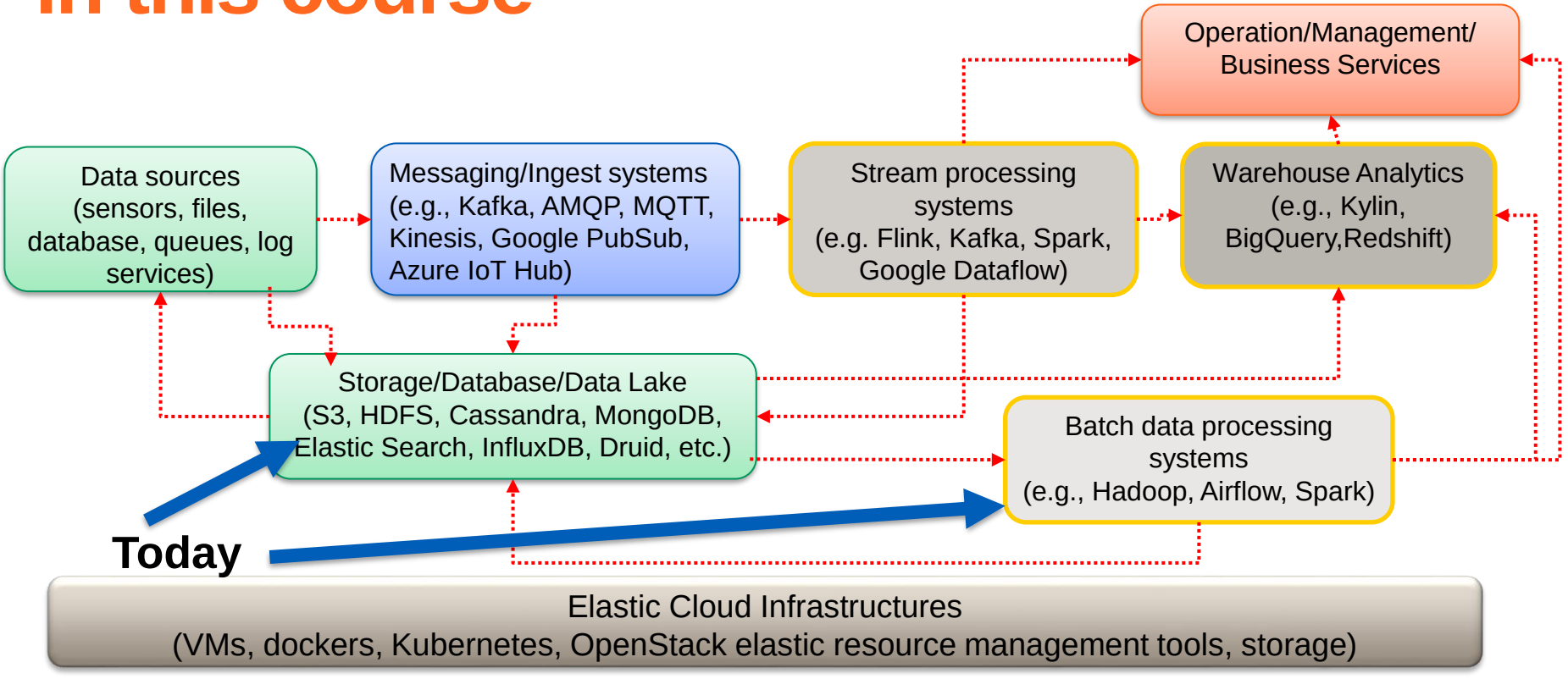
Hadoop and its Big Data Ecosystems

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, https://rdsea.github.io

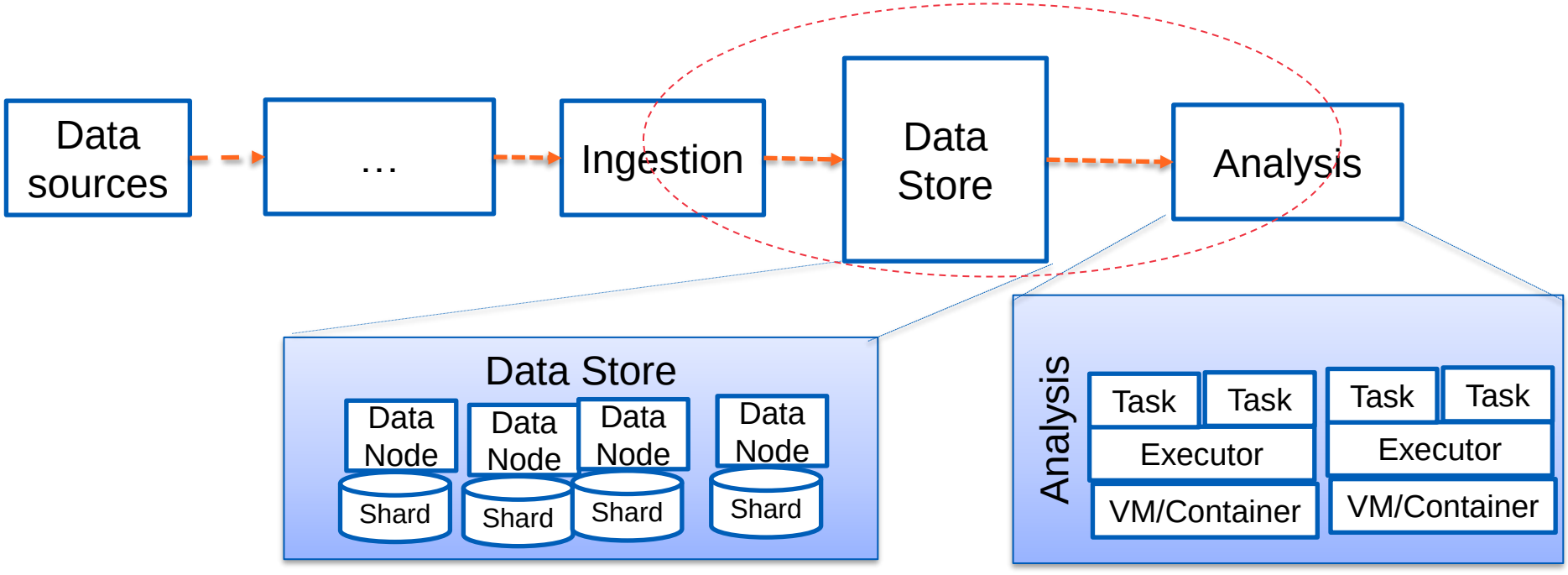
Learning objectives

- Understand massive scale data management and processing with Hadoop
- Understand and able to use Hadoop components for big data platform designs
- Able to integrate Hadoop with other frameworks for data ingestion and analytics systems

