信息检索与Web搜索

第14讲 Web采集

Crawling

授课人：高曙明

改编自“现代信息检索”网上公开课件（http://ir.ict.ac.cn/~wangbin）

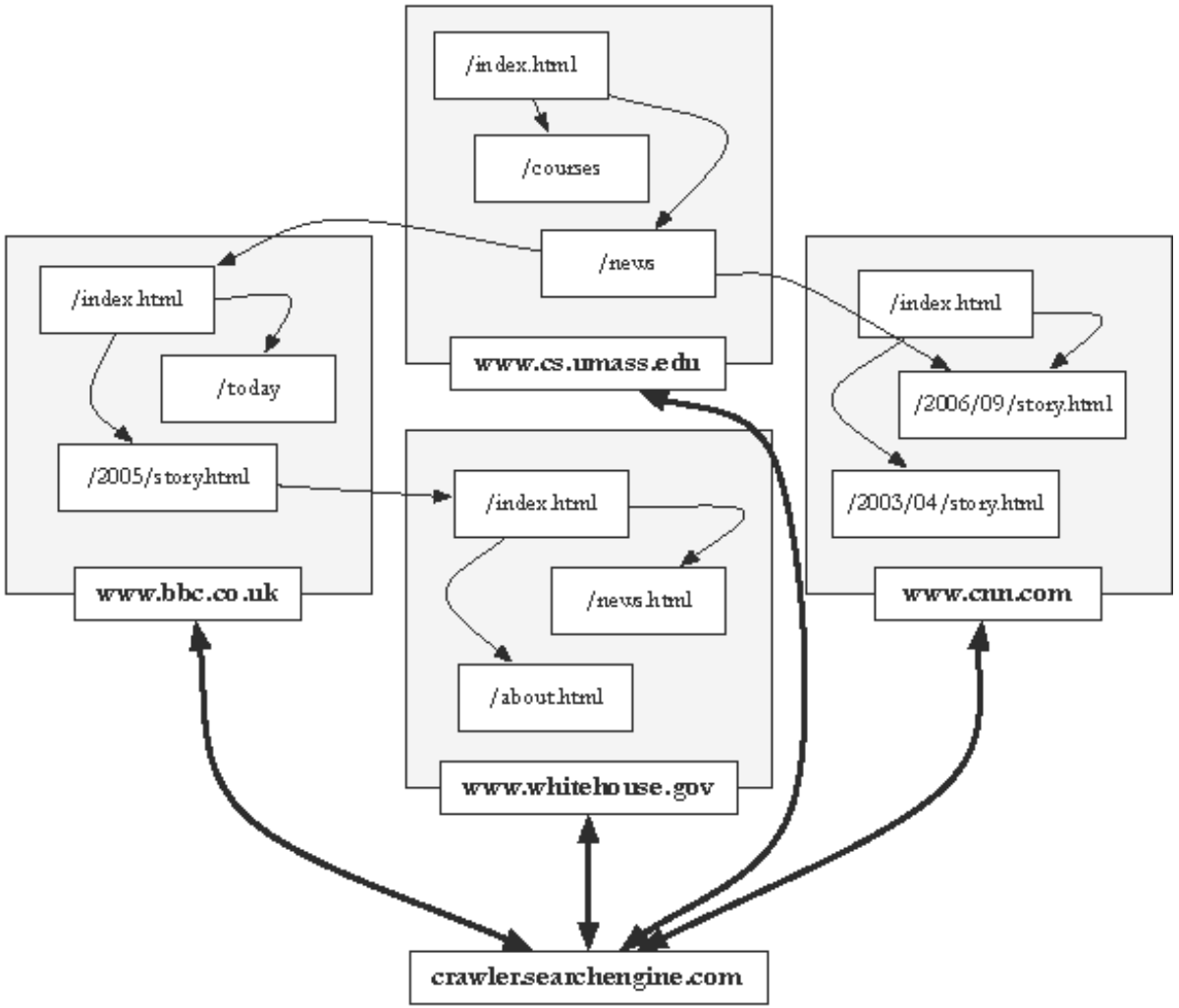
概述

- **Web采集：**是从Web中收集网页的过程
- **必要性：**因为没有集中的有组织的网页集
- **目标：**尽可能高效地采集更多数目的有用页面
- 相应的软件称为Web采集器、网络爬虫、网络蜘蛛（Web Crawler, spider）

基本的采集过程

- 初始化采集URL种子队列;
- 重复如下过程:
  - 从队列中取出URL
  - 下载并分析网页
  - 从网页中抽取更多的URL
  - 将这些URL放到队列中
- 这里有个“Web的连通性很好”的基本假设

Web采集示意图


/index.html
/courses
/news
www.cs.umass.edu
/index.html
/today
/2005/story.html
www.bbc.co.uk
/index.html
/news.html
/about.html
www.whitehouse.gov
/index.html
/2006/09/story.html
/2003/04/story.html
www.cnn.com
crawlersearchengine.com

课堂思考：下列爬虫有什么问题？

```
urlqueue := (some carefully selected set of seed
urls)
while urlqueue is not empty:
myurl := urlqueue.getlastanddelete()
mypage := myurl.fetch()
fetchedurls.add(myurl)
newurls := mypage.extracturls()
for myurl in newurls:
if myurl not in fetchedurls and not in urlqueue:
urlqueue.add(myurl)
addtoinvertedindex(mypage)
```

采集器面对的问题

- **规模问题：**规模巨大，必须规模可扩展，要分布式处理
- **质量问题：**抓取“有用”网页，必须进行选择、去重等
- **效率问题：**一秒采集数百数千网页，必须充分利用资源
- **鲁棒性问题：**存在采集器陷阱和作弊网页
- **礼貌性问题：**对同一网站的访问要遵照协议规定，并且访问的间隔必须要足够
- **新鲜度(freshness)问题：**保证包含索引网页的较新版本，必须要定期更新或者重采

