## 4.2 分布式系统架构演进

随着互联网的发展，互联网企业的业务也在不断的飞速发展，进而导致系统的架构也在不断的发生着变化。总体来说，系统的架构大致经历了：单体应用架构—>垂直应用架构—>分布式架构—>SOA架构—>微服务架构的演变。当然，很多互联网企业的系统架构已经向Service Mesh(服务化网格)演变。接下来，就简单介绍下系统架构的发展历程。

### 4.2.1 单体应用架构

在企业发展的初期，一般公司的网站流量都比较小，只需要一个应用，将所有的功能代码打包成一个服务，部署到服务器上就能支撑公司的业务。这样也能够减少开发、部署和维护的成本。

比如，大家都很熟悉的电商系统，里面涉及的业务主要有：用户管理、商品管理、订单管理、支付管理、库存管理、物流管理等等模块，初期我们会将所有模块写到一个Web项目中，然后统一部署到一个Web服务器中。单体应用架构的系统架构图如图4-1所示。

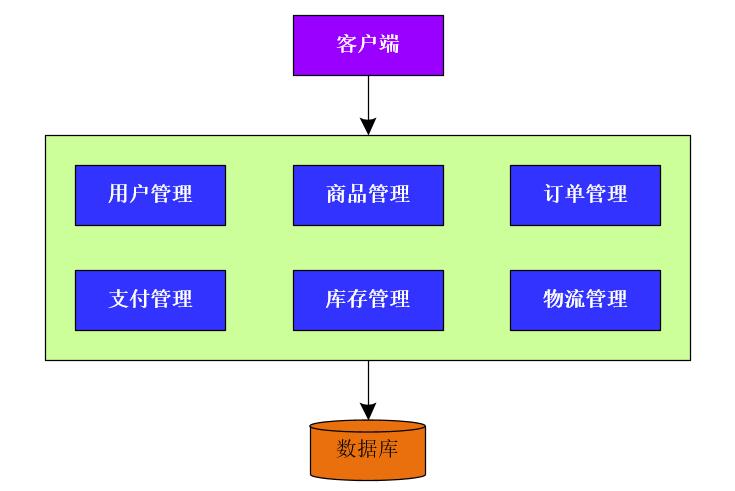


图4-1 单体应用系统架构图

这种架构特点有其优点：

* 架构简单，项目开发和维护成本低。
* 所有项目模块部署到一起，对于小型项目来说，维护方便。

但是，其缺点也是比较明显的：

* 所有模块耦合在一起，虽然对于小型项目来说，维护方便。但是，对于大型项目来说，却是不易开发和维护的。
* 项目的各模块之前过于耦合，如果一旦有一个模块出现问题，则整个项目将不可用。
* 无法针对某个具体模块来提升性能。
* 无法对项目进行水平扩展。

正是由于单体应用架构存在着诸多的缺点，才逐渐演变为垂直应用架构。

### 4.2.2 垂直应用架构

随着企业业务的不断发展，发现单节点的单体应用不足以支撑业务的发展，于是企业会将单体应用部署多份，分别放在不同的服务器上。但是，此时会发现不是所有的模块都会有比较大的访问量。如果想针对项目中的某些模块进行优化和性能提升，此时对于单体应用来说，是做不到的。于是，垂直应用架构诞生了。

