Managing HDFS Cluster

technologies that make our cluster work

Bernard Lee Kok Bang

Hadoop Yarn

Yet Another Resource Negotiator

Bernard Lee Kok Bang

HDFS - data storage manager - distribute our big data into different partition or differect blocks of nodes

YARN -> compute resource manager

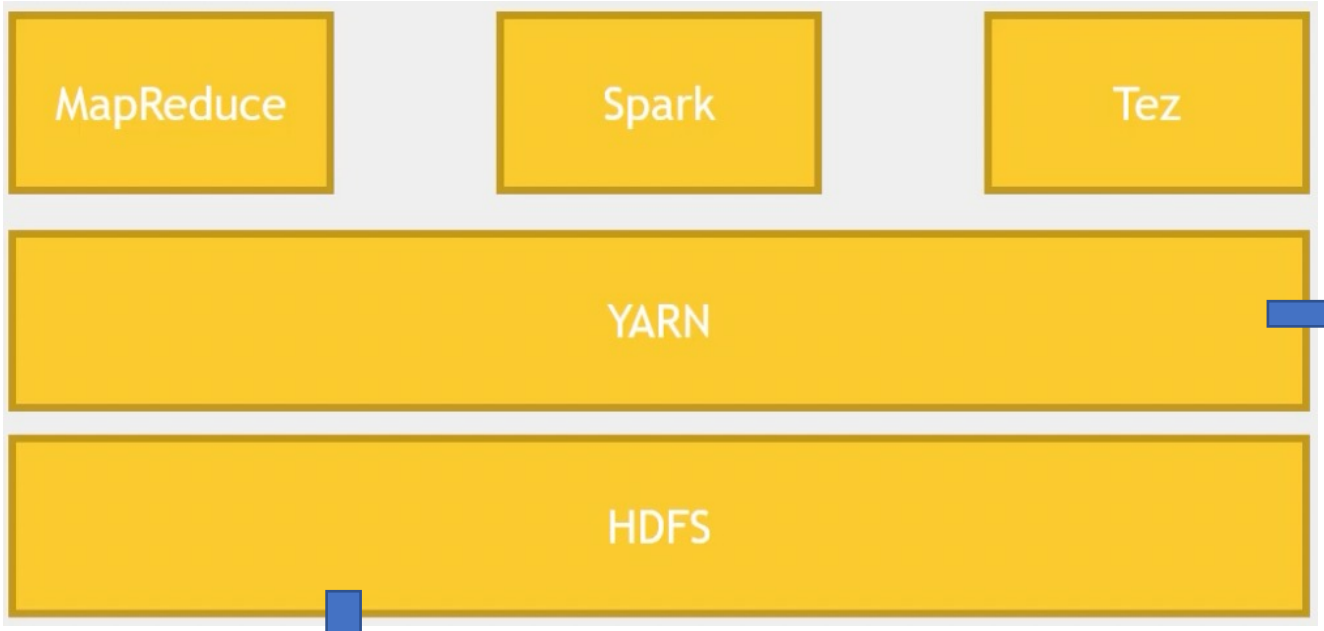
- Fundamental piece of Hadoop version 2
- Manages all the compute resources on our Hadoop cluster
- Sit under the hood [running in the background], we don't really think about it ☺
- Doing complicated stuff, but we don't have to worry about the details

What is YARN?

- Yet Another Resource Negotiator
  - *Introduced in Hadoop 2*
  - *In Hadoop 1, MapReduce ran mappers, reducers, and also resource negotiation (monolithic in nature)*
  - *Separated the problem of managing resources on the cluster from MapReduce*
  - *Enabled development of MapReduce alternatives (Spark, Tez) built on top of Yarn*
  - *turns out Spark and Tez can outperform MapReduce through directed acyclic graph approach*
- It's just there, under the hood, managing the usage of our cluster

Where YARN fits in architecturally?

