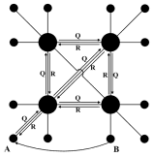
DATAGURU 炼数成金

Leader-us系列



大型分布式系统案例实战 第9周

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

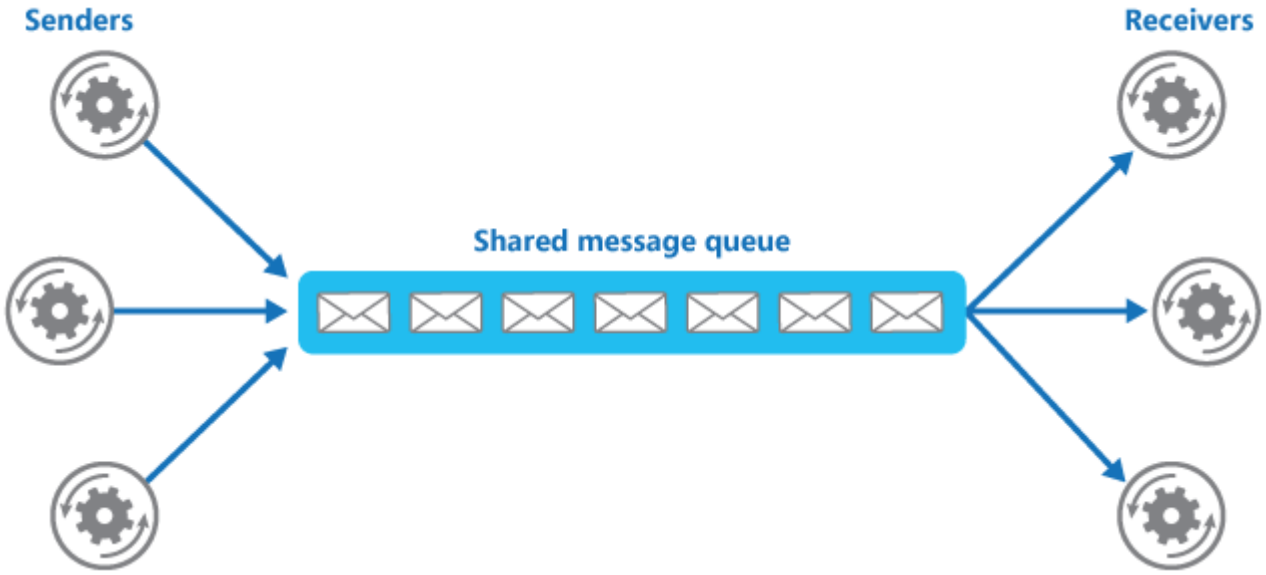
课程详情访问炼数成金培训网站

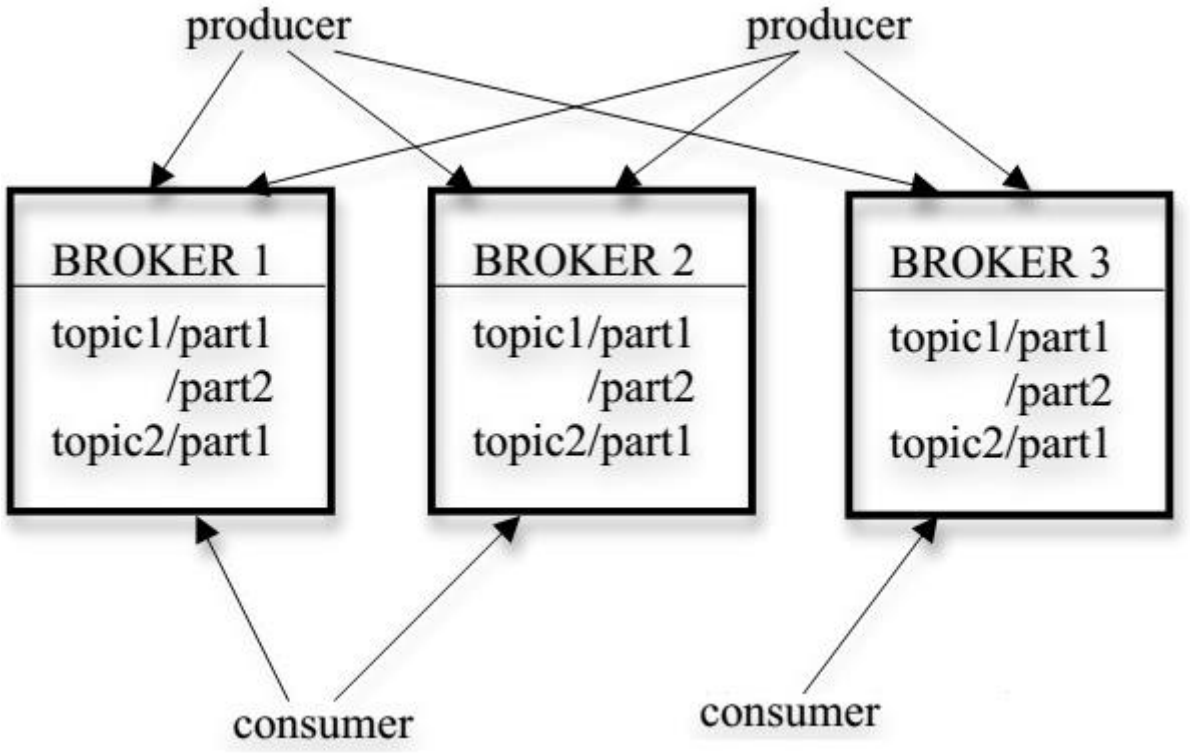
http://edu.dataguru.cn

分布式系统之消息队列

- 消息队列概述
- 经典的JMS模型
- STOMP\MQTT \AMQP
- 下一代消息队列——kafka
- 独行侠ZeroMQ

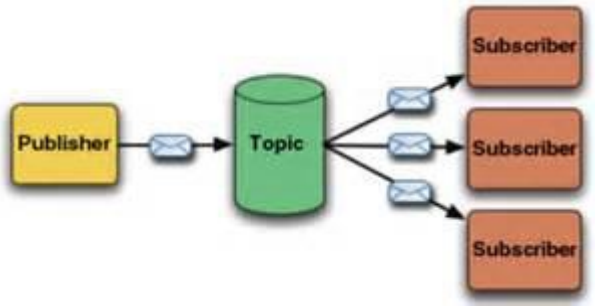
消息队列概述——概念和特性





排队——先进先出
异步——松耦合
弹性——消息可以适度积压

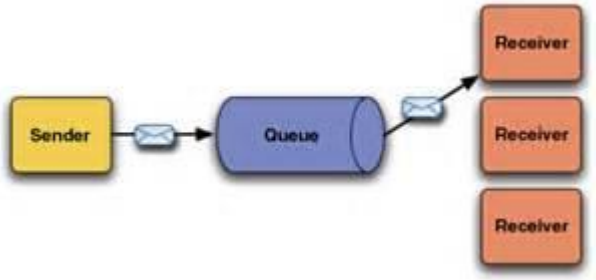
消息队列概述——Topic模式与订阅模式



topic是广播的形式，一个topic地址的多个sub都能收到

topic数据默认不落地，是无状态的

