Table of Contents

<Go语言的并发网络爬虫-Go与C的混合高效率编程>

作者： 传智播客C++学院 刘丹冰

Why Go?

现有的工程语言的一些缺点

1 php python ruby 系列

这些都是动态语言，性能太慢不说，一旦代码规模庞大就很难控制代码质量。

2 java scala c#系列

这一类语言性能强大 内库完善，带着一个高效的gc(内存垃圾回收机制)， 看起来不错。
但同样他们也存在一个巨大的缺点，带一个硕大的运行时。导致各种资源消耗。
同时java语法的繁琐也经常被人诟病。

3 c/c++系列

这一类语言偏偏对处理字符串比较弱。而且没有gc。很难写出稳定性强的web程序。
而且c++的编译性能弱爆了，你开一个大型项目，随便make一下，一天就这么过去了。

那么go语言呢？

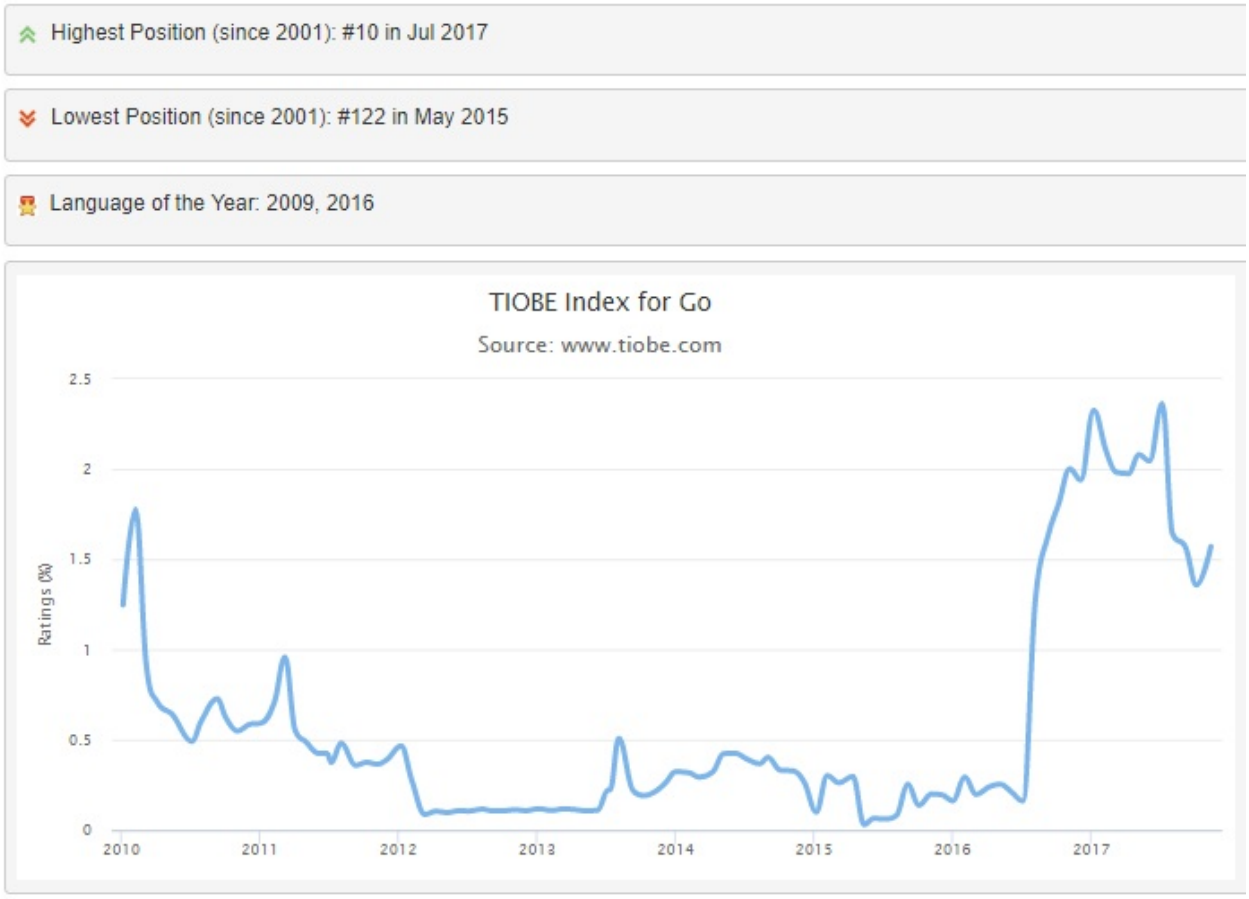
业界一直需要一个轻量级,高性能,语法简洁同时带有一个完善的gc的编程语言，这个时候，刚好google的go语言发布，有强大的公司背景，go语言开发者也都是业界的大牛。而且每一个发布版质量可靠，性能稳定。自然不出意外收获了广泛的重视。

go语言虽然很多特性都被人诟病，语法层面乏善可陈，可是他毕竟解决了码农的痛点。迅速形成了一个强大的社区。强大的社区又会导致语言进一步强化自身，形成正反馈，想不成功都难。

