Plasma와 Hadoop MapReduce 환경 비교 분석

김동원, 임정표
Pohang University of Science and Technology

MapReduce

A software **framework** introduced by **Google**

Supporting **distributed computing** on **large data sets** on **clusters** of computers

High-level **abstraction**

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

Users just need to define **two placeholder functions**

: "**map**" and "**reduce**"

Motivated by **Map** and **Fold** functions in functional languages

map : ('a -> 'b) -> 'a list -> 'b list

ex) map square [1, 2, 3, 4, 5] = [square 1, square 2, square 3, square 4, square 5]
= [1, 4, 9, 16, 25]

fold : ('a -> 'b -> 'a) -> 'a -> 'b list -> 'a

ex) reduce (+) 0 [1, 2, 3, 4, 5] = (((((0 + 1) + 2) + 3) + 4) + 5)
= 15

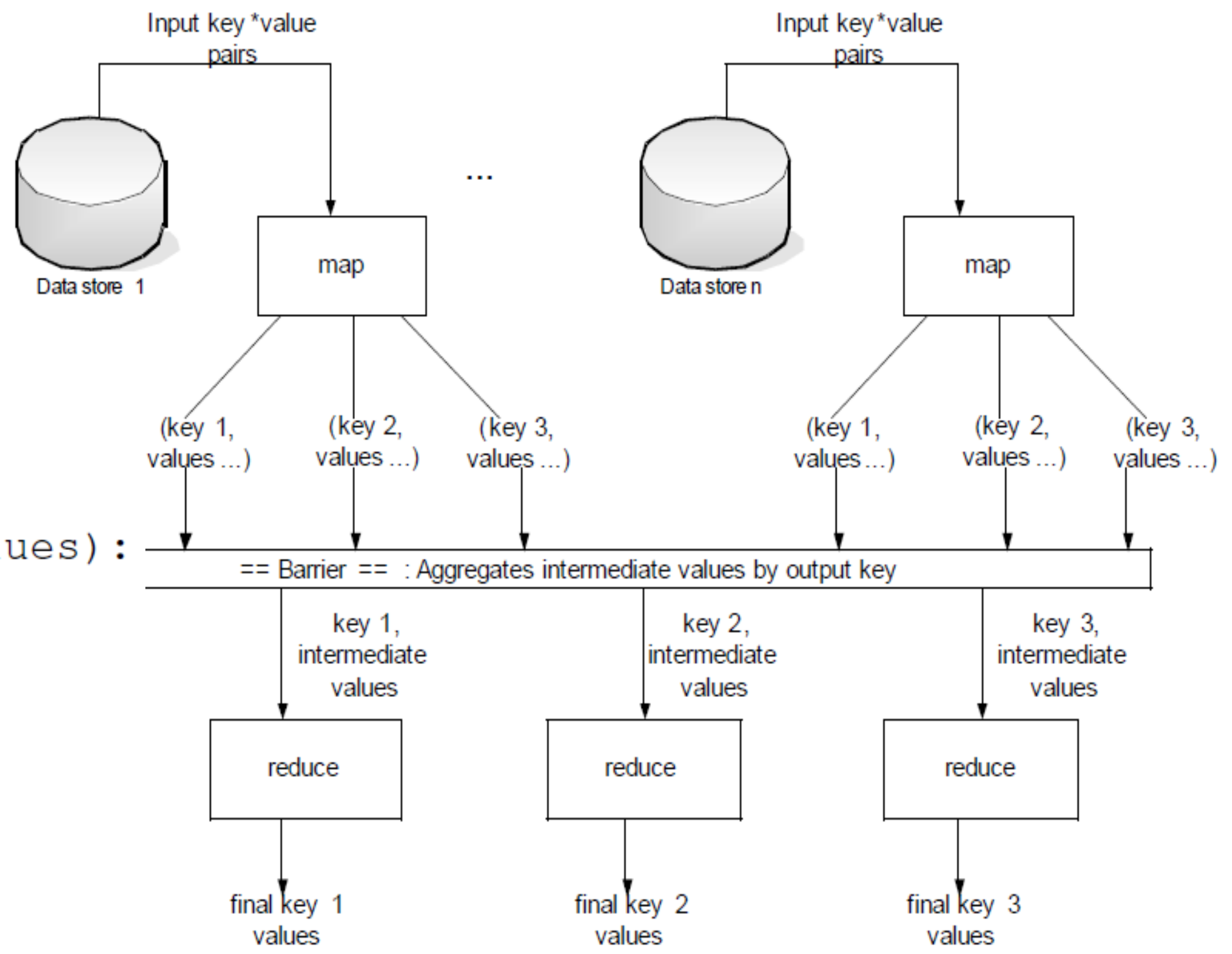
"Map" and "Reduce" in MapReduce framework

