ODF 2017 开源数据库论坛(北京)

开源数据库正在改变世界

2017年8月24日-25日　北京 - 京仪大酒店

大数据时代系统体系架构和对比：存储和计算

吴国泉 小米HBase研发工程师

目录

CONTENTS

存储

MySQL、HBase、Elastcisearch的特点和区别

存储方式

MySQL

- 广泛应用于OLTP业务，支持事务，提供二级索引

HBase

- 面向海量数据存储，良好的写性能，读性能稍差，不支持事务和二级索引

ES

- 适用于复杂查询和全文检索，不支持事务

存储

1. 存储方式
   - 行存储
   - 列存储
   - 索引存储
2. 读写方式
   - 随机IO
   - 顺序IO

存储方式

1. 行存储 VS 列存储

ID	name	age	sex	aihao
1	小明	21	男	nv
2	隔壁老王	25		隔壁孩子
3	小丽	3	女	

行存储：适合OLTP

列存储：适合OLAP

存储方式

1. HBase列族存储

RowKey	family（列族）			family（列族）	
1	name：小明	age：21	sex：男	aihao：nv	addr：北京
2	name：小丽	…			

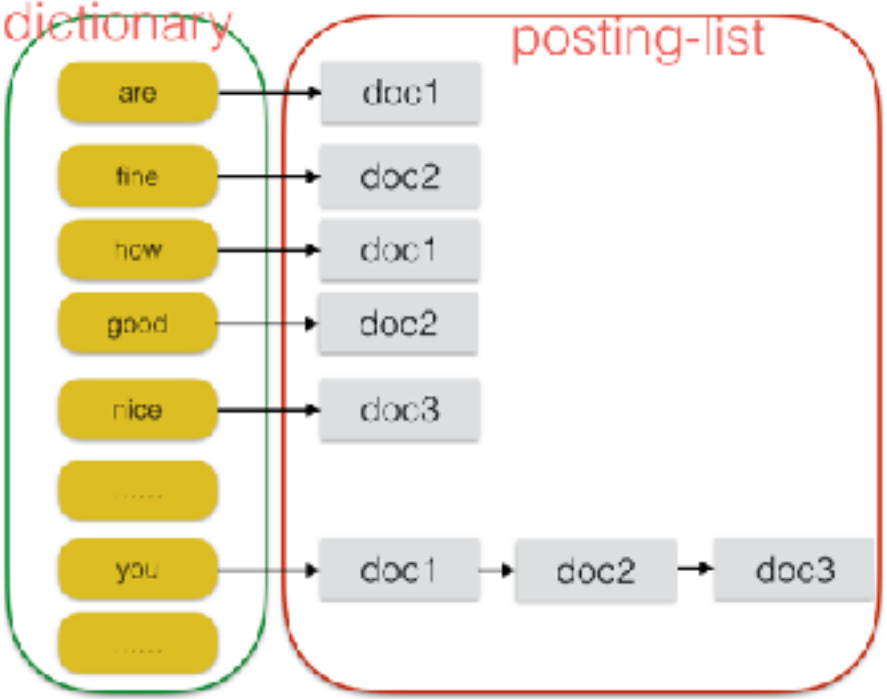
存储方式

1. ES

原始文档

文档id	文档内容
doc1	how are you?
doc2	Fine，thank you
doc3	nice to meet you

