

(12) 发明专利申请

(10) 申请公布号 CN 102663058 A

(43) 申请公布日 2012. 09. 12

(21) 申请号 201210090259. 0

(22) 申请日 2012. 03. 30

(71) 申请人 华中科技大学

地址 430074 湖北省武汉市洪山区珞喻路
1037 号

(72) 发明人 邹复好 凌贺飞 李平 刘学
邱荷花

(74) 专利代理机构 华中科技大学专利中心

42201

代理人 朱仁玲

(51) Int. Cl.

G06F 17/30 (2006. 01)

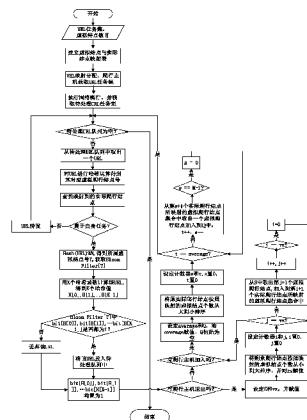
权利要求书 2 页 说明书 7 页 附图 3 页

(54) 发明名称

一种分布式网络爬虫系统中的 URL 去重方法

(57) 摘要

本发明提供了一种分布式网络爬虫系统中的 URL 去重方法，通过引入虚拟爬行结点，实现了高效的任务划分策略，从而更好地适应分布式网络爬虫系统中实际爬行结点的动态变化，在任务划分策略基础上使用一种分布式的 URL 去重方式，从而避免实际爬行结点变化过程中造成的重复爬行。本发明在任务划分时变动规模小，能保证爬虫系统稳定持久运行，划分策略具有动态适应性，能实现实际爬行结点的负载均衡。采用多个布隆过滤器去重结构，减小了去重对内存大小的需求，可实现基于内存的快速去重，在需要时能高效转移和备份，防止由于去重信息缺失而造成爬虫系统重复爬行。本发明效率高，可操作性好，具有极高的应用价值。



1. 一种分布式网络爬虫系统中的 URL 去重方法,其特征在于,包括以下步骤:
 - (1) 当前爬行结点根据初始 URL 任务集进入网络爬行状态,并获取待处理 URL 任务集;
 - (2) 判断待处理 URL 任务集是否为空,若为空则过程结束,否则进入步骤 (3);
 - (3) 从待处理 URL 任务集中获取 URL 任务;
 - (4) 对获取的 URL 任务进行哈希运算得到其对应的虚拟爬行结点号;
 - (5) 根据该虚拟爬行结点号,查找虚拟爬行结点与实际爬行结点的映射关系表,从而得到对应的实际爬行结点;
 - (6) 判断获取的 URL 任务是否属于当前爬行结点,如果是则进入步骤 (9),否则转入步骤 (7);
 - (7) 将获取的 URL 任务转发给实际爬行结点;
 - (8) 对于待处理 URL 任务集中的所有 URL 任务,重复上述步骤 (2) 至 (7),直到所有 URL 任务均处理完毕为止;
 - (9) 对该 URL 任务进行哈希运算,以找到该 URL 任务所属的虚拟爬行结点,并找到对应的布隆过滤器去重结构;
 - (10) 对 URL 任务用 K 个哈希函数计算,以得到 K 个哈希值 $H[0], H[1], H[2], \dots, H[K-1]$;
 - (11) 根据 K 个哈希值查找布隆过滤器去重结构的位数组中对应的第 $H[0], H[1], H[2], \dots, H[K-1]$ 位,以判断第 $H[0], H[1], H[2], \dots, H[K-1]$ 位是否均为 1,若均为 1,则进入步骤 (12),否则进入步骤 (13);
 - (12) 丢弃该 URL 任务,然后转入步骤 (14);
 - (13) 表明该 URL 任务不是处于当前爬行结点的 URL 任务集合中,将该 URL 任务加入当前爬行结点的待处理队列中;
 - (14) 将布隆过滤器去重结构的位数组中对应的第 $H[0], H[1], H[2], \dots, H[K-1]$ 位全部置 1;
 - (15) 判断是否有实际爬行结点退出,若有则转入步骤 (16),否则转入步骤 (25);
 - (16) 设定退出的实际爬行结点所映射的虚拟爬行结点集合为 S, S 中虚拟爬行结点个数记为 vz ;
 - (17) 将剩余的实际爬行结点按照其所映射的虚拟爬行结点数从小到大排序,剩余的实际爬行结点个数记为 rz ;
 - (18) 初始化计数器 i 和计数器 j 均为 0,其中 i 指向排序后的实际爬行结点集合中第 $i+1$ 个实际爬行结点, j 指向虚拟爬行结点集合 S 中第 $j+1$ 个虚拟爬行结点;
 - (19) 判断 j 是否等于 vz ,若是则转入步骤 (25),否则转入步骤 (20);
 - (20) 从虚拟爬行结点集合 S 中取出第 $j+1$ 个虚拟爬行结点,并将其加入到第 $i+1$ 个实际爬行结点所映射的虚拟爬行结点集合中;
 - (21) 设置 $i = i+1, j = j+1$;
 - (22) 判断 i 是否等于 rz ,若是则转入步骤 (23),否则转入步骤 (24);
 - (23) 将 i 值置 0;
 - (24) 重复上述步骤 (19) 至 (23),直到 j 值等于 vz 为止;
 - (25) 判断是否有新的实际爬行结点加入,若有则转入步骤 (26),否则返回步骤 (2);

