消息中间件的简易实现

学号 2019104267

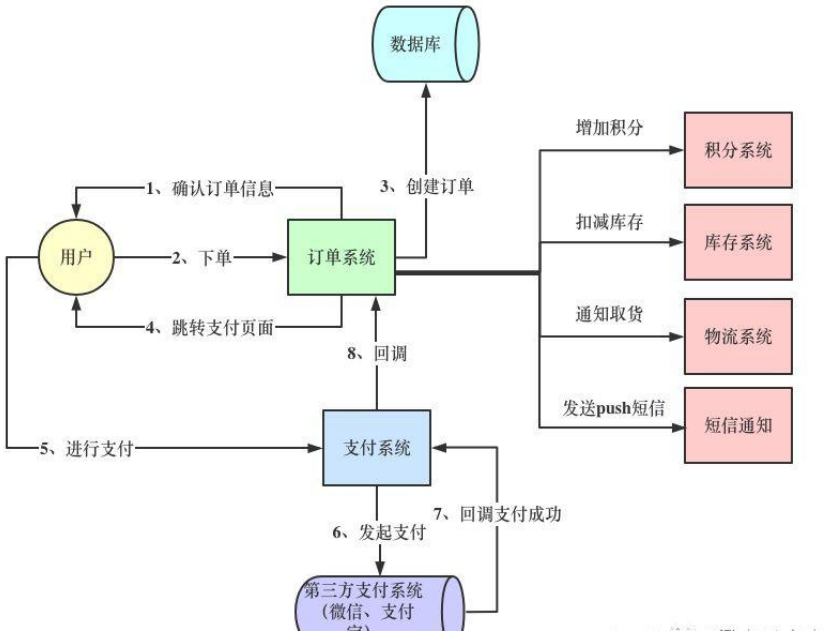
姓名 王青青

目录

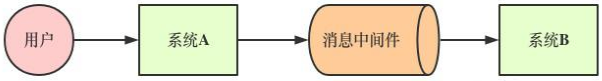
一、消息中间件原理及作用

对一个交易系统来说，每卖掉一件商品，需要在多个系统，例如积分系统/库存系统/物流系统/短信系统中进行操作，这些操作同步进行完成大概需要 1-2 秒。如果在负载量较大的高峰期，服务器的磁盘，I/O，CPU 负载会很高，数据库性能也在下降，最后导致完成一次交易可能需要用户等待几秒。

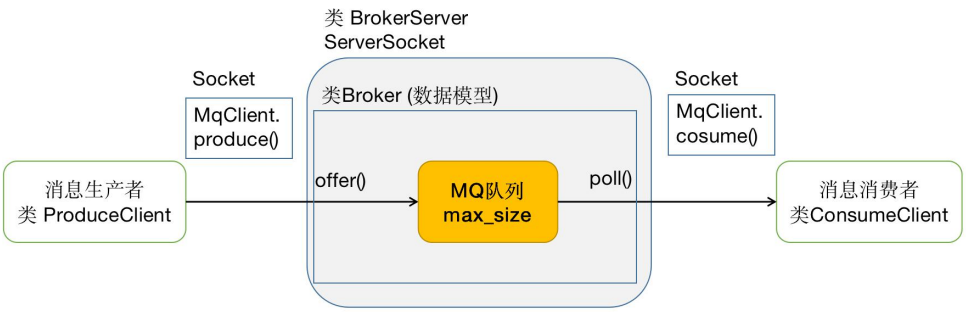
能否将订单系统与后台其他系统解耦，用户在订单系统下单后，不需要等待/关心其他系统是否执行完成，就能完成本次下单？



可以引入消息中间件来解决这个问题。如下图，系统 A 发送消息到队列中，就认为已经完成了，系统 B 可以在之后的任何时间依据自己的资源情况来读取新消息，然后完成相应操作，这个过程可以称做异步调用。



二、消息中间件简单实现框架



1. 创建使用 ArrayBlockingQueue 数据模型作为消息自定义中间件的消息载体，及消费生产模式。
2. 创建线程模拟 MQ 的消息接受及发送，启动后开启消息接受及发送。

