Yandex

MapReduce

MapReduce

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

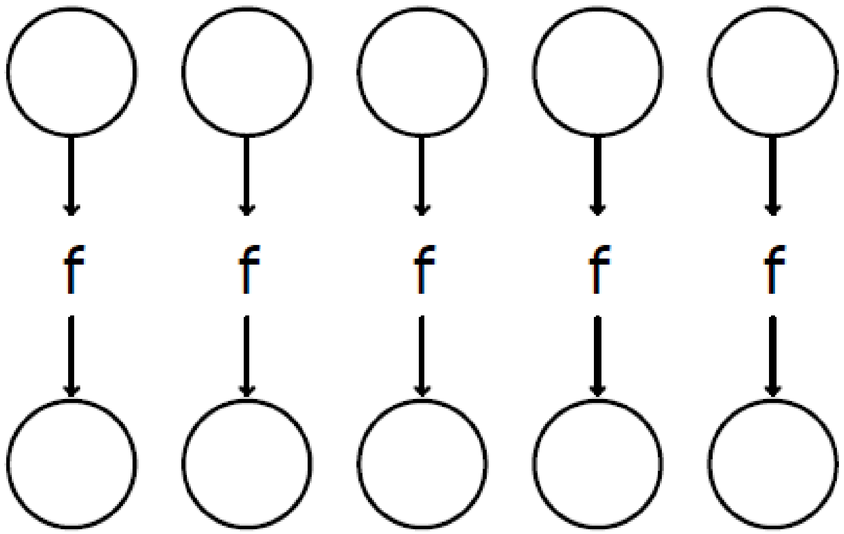
Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

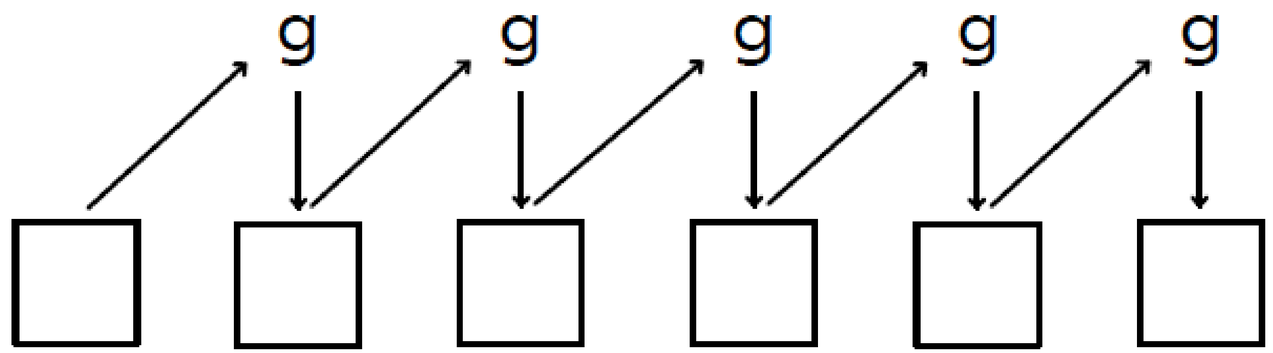
MapReduce: Simplified Data Processing on Large Clusters, Symposium on Operating Systems Design and Implementation (OSDI, 2004)

Map




```
>>> map(lambda x: x*x, [1,2,3,4])
[1, 4, 9, 16]
```

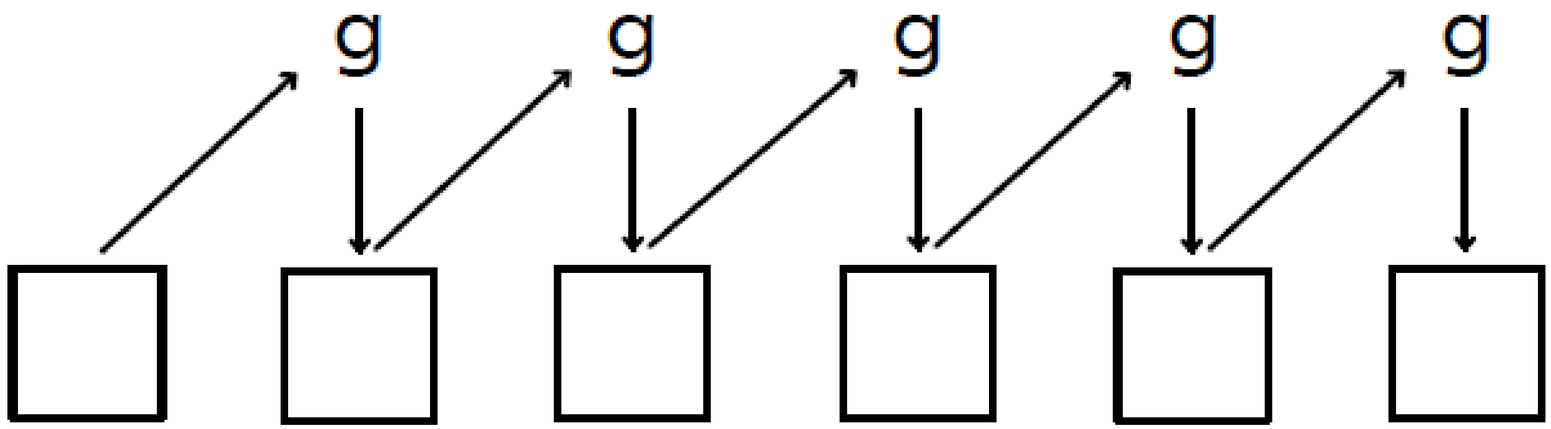
Fold / Reduce / Aggregate



왼쪽에서 오른쪽으로 누적 계산한다.

```
from functools import reduce
import operator
>>> reduce(operator.add, [1, 4, 9, 16])
>>> reduce(operator.add, [5, 9, 16])
>>> reduce(operator.add, [14, 16])
30
```