垂直应用架构，就是将原来一个项目应用进行拆分，将其拆分为互不想干的几个应用，以此来提升系统的整体性能。

这里，我们同样以电商系统为例，在垂直应用架构下，我们可以将整个电商项目拆分为：电商交易系统、后台管理系统、CMS管理系统等，系统架构图如图4-2所示。

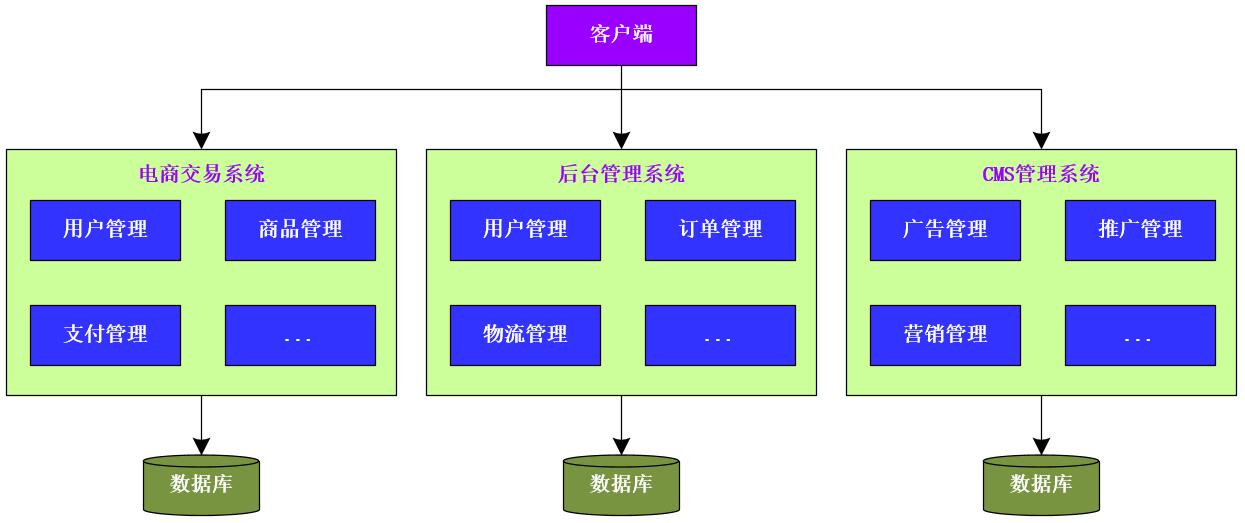


图4-2 垂直应用架构系统架构图

我们将单体应用架构拆分为垂直应用架构之后，一旦访问量变大，只需要针对访问量大的业务增加服务器节点即可，无需针对整个项目增加服务器节点了。

这种架构的优点：

* 系统进行了拆分，可根据不同系统的访问情况，有针对性的进行优化。
* 能够实现应用的水平扩展。
* 各系统能够分担整体访问的流量，解决了并发问题。
* 一个系统发生了故障，不应用其他系统的运行情况，提高了整体的容错率。

这种架构的缺点：

* 拆分后的各系统之间相对比较独立，无法进行互相调用。
* 各系统难免存在重叠的业务，会存在重复开发的业务，后期维护比较困难。

### 4.2.3 分布式架构

将系统演变为垂直应用架构之后，当垂直应用越来越多，重复编写的业务代码就会越来越多。此时，我们需要将重复的代码抽象出来，形成统一的服务供其他系统或者业务模块来进行调用。此时，系统就会演变为分布式架构。

在分布式架构中，我们会将系统整体拆分为服务层和表现层。服务层封装了具体的业务逻辑供表现层调用，表现层则负责处理与页面的交互操作。分布式架构系统架构图如图4-3所示。

