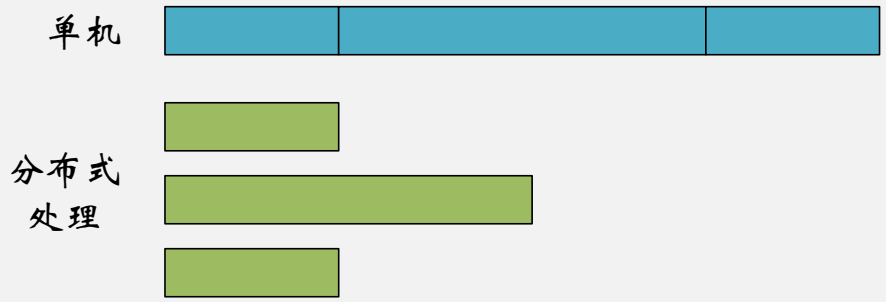
数据倾斜(Data Skew)

目录

1.背景

- 定义：
  - 数据倾斜是指数据的分布不均衡，不服从均匀分布，而是类似Zipf分布，导致hash等方式散列切分任务的不均衡，广泛存在与各个场景
- 影响：
  - 大数据处理系统shuffle时的长尾任务、OOM
  - 数据库系统中hash join性能退化
  - 限制分布式连接的性能扩展性



2.大数据系统数据倾斜常见解决方案

- 使用ETL预处理数据
- 过滤少数导致倾斜的key

预处理

- 提高shuffle操作的并行度
- 两阶段聚合（局部聚合+全局聚合）
- reduce join转为map join（数据广播）
- 采样倾斜key并分拆join操作
- 使用随机前缀和扩容RDD进行join

聚合与join

方案一、使用ETL预处理数据

- 场景：某数据表本身倾斜，业务场景需要频繁对该表进行分析
- 实现：预先对数据按照key进行聚合或join并存储，后续使用存储的预处理后的数据
- 优点：
  - 实现简单，完全规避到数据倾斜
  - 一次计算，多次复用
- 缺点：
  - 只是下推任务，ETL过程仍然存储数据倾斜
  - 倾斜表经常改变时，存储的预处理结果可能不满足增量可计算
    - 分布性聚合运算，全局性聚合运算

方案一、使用ETL预处理数据

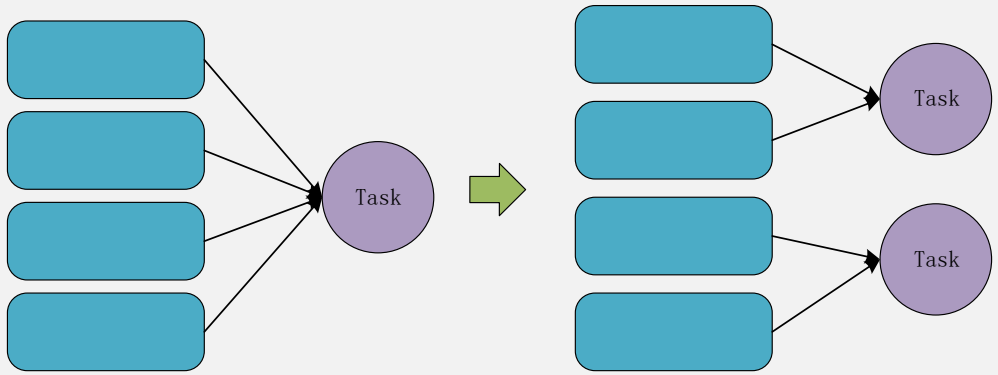
- 常见的聚合运算有sum、count、avg、max、min等
- **分布性聚合运算**：如果聚合运算的最新结果仅由它**存在的值**和**操作(插入/删除)的值**算出（插入操作为新值，删除操作为旧值），那么该聚合运算就是在该操作上（插入或删除）分布的
  - sum和count对插入操作和删除操作都是可分布
  - avg可由sumn和count推算，所以也是可分布
  - min和max对插入操作是可分布
- **全局性聚合运算**：如果计算聚合运算的最新结果所需要的存储空间没有固定的界，那么该聚合运算就是全局的
  - min和max对删除操作是全局的
- 增量可计算：对于插入和删除（更新=删除+插入）操作，聚合运算的新值能由它的原值和增量计算出，那么这个聚合运算就是增量可计算
  - sum,count,avg

