文章编号:1671-4598(2018)10-0269-07 DOI:10.16526/j.cnki.11—4762/tp.2018.10.058 中图分类号:TP301 文献标识码:A

基于 Hadoop 的分布式并行增量爬虫技术研究

刘芳云，张志勇，李玉祥
（河南科技大学 信息工程学院，河南 洛阳 471023）

摘要：面对多媒体社交网络中在线视频的爆炸式增长，使用单机模式下爬虫提取新视频页面的效率低下，为此，提出一种基于 Map/Reduce 的并行算法，大大提高了爬虫的效率；但是为了进一步改善数据冗余问题，减少过时页面的更新，改进了一种精度感知增量更新算法，利用监控技术监控网页变化情况，分析网页更新模式，增加新鲜度评估和降维处理，使用混合整数二次规划方法为发生更改的网页制定最优的刷新策略；实验证明，相比单机模式下定期频繁的刷新策略，该并行增量方法以原刷新代价的 36.7%获得了 79%的信息精确度，爬虫效率提高了 167 倍。

关键词：Hadoop 集群；分布式爬虫；并行爬虫；增量爬虫；刷新策略

Research on Distributed Parallel Incremental Crawlers Technology Based on Hadoop

Liu Fangyun, Zhang Zhiyong, Li Yuxiang
(College of Information Engineering, Henan University of Science and Technology, Luoyang 471023, China)

Abstract: In response to the explosive growth of online video in multimedia social networks, the use of crawlers in stand-alone mode to extract new video pages is inefficient. a parallel algorithm based on Map/Reduce is proposed, which greatly improves the crawler efficiency. But in order to further handle the problem of data redundancy and reduce outdated page updates, a improved accuracy-aware incremental updating algorithm is proposed. The monitoring technique is used to monitor the web page changes, analyze the web page update mode, increase the freshness assessment and dimensionality reduction, and use the improved mixed integer quadratic programming (MIQP) so to make the optimal refresh strategy. Experiments show that compared with the frequent refresh strategy in the stand-alone mode, the parallel incremental method achieves 79% of the information accuracy with the original refresh rate of 36.7%, and the crawler efficiency is improved by 167 times.

Keywords: Hadoop cluster; distributed crawler; parallel crawler; incremental crawler; refresh strategy

0 引言

爬虫[1]一般用于搜索引擎，实现快速、高效的从庞大的互联网信息中找到用户感兴趣的信息。爬虫还用于研究人员所需要的研究数据的采集[2]。但是随着大数据和 Web2.0 时代的到来，多媒体社交网络上的各类信息都“爆炸式”增长，单机爬虫的效率和更新速度已经无法满足用户的需要，将并行技术用于爬虫可有效改善爬虫的效率[3]，大数据环境下，并行爬虫都是在分布式架构[4-5]上实现的，因为分布式爬虫比单机多核并行爬虫更适用于大数据环境。

为了获取互联网上的最新信息，全部重爬技术存在刷新代价大和数据冗余等问题，增量爬虫技术可以有效解决该问题[6]。增量爬虫技术就是利用特定的页面刷新策略保证页面副本的时新性。其中，针对页面变化的研究是制定页面刷新策略的重点。通过采样根据样本的变化规律估计整个网站的变化规律的方法可以保证页面副本的时新性，但是还是存在数据冗余的问题。故通过历史记录监控网页的变化，提出基于时间感知的增量更新算法，最后和基于 Map/Reduce 的并行算法相结合，在提出的分布式集群架构上验证算法的性能和效率。

主要贡献：1）在 Hadoop 分布式集群的基础上提出基于 Map/Reduce 框架的并行算法；2）提出基于时间感知的增量更新算法，监控网页的更新模式，计算出不同时间产生的页面的相似性得分，通过页面的相似性得分序列计算出时间感知相似性协方差矩阵，通过调整爬虫频次和最大相似性阈值等参数，使用混合整数二次规划方法优化目标函数，得出最优的刷新策略，能以更低的刷新代价获得更好的信息精确度和信息新鲜度；3）将增量爬虫和分布式并行爬虫相结合，应用于真实的数据集上，证明算法有较好的性能和效率。

余下部分的组织如下：第 1 部分是研究的相关工作进展，第 2 部分介绍分布式并行爬虫框架、并行爬虫和增量爬虫及相关算法，第 3 部分是实验结果分析及应用，最后是给出结论。

收稿日期:2017-12-11； **修回日期**:2018-01-08。

基金项目:基金项目国家自然科学基金(61772174,61370220);河南省科技创新杰出人才计划项目(174200510011);河南省高校科技创新团队支持计划项目(15IRTSTHN010)。

作者简介:刘芳云(1991-),女,河南漯河人,硕士生,主要从事并行计算等方向的研究。

通讯作者:张志勇(1975-),男,河南新乡人,博士后,教授,主要从事社交网络安全、并行计算等方向的研究。

1 相关研究工作

文献［7］为了快速获取微博数据，在单进程爬虫的基础上进行了并行框架的扩展，实现了基于 MPI 的并行数据抓取功能，该并行爬虫拥有较好的加速比，可以快速地获取数据，并且这些数据具有实时性和准确性。文献［8］提出一种基于 Kademlia 的全分布式爬虫集群方法，该方法能构建高效、均衡、可靠、可大规模拓展的全分布式爬虫集群。文献［9］研究的是基于 Hadoop 的分布式爬虫技术，并在该分布式爬虫的基础上实现爬虫的并行化处理，从节点不仅在各节点之间并行处理主节点分配的各个分任务，而且从节点内部也多线程的并行处理内部任务。

