

C/C++Linux服务器开发 高级架构师课程

三年课程沉淀

五次精益升级

十年行业积累

百个实战项目

十万内容受众

专注于IT职业提升，为工程师提供优质完善的成长体系。

缩短工程师的学习时间，

增强工程师的学习效果，提升工程师的资薪待遇。

为工程师的技术提升穿针引线，为工程师的职业成长搭桥铺路。

办学宗旨：**一切只为渴望更优秀的你。**



课题：消息队列和ZeroMQ

- 消息队列
- 使用消息队列的场景
- 消息队列基本概念和原理
- 可供选择的消息队列产品
- ZeroMQ应用



1 消息队列

消息+队列（MessageQueue，简称MQ）。

本质是就是个队列，FIFO先入先出，只不过队列中存放的内容是message，从而叫消息队列。

主要用途：不同服务server、进程process、线程thread之间通信。



为什么要引入消息队列？



2 使用消息队列的场景

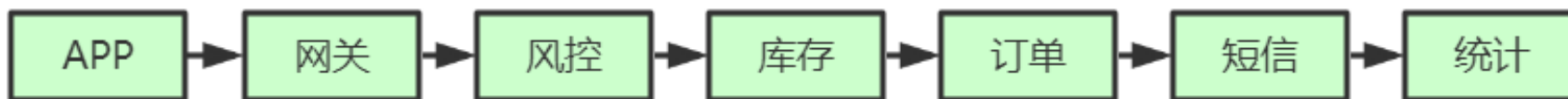
1. 异步处理
2. 流量控制
3. 服务解耦
4. 发布订阅
5. 高并发缓冲



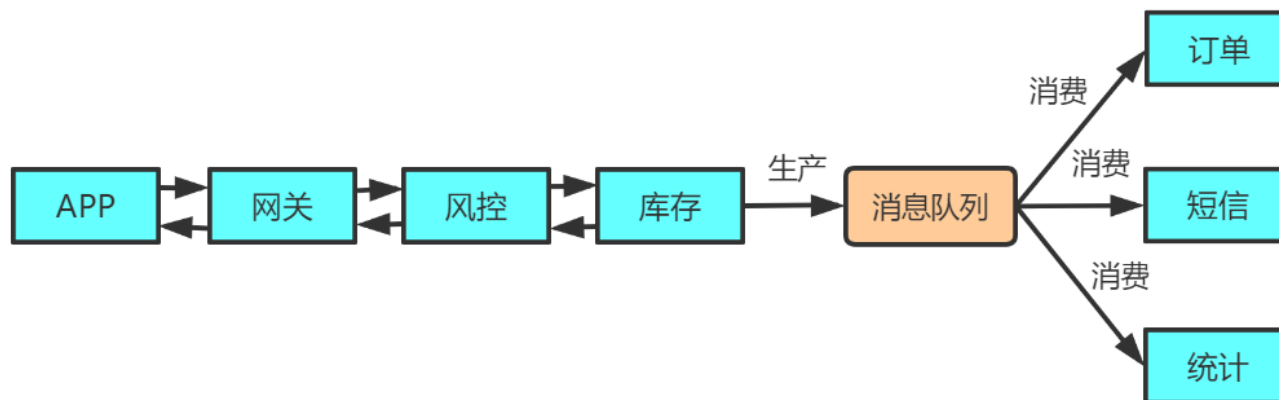
2.1 消息队列-异步处理

短信通知、终端状态推送、App推送、用户注册等

同步处理:



异步处理:



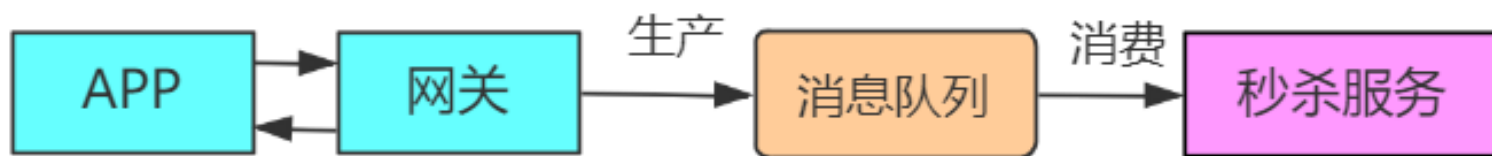
更快速返回结果;
减少等待, 实现并发处理, 提升系统总体性能。



2.2 消息队列-流量控制(削峰)

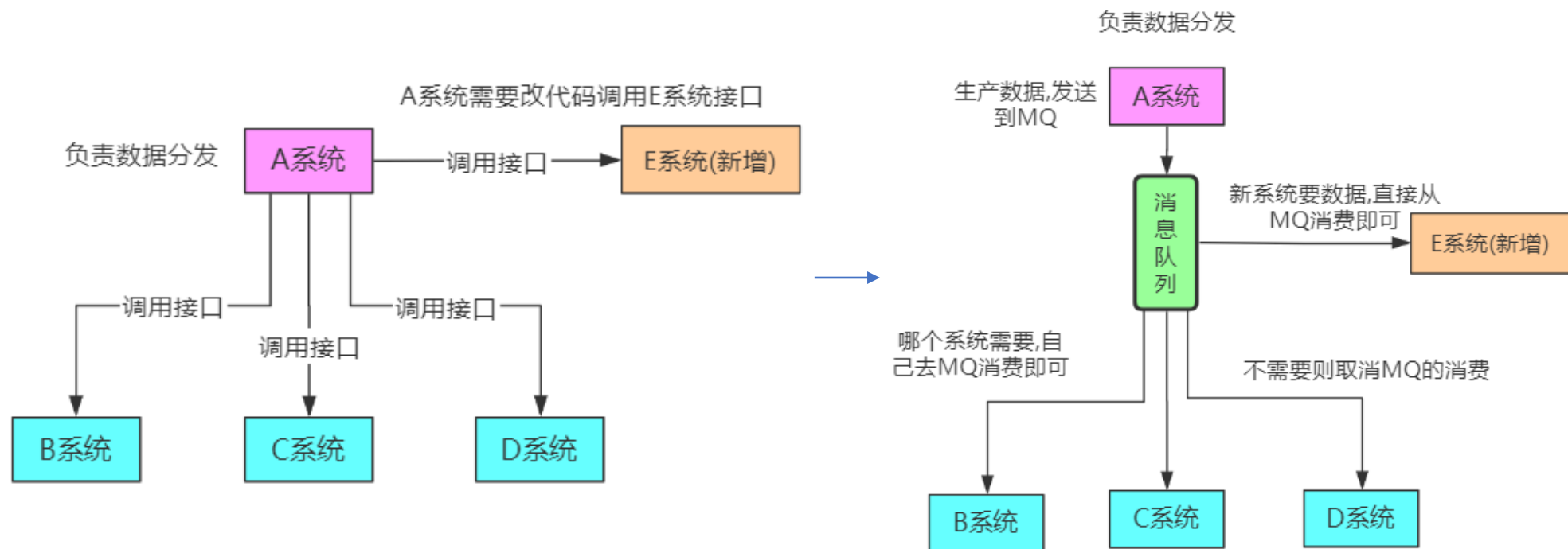
秒杀场景下的下单状态

使用消息队列隔离网关和后端服务，以达到流量控制和保护后端服务的目的。



2.3 消息队列-服务解耦

使用消息队列实现系统的解耦



2.4 消息队列-发布订阅

- 比如游戏里面跨服:
 - 广播今天整体还剩多少把屠龙刀可以暴
 - 广播用户暴的屠龙刀的消息



2.5 消息队列-高并发缓冲

- kafka 日志服务、监控上报



3 消息队列-基本概念和原理1

1. Broker

Broker的概念来自与Apache ActiveMQ，通俗的讲就是MQ的服务器。

2. 消息的生产者、消费者

消息生产者Producer：发送消息到消息队列。

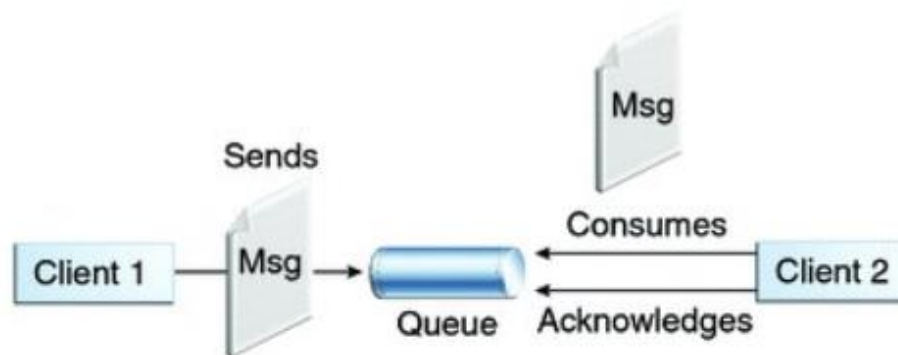
消息消费者Consumer：从消息队列接收消息。



3 消息队列-基本概念和原理2

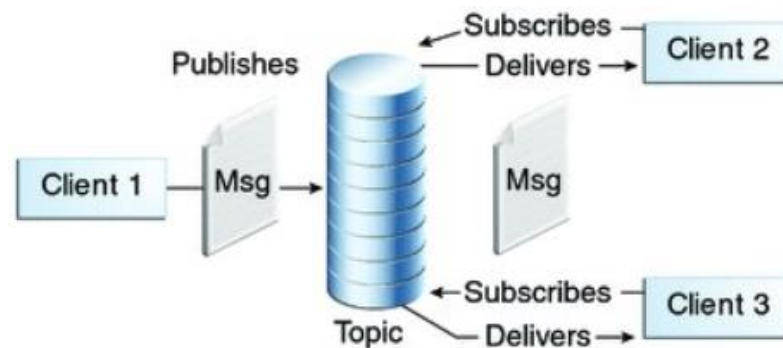
3. 点对点消息队列模型

消息生产者向一个特定的队列发送消息，消息消费者从该队列中接收消息；
一条消息只有一个消费者能收到；



4. 发布订阅消息模型-Topic