Big data at large-scale: the big picture in this course

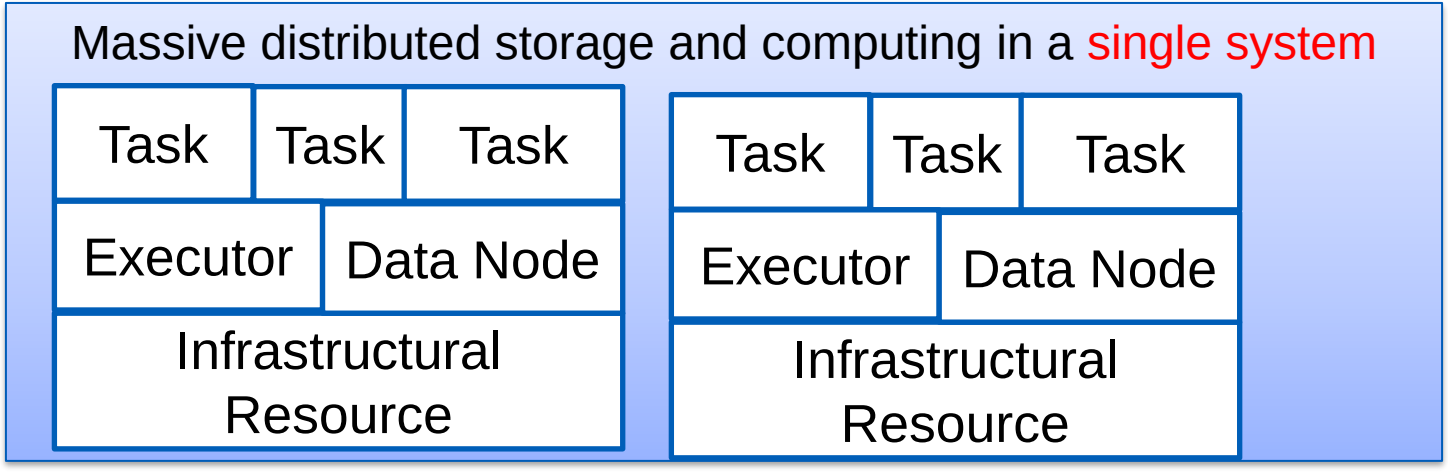


Consider again our big data pipelines



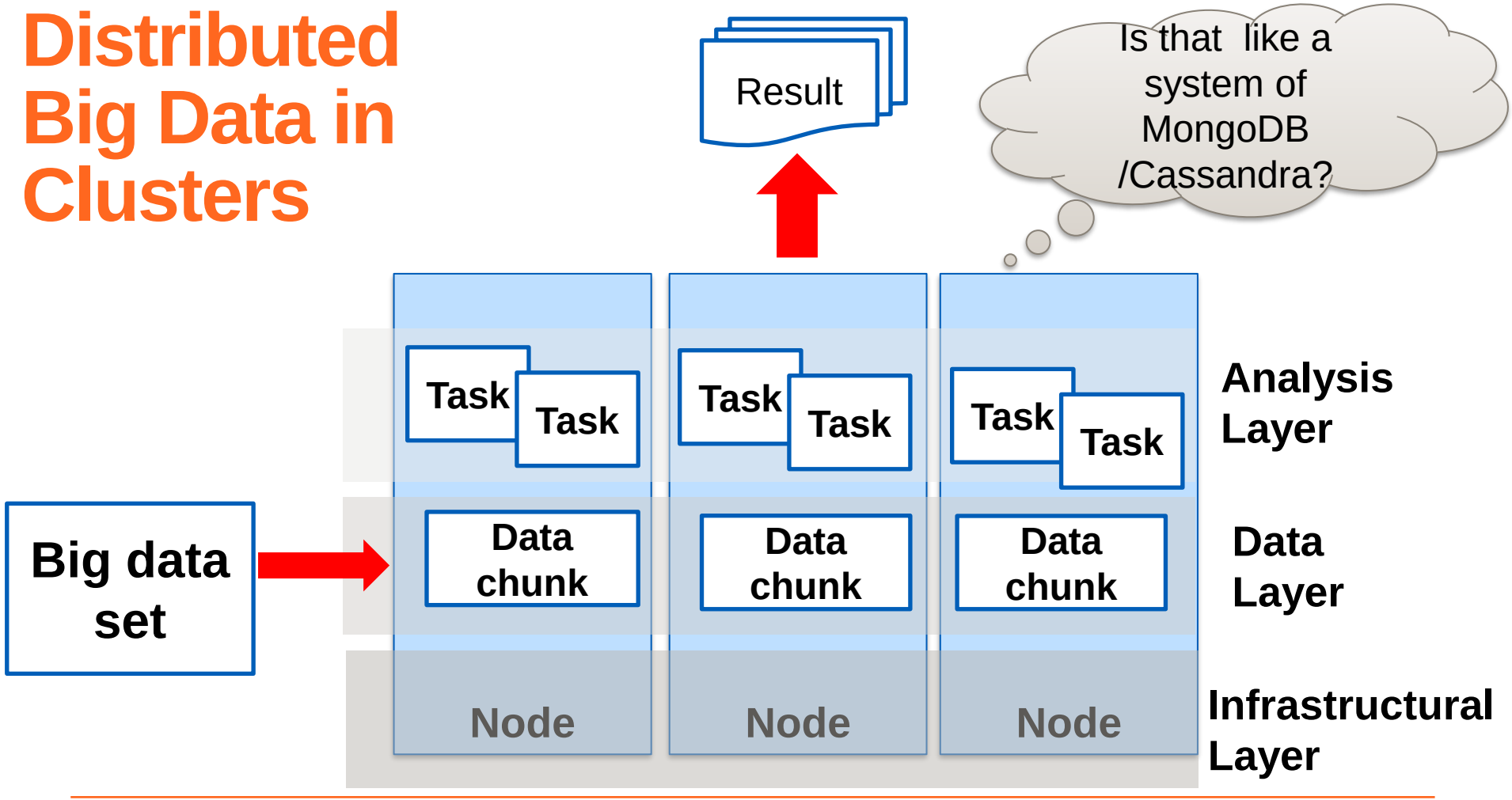
Combing data storage and analysis

Data locality, massive parallel and distributed data processing with large-scale data store



Single system: many machines connected into a type of "cluster"

Distributed Big Data in Clusters



Benefit

- Moving data is much more expensive than moving computation
- Consolidate and integrate various types of data
- Save shared infrastructural resources
- Solutions are suitable for different types of customers/clients and different infrastructures

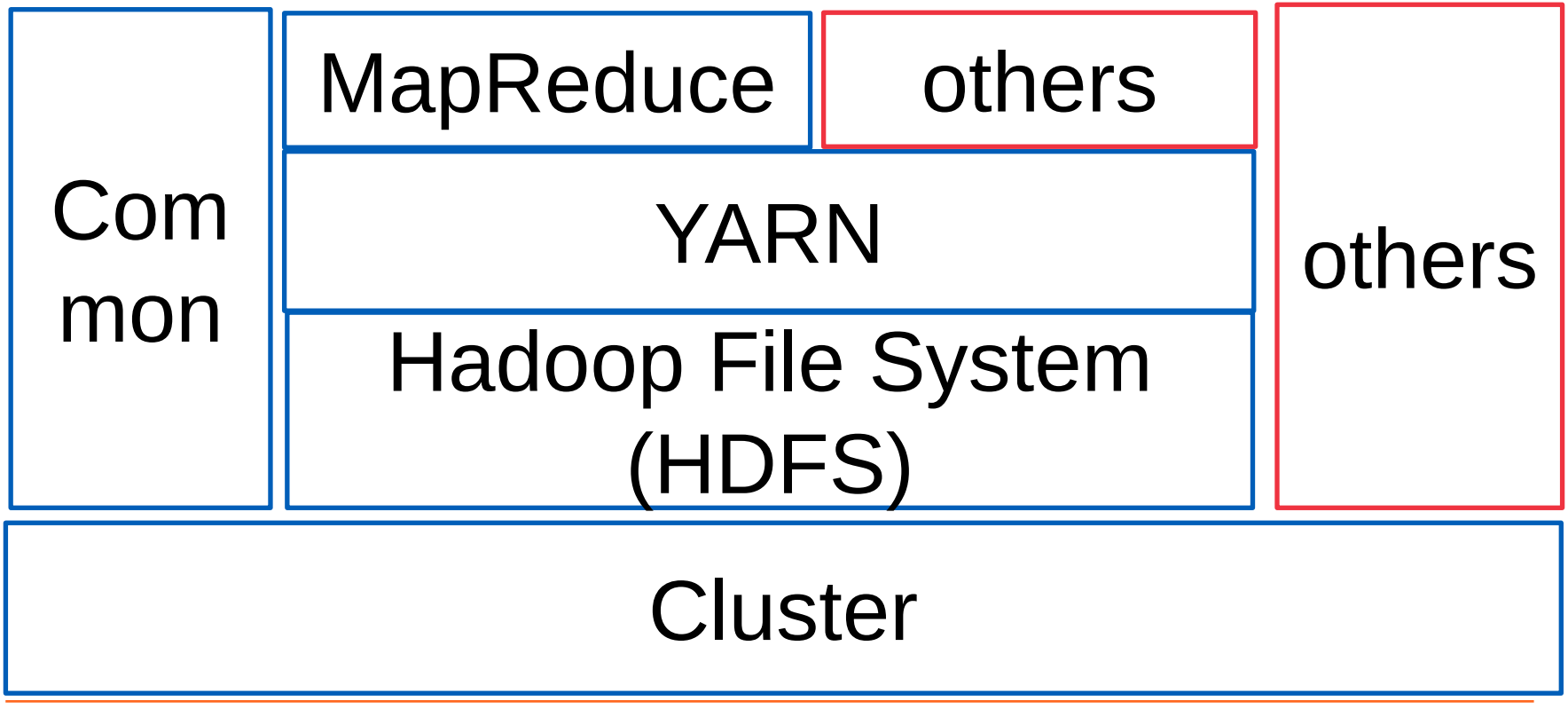
Benefit - Variety

- **Multiple data formats**
- **In big data we have a lot of files**
  - Should we always transform them into a single format?
  - Must always be in a database form?
- **Processing files directly is a benefit!**
- **The datalake concept:**
  - multiple types of big data analytics with high concurrent/parallel data writes/reads
  - dealing with different data access frequencies: **hot, warm and cold data**

Hadoop

- **http://hadoop.apache.org/, original from Yahoo**
- **The goal is to combine storage and processing in the same cluster system**
- **Designed for *massive scale of data and computing***
  - Commodity hardware, highly scalability, fault tolerance, easy to extend
- **Suitable for both on-premise and clouds**
- **There are very rich software ecosystems centered around Hadoop**

