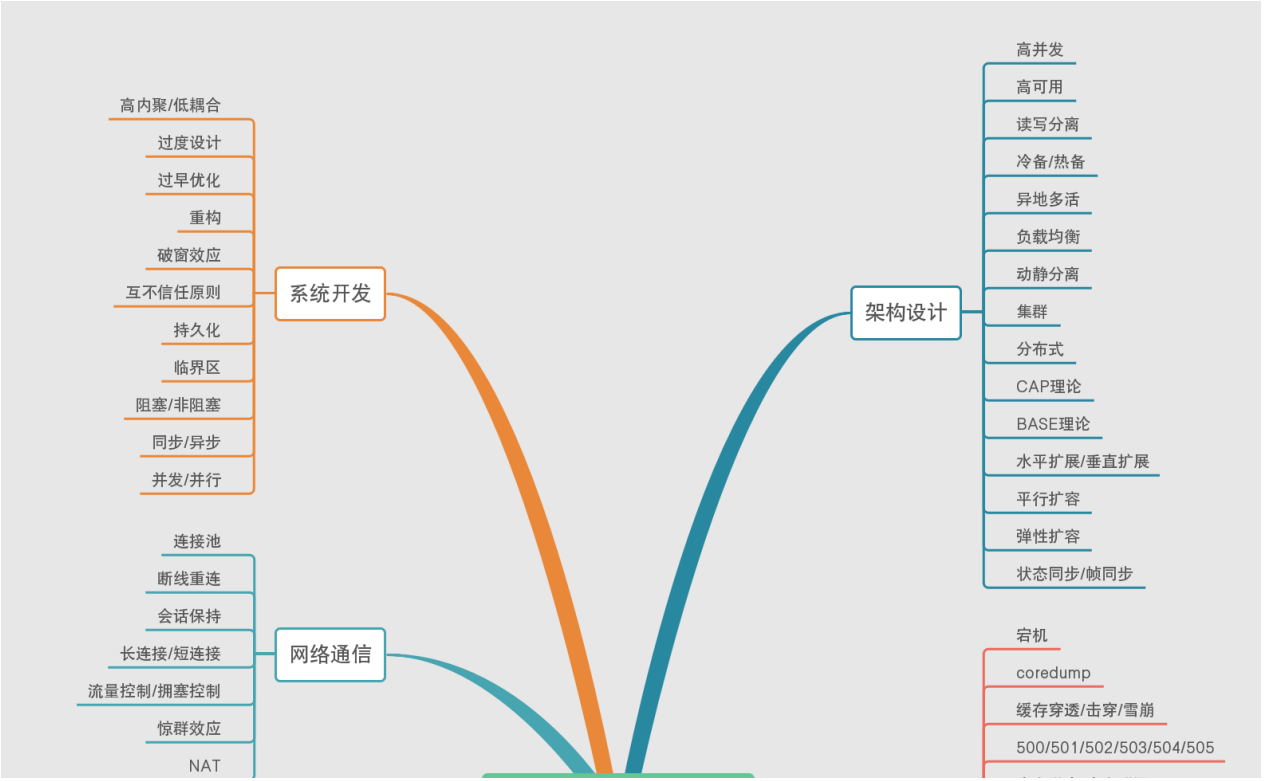
后端开发术语大全

willlv 2019年07月30日 09:45

| 导语 工欲善其事，必先利其器；士欲宣其义，必先读其书。



关于作者

willlv(吕琢)

S3\腾讯教育创想合作中心\K12产品...

作者文章

- Kubernetes Deployment故障排除图解指南
- 几张图讲讲什么是 Kubernetes Serverless
- 几张图解释明白大家都在说的 Istio
- 几张图解释明白 Kubernetes Ingress
- 几张图就把 Kubernetes Service 掰扯清楚了

