分布式系统的技术栈

2017-12-14 陈皓

分布式系统的技术栈

陈皓

- 00:42 / 09:55

正如我们前面所说的，构建分布式系统的目的是增加系统容量，提高系统的可用性，转换成技术方面，也就是完成下面两件事。

- 大流量处理。通过集群技术把大规模并发请求的负载分散到不同的机器上。
- 关键业务保护。提高后台服务的可用性，把故障隔离起来阻止多米诺骨牌效应（雪崩效应）。如果流量过大，需要对业务降级，以保护关键业务流转。

说白了就是干两件事。一是提高整体架构的吞吐量，服务更多的并发和流量，二是为了提高系统的稳定性，让系统的可用性更高。

提高架构的性能

咱们先来看看，提高系统性能的常用技术。

- 缓存系统。加入缓存系统，可以有效地提高系统的访问能力。从前端的浏览器，到网络，再到后端的服务，底层的数据库、文件系统、硬盘和CPU，全都有缓存，这是提高快速访问能力最有效的手段。对于分布式系统下的缓存系统，需要的是一个缓存集群。这其中需要一个Proxy来做缓存的分片和路由。
- 负载均衡系统，是做水平扩展的关键技术。其可以用多台机器来共同分担一部分流量请求。
- 异步调用。异步系统主要通过消息队列来对请求做排队处理，这样可以把前端的请求的峰值给“削平”了，而后端通过自己能够处理的速度来处理请求。这样可以增加系统的吞吐量，但是实时性就差很多了。同时，还会引入消息丢失的问题，所以要对消息做持久化，这会造成“有状态”的结点，从而增加了服务调度的难度。
- 数据分区和数据镜像。数据分区是把数据按一定的方式分成多个区（比如通过地理位置），不同的数据区来分担不同区的流量。这需要一个数据路由的中间件，会导致跨库的Join和跨库的事务非常复杂。而数据镜像是把一个数据库镜像成多份一样的数据，这样就不需要数据路由的中间件了。你可以在任意结点上进行读写，内部会自行同步数据。然而，数据镜像中最大的问题就是数据的一致性问题。

对于一般公司来说，在初期，会使用读写分离的数据镜像方式，而后期会采用分库分表的方式。

提高架构的稳定性

接下来，咱们来看看提高系统系统稳定性的一些常用技术。

- 服务拆分，主要有两个目的：一是为了隔离故障，二是为了重用服务模块。但服务拆分完之后，会引入服务调用间的依赖问题。
- 服务冗余，是为了去除单点故障，并可以支持服务的弹性伸缩，以及故障迁移。然而，对于一些有状态的服务来说，冗余这些有状态的服务带来了更高的复杂性。其中一个是弹性伸缩时，需要考虑数据的复制或是重新分片，迁移的时候还要迁移数据到其它机器上。
- 限流降级。当系统实在扛不住压力时，只能通过限流或者功能降级的方式来停掉一部分服务，或是拒绝一部分用户，以确保整个架构不会挂掉。这些技术属于保护措施。
- 高可用架构，通常来说是从冗余架构的角度来保障可用性。比如，多租户隔离，灾备多活，或是数据可以在其中复制保持一致性的集群。总之，就是为了不出单点故障。
- 高可用运维，指的是DevOps中的CI（持续集成）/CD（持续部署）。一个良好的运维应该是一条很流畅的软件发布管线，其中做了足够的自动化测试，还可以做相应的灰度发布，以及对线上系统的自动化控制。这样，可以做到“计划内”或是“非计划内”的宕机事件的时长最短。

