

RocketMQ

学习笔记

寿命齿轮

2024 年 1 月 20 日

作者的话

本文用于记录作者在学习 RocketMQ 过程中所记录的笔记。

学习资料来源：

1. RocketMQ 官网
2. 编程导航星球知识库：Yes 大佬的消息队列专栏
3. 网上各类与消息队列相关的博客

使用系统:Ubuntu22.04(云服务器)

目录

第一章 初识：消息队列	1
1.1 介绍	1
1.2 作用与优点	1
1.3 适用场景	2
1.3.1 异步场景举例：用户注册	2
1.3.2 解耦场景举例：订单-库存管理	2
1.3.3 削峰场景举例：秒杀活动	3
1.3.4 日志处理场景	3
1.3.5 消息通讯场景	4
1.4 常用消息队列框架	4
第二章 启动：RocketMQ	6
2.1 下载二进制文件包	6
2.2 启动 NameServer	7
2.3 启动 Broker+Proxy	9

第一章 初识：消息队列

1.1 介绍

消息队列，顾名思义，存放**消息**（可类比为请求）的**队列**（一种先进先出的数据结构）。

其是一种常用于分布式系统的中间件，可以在不同的应用程序、服务或系统之间传递消息，并且常用于解耦合不同部分的系统，使得系统更加可扩展和灵活。

基本原理：发送者将消息放入队列，接收者从队列中获取消息并处理。

消息队列实质是一种方式，一种**在不同组件之间传递消息的通信方式**。发送者和接收者之间不需要直接通信，它们只需了解如何发送和接收消息即可。

1.2 作用与优点

由上述内容，可推断出消息队列的一些作用：

- **解耦**：发送者和接收者只需要关心发送消息和接受消息，不用关心彼此。
- **异步**：发送者不关心消息的处理，即不用等待消息的响应，故支持异步。
- **削峰**：某些活动的流量过大、请求过多，可能导致系统宕机；消息队列可以作为缓冲区，将这些请求暂时存储起来，以避免瞬时高流量，然后按照系统处理能力逐步消费，实现流量的平滑处理，从而降低系统的压力，避免宕机。

以及身为分布式系统的固有优点：

- **可扩展性**：在解耦后，可方便地单独对发送者或接收者或消息队列进行动态伸缩。
- **可靠性**：由于消息队列允许多个消费者和生产者，并且通常支持消息持久化和复制，因此即使其中一个组件出现故障，系统仍然可以继续运行并且消息也不会丢失。

1.3 适用场景

（真实适用场景还是需要多实践才能掌握，这里仅介绍一些常用场景）

1.3.1 异步场景举例：用户注册

需求

用户注册后需向其发送注册邮件和注册短信。

设计

用户注册后，将注册信息写入数据库；发送注册邮件；发送短信。

如不使用消息队列，不进行异步解耦，即注册服务器需要同步远程调用写入数据库、发送注册邮件、发送短信的三个函数，将与其他应用发生多次交互，同时还得等待响应，假设一个操作需要 0.5s，则该操作会占用注册服务器一个线程的 1.5s。

使用消息队列后，注册服务器直接向消息队列中写入三个消息（数据库写入消息、邮件发送消息、短信发送消息），并且是异步发送不用等待返回，假设一次发送消息为 0.1s，也仅需 0.3s。

1.3.2 解耦场景举例：订单-库存管理

需求

用户下订单后，库存系统需要减少相对数量。

设计

用户下单后，订单系统需要通知库存系统。

详细设计

原设计：订单系统调用库存系统的接口。存在缺陷：假如库存系统无法访问，则订单减库存将失败，从而导致订单失败；订单系统依赖库存系统接口，存在耦合。

改进后：订单系统发送订单消息（用户下单后，订单系统完成持久化处理，将消息写入消息队列，返回用户订单下单成功），库存系统读取订单消息并自行处理（订阅订单消息，采用拉/推的方式，获取下单信息，库存系统根据下单信息，进行库存操作）。解决缺陷：假如库存