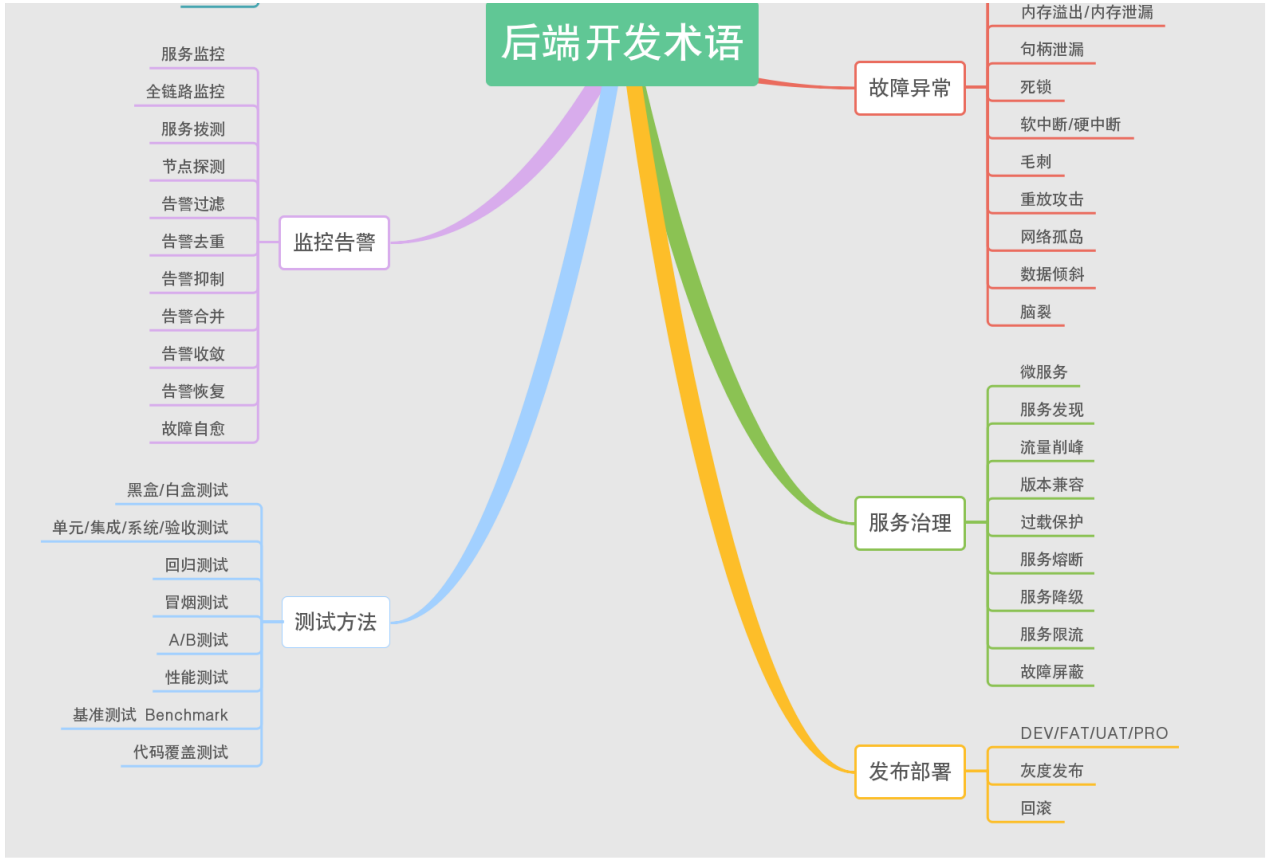
收录于

2019腾讯年度好文入围...

浏览 3944 收藏 315

奇技淫巧

腾讯知识奖2019年6-7月入围文章



一. 系统开发

高内聚/低耦合

高内聚指一个软件模块是由相关性很强的代码组成，只负责一项任务，也就是常说的单一责任原则。模块的内聚反映模块内部联系的紧密程度。
模块之间联系越紧密，其耦合性就越强，模块的独立性则越差。模块间耦合高低取决于模块间接口的复杂性、调用的方式及传递的信息。一个完整的系统，模块与模块之间，尽可能的使其

独立存在。
通常程序结构中各模块的内聚程度越高，模块间的耦合程度就越低。

过度设计

过度设计就是进行了过多的面向未来的设计或者说把相对简单的事情想复杂了，过度追求模块化、可扩展性、设计模式等，为系统增加了不必要的复杂度。

过早优化

过早指的不是在开发过程的早期，而是在还没弄清楚需求未来的变化的走向的时候。你的优化不仅可能导致你无法很好地实现新的需求，而且你对优化的预期的猜测有可能还是错的，导致实际上你除了把代码变复杂以外什么都没得到。
正确的方法是，先有质量地实现你的需求，写够testcase，然后做profile去找到性能的瓶颈，这个时候才做优化。

> 扩展阅读：https://blog.csdn.net/jinzhencs/article/details/50580650

重构 (Refactoring)

重构（Refactoring）就是通过调整程序代码改善软件的质量、性能，使其程序的设计模式和架构更趋合理，提高软件的扩展性和维护性。

> 扩展阅读：https://www.jianshu.com/p/520a7d13a7ae

破窗效应

又称破窗理论，破窗效应（Broken windows theory）是犯罪学的一个理论。此理论认为环境中的不良现象如果被放任存在，会诱使人们仿效，甚至变本加厉。一幢有少许破窗的建筑为例，如果那些窗不被修理好，可能将会有破坏者破坏更多的窗户。最终他们甚至会闯入建筑内，如果发现无人居住，也许就在那里定居或者纵火。
应用在软件工程上就是，一定不能让系统代码或者架构设计的隐患有冒头的机会，否则随着时间的推移，隐患会越来越重。反之，一个本身优质的系统，会让人不由自主的写出优质的代码。

互不信任原则

指在程序运行上下游的整个链路中，每个点都是不能保证绝对可靠的，任何一个点都可能随时发生故障或者不可预知的行为，包括机器网络、服务本身、依赖环境、输入和请求等，因此要处处设防。

> 扩展阅读：https://cloud.tencent.com/developer/article/1005918

持久化 (Persistence)

持久化是将程序数据在临时状态和持久状态间转换的机制。通俗的讲，就是临时数据（比如内存中的数据，是不能永久保存的）持久化为持久数据（比如持久化至数据库或者本地磁盘中，能够长久保存）。

临界区

临界区用来表示一种公共资源或者说是共享数据，可以被多个线程使用，但是每一次，只能有一个线程使用它，一旦临界区资源被占用，其他线程要想使用这个资源，就必须等待。

阻塞/非阻塞

阻塞和非阻塞通常形容多线程间的相互影响。比如一个线程占用了临界区资源，那么其它所有需要这个资源的线程就必须在这个临界区中进行等待，等待会导致线程挂起。这种情况就是阻塞。此时，如果占用资源的线程一直不愿意释放资源，那么其它所有阻塞在这个临界区上的线程都不能工作。而非阻塞允许多个线程同时进入临界区。

同步/异步

通常同步和异步是指函数/方法调用方面。
同步就是在发出一个函数调用时，在没有得到结果之前，该调用就不返回。异步调用会瞬间返回，但是异步调用瞬间返回并不代表你的任务就完成了，他会在后台起个线程继续进行任务，等任务执行完毕后通过回调callback或其他方式通知调用方。

