

# zxx的专栏

Come on! Let's Hello World! !

目录视图

个人资料



xxxxxx91116

关注

发私信

访问：821950次

积分：3968

等级：

BLOG > 5

排名：第5817名

原创：65篇

转载：21篇

译文：3篇

评论：136条

文章搜索

- 文章分类
- 非主流学科篇 (4)

shell script篇 (3)

工具篇 (12)

语言C/C++/JAVA/汇编篇 (11)

操作系统开发篇 (9)

攻击与安全篇 (2)

杂谈篇 (5)

网络篇 (12)

linux系统应用篇 (13)

MFC操作office (6)

Arduino/Hello (3)

JAVA/WEB (4)

开源项目学习 (6)

- 文章存档
- 2016年04月 (1)

2015年12月 (6)

2015年06月 (1)

2015年05月 (1)

2015年01月 (2)

展开

阅读排行

【P2P网络】磁力链接转换为...

- 【1024程序员节】获奖结果公布

【观点】有了深度学习，你还学传统机器学习算法么？

【资源库】火爆了的React Native都在研究什么

《RocketMQ》五、Consumer消费者

标签：RocketMQ

2015-12-23 23:18

4518人阅读

评论

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?)

[+]

Consumer-集群Push模式-简介：

## 0、背景介绍

Consumer主要用于向Broker请求Producer产生的消息，对其进行消费；对于RocketMQ，我们一定很好奇，如Consumer消费，如何保证Consumer的顺序性，不重复性呢？

存在的问题：

1. 如果在集群模式中新增/减少 组(group) 消费者，可能会导致重复消费；原因是：  
假设新增消费者前，ConsumerA正在消费MessageQueue-M，消费到第3个offset，  
这个时候新增了ConsumerB，那么根据集群模式的AllocateMessageQueue的策略，可能MessageQueue-M被分配给ConsumerA由于消费的offset没有实时更新回去，会导致ConsumerB和ConsumerA之前的消费有重叠；
2. 消费失败怎么办？
3. 异常处理
4. 线程，Auto变量的使用

## 一、术语介绍

topic: 最细粒度的订阅单位，一个group可以订阅多个topic的消息  
group: 组，一个组可以订阅多个topic  
clientId: 一个服务(IP/机器)的标识，一个机器可以有多个group；同时，多个相同group的clientId组成一个集群，一起  
messageQueue:消息队列，一个broker的一个topic有多个messageQueue  
offset: 每一个消息队列里面还有偏移(commitOffset, offset)的区别，为什么有2个offset呢？

集群消费：

广播消费：

立即消费：

顺序消费：

消费位置：

offsetStore-----commitOffset：消费到的offset

PullRequest ----- offset的区别：拉取的位置

fatal error: openssl/sha.h: N...	(512520)
fatal error: openssl/sha.h: N...	(24002)
【utorrent】ubuntu 安装uto...	(18222)
【Linux】limits.conf 不重启...	(15530)
【P2P网络】BitTorrent的DH...	(13322)
【MFC/C++操作word】Wor...	(13001)
【终端快捷键】Linux termin...	(11176)
【vim】vim行首加入某字符	(9163)
【MFC/C++操作Excel】Exce...	(8841)
【P2P网络】Extension for P...	(8831)

评论排行

【P2P网络】磁力链接转换为...	(43)
【orange】关于下载的freed...	(13)
【MFC/C++操作word】Wor...	(12)
shell脚本中调用vim 替换指定...	(8)
【图形学】直线扫描算法之---	(7)
【java】JCombox事件消息，...	(4)
【P2P网络】BitTorrent的DH...	(4)
【orange】OrangeS一个操...	(3)
fatal error: openssl/sha.h: N...	(3)
【utorrent】ubuntu 安装uto...	(3)