Clients

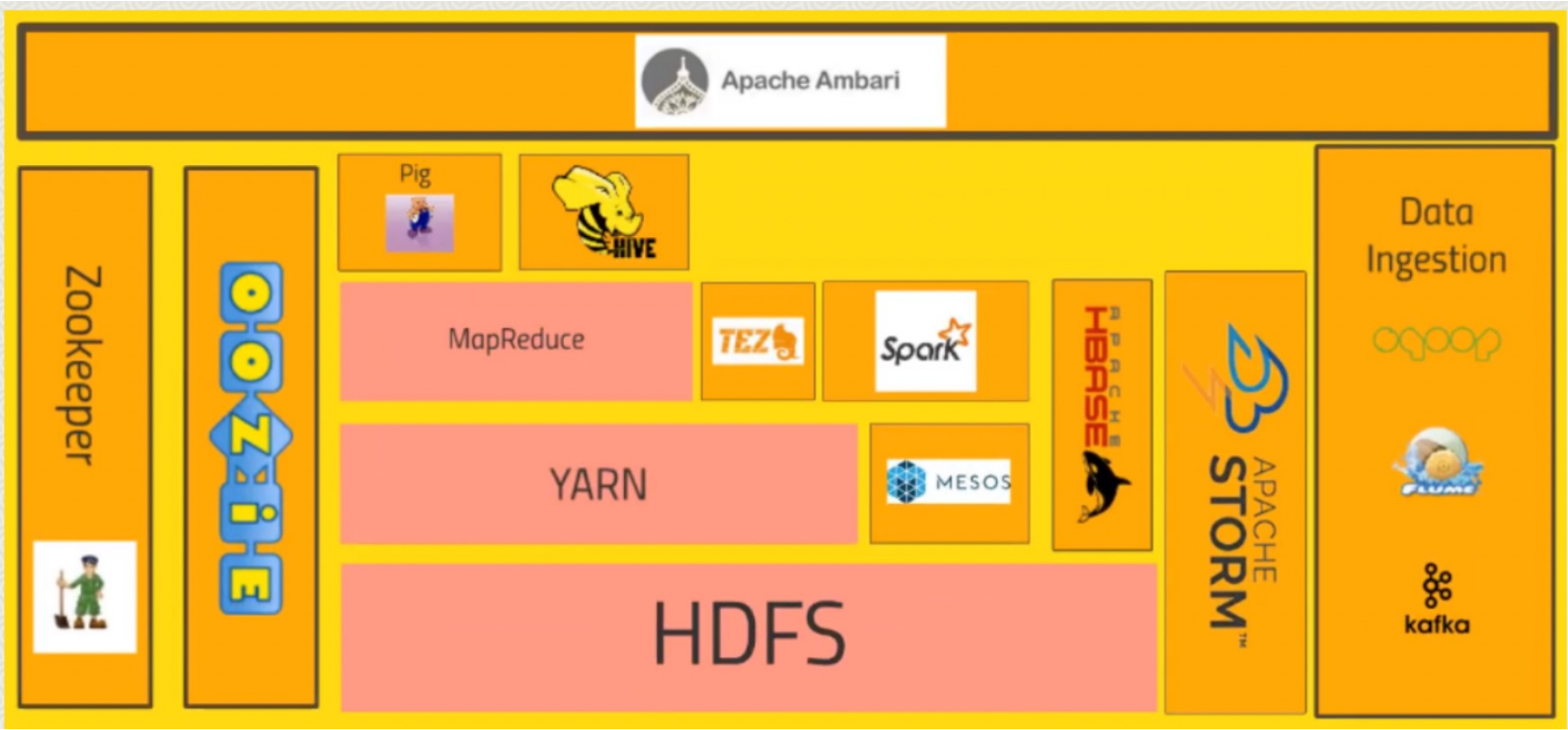


YARN Applications

Cluster Compute Layer
- maintain data locality
- try to align specific jobs on the same actual physical host as much as possible

