软件架构案例

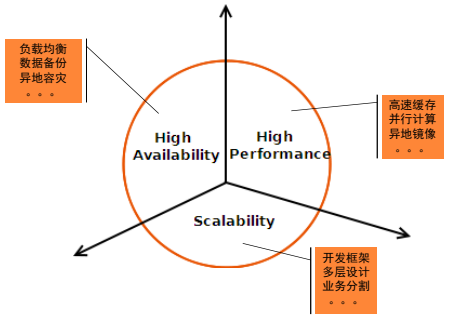
沈备军
上海交通大学软件学院

实例研究：大型网站的架构

- 日均流量IP>1,000,000

网站	日均流量[IP/PV]
www.hao123.com	IP≈ 5,972,587 PV≈ 9,376,962
www.facebook.com	IP≈229,680,000 PV≈2,955,981,600
www.sina.com.cn	IP≈25,680,000 PV≈222,132,000
www.tianya.cn	IP≈5,532,000 PV≈25,723,800
www.pingan.com	*IP≈300,000 PV≈747,000*

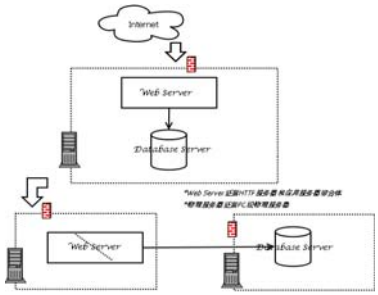
大型网站架构的目标与挑战



每个目标背后面临着技术、设计、维护等诸多方面的挑战。
而目标本身的期望值也会根据实际情况进行调整，这也意味着**网站架构建设是个不断调整的过程。**

大型网站架构演变

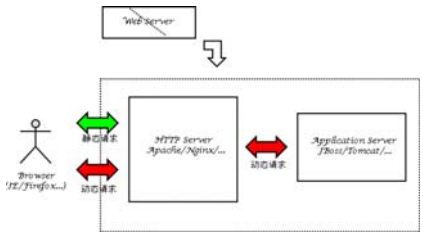
■**[Step1]Web动静态资源分离及其与DB物理分离**



■优点：“**简单**”、安全性提高
■缺点：存在单点，谈不上高可用性（high availability架构目标）
■技术点：应用设计要保证可扩展（framework很重要Spring/Beetle）、Web Server动/静态资源分离
Web Server（Apache\Nginx\IIS\JBoss...）、
Database Server（Mysql\Oracle\Redis...）

大型网站架构演变

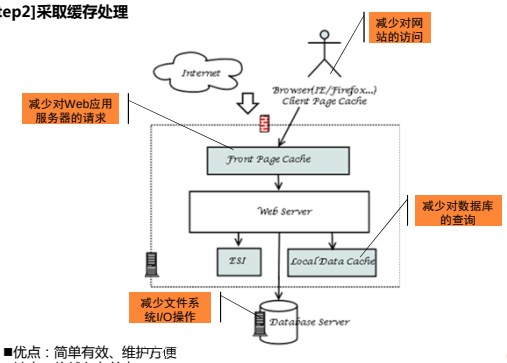
■**[Step1]技术点—Web动静态资源分离**



img,doc,js,css等静态资源使用单独的Web HTTP Server处理请求
动态页面静态化处理

大型网站架构演变

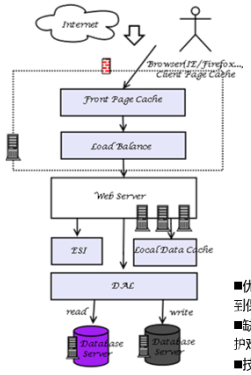
■**[Step2]采取缓存处理**



■优点：简单有效、维护方便
■缺点：依然存在单点
■技术点：客户端（浏览器）缓存、前端页面缓存、页面片段缓存、本地数据缓存/数据库缓存

大型网站架构演变

■[Step3]增加机器做HA、数据库读写分离



■优点：增加服务器和HA机制，系统性能及可用性得到保证
■缺点：读写分离，增加程序难度，架构变复杂，维护难度增加
■技术点：负载均衡、DAL、数据库读写分离

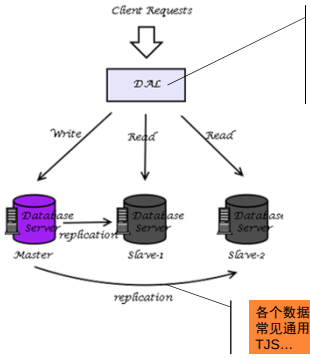
大型网站架构演变

■[Step3]技术点—负载均衡

类型	说明
DNS负载均衡	实现简单，有Cache，缺乏灵活性，但对分区域（如构建CDN方案）访问简单有效
反向代理软件	HAProxy、Nginx、Apache、Lighttpd等
硬件产品	F5、NetScaler等
LVS(Linux Virtual Server)	http://www.linuxvirtualserver.org/
SMART Client	自己写代码某些情况下简单有效

