Aalto University
School of Science

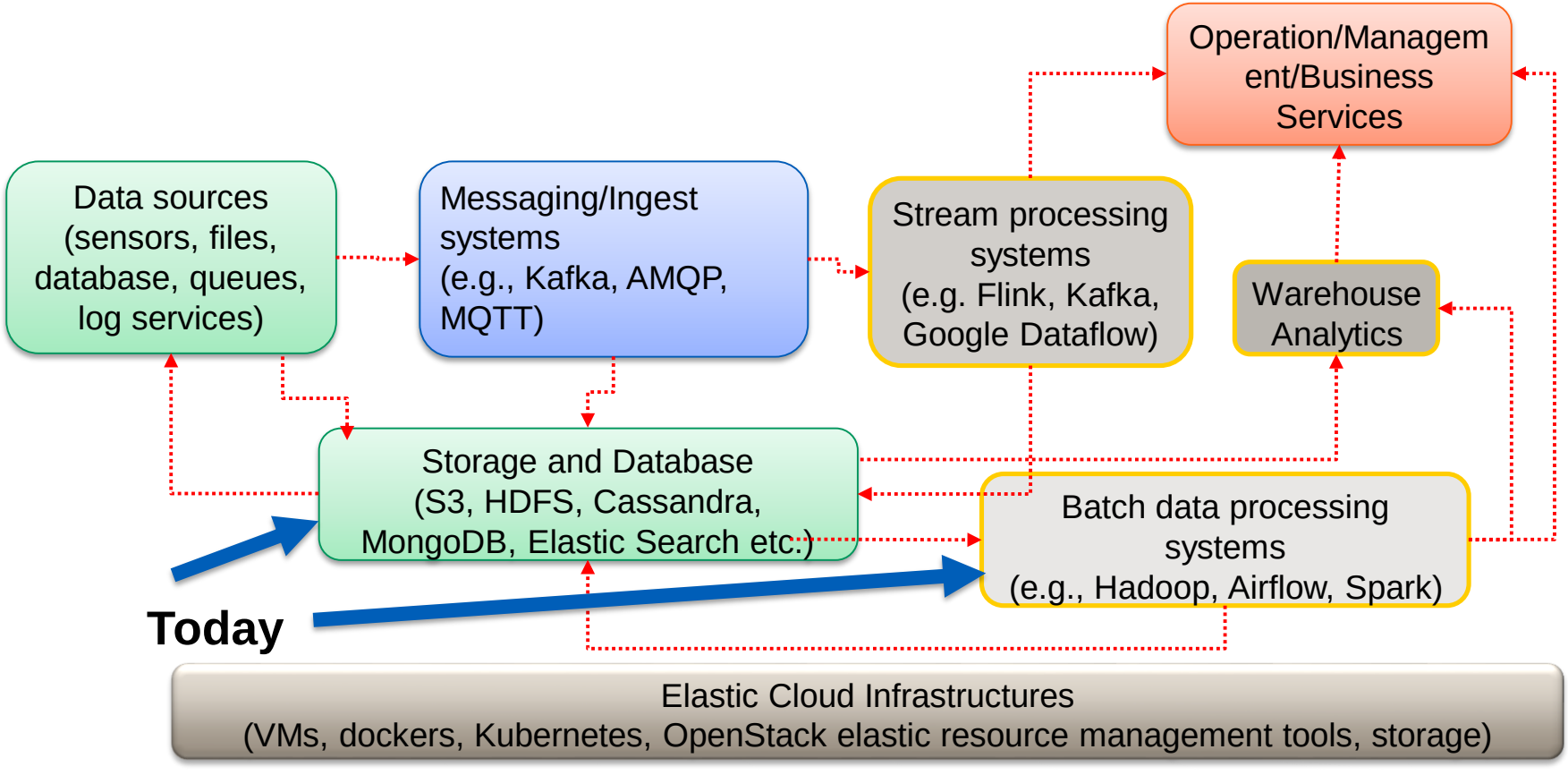
Hadoop and its Big Data Ecosystems

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

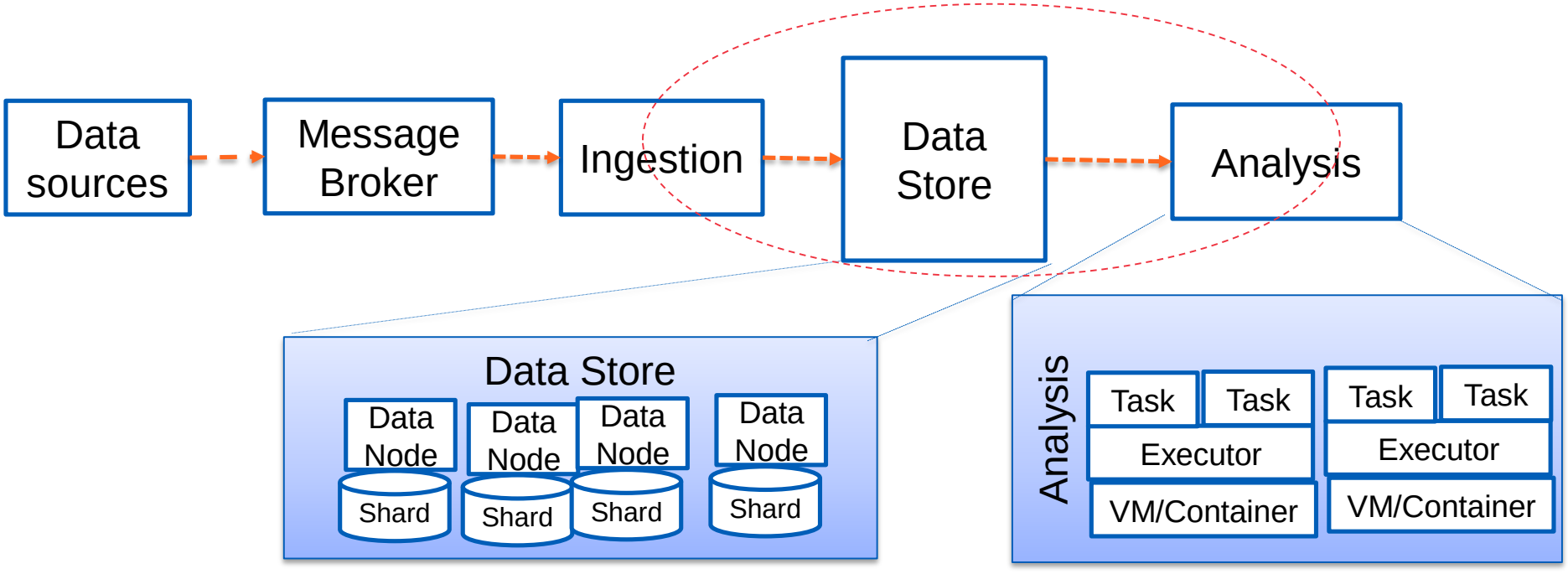
Schedule

- **Hadoop**
  - Hadoop file systems
  - Yarn
- **Native Hadoop-based systems**
- **Integration patterns**

Big data at large-scale



Consider again our big data pipelines



Recall: “Copy and Process”

