分布式消息队列

NSQ原理与实践2

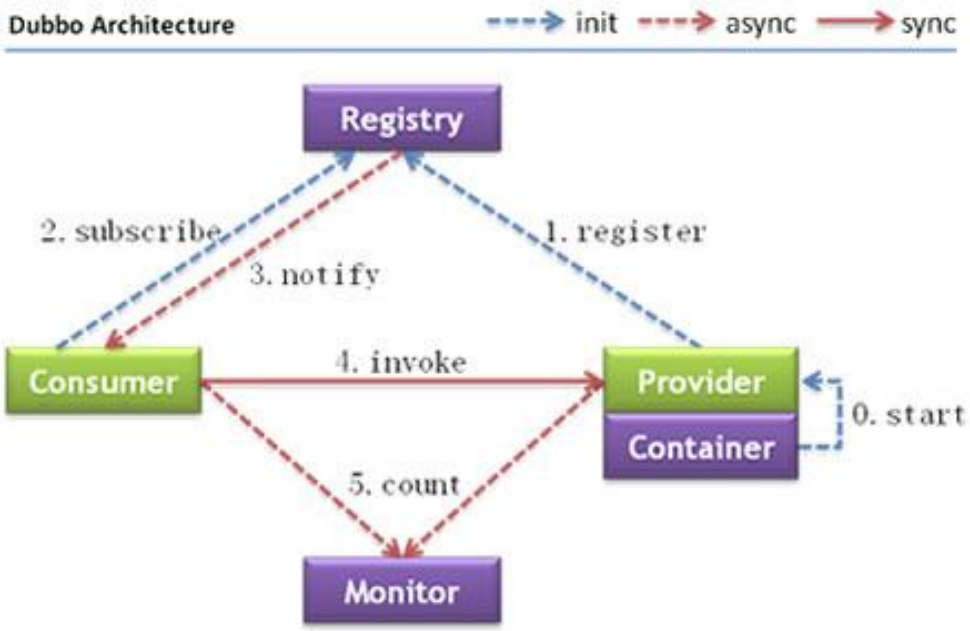
骆爽（imluoshuang@gmail.com）

NSQ解耦的架构

* Nsqlookupd 担负注册中心的作用
* Nsqd服务通过nsqlookupd的TCP端口进行连接
* Consumer和nsqadmin通过nsqlookupd的HTTP端口进行连接

图解

* 借用dubbo的架构图来展示
* Registry – nsqlookupd
* Consumer – consumer
* Provider – nsqd
* Monitor – nsqadmin等



启动服务的流程

启动nsqlookupd

* 默认监听在4160和4161两个端口上，分别提供TCP和HTTP服务
* 在没有nsqd实例连接上来之前，没有其他log输出

```
[devop@gitlab ~]$ nsqlookupd
[nsqlookupd] 2014/11/03 16:07:12.748745 nsqlookupd v0.3.0-alpha (built w/go1.3)
[nsqlookupd] 2014/11/03 16:07:12.749467 TCP: listening on [::]:4160
[nsqlookupd] 2014/11/03 16:07:12.749496 HTTP: listening on [::]:4161
```

消费者先于生产者连接的情况

* 一般来说，应该生产者先于消费者启动
* 如果不然，则消费者或得到TOPIC_NOT_FOUND并过段时间重试

```
[devop@gitlab ~]$ nsq_tail -lookupd-http-address=127.0.0.1:4161 -topic=test -channel=tail1
2014/11/03 16:08:50 INF    1 [test/tail1] querying nsqlookupd http://127.0.0.1:4161/lookup?topic=test
2014/11/03 16:08:50 ERR    1 [test/tail1] error querying nsqlookupd (http://127.0.0.1:4161/lookup?topic=te
- got response 404 Not Found "{\"message\":\"TOPIC_NOT_FOUND\"}"
```