方案二、过滤少数导致倾斜的key

- 场景：导致倾斜的key只少数几个，并且不影响最终结果
- 实现：采样取top-k，使用filter过滤倾斜key
- 优点：
  - 实现简单，完全规避掉倾斜
- 缺点：
  - 适用场景有限，倾斜的key数量限制

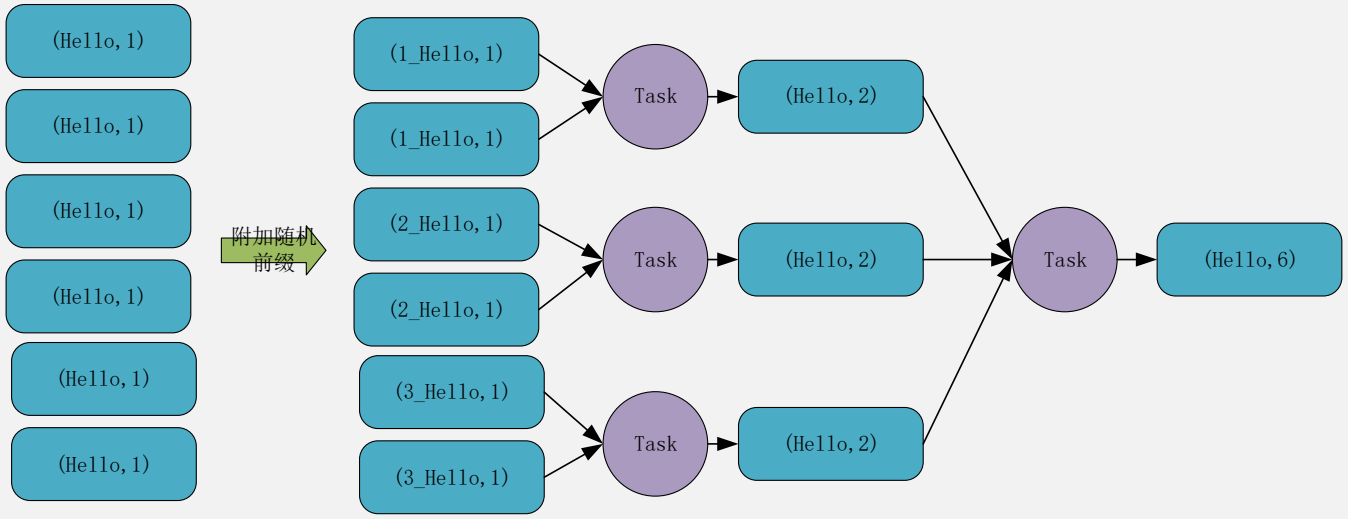
方案三、提高shuffle操作的并行度

- 场景：最基本的处理倾斜方法
- 实现：增加task数量，降低每个task的key的数量（堆资源）
- 优点：
  - 实现简单，可有效缓解数据倾斜影响
- 缺点：
  - 只是缓解，效果有限，资源消耗增加
  - 也无法解决单key倾斜问题



方案四、两阶段聚合（局部+全局聚合）

- 场景：reduce、groupby等的分组聚合
- 实现：二阶段聚合，首先給key打一个随机数标签（将原来一个task处理的key分散到多个task），进行局部聚合，然后去除标签，进行全局聚合
- 优点：
  - 对于聚合类shuffle效果较好
- 缺点：
  - 仅适用聚合，join类操作不适用
  - 增加了聚合次数