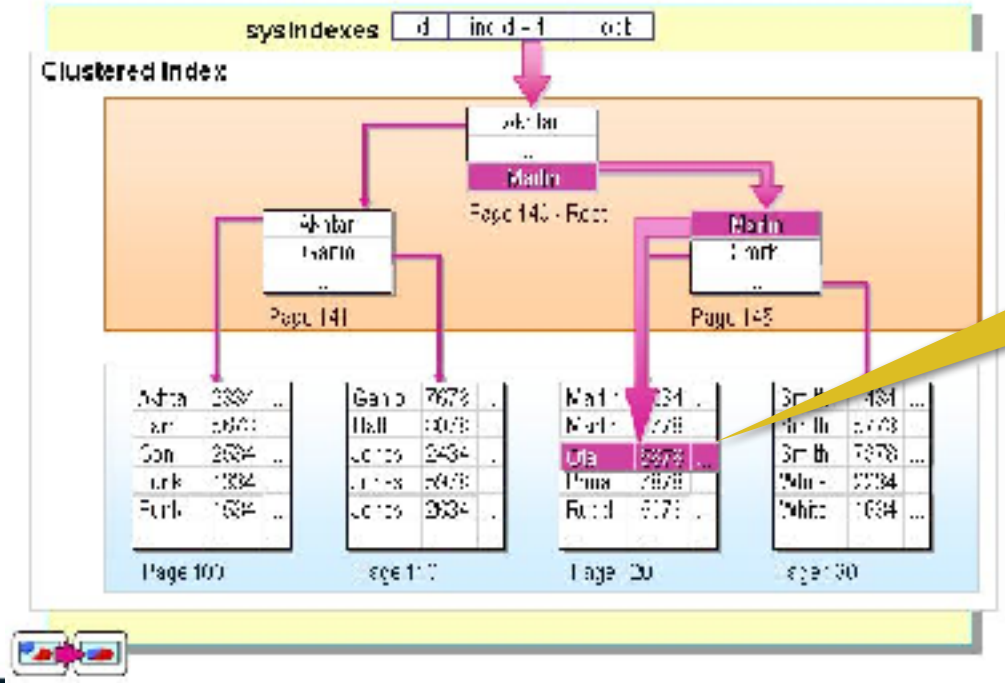
倒排索引



读写方式

MySQL

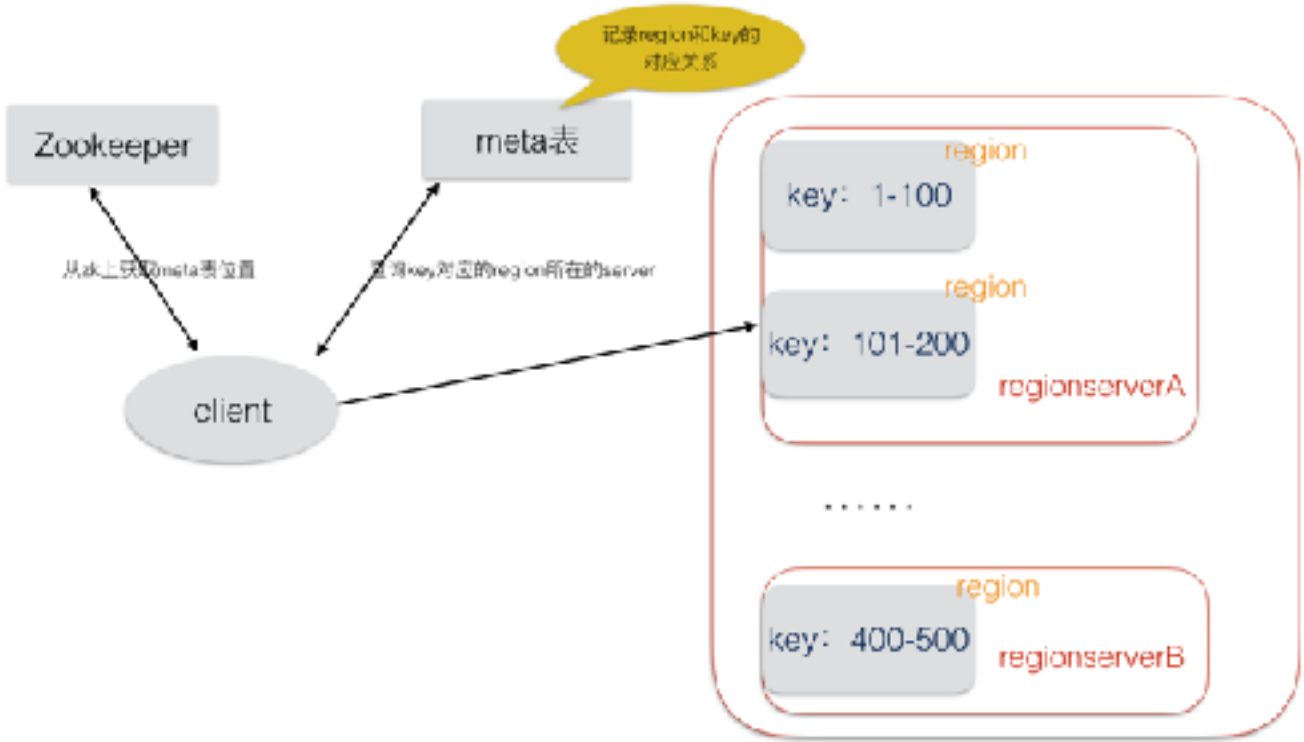
Finding Rows in a Clustered Index



读写可能都有随机IO

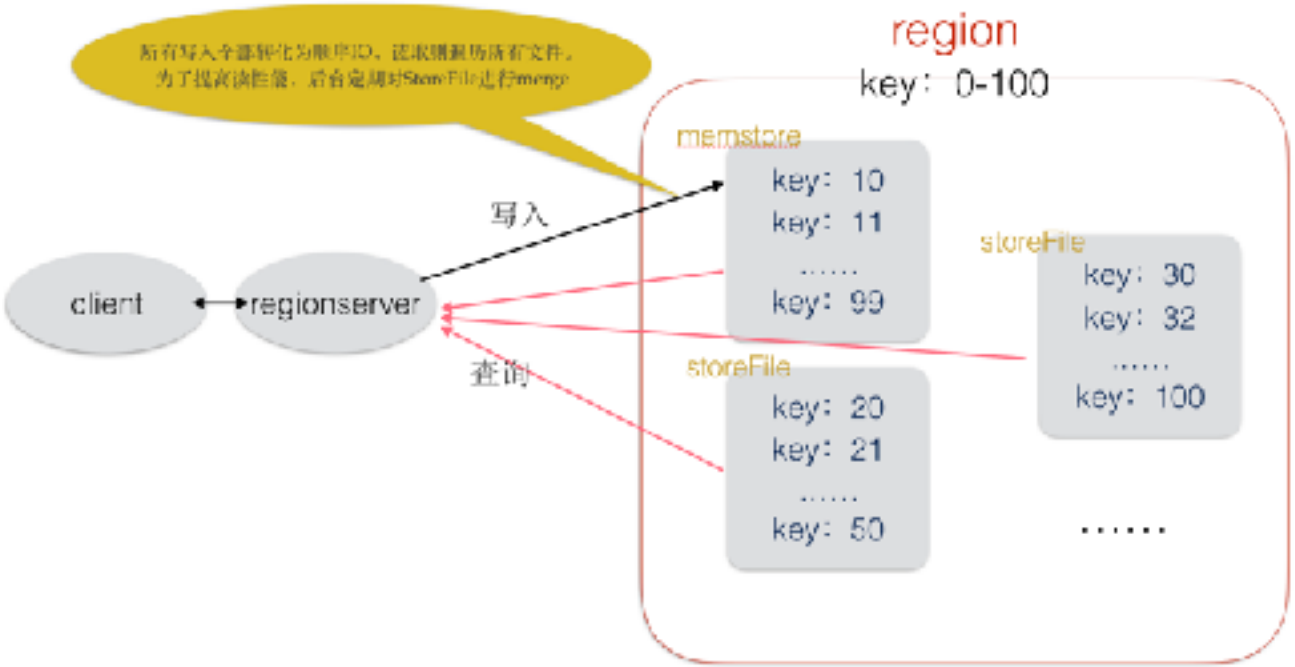
读写方式

HBase


记录region和key的
对应关系
Zookeeper
meta表
client
region
key: 1-100
region
key: 101-200
regionserverA
.....
region
key: 400-500
regionserverB

读写方式

HBase


region
key: 0-100
memstore
key: 10
key: 11
.....
key: 99
storeFile
key: 30
key: 32
.....
key: 100
storeFile
key: 20
key: 21
.....
key: 50
.....
写入
client
regionserver
查询

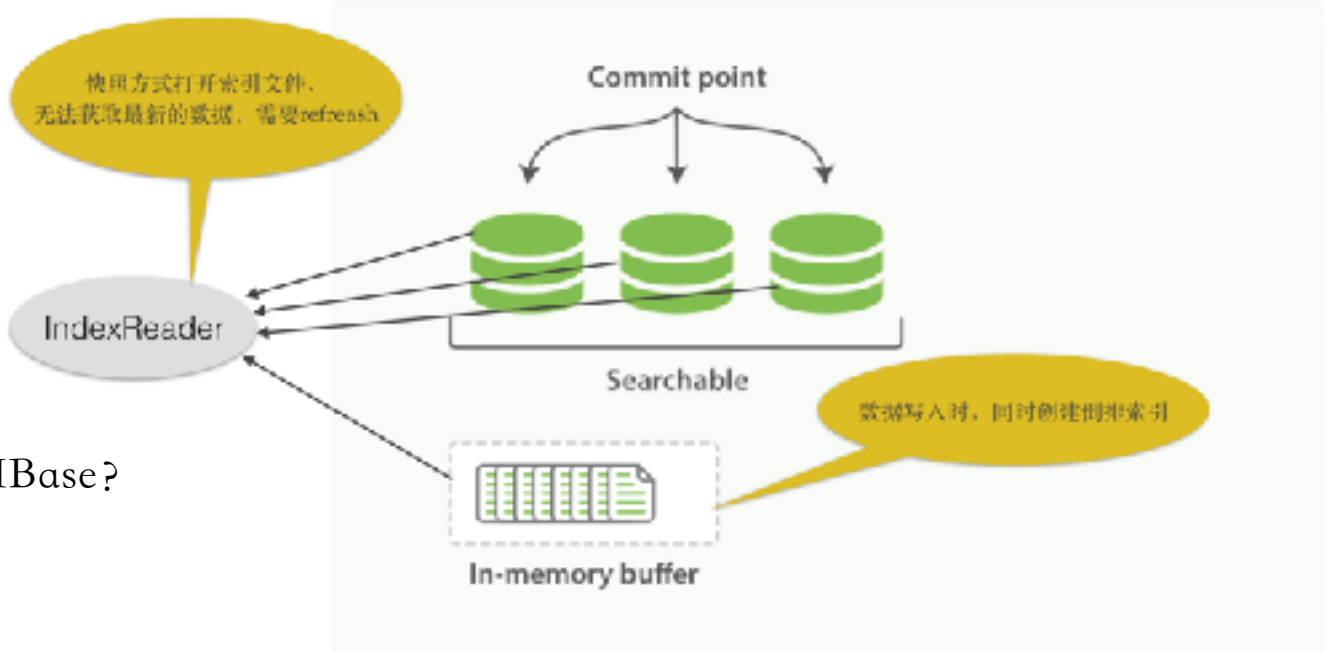
读写方式

ES


数据根据一定hash规则，写入对应的shard
CLUSTER
NODE 1 - ★MASTER
P0
P1
P2
index
CLUSTER
NODE 1 - ★MASTER
R0
P1
NODE 2
R0
R1
NODE 3
P0
R1

