# 第一部分

## RabbitMQ和应用程序体系结构

在这本书得这一部分，我们将会探索AMQ协议构架，使用RabbitMQ实现你们的应用程序的通讯。我们也将会关注消息本身，使用消息头，优先等级和更多的属性来增强信息的交互。我们也会探索性能的取舍、平衡稳定性和事务安全性与高性能吞吐量之间的关系，而不提供任何保证。此外，我们还会研究不同的交换类型和它们的工作方式。

## 基础

## RabbitMQ的特性和优点

RabbitMQ有很多的特性和优点，重要的如下：

开源：原始开发的是LShift, LTD, and Cohesive FT为RabbitMQ的合作开发技术，现在是Pivotal Software Inc.拥有。而且由Mozilla颁布公共许可证。作为一个用Erlang编写的开源项目，RabbitMQ在充分利用其作为产品的关键地位的同时，享有自由和灵活性。在RabbitMQ社区的开发人员和工程师能够提供增强和附加组件，Pivotal能够提供商业支持，并为持续的提供稳定的成熟产品。

平台和RabbitMQ的使用没有关系：作为一个实现平台和厂商无关的高级消息队列协议（AMQP）规范的消息代理，几乎可以在任何编程语言和所有主要计算机平台上使用客户机。

轻量级：他是一个轻量级的，运行RabbitMQ核心模块和相关组件（如管理用户界面）使用不到40M的RAM内存，特别注意，增加他的消息到队列中，可以增加他的内存的使用率

客户库为现代的大多数编程语言：客户库针对大多数多平台现代语言，RabbitMQ是一个可以信任的程序。在选择如何编写将和RabbitMQ对接的程序时，没有平台或语言硬性条件。事实上，在用不同语言编写的应用程序之间使用RabbitMQ作为消息中心并不少见。RabbitMQ还提供了一个有用的链接功能，它允许Java、Ruby、Python、PHP、JavaScript和C\*\*等语言在操作系统和环境中的数据共享。

控制信息交互的灵活性：RabbitMQ提供了灵活性，表现在利用消息吞吐量和性能控制可靠消息传递的权衡方面。因为它不是一种“one size fits all”的应用程序类型，所以消息可以指明在传递之前是否应将其持久化到磁盘上，并且，如果在集群中配置，可以将队列设置为高度可用，跨越多个服务器，以确保在服务器故障时不会丢失消息。

用于高延时环境的插件：由于并不是全部的网络拓扑和系统构架一样的，RabbitMQ为消息提供了在低延迟环境中提供消息传递，和为高延迟环境（如Internet）的插件。这允许rabbitmq集成在同一个本地网络上，并在多个数据中心共享共享消息。

第三方插件：作为一个为程序集成的中心点，RabbitMQ提供了一个灵活的插件系统。例如，第三方插件用于将消息直接存储到数据库中，将RabbitMQ信息数据直接写入数据库中。

安全层面：在RabbitMQ中，安全被提供在多个层面中，客户连接的安全是由SSL-only通讯和客户证书认证来实现。用户访问权限被管理在虚拟主机级别，从而在高级别上隔离消息和资源。此外，通过正则表达式（regex）模式匹配来管理对配置功能的访问、从队列中读取和对交换的写入。最后，插件可以用于集成到外部认证系统（如LDAP）中。

我们将会在后面的章节中探讨这个列表中的特性，但现在我想重点介绍rabbitmq的两个最基本的特性：它在（erlang）中编程的语言和它基于的模型（高级消息队列模型），这是一个定义rabbitmq词汇及其行为的规范。

## RabbitMQ和Erlang

作为一个高性能，稳定且可集群的消息代理，RabbitMQ已经在关键任务环境中找到了一个归宿，例如大型消息传递架构的核心，这并不奇怪。 它是用Erlang编写的，这是一种由爱立信计算机科学实验室在20世纪80年代中后期设计的电信级函数式编程语言。 Erlang被设计为分布式，容错，软实时系统，适用于需要99.999％正常运行时间的应用。 作为一种语言和运行时系统，Erlang专注于轻量级进程，它们相互传递消息，提供高级并发性和无共享状态。

实时系统—实时系统是硬件平台，软件平台或两者的组合。 它需要定义何时必须响应事件返回。 软实时系统将牺牲执行任务的不太重要的最后期限，这将使更重要的任务受益。

Erlang的设计侧重于并发处理和消息传递，使其成为RabbitMQ等消息代理的自然选择：作为应用程序，消息代理维护并发连接，路由消息和管理其状态。 此外，Erlang的分布式通信架构使其成为RabbitMQ集群机制的理想选择。 RabbitMQ集群中的服务器使用Erlang的进程间通信（IPC）系统来卸载竞争性消息代理必须实现的许多功能，以便增加集群功能（图1.1）。

