Using Spark for Anomaly detection

see: *Advanced Analytics with Spark*, Ryza, Laserson, Owen and Wills, 2015, O'Reilly

Spark Machine Learning

http://spark.apache.org/docs/latest/ml-guide.html

- Linear Regression
- Logistic Regression
- Linear Support Vector Machines
- Regularization
- Decision Trees
- Native Bayes
- K-Means++
- Principal Component Analysis
- Stochastic Gradient Descent

Growing fast!

KDD Cup 1999

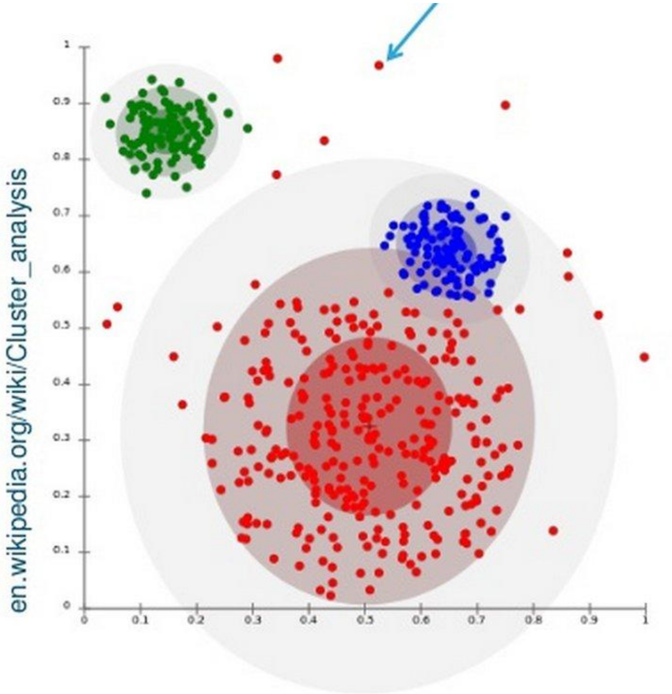
knowledge discovery and datamining contest

We will use the KDD Cup 1999 Dataset

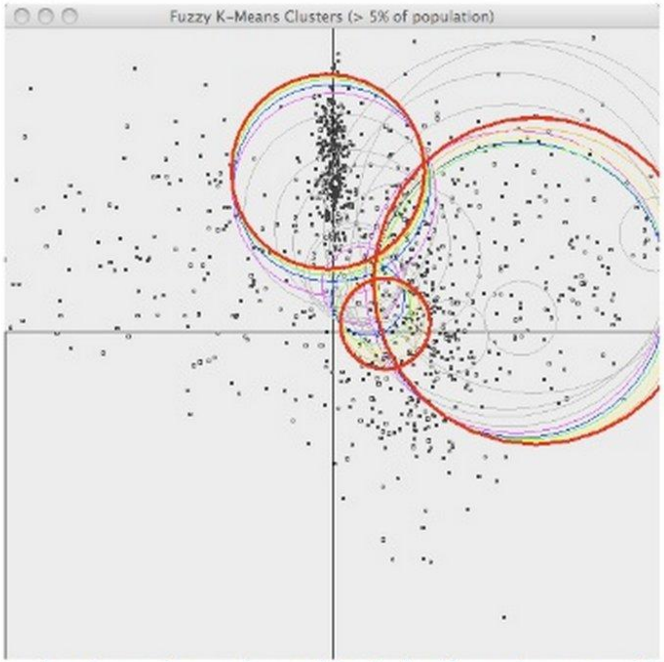
- Annual ML competition
  www.sigkdd.org/kddcup/index.php
- 1999: Network intrusion detection
- 4.9M network sessions
- Some normal; many known attacks
- Not a realistic sample!