Hadoop: Layers



Hadoop key components

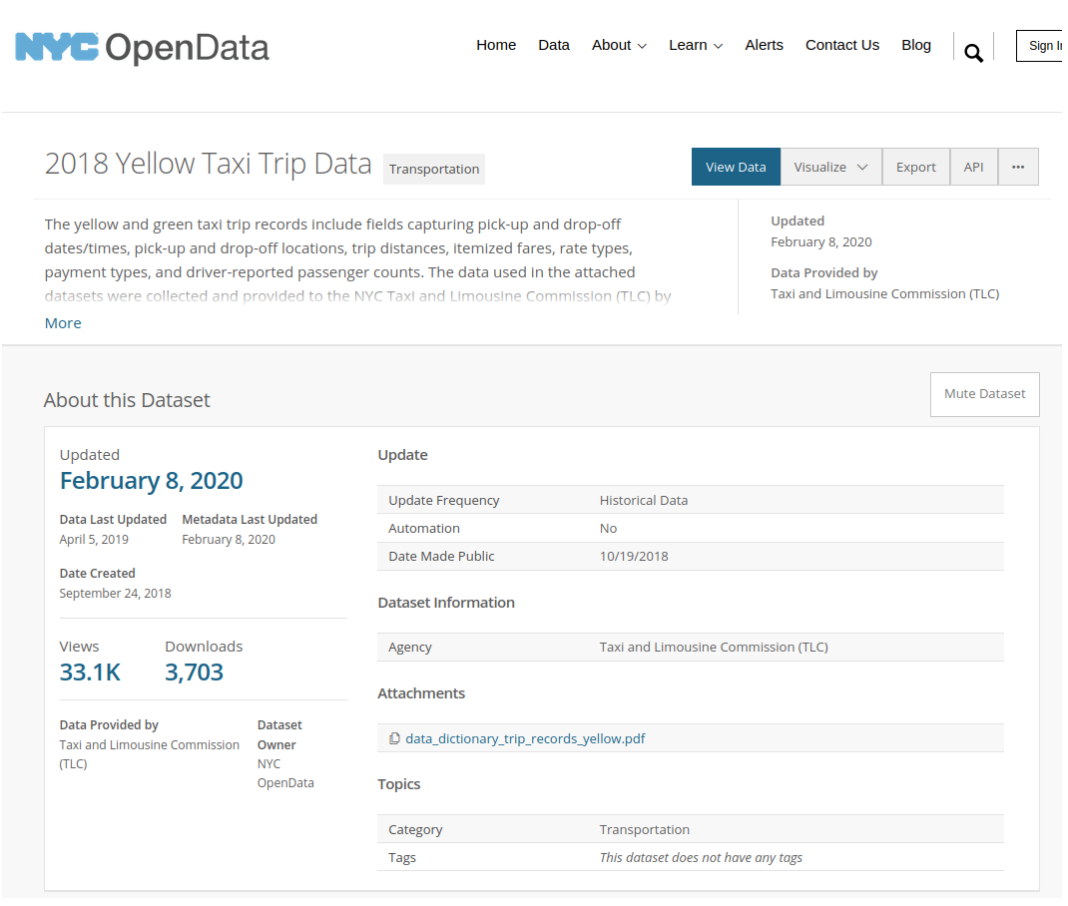
- **HDFS as a distributed file system**
  - for managing data
- **YARN as a resource management system**
  - for executing and managing tasks
- **MapReduce as one programming model**
  - for MapReduce applications
- **Coordination (ZooKeeper)**
  - for fault tolerance and metadata

Hadoop File System (HDFS)

- **For handling very big data files**
  - GBs of data within a single file
- **Assume model of data**
  - Write-once-read-many
  - It is not suitable for random-access update
- **Deal with hardware failures, support data locality, reliability**

Example

e.g., 112M rows

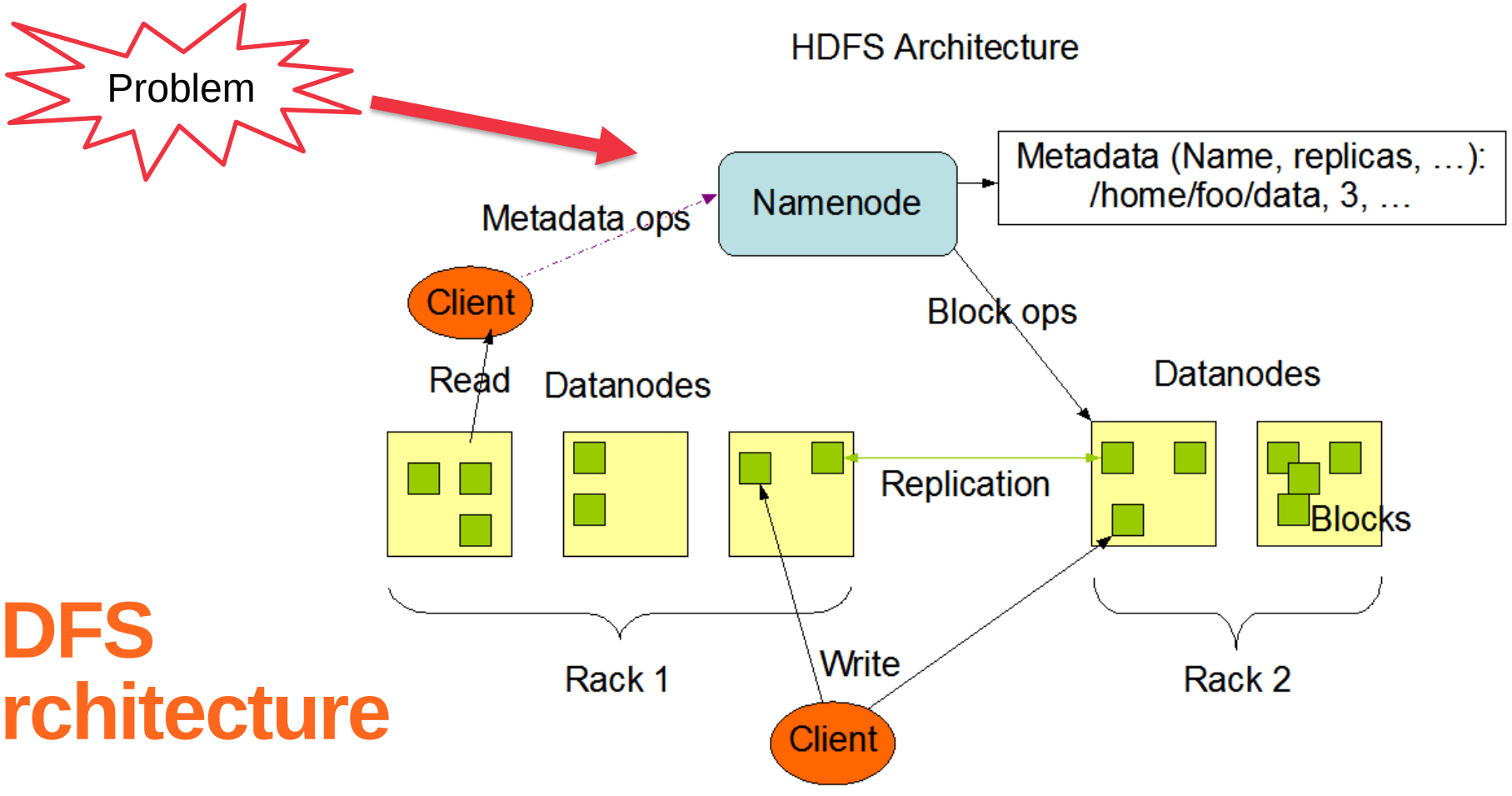


Snapshot from: https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data/t29m-gskq

HDFS - data blocks

- **Files are stored in many nodes**
  - but we access them just like “typical file systems”
- **A file includes many blocks**
- **File blocks are replicated and distributed across nodes**
- **Conventional way of access data**
  - naming resolving: hdfs://
  - common operations: list, put, get, …

HDFS Architecture



Source: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

File blocks, metadata and data replication

- **Block size is 128MB (default)**
  - Can be configurable but should not small size
  - All blocks of the same file are the same, except the last one
- **Data is replicated across the cluster**
  - Usually, replication factor is 3
- **NameNode manages file system metadata**

