Design Methodology and Tools for Throughput Optimization of Web Servers

Abstract

Traditionally, CAD has focused on the design needs of integrated circuits and, more recently, electronic systems. However, the emerging and rapid growth of several system-level technologies and application domains has provided design, and particularly computer-aided design, with new and important domain-specific synthesis challenges. Today, perhaps the most popular and widely used system application and technology are the World Wide Web (WWW) and multiple-disk platforms.

The explosion of interest in the WWW has made popular Web servers so heavily accessed that many are unable to achieve acceptable throughput and response time. The immense data storage and transfer requirements of a Web server to support a growing volume of service requests requires that such servers be founded on multiple disks. The focus of this paper is on the methodology, algorithms and tools for proper design and operation of multiple-disk Web servers that achieve high throughput and low response time.

We identify four key problems in designing a multiple-disk Web server to optimize throughput under cost constraints: (i) disk selection, (ii) partitioning of files onto separate disks, (iii) assignment of files to specific cylinders on each disk, and (iv) scheduling of disk access requests for each disk. We present methods to optimize the throughput for each of the questions, and empirically validate and evaluate the methods through extensive trace-driven simulation.

Our experimental results indicate that the optimized system improves throughput by 65.4% upon the base system which employs C-SCAN scheduling, random assignment and random partitioning as is often the case in practice.

1 Introduction

1.1 Motivation

Design methodologies and computer-aided design tools have traditionally been aimed at the specification, synthesis and analysis of integrated circuits and, more recently, electronic systems. However, new domain-specific synthesis challenges to design – and computer-aided design in particular – are emerging in a number of exponentially growing system-level technologies and application domains. For example, the exceptionally rapid advance in microprocessor and general-purpose computer performance has been significantly outpaced by advances in disk storage capacity and network bandwidth. While the performance of the integrated circuits has increased at a compound annual rate of 25% [22], the capacity of magnetic disk storage has been increasing at a compound annual rate greater than 60% [23], and network bandwidth has recently increased by more than 3 orders of magnitude.

On the application side, probably the most popular and widely used system application and technology is embodied in the World Wide Web (WWW) and multiple disk-based WWW servers. The current explosive growth of interest in the WWW has made popular Web servers so heavily accessed that many are unable to achieve acceptable throughput and response time. The immense data storage and transfer requirements for a Web server to support continued growth in service requests forces such a server to be founded on multiple disks. This emerging domain provides numerous challenges and opportunities in design and optimization. There is a strong need for new synthesis approaches and design tools which can facilitate development and management of Web servers. Thus, our paper focuses on the methodology, algorithms and tools for proper design and operation of multiple-disk Web servers that achieve high throughput and low response time. We observe that the methodologies can be also used for synthesis of distributed memory-intensive systems. Our primary goal is to develop a simulation-based design methodology and infrastructure that facilitates efficient design of Web servers. Equally important, we seek to analyze the importance and relative impact of key design steps and isolate the key design trade-offs that are made during synthesis of Web servers.

We start by identifying four key problems for designing a multiple-disk Web server to optimize throughput under cost constraints: (i) disk selection, (ii) partitioning of files onto separate disks, (iii) assignment of files to specific cylinders on each disk, and (iv) scheduling of disk access requests for each disk. Many economic factors and performance issues are considered in the disk selection problem. For the partitioning problem, the main idea that we propose is to balance the workload over disks in both the short and long term by considering file size, frequency of access to files, as well as file access interdependencies. Our proposed file assignment heuristic places popular files in the center of the disk while also attempting to assign highly interdependent files closely together. To optimize throughput with low response time, our scheduling algorithm employs a server-initiated drop policy by assigning deadlines for requests. Since disk access requests arrive according to a probability distribution and are associated with assigned deadlines, our work involves statistical soft real-time system optimization. Finally, we empirically validate and evaluate the methods that we discuss, using extensive trace-driven simulation.

From the design point of view, the most interesting aspect of our work is the close coupling of simulation and design processes. From the optimization point of view, our main contribution is the development and evaluation of partitioning techniques for clustering of exceptionally large multidimensional data.

1.2 Motivational Examples: Scheduling, Assignment, Partitioning

For a Web server employing multiple disks to store its files, three questions naturally arise. First, how do we partition files into separate disks? Second, how do we assign files to specific cylinders on each disk? And third, how do we schedule disk access requests for each disk? For each of these questions, we now provide a a simple motivational example to show the importance of the problem.

When partitioning files onto disks to optimize throughput, many factors including file size and frequency of access to files should be considered. Suppose that there are two disks, and that files are partitioned only by balancing file sizes. This may result in a partition whereby one disk receives 90% of the total requests. the more popular disk will be too busy to serve requests, and the system will exhibit poor response time and throughput while the other disk sits idle. Section 6 presents our method to optimize throughput by partitioning. The main idea is to balance the workload for each disk in both the short and long term by considering file size, frequency of access to files, and file access interdependencies.

