

说说这篇「我为什么从python转向go」



CMGS (/users/Vav4bd) 2015.05.17 15:47* 3748 字 13919 次阅读

恩看了这篇我为什么从python转向go (<http://www.jianshu.com/p/afa14e631930>)，看来作者也是 KSO 轻办公/企业快盘团队的。作为快盘从无到有时期的工程师之一（总是被潇洒哥说他们改我留下的 bug），又恰好是 Python/Go 双修（大雾其实我是 Rust 党），其实一开始我是拒绝的，duang duang duang，那就随手写一点把。

一段段来吧，首先作者说 Python 是动态语言

python是一门动态语言，不是强类型系统。对于一个变量，我们有时候压根不知道它是什么类型，然后就可能出现int + string这样的运行时错误。

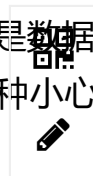
在python里面，可以允许同名函数的出现，后一个函数会覆盖前一个函数，有一次我们系统一个很严重的错误就是因为这个导致的。

事实上，如果是静态检查，pylint 和 pyflakes 是可以做这件事的，虽然不能和 go 那种静态编译型语言比，但也足够了。如果没记错的话，阿通当年是要求全组都在提交前做静态检查的。我认为这种问题更多的应该是人员素质上来避免，毕竟葱头也说过，代码自己写的就要多回头看看，看能不能重构，能不能做得更好。不是说偷懒不行，但是从中得出 Python 动态特性太灵活，Python：怪我咯？

另外，函数作为第一对象，在 Python 中是 feature，Go 要写个 mock，简直虐得不要不要的。

其实这个一直是很多人吐槽python的地方，不过想想，python最开始是为了解决啥问题而被开发出来的？我们硬是要将他用到高性能服务器开发上面，其实也是有点难为它。

如果没记错，无论是轻办公还是快盘，是重 IO 不重 CPU，最大耗时是数据块加密那块，我在的时候是 Java 写的。另外高性能服务器选 Go 也是虐得不要不要的，各种小心翼翼避免 GC。大多数极端情况下，pypy 的性能足矣胜任了，我认为这不算充分条件。



python的GIL导致导致无法真正的多线程，大家可能会说我用多进程不就完了。但如果一些计算需要涉及到多进程交互，进程之间的通讯开销也是不得不考虑的。

其实，Python 有宏可以绕开这个 GIL，但是呢架构设计得好其实可以避免的，到异步那块我会说。

无状态的分布式处理使用多进程很方便，譬如处理http请求，我们就是在nginx后面挂载了200多个django server来处理http的,但这么多个进程自然导致整体机器负载偏高。

但即使我们使用了多个django进程来处理http请求，对于一些超大量请求，python仍然处理不过来。所以我们使用openresty，将高频次的http请求使用lua来实现。可这样又导致使用两种开发语言，而且一些逻辑还得写两份不同的代码。

如果推测没错，你们现在还在用五年前写的 Gateway？那个基于 django route 的流量分发层？四年前我离开的时候已经小范围的使用 Flask+Gevent Demo 测试过了，无论是性能还是负载都比同步模型的 django 有优势。如果还是 django 这套的话，我只能说比较遗憾，毕竟当年金山新员工大赛头牌就是我和几个小伙伴写的实时同步在线文档编辑系统，用的就是这套技术。

因此这是个工程问题，并非语言问题。Python 提供给了你了这么多工具，硬要选一个传统的，Old fashion 的，Python：怪我咯？

django的网络是同步阻塞的，也就是说，如果我们需要访问外部的一个服务，在等待结果返回这段时间，django不能处理任何其他的逻辑（当然，多线程的除外）。如果访问外部服务需要很长时间，那就意味着我们的整个服务几乎在很长一段时间完全不可用。

为了解决这个问题，我们只能不断的多开django进程，同时需要保证所有服务都能快速的处理响应，但想想这其实是一件很不靠谱的事情。

同步模型并非不行，因为 overhead 足够低，很多业务场景下用同步模型反而会取得更好的效果，比如豆瓣。同步模型最大的问题是对于 IO 密集型业务等待时间足够长，这时候需要的不是换语言，而是提醒你是不是架构要改一下了。