- (26) 设定 $\text{average} = N/M$, M 为加入新实际爬行结点后实际爬行结点的总数, 设定新实际爬行结点所映射的虚拟爬行结点集合为 Q , 其初始为空 ;
- (27) 将原有实际爬行结点按照映射的虚拟爬行结点数从大到小排序 ;
- (28) 初始化计数器 s 和 t 均为 0, 其中 s 指向排序后的实际爬行结点集合中第 $s+1$ 个实际爬行结点, t 为虚拟爬行结点集合 Q 中虚拟爬行结点个数 ;
- (29) 判断 t 是否等于 average , 若是则返回步骤 (2), 否则转入步骤 (30) ;
- (30) 从第 $s+1$ 个实际爬行结点所映射的虚拟爬行结点集合中取出一个虚拟爬行结点加入到 Q 中, 并设置 $t = t+1$, $s = s+1$;
- (31) 判断 s 值是否等于 $M-1$, 若是则转入步骤 (32), 否则转入步骤 (33) ;
- (32) 将 s 值置 0 ;
- (33) 重复执行上述步骤 (29) 至 (32), 直到 t 等于 average 为止。

2. 根据权利要求 1 所述的方法, 其特征在于, 还包括 : 在所述当前爬行结点根据初始 URL 任务集进入网络爬行状态, 并获取待处理 URL 任务集的步骤之前, 设定系统的虚拟爬行结点数 N 及初始 URL 任务集, 每一个虚拟爬行结点对应一个布隆过滤器去重结构, 布隆过滤器去重结构在内存中申请有一个位数组, 位数组大小为 n , 将虚拟爬行结点平均分配给实际爬行结点, 建立虚拟爬行结点与实际爬行结点的映射关系, 采用基于模的哈希运算方式将初始 URL 任务集映射到各个虚拟爬行结点, 并根据虚拟爬行结点与实际爬行结点的映射关系将初始 URL 任务集映射到对应的实际爬行结点。

一种分布式网络爬虫系统中的 URL 去重方法

技术领域

[0001] 本发明属于网络应用技术领域，具体涉及一种分布式网络爬虫系统中的 URL 去重方法。

背景技术

[0002] 随着互联网信息爆炸式增长，用户感兴趣的信息淹没于大量无关信息中，利用搜索引擎获取感兴趣的信息已经成为人们获取信息较为便捷的方式。作为搜索引擎的基础构件之一的网络爬虫，需要直接面向互联网，不间断地从互联网上搜集信息，为搜索引擎提供数据来源。搜索结果是否丰富、获得的信息是否包括新近更新的内容，均与网络爬虫的效率紧密相关。然而互联网规模十分庞大，根据第 27 次中国互联网报告，2010 年中国网站数目已经达到 191 万，网页数量超过 600 亿。如此海量的数据对网络爬虫的设计与实现提出了更高的要求，构建分布式网络爬虫系统是一个有效的解决方案。相应地，作为网络爬虫核心技术的 URL 去重方法对爬虫系统的性能影响尤为重要。

[0003] 网络爬虫是一个机器人程序，它从指定 URL 地址开始下载页面文档，提取其中的 URL 地址，再从提取的 URL 地址开始继续爬行。由于新提取到的 URL 地址可能已在前面处理过，继续下载这些 URL 地址只会造成重复下载，浪费计算资源，因此对提取到的每个 URL 地址必须判断是否已经 处理过，去除已下载的重复 URL 地址。在分布式网络爬虫系统中，需要采用某种策略将 URL 任务平均分配到多个主机并行爬行，划分策略必须高效且易于实现。而在分布式环境下，某个主机提取到的 URL 地址可能已被系统其它主机处理过，因此系统要有一种分布式的 URL 去重机制。

[0004] 目前已有的 URL 去重方法主要有基于内存的去重和基于数据库的去重。在基于内存的 URL 去重方式中，爬虫将系统 URL 全部放在内存中，并使用一个易于查找的数据结构（如哈希表）进行维护，每当遇到新的 URL 时，立即查找内存 URL 集中是否存在该 URL 地址。由于内存有限，而 URL 数量庞大，内存不可能容纳所有的 URL，因此一个改进的去重方法是将一部分 URL 放在内存中，而将大部分 URL 存放在外存中，通过某种调度算法（如最近最少使用 LRU）实现类似缓存的效果，以应对内存不足问题。基于数据库的去重则是在数据库中维护 URL 表结构，将解析到的 URL 插入数据库表中，依靠数据库的唯一性约束条件对 URL 进行是否重复的判断。

