Efficient partitioning strategies for distributed Web crawling

José Exposto[1], Joaquim Macedo[2], António Pina[2], Albano Alves[1], and José Rufino[1]

[1] ESTiG - IPB, Bragança - Portugal
[2] DI - UM, Braga - Portugal

Abstract. This paper presents a multi-objective approach to Web space partitioning, aimed to improve distributed crawling efficiency. The investigation is supported by the construction of two different weighted graphs. The first is used to model the topological communication infrastructure between crawlers and Web servers and the second is used to represent the amount of link connections between servers' pages. The values of the graph edges represent, respectively, computed RTTs and pages links between nodes.
The two graphs are further combined, using a multi-objective partitioning algorithm, to support Web space partitioning and load allocation for an adaptable number of geographical distributed crawlers.
Partitioning strategies were evaluated by varying the number of partitions (crawlers) to obtain merit figures for: i) download time, ii) exchange time and iii) relocation time. Evaluation has showed that our partitioning schemes outperform traditional hostname hash based counterparts in all evaluated metric, achieving on average 18% reduction for download time, 78% reduction for exchange time and 46% reduction for relocation time.

1 Introduction

The importance of Web search services is undeniable. Its effectiveness and utility strongly depends on the efficiency and coverage of the underlying crawling mechanisms.

Crawling a dynamic and evolving Web is a complex task which requires a large amount of network bandwidth and processing power, thus suggesting crawling distribution as a suitable approach to Web information retrieval.

The quality of a distributed crawling system is largely dependent on: 1) the amount of computer nodes and storage capabilities to increase computing and data processing power and 2) the existence of multiple network connection points, to increase the total communication bandwidth and the dispersion of network overload. The deployment of these systems can also take advantage of the distributed infrastructure to obtain considerable gains when downloaded pages are fed to the information retrieval modules for indexing.

The slicing of the target Web space and the assignment of each part to the crawlers is a very well known partition problem, defined as it follows:

For a given number of C distributed crawlers, residing in a pre-defined Web graph Internet topology, compute the C partitions that minimize page download time and crawler page exchange information, maintaining balance as possible.

Several different graphs may be obtained from the Web by using Internet communication metrics, the number of links between pages and other relevant data. When considering a significant period of time, graphs may evolve in conjunction with changes in: the amount of Web sites, the number of pages and links to pages and the Internet communication infrastructure, thus imposing a periodic update of the older computed partitions.

The information generated by the partitioning process may be spread among crawlers and used for routing the URLs. The routing process combines an IP aggregation mechanism similar to the Classless Inter-Domain Routing (CIDR), allowing a considerable reduction on the routing table's size.

To estimate download and exchange times metrics both average Round Trip Time (RTT) and bandwidth between each crawler and servers need to be computed. The available bandwidth for each crawler is bounded in each of the measurements of the RTT derived from real measures. The total crawler download time is a function of the number of pages available at each server and the RTT between crawlers and servers. The number of existent links between servers is used to estimate the value for URLs exchange time, between two crawlers in different partitions connected by Web links.

Presently, our results have been only compared with the traditional hostname based hash assignment. Comparison with other authors' approaches is envisaged to future work.

2 Related work

A taxonomy for crawl distribution has been presented by Cho and Molina [1]. They also proposed several partitioning strategies, evaluated using a set of defined metrics. The work included guidelines on the implementation of parallel crawler's architecture, using hostname and URL hashing as the base partitioning technique. In what follows, we will use a hostname hash based partitioning scheme to compare it with our own work results, discarding URL hash based distribution schemes, because it generates a large number of inter-partition links.

In [2] a hostname based hash assignment function is used as well, focusing a consistent hash mechanism.

The work in [3] proposes a redirection mechanism to allow a crawler to overcome balancing problems using a URL hash function after the hostname hash function is applied.

In [1] coordination is classified as: 1) independent, 2) static assignment and 3) dynamic assignment. Following this approach, the current proposal adopt an hybrid coordination strategy that uses both static and dynamic assignment. To accommodate Web evolution, dynamic assignment is reflected on the information collected in previous crawls where a central coordinator decides the initial partitions. Static assignment is reflected between partitioning stages, where each crawler decides on its own where to send the links found.