3. 创建生产及消费方法，模拟消息机制
4. 创建消息接受者
5. 创建消息生产者

三、具体实现代码附录

```
package MyQueue;
import java.util.concurrent.ArrayBlockingQueue;
public class Broker {
    //消息队列存储最大值
    private final static int MAX_SIZE=3;

    //保存消息数据的容器
    private static ArrayBlockingQueue<String>
messageQueue=new ArrayBlockingQueue<>(MAX_SIZE);

    //生产消息
    public static void produce(String msg){
        if (messageQueue.offer(msg)) {
            System.out.println("消息发送成功，msg："+msg+"暂存队
列中的消息数量是:"+messageQueue.size());
        }else{
            System.out.println("消息处理数据中心数据达到最大负荷，
不能继续放入消息");
        }
        System.out.println("============================");
    }

    //消费消息
    public static String consume(){
        String msg=messageQueue.poll();
        if (msg!=null) {
            //消费条件满足时从消息容器中取出一条消息
            System.out.println("已经消费消息："+msg+"单签暂存的
消息数"+messageQueue.size());
        }else{
            System.out.println("消息处理中心内没有消息可供消费!
");
        }
        System.out.println("============================");
        return msg;
    }
}

package MyQueue;
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
public class BrokerServer implements Runnable {
    public static int SERVER_PORT=9999;
    private final Socket socket;
    public BrokerServer(Socket socket){
        this.socket=socket;
    }

    @Override
    public void run(){
        try {
            BufferedReader in=new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter out=new
PrintWriter(socket.getOutputStream());
            while(true){
                String str=in.readLine();
                if (str==null) {
                    continue;
                }
                System.out.println("接受到原始数据: "+str);

                if (str.equals("CONSUME")) {//consume 表示需要
消费一条消息
                    String meString=Broker.consume();
                    out.println(meString);
                    out.flush();
                }else{
                    //其他情况都表示生产消息放到消息队列中
Broker.produce(str);
                }
            }

        } catch (IOException e) {
            System.out.println(e.getMessage());
        }

    }
```

```
    public static void main(String[] args) throws IOException
{
        ServerSocket server = new ServerSocket(SERVER_PORT);
        while (true) {
            BrokerServer borkerServer=new
BrokerServer(server.accept());
            new Thread(borkerServer).start();
        }
    }

}
package MyQueue;
import java.io.BufferedReader;import
java.io.IOException;import
java.io.InputStreamReader;import
java.io.PrintWriter;import java.net.InetAddress;import
java.net.Socket;import java.net.UnknownHostException;

public class MqClient {
    //生产者
    @SuppressWarnings("unused")
    public static void produce(String message) throws
IOException {
        @SuppressWarnings("resource")
        Socket socket = new
Socket(InetAddress.getLocalHost(),BrokerServer.SERVER_POR
T);
        PrintWriter out=new
PrintWriter(socket.getOutputStream());
                out.println(message);
                out.flush();
    }

    //消费消息ss
    public static String consume() throws
UnknownHostException, IOException{
        @SuppressWarnings("resource")
        Socket socket = new
Socket(InetAddress.getLocalHost(),BrokerServer.SERVER_POR
T);
        BufferedReader in=new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out=new
PrintWriter(socket.getOutputStream());
```

```
        //先向消息列队发送字符串“CONSUME”表示消费
                out.println("CONSUME");
                out.flush();
                //再从消息列队获取一条消息
                String message=in.readLine();
                return message;
    }
}
```

```
package MyQueue;
import java.io.IOException;import
java.net.UnknownHostException;
public class ConsumeClient {
       public static void main(String[] args) throws
UnknownHostException, IOException {
           MqClient mq=new MqClient();
           String mes=mq.consume();
           System.out.println("获取的消息为: "+mes);
       }
}
```

```
package MyQueue;
import java.io.IOException;
public class ProduceClient {
       public static void main(String[] args) throws
IOException {
           MqClient client=new MqClient();
           client.produce("Hello new World!");
       }
}
```