Apache Hadoop, Spark and Big Data Foundations

Douglas Eadline

Presenter

Douglas Eadline
deadline@basement-supercomputing.com
@thedeadline

- HPC/Hadoop Consultant/Writer
- http://www.basement-supercomputing.com

Outline

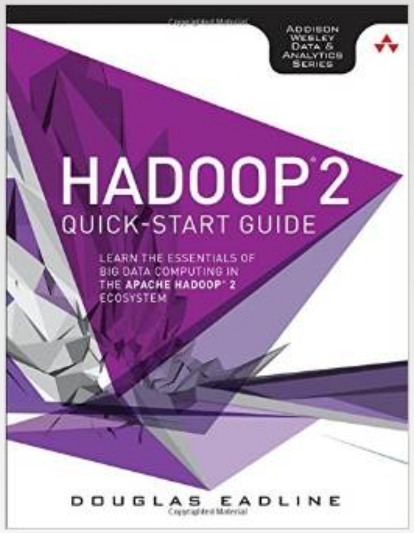
- **Segment 1:** Why is Hadoop Such a Big Deal? (50 m)
- Break (5 m)
- **Segment 2:** Hadoop Distributed File System (HDFS) Basics (25 m)
- **Segment 3:** Hadoop MapReduce Framework (25 m)
- Break (5 m)
- **Segment 4:** Making life Easier: Spark (20 m)
- **Segment 5:** Real World Applications/Wrap-up (15 M)

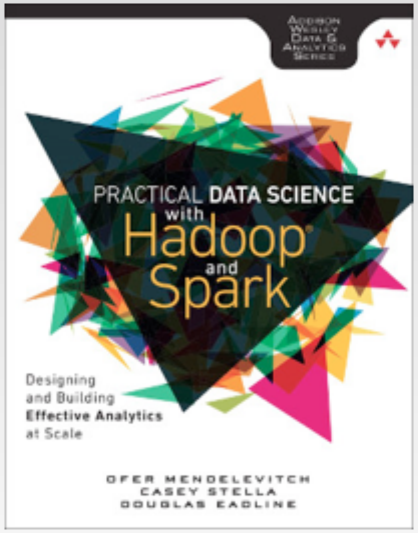
Supporting Materials

Hadoop 2 Quick-Start: http://www.clustermonkey.net/Hadoop2-Quick-Start-Guide

Hadoop Fundamentals: https://www.safaribooksonline.com/library/view/hadoop-fundamentals-livelessons/9780134052489

Practical Data Science with Hadoop and Spark: http://www.clustermonkey.net/Practical-Data-Science-with-Hadoop-and-Spark

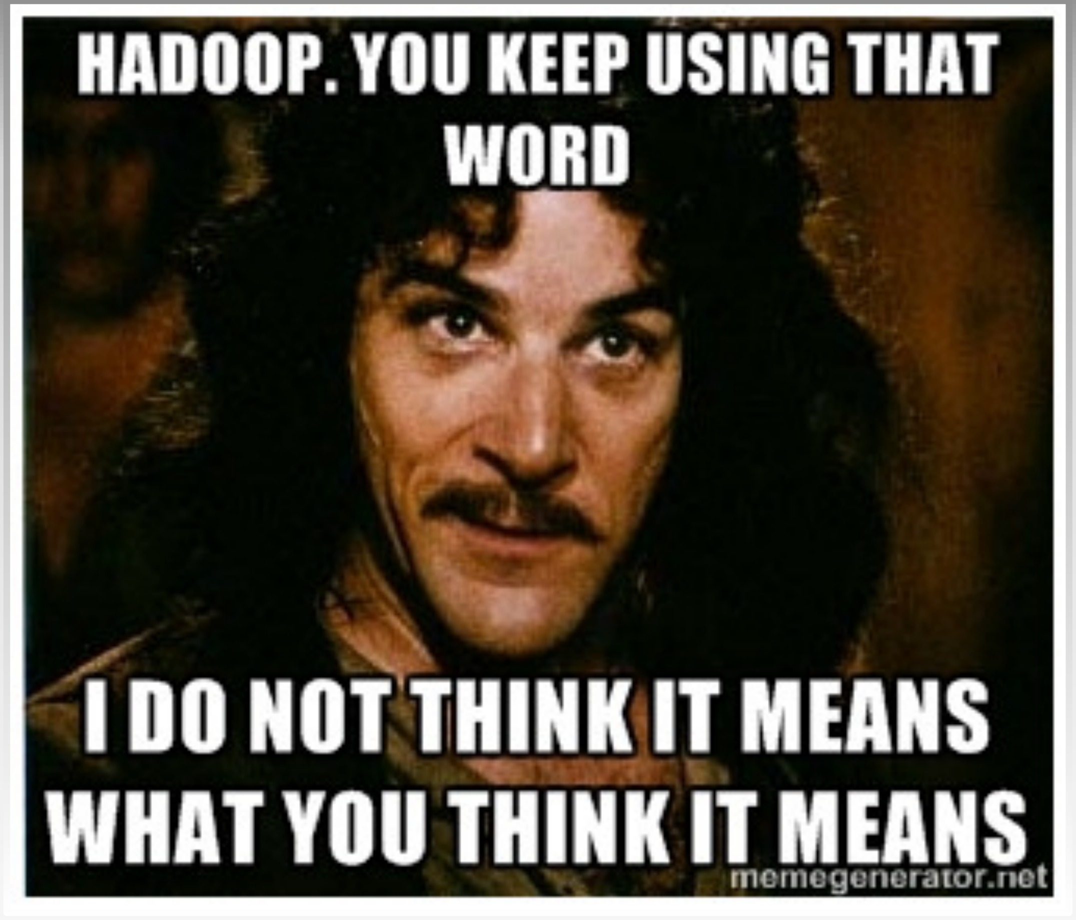

Addison Wesley Data & Analytics Series
HADOOP 2
QUICK-START GUIDE
LEARN THE ESSENTIALS OF BIG DATA COMPUTING IN THE APACHE HADOOP 2 ECOSYSTEM
DOUGLAS EADLINE


Addison Wesley Data & Analytics Series
PRACTICAL DATA SCIENCE with Hadoop and Spark
Designing and Building Effective Analytics at Scale
OFER MENDELEVITCH
CASEY STELLA
DOUGLAS EADLINE

Segment 1

Why is Hadoop Such a Big Deal?

Hadoop is the next Big Thing!


HADOOP. YOU KEEP USING THAT WORD
I DO NOT THINK IT MEANS WHAT YOU THINK IT MEANS
memegenerator.net

Data Volume – Let’s Call It a Lake

Data Velocity – Let’s Call It a Waterfall

Data Variation – Let’s Call it Recreation

What is Big Data?
(besides a marketing buzzword)

- The Internet helped create this situation
- Three V's (Volume, Velocity, Variability)
- Data that are large (lake), growing fast (waterfall), unstructured (recreation) - not all may apply.
- May not fit in a "relational model and method"
- Can Include: video, audio, photos, system/web logs, click trails, IoT, text messages/email/tweets, documents, books, research data, stock transactions, customer data, public records, human genome, and many others.

Why All the Fuss about Hadoop?

- We hear about Hadoop scale
- We hear about Hadoop large file systems
- We hear about Hadoop parallel processing – something called "MapReduce"

At Yahoo, 100,000 CPUs in more than 40,000 computers running Hadoop, 455 petabytes of storage (10E15 Bytes)

Is BIG DATA Really That big

Two analytics clusters at Yahoo and Microsoft, median job input sizes are under 14GB and 90% of jobs on a Facebook cluster have input sizes under 100 GB. ("Nobody ever got fired for using Hadoop on a cluster," HotCDP 2012)

What Is Really Going on Here?