截止2017.11月份 语言排行榜

Nov 2017	Nov 2016	Change	Programming Language	Ratings	Change
1	1		Java	13.231%	-5.52%
2	2		C	9.293%	+0.09%
3	3		C++	5.343%	-0.07%
4	5	^	Python	4.482%	+0.91%
5	4	v	C#	3.012%	-0.65%
6	8	^	JavaScript	2.972%	+0.27%
7	6	v	Visual Basic .NET	2.909%	-0.26%
8	7	v	PHP	1.897%	-1.23%
9	16	^^	Delphi/Object Pascal	1.744%	-0.21%
10	9	v	Assembly language	1.722%	-0.72%
11	19	^^	R	1.605%	-0.11%
12	15	^	MATLAB	1.604%	-0.36%
13	14	^	Ruby	1.593%	-0.39%
14	13	v	Go	1.570%	-0.43%
15	10	vv	Perl	1.562%	-0.80%
16	26	^^	Scratch	1.550%	+0.47%
17	17		Visual Basic	1.489%	-0.43%
18	20	^	PL/SQL	1.453%	-0.06%
19	11	vv	Objective-C	1.412%	-0.83%
20	12	vv	Swift	1.389%	-0.65%

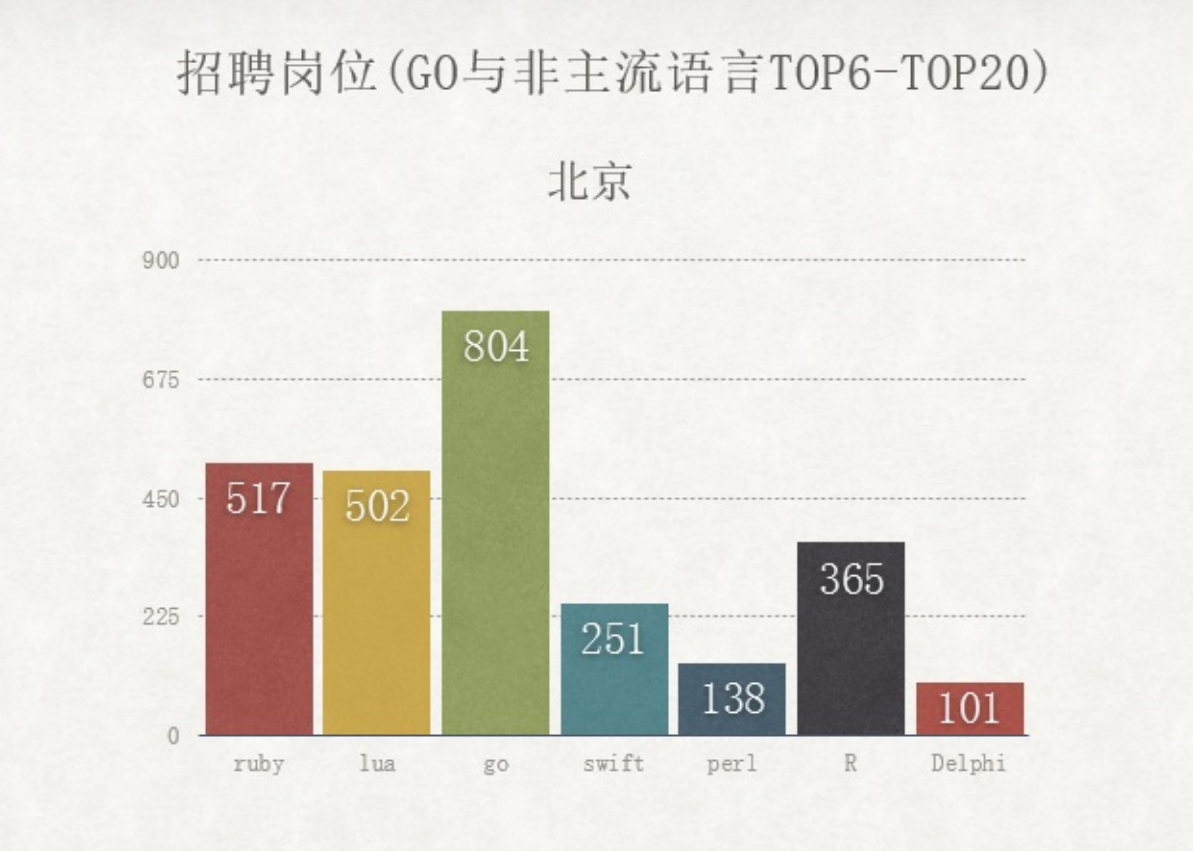
go语言近年趋势：

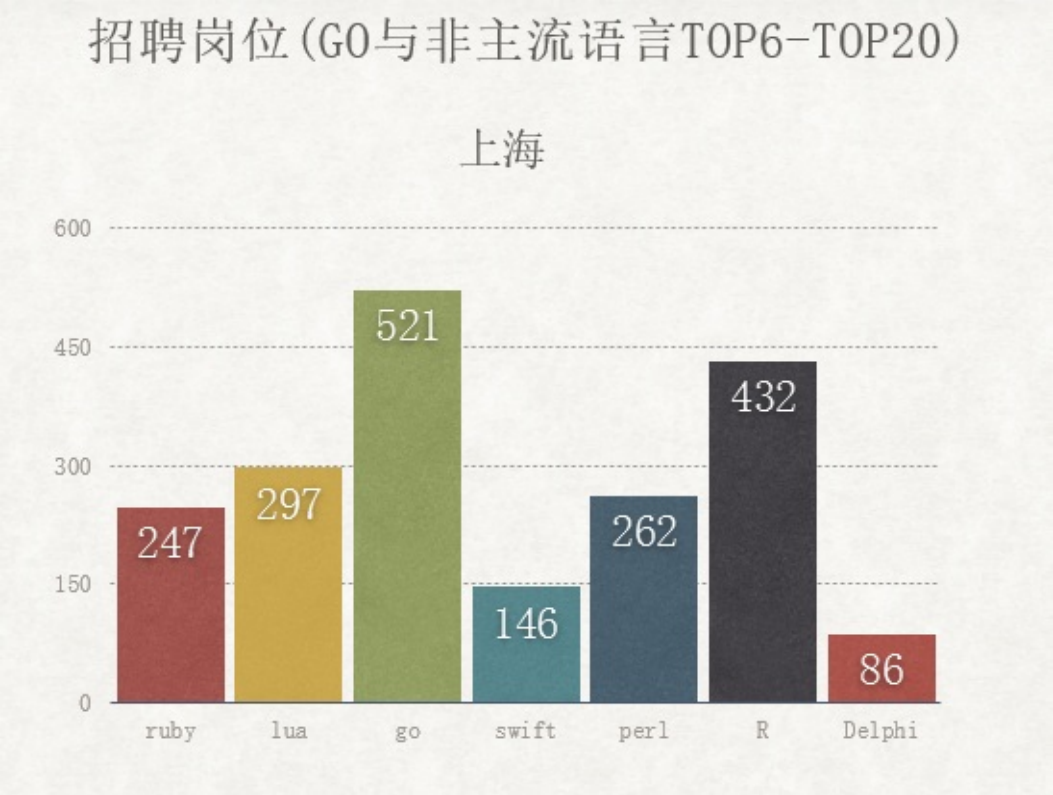


历年语言涨幅冠军，**go**语言在**2016**年夺冠

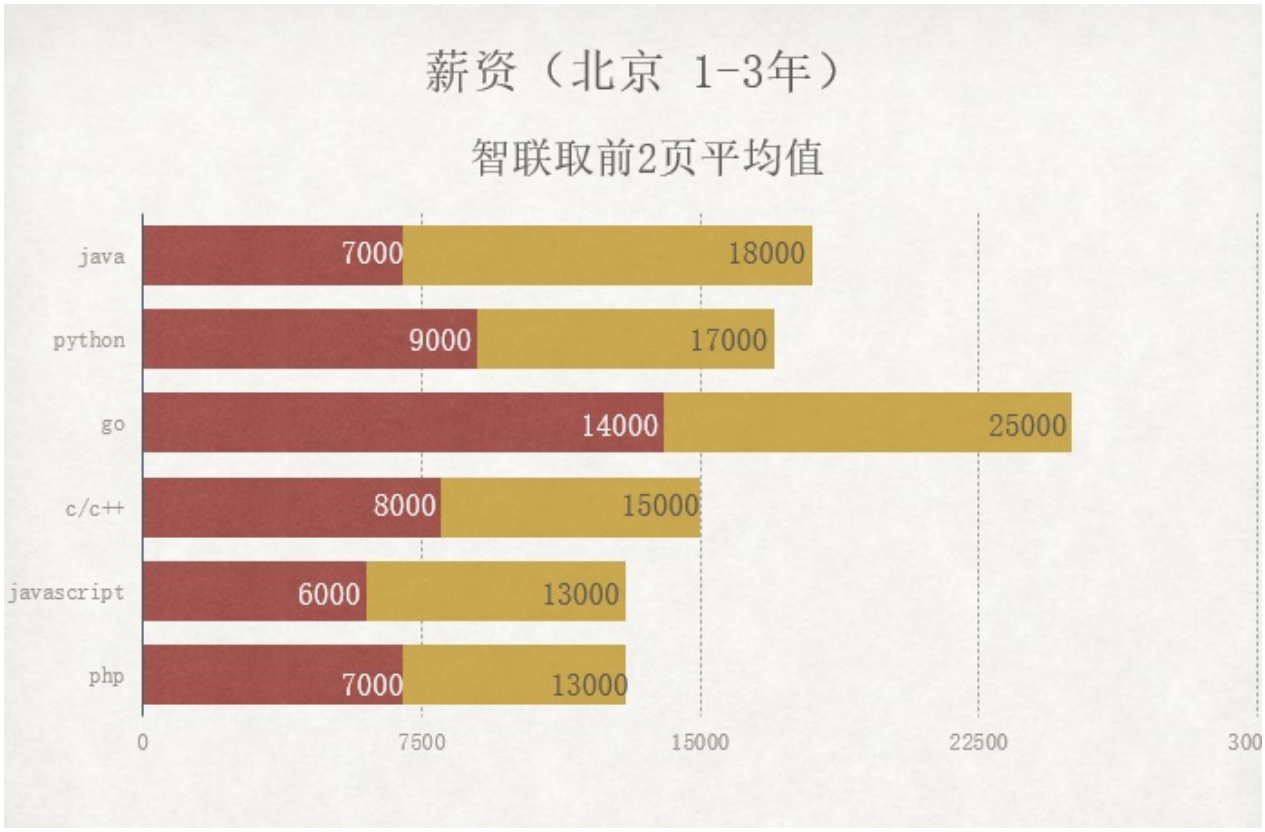
Year	Winner
2016	Go
2015	Java
2014	JavaScript
2013	Transact-SQL
2012	Objective-C
2011	Objective-C
2010	Python
2009	Go
2008	C
2007	Python
2006	Ruby
2005	Java
2004	PHP
2003	C++

