Comparing Two Different MapReduce Implementations: Hadoop and Plasma

Dongwon Kim and **Sungwoo Park**
Pohang University of Science and Technology

MapReduce

A software **framework** introduced by **Google** for **distributed computing** on large data sets over computer clusters

High-level **abstraction** – *a great advantage of MapReduce*

- **Hide** all details of parallelism
- **Hide** machine management
- **Hide** fault tolerance

"**map**" and "**reduce**" functions – *what users need to define*

- **MapReduce processes data in the form of (key, value) pairs**
  - Each **map** produces an intermediate (key, value) pairs
  - Each **reduce** merges all intermediate (key, value) pairs associated with the same intermediate key
- These functions are motivated by **map** and **fold** functions from **functional languages**
  - **map :** map square [1, 2, 3, 4, 5] = [square 1, square 2, square 3, square 4, square 5] = [1, 4, 9, 16, 25]
  - **fold :** foldl (+) 0 [1, 2, 3, 4, 5] = (((((0 + 1) + 2) + 3) + 4) + 5) = 15

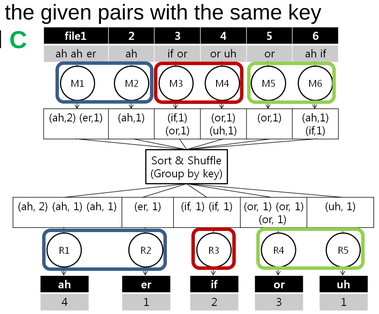
A MapReduce example

Counting how often each word occurs in a collection of text files

- **Each map task** counts the occurrence of each word in a given input file
- **Each reduce task** adds values of all the given pairs with the same key

Let's assume **3 machines**: **A**, **B**, and **C**

- M_i is a map task
- R_i is a reduce task
- At **Map** phrase
  - **A** : M_1 and M_2
  - **B** : M_3 and M_4
  - **C** : M_5 and M_6
- At **Reduce** phrase
  - **A** : R_1 and R_2
  - **B** : R_3
  - **C** : R_4 and R_5



All map & reduce tasks are executed in parallel

- Every map task is executed **independently**
- Every reduce task is executed **independently**

MapReduce takes care of

- Partitioning the input data
- Scheduling & distributing tasks
- Grouping intermediate (key, value) tuples
- Handing errors

What else can we do using MapReduce?

- Distributed Grep
  - **Each map** emits a line if it matches a given pattern
  - **Each reduce** is an identify function; it just copies the supplied intermediate data to the output
- Inverted Index
  - **Each map**
    (a) parses each document
    (b) emits a sequence of <word, document ID> pairs
  - **Each reduce**
    (a) accepts all pairs for a given word
    (b) sorts the corresponding document IDs
    (c) emits a <word, list(document ID> pair, which is an inverted index entry

MapReduce researches

Google

Representative publications:

- MapReduce: Simplified Data Processing on Large Clusters, **OSDI 2004**
- The Google File system, **SOSP 2003**
- MapReduce: a flexible data processing tool, **Comm. of ACM 2010**

➔ **MapReduce serves Google's purpose!**

Database community

Representative publications:

- A Comparison of Approaches to Large-Scale Data Analysis, **SIGMOD 2009**
- The Performance of MapReduce**:** An In-depth Study, **VLDB 2009**
- MapReduce and parallel DBMSs: friends or foes?, **Comm. of ACM 2010**

➔ **Can MapReduce substitute for DBMSs?**

System & Network community

Representative publications:

- Improving MapReduce Performance in Heterogeneous Environments, **OSDI 2008**
- The Hadoop distributed FileSystem: Balancing Portability and Performance, **ISPASS 2010**
- MapReduce Online, **NSDI 2010**

➔ **Let's measure MapReduce performance!**
➔ **How to modify MapReduce for various environments?**

MapReduce is one of the hottest topic in CS!!

What kind of MapReduce implementations they use?

- **Google** uses its own **proprietary** implementation
- **Most research papers** use **Hadoop** for their experiments

Apache Hadoop

What is Hadoop?

- An open-source MapReduce **implementation** written in **JAVA**
- **Hadoop** is a top-level Apache project
- **Yahoo!** has been the largest contributor to **Hadoop**
- **Yahoo!** uses **Hadoop** extensively across its businesses

High performance computing using Java?

- **Java** is well-known for its **Portability**, **Network centricity**, and **maintainability, not performance**

An article from the Hadoop homepage

- This news shows the excellence of Hadoop
- How about **implementing** MapReduce with other programming languages and **comparing** their performance with **Hadoop**?

July 2008 - Hadoop Wins Terabyte Sort Benchmark

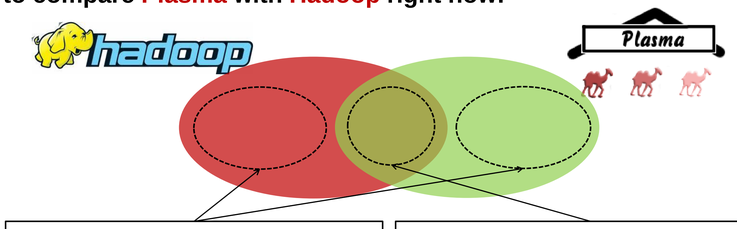
Hadoop Wins Terabyte Sort Benchmark: One of Yahoo's Hadoop clusters sorted 1 terabyte of data in 209 seconds, which beat the previous record of 297 seconds in the annual general purpose (Daytona) terabyte sort benchmark. This is the first time that either a Java or an open source program has won.

Plasma

What is Plasma?

- Another MapReduce implementation written in **Ocaml**
- Gerd Stolpmann's private project
- **Still in development and an alpha version is available**

We are to compare Hadoop and Plasma, but it is too premature to compare Plasma with Hadoop right now!



Inherent differences

- **Architectural** differences
  - Different block-size preference
  - Different workflows
- **Linguistic** differences
  - JVM-related issues
  - Portability issues

➔ **To do :** to find out **more inherent differences** between Plasma and Hadoop

Common features

Plasma needs to catch up with **Hadoop**

-More functionality needed

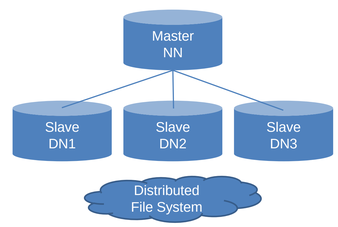
-More Optimization needed

➔**To do :** to **make a contribution to Plasma** in order to make the comparison **fair and square**

Preliminary experiments

We used total 4 machines: DN1, DN2, DN3 for **slaves** and NN for **the master**

	DN1	DN2	DN3	NN
# of cores	2	2	2	16
Clock speed (GHz)	3.4	3.0	3.2	1.6
Main Memory (GB)	2	2	2	8
Cache size (M)	2	2	2	2



Wordcount example (average execution time with a 300MB file)

- Hadoop : 69.43 sec (with some optimization techniques turned on)
- Plasma : 80.33 sec

➔ **Plasma** shows **comparable results** to **Hadoop** for its age
(Note that **Hadoop** is already **mature** enough for production use)

Ongoing work

- To analyze **Hadoop** in detail and to make a contribution to **Plasma**
- To find out inherent differences of **Hadoop** and **Plasma**
- To conduct experiments of **Hadoop** and **Plasma** on a cluster with 100~200 nodes