海量数据处理经验分享

队名：yfy　成员：俞飞樾

2016.8

目录

- 概述
- 解决方案
- 改进过程
- 难点解决
- 亮点展示

需求

- 针对百G级别订单信息，商品信息，买家信息，不得使用现有数据库代码，实现各种查询
- 提供交易ID，查询某次交易的某些属性
- 查询某位买家某个时间范围内的所有交易信息
- 查询某位买家某个时间范围内的所有交易信息
- 对某个商品的所有交易信息进行求和

完成

- 索引构建
- 单条记录查询
- 范围查询
- 查询排序
- 查询求和
- join功能
- 支持并发查询

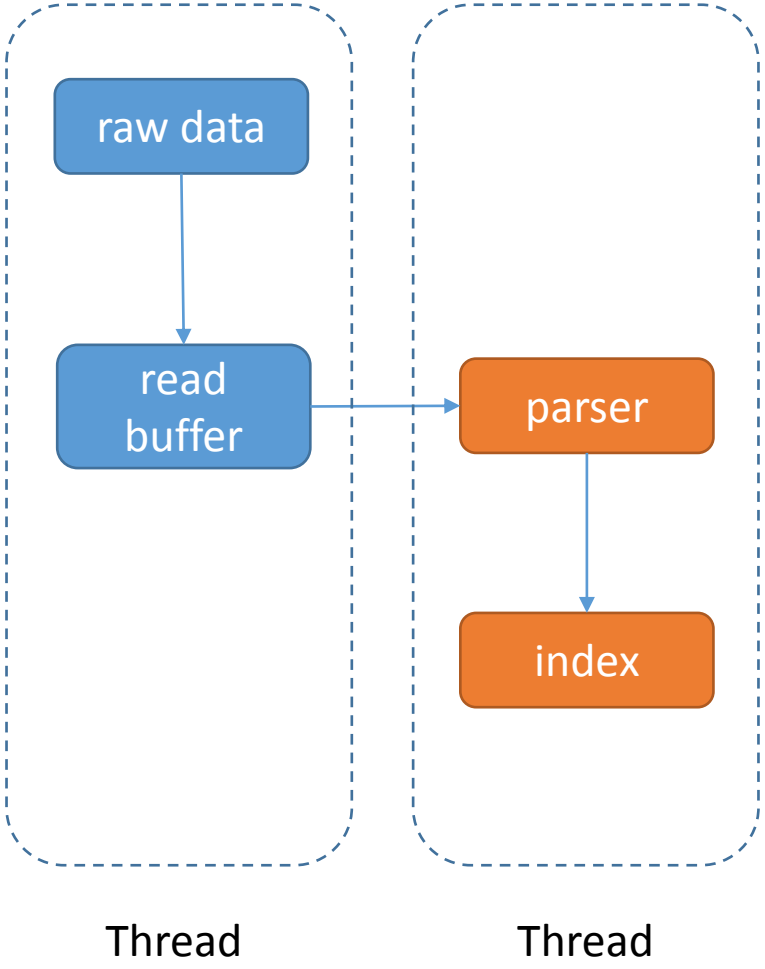
结果

- 1946支参赛队
- 第7名
- 至8月2日最好成绩，建立索引2955122ms，一小时查询量181238次，查询平均响应时间19.86ms

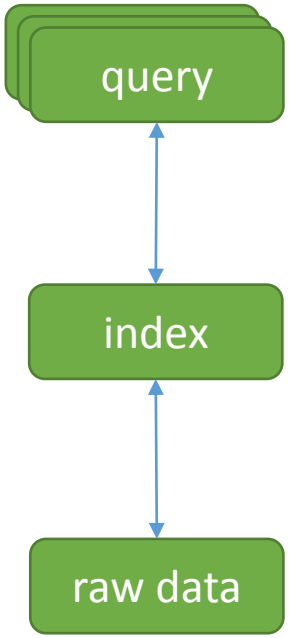
目录

- 概述
- 解决方案
- 改进过程
- 难点解决
- 亮点展示

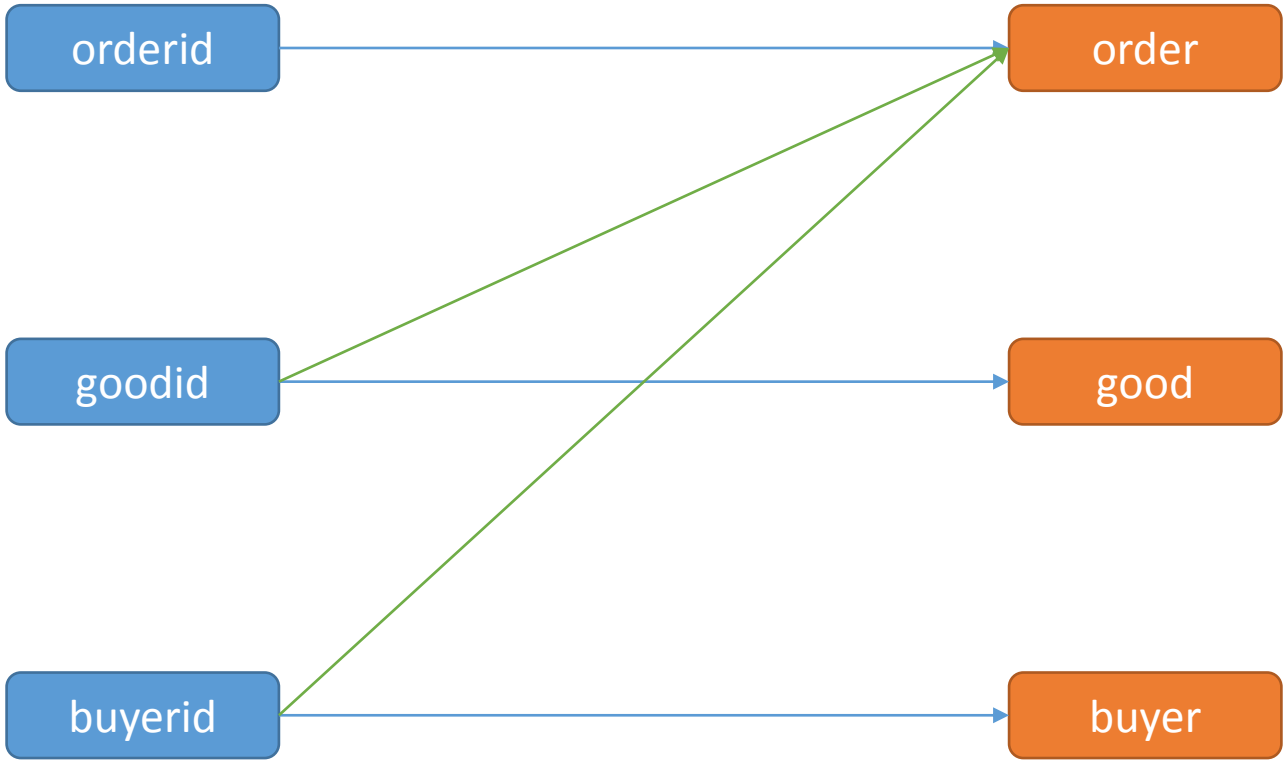
构建


raw data
read buffer
parser
index
Thread
Thread

查询

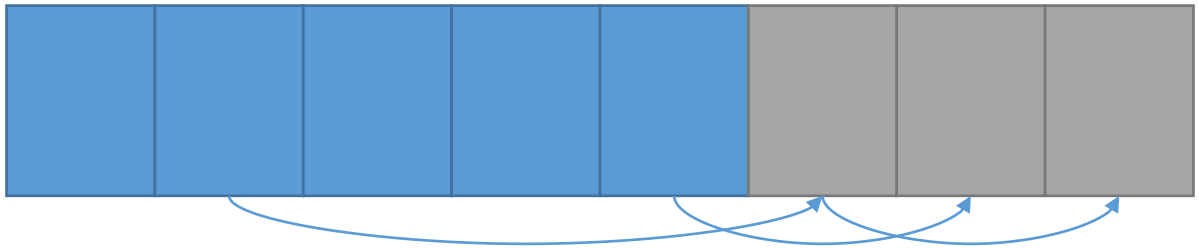

query
index
raw data

索引设计



索引设计

- 3个主键索引，o2o，g2g，b2b
- 2个一对多的索引，g2o，b2o
- 全部使用散列表存储
  - 简单，易于实现
  - 期望时间复杂度是O(1)
  - 数据量已知



索引设计原则

- 单个磁盘顺序访问
  - 测试试跑数据索引随机写磁盘，十几分钟
  - 先放内存，再写磁盘，放不下则拆分索引
- 只用byte数组存储
  - 商品表和买家表索引，HashMap，2G，自己实现，500M
- 压缩索引大小
  - 根据数据特点，减少无用的byte消耗

构建过程

- order表读3遍，
  - o2o, entrySize = 10, key, fileId, fileOff
  - g2o, entrySize = 5, fileId, fileOff
  - b2o, entrySize = 5, fileId, fileOff