发布订阅消息模型中，支持向一个特定的主题Topic发布消息，0个或多个订阅者接收来自这个消息主题的消息。在这种模型下，发布者和订阅者彼此不知道对方。实际操作过程中，发布订阅消息模型中，支持向一个特定的主题Topic发布消息，0个或多个订阅者接收来自这个消息主题的消息。在这种模型下，发布者和订阅者彼此不知道对方。



3 消息队列-基本概念和原理3

5. 消息的顺序性保证

基于Queue消息模型，利用FIFO先进先出的特性，可以保证消息的顺序性。

6. 消息的ACK确认机制

即消息的Acknowledge确认机制，为了保证消息不丢失，消息队列提供了消息Acknowledge机制，即ACK机制，当Consumer确认消息已经被消费处理，发送一个ACK给消息队列，此时消息队列便可以删除这个消息了。如果Consumer宕机/关闭，没有发送ACK，消息队列将认为这个消息没有被处理，会将这个消息重新发送给其他的Consumer重新消费处理。

7. 消息的持久化

消息的持久化，对于一些关键的核心业务来说是非常重要的，启用消息持久化后，消息队列宕机重启后，消息可以从持久化存储恢复，消息不丢失，可以继续消费处理。



3 消息队列-基本概念和原理3

8. 消息的同步和异步收发

同步：消息的收发支持同步收发的方式 **一红一答**。

同时还有另一种同步方式：同步收发场景下，消息生产者和消费者双向应答模式，例如：张三写信送到邮局中转站，然后李四从中转站获得信，然后在写一份回执信，放到中转站，然后张三去取，当然张三写信的时候就得写明回信地址；

