《IRLbot: Scaling to 6 Billion Pages and Beyond》阅读报告

（一） 问题描述

1. 爬虫的主要问题：

一个理想的爬虫，能够从一个给定的种子 URL 集合开始，将所有能够被发现到的链接地址对应的 HTML 内容下载下来，并在这个过程中动态地改变下载顺序，最终可以下载到所有有用的网页；同时，在下载的过程中，无论规模多大，都应当保持一定的速度。事实上，爬虫在实际工作中还有其他的限制与要求——爬虫需要限制对于单一网站、单一服务器的访问频率，也需要避免被一些虚假的垃圾网站（spam）所困住，导致爬虫的效率降低。

具体而言，可以分为下列三类问题。

2. 规模问题：

每个爬虫系统都必须面对一个固有的取舍：在规模、效率和硬件资源使用三者中做出权衡。一般来说，较大的规模将导致较低的效率与较高的资源使用，较高的效率需要降低规模、增加资源使用。因而，大多数爬虫只能兼顾三者之二，如大而慢的爬虫、小而快的爬虫，大而快却需要占用大量资源的爬虫。论文中希望给出一种能够兼顾三者的方案。

3. 网站名声与垃圾网站问题：

网页脚本带来了具有高复杂性的动态网站，大量的垃圾网站日益猖獗，二者性质不同，却都给爬虫带来了一个新的挑战：必须要有一种在爬虫爬取网页的过程中实时决定爬取优先级的方法。因为，传统的广度优先搜索往往

会由于 URL 过多，DNS 解析不及，网页延迟等等而陷入到这样的网站中，极大影响爬虫的效率。

4. 礼貌问题：

网络爬虫对某一服务器的频繁访问，往往会对服务器的正常性能造成影响，因而也容易招致服务器的拒绝访问或是举报、诉讼，因而需要对爬虫设置一定速度的限制。直接给爬虫设置这种对单一服务器、单一 IP 地址的访问速度限制并不复杂，却容易导致爬虫的效率在特定情况下（待爬取的 URL 只来自于极少的几个服务器或 IP，由于限制不得不减慢速度）极大地降低效率。因而需要设计一种可以避免这种情况的发生的爬虫。

（二） 问题的解决与性能分析

1. 规模问题：

a) 动机：规模问题最终体现在确认 URL 的“唯一性”和机器人规则上，主要牵涉到硬盘与存储器的交互。在先前的方法中，无论是 Mrcator-B 还是 Ploybot，随着爬取规模的增大，执行这一步骤的开销都会快速增长。为了降低这一开销，需要一种更有效的数据存储结构。

b) 解决方法：论文中提出了 DRUM（Disk Repository with Update Management）的技术。这个技术结合了桶排序和哈希算法，将刚读入的数据元组<key,value,aux>分割为<key,value>与 aux 两个部分，分入内存中的各桶，并在一次操作中将所有的桶中的数据与硬盘中的数据进行合并。通过这样的设计，可以实现对大规模键值对数据的存储，并实现快速的检查（check）、更新（update）、检查 + 更新（check+update）的操作。

下图给出了 DRUM 的详细操作:

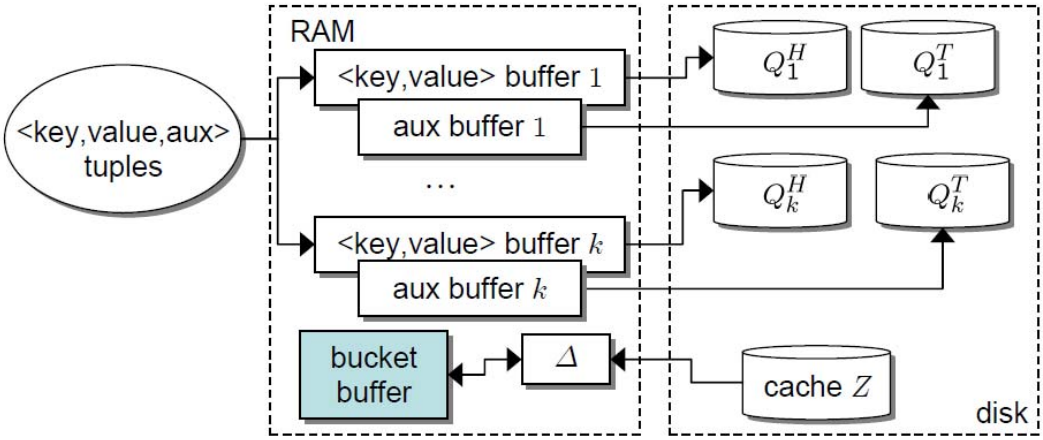


Figure 1: Operation of DRUM.

