Algorithms for Selecting Multiple Mirror Sites for Parallel Download

Yu Cai
Michigan Technological University
Houghton, MI 49931, USA

C. Edward Chow
University of Colorado at Colorado Springs
Colorado Springs, CO 80933, USA

Summary

In this paper, we present a mathematical model that describes the problem of parallel download from multiple mirror sites. Based on the model, we present algorithms for selecting the best subset of mirror sites for parallel download. The versions of brutal force algorithms and genetic algorithms are implemented. Performance of these algorithms on the simulated network topology as well as a real-world network topology is presented.

Key words:
Server selection, Parallel download, Genetic algorithms.

1. Introduction

With the recent development of internet, we are able to retrieve documents from multiple server sites, like the mirror sites, to increase the downloading speed, make better use of available network bandwidth and parallel processing speed of servers.

Recent work by Rodriguez, Kirpal, and Biersack [[1] studied how to use the existing HTTP protocol for retrieving documents from mirror sites in parallel to reduce the download time and to improve the reliability. The proposed approach utilizes the HTTP 1.1 byte range header to retrieve specific data in a mirror server site, which requires no changes on existing server and client settings.

However, choosing the best mirror sites is not a trivial task and a bad choice will give a poor performance. Testing data [[1, 2] shows that the performance of a bad choice might be 10 times slower than the best choice.

The document delivery speed between a server and a client depends on the server load, the file retrieving speed at the server, the available bandwidth between the server and the client, and the client load. Given a client, the document delivery speed from a particular server can be estimated by downloading a common short document existing on all mirror servers. Application layer anycast was proposed for selecting one of the replicated web server [9].

By using networking measurement tools, like pathchar, cprobe, we can estimate the network bottleneck and available bandwidth [3]. Kevin Lai and Mary Baker of Stanford University improve accuracy of the bottleneck bandwidth estimation by using better filtering technique in dynamic environment [4]. However, the accuracy of current network measurement methods still needed to be improved.

In this paper, we investigate two problems: one deals with finding the maximum parallel download speed without restricting the number of mirror servers. We call it pds problem. The other deals with finding the best group of k servers for parallel download. We called it k-pds problem. It is assumed that the network topology, the path bandwidth and server performance are known and static. Two versions of brutal force algorithm, as well as two versions of genetic algorithm, were implemented.

GT-ITM (Georgia Tech Internetwork Topology Models), which is one of the most commonly used internet topology models [6], is used to generate network topologies of varying sizes for evaluating their impact on the performance of the algorithms.

This paper is structured as follows. Section 2 defines the problems. Section 3 presents the algorithms for selecting multiple mirror servers for parallel download. Section 4 discusses the performance results. Section 5 is the conclusion.

2. Selecting Multiple Mirror Sites for Parallel Download

Assume that the network topology, the path bandwidth and server performance are static, the path between two end nodes will be fixed. The routing path map between a client and a set of mirror servers can then

Manuscript received October 5, 2007
Manuscript revised October 20, 2007

be modeled as a tree. We also assume that the documents to be retrieved are available on all mirror sites.

2.1. Network Model

Let G=(V, E) be a graph models the network topology, where V represents the set of nodes including those of the mirror server nodes, the clients, and the routers between the clients and the mirror server nodes; E represents the set of edges or link segments that connect the nodes in the network. Let M be the set of mirror servers. For a mirror server m, speed(m) represents the document retrieval and transfer speed of m. For an edge e, speed(e) represents the available bandwidth on edge e.

Given a client c and a set of mirror server M in G, there exists a set of shortest paths, P, from each of the mirror servers to the client c. Let path(c,m) be the path from mirror server m to the client c. Each path p in P, consists of a set of edges, $e_1,\ldots,e_n$. where n is the number of edges, or link segments in p. Without other traffic, the transfer speed over a path p in P, speed(p), is decided by the link segment with the smallest available bandwidth:

Speed(p)=min{speed(e_1),…,speed(e_n)}.

Without other traffic, the end-to-end download speed for selecting a mirror server, m, can be defined as endSpeed(c, m) and

endSpeed(c,m)=min{speed(m),speed(path(c,m))}.

