About Crawling Scheduling Problems

Andrey A. Pechnikov[1] Denis I. Chernobrovkin[2] Anthony M. Nwohiri[3]

[1] Institute of Applied Mathematical Research of Karelian Research Centre of the Russian Academy of Sciences, Petrozavodsk, Republic of Karelia, Russia
[2] DataArt Inc, St. Petersburg, Russia
[3] Department of Computer Science, Faculty of Science, University of Lagos, Nigeria
anthony.nwohiri@outlook.com

Abstract

This paper investigates the task of scheduling jobs across several servers in a software system similar to the Enterprise Desktop Grid. One of the features of this system is that it has a specific area of action – collects outbound hyperlinks for a given set of websites. The target set is scanned continuously (and regularly) at certain time intervals. Data obtained from previous scans are used to construct the next scanning task for the purpose of enhancing efficiency (shortening the scanning time in this case). A mathematical model for minimizing the scanning time for a batch mode is constructed; an approximate algorithm for the solution of the model is proposed. A series of experiments are carried out in a real software system. The results obtained from the experiments enabled to compare the proposed batch mode with the known round robin mode. This revealed the advantages and disadvantages of the batch mode.

Keywords: Enterprise Desktop Grid, Web Crawling, Round-Robin Scheduling, Batch Job Scheduling.

1 Introduction

Construction and study of web fragments are classified under such web science field as link analysis [The04]. Web crawlers (sometimes called web spiders, internet bots or search robots) play an essential role here. These are specialized programs that surf through web pages in a systematic manner to retrieve required information. In general, crawling can be considered a multi-purpose search problem with some constraints. The task becomes a very challenging one thanks to the large variety of objective functions, together with lack of relevant knowledge about where to search for [PSM04]. The BeeBot crawler is described in [PC14]. This robot traverses the web and collects information about outbound (external) hyperlinks from a set of websites defined by direct enumeration. The crawler is implemented in C# language for .NET Framework Version 4.0, using Microsoft SQL Server 2008 R2 to run on a standard personal computer running Windows XP and higher. The features of the BeeBot crawler can be enhanced, making it both parallel and distributed in nature [KK12]. This can be achieved using an approach similar to that used in Enterprise Desktop Grid systems. Round-robin is one of the most famous algorithms employed by process and network schedulers in computing for such. In the algorithm, a load is assigned to a computing system by enumerating and ordering its elements in a circular order [Kle64]. In our problem statement, we proceed from the following:

In: E. Ivashko, A. Rumyantsev (eds.): Proceedings of the Third International Conference BOINC:FAST 2017, Petrozavodsk, Russia, August 28 - September 01, 2017, published at http://ceur-ws.org

- That the set of scanned sites is large enough and is constantly (and regularly) scanned at certain intervals
- That there are some computers that are integrated into Enterprise Desktop Grid
- That the cost of scanning each site in each computer is known (or can be calculated).

Hence, the task involves formation of batch jobs (for each server) that satisfy a certain optimality criterion for the system as a whole. This paper proposes an algorithm that can create batch jobs to minimize the running time of the system. The algorithm is based on solving the minimax optimization problem. A series of experiments performed showed that the proposed algorithm is faster than the round-robin algorithm.

2 Parallel distributed crawler: BeeBot-EDG

The name BeeBot-EDG comes from a combination of the crawler BeeBot and abbreviation EDG (Enterprise Desktop Grid). Parallel distributed crawler BeeBot-EDG was developed based on the principles behind the well-known and popular grid computing platform BOINC [a6]. Accordingly, BeeBot-EDG and BOINC have similar management and operation principles. BeeBot-EDG has client-server architecture. The two components (server and client) will be discussed in more detail. Briefly, the general mode of operations can be represented as follows: client crawlers are launched by users on the working machines in an Enterprise Desktop Grid and connected to the master server. Then, the server completely manages the scanning process – clients execute only the batch jobs sent to them by the server, and send received data to the server. Microsofts .NET Framework component stack (.Net Framework 4.6.1 and C# 6.0 programming language) was used as the software framework. Graphical subsystem WPF (Windows Presentation Foundation), architectural pattern MVVM (Model-View-ViewModel) and library MVVM Light Toolkit were used to render the graphical user interface (GUI). Microsoft SQL Server Express was used to store data obtained. Object-relational mapping Entity Framework was used for easy access to the database. Duplex service WCF (Windows Communication Foundation) and TCP/IP protocol (as a lower-level alternative to the currently popular data transfer via high-level HTTP protocol) were used for client-server data exchange. The HTMLAgilityPack library was used to provide convenient syntactic parsing of information contained in downloaded web pages.

