1. 概述

架构模式

网站架构模式

1. 分层

一般分层主要是分成三层，分别是应用层，服务层，数据层，也就是MVC，

应用层：主要是具体业务或者视图，

服务层：主要是对应用层提供各种服务功能，比如订单服务，购物车服务

数据层：主要提供数据存储功能，比如数据库，缓存，文件存储，搜索

1. 分割

分层是横向的划分，那分割就是纵向的业务划分

将不同的功能和业务进行划分为不同的模块，提供不同的服务，比如登录服务模块，搜索服务模块，不同模块由不同的项目组负责。

1. 分布式

分层和分割之后，便可以就进行分布式部署。分布式主要是适应高并发和大数据量的挑战。

1.分布须要保证各模块之间的通信，如果通信异常，则性能影响比较大。

2. 分布式要求多个服务器同时提供服务，如果单个服务器宕机，会导致整个网站可用向降低

3. 分布式要求数据的高并发读取，这对数据的一致性要求颇高

4. 分布式网站服务依赖错综复杂，开发维护困难

分布式常用类型

1. 分布式应用和服务（复用服务）
2. 分布式静态资源
3. 分布式数据和存储
4. 分布式计算
5. 分布式配置
6. 分布式锁
7. 分布式文件
8. 集群

处于高并发的压力，对独立部署的服务器集群化，多台服务器部署相同的应用构成一个集群，通过负载均衡设备共同对外提供服务。

1. 缓存

缓存就是将数据放在距离计算最近的位置以加快处理速度

CDN：即内容分发网络，用户的请求总是先到达他的最近的网络服务商那里，在那里缓存网站的一些静态资源（较少的变化数据）可以就近以最快的速度返回给用户。

反向代理：属于网站前端架构的一部分，当用户的请求到达网站的数据中心时，最先访问到的就是反向代理服务器，这里可以缓存着网站的静态资源，无需将请求继续转发给应用服务器就能返回给用户

本地缓存：在应用服务器本地缓存着热点数据，应用程序可以在本机内存直接访问数据，而不用访问数据库

分布式缓存：将数据缓存在专门的分布式缓存集群中，应用通过网络通信访问缓存数据。

使用缓存的条件

1. 数据访问热点不均衡，某些数据会更频繁的访问
2. 数据在某个时间段内有效，不会很快过期
3. 异步

业务之间的消息传递不是同步调用，而是将一个业务操作分成多个阶段，每个阶段之间通过共享数据的方式异步执行进行协作。

异步消息队列特征：

1. 提高系统的可用性
2. 加快网站的响应速度
3. 消除并发访问高峰
4. 冗余

数据的冗余备份，当某台服务器宕机时，可以将其上的服务和数据访问转移到其他的机器上。

冷备份，热备份，灾备数据中心

1. 自动化

发布过程自动化

自动化代码管理

自动化测试

自动化安全检测

自动化部署

自动化监控

自动化报警

自动化失效转移

自动化失效恢复

自动化降级

自动化分配资源

1. 安全
2. 通过密码和手机校验码进行身份认证
3. 登录，交易等操作需要对通信进行加密，网站服务器上存储的敏感数据如用户信息等也进行加密处理
4. 对于常见的用于攻击网站的XSS攻击，SQL注入，进行编码转换等相应处理

架构要素

性能

可用性

伸缩性

扩展性

安全性

1. 架构

**高性能网站架构**

网站的性能测试

性能指标

1. 响应时间
2. 并发数（系统负载）
3. 吞吐量：指单位时间内系统处理的请求数量
4. 性能计数器：描述服务器或操作系统性能的一些数据指标

性能测试方法

负载测试

压力测试

稳定性测试

1. 实例

Web前端性能优化

1. 浏览器访问优化

减少http请求

使用浏览器缓存

启动压缩

CSS放在页面最上面，JavaScript放在页面最下面

减少Cookies传输

1. CDN加速

CDN，内容分发网络，本质任然是一个缓存，而且将数据缓存存在离用户最近的地方，使用户以最快的速度获取数据

1. 反向代理

传统代理服务器位于浏览器一侧，代理浏览器将HTTP请求发送到互联网上，而反向代理服务器位于网站机房一侧，代理网站Web服务器接收HTTP请求。

反向代理也可以实现负载均衡的功能，而通过负载均衡构建的应用集群可以提高系统总体的处理能力，进而改善网站高并发情况下的性能

应用服务器性能优化（缓存，集群，异步）

1.分布式缓存（网站性能优化第一定律：优化考虑使用缓存优化性能）

缓存数据=频繁修改的数据+没有热点的访问+数据不一致与脏读

缓存问题：

缓存预热：在缓存系统启动时就把热点数据加载好，这个缓存预加载手段叫做缓存预热

缓存穿透：如果因为不恰当的业务，或者恶意攻击持续高并发的请求某个不存在的数据，由于缓存没有保存改数据，所有的请求都会落到数据库上，会对数据库造成很大的压力，甚至崩溃。

