Apache Spark™

Lightning-Fast Cluster Computing

Overview and use cases

Me

- “Hadoopified” and deeply fell in love with “Big Data” in 2010
- Software development Engineer – Relational DBs – @Amadeus – (2 years)
- “Big” Data Engineer – Hadoop , Data Analysis & NoSQL @Amadeus – (1 year)
- Now starting to be “Sparkified” ;-)

http://www.linkedin.com/in/ericdjatsa

Agenda

- Big Data
  - Short overview
  - New challenges
- Hadoop – the Big elephant that solves "Big" problems
  - Overview
  - Hadoop map-reduce – Advantages and limitations
- Spark – will make the Big elephant "Fly"
  - Short overview
  - Architecture
  - Demo-use cases : bookings analysis, flight delay prediction (Spark SQL, Spark Streaming, Machine Learning)

1 Big Data

Big Data – short explanation

A graceful mixture of all three V : (Big) **Variety,** (Big) **Velocity,** (Big) **Volume of Data**

2000 – 2010

Enterprise Data Lake

2010 – ….

Continuous stream of data (structured and unstructured)

1980 – 2000

Enterprise data warehouses

Big Data – new challenges

- A single machine can't deal with Big – Volume, Variety, Velocity of Data
  - Solution : use many machines (*sounds trivial* ☺)

- New Challenges
  - Distributed Storage
  - Distributed Processing
  - Communication between machines
  - Network and machine failure
  - …

2

Hey we found a solution !

