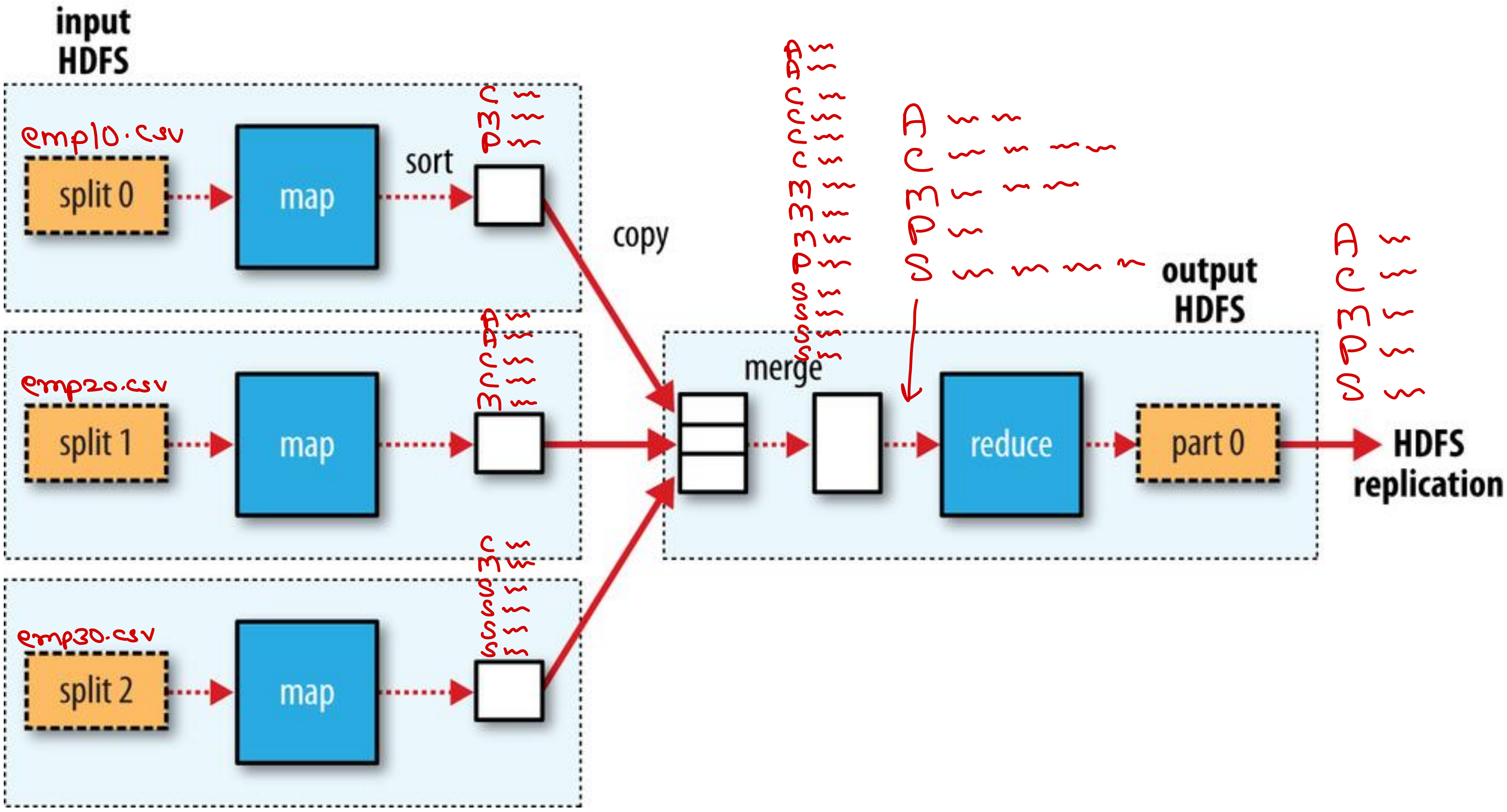
Big Data – Hadoop

Trainer: Mr. Nilesh Ghule.