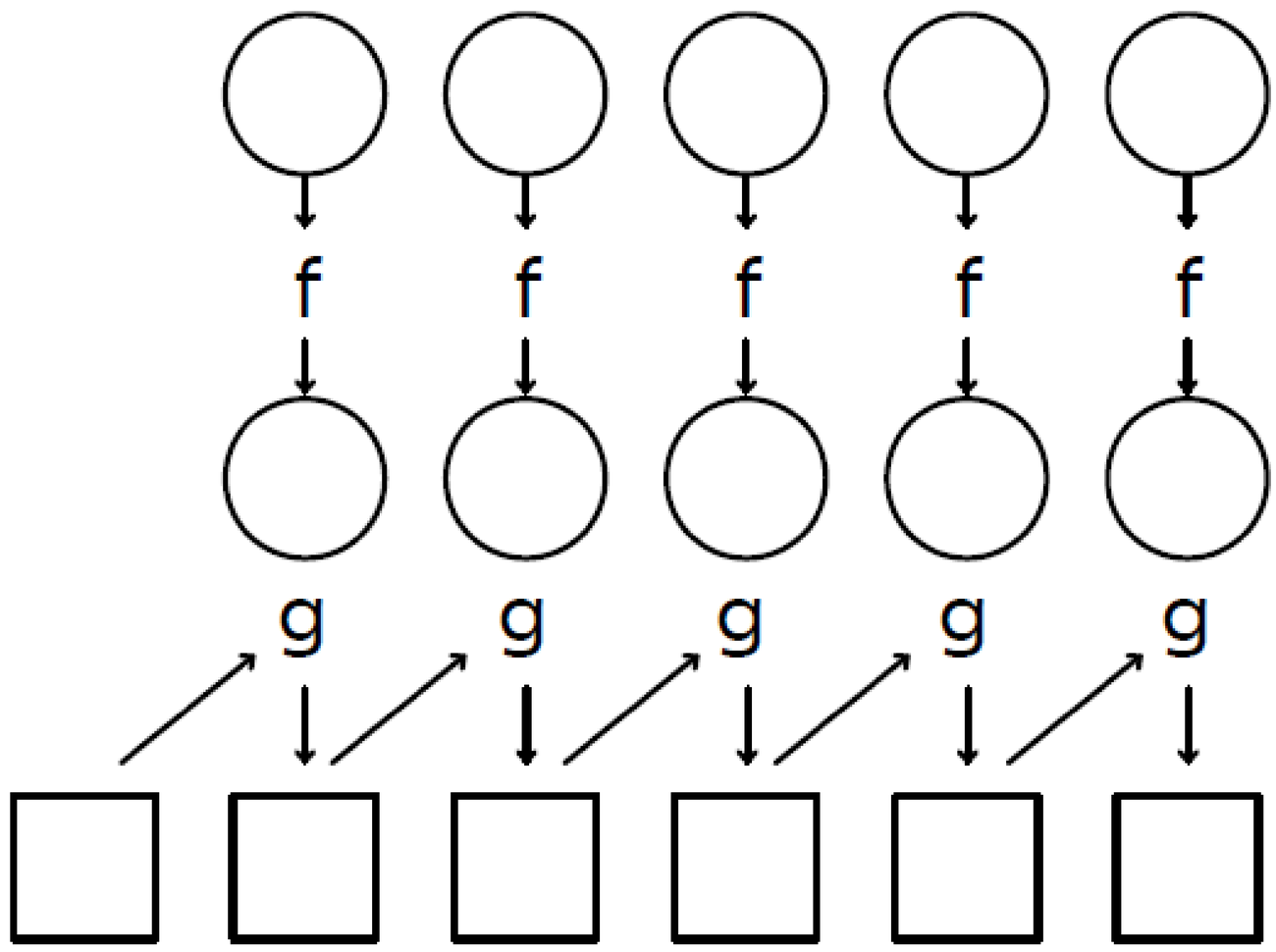
Fold / Reduce / Aggregate



평균 계산같은 경우는 계산 방향에 따라 값이 달라지는 문제가 있다.

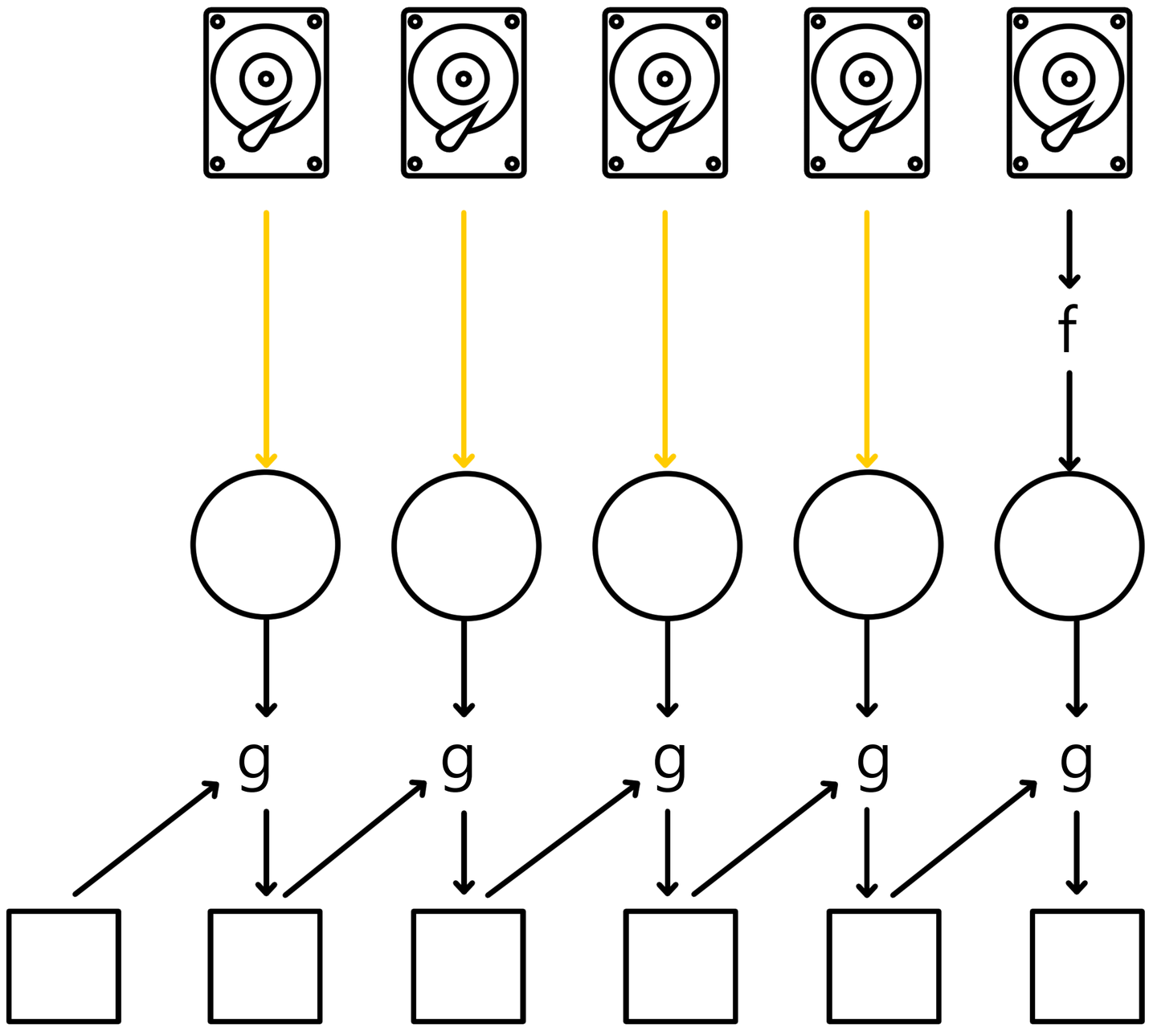
```
>>> average = lambda x, y: (x + y) / 2.
>>> reduce(average, [1, 2, 3])
>>> reduce(average, [1.5, 3])
2.25
>>> reduce(average, [3, 2, 1])
>>> reduce(average, [2.5, 1])
1.75
```

MapReduce




```
>>> reduce(operator.add, map(lambda x: x*x,
[1, 2, 3, 4]))
30
```