并不保证publisher发布的每条数据，Subscriber都能接受到。

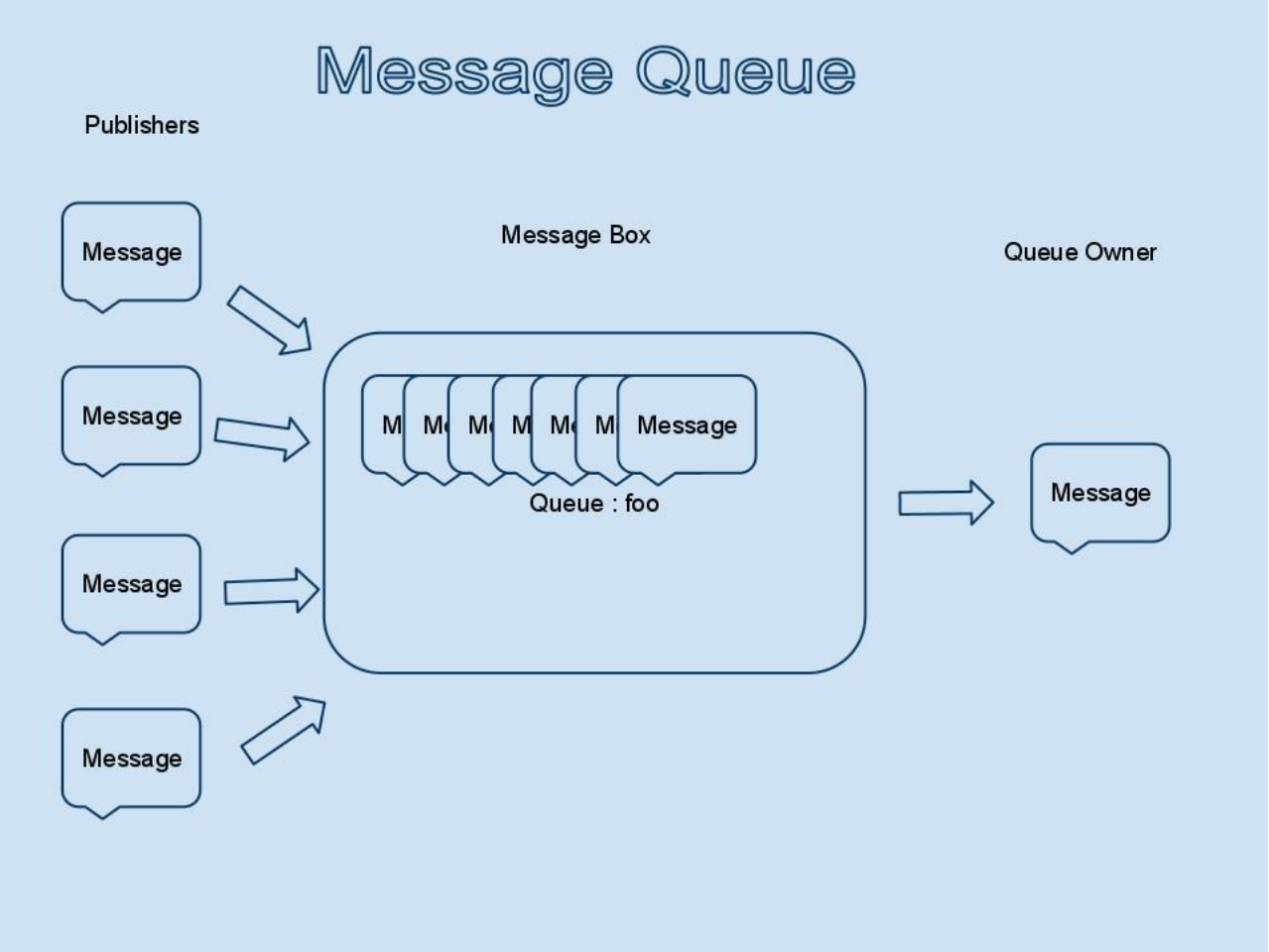


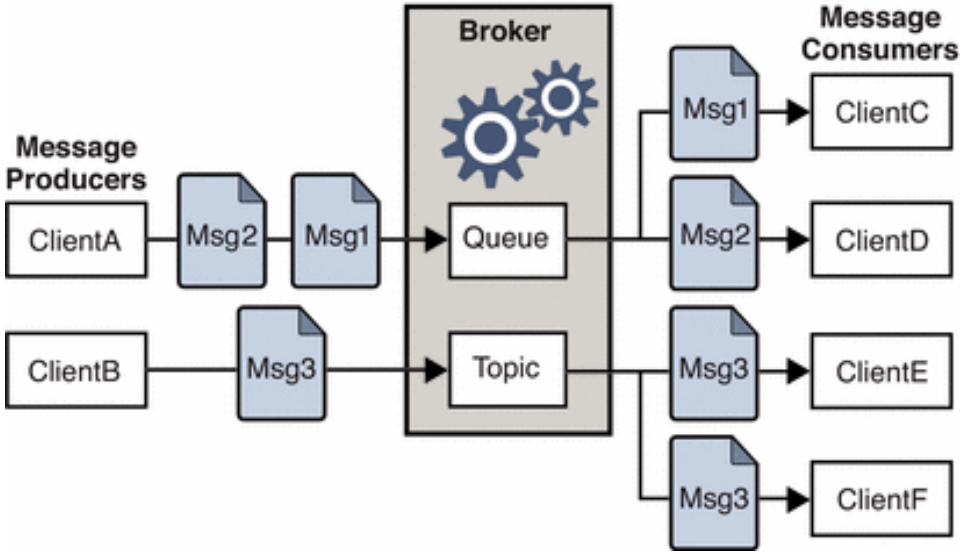
Point-to-Point 点对点

Queue数据默认会在mq服务器上以文件形式保存

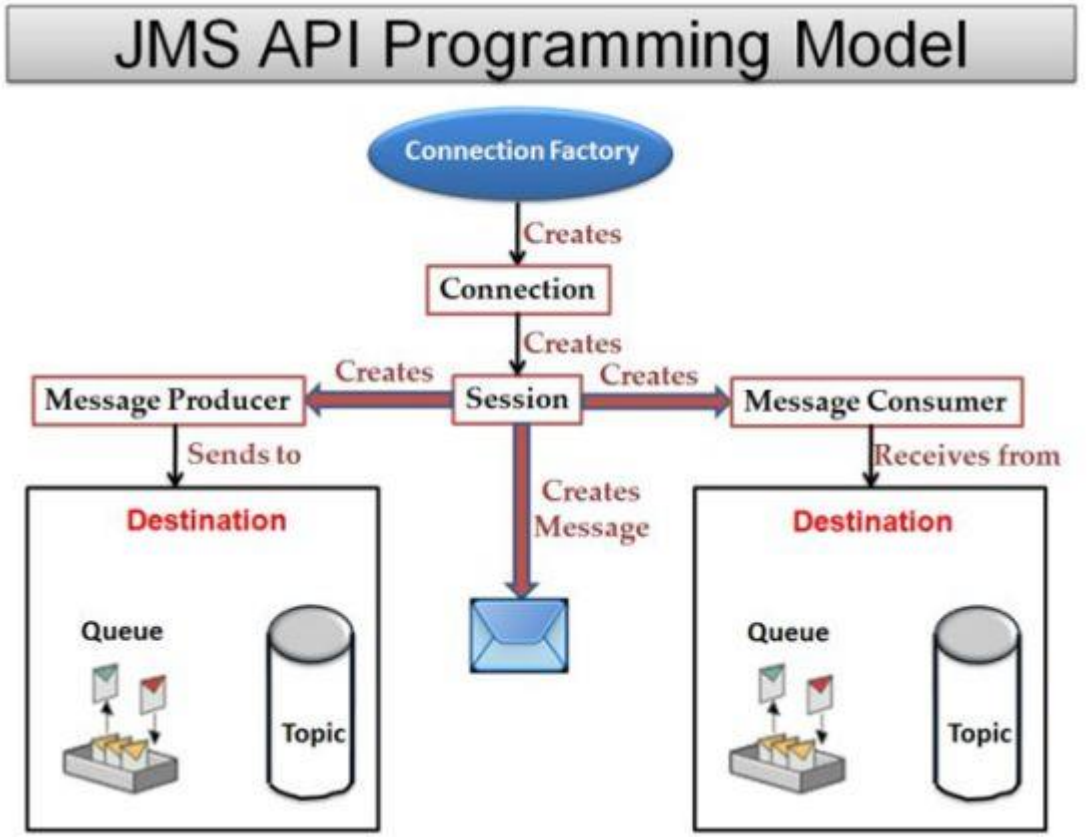
保证每条数据都能被receiver接收

消息队列概述——架构组成

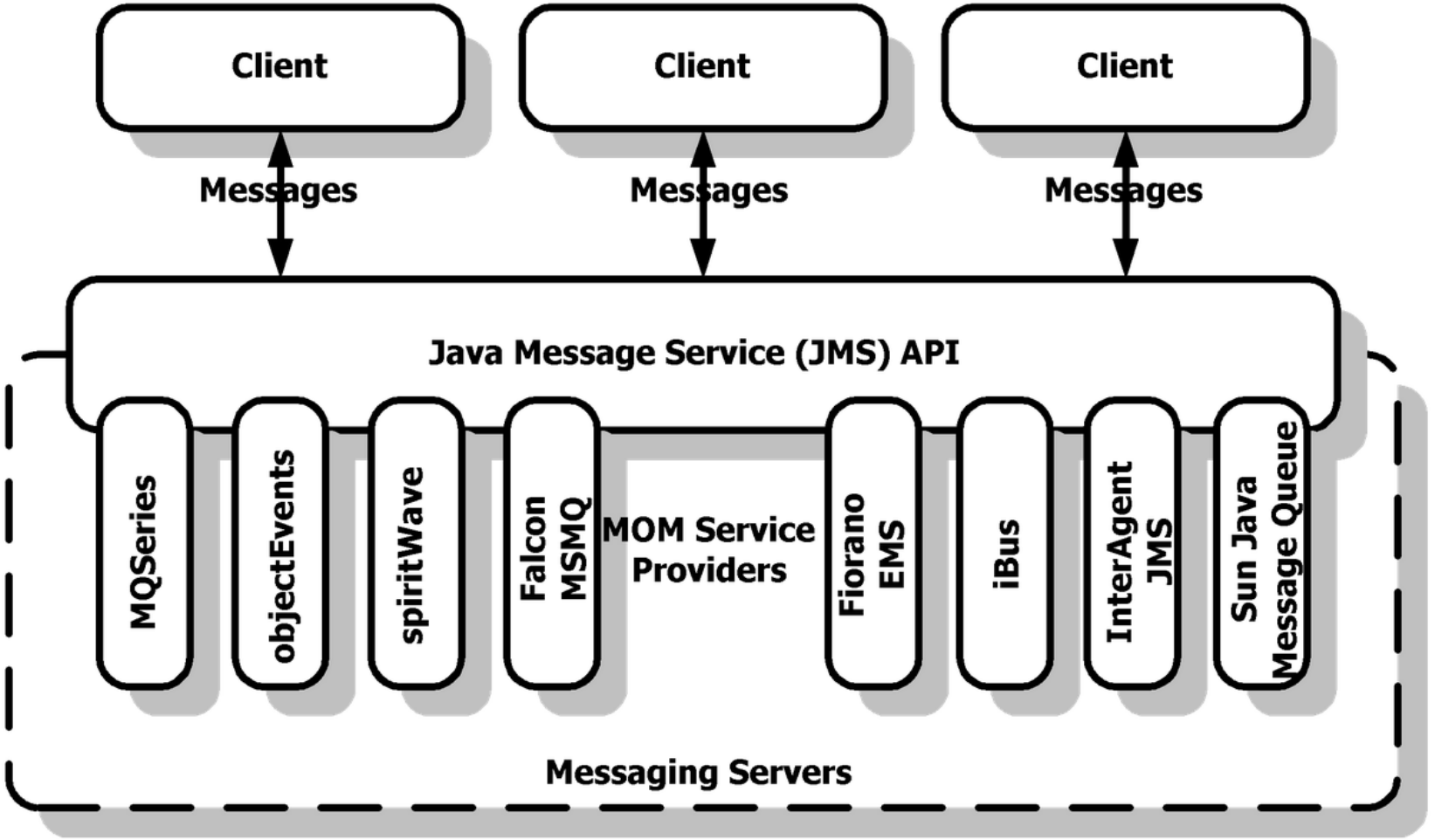

Message Queue
Publishers
Message
Message
Message
Message
Message Box
Queue Owner
Message
Queue : foo
Message


Broker
Message Consumers
Message Producers
ClientA
Msg2
Msg1
Queue
Msg1
ClientC
Msg2
ClientD
ClientB
Msg3
Topic
Msg3
ClientE
Msg3
ClientF