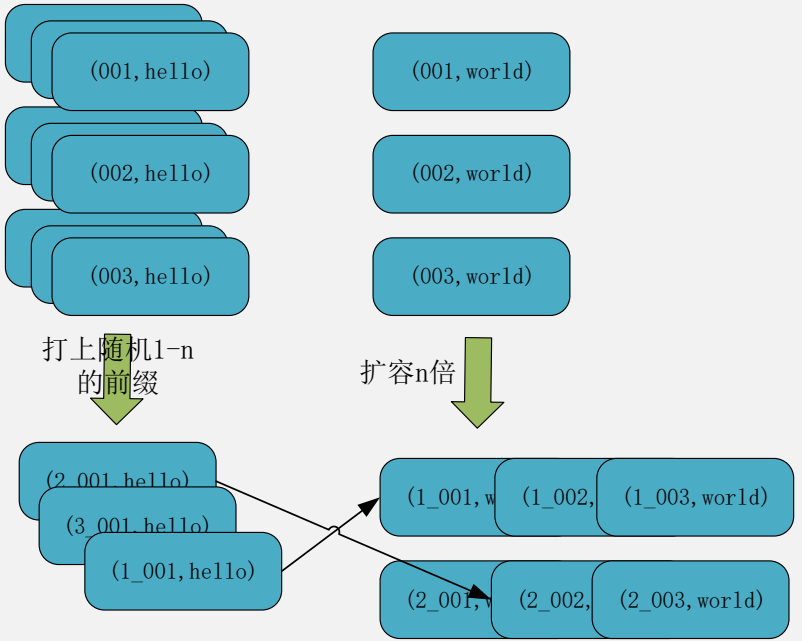


方案五、reduce join转为map join（数据广播）

- 场景：join时一个RDD较小（几百M，1G）
- 实现：替换join类算子，使用broadcast变量和map类算子，将本地RDD逐行按join key与小表的RDD的broadcast变量比较
- 优点：
  - 避免join产生的shuffle，避免发生数据倾斜
- 缺点：
  - 小表广播，增加内存消耗

方案六、采样倾斜key并分拆join操作

- 场景：其中某个RDD的少数key数据量过大，其余均匀，**单边点倾斜**
- 实现：利用分治思想，采样点倾斜的RDD，分出倾斜key，并打上n以内的前缀形成一个RDD，非倾斜的形成另一个RDD，不倾斜的join的RDD过滤倾斜key，膨胀为n条倍，顺序加上1-n的前缀，形成一个RDD，不倾斜的形成另一个RDD。两类RDD分别join，最后union结果
- 优点：
  - 内存消耗少
- 缺点：
  - 限制多，要求倾斜key少



