Hálózatba kapcsolt erőforrás platformok és alkalmazásaik

Simon Csaba
TMIT
2018

What is Hadoop?

- Apache top level project, open-source implementation of frameworks for reliable, scalable, distributed computing and data storage.
- It is a flexible and highly-available architecture for large scale computation and data processing on a network of commodity hardware.

What is Hadoop?

- **Hadoop:**
  - an open-source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license.
- **Goals / Requirements:**
  - Abstract and facilitate the storage and processing of large and/or rapidly growing data sets
    - Structured and non-structured data
    - Simple programming models
  - High scalability and availability
  - Use commodity (cheap!) hardware with little redundancy
  - Fault-tolerance
  - Move computation rather than data

Hadoop’s Developers

Doug Cutting

2005: Doug Cutting and Michael J. Cafarella developed Hadoop to support distribution for the Nutch search engine project.

The project was funded by Yahoo.

2006: Yahoo gave the project to Apache Software Foundation.

Google Origins

2003

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google*

hadoop HDFS

2004

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

hadoop Map Reduce

2006

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com

Google, Inc.

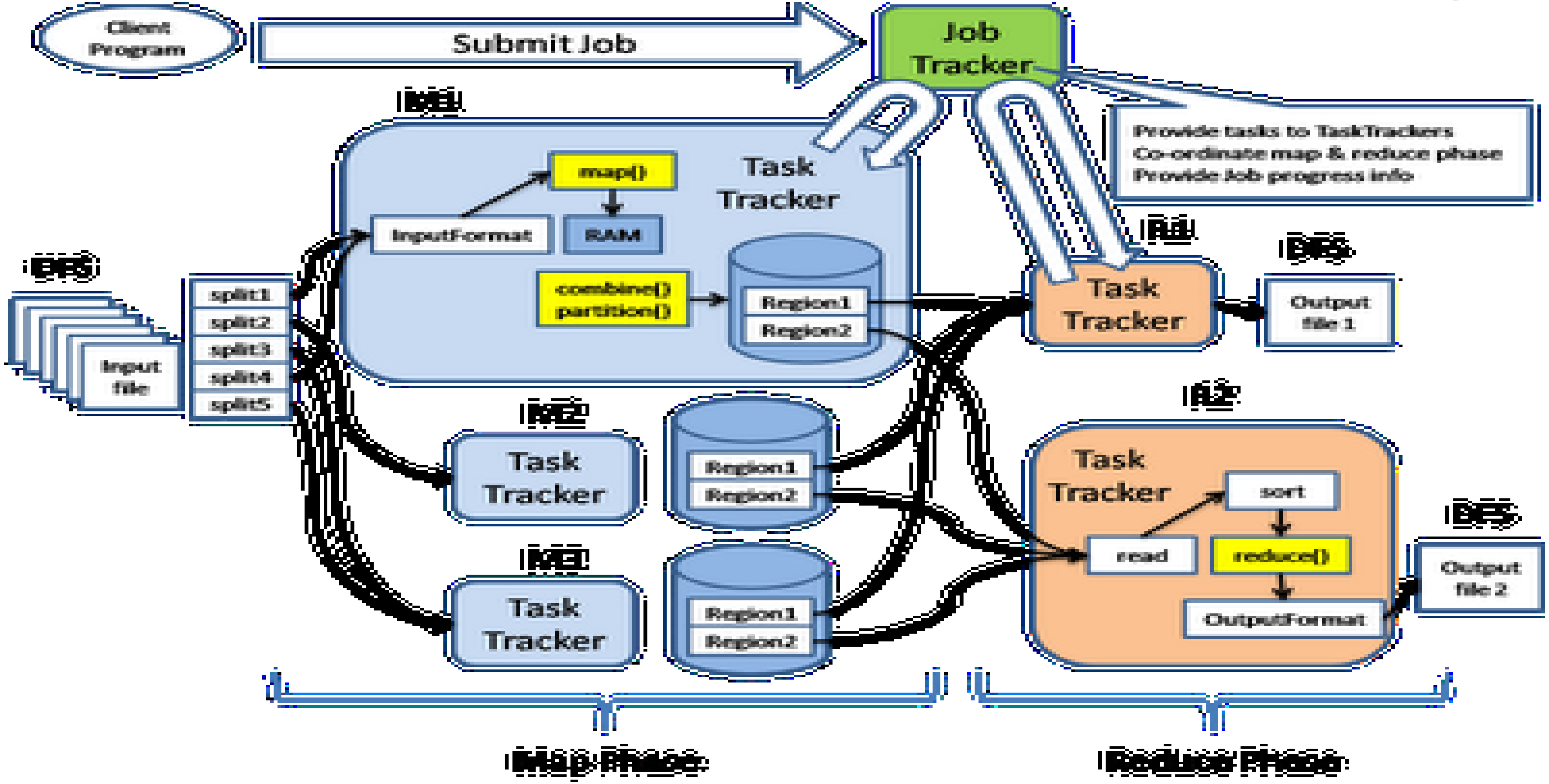
APACHE HBASE

Abstract

igtable is a distributed storage system for managing tured data that is designed to scale to a very large petabytes of data across thousands of commodity ers. Many projects at Google store data in Bigtable, uding web indexing, Google Earth, and Google Fi- e. These applications place very different demands Bigtable, both in terms of data size (from URLs to

achieved scalability and high performance, but Big provides a different interface than such systems. Big does not support a full relational data model; inste provides clients with a simple data model that sup dynamic control over data layout and format, an lows clients to reason about the locality properties o data represented in the underlying storage. Data i dexed using row and column names that can be arbi strings. Bigtable also treats data as uninterpreted str