Map : (k1, v1) -> [(k2, v2)]

```
map(String key, String value
  // key: document name
  // value: document content
  for each word w in value:
    EmitIntermediate(w, "1")
```

Reduce : (k2, [v2]) -> [v3]

```
reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

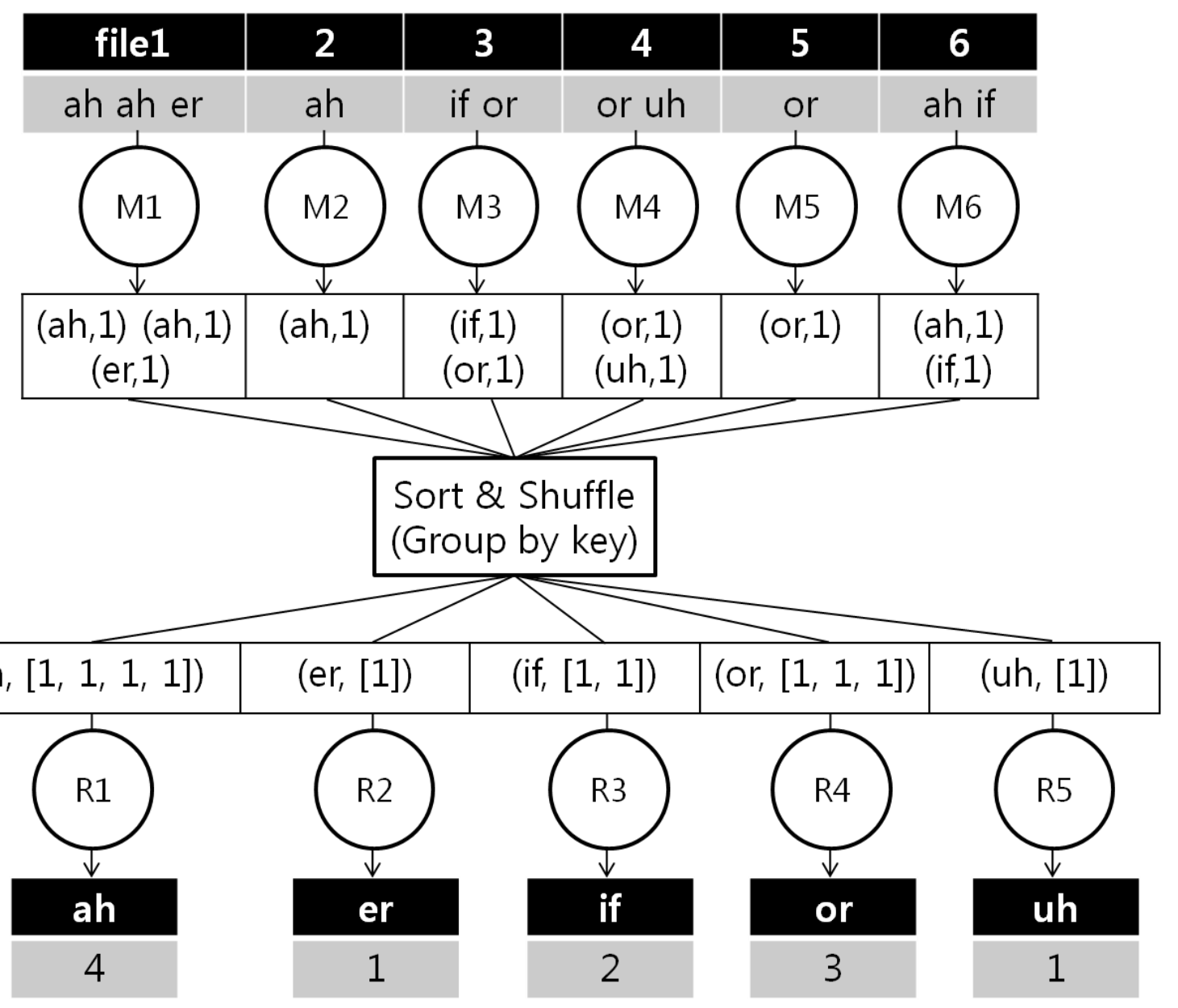


Every map task is executed **independently** by a map task process
Every reduce task is executed **independently** by a reduce task process

Example – Wordcount program

Assume we have **3 machines**: **A**, **B**, and **C**

-Mi is a map task
-Ri is a reduce task
-At **Map** phrase
  -A : take M1 and M2
  -B : take M3 and M4
  -C : take M5 and M6
-At **Reduce** phrase
  -A : take R1 and R2
  -B : take R3
  -C : take R4 and R5



<Word counting example>

Hadoop vs PlasmaMR

hadoop	A n open source MapReduce implementation written in Java A top-level Apache project with several subprojects Yahoo! Has been the largest contributor to Hadoop
Plasma	An open source MapReduce implementation written in **Ocaml** Gerd Stoplmann's private project

(1) Own distributed file systems(HDFS and PlasmaFS)

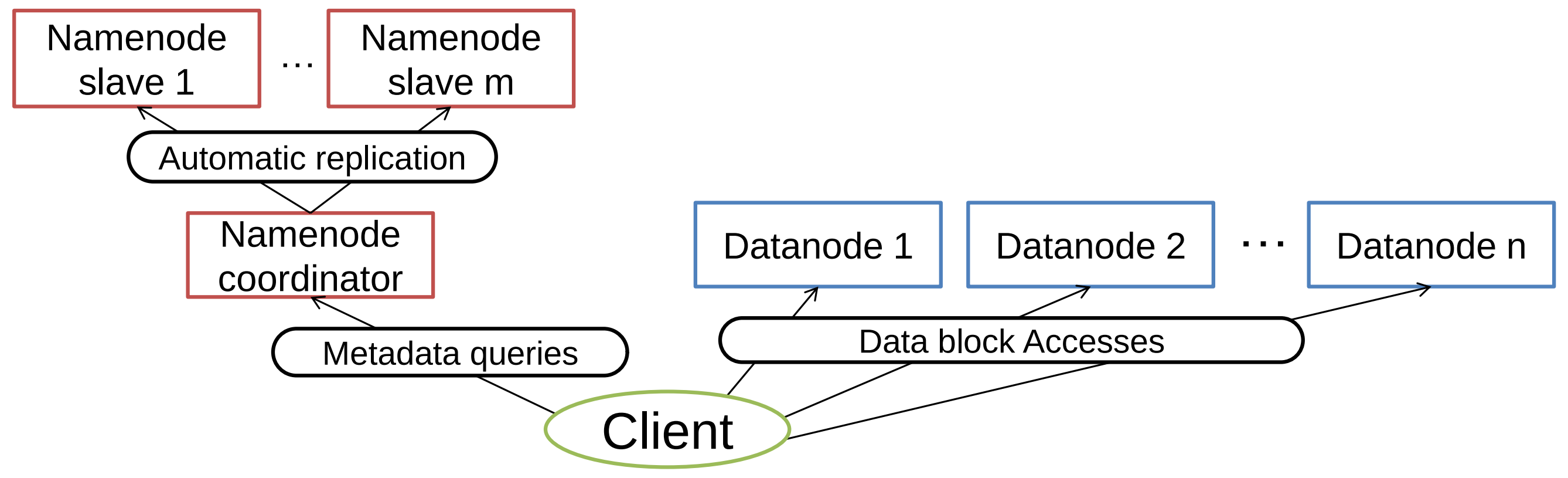
Similar DFS architectures

Distributed Running over a number of nodes