采集器必须具有的功能特点

鲁棒性

- 能够处理采集器陷阱、重复页面、超大页面、超大网站、动态页面等问题

礼貌性

- 不要高频率采集某个网站
- 仅仅采集robots.txt所规定的可以采集的网页

Robots.txt文件

- 1994年起使用的采集器协议，规定了采集器对网站的访问限制
- 例子：
  - User-agent: *

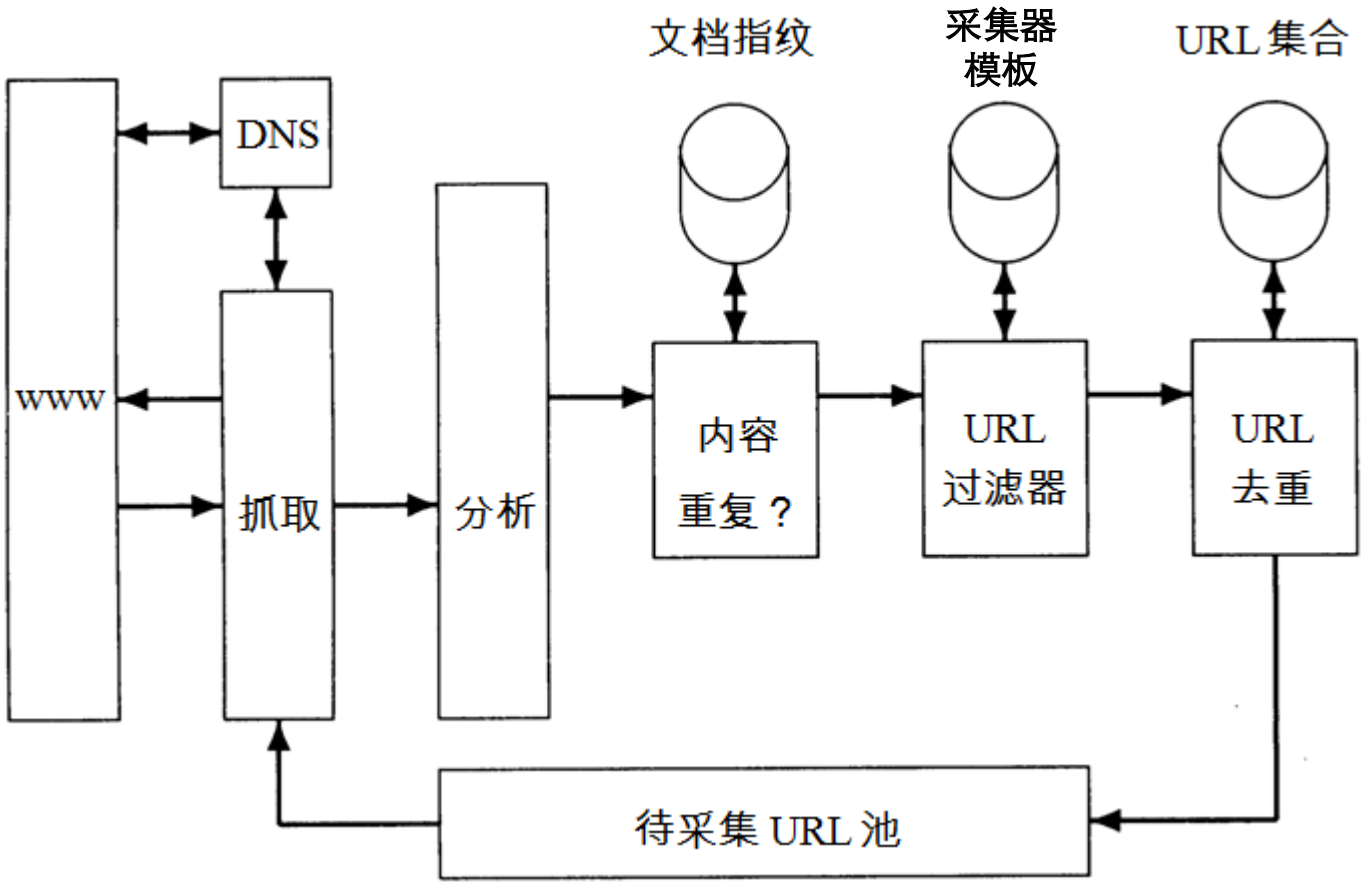
    Disallow: /yoursite/temp/
  - User-agent: searchengine

    Disallow: /
- 采集时，要将每个站点的 robots.txt放到高速缓存中，这一点相当重要

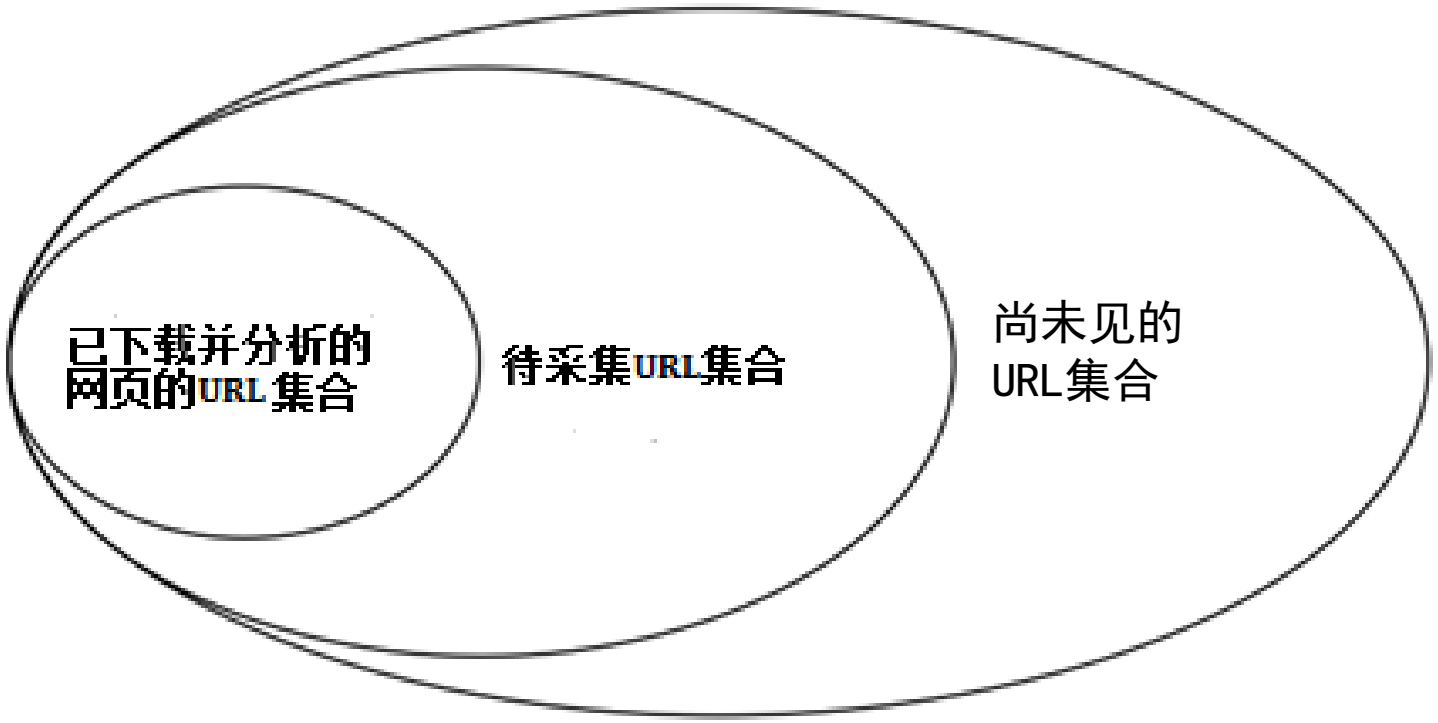
Example of a robots.txt (nih.gov)

```
User-agent: PicoSearch/1.0
Disallow: /news/information/knight/
Disallow: /nidcd/
...
Disallow: /news/research_matters/secure/
Disallow: /od/ocpl/wag/
User-agent: *
Disallow: /news/information/knight/
Disallow: /nidcd/
...
Disallow: /news/research_matters/secure/
Disallow: /od/ocpl/wag/
Disallow: /ddir/
Disallow: /sdminutes/
```

采集器的基本架构



URL集合分类



URL 规范化(normalization)

- 从网页中抽取的URL有些是相对地址
- 比如，在http://mit.edu网站下，会采集到aboutsite.html
- 该页面的绝对地址为： http://mit.edu/aboutsite.html
- 在网页分析过程中，必须要将相对URL地址规范化

