

# RabbitMQ



# 目录

- ◆ MQ 的基本概念
- ◆ RabbitMQ 的安装和配置
- ◆ RabbitMQ 快速入门
- ◆ RabbitMQ 的工作模式



# 1. MQ 的基本概念

## 1.1 MQ概述

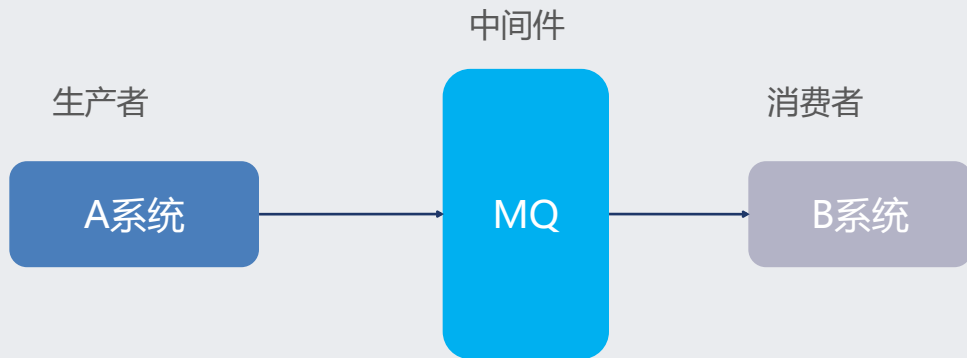
MQ全称 **M**essage **Q**ueue（消息队列），是在消息的传输过程中保存消息的容器。多用于分布式系统之间进行通信。



# 1. MQ 的基本概念

## 1.1 MQ概述

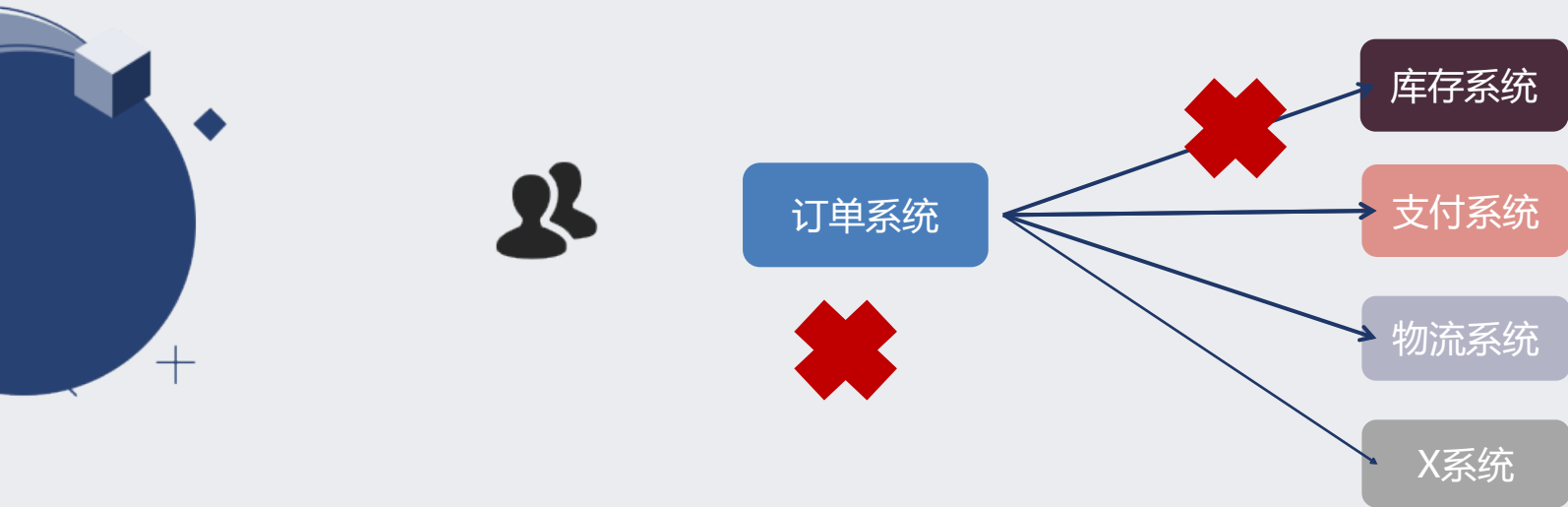
MQ全称 **M**essage **Q**ueue（消息队列），是在消息的传输过程中保存消息的容器。多用于分布式系统之间进行通信。



