**Linux下Rabbitmq的配置**

目录

[一 rabbitmq介绍 1](#_Toc485223021)

[1.1 Mq及其rabbitmq的概念 1](#_Toc485223022)

[1.2 应用场景 1](#_Toc485223023)

[1.3 系统架构 2](#_Toc485223024)

[二 linux下rabbitmq的搭建 3](#_Toc485223025)

[2.1 部署环境说明 3](#_Toc485223026)

[2.2 rabbitmq的安装 3](#_Toc485223027)

[2.2.1 Erlang安装配置 3](#_Toc485223028)

[2.2.2 下载安装rabbitmq 4](#_Toc485223029)

[2.3 验证rabbitmq消息的发送接收 7](#_Toc485223030)

[2.3.1 基本包的安装 7](#_Toc485223031)

[2.3.2 创建消息发送接收脚本 7](#_Toc485223032)

[2.3.3 测试消息的发送和接收 9](#_Toc485223033)

# 一 rabbitmq介绍

## Mq及其rabbitmq的概念

MQ全称为Message Queue, 消息队列(MQ)是一种应用程序对应用程序的通信方法。应用程序通过读写出入队列的消息(针对应用程序的数据)来通信，而无需专用连接来链接它们。消 息传递指的是程序之间通过在消息中发送数据进行通信，而不是通过直接调用彼此来通信，直接调用通常是用于诸如远程过程调用的技术。排队指的是应用程序通过 队列来通信。队列的使用除去了接收和发送应用程序同时执行的要求。

RabbitMQ是一个由erlang开发的在AMQP（Advanced Message Queue）基础上完整的，可复用的开源的企业消息系统

## 应用场景

在项目中，将一些无需即时返回且耗时的操作提取出来，进行了异步处理，而这种异步处理的方式大大的节省了服务器的请求响应时间，从而提高了系统的吞吐量。

对于一个大型的软件系统来说，它会有很多的组件或者说模块或者说子系统或者（subsystem or Component or submodule）。那么这些模块的如何通信？这和传统的IPC有很大的区别。传统的IPC很多都是在单一系统上的，模块耦合性很大，不适合扩展（Scalability）；如果使用socket那么不同的模块的确可以部署到不同的机器上，但是还是有很多问题需要解决。比如：

1）信息的发送者和接收者如何维持这个连接，如果一方的连接中断，这期间的数据如何方式丢失？

2）如何降低发送者和接收者的耦合度？

3）如何让Priority高的接收者先接到数据？

4）如何做到load balance？有效均衡接收者的负载？

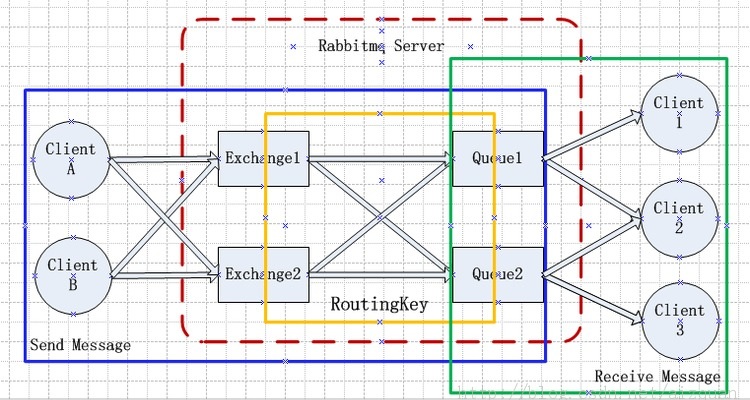
5）如何有效的将数据发送到相关的接收者？也就是说将接收者subscribe 不同的数据，如何做有效的filter。