抓取与分析

- **抓取：**利用http协议返回某个URL对应的网页，

  抓取到的页面被写入一个临时存储器

- **分析：**从采集到的网页中抽取文本及链接，并将文本、锚文本、链接等信息传给索引器和其它需要这些信息的模块

内容重复判别和URL过滤

□ 内容重复判别

- 目的: 对每个抓取的页面，判断它是否已被索引
- 方法: 采用文档指纹或者shingle的方法判别
- 结果: 忽略那些已经在索引中的重复页面

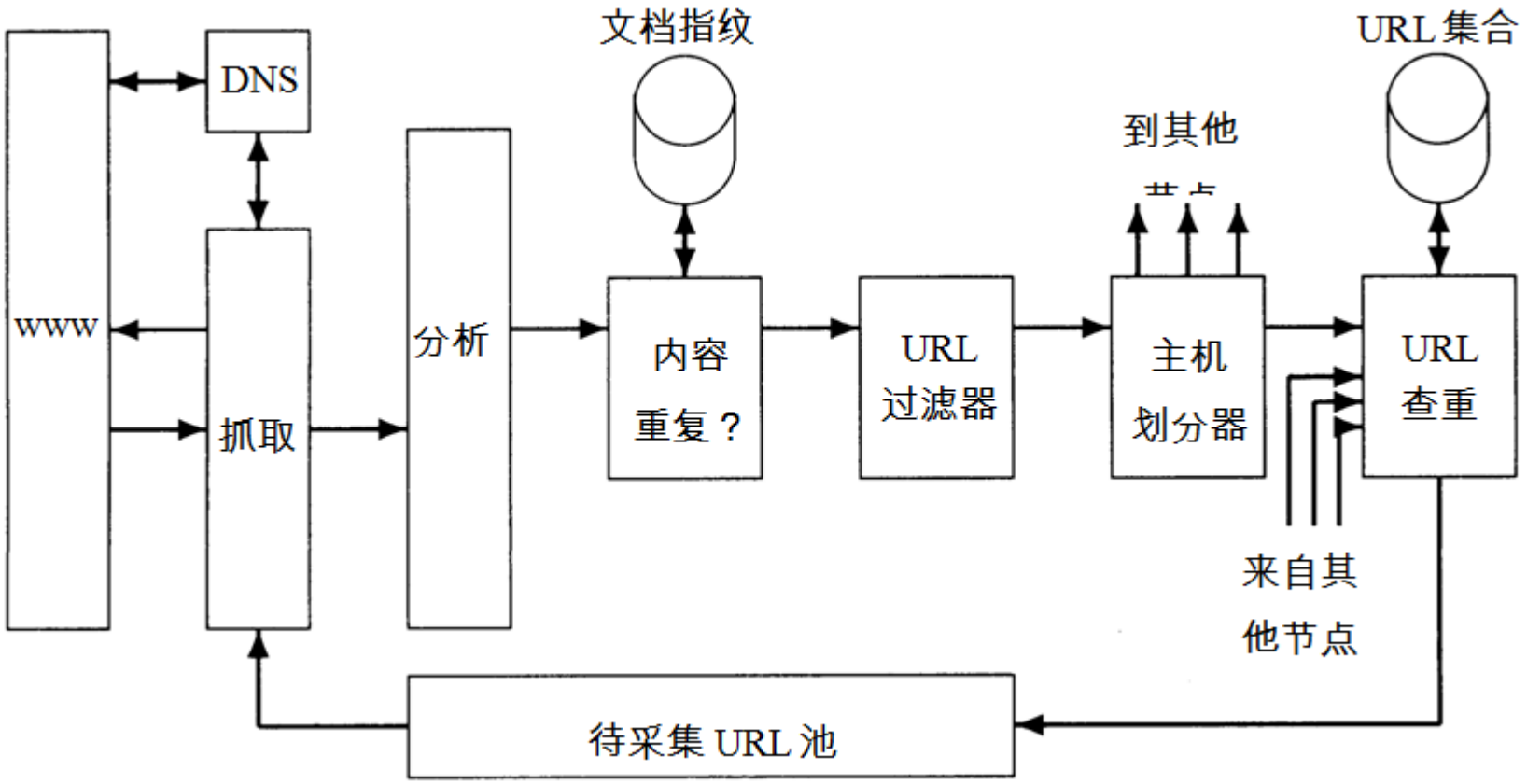
□ URL过滤

- 目的: 确定某个抽取的URL是否应该被URL池收录
- 方法: 排除某些域、基于robots.txt进行排除
- 结果: 忽略那些不必要和不允许采集的URL

分布式采集

- **基本思想：**运行多个采集线程，分布在不同节点上，节点分散在不同地理位置上
- **具体做法：**将需要采集的主机分配到不同节点上，原则是每个节点对“就近”的主机进行采集，比如，地理位置在亚洲的采集器主要关注亚洲域名下网站的采集
- **问题：**前面介绍的采集流程是否还适用？如何实现 每个节点对“就近”的主机进行采集?

分布式采集架构


文档指纹
URL集合
DNS
到其他
www
分析
内容
重复？
URL
过滤器
主机
划分器
URL
查重
抓取
来自其
他节点
待采集URL池

分布式采集中的内容重复检测

- 分布式采集架构中的内容重复检测会很复杂
  - 很难基于主机名划分文档的指纹或shingle，即无法防止相同或者高度相似的内容会出现在不同的Web服务器上
  - 文档的指纹流或shingle流当中几乎不存在局部性
  - 文档会随时间不断变化，因此在连续性采集中，需要从已下载内容集合中去掉过时的指纹或shingle
    - 将文档的指纹或shingle与URL绑定

Google数据中心(wazfaring. com)

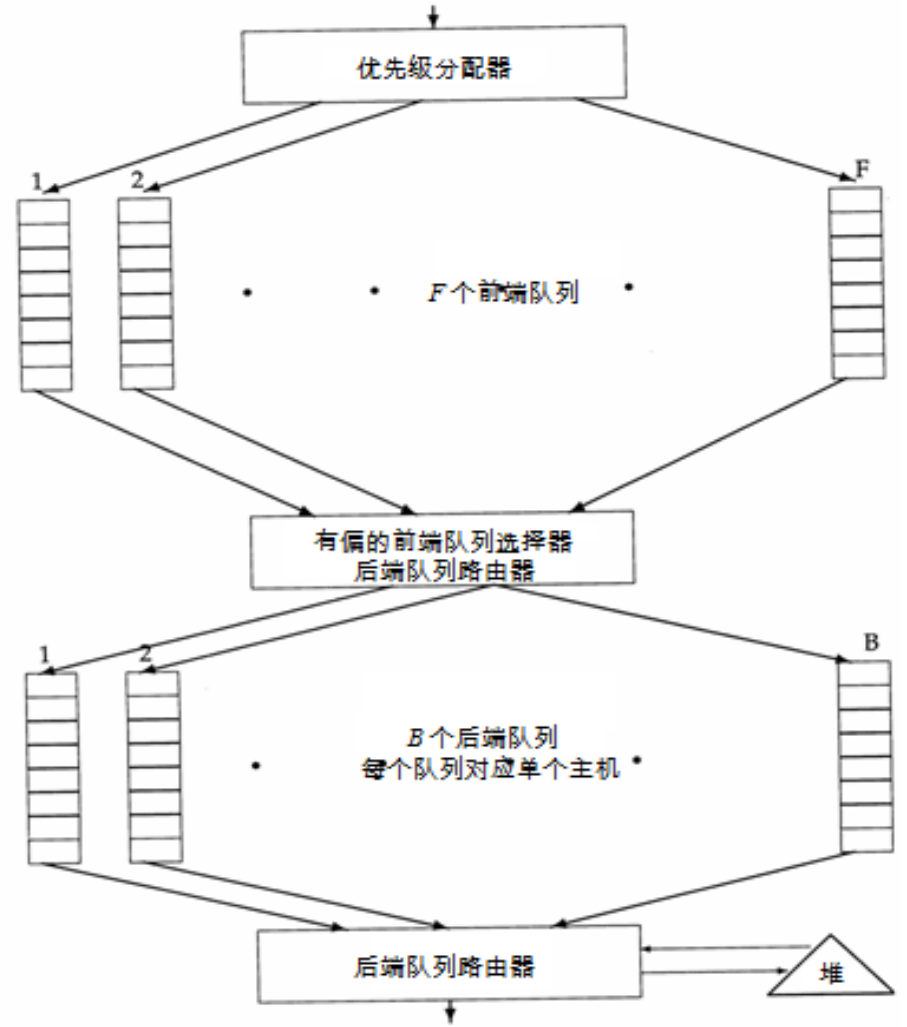
待采集URL池

- 用于存放并管理那些已经发现但是还没有采集的URL集合，其核心是一个数据结构
- 可能包含来自同一主机的不同页面
- 不同页面的质量和更新率往往不同
- 面向多采集线程

