Pergamon

Information Systems Vol. 19, No. 4, pp. 33–54, 1994

0306-4379(94)E0004-9

0306-4379/94 $7.00 + 0.00

AN ANALYSIS OF SCHEDULES FOR PERFORMING MULTI-PAGE REQUESTS

Bernard Seeger

Fachgebiet Informatik, Universität Marburg, Hans-Meerwein-Str., D-35032 Marburg, Germany

(Received 18 June 1993; in final revised form 1 November 1993)

Abstract — In this paper, we address the problem of efficiently reading a set of pages from a file which is kept on some cylinders of a magnetic disk. It is assumed that several pages can be read from disk in a single multi-page request without interruption by other requests. First, a simple algorithm is presented for computing a schedule how the required pages are read from disk. Then, the expected cost of the schedules is analyzed by using a pure analytical model. In addition to the disk geometry, the analysis takes into account the three major cost components (seek time, rotational delay and transfer time) which occur when pages are retrieved from magnetic disk. The derived cost function depends on two parameters: the number of required pages and the degree of clustering of the underlying file. The cost function demonstrates that significant performance improvements can be achieved by using multi-page requests when the required pages are read according to a well-computed schedule. In addition to the response time of a query exclusively performed in the system, we also examine the impact of multi-page requests on the average response time when multiple queries are concurrently performed at a time. Results of an experimental performance comparison demonstrate that the average response time can considerably be improved on a highly loaded system by several factors when multi-page requests are used in comparison to reading the required pages one at a time.

Key words: I/O performance, multi-page requests, cost functions, query optimization, throughput

1. INTRODUCTION

Writing and reading some subset of disk pages is one of the most frequently used elementary operations in a database system. This operation occurs, for example, when several modified buffer pages are written back to disk [5, 11] or when an index is used for the evaluation of a selection query, e.g. range query. One way to read (write) a set of disk pages is to access the pages in random order without considering their physical position on disk. In general, the time required for reading (writing) a page can be improved when the pages are accessed according to a well-computed schedule. Such a schedule has to take into account that the cost for reading (writing) a page depends on the position of the page previously accessed. Consequently, pages close to each other on disk should be adjacent in the schedule.

The problem studied in this paper is as follows. Given a list of pages to be retrieved from a file stored on a magnetic disk drive, what is an efficient schedule of retrieving them and what is the corresponding retrieval cost. It is assumed that multiple pages scattered on disk can be read with a single request, called multi-page request in the following. The cost of reading a page consists of three components: seek time, rotational delay and transfer time. Transfer time is assumed to be fixed for all pages, whereas seek time and rotational delay depends on the previous position of the disk arm. Moreover, the geometry of a magnetic disk is also taken into account. A disk is considered as a collection of cylinders, each cylinder is a stack of tracks and each track consists of a fixed number of pages. This disk model is more accurate, but also more complex than the model assumed in one of our previous studies [17].

During the last few years the problem of computing efficient schedules has attracted much attention. Due to the availability of safe RAM [6, 7], "dirty" pages can be kept in a cache without facing the risk of loosing the updates through a system crash. Then, in order to improve write performance, the modified pages can be written back on disk at an appropriate point later in time. This approach is particularly beneficial for disk arrays [14, 13]. Polyzois et al [14] presented a first analysis and showed the advantage of using efficient schedules in comparison to random schedules.

The analysis is based on a hybrid of analytic-simulation model whereas our analysis (which gives similar results) is completely based on an analytical model.

The technique of multi-page request is frequently considered for improving the response time of various types of queries in database systems. For reading large objects, Weikum [21] and Biliris [1] have found that multi-page requests considerably improve the response time. In order to improve processing of multidimensional range queries, Hutflesz et al [12] have proposed a multidimensional access method that exploits multi-page requests. Cheng et al [4] have proposed a new join method, called Hybrid join, which uses multi-page requests for efficient data access. Brinkhoff and Kriegel [2] applied multi-page requests to processing spatial joins. In contrast to these approaches, Seeger et al [17] studied the general problem of reading a set of disk pages without considering a special operation of the database system. However, their results were derived under the assumption of the so-called *linear disk model* that is more related to a storage tape than to a disk drive. The linear model is based on the following simplifying assumptions: A disk refers to a linear sequence of pages and seek time is a constant. Both of these assumptions will be abolished in the disk model which is considered in the following.

The remainder of the paper is organized as follows. In section two, the disk model is introduced, the problem is stated and a simple algorithm is presented for computing efficient read schedules. In section three, the expected cost of the schedules is derived under the assumption that the required pages are located on one cylinder. The corresponding cost formula makes use of two recurrence relations. In section four, the analysis is extended to the case when the required pages are distributed over several cylinders. In section five, the problem is discussed how multi-page requests affect the response time of queries when multiple of them are performed at a time. Section six summarizes the results and concludes the paper.

2. PRELIMINARIES

In this section, a summary is first given on the most important assumptions of our disk model. The problem we address in the paper is then precisely stated. Finally, a simple algorithm is given for reading a set of disk pages.

2.1. The Disk Model

Magnetic disks are the most important storage medium to date. Recently, Ruemmler and Wilkes [16] have given an overview of the current state of the art in disk technology. In particular, they provided an excellent description on an accurate magnetic disk model that also serves as a guideline for the following model.

A magnetic disk is considered as a storage device with direct access to pages of data. In general, the I/O performance of queries is expressed in terms of page accesses assuming that the access cost is approximately equal for each page. In order to model disks more accurately we assume that the total cost of reading a page consists of three components: seek time, rotational delay and transfer time. Seek time denotes the time to move the disk arm to the desired cylinder. For today's disks, seek time can be modeled by a two-part function only depending on the distance from the starting cylinder to the target cylinder. The one part used for small seeks refers to a square-root function, whereas the other part refers to a linear function. After arriving at the target cylinder, the arm waits until the disk is rotated into the position such that the desired page appears under the arm. The corresponding time is called rotational delay. Finally, the page is transferred into main memory and the required time is known as transfer time.

Seek time and rotational delay depend on the previous position of the disk arm, whereas the transfer time is assumed to be constant for all pages on the disk. The transfer time of a page is taken as the cost unit and the cost is expressed in terms of page transfers. Suppose that page A has to be read from disk. Let B be the page where the read/write head was positioned on when the request for page A received service. The cost of reading page A is then given as

$$s(A,B) + r(A,B) + 1$$

where $s(A,B)$ and $r(A,B)$ denote the ratio of seek time and rotational delay to transfer time, respectively. Thus, a lower bound for reading a page is 1 (page transfer). For today's magnetic disks, seek time and rotational delay dominate transfer time, typically by a factor of about 15.

In order to make the analysis tractable, we simplify our disk model as follows: First, we do not consider the overhead of the disk controller (which is generally less than 1 *ms*) and channel contention (when multiple disks share a common disk controller). Second, head switch time is also not considered as part of the total cost. Third, we assume that the buffers available on the disk and in main memory are not used for query processing. Although Ruemmler and Wilkes [16] have shown that a buffer on the disk has a serious impact on the disk performance, it would have only a limited impact on the performance of *data-intensive* selection queries. In order to compute the result of those queries, various pages must to be read from disk, but each of these pages is retrieved and processed only once. This type of query is primarily considered throughout the paper.

The layout of pages on a disk is assumed to be as follows. First, each track of the disk contains PT pages where PT is an integer. Second, the starting positions of the tracks are aligned in a cylinder. Third, all of the pages on a disk can be used and therefore, remapping of references to bad pages is not considered in our approach. These assumptions are mild abstractions from the layout of pages found on today's magnetic disks. In particular, a cylinder of the disk can now be represented as a two-dimensional array $C_{i,j}$ of pages where $0 \leq i < TC$ and $0 \leq j < PT$. Here, TC denotes the number of tracks. For a cylinder $(C_{i,j})$, the set $\{C_{i,j} | 0 \leq i < TC\}$ is also called the j-th *column*, $0 \leq j < PT$. It follows that at most one page can be read from a column during a revolution of the disk. Since head switches are assumed to cause no time delay, an arbitrary page from column j can be read without any time delay when the previous page was read from column $(j-1)\ mod\ PT$.

2.2. Problem Statement

Consider a file F as a collection of pages of fixed size distributed over a magnetic disk drive. In order to perform a query on the file, it is assumed that pages $T_1, \ldots, T_N$ of the file must be read from the disk into main memory. These pages are called *target pages* and the set of target pages is called the *target set*. It is assumed that the complete target set is known in advance before actual retrieval of the pages begins. This occurs rather frequently in query processing, e.g. retrieval of pages supported by an index.