go语言与其他非主流语言在找岗位对比

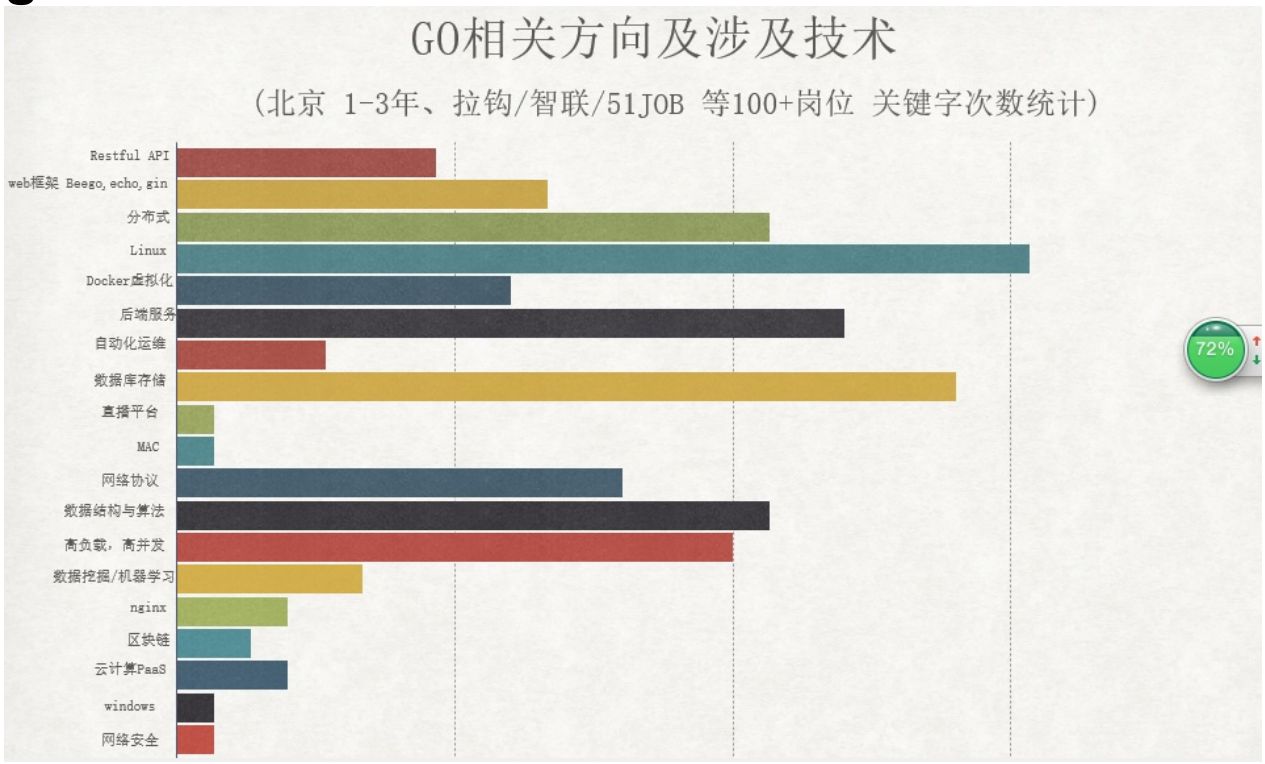

招聘岗位(GO与非主流语言TOP6-TOP20)
北京
900
675
450
225
0
517
502
804
251
138
365
101
ruby
lua
go
swift
perl
R
Delphi


招聘岗位(GO与非主流语言TOP6-TOP20)
上海
600
450
300
150
0
247
297
521
146
262
432
86
ruby
lua
go
swift
perl
R
Delphi