# 1. MQ 的基本概念

## 1.3 MQ 的优势

### 1. 应用解耦

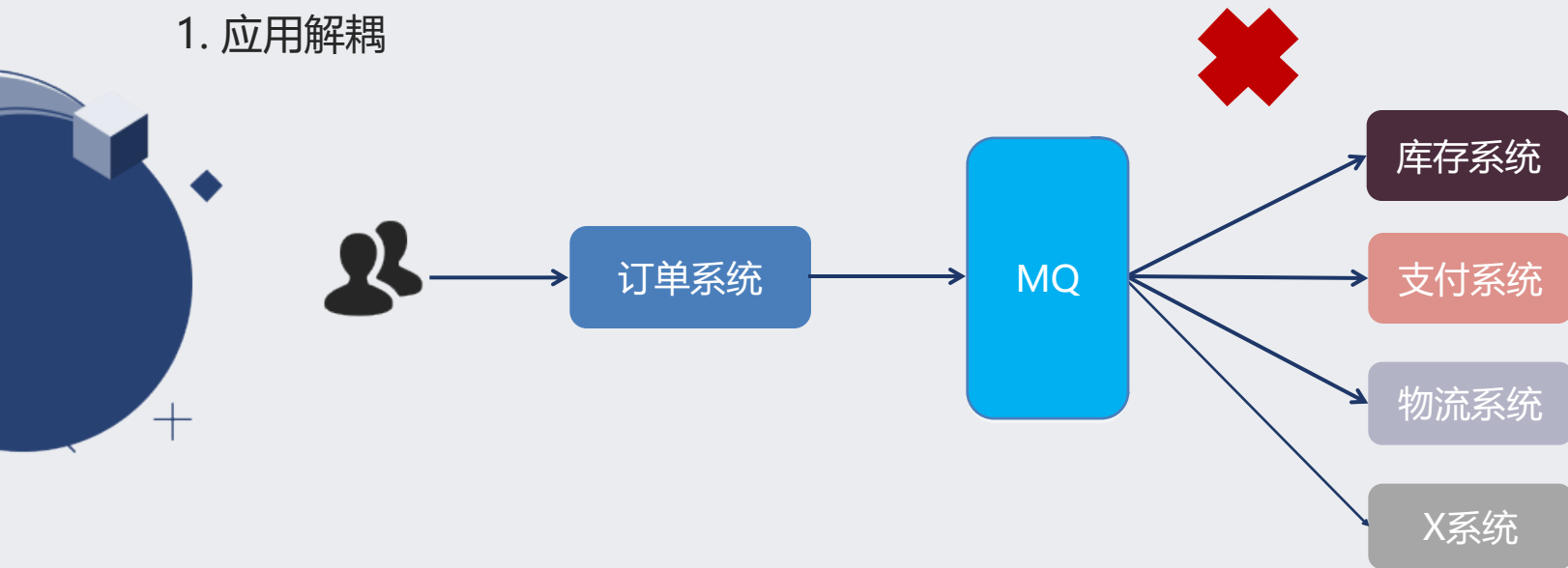


系统的耦合性越高，容错性就越低，可维护性就越低。

# 1. MQ 的基本概念

## 1.3 MQ 的优势

### 1. 应用解耦

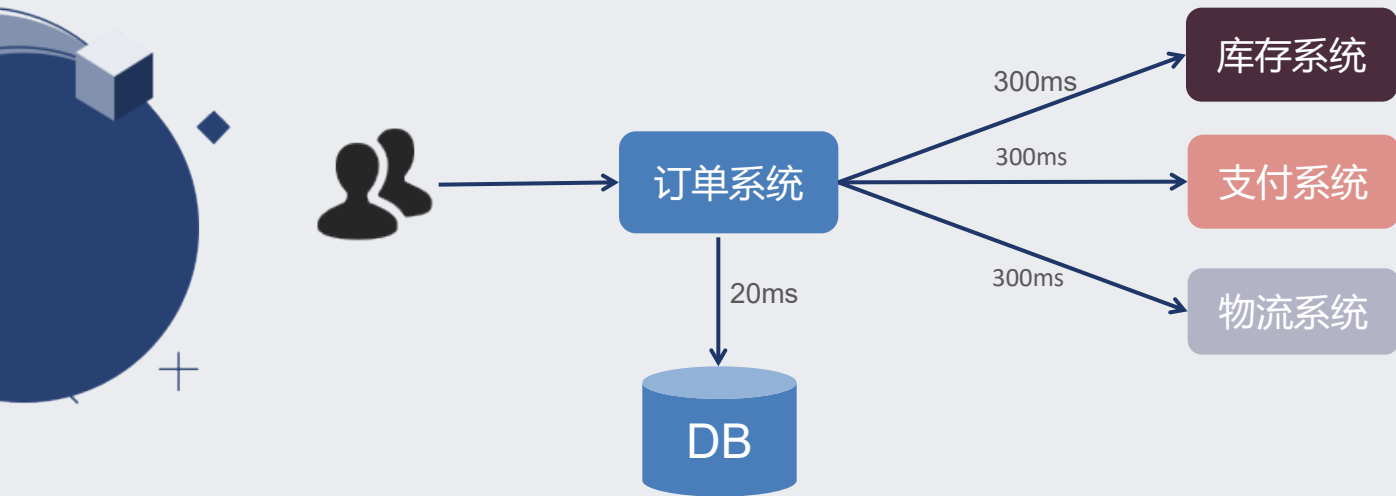


使用 MQ 使得应用间解耦，提升容错性和可维护性。

# 1. MQ 的基本概念

## 1.3 MQ 的优势

### 2. 异步提速



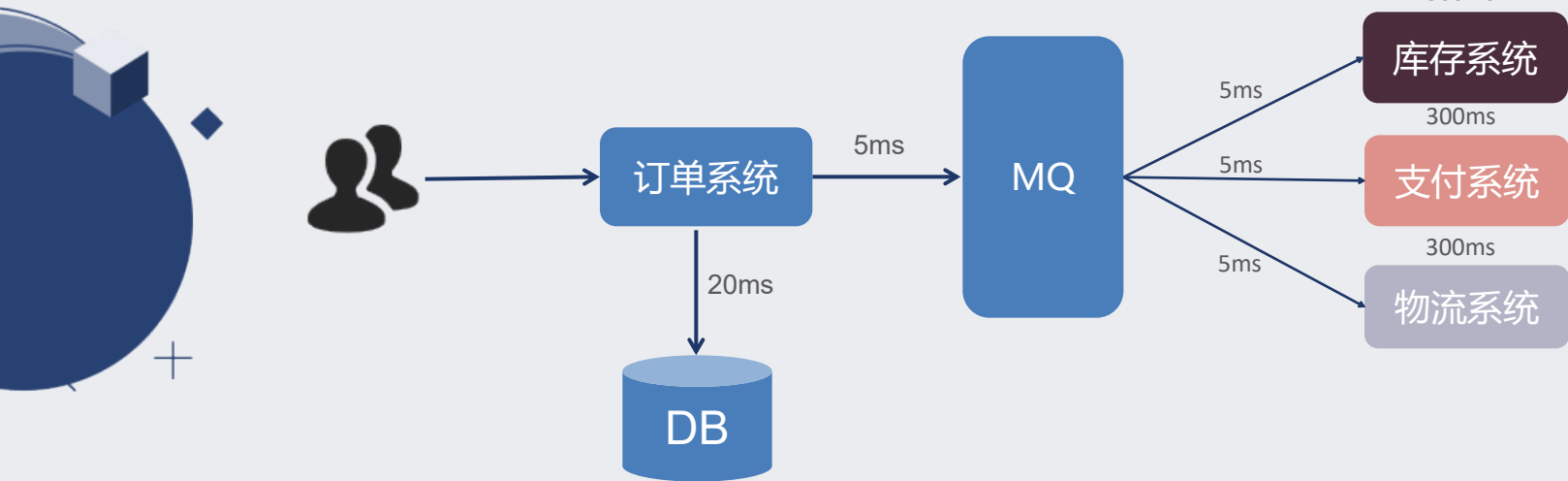
一个下单操作耗时： $20 + 300 + 300 + 300 = 920\text{ms}$

