MapReduce

OutLine

- MapReduce
- Hadoop Streamin

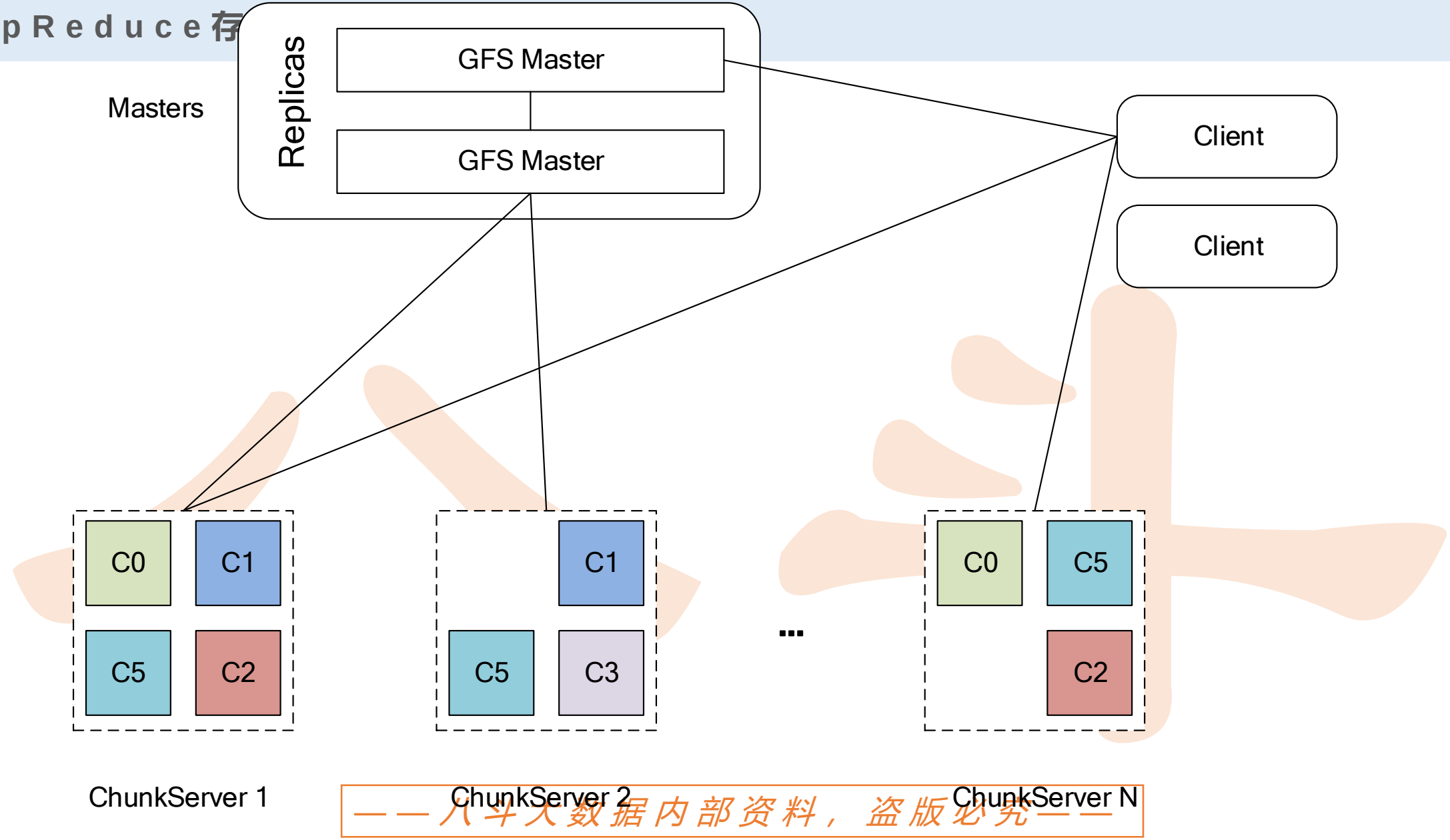
MapReduce简介

- MapReduce是一个用于处理海量数据的分布式计算框架。
- 这个框架解决了
  - 数据分布式存储
  - 作业调度、
  - 容错、
  - 机器间通信等复杂问题

MapReduce存储

- 为什么HDFS
  - 系统可靠性
  - 可扩展性
  - 并发处理

MapReduce存



MapReduce基础

- 一个简单的WordCount如何编程?
- 1000个词的如何做
  - So easy
- 1000G大小的时候如何做
  - 还行吧
- 1000G*1000G大小的时候如何做
  - 晕了
- 1000G*1000G*1000...如何办
  - 怒了

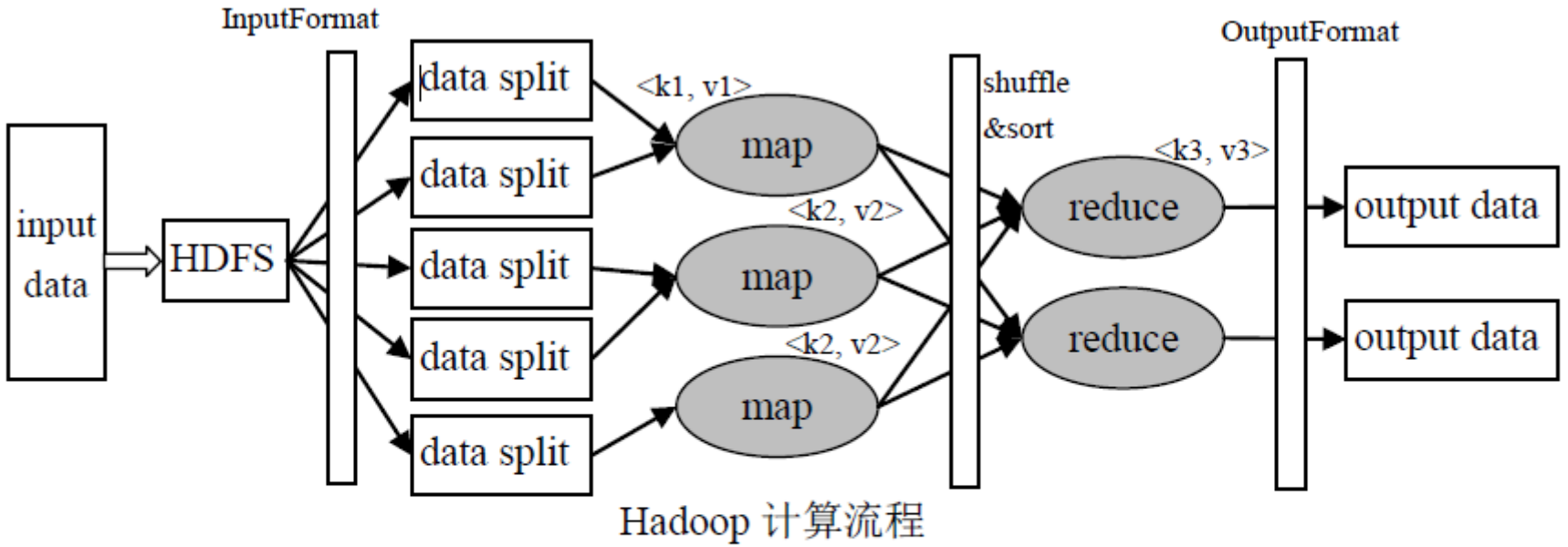
MapReduce分而治之思想

- 数钱实例：一堆钞票，各种面值分别是多少
  - 单点策略
    - 一个人数所有的钞票，数出各种面值有多少张
  - 分治策略
    - 每个人分得一堆钞票，数出各种面值有多少张
    - 汇总，每个人负责统计一种面值
- 解决数据可以切割进行计算的应用

MapReduce分而治之思想

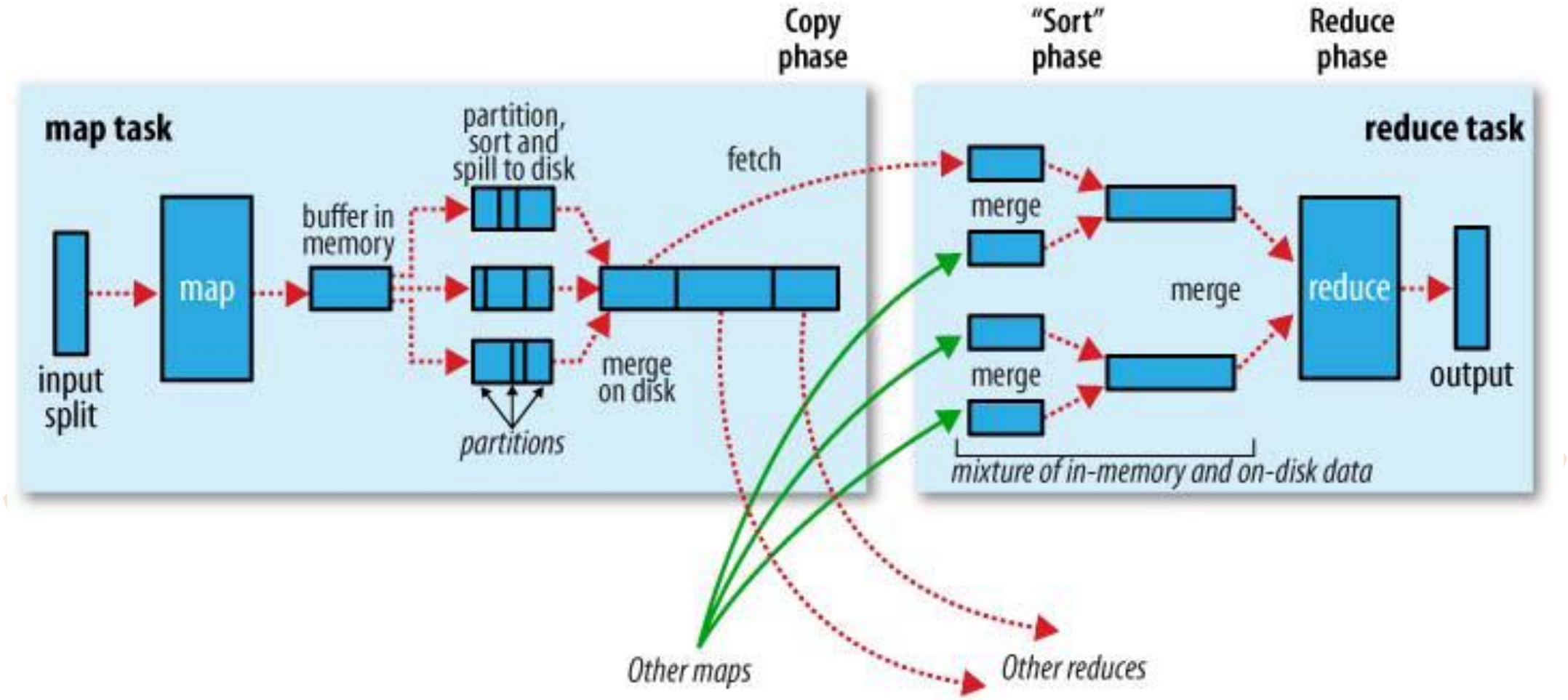
- 分治思想
  - 分解
  - 求解
  - 合并
- MapReduce映射
  - 分：map
    - 把复杂的问题分解为若干“简单的任务”
  - 合：reduce