读写方式

ES



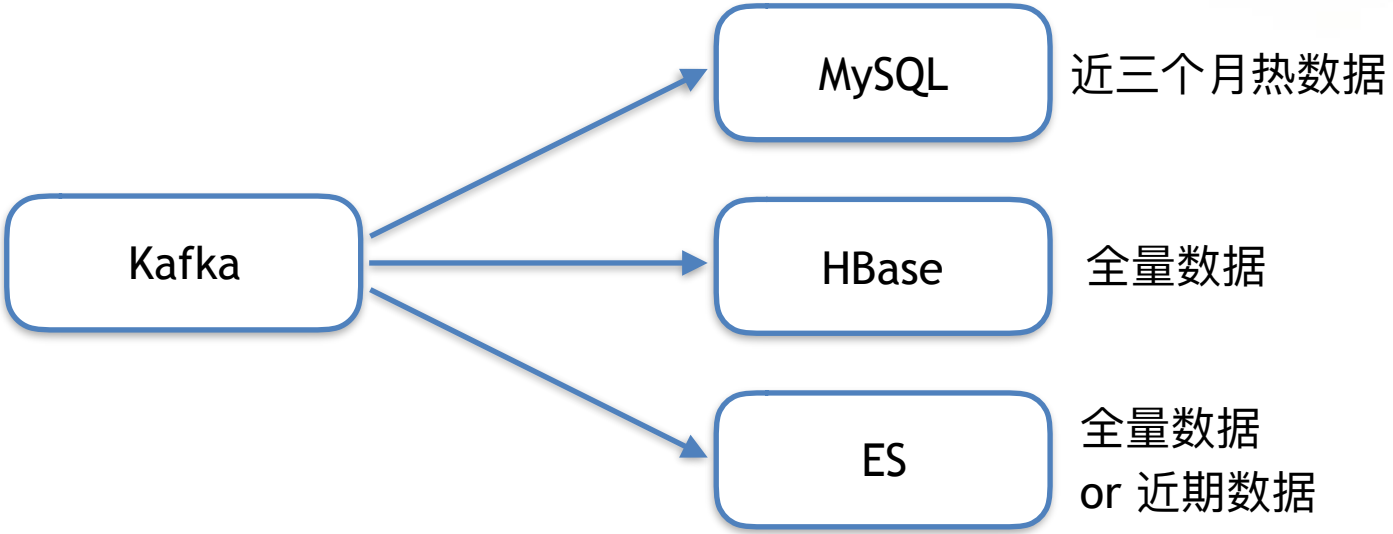
写性能媲美HBase?

存储

	MySQL	HBase	ES
存储方式	行存储 适用于OLTP业务	列族存储 平衡了OLTP、OLAP业务	索引存储 适用于各种检索业务
扩展性	单机，扩展性较差	水平扩展	水平扩展
事务	支持	不支持	不支持
一致性	strong consistency	strong consistency timeline consistency	可配置
secondary index	支持	不支持	支持
全文检索	支持 但很鸡肋	不支持	支持

例子

1. 监控系统



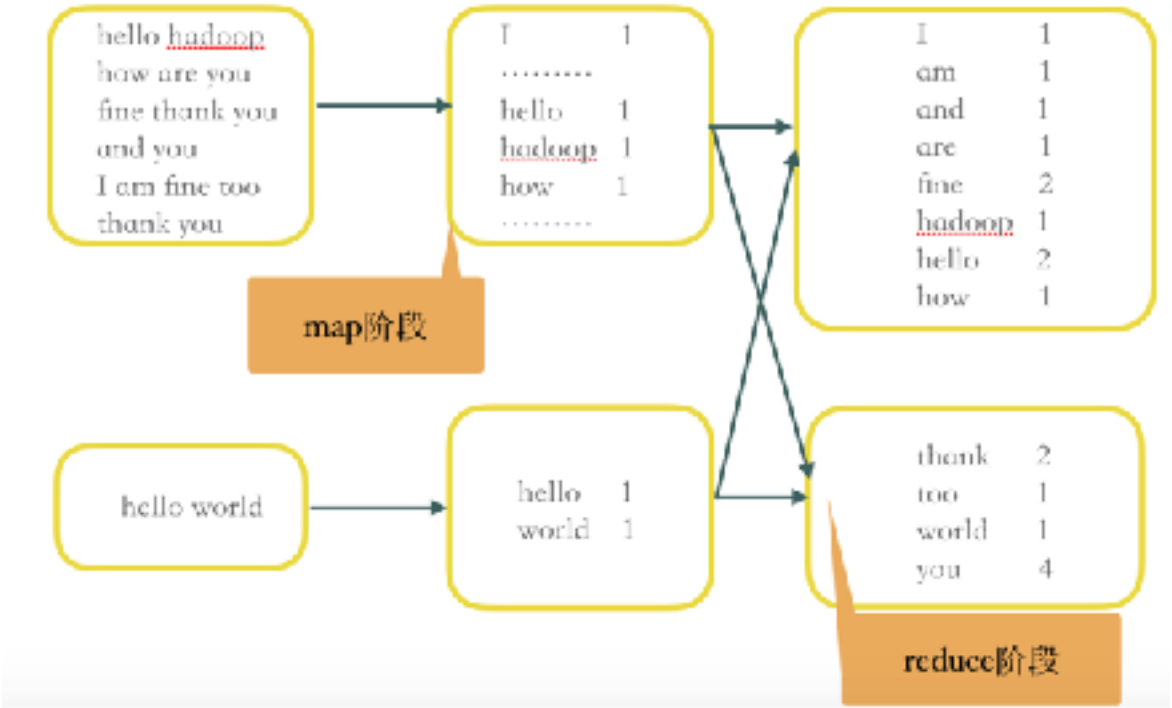
计算

计算

1. 离线计算
    1. MapReduce
    2. Spark
2. 实时计算
    1. 吞吐
    2. exactly once
    3. 数据乱序