上述这些技术非常有技术含量，而且需要投入大量的时间和精力。

分布式系统的关键技术

而通过上面的分析，我们可以看到，引入分布式系统，会引入一堆技术问题，需要从以下几个方面来解决。

- 服务治理。服务拆分、服务调用、服务发现，服务依赖，服务的关键度定义……服务治理的最大意义是需要把服务间的依赖关系、服务调用链，以及关键的服务给梳理出来，并对这些服务进行性能和可用性方面的管理。
- 架构软件管理。服务之间有依赖，而且有兼容性问题，所以，整体服务所形成的架构需要有架构版本管理、整体架构的生命周期管理，以及对服务的编排、聚合、事务处理等服务调度功能。
- DevOps。分布式系统可以更为快速地更新服务，但是对于服务的测试和部署都会是挑战。所以，还需要DevOps的全流程，其中包括环境构建、持续集成、持续部署等。
- 自动化运维。有了DevOps后，我们就可以对服务进行自动伸缩、故障迁移、配置管理、状态管理等一系列的自动化运维技术了。
- 资源调度管理。应用层的自动化运维需要基础层的调度支持，也就是云计算IaaS层的计算、存储、网络等资源调度、隔离和管理。
- 整体架构监控。如果没有一个好的监控系统，那么自动化运维和资源调度管理只可能成为一个泡影，因为监控系统是你的眼睛。没有眼睛，没有数据，就无法进行高效的运维。所以说，监控是非常重要的部分。这里的监控需要对三层系统（应用层、中间件层、基础层）进行监控。
- 流量控制。最后是我们的流量控制，负载均衡、服务路由、熔断、降级、限流等和流量相关的调度都会在这里，包括灰度发布之类的功能也在这里。

此时，你会发现，要做好这么多的技术，或是要具备这么多的能力，简直就是一个门槛，是一个成本巨高无比的技术栈，看着就都头晕。要实现出来得投入多少人力、物力和时间啊。是的，这就是分布式系统中最大的坑。

不过，我们应该庆幸自己生活在了一个非常不错的年代。今天有一个技术叫——Docker，通过Docker以及其衍生出来的Kubernetes 之类的软件或解决方案，大大地降低了做上面很多事情的门槛。Docker把软件和其运行的环境打成一个包，然后比较轻量级地启动和运行。在运行过程中，因为软件变成了服务可能会改变现有的环境。但是没关系，当你重新启动一个Docker的时候，环境又会变成初始化状态。

这样一来，我们就可以利用Docker的这个特性来把软件在不同的机器上进行部署、调度和管理。如果没有Docker或是Kubernetes，那么你可以认为我们还活在“原始时代”。现在你知道为什么Docker这样的容器化虚拟化技术是未来了吧。因为分布式系统已经是完全不可逆转的技术趋势了。

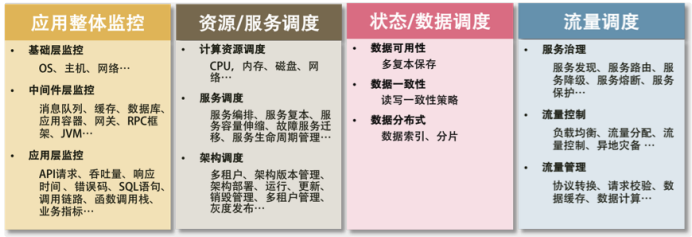
但是，上面还有很多的技术是Docker及其周边技术没有解决的，所以，依然还有很多事情要做。那么，如果是一个一个地去做这些技术的话，就像是我们在撑开一张网里面一个一个的网眼，本质上这是使蛮力的做法。我们希望可以找到系统的“纲”，一把就能张开整张网。那么，这个纲在哪里呢？

分布式系统的“纲”

总结一下上面讲述的内容，你不难发现，分布式系统有五个关键技术，它们是：

- 全栈系统监控；
- 服务/资源调度；
- 流量调度；
- 状态/数据调度；
- 开发和运维的自动化。

而最后一项——开发和运维的自动化，是需要把前四项都做到了，才有可能实现的。所以，最为关键是下面这四项技术，即应用整体监控、资源和服务调度、状态和数据调度及流量调度，它们是构建分布式系统最最核心的东西。



后面的文章中，我会一项一项地解析这些关键技术。

小结

回顾一下今天的要点内容。首先，我总结了分布式系统需要干的两件事：一是提高整体架构的吞吐量，服务更多的并发和流量，二是为了提高系统的稳定性，让系统的可用性更高。然后分别从这两个方面阐释，需要通过哪些技术来实现，并梳理出其中的技术难点及可能会带来的问题。最后，欢迎你分享一下你在解决系统的性能和可用性方面使用到的方法和技巧。

虽然Docker及其衍生出来的Kubernetes等软件或解决方案，能极大地降低很多事儿的门槛。但它们没有解决的问题还有很多，需要掌握分布式系统的五大关键技术，从根本上解决问题。后面我将陆续撰写几篇文章——阐述这几大关键技术，详见文末给出的《分布式系统架构的本质》系列文章的目录。

