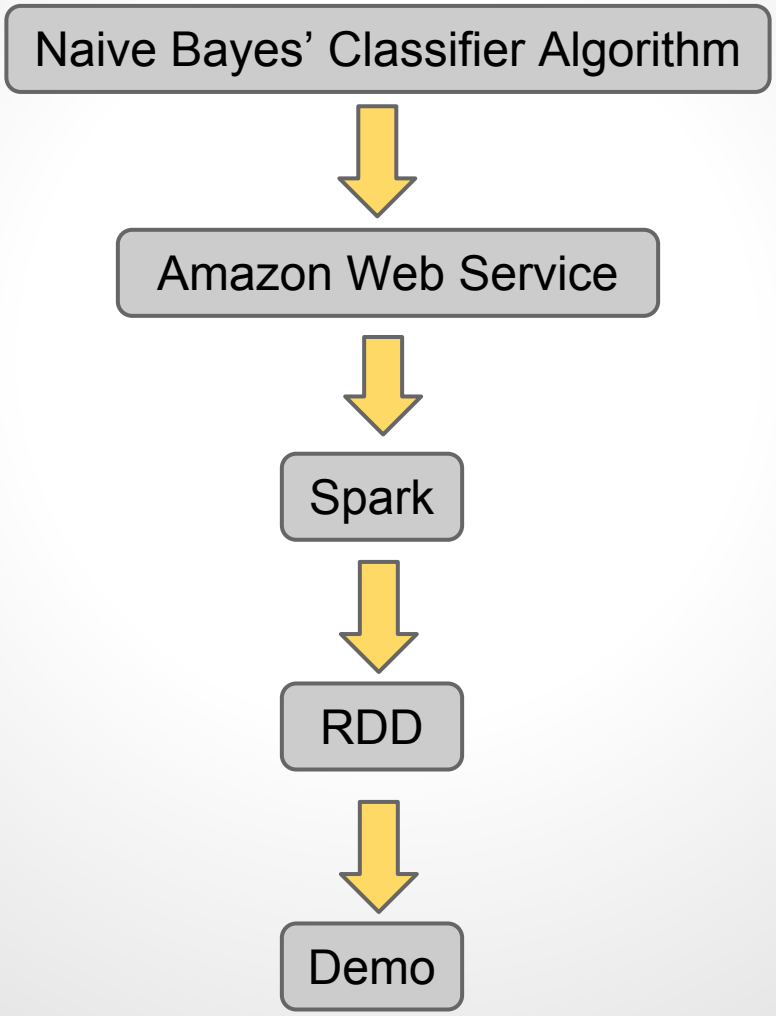
Spark Application on AWS EC2 Cluster

COEN241 - Cloud Computing

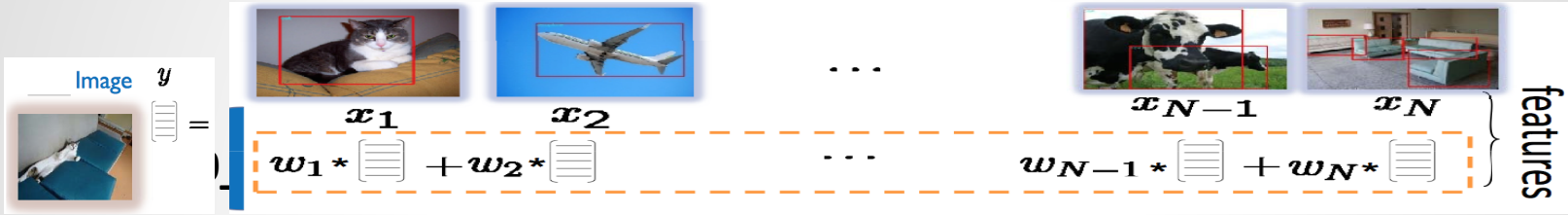
Group 6

Chao-Hsuan Shen, Patrick Loomis, John Geevarghese

Explanation Outline



Raw Data of Imagenet



n01484850_18,470 507 507 988 502 471 809 128 128 177 771 771 771 771 401 417

n01491361_23,491 202 939 57 882 937 752 752 677 748 314 314 794 434 314 314 729 771 104 629 725 771 771 961 771 771 794 314 909 894 551 689 130 684 576 895 605 865 314 314 44 865 314 446 202 910 910 446 446 961 909 910 960 191 314 446 446 321 744 752 957 882