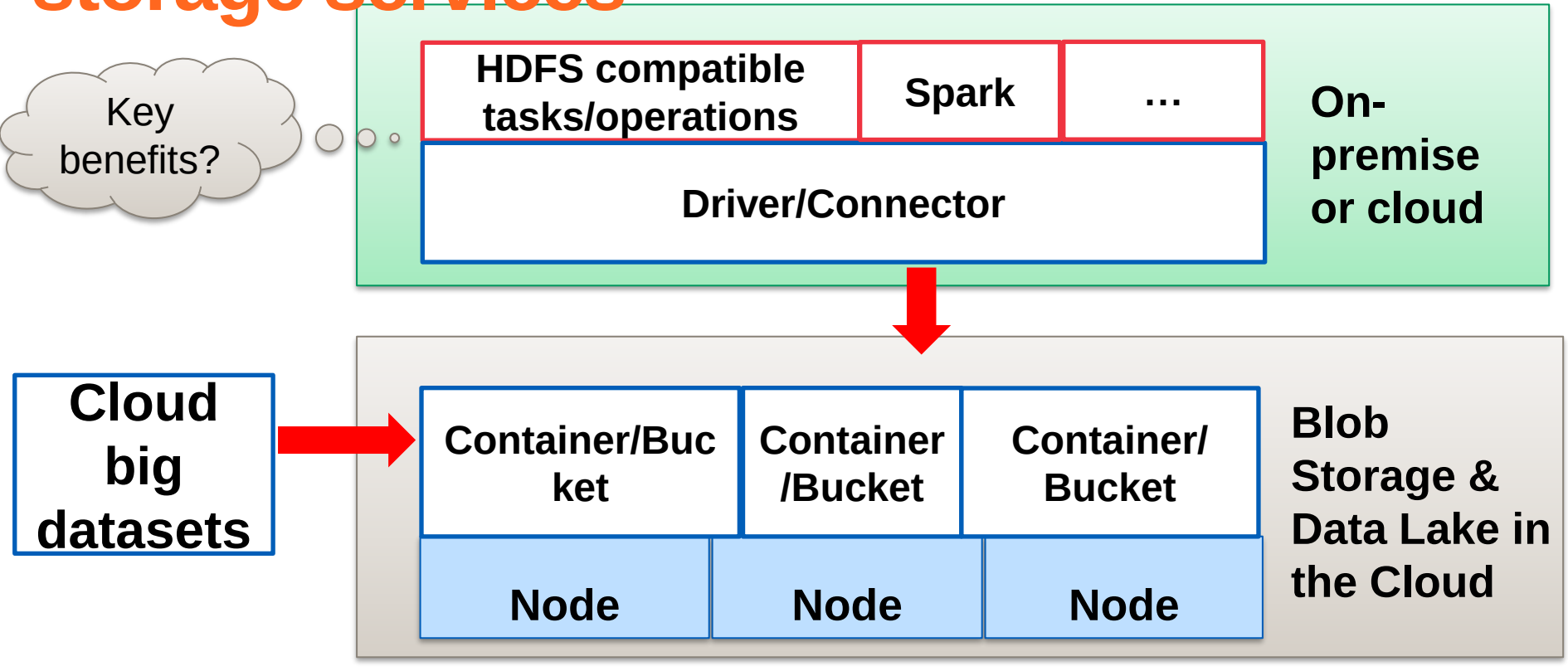
HDFS Fault Torelance

- **Data blocks**
  - File blocks are replicated and distributed across nodes
- **Replica placement using “Racks”**
  - Avoid communication problems between nodes in different racks
- **Monitoring**
  - DataNode reports to NameNode
- **Read and write**
  - Using NameNode for metadata and for information of DataNodes
  - NameNode has replication (master-slave)

Compatible file systems with HDFS

- For integration and analysis purpose: many file systems are compatible with HDFS
- Amazon S3
- Azure Blob Storage
- Azure Data Lake Storage
- OpenStack Swift

Integration models with other cloud storage services

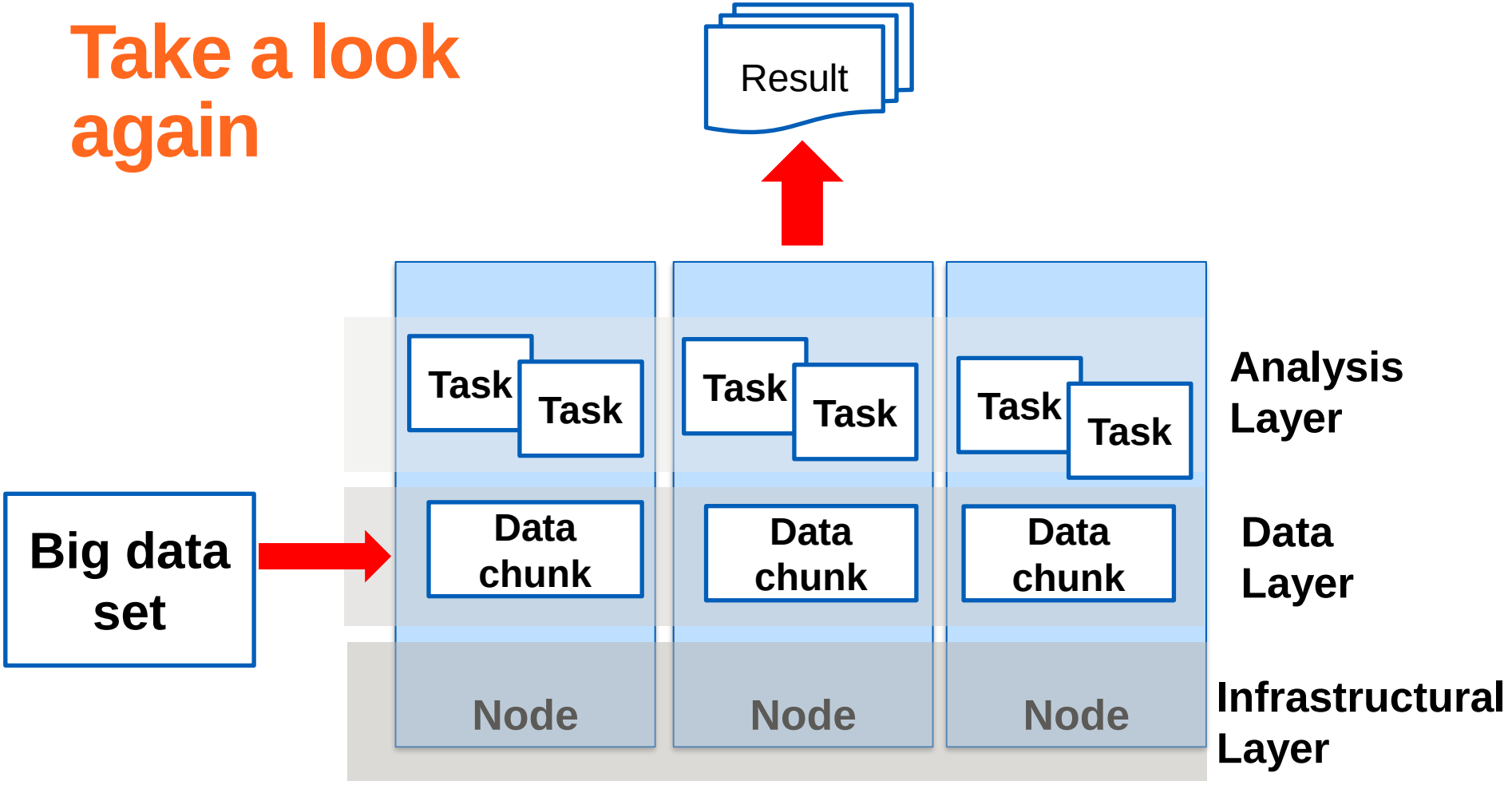


YARN (Yet Another Resource Negotiator)

If HDFS can be used to store different files, what would be the good way to enable "data processing" atop HDFS?

Key requirements: scalability/elasticity, high utilization, serviceability for multiple users/application types

Take a look again



How to leverage the same infrastructure and data management layer to perform data analysis?

How to enable different (distributed) programming models?

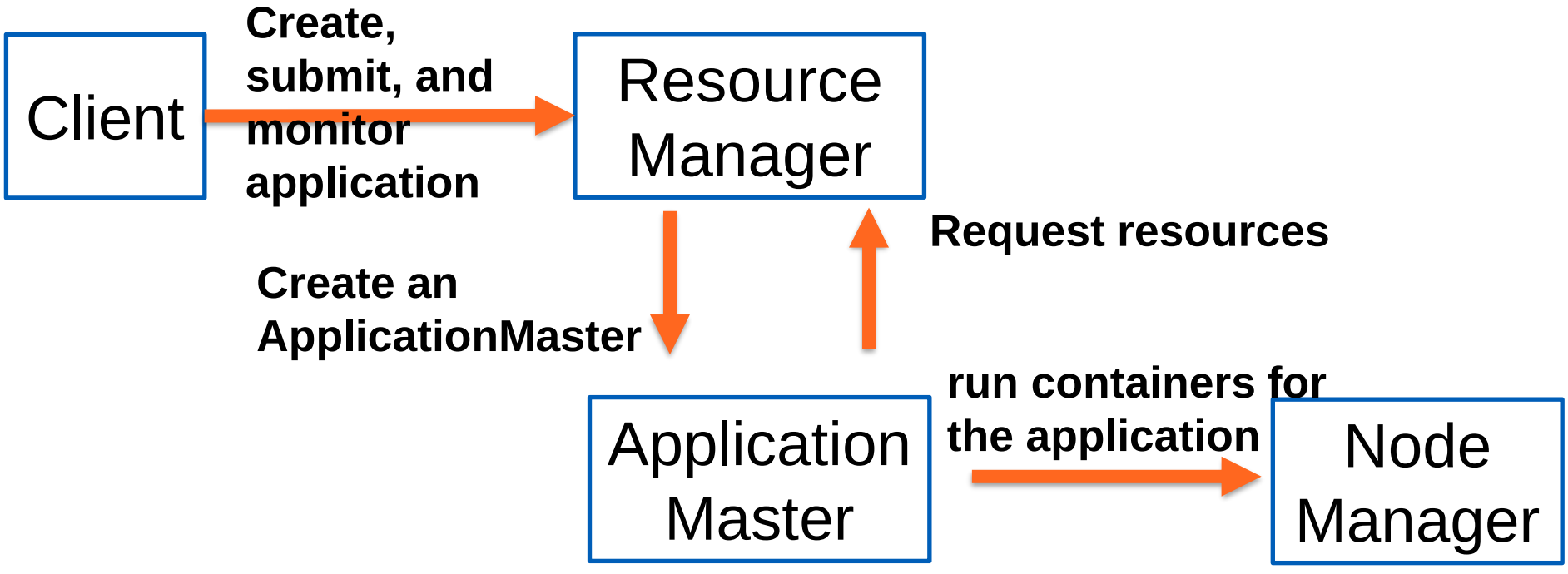
YARN (Yet Another Resource Negotiator)