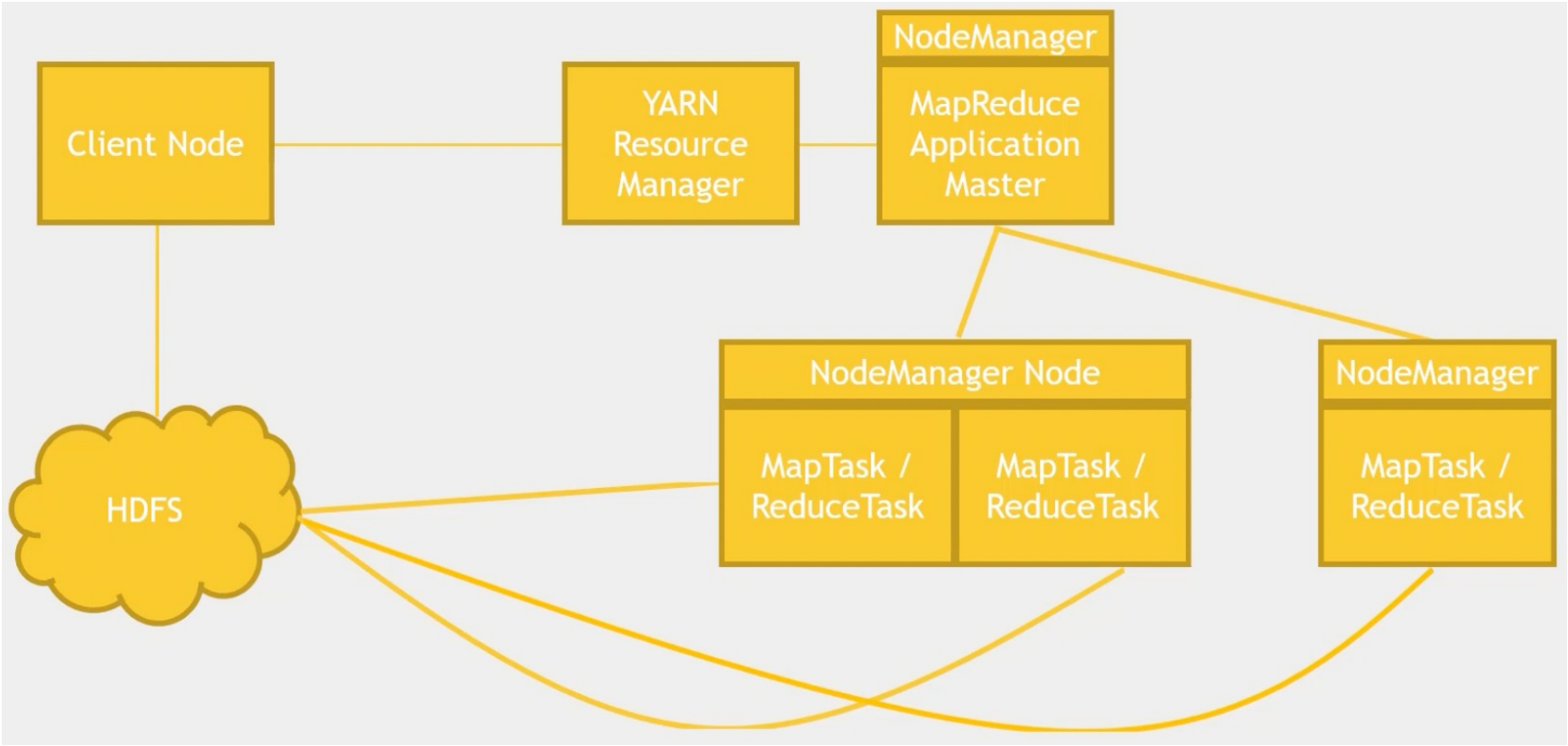
Cluster Storage Layer
- a system that allows us to spread out the storage of big data across the cluster, by breaking it up into blocks that are replicated across various nodes in the cluster

