<https://mp.weixin.qq.com/s?__biz=MjM5ODYxMDA5OQ==&mid=2651960012&idx=1&sn=c6af5c79ecead98daa4d742e5ad20ce5&chksm=bd2d07108a5a8e0624ae6ad95001c4efe09d7ba695f2ddb672064805d771f3f84bee8123b8a6&mpshare=1&scene=1&srcid=04054h4e90lz5Qc2YKnLNuvY>

# 什么时候该使用MQ？

## **一、缘起**

一切脱离业务的架构设计与新技术引入都是耍流氓。

引入一个技术之前，首先应该解答的问题是，这个技术解决什么问题。

就像微服务分层架构之前，应该首先回答，为什么要引入微服务，微服务究竟解决什么问题（详见《[互联网架构为什么要做微服务？](http://mp.weixin.qq.com/s?__biz=MjM5ODYxMDA5OQ==&mid=2651959519&idx=1&sn=065074b135fc9cb243abe897261e1a72&scene=21#wechat_redirect)》）。

最近分享了几篇MQ相关的文章：

《[MQ如何实现延时消息](http://mp.weixin.qq.com/s?__biz=MjM5ODYxMDA5OQ==&mid=2651959961&idx=1&sn=afec02c8dc6db9445ce40821b5336736&chksm=bd2d07458a5a8e5314560620c240b1c4cf3bbf801fc0ab524bd5e8aa8b8ef036cf755d7eb0f6&scene=21#wechat_redirect)》

《[MQ如何实现消息必达](http://mp.weixin.qq.com/s?__biz=MjM5ODYxMDA5OQ==&mid=2651959966&idx=1&sn=068a2866dcc49335d613d75c4a5d1b17&chksm=bd2d07428a5a8e54162ad8ea8e1e9302dfaeb664cecc453bd16a5f299820755bd2e1e0e17b60&scene=21#wechat_redirect)》

《[MQ如何实现幂等性](http://mp.weixin.qq.com/s?__biz=MjM5ODYxMDA5OQ==&mid=2651960002&idx=1&sn=c0775231bccf002c3178eabe43f1cdcb&chksm=bd2d071e8a5a8e08c3a5287247ea41dee6b2621e6ffafbf909ec1e8a866b7c816eeeea227246&scene=21#wechat_redirect)》

不少网友询问，究竟什么时候使用MQ，MQ究竟适合什么场景，故有了此文。

## **二、MQ是干嘛的**

消息总线（Message Queue），后文称MQ，是一种跨进程的通信机制，用于上下游传递消息。



在互联网架构中，MQ是一种非常常见的上下游“逻辑解耦+物理解耦”的消息通信服务。

使用了MQ之后，消息发送上游只需要依赖MQ，逻辑上和物理上都不用依赖其他服务。

## **三、什么时候不使用消息总线**



既然MQ是互联网分层架构中的解耦利器，那所有通讯都使用MQ岂不是很好？**这是一个严重的误区**，调用与被调用的关系，是无法被MQ取代的。

MQ的**不足**是：

1）系统更复杂，多了一个MQ组件

2）消息传递路径更长，延时会增加

3）消息可靠性和重复性互为矛盾，消息不丢不重难以同时保证

4）上游无法知道下游的执行结果，这一点是很致命的

举个**栗子**：用户登录场景，登录页面调用passport服务，passport服务的执行结果直接影响登录结果，此处的“登录页面”与“passport服务”就必须使用调用关系，而不能使用MQ通信。