经典的JMS架构 HornetQ



经典的JMS架构

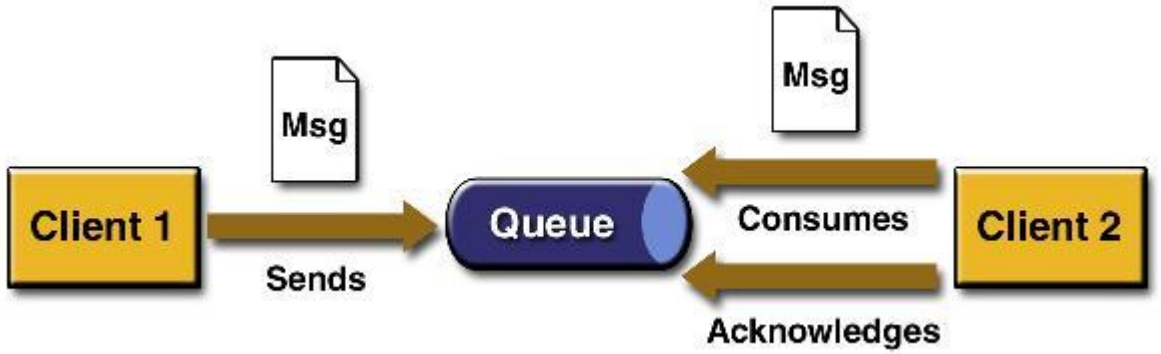

Client
Client
Client
Messages
Messages
Messages
Java Message Service (JMS) API
MQSeries
objectEvents
spiritWave
Falcon MSMQ
MOM Service Providers
Fiorano EMS
iBus
InterAgent JMS
Sun Java Message Queue
Messaging Servers

STOMP & ActiveMQ

STOMP，Streaming Text Orientated Message Protocol

STOMP协议工作于TCP协议之上，使用了下列命令：

* SEND 发送
* SUBSCRIBE 订阅
* UNSUBSCRIBE 退订
* BEGIN 开始
* COMMIT 提交
* ABORT 取消
* ACK 确认
* DISCONNECT 断开




```
[80@vm-icbu-qa-142-156 ~]$ nc 10.12.136.111 61613
CONNECT
accept-version:1.0,1.1,2.0

^@
CONNECTED
session:session-QkGRI8Q4ermqE5vhW-6lU-
heart-beat:0,0
server:RabbitMQ/2.8.7
version:1.1
```

```
SEND
destination:/queue/my_queue

hello stomp!
@
```

MQTT

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

Mnemonic	Enumeration	Description
Reserved	0	Reserved
CONNECT	1	Client request to connect to Server
CONNACK	2	Connect Acknowledgment
PUBLISH	3	Publish message
PUBACK	4	Publish Acknowledgment
PUBREC	5	Publish Received (assured delivery part 1)
PUBREL	6	Publish Release (assured delivery part 2)
PUBCOMP	7	Publish Complete (assured delivery part 3)
SUBSCRIBE	8	Client Subscribe request
SUBACK	9	Subscribe Acknowledgment
UNSUBSCRIBE	10	Client Unsubscribe request
UNSUBACK	11	Unsubscribe Acknowledgment
PINGREQ	12	PING Request
PINGRESP	13	PING Response
DISCONNECT	14	Client is Disconnecting
Reserved	15	Reserved

MQTT有可能成为物联网的重要协议

CONNECT

TCP连接建立完毕后，Client向Server发出一个Request。
如果一段时间内接收不到Server的Response，则关闭socket，重新建立一个session连接。
如果一个ClientID已经与服务器连接，则持有同样ClientID的旧有连接必须由服务器关闭后，新建立才能建立。

CONNACK

Server发出Response响应。
0x00 Connection Accepted
0x01 Connection Refused: unacceptable protocol version
0x02 Connection Refused: identifier rejected
0x03 Connection Refused: server unavailable
0x04 Connection Refused: bad user name or password
0x05 Connection Refused: not authorized

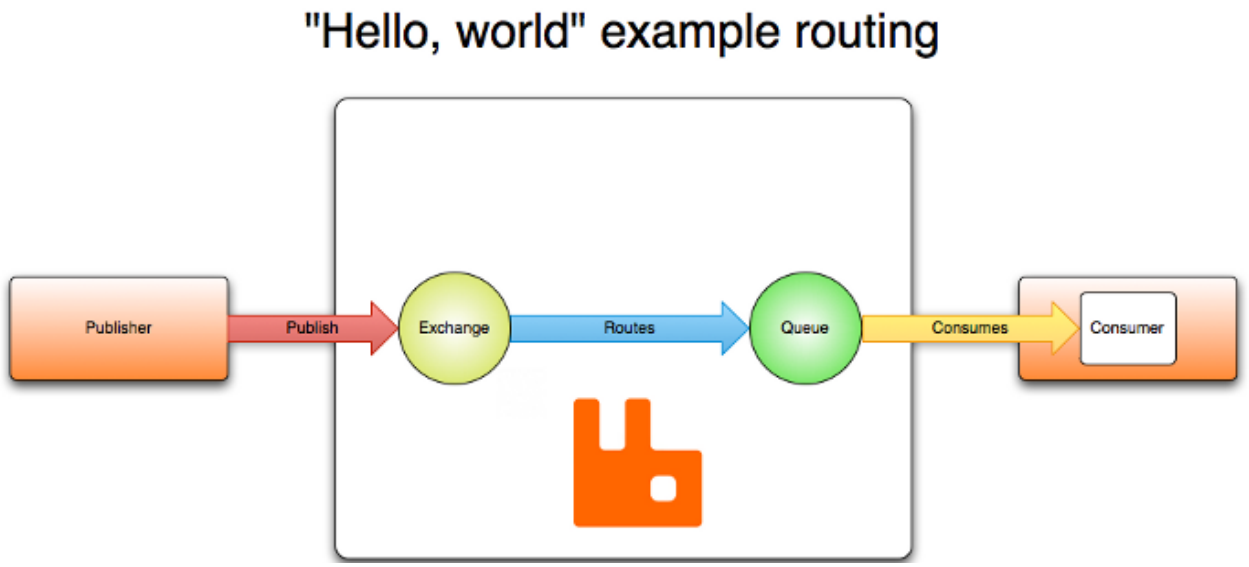
PUBLISH 发布消息

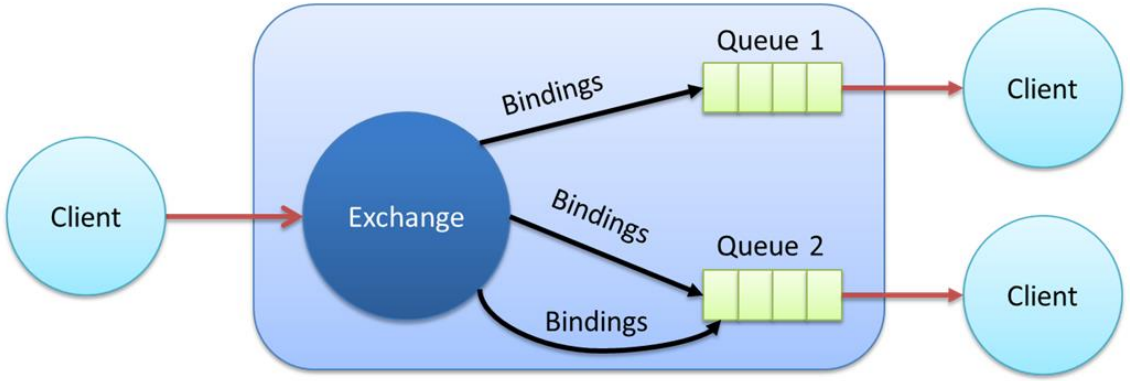
Client/Servier均可以进行PUBLISH。
publish message 应该包含一个TopicName(Subject/Channel)，即订阅关键词。

Apollo: 下一代ActiveMQ

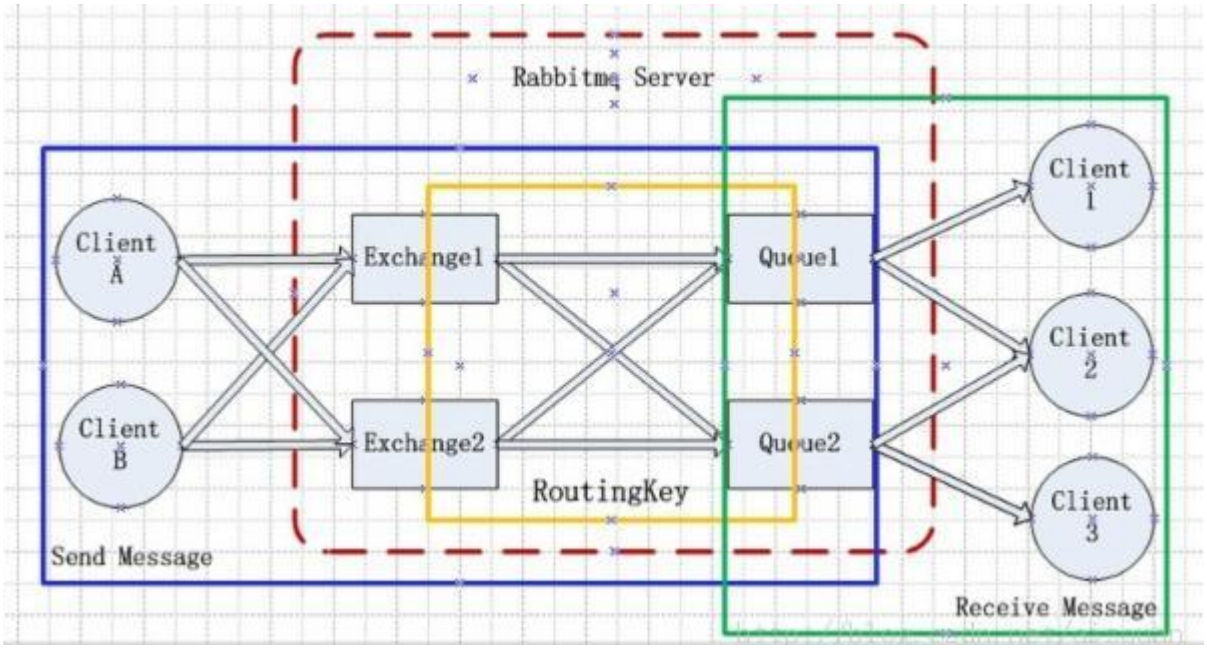
STOMP Protocol Support
AMQP Protocol Support
MQTT Protocol Support
OpenWire Protocol Support
Topics and Queues
Queue Browsers
Durable Subscriptions on Topics
Mirrored Queues
Reliable Messaging
Message Expiration
Message Swapping
Message Selectors
Message Groups
JAAS Authentication
ACL based Authorization
SSL/TLS Support and Certificate based Authentication
REST Management API