Core Hadoop Ecosystem

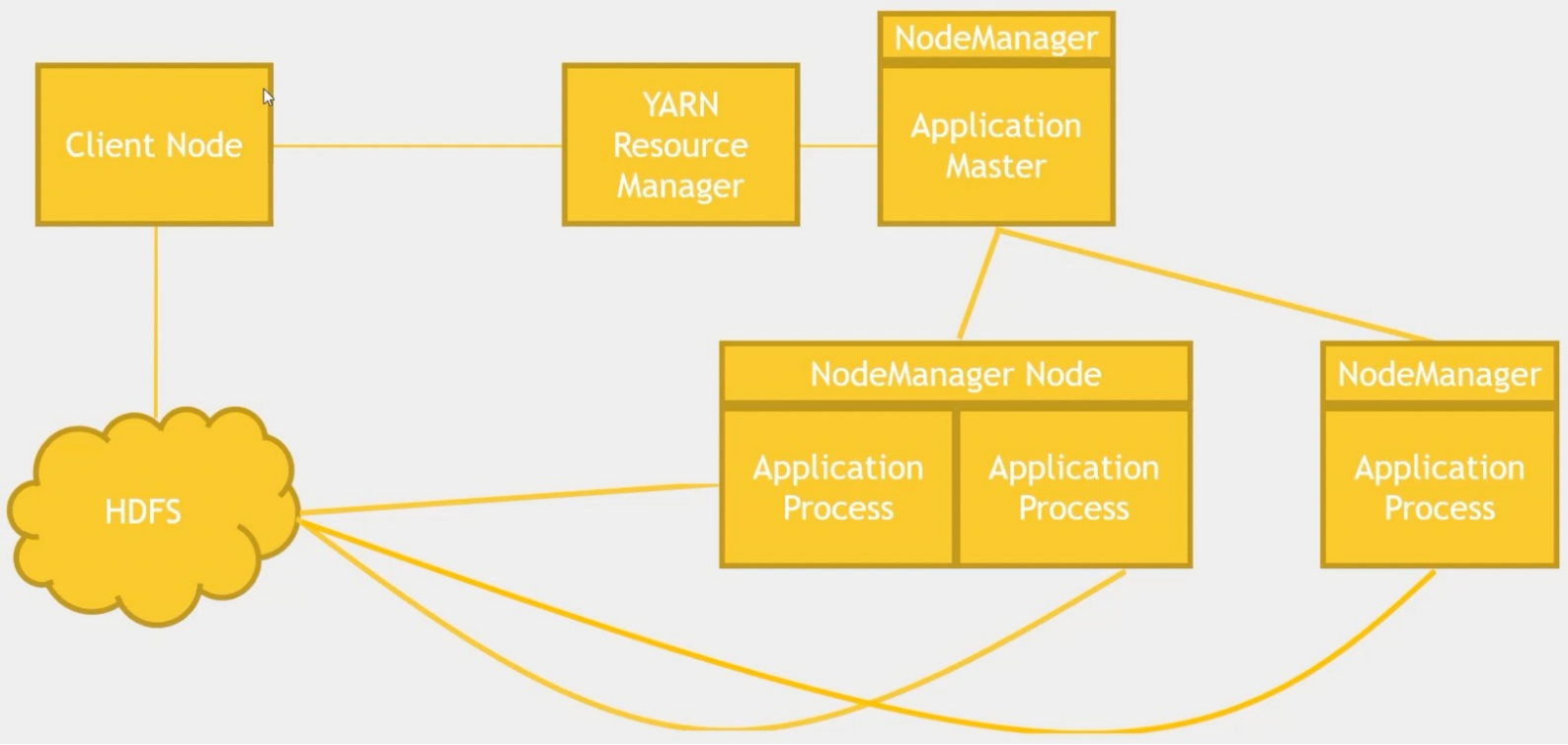


https://bit.ly/3tWMW34

MapReduce framework



YARN Framework



YARN - minimize data getting shuffled around the network of clusters
- trying to optimize the usage of the cluster in terms of CPU cycles

How YARN works

- Our application talks to the Resource Manager to distribute the work to the cluster
- We can specify data locality - which HDFS block(s) we want to access?
  - YARN will try to get the process on the same node that has the HDFS blocks
- We can also specify different scheduling options for applications
  - can run more than one application at once on the cluster
  - FIFO, Capacity, and Fair schedulers
    - FIFO runs jobs in sequence, first in first out
    - Capacity may run jobs in parallel if there's enough spare capacity
    - Fair may cut into a larger running job if we want to squeeze in a small one

Want to practice “using YARN”?

- Well, we have been using YARN the whole time in this course!!!
- We did that with all the jobs that execute MapReduce, Spark, Hive, Pig….
- We just need to know it’s there, under the hood, managing the cluster’s resources for us!!!