离线计算

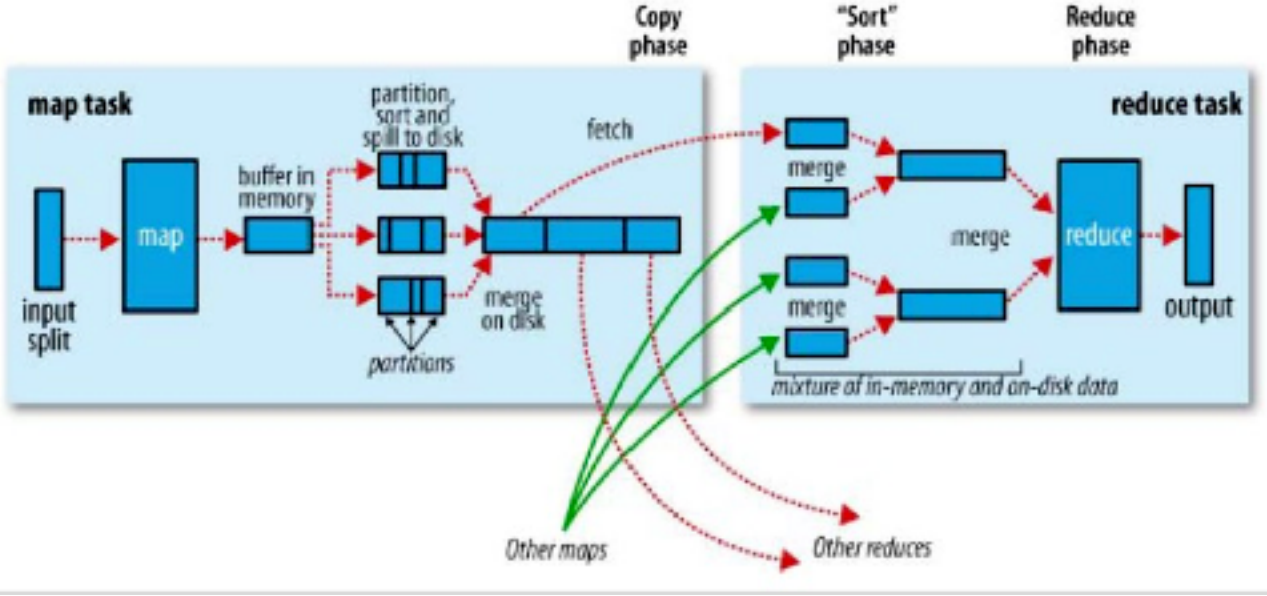
WordCount



离线计算

MapReduce

1.不能有效利用内存



离线计算

迭代计算

$$x^2-2x+1=0$$

第一步先把方程转换

$$X_{n+1}=\sqrt{2X_n-1}$$

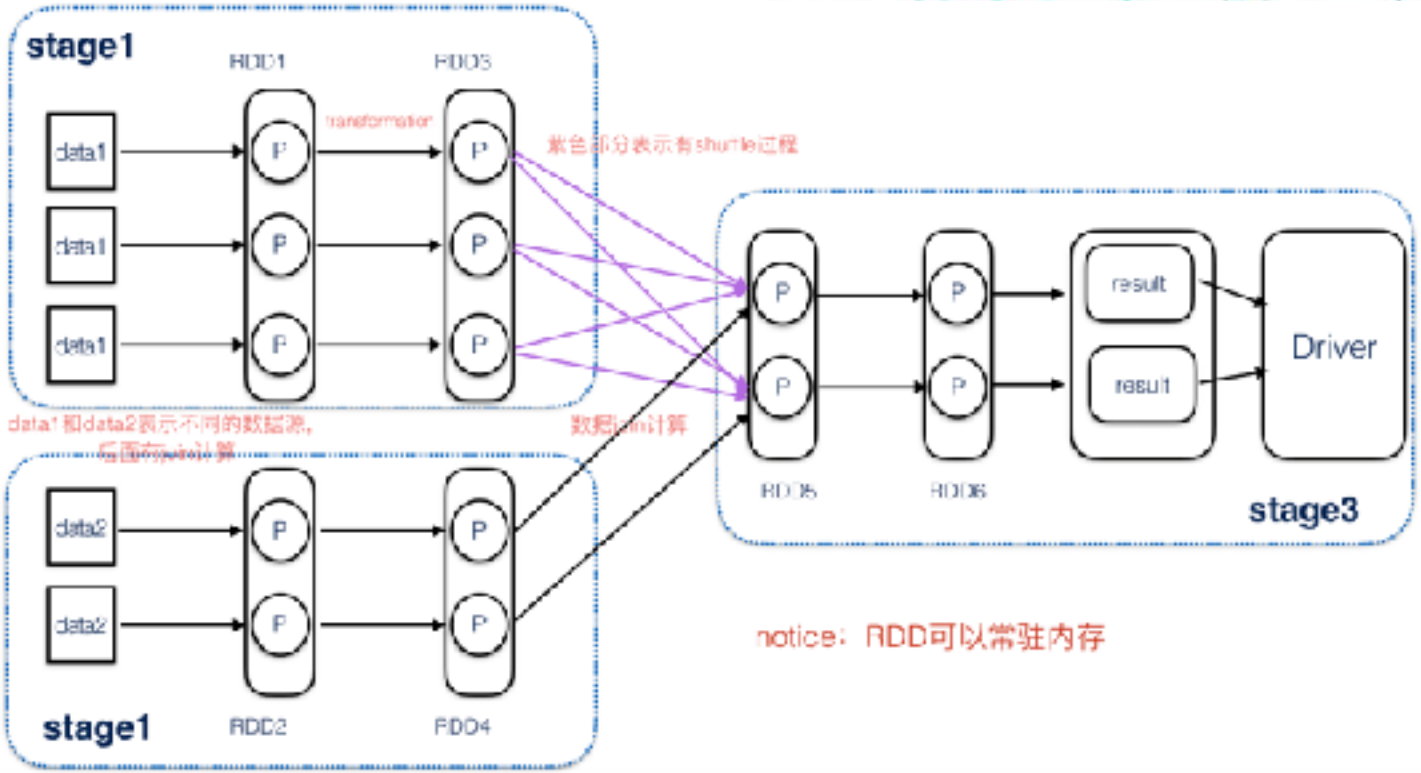
假设一个初始值
然后不断迭代

x = 1.5
x = 1.414
x = 1.34
x = 1.26
x = 1.22
.....
x = 1.04
.....
x = 1.000099
x = 1.000098

离线计算

Spark

1.RDD支持内存迭代
2.更友好的API



实时计算

1.广告实时推荐

2.feed流新闻

3.系统状态实时统计

.....

实时计算

~~离线=批量~~

~~实时=流式~~

离线和实时：数据处理的延迟

批量和流式：数据处理的方式

实时计算

统计：count（*）

流式计算可以完成批量计算的工作，
那为什么还要批量计算框架呢?