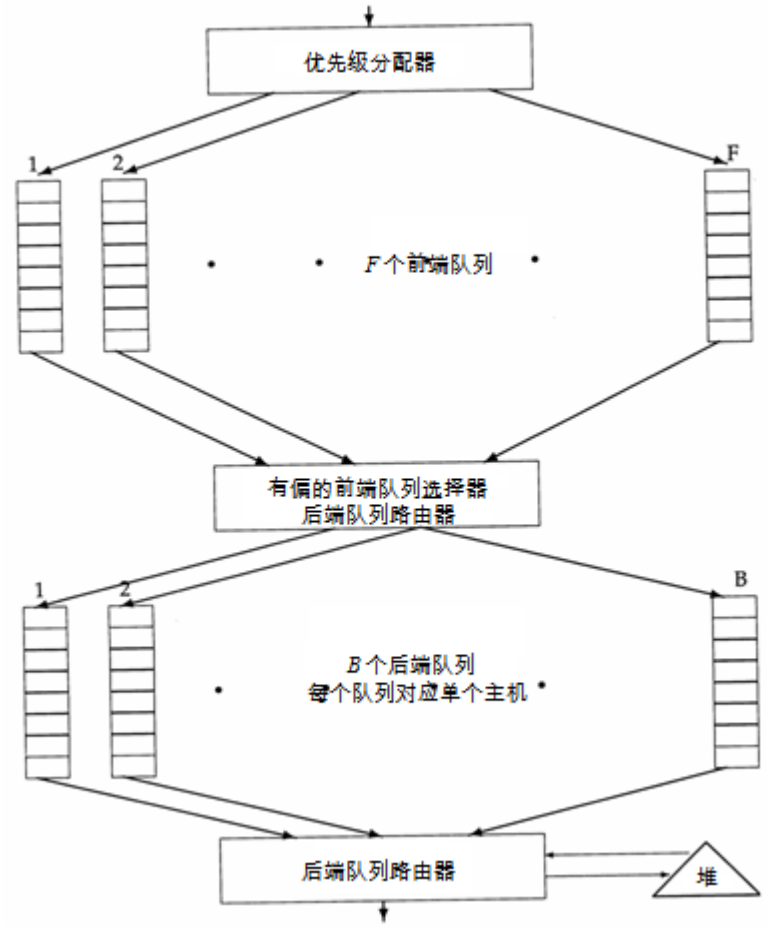
待采集URL的输出策略

- 能否采用一个简单的优先级队列加以实现?
- 确定URL输出次序需要考虑的因素:
  - **礼貌性:** 不要非常频繁的访问某个Web服务器
    - 比如，可以在两次服务器访问之间设置一个时间间隔
  - **优先级:** 对频繁改变的高质量网页优先
  - **饱和度:** 保证所有采集线程任务饱和