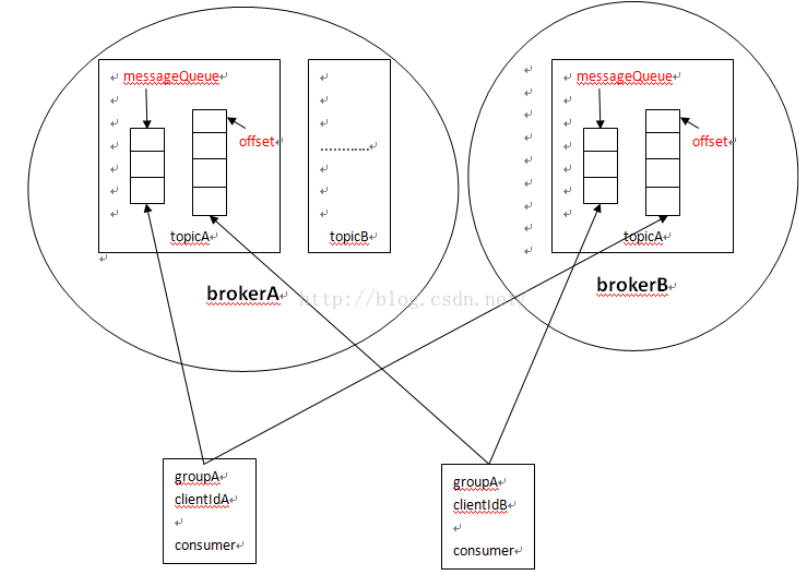
推荐文章

- \* 2016 年最受欢迎的编程语言是什么？
- \* Chromium扩展（Extension）的页面（Page）加载过程分析
- \* Android Studio 2.2 来啦
- \* 手把手教你做音乐播放器（二）技术原理与框架设计
- \* JVM 性能调优实战之：使用阿里开源工具TProfiler 在海量业务代码中精确定位性能代码

最新评论

- 《RocketMQ》三、NameServer  
adrianjian : > 2. Producer获取topic下的所有BrokerQueue，put消息> 3. Cons...
- 【P2P网络】磁力链接转换为种子文件 m...  
szoyj : 上面所有的方法，现在全失效了吧？
- 【MFC/C++操作Excel】Excel篇  
Persisterfan : 我也是这么认为的 有demo吗
- 【vim】vim行首加入某字符  
我在旷野漂流 : 我想问一下，vim能处理这个吗？文本如下：13.20子貢問曰：“何如斯可謂之士矣？”子曰：“行己有恥...
- 【mentohust】ubuntu 使用锐捷上网  
风中的狙击手 : +1
- 【P2P网络】磁力链接转换为种子文件 m...  
qq\_33732313 : @xxxxxx91116: 大神 那个迅雷的请求格式好像不行了 请问现在是改成什么了？
- MFC操作office通用分析方法  
12期-王金龙 : 学习了
- 【P2P网络】BitTorrent的DHT协议(译自...  
sagasarate : 最近在git上找到一个DHT的示例项目https://github.com/h31h31/H31-DH...
- 【P2P网络】磁力链接转换为种子文件 m...  
leisun321 : @Amayadream:服务器天天崩，没啥用处了。
- 【MFC/C++操作word】Word篇  
xxxxxx91116 : 上面的#define#define wd Extend 1 #define wdMove 0

二、总体框架



三、数据结构

数据结构主要分为2个部分来讲解：

一部分是在MQClientInstance中进行统一管理的，不管是Consumer还是Producer，能够统一管理的部分都放在了这，还有一部分是在Consumer或Producer中区分管理的，比如各自订阅的MessageQueue，下面对这两个部分分别介绍；

-----Part I：MQClientInstance-----

1. TopicRouteData：

用于保存了所有的Queue信息，不管是consumer还是producer的

```
[java]
01. private String orderTopicConf;//brokerName:num count
02. private List<QueueData> queueDatas;
03. private List<BrokerData> brokerDatas;
04. private HashMap<String/* brokerAddr */, List<String>/* Filter Server */> filterServerTable;
```