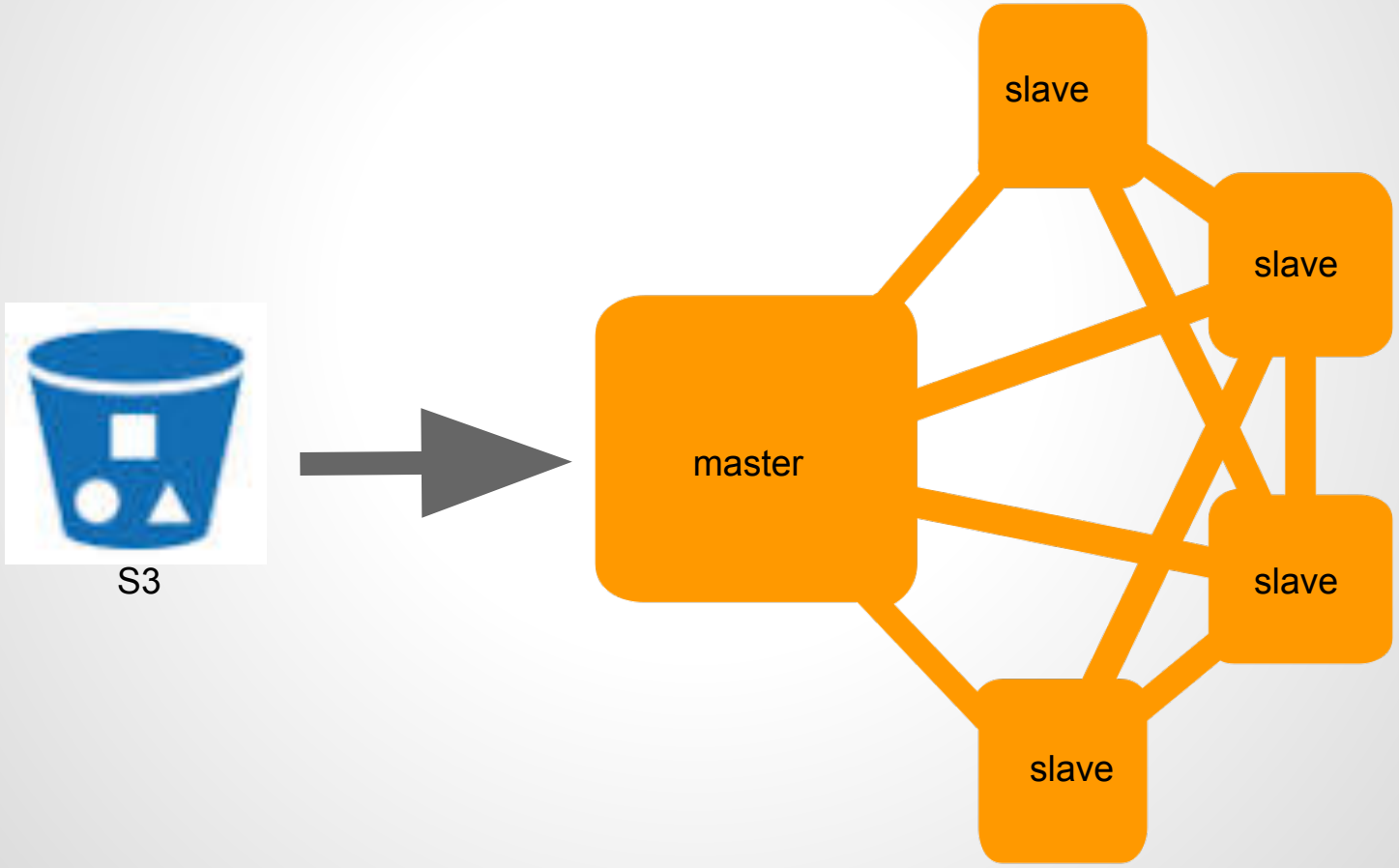
:

:

Key Features:

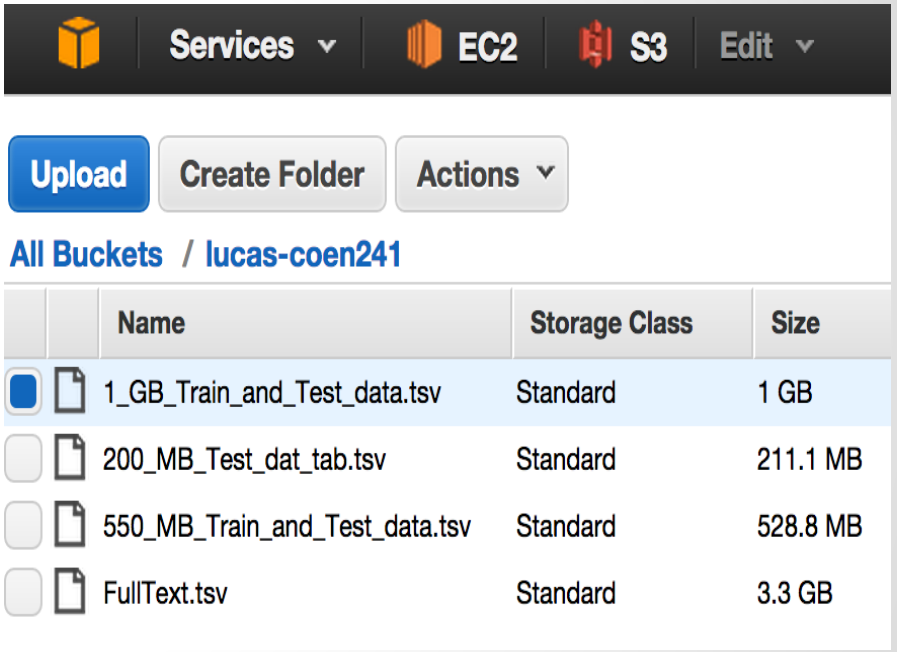
- 1,000 classes/files
- 10 million labeled images, depicting 10,000+ object categories
  - (all done by hand)

AWS - Architecture



AWS - Instance & S3

Instance Type	vCPU	Memory (GiB)	Storage (GB)	Networking Performance	Physical Processor	Clock Speed (GHz)
t2.micro	1	1	EBS Only	Low to Moderate	Intel Xeon family	2.5
t2.small	1	2	EBS Only	Low to Moderate	Intel Xeon family	2.5
t2.medium	2	4	EBS Only	Low to Moderate	Intel Xeon family	2.5
m3.medium	1	3.75	1 x 4 SSD	Moderate	Intel Xeon E5-2670 v2*	2.5
m3.large	2	7.5	1 x 32 SSD	Moderate	Intel Xeon E5-2670 v2*	2.5
m3.xlarge	4	15	2 x 40 SSD	High	Intel Xeon E5-2670 v2*	2.5
m3.2xlarge	8	30	2 x 80 SSD	High	Intel Xeon E5-2670 v2*	2.5



