第三届阿里中间件性能挑战赛比赛攻略

穿山甲团队 成员：杨晓 徐哲 张蜀男

一 赛题背景分析及理解

赛题背景

题目主要解决的是数据同步领域范畴：实时增量同步，主要的技术挑战为模拟数据库的主备复制，提供"高效"的实时同步能力

关键信息:

1 模拟流式数据处理，只允许单线程读日志文件
2 高效地实现数据的增删改查
3 重做过程并行化

整个程序分为两个阶段

重建阶段，根据日志重建数据
收集阶段，进行数据查询

问题的关键

1 充分利用多核性能
2 高效地组织数据

如何充分利用多核性能?

1 并行解析
2 并行重建，需要解决主键 update 问题

二 核心思路

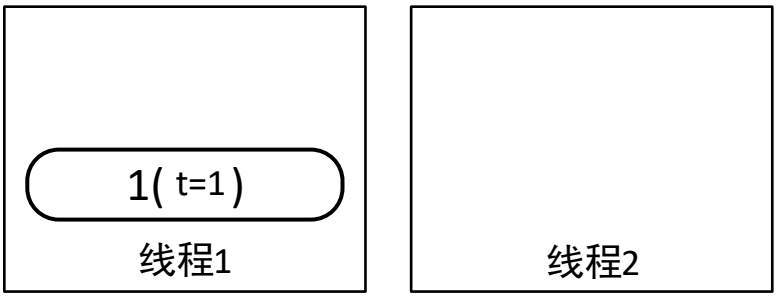
多版本并行重建

主键 update 的过程中，线程 B 需要获取旧记录，但是后面的增删改操作其实并不依赖旧记录，只有查询的时候需要访问旧记录
方案:
1 根据 id 的哈希，分发给不同线程处理
2 对每次主键 update，我们并不需要得到旧记录，只需要保证**查询时可以找到旧记录**
3 每次主键 update，为旧 id 生成一个新的数据版本
4 如何找到需要的版本？比较**时间戳**

算法图解

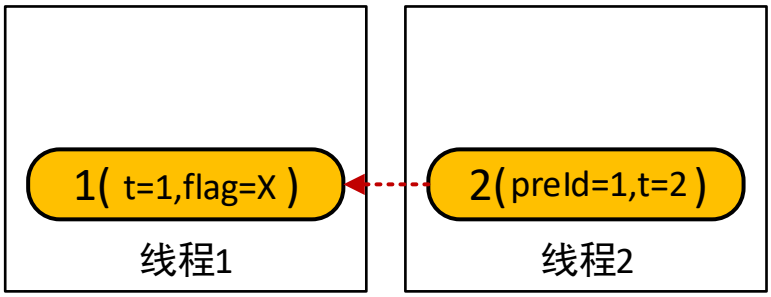
1 插入 id=1
2 更新 id: 1->2
3 插入 id=1
4 查询 id=2

1 插入 id=1
线程 A 添加一条记录，时间戳为 1



a) 插入1

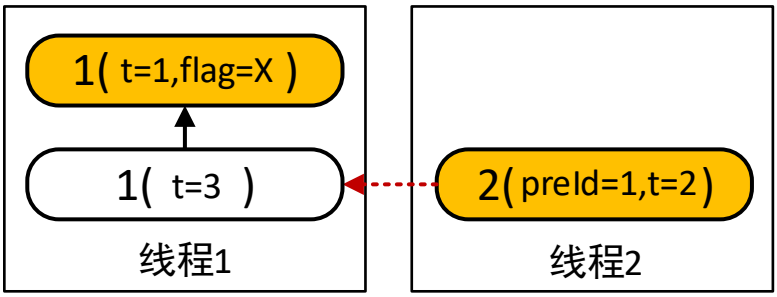
2 更新 id: 1->2
线程 B 添加一条记录，并记录时间戳 Time 和旧主键 PreID
将线程 A 的旧记录标记为 X，表示已经被 update 到其他 id