- **Manage Resources for Processing Tasks**
  - Each node in the cluster provides resources for executing tasks
- **Resource types:**
  - CPU, Memory and Disks
  - Also support GPU and FPGA Node
- **Resources are abstracted into "Containers"**
  - Don't confuse it as a (Docker) container!
- **Multi-tenancy support**

YARN Components

- **Resource Manager**
  - Scheduler: how to schedule tasks atop resources
- **NodeManager**
  - for managing resources of execution tasks in a node
- **ApplicationMaster**
  - Application-specific manager for each application
  - Work with Resource Manager and NodeManager provisioning resources and manage application execution
  - Handle application-specific tasks

YARN basic model



YARN Architecture

User applications running within "Containers":

This model decouples from specific types of applications

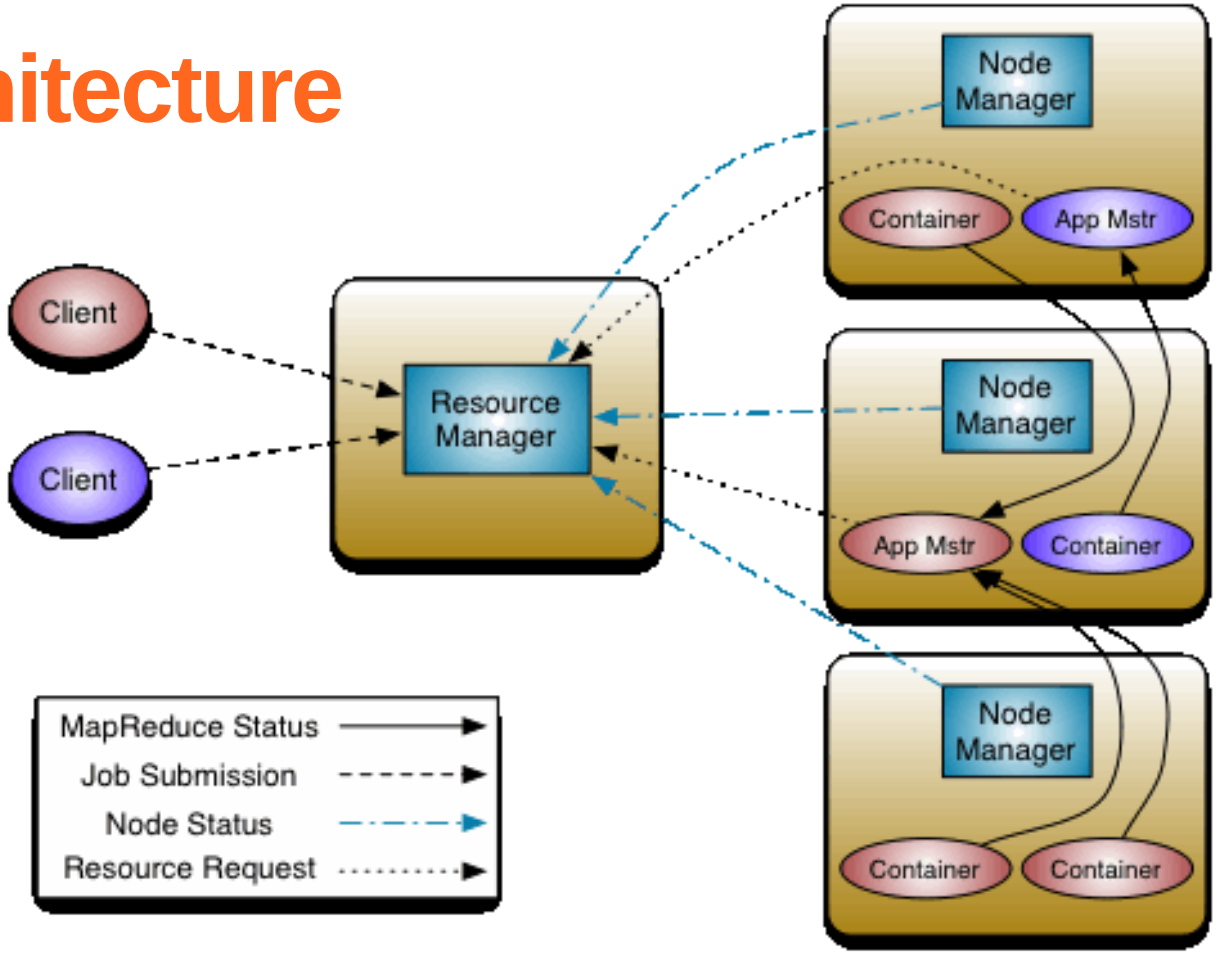


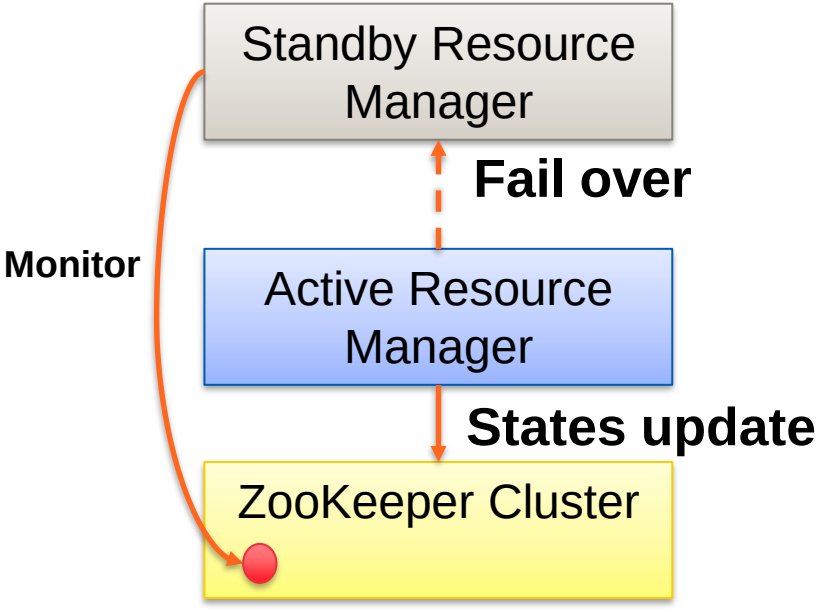
Figure source: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

Scheduling of tasks

- **Scheduling**
  - FIFO
  - CapacityScheduler
    - *Use multiple queues, each with a limit of resources*
  - FairScheduler
    - *All apps will get an average share of resources over time*
- **You can research and add new types of scheduling algorithms**

Fault tolerance

- **Resource Manager is a critical component**
  - Active-passive Resource Manager
  - Zookeeper quorum failover
- **ApplicatonMaster**
  - ApplicationMaster is application-specific
  - Resource Manager restarts AM
- **Node**
  - Remove out of the cluster