Clustering

- Find areas dense with data (conversely, areas without data)
- Anomaly = far from any cluster
- Unsupervised learning
- Supervise with labels to improve, interpret



K-Means clustering



- Assign points to nearest center, update centers, iterate
- Goal: points close to nearest cluster center
- Must choose k = number of clusters
- ++ means smarter starting point

Location of source code for MLlib:

- http://github.com/apache/spark/blob/master/mllib/src/main/scala/org/apache/spark/mllib/clustering
- **Class: KMeans.scala**

Data

Service

Bytes Received

```
0,tcp,http,SF,215,45076,
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,
0,0,0.00,0.00,0.00,0.00,0.00,0.00,
0.00,0.00,normal.
```

% SYN errors

Label

Clustering, Take 1

Load the data

```
$ spark-shell

scala> val rawData = sc.textFile("file:///home/emma/datasets/kdd/kddcup.data")

scala> rawData.first();
```

```
2015-04-16 16:29:51,173 INFO  [main] scheduler.DAGScheduler (Logging.scala:logInfo(59)) - Job 0 finished: first at <console>:15, took 0.18
2577 s
res0: String = 0,tcp,http,SF,215,45076,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0,0,0.00,0.00,0.00,0.00,0.00
,0.00,0.00,0.00,normal.

scala> 2015-04-16 16:29:51,195 INFO  [task-result-getter-0] scheduler.TaskSetManager (Logging.scala:logInfo(59)) - Finished task 0.0 in st
age 0.0 (TID 0) in 101 ms on localhost (1/1)
2015-04-16 16:29:51,202 INFO  [task-result-getter-0] scheduler.TaskSchedulerImpl (Logging.scala:logInfo(59)) - Removed TaskSet 0.0, whose
tasks have all completed, from pool
```

Refine the data (1)

not interested in elements at position 1, 2 and 3.

```
0,tcp,http,SF,215,45076,
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,
0,0,0.00,0.00,0.00,0.00,0.00,0.00,
0.00,0.00,normal
```

use the last element as the key

Refine the data (2)

```
scala> import org.apache.spark.mllib.linalg.{Vector, Vectors}
import org.apache.spark.mllib.linalg.{Vector, Vectors}

scala> val labelsAndData = rawData.map { line=>
     | val buffer = line.split(',').toBuffer
     | buffer.remove(1,3)
     | val label = buffer.remove(buffer.length-1)
     | val vec = Vectors.dense(buffer.map(_.toDouble).toArray)
     | (label, vec)
     | }
labelsAndData: org.apache.spark.rdd.RDD[(String, org.apache.spark.mllib.linalg.Vector)] = MappedRDD[2]

scala>
```

Cache the data and run the clustering algorithm

```
scala> val data = labelsAndData.values.cache()
data: org.apache.spark.rdd.RDD[org.apache.spark.mllib.linalg.Vector] = MappedRDD[3] at values at <console>:17

scala> import org.apache.spark.mllib.clustering._
import org.apache.spark.mllib.clustering._

scala> val kmeans = new KMeans()
kmeans: org.apache.spark.mllib.clustering.KMeans = org.apache.spark.mllib.clustering.KMeans@5fb79c5c

scala> val model = kmeans.run(data)
2015-04-16 17:00:16,281 INFO  [main] spark.SparkContext (Logging.scala:logInfo(59)) - Starting job: takeSample at KMeans.scala:277
2015-04-16 17:00:16,282 INFO  [sparkDriver-akka.actor.default-dispatcher-2] scheduler.DAGScheduler (Logging.scala:logInfo(59)) - Got job 1
 (takeSample at KMeans.scala:277) with 23 output partitions (allowLocal=false)
2015-04-16 17:00:16,282 INFO  [sparkDriver-akka.actor.default-dispatcher-2] scheduler.DAGScheduler (Logging.scala:logInfo(59)) - Final sta
ge: Stage 1(takeSample at KMeans.scala:277)
2015-04-16 17:00:16,282 INFO  [sparkDriver-akka.actor.default-dispatcher-2] scheduler.DAGScheduler (Logging.scala:logInfo(59)) - Parents o
f final stage: List()
2015-04-16 17:00:16,286 INFO  [sparkDriver-akka.actor.default-dispatcher-2] scheduler.DAGScheduler (Logging.scala:logInfo(59)) - Missing p
arents: List()
2015-04-16 17:00:16,286 INFO  [sparkDriver-akka.actor.default-dispatcher-2] scheduler.DAGScheduler (Logging.scala:logInfo(59)) - Submittin
```

and wait…

Spark jobs - localhost:4040/jobs/

Spark | Jobs | Stages | Storage | Environment | Executors | **Spark shell** application UI

Spark Jobs (?)