The disk seek time to serve requests, and thus the throughput, will depend on how files are assigned to specific cylinders. Suppose that two popular files are separated at opposite ends of a disk. Suppose also that the files are highly interdependent, such that if one is accessed, then the other file is also accessed at the same time (one possible scenario is that an HTML (HyperText Markup Language) file contains an image file. In this case the disk head must travel from one end to the other to serve the requests. The main idea of our method in Section 5 is to place popular files in the center of the disk while at the same time assigning highly interdependent files close to each other.

Finally, throughput can be improved by scheduling requests for disk access. Consider an ordered disk queue with requests according to arrival times involving cylinders 100, 2000, 10, 3000, and 500, and assume that the current disk head position is in cylinder 1900. If the scheduling policy is first-come first-served, the disk head must traverse 11180 cylinders to serve the 5 requests. If the shortest seek time first scheduling algorithm is used and no new requests arrive before all 5 requests are served, then the disk head will traverse only 4090 cylinders in serving all 5 requests. We see that the choice of scheduling policy significantly affects the disk seek time, which is a critical determinant of throughput. We also observe from our simulation study that highly overloaded Web servers can suffer unacceptably high response time even when good scheduling algorithms are used. To optimize throughput with low response time, we propose a scheduling solution that employs a server-initiated drop policy as described in Section 5.

1.3 Paper Organization

The rest of our paper is organized as follows. In the next section, we summarize the models used for our trace-driven simulation. In Section 3, we review related works on Web servers and other multimedia servers, as well as the scheduling, assignment, and partitioning problems. Section 4 describes a global synthesis approach for designing a multiple-disk Web server. Sections 5 and 6 form the technical core of the paper and present methods to optimize throughput via scheduling, assignment, and partitioning. Section 7 presents comprehensive experimental results and their analysis, and the paper concludes in Section 8.

2 Preliminaries

In this section we first discuss the selected model for describing the operation of a Web server. After that we provide a model which includes the characterization of files as well as disk access traffic.

2.1 Model of Web Server

The WWW is based on the client-server model: a Web client uses a Web browser such as Netscape, Mosaic, or Internet Explorer to send requests for documents or files to a Web server which responds with the requested documents. Web clients and servers communicate using the HyperText Transfer Protocol (HTTP) which runs on top of a transfer control protocol (TCP), a connection-oriented reliable byte stream protocol. Once communication between a Web server and its client is established using the HTTP and TCP protocols and a request is sent to the server from the client, the server must respond to the request. If the requested file is found and the client does not have a copy of it, the server transmits the requested file. For other cases, the explanation of the server's response (for example, no such document exists) is sent to the client.

We assume that a Web server consists of three components: disk, main memory and processor. The processor by itself can be comprised of multiple processors. Main memory is not used to store files.

The disk can be comprised of multiple disks. Our focus is on the throughput of the disk subsystem, which is a major bottleneck in Web servers. We assume that the disk subsystem is a separate unit, as is almost always the case in industrial practice. Finally, we assume that no file caching is performed; even though several studies (e.g., [7]) indicate that file caching on can improve Web server throughput by reducing disk access requests, this assumption does not impact any of our proposed methods since we seek to improve the throughput of the disk subsystem under any disk access traffic rate.

seek distance range [tracks]	seek time
1-50	$1.9 + \sqrt{d} - \frac{d}{50}$
51-100	$8.1 + 0.044 * (d - 50)$
101-500	$10.3 + 0.025 * (d - 100)$
501-884	$20.4 + 0.017 * (d - 500)$

Table 1: Typical seek time for an IBM disk

Disk capacity	2 GBytes
Spindle speed	5400 RPM
Average rotational latency	5.56 msec
Transfer rate	16.6 MBytes/sec
Average seek time	10.0 msec
Track-to-track seek time	2.0 msec
Cylinders	4095

Table 2: Disk parameters of Seagate-Medalist 2140 disk used in the paper

For a disk to serve a request, its disk head has to move to the cylinder where the file resides (seek time), the disk has to locate the position of the file in the cylinder (rotational latency), and then the transfer of the file can start (transfer time). Recently, several timing models of a magnetic disk have been proposed, especially with regard to seek time [31]. Common to all of these models is that longer distance that the disk arm must travel corresponds to a larger time overhead. Typical seek times for an IBM disk are given in Table 1 [22]. Disk transfer time is a function of the file size to transfer; we assume that it is actually proportional to the file size. We assume that the average rotational latency, which is half the time required for full rotation, is spent for each request. For the most up-to-date magnetic disk performance data see, e.g., [24]. The disk characteristics used in the paper are given in Table 2.

