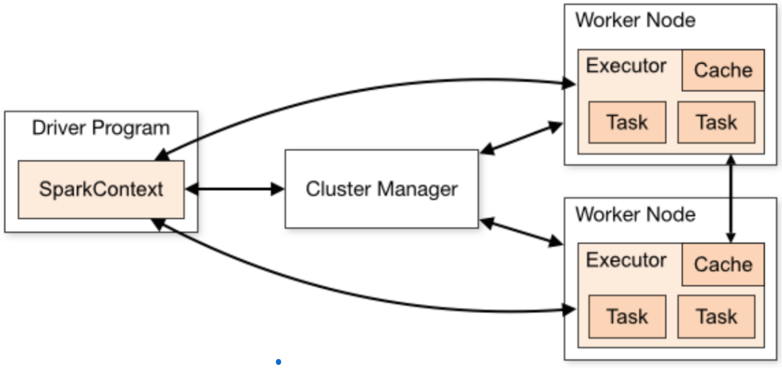
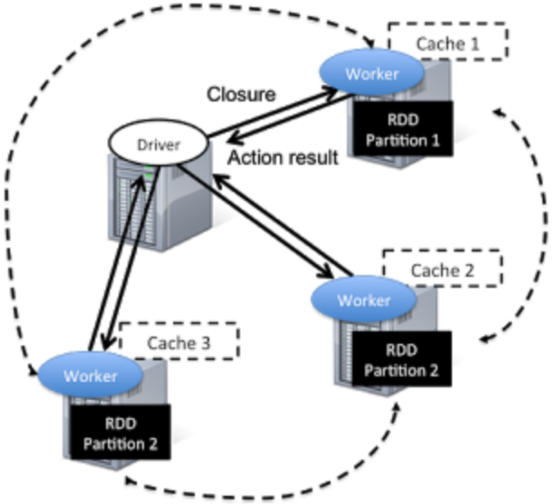

Spark
Marilyn Waldman
@mdwaldman

Agenda

- What is Spark - Why Spark
- lab 1 - Functional programming -map and reduce
- lab 2 - RDD's
- lab 3 - Word Count
- lab 4 - Spark SQL
- Conclusions

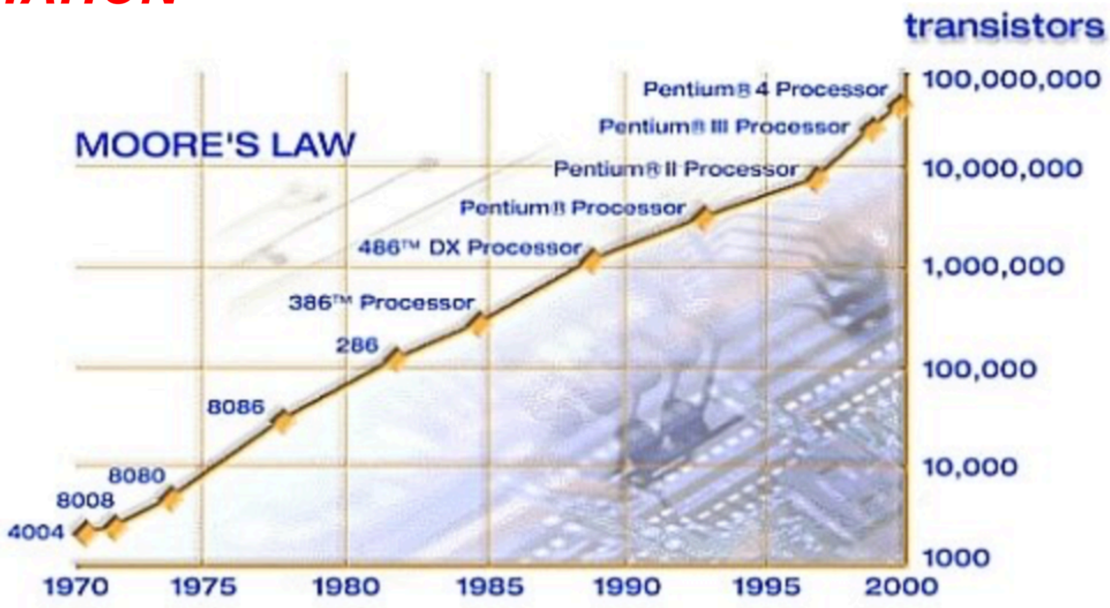
Cluster computing platform, a ***distributed system***





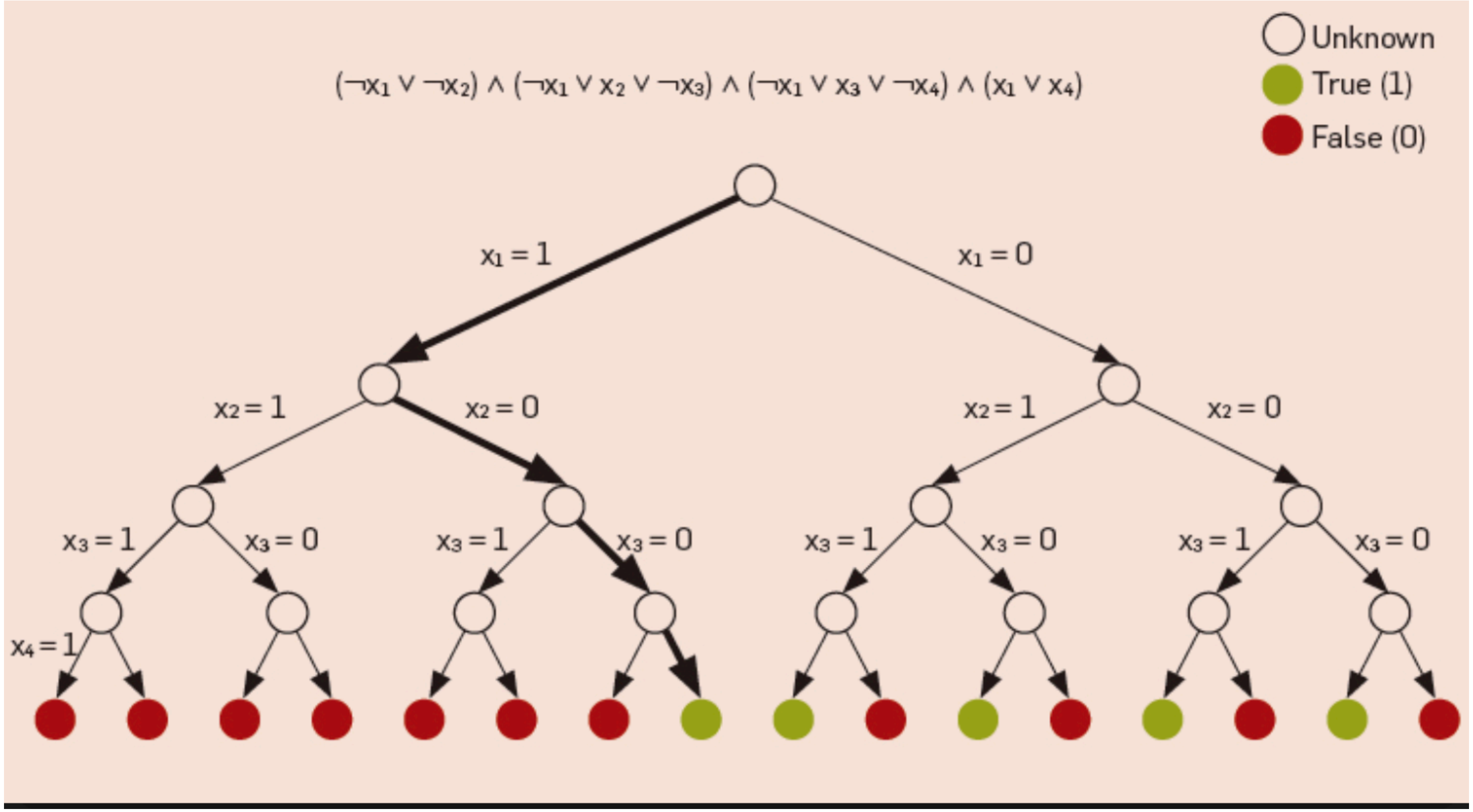
- genomics
- regression - optimization
- real-time data
- anomaly detection
- fraud detection
- codeneuro