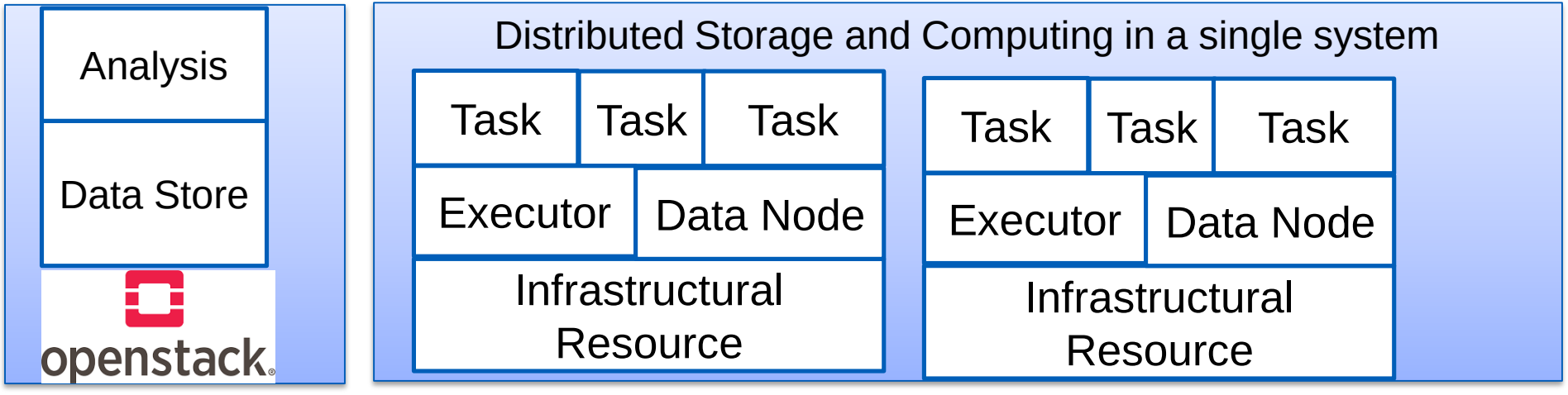
Client libraries are used to move data from storages and databases to processing places

Examples:

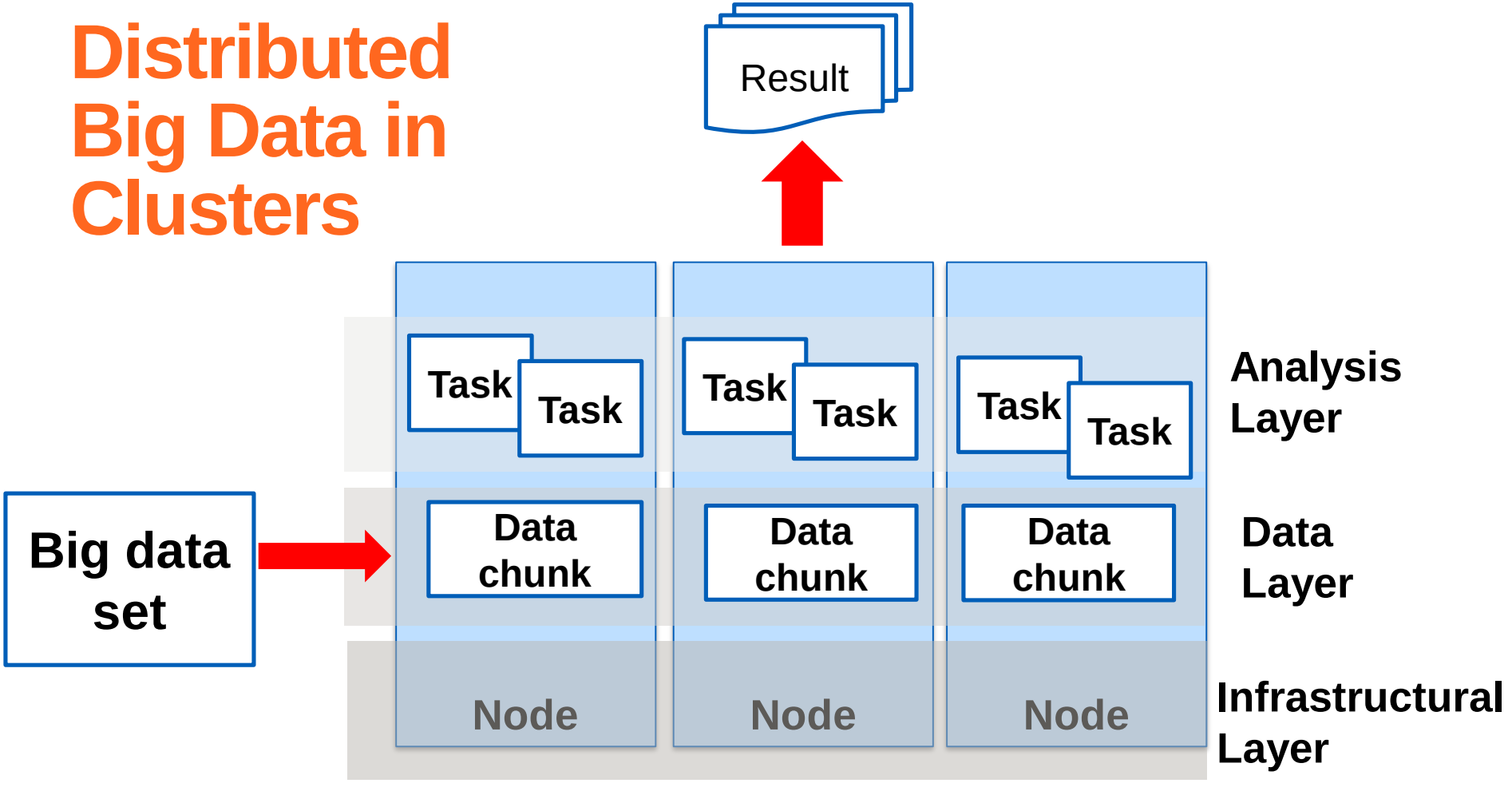
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from cassandra.cluster import Cluster

