Spark

Apache Storm

ECE 4150

Vijay Madisetti

Data Analytics using Spark, Storm, …

World is updating… its drivers

Relevance and Accuracy at Scale

HUMAN EXPERIENCE USED TO DRIVE HUMAN DECISIONS

THEN **DATA (USING BIG DATA ANALYTICS)** GUIDED HUMAN DECISIONS !

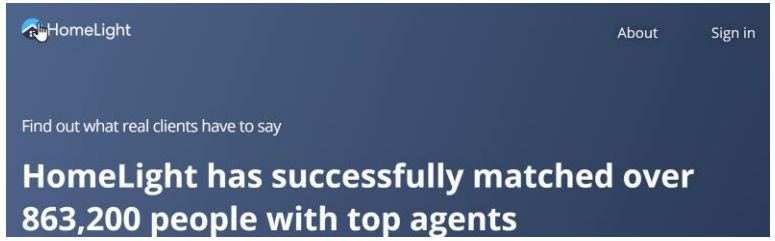
2015-2022

VKM, March 2024

NOW, AI IS USING BIG DATA TO **CREATE** DECISIONS FOR HUMANS

2023- Present

Case Study – **HomeLight**



HomeLight is here to help. Our technology crunches the numbers to determine which real estate agent or which instant offer company will get you the most money for your home.

We build products that put more power in your hands and make it easier to get the best outcome when you buy or sell a home.

Est. Annual Revenue	$60M
Est. Employees	250

Data-driven and objective

40M transactions and 1.2m agents

Unparalleled network

70,000 agents and 150+ cash buyers

Operating at scale

365,000+ happy clients and counting

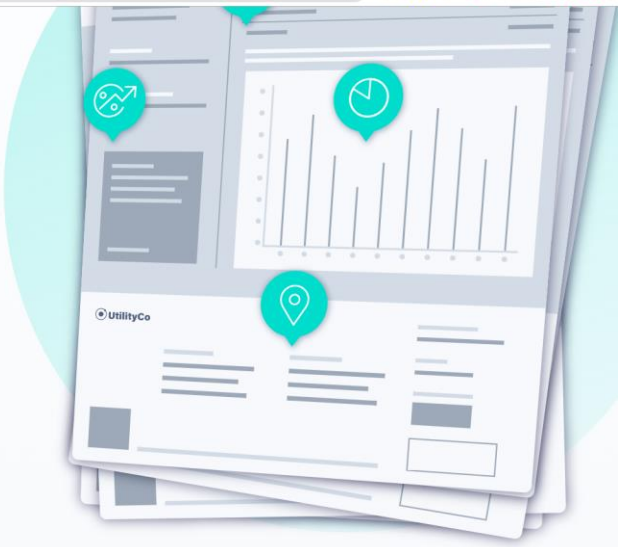
Example of Analytics Marketed as Products

urjanet.com/services/utility-bill-data/

Drive innovation and efficiency with Urjanet Utility Bill Data

The Urjanet Utility Data Platform connects to a global network of over 6,000 utility providers to deliver utility bill data directly into the application of your choosing. Whether you need historical or ongoing data, scheduled delivery or on-demand access, our platform is built to work for you.

GET A DEMO



From 1 to 400 Employees in 8 years (on Spring Street, Atlanta)

Urjanet's cloud-based, intelligent technology automatically aggregates the utility bill data you need from a growing network of more than 6,000 utility providers.

Granular insights

The Urjanet Utility Data Platform delivers every data point on the bill, along with the original bill image.

Flexible access

Get utility bill data on demand or on a set delivery schedule that makes sense for you. Whatever your needs are, we've got you covered.

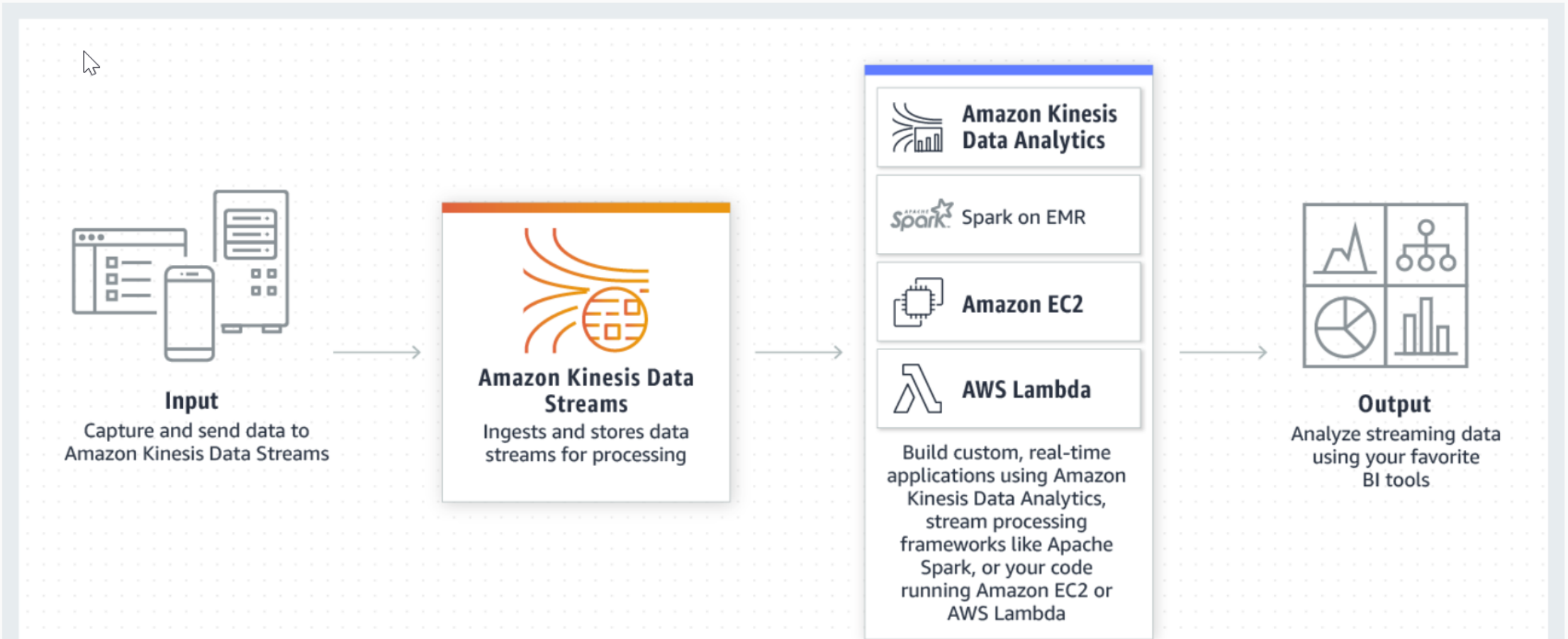
Powerful automation

Gone are the days of chasing down invoices or manually keying in data. Let our machine learning technology take the heavy lifting off your hands.

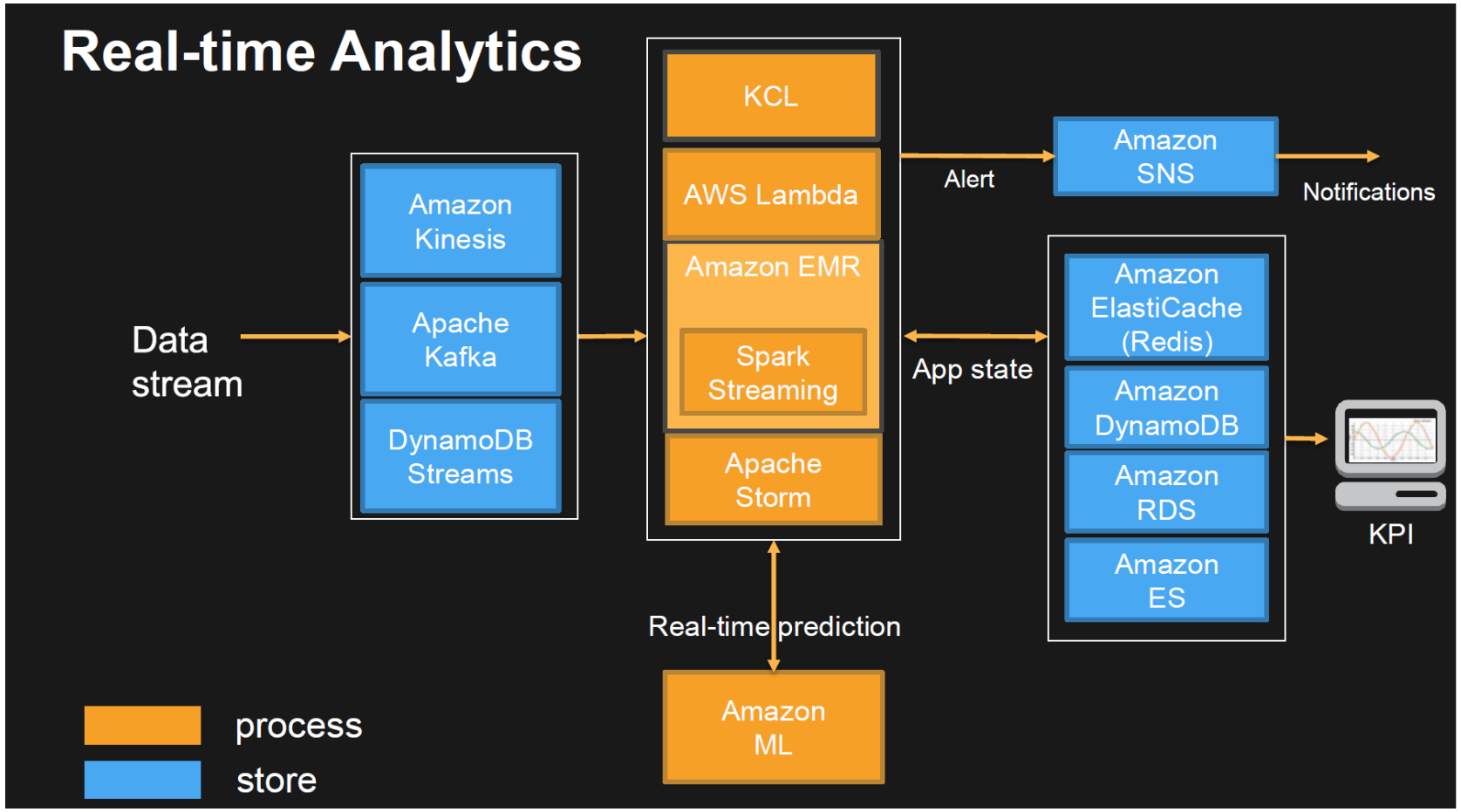
Secure and scalable

Our cloud-based platform grows with your business, upholding the highest standards of data security and encryption.

Amazon Kinesis Video Streams | Amazon Kinesis Data Streams | Amazon Kinesis Data Firehose | Amazon Kinesis Data Analytics