用户点击完下单按钮后，需要等待920ms才能得到下单响应，太慢！

# 1. MQ 的基本概念

## 1.3 MQ 的优势

### 2. 异步提速



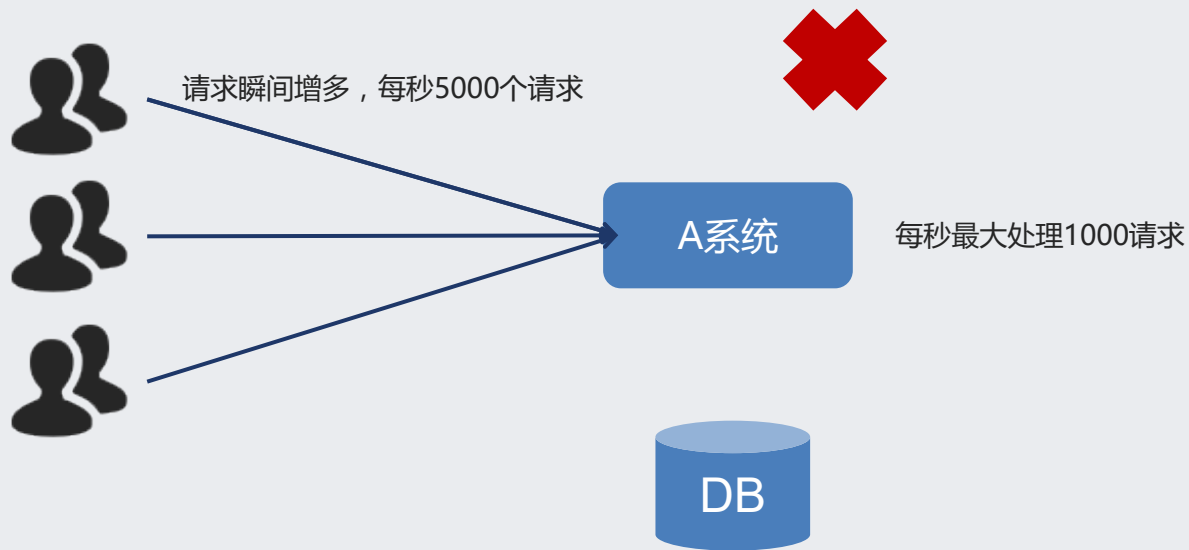
用户点击完下单按钮后，只需等待25ms就能得到下单响应 ( $20 + 5 = 25\text{ms}$ )。