Mercator中的待采集URL缓冲池

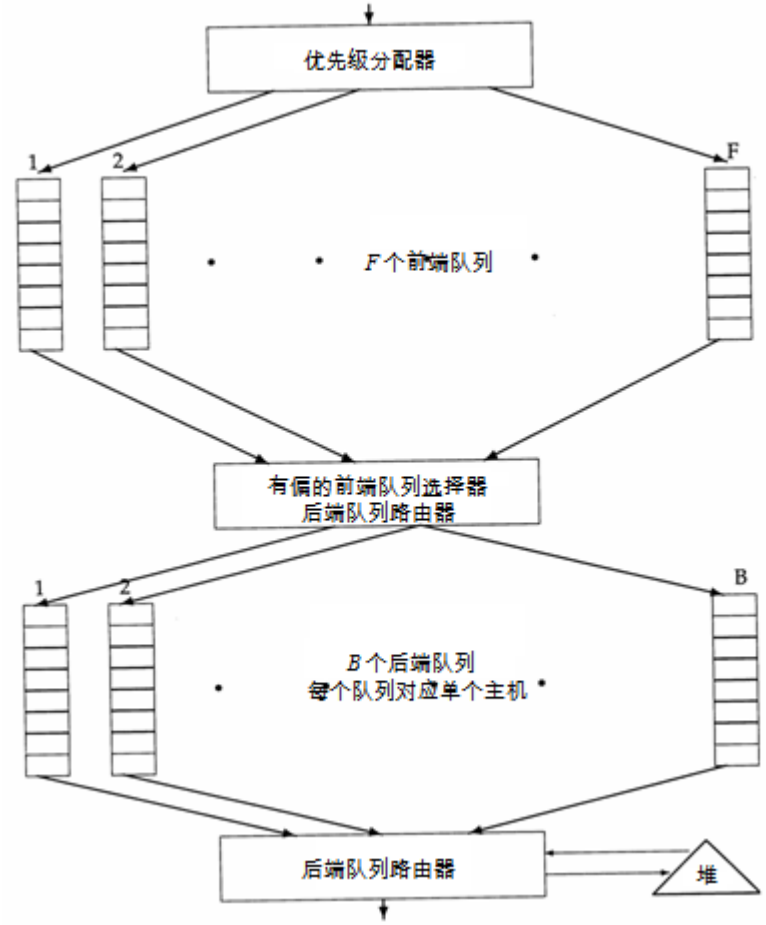


Mercator中的待采集URL缓冲池



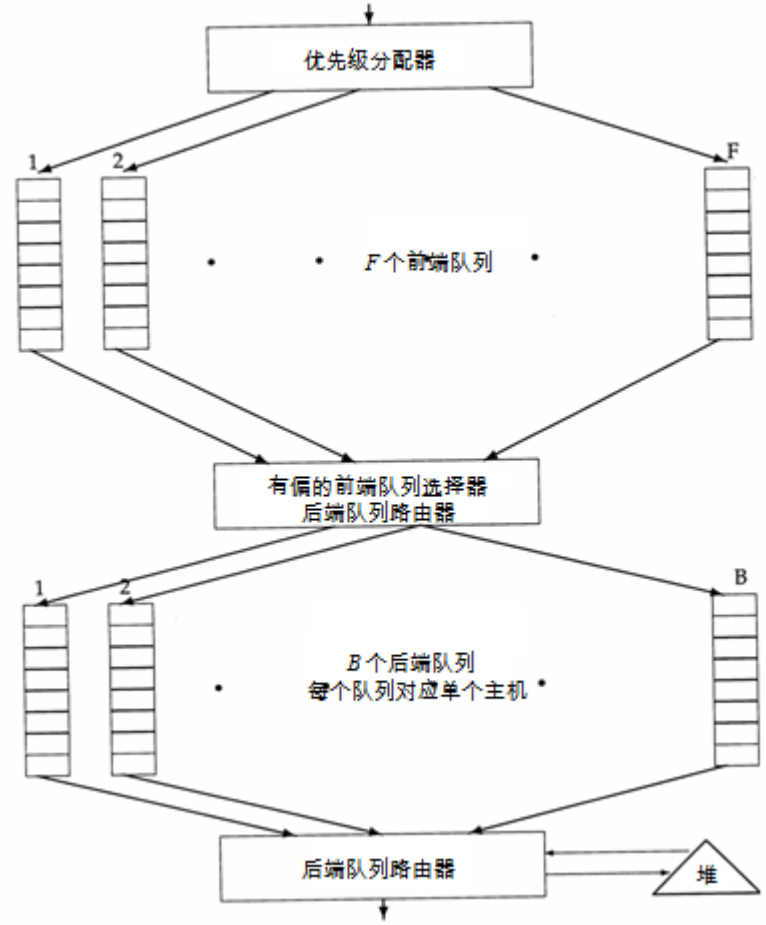
- URL从上部流入缓冲池

Mercator中的待采集URL缓冲池



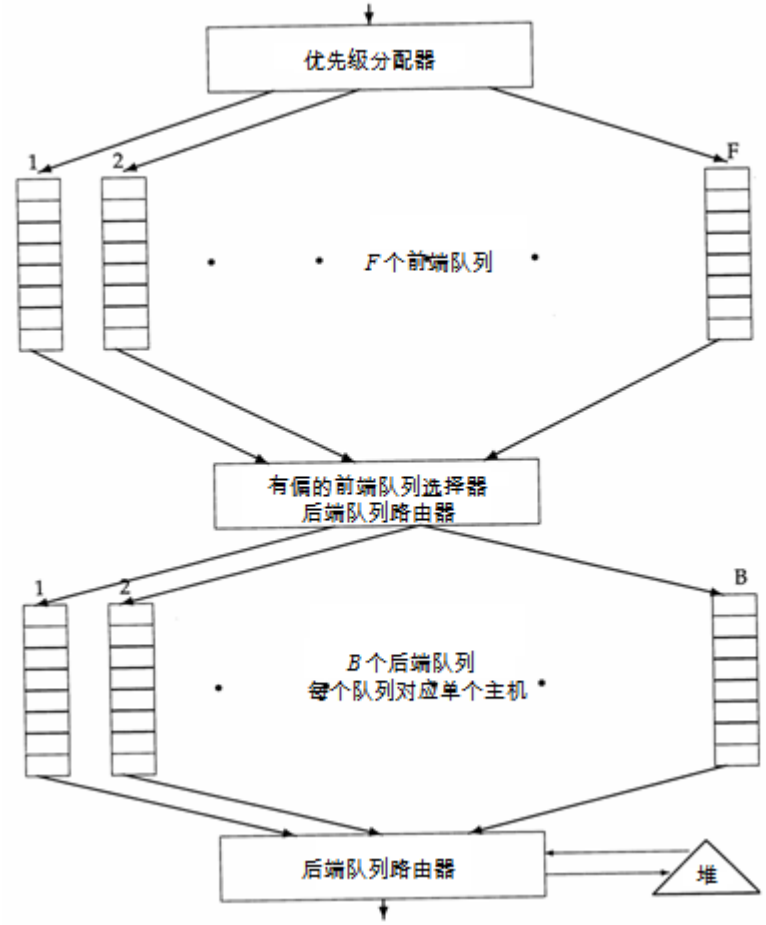
- URL从上部流入缓冲池
- 前端队列(Front queue)管理优先级

Mercator中的待采集URL缓冲池



- URL从上部流入缓冲池
- 前端队列(Front queue)管理优先级
- 后端队列(Back queue) 保证礼貌性

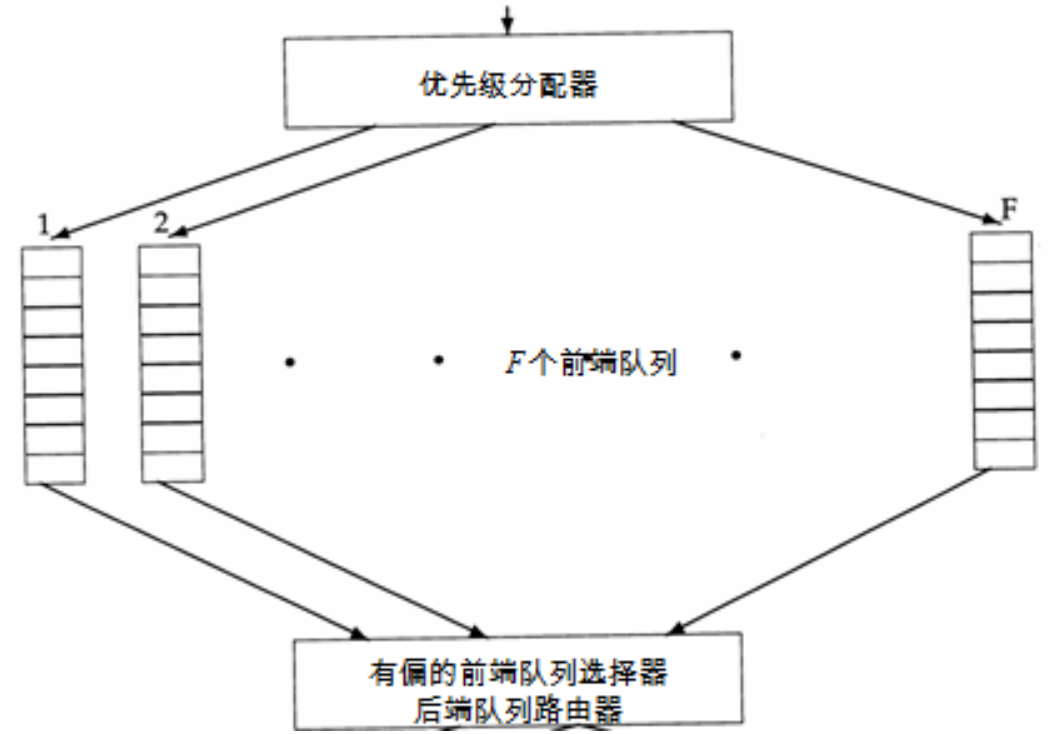
Mercator中的待采集URL缓冲池



- URL从上部流入缓冲池
- 前端队列(Front queue)管理优先级
- 后端队列(Back queue)实现礼貌性
- 每个队列都是FIFO

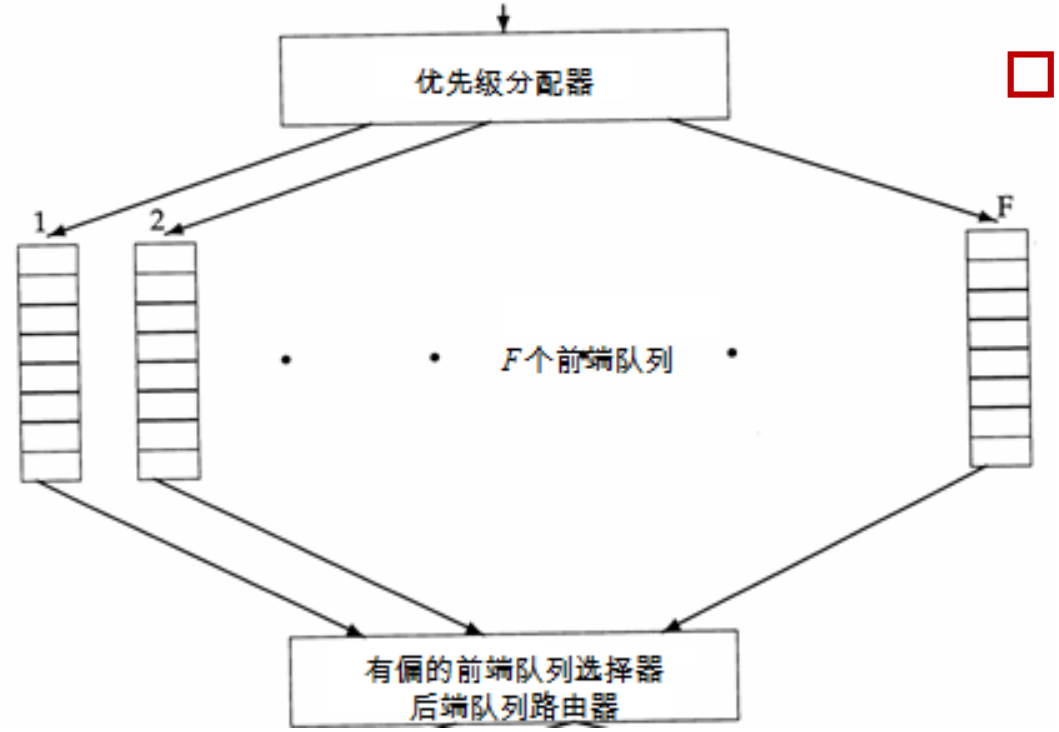
Mercator中的待采集URL缓冲池

- 前端队列(Front queue)