COMPUTATION



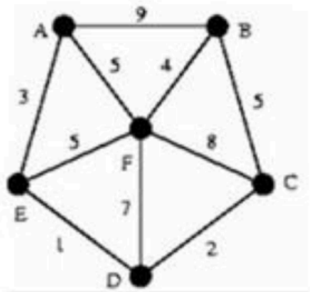
Moore's Law states that the number of transistors that can be placed on an integrated circuit for the same price will increase exponentially by a factor of 2 every 18 to 24 months. In other words, put simply *Moore's Law* claims that CPU processing power will double approximately every two years for the price of 1,000 dollars. (Graph copyright Ray Kurzweil)

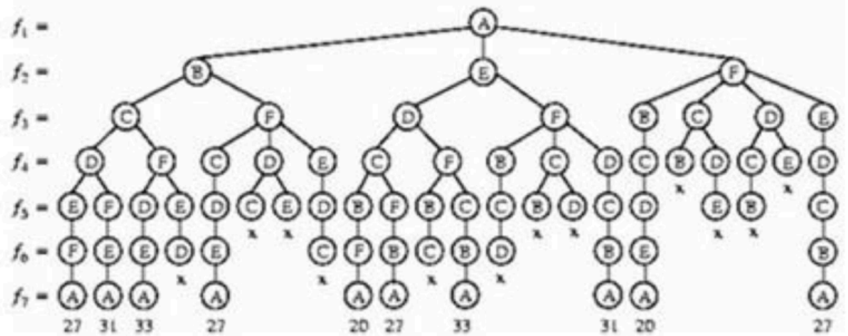
NP-HARD and Intractable



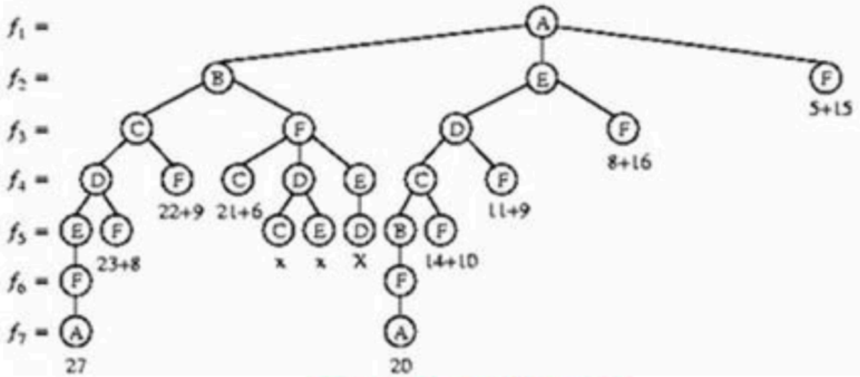
Exhaustive Search v.s. Branch and Bound

.TSP example





Backtracking/exhaustive search

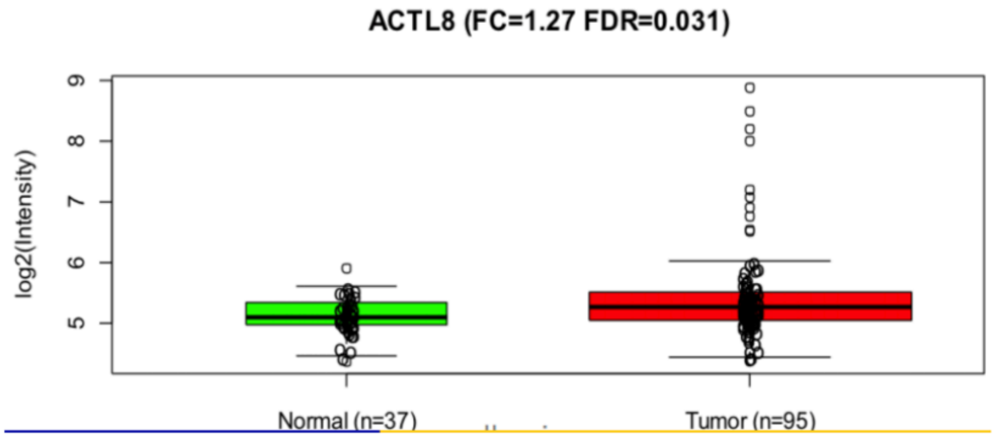


Branch and bound

DATA

At its core, COPA is a technique for analyzing key-vale pairs of gene expression data and to detect outliers, which are the candidates for cancer. A parallelized algorithm was necessary owing to the size of the data involved, says Parsian, who also teaches at the University of Santa Clara.

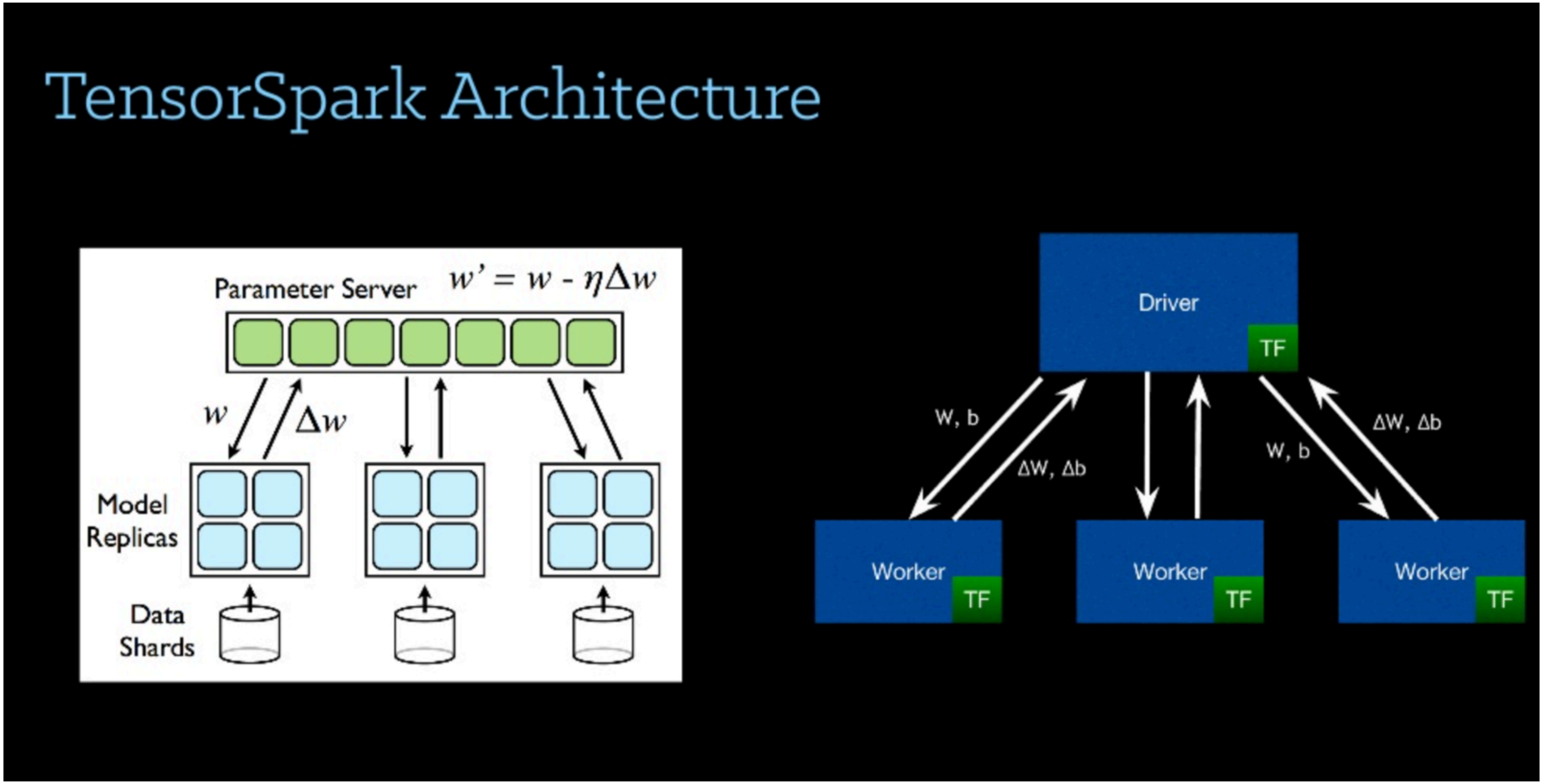
- COPA = Cancer Outlier Profile Analysis
- Statistics designed to identify outliers in cancer gene expression profile