In the following, we deal with the problem of reading the target pages into main memory as fast as possible using one or several multi-page requests. A *multi-page request* transfers a set of pages from magnetic disk into main memory without interruption by other requests. In contrast to ordinary read commands, it is *not* necessary that the pages are contiguous on magnetic disk. In order to perform a multi-page request, a sufficiently large buffer is required where the target pages can be copied to. For the sake of simplicity, we first assume to have an infinitely large buffer, i.e. the buffer size is not of concern.

Let us first assume that the target set $\{T_1, \ldots, T_N\}$ is read by using one multi-page request. Let Π_N be defined as the set of all permutations on the vector $(1, \ldots, N)$. Then, a sequence $S = (T_{i_1}, \ldots, T_{i_N})$ is called a *read schedule* for the target set $\{T_1, \ldots, T_N\}$, if $(i_1, \ldots, i_N) \in \Pi_N$. A read schedule $(T_{i_1}, \ldots, T_{i_N})$ determines the order in which target pages are read from disk.

The selected schedule has a substantial impact on the cost for the multi-page request. The reason is that the cost for reading a target page $T_{i_{j+1}}$, $1 \leq j < N$, depends on the (disk) position of the target page T_{i_j} previously read. For a given target set $\{T_1, \ldots, T_N\}$, the goal is to find a schedule $(T_{i_1}, \ldots, T_{i_N})$ such that the total elapsed time of the multi-page request is minimized. Let T_0 be the page on the disk where the read/write head was positioned on when the multi-page request received service. The total elapsed time of a multi-page request is then given by

$$s(T_{i_1}, T_0) + \sum_{j=1}^{N-1} s(T_{i_{j+1}}, T_{i_j}) + r(T_{i_1}, T_0) + \sum_{j=1}^{N-1} r(T_{i_{j+1}}, T_{i_j}) + N$$

For a given schedule $S = (T_{i_1}, \ldots, T_{i_N})$, the elapsed time of a multi-page request can be decomposed into three components. Similarly to an ordinary single-page request, we call these

components seek time, rotational delay and transfer time of the schedule. The *seek time* of the schedule S is defined as

$$st(T_{i_1},\ldots,T_{i_N}) = s(T_{i_1},T_0) + \sum_{j=1}^{N-1} s(T_{i_{j+1}},T_{i_j})$$

The *rotational delay* is defined as

$$rd(T_{i_1},\ldots,T_{i_N}) = r(T_{i_1},T_0) + \sum_{\substack{1 \le j < N \\ s(T_{i_{j+1}},T_{i_j})>0}} r(T_{i_{j+1}},T_{i_j})$$

and the *transfer time* is given as

$$tt(T_{i_1},\ldots,T_{i_N}) = N + \sum_{\substack{1 \le j < N \\ s(T_{i_{j+1}},T_{i_j})=0}} r(T_{i_{j+1}},T_{i_j})$$

Let us emphasize that the transfer time of a multi-page request does not only correspond to the actual transfer time N, but it also includes the rotational delays of the pages which are read from the same cylinder as the predecessor page (i.e. no seek time occurs).

2.3. The Algorithm

For a given target set, the problem of computing an optimal schedule is NP-complete [22]. Therefore, heuristics are necessary to compute approximate schedules. We follow a common approach: First, the cylinders which contain at least one target page are accessed following the SCAN policy [9]. For each of these cylinders, an algorithm is used for reading all target pages from the cylinder in minimum time. The schedules produced by this simple algorithm are studied and their average-case performance is analyzed in this paper.

A remaining problem is then how to compute optimal schedules, when target pages are stored on one cylinder. Under the assumptions of our disk model, the *shortest-latency-time-first* (SLTF) policy [9, 19] gives an appropriate algorithm for computing such optimal schedules. Before going into more details, let us first introduce to some of our notations. Let $(C_{i,j})$, $0 \le i < TC$ and $0 \le j < PT$, be a cylinder where $N \le PT * TC$ of the pages are target pages. In the following, PC $(= PT * TC)$ denotes the number of pages on a cylinder. Furthermore, $C_{*,j}$, $0 \le j < PT$, denotes the set of target pages in the j-th column. Without loss of generality let us assume that the initial position of the disk arm is at the beginning of the track. The SLTF algorithm reads one target page from each of the non-empty sets $C_{*,j}$, $0 \le j < PT$, in a single disk revolution as long as one target page is unprocessed in $C_{*,j}$. In the last disk revolution required for reading the target pages, the algorithm stops reading after the read/write head has passed over the last column (i.e. farthest away from the beginning of the track) with the maximum number of target pages.

In [19] it was shown that the schedules produced by the algorithm are optimal, i.e. the target pages are read in minimum time. In order to differentiate (elapsed) time into transfer time and rotational delay, the algorithm is slightly modified. The algorithm does not start transferring target pages until the first set $C_{*,f}$, $0 \le f < PT$, is reached with the maximum number of target pages. Then, the transferring of pages is performed in the same manner as proposed originally. It will be stated in the following lemma that the schedule of the modified algorithm is also optimal.

Lemma 1 *Let $\{T_1,\ldots,T_N\}$, $N \le PC$, be the target set completely stored on a cylinder and let $(T_{i_1},\ldots,T_{i_N})$ be the schedule produced by the modified algorithm. Furthermore, let $C_{*,0},\ldots,C_{*,PT-1}$ denote the distribution of the target pages among the columns of the cylinder and let M be defined as $\max_{0\le i<PT}|C_{*,i}|$. Then, the schedule $(T_{i_1},\ldots,T_{i_N})$ is an optimal one. Its transfer time tt is given (in units of page transfers) as*

$$tt(T_{i_1},\ldots,T_{i_N}) = 1 + PT*(M-1) + \max\{j-i \mid |C_{*,j}| = |C_{*,i}| = M, 0 \le i,j < PT\}$$

Fig. 1: Target pages in a cylinder

and its rotational delay rd is given (in units of page transfers) as

$$rd(T_{i_1}, \ldots, T_{i_N}) = \min\{j \mid \ |C_{*,j}| = M, 0 \leq j < PT\}$$

Proof: Let M be the maximum number of target pages in a column of the cylinder and let $C_{*,f}$ and $C_{*,l}$, $0 \leq f < l < PT$, be the first and the last column with maximum number of target pages, respectively. The elapsed time of an optimal schedule is then given by $1 + PT * (M-1) + l$.

The elapsed time of a schedule produced by the modified algorithm corresponds to the sum of transfer time and rotational delay. The rotational delay is simply f. After moving the disk arm on position f, the "optimal" algorithm is used to read the the target pages. Since each of the first $f-1$ columns of the cylinder contains less than M target pages, the transfer time of the schedule is $1 + PT * (M-1) + l - f$. It follows that the schedule produced by the modified algorithm is also optimal. □

An example is illustrated in Figure 1. A cylinder consists of four tracks and eight columns. Target pages received the label 1, whereas other pages received the label 0. Overall, 10 pages have to be read from disk. The maximum occurs three times, at the second, fourth and fifth column. The rotational delay then corresponds to 2 page transfers and the transfer time corresponds to 12 page transfers.

In order to present an *average-case analysis* of the schedules produced by the algorithm, the problem is first restricted to the case where all target pages are on the same cylinder. The analysis for the more general problem, when pages are distributed over several cylinders, is presented in section 4. A summary of the most important notations used in the remainder of the paper is given in Table 1.

3. AVERAGE-CASE ANALYSIS OF MULTI-PAGE REQUESTS ON A CYLINDER

In this section, the case is studied when all target pages are stored on one cylinder ($C_{i,j}$), $0 \leq i < TC$ and $0 \leq j < PT$. We present an analysis on the average-case cost of the schedules produced by the (optimal) algorithm. The number of required rotations is determined by the maximum number M of elements in the sets $C_{*,j}$, $0 \leq j < PT$. Without loss of generality, the initial position of the disk arm is assumed to be in front of column 0, i.e. when the first page is read from column l, $0 \leq l < PT$, the rotational delay corresponds to l page transfers.

In the following, cost formulas are derived for the expected transfer time tt_X and the expected rotational delay rd_X. The analysis is made under the basic assumption that target pages are uniformly distributed among the pages of the cylinder. Thus, the formulas only vary in N, the number of target pages. In accordance with Lemma 1, it is sufficient to compute the expected value of M and the expected value of the position of the last column where the maximum is adopted. The analysis is therefore structured into two parts.

