读书笔记：【大型网站技术架构】核心原理与案例分析

作者：flhuoshan
原文链接：https://ld246.com/article/1486559410777
来源网站：链滴

<blockquote>
前言：春节前读了这本书，原计划是春节假期就把读书笔记整理好，后来眼睛不舒服，未遂。节经过几个晚上的整理，现将内容发上来供参考。注意本文内容只有核心原理，没有案例分析，因为设案例分析，必须画图，否则讲不清，容以后再写。
</blockquote>
<h3 id="第一章-大型网站架构演化">第一章：大型网站架构演化</h3>
<blockquote>
<h4 id="1-1大型网站软件系统的特点-高并发-大流量-高可用-海量数据-用户分布广泛-网络情况复杂安全环境恶劣-需求快速变更-发布频繁-渐进式发展-">1.1 大型网站软件系统的特点：高并发、大流，高可用，海量数据，用户分布广泛、网络情况复杂，安全环境恶劣，需求快速变更、发布频繁，渐式发展。</h4>
</blockquote>
<blockquote>
<h4 id="1-2大型网站架构演化发展历程">1.2 大型网站架构演化发展历程</h4>
1.2.1 初始阶段：应用、数据库、文件在同一台服务器上。
1.2.2 应用服务和数据服务分离
1.2.3 使用缓存改善网站性能，二八定理 80% 的业务集中在 20% 的数据上，缓存又分为本地缓存和布式缓存
1.2.4 应用服务器集群
1.2.5 数据库读写分离
1.2.6 使用反向代理和 CDN 加速网站响应
1.2.7 使用分布式文件系统和分布式数据库
1.2.8 使用 NoSql 和搜索引擎
1.2.9 业务拆分
1.2.10 分布式服务
</blockquote>
<h3 id="第二章-大型网站架构模式">第二章：大型网站架构模式</h3>
<h4 id="模式-经过验证的-描述并解决某一类问题的固有方案-">模式：经过验证的，描述并解决某类问题的固有方案。</h4>
<blockquote>
<h4 id="2-1网站架构模式">2.1 网站架构模式</h4>
</blockquote>
2.1.1 分层：将系统在横向维度上切分，每个部分负责相对单一的职责，通过上层对下层的依赖调用组成一个完整的系统。在网站系统中，可以将系统从上至下分为应用层、服务层、数据层。
2.1.2 分割：将系统在纵向维度上切分，将不同的功能切分成相对独立的模块。
2.1.3 分布式：包括分布式应用和服务、分布式静态资源、分布式数据和存储、分布式计算、分布式件系统等。
2.1.4 集群：将多台服务器部署相同的应用来构成一个集群，并通过负载均衡技术实现对外服务。<br

2.1.5 缓存：这里不仅仅指内存的缓存，还有内容的缓存。包括 CDN，反向代理、本地缓存、分布式存等。
2.1.6 异步：可用分布式消息队列实现。
2.1.7 冗余：主要用于灾备
2.1.8 自动化：主要是发布过程的自动化。
2.1.9 安全：主要使用密码，验证码，加密，网络攻击编码转码，垃圾信息过滤，敏感信息风控等。</
>
<h3 id="第三章-大型网站核心架构要素">第三章：大型网站核心架构要素</h3>
<blockquote>
<h4 id="架构--最高层次的规划-难以改变的决定--这些规划和决定奠定了事物未来发展的方向和最的蓝图">架构：“最高层次的规划，难以改变的决定”。这些规划和决定奠定了事物未来发展的方向最终的蓝图</h4>
<h4 id="软件架构--有关软件整体结构与组件的抽象描述-用于指导大型软件系统各个方面的设计--">软件架构：“有关软件整体结构与组件的抽象描述，用于指导大型软件系统各个方面的设计”。</h4