Distributed Shell: **grep 명령 수행**



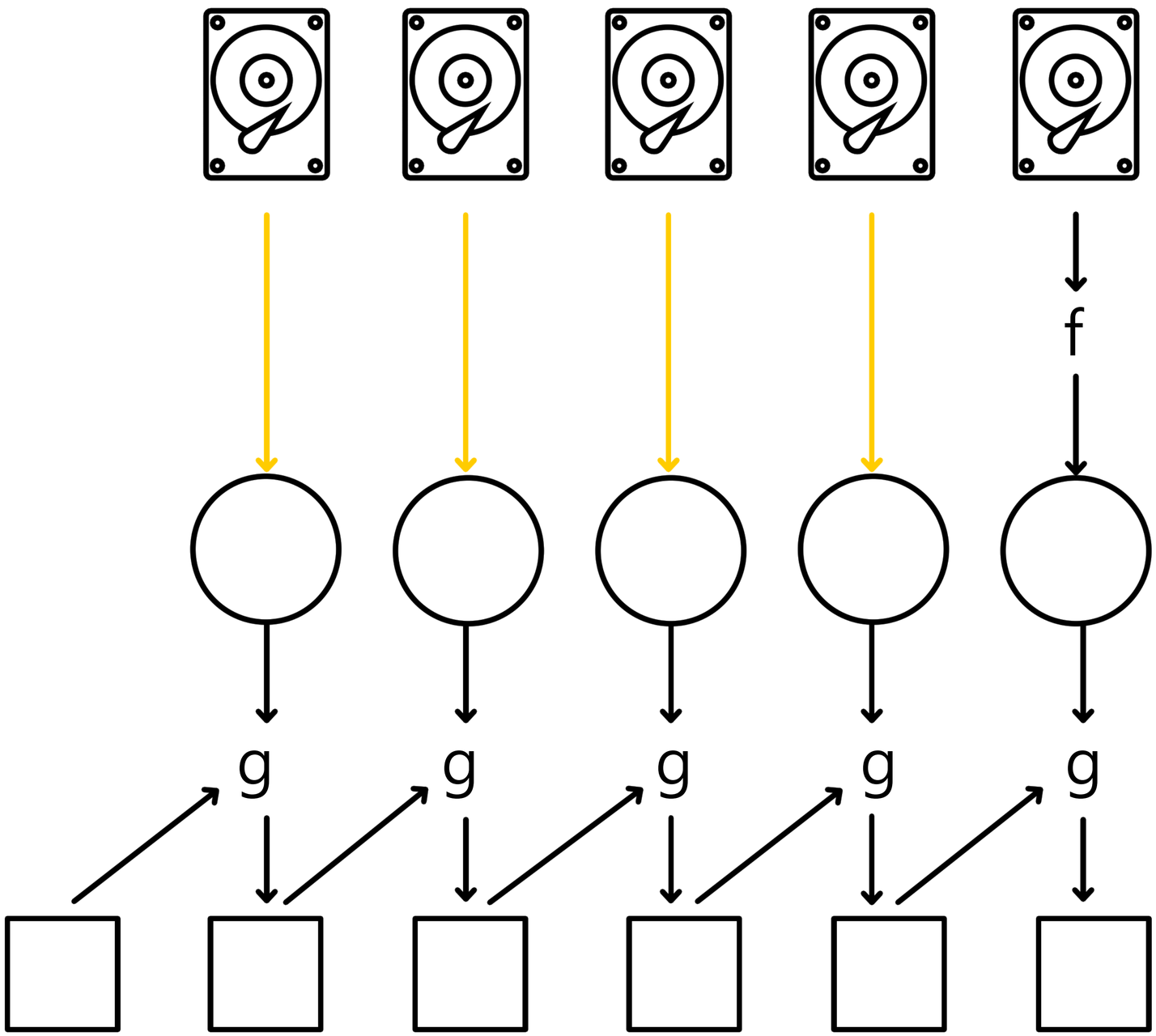
map = grep

단어를 찾는 작업은 map만 수행해서 결과만 합치면 된다.

reduce = None

MapReduce application에서 항상 map과 reduce를 둘 다 사용할 필요는 없다는 예시

Distributed Shell: **head 명령 수행**



~~map = head (?)~~

파일이 분산 저장되어 있기 때문에
head를 수행하기 위해
map 작업은 비효율적이고 복잡하다.

hdfs client 명령어로 수행할 수 있다.

~~reduce = None (?)~~

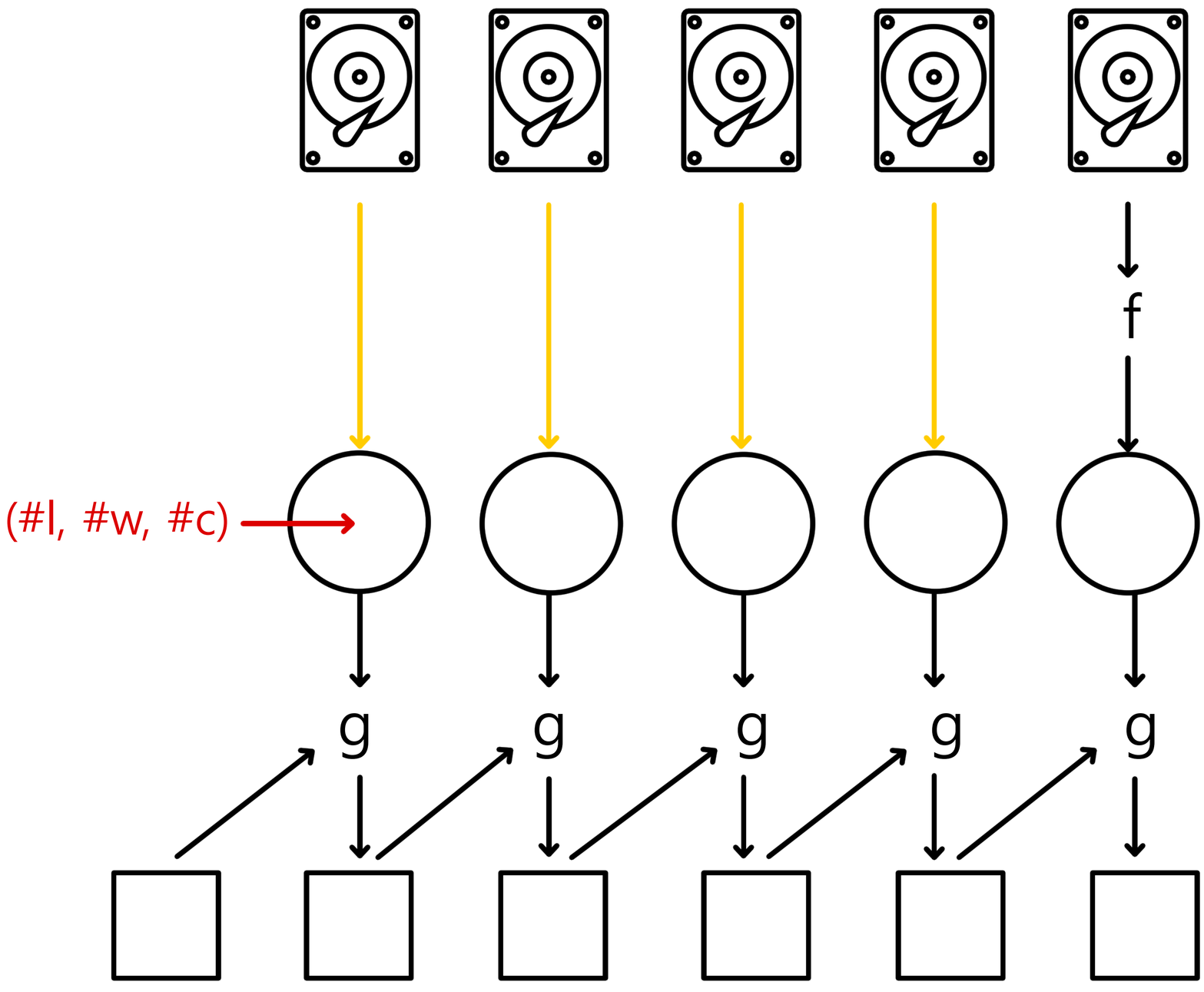
```
hdfs dfs -text distributed_A.txt | head
```

Distributed Shell: **wc 명령 수행**

```
$ wc <file>

$ wc A.txt

 269  4319  28001 A.txt
Lines words bytes
```



map = wc

reduce = operator.add for tuples

Word Count

Apache Hadoop (/hə`du:p/) is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware.