APACHE TEZ

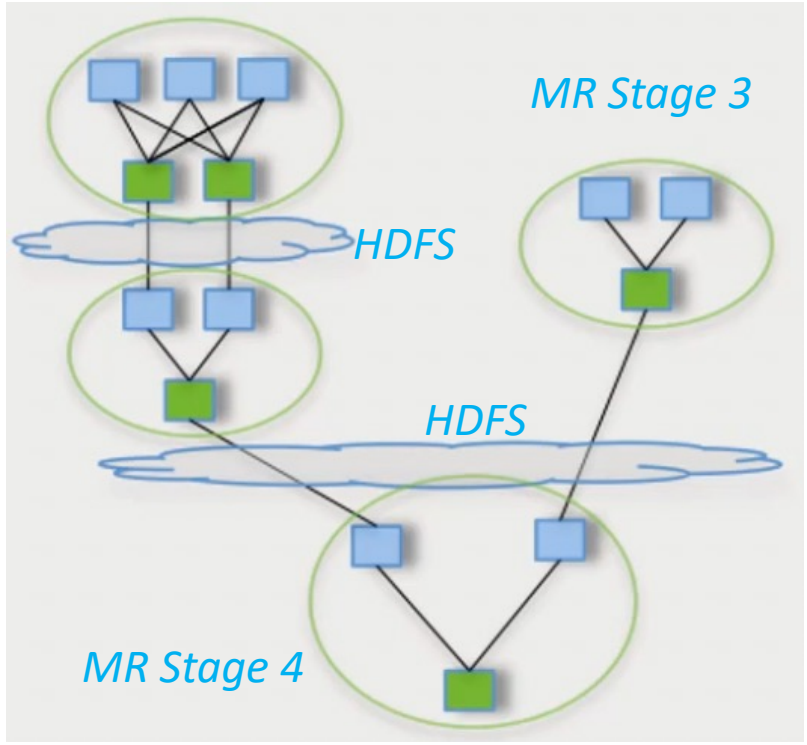
Directed Acyclic Graph Framework - make applications run faster!!!

Bernard Lee Kok Bang

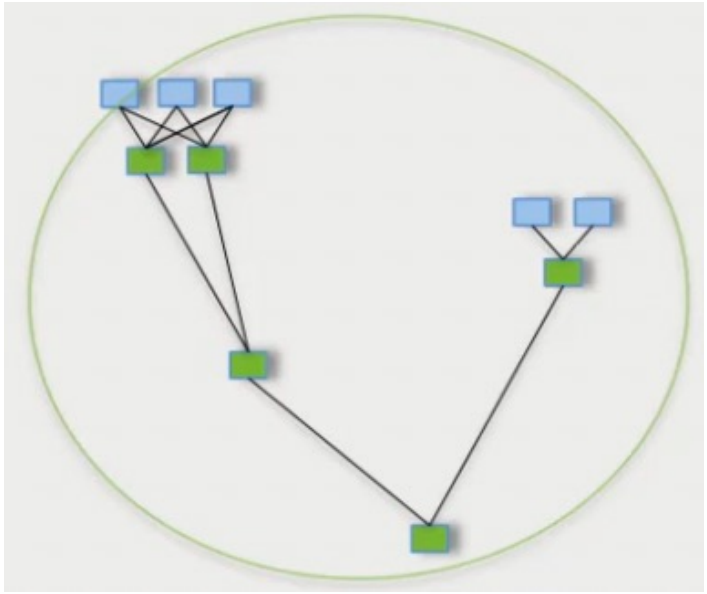
What is TEZ?

- Another bit of infrastructure we can use under the hood
  - *Makes Hive, Pig, or MapReduce jobs faster!*
  - *It's an application framework that clients (Hive & Pig) can code against as a replacement for MapReduce*
- Constructs Directed Acyclic Graphs (DAGs) for more efficient processing of distributed jobs
  - *Relies on a more holistic view of our job; eliminate unnecessary steps and dependencies [runs jobs in parallel]* waits until the very last moment
- Optimize physical data flow and resource usage