MapReduce计算框架-执行流程

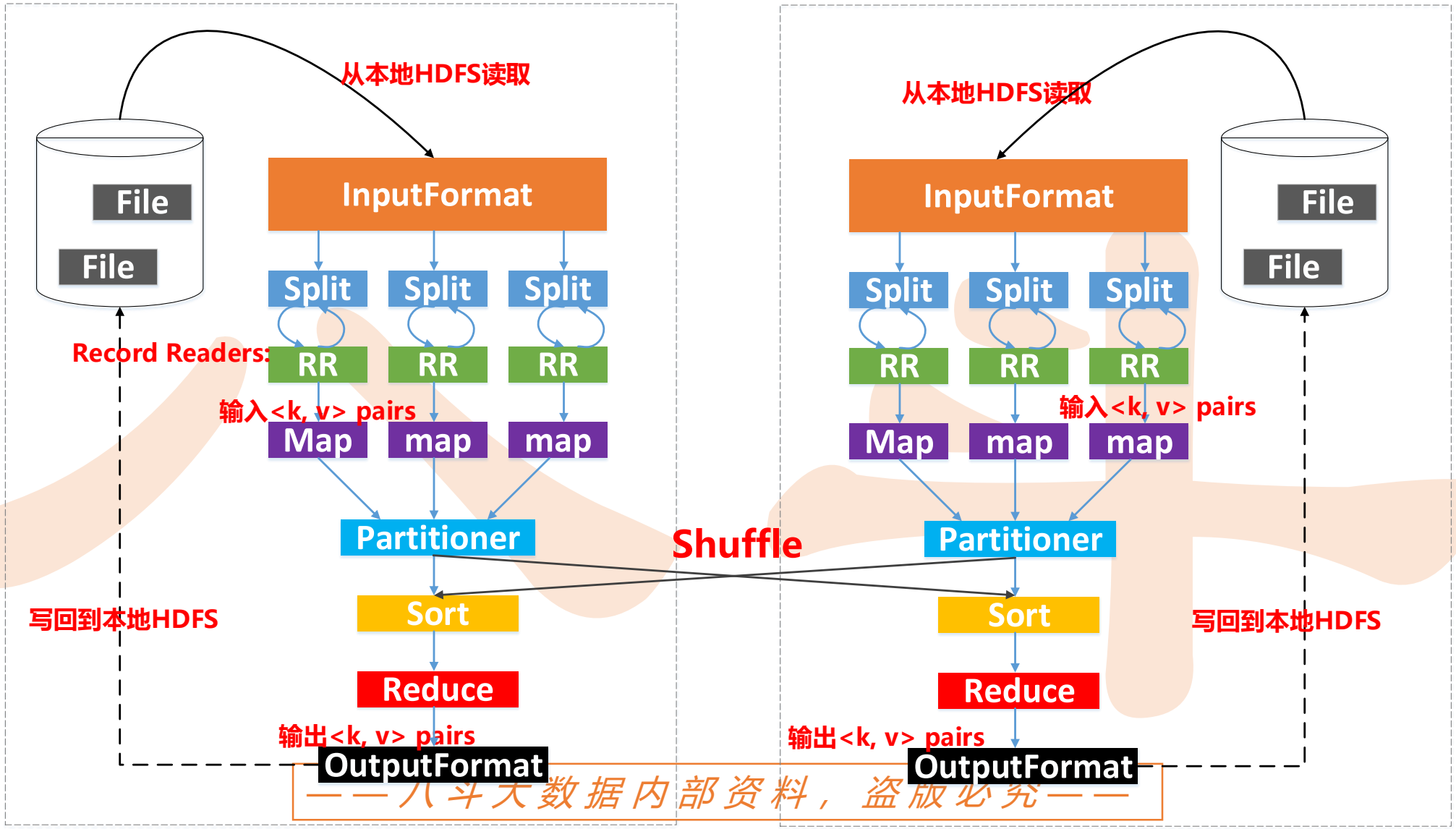


- 应用程序开发人员一般情况下需要关心的是图中灰色的部分!

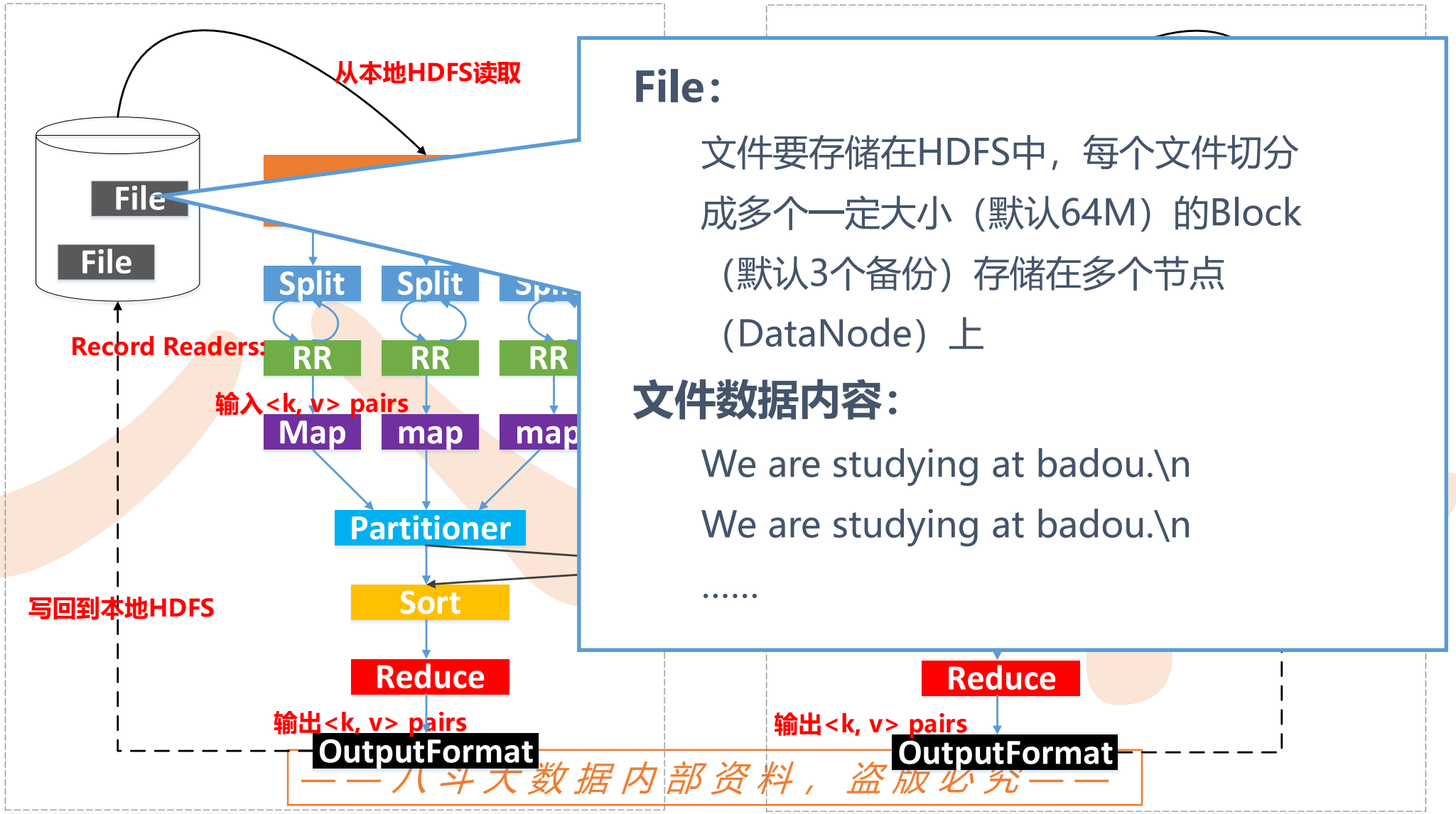
MapReduce计算框架-执行流程



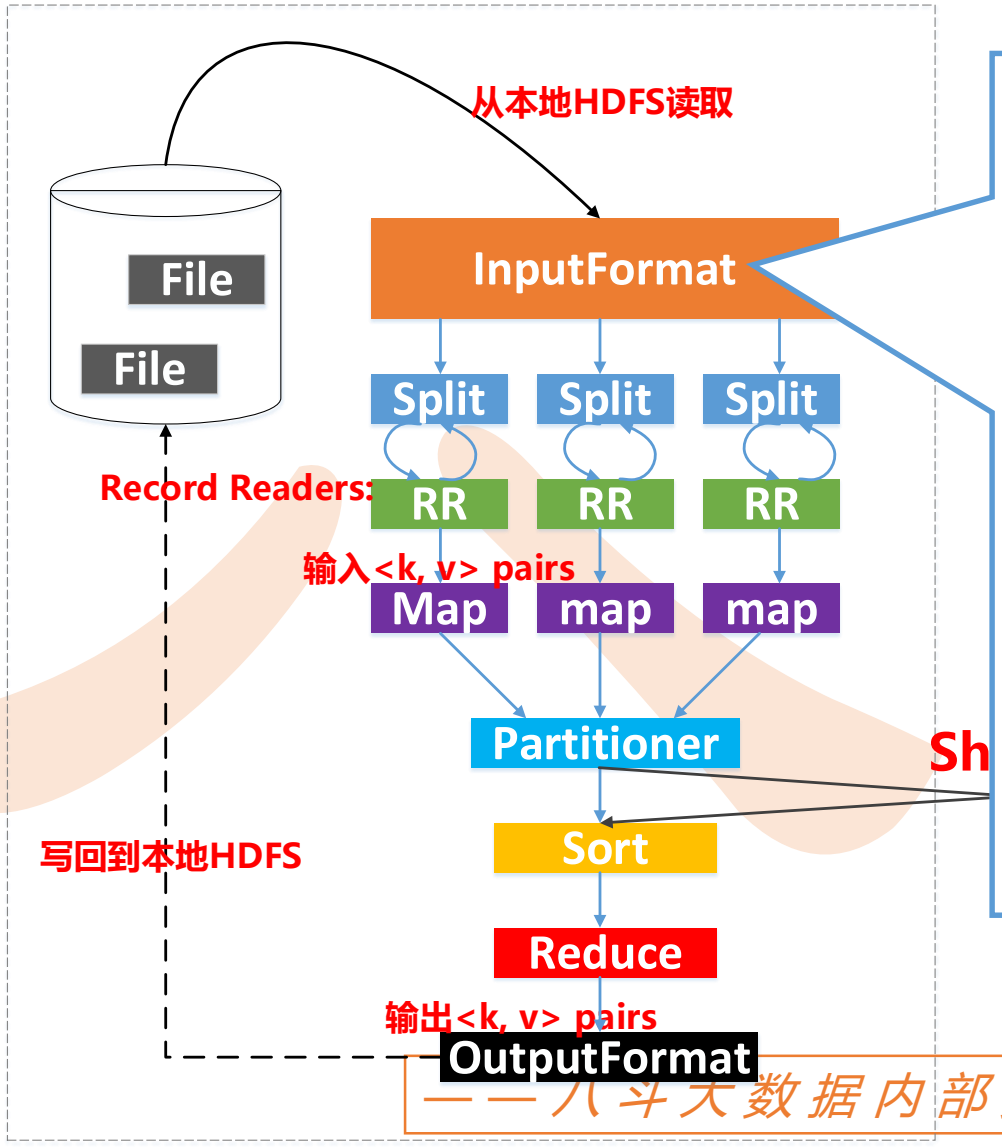
MapReduce计算框架-执行流程



MapReduce计算框架-执行流程



MapReduce计算框架-执行流程



InputFormat:

MR框架基础类之一

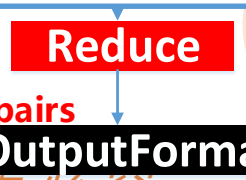
- **数据分割（Data Splits）**
- **记录读取器（Record Reader）**

例子:

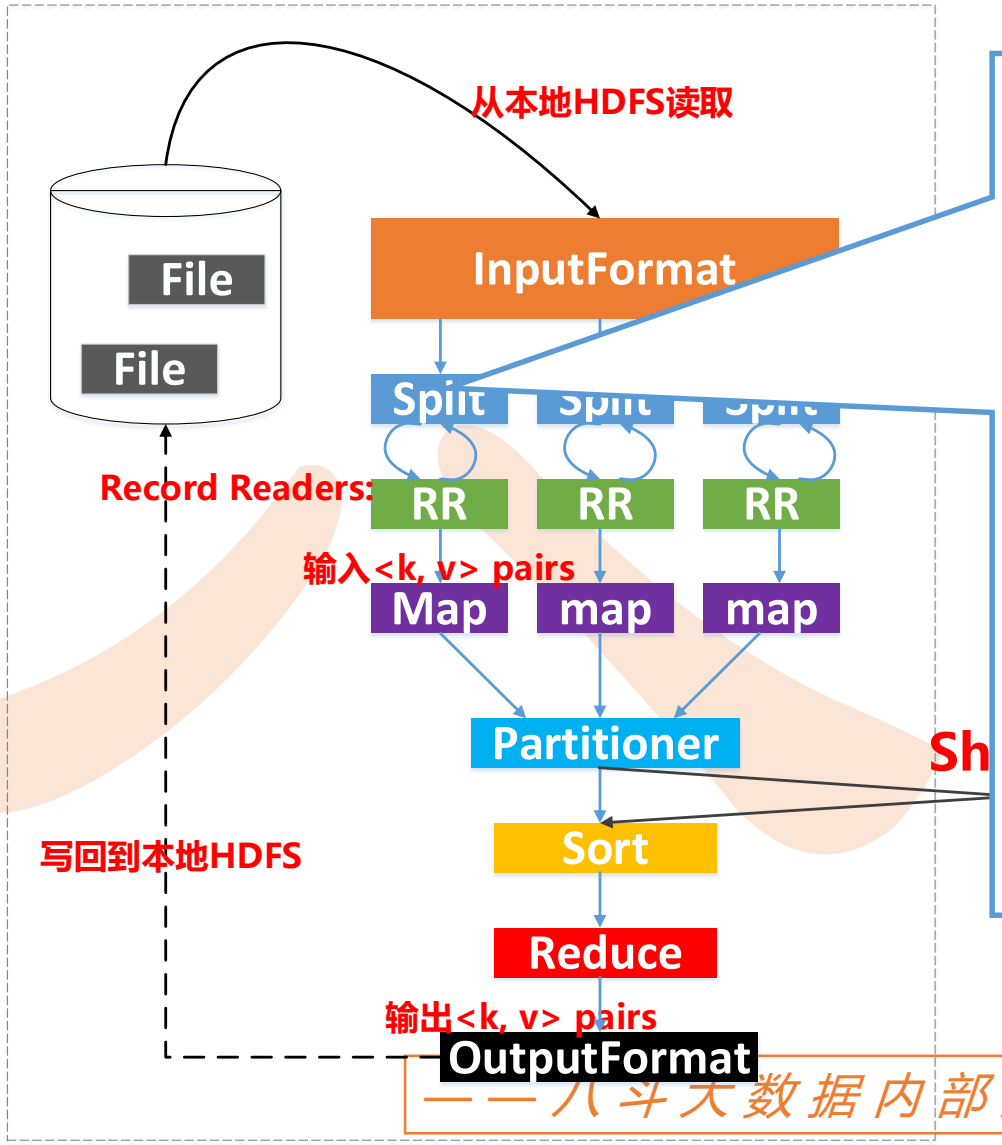
数据格定义，如果以“\n”分割每条记录，以空格区分一个目标单词

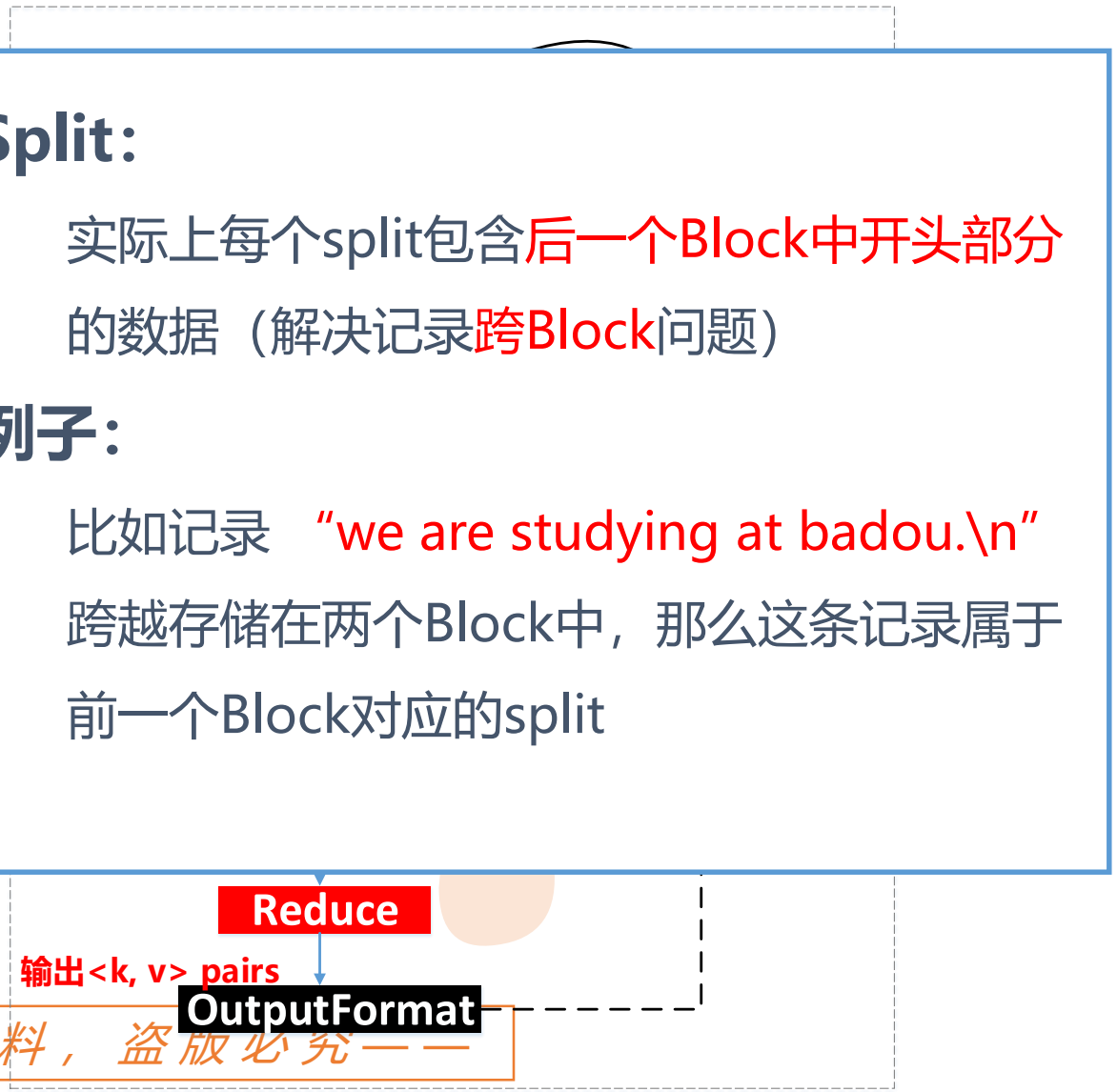
“we are studying at badou.”为一条记录

“are” “at”等为一个目标单词



MapReduce计算框架-执行流程





Split：

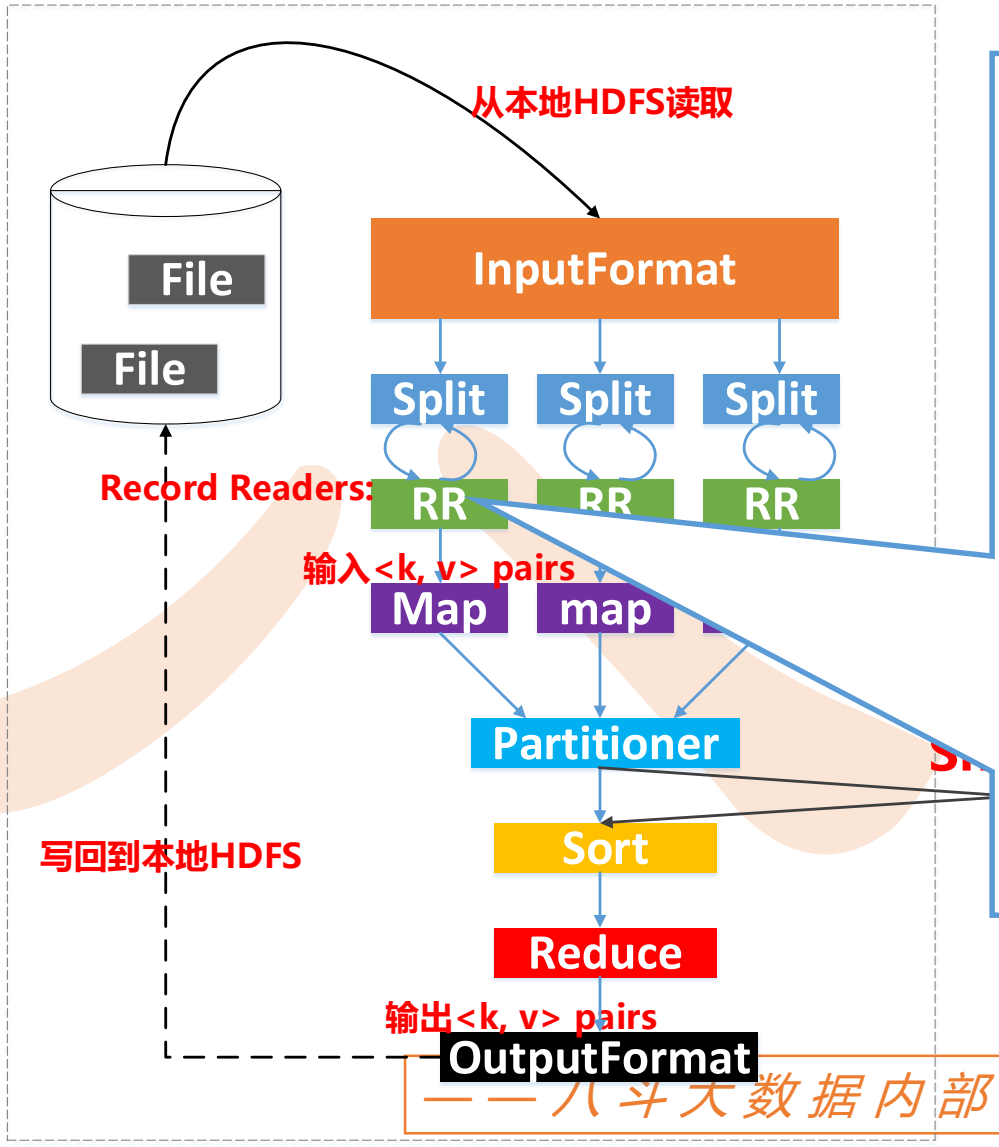
实际上每个split包含后一个Block中开头部分的数据（解决记录跨Block问题）