Data flow of MR job (Single reducer) Job wise total salary

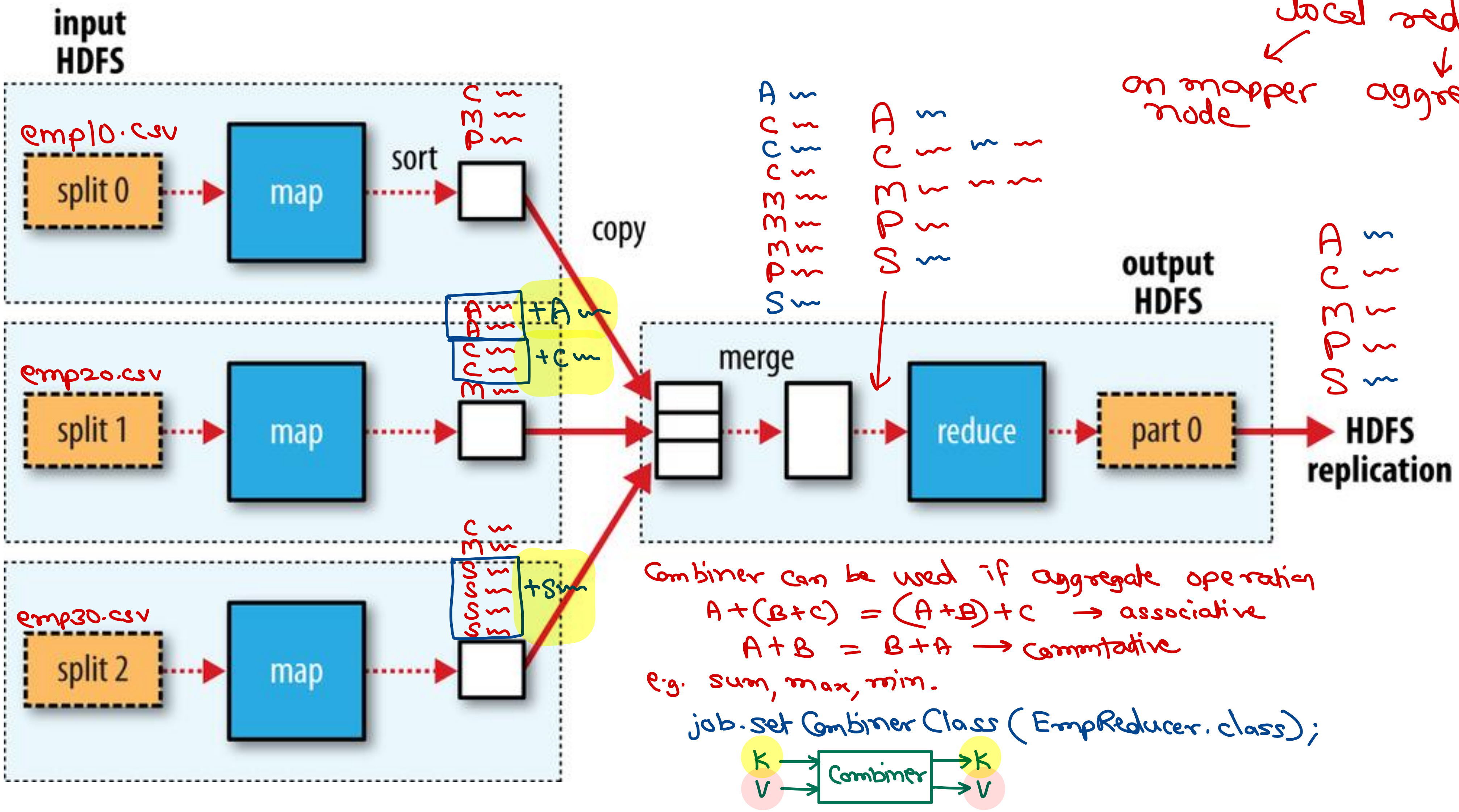


Data flow of MR job (Single reducer)   Job wise total salary using Combiner

local reducer

on mapper node

aggregation



Combiner can be used if aggregate operation

A+(B+C) = (A+B)+C → associative

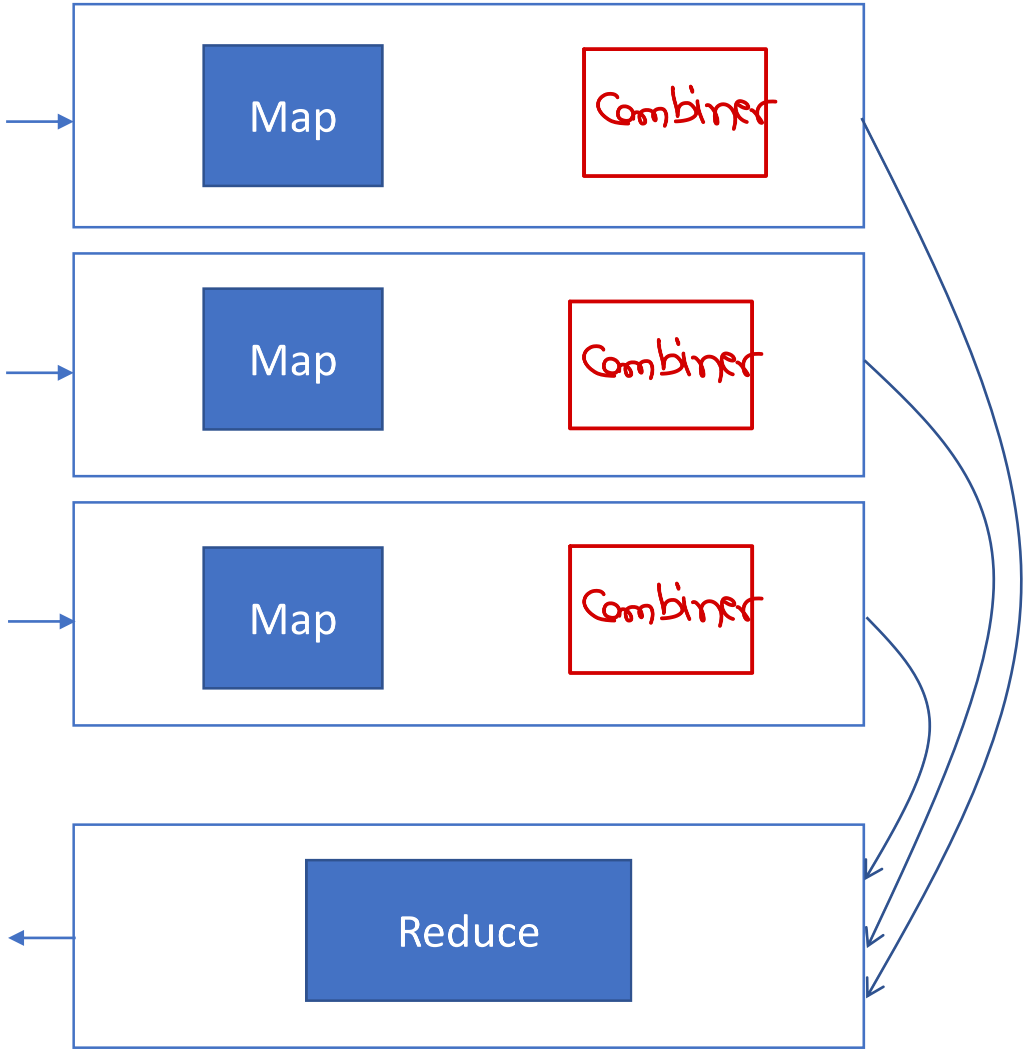
A+B = B+A → commutative

e.g. sum, max, min.

job.setCombinerClass(EmpReducer.class);