Hadoop's Architecture: MapReduce Engine



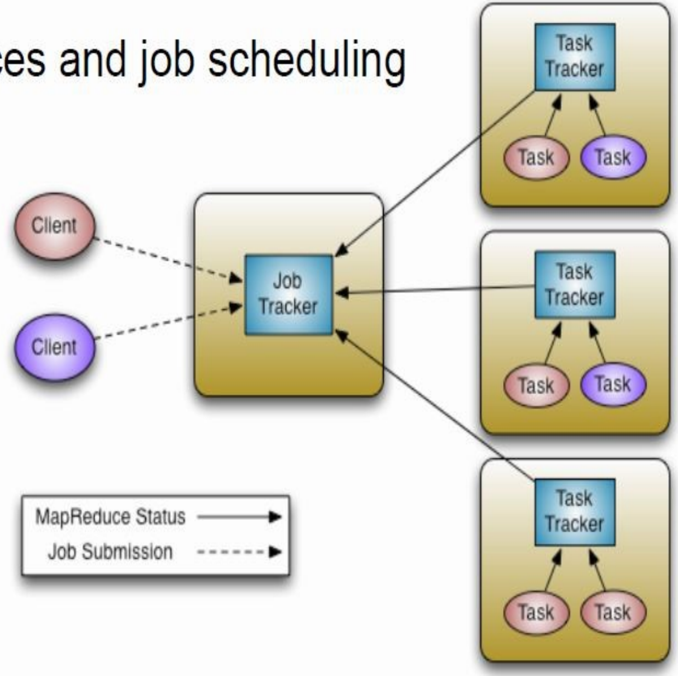
Hadoop’s Architecture

MapReduce Engine:

- JobTracker & TaskTracker
- JobTracker splits up data into smaller tasks(“Map”) and sends it to the TaskTracker process in each node
- TaskTracker reports back to the JobTracker node and reports on job progress, sends data (“Reduce”) or requests new jobs

Hadoop MapReduce Classic

- JobTracker
  - Manages cluster resources and job scheduling
- TaskTracker
  - Per-node agent
  - Manage tasks



Current MapReduce Limitations

- Scalability
  - Maximum Cluster Size – 4000 Nodes
  - Maximum Concurrent Tasks – 40000
  - Coarse synchronization in Job Tracker
- Single point of failure
  - Failure kills all queued and running jobs
  - Jobs need to be resubmitted by users
- Restart is very tricky due to complex state

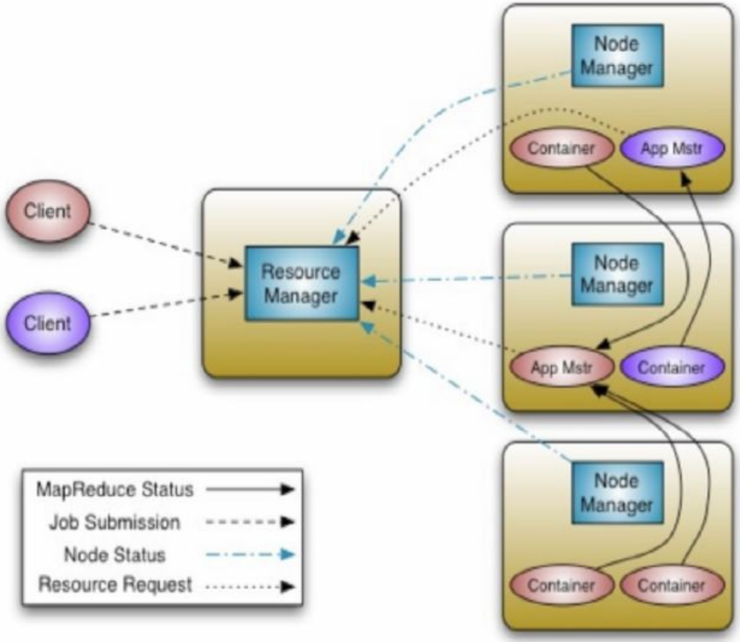
Hadoop proposal: YARN

- Yet Another Resource Negotiator
- YARN Application Resource Negotiator(Recursive Acronym)
- Remedies the scalability shortcomings of “classic” MapReduce
- Is more of a general purpose framework of which classic mapreduce is one application.