无论如何，记住这个**结论**：**调用方实时依赖执行结果的业务场景，请使用调用，而不是MQ**。

## **四、什么时候使用MQ**

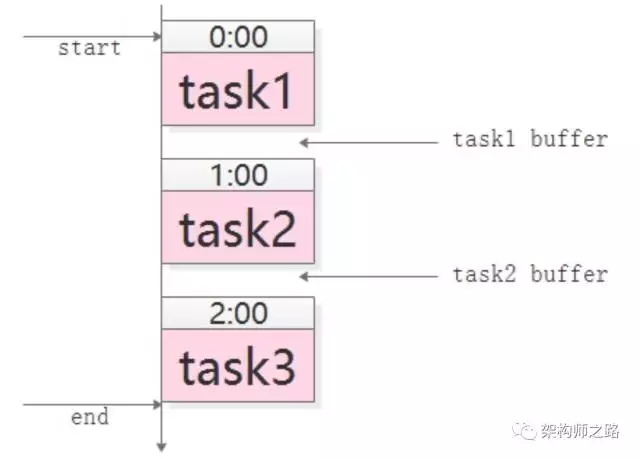
### ****【典型场景一：数据驱动的任务依赖】****

 什么是任务依赖，举个**栗子**，互联网公司经常在凌晨进行一些数据统计任务，这些任务之间有一定的依赖关系，比如：

1）task3需要使用task2的输出作为输入

2）task2需要使用task1的输出作为输入

这样的话，tast1, task2, task3之间就有任务依赖关系，必须task1先执行，再task2执行，载task3执行。



对于这类需求，**常见的实现方式**是，使用cron人工排执行时间表：

1）task1，0:00执行，经验执行时间为50分钟

2）task2，1:00执行（为task1预留10分钟buffer），经验执行时间也是50分钟

3）task3，2:00执行（为task2预留10分钟buffer）

这种方法的**坏处**是：

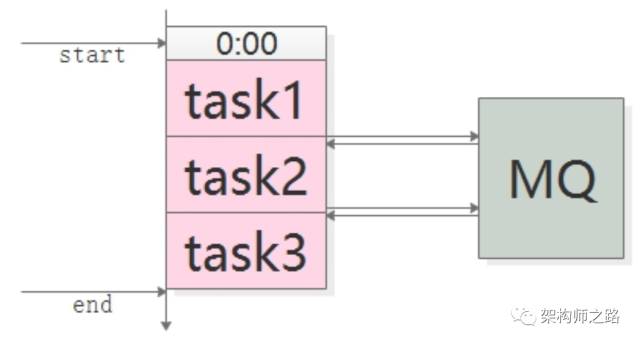
1）如果有一个任务执行时间超过了预留buffer的时间，将会得到错误的结果，因为后置任务不清楚前置任务是否执行成功，此时要手动重跑任务，还有可能要调整排班表

2）总任务的执行时间很长，总是要预留很多buffer，如果前置任务提前完成，后置任务不会提前开始

3）如果一个任务被多个任务依赖，这个任务将会称为关键路径，排班表很难体现依赖关系，容易出错

4）如果有一个任务的执行时间要调整，将会有多个任务的执行时间要调整

无论如何，采用“cron排班表”的方法，各任务耦合，**谁用过谁痛谁知道**（采用此法的请评论留言）



优化方案是，采用MQ解耦：

1）task1准时开始，结束后发一个“task1 done”的消息

2）task2订阅“task1 done”的消息，收到消息后第一时间启动执行，结束后发一个“task2 done”的消息

3）task3同理

采用MQ的**优点**是：

1）不需要预留buffer，上游任务执行完，下游任务总会在第一时间被执行

2）依赖多个任务，被多个任务依赖都很好处理，只需要订阅相关消息即可

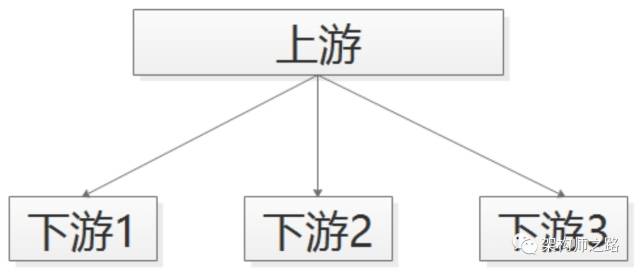
3）有任务执行时间变化，下游任务都不需要调整执行时间

需要**特别说明**的是，MQ只用来传递上游任务执行完成的消息，并不用于传递真正的输入输出数据。

### ****【典型场景二：上游不关心执行结果】****

上游需要关注执行结果时要用“调用”，上游不关注执行结果时，就可以使用MQ了。

举个**栗子**，58同城的很多下游需要关注“用户发布帖子”这个事件，比如招聘用户发布帖子后，招聘业务要奖励58豆，房产用户发布帖子后，房产业务要送2个置顶，二手用户发布帖子后，二手业务要修改用户统计数据。