Total Duration: 1.5 h
Scheduling Mode: FIFO
Active Jobs: 1
Completed Jobs: 6
Failed Jobs: 0

Active Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
6	collect at KMeans.scala:297	2015/04/16 17:30:57	1.6 min	0/1	9/23

Completed Jobs (6)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
5	collectAsMap at KMeans.scala:289	2015/04/16 17:25:02	5.9 min	2/2	46/46
4	collect at KMeans.scala:297	2015/04/16 17:19:19	5.7 min	1/1	23/23
3	collectAsMap at KMeans.scala:289	2015/04/16 17:13:11	6.1 min	2/2	46/46
2	takeSample at KMeans.scala:277	2015/04/16 17:06:26	6.8 min	1/1	23/23
1	takeSample at KMeans.scala:277	2015/04/16 17:00:16	6.2 min	1/1	23/23
0	first at <console>:15	2015/04/16 16:29:51	0.2 s	1/1	1/1

Failed Jobs (0)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total

Spark 1.2.0

Spark stages - localhost:4040/stages/

Spark | Jobs | Stages | Storage | Environment | Executors | **Spark shell** application UI

Spark Stages (for all jobs)

Total Duration: 1.5 h
Scheduling Mode: FIFO
Active Stages: 1
Completed Stages: 8
Failed Stages: 0

Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
8	collect at KMeans.scala:297 +details (kill)	2015/04/16 17:30:58	3.0 min	14/23	298.9 MB			

Completed Stages (8)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
7	collectAsMap at KMeans.scala:289 +details	2015/04/16 17:30:57	0.4 s	23/23				
6	flatMap at KMeans.scala:285 +details	2015/04/16 17:25:02	5.9 min	23/23	636.7 MB			4.5 KB
5	collect at KMeans.scala:297 +details	2015/04/16 17:19:19	5.7 min	23/23	636.7 MB			
4	collectAsMap at KMeans.scala:289 +details	2015/04/16 17:19:18	0.4 s	23/23				
3	flatMap at KMeans.scala:285 +details	2015/04/16 17:13:12	6.1 min	23/23	755.5 MB			4.5 KB
2	takeSample at KMeans.scala:277 +details	2015/04/16 17:06:26	6.8 min	23/23	1096.3 MB			

http://localhost:4040/stages/

Output

results are in model (KMeanModel):

- **clusterCenters:** a list of cluster centers
- **computeCost**: the model's residual error - (sum of squared distances of data points to their nearest center)
- **k**: Total number of clusters.
- **predict(points)**: clusters to which the points belong

```
>> model.computeCosts(data)
   4.6634585670252554E18

   Huh?
```

Terrible

```
0,tcp,http,SF,215,45076,
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,
0,0,0.00,0.00,0.00,0.00,0.00,0.00,
0.00,0.00,normal.
```

0	back.	2203	0	perl.	3
0	buffer_overflow.	30	0	phf.	4
0	ftp_write.	8	0	pod.	264
0	guess_passwd.	53	0	portsweep.	10412
0	imap.	12	0	rootkit.	10
0	ipsweep.	12481	0	satan.	15892
0	land.	21	0	smurf.	2807886
0	loadmodule.	9	0	spy.	2
0	multihop.	7	0	teardrop.	979
0	neptune.	1072017	0	warezclient.	1020
0	nmap.	2316	0	warezmaster.	20
0	normal.	972781	1	portsweep.	1

Clustering, Take 2

- Choose better k values
- Score each cluster using "density"

```scala
def distToCentroid(v: Vector, model: KMeansModel) = {
  val centroid = model.clusterCenters(model.predict(v))
  distance(centroid, datum)
}

def clusteringScore(data: RDD[Vector], k: Int) = {
  val kmeans = new KMeans()
  kmeans.setK(k)
  val model = kmeans.run(data)
  data.map(d => distToCentroid(d, model)).mean()
}

(5 to 40 by 5).map(k => (k, clusteringScore(data, k)))
```

