41 | 弹力设计篇之“认识故障和弹力设计”

2018-2-20 陈皓

我前面写的《分布式系统架构的本质》系列文章，从分布式系统的业务层、中间件层、数据库层等各个层面介绍了高并发架构、异地多活架构、容器化架构、微服务架构、高可用架构、弹性化架构等，也就是所谓的“纲”。通过这个“纲”，你能够按图索骥，掌握分布式系统中每个部件的用途与总体架构思路。

为了让你更深入地了解分布式系统，在接下来的几期中，我想谈谈分布式系统中一些比较关键的设计模式，其中包括容错、性能、管理等几个方面。

容错设计又叫弹力设计，其中着眼于分布式系统的各种“容忍”能力，包括容错能力（服务隔离、异步调用、请求幂等性）、可伸缩性（有/无状态的服务）、一致性（补偿事务、重试）、应对大流量的能力（熔断、降级）。可以看到，在确保系统正确性的前提下，系统的可用性是弹力设计保障的重点。

管理篇会讲述一些管理分布式系统架构的一些设计模式，比如网关方面的，边车模式，还有一些刚刚开始流行的，如Service Mesh相关的设计模式。

性能设计篇会讲述一些缓存、CQRS、索引表、优先级队列、业务分片等相关的架构模式。

我相信，你在掌握了这些设计模式之后，无论是对于部署一个分布式系统，开发一个分布式的业务模块，还是研发一个新的分布式系统中间件，都会有所裨益。

今天分享的就是《分布式系统设计模式》系列文章中的第一篇《弹力设计篇之“认识故障和弹力设计”》。

系统可用性测量

对于分布式系统的容错设计，在英文中又叫Resiliency（弹力）。意思是，系统在不健康、不顺，甚至出错的情况下有能力hold得住，挺得住，还有能在这种逆境下力挽狂澜的能力。

要做好一个设计，我们需要一个设计目标，或是一个基准线，通过这个基准线或目标来指导我们的设计，否则在没有明确基准线的指导下，设计会变得非常不明确，并且也不可预测，不可测量。可测试和可测量性是软件设计中非常重要的事情。

我们知道，容错主要是为了可用性，那么，我们是怎样计算一个系统的可用性的呢？下面是一个工业界里使用的一个公式：

$$Availability=\frac{MTTF}{MTTF +MTTR}$$

其中，

MTTF 是 Mean Time To Failure，平均故障前的时间，即系统平均能够正常运行多长时间才发生一次故障。系统的可靠性越高，MTTF越长。（注意：从字面上来说，看上去有Failure的字样，但其实是正常运行的时间。）

MTTR 是 Mean Time To Recovery，平均修复时间，即从故障出现到故障修复的这段时间，这段时间越短越好。

这个公式就是计算系统可用性的，也就是我们常说的，多少个9，如下表所示。

系统可用性%	宕机时间/年	宕机时间/月	宕机时间/周	宕机时间/天
90% (1 个 9)	36.5 天	72 小时	16.8 小时	2.4 小时
99% (2 个 9)	3.65 天	7.20 小时	1.68 小时	14.4 分
99.9% (3 个 9)	8.76 小时	43.8 分	10.1 分钟	1.44 分
99.99% (4 个 9)	52.56 分	4.38 分	1.01 分钟	8.66 秒
99.999% (5 个 9)	5.26 分	25.9 秒	6.05 秒	0.87 秒