AWS- Cluster

- Stand alone cluster

```
./sbin/start-master.sh

./bin/spark-class org.apache.spark.deploy.worker.Worker spark://IP:PORT
```

- EC2 cluster

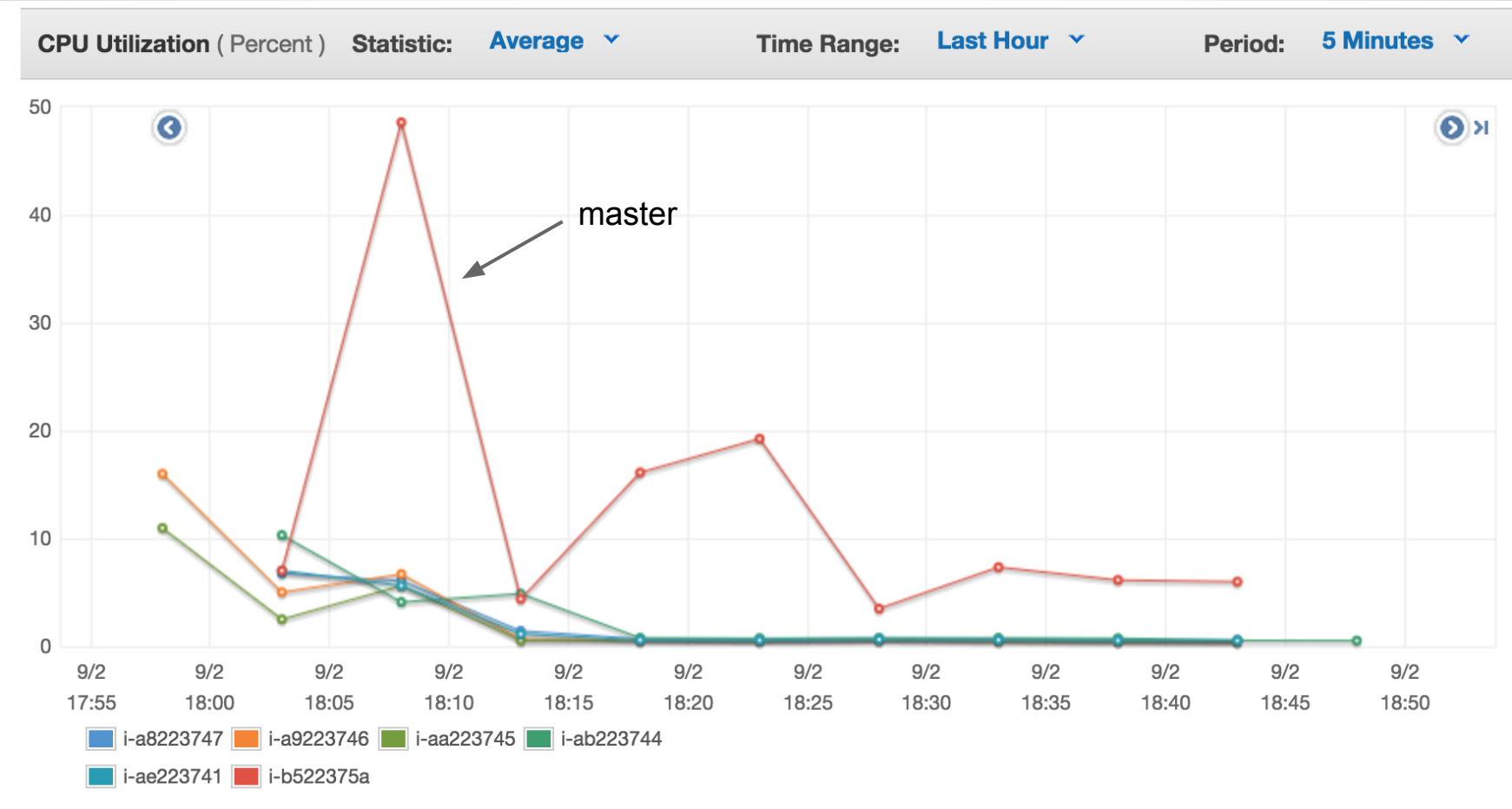
```
./spark-ec2 -k <key pair> -i <key file path> launch <cluster-name>
./spark-ec2 -k <key pair> -i <key file path> login <cluster-name>
./spark-ec2 -k <key pair> -i <key file path> start|stop <cluster-name>
```

- on YARN && Mesos
  - Probably better idea (buggy scripts..)
  - for existing distributed data on existing cluster
  - Amazon Elastic MapReduce

AWS- Cluster set-up roadmap

1. check identity (ASW key-pair)
2. set-up security group
3. Launch slaves
4. Launch master
5. set-up keyless ssh (rsa)
6. install Apache Mesos (cluster manager)
7. install spark, shark, hdfs, tachyon, ganglia on cluster
8. running setup.sh on master node

AWS- CloudWatch



CPU, Disk Read, Disk Write, Network in, Network out

AWS- Obstacles Faced

- Keyless SSH setup

  “Failed to SSH to remote host ec2-54-213-154-133.us-west-2.compute.amazonaws.com.”