<h4 id="软件架构关注以下五个要素-性能-可用性-伸缩性-扩展性-安全性-">软件架构关注以下五个素：性能，可用性，伸缩性，扩展性，安全性。</h4>
</blockquote>
<h3 id="第四章-瞬时响应-网站的高性能架构">第四章：瞬时响应：网站的高性能架构</h3>
<blockquote>
<h4 id="4-1网站性能测试-性能测试是为性能优化的前提和基础-也是性能优化结果的检查和度量标准不同视角下的网站性能有不同的标准-也有不同的优化手段-">4.1 网站性能测试：性能测试是为性能化的前提和基础，也是性能优化结果的检查和度量标准。不同视角下的网站性能有不同的标准，也有同的优化手段。</h4>
</blockquote>
4.1.1 不同视角下的网站性能
4.1.1.1 用户视角的网站性能：用户感受到的时间，包括用户计算机和网站服务器通信的时间、网站服器处理的时间、用户计算机浏览器构造请求解析响应数据的时间。优化策略：优化 html、调整浏览缓存策略、使用 cdn 服务、反向代理等。
4.1.1.2 开发人员视角的网站性能：关注响应延迟、系统吞吐量、并发处理能力、系统稳定性等指标。化策略：缓存加速数据读取、集群提高吞吐能力、异步消息加快请求响应及实现削峰、代码优化等。br>
4.1.1.3 运维人员视角：关注基础设施性能、资源利用率。优化手段：建设优化骨干网、使用高性价比制服务器、利用虚拟化技术优化资源利用率。
4.1.2 性能测试指标：响应时间，并发数、吞吐量、性能计数器。
4.1.3 性能测试方法：性能测试、负载测试、压力测试、稳定性测试
<h4 id="4-2-Web前端性能优化">4.2 Web 前端性能优化</h4>
4.2.1 浏览器访问优化：减少 http 请求（合并 css,Javascript,图片等）、使用浏览器缓存、启用缩、css 放在最上面且 JavaScript 放在页面最下面、减少 cookie 传输。
4.2.2CDN 加速
4.2.3 反向代理：反向代理三作用：安全功能、缓存加速、应用层负载均衡。
<h4 id="4-3应用服务器性能优化-应用服务器就是处理网站业务的服务器-网站的业务代码都部署在里-是网站开发最复杂-变化最多的地方-优化主要手段有缓存-集群-异步等-">4.3 应用服务器性能优：应用服务器就是处理网站业务的服务器，网站的业务代码都部署在这里，是网站开发最复杂，变化多的地方，优化主要手段有缓存、集群、异步等。</h4>
4.3.1 分布式缓存：网站性能优化第一定律：优先考虑使用缓存优化性能。缓存的本质是一个内存 Hash 表，关于 Hash 表的实现和原理可以单独开一个话题说了，缓存主要用来存放读写比很高、变很少的数据，二八定律：80% 的访问落在 20% 的数据上。
4.3.1.2 合理使用缓存：频繁修改的数据、没有热点的访问、注意数据不一致与脏读、缓存可用性、缓预热、缓存穿透。
4.3.1.3 分布式缓存架构：JBoss Cache 为代表的需要更新同步的分布式缓存、以 Memocached 为表的户不通信的分布式缓存（其原理核心在于一致性 Hash 等路由算法）。
4.3.2 异步：异步的思路核心在于消息队列，在传统的应用服务器和数据库服务器中间加入了消息队服务器以实现异步功能，削峰利器，广泛应用于秒杀业务，双十一剁手活动等。任何可以晚点做的事都应该晚点做。
4.3.3 使用集群：负载均衡技术
4.3.4 代码优化
4.3.4.1 多线程：对网站而言，不管有没有进行多线程编程，工程师写的每一行代码都会被多线程执行因为用户的请求是并发提交的，也就是说，所有的资源，包括对象、内存、文件、数据库，乃至另一线程都可能被多线程并发访问。
4.3.4.1.1 将对象设计为无状态对象：指对象本身不存储状态信息（对象无成员变量或者成员变量也是状态对象），这样就不会多线程并发访问的时候就不会出现状态不一致的问题，Servlet 便被设计成状态对象。
4.3.4.1.2 使用局部对象
4.3.4.1.3 并发访问资源时使用锁
4.3.4.2 资源复用：单例、对象池（连接池、线程池）
4.3.4.3 使用良好的数据结构
4.3.4.4 垃圾回收：合理设置 Young Generation 和 Old Generation 大小，减少 Full GC。
<h4 id="4-4存储性能优化-Raid-HDFS等">4.4 存储性能优化：Raid，HDFS 等</h4>