方案七、随机前缀和扩容RDD进行join

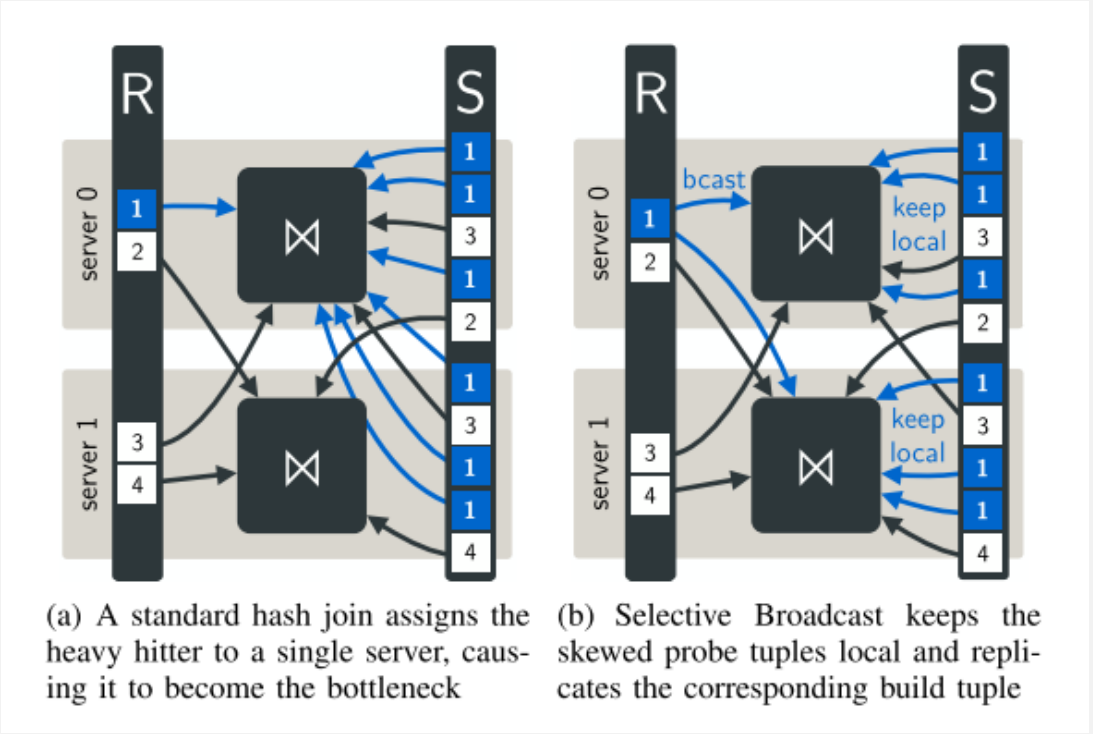
- 场景：单边倾斜
- 实现：将相对的大表的RDD每条数据打上随机1-n的前缀，将相对的小表扩容n倍
- 优点：
  - join类型倾斜的适用性强
- 缺点：
  - 内存资源消耗高

3.自适应数据倾斜分布式join：Flow-join

- Flow-Join: Adaptive Skew Handling for Distributed Joins over High-Speed Networks (ICDE 2016)
- 本文动机：
  - 高速网络使join倾斜的影响变得明显
- 之前方案未解决的问题：
  - 双边倾斜
  - 无统计信息
  - 并且非倾斜数据时，额外的分析代价（预先采样）

Flow join(主外键连接)

- 探测阶段
  - 根据1%的数据，检查（S表）是否存在倾斜的key(随着处理，更新倾斜key)，spaceSving算法统计频数，收集各节点上左表的频数统计，然后合并，检查阈值
- 识别并交换
  - 存在时，将R表中的对应的S表的倾斜key广播到所有的处理节点，构建hash桶，右表（探测表）的倾斜key保留在本地处理节点(避免网络开销)，进行处理，而非发送到根据key映射的处理节点集中进行处理。



(a) A standard hash join assigns the heavy hitter to a single server, causing it to become the bottleneck

(b) Selective Broadcast keeps the skewed probe tuples local and replicates the corresponding build tuple

好处：

1.倾斜元组在本地，减少网络传输量

2.避免了单个服务器计算负载过高

倾斜key的检测

- 传统方法：
  - 统计信息，对中间结果估计不准，大数据集也无能为力
  - 实时预计算，增加额外代价
- SpaceSaving:
  - 最好的软件方案计算

