# Algorithms for the Database Layout Problem

Gagan Aggarwal * Tomás Feder † Rajeev Motwani ‡ Rina Panigrahy § An Zhu ¶

February 27, 2004

**Abstract**

We present a formal analysis of the database layout problem, i.e., the problem of determining how database objects such as tables and indexes are assigned to disk drives. Optimizing this layout has a direct impact on the I/O performance of the entire system. The traditional approach of striping each object across all available disk drives is aimed at optimizing I/O parallelism; however, it is suboptimal when queries co-access two or more database objects, e.g., during a merge join of two tables, due to the increase in random disk seeks. We adopt a model proposed elsewhere, which takes into account both the benefit of I/O parallelism and the overhead due to random disk accesses, in the context of a query workload which includes co-access of database objects. The resulting optimization problem is intractable in general and we employ techniques from approximation algorithms to present provable performance guarantees. We show that while optimally exploiting I/O parallelism alone suggest uniformly striping data objects (even for heterogeneous files and disks), optimizing random disk access alone would assign each data object to a single disk drive. This confirms the intuition that the two effects are in tension with each other. We provide approximation algorithms in an attempt to optimize the trade-off between the two effects. We show that our algorithm achieves the best possible approximation ratio.

*Department of Computer Science, Stanford University, Stanford, CA 94305. Supported by a Stanford Graduate Fellowship. E-mail: gagan@cs.stanford.edu.

†E-mail: tomas@theory.stanford.edu.

‡Department of Computer Science, Stanford University, Stanford, CA 94305. Supported in part by NSF Grants IIS-0118173, EIA-0137761, ITR-0331640, an SNRC grant, and grants from Microsoft and Veritas. E-mail: rajeev@cs.stanford.edu.

§E-mail: rinap@cisco.com.

¶Department of Computer Science, Stanford University, Stanford, CA 94305. Supported by a GRPW fellowship from Bell Labs, Lucent Technologies. E-mail : anzhu@cs.stanford.edu.

## 1 Introduction

As relational databases keep growing in size, good overall performance for queries and updates necessitates the optimization of I/O performance on secondary storage. A significant aspect of I/O performance is *database layout*, i.e., how database objects such as tables, indexes, materialized views, etc, are assigned to the available disk drives.

The traditional approach has been to spread out each database object uniformly over all available disk drives, called *full striping*, in order to obtain I/O parallelism. As long as only one object is accessed at a time, this solution can be shown to be optimal with respect to I/O performance. However, when dealing with a query workload which involves co-access of of two or more database objects, e.g., a merge join of two tables, there is a distinct possibility that uniform striping could lead to substantially suboptimal performance. The main reason for this is that if the concurrently-accessed (co-accessed) objects are co-located on a disk drive, then instead of the transfer time to main memory, the disk seek time needed for the random or interleaved access to the disk blocks containing the two objects begins to dominate the I/O cost. As a result, there is a trade-off between the benefit due to I/O parallelism and the overhead due to random I/O accesses. For queries co-accessing multiple objects, I/O performance might be improved by choosing a database layout which differs from full striping.

In our study of the database layout problem, we adopt the framework and cost model proposed by Agrawal, Chaudhuri, Das, and Narasayya [2], which combines the effects of both I/O parallelism and random disk access. Using their framework, we model the problem as follows. All database objects are referred to as *files*, each with a specified size $r_i$ in terms of number of disk blocks. In addition, we are given a set of disk drives, each parametrized by its *transfer rate* $1/\alpha_j$, its average *seek time* $\beta_j/2$, and capacity $L_j$. The transfer rate $1/\alpha_j$ is the rate at which disk $j$ can read data once the read head is positioned correctly. It captures the cost of reading data from the disk without accounting for the time needed to adjust the head for random I/O accesses. The average seek time $\beta_j/2$ captures the average time taken to position the read head for random accesses.

The objective in the layout problem is to give a (possibly fractional) assignment of files to disks (without replication), respecting the disk capacity constraints. Let $r_{ij}$ be the fraction of file $i$ assigned to disk $j$. Then, for a query co-accessing two files $i$ and $i'$, the *transfer time* from disk $j$ is given by $\alpha_j(r_{ij} + r_{i'j})$. The *seek time* for disk $j$ is given by $\beta_j \cdot \min\{r_{ij}, r_{i'j}\}$. The reasoning behind this cost model is as follows: without loss of generality, let $r_{ij} \leq r_{i'j}$; then, the number of times the disk head switches to file $i$ from file $i'$ on disk $j$ is bounded by the number of blocks file $i$ has on disk $j$, and the number of times it switches to file $i'$ from file $i$ on disk $j$ is also bounded by the same number.

For the sake of brevity, we focus our exposition on the case of two-object queries, i.e., queries co-accessing exactly two files. Note that single-object queries are trivially handled by viewing queries accessing a single file $i$ as co-accessing files $i$ and $x$, where $x$ is an imaginary file of size zero. In general, if a query co-accesses more than two files, we can replace the query with a set of two-object queries [2]; the details are omitted for the purpose of this extended abstract. Following [2], we assume that a query workload is specified and that the objective is to minimize the overall access time of the workload, where the access time for an individual query is defined to be the maximum over all disks of the sum of the transfer and seek times. Formally, the problem is defined as follows.

