SQS详细设计

目录

[主要解决的问题： 3](#_Toc325008556)

[主要逐句结构及其功能函数： 3](#_Toc325008557)

[Client端 3](#_Toc325008558)

[Server端 3](#_Toc325008559)

[部分流程图： 5](#_Toc325008560)

[Send 5](#_Toc325008561)

[Receive 5](#_Toc325008562)

[Delete 5](#_Toc325008563)

## 主要解决的问题：

1. 一致性：

在删除队列中消息的操作执行时对当前操作的消息对象进行加锁

1. 负载均衡：

将所有队列中的消息及其备份均匀散布于各个server中，使得每个server的负载相对均衡

1. 容错性：

所有消息均在不同server上备份，以预防崩溃时队列内容的丢失

1. 性能：

使用hash函数对消息的插入，删除，查询进行操作，时间复杂度低，查找效率高

## 主要逐句结构及其功能函数：

### Client端

1. 一个数组对象用于存储当前server端的信息诸如ip地址、端口号等用于建立连接
2. 粗粒度的Hash函数(用于决定消息的去向)

Server 3

Server 2

Server 4

Server 1

Hash(MsgID)

### Server端

1. 粗粒度的Hash函数：

用于receive时保证存取一致性

其具体实现与client端的相同

1. 细粒度的Hash函数：

基于MsgID决定当前消息在本地数据结构中的存储位置

同时用于取，删时的查找

Hash(MsgID) ------------->Index

Index

MsgID

Content

1. 本地服务器中的队列组：

以队列名为Index, 每个队列名对应一个相应的循环队列

此处的循环队列即2. 中Message的存储结构

Q2

Q3

Q4

Q5

Q1

1. 一个数组对象用于存储当前server端的信息诸如ip地址、端口号等用于建立连接

## 部分流程图：

### Send

1. Client自动生成一个唯一ID（根据时间等方法）
2. 调用Client端的Hash(MsgID)决定Message去向
3. 查询Client端的Server通信表，像选定server发送消息 Insert[MsgID , Queue Name , Content]
4. 在server中：
5. 接受数据[MsgID , Queue Name , Content]
6. 根据Queue Name决定Message存储队列
7. Hash(MsgID)决定Message存储位置

### Receive

1. Client端逐个向Server通信表中的Server发送请求
2. 若Server中无当前Queue的存储，则返回相应答复

若有，则返回消息，对当前消息进行加锁

同时使用Hash(MsgID)计算当前消息所备份的服务器，并发送对此消息加锁请求

### Delete

1. 若用户端主动发出删除消息请求：DeleteMessage(int64 MessageID)
2. 调用Client端的Hash(MsgID)决定Message去向
3. 查询Client端的Server通信表，像选定server发送消息 Delete[MsgID]
4. 在server中：
5. 接受数据Delete[MsgID]
6. Hash(MsgID)决定Message存储位置
7. 删除消息
8. 若超过了Visibility Time则服务器自动删除消息
9. Server删除当地本条消息
10. 使用Hash(MsgID)计算当前消息所备份的服务器，并发送对此消息删除请求