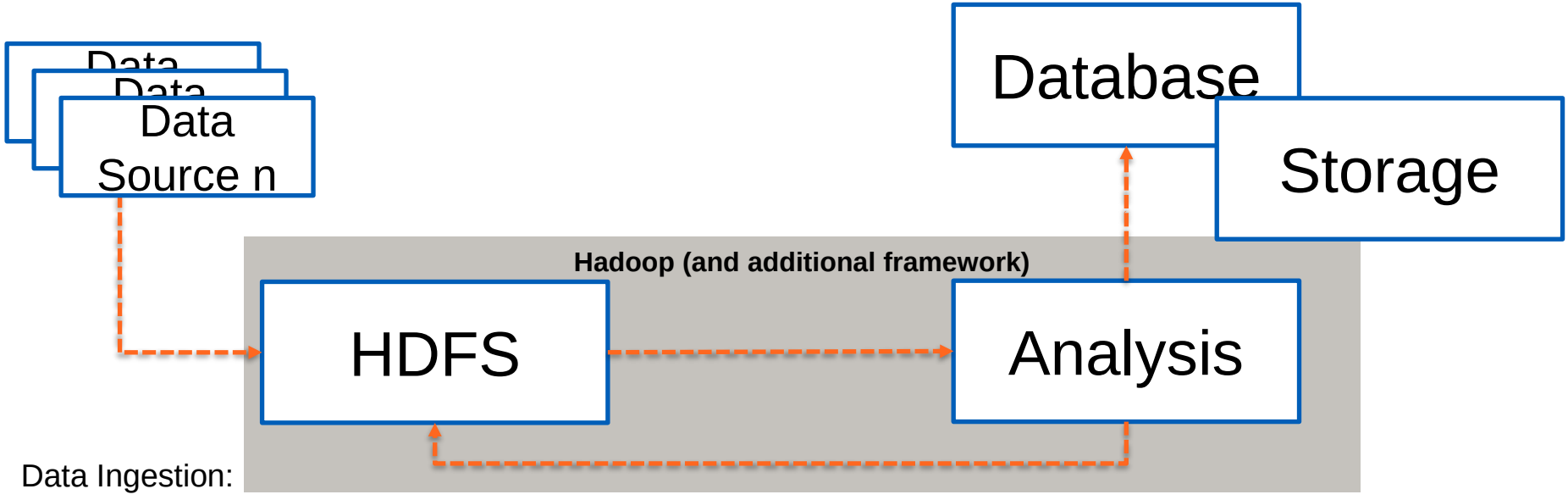
Programming models

- **YARN allows different programming models for applications**
  - MapReduce
  - Apache Spark
  - Workflows
    - *E.g., Apache Tez*

Integration models

- **Using Hadoop for developing large-scale data analysis**
  - Apache Spark, HBase, Hive, Apache Tez
- **Using Hadoop HDFS as components in a big data system**
  - Hadoop HDFS can be come data store
  - Emerging datalake models, combined batch and stream ingestions for incremental processing
    - *e.g., Apache Hudi*

Using Hadoop as part of ETL



- Data Ingestion:
  - Sqoop
  - HDFS Client/Hadoop Streaming
  - Spark Streaming
  - Apache Kafka
  - Apache Nifi
- Computing/Data Processing Framework
  - Apache Spark
  - Hadoop MapReduce
  - Apache Tez

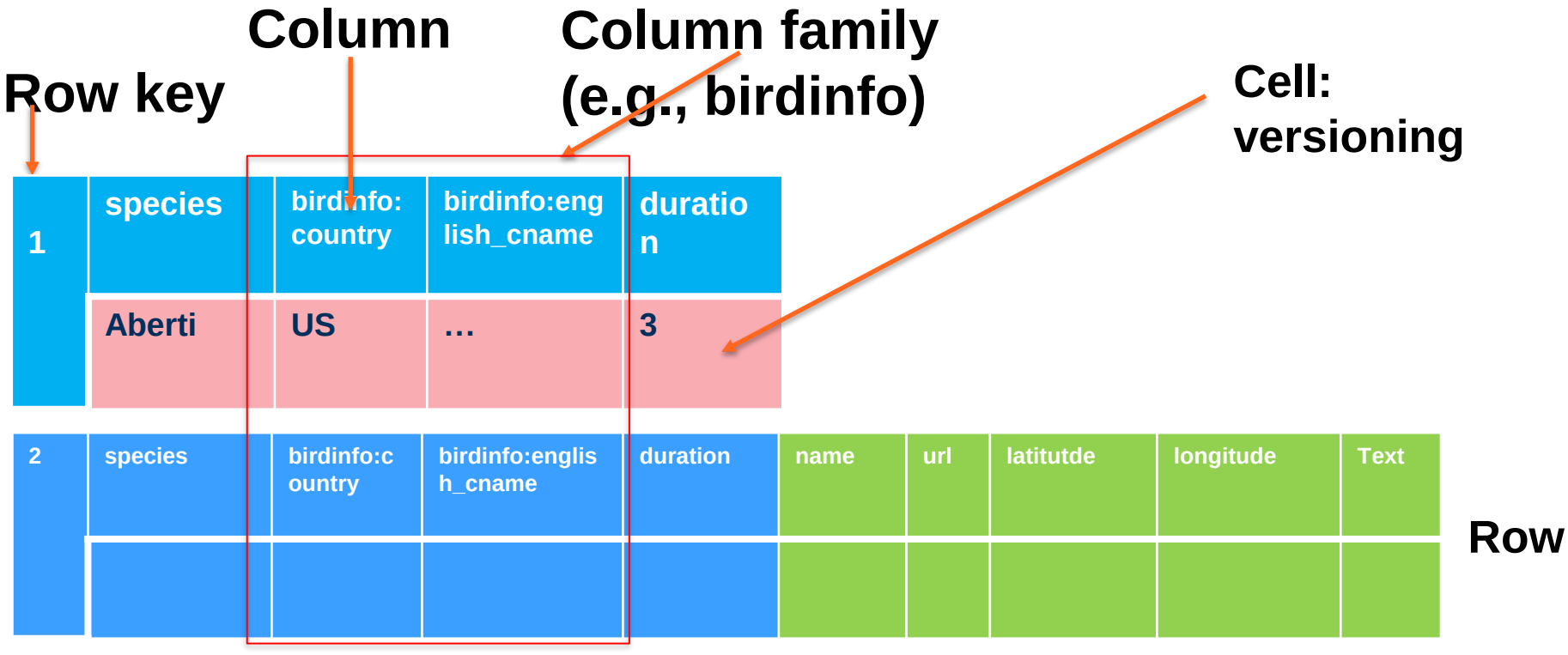
Hadoop-native big database/data warehouse systems

HBase

- **NoSQL database atop Hadoop**
  - Use HDFS for storing data
  - Use YARN for running jobs on data
- **Follow a master-based architecture**

Reading – Why HBase?
https://engineering.fb.com/2010/11/15/core-data/the-underlying-technology-of-messages/
https://engineering.fb.com/2014/06/05/core-data/hydrabase-the-evolution-of-hbase-facebook/
https://engineering.fb.com/2018/06/26/core-data/migrating-messenger-storage-to-optimize-performance/

Example of a data model in HBase



Data Model

- **Example with families: birdinfo, songinfo, location**

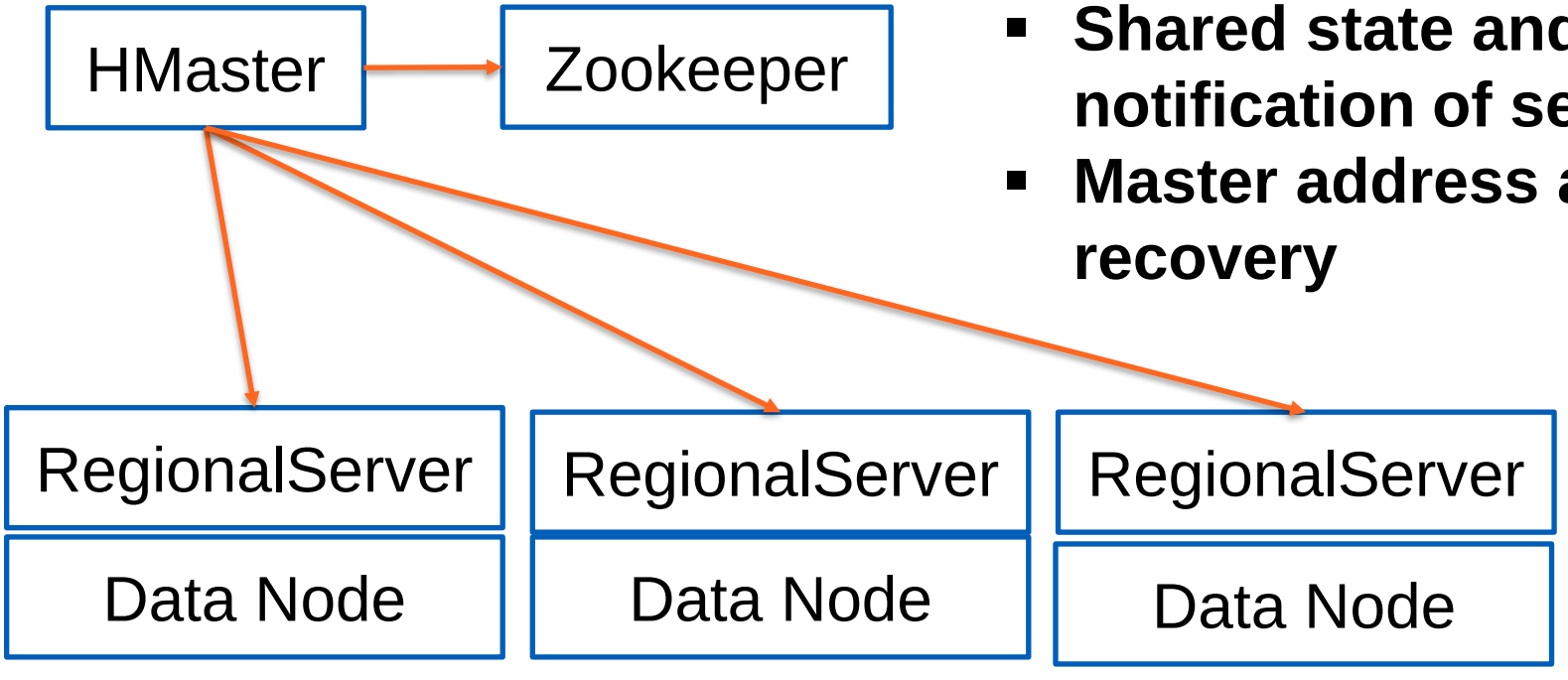
```
hbase(main):045:0> scan 'hbird0'
ROW                          COLUMN+CELL
 17804                       column=birdinfo:country, timestamp=1570281245749, value=United States
 17804                       column=birdinfo:english_cname, timestamp=1570280912421, value=Aberts Towhee
 17804                       column=songinfo:duration, timestamp=1570280912454, value=3
 17804                       column=songinfo:file_id, timestamp=1570280912487, value=17804
 17804                       column=songinfo:file_name, timestamp=1570280912510, value=XC17804.mp3
 71852                       column=birdinfo:country, timestamp=1570281835120, value=Mexico
 71852                       column=birdinfo:english_cname, timestamp=1570281835164, value=Ash-throated Flycatcher
 71852                       column=birdinfo:species, timestamp=1570281835196, value=cinerascens
 71852                       column=location:latitude, timestamp=1570281835211, value=32.156
 71852                       column=location:longitude, timestamp=1570281835225, value=-115.79299999999999
 71852                       column=songinfo:duration, timestamp=1570281835148, value=28
 71852                       column=songinfo:file_id, timestamp=1570281835180, value=71852
```

