一座城堡

蓝色

日志

[转]Google Architecture [转]Scaling Twitter: Making Twitter 1000 Percent Faster

[转]YouTube Architecture

2008-04-02 12:34:04 | 分类： 大公司技术 订阅 | 字号

YouTube发展迅速，每天超过1亿的视频点击量，但只有很少人在维护站点和确保伸缩性。

平台

Apache
Python
Linux(SuSe)
MySQL
psyco，一个动态的Python到C的编译器
lighttpd代替Apache做视频查看

状态

支持每天超过1亿的视频点击量
成立于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
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

处理飞速增长的流量

```
while (true) { identify_and_fix_bottlenecks(); drink(); sleep(); notice_new_bottleneck(); }
```

每天运行该循环多次

Web服务器

1，NetScaler用于负载均衡和静态内容缓存
2，使用mod_fast_cgi运行Apache
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
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
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。这是个使用不当的策略。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。只需弄一个代理来监听更改，预计算，然后发送。

视频服务

1，花费包括带宽，硬件和能源消耗
2，每个视频由一个迷你集群来host，每个视频被超过一台机器持有

3，使用一个集群意味着：
-更多的硬盘来持有内容意味着更快的速度
-failover。如果一台机器出故障了，另外的机器可以继续服务
-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
5，大部分流行的内容移到CDN：
-CDN在多个地方备份内容，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，因为内容太流行以致很少有内容进出内存的颠簸
6，不太流行的内容(每天1-20浏览次数)在许多colo站点使用YouTube服务器
-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

视频服务关键点
1，保持简单和廉价
2，保持简单网络路径，在内容和用户间不要有太多设备
3，使用常用硬件，昂贵的硬件很难找到帮助文档
4，使用简单而常见的工具，使用构建在Linux里或之上的大部分工具
5，很好的处理随机查找(SATA，tweaks)

缩略图服务
1，做到高效令人惊奇的难
2，每个视频大概4张缩略图，所以缩略图比视频多很多
3，缩略图仅仅host在几个机器上
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不是个好主意
-每秒大量的请求，因为Web页面可能在页面上显示60个缩略图
-在这种高负载下Apache表现的非常糟糕
-在Apache前端使用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。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，为了解决所有这些问题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，错误容忍
-更低的延迟，因为它使用分布式多级缓存，该缓存与多个不同collocation站点工作
-更多信息参考Google Architecture，GoogleTalk Architecture和BigTable

数据库
1，早期
-使用MySQL来存储元数据，如用户，tags和描述
-使用一整个10硬盘的RAID 10来存储数据
-依赖于信用卡所以YouTube租用硬件
-YouTube经过一个常见的革命：单服务器，然后单master和多read slaves，然后数据库分区，然后sharding方式
-痛苦与备份延迟。master数据库是多线程的并且运行在一个大机器上所以它可以处理许多工作，slaves是单线程的并且通常运行在小一些的服务器上并且备份是异步的，所以slaves会远远落后于master
-更新引起缓存失效，硬盘的慢I/O导致慢备份
-使用备份架构需要花费大量的money来获得增加的写性能
-YouTube的一个解决方案是通过把数据分成两个集群来将传输分出优先次序：一个视频查看池和一个一般的集群
2，后期
-数据库分区
-分成shards，不同的用户指定到不同的shards

-扩散读写
-更好的缓存位置意味着更少的IO
-导致硬件减少30%
-备份延迟降低到0
-现在可以任意提升数据库的伸缩性

数据中心策略

1，依赖于信用卡，所以最初只能使用受管主机提供商
2，受管主机提供商不能提供伸缩性，不能控制硬件或使用良好的网络协议
3，YouTube改为使用colocation arrangement。现在YouTube可以自定义所有东西并且协定自己的契约
4，使用5到6个数据中心加CDN
5，视频来自任意的数据中心，不是最近的匹配或其他什么。如果一个视频足够流行则移到CDN
6，依赖于视频带宽而不是真正的延迟。可以来自任何colo
7，图片延迟很严重，特别是当一个页面有60张图片时
8，使用BigTable将图片备份到不同的数据中心，代码查看谁是最近的

学到的东西

1，Stall for time。创造性和风险性的技巧让你在短期内解决问题而同时你会发现长期的解决方案
2，Proioritize。找出你的服务中核心的东西并对你的资源分出优先级别
3，Pick your battles。别怕将你的核心服务分出去。YouTube使用CDN来分布它们最流行的内容。创建自己的网络将花费太多时间和太多money
4，Keep it simple！简单允许你更快的重新架构来回应问题
5，Shard。Sharding帮助隔离存储，CPU，内存和IO，不仅仅是获得更多的写性能
6，Constant iteration on bottlenecks：
-软件：DB，缓存
-OS：硬盘I/O
-硬件：内存，RAID
7，You succeed as a team。拥有一个跨越条律的了解整个系统并知道系统内部是什么样的团队，如安装打印机，安装机器，安装网络等等的人。With a good team all things are possible。

转自： http://hi.baidu.com/sujun/blog/item/33d29c3dcbd45f04baa16745.html
http://highscalability.com/youtube-architecture

[转]Google Architecture
[转]Scaling Twitter: Making Twitter 1000 Percent Faster