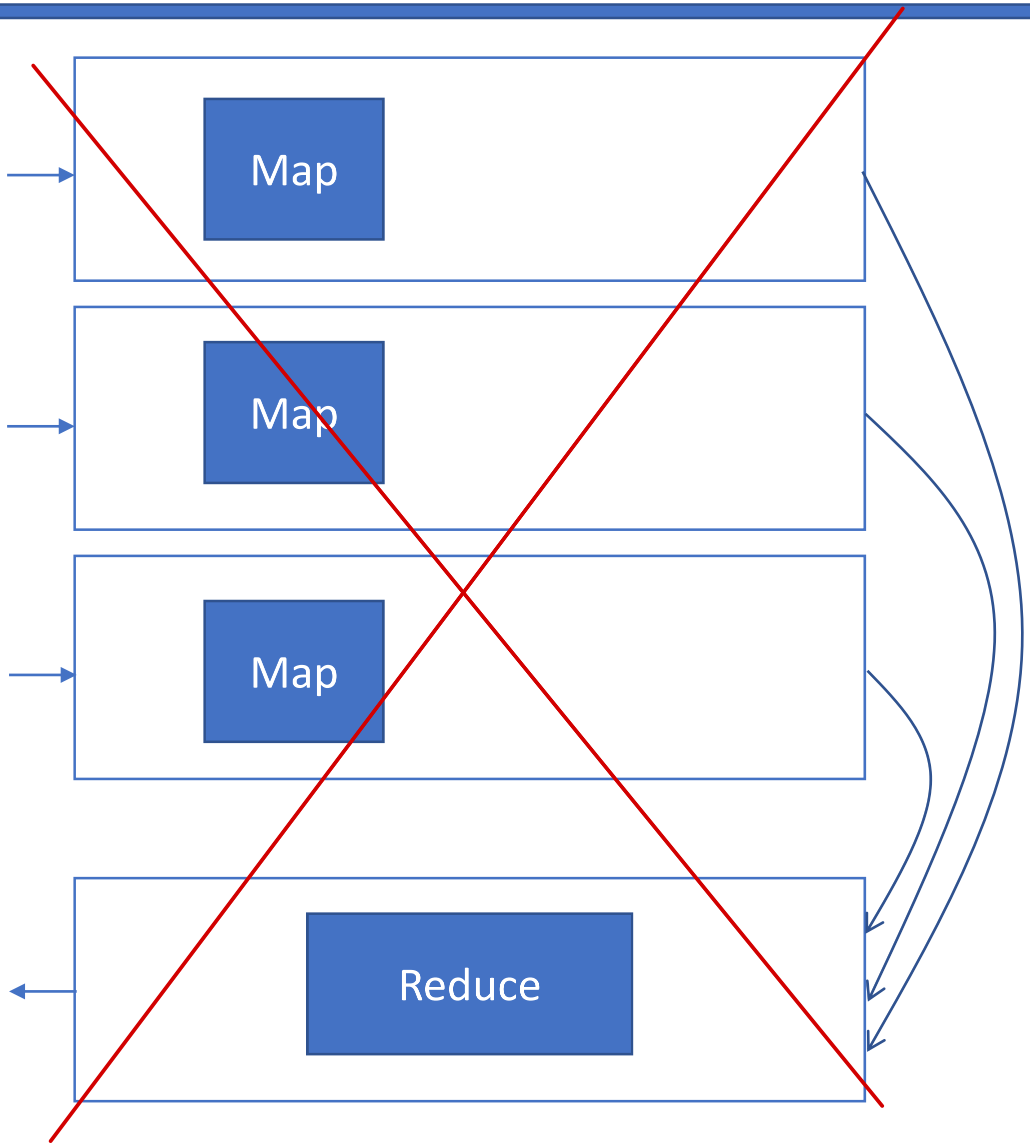
Combiner

- Combiner is a local reducer i.e. runs reducer (aggregation logic) within mapper task process.
- Minimize output for mapper task
  - Less merge & shuffle
  - Less network transfer
  - Less aggregation in reducer
- Combiner is optional.
- Works only for commutative & associative aggregate functions only.
  - A + B = B + A
  - A + (B + C) = (A + B) + C



Partitioner

- By default MR job have single reducer.
- Having huge data for aggregation may lead to out of memory error.
- Number of reducers can be configured in job configuration file or in driver code.
  - job.setNumReduceTasks(2);
  - mapreduce.job.reduces = 2
- Number of partitions = Number of reducers
- Output of mapper is divided into multiple partitions based produced key
- By default HashPartitioner is used, that distributes mapper output in number of partitions uniformly.