All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework...

```
'the': 3, 'of': 3, 'hadoop': 2, …
```

Word Count

uniq : 같은 문장끼리 인접해 있어야 count를 할 수 있다.
tr A B : A를 B로 변경 (translate)

```
one computer: cat * | tr ' ' '\n' | sort | uniq -c
```

(공백을 \n으로 변경)

apache hadoop
apache hadoop

apache
hadoop
apache
hadoop

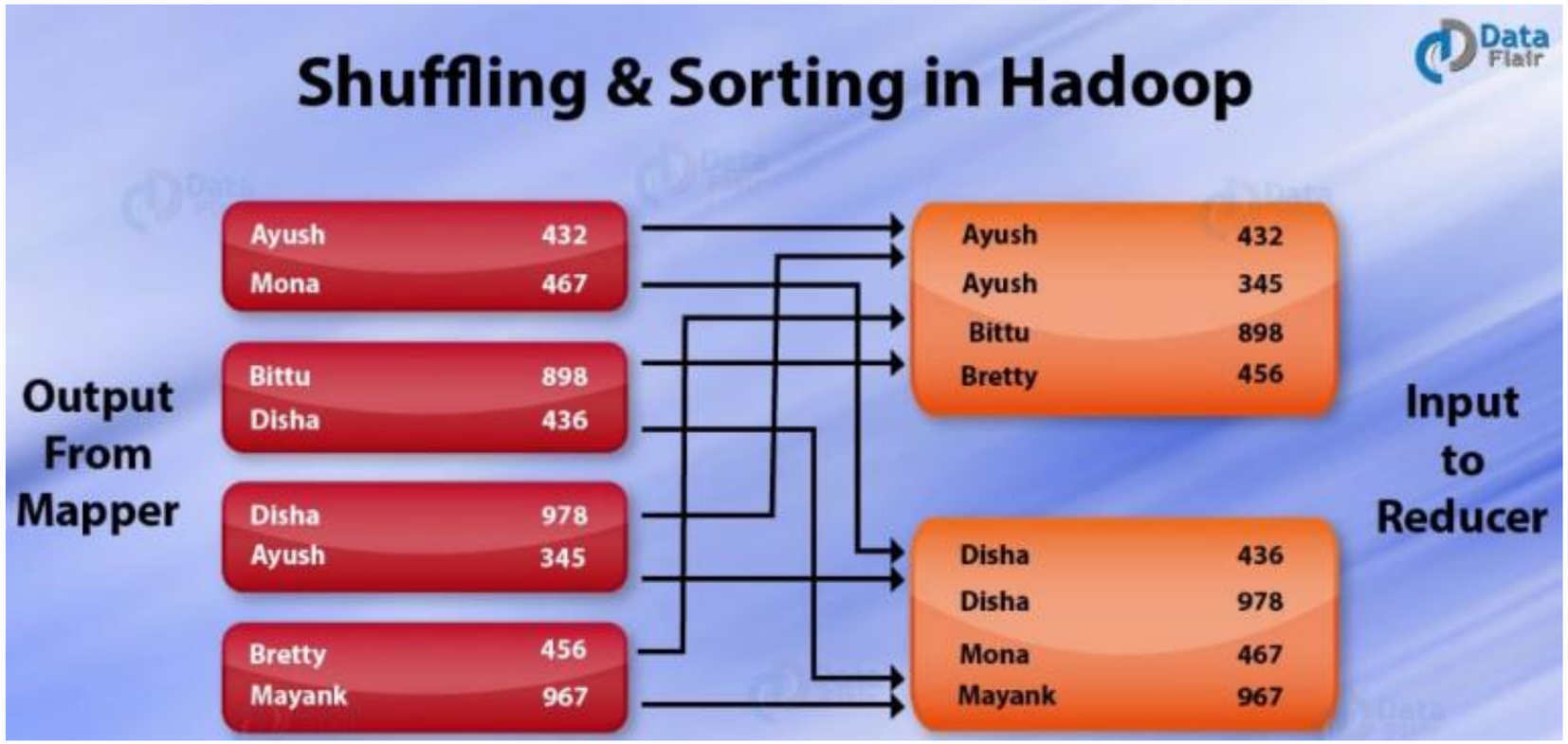
apache
apache
hadoop
hadoop

apache 2
Hadoop 2

```
distributed: cat * | tr ' ' '\n' | sort | uniq -c
```

Sort 명령어는 map 명령 수행 불가능 (모든 단어가 한 node에 있어야 가능) / reduce 명령 수행 불가능 (메모리 제한)

Map → Shuffle & Sort → Reduce



참고 출처 : https://data-flair.training/blogs/shuffling-and-sorting-in-hadoop/
https://techvidvan.com/tutorials/hadoop-reducer/

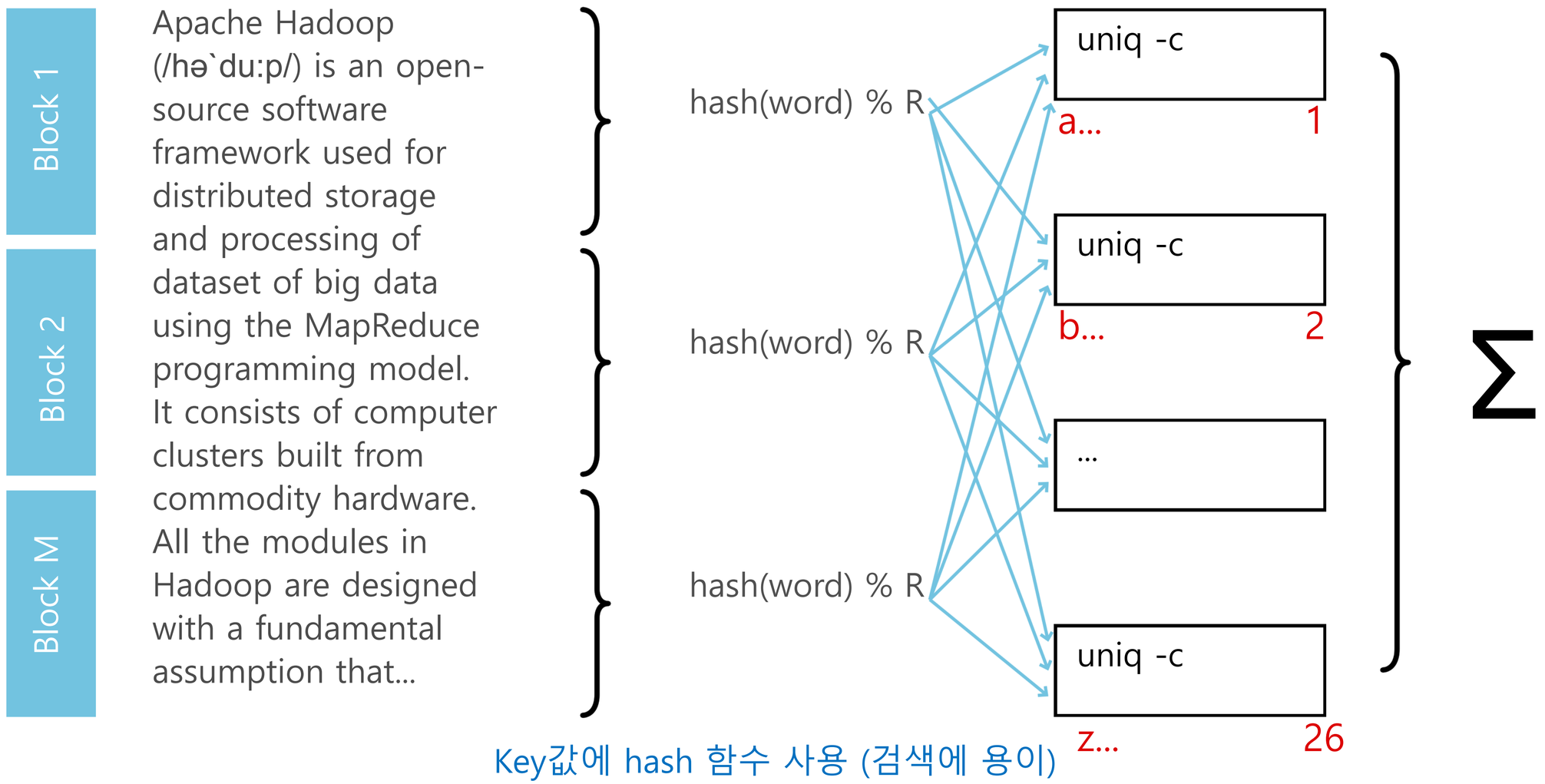
Shuffle : map 결과의 key들은 mapreduce framework에 의해 정렬된다.
정렬된 key값을 기준으로 데이터가 보내질 reducer 위치가 결정되고 전송하는 과정