N	number of target pages
PT	number of pages on a track
TC	number of tracks per cylinder
PC	$= PT * TC$
$\{T_1, \ldots, T_N\}$	target set
$C_{i,j}$	denotes the pages of a cylinder, $0 \leq i < TC$ and $0 \leq j < PT$
$(C_{i,j})$	denotes a cylinder
$C_{*,j}$	denotes the set of target pages in the j-th column, $0 \leq j < PT$
M	maximum number of target pages in a column of a given cylinder
Cyl	number of cylinders
C_F	number of cylinders occupied by the file (file cylinders)
C_{act}	number of cylinders with at least one target page
tt_X	expected transfer time
rd_X	expected rotational delay
st_X	expected seek time
P, Q, R, H	various probability distributions

Table 1: List of symbols

First, the probability $P(N, PT, m)$ is computed that none of the sets $C_{*,j}$, $0 \leq j < PT$, contains more than m of the N target pages which are randomly selected from the PC pages. This probability allows the computation of the expected value of M using the following formula:

$$P(N, PT, 1) + \sum_{i=2}^{\min(N,TC)} ((P(N, PT, i) - P(N, PT, i-1)) * i$$

The formula can be simplified to

$$1 + \sum_{i=1}^{\min(N,TC)} (1 - P(N, PT, i)) \tag{1}$$

Second, we present a formula for computing the probability $Q(N, x, M)$ that the maximum M, $1 \leq M \leq TC$, occurs x times, $1 \leq x \leq PT$, assuming that N target pages are selected from the cylinder. Since every column can adopt the maximum with the same probability, the expected position of the last column with M target pages can easily be computed by using elementary probability theory.

In order to compute the probabilities $P(N, PT, m)$ and $Q(N, x, m)$ accurately, two recurrence relations are given in the following subsections. The expected rotational delay and the expected transfer time are then derived from the recurrence relations.

3.1. Recurrence Relation for Computing Probability P

Let $(C_{i,j})$ be a cylinder consisting of PT columns and TC tracks. Furthermore, let $F(n, p, m)$ be the number of possibilities in which n target pages, $1 \leq n \leq PC$, can be selected from p columns, $1 \leq p \leq PT$, of the cylinder such that at most m target pages, $1 \leq m \leq TC$, are taken from each of the p columns. It follows that the probability that none of the p columns contains more than m target pages is given as

$$P(n, p, m) = \frac{F(n, p, m)}{\binom{p*TC}{n}} \tag{2}$$

Suppose that n target pages, $n < p * TC$, have already been randomly selected from the p columns of the cylinder and each of the p columns delivers at most m target pages. Let the next target

page be randomly selected from the remaining $p * TC - n$ pages. The term $R(n+1, p, m)$ denotes the conditional probability that the $(n+1)$-st target page is selected from a column that contains less than m of the n previously selected target pages. Then $R(n+1, p, m)$ can be expressed as

$$R(n+1, p, m) = \frac{P(n+1, p, m)}{P(n, p, m)} \tag{3}$$

The $(n+1)$-st target page will produce a column that has delivered more than m target pages if and only if the target page is taken from a column with $TC - m$ remaining pages. The number of possibilities that a certain column contains m target pages is given as

$$\binom{TC}{m} * F(n-m, p-1, m)$$

Furthermore, the probability that this column receives the $(n+1)$-st target page is $\frac{TC-m}{p*TC-n}$. There are p columns and hence, the probability of the $(n+1)$-st target page taken from a column with $TC - m$ remaining pages can be expressed as

$$1 - R(n+1, p, m) = p * \frac{TC - m}{p * TC - n} * \frac{\binom{TC}{m} * F(n-m, p-1, m)}{F(n, p, m)} \tag{4}$$

By combining equations 2, 3 and 4, we obtain the following recurrence relation

$$P(n+1, p, m) = P(n, p, m) - \frac{p * (TC - m)}{p * TC - n} * \frac{\binom{TC}{m}\binom{(p-1)TC}{n-m}}{\binom{p*TC}{n}} P(n-m, p-1, m)$$

The recurrence relation can be simplified by using the following equality:

$$\frac{TC - m}{p * TC - n} * \frac{\binom{TC}{m}}{\binom{p*TC}{n}} = \frac{\binom{TC}{m+1}}{\binom{p*TC}{n+1}}$$

The recurrence relation is then given as

$$P(n+1, p, m) = P(n, p, m) - p * \frac{\binom{TC}{m+1}\binom{(p-1)TC}{n-m}}{\binom{p*TC}{n+1}} P(n-m, p-1, m) \tag{5}$$

In order to compute $P(N, PT, m)$ we need to calculate $P(n, p, m)$ for $n = m+1, \ldots, p * TC$ and $p = 1, 2, \ldots, PT$. The recursive relation is initialized by using the formula $P(n, p, m) = 1$ for $n \leq m$. Although the above recurrence relation looks complicated at first glance, the computation can be organized so that the evaluation of the next value of $P(n, p, m)$ requires only a few arithmetic operations. In particular, the product of the three binomial coefficients can also be computed in a recursive fashion.

To the best of the author's knowledge the computational problem of computing the probability P has not been solved so far. The problem is related to an urn problem which has been addressed by [15] in the context of hashing. Our problem is however different to the urn problem since the result of the n-th selection depends on the results of the previous selections.

3.2. Recurrence Relation for Computing Probability Q

Let $< n, x, m >$ be a three-tuple that denotes the state of the selection process after n target pages, $n < PC$, are read from the cylinder. The parameter m, $m \leq TC$, refers to the maximum number of target pages in a column of the cylinder after reading n pages, and x, $x \leq PT$, denotes the number of columns where m target pages can be found. Let $Q(n, x, m)$ be the probability of obtaining the state $< n, x, m >$ after distributing n target pages starting from state $< 1, 1, 1 >$. Thus, $Q(1, 1, 1) = 1$ and $Q(1, x, m) = 0$ for $m > 1$ and $x > 1$. Now, let us consider the situation after the $(n+1)$-st target page is randomly selected from the remaining $PC - n$ pages.

First, the case is discussed for obtaining a transition state $< n+1, x, m >$ with $x = 1$. There are two possible situations. In the first one the $(n+1)$-st page is selected from a column where the (current) maximum number of target pages can be found, i.e. there are $TC - m + 1$ remaining pages in the column. Hence, the probability of this event is $\frac{TC-m+1}{PC-n}$. Then, the value of m is incremented by 1 and only a single column adopts the maximum. In the second situation, $x = 1$ is already true before taking the $(n+1)$-st target page from the cylinder. If the $(n+1)$-st target page is selected from a column with more than $TC - (m-1)$ remaining pages, the value of m remains unchanged. Thus, there is still exactly one column with m target pages which are already transferred into main memory. Note that $R(n-m, x-1, m-1)$ refers to the probability that the $(n+1)$-st target page is in a column with more than $TC - (m-1)$ remaining pages. Apart from these situations, there is no way to obtain a transition state with $x = 1$. For $n \geq 1$ and $x = 1$, we obtain the following recurrence relation:

$$\begin{aligned} Q(n+1,1,m) &= \sum_{j=1}^{PT} j * \frac{TC-m+1}{PC-n} * Q(n,j,m-1) + \\ &\quad (1 - \frac{TC-m}{PC-n}) * Q(n,1,m) * R(n-m, PT-1, m-1) \end{aligned} \tag{6}$$

Second, let us consider a transition state $< n+1, x, m >$ for $x > 1$. This state can only be realized for the following two cases. In the first case, the original state has been $< n, x, m >$ and the $(n+1)$-st target page is selected from a column with more than $TC - (m-1)$ pages. In the second case, the original state has been $< n, x-1, m >$ and the $(n+1)$-st target page is taken from a cylinder with $TC - (m-1)$ target pages. The corresponding probability is given as $(1 - R(N+1-(x-1)*m, PT-(x-1), m-1))$. Then, the recurrence relation can be computed for $x > 1$ as

$$\begin{aligned} Q(n+1,x,m) &= (1 - x * \frac{TC-m}{PC-n}) * R(n+1-x*m, PT-x, m-1) * Q(n,x,m) + \\ &\quad (1 - (x-1) * \frac{TC-m}{PC-m}) * \\ &\quad (1 - R(n+1-(x-1)*m, PT-(x-1), m-1)) * Q(n,x-1,m) \end{aligned} \tag{7}$$

The probability that for a request of N pages the maximum occurs x times is then given as

$$\sum_{m=1}^{N} Q(N,x,m) \tag{8}$$

3.3. Expected Rotational Delay

In the previous subsection, the probabilities P and Q are derived as recurrence relations. In this subsection, probability Q is used for computing the expected rotational delay.

If the response set only consists of a single page, the expected rotational delay is half a revolution $(= PT/2)$. The same result holds when the maximum number of target pages is adopted only once. In general, the expected rotational delay refers to the expected distance from the beginning of a track to the *first* column with the maximum number of target pages. In order to obtain the expected distance, formula 8 is used for computing the probability that the maximum occurs i times, $1 \leq i \leq PT$. The property that the probability for realizing the maximum is the same for each of the columns can then be exploited.