虽然tornado是异步的，但是python的mysql库都不支持异步，这也就意味着如果我们在tornado里面访问数据库，我们仍然可能面临因为数据库问题造成的整个服务不可用。

tornado 是有这个问题，但是 gevent 已经解决了。我在 node.js 的某问题下曾经回答过，对于 node 而言，能选择的异步模型只有一个，而 Python 就是太多选择了。另外 pypy+tornado+redis 可以随意虐各种长连接的场景，比如我给我厂写过的一个 push service。

其实异步模型最大的问题在于代码逻辑的割裂，因为是事件触发的，所以我们都是通过callback进行相关处理，于是代码里面就经常出现干一件事情，传一个callback，然后callback里面又传callback的情况，这样的结果就是整个代码逻辑非常混乱。

这个还真不是，如果说没有 ES6 的 JavaScript，可能真有 Callback hell，但这是 Python 啊！Python 早就实现了左值绑定唉，yield 那姿势比某些天天吹的语言不知道高到哪里去了，当然我说的是完整版的 Python3 yield。即便是不完整的 Python 2 yield 用于异步表达式求值也是完全足够的，tornado 的 gen.coroutine 啊。

同步形态写异步，在 Python 实力强的公司里面早普及了，这是个工程问题，并非语言问题。当然把这种事怪在 Python 身上，Python：怪我咯？

python没有原生的协程支持，虽然可以通过gevent，greenlet这种的上patch方式来支持协程，但毕竟更改了python源码。另外，python的yield也可以进行简单的协程模拟，但毕竟不能跨堆栈，局限性很大，不知道3.x的版本有没有改进。

无论是 Gevent 还是 Greenlet 均没修改 Python 源码，事实上这货已经成为了 Py2 coroutine 的标准，加上豆瓣开源出来的greenify (<http://github.com/douban/greenify>)，基本上所有的库都可以平滑的异步化，包括 MySQL 等 C 一级的 lib。自从用上这套技术后，豆瓣的 Python dev 各种爽得不要不要的。

当我第一次使用python开发项目，我是没成功安装上项目需要的包的，光安装成功mysql库就弄了很久。后来，是一位同事将他整个python目录打包给我用，我才能正常的将项目跑起来。话说，现在有了docker，是多么让人幸福的一件事情。

而部署python服务的时候，我们需要在服务器上面安装一堆的包，光是这一点就让人很麻烦，虽然可以通过puppet，salt这些自动化工具解决部署问题，但相比而言，静态编译语言只用扔一个二进制文件，可就方便太多了。

恰好我又是在开发基于 docker 的平台，docker 还真不是用来做部署这事的。首先，Python 是有 virtualenv 这个工具的，事实上对比包管理和包隔离，Python 比 Go 高得不知道哪里去了。Python 跟 Git 谈笑风生的时候，Go 的 dev 们还得考虑我怎样才能使得 import 的包稳定在一个

版本上（当然现在有很多第三方方案）。Virtualenv + Pip 完全可以实现 Python 部署自动化，所以这个问题我认为是，工具链选取问题。毕竟是个十几年的老妖怪了，Python 啥情况没见过啊，各种打包工具任君选择，强行说 Python 部署不方便，Python：怪我咯？

python非常灵活简单，写c几十行代码才能搞定的功能，python一行代码没准就能解决。但是太简单，反而导致很多同学无法对代码进行深层次的思考，对整个架构进行细致的考量。来了一个需求，啪啪啪，键盘敲完开速实现，结果就是代码越来越混乱，最终导致了整个项目代码失控。

曾经知乎有个帖子问 Python 会不会降低程序员编程能力 (<http://www.zhihu.com/question/19900260/answer/30160057>)，我只能说这真的很人有关。你不去思考深层次的东西怪语言不行是没道理的，那好，Go 里面 goroutine 是怎么实现的，一个带 socket 的 goroutine 最小能做到多少内存，思考过？任何语言都有自己的优势和劣势，都需要执行者自己去判断，一味的觉得简单就不会深入思考这是有问题的。另外，代码混乱我认为还是工程上的控制力不够，豆瓣有超过10W行的 Python 实现，虽然不说很完美，大体上做到了不会混乱这么个目标。