大型网站架构演变

■[Step3]技术点—数据库读写分离及DAL

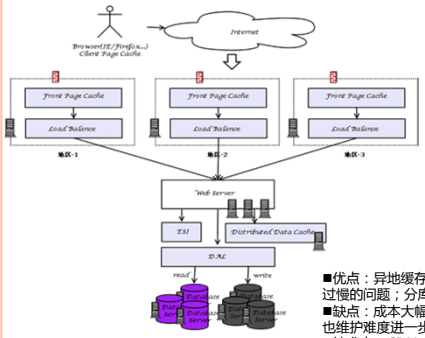


■读写分离逻辑分批
■负载均衡
■失效转移（failover）
■数据库分区透明支持
■两大实现模式：独立Proxy服务器；单独API库文件

各个数据库厂商都有自己复制方案
常见通用方案：ETL、GoldenGate TJS...

大型网站架构演变

■[Step4]CDN、分布式缓存、分库



■优点：异地缓存有效解决不同地方用户访问过慢的问题；分库策略带来网站性能整体提升
■缺点：成本大幅增加，架构进一步复杂化，也维护难度进一步增大，架构开始臃肿了
■技术点：CDN、分布式缓存、Shard分库

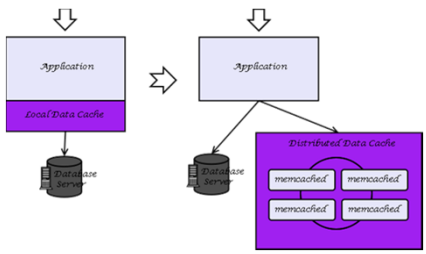
大型网站架构演变

■[Step4]技术点—CDN

■CDN(Content Delivery Network)内容分发网络
■将网站的内容分发到最接近用户的网络“边缘”，使用户可以就近获取，从而解决互联网网络拥挤的状况，提高用户访问的响应速度。
■适合静态内容很多（如：静态页面、图片、视频等）及页面内容实时性要求不高的网站，如：新闻类门户网站
■CDN构建可以做的很简单，也可以很复杂，主要根据自己网站实际情况而定

大型网站架构演变

■[Step4]技术点—分布式缓存



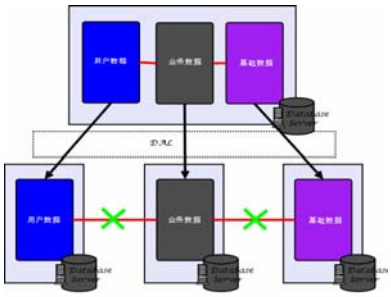
■本地缓存性能优秀，但容量有限，无伸缩性
■采用分布式缓存方案突破容量限制，具备良好伸缩性；但分布式涉及远程网络通信消耗其性能本地缓存来得优秀，并可涉及节点状态维护及数据复制问题，其稳定性和可靠性是个挑战。
■目前流行分布式缓存方案：memcached、membase、redis等，基本上当前的NoSQL方案都可以用来做分布式缓存方案