6）如何做到可扩展，甚至将这个通信模块发到cluster上？

7）如何保证接收者接收到了完整，正确的数据？

AMDQ协议解决了以上的问题，而RabbitMQ实现了AMQP。

## 系统架构



RabbitMQ Server： 也叫broker server，它不是运送食物的卡车，而是一种传输服务。原话是RabbitMQisn’t a food truck, it’s a delivery service. 他的角色就是维护一条从Producer到Consumer的路线，保证数据能够按照指定的方式进行传输。但是这个保证也不是100%的保证，但是对于普通的应用来说这已经足够了。当然对于商业系统来说，可以再做一层数据一致性的guard，就可以彻底保证系统的一致性了。

Client A & B： 也叫Producer，数据的发送方。createmessages and publish (send) them to a broker server (RabbitMQ).一个Message有两个部分：payload（有效载荷）和label（标签）。payload顾名思义就是传输的数据。label是exchange的名字或者说是一个tag，它描述了payload，而且RabbitMQ也是通过这个label来决定把这个Message发给哪个Consumer。AMQP仅仅描述了label，而RabbitMQ决定了如何使用这个label的规则。

Client 1，2，3：也叫Consumer，数据的接收方。Consumersattach to a broker server (RabbitMQ) and subscribe to a queue。把queue比作是一个有名字的邮箱。当有Message到达某个邮箱后，RabbitMQ把它发送给它的某个订阅者即Consumer。当然可能会把同一个Message发送给很多的Consumer。在这个Message中，只有payload，label已经被删掉了。对于Consumer来说，它是不知道谁发送的这个信息的。就是协议本身不支持。但是当然了如果Producer发送的payload包含了Producer的信息就另当别论了。

对于一个数据从Producer到Consumer的正确传递，还有三个概念需要明确：exchanges, queues and bindings。

Exchanges are where producers publish their messages.

Queuesare where the messages end up and are received by consumers

Bindings are how the messages get routed from the exchange to particular queues.

还有几个概念是上述图中没有标明的，那就是Connection（连接），Channel（通道，频道）。

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。

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。

那么，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。

# 二 linux下rabbitmq的搭建

## 2.1 部署环境说明

|  |  |  |  |
| --- | --- | --- | --- |
| Ip地址 | 系统版本 | Rabbitmq版本 | 用途 |
| 10.0.80.12 | Centos 7 | 3.6.8 | Rabbitmq server |
| 10.10.11.250 | Centos 7 | ----- | 消息发送端和接收端 |

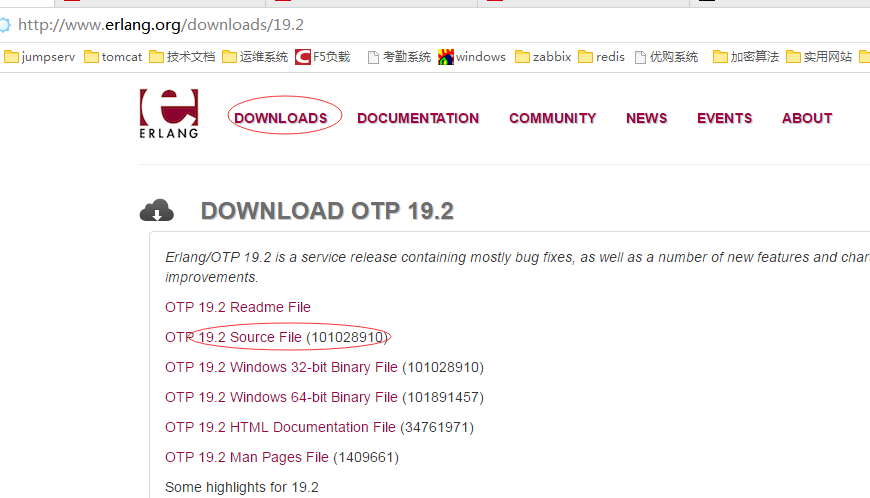
## 2.2 rabbitmq的安装

安装依赖包

[root@test-rabbitmq /]#yum install gcc glibc-devel make ncurses-devel openssl-devel xmlto

