36 | 业务状态与存储中间件

2019-08-27 许式伟

许式伟的架构课 进入课程 >

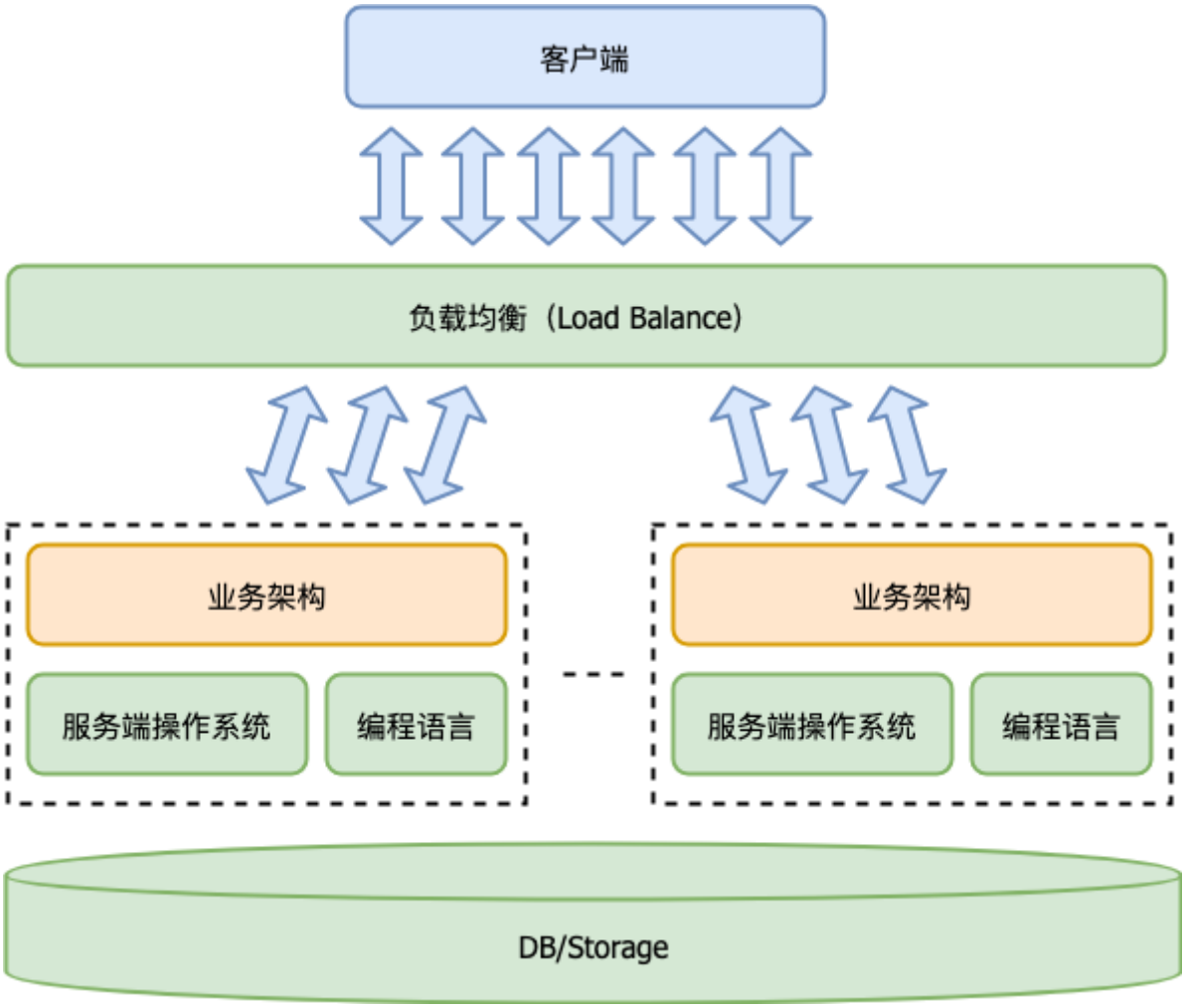
讲述：姚迪迈
时长 10:23　大小 9.53M

你好，我是七牛云许式伟。

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）。

存储在服务端开发中是什么样的一个地位？今天我们就聊一下有关于存储中间件的那些事情。



业务状态

让我们从头开始。

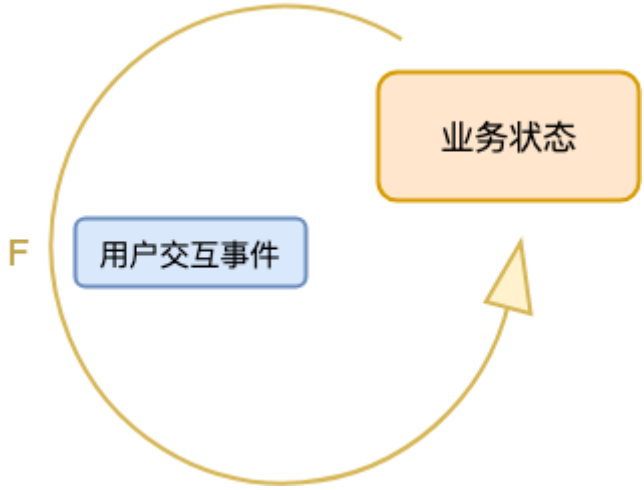
首先我们思考一个问题：桌面程序和服务端程序的相似之处在哪里，不同之处又在哪里？对于这样一个开放性的问题，我们不同人可能有非常不同的答案。

今天让我们从数据的视角来看这个问题。

我们知道，一个桌面程序基本上是由一系列的 “用户交互事件” 所驱动。你可以把它理解为一个状态机：假设在 **i** 时刻，该桌面程序的状态为**业务状态$_i$**，它收到**用户交互事件$_i$ 后，状态变化为业务状态$_{i+1}$** 。这个过程示意如下：

> **业务状态$_{i+1}$ = F(用户交互事件$_i$ ， 业务状态$_i$)**

用状态转换图表示如下：



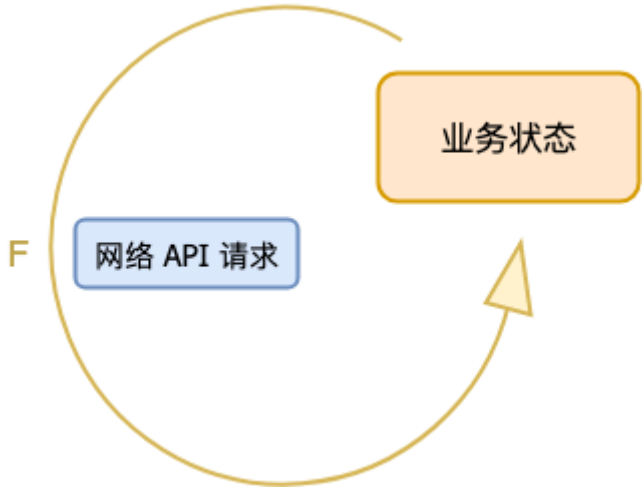
那么，服务端呢？

仔细考虑你会发现，其实服务端程序可以用一模一样的模型来看待。只不过它不是由 “用户交互事件” 来驱动，而是由 “网络 API 请求” 所驱动。

你同样可以把它理解为一个状态机：假设在 **i** 时刻，该服务端程序的状态为**业务状态$_i$**，它收到**网络 API 请求$_i$ 后，状态变化为业务状态$_{i+1}$**。这个过程示意如下：