  数据流中频繁项和Top-k算法
  - spaceSaving误差<=1/k

- 倾斜判断
  - 阈值
  - 1%前，每个工作线程决定
  - 1%后，达成全局共识
  - 连接过程定期重复

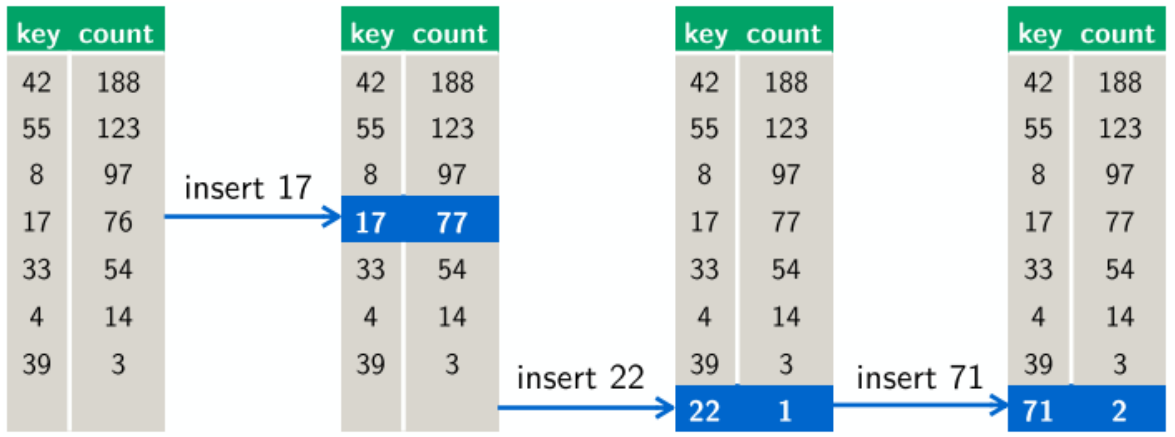
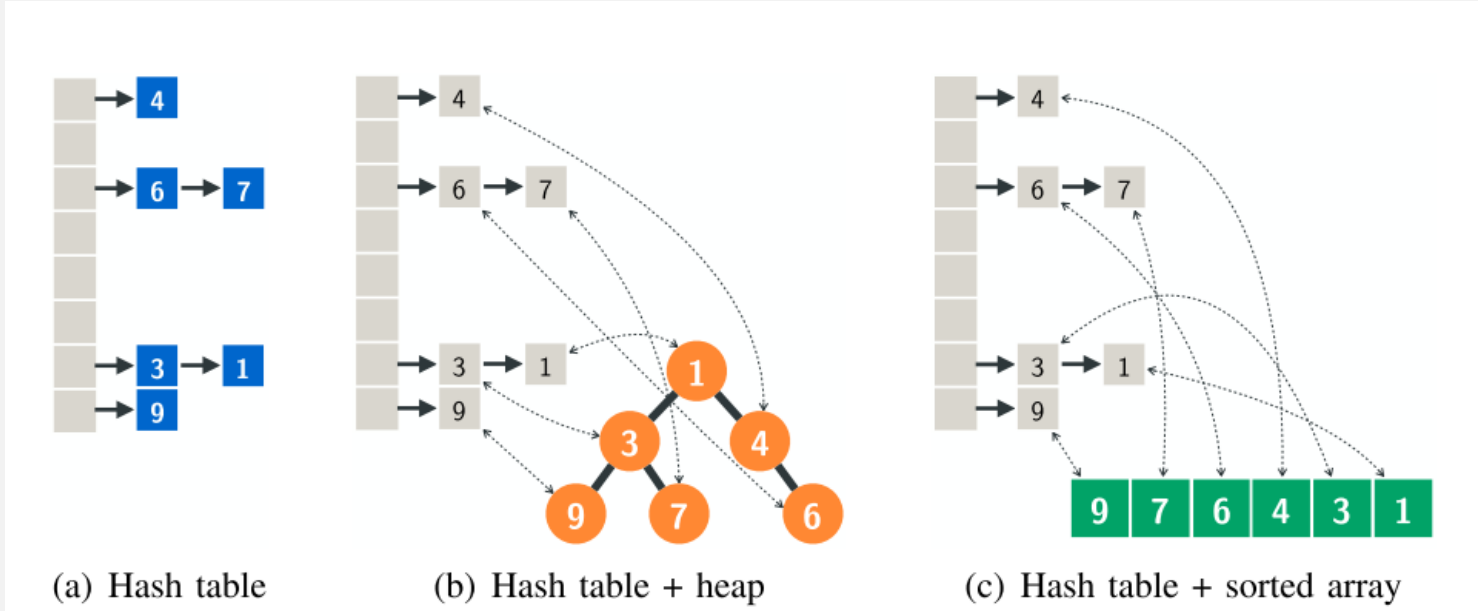
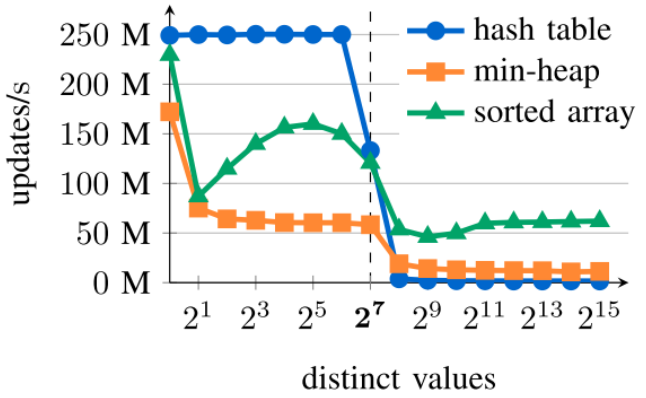