例子：

比如记录 “we are studying at badou.\n”

跨越存储在两个Block中，那么这条记录属于前一个Block对应的split

MapReduce计算框架-执行流程



RecordReader:

每读取一条记录，调用一次map函数

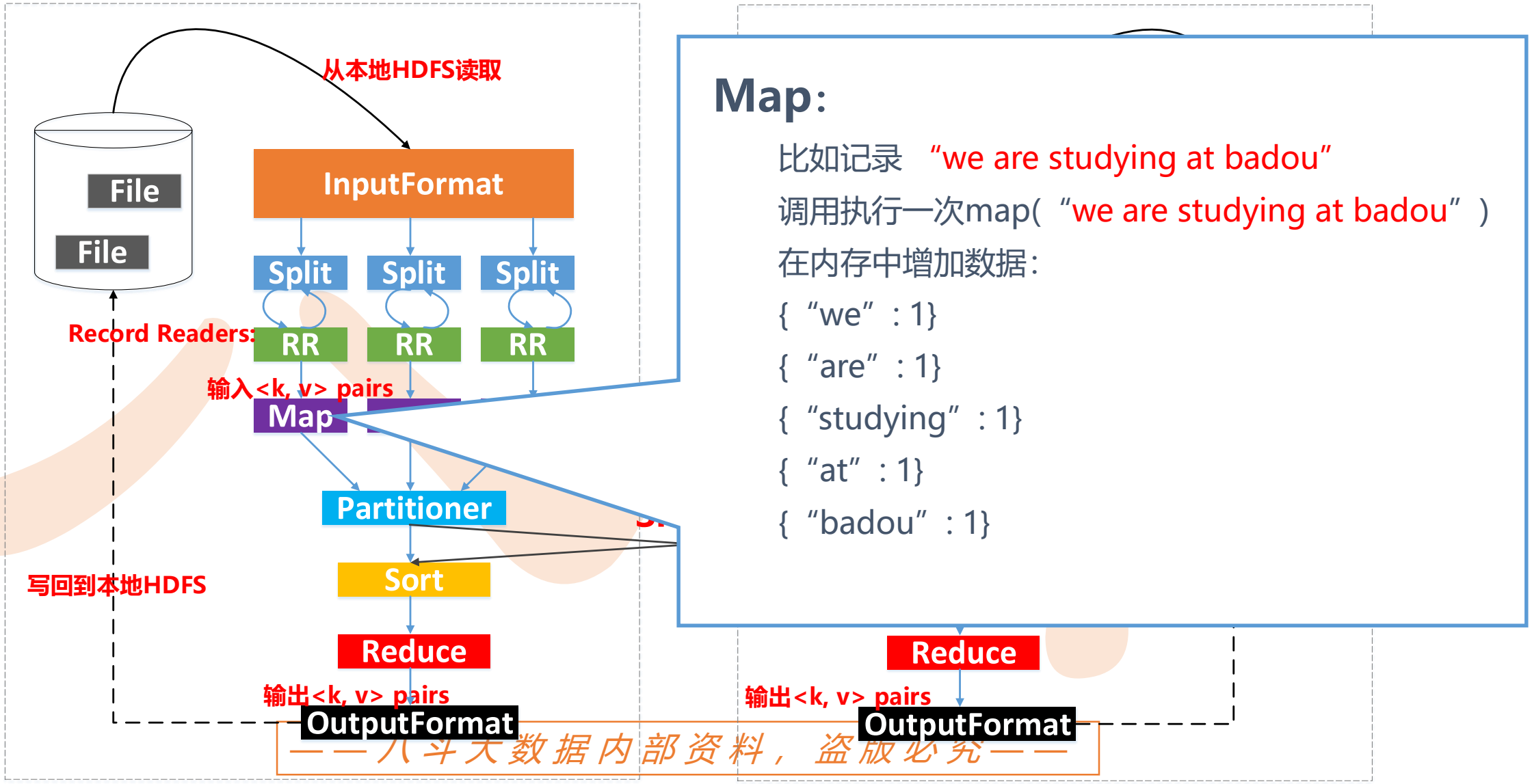
例子:

比如记录 “we are studying at badou.”作为参数v，调用map(v)

