NSQ 开源分布式消息中间件

郑佳豪

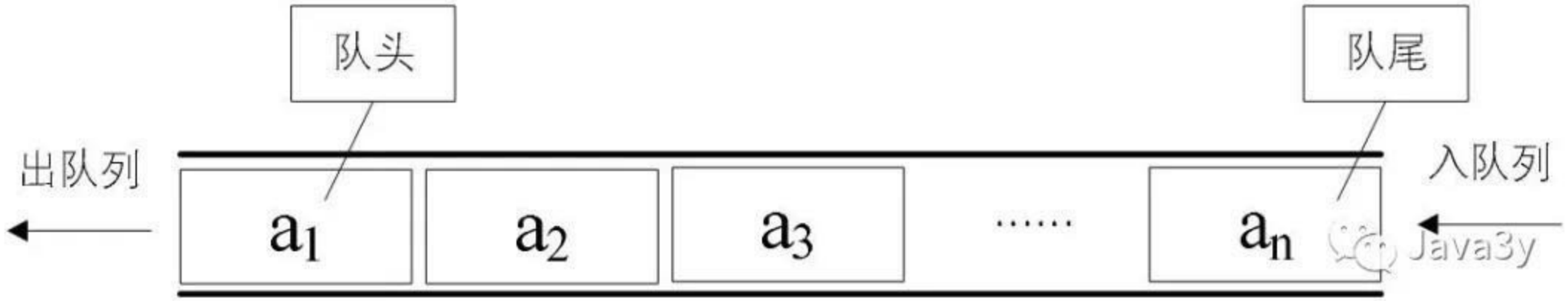
GitHub: Jiahonzheng

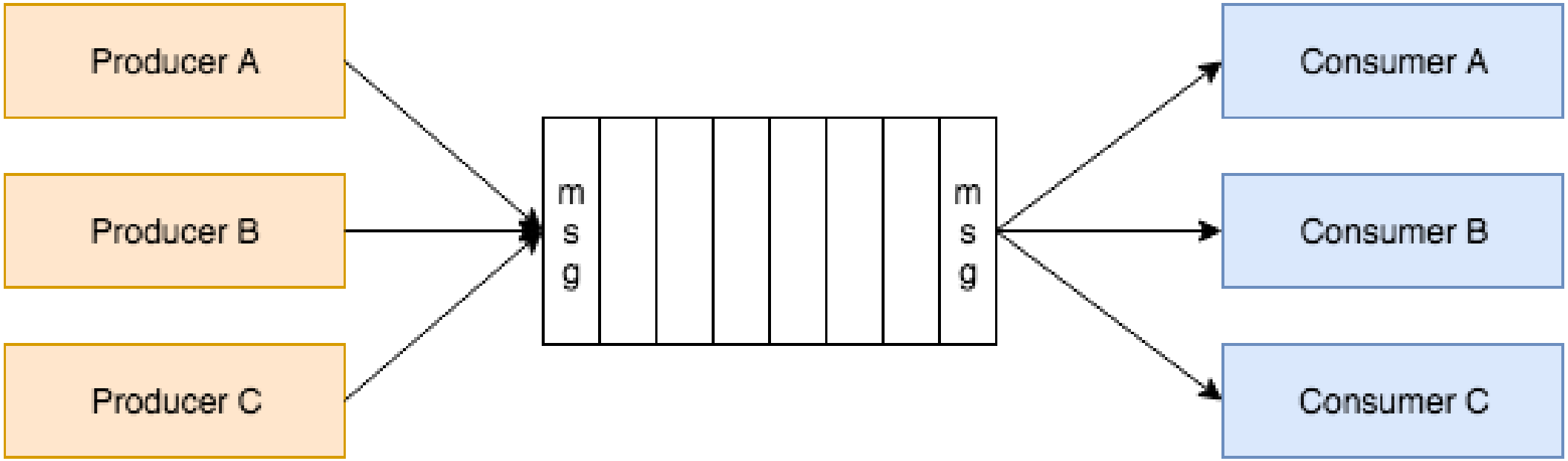
消息队列

首先，我们得了解**什么是消息队列**。

队列（Queue）是一种**先进先出**（FIFO）的数据结构：在下图中，人员 A 可从右边（**队尾**）添加数据，人员 B 可从左边（**队头**）消费数据。

