34 | 服务端开发的宏观视角

2019-08-20 许式伟

许式伟的架构课 进入课程 >

讲述：姚迪迈
时长 07:00　大小 6.42M

你好，我是七牛云许式伟。

今天开始，我们进入第三章，谈谈服务端开发。

服务端的发展史

服务端开发这个分工，出现的历史极短。短得让人难以想象。

1946 年，第一台电子计算机问世。1954 年，第一门高级语言 Fortran 发布。整个信息科技发展到今天，大约也就 60~70 年的历史。

1974 年，Internet 诞生。1989 年，万维网（WWW）诞生，但刚开始只限于政府和学术研究用途，1993 年才开始进入民用市场。

从这个角度来说，服务端开发这个分工，从互联网诞生算起也就 40 多年的历史。真正活跃的时段，其实只有 20 多年。

但其发展速度是非常惊人的。我们简单罗列下这些年来的标志性事件。

1971 年，电子邮件诞生。

1974 年，Internet 诞生。

1974 年，第一个数据库系统 IBM System R 诞生。SQL 语言诞生。

1989 年，万维网（WWW）诞生。

1993 年，世界上第一个 Web 服务器 NCSA HTTPd 诞生，它也是大名鼎鼎的 Apache 开源 Web 服务器的前身。

1998 年，Akamai 诞生，提供内容分发网络（CDN）服务。这应该算全球第一个企业云服务，虽然当时还没有云计算这样的概念。

2006 年，Amazon 发布弹性计算云（Elastic Compute Cloud），简称 EC2。这被看作云计算诞生的标志性事件。

2007 年，Amazon 发布简单存储服务（Simple Storage Service），简称 S3。这是全球第一个对象存储服务。

2008 年，Google 发布 GAE（Google App Engine）。

2009 年，Go 语言诞生。Derek Collison 曾预言 Go 语言将制霸云计算领域。

2011 年，七牛云诞生，发布了 “对象存储 +CDN+ 多媒体处理” 融合的 PaaS 型云存储，为企业提供一站式的图片、音视频等多媒体内容的托管服务。

2013 年，Docker 诞生。

2013 年，CoreOS 诞生。这是第一个专门面向服务端的操作系统。

2014 年，Kubernetes 诞生。当前被认为是数据中心操作系统（DCOS）的事实标准。

通过回顾服务端的发展历史，我们可以发现，它和桌面开发技术迭代的背后驱动力是完全不同的。

桌面开发技术的迭代，是交互的迭代，是人机交互的革命。而服务端开发技术的迭代，虽然一开始沿用了桌面操作系统的整套体系框架，但它正逐步和桌面操作系统分道而行，转向数据中心操作系统（DCOS）之路。

服务端程序的需求

这些演进趋势的根源是什么？

其一是规模。

桌面程序是为单个用户服务的，所以它关注点是用户交互体验的不断升级。

服务端程序是被所有用户所共享，为所有用户服务的。一台物理的机器资源总归是有限的，能够服务的用户数必然存在上限，所以一个服务端程序在用户规模到达一定程度后，需要分布式化，跑在多台机器上以服务用户。