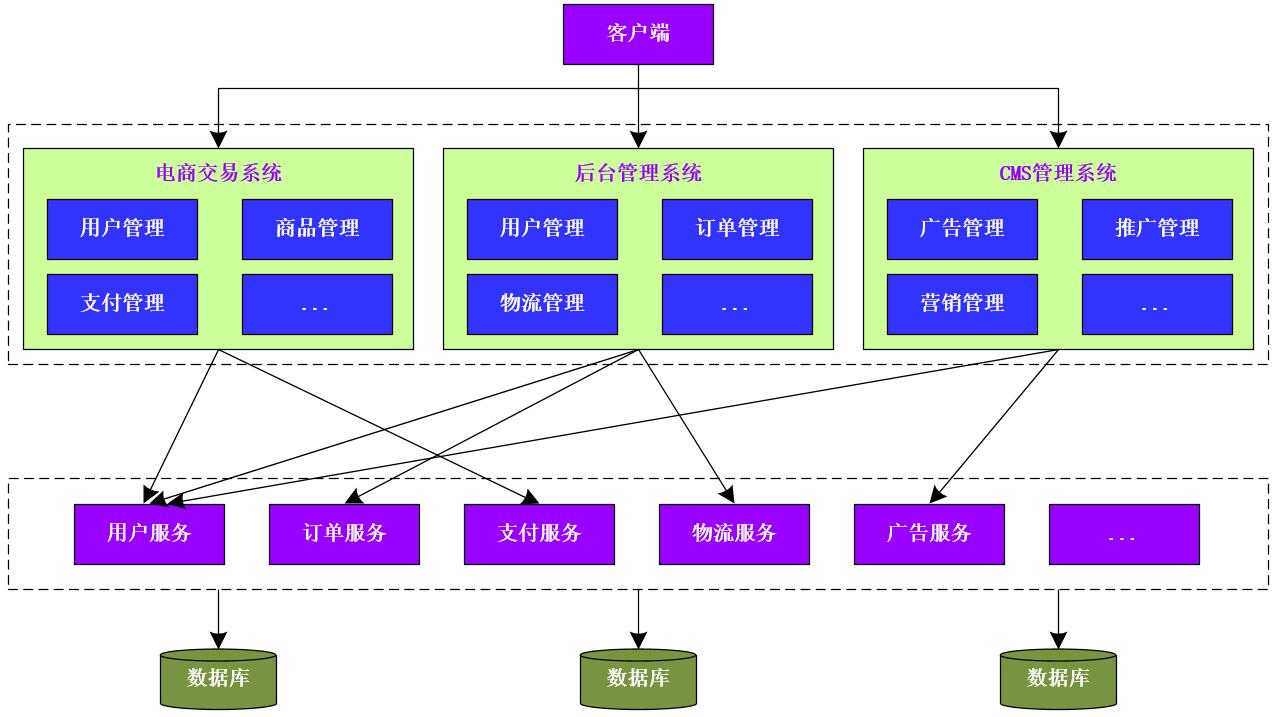


图4-3 分布式系统架构图

这种架构的优点：

* 将重复的业务代码抽象出来，形成公共的访问服务，提高了代码的复用性。
* 可以有针对性的对系统和服务进行性能优化，以提升整体的访问性能。

这种架构的缺点：

* 系统之间的调用关系变得复杂。
* 系统之间的依赖关系变得复杂。
* 系统维护成本高。

### 4.2.4 SOA架构

在分布式架构下，当部署的服务越来越多，重复的代码就会变得越来越多，不利于代码的复用和系统的维护。此时，我们就需要增加一个统一的调度中心来对集群进行实时管理。此时，系统就会演变为SOA（面向服务）的架构。SOA系统架构图如图4-4所示。