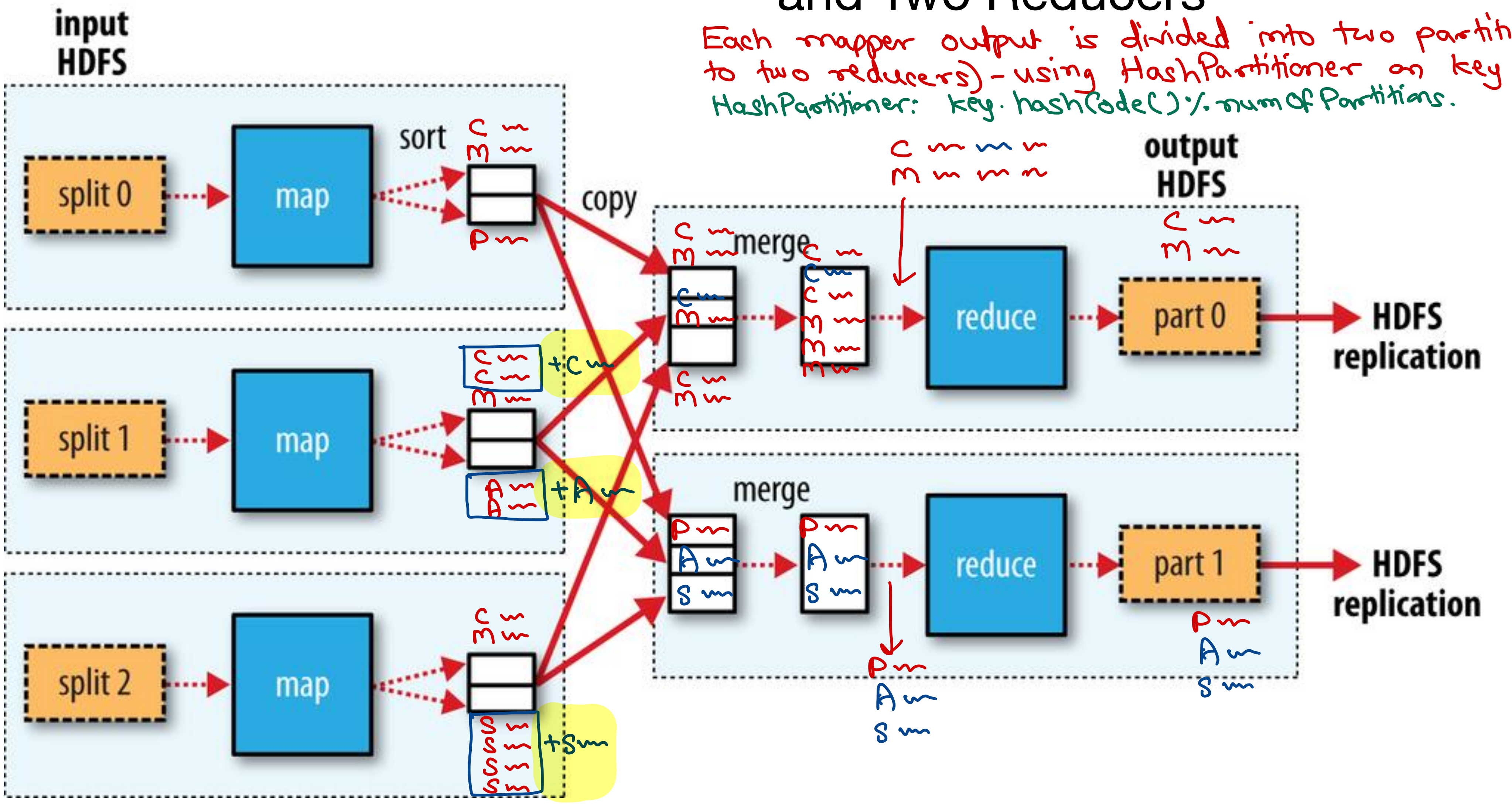


Data flow of MR job (Multiple reducers)