YARN

- Split up the two major responsibilities of the JobTracker/TaskTracker into separate entities
- JobTracker
  - global Resource Manager - Cluster resource management
  - per application Application Master – doing job scheduling and monitoring, negotiating the resource containers from the Scheduler, tracking their status and monitoring for progress
- Tasktracker
  - new per-node slave Node Manager (NM) - responsible for launching the applications' containers, monitoring their resource usage (cpu, memory, disk, network) and reporting to the Resource Manager
  - (a per-application Container running on a NodeManager)
- YARN maintains compatibility with existing MapReduce applications and users

YARN – Architectural Overview



- Scalability - Clusters of 6,000-10,000 machines
  - Each machine with 16 cores, 48G/96G RAM, 24TB/36TB disks
  - 100,000+ concurrent tasks
  - 10,000 concurrent jobs

Comparing YARN with MapReduce

Criteria	**YARN**	**MapReduce**
Type of processing	Real-time, batch, interactive processing with multiple engines	Silo & batch processing with single engine
Cluster resource optimization	Excellent due to central resource management	Average due to fixed Map & Reduce slots
Suitable for	MapReduce & Non – MapReduce applications	Only MapReduce applications
Managing cluster resource	Done by YARN	Done by JobTracker

…supported by Hadoop's own filesystem

- The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant.
  - highly fault-tolerant and is designed to be deployed on low-cost hardware.
  - provides high throughput access to application data and is suitable for applications that have large data sets.
  - relaxes a few POSIX requirements to enable streaming access to file system data.
  - part of the Apache Hadoop Core project. The project URL is http://hadoop.apache.org/core/.

MapReduce with HDFS

- Distributed, with some centralization
- Main nodes of cluster are where most of the computational power and storage of the system lies
- Main nodes run TaskTracker to accept and reply to MapReduce tasks, and also DataNode to store needed blocks closely as possible
- Central control node runs NameNode to keep track of HDFS directories & files, and JobTracker to dispatch compute tasks to TaskTracker
- Written in Java, also supports Python and Ruby

HDFS properties

- Hadoop Distributed Filesystem
- Tailored to needs of MapReduce
- Targeted towards many reads of filestreams
- Writes are more costly
- High degree of data replication (3x by default)
- No need for RAID on normal nodes
- Large blocksize (64MB)
- Location awareness of DataNodes in network