AMQP



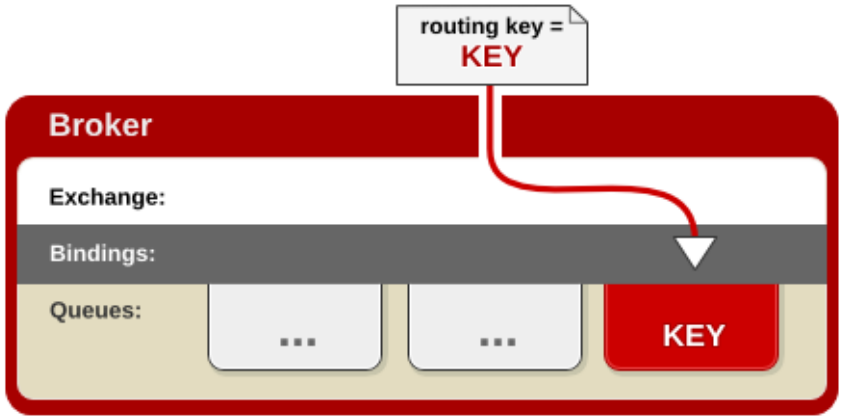


RabitMQ

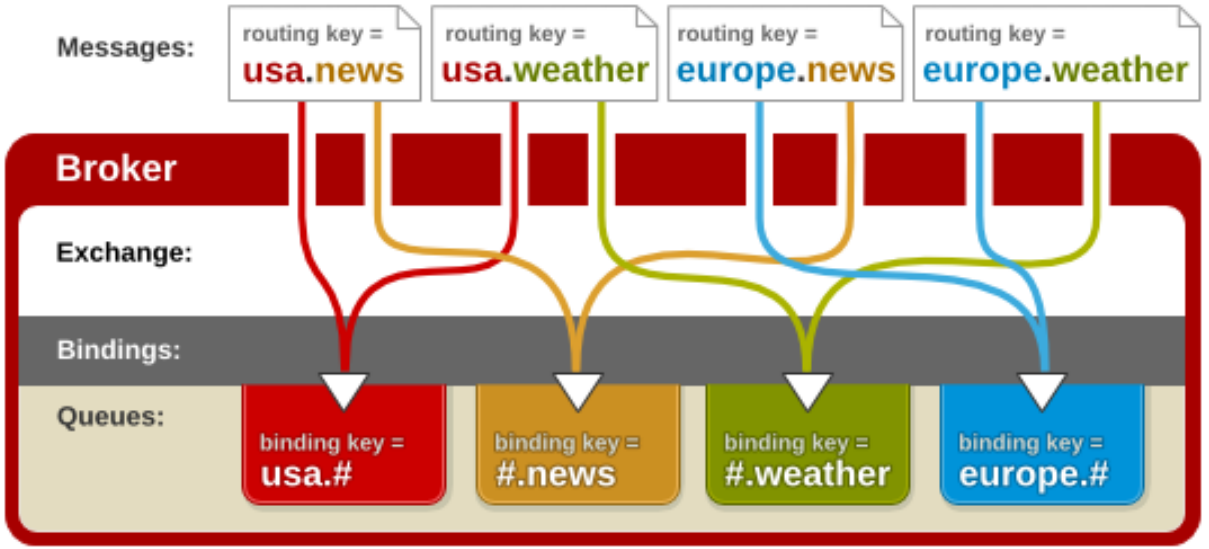


AMQP

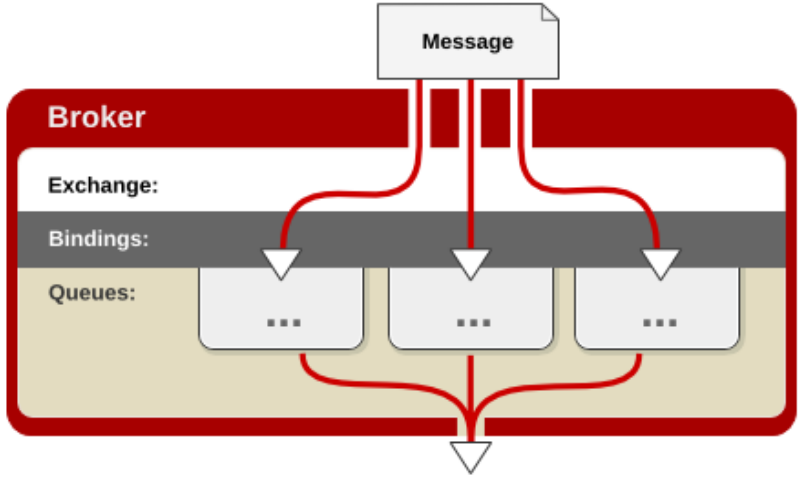
Direct Exchange



Topic Exchange

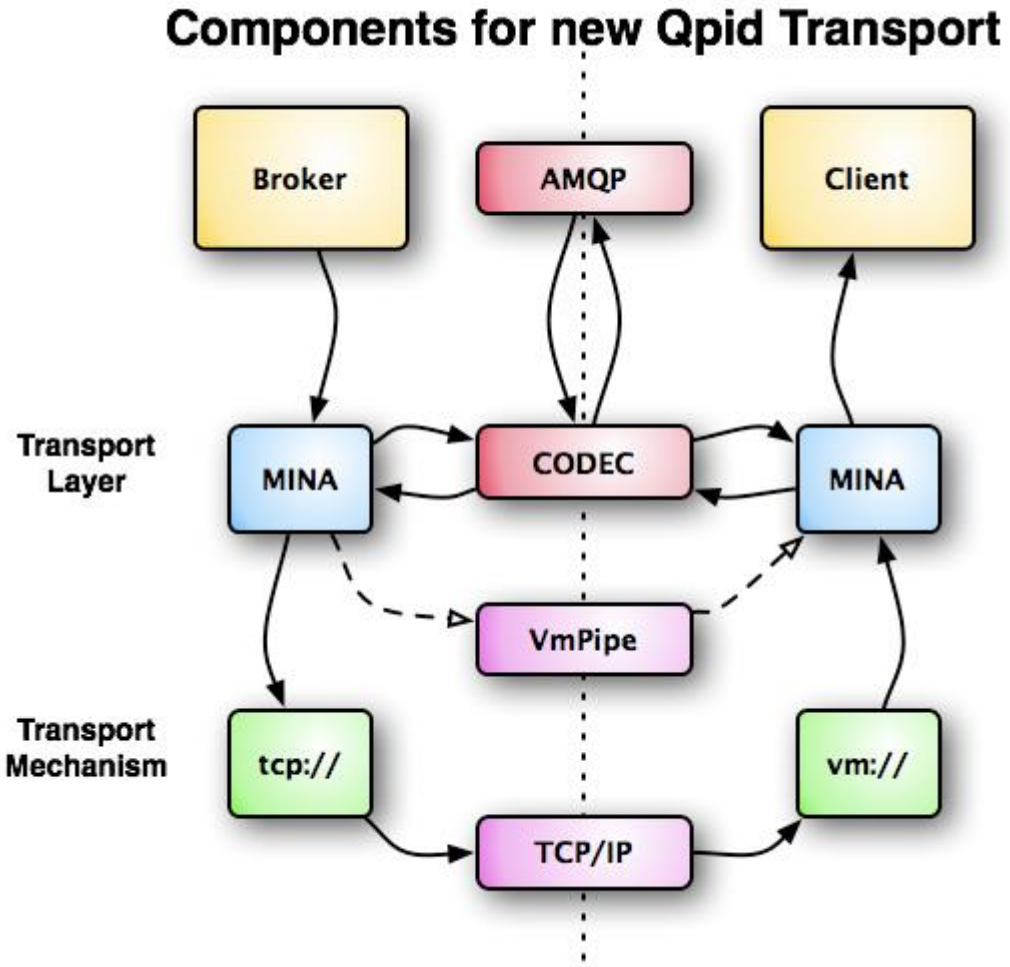


Fanout Exchange



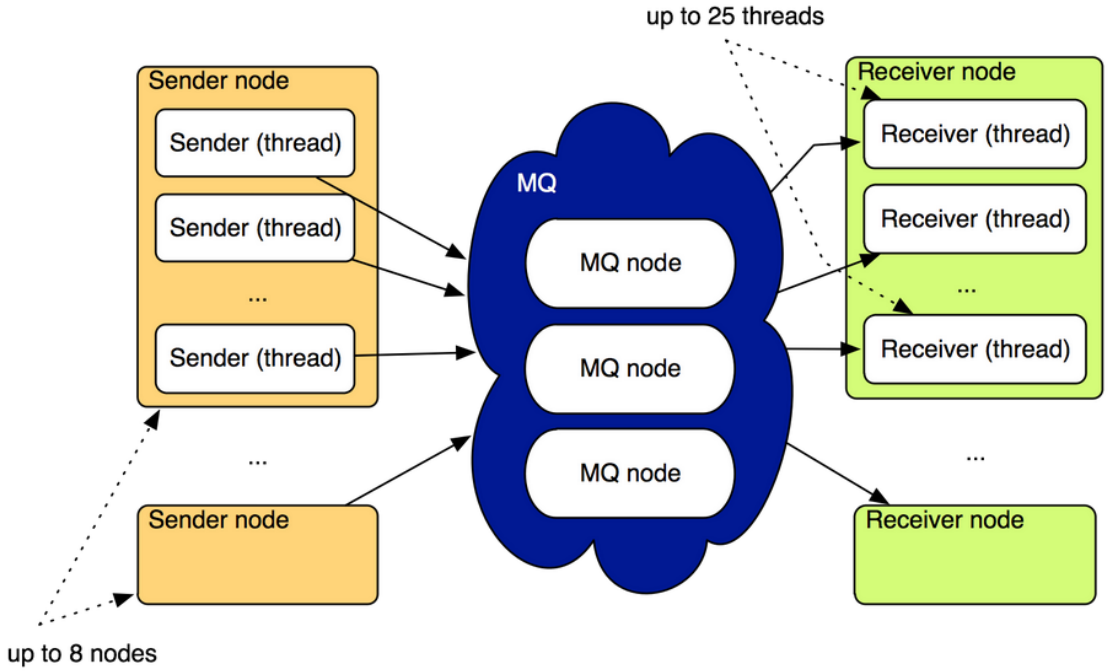