关于增量爬虫的页面的刷新策略，Tan[10] 等人利用采样样本的方式来确定刷新时刻。文献［11］提出了基于泊松（Poisson）分布的页面刷新策略，作为增量爬虫的页面刷新方法。大量研究证明网页的变化一般遵循泊松过程，根据这个规律可以为网页的变化建立刷新模型，以此来预测网页的下次更新时间。C Olston[12] 等人将基于网页特征的采样和符合 Poisson 分布的这种周期性的变化结合起来，提出了一种基于信息周期的刷新策略，根据上下效用值边界来动态地调整网页的刷新周期。文献［13］在 C Olston 等人的基础上改进 Super－shingle 算法，使其适用于视频资源的爬虫。由于 C Olston 等人引入的估计效用值的方法没有实际意义，故文献［14］给出了一种实用有效的估计效用阈值的方法，与以前的基于边界的方法相比，该基于效用值的方法更好的平衡了新鲜度和刷新成本，以更低的成本获得更好的新鲜度。K Gupta[15] 等人提出一种精度感知的云爬虫技术，使得在资源/预算受限环境中对本地数据重新抓取以便高精度的检索最大信息量。

综上所述，现有的爬虫技术还没有在效率、刷新代价和新鲜度上有较好的平衡，故提出了分布式并行爬虫来提高爬虫的效率，提出基于时间感知的增量更新算法以更低的刷新代价获得更好的新鲜度，并将两者结合起来更好的平衡效率、刷新代价和新鲜度。

2 分布式并行增量爬虫

2.1 基于 Hadoop 的分布式并行爬虫框架

论文使用的分布式爬虫系统采用主从结构，即一个主控节点控制所有从节点执行抓取任务，主控节点负责分配任务，保证集群中所有从节点的负载均衡。使用的分配算法是计算每一个 URL 对应主机的哈希值，然后将相同主机的 URL 分到一个分区里。这样做的目的是使相同主机的 URL 在一台机器上被爬取。分布式爬虫可以看作是多个集中式爬虫系统组合而成，每一个从节点都相当于一个集中式爬虫系统，这些集中式爬虫系统在分布式爬虫系统中由一个主控节点来控制和管理使其能够协同工作。

从图 1 的分布式并行爬虫框架图中我们可以看到，框架中的主要部分包括负责任务生成、任务分配和调度以及控制管理整个系统（如爬虫的深度，更新时间配置、系统的启动和停止等）的主控节点；负责并行下载页面的爬虫集群；负责解析页面、优化链接和网页更新的 MapReduce 功能模块；负责主控节点、爬虫节点和集群之间的通信和协作（如日志管理、爬虫集群间数据交换及运行维护）的消息中间件；和负责数据存储的分布式文件系统（HDFS）。

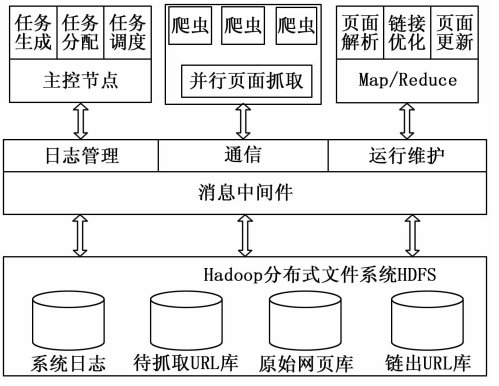

图 1 基于 Hadoop 的分布式并行爬虫框架图

2.2 并行爬虫

为了能对存储在 HDFS 中的大规模数据进行并行化的计算处理，Hadoop 又提供了一个称为 Map/Reduce 的并行化计算框架。该框架能有效管理和调度整个集群中的节点来完成并行化程序的执行和数据处理，并能让每个从节点尽可能对本地节点上的数据进行本地化计算。

整个爬虫系统的核心部分可以分为 3 大模块，分别为下载模块，解析模块和优化模块。每个模块都是一个独立的功能模块，每个模块对应着一个 Map/Reduce 过程。

1）下载模块可以实现并行下载未抓取列表。具体下载是在 Reduce 阶段完成的。

2）解析模块可以实现并行分析已下载网页，提取出链出链接。该模块只需要一个 Map 阶段即可完成目标，还通过规则限制链出链接的类型，防止抽取出的链接链接到其他网站上。

3）优化模块可以实现并行优化链出链接集合，过滤掉重复链接。

可以看出 Web 爬虫系统的并行化是通过这 3 个可并行化的模块实现的，实质上也就是通过 Map/Reduce 的并行化计算框架实现的。

在 Map/Reduce 任务开始时，输入数据会被切分成若干个分片（split），默认每 64M 为一个分片。每一个分片都由一个 Map 进程处理，一个爬虫可以同时开启多个 Map 进程。所有 Map 的输出结果合并之后，根据分区算法，将相同站点的 URL 分配到一个分区中，这样可以使相同站点的 URL 在同一台机器上被爬取，每一个分区的任务由一个 Reduce 进程处理，若干个分区有若干个 Reduce 进程并行处理，同时一个爬虫还可以开启多个 Reduce 进程。最后将并行执行的结果保存到 HDFS 上。