提升用户体验和系统吞吐量（单位时间内处理请求的数目）。



## 1.3 MQ 的优势

### 3. 削峰填谷

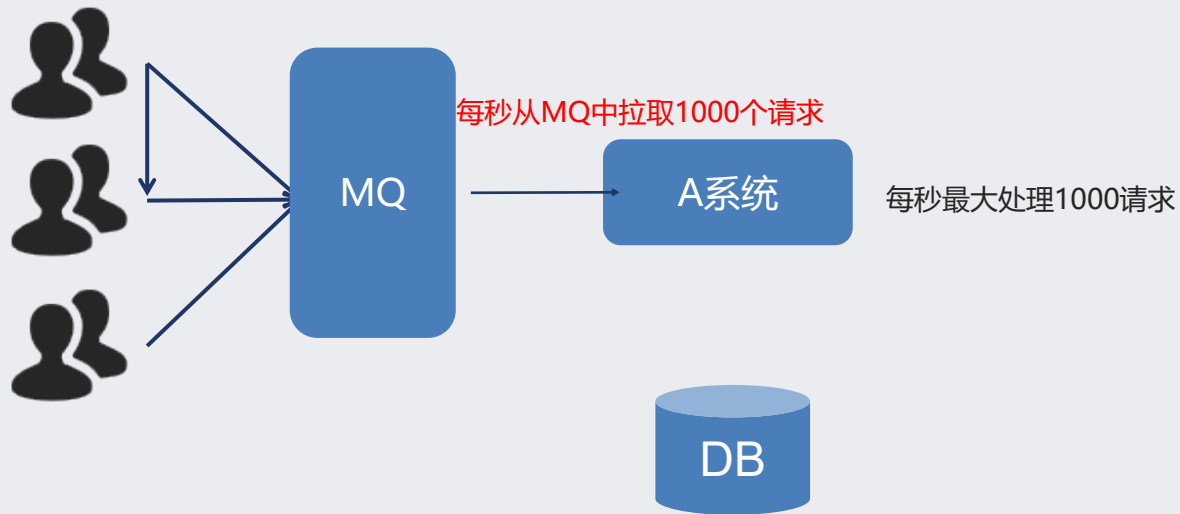


# 1. MQ 的基本概念

## 1.3 MQ 的优势

### 3. 削峰填谷

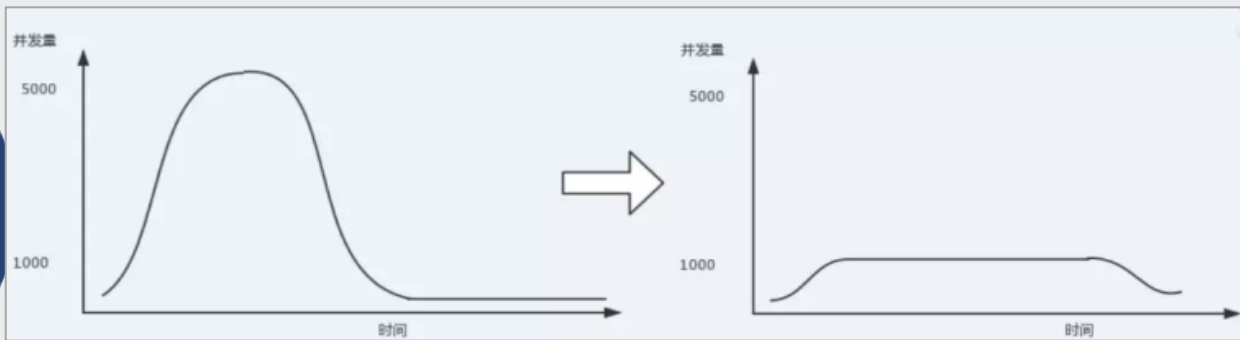
请求瞬间增多，每秒5000个请求



# 1. MQ 的基本概念

## 1.3 MQ 的优势

### 3. 削峰填谷



使用了 MQ 之后，限制消费消息的速度为1000，这样一来，高峰期产生的数据势必会被积压在 MQ 中，高峰就被“削”掉了，但是因为消息积压，在高峰期过后的一段时间内，消费消息的速度还是会维持在1000，直到消费完积压的消息，这就叫做“填谷”。

使用MQ后，可以提高系统稳定性。

# 1. MQ 的基本概念

## 1.3 MQ 的优势

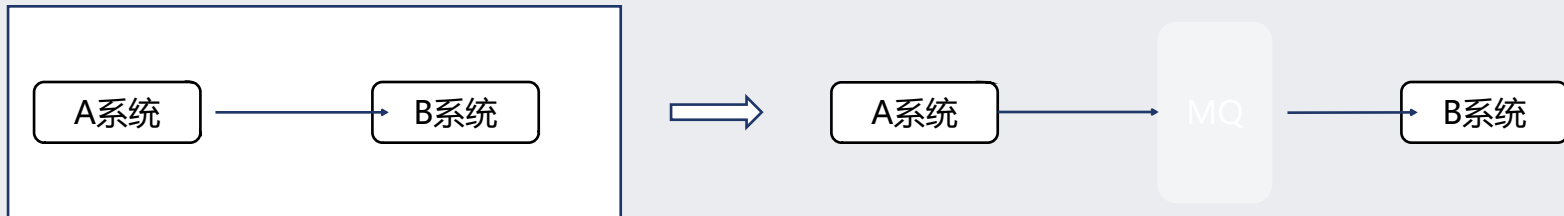
### 小结

- 应用解耦：提高系统容错性和可维护性
- 异步提速：提升用户体验和系统吞吐量
- 削峰填谷：提高系统稳定性



# 1. MQ 的基本概念

## 1.4 MQ 的劣势



- 系统可用性降低

系统引入的外部依赖越多，系统稳定性越差。一旦 MQ 宕机，就会对业务造成影响。如何保证MQ的高可用？

- 系统复杂度提高

MQ 的加入大大增加了系统的复杂度，以前系统间是同步的远程调用，现在是通过 MQ 进行异步调用。如何保证消息不被丢失等情况？

# 1. MQ 的基本概念

## 1.5 常见的 MQ 产品

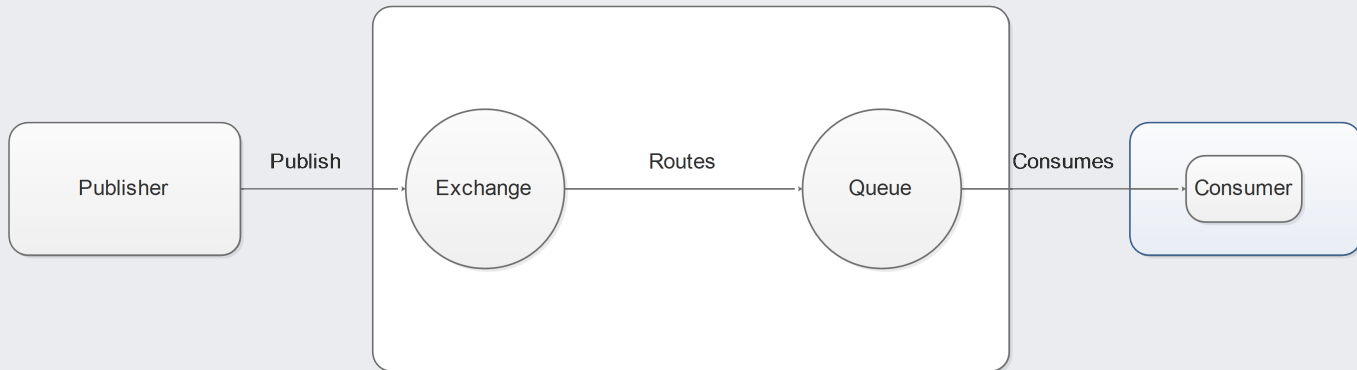
	RabbitMQ	ActiveMQ	RocketMQ	Kafka
公司/社区	Rabbit	Apache	阿里	Apache
开发语言	Erlang	Java	Java	Scala&Java
协议支持	AMQP , XMPP , SMTP , STOMP	OpenWire,STOMP , REST,XMPP,AMQP	自定义	自定义协议, 社区封装了http协议支持
客户端支持语言	官方支持Erlang , Java , Ruby等,社区产出多种API , 几乎支持所有语言	Java , C , C++ , Python , PHP , Perl , . net等	Java , C++ ( 不成熟 )	官方支持Java,社区产出多种API , 如PHP , Python等
单机吞吐量	万级 ( 其次 )	万级 ( 最差 )	十万级 ( 最好 )	十万级 ( 次之 )
消息延迟	微妙级	毫秒级	毫秒级	毫秒以内
功能特性	并发能力强, 性能极其好 , 延时低, 社区活跃, 管理界面丰富	老牌产品, 成熟度高, 文档较多	MQ功能比较完备, 扩展性佳	只支持主要的MQ功能 , 毕竟是为大数据领域准备的。