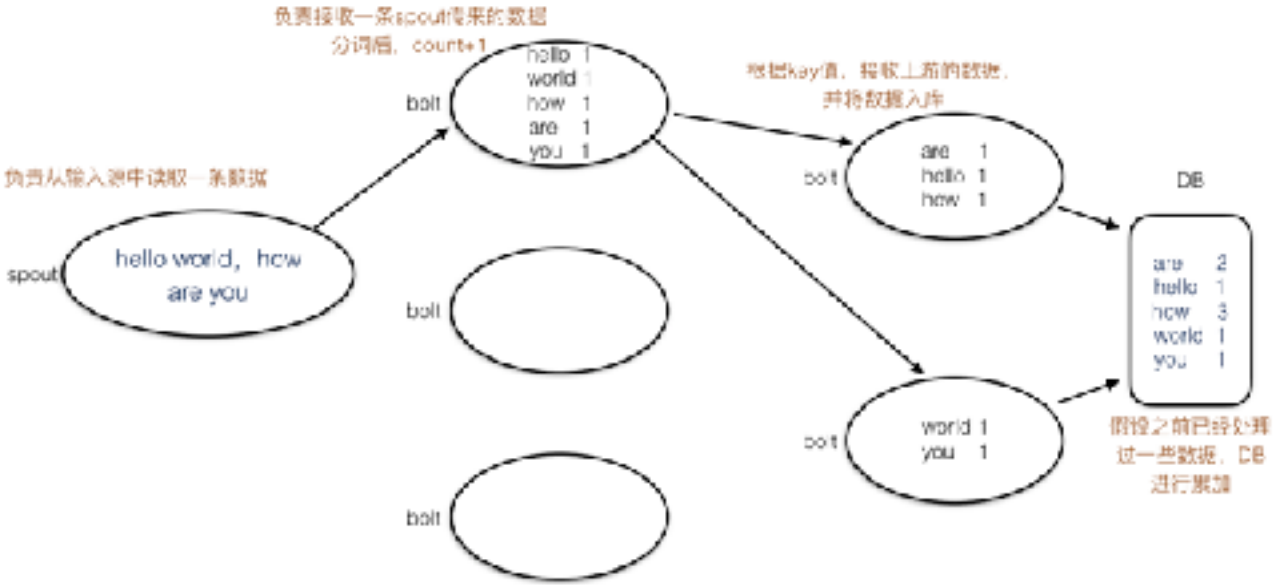
一个设计良好的流式系统可以完全取代批量系统

实时计算难点

1. 吞吐
2. exactly once
3. 数据乱序

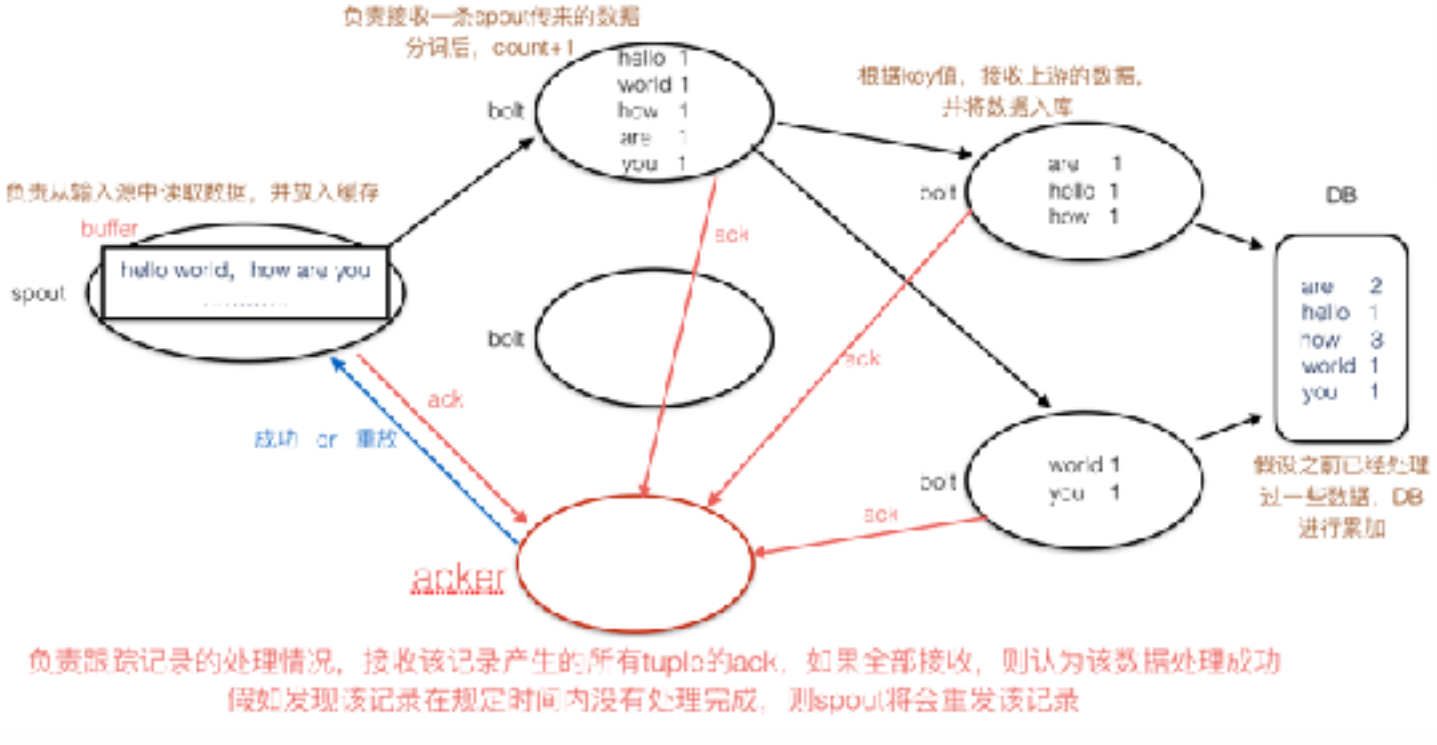
实时计算

Storm


负责接收一条spout传来的数据
分词后，count+1
bolt
hello 1
world 1
how 1
are 1
you 1
负责从输入源中读取一条数据
spout
hello world, how
are you
bolt
bolt
根据key值，接收上面的数据，
并将数据入库
bolt
are 1
hello 1
how 1
bolt
world 1
you 1
DB
are 2
hello 1
how 3
world 1
you 1
假设之前已经处理
过一些数据，DB
进行累加