Mercator中的待采集URL缓冲池

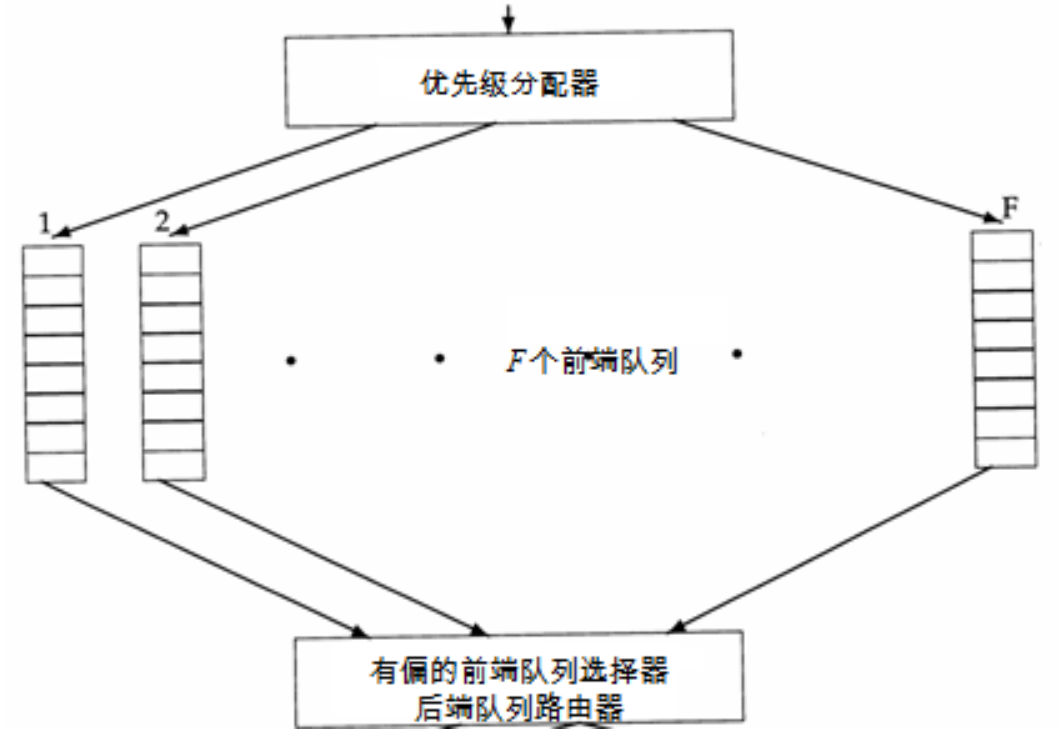
- 前端队列(Front queue)



- 优先级分配器给每个URL分配一个0到*F*之间的优先级整数

Mercator中的待采集URL缓冲池

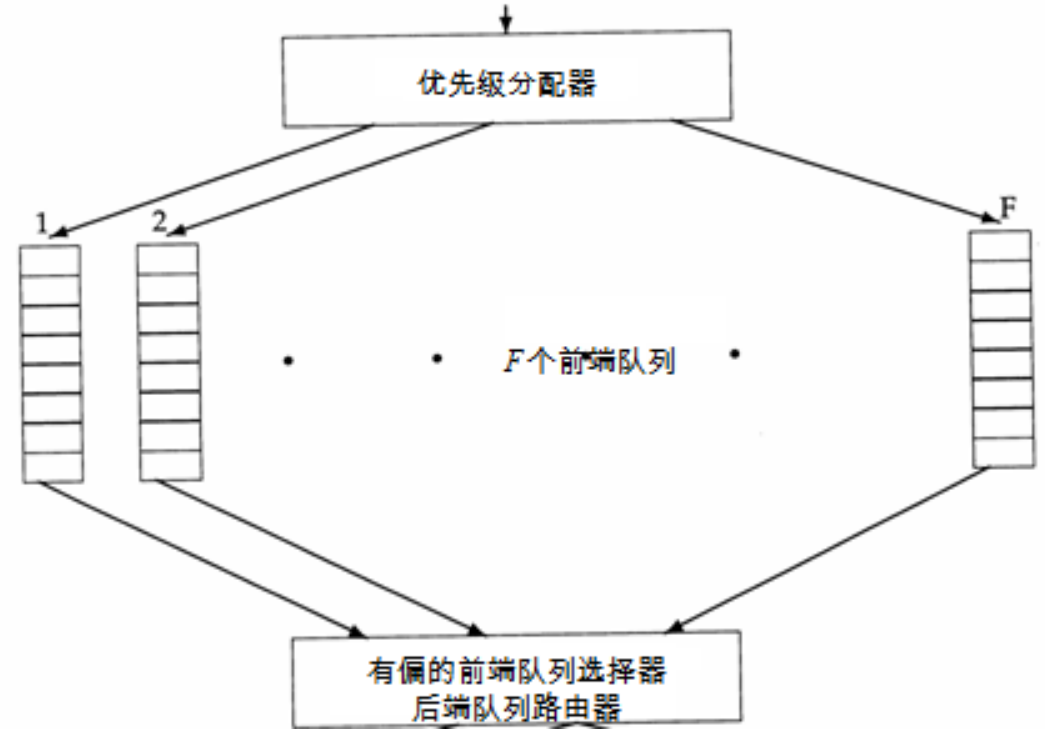
- 前端队列(Front queue)



- 优先级分配器给每个URL分配一个0到*F*之间的优先级整数
  - 分配优先级可以基于启发式信息：更新率、PageRank等等

Mercator中的待采集URL缓冲池

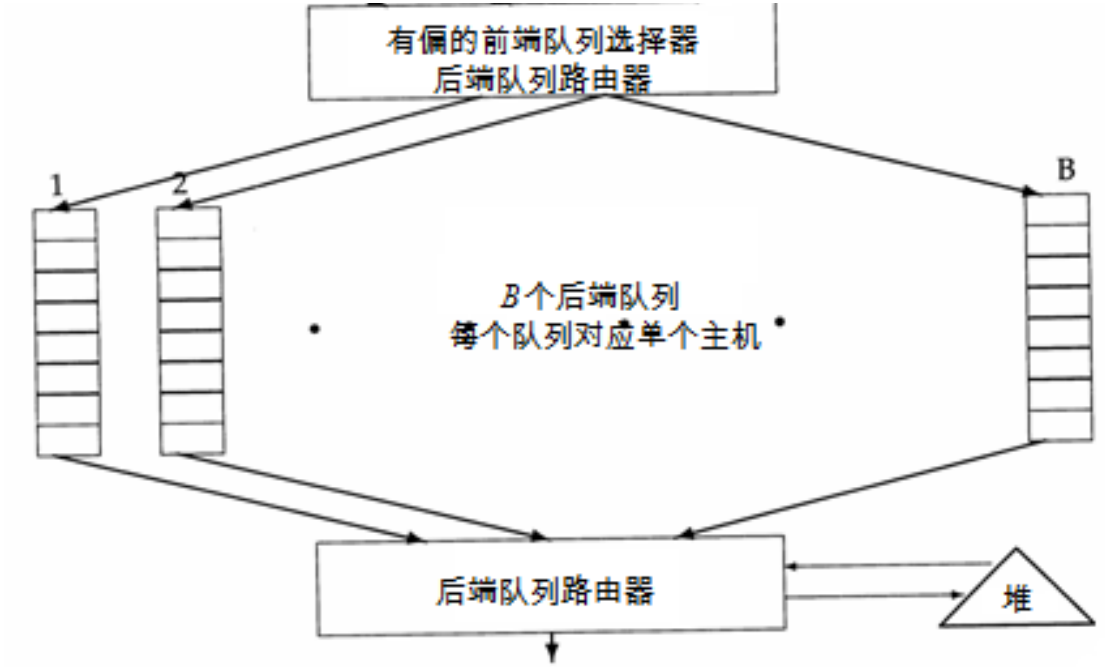
- 前端队列(Front queue)