# 1. MQ 的基本概念

## 1.6 RabbitMQ 简介

AMQP，即 **Advanced Message Queuing Protocol**（高级消息队列协议），是一个网络协议，是应用层协议的一个开放标准，为面向消息的中间件设计。基于此协议的客户端与消息中间件可传递消息，并不受客户端/中间件不同产品，不同的开发语言等条件的限制。2006年，AMQP 规范发布。类比HTTP。

Hello, world example routing

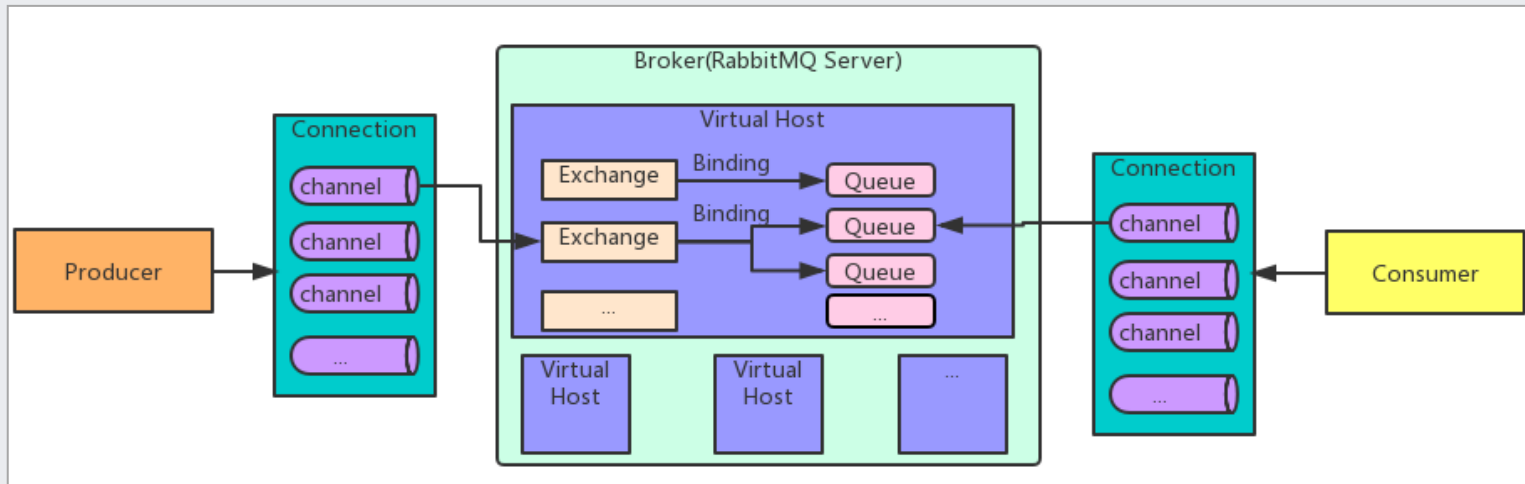


# 1. MQ 的基本概念

## 1.6 RabbitMQ 简介

2007年，Rabbit 技术公司基于 AMQP 标准开发的 RabbitMQ 1.0 发布。RabbitMQ 采用 Erlang 语言开发。Erlang 语言由 Ericson 设计，专门为开发高并发和分布式系统的一种语言，在电信领域使用广泛。

RabbitMQ 基础架构如下图：





# 1. MQ 的基本概念

## 1.6 RabbitMQ 简介

RabbitMQ 中的相关概念：

- **Broker**：接收和分发消息的应用，RabbitMQ Server就是 Message Broker
- **Virtual host**：出于多租户和安全因素设计的，把 AMQP 的基本组件划分到一个虚拟的分组中，类似于网络中的 namespace 概念。当多个不同的用户使用同一个 RabbitMQ server 提供的服务时，可以划分出多个vhost，每个用户在自己的 vhost 创建 exchange / queue 等
- **Connection**：publisher / consumer 和 broker 之间的 TCP 连接
- **Channel**：如果每一次访问 RabbitMQ 都建立一个 Connection，在消息量大的时候建立 TCP Connection的开销将是巨大的，效率也较低。Channel 是在 connection 内部建立的逻辑连接，如果应用程序支持多线程，通常每个thread创建单独的 channel 进行通讯，AMQP method 包含了channel id 帮助客户端和message broker 识别 channel，所以 channel 之间是完全隔离的。Channel 作为轻量级的 Connection 极大减少了操作系统建立 TCP connection 的开销



# 1. MQ 的基本概念

## 1.6 RabbitMQ 简介

RabbitMQ 中的相关概念：

- **Exchange**：message 到达 broker 的第一站，根据分发规则，匹配查询表中的 routing key，分发消息到 queue 中去。常用的类型有：direct (point-to-point), topic (publish-subscribe) and fanout (multicast)
- **Queue**：消息最终被送到这里等待 consumer 取走
- **Binding**：exchange 和 queue 之间的虚拟连接，binding 中可以包含 routing key。Binding 信息被保存到 exchange 中的查询表中，用于 message 的分发依据



# 1. MQ 的基本概念

## 1.6 RabbitMQ 简介

RabbitMQ 提供了 6 种工作模式：简单模式、work queues、Publish/Subscribe 发布与订阅模式、Routing 路由模式、Topics 主题模式、RPC 远程调用模式（远程调用，不太算 MQ；暂不作介绍）。

