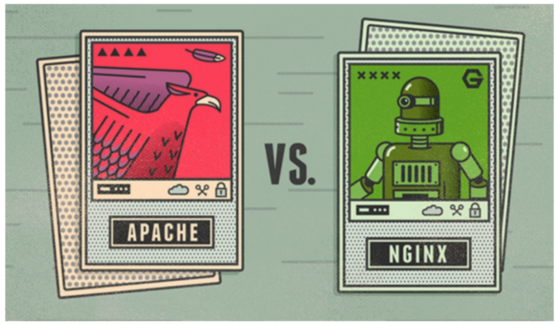

网络编程（一）：演进——从Apache到Nginx

auxten
CovenantSQL 联合创始人

Apache

Apache HTTP服务器是 Robert McCool 在1995年写成，并在1999年开始在Apache软件基金会的 框架下进行开发。由于Apache HTTP服务器是基金会最开始的一个项目也是最为有名的一个项

行 很大程度上是由于相比其他的软件项目，在Apache基金会的精心维护下他的文档十分的详尽还有 集成的支持服务。

Apache由于其可变性、高性能和广泛的支持，经常是系统管理员的首选。他可以通过一系列 的语言相关的扩展模块支持很多解释型语言的后端，而不需要连接一个独立的后端程序。

Apache软件基金会也是利用开源软件盈利的一个范本。时至今日，Apache软件基金会 已经枝繁叶茂，在基金会名下的开源项目我们耳熟能详的有：

- Apache HTTP Server
- Ant（Java的编译工具）
- ActiveMQ（MQ集群）
- Cassandra（强一致的分布式KV数据库）
- CloudStack（OpenStack的劲敌）
- CouchDB（KV数据库）
- Flume（日志收集工具）
- Hadoop、Hbase、Hive
- Kafka（流式计算）
- Lucene（开源搜索引擎）
- Maven（Java编译&依赖管理工具）
- Mesos（分布式协调）
- OpenNLP（开源自然语言处理库）
- OpenOffice（开源的类Office工具）
- Perl（Perl语言）
- Spark（分布式计算集群）

- Subversion（SVN，你懂的）
- Tcl（Tcl语言）
- Thrift（Java网络框架）
- Tomcat（大名鼎鼎的Java容器）
- ZooKeeper（分布式协调集群）

完整的Apache基金会的项目列表参见：Welcome to The Apache Software Foundation!

Nginx

2002年，一个叫Igor Sysoev的俄罗斯哥们儿（貌似俄罗斯叫Igor的人挺多的） 写出了一个叫Nginx（和Engine X谐音，取引擎之义）。 那时候有一个时代背景，当时C10K（Concurrency 10K，1万并发）问题还是困扰绝大多数 web服务器的一个难题。Nginx利用异步事件驱动的架构写成，是C10K问题的一个很好的答卷。 Nginx的第一个公开发行版是在2004年发布的，之前都是作为俄罗斯访问量第二的网站Rambler 的内部使用。

Nginx的主要优势在于“轻、快、活”：

轻

> 很低的资源占用，甚至能在很多嵌入式设备上运行。

快

> 响应速度超快，几乎不会由于高并发影响响应速度。

> 配置灵活，广泛的模块支持。

网上关于Apache和Nginx性能比较的文章非常多，基本上有如下的定论：

1. Nginx在并发性能上比Apache强很多，如果是纯静态资源（图片、JS、CSS）那么Nginx是不二之选。
2. Apache有mod_php、在PHP类的应用场景下比Nginx部署起来简单很多。一些老的PHP项目用Apache 来配置运行非常的简单，例如Wordpress。
3. 对于初学者来说Apache配置起来非常复杂冗长的类XML语法，甚至支持在子目录放置.htaccess文件来配置子目录的属性。Nginx的配置文件相对简单一点。

4. Nginx的模块比较容易写，可以通过写C的mod实现接口性质的服务，并且拥有惊人的性能。
   分支OpenResty，可以配合lua来实现很多自定义功能，兼顾扩展性和性能。

这里我们要着重讨论的是为什么Nginx在并发性能上比Apache要好很多。

想要了解这个问题，不得不先做一些铺垫，讲讲并发网络编程的一些历史：

壹 最原始

最原始的网络编程的伪代码大致是这样：

```
00 listen(port)  # 监听连接收服务的端口上
01 while True:   # 一直循环
02   conn = accept()    # 接收连接
```


我们需要了解，最原始的Linux中accept、read、write调用都是 阻塞的（现在，阻塞也是这些调用的默认行为）。这就导致了以上代码只能同时 处理一个连接，所以就有了下面的方法：

贰 每个连接开一个进程

后来，大家想到了办法：

```
00 listen(port)  # 监听连接收服务的端口上
01 while True:   # 一直循环
02   conn = accept()    # 接收连接
03   if fork() == 0:
04     # 子进程
05     read_content = read(conn) # 读取连接发送过来的请求
06     response = process(conn) # 执行业务逻辑，并得到给客户端回应的内容
07     conn.write(response) # 将回应写回给连接
```

用子进程来处理连接，父进程继续等待连接进来。但这种方式有如下两个明显的缺陷：

1. fork()调用比较费时，需要对进程进行内存拷贝。即便现在的Linux普遍 引入了COW（Copy On Write）技术（fork的时候不做内存拷贝，只有其中一个 副本发生了write的时候才进行copy）加速了fork的效率，但fork依旧是个 比较“重”的系统调用。
2. 较多的内存占用，也是由于上述的内存复制造成的。