[0005] 直接使用内存的去重方式在小规模爬虫中使用较多，由于内存空间有限，因而该方法很难扩展到大规模的 URL 去重任务。大规模爬虫通常采用内存与外存结合的去重方式，但通常 URL 查找的内存命中率存在波动（当爬行进入一个新的域名网站时，命中率迅速下降），调度算法的实现同样加大了系统的复杂性，而且在分布式环境下，某个实际爬行结点可能因故障等原因退出爬行，新的主机也可随时加入系统爬行，这时系统必须重新进行任务划分，因此用于 URL 去重的数据结构也需进行转移，以保证某个主机爬行过的 URL 任务不会被其它主机重复爬行，这种内存外存调度的数据结构不适合转移，或者是转移开销过大。而在基于数据库的判重方式下，所有 URL 任务被发往数据库，导致数据压力过大，当

数据存放的 URL 任务过多时,数据库的性能急剧下降,并且去重依赖于数据库,这对数据库的稳定性提出极为严格的要求,数据库异常将直接导致爬虫系统的崩溃。综上所述,无论是基于内存的 URL 去重还是基于数据库的 URL 去重都很难适应于大规模的分布式网络爬行环境,无法有效解决分布式网络爬行系统中实际爬行结点动态加入与退出所造成的重复爬行问题。

发明内容

[0006] 本发明的目的在于提供一种分布式网络爬虫系统中的 URL 去重方法,其可有效解决实际爬行结点动态加入与退出所造成的重复爬行问题,通过引入虚拟爬行结点及布隆过滤器(英文为 Bloom Filter)去重结构,实现了分布式 URL 去重方法,使得分布式网络爬行系统能稳定持久运行,去重数据结构可全部存放内存中,去重效率高,且当系统重新进行任务划分后,相应的 URL 去重数据结构能高效转移,保证系统不会重复爬行。

[0007] 本发明是通过以下技术方案实现的:

[0008] 一种分布式网络爬虫系统中的 URL 去重方法,在初始阶段先设定系统的虚拟爬行结点数 N 及初始 URL 任务集,每一个虚拟爬行结点对应一个布隆过滤器去重结构,布隆过滤器去重结构在内存中申请有一个位数组,位数组大小为 n,将虚拟爬行结点平均分配给实际爬行结点,建立虚拟爬行结点与实际爬行结点的映射关系,采用基于模的哈希运算方式将初始 URL 任务集映射到各个虚拟爬行结点,并根据虚拟爬行结点与实际爬行结点的映射关系将初始 URL 任务集映射到对应的实际爬行结点。

[0009] 本发明的方法还包括以下步骤:

[0010] (1) 当前爬行结点根据初始 URL 任务集进入网络爬行状态,并获取待处理 URL 任务集;

[0011] (2) 判断待处理 URL 任务集是否为空,若为空则过程结束,否则进入步骤(3);

[0012] (3) 从待处理 URL 任务集中获取 URL 任务;

[0013] (4) 对获取的 URL 任务进行哈希运算得到其对应的虚拟爬行结点号;

[0014] (5) 根据该虚拟爬行结点号,查找虚拟爬行结点与实际爬行结点的映射关系表,从而得到对应的实际爬行结点;

[0015] (6) 判断获取的 URL 任务是否属于当前爬行结点,如果是则进入步骤(9),否则转入步骤(7);

[0016] (7) 将获取的 URL 任务转发给实际爬行结点;

[0017] (8) 对于待处理 URL 任务集中的所有 URL 任务,重复上述步骤(2)至(7),直到所有 URL 任务均处理完毕为止;

[0018] (9) 对该 URL 任务进行哈希运算,以找到该 URL 任务所属的虚拟爬行结点,并找到对应的布隆过滤器去重结构;

[0019] (10) 对 URL 任务用 K 个哈希函数计算,以得到 K 个哈希值 H[0],H[1],H[2],...,H[K-1];

[0020] (11) 根据 K 个哈希值查找布隆过滤器去重结构的位数组中对应的第 H[0],H[1],H[2],...,H[K-1] 位,以判断第 H[0],H[1],H[2],...,H[K-1] 位是否均为 1,若均为 1,则进入步骤(12),否则进入步骤(13);

