基于 MySQL Binlog 的 ElasticSearch 数据同步实践

背景

随着马蜂窝的逐渐发展，我们的业务数据越来越多，单纯使用 MySQL
已经不能满足我们的数据查询需求，例如对于商品、订单等数据的多维度检索。

使用 Elasticsearch 存储业务数据可以很好的解决我们业务中的搜索需求。而数据进行异构存储后，随之而来的就是数据同步的问题。

现有方法及问题

对于数据同步，我们目前的解决方案是建立数据中间表。把需要检索的业务数据，统一放到一张 MySQL 表中，这张中间表对应了业务需要的 Elasticsearch
索引，每一列对应索引中的一个Mapping 字段。通过脚本以 Crontab 的方式，读取 MySQL
中间表中 UTime 大于上一次读取时间的所有数据，即该段时间内的增量，写入Elasticsearch。

所以，一旦业务逻辑中有相应字段的数据变更，需要同时顾及 MySQL 中间表的变更；如果需要 Elasticsearch 中的数据即时性较高，还需要同时写入 Elasticsearch。

随着业务数据越来越多，MySQL 中间表的数据量越来越大。当需要在 Elasticsearch
的索引中新增 Mapping 字段时，相应的 MySQL
中间表也需要新增列，在数据量庞大的表中，扩展列的耗时是难以忍受的。

而且 Elasticsearch 索引中的 Mapping 字段随着业务发展增多，需要由业务方增加相应的写入 MySQL 中间表方法，这也带来一部分开发成本。

方案设计

整体思路

现有的一些开源数据同步工具，如阿里的 DataX 等，主要是基于查询来获取数据源，这会存在如何确定增量（比如使用utime字段解决等）和轮询频率的问题，而我们一些业务场景对于数据同步的实时性要求比较高。为了解决上述问题，我们提出了一种基于 MySQL Binlog 来进行 MySQL 数据同步到 Elasticsearch 的思路。Binlog 是 MySQL 通过 Replication
协议用来做主从数据同步的数据，所以它有我们需要写入 Elasticsearch
的数据，并符合对数据同步时效性的要求。

使用 Binlog 数据同步 Elasticsearch，业务方就可以专注于业务逻辑对 MySQL
的操作，不用再关心数据向 Elasticsearch
同步的问题，减少了不必要的同步代码，避免了扩展中间表列的长耗时问题。

经过调研后，我们采用开源项目 go-mysql-elasticsearch
实现数据同步，并针对马蜂窝技术栈和实际的业务环境进行了一些定制化开发。

数据同步正确性保证

公司的所有表的 Binlog
数据属于机密数据，不能直接获取，为了满足各业务线的使用需求，采用接入 Kafka
的形式提供给使用方，并且需要使用方申请相应的 Binlog
数据使用权限。获取使用权限后，使用方以 Consumer Group 的形式读取。

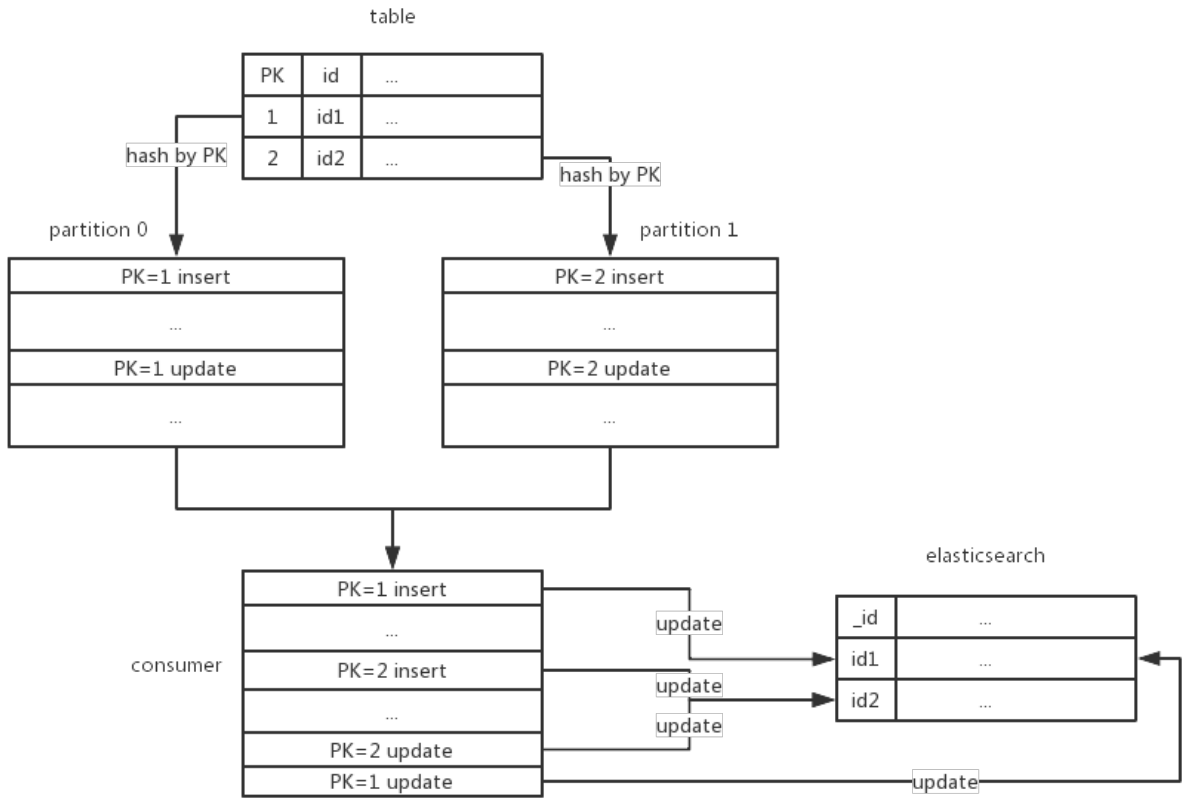
这种方式保证了 Binglog 数据的安全性，但是对保证数据同步的正确性带来了挑战。因此我们设计了一些机制，来保证数据源的获取有序、完整。