Now, let us suppose that the maximum occurs x times, $1 \leq x \leq PT$. This corresponds to randomly selecting x out of PT columns. The distance to the *first* column can be computed by using elementary probability theory. There are

$$\binom{PT}{x}$$

possibilities to select x columns. Under the assumption that the j-th column, $j \geq x$, is the first one, there are

$$\binom{PT-j}{x-1}$$

possibilities to select the remaining $(x-1)$ columns such that each of them is behind the j-th column. Hence, the expected distance is given as

$$\sum_{j=1}^{PT-(x-1)} \frac{\binom{PT-j}{x-1}}{\binom{PT}{x}} * j \tag{9}$$

and consequently, the expected rotational delay is given as

$$rd_X(N) = \sum_{x=1}^{PT}\{\sum_{m=1}^{N} Q(N,x,m)\} * \{\sum_{j=1}^{PT-(x-1)} \frac{\binom{PT-j}{x-1}}{\binom{PT}{x}} * j\} - 1 \tag{10}$$

3.4. Expected Transfer Time

The transfer of pages from the disk starts when the first column, say $C_{*,f}$, $0 \leq f < PT$, with M target pages is under the disk arm. Then, $M-1$ disk revolutions are required for reading all pages from column $C_{*,f}$. If all other columns contain fewer target pages than column $C_{*,f}$, the transfer is completed. Otherwise, there is a column, say $C_{*,l}$, $f < l < PT$, which refers to the last column with M target pages. Then the transfer of pages continues until the disk arm has passed over the column $C_{*,l}$. For a multi-page request of N pages, the expected cost for the complete revolutions can easily be derived from formula 1. We yield

$$PT * \sum_{j=1}^{\min(N,TC)} (1 - P(N,PT,j)) \tag{11}$$

In order to obtain the expected transfer time, the expected distance between column $C_{*,l}$ and column $C_{*,f}$ has to be added to formula 11. Assume that the maximum is realized x times, $1 \leq x \leq PT$. Then, the expected distance to the first column with M target pages is already given by formula 9. In a similar way, the expected distance to the last column with M target pages can be computed as

$$\sum_{j=x}^{PT} \frac{\binom{j-1}{x-1}}{\binom{PT}{x}} * j = PT - \sum_{j=1}^{PT-(x-1)} \frac{\binom{PT-j}{x-1}}{\binom{PT}{x}} * j$$

Using formula 11, the following formula is obtained for the expected transfer time:

$$\begin{aligned} tt_X(N,p) \quad = \quad & 1 + PT * \sum_{j=1}^{\min(N,TC)} (1 - P(N,PT,j)) \\ & + \sum_{x=2}^{PT}\{\sum_{m=1}^{N} Q(N,x,m)\} * (PT - 2 * \sum_{j=1}^{PT-(x-1)} \frac{\binom{PT-j}{x-1}}{\binom{PT}{x}} * j) \end{aligned} \tag{12}$$

3.5. Discussion

In this subsection, we first briefly discuss the cost of computing the cost formulas. Next, the cost is discussed for reading N target pages from one cylinder.

Let us restrict our discussion to cost formula 12. Its calculation requires $O(PT*TC)$ summation steps in the worst-case. For each summation step, a complex function has to be computed in an iterative fashion. In order to reduce the number of iterative computations, we decided to

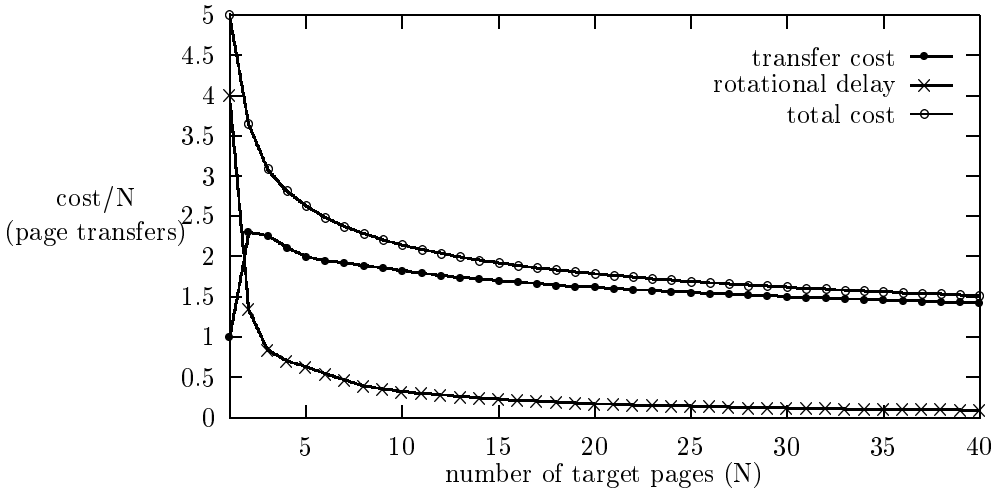

Fig. 2: Expected rotational delay and expected transfer cost ($TC = 20, PT = 8$)

precompute $\sum_{m=1}^{N} Q(N, x, m)$ for $x = 1, \ldots, PT$ and $N = 1, 2, \ldots$ and to store the results in a temporary table. The development of accurate approximations that completely avoid iterative computations is beyond the scope of this paper. This is however considered as an interesting subject for future research.

The cost formulas derived in the previous sections can directly be used to show the expected cost of multi-page requests. In Figure 2, the expected rotational delay, the transfer cost and the sum of both are plotted as a function of the number of target pages. The cylinder consists of 20 tracks, each of them containing 8 pages ($TC = 20, PT = 8$). The cost is expressed as the expected cost per target page.

For $N = 1$ (i.e. there is only one target page), we obtain the well-known result that the expected rotational delay is half a rotation. As expected, the cost per target page decreases with an increasing number of target pages. For $N = 10$, the cost per target page is only 2 page transfers. For larger values of N, the cost per target page approaches the minimum of one page transfer.

When pages are read in random order, the cost per target page would be the same as reading a single target page from the cylinder (5 page transfers in the example). Even if the target pages are already clustered on a single cylinder, the graph demonstrates the importance of reading the target pages in the order given by an optimal schedule.

4. A GLOBAL COST FUNCTION

In this section, a cost function is presented for reading target pages which are distributed over several cylinders of the disk. Thus, the cost function has to consider seek time as a major component of the total cost.

In a scenario where a query is performed on a file, it is very unlikely that target pages are uniformly distributed over the disk. On the contrary, it is expected that the underlying file system clusters a file on a few cylinders. Accordingly, target pages would be distributed on a few cylinders. In order to take into account the effect of clustering, the cost function does not only depend on the number of target pages, but also on the cylinders on which the pages of the file are distributed. A cylinder that contains a page of the file is called a *file cylinder* in the following.

In order to make the analysis tractable, the simplifying assumptions are made that the file

cylinders are distributed uniformly over the disk and that the target pages are uniformly distributed over the file cylinders. Moreover, we assume that the initial position of the disk arm is at the outermost cylinder. The algorithm for reading the pages visits only those file cylinders where a target page can actually be found in linear order from outside to inside. A file cylinder where a target page is located is called a *target cylinder*.

First, a cost formula is given for the required seek operations. Note that a seek consists of several components. First, the arm is accelerated until it reaches half the seek distance or a constant velocity. In the latter case, the arm coasts across some cylinders at its maximum speed. Third, the arm decelerates its speed and stops on top of the desired track. Therefore, the seek time can be well modeled as a square root function for small seeks and as a linear function for large seeks. The *seek time of a multi-page request* is simply the sum of the seek time required to move from one target cylinder to the next target cylinder. The distribution of target cylinders is determined through the number of target pages and the number of file cylinders. The exact distribution is again given as a recurrence relation. The analysis of the seek cost is first based on the exact distribution of target cylinders. Thereafter, an approximate formula is derived for the expected seek cost which avoids the computation of the recurrence relation. This formula produces results which are very close to the ones obtained from the exact formula.

Let Cyl be the number of cylinders on the disk. For a file F, the parameter C_F denotes the number of file cylinders. Consider N target pages randomly distributed over C_F file cylinders. Let $< n, j >$, $n \geq 1, 1 \leq j \leq C_F$, refer to a transition state where j of the C_F file cylinders are target cylinders. Furthermore, let $S(n, j)$ be the probability of realizing the transition state $< n, j >$. Note that $S(1, 1) = 1$ and $S(1, j) = 0$, $2 \leq j \leq C_F$. For $n > 1$, two situations are distinguished: either the n-th target page is on a cylinder which already contains one of the other $n - 1$ target pages, or the n-th target page is on an "empty" cylinder that does not contain any of the other target pages. Then the probability S can be computed for $n > 1$ by using the following recurrence relation:

$$S(n, j) = \frac{(C_F - (j-1)) * PC}{C_F * PC - (n-1)} S(n-1, j-1) + \frac{j * PC - (n-1)}{C_F * PC - (n-1)} S(n-1, j) \qquad (13)$$

Note that this recurrence relation is actually a special case of the formulas 6 and 7.