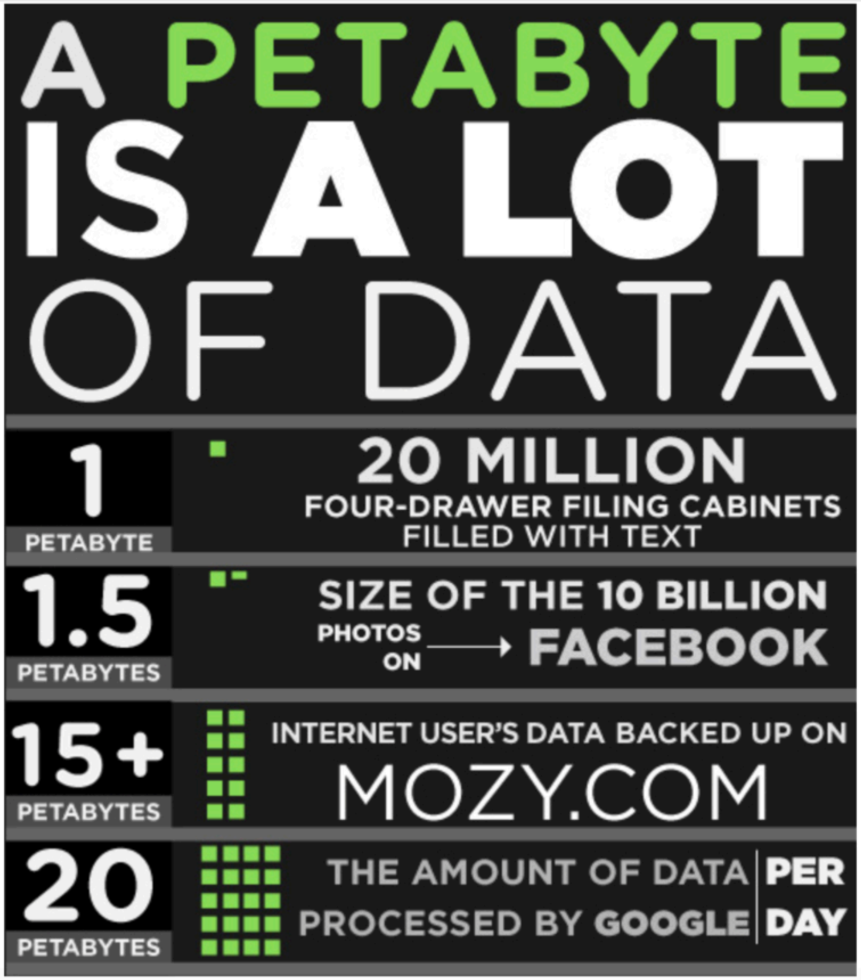
COPA has proven its value in detecting genetic mutations linked to prostate cancer, and is now being studies with other types of cancers

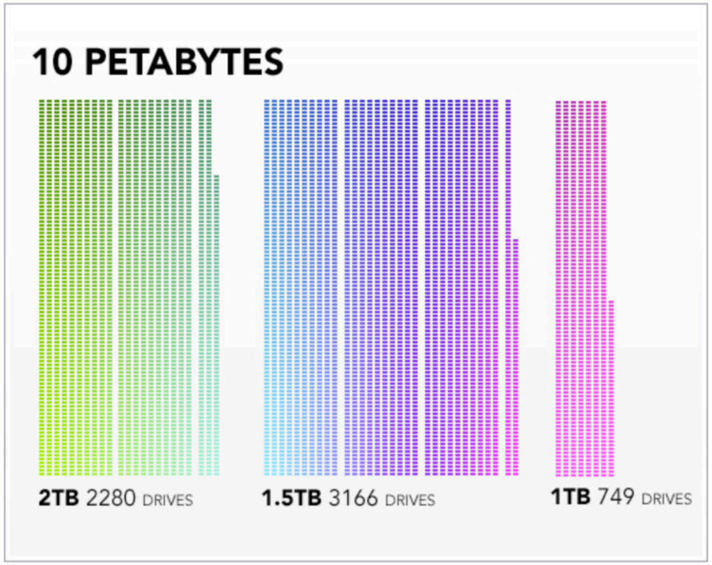
"Of course, if the data size is small, you can detect it visually or by writing some sample programs," Parsian says. "But when you have terabytes of data you're analyzing, detecting mutation is impossible to do visually."

There are many ways to solve this problem, and any number of algorithms could do the



credit: ***http://www.slideshare.net/arimoinc/distributed-tensorflow-scaling-googles-deep-learning-library-on-spark-***


A PETABYTE
IS A LOT
OF DATA
1
PETABYTE
20 MILLION
FOUR-DRAWER FILING CABINETS
FILLED WITH TEXT
1.5
PETABYTES
SIZE OF THE 10 BILLION
PHOTOS ON
FACEBOOK
15+
PETABYTES
INTERNET USER'S DATA BACKED UP ON
MOZY.COM
20
PETABYTES
THE AMOUNT OF DATA
PROCESSED BY GOOGLE
PER
DAY