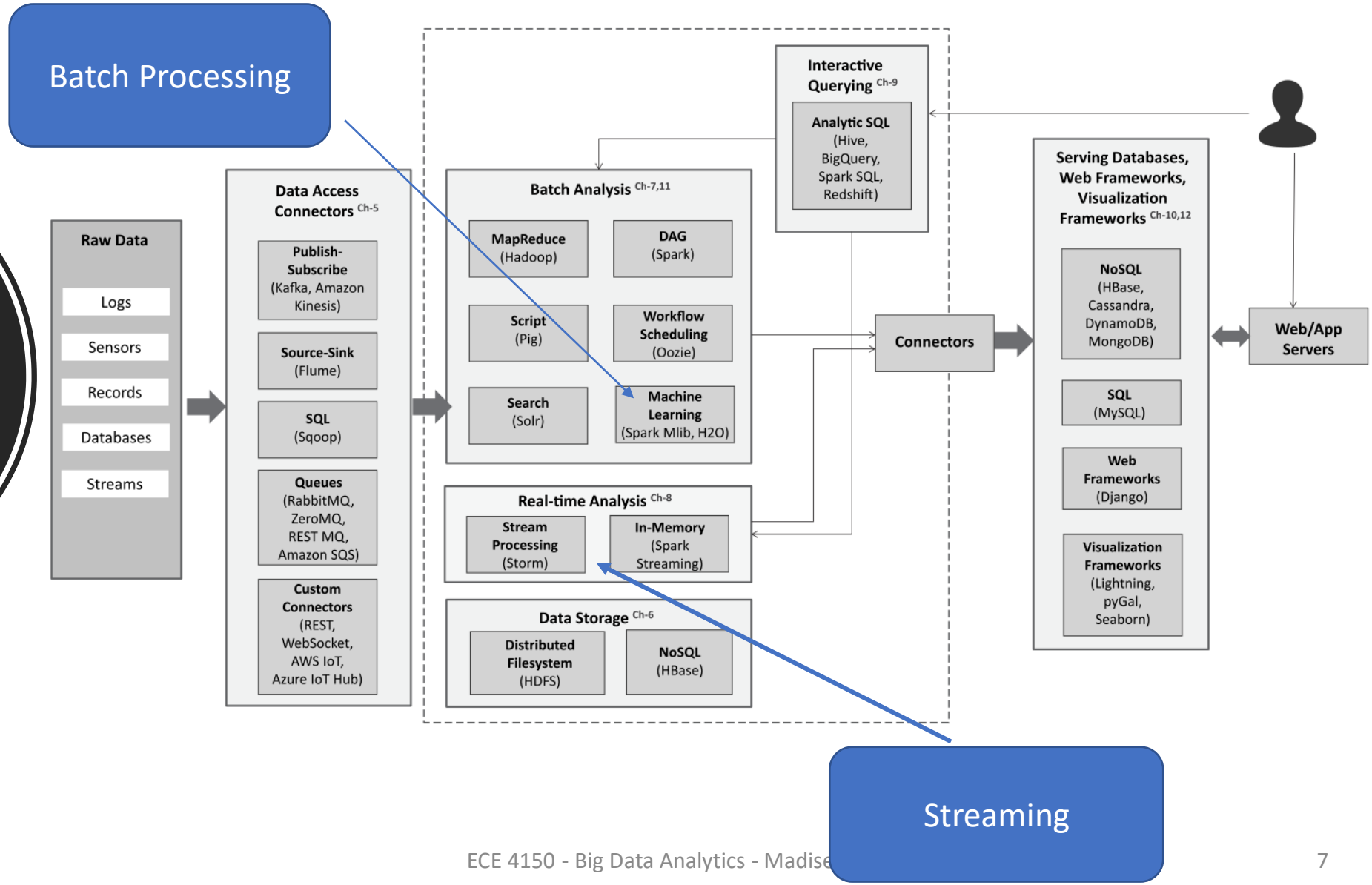


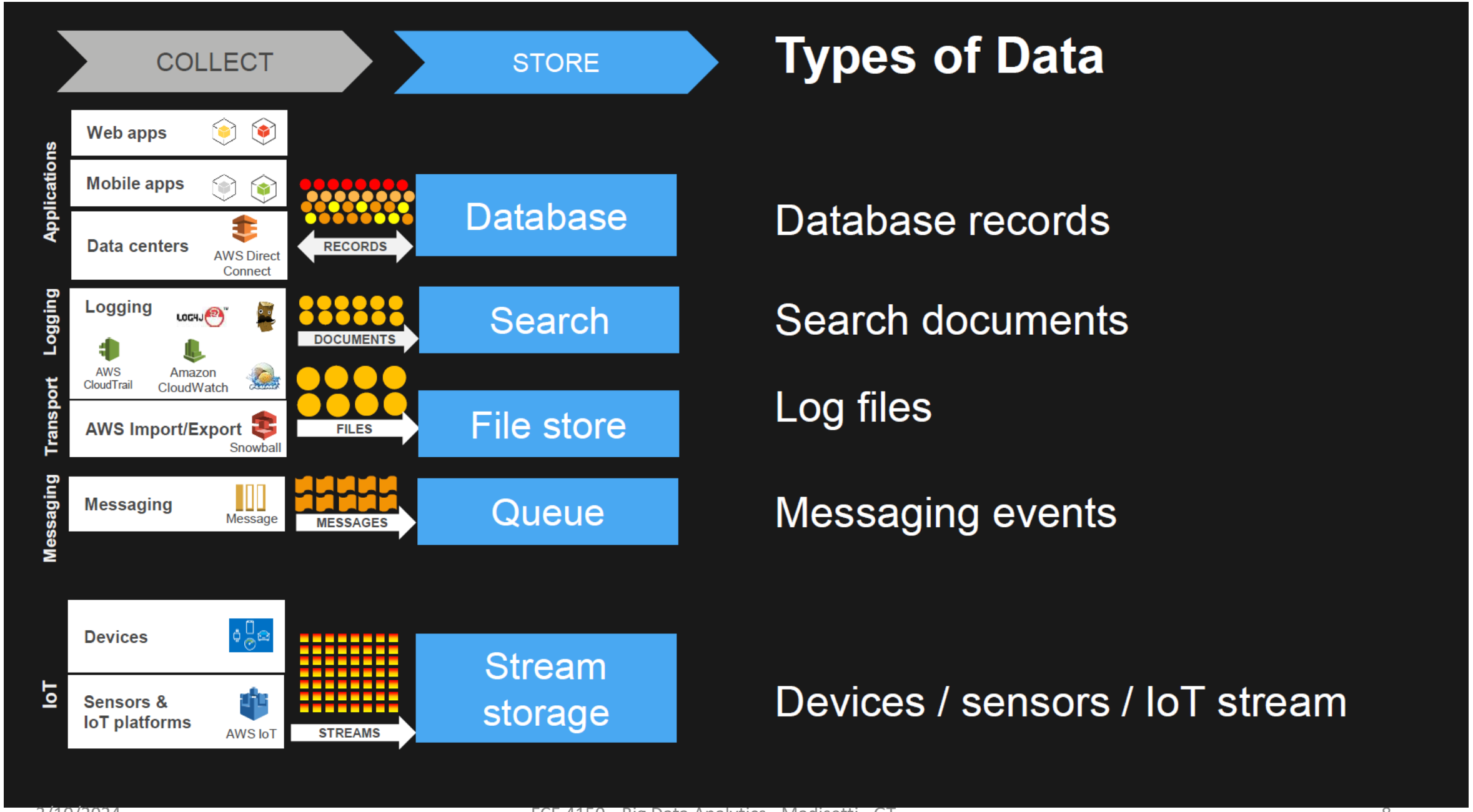
Summary - Streaming Implementation Architecture



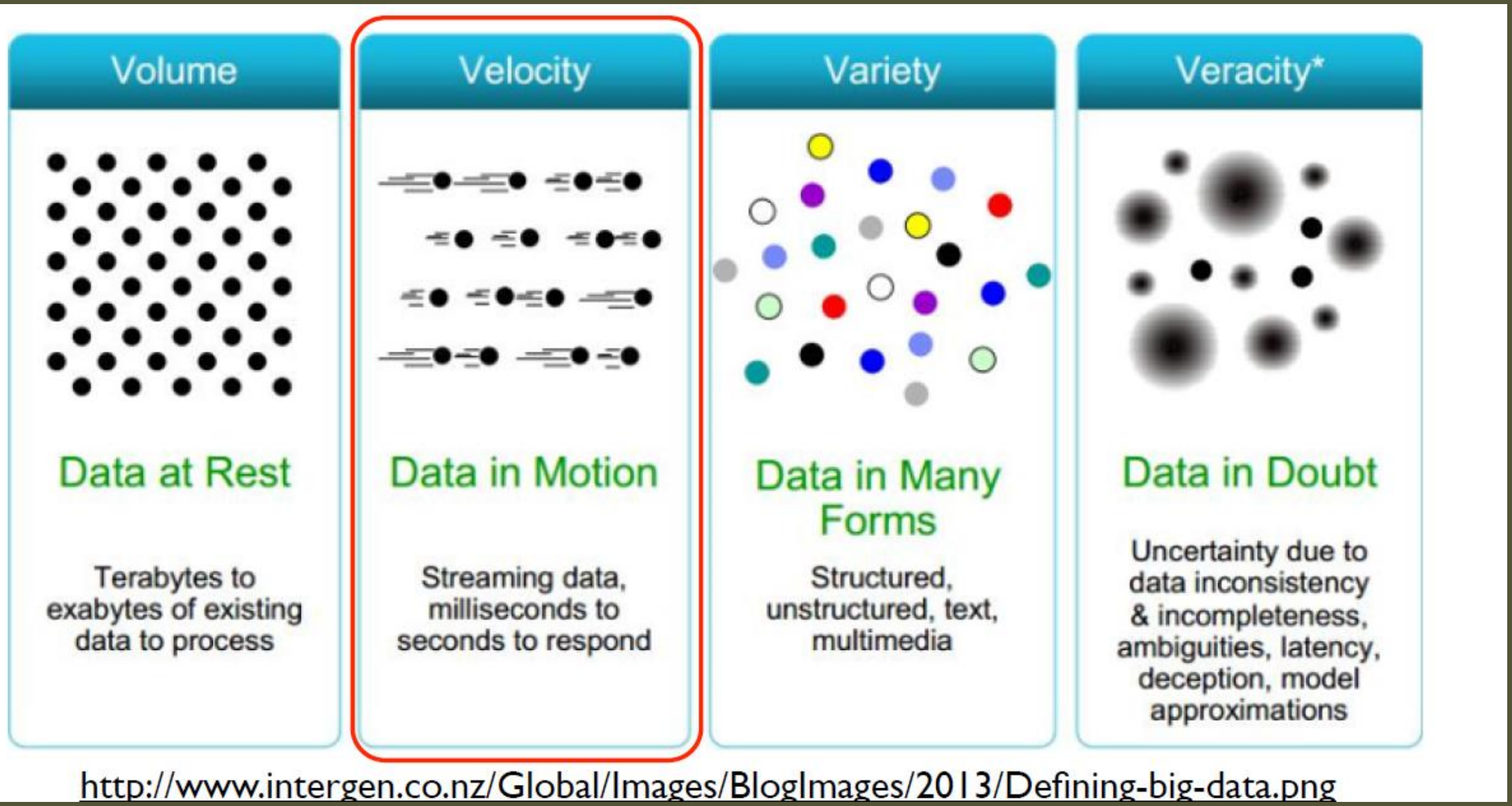
Cloud Computing Applications Architecture

Big Data Stack

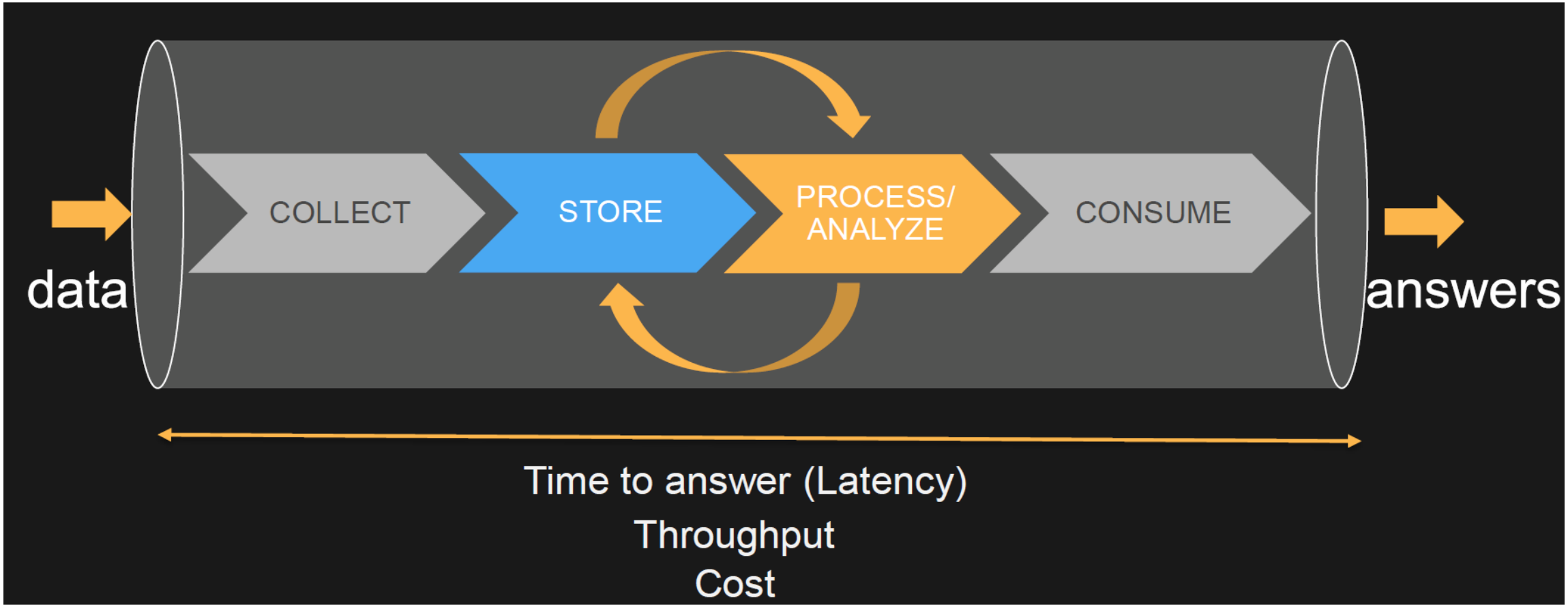



COLLECT
STORE
Types of Data
Applications
Web apps
Mobile apps
Data centers
AWS Direct Connect
RECORDS
Database
Database records
Logging
Logging
AWS CloudTrail
Amazon CloudWatch
DOCUMENTS
Search
Search documents
Transport
AWS Import/Export
Snowball
FILES
File store
Log files
Messaging
Messaging
Message
MESSAGES
Queue
Messaging events
IoT
Devices
Sensors & IoT platforms
AWS IoT
STREAMS
Stream storage
Devices / sensors / IoT stream

Cloud Computing Deals with Big Data in Many Forms



http://www.intergen.co.nz/Global/Images/BlogImages/2013/Defining-big-data.png



Amazon Web Services

Data / Access Characteristics: Hot, Warm, Cold

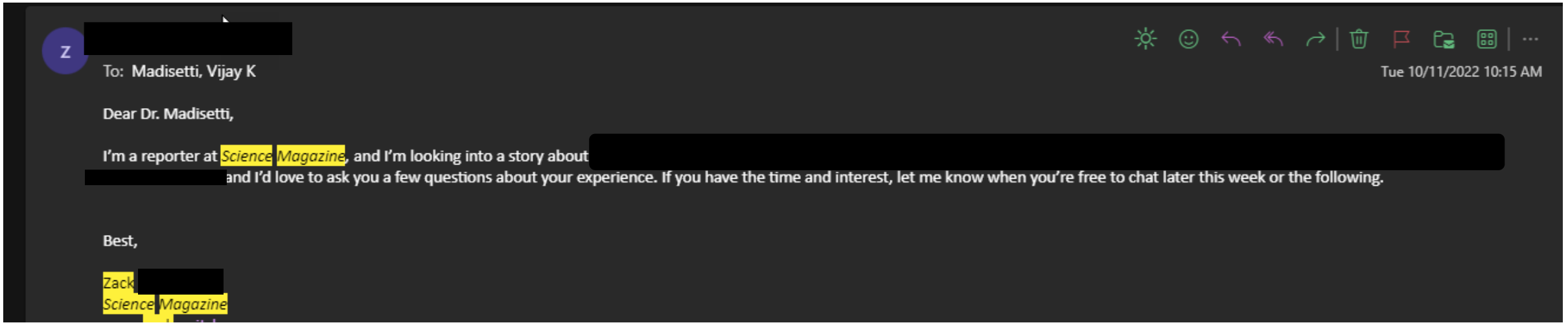
	Hot	Warm	Cold
Volume	MB–GB	GB–TB	PB
Item size	B–KB	KB–MB	KB–TB
Latency	ms	ms, sec	min, hrs
Durability	Low – high	High	Very high
Request rate	Very high	High	Low
Cost/GB	$$-$	$-¢¢	¢

Amazon Web Services

Process / Analyze

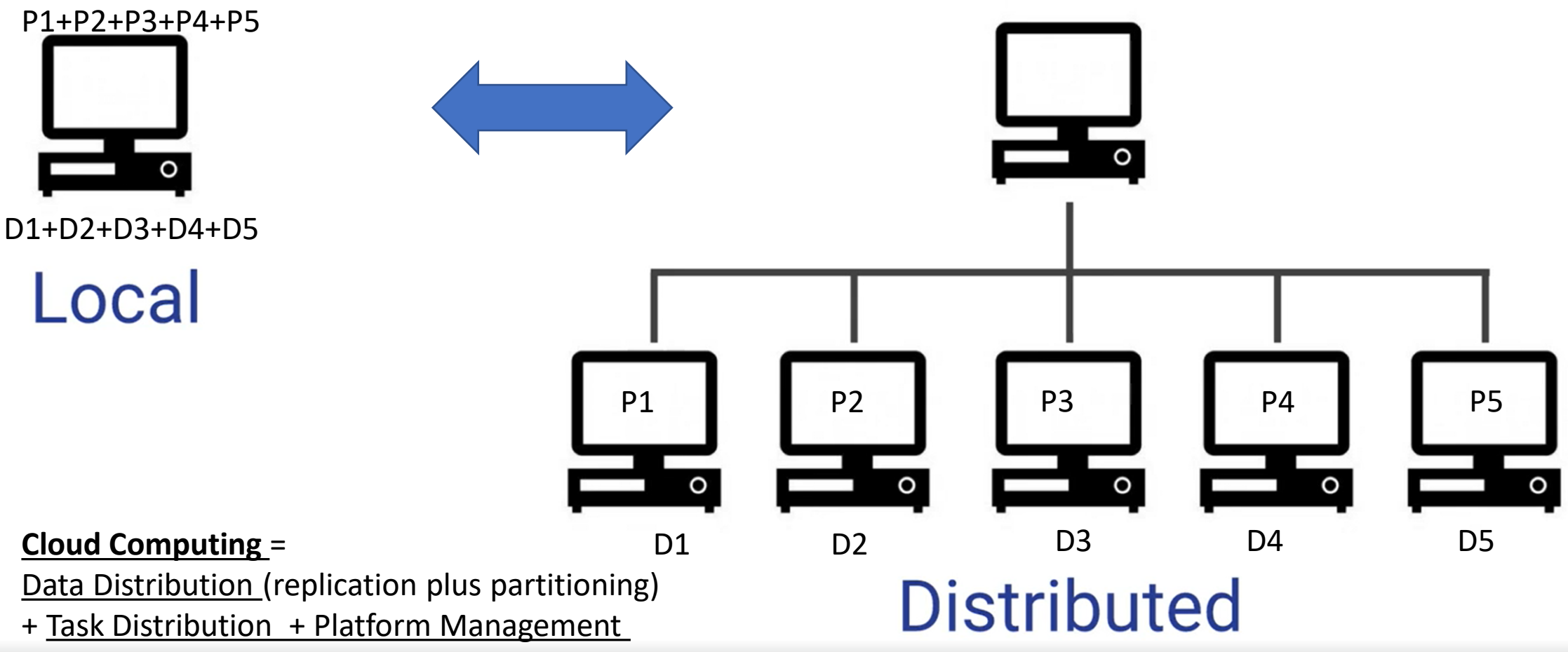
- Batch - Minutes or hours on cold data
  - Daily/weekly/monthly reports
- Interactive – Seconds on warm/cold data
  - Self-service dashboards
- Messaging – Milliseconds or seconds on hot data
  - Message/event buffering
- Streaming - Milliseconds or seconds on hot data
  - Billing/fraud alerts, 1 minute metrics

Lots of services, frameworks & tools - How do use them effectively ?