b) 1更新到2

3 插入 id=1

线程 A 新加一条记录，添加一个新的数据版本

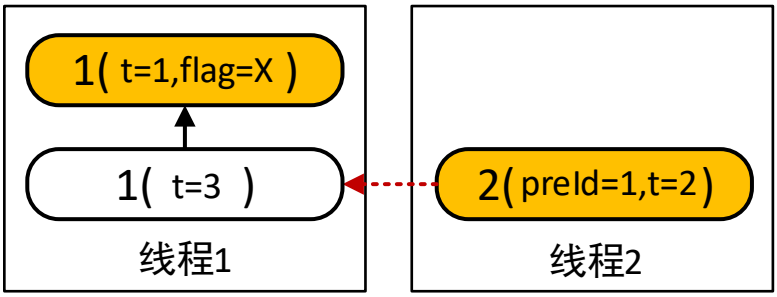


c) 插入新的1

4 查询 id=2

先去在线程 B 中查找 id=2，然后根据 PreID 和时间戳找到前一条记录

不断重复这个过程，直到得到所有列的数据



算法分析

优势:

重建过程**完全并行**，重建线程之间不需要任何同步

劣势:

保存多个版本需要更多的存储空间

查询阶段需要追溯多个数据版本

存储结构

两层存储

数据的存储分为两层
1 id 小于 8M 的记录，使用固定位置存储
2 id 大于 8M 的记录，动态分配空间，哈希表记录 id 到存储地址的映射

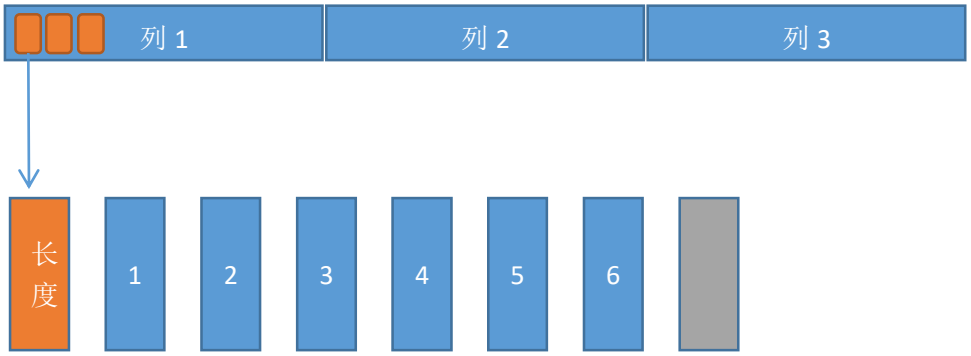
对于小于 8M 的记录，避免了哈希表的开销

列存储

将列数据连续存储
更新操作批量对某列进行操作，列存储 cache 命中率更高

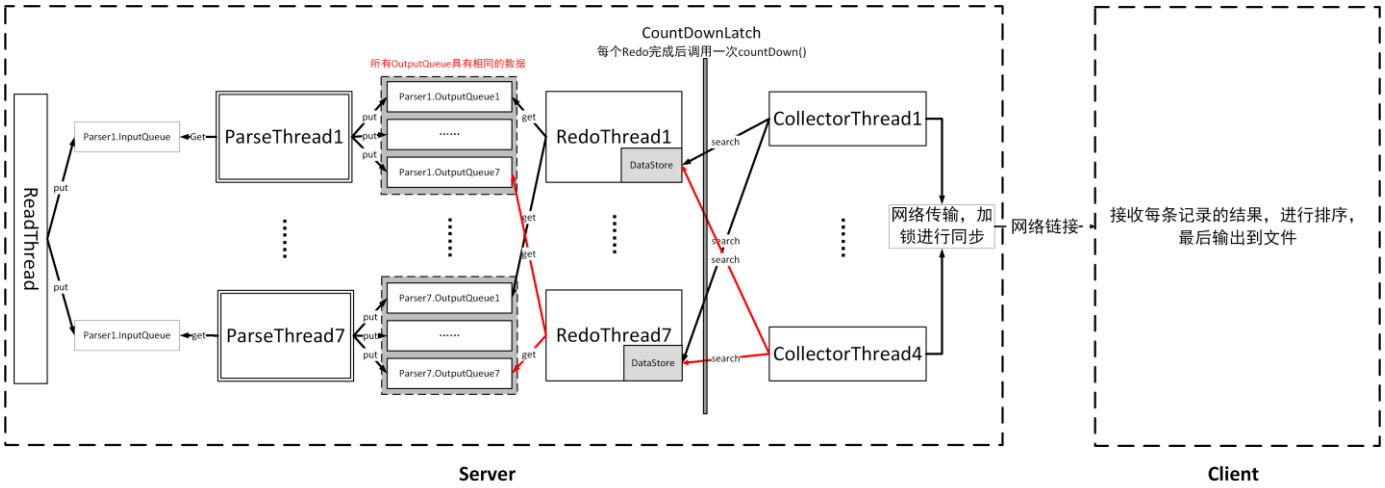