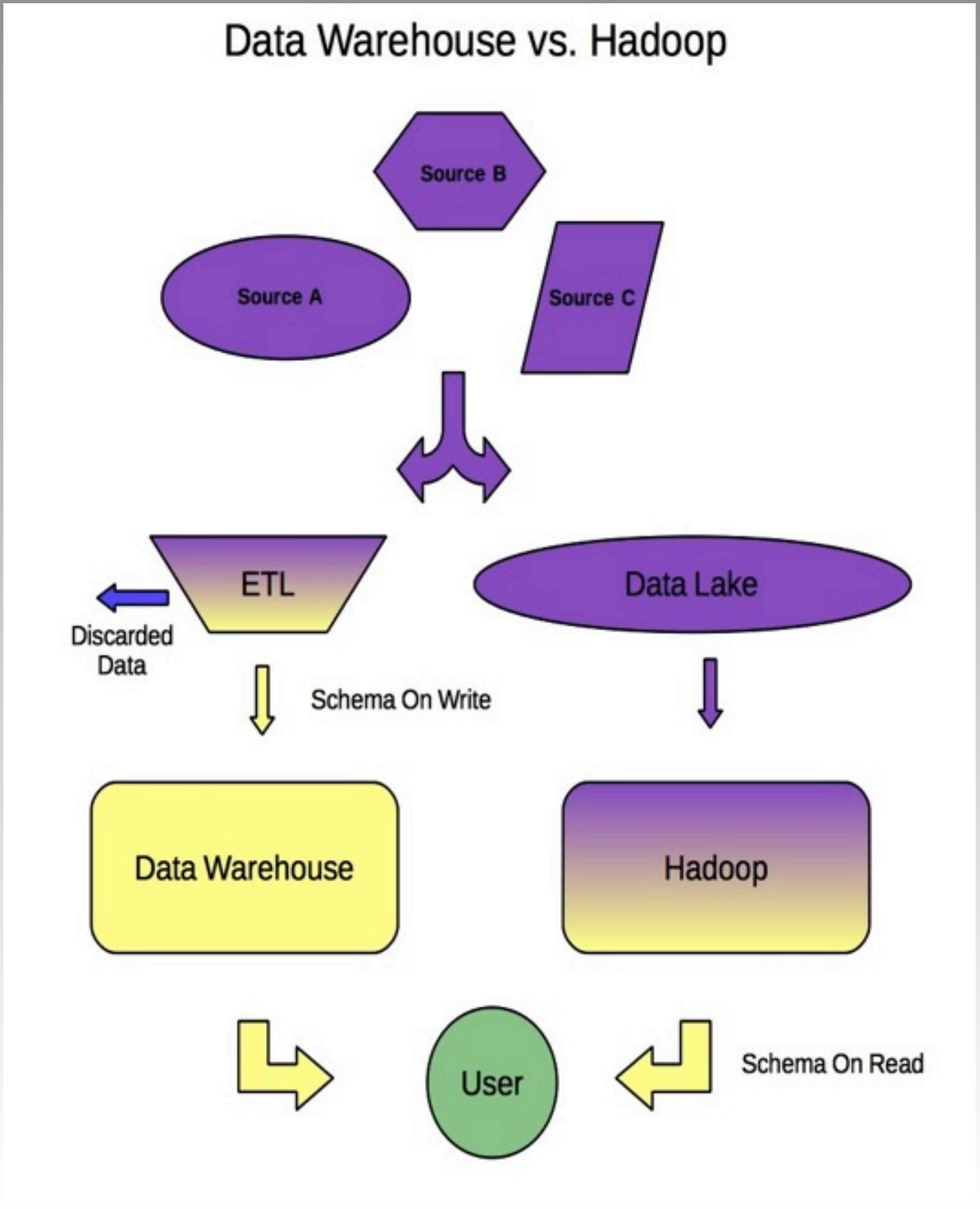
- Hadoop is not necessarily about scale
  (although it helps to have a scalable solution that can handle large capacities and large capabilities)
- Big Data is not necessarily about volume
  (although volume and velocity are increasing, so is hardware)

The “Hadoop Data Lake“ represents a fundamentally new way to manage data.

Hadoop Data Lake

Data Warehouse applies "schema on write" and has an Extract, Transform, and Load (ETL) step.

Hadoop applies "schema on read" and the ETL step is part of processing. All input data are placed in the lake in raw form.



Hadoop History

- Developed by Doug Cutting and Michael J. Cafarella at Yahoo
- Based On Google MapReduce and Bigtable technology
- Designed to handle large amounts of data and be robust
- Donated to Apache Software Foundation in 2006 by Yahoo.
- Some areas where Hadoop is used:
  - Social media
  - Retail
  - Financial services
  - Web Search
  - Government
  - Everywhere there is large amounts of unstructured data
- Prominent users:
  Yahoo! Facebook, Amazon, EBay, American Airlines, The New York Times, Federal Reserve Board of Governors, Chevron, IBM, and many others

Defining Hadoop (Version 2)

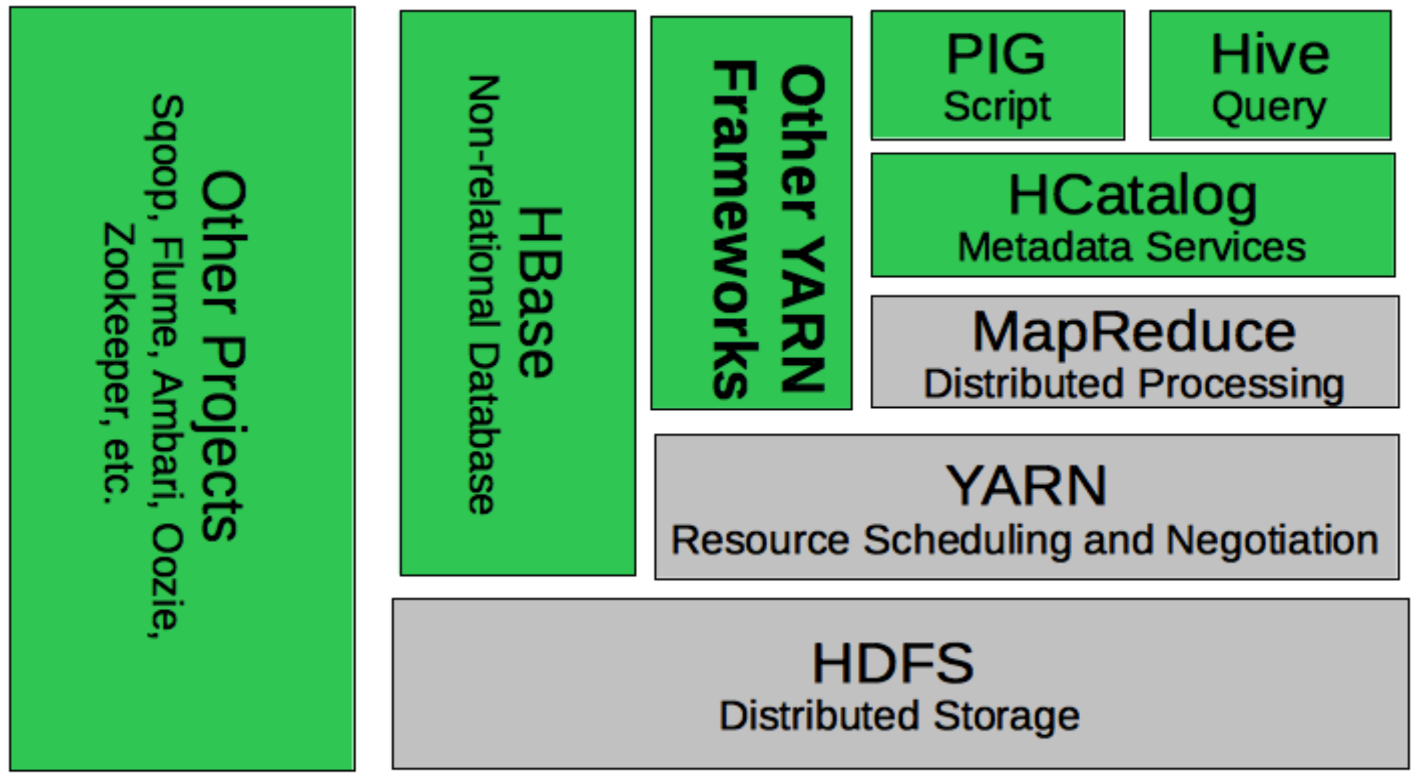
A PLATFORM for data lake analysis that supports software tools, libraries, and methodologies

- Core and most tools are open source (Apache License)
- Large unstructured data sets (Petabytes)
- Written in Java, but not exclusively a Java platform
- Primarily GNU/Linux, Windows versions available
- Scalable from single server to thousands of machines
- Runs on commodity hardware and in the cloud
- Application level fault tolerance possible
- Multiple processing (MapReduce) models and libraries

What is Hadoop is NOT

- A replacement for data warehouses
- The only data analytics tool (i.e. Hadoop is a platform for data analytics at scale, in real-time, on variable data)
- A single application
- As complicated as it used to be
- The solution to every analytics or data problem
- A monolithic MapReduce engine

Hadoop Components



Hadoop Core Components

- **HDFS – Hadoop Distributed File System.** Designed to be a fault tolerant streaming file system. Default block size is 64 MB vs. 4 KB for Linux ext4. Runs on commodity servers, master "NameNode" and multiple "DataNodes." Data are replicated on multiple servers.
- **YARN – Yet Another Resource Negotiator.** Master scheduler and resource allocator for the entire Hadoop cluster. User jobs ask YARN for resources (containers) and data locality. Provides dynamic resource allocation(run-time). Runs on commodity servers, master "ResouceManager" and multiple "NodeManagers."
- **MapReduce – YARN Application.** Provides MapReduce processing.
- **Cluster servers (nodes) are usually both DataNodes and NodeManagers (move processing to data)**