Z
[redacted]
To: Madisetti, Vijay K
Tue 10/11/2022 10:15 AM
Dear Dr. Madisetti,
I'm a reporter at Science Magazine, and I'm looking into a story about [redacted]
[redacted] and I'd love to ask you a few questions about your experience. If you have the time and interest, let me know when you're free to chat later this week or the following.
Best,
Zack [redacted]
Science Magazine

What Streaming / Messaging Technology Should I Use?

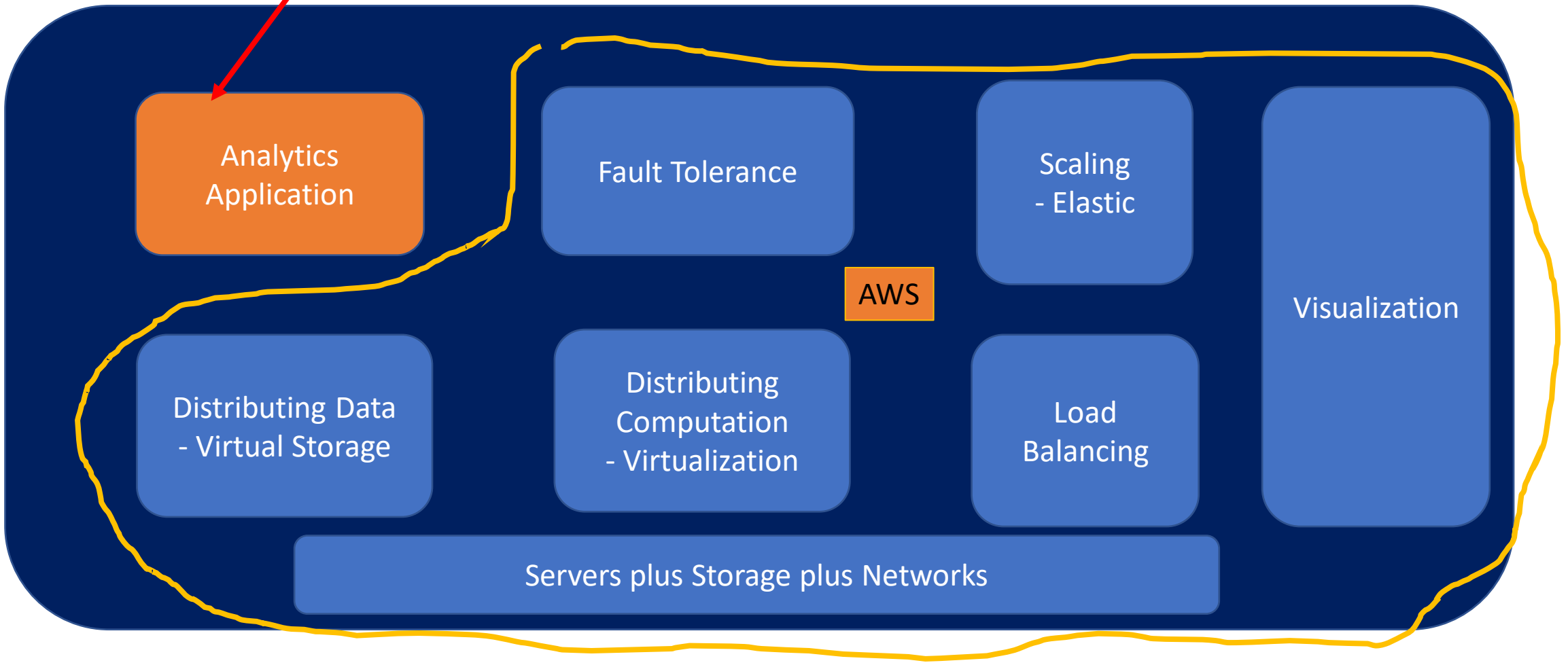
	Spark Streaming	Apache Storm	Kinesis KCL Application	AWS Lambda	Amazon SQS Apps
Scale	~ Nodes	~ Nodes	~ Nodes	Automatic	~ Nodes
Micro-batch or Real-time	Micro-batch	Real-time	Near-real-time	Near-real-time	Near-real-time
AWS managed service	Yes (EMR)	No (EC2)	No (KCL + EC2 + Auto Scaling)	Yes	No (EC2 + Auto Scaling)
Scalability	No limits ~ nodes	No limits ~ nodes	No limits ~ nodes	No limits	No limits
Availability	Single AZ	Configurable	Multi-AZ	Multi-AZ	Multi-AZ
Programming languages	Java, Python, Scala	Any language via Thrift	Java, via MultiLang Daemon (.NET, Python, Ruby, Node.js)	Node.js, Java, Python	AWS SDK languages (Java, .NET, Python, …)
	Fast		Fast		Fast



Cloud Computing =
Data Distribution (replication plus partitioning)
+ Task Distribution + Platform Management

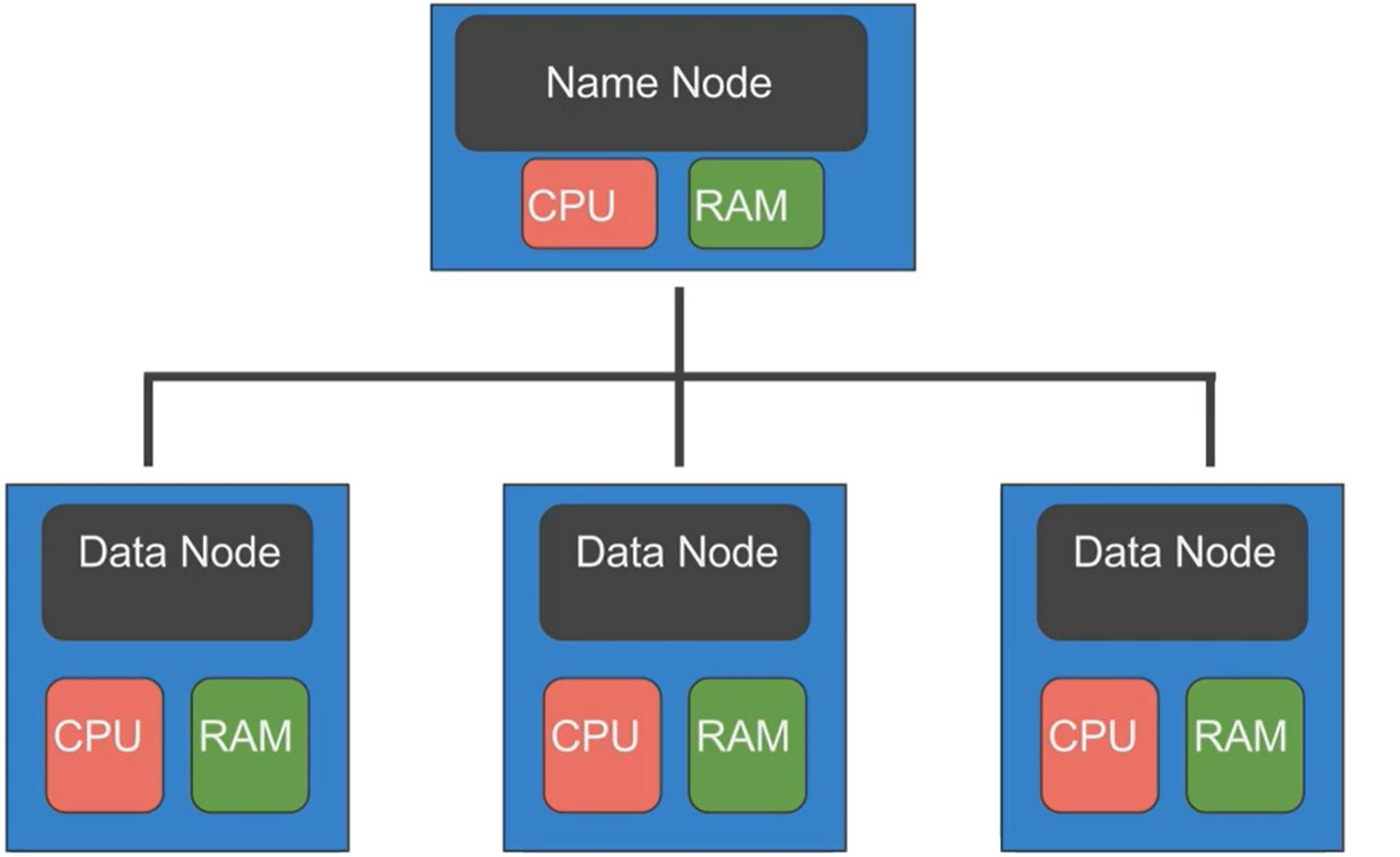
What is Cloud Computing ?

Customer Develops Application and "pays per use")