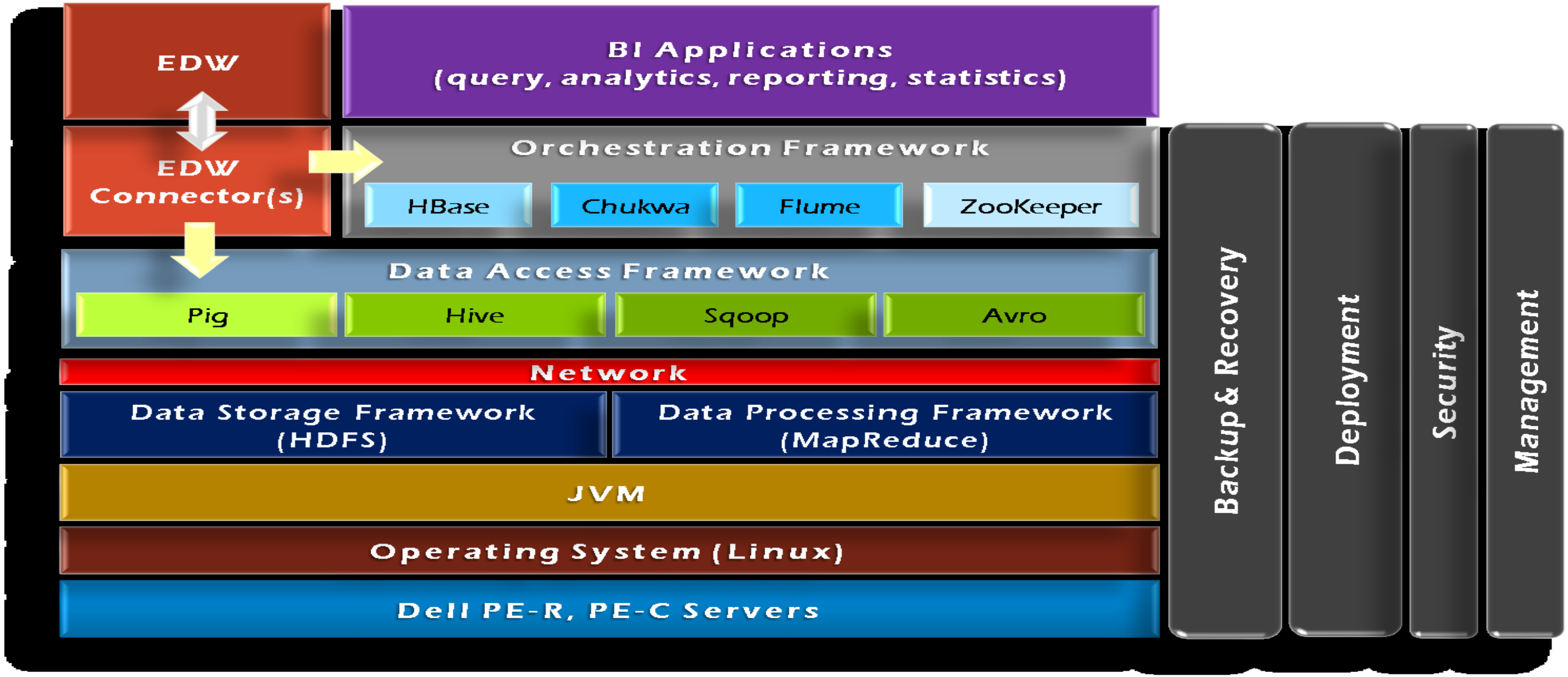
HDFS Name Node

- Stores metadata for the files, like the directory structure of a typical FS.
- The server holding the NameNode instance is quite crucial, as there is only one.
- Transaction log for file deletes/adds, etc. Does not use transactions for whole blocks or file-streams, only metadata.
- Handles creation of more replica blocks when necessary after a DataNode failure

HDFS Data Node

- Stores the actual data in HDFS
- Can run on any underlying filesystem (ext3/4, NTFS, etc)
- Notifies NameNode of what blocks it has
- NameNode replicates blocks 2x in local rack, 1x elsewhere

Hadoop Framework Tools



Does Hadoop require HDFS?

- Errr, actually...
- None of these components are necessarily limited to using HDFS
- Many other distributed file-systems with quite different architectures work
- IF Hadoop knows which hosts are closest to the data THEN reduces network traffic
- Many other software packages besides Hadoop's MapReduce platform make use of HDFS

YARN advantage over MapReduce

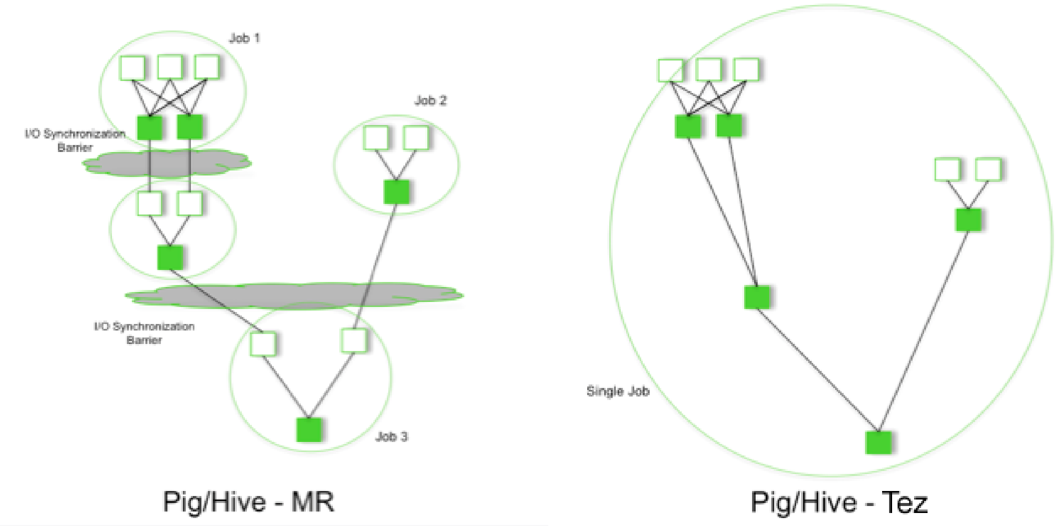
- Support for programming paradigms other than MapReduce (Multi tenancy)
  - Tez – Generic framework to run a complex DAG
  - HBase on YARN (HOYA)
  - Compute engine (e.g., Machine Learning): Spark
  - Graph processing: Giraph
  - Real-time processing: Apache Storm
  - Enabled by allowing the use of paradigm-specific application master
  - *Run all on the same Hadoop cluster!*

Tez on YARN

- Hindi for speed
- Provides a general-purpose, highly customizable framework that creates simplifies data-processing tasks across both small scale (low-latency) and large-scale (high throughput) workloads in Hadoop.
- Generalizes the MapReduce paradigm to a more powerful framework by providing the ability to execute a complex DAG
- Enables Apache Hive, Apache Pig and Cascading can meet requirements for human-interactive response times and extreme throughput at petabyte scale

Tez on YARN

- Original MapReduce requires disk I/O after each stage
- A series of MapReduce jobs following each other would result in lots of I/O
- Tez eliminates these intermediate steps, increasing the speed and lowering the resource usage



Tez on YARN

- Performance gains over Mapreduce
  - Eliminates replicated write barrier between successive computations
  - Eliminates job launch overhead of workflow jobs
  - Eliminates extra stage of map reads in every workflow job
  - Eliminates queue and resource contention suffered by workflow jobs that are started after a predecessor job completes

HBase on YARN(HOYA)

- Be able to create on-demand HBase clusters easily -by and or in apps
  - With different versions of HBase potentially (for testing etc.)
- Be able to configure different HBase instances differently
  - For example, different configs for read/write workload instances
- Better isolation
  - Run arbitrary co-processors in user's private cluster
  - User will own the data that the hbase daemons create