2.QueueData：内部通过write或者read来区分queue属于Consumer(read)/Producer(write)

```
[java]
01. private String brokerName;
02. private int readQueueNums;
03. private int writeQueueNums;
04. private int perm;
05. private int topicSynFlag;
```

3.BrokerData：Broker的地址信息

```
[java]
01. private String brokerName;
02. private HashMap<Long/* brokerId */, String/* broker address */> brokerAddrs;
```

4. PullRequest：拉取请求信息，包括所属组信息，要拉取的offset信息，Queue信息，消费进度信息

```
[java]
01. private String consumerGroup;
02. private MessageQueue messageQueue;
03. private ProcessQueue processQueue;
04. private long nextOffset;
```



5. PullMessageService：拉取信息的服务，会不断遍历每一个PullRequest进行信息的拉取

```
[java]
01. private final LinkedBlockingQueue<PullRequest> pullRequestQueue = new LinkedBlockingQueue<PullRequest>();
02. private final MQClientInstance mQClientFactory;
```

-----Part II：区分consumer -----

1. TopicPublishInfo：这个是Producer使用的保存MessageQueue的数据结构

```
[java]
01. private boolean orderTopic = false;
02. private boolean haveTopicRouterInfo = false;
03. private List<MessageQueue> messageQueueList = new ArrayList<MessageQueue>();
04. private AtomicInteger sendWhichQueue = new AtomicInteger(0);
```

2. SubscriptionData：包装consumer的消费信息，比如topic，订阅的tags

```
[java]
01. public final static String SUB_ALL = "*";
02. private boolean classFilterMode = false;
03. private String topic;
04. private String subString;
05. private Set<String> tagsSet = new HashSet<String>();
06. private Set<Integer> codeSet = new HashSet<Integer>();
07. private long subVersion = System.currentTimeMillis();
```

3.RebalanceImpl

```
[java]
01. ConcurrentHashMap<String/* topic */, Set<MessageQueue>> topicSubscribeInfoTable
02. ConcurrentHashMap<String /* topic */, SubscriptionData> subscriptionInner
03. <MessageQueue, ProcessQueue> processQueueTable
```

4.MessageQueue

```
[java]
01. private String topic;
02. private String brokenName;
03. private int queueId;
```

5. ProcessQueue

```
[java]
01. private final TreeMap<Long, MessageExt> msgTreeMap = new TreeMap<Long, MessageExt>();
02. private volatile long queueOffsetMax = 0L;
03. private final AtomicLong msgCount = new AtomicLong();
```

6.RemoteBrokerOffsetStore

```
[java]
01. private final MQClientInstance mQClientFactory;
02. private final String groupName;
03. private final AtomicLong storeTimesTotal = new AtomicLong(0);
04. private ConcurrentHashMap<MessageQueue, AtomicLong> offsetTable =
05.     new ConcurrentHashMap<MessageQueue, AtomicLong>();
```

# 四、主要类管理(group, instance, topic)

- 4.1 DefaultMQPushConsumer(group)：用于设置主要的参数，包括：组名，消费模式，消费offset，线程数目，批量
- 4.2 DefaultMQPushConsumerImpl(group)：包括RebalanceImpl，OffsetStore，AllocateStrategy
- 4.3 OffsetStore(group)：有2种模式，集群模式和广播模式不同；第一种是：RemoteBrokerOffsetStore，第二种是L  
将会记录我们消费到的offset位置
- 4.4 RebalanceImpl(group)：有2种模式，RebalancePushImpl，RebalancePullImpl，分别对应推拉2种模式的处理。  
MessageQueue进行一个平均分配，然后进行消费；对于推的模式，会根据不同位置拉取；对于拉的模式，它的拉取位
- 4.5 PullMessageService：循环所有的PullRequest，不断调用pullMessage进行MessageQueue的拉取
- 4.6 RebalanceService：循环所有的Consumer，对所有的consumer调用doRebalance
- 4.7 AllocateMessageQueueStrategy：分配消息的策略，将所有的MessageQueue均分到各个instance上面
- 4.8 PullAPIWrapper
- 4.9 ConsumeMessageService：有2种模式，ConsumeMessageConcurrentlyService和ConsumeMessageOrderlyService，并  
进行具体消费
- 4.10 MessageListener：客户端实现的接口，用于业务逻辑处理
- 4.11 MQClientAPIImpl：用于网络连接处理

