电子科技大学

University of Electronic Science and Technology of China

Distributed Processing

Feng Huang

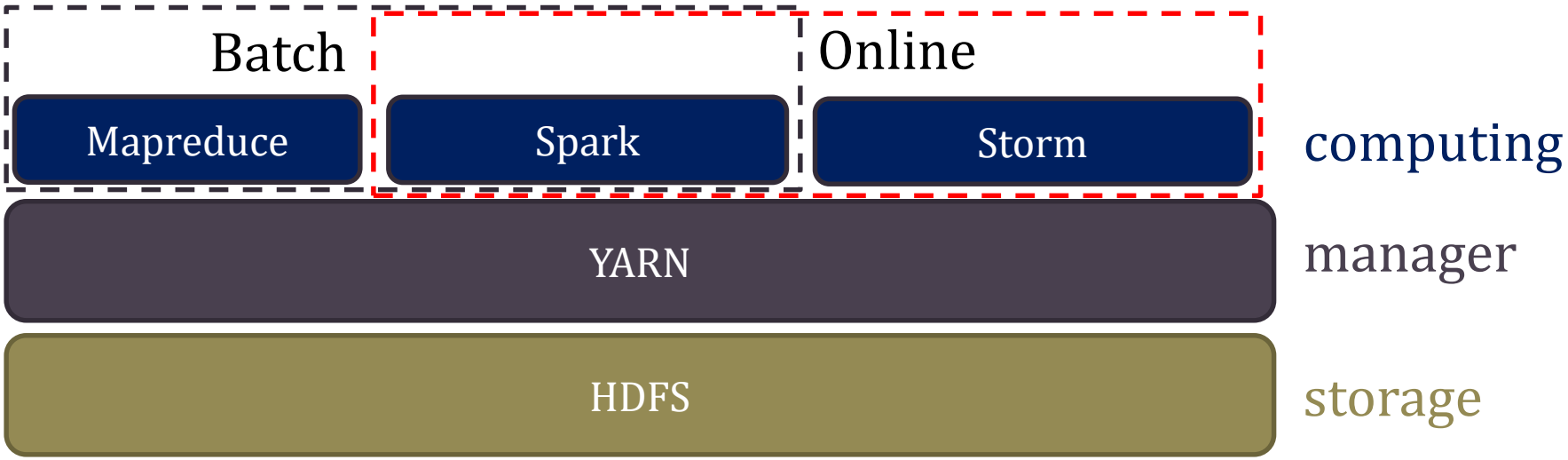
Data Mining Lab, Big Data Research Center, UESTC
Email：junmshao@uestc.edu.cn
http://staff.uestc.edu.cn/shaojunming

Outline

- Programming models of Distributed Processing
- SCOPE
- Programming models of Distributed Stream Processing
- Spark Streaming mllib

Big Data

The Hadoop Ecosystem is like software collections (e.g. Filesystem, OS, DBMS, Computing framework)of single machine, building upon **distributed environment** is the only difference.



Word Count

Question1:
How do you(**only one person**) count the number of occurrences of each word in a given input set ?

Question2:
If you are **a master and have many slaves**, How do you solve the problem?

Deer Bear River
Car Car River
Deer Car Bear

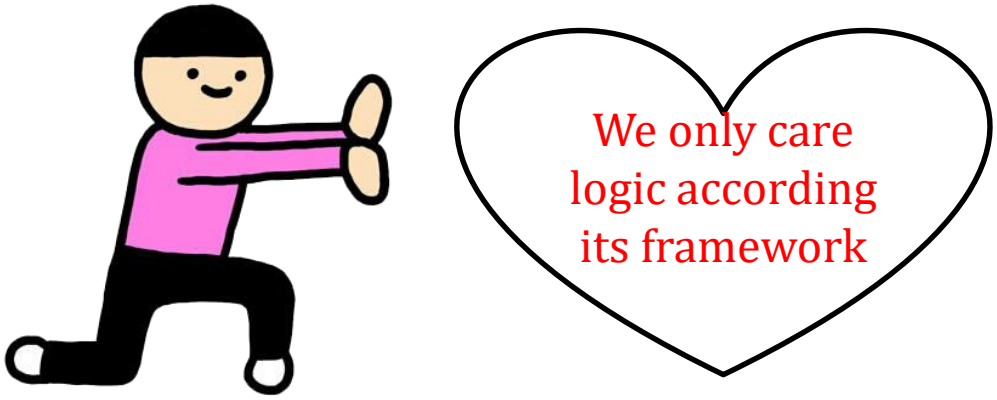
Word Count

In Question2, you can be regarded as a master node which **distributes tasks to slave nodes**. And communication among you and your slaves are like information transferring.

So we actually can write Programs to solve the problem in distributed environment?

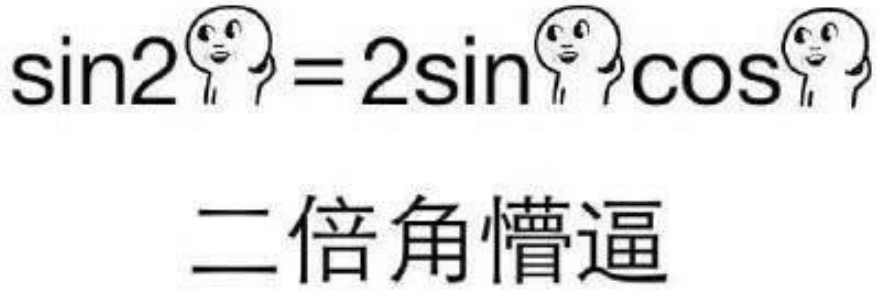
Word Count

Programming models of Distributed Processing help us scale up from single servers to thousands of machines. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.



Word Count

The key point is to move computing according data (**make computing especially "near" data**) instead of moving data by network.



Map Reduce

How to write a mapreduce program?