Replication Data blocks are stored in multiple replicas
Recovery Fault detection and quick, automatic recovery

Different block size preferences

HDFS(for Hadoop) prefers to use **more than 64MB**
- For high data locality in HDFS

PlasmaFS(for PlasmaMS) prefers to use **around 1MB**
- **Static allocation**(pre-allocation) allows a small block size
- Plasma achieves **high data locality** without a large block size
- Less memory consumption
- Better compatibility with small block software and protocols



HDFS and PlasmaFS have a similar distributed file system architecture

(2) Compiling programs

Hadoop
- Hadoop programs are compiled into **Java byte code**
- Hadoop is **platform-independent**
- Machines execute **each task process** on **a Java Virtual Machine**
- Individual JVM for each task process can be **a performance bottleneck**

Plasma
- Plasma programs are compiled into **machine code**
- Machines are required to have **the same platform** to execute machine code distributed by a client

(3) Shared memory issue

Hadoop
- Hadoop **does not** use shared memory

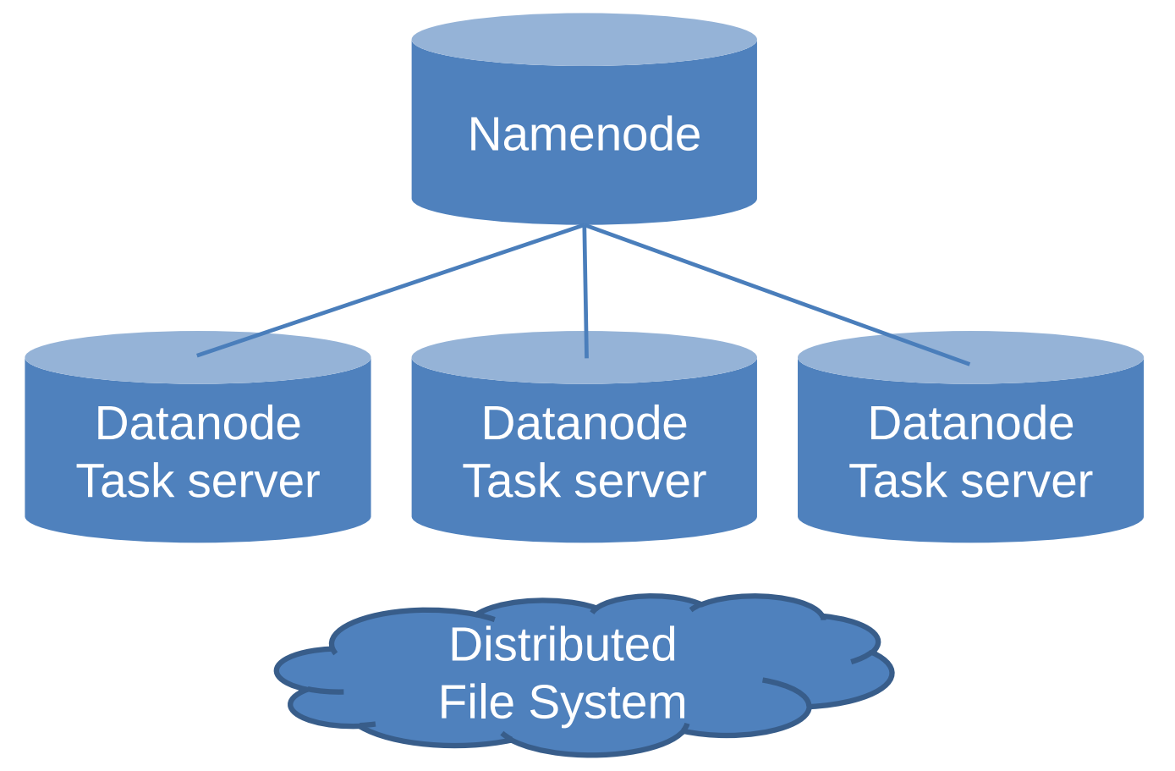
Plasma
- Plasma uses shared memory
- File buffers are kept in shared memory
- **Fast data paths** between **task processes** and **the datanode server** in the same machine

Preliminary experiments

We use total 4 machines, DN1, DN2, DN3, and NN:

DN1, DN2, DN3 for **datanodes and task servers** and NN for the namenode.

	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
- Plasma : 80.33 sec

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

Assumptions on Hadoop's poor scalability

Hadoop shows **poor scalability**

- A namenode with a large number of data nodes
  ➔ **An inherent problem of MapReduce frameworks**
  ➔ Efforts from Hadoop communities
- A **Java Virtual Machine** executes a single task
  ➔ **An inherent problem of Hadoop**
  ➔ Plasma distributes source files in the form of **machine code**

In addition, Plasma uses **shared memory** for **IPC** between datanode daemon processes and task processes
➔ Known to be difficult between JVMs

Ongoing work

Conduct experiments on a cluster with 100~200 nodes
- to identify **performance bottlenecks** of both Plasma and Hadoop

Find Plasma's distinct advantages as a MapReduce framework

Modify **Plasma source code** to improve its performance
- to make a contribution to the Plasma project