消息的接收如果以同步的方式(Pull)进行接收，如果队列中为空，此时接收将处于同步阻塞状态，会一直等待，直到消息的到达。

异步：消息的收发同样支持异步方式：**异步发送消息，不需要等待消息队列的接收确认**；异步接收消息，以Push的方式触发消息消费者接收消息。



4 可供选择的消息队列产品

1. RabbitMQ 2. RocketMQ 3. Kafka 4. ZeroMQ 轻量级

特性	RabbitMQ	RocketMQ	Kafka	ZeroMQ
单机吞吐量	万级，比 RocketMQ、Kafka 低一个数量级	10 万级，支撑高吞吐	10 万级，高吞吐，一般配合大数据类的系统来进行实时数据计算、日志采集等场景	100万级别，最早设计用于股票实时交易系统
topic 数量对吞吐量的影响		topic 可以达到几百/几千的级别，吞吐量会有较小幅度的下降，这是 RocketMQ 的一大优势，在同等机器下，可以支撑大量的 topic	topic 从几十到几百个时候，吞吐量会大幅度下降，在同等机器下，Kafka 尽量保证 topic 数量不要过多，如果要支撑大规模的 topic，需要增加更多的机器资源	
时效性	微秒级，这是 RabbitMQ 的一大特点，延迟最低	ms 级	延迟在 ms 级以内	延迟在微妙级别/毫秒级别
可用性	高，基于主从架构实现高可用	非常高，分布式架构	非常高，分布式，一个数据多个副本，少数机器宕机，不会丢失数据，不会导致不可用	不是一个独立的服务，要嵌套到自己的程序里面去
消息可靠性	基本不丢	经过参数优化配置，可以做到 0 丢失	同 RocketMQ	
功能支持	基于 erlang 开发，并发能力很强，性能极好，延时很低	MQ 功能较为完善，支持分布式部署，扩展性好	功能较为简单，主要支持简单的 MQ 功能，在大数据领域的实时计算以及日志采集被大规模使用	如果你的需求是将消息队列的功能集成到你的系统进程中，可以考虑使用 ZeroMQ。

