作业帮一课

在家学的名师直播课

NMQ消息队列

服务学习

邓瑞龙

2019-09-05

纲要

作业帮一课
在家学的名师直播课

消息队列简述

NMQ简介

NMQ架构

NMQ集群

NMQ接口

NMQ双活

消息队列简述

作业帮一课
在家学的名师直播课

概念	特点	作用	分类
消息： - 动作 / 内容 - 动作 & 内容 队列： - FIFO 消息队列： - 用于进程间通信，完成接收存储转发消息的FIFO队列	时序性： - FIFO队列 一致性： - 消息一定会被处理	异步： - 进程间通信非阻塞处理 解耦： - 让进程关注于自身，对于其他进程的使用只关心通知而非处理 缓冲： - 流量消峰处理	点对点（P2P）： - Redis队列（List） 发布订阅（PS）： - Kafka – Apache - ActiveMQ - Apache - RabbitMQ – Rabbit - RocketMQ – 阿里 - ZeroMQ - NMQ – 百度（未开源）

NMQ简介

作业帮一课
在家学的名师直播课

服务简述	开发背景	设计考量	其他需求

名词解释：
- NMQ：New Message Queue

服务简述：
- 简单地说，NMQ接收上游提交过来的消息，将它保存起来，另一方面，又将储存起来的消息发送出去

起源大社区：