人员 A ：**生产者**（Producer）
人员 B ：**消费者**（Consumer）




Producer A
Producer B
Producer C
m
s
g
m
s
g
Consumer A
Consumer B
Consumer C

消息队列

在工业生产中，我们使用消息队列的原因，主要是因为它的三个特性：

- 解耦
  衔接**不变量**和**变量**的桥梁
- 异步
  常见场景：发送注册邮件、发送短信验证码
- 削峰
  常见场景：618、双十一、商城秒杀活动

在 LongSight 中，使用消息队列（初定是 NSQ）的原因是出于**解耦**的考虑，当然还有 NSQ 本身的**实时**特性（主要是不想造轮子~）。

工作模式

市面上的消息队列，其工作模型有两种：**Pull**、**Push**。

Pull

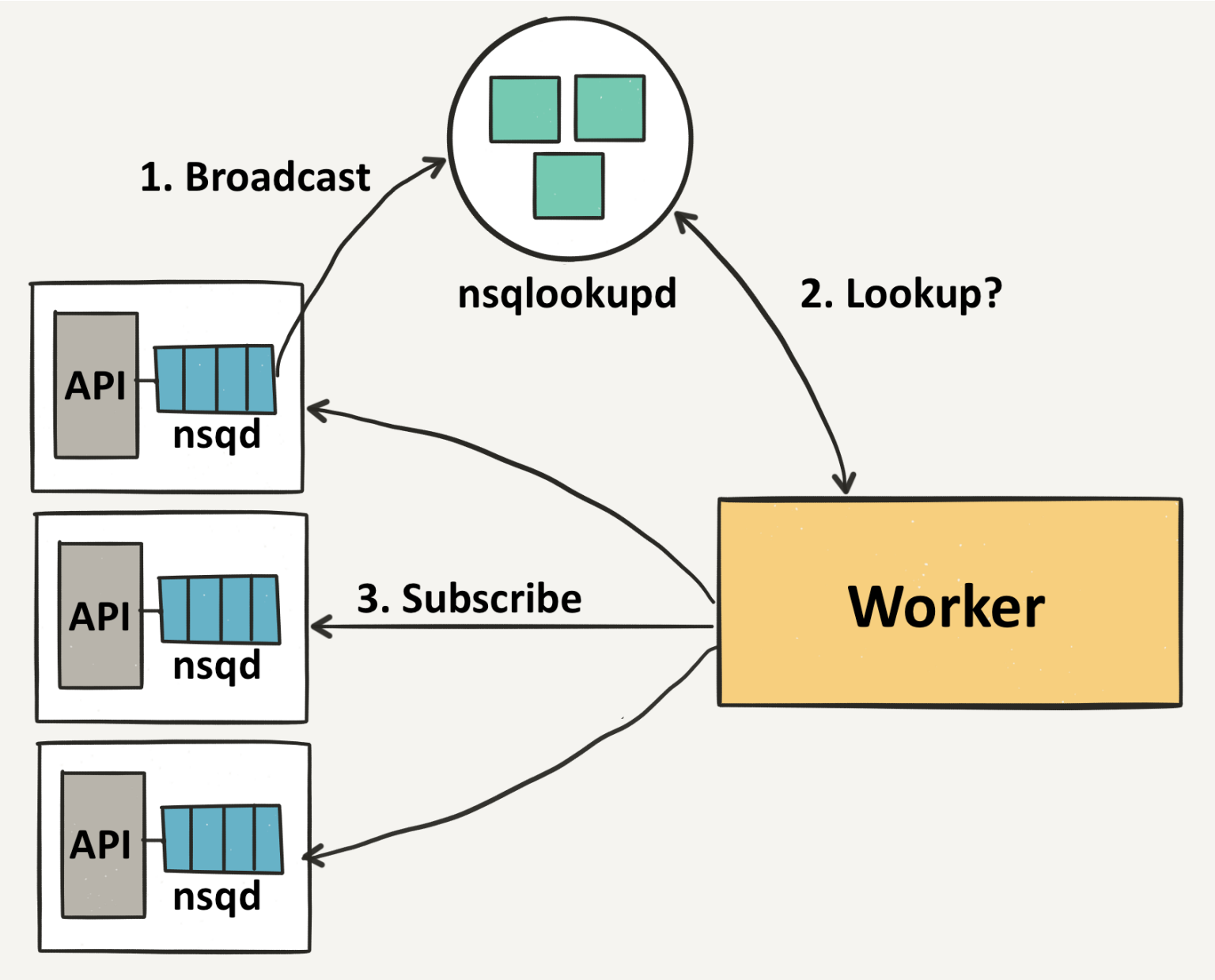
在此模式下，消费者可根据自身需要，主动从消息队列中**拉取**（Pull）消息，进行消费，可实现**流量控制**。但如果消息生产速率不过快，消费者会**轮询**，消耗 CPU 资源。此模式的典型中间件：Kafka。

Push

在此模式下，生产者可不受限于消费速度，实现更快地、批量地将数据**推送**（Push）至消息队列，因此**实时性比 Pull 模型要优异**，NSQ 使用的正是此工作模型。

NSQ 概念

- **topic**
  发布消息的一个**逻辑键**，即**消息的主题**。
- **channel**
  存放消息的队列，消息在发布时会被复制到所有 channel 上供消费者消费。若没有消费者读取消息，消息首先会在内存中排队，当量太大时就会被保存到磁盘中。
- **nsqd**
  负责**接收、排队、转发消息**到客户端的守护进程。
- **nsqlookupd**
  管理拓扑信息，提供最终一致性的服务发现服务。


1. Broadcast
nsqlookupd
2. Lookup?
API
nsqd
API
nsqd
3. Subscribe
Worker
API
nsqd

Guarantees

> NSQ 保证消息至少交付一次。

我们通过以下过程去理解 NSQ 的**消息担保**机制。

- 客户端连接至 NSQ，表示已准备好接收信息。
- NSQ 发送一条消息至客户端，并存储在本地（Re-Queue 或 Timeout 队列）。
- 客户端接收消息后，进行业务处理，并返回 FIN 或 REQ 响应，分别表示**消费成功**或**消费失败**。若客户端长时间没有响应，NSQ 则对消息进行重排列（**消息并没有丢失**）。

担保机制确保了导致消息丢失的唯一情况是**不正确地结束 nsqd 进程**。

在项目中使用 NSQ

为了实现**观测节点**与**PC**之间的解耦，我们引入 NSQ 消息队列。

- 每个观测节点，创建独自使用的 Channel 。
- PC 生产消息，观测节点消费信息，进行对应的操作。

参考链接

- 消息队列怎么能通俗点解释？
- NSQ 简介
- 消息中间件 NSQ 深入与实践

谢谢大家

> 感谢大家能够不厌其烦地听我 BB ~