We assume that a Web server can drop requests when necessary to improve the throughput. When a disk receives a request from a processor, a disk scheduler assigns a maximum time within which the request must be served before it is dropped by the disk scheduler, the request then is placed on the queue. If the request is not served before the maximum time, the disk scheduler notifies the requesting processor that the request is dropped. Then, the processor responds to the Web client with the message that its request has been dropped due to too many requests at this time. In our model, we assign the deadline according to the function $arrival\ time\ of\ request + 5.0 + \frac{requested\ file\ size}{50\ KBytes}$ in seconds.

2.2 Traffic and File Model

Our model mostly follows the recent Web server workload characterization made from extensive real-life server data analysis by Arlitt and Williamson [4], the user access pattern analysis by Kwan et al. [18], and the file access interdependency study performed by Bestavros [6].

The inter-request times are assumed to be exponentially distributed and independent. Following the observations of [4], we model concentration of file references: (i) all files are sorted in decreasing order based on frequency of references, and (ii) each file is associated with its probability of reference which is drawn from a discretized exponential distribution such that 10% of the files are responsible for 90% of the total requests. File size follows the Pareto distribution [14] with $\alpha = 0.5$. Each file is assumed to be stored contiguously on a disk.

Since significant file access interdependencies are observed [6], our model incorporates this phenomenon. Given that a Web client has requested a particular file, we assume some known probability that it will request another file within a limited window of time (say, within t seconds). For example, if a file is embedded in another HTML file, then the probability is always 1. In general, collecting such probabilities requires thorough analysis of the client access patterns. We assume that the time window t is 1 second and that the arrival times of interdependent files follows the truncated Normal distribution with mean 0.5, standard deviation 0.15, minimum value 0.0, and maximum value 1.0. We also assume for each file that the number of files with nonzero access interdependency is randomly drawn between 1 and 10.

3 Related Efforts

In this section, we survey the related work. We first outline the directly relevant Web server and technologically and conceptually related video server literatures. We then provide a literature survey related to magnetic disks, disk modeling, and disk scheduling and assignment. Finally, we survey partitioning-related results.

3.1 Web and Video Servers

In the early 1990s, video-on-demand attracted a great deal of attention. As a consequence, a large number of disk-based storage schemes for retrieval of continuous video and audio data have been proposed and analyzed in the multimedia literature [21]. Since disk technologies were too expensive and compression schemes were not effective enough, many of the proposed approaches advocated use of tape-based systems. As it turned out, technological and economic trends and factors prevented widespread adoption of video-on-demand and related technologies. At the same time, the WWW emerged as an exceptionally popular and fast-growing market.

Kwan et al. [19] described design trade-offs and performance evaluation of the very popular NCSA WWW server. Andersen et al. [3] and Dias et al. [9] proposed approaches for design of scalable Web servers.

Recently, Arlitt and Williamson [4] and Braun and Claffy [7] analyzed the characteristics of Web server workload. It is interesting to note that a model of Web traffic was even used as the starting point for design of a video-on-demand server [28].

3.2 Disks and Disk Modeling

Magnetic disks provide very a favorable trade-off point in terms of cost, memory capacity, latency, bandwidth, and reliability in comparison with all other mass storage alternatives such as tapes, optical disks, flash memories, and emerging biological and holographic memories. Detailed description of magnetic disks and their operation can be found in many books, e.g., [15]. Wood and Hodhes [30] provide a survey of the state-of-the-art and technology trends in direct access storage devices, mainly magnetic disks.

Disk modeling has recently attracted a great deal of interest [22] [23] [31]. Relatively simple linear models have been used in the past (e.g., [29]), but the majority of recent disk models capture a variety of technological effects and provide exceptional statistical matching with measurements conducted on real-life disks under typical workloads [31].

3.3 Disk Scheduling and Assignment

Early disk-related research efforts emphasized the development of scheduling algorithms for efficient use of high-volume storage in time-shared operating systems [8] [27]. It appears that Denning was first to describe the first-come-first-served, shortest-seek-time-first, and SCAN disk scheduling algorithms [8]. A variety of relatively simple disk scheduling algorithms, including circular-SCAN (C-SCAN), were analyzed assuming typical mainframe workload in [27]. The proliferation of new disk technologies resulted in development of a series of new disk scheduling algorithms for new general-purpose computing platforms [25].

Numerous real-time scheduling techniques can be also used as starting points for development of disk scheduling algorithms [26].

In situations where a linear seek time model holds, along with known (non-correlated) access probabilities per datum, the Organ-Pipe permutation based on access probabilities generates optimal track assignments in terms of minimizing seek time on magnetic disks [29]. In Organ-Pipe permutation, the data with the largest access probability is positioned in the center of the disk, and the rest of the data are positioned, in decreasing order of access probability, on alternating sides of the center.

Berson et al. [5] advocate fully randomized data assignment as a conceptually and technically simple technique to provide probabilistic bounds on performances of mixed multimedia traffic. Docy et al. [10] proposed a heuristic for optimal assignment of data with correlated access patterns.