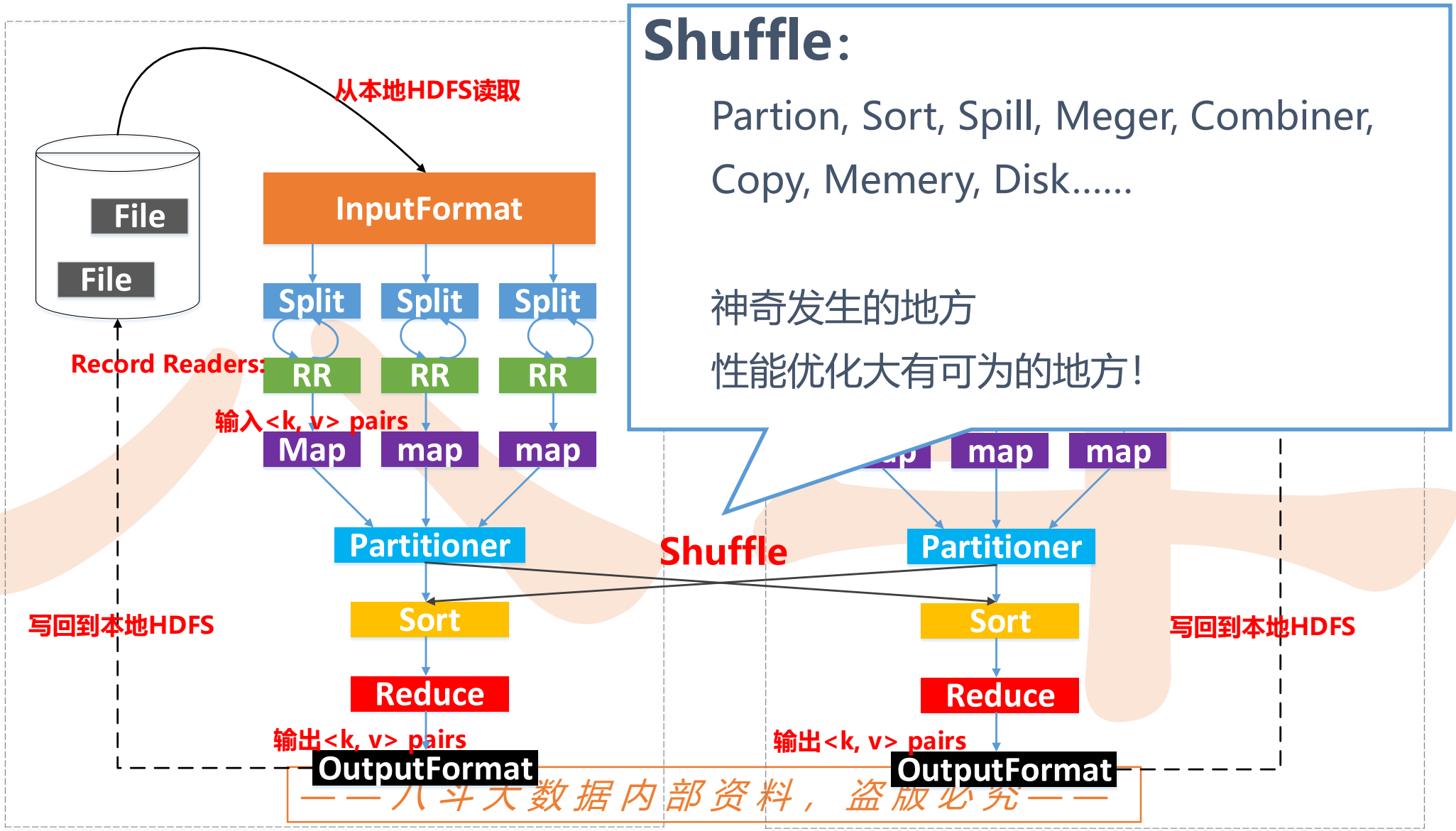
然后继续这个过程，读取下一条记录直到split尾部。

Reduce
输出<k, v> pairs
OutputFormat

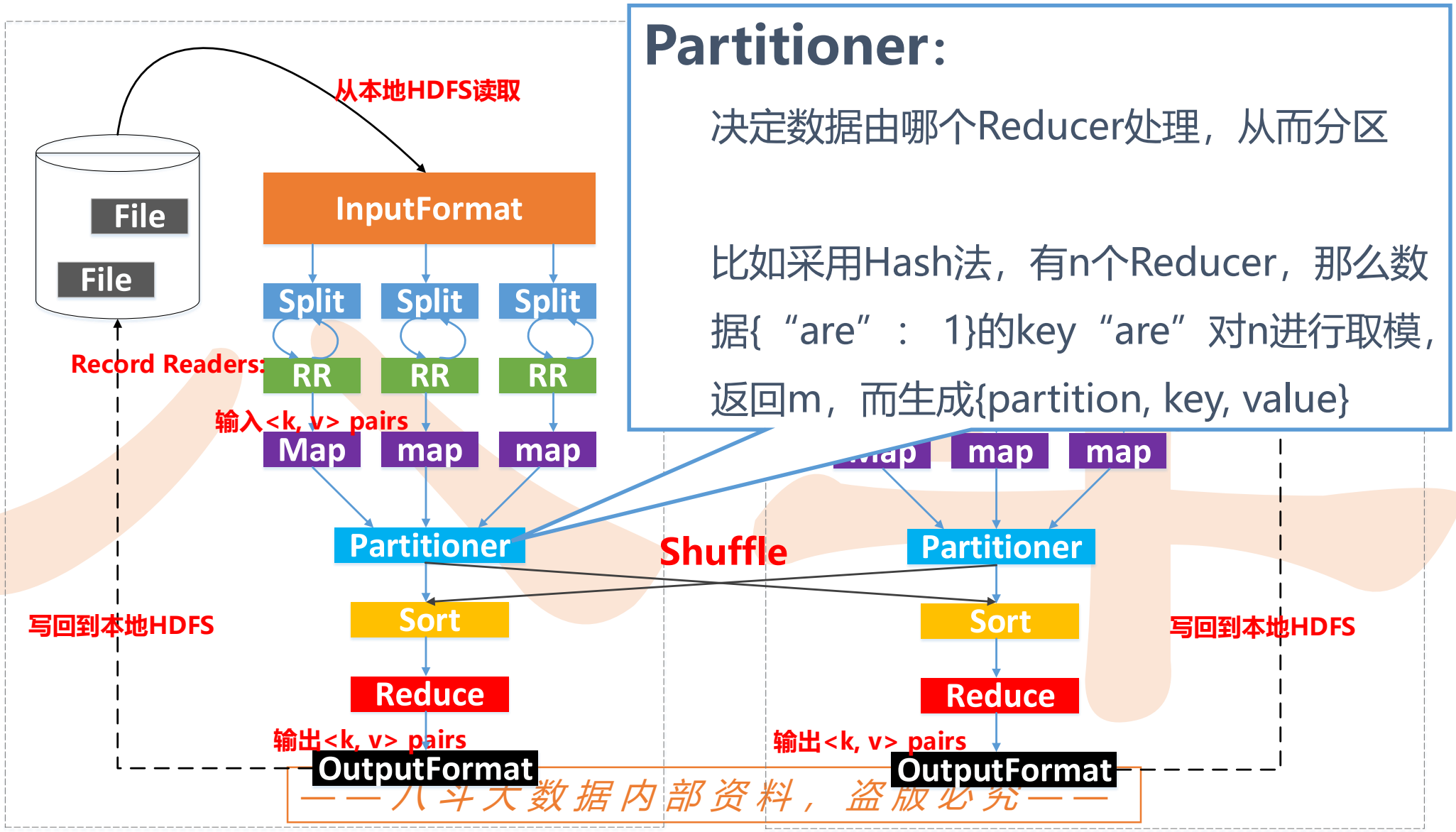
MapReduce计算框架-执行流程



MapReduce计算框架-执行流程



MapReduce计算框架-执行流程

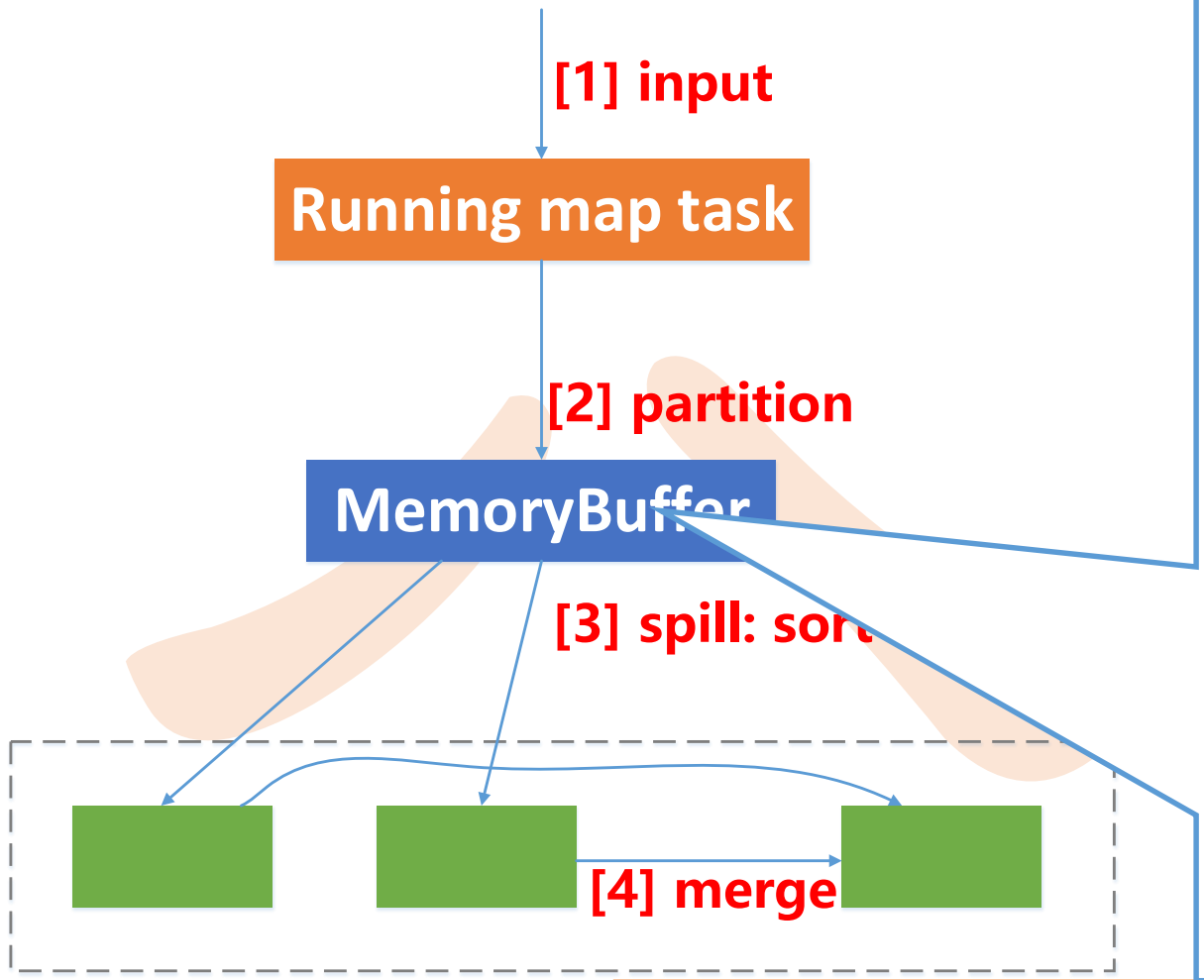


Partitioner:

决定数据由哪个Reducer处理，从而分区

比如采用Hash法，有n个Reducer，那么数据{“are”： 1}的key“are”对n进行取模，返回m，而生成{partition, key, value}

Map的MemoryBuffer



MemoryBuffer:

内存缓冲区，每个map的结果和partition处理的key value结果都保存在缓存中

缓冲区大小：默认100M

溢写阈值：100M * **0.8** = 80M

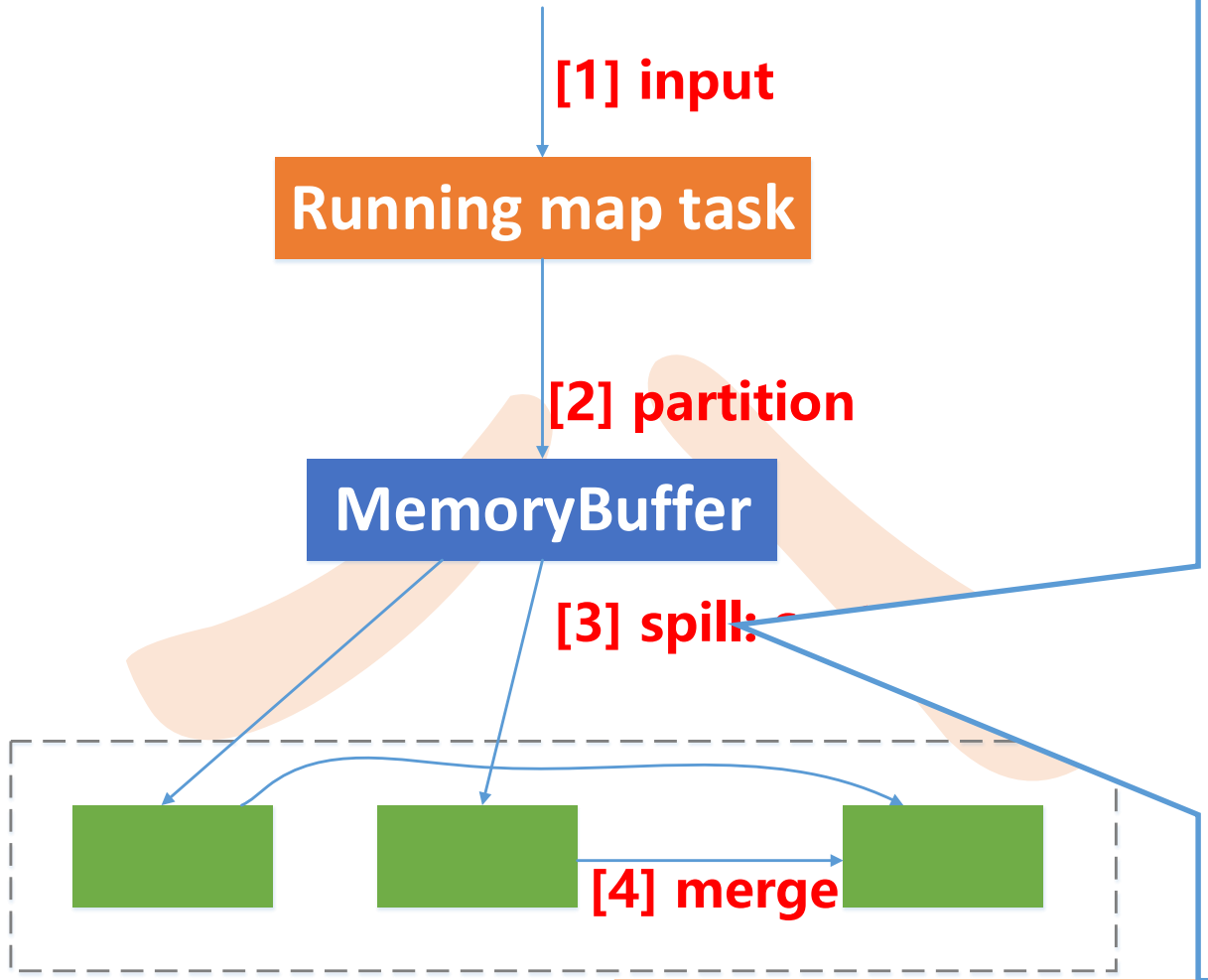
缓冲区中的数据：partition key value 三元组数据

{ “1” , “are” : 1}

{ “2” , “at” : 1}

{ “1” , “we” : 1}

Map的Spill



Spill:

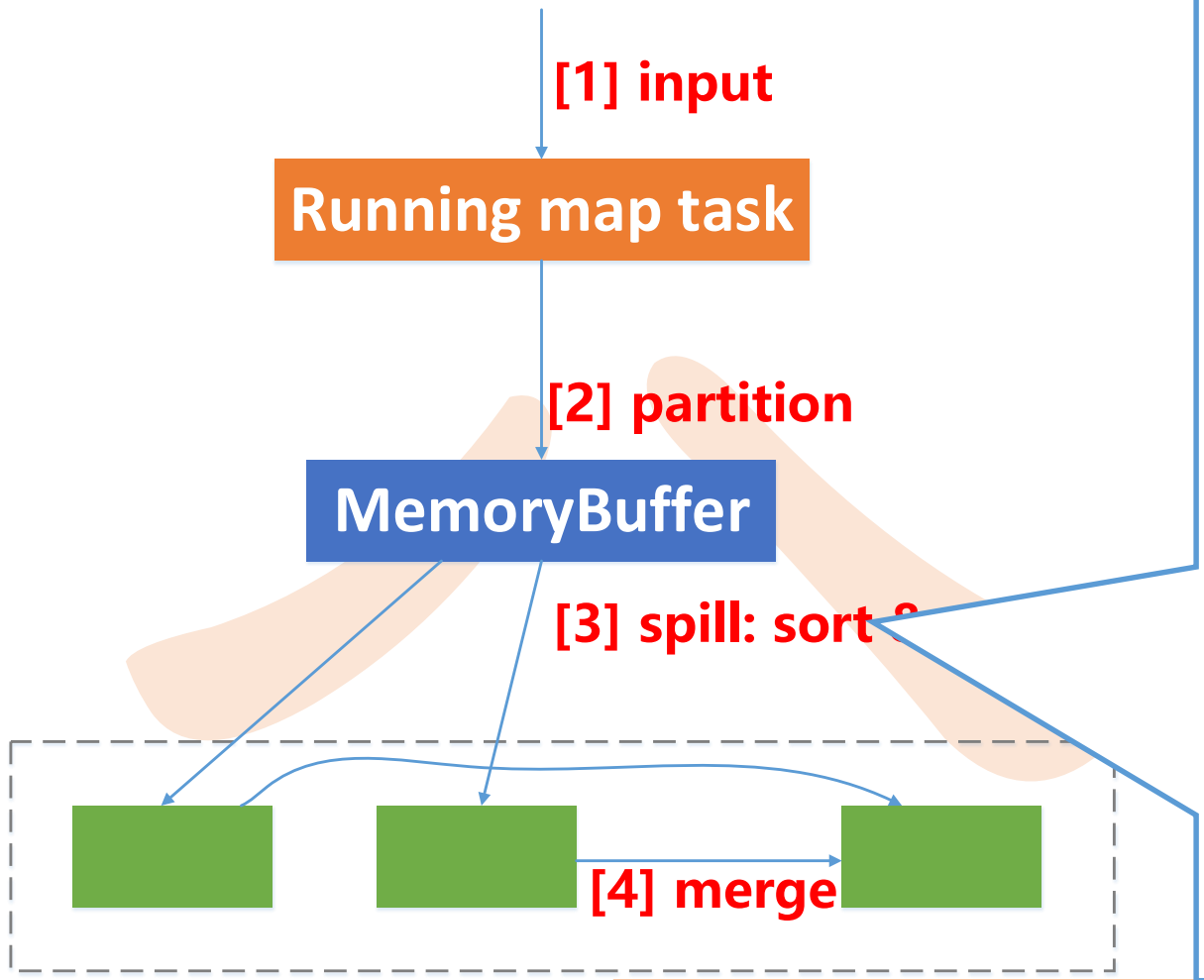
内存缓冲区达到阈值时，溢写spill线程锁住这80M的缓冲区，开始将数据写出到本地磁盘中，然后释放内存。

每次溢写都生成一个数据文件。

溢出的数据到磁盘前会**对数据进行key排序sort**，以及合并combiner

发送相同Reduce的key数量，会拼接到一起，减少partition的索引数量。

Map的Sort



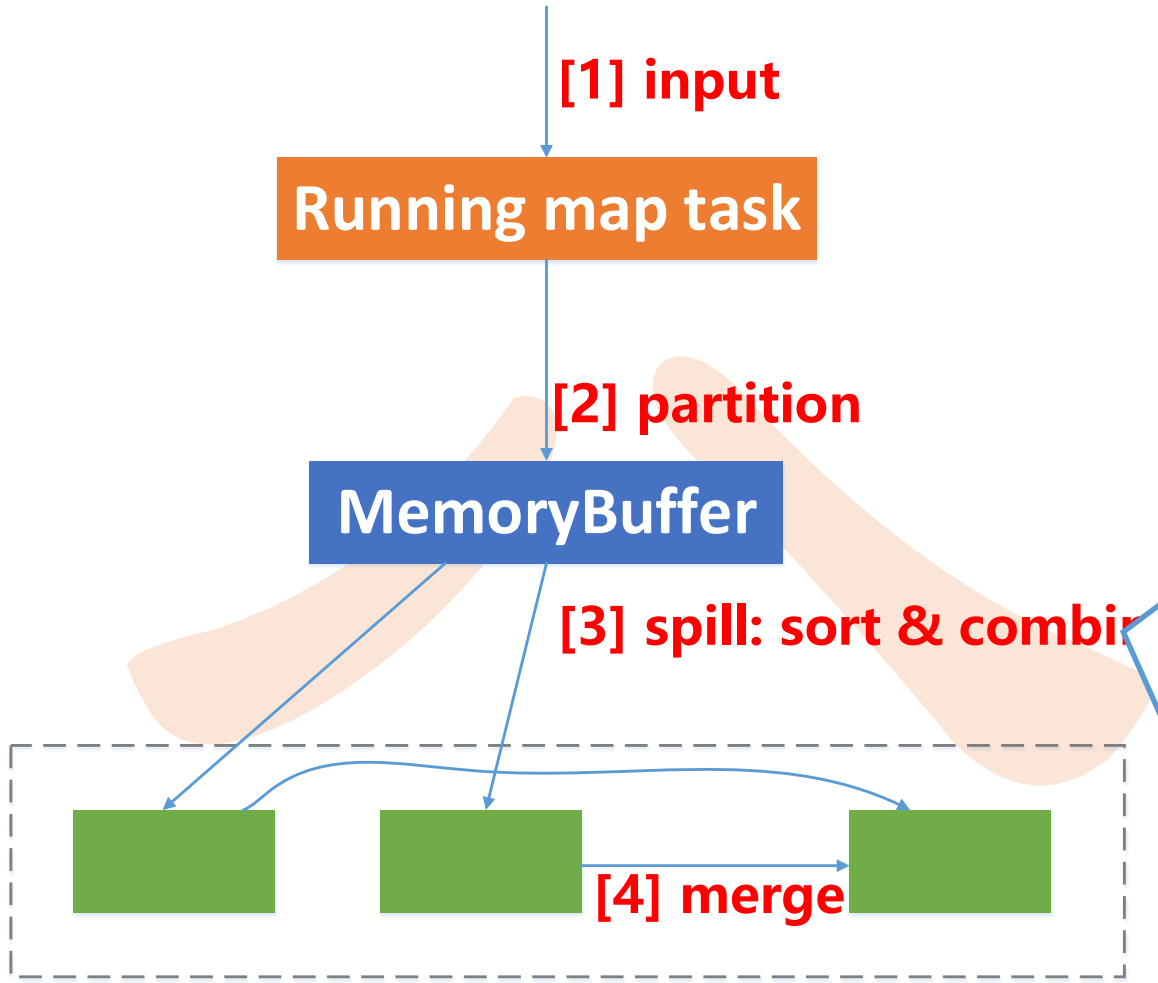
Sort:

缓冲区数据按照key进行排序

{ “1” , “are” , 1}
{ “2” , “at” , 1}
……
{ “1” , “are” , 1}
{ “1” , “we” , 1}

{ “1” , “are” , 1}
{ “1” , “are” , 1}
{ “1” , “we” , 1}
……
{ “2” , “at” , 1}

Map的Combiner



Combiner：

数据合并，相同的key的数据，value值合并，减少输出传输量

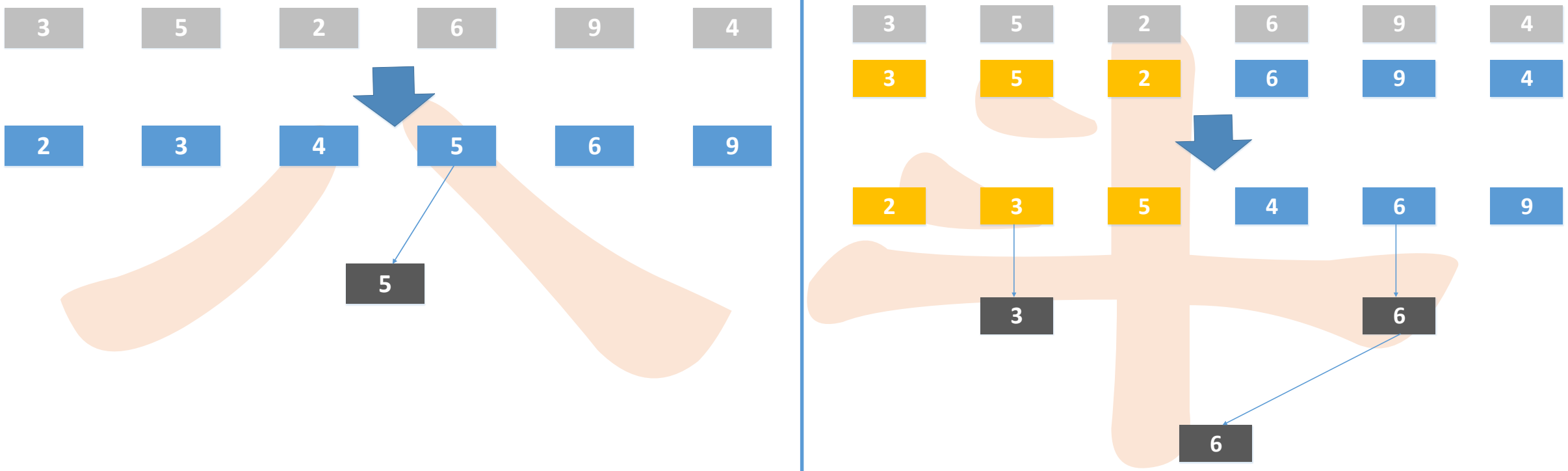
Combiner函数事实上是reducer函数，满足combiner处理不影响{sum，max等}最终reduce的结果时，可以极大提升性能