Enable analytics based on column families (as well as data management)

Data Model – Sharding and Storage

- **Table includes multiple Regions**
  - A region keeps related row data of a table (partitioning)
- **Auto-sharding**
  - Regions are spitted based on policies
- **Region has multiple column families**
  - Different column families will be stored in different files
  - HFiles are used to store real data (also include index data)

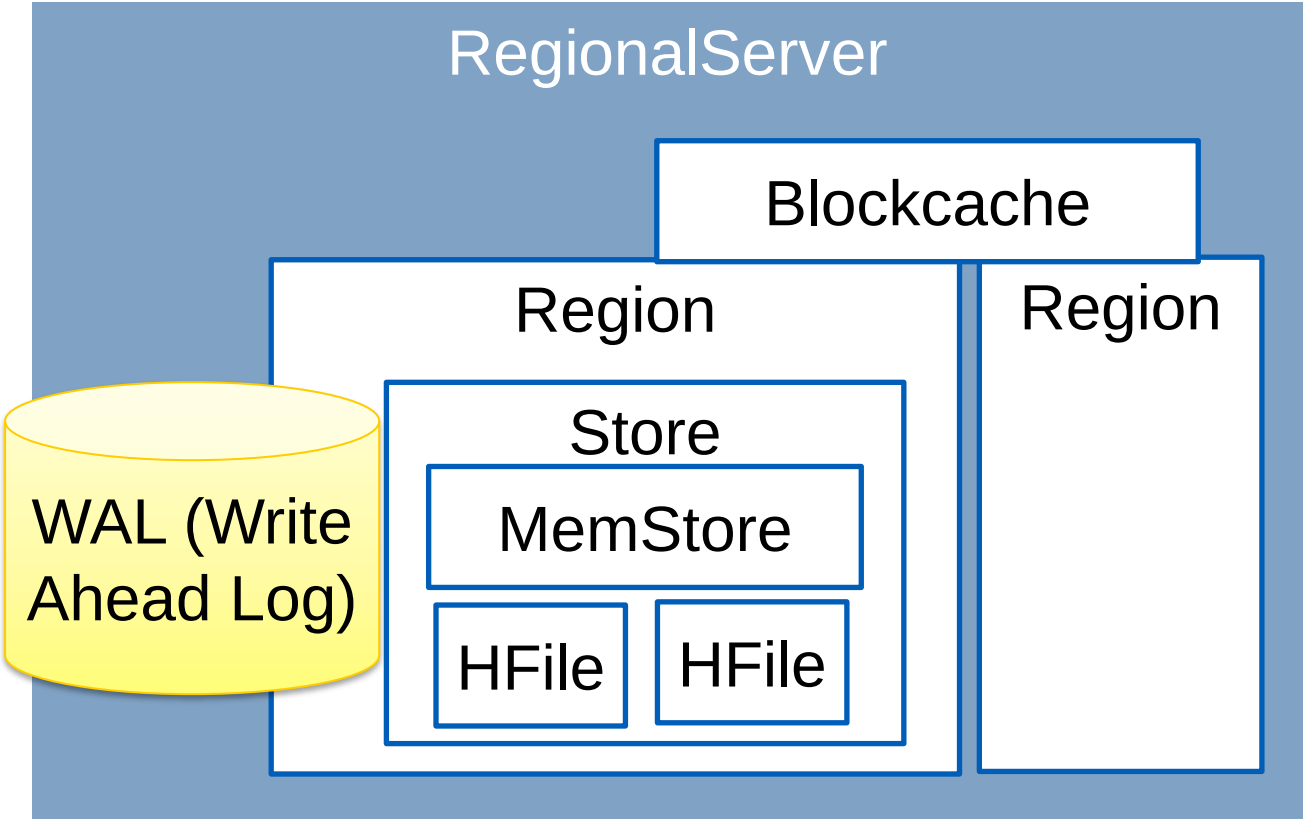
HBase Architecture



Zookeeper:

- **Shared state and failure notification of servers**
- **Master address and recovery**

HBase Architecture



MemStore: write cache for data in memory before written into files

BlockCache: for read cache

WAL is for durability

ACID

- **Atomic within a row**
- **Consistency**
  - Can be programmed: e.g., STRONG and TIMELINE (read might not be consistent)
- **Durability**
  - Can be programmed
  - WAL (write ahead log)

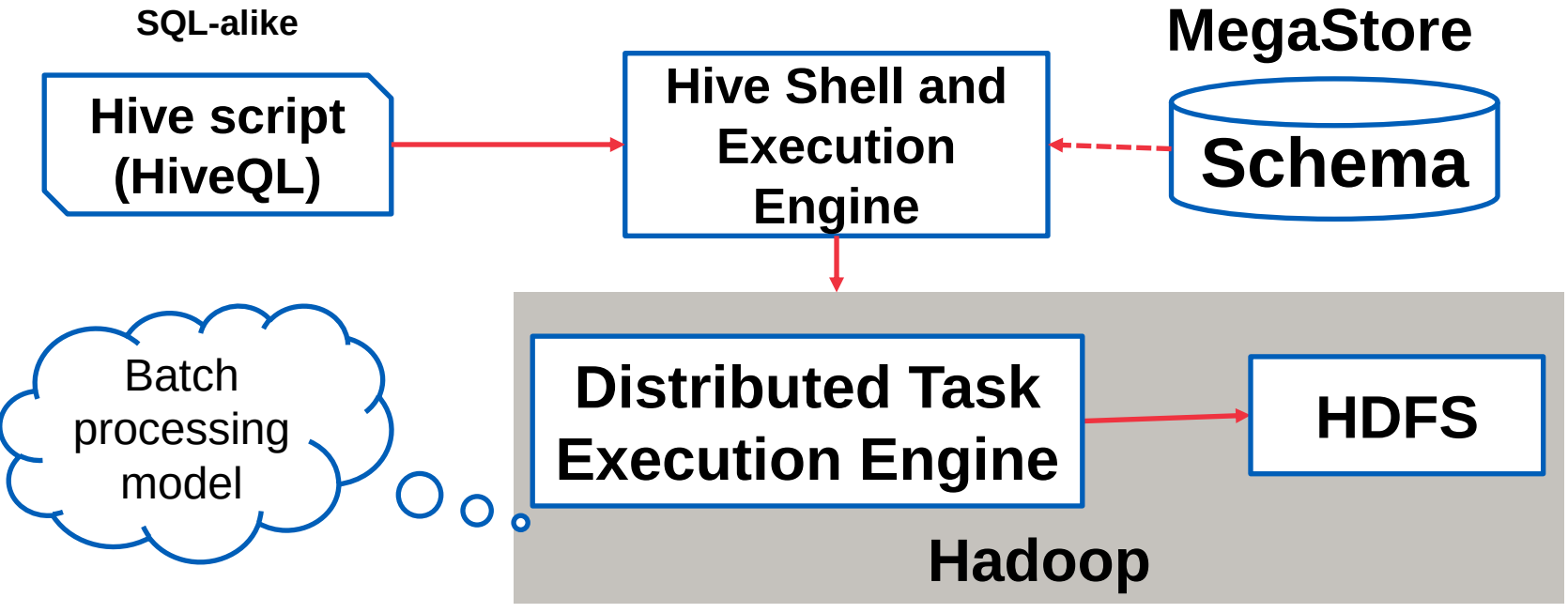
Apache Hive

Apache Hive

- **http://hive.apache.org/, on top of Hadoop**
  - Data warehouse
  - Access data in HDFS or HBase
- **Support access to data via SQL styles**
  - You can do extract/transform/load (ETL), reporting, and data analysis using SQL styles
- **Provide command line tools & JDBC and server for integration**