cluster = Cluster(contact_points=hosts, port=9042,auth_provider=auth_provider)
session = cluster.connect(“tutorial12345”)
sql_query = "SELECT * FROM tutorial12345.bird1234;“
df = pd.DataFrame()
rows= session.execute(sql_query)
df = rows._current_rows
print(df)
```

Consider again our pipelines

Can we combine storage and analytics some how in the same system/framework?



Distributed Big Data in Clusters



Benefit

- **Moving data is much more expensive than moving computation**
- **Save shared resources**
- **Solutions are suitable for different types of customers/clients and different hardware**

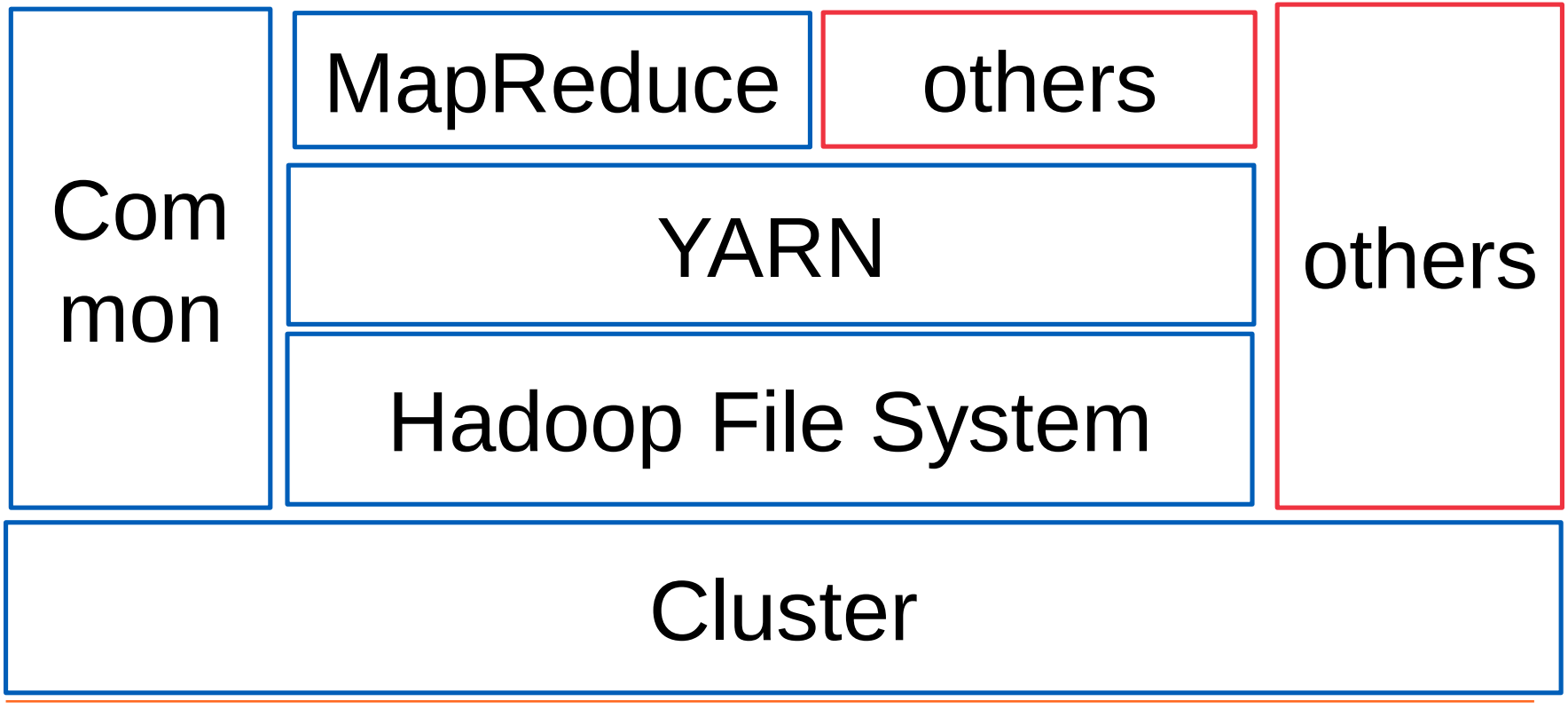
Benefit - Variety

- **Multiple data formats**
- **In big data we have a lot of files**
  - Should we always transform them into a single format?
- **Processing files directly is a benefit!**

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

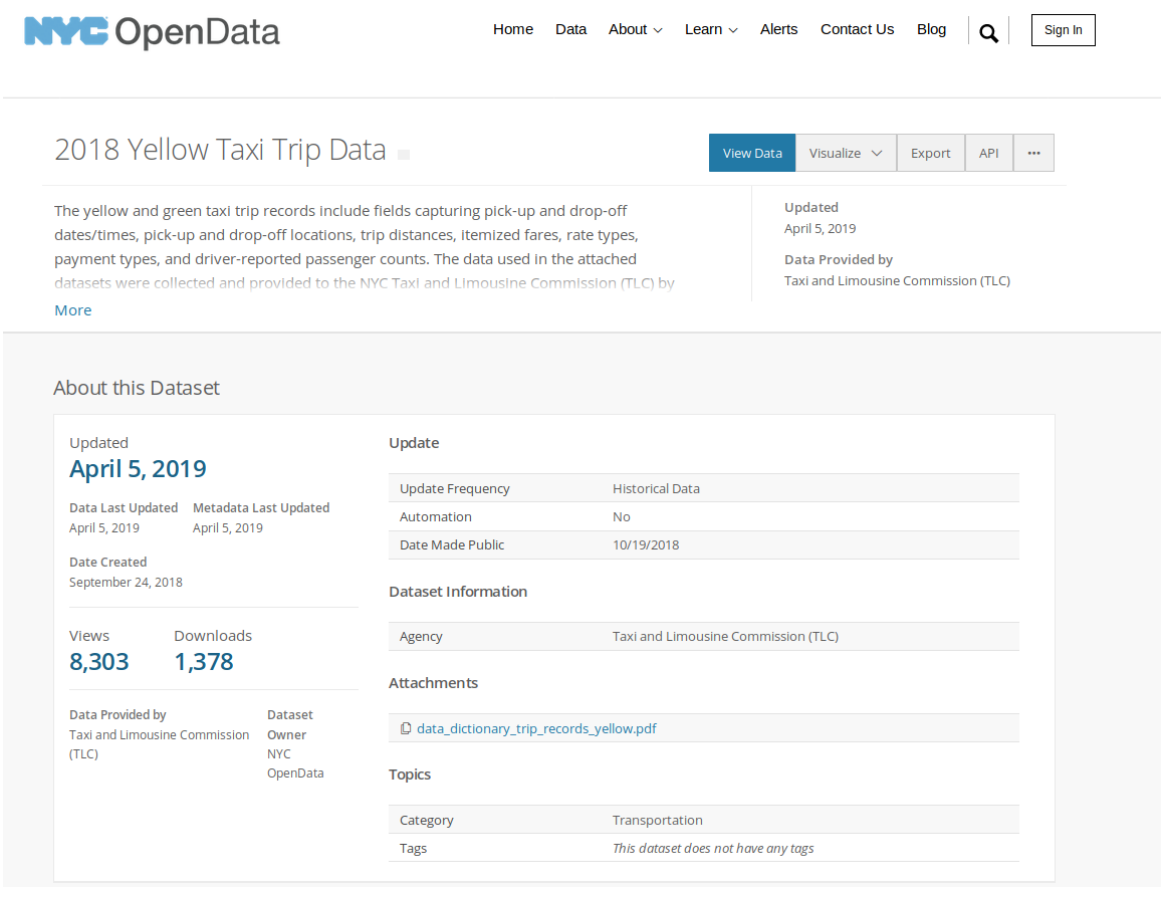
- **HDFS as a distributed file system**
  - For managing data
- **YARN as a resource management system**
  - For executing and managing tasks
- **MapReduce as one programming model**
  - For MapReduce applications
- **Coordination (ZooKeeper)**
  - for fault tolerance and metadata

Hadoop File System (HDFS)

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

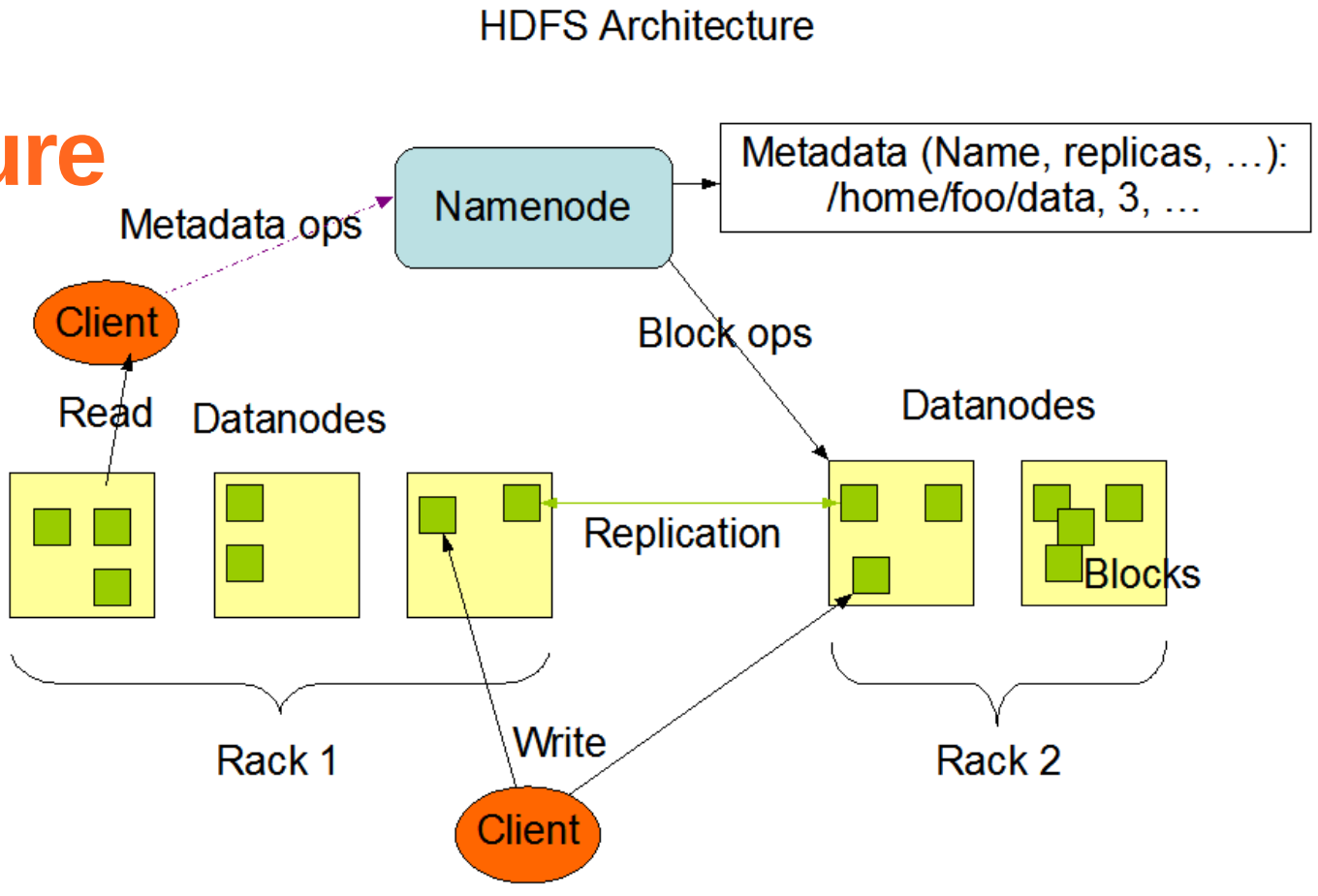
Example

nytaxi2019.csv:
10428263736 bytes



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

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**
  - 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

Quick Test

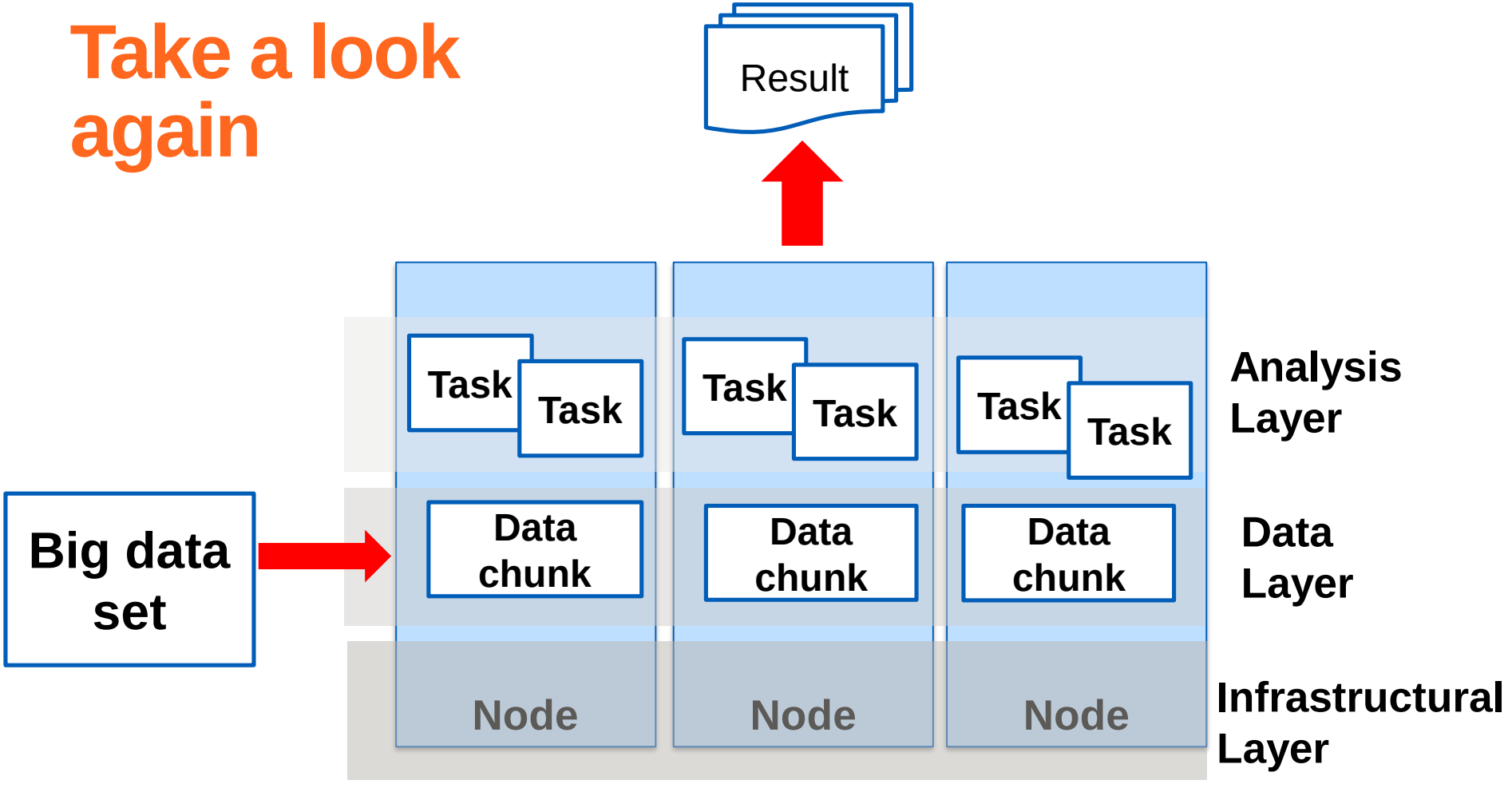
https://tinyurl.com/y2xjebrs

```
Status: HEALTHY
 Total size:    10428263736 B
 Total dirs:    0
 Total files:   1
 Total symlinks:                0
 Total blocks (validated):      78 (avg. block size 133695688 B)
 Minimally replicated blocks:   78 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    2
 Average block replication:     2.0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          4
 Number of racks:               1
```

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?

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?

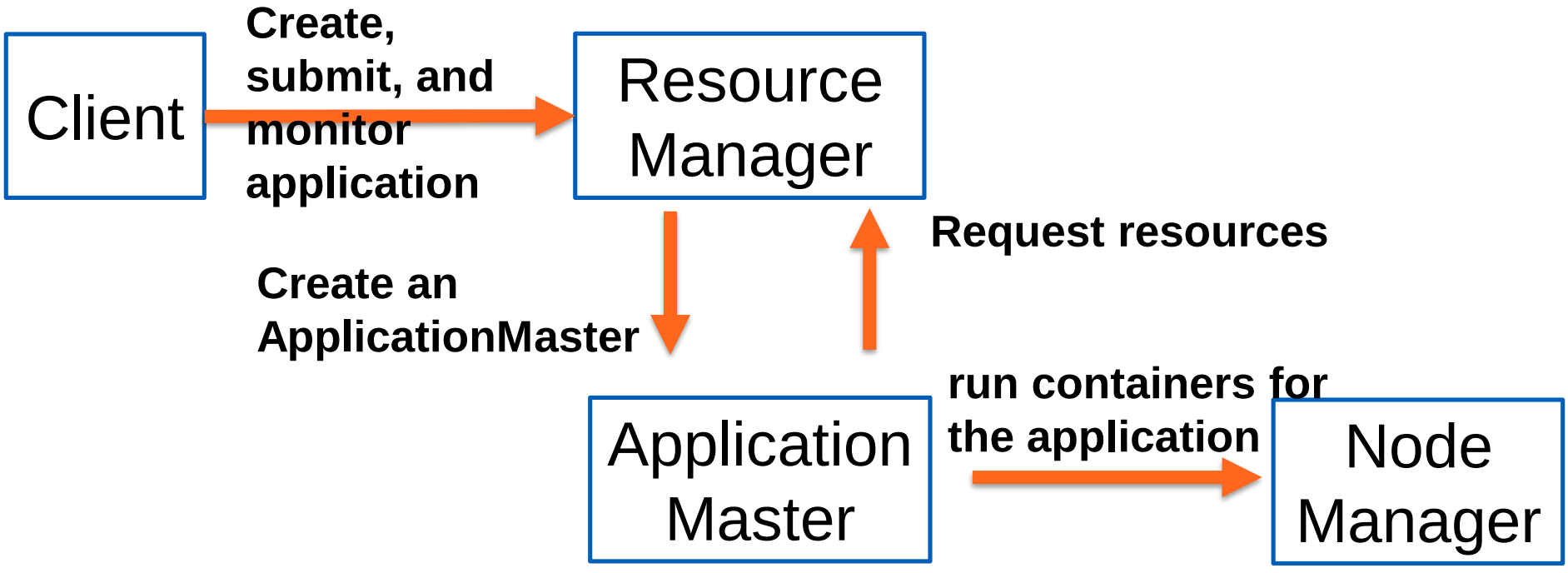
YARN (Yet Another Resource Negotiator)

- **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"**
  - It is not like (Docker) container
- **Multi-tenancy support**

YARN Components

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

YARN basic model



YARN Architecture

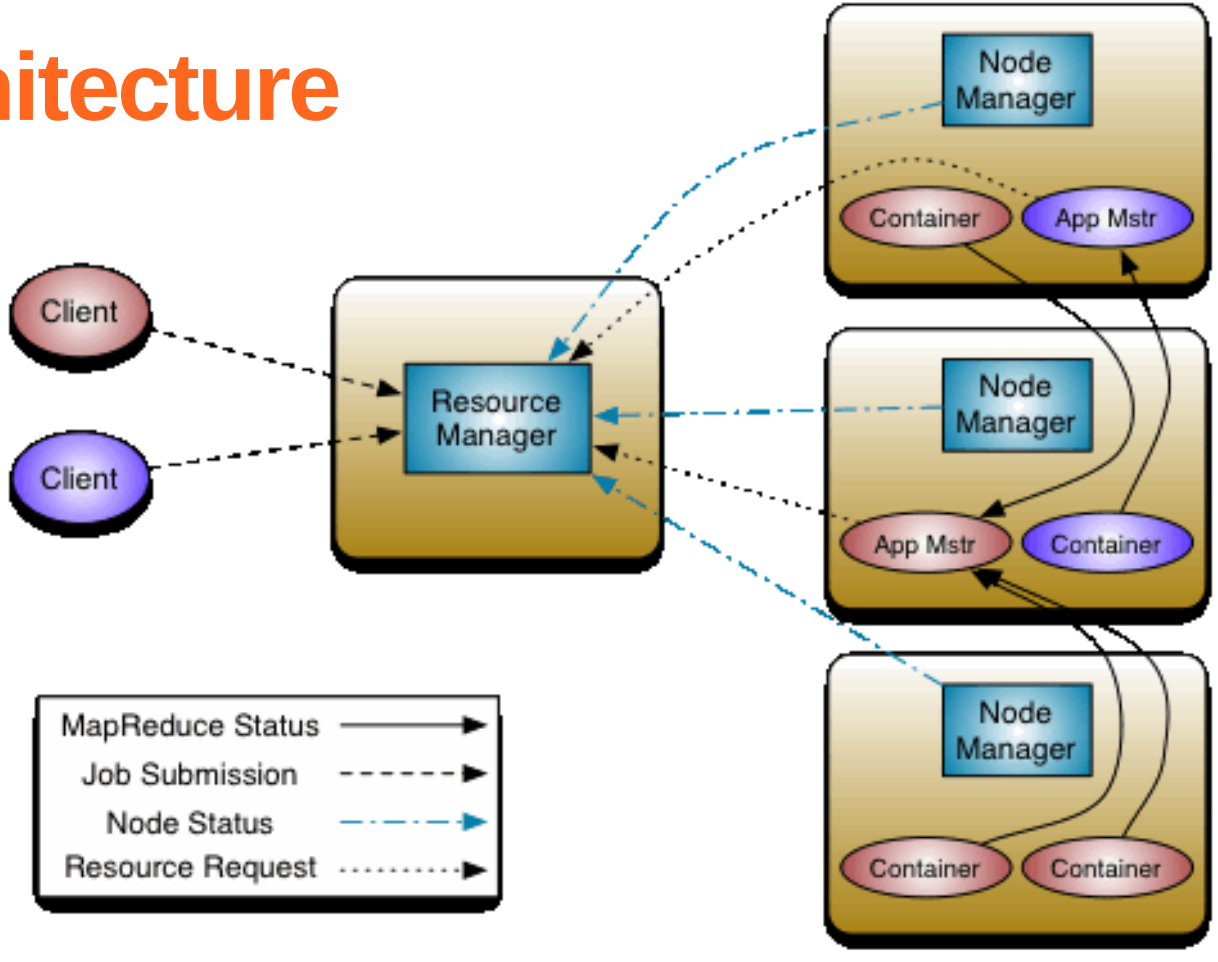


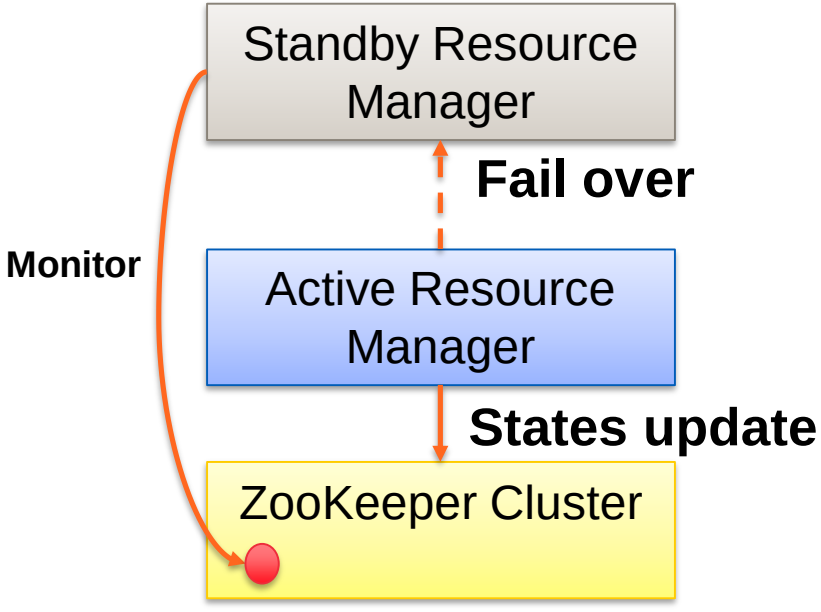
Figure source: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

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*

What would be the benefit of coupling HDFS and YARN?

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**

Data Model

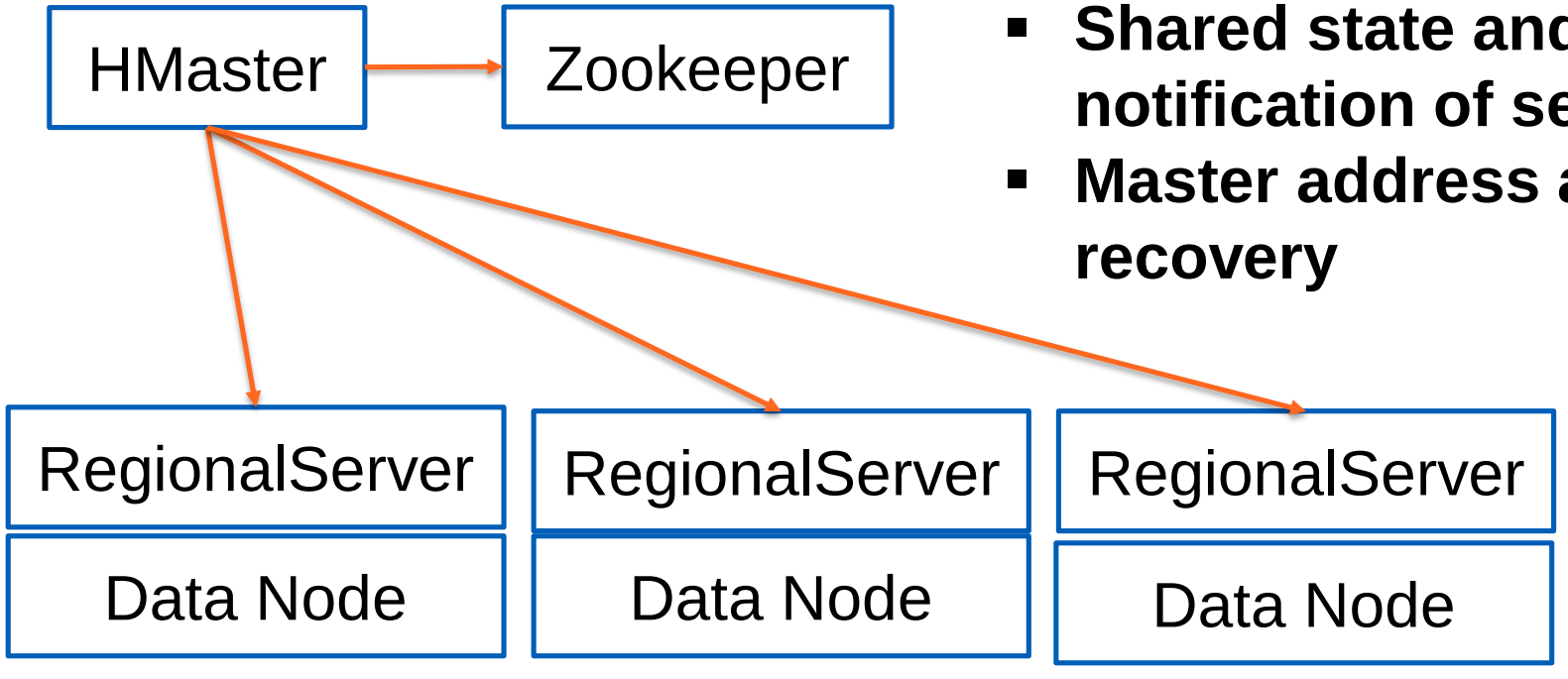
- **Column family**
  - Follow the BigTable (not like Cassandra)
- **Example with families: birdinfo, songinfo, location**

```
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
```

Data Model

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

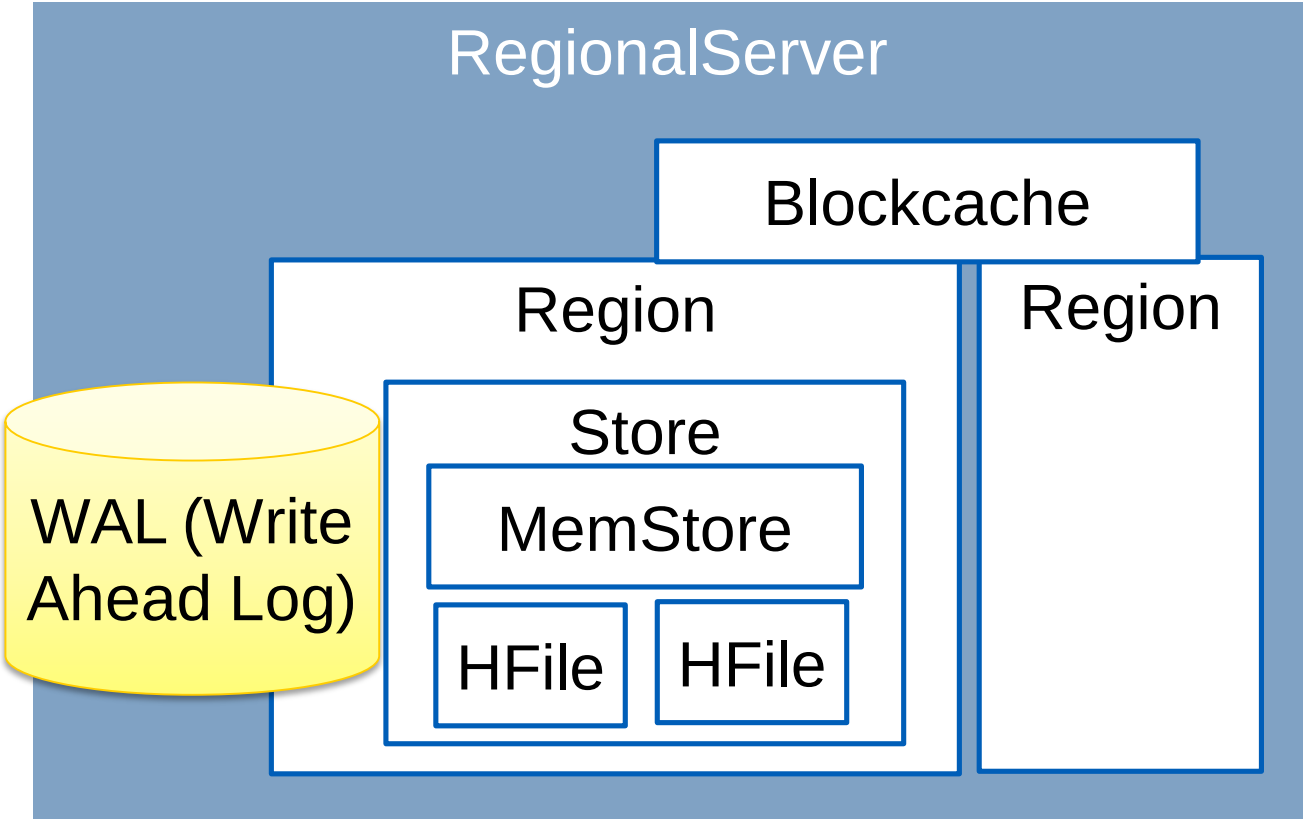
Hbase Architecture



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)

A short HBase walkaround

- **Install a single node version**
  - In a single laptop
- **http://localhost:16010/master-status**

Remember Cassandra?

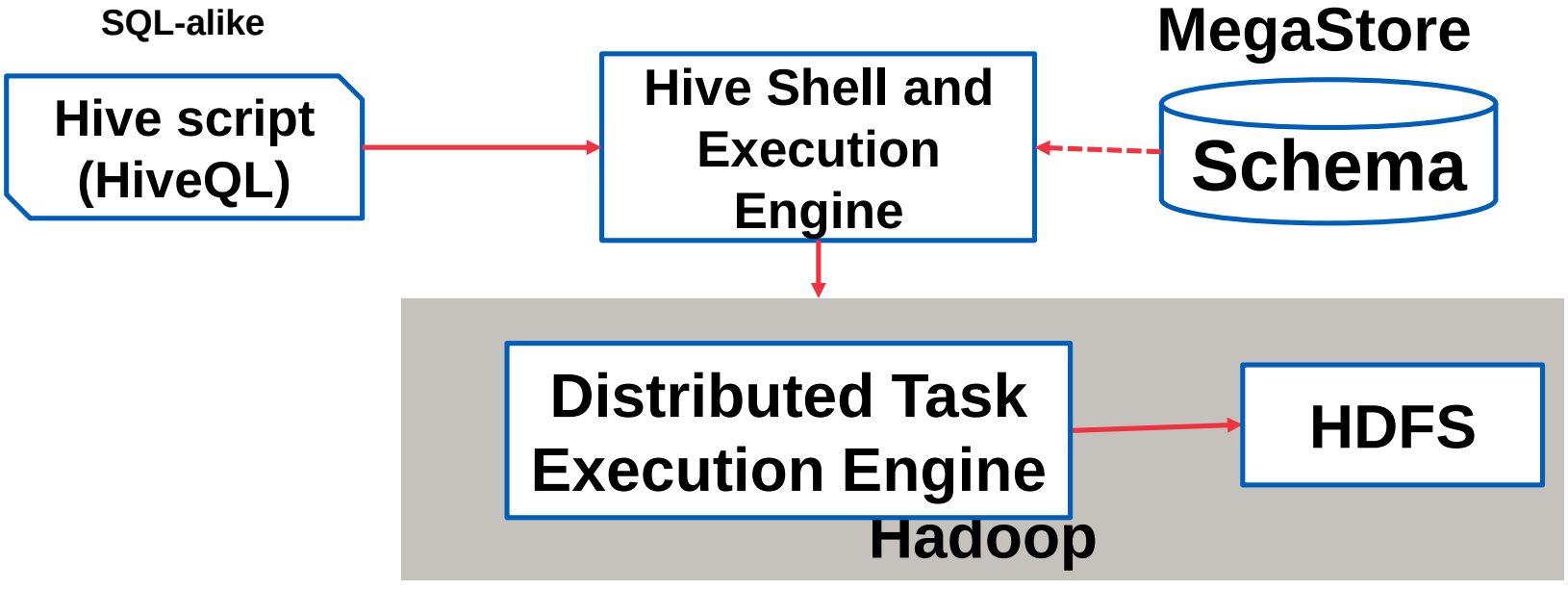
Any differences between HBase and Cassandra that you see?

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**

High-level data flow language & programs



Hive building blocks

Distributed tasks with MapReduce, Tez (Workflow) or Spark

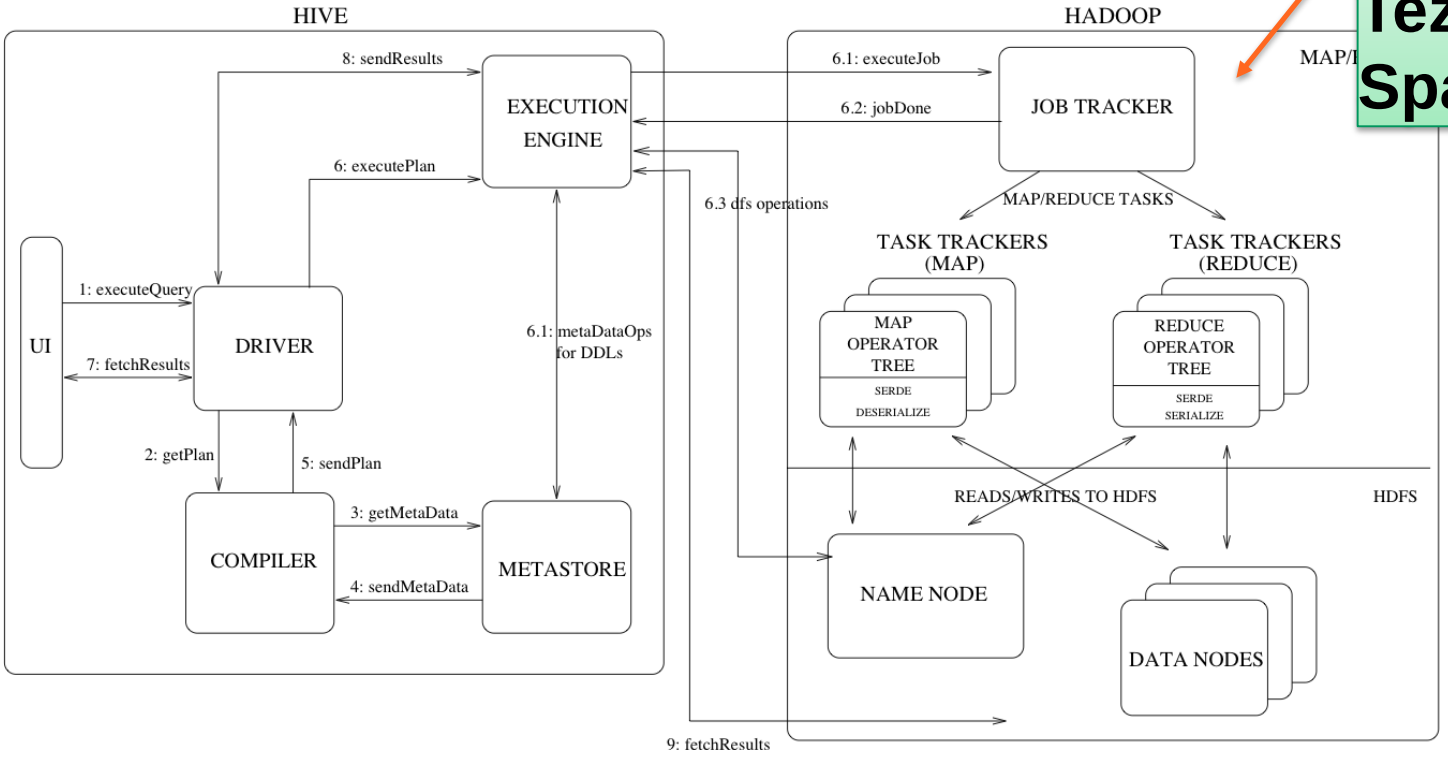


Figure source: https://cwiki.apache.org/confluence/display/Hive/Design

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

```
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

- **Partition:**
  - partition keys determine how data in Table will be
    - *E.g. date or countries*
  - Each partition is stored as a subdirectory
- **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

ACID

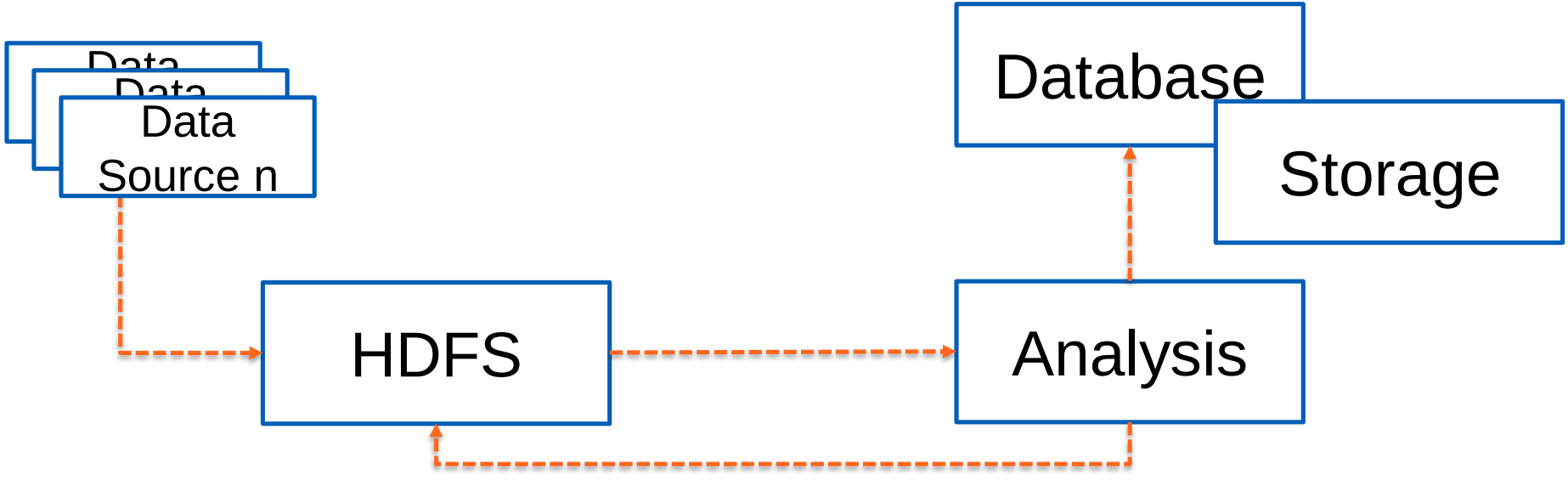
- **Full ACID support**
  - Row-level transaction
- **Locks are used for data isolation**
  - Shared lock: for concurrent read of tables/partitions
  - Exclusive lock: for modifying table/partition