还有，C 写几十行搞定的 Python 一行解决这绝对是重大 feature，生产力啊，人员配置啊，招人培养的成本啊，从工程上来说，Python 在这一块完全是加分项，不是每个项目都要求极致的并发，极致的效率，做工程很多时候都是要取舍的。

虽然java和php都是最好的编程语言（大家都这么争的），但我更倾向一门更简单的语言。而 openresty，虽然性能强悍，但lua仍然是动态语言，也会碰到前面说的动态语言一些问题。最后，前金山许式伟用的go，前快盘架构师葱头也用的go，所以我们很自然地选择了go。

Openresty 用 lua 如果按照动态语言的角度去看，还真算不上，顶多是个简单点的 C。许式伟走的时候大多数都是 CPP，葱头目前我还不知道他创业用的是什么写的，不过他肯定没语言倾向。当年无论是 leo 还是 ufa，一个用 Python 一个用 Java，他都是从工程实际来选择使用什么样的语言。

error，好吧，如果有语言洁癖的同学可能真的受不了go的语法，尤其是约定的最后一个返回值是 error。

这其实是 Go style，无论是 go fmt 还是 error style，Go 其实是想抹平不同工程师之间的风格问题。不再为了一个缩进和大括号位置什么的浪费时间。这种方法并不是不好，只是我个人觉得没 rust 那种返回值处理友善。

GC, java的GC发展20年了, go才这么点时间, gc铁定不完善。所以我们仍然不能随心所欲的写代码, 不然在大请求量下面gc可能会卡顿整个服务。所以有时候, 该用对象池, 内存池的一定要, 虽然代码丑了点, 但好歹性能上去了。

1.4 开始 go 就是 100% 精确 GC 了, 另外说到卡顿啊, 完全和你怎么用对象有关, 能内联绝不传引用大部分场景是完全足够的, 这样 gc 的影响程度会最低。实在想用池.....只能说为啥不选 Java。

天生的并行支持, 因为goroutine以及channel, 用go写分布式应用, 写并发程序异常的容易。没有了蛋疼的callback导致的代码逻辑割裂, 代码逻辑都是顺序的。

这是有代价的, goroutine 的内存消耗计算(当然1.3还是1.4开始得到了很大的改善, 内存最小值限制已经没了), channel 跨线程带来的性能损耗(跨线程锁), 还有对 goroutine 的控制力几乎为 0 等。总之这种嘛, 算不上是杀手级特性, 大家都有, 是方便了一点, 但也有自己的弊端。比如我们用 go 吧, 经常就比较蛋疼 spawn 出去的 goroutine 怎么优美的 shutdown, 反而有时候把事情做复杂化了。

性能, go的性能可能赶不上c, c++以及openresty, 但真的也挺强悍的。在我们的项目中, 现在单机就部署了一个go的进程, 就完全能够胜任以前200个python进程干的事情, 而且CPU和MEM占用更低。

我不严谨的实测大概 gevent+py2 能达到同样逻辑 go 实现的 30%~40%, pypy+tornado 能达到 80%~90%, 混合了一些计算和连接处理什么的。主要还是看业务场景吧, 纯粹的 CPU bound 当然是 go 好, 纯粹的 IO bound 你就是用 C 也没用啊。

运维部署, 直接编译成二进制, 扔到服务器上面就成, 比python需要安装一堆的环境那是简单的太多了。当然, 如果有cgo, 我们也需要将对应的动态库给扔过去。

我们现在根据 glibc 所处的 host 版本不同有2套编译环境, 看上去是部署简单了, 编译起来坑死你。另外虽然说 disk 便宜, 这几行代码就几M了, 集群同步部署耗时在某些情况下还真会出篓子。

开发效率, 虽然go是静态语言, 但我个人感觉开发效率真的挺高, 直觉上面跟python不相上下。对于我个人来说, 最好的例子就是我用go快速开发了非常多的开源组件, 譬如ledisdb, go-mysql等, 而这些最开始的版本都是在很短的时间里面完成的。对于我们项目来说, 我们也是用go在一个月就重构完成了第一个版本, 并发布。

go 的开发效率高是对比 C , 对比 python , 大概后者只需要3天吧.....