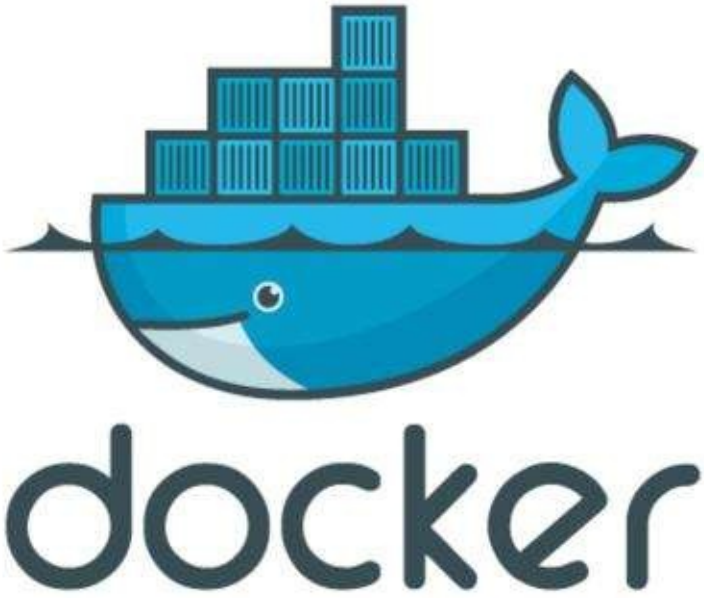
go语言目前薪资对比



go语言的行业相关技术及方向



1.docker



到现在，Docker几乎是Go再难找到也难以复制的一个成功案例。Docker项目在2014年9月份就拿到了C轮4000万美元融资，版本迭代速度超快，目前从GitHub看到已有78个版本，而它仅仅是再2013年初才正式开始的一个项目而已。目前，国内Docker技术推广也进行的如火如荼，比如Docker中文社区，CSDN也建立了Docker专区。CSDN CODE也将在近期与Docker中文社区合作，推出Docker技术文章翻译活动，届时也请大家多多关注，及时关注与参与。

Docker团队之所以喜欢用Go语言，主要是Go具有强大的标准库、全开发环境、跨平台构建的能力。

GitHub托管地址：

https://github.com/docker/docker

2. Kubernetes

Kubernetes是2014年夏天Google推出的Kubernetes，基于Docker，其目的是让用户通过Kubernetes集群来进行云端容器集群的管理，而无需用户进行复杂的设置工作。系统会自动选取合适的工作节点来执行具体的容器集群调度处理工作。其核心概念是Container Pod（容器仓）。

GitHub托管地址：

https://github.com/GoogleCloudPlatform/kubernetes

3. Etcd &Fleet

etcd是由CoreOS开发并维护键值存储系统，它使用Go语言编写，并通过Raft一致性算法处理日志复制以保证强一致性。目前，Google的容器集群管理系统Kubernetes、开源PaaS平台Cloud Foundry和CoreOS的Fleet都广泛使用了etcd。详情，可了解《

Etcd：用于服务发现的键值存储系统》。Fleet则是一个分布式的初始化系统。它们之所以选择使用Go语言，则是因为Go语言对跨平台的良好支持，以及其背后的强大社区。

GitHub托管地址：

https://github.com/coreos/etcd

4. Deis

@InfoQ

Deis是一个基于Docker和CoreOS的开源PaaS平台，旨在让部属和管理服务器上的应用变得轻松容易。它可以运行在AWS、GCE以及Openstack平台下。详情，可了解《Deis v1.0正式发布！》。

GitHub托管地址：

https://github.com/deis/deis

5. Flynn

Flynn是一个使用Go语言编写的开源PaaS平台，可自动构建部署任何应用到Docker容器集群上运行。Flynn项目受到Y Combinator的支持，目前仍在开发中，被称为是下一代的开源PaaS平台。

GitHub托管地址：

https://github.com/flynn/Flynn

1 Beego

Beego是一个完全的MVC框架，你可以使用你的Go语言专业技术构建你的web应用程序。
Beego框架下，你可以

自动化地实现测试、打包和部署。

更主要原因Beego出自国产。

2 Gin

机器学习与人工智能相关

- **Golearn**

GoLearn是自称“内置电池”的机器学习资料库，绝对是首选项之一。

作者在项目描述中提到——简洁、易定制是其追求的目标。GoLearn中一些接口使用的数据处理方式和scikit-learn（一个非常流行的Python机器学习项目）是非常相似的。想要逃离Python的用户应该可以用它做一些短期的工作。其中另外还有一些使用C++构筑的线性模型

资料库，但是其他的全是Go语言编写的。GoLearn实现了熟悉的Scikit-learn 适应/预测界面，可实现快速预估测试和交换。GoLearn是一个成熟的项目，它提供了交叉验证和训练/测试等辅助功能。

- **Goml**

Goml自诩为“在线**Golang**机器学习工具”，据其开发者所言意思是其“包含了许多工具，能让你以在线方式学习其频道的数据内容。”这个项目之所以突出是因为其强调了其作为其他应用一部分存在的可能性，使得构筑“综合测试、大量文档以及简洁、高效、模块化的源代码”更加容易些了。但是如果你需要的知识解决基础的二元分类问题（是否是垃圾邮件？），你可能更适合使用Hector这个更小型的资料库。

- **Gorgonia**

最新的一个分支（或者某种程度上说最令人感兴趣的）是Gorgonia。

这个机器学习资料库完全是用Go语言编写而成，据其开发者“chewxy”称能“提供动态建立神经网络及相关算法必需条件。”

关键在于“动态”。和之前的机器学习资料库Theano一样，Gorgonia允许你使用一系列原始资料库中的高阶术语来描述神经网络的行为。TensorFlow资料库也使用这种方式，使得开发者不用再亲自编写算法，也不用再提交那些能在不同项目中重复使用的项目。

为什么使用Go语言来编写这个机器学习项目Gorgonia？

其开发者在接受采访中提到：“我写Gorgonia其中一个原因是我曾经花费太久的时间尝试云端中部署Theano（大约在两年前）。”

其他全部框架及组件

服务框架

1.
    - gizmo, a microservice toolkit from The New York Times ★
    - go-micro , a microservices client/server library ★
    - gocircuit, dynamic cloud orchestration
    - gotalk, async peer communication protocol &library
    - h2, a microservices framework ★
    - Kite, a micro-service framework

2. 独立组件

    afex/hystrix-go, client-side latency and fault tolerance library

    armon/go-metrics, library for exporting performance and runtime metrics to external metrics systems

    codahale/lunk, structured logging in the style of Google's Dapper or Twitter's Zipkin

    eapache/go-resiliency, resiliency patterns

    sasbury/logging, a tagged style of logging

    grpc/grpc-go, HTTP/2 based RPC

    inconshreveable/log15, simple, powerful logging for Go ★

    mailgun/vulcand, programmatic load balancer backed by etcd

    mattheath/phosphor, distributed system tracing

    pivotal-golang/lager, an opinionated logging library

    rubyist/circuitbreaker, circuit breaker library

    Sirupsen/logrus, structured, pluggable logging for Go ★

    sourcegraph/appdash, application tracing system based on Google's Dapper

    spacemonkeygo/monitor, data collection, monitoring, instrumentation, and Zipkin client library

    streadway/handy, net/http handler filters

    vitess/rpcplus, package rpc + context.Context

    gdamore/mangos, nanomsg implementation in pure Go

3. **Web 框架**

   Beego

   Gin

   Goji

   Gorilla

   Martini

   Negroni

   Revel (considered harmful)

go-网络爬虫

1 网络爬虫是何物？

说道网络爬虫，然后它并不是一种爬虫，而是一种可以在网上任意搜索的一个脚本程序。

有人说一定要结束网络爬虫到底是干毛用的。

尝试用了很多种解释，最终归纳为一句话。

> “你再也不必用鼠标一条一条从网页上拷贝信息了！”

