# Recitation 3

Song Bo 11302010003

April 9, 2013

# 1 MapReduce

## 1.1 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.

## 1.2 Implementation

### 1.2.1 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.

### 1.2.2 Exection

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.

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

# 2 Redundant Arrays of Inexpensive Disks

## 2.1 Motivation

An individual high Performance and big size disk is very expensive and there is a big gap between CPU improvement rate and IO improvement rate. The RAID technology aims to use several low-power, small size and inexpensive disks to accomplish the same or better performance.

## 2.2 Implementation

- **AID** - Arrays of inexpensive disks. High performance and very poor reliability(MTTF is only 300 hours).
- **RAID level1** - Mirrored disk. High access time and reliability. Duplicating data leads to low storage utilization.
- **RAID level2** - Using hamming code which requires several chips to check data. It is suitable for huge data transfers.
- **RAID level3** - Using single check disk which contains ECC. Relative high performance and low reliability overhead.
- **RAID level4** - Using independent read or write which use check disk every time. Good for small writes and check disk is bottleneck.
- **RAID level5** - Distributing data and check information to all the disks to reduce check disk bottleneck. However, the write operation will cost actually four read and write operations.

# 3 Summary and Thoughts

Managing failure situation in the both two 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. In RAID, the MTTF becomes a main problem to solve and has pushed the development of RAID level by level.