HBase on YARN(HOYA)

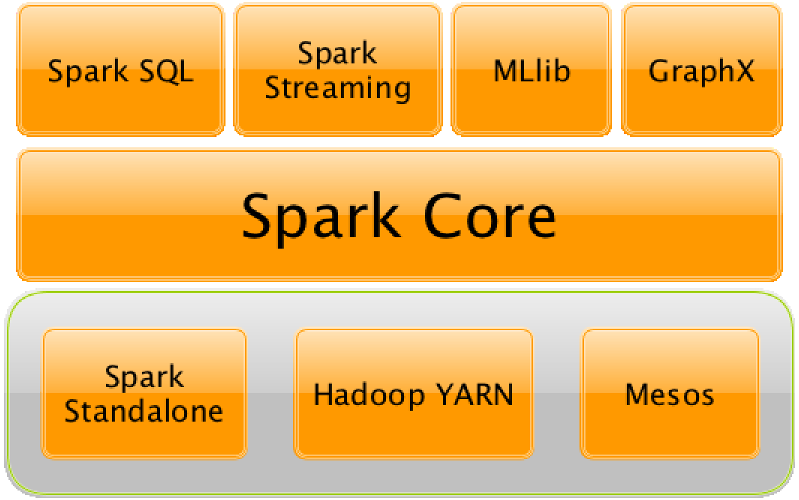
- MR jobs should find it simple to create (transient) HBase clusters
  - For Map-side joins where table data is all in HBase, for example
- Elasticity of clusters for analytic / batch workload processing
  - Stop / Suspend / Resume clusters as needed
  - Expand / shrink clusters as needed
- Be able to utilize cluster resources better
  - Run MR jobs while maintaining HBase's low latency SLAs

Hadoop Subprojects - Summary

- Pig
  - High-level language for data analysis
- HBase
  - Table storage for semi-structured data
- Zookeeper
  - Coordinating distributed applications
- Hive
  - SQL-like Query language and Metastore
- Mahout
  - Machine learning

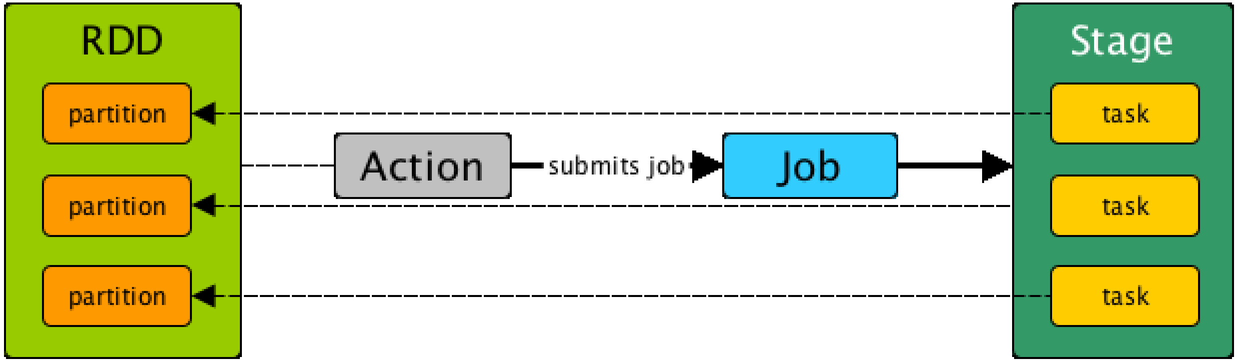
Apache Spark

- Standalone generic BigData computational framework
  - In-memory data processing
  - Batch and streaming mode
- Can be combined with Hadoop
- Without Hadoop, e.g., Kubernetes