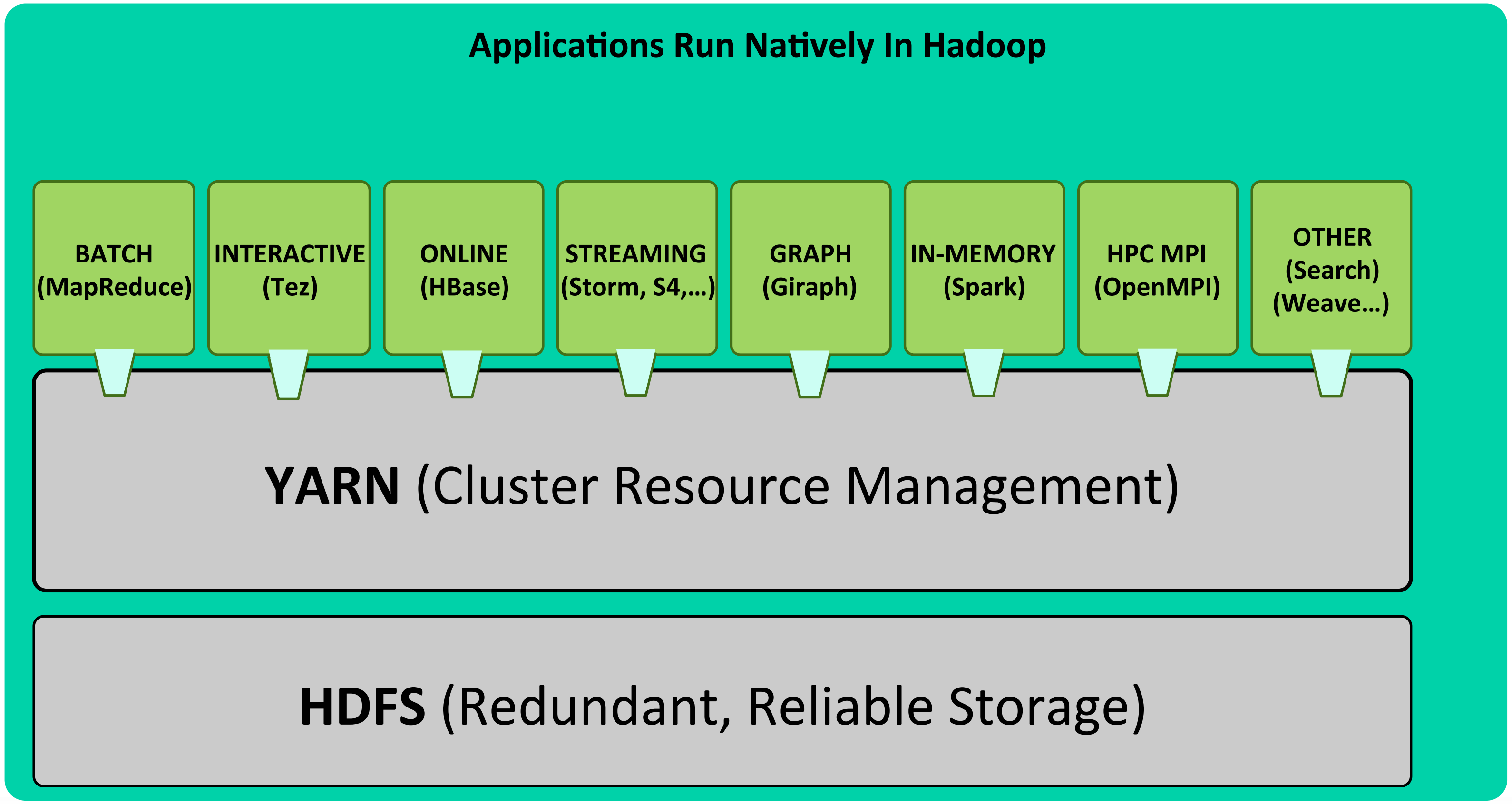
Hadoop Component List

- **Apache Pig** is a high-level language for creating MapReduce programs used with Hadoop.
- **Apache Hive** is a data warehouse infrastructure built on top of Hadoop for providing data summarization, ad-hoc queries, and the analysis of large datasets using a SQL-like language called HiveQL.
- **Apache HCatalog** Apache HCatalog is a table and storage management service for data created using Apache Hadoop.
- **Apache HBase** HBase (Hadoop Database) is a distributed and scalable, column oriented database. Similar to Google Big Table.
- **Apache ZooKeeper** is a centralized service used by applications for maintaining configuration, health, etc. on and between nodes.

More Hadoop Components

- **Apache Ambari** is a tool for provisioning, managing, and monitoring Apache Hadoop clusters.
- **Apache Oozie** is a workflow/coordination system to manage multistage Hadoop jobs.
- **Apache Sqoop** is a tool designed for efficiently transferring bulk data between Hadoop and relational databases.
- **Apache Flume** is a distributed service for efficiently collecting, aggregating, and moving large amounts of log data.

Application Frameworks



Questions ?

5 Minute BREAK

Segment 2

Hadoop Distributed File System (HDFS)

Why Do We Need a New File System?

- Traditional File Systems are not designed for large files and fast streaming reads
- Write once/read many use cases
- Low coherency/concurrency overhead (no random file writes, one user writing at a time)
- We want to move computation to data
- Converged data-computation model – Slice data file and place it on multiple computational nodes/ servers

HDFS Trade-offs vs. Traditional FS

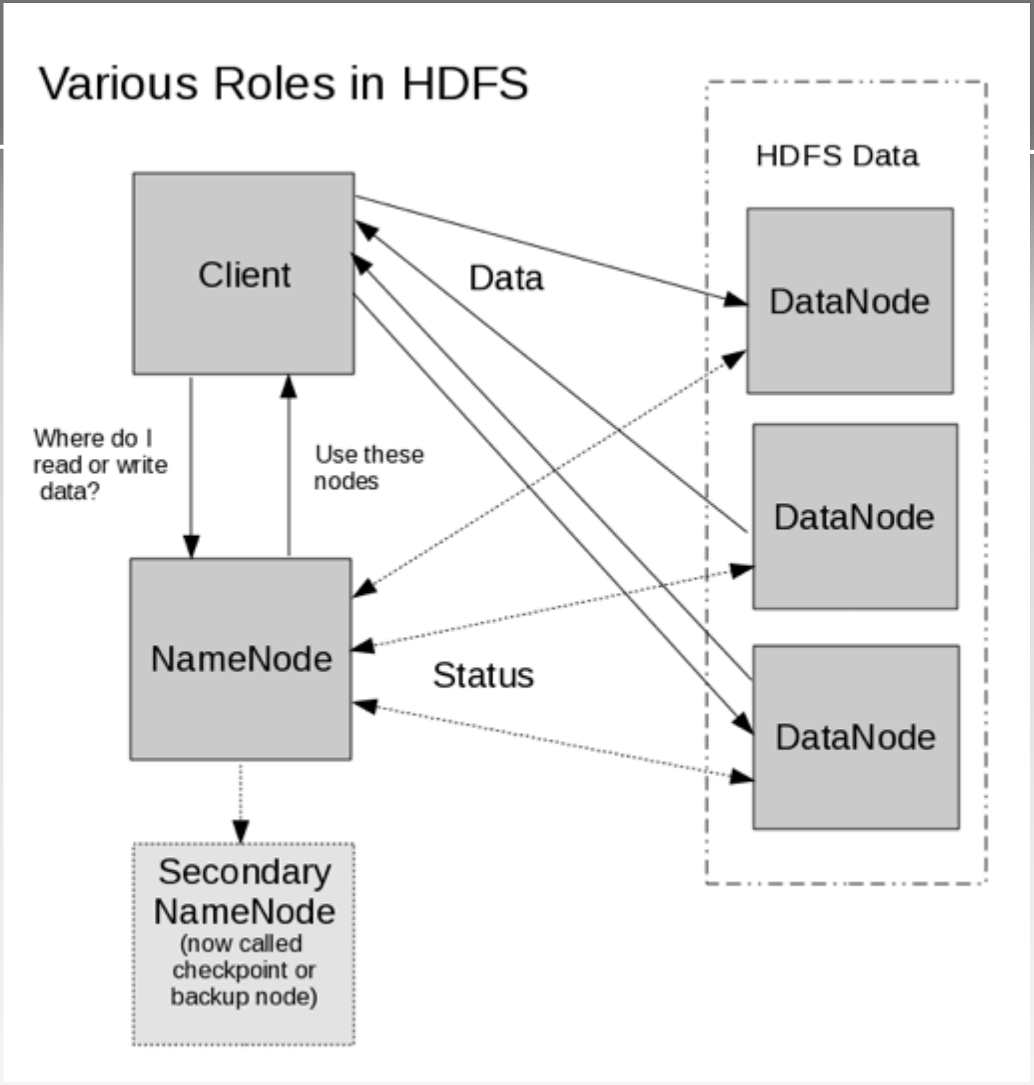
- Optimized for streaming reads makes it poor for random seeks
- Default block size is 64 MB (4K typical in other FS)
- Write once file system
- No local cache or RAID needed
- With hardware failure comes reduced performance, but still preserves data
- A specialized streaming file system, not designed for general use

NameNode and DataNodes

- Master/Slave model
- NameNode - metadata server or "data traffic cop”
- NameNode metadata in memory for performance
- Single Namespace - managed by the NameNode
- DataNodes - where the data live
- Secondary NameNode - checkpoints NameNode metadata, but not failover