10 PETABYTES
2TB 2280 DRIVES
1.5TB 3166 DRIVES
1TB 749 DRIVES

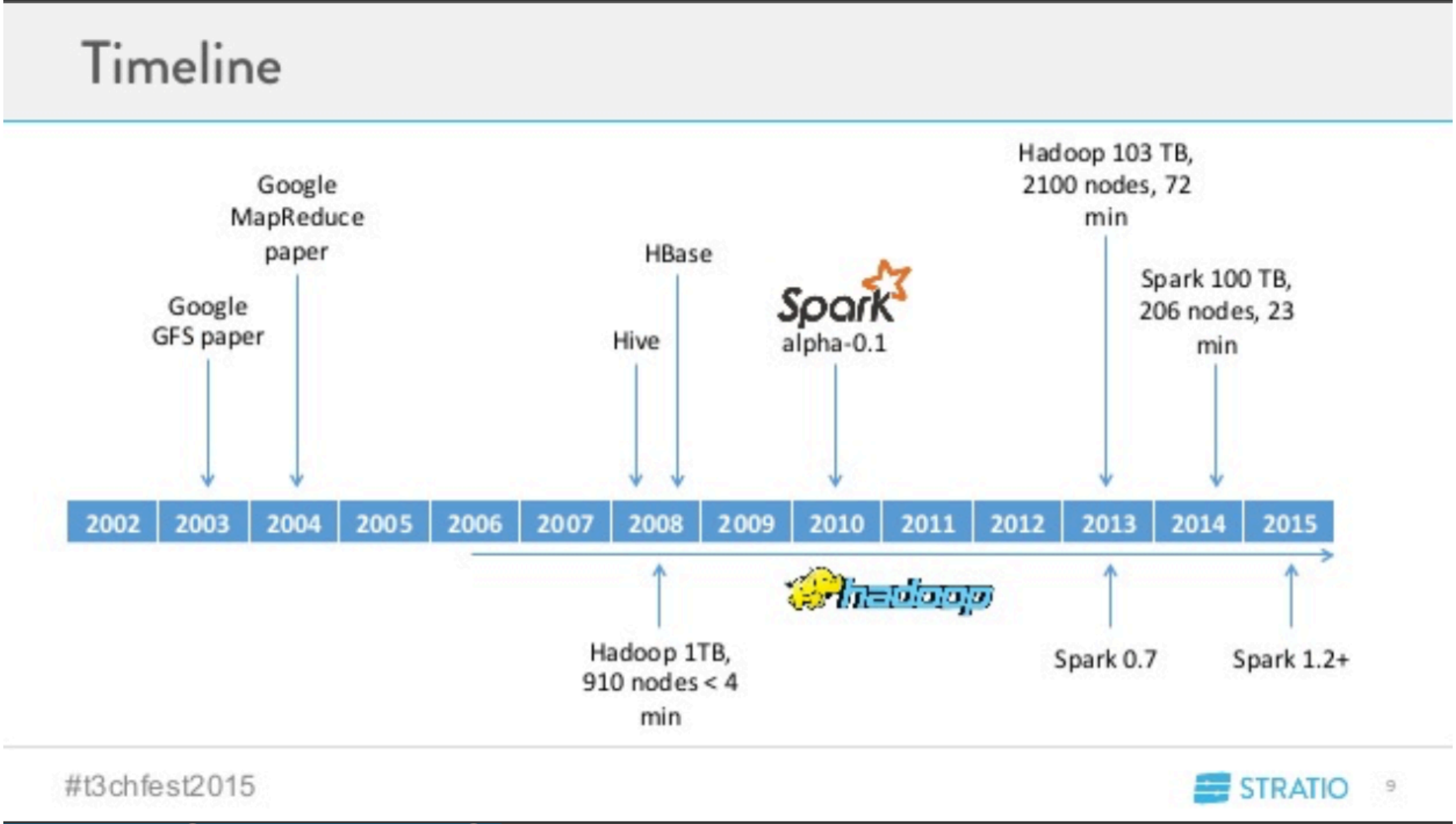
Dijkstra - Cooperating Sequential Processes

communicate

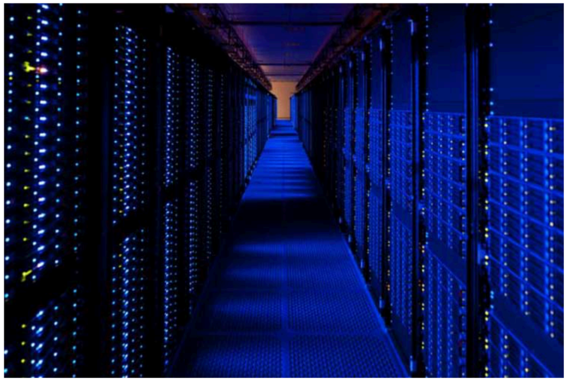
cooperate

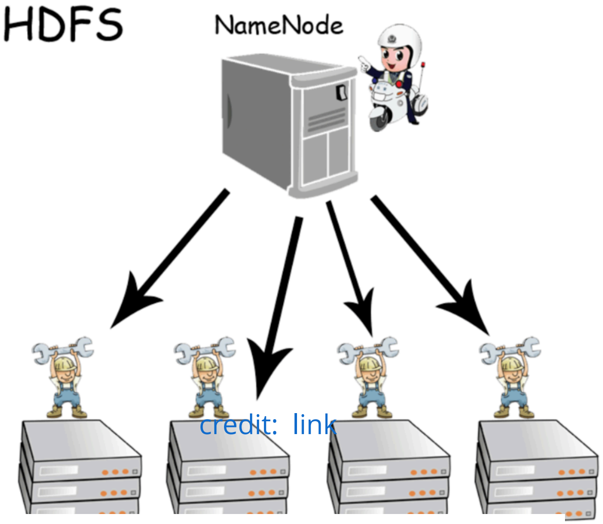
synchronize

Caution: Distributed Processing is a hard problem. Keeping track of all the moving parts is challenging.



credit : Adios hadoop, Hola Spark! T3chfest 2015





The project includes these modules:

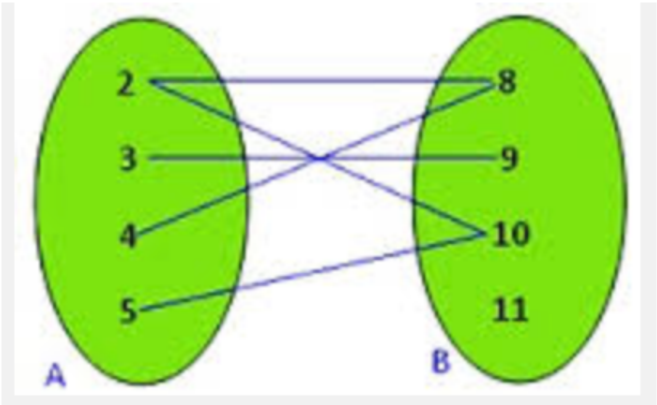
- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

- Began at UC Berkeley in 2009
- Fast and general purpose cluster computing
- 10x faster on disk. 100x faster in-memory
- Integrates with Hadoop and can read existing data
- API's - Java, Python, Scala
- Deeply embraced due to ***elegance*** of use

Functional Programming

vs

Imperative Programming



Functional


```
range = domain.map(lambda x : x*x)
```

domain is immutable

range is a new fresh object

required for parallel processing

Imperative

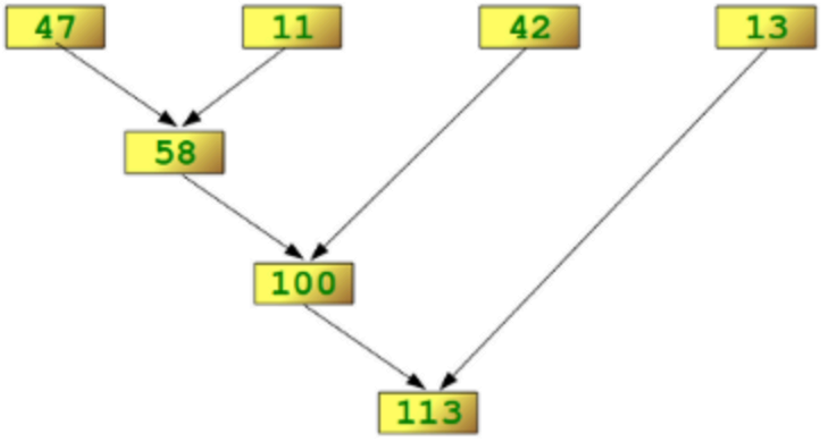
```
for i in domain:
        domain[i] = domain[i] * domain[i]
```

danger: side effects, mutations

We illustrate this process in the following example:

```
>>> reduce(lambda x,y: x+y, [47,11,42,13])
113
```

The following diagram shows the intermediate steps of the calculation:



credit: http://www.python-course.eu/lambda.php

```
In [17]: reduced = mappedRdd.reduce(lambda x, y: x+y)
         print type(reduced)
         print reduced

         <type 'int'>
         328350
```

jupyter lab1-map-reduce Last Checkpoint: a minute ago (unsaved changes)