官网对应模式介绍：<https://www.rabbitmq.com/getstarted.html>

### 1 "Hello World!"

The simplest thing that does something



Python | Java | Ruby | PHP  
| C# | Javascript | Go

### 2 Work queues

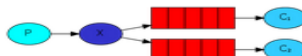
Distributing tasks among workers



Python | Java | Ruby | PHP  
| C# | Javascript | Go

### 3 Publish/Subscribe

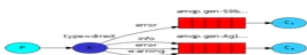
Sending messages to many consumers at once



Python | Java | Ruby | PHP  
| C# | Javascript | Go

### 4 Routing

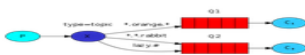
Receiving messages selectively



Python | Java | Ruby | PHP  
| C# | Javascript | Go

### 5 Topics

Receiving messages based on a pattern



Python | Java | Ruby | PHP  
| C# | Javascript | Go

### 6 ~~RPC~~

~~Remote procedure call implementation~~



~~Python | Java | Ruby | PHP  
| C# | Javascript | Go~~


# 目录






- ◆ MQ 的基本概念
- ◆ RabbitMQ 的安装和配置
- ◆ RabbitMQ 快速入门
- ◆ RabbitMQ 的工作模式




## 2. RabbitMQ 的安装和配置

- RabbitMQ 官方地址：<http://www.rabbitmq.com/>
- 安装文档：参考笔记RabbitMQ安装说明.pdf



名称	修改日期
 01RabbitMQ入门篇.pptx	2020/11/27 14
 02RabbitMQ与Spring_SpringBoot整合.ppt	2020/11/23 13
 03RabbitMQ高级特性.ppt	2020/11/26 14
 RabbitMQ安装说明.pdf	2020/11/27 13
 RabbitMQ集群搭建.pdf	2020/11/27 13





# 目录

- ◆ MQ 的基本概念
- ◆ RabbitMQ 的安装和配置
- ◆ RabbitMQ 快速入门
- ◆ RabbitMQ 的工作模式
- ◆ Spring 整合 RabbitMQ

## 3. RabbitMQ 快速入门

### 3.1 入门程序

需求：使用简单模式完成消息传递

步骤：

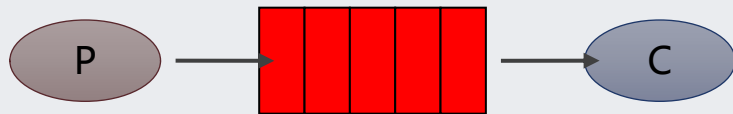
- ① 创建工程（生成者、消费者）
- ② 分别添加依赖
- ③ 编写生产者发送消息
- ④ 编写消费者接收消息



## 3. RabbitMQ 快速入门

### 3.2 小结

上述的入门案例中其实使用的是如下的简单模式：



在上图的模型中，有以下概念：

- P：生产者，也就是要发送消息的程序
- C：消费者：消息的接收者，会一直等待消息到来
- queue：消息队列，图中红色部分。类似一个邮箱，可以缓存消息；生产者向其中投递消息，消费者从其中取出消息



# 目录

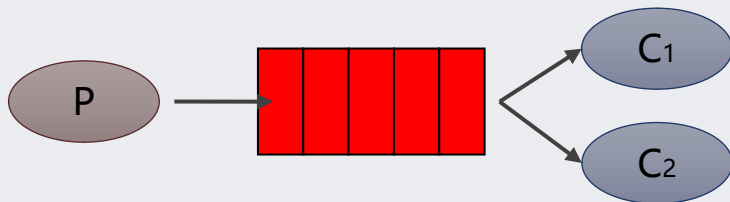
- ◆ MQ 的基本概念
- ◆ RabbitMQ 的安装和配置
- ◆ RabbitMQ 快速入门
- ◆ RabbitMQ 的工作模式
- ◆ Spring 整合 RabbitMQ



## 4. RabbitMQ 的工作模式

### 4.1 Work queues 工作队列模式

#### 1. 模式说明



- Work Queues：与入门程序的简单模式相比，多了一个或一些消费端，多个消费端共同消费同一个队列中的消息。
- 应用场景：对于任务过重或任务较多情况使用工作队列可以提高任务处理的速度。