3.4 Partitioning

Partitioning has been a widely discussed optimization problem in many engineering and scientific fields. A comprehensive survey of lower-level CAD approaches is given in [1]. Numerous high-level synthesis partitioning techniques include works presented in [20]. Gupta and De Micheli [13] compared the effectiveness of simulated annealing and heuristic approaches for partitioning of programmable and application-specific designs.

4 Global Design Flow

The first stage of the design flow is to decide which type of disks are used to store files. The problem is complex where many economical factors and performance issues must be considered.

The second stage is to partition files into separate disks. The main idea is to balance the workload for each disk both in the short term and in the long run by considering file size, frequency of access to files, and file access interdependencies.

The third stage is to assign partitioned files to specific cylinders on each disk. The main idea is to place popular files in the center of the disk while assigning the highly interdependent files close to each other.

The fourth stage is to choose an scheduling algorithm of scheduling requests for disk access. To optimize the throughput with low response time, our scheduling algorithm employs a server-initiated drop policy by assigning deadlines for requests.

In the final stage we empirically validate and evaluate the methods through extensive trace driven simulation.

5 Disk Scheduling and Assignment Algorithms and Evaluation

In this section we formally define the disk scheduling and assignment problems and establish their computational complexity. We then propose heuristics and compare their effectiveness through extensive experimental results. Last, we discuss design methodology issues regarding the disk scheduling and assignment problems.

5.1 Problem Formulations

5.1.1 Disk Scheduling

We define *throughput* to be the percentage of requests served within the server-specified deadlines. A request is *dropped* by a server if it is not served by its deadline. The deadline for each request, i.e., the maximum allowed response time, should be decided by considering many factors such as the size of the requested file, how long requesters usually wait, the importance of requests, and so forth. If deadlines are set too long, throughput will be high, but average response time will also be high. If deadlines are set too short, average response time will be low, but throughput will also be low.

Traditionally, research has focused on fair disk scheduling algorithms because all requests must eventually be served in a disk subsystem for general purpose computers. SCAN and C-SCAN have been popular choices because these fair algorithms produce reasonably low response time. However, for a popular Web server, throughput and response time will be much more important objectives than fairness for the disk subsystem because the server can afford to drop a few requests for higher throughput and lower response time.

Informally, the disk scheduling objective for a Web server is to dynamically schedule requests to maximize for throughput given the deadlines for all current requests, and a probability distribution for future requests. Nonpreemption is assumed. Using standard Garey-Johnson [12] format:

Problem: Disk Scheduling Problem.

Instance: Given a current disk head position; a set of current requests described by deadline and execution time which is a function of current head position; a probability distribution for future request arrivals; and a constant k.

Question: Is there a nonpreemptive schedule for both current and future requests that achieves a throughput larger than k?

The special case, even when all requests are known and a constant execution time for each request is assumed, is known to be NP-complete [26].

5.1.2 Disk Assignment

Each file must be assigned to a specific position on a disk. The throughput of a disk depends on how files are assigned on the disk. Informally our problem is that given a scheduling algorithm, we must find an optimal assignment to maximize throughput.

Problem: Disk Assignment Problem.

Instance: Given a scheduling algorithm; a set of files described by sizes, frequencies of access, and access interdependencies; disk timing characteristics such as seek time, transfer time, and rotational latency; a probability distribution for future request arrivals; and a constant k.

Question: Is there an assignment of files to a disk so that the throughput is greater than k?

The special case, where all the requests are known and scheduled, and the objective is to minimize disk seek time, is known to be NP-complete [10].

5.2 Scheduling Heuristics

Since the disk scheduling must be performed on-line, any heuristic must be efficient. On the other hand, a heuristic can afford moderate time complexity due to the enormous speed difference between processor and disk: a processor has about 4 million clock cycles while a disk serves a 100 Kbyte file. (a modern processor runs in 5 nsec clock cycle time while disk access for such a file usually requires a seek time of 10 msec, a transfer time of 6 msec, and a rotational latency of 5 msec).

We have considered six different heuristics. Since our scheduling problem involves deadlines of requests, no heuristics will serve a request after its deadline has passed.

The first heuristic, *first-come first-served* is the simplest form of disk scheduling. Though the heuristic is extremely efficient, it suffers from high seek time cost and is unsuitable for a high-workload disk subsystem. The heuristic has poorly performed in our simulation study as expected, so the experimental results are not included in the paper.

The second heuristic is *shortest seek time first* (SSTF). The SSTF algorithm selects the request with the minimum seek time from the current head position.

The third heuristic is SCAN. The disk head starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder. When the head reaches the other end of the disk, the direction of head movement is reversed and servicing continues. Its variant, C-SCAN, moves the head from one end of the disk to the other, but does not service any requests on the return trip. Both SCAN and C-SCAN were used in our simulation study, but since they produced almost identical results, we report results for C-SCAN only.

The fourth heuristic is *earliest deadline first* (EDF). The EDF algorithm selects the request with the earliest deadline. Note that EDF is optimal when all the requests have the same arrival times and service times for requests are constant [26].