总之，Go 不是不好，Python 也不是不行，做工程嘛，无外乎就是考虑成本，时间成本，人力成本，维护成本等等。Go 和 Python 互有千秋，就看取舍了。当然一定要说 Python 不行，Python：怪我咯？


我为什么从python转向go - 简书 — 应puppet大拿刘宇的邀请，我去西山居运维团队做了一个简短分享，谈谈为什么我要将我们的项目从python转向go。坦白的讲，在一帮python用户面前讲为什么放弃python转而去用go其实是一件压力蛮大的事情，语言之争就跟vim和emacs之争一样，是一个永恒的无解话题，稍微不注意就可... (http://www.jianshu.com/p/afa14e631930) CMGS (/users/Vav4bd) · www.jianshu.com → (http://www.jianshu.com/p/afa14e631930)

➕ 推荐拓展阅读 (/sign_in)

♡ 喜欢

43

作者



CMGS (/users/Vav4bd)

+ 添加关注 (/sign_in)

≡

(http://www.jianshu.com/p/afa14e631930) (/users/Vav4bd)

2

文章 (/users/Vav4bd)


66

粉丝 (/users/Vav4bd/followers)

56


喜欢 (/users/Vav4bd/top_articles)

登录后发表评论 (/sign_in)

 (/users/1acc906001d2) 戈戈 (/users/1acc906001d2) 2015.05.22 13:52 (/p/xiQzpL/comments/333297#comment-333297)


兄台，我好喜欢你啊

↩ 回复

 (/users/a6469e3f5dbc) soulpower (/users/a6469e3f5dbc) 2015.05.21 15:14
(/p/xiQzpL/comments/330815#comment-330815)


大神我给你点赞了

↩ 回复

 (/users/42a56a5e23d3) 一际孤鸿_FBW (/users/42a56a5e23d3) 2015.05.20 05:55
(/p/xiQzpL/comments/328219#comment-328219)


大神，pulsar这框架用的如何？已经发布稳定版1.0了，只支持python3.4+

↩ 回复

 (/users/a60cc254217f) linyehui (/users/a60cc254217f) 2015.05.19 15:49
(/p/xiQzpL/comments/326927#comment-326927)


要不要

↩ 回复

 (/users/9b085e630e1d) 化浊 (/users/9b085e630e1d) 2015.05.18 11:44
(/p/xiQzpL/comments/324211#comment-324211)

害怕c++，喜欢python，不知道啥是go。

↩ 回复

 (/users/24196d57df5e) MengZhuo (/users/24196d57df5e) 2015.05.17 23:59
(/p/xiQzpL/comments/323440#comment-323440)

>> 所以我们仍然不能随心所欲的写代码，不然在大请求量下面gc可能会卡顿整个服务。
主要是用法不对，肯定是上来就创建struct对象，然后传引用escape作用域，导致大量对象需要标记、清除。

↩ 回复

作者的其他热门文章：

Project Eru (/p/e41eb80a400e?
utm_campaign=maleskine&utm_content=note&utm_medium=pc_author_hots&u
tm_source=recommendation)

收获了 13 个喜欢, 2 条评论

你可能会感兴趣：

变成自己喜欢的样子 (/p/2782e4852c88?utm_campaign=maleskine&utm_content=note&utm_medium=pc_all_hots&utm_source=recommendation)

诺佳 (/users/929551ca6a8b) 收获了 2241 个喜欢, 264 条评论

2015年只剩下了三分之二，快趁着青年节收下这几款能提升自己的应用，做个进步青年吧！
(/p/13d74091ff05?utm_campaign=maleskine&utm_content=note&utm_medium=pc_all_hots&utm_source=recommendation)

元茜姑娘 (/users/2ec3c7ca3277) 收获了 2358 个喜欢, 54 条评论

18岁的我们，22岁的我和你 (/p/1d276a49ae36?utm_campaign=maleskine&utm_content=note&utm_medium=pc_all_hots&utm_source=recommendation)

生椒牛肉 (/users/735ef8e7f459) 收获了 801 个喜欢, 253 条评论