3 Measuring client performance and site size: challenges

Constructing the mathematical model of a computing process in Beebot-EDG requires identifying the controlled and uncontrolled parameters of the model [Ack78]. In this section, we discuss the uncontrolled parameters and the challenges inherent in their measurement. Running the BeeBot program on a client computer requires various system resources. The following is the list of uncontrolled system performance parameters:

- Response time of the Beebot-EDG server,
- Time it takes the batch job to load from the server to the client,
- Time it takes to transfer the results file from the client to the server,
- DNS server response time,
- TCP connection time,
- Web server response time,
- Web page download time,
- Client computer speed.

Due to the above parameters, the client's performance is difficult to evaluate by any single integral parameter. Computing performance, network speed, and even geographical location of the computer are all important. What matters is to know which applications (and how many) are executed by the client's computer when BeeBot is launched. Obviously, measuring the performance of Beebot-EDGs client computers for each parameter will require additional system resources, which would be better re- channeled to the scanning process. In fact, in this case, what obtains is a performance testing problem for a software and hardware complex [KFNH99], which is clearly outside the scope of this study. One of the possible solutions to this situation is an approach we call use of a reference site. The reference site must meet a number of requirements, namely:

- That the website should be hosted on a dedicated server that is solely hosting this website and does not receive any other loads from third-party applications while scanning the reference site;
- That the site should be geographically located in the same region as the set of sites being investigated;
- That the site should have constant number of pages for a certain scanning period of time;
- That the site structure should be similar to those of the investigated sites;
- That the number of web pages should be close to the average value for the investigated set;
- That the content of the site should be similar by nature to the contents of the investigated sites.

Next, the reference site is scanned with client crawlers. The scanning time is recorded. The values obtained (relative and not absolute values) are relevant here – they can be used in algorithms for solving optimization problems. During the crawlers scanning process, the following characteristics and parameters of the scanned site are of great importance – domain name, site size (number of pages) and data quantity (number of external hyperlinks) collected via scanning. It is obvious that the size of a site and number of external hyperlinks collected from the site can be found only by scanning that site. We proceed from the fact that the investigated set of sites is constantly (and regularly) scanned at certain time intervals. Therefore, by scanning a given set of sites in a round-robin mode once and recording the data (number of webpages and external hyperlink count for each site) obtained, the values received can be used in the next scanning. Note that the obvious dynamics of the web leads to the fact that at the next scanning, current values can significantly differ from previous values. In this case, it is possible that the results obtained from multiple consecutive site scans can solve the problem of dynamic series for such parameters as site size and external hyperlink count for each site [Ken76].

4 Batch job creation

In this study, job means a request to one of the clients to scan a given website, indicating the main scanning parameters. Let $Z = 1, 2, \ldots, n$ be a set of all jobs. The following are descriptions of the uncontrolled parameters of job $i \in Z$:

1. $dname_i$ – Domain name of the website (a character string without indication of the protocol and ports);
2. P_i – Site size;
3. h_i – Size of file containing outbound hyperlinks that are pointing from site i . This file is obtained after scanning the site and is returned to the server by the client.

Kilobyte (KB) is the unit of measurement for the second and third parameters. Let job subset $U \subseteq Z$ be a batch job (or simply a batch). The size of all the websites in a batch is equal to the sum of the sizes of the websites of jobs included in a batch – that is $\sum_{z \in U} P_z$, where z is a job included in batch U.