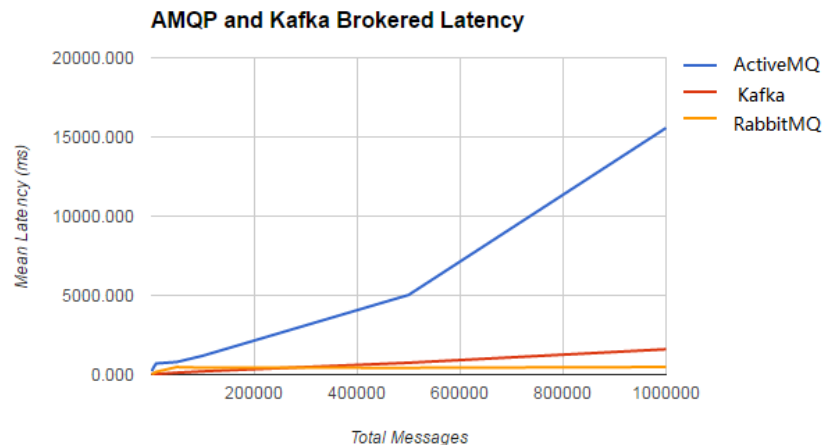
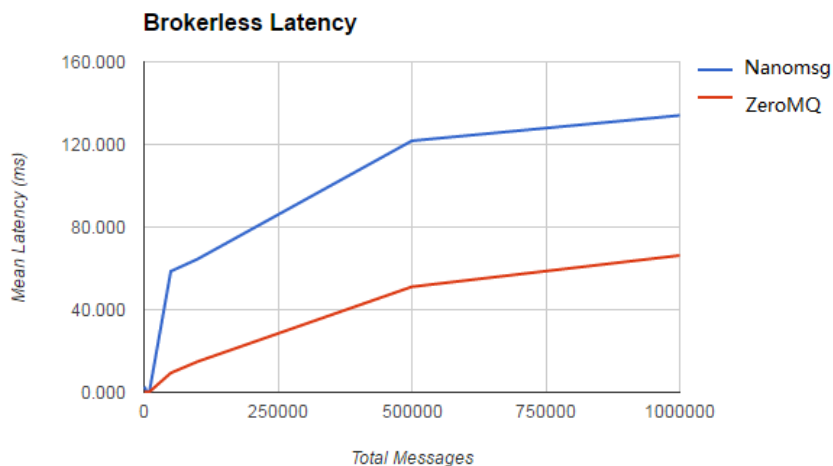
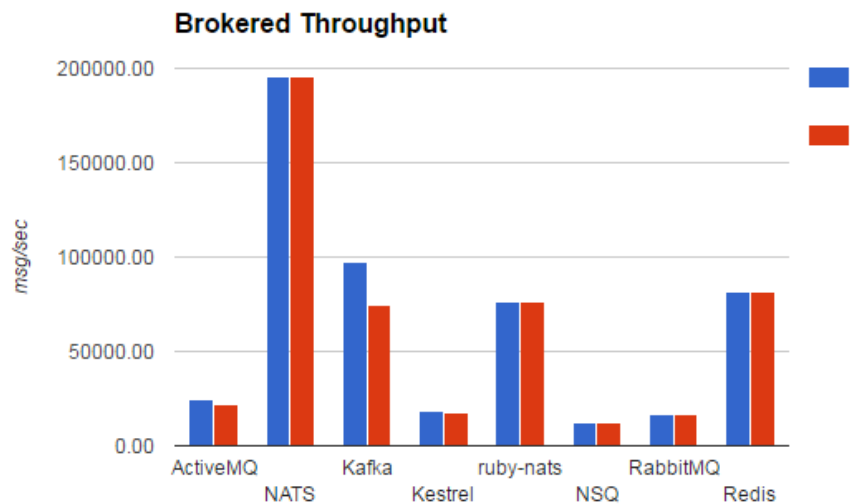
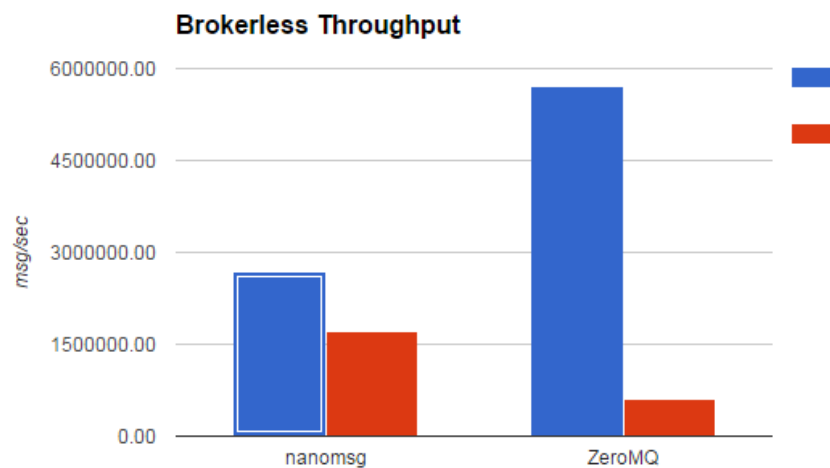


5.1 ZeroMQ架构背后的历史

第一年	第二年	第三年	第四年
最初设想是作为股票交易中的一个极快速的消息通信系统，因此重点放在了高度优化上。项目开始的头一年都花在制定性能基准测试的方法上了，并尝试设计出一个尽可能高效的架构。	开发的重点转变成成为构建分布式应用程序而提供一个通用系统，支持任意模式的消息通信、多种传输机制、对多种编程语言的绑定等等。。	重点主要集中于提高系统的可用性，将学习曲线平坦化。我们已经采用了BSD套接字API，尝试整理单个消息通信模式的语义等等	开源和发布等等.....