hadoop

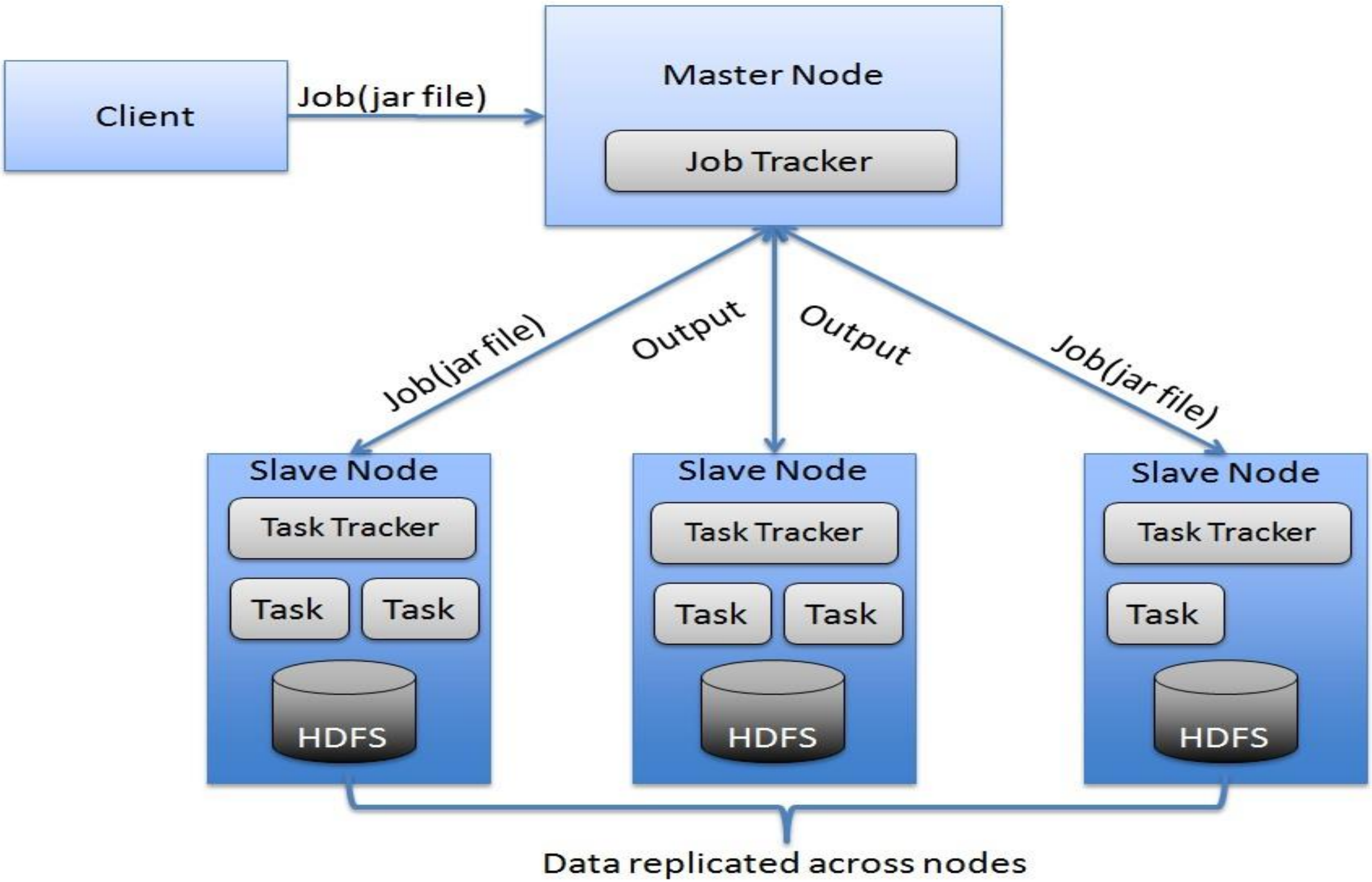
Hadoop

Big Data Platform

- Hadoop platform
  - Distributed file system (hdfs)
  - Parallel computing framework (originaly Map/Reduce)
  - Fault tolerant, handle replication, Data locality , node failures, …
  - Scalable (Yahoo! has a 4000-node cluster)
  - High performances : sorted a TB of random integers in 62 seconds
  - Cheap ☺ : deploy on commodity machines