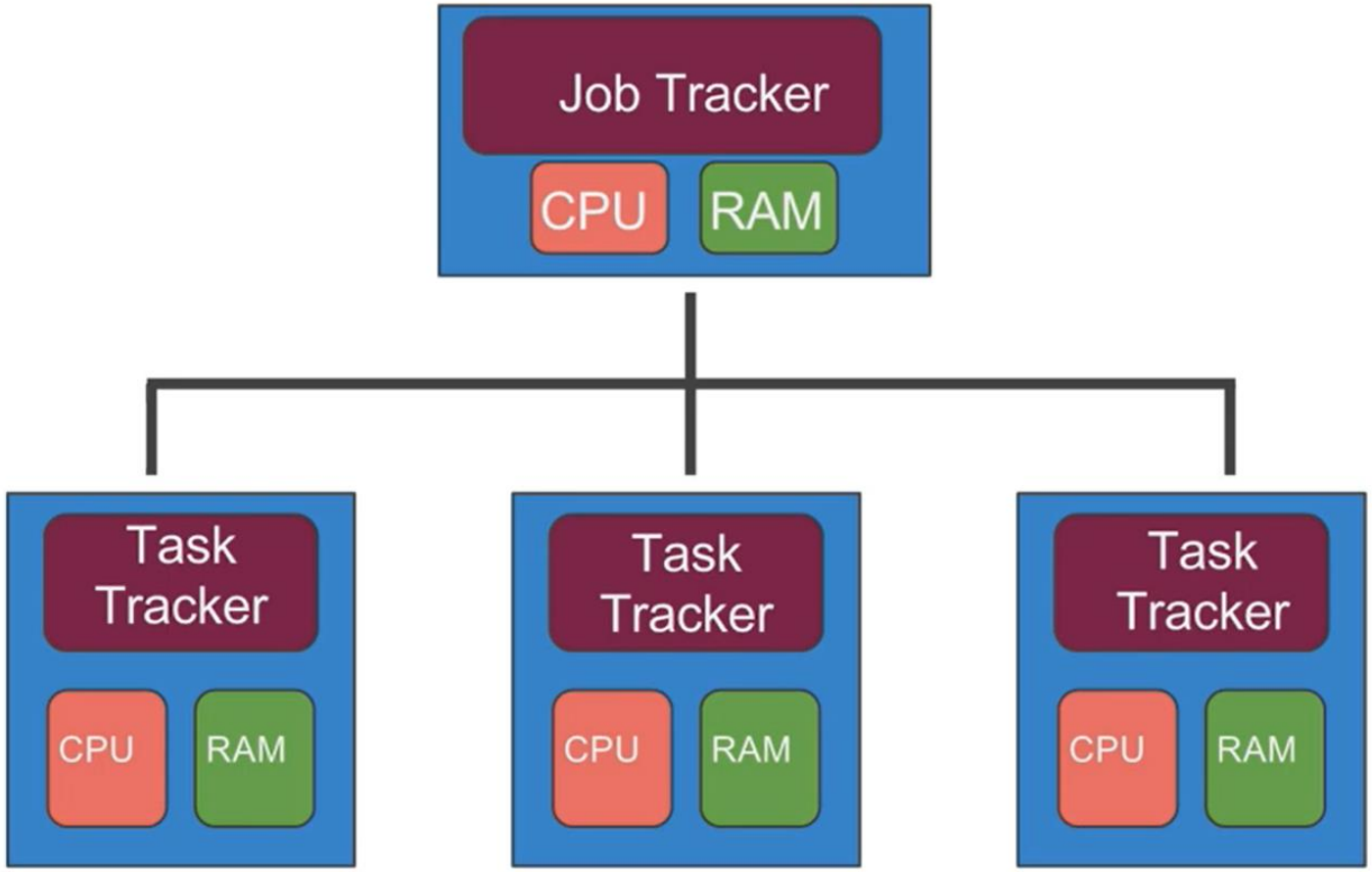


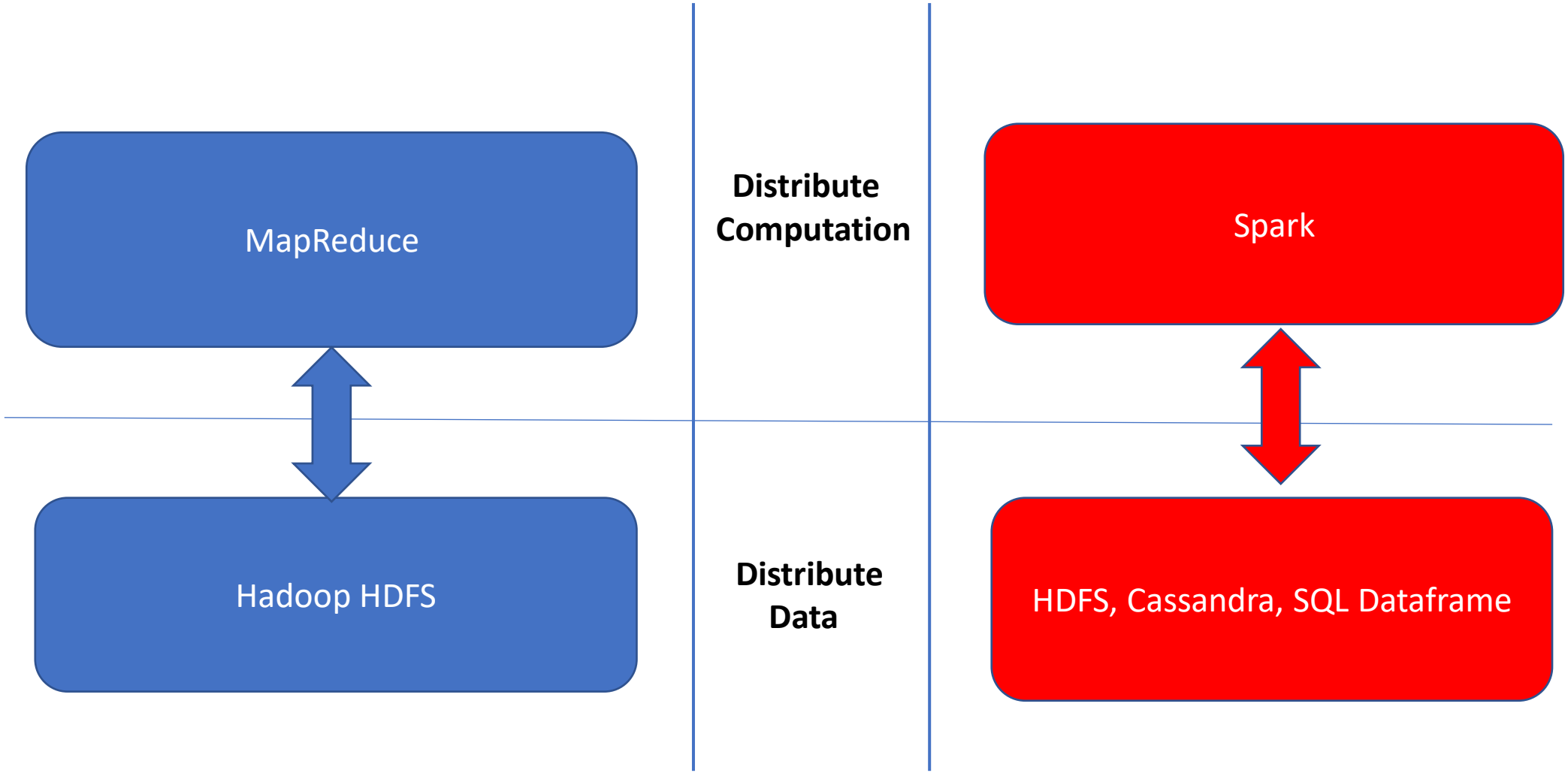
AWS Provides All Blue Items (Billions of Dollars!)

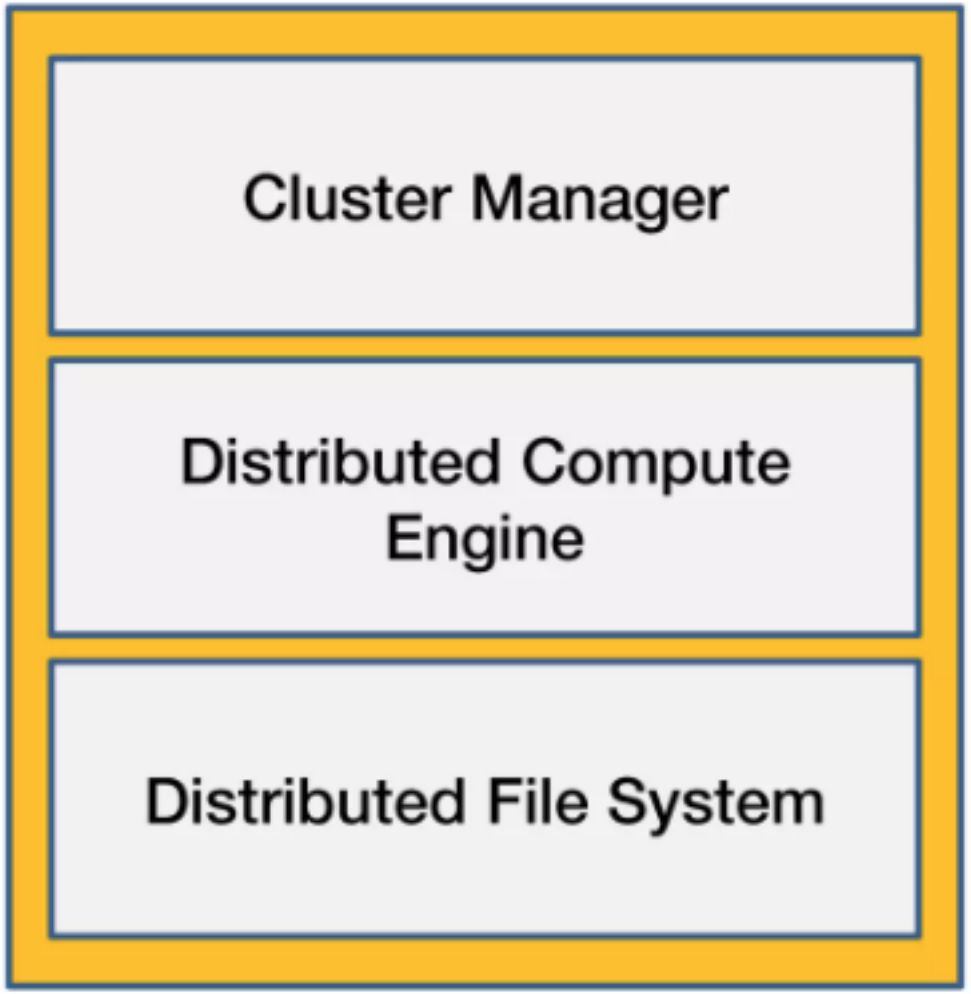
Map Reduce Models tied to HDFS

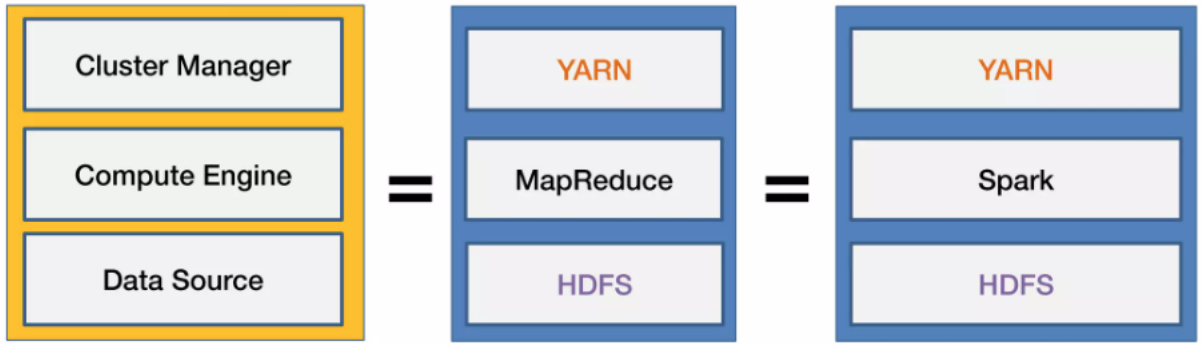


Spark is not tied to HDFS (Can use S3, Cassandra, NFS..)

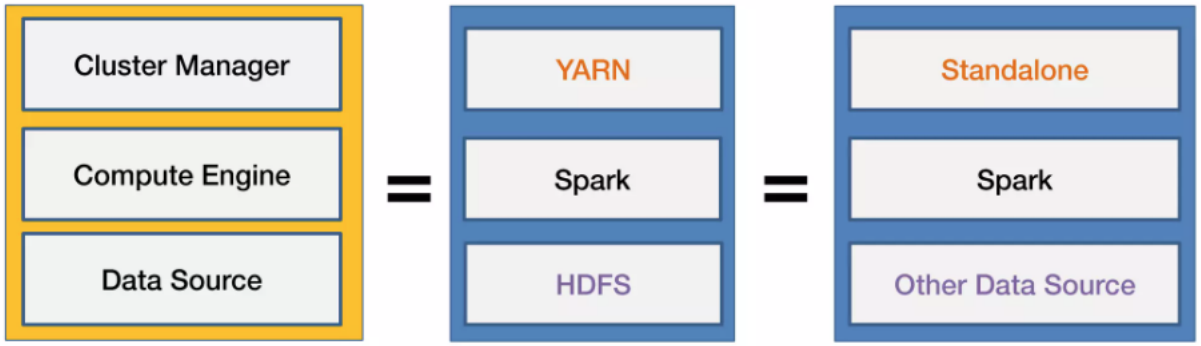


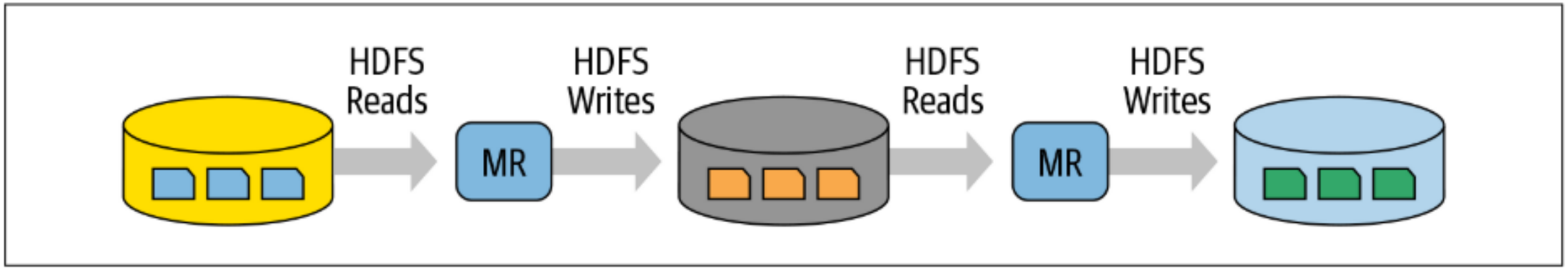

MapReduce
Distribute Computation
Spark
Hadoop HDFS
Distribute Data
HDFS, Cassandra, SQL Dataframe


Cluster Manager
Distributed Compute Engine
Distributed File System


Cluster Manager
Compute Engine
Data Source
=
YARN
MapReduce
HDFS
=
YARN
Spark
HDFS

Spark

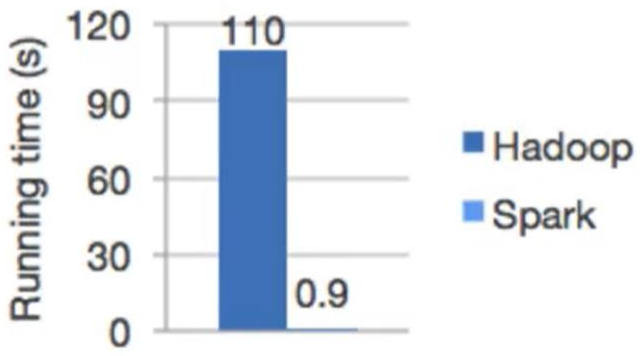

Cluster Manager
Compute Engine
Data Source
=
YARN
Spark
HDFS
=
Standalone
Spark
Other Data Source


HDFS
Reads
MR
HDFS
Writes
HDFS
Reads
MR
HDFS
Writes

Why Spark for Analytics Applications ? – Primarily Reading (and not Updating) Large Data Sets

Spark is quickly becoming one of the most powerful Big Data tools!

Run programs up to 100x faster than Hadoop MapReduce in memory



Featured product

IBM Analytics for Apache Spark

IBM Analytics for Apache Spark™ gives you the power of Apache Spark with integrated Jupyter Notebooks, so that you can iterate faster, and get to answers faster. The service is fully-managed, which gives you immediate access to hassle-free Apache Spark.

Try it free

Watch overview (01:32)

Related products

IBM BigInsights on Cloud

Accelerate to enterprise-grade advanced analytics built on proven open source Hadoop® technology.

Learn more

Watch overview (01:51)

IBM Data Science Experience

Cloud-based, social workspace that helps data scientists consolidate their use of and collaborate across multiple open source tools such as R and Python.

Learn more

taking advantage of the operational and cost benefits provided by Amazon EMR.

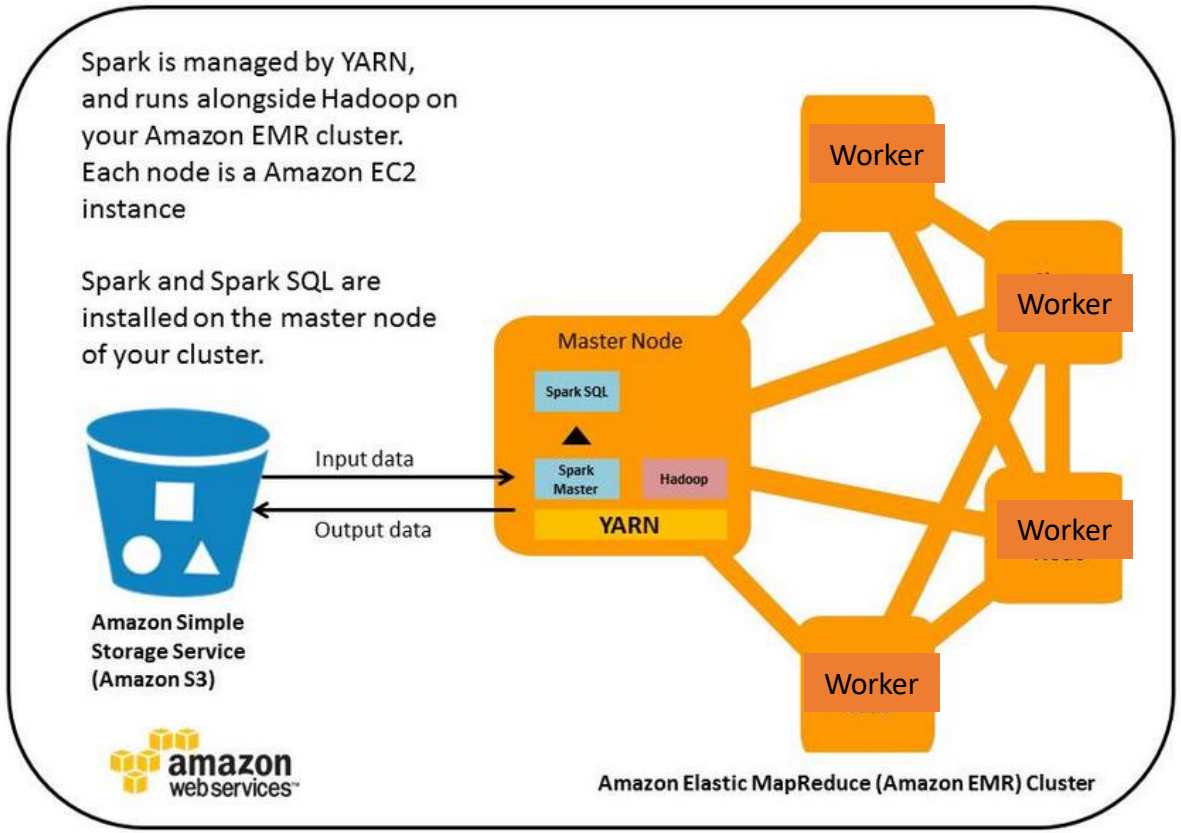
The following diagram illustrates running Spark on a Hadoop cluster managed by Amazon EMR. When you launch a cluster, Amazon EMR provisions virtual server instances from Amazon EC2. You can then use an Amazon EMR bootstrap action to install Spark on the cluster. Spark on Amazon EMR can process data from an Amazon S3 bucket or stored in HDFS, and results from queries can also be persisted back to Amazon S3 or HDFS after analysis.