Another closely related research [4] presented a partitioning mechanism based on a hierarchical structure derived from the URL components. Coordination uses IBM TSpaces to provide a global communication infrastructure. In our work, Web space partitioning results from the application of a specific partitioning algorithm.

In a previous work [5], we evaluated a scalable distributed crawling supported by the Web link structure and a geographical partitioning.

3 Partitioning strategies

Traditional hostname hash based partitioning strategies are considered good partitioning schemes, because they allow a crawler to extract all the URLs belonging to the same Web site thus reducing inter-partition communication. This scheme although robust, mainly because it allows a light scalable and decentralized URL routing mechanism, it does not take into account the real network and link infrastructures, thus reducing the chances to crawling optimizations.

Distributed crawling is aimed to: i) reduce Web page download times, ii) minimize the amount of exchanged information between partitions (crawlers); and iii) balance Web space load among crawlers. In our approach we create a simplified model of the Internet communication infrastructure and of the Web topology based on the relevant data collected by a first crawling run. Because there is no routing information available for the first URLs, a hash assignment mechanism is used to reach a first partitioning stage.

Subsequent partitioning stages are based on the previous graph configurations and partitions, thus preserving old graph values and updating only the new data, to be able reduce the time to calculate the new partitions. In this paper, the process of graph evolution is not described.

To optimize Web page download time, we use the communication distance between Web servers and crawler clients, which is computed as the Round Trip Times (RTT) between crawlers and servers, obtained by using a *traceroute* tool. The parsing the downloaded pages allows to calculate the amount of links pointing to pages allocated to other partitions, which will be used to compute the exchange time.

For each Web server, to balance the page download work assigned to each crawler, we also take into account the number of pages per server. Next, two separate graph representations are created using RTT and Web link data.

The partitioning of multiple graphs, whose vertices are shared, may be viewed as a single multi-objective graph with multi-edge weights. In what follows we focus on the RTT and link graphs. The study and exploitation of other sort of graphs, like geographic proximity and content awareness, are also under investigation.

3.1 Multi-level partitioning

The graph partitioning problem aims to divide the vertices of a graph into a number of roughly equal parts, such that the sum of the weights of the edges is minimized between different parts. Given a graph $G = (V, E)$ where $|V| = n$, the partitioning of V into k subsets, $V_1, V_2, \ldots, V_k$, is such that $V_i \cap V_j = \emptyset, \forall_{i \neq j}$, $|V_i| = n/k$, $\bigcup_i V_i = V$, and $\min \sum_{j \in cut} w_j^{\mathrm{e}}$, where *cut* is the set of edges of E whose incident vertices belong to different subsets, and w_j^{e} is the weight of edge j. $\sum_{j \in cut} w_j^{\mathrm{e}}$ is also known as the edge-cut.

Partitioning problems are considered NP-complete, and several algorithms have been developed that find good partitions in reasonable time. Multilevel partitioning addresses the partitioning problem by successive coarsening phases, that transform the initial graph into smaller graphs.

Afterwards, a k-way partitioning algorithm is used to process the smaller graphs and produce other partitions that will be back projected (uncoarsed) to the initial graph. We use a k-way partitioning scheme adapted from the original Kernighan-Lin algorithm (KL) [6] which is similar to that described in [7]. Basically, the KL algorithm starts with an initial partition that iterates to find a set of vertices from each partition, such that moving that set to a different partition would yield a lower edge-cut. Whenever it finds a set that meets the required condition, the set is effectively moved to that partition and the algorithm restarts, until no further reduction of the edge-cut is achieved.

We started using Metis [8], which is a suitable tool for graph partitioning, however our approach imposes some particular constraints. First, we must ensure that each partition contains just one crawler. Second, the crawlers must be constrained to a single? partition. Then, moving a crawler using KL algorithm must be avoided. Finally, the initial phases of the coarsening process should benefit of the existent Web topology, to favour optimization and take advantages of the natural organization of the Web pages into Web hosts and IPs.

3.2 Multi-objective partitioning

To achieve the multi-objective partitioning, we followed the procedure suggested in [9] and named it the Multi-objective Combiner. First, each graph representing each one of the objectives is partitioned separately. Afterwards, a third graph is created by assigning weights to the edges computed using the sum of the original weights normalized by the edge-cut of the associated objective.

