

计算机应用编程

网站页面分析

田野@网络技术研究院

yetian@bupt.edu.cn

2018.7.2

关于考核

- 课程安排
 - 2018.7.2 网站页面分析
 - 2018.7.9 机器学习实验
 - 2018.7.20 考核
- 两个大作业
 - 2人一组完成
 - 提交可编译通过源代码
 - 实验报告
- 助教：科研楼**408**房间
 - 徐岩: slow_slone@163.com
 - 王舜尧: shunyaowang@foxmail.com
- 课程邮箱: **buptsummerschool@163.com** 密码**bupt2018**



实验任务

- 目标
 - 设计一个网站页面分析系统
 - 爬取十几万个页面的网站，构建页面链接网络，计算pagerank最高的前10个页面
- 编程技能
 - C语言编程
 - 网络编程与HTTP协议
 - 并发网络编程
 - 稀疏矩阵与图结构
 - 随机过程
 - 特征值与特征向量
 - PageRank算法

实验意义

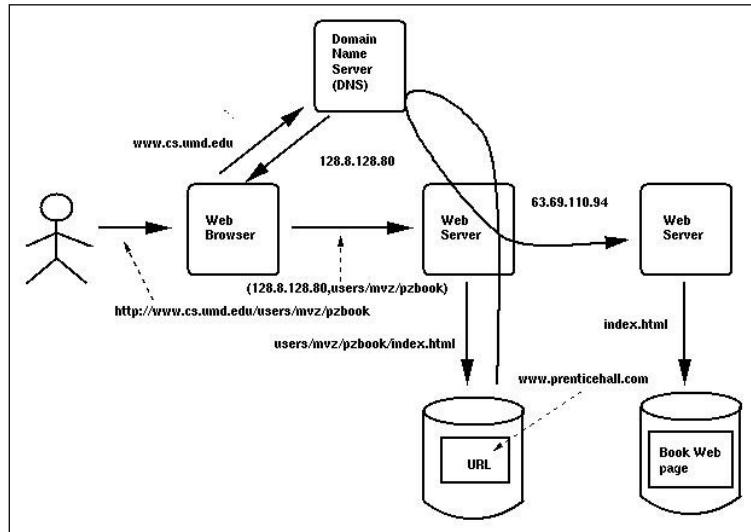
- 各种实际应用

- 科研或产品研究都需要从互联网抓取数据
- 如何设计高性能的网络爬虫，实现高速并发抓取
- 垃圾邮件过滤、网页去重等都需要快速去重
- 基于图结构分析复杂网络，例如社交网络、科研合作网络、欺诈网络等
- 如何利用矩阵来建模图并理解矩阵的作用

计算机应用编程实验2018



网页浏览原理



计算机应用编程实验2018

HTTP协议

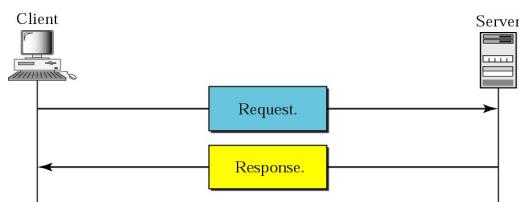
- **HTTP layered over bidirectional byte stream**

- TCP port 80
 - **RFC 1945**

- **功能**

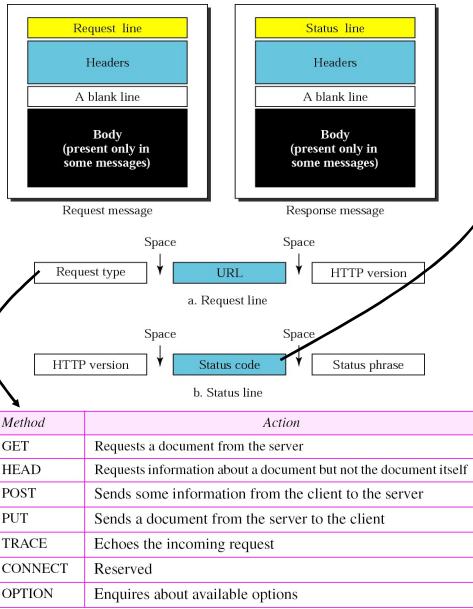
- 比之前的FTP简单有效
 - 客户可以从Web服务器上下载几乎所有类型的文件，包括HTML文件，图像/视频/音频等多媒体文件，Java Applet等对象，甚至应用程序等。

- **C/S模式**



计算机应用编程实验2018

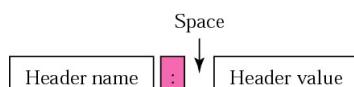
HTTP请求和响应



Code	Phrase	Description
Informational		
100	Continue	The initial part of the request has been received and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
Success		
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.
Redirection		
301	Multiple choices	The requested URL refers to more than one resource.
302	Moved permanently	The requested URL is no longer used by the server.
304	Moved temporarily	The requested URL has moved temporarily.
Client Error		
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
403	Forbidden	Service is denied.
404	Not found	The document is not found.
405	Method not allowed	The method is not supported in this URL.
406	Not acceptable	The format requested is not acceptable.
Server Error		
500	Internal server error	There is an error, such as a crash, at the server site.
501	Not implemented	The action requested cannot be performed.
503	Service unavailable	The service is temporarily unavailable, but may be requested in the future.

计算机应用编程实验2018

HTTP Header格式



Request headers

Header	Description
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
From	Shows the e-mail address of the user
Host	Shows the host and port number of the client
If-modified-since	Send the document if newer than specified date
If-match	Send the document only if it matches given tag
If-non-match	Send the document only if it does not match given tag
If-range	Send only the portion of the document that is missing
If-unmodified-since	Send the document if not changed since specified date
Referrer	Specifies the URL of the linked document
User-agent	Identifies the client program

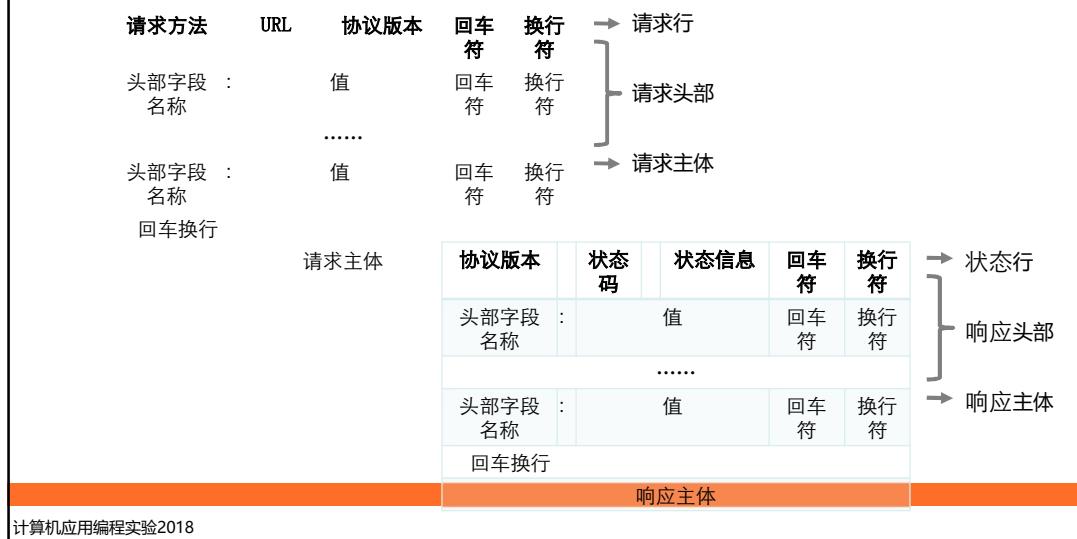
Response headers

Header	Description
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

计算机应用编程实验2018

http协议

- Web客户端和服务器之间交互用的基于文本的应用级协议
- 请求报文，响应报文

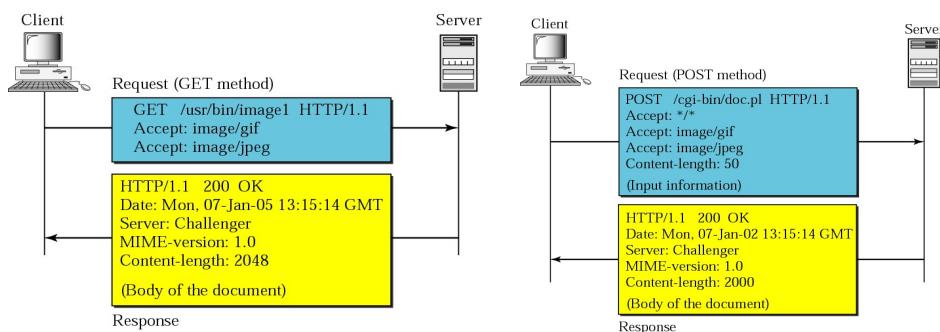


计算机应用编程实验2018

HTTP Example

使用GET方法获取图像
/usr/bin/image1，客户可以接收
GIF或JPEG格式。

客户机使用POST方法发送数据
给服务器，服务器使用perl脚本
处理数据。



计算机应用编程实验2018

HTTP Example2

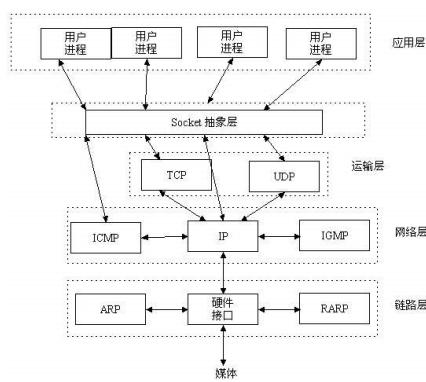
Headers Sent	Value
(Request-Line)	POST /auction/coupon/validate_exchange.htm
Accept	image/gif, image/jpeg, image/pjpeg, image/pj
Accept-Encoding	gzip, deflate
Accept-Language	zh-cn
Cache-Control	no-cache
Connection	Keep-Alive
Content-Length	25
Content-Type	application/x-www-form-urlencoded
Cookie	ab=24; sslogin=; tracknick=honda418; tg=0;
Host	auction1.taobao.com
Referer	http://auction1.taobao.com/auction/coupon/
User-Agent	Mozilla/4.0 (compatible; MSIE 8.0; Windows N