Let us consider that C_{act} cylinders are target cylinders, i.e. they contain at least one of the target pages. Recall that the disk arm is assumed to be on the first cylinder and that target cylinders are uniformly distributed. The first cylinder is a target cylinder with probability C_{act}/Cyl. In general, the probability that the first target cylinder is the k-th cylinder on the disk, $1 \leq k \leq Cyl - C_{act} + 1$, is given as

$$\frac{\binom{Cyl-k}{C_{act}-1}}{\binom{Cyl}{C_{act}}}$$

Assume that the disk requires $seek\_time(i)$ to pass over i cylinders. When the first cylinder is a target cylinder, the expected cost to move from one target cylinder to another is given by $H(Cyl - 1, C_{act} - 1)$. The function H is computed as

$$H(m, l) = \sum_{k=1}^{m-l+1} \frac{\binom{m-k}{l-1}}{\binom{m-1}{l}} * seek\_time(k)$$

When the first cylinder is not a target cylinder, the expected cost to move from one target cylinder to another is given as $H(Cyl - 1, C_{act})$. This event occurs with probability $1 - C_{act}/Cyl$. Because target cylinders are uniformly distributed, the expected distance between two target cylinders is independent of their actual position. Under the assumption of j target cylinders the expected cost for the seek operation is then given as

$$G_{Cyl}(j) = \frac{j}{Cyl} * (j-1) * H(Cyl-1, j-1) + (1 - \frac{j}{Cyl}) * j * H(Cyl-1, j)$$

Finally, the expected seek time st_X can be obtained by combining the formulas in the following way:

$$st_X(N, C_F) = \sum_{j=1}^{min(N,C_F)} S(N, j) * G_{Cyl}(j) \tag{14}$$

When the target set is distributed over several cylinders, the rotational delay and transfer time of the multi-page request is defined as the sum of the rotational delays and transfer times which occur on the target cylinders, respectively. The expected rotational delay and the expected transfer time can be approximately computed under the assumption that each of the target cylinders receives target pages independently of each other, i.e. the values computed for the one cylinder has no effect on the values of anther cylinder. Then, a target cylinder receives j of the N pages with probability

$$\binom{N}{j} (\frac{1}{C_F})^j * (1 - \frac{1}{C_F})^{N-j}$$

The formulas 14, 12 and 10 can be combined for computing the total expected cost of reading N target pages distributed on C_F cylinders. They yield

$$Cost(N, C_F) = st_X(N, C_F) + \sum_{j=1}^{N} \{ \binom{N}{j} (\frac{1}{C_F})^j * (1 - \frac{1}{C_F})^{N-j} * (tt_X(j) + rd_X(j)) \} \tag{15}$$

4.1. An Approximation for the Expected Seek Time

The computation of probability S using the recurrence relation, see formula 13, is time consuming when the number of file cylinders and the number of target pages is high. In this subsection, an approximate formula is given for the seek cost which is based on the expected number of target cylinders. The (exact) expected number of target cylinders can iteratively be computed by using a formula that was independently derived by Waters [20] and Yao [23]. The formula was originally proposed for a problem in a different context. In order to avoid iterative computations several approximations of the formula have been proposed in the literature [3, 20, 10]. In [10], an upper bound and a lower bound of the formula were derived that are both accurate approximations. By using the upper bound, the number of target cylinders is approximately given as

$$x = C_F * (1 - (1 - \frac{1}{C_F})^{\frac{N}{2}} * (1 - \frac{PC}{PC * C_F - N + 1})^{\frac{N}{2}})$$

Since x is not an integer, $\lfloor x \rfloor$ and $\lceil x \rceil$ are used for computing a linear combination. Eventually, the approximate formula is given as

$$st_X(N, C_F) \simeq (x - \lfloor x \rfloor) * \lceil x \rceil * G_{Cyl}(\lceil x \rceil) + (\lceil x \rceil - x) * \lfloor x \rfloor * G_{Cyl}(\lfloor x \rfloor) \tag{16}$$

In [14] the formula $x * seek\_time(Cyl/x)$ has been proposed for approximating the seek cost. This simple approximation also gives accurate results (relative errors of about 2% and less) if the number of target pages is high. However, for a small number of target pages, the relative error of the seek cost is too high. In contrast, formula 16 still produces accurate results (with a relative error less than 1%) in such a situation.

4.2. Discussion

In order to demonstrate that the cost function (see formula 15) is indeed accurate, we compared the results of the cost functions with the ones obtained from simulations. The underlying disk parameters correspond to the one of the Fujitsu Eagle disk (see Table 2) which is a rather old disk, but is frequently used in various experimental comparisons. Note that the timings of Table 2 are given in page transfers. Moreover, the seek time of the Fujitsu Eagle is well approximated by using the following function [18]:

$$seek\_time(x) = \begin{cases} 0 & \text{if } x = 0 \\ 2.3 + 4.35\sqrt{x} & \text{if } x \leq 239 \\ 9 + 0.14(x - 239) & \text{if } x > 239 \end{cases} \tag{17}$$

Cyl	840
TC	20
PT	8
page size [KB]	4
average seek time	9
average rotational delay	4

Table 2: Specification of the Fujitsu Eagle disk

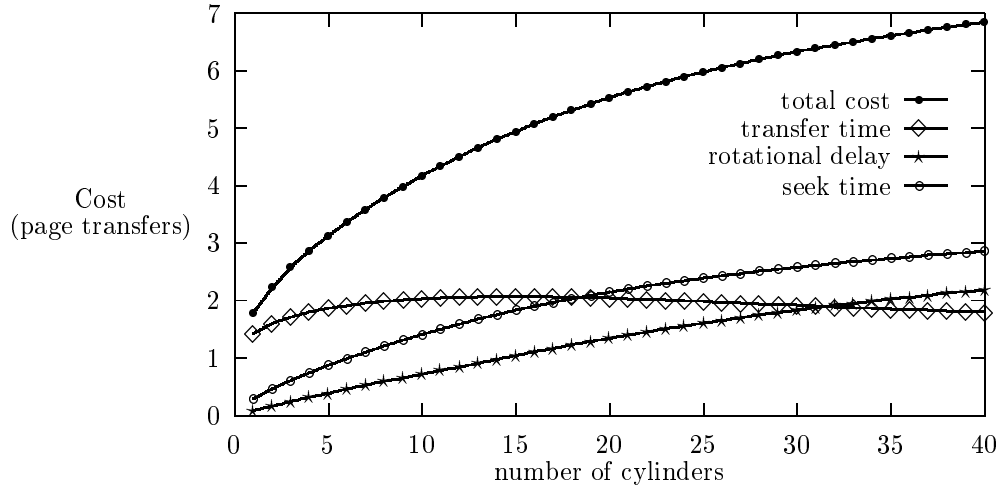

Fig. 3: Cost function (varying in number of cylinder, constant number of 40 pages)

For one of our experiments, the results of the cost functions and the results of the simulations are reported in Table 3. In this experiment, the number of target pages is fixed ($N = 40$), whereas the number of file cylinders is varying from one to forty. The cost is decomposed into three components: transfer cost, rotational delay and seek cost. For each of these cost components, we report the results of the cost function, the results of the simulation and additionally, the relative error given in percent. As demonstrated in Table 3, the relative errors are less than 1% for each of the cost components.

In Figure 3, the expected cost is plotted for the three cost components as a function of the number of file cylinders. Additionally, a fourth curve shows the total cost per target page given as the sum of the three cost components. The cost is measured in units of page transfers. As expected, the total cost increases with an increasing number of file cylinders. Best performance is obtained when the file is kept only on a single cylinder. Reading a target page in this case only requires less than two page transfers. In general, it is more likely that files will consist of a large number of pages and therefore the number of file cylinders has to be greater than one. For example, let us consider that target pages are distributed on 5 cylinders. Then the approach of reading all target pages in a single multi-page request requires only three page transfers (per target page).

Let us compare these results to the ones obtained when target pages are read using a random schedule, i.e the target pages are read in a random sequence without considering their physical position on disk. Then, when the disk arm is already on the desired cylinder, 5 page transfers

cylinder	rotational delay			transfer time			seek time		
	model	simu.	error	model	simu.	error	model	simu.	error
1	1.421	1.425	0.287	0.087	0.087	0.083	0.284	0.285	0.371
2	1.608	1.608	0.013	0.169	0.168	0.789	0.464	0.463	0.086
3	1.722	1.717	0.302	0.247	0.247	0.136	0.613	0.613	0.058
4	1.803	1.804	0.047	0.321	0.322	0.070	0.748	0.748	0.036
5	1.864	1.865	0.053	0.394	0.392	0.509	0.873	0.874	0.106
6	1.912	1.914	0.067	0.465	0.464	0.154	0.992	0.992	0.072
7	1.952	1.956	0.171	0.532	0.532	0.101	1.105	1.105	0.045
8	1.986	1.992	0.308	0.597	0.599	0.328	1.212	1.213	0.033
9	2.013	2.011	0.101	0.660	0.660	0.089	1.315	1.315	0.003
10	2.034	2.038	0.215	0.724	0.723	0.167	1.413	1.413	0.042
11	2.050	2.051	0.077	0.788	0.789	0.043	1.505	1.505	0.025
12	2.061	2.063	0.112	0.852	0.847	0.590	1.593	1.594	0.078
13	2.067	2.069	0.114	0.917	0.916	0.065	1.676	1.677	0.064
14	2.070	2.070	0.039	0.981	0.980	0.090	1.755	1.758	0.133
15	2.069	2.073	0.198	1.044	1.045	0.086	1.830	1.833	0.159
16	2.066	2.066	0.039	1.107	1.108	0.091	1.900	1.903	0.143
17	2.060	2.063	0.125	1.169	1.170	0.093	1.967	1.971	0.208
18	2.053	2.058	0.249	1.229	1.233	0.309	2.030	2.032	0.103
19	2.044	2.045	0.046	1.288	1.289	0.063	2.090	2.093	0.140
20	2.034	2.033	0.071	1.346	1.348	0.177	2.146	2.146	0.006
21	2.024	2.025	0.079	1.402	1.404	0.157	2.200	2.204	0.178
22	2.012	2.012	0.014	1.456	1.457	0.083	2.251	2.255	0.180
23	2.000	2.000	0.027	1.509	1.510	0.046	2.300	2.301	0.053
24	1.988	1.991	0.175	1.560	1.558	0.142	2.346	2.348	0.095
25	1.975	1.975	0.023	1.610	1.617	0.407	2.390	2.396	0.261
26	1.962	1.963	0.064	1.658	1.661	0.145	2.432	2.437	0.198
27	1.949	1.951	0.105	1.705	1.707	0.134	2.472	2.475	0.137
28	1.936	1.935	0.050	1.750	1.754	0.227	2.510	2.514	0.156
29	1.923	1.924	0.043	1.794	1.793	0.018	2.546	2.548	0.088
30	1.910	1.910	0.001	1.836	1.837	0.080	2.581	2.585	0.163
31	1.898	1.898	0.034	1.877	1.878	0.101	2.614	2.618	0.143
32	1.885	1.882	0.153	1.916	1.922	0.285	2.646	2.653	0.254
33	1.873	1.872	0.044	1.954	1.951	0.168	2.677	2.681	0.142
34	1.860	1.857	0.197	1.991	1.994	0.144	2.706	2.709	0.102
35	1.848	1.848	0.041	2.027	2.031	0.198	2.734	2.739	0.191
36	1.836	1.836	0.026	2.062	2.063	0.038	2.761	2.763	0.075
37	1.825	1.822	0.167	2.095	2.097	0.058	2.787	2.791	0.131
38	1.813	1.808	0.266	2.128	2.132	0.192	2.812	2.819	0.257
39	1.802	1.801	0.077	2.159	2.160	0.033	2.836	2.841	0.161
40	1.791	1.792	0.033	2.190	2.187	0.156	2.859	2.861	0.073

Table 3: A comparison of results obtained from simulations and the cost function ($N = 40$)

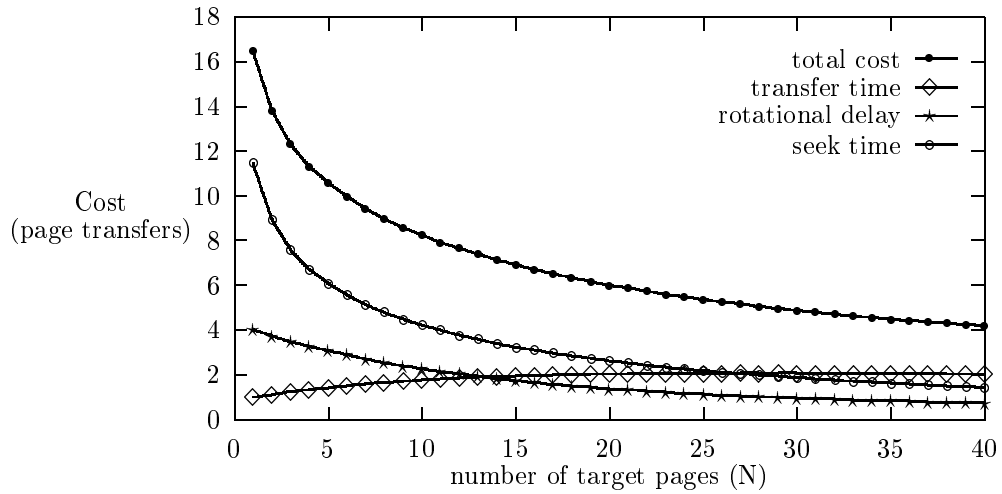

Fig. 4: Cost function (varying number of pages, constant number of 10 cylinders)

will be required on the average for reading a page from the cylinder. Otherwise, the disk arm has to move to the desired cylinder first. The average seek time of the Fujitsu Eagle disk is roughly 9 page transfers. The total time is then 14 page transfers for reading a page. The probability is approximately 1/5 that a target page is read from the same file cylinder as the previous target page. Thus, the expected cost for reading a page is about 12 page transfers. In comparison to reading target pages using the proposed schedule, a random schedule is less efficient by a factor of four. However, the advantage of using efficient schedules decreases with an increasing number of file cylinders. For example, when 40 target pages are randomly distributed over 40 file cylinders, the cost of reading a target page corresponds to 7 page transfers, i.e. it is still half the cost of reading pages randomly.

As depicted in Figure 3 the total cost depends on its three components as follows. Transfer time is the dominant component for $C_F \leq 18$. For $C_F > 18$, the seek cost is higher than the transfer cost. For a large number of file cylinders, transfer cost can even be lower than the rotational delay.

In Figure 4, the cost is illustrated for a fixed number of file cylinders ($C_F = 10$) and a varying number of target pages. As long as the number of target pages is low, the performance gain of reading target pages using the proposed schedules is fairly low in comparison to random schedules. However, the total cost per target page constantly decreases with an increasing number of target pages.

5. MULTIPLE QUERIES

In the previous sections, the performance of multi-page requests is examined under the assumption that the disk belongs exclusively to one query. This is typically true for a lowly loaded or a batch-oriented system. In a heavily loaded system where several queries are concurrently processing, it is not acceptable that the disk arm belongs to a single query from beginning to end. Otherwise, a huge request may defer the execution of small requests (which may require only a few pages) such that their response time becomes unacceptably high. However, the primary goal in a heavily loaded system is not to improve the response time of an individual query, but to improve the average response time of queries under a given system load (throughput). The throughput is defined for a given mix of queries by the arrival rate of queries per second. The question arises

Fig. 5: The implementation of the simulation

how the policy of multi-page requests influences the average response time when multiple queries are concurrently processed at a time? This question is discussed under the assumption that the response time of a query is determined by its I/O time.

In the following, we report the results of a preliminary performance comparison of multi-page requests and ordinary single-page requests when multiple queries are processed concurrently. All results are obtained from simulations which are based on our disk model and on the corresponding cost function. The disk used in our simulations refers to a Fujitsu Eagle whose seek time corresponds to formula 17. The cylinder time (i.e. the time spent on the cylinder) is obtained as the sum of formula 12 and formula 10 derived in section 3. For a request of one target page, the cylinder time is then roughly $10.3\,ms$.

The implementation of the simulation is illustrated in Figure 5. A process *CreateQ* creates the queries with respect to a given query profile. A query sends multi-page requests (or single-page requests) to the disk one by one. A multi-page request contains all target pages of the query which belong to the same cylinder of the disk. Since a multi-page request is processed without interruption by other requests, the disk gives services to a query as long as target pages are still unprocessed on the cylinder where the disk arm is currently positioned. We assume that the query is blocked while the disk satisfies one of its requests. Several multi-page requests of different queries are waiting in front of the disk in a request queue to receive service from the disk. The circular SCAN (C-SCAN) [8] scheduling strategy is used for selecting the next request which receives service from the disk. This approach to query processing has basically three advantages. First, it prevents a multi-page request from occupying the disk for a very long time period. Second, the size of the buffer required for a multi-page request is limited by the capacity of a cylinder. Third, any of the well-known disk scheduling policies [8] can be used for organizing the request queue.

A *query profile* is characterized by one or several query types. Each query type is described by the following parameters: the number of target pages (N) and the number of (file) cylinders (C_F). We assume that the target pages are uniformly distributed over the file cylinders and that the file cylinders are uniformly distributed among the cylinders of the disk. For each query type, the probability of selecting a query of this type has to be specified. In addition to the query types, a query profile is characterized by the arrival rate of the queries. In the following, we assume that the arrivals of queries follow an exponential distribution where λ denotes the average number of queries which arrive in one second. The assumption of an exponential distribution is rather common for this kind of simulations and therefore, it is also used in our experiments. Overall, a query profile QPF can be described by the following list of parameters

$$QPF = (\lambda, pb_1, QT_1(N_1, C_1), \ldots, pb_k, QT_k(N_k, C_k))$$

where λ denotes the average number of queries per second and QT_i is the i-th query type which delivers queries with probability pb_i, $1 \leq i \leq k$. The parameters N_i and C_i denote the number of

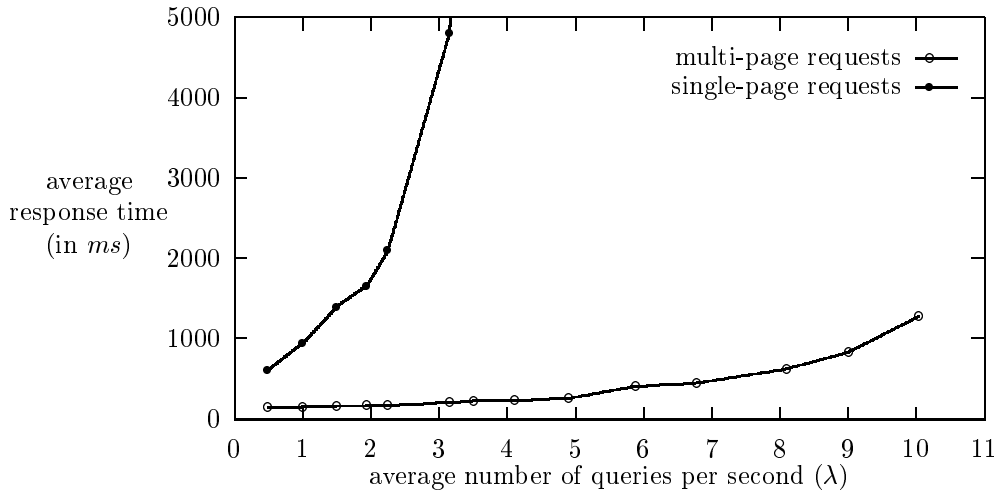

Fig. 6: Results of query profiles $QPF_1(\lambda)$ varying in λ

target pages and the number of file cylinders of query type QT_i.

In our first set of experiments, we investigated query profiles

$$QPF_1(\lambda) = (\lambda, 1.0, QT(20,5))$$

where parameter λ is varying in the range between 0.5 and 10. In Figure 6 the average response time (expressed in *ms*) is depicted. The one curve refers to the policy when target pages are read using ordinary read requests, whereas the other curve depicts the results when multi-page requests are used. For $\lambda = 1$, 20 target pages on the average are required to be read from disk. For $\lambda = 0.5$, the queuing delay does not have much influence on the response time of the queries and therefore, these results correspond to the ones obtained when queries are performed one by one. As depicted in Figure 6, when queries are performed using single-page requests the average response time per query dramatically increases for $\lambda > 3$, whereas multi-page requests still offer low response time of the queries. Only for $\lambda > 10$, response times are not acceptable anymore for multi-page requests.

In the second set of experiments, query profiles

$$QPF_2(\lambda) = (\lambda, 0.5, QT(1,1), 0.4, QT(20,5), 0.1, QT(100,10))$$

are considered, $0.5 \leq \lambda \leq 14$, each of them containing three different query types. A query that requires only one page is assumed to occur with probability 50%, whereas the probability of a query that requires 100 target pages is only 10%. The average response time of queries of query profile QPF_2 shows almost the same characteristics as the ones observed for query profile QPF_1.

One serious problem of using multi-page requests might be that the multi-page requests of data-intensive queries occupy the disk for a very long time period such that the response time of small queries which require only one page (or a few pages) increases substantially. An interesting question is therefore how the average response times of small queries develops when all queries are performed using multi-page requests. In order to answer this question for our query profiles $QPF_2(\lambda)$, the average response time of the queries of type $QT(1,1)$ is plotted in Figure 7. The response time of a query of type $QT(1,1)$ obviously cannot take advantage from multi-page request. Therefore, both graphs show similar results for a lowly loaded system (i.e. small λ). On a heavily loaded system, however, the average response time of these queries is considerably lower when multi-page requests are used in comparison to using single-page requests.

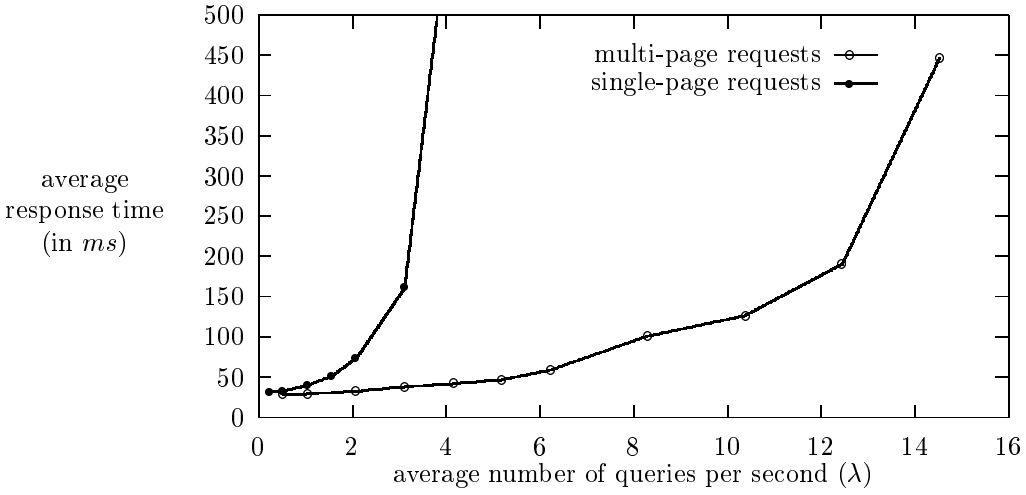

Fig. 7: Average response time queries from type $QT(1,1)$ of profile $QPF_3(\lambda)$

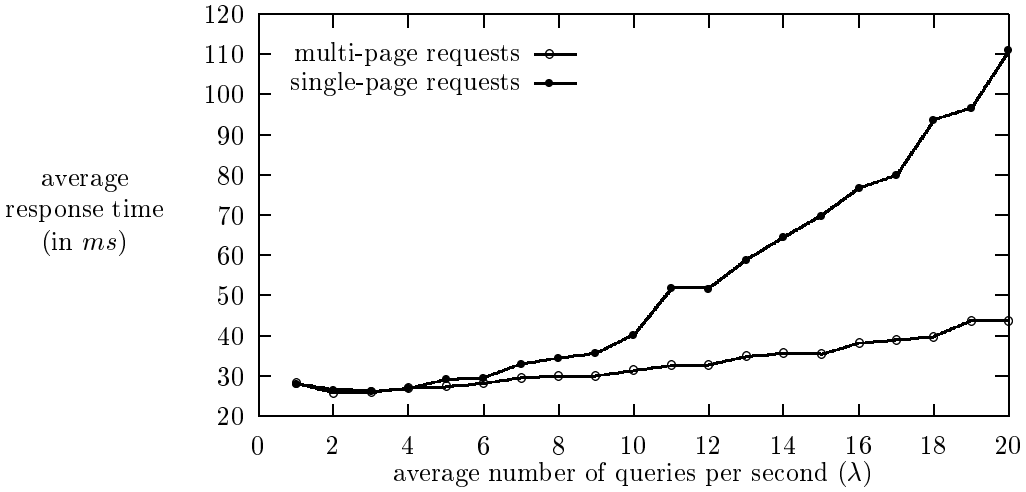

Fig. 8: Average response time of queries from type $QT(1,1)$ of profile $QPF_3(\lambda)$

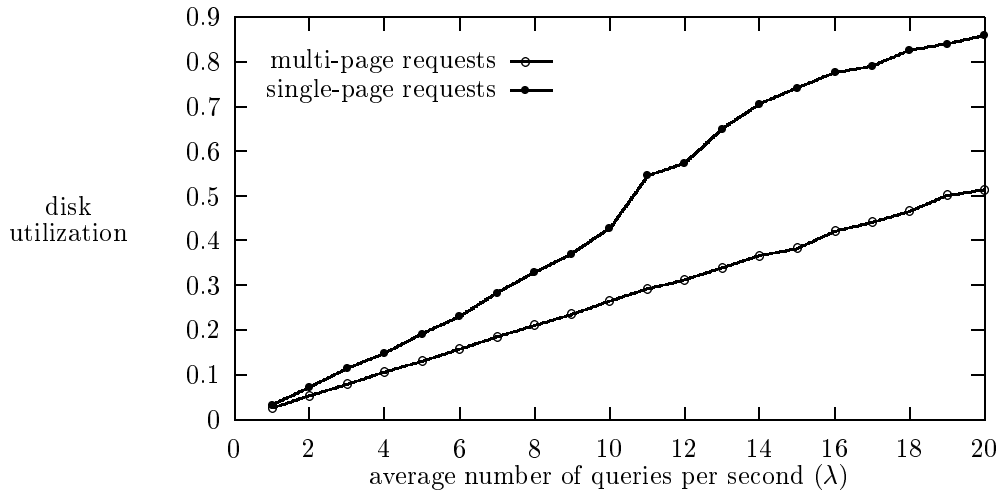

Fig. 9: Disk utilization for profile $QPF_3(\lambda)$

In a third set of experiments, we examined query profiles

$$QPF_3(\lambda) = (\lambda, 0.9, QT(1,1), 0.1, QT(20,1))$$

for $\lambda = 1, \ldots, 20$. These inhomogeneous profiles are characterized by two completely different query types. Most of these queries (90 %) require only one page from disk, whereas the remaining queries (10 %) read 20 target pages from one cylinder. The average response time of the queries of type $QT(1,1)$ is plotted in Figure 8. Two curves show the results when multi-page requests and single-page requests are used, respectively. A similar trend can be observed for profile QPF_3 as for QPF_2. For a lowly loaded system, there is almost no performance difference, whereas for a highly loaded system queries of type $QT(1,1)$ greatly benefit from exploiting multi-page requests.