Let S be the subset of mirror servers selected for parallel download and n be the number of mirror servers in S. S={S_1, S_2,…,S_n}. Let pds(c, S) be the parallel download speed of using mirror server set S to client c. Note that if all paths from the mirror server in S to client c do not share any link segment. Then

$$\text{pds(c, S)}=\sum_{i=1}^{n} endSpeed(c, S_i).$$

Unfortunately, some of these mirror servers may share some link segments to the client c. The actual download speed from a set of mirror servers may then be limited by the available bandwidth of the shared link segments.

Note that the set of link segments of path set P forms a tree. See Figure 1. The leaf nodes of the tree, S_1, S_2, S_3, and S_4, are mirror server nodes with 30, 25, 20, 9 as their document retrieval and transfer speed. R_1, R_2, and R_3 are the router nodes and root node of the tree is the client c. The edge label represents the available bandwidth of the edge. For the non-leaf node r, let mds(r, S) be the maximum download speed at node r using mirror server set S.

$$\text{mds(r,S)}=\sum_{i=1}^{ntree(r)} \min(speed(e_i), mds(r_i, S)) \quad (1)$$

where ntree(r) is the number of subtrees underneath r, and e_i is the edge that connects r to the subtree r_i. It is a recursive function. mds(m,S)=speed(m) if m is a mirror server node in S. mds(r, S)=pds(c, S) if r is the client node.

2.2. Problems to be solved:

There are several problems related to parallel download from a set of mirror servers. The first problem is the pds problem: given a network topology G, a client c, a set of mirror servers S, find the maximum download speed pds(c, S). The related question is how many mirror sites are needed to achieve the global max speed and how to choose the mirror sites? We can solve these problems with formula (1) above.

We have implemented a parallel download program similar to that in reference [1], and observed that the performance peaks typically at 5 or 6 Redhat mirror servers for a Linux machine at UCCS. Part of the reason is the re-assembly overhead. Therefore if we only want to choose a certain number of mirror sites, say 5 sites, what is the maximum download speed for 5 mirror sites? And a related question is which 5 sites to choose for achieving this speed? This is referred to as k-pds problem.

Let k-pds(c,S) be the maximum parallel download speed for the client c to use k of the mirror server set S, and k-pdsSubset(c, S) be the optimal subset of k servers with the maximum parallel download speed.

$$\begin{cases} \text{k-pds(c,S)}=\max\{\text{mds(c, S')}|\text{S'} \in \text{S, S' has k nodes}\} \\ \text{mds(c, S')}=\sum_{i=1}^{ntree(r)} \min(speed(e_i), mds(r_i, S')) \end{cases} \quad (2)$$

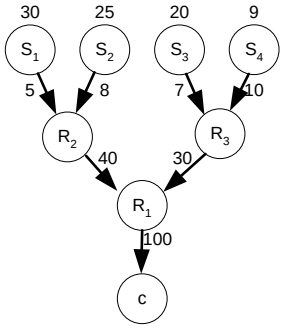
In Figure 1, we give an example using formula (1) and (2).

mds(R2,S)=min(mds(S1,S),5)+min(mds(S2,S),8)=min(30,5)+min(25,8)=13, similarly, mds(R3, S)=16,

mds(R1,S)=min(mds(R2,S),40)+min(mds(R3,S),30)=min(13,40)+min(16,30)=29,

pds(c, S)=mds(c, S)=mds(R1, S),

3-pds(c,S)=max{mds(c,{S1,S2,S3}),mds(c,{S1,S2,S4}),mds(c,{S1,S3,S4}),mds(c,{S2,S3,S4})}=max{20,22,21,24}=24, and the subset of mirror servers to use 3-pdsSubset(c,S)={S_2,S_3,S_4}. Similarly, 2-pds(c,S)=17 and 2-pdsSubset(c, S)={S_2,S_4}.



$S=\{S_1,S_2,S_3,S_4\}$
$mds(R_2,S)=13$
$mds(R_3,S)=16$
$mds(R_1,S)=29$
$pds(c,S)=mds(c,S)=29$
$3\text{-}pds(c,S)=3\text{-}pds(c,\{S_2,S_3,S_4\})=24$
$2\text{-}pds(c,S)=2\text{-}pds(c,\{S_2,S_4\})=17$
$1\text{-}pds(c,S)=1\text{-}pds(c,\{S_4\})=9$

Figure 1. Parallel Download Tree and related download speed values.

3. Algorithm Implementation

Versions of brutal force algorithms and genetic algorithms were developed to solve the problem.

a) Brutal Force Algorithm for pds:

We use the algorithm to find the maximum parallel download speed pds(c, S). We refer to it as BF-pds algorithm. It implements formula (1) in Section 2.1. The complexity of this algorithm in worst case scenario is O(n), where n is the number of nodes in the network topology.

This algorithm is quick, but we have no control over how many mirror sites and which mirror sites to be chosen. In practice, we use it to find the upper bound of the maximum parallel download speed in network topology.

b) Brutal Force Algorithm for k-pds:

We use the algorithm to find the maximum download speed for the client c to use k of the mirror server set S, which is k-pds(c, S). We refer to it as BF-k-pds algorithm. It implements formula (2) in Section 2.2. The complexity of this algorithm in worse case scenario is $O(n^k)$, where n is the number of nodes, and k is the number of servers we want to choose.

c) Genetic Algorithm

We implement a fix-length genetic algorithm and a variable-length genetic algorithm.

The fix-length algorithm is used to find the k-pds(c,S) speed, the length of chromosomes is k and fixed. We refer to it as GA-k-pds algorithm.

The variable-length algorithm is used to find the pds(c, S), the length of chromosomes is smaller than a given number, and can be changed. We refer to it as GA-pds algorithm.

Formula (1) and (2) in Section 2 are recursive functions, therefore it is difficult to implement them in genetic algorithm. We used a revised version in genetic algorithm by scanning through all the server nodes to client node.

The variable-length genetic algorithm works as follow:

1) Assign the sequential server number, node number and path number to denote each server, node and path. Assign the initial bandwidth and server speed.
2) Initialize the first generation of chromosomes with random length by filling server number in chromosome.
3) Crossover and mutation at certain probability. Make sure no duplicated server in chromosome, and the length of chromosome is less than the given number. Several different crossover and mutation methods have been combined together for better performance [8].
4) Fitness function. For a given chromosome S', use the max download speed mds(c, S') as fitness function.
5) Run certain generations, and output the result.

Genetic algorithm provides more control and flexibility over the server selection. For example, if there are multiple selections of mirror servers, and all selections achieve the max download speed pds(c, S) or k-pds(c, S), how to choose the best one from all these selections? We can easily implement the selection criteria in genetic algorithm.

4. Testing Results

We tested the algorithms on simulated network topologies as well as a real-world network topology

Figure 2 is a sample routing tree with 20 nodes and 10 mirror sites [1]. Below is the testing result:

20 nodes, 10 mirror sites	
BF-pds	0.6 s
BF-k-pds	10 s
GA-k-pds	1.5 s
GA-pds	1.5 s

(s: second)

Figure 4 is a sample routing tree with 114 nodes and 11 mirror sites, starting from a machine in UCCS, to the mirror sites of Redhat [5]. Below is the testing result

114 nodes, 11 mirror sites	
BF-pds	0.7 s
BF-k-pds	2 m
GA-k-pds	2 s
GA-pds	2 s

(s: second, m: minute)

Figure 3 is a sample transit-stub hierarchical network topology derived from GT-ITM [7], we can derive routing tree structure from the network topology, by

assuming the route from node to node is always the shortest route in terms of network bandwidth. Below is the test result on GT-ITM simulated network:

	BF-pds	BF-k-pds	GA-k-pds	GA-pds
150 nodes, 20 mirror sites	0.8 s	2 m	2 s	2 s
200 n, 20 m	0.8 s	2.5 m	2 s	2 s
300 n, 30 m	0.9 s	3 m	2 s	2 s
500 n, 50 m	0.9s	7 m	5 s	5 s
800 n, 100 m	1 s	12 m	6 s	6 s
1000 n, 100 m	1 s	20 m	7 s	7 s
1000 n, 200 m	1 s	30 m	8 s	8 s

(s: second, m: minute)

Figure 5 is a chart of algorithm execution time vs. simulated network size. BF-k-pds algorithm execution time uses primary y axis (on the left), and is measured by minutes. BF-pds, GA-pds and GA-k-pds algorithm execution time use secondary y axis (on the right), and are measured by seconds.

5. Conclusion

We discuss the related problems of selecting subset of mirror servers for parallel download. Given a network topology, a client and a set of mirror servers, we presented brutal force algorithms and genetic algorithms for finding the maximum parallel download speed pds(c, S) and for finding the subset of mirror server with maximum parallel download speeding k-pds(c, S), given the size of the subset as k.

Further work is needed to investigate how effective are the above algorithms in selecting the right subset of mirror servers for parallel download.

