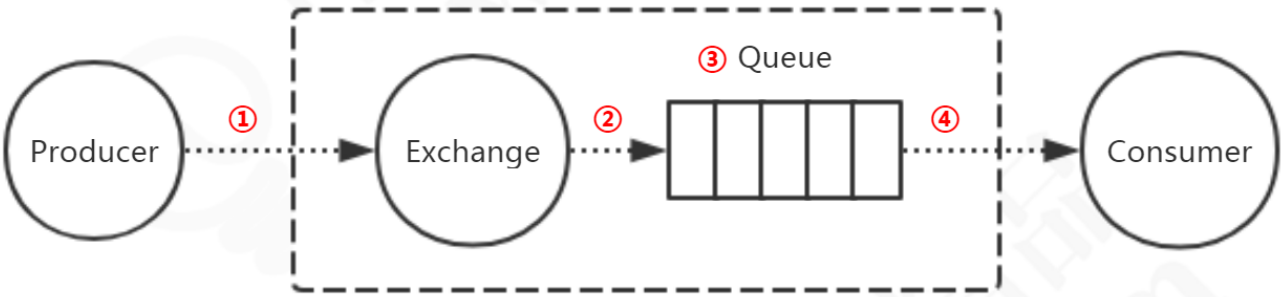
RabbitMQ 2-可靠性投递与生产实践

可靠性投递

首先需要明确，效率与可靠性是无法兼得的，如果要保证每一个环节都成功，势必会对消息的收发效率造成影响。

如果是一些业务实时一致性要求不是特别高的场合，可以牺牲一些可靠性来换取效率。



① 代表消息从生产者发送到Exchange;

② 代表消息从Exchange路由到Queue；

③ 代表消息在Queue中存储；

④ 代表消费者订阅Queue并消费消息。

1、确保消息发送到RabbitMQ服务器

可能因为网络或者Broker的问题导致①失败，而生产者是无法知道消息是否正确发送到Broker的。

有两种解决方案，第一种是Transaction（事务）模式，第二种Confirm（确认）模式。

在通过channel.txSelect方法开启事务之后，我们便可以发布消息给RabbitMQ了，**如果事务提交成功，则消息一定到达了RabbitMQ中**，如果在事务提交执行之前由于RabbitMQ异常崩溃或者其他原因抛出异常，这个时候我们便可以将其捕获，进而通过执行channel.txRollback方法来实现事务回滚。使用事务机制的话会“吸干”RabbitMQ的性能，一般不建议使用。

生产者通过调用channel.confirmSelect方法（即Confirm.Select命令）将信道设置为confirm模式。一旦消息被投递到所有匹配的队列之后，RabbitMQ就会发送一个确认（Basic.Ack）给生产者（包含消息的唯一ID），这就使得生产者知晓消息已经正确到达了目的地了。

参考：

com.gupaoedu.transaction

com.gupaoedu.confirm

2、确保消息路由到正确的队列

可能因为路由关键字错误，或者队列不存在，或者队列名称错误导致②失败。

使用mandatory参数和ReturnListener，可以实现消息无法路由的时候返回给生产者。

另一种方式就是使用备份交换机（alternate-exchange），无法路由的消息会发送到这个交换机上。

```
Map<String,Object> arguments = new HashMap<String,Object>();
arguments.put("alternate-exchange","ALTERNATE_EXCHANGE"); // 指定交换机的备份交换机

channel.exchangeDeclare("TEST_EXCHANGE","topic", false, false, false, arguments);
```

参考：

com.gupaoedu.returnlistener

3、确保消息在队列正确地存储

可能因为系统宕机、重启、关闭等等情况导致存储在队列的消息丢失，即③出现问题。

解决方案：

1、队列持久化

```
// String queue, boolean durable, boolean exclusive, boolean autoDelete, Map<String,
Object> arguments
channel.queueDeclare(QUEUE_NAME, true, false, false, null);
```

2、交换机持久化

```
// String exchange, boolean durable
channel.exchangeDeclare("MY_EXCHANGE","true");
```

3、消息持久化

```
AMQP.BasicProperties properties = new AMQP.BasicProperties.Builder()
        .deliveryMode(2)   // 2代表持久化，其他代表瞬态
        .build();

channel.basicPublish("", QUEUE_NAME, properties, msg.getBytes());
```

4、集群，镜像队列，参考下一节

4、确保消息从队列正确地投递到消费者

如果消费者收到消息后未来得及处理即发生异常，或者处理过程中发生异常，会导致④失败。

为了保证消息从队列可靠地达到消费者，RabbitMQ提供了消息确认机制（message acknowledgement）。消费者在订阅队列时，可以指定autoAck参数，当autoAck等于false时，RabbitMQ会等待消费者显式地回复确认信号后才从队列中移去消息。