Let $C = 1, 2, \ldots, m$ be a set of clients. We introduce the following performance parameters for client $j \in C$:
tos_j – Response time of the Beebot-EDG server for the j -th client,

vsc_j – Speed it takes the batch job to download from the Beebot-EDG server to the j-th client and back,

$tdns_j$ – Response time of the DNS server for the j-th client, $ttcp_j^z$ – Time of establishing TCP connection between the j-th client and the web server containing the site contained in job z,

$tweb_j^z$ – Response time of the web server containing the site contained in job z for the j-th client,

$vweb_j^z$ – The speed it takes the content of the site contained in job z to download from the web server to the j-th client,

v_j – Computer speed of the j-th client.

We take KB/sec (kilobytes per second) as the unit of measurement for speed and response time, and sec (second) as the unit of measurement for time. Let U_j be a batch sent by the server to the j-th client for execution. So the time spent by the j-th client to execute batch U_j consists of the following three time components:

- Time taken to receive batch jobs
$$t_j^{in} = tos_j + q_j/vsc_j$$
(q_j is the size (in KB) of the file containing the list of domain names of sites that make up batch U_j),

- Time taken to process the batch job
$$t_j^{work} = \sum_{z \in U_j} (tdns_j + ttcp_j^z + tweb_j^z + P_z/vweb_j^z + P_z/v_j),$$

- Time taken to send results obtained from the processing of batch U_j to the server
$$t^{out} = (\sum_{z \in U_j} h_z)/vsc_j.$$

So, the total amount of time spent by client j to execute batch U_j is equal to $T_j = t_j^{in} + t^{work} + t_j^{out}$. To generalize the problem statement, we will assume that if the server does not send a batch to client j, it means that the client receives empty batch $U_j = \varnothing$. In the general case, we are dealing with the partitioning of set Z into a set of disjoint subsets $U = U_1, U_2, \dots, U_m$, such that,

$$\forall i, j \in C : U_i \cap U_j = \varnothing, \tag{1}$$
$$U_1 \cup U_2 \cup \dots \cup U_m = Z. \tag{2}$$

Let U^m be a set of all subsets of the form $U = U_1, U_2, \dots, U_m$ satisfying (1) and (2).

The essence of the problem statement is that we want to partition the set such that the minimum execution time is achieved by the client with maximum execution time. This can be formally expressed as follows:

$$\max\{T_1, T_2, \dots, T_m\} \rightarrow_{U \in U^m} \min. \tag{3}$$

Optimization problem (1)–(3) belongs to a large class of set partitioning problems. As a rule, obtaining an optimal solution in such cases is possible only by complete enumeration. In our case, evaluation of the complexity of a complete enumeration algorithm is of the order of magnitude $O(m^n)$.

5 Reduced problem and heuristic algorithm for its solution

We will further apply a combination of both approaches.

We will assume that all client computers are approximately on equal footing with respect to the used external servers and communication channels. Assuming also that file sizes q_j are approximately equal, we can assume that t_j^{in} are equal for $\forall j \in C$. For the same reasons, we will also assume that t_j^{out} are equal for $\forall j \in C$.

By removing the first and last terms from all $T_1, T_2, \dots, T_m$, we obtain the expression:

$$\sum_{z \in U_j} (tdns_j + ttcp_j^z + tweb_j^z + P_z/vweb_j^z + P_z/v_j)$$

From the same considerations that all client computers are approximately on equal footing with respect to used external servers and communication channels, we assume that for $\forall j \in C$, $tdns_j = tdns$ and for $\forall z \in U_j$, $ttcp_j^z = ttcp$ and $tweb^z = tweb$. Besides, for $\forall z \in U_j$, $vweb_j^z = vweb_j$. After some transformations, we get:

$$\sum_{z \in U_j} (tdns + ttcp + tweb + P_z/vweb_j + P_z/v_j) = ||U_j|| \times (tdns + ttcp + tweb) + \sum_{z \in U_j} P_z \frac{1}{1/vweb_j + 1/v_j},$$

where $||U_j||$ is the power of subset U_j.