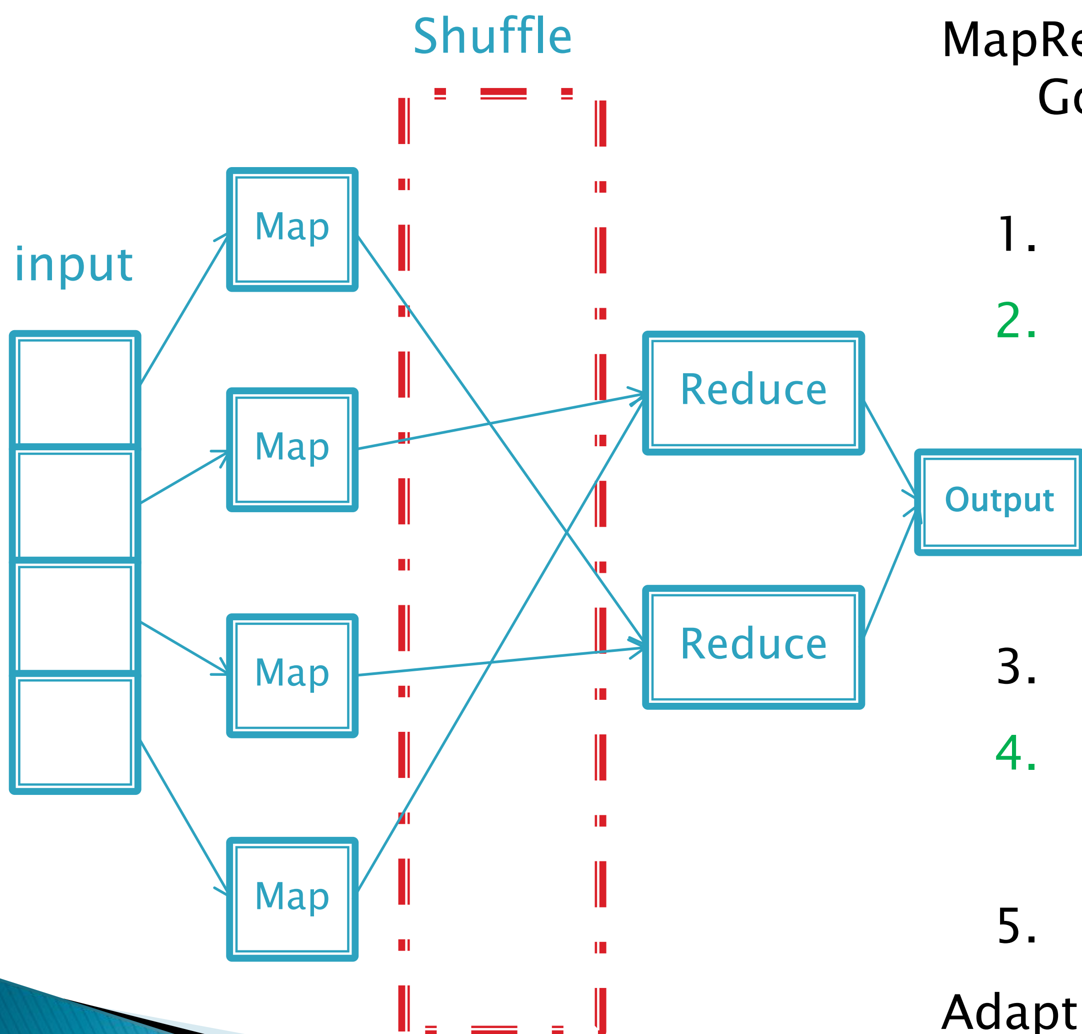
Hadoop

Architecture



Hadoop

Map-Reduce : core processing framework



MapReduce Framework
Google patent (2004)

1. Read a lot of data
2. Map: extract something you care about
3. Shuffle and sort
4. Reduce: aggregate, summarize, filter, transform
5. Write the results

Adapt Map and Reduce part to solve different use cases

Hadoop

Map-Reduce limitations

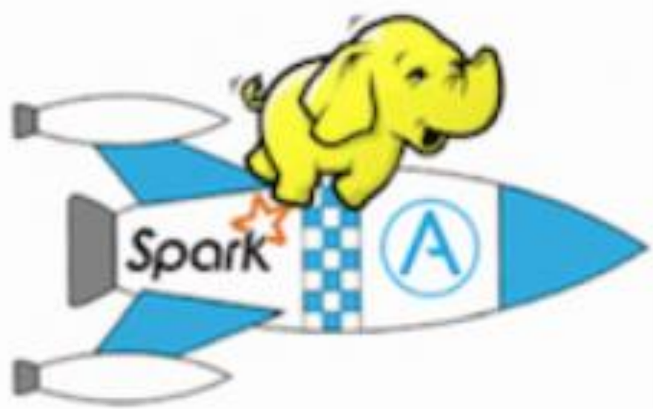
- Relies on *persistent storage (disk)* to provide *fault tolerance* between **map**, **shuffle** and **reduce** phases
  - As a consequence persistent storage introduces higher latency

- It's one-pass computation model makes it a *poor fit* for *low-latency applications* and *iterative computations*, such as *machine learning and graph algorithms.*

- Hadoop Map-Reduce not good at:
  - More **complex**, multi-stage applications (graph algorithms, machine learning)
  - More **interactive** ad-hoc queries
  - More **real-time** online processing

- All three of these apps require *fast data sharing* across parallel jobs.