如果消息消费失败，也可以调用Basic.Reject或者Basic.Nack来拒绝当前消息而不是确认。如果requeue参数设置为true，可以把这条消息重新存入队列，以便发给下一个消费者（当然，只有一个消费者的时候，这种方式可能会出现无限循环重复消费的情况，可以投递到新的队列中，或者只打印异常日志）。

5、消费者回调

消费者处理消息以后，可以再发送一条消息给生产者，或者调用生产者的API，告知消息处理完毕。

参考：二代支付中异步通信的回执，多次交互。某提单APP，发送碎屏保消息后，消费者必须回调API。

6、补偿机制

对于一定时间没有得到响应的消息，可以设置一个定时重发的机制，但要控制次数，比如最多重发3次，否则会造成消息堆积。

参考：ATM存款未得到应答时发送5次确认；ATM取款未得到应答时，发送5次冲正。根据业务表状态做一个重发。

7、消息幂等性

服务端是没有这种控制的，只能在消费端控制。

如何避免消息的重复消费？

消息重复可能会有两个原因：

1、生产者的问题，环节①重复发送消息，比如在开启了Confirm模式但未收到确认。

2、环节④出了问题，由于消费者未发送ACK或者其他原因，消息重复投递。

对于重复发送的消息，可以对每一条消息生成一个唯一的业务ID，通过日志或者建表来做重复控制。

参考：银行的重账控制环节。

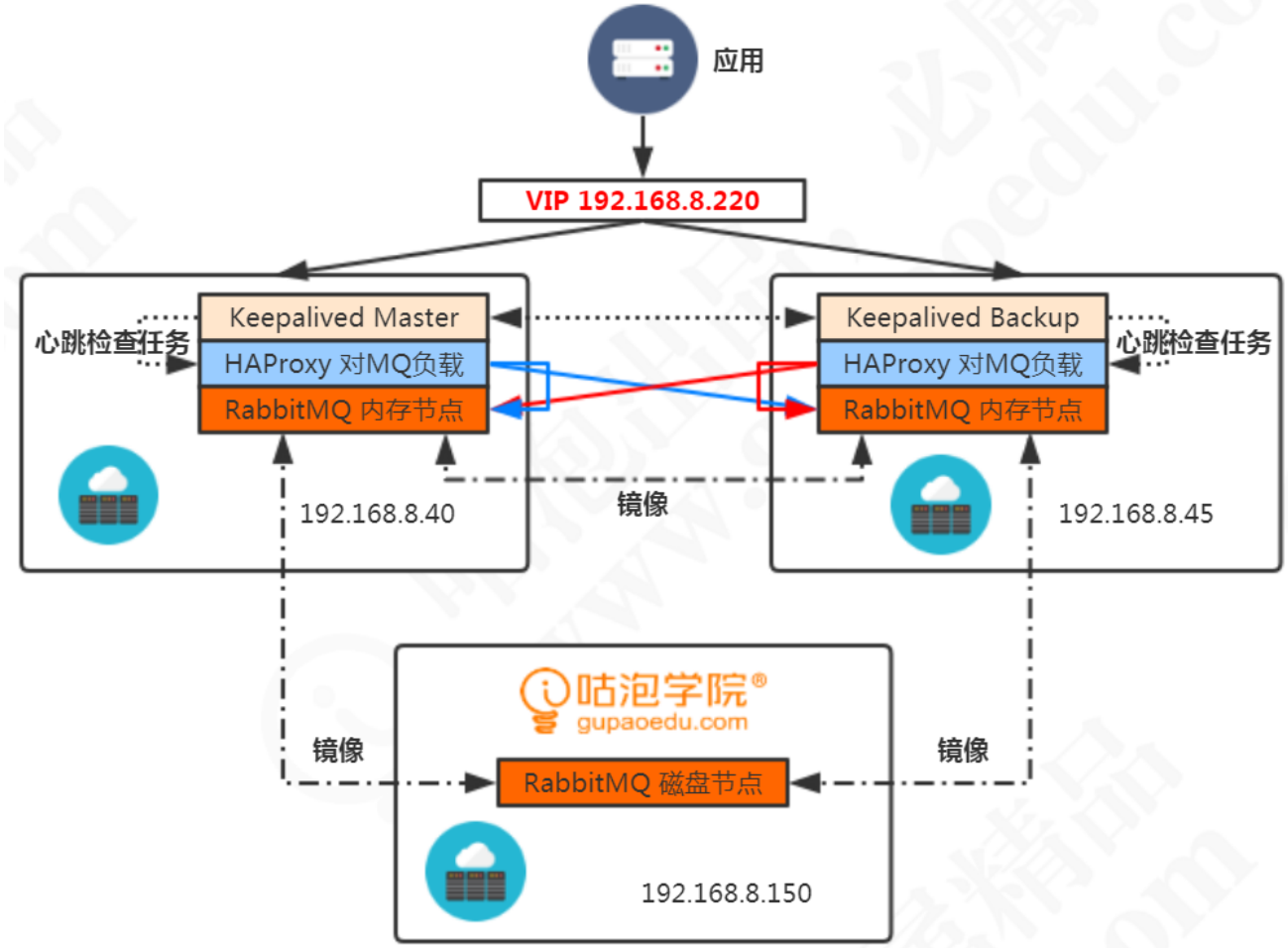
8、消息的顺序性

消息的顺序性指的是消费者消费的顺序跟生产者产生消息的顺序是一致的。

在RabbitMQ中，一个队列有多个消费者时，由于不同的消费者消费消息的速度是不一样的，顺序无法保证。

参考：消息：1、新增门店 2、绑定产品 3、激活门店，这种情况下消息消费顺序不能颠倒。

高可用架构



RabbitMQ集群

集群主要用于实现高可用与负载均衡。

RabbitMQ通过/var/lib/rabbitmq/.erlang.cookie来验证身份，需要在所有节点上保持一致。

集群有两种节点类型，一种是磁盘节点，一种是内存节点。集群中至少需要一个磁盘节点以实现元数据的持久化，未指定类型的情况下，默认为磁盘节点。

集群通过25672端口两两通信，需要开放防火墙的端口。

需要注意的是，RabbitMQ集群无法搭建在广域网上，除非使用federation或者shovel等插件。

集群的配置步骤：

1、配置hosts

2、同步erlang.cookie

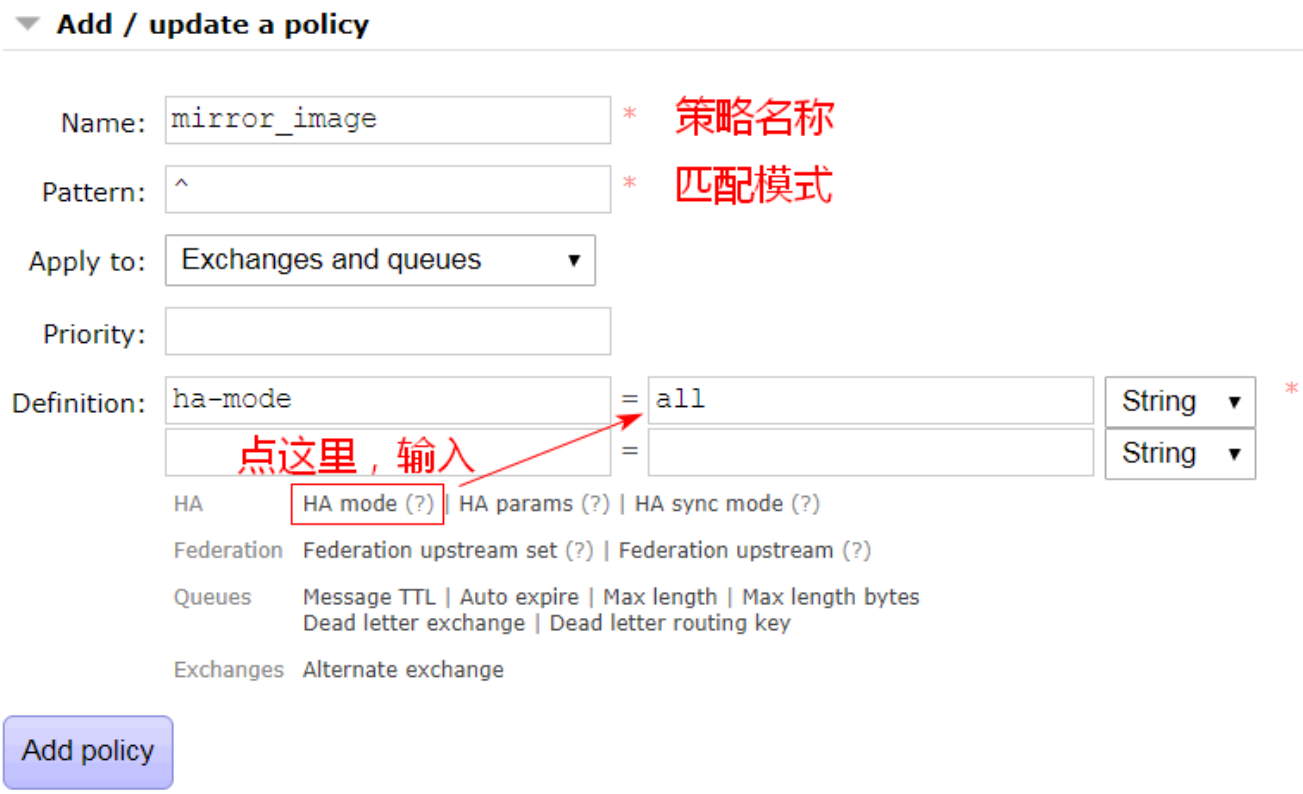
3、加入集群

RabbitMQ镜像队列

集群方式下，队列和消息是无法在节点之间同步的，因此需要使用RabbitMQ的镜像队列机制进行同步。

操作方式	命令或步骤
rabbitmqctl (Windows)	rabbitmqctl set_policy ha-all "^ha." "{""ha-mode"":""all""}"
HTTP API	PUT /api/policies/%2f/ha-all {"pattern":"^ha.", "definition":{"ha-mode":"all"}}