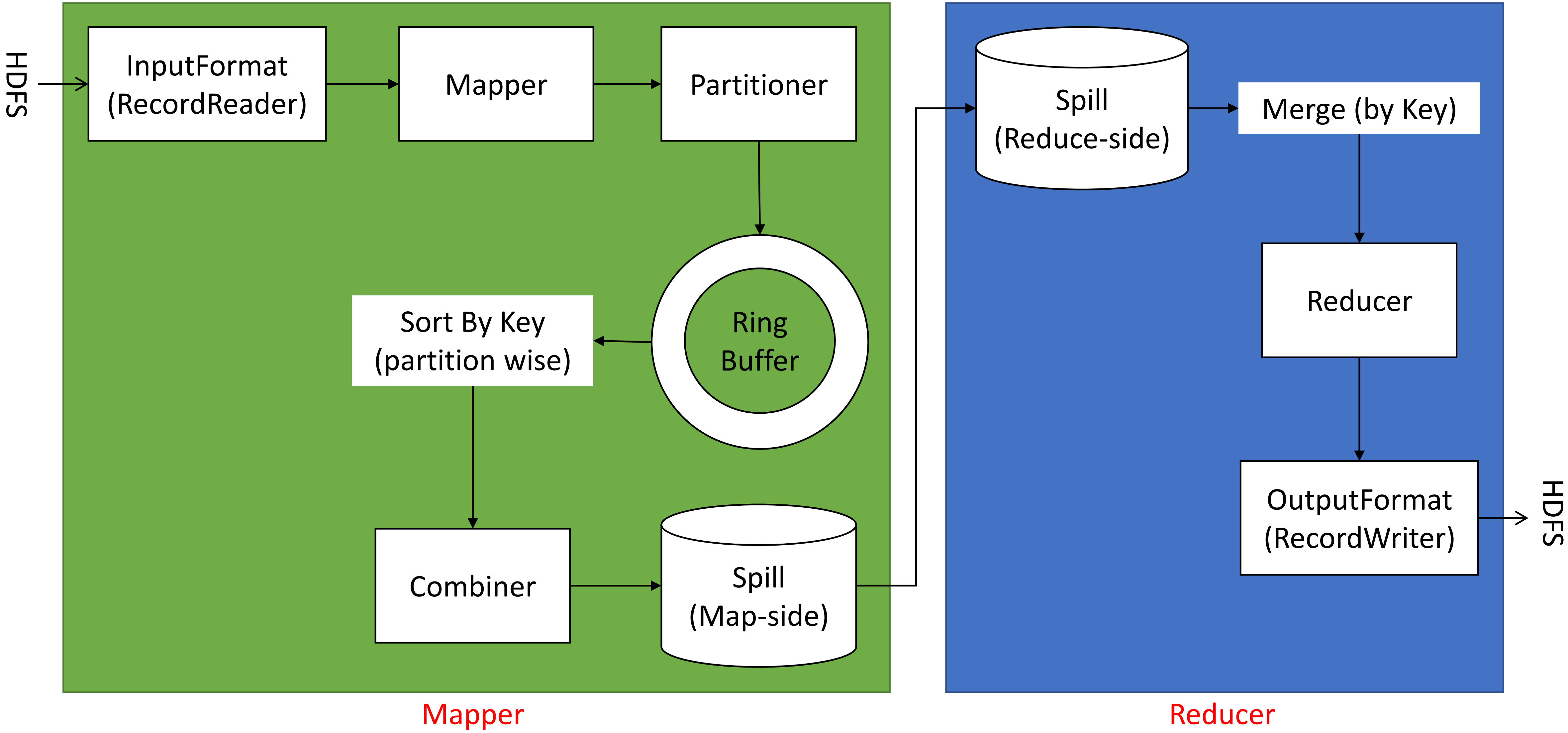
Job wise total salary using Combiner and Two Reducers

