Atitit 批量处理问题分析结果与总结

特别针对大批量循环情况下时间延迟问题。。

目录

[1. 目前分析的结果 1](#_Toc15916)

[2. 希望的目标 1](#_Toc20520)

[2.1. 面向未来 1](#_Toc27302)

[2.2. 通用化 1](#_Toc9093)

[2.3. 简化业务逻辑 2](#_Toc13010)

[2.4. 业务逻辑配置化，可直接在线调整 2](#_Toc7779)

[3. 解决方法总结 2](#_Toc31160)

[3.1. 更近的距离原则 2](#_Toc3664)

[3.2. 更快的响应速度，减少对库表占用 2](#_Toc6835)

[3.3. 压缩 2](#_Toc7478)

[3.4. 更快的存储 3](#_Toc19481)

[3.5. 业务逻辑调整（风险较大） 3](#_Toc21473)

[3.6. 更多的线程（协程等） 3](#_Toc6180)

[3.7. 改变库表结构（需要改动代码且数据迁移 风险高） 3](#_Toc11705)

[3.8. 升级数据库版本 3](#_Toc9810)

# 目前分析的结果

大批量下网络io占用时间积少成多

所有的操作，查询与修改数据时间都积少成多

数据库表占用时间也积少成多

# 希望的目标

## 面向未来

不仅解决现有问题，而且对未来问题也一样解决

## 通用化

， 不动业务逻辑代码来解决（通过配置优化 运维部署模式等提升）

对其他类似批量处理的问题也可一并提升，不必一一去调整

## 简化业务逻辑

，现在业务逻辑为了解决性能问题绕来绕去，业务复杂性大增

微调代码类其次。Sql微调合并类。

大动代码类的方法再其次。

更改逻辑方法最后，会造成业务逻辑复杂化，

## 业务逻辑配置化，可直接在线调整

更快的调整响应速度。。

免编译，免打包，如能免部署更佳

# 解决方法总结

## 更近的距离原则

包括同机部署（推荐。无需改动代码 ，提升应该在3-5倍左右，根据小规模测试结果），优点是无需改动代码，而且可以推广到其他批量处理任务上。。

Redis与db都可同机部署

存储过程（改动太多，较为繁琐 ，提升在一个数量级10倍左右，或许均衡一下，部分变为sp是个较好选择）

Map reduce模式（将代码发送至数据所在地进行运算），因为代码体积小而且只需发送一次。如果采用把数据拉入代码方向，则交互太多而且体积较大。。。需要改动代码

## 更快的响应速度，减少对库表占用

## 压缩

压缩网络交互

Sql压缩与合并 （sql微调整类。。相对），这个根据调整量，微调应该提升在30-50%左右，，大调整（部分变为存储过程）可能提升一俩倍到三五倍左右

本地开启直连localhost，unicsocket 命名管道模式，去除tcp开销可提升30%

压缩业务数据方向，需要改动大量逻辑代码，而且因为注单需要每单明细循环，经过讨论，合并数据可能带来较大问题。。

## 更快的存储

开启查询缓存 提升20-30% (推荐，已经开启，无需改变代码）

使用更快对存储redis等缓存注单以及处理结果，然后将处理结果刷新至数据库（需要大动代码）

使用mybatis二级缓存（集成redis）可以缓存注单类查询

是不是能使用mq缓存下处理结果？？再刷新到库（待测试）

## 业务逻辑调整（风险较大）

非主流业务流程看是否可以延后处理（天天分钱等），小规模测试提升可在50%左右

需要调整代码

## 更多的线程（协程等）

经过小规模测试，虽然这个提升并发，但现有的mysql5.6库并发多了貌似容易上锁，导致实际并发是有限的。。

## 改变库表结构（需要改动代码且数据迁移 风险高）

## 升级数据库版本

无需业务调整，但需要数据迁移

8.0比5.6应该要提升六七倍。。但目前分析主要不是数据库压力导致的，而是网络交互时间。。