  When Instance say it is running, it is not actually fully running…

- Old version instance vs new version
  - m1.large → Good^^
  - m3.large → Problems…@.@
- S3 v.s HDFS
  - S3 is freaking slow to load big file into cluster. after 40 min loading data from S3… I quit
- Iaas v.s Paas
  - update software yourself → older version of AWS CLI..==

Spark - Scala & Akka

- Functional Programming + Object-Oriented

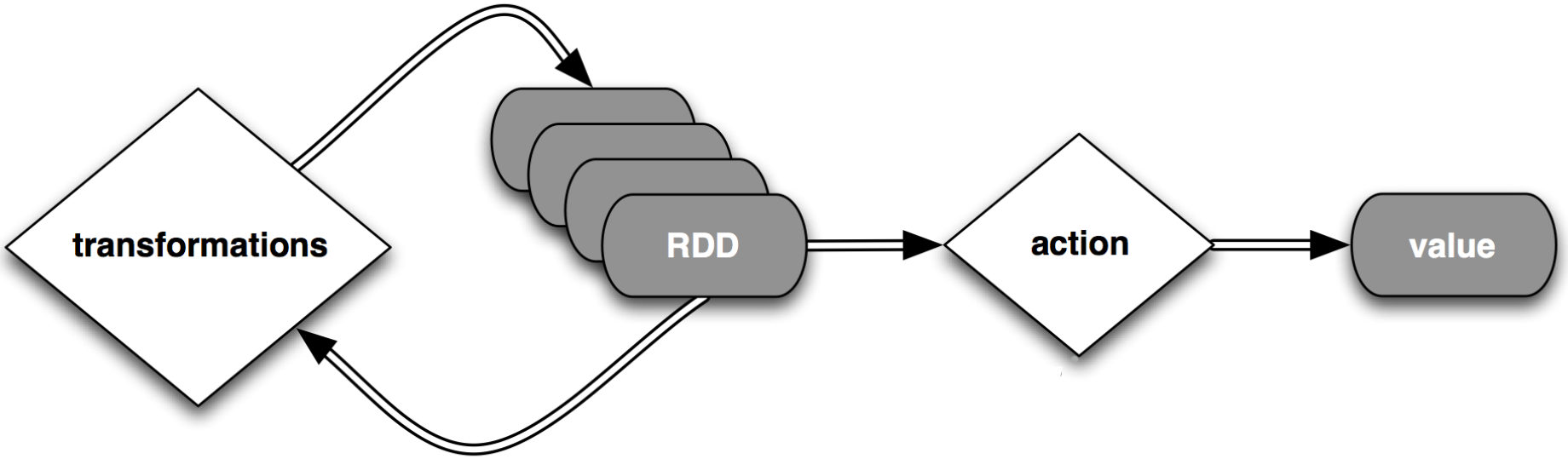
1. every value is an object and every operation is a method-call
2. first-class functions
3. a library with efficient immutable data structures