Each mapper output is divided into two partitions (due to two reducers) - using HashPartitioner on key.

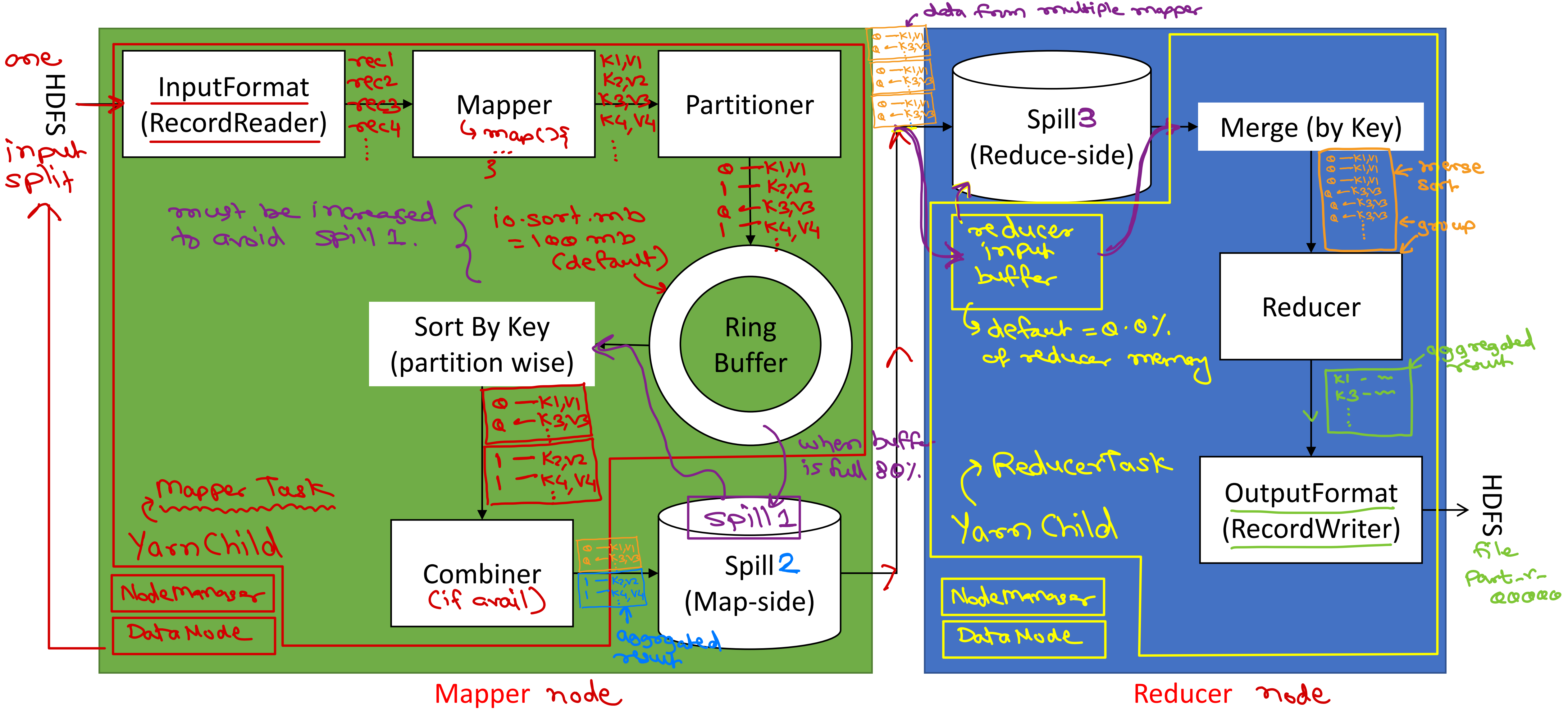
HashPartitioner: key.hashCode() % numOfPartitions.