定义 1：$Crawler=\{c_1, c_2, \ldots, c_n\}$：表示集群中爬虫节点的集合。$c_i$ 表示第 i 个爬虫节点。一个爬虫节点可以开

启的最大 Map 进程数和最大 Reduce 进程数由节点上的处理器个数决定。

定义 2：$\{split_0, split_1, \ldots, split_{m-1}\}$：表示文件分片的集合。一个分片由一个 Map 进程处理。

定义 3：$\{part_0, part_1, \ldots, part_{k-1}\}$：表示文件分区的集合。一个分区由一个 Reduce 进程处理。

假设 $m = 2n, k = n$，那么基于 Map/Reduce 的并行算法如 Algorithm 1 所示。Algorithm 1：Parallel algorithm based Map/ReduceInput：Input－File

Output：Output－File

1：Begin

2：Split Input－File into m slices ＝＞ $\{split_0, split_1, \ldots, split_{m-1}\}$；

3：send $split_0$，$split_1$ to c_1，open two Map processes to process this two slices

4：and send $split_2$，$split_3$ to c_2，open two Map processes to process this two slices

5：and ....

6：and send $split_{m-2}$，$split_{m-1}$ to c_n，open two Map processes to process this two slices；

7：Combiner all Map output ＝＞ Inter－results；

8：partitioner（Inter－results）＝＞ $\{part_0, part_1, \ldots, part_{k-1}\}$；

9：send $part_0$ to c_1，open a Reduce process to process $part_0$ ＝＞ partition_0

10：and send$part_1$ to c_2，open a Reduce process to process $part_1$ ＝＞ partition_1

11：and ....

12：and send$part_{k-1}$ to c_n，open a Reduce process to process-$part_{k-1}$ ＝＞ partition_n－1；

13：Output－File ＝ {partition_0，partition_1，...，partition_n－1}；

14：End

2.3　增量爬虫

增量爬虫的主要作用就是对数据集合的日常维护与即时更新。该技术只会在需要的时候下载新产生的网页或发生更新的网页，并不重新下载没有发生变化的页面，可有效减少数据下载量，及时更新已爬取的网页，减小资源的耗费，保持本地数据和集成的 Web 数据的一致性。

下面是基于精度感知的页面刷新策略[15]的改进。首先根据爬取网页目标的不同设置不同的爬行计划，然后在精度感知的基础上增加新鲜度感知来确定最优的爬行计划，最后对基于页面相似性得分的时间感知相似性协方差矩阵进行降维处理来优化混合整数二次规划方法，提高算法的效率。

定义 4：$T = \{t_1, t_2, \ldots, t_n\}$：表示初始爬行计划中的爬行时间戳列表。

定义 5：页面 P 的爬行计划（表示为 T_P）是执行爬行的时间戳的顺序集。爬行计划 T_P 中的时间戳数 $|T_P|$ 称为爬行频次 f。其中，$f = |T_P| < |T|$。

定义 6：$P = \{P_1, P_2, \ldots, P_n\}$：表示页面在不同时间戳产生的快照的集合，$P_i$ 是页面 P 在时间戳 t_i 的快照。

为了简化研究，只考虑从给定页面 P_i 提取相关科技链接，而不是完整的页面内容。因为新的科技以新的链出链接的形式发布重要内容，其中每个科技链出链接都链接到视频页面。总之，每个 P_i 现在可以被看作是相关科技链接的集合。

定义 7：$Accuracy$：爬行计划的信息精确度，表示为相对于基线爬行计划 T 捕获的信息 N_T，通过爬行计划 T_P 捕获的信息的百分比。令 N_{TP} 是使用爬行计划 T_P 获得的一组内容（链出链接）。那么，$Accuracy_{TP}$ 定义如下：

$$Accuracy_{T_P} = 100 * \frac{|N_{T_P}|}{|N_T|} \tag{1}$$

定义 8：$Freshness$：爬行计划的信息新鲜度，从侧面反应着所抓取的元素中当前为新元素的比例。即相对于基线爬行计划 T 捕获的信息 N_T，通过爬行计划 T_P 捕获的最新信息的百分比。令 F_{TP} 是使用爬行计划 T_P 获得的一组最新内容。那么，$Freshness_{TP}$ 定义如下：

$$Freshness_{T_P} = 100 * \frac{|F_{T_P}|}{|N_T|} \tag{2}$$

定义 9：$S(P_i, P_j)$：表示不同时间戳上爬行的两个页面之间科技视频链接的相似性得分。

$$S(P_i, P_j) = \frac{|P_i \cap P_j|}{|P_i \cup P_j|} \tag{3}$$

定义 10：PTS：表示页面相似性得分时间序列。给定 n 个爬行页 $P = \{P_1, P_2, \ldots, P_n\}$ 的顺序集，页面相似性得分的时间序列是计算 P 中连续的两个页面相似性得分的 $n-1$ 个值的序列。

$$PTS = \{S(P_1, P_2), S(P_2, P_3), \ldots, S(P_{n-1}, P_n)\} \tag{4}$$

例如，保持爬行固定时间段大小为 1 个月，因为科技类视频网站更新频率低。初始爬行计划是 T ＝ {1 号，2 号，...，30 号}。如果每月爬行次数为 5 次，在随机选择技术中，从 T 随机选择不同的时间戳。在统一分配方案中，以均匀的间隔从 T 选择爬行时间戳。则每 6 天即可进行统一的爬行策略，T_P ＝ {1 号，7 号，13 号，19 号，25 号}，这两种方法是无监控的，没有利用网页更新模式的特征，无法获得更好的精度和时新性。故需要使用监控技术获取网页的更新模式，分析计算出最优的爬行计划使得网页更新的精度和新鲜度都是最优的。