5.2 ZeroMQ吞吐量 and 延时性



5.3 ZeroMQ 解决传统网络编程的问题

- 调用的socket接口较多;
- TCP是一对一的连接; - 一对多, reactor模式
- 编程需要关注很多socket细节问题;
- 不支持跨平台编程;
- 需要自行处理分包、组包问题;
- 流式传输时需处理粘包、半包问题;
- 需自行处理网络异常, 比如连接异常中断、重连等;
- 服务端和客户端启动有先后;
- 自行处理IO模型;
- 自行实现消息的缓存 (消息水位) ;
- 自行实现对消息的加密



5.4 ZeroMQ入门和进阶

ZeroMQ编译和测试：《ZMQ编译安装和测试.pdf》

官方API: <http://api.zeromq.org/>

英文指南: <http://zguide.zeromq.org/page:all>

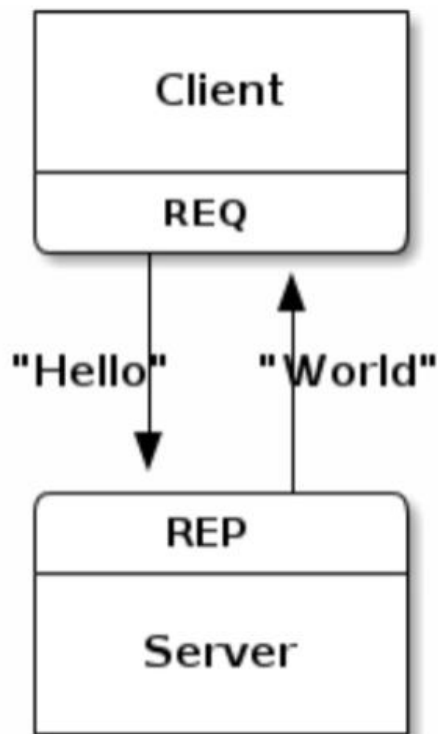
中文指南: <https://github.com/anjuke/zguide-cn>, 文档里面的代码有些是过时的, 需要参考提供的github链接的代码

