YouTube 网站的架构演进

21CTO 2017-02-26

概述

YouTube 在国内是个404网站，需要翻墙得见，这是有用的废话，先铺垫一下。

从全球网站来看，它仅次于母公司 Google，全球排名位列第2。每天超过5亿以上视频播放量，平均每个用户点击10-15个视频。

就是这样一个巨大型网站，却只有很少的人在维护网站，保证其可用性和可伸缩性。
是什么原因呢？肯定不是Google舍不得花钱建团队，也不能只靠人品，但也不能靠寂寞。

那么到底是什么呢？接下来我们就来了解 YouTube的基础技术架构。

基础平台

- Apache
- Python
- Linux(SuSe)

- MySQL
- psyco，一个动态的Python到C的编译器
- lighttpd代替Apache做视频播放

状态

- 支持每天超过5亿的视频点击量
- 由美籍华人陈士骏（生于1978年）创立于2005年2月
- 于2006年3月达到每天3千万的视频点击量
- 于2006年7月达到每天1亿的视频点击量
- 于2016年3月达到每天5亿+的视频点击量
- 2个系统管理员，2个伸缩性软件架构师
- 2个软件开发工程师，2个网络工程师，1个DBA以（在2010年左右的人数）。

Web服务器

1、使用 NetScaler 做为负载均衡和静态内容缓存
2、使用mod_fast_cgi 运行Apache服务器
3、使用一个Python应用服务器来处理请求的路由
4、应用服务器与多个数据库和其他信息源交互来获取数据和格式化html页面
5、一般可以通过添加更多的机器来在Web层提高伸缩性
6、Python的Web层代码通常不是性能瓶颈，大部分时间阻塞在RPC层
7、Python允许快速而灵活的开发和部署
8、通常每个页面服务少于100毫秒的时间
9、使用psyco(一个类似于JIT编译器的动态的Python到C的编译器)来优化内部循环
10、对于像加密等密集型CPU活动，使用C扩展
11、对于一些开销昂贵的块使用预先生成并缓存的html
12、数据库里使用行级缓存
13、缓存完整的Python对象（类似于php中的OpCode或Java的ByteCode）
14、有些数据被计算出来并发送给各个程序，所以这些值缓存在本地内存中—注：这个策略有些使用不当。

应用服务器里最快的缓存将预先计算的值发送给所有服务器也花不了多少时间，使用一个代理来监听更改，预计算，然后再发送。

视频服务

1，花费包括带宽，硬件和电力消耗

2，每个视频由一个小的服务器集群来处理，每个视频都是多机机器提供数据

3，使用一个集群意味着：

- 更多的硬盘来保存视频内容，提高更快的速度
- 高可用与灾难恢复。或一台机器出现故障，其它机器可以继续服务
- 在线备份

4，使用lighttpd作为Web服务器来提供视频服务：

- Apache开销太大
- 使用epoll来等待多个FDS
- 从单进程配置转变为多进程配置来处理更多的连接
- 后来从lighthttpd之后换为Nginx，显示为YouTubeFrontEnd，具体是什么未知

5，大部分流行的内容转移至CDN：

- CDN在多个地方缓存内容，这样内容离用户更近的机会就会更高
- CDN机器经常内存不足，因为内容读取频繁，会出现内存与外存的交换瓶颈，即内存颠簸

6，一些较冷的内容(每天1-20浏览量)，外部链接使用YouTube服务

- 长尾效应。一个视频可以有多个播放，但是许多视频正在播放。随机硬盘块被访问
- 在这种情况下缓存不会很好，所以花钱在更多的缓存上可能没太大意义。
- 调节RAID控制并注意其他低级问题
- 调节每台机器上的内存，不要太多也不要太少

视频服务架构关键点

1、保持简单和低成本

2、保持简单网络拓扑，内容和用户之间不要有太多路由

3、使用常用硬件，使用昂贵的硬件找到帮助文档也不易

4、使用简单常见的工具，构建在Linux平台，以及其上的工具

5、很好的处理随机查找(SATA，tweaks)