顺序性

通过 Kafka 获取 Binlog 数据，首先需要保证获取数据的顺序性。严格说，Kafka
是无法保证全局消息有序的，只能局部有序，所以无法保证所有 Binlog 数据都可以有序到达 Consumer。

但是每个 Partition 上的数据是有序的。为了可以按顺序拿到每一行 MySQL 记录的
Binglog，我们把每条 Binlog 按照其 Primary Key，Hash 到各个 Partition 上，保证同一条 MySQL 记录的所有 Binlog 数据都发送到同一个 Partition。

如果是多 Consumer 的情况，一个 Partition 只会分配给一个 Consumer，同样可以保证
Partition 内的数据可以有序的 Update 到 Elasticsearch 中。

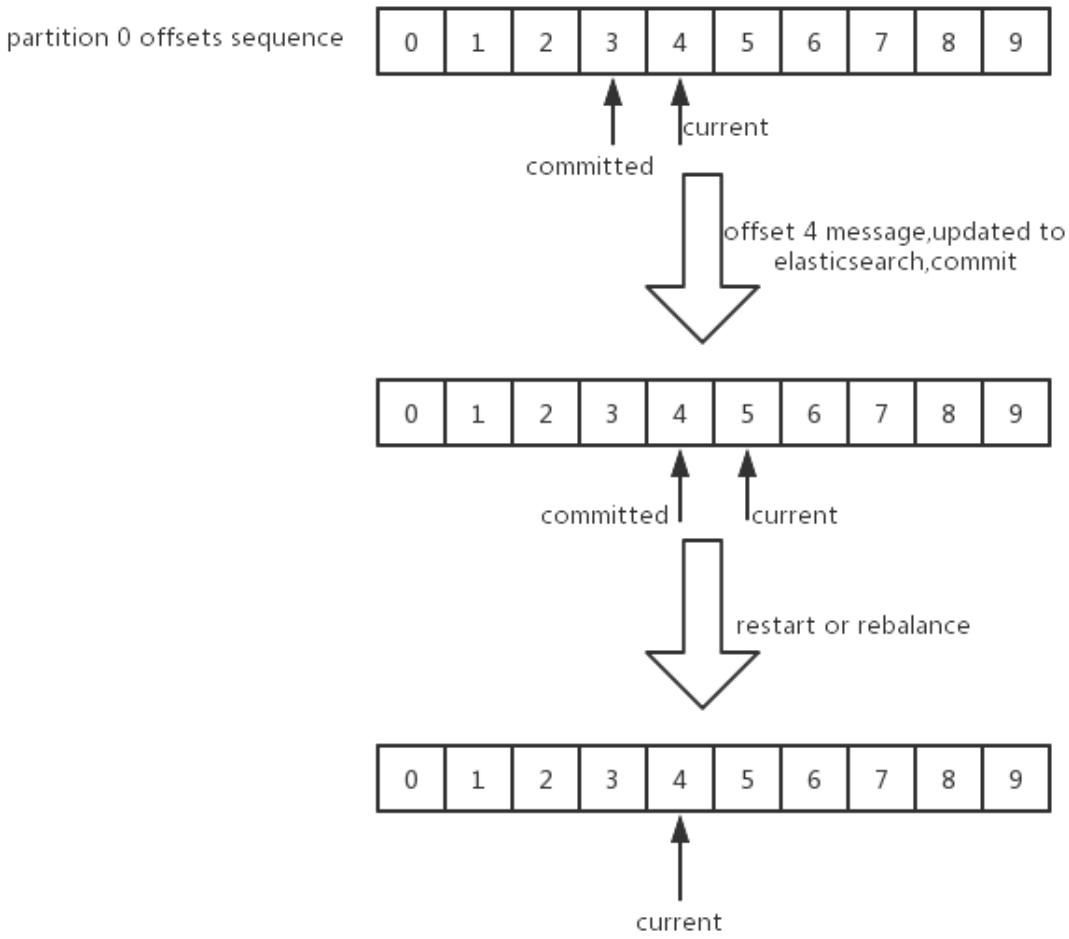


如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

完整性

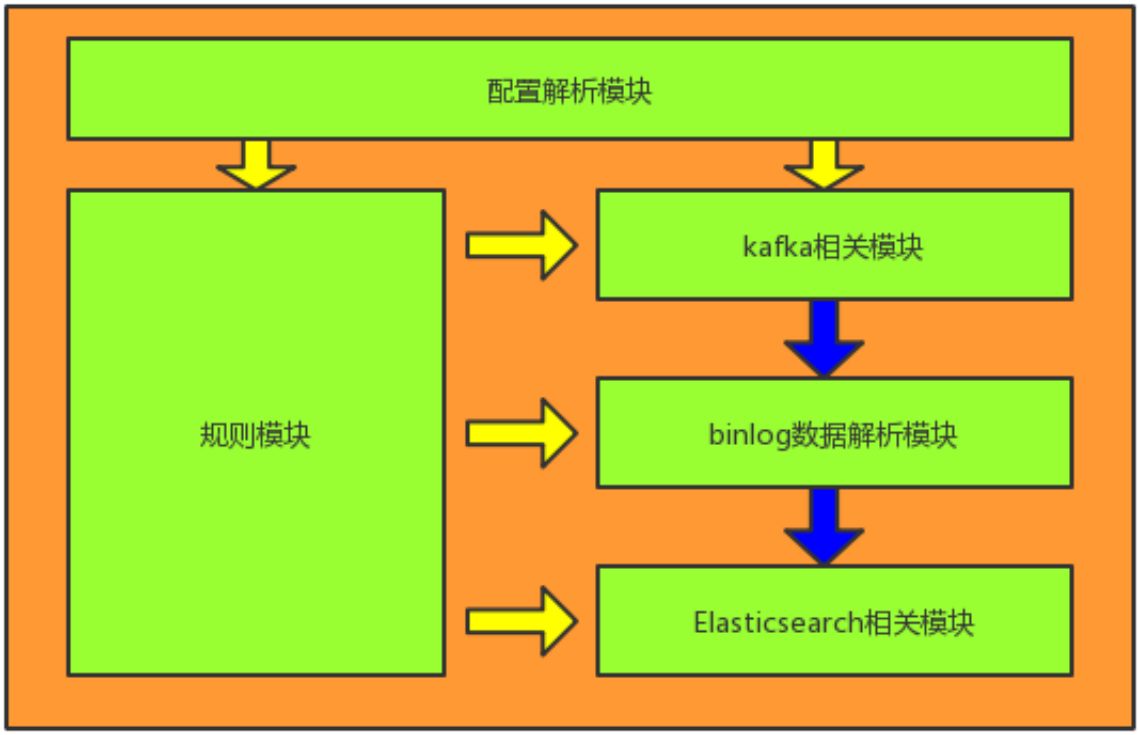
考虑到同步程序可能面临各种正常或异常的退出，以及 Consumer 数量变化时的 Rebalance，我们需要保证在任何情况下不能丢失 Binlog 数据。

利用 Kafka 的 Offset 机制，在确认一条 Message 数据成功写入 Elasticsearch 后，才 Commit 该条 Message 的 Offset，这样就保证了数据的完整性。而对于数据同步的使用场景，在保证了数据顺序性和完整性的情况下，重复消费是不会有影响的。



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

技术实现



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

功能模块

配置解析模块

负责解析配置文件（toml 或 json 格式），或在配置中心（Skipper）配置的 json 字符串。包括 Kafka 集群配置、Elasticsearch 地址配置、日志记录方式配置、MySQL 库表及字段与 Elasticsearch 的 Index 和 Mapping 对应关系配置等。