性能测试: <http://wiki.zeromq.org/results:perf-howto>

中文参考书籍: 《ZeroMQ 云时代极速消息通信库 .pdf》



5.5 ZeroMQ模型1-REQ/REP 请求响应模型



1.REQ/REP模型

hwserver.c

REP

hwclient.c

REQ

REQ

0100

0005 4865 6c6c 6f

REP

0100

0005 576f 726c 64

Req端请求的消息，rep端不需要答复给req端

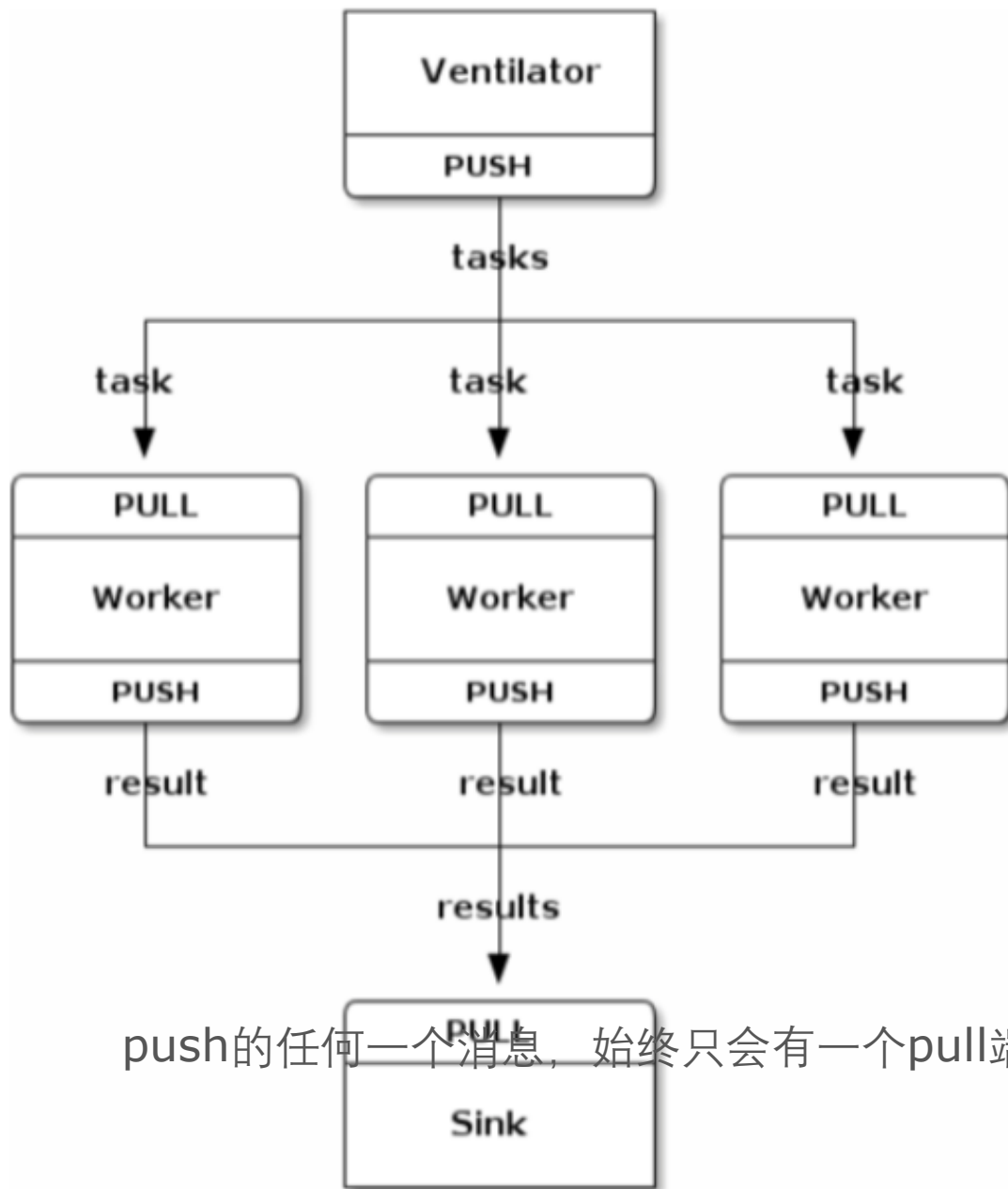


5.5 ZeroMQ模型2-PUB/SUB 发布订阅模型



发布者(Publisher)只能发送数据，它发送时指明发送数据的类型，订阅者(Subscriber)则只接收它关心的类型的消息。

5.5 ZeroMQ的推/拉堆栈模型



3. PUSH/PULL模型

taskwork.c

taskvent.c

tasksink.c