计算机应用编程实验2018

Socket网络编程

- **Socket是什么？**

- 应用层与TCP/IP协议族通信的中间软件抽象层，是一组接口。
- Socket是一种设计模式，它把复杂的TCP/IP协议族隐藏在Socket接口后面，对用户来说，一组简单的接口就是全部，让Socket去组织数据，以符合指定的协议



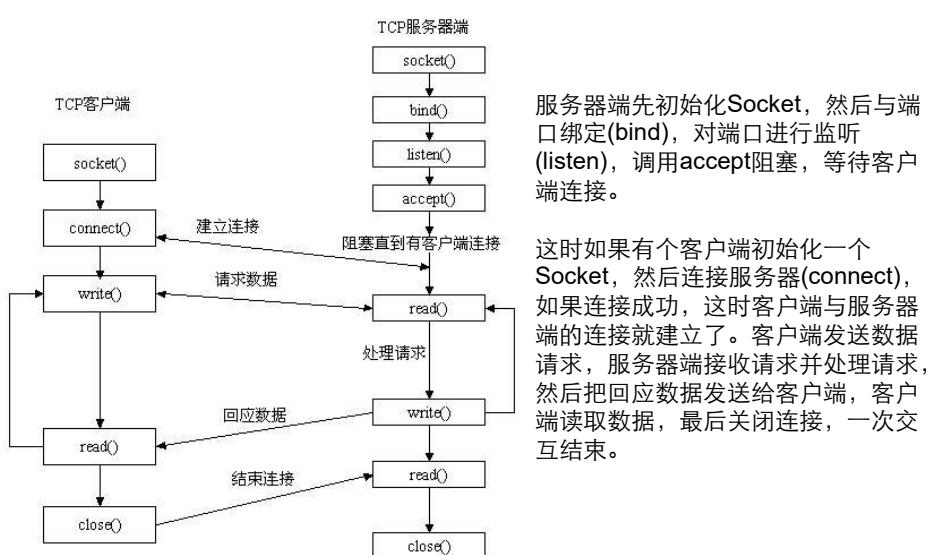
计算机应用编程实验2018

Socket概念

echo	7	验证2台计算机连接有效性
• daytime	13	服务器当前时间文本描述
ftp	20/21	21用于命令,20用户数据
telnet	23	远程登录
• smtp	25	邮件发送
whois	43	网络管理的目录服务
finger	79	主机用户信息
http	80	HTTP
pop3	110	邮局协议
nntp	119	网络新闻传输协议, 发布Usenet新闻

计算机应用编程实验2018

Socket接口框架



计算机应用编程实验2018

来个简单的例子

计算机应用编程实验2018

简单爬虫流程

- **waitL=null, fetchedL=null**
- **While waitL不为null:**
 - 取出waitL中的队列头url
 - send()发送请求到服务器
 - Page=recv (url) 等待返回网页内容
 - fetchedL.add(url)
 - Ulist = parseurl(page)
 - Ulist=Ulist-fetchedL)//去重
 - waitL=waitL.add(Ulist)

计算机应用编程实验2018

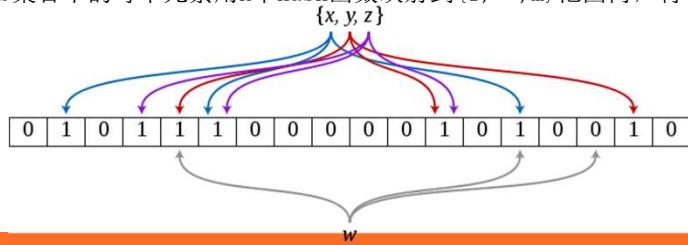
链接解析

- **HTML页面字符集识别**
- **URL提取**
 - 相对路径转换为绝对路径
 - 方法一：原始字符串匹配和暴力分析
 - String match "<a href=", use strstr() function
 - 可能遇到以下类型的标签
 - 的标签
 - <a ... href="javascript:void(0)" ...>
 - <a ... href=".//a.html" ...>
 - <a ... href='a.html' ...>
 - 不完备容易漏掉
 - 方法二：正则表达式
 - Regex正则表达式库
 - 正则表达式要考虑周全、计算复杂度高

计算机应用编程实验2018

基于Bloom Filter的URL去重

- **概念**
 - 1970年Burton Bloom提出，用于检索一个元素是否在一个集合中
 - 准确率换空间思想延伸，空间效率和查询时间非常高，有一定的误识别率
- **定义**
 - 包含n个元素集合 $S=\{x_1, x_2, \dots, x_n\}$
 - 一个包含m位的二进制位数组存储
 - K个相互独立的哈希函数映射到 $\{1, \dots, m\}$ 的范围
 - S集合中的每个元素用k个hash函数映射到 $\{1, \dots, m\}$ 范围内，将相应的位置为1



计算机应用编程实验2018

Bloom Filter原理

初始化时 m 位数组置零

B	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-----	---

对集合中的每个元素 x_j 分别进行 k 次hash, If $H_k(x_j) = a$, set $B[a] = 1$.

B	0 1 0 0 1 0 1 0 0 1 1 1 1 0 1 1 0
-----	---

要检测 y 是否在集合 S , 测试所有 k 个 $B[H_k(y)]$ 是否都为1.

B	0 1 0 0 1 0 1 0 0 1 1 1 1 0 1 1 0
-----	---

可能出现false positive: 即所有 k 个值都是1, 但 y 不在集合 S 中

B	0 1 0 0 1 0 1 0 0 1 1 1 1 0 1 1 0
-----	---

n items

$m = cn$ bits

k hash functions

21

计算机应用编程实验2018

错误率估计

- 初始位向量为0
- 插入一个元素后, 被 k 个哈希函数 (完全独立) 映射到位向量后, 某一位还是为0的概率是: $\left(1 - \frac{1}{m}\right)^k$.
- 集合 $S = \{x_1, x_2, \dots, x_n\}$ 的所有元素都插入到位向量后, 某一位还是为0的概率就是 $p' = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m}$. $\lim_{x \rightarrow \infty} \left(1 - \frac{1}{x}\right)^{-x} = e$
- 令 ρ 为位向量中0的比例, 则 ρ 的数学期望 $E(\rho) = p'$ 。令 $p = e^{-kn/m}$, 在 p 已知时要求的错误率 (false positive rate) 为:

$$(1 - \rho)^k \approx (1 - p')^k \approx (1 - p)^k. \quad f = \left(1 - e^{-kn/m}\right)^k = (1 - p)^k.$$

$(1 - \rho)^k$ 为位数组中1的比例, $(1 - p)^k$ 就表示 k 次哈希都刚好选中1的区域

M. Mitzenmacher 已经证明位向量中0的比例非常集中地分布在它的数学期望值的附近

计算机应用编程实验2018

最优的哈希函数个数k

• 问题

- Bloom Filter要用多个哈希函数将集合映射到位向量中，应该选择几个哈希函数才能使元素查询时的错误率降到最低？
- 对 $f = \left(1 - e^{-kn/m}\right)^k = (1-p)^k$.
- 令 $g = k \ln(1 - e^{-kn/m})$ ，让 g 取到最小， f 自然也取到最小

□ 将 g 写成

$$g = -\frac{m}{n} \ln(p) \ln(1-p),$$

• 结果

- 很容易看出当 $p = 1/2$ ，也就是 $k = \ln 2 \cdot (m/n)$ 时， g 取得最小值
- 在这种情况下，最小错误率 f 等于 $(1/2)^k \approx (0.6185)^{m/n}$
- 另外，注意到 p 是位数组中某一位仍是0的概率，所以 $p = 1/2$ 对应着位数组中0和1各一半

计算机应用编程实验2018

位向量大小m

• 问题

- 在不超过一定错误率的情况下，Bloom Filter至少需要多少位才能表示全集中任意 n 个元素的集合。假设全集中共有 u 个元素，允许的最大错误率为 ϵ ，求位数组的位数 m 。
- 一个确定的位向量可以表示的集合个数： $\binom{n + \epsilon(u-n)}{n}$
- m 位的位数组可以表示的集合个数： $2^m \binom{n + \epsilon(u-n)}{n}$
- 全集中 n 个元素的集合个数： $\binom{u}{n}$
- 要让 m 位的位数组能够表示所有 n 个元素的集合，必须有 $2^m \binom{n + \epsilon(u-n)}{n} \geq \binom{u}{n}$
- 即： $m \geq \log_2 \frac{\binom{u}{n}}{\binom{n + \epsilon(u-n)}{n}} \approx \log_2 \frac{\binom{u}{n}}{\binom{\epsilon u}{n}} \geq \log_2 \epsilon^{-n} = n \log_2(1/\epsilon)$
- 上页曾算出当 $k = \ln 2 \cdot (m/n)$ 时错误率 f 最小，这时 $f = (1/2)^k = (1/2)^{m \ln 2 / n}$ 。现在令 $f \leq \epsilon$ ，可以推出，约大了 $m \geq n \frac{\log_2(1/\epsilon)}{\ln 2} = n \log_2 \epsilon \cdot \log_2(1/\epsilon)$.