Figure 7. Example for the SpaceSaving algorithm using capacity k = 8 and replacing the entry with the minimum count once the histogram is full

SpaceSaving的数据结构

- 操作：
  - 更新计数
  - 删除最小值
- 三种方案：
  - Hash表O(K)
  - Hash表+堆O(logK)
  - Hash表+有序数组(O(1))
- 更多方案
  - 并行（基于比较树，FPGA）
  - 流水线（最多同时处理k/2个项）

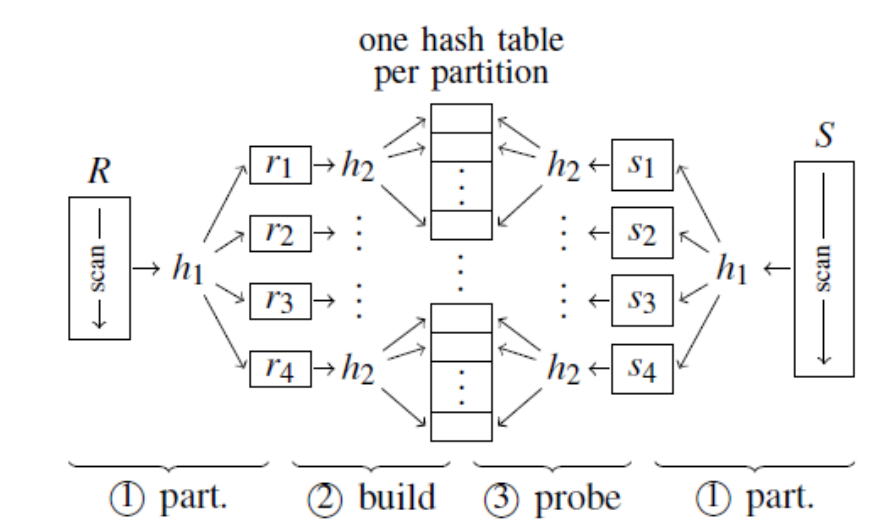




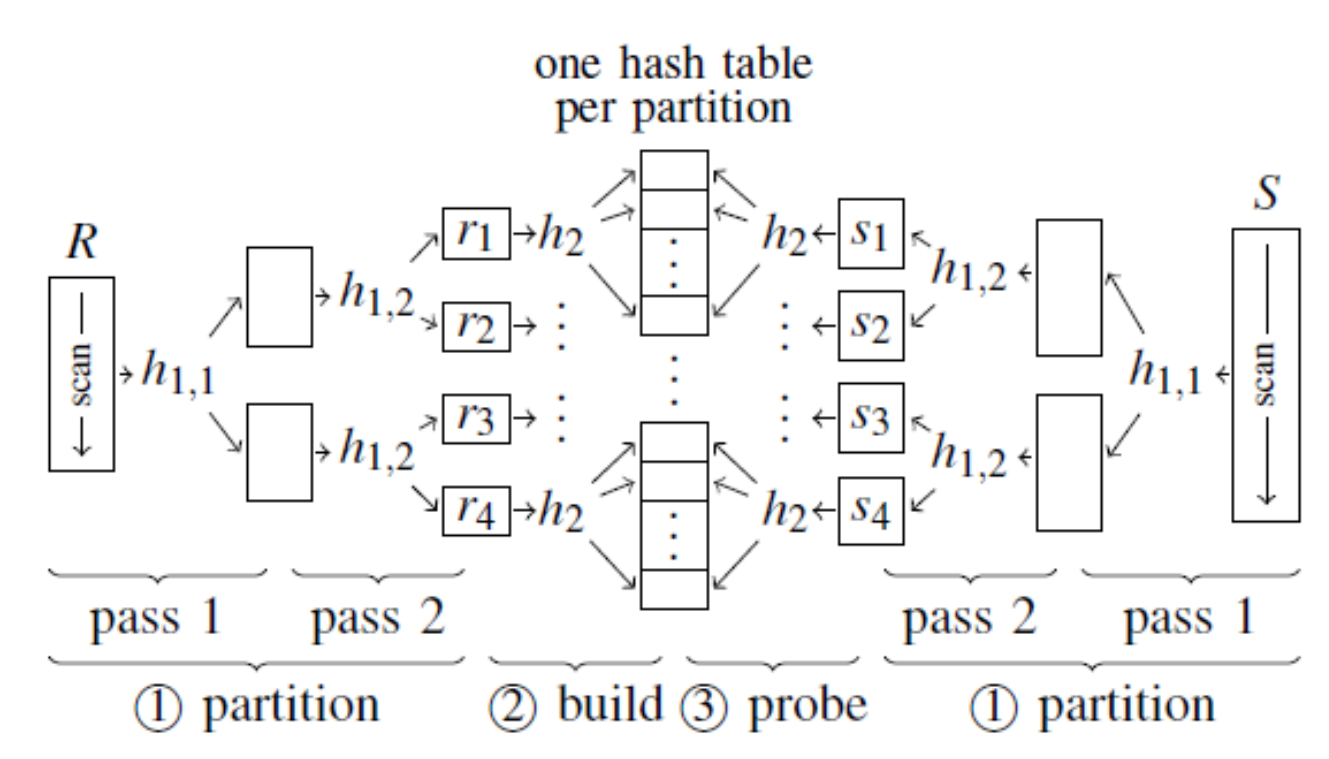
(d) Single-threaded update rate, $k = 128$

分布式连接算法：Radix Hash Join

- Radix hash join是一种**Hardware-conscious hash join**
- Radix hash join在partition阶段对数据进行多趟分区，每趟分区操作限制分区的数量
- 缓解了Probe阶段cache丢失问题
- 增加线程同步代价



Partitioned hash join



Radix hash join

Generalized Flow join

- 基本思想：对于两表都存在倾斜key时，广播R、S表的倾斜key（且非S、R表倾斜key），对称片段复制(SFR)处理R、S相同的倾斜key
- 流水线探测步骤:
  - 交换R：对R中的每个元组
    - 更新近似直方图
    - 检查是否满足倾斜阈值
      - 倾斜：插入本地hash表
      - 否则：根据hash发送到目标服务器
    - 创建全局的R的倾斜key
  - 交换S：对S中每个元组
    - 更新近似直方图
    - 检查是否满足倾斜阈值

Generalized Flow join

        - 在S中倾斜：物化S元组，因为此时不广播相应的R元组
        - 在R（但不是S）中倾斜：向所有服务器广播S元组
        - 否则：将S元组发送到目标服务器
    - 创建S的全局倾斜key
- 处理倾斜
    - 广播与S中的倾斜key连接的R元组(在R中不是倾斜key)，连接相应的物化S元组
    - 通过对称片段复制（SFR）方法重新分配其连接键在R和S中都是倾斜key的元组