Web UI	Navigate to Admin > Policies > Add / update a policy Name输入：mirror_image Pattern输入：^（代表匹配所有） Definition点击 HA mode，右边输入：all

如图：



参考资料：

RabbitMQ之镜像队列

HAproxy负载+Keepalived高可用

在两个内存节点上安装HAProxy

```
yum install haproxy
```

编辑配置文件

```
vim /etc/haproxy/haproxy.cfg
```

内容修改为：

```
global
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
    stats socket /var/lib/haproxy/stats

defaults
    log                     global
    option                  dontlognull
    option                  redispatch
    retries                 3
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    maxconn                 3000

listen http_front
        mode http
        bind 0.0.0.0:1080           #监听端口
        stats refresh 30s           #统计页面自动刷新时间
        stats uri /haproxy?stats    #统计页面url
        stats realm Haproxy Manager #统计页面密码框上提示文本
        stats auth admin:123456     #统计页面用户名和密码设置

listen rabbitmq_admin
    bind 0.0.0.0:15673
    server node1 192.168.8.40:15672
    server node2 192.168.8.45:15672

listen rabbitmq_cluster 0.0.0.0:5673
    mode tcp
    balance roundrobin
    timeout client 3h
    timeout server 3h
    timeout connect 3h
    server   node1 192.168.8.40:5672 check inter 5s rise 2 fall 3
    server   node2 192.168.8.45:5672 check inter 5s rise 2 fall 3
```

启动HAProxy

```
haproxy -f /etc/haproxy/haproxy.cfg
```

安装Keepalived

```
yum -y install keepalived
```

修改配置文件

```
vim /etc/keepalived/keepalived.conf
