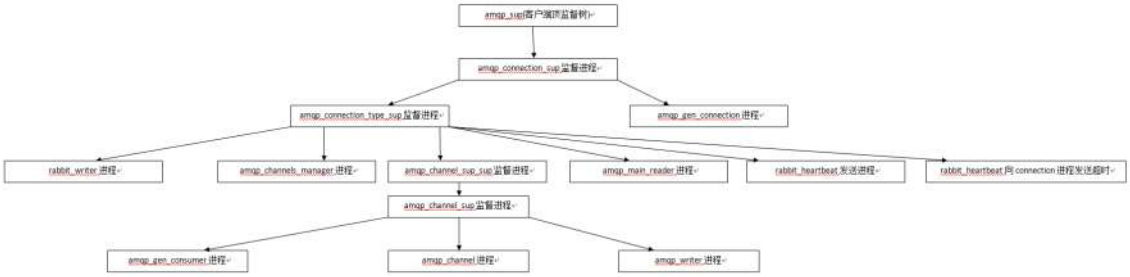
RabbitMQ系统客户端连接到RabbitMQ服务端消息通信过程

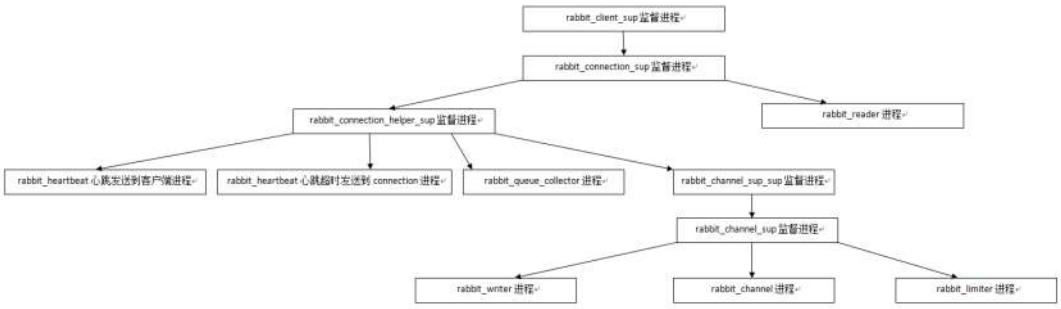
笔记本： RabbitMq源代码阅读系统原理分析
创建时间： 2015/6/24 20:17 **更新时间：** 2015/11/17 16:00
作者： 兴文哥

AMQP客户端讲程启动图：



AMQP客户端进程层级图.docx
2015/6/25 20:12, 13.8KB

RabbitMO系统客户端子讲程启动层级图：



RabbitMQ系统中客户端子系统进程层级图.docx
2015/6/25 20:13, 13.3KB

一.RabbitMQ AMQP客户端连接到RabbitMQ系统的过程

1.首先客户端调用amqp_connection_sup:start_link()在amqp_sup监督进程下启动amqp_connection_sup监督进程，在启动amqp_connection_sup监督进程的同时，在它下面启动amqp_gen_connection进程和amqp_connection_type_sup监督进程，然后调用amqp_gen_connection:connect()函数，在amqp_connection_type_sup监督进程下启动amqp_channel_sup_sup监督进程，amqp_channel_manager进程，rabbit_writer进程，amqp_main_reader进程(主要是接收RabbitMQ的数据，然后根据不用的channel发送到不同的channel进程).客户端通过gen_tcp:connect连接服务端，RabbitMQ服务端在rabbit_client_sup监督进程下启动rabbit_connection_sup监督进程，然后在rabbit_connection_sup监督进程下启动rabbit_connection_helper_sup监督进程和rabbit_reader进程，rabbit_reader进程主要是来获取客户端通过Socket发送过来的数据.