## 五、总体模块

Consumer主要分为以下几个模块：

### 1. Rebalance模块：

主要包含以下几个部分：

RebalanceImpl

AllocateMessageQueueStrategy

RebalanceService

新增PullRequest

主要工作如下：

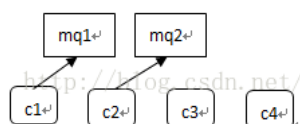
用于将某个topic的mqSet按策略分配到各个消费者cidSet，解释一下各个术语：

mqSet：是可以消费的所有Queue，可以理解成一块大蛋糕；

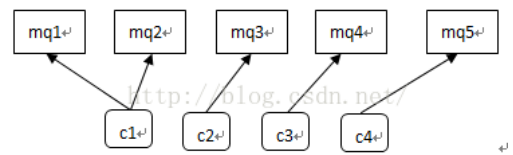
cidSet：可以理解成该topic的所有消费者，吃这块蛋糕的人。

这里采用的策略是遍历每一个consumer，再遍历每一个consumer的topic，对每个topic调用rebalanceByTopic；这些  
获得所有的midSet和cidSet，然后将他们进行均分，按图说话：

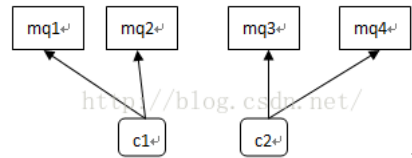
A. midSet < cidSet



B. midSet > cidSet, 且 midSet % cidSet != 0



C.midSet >= cidSet, 且midSet%cidSet=0



2. PullMessage模块:

主要包含以下几个部分：

- PullMessageService
- pullAPIWrapper
- PullCallback
- ConsumeMessageConcurrentlyService.processConsumeResult
- ConsumeMessageConcurrentlyService.ConsumeRequest

主要工作如下：

遍历PullMessageService的pullRequestQueue，take每一个PullRequest，然后调用pullMessage进行消息的拉取，进行回调处理

3. RemoteBrokerOffsetStore模块

在offsetTable中维护了一个offset变量，对这个offset的操作有2种，第一种是操作RemoteBrokerOffsetStore里面的c offset；还有一种是persist，将这些变量固化到远程的broker中

3.1 updateConsumeOffsetToBroker

设置UpdateConsumerOffsetRequestHeader为头部，然后调用updateConsumerOffsetOneway，以UPDATE\_CON 码，向broker服务器发送信息

3.2 设置removeOffset，将它从offsetTable里面移除

3.3 查询消费者序列long offset，queryConsumerOffset，QUERY\_CONSUMER\_OFFSET

4. Consumer模块:

这里和上面的PullMessage融合在一起处理，当pullMessage结束后，将会回调PullCallback。这里将调用consumeM submitConsumeRequest进行处理，而后更新offsetStore的消费位置等信息

5. update模块：

- 更新namesrv
- 更新topicRouteInfoFromServer：这里涉及到新增Subscribe
- 更新sendHeartbeat：注册consumer
- 更新persistAllConsumerSetInterval：更新offsetStore
- 更新线程池

6. 网络传输模块

MQClientAPIImpl

## 六、主要流程（Push+集群模式）

粗略篇：

1. DefaultMQPushConsumer创建组"CID\_001"
2. 调用subscribe，将会向rebalanceImpl中注册<topic,SubscriptionData>，用于后续的消息过滤
3. DefaultMQPushConsumerImpl.start()
  - 3.1 copySubscription(): 将DefaultMQPushConsumer的subscribe信息复制到DefaultMQPushConsumerImpl里面
  - 3.2 获取MQClientInstance
  - 3.3 设置RebalanceImpl的信息
  - 3.4 创建PullAIPWrapper
  - 3.5 创建offsetStore，(BROADCASTING)LocalFileOffsetStore,(CLUSTERING) RemoteBrokerOffsetStore