HDFS Roles

NameNode keeps track of metadata and uses DataNodes for storage

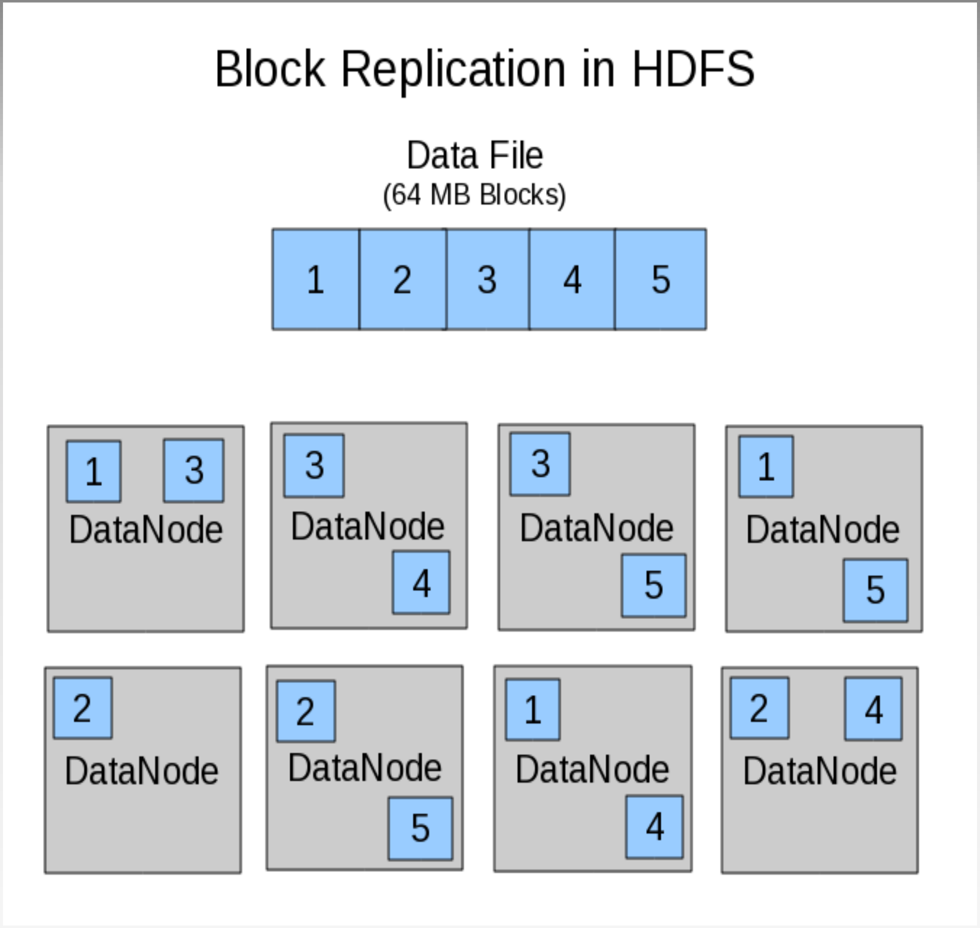


The HDFS File System Namespace

- Traditional hierarchical file system with files and directories
- Users can create, remove, move, rename, copy
- No <u>random read or write</u> only appends
- Must access HDFS through Hadoop layer, not directly on DataNodes (or NFSv3)
- Features: High Availability (multiple NameServers), Federation (spread namespace across NameServers), Snapshots (instant read-only point-in-time copies), NFSv3 access

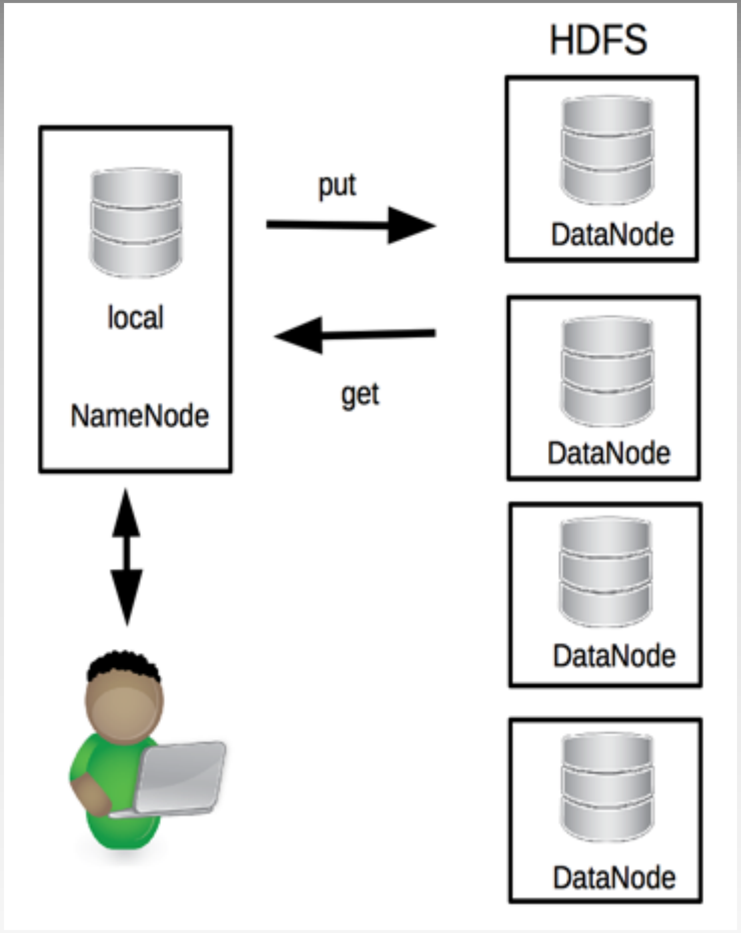
Replication In HDFS

- Data are replicated (tunable, default is 3 copies)
- Replication is used for redundancy
- Replication is used for fast access



How the User “Sees” HDFS

- HDFS is a separate file system from the host machine
- Data must be moved to (put) and from (get) HDFS
- Hadoop processing happens in HDFS



Questions ?

Segment 3

Hadoop MapReduce Framework

What Is Map Reduce?

Map Reduce Is a Simple Algorithm

(and can run in parallel due to no side-effects)

```
grep something | wc –l
```

- Distinct steps and one-way communication
- Map then Reduce
- Uses (key, value) pair
- Works great in many cases (even for some HPC applications), but it is not the only algorithm for Hadoop version 2. It is the basis for many tools including Pig and Hive.

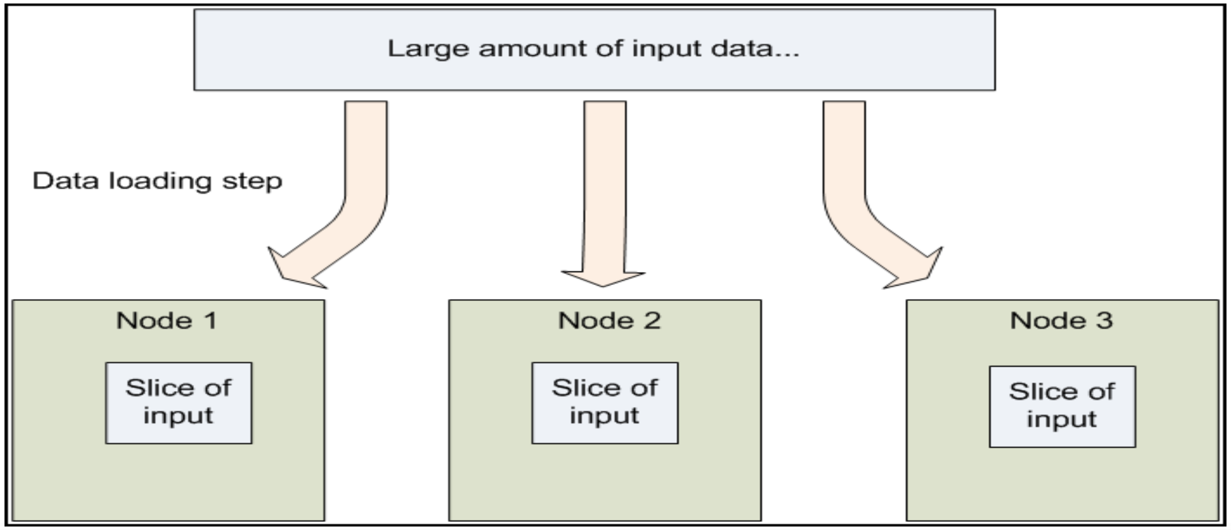
BASEMENT SUPERCOMPUTING

MapReduce Design Principles

- Moving computation is cheaper than moving data
- Hardware will fail, manage it in software
- Hide execution details from the user
- Optimize the file system for big blocks and streaming data access
- Simple file system coherency model

Hadoop MapReduce Big Picture

1. Data are sliced and stored in a distributed and redundant parallel file system (Hadoop File System or HDFS)
2. Queries are applied (mapped) to each distributed data set