<h3 id="第五章-万无一失-网站的高可用架构">第五章：万无一失：网站的高可用架构</h3>
<blockquote>
<h4 id="5-1网站可用性的度量与考核">5.1 网站可用性的度量与考核</h4>
<h4 id="5-2高可用的网站架构-大型网站的分层架构及物理服务器的分布式部署使得位于不同层次的务器具有不同的可用性特点-关闭服务器或者宕机时产生的影响也不相同-高可用的解决方案也差异巨大">5.2 高可用的网站架构：大型网站的分层架构及物理服务器的分布式部署使得位于不同层次的服务具有不同的可用性特点。关闭服务器或者宕机时产生的影响也不相同，高可用的解决方案也差异巨大</h4>
<h4 id="5-3高可用的应用">5.3 高可用的应用</h4>
</blockquote>
5.3.1 通过负载均衡实现对无状态服务的失效管理
5.3.2 对于有状态情况的方案：Session 复制（不适合大规模集群，因巨量 session），Session 绑定会话粘滞，不能当机，否则导致不可用），使用浏览器本地的 cookie 记录 session，单独部署公共的 Session 服务器处理 session。
<blockquote>
<h4 id="5-4高可用服务策略-分级管理-优先级--超时设置-失效转移--异步调用-消息队列--服务降级幂等性设计-调用多次和调用一次结果相同--">5.4 高可用服务策略：分级管理（优先级），超时设置失效转移），异步调用（消息队列），服务降级，幂等性设计（调用多次和调用一次结果相同）。</4>
<h4 id="5-5高可用数据">5.5 高可用数据</h4>
</blockquote>
5.5.1CAP 原理：一个提供数据服务的存储系统无法同时满足数据一致性，数据可用性，分区耐性这三个条件。
5.5.2 数据备份：冷备份及热备份，热备份又分为异步及同步热备。
5.5.3 失效转移：三部分：失效确认、访问转移、数据恢复。
<blockquote>
<h4 id="5-6高可用网站的软件质量保证-网站发布-自动化测试-预发布验证-代码控制-1-主干开发-支发布-代码修改都在主干上进行-需要发布的时候-从主干拉一个分支发布-该分支即成为一个发布版本如果该版本发现bug-继续在该分支上修改发布-并将修改合并到主干-直到下次主干发布-2-分支发布-干开发-任何修改都不得在主干上直接进行-需要开发一个新功能或修改bug时-从主干拉一个分支进行发-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--动化发布-灰度发布-">5.6 高可用网站的软件质量保证：网站发布、自动化测试、预发布验证、代码制（1、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 bug,继续在该分支上修改发布，并将修改合并到主干直到下次主干发布。2、分支发布，主干开发：任何修改都不得在主干上直接进行，需要开发一个新能或修改 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，然后从主干行发布，主干上的代码永远是最新发布的版本）、自动化发布、灰度发布。</h4>
<h4 id="5-7网站运行监控--不允许没有监控的系统上线--">5.7 网站运行监控：“不允许没有监控系统上线”。</h4>
</blockquote>
5.7.1 监控数据采集：用户行为日志收集、服务器性能监控、运行数据报告。
5.7.2 监控管理：系统报警、失效转移、自动优雅降级。
<h3 id="第六章-永无止境-网站的伸缩性架构">第六章：永无止境：网站的伸缩性架构</h3>
<blockquote>
<h4 id="网站的伸缩性是指不需要改变网站的软硬件设计-仅仅通过改变部署的-服务器数量就可以扩或缩小网站的服务处理能力-">网站的伸缩性是指不需要改变网站的软硬件设计，仅仅通过改变部署的服务器数量就可以扩大或缩小网站的服务处理能力。</h4>
<h4 id="6-1网站的伸缩性设计-一类是根据功能进行物理分离实现伸缩-第二是单一功能通过集群实伸缩-">6.1 网站的伸缩性设计：一类是根据功能进行物理分离实现伸缩，第二是单一功能通过集群实伸缩。</h4>
</blockquote>
6.1.1 不同功能进行物理分离实现伸缩：单一服务器处理所有服务-> 数据库冬应用服务器分离 > 缓存从应用服务器分离-> 静态资源从应用服务器分离。
<blockquote>