Sort (by key) : reducer가 여러 mapper에서 온 data를 다시 key-value pair로 정렬

e.g.) (a, 1), (b,2) (a, 2) (c,3) (b, 4)=> (a, [1,2]) (b, [2,4]) (c,3)

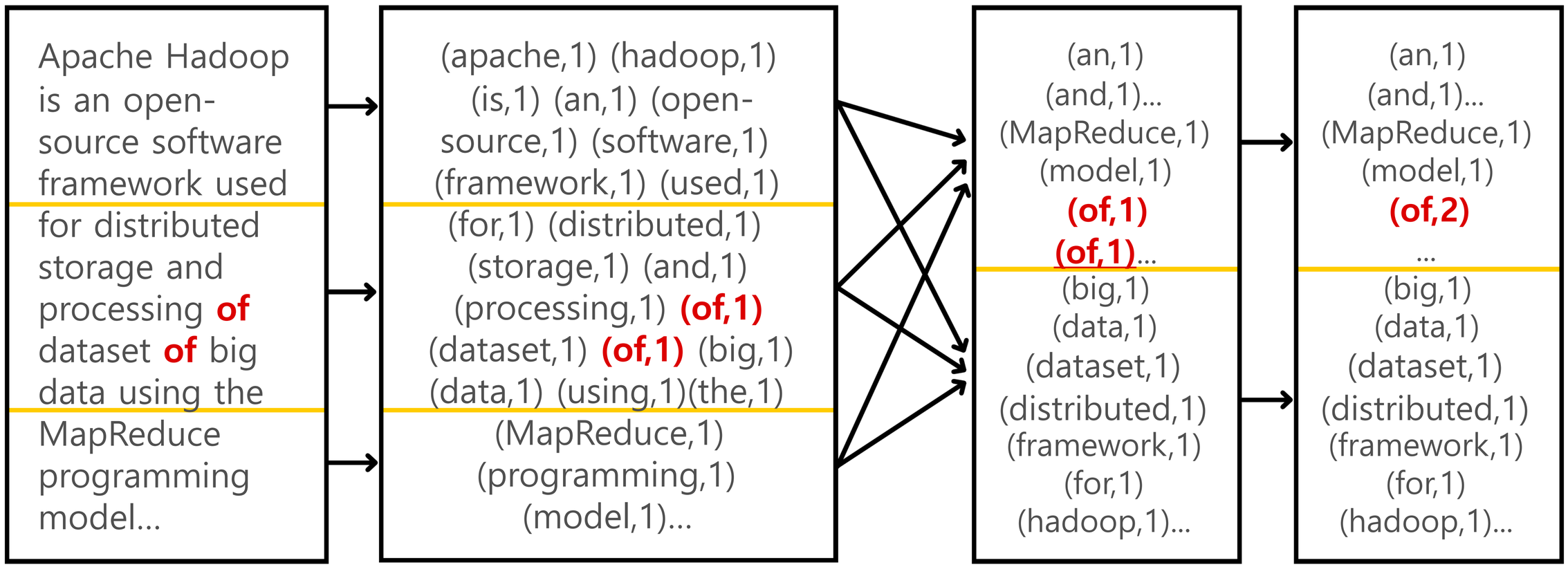
MapReduce (example → WordCount)