尽管RabbitMQ通过使用Erlang获得了优势，但Erlang环境可能是一个绊脚石。 学习一些Erlang会很有帮助，因此你有信心管理RabbitMQ配置文件并使用Erlang收集有关RabbitMQ当前运行时状态的信息。

在分布式集群之中，RabbitMQ使用Erlang来完成进程之间的通信。Erlang IPC即同步服务器，在RabbitMQ系统之间进行通信，通过它来传递消息，状态和配置。

服务器2

Erlang虚拟机

服务器1

Erlang虚拟机

服务器3

Erlang虚拟机

不同服务器

相互之间

Erlang IPC系统使用TCP / IP在两个RabbitMQ服务器之间进行通信。

图1.1：RabbitMQ分布式集群使用VM中的本地化的Erlang进程间通信机制进行跨节点通信，共享状态信息并允许在整个集群中发布和使用消息。

## RabbitMQ和AMQP

RabbitMQ于2007年首次发布，其可操作性，性能和稳定性是其开发过程的主要目标。 RabbitMQ是首批实现AMQP规范的消息代理之一。 在所有方面，它都是参考实现。 AMQP规范分为两部分。 它不仅定义了与RabbitMQ交互的有线协议，还定义了概述RabbitMQ核心功能的逻辑模型。

注意，AMQP规范有多个版本。 出于本书的目的，我们将仅关注AMQP 0-9-1。 虽然较新版本的RabbitMQ支持AMQP 1.0作为插件扩展，但核心RabbitMQ架构与AMQP 0-8和0-9-1更紧密相关。 AMQP规范由两个主要文档组成：描述AMQ模型和AMQ协议的顶级文档，以及提供有关每个类，方法，属性和字段的不同级别信息的更详细文档。 有关AMQP的更多信息，包括规格，可在以下网站找到：<http://www.amqp.org.http://www.amqp.org>.

有许多流行的消息代理和消息传递协议，你需要考虑协议和代理对你的应用程序的影响。 RabbitMQ支持AMQP，但它也支持其他的协议，例如MQTT。Stomp和XMPP。 和其他流行的消息代理相比，RabbitMQ的协议中的中立性和插件的可扩展性使RabbitMQ成为多协议应用程序架构的理想选择。它是AMQP规范中RabbitMQ的基本，并概述了其主要架构和通信方法。 这是评估RabbitMQ和其他消息代理的重要区别。 与AMQP一样，RabbitMQ是一种独立于供应商的平台，可满足面向消息传递的体系结构的复杂需求，例如灵活的消息路由，可配置的消息持久性和数据中心通信。

## 谁在使用RabbitMQ，怎么使用RabbitMQ

作为一个开源软件包，RabbitMQ正在迅速成为主流框架被采用，并为互联网上一些最大和最多的流量网站提供支持。众所周知，RabbitMQ已经可以在许多不同的环境和许多不同类型的公司和企业中运行：

Reddit：现在流行的在线社区Reddit在其应用平台的核心中广泛使用RabbitMQ，该平台每月服务数十亿页。 当用户在网站上注册，提交新闻帖或对链接进行投票时，会向RabbitMQ发送一条消息，供消费者应用程序进行异步处理。

NASA：NASA选择RabbitMQ作为其Nebula平台的消息代理，该平台是其服务器基础架构的集中式服务器管理平台，该平台已发展为OpenStack平台，OpenStack平台是一个非常流行的软件平台，用于构建私有云和公共云平台服务。

Agoura Games：RabbitMQ是Agoura Games面向社区的在线游戏平台的核心，可以传输大量的实时单人和多人游戏数据和事件

the Ocean Observations Initiative：对于海洋观测计划，RabbitMQ将关键的物理，化学，地质和生物数据传输到分布式研究计算机网络。 从南太平洋和大西洋的传感器收集的数据是国家科学基金会项目的一个组成部分，该项目涉及在海洋和海底建立大型传感器网络。

Rapportive：收件箱中的信息使用RabbitMQ作为其数据处理系统的粘合剂。 每月数十亿条消息通过RabbitMQ向Rapportive的爬行引擎和分析系统提供数据，并从其Web服务器卸载长时间运行的操作。

MercadoLibr：MercadoLibr是拉丁美洲最大的电子商务生态系统。 它使用RabbitMQ作为其企业服务总线（ESB）架构的核心，将数据与紧密耦合的应用程序分离，并允许在其应用程序架构中灵活地与各种组件集成。