3 Well... let's make the elephant Fly !



Apache Spark

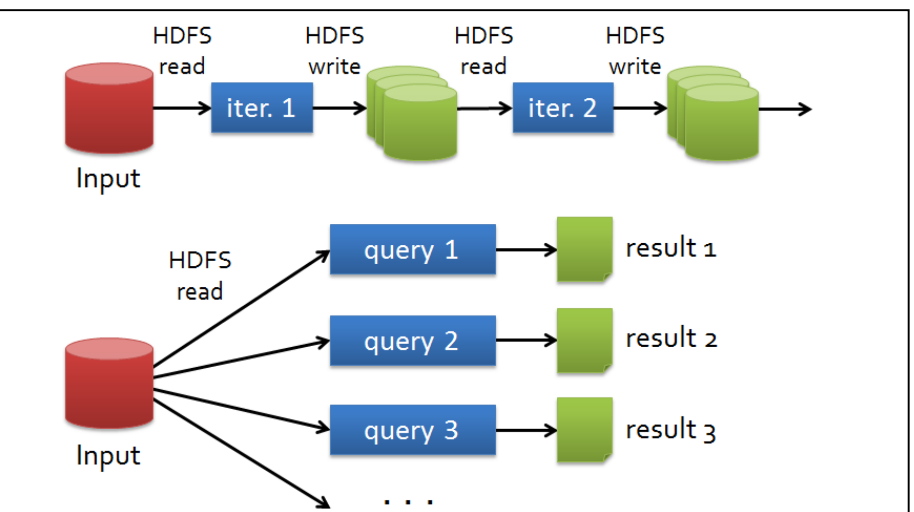
overview

- fast and general engine (map-reduce like but not restricted) for large-scale data processing

- *In-memory* *(as much as possible)* data storage for very fast iterative queries, with *fault-tolerance* mechanisms

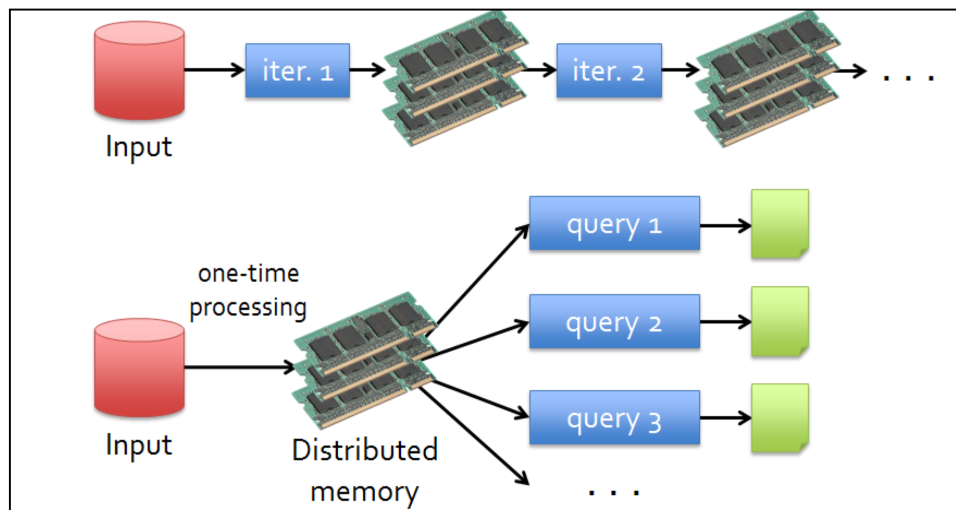
- Compatible with Hadoop's storage APIs

Hadoop Map-Reduce



Slow due to replication, serialization, and disk IO

Apache Spark



10-100× faster than network and disk

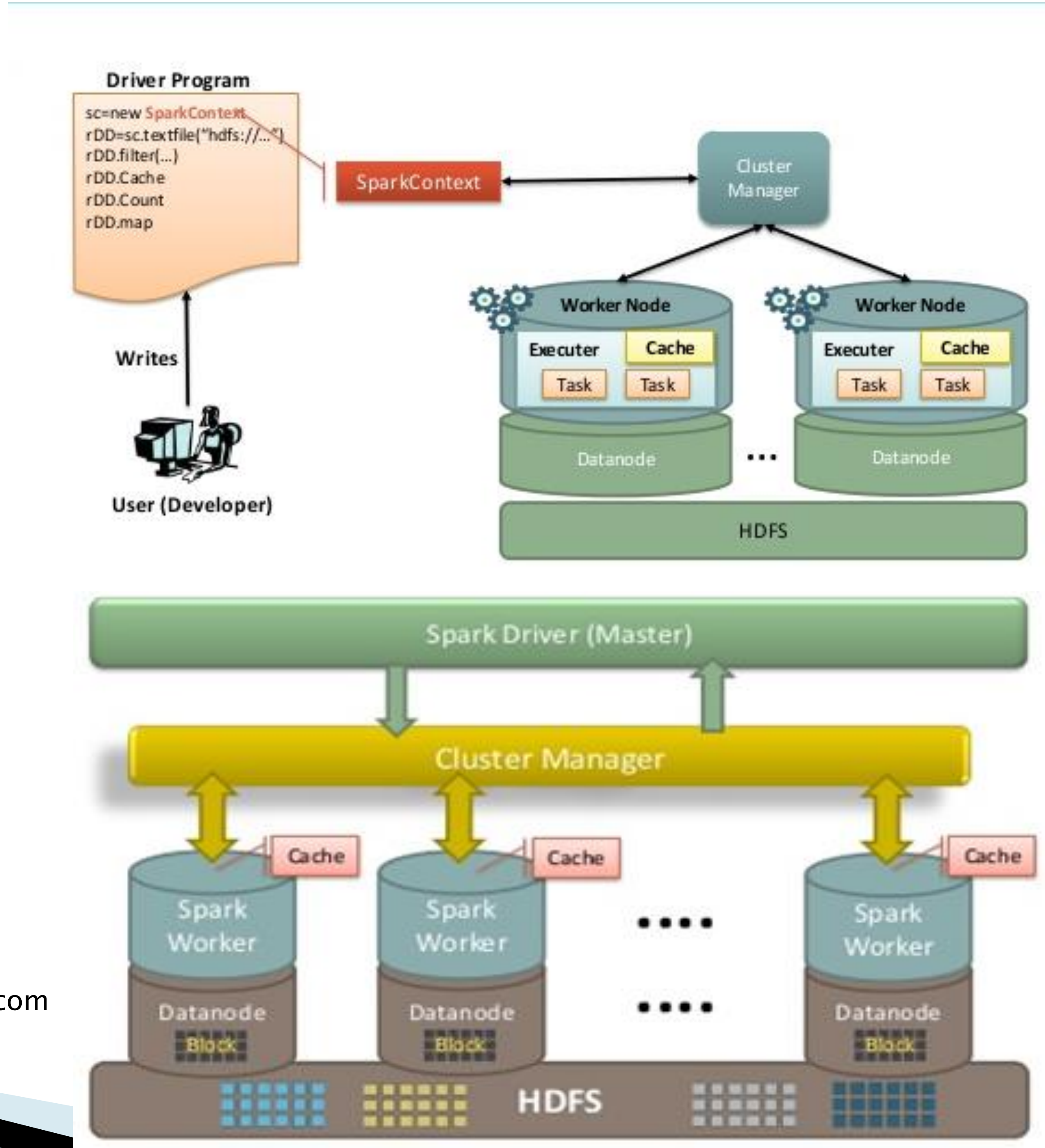
Spark Main Goal

- Provide distributed *memory* abstractions for clusters to support apps with working sets
- Retain the attractive properties of Hadoop MapReduce:
  - Fault tolerance (for crashes & stragglers)
  - Data locality
  - Scalability

Challenge : How to design a **distributed memory abstraction** that is both **fault-tolerant** and **efficient** ?