wikipedia.dump | tr ' ' '₩n' | sort | uniq -c

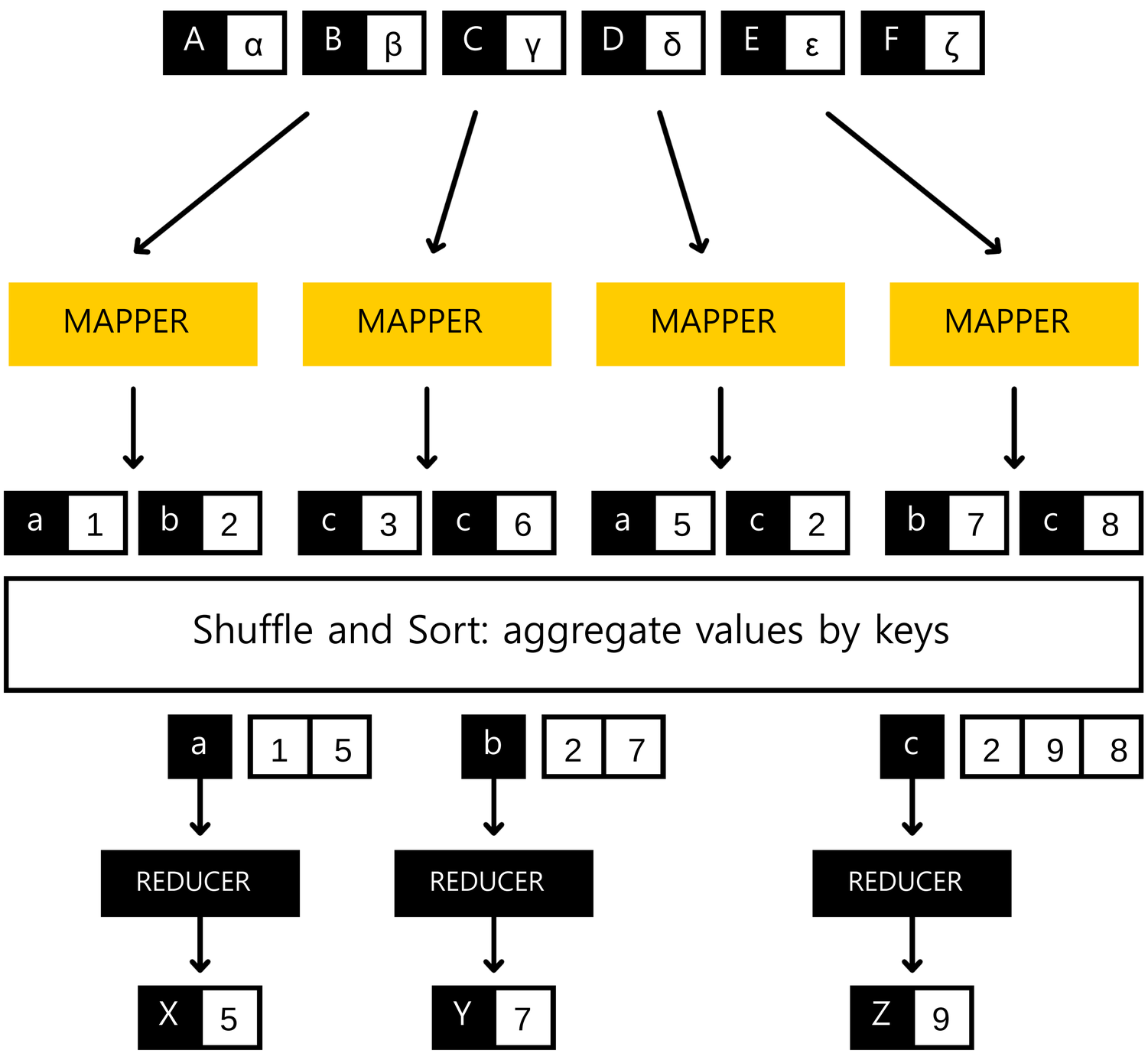


map: (key, value) → (key, value)
reduce: (key, value) → (key, value)

Word Count


Apache Hadoop is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model…
(apache,1) (hadoop,1) (is,1) (an,1) (open-source,1) (software,1) (framework,1) (used,1) (for,1) (distributed,1) (storage,1) (and,1) (processing,1) (of,1) (dataset,1) (of,1) (big,1) (data,1) (using,1)(the,1) (MapReduce,1) (programming,1) (model,1)…
(an,1) (and,1)… (MapReduce,1) (model,1) (of,1) (of,1)… (big,1) (data,1) (dataset,1) (distributed,1) (framework,1) (for,1) (hadoop,1)…
(an,1) (and,1)… (MapReduce,1) (model,1) (of,2) … (big,1) (data,1) (dataset,1) (distributed,1) (framework,1) (for,1) (hadoop,1)…
Text (Big Data)
Map
Shuffle & Sort
Reduce

MapReduce



> read: [(k_in, v_in), …]

> map: (k_in, v_in) —> [(k_interm, v_interm), …]

Intermediate data : output of map & sorted (on disk)

> Shuffle & Sort: sort and group by k_interm

여러 mapper에서 온 data를 재정렬

> reduce: (k_interm, [(v_interm, …)]) —> [(k_out, v_out), …]

MapReduce

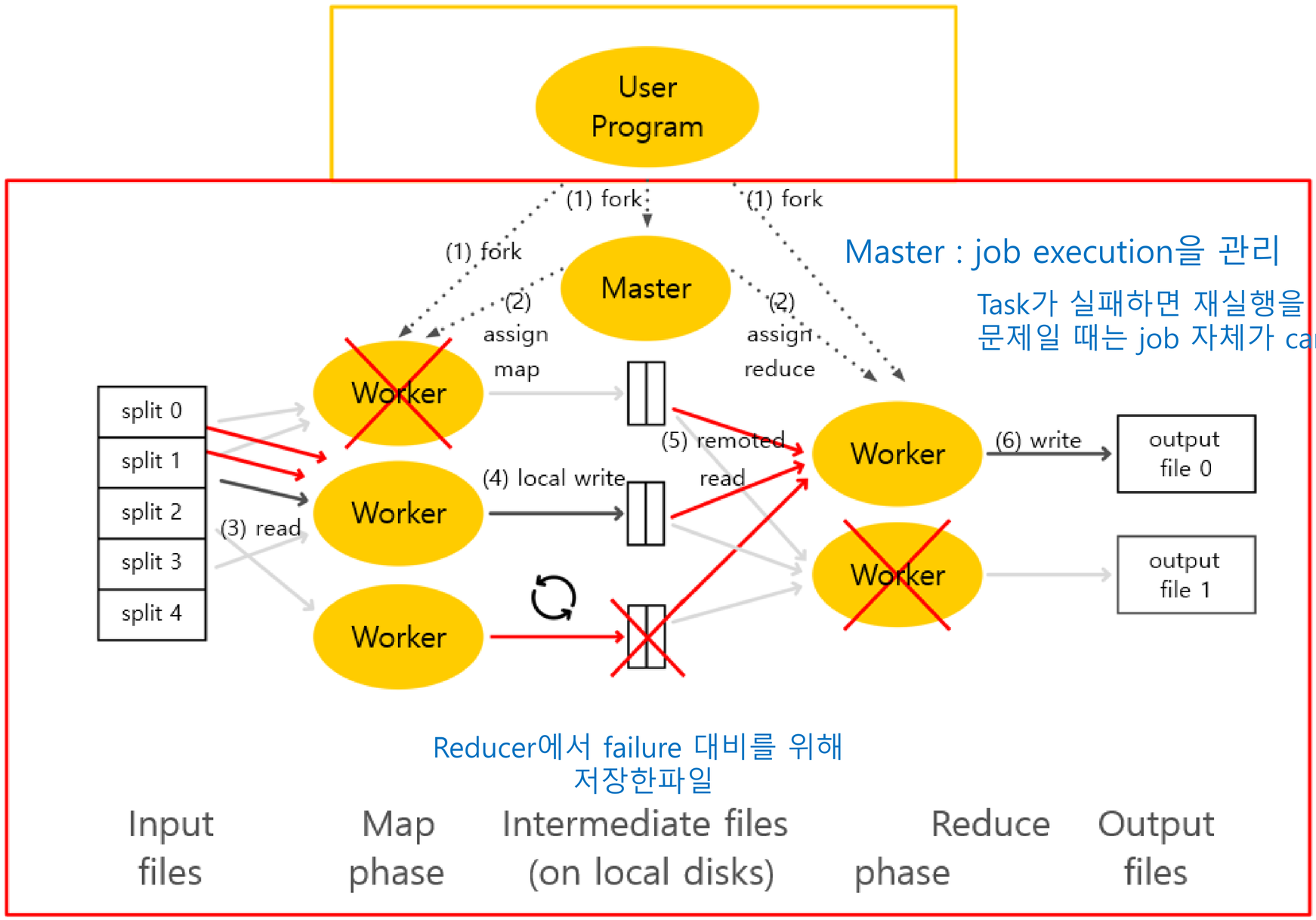
- You know the phases of MapReduce: Map, Shuffle & Sort, Reduce;
- You know how to solve simple tasks such as distributed “grep”, “head”, “wc” and “Word Count” with MapReduce.

MapReduce

Fault Tolerance

MapReduce Framework는 job execution동안 node failure에 robust해야한다.

Original MapReduce 그림 – 하둡과는 약간 다르지만 기본 원리는 같다.