File Edit View Insert Cell Kernel Help | Python 2

Code | Cell Toolbar: None

Introduction to the Map, Reduce and Filter Abstractions

The Map abstraction

In [1]:

```
#Initialize data
nums = range(10)
print (nums)
print (type(nums))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
<type 'list'>
```

In [2]:

```
#Python only

map(lambda x: x*x, nums)
```

Out[2]:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [3]:

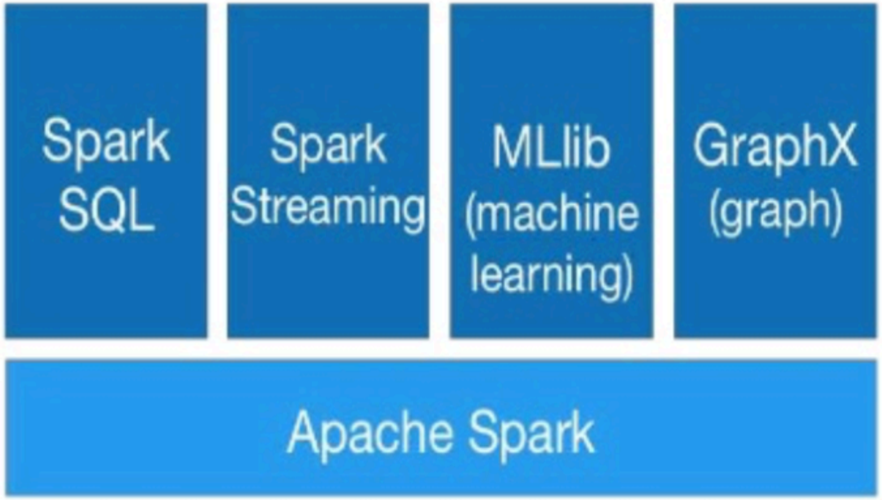
```
#use a function definition.  This is called a clousure
def square(x):
    return x*x
```

In [4]:

```
#python only
results = map(square, nums)
for num in results:
```

Spark Stack

- Spark SQL
  - For SQL and unstructured data processing
- MLib
  - Machine Learning Algorithms
- GraphX
  - Graph Processing
- Spark Streaming
  - stream processing of live data streams



http://spark.apache.org

1. Most machine learning programs are iterative. Each iteration improves results
2. With MapReduce each iteration is written to disk. This is expensive.
3. Spark runs ***in memory*** using an abstraction known as an ***RDD***

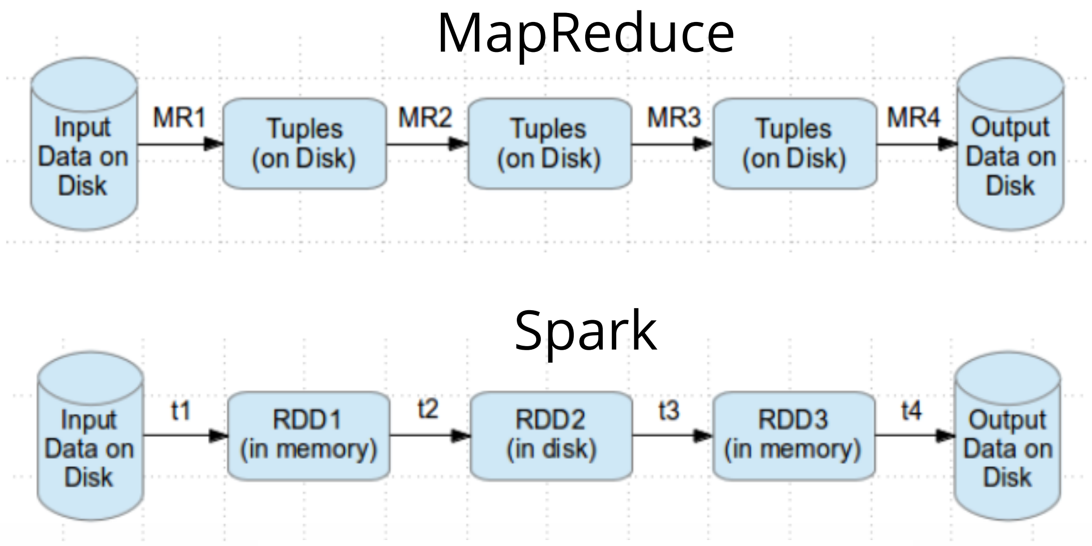
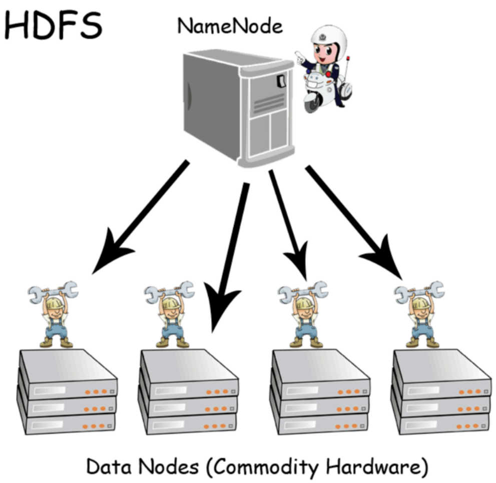


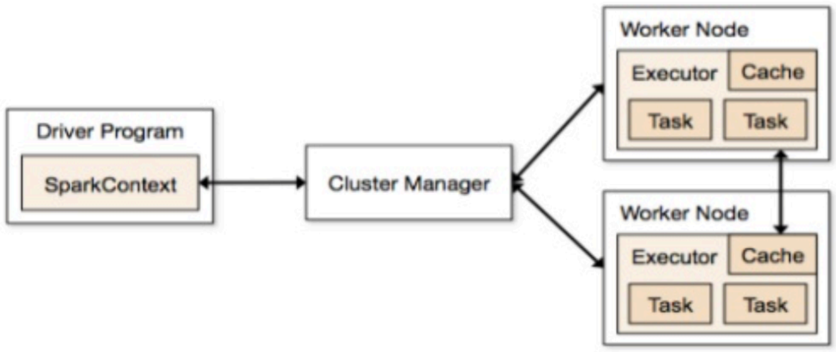
Image Credits: Datatamasha.com

Take the compute to the data



HDFS (Hadoop Distributed File System)

Execution Flow



http://spark.apache.org/docs/latest/cluster-overview.html

Coding Exercise: *WordCount*

```java
public class WordCount {
  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length < 2) {
      System.err.println("Usage: wordcount <in> [<in>...] <out>");
      System.exit(2);
    }
    Job job = new Job(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    for (int i = 0; i < otherArgs.length - 1; ++i) {
      FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
    }
    FileOutputFormat.setOutputPath(job,
      new Path(otherArgs[otherArgs.length - 1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