规则模块

规则模块决定了一条 Binlog 数据应该写入到哪个 Elasticsearch 索引、文档_id 对应的 MySQL 字段、Binlog 中的各个 MySQL 字段与索引 Mapping 的对应关系和写入类型等。
在本地化过程中，根据我们的业务场景，增加了对 MySQL 表各字段的 where 条件判断，来过滤掉不需要的 Binlog 数据。

Kafka 相关模块

该模块负责连接 Kafka 集群，获取 Binlog 数据。

在本地化过程中，该模块的大部分功能已经封装成了一个通用的 Golang Kafka Consumer Client。包括 Dba Binlog 订阅平台要求的 SASL 认证，以及从指定时间点的 Offset 开始消费数据。

Binlog 数据解析模块

原项目中的 Binlog 数据解析针对的是原始的 Binlog 数据，包含了解析 Replication 协议的实现。在我们的使用场景中，Binlog 数据已经是由 canal 解析成的 json 字符串，所以对该模块的功能进行了简化。

binlog json字符串示例

binlog json字符串示例

```
{
    "data": [
        {
            "primarykey": "65449415",
            "id": "22840702019050939653l4",
            "name": "测试名称",
            "product_id": "1112233",
            "product_name": "测试产品名称",
            ...
        }
    ],
    "database": "somedb",
    "old": [
        {
            "name": ""
        }
    ],
    "table": "sometable",
    "type": "UPDATE"
}
```

如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

上面是一个简化的 binlog json 字符串，通过该条 binlog 的 database 和 table 可以命中一条配置规则，根据该配置规则，把 Data 中的 key-value 构造成一个与对应 Elasticsearch 索引相匹配的 key-value map，同时包括一些数据类型的转换：

MySQL字段类型	写入ES时类型
tinyint, smallint, mediumint, int, bigint, year	int64
float,double	float32
char,varchar,text	string
datetime,timestamp	string,Y-m-d H:i:s
date	string,Y-m-d

如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

Elasticsearch相关模块

Binlog 数据解析模块生成的 key-value map，由该模块拼装成请求_bulk 接口的 update payload，写入 Elasticsearch。考虑到 MySQL 频繁更新时对 Elasticsearch 的写入压力，key-value map 会暂存到一个 slice 中，每 200ms 或 slice
长度达到一定长度时（可以通过配置调整），才会调用 Elasticsearch 的_bulk 接口，写入数据。

定制化开发

适应业务需求

upsert
业务中使用的索引数据可能是来自多个不同的表，同一个文档的数据来自不同表的时候，先到的数据是一条 index，后到的数据是一条 update，在我们无法控制先后顺序时，需要实现 upsert 功能。在_bulk 参数中加入

```
{
   "doc_as_upsert" : true
}
```

Filter
实际业务场景中，可能业务需要的数据只是某张表中的部分数据，比如用 type
字段标识该条数据来源，只需要把 type=1或2的数据同步到 Elasticsearch
中。我们扩展了规则配置，可以支持对 Binlog 指定字段的过滤需求，类似：

select * from sometable where type in (1,2)