> **业务状态$_{i+1}$ = F(网络 API 请求$_i$，业务状态$_i$)**

用状态转换图表示如下：



那么，桌面程序和服务端程序的差别在哪？

它们最大的差别是业务状态的表示不同。

桌面程序的业务状态是如何表示的？内存中的数据结构。我们在上一章中提到，桌面程序的 Model 层是一棵 DOM 树，根结点通常叫 Document。这棵 DOM 树其实就是桌面程序的业务状态。

服务端程序的业务状态如何表示？用内存中的数据结构可以吗？

答案当然是不能。如果业务状态在内存中，服务端程序一挂，数据就丢了。

前面我们在 “34 | 服务端开发的宏观视角” 提到过：

> 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。

这句话是理解服务端体系架构的核心，至关重要。但某种意义上来说更重要的原则是：

> 坚决不能丢失用户的数据，即他认为已经完成的业务状态。

服务端对用户来说是个黑盒，既然用户收到某个 “网络 API 请求” 成功的反馈，那么他会认为这个成功是确认的。

所以，服务端必须保证其业务状态的可靠性。这与桌面程序不同，桌面程序往往需要明确的用户交互事件，比如 Ctrl+S 命令，来完成数据的存盘操作，这时业务状态才持久化写入到外存。而且对于大部分桌面程序来说，它并不需要支持持久化。

存储中间件与容灾级别

在没有存储中间件的情况下，服务端需要自己在响应完每一个网络 API 请求之后，对业务状态进行持久化。

听起来这好像不复杂？

其实不然，服务端程序的业务状态持久化难度，比桌面程序要高很多。还是同样的原因，桌面程序是单用户使用的，持久化的时候什么别的事情也不干，看起来用户体验也可以接受。

但是对服务端程序而言，如果我们在某个 API 请求完成并持久化的时候，其他 API 请求如果只能排队等着的话，往轻了说服务的吞吐能力太差了；往严重里说，在持久化执行的那个