```scala
val f = sc.textFile(inputPath)
val w = f.flatMap(l => l.split(" ")).map(word => (word, 1)).cache()
w.reduceByKey(_ + _).saveAsText(outputPath)
```

WordCount in 3 lines of Spark

WordCount in 50+ lines of Java MR

credit: Sparkcamp @ Strata CA: Intro to Apache Spark with Hands-on Tutorials

The Magic of Spark

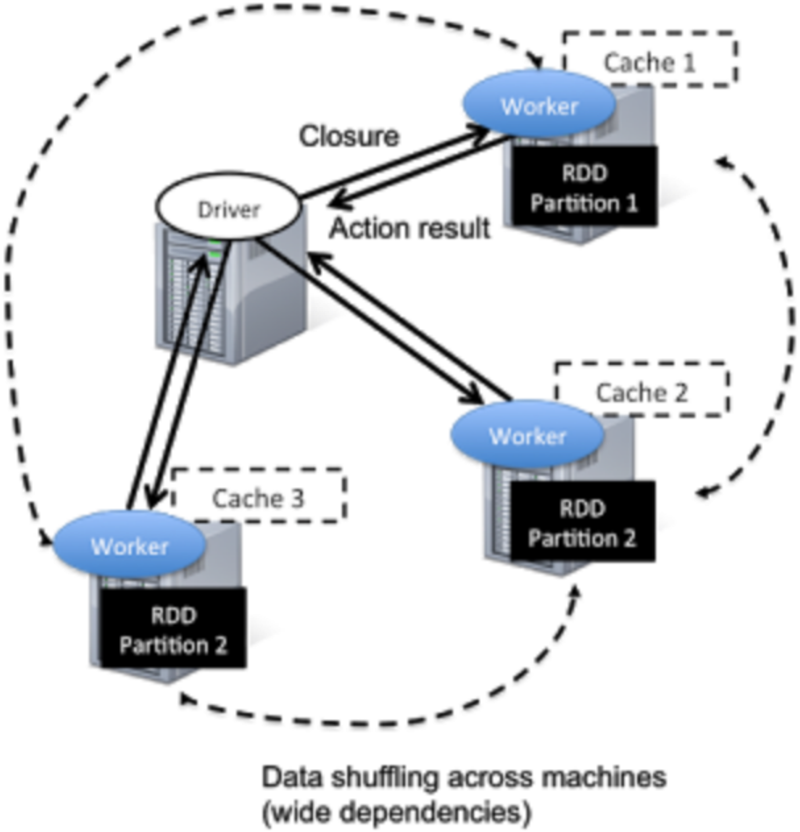
RDD's

Transformations

Actions

Resilient Distributed Dataset (RDD)

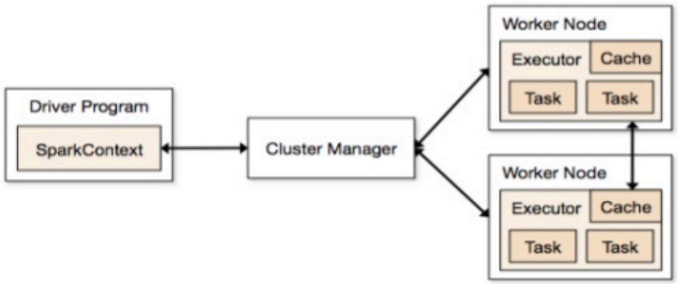
- ***RDD*** is a basic abstraction
- ***Immutable***, partitioned collection of elements that can be operated in parallel
- Basic operations - map, filter, persist
- Multiple implementations - PairRDD <key,value> and Sequence Files



How do I create an RDD?

1. **Parallelized Collections**
2. **External Datasets**
3. **Streaming Data**

Execution Flow




```
data = [1, 2, 3, 4, 5]
distData = sc.parallelize(data)
```

http://spark.apache.org/docs/latest/cluster-overview.html

```
>>> distFile = sc.textFile("data.txt")
```

What can I do with an RDD?

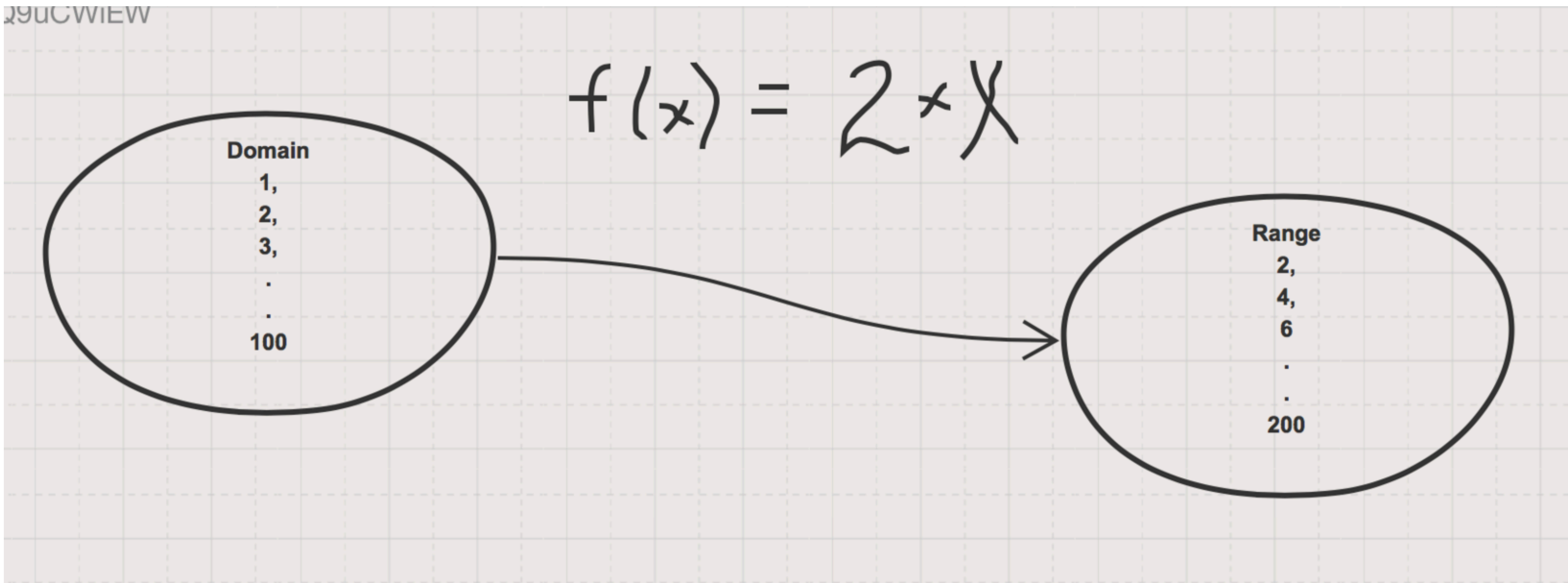
RDD Operations

RDDs support two types of operations: *transformations*, which create a new dataset from an existing one, and *actions*, which return a value to the driver program after running a computation on the dataset.

Map is a transformation that passes each dataset element through a function and returns a new RDD representing the results.

Reduce is an action that aggregates all the elements of the RDD using some function and returns the final result to the driver program.

What functions? Monoids



range = domain.map(lambda x : 2*x)

sum = domain.reduce(lambda x,y : x+y)

RDD's are a collection of records

```
rdd = sc.parallelize(range(1000),5)
```

Transformations create new RDD's from existing ones