AdMob：谷歌的AdMob移动广告网络使用RabbitMQ作为其Rock Steady项目的核心。 它将RabbitMQ的消息软件收集到Esper中，这是一个复杂的事件处理系统，用于实时索引分析和故障检测。

Aandhaar：印度的生物识别数据库系统Aandhaar使用RabbitMQ在其工作流程的各个阶段处理数据，为其监控工具，数据仓库和基于Hadoop的数据处理系统提供数据。 Aandhaar旨在为印度的每个居民提供在线便携式识别系统，覆盖12亿人口。

正如你所看到的，RabbitMQ不仅被互联网上的一些最大的网站使用，而且还被大规模科学研究的学者使用。 NASA发现它适合使用RabbitMQ来管理其网络基础架构核心的堆栈。 正如这些例子所示，RabbitMQ已被用于不同环境和行业的许多任务关键型应用程序，并取得了巨大成功。

## 低耦合的优点

当我第一次开始实现基于消息传递的体系结构时，，我一直在寻找一种方法来解耦与成员登录网站时相关的数据库更新。 该网站的发展速度非常快，而且由于我们编写网站的方式，它的原始设计并不是为了扩展。 当用户登录到Web站点时，需要使用登录时间戳更新多个数据库服务器的表（图1.2）。 此时间戳需要实时更新，因为网站上最有吸引力的活动部分是由时间戳值驱动的。 登录后，与活跃的在线用户相比，会员在社交游戏中享有优先权。

登陆程序

数 据 库

经过认证

更新会员信息

更新游戏

#1信息

更新游戏

#2信息

重定向现已验证的成员息

使用上次登录时间戳更新成员数据记录

更新游戏＃1表中的上次登录时间戳

更新游戏＃2表中的上次登录时间戳

图1.2：旧版本，一旦用户登录，每个数据库将按顺序依赖于时间戳更新。 你添加的表越多，所需的时间就越长。

随着网站数量的增加，会员登录所需的时间也在增加。原因很简单：添加使用成员上次登录时间戳的新应用程序时，其数据库表将通过删除跨数据库连接尽快携带该值。为了使数据保持最新和准确，当成员登录时会更新新表。不久之后，许多表都以这种方式维护。性能问题开始蔓延，因为数据库更新是连续执行的。更新成员上次登录时间戳的每个查询必须在下次启动之前完成。十个查询被认为是高性能，每个查询在50毫秒内完成，仅在数据库更新中增加了半秒。必须在发送授权响应之前完成所有这些查询，并将其重定向回用户。此外，数据库服务器上的任何操作问题都可能使问题变得更加复杂。如果数据库服务器缓慢或无响应地开始响应，则成员将无法再登录该站点。

为了将面向用户的登录应用程序与直接写入数据库分开，我研究了将消息发布到面向消息的中间件或集中式

然后，消息代理将消息分发给任何数量的写入所需数据库以供执行的使用者应用程序。 我尝试了几个不同的消息代理，最终我选择了RabbitMQ作为我的代理程序。

面向消息的中间件（MOM）定义为允许从分布式系统发送和接收消息的软件或硬件基础结构。 RabbitMQ可以轻松使用提供高级路由和消息分发的功能。即使具有广域网（WAN）容限，它也可以支持可以轻松与其他系统互连的可靠分布式系统。

在将登录过程与所需的数据库更新分开后，我发现了一个新的自由度。成员可以快速登录，因为我们在身份验证期间不再更新数据库。而是发布成员登录消息，其中包含更新任何数据库所需的所有信息以及独立更新每个数据库表的使用者应用程序（图1.3）。此登录消息不包含成员的身份验证信息，但仅包含维护成员在各种数据库和应用程序中的上次登录状态所需的信息。这允许我们通过更多控制水平扩展数据库写入。通过控制写入特定数据库服务器的使用者应用程序的数量，我们可以限制在新站点增长所产生的负载下开始变形的服务器的数据库写入次数，同时解决他们自己独特的扩展问题。

详细说明基于消息传递的体系结构的优点时，重要的是要注意这些优点也可能影响系统的性能，例如所描述的登录体系结构。任何数量的问题都可能影响发布了的程序的性能，从网络问题到对消息发布者的RabbitMQ限制。发生此类事件时，您的应用程序将看到性能下降。除了横向扩展消费者之外，计划横向扩展消息代理是明智的，以实现更好的消息吞吐量和发布者性能。