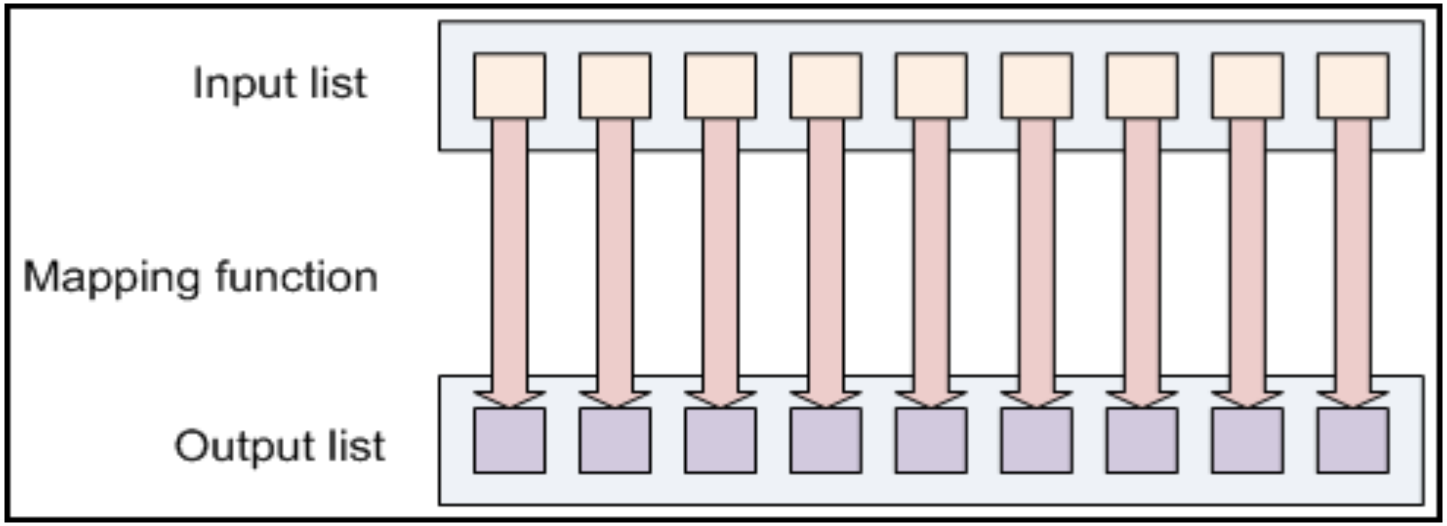


BASEMENT SUPERCOMPUTING

Map Step

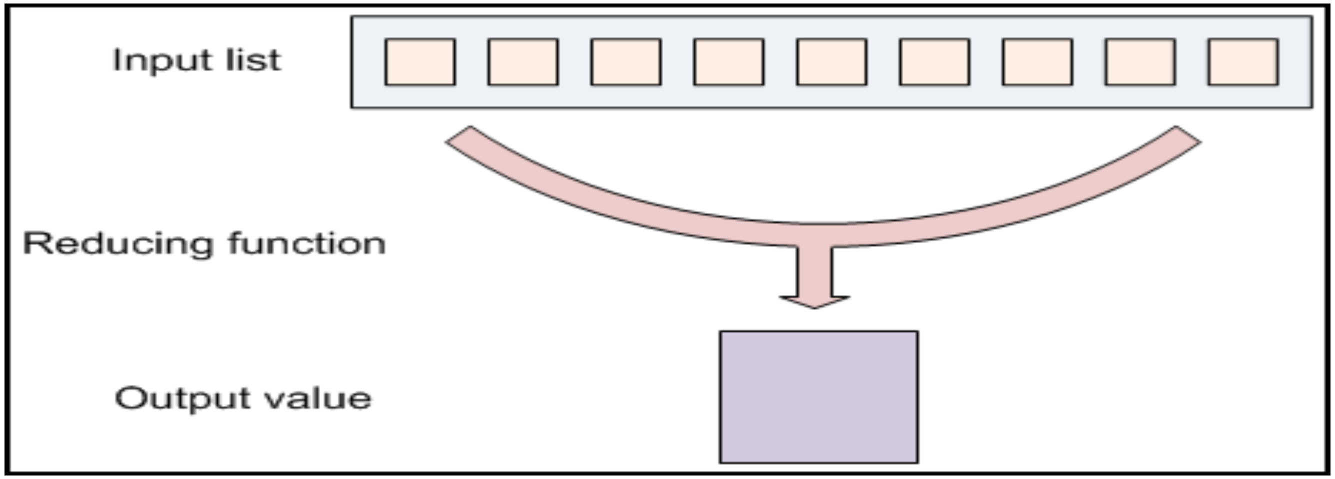
Step 1: Files loaded in to HDFS (performed one time)
    Example: load "War and Peace"

Step 2: User query is "Mapped" to all nodes
    Example: How many times is the name *Kutuzov* mentioned in this slice?