{ “1” ， “are” ， 1}
{ “1” ， “are” ， 1}
{ “1” ， “we” ， 1}

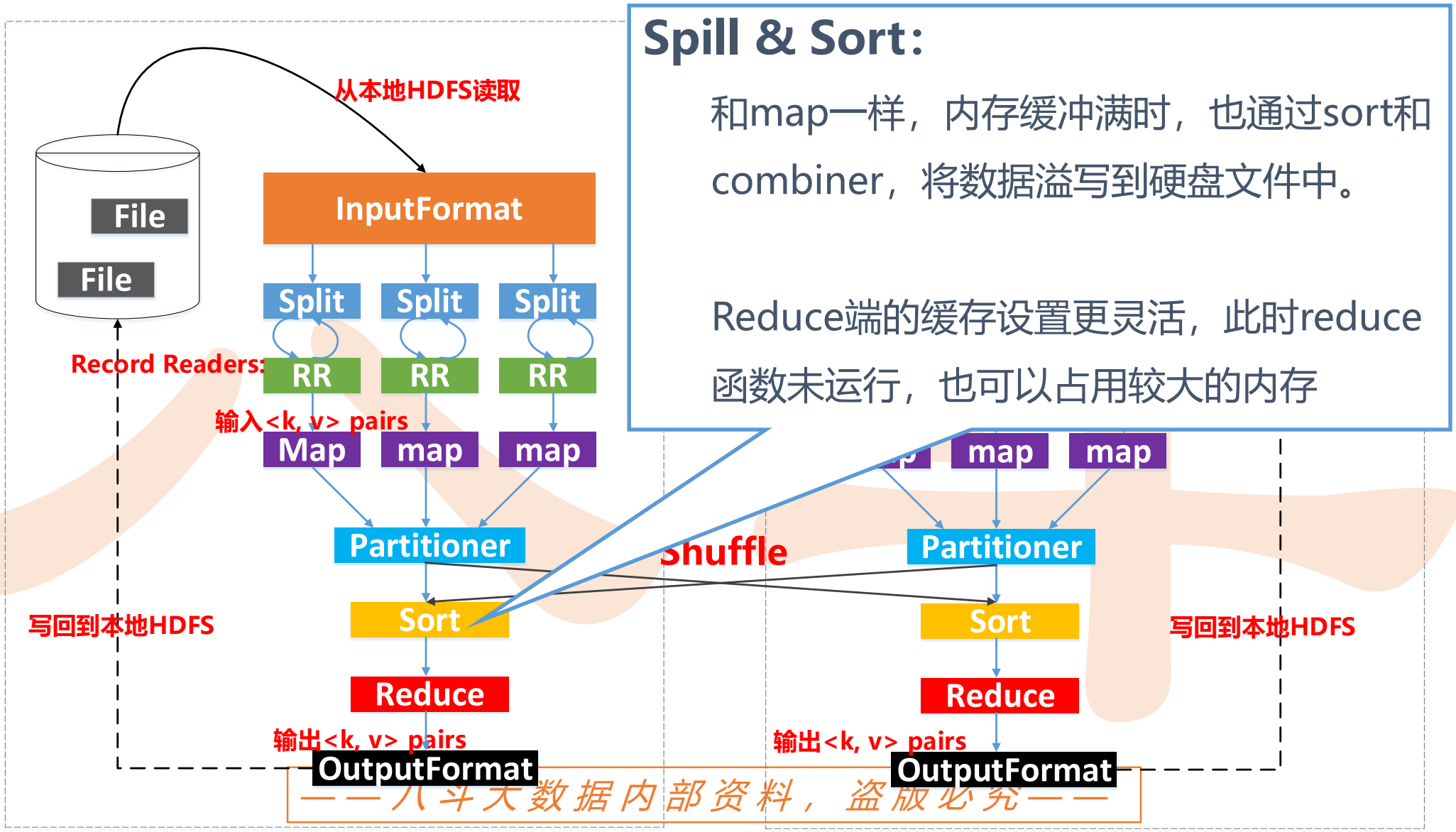
{ “1” ， “are” ， 2}
{ “1” ， “we” ， 1}

Map的Combiner的badcase

- 求中值:



MapReduce计算框架-执行流程

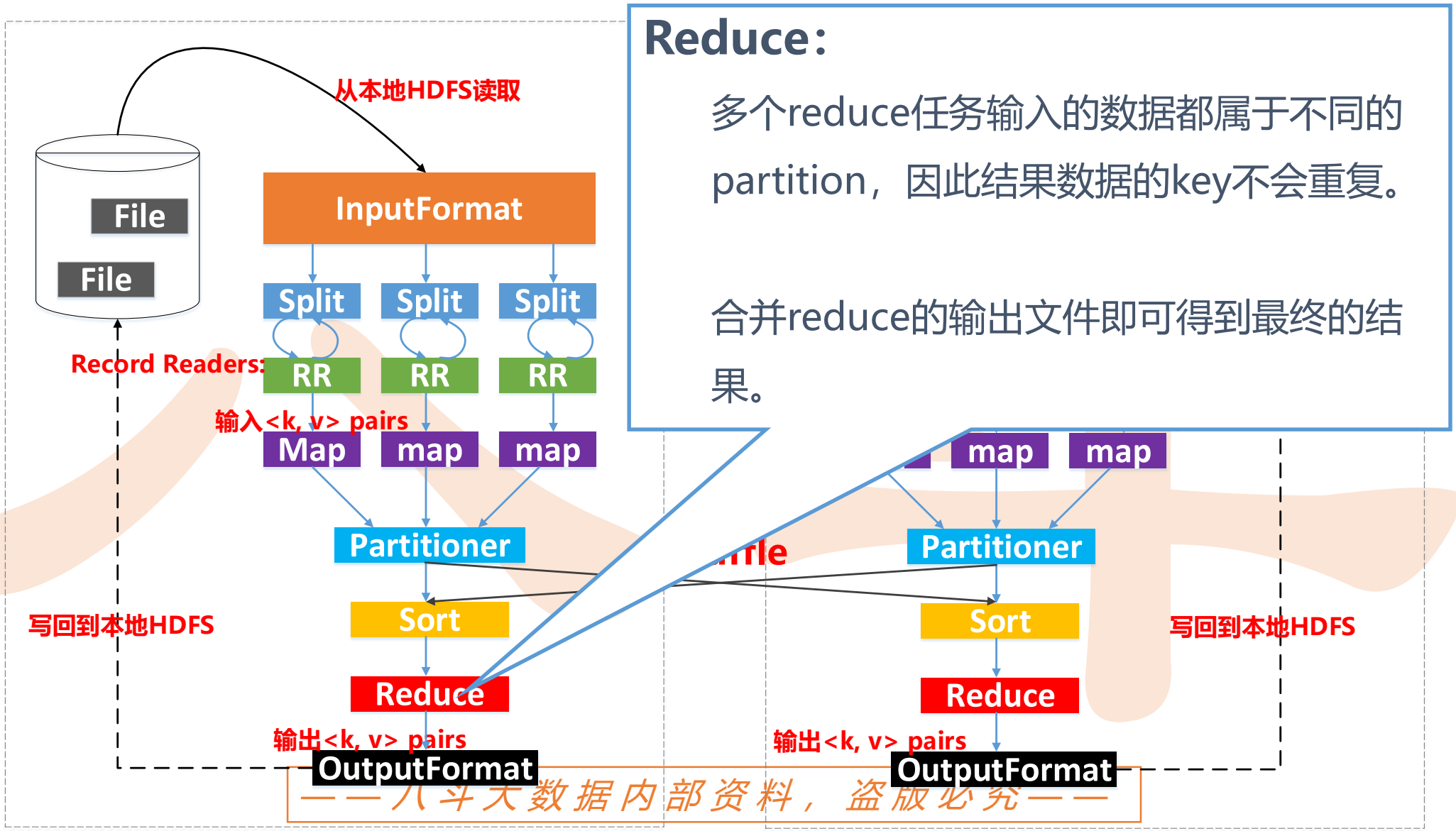


Spill & Sort:

和map一样，内存缓冲满时，也通过sort和combiner，将数据溢写到硬盘文件中。

Reduce端的缓存设置更灵活，此时reduce函数未运行，也可以占用较大的内存

MapReduce计算框架-执行流程



Reduce：

多个reduce任务输入的数据都属于不同的partition，因此结果数据的key不会重复。

合并reduce的输出文件即可得到最终的结果。

MapReduce物理配置

- 文件句柄个数
  - ulimit
- cpu
  - 多核
- 内存
  - 8G以上
- 合适的slot
  - 单机map、reduce个数
  - mapred.tasktracker.map.tasks.maximum（默认2）
  - mapreduce.tasktracker.tasks.reduce.maximum（默认2）
  - 内存限制
  - cpu核数-1
  - 多机集群分离
- 磁盘情况
  - 合适单机多磁盘
  - mapred.local.dir和dfs.data.dir

编 程 模 型

- 借鉴函数式的编程方式
- 用户只需要实现两个函数接口：
  - Map(in_key, in_value)

    -> (out_key, intermediate_value) list
  - Reduce (out_key, intermediate_value list)

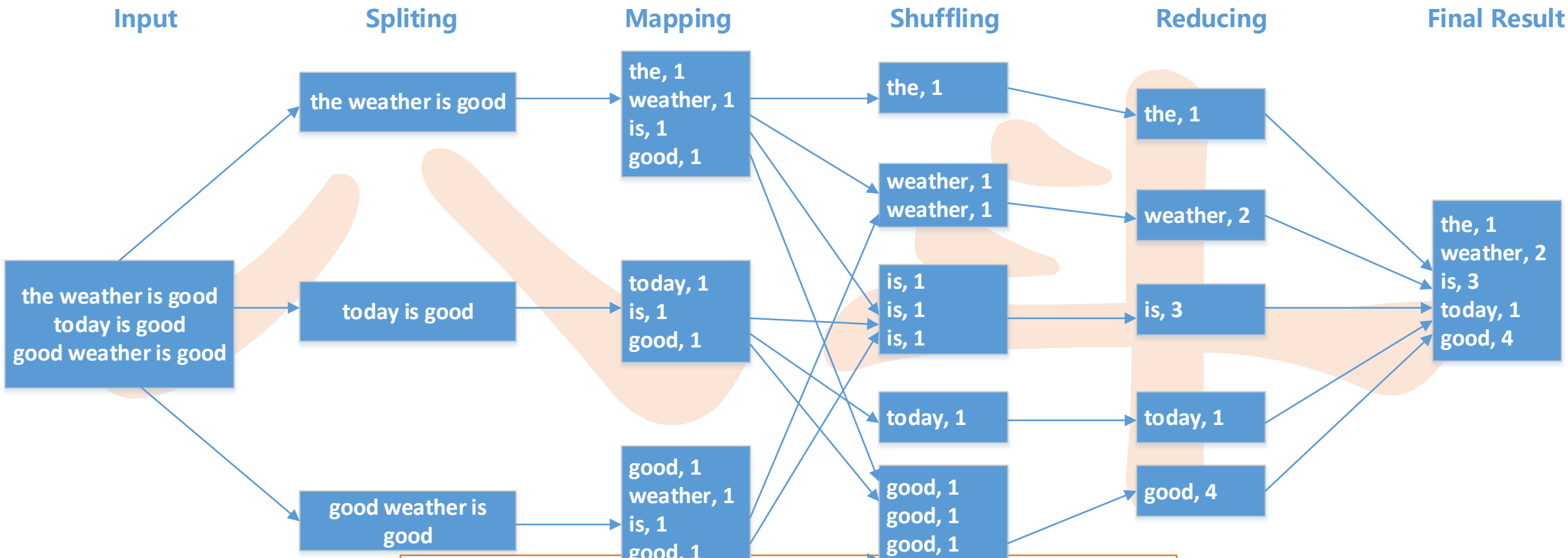
    ->out_value list

编程示例

- WordCount实现过程
- 源数据
  - Document 1
    - the weather is good
  - Document 2
    - today is good
  - Document 3
    - good weather is good