Definition 1.1. **[Database Layout Problem]**

Input: *Files $F = \{1, 2, \ldots, n\}$, with file $i$ having size $r_i$. A set of disks $D = \{1, 2, \ldots, m\}$, where each disk $j$ has transfer rate $1/\alpha_i$, average seek time $\beta_j/2$, and capacity $L_j$. A query workload consisting of a set of queries $Q = \{q = (i, i') : i, i' \in F\}$, with each query $q = (i, i')$ having frequency weight $\phi(i, i')$.*

OBJECTIVE: *Determine a fractional assignment* $r_ij$ *of file* $i$ *to disks* $j$, *such that* $\sum_{j\in D} r_{ij} = r_i$ *for all files* $i$ *and* $\sum_{i\in F} r_{ij} \leq L_j$ *for all disks* $j$, *so as to minimize the overall query access time* $\sum_{q\in Q} P(q)$, *where the access time for query* $q = (i, i')$ *is* $P(q) = \phi(i, i') \cdot \max_{j\in D}(r_{ij}\alpha_j + r_{i'j}\alpha_j + \min\{r_{ij}\beta_j, r_{i'j}\beta_j\})$.

We present a formal analysis of this optimization problem. First, we consider single-object queries, i.e., queries which do not involve co-access of files. For a workload consisting of only such queries, we give a simple and fast greedy algorithm, showing that it always finds the optimal assignment of files to disks. This algorithm is much faster compared to the straight forward linear programming method. Next, we consider workloads consisting of queries that co-access two files simultaneously. Considering only the transfer time, while ignoring seek time, we show that a natural weighted variant of *full striping* gives the optimal solution, provided that disk capacity is not a constraint; taking into account disk capacities, we can find the optimal solution in polynomial-time based on a linear programming formulation. Conversely, considering only the seek time, while ignoring the transfer time, we show that there exists an optimal solution that assigns each file to a single disk, i.e., no files are split across disks. This observation helps us relate the problem to a well-studied NP-complete problem, allowing us to infer its intractability. Finally, we consider the general objective of minimizing the combination of transfer and seek times. Define $\gamma = \max_{j\in D} \frac{\beta_j}{\alpha_j}$. We show that the general problem is NP-hard to *approximate* within a factor of $\rho\gamma$, for some constant $0 < \rho < 1$. On a positive note, we present an $(1+\gamma/2)$-approximation polynomial-time algorithm for the general objective function. By the preceding negative result, our approximation algorithm is optimal up to small constant factors.

The rest of this paper is organized as follows. Section 2 presents algorithms for single-object queries. Section 3 investigates the structure of database layout while minimizing only the transfer time, providing an optimal algorithm for this objective function. Section 4 investigates the structure of database layout while minimizing only the seek time. Section 5 deals with the problem of minimizing the combination of transfer and seek times. Finally, we conclude in Section 6 with an overview of some related work in this area.

## 2 Algorithm for Single Object Queries

In this section, we present an efficient algorithm to find the optimal assignment for single-object (file) queries. Since each query consists of only one file, there is no random access cost, i.e., we have to consider only the transfer time for each file, and $P(q)$ is redefined as $\max_{j\in D}\{r_{ij}\alpha_j\}$ for a query $q$ accessing file $i$. If every file has the same size, and the disks are homogeneous, then it is clear that striping each file uniformly across all disks is an optimal solution. Here, we analyze the general case where the files are of different sizes, and the disks are heterogeneous. For this section only, we assume each file $i$ has access frequency $\phi_i$. We could easily write a linear program to solve the instance, but we present an optimal greedy algorithm, which is simpler and faster.

The greedy algorithm is as follows. Consider the files one by one, in decreasing order of $\phi_i$. When file $i$ is considered, we first attempt to split file $i$ across all $m$ disks in a way that $r_{ij}\alpha_j$ is a constant, i.e., the transfer time of file $i$ is the same across all disks, and $\sum_j r_{ij} = r_i$. If for all disks $j$, adding $r_{ij}$ to the existing assignment to disk $j$ does not exceed $L_j$, then $r_{ij}$ blocks of file $i$ are assigned to disk $j$ and file $i$ is removed from consideration. Otherwise, let $\kappa$ be the largest constant such that, for all disks $j$, assigning $\frac{\kappa}{\alpha_j}$ to the existing assignment on disk $j$ does not exceed the disk capacity $L_j$. Then we partially assign file $i$ by assigning $r_{ij} = \frac{\kappa}{\alpha_j}$ to disk $j$. Clearly, $r_{ij}\alpha_j$ is still a constant, and no disk capacity is violated. In this case $\sum_j r_{ij} < r_i$, while the capacity of one of the disks is saturated. The saturated disk is removed from further consideration while the remainder of file $i$ is considered at the next step. Notice that in each step, we either assign a file completely to disks, or saturate the capacity of one of the disks. This implies that the algorithm terminates after at most $n + m$ such operations. The pseudo-code of the greedy algorithm is presented in the Appendix.

We first prove the following lemma about the greedy algorithm.

LEMMA 2.