Each normalized weight is affected by a preferential factor, to smooth the differences of magnitude existent between each of the initial graphs. The multi-objective partitioning will result from the application of the partitioning algorithm to the third graph. In order words, if we consider k objectives and k graphs, one for each objective, $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \ldots, G_k = (V_k, E_k)$, after the partitioning of these graphs there is one edge-cut for each: $\mathrm{ec}_1, \mathrm{ec}_2, \ldots, \mathrm{ec}_k$. The combined graph $G_c = (V_c, E_c)$ where $V_c = \bigcup_{i=1}^{k} V_i$ and $E_c = \bigcup_{i=1}^{k} E_i$ and each E_c has a weight of $w_c^{\mathrm{e}} = \sum_{i=1}^{k} \frac{p_i w_i^{\mathrm{e}}}{\mathrm{ec}_i}$, where p_i is the preferential factor for each graph i. In what follows we will use constant values for p_i factors, however, preliminary experiments revealed promising results when considering p_i variation.

Two scenarios were considered for evaluation: 1) Graph vertices as Web hosts and 2) Graph vertices as IP hosts. Graph vertices as Web pages were not considered due to the additional complexity in the partitioning algorithms and to the difficulty to achieve a scalable representation of the routing tables.

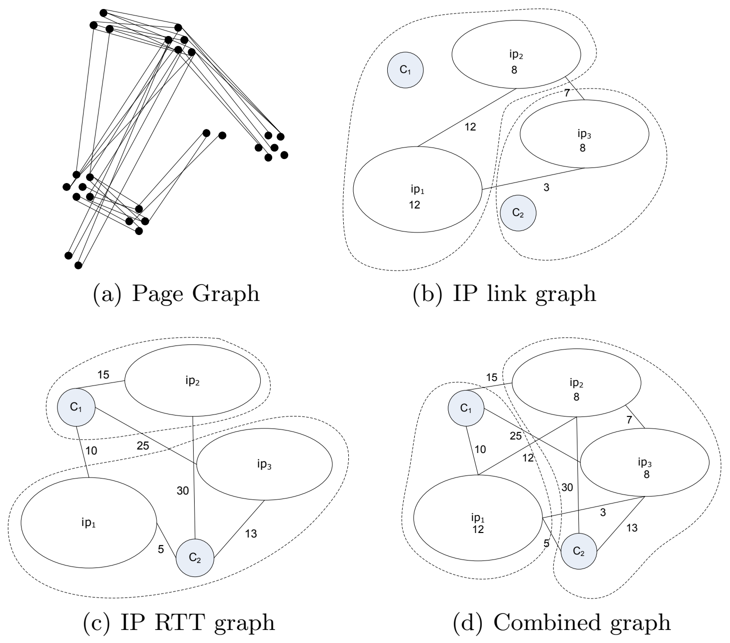


Fig. 1. Graph examples

Figure 1(a) depicts an example graph containing 28 pages (vertices) and 38 links (edges) among them. The partitioning algorithm starts by coarsening the graph into a IP graph like the one in Fig. 1(b). Hostname coarsening is not shown in the figure. In the coarser graph, pages are collapsed in the same vertex IP, resulting in a vertex weight calculated as the sum of the pages contained in the Web site at that vertex being the values of new edges computed as the sums of the link edges of the previous graph. Assuming that the depicted Web pages are hosted by some IPs and considering that there are two crawlers to download these pages, we may represent the RTT graph as in Fig. 1(c). Vertices weights in this graph are the same for all vertices. The dashed lines represent the resultant partitions.

Depending on the chosen scenario, we could have coarsened the page graph and we use the hostnames as vertices, or coarse one level deeper the hostname graph and we use IPs as vertices. In this example, we used IP graph coarsening. Each vertex has a weight which is the sum of weights of its hosts, being the edge weights the sums of the edge weights of the finer graph.

After each one of these graphs is partitioned (two partitions in this example) and their edge-cuts computed, a new combined graph is generated with their edges affected accordingly as mentioned before. The final partitions obtained from the partitioning of this combined graph are shown in Fig. 1(d). In these examples, the graphs weights do not correspond to actual RTT or link values.

