MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

Presenter: Yi Zhang

October 30, 2014

Acknowledgement: all figures from the presented paper.

Overview

Problem Setting

- In a distributed system
- Goal is to process large amount data
- Fast (in parallel)

Motivation

- Provide a uniform interface for data processing
- Automatic parallelization and fault tolerance

What is MapReduce

- A general purpose distributed programming framework and its implementation
- Can be used for large data processing and analysis

Map and Reduce

To submit: data, functions `map`, and `reduce`

$$\begin{tikzcd}\text{Data} \arrow[r, "\texttt{map}"] & \text{Intermediate pairs} \arrow[r, "\texttt{reduce}"] & \text{Results}\end{tikzcd}$$

map	`(k1,v1)`	→ `list(k2,v2)`
reduce	`(k2,list(v2))`	→ `list(v2)`

Example: word counting

- Count the number of occurrences of each word in a collection of documents

$$\begin{tikzcd}\text{Data} \arrow[r, "\texttt{map}"] & \text{Intermediate pairs} \arrow[r, "\texttt{reduce}"] & \text{Results}\end{tikzcd}$$

`map`: (docName, docContent)

for each word, produce:
<word, 1>

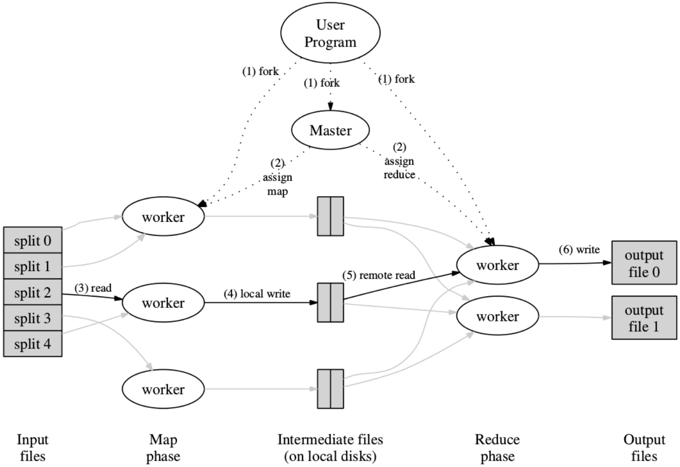
`reduce`: (word, listOfPairs<word,1>)

count number of pairs in the list
produce: n

Other Examples

- Count of URL access frequency
- Inverted index
- Distributed sort and grep

Execution Overview



Fault Tolerance

- Worker: master pings every worker, reassign jobs to others if no response
- Master: just start over again

Backup Tasks

- Machine could take unusually long towards the end
- Master schedules backup executions of the remaining in-progress tasks
- Job is complete when either is done

Bad Records

- Sometimes `map` or `reduce` crashes deterministically on certain records
- If more than one failures on certain records, skip them

Status Information

- Master runs an internal HTTP server
- Master exports progress information to status pages

Local Execution

- Have the option to sequentially execute MapReduce locally
- For debugging, profiling purposes

Performance Comparison

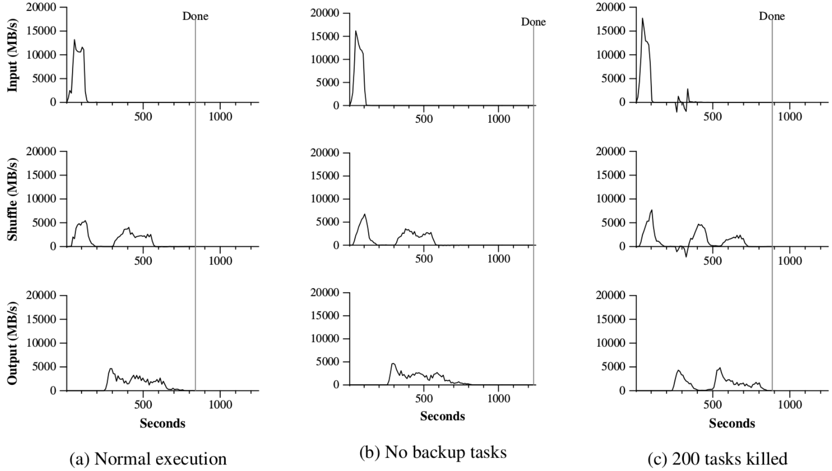


Figure 3: Data transfer rates over time for different executions of the sort program

Benefits

- Widely applicable to many tasks
- Uniform interface for developers to submit jobs
- Automatic parallelization, local optimization, load balancing, fault tolerance
- Highly scalable

MapReduce

Thank you.