根据上面的这个公式，为了提高可用性，我们要么提高系统的无故障时间，要么减少系统的故障恢复时间。

然而，我们要明白，我们运行的是一个分布式系统，对于一个分布式系统来说，要不出故障简直是太难了。

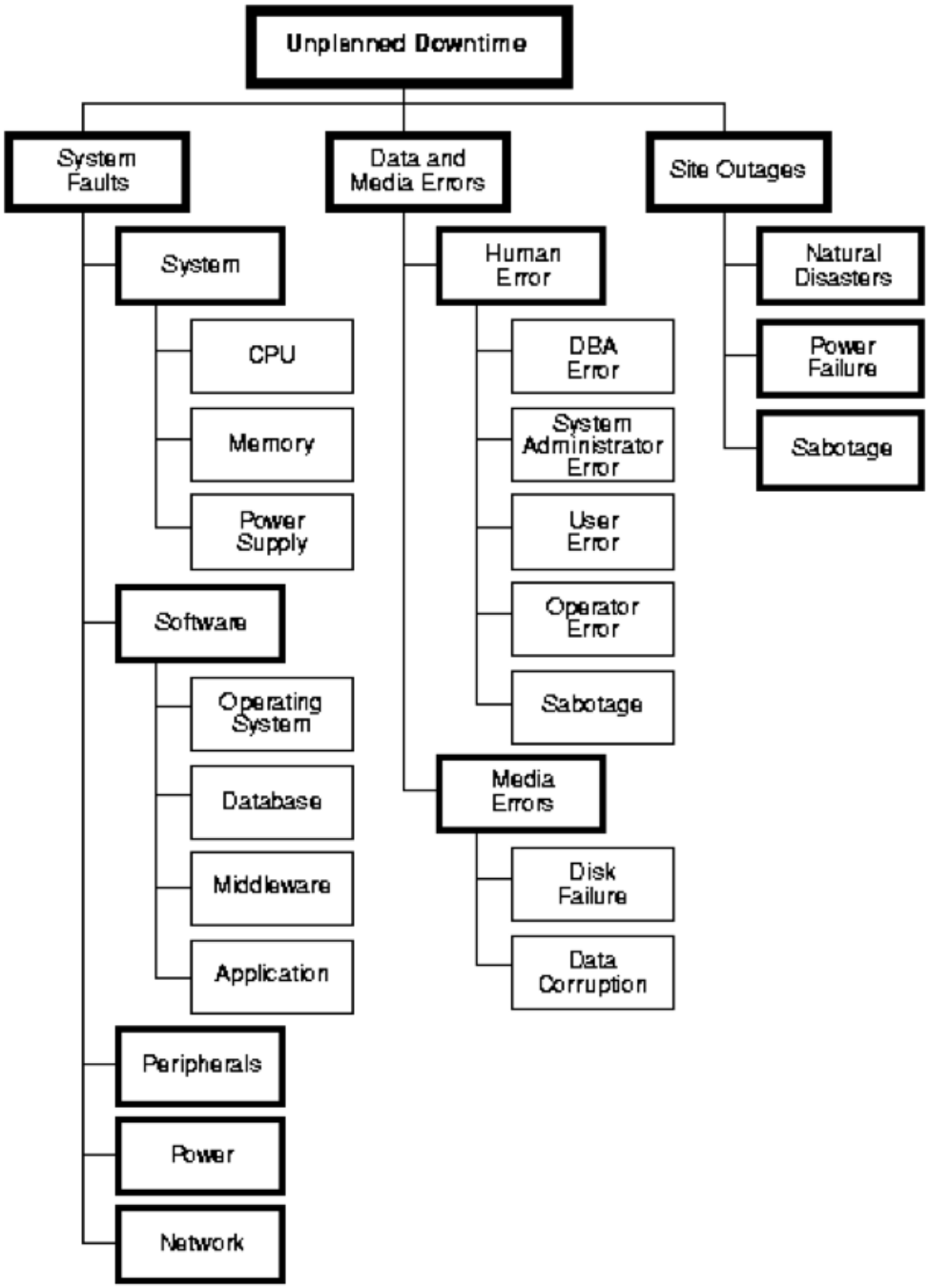
故障原因

老实说，我们很难计算我们设计的系统有多少的可用性，因为影响一个系统的因素实在是太多了，除了软件设计，还有硬件，还有第三方服务（如电信联通的宽带SLA），当然包括“建筑施工队的挖掘机”。

