**SUBWAY 1.1.0业务分析说明书**

（设计文件）

**二零一七年十二月**

**修订历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 日期 | 状态 | 修订人 | 摘要 |
| 1.0.0 | 2017-12 | C | 吴志明 | 起草 |
| 1.1.0 | 2018-01 | A | 吴志明 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

状态标识：C – Created A - Added M - Modified D - Deleted

# 业务概述

## 业务背景

当用户量大的时候，可能没办法一个server对接所有的client。为了可以让多个server提供相同的服务已达到负载均衡或者说充分发挥多个主机的运算能力。Subway作为消息代理服务器。把client的请求准确的送达参与服务的server上，达到动态增删server和多渠道处理客户端请求的目的。

## 目的和范围

## 术语和定义

subway:“地铁”服务，以此名字只是为了好理解

## 参考资料

暂无

# 需求概述

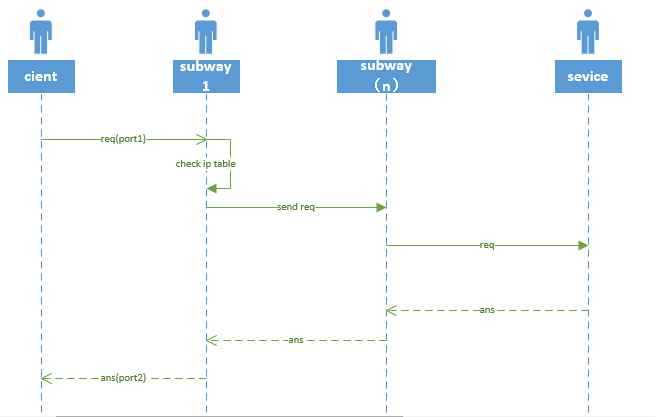
## 设计要点及设计约束

V1.0.0 :简单的实时消息队列，发布订阅消息。

1. 增加监控模块，当超过处理能力时拒绝接收消息。
2. 接收模块优化，限制链接开启的协程数。
3. 增加持久化，使消息为消息增加一个编号。接收端根据编号读取文件内容。（开关选项，加一层缓存）

# 设计概述

## V1.0系统架构



## 3.2、模块架构

**3.2.1、V1.0.0**

**一、接收模块**

1、一个连接以4K为一个处理单位。每接收4K开一个协程处理。

**二、处理模块**

1、根据消息头定义将消息发送到哪个队列和从哪个队列接收消息。

2、每个队列都存储在内存中。

**三、订阅模块**

1、为每个消息寻找订阅者，找到订阅者后。将消息推送给订阅者。

2、消息存储在固定的消息队列，消息插入和发布都需要上锁。

**四、当前架构思考**

1、经过测试、当数据量大的时候，且如果只有接受端的时候，subway的内存会爆涨。

2、当前代码中，每读取4K开一个协程处理。当数据交互比较大的时候，内存利用率不高。

3、由于消息入队和发布在同一个队列上，在操作队列的时候都需要加锁。使接收模块和订阅模块其实是加密耦合的。

4、消息丢包无法排查。

五、当前架构适用性

当前架构只适合在并发量较低且数据量较低，实时的场景

**3.2.1、V1.1.0**

**一、消息大小分析：**

1、小消息：1K以内的消息。一般作为远程调用入参等消息。

2、大小为4K左右的消息。-----定长消息。效率较高

3、大消息。 ----定长或者有长度范围的大消息。

4、长度无法确定的消息。 ---复杂的业务场景。

综合考虑：使用第4种。大消息需要是4K的整数倍。且需要独占一个队列。

**一、接收模块设计**

1.1、协程数量计算：

1、结果测试、一个协程的内存开销约为4K。

2、底层tcp协议每个socket连接也要消耗一定的内存空间。约4K（一个tcp网络链接内存消耗）

3、单节点以100W用户量计算，在无传输数据的情况下的内存使用量为1024\*1024 \* 8 \* 1024 = 8G。但是实际上很少场景会达到。以10W用户同时在线的情况下，一个连接对应2个协程使用1.6G。

1.2、多协程的副作用。

在golang中，需要特别注意的一点是所有的调度是平均的，如果要保证核心处理协程高效，需要尽量减少协程数量。如果100W个用户同时在线，以一个用户链接使用两个协程计算，那么就会有200W个协程产生，读写压力不说，光golang的调度，已经是一件很消耗cpu的了。需要注意，可能又时候读取完成到处理完成会不连续。

1.3、为每个链接配置两个两个协程处理。一个处理接收消息，一个处理发送消息。

**二、模块设计**

2.1、主题消息进行文件存储。一个是文件索引消息位置，一个是存储的文件。文件可以通过配置按需删除。

2.2、处理模块依然以4K为单位接收数据，但是处理的时候需要判断数据大小，4K的数据头肯定还是消息头，如果是短消息，支持在4K数据里边包含多个消息体。写文件以4K的整数倍大小写入。采用append的方式写入文件。

2.3、读取消息体的时候，通过消息序号来获取消息。每个主题对应一个文件存储，还有一个状态文件用来定位消息在文件中的偏移。由于状态文件会比较小，所以每次处理的时候会读取状态文件前后4K大小的内容到内存中。

**三、订阅模块**

3.1、消息订阅者发送消息到subway的时候需要发送订阅的主题以及编号，消息体为空即可。

3.2、subway通过调用者的订阅主题和编号，从状态文件中获取消息体位置，查看预留内存中是否有消息内容，然后获取消息返回给订阅者。并将消息后4M（可配置）文件内容存入内存。

**四、监控模块**

**3.1、内存监控**

使用golang内存监控方式监控内存，如果使用的内存超过健康值，采用的应对措施：

1. 请求拒绝、客户端做优化，如果服务端拒绝，则隔一段时间后重传。
2. 减少缓存长度。
3. 调用GC

**3.2、磁盘监控**

如果磁盘读写太大，发出告警。