try several values for k

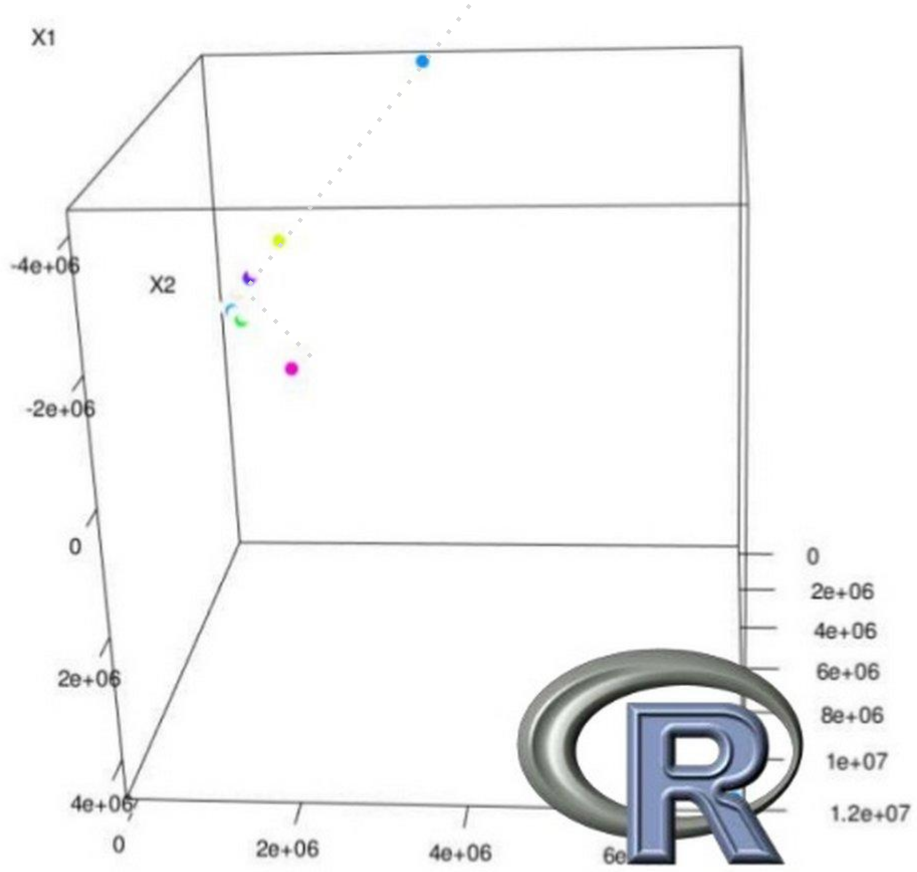
run k-means and score the clusters using their “density”

```
( 5,1938.858341805931)
(10,1689.4950178959496)
(15,1381.315620528147)
(20,1318.256644582388)
(25, 932.0599419255919)
(30, 594.2334547238697)
(35, 829.5361226176625)
(40, 424.83023056838846)
```

```
kmeans.setRuns(10)
kmeans.setEpsilon(1.0e-6)

(30 to 100 by 10).par.map(k =>
  (k, clusteringScore(data, k)))
```

```
( 30,862.9165758614838)
( 40,801.679800071455)
( 50,379.7481910409938)
( 60,358.6387344388997)
( 70,265.1383809649689)
( 80,232.78912076732163)
( 90,230.0085251067184)
(100,142.84374573413373)
```



X1
X2
-4e+06
-2e+06
0
2e+06
4e+06
0
2e+06
4e+06
0
2e+06
4e+06
6e+06
8e+06
1e+07
1.2e+07

Clustering, Take 3
normalize the data

```
0,tcp,http,SF,215,45076,
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,
0,0,0.00,0.00,0.00,0.00,0.00,0.00,
0.00,0.00,normal.
```

