MapReduce: Simplified Data Processing on Large Clusters

Liam Cain

November 24, 2013

CMPT308

Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters." Google, Inc., 2004. 23 Nov. 2013.

What is MapReduce

- Highly scalable programming model
- specify a `map` and `reduce` function to be run in parallel across a large cluster of commodity machines

	Input		Output
`Map`	`(key1, value1)`	→	`list(key2, value2)`
`Reduce`	`(key2, list(value2))`	→	`list(v2)`

- `Map` returns a list of keys and values
- Those values are then groups by key and passed to `reduce`.

Implementation

- User sends `map` and `reduce` functions to a master computer
- **Master computer**:
  - **Assigns workers** to perform either `map` or `reduce` on a partition of the overall data
  - Handles **load balancing** to maximize efficient use of each computer
  - Reassigns tasks in the event of **worker failure**
  - Uses **redundancy** to minimize risk in event of worker failure

Map Worker

- Performs `map` function
- Writes output to local disk

Reduce Worker

- Remotely reads data
- Performs `reduce` function
- Remote write to output file

Analysis

- Processing data in large clusters is a daunting task and has become increasingly important in recent years

- MapReduce provides a simple interface for programmers to perform analysis on terabytes of data without worrying about:
  - parallelized data
  - redundancy
  - load balancing
- Using commodity machines makes MapReduce cost effective and easily scalable
- Perfect for performing simple data analysis tasks on large amounts of data.

Advantages

- Simple Interface
- Highly scalable
- Extendible
  - Can specify the number of reduce tasks or output files
  - Add a `combiner` function
    - An additional function to combine values on the machine that performs a map task
  - Users can add support for inputting custom data types

Disadvantages

- Not everything can fit into the `map` and `reduce` model
  - Simple tasks often have to be rethought to fit the key/value model
- Rigid Model
  - Cannot adapt on-the-fly
- Batch processing
  - Doesn't support an incoming stream of information

Real-World Use Cases

- Indexes search data for Google web search

- Large scale tasks
  - Machine learning problems
  - Graph computations
  - Extracting information from web pages