4 Evaluation

To study and evaluate the proposed approach for multi-objective graph partition, we merged two independent Portuguese Web collections into a single one comprising 16,859,287 URLs (NetCensus [10] and WPT03 [11]).

Since then, the derived collection has supported the development and validation of the partitioning algorithms and produced statistics of the Portuguese Web. In particular, it is the source of the data information used to obtain the RTTs and the extracted link data, along with other sort of information related to Internet topology entities and their associated geographic entities.

As topological entities, we identified the Internet address block (Address Block), the address aggregate published by autonomous systems in BGP routing (Address Aggregate) and the autonomous system (AS). The identified geographical entities include cities and respective countries. The number of IPs for each of this entities is the following: 46,650 Hostnames, 6,394 IPs, 1,875 Address blocks, 620 Address aggregates, 363 ASs, 339 Cities and 24 Countries.

It is interesting to point out that a number of 4,627 additional IP routers, not previously included were discovered during *traceroute* operation. Another, surprising fact is that 52% of the IP servers actually reside outside of Portugal despite the fact that the evaluation refers to Portuguese Web space.

To obtain partitioning results, we used a variable number of up to 30 crawlers to process $1,903,336$ URLs, containing $26,472$ hostnames associated to a total of $1,700$ IPs. We initially constructed two different sets of graphs, based on RTT and Web link data obtained in previous experiments. The first set includes an IP graph and a hostname graph whose arcs are weighted by the computed RTT between nodes. The second set is made of two link graphs with IP and hostname nodes constructed using the method described in Sect. 3.2.

Partitioning strategies were evaluated by varying the number of partitions (crawlers) to obtain the following metrics: i) download time; ii) exchange time; and iii) relocation time. Although we believe that the maximum number of partitions used is acceptable, we plan to increase this number in future work.

The following metrics reflect a communication latency between crawler and servers, although we disregard server load, we assume this latency is far small than communication latency.

Download Time: For a set of partitions, the download time estimates the maximum time needed to download the totality of the pages included in the set. For a server i the download time needed by crawler j may be approximated by the formula: $dts_i = \frac{M_i}{L_j}(2RTT_{ij} + \frac{L_j \cdot ps_i}{BW_{ij}} + PT_i)$, where L_j is the number of pages downloaded in pipeline by `http` persistent connections between crawler j and server i; M_i is the number of pages of the server i; RTT_{ij} and BW_{ij} are the RTT and the available bandwidth between the crawler j and the server i, respectively; ps_i is the average page size of the server i. PT_i is a politeness wait time interval between consecutive connections to the same server. Note that, at each moment, a crawler can only have a single connection to a server.

Considering a total load of S_j servers for crawler j and N_j `http` simultaneous connections, the total download time for the crawler j is given by the equation: $dt_j = \frac{1}{N_j}\sum_{l=1}^{S_j} dts_l$.

The maximum total download time was defined by the time taken by the slowest crawler or heaviest partition of p partitions, given by the expression: $\max_{i=1}^{p} dt_i$.

Exchange Time: As each partition j has several links to other Web sites assigned to different partitions, the associated URLs have to be forwarded to the crawlers responsible for the associated partitions. The estimated time needed to exchange the foreign URLs is given by the equation: $et_j = \frac{1}{N_j}\sum_{l=1}^{P} 2RTT_{jl} + \frac{su \cdot nl_{jl}}{BW_{jl}}, l \neq j$, where RTT_{jl} is the RTT between crawlers l and j, nl_{jl} the total number of links from the partition j to the partition l, su is the average size of a single URL, BW_{jl} is the bandwidth between crawlers and N_j is the number of simultaneous links forwarded. The total time to process all the partitions is estimated as the maximum value over all the partitions.

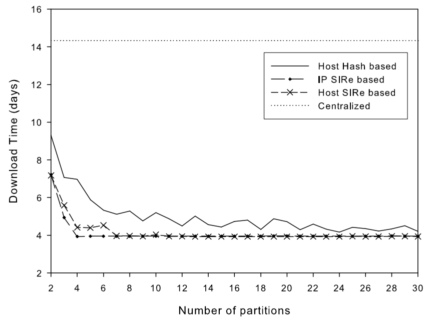
Relocation Time: Relocation time estimates the amount of time taken to move the URLs from an original partition to a destination partition, whenever the addition of a crawler imposes a reassignment of the initial partition configuration.