细致篇：

对应于，一个topic，对应了一个SubscriptionData，对应了很多的MessageQueue；

而每一个MessageQueue，又对应了ProcessQueue，ProcessQueue对应了每一个队列的消费进度

1.1 主要函数：lock, unlock，向函数给出的addr发出锁定，或者解锁mq的操作，以便于后续的消费

1.2 主要函数：doRebalance；遍历<String,SubscriptionData> subscriptionInner结构的每一个topic，调用rebalanceByTopic；

1.2.1 如果是广播模式

1.2.2 如果是集群模式

1.2.2.1 首先得到topic对应的所有MessageQueue，mqAll，这个是消息队列

1.2.2.2 得到对应group下面所有的cidAll，这个是消费者队列

1.2.2.3 调用strategy.allocate得到该consumer要消费的Set<MessageQueue> allocateResultSet

1.2.2.4 调用updateProcessQueueTableRebalance(topic,allocateResultSet)来更新processQueueTable，

A.首先，遍历processQueueTable，找到其有，而allocateResultSet没有的，调用removeUnnecessaryMessageQueue

B.其次，如果二者都有，但是在Push模式下，达到了pullExpired时间的，调用processQueueTable；

C. 遍历allocateResultSet，找到processQueueTable中没有的记录，将其加入到List<PullRequest>pullRequestList  
processQueueTable.put(mq, pullRequest.getProcessQueue())

D. 将上述新增的List<PullRequest>作为参数，调用dispatchPullRequest(pullRequestList);

未完待续，上述2个函数

removeUnnecessaryMessageQueue

dispatchPullRequest(pullRequestList);

## 七、一些实践阅读心得

1. HeartBeat：心跳需要进行加锁，因为心跳相当于注册，而unregister的时候相当于注销，加锁是防止在注销后，再注册。这里的临界变量是consumerTable

2. volatile：多线程操作某个变量时，使用该关键字可以防止由于编译器优化，导致从寄存器中读，而不是实时从内存读

3. ConcurrentHashMap：分段加锁，保证线程安全

4. AtomicInteger：原子自增自减

顶

2

踩

0

- [上一篇](#)    [《RocketMQ》四、Producer生产者](#)
- [下一篇](#)    [《RocketMQ》六、Broker中心节点](#)

参考知识库



算法与数据结构知识库

8304 关注 | 3135 收录

猜你在找

- 开源信息安全管理平台OSSIM入门

数据结构基础系列(3)：栈和队列

在Windows下SVN的版本管理与实战

Windows Server 2012 AD RMS 文…

sql server 性能优化和日常管理…
- 详解RocketMQ中的consumer

Rocketmq生产者和push消费者demo

rocketmq 消费者负载均衡-分布…

RocketMQ消费者设置setConsumeF…

rocketmq问题汇总-如何将特定消…

7



1.00/个

极化电源、高压极化电源、极化工艺专用高压

8



0.20/PCS

批发拆机UC3843 KA3843 TL3843

9



3.00/个

供应EE/EC型高频变压器、变压器

广告

查看评论

暂无评论

您还没有登录,请[登录](#)或[注册](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

Swift

智能硬件

Docker

OpenStack

VP
- IE10

Eclipse

CRM

JavaScript

数据库

Ubuntu

NFC

WAP

jQuery

BI

HTML5

Spring

Apa
- HTML

SDK

IIS

Fedora

XML

LBS

Unity

Splashtop

UML

components

Windows Mobile

Rai
- Cassandra

CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App

S
- Compuware

大数据

aptech

Perl

Tornado

Ruby

Hibernate

ThinkPHP

HBase

Pure

Solr

An
- Cloud Foundry

Redis

Scala

Django

Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服

杂志客服

微博客服

webmaster@csdn.net

400-600-2320

北京创新乐知信息技术有限公司 版权所有

江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved