Load Balancing in Multimedia Servers

Joep Aerts[1,2], Jan Korst[1], Wim Verhaegh[1]
[1] *Philips Research Laboratories, Prof. Holstlaan 4, WY-21, 5656 AA Eindhoven*
aertsj@natlab.research.philips.com
[2] *Eindhoven University of Technology, Dept. of Mathematics and Computing Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

Abstract

An important resource management issue in multimedia servers is to balance the load on the available hard disks. Using random redundant storage strategies we can balance block requests over the disks by exploiting the freedom to service them by different disks. This induces a so-called retrieval problem in which to decide for each block which disk to use for its retrieval. We formulate this problem as an MILP problem, prove that it is $\mathcal{NP}$-complete, and show that the problem can be seen as a multiprocessor scheduling problem. Next, we describe a heuristic with a performance guarantee based on an LP-relaxation.

1 Introduction

A multimedia server offers continuous streams of multimedia data to multiple users. In a multimedia server one can often distinguish three parts: an array of hard disks to store the data, an internal network, and a fast memory from which the users consume, which is usually implemented in RAM. The multimedia data is stored on the hard disks in blocks, such that a data stream is realized by a series of blocks. The total RAM is split up in a number of buffers, one for each user. A user consumes, possibly at a variable bitrate, from his/her buffer and the buffer is refilled with blocks from the hard disks. A buffer generates a request for a new block as soon as the amount of data in the buffer becomes smaller than a certain threshold. We assume that the requests are handled periodically in batches in such a way that the requests that arrive during one period are serviced in the next one. In the server we need a cost-efficient storage and retrieval strategy, that guarantees, either deterministically or statistically, that the buffers do not underflow or overflow.

The cost of a multimedia server mainly consists of the costs of hard disks and RAM. The number of hard disks needed for a given number of users depends on the disk storage and disk transfer rate requirements of the entire system. Our analysis focuses on applications for which the transfer rate requirements form the bottleneck, as in some video on demand applications. This means that it is important to use the available bandwidth of the disk array as efficiently as possible and for that we have to (1) use each single disk efficiently, i.e., to optimize the effective reading time, and (2) balance the workloads of the disks. To optimize the effective reading time we should make the data blocks as large as possible, but this is in contrast with the buffer costs, as the amount of RAM per user grows linearly in the blocksize. In this paper we do not discuss

this trade-off any further; we focus on the load balancing performance of redundant storage strategies.

The load balancing performance of a storage strategy is an important factor in the disk costs, as an efficient usage of the available bandwidth of the disk array increases the maximum number of allowed users. In this paper we analyze the load balancing performance of random redundant storage strategies: each data block of a stream is stored on a randomly chosen subset of the disks. Redundancy gives the freedom to obtain a balanced load. To exploit this freedom we have to solve in each period a retrieval problem, i.e., to decide for each data block which of the disks to use for its retrieval. Prior work [1, 3, 5] shows that redundant storage strategies have a good load balancing performance. A drawback of the redundant storage strategies is the disk storage overhead, but in case the disk transfer rate forms the bottleneck instead of the disk storage capacity this need not be a serious drawback. In this paper we describe a general model for the retrieval problem, describe the problem in terms of multiprocessor scheduling, analyze its complexity, and give approximation algorithms and their performance.

2 Problem formulation

We consider a multimedia data server that handles requests periodically in such a way that in each period the data blocks are fetched that were requested in the preceding period. Each data block is stored on a given subset of disks. As we consider multi-zoned disks, the read time for a block depends on the zone in which the block is stored. For each data block we have to decide from which disk(s) to retrieve it. We allow that different parts of one block are retrieved from different disks with the advantage that we have more load balancing freedom. The drawback of splitting up a block access is that the total number of accesses increases, which generally results in a larger total switch time. We approximate the switch time per disk using a linear function in the number n of blocks to retrieve from that disk, i.e., $s(n) = s \cdot n + c$. The problem is to find for each period an assignment of the data blocks to the disks that results in a minimal period length, where the period length is defined as the maximum over the disks of the sum of the switch time and the read time. More formally, the problem can be defined as follows.

Problem 1 [Retrieval Problem].
Given are a set M of disks and a set J of data blocks. For each block $j \in J$ and disk $m \in M$ a 0/1-parameter u_{jm} indicates whether block j is stored on disk m and p_{jm} gives the transfer time to read block j from disk m. Furthermore the parameters s and c of the switch time function are given. We introduce for all $j \in J$ and $m \in M$ a decision variable x_{jm}, indicating the fraction of block j to retrieve from disk m, and a corresponding 0/1-variable y_{jm} that indicates whether or not disk m is accessed for (a part of) block j. Note that $x_{jm} > 0$ implies that $y_{jm} = 1$. The problem is to find values for x_{jm} and y_{jm} such that

$$\max_{m \in M} \left(\sum_{j \in J} x_{jm} p_{jm} + s \sum_{j \in J} y_{jm} + c \right) \tag{2.1}$$