- functional programming w/out fully OO

$fn_2(*args_2, fn_1(*args_1, fn_0(*args_0)))$

- **Akka actor** as backbone of Spark

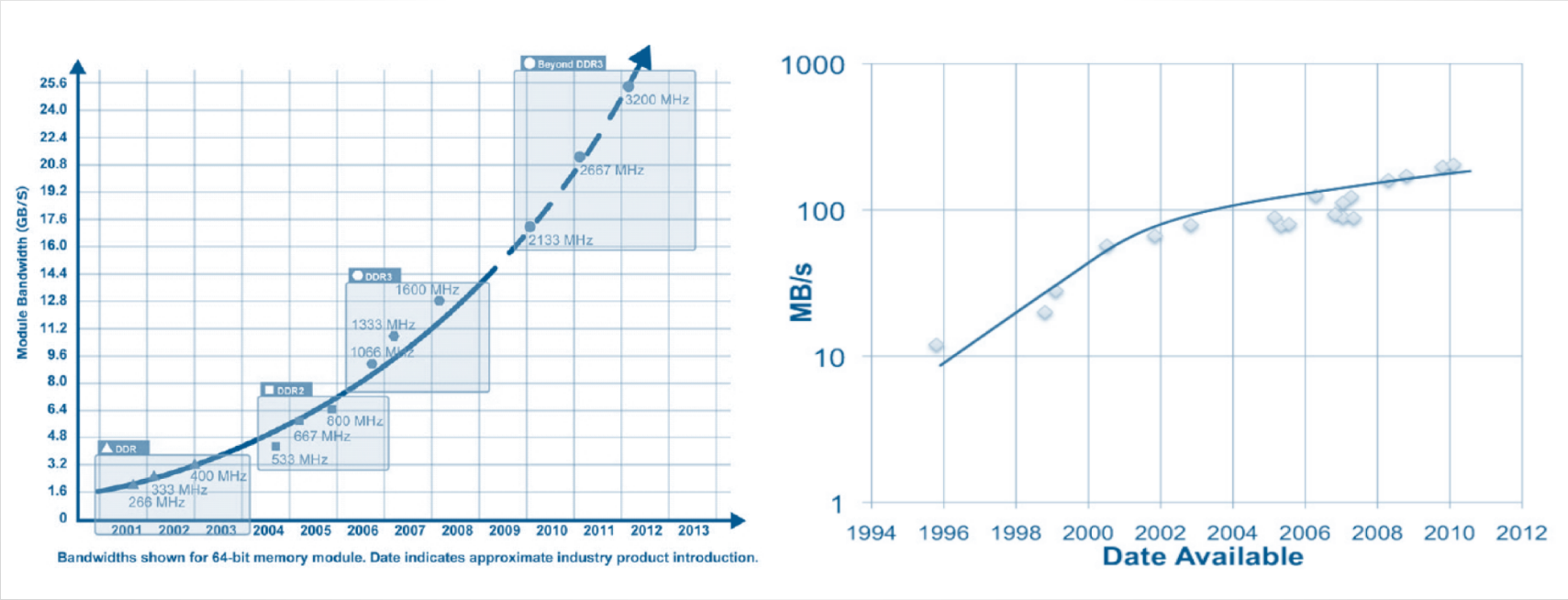
Spark- RDD



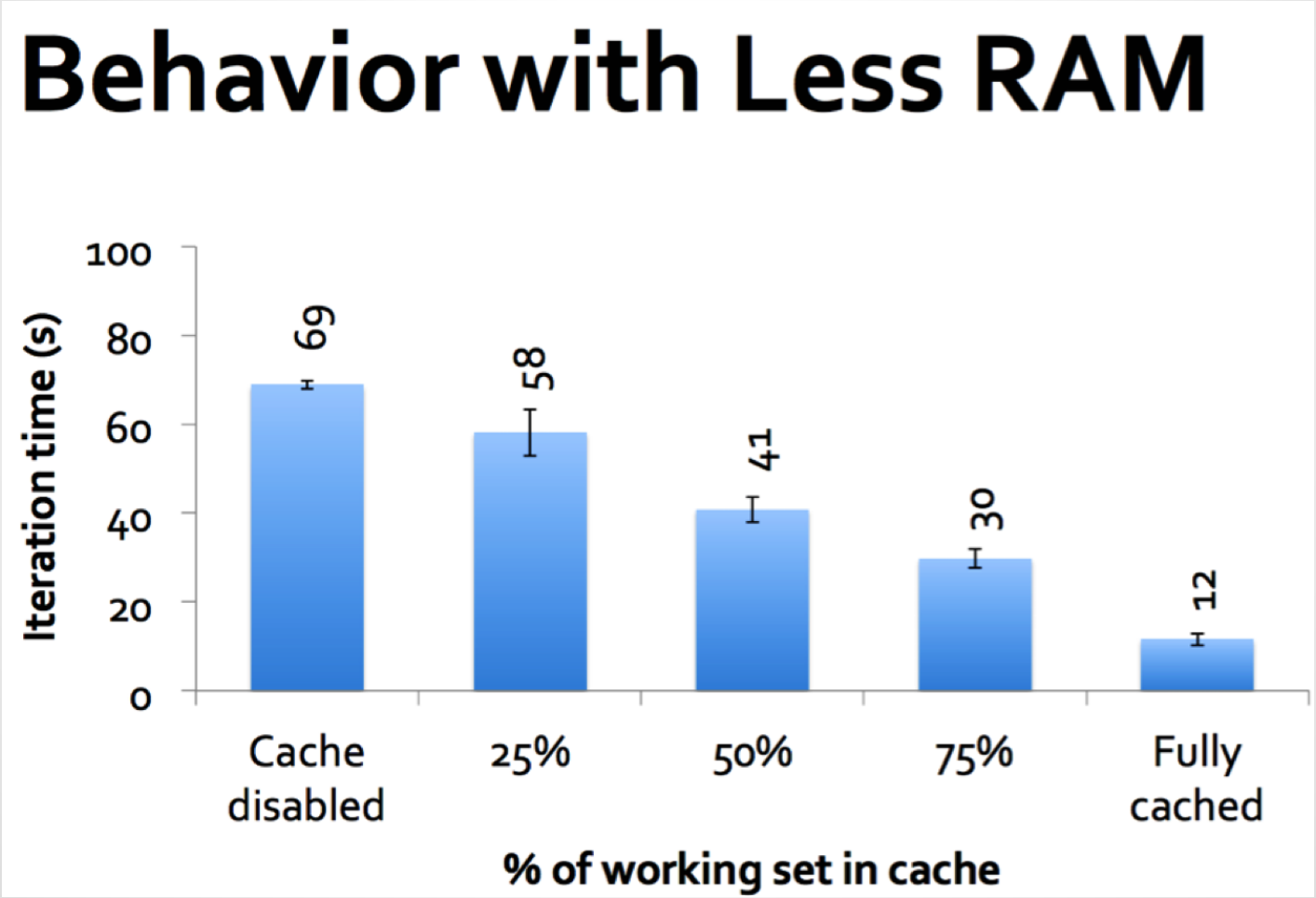
Why in-memory processing?

Support **batch**, **streaming**, and **interactive** computations…… in a unified framework.