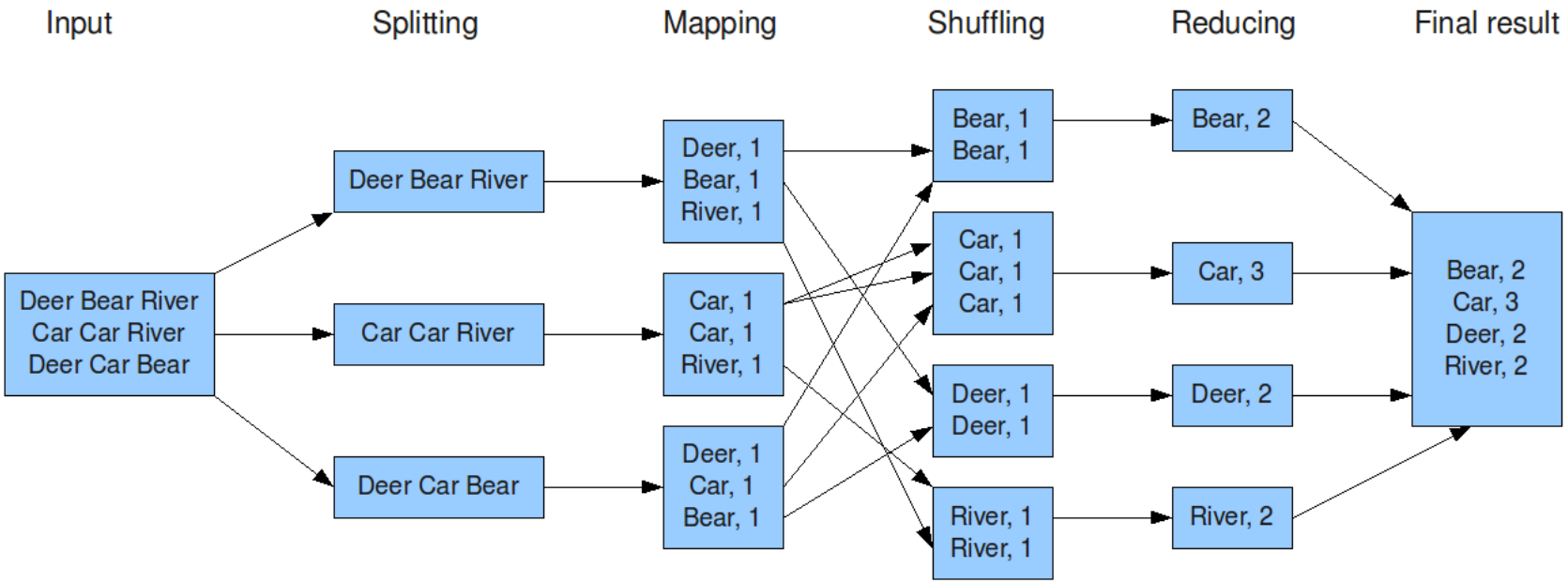
Map:
Design how to transform input to (key, value).

Reduce:
Think about how to operate a value collection which are same key(key, collection<value>).

Main:
configure all.

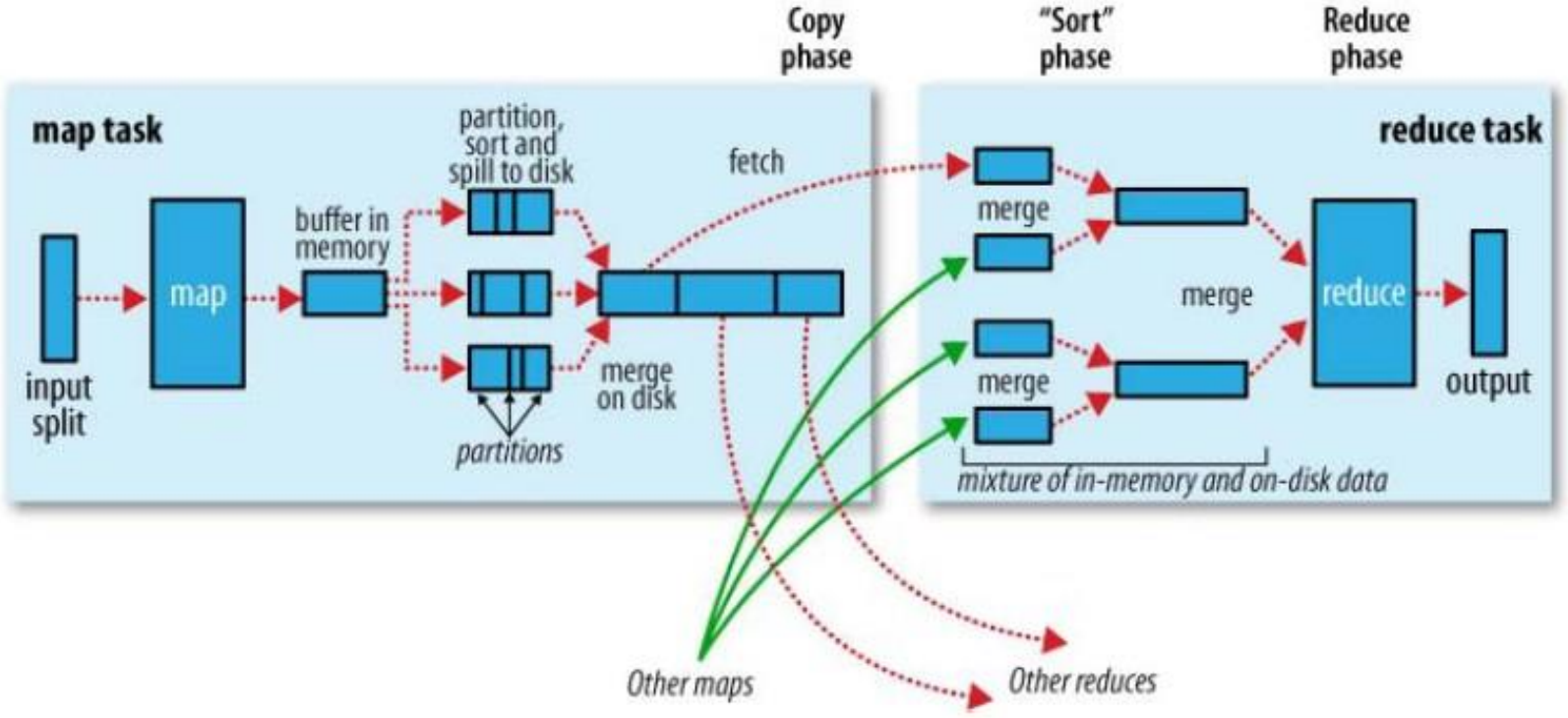
Word count of Map reduce

The whole process contains **splitting**(how to split data), **map**(how to transform raw data)，**shuffle**(sort over nodes), **reduce**(aggregate and handle by key).