The fifth heuristic is *maximum requests first* (MRF). The MRF algorithm selects the cylinder with the maximum number of requests and serves all the requests in the selected cylinder.

The sixth heuristic is *maximum reward per time first* (MRPTF). The reward per time for each cylinder is defined to be the number of requests in the cylinder divided by the required service time for the requests (i.e., the sum of seek time, rotational latency, transfer time for the requests). The MRPTF algorithm selects the cylinder with the maximum reward per time and serves all the requests in the cylinder.

5.3 Assignment Heuristics

We have tested three different heuristics. The first heuristic is the simplest one which places files on random cylinder locations.

The second heuristic is the modified Organ-Pipe (MOP) heuristic which takes into account the access frequencies and file sizes. In the original Organ-Pipe heuristic, the file with the largest access probability is positioned in the center of the disk, and the rest of the files are positioned, in decreasing order of access probability, on alternating sides of the center. Due to file size differences, this might not yield an assignment for which the total access probability for each cylinder follows the organ-pipe permutation. The modified Organ-Pipe heuristic sorts all files in decreasing order of $\frac{access\ probability}{file\ size}$ and applies the original Organ-Pipe permutation in this order.

The third heuristic is based on the idea to place popular files in the center of the disk while placing interdependent files in close locations. Simulated annealing [17] is used to minimize $\sum_{i=1}^{n} \alpha p_i |c_i - m| + \sum_{i=1}^{n} \sum_{j=1}^{n} \beta p_i d_{ij} |c_i - c_j|$ with the size constraint for each cylinder, where n is the number of files, α, β are constant, p_i is an access probability for file i, c_i is a cylinder number for file i, m is the cylinder number for the cylinder in the center of the disk, and d_{ij} is a file access interdependency probability from file i to file j. α and β are empirically determined to be 1.0 and 0.2, respectively. The details of the method are presented in the extended version of this paper.

5.4 Experimental Results

For experiments, we use the following values in addition to the models defined in Section 2. The number of files stored on a disk is 70000 and their total size is approximately 1.6 GBytes. The number of cylinders used to store files is 3719 and the initial head position is assumed to be in the middle of the disk. All results reported in the paper are based on the averages of 10 simulation runs.

First, we demonstrate the effectiveness of server-initiated drop policy on the throughput and response time of disk subsystem. We have tested all combinations of the assignment and scheduling heuristics under two different workloads. Table 3 presents the experimental results which clearly show the effectiveness of the server-initiated drop policy. We have also conducted simulation studies under heavier workloads and for longer simulation times; results follow the same trend as those in Table 3.

	Random Assignment				MOP Assignment				SA Assignment			
	200 reqs/sec		400 reqs/sec		200 reqs/sec		400 reqs/sec		200 reqs/sec		400 reqs/sec	
Schedule	WOD	WD	WOD	WD	WOD	WD	WOD	WD	WOD	WD	WOD	WD
C-SCAN	4.9	58.7	2.0	30.7	6.2	64.6	3.0	32.3	6.9	66.8	3.7	34.9
SSTF	28.1	59.3	19.4	30.1	33.2	66.1	22.1	33.3	34.4	70.5	23.2	36.2
EDF	4.4	27.8	1.7	18.3	5.8	61.0	2.2	33.8	6.2	65.2	2.8	35.6
MRF	9.2	55.7	2.8	32.0	11.5	70.8	3.5	39.0	12.7	76.9	4.0	45.1
MRPTF	39.2	68.5	14.1	38.8	45.2	73.5	14.9	41.6	47.3	78.9	15.8	45.7

Table 3: Throughput results for a simulation time of 600 seconds; WOD - without server-initiated drops, WD - with server-initiated drops.

	Random Assignment				MOP Assignment				SA Assignment			
Schedule	200 reqs/sec	400	600	800	200	400	600	800	200	400	600	800
C-SCAN	58.5	31.6	22.6	16.3	64.5	32.3	24.4	16.5	66.5	33.6	25.3	17.2
SSTF	59.2	30.0	22.1	15.5	65.4	33.1	25.1	16.8	68.9	34.5	26.2	18.8
EDF	27.6	18.2	15.3	12.7	65.2	33.7	25.6	18.2	66.5	34.2	26.6	18.4
MRF	55.6	31.8	24.2	16.9	70.4	38.9	29.5	20.5	76.3	45.2	33.4	22.1
MRPTF	68.7	38.7	29.9	20.8	73.2	41.3	31.8	21.5	79.2	46.6	35.2	23.1

Table 4: Throughput results for a simulation time of 1200 seconds with server-initiated drops

For more thorough study, the simulation is performed under more diverse traffic workloads for longer time. The results provided in Table 4 indicate the same trend as that from Table 3. Among the five scheduling algorithms, the MRPTF algorithm performs best for all three different assignment algorithms when server-initiated drop policy has been employed. For all different heuristics, the SA assignment algorithm produces best throughput results while the MOP assignment algorithm performs comparably.