大型网站架构演变

■**[Step4]技术点—分库**

■读写分离（简单有效，前面已介绍）

■垂直分区

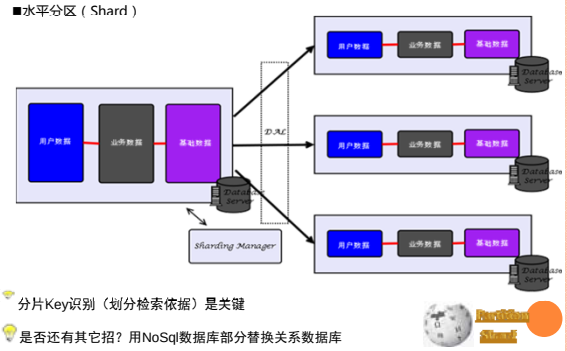


良好的松耦合的模块化设计是垂直分库的前提

大型网站架构演变

■**[Step4]技术点—分库**

■水平分区（Shard）

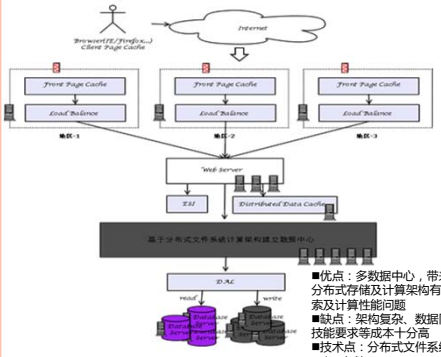


分片Key识别（划分检索依据）是关键

是否还有其它招？用NoSql数据库部分替换关系数据库

大型网站架构演变

■**[Step5]多个数据中心，向分布式存储和计算的架构体系迈进**



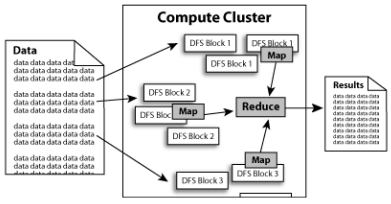
■优点：多数据中心，带来更高质量区域服务体验；分布式存储及计算架构有效解决pb级数据量存储、检索及计算性能问题

■缺点：架构复杂、数据同步、一致性及系统维护、技能要求等成本十分高

■技术点：分布式文件系统、Map/Reduce、Key-Value存储

大型网站架构演变

■**[Step5]技术点—向分布式存储计算解决方案[DFS、Map/Reduce、Key-Value DB]**



■DFS分布式文件系统，如：Lustre\HDFS\GFS\TFS\FreeNas等

■Map/Reduce算法（计算框架），基本上现有NoSQL数据库中都支持此算法。

■Key-Value DB，也作为NoSQL解决方案，如：BigTable\Tair\Hbase\ HyperTable等

■提供完整解决方案：

Google(GFS|Map/Reduce|BigTable)

Apache Hadoop(HDFS|Map/Reduce|HBase)

大型网站架构演变

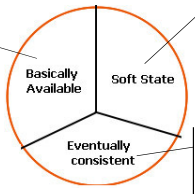
■**关于数据一致性—ACID vs BASE**

■ACID（ Atomicity 、 Consistency 、 Isolation 、 Durability ）是关系型数据库的最基本原则，遵循ACID原则强调一致性，对成本要求很高，对性能影响很大。

■问题：ACID原则适用于互联网应用吗？可用性似乎比一致性重要些

■BASE（ **B**asically **A**vailable 、 **S**oft state 、 **E**ventually consistent ）策略

基本可用
数据能够保证80%一致性就够了，剩下20%就不要过于纠结了。可参考八二定律



软状态
在不过分追求数据一致性（强一致性）前提下考虑软状态策略，例如把数据缓存（State）在客户端一段时间，过后若没有新请求的话，就清除此缓存（Soft）

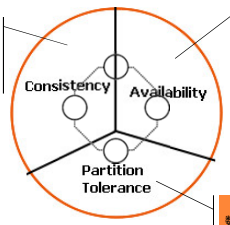
最终一致性
在某一段短时间内允许数据不一致，但经过一段较长时间，等所有节点上数据的拷贝都整合在一起的时候，数据会最终达到完全一致

BASE策略与ACID不同，其基本思想就是通过牺牲强一致性，以获得更好的可用性或可靠性

大型网站架构演变

■**关于分布式系统—CAP理论**

一致性
分布式系统中，数据一般会存储在不同节点，一致性就是要保证对数据操作的原子性



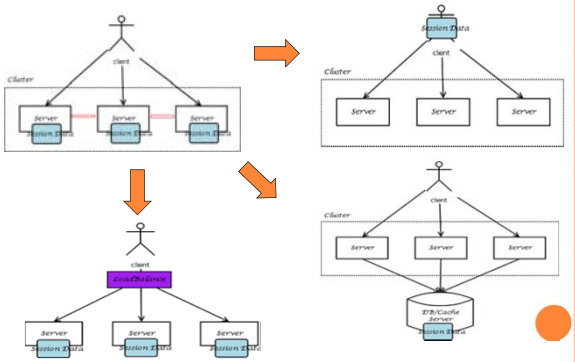
可用性
确保客户访问数据时可得到响应。不强调各个节点上数据要保持一致性。

分区容忍性
数据分区存储后，即使部分分区组件不可用，其施加的操作也能够完成

CAP理论指出：一个分布式系统不可能同时满足一致性、可用性和分区容忍性这三项需求，最多只能同时满足其中两个。

大型网站架构演变

■无共享架构（Share Nothing Architecture）



大型网站架构演变

■ED-SOA架构

■ED-SOA，事件驱动，面向服务架构

■SOA是系统组件化、模块化构建性理论；ED是系统组件之间同步通信，采取事件机制异步化，提高响应速度

■基于ED-SOA构建松耦合系统可以显著改善网站可伸缩性

■架构进化与退化--奥卡姆剃刀原理

■进化—寻找最适合的；退化—简化不必要的

■简单就好，慎防过渡设计

大型网站架构演变

■考量成本，先硬后软原则