One of the reasons that explains the clear superiority of multi-page requests over single-page requests is related to the disk utilization. The disk utilization will be considerably lower when multi-page requests are used for processing queries in comparison to using single-page requests. This is illustrated in Figure 9 where the disk utilization is depicted for query profiles $QPF_3(\lambda), 1 \leq \lambda \leq 20$. A lower disk utilization results in shorter request queues and therefore, the queuing time of a request also decreases. The queuing time is however an important part of the total cost in case of a heavily loaded system. Overall, we conclude that multi-page requests do not only reduce the response time of data-intensive queries on a heavily loaded system, but also of those queries which require only a few pages from disk.

6. CONCLUSION

In this paper, we studied the performance of data-intensive queries which require a set of target pages from a file scattered on magnetic disk. Assuming that the set of target pages is known before actual retrieval begins, we investigated the impact of multi-page requests on the response time of a query for two different situations: First, the query was exclusively performed in the system and second, other queries queries were concurrently performed at the same time. In contrast to ordinary (single) page requests, a multi-page request retrieves a set of disk pages (not necessarily contiguous) without interruption by other requests. The cost of a multi-page request is influenced considerably by the schedule which defines the order of retrieving the pages from disk.

First, we studied the problem of improving the performance of an individual query by using one multi-page request. A simple algorithm was first introduced for computing the corresponding schedule. Next, the average-case performance of the query was analyzed by using a pure analytical model. As a result, we derived a cost function for the expected cost of reading a set of disk pages. The cost function varies in the number of required pages and in the number of cylinders where pages of the file are kept. The cost function illustrates the dependency between cost, clustering of data, and scheduling policies. In particular, we found that although the required pages are clustered on a single cylinder the cost for reading the pages can be reduced considerably when an efficient read schedule is used. Moreover, the cost function also shows the relationship between its three components such as seek time, rotational delay and transfer time.

In a case study where the Fujitsu Eagle was assumed to be the underlying disk, we found that multi-page requests can reduce by a factor of up to 7 the cost of queries in comparison to reading pages one at a time. In contrast to ordinary (single) page requests, transfer time dominated seek time and rotational delay for many queries in our experiments.

Moreover, we also studied the impact of multi-page requests on the response time of a query when multiple queries were concurrently processed at a time. Results obtained from an experimental performance comparison showed that for a high system load the average response time can be lower by several factors when multi-page requests are used in comparison to reading pages one at a time. In particular, we observed in our experiments that small queries (which may require only one page) do not suffer under the optimization of data-intensive queries. Despite our suppositions to the contrary, substantial performance improvements can be achieved for small queries when the system load is sufficiently high.

In order to obtain our analysis several simplifying assumptions have to be made with respect to the distribution of files, the distribution of target pages and the architecture of the disk:

- A file is assumed to be distributed over some cylinders which are randomly selected from the disk.
- Target pages are assumed to be randomly selected from the pages of the file. This assumption is fulfilled for many queries in a DBMS.
- The disk is considered as a sequence of cylinders, each of them is modeled as a two-dimensional array of pages. The one dimension refers to the track and the other refers to the position on the track where a page is stored. The most restrictive assumption of our disk model is that head-switch time is not considered so far. Although head switch time is not expected to dominate the total cost, it might have an important impact.

Our current and future work in the area focuses on three major tasks. First, we are currently working in extending the analysis to the case of incorporating head switch time in the disk model. In particular, we are interested in (approximate) cost functions whose evaluation avoids expensive iterative computations. Second, the approach of multi-page request is being applied to several queries typically occurring in spatial data base systems. In particular, window queries, spatial joins and the transfer of large spatial objects are operations which may benefit from using multi-page requests. Third, we are interested in exploiting multi-page requests for modern storage devices such as two-headed disk systems, disk arrays, magneto-optical disks and optical disks.

ACKNOWLEDGEMENT

I want to thank the anonymous referees for their very helpful suggestions that led to a great improvement of the paper.

REFERENCES

[1] A. Biliris. The performance of three database storage structures for managing large objects. In *Proc. of the ACM SIGMOD*, pp. 276–285 (1992).

[2] T. Brinkhoff and H. P. Kriegel. The impact of global clustering on spatial database systems. In *Proc. of the Conf. on Very Large Databases (VLDB)* (1994).

[3] A. F. Cardenas. Analysis and performance of inverted data base structures. *Communications of the ACM*, **18**(5):253–263 (1975).

[4] J. Cheng, D. Haderle, R. Hedge, B. Iyer, T. Messinger, C. Mohan, and Y. Wang. An efficient hybrid join algorithm: A DB2 prototype. In *Proc. of the IEEE Conference on Data Engineering*, pp. 171–180 (1991).

[5] J. M. Cheng, C. R. Loosley, A. Shibamiya, and P. S. Worthington. IBM Database 2 performance: design, implementation, and tuning. *IBM System Journal*, **23**(2):189–210 (1984).

[6] E. I. Cohen, G. M. King, and J. T. Brady. Storage hierarchies. *IBM System Journal*, **28**(1):62–76 (1989).

[7] George Copeland, Tom Keller, Ravi Krishnamurthy, and Marc Smith. The case for safe RAM. In *Proc. of the Conf. on Very Large Databases (VLDB)*, pp. 327–335 (1989).

[8] H. M. Deitel. *Operating Systems.* Addison-Wesley (1990).

[9] P. J. Denning. Effects of scheduling on file memory operations. In *Proc. AFIPS*, pp. 9–21 (1967).

[10] G. Diehr and A. Saharia. Estimating block accesses in database organizations. *IEEE Transactions on Knowledge and Data Engineering*, **6**(3):497–499 (1994).

[11] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufman (1993).

[12] A. Hutflesz, H.-W. Six, and P. Widmayer. Globally order preserving multidimensional linear hashing. In *Proc. of the IEEE Conference on Data Engineering*, pp. 572–579 (1988).

[13] J. Menon and J. Cortney. The architecture of a fault-tolerant cached RAID controller. In *Proc.IEEE*, pp. 76–86 (1993).

[14] C. A. Polyzois, A. Bhide, and D. Dias. Disk mirroring with alternating deferred updates. In *Proc. of the Conf. on Very Large Databases (VLDB)*, pp. 604–617 (1993).

[15] M. V. Ramakrishna. Computing the probability of hash table / urn overflow. *Comm. in Statistics - Theorey and Methods*, **16**(11):3343–3353 (1987).

[16] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, **27**(3):17–28 (1994).

[17] B. Seeger, P.-Å. Larson, and R. McFadyen. Reading a set of disk pages. In *Proc. of the Conf. on Very Large Databases (VLDB)*, pp. 592–603 (1993).

[18] M. Seltzer, P. Chen, and J. Ousterhout. Disk scheduling revisited. In *Proceedings of the 1990 Winter USENIX Conference*, pp. 313–323 (1990).

[19] H. S. Stone and H. Fuller. On the near-optimality of the shortest-latency-time-first drum scheduling discipline. *Communications of the ACM*, **16**(6):352–353 (1973).

[20] S. J. Waters. Hit ratio. *Computer Journal*, **19**(1):21–24 (1976).

[21] G. Weikum. Set-oriented disk access to large complex objects. In *Proc. of the IEEE Conference on Data Engineering*, pp. 426–433 (1989).

[22] C. K. Wong. Minimizing expected head movement in one-dimensional and two-dimensional mass storage systems. *Computer Surveys*, **12**(2):167–177 (1980).

[23] S. B. Yao. Approximating block accesses in database organizations. *Communications of the ACM*, **20**(4):260–261 (1977).