It is equivalent to exchange time except for nl_{jl} that is the number of relocated links from the partition assigned to crawler j to the partition assigned to crawler l in the new configuration. A maximum for all partitions is calculated.

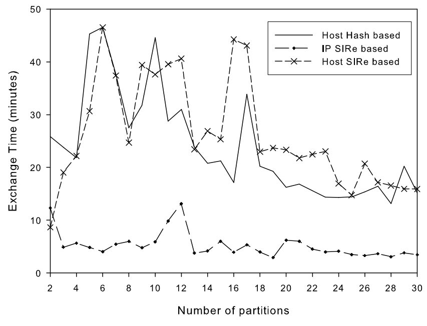
The results produced by the formulas above constitute the base for a series of experiments conduced to compare our results (which we call SIRe) and hostname hash based partitioning schemes, using the following values: $L_j = 10$, $BW_{ij} = 16$ Kbps, $ps_i = 10$ KB, $PT_i = 15$, $su = 40$ bytes and $N_j = 10$. The time values obtained may seem high because of the low bandwidth used.

Figure 2(a) shows the estimated values for download time considering four schemes: i) SIRe with hostnames as vertices, ii) SIRe with IPs as vertices, iii) hostname hash based partitioning and iv) a centralized solution.

As expected, the increase in the number of partitions diminishes download time for all partitioning methods, effectively improving the centralized approach. Both SIRe based schemes outperform the hash based partitioning method, however no improvement is achieved by hostname SIRe based in comparison with IP SIRe based partitioning. In fact, there is a tendency for equality. On average an 18% reduction is achieved using IP SIRe based over hostname hash based partitioning.



(a)



(b)

Fig. 2. Download and Exchange Time for SIRe and Hash Host based partitioning

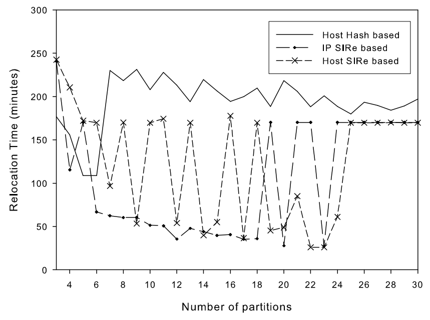


Fig. 3. Relocation Time for SIRe and Hash Host based partitioning

Figure 2(b) presents the results obtained for the estimation of the exchange time. Host SIRe based partitioning behaves similar to hostname hash based counterpart. On the other hand, IP SIRe based partitioning clearly outperform the other methods, achieving on average 78% reduction over hostname hash based partitioning.

Figure 3 presents the results for the estimated relocation time of the URLs assigned to the partitions after a new crawler is added. SIRe based schemes outperform the hostname hash based scheme for the majority of partitioning configurations. However, IP SIRe based scheme tends to achieve better results, with a reduction of 46% on average of hostname hash based partitioning.

5 Conclusions and future work

This paper presents SIRe´s (Scalable Information Retrieval environment) approach to Web partitioning, a project that seeks to optimize Web crawler download and exchange times. The approach is supported by the construction of two different weighted graphs. The first graph is used to model the topological communication infrastructure between crawlers and Web servers. The second graph allows to represent the amount of link connections between server's pages. The values of the edges represent, respectively, computed RTTs and pages links between nodes. The two graphs are further combined to support Web space partitioning and load balancing for variable number of geographical distributed crawlers by means of a multi-objective partitioning algorithm.

The proposed approach differentiates instead of using a deterministic hash function to slice the target Web space, as most of the approaches referenced in the literature, is based on the possibility of using earlier knowledge of the servers and communication paths infrastructure to build an appropriate graph representation of the Web from where to derive a partitioning scheme.

To validate our work proposal, the partitioning algorithms were adapted to the specific requirements of the Web space under study, namely, crawler separation and compulsory assignment to at least one partition, and the need to provide additional control over the coarsening mechanism.