Normalizing the data

$$\frac{x_i - \mu_i}{\sigma_i}$$

- Standard or "z" score
- σ (standard deviation): normalize away scale
- μ (mean): *doesn't really matter here*
- Assumes normal*ish* distribution

```
val dataArray = data.map(_.toArray)
val numCols = dataArray.first().length
val n = dataArray.count()

val sums = dataArray.reduce((a,b) => a.zip(b).map(t => t._1 + t._2))
val sumSquares = dataAsArray.fold(new Array[Double](numCols))(
 (a,b) => a.zip(b).map(t => t._1 + t._2 * t._2)
)
val stdevs = sumSquares.zip(sums).map {
  case(sumSq,sum) => math.sqrt(n*sumSq - sum*sum)/n
}
val means = sums.map(_ / n)

def normalize(v: Vector) = {
  val normed = (v.toArray, means, stdevs).zipped.map(
    (value, mean, stdev) => (value - mean) / stdev)
  Vectors.dense(normed)
}
```

```
( 60,0.0038662664156513646)
( 70,0.003284024281015404)
( 80,0.0030876845856813 1)
( 90,0.0028326001931487516)
(100,0.002550914511356702)
(110,0.002516106387216959)
(120,0.0021317966227260106)
```

Clusters scored (take 2)

Use entropy and the label (e.g. “normal”)

Calculating the entropy of a cluster

$$-\sum p_i \log p_i$$

$$\sum p_i \log (1/p_i)$$

- Information theory concept
- Measures mixed-ness
- Function of label proportions, p_i
- Good clusters have homogeneous labels
- Homogeneous = low entropy = good clustering

```
def entropy(counts: Iterable[Int]) = {
  val values = counts.filter(_ > 0)
  val n: Double = values.sum
  values.map { v =>
    val p = v / n
    -p * math.log(p)
  }.sum
}

def clusteringScore(...) = {
  ...
  val labelsAndClusters =
    normalizedLabelsAndData.mapValues(model.predict)
  val clustersAndLabels = labelsAndClusters.map(_.swap)
  val labelsInCluster =
    clustersAndLabels.groupByKey().values
  val labelCounts = labelsInCluster.map(
    _.groupBy(l => l).map(_._2.size))
  val n = normalizedLabelsAndData.count()
  labelCounts.map(m => m.sum * entropy(m)).sum / n
}
```

```
(80,1.0079370754411006)
(90,0.9637681417493124)
(100,0.9403615199645968)
(110,0.4731764778562114)
(120,0.3705663690683805)
(130,0.3658424954256571 7)
(140,0.1053252946374940 2)
(150,0.1038031976230395 9)
(160,0.1446912989257944 4)
```

0	back.	6
0	neptune.	821239
0	normal.	255
0	portsweep.	114
0	satan.	31
...		
90	ftp_write.	1
90	loadmodule.	1
90	neptune.	1
90	normal.	41253
90	warezclient.	12
...		
93	normal.	8
93	portsweep.	7365
93	warezclient.	1

Notebooks from Databricks

https://databricks.com/resources/type/example-notebooks

Mixing SQL and Machine Learning with ease

```
val trainingDataTable = sql(""" SELECT
e.action, u.age, u.latitude, u.logitude FROM Users u
JOIN Events e ON u.userId = e.userId""")// Since `sql`
returns an RDD, the results of can be easily used in MLlib

val trainingData = trainingDataTable.map { row =>
  val features = Array[Double](row(1), row(2), row(3))
  LabeledPoint(row(0), features)
}

val model = new LogisticRegression().run(trainingData)
```

```
model.predict(query.age, query.latitude, query.longitude)
```

References

Download the data from:

kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Select kddcup.data.gz

Next KDD conference: http://www.kdd.org/kdd2017/

- **in August, in Nova Scotia**

Also see:

- http://spark.apache.org/docs/latest/programming-guide.html
- https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.package
- https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.rdd.RDD