AWS PRODUCTS

Amazon CloudFront
Amazon EC2
Amazon Flexible Payments Service
Amazon SimpleDB
Amazon SQS
Amazon S3
AWS Elastic Beanstalk
Amazon SES

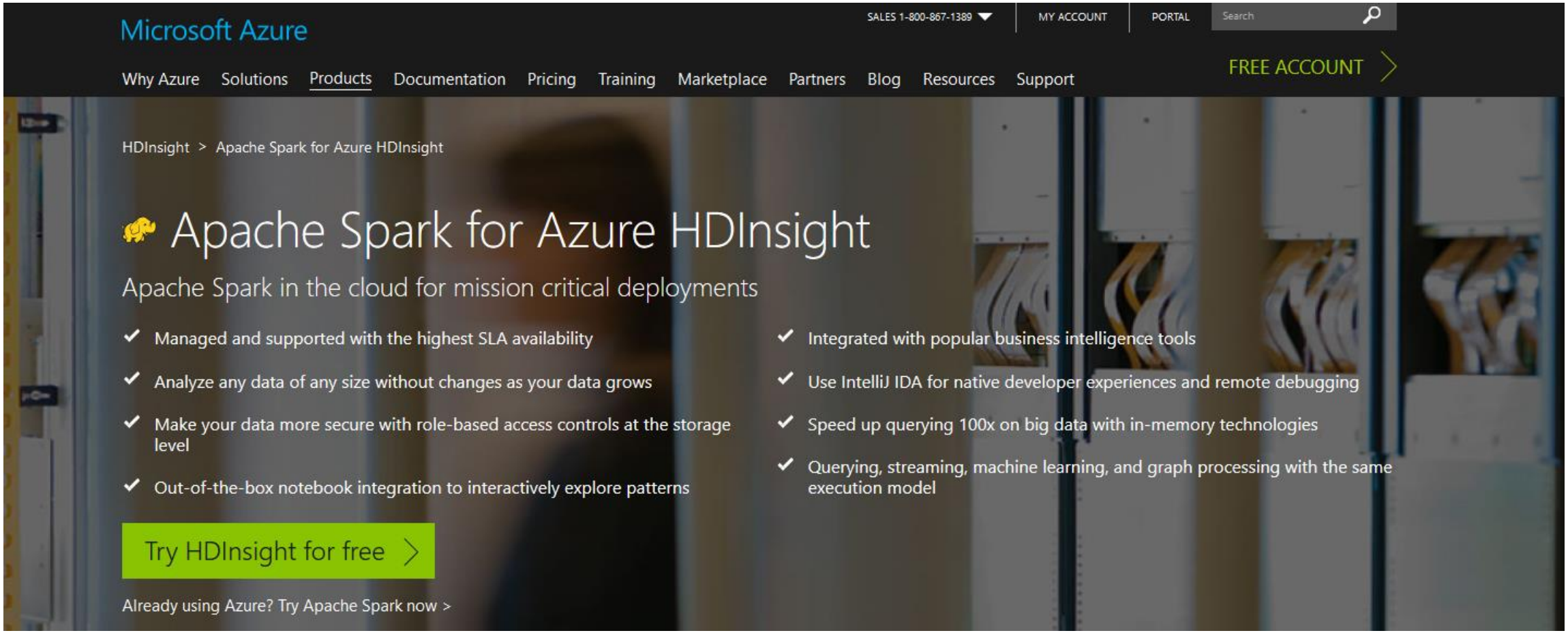
TECHNOLOGY

Java
Windows
.NET
Ruby
Python
PHP
Mobile



Check This !

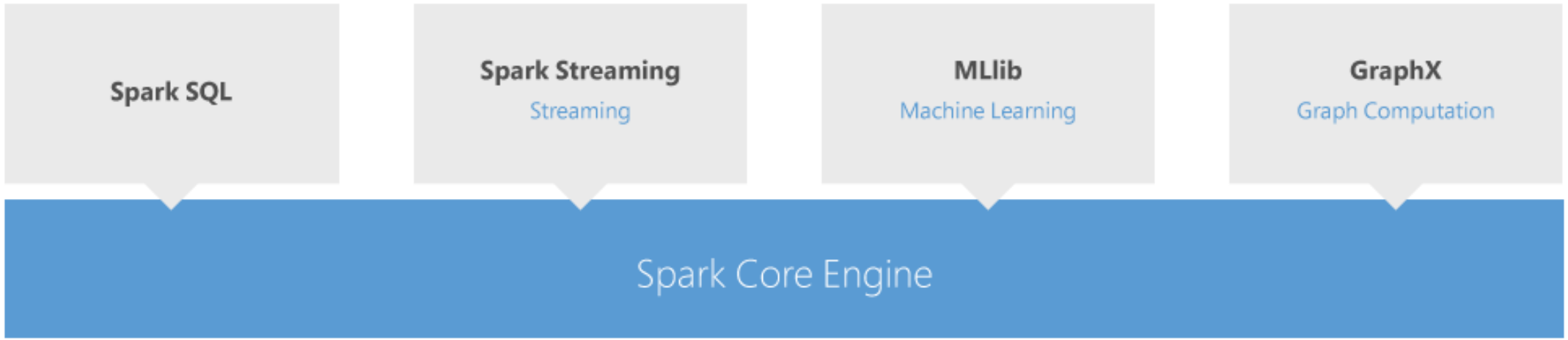
https://aws.amazon.com/articles/run-spark-and-spark-sql-on-amazon-elastic-mapreduce/


Microsoft Azure
SALES 1-800-867-1389
MY ACCOUNT
PORTAL
Search
Why Azure
Solutions
Products
Documentation
Pricing
Training
Marketplace
Partners
Blog
Resources
Support
FREE ACCOUNT
HDInsight > Apache Spark for Azure HDInsight
Apache Spark for Azure HDInsight
Apache Spark in the cloud for mission critical deployments
Managed and supported with the highest SLA availability
Analyze any data of any size without changes as your data grows
Make your data more secure with role-based access controls at the storage level
Out-of-the-box notebook integration to interactively explore patterns
Integrated with popular business intelligence tools
Use IntelliJ IDA for native developer experiences and remote debugging
Speed up querying 100x on big data with in-memory technologies
Querying, streaming, machine learning, and graph processing with the same execution model
Try HDInsight for free
Already using Azure? Try Apache Spark now >

What is Apache Spark?

Apache Spark is an open-source processing framework that runs large-scale data analytics applications. Spark is built on an in-memory compute engine, which enables high performance querying on big data. It takes advantage of a parallel data processing framework that persists data in-memory and disk if needed. This allows Spark to deliver 100x faster speed and a common execution model for tasks such as extract, transform, load (ETL), batch, interactive queries, and others on data in an Apache Hadoop Distributed File System (HDFS). Azure makes Apache Spark easy and cost effective to deploy with no hardware to buy, no software to configure, a full notebook experience to author compelling narratives, and integration with partner business intelligence tools.

Watch an Apache Spark overview video



Spark Models

One execution model for multiple tasks

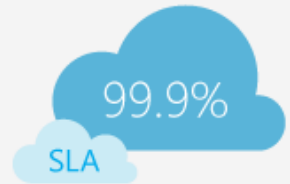
Apache Spark takes advantage of a common execution model for doing multiple tasks like ETL, batch queries, interactive queries, real-time streaming, machine learning, and graph processing on data stored in Azure Data Lake Store. This allows you to use Spark for Azure HDInsight to solve big data challenges in near real-time, such as fraud detection, click stream analysis, financial alerts, telemetry from Internet of Things (IoT) sensors and devices, social analytics, always-on ETL pipelines, and network monitoring.

In-memory processing for interactive scenarios

Customers today expect quick answers to their questions, instead of waiting minutes, hours, or days. Apache Spark delivers by persisting data in-memory to get up to 100x faster queries, while processing large datasets in Hadoop. This makes Spark for Azure HDInsight ideal to speed up intensive big data applications.

Highest availability for business continuity

To run Spark at the highest scale, Microsoft gives you the industry's highest availability SLA at 99.9% to ensure your business continuity and protection against catastrophic events. We co-led with Cloudera and the project Livy to create an open-source Apache-licensed REST web service for managing long-running Spark contexts and submitting Spark jobs. This new capability is designed to make Spark a more robust back end for running interactive notebooks and allow other applications to take advantage of Spark for their interactive workloads.

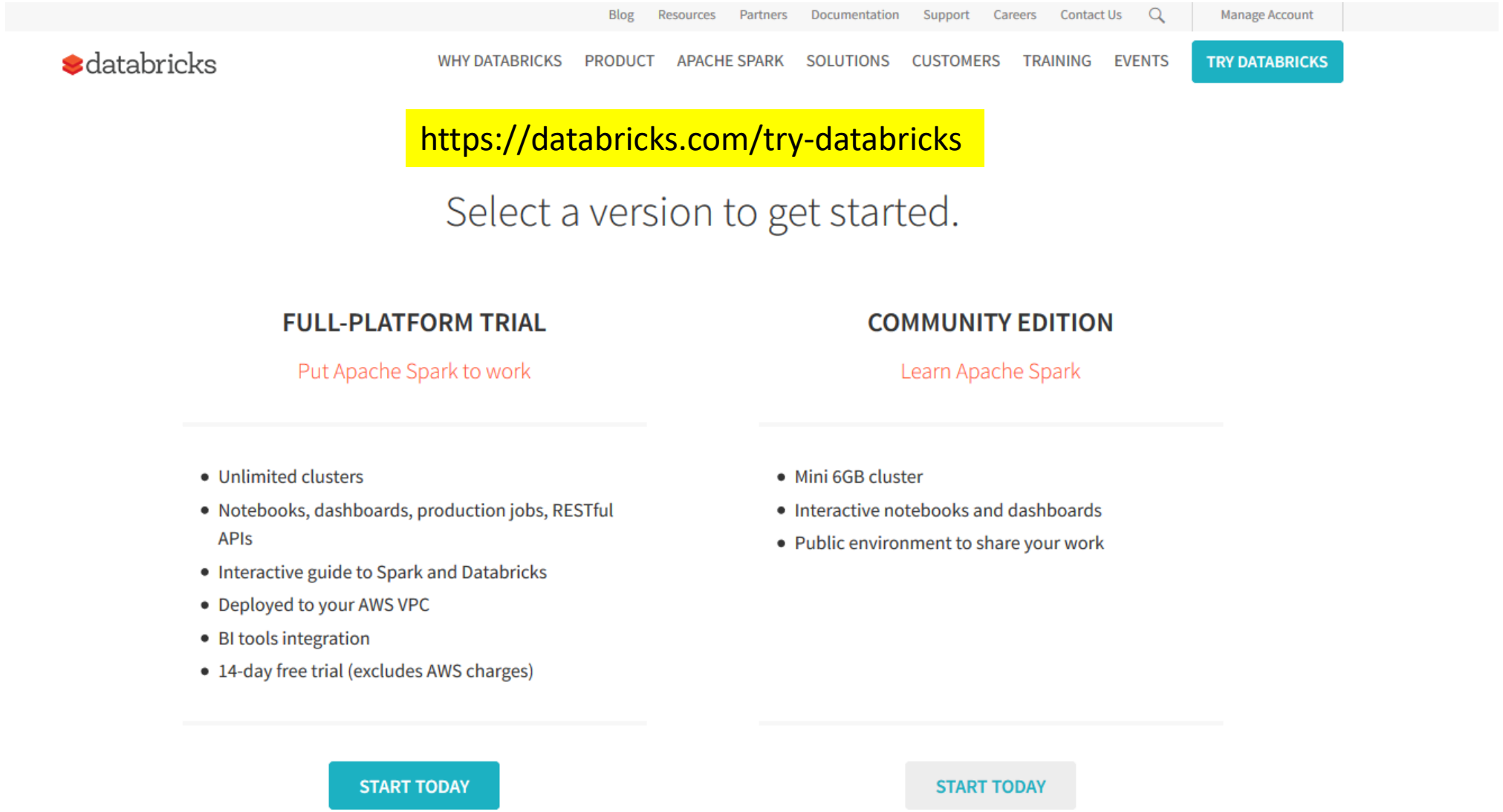


Analyze any data of any size without changes as data grows

To make sure Spark runs at scale, we integrated Spark with Azure Data Lake Store. This integration is uniquely available from Microsoft and allows Spark to store and process data that scales to any size, without forcing changes to your application as data grows. Through this integration, you can implement role-based data access controls at the storage level.

Real-time processing for real-time scenarios

Today's connected world is defined by big data that arrives in real-time. Spark Stream for HDInsight is ideal for challenging real-time scenarios. It enables various opportunities including Internet of Things (IoT) scenarios, real-time remote management and monitoring, and getting insights from devices like mobile phones or connected cars.