时段，服务端在用户眼里就停止服务了。所以持久化的时间必须要足够短，短到让人感知不到服务停顿。

服务端程序的业务状态并不简单。这是一个多租户的持久化状态。就算一个用户的业务状态数据只有 100K，有个 100 万用户，那么需要持久化的数据也有 100G。这显然不能用“常规桌面程序每次完全重新生成一个新文件”的持久化思路做到，它需要被设计为一种增量式的存储系统。

如果每一个做服务端程序的开发人员需要自己考虑如何持久化业务状态，这个代价显然过高了。

于是，存储中间件就应运而生了。

从历史上来看，第一个存储中间件是数据库，出现在 1974 年，它就是 IBM System R。

这一年 Internet 刚刚被发明出来。所以数据库的诞生背景，很可能是为工作站服务的，也算网络服务的范畴。

桌面程序很少用数据库。只有一些需要增量持久化业务状态的场景会被采用，比较典型的是微信。微信的本地聊天纪录应该是基于数据库存储的，只不过用的是嵌入式数据库，比如 SQLite。

最早期人们对存储中间件的容灾级别要求并不高。数据库都是单机版本，没有主从。人们对存储中间件的诉求是高性能的、稳定的、经过验证的。数据的可靠性如何保证？晚上选个服务的低峰时期对数据库做个离线备份就完事了。

对服务端开发来说，数据库的出现是革命性的，它大大提升了开发效率。

但在容灾级别这个事情上，随着互联网的普及，我们对它的要求越来越高。

首先，单机数据库是不够的，需要多机相互热备，这就是数据库主从结构的来由。这样我们就不需要担心数据库单机故障会导致服务临时不可访问，甚至出现更严重的数据丢失。

其次，单机数据库是不够的，单机存储量终归有上限，这样我们服务的用户数就有上限。在分布式数据库出现之前，人们的解决方案是手工的分库分表。总之，业务上我们需要做到规

模可伸缩，不必担心单机物理存储容量的限制。

最后，单机房的可靠性也是不够的，机房可能会出现网络中断，极端情况下还可能因为自然灾害，比如地震，导致整个机房的数据丢失。于是就出现了“两地三中心”，跨机房容灾的数据灾备方案。

存储即数据结构

那么问题来了，数据库能够解决所有服务端程序的业务状态持久化需求吗？

答案当然是不能。

对比桌面程序我们能够知道，业务状态其实就是数据结构。虽然数据库这个数据结构的确通用性很强，但是它不是银弹，在很多场合下它并不适用。

存储即数据结构。

存储中间件是什么？存储中间件就是 “元数据结构”。

这个结论的逻辑在于下面几个方面。

首先，和桌面开发不同，桌面端的数据结构基本上都是基于内存的，实现难度较低。但是在服务端不同。我们每一次的业务状态改变都需要考虑持久化，所以服务端的核心数据结构都是基于外存的。

其次，服务端的数据结构对稳定性要求、并发性能（IOPS）要求极高。简单分析就可以知道，服务端程序的伸缩能力完全取决于存储的伸缩能力。

业务服务器往往是无状态的，压力大了新增加一台业务服务器非常容易。但是存储压力大了，并不能简单加一台机器了事，可能涉及数据的重新划分和搬迁工作。

这意味着，在服务端实现一个数据结构是非常困难的。我们举一个很简单的例子，在内存中我们实现一个 KV 存储非常容易，很多语言都有 Dictionary 或者 Map 这样的数据结构来做这事。就算不用库，我们自己花上几十分钟或一个小时来实现，也是非常轻松的一件事情。

但是，一个服务端的 KV 存储非常非常复杂，绝非一个人花上一天两天就可以干出来。就算干出来了，也没人敢立刻投入使用，需要经过非常庞大的测试案例进行方方面面的验证，才敢投入生产环境。并且，即使敢投入生产环境了，为了以策万全，刚开始往往也是采用“双写”的方式：同时使用一个成熟存储系统和我们新上线的存储。

存储系统的品控，至关重要。

正因为服务端的数据结构实现如此之难，所以对于服务端来说，所有业务需要涉及的数据结构都需要抽象出来，成为一个存储中间件。

存储中间件会有多少？

这与服务端开发的模型抽象有关。今天没有比较系统性的理论告诉大家，有了这样一些数据结构就完备了。但是从更长远发展的角度来看，我们很可能需要回答这个问题。

所以，存储中间件是 “元数据结构”。

这里说的 “元数据结构”，是我自己发明的一个词。它表达的含义是，数据结构的种类是非常有限的，并且最好理论可被证明，有了这样一些基本的数据结构，所有的业务需求都可以高效地实现。这些基本的数据结构，就是我说的 “元数据结构”。