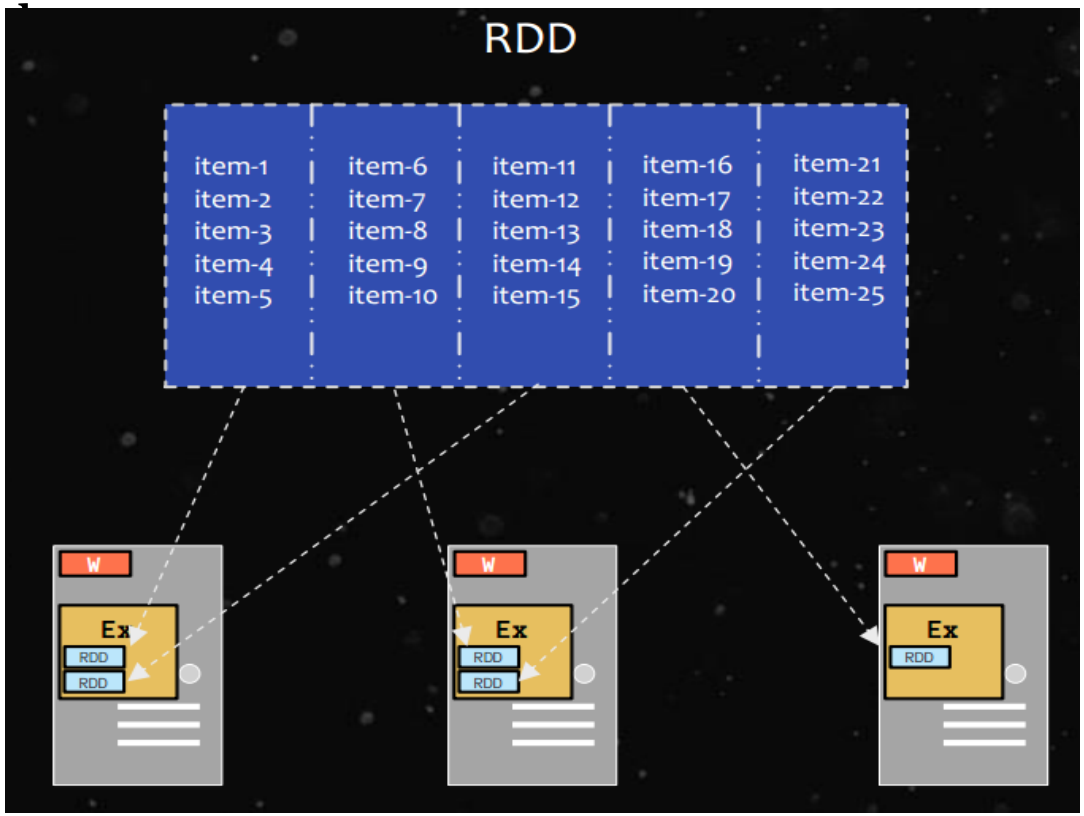
Procedure of MapReduce

Input to the Reducer is the **sorted output of the mappers.** In this phase the framework fetches the relevant partition of the output of all the mappers, via HTTP



Spark

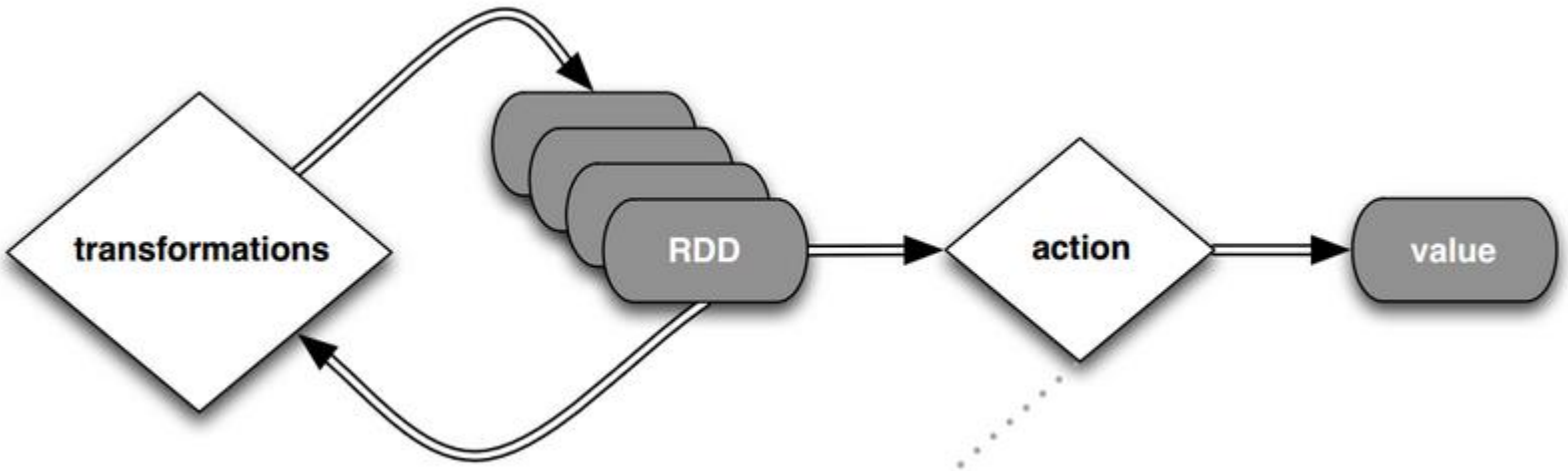
The main abstraction in Spark is that of **a resilient distributed dataset** (RDD), which represents a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition i



1. Collections of objects spread across a cluster, stored in RAM or on Disk
2. Built through parallel transformations
3. Automatically rebuilt on failure