Blog Resources Partners Documentation Support Careers Contact Us Manage Account
databricks
WHY DATABRICKS PRODUCT APACHE SPARK SOLUTIONS CUSTOMERS TRAINING EVENTS
TRY DATABRICKS
https://databricks.com/try-databricks
Select a version to get started.
FULL-PLATFORM TRIAL
Put Apache Spark to work
Unlimited clusters
Notebooks, dashboards, production jobs, RESTful APIs
Interactive guide to Spark and Databricks
Deployed to your AWS VPC
BI tools integration
14-day free trial (excludes AWS charges)
START TODAY
COMMUNITY EDITION
Learn Apache Spark
Mini 6GB cluster
Interactive notebooks and dashboards
Public environment to share your work
START TODAY

Comparing Spark with EMR

EMR offers only two operators – map and reduce

Spark offers API with 80+ operators

Spark minimizes code further, improving developer productivity

Spark also uses in-memory computation

Spark offers many models of computation – batch, streaming, interactive, graph computing, ….

```
01 import java.io.IOException;
02 import java.util.StringTokenizer;
03
04 import org.apache.hadoop.conf.Configuration;
05 import org.apache.hadoop.fs.Path;
06 import org.apache.hadoop.io.IntWritable;
07 import org.apache.hadoop.io.Text;
08 import org.apache.hadoop.mapreduce.Job;
09 import org.apache.hadoop.mapreduce.Mapper;
10 import org.apache.hadoop.mapreduce.Reducer;
11 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
13
14 public class WordCount {
15
16   public static class TokenizerMapper
17        extends Mapper<Object, Text, Text, IntWritable>{
18
19     private final static IntWritable one = new IntWritable(1);
20     private Text word = new Text();
21
22     public void map(Object key, Text value, Context context
23                     ) throws IOException, InterruptedException {
24       StringTokenizer itr = new StringTokenizer(value.toString());
25       while (itr.hasMoreTokens()) {
26         word.set(itr.nextToken());
27         context.write(word, one);
28       }
29     }
30   }
31
32   public static class IntSumReducer
33        extends Reducer<Text,IntWritable,Text,IntWritable> {
34     private IntWritable result = new IntWritable();
35
36     public void reduce(Text key, Iterable<IntWritable> values,
37                        Context context
38                        ) throws IOException, InterruptedException {
39       int sum = 0;
40       for (IntWritable val : values) {
41         sum += val.get();
42       }
43       result.set(sum);
44       context.write(key, result);
45     }
46   }
47
48   public static void main(String[] args) throws Exception {
49     Configuration conf = new Configuration();
50     Job job = Job.getInstance(conf, "word count");
51     job.setJarByClass(WordCount.class);
52     job.setMapperClass(TokenizerMapper.class);
53     job.setCombinerClass(IntSumReducer.class);
54     job.setReducerClass(IntSumReducer.class);
55     job.setOutputKeyClass(Text.class);
56     job.setOutputValueClass(IntWritable.class);
57     FileInputFormat.addInputPath(job, new Path(args[0]));
58     FileOutputFormat.setOutputPath(job, new Path(args[1]));
59     System.exit(job.waitForCompletion(true) ? 0 : 1);
60   }
61 }
```

EMR

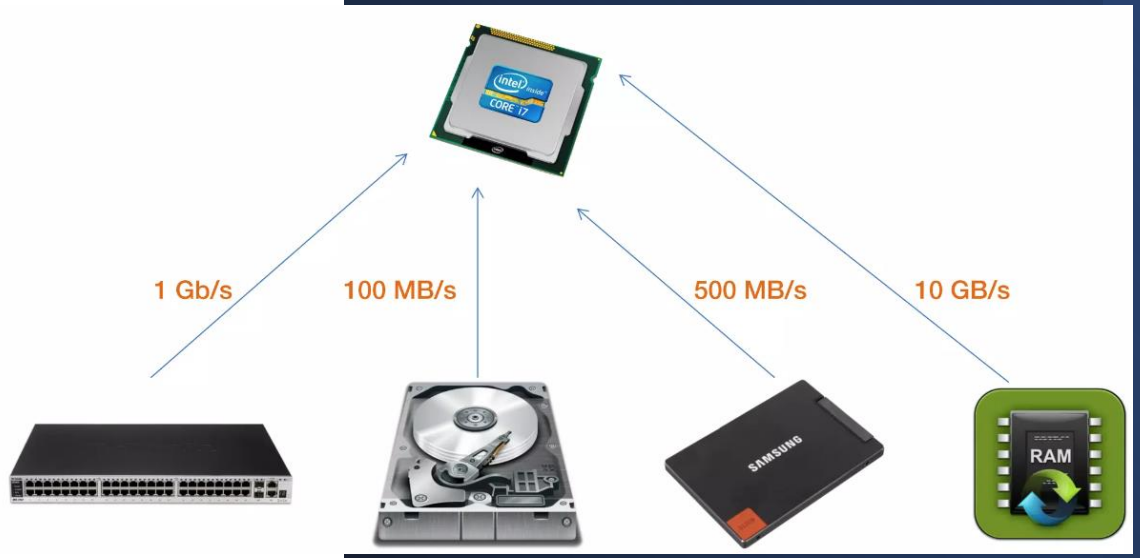
```
01 import org.apache.spark.SparkContext
02 import org.apache.spark.SparkContext._
03
04 object WordCount {
05   def main(args: Array[String]): Unit = {
06     val sc = new SparkContext()
07     val lines = sc.textFile(args(0))
08     val wordCounts = lines.flatMap {line => line.split(" ")}
09                           .map(word => (word, 1))
10                           .reduceByKey(_ + _)
11     wordCounts.saveAsTextFile(args(1))
12   }
13 }
```

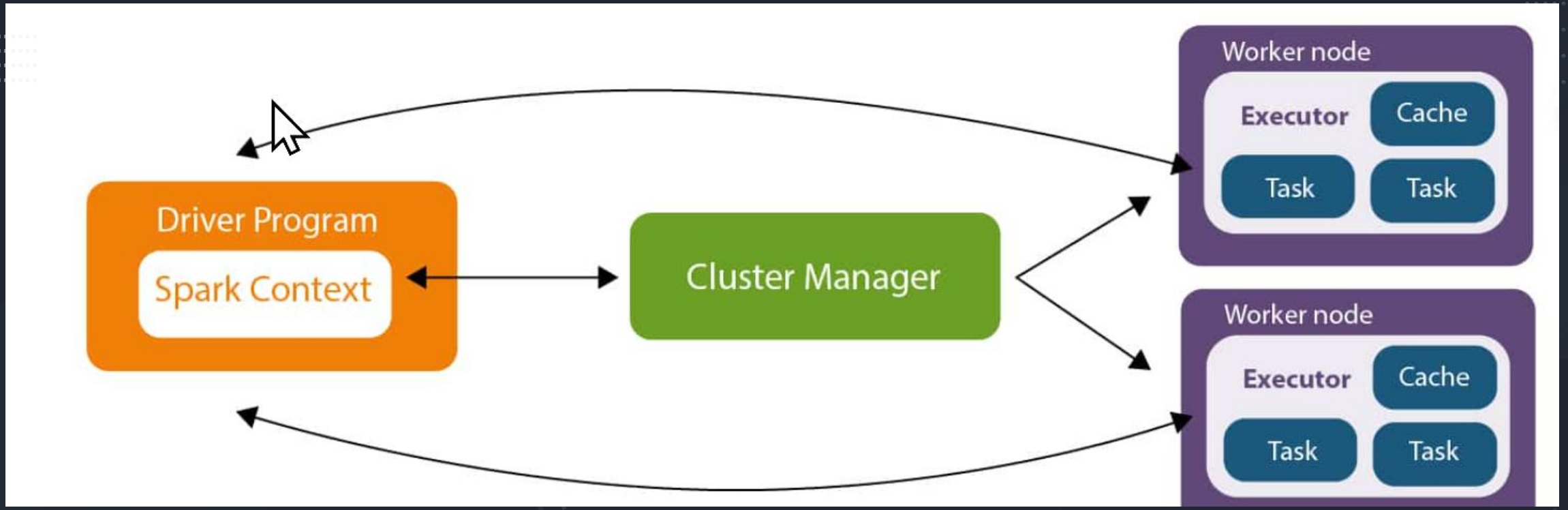
SPARK

Word Histogram Calculation

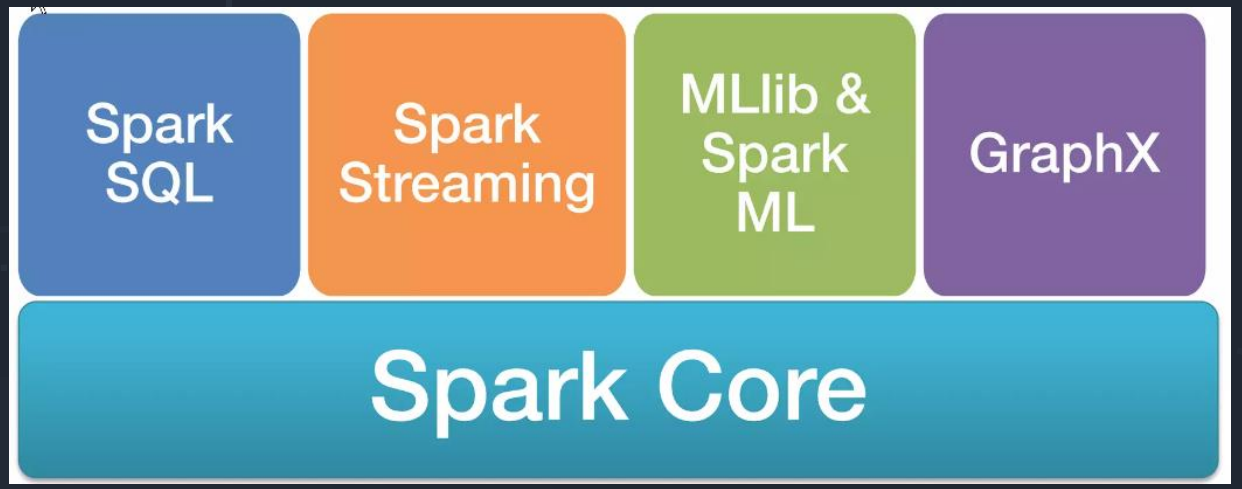
In-Memory versus Disk

- Map Reduce breaks a complex job into map and reduce, and each job reads from and writes to HDFS disk.
- Disk I/O is slow and expensive
- Spark operators from data stored in memory (RAM)





- **Spark Execution Architecture**



Closer look at Spark

https://grouplens.org/datasets/movielens/100k/

User ID (total 943 users)
Movie ID (total 1682)
Rating 1-5

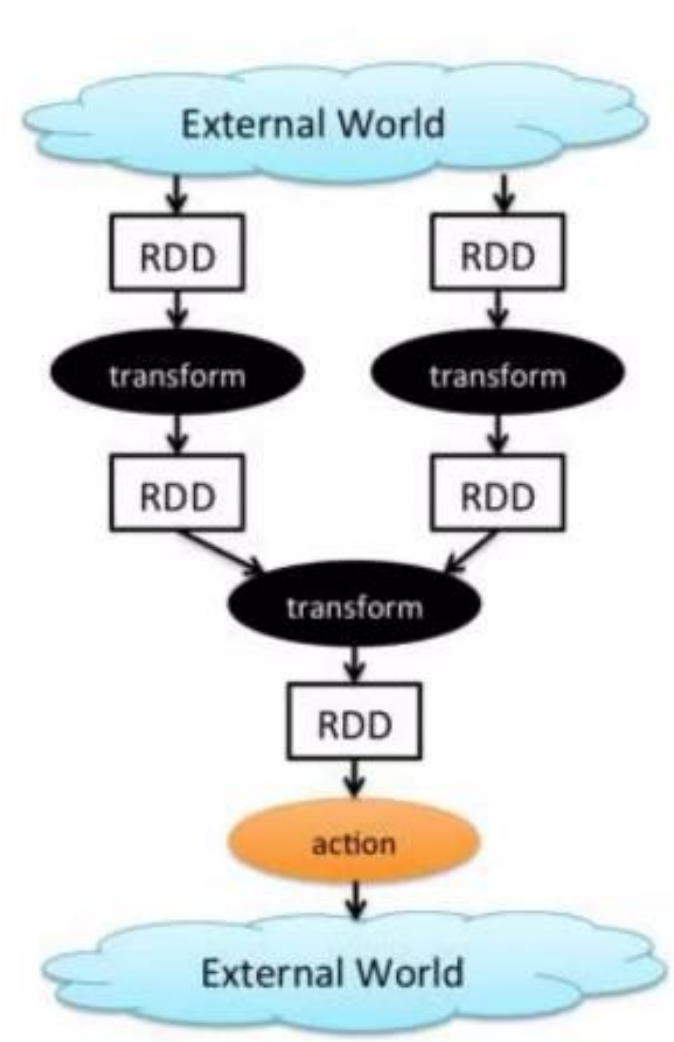
```
MovieLens data sets were collected by the GroupLens Research Project
at the University of Minnesota.

This data set consists of:
        * 100,000 ratings (1-5) from 943 users on 1682 movies.
        * Each user has rated at least 20 movies.
        * Simple demographic info for the users (age, gender, occupation, zip)
```

```
vkm@VKM-Surface-21:~/netflix/ml-100k$ more u.user
1|24|M|technician|85711
2|53|F|other|94043
3|23|M|writer|32067
4|24|M|technician|43537
5|33|F|other|15213
6|42|M|executive|98101
7|57|M|administrator|91344
8|36|M|administrator|05201
9|29|M|student|01002
10|53|M|lawyer|90703
11|39|F|other|30329
12|28|F|other|06405
13|47|M|educator|29206
14|45|M|scientist|55106
15|49|F|educator|97301
16|21|M|entertainment|10309
17|30|M|programmer|06355
18|35|F|other|37212
19|40|M|librarian|02138
20|42|F|homemaker|95660
```

```
vkm@VKM-Surface-21:~/netflix/ml-100k$ more u.data
196     242     3       881250949
186     302     3       891717742
22      377     1       878887116
244     51      2       880606923
166     346     1       886397596
298     474     4       884182806
115     265     2       881171488
253     465     5       891628467
305     451     3       886324817
6       86      3       883603013
62      257     2       879372434
286     1014    5       879781125
200     222     5       876042340
210     40      3       891035994
224     29      3       888104457
303     785     3       879485318
122     387     5       879270459
194     274     2       879539794
291     1042    4       874834944
234     1184    2       892079237
119     392     4       886176814
167     486     4       892738452
299     144     4       877881320
291     118     2       874833878
308     1       4       887736532
95      546     2       879196566
38      95      5       892430094
102     768     2       883748450
63      277     4       875747401
160     234     5       876861185
50      246     3       877052329
```

```
1318|Catwalk (1995)|07-Jun-1996||http://us.imdb.com/Title?Catwalk+(1995/I)|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0
1319|Neon Bible, The (1995)|01-Mar-1996||http://us.imdb.com/M/title-exact?Neon%20Bible,%20The%20(1995)|0|0|0|0|0|0|0|1|
|0|0|0|0|0|0|0|0
1320|Homage (1995)|03-May-1996||http://us.imdb.com/M/title-exact?Homage%20(1995)|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1321|Open Season (1996)|10-May-1996||http://us.imdb.com/Title?Open+Season+(1996)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0
1322|Metisse (Caf au Lait) (1993)|01-Jan-1993||http://us.imdb.com/Title?M%E9tisse+(1993)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|
0|0
1323|Wooden Man's Bride, The (Wu Kui) (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Wu%20Kui%20(1994)|0|0|0|0|0|0
|1|0|0|0|0|0|0|0|0|0
1324|Loaded (1994)|12-Apr-1996||http://us.imdb.com/M/title-exact?Loaded%20(1994)|0|0|0|0|0|0|0|1|0|0|0|0|0|0|1|0|0
1325|August (1996)|12-Apr-1996||http://us.imdb.com/M/title-exact?August%20(1996)|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1326|Boys (1996)|10-May-1996||http://us.imdb.com/M/title-exact?Boys%20(1996)|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1327|Captives (1994)|16-Sep-1994||http://us.imdb.com/Title?Captives+(1994)|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1328|Of Love and Shadows (1994)|10-May-1996||http://us.imdb.com/M/title-exact?Of%20Love%20and%20Shadows%20(1994)|0|0|0|0
|0|0|1|0|0|0|0|0|0|0|0|0
1329|Low Life, The (1994)|10-May-1996||http://us.imdb.com/Title?Low+Life,+The+(1994/I)|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0
1330|An Unforgettable Summer (1994)|01-Jan-1994||http://us.imdb.com/Title?Un+%E9t%E9+inoubliable+(1994)|0|0|0|0|0|0|0|1
0|0|0|0|0|0|0|0|0
1331|Last Klezmer: Leopold Kozlowski, His Life and Music, The (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Last%
ezmer%3A%20Leopold%20Kozlowski%2C%20His%20Life%20and%20Music%2C%20The%20%281995%29|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
```