2.客户端连接后立刻发送RabbitMQ协议头(<<"AMQP", 0, 0 , 9, 1>>)给服务端，服务端接收

到该协议头握手成功后，向客户端发送connection.start消息（该消息里主要包括RabbitMQ服务端中的特性），该消息主要是发送服务器的特性给客户端，客户端接收到该消息后，立刻向RabbitMQ系统发送connection.start_ok消息给RabbitMQ服务器，该消息主要是发送客户端的特性(包括客户端的版本等信息，response字段里面有该连接的用户名以及用户对应的密码)到服务器

3.如果服务器在接收到的connection.start_ok消息里response字段的用户名和用户密码验证成功后，立即向客户端发送connection.tune消息，客户端接收到该消息后，对比下客户端服务端的频道最大个数，心跳时间间隔等配置得到最新的配置然后通过connection.tune_ok消息发送给RabbitMQ服务端，该过程主要是通过服务端和客户端的配置得到一个优化后的频道最大个数，心跳间隔时间等配置（就是同步一下服务端和客户端该connection连接的配置）客户端在接收到该消息的同时，会在amqp_connection_type_sup监督进程下启动一个间隔时间向服务端发送心跳消息的进程，另外还会启动一个定时向amqp_gen_connection进程发送心跳超时的消息,同时RabbitMQ服务器系统在接收到该消息后会在rabbit_connection_helper_sup监督进程下启动一个定时向客户端发送心跳消息的进程，另外还会启动一个定时向rabbit_reader进程发送心跳超时的进程

4.客户端发送完connection.tune_ok消息后，立刻的发送connection.open消息，RabbitMQ服务端接收到该消息后，检查该消息里发送的VirtualHost是否可用，如果可用，则在rabbit_connection_helper_sup监督进程启动rabbit_channel_sup_sup监督进程

5.客户端调用amqp_connection:open_channel()则到达amqp_channel_mananger进程，在该进程下调用amqp_channel_sup_sup监督进程的接口start_channel_sup后启动amqp_channel_sup监督进程，在启动该进程的同时，启动三个进程，amqp_gen_consumer进程，amqp_channel进程，rabbit_writer进程，然后发送channel.open消息到RabbitMQ服务端，服务端则在rabbit_connection_helper_sup监督进程下创建rabbit_channel_sup_sup监督进程下启动rabbit_channel_sup监督进程，同时在rabbit_channel_sup监督进程下rabbit_writer进程，rabbit_limiter进程，rabbit_channel进程

二.客户端创建一个频道的操作步骤

1.客户端调用amqp_connection:open_channel函数，实际调用的是amqp_gen_connection:open_channel函数，然后amqp_gen_connection进程会调用amqp_channels_manager进程去启动channel进程树相关进程，然后再amqp_channel_sup_sup监督进程下启动一个amqp_channel_sup监督进程，在启动该监督进程的时候，同时会在amqp_channel_sup监督进程下启动amqp_gen_consumer，amqp_channel，rabbit_writer进程.

2.客户端频道相关进程创建完毕后，在amqp_gen_connection模块让新创建的amqp_channel进程向RabbitMQ系统服务器发送channel.open消息

RabbitMQ概念性问题

Connection： 就是一个TCP的连接。Producer和Consumer都是通过TCP连接到RabbitMQ Server的。以后我们可以看到，程序的起始处就是建立这个TCP连接。

在上篇文章中提到，客户端连上rabbit后，需要向rabbit发送AMQP协议头，rabbit在收到协议头后，开始在0号channel 上跟客户端进行交互（AMQP中一个连接可以多路复用，1~65535为可用的channel编号，0号channel，也就是frame中 channel的索引为0，被认为是全局于整个连接）。

根据AMQP协议，经过**connection.start** -> connection.start_ok -> **connection.secure** -> connection.secure_ok -> **connection.tune** -> connection.tune_ok（这时rabbit会建立一个心跳进程）-> connection.open -> **connection.open_ok**后，客户端与rabbit之间就认为已经建立了连接（相关代码参见[$RABBIT_SRC/src/rabbit_reader.erl --> handle_method0/2]）。

（粗体指令由rabbit服务器发出）

Channels： 虚拟连接。它建立在上述的TCP连接中。数据流动都是在Channel中进行的。也就是说，一般情况是程序起始建立TCP连接，第二步就是建立这个Channel。

当客户端发来的frame中，channel的索引不为0时，rabbit认为这些数据从属于某个channel。如果该channel进程不 存在，则会创建一个channel进程（rabbit_channel，具体参见[$RABBIT_SRC/src/rabbit_reader.erl --> create_channel/2），并由此进程负责该channel上的所有数据（[$RABBIT_SRC/src /rabbit_reader.erl --> process_channel_frame/3）。

