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

数据镜像

数据分区

缓存分区 缓存更新 缓存命中

负载均衡 网关系统 负载均衡

服务路由

服务发现

异步系统 消息队列 消息持久 异步事务

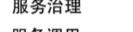
分区策略 数据访问层 数据一致性

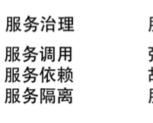
由。 • 负载均衡系统,是做水平扩展的关键技术。其可以用多台机器来共同分担一部分流量请求。 • 异步调用。异步系统主要通过消息队列来对请求做排队处理,这样可以把前端的请求的峰值给"削

- 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





入服务调用间的依赖问题。

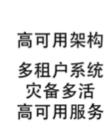


的复制或是重新分片,迁移的时候还要迁移数据到其它机器上。

服务冗余



限流降级



高可用架构



自动化运维

 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

• 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

• 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

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。

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

• **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

• 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

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。

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。

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。

一系列的自动化运维技术了。

会在这里,包括灰度发布之类的功能也在这里。

决。

最大的坑。

服务/资源调度;

状态/数据调度;

• 开发和运维的自动化。

应用整体监控

OS、主机、网络…

消息队列、缓存、数据库、

应用容器、网关、RPC框

API请求、吞吐量、响应

调用链路、函数调用栈、

时间、错误码、SQL语句、

流量调度;

最核心的东西。

• 基础层监控

中间件层监控

架、JVM···

业务指标…

应用层监控

• 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

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

• **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

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

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。

• 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、

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

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

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。

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。

资源/服务调度

CPU, 内存、磁盘、网

服务编排、服务复本、服 务容量伸缩、故障服务迁 移、服务生命周期管理…

多租户、架构版本管理、 架构部署、运行、更新、 销毁管理、多租户管理、

计算资源调度

服务调度

架构调度

灰度发布…

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;

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

状态/数据调度

数据可用性

数据一致性

数据分布式

多复本保存

读写一致性策略

数据索引、分片

流量调度

服务发现、服务路由、服

务降级、服务熔断、服务

负载均衡、流量分配、流

协议转换、请求校验、数

量控制、异地灾备 …

据缓存、数据计算…

服务治理

保护…

流量控制

流量管理

• 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

• 分布式系统架构的冰与火

每邀请一位好友订阅 你可获得36元 现金返现

获取海报 🕥

戳此获取你的专属海报

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。

- 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,
- 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 极客时间 左耳朵耗子 灰深技品 全年独家专栏《左耳听风》