系统无法访问，订单系统仅需要发送消息，可保持运转；订单消息仅发送消息，消息解读由库存系统进行（发布-订阅或消息队列模式），降低耦合度。

1.3.3 削峰场景举例：秒杀活动

需求

在秒杀活动中，大量用户同时抢购商品，可能会导致系统压力激增。为了应对这一情况，需要一种机制来平稳处理激增的请求流量，避免系统崩溃或性能下降。

设计

传统的处理方式可能会导致系统崩溃或性能下降。为了解决这个问题，可以使用消息队列来削峰填谷。

详细设计

1. 秒杀活动开始：当秒杀活动开始时，用户可以提交秒杀请求。
2. 请求入队：订单系统接收到用户的秒杀请求后，将请求消息写入消息队列，而不是立即处理。
3. 消息处理：秒杀请求消息被消息队列按照一定的规则（如先进先出）分发给后端处理程序。
4. 后端处理：后端处理程序逐条处理消息，检查库存并进行相应的处理（如减少库存、生成订单等）。

以此消息队列可平滑处理激增的请求流量，避免系统因突发流量而崩溃。

1.3.4 日志处理场景

需求

需要一种解决大量日志传输和实时处理的方案，以便对日志数据进行分析 and 可视化展示。

设计

设计一个分布式日志处理系统，包括以下组件：

1. 日志采集客户端：负责从各个日志源采集日志数据，并将数据定期写入消息队列中。

2. 消息队列：接收来自日志采集客户端的日志数据，负责数据的存储和转发。
3. 日志处理应用：订阅并消费 **Kafka** 队列中的日志数据，进行实时处理和分析。
4. **Logstash**：作为日志处理应用的一部分，负责对原始日志进行解析和转换，统一输出为 **JSON** 格式的数据。
5. **Elasticsearch**：作为日志处理应用的核心数据存储服务，接收 **Logstash** 处理后的 **JSON** 格式日志数据，实现实时的数据索引和查询。
6. **Kibana**：基于 **Elasticsearch** 的数据可视化组件，用于将 **Elasticsearch** 中的数据进行可视化展示和分析。

1.3.5 消息通讯场景

需求

需要一种高效的消息通讯机制，可以用于点对点通讯或者创建聊天室等场景，以实现实时的消息传递和交流。

设计

设计一个基于消息队列的消息通讯系统，包括以下两种场景：

1. 点对点通讯：客户端 **A** 和客户端 **B** 使用同一队列进行消息通讯；消息队列负责接收和转发客户端 **A** 和客户端 **B** 的消息。
2. 客户端 **A**、客户端 **B** 等多个客户端订阅同一主题：当有客户端发布消息时，消息队列将消息广播给所有订阅了该主题的客户端，客户端收到消息后进行展示。

1.4 常用消息队列框架

1. **RabbitMQ**：**RabbitMQ** 是一个开源的消息队列系统，实现了高级消息队列协议（**AMQP**），它是一个可靠、高可用、可扩展的消息代理。**RabbitMQ** 提供了多种消息传递模式，如点对点、发布/订阅等，适用于各种场景的应用程序。
2. **RocketMQ**：**RocketMQ** 是阿里巴巴开源的分布式消息队列系统，具有高吞吐量、低延迟、高可用性等特点。它支持丰富的消息模型，包括顺序消息、事务消息等，适用于大规模分布式系统的消息通信。