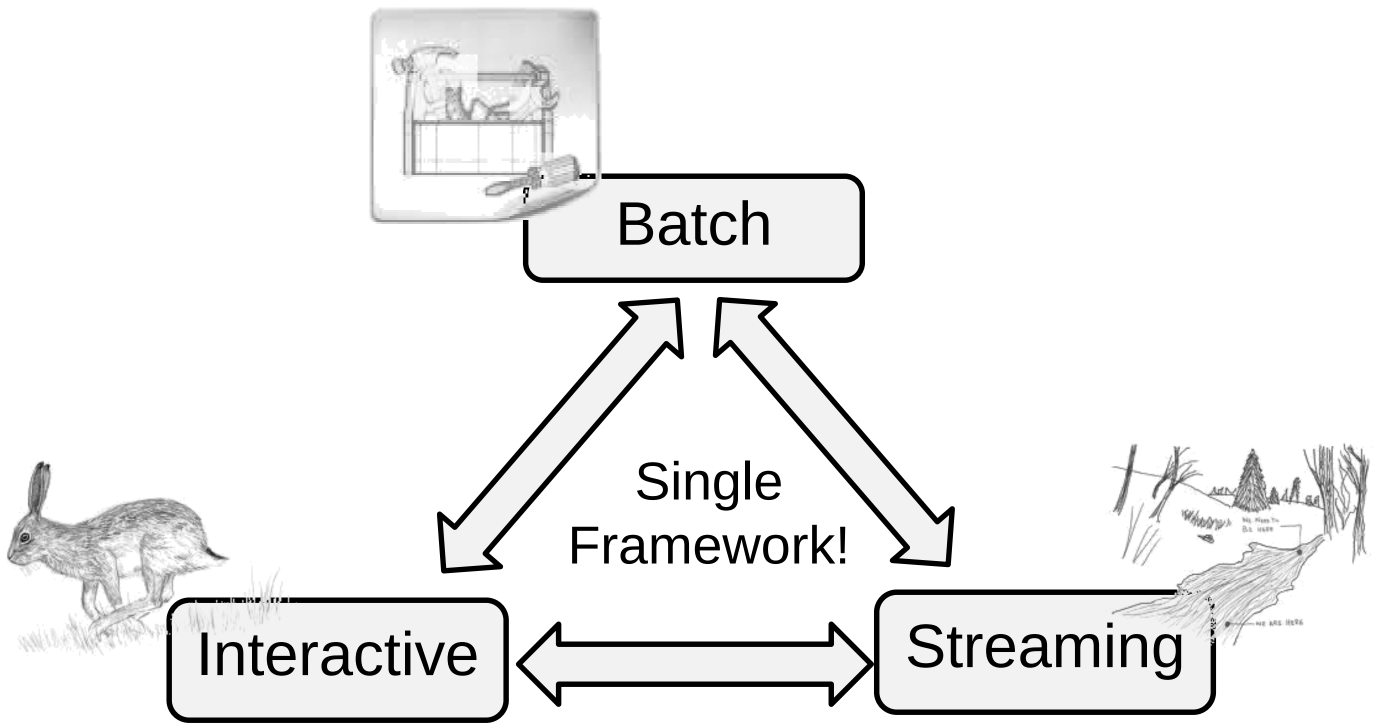
Solution **:** augment data flow model with "resilient distributed datasets" (***RDDs***)

Spark
Architecture



Credit : www.unicomlearning.com

Spark Goal: unification



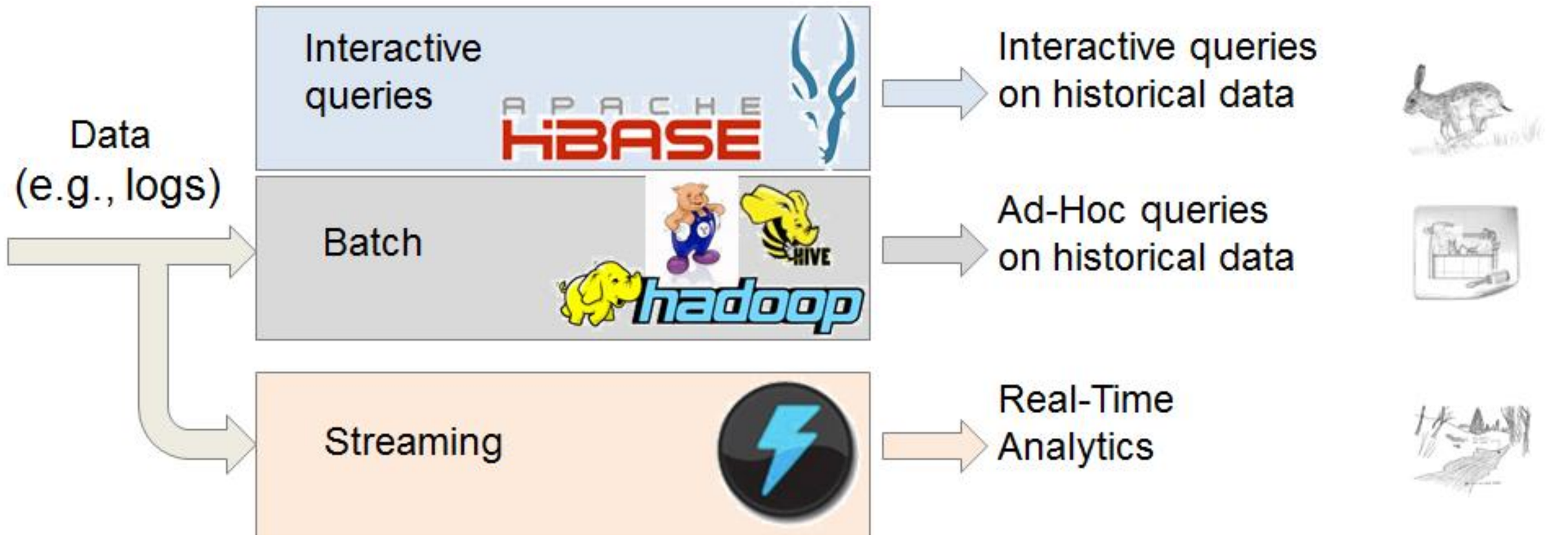
Support *batch*, *streaming*, and ***interactive*** computations…