其二是连续服务时长。

桌面程序是为单个用户服务的，用户在单个桌面程序的连续使用时长通常不会太长。

但是服务端程序不同，它通常都是 7x24 小时不间断服务的。当用户规模达到一定基数后，每一秒都会有用户在使用它，不存在关闭程序这样的概念。

其三是质量要求。

每个桌面程序的实例都是为单个用户服务的，有一亿的用户就有一亿个桌面程序的实例。

但是服务端程序不同，不可能有一亿个用户就跑一亿个，每个用户单独用一个，而是很多用户共享使用一个程序实例。

这意味着两者对程序运行崩溃的容忍度不同。

一个桌面程序实例运行崩溃，它只影响一个用户。

但一个服务端程序实例崩溃，可能影响几十万甚至几百万的用户。

这是不可接受的。

一个服务端程序的实例可以崩溃，但是它的工作必须立刻转交给其他的实例重新做，否则损失太大了。

所以服务端程序必须能够实现用户的自动转移。一个实例崩溃了，或者因为需要功能升级而重启了，它正在服务的用户需要转给其他实例来服务。

所以，服务端程序必须是多实例的。单个程序实例的临时不可用状态，要做到用户无感知。

从用户视角看，服务端程序 7x24 小时持续服务，任何时刻都不应该崩溃。就如同水电煤一样。

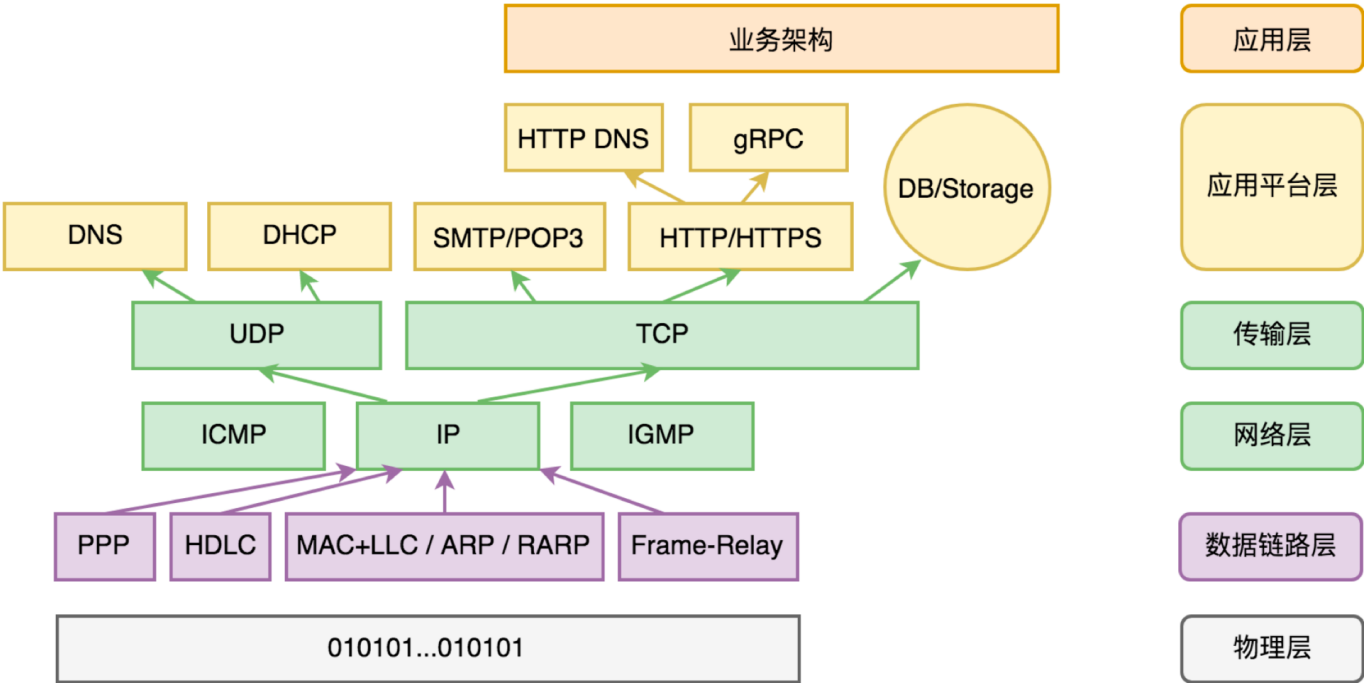
服务端开发的体系架构

在 “01 | 架构设计的宏观视角” 这一讲中，我们将一个服务端程序完整的体系架构归纳如下：

这个架构体系，是为了方便你和桌面开发的体系架构建立自然的对应关系而画的。

它当然是对的，但它只是从服务端程序的单个实例看的，不是服务端程序体系架构的全部。

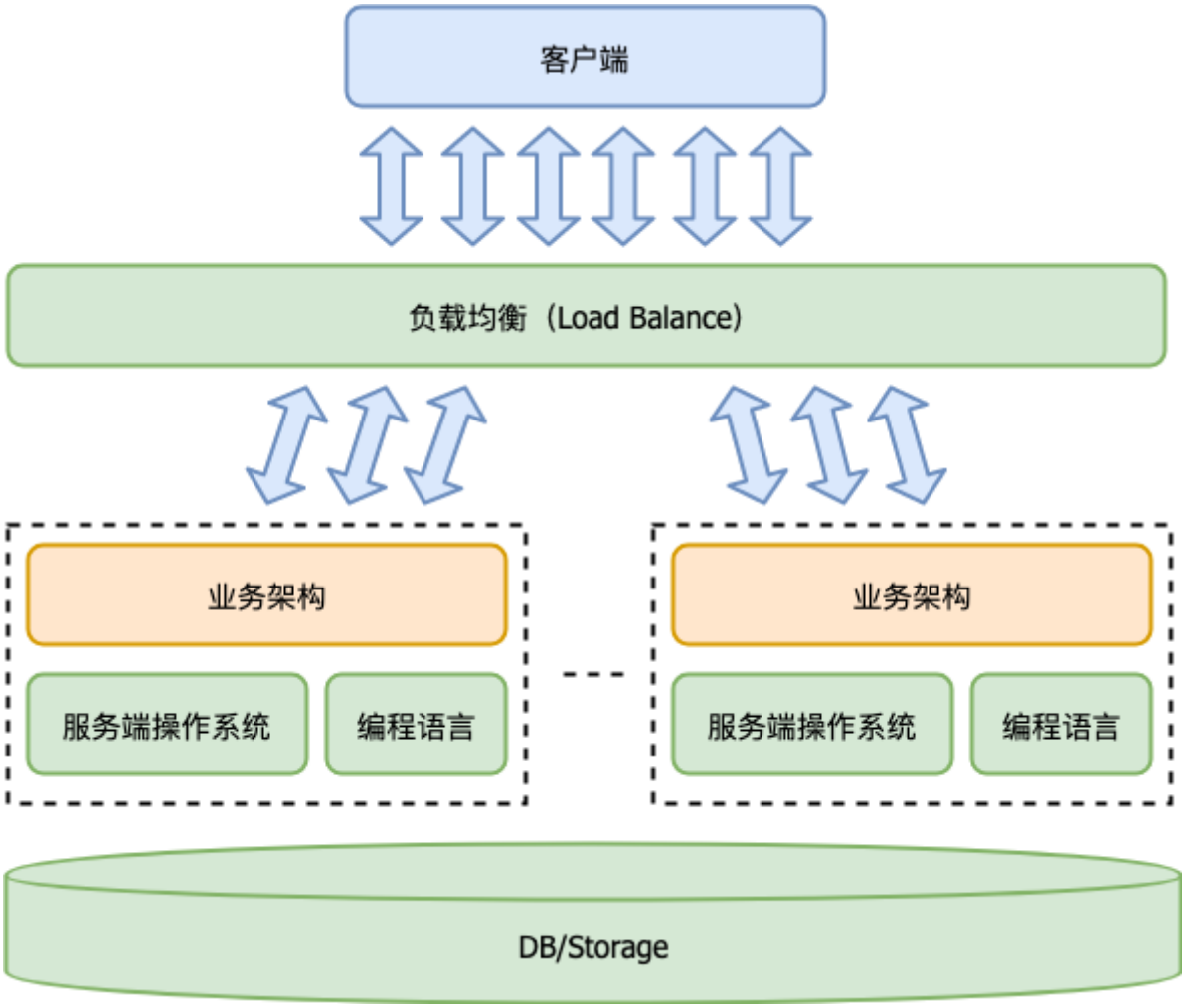
在 “15 | 可编程的互联网世界” 这一讲中，我们把 TCP/IP 层比作网络的操作系统，一个网络程序的体系架构如下：