并发/并行

- **并行(parallel)**
  指在同一时刻，有多条指令在多个处理器上同时执行。所以无论从微观还是从宏观来看，二者都是一起执行的。
- **并发(concurrency)**
  指在同一时刻只能有一条指令执行，但多个进程指令被快速的轮换执行，使得在宏观上具有多个进程同时执行的效果，但在微观上并不是同时执行的，只是把时间分成若干段，使多个进程快速交替的执行。

> 扩展阅读：https://www.jianshu.com/p/cbf9588b2afb

二. 架构设计

高并发 (High Concurrency)

由于分布式系统的问世，高并发（High Concurrency）通常是指通过设计保证系统能够同时并行处理很多请求。通俗来讲，高并发是指在同一个时间点，有很多用户同时的访问同一 API 接口或者 Url 地址。它经常会发生在有大活跃用户量，用户高聚集的业务场景中。

> 扩展阅读：https://www.jianshu.com/p/be66a52d2b9b

高可用 (High Availability)

高可用HA（High Availability）是分布式系统架构设计中必须考虑的因素之一，它通常是指，一个系统经过专门的设计，以减少停工时间，而保持其服务的高度可用性。

> 扩展阅读：https://www.cnblogs.com/shizhiyi/p/7750530.html

读写分离

为了确保数据库产品的稳定性，很多数据库拥有双机热备功能。也就是，第一台数据库服务器，是对外提供增删改业务的生产服务器；第二台数据库服务器，主要进行读的操作。

冷备/热备

- **冷备**
  两个服务器，一台运行，一台不运行做为备份。这样一旦运行的服务器宕机，就把备份的服务器运行起来。冷备的方案比较容易实现，但冷备的缺点是主机出现故障时备机不会自动接管，需要主动切换服务。
- **热备**
  即是通常所说的active/standby方式，服务器数据包括数据库数据同时往两台或多台服务器写。当active服务器出现故障的时候，通过软件诊测（一般是通过心跳诊断）将standby机器激活，保证应用在短时间内完全恢复正常使用。当一台服务器宕机后，自动切换到另一台备用机使用。

> 扩展阅读：https://cloud.tencent.com/developer/news/316050

异地多活

异地多活一般是指在不同城市建立独立的数据中心，“活”是相对于冷备份而言的，冷备份是备份全量数据，平时不支撑业务需求，只有在主机房出现故障的时候才会切换到备用机房，而多活，是指这些机房在日常的业务中也需要走流量，做业务支撑。

> 扩展阅读：https://www.cnblogs.com/jaychan/p/9242325.html

负载均衡 (Load Balance)

负载均衡，是对多台服务器进行流量分发的负载均衡服务。可在多个实例间自动分配应用程序的对外服务能力，通过消除单点故障提升应用系统的可用性，让您实现更高水平的应用程序容错能力，从而无缝提供分配应用程序流量所需的负载均衡容量，为您提供高效、稳定、安全的服务。

> 扩展阅读：https://www.cnblogs.com/xybaby/p/7867735.html

动静分离

动静分离是指在web服务器架构中，将静态页面与动态页面或者静态内容接口和动态内容接口分开不同系统访问的架构设计方法，进而提升整个服务访问性能和可维护性。

> 扩展阅读：https://blog.csdn.net/xuxiaopang0417/article/details/80003044

集群

单台服务器的并发承载能力总是有限的，当单台服务器处理能力达到性能瓶颈的时，将多台服务器组合起来提供服务，这种组合方式称之为集群，集群中每台服务器就叫做这个集群的一个“节点”，每个节点都能提供相同的服务，从而成倍的提升整个系统的并发处理能力。

分布式

分布式系统就是将一个完整的系统按照业务功能拆分成很多独立的子系统，每个子系统就被称为“服务”，分布式系统将请求分拣和分发到不同的子系统，让不同的服务来处理不同的请求。在分布式系统中，子系统独立运行，它们之间通过网络通信连接起来实现数据互通和组合服务。

CAP理论

CAP理论，指的是在一个分布式系统中，Consistency(一致性)、Availability(可用性)、Partition Tolerance(分区容错性)，不能同时成立。

一致性：它要求在同一时刻点，分布式系统中的所有数据备份都相同或者都处于同一状态。

可用性：在系统集群的一部分节点宕机后，系统依然能够正确的响应用户的请求。

分区容错性：系统能够容忍节点之间的网络通信的故障。

简单的来说，在一个分布式系统中，最多能支持上面的两种属性。但显然既然是分布式注定我们是必然要进行分区，既然分区，我们就无法百分百避免分区的错误。因此，我们只能在一致性和可用性去作出选择。

在分布式系统中，我们往往追求的是可用性，它的重要性比一致性要高，那么如何实现高可用，这里又有一个理论，就是 BASE 理论，它给 CAP 理论做了进一步的扩充。

BASE理论

BASE 理论指出：

Basically Available（基本可用）
Soft state（软状态）
Eventually consistent（最终一致性）

BASE 理论是对 CAP 中的一致性和可用性进行一个权衡的结果，理论的核心思想就是：我们无法做到强一致，但每个应用都可以根据自身的业务特点，采用适当的方式来使系统达到最终一致性。

水平扩展/垂直扩展

- **水平扩展 Scale Out**
  通过增加更多的服务器或者程序实例来分散负载，从而提升存储能力和计算能力。
- **垂直扩展 Scale Up**
  提升单机处理能力。垂直扩展的方式又有两种：
  1. 增强单机硬件性能，例如：增加CPU核数如32核，升级更好的网卡如万兆，升级更好的硬盘如SSD，扩充硬盘容量如2T，扩充系统内存如128G;
  2. 提升单机软件或者架构性能，例如：使用Cache来减少IO次数，使用异步来增加单服务吞吐量，使用无锁数据结构来减少响应时间；