计算机应用编程实验2018

测试与参数选择

- 进行3组实验，每组取5个N
 - 取FP1 =0.01%， N=[50W, 100W, 300W, 500W, 1000W]
 - 取FP 2=0.001%， N=[50W, 100W, 300W, 500W, 1000W]
 - 取FP3 =0.0001%， N=[50W, 100W, 300W, 500W, 1000W]

N	(Vector size)m			内存			{Hash num} k		
	FP1	FP2	FP3	FP1	FP2	FP3	FP1	FP2	FP3
50W	958W	1198 W	1677W	1M	1M	1M	13	17	23
100W	1917 W	2396 W	3355W	2M	2M	3M	13	17	23
300W	5751 W	7188 W	10064 W	6M	8M	11M	13	17	23
500W	9585 W	11981 W	16773 W	11 M	14M	19M	13	17	23
1000W	19170 W	23962W	33547 W	22 M	28M	39M	13	17	23

计算机应用编程实验2018

Hash算法选择

- K个hash越相互独立效果越好
- 常用的HASH算法
 - <http://www.partow.net/programming/hashfunctions/>
 - unsigned int RSHash (char* str, unsigned int len);
 - unsigned int JSHash (char* str, unsigned int len);
 - unsigned int PJWHash (char* str, unsigned int len);
 - unsigned int ELFHash (char* str, unsigned int len);
 - unsigned int BKDRHash (char* str, unsigned int len);
 - unsigned int SDBMHash (char* str, unsigned int len);
 - unsigned int DJBHash (char* str, unsigned int len);
 - unsigned int DEKHash (char* str, unsigned int len);
 - unsigned int BPHash (char* str, unsigned int len);
 - unsigned int FNVHash (char* str, unsigned int len);
 - unsigned int APHash (char* str, unsigned int len);

计算机应用编程实验2018

怎样更快？

计算机应用编程实验2018

阻塞I/O模式

● 模型

□ 创建时默认阻塞模式，当socket不能立即完成I/O操作时，进程或线程进入等待

状态，直到操作完成

● 以recv函数为例，

□ 阻塞模式下，程序调用了recv函数后将一直处于等待状态，直到接收完数据



计算机应用编程实验2018

非阻塞I/O模式

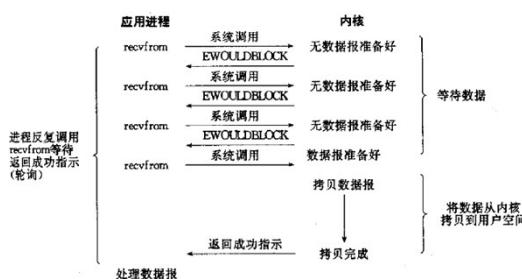
• 语义

- 把socket设置成非阻塞模式，无数据时，也不会进入等待，而是立即返回特定错误

• 实现方法

- 使用ioctlsocket设置非阻塞模式
- 示例代码：

```
U_long ul = 1;  
SOCKET s =  
socket(AF_INET, SOCK_STREAM, 0);  
ioctlsocket(s, FIONBIO, (u_long)
```



这种模式在没有数据可以接收时，可以进行其他的一些操作，比如有多个socket时，可以去查看其他socket有没有可以接收的数据；
实际应用中，它需要不停的查询，而这些查询大部分会是不必要的调用，白白浪费了系统资源；非阻塞I/O是一个铺垫，为I/O复用和信号驱动奠定了非阻塞使用的基础

计算机应用编程实验2018

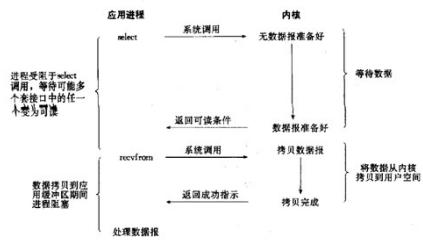
I/O复用 (multiplexing)

• 思想

- 查询多个socket可读或可写是否准备好的状态
 - 并发服务器管理多个客户连接IO、本地文件IO等
 - 要求为非阻塞Socket

• 实现方式

- Select经典方式(*nix, Windows)
- poll、基于内核通知的epoll (Linux)
- kqueue (freebsd)
- Epoll和kequeue多连接时高性能



计算机应用编程实验2018

Select方法

- 功能

- 确定一个或多个套接字是否有连接到达、已连接socket是否有数据到达、已连接socket是否可以写数据，以便执行同步I/O

- 实现

- 用户态实现
- 最大并发数限制为1024
- 线性扫描全FD 集合效率低



```
#include <sys/select.h>
#include <sys/time.h>

将套接字集合清空 FD_ZERO(*set)
将某个套接字从集合中清除 FD_CLR(s, *set)
检查套接字 是否在集合set中 FD_ISSET(s, *set)
将套接字放入集合set中 FD_SET(s, *set)

int select(
    int nfds, //忽略, 仅为了与berkaley套接字兼容
    fd_set * readfds, //一个套接字集合, 用于检查可读性
    fd_set * writefds, //一个套接字集合, 用于检查可写性
    fd_set * exceptfds, //一个套接字集合, 用于检查错误
    const struct timeval * timeout //指定此函数等待的最长时间, 若为NULL, 则最长时间为无限大
)
当有 I/O 事件到来时, select 通知应用程序有事件到了快去处理, 而应用程序必须轮询所有的 FD 集合, 测试每个 FD 是否有事件发生, 并处理事件: 代码像下面这样:
int res = select(maxfd+1, &readfds, NULL, NULL, 120);
if (res > 0){
    for (int i = 0; i < MAX_CONNECTION; i++){
        if (FD_ISSET(allConnection[i], &readfds)){
            handleEvent(allConnection[i]);
        }
    }
}
// if(res == 0) handle timeout, res < 0 handle error
```

计算机应用编程实验2018

Epoll/poll方法

- 功能

- Select类似, 确定哪些IO准备好了
- 用一个结构保存要监视的每个连接的数组, 当在网络套接字上发现数据时, 通过回调机制调用处理函数。

- 优缺点

- 不限制FD集合大小
- 避免了FD轮询
- 内核事件通知效率高
- 结构会非常大, 在列表中添加新的网络连接时, 修改结构会增加负载并影响性能

```
int epoll_create(int size);
```

生成一个 Epoll 专用的文件描述符, 其实是申请一个内核空间, 用来存放你想关注的 socket fd 上是否发生以及发生了什么事。size 就是你在这个 Epoll fd 上能关注的最大 socket fd 数, 大小自定, 只要内存足够。

```
int epoll_ctl(int epfd, int op, int fd, struct epoll_event *event );
```

控制某个 Epoll 文件描述符上的事件: 注册、修改、删除。其中参数 epfd 是 epoll_create() 创建 Epoll 专用的文件描述符。相对于 select 模型中的 FD_SET 和 FD_CLR 宏。

```
int epoll_wait(int epfd,struct epoll_event * events,int maxevents,int timeout);
```

等待 I/O 事件的发生; 参数说明:
epfd: 由 epoll_create() 生成的 Epoll 专用的文件描述符;
epoll_event: 用于回传待处理事件的数组;
maxevents: 每次能处理的事件数;
timeout: 等待 I/O 事件发生的超时值;
返回发生事件数。
#include <sys/select.h>
#include <sys/time.h>
int res = epoll_wait(epfd, events, 20, 120);

for (int i = 0; i < res;i++)
{
 handleEvent(events[n]);
}

计算机应用编程实验2018

并发爬虫流程

- `waitL=null, fetchedL=null, max_conn=100`
- While `waitL不为null:`
 - While 当前连接数小于100
 - 取出`waitL`中的队列头url
 - 创建socket加入fd集合, send请求
 - Epoll轮询fd集合的所有事件, 对每个事件
 - Case Socket连接成功: 发送send请求
 - Case send返回响应:
 - 调用recv读取返回数据,
 - `page=recv(url)`
 - `fetchedL.add(url)`
 - `Ulist = parseurl(page)`
 - `Ulist=Ulist-(Ulist AND fetchedL)//去重`
 - `waitL=waitL.add(Ulist)`
 - Case 关闭socket: 当前连接减1