6. Reference

[1] Pablo Rodriguez Andreas Kirpal Ernst W. Biersack, "Parallel-Access for Mirror Sites in the Internet", Proceeding of Infocom, 2000. http://www.ieee-infocom.org/2000/papers/65.ps

[2] Ratul Mahajan, "Aggregate Based Congestion: Detection and Control", April 2001. University of Washington.

[3] Vern Paxson, "Measurements and Analysis of End-to-End Internet Dynamics ", Ph.D. dissertation at UC Berkley.

[4] Kevin Lai and Mary Baker, "Nettimer: A Tool for Measuring Bottleneck Link Bandwidth", Proceedings of the USENIX Symposium on Internet Technologies and Systems, March 2001.

[5] Jing Yang and Zhong Li, "Selecting best Redhat Mirror Sites for parallel download", http://cs.uccs.edu/~cs522/proj2001/jyang.ppt

[6] Ellen W. Zegura, "GT-ITM: Georgia Tech Internetwork Topology Models", http://www.cc.gatech.edu/projects/gtitm/

[7] Thierry Ernst, "Existing NS-2 Presentation: GT-ITM. Topologies",http://www.inrialpes.fr/planete/pub/mobiwan/Documents/ernst-ns-mobiwan-0501.ppt

[8] John R. Koza, "Genetic Programming", MIT Press, 1992.

[9] Ellen W. Zegura, Mostafa H. Ammar, Zongming Fei, and Samrat Bhattacharjee, "Application-Layer Anycasting: A Server Selection Architecture and Use in a Replicated Web Service," IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 8, NO. 4, AUGUST 2000, pp. 455-466.

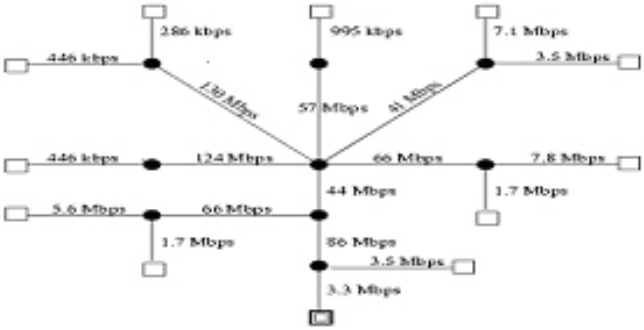


Figure 2. A sample routing tree with 20 nodes and 10 mirror sites [1].

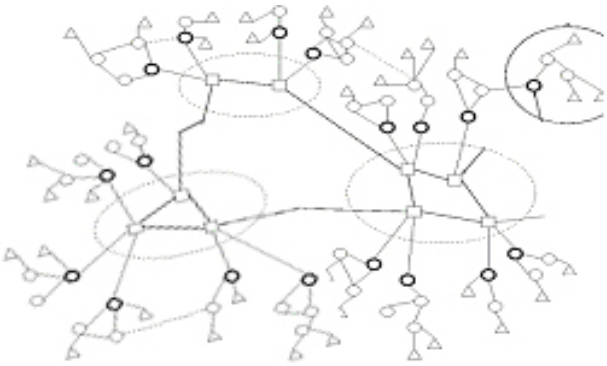


Figure 3. A sample transit-stub hierarchical network topology generate in GIT-ITM [6].

Dr. Yu Cai is an assistant professor at School of Technology in Michigan Technological University. His research interests include network protocols, distributed systems and cyber security. He received his Ph.D. in Computer Science from University of Colorado. He is a member of IEEE and ACM.

Dr. C. Edward Chow is a professor in Department of Computer Science at University of Colorado at Colorado Springs. He received his Ph.D. in Computer Science from University of Texas at Austin. His research interests include distributed systems and network security.