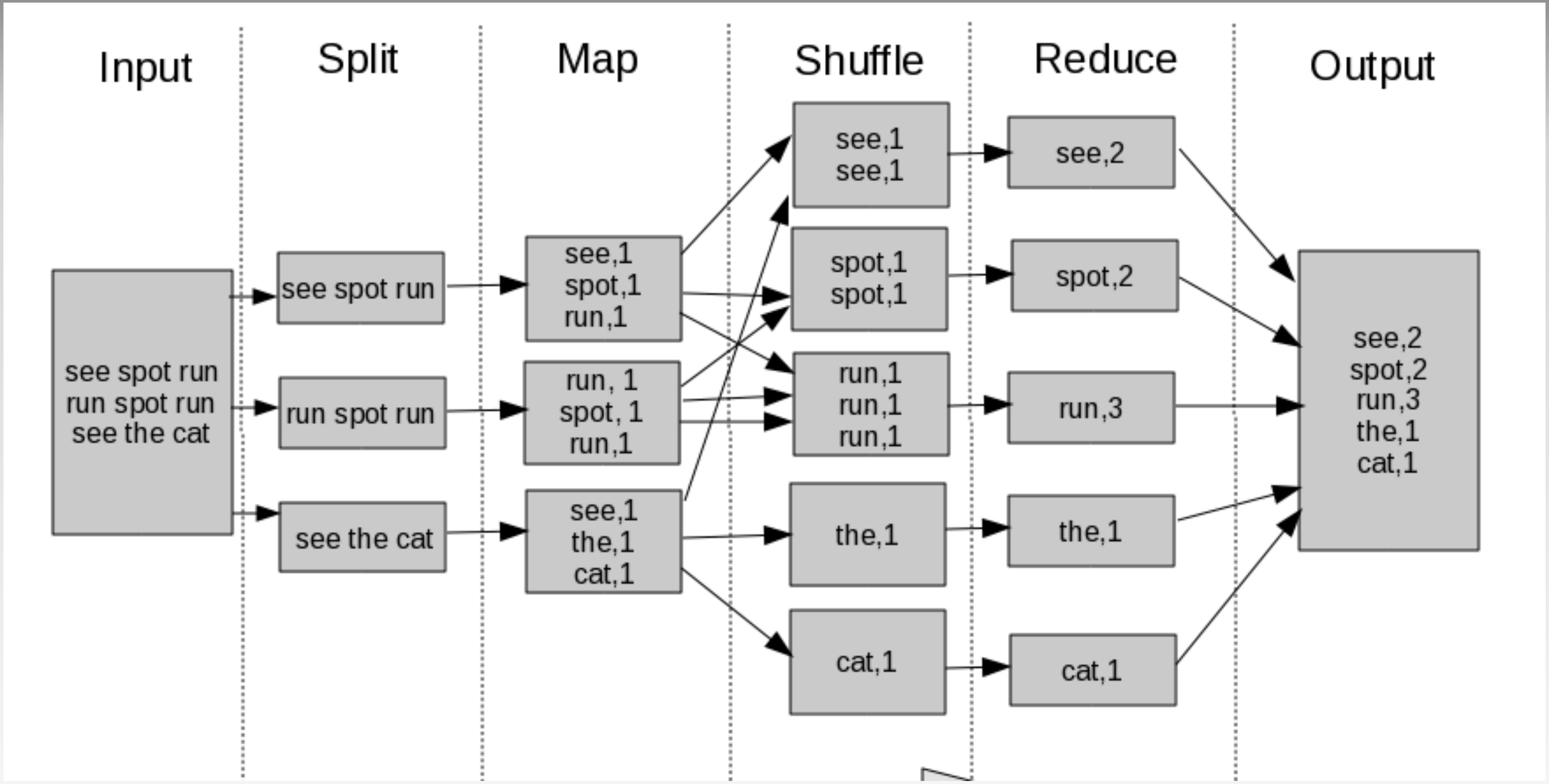


Reduce Step

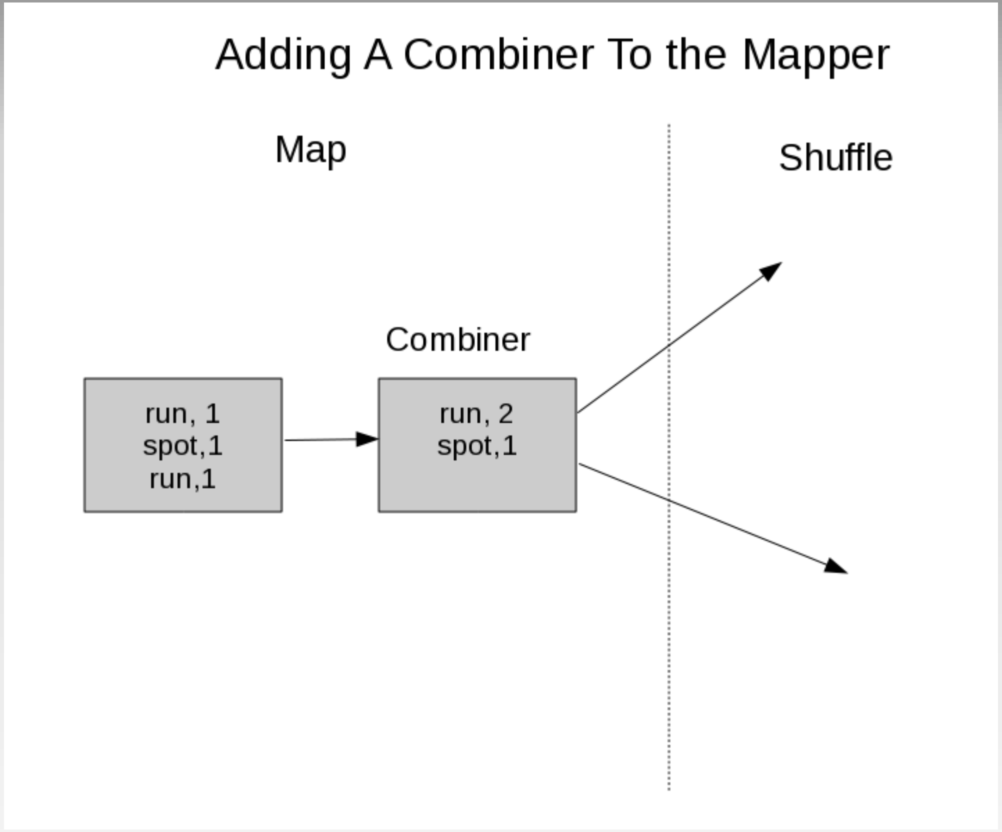
Step 3: Reduce the Output list.
Example: Collect and sum of the counts for Kutuzov from each input list (was the output list of the Map step) Answer is a single number.



MapReduce In Parallel



MapReduce Improvements



Programming MapReduce

- Natively MapReduce Uses Java
- Other languages can use a Streaming Interface (stdin, stdout, and text only)
- C++ Pipes Interface
- Pig and Hive in (batch mode or with Tez acceleration)
- MapReduce applications can scale with no change to application
- Computation moved to the data in HDFS
- There is no need to manage side-effects or process state
- HW faults are handled by automatic restart of tasks, transparent to the application

Advantages of MapReduce

- Functional approach (no change to original data)
- Single one way communication path
- Provides the Following Features:
  - Highly scalable (same user code)
  - Easily Managed Workflow
  - Fault tolerant
- MapReduce is a powerful paradigm for solving many problems. Often referred to a "Data Parallel“ or Single Instruction Multiple Data (SIMD) problem

Questions ?

5 Minute
BREAK

Segment 4

Making life Easier: Spark

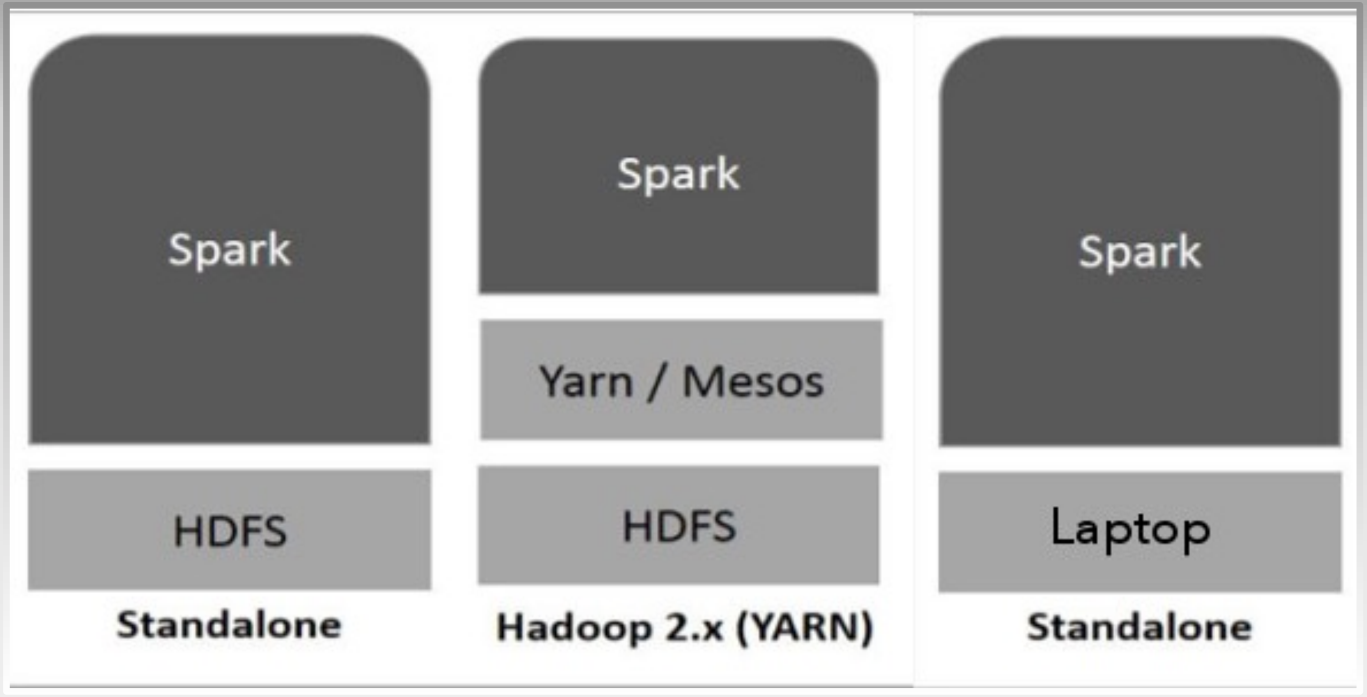
Spark (and Hadoop)

- Developed in 2009 in UC Berkeley's AMPLab by Matei Zaharia.
- It was Open Sourced in 2010 under a BSD license.
- It was donated to Apache software foundation in 2013
- Similar to Hadoop MapReduce, but is independent project

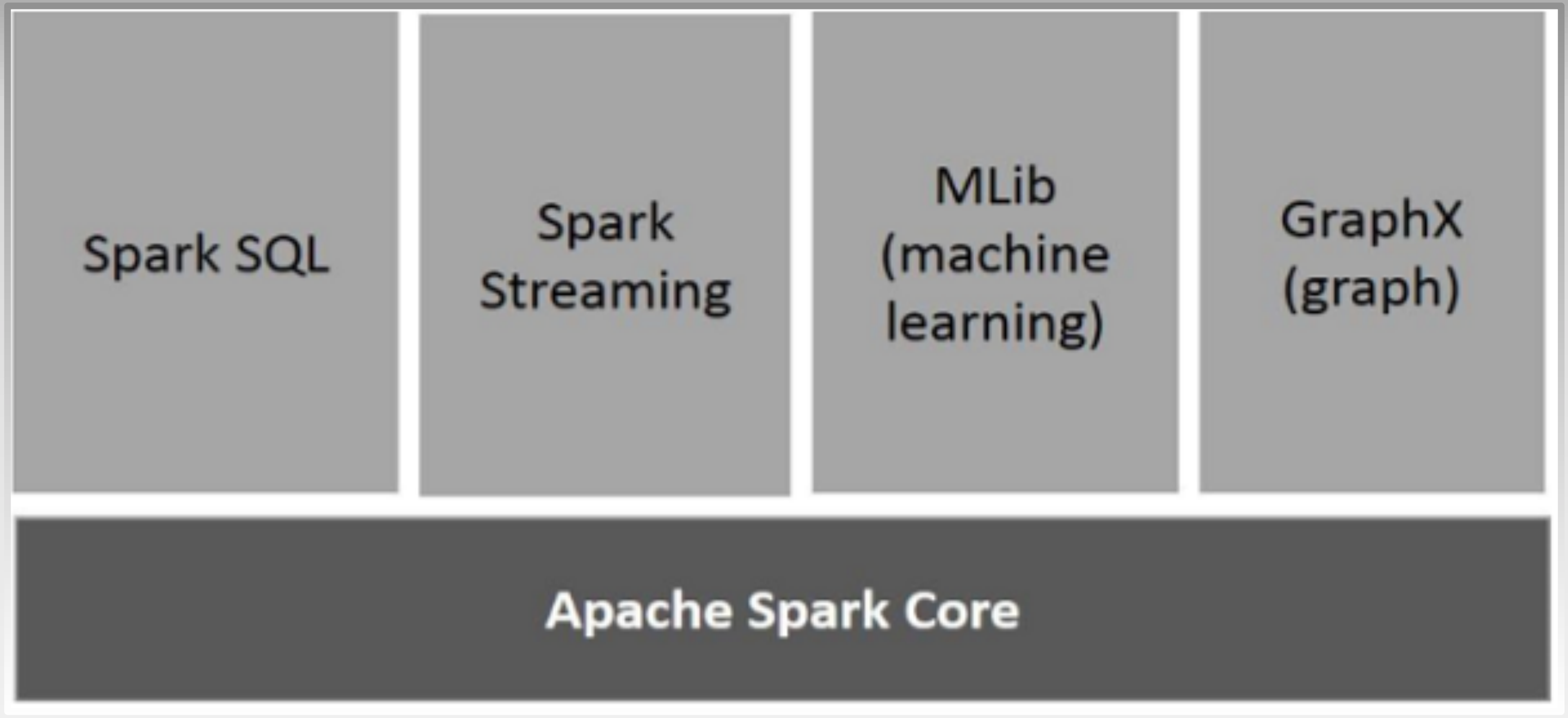
What is So Special About Spark?