定义 11：$\boldsymbol{M}$：表示基于页面相似性得分的时间感知相似性协方差矩阵。令 n 为 T 的基数。$\boldsymbol{M}$ 是 $n \times n$ 矩阵，其中单元格（i，j）中的条目表示在时间戳 t_i 和 t_j 上爬行的页面之间的相似性得分的平均值。假设我们已经监视了 d 个月的页面更新，则单元格（i，j）中的条目表示为：

$$M_{i,j} = \frac{\sum_{k=1}^{d} S(P_i, P_j)}{d} \tag{5}$$

其中：$S(P_i, P_j)$ 表示在第 k 个月爬行的页面 P_i 和 P_j 之间的页面相似性得分。如果页面 P_i 和 P_j 不是连续的

页面，中间隔着一个页面 P_k，那么，

$$S(P_i,P_j) = S(P_i,P_k) \times S(P_k,P_j) \tag{6}$$

理想情况下，应该选择合适的时间戳，使它们在 $\boldsymbol{M}$ 矩阵中具有较小值。间接地，应该寻找一个子集 T'，使得以下功能被较小化。

$$Q(T') = \sum_{i \in Tj \in T} M_{i,j} \tag{7}$$

用枚举的方法可以获得使公式（7）较小化的子集 T'，但是这种方法的时间复杂度是指数级的，为了优化这个方法需要从另一个角度考虑问题。这个问题的本质实际上就是给定一组变量，必须选择一些变量才能实现目标。这里，一组变量对应于爬行的时间戳，目标是使公式（7）中给出的函数较小化。正式的问题定义如下：

假设 t_1，t_2，...，t_n 是爬行时间戳。每个 t_i 与布尔参数 b_i 相关联，使得：

$$b_i = \begin{cases} 1 & \text{if } t_i \text{ is } selected \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

给定子集 T' 的基数 f 和时间感知相似性协方差矩阵 M，重构问题如下所示：

$$\begin{cases} \min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} b_i b_j M_{i,j} \\ s.t. \quad \sum_{i=1}^{n} b_i = f \end{cases} \tag{9}$$

这是一个二进制二次规划问题，可以使用混合整数二次规划求解。混合整数二次规划的目标函数是形式如下。

$$\min \quad 0.5x^T Hx + \alpha^T x \tag{10}$$

其中：$H=2$ M。为了使用混合整数二次规划（MIQP）求解二进制二次规划问题，将 α 设为零向量，将 x 设为大小为 n 的布尔向量。使用混合整数二次规划工具箱 miqp 解决 MIQP 问题。该算法的时间复杂度是多项式级别的，即 $O(|T|^2)$。

可以通过自定义爬行频次 f 调用 MIQP 算法得到页面更新的最佳爬行时间序列 T'，但是如果在最佳爬行时间序列中某个时间戳上的页面几乎没有太明显的更新，或者实现这些更新会花费更多的代价，那么对于这些计划内的页面更新是没有必要的，所以，需要先把页面相似性得分时间序列 PTS 中相似性得分平均值偏高的时间戳过滤掉，然后再计算最佳的爬行时间序列，通过这种降维的方式可以降低计算公式（10）的时间复杂度。如果降维后的维数小于爬行频次 f，那么把该网页标记为“无变化”，在增量抓取过程中不将该网页放入待抓取队列中。基于时间感知的增量更新算法如 Algorithm 2 所示。

2.4 增量式更新分布式并行爬虫

增量更新分布式并行爬虫基本流程说明如下：

1）收集种子集合。先为每一个爬虫目标收集一个 URL 种子作为下载数据的入口链接。同时，设置已抓取层数为 0。

2）判断待抓取列表是否为空。若是，跳转到 7）；否则，执行 3）。

3）并行下载待抓取列表中的页面。

4）并行解析已抓取的网页。

5）并行优化解析出来的链出链接。将优化结果放入待抓取列表中，等待下一轮抓取。

6）判断已抓取层数是否小于 depth。若是，“已抓层数”自加 1，返回 2）；否则进入 7）。

7）合并去重。将每层抓取的网页合并，去掉重复抓取的网页。

8）更新本地数据库。将合并去重后的原始网页保存到本地数据库。检索整个本地数据库，根据 Algorithm 2 得出的网页的爬行计划将拥有此刻爬行任务的页面添加到待爬取列表中。

9）对网页内容做进一步解析。并行分析网页内容，从合并去重后的网页中解析出需要的属性信息，本系统需要的属性信息有标题，发布时间，来源，视频源等。

10）根据解析出的属性信息做进一步筛选。如果属性信息满足用户规则，如发布时间在最近 7 天内。就将满足条件的属性信息包括网页的 URL 上传到服务器数据库中。否则，舍去。

Algorithm 2：Time － aware incremental updating algorithmInput：PTS set，Crawling frequency：f，Initial crawling plan：$T=\{t_1, t_2, \ldots, t_n\}$，the Maximum similarity：$\delta$

Output：Crawl_Queue

1：Begin

2：new_T＝{t_1}，new_PTS＝{}

3：for $m \leftarrow 1$ to N// N indicates the number of Local page

4：if PTS_m isExist

5：for$k \leftarrow 1$ to n－1

6：if$PTS_{mk} < \delta$

7：new_PTS. add(PTS_{mk})

8：new_T. add(t_{k+1})

9：endif

10：endfor

11：if| new_T| >＝ f

12：new_PTS ＝> M

13：MIQP(f，M，new_T) ＝> T’；

14：Crawl_Queue. add(P_m) by schedule T’；

15：endif

16：endif

17：generate a new PTS for P_m

18：endfor

19：End

3 实验及结果分析

为了快捷有效的汇聚海内外最新最前沿的科技视频，开发了基于 Map/Reduce 的并行增量爬虫，该爬虫在分布式框架 Hadoop 集群上实现。搭建的 Hadoop 集群由三台服务器组成，一台作为 Master 节点，两台作为 Slave 节点，节点之间局域网连接，可以相互通信。由三台服务器搭建的

Hadoop 集群的配置如表 1 所示。

表 1　Hadoop 集群配置

Node	Server System	CPU	RAM	Hard Disk Size	Hadoop Version
Master	Ubuntu 16.04	4	250 G	2 T	Hadoop 2.7.0
Slave1	Ubuntu 16.04	2	8 G	270 G	Hadoop 2.7.0
Slave2	Ubuntu 16.04	2	8 G	100 G	Hadoop 2.7.0

3.1　实验数据收集与数据分析

先使用经验数据来说明 MIQP 方案的表现。首先，收集数据并分析。接下来，专注于找到针对给定爬行频次的最佳爬行时间表的问题。与基准策略相比，将基于优化技术来说明解决方案的性能。

3.1.1　实验数据收集

爬虫从 2017 年 11 月 1 日开始，到 2018 年 1 月 30 日结束，以 1 天为间隔监控 3 种不同的科技来源。表 2 列出了数据集中存在的 3 个主要科技来源。

表 2　主要科技来源

ID	Source	URL
1	优酷科技	http://tech.youku.com/? spm=a2hww.20027244.topNav.1～3！9～1～3！2～5～DL～DD～A！17
2	搜狐科技	http://tv.sohu.com/ugc/tec/
3	乐视科技	http://tech.le.com/

收集的数据为每次爬虫的日志数据，部分数据如表 3 所示，Out _ links 字段存储当前页面上的所有链出链接 ID 的集合，对照表 4（本地数据），可找出具体的链出链接集合。

表 3　爬虫记录表

ID	URL	Time	Out_links
2	http://tv.sohu.com/ugc/tec/	2017－11－01 00:03:49	4,5,6,7,8
2	http://tv.sohu.com/ugc/tec/	2017－11－02 00:01:14	4,5,6,7,8
2	http://tv.sohu.com/ugc/tec/	2017－11－03 00:02:38	4,5,6,7,8
2	http://tv.sohu.com/ugc/tec/	2017－11－04 00:02:56	4,5,6,7,8,9
2	http://tv.sohu.com/ugc/tec/	2017－11－05 00:03:14	4,5,6,7,8,9
2	http://tv.sohu.com/ugc/tec/	2017－11－06 00:02:12	4,5,6,7,8,9,10
2	http://tv.sohu.com/ugc/tec/	2017－11－07 00:02:51	4,5,6,7,8,9,10
2	http://tv.sohu.com/ugc/tec/	2017－11－08 00:05:24	4,5,6,7,8,9,10,11
2	http://tv.sohu.com/ugc/tec/	2017－11－09 00:02:23	4,5,6,7,8,9,10,11
2	http://tv.sohu.com/ugc/tec/	2017－11－10 00:03:09	4,5,6,7,8,9,10,11
……			

表 4　本地数据表

ID	URL
1	http://tech.youku.com/? spm=a2hww.20027244.topNav.1～3！9～1～3！2～5～DL～DD～A！17
2	http://tv.sohu.com/ugc/tec/
3	http://tech.le.com/
4	http://my.tv.sohu.com/pl/9112584/90203489.shtml
5	http://my.tv.sohu.com/pl/9253242/90206573.shtml
6	http://my.tv.sohu.com/pl/9034610/80791609.shtml
7	http://my.tv.sohu.com/pl/9204736/88292047.shtml
8	http://my.tv.sohu.com/pl/9034610/85548439.shtml
9	http://my.tv.sohu.com/pl/9253242/91377859.shtml
10	http://my.tv.sohu.com/pl/9211490/88937537.shtml
11	http://my.tv.sohu.com/pl/9211490/88937596.shtml
……	

3.1.2　实验数据分析

将 MIQP 策略和穷举策略，随机策略，均匀策略的进行对比来说明 MIQP 策略的综合性能最优。

首先介绍第 2.3 节讨论的穷举策略和 MIQP 策略的爬虫结果。如前所述，已经爬取了 3 个属于科技类别的新资源 3 个月。使用前 2 个月的数据作为训练数据，以制定最佳爬行计划，剩余的一个月数据作为测试数据。使用训练数据发现给定爬行次数的爬行计划，然后构建跟踪测试数据中每月发现的计划的爬虫程序。该分析的评估指标是信息精确度（见定义 7）和信息新鲜度（见定义 8）。初始计划以 1 天的时间间隔从月初 1 号开始爬行科技视频，到月末 30 号结束。表 5 显示了针对不同爬行频次值的搜狐科技的穷举策略和混合整数二次规划策略的最佳爬行计划。