### 2.2.1 Erlang安装配置

从官网（http://www.erlang.org）下载安装包，我们选择19.2 的版本下载.tar.gz 的安装包



编译安装

[root@test-rabbitmq /]# cd /usr/local/

[root@test-rabbitmq local]# tar -xvf otp\_src\_19.2.tar.gz

[root@test-rabbitmq local]# cd otp\_src\_19.2

[root@test-rabbitmq local]# ./configure --prefix=/usr/local/erlang

[root@test-rabbitmq local]#make && make install

[root@test-rabbitmq local]#

检查安装结果,显示如下表示安装完成

[root@test-rabbitmq /]#cd /usr/local/erlang/bin

[root@test-rabbitmq bin]# erl

Erlang/OTP 19 [erts-8.2] [source] [64-bit] [async-threads:10] [hipe] [kernel-poll:false]

Eshell V8.2 (abort with ^G)

1>

配置Erlang环境变量,vi /etc/profile文件，增加下面的环境变量:

[root@test-rabbitmq /]# cat /etc/profile

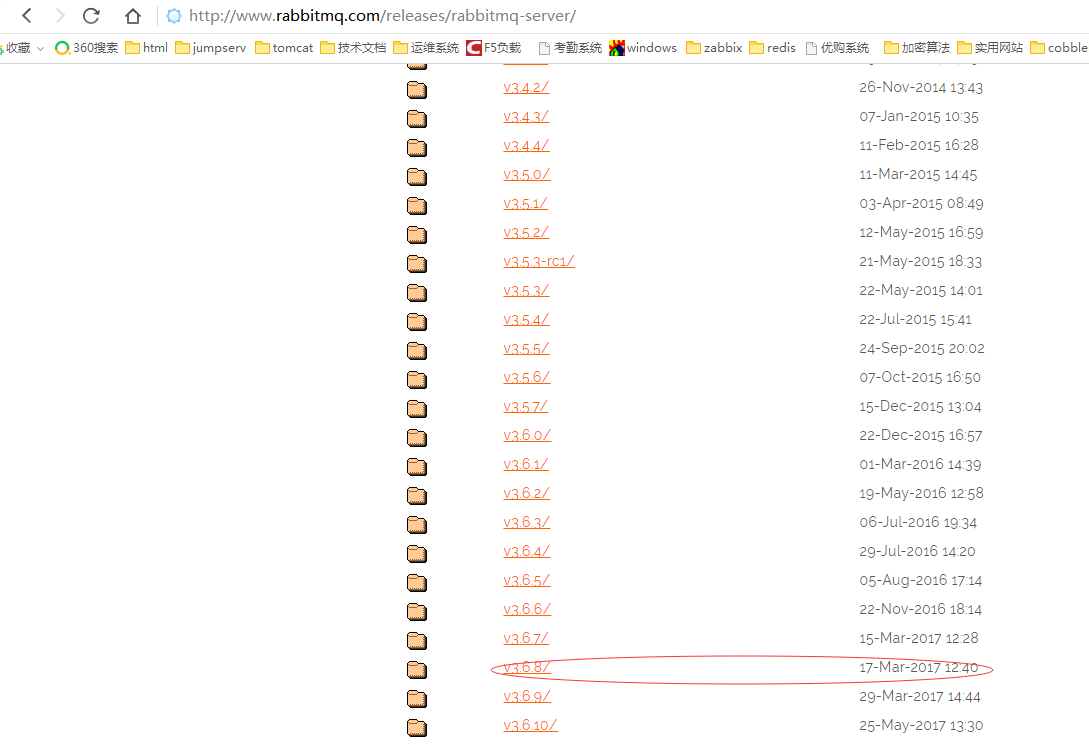
export PATH=$PATH:/usr/local/erlang/bin

source /etc/profile使得文件生效

[root@test-rabbitmq /]# source /etc/profile

### 2.2.2 下载安装rabbitmq

# 从官网（http://www.rabbitmq.com）下载rabbitmq，选择V3.6.8版本如下：



# 安装rabbitmq

[root@test-rabbitmq /]# cd /usr/local/

[root@test-rabbitmq local]# xz -d rabbitmq-server-generic-unix-3.6.8.tar.xz

[root@test-rabbitmq local]# tar -xvf rabbitmq-server-generic-unix-3.6.8.tar

[root@test-rabbitmq local]# mv rabbitmq\_server-3.6.8 rabbitmq

# 配置rabbitmq环境变量,vi /etc/profile文件，增加下面的环境变量:

[root@test-rabbitmq /]# cat /etc/profile

export PATH=$PATH:/usr/local/rabbitmq/sbin

source /etc/profile使得文件生效

[root@test-rabbitmq /]# source /etc/profile