对于这类需求，**常见的实现方式**是，使用调用关系：

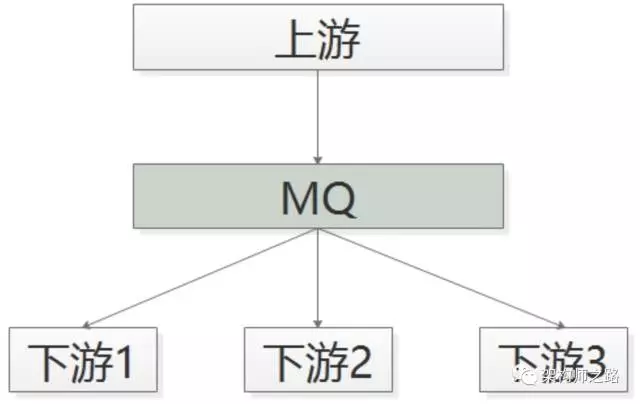
帖子发布服务执行完成之后，调用下游招聘业务、房产业务、二手业务，来完成消息的通知，但事实上，这个通知是否正常正确的执行，帖子发布服务根本不关注。

这种方法的**坏处**是：

1）帖子发布流程的执行时间增加了

2）下游服务当机，可能导致帖子发布服务受影响，上下游逻辑+物理依赖严重

3）每当增加一个需要知道“帖子发布成功”信息的下游，修改代码的是帖子发布服务，这一点是最恶心的，属于架构设计中典型的依赖倒转，**谁用过谁痛谁知道**（采用此法的请评论留言）



优化方案是，采用MQ解耦：

1）帖子发布成功后，向MQ发一个消息

2）哪个下游关注“帖子发布成功”的消息，主动去MQ订阅

采用MQ的**优点**是：

1）上游执行时间短

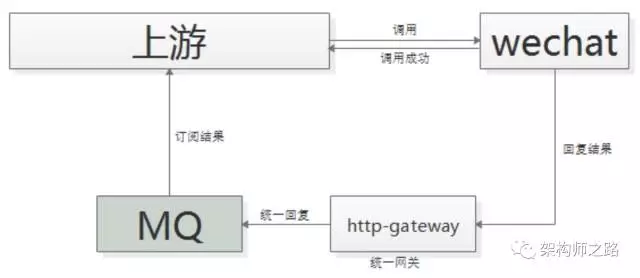
2）上下游逻辑+物理解耦，除了与MQ有物理连接，模块之间都不相互依赖

3）新增一个下游消息关注方，上游不需要修改任何代码

### ****【典型场景三：上游关注执行结果，但执行时间很长】****

 有时候上游需要关注执行结果，但执行结果时间很长（典型的是调用离线处理，或者跨公网调用），也经常使用**回调网关+MQ**来解耦。

举个**栗子**，微信支付，跨公网调用微信的接口，执行时间会比较长，但调用方又非常关注执行结果，此时一般怎么玩呢？



一般采用“回调网关+MQ”方案来解耦：

1）调用方直接跨公网调用微信接口

2）微信返回调用成功，此时并不代表返回成功

3）微信执行完成后，回调统一网关

4）网关将返回结果通知MQ

5）请求方收到结果通知

这里**需要注意**的是，不应该由回调网关来调用上游来通知结果，如果是这样的话，每次新增调用方，回调网关都需要修改代码，仍然会反向依赖，使用**回调网关+MQ**的方案，新增任何对微信支付的调用，都不需要修改代码啦。

**五、总结**

**MQ是一个互联网架构中常见的解耦利器。**

# **什么时候不使用MQ？**

上游实时关注执行结果