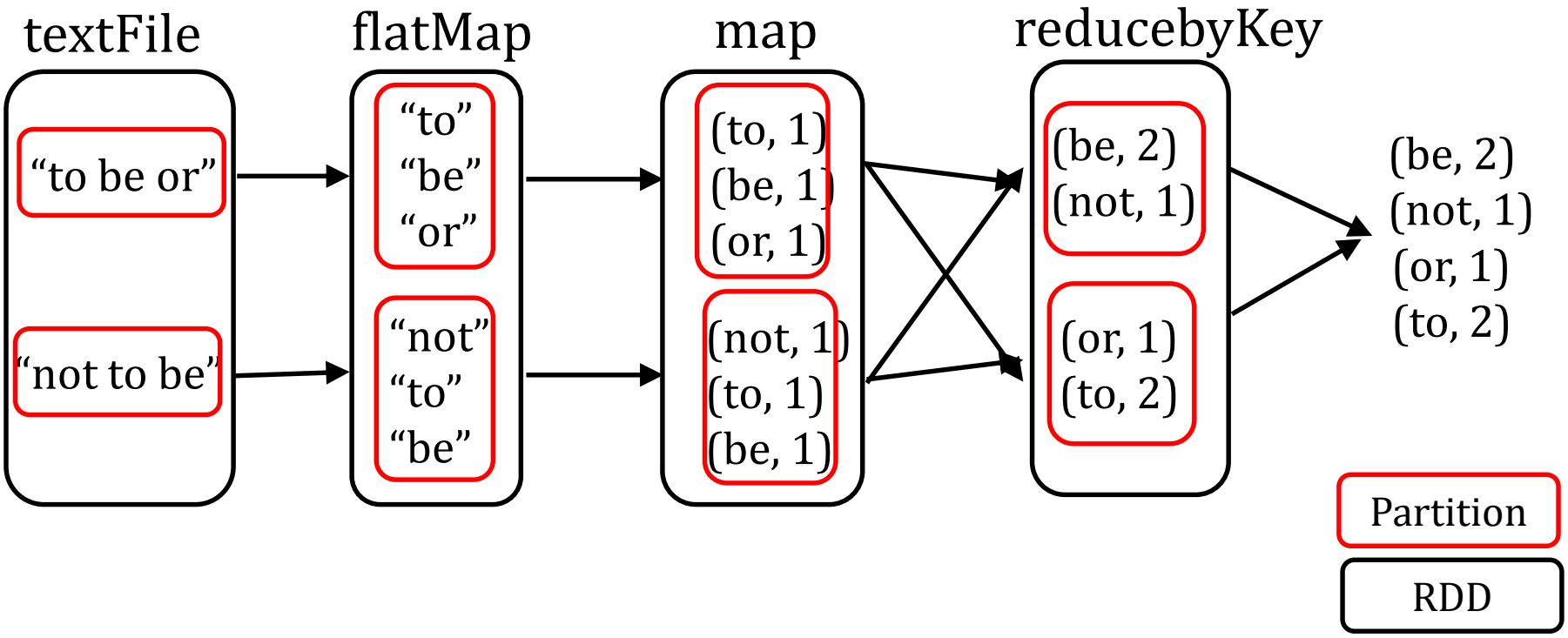
Operations on RDD

1. **Transformations**
(e.g. map, filter, groupBy)
2. **Actions**
(e.g. count, collect, save)

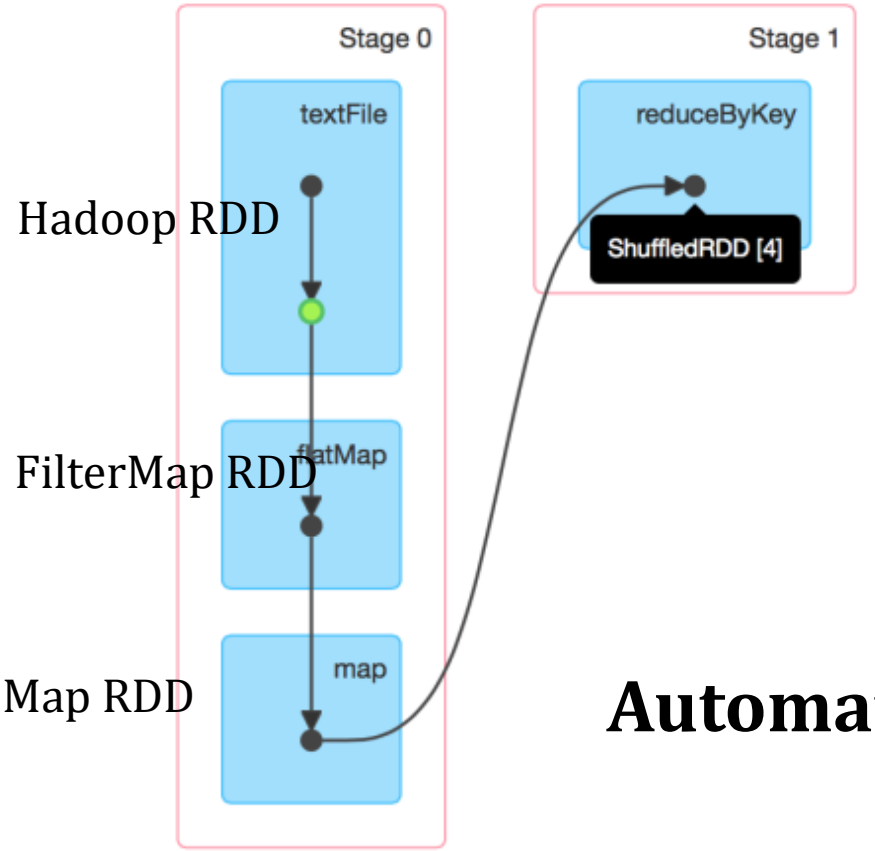


Word Count of Spark


```
val lines = sc.textFile("hamlet.txt")
val counts = lines.flatMap( line=> line.split(" "))
             .map(word => (word, 1))
             .reduceByKey((x,y)=> x + y).collect()
```



DAG of Word Count



Stage Graph

Needs to compute my parents, parents, parents, etc all the way back to an RDD with no dependencies .