3. **Kafka:** Kafka 是由 Apache 软件基金会开发的分布式流处理平台和消息队列系统。Kafka 设计用于支持大规模的消息处理，具有高吞吐量、持久性、分区等特点，广泛应用于大数据领域。
4. **ActiveMQ:** ActiveMQ 是一个开源的消息中间件，实现了 Java Message Service (JMS) 规范。它支持多种传输协议，如 TCP、UDP、SSL 等，提供了丰富的功能，包括消息持久化、事务支持等。
5. **Amazon SQS:** Amazon SQS (Simple Queue Service) 是亚马逊提供的消息队列服务，可帮助构建分布式应用程序。它具有高可用性、可扩展性、灵活性等特点，适用于构建在亚马逊云平台上的应用程序。

本文将使用 RabbitMQ。

第二章 启动：RocketMQ

2.1 下载二进制文件包

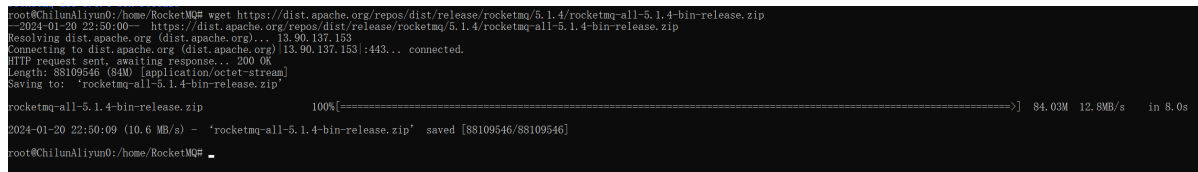
官网地址：<https://rocketmq.apache.org/zh/docs/quickStart/01quickstart>

获得二进制压缩包下载地址：

<https://dist.apache.org/repos/dist/release/rocketmq/5.1.4/rocketmq-all-5.1.4-bin-release.zip>

使用 `wget` 命令下载压缩包：

`wget https://dist.apache.org/repos/dist/release/rocketmq/5.1.4/rocketmq-all-5.1.4-bin-release.zip`



```
root@ChilunAliyun0:/home/RocketMQ# wget https://dist.apache.org/repos/dist/release/rocketmq/5.1.4/rocketmq-all-5.1.4-bin-release.zip
--2024-01-20 22:50:00-- https://dist.apache.org/repos/dist/release/rocketmq/5.1.4/rocketmq-all-5.1.4-bin-release.zip
Resolving dist.apache.org (dist.apache.org)... 13.90.137.153
Connecting to dist.apache.org (dist.apache.org) |13.90.137.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 88109546 (84M) [application/octet-stream]
Saving to: 'rocketmq-all-5.1.4-bin-release.zip'

rocketmq-all-5.1.4-bin-release.zip 100%[=====] 84.03M 12.5MB/s in 8.0s

2024-01-20 22:50:09 (10.6 MB/s) - 'rocketmq-all-5.1.4-bin-release.zip' saved [88109546/88109546]

root@ChilunAliyun0:/home/RocketMQ#
```