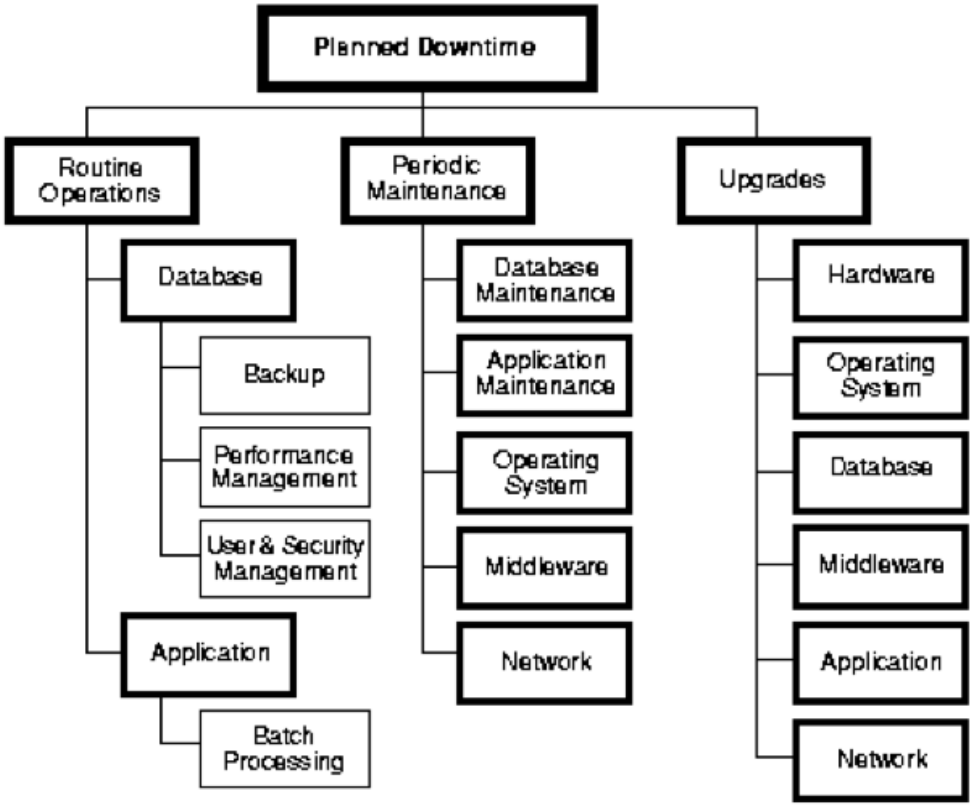
所以，正如SLA的定义，这不只是一个技术指标，而是一种服务提供商和用户之间的contract或契约。这种工业级的玩法，就像飞机一样，并不是把飞机造出来就好了，还有大量的无比专业的配套设施、工具、流程、管理和运营。