Automatically rebuilt on failure

MapReduce VS Spark

MapReduce

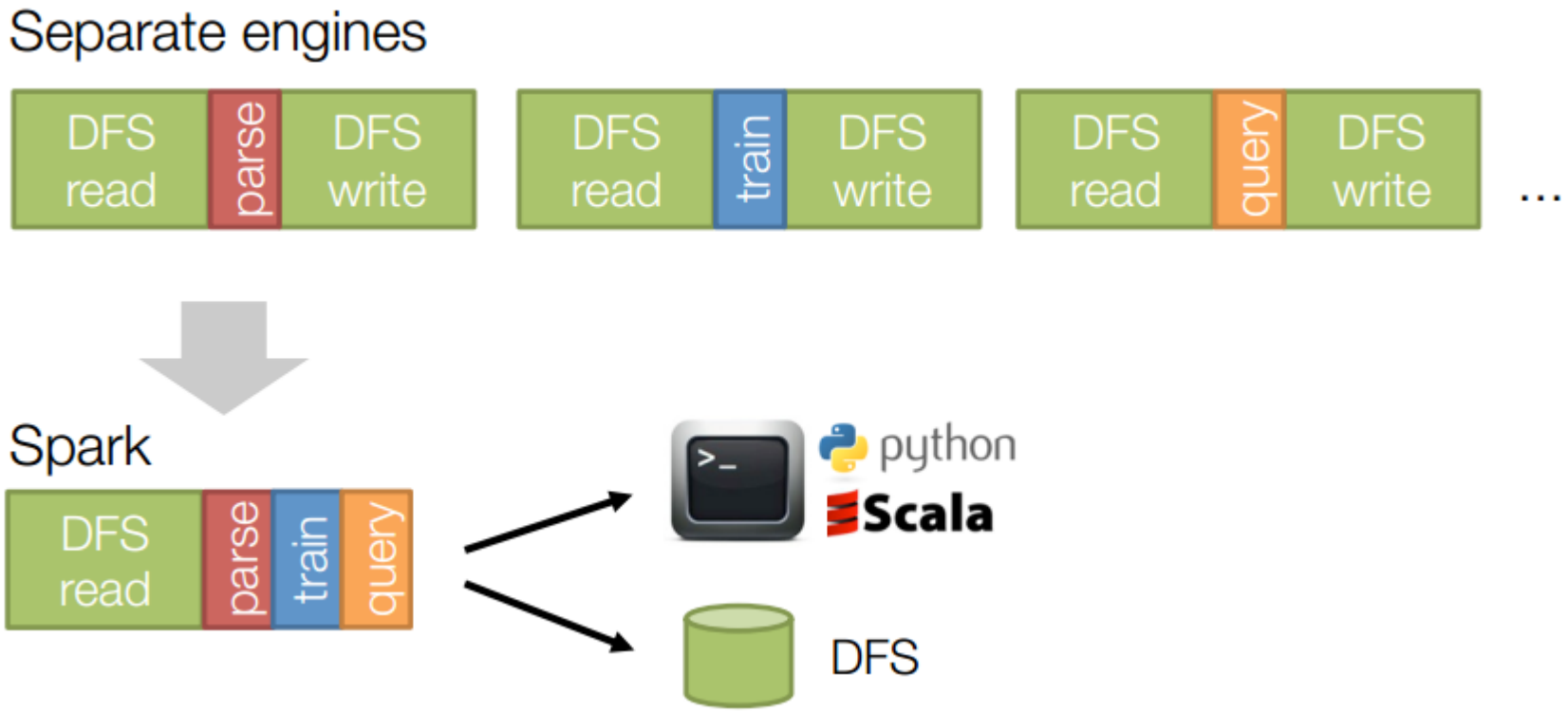
- Great **at one-pass** computation, but inefficient for *multi-pass* algorithms.
- No efficient primitives for data sharing.

Spark

- Extends a programming language with **a distributed collection data-structure（RDD）.**
- Clean APIs in Java, Scala, Python, R.

MapReduce Vs Spark

Same engine performs data extraction, model training and interactive queries



Kmeans of MapReduce

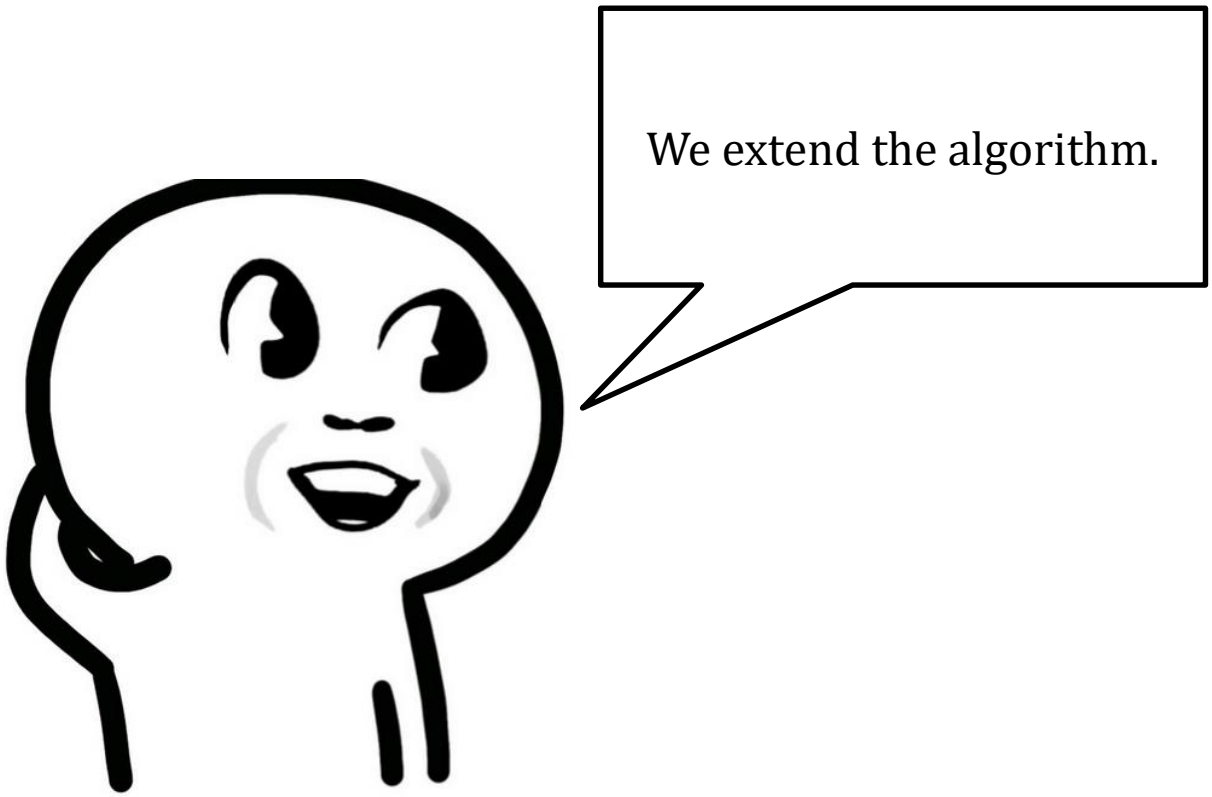
How to design a kmeans of mapreduce? It means what you do in **the procedure of map and reduce** to realize kmeans.