4.异步操作

使用消息队列将调用异步化，可以改善网站的扩展性，事实上，使用消息队列还可以改善网站系统的性能

5.使用集群

在网站高并发访问的场景下，使用负载均衡技术为一个应用构建一个由多台服务器组成的服务器集群。

6.代码优化

1.多线程

将对象设计成无状态对象

使用局部对象

并发访问资源时，使用锁。

2.资源复用

系统运行时，要尽量减少那些开销很大的系统资源的创建和销毁，比如数据库连接，网络通信连接，线程。资源复用两种模式：单例（Singleton）和对象池（Object Pool）

3．’数据结构

4.垃圾回收

JVM分代垃圾回收机制

存储性能优化

1. 机械硬盘和固态硬盘
2. B+树和LSM树

传统数据库使用B+树，B+树是一种专门针对磁盘存储而优化的N叉排序树

许多NoSQL产品采用LSM树，LSM树可以看做N阶合并树。速度快

1. RAID和HDFS

RAID（廉价磁盘冗余阵列）技术主要是为了改善磁盘的访问延迟，增强磁盘的可用性和容错能力，目前服务器级别的计算机都支持插入多块磁盘（8块或者更多），通过使用RAID技术，实现数据在多块磁盘上的并发读写和数据备份

HDFS（Hadoop分布式文件系统）中，系统在整个存储集群的多台服务器上进行数据并发读写和备份，可以看做在服务器集群规模上实现了类似RAID的功能，因此不需要磁盘RAID

**高可用网站架构**

高可用的应用

1.通过负载均衡进行无状态服务的失效转移

负载均衡，主要使用在业务量和数据量较高的情况下，当单台服务器不足以承担所有的负载压力时，通过负载均衡手段，将流量和数据分摊到一个集群组成的多台服务器上，以提高整体的负载处理能力。

1. 应用服务器集群的Session管理

集群环境下，Sesssion管理主要有以下几种手段

1. sesion复制

应用服务器开启Web容器的Session复制功能，在集群中的几台服务器之间同步Session对象。使得每台服务器上都保存着用户的Session信息

优点：读取sesion，快速，简单

缺点：只能在集群规模比较小的情况下，当集群的规模较大时，会占用大力的资源，系统不堪重负。其二，所有的用户的session在每台服务器上都有备份，在大量用户访问的情况下，会出现内存不够的情况。

1. session绑定

session绑定可以利用负载均衡的源地址hash算法实现，负载均衡服务器总是将来源同一IP的请求分发到同一台服务器上（也可以根据Cookies信息将同一个用户的请求总是分发到同一台服务器上，当然这时负载均衡服务器必须在HTTP协议层上），这样在整个会话期间，用户的所有请求都在一个服务器上处理。即sesion绑定在某台特定的服务器上。保证session总能在这台服务器上获取，这种方法又被称为会话黏滞。

缺点：一台服务器宕机，那么session信息就消失了，不可恢复

1. 利用Cookies记录Session

利用浏览器支持的cookie记录session

缺点：受Cookie大小限制，能记录的信息有限

每次请求响应都需要传输cookie，影响性能

如果用户关闭cookie，访问就会不正常。

优点：cookie简单易用，可用性高，支持应用服务器的线性伸缩

1. session服务器

利用独立部署的session服务器（集群）统一管理session，应用服务器每次读写session时，都访问session服务器。

这种解决方案事实上是将服务器的状态分离，分为无状态的应用服务器和有状态的session服务器，然后针对这两种服务器的不同特性分别设计其架构

对于有状态的session服务器，一种比较简单的方法是利用分布式缓存，数据库等，在这些产品的基础上进行包装，使其符合session的存储和访问要求，如果业务场景对session管理有比较高的要求，比如利用session服务集成单点登录（SSO），用户服务等功能，这需要开发专门的session服务管理平台。