- 贴吧 知道 空间 文库等

避免造轮子：
- 重复开发、分散运维；极大的人力浪费

架构的发展：
- 让老的系统不在适合

业务的发展：
- 对性能、可扩展性有了更高的要求

数据安全性：其实还好

传输实时性：要求很高

吞吐的需求：很大

时序的需求：真的需要

消费方形态：多样

关联的形态：1vN

集中运维

服务解耦

运维平台化&自动化

功能完善：强大的时序+并发控制

数据互通：支持国际化数据互通

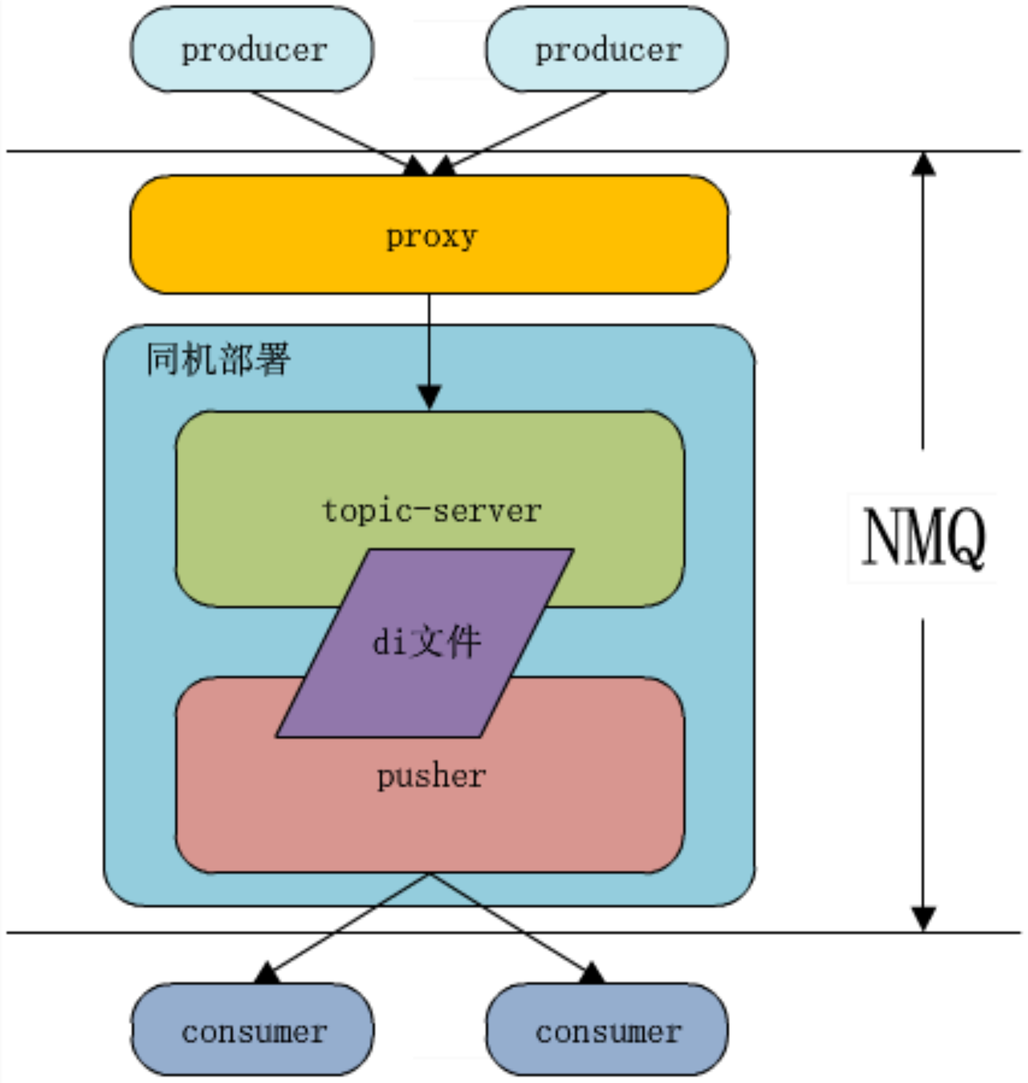
NMQ架构

组成部分

Proxy：
- 消息入口：以topic为单位进行分流，接收数据发送给指定的topic

Topic-Server：
- 消息存储：存放proxy传递过来的数据，并等待pusher读取

Pusher：
- 消息订阅：从topic读取数据，并按照配置进行处理

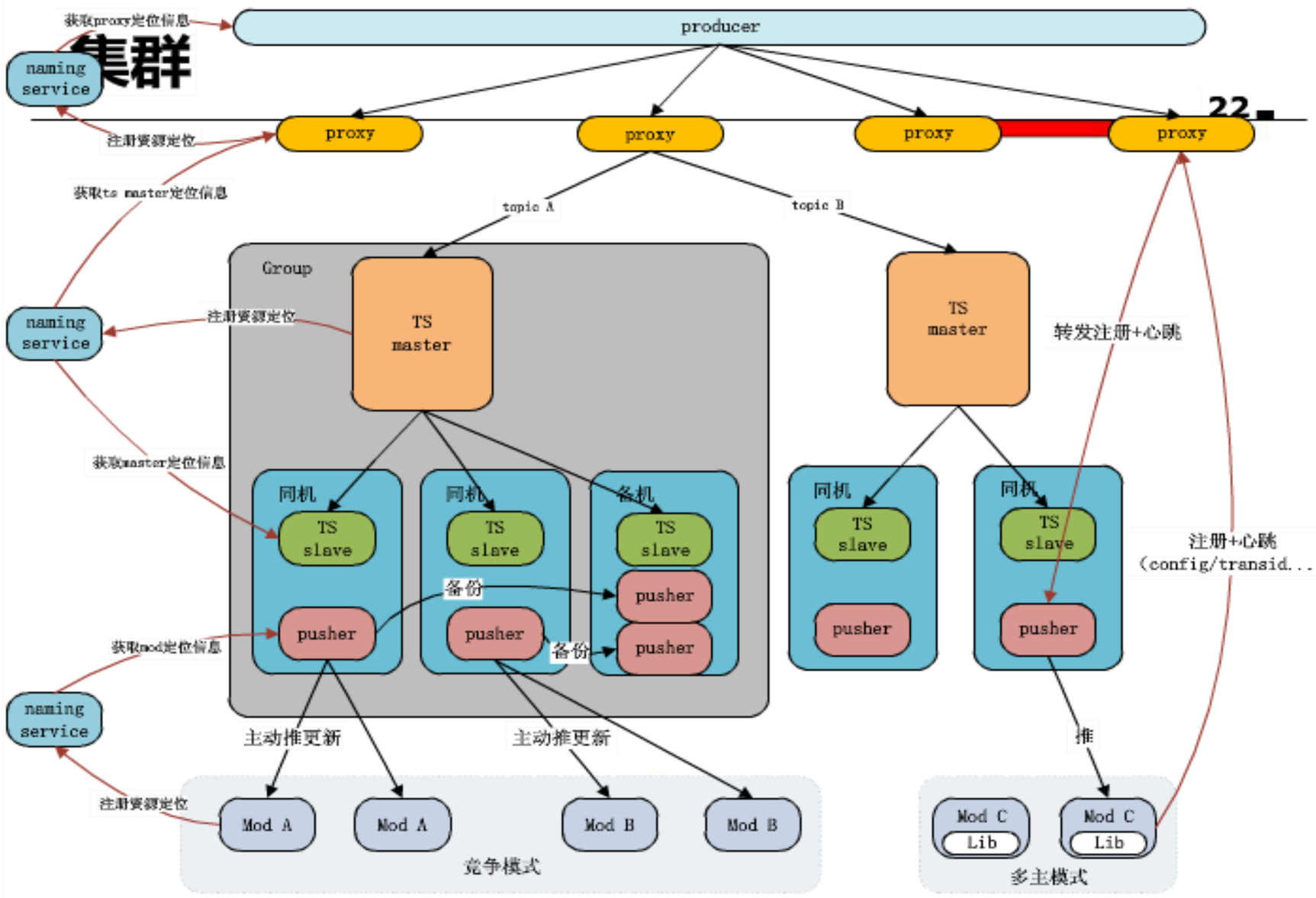


NMQ级联

服务扩容

垂直扩展：
- 问题：当接收同一个topic的consumer增多，导致pusher出现性能瓶颈
- 方案：可以通过ts级联扩展多个pusher解决，支持多级级联

水平扩展：
- 问题：当一个topic的命令增多，导致超过单机ts性能极限
- 方案：可以通过将该topic拆分到多个ts解决；比如按照某个partition key进行拆分；拆分后，只有相同pk的消息才能保证时序



NMQ接口

作业帮一课
在家学的名师直播课

生产者 短连接 → NmqProxy → Topic → Pusher → 长连接 消费者

- Prod指定存储哪个产品线
- Topic指定存储在哪个队列
- Cmd指定什么命令号

NmqProxy

接收协议：
nshead+mcpack1/2

接收内容：
- _product
- _topic
- _cmd
- 消息数据

主要配置: nmqproxy.conf
topic配置，指定对应某一topic的消息，发往哪一个nmqtopic

Topic

接收协议：
nshead+mcpack2

划分方式：根据业务类型划分

设置di位置：
topic/conf/topic.conf

主从配置：
replication.conf
is_master=1

Pusher

发送协议: http/mcpack
过滤方式: prod.topic.cmd

主要配置：
machine_xxx.conf：发送到哪里去（IP：PORT，或者bns）
module_xxx.conf：发送方式（http/mcpack）、该模块可以处理的消息（prod.topic.cmd）、发送uri的格式