登陆程序

Rabbit

DB+

该成员进行身份验证，并将消息发布到RabbitMQ。 然后，应用程序将经过身份验证的成员重定向到登录的URL

RabbitMQ将登录事件消息发布给应该接收它的所有消费者

每个消费者独立执行自己的数据库任务

DB+

DB+

图1.3使用RabbitMQ之后：使用RabbitMQ，松散耦合的数据以异步方式独立发布到每个数据库，允许登录应用程序继续进行而无需等待任何数据库写入。

## 解耦程序

面向消息传递的中间件的使用可以为希望创建灵活的以数据为中心的应用程序体系结构的组织提供巨大的优势。 通过将RabbitMQ移动到松散耦合的设计，应用程序体系结构不再受数据库写入性能的限制，并且可以轻松添加新应用程序来操作数据，而无需触及任何核心应用程序。 考虑图1.4，演示了用于与数据库通信的紧密耦合应用程序的设计。

应用程序

在紧密耦合的应用程序中，数据库写入直接与数据库通信。

DB+

图1.4与数据库通信时，紧密耦合的应用程序必须等待数据库服务器响应才能继续处理。

## 解耦数据写入

在紧密耦合的体系结构中，应用程序必须等待数据库服务器的响应才能完成事务。此设计可能会在同步和异步应用程序中产生性能瓶颈。如果数据库服务器由于调整不佳或硬件问题而变慢，则应用程序将变慢。如果数据库停止响应或崩溃，应用程序可能会崩溃。

通过将数据库与应用程序分离，可以创建松散耦合的体系结构。在此体系结构中，RabbitMQ充当面向消息传递的中间件，用于在数据库中执行某些操作之前调解数据。消费者应用程序从RabbitMQ服务器检索数据并执行数据库操作（图1.5）。在此模型中，如果数据库需要脱机以进行维护，或者写入工作负载变得过重，则可以限制或停止用户应用程序。数据将保留在队列中，直到消费者可以接收消息。暂停或限制使用者应用程序行为的能力只是使用此类体系结构的一个优点

DB+

在松散耦合的应用程序中，应用程序将带有数据的消息发布到RabbitMQ。 消费者应用程序在收到每条消息时将数据库写入数据库传送给数据库。

在松散耦合的应用程序中，应用程序将带有数据的消息发布到RabbitMQ。

登陆程序

Rabbit

RabbitMQ将消息传递给订阅的消费者应用程序。

图1.5松散耦合的应用程序允许直接在数据库中保存数据的应用程序将数据发布到RabbitMQ，从而允许异步处理数据。

## 无缝添加新功能

使用RabbitMQ的松散耦合架构也可以重用数据。 最初仅写入数据库的数据也可用于其他目的。 RabbitMQ处理所有重复的消息内容，并可以将其路由到多个消费者用于多种目的（图1.6）

DB+

登陆程序

Rabbit

云服

应用程序没有任何改变; 它仍以同样的方式将数据发布到RabbitMQ。

原始使用者仍然管理应用程序的数据库写入

新的消费者可以将相同的数据提供给第三方基于云的服务

RabbitMQ现在将数据提供给两个消费者而不是一个

图1.6通过使用RabbitMQ，不需要更改发布应用程序，以便向新的基于云的服务和原始数据库提供相同的数据。

## 复制数据和事件

RabbitMQ扩展了该模型，为跨数据中心的数据分发提供内置工具，允许联合交付和应用程序同步。 Federation允许RabbitMQ将消息推送到远程RabbitMQ实例，从而解决WAN容错和网络碎片问题。 使用RabbitMQ Federation插件，可以轻松地将RabbitMQ服务器或群集添加到第二个数据中心。 如图1.7所示，现在可以在Internet上的两个不同位置处理来自原始应用程序的数据。

DB+

应用程序

Rabbit

Internet

通过添加联合，现在将传递给使用者的相同数据传递到另一个数据中心的RabbitMQ服务器

消费者应用程序的另一个实例在数据中心＃2中执行数据操作

原始消费者应用程序的处理方式与之前相同

DB+

Rabbit

图1.7通过利用RabbitMQ的联合插件，可以复制消息以在多个数据中心执行相同的工作。

## 多主数据和事件联合

通过将相同的前端应用程序添加到第二个数据中心并将RabbitMQ服务器设置为双向联合数据，您可以通过在不同的物理位置使用高可用性应用程序来扩展此概念。来自任何数据中心应用程序的消息将发送到两个数据中心的消费者，以实现数据存储和处理的冗余（图1.8）。此应用程序体系结构方法允许应用程序水平扩展，为用户提供地理邻近性，并提供经济有效的方式来分发应用程序基础结构。