```
errors = rdd.filter(lambda line: "ERROR" in line)
```

Actions materialize a value in the user program

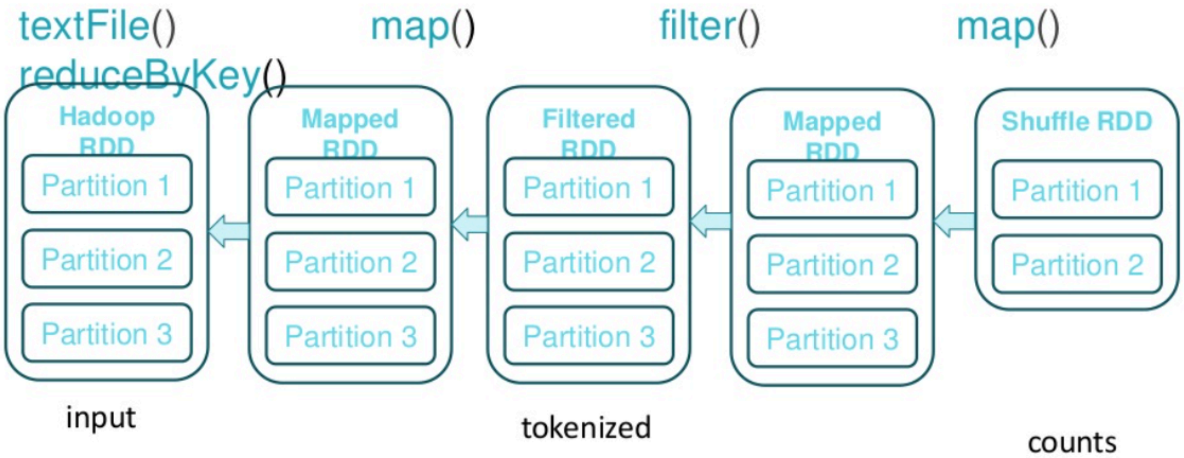
```
size = errors.count()
```

THE DAG

(directed acyclic graph)

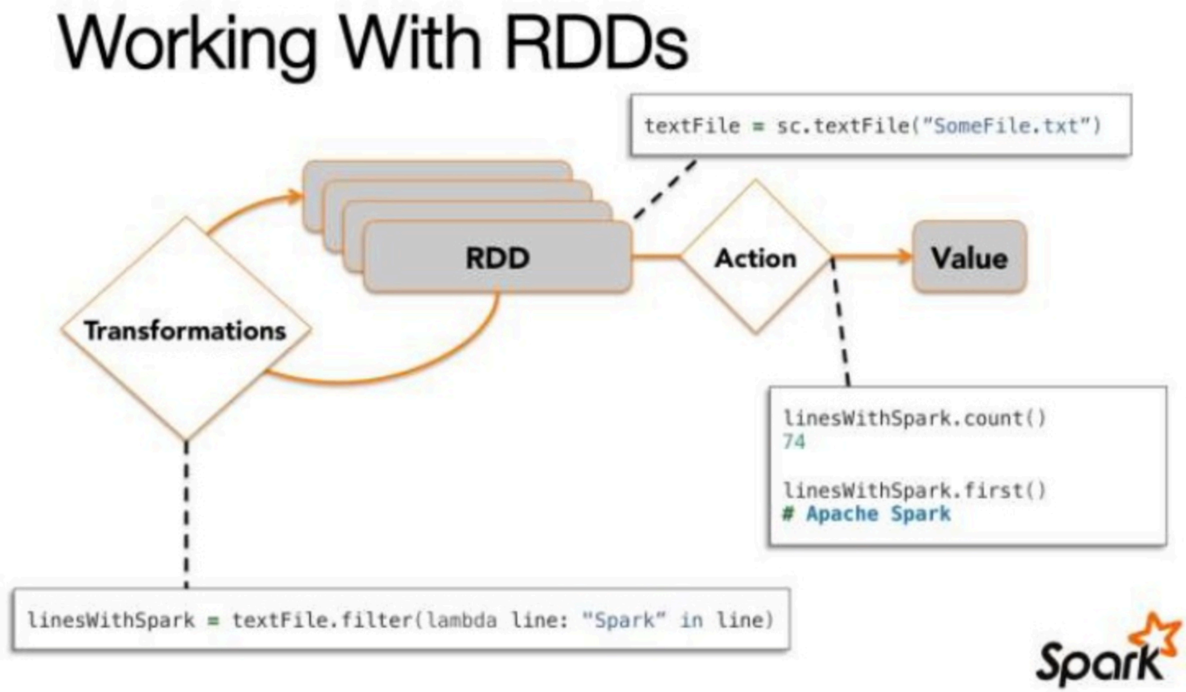
sc.textFile().map().filter().map().reduceByKey()

DAG View of RDD's



credit: data bricks

Lazy Evaluation



Code runs only upon encountering an action

credit: link

Persistence layers for Spark

Distributed

- Hadoop (HDFS)
- Local file system
- Cassandra
- Amazon S3
- Hive
- Base

File formats

- Text - CSV, Plain Txt
- Sequence File
- AvRO
- Parquet

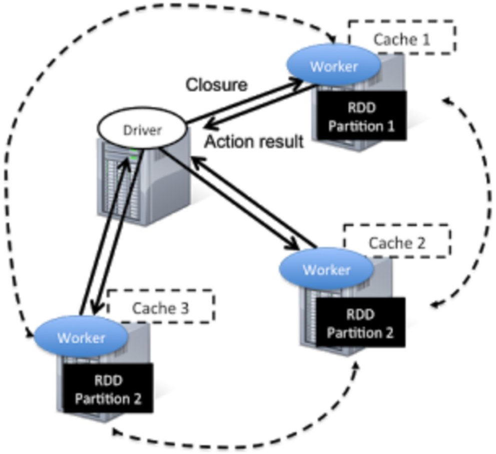
WHEN IS the DAG EXECUTED?

```python
import sys
import os

logFile = os.path.join('data', 'logfile')

""" Read and parse log file """
parsed_logs = (sc
         .textFile(logFile)
         .filter(lambda line: 'GET' in line))

print parsed_logs.count()
```



Data shuffling across machines
(wide dependencies)

```
in24.inetnebr.com - - 01/Aug/1995:00:00:01 -0400 "GET /shuttle/missions/sts-6.txt HTTP/1.0" 200 1839
uplherc.upl.com - - 01/Aug/1995:00:00:07 -0400 "GET / HTTP/1.0" 304 0
uplherc.upl.com - - 01/Aug/1995:00:00:08 -0400 "GET /images/ksc.gif HTTP/1.0" 304 0
uplherc.upl.com - - 01/Aug/1995:00:00:08 -0400 "GET /images/MOSAIC.gif HTTP/1.0" 304 0
```

Transformations

```
map(func)
filter(func)
flatMap(func)
mapPartitions(func)
mapPartitionsWithIndex(func)
union(otherDataset)
intersection(otherDataset)
distinct([numTasks]))
groupByKey([numTasks])
sortByKey([ascending], [numTasks])
```

```
reduceByKey(func, [numTasks])
aggregateByKey(zeroValue)(seqOp,
     combOp, [numTasks])
join(otherDataset, [numTasks])
cogroup(otherDataset, [numTasks])
cartesian(otherDataset)
pipe(command, [envVars])
coalesce(numPartitions)
sample(withReplacement,fraction, seed)
repartition(numPartitions)
```

Actions

reduce(func)
collect()
count()
first()
countByKey()
foreach(func)

take(n)
takeSample(withReplacement,num, [seed])
takeOrdered(n, [ordering])
saveAsTextFile(path)
saveAsSequenceFile(path)
(Only Java and Scala)
saveAsObjectFile(path)
(Only Java and Scala)

Lab2

jupyter lab2_sparkRDDs Last Checkpoint: 18 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help | Python 2

Markdown Cell Toolbar: None

In [2]:
```
#range(start, end=None, step=1, numSlices=None)