```

内容改成（物理网卡和当前主机IP要修改）：

```
global_defs {
   notification_email {
     acassen@firewall.loc
     failover@firewall.loc
     sysadmin@firewall.loc
   }
   notification_email_from Alexandre.Cassen@firewall.loc
   smtp_server 192.168.200.1
   smtp_connect_timeout 30
   router_id LVS_DEVEL
   vrrp_skip_check_adv_addr
   # vrrp_strict    # 注释掉，不然访问不到VIP
   vrrp_garp_interval 0
   vrrp_gna_interval 0
}
global_defs {
   notification_email {
     acassen@firewall.loc
     failover@firewall.loc
     sysadmin@firewall.loc
   }
   notification_email_from Alexandre.Cassen@firewall.loc
   smtp_server 192.168.200.1
   smtp_connect_timeout 30
   router_id LVS_DEVEL
   vrrp_skip_check_adv_addr
   # vrrp_strict    # 注释掉，不然访问不到VIP
   vrrp_garp_interval 0
   vrrp_gna_interval 0
}

# 检测任务
vrrp_script check_haproxy {
    # 检测HAProxy监本
    script "/etc/keepalived/script/check_haproxy.sh"
    # 每隔两秒检测
    interval 2
    # 权重
    weight 2
}

# 虚拟组
vrrp_instance haproxy {
    state MASTER # 此处为`主`，备机是 `BACKUP`
```

```
    interface ens33 # 物理网卡，根据情况而定
    mcast_src_ip 192.168.8.40 # 当前主机ip
    virtual_router_id 51 # 虚拟路由id，同一个组内需要相同
    priority 100 # 主机的优先权要比备机高
    advert_int 1 # 心跳检查频率，单位：秒
    authentication { # 认证，组内的要相同
        auth_type PASS
        auth_pass 1111
    }
    # 调用脚本
    track_script {
        check_haproxy
    }
    # 虚拟ip，多个换行
    virtual_ipaddress {
        192.168.8.201
    }
}
```