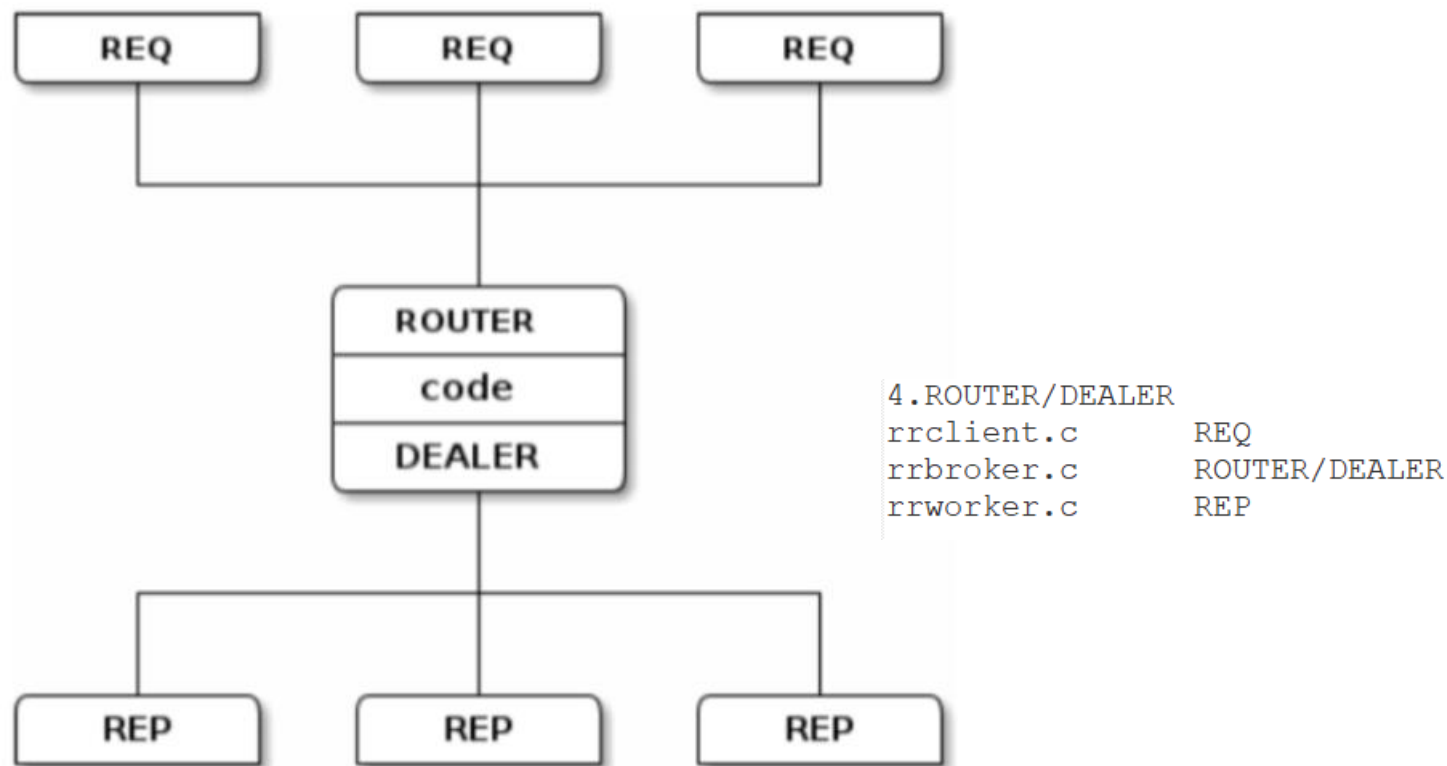
PULL-PUSH 可以多个work处理任务

PUSH 产生任务

PULL 汇总处理结果

push的任何一个消息，始终只会有一个pull端收到消息

5.5 ZeroMQ模型4-Router/Dealer 模型



代理应答与请求，router/dealer类似负载均衡的角色，可以分配不同的REP响应REQ



5.6 ZeroMQ 应用范例

见代码





零声学院

www.0voice.com

一切只为渴望更优秀的你!

为您的职业
添砖加瓦
升职加薪

努力方向

系统提升

项目实战

全职指导



零声学院 | C/C++架构师课程 | Darren老师: 326873713 | 柚子老师: 2690491738

下 节 课 再 见



联系Darren老师



课程顾问微信



课程咨询微信: 2207032995



零声学院 | C/C++架构师课程 | Darren老师: 326873713 | 柚子老师: 2690491738