rdd = sc.parallelize(range(0, 100, 1), 5)
print (rdd.collect())
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31
, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 9
0, 91, 92, 93, 94, 95, 96, 97, 98, 99]
```

What does this look like?

- `glom`: Returns an RDD list from each partition of an RDD.
- `collect`: Returns a list from all elements of an RDD to the DRIVER.

In [3]:
```
for x in rdd.glom().collect():
    print(x)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
[20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]
[40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]
[60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79]
[80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]
```

Use map and square each element of the RDD

In [4]:
```
#two transformations - A DAG with two tasks
mappedRdd = (rdd.map(lambda x: x*x)
                .filter(lambda x : x > 1000))
#one action
```

Key Value Pairs

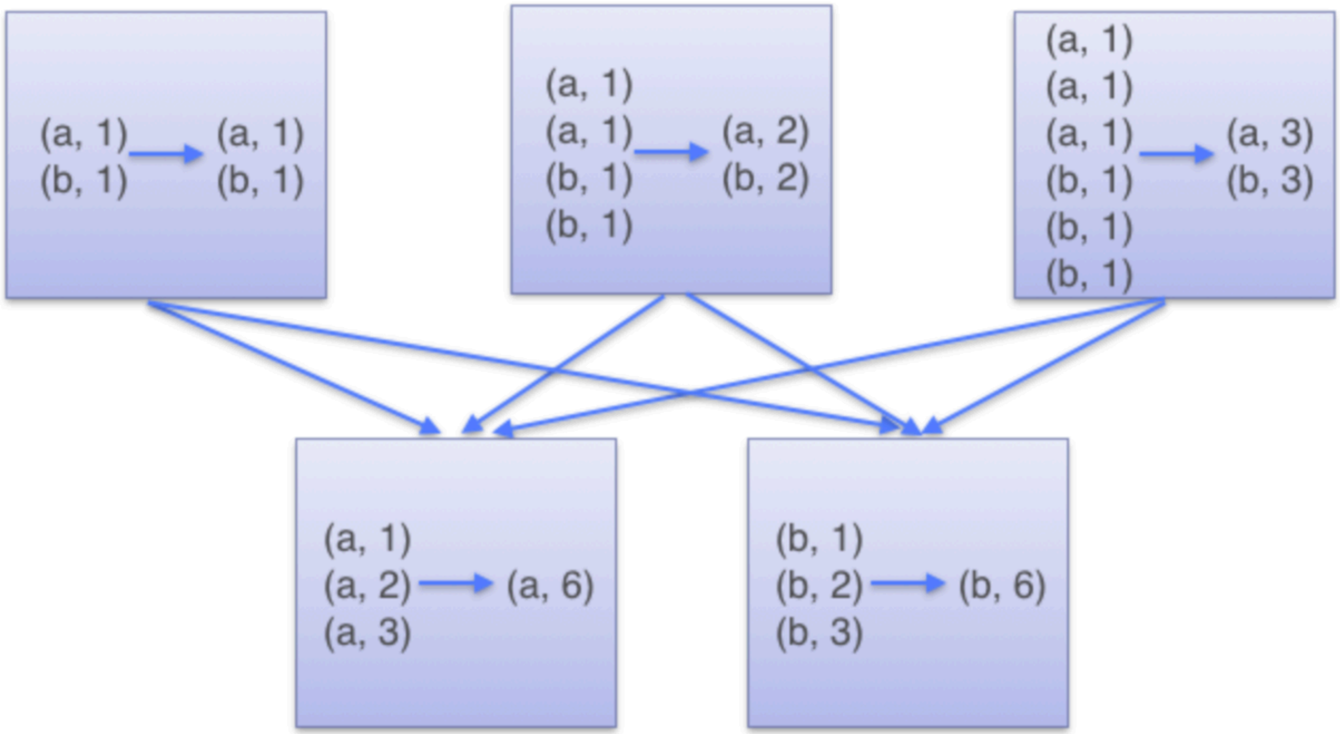
Transformations

- rdd.reduceByKey(func)
- rdd.groupByKey()
- rdd.mapValues(fund)
- rdd.keys()
- rdd.values()
- rdd.sortByKey()

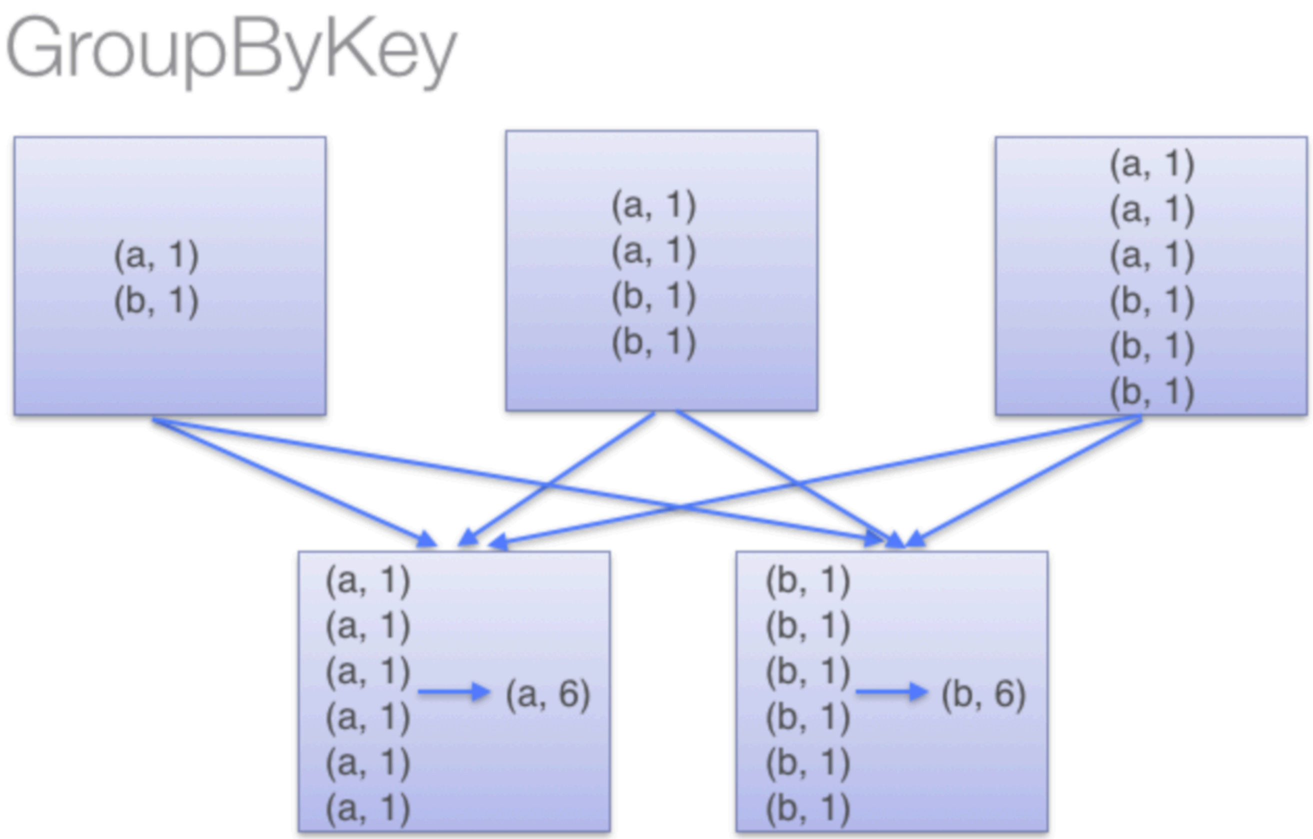
Actions

- rdd.countByKey()
- rdd.collectAsMap()
- rdd.lookup(key)

ReduceByKey



credit: https://databricks.gitbooks.io/databricks-spark-knowledge-base/content/best_practices/prefer_reducebykey_over_groupbykey.html



credit: https://databricks.gitbooks.io/databricks-spark-knowledge-base/content/best_practices/prefer_reducebykey_over_groupbykey.html

```scala
val lines  =  sc.textFile("input.txt")
val words =  lines.flatMap(line => line.split(" "))
val ones   =  words.map(s => (s,1))
val count  =  ones.reduceByKey((a,b) => a + b)
val result =  count.collectAsMap( )
```

RDD lineage DAG built on driver side

data source RDD

transformation RDD, transformation

action, action RDD

Lab 3 WordCount

Cluster Deployment

- Standalone Deploy Mode
  - simplest way to deploy Spark on a private cluster
- Amazon EC2
  - EC2 scripts are available
  - Very quick launching a new cluster
- Apache Mesos
- Hadoop YARN

Conclusion

Software is Eating the World