主从配置: replication.conf

Prod.topic.cmd决定着下游如何消费，这个信令将透传到下游URI参数中，下游根据情况做适当的操作

NMQ目录结构&配置详情：http://wiki.afpai.com/x/_tYuAw

线上部署：单机模式

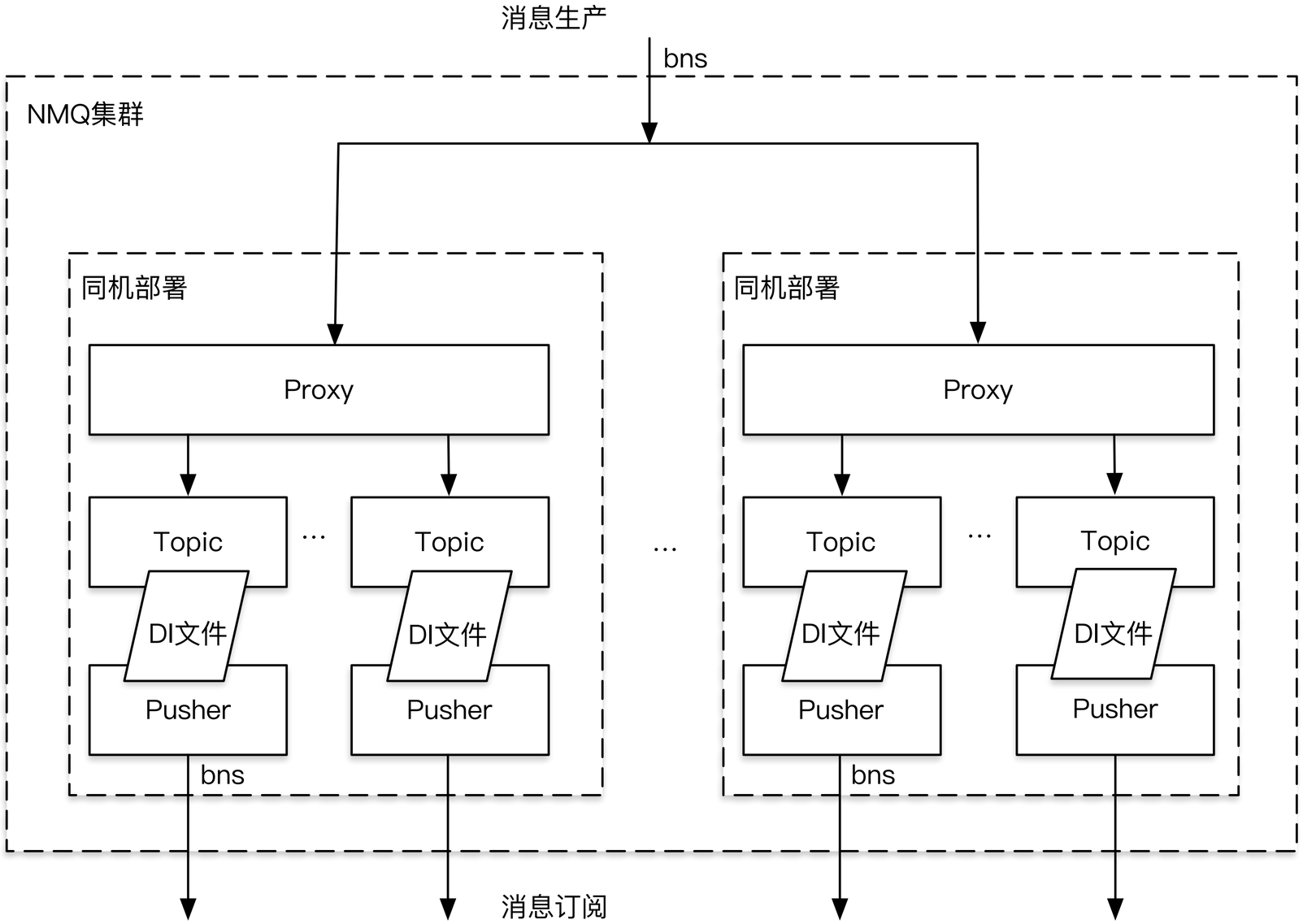
高性能：
没有额外的级联系统开销

易维护：
模式简单，易维护，易扩容

无关联：
各个消息之间无关联

低可靠：
机器损坏则未消费的消息永久丢失

异步性：
可接受单机服务宕机，重启之后下游还可以继续消费，对稳定性要求不高



NMQ实例：新增pusher服务

在pusher/conf/中增加两个文件machine_xxx.conf与module_xxx.conf

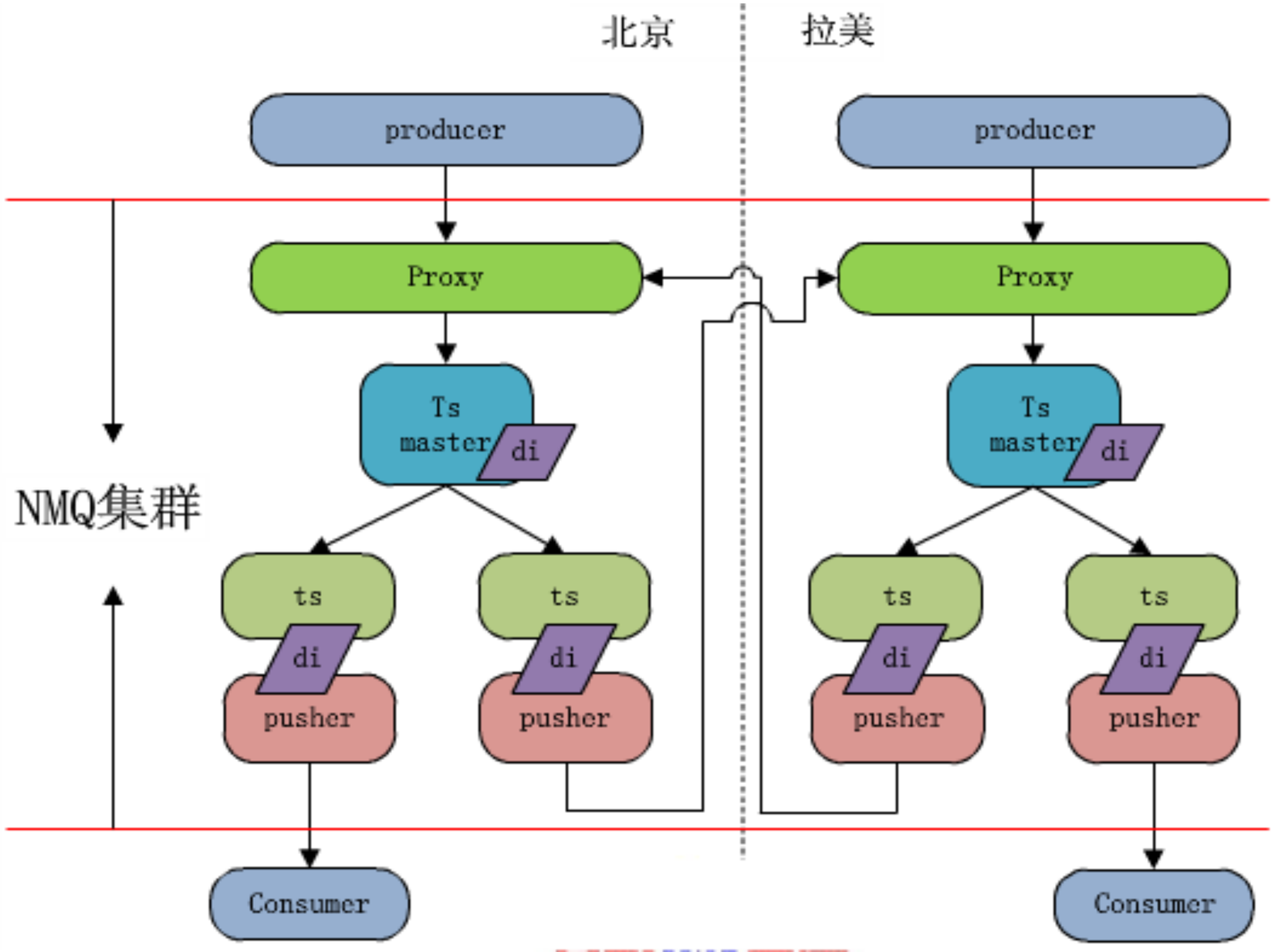
在pusher/conf/pusher_clients.conf中增加$include:machine_xxx.conf
在pusher/conf/pusher_talk.conf中增加$include:module_xxx.conf

如需新增命令号：直接在pusher/conf/module_xxx.conf中增加 @filter:_product._topic._cmd

```
[root@na-ldap-7-62 conf]# cat machine_pcassistant.conf
[UbClientConfig]
[.Naming]
[..@Service]
Name  : pcassistant
WebfootServiceName : mq-test.offline.qcvmbj4
```

```
[root@na-ldap-7-62 conf]# cat module_pcassistant.conf  | grep -v '^#' | grep -v '^\s*$'
[modules]
[.@module]
name : pcassistant
flag : 1
sending_type : 0
sending_protocol : http
sending_window_size : 2
sending_thread_num : 1