启动生产者

* Nsqd有许多条启动配置项，之前我们仅使用了data-path来指定数据文件的存储位置，现在我们如下启动nsqd实例。

启动生产者

```
[devop@gitlab ~]$ nsqd -data-path=nsqdata/ -broadcast-address=10.24.177.86 -lookupd-tcp-address=127.0.0.1:4
[nsqd] 2014/11/03 16:12:37.491327 nsqd v0.3.0-alpha (built w/go1.3)
[nsqd] 2014/11/03 16:12:37.491391 ID: 161
[nsqd] 2014/11/03 16:12:37.494321 TOPIC(test): created
[nsqd] 2014/11/03 16:12:37.494506 TOPIC(test): new channel(tail1)
[nsqd] 2014/11/03 16:12:37.494606 TOPIC(test): new channel(tail2)
[nsqd] 2014/11/03 16:12:37.494651 NSQ: persisting topic/channel metadata to nsqdata/nsqd.161.dat
[nsqd] 2014/11/03 16:12:37.518896 LOOKUP: adding peer 127.0.0.1:4160
[nsqd] 2014/11/03 16:12:37.518919 LOOKUP connecting to 127.0.0.1:4160
[nsqd] 2014/11/03 16:12:37.519027 TCP: listening on [::]:4150
[nsqd] 2014/11/03 16:12:37.519056 HTTP: listening on [::]:4151
[nsqd] 2014/11/03 16:12:37.519775 LOOKUPD(127.0.0.1:4160): peer info {TcpPort:4160 HttpPort:4161 Version:0.
-alpha BroadcastAddress:gitlab}
[nsqd] 2014/11/03 16:12:37.519831 LOOKUPD(127.0.0.1:4160): topic REGISTER test
[nsqd] 2014/11/03 16:12:37.519898 NSQ: persisting topic/channel metadata to nsqdata/nsqd.161.dat
[nsqd] 2014/11/03 16:12:37.548640 LOOKUPD(127.0.0.1:4160): REGISTER test tail1
[nsqd] 2014/11/03 16:12:37.548960 LOOKUPD(127.0.0.1:4160): REGISTER test tail2
[nsqd] 2014/11/03 16:12:37.549170 LOOKUPD(127.0.0.1:4160): channel REGISTER test tail1
[nsqd] 2014/11/03 16:12:37.549248 NSQ: persisting topic/channel metadata to nsqdata/nsqd.161.dat
[nsqd] 2014/11/03 16:12:37.549589 LOOKUPD(127.0.0.1:4160): channel REGISTER test tail2
[nsqd] 2014/11/03 16:12:37.564629 NSQ: persisting topic/channel metadata to nsqdata/nsqd.161.dat
```

启动生产者

* 其中lookupd-tcp-address很容易理解
* 而broadcast-address指的是向nsqlookupd注册的提供服务的地址，如果不提供，则默认使用hostname
* 在生产环境中，我们一般不使用hostname来定位服务器，因此提供该机器的内网/外网IP是合适的

观察nsqlookupd和nsqd

* TCP长连接建立了，nsqd定期向nsqlookupd发送心跳信号

```
[nsqlookupd] 2014/11/03 16:12:37.519189 TCP: new client(127.0.0.1:60073)
[nsqlookupd] 2014/11/03 16:12:37.519219 CLIENT(127.0.0.1:60073): desired protocol magic '  V1'
[nsqlookupd] 2014/11/03 16:12:37.519407 CLIENT(127.0.0.1:60073): IDENTIFY Address:10.24.177.86 TCP:4150 HTT
[nsqlookupd] 2014/11/03 16:12:37.519429 DB: client(127.0.0.1:60073) REGISTER category:client key: subkey:
[nsqlookupd] 2014/11/03 16:12:37.519993 DB: client(127.0.0.1:60073) REGISTER category:topic key:test subkey
[nsqlookupd] 2014/11/03 16:12:37.548816 DB: client(127.0.0.1:60073) REGISTER category:channel key:test subk
[nsqlookupd] 2014/11/03 16:12:37.549070 DB: client(127.0.0.1:60073) REGISTER category:channel key:test subk
[nsqlookupd] 2014/11/03 16:12:52.520187 CLIENT(127.0.0.1:60073): pinged (last ping 15.000773417s)
[nsqlookupd] 2014/11/03 16:13:07.520167 CLIENT(127.0.0.1:60073): pinged (last ping 14.999988186s)
```

```
[nsqd] 2014/11/03 16:12:52.520021 LOOKUPD(127.0.0.1:4160): sending heartbeat
[nsqd] 2014/11/03 16:13:07.520017 LOOKUPD(127.0.0.1:4160): sending heartbeat
```