Reading – Hive in Facebook
https://engineering.fb.com/2009/06/10/web/hive-a-petabyte-scale-data-warehouse-using-hadoop/

High-level data flow language & programs



Hive building blocks

Distributed tasks with MapReduce, Tez (Workflow) or Spark

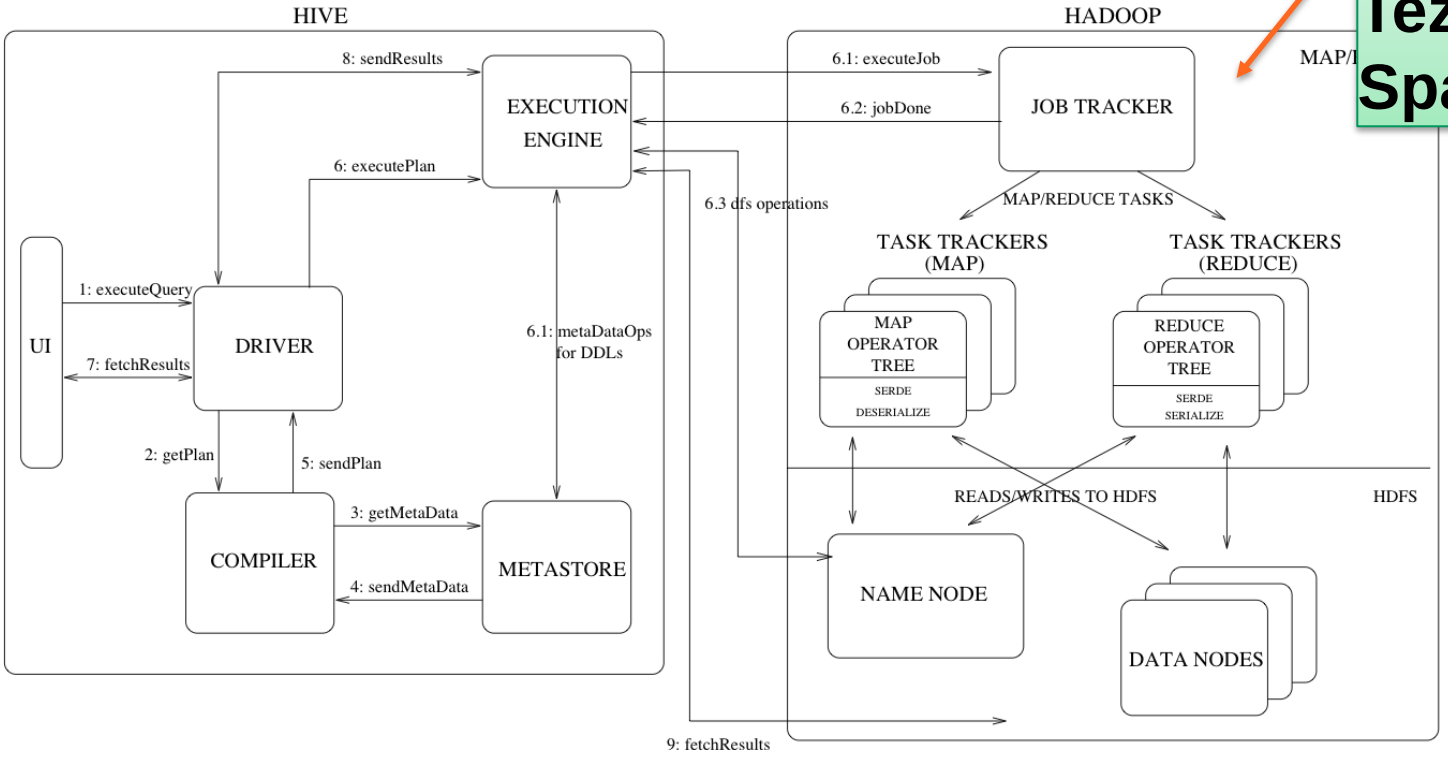


Figure source: https://cwiki.apache.org/confluence/display/Hive/Design

Hive Data Organization

- **Databases**
- **Table**
  - Managed table versus external tables
  - External table: data is referenced so delete only table metadata but not the data)
  - Table is mapped to a directory in HDFS

Example

Large tables lead to performance issues

```
0: jdbc:hive2://localhost:10000> describe taxiinfo;
+-------------------------+------------+----------+
|        col_name         | data_type  | comment  |
+-------------------------+------------+----------+
| vendorid                | int        |          |
| tpep_pickup_datetime    | timestamp  |          |
| tpep_dropoff_datetime   | timestamp  |          |
| passenger_count         | int        |          |
| trip_distance           | float      |          |
| ratecodeid              | int        |          |
| store_and_fwd_flag      | int        |          |
| pulocationid            | string     |          |
| dolocationid            | int        |          |
| payment_type            | int        |          |
| fare_amount             | float      |          |
| extra                   | float      |          |
| mta_tax                 | float      |          |
| tip_amount              | float      |          |
| tolls_amount            | float      |          |
| improvement_surcharge   | float      |          |
| total_amount            | float      |          |
+-------------------------+------------+----------+
```

Hive Data Organization for performance optimization

- **Partition:**
  - partition keys determine how data in Table will be divided
    - *E.g. date or countries*
  - Each partition is stored as a subdirectory
    - *Avoid many sub directories!*
- **Buckets**
  - Avoid large number of small partitions
  - Buckets using a hash function of a column for grouping records into the same bucket, each is a file
  - Bucket columns should be optimized for join/filter operation

Example of partitions

```
CREATE TABLE taxiinfo1 (….)
 PARTITIONED BY  (year int, month int)
…;
```

Indicate partition info

Define partition names

```
LOAD DATA LOCAL INPATH …… INTO TABLE taxiinfo1
PARTITION (year=2019, month=11);
```

```
truong@aaltosea:/opt/hadoop$ bin/hdfs dfs -ls /user/hive/warehouse/taxiinfo1
Found 4 items
drwxr-xr-x   - truong supergroup          0 2021-03-02 22:37 /user/hive/warehouse/taxiinfo1/year=2017
drwxr-xr-x   - truong supergroup          0 2021-03-02 22:37 /user/hive/warehouse/taxiinfo1/year=2018
drwxr-xr-x   - truong supergroup          0 2021-03-02 22:36 /user/hive/warehouse/taxiinfo1/year=2019
drwxr-xr-x   - truong supergroup          0 2021-03-02 22:33 /user/hive/warehouse/taxiinfo1/year=__HIVE_DEFAULT_PA
truong@aaltosea:/opt/hadoop$ bin/hdfs dfs -ls /user/hive/warehouse/taxiinfo1/year=2019
Found 2 items
drwxr-xr-x   - truong supergroup          0 2021-03-02 22:36 /user/hive/warehouse/taxiinfo1/year=2019/month=11
drwxr-xr-x   - truong supergroup          0 2021-03-02 22:36 /user/hive/warehouse/taxiinfo1/year=2019/month=12
```

Example of buckets

```
CREATE TABLE taxiinfo1 (VendorID int, ….)
 CLUSTERED BY (VendorID) INTO 2 BUCKETS
……;
```

Identify bucket column

```
truong@aaltosea:/opt/hadoop$ bin/hdfs dfs -ls /user/hive/warehouse/taxiinfo2
Found 2 items
-rw-r--r--   1 truong supergroup      66784 2021-03-02 22:54 /user/hive/warehouse/taxiinfo2/000000_0
-rw-r--r--   1 truong supergroup      31339 2021-03-02 22:54 /user/hive/warehouse/taxiinfo2/000001_0
```

Combine partitions with buckets

```
truong@aaltosea:/opt/hadoop$ bin/hdfs dfs -ls /user/hive/warehouse/taxiinfo3/year=2019/month=11
Found 2 items
-rw-r--r--   1 truong supergroup      66784 2021-03-02 23:04 /user/hive/warehouse/taxiinfo3/year=2019/month=11/000000_0
-rw-r--r--   1 truong supergroup      31339 2021-03-02 23:04 /user/hive/warehouse/taxiinfo3/year=2019/month=11/000001_0
```

ACID

- **Full ACID support**
  - Row-level transaction
- **Locks are used for data isolation**
  - Shared lock: for concurrent read of tables/partitions
  - Exclusive lock: for modifying table/partition

Revisit your personal techradar

- **Hadoop software ecosystem is very powerful**
  - Many applications and use cases have been developed
- **Managed Hadoop ecosystem services by cloud providers**
  - Try to look at Azure HDInsight, Google Dataproc, and Amazon EMR
- **Understand the combination of data management with data processing techniques in the same system with Hadoop that simplify your big data tasks**

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io