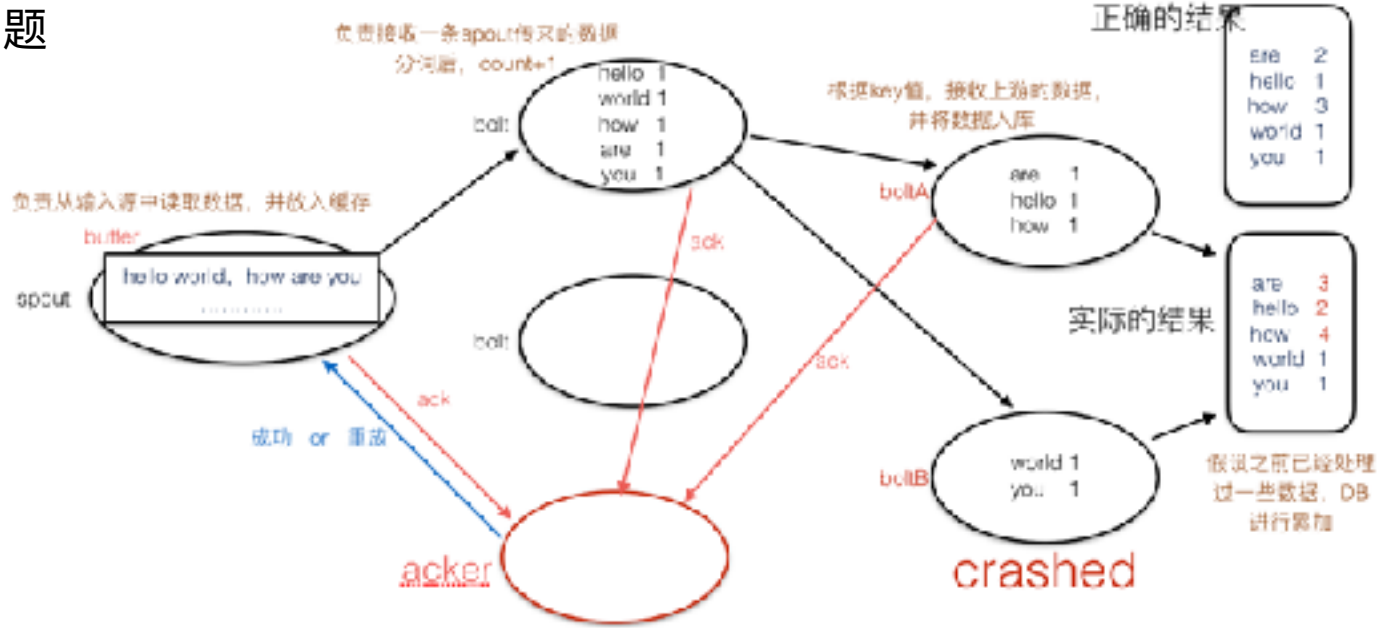
实时计算

容灾问题


负责接收一条spout传来的数据
分词后，count+1
bolt
hello 1
world 1
how 1
are 1
you 1
根据key值，接收上游的数据，
并将数据入库
负责从输入源中读取数据，并放入缓存
buffer
spout
hello world, how are you
bolt
bolt
are 1
hello 1
how 1
DB
are 2
hello 1
how 3
world 1
you 1
ack
ack
ack
成功 or 重放
world 1
you 1
ack
acker
假设之前已经处理
过一些数据，DB
进行累加
负责跟踪记录的处理情况，接收该记录产生的所有tuple的ack，如果全部接收，则认为该数据处理成功
假如发现该记录在规定时间内没有处理完成，则spout将会重发该记录