Spark Operates on RDDs in RAM



Resilient distributed datasets (RDDs)
» Immutable, partitioned collections of objects

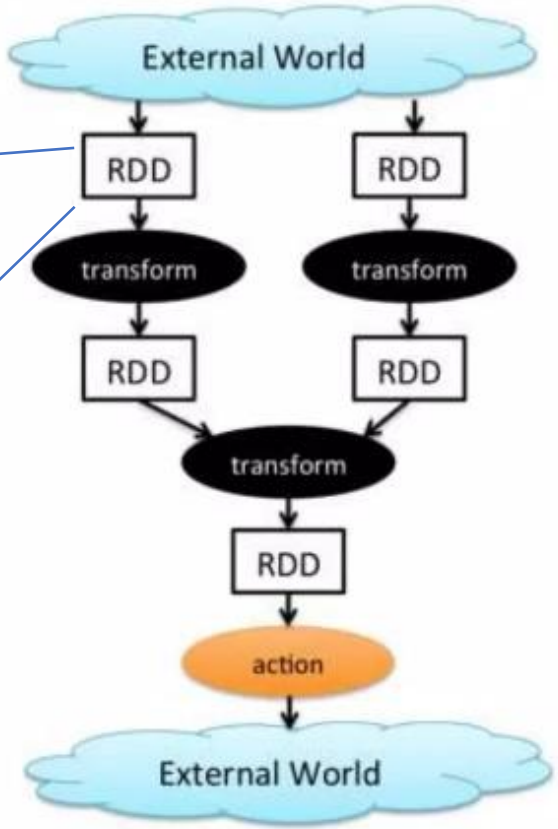
Transformations (e.g. map, filter, groupBy, join)
» Lazy operations to build RDDs from other RDDs

Actions (e.g. count, collect, save)
» Return a result or write it to storage

Creating RDDs

Creating RDDs

```
// Turn a Scala collection into an RDD
sc.parallelize(List(1, 2, 3))

// Load text file from local FS, HDFS, or S3
sc.textFile("file.txt")
sc.textFile("directory/*.txt")
sc.textFile("hdfs://namenode:9000/path/file")

// Use any existing Hadoop InputFormat
sc.hadoopFile(keyClass, valClass, inputFmt, conf)
```

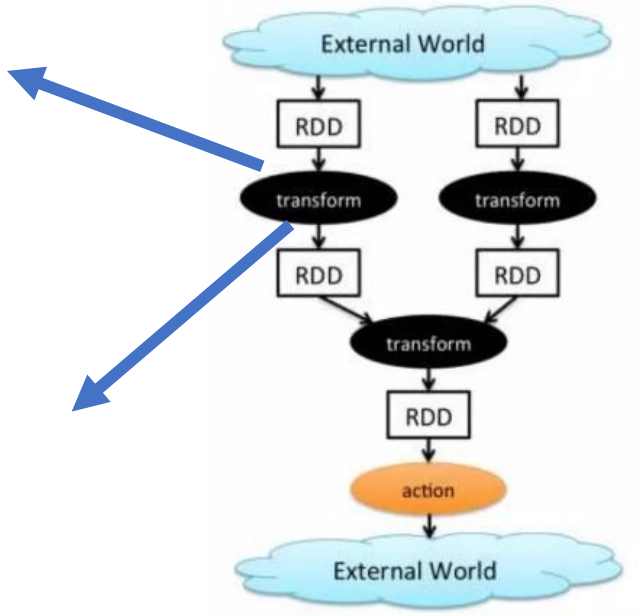


Transformations

Basic Transformations

```
val nums = sc.parallelize(List(1, 2, 3))

// Pass each element through a function
val squares = nums.map(x => x*x)   // {1, 4, 9}

// Keep elements passing a predicate
val even = squares.filter(_ % 2 == 0)  // {4}

// Map each element to zero or more others
nums.flatMap(x => 1 to x)  // => {1, 1, 2, 1, 2, 3}
```

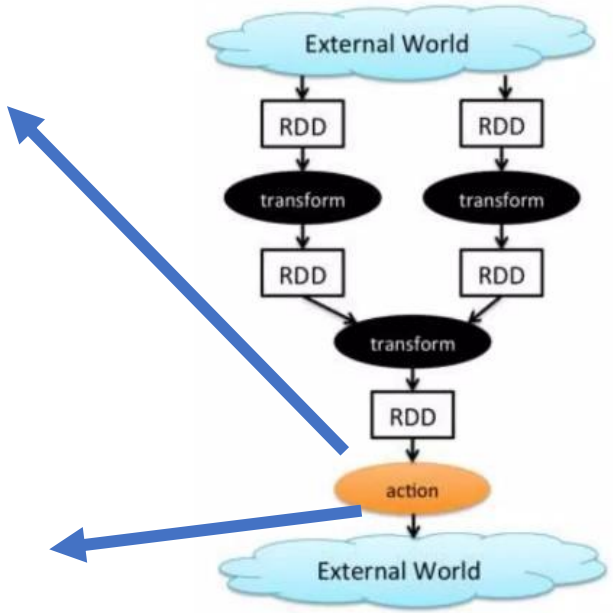
Range object (sequence of numbers 1, 2, …, x)



Transformations

Basic Actions

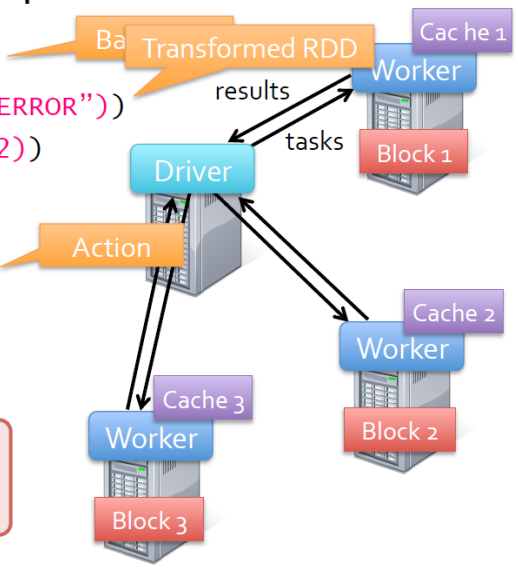
```
val nums = sc.parallelize(List(1, 2, 3))

// Retrieve RDD contents as a local collection
nums.collect() // => Array(1, 2, 3)

// Return first K elements
nums.take(2)   // => Array(1, 2)

// Count number of elements
nums.count()   // => 3

// Merge elements with an associative function
nums.reduce(_ + _)  // => 6

// Write elements to a text file
nums.saveAsTextFile("hdfs://file.txt")
```



How to Scan a TB of Data in 5 seconds – Looking for Repeated Login Errors of Certain Types in Security Logs of a Large Organization over Past 30 Days

Load error messages from a log into memory, then interactively search for various patterns

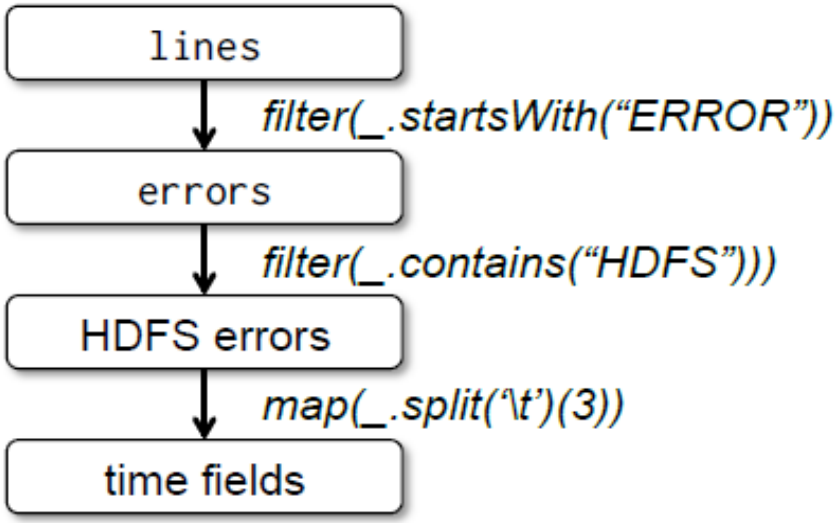
```
val lines = spark.textFile("hdfs://...")
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split('\t')(2))
messages.cache()

messages.filter(_.contains("foo")).count
messages.filter(_.contains("bar")).count
. . .
```



Result: scaled to 1 TB data in 5-7 sec (vs 170 sec for on-disk data)

Error Logs Analysis

3/10/2024



Spark Code

```
// Return the time fields of errors mentioning
// HDFS as an array (assuming time is field
// number 3 in a tab-separated format):
errors.filter(_.contains("HDFS"))
      .map(_.split('\t')(3))
      .collect()
```

Spark allows Key Value Operations for MR

Some Key-Value Operations

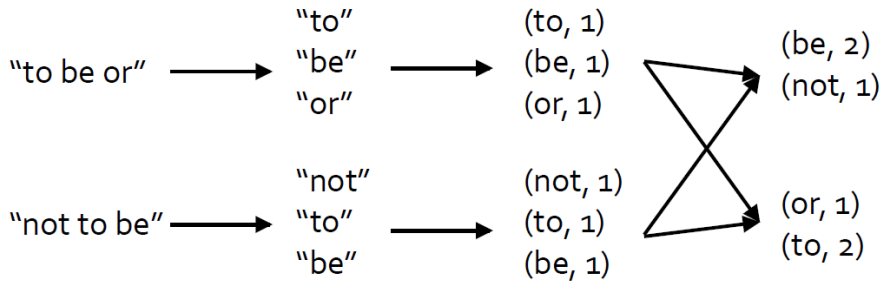
```
val pets = sc.parallelize(
             List(("cat", 1), ("dog", 1), ("cat", 2)))

pets.reduceByKey(_ + _) // => {(cat, 3), (dog, 1)}

pets.groupByKey() // => {(cat, Seq(1, 2)), (dog, Seq(1)}

pets.sortByKey()  // => {(cat, 1), (cat, 2), (dog, 1)}
```

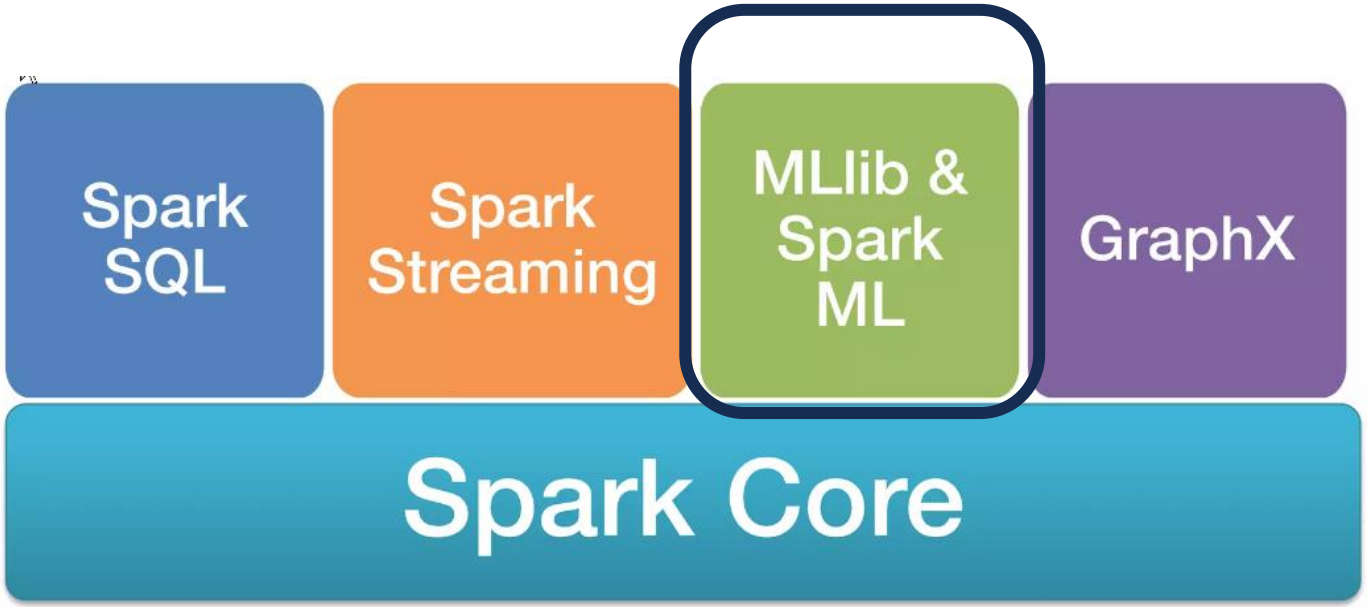
reduceByKey also automatically implements combiners on the map side

Spark's WordCount (Map Reduce)

Example: Word Count

```
val lines = sc.textFile("hamlet.txt")

val counts = lines.flatMap(line => line.split(" "))
                  .map(word => (word, 1))
                  .reduceByKey(_ + _)
```






```python
# In Python
from pyspark.ml.classification import LogisticRegression
...
training = spark.read.csv("s3://...")
test = spark.read.csv("s3://...")

# Load training data
lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)

# Fit the model
lrModel = lr.fit(training)

# Predict
lrModel.transform(test)
...
```

Using Spark MLIB is Easy

Spark Example 1 – Movie Ratings

The RDD approach

```
vkm@VKM-Surface-21:~/netflix$ more ratings*
from pyspark import SparkConf, SparkContext
import collections

conf = SparkConf().setMaster("local").setAppName("RatingsHistogram")
sc = SparkContext (conf = conf)

lines = sc.textFile("file:///home/vkm/netflix/ml-100k/u.data")
ratings = lines.map(lambda x: x.split() [2])
result = ratings.countByValue()

sortedResults = collections.OrderedDict(sorted(result.items()))
for key, value in sortedResults.items() :
    print("_____")
    print("ECE 4150 Presents _____")
    print("%s %i" % (key, value))

vkm@VKM-Surface-21:~/netflix$ _
```

Spark Example 2 - Most Popular Movie

```
vkm@VKM-Surface-21:~/netflix$ more popular_most.py
from pyspark.sql import SparkSession
from pyspark.sql import functions as func
from pyspark.sql.types import StructType, StructField, IntegerType, LongType

spark = SparkSession.builder.appName("PopularMovies").getOrCreate()

# Create schema when reading u.data
schema = StructType([ \
                     StructField("userID", IntegerType(), True), \
                     StructField("movieID", IntegerType(), True), \
                     StructField("rating", IntegerType(), True), \
                     StructField("timestamp", LongType(), True)])

# Load up movie data as dataframe
moviesDF = spark.read.option("sep", "\t").schema(schema).csv("file:///home/vkm/netflix/ml-100k/u.data")

# Some SQL-style magic to sort all movies by popularity in one line!
topMovieIDs = moviesDF.groupBy("movieID").count().orderBy(func.desc("count"))

# Grab the top 10
topMovieIDs.show(10)

# Stop the session
spark.stop()
vkm@VKM-Surface-21:~/netflix$
```