<h4 id="纵向分离-分层后分离--将业务处理流程上的不同部分分离部署-实现系统伸缩性-">纵向分（分层后分离）：将业务处理流程上的不同部分分离部署，实现系统伸缩性。</h4>
<h4 id="横向分离-业务分割后分离--将不同的业务模块分离部署-实现系统伸缩性-">横向分离（业分割后分离）：将不同的业务模块分离部署，实现系统伸缩性。</h4>
</blockquote>
6.1.2 单一功能通过集群规模实现伸缩。
6.2 应用服务器集群的伸缩性设计：Http 重定向负载均衡，DNS 域名解析负载均衡，反向代理负载衡（应用层），IP 负载均衡（网络层）、数据链路层负载均衡。
<blockquote>
<h4 id="负载均衡算法-轮询-加权轮询-随机-虽少连接-源地址散列-哈希--">负载均衡算法：轮询、权轮询、随机、虽少连接，源地址散列（哈希）。</h4>
<h4 id="6-3分布式缓存集群的伸缩性设计-分布式缓存的一致性哈希算法-使用虚拟结点的一致性has环--">6.3 分布式缓存集群的伸缩性设计：分布式缓存的一致性哈希算法（使用虚拟结点的一致性 has 环）。</h4>
<h4 id="6-4数据存储服务器集群的伸缩性设计">6.4 数据存储服务器集群的伸缩性设计</h4>
</blockquote>
6.4.1 关系数据库集群的伸缩性设计：读写分离，分库（在分库的的技术上对单表大数据进行切，数据分布到多个数据库中），mysql 集群伸缩使用一致性 hash 算法进行数据迁移（集群扩容时）
6.4.2NoSql 数据库集群的伸缩性设计
<h3 id="第七章-随需应变-网站的可扩展架构">第七章：随需应变：网站的可扩展架构</h3>
<blockquote>
<h4 id="扩展性-指对现有系统影响最小的情况下-系统功能可持续扩展或提升的能力-表现在系统基设施稳定不需要经常变更-应用之间较少依赖和耦合-对需求变更可以敏捷响应-是系统设计层面的开闭则-对扩展开放-对修改关闭--架构设计考虑未来功能扩展-当系统增加新功能时-不需要对现有的结构代码进行修改-">扩展性：指对现有系统影响最小的情况下，系统功能可持续扩展或提升的能力。表在系统基础设施稳定不需要经常变更，应用之间较少依赖和耦合，对需求变更可以敏捷响应。是系统计层面的开闭原则（对扩展开放，对修改关闭），架构设计考虑未来功能扩展，当系统增加新功能时不需要对现有的结构和代码进行修改。</h4>
<h4 id="伸缩性-指系统能够通过增减自身资源规模的方式增减自己计算处理事务的能力-如果这种增是成比例的-就被称作线性伸缩性-在网站架构中-通常指利用集群的方式增加服务器数量-提高系统的务吞吐能力-">伸缩性：指系统能够通过增减自身资源规模的方式增减自己计算处理事务的能力。如这种增减是成比例的，就被称作线性伸缩性。在网站架构中，通常指利用集群的方式增加服务器数量提高系统的事务吞吐能力。</h4>
<h4 id="7-1构建可扩展的网站结构-设计网站可扩展架构的核心思想是模块化-并在此基础上-降低模间的耦合性-提高模块的复用性-">7.1 构建可扩展的网站结构：设计网站可扩展架构的核心思想是模化，并在此基础上，降低模块间的耦合性，提高模块的复用性。</h4>
<h4 id="7-2利用分布式消息队列降低系统耦合性-事件驱动架构-分布式消息队列-先进先出-分布式步调用-">7.2 利用分布式消息队列降低系统耦合性：事件驱动架构，分布式消息队列（先进先出，分式异步调用）</h4>
<h4 id="7-3利用分布式服务打造可复用的业务平台-WebService">7.3 利用分布式服务打造可复用业务平台：WebService</h4>
<h4 id="7-4可扩展的数据结构">7.4 可扩展的数据结构</h4>
</blockquote>
<h3 id="第八章-固若金汤-网站的安全架构">第八章：固若金汤：网站的安全架构</h3>
<blockquote>
<h4 id="8-1道高一尺魔高一丈的网站应用攻击与防御">8.1 道高一尺魔高一丈的网站应用攻击与防</h4>
</blockquote>
8.1.1XSS 攻击（跨站脚本攻击），指黑客通过篡改网页，注入恶意 HTML 脚本，在用户浏览网时，控制用户浏览器进行恶意操作的工具方式。应对手段：消毒，HttpOnly（防止 xss 窃取 cookie。
8.1.2 注入攻击
8.1.3CSRF 攻击

8.1.4 其他攻击和漏洞
8.1.5Web 应用防火墙
8.1.6 网站安全漏洞扫描
<blockquote>
<h4 id="8-2信息加密技术及密钥安全管理">8.2 信息加密技术及密钥安全管理</h4>
</blockquote>
8.2.1 单向散列加密：如 MD5，SHA，还可以基于这些算法进行“加盐”，相当于加密的密钥。
8.2.2 对称加密：加解密使用同一个密钥，有 DES，RC 算法等。
8.2.3 非对称加密：一个公钥一个私钥，公钥加密后只有私钥可以打开，反之一样，RSA。
<blockquote>
<h4 id="8-3信息过滤与反垃圾-文本匹配-正则-多级Hash表-双数组Tire--分类算法-贝叶斯分类算法-黑名单-">8.3 信息过滤与反垃圾：文本匹配（正则，多级 Hash 表，双数组 Tire）,分类算法（贝叶分类算法），黑名单。</h4>
</blockquote>

 该文章同步自 B3log 社区