在表 5 中，我们可以看到，穷举策略和混合整数二次规划（MIQP）策略生成的时间表（以天为单位，1＃代表 1 号）并不完全相同。但是，这两个策略的性能方面仍然与图 2 所示的相似。我们可以看出，穷举策略执行最好，然后是混合整数二次规划策略。这两个计划之间的精确度差异很小，而随机策略和均匀策略的表现是不可预测的。除了考虑信息的精确度之外，还要考虑信息的新鲜度，如图 3 所示，与穷举策略相比，MIQP 策略的信息新鲜度除了个别的差异较大之外，都和穷举策略的相似。然而，如图 4 所示，与穷举策略相比，MIQP 策略的时间复杂度非常低。

综合考虑图 2、图 3 和图 4 的信息精确度数据、信息新鲜度数据和时间复杂度数据，当爬行频次为 11 时，爬虫可以以较小的代价获得最优的性能。

在爬行频次一定的情况下，还可以继续优化 MIQP 策略的效率，如表 6 和图 5 所示，随着最大相似性阈值的降低，信息精确度和新鲜度几乎没有多大变化，而 MIQP 策略的效率在明显提高，当最大相似性阈值继续降低时，信息精确度和新鲜度也明显降低。其中，表 6 中 δ 代表最大相似性阈值，$|\boldsymbol{M}|$ 代表矩阵的维数。

表 5 最优爬行计划表

爬行频次	穷举策略	MIQP 策略
2	1#，27#	2#，27#
3	1#，14#，27#	2#，15#，28#
4	1#，12#，21#，30#	1#，12#，21#，30#
5	1#，8#，16#，24#，30#	1#，8#，16#，24#，30#
6	1#，6#，12#，18#，24#，30#	1#，6#，12#，18#，24#，30#
7	1#，6#，12#，16#，21#，27#，30#	1#，6#，12#，16#，21#，27#，30#
8	1#，5#，9#，14#，19#，22#，27#，30#	1#，5#，9#，14#，19#，22#，27#，30#
9	1#，4#，8#，12#，15#，20#，23#，27#，30#	1#，4#，8#，12#，15#，20#，23#，27#，30#
10	1#，3#，7#，11#，14#，18#，21#，25#，28#，30#	1#，3#，7#，11#，14#，18#，21#，25#，28#，30#
11	1#，3#，6#，9#，13#，15#，19#，21#，25#，28#，30#	1#，3#，6#，9#，13#，15#，19#，21#，25#，28#，30#
12	1#，3#，6#，8#，12#，14#，18#，20#，23#，27#，28#，30#	1#，3#，6#，8#，12#，14#，18#，20#，23#，26#，28#，30#

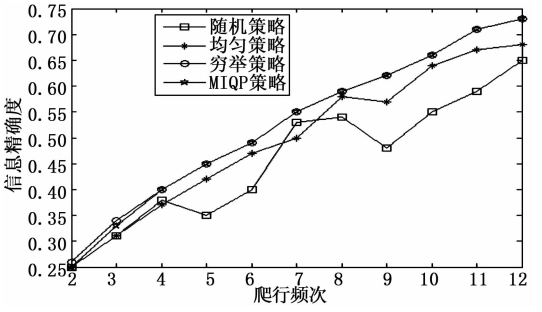

图 2 不同爬行频次下不同策略的信息精确度比较

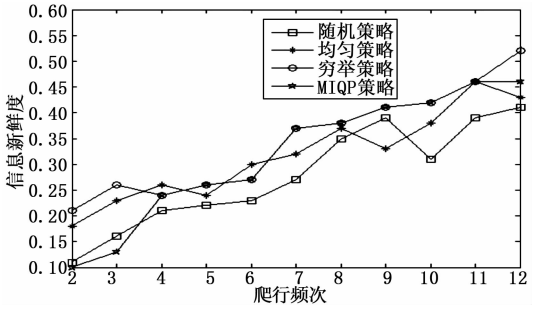

图 3 不同爬行频次下不同策略的信息新鲜度比较

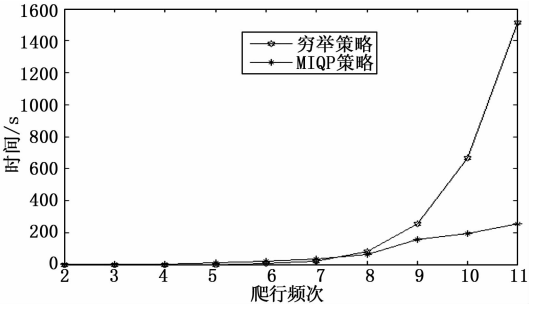

图 4 MIQP 策略与穷举策略的时间复杂度比较

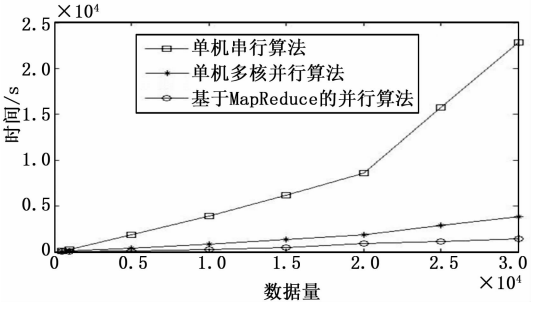

图 5 不同最大相似性阈值下 MIQP 策略的时间复杂度

表 6 最大相似性阈值选取

f	δ	Freshness/%	Accuracy/%	Cost time/s	$\|M\|$
11	1	46	71	254.1	30
	0.95	46	71	201.3	29
	0.9	46	71	140.7	28
	0.85	46	71	78.6	27
	0.8	46	71	14.7	22
	0.75	46	71	3.4	17
	0.7	46	71	2.5	13
	0.65	46	71	1.1	12
	0.6	0	0	0	10

