MS&E 317/CS 263: Algorithms for Modern Data Models, Spring 2014
`http://msande317.stanford.edu`.
Instructors: Ashish Goel and Reza Zadeh, Stanford University.

Lecture 1, 3/31/2014. Scribed by Neeloy Banerjee.

1 Introduction

1.1 Course summary

- Distributed computing
- Performance Measures
- Distributed systems
  - Distributed hash tables (DHT)
  - MapReduce (MR)
  - Pig/Hive
  - Spark
  - Storm

1.2 What does 'modern' mean

Modern data models has a large overlap with distributed data models and this is to do with scaling. Computer clock speed is not increasing any more, due to an inability to dissipate heat fast enough inside the processor, so we add more cores and more machines instead. This allows us to use cheap cores and scale efficiently (just add more machines to the cluster).

This new computing paradigm has extra costs - communication costs - as there is no shared memory. We must build algorithms for these new models.

The biggest problem for distributed computing paradigms is *fault tolerance*. If the chance of a machine going down on a particular day is 0.001, that is an acceptable probability for one machine. With a cluster of 1000 machines, we expect one to go down each day and we expect the system to run it without those faults affecting our calculations.

2 Distributed systems

2.1 Distributed hash tables

A hash table whose contents are spread across the memory of multiple machines, eg

- SSTables at Google (DHT for web pages)

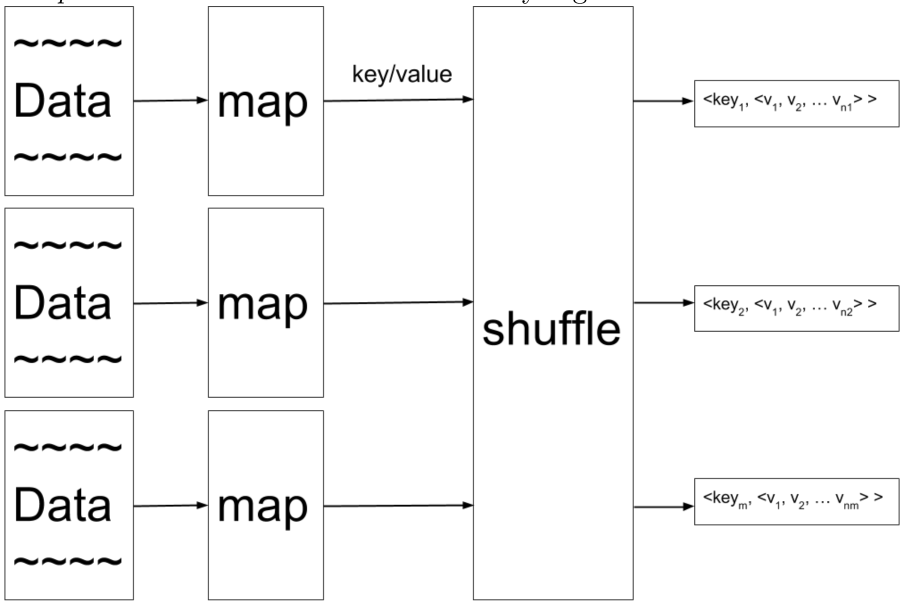
- Apache Cassandra - the open source version

NB: alongside the DHT there exist distributed file systems:

- Hadoop Distributed File System
- Google File System

2.2 MapReduce

MR gives you an interface that blackboxes the recovery/fault resilience of distributed systems.
Map phase: Input data, output key-value pairs
Reduce phase: Puts values with the same key together



2.2.1 Inverted index MR

Mappers take in web pages

```
mapper(webpage d)
    for w in d    // w are the words in webpage d
        Emit(w, docid(d))

reduce( <w, <t1, t2, ... , tn > >)    // t1 .. tn are the ids of the webpages w occurs in
    put <w, <t1, t2, ... , tn > > in a DHT and serve them
    // assume it can fit in a single machine's RAM
```

2.3 Pig/Hive

We want to auto-generate MR jobs from SQL code. Rather than writing each MR job for every SQL statement, Pig and Hive are tools that:

- contain a subset of SQL functionality
- operate on tables of data and do SQL-like things
- do away with the pain of writing multiple MR jobs whenever we want to do an SQL-like call

No need to write MR jobs in Pig/Hive. Can instead write the SQL equivalents 'JOIN names by id, grades by id' or 'FILTER names WHERE name= "Reza"

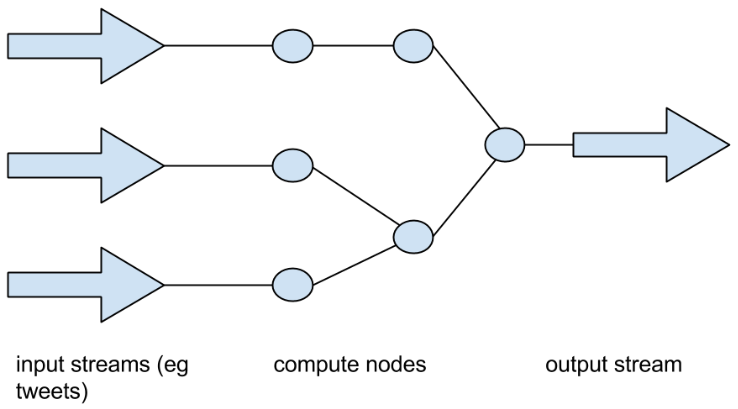
2.4 Spark

Spark contract: imagine some vector in any language that was *distributed*. Spark provides distributed collections (resilient distributed datasets) which are immutable.

- Does all the things you're used to with a vector
- Spork, Shark are the Pig, Hive for Spark
- Records history of RDD creations leads to fault tolerance

2.5 Storm

Defines streams and bolts. Data comes in streams, is operated on by bolts, and pushed out in streams.



2.6 Summary remarks

MR scales the largest (scales to 10,000s machines) and is the most mature.
We can't analyze just by the number of operations but also by the amount of communication done.