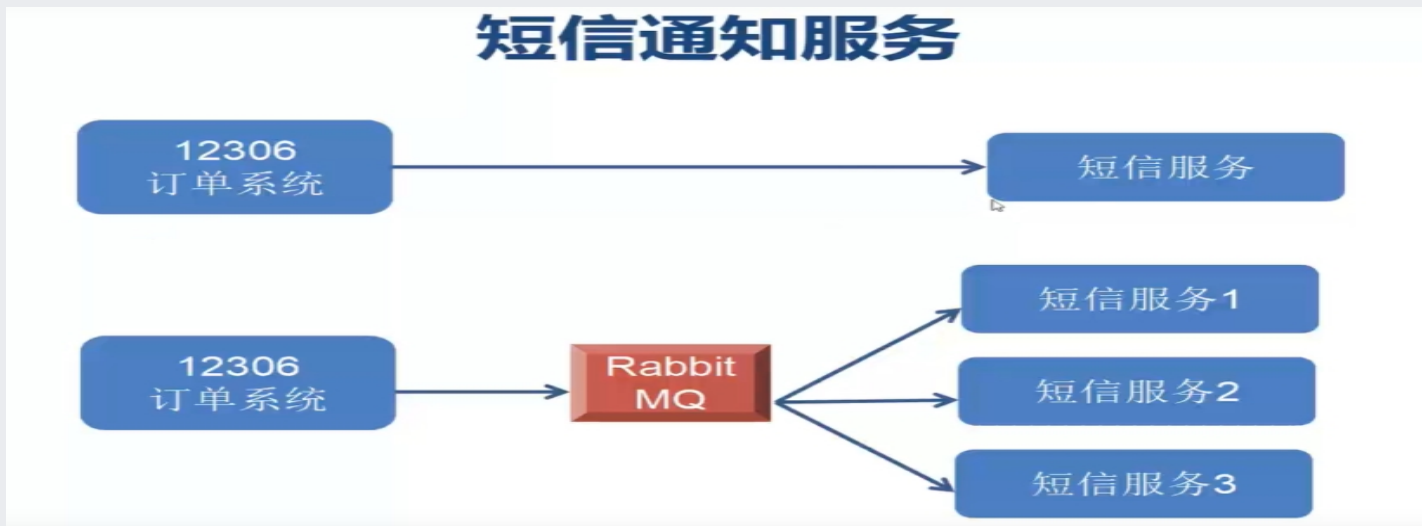
## 4. RabbitMQ 的工作模式

### 4.1 Work queues 工作队列模式

#### 2. 代码编写

**Work Queues** 与入门程序的简单模式的代码几乎是一样的。可以完全复制，并多复制一个消

◆ 费者进行多个消费者同时对消费消息的测试。



## 4. RabbitMQ 的工作模式

### 4.1 Work queues 工作队列模式

#### 3. 小结

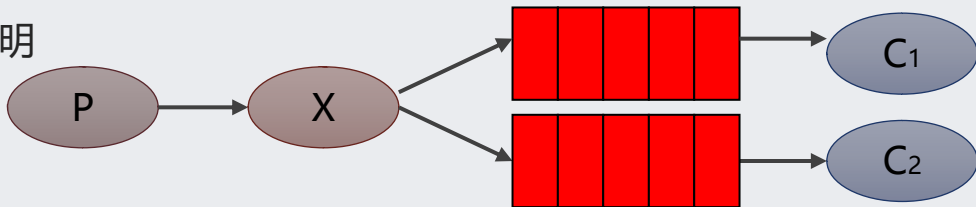
1. 在一个队列中如果有多个消费者，那么消费者之间对于同一个消息的关系是**竞争**的关系。
2. **Work Queues** 对于任务过重或任务较多情况使用工作队列可以提高任务处理的速度。  
。例如：短信服务部署多个，只需要有一个节点成功发送即可。



# 4. RabbitMQ 的工作模式

## 4.2 Pub/Sub 订阅模式

### 1. 模式说明



◆ 在订阅模型中，多了一个 Exchange 角色，而且过程略有变化：

- P：生产者，也就是要发送消息的程序，但是不再发送到队列中，而是发给X（交换机）
- C：消费者，消息的接收者，会一直等待消息到来
- Queue：消息队列，接收消息、缓存消息
- Exchange：交换机（X）。一方面，接收生产者发送的消息。另一方面，知道如何处理消息，例如递交给某个特别队列、递交给所有队列、或是将消息丢弃。到底如何操作，取决于Exchange的类型。Exchange有常见以下3种类型：
  - Fanout：广播，将消息交给所有绑定到交换机的队列
  - Direct：定向，把消息交给符合指定routing key 的队列
  - Topic：通配符，把消息交给符合routing pattern（路由模式）的队列

**Exchange**（交换机）只负责转发消息，不具备存储消息的能力，因此如果没有任何队列与 Exchange 绑定，或者没有

# 发布/订阅模式使用场景

- ◆ 发布订阅模式因为所有消费者获得相同的消息，所以特别适合“数据提供商与应用商”。
- ◆ 例如：中国气象局提供“天气预报”送入交换机，网易、新浪、百度、搜狐等门户接入通过队列绑定到该交换机，自动获取气象局推送的气象数据。

## 4. RabbitMQ 的工作模式

### 4.3 Pub/Sub 订阅模式

#### 3. 小结

1. 交换机需要与队列进行绑定，绑定之后；一个消息可以被多个消费者都收到。
2. 发布订阅模式与工作队列模式的区别：
  - 工作队列模式不用定义交换机，而发布/订阅模式需要定义交换机
  - 发布/订阅模式的生产方是面向交换机发送消息，工作队列模式的生产方是面向队列发送消息(底层使用默认交换机)
  - 发布/订阅模式需要设置队列和交换机的绑定，工作队列模式不需要设置，实际上工作队列模式会将队列绑定到默认的交换机

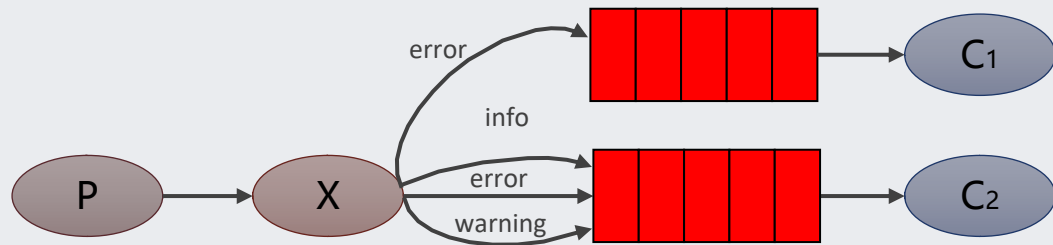


## 4. RabbitMQ 的工作模式

### 4.3 Routing 路由模式

#### 1. 模式说明：

- 队列与交换机的绑定，不能是任意绑定了，而是要指定一个 RoutingKey (路由key)
- 消息的发送方向 Exchange 发送消息时，也必须指定消息的 RoutingKey
- Exchange 不再把消息交给每一个绑定的队列，而是根据消息的 Routing Key 进行判断，只有队列的 Routingkey 与消息的 Routing key 完全一致，才会接收到消息



图解：

- P：生产者，向 Exchange 发送消息，发送消息时，会指定一个 routing key
- X：Exchange（交换机），接收生产者的消息，然后把消息递交给与 routing key 完全匹配的队列
- C1：消费者，其所在队列指定了需要 routing key 为 error 的消息
- C2：消费者，其所在队列指定了需要 routing key 为 info、error、warning 的消息