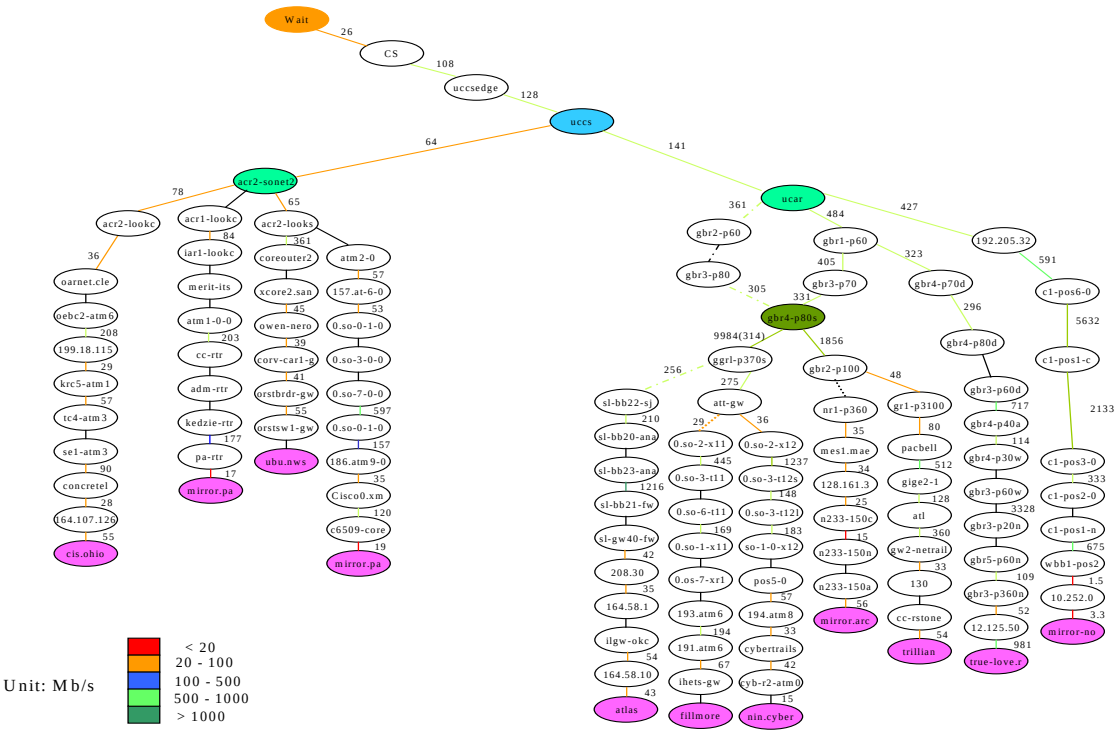


Figure 4. Network Topology of a UCCS client to a subset of Redhat mirror servers.

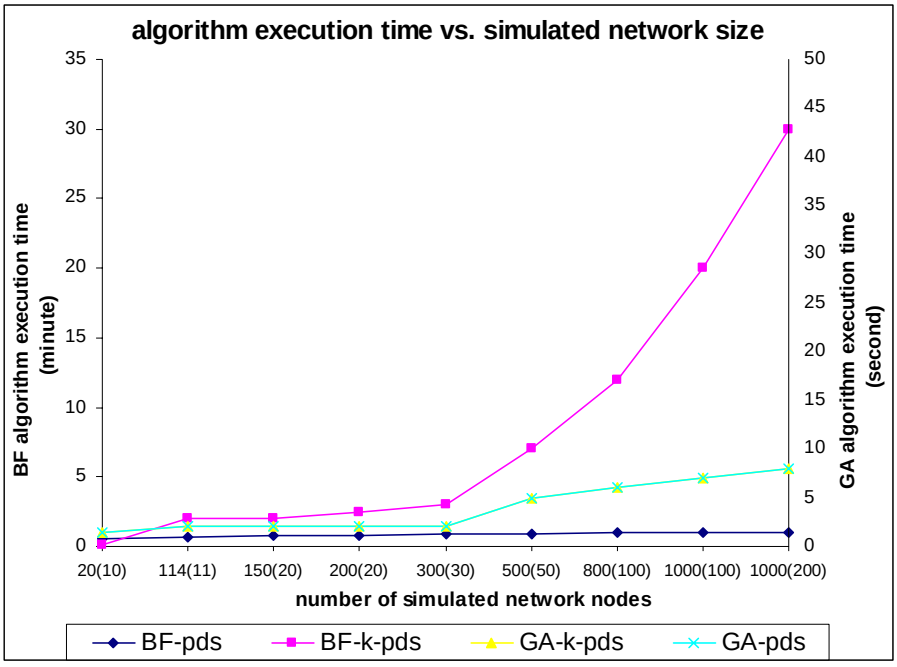


Figure 5: Algorithm execution time vs. simulated network size.
BF-k-pds algorithm execution time is measured by minutes.
BF-pds, GA-pds and GA-k-pds algorithm execution time are measured by seconds.