上机实践-WordCount

- 我的第一个MapReduce任务



编程示例

- Map 输出
  - Worker 1
    - (the 1), (weather 1), (is 1), (good 1)
  - Worker 2
    - (today 1), (is 1), (good 1)
  - Worker 3
    - (good 1), (weather 1), (is 1), (good 1)

编程示例

- Reduce 输入
  - Worker 1
    - (the 1)
  - Worker 2
    - (is 1), (is 1), (is 1)
  - Worker 3
    - (weather 1), (weather 1)
  - Worker 4
    - (today 1)
  - Worker 5
    - (good 1), (good 1), (good 1), (good 1)

编程示例

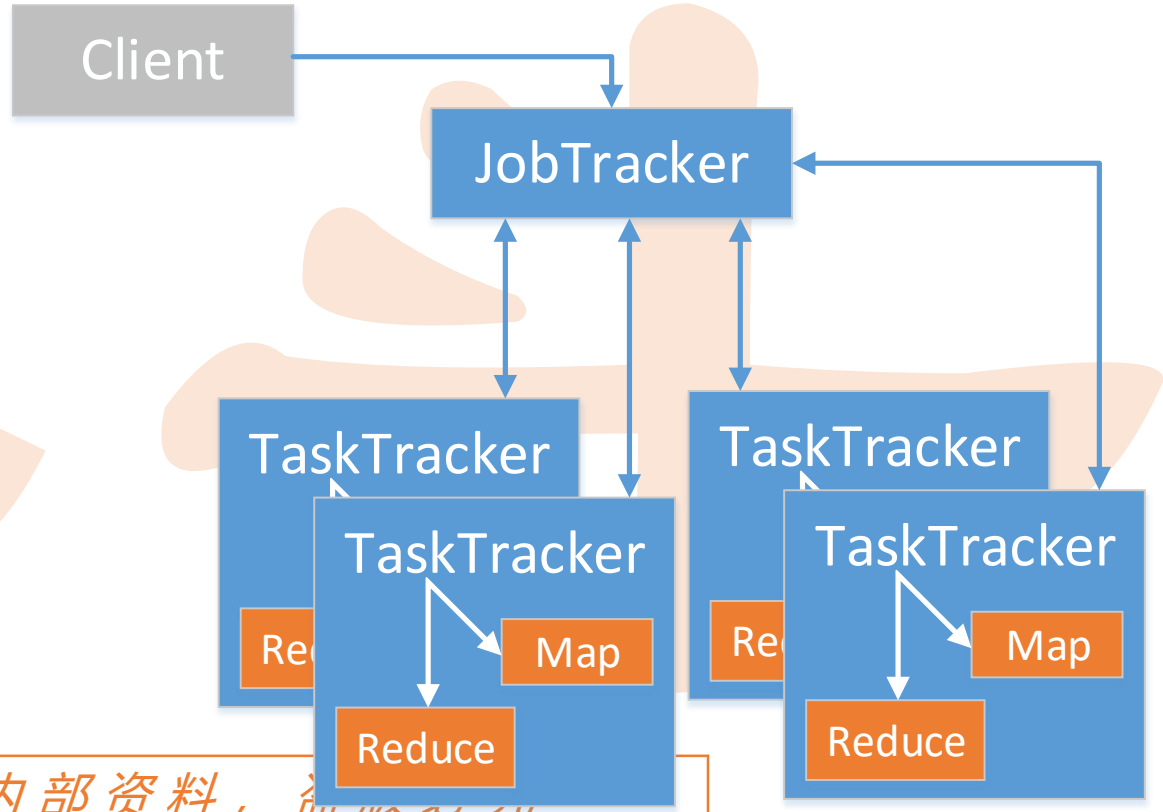
- Reduce 输出
  - Worker 1
    - (the 1)
  - Worker 2
    - (is 3)
  - Worker 3
    - (weather 2)
  - Worker 4
    - (today 1)
  - Worker 5
    - (good 4)

MapReduce实现架构

- 两个重要的进程
  - JobTracker
    - 主进程，负责接收客户作业提交，调度任务到作节点上运行，并提供诸如监控工作节点状态及任务进度等管理功能，一个MapReduce集群有一个jobtracker，一般运行在可靠的硬件上。
    - tasktracker是通过周期性的心跳来通知jobtracker其当前的健康状态，每一次心跳包含了可用的map和reduce任务数目、占用的数目以及运行中的任务详细信息。Jobtracker利用一个线程池来同时处理心跳和客户请求。
  - TaskTracker
    - 由jobtracker指派任务，实例化用户程序，在本地执行任务并周期性地向jobtracker汇报状态。在每一个工作节点上永远只会有一个tasktracker

MapReduce工作原理

- JobTracker一直在等待JobClient提交作业
- TaskTracker每隔3秒向JobTracker发送心跳询问有没有任务可做，如果有，让其派发任务给它执行
- Slave主动向master拉生意



OutLine

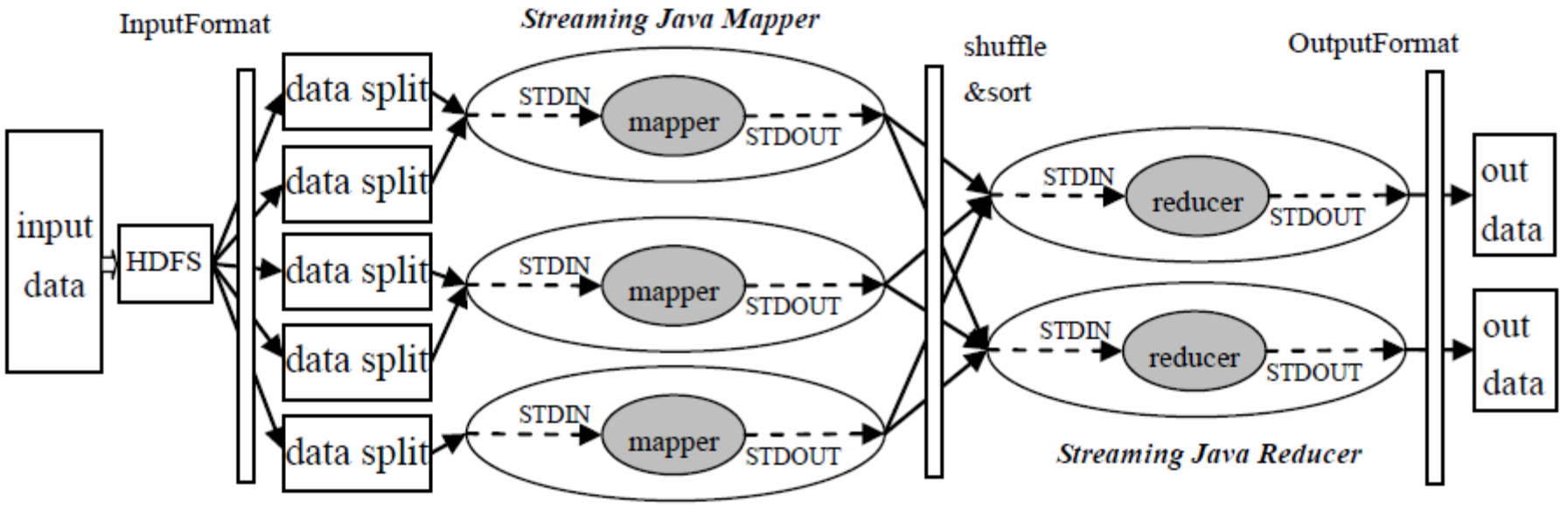
MapReduce

Hadoop Streaming

Streaming简介

- MapReduce和HDFS采用Java实现，默认提供Java编程接口
- Streaming框架允许任何程序语言实现的程序在Hadoop MapReduce中使用
- Streaming方便已有程序向Hadoop平台移植

Streaming原理



Hadoop Streaming 计算流程

Streaming优点

- 开发效率高
  - 方便移植Hadoop平台，只需按照一定的格式从标准输入读取数据、向标准输出写数据就可以
  - 原有的单机程序稍加改动就可以在Hadoop平台进行分布式处理
  - 容易单机调试

cat input | mapper | sort | reducer > output

- 程序运行效率高
  - 对于CPU密集的计算，有些语言如C/C++编写的程序可能比用Java编写的程序效率更高一些
- 便于平台进行资源控制
  - Streaming框架中通过limit等方式可以灵活地限制应用程序使用的内存等资源

Quick Start

```
$HADOOP_HOME/bin/hadoo
    -input /user/test/input \
    -output /user/test/output \
    -mapper  “python mapper.sh” \
    -reducer  “python reducer
    -file mapper.sh \
    -jobconf mapred.job.name
```

input：

指定作业的输入文件的HDFS路径，支持使用*通配符，支持指定多个文件或目录，可多次使用

output：

指定作业的输出文件的HDFS路径，路径必须不存在，并且具备执行作业用户有创建该目录的权限，只能使用一次

Quick Start

```
$HADOOP_HOME/bin/hadoop
    -input /user/test/input \
    -output /user/test/output \
    -mapper “python mapper.py” \
    -reducer “python reducer.py” \
    -file mapper.sh \
    -jobconf mapred.job.name=” xxx”
```

mapper:

用户自己写的mapper程序

reduer:

用户自己写的reduce程序

Quick Start

```
$HADOOP_HOME/bin/hadoop
    -input /user/test/input \
    -output /user/test/output \
    -mapper “python mapper
    -reducer “pyth
    -file mapper.sh \
    -jobconf mapred.job.name=” xxx”
```

file:

打包文件到提交的作用中，

（1）map和reduce的执行文件

（2）map和reduce要用输入的文件，如配置文件

类似的配置还有-cacheFile, -cacheArchive分别用于向计算节点分发HDFS文件和HDFS压缩文件

Quick Start

```
$HADOOP_HOM
    -input /user/t
    -output /user/
    -mapper “py
    -reducer “py
    -file mapper.sh \
    -jobconf mapred.job.name=” xxx”
```

jobconf:

提交作业的一些配置属性

常见配置：

（1）mapred.map.tasks：map task数目

（2）mapred.reduce.tasks：reduce task数目

（3）stream.num.map.output.key.fields：指定map task输出记录中key所占的域数目

（4）num.key.fields.for.partition指定对key分出来的前几部分做partition而不是整个key

-jobconf

mapred.job.name	作业名
mapred.job.priority	作业优先级
mapred.job.map.capacity	最多同时运行map任务数
mapred.job.reduce.capacity	最多同时运行reduce任务数
mapred.task.timeout	任务没有响应（输入输出）的最大时间
mapred.compress.map.output	map的输出是否压缩
mapred.map.output.compression.codec	map的输出压缩方式
mapred.output.compress	reduce的输出是否压缩
mapred.output.compression.codec	reduce的输出压缩方式
stream.map.output.field.separator	map输出分隔符

Q & A

@八斗学院