Directed Acyclic Graphs



Pig/Hive - MR



Pig/Hive - TEZ

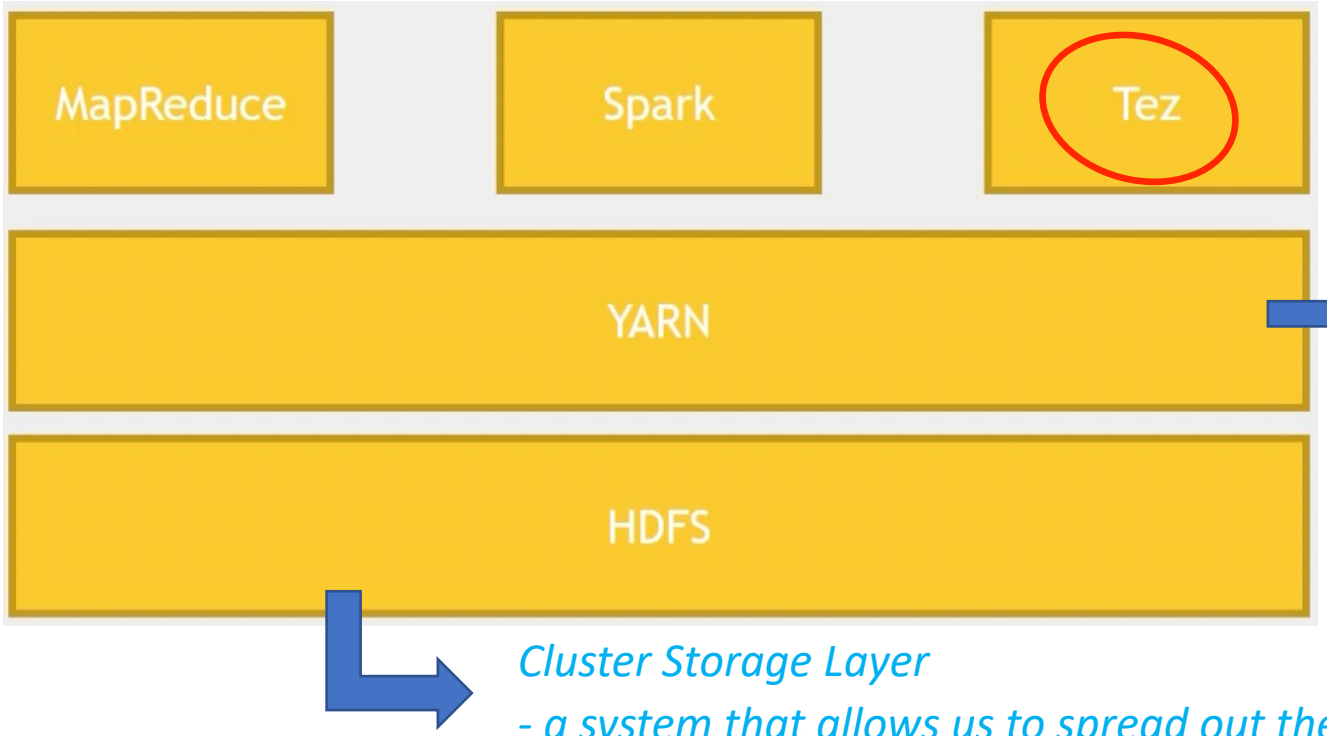
One sequence of dependencies

DAG - eliminate replicated write barrier between successive computations

- *eliminate job launches*
- *eliminate extra stages of MR*

Where TEZ fits in architecturally?