Apache Qpid

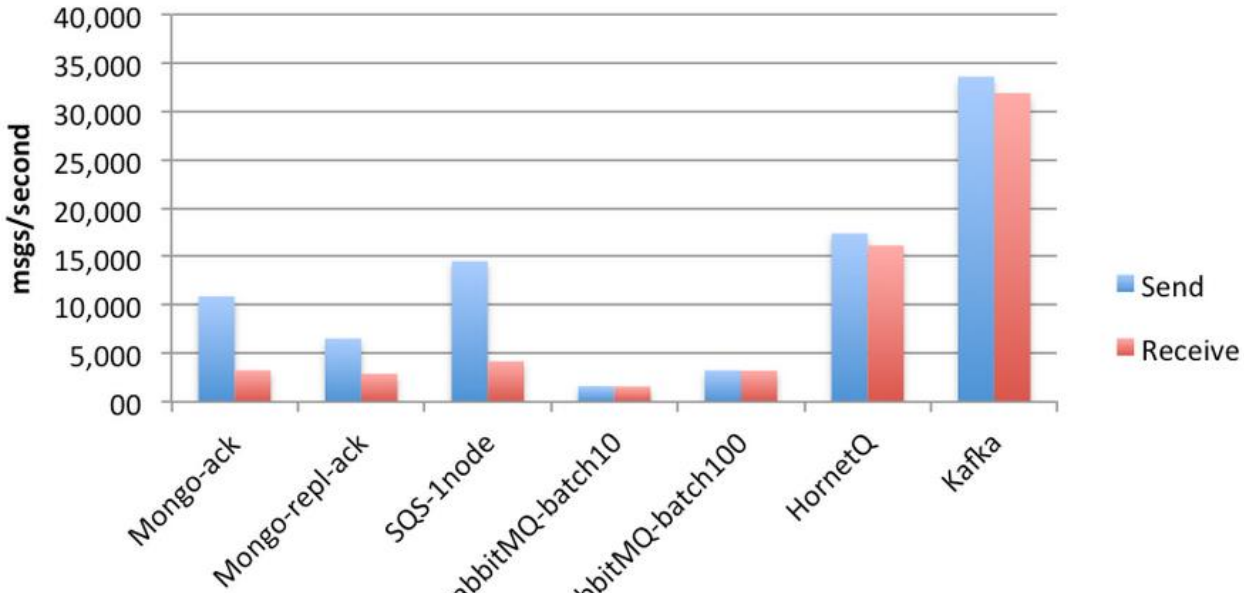
目标是100% Java实现AMQP协议

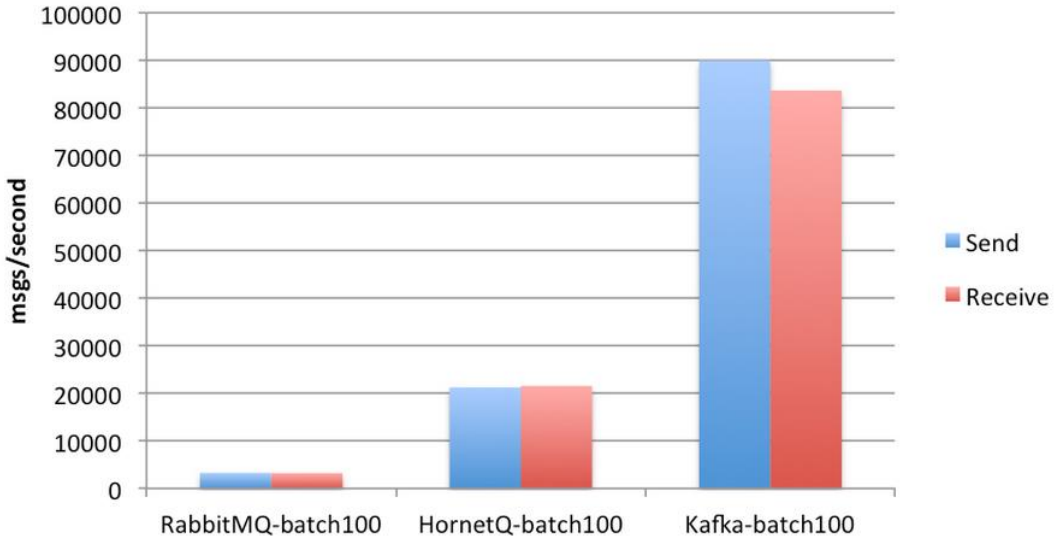


“下一代”消息队列——kafka

先看看一个性能测试对比



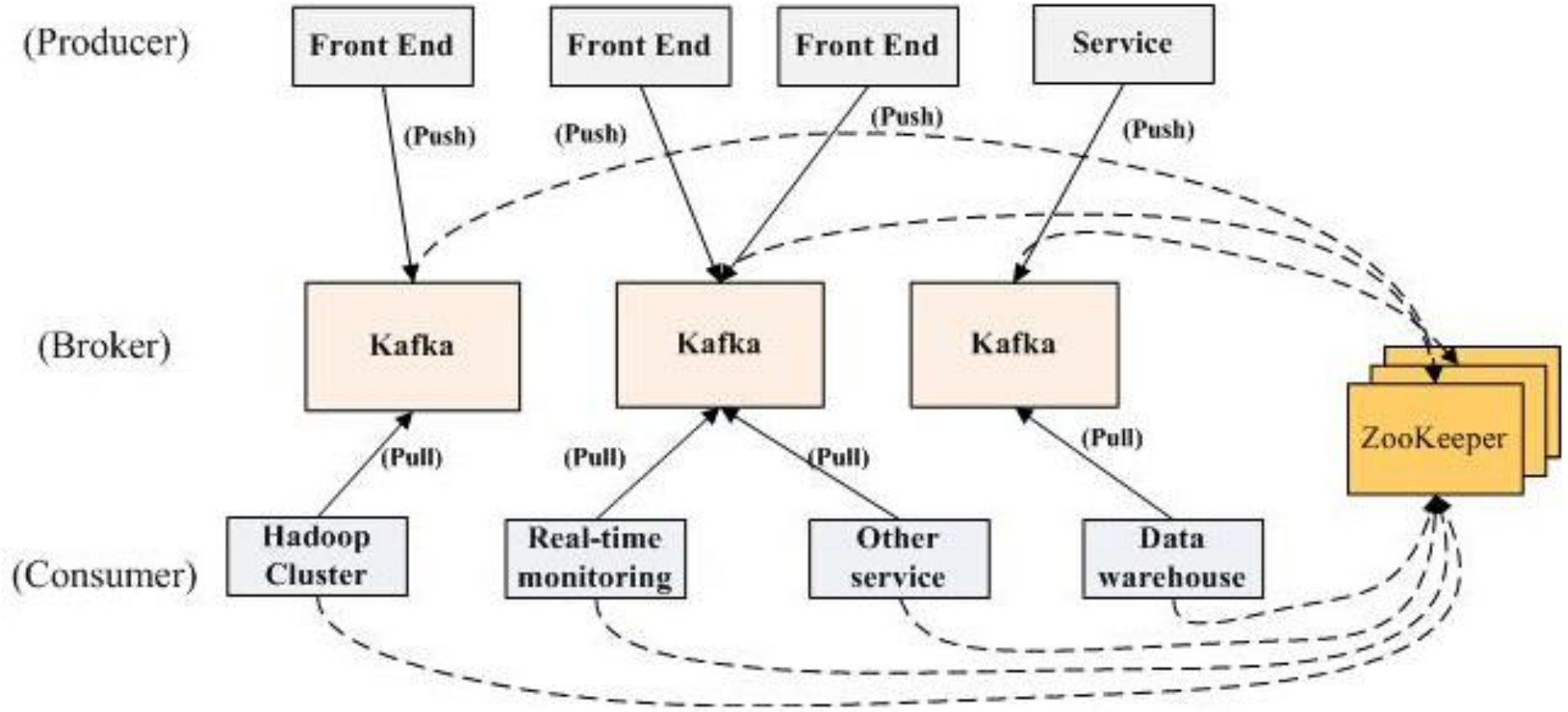




HornetQ has great performance with a very rich messaging interface and routing options