图 2.1: 下载二进制压缩包

使用 `unzip` 命令解压二进制文件压缩包：

`unzip rocketmq-all-5.1.4-bin-release.zip`

```
root@ChilunAliyun0:/home/RocketMQ# ls
rocketmq-all-5.1.4-bin-release.zip
root@ChilunAliyun0:/home/RocketMQ# unzip rocketmq-all-5.1.4-bin-release.zip
Archive:  rocketmq-all-5.1.4-bin-release.zip
  inflating: rocketmq-all-5.1.4-bin-release/LICENSE
  inflating: rocketmq-all-5.1.4-bin-release/NOTICE
  inflating: rocketmq-all-5.1.4-bin-release/README.md
   creating: rocketmq-all-5.1.4-bin-release/benchmark/
  inflating: rocketmq-all-5.1.4-bin-release/benchmark/consumer.sh
  inflating: rocketmq-all-5.1.4-bin-release/benchmark/runclass.sh
  inflating: rocketmq-all-5.1.4-bin-release/benchmark/tproducer.sh
  inflating: rocketmq-all-5.1.4-bin-release/benchmark/batchproducer.sh
  inflating: rocketmq-all-5.1.4-bin-release/benchmark/producer.sh
  inflating: rocketmq-all-5.1.4-bin-release/benchmark/shutdown.sh
   creating: rocketmq-all-5.1.4-bin-release/bin/
  inflating: rocketmq-all-5.1.4-bin-release/bin/tools.cmd
  inflating: rocketmq-all-5.1.4-bin-release/bin/runserver.sh
  inflating: rocketmq-all-5.1.4-bin-release/bin/mqbroker.numanode0
  inflating: rocketmq-all-5.1.4-bin-release/bin/mqproxy.cmd
  inflating: rocketmq-all-5.1.4-bin-release/bin/runbroker.cmd
  inflating: rocketmq-all-5.1.4-bin-release/bin/mqbroker.numanodel
  inflating: rocketmq-all-5.1.4-bin-release/bin/mqbrokercontainer
  inflating: rocketmq-all-5.1.4-bin-release/bin/mqproxy
  inflating: rocketmq-all-5.1.4-bin-release/bin/mqshutdown.cmd
  inflating: rocketmq-all-5.1.4-bin-release/bin/mqcontroller
  inflating: rocketmq-all-5.1.4-bin-release/bin/runserver.cmd
  inflating: rocketmq-all-5.1.4-bin-release/bin/mqadmin
  inflating: rocketmq-all-5.1.4-bin-release/bin/mqadmin.cmd
   creating: rocketmq-all-5.1.4-bin-release/bin/controller/
```

图 2.2: 解压二进制文件

2.2 启动 NameServer

进入目录 `rocketmq-all-5.1.4-bin-release`, 执行命令:

`nohup sh bin/mqnamesrv &`

命令讲解:

- **nohup**: 这代表“不挂起”。在终端中执行命令然后关闭终端时, 与该命令相关联的进程通常也会终止。nohup 可以防止这种情况发生。
- **sh**: 执行脚本文件的 shell 命令。
- **bin/mqnamesrv**: 要运行的脚本路径。
- **&**: 后台运行。

发现运行失败:

```
root@ChilunAliyun0:/home/RocketMQ/rocketmq-all-5.1.4-bin-release# nohup sh bin/mqnamesrv &
[1] 107550
root@ChilunAliyun0:/home/RocketMQ/rocketmq-all-5.1.4-bin-release# nohup: ignoring input and app
ending output to 'nohup.out'

[1]+  Exit 1                  nohup sh bin/mqnamesrv
root@ChilunAliyun0:/home/RocketMQ/rocketmq-all-5.1.4-bin-release#
```

图 2.3: 名字服务器启动失败

查看 nohup.out 文件, 发现报错: **OpenJDK 64-Bit Server VM warning: INFO: os::commit_memory(0x0000000700000000, 4294967296, 0) failed; error='Not enough space' (errno=12)**

操作系统内存不足 (由于 RocketMQ 对内存要求极高, 所以自己用云服务器运行基本都会报错), 进行修改:

进入 rocketmq-all-5.1.4-bin-release/bin 目录, 对 runserver.sh 和 runbroker.sh 以及 tools.sh 进行修改。(可使用 vim 的/+ 关键字进行查找)

1. runserver.sh:

```
JAVA_OPT="$JAVA_OPT -server -Xms4g -Xmx4g -Xmn2g -XX:MetaspaceSize=128m  
-XX:MaxMetaspaceSize=320m"
```

替换为

```
JAVA_OPT="$JAVA_OPT -server -Xms256m -Xmx256m -Xmn128m  
-XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m"
```

注意有两处。

2. runbroker.sh:

```
JAVA_OPT="$JAVA_OPT -server -Xms8g -Xmx8g"
```

替换为 **JAVA_OPT="\$JAVA_OPT -server -Xms256m -Xmx256m",**

```
JAVA_OPT="$JAVA_OPT -Xmn8G -XX:+UseConcMarkSweepGC
```