平行扩容

与水平扩展类似。
集群服务器中的节点均为平行对等节点，当需要扩容时，可以通过添加更多节点以提高集群的服务能力。一般来说服务器中关键路径（如服务器中的登录、支付、核心业务逻辑等）都需要支持运行时动态平行扩容。

弹性扩容

指对部署的集群进行动态在线扩容。
弹性扩容系统可以根据实际业务环境按照一定策略自动地添加更多的节点（包括存储节点、计算节点、网络节点）来增加系统容量、提高系统性能或者增强系统可靠性，或者同时完成这三个目标。

状态同步/帧同步

- **状态同步**
  状态同步是指服务器负责计算全部的游戏逻辑，并且广播这些计算的结果，客户端仅仅负责发送玩家的操作，以及表现收到的游戏结果。状态同步安全性高，逻辑更新方便，断线重连快，但是开发效率较低，网络流量随游戏复杂度增加，服务器需要承载更大压力。

- **帧同步**
  服务端只转发消息，不做任何逻辑处理，各客户端每秒帧数一致，在每一帧都处理同样的输入数据。帧同步需要保证系统在相同的输入下，要有相同的输出。帧同步开发效率高，流量消耗低而且稳定，对服务器的压力非常小。但是网络要求高，断线重连时间长，客户端计算压力大。

> 扩展阅读：https://www.cnblogs.com/little-fly/p/10034579.html

三. 网络通信

连接池

预先建立一个连接缓冲池，并提供一套连接使用、分配、管理策略，使得该连接池中的连接可以得到高效、安全的复用，避免了连接频繁建立、关闭的开销。

常用的连接池有数据库连接池、redis连接池、TCP连接池等等，其主要目的是通过复用来减少创建和释放连接的开销。连接池实现通常需要考虑以下几个问题：

1）初始化：启动即初始化和惰性初始化。启动初始化可以减少一些加锁操作和需要时可直接使用，缺点是可能造成服务启动缓慢或者启动后没有任务处理，造成资源浪费。惰性初始化是真正有需要的时候再去创建，这种方式可能有助于减少资源占用，但是如果面对突发的任务请求，然后瞬间去创建一堆连接，可能会造成系统响应慢或者响应失败，通常我们会采用启动即初始化的方式。

2）连接数目：权衡所需的连接数，连接数太少则可能造成任务处理缓慢，太多不但使任务处理慢还会过度消耗系统资源。

3）连接取出：当连接池已经无可用连接时，是一直等待直到有可用连接还是分配一个新的临时连接。

4）连接放入：当连接使用完毕且连接池未满时，将连接放入连接池（包括3中创建的临时连接），否则关闭。

5）连接检测：长时间空闲连接和失效连接需要关闭并从连接池移除。常用的检测方法有：使用时检测和定期检测。

断线重连

由于网络波动造成用户间歇性的断开与服务器的连接，待网络恢复之后服务器尝试将用户连接到上次断开时的状态和数据。

会话保持

会话保持是指在负载均衡器上的一种机制，可以识别客户端与服务器之间交互过程的关连性，在作负载均衡的同时还保证一系列相关连的访问请求都会分配到一台机器上。用人话来表述就是：在一次会话过程中发起的多个请求都会落到同一台机器上。

> 扩展阅读：https://blog.csdn.net/li12412414/article/details/81026209

长连接/短连接

通常是指TCP的长连接和短连接。
长连接就是建立TCP连接后，一直保持这个连接，一般会中间彼此发送心跳来确认对应的存在，中间会做多次业务数据传输，一般不会主动断开连接。
短连接一般指建立连接后，执行一次事务后（如：http请求），然后就关掉这个连接。

> 扩展阅读：https://www.cnblogs.com/gotodsp/p/6366163.html

流量控制/拥塞控制

- **流量控制**
  防止发送方发的太快，耗尽接收方的资源，从而使接收方来不及处理。

- **拥塞控制**
  防止发送方发的太快，使得网络来不及处理产生拥塞，进而引起这部分乃至整个网络性能下降的现象，严重时甚至会导致网络通信业务陷入停顿。

> 扩展阅读：https://blog.csdn.net/dangzhangjing97/article/details/81008836

惊群效应

惊群效应也有人叫做雷鸣群体效应，不过叫什么，简言之，惊群现象就是多进程（多线程）在同时阻塞等待同一个事件的时候（休眠状态），如果等待的这个事件发生，那么他就会唤醒等待的所有进程（或者线程），但是最终却只可能有一个进程（线程）获得这个时间的“控制权”，对该事件进行处理，而其他进程（线程）获取“控制权”失败，只能重新进入休眠状态，这种现象和性能浪费就叫做惊群。

> 扩展阅读：https://www.cnblogs.com/zafu/p/8251849.html

NAT

NAT（Network Address Translation，网络地址转换），就是替换IP报文头部的地址信息。NAT通常部署在一个组织的网络出口位置，通过将内部网络IP地址替换为出口的IP地址提供公网可达性和上层协议的连接能力。

> 扩展阅读：https://www.cnblogs.com/imstudy/p/5458133.html

四. 故障异常

宕机

宕机，一般情况下指的就是计算机主机出现意外故障而死机。其次，一些服务器例如数据库死锁也可以称为宕机，一些服务器的某些服务挂掉了，就可以这么说。

coredump

当程序出错而异常中断时，OS会把程序工作的当前状态存储成一个coredunmp文件。通常情况下coredump文件包含了程序运行时的内存，寄存器状态，堆栈指针，内存管理信息等。

