# Recitation 3

Song Bo 11302010003

April 17, 2013

## 1 Motivation

Suppose we want to solve a large scale problem and we have a lot of machines, how can we do it efficiently? MapReduce technology gives us a model that divide a big problem into many small problems in order to improve parallelism among many machines. The big problem and small problems are assemble in essence, but differs from each other in scale.

## 2 Implementation

### 2.1 Interfaces

There are two operations that the user must define:

- **Map** - Take an input pair and produces a set of intermediate key/value pairs.
- **Reduce** - Accept an intermediate key I and a set of values for that key, then it mergse together these values to form a possibly smaller set of values.

And several optional user defined operations that improve performance:

- **Input Reader** - Define how to extract initial key/value pairs.
- **Output Writer** - Define how to write the final result in user desired form.
- **Partition Function** - Partition the intermediate pairs to desired groups.
- **Compare Function** - Used in sorting in Reduce.
- **Combiner Function** - An optimized reduce function running after map and running on map machine.

### 2.2 Execution

There are seven main steps:

- **1** Divide program inputs and fork.
- **2** One master and serval worker scheduled by master.

- **3** Map workers read inputs, do their job and buffer the intermediate pairs.
- **4** Load pairs in local disk and inform master to notify reduce workers.
- **5** The notified reduce worker sorts the intermediate keys
- **6** The reduce worker does it job and writes to each output.
- **7** The master wakes up user program.

## 2.3 Failure Tolerance

Except for main execution steps, the system deals with worker and master failure properly and skips bad records.

### 2.3.1 Worker

The master pings every worker periodically. Then the master re-execute the work on another worker with a mechanism.

### 2.3.2 Master

Aborting the MapReduce computation if the master fails. But I wonder why not using checkpoint strategy instead of letting clients do it.

# 3 Summary and Thoughts

Managing failure situation in the papers impresses me most. With the increasing of the scale of modern system, the occurrence and importance of different kinds of failure beyonds my belief. In MapReduce, disk failure may happen every minute among thousands of disk.

The scalability of it is also impressive. When the some disks or machines are added to the whole system, there is only a few things to apply map-reduce to the new system, since the map-reduce does not need some very specific information and there is no strong link between each machine.