- **Speed** – Spark keeps intermediate results in memory. Many analytics jobs consist of stages, traditional MapReduce writes results to disk between stages. Spark stores the intermediate processing data in memory.
- **Supports Multiple Languages** – Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark includes 80 high-level operators for interactive querying.
- **Advanced Analytics** – Spark not only supports “Map” and “Reduce.” It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms. It is higher level than native MapReduce, it can be used in place of Hive and/or Pig

Spark Deployment



Components of Spark



Spark Components

- **Apache Spark Core -** Spark Core is the underlying general execution engine for spark
- **Spark SQL -** Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.
- **Spark Streaming -** Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics.
- **MLlib (Machine Learning Library) -** MLlib is a distributed machine learning framework
- **GraphX -** GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation.

Spark RDDs

- Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark.
- It is an immutable distributed collection of objects.
- Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster.
- RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Spark Operations: Transformations and Actions

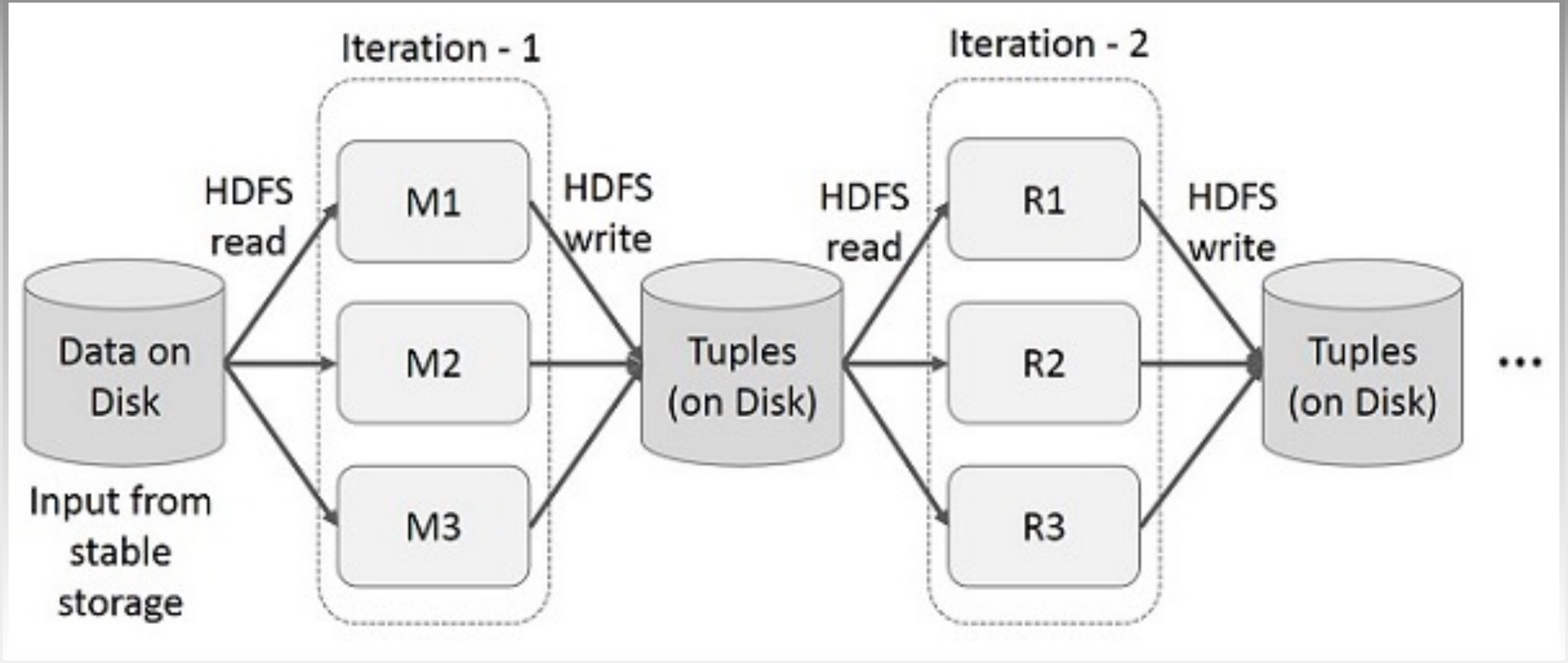
- **RDD Transformations** return a pointer to new RDD . The original RDD cannot be changed. Spark is lazy, so nothing will be executed unless a transformation or action is called.

  An RDD transformation is not a set of data, but is a step in a program (might be the only step) telling Spark how to get data and what to do with it.
- **RDD Actions** return values (e.g. collect, count, take, save-as).

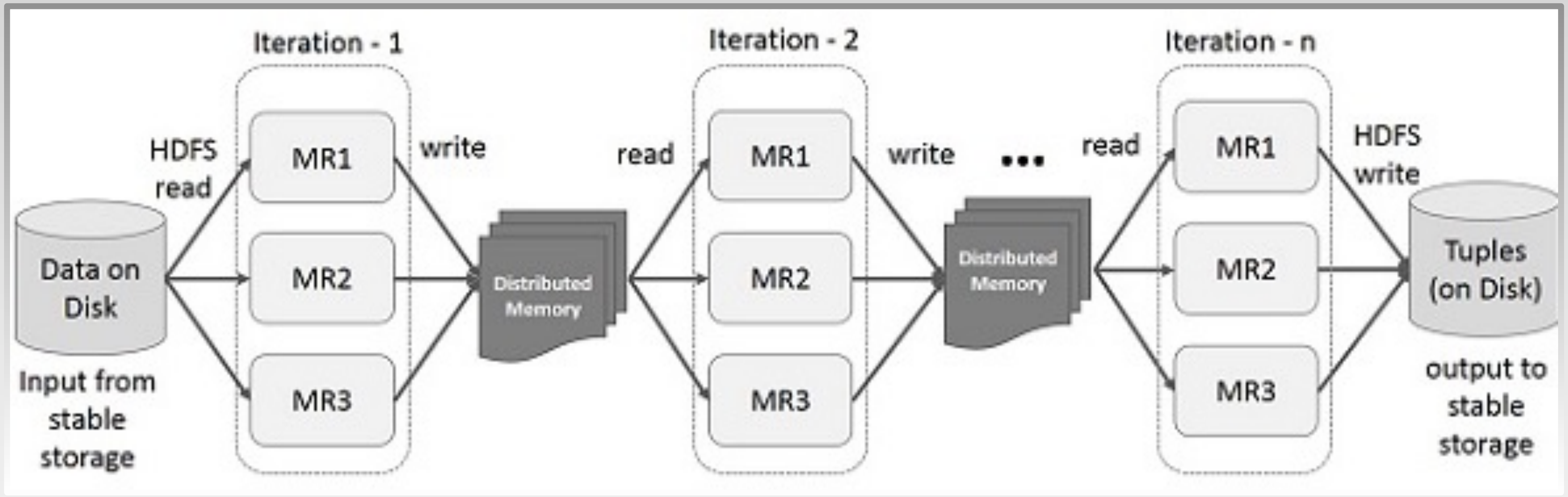
Spark RDD vs. Data Frame

- An **RDD** is blind structure partitioned across the nodes of the cluster and provides many transformation methods, such as map(), filter(), and reduce(). Each of these methods results in a new RDD representing the transformed data.
- The **DataFrame** introduces the concept of a schema to describe the data (named columns), allowing Spark to manage and optimize computation across nodes. Conceptually equivalent to a table in a relational database or a R/Python Dataframe.
- Think of a **DataFrame** as a distributed database table and an **RDD** as distributed raw data. (new: DataSets)

Iterative Operations Using Traditional MapReduce (Batch Mode)



Iterative Operations Using Spark RDD



In Reality, However

- Hadoop MapReduce has been optimized using Tez (keeps tuples in memory) Pig and Hive SQL have become more interactive (like Spark) and less like batch jobs.
- Spark uses “in memory” computing, but if it runs out of memory then intermediate results will spill to disk. And if the job does not fit in memory, then traditional Hadoop MapReduce can be used.