> 扩展阅读：https://blog.csdn.net/starzpc/article/details/78262819

缓存穿透/击穿/雪崩

- **缓存穿透**
  缓存穿透是指查询一个一定不存在的数据，由于缓存是不命中时需要从数据库查询，查不到数据则不写入缓存，这将导致这个不存在的数据每次请求都要到数据库去查询，进而给数据库带来压力。

- **缓存击穿**
  缓存击穿是指热点key在某个时间点过期的时候，而恰好在这个时间点对这个Key有大量的并发请求过来，从而大量的请求打到db。

- **缓存雪崩**
  缓存雪崩是指缓存中数据大批量到过期时间，而查询数据量巨大，引起数据库压力过大甚至down机。
  与缓存击穿不同的是：存击穿是热点key失效；缓存雪崩是大量的key同时失效。

> 扩展阅读：http://km.oa.com/group/626/articles/show/375512

500/501/502/503/504/505

- **500**
  Internal Server Error. 内部服务错误，一般是服务器遇到意外情况，而无法完成请求。
  可能原因:

1、程序错误，例如：ASP或者PHP语法错误；
2、高并发导致，系统资源限制不能打开过多的文件所致。

- **501**
  Not implemented. 服务器不理解或不支持请求的HTTP请求。

- **502**
  Bad Gateway. WEB服务器故障，可能是由于程序进程不够，请求的php-fpm已经执行，但是由于某种原因而没有执行完毕，最终导致php-fpm进程终止。
  可能原因：
  1、Nginx服务器，php-cgi进程数不够用；
  2、PHP执行时间过长；
  3、php-cgi进程死掉；

- **503**
  Service Unavailable. 服务器目前无法使用。系统维护服务器暂时的无法处理客户端的请求，这只是暂时状态。可以联系下服务器提供商。

- **504**
  Gateway Timeout. 服务器504错误表示超时，是指客户端所发出的请求没有到达网关，请求没有到可以执行的php-fpm，一般是与nginx.conf的配置有关。

- **505**
  HTTP Version Not Supported. 服务器不支持请求中所用的 HTTP 协议版本。（HTTP 版本不受支持）

除了500错误可能是程序语言错误，其余的报错，都大概可以理解为服务器或者服务器配置出现问题。

内存溢出/内存泄漏

- **内存溢出**
  内存溢出（Out Of Memory）指程序申请内存时，没有足够的内存供申请者使用，或者说，给了你一块存储int类型数据的存储空间，但是你却存储long类型的数据，那么结果就是内存不够用，此时就会报错OOM,即所谓的内存溢出。

- **内存泄漏**
  内存泄漏（Memory Leak）指程序中己动态分配的堆内存由于某种原因程序未释放或无法释放，造成系统内存的浪费，导致程序运行速度减慢甚至系统崩溃等严重后果。

> 扩展阅读：https://cloud.tencent.com/developer/article/1446591

句柄泄漏

句柄泄漏是进程在调用系统文件之后，没有释放已经打开的文件句柄。
一般句柄泄漏后的现象是，机器变慢，cpu飙升，出现句柄泄漏的cgi或server的cpu使用率增加。

> 扩展阅读：http://km.oa.com/group/19143/articles/show/162768

死锁

死锁是指两个或两个以上的线程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象，若无外力作用，它们都抑制处于阻塞状态并无法进行下去，此时称系统处于死锁状态或系统产生了死锁。

> 扩展阅读：https://baike.baidu.com/item/死锁/2196938

软中断/硬中断

- **硬中断**
  我们通常所说的中断指的是硬中断(hardirq)。由与系统相连的外设(比如网卡、硬盘)自动产生的。主要是用来通知操作系统系统外设状态的变化。

- **软中断**
  1.通常是硬中断服务程序对内核的中断；
  2.为了满足实时系统的要求，中断处理应该是越快越好。linux为了实现这个特点，当中断发生的时候，硬中断处理那些短时间就可以完成的工作，而将那些处理事件比较长的工作，放到中断之后来完成，也就是软中断(softirq)来完成。

> 扩展阅读：https://www.cnblogs.com/widic/p/7392485.html

毛刺

在短暂的某一刻，服务器性能指标（如流量、磁盘IO、CPU使用率等）远大于该时刻前后时间段。毛刺的出现代表这服务器资源利用不均匀，不充分，容易诱发其他更严重的问题。

重放攻击

攻击者发送一个目的主机已接收过的包，来达到欺骗系统的目的，主要用于身份认证过程，破坏认证的正确性。它是一种攻击类型，这种攻击会不断恶意或欺诈性地重复一个有效的数据传输，重放攻击可以由发起者，也可以由拦截并重发该数据的敌方进行。攻击者利用网络监听或者其他方式盗取认证凭据，之后再把它重新发给认证服务器。

网络孤岛

网络孤岛指集群环境中，部分机器与整个集群失去网络连接，分裂为一个小集群并且发生数据不一致的状况。

数据倾斜

对于集群系统，一般缓存是分布式的，即不同节点负责一定范围的缓存数据。我们把缓存数据分散度不够，导致大量的缓存数据集中到了一台或者几台服务节点上，称为数据倾斜。一般来说数据倾斜是由于负载均衡实施的效果不好引起的。

脑裂

脑裂是指在集群系统中，部分节点之间网络不可达而引起的系统分裂，不同分裂的小集群会按照各自的状态提供服务，原本的集群会同时存在不一致的反应，造成节点之间互相争抢资源，系统混乱，数据损坏。