高可用的服务

1. 分级管理

运维上将服务器进行分级管理，核心应用和服务优先使用更好的硬件，在运维响应速度上也格外迅速。

1. 超时设置

在应用程序中设置服务调用的超时时间，一旦超时，通信框架就会抛出异常。

1. 异步调用

应用对服务的调用通过消息队列等异步方式完成，避免一个服务失败导致整个应用请求失败的情况。

1. 服务降级

为了保证核心应用和功能在网站高峰期，大量并发调用时，正常运行，需要对服务进行降级，

拒绝服务，关闭功能

1. 幂等设计

在服务层保证服务重复调用和调用第一次产生的结果相同，即服务具有幂等性

高可用的数据

1. CAP原理

数据的持久性

数据的可访问性

数据的一致性

CAP原理认为：一个提供服务的存储系统无法同时满足数据一致性，数据可用性，分区耐受性这三个条件

1. 数据备份

数据冷备：定期将数据复制到某种存储介质（磁带，光盘等）上并物理存档保管，数据系统存储损坏，那么就从冷备的存储设备上恢复数据

优点：简单，廉价成本和技术难度低

缺点：不能保证数据最终一致，由于数据时定期复制，因此备份中的数据比系统中的数据陈旧，同时也不能保证数据可用性，从冷备中恢复数据需要较长的时间，这段时间无法访问数据，系统也不可用

数据热备：异步热备，同步备份

异步方式是指多分数据副本的写入操作异步完成，应用程序收到数据服务系统的操作成功响应时，只写成功一份，存储系统将会异步的写入其他副本（过程可能会失败）

1. 失效转移

如果服务器集群中任何一台服务器宕机，那么应用程序针对这台服务器的所有读写操作都需要重新路由到其他服务器，保证数据访问不会失败，这个过程叫做失效转移。

1. 失效确认

心跳检测和应用程序访问失败报告

在访问失败报告出来后最好用心跳检测一次，确保不会浪费资源

1. 访问转移

确认某台数据服务器宕机以后，就需要将数据读写访问重新路由到其他服务器上。

对等服务器，可以直接切换到对等服务器上

不对等服务器，重新计算路由，选择存储服务器

1. 数据恢复

某台服务器宕机恢复之后要及时将副本的数量恢复到系统设定的值，否则会出现最后无法访问转移的情况。

高可用网站的软件质量保证

1. 网站发布

发布过程中，每次关闭的服务器都是集群中的一小部分，并在发布完成后立即可以访问，因此整个发布过程不影响用户的使用

1. 自动化测试

大型网站通常也会开发自己的自动化测试工具，可以一键完成系统部署，测试数据生成，测试执行，测试报告生成等全部测试过程。

1. 预发布验证

即使经过严格的测试，软件部署到线上服务器之后还是经常会出现各种问题。因此在网站发布时，并不是把测试通过的代码直接发布到线上服务器，而是先发布到预发布机器上，开发工程师和测试工程师在预发布服务器上进行预发布验证，执行典型的业务流程，确认系统没有问题后才正式发布。

1. 代码控制
2. 主干开发，分支发布

代码修改都在主干上进行，需要发布的时候，从主干上拉一个分支发布。该分支即成为一个发布版本，如果该版本发现了bug,继续在该分支上修改发布，并将修改合并回主干，直到下次主干发布。

1. 分支开发，主干发布

任何修改都不能在主干上直接进行，需要开发一个新功能或者修复一个bug时，从主干拉一个分支进行开发，开发完成且测试通过后，合并回主干，然后从主干进行发布，主干上的代码永远是最新发布的版本。

1. 自动化发布

火车发布模型

人的干预越少，自动化程度越高，引入故障的可能性就越小。

1. 灰度发布

灰度发布模式，将集群分成若干个部分，每天只发布一部分服务器，观察运行稳定没有故障，第二天发布一部分服务器，持续几天才把整个集群全部发布完毕，期间如果发现问题，只需要回滚已发布的一部分服务器即可。

灰度发布页常用于用户测试，即在部分服务器上发布新版本，其余服务器保持老版本（或者发布另一个版本），然后监控用户操作行为，收集用户体验报告，比较用户对两个版本的满意度，以确定最终的发布版本，这种手段也被称为AB测试。

1. 网站运行监控

不允许没有监控的系统上线

1. 监控数据采集

用户行为日志收集

服务端日志收集，客户端浏览器日志收集

1. 服务性能监控

收集服务器性能指标，内存占用，磁盘io，网络io，及早做出故障预警。

Ganglia：开源性能监控工具，支持大规模服务器集群，支持以图形操作

1. 运行数据报告

具体业务场景相关的技术和业务指标。缓存命中率。。

监控管理

1. 系统报警

超过阈值，即刻报警

1. 失效转移

系统通知应用，自动失效转移

1. 自动优雅降级

为应对突发的访问高峰，主动关闭部分功能，释放部分系统资源，保证网站核心功能正常访问的一种手段