Spark components

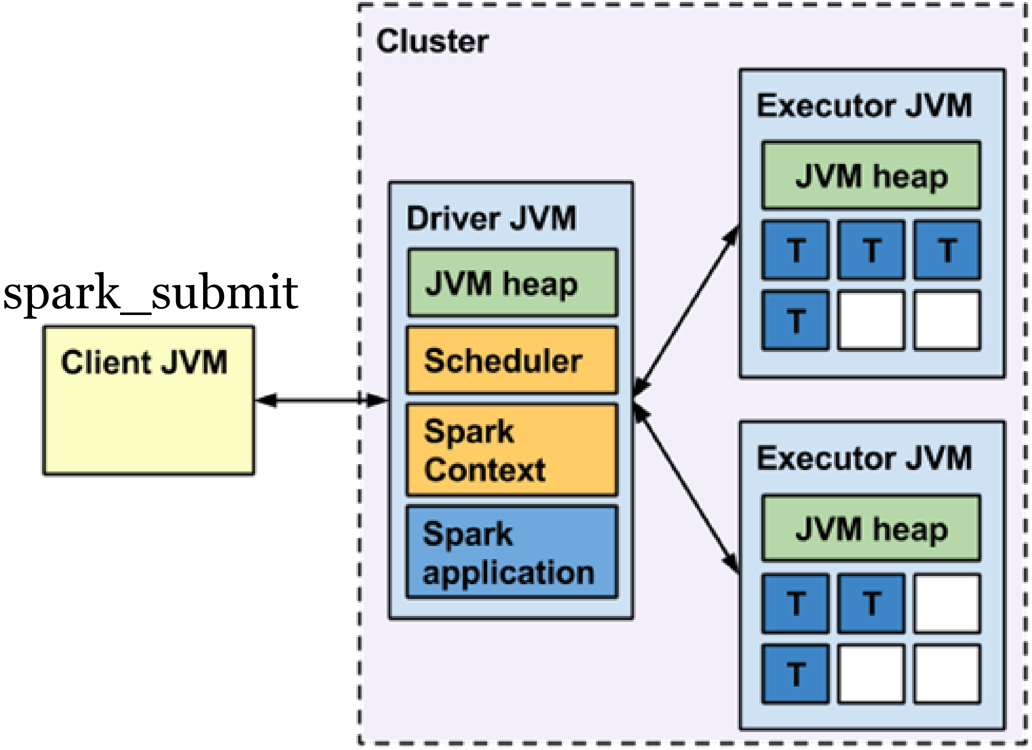
- **Resilient Distributed Dataset (RDD)**
  - the primary data abstraction and the core of Spark
  - resilient and distributed collection of records spread over many partitions
  - Shuffling: redistributing data across partitions
- **Stage**
  - physical unit of execution



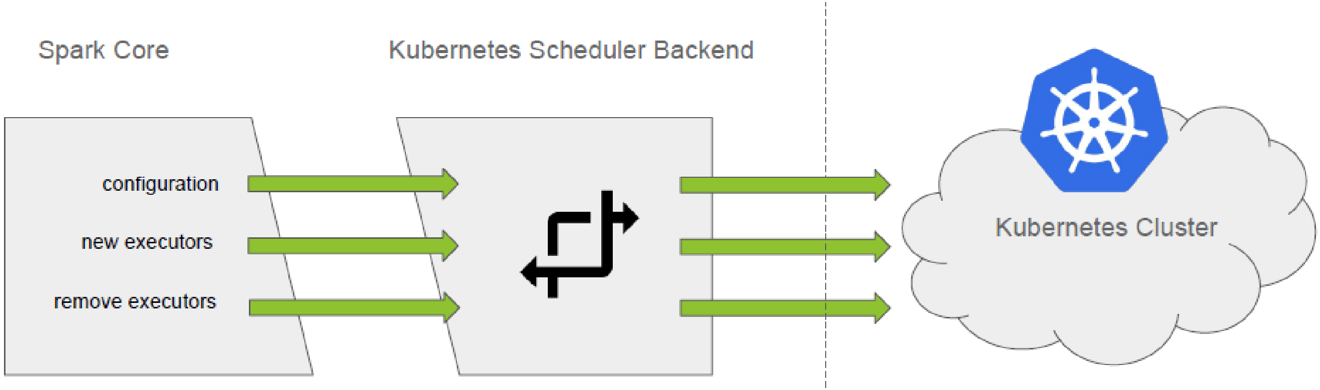
Spark terminology

Driver	*Process that contains the SparkContext*
Executor	*Process that executes one or more Spark tasks*
Master	*Process that manages applications across the cluster*
Worker	*Process that manages executors on a particular node*