简而言之，SLA的几个9就是能持续提供可用服务的级别。不过，工业界中，会把服务不可用的因素分成两种：一种是有计划的，一种是无计划的。

无计划的宕机原因。下图来自Oracle的 High Availability Concepts and Best Practices。



有计划的宕机原因。下图来自Oracle的High Availability Concepts and Best Practices。



可以看到，宕机原因主要有以下这些。

无计划的

系统级故障，包括主机、操作系统、中间件、数据库、网络、电源以及外围设备。

数据和中介的故障，包括人员误操作、硬盘故障、数据乱了。

还有自然灾害、人为破坏，以及供电问题等。

有计划的

日常任务：备份，容量规划，用户和安全管理，后台批处理应用。

运维相关：数据库维护、应用维护、中间件维护、操作系统维护、网络维护。

升级相关：数据库、应用、中间件、操作系统、网络，包括硬件升级。

我们再给它们归个类。

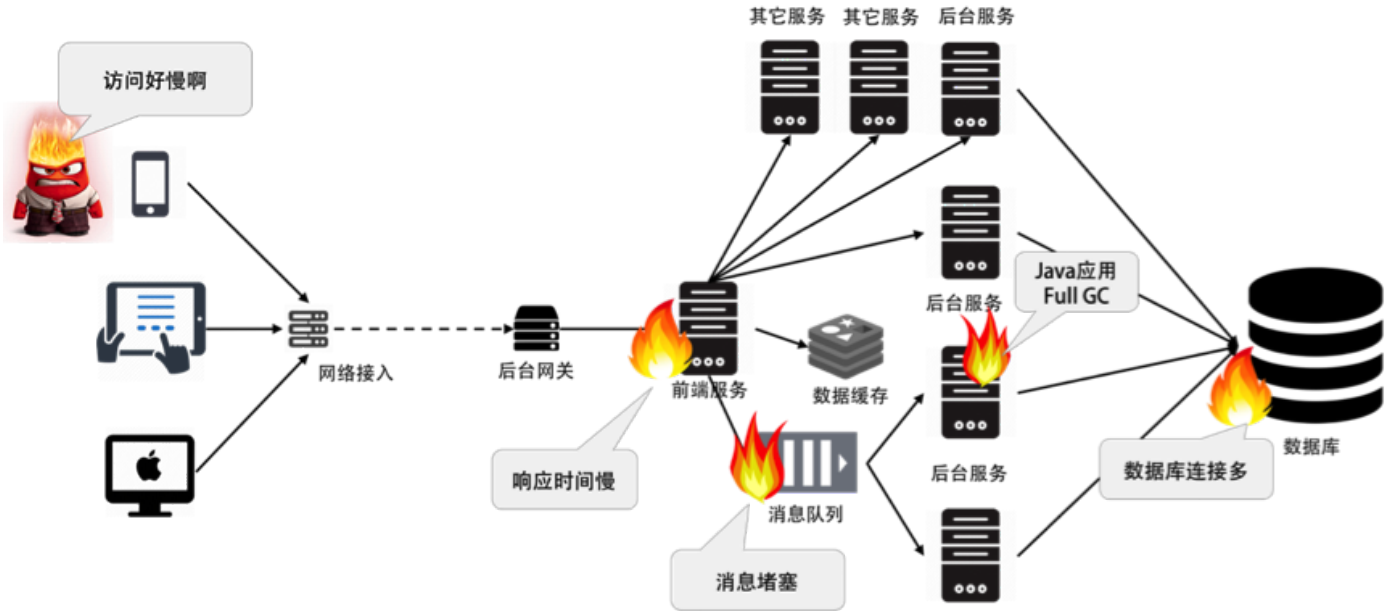
1. **网络问题**。网络链接出现问题，网络带宽出现拥塞……
2. **性能问题**。数据库慢SQL、Java Full GC、硬盘IO过大、CPU飙高、内存不足……
3. **安全问题**。被网络攻击，如DDoS等。

4. **运维问题**。系统总是在被更新和修改，架构也在不断地被调整，监控问题……
5. **管理问题**。没有梳理出关键服务以及服务的依赖关系，运行信息没有和控制系统同步……
6. **硬件问题**。硬盘损坏、网卡出问题、交换机出问题、机房掉电、挖掘机问题……

故障不可避免

如果你看过我写过的《分布式系统架构的本质》和《故障处理》这两个系列的文章，就会知道要管理好一个分布式系统是一件非常难的事。对于大规模的分布式系统，出现故障基本上就是常态，甚至还有些你根本就不知道会出问题的地方。

在今天来说，一个分布式系统的故障已经非常复杂了，因为故障是分布式的、多米诺骨牌式的。就像我在《分布式系统架构的本质》中展示过的这个图一样。



如果你在云平台上，或者使用了“微服务”，面对大量的IoT设备以及不受控制的用户流量，那么系统故障会更为复杂和变态。因为上面这些因素增加了整个系统的复杂度。

所以，要充分地意识到下面两个事。

故障是正常的，而且是常见的。

故障是不可预测突发的，而且相当难缠。

所以，亚马逊的AWS才会把Design for Failure做为其七大Design Principle的重点。这告诉我们，不要尝试着去避免故障，而是要把处理故障的代码当成正常的功能做在架构里写在代码里。

因为我们要干的事儿就是想尽一切手段来降低MTTR——故障的修复时间。

这就是为什么我们把这个设计叫做弹力（Resiliency）。

> 一方面，在好的情况下，这个事对于我们的用户和内部运维来说是完全透明的，系统自动修复不需要人的干预。
>
> 另一方面，如果修复不了，系统能够做自我保护，而不让事态变糟糕。

这就是所谓的“弹力”——能上能下。这让我想到三国杀里赵云的技能——“能进能退乃真正法器”，哈哈。

小结

好了，今天的内容就到这里。相信通过今天的学习，你应该已经明白了弹力设计的真正目的，并对系统可用性的衡量指标和故障的各种原因有所了解。下一篇文章，我们将开始罗列一些相关的设计模式。