6 Partitioning

In this section, the problem of partitioning files onto multiple disks is defined and an optimization algorithm is proposed.

6.1 Problem Formulation

The performance of multiple-disk Web servers depends on the distribution of the workload among the disks. The throughput of the multiple disks is higher when the imbalance in the workload distribution among the disks is smaller. Thus, the total amount of data and the total probability of reference summed over all the files on a disk should be comparable to the totals on other disks.

The balance of the data amount and frequency of reference targets long-term workload balancing of disks. When file access interdependencies are considered, short-term workload balancing is also necessary. Because highly interdependent files are likely to be accessed closely in time, it is better to put them in different disks.

Our goal in the partitioning problem is to partition files such that the size and probability of access for each partition is balanced, and the interdependency probability between files in *different* partitions is maximized. The problem is NP-complete by a simple reduction from graph MIN CUT [12].

The problem can be reformulated as a graph MAX CUT partitioning problem, where each file represents a vertex, the interdependency probability between files is the weight of the edge between nodes, the size and probability of access for a file are two different weights for the vertex, and the objective is to maximize the sum of edge weights between vertices in different partitions.

6.2 Optimization Algorithm

We propose two optimization heuristics: (i) random partitioning and (ii) MAX CUT partitioning. The random partitioning is simple, where each file is randomly placed onto a disk. This heuristic is very simple-minded, but is nevertheless tested because there is evidence that the approach may perform well, e.g., Berson et al. [5] advocate fully randomized data partitioning and assignment.

For MAX CUT partitioning, we use an iterative multi-level partitioning engine that is a modification of code originally due to Alpert [2]. The input to such a partitioning engine is a vertex- and edge-weighted graph $G = (V, E)$ with n vertices $V = \{v_1, v_2, \ldots, v_n\}$ and m edges $E = \{e_1, e_2, \ldots, e_m\}$. We use $w(e)$, $w(v)$, etc. to denote the weight of edges, vertices, sets of vertices, etc. A *bipartitioning* $P = \{X, Y\}$ is a pair of disjoint *clusters* (i.e., subsets of V) X and Y such that $X \cup Y = V$. The *cut* of a bipartitioning $P = \{X, Y\}$ is the number of edges which contain vertices in both X and Y, i.e., $cut(P) = |\{e \mid e \cap X \neq \emptyset, e \cap Y \neq \emptyset\}|$. Given a parameter r that indicates the desired ratio of $w(X)$ to $w(V)$, along with a tolerance (or deviation) parameter d, the *min-cut bipartitioning problem* seeks a solution $P = \{X, Y\}$ that minimizes $cut(P)$, subject to $\frac{r-d}{\sum_{i=1}^{n} w(v_i)} \leq w(X) \leq \frac{r+d}{\sum_{i=1}^{n} w(v_i)}$. A common variation assigns the cost of a bipartitioning solution to be the sum of the edge *weights* crossing the cut.

A standard bipartitioning approach is iterative improvement based on the Kernighan-Lin (KL) [16] algorithm, which was later improved by Fiduccia-Mattheyses (FM) [11]. FM has been widely adopted, particularly in the VLSI CAD domain, for its short runtimes and ease of implementation. Our partitioning engine is a significant improvement to FM which integrates *clustering* into "two-phase" or "multi-level" approaches and supports k-way partitioning [2] Runtimes are quite adequate, e.g., the code requires 32 seconds of CPU time on a Sun Ultra-1 for 4-way partitioning of a 25,000-node circuit netlist. We have adapted this partitioner, as we now describe.

Our approach requires several non-traditional capabilities of the minimum-cut bipartitioner: (i) handling a max-cut formulation, (ii) producing multi-way cuts, and (iii) maintaining feasibility with respect to a "multi-dimensional" balance constraint. Heuristic max-cut partitioning is achieved by multiplying all edge weights by -1 and then applying the minimum-cut objective. Multi-way cuts are obtained by recursive bipartitioning with appropriate target cluster weight ratios and tolerances. It is possible for a non-hierarchical k-way partitioner to achieve solution qualities that a hierarchical bipartitioner cannot, but for our present studies hierarchical bipartitioning is simpler and easier to control.

The multi-dimensional balance constraint arises because we are trying to balance multiple "weight" criteria, i.e., each vertex (file) has a file size and an access frequency, and no cluster (disk) should be assigned to much of either type of weight. We have adapted the feasibility checking in the bipartitioner to handle multi-dimensional balance constraints: multiple weight types are supported with separate target ratio and tolerance for each weight type, and every proposed move is checked with respect to all balance constraints before being implemented. We use a simple heuristic to ensure that an initial partitioning solution for FM is feasible with respect to the multi-dimensional balance constraints: (i) if all excess violations are "in the same direction" (e.g., cluster X has too much file size and too much access frequency) then a random vertex is moved out of cluster X; and (ii) if excess violations are in multiple directions (e.g., cluster X has too much file size and cluster Y has too much access) then a random vertex is moved out of the cluster that has a proportionally larger degree of violation. Such moves are repeated until balance is obtained or too many attempts have failed.