- 优先级分配器给每个URL分配一个0到F之间的优先级整数
- 分配优先级可以基于启发式信息：更新率、PageRank等等
- 然后将URL添加到相应的队列中

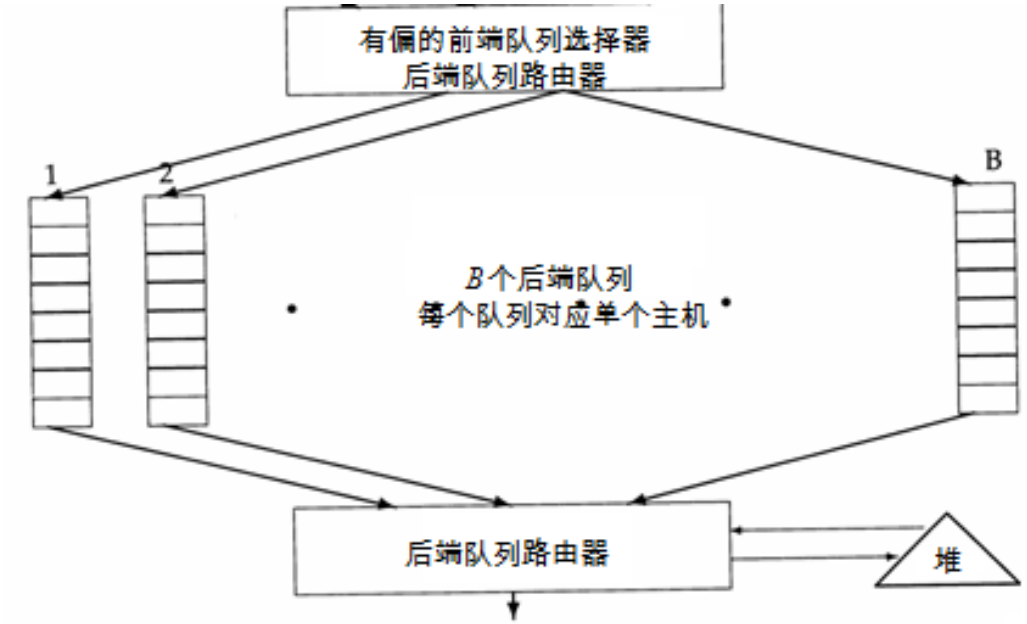
Mercator中的待采集URL缓冲池

- 后端队列(Back queue)



Mercator中的待采集URL缓冲池

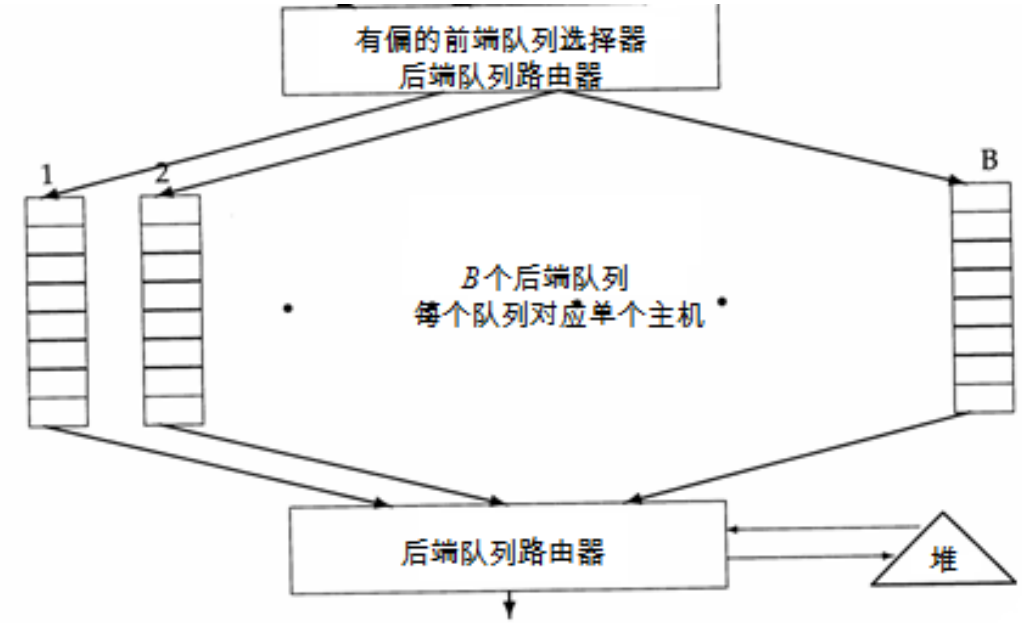
- 后端队列(Back queue)



- **恒定情形1：** 当采集器在运行时，每个后端队列不为空
- **恒定情形2：** 每个后端队列中仅存放来自同一主机的URL
- 维护一张主机到后端队列的表

Mercator中的待采集URL缓冲池

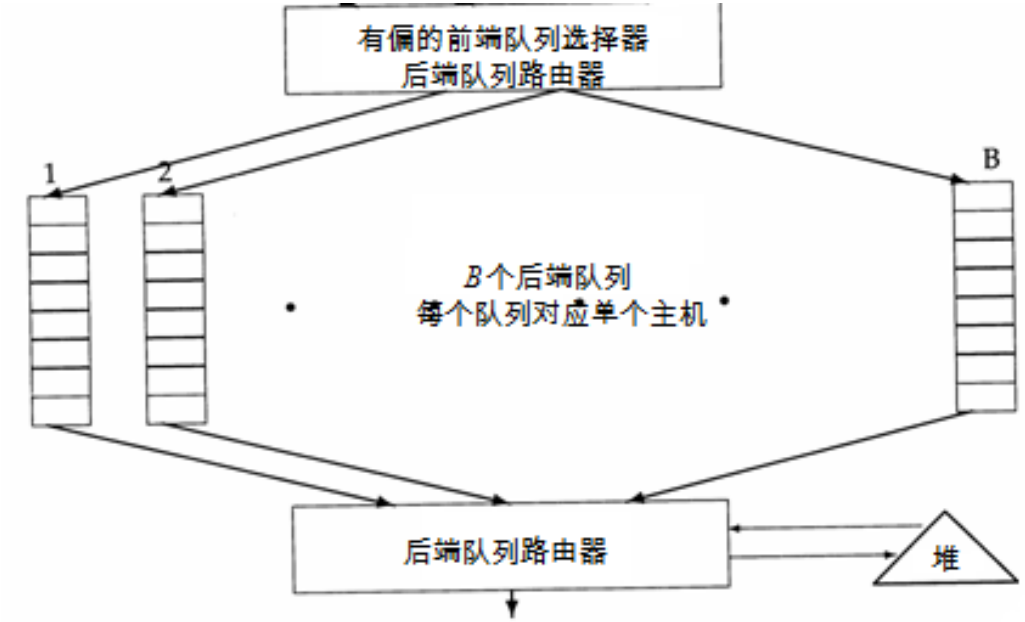
- 后端队列(Back queue)



- 维护一个堆结构，在堆中：
  - 每个后端队列对应一个元素
  - 元素值为该队列对应的主机重新访问的最早时间 t_e
  - 该时间t_e由下列因素确定 (i) 上次访问该主机的时间 (ii) 时间间隔的启发式方法