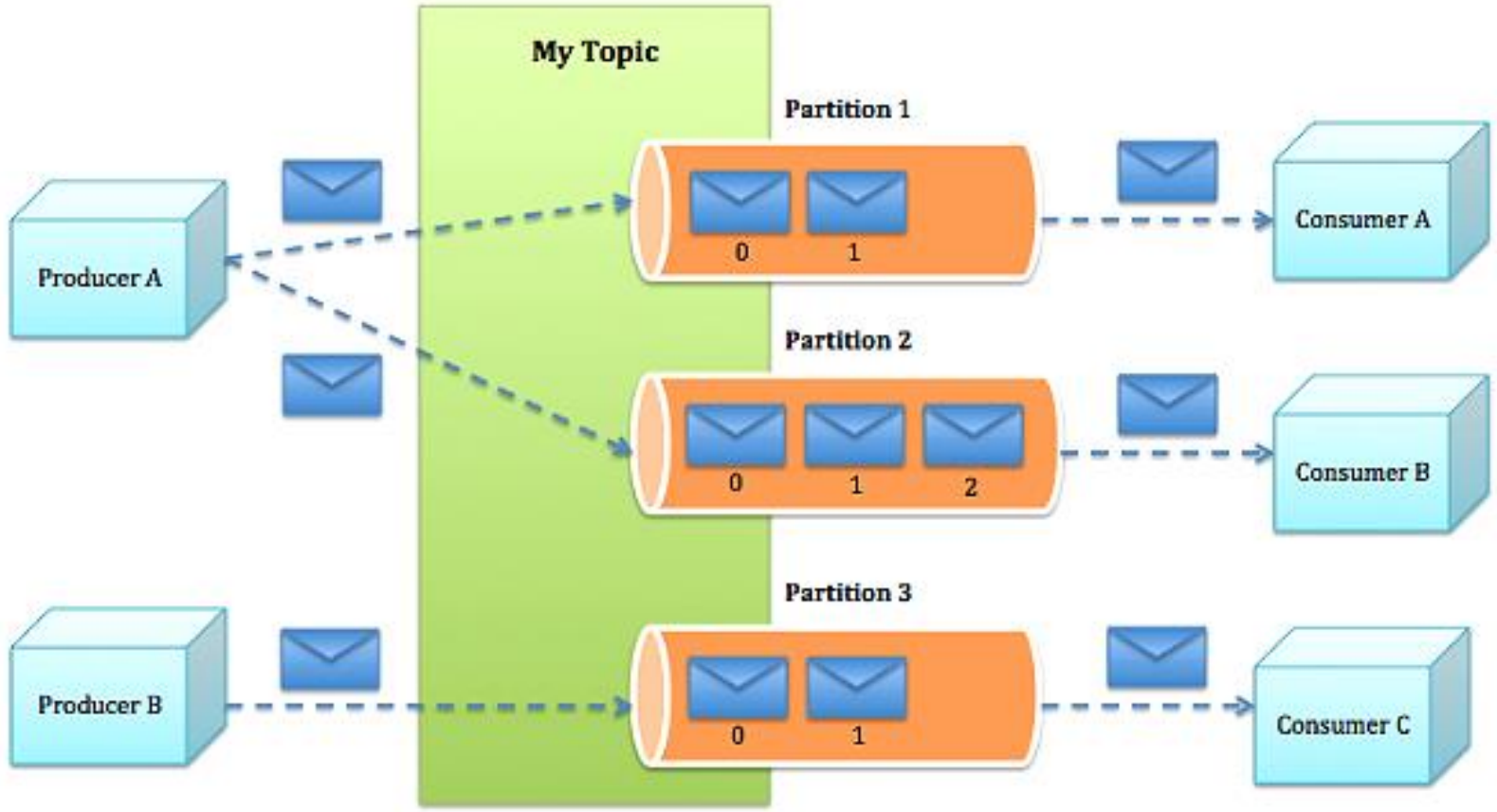
Kafka offers the best performance and scalability

“下一代”消息队列——kafka



“下一代”消息队列——kafka

消息分区



容错机制

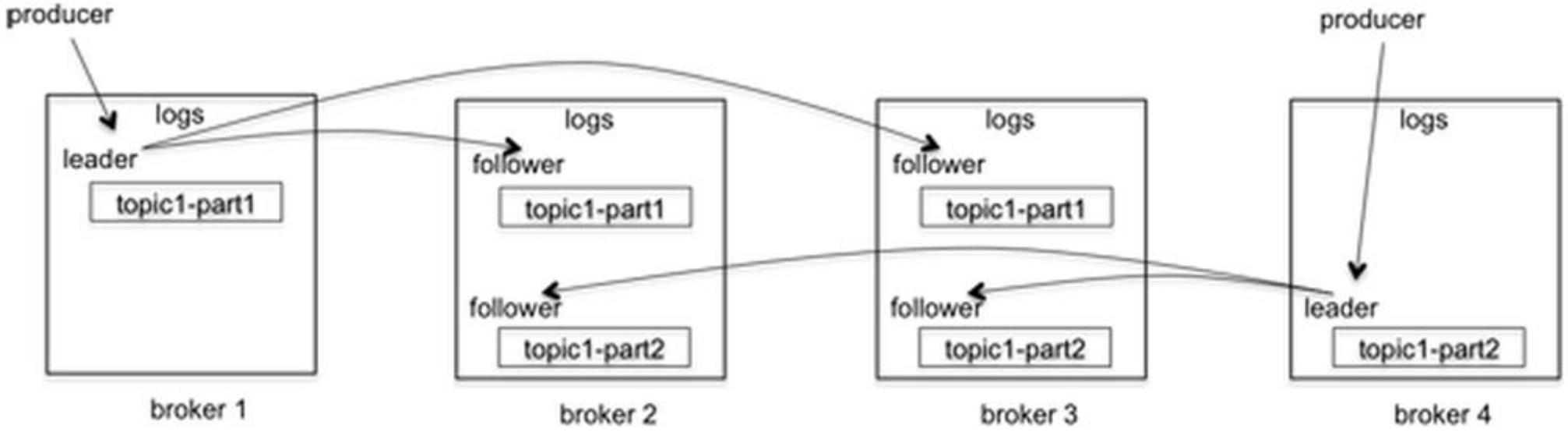
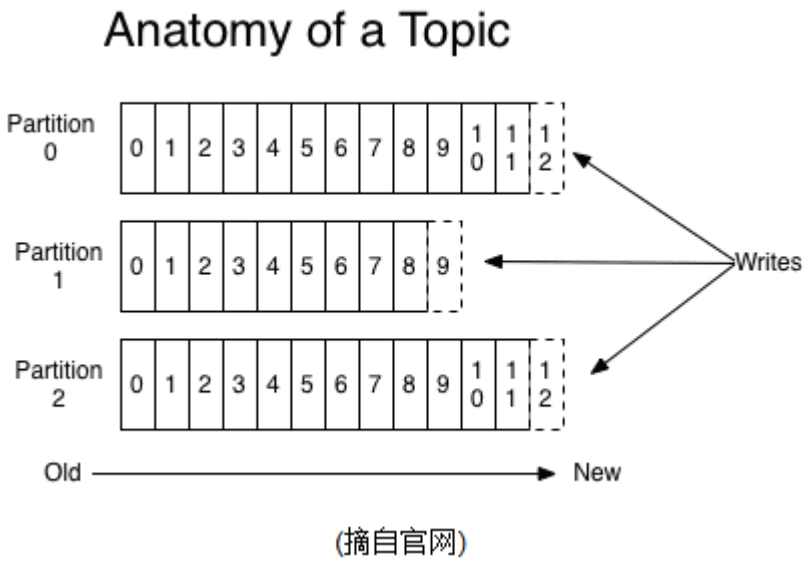


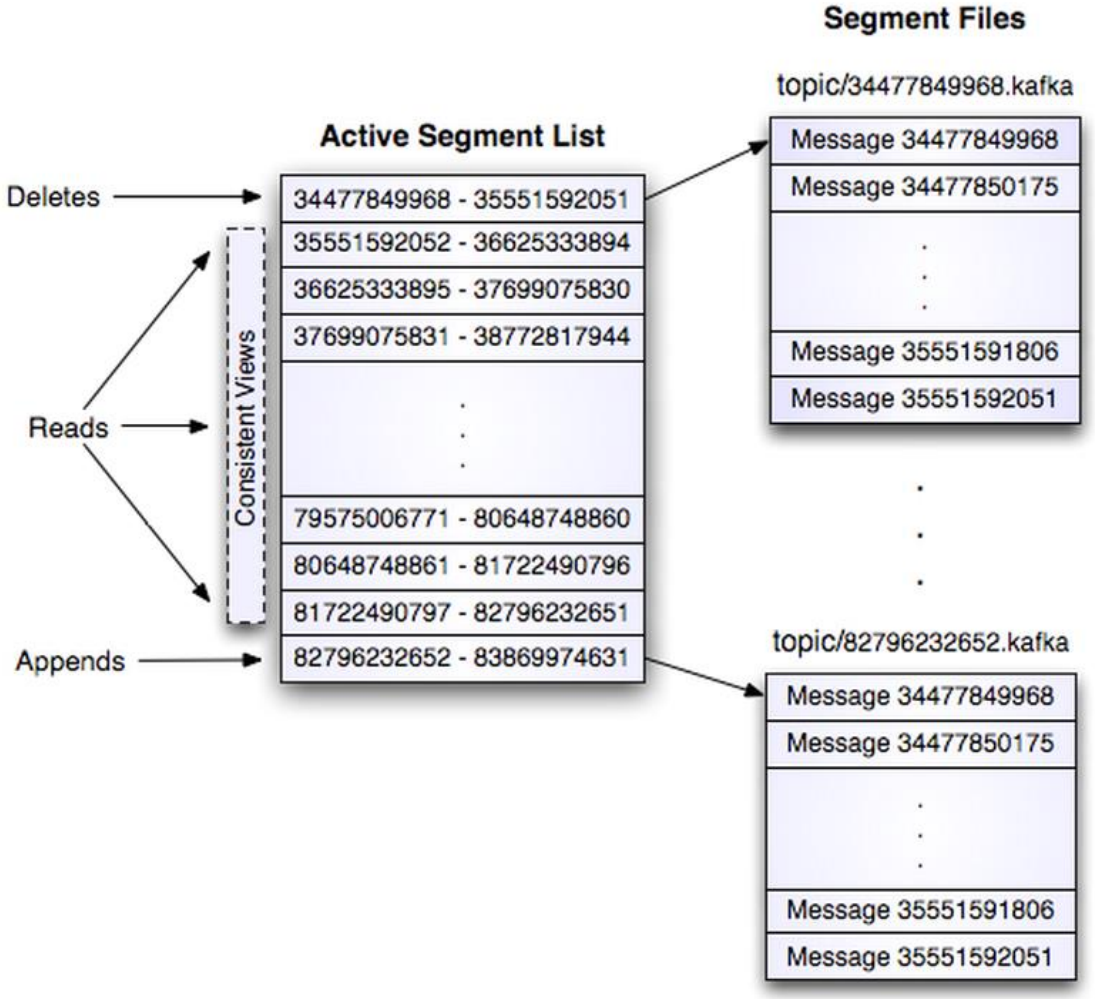
Figure 1. A Kafka cluster with 4 brokers, 1 topic and 2 partitions, each with 3 replicas