Map:
Assign each data to the cluster, we can get (clusterID, Data)

Reduce:
Update the cluster means (clusterID, collection(Data))

Other Algorithms

Some algorithm are not parallel such like SGD, how do we do?



SCOPE

- Large scale regularized logistic regression

$$\min_w \frac{1}{n}\sum_{i=1}^{n}\left[\underbrace{\ln(1+e^{-w^\top x_i y_i})+\frac{\lambda}{2}\|w\|_2^2}_{f_i(w)}\right]$$

  - data (x_i, y_i) with $y_i \in \{\pm 1\}$
  - parameter vector w.

- big data: $n \sim 10 - 100$ billion
- high dimension: $\dim(x_i) \sim 10 - 100$ billion

Solve

$$w_* = \arg\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w).$$

Gradient Descent (GD):

$$w_k = w_{k-1} - \eta_k \nabla f(w_{k-1}).$$

If

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w),$$

GD requires the computation of full gradient, which is extremely costly

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w)$$

Idea: stochastic optimization employs random sample (mini-batch) B

to approximate

$$\nabla f(w) \approx \frac{1}{|B|}\sum_{i\in B} \nabla f_i(w)$$

- It is an unbiased estimator
- more efficient computation but introduces variance

Want to optimize

$$\min_{w} f(w)$$

Full gradient $\nabla f(w)$.

Given unbiased random estimator $\mathbf{g}_i$ of $\nabla f(w)$, and SGD rule

$$w \to w - \eta \mathbf{g}_i,$$

SCOPE

Objective function

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) = \frac{1}{n}\sum_{i=1}^{n} \tilde{f}_i(w),$$

where

$$\tilde{f}_i(w) = f_i(w) - \underbrace{(\nabla f_i(\tilde{w}) - \nabla f(\tilde{w}))^\top w}_{\text{sum to zero}}.$$

Pick $\tilde{w}$ to be an approximate solution (close to w_*).
The SVRG rule (control variates) is

$$w_t = w_{t-1} - \eta_t \nabla \tilde{f}_i(w_{t-1}) = w_{t-1} - \eta_t \underbrace{[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w})]}_{\text{small variance}}.$$

SCOPE

Procedure SVRG

Parameters update frequency m and learning rate η
Initialize $\tilde{w}_0$
Iterate: for $s = 1, 2, \ldots$
  $\tilde{w} = \tilde{w}_{s-1}$
  $\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} \nabla \psi_i(\tilde{w})$
  $w_0 = \tilde{w}$
  Iterate: for $t = 1, 2, \ldots, m$
    Randomly pick $i_t \in \{1, \ldots, n\}$ and update weight
      $w_t = w_{t-1} - \eta(\nabla \psi_{i_t}(w_{t-1}) - \nabla \psi_{i_t}(\tilde{w}) + \tilde{\mu})$
  end
  Set $\tilde{w}_s = w_m$
end

SCOPE

Algorithm 1 Task of Master in SCOPE

Initialization: p Workers, $\mathbf{w}_0$;
for $t = 0, 1, 2, \ldots, T$ **do**
  Send $\mathbf{w}_t$ to the Workers;
  Wait until it receives $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_p$ from the p Workers;
  Compute the *full gradient* $\mathbf{z} = \frac{1}{n}\sum_{k=1}^{p}\mathbf{z}_k$, and then send $\mathbf{z}$ to each Worker;
  Wait until it receives $\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2, \ldots, \tilde{\mathbf{u}}_p$ from the p Workers;
  Compute $\mathbf{w}_{t+1} = \frac{1}{p}\sum_{k=1}^{p}\tilde{\mathbf{u}}_k$;
end for

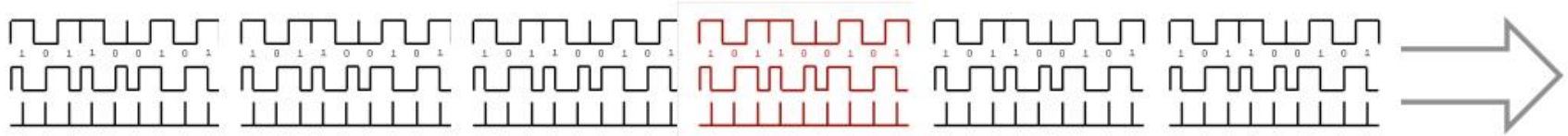
Algorithm 2 Task of Workers in SCOPE

Initialization: initialize η and c;
For the Worker_k:
for $t = 0, 1, 2, \ldots, T$ **do**
  Wait until it gets the newest parameter $\mathbf{w}_t$ from the Master;
  Let $\mathbf{u}_{k,0} = \mathbf{w}_t$, compute the *local gradient sum* $\mathbf{z}_k = \sum_{i\in\mathcal{D}_k}\nabla f_i(\mathbf{w}_t)$, and then send $\mathbf{z}_k$ to the Master;
  Wait until it gets the full gradient $\mathbf{z}$ from the Master;
  for $m = 0$ to $M - 1$ **do**
    Randomly pick up an instance with index $i_{k,m}$ from $\mathcal{D}_k$;
    $\mathbf{u}_{k,m+1} = \mathbf{u}_{k,m} - \eta(\nabla f_{i_{k,m}}(\mathbf{u}_{k,m}) - \nabla f_{i_{k,m}}(\mathbf{w}_t) + \mathbf{z} + c(\mathbf{u}_{k,m} - \mathbf{w}_t))$;
  end for
  Send $\mathbf{u}_{k,M}$ or the average of these $\{\mathbf{u}_{k,m}\}$, which is called the *locally updated parameter* and denoted as $\tilde{\mathbf{u}}_k$, to the Master;
end for