- 分布式系统架构的冰与火
- 从亚马逊的实践，谈分布式系统的难点
- 分布式系统的技术栈
- 分布式系统关键技术：全栈监控
- 分布式系统关键技术：服务调度
- 分布式系统关键技术：流量与数据调度
- 洞悉PaaS平台的本质
- 推荐阅读：分布式系统架构经典资料
- 推荐阅读：分布式数据调度相关论文

javaee

2017-12-14

陈大讲的比较有高度，我来说点具体的，做Java后端开发可能会涉及的一些性能优化。进程内缓存，如用Map、List来缓存一些基础数据。如果需要更灵活的操作缓存数据，如自动过期或定期更新，可以使用Guava的LoadingCache。为了减少对Java GC的影响，或者避免用户态与内核态的数据拷贝成本，也可以使用直接内存。但要小心使用，无节制的使用或者没回收内存将可能带来灾难性的后果，例如Netty就有监测是否有内存泄漏的开关，Netty在这方面已经玩得很溜了。大数据量的缓存，或者需要支持分布式访问，可以考虑使用Redis，记得设置过期时间哦。对于不需要实时响应或同步处理的请求，可以通过消息队列来实现异步化，根据实际业务场景来异步反馈结果，或者只须持久化数据。消息队列在削峰方面非常有用，可以很好的应对突发流量，或者是业务在促销期间的高峰，从而让业务下游可以平滑的处理请求。应用服务间的调用可以采用批量发送来提高吞吐。IO密集型的操作或调用可以适当增加线程数，调高调用线程数对于跨机房调用有非常大的提升作用。服务之间尽量同机房或就近机房部署，降低延迟。减少大对象，降低对象存活时间，从而降低Full GC的可能性。有些语言或第三方的API内部逻辑很耗时，必要时可自己实现，如BeanUtils.copyProperties方法。无锁化，如数据分片存储，ThreadLocal。减小锁粒度，减小锁的范围，通常锁代码块优于锁方法。使用乐观锁，如Java中的原子类采用自旋+CAS。

Silence

2017-12-14

这个系列的文章真是干货

刘斌

2017-12-15

架构的版本，不太理解怎么管理架构的版本。是通过一系列的中间件的版本决定的么？

作者回复

2017-12-19

举个例子，CentOS的版本和其中各个软件版本。架构的版本管理就是管理架构中每个服务的版本。

foruok

2018-01-02

这个系列超级赞！

永靖

2017-12-15

你说服务需要具备，4个接口，start，stop，appconfig，healthcheck。
有2点疑问，1start不是系统外部命令行启动的时候，运行的main函数？2appconfig，很多参数比如配置文件或者配置文件地址，都应该是允许时传人参数，允许时可以读取配置，可以修改部分配置，例如修改日志打印级别等，是不是这些？

最初的信仰

2017-12-14

期待后面的文章

永靖

2017-12-14

全栈监控的各项核心技术指标能给详细介绍一下吗

Case

2017-12-14

高内聚低耦合，强大的可扩展性，从很多大型软件后台，都可以看到他们的发展趋势，基本就是分而治之，随着业务发展而形成很多独立的模块，然而那是花了大量时间和人力来实现的，拥抱docker＋云时代。

李志博

2017-12-14

马上要搞监控了，超级期待监控的文章

关冕

2018-06-22

"架构软件管理"可以再展开讲一下吗？

猪猪

2018-06-13

真的是干货，不错

Geek_22d08b

2018-05-15

请问如果采用阿里云华为云的话，那么多技术要实现是不是只要购买阿里云他们相应的产品，然后配置下就可以了，就没程序员什么事了？

朱海峰

2018-03-27

老师，分布式 ，去中心化，讲讲

进阶的码农

2018-01-19

全是干货 受益匪浅 谢谢浩子哥

雪花飘飘

2017-12-22

陈老师，希望讲解下分布式事物的实现方式

作者回复

2017-12-24

后面的文章会讲的

吉祥

2017-12-20

en hao

木偶

2017-12-15

干货满满 每一项都可以 单独拿出来 深入调研学习