sending_retry_time: 500
[..msg_filter]
@filter : zb.core.830002
@filter : zb.core.210003
[..sequence_control]
mutex_key :
force_sequence_when_no_mutex_key : 0
[..ext_config]
custom_key : sample
[..http]
[...default_conf]
max_retry_times : -1
send_pack : 1
send_pack_type : 0
send_pack_key : data
server_redundancy_policy : 0
uri : /pcassistant/commit/commit?topic={{_topic}}&transid={{_transid}}&cmdno={{_cmd}}
http_header : User-Agent: NuSOAP/0.6.6\r\ncharset=UTF-8
```

NMQ双活-原生


北京
拉美
producer
producer
Proxy
Proxy
Ts
master
di
Ts
master
di
NMQ集群
ts
di
pusher
ts
di
pusher
ts
di
pusher
ts
di
pusher
Consumer
Consumer

NMQ最佳实践

作业帮一课
在家学的名师直播课

自助工具：
nmq-di-tool，读取di数据，或者根据时间戳查找消息的transid

最新序号：
cat topic/data/di/topic_di.last_write.transid

处理序号：
cat pusher/data/talk_status/xxx.status

命令拥堵：
talk_status序号长时间没有更新，说明命令拥堵

堵塞耗时：
NMQ平台定期检查talk_status，找出transid变化前后的时间戳。耗时粒度精确到分钟级

清理数据：
默认情况下，NMQ DI数据会无限写入，没有删除策略的话，会把磁盘打盘。线上删除策略：CT脚本定期删除DI数据目录，最少保留最近两个DI目录

作业帮一课

在家学的名师直播课

Q&A