计算机应用编程实验2018

基于Libevent异步I/O库的并发抓取

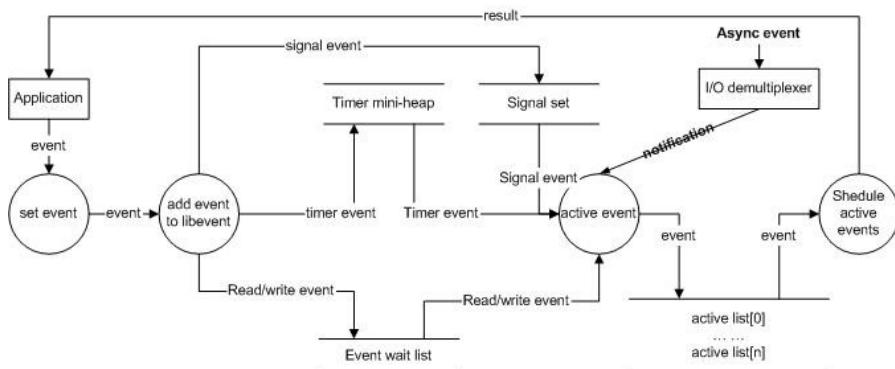
- 概述
 - 一个基于事件触发的网络库
- 特点
 - 事件驱动 (event-driven), 高性能;
 - 轻量级, 专注于网络
 - 跨平台, 支持Windows、Linux、*BSD和Mac Os;
 - 支持多种I/O多路复用技术, epoll、poll、dev/poll、select和kqueue等;
 - 支持I/O, 定时器和信号等事件;
- 应用
 - Libevent已被广泛应用, 作为底层的网络库; 比如memcached、Vomit、Nylon、Netchat等等。
 - Libevent代码里有很多有用的设计技巧和基础数据结构, 比如信息隐藏、函数指针、c语言的多态支持、链表和堆等。

计算机应用编程实验2018

Libevent原理

• 运行原理

- 使用Libevent也是向Libevent框架注册相应的事件和回调函数
- 当这些事件发生时，Libevent 会调用这些回调函数处理相应的事件（I/O读写、定时和信号）。



计算机应用编程实验2018

```
struct event {
    TAILQ_ENTRY(event) ev_next;
    TAILQ_ENTRY(event) ev_active_next;
    TAILQ_ENTRY(event) ev_signal_next;
    unsigned int min_heap_idx; /* for managing timeouts */
    struct event_base *ev_base;
    int ev_fd;
    short ev_events;
    short ev_ncalls;
    short *ev_pncalls; /* Allows deletes in callback */
    struct timeval ev_timeout;
    int ev_pri; /* smaller numbers are higher priority */
    void (*ev_callback)(int, short, void *arg);
    void *ev_arg;
    int ev_res; /* result passed to event callback */
    int ev_flags;
};
```

向libevent添加事件，需先设置event对象，通过调用libevent提供的函数有：

```
void event_set(struct event *ev, int fd, short events, void
(*callback)(int, short, void *),
void *arg)
```

1. 设置事件ev绑定的文件描述符或者信号，对于定时事件，设为-1即可；
2. 设置事件类型，比如EV_READ|EV_PERSIST, EV_WRITE, EV_SIGNAL等；
3. 设置事件的回调函数以及参数arg；
4. 初始化其它字段，比如缺省的event_base和优先级；

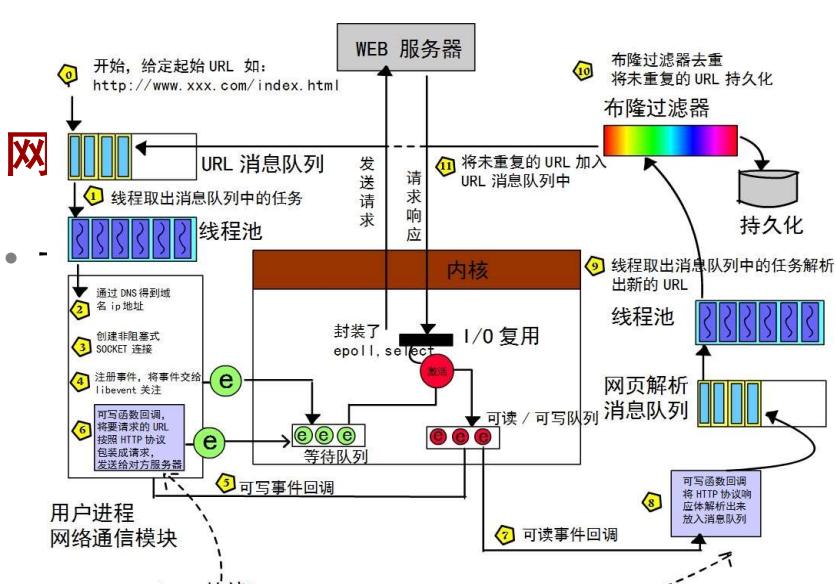
```
int main(int argc, char **argv)
{
    ...
    ev_init();
    /* Setup listening socket */
    event_set(&ev_accept, listen_fd, EV_READ|EV_PERSIST, on_accept, NULL);
    event_add(&ev_accept, NULL);
    /* Start the event loop. */
    event_dispatch();
}
```

计算机应用编程实验2018

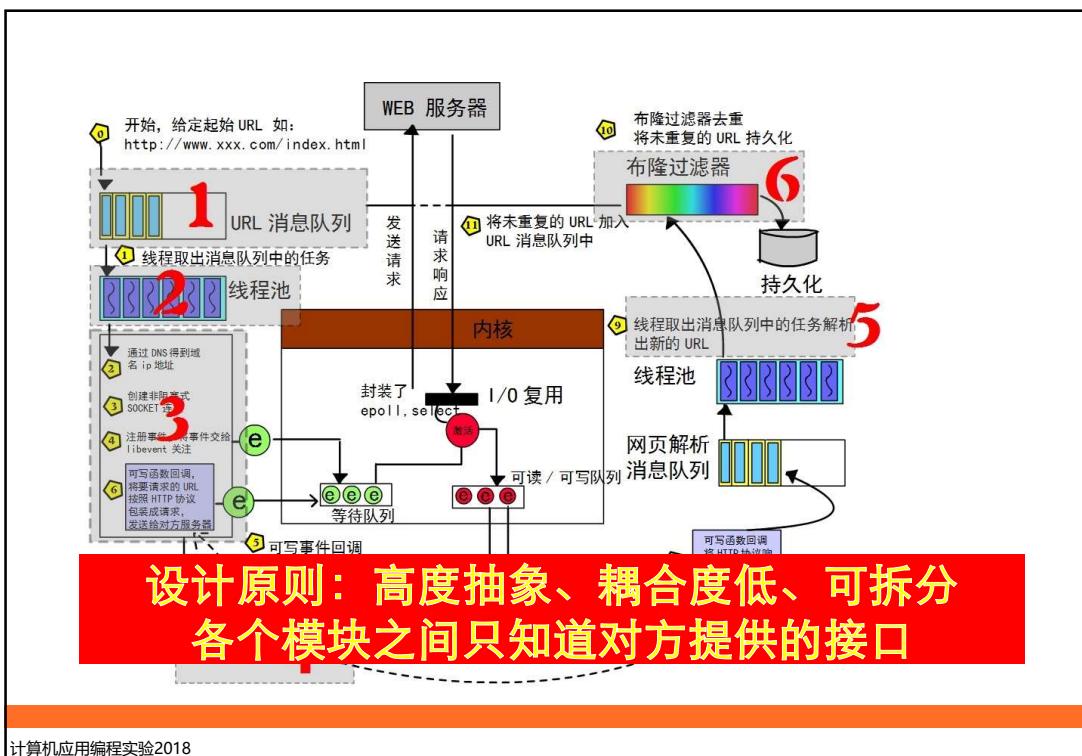
几个实现要点

- Connection: keep-alive
 - 在HTTP响应的头部字段中有一项keep-alive选项，服务器不会马上断开连接，需要对数据何时结束进行判断
 - 静态页面：Content-Length：
 - 动态页面：Transfer-Encoding: chunked
- Accept-Encoding: gzip, deflate
 - 传输编码为gzip，需要对gzip编码在内存中进行解压，然后进行分析
 - 利用zlib库进行解压
- HTTP响应处理
 - 只存储200 OK成功响应，其他无视
- URL去重
 - 检查某个URL是否已经被抓过了
 - 基于Bloom filter