“下一代”消息队列——kafka

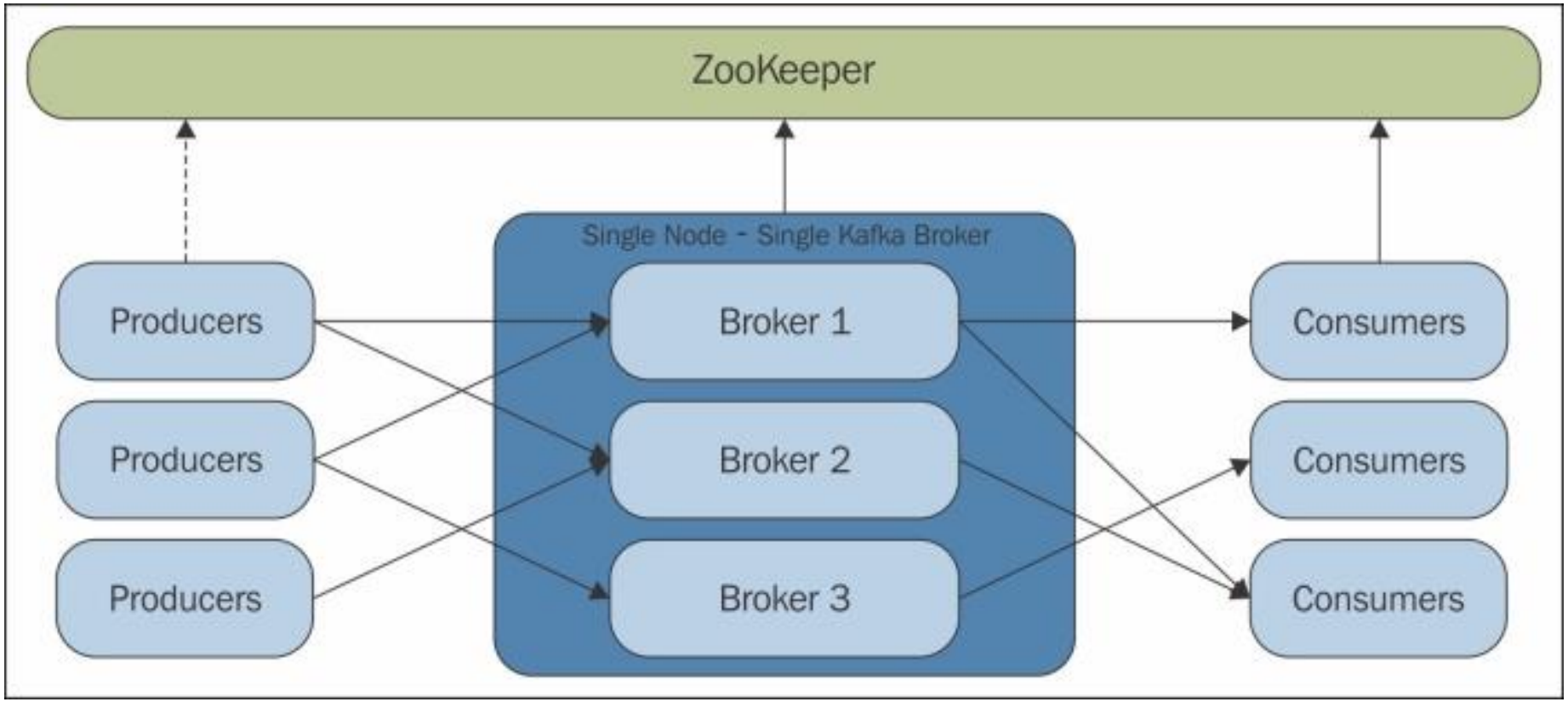
消息存储机制分析



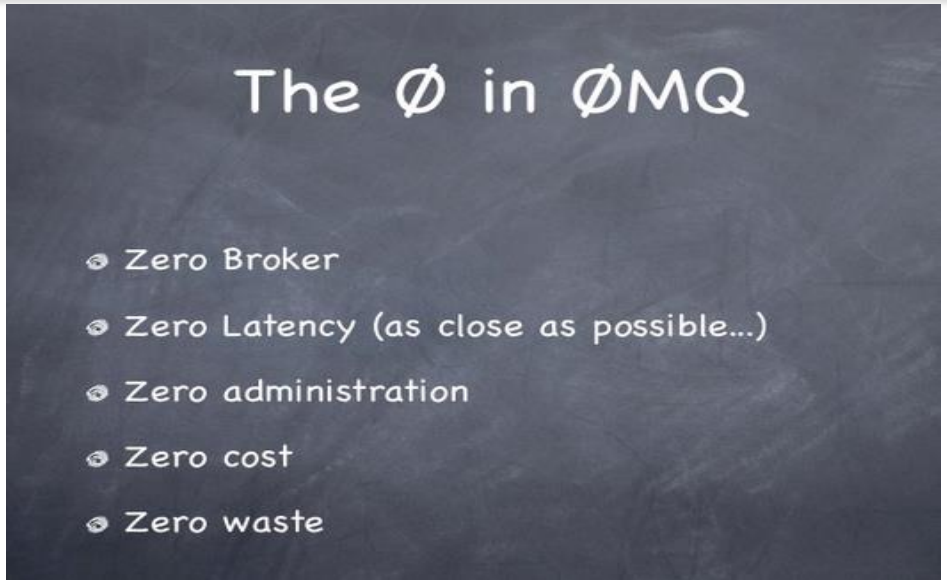
(摘自官网)