利用这一技术, 在论文中的爬虫系统创建了多个存储模块, 包括 URLseen 模块、RobotsCache 模块、RobotsRequested 模块、PLDindegree 模块, 分别赋予元组一定的意义，赋予各模块特定的操作，以此大大提高系统规模化的效率。

c) 性能分析：论文中主要通过给定一系列参数来推导 URLseen 的开销, 从而比较各种数据结构的优劣。这里摘录如下:

Variable	Meaning	Units
N	Crawl scope	pages
p	Probability of URL uniqueness	–
U	Initial size of URLseen file	pages
R	RAM size	bytes
l	Average number of links per page	–
n	Links requiring URL check	–
q	Compression ratio of URLs	–
b	Average size of URLs	bytes
H	URL hash size	bytes
P	Memory pointer size	bytes

符号定义

Mrcator-B:

$$\omega(n, R) = \frac{2(H+P)pH}{R} n^2$$

Ploybot：

$$\omega(n, R) = \frac{2(b+4P)pbq}{R} n^2.$$

DRUM：

$$\omega(n, R) = nb\left(\frac{(H+b)(2UH+pHn)}{bD} + 2 + p + \frac{2H}{b}\right)$$

并做比较如下表：

N	Mercator-B	Polybot	DRUM
800M	11.6	69	2.26
8B	93	663	2.35
80B	917	6,610	3.3
800B	9,156	66,082	12.5
8T	91,541	660,802	104

Table 2: Overhead $\alpha(n, R)$ for $R = 1$ GB and $D = 4.39$ TB.

可以看出，DRUM 的开销显著小于先前的技术。

2. 网站名声与垃圾网站问题

a) 动机：拥有大量动态网页的合法网站与制造大量垃圾网页的恶意网站，都使得爬虫在礼貌性原则、DNS 访问以及爬取本身的限制下变得低效，也会带来带宽的巨大浪费。然而，由于互联网规模不断扩大，拥有同样有用大量网页的合法网站与恶意网站相互混杂，使得简单的设置阈值并不能合理的解决这个问题。而在之前的研究中，无论是 BFS 的爬取策略，还是 PageRank、BlockRank、SiteRank 算法，也极易使爬虫陷入到这种不断生成动态网页的网站中。只有将爬虫访问某一网页的频率与该网页

的名声（reputation）结合起来，用少量资源即时判断网页类型、决定对某一域名的网站搜索的深度，才能提高爬虫效率。

b) 解决方法：论文指出，只要在“网站名声”的基础上给每个 PLD（Pay-Level Domain）分配预算，即可侦测出垃圾网站。论文在 PLD 这个粒度上进行“预算”的计算，流程如下图：

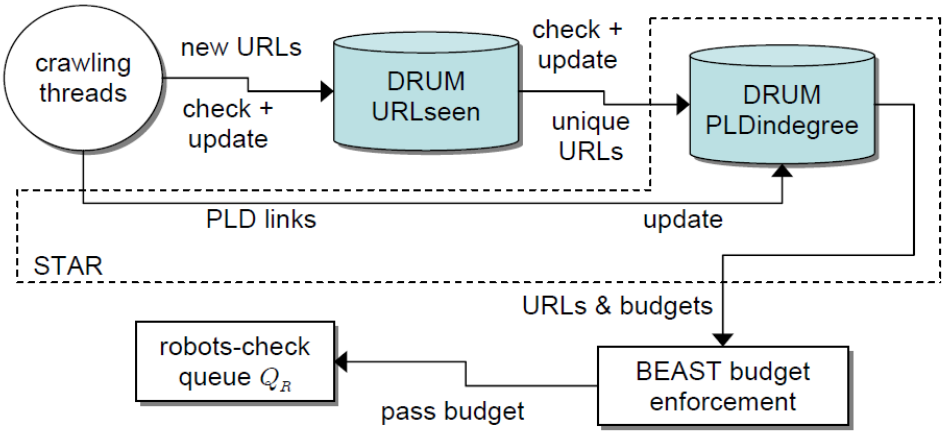


Figure 3: Operation of STAR.

为了得到网站的“预算”，论文给出了 STAR（Spam Tracking and Avoidance through Repitation）算法：利用 DRUM 的存储结构，存储爬虫在爬去过程中得到的 PLD 网络图的信息，构造 PLDindegree 模块。通过模块中考察域名的链入数，为各个域名动态地分配“预算”，并按照“预算”指示单位时间内爬虫能够从该域名爬取多少新的链接，最终避免垃圾网站的困扰。

c) 性能分析：从理论上说，从其他 PLD 获得链入需要付出（金钱）代价，一般的垃圾网站在代价面前很可能不会获得高的“预算”，故使用这一方法来鉴别网站的质量。

3. 礼貌问题：

a) 动机：短时间对某一服务器的频繁访问将给服务器带来很大压力，甚至形成 DOS 攻击。而在之前的研究中，简单地设置单个主机的访问延迟，

可能会导致“多主机共用”的服务器崩溃，若设置单个服务器的访问延迟，又将大大降低效率，甚至可能在大规模的网页中最终无法正常工作。

另一方面，在已经得到各网站“预算”的情况下，若仅仅只是重复地扫描未爬取的链接队列并从中选取需要爬取的链接，只能在高昂的花费下得到极少的有用链接。因而需要想办法更有效地利用“预算”的结果给出爬取各网页的延迟，才能实现高效的爬虫。

b) 解决办法：论文中给出了 BEATS（Budget Enforcement with Anti-Spam Tactics）的解决办法，其基本思想是，不丢弃任何一个链接，而是在确定它的“合法性”之前，延迟下载这个链接的时机。

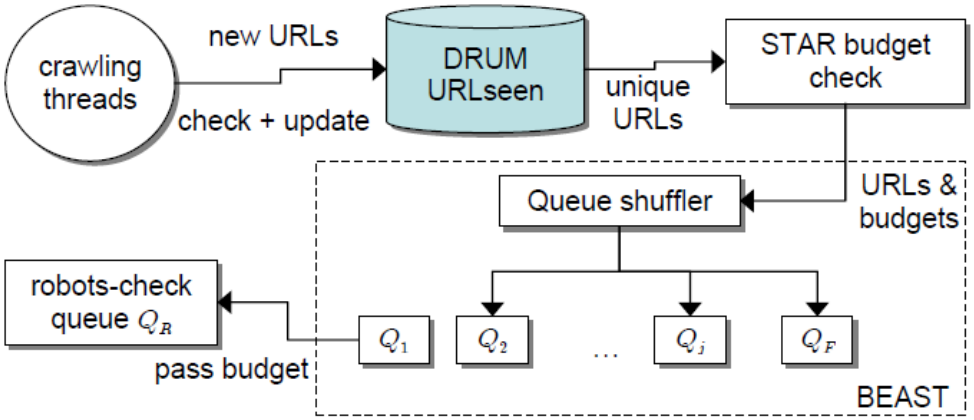


Figure 4: Operation of BEAST.

如图所示，在经过修正之后，论文给出了一种不需要依赖数据规模增大硬盘读写能力的实现方式：将通过 STAR 赋予了一定预算的成批链接进行检查，将通过检查的、有较高预算的链接按照预算排名高低分到j个队列中，将暂时未通过检查的、只有排名的链接分到一个单独的队列Q_F中；当前j个队列中的链接全部爬取完成后，重新检查队列Q_F，并将其中通过检查的链接分到已有队列的两倍数量的队列中。不断重复上述过程，不

断动态地增加队列的数量，直至达到某些停止条件。用这种办法可以合理地决定网页的爬取顺序。

c) 性能分析：采取 BEATS 的办法，一方面保留了队列的不同优先级，使得Q_F中具有较高预算的链接可以尽快地得到爬取；另一方面利用不断增长的j使得预算较低的链接不断地推迟被爬取的时机，实现爬取的延迟。同时，从硬盘读写的角度来看，

所需要速度如下：

$$\lambda = 2Sb\left[\frac{2\alpha_N}{1+\alpha_N}(L-1)+1\right] \leq 2Sb(2L-1).$$

当 N 增大时，速度要求不会超过$2Sb(2L-1)$。

所需队列数量如下：

$$j = 2^{\lceil \log_2(\alpha_N+1)\rceil - 1}$$

容易得出，队列的数量可以接受。

（三） 实验验证：

1. 系统设计：

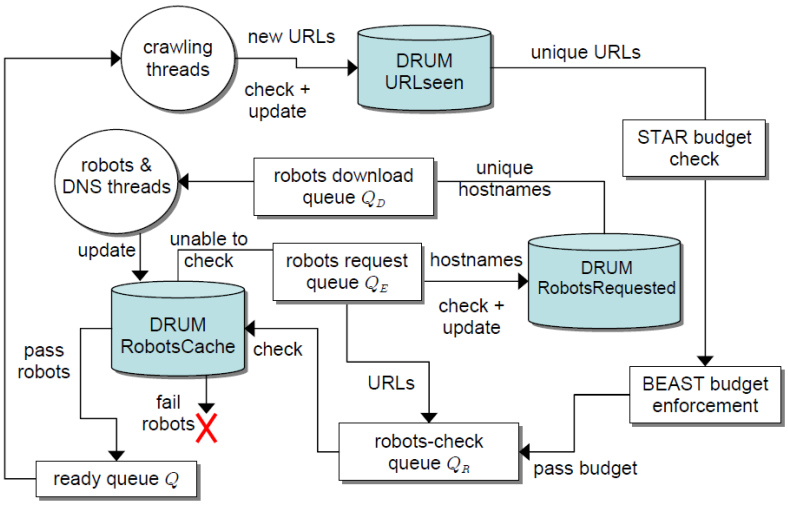


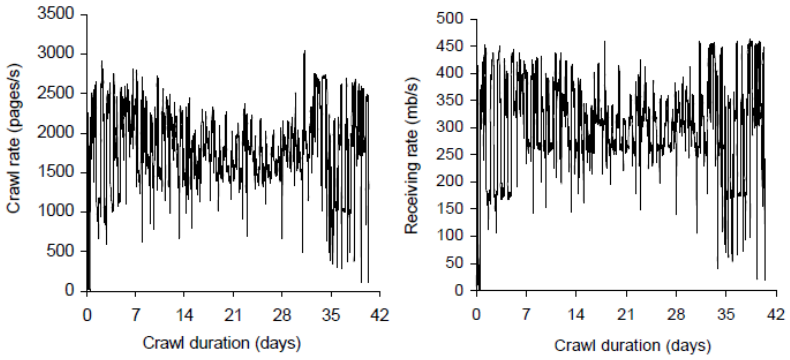
Figure 2: High level organization of IRLbot.

在充分整合 DRUM,STAR,BEATS 技术之后，论文搭建形成了如图所示的爬虫系统 IRLbot，并从 crawling threads 得到新链接开始，涉及到 URL 唯一性确认，“预算”的确认，机器人规则的确认，“预算”的检查以及最终页面的下载，形成了一个完整的处理流程。

2. 实验验证

a) 爬取结果与效率验证

论文中，在 2007 年 6 月 9 日至 8 月 3 日的这段时间，IRLbot 运行在带有 2.6GHz 的 4 核 AMDCPU，16GB 的内存，RAID-5 存储系统的服务器上，并以 1-gb/s 的链接连接互联网。最终在 41.27 天的运行过程中，IRLbot 爬虫尝试了 7,606,109,371 次尝试，收到了 7,437,281,300 次的有效 HTTP 响应，排除了无 HTML 内容、HTTP 错误和重定向内容后，最终收到 6,380,051,942 次响应并爬取这些页面的数据。下载的平均速度是 319 mb/s (每秒 1,789 个页面)。总共收到了 143TB 的数据，分离出 394,619, 023, 142 个链接，最终得到 374, 707, 295, 503 个可爬取的链接，得到了 332GB 的 URLseen 模块，其中保存了 41,502,195,631 个互不相同的链接，分布在 641,982,061 个不同的主机上。容易看出，大量页面仍未爬取。

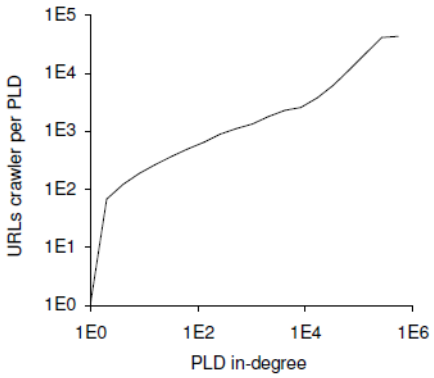


(a) pages/s (b) mb/s

Figure 5: Download rates during the experiment.

b) STAR 算法的验证

论文指出，收到的回复来自 117,576,295 个网页，属于 33,755,361 个 PLD。在 PLD 网络图中，共 89,652,630 个节点，1,832,325,052 条边。通过一种保守的预算赋值的办法，在对较高排名的链接赋予中等预算，而扩大较低排名的链接范围后，可以得到下图所示的效果：



(b) effectiveness of STAR

可以看出，域名的流行度（链入）和该域名在爬虫运行时被分配到的带宽之间有着强烈的关联。

同时，通过分析排名最高的 1000 个网站，发现其中的大部分网站都非常知名，这也说明了 Reputation 计算的有效性。分析结果如下表所示：

Rank	Domain	In-degree	PageRank	Pages
1	microsoft.com	2,948,085	9	37,755
2	google.com	2,224,297	10	18,878
3	yahoo.com	1,998,266	9	70,143
4	adobe.com	1,287,798	10	13,160
5	blogspot.com	1,195,991	9	347,613
7	wikipedia.org	1,032,881	8	76,322
6	w3.org	933,720	10	9,817
8	geocities.com	932,987	8	26,673
9	msn.com	804,494	8	10,802
10	amazon.com	745,763	9	13,157

Table 4: Top ranked PLDs, their PLD in-degree, Google PageRank, and total pages crawled.