Marvel+Graph.txt

```
vkm@VKM-Surface-21:~/netflix$ more Marvel+Graph.txt
5988 748 1722 3752 4655 5743 1872 3413 5527 6368 6085 4319 4728 1636 2397 3364 4001 1614 1819 1585 732 2660 3952 2507 38
91 2070 2239 2602 612 1352 5447 4548 1596 5488 1605 5517 11 479 2554 2043 17 865 4292 6312 473 534 1479 6375 4456
5989 4080 4264 4446 3779 2430 2297 6169 3530 3272 4282 6432 2548 4140 185 105 3878 2429 1334 4595 2767 3956 3877 4776 49
46 3407 128 269 5775 5121 481 5516 4758 4053 1044 1602 3889 1535 6038 533 3986
5982 217 595 1194 3308 2940 1815 794 1503 5197 859 5096 6039 2664 651 2244 528 284 1449 1097 1172 1092 108 3405 5204 387
 4607 4545 3705 4930 1805 4712 4404 247 4754 4427 1845 536 5795 5978 533 3984 6056
5983 1165 3836 4361 1282 716 4289 4646 6300 5084 2397 4454 1913 5861 5485
5980 2731 3712 1587 6084 2472 2546 6313 875 859 323 2664 1469 522 2506 2919 2423 3624 5736 5046 1787 5776 3245 3840 2399

5981 3569 5353 4087 2653 2058 2218 5354 5306 3135 4088 4869 2958 2959 5732 4076 4155 291
5986 2658 3712 2650 1265 133 4024 6313 3120 6066 3546 463 545 4860 4337 2295 5467 128 2399 5999 5516 5309 4731 2557 5013
 4132 5306 5615 2397 945 533 5694 824 1383 3771 592 5017 704 3778 1127 1480 274 5768 6148 4204 5250 4804 1715 2069 2548
525 2664 528 522 4978 6306 1259 5002 449 2449 1231 3662 3959 2603 2931 3319 3955 3210 5776 5088 2273 5576 1649 518 1535
3356 5874 5973 1660 4359 4188 2614 2613 3594 3805 3750 331 3757 1347 4366 66 2199 3296 3008 1425 3454 1638 1587 731 183
2 2689 505 5021 2629 5834 4441 2184 4607 4603 5716 969 867 6196 604 2438 155 2430 3632 5446 5696 4454 3233 6227 1116 117
7 563 2728 5736 4898 859 5535 5046 2971 1805 1602 1289 3220 4589 3989 5931 3986 1369
5987 2614 5716 1765 1818 2989 6436 1587 6451 5661 4069 1962 66 6084 6148 2051 4045 6005 6419 6432 3626 2119 6417 5707 29
87 3388 6438 2666 1380 6066 725 1469 2508 5905 5332 6059 2107 197 6857 3762 2467 2723 2810 5099 3935 208 2422 611 713 13
31 1330 2183 4621 5736 1420 5323 2449 2506 3724 2452 2350 58 1645 1647 1554 3354 1766 2896 4070 53 5815 4480 4898 260 49
```

Superhero

Superheros associated with…

Marvel+Names.txt

```
vkm@VKM-Surface-21:~/netflix$ more Marvel+Names.txt
1 "24-HOUR MAN/EMMANUEL"
2 "3-D MAN/CHARLES CHAN"
3 "4-D MAN/MERCURIO"
4 "8-BALL/"
5 "A"
6 "A'YIN"
7 "ABBOTT, JACK"
8 "ABCISSA"
9 "ABEL"
10 "ABOMINATION/EMIL BLO"
11 "ABOMINATION | MUTANT"
12 "ABOMINATRIX"
13 "ABRAXAS"
14 "ADAM 3,031"
15 "ABSALOM"
16 "ABSORBING MAN/CARL C"
17 "ABSORBING MAN | MUTA"
18 "ACBA"
19 "ACHEBE, REVEREND DOC"
20 "ACHILLES"
21 "ACHILLES II/HELMUT"
22 "ACROBAT/CARL ZANTE"
23 "ADAM X"
24 "ADAMS, CINDY"
```

Spark Example 3 – Most Popular Superhero

vkm@VKM-Surface-21: ~/netflix

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as func
from pyspark.sql.types import StructType, StructField, IntegerType, StringType

spark = SparkSession.builder.appName("MostPopularSuperhero").getOrCreate()

schema = StructType([ \
                     StructField("id", IntegerType(), True), \
                     StructField("name", StringType(), True)])

names = spark.read.schema(schema).option("sep", " ").csv("file:///home/vkm/netflix/Marvel+Names.txt")

lines = spark.read.text("file:///home/vkm/netflix/Marvel+Graph.txt")

connections = lines.withColumn("id", func.split(func.trim(func.col("value")), " ")[0]) \
    .withColumn("connections", func.size(func.split(func.trim(func.col("value")), " ")) - 1) \
    .groupBy("id").agg(func.sum("connections").alias("connections"))

mostPopular = connections.sort(func.col("connections").desc()).first()

mostPopularName = names.filter(func.col("id") == mostPopular[0]).select("name").first()

print("_____________________________________________")
print("_____________________________________________")
print("You had to wait till you took ECE 4150 to know that....")
print(mostPopularName[0] + " is the most popular superhero with " + str(mostPopular[1]) + " co-appearances.")
print(mostPopularName[0] + " is the most popular superhero with " + str(mostPopular[1]) + " co-appearances.")
print("_____________________________________________")
print("_____________________________________________")
                                                                    23,1          All
```

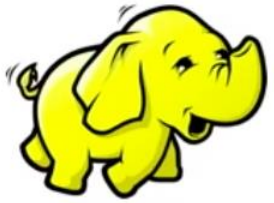
Spark Example 4 – Min Temperature Sensor

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as func
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType

spark = SparkSession.builder.appName("MinTemperatures").getOrCreate()

schema = StructType([ \
                     StructField("stationID", StringType(), True), \
                     StructField("date", IntegerType(), True), \
                     StructField("measure_type", StringType(), True), \
                     StructField("temperature", FloatType(), True)])

# // Read the file as dataframe
df = spark.read.schema(schema).csv("file:///home/vkm/netflix/1800.csv")
df.printSchema()

# Filter out all but TMIN entries
minTemps = df.filter(df.measure_type == "TMIN")

# Select only stationID and temperature
stationTemps = minTemps.select("stationID", "temperature")

# Aggregate to find minimum temperature for every station
minTempsByStation = stationTemps.groupBy("stationID").min("temperature")
minTempsByStation.show()

# Convert temperature to fahrenheit and sort the dataset
minTempsByStationF = minTempsByStation.withColumn("temperature",
                                                  func.round(func.col("min(temperature)") * 0.1 * (9.0 / 5.0) + 32.0, 2))\
                                                  .select("stationID", "temperature").sort("temperature")

# Collect, format, and print the results
results = minTempsByStationF.collect()

for result in results:
    print(result[0] + "\t{:.2f}F".format(result[1]))

spark.stop()
```

Spark Example 5 – Word Counter

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as func

spark = SparkSession.builder.appName("WordCount").getOrCreate()

# Read each line of my book into a dataframe
inputDF = spark.read.text("file:///home/vkm/netflix/book.txt")

# Split using a regular expression that extracts words
words = inputDF.select(func.explode(func.split(inputDF.value, "\\W+")).alias("word"))
words.filter(words.word != "")

# Normalize everything to lowercase
lowercaseWords = words.select(func.lower(words.word).alias("word"))

# Count up the occurrences of each word
wordCounts = lowercaseWords.groupBy("word").count()

# Sort by counts
wordCountsSorted = wordCounts.sort("count")

# Show the results.
wordCountsSorted.show(wordCountsSorted.count())
```

Spark Hadoop & MapReduce



Spark Streaming

Apache Storm

batch processing

batch + stream processing

stream processing

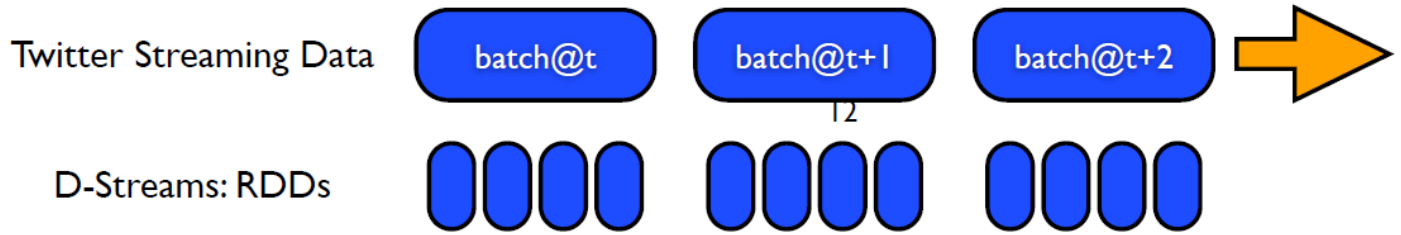
Spark Streaming

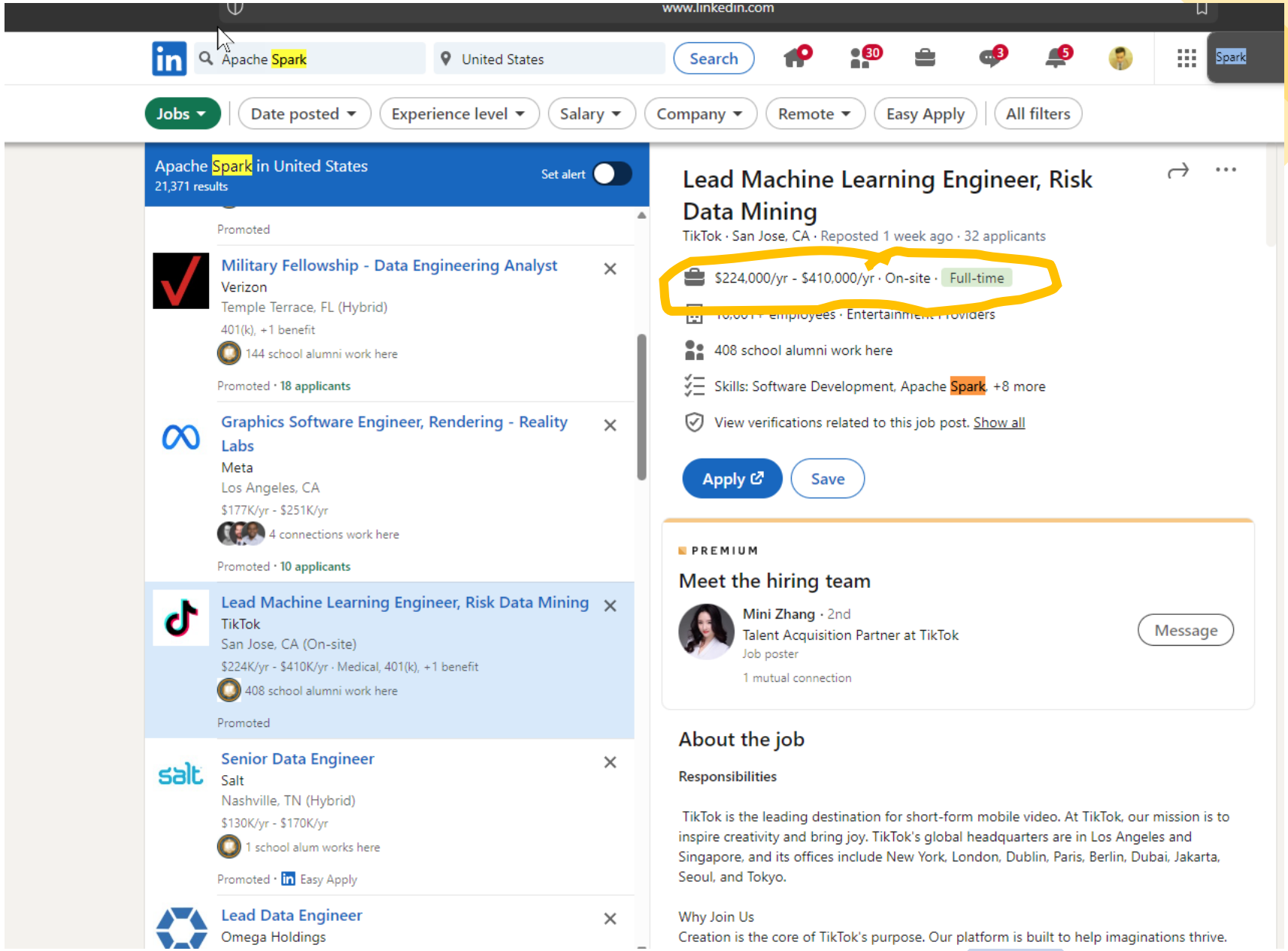
- (Near) real-time processing of stream data
- New programming model
  - Discretized streams (D-Streams)
  - Built on Resilient Distributed Datasets (RDDs)
- Based on Spark
- Integrated with Spark batch and interactive computation modes

Spark Streaming

- D-Streams
  - Treat a streaming computation as a series of deterministic batch computations on a small time intervals
  - Latencies can be as low as a second, supported by the fast execution engine Spark

```
val ssc = new StreamingContext(sparkUrl, "Tutorial", Seconds(1), sparkHome, Seq(jarFile))
val tweets = ssc.twitterStream(twitterUsername, twitterPassword)
val statuses = tweets.map(status => status.getText())
statuses.print()
```




www.linkedin.com
Apache Spark
United States
Search
Jobs
Date posted
Experience level
Salary
Company
Remote
Easy Apply
All filters
Apache Spark in United States
21,371 results
Set alert
Promoted
Military Fellowship - Data Engineering Analyst
Verizon
Temple Terrace, FL (Hybrid)
401(k), +1 benefit
144 school alumni work here
Promoted · 18 applicants
Graphics Software Engineer, Rendering - Reality Labs
Meta
Los Angeles, CA
$177K/yr - $251K/yr
4 connections work here
Promoted · 10 applicants
Lead Machine Learning Engineer, Risk Data Mining
TikTok
San Jose, CA (On-site)
$224K/yr - $410K/yr · Medical, 401(k), +1 benefit
408 school alumni work here
Promoted
Senior Data Engineer
Salt
Nashville, TN (Hybrid)
$130K/yr - $170K/yr
1 school alum works here
Promoted · Easy Apply
Lead Data Engineer
Omega Holdings
Lead Machine Learning Engineer, Risk Data Mining
TikTok · San Jose, CA · Reposted 1 week ago · 32 applicants
$224,000/yr - $410,000/yr · On-site · Full-time
408 school alumni work here
Skills: Software Development, Apache Spark, +8 more
View verifications related to this job post. Show all
Apply
Save
PREMIUM
Meet the hiring team
Mini Zhang · 2nd
Talent Acquisition Partner at TikTok
Job poster
1 mutual connection
Message
About the job
Responsibilities
TikTok is the leading destination for short-form mobile video. At TikTok, our mission is to inspire creativity and bring joy. TikTok's global headquarters are in Los Angeles and Singapore, and its offices include New York, London, Dublin, Paris, Berlin, Dubai, Jakarta, Seoul, and Tokyo.
Why Join Us
Creation is the core of TikTok's purpose. Our platform is built to help imaginations thrive.

Summary

- Spark is a very powerful environment for implementing big data analytics
- It is fast, has more primitives that EMR, and also supports SQL, ML-Libraries and In-Memory Processing for speed