- [0021] (12) 丢弃该 URL 任务,然后转入步骤 (14) ;
- [0022] (13) 表明该URL任务不是处于当前爬行结点的URL任务集合中, 将该URL任务加入当前爬行结点的待处理队列中 ;
- [0023] (14) 将布隆过滤器去重结构的位数组中对应的第 H[0], H[1], H[2], … , H[K-1] 位全部置 1 ;
- [0024] (15) 判断是否有实际爬行结点退出,若有则转入步骤 (16),否则转入步骤 (25) ;
- [0025] (16) 设定退出的实际爬行结点所映射的虚拟爬行结点集合为 S, S 中虚拟爬行结点个数记为 vz ;
- [0026] (17) 将剩余的实际爬行结点按照其所映射的虚拟爬行结点数从小到大排序,剩余的实际爬行结点个数记为 rz ;
- [0027] (18) 初始化计数器 i 和计数器 j 均为 0,其中 i 指向排序后的实际爬行结点集合中第 i+1 个实际爬行结点, j 指向虚拟爬行结点集合 S 中第 j+1 个虚拟爬行结点 ;
- [0028] (19) 判断 j 是否等于 vz,若是则转入步骤 (25),否则转入步骤 (20) ;
- [0029] (20) 从虚拟爬行结点集合 S 中取出第 j+1 个虚拟爬行结点,并将其加入到第 i+1 个实际爬行结点所映射的虚拟爬行结点集合中 ;
- [0030] (21) 设置 i = i+1, j = j+1 ;
- [0031] (22) 判断 i 是否等于 rz,若是则转入步骤 (23),否则转入步骤 (24) ;
- [0032] (23) 将 i 值置 0 ;
- [0033] (24) 重复上述步骤 (19) 至 (23),直到 j 值等于 vz 为止 ;
- [0034] (25) 判断是否有新的实际爬行结点加入,若有则转入步骤 (26),否则返回步骤 (2) ;
- [0035] (26) 设定 average = N/M, M 为加入新实际爬行结点后实际爬行结点 的总数,设定新实际爬行结点所映射的虚拟爬行结点集合为 Q,其初始为空 ;
- [0036] (27) 将原有实际爬行结点按照映射的虚拟爬行结点数从大到小排序 ;
- [0037] (28) 初始化计数器 s 和 t 均为 0,其中 s 指向排序后的实际爬行结点集合中第 s+1 个实际爬行结点, t 为虚拟爬行结点集合 Q 中虚拟爬行结点个数 ;
- [0038] (29) 判断 t 是否等于 average,若是则返回步骤 (2),否则转入步骤 (30) ;
- [0039] (30) 从第 s+1 个实际爬行结点所映射的虚拟爬行结点集合中取出一个虚拟爬行结点加入到 Q 中,并设置 t = t+1, s = s+1 ;
- [0040] (31) 判断 s 值是否等于 M-1,若是则转入步骤 (32),否则转入步骤 (33) ;
- [0041] (32) 将 s 值置 0 ;
- [0042] (33) 重复执行上述步骤 (29) 至 (32),直到 t 等于 average 为止。
- [0043] 与现有技术相比,本发明具有以下的优点和技术效果 :
- [0044] 1、能有效地处理分布式爬行系统中实际爬行结点的动态加入与退出所造成 的重复爬行问题 ;
- [0045] 2、在参与实际爬行结点数目的变化这一过程中,能很快重新进行任务划分,提高了系统调度的效率,极大地减少了系统调度的性能损失 ;
- [0046] 3、URL 去重数据结构可存放在内存中,进行快速去重,在需要时可方便地进行转移和备份,防止由于去重信息不全或缺失造成爬虫系统重复爬行 ;

[0047] 4、本发明满足了分布式网络爬行系统中的任务调度与 URL 去重的要求,具有极高的应用价值。

附图说明

[0048] 图 1 是本发明分布式网络爬虫系统中的 URL 去重方法的流程图。

[0049] 图 2 是本发明分布式网络爬虫系统中的 URL 去重方法的示意图。

[0050] 图 3 是本发明实际爬行结点退出时的示意图。

[0051] 图 4 是本发明实际爬行结点加入时的示意图。

具体实施方式

[0052] 以下首先对本发明的技术术语进行解释和说明：

[0053] 布隆过滤器去重结构：于 1970 年被提出,是一种用于判断集合中元素是否存在的基于哈希的查找结构。

[0054] 下面结合附图和具体实施方式对本发明技术方案做进一步详细说明。

[0055] 本发明的主要步骤为 URL 任务划分和对 URL 的去重,即将 URL 任务平均分配到每一个实际爬行结点,并对实际爬行结点所维护的 URL 任务集进行处理,去除重复的 URL,同时采用高效的调度算法处理实际爬行结点的动态加入与退出。