替换为

```
JAVA_OPT="$JAVA_OPT -Xmn256m -XX:+UseConcMarkSweepGC
```

3. tools.sh:

```
JAVA_OPT="$JAVA_OPT -server -Xms1g -Xmx1g -Xmn256m -XX:MetaspaceSize=128m  
-XX:MaxMetaspaceSize=128m"
```

替换为

```
JAVA_OPT="$JAVA_OPT -server -Xms256g -Xmx256g -Xmn128m  
-XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=128m"。
```

再次输入命令: **nohup sh bin/mqnamesrv &**

未报错, 查看 nohup.out 文件, 发现启动成功:

```
# There is insufficient memory for the Java Runtime Environment to continue.
# Native memory allocation (mmap) failed to map 4294967296 bytes for committing reserved memory
# An error report file with more information is saved as:
# /home/RocketMQ/rocketmq-all-5.1.4-bin-release/hs_err_pid107578.log
The Name Server boot success. serializeType=JSON, address 0.0.0.0:9876
```

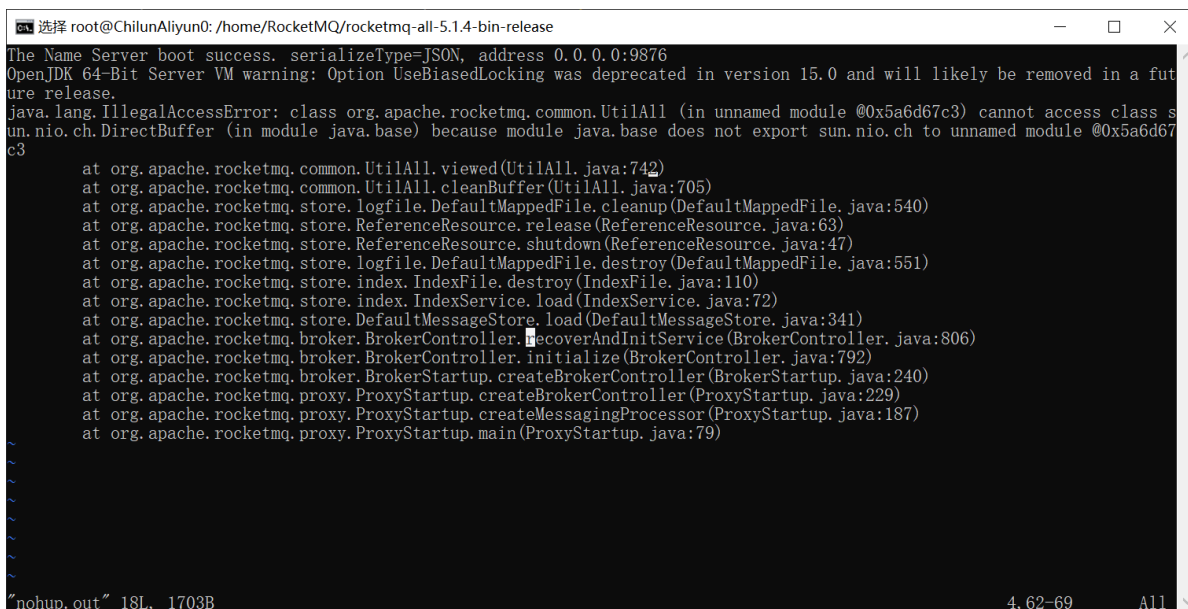
图 2.4: 名称服务器启动成功

2.3 启动 Broker+Proxy

执行命令:

nohup sh bin/mqbroker -n localhost:9876 --enable-proxy &

未报错, 查看 nohup.out 文件, 发现启动还是失败:



```
选择 root@ChilunAliyun0: /home/RocketMQ/rocketmq-all-5.1.4-bin-release
The Name Server boot success. serializeType=JSON, address 0.0.0.0:9876
OpenJDK 64-Bit Server VM warning: Option UseBiasedLocking was deprecated in version 15.0 and will likely be removed in a future release.
java.lang.IllegalAccessException: class org.apache.rocketmq.common.UtilAll (in unnamed module @0x5a6d67c3) cannot access class sun.nio.ch.DirectBuffer (in module java.base) because module java.base does not export sun.nio.ch to unnamed module @0x5a6d67c3
    at org.apache.rocketmq.common.UtilAll.viewed(UtilAll.java:742)
    at org.apache.rocketmq.common.UtilAll.cleanBuffer(UtilAll.java:705)
    at org.apache.rocketmq.store.logfile.DefaultMappedFile.cleanup(DefaultMappedFile.java:540)
    at org.apache.rocketmq.store.ReferenceResource.release(ReferenceResource.java:63)
    at org.apache.rocketmq.store.ReferenceResource.shutdown(ReferenceResource.java:47)
    at org.apache.rocketmq.store.logfile.DefaultMappedFile.destroy(DefaultMappedFile.java:551)
    at org.apache.rocketmq.store.index.IndexFile.destroy(IndexFile.java:110)
    at org.apache.rocketmq.store.index.IndexService.load(IndexService.java:72)
    at org.apache.rocketmq.store.DefaultMessageStore.load(DefaultMessageStore.java:341)
    at org.apache.rocketmq.broker.BrokerController.recoverAndInitService(BrokerController.java:806)
    at org.apache.rocketmq.broker.BrokerController.initialize(BrokerController.java:792)
    at org.apache.rocketmq.broker.BrokerStartup.createBrokerController(BrokerStartup.java:240)
    at org.apache.rocketmq.proxy.ProxyStartup.createBrokerController(ProxyStartup.java:229)
    at org.apache.rocketmq.proxy.ProxyStartup.createMessagingProcessor(ProxyStartup.java:187)
    at org.apache.rocketmq.proxy.ProxyStartup.main(ProxyStartup.java:79)

"nohup.out" 18L, 1703B                                4,62-69      All
```

图 2.5: broker 启动失败

原因是 JAVA 版本过高, 进行修复。

修改 runbroker.sh 文件:

在 **numactl --interleave=all pwd > /dev/null 2>&1** 上方添加

\$JAVA \$JAVA_OPT --add-exports=java.base/sun.nio.ch=ALL-UNNAMED \$@

然后再次运行 **nohup sh bin/mqbroker -n localhost:9876 --enable-proxy &**

并查看 nohup.out, 发现为不断更新的日志文件, 推测运行成功。

查看/root/logs/rocketmqlogs, 发现运行成功。

```

root@ChilunAliyun0: ~/logs/rocketmqlogs
2024-01-20 23:55:14 INFO main - Try to start service thread:QueueLockManager started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:QueueLockManager started:true lastThread:Thread[QueueLockManager, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:PopRevokeService_0 started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:PopRevokeService_0 started:true lastThread:Thread[PopRevokeService_0, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:PopRevokeService_1 started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:PopRevokeService_1 started:true lastThread:Thread[PopRevokeService_1, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:PopRevokeService_2 started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:PopRevokeService_2 started:true lastThread:Thread[PopRevokeService_2, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:PopRevokeService_3 started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:PopRevokeService_3 started:true lastThread:Thread[PopRevokeService_3, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:PopRevokeService_4 started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:PopRevokeService_4 started:true lastThread:Thread[PopRevokeService_4, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:PopRevokeService_5 started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:PopRevokeService_5 started:true lastThread:Thread[PopRevokeService_5, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:PopRevokeService_6 started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:PopRevokeService_6 started:true lastThread:Thread[PopRevokeService_6, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:PopRevokeService_7 started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:PopRevokeService_7 started:true lastThread:Thread[PopRevokeService_7, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:PopLongPollingService started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:PopLongPollingService started:true lastThread:Thread[PopLongPollingService, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:TopicQueueMappingCleanService started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:TopicQueueMappingCleanService started:true lastThread:Thread[TopicQueueMappingCleanService, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:FileWatchService started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:FileWatchService - Start topic queue mapping clean service thread!
2024-01-20 23:55:14 INFO main - Try to start service thread:PullRequestHoldService started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:PullRequestHoldService started:true lastThread:Thread[PullRequestHoldService, 5, main]
2024-01-20 23:55:14 INFO FileWatchService - FileWatchService service started
2024-01-20 23:55:14 INFO main - Start service thread:PullRequestHoldService started:true lastThread:Thread[PullRequestHoldService, 5, main]
2024-01-20 23:55:14 INFO PullRequestHoldService - PullRequestHoldService service started
2024-01-20 23:55:14 INFO main - Try to start service thread:BroadcastOffsetManager started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:BroadcastOffsetManager started:true lastThread:Thread[BroadcastOffsetManager, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:ColdDataPullRequestHoldService started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:ColdDataPullRequestHoldService started:true lastThread:Thread[ColdDataPullRequestHoldService, 5, main]
2024-01-20 23:55:14 INFO main - Try to start service thread:ColdDataCgCtrService started:false lastThread:null
2024-01-20 23:55:14 INFO main - Start service thread:ColdDataCgCtrService started:true lastThread:Thread[ColdDataCgCtrService, 5, main]
2024-01-20 23:55:14 INFO main - ScheduleServiceStatus changed to true
2024-01-20 23:55:14 INFO main - load /root/store/config/delayOffset.json OK
2024-01-20 23:55:14 INFO main - TransactionCheckService status changed to true
2024-01-20 23:55:14 INFO main - TransactionCheckService started:false lastThread:null
2024-01-20 23:55:14 INFO main - Try to start service thread:TransactionalMessageCheckService started:true lastThread:Thread[TransactionalMessageCheckService, 5, main]
2024-01-20 23:55:14 INFO main - Set PopRevokeService Status to true
2024-01-20 23:55:14 INFO main - The broker [ChilunAliyun0, 172.18.187.94:10911] boot success, serialType=KNOX and name server is localhost:9876
2024-01-20 23:55:23 INFO BrokerControllerScheduledThread1 - Dispatch task fall behind commit log Obytes
2024-01-20 23:55:24 INFO brokerOutApi_thread_2 - Registering current broker to name server completed. TargetHost=localhost:9876
2024-01-20 23:55:24 INFO brokerOutApi_thread_3 - Registering current broker to name server completed. TargetHost=localhost:9876
2024-01-20 23:55:24 INFO BrokerControllerScheduledThread1 - Dispatch task fall behind commit log Obytes
2024-01-20 23:55:24 INFO brokerOutApi_thread_4 - Registering current broker to name server completed. TargetHost=localhost:9876
2024-01-20 23:55:24 INFO brokerOutApi_thread_1 - Registering current broker to name server completed. TargetHost=localhost:9876
2024-01-20 23:55:24 INFO BrokerControllerScheduledThread1 - Dispatch task fall behind commit log Obytes
2024-01-20 23:55:24 INFO brokerOutApi_thread_2 - Registering current broker to name server completed. TargetHost=localhost:9876
2024-01-20 23:55:24 INFO brokerOutApi_thread_3 - Registering current broker to name server completed. TargetHost=localhost:9876
2024-01-20 23:55:24 INFO BrokerControllerScheduledThread1 - Dispatch task fall behind commit log Obytes
2024-01-20 23:55:24 INFO brokerOutApi_thread_4 - Registering current broker to name server completed. TargetHost=localhost:9876
2024-01-20 23:55:24 INFO brokerOutApi_thread_1 - Registering current broker to name server completed. TargetHost=localhost:9876
/ success

```

图 2.6: broker 启动成功

至此，RocketMQ 启动成功。