Pig, Hive



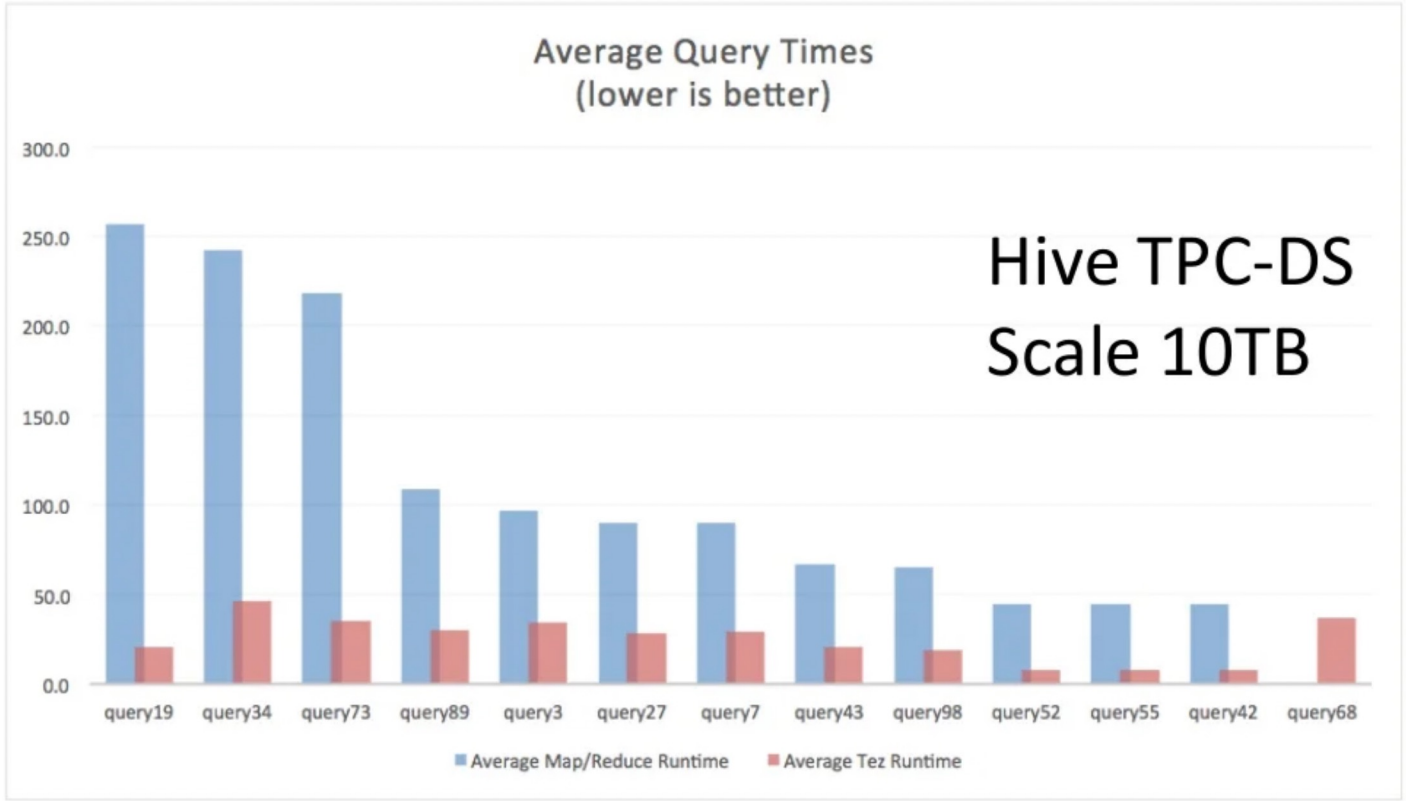
YARN Applications

Cluster Compute Layer
- maintain data locality
- try to align specific jobs on the same actual physical host as much as possible

Cluster Storage Layer
- a system that allows us to spread out the storage of big data across the cluster, by breaking it up into blocks that are replicated across various nodes in the cluster

We Just need to tell Hive / Pig to use TEZ

- Probably **already run TEZ by default** Hive by default is using TEZ
- It really is running a lot faster!!