[0056] 如图 1 所示。本发明分布式网络爬虫系统中的 URL 去重方法包括以下步骤：

[0057] (1) 设定系统的虚拟爬行结点数 N 及初始 URL 任务集,每一个虚拟爬行结点对应一个布隆过滤器去重结构,布隆过滤器去重结构在内存中申请有一个位数组,位数组大小为 n;实际爬行结点数是指实际运行爬行程序的主机的数目,而虚拟爬行结点数为假设的爬行结点数目,这一数目可灵活配置,但应大于实际爬行结点数,位数组大小 n 根据虚拟爬行结点和实际爬行结点数灵活设置,初始将位数组的所有位均置为 0。

[0058] (2) 将虚拟爬行结点平均分配给实际爬行结点,建立虚拟爬行结点与实际爬行结点的映射关系;通常多个虚拟爬行结点对应一个实际爬行结点,每个实际爬行结点都保存一份虚拟爬行结点与实际爬行结点映射关系的表结构;

[0059] (3) 采用基于模的哈希运算方式将初始 URL 任务集映射到各个虚拟爬行结点,并根据虚拟爬行结点与实际爬行结点的映射关系将初始 URL 任务集映射到对应的实际爬行结点;映射模型为 : $\text{hash}(\text{key}) \% N$,key 值取 URL 的主机部分(域名),以便尽量将同一个主机的 URL 任务分配到同一个实际爬行结点上;

[0060] (4) 当前爬行结点根据初始 URL 任务集进入网络爬行状态,并获取待处理 URL 任务集;

[0061] (5) 判断待处理 URL 任务集是否为空,若为空则过程结束,否则进入步骤 (6);

[0062] (6) 从待处理 URL 任务集中获取 URL 任务;

[0063] (7) 对获取的 URL 任务进行哈希运算得到其对应的虚拟爬行结点号;

[0064] (8) 根据该虚拟爬行结点号,查找虚拟爬行结点与实际爬行结点的映射关系表,从而得到对应的实际爬行结点;

[0065] (9) 判断获取的 URL 任务是否属于当前爬行结点,如果是则进入步骤 (12),否则转入步骤 (10);

- [0066] (10) 将获取的 URL 任务转发给实际爬行结点；
- [0067] (11) 对于待处理 URL 任务集中的所有 URL 任务，重复上述步骤 (5) 至 (10)，直到所有 URL 任务均处理完毕为止；
- [0068] (12) 对该 URL 任务进行哈希运算，以找到该 URL 任务所属的虚拟爬行结点，并找到对应的布隆过滤器去重结构；
- [0069] 本发明利用多个布隆过滤器去重结构实现分布式 URL 去重，每个虚拟爬行结点分配一个布隆过滤器去重结构。虚拟爬行结点数越多，单个虚拟爬行结点被分配映射到的 URL 任务越少，则布隆过滤器去重结构需要的内存也随之减少。参与爬行的主机越多，单个实际爬行结点映射的虚拟爬行结点数目也越少，需要的内存也越少。
- [0070] 具体而言，如图 2 所示，N 个虚拟爬行结点对应有 N 个布隆过滤器去重结构，分别为 Bloom Filter[0], Bloom Filter[1], …, Bloom Filter[N-1]，对 URL 任务哈希运算找到所属的虚拟爬行结点 T，进而找到该虚拟爬行结点对应的 Bloom Filter[T]。
- [0071] (13) 对 URL 任务用 K 个哈希函数计算，以得到 K 个哈希值 H[0], H[1], H[2], …, H[K-1]；
- [0072] (14) 根据这 K 个哈希值查找布隆过滤器去重结构的位数组中对应的第 H[0], H[1], H[2], …, H[K-1] 位，以判断第 H[0], H[1], H[2], …, H[K-1] 位是否均为 1，若均为 1，则进入步骤 (15)，否则进入步骤 (16)；
- [0073] 具体而言，如图 2 所示，位数组表示形式为 bit[0], bit[1], …, bit[n-1]，在 Bloom Filter[T] 中查找 bit[H[0]], bit[H[1]], …, bit[H[K-1]] 的值是否均为 1，若均为 1，表示该 URL 任务处于当前爬行结点的 URL 任务集合中，否则表明该 URL 任务不是处于当前爬行结点的 URL 任务集合中。
- [0074] (15) 表明该 URL 任务处于当前爬行结点的 URL 任务集合中，丢弃该 URL 任务，然后转入步骤 (17)；
- [0075] (16) 表明该 URL 任务不是处于当前爬行结点的 URL 任务集合中，将该 URL 任务加入当前爬行结点的待处理队列中；
- [0076] (17) 将布隆过滤器去重结构的位数组中对应的第 H[0], H[1], H[2], …, H[K-1] 位全部置 1；
- [0077] 具体而言，如图 2 所示，将 Bloom Filter[T] 位数组中的 bit[H[0]], bit[H[1]], …, bit[H[K-1]] 全部置 1。
- [0078] (18) 判断是否有实际爬行结点退出，若有则转入步骤 (19)，否则转入步骤 (28)；
- [0079] (19) 设定退出的实际爬行结点所映射的虚拟爬行结点集合为 S，S 中虚拟爬行结点个数记为 vz；
- [0080] 具体而言，如图 3 所示，实际爬行结点 (4 号) 退出爬行时，S 即为 4 号实际爬行结点所映射的虚拟爬行结点集合，该集合有虚拟结点号 {19, 20, 21, 22, 23, 24}，vz 值为 6。
- [0081] (20) 将剩余的实际爬行结点按照其所映射的虚拟爬行结点数从小到大排序，剩余的实际爬行结点个数记为 rz；
- [0082] 如图 3 所示，将实际爬行结点 1, 2, 3 按照映射的虚拟爬行结点数从小到大排序，此处三个实际爬行结点所映射的虚拟爬行结点个数相等，rz 值即为 3。
- [0083] (21) 初始化计数器 i 和计数器 j 均为 0，其中 i 指向排序后的实际爬行结点集合