计算机应用编程实验2018



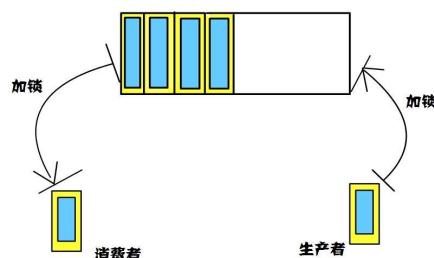
计算机应用编程实验2018



计算机应用编程实验2018

消息队列

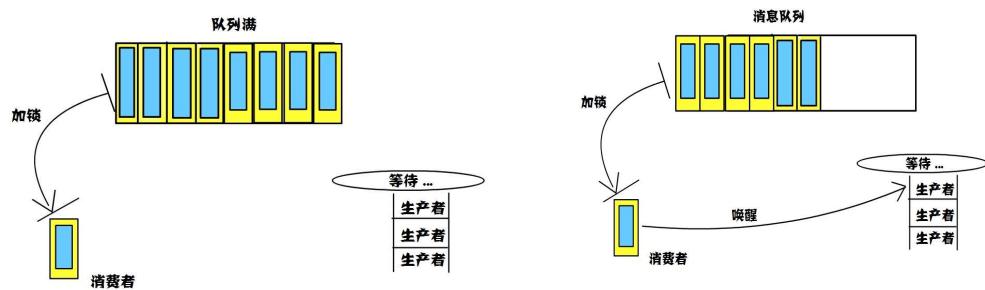
- 我们采用两把锁和两个条件变量来实现阻塞式的消息队列
 - 使用的是生产-消费者模式。



计算机应用编程实验2018

消息队列

- 当队列满时，生产者进行等待，阻塞生产。当消费者消费了消息后，对等待队列中的生产者进行唤醒，生产继续。



计算机应用编程实验2018



计算机应用编程实验2018

```

#define MAX_MESSAGE_NUM 10
struct msg
{
    int id;
    char* data;//response body or url
    struct msg * next;
};

struct mqueue
{
    int qsize;
    struct msg *next;
    struct msg *tail;
    pthread_mutex_t _not_full_mutex;
    pthread_mutex_t _not_empty_mutex;
    pthread_cond_t _not_full_cond;
    pthread_cond_t _not_empty_cond;
};

void init_message_queue(struct mqueue *msgq);
struct msg * get_message(struct mqueue *msgq);
void add_message(struct mqueue *msgq,char * data,int id);

```

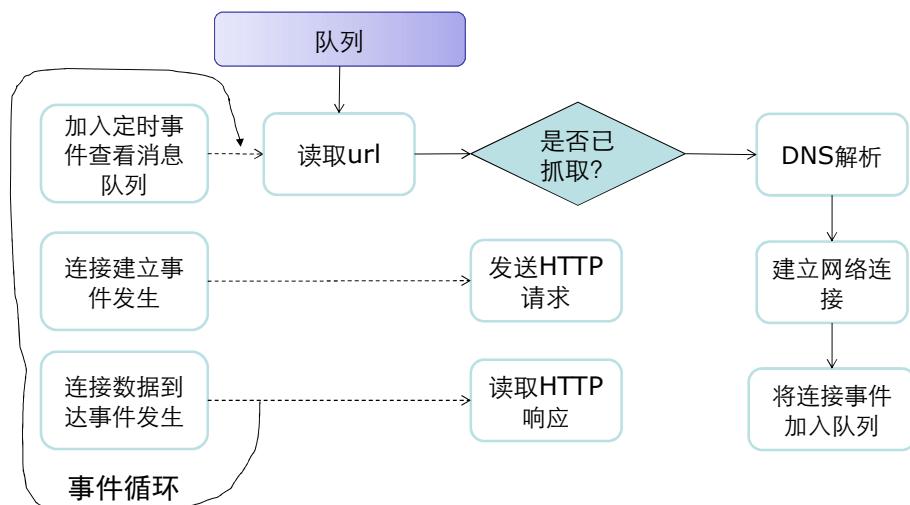
```

BiRongdeMacBook-Pro:messageQueue birong$ ./main
create queue
add success : id: 0 size: 1
add success : id: 1 size: 2
add success : id: 2 size: 3
add success : id: 3 size: 4
add success : id: 4 size: 5
add success : id: 5 size: 6
add success : id: 6 size: 7
add success : id: 7 size: 8
get success id: 0 size : 7