一个爬虫程序将会高效并且准确的从网上拿到你希望得到的所有信息，从而帮我们省去以下行为：

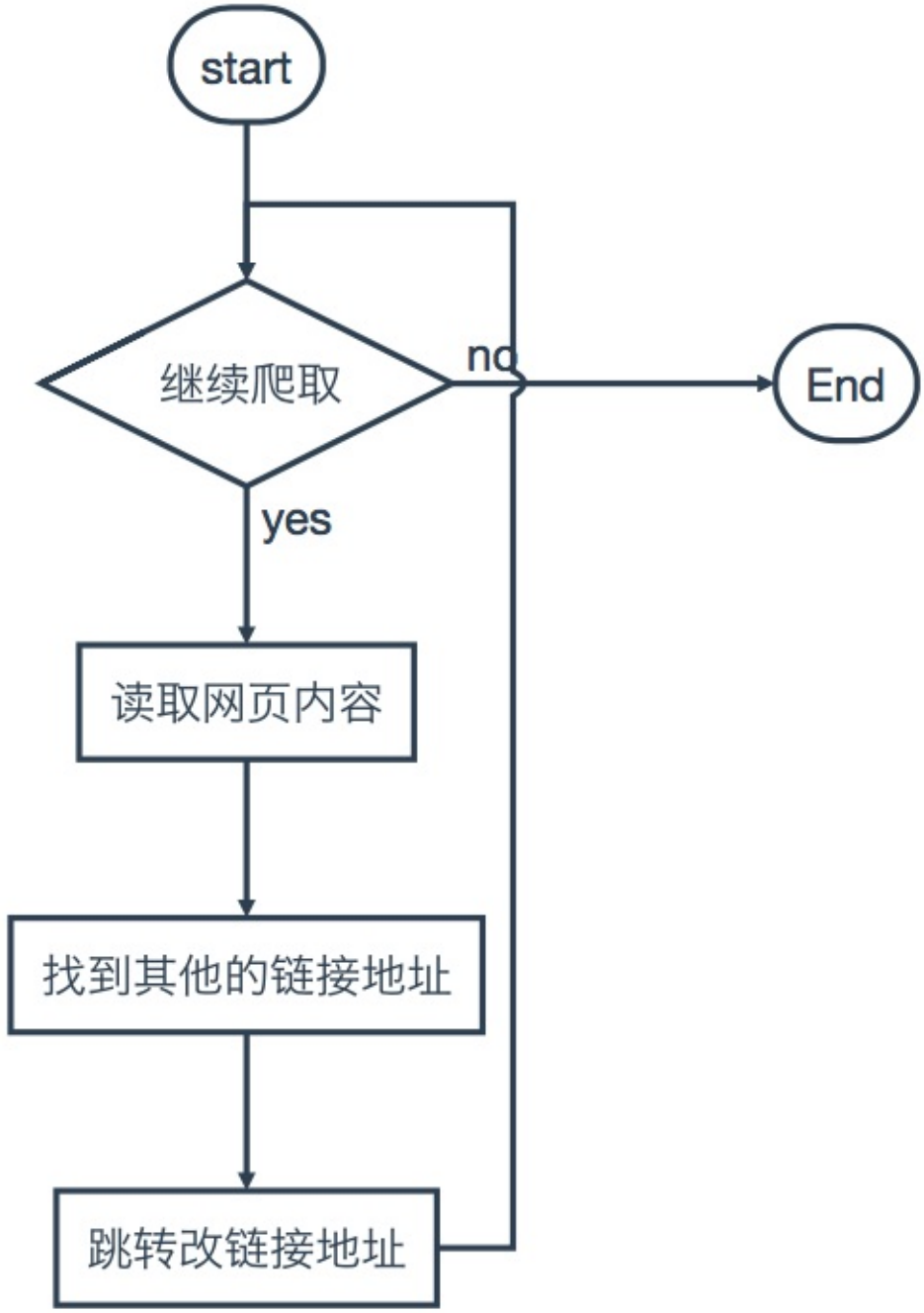
```
while (no_dead) {
    寻找网页
    鼠标点击
    ctrl-c
    ctrl-v
    翻页
}
```

当然网络爬虫的眞正意义不仅如此，由于它可以自动提取网页信息，使他成为了 搜索引擎 从万维网上下载网页的重要利器。下面我们来介绍一下网络爬虫的正经定义。

网络爬虫 (web Spider)，Spider是蜘蛛的意思，实际上名字是很形象的，他们把 互联网 比喻成一个 蜘蛛网 ，那么所谓的这个spider就在网上爬来爬去。这个网络蜘蛛是通过网页的链接地址来寻找网页的。

蜘蛛的主要行径： 网页首页—>读取网页内容—>找到网页中其他的链接地址—>其他网页的首页—>.....这样的循环下去，直到将这个网站上所有的网页都吃光（网页上所有的信息全部用蜘蛛得

到）。



这样的循环下去，直到将这个网站上所有的网页都吃光（网页上所有的信息全部用蜘蛛得

到）。如果你敢把互联网比喻成一个网站，一定会有那么一个网络蜘蛛能够可以把 整个互联网 的资

源全部吃光！！

ok，那么显而易见，网络爬虫的基本操作就是抓取网页。网页地址就是一个叫URL的东西，

2 URL初步概念

我们先来介绍一下浏览网页的基本过程。
比如我们在浏览器地址栏输入

http://www.itcast.cn

整个过程大致会发生以下步骤:

1. 本地浏览器(客户端) ———请求———> 传智服务器(服务端)
2. 本地浏览器(客户端) <———文件数据—— 传智服务器(服务端)
3. 本地浏览器(客户端) 进行解析文件数据并且展现。

那么实际上浏览器用的是一种叫html标记的语言来进行解析的。

html标记语言 ： `http://www.w3school.com.cn/`

ok，那么到底谁是URL呢，说了半天， http://www.itcast.cn

它！就是URL！没错，就是它！

我们给浏览器输入的地址，实际上就是一个url(Uniform Resource Locator) 统一资源定位符。

就是 地址 啦，搞学术的人非得弄的很高端。

明明是高利贷，他们非得说成p2p，明明是算命的，他们非得说成分析师~

> URL的一般格式是： protocol:// hostname[:port] / path / [;parameters][?query]#fragment

基本上是由三部分组成：

1 协议(HTTP呀，FTP呀~~等等)

2 主机的IP地址(或者域名)

3 请求主机资源的具体地址（目录，文件名等）

其中:

> 第一部分和第二部分用 “://” 分割
>
> 第二部分和第三部分用 “/” 分割

```
1://2/3 -->  http://www.itcast.cn/channel/teacher.shtml#ac
```