- good表
  - g2g, entrySize = 29, key, fileId, fileOff, keyId
- buyer表
  - b2b, entrySize = 29, key, fileId, fileOff, keyId

目录

- 概述
- 解决方案
- 改进过程
- 难点解决
- 亮点展示

索引合并

- 因内存有限，故对索引进行了拆分
  - 查询时可能访问多个部分索引
- 后发现g2o, b2o可以不拆分，但需多次读订单表，需牺牲查询平均访问次数，提高载入率，溢出桶增加
- 7万 -> 9万

数据重组

- 范围查询，访问大量随机订单数据
- 存在热点数据，将已读入的订单写在一起，下次查询可一次读出
- b2o，9万 -> 10万
- g2o，10万 -> 11万
- 同时读写，一致性，不需额外加锁，普通查询同步机制，保证索引修改原子性
- 实现并不完整，粗略实现存在冗余操作，提升不理想

索引位置

- 堆内
  - g2g, b2b, 500M
- 堆外（新增）
  - (½)o2o, 2.7G
- 磁盘
  - 剩余索引
- 11万 -> 18万
- 对缓存的判断战略失误，（收益/成本）远高于预期，这是倒数第2次提交

目录

- 概述
- 解决方案
- 改进过程
- 难点解决
- 亮点展示

构建过程

- 11号到29号，一直在构建
- 失败的尝试
  - 读，处理，写，单线程
  - 索引分成很多小文件
    - 经测试，大文件速度慢
  - 写缓冲，请求有序排放，开线程一直写，复杂的同步
    - 请求中保存数据，gc导致cpu满载
    - 写缓冲每个桶自带缓冲，局部填满导致全局等待
    - 读写冲突，非顺序访问
- 成功的尝试
  - 索引设计原则
  - 回归简单

目录

- 概述
- 解决方案
- 改进过程
- 难点解决
- 亮点展示

亮点展示

- 自己实现的HashMap比Java自带的在本项目benchmark上好很多，1/4空间消耗
- 实现简单，无第三方类库，1748行代码
- 单人承包所有工作
- 测试驱动，高质量的测试代码，自有成绩以来，所有改动均正确
- 构建快速，稳定3000s以内
- 代码结构良好，自如使用接口，抽象类，继承等特性，代码复用性好，可扩展

Q&A