列数据用一个 long 存储

1 个字节存长度, 6 个字节存数据
直接读写 long 比操作 8 个字节速度更快

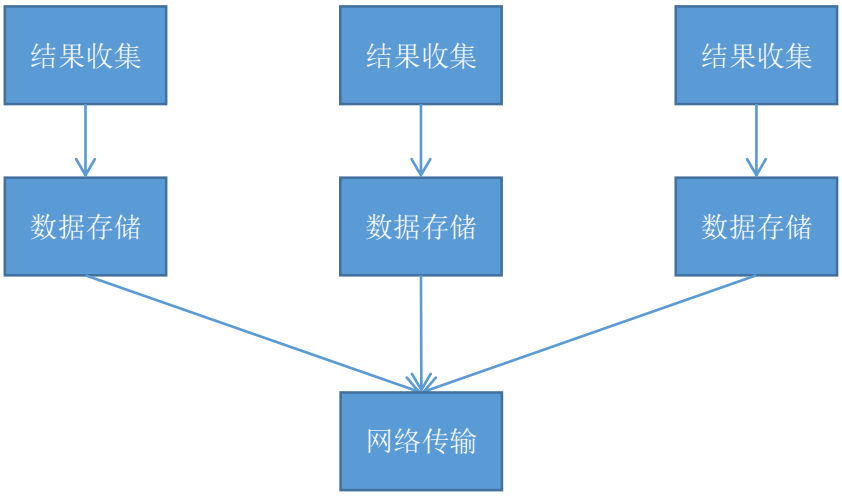


四 整体架构

重做阶段:
单线程读取,多线程解析,多线程重做



收集阶段:
多线程并行收集，分别发送给客户端
最后由客户端进行排序，写入文件



五 代码优化

利用堆外内存快速解析

使用堆外内存可以避免拷贝数据到 jvm 内存，同时配合使用 Unsafe.getByte 访问 Direct Buffer
单核读取 1G 的文件，使用堆内存需要 550ms，使用 Direct Buffer 仅需 250ms
效果:重建速度从 4.2s 优化到 3.2s

避免 Full GC

使用对象池进行内存复用
整个运行过程只有 1 次 gc

使用不装箱的 HashMap

避免原生 HashMap 的装箱开销，速度更快，内存更少
插入 150W 的<long,int>对，原生 HashMap 消耗 710ms，120M 内存，优化后的哈希表仅使用 170ms, 31M 内存