在文章的最后，很想听听大家在设计一个分布式系统时，设定了多高的可用性指标？实现的难点在哪里？踩过什么样的坑？你是如何应对的？

文末给出了《分布式系统设计模式》系列文章的目录，希望你能在这个列表里找到自己感兴趣的内容。

降级设计degradation

弹力设计总结

管理设计篇

分布式锁Distributed Lock

配置中心Configuration Management

边车模式Sidecar

服务网格Service Mesh

网关模式Gateway

部署升级策略

性能设计篇

缓存Cache

异步处理Asynchronous

数据库扩展

秒杀Flash Sales

边缘计算Edge Computing

精选留言 21

华烬
1519630230

看到挖掘机的时候我笑了，印象中真的经历过光纤被挖断的故障

songyy
1519221992

我觉得自己缺少解决 大规模 高可用 分布式 问题的经验，一直希望在这方面进行深挖但无奈工作范围限制，没有相关的问题可以遇到。期待能在这个系列之中看到更多的例子 😁

一飞
1553785412

异步调用为啥是容错设计？ 应该是提高性能的一种策略。

蓝海
1519275252

耗子哥可否在后面出一篇有关gcc优化带来的相关问题（各种崩溃，优化选项对程序做了哪些假设，哪些"非标准"的代码会导致优化错误），如何判断崩溃是由于优化，二进制不兼容，链接错误导致，而非一般的代码错误。gcc的优化选项看了官网说明很多遍，但说明过于简洁（编译原理只停留在前端印象，优化技术生疏），想了解的感性一些。这些bug问题解决都很费力，想归纳出一条方法经验论，怎样的代码要求才能对各种优化级别不出错（gcc本身bug除外）。以上的问题以及问题本身是否成立，想请耗子哥指导

卢俊杰 _JAY
1519084257

以前或多或少写过一些数据库, MQ自动重连的代码，不过还没有一个整体的认识，多谢作者把这个事情系统化，条理清晰多了

疾风紫狼
1565261570

能进能退乃真正法器可还行。

Dimple
1564125590

看完分布式，以为一块难点看完了；这下好了，还有分布式系统设计模式，又得扒一层皮，慢慢消化。耗子叔的输出，真的帮助太大

雨巷
1563671403

偶然再回顾了一下，容错设计又叫弹力设计，弹力设计这个名字我google了一下，并没有官方的定义，放在这里任意误导读者

edisonhuang
1561941204

分布式系统出故障是不可避免的，弹力设计的关键是要提高系统的可用性，提高MTTF，提高MTTF，一是拉长系统稳定运行时间，一是减少故障恢复时间。
由于分布式系统故障呢普遍性，因此在分布式系统设计的和开发的过程中，就要把故障当作不可或缺的一环来处理，尽可能让故障恢复过程自动化，从而真正提高系统可用

godtrue
1548203450

高度上来了，分布式系统也在弄，不过整体系统认知不足，正好学习下。
分布式系统太过分散和复杂，网络环境有不稳定，所以，问题不可避免，痛点感觉都在网络通信这一块了。

十八哥
1546866870

首先所有接口设计足够协议化，这是弹力前提。

痴痴笑笑(Bruce)
1546605456

目前正在做这方面的工作，刚好学以致用

妮妮
1543741953

很多公司应该支撑不到挖掘机这个阶段就倒闭了。

LeO
1536803293

提纲挈领，非常的系统地介绍了系统的高可用的设计理念

LeO
1536803292

提纲挈领，非常的系统地介绍了系统的高可用的设计理念

蜗Amazon牛

1534940242

老师你说的分布式系统架构本质的文章指的那几篇呢？分布式关键技术那几篇吗？其余的我也没看到啊

data

1522802246

老师可以提供代码案例来讲解吗这样感觉可以学的更多哈哈

作者回复 这些东西的代码量太大，你可以参看开源软件

厉害了我的国

1520299314

能进能退乃真正法器

郎哲

1519619124

有一段时间总是喜欢把iot和im 划等号，iot最大特点就是您文章提到流量不受控制还一直在线。

楊_宵夜

1519273753

耗子叔每篇文章真是干货十足。

Tony Du

1519205242

终于上新了