快速增量

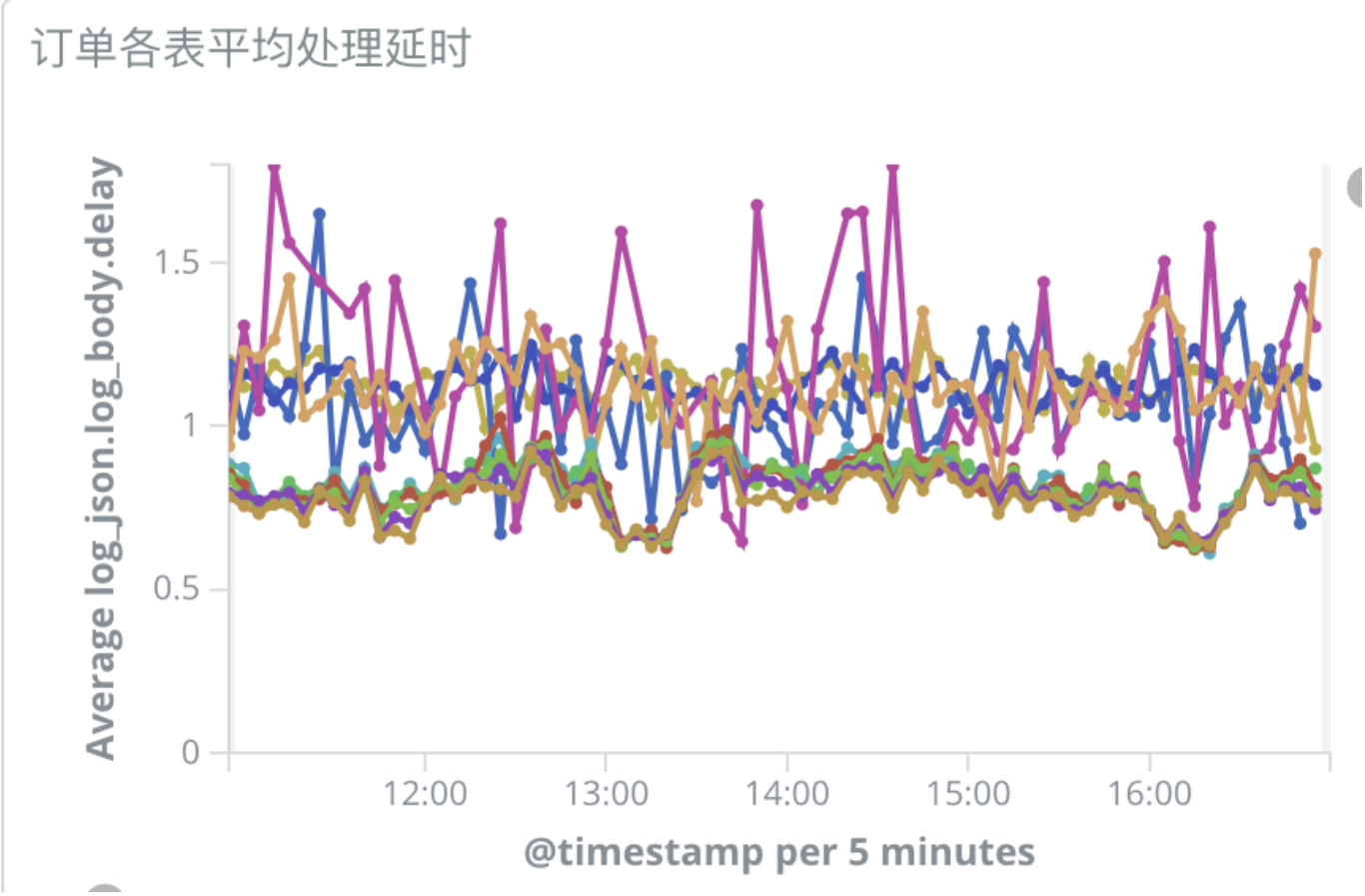
数据同步一般分为全量和增量。接入一个业务时，首先需要把业务现有的历史 MySQL 数据导入到 Elasticsearch
中，这部分为全量同步。在全量同步过程中以及后续增加的数据为增量数据。

在全量数据同步完成后，如果从最旧开始消费 Kafka，队列数据量很大的情况下，需要很长时间增量数据才能追上当前进度。为了更快的拿到所需的增量 Binlog，在 Consumer Group 消费 Kafka 之前，先获取各个 Topic 的 Partition 在指定时间的 offset 值，并 commit 这些 offset，这样在 Consumer Group 连接 Kafka 集群时，会从刚才提交的 offset 开始消费，可以立即拿到所需的增量 Binlog。

微服务和配置中心

项目使用马蜂窝微服务部署，为新接入业务提供了快速上线支持，并且在业务 Binlog 数据突增时可以方便快速的扩容 Consumer。
马蜂窝配置中心支持了各个接入业务的配置管理，相比于开源项目中的 toml 格式配置文件，使用配置中心可以更方便的管理不同业务不同环境的配置。

日志与监控



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

从上图中可以看出，订单各个表的数据同步延时平均在 1s 左右。把延时数据接入
ElastAlert，在延时数据过多时发送报警通知。

另一个监控指标是心跳检测，单独建立一张独立于业务的表，crontab 脚本每分钟修改一次该表，同时检查上一次修改是否同步到了指定的索引，如果没有，则发送报警通知。该心跳检测，监控了整个流程上的 Kafka、微服务和
ES，任何一个会导致数据不同步的环节出问题，都会第一个接到通知。

结语

目前接入的最重要业务方是电商的订单索引，数据同步延时稳定在 1s
左右。这次的开源项目本地化实践，希望能为一些有 Elasticsearch
数据同步需求的业务场景提供帮助。

本文作者：张坤，马蜂窝电商研发团队度假业务高级研发工程师。原文地址 基于 MySQL Binlog 的 Elasticsearch 数据同步实践

本文链接: 【】（）