划分过程中哈希函数不发生变化，不会导致 URL 的哈希值发生变化，因此系统 URL 任务划分是稳定明确的，不会导致任务混乱和大规模数据转移的开销。

[0104] 采用多个布隆过滤器进行去重，可以减少单个布隆过滤器占用的内存大小，以便将其全部放入内存中，获得更快的去重速度。另一方面，在网络爬虫的实际爬行结点数出现变化（加入、退出）时，需要将对应变化实际爬行结点的去重数据结构进行转移，以保证系统整体不会重复爬行。以实例论述，假设 crawler_a 对应 1 号和 2 号虚拟爬行结点，拥有两个 URL 去重结构 bf_1 和 bf_2，crawler_b 对应 3 号和 4 号虚拟爬行结点，拥有两个 URL 去重结构 bf_3 和 bf_4，新实际爬行结点 crawler_c 加入系统爬行时，经过任务重新划分，crawler_c 对应 2 号和 4 号虚拟爬行结点，则 crawler_a 需将 2 号虚拟爬行结点去重结构 bf_2 转移到 crawler_c，crawler_b 同样亦需将 4 号虚拟爬行结点去重结构 bf_4 转移到 crawler_c，这样 crawler_c 就获取了对应任务虚拟爬行结点的去重信息，不会重复爬行下载。并且新实际爬行结点加入时，所有实际爬行结点的任务变动规模小，不会发生大量的去重布隆过滤器的转移，减少了通信带宽和对实际爬行结点的压力。当实际爬行结点退出时，操作方式与此类似，只是转移的方向为退出实际爬行结点到其他正常实际爬行结点，综合而言，该方法具有很好的操作性和稳定性，适用于大规模分布式网络爬虫的并行爬行。