六 关键代码

处理更新操作

1 如果没有主键更新，只需要更新数据
2 如果有主键更新，需要保存旧版本数据，并保存旧主键和时间戳

```
if (opType == 'U') {
    if (preId != id) {//主键更新
        //获取数据的存储地址
        int next = hashing.getOrDefault(id, 0);
        //分配存储空间
        int newNode = allocateBlock();
        //保存旧版本存储地址
        writeLongToLevel2(OFF_NEXT + newNode, next);
        //保存时间戳
        writeLongToLevel2(OFF_SEQ + newNode, logBlock.seqs[logPos]);
        //保存旧主键
        writeLongToLevel2(OFF_PREID + newNode, preId);
        //保存旧版本数据
        writeLogDataToLevel2(newNode + OFF_CELL, colData, colDataPos,
colDataLen);
        //保存新的存储地址
        hashing.put(id, newNode);
    } else {
        int node = hashing.getOrDefault(id, 0);
        //更新数据
        writeLogDataToLevel2(node + OFF_CELL, colData, colDataPos,
colDataLen);
```

```
    }
}
```

根据时间戳进行查询某个版本的数据

在数据版本链表上循环查找，直到时间戳小于查询值

```
while (node != 0) {
    //获取时间戳
    long nodeSeq = readLong(node + OFF_SEQ);
    //找到时间戳小于 seq 的记录
    if (nodeSeq >= seq) {
        int next = (int) readLong(node + OFF_NEXT);
        node = next;
    } else {
        break;
    }
}
```

查询某个 id 的最终结果

遍历主键 update 形成的数据链，直到查到所有列数据：

```
//当前记录的时间戳
long lastSeq = recordData.seq;
//旧主键
long preId = recordData.preid;
//直到获取所有列数据为止，cellCount 是列数量
while (colCount != cellCount && preId != -1) {
    data = getDataMap(preId);
    //根据旧主键和时间戳查询数据
    RecordData preRecord = data.getRecord(preId, lastSeq);
    for (int i = 0; i < cellCount; i++) {
        //更新列值
        if ((datas[i] & 0xff) == 0 && (preRecord.colData[i] & 0xff) != 0){
            datas[i] = preRecord.colData[i];
            colCount++;
        }
    }
    //更新旧主键和时间戳
    preId = preRecord.preid;
    lastSeq = preRecord.seq;
}
```

日志解析:获取列数据

使用 unsafe 直接读取堆外内存，结果存入一个 long

```
//获取列值
private long nextColValue() {
    int old = parsePos;
    //使用一个 Long 存储列值
    colValue = 0;
    //使用 unsafe 直接读取堆外内存
    while (unsafe.getByte(buffAddr + parsePos) != '|') {
        //写入一个字节
        colValue = colValue << 8 | ((long) unsafe.getByte(buffAddr + parsePos)
& 0xff);
        parsePos++;
    }
    //写入数据长度
    colValue = colValue << 8 | ((long) (parsePos - old) & 0xff);
    //跳过分隔符
    parsePos++;
    return colValue;
}
```

七 总结与感想

我们最终在复赛中取得了第三名的成绩，对于我们来说，是一个非常满意的成绩。下面针对我们在整个程序开发中的收获做如下总结。

在整个程序设计中，每一个细节都应该认真对待。我们在实现操作日志的重做线程归属计算，这一条简单语句是放在解析线程还是放在重做线程中上，就经过了反复测试。另外，很重要的一个收获就是，实践为王，很多理论上性能更好的设计，在实际测试中效果并不理想。因此，我们在设计每一个重要功能时，都针对多种方案进行了反复性能测试。这也另我们在工程实现上收获了大量宝贵经验。

通过参加这次比赛，我们队伍每一个人都收获颇多。不但收获了技术上的进步，也培养了自己攻克难题的不懈精神。更重要的是，通过这场比赛，我们三人成了好朋友、兄弟。遇到问题一起讨论、交流，头脑风暴式的辩论，这才是在这次比赛中无可替代的收获。感谢这次比赛的举办方阿里巴巴公司及其中间件团队，希望这个比赛越办越好。