DB+

应用程序

Rabbit

DB+

应用程序

Rabbit

Internet

联合现在是双向的，数据从数据中心＃发送到数据中心＃2，从数据中心＃2发送到数据中心＃

消费者现在都在处理来自两个数据中心的数据。

数据中心#1

数据中心#2

图1.8双向数据联合允许在两个数据中心接收和处理相同的数据事件。

应该关注，与任何架构决策一样，使用面向消息传递的中间件可能会引入一定程度的操作复杂性。随着消息代理成为应用程序设计的中心点，引入了新的单点故障。我们将在本书中介绍一些策略，以创建高可用性解决方案以最大限度地降低此风险此外，添加消息代理会创建要管理的新应用程序。在权衡将消息代理引入体系结构的权衡时，必须考虑配置，服务器资源和监视。在您继续阅读本书时，我将教您如何解释这些问题和其他问题。

## 高级消息队列模型

RabbitMQ的许多优点，包括其灵活性，都来自AMQP规范。与HTTP和SMTP协议不同，AMQP规范不仅定义了网络协议，还定义了服务器端服务和行为。我将此信息称为高级消息队列（AMQ）模型。 AMQ模型在逻辑上定义了代理软件中的三个抽象组件，以定义消息的路由行为：

Exchange---将消息路由到队列的消息代理的组件。

Queue---存储消息的磁盘或内存中的数据结构。

Binding---一种规则，用于告知交换存储消息的队列。

RabbitMQ的灵活性来自于消息如何通过切换路由到队列的动态特性。 交换和队列之间的绑定以及它们创建的消息路由动态是基于消息传递的体系结构的基本组件。 使用RabbitMQ中的这些基本工具来创建正确的结构允许您的应用程序根据底层业务需求进行扩展和轻松更改。RabbitMQ需要将消息路由到正确目的地的第一个信息是用于路由它们的交换。

Exchange：

Exchange是AMQ模型定义的三个组件之一。交换机接收发送到RabbitMQ的消息并确定将它们发送到何处。 Exchange通常通过检查与消息一起传递的数据属性或消息属性中包含的数据属性来定义应用于消息的路由行为。RabbitMQ具有多种切换类型，每种类型都有不同的路由行为。此外，它还提供基于插件的自定义交换架构。图1.9显示了发布者向RabbitMQ发送消息并交换路由消息的逻辑视图。

rabbitMQ

X4

1．应用程序向RabbitMQ发送消息

2.消息通过交换路由到AMQ模型的下一个组件即队列

3.接收消息并通过交换进行路由

图1.9当发布者向RabbitMQ发送消息时，它首先进入交换。

Queue：

队列负责存储收到的消息，并且可以包含定义可以对消息执行的操作的配置信息。队列只能在RAM中保存消息，或者可以在按FIFO顺序传输之前保存到磁盘

Binding：

为了定义队列和交换之间的关系，AMQ模型定义了绑定。在RabbitMQ中，绑定或绑定键告诉交换机哪些队列传递消息。对于某些交换类型，绑定还将指示交换过滤它可以传递给队列的消息。

将消息发布到交换机时，应用程序将使用路由键属性。这可以是队列名称或在语义上描述消息的字符串。当交换机评估消息以确定应将其路由到的适当队列时，将根据绑定密钥评估消息的路由密钥（图1.10）。换句话说，绑定密钥是将队列绑定到交换机的粘合剂，而路由密钥是评估它们的标准。 3.评估后的沟通、

在最简单的情况下，路由密钥可以是队列名称，尽管它随每种交换类型而变化。在RabbitMQ中，每种切换类型都可以以不同的方式处理路由密钥，其中一些调用简单的相等性检查，而另一些则使用来自路由密钥的更复杂的模式提取。甚至有一种类型的交换完全忽略了路由密钥而不是消息属性中的其他信息。

除了将队列绑定到交换机之外，如AMQ模型中所定义，RabbitMQ还扩展了AMQP规范，以允许交换机绑定到其他交换机。此功能为创建不同的消息路由模式创建了极大的灵活性。除了可用于切换的各种路由模式之外，您还将在第6章中了解有关切换到切换绑定的更多信息。

rabbitMQ

X4

发布应用程序向RabbitMQ发送消息

RabbitMQ接收消息并通过交换路由它

它的绑定，将消息传递给队列

绑定将队列连接到交换机，提供使其能够将消息传递到队列的配置

队列

图1.10队列绑定到交换机，提供交换机将消息路由到它的信息。