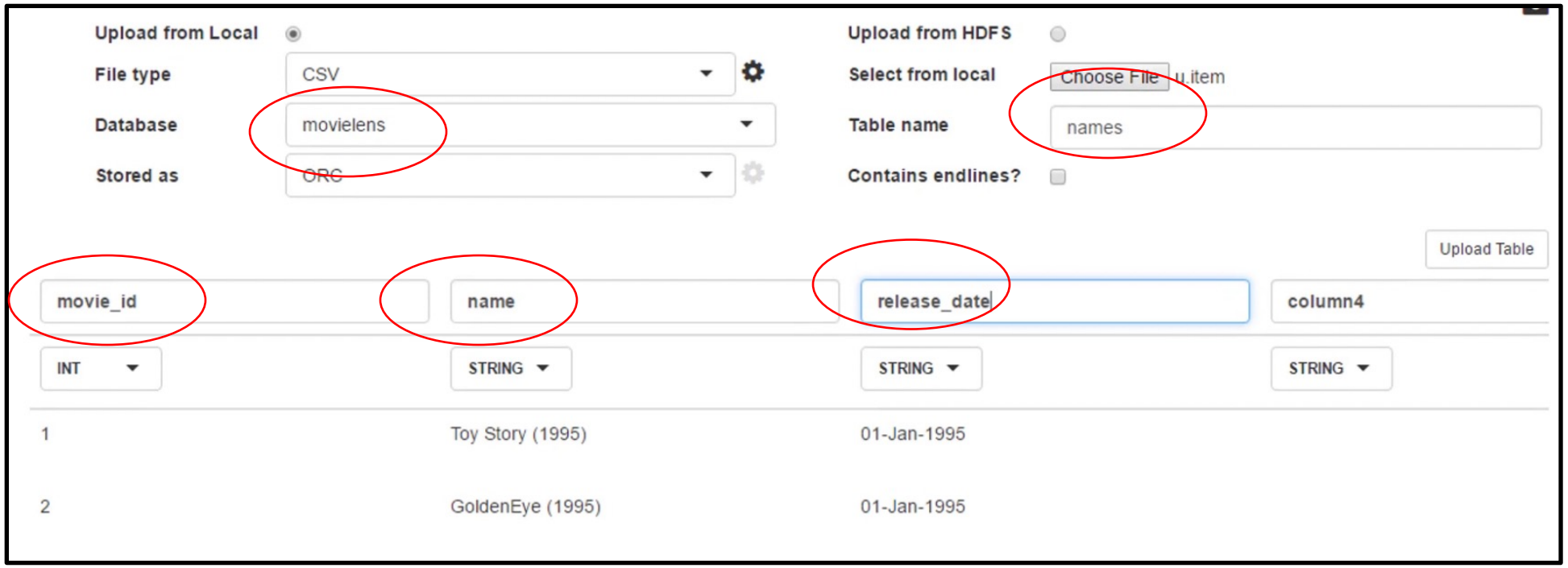


https://bit.ly/3yfkLyW

Let’s try it out

- Compare performance of a Hive query using Tez vs. MapReduce
- Login to Ambari as *admin*
  - *127.0.0.1:8080*
- Go in *Hive view*
- You should already have *movielens* database with *ratings* (*u.data - tab delimited; ASCII characters #9*) table ready
  - *CREATE DATABASE movielens;*
- Upload movie title dataset (*u.item - pipe delimited; ASCII characters #124*) into Hive under the *movielens* database with the table called *names*

Uploading u.item data



Hive Query

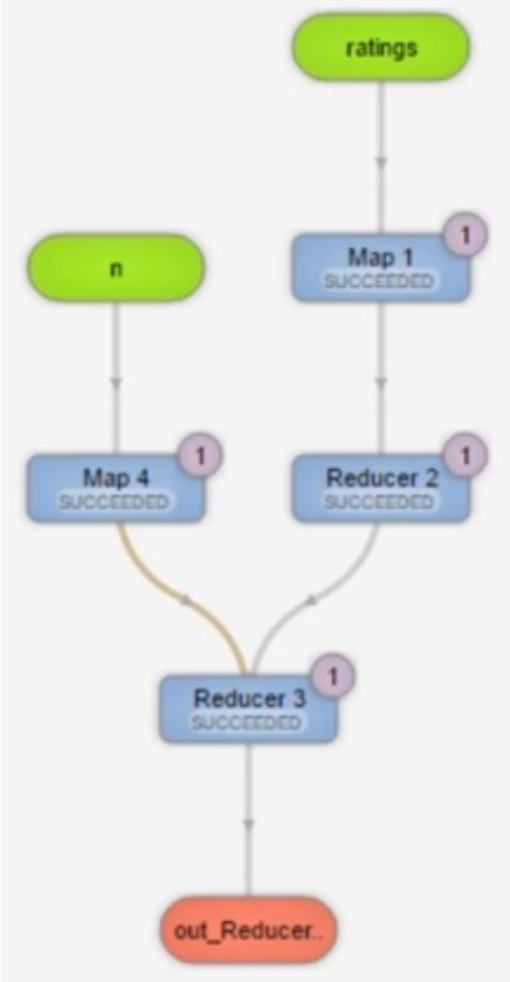
```
1  DROP VIEW IF EXISTS topMovieIDs;
2
3  CREATE VIEW topMovieIDs AS
4  SELECT movie_id, count(movie_id) as ratingCount
5  FROM movielens.ratings
6  GROUP BY movie_id
7  ORDER BY ratingCount DESC;
8
9  SELECT n.name, ratingCount
10 FROM topMovieIDs t JOIN movielens.names n ON t.movie_id = n.movie_id;
```

TEZ VS. MapReduce

~ 30 seconds

1 min 30 seconds

TEZ Graphical View



- acyclic graph
- jobs run in parallel
- While reducing the first MR stage, already start mapping the second one
- combining all the results in one step in final reducer