Use Hadoop for complex data management and analytics

Integration models

- **Using Hadoop for developing large-scale data analysis**
  - Apache Spark
  - More on other lectures (#ingestion)
- **Using Hadoop as components in a big data system**
  - Hadoop can be come data lake/data store
  - Many ETL tasks

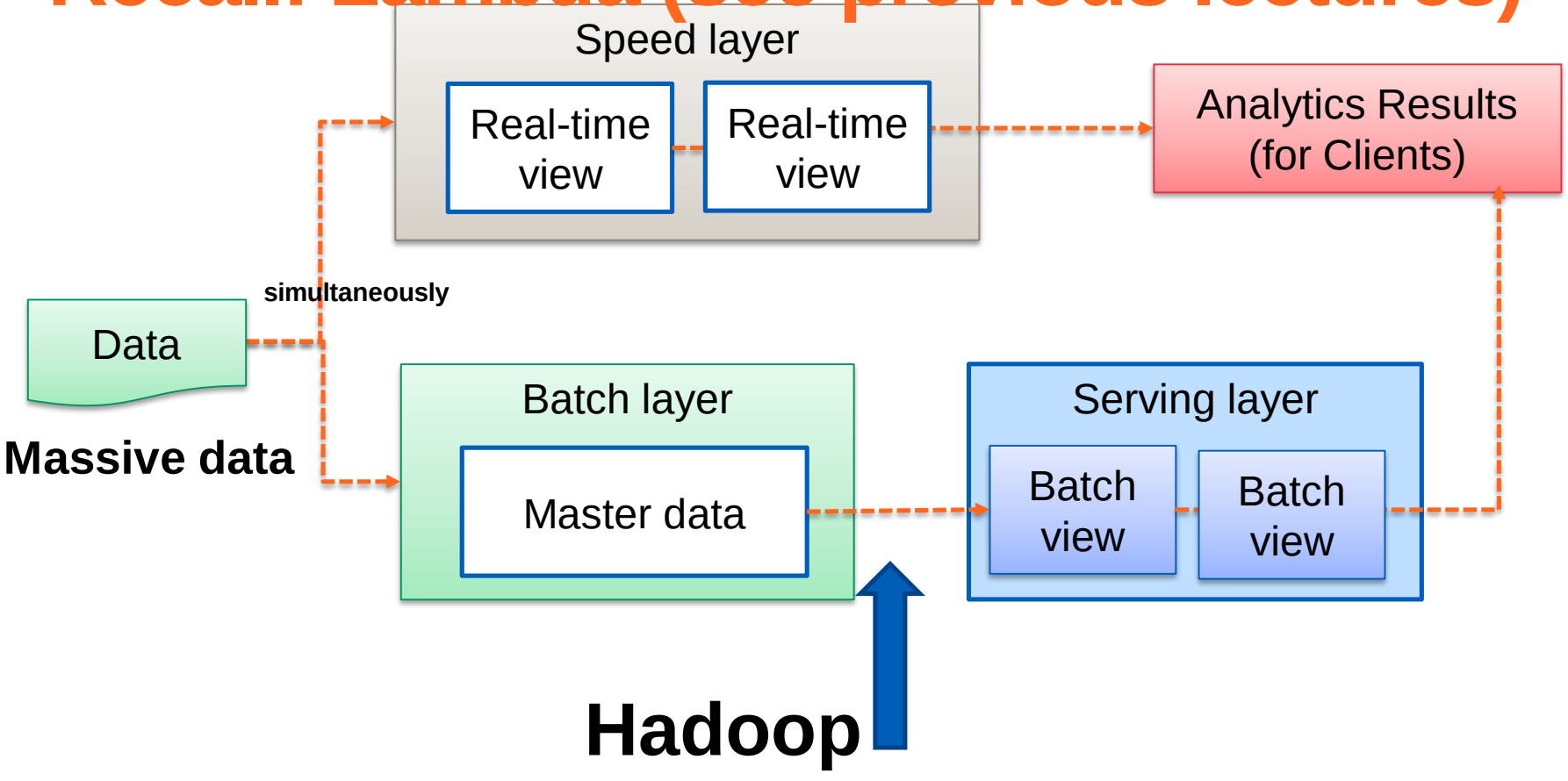
Using Hadoop as part of ETL



Example: integration with Spark Streaming

Source: http://spark.apache.org/docs/latest/streaming-programming-guide.html

Recall: Lambda (see previous lectures)



Example of MongoDB integration pattern

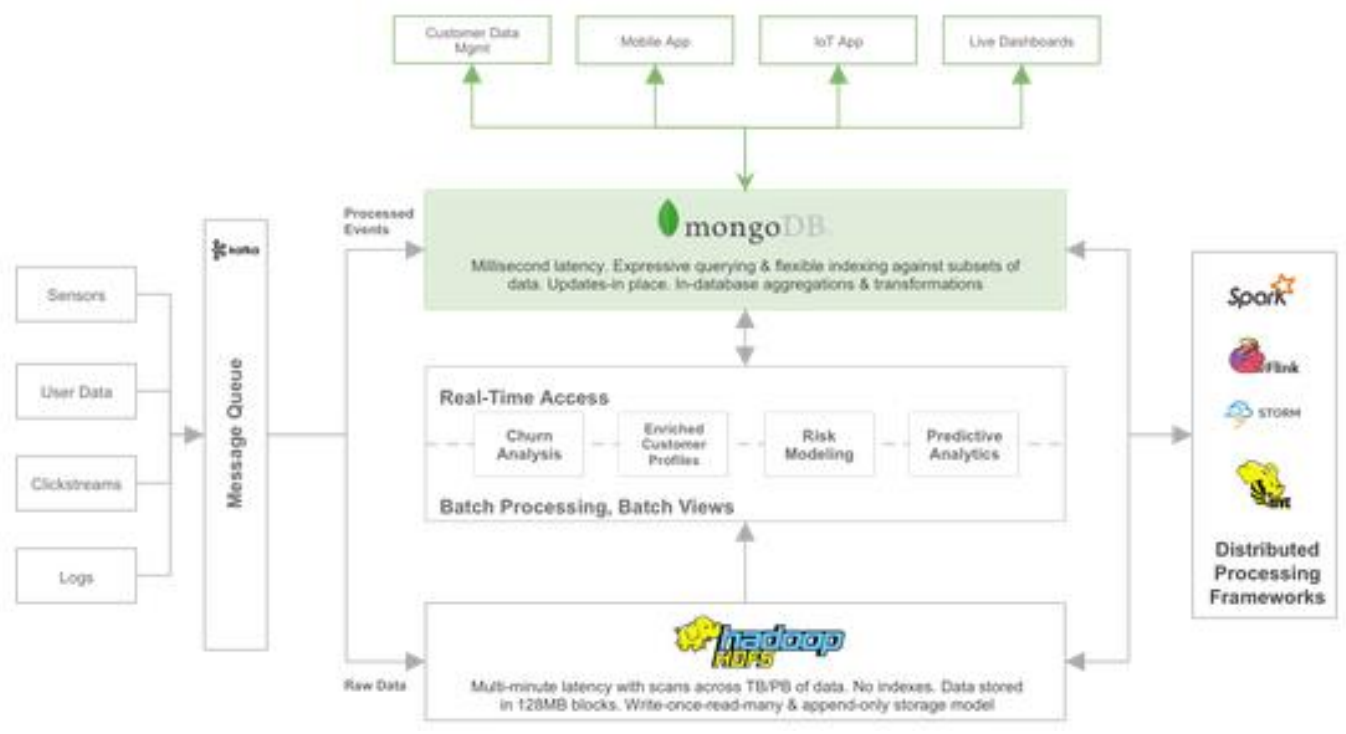


Figure source: https://www.mongodb.com/hadoop-and-mongodb

Summary

- **Hadoop software ecosystem is very powerful**
  - Many applications and use cases have been developed
  - A lot of tools built atop Hadoop
  - Hadoop has many connectors to many other systems
- **Combining data management with data processing techniques in the same system is the story of Hadoop**
- **Available for different types of customers**

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io