```

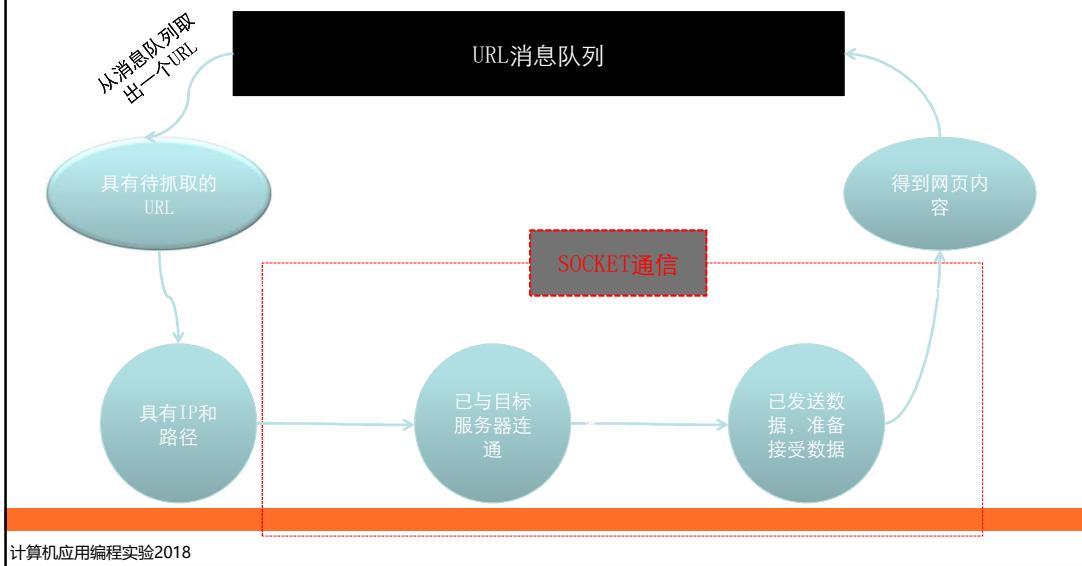
计算机应用编程实验2018

爬取模块总体流程

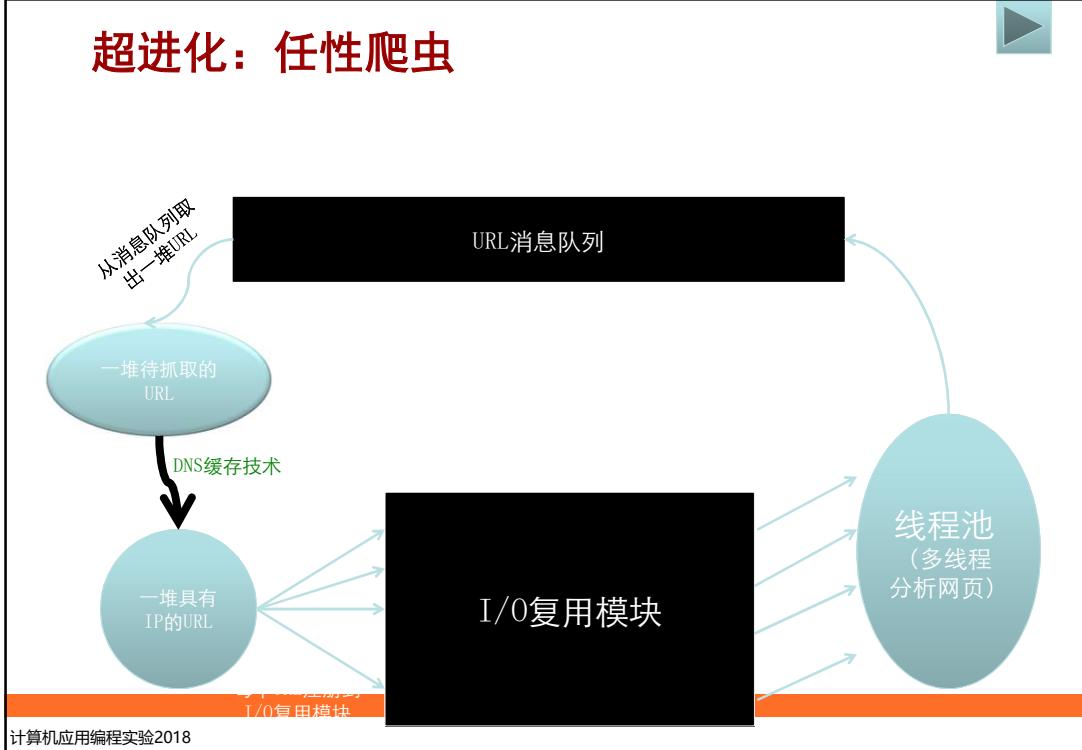


计算机应用编程实验2018

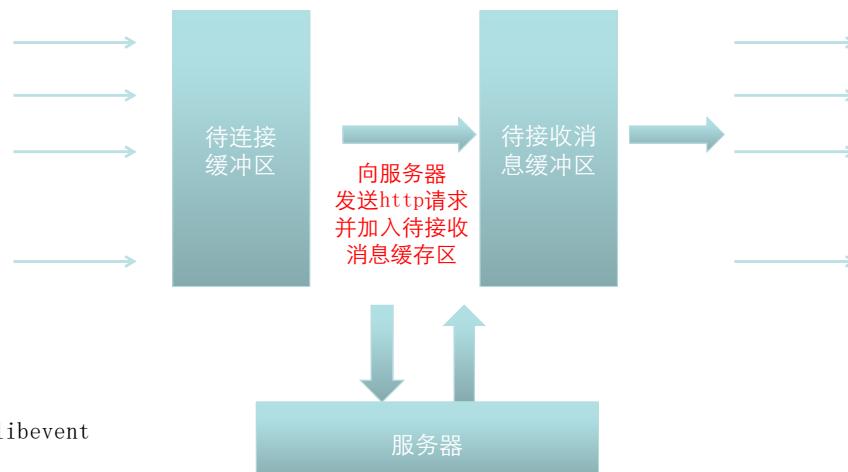
初级形态：单线程爬虫



超进化：任性爬虫

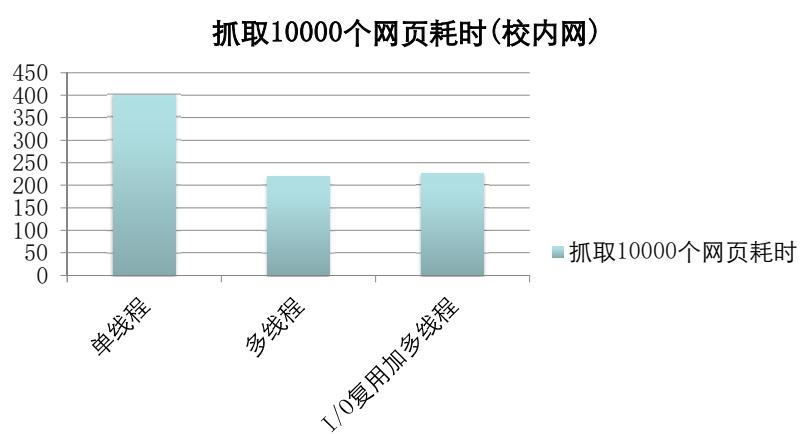


I/O复用技术

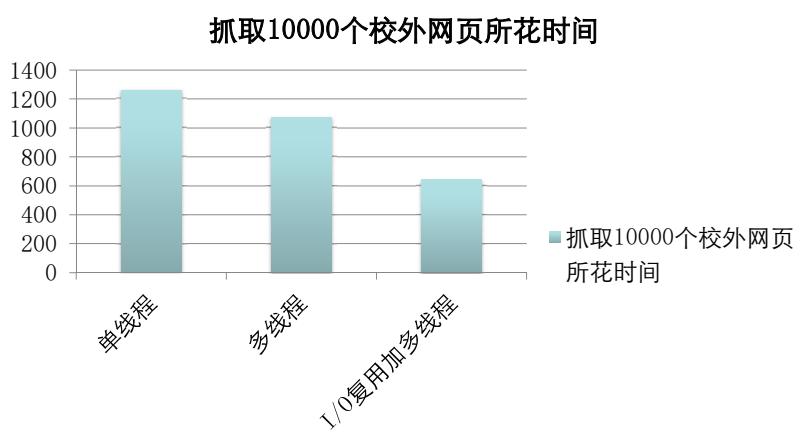


计算机应用编程实验2018

看看那个更6



计算机应用编程实验2018



计算机应用编程实验2018



PageRank身世

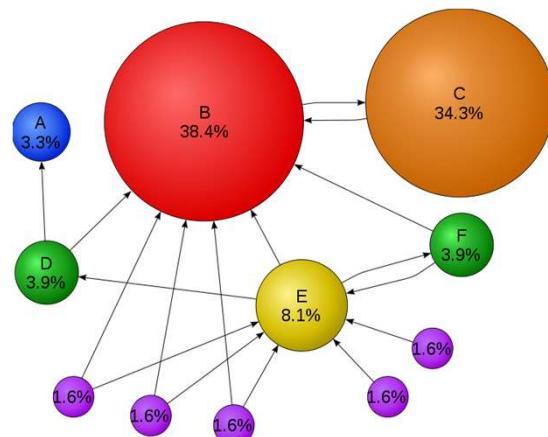


- 提出
 - Google的创始人之一Larry Page于1998年提出了PageRank，并应用在Google搜索引擎的检索结果排序上，该技术也是Google早期的核心技术之一
 - 有向图上的特征向量中心性

- 核心思想
 - 一个节点的“得票数”由所有链向它的节点的重要性来决定，到一个节点的边相当于对该节点投一票。
 - 一个节点的PageRank是由所有链向它的节点的重要性经过递归算法得到的。
 - 一个有较多链入的节点会有较高的等级，相反如果一个节点没有任何链入边，那么它没有等级。

计算机应用编程实验2018

PageRank

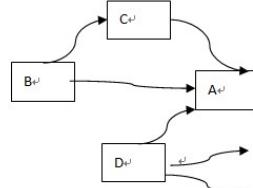


PageRank 是基于「**从许多优质的网页链接过来的网页，必定还是优质网页**」的回归关系，来判定所有网页的重要性。

链向网页E的链接远远多于链向网页C的链接，但是网页C的重要性却大于网页E。这是因为因为网页C被网页B所链接，而网页B有很高的重要性。

计算机应用编程实验2018

PageRank简单计算



- 假设一个由只有4个页面组成的集合：**A, B, C和D**。如果所有页面都链接到**A**，那么**A**的PR（PageRank）值将是**B, C及D**的和。

$$PR(A) = PR(B) + PR(C) + PR(D)$$

- 继续假设**B**也有链接到**C**，并且**D**也有链接到包括**A**的3个页面。一个页面不能投票2次。所以**B**给每个页面半票。以同样的逻辑，**D**投出的票只有三分之一算到了**A**的PageRank上。

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}$$

- 换句话说，根据链出总数平分一个页面的PR值。

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}$$

计算机应用编程实验2018

PR形式化表示

- 定义邻接矩阵为G，若网页j到网页i有超链接，则 $g_{ij} = 1$ ；反之， $g_{ij} = 0$
- 设共有m个网页，分别编号为1、2、3、...、m，它们的级别（重要性）分别记为 $r_1, r_2, r_3, \dots, r_m$ ，G表示由这些网页组成的有向图的邻接矩阵。根据有向图理论：

$$r(u) = \sum_{v \in B_u} \frac{r(v)}{n_v} \rightarrow r_i = \sum_{j=1}^m \frac{g_{ij}}{n_j} r_j$$

G中第j列的列和

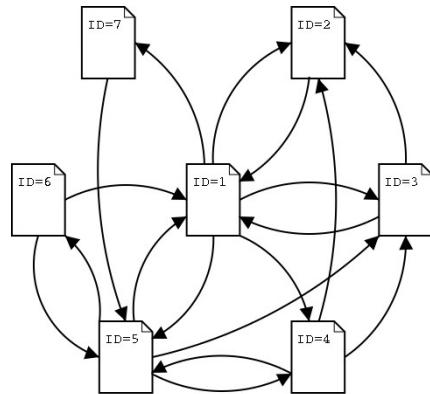
$$r = G_m \cdot r$$

$$\begin{cases} r = (r_1, r_2, \dots, r_m)^T \\ G_m = \{g_{ij} / n_j\} \end{cases}$$

可知 r 是 G_m 的对应于特征值为 1 的特征向量

计算机应用编程实验2018

7个网页的链接关系图与邻接矩阵



$$G = \begin{matrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

计算机应用编程实验2018

PageRank的计算：Gm矩阵

$$G_m = \begin{matrix} 0 & 1 & 1/2 & 0 & 1/4 & 1/2 & 0 \\ 1/5 & 0 & 1/2 & 1/3 & 0 & 0 & 0 \\ 1/5 & 0 & 0 & 1/3 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 1/3 & 0 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

计算机应用编程实验2018

PageRank的计算

求矩阵Gm特
征值1对应的
特征向量

0. 699456533837389
0. 382860418521518
0. 323958815672054
0. 242969111754040
0. 412311219946251
0. 103077804986563
0. 139891306767478

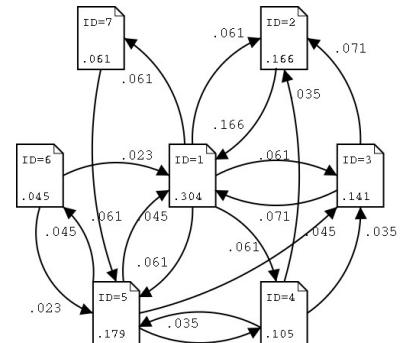
归一化
→

0. 303514376996805
0. 166134185303514
0. 140575079872204
0. 105431309904153
0. 178913738019169
0. 0447284345047923
0. 0607028753993610