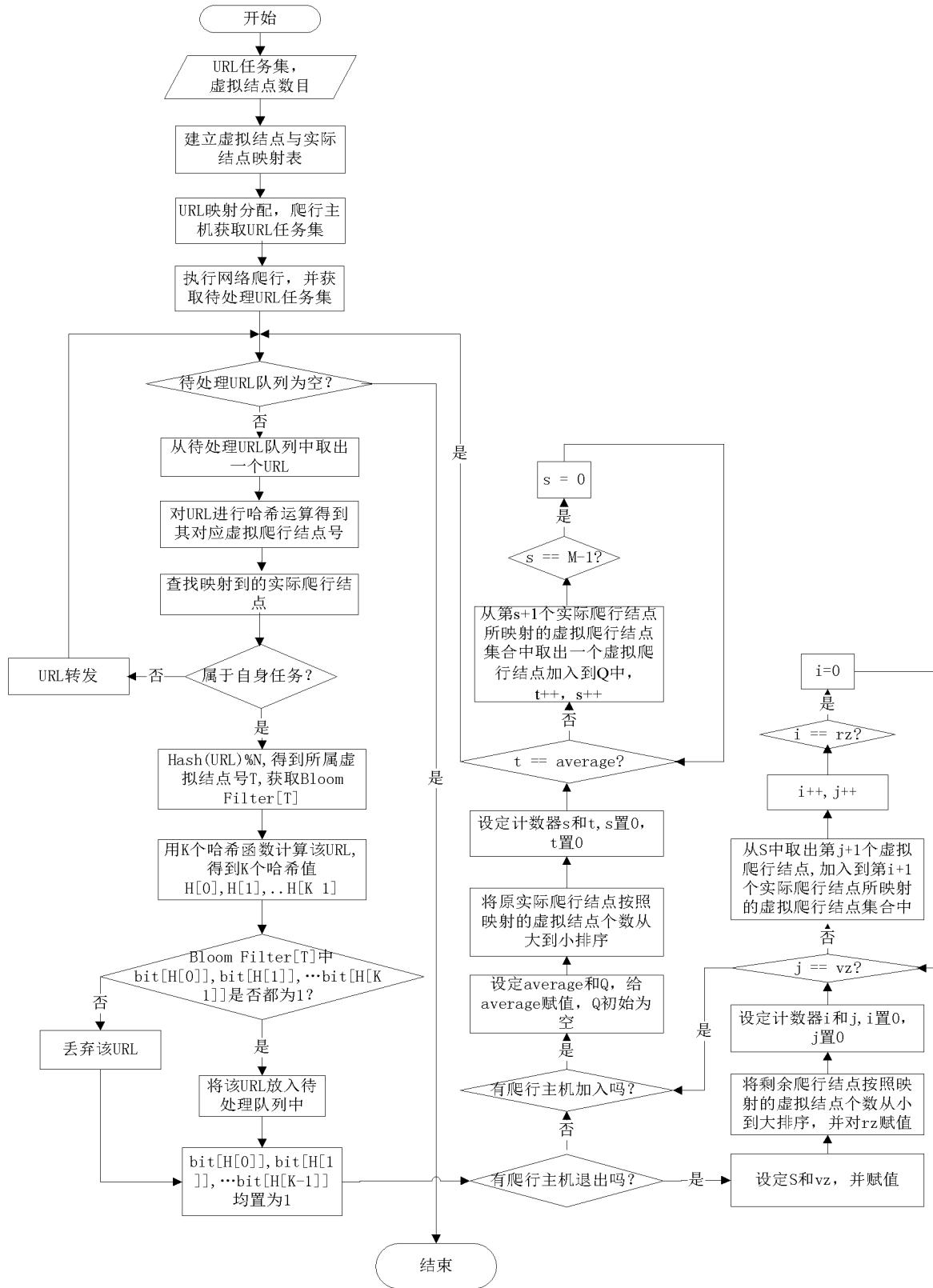


图 1

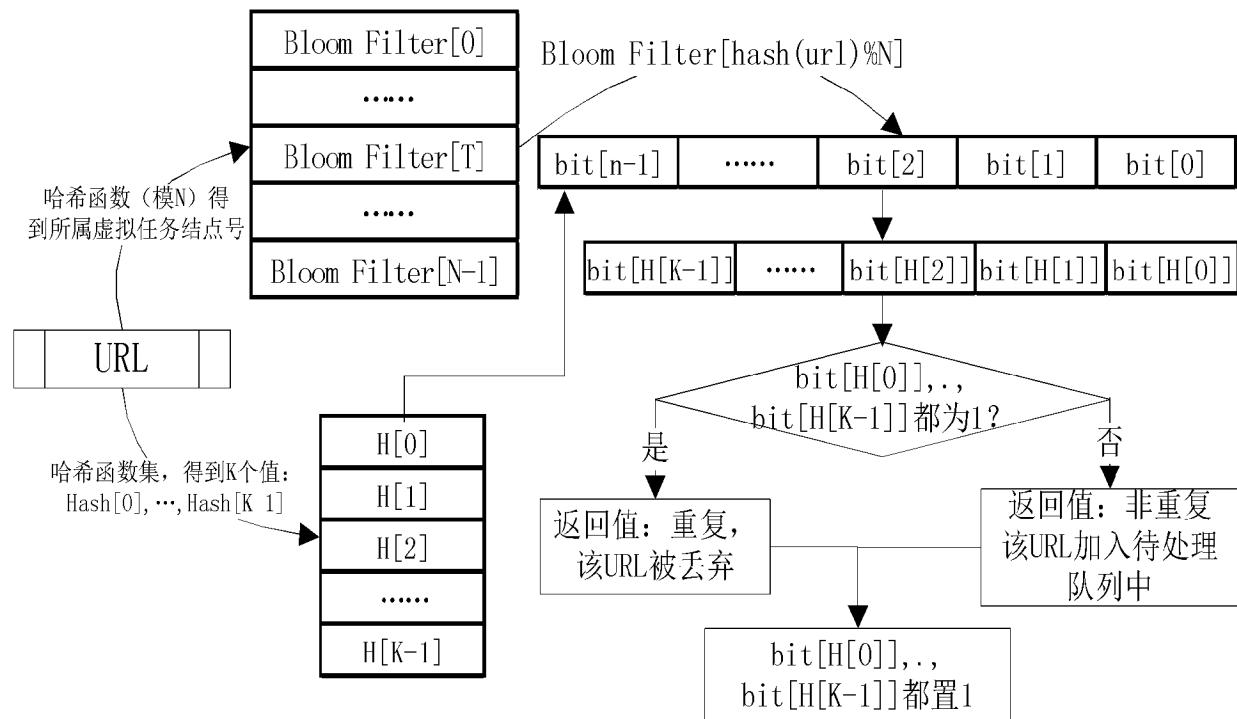


图 2