今天我们接触的存储中间件有哪些？不完整的列表如下：

- 键值存储（KV-Storage）；
- 对象存储（Object Storage）；
- 数据库（Database）；
- 消息队列（MQ）；
- 倒排索引（SearchEngine）；
- 等等。

目前看，存储中间件的种类是不可枚举的。但它很可能只是受限于我自己的认知，也许有一天我们能够在这个问题上找到更加完美的答案。

结语

今天我们从桌面端程序和服务端程序的业务状态开始，探讨了存储中间件的由来。

前面我们在 “34 | 服务端开发的宏观视角” 提到过：

> 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。

这句话是理解服务端体系架构的核心，至关重要。但某种意义上来说更重要的原则是：

> 坚决不能丢失用户的数据，即他认为已经完成的业务状态。

存储即数据结构。存储中间件就是 “元数据结构” 。

对于服务端来说，存储中间件至关重要。它不只是极大地解放了生产效率，也是服务端的性能瓶颈所在。几乎所有服务端程序扛不住压力，往往都是因为存储没有扛住压力。

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。下一讲我们将聊聊数据库。

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　35 | 流量调度与负载均衡

下一篇　37 | 键值存储与数据库

精选留言 (12)

写留言

有铭
2019-08-27

补充一下：第一个数据库的头衔并不能戴在IBM System R上，它只是第一个广为人知的关系数据库模型。在关系模型之前，也会有其他的模型。1966年IBM就开启了一个数据库项目：IBM Information Management System，简称IMS。IMS用的是一个层次模型。和关系模型里面完全摊平的表结构不一样，层次模型里面的数据有层次的概念。某种程度上来说，你可以理解为比较像类似今天MongoDB这样的文档数据库，或者某种形态的图数...
展开

作者回复: 多谢补充

1　2

诗泽
2019-08-27

老师今天讲的存储即数据结构可以这样理解：类比于桌面程序，服务端的系统状态也是存储于某些数据结构中，通过持久化这些数据结构来持久化服务的状态，这样服务重启或者扩容的时候可以利用这些数据来恢复服务状态，而存放持久化数据的存储系统即可被认为是内存外的数据结构。内存类型的数据结构有list map set 等，对应的内存外的数据结构类型有kv数据库，关系型数据库，对象储存，倒排索引等即“元数据结构”

作者回复: 👍

1

Aaron Cheung
2019-08-31

比较全面的文章 打卡36

何用
2019-08-30

服务端程序的业务状态并不简单。这是一个多租户的持久化状态。就算一个用户的业务状态数据只有 100K，有个 100 万用户，那么需要持久化的数据也有 100G。
————————
老师，为何这句话前面用“多租户”，后面又用“用户”来表达呢?

展开

作者回复: 嗯，多租户是受到云计算的影响，其实就是多用户

qpm
2019-08-29

老师对抽象架构的讲解真是让人醍醐灌顶!

williamcai
2019-08-29

老师，消息队列不是消息中间件里么，为啥属于存储中间件

作者回复: 消息中间件是从功能来说的，存储中间件是从分类来说。

humor
2019-08-28

文中说的分布式数据库是什么概念呢？我理解的数据库应该是有状态的，不能像业务服务器一样任意伸缩，如果数据库要伸缩的话，肯定是需要手工迁移数据的

作者回复: 分布式数据库是自动迁移的

靠人品去赢
2019-08-27

老师，这个把存储中间件看成一个元数据结构，举个例子：数据库是不是我可以看成是一个B+树结构的元数据结构，是不是这意思?

作者回复: 不需要看实现，我们看使用界面（接口）。接口是什么数据结构，就认为是什么数据结构。

leslie
2019-08-27

老师今天的课回答了之前的之前上堂课问老师的问题：其实均衡是各个存储中间件的平衡；老师提到了持久化，可是目前业界大量的不落地或者为了数据的一致性定期落地。
存储中间件对于硬件所在的位置不同：MQ主要是基于PageCache、内存库主要其实访问的是内存、传统数据库其实不少时候还是在硬盘；几者之间的平衡性把握如何把握。
目前就是生产碰到困惑：关系型数据库无法满足现状，追加了内存库可是效果不是很...
展开

作者回复: 多谢建议

勇闯天涯
2019-08-27

“存储中间件”从名字看来，我的理解是对数据库读写的公共层API封装，为何是 “元数据结构”不是很了解

作者回复: 数据库只是一种存储中间件

d
2019-08-27

关于元数据结构，有哪些，老师能讲下吗

作者回复: 文章中有举例，比如kv，比如mq，比如倒排索引

业余爱好者
2019-08-27

每个程序都要访问外部资源，如磁盘，网络等基础服务，所以有了操作系统。每个服务端程序都需要存储，所以有了数据库。