RabbitMQ集群架构全解

前言

本文将系统的介绍一下RabbitMQ集群架构的特点、异常处理、搭建和使用中要注意的一些细节。

知识点

一、为什么使用集群？

二、集群的特点

三、集群异常处理

四、集群节点类型

五、集群搭建方法

六、镜像队列

一、为什么使用集群？

内建集群作为RabbitMQ最优秀的功能之一，它的作用有两个：

1. 允许消费者和生产者在Rabbit节点崩溃的情况下继续运行；
2. 通过增加节点来扩展Rabbit处理更多的消息，承载更多的业务量；

二、集群的特点

RabbitMQ的集群是由多个节点组成的，但我们发现不是每个节点都有所有队列的完全拷贝。

RabbitMQ节点不完全拷贝特性

为什么默认情况下RabbitMQ不将所有队列内容和状态复制到所有节点？

有两个原因：

1. 存储空间——如果每个节点都拥有所有队列的完全拷贝，这样新增节点不但没有新增存储空间，反而增 加了更多的冗余数据。
2. 性能——如果消息的发布需安全拷贝到每一个集群节点，那么新增节点对网络和磁盘负载都会有增加， 这样违背了建立集群的初衷，新增节点并没有提升处理消息的能力，最多是保持和单节点相同的性能甚 至是更糟。

所以其他非所有者节点只知道队列的元数据，和指向该队列节点的指针。

三、集群异常处理

根据节点不无安全拷贝的特性，当集群节点崩溃时，该节点队列和关联的绑定就都丢失了，附加在该队列的消费者丢失了其订阅的信息，那么怎么处理这个问题呢？

这个问题要分为两种情况：

1. 消息已经进行了持久化，那么当节点恢复，消息也恢复了；
2. 消息未持久化，可以使用下文要介绍的双活冗余队列，镜像队列保证消息的可靠性；

四、集群节点类型

节点的存储类型分为两种：

- 磁盘节点
- 内存节点

磁盘节点就是配置信息和元信息存储在磁盘上，内次节点把这些信息存储在内存中，当然内次节点的性能是大大超越磁盘节点的。

单节点系统必须是磁盘节点，否则每次你重启RabbitMQ之后所有的系统配置信息都会丢失。

RabbitMQ要求集群中至少有一个磁盘节点，当节点加入和离开集群时，必须通知磁盘节点。

特殊异常：集群中唯一的磁盘节点崩溃了

如果集群中的唯一一个磁盘节点，结果这个磁盘节点还崩溃了，那会发生什么情况？

如果唯一磁盘的磁盘节点崩溃了，不能进行如下操作：

- 不能创建队列
- 不能创建交换器
- 不能创建绑定
- 不能添加用户
- 不能更改权限
- 不能添加和删除集群几点

总结：如果唯一磁盘的磁盘节点崩溃，集群是可以保持运行的，但你不能更改任何东西。

解决方案：在集群中设置两个磁盘节点，只要一个可以，你就能正常操作。

五、集群搭建方法

本章我们用Docker来创建RabbitMQ集群，一来是因为操作简便，二是因为可以更充分的利用服务器硬件资源，三来是Docker也是现在的主流部署方案，关于更多的Docker详情可以查看我的另一篇：《使用Docker部署RabbitMQ集群)》 接下来，进入我们的正文，集群搭建分为两步：

- 步骤一：安装多个RabbitMQ
- 步骤二：加入RabbitMQ节点到集群

步骤一：安装多个RabbitMQ

```
docker run -d --hostname rabbit1 --name myrabbit1 -p 15672:15672 -p 5672:5672 -e
RABBITMQ_ERLANG_COOKIE='rabbitcookie' rabbitmq:3.6.15-management

docker run -d --hostname rabbit2 --name myrabbit2 -p 5673:5672 --link myrabbit1:rabbit1 -e
RABBITMQ_ERLANG_COOKIE='rabbitcookie' rabbitmq:3.6.15-management
```

```
docker run -d --hostname rabbit3 --name myrabbit3 -p 5674:5672 --link myrabbit1:rabbit1 --link myrabbit2:rabbit2 -e RABBITMQ_ERLANG_COOKIE='rabbitcookie' rabbitmq:3.6.15-management
```

具体的参数含义，参见上文“启动RabbitMQ”部分。

注意点：

1. 多个容器之间使用“--link”连接，此属性不能少；
2. Erlang Cookie值必须相同，也就是RABBITMQ_ERLANG_COOKIE参数的值必须相同，原因见下文“配置相同Erlang Cookie”部分；

步骤二：加入RabbitMQ节点到集群

设置节点1：

```
docker exec -it myrabbit1 bash
rabbitmqctl stop_app
rabbitmqctl reset
rabbitmqctl start_app
exit
```