… in a ***unified*** framework… allowing ***code-reuse*** across all type of computations

Easy to develop ***sophisticated*** algorithms (e.g., graph processing , machine learning, near real-time analytics)

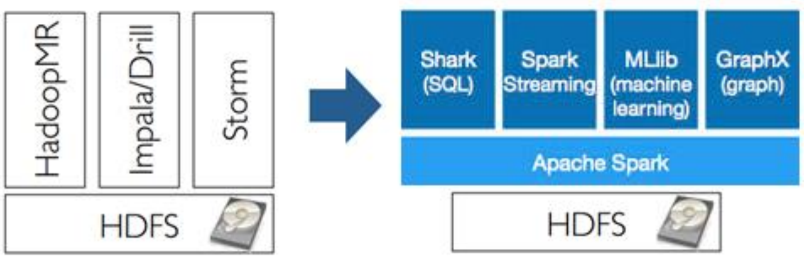
The Need For Unification

- Today's state-of-art analytics stack



Challenge 1 : need to maintain three stacks

- Expensive and complex
- Hard to compute consistent metrics across stacks



Challenge 2 : hard/slow to share data, e.g.,

» Hard to perform interactive queries on streamed data

What's the magic behind Spark ?

Spark RDD

RDD (*Resilient distributed dataset*) :

- ***distributed memory abstraction***
- enables to perform ***parallel*** ***in-memory*** computations on large clusters in a ***fault-tolerant*** manner.
- can be considered as a ***read-only, partitioned collection of records***.
- can be re-computed easily thanks to it's list of parent RDDs

RDD ***interface*** :

- A set of partitions
- A set of Dependencies on parent RDDs
- A function (compute) for computing it from its parents
- Optionally a partitioner for key-value RDD (e.g to say that the RDD is hash partitioned
- Optionally a list of preferred locations to compute each split

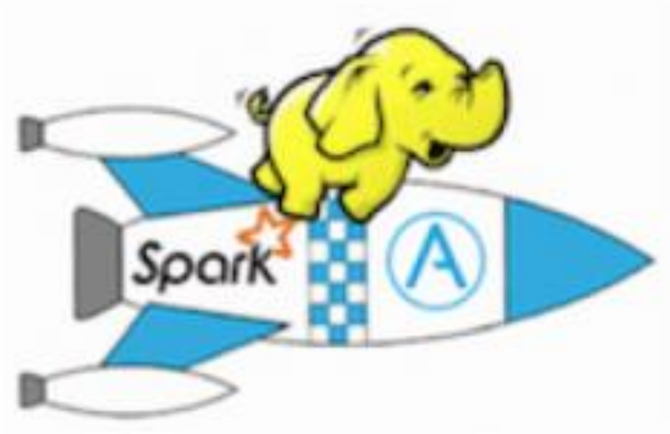
Example Spark code snippet

```
val lines = spark.textFile("hdfs://.../serverlogs")
val  errors = lines.filter(_.startsWith("ERROR"))
val tsField = errors.map(_.split('\t')(3))
```