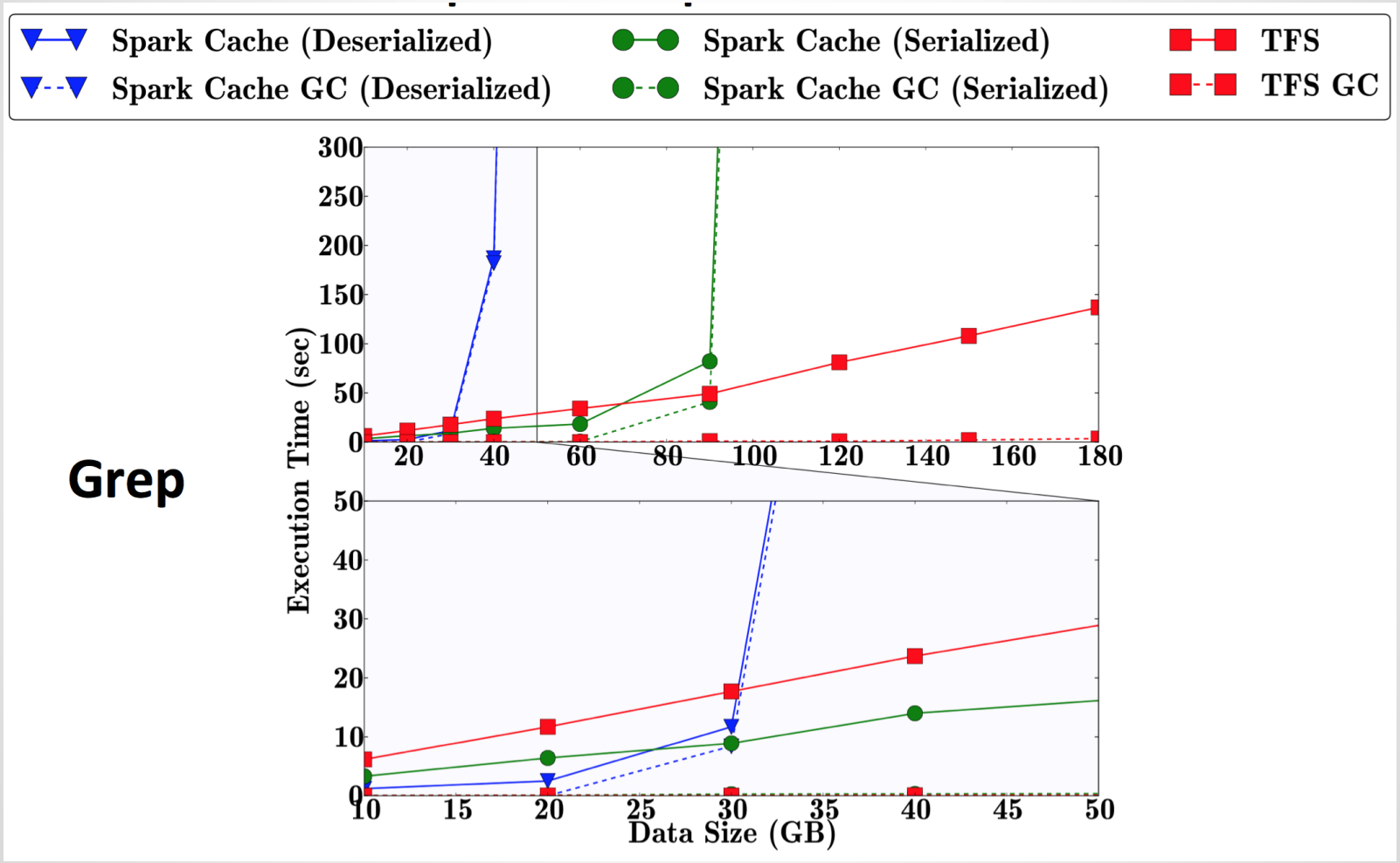
Memory Throughput matters



Spark- Cache


Behavior with Less RAM
Iteration time (s)
100
80
60
40
20
0
69
58
41
30
12
Cache disabled
25%
50%
75%
Fully cached
% of working set in cache

Spark- JVM vs Tachyon



Spark- Monitoring Cluster

Spark Master at spark://ec2-54-85-149-164.compute-1.amazonaws.com:7077

URL: spark://ec2-54-85-149-164.compute-1.amazonaws.com:7077
Workers: 5
Cores: 10 Total, 10 Used
Memory: 31.4 GB Total, 30.0 GB Used
Applications: 1 Running, 9 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Id	Address	State	Cores	Memory
worker-20140902200238-ip-172-31-15-122.ec2.internal-52597	ip-172-31-15-122.ec2.internal:52597	ALIVE	2 (2 Used)	6.3 GB (6.0 GB Used)
worker-20140902200238-ip-172-31-15-123.ec2.internal-56830	ip-172-31-15-123.ec2.internal:56830	ALIVE	2 (2 Used)	6.3 GB (6.0 GB Used)
worker-20140902200238-ip-172-31-15-124.ec2.internal-58044	ip-172-31-15-124.ec2.internal:58044	ALIVE	2 (2 Used)	6.3 GB (6.0 GB Used)
worker-20140902200238-ip-172-31-15-125.ec2.internal-50410	ip-172-31-15-125.ec2.internal:50410	ALIVE	2 (2 Used)	6.3 GB (6.0 GB Used)
worker-20140902200238-ip-172-31-15-126.ec2.internal-54495	ip-172-31-15-126.ec2.internal:54495	ALIVE	2 (2 Used)	6.3 GB (6.0 GB Used)

Running Applications

ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20140902212320-0009	NaiveBayes Application	10	6.0 GB	2014/09/02 21:23:20	root	RUNNING	3.3 min

Completed Applications

ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20140902211703-0008	NaiveBayes Application	10	6.0 GB	2014/09/02 21:17:03	root	FINISHED	3.4 min
app-20140902211008-0007	NaiveBayes Application	10	6.0 GB	2014/09/02 21:10:08	root	FINISHED	3.5 min
app-20140902210625-0006	NaiveBayes Application	10	6.0 GB	2014/09/02 21:06:25	root	FINISHED	3.4 min
app-20140902210220-0005	NaiveBayes Application	10	6.0 GB	2014/09/02 21:02:20	root	FINISHED	3.5 min
app-20140902205423-0004	NaiveBayes Application	10	6.0 GB	2014/09/02 20:54:23	root	FINISHED	3.5 min

Spark- Monitoring running App

Spark Stages

Total Duration: 6.3 min
Scheduling Mode: FIFO
Active Stages: 1
Completed Stages: 6
Failed Stages: 0

Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Shuffle Read	Shuffle Write
7	(kill) count at naive.scala:77	2014/09/02 21:24:27	5.1 min	0/2		

Completed Stages (6)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Shuffle Read	Shuffle Write
5	collect at NaiveBayes.scala:96	2014/09/02 21:24:22	1 s	2/2	1932.6 KB	
6	combineByKey at NaiveBayes.scala:91	2014/09/02 21:23:49	33 s	2/2		3.8 MB
4	collect at naive.scala:28	2014/09/02 21:23:33	16 s	2/2		
2	collect at naive.scala:21	2014/09/02 21:23:32	1 s	2/2	8.3 KB	
0	apply at Option.scala:120	2014/09/02 21:23:31	1 s	1/1	4.3 KB	
1	distinct at naive.scala:19	2014/09/02 21:23:20	10 s	2/2		8.3 KB

Failed Stages (0)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Shuffle Read	Shuffle Write	Failure Reason

Mahout vs MLlib
Future

Mahout had fairly straight forward operations

- Sequence file → Sparse Matrix → TF-IDF → train model

Spark MLlib did not provide all of the functions needed to classify

- Text classification was initially implemented in MLI, but not the stable release of MLlib did not include any of the functions needed
- TFHashing, Word2Vec, and a few other functions are being implemented in an upcoming stable release - *Spark 1.1.0*

If we had more time we could contribute our own implementation of text classification using multinomial naive bayes':

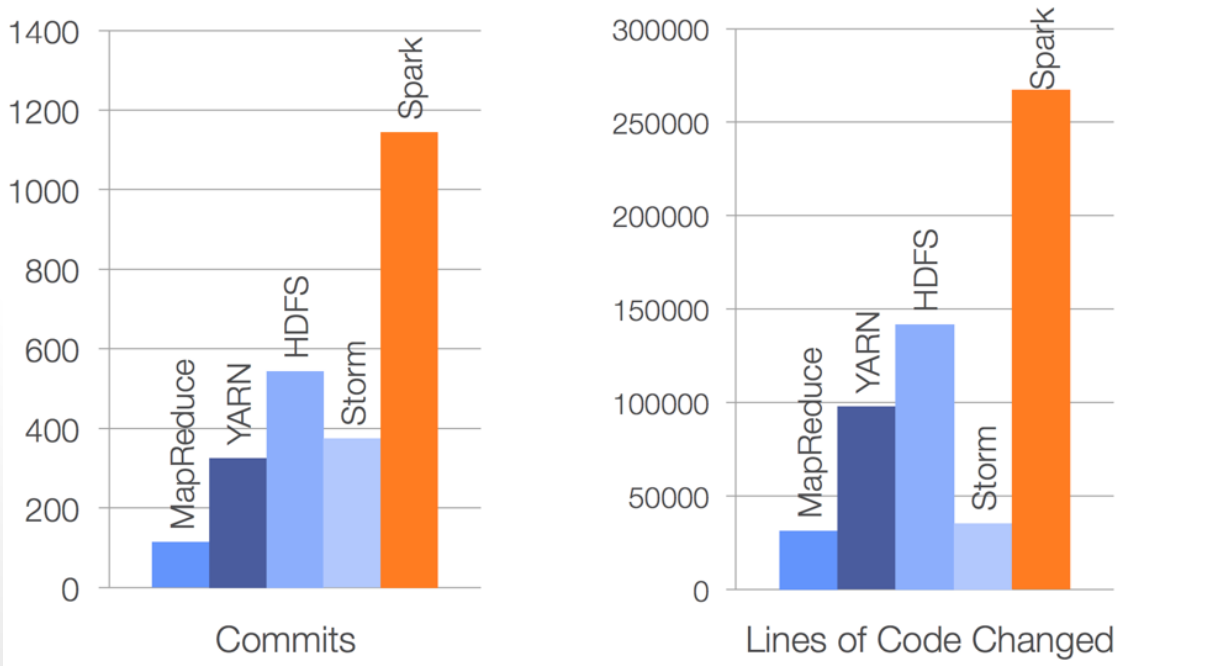
Mahout vs MLlib - Ecosystem

Mahout is moving away from its implementation of MapReduce, and moving towards DSL for linear algebraic operations

- which are automatically optimized and executed in parallel on Spark
- Unfortunately, NB is still in development for Spark