Evaluation showed that SIRe´s based partitioning schemes outperforms hostname hash based counterparts for all the evaluated metrics. Justification for these results resides in the fact that traditional hostname hash based schemes do not consider the amount of pages in each server. SIRe´s improvement is particularly relevant in situations where there is a significant difference in the total number of Web pages contained in each of the Web servers.

Differences between Host and IP SIRe approaches are not noteworthy for download times. We claim that IP SIRe is a better overall solution, even considering that Host SIRe partitioning strategy take longer time to run, because of the power-law distribution of pages on Web servers. In fact, a lot of servers with a small number of pages and few servers with a large number of pages, impose a maximum download time for the partition that hold heavy page load servers.

Host SIRe based partitioning exchange time was not able outperform Host Hash based scheme most likely due to the multi-objective preferential factor assigned to the link objective. IP SIRe based partitioning has considerably better performance because hostnames belonging to same IP have a larger number of links between them, thus decreasing the exchange time.

SIRe based partitioning relocation times are better than Host hash based scheme, although very unstable. Boundaries around 170 minutes for SIRe based schemes relocation time correspond to the relocation of heavy page load servers. In future work we will take into account this issue.

For all the 30 configurations considered, IP SIRe based partitioning strategy achieved a considerable time reduction, when compared to the hostname hash based scheme.

SIRe partitioning schemes seek to optimize multiple objectives which are optimized simultaneously, thus both download times and exchange times are counterbalanced. Changing the preferential factors for graph weight combination would certainly bias the results toward improved download times and worst exchange times, and vice versa. In the future we plan to vary preferential factors to check for better partitioning results.

Currently we are working to include other optimization objectives including i) content similarity between Web pages and ii) download optimization, taking into account the page size and frequency of change, when crawlers are working in incremental mode. The two graph edge weights used so far, might be extended with some content similarity between pages, thus allowing content aware partitioning to enable the creation of focused partitions; possible exchange overhead reduction may also be achieved based on page links affinity due to page similarity. Also, in incremental crawling mode, pages are visited several times based on its change frequency. Frequently visited pages induce an additional load to the crawler responsible for those pages, causing load unbalance among crawlers. The addition to the graph representation of the page change frequency estimation may result in the improvement of load balancing.

Finally, we think that graph evolution with real time updates and partitioning algorithm distribution is also a hot topic for further work.

References

1. Cho, J., Garcia-Molina, H.: Parallel crawlers. In: Proc. of the 11th International World–Wide Web Conference. (2002)
2. Boldi, P., Codenotti, B., Santini, M., Vigna, S.: Ubicrawler: A scalable fully distributed web crawler. Software: Practice & Experience **34**(8) (2002) 711–726
3. Loo, B., Krishnamurthy, S., Cooper, O.: Distributed Web Crawling over DHTs. Technical report, EECS Department, University of California, Berkeley (2004)
4. Teng, S.H., Lu, Q., Eichstaedt, M., Ford, D., Lehman, T.: Collaborative Web Crawling: Information Gathering/Processing over Internet. In: Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences - Volume 5. (1999)
5. Exposto, J., Macedo, J., Pina, A., Alves, A., Rufino, J.: Geographical Partition for Distributed Web Crawling. In: 2nd International ACM Workshop on Geographic Information Retrieval (GIR 2005), Bremen, Germany, ACM Press (2005) 55–60
6. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Sys. Tech. J. **49**(2) (1970) 291–308
7. Fiduccia, C.M., Mattheyses, R.M.: A linear-time heuristic for improving network partitions. In: DAC '82: Proceedings of the 19th conference on Design automation, Piscataway, NJ, USA, IEEE Press (1982) 175–181
8. Karypis, G., Kumar, V.: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices – Version 4.0. Technical report, University of Minnesota, Department of Computer Science / Army HPC Research Center (1998)
9. Schloegel, K., Karypis, G., Kumar, V.: A New Algorithm for Multi-objective Graph Partitioning. In: European Conference on Parallel Processing. (1999) 322–331
10. Macedo, J., Pina, A., Azevedo, P., Belo, O., Santos, M., Almeida, J.J., Silva, L.: NetCensus Project. http://marco.uminho.pt/∼macedo/netcensus/ (2001)
11. XLDB Group: WPT03. Linguateca, http://www.linguateca.pt (2003)