观察消费者和nsqd

* 消费者在生产者注册后的一次轮询中连接到了nsqd
* 随后依然保持轮询，使其能够尽快从nsqlookupd获知最新的生产者信息（新增/减少等）

```
2014/11/03 16:11:50 ERR    1 [test/tail1] error querying nsqlookupd (http://127.0.0.1:4161/lookup?topic=te
- got response 404 Not Found "{\"message\":\"TOPIC_NOT_FOUND\"}"
2014/11/03 16:12:50 INF    1 [test/tail1] querying nsqlookupd http://127.0.0.1:4161/lookup?topic=test
2014/11/03 16:12:50 INF    1 [test/tail1] (10.24.177.86:4150) connecting to nsqd
2014/11/03 16:13:50 INF    1 [test/tail1] querying nsqlookupd http://127.0.0.1:4161/lookup?topic=test
2014/11/03 16:14:50 INF    1 [test/tail1] querying nsqlookupd http://127.0.0.1:4161/lookup?topic=test
```

```
[nsqd] 2014/11/03 16:12:50.836021 TCP: new client(10.24.177.86:51455)
[nsqd] 2014/11/03 16:12:50.836117 CLIENT(10.24.177.86:51455): desired protocol magic '  V2'
[nsqd] 2014/11/03 16:12:50.836477 [10.24.177.86:51455] IDENTIFY: {ShortId:gitlab LongId:gitlab ClientID:gi
 Hostname:gitlab HeartbeatInterval:30000 OutputBufferSize:16384 OutputBufferTimeout:250 FeatureNegotiation
e TLSv1:false Deflate:false DeflateLevel:6 Snappy:false SampleRate:0 UserAgent:nsq_tail/0.3.0-alpha go-nsq/
.1-alpha MsgTimeout:0}
```

启动nsqadmin

* 同样不再具体制定nsqd实例，而是为nsqadmin设置了nsqlookupd的HTTP地址

```
[devop@gitlab ~]$ nsqadmin -lookupd-http-address=127.0.0.1:4161
[nsqadmin] 2014/11/03 16:19:46.885410 nsqlookupd v0.3.0-alpha (built w/go1.3)
[nsqadmin] 2014/11/03 16:19:46.903608 HTTP: listening on [::]:4171
```

结束服务的情况

从nsqlookupd解除注册

* Nsqd实例退出时会从nsqlookupd解除注册

```
[nsqlookupd] 2014/11/03 16:23:16.161546 CLIENT(127.0.0.1:60090): closing
[nsqlookupd] 2014/11/03 16:23:16.161582 DB: client(127.0.0.1:60090) UNREGISTER category:channel key:test su
y:tail1
[nsqlookupd] 2014/11/03 16:23:16.161596 DB: client(127.0.0.1:60090) UNREGISTER category:channel key:test su
y:tail2
[nsqlookupd] 2014/11/03 16:23:16.161606 DB: client(127.0.0.1:60090) UNREGISTER category:client key: subkey:
[nsqlookupd] 2014/11/03 16:23:16.161616 DB: client(127.0.0.1:60090) UNREGISTER category:topic key:test subk
[nsqlookupd] 2014/11/03 16:23:16.161627 ERROR: client(127.0.0.1:60090) - EOF
```

消费者被动断开nsqd连接

```
2014/11/03 16:23:16 ERR    1 [test/tail1] (10.24.177.86:4150) IO error - EOF
2014/11/03 16:23:16 INF    1 [test/tail1] (10.24.177.86:4150) beginning close
2014/11/03 16:23:16 INF    1 [test/tail1] (10.24.177.86:4150) readLoop exiting
2014/11/03 16:23:16 INF    1 [test/tail1] (10.24.177.86:4150) breaking out of writeLoop
2014/11/03 16:23:16 INF    1 [test/tail1] (10.24.177.86:4150) writeLoop exiting
2014/11/03 16:23:16 INF    1 [test/tail1] (10.24.177.86:4150) finished draining, cleanup exiting
2014/11/03 16:23:16 INF    1 [test/tail1] (10.24.177.86:4150) clean close complete
2014/11/03 16:23:16 WRN    1 [test/tail1] there are 0 connections left alive
```

其他

* 消费者主动断开连接不会对nsqlookupd产生任何影响，只有之前连接到的nsqd实例会得到通知

```
[nsqd] 2014/11/03 16:27:45.500425 PROTOCOL(V2): [10.24.177.86:51480] exiting ioloop
[nsqd] 2014/11/03 16:27:45.500500 ERROR: client(10.24.177.86:51480) - failed to read command - EOF
[nsqd] 2014/11/03 16:27:45.500519 PROTOCOL(V2): [10.24.177.86:51480] exiting messagePump
```

结论

* Nsqlookupd的存在，使得生产者和消费者都不需要知道彼此的实际地址，仅需注册/查询的操作就可以相互建立连接，实现了生产者和消费者的解耦。
* 因此多个nsqd实例可以对同一topic进行处理，只要其中一个存活，生产-消费过程就可以继续，从而避免了单点故障。

结论

* Nsqlookupd因此成为系统的核心，成为了新的可能的单点，但因其逻辑简单且负载较低、恢复无需任何配置参数，所以总体上降低了系统风险。
* 每个nsqd实例和消费者程序都可以同时连接到多个nsqlookupd，一个挂掉时，其他nsqlookupd依然能够提供全部服务。缺点是每个nsqd和消费者启动时都要配置多个nsqlookupd的地址。