实时计算

重复计算问题

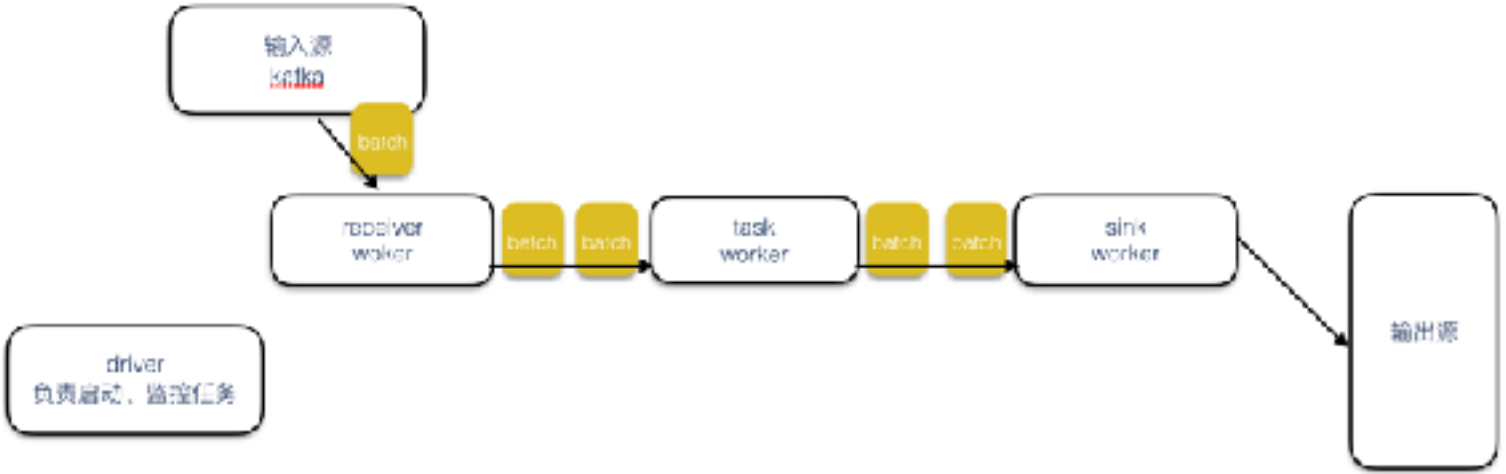


假如该record处理过程中，boltA处理成功并入库，boltB任务失败了，此时会导致数据重放，
DB中的部分数据会重复累加

实时计算

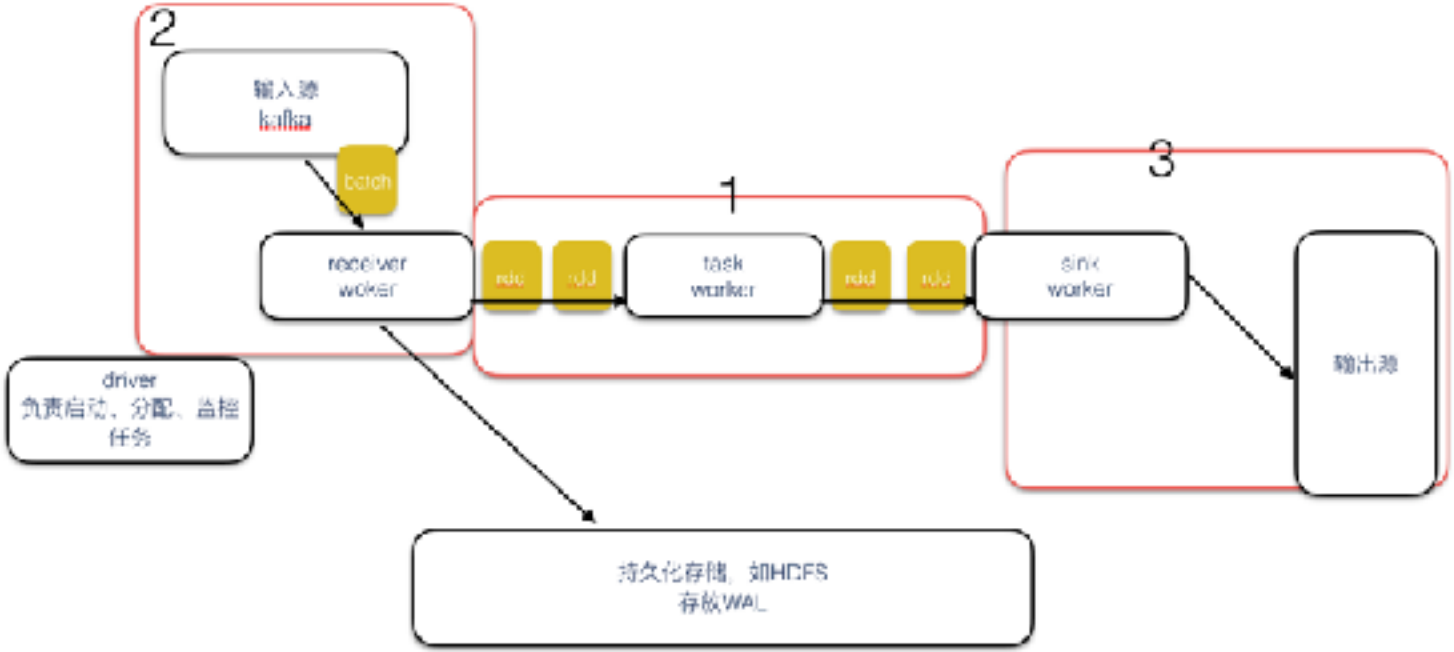
Spark streaming

吞吐



实时计算

exactly once

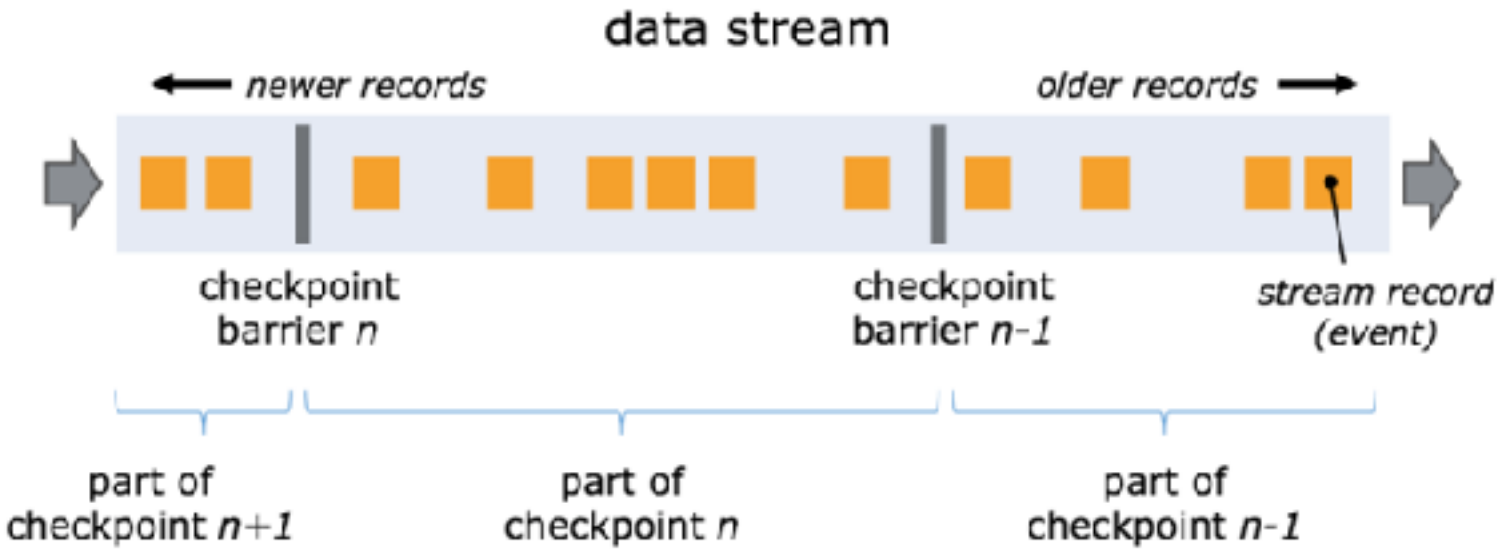


实时计算

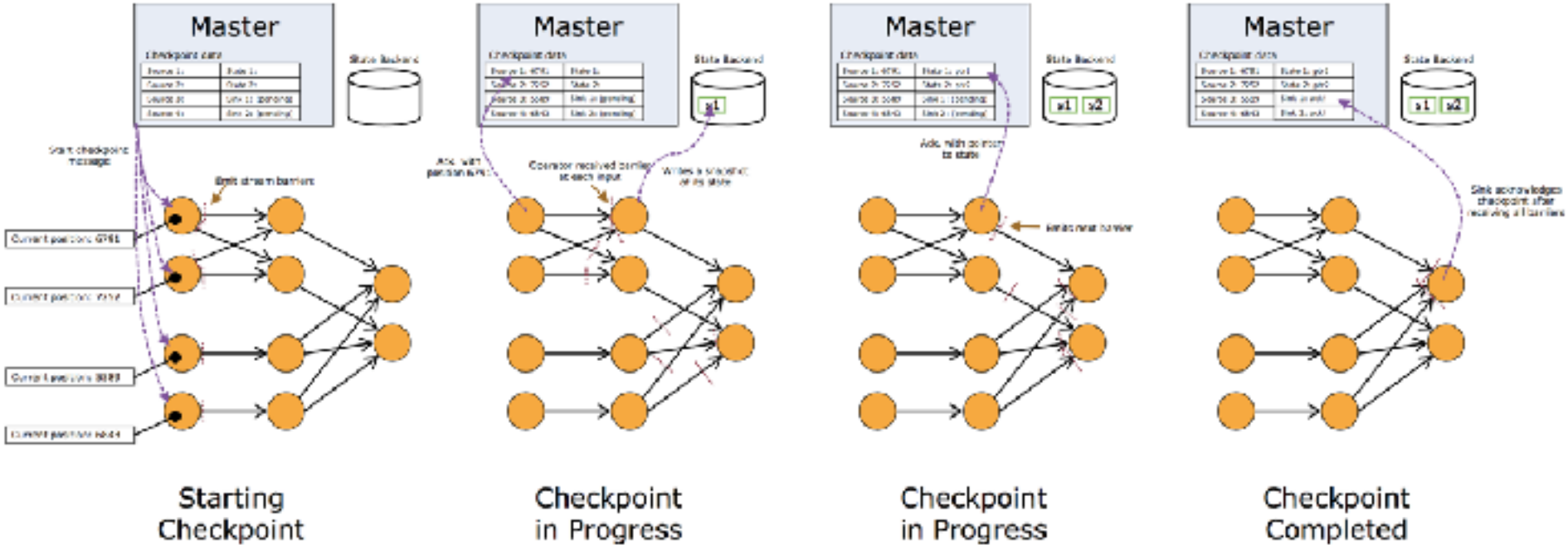
是否必须使用micro-batch的方式才能实现高吞吐和exactly once?