HadoopRDD
FilteredRDD
MappedRDD

RDD

3 Ok! Now let's see all this in Play !



Demo1

Flights Bookings count per (departure, destination) airport pair – in Batch

Bookings log data

BookingID, BookingTimestamp , CarrierCode, FlightID, CustomerID, AgeCategory , DepartureAirport, DestinationAirport, DepartureDate, ReturnDate, NbPassengers , TicketPrice
1,20141204-103000,AF,153,2145,SENIOR,NCE,CDG,20141220,20141231,2,150
2,20141204-103005,AF,124,1462,ADULT,LYS,MRS,20141215,20150105,1,115
3,20141206-154500,BA,326,4527,ADULT,LGW,BOD,20150107,20150121,1,200
4,20141207-112000,AL,147,5982,INFANT,NCE,CDG,20150204,20150210,1,100
….

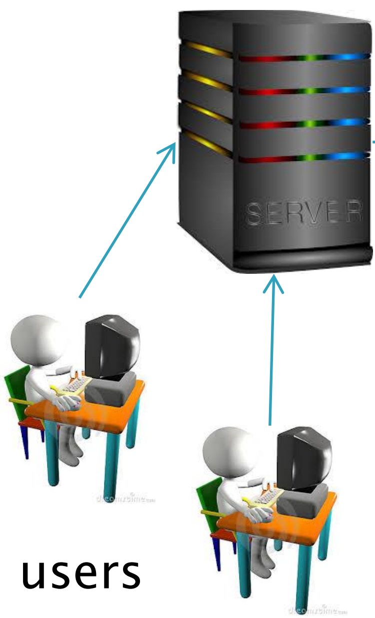
Spark Spark SQL

(NCE,CDG) , 2
(LYS,MRS) , 1
(LGW,BOD), 1

Demo2

Flights Bookings count per (departure, destination) airport pair – Streaming (near real-time)

Booking server



users

logs

Booking logs generated in real-time

BookingID, BookingTimestamp …. , DepartureAirport, DestinationAirport, ….
1,20141204-103000, AF,153,2145,SENIOR,NCE,CDG,
3 20141204-103000, LF,147,5982,INFANT,NCE,CDG, …
2,20141204-103002, AF,124,1462,ADULT,LYS,MRS,201…
….

Spark Streaming + Spark SQL

Time : 20141204-103000

(NCE,NICE,CDG,PARIS) , 2
Time : 20141204-103002

(LYS,LYON,MRS,MARSEILLE) , 1

Demo3

Flights Delay prediction – Machine Learning

Training set

Year,Month,…FlightNum,
DepartureDelay,….
2007,1,…,256,17
2007,3,…,365,10

Test set

Year,Month,…FlightNum,
DepartureDelay,….
2007,6,…,358,12
2007,8,…,627,14

Spark MLib (machine learning)

Flight Delays Prediction Model

Live flights data

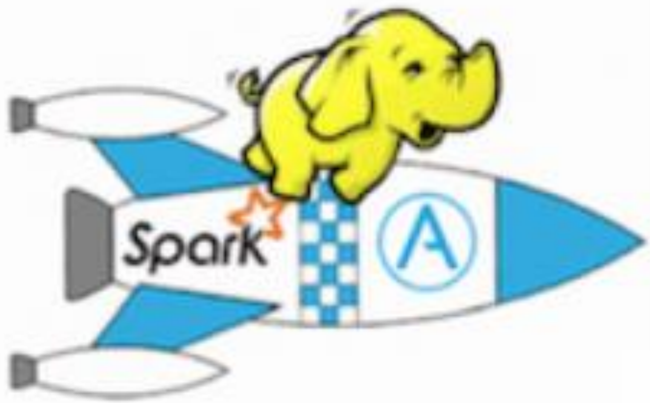
Year,Month,…FlightNum,
DepartureDelay,….
2008,7,…,256,17
2008,11,…,365,10

Spark Streaming + Spark MLib (machine learning)

Flight delays prediction on live data

Flight No 7, from ORD, destination = HLN:prediction for delay= 1.0 , actual delay=1.0
Flight No 11, from ORD, destination = PHX:prediction for delay= 1.0, actual delay=1.0
Flight No 1, from DEN, destination = GEG:prediction for delay= 1.0, actual delay=0.0
…

Sparkling Thanks !



Questions ?