启动keepalived

```
keepalived -D
```

网络分区

为什么会出现分区？因为RabbitMQ对网络延迟非常敏感，为了保证数据一致性和性能，在出现网络故障时，集群节点会出现分区。

参考脑图.xmind

RabbitMQ Network Partitions

RabbitMQ Network Partitions 处理策略

模拟RabbitMQ网络分区

广域网的同步方案

federation插件

shovel插件

实践经验总结

1、配置文件与命名规范

集中放在properties文件中

体现元数据类型（_VHOST _EXCHANGE _QUEUE）；

体现数据来源和去向（XXX_TO_XXX）；

2、调用封装

可以对Template做进一步封装，简化消息的发送。

3、信息落库+定时任务

将需要发送的消息保存在数据库中，可以实现消息的可追溯和重复控制，需要配合定时任务来实现。

4、运维监控

参考：

zabbix系列zabbix3.4监控rabbitmq

5、插件

tracing

https://www.rabbitmq.com/plugins.html

```
 [root@rabbit1 ~]# rabbitmq-plugins list
 Configured: E = explicitly enabled; e = implicitly enabled
 | Status:   * = running on rabbit@rabbit1
 |/
[e*] amqp_client                       3.6.12
[e*] cowboy                            1.0.4
[e*] cowlib                            1.0.2
[   ] rabbitmq_amqp1_0                 3.6.12
[   ] rabbitmq_auth_backend_ldap       3.6.12
[   ] rabbitmq_auth_mechanism_ssl      3.6.12
[   ] rabbitmq_consistent_hash_exchange 3.6.12
[   ] rabbitmq_event_exchange          3.6.12
[   ] rabbitmq_federation              3.6.12
[   ] rabbitmq_federation_management   3.6.12
[   ] rabbitmq_jms_topic_exchange      3.6.12
[E*] rabbitmq_management               3.6.12
[e*] rabbitmq_management_agent         3.6.12
[   ] rabbitmq_management_visualiser   3.6.12
[   ] rabbitmq_mqtt                    3.6.12
[   ] rabbitmq_recent_history_exchange 3.6.12
[   ] rabbitmq_sharding                3.6.12
[   ] rabbitmq_shovel                  3.6.12
[   ] rabbitmq_shovel_management       3.6.12
[   ] rabbitmq_stomp                   3.6.12
[   ] rabbitmq_top                     3.6.12
[   ] rabbitmq_tracing                 3.6.12
[   ] rabbitmq_trust_store             3.6.12
[e*] rabbitmq_web_dispatch             3.6.12
[   ] rabbitmq_web_mqtt                3.6.12
[   ] rabbitmq_web_mqtt_examples       3.6.12
[   ] rabbitmq_web_stomp               3.6.12
[   ] rabbitmq_web_stomp_examples      3.6.12
[   ] sockjs                           0.3.4
```

6、如何减少连接数

合并消息的发送，建议单条消息不要超过4M（4096KB）

思考

消费者的集群或者微服务的多个实例，会不会重复接收消息？

生产者先发送消息还是先登记业务表？（打款错误的例子）

谁来创建对象（交换机、队列、绑定关系）？

重复创建会有什么问题？

持久化的队列和非持久化的交换机可以绑定吗？可以

如何设计一个MQ服务？ http://www.xuxueli.com/xxl-mq/#/

面试题

1、消息队列的作用与使用场景？

2、创建队列和交换机的方法？

3、多个消费者监听一个生产者时，消息如何分发？

4、无法被路由的消息，去了哪里？

5、消息在什么时候会变成Dead Letter（死信）？

6、RabbitMQ如何实现延迟队列？

7、如何保证消息的可靠性投递？

8、如何在服务端和消费端做限流？

9、如何保证消息的顺序性？

10、RabbitMQ的节点类型？

课后作业

1、安装Erlang、RabbitMQ 2、高可用集群搭建（可选） 3、编写 Java API 4、SpringBoot 集成RabbitMQ

参考资料

1、GitLib代码 2、安装文件 3、PDF书籍 4、常用命令