通过分析可以知道当爬行频次为 11，最大相似性阈值为 0.65 时，可以获得 46%的信息新鲜度，71%的信息精确度，MIQP 策略的效率提升了约 230 倍。

3.1.3 算法评价指标及实验数据验证

提出的时间感知增量更新算法的评价指标有三个，分别为信息精确度，信息新鲜度和时间复杂度。信息精确度的定义如 2.3 节中的定义 7 所示。信息新鲜度的定义如 2.3 节中的定义 8 所示。信息精确度和信息新鲜度之间的区别就在于通过爬行计划 T_P 捕获的信息是否是最新的。例如，在初始爬行计划 T = {1 号，2 号，...，30 号} 下每天爬取的最新信息的量形成的序列为 $Newnum$ = {64，11，12，10，8，12，0，0，8，11，9，11，9，0，0，10，9，16，11，11，0，0，14，14，12，7，11，0，0，15}，也即第一天爬取 64 个最新视频，第二天爬取 11 个最新视频（相比于第一天），第三天爬取 12 个最新视频（相比第二天），以此类推，假如，最佳爬行频次为 3，最佳爬行时间序列为 T' = {1 号，13 号，30

号}，则信息精确度和信息新鲜度的分母即基线爬行计划 T 捕获的信息量是一定的，等于基线爬行计划中每天爬取的最新信息之和，而信息精确度的分子（这些信息中某些信息已经失去了时效性）等于第 1 天的最新信息量加上第 13 天相对于第 1 天捕获的最新信息量再加上第 30 天相对于第 13 天捕获的最新信息量，由于页面上的信息都存在一定的信息周期，所以第 13 天相对于第 1 天捕获的最新信息量并不等于 Newnum 序列中第 2 个元素到第 13 个元素之间所有元素的和，而是小于该和。而信息新鲜度的分子（这些信息都具有时效性）等于 Newnum 序列中第 1 个，第 13 个和第 30 个元素之和。时间复杂度的概念我们并不陌生，它描述了算法的运行时间，算法的时间复杂度越高，说明该算法的效率越低。

通过上述的数据分析可知，当爬行频次为 11，可以获得最优的爬行计划 T_{opt} = {1#，3#，6#，9#，13#，15#，19#，21#，25#，28#，30#}，在该爬行计划下对最后一个月收集的实验数据进行验证，通过计算可知，通过该最优爬行计划可以获得 79%的信息精确度和 42%的信息新鲜度。定义的信息新鲜度的新鲜性不超过一天，也即信息的获取时间与信息的发布时间的差值小于一天。由于科技类视频的生存周期比较长，那么我们也可以把信息新鲜度的新鲜性规定在两天之内，这样的话我们通过最优爬行计划就可以获得 73%的信息新鲜度。

3.2　实验结果

3.2.1　基于 MapReduce 的并行爬虫与单机爬虫对比

以搜狐科技频道为例，对单机单核串行爬虫、单机多核并行爬虫以及基于 MapReduce 的并行爬虫作对比，比较它们在处理相同规模的数据时，所花费的时间，以此作为评价爬虫效率的标准。在实验中一共进行了 5 次对比数据，数据量从 500 条增加到 3 万条，实验结果如图 6 所示。

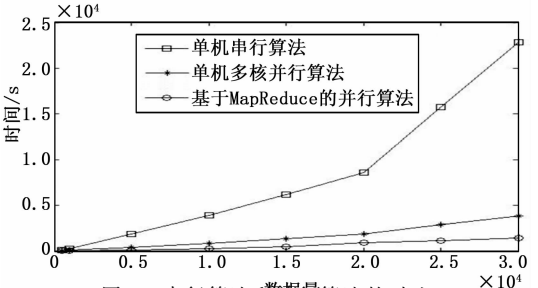

图 6　串行算法和并行算法的对比

从图中可以看出，基于 MapReduce 的并行算法的效率最高，其次是单机多核并行算法，而单机串行算法的效率最低。当数据量达到 2 万时，随着数据量的增加，单机爬虫所花费的时间急剧增加，而基于 MapReduce 的并行爬虫所花费的时间在缓慢增加。

3.2.2　分布式并行增量爬虫与非分布式爬虫效率对比

以搜狐科技为例，以一个月为周期使用分布式并行增量爬虫更新本地搜狐科技数据，单机模式下增量式更新本地数据，单机模式下每天定期重爬。对这三种爬虫的效率，产生的冗余和花费的代价（爬虫更新频次）进行比较。如表 7 所示。

表 7　不同爬虫类型性能对比

爬虫类型	代价	冗余百分率/%	花费的时间/s
分布式并行增量爬虫	11	0.6	341
串行增量爬虫	11	0.6	13 411
串行爬虫	30	78.8	57 150