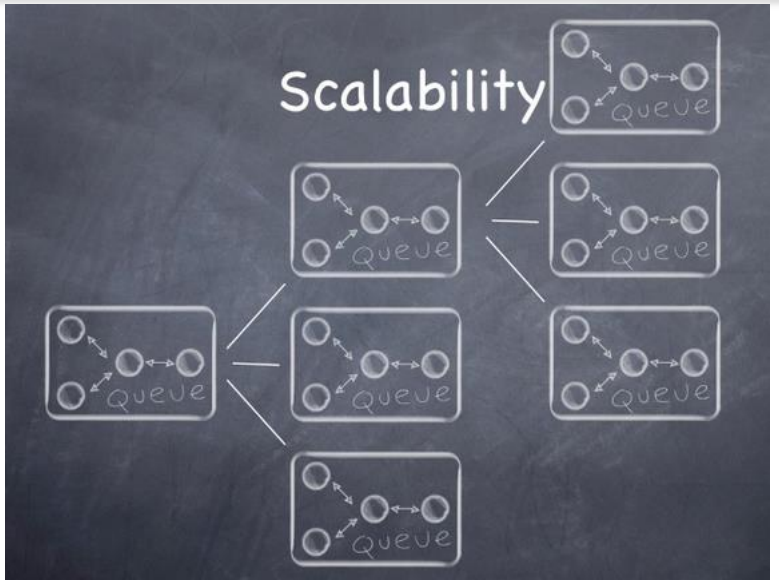


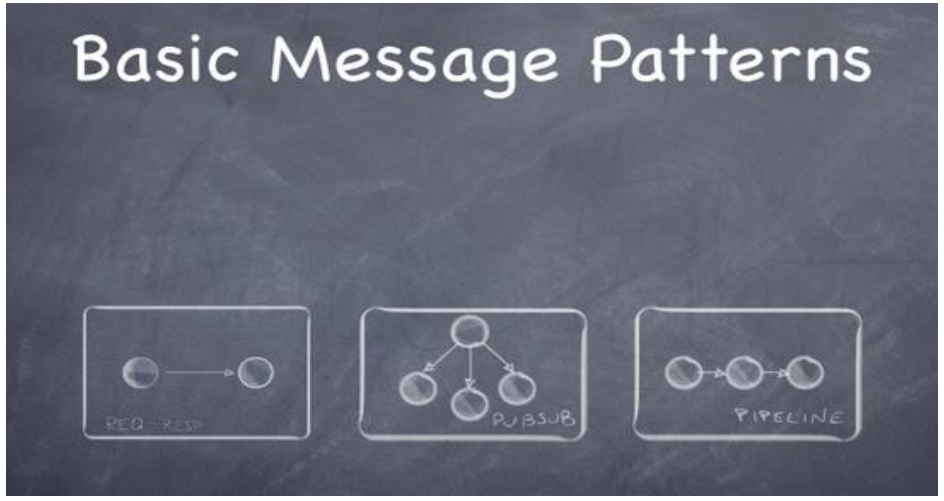
“下一代”消息队列——kafka

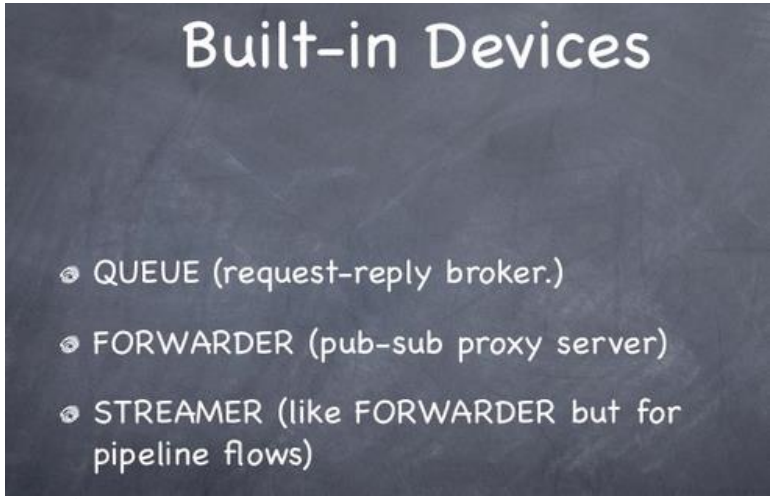


独行侠般的 ØMQ

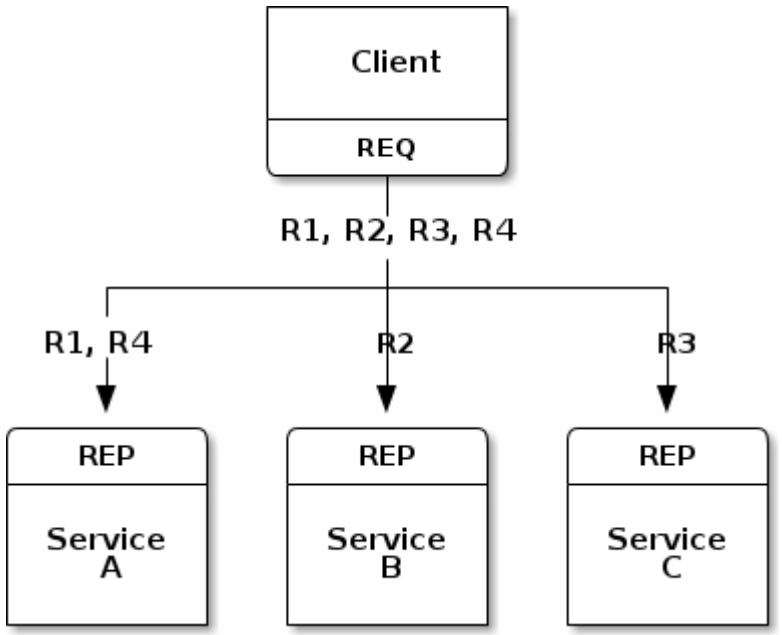

The Ø in ØMQ
Zero Broker
Zero Latency (as close as possible...)
Zero administration
Zero cost
Zero waste

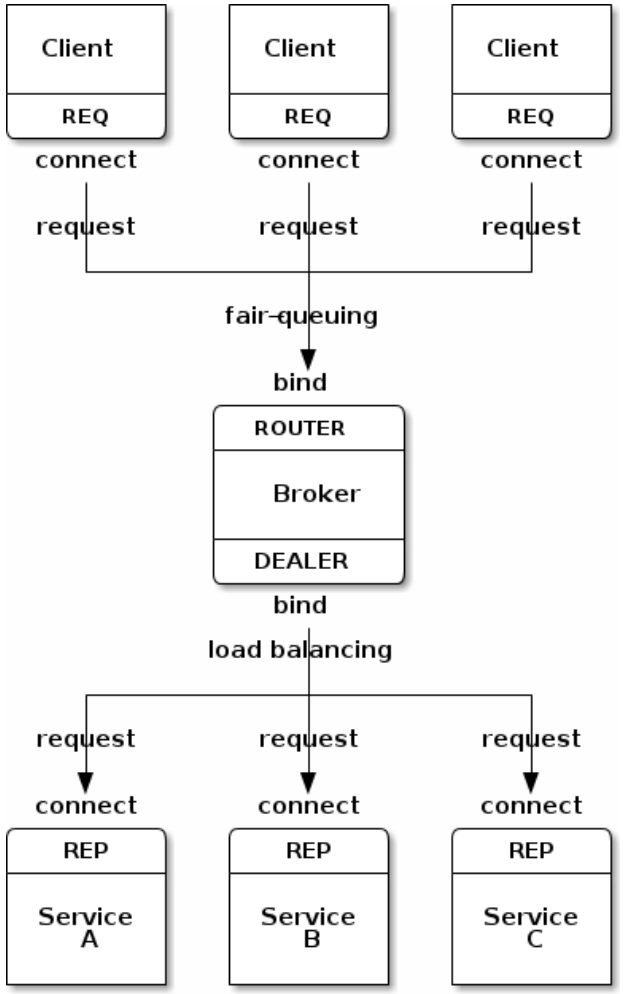

Scalability
QUEUE


Basic Message Patterns
REQ - REP
PUBSUB
PIPELINE


Built-in Devices
QUEUE (request-reply broker.)
FORWARDER (pub-sub proxy server)
STREAMER (like FORWARDER but for pipeline flows)

独行侠般的 ØMQ


Client
REQ
R1, R2, R3, R4
R1, R4
R2
R3
REP
Service A
REP
Service B
REP
Service C


Client
REQ
connect
request
Client
REQ
connect
request
Client
REQ
connect
request
fair-queuing
bind
ROUTER
Broker
DEALER
bind
load balancing
request
connect
REP
Service A
request
connect
REP
Service B
request
connect
REP
Service C

nanomsg

ØMQ

DATAGURU 炼数成金

0.5-beta, released on November 14th, 2014.

1,355 commits | 6 branches | 5 releases | 67 contributors

nanomsg library is MIT-licensed. What it means is that, unlike with ZeroMQ, you can modify the source code and re-release it under a different license, as a proprietary product

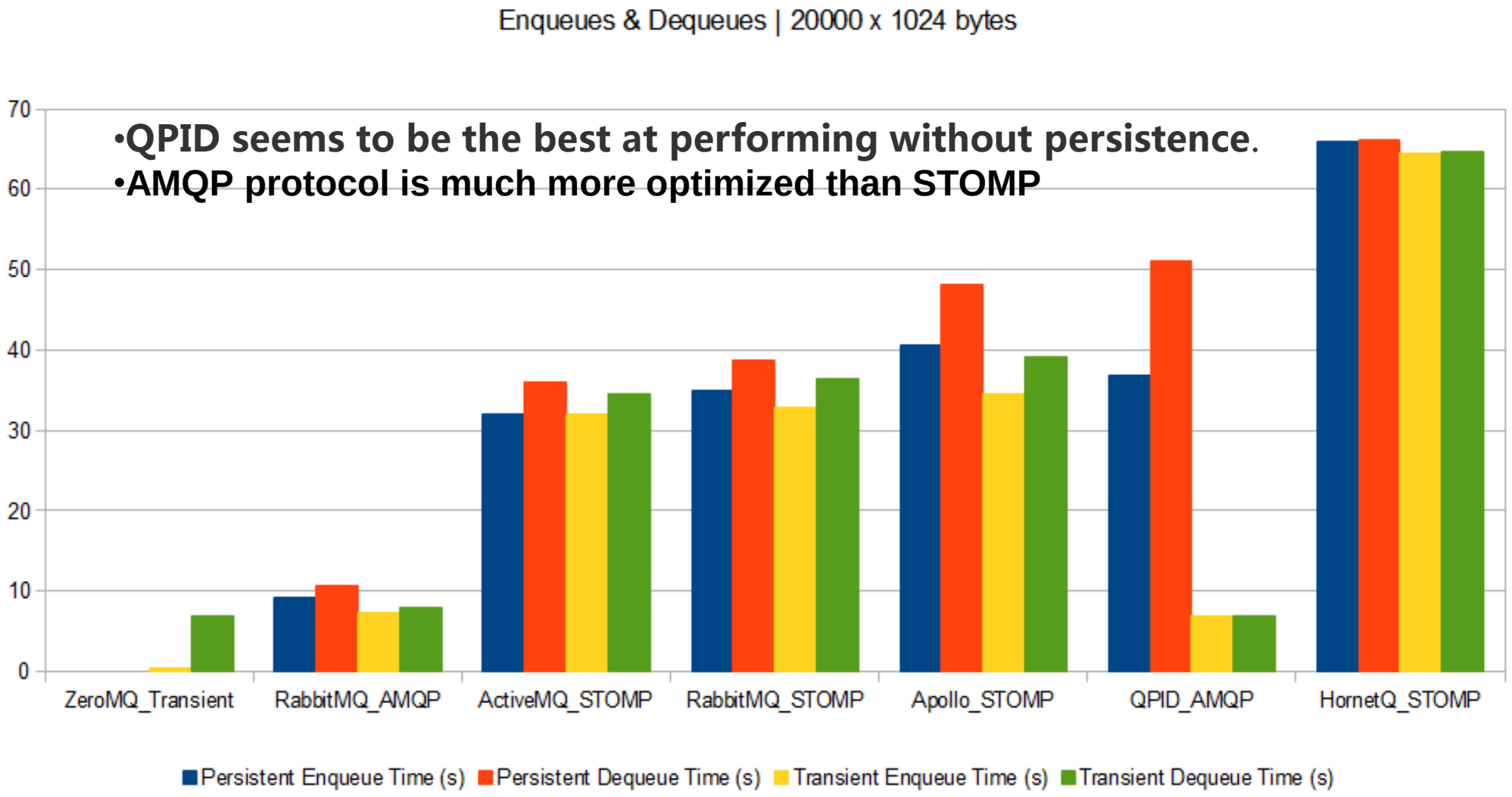
ZeroMQ API, while modeled on BSD socket API, doesn't match the API fully. nanomsg aims for full POSIX compliance.

The library is implemented in C instead of C++.

Threading Model Changed

RDMA **Zero-Copy**

性能大比拼


Enqueues & Dequeues | 20000 x 1024 bytes
•QPID seems to be the best at performing without persistence.
•AMQP protocol is much more optimized than STOMP
70
60
50
40
30
20
10
0
ZeroMQ_Transient
RabbitMQ_AMQP
ActiveMQ_STOMP
RabbitMQ_STOMP
Apollo_STOMP
QPID_AMQP
HornetQ_STOMP
Persistent Enqueue Time (s)
Persistent Dequeue Time (s)
Transient Enqueue Time (s)
Transient Dequeue Time (s)

Thanks

FAQ时间