Measurements conducted on a set of research and educational sites show that the response time $tdns + ttcp + tweb$ is 0.2-0.3 sec. Measurements of the client computer performance for a site containing 1000 pages show the time taken to process one site in a 200-300 sec interval. Of the two terms to the right of the equality sign, the contribution of the second term is three orders of magnitude larger than that of the first. So we can approximately leave it for reduced problem:

$$T_j^R = \sum_{z \in U_j} P_z \frac{1}{1/vweb_j + 1/v_j},$$

The objective function of the reduced optimization problem looks like this

$$\max\{T_1^R, T_2^R, \dots, T_m^R\} \to_{U \in U^m} \min \tag{4}$$

with constraints (1)-(2).

The heuristic algorithm for solving problem (1,2,4) is based on the well-known property of thematic web spaces (university, research, government and other web spaces), which is that the distribution of site sizes is very close to discrete power-law distribution [CSN09]. A brief description of the algorithm is as follows:

Step 1. Sizes of the websites of the scanned set and the performance of client computers are arranged in descending order;

Step 2. The total amount of sizes of the scanned sites is calculated;

Step 3. The share of the amount of site sizes for each computer is calculated proportionally to the computer performance;

Step 4. For each client computer (in descending order of performance), a batch job is generated (in descending order of sizes) by retrieving the next site from the list (set) of scanned sites and adding it to the batch until the total amount of sizes of the sites in the batch becomes greater than the value calculated in step 3. Then the last site of the batch is returned to the list of scanned sites;

Step 5. All the generated batch jobs are supplemented (in the reverse order) with sites remaining in the scanned set after step 4 has been executed – from the smallest to the largest.

The scanning of target set sites in accordance with the batch jobs distributed among several clients shall be regarded as a batch job work.

6 Natural experiments conducted: description and results

As the founding fathers of cybernetics said "... the best material model for a cat is another, or preferably the same cat. In other words, should a material model thoroughly realize its purpose, the original situation could be grasped in its entirety and a model would be unnecessary" [RW45, p. 320].

In our case, we have a good example of when experiments can be conducted on a real object – in the BeeBot-EDG system. The general description of the experiment is summarized as follows:

1. A set of client computers is created;
2. The performance of client computers on a reference site is measured;
3. A target set of scanned sites is created;
4. The target set of scanned sites is scanned in a round-robin mode. The time spent on scanning and the site sizes are recorded;
5. The proposed heuristic algorithm is used to solve the batch job creation problem, where the data obtained from measurement of performance (step 2) and site size (step 4) are taken as the initial data;
6. The target set of scanned sites is scanned in a batch job mode. The scanning time is measured.

Out of the large series of experiments carried out, we will, as an example, describe the outcome of eight of them conducted on one target set using three and five client computers. The numbers of clientcomputers were chosen based on the resources available to the authors. It seems that the results obtained graphically characterize the conclusions drawn below.

Experiment	Scanning time of BeeBot-EDG (hour:min:sec)
Round robin1 99x3	3:30:01
Round robin2 99x3	6:25:03
Batch job1 99x3	3:42:39
Batch job2 99x3	3:23:59
Round robin1 99x5	2:48:10
Round robin2 99x5	4:18:44
Batch job1 99x5	3:25:51
Batch job2 99x5	2:47:51

Table 1: Empirical data

The time taken by the five client computers to scan a “reference” website featuring 800 web pages is 95.0, 84.6, 105.7, 110.4 and 85.8 seconds, respectively. For the three client computers, the scanning time is 105.7, 110.4 and 85.8 seconds. The widest difference in performance speed between the client computers reaches 25%.

The target set is a sample of 99 websites obtained from a list containing over 400 websites of official institutions of the Russian Academy of Sciences (Webometric Ranking of Russian Scientific Institutions, http://webometrics-net.ru). The websites were scanned up to the third level. Results of the experiments are presented in the Table 1.