缩略图服务

1，做到处理最高效

2，每个视频要生成4张缩略图，所以缩略图比视频多很多

3，缩略图只保存在几台机器上

4，持有一些小东西所遇到的问题：

- OS级别的大量的硬盘查找和inode和页面缓存问题
- 单目录文件限制，特别是Ext3，后来移到多分层的结构。
- 内核2.6的最近改进可能让 Ext3允许大目录，但在一个文件系统里存储大量文件肯定不是个好主意

- 每秒大量的同步请求——Web页面可能在页面上显示60几个缩略图
- 在这种高负载下Apache表现的非常糟糕
- 在Apache前端加入了squid。这种方式工作了一段时间，但是由于负载继续增加而以失败告终，虽然它让每秒300个请求变为20个
- 尝试使用lighttpd但是由于使用单线程它陷于困境。遇到多进程的问题，因为它们各自保持自己单独的缓存
- 如此多的图片以致一台新机器只能接管24小时
- 重启机器需要6-10小时来缓存

5，为解决以上问题，YouTube开始使用Google的BigTable—一个分布式数据存储：

- 避免小文件问题，因为它将文件收集到一起
- 快，容错率高
- 较低的延迟。使用分布式多级缓存，缓存与多个不同collocation站点工作
- 更多信息请大家参考Google Architecture，GoogleTalk Architecture和BigTable相关资料

> **数据库**

1，早期

- 使用MySQL来存储元数据，如用户，标签（tags）和视频文字介绍、评论信息
- 使用一个RAID 10的磁盘阵列来存储数据
- YouTube经过一个常见系统的架构演进：
  从单服务器开始，然后单master和多read slave，接着做数据库partition分区，然后再hash sharding方式
- 备份慢的痛苦。master数据库是多线程的并且运行在一个大型机上，可以处理许多工作；slave是单线程的且运行在小一些的服务器上，备份是异步的，所以slave会远远慢于master主机
- 更新引起缓存失效，硬盘的I/O缓慢导致备份迟延
- 使用备份架构花费不少钱来增加写的性能
- YouTube解决方案把数据分成两个集群来将传输分出优先次序：一个视频查看的数据库集群，另一个是处理其它业务的集群。

2，后期

- 数据库分区
  分成shardings，不同的用户被分发到不同的sharding
- 扩散读写
- 更好的缓存位置意味着更少的I/O
  硬件设备减少30%
- 备份延迟降低到0

- 到现在，可以任意提升数据库的伸缩性

数据中心策略

1、信用卡与支付处理，受控于支付方的主机提供商
受管主机提供商不能提供可伸缩性，亦不能控制硬件，使用更好的协议。后改为托管安置（colocation arrangement）。现在YouTube可以自定义所有模块并且制定自己的协议
2、扩展到十几个自有IDC数据中心，以及CDN服务器
3、视频内容由任意的IDC提供，没有经过最近地址匹配。如果一个视频很流行则会被分发到CDN来承载。
4、视频速度依赖于带宽而非真正的延迟
5、遇图片加载延迟严重，场景是是当一个页面有60几张图片时
6、使用BigTable将图片备份到不同的数据中心，由代码逻辑定位哪张离用户最近

小结

1、Stall for time。创新和敢为让你在短期内解决问题，需要找到长期的解决方案
2、Proioritize。找出你的服务中核心的东西，并对资源拆分，划分优先级
3、Pick your battles。无需担心将核心服务分出去。
Youtube使用CDN来分布最流行的内容。如果都多地自建IDC机房，将需要很长时间和较高的花费
4、Keep it simple。保持简单，以此原则迭代架构，来响应出现的问题
5，ShardIng。分布式架构帮助我们隔离存储，CPU，内存和IO设备的负载，不仅仅是获得更多写的性能。
6，系统瓶颈的持续迭代：

- 软件：DB，缓存
- OS：硬盘I/O
- 硬件：内存，RAID

就是这样的架构支撑着Youtube的巨大流量，我们会继续跟踪它的架构演进。如果你知道更多，欢迎留言&投稿。

> 作者：21CTO社区
> 说明：由社区整理翻译，有部分更新与内容增加。
> 来源：http://highscalability.com/youtube-architecture

阅读原文