Spark Code Example (Pi estimation)

```
from pyspark import SparkContext
from numpy import random
n=5000000

def sample(p):
    x, y = random.random(), random.random()
    return 1 if x*x + y*y < 1 else 0

count = sc.parallelize(xrange(0,n)).map(sample) \
.reduce(lambda a, b: a + b)

print "Pi is roughly %f" % (4.0 * count / n)
```

MapReduce Java Pi Estimation

```
    /**
      * Mapper class for Pi estimation. /**
      * Mapper class for Pi estimation.
      * Generate points in a unit square
      * and then count points inside/outside of the inscribed circle of the squa
      */
     public static class QmcMapper extends
         Mapper<LongWritable, LongWritable, BooleanWritable, LongWritable> {

       /** Map method.
        * @param offset samples starting from the (offset+1)th sample.
        * @param size the number of samples for this map
        * @param context output {ture->numInside, false->numOutside}
        */
       public void map(LongWritable offset,
                       LongWritable size,
                       Context context)
           throws IOException, InterruptedException {

         final HaltonSequence haltonsequence = new HaltonSequence(offset.get());
         long numInside = 0L;
         long numOutside = 0L;

         for(long i = 0; i < size.get(); ) {
           //generate points in a unit square
           final double[] point = haltonsequence.nextPoint();

           //count points inside/outside of the inscribed circle of the square
           final double x = point[0] - 0.5;
           final double y = point[1] - 0.5;
           if (x*x + y*y > 0.25) {
             numOutside++;
           } else {
             numInside++;
           }

           //report status
           i++;
           if (i % 1000 == 0) {
             context.setStatus("Generated " + i + " samples.");
           }
         }

         //output map results
         context.write(new BooleanWritable(true), new LongWritable(numInside));
         context.write(new BooleanWritable(false), new LongWritable(numOutside))
       }
     }

     /**
      * Reducer class for Pi estimation.
      * Accumulate points inside/outside results from the mappers.
      */
     public static class QmcReducer extends
         Reducer<BooleanWritable, LongWritable, WritableComparable<?>, Writable>

       private long numInside = 0;
```

```
       /**
        * Reduce task done, write output to a file.
        */
       @Override
       public void cleanup(Context context) throws IOException {
         //write output to a file
         Path outDir = new Path(TMP_DIR, "out");
         Path outFile = new Path(outDir, "reduce-out");
         Configuration conf = context.getConfiguration();
         FileSystem fileSys = FileSystem.get(conf);
         SequenceFile.Writer writer = SequenceFile.createWriter(fileSys, conf,
             outFile, LongWritable.class, LongWritable.class,
             CompressionType.NONE);
         writer.append(new LongWritable(numInside), new LongWritable(numOutside));
         writer.close();
       }
     }
      * Generate points in a unit square
      * and then count points inside/outside of the inscribed circle of the square.
      */
     public static class QmcMapper extends
         Mapper<LongWritable, LongWritable, BooleanWritable, LongWritable> {

       /** Map method.
        * @param offset samples starting from the (offset+1)th sample.
        * @param size the number of samples for this map
        * @param context output {ture->numInside, false->numOutside}
        */
       public void map(LongWritable offset,
                       LongWritable size,
                       Context context)
           throws IOException, InterruptedException {

         final HaltonSequence haltonsequence = new HaltonSequence(offset.get());
         long numInside = 0L;
         long numOutside = 0L;

         for(long i = 0; i < size.get(); ) {
           //generate points in a unit square
           final double[] point = haltonsequence.nextPoint();

           //count points inside/outside of the inscribed circle of the square
           final double x = point[0] - 0.5;
           final double y = point[1] - 0.5;
           if (x*x + y*y > 0.25) {
             numOutside++;
           } else {
             numInside++;
```

• . . .

Questions ?

Segment 5

Real World Applications/ Wrap-up

BASEMENT SUPERCOMPUTING

Hadoop Use Cases - Volume

A leading **retail bank** is using Hadoop to validate data accuracy and quality to comply with federal regulations like Dodd-Frank. The platform allows analyzing trillions of records which currently result in approximately one terabyte per month of reports. Other solutions were not able to provide a complete picture of all the data.

Hadoop Use Cases - Velocity

US Xpress, one of the largest trucking companies in the US, is using Hadoop to store sensor data (100s of data points and geo data) from thousands of trucks. The intelligence they mine out of this, saves them $6 million year in fuel cost alone. Hadoop allows storing enormous amount of sensor data and querying/ joining this data with other data sets.

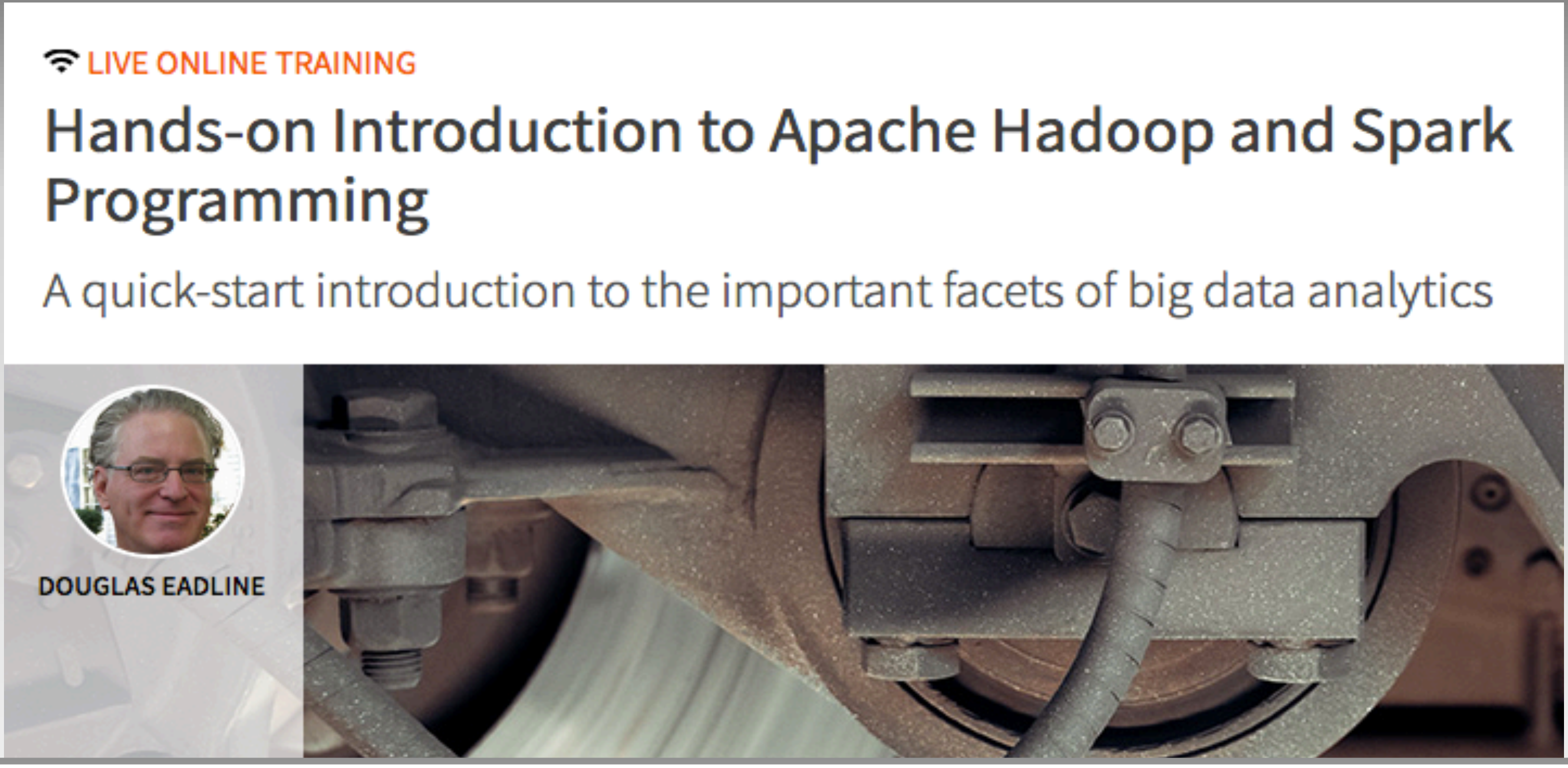
Hadoop Use Cases - Variability

NextBio is using Hadoop MapReduce and HBase to process multi-terabyte data sets of human genome data. Processing wasn't feasible using traditional databases like MySQL.

Hadoop/Spark Take Always

- Hadoop is a Big Data Analytics PLATFORM and File Systems (HDFS).
- Hadoop is an ECOSYSTEM of tools and applications.
- DATA LAKES are a new way to mange VOLUME, VELOCITY, and VARIABILITY.
- SPARK is a powerful language for data analytics.
- MAP-REDUCE is core part of Hadoop, but not the only part.
- Do not need a big cluster to try Hadoop/Spark!

Next Steps – Hands-On



LIVE ONLINE TRAINING

Hands-on Introduction to Apache Hadoop and Spark Programming

A quick-start introduction to the important facets of big data analytics

DOUGLAS EADLINE

https://www.safaribooksonline.com
Search for "Eadline"

BASEMENT SUPERCOMPUTING

Questions ?

Thank-You

BASEMENT SUPERCOMPUTING