Spark Streaming

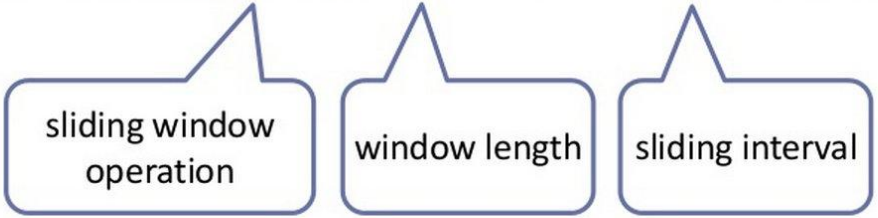
A Story for another time

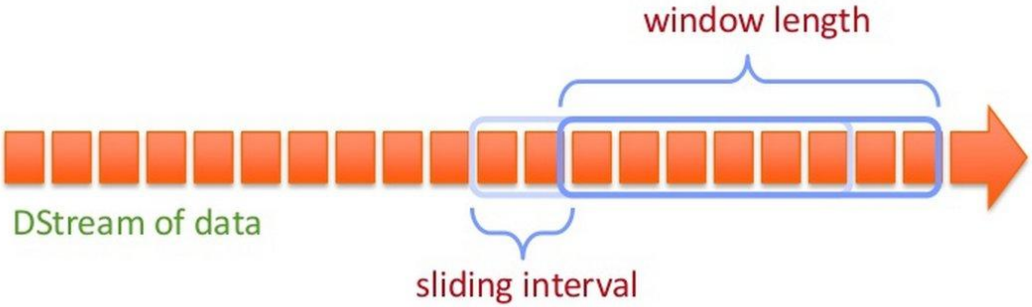
Spark Streaming

- Chop up the live stream into batches of ½ second or more
- Each micro-batch defines an RDD
- Use the Spark APIs to process streams
- Combine batch and stream processing into one system

Window-based transformations

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap(status => getTags(status))
val tagCounts = hashTags.window(Minutes(1), Seconds(5)).countByValue()
```





Spark streaming

for each microbatch (1 second)

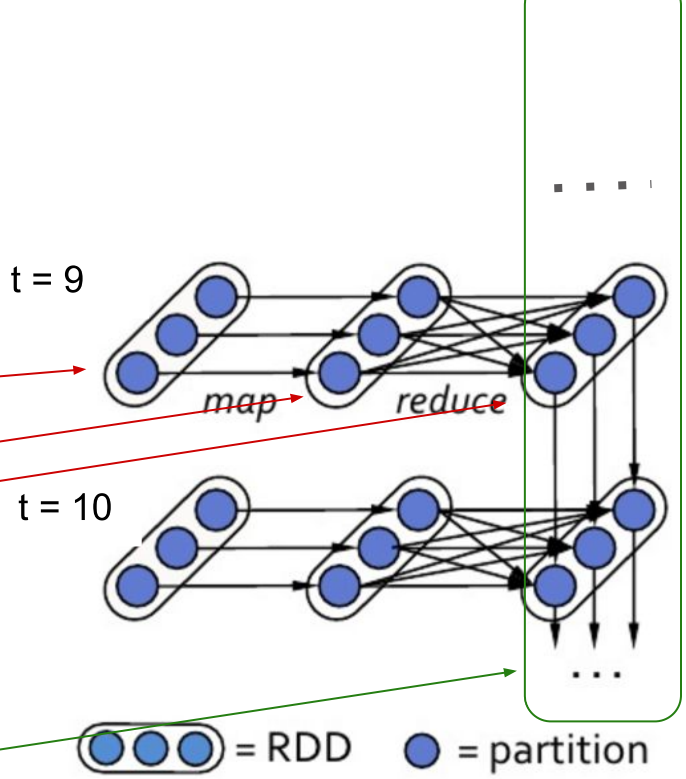
load page view info from stream

map: extract urls from page views

reduce: countByValue

for each slide (2 seconds)

count urls for the past 10 seconds



Spark Streaming: counting URLs in a stream

```
val host = args(0)
val port = args(1).toInt

// Create the job context
val ssc = new StreamingContext(masterName, appName, Seconds(1),
   System.getenv("SPARK_HOME"), StreamingContext.jarOfClass(this.getClass).toSeq)

// Create a stream for host:port and convert each line to a PageView
val pageViews = ssc.socketTextStream(host, port).flatMap(_.split("\n")).map(PageView.fromString(_))

// Map and return a count of views per URL seen in each batch
val counts = pageViews.map(view => view.url).countByValue()

// Slide over 2 seconds, create a 10 second window, and count views per URL in window
val slidingPageCounts = pageViews.map(view => view.url).countByValueAndWindow(Seconds(10),
Seconds(2))

pageCounts.print()
slidingPageCounts.print()

ssc.start()
```

Streaming + Batch + Ad-Hoc

Combining streams with historical data:

```
pageViews.join(historicCounts).map(...)
```

Queries on stream slices

```
counts.slice("21:00", "21:05").topK(10)
```