The *Round robin1 99x3* experiment was conducted in a round robin mode for 99 websites and on three client computers.

Round robin2 99x3 experiment: round robin mode, 99 websites, three client computers. But, unlike *Round robin1*, the websites were scanned in the reverse order.

Batch job1 99x3 experiment: 99 websites, three client computers. It used a batch job mode based on partitioning into batches as described in the “Reduced Problem and Heuristic Algorithm for its Solution” section.

Batch job2 99x3 experiment: 99 sites, three clients. It used a batch job mode based on partitioning into batches as described in the “Reduced Problem and Heuristic Algorithm for its Solution” section. Here, instead of site sizes, the average scanning time obtained in previous experiments was taken as the measure of complexity.

The *Round robin1 99x5, Round robin2 99x5, Batch job1 99x5* and *Batch job2 99x5* experiments were carried out similarly but for five client computers.

The main conclusion from the experiments is that if the scanning data obtained from full scanning of the target set in a (any) mode are taken into account, we will be able to build a new scanning process that is more efficient than the previous processes. Let us confirm this through examples from the table.

If we assume that the “previous” scanning was performed in *Round robin2*, then with the site sizes (number of pages) obtained, we can create a list for “subsequent” scanning, which would ensure that ”large” sites appear in the beginning of the list, and “small” sites at the end. Here, *Round robin1* becomes a more efficient variant.

The *Batch job1* variant that uses number of pages as measure of complexity turned out to be more efficient than *Round robin2*, but less efficient than *Round robin1.* This result confirms the main conclusion that if we take data from previous scans into account, we can construct a job that is more efficient than the worst case scenario. It also supports the fact that evaluation of scanning complexity based on number of webpages on the site is a rough and inaccurate evaluation.

The use of average scanning time instead of number of webpages in the *Batch job2* variant yields the fastest time in the BeeBot-EDG system when scanning 99 websites in three client computers.

The scanning results for the five client computers almost confirm the conclusions made for the three client computers.

7 Conclusion

This paper investigated the scheduling of jobs across several servers in software system BeeBot-EDG. The purpose was to collect outbound hyperlinks for a given set of websites. Since the target set isscanned repeatedly after a certain time interval, data obtained from previous scans can be used to construct a job for the next scan in order to improve on efficiency.

Two known job scheduling modes – round robin and batch job – were compared. A series of experiments, implemented on a real software system, allowed to compare these modes and reveal certain advantages of the batch job mode over the round robin mode.

At the same time, experiments conducted showed that there is need for further research in this direction in order to develop more efficient job scheduling algorithms.

References

[a6] Open-source software for volunteer computing. https://boinc.berkeley.edu (date accessed: 17 February 2017).

[Ack78] R. Ackoff. *The Art of Problem Solving.* John Wiley and Sons, Ltd., 1978.

[CSN09] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[Ken76] M. Kendall. *Time-Series.* Charles Griffin&Co. Ltd., 2nd edition edition, 1976.

[KFNH99] C. Kaner, J. Falk, and Q. Nguen H. *Testing Computer Software.* John Wiley and Sons, Ltd., 1999.

[KK12] D. Khurana and S. Kumar. Web crawler: A review. *International Journal of Computer Science & Management Studies*, 12(1):401–405, 2012.

[Kle64] Leonard Kleinrock. Analysis of a time-shared processor. *Naval Research Logistics Quarterly*, 11(1):59–73, 1964.

[PC14] A. A. Pechnikov and D. I. Chernobrovkin. Adaptive crawler for external hyperlinks search and acquisition. *Automation and Remote Control*, 75(3):587–593, Mar 2014.

[PSM04] Gautam Pant, Padmini Srinivasan, and Filippo Menczer. *Crawling the Web*, pages 153–177. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[RW45] Arturo Rosenblueth and Norbert Wiener. The role of models in science. *Philosophy of Science*, 12(4):316–321, 1945.

[The04] M. Thelwall. *Link Analysis: An Information Science Approach.* Elsevier Academic Press, Amsterdam, 2004.