Generalized Flow join

对称分段复制SFR

- 假设R和S上在每台服务器上都有x和y行的倾斜key
  - 小表广播：传输*n(n-1)x*的传输数据量，造成网络拥塞
  - 单点连接：传输*(n-1)×(x+y)*的数据量，长尾任务
- SFR：指将服务器逻辑组织呈一个*n1×n2*的矩形*(n1×n2=n*,n为服务器的数量)，然后将R和S上的倾斜key分别同行同列之间进行广播
  - 减少全域广播某一个小表R时导致的网络拥塞
  - 同时也避免单点处理的时间瓶颈
  - 类似计算机体系结构中矩阵乘法分块并行处理的技术

对称分段复制SFR

- Server0= $R_{0,3,6} \uplus S_{0,1,2}$
- Server3= $R_{0,3,6} \uplus S_{3,4,5}$
- Server6= $R_{0,3,6} \uplus S_{6,7,8}$
- All= $R_{0-8} \uplus S_{0-8}$

- 性能：
  - 数据量：$n((n1-1)x+(n2-1)y)$
  - 最好：$n1=n2$ 性能提升 $\frac{\sqrt{n}+1}{2}$ 倍

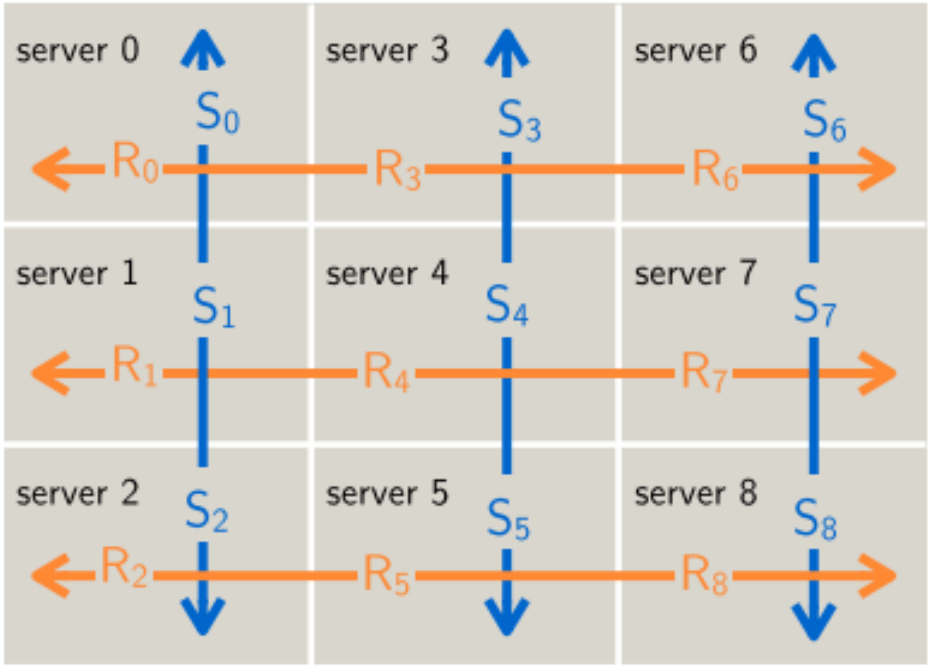


Figure 10. The Symmetric Fragment Replicate redistribution scheme logically organizes the servers of the cluster in a rectangle; heavy hitter tuples are replicated across rows for one input and across columns for the other input

参考：

- 美团技术团队：Spark性能优化指南——高级篇 https://tech.meituan.com/2016/05/12/spark-tuning-pro.html
- Rodiger, W., Idicula, S., Kemper, A., & Neumann, T. (2016). **Flow-Join: Adaptive skew handling for distributed joins over high-speed networks**. *2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016*, 1194–1205.
- https://gitee.com/daseATecnu/bds2017/wikis/Parallel-Hash-Join-%231?sort_id=157275
- https://gitee.com/daseATecnu/bds2017/wikis/Parallel-Hash-Join-%232?sort_id=157271