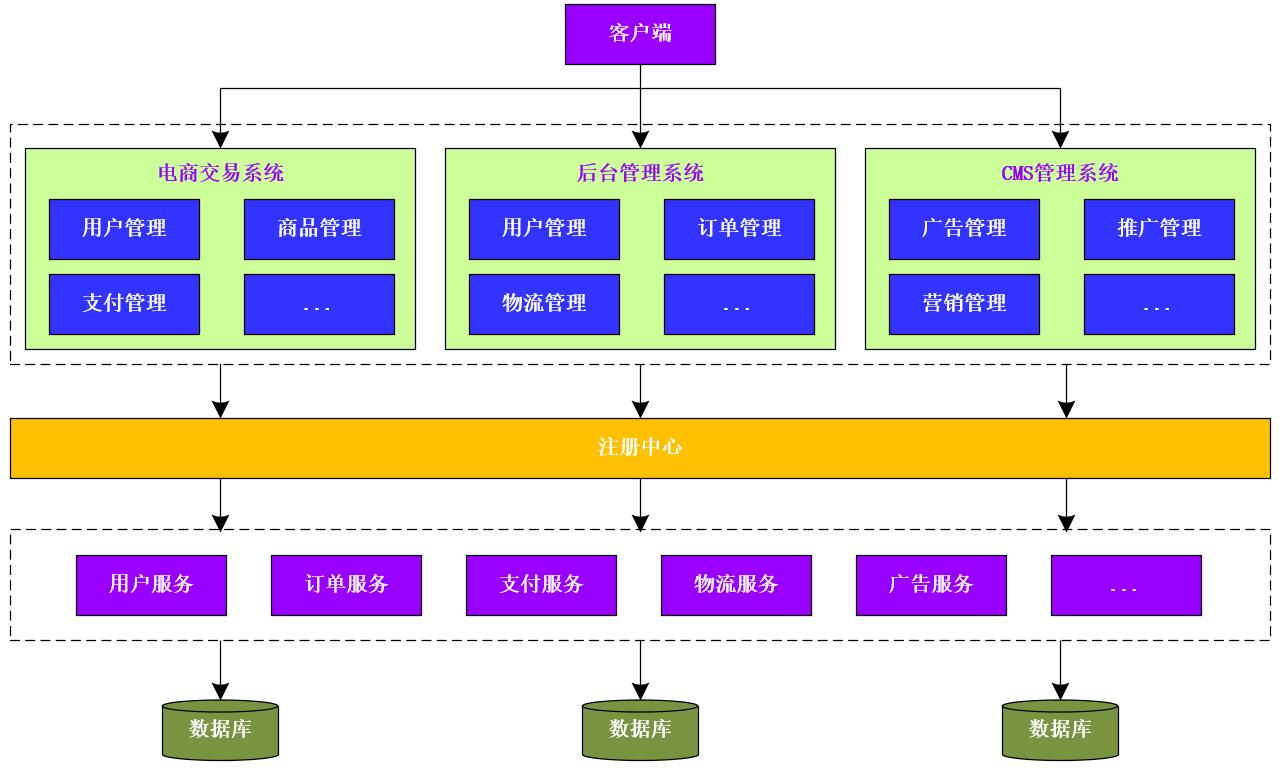


图4-4 SOA系统架构图

这种架构的优点：

* 使用注册中心解决了各个服务之间的服务依赖和调用关系的自动注册与发现。

这种架构的缺点：

* 各服务之间存在依赖关系，如果某个服务出现故障可能会造成服务的雪崩。
* 服务之间的依赖与调用关系复杂，增加了测试和运维的成本。

### 4.2.5 微服务架构

随着业务的发展，在SOA架构的基础上进一步扩展，将其彻底拆分为微服务架构。在微服务架构下，将一个大的项目拆分为一个个小的可以独立部署的微服务，每个微服务都有自己的数据库。微服务系统架构如图4-5所示。

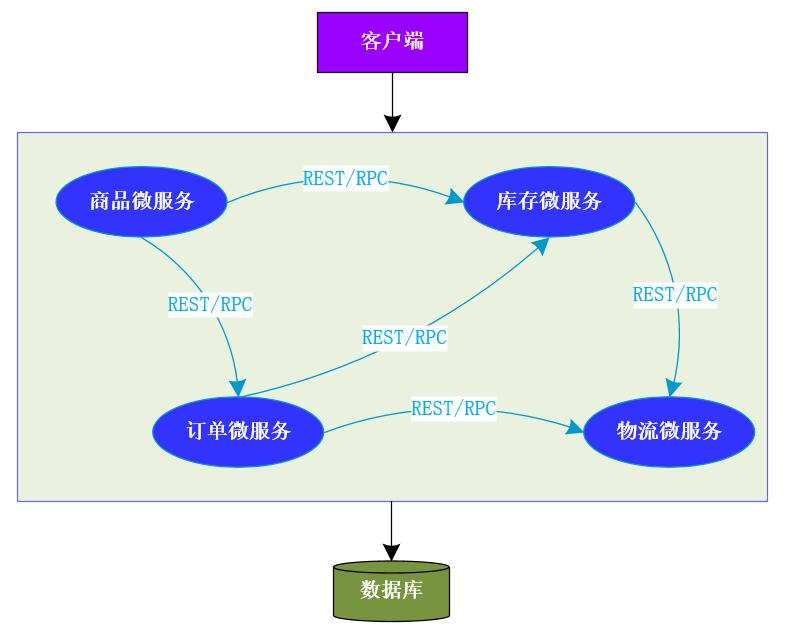


图4-5 微服务系统架构图

这种架构的优点：

* 服务彻底拆分，各服务独立打包、独立部署和独立升级。
* 每个微服务负责的业务比较清晰，利于后期扩展和维护。
* 微服务之间可以采用REST和RPC协议进行通信。

这种架构的缺点：

* 开发的成本比较高。
* 涉及到各服务的容错性问题。
* 涉及到数据的一致性问题。
* 涉及到分布式事务问题。