五. 监控告警

服务监控

服务监控主要目的在服务出现问题或者快要出现问题时能够准确快速地发现以减小影响范围。服务监控一般有多种手段，按层次可划分为：

- 系统层（CPU、网络状态、IO、机器负载等）
- 应用层（进程状态、错误日志、吞吐量等）
- 业务层（服务/接口的错误码、响应时间）
- 用户层（用户行为、舆情监控、前端埋点）

全链路监控

服务拨测

服务拨测是探测服务（应用）可用性的监控方式，通过拨测节点对目标服务进行周期性探测，主要通过可用性和响应时间来度量，拨测节点通常有异地多个。

节点探测

节点探测是用来发现和追踪不同的机房（数据中心）节点之间网络可用性和通畅性的监控方式，主要通过响应时间、丢包率、跳数来度量，探测方法一般是ping、mtr或其他私有协议。

告警过滤

对某些可预知的告警进行过滤，不进入告警统计的数据，如少量爬虫访问导致的http响应500错误，业务系统自定义异常信息等。

告警去重

当一个告警通知负责人后，在这个告警恢复之前，不会继续收到相同的告警。

告警抑制

为了减少由于系统抖动带来的干扰，还需要实现抑制，例如服务器瞬间高负载，可能是正常的，只有持续一段时间的高负载才需要得到重视。

告警恢复

开发/运维人员不仅需要收到告警通知，还需要收到故障消除告警恢复正常的通知。

告警合并

对同一时刻产生的多条相同告警进行合并，如某个微服务集群同一时刻出现多个子服务负载过高的告警，需要合并成为一条告警。

告警收敛

有时某个告警产生时，往往会伴随着其它告警。这时可以只对根本原因产生告警，其它告警收敛为子告警一并发送通知。如云服务器出现CPU负载告警时往往伴随其搭载的所有系统的可用性告警。

故障自愈

实时发现告警，预诊断分析，自动恢复故障，并打通周边系统实现整个流程的闭环。

六. 服务治理

微服务

微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间相互协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务和服务之间采用轻量级的通信机制相互沟通（通常是基于HTTP的Restful API).每个服务都围绕着具体的业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。

> 扩展阅读：https://www.jianshu.com/p/009d98e30b2a

服务发现

服务发现是指使用一个注册中心来记录分布式系统中的全部服务的信息，以便其他服务能够快速的找到这些已注册的服务。服务发现是支撑大规模 SOA 和微服务架构的核心模块，它应该尽量做到高可用。

> 扩展阅读：https://blog.csdn.net/u011714033/article/details/94395175

流量削峰

如果观看抽奖或秒杀系统的请求监控曲线，你就会发现这类系统在活动开放的时间段内会出现一个波峰，而在活动未开放时，系统的请求量、机器负载一般都是比较平稳的。为了节省机器资源，我们不可能时时都提供最大化的资源能力来支持短时间的高峰请求。所以需要使用一些技术手段，来削弱瞬时的请求高峰，让系统吞吐量在高峰请求下保持可控。
削峰也可用于消除毛刺，使服务器资源利用更加均衡和充分。
常见的削峰策略有队列，限频，分层过滤，多级缓存等。

> 扩展阅读：https://www.jianshu.com/p/6746140bbb76

版本兼容

在升级版本的过程中，需要考虑升级版本后，新的数据结构是否能够理解和解析旧数据，新修改的协议是否能够理解旧的协议以及做出预期内合适的处理。这就需要在服务设计过程中做好版本兼容。

过载保护

过载是指当前负载已经超过了系统的最大处理能力，过载的出现，会导致部分服务不可用，如果处置不当，极有可能引起服务完全不可用，乃至雪崩。过载保护正是针对这种异常情况做的措施，防止出现服务完全不可用的现象。

服务熔断

服务熔断的作用类似于我们家用的保险丝，当某服务出现不可用或响应超时的情况时，为了防止整个系统出现雪崩，暂时停止对该服务的调用。

服务降级

服务降级是当服务器压力剧增的情况下，根据当前业务情况及流量对一些服务和页面有策略的降级，以此释放服务器资源以保证核心任务的正常运行。降级往往会指定不同的级别，面临不同的异常等级执行不同的处理。
根据服务方式：可以拒接服务，可以延迟服务，也有时候可以随机服务。
根据服务范围：可以砍掉某个功能，也可以砍掉某些模块。
总之服务降级需要根据不同的业务需求采用不同的降级策略。主要的目的就是服务虽然有损但是总比没有好。

- **熔断VS降级**
  ~ 相同点
  目标一致：都是从可用性和可靠性出发，为了防止系统崩溃；
  用户体验类似：最终都让用户体验到的是某些功能暂时不可用；
  ~ 不同点：
  触发原因不同：服务熔断一般是某个服务（下游服务）故障引起，而服务降级一般是从整体负荷考虑；

服务限流

限流可以认为服务降级的一种，限流就是限制系统的输入和输出流量已达到保护系统的目的。一般来说系统的吞吐量是可以被测算的，为了保证系统的稳定运行，一旦达到的需要限制的阈值，就需要限制流量并采取一些措施以完成限制流量的目的。比如：延迟处理，拒绝处理，或者部分拒绝处理等等。