实时计算

Flink

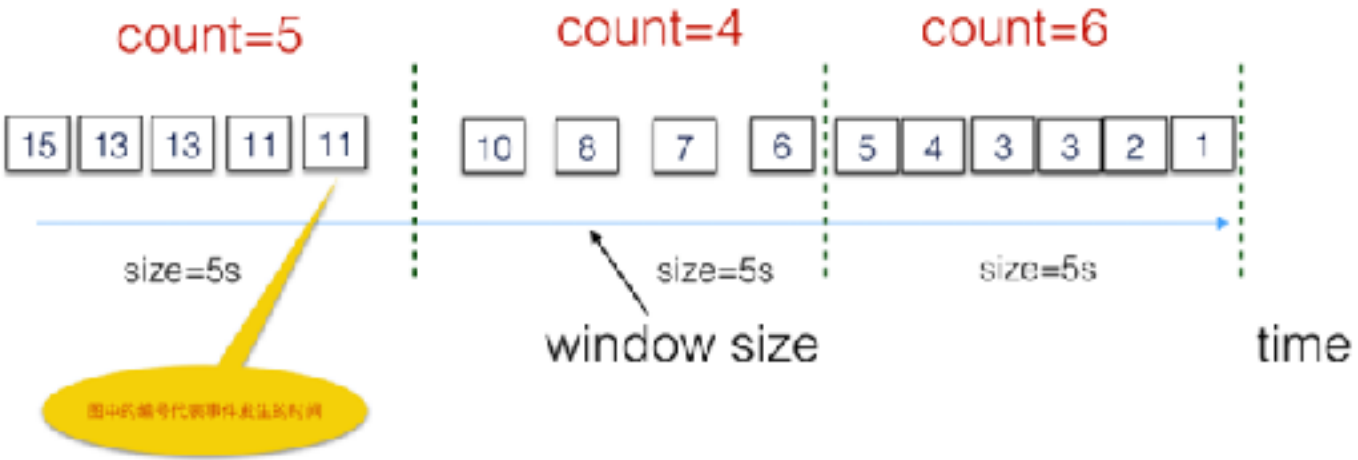

data stream
newer records
older records
checkpoint barrier n
checkpoint barrier n-1
stream record (event)
part of checkpoint n+1
part of checkpoint n
part of checkpoint n-1

实时计算


Master
State Backend
Starting Checkpoint
Checkpoint in Progress
Checkpoint in Progress
Checkpoint Completed

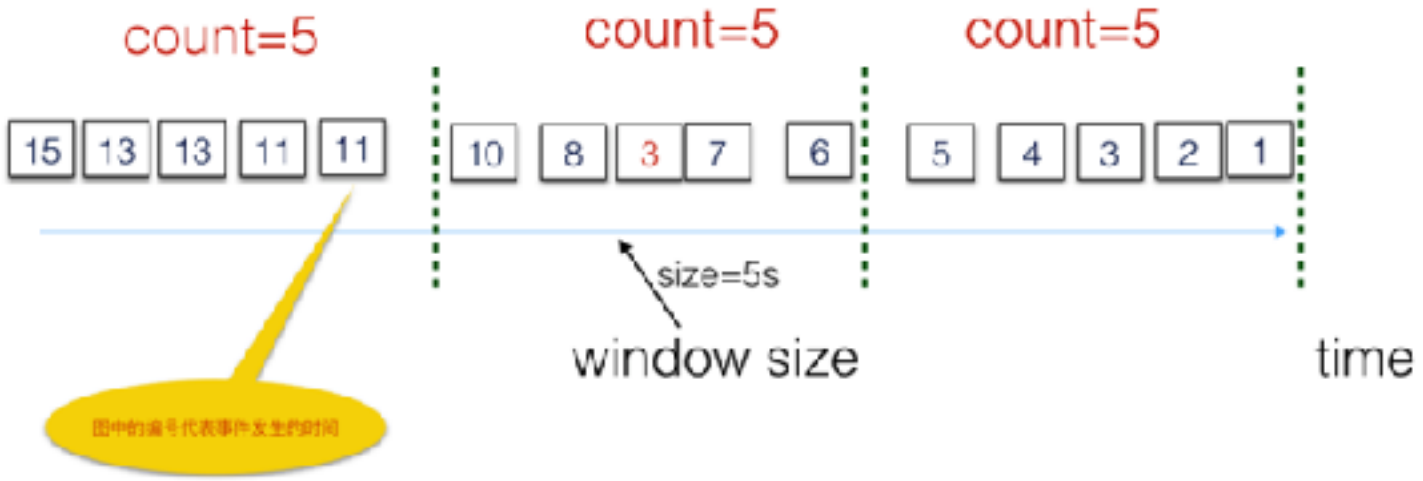
实时计算

采用window统计qps



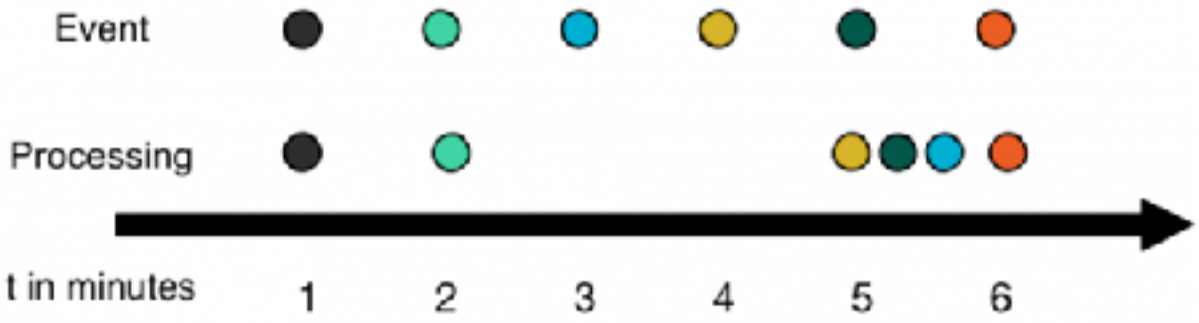
实时计算

数据乱序


count=5
count=5
count=5
15
13
13
11
11
10
8
3
7
6
5
4
3
2
1
size=5s
window size
time
图中的编号代表事件发生的时间

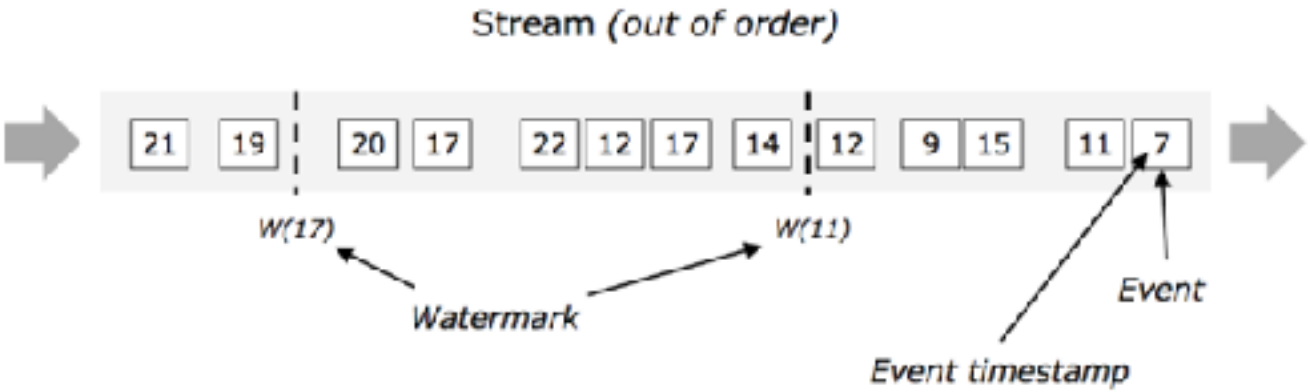
实时计算

event time： 数据真正产生的时间
process time： 系统处理该数据的时间



实时计算

Watermark



实时计算

	Storm	Spark Streaming	Flink
处理方式	streaming	micro-batch	streaming & batch
延迟	low	high	low
吞吐	low	high	high
exactly once	不支持	支持	支持
out of order	不支持	支持（2.0引入）	支持
API	Compositional	Declarative	Declarative

Thanks

关注开源数据库论坛