Big *Streaming* Data Processing

Fraud detection in bank transactions

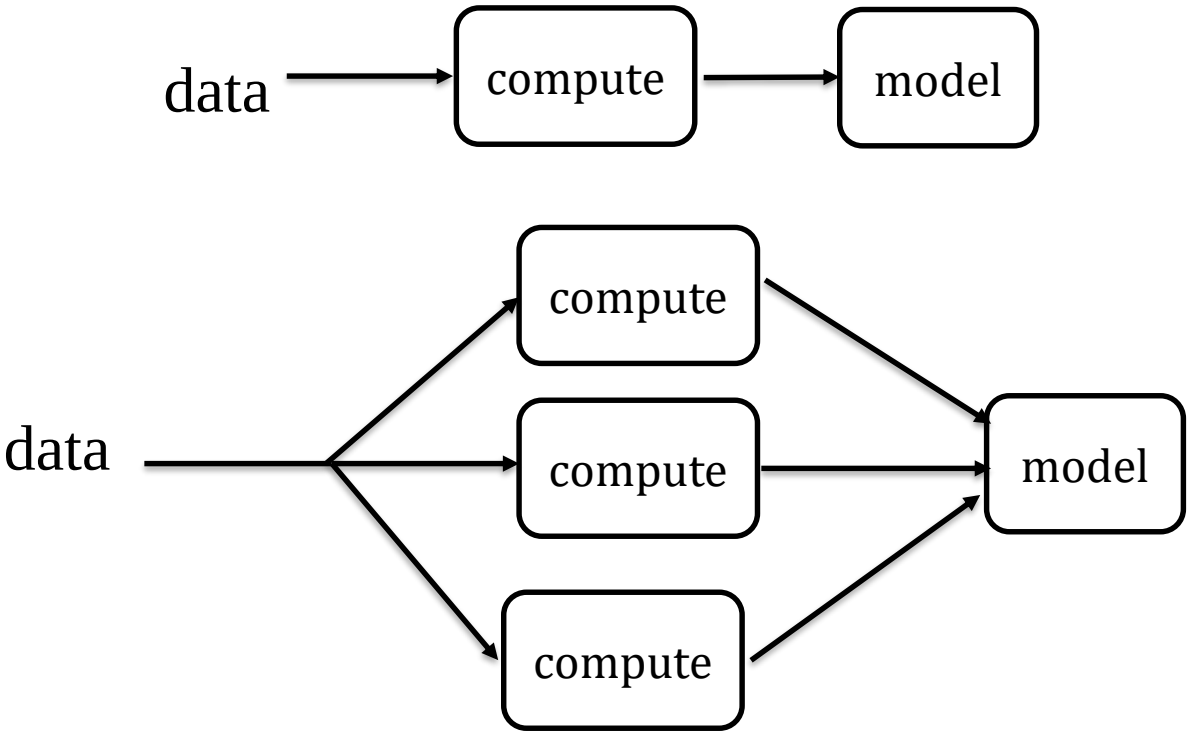
Anomalies in sensor data



Cat videos in tweets

Big *Streaming* Data Processing

How can we take advantage distributed resource to solve the problem of data streaming mining. Data Stream is **sequential, but if we handle data sequential, we only can use one machine.**

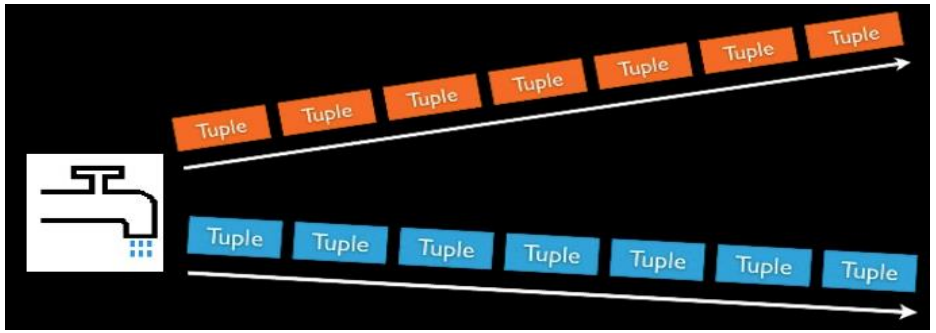


Spark Streaming

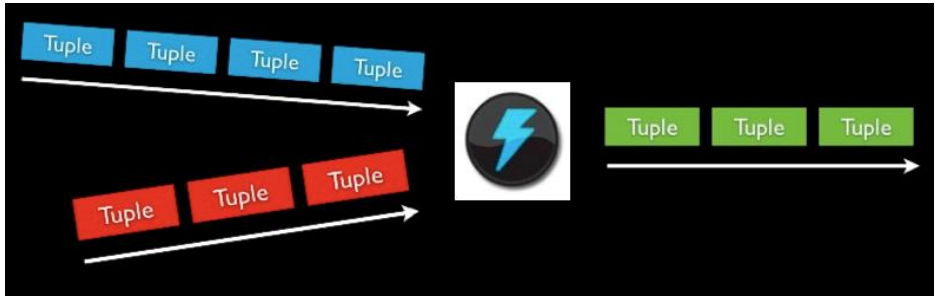
Internally, it works as follows. Spark Streaming receives live input data streams and divides the data into batches, which are then processed by the Spark engine to generate the final stream of results in batches.