设置节点2，加入到集群：

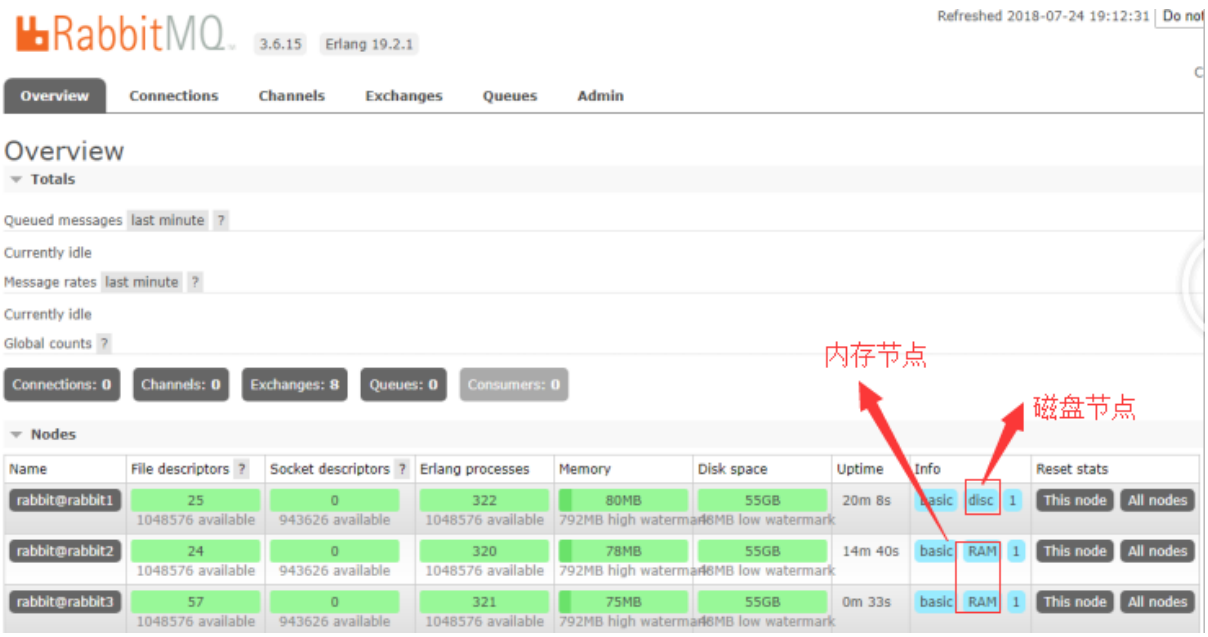
```
docker exec -it myrabbit2 bash
rabbitmqctl stop_app
rabbitmqctl reset
rabbitmqctl join_cluster --ram rabbit@rabbit1
rabbitmqctl start_app
exit
```

参数“--ram”表示设置为内存节点，忽略次参数默认为磁盘节点。

设置节点3，加入到集群：

```
docker exec -it myrabbit3 bash
rabbitmqctl stop_app
rabbitmqctl reset
rabbitmqctl join_cluster --ram rabbit@rabbit1
rabbitmqctl start_app
exit
```

设置好之后，使用http://物理机ip:15672 进行访问了，默认账号密码是guest/guest，效果如下图：



到此为止，我们已经完成了RabbitMQ集群的建立，启动了3个节点，1个磁盘节点和2个内存节点。

设置节点类型

如果你想更换节点类型可以通过命令修改，如下：

```
rabbitmqctl stop_app

rabbitmqctl change_cluster_node_type dist

rabbitmqctl change_cluster_node_type ram

rabbitmqctl start_app
```

移除节点

如果想要把节点从集群中移除，可使用如下命令实现：

```
rabbitmqctl stop_app

rabbitmqctl restart

rabbitmqctl start_app
```

集群重启顺序

集群重启的顺序是固定的，并且是相反的。如下所述：

- 启动顺序：磁盘节点 => 内存节点
- 关闭顺序：内存节点 => 磁盘节点

最后关闭必须是磁盘节点，不然可能回造成集群启动失败、数据丢失等异常情况。

六、镜像队列

镜像队列是Rabbit2.6.0版本带来的一个新功能，允许内建双活冗余选项，与普通队列不同，镜像节点在集群中的其他节点拥有从队列拷贝，一旦主节点不可用，最老的从队列将被选举为新的主队列。

镜像队列的工作原理：在某种程度上你可以将镜像队列视为，拥有一个隐藏的fanout交换器，它指示者信道将消息分发到从队列上。

设置镜像队列

设置镜像队列命令：“rabbitmqctl set_policy 名称 匹配模式（正则） 镜像定义”， 例如，设置名称为mypolicy的镜像队列，匹配所有名称是amp开头的队列都存储在2个节点上的命令如下：

```
rabbitmqctl set_policy mypolicy "^amp*" '{"ha-mode":"exactly","ha-params":2}'
```

可以看出**设置镜像队列，一共有三个参数，每个参数用空格分割。**

1. 参数一：名称，可以随便填；
2. 参数二：队列名称的匹配规则，使用正则表达式表示；
3. 参数三：为镜像队列的主体规则，是json字符串，分为三个属性：ha-mode | ha-params | ha-sync-mode，分别的解释如下：

- ha-mode：镜像模式，分类：all/exactly/nodes，all存储在所有节点；exactly存储x个节点，节点的个数由ha-params指定；nodes指定存储的节点上名称，通过ha-params指定；
- ha-params：作为参数，为ha-mode的补充；
- ha-sync-mode：镜像消息同步方式：automatic（自动），manually（手动）；

设置好镜像队列存储2个节点的效果如下图：

查看镜像队列

```
rabbitmqctl list_policies
```

删除镜像队列

```
rabbitmqctl clear_policy
```