Motivation: Google Example

- 20+ billion web pages x 20KB = 400+ TB
- 1 computer reads 30-35 MB/sec from disk
- ~4 months to read the web
- ~1,000 hard drives to store the web
- Much more time to do something useful with the data!
Cluster Architecture

1 Gbps between any pair of nodes in a rack

2-10 Gbps backbone between racks

Each rack contains 16-64 nodes

In 2011 it was guestimated that Google had 1M machines, [http://bit.ly/Shh0RO](http://bit.ly/Shh0RO)

Machines fail:

- One server may stay up 3 years (1,000 days)
- If you have 1,000 servers, expect to loose 1/day
- People estimated Google had ~1M machines in 2011
  - 1,000 machines fail every day!
Distributed File System

- **Master node (Name Node in Hadoop):** keeps metadata
MapReduce: Overview

- Sequentially read units of data
- **Map:** for each unit, extract some “thing” you care about
- **Group by key:** Sort the things and Shuffle
- **Reduce:** Aggregate, filter, transform all the “equal” things
- Write the result
MapReduce: The Map Step

Input key-value pairs

Intermediate key-value pairs

...
MapReduce: The **Reduce** Step

**Intermediate key-value pairs**

- \( k \)
  - \( v \)
- \( k \)
  - \( v \)
- \( k \)
  - \( v \)

**Group by key**

**Key-value groups**

- \( k \)
  - \( v \)
  - \( v \)
  - \( v \)
- \( k \)
  - \( v \)
  - \( v \)

**Output key-value pairs**

- \( k \)
  - \( v \)
- \( k \)
  - \( v \)
- \( k \)
  - \( v \)
The crew of the space shuttle Endeavor recently returned to Earth as ambassadors, harbingers of a new era of space exploration. Scientists at NASA are saying that the recent assembly of the Dextre bot is the first step in a long-term space-based man/machine partnership. "The work we're doing now -- the robotics we're doing -- is what we're going to need ………………….
**Map-Reduce: A diagram**

**MAP:**
Read input and produces a set of key-value pairs

**Intermediate**

**Group by key:**
Collect all pairs with same key
(Hash merge, Shuffle, Sort, Partition)

**Reduce:**
Collect all values belonging to the key and output

---

**Big document**

**Input**

**MAP:**
Read input and produces a set of key-value pairs

**Intermediate**

**Group by key:**
Collect all pairs with same key
(Hash merge, Shuffle, Sort, Partition)

**Reduce:**
Collect all values belonging to the key and output

**Output**
map(key, value):
  // key: document name; value: text of the document
  for each word w in value:
    emit(w, 1)

reduce(key, values):
  // key: a word; value: an iterator over counts
  result = 0
  for each count v in values:
    result += v
  emit(key, result)
Hadoop: Environment

Hadoop environment takes care of:

- Partitioning the input data
- Scheduling the execution across a set of machines
- Performing the group by key step
- Handling machine failures
- Managing required inter-machine communication
Back to our word counting example:

- Combiner combines the values of all keys of a single mapper (single machine):

  Much less data needs to be copied and shuffled!