# MapReduce:<br>Simplified Data Processing on Large Clusters

Jeffery Dean and Sanjay Ghemawat
Google, Inc.

Kelsey O’Brien

12 November, 2013

# Main Idea

Give a detailed explanation of the MapReduce programming model and the associated implementation for processing and generating large data sets.

# What is MapReduce?

- Programming model used to process large data sets

- User specified *map* function processes a key/value pair and generates a set of intermediate key/value pairs.

- User specified *reduce* function then merges all intermediate values associated with the same key.

# Implementation

- Specific implementation depends on the environment
- Google’s specific implementation:

1. MapReduce library in the user program splits input files into M pieces and starts up copies of the program on cluster of machines.
2. One copy is the master who assigns *map* tasks or *reduce* tasks to the rest (called workers).
3. Map workers read input, parse key/value pairs, pass pairs to *map* function, and buffers results in memory
4. The buffered pairs are sometimes written to local disk. Locations on disk are passed back to master for forwarding to *Reduce.*
5. Reduce workers read intermediate data from local dicks and sort by the intermediate keys.
6. Reduce worker iterates over sorted data, passes each unique key and corresponding data to *Reduce* function, and appends to final output file.
7. When all map and reduce tasks are complete, master wakes up user program and returns back to the user code.

# Analysis

- Using MapReduce is very logical and straight forward way to process big data.
- Google’s implementation of MapReduce is impressive
  - Advantages outweigh disadvantages
  - Seem to handle all “disadvantages” presented
  - Real world use cases relevant across many industries

# Advantages & Disadvantages

- Computations are conceptually straightforward
- Resilient to large-scale worker failure
- Input data is stored on local disks of the machines in a cluster to conserve bandwidth
- Includes handlers for bugs in user code
- Status pages presented to user
- Includes a counter facility
- Guaranteed order
- Combiner functionality provided

- Because of large input data computations must be spread across hundreds/thousands of machines in order to finish in reasonable amount of time
- Must tolerate machine failures
- Must tolerate worker failures
- Bugs in user code
- Debugging problems in *Map* or *Reduce* functions is tricky

# Real-World Use Cases

- **Counting URL access frequency**
- **Reverse web-link graph** – for each URL find all pages pointing to it
- **Grep** – find all likes matching some pattern
- **Inverted Index** – Takes a given word and creates a list of documents containing that word
- **Distributed Sorting**
- **Tern-vector per host** – Summarizes most important words that occur in a document of set of documents