计算机应用编程实验2018

PageRank结果

名次	PageRank	文件 ID	发出链接 ID	被链接 ID
1	0. 304	1	2, 3, 4, 5, 7	2, 3, 5, 6
2	0. 179	5	1, 3, 4, 6	1, 4, 6, 7
3	0. 166	2	1	1, 3, 4
4	0. 141	3	1, 2	1, 4, 5
5	0. 105	4	2, 3, 5	1, 5
6	0. 061	7	5	1
7	0. 045	6	1, 5	5



分析

- ID=1 的页面的PageRank 是0. 304，占据全体的三分之一。
- 特别需要说明的是，起到相当大效果的是从排在第3位的 ID=2 页面中得到了所有的PageRank (0.166) 数。ID=2页面有从3个地方过来的链入链接，而只有面向 ID=1页面的一个链接，因此(面向ID=1页面的)链接就得到ID=2的所有的PageRank 数。
- ID=1页面是链出链接和链入链接最多的页面，它是最受欢迎的页面。
- 即使有同样链入链接的数目，链接源页面评价的高低也影响 PageRank 的高低。

计算机应用编程实验2018

58

PageRank是求解 Gm 的特征值为1的特征向量

1、矩阵 G_m 一定有特征值1吗？即上面的方程是否有解？

如果 $G = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ ，则 $r_1 = r_2$ ，此时就无法进行求解

从代数角度

- **Perron-Frobenius定理**

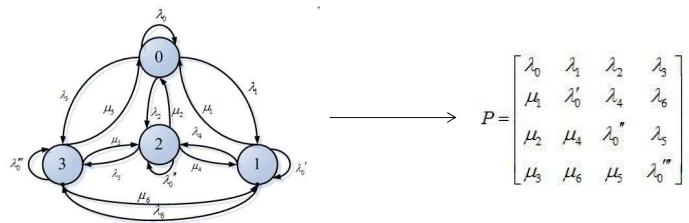
- 随机矩阵
- 不可约
- 非周期
- 结论
 - 该方程组解存在且唯一
 - x 是 A 的最大特征值1所对应的特征向量

如何构造 A 使之不可约？

从时间角度

- 马尔可夫过程

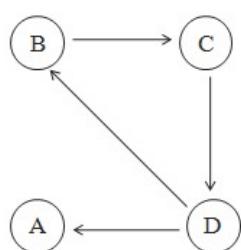
- “将来只由现在决定，和过去无关”
- 考虑一个图，图上每个点有一个值，会被不断更新。每个点通过一些边连接到其它一些点上，对于每个点，这些边的值都是正的，和为1。在图上每次更新一个点的值，就是对和它相连接的点的值加权平均。



- 如果图是联通并且非周期（数学上叫各态历经性，ergodicity），那么这个过程最后会收敛到一个唯一稳定的状态（平衡状态）。

计算机应用编程实验2018

举个栗子—Rank Leak



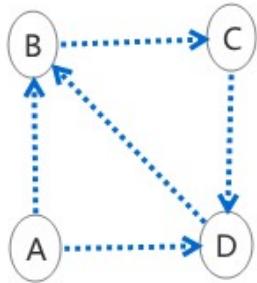
	PR(A)	PR(B)	PR(C)	PR(D)
初始	0.25	0.25	0.25	0.25
一次迭代	0.125	0.125	0.25	0.25
二次迭代	0.125	0.125	0.125	0.25
三次迭代	0.125	0.125	0.125	0.125
...
n次迭代	0	0	0	0

Rank leak: 一个独立的网页如果没有外出的链接就产生等级泄漏
解决办法：

1. 将无出度的节点递归地从图中去掉，待其他节点计算完毕后再加上
2. 对无出度的节点添加一条边，指向那些指向它的顶点

计算机应用编程实验2018

举个栗子—Rank Sink



	PR(A)	PR(B)	PR(C)	PR(D)
初始	0.25	0.25	0.25	0.25
一次迭代	0	0.375	0.25	0.375
二次迭代	0	0.375	0.375	0.25
三次迭代	0	0.25	0.375	0.375
四次迭代	0	0.375	0.25	0.375
五次迭代	0

Rank sink: 整个网页图中的一组紧密链接成环的网页如果没有外
出的链接就产生**Rank sink**

计算机应用编程实验2018

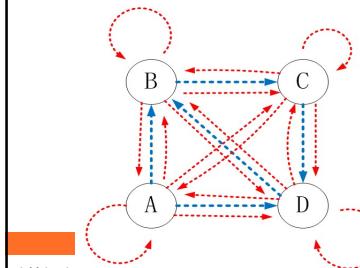
随机浏览修正

• 不可约

- 当浏览器所浏览的网页矩阵存在不可达或周期连通分量时，该浏览器将无法继续浏览其它网页。

• 修正

- 每次访问网页时，可以随机选择一个其它的网页重新开始浏览
 - ① 这种随机模型更加接近于用户的浏览行为
 - ② 一定程度上解决了rank leak和rank sink的问题
 - ③ 保证pagerank具有唯一值



设定任意两个顶点之间都有直接通路，
在每个顶点处以概率 α 按原来蓝色方向转移，以概率 $1-\alpha$ 按红色方向转移。

64

计算机应...

修正模型

- 让浏览者每次以一定的概率 α 沿着链接走，以概率 $(1-\alpha)$ 重新随机选择一个新的起始节点
- α 选在0.1和0.2之间，被称为阻尼系数
- 矩阵 $M = (1-\alpha)G_m + \alpha/N(1_N)$ 满足不可约特性，存在平稳分布 r ，

$$r = \left((\alpha \times G_m + \frac{1-\alpha}{n}(1_N)) \right) r$$

一般取0.15

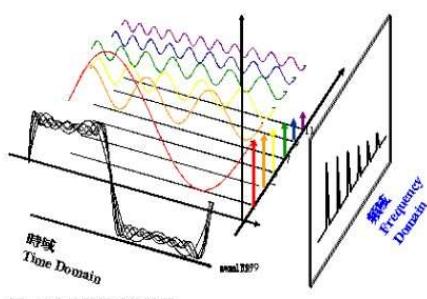
各元素均为1的N阶矩阵

计算机应用编程实验2018

矩阵 G_m 特征向量怎么求？

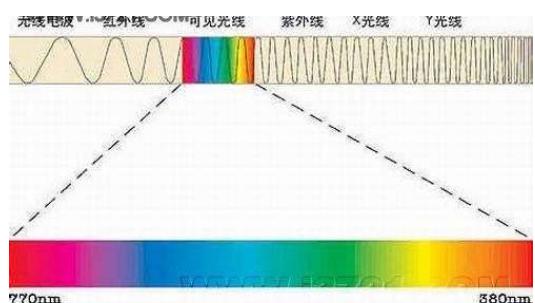
- 特征向量计算规模是 $O(n^3)$
- 特征向量的求解，就是求解方程 $A\alpha = \alpha$ 是 N 元一次方程组，一般不能得到分析解，所以只能解其数值。

特征向量的内涵-谱



圖二 時域與頻域的差異

- 普通的线性代数课本会告诉你定义：如果 $A v = c v$, 那么 c 就是 A 的特征值, v 就叫特征向量。
- 这仅仅是数学家发明的一种数学游戏么？
- 如果我们把这个东西看成是一些分量叠加而成，那么这些分量以及它们各自所占的比例，就叫这个东西的谱。



计算机应用编程实验2018

矩阵的谱结构

• 矩阵谱

- 它的特征值和特征向量
- 谱代表了一种分量结构，它使用“分而治之”策略来研究矩阵
- 把矩阵理解为一个变换，它的作用就是把一个向量变成另外一个向量： $y = Ax$
- 对于某些向量， $A v = cv$, 相当于就把这个向量 v 拉长了 c 倍。把这种和矩阵 A 能如此密切配合的向量 v_1, v_2, \dots 叫做特征向量，这个倍数 c_1, c_2, \dots 叫特征值。
- 当出现一个新的向量 x 的时候，可以把 x 分解为这些向量的组合， $x = \alpha_1 v_1 + \alpha_2 v_2 + \dots$, 那么 A 对 x 的作用就可以分解： $Ax = A(\alpha_1 v_1 + \alpha_2 v_2 + \dots) = \alpha_1 c_1 v_1 + \alpha_2 c_2 v_2 \dots$
- 所以，矩阵的谱是用于分解一个向量。

计算机应用编程实验2018

迭代计算特征向量

- **计算方法**

- 设 A 是马尔可夫过程的转移概率矩阵，数学上可以严格证明，对于上述的转移概率矩阵，最大的特征值就是1，这里对应于平衡状态 v_1 ，其它的特征状态 v_2, v_3, \dots 对应于特征值 $1 > c_2 > c_3 > \dots >$
- 给定任意一个初始状态 v （各节点的值），迭代计算 $v(t+1) = A v(t)$ 。
- 把 v 分解成 $v = v_1 + c_2 v_2 + c_3 v_3 + \dots$ 。
- 可以看到，当更新进行了 t 步之后，状态变成 $v(t) = v_1 + c_2^t v_2 + c_3^t v_3 + \dots$ ，除了代表平衡状态的分量保持不变外，其它分量随着 t 增长而指数衰减
- 最后得到系统的稳态 $v = Av$ ，稳定状态就是 A 的一个特征向量，特征值就是1
- 收敛速度取决于第二大特征值 c_2 ， c_2 的大小越接近于1，收敛越慢，越接近于0，收敛越快。