7 Putting it all Together

In this section, we first discuss the disk selection problem and demonstrate the effectiveness of the proposed design methods by showing the experimental results of comprehensive simulation study on multiple-disk platform.

7.1 Disk Selection

The problem of deciding which type of disks are to be used for storing files is complex where many economical factors and performance issues must be considered. We identify such relevant decision factors as cost, capacity, seek time, transfer rate, and rotational latency of disks. A possible way is to optimize performance factors, seek time, transfer rate, and rotational latency in relative terms to cost while satisfying capacity constraint. The problem is a topic for further study.

7.2 Experimental Results

For experiments, we use the following values in addition to the models defined in Section 2. The number of files stored are 140000 and their total size is approximately 3.2 GBytes. The number of disks used to store the files is 2 or 4.

First, the performance of MAX CUT and random partitioning heuristics is compared with respect to the results of weighted cut. The number of files considered varies between 10000 and 140000. The experimental results are provided in Table 5. The MAX CUT partitioning heuristic outperforms the random partitioning heuristic by 68.9(25.2)% for the case of 2(4) disks on average.

# Files	# Disks	
	2	4
10000	72.5%	23.9%
30000	68.7%	22.6%
60000	71.3%	27.3%
80000	68.8%	25.2%
110000	65.2%	26.3%
140000	66.9%	25.7%

Table 5: Percentage improvement of cut results for MAX CUT partitioning heuristic compared to random partitioning heuristic

Next, the performance of partitioning heuristics with the best performing scheduling and assignment algorithms is compared under several different workloads during the simulation time of 1200 seconds. Table 6 presents the experimental results which clearly show that the MAX CUT partitioning heuristic produces significantly higher throughput than the random partitioning heuristic.

Finally, the throughput of optimized system which uses best performing scheduling, assignment, and partitioning algorithms is compared to the base system which uses C-SCAN scheduling with server-initiated drop policy, random assignment, and random partitioning heuristics under various workloads during the simulation time of 1200 seconds on multiple-disk platform with different numbers of disks. The results are presented in Table 7. The average throughput improvement is 65.4% excluding the effect of a server-initiated drop policy which enormously improves throughput and response time under heavy traffic workload.

# of disks	Random Partitioning			MAX CUT Partitioning			CUT Imp	Throughput Imp
	400 reqs/sec	800	1600	400	800	1600		
2	73.6	42.1	20.8	81.8	47.9	24.5	66.9%	13.5%
4	96.7	71.8	39.3	99.8	81.5	47.2	25.7%	12.3%

Table 6: Comparison of throughput results between partitioning heuristics

# of disks	Base System			Optimized System			Improvement
	400 reqs/sec	800	1600	400	800	1600	
1	31.6	16.3	7.9	46.6	23.1	11.5	44.9%
2	50.2	23.8	11.7	81.8	47.9	24.5	91.2%
4	83.3	51.9	23.2	99.8	81.5	47.2	60.1%

Table 7: Comparison of throughput results between optimized system and base system

7.3 Future Directions

Replication of files seems a viable approach to further improve the throughput of Web server. A related study [10] also indicates it is a promising direction to pursue.

The disk selection problem deserves more attention. Optimization intensive problem formulation and analysis needs to be done. One very interesting question is what can be done to optimize the throughput when adding new disks to existing ones.

A Web server can be founded on a set of heterogeneous disks with respect to capacity and performance. For this type of system, a further study on the partitioning problem is necessary. When heterogeneous disks are allowed, the disk selection problem will be more complicated, but wider set of design possibilities can be considered, and more cost-effective design can be produced.

8 Conclusion

Methodologies for designing multiple-disk Web servers to achieve high throughput and low response time are proposed. In this paper we identified four key problems for designing a multiple-disk Web server to optimize throughput: (i) disk selection, (ii) partitioning of files into separate disks, (iii) assignment of files to specific cylinders on each disk, and (iv) scheduling of disk access requests for each disk. We presented methods to optimize the throughput for each of the questions, and empirically validated and evaluated the methods through extensive trace driven simulation. The experimental results indicate that the optimized system improves throughput by 65.4% upon the base system.

References

[1] C.J. Alpert, A.B. Kahng, " Recent Directions in Netlist Partitioning: a Survey", Integration, The VLSI Journal, Vol. 19, No. 1-2, pp. 1-81, 1995.

[2] C. J. Alpert, "Multi-way Graph and Hypergraph Partitioning", Ph.D. Thesis, UCLA Computer Science Department, 1996.