Master : job execution을 관리

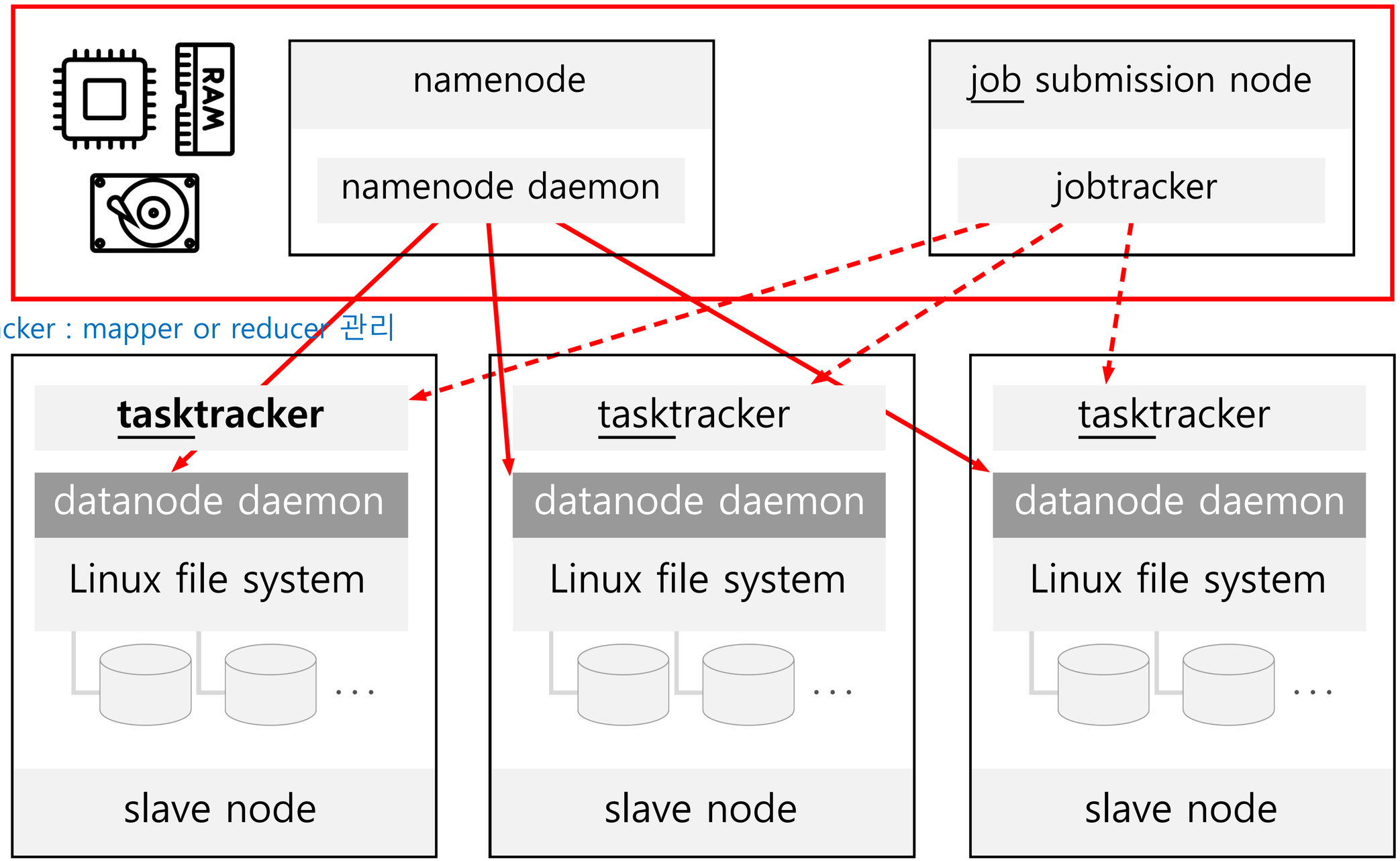
Task가 실패하면 재실행을 명령한다. 그러나, Master가 문제일 때는 job 자체가 cancel된다.

Map or Reduce function은 functional programming 방식(실행의 결과는 항상 같아야 한다. / deterministic) => re-execution을 위한 대비

Hadoop MRv1 MapReduce version1

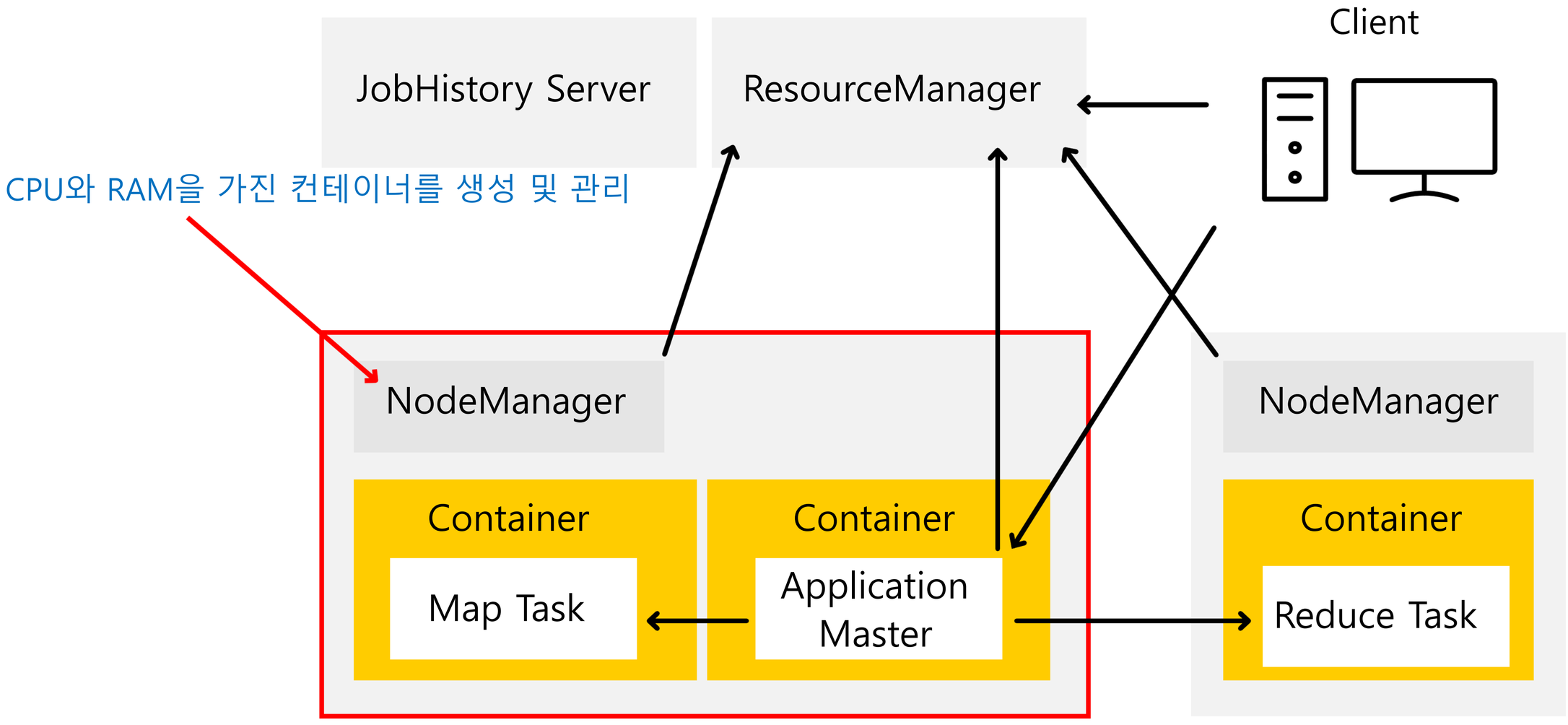
하나의 global jobtracker가 master 역할을 수행한다.