计算机应用编程实验2018

幂迭代计算

- **解决方法—Power Iteration幂迭代**

当矩阵 A 的阶很大，无法直接计算其特征值和特征向量时，需要使用该方法

- 1) 输入矩阵 A 和迭代初始向量 v ，以及精度 $\epsilon > 0$ （例如0.0001，向量各元素对应差值绝对值），令 $k = 0$ ；
- 2) 计算： $v_{k+1} = Av_k$ ；
- 3) 如果 $|v_{k+1} - v_k| < \epsilon$ ，则计算 PageRank 值并停止。否则转第二步。

PageRank算法只有两步：构造矩阵A和迭代求解

计算机应用编程实验2018

幂迭代计算

- 幂迭代算法

- 确定合适的初始向量 v ，例如 $v = \text{indegree}(\text{node}) / |E|$
- 每次迭代是一次矩阵向量乘法复杂度 $> O(n^2)$ ，但 A 是稀疏矩阵，所以整个迭代速度非常快
- 收敛速度取决于 c_2
- 3亿个页面的Web Graph
→ 50 iterations to convergence (Brin and Page, 1998)

- 幂迭代算法的马尔可夫链模型

- page importance \Leftrightarrow steady-state Markov probabilities \Leftrightarrow eigenvector
- Larry Page和Sergey Brin 的贡献，一方面加入随机游走解决了矩阵收敛问题，另一方面由于互联网网页数量巨大，生成的二维矩阵巨大，两人利用稀疏矩阵计算简化了计算量。

计算机应用编程实验2018

Over?

矩阵 G_{π} 怎么计算？

- 假设 N 是 10000，通常数值计算程序内部行列和矢量是用双精度记录的， N 次正方行列 A 的存储空间为 $\text{sizeof(double)} * N * N = 8 * 104 * 104 = 800\text{MB}$
- 本实验 N 是 160000 个，方阵存储空间超过 200GB

稀疏矩阵

● 概念

- 设在矩阵A中，有s个非零元素。令 $e=s/(m*n)$ ，称e为矩阵的稀疏因子。
- 通常认为 $e \leq 0.05$ 时称之为稀疏矩阵。

● 存储方式

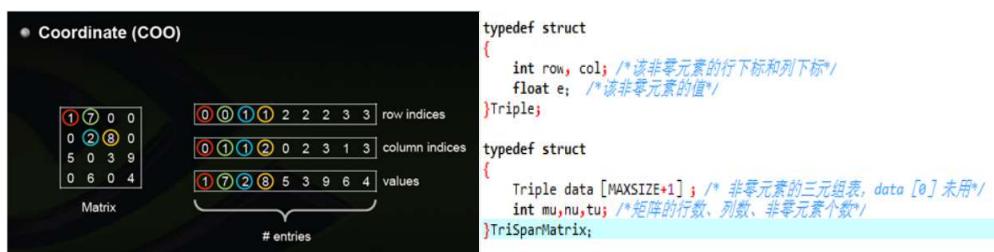
- 非零元素的分布一般没有规律，存储非零元素的同时，还必须同时记下它所在的行和列的位置 (i, j) 。
- 两种存储方式：
 - 三元组 (i, j, a_{ij}) 唯一确定了矩阵A的一个非零元。因此，稀疏矩阵可由表示非零元的三元组及其行列数唯一确定。
 - 十字链表方法，矩阵的每一个非零元素用一个结点表示，该结点除了 $(row, col, value)$ 以外，还要有以下两个链域：right：用于链接同一行中的下一个非零元素；down：用于链接同一列中的下一个非零元素。

计算机应用编程实验2018

稀疏矩阵存储

● 稀疏矩阵存储结构

- Coordinate (COO)
- 特点：格式简单，更加灵活，易于操作，常用于从文件中进行稀疏矩阵的读写，如matrix market即采用COO格式，但计算效率一般

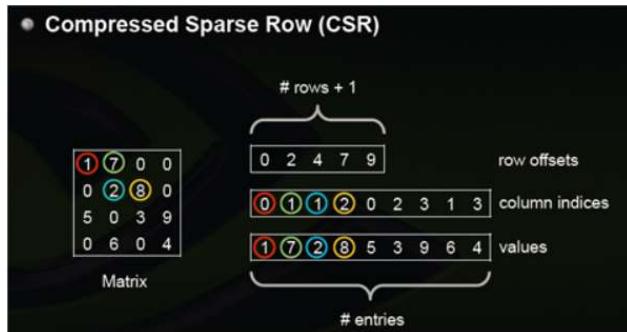


计算机应用编程实验2018

稀疏矩阵存储

- 稀疏矩阵存储结构

- Compressed Sparse Row (CSR)
- 特点：相对于COO，节省了行存储开销，常用于读入数据后进行稀疏矩阵计算

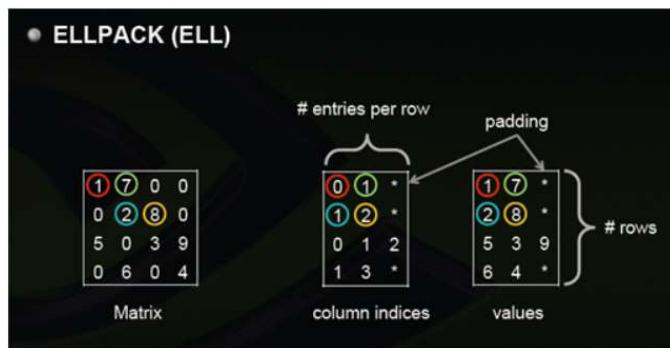


计算机应用编程实验2018

稀疏矩阵存储

- 稀疏矩阵存储结构

- ELLPACK (ELL)
- 特点：在进行稀疏矩阵-矢量乘积时效率最高，是应用迭代法解稀疏线性系统最快的格式；但如果某一行很多元素，那么后面两个矩阵就会很胖，其他行结尾*很多，浪费存储空间



计算机应用编程实验2018

稀疏矩阵存储

- 稀疏矩阵存储结构

- Diagonal (DIA)
- 特点：如果原始矩阵就是一个对角性很好的矩阵那压缩率会非常高，但对于随机矩阵很糟糕

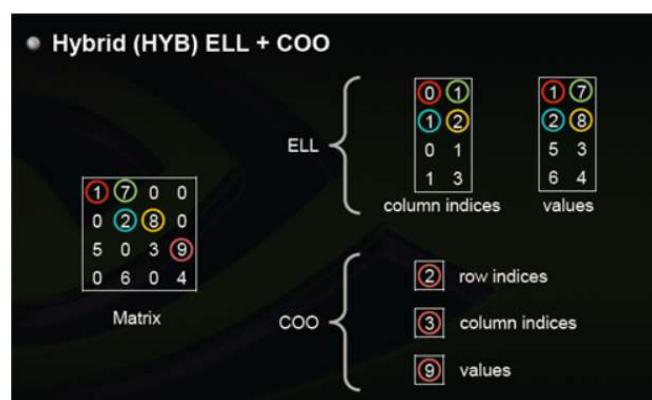


计算机应用编程实验2018

稀疏矩阵存储

- 稀疏矩阵存储结构

- Hybrid (HYB) ELL + COO
- 特点：ELL的优点是快速，而COO优点是灵活，二者结合后的HYB格式是一种不错的稀疏矩阵表示格式



计算机应用编程实验2018

实现要点

- 为了降低抓取开销，需要将抓取的链接url保存到文件
- 需要将稀疏矩阵保存到文件，以备重复计算

计算机应用编程实验2018



提交程序和实验报告

● 数据来源

- 大约16万个网页，不存在的链接过滤掉

文件总数 161,033
文件总大小 17,052,578,841
压缩包大小 3,898,532,313

● 程序要求

- 命令行格式

- ./crawler ip port url.txt

- ./pagerank url.txt top10.txt

- Input: ip port: web服务器ip和端口

- Output:

- url.txt: 一行一个url及其编号, 空行, 后面是url链接关系, 一行两个编号表示链接关系

- Top10.txt: top 10的url, 一行一个, 每行: url rank值

● 实验报告

- 包括实验目标、系统设计实现、实验结果、结论等几个部分

- 主要数据结构

- 数据流程图

计算机应用编程实验2018

进度要求

● 实验进度

- W1: 熟悉linux环境和socket编程, 构建可以发送简单请求的程序
- W2: 在自己电脑上搭建nginx和网站环境, 开发可构造http请求协议的客户端, 可以抽取超链和利用bloomfilter去重
- W3: 熟悉libevent设计模式和epoll的io多路复用原理
- W4: 开发基于libevent的并行网页下载请求, 并保存url和链接关系到文件
- W5: 设计保存所有超链的图结构和文件结构, 并进行持久化存储到文件
- W6: 利用矩阵迭代算法计算pagerank并计算最重要的网页

计算机应用编程实验2018