Storm Concepts

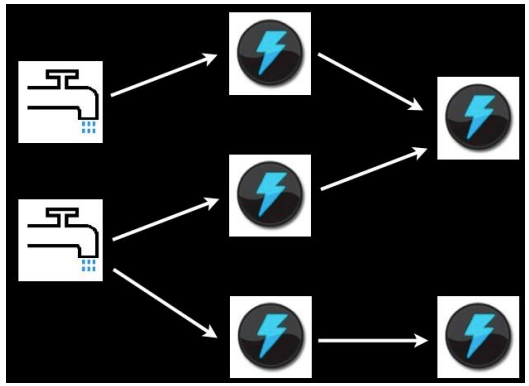
1. Streams
   - Unbounded sequence of tuples



2. Spout
   - Source of Stream
   - E.g. Read from Twitter streaming API



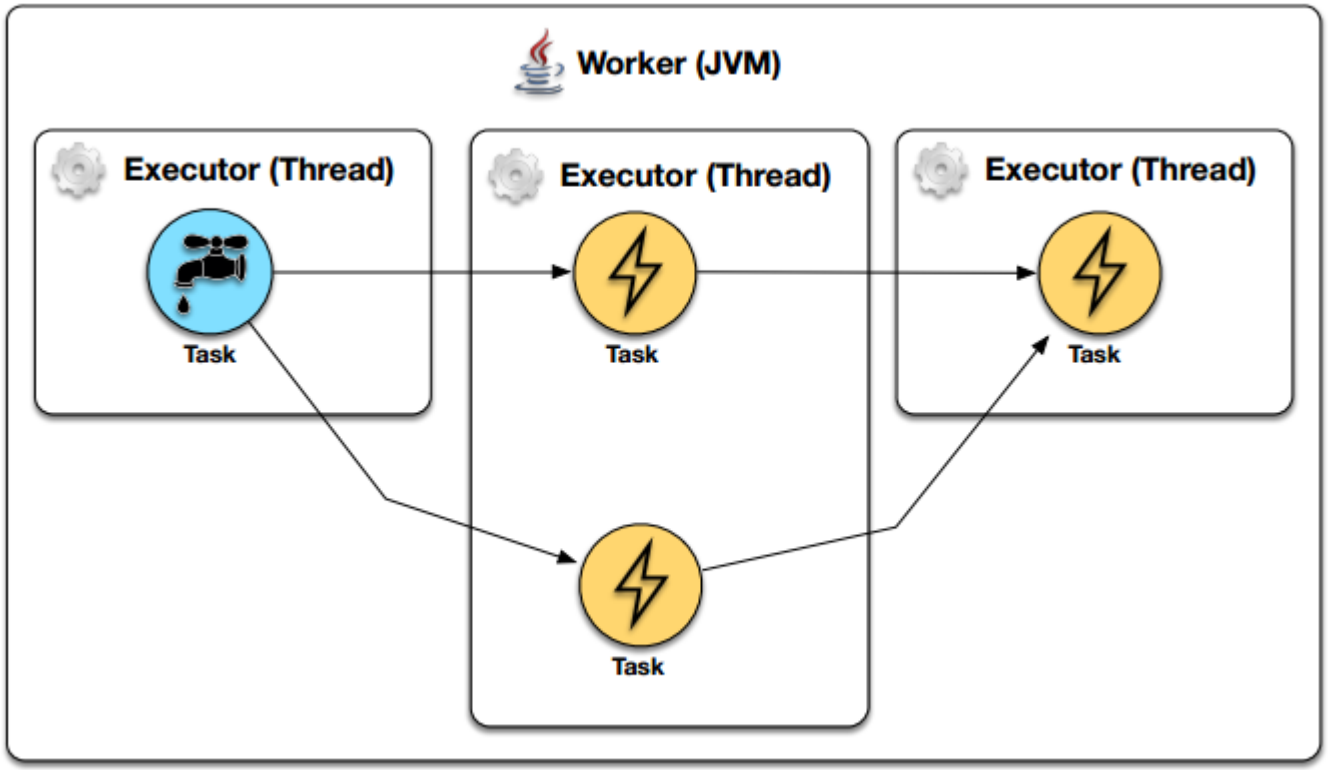
3. Bolts
   - Processes input streams and produces new streams
   - E.g. Functions, Filters, Aggregation, Joins



4. Topologies
   - Network of spouts and bolts

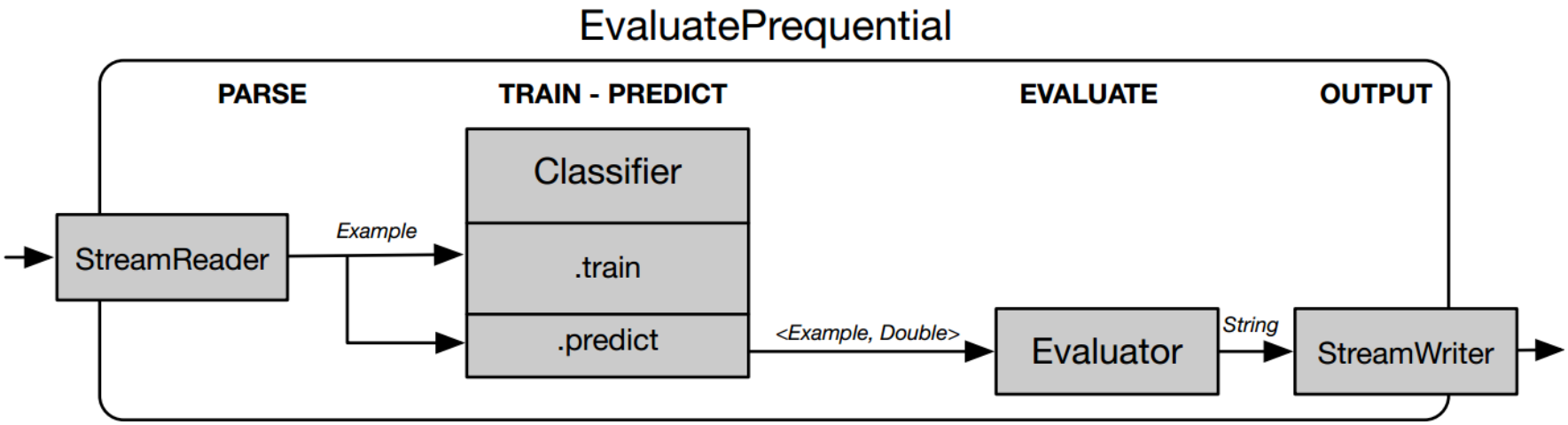
Storm

A Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation however needed



Spark Streaming

Update model by using mini-batch model.



Spark – Stream kmeans

For each batch of data, we assign all points to their nearest cluster, compute new cluster centers, then update each cluster using:

$$c_{t+1} = \frac{c_t n_t \alpha + x_t m_t}{n_t \alpha + m_t}$$

$$n_{t+1} = n_t + m_t$$

Where c_t is the previous center for the cluster, n_t is the number of points assigned to the cluster thus far, x_t is the new cluster center from the current batch, And m_t is the number of points added to the cluster in the current batch. The decay factor α can be

Thanks