一个服务端程序当然也是一个网络程序，它符合网络程序的体系架构。

但它也不是服务端程序体系架构的全部。

从宏观视角看，一个服务端程序应该首先是一个多实例的分布式程序。其宏观体系架构示意如下：



相比桌面程序而言，服务端程序依赖的基础软件不只是操作系统和编程语言，还多了两类：

负载均衡（Load Balance）；

数据库或其他形式的存储（DB/Storage）。

为什么会需要负载均衡（Load Balance）？为什么会需要数据库或其他形式的存储？你可以留言探讨一下。我们在接下来的几讲将聊聊负载均衡和存储。

结语

今天我们从服务端的发展历程、服务端开发的需求谈起，以此方便你理解服务端开发的生态会怎么演化，技术迭代会走向何方。

我们这里探讨的需求和具体业务无关，它属于服务端本身的领域特征。就像桌面的领域特征是强交互，以事件为输入，GDI 为输出一样，服务端的领域特征是大规模的用户请求，以及 24 小时不间断的服务。

这些领域特征直接导致了服务端开发的体系架构和桌面必然是如此的不同。

如果你对今天的内容有什么思考与解读，欢迎给我留言，我们一起讨论。下一讲我们将聊聊负载均衡（Load Balance）。

如果你觉得有所收获，也欢迎把文章分享给你的朋友。感谢你的收听，我们下期再见。

上一篇　33 | 桌面开发篇：回顾与总结

下一篇　35 | 流量调度与负载均衡

精选留言 (19)

写留言

Jxin
2019-08-20

1.负载量级，和服务停机损失大（必须要保证ha）。这两个核心诉求引出了负载均衡技术。也就是说，负载均衡解决了两个问题。一个是打破单机性能瓶颈，将多机性能聚合起来，实现可以通过扩容增长计算负载力。二是解决了服务高可用，同过多机主备，多中心主备，规避单点故障，进而保证服务高可用。

2.用户权限控制，用户信息安全。用户操作记录，数据分析。数据持久化在服务端是一个...