根据AMQP协议，经过channel.open -> **channel.open_ok**（[$RABBIT_SRC/src/rabbit_channel.erl --> handle_method/3）后，客户端就可以开始在该channel上发送数据了。

Exchange

当rabbit收到来自客户端的exchange.declare指令时，rabbit会根据客户端的参数创建一个exchange。首先 rabbit会向mnesia的表rabbit_exchange写入一条记录，包含客户端请求的exchange类型信息（默认4种类 型：direct，topic，fanout，head）及相关参数，如果exchange是需要持久化的（durable），则还需要向 rabbit_durable_exchange表中写入相同信息。然后，rabbit会通过rabbit_event发送 exchange_created的事件（统计作用）。

Queue

queue在创建时，需要确定要创建queue的类型（rabbit里称为backing_queue）：一般情况下，queue只在当前结点（客户端所连接的结点）创建，对应backing_queue为rabbit_variable_queue；当有HA策略时，queue需要在集群中的多个结点上创建（这时候，有master结点和slave结点之分，master结点未必是当前结点），master结点创建队列对应backing_queue为rabbit_mirror_queue_master，slave结点对应backing_queue为rabbit_mirror_queue_slave（master，slave实际最终也会创建一个rabbit_variable_queue）。创建一个队列首先会创建一个rabbit_amqqueue_process进程。然后同exchange类似，都需要先在mnesia表里写 入queue的基本信息（rabbit_queue，rabbit_durable_queue）。然后初始化对应的backing_queue，最后发 送queue_create事件。

我们来看一下backing_queue为rabbit_variable_queue时，初始化需要做什么：1）初始化queue的索引 （rabbit_queue_index:init/2）或者从以前的队列恢复索引（rabbit_queue_index:recover /5，durable队列）；2）创建message store或者恢复message store（rabbit_msg_store:client_init/4，恢复仅对于durable队列）。

rabbit_mirror_queue_master初始化：1）创建一个rabbit_mirror_queue_coordinator 及相应GM（Guaranteed Multicast）；2）获取该队列相关的镜像节点，并调用rabbit_mirror_queue_misc:add_mirror/2启动镜像队列进程；3）在当前结点（或者master结点）初始化一个rabbit_variable_queue队列；4）通过GM向所有镜像广播3中初始化队列的 长度。

rabbit_mirror_queue_misc:add_mirror/2启动镜像队列进程时，启动的是一个rabbit_mirror_queue_slave进程，相比rabbit_variable_queue，它只是多了个初始化GM的工作。最后也会初 始化一个rabbit_variable_queue队列。

Binding

用于将queue绑定到一个exchange。主要涉及到几个数据表写入（见下表，true或者false代表相应对象是不是durable），无其它复杂逻辑，写入完成后会发送binding_created事件。

exchage	queue	table
true	true	rabbit_durable_route

		rabbit_semi_durable_route rabbit_route rabbit_reverse_route
false	true	rabbit_semi_durable_route rabbit_route rabbit_reverse_route
true/false	false	rabbit_route rabbit_reverse_route

topic类型的exchange还需要将binding信息写入以下数据表：rabbit_topic_trie_edge，rabbit_topic_trie_binding，rabbit_topic_trie_node（基于trie数据结构，用于route key的匹配）。

为什么使用Channel，而不是直接使用TCP连接？

对于OS来说，建立和关闭TCP连接是有代价的，频繁的建立关闭TCP连接对于系统的性能有很大的影响，而且TCP的连接数也有限制，这也限制了系统处理 高并发的能力。但是，在TCP连接中建立Channel是没有上述代价的。对于Producer或者Consumer来说，可以并发的使用多个 Channel进行Publish或者Receive。有实验表明，1s的数据可以Publish10K的数据包。当然对于不同的硬件环境，不同的数据包 大小这个数据肯定不一样，但是我只想说明，对于普通的Consumer或者Producer来说，这已经足够了。如果不够用，你考虑的应该是如何细化 split你的设计。