## 4. RabbitMQ 的工作模式

### 4.3 Routing 路由模式

#### 3. 小结

**Routing** 模式要求队列在绑定交换机时要指定 **routing key** , 消息会转发到符合 routing key 的队列。



## 4. RabbitMQ 的工作模式

### 4.4 Topics 通配符模式

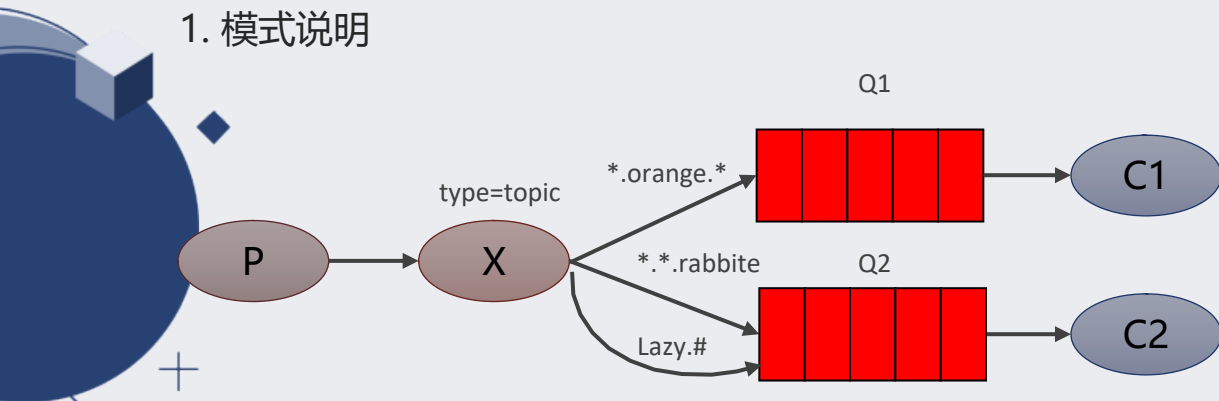
#### 1. 模式说明

- Topic 类型与 Direct 相比，都是可以根据 RoutingKey 把消息路由到不同的队列。只不过 Topic 类型Exchange 可以让队列在绑定 Routing key 的时候使用通配符！
- Routingkey 一般都是有一个或多个单词组成，多个单词之间以“.”分割，例如：item.insert
- 通配符规则：# 匹配一个或多个词，\* 匹配不多不少恰好1个词，例如：item.# 能够匹配 item.insert.abc 或者 item.insert，item.\* 只能匹配 item.insert

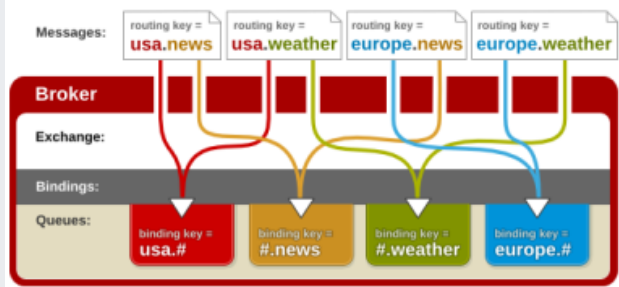
## 4. RabbitMQ 的工作模式

### 4.4 Topics 通配符模式

#### 1. 模式说明



Topic Exchange



图解：

- 红色 Queue：绑定的是 `usa.#`，因此凡是以 `usa.` 开头的 routing key 都会被匹配到
- 黄色 Queue：绑定的是 `#.news`，因此凡是以 `.news` 结尾的 routing key 都会被匹配

## 4. RabbitMQ 的工作模式

### 4.4 Topics 通配符模式

#### 3. 小结

Topic 主题模式可以实现 Pub/Sub 发布与订阅模式和 Routing 路由模式的功能，只是 Topic 在配置routing key 的时候可以使用通配符，显得更加灵活。



## 4. RabbitMQ 的工作模式

### 4.5 工作模式总结

#### 1. 简单模式 HelloWorld

一个生产者、一个消费者，不需要设置交换机（使用默认的交换机）。

#### 2. 工作队列模式 Work Queue

一个生产者、多个消费者（竞争关系），不需要设置交换机（使用默认的交换机）。

#### 3. 发布订阅模式 Publish/subscribe

需要设置类型为 fanout 的交换机，并且交换机和队列进行绑定，当发送消息到交换机后，交换机会将消息发送到绑定的队列。

#### 4. 路由模式 Routing

需要设置类型为 direct 的交换机，交换机和队列进行绑定，并且指定 routing key，当发送消息到交换机后，交换机会根据 routing key 将消息发送到对应的队列。

#### 5. 通配符模式 Topic

需要设置类型为 topic 的交换机，交换机和队列进行绑定，并且指定通配符方式的 routing key，当发送消息到交换机后，交换机会根据 routing key 将消息发送到对应的队列。



# RabbitMQ消息确认机制

- ◆ RabbitMQ在传递消息的过程中充当了代理人 (Broker) 的角色，那生产者 (Producer)怎样知道消息被正确投递到 Broker了呢？
- ◆ RabbitMQ提供了监听器 (Listener)来接收消息投递的状态。
- ◆ 消息确认涉及两种状态：Confirm与Return。

# Confirm & Return

- ◆ **Confirm**代表生产者将消息送到Broker时产生的状态，后续会出现两种情况：
  - ack 代表Broker已经将数据接收。
  - nack 代表Broker拒收消息。原因有多种，队列已满，限流，IO异常...
- ◆ **Return**代表消息被Broker正常接收 (ack)后，但Broker没有对应的队列进行投递时产生的状态，消息被退回给生产者。
- ◆ **注意：**上面两种状态只代表生产者与Broker之间消息投递的情况。与消费者是否接收/确认消息无关。