Global jobtracker에서 문제가 생기면 job이 cancel된다.



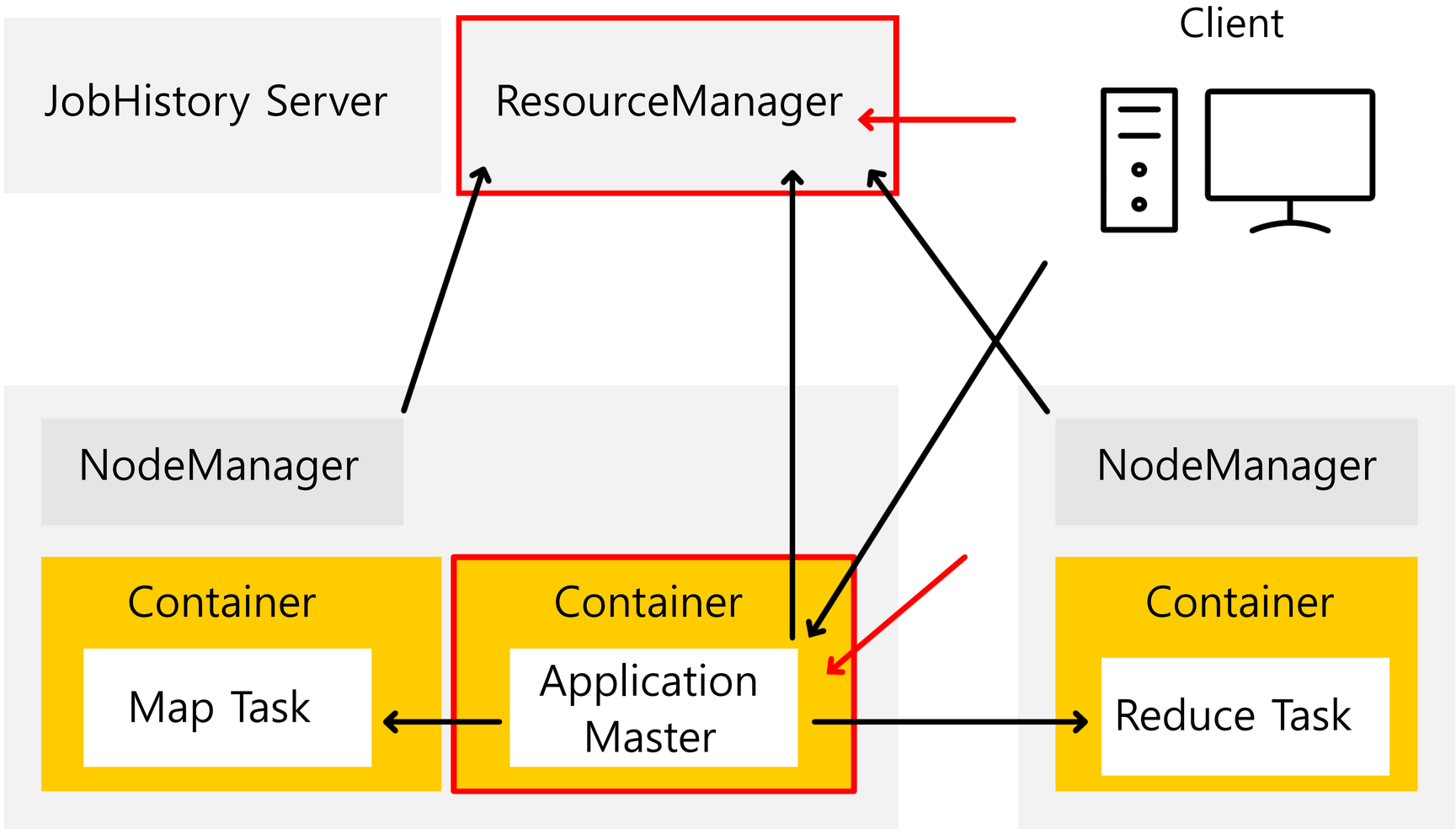
MRv1 vs YARN (Yet Another Resource Negotiation)

ResourceManager: cluster resources and processes requests from Node Managers



Tasktracker 대신 nodemanager가 생김

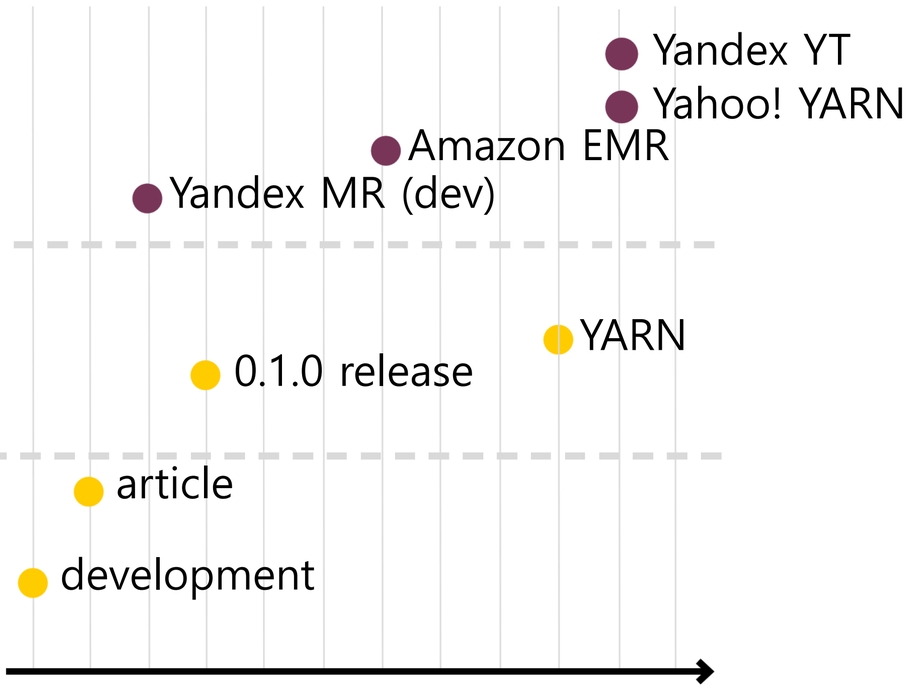
MRv1 vs YARN



Jobtracker 대신 application master는 어느 node에서든 실행이 가능하다.

Application Master : A service to run and monitor containers for application-specific processes on cluster nodes.

MapReduce Frameworks: History Timeline



- [2003] Google MapReduce (development)
- [2004] Google MapReduce (article)
- [2005] Yandex MapReduce (development)
- [2006] Hadoop 0.1.0 release
- [2009] Amazon EMR (Hadoop inside)
- [2012] MapReduce —> YARN
- [2013] Yahoo! YARN deployed in production
- [2013] Yandex YT
- …
- MapReduce in MongoDB, Riak, …

MapReduce Framework

- You can explain what will happen if Mapper or Reducer dies;
- You know what JobTracker and TaskTracer in MRv1, ResourceManager and NodeManager in YARN are.

BigDATAteam