由表 7 我们可以分析得出，与单机模式下定期全部重爬的爬虫相比，分布式并行增量爬虫以原刷新代价的 36.7%，消除了 99.4%的冗余，爬虫效率提高了 167 倍。与串行增量爬虫相比，分布式并行增量爬虫的效率提高了 39 倍。由此可知，提出的分布式并行增量爬虫可以以较低的刷新代价和较高的爬虫效率获得较优的爬虫性能。

3.2.3　爬虫结果在多媒体社交网络平台（CyVOD.net）的应用

使用开发的分布式并行增量爬虫系统同时对搜狐视频网站、乐视视频网站和优酷视频网站上的科技类最新最热视频进行并行自动化抓取。图 7 展示了搜狐视频网站上抓取的部分数据。其中主要抓取的视频数据信息有视频名称，视频网址，视频发布时间和视频源。图 7 中的 crawltime 字段表示数据的抓取时间，videosource 字段表示视频源文件的存放地址，而视频的原文件存放在云服务器上。自动化程序抓取的数据存放在 CyVOD 平台的数据库中，可以直接在 CyVOD 平台首页展示出来。如图 8 所示是系统运行结果的部分科技视频展示。

id	techname	url	fabutime	crawltime	videosource
6533	废纸能秒变新纸，以后不用买…	http://my.tv.sohu.com/us/2…	2018-02-22	2018-02-24	content/废纸能秒变新…
6534	2018年手机预测：更贵更智…	http://my.tv.sohu.com/us/2…	2018-02-17	2018-02-24	content/2018年手机预…
6535	又一个新东方名师创业，口才…	http://my.tv.sohu.com/us/3…	2018-02-22	2018-02-24	content/又一个新东方…
6536	农村小伙靠墙画年赚百万，画…	http://my.tv.sohu.com/us/1…	2018-02-22	2018-02-24	content/赚疯！小米今…
6537	当心！这样用指纹识别一块橘…	http://my.tv.sohu.com/pl/9…	2018-02-21	2018-02-24	content/赚疯！小米今…
6538	弹幕视频网站AcFUN因何衰落 …	http://my.tv.sohu.com/pl/9…	2018-02-20	2018-02-24	content/弹幕视频网站A…
6539	燃爆！中国民间特技飞行队参…	http://my.tv.sohu.com/pl/9…	2018-02-23	2018-02-24	content/燃爆！中国民…
6540	简单一招建个时间机器,给未…	http://my.tv.sohu.com/pl/9…	2018-02-19	2018-02-24	content/简单一招建个…
6541	微信更新之后这个设置不关掉…	http://my.tv.sohu.com/us/2…	2018-02-22	2018-02-24	content/微信更新之后…
6542	长的像雨伞的骨子，收纳后直…	http://my.tv.sohu.com/pl/9…	2018-02-19	2018-02-24	content/长的像雨伞的…
6543	索尼"复活"十几年前被迫停产…	http://my.tv.sohu.com/us/3…	2018-02-22	2018-02-24	content/索尼"复活"十几…
6544	老美颠覆2块钱粘钩，不用胶…	http://my.tv.sohu.com/us/3…	2018-02-22	2018-02-24	content/老美颠覆2块钱…
6545	总价2000万！平昌冬奥会人手…	http://my.tv.sohu.com/pl/9…	2018-02-22	2018-02-24	content/总价2000万！…
6546	三星S8、S8+新配色勃艮第红…	http://my.tv.sohu.com/pl/9…	2018-02-22	2018-02-24	content/三星S8、S8+新…
6547	小米MIX 2、小米MIX、小米6…	http://my.tv.sohu.com/us/1…	2018-02-22	2018-02-24	content/小米MIX 2、小…
6548	3毫米厚的衣服 在零下196度…	http://my.tv.sohu.com/us/3…	2018-02-22	2018-02-24	content/3毫米厚的衣服 …
6549	价格惊喜！三星S9真机上手来…	http://my.tv.sohu.com/pl/9…	2018-02-22	2018-02-24	content/价格惊喜！三…
6550	你的iPhone被降频了吗？手把…	http://my.tv.sohu.com/us/3…	2018-02-22	2018-02-24	content/你的iPhone被降…
6551	安卓最强芯片！三星猎户座98…	http://my.tv.sohu.com/pl/9…	2018-02-22	2018-02-24	content/安卓最强芯片…
6552	世界首款太阳能电池涂层，涂…	http://my.tv.sohu.com/pl/9…	2018-02-22	2018-02-24	content/世界首款太阳…
6553	华为转行做图片，诺基亚五摄…	http://my.tv.sohu.com/us/3…	2018-02-22	2018-02-24	content/华为转行做图…
6554	1分钟带你看会下海的公交车…	http://my.tv.sohu.com/us/2…	2018-02-22	2018-02-24	content/1分钟带你看会…
6555	iPhone X掉漆愈演愈烈 工作人…	http://my.tv.sohu.com/us/2…	2018-02-22	2018-02-24	content/iPhone X掉漆愈…

图 7　搜狐科技视频部分数据

4　结束语

由于用户对用爬虫来搜索或采集信息的效率，性能和新鲜度的要求越来越高，为了适应互联网激增的数据和网页的动态变化，设计并实现了基于分布式集群的并行爬虫算法。然后通过监控分布式并行爬虫的本地数据库网页更新

（下转第 308 页）