Mercator中的待采集URL缓冲池

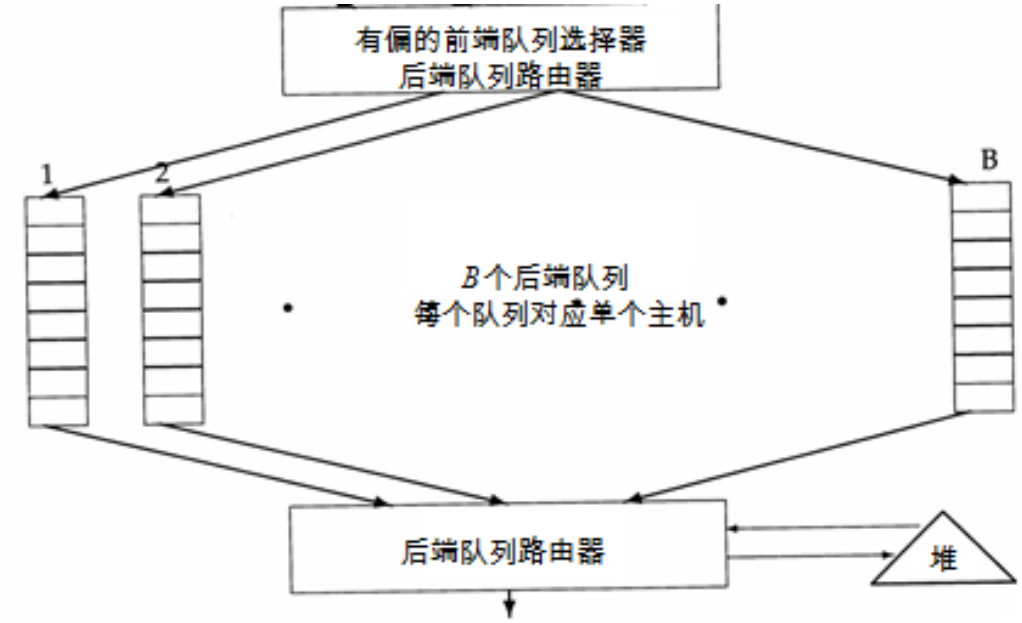
- 后端队列(Back queue)



- 采集线程与后端队列的交互
  - (i) 抽取堆中的当前根节点 q
  - (ii) 抓取q所对应的后端队列中的头部URL u . . .
  - 判断q是否为空，并对为空的情况进行处理

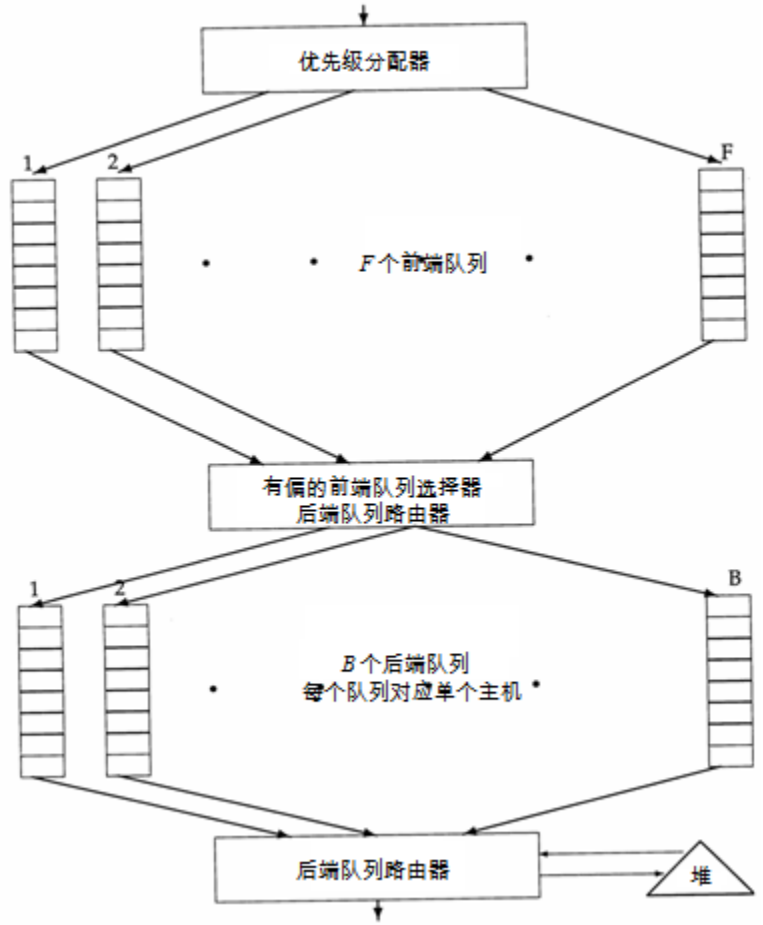
Mercator中的待采集URL缓冲池

- 后端队列(Back queue)



- q对应的后端队列为空后的处理
  - 重复下列操作：(i) 从前端队列中将一系列URL u推入；(ii) 将 u 加到相应的后端队列中. . .
  - . . . 直到得到一个u，u的主机没有对应的后端队列为止
  - 然后将 u 放入 q对应的后端队列并为它在堆中建立一个新元素

Mercator中的待采集URL缓冲池



- URLs从上流入到缓冲池
- 前端队列管理优先级
- 后端队列及其堆结构保证礼貌性和采集线程的忙碌度

分布式索引

基于词项分割的分布式索引

- 将词项词典划分成多个子集，每个子集及其倒排记录表放在一个节点上
- 由于包含不同查询词项的查询流往往与不同的机器集合相关，因此允许更高的并发度

存在问题

- 多词查询处理的开销非常大，甚至可能超过高并发度带来的好处
- 实现划分结果的负载平衡很困难，实现动态索引困难

分布式索引

基于文档分割的分布式索引

- 每个节点包含一个文档子集的索引
- 查询时，将查询广播到所有节点，每个节点返回排名最高的k个结果，然后合并产生最终k个结果
- 更为普遍使用

文档分割方法

- 将来自同一个主机的网页全部分到同一个节点，可能造成访问的均衡性不好
- 将每个URL哈希到索引节点空间

连接服务器(connectivity server)

- **目的：** 支持Web图连接查询的快速处理
  - 给定的URL指向了哪些URL？给定的URL被哪些URL所指向？
- **核心任务：** 在内存中有效存储Web图
  - 对每个网页采用整数编号
  - 建立两个邻接表，分别记录每个网页指向了哪些URL，被哪些URL所指向

```
1: 1, 2, 4, 8, 16, 32, 64
2: 1, 4, 9, 16, 25, 36, 49, 64
3: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
4: 1, 4, 8, 16, 25, 36, 49, 64
```

减少所需存储空间的方法

- **基本思想：** 利用邻接表中许多行存在其相似行的特点，对邻接表进行压缩表示
- **具体方法：** 从上到下遍历邻接表，对每一行基于前面的7行进行编码：指定行偏移量，给出删除的整数和增加的整数
  - 一个例子：可以将下面的第4行进行如下编码：2,9,8

```
1: 1, 2, 4, 8, 16, 32, 64
2: 1, 4, 9, 16, 25, 36, 49, 64
3: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
4: 1, 4, 8, 16, 25, 36, 49, 64
```

减少所需存储空间的方法

- 只使用前7行进行编码的优点
  - 偏移可以通过3个bit来表示，这种选择在经验上最优
  - 将最大的偏移固定在一个较小的值，能够减少搜索相似行的开销
- 无相似行的处理
  - 将本行表示成从一个空集开始并不断加入本行所有整数的过程
  - 采用间隔码进一步减少存储空间

参考资料

- 《信息检索导论》第20章
- http://ifnlp.org/ir
  - Heydon等人撰写的有关Mercator的论文
  - 采集协议标准

课后作业

- 见课程网页:

http://10.76.3.31