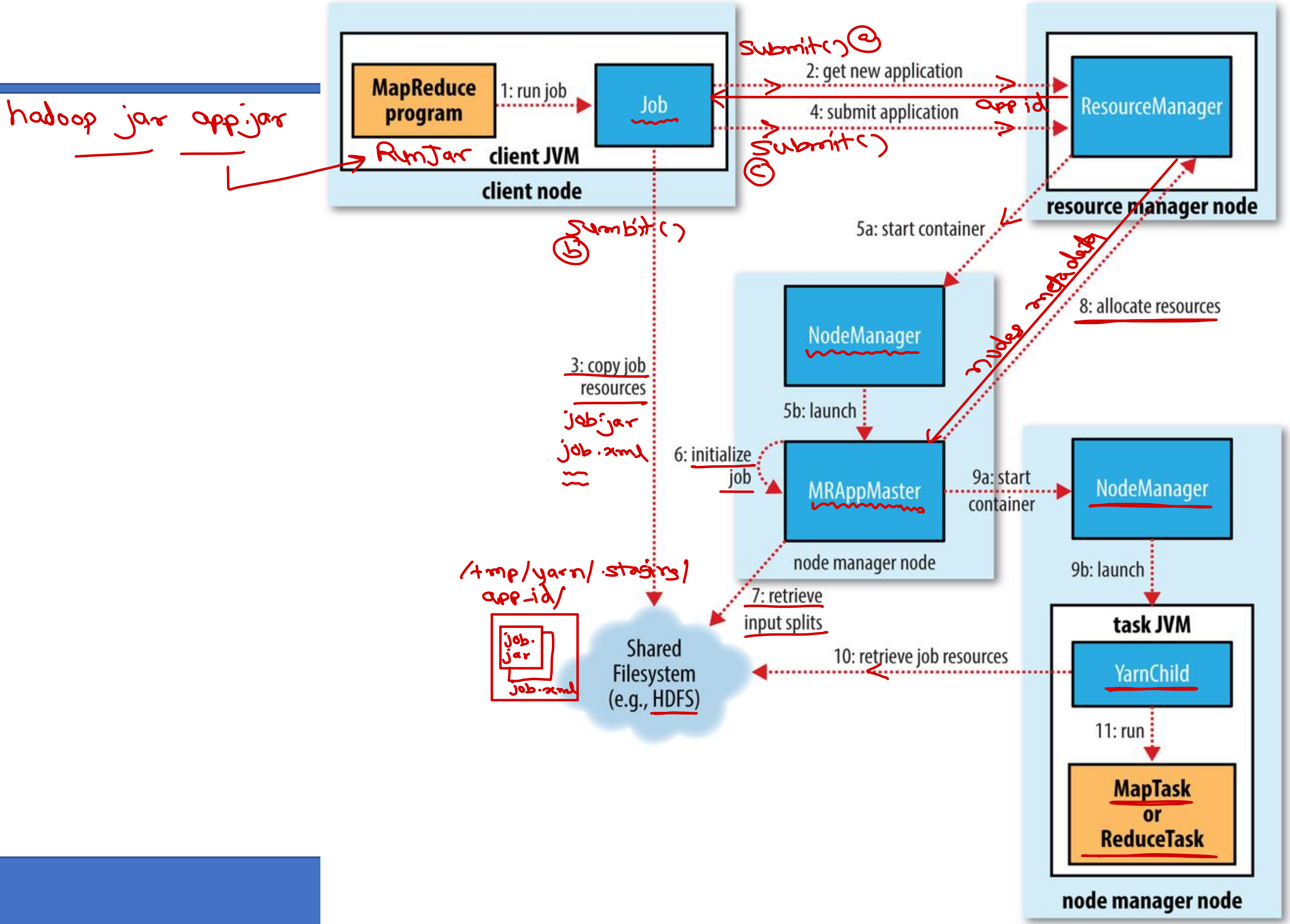
Hadoop MR data flow (detailed)



Hadoop MR data flow (detailed)



MR on YARN


hadoop jar app.jar
RunJar
MapReduce program
1: run job
Job
client JVM
client node
submit() ⓐ
2: get new application
app id
4: submit application
submit() ⓒ
ResourceManager
resource manager node
submit() ⓑ
5a: start container
8: allocate resources
nodes metadata
NodeManager
5b: launch
6: initialize job
MRAppMaster
node manager node
9a: start container
NodeManager
9b: launch
3: copy job resources
job.jar
job.xml
/tmp/yarn/staging/app_id/
job.jar
job.xml
Shared Filesystem (e.g., HDFS)
7: retrieve input splits
10: retrieve job resources
task JVM
YarnChild
11: run
MapTask or ReduceTask
node manager node