Spark / cluster mode deployment



Spark on K8S

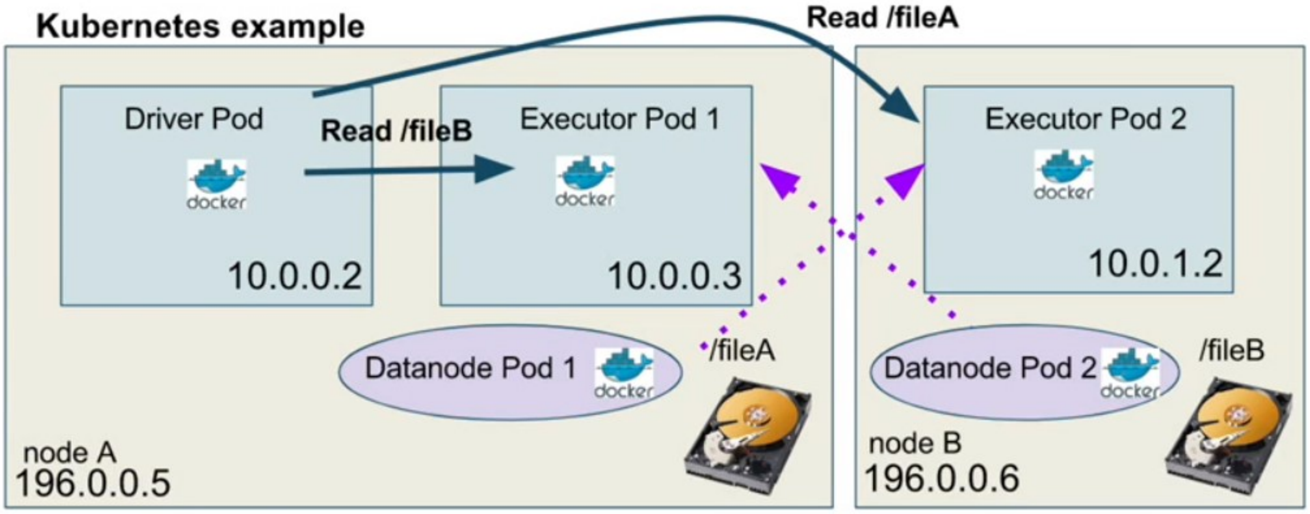


- Resource Requests
- Authnz
- Communication with K8s

- Runs Spark Drivers/Executors
- Runs Shuffle Service
- Runs Additional Components for Spark jobs

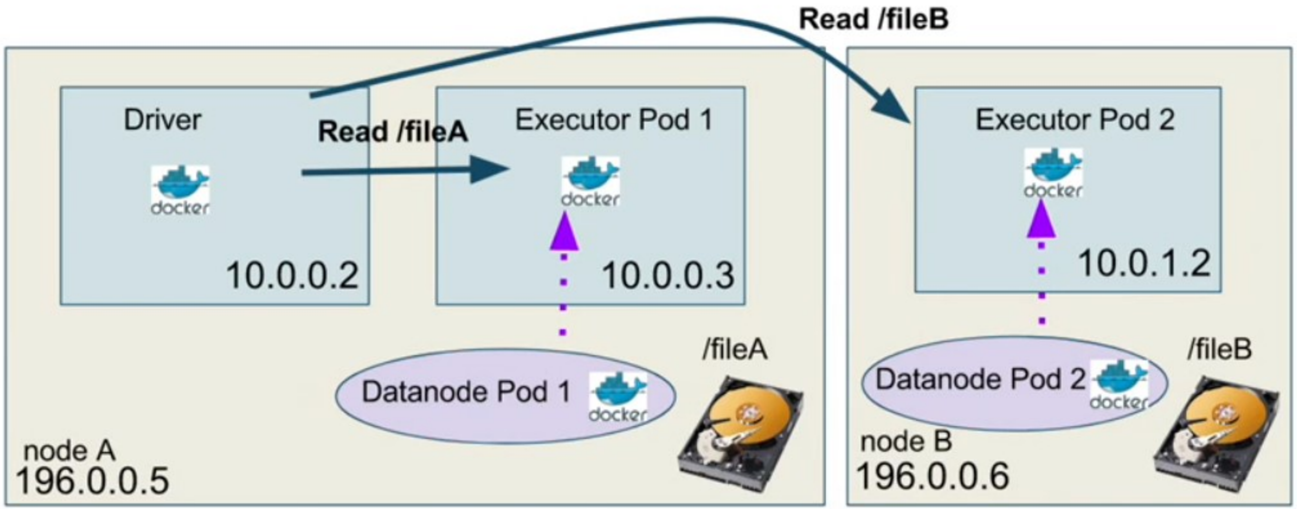
Spark on K8s + HDFS

- No YARN, no data locality?



Spark on K8s + HDFS

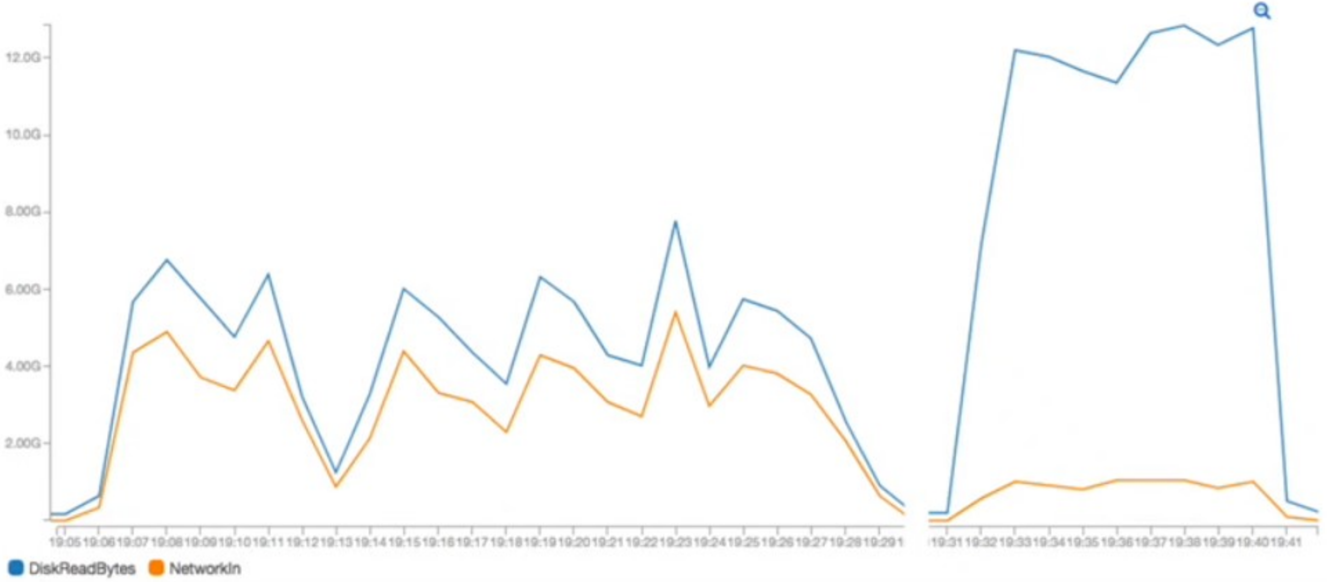
- K8S master provides an API to match executor and datanode host IDs (IP, label)