> 扩展阅读：https://blog.csdn.net/jek123456/article/details/79901035

故障屏蔽

将故障机器从集群剔除，以保证新的请求不会分发到故障机器。

七. 测试方法

黑盒/白盒测试

黑盒测试不考虑程序内部结构和逻辑结构，主要是用来测试系统的功能是否满足需求规格说明书。一般会有一个输入值，一个输入值，和期望值做比较。

白盒测试主要应用在单元测试阶段，主要是对代码级的测试，针对程序内部逻辑结构，测试手段有：语句覆盖、判定覆盖、条件覆盖、路径覆盖、条件组合覆盖

单元/集成/系统/验收测试

软件测试一般分为4个阶段：单元测试、集成测试、系统测试、验收测试。

- **单元测试**
  单元测试是对软件中的最小可验证单元进行检查和验证，如一个模块、一个过程、一个方

法等。
单元测试粒度最小，一般由开发小组采用白盒方式来测试，主要测试单元是否符合“设计”。

- **集成测试**
  集成测试也叫做组装测试，通常在单元测试的基础上，将所有的程序模块进行有序的、递增的测试。
  集成测试界于单元测试和系统测试之间，起到“桥梁作用”，一般由开发小组采用白盒加黑盒的方式来测试，既验证“设计”，又验证“需求”。

- **系统测试**
  系统测试时将经过集成测试的软件，作为计算机系统的一部分，与系统中其他部分结合起来，在实际运行环境下进行一系列严格有效的测试，以发现软件潜在的问题，保证系统的正常运行。
  系统测试的粒度最大，一般由独立测试小组采用黑盒方式来测试，主要测试系统是否符合“需求规格说明书”。

- **验收测试**
  验收测试也称交付测试，是针对用户需求、业务流程进行的正式的测试，以确定系统是否满足验收标准，由用户、客户或其他授权机构决定是否接受系统。
  验收测试与系统测试相似，主要区别是测试人员不同，验收测试由用户执行。

> 扩展阅读：https://blog.csdn.net/lb838315586/article/details/85099064

回归测试

当发现并修改缺陷后，或在软件中添加新的功能后，重新测试。用来检查被发现的缺陷是否被改正，并且所做的修改没有引发新的问题。

冒烟测试

这一术语源自硬件行业。对一个硬件或硬件组件进行更改或修复后，直接给设备加电。如果没有冒烟，则该组件就通过了测试。在软件中，“冒烟测试”这一术语描述的是在将代码更改嵌入到产品的源树中之前对这些更改进行验证的过程。
冒烟测试是在软件开发过程中的一种针对软件版本包的快速基本功能验证策略，是对软件基本功能进行确认验证的手段，并非对软件版本包的深入测试。
比如：对于一个登录系统的冒烟测试，我们只需测试输入正确的用户名、密码，验证登录这一个核心功能点，至于输入框、特殊字符等，可以在冒烟测试之后进行。

性能测试

性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。负载测试和压力测试都属于性能测试，两者可以结合进行。
通过负载测试，确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。
压力测试是通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的最大服务级别的测试。

基准测试

基准测试（Benchmark）也是一种性能测试方式，用来测量机器的硬件最高实际运行性能，以及软件优化的性能提升效果, 同时也可以用来识别某段代码的CPU或者内存效率问题. 许多开发人员会用基准测试来测试不同的并发模式, 或者用基准测试来辅助配置工作池的数量, 以保证能最大化系统的吞吐量.

A/B测试

A/B测试，是用两组及以上随机分配的、数量相似的样本进行对比，如果实验组和对比组的实验结果相比，在目标指标上具有统计显著性，那就可以说明实验组的功能可以导致你想要的结果，从而帮你验证假设或者做出产品决定。

> 扩展阅读：https://www.jianshu.com/p/32cde5a205c9

代码覆盖测试

代码覆盖（Code coverage）是软件测试中的一种度量，描述程式中源代码被测试的比例和程度，所得比例称为代码覆盖率。
在做单元测试时，代码覆盖率常常被拿来作为衡量测试好坏的指标，甚至，用代码覆盖率来考核测试任务完成情况，比如，代码覆盖率必须达到80%或 90%。于是乎，测试人员费尽心思设计案例覆盖代码。

> 扩展阅读：https://blog.csdn.net/shenggaofei/article/details/52905428

八. 发布部署

DEV/PRO/FAT/UAT

- **DEV**
  Development environment
  开发环境 ，用于开发人员调试使用，版本变化较大。
- **FAT**
  Feature Acceptance Test environment
  功能验收 测试环境 ，用于软件测试人员测试使用。

- **UAT**
  User Acceptance Test environment
  用户验收测试环境，用于生产环境下的功能验证，可作为 预发布环境 。
- **PRO**
  Production environment
  生产环境 ，正式线上环境。

> 扩展阅读：https://www.cnblogs.com/chengkanghua/p/10607239.html

灰度发布

灰度发布是指在升级版本过程中，通过分区控制，白名单控制等方式对一部分用户先升级产品特性，而其余用户则保持不变，当一段时间后升级产品特性的用户没有反馈问题，就逐步扩大范围，最终向所有用户开放新版本特性，灰度发布可以保证整体系统的稳定，在初始灰度的时候就可以发现、修改问题，以保证其影响度。

回滚 (Rollback)

指的是程序或数据处理错误时，将程序或数据恢复到上一次正确状态(或者是上一个稳定版本)的行为。

最后更新于 2020-11-12 03:40

赏

2人已赞赏


分类：技术·运维 标签：后台开发(2) 运维监控(1) 后台(2) 实习生(1) 技术架构...(1)

本文专属二维码，扫一扫还能分享朋友圈

想要微信公众号推广本文章？点击获取链接

我顶 (237)　　已收藏 (1482)

大家评论

boxerzhang 2019-07-30 18:57:12

黑话大全

顶 (6) 回复

kennyge 2019-07-31 10:11:32

赞 这就是我想找的

顶 (1) 回复 (1)

willlv (楼主) 2019-08-06 20:15:21

@kennyge 获益社区，反哺社区

回复

fazhangliu 2019-07-31 10:44:26

怎么没有“裁撤”