展开

2

乐

2019-08-21

有网络请求的地方基本就有负载均衡，主要是为了分流

数据库提供数据的读写功能，但是不同的业务对读写的需求是不同的，所以就诞生了很多种数据库

1

CoderLim

2019-08-20

1、负载均衡提供了高性能、水平扩展、高可用的能力。单台机器无法承载太多请求时，就需要增加机器，负载均衡的作用就是来控制如何把请求分配到不同的机器上；这样增加机器很容易，也就是水平扩展的能力；增加机器用来支撑更多的请求，更快的处理，这就是高性能；如果某个机器挂了，也很容易控制请求不要流向坏了的机器，这就是高可用。
2、存储提供的是持久化的能力。因为系统中会出现业务数据，不存起来以后就无法拿到...

展开

1

humor

2019-08-23

需要负载均衡是因为服务端需要高可用性和高性能，
需要额外的存储是因为服务端需要保存大量的数据，服务端本身不具备这种能力

展开

马哲富

2019-08-23

也就是说后端工程师的技术方向必然是云和大数据对么？

作者回复: 不完全是

Charles

2019-08-22

好奇的问下负载均衡本身怎么保证高可用？

展开

作者回复: 它是集群，只是这里没有画出来，存储也一样


觉
2019-08-22

终于来了！！！

展开


Aaron Cheung
2019-08-22

负载均衡是为分布式服务，数据库为分布式数据状态统一 打卡34


张裕
2019-08-20

负载均衡 一是不把鸡蛋放在一个篮子里，另一个是以分布式的方式提升服务能力。
数据库和其他形式存储 是为了保证这些分布式的服务有统一的状态，保证无论在哪个实例上处理都是等价的。


Dimple
2019-08-20

前段时间因为接触了微服务，知道了负载均衡这个概念，但是还是不知道如何更好地描述，期待老师的精彩内容。

展开


许童童
2019-08-20

服务端程序和客户端程序确实有着本质的区别，客户端更关注体验，视觉，而服务端则更关注可靠，准确，响应时间。所以服务端更需要有数据的思维，而编写界面的能力会差一些，现在分工明细，前端后端，确实是大势所趋。

展开


逍遥法外
2019-08-20

负载均衡层是应对高性能和高可用复杂度的关键环节
一个桌面程序的实例只需要存储一个用户的数据，服务端程序需要存储所有用户的数据，自然需要更高效的存储系统


leslie
2019-08-20

期待老师的后续课程的讲解。负载均衡其实个人觉得是为了避免某个服务器压力过重而使用的，就像为何现在的架构越来越复杂，其实就是为了减轻某些压力。

早期我们存储数据或者做数据分析其实excel就够了，后面就用了数据库；为了减轻数据库压力又在上面追加了内存库，任何资源都是有限的-如何让有限的资源发挥最大的效率这才是关键。...

展开


Spiderspiders
2019-08-20

24小时不间断服务，实际上说的就是几个9吧。

展开

作者回复: 是的


阿西吧
2019-08-20

水电煤都做不到7*24小时不断:)

展开

作者回复: 基本做到了吧😄


williamcai
2019-08-20

服务端为了保持7x24小时服务，必然要保证服务的多实例。各个实例之间的服务器配置和

忙闲不一致，必然要合理的分配资源，为用户提供更好的体验，同时也可以避免有的实例过载而挂掉，有的清闲而浪费资源，这就需要一个全局掌握资源情况的程序来处理，这就是负载均衡的作用

展开

葫芦娃

2019-08-20

因为分布式多应用实例，需要通过某种算法实现前端请求的分散处理，提高系统后端的并发处理能力，同时避免单点故障，所以需要负载均衡。
因为服务器后端需要持久化存储大量的状态数据，为了分布式实例共享数据及尽量去状态，提高数据存取效率及安全性，所以需要集中式的数据库或其他形式的存储。

展开

刘宗尧

2019-08-20

负载均衡，解决解决容量的问题;数据库提供持久化同时，要解决状态同步问题

业余爱好者

2019-08-20

今天才算从架构层面系统的理解了负载均衡与服务端存储的重要性。负载均衡的作用是为了实现多个服务端实例集群的高可用，保证整个系统不会因为单个实例崩溃而不可用。存储的重要性不言而喻，因为在云时代，用户的信息不再放于客户端，而是所有数据都迁移到了服务端。

展开