Hadoop Streaming

```
#!/usr/bin/python3
# mapper.py
import sys
for line in sys.stdin:
    words = line.split()
    for word in words:
        print(f"{word}\t1")
```

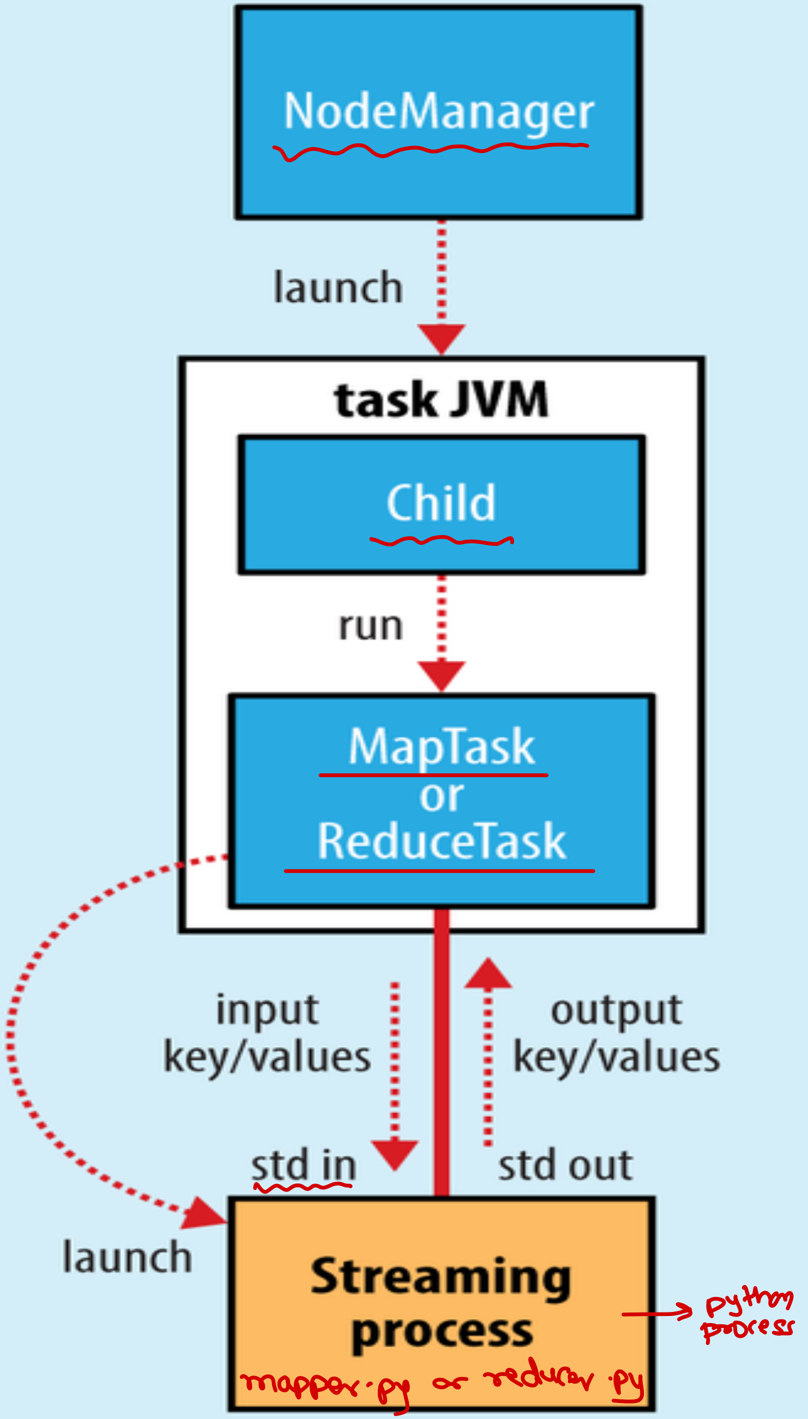
Stdout

```
#!/usr/bin/python3
# reducer.py
di = dict()
for line in sys.stdin:
    (word,cnt) = line.split()
    newcnt = di.get(word, 0) + int(cnt)
    di[word] = newcnt
for word,total in di.items():
    ~~print~~ print(f"{word}\t{total}")
```

Stdout

```
hadoop jar $HADOOP_HOME/share/…/hadoop-streaming-2.7.3.jar \
-files mapper.py,reducer.py  \
-input /user/nilesh/wc/input  \
-output /user/nilesh/wc/output \
-mapper mapper.py -reducer reducer.py
```

contains driver code & other helper code.


NodeManager
launch
task JVM
Child
run
MapTask
or
ReduceTask
input
key/values
output
key/values
std in
std out
launch
Streaming
process
mapper.py or reducer.py
python process