# 默认情况下rabbitmq.conf配置文件不存在需要额外增加如下内容，但是必须符合标准的Erlang 配置文件

[root@test-rabbitmq sh]# cd /usr/local/rabbitmq/etc/rabbitmq/

[root@test-rabbitmq rabbitmq]# cat rabbitmq.config

[{rabbit, [{loopback\_users, []}]}].

[root@test-rabbitmq rabbitmq]#

如上增加的内容确保了mq支持远程连接

# rabbitmq.config的详细配置参数及其释义可以参考：<http://blog.csdn.net/super_rd/article/details/70327712>

# 启动rabbitmq服务

[root@test-rabbitmq ~]# rabbitmq-server –detached

关闭服务使用如下命令

[root@test-rabbitmq ~]#rabbitmqctl stop

# 启动网页插件：

[root@test-rabbitmq /]# rabbitmq-plugins enable rabbitmq\_management

# 配置远程访问

启动网页插件后rabbitmq就配置完成，其中15762位网页管理端口，5672位AMQP服务端口，但是默认网页是不允许访问的，需要增加一个用户并修改权限，如下：

# 添加用户 格式 rabbitmqctl add\_user {username} {password}

[root@test-rabbitmq /]#rabbitmqctl add\_user pengxin pengxin

# 添加权限

[root@test-rabbitmq /]# rabbitmqctl set\_permissions -p "/" pengxin ".\*" ".\*" ".\*"

# 修改用户角色

[root@test-rabbitmq /]# rabbitmqctl set\_user\_tags pengxin administrator

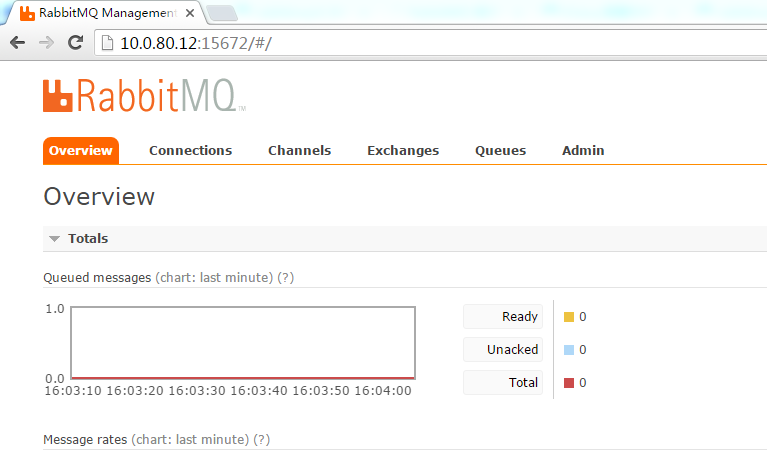
rabbitmqctl set\_permissions -p /myvhost tonyg "^tonyg-.\*" ".\*" ".\*"

此命令表示RabbitMQ broker授予tonyg 用户可访问 /myvhost虚拟主机,并在资源名称以"tonyg-"开头的所有资源上都具有配置权限，并在所有资源上都拥有读写权限。

验证远程登陆访问

登陆<http://10.0.80.12:15672>,用户名pengxin密码pengxin，登陆如下：





## 2.3 验证rabbitmq消息的发送接收

### 2.3.1 基本包的安装

RabbitMQ使用AMQP协议，因此我们需要一个AMQP协议库来与Rabbit通信。对于各种不同的语言会有不同的库可以选择。对于python来说，它也一样有amqp库，3个：py-amqplib、txAMQP、pika。在测试过程中我们采用pika库