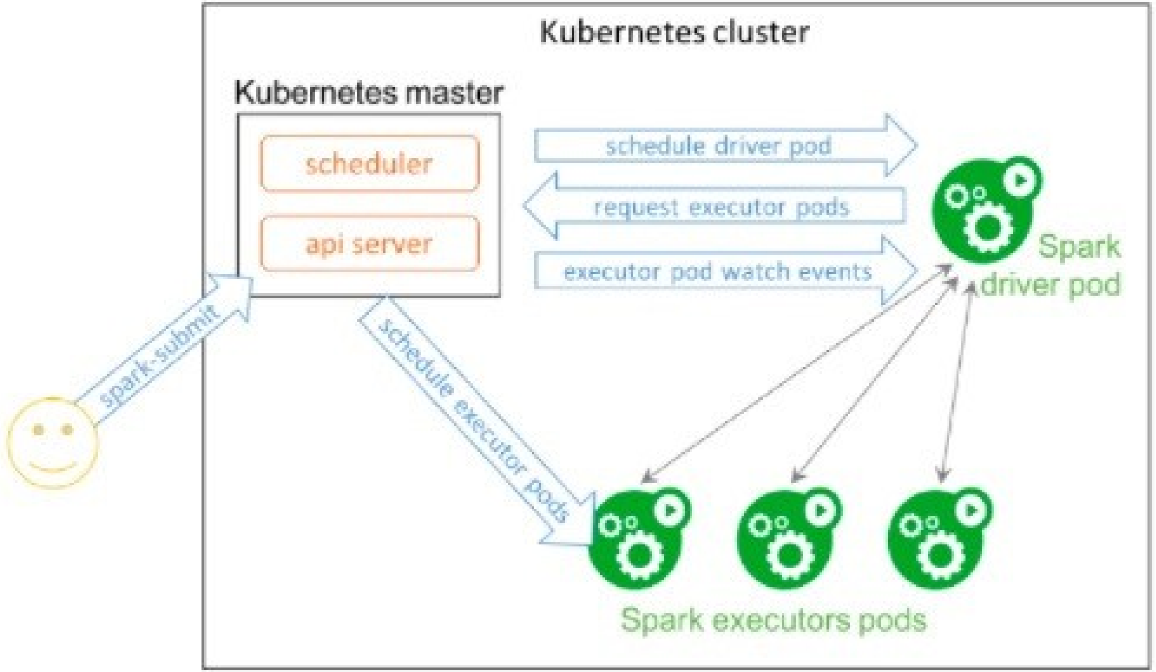
The effect of data locality

without data locality fix
- duration: 25 minutes

with data locality fix
- duration: 10 minutes

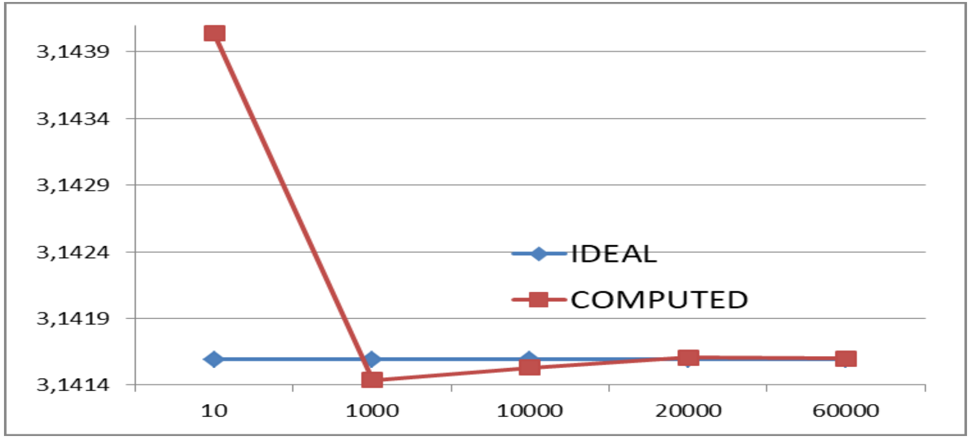


SparkPI example over Kubernetes



Apache Spark running natively in a Kubernetes cluster

Pi számítása BigData klaszterben



Pi számítása BigData klaszterben