得益于之前提过的Linux对于线程的引入，上面例子的开进程，被换成了开线程，这样，上一小节说的两个缺陷都大大的被缓解了。

肆 进程/线程池

计算机领域有很多算法或者是方法都会用到一种智慧：“空间换时间”。 即用便用更多内存的方式换取更快的运行速度：事先创建出很多进程/线程 ，就像一个池子，这样虽然会浪费一部分的内存，但连接过来的时候就省去了 开启进程/线程的时间。

但这种方式会有一个比较显著的缺陷：当并发数大于进程/线程池的大小的时候 性能就会发生很大的下滑，退化成“贰”的情况。

伍 非阻塞&事件驱动

那么，是不是想要达到高性能就一定要付出高系统资源占用呢？ 答案是否定的，如果我们注意观察生活中的一个细节，肯德基和麦当劳的不同 服务方式：

- 肯德基
  1. 服务员在前台问：“先生/小姐，有什么可以帮你？”
  2. 顾客，思考一下点什么比较好：“我要，xxxxx”
  3. 服务员去后台配餐、取餐，3分钟过去了：“您的餐齐了，下一位”
- 麦当劳
  1. 服务员在前台问：“先生/小姐，有什么可以帮你？“
  2. 顾客：“我要，xxxxx”。如果顾客思考超过5秒：“后面的顾客请先点”； 点完餐，前台服务员继续为下一位顾客点餐。后台有别的服务员完成配餐。

1. 在肯德基，如果遇到需要纠结半天吃什么的客户。服务员和后面的顾客 都会陷入较长时间的等候。原因就是如果最前面的客户先让后面的顾客点餐 ，他想好了还需要较长时间的等候。相比之下，麦当劳就更胜一筹。
2. 在麦当劳，后面配餐的服务员如果发现有两个订单都要了可乐。他可以 智能地把两个订单的可乐一次性灌好，这样会大大的提高效率。各个岗位上 的服务员可以灵活的采用各种方式优化自己的工作效率。

这里，肯德基的服务方式就是古老的进程/线程池；麦当劳的服务方式 就是一个简单的非阻塞&事件驱动。

那么，非阻塞&事件驱动这么好，为什么大家没有一开始就采用这种方式呢？ 原因有二：

1. 非阻塞&事件驱动需要系统的支持，提供non-blocking版的整套 系统调用。
2. 非阻塞&事件驱动编程难度较大，需要很高的抽象思维能力， 把整个任务拆解；采用有限状态机编程才能实现。

更多精彩，请见 Reboot教育 - 高效你的学习

编辑于 2016-07-28

网络编程 Linux 高性能

文章被以下专栏收录

推荐阅读

网络编程（□二）：戏说非阻塞网络编程

auxte... 发表于面向工资编...

漫谈CGI FastCGI WSGI

auxte... 发表于面向工资编...

会用python把linux命令写一遍的人，进大厂有多容易?

只有我 发表于一起学Py...

网络编程（□□六）：端口那些事儿

auxte... 发表于面向工资编...

25 条评论

切换为时间排序

写下你的评论...

知乎用户 4 年前
刚好在看网络编程基础 受教了
1

775CPU 回复 知乎用户 1 年前
2002年，一个叫Igor Sysoev的俄罗斯哥们儿（貌似俄罗斯叫Igor的人挺多的）写出了一个叫Nginx（和Engine X谐音，取引擎之义）
赞

人生踏漫漫 4 年前
分析的很透彻
2

锅牛老湿 4 年前
点赞
1

liuyangc3 4 年前
老师讲讲 有限状态机呗：）
5

siuis 4 年前
实力黑一波肯德基.....
14

王赛 4 年前
期待更新
赞

知乎用户 4 年前
既然提到了非阻塞&事件驱动，提到了状态机，就讲讲Traffic Server吧~
2

crazyjohn 4 年前
no-blocking和fsm有什么直接关系呢?
赞

马驰 4 年前
callback hell
1

知乎用户 4 年前
文中有一处错误，应为struts
赞

auxten (作者) 回复 知乎用户 4 年前
感谢指出
赞

Alex 回复 auxten (作者) 4 年前
Yy频道写错了?
赞

展开其他 1 条回复

辛塞凯诺卡密 3 年前
什么时候有这两张万智牌单卡啥！ ~
赞

张盼 3 年前
non-blocking配合IO多路复用
赞

杨无厌 3 年前
您好！怎么理解异步呢?
赞

jinlong 3 年前
肯德基的体验真是太差了
赞

王成 3 年前
说的很明白，受教了，多谢。
赞

NeverMoes 2 年前
赶紧学习一波。
赞

知乎用户 2 年前
妙啊妙【诸葛亮脸】
赞

知乎用户 2 年前
写得很好，前面的都能懂，哪位大佬能分享点有效状态机的资料
赞

酒窝 2 年前
多线程非阻塞如何解决并发请求对同一个数据的操作，保持数据一致性呢
1

auxten (作者) 回复 酒窝 2 年前
锁、消息传递。这是基本功
1

Tom 1 年前
b
mn
。
1