顶 (3) 回复 (1)

willlv (楼主) 2019-07-31 11:29:12

欢迎补充哈，

回复

geraltyang 2019-07-31 11:12:38

答辩专用？不明觉厉

顶 (1) 回复 (1)

willlv (楼主) 2019-08-02 08:44:14

哈哈哈，可以搜索文集：“答辩那些事儿”

回复

victordong 2019-07-31 13:12:34

厉害 

顶 (1) 回复

nofreezou 2019-07-31 13:36:36

666，曾经起过这样的念头~

从墙外传来的新事物，翻译一般都达不到信雅达，比如：

Group A

1. 扩展性（Extensibility），一般是指功能接口层面的可扩展性设计
2. 伸缩性（Scalability），一般是指性能容量层面的扩缩能力

当然既然大家都叫水平扩展，那就随大流喽~

Group B

“一致性哈希”， 一致性这种翻译也是比较容易让人蒙蔽的 101.gif

这类词应该也不少~

顶 (2) 回复

yinshawnrao 2019-07-31 16:47:46

好东西

顶 回复

jamesleeli 2019-07-31 18:48:20

 流弊

顶 回复

saulschen 2019-07-31 19:24:24

nb

顶　　回复

huentinyang　　2019-07-31 21:21:09

流弊

顶　　回复

lipinhuang　　2019-08-02 08:45:12

牛

顶　　回复

yudongcheng　　2019-08-04 14:48:41

牛！学习了！

顶　　回复

roysun　　2019-08-05 11:39:23

对新人不能更友好，学习了~

顶 (1)　　回复

kendywang　　2019-08-05 14:22:05

好文

顶　　回复

zhihaowang　　2019-08-06 10:19:12

专业， 收藏

顶　　回复

lyndonsu　　2019-08-06 17:37:42

mark

顶　回复

andyawang　2019-08-06 17:39:00

好全

顶　回复

johannli　2019-08-06 17:39:24

是真的强

顶　回复

yvesyan　2019-08-06 17:40:57

Word天太厉害了，能不能出一份算法术语大全，跪求了

顶　回复 (1)

willlv (楼主)　2019-08-06 18:41:23

这个可以有，不过算法我不太擅长，要不你试试?

回复

rekiili　2019-08-06 17:47:27

乍看成了后端骗术大全

顶 (1)　回复 (1)

willlv　(楼主)　2019-08-06 20:13:02

@rekiili 独秀，你坐下!

回复

zzbozheng　2019-08-06 17:58:28

每个点都展开，并且写一下具体怎么操作，就很强悍了。

顶　回复 (1)

willlv (楼主)　2019-08-06 20:13:19

@zzbozheng 然后就可以出书了?

回复

ruibaoyang　2019-08-06 18:01:35

就差手把手入门后台开发了

顶　回复 (1)

willlv (楼主)　2019-08-06 20:14:16

@ruibaoyang 手把手音视频编解码

回复

joyyzhang　2019-08-07 14:22:24

Mark

顶　回复

wellingchen　2019-08-07 14:43:18

Mark 太流弊了

顶　回复

v_hbzzhao　2019-08-08 19:40:44

受益匪浅

与缓存击穿不同的是：存击穿是热点key失效；缓存雪崩是大量的key同时失效。

缓存击穿，也就能提点这种书面上的点了 🏀。

顶　回复

philshi　2019-08-29 16:27:02

:thumbsup:

顶　回复

sergiozhang　2019-09-06 14:48:44

牛批牛皮

顶　回复

leafchen　2019-09-12 10:24:36

好文，值得反复查看

顶　回复

leolwei　2019-11-13 18:28:05

好文，好强大

顶　回复

davidning　2020-01-16 16:29:37

66666，这个能够帮助我们把平时遇到的问题梳理和归类，从而去业界寻找最佳实践

顶　回复 (1)

willlv　(楼主)　2020-06-11 11:05:01

@davidning 抛个砖哈

回复

ivanfywang　2020-03-27 17:52:50

很好的文章，新人好希望有个公司内部所有英文简写的解释大全，前期看着KM上一堆持续变更的简写难受得很。

顶　回复 (1)

willlv　(楼主)　2020-06-11 11:05:41

@ivanfywang 哈哈，重在积累

回复

lijanzhuang 2020-05-23 17:22:51

顶 回复

lairayzheng 2020-06-11 10:33:06

阻塞和非阻塞那里，提到“形容多线程间的相互影响”以及“非阻塞允许多个线程同时进入临界区”，这两点让我有些困惑，我还是第一次看到阻塞/非阻塞和多线程联系到一起的。

我的理解是：阻塞发起某个调用，没返回期望的结果（比如读一定长度的数据）就一直等待，而非阻塞发起调用但不能保证结果，返回的是有效数据（不一定读到期望的长度的数据）或空值（可能没读到数据），总之，这一过程是无关多线程的

个人理解，要是有错误还请指出

顶 回复

cheunglong 2020-06-17 01:05:41

了解一下，知己知彼方能百战百胜 

顶 回复

tulliuslin 2020-07-10 20:43:20

（多）线程、（多）进程的经典问题没有：）另外还有金丝雀发布等CICD领域的术语也可以补充。还可以加一些鹅厂后端开发常用的术语，比如名字服务。

顶 回复 (1)

willlv (楼主) 2020-07-10 21:30:07

@tulliuslin 来来，笔给你[emoji]

回复

familyyan 2020-07-11 00:23:27

好东西，明天就背下来

顶 回复 (1)

willlv (楼主) 2020-07-11 22:41:16

你值得拥有

回复

carryxiao 2020-09-10 14:22:18

实用！感觉可以继续补充。产品也能了解。

顶 回复

切换到更多功能

发表评论

[403/463/260 ms]