[3] D. Andersen et al., "SWEB: Towards a Scalable World Wide Web Server on Multicomputers", International Parallel Processing Symposium, pp. 850-856, 1996

[4] M.F. Arlitt, C.L. Williamson, "Web Server Workload Characterization: The Search for Invariants", ACM SIGMETRICS Conference on the Measurement and Modeling of Computer Systems, pp. 126-137, 1996.

[5] S. Berson, R.R. Muntz, W.R. Wong, " Randomized Data Allocation for Real-time Disk I/O", COMPCON '96 Technologies for the Information Superhighway, pp. 286-90, 1996.

[6] A. Bestavros, "Using Speculation to Reduce Server Load and Service Time on the WWW", The Fourth ACM International Conference on Information and Knowledge Management, pp. 403-410, 1995.

[7] H. Braun, K. Claffy, "Web Traffic Characterization: An Assessment of the Impact of Caching Documents from NCSA's Web Server", Computer Networks and ISDN Systems, Vol. 28, No. 1-2, pp. 37-51, 1995.

[8] P.J. Denning, "Effects of Scheduling on File Memory Operations", Proc. of the AFIPS Spring Joint Computer Conference, pp. 9-21, 1967.

[9] D.M. Dias et al., "A Scalable and Highly Available Web Server", COMPCON '96. Technologies for the Information Superhighway. pp. 85-92, 1996.

[10] S. Docy, I. Hong, M. Potkonjak, "Throughput Optimization in Disk-Based Real-time Application Specific Systems", International Symposium on System Synthesis, November 1996.

[11] C. M. Fiduccia and R. M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions", Design Automation Conference, pp. 175-181, 1982.

[12] M. R. Garey, D. S. Johnson, "Computer and Intractability: A Guide to the theory of NP-Completeness", W. H. Freeman and Co., New York, NY, 1979.

[13] R. Gupta, G. De Micheli, "Partitioning of Functional Models of Synchronous Digital Systems", International Conference on Computer-Aided Design, pp. 216-219, 1990.

[14] N. Johnson, S. Kotz, "Encyclopedia of Statistical Sciences", Volumes 6 and 9, John Wiley & Sons, Inc., New York, 1988.

[15] F. Jorgensen, "The Complete Handbook of Magnetic Recording", 4th ed, TAB Books, New York, NY, 1996.

[16] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", Bell Systems Tech. J., Vol. 49, No. 2, pp. 291-307, 1970.

[17] S. Kirkpatrick, C. Gelatt, M. Vecchi, "Optimization by Simulated Annealing," Science, Vol. 220, No. 4598, pp. 671-680, 1983.

[18] T.T. Kwan, R.E. McGrath, D.A. Reed, "User Access Patterns to NCSA's World Wide Web Server", Technical Report UIUCDCS-R-95-1934, University of Illinois, Urbana, 1995.

[19] T.T. Kwan, R.E. McGrath, D.A. Reed, " NCSA's World Wide Web server: Design and Performance", Computer, Vol. 28, No. 11, pp. 68-74, 1995.

[20] M.C. McFarland, "Computer-Aided Partitioning of Behavioral Hardware Descriptions", Design Automation Conference, pp. 472-480, 1983.

[21] A.N. Mourad, "Issues in the Design of a Storage Server for Video-on-demand", Multimedia Systems, Vol. 4, No. 2, pp. 70-86, 1996.

[22] D.A. Patterson, J.L. Hennessy, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann, San Mateo, CA, 1990.

[23] C. Ruemmler, J. Wilkes, "An Introduction to Disk Drive Modeling", IEEE Computer Magazine, Vol. 27, No. 3, pp. 17-28, 1994.

[24] http://www.seagate.com/disc/prodmatrix.shtml

[25] M. Seltzer, P. Chen, J. Ousterhout, "Disk Scheduling Revisited", Proc. of USENIX, pp. 313-323, 1990.

[26] J.A. Stankovic et al.,"Implications of Classical Scheduling Results for Real-time Systems", Computer, Vol. 28, No. 6, pp. 16-25, 1995.

[27] T.J. Teorey, T.B. Tinkerton, "A Comprehensive Analysis of Disk Scheduling Policies", Communications of the ACM, Vol. 15, No. 3, pp. 177-184, 1972.

[28] D. Venkatesh, T.D.C. Little, "Investigation of Web Server Access as a Basis for Designing Video-on-demand Systems", Proceedings of the SPIE - The International Society for Optical Engineering, Vol. 2617, pp. 2-11, 1996.

[29] C.K. Wong, "Algorithmic Studies in Mass Storage Systems", Springer-Verlag, Berlin, Germany, 1983.

[30] C. Woods, P. Hodges, "DASD Trends: Cost, Performance, and Form Factor", Proc. of the IEEE, Vol. 81, No. 4, pp. 573-585, 1993.

[31] B.L. Worthington et al., "On-line Extraction of SCSI Disk Drive Parameters", Performance Evaluation Review, Vol. 23, No. 1, pp. 146-156, 1995.