is minimized, subject to

$$\forall_{j \in J, m \in M} \quad 0 \leq x_{jm} \leq u_{jm}, \tag{2.2}$$

$$y_{jm} \in \{0, 1\}, \tag{2.3}$$

$$x_{jm} \leq y_{jm}, \tag{2.4}$$

$$\forall_{j \in J} \quad \sum_{m \in M} x_{jm} = 1. \tag{2.5}$$

The retrieval problem can be seen as a multiprocessor scheduling problem, by viewing the disks as machines and the block requests as jobs. In the well-known three-field classification the machine environment of our problem is given by unrelated parallel machines, the special constraints on the jobs are machine eligibility and preemption with set-up cost, and the optimization criterion is the makespan. The result is that the retrieval problem can be denoted by $R \mid M_j, \text{pmtn} + \text{set-up} \mid C_{\max}$. A difference between the retrieval problem and a standard multiprocessor scheduling problem is that it is allowed that two disks work on the same block at the same time in case of preemption. This means that in contrast to multiprocessor scheduling a preemted assignment of the blocks to the disks is automatically a feasible schedule for retrieval.

3 Complexity and solution approach

We can prove that the feasibility variant of the retrieval problem is NP-complete in the strong sense by a reduction from 3-partition, which is NP-complete in the strong sense [2].

Theorem 1 *NP-completeness The feasibility variant of the retrieval problem is NP-complete in the strong sense.*

Proof (sketch). Considering an instance of 3-partition with items $a_1, \ldots, a_{3n}$ and a binsize $B = \frac{1}{n} \sum a_i$, we construct an instance of the retrieval problem in the following way. For each a_j of the 3-partition instance we define a block j which is stored on all disks, i.e., $u_{jm} = 1$ for all $m \in M$, and has a transfer time $p_{jm} = p_j = a_j$ on all disks. Furthermore we define the value s in the objective function as $s = B$, the makespan bound T of the retrieval problem as $T = 4B$, and the constant c of the switch time function as 0. With this transformation we can conclude that 3-partition is a special case of the retrieval problem. □

In the full paper we show that the retrieval problem remains NP-complete in the strong sense in case of random duplicate storage, i.e., each block is stored on exactly two randomly chosen disks.

A straightforward way to derive a heuristic for the retrieval problem is by solving its LP-relaxation and rounding up the y-variables. In the solution of the LP-relaxation each y-variable will have the same value as the corresponding x-variable, as we want to minimize the period length and $s > 0$. This means that we can omit the y-variables from the formulation. We use an LP-solver to solve the resulting LP-problem, in which we minimize T subject to:

$$\forall_{m \in M} \quad \sum_{j \in J} x_{jm}(p_{jm} + s) + c \leq T, \tag{3.1}$$

$$\forall_{j \in J, m \in M} \quad 0 \leq x_{jm} \leq u_{jm}, \tag{3.2}$$

$$\forall_{j \in J} \quad \sum_{m \in M} x_{jm} = 1. \tag{3.3}$$

We construct an algorithm ALG1 for the retrieval problem as follows. ALG1 solves the LP-relaxation, rounds up the y-variables and computes the the actual cost using equation (2.1).

Theorem 2 *Performance ratio of ALG1 The absolute performance ratio of ALG1 is bounded by* $1 + \frac{|M|^2 \cdot s}{|J| \cdot (p_{\min} + s)}$ *in which* $p_{\min}$ *is the minimum transfer time.*

In practice, the ratio between $|J|$ and $|M|$ depends on the ratio between disk transfer rate and consumption rate, such that it gives an indication for the number of users that can be serviced by one disk. For a given set of system parameters this ratio is more or less constant, and consequently, the other $|M|$ factor in the numerator of the formula in Theorem 2 makes that the performance guarantee grows in the size of the system. To improve on this, we describe in the full paper a second algorithm based on the work of Lenstra, Shmoys, and Tardos [4] for the non-preemptive scheduling problem $R \,||\, C_{\max}$, which results in a performance ratio of $1 + \frac{\min\{|M|s, p_{\max} + s\}}{\frac{|J|}{|M|}(p_{\min} + s)}$.

References

[1] J. Aerts, J. Korst, and S. Egner, Random duplicate storage strategies for load balancing in multimedia servers, submitted to Information Processing Letters.

[2] M.R. Garey and D.S. Johnson, (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, San Fransisco.

[3] J. Korst, (1997) Random duplicated assignment: An alternative to striping in multimedia servers, Proceedings ACM Multimedia '97, 219-226.

[4] J.K. Lenstra, D.B. Shmoys, and E. Tardos, (1990) Approximation algorithms for scheduling unrelated parallel machines, Mathematical Programming 46, 259-270.

[5] P. Sanders, S. Egner, and J. Korst, (2000) Fast concurrent access to parallel disks, to appear in Proceedings SODA 2000.