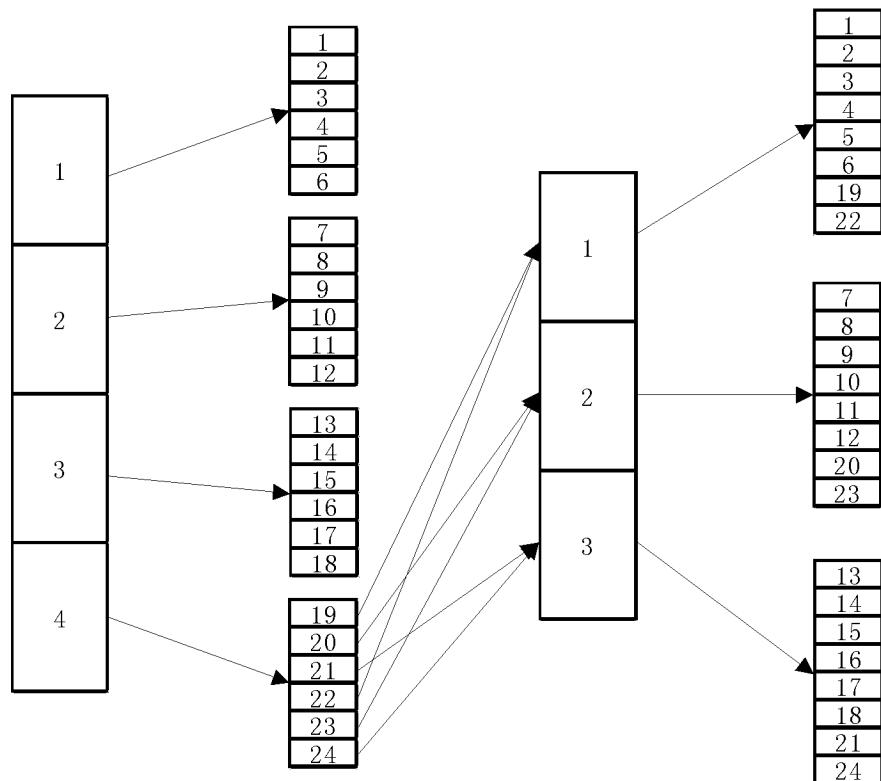


图 3

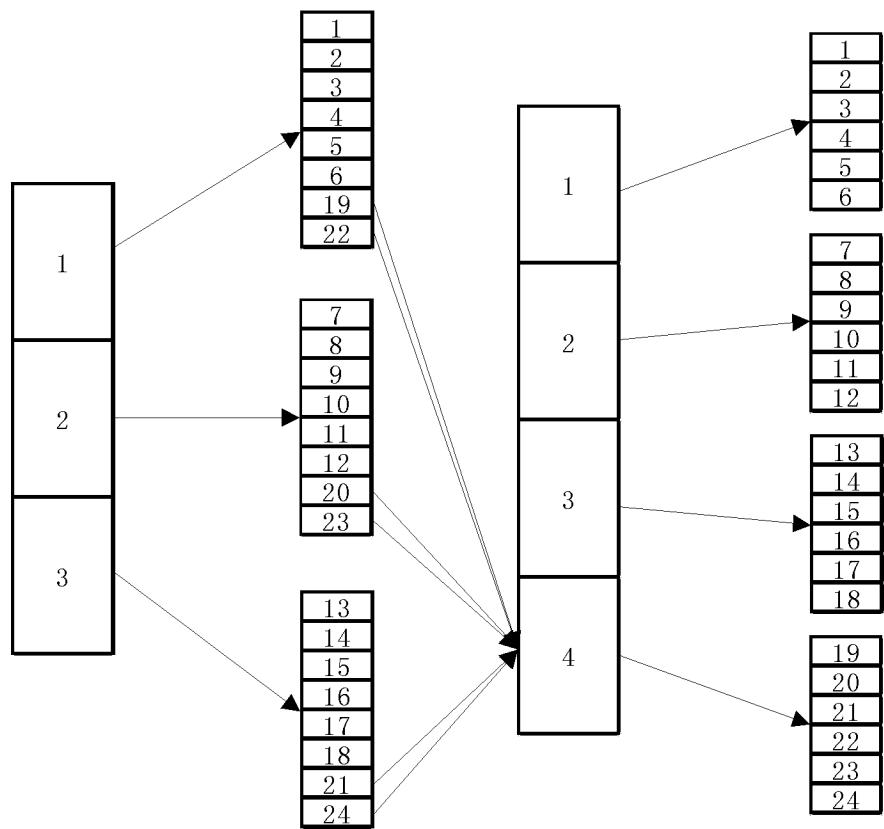


图 4