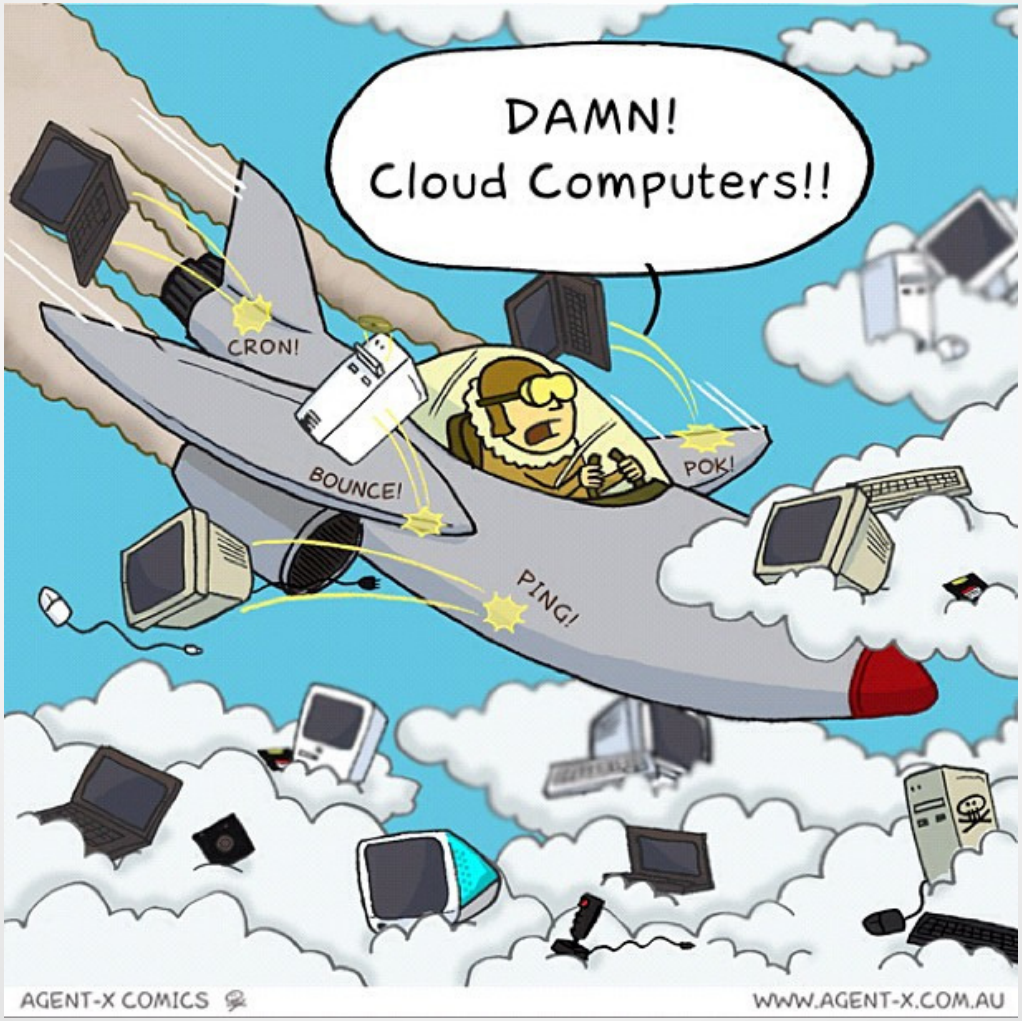
Contributions to Spark are rising dramatically:

	June 2013	July 2014
Total Contributions	68	255
Companies Contributing	17	50
Total lines of code	63,000	175,000



DEMO

Fin


DAMN!
Cloud Computers!!
CRON!
BOUNCE!
PING!
POK!
AGENT-X COMICS
WWW.AGENT-X.COM.AU

Naive Bayes' Classifier (1)

Two things of note:

1. Conditional Probability
    i. What is the probability that something will happen, given that something else has already happened?
    ii. Example:
2. Bayes' Rule
    i. Predict an outcome given *multiple evidence*
    ii. 'uncouple' multiple pieces of evidence and treat each piece of evidence as independent = *naive* Bayes
    iii. Multiple *prior* so that we give high probability to more common outcomes, and low probabilities to unlikely outcomes. These are also called *base rates* and they are a way to scale our predicted probabilities

Naive Bayes' Classifier (2) - TF-IDF

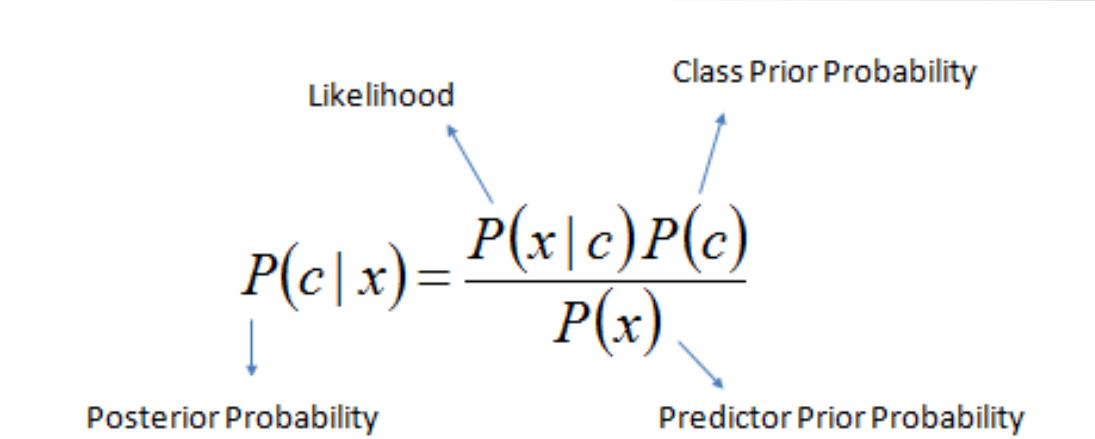
Convert word counts of a document using the TF-IDF transformation before applying the learning algorithm. The resulting formula gives a weighted importance of word

- f = word frequency
- D = number of documents
- df = number of documents containing word under consideration

$$TFIDF(word) = log(f + 1) \times log(\frac{D}{df})$$

Naive Bayes' Classifier (3)
Naive Bayes Multinomial

- Class label denoted by C; n is size of vocabulary
- Class Prior, Pr(c) = # of documents belonging to class c divided by the total # of documents
- Likelihood, Pr(x|c) = Probability of obtaining a document like x in class c
- Predictor Prior (Evidence), Pr(x) = count of a feature over the total count of features
- MNB assigns a test document x to the class that has the highest probability Pr(c | x)



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$