下面看几个URL例子：

http://xianluomao.sinaapp.com/game

其中

- 协议： http，
- 计算机域名： xianluomao.sinaapp.com,
- 请求目录：game

http://help.qunar.com/list.html

其中

- 协议：http，
- 计算机域名：help.qunar.com
- 文件：list.html

网络爬虫的主要处理对象就是类似于以上的URL，爬虫根据URL地址取得所需要的文件内容，然后对它进一步的处理。

3 Go语言发送 http请求？

```
package main

import "fmt"
import "net/http"
import "io/ioutil"

func httpGet(url string) (content string, statusCode int) {

        //发送http.Get方法请求远程url
        resp, err := http.Get(url)
        if err != nil {
                fmt.Println(err)
                statusCode = -100
                return
        }

        //关闭网络通信文件
        defer resp.Body.Close()

        //通过ioutil库从服务器得到返回的结果
        //data 即为 返回的网络数据
        data, err := ioutil.ReadAll(resp.Body)
        if err != nil {
                fmt.Println(err)
                statusCode = resp.StatusCode
                return
        }

        //content为网络数据
        content = string(data)
        //statusCode为返回的http 状态值
        statusCode = resp.StatusCode

        return
}

func main() {

        content, rcode := httpGet("http://www.baiduhuaidan.com")
        fmt.Println("content = ", content)
        fmt.Println("rocde = ", rcode)
}
```

4 go语言实现百度贴吧小爬虫

```
package main

import (
        "fmt"
```

```
        "io/ioutil"
        "net/http"
        "os"
        "strconv"
)

//第1页：https://tieba.baidu.com/f?kw=lol&ie=utf-8&pn=0
//第2页：https://tieba.baidu.com/f?kw=lol&ie=utf-8&pn=50
//第3页：https://tieba.baidu.com/f?kw=lol&ie=utf-8&pn=100
//第4页：https://tieba.baidu.com/f?kw=lol&ie=utf-8&pn=150

func httpGet(url string) (content string, statusCode int) {

        resp, err := http.Get(url)
        if err != nil {
                fmt.Println(err)
                statusCode = -100
                return
        }

        defer resp.Body.Close()

        data, err := ioutil.ReadAll(resp.Body)
        if err != nil {
                fmt.Println(err)
                statusCode = resp.StatusCode
                return
        }

        content = string(data)
        statusCode = resp.StatusCode

        return
}

//主业务爬取函数
func spider_tieba(begin_page int, end_page int) {

        var pn int
        fmt.Println("准备爬取从 ", begin_page, " 到 ", end_page, " 页")

        for page := begin_page; page < end_page+1; page++ {
                fmt.Println("正在爬取 第", page, "页")

                //明确目标
                pn = (page - 1) * 50
                url := "https://tieba.baidu.com/f?kw=lol&ie=utf-8&pn=" + strconv.Itoa(
pn)
                fmt.Println("url=", url)

                //开始爬取
                content, rcode := httpGet(url)
                if rcode < 0 {
```

```
                        fmt.Println("httpGet error, rcode = ", rcode, "page = ", page)
                        continue
                }

                //处理数据(把数据保存在本地文件中)
                filename := strconv.Itoa(page) + ".html"
                if f, err := os.Create(filename); err == nil {
                        //打开文件成功
                        f.WriteString(content)
                        f.Close()
                }

        }
}

func main() {

        var begin_page string
        var end_page string

        fmt.Println("请输入要爬取的起始页码")
        fmt.Scanf("%s\n", &begin_page)
        fmt.Println("请输入要爬取的终止页码")
        fmt.Scanf("%s\n", &end_page)

        b, _ := strconv.Atoi(begin_page)
        e, _ := strconv.Atoi(end_page)

        spider_tieba(b, e)

}
```

go与C的高度兼容混合编程

```
package main

/*
#include <stdio.h>
#include <stdlib.h>
#include "haha.h"

#cgo linux CFLAGS: -I./
#cgo linux LDFLAGS: -L./ -lhaha

void c_print(char *str)
{
         printf("%s\n", str);
}

void foo() {
        printf("i am foo\n");
}

*/
import "C" //import "C" 必须单起一行，并且紧跟在注释行之后

import "unsafe"

func main() {
        s := "Hello Cgo"
        cs := C.CString(s)                //将go的字符串变成c的字符串
        C.c_print(cs)                     //调用C函数
        defer C.free(unsafe.Pointer(cs)) //释放内存

        C.foo()
        C.haha()
}
```

C语言和go的结合非常兼容，不需要任何太多的数据类型转换。

传智播客C++学院

http://www.itcast.cn/subject/czly/index.shtml

近期开班计划

C/C++ 基础班	2017-12-10	火爆报名中	距离开班 11 天 16 时 17 分 41 秒
C/C++ 基础班	2018-01-04	火爆报名中	距离开班 36 天 16 时 4 分 23 秒
C/C++ 就业班	2018-01-28	火爆报名中	距离开班 60 天 16 时 4 分 17 秒
C/C++ 基础班	2018-02-26	火爆报名中	距离开班 89 天 15 时 59 分 13 秒
C/C++ 就业班	2018-03-30	火爆报名中	距离开班 121 天 15 时 59 分 8 秒