在10.10.11.250上安装对应的包：

[root@pengxin /]# yum install python-pip git-core

安装pika

[root@pengxin /]# cd /usr/local/

[root@pengxin local]# wget <https://pypi.Python.org/packages/source/p/pika/pika-0.9.9.tar.gz>

[root@pengxin local]# tar -xvf pika-0.9.9.tar.gz

[root@pengxin local]# cd pika-0.9.9

[root@pengxin local]# python setup.py install

### 2.3.2 创建消息发送接收脚本

创建消息发送脚本和消息接收脚本

消息发送脚本：

[root@pengxin /]# cat /data/sh/rabbit-mq-send.py

#!/usr/bin/python

import pika

connection=pika.BlockingConnection(pika.ConnectionParameters(

host='10.0.80.12')) **#连接rabbitmq服务器**

channel=connection.channel()  **#生成管道，在管道里跑不同的队列**

**#声明queue，消息将在这个队列中进行传递。如果将消息发送到不存在的队列，rabbitmq将会自动清除这些消息**

**#如果加上durable=True，服务器异常时，消息不丢失，持久化**

channel.queue\_declare(queue='hello')

**#发送消息到上面声明的hello队列**

**#exchange表示交换器，能精确指定消息应该发送到哪个队列（先把数据发给exchange交换器,exchage再发给相应队列），routing\_key: 设置为队列的名称，body: 发送的内容**

channel.basic\_publish(exchange='',

routing\_key='hello', **#向"hello'队列发数据**

body='pengxin linux')  **#发的消息**

print" [x] Sent 'Hello World!'"

connection.close()

[root@pengxin /]#

消息接收脚本：

[root@pengxin /]# cat /data/sh/rabbit-mq-recivew.py

#!/usr/bin/python

import pika

connection=pika.BlockingConnection(pika.ConnectionParameters(

host='10.0.80.12'))

channel=connection.channel()  **#生成管道**

channel.queue\_declare(queue='hello') **#声明队列，保证程序不出错**

print' [\*] Waiting for messages. To exit press CTRL+C' **#无消息时显示的内容**

def callback(ch,method,properties, body): **#回调函数**

print " [x] Received %r"% (body,)

channel.basic\_consume(callback,

queue='hello',

**#无应答 如果是False，在处理完后应答，如果没应答，说明这次指令没执行完，下次继续发布**

**#可以防止消息丢失**

no\_ack=True)

channel.start\_consuming()

[root@pengxin /]#

### 2.3.3 测试消息的发送和接收

# 运行接收脚本，接收方一致是等待状态无收到的消息

[root@pengxin sh]# ./rabbit-mq-recivew.py

[\*] Waiting for messages. To exit press CTRL+C

#rabbitmq上消息发送的频率为0



# 运行客户端的脚本发送消息，重复执行多次，接收端有收到

发送端：

[root@pengxin sh]# ./rabbit-mq-send.py

^[[A [x] Sent 'Hello World!'

[root@pengxin sh]# ./rabbit-mq-send.py

^[[A [x] Sent 'Hello World!'

[root@pengxin sh]# ./rabbit-mq-send.py

^[[A [x] Sent 'Hello World!'

[root@pengxin sh]# ./rabbit-mq-send.py

^[[A [x] Sent 'Hello World!'

[root@pengxin sh]# ./rabbit-mq-send.py

[x] Sent 'Hello World!'

[root@pengxin sh]# ./rabbit-mq-send.py

[x] Sent 'Hello World!'

[root@pengxin sh]#

接收端：

[root@pengxin sh]# ./rabbit-mq-recivew.py

[\*] Waiting for messages. To exit press CTRL+C

[x] Received 'pengxin linux'

[x] Received 'pengxin linux'

[x] Received 'pengxin linux'

[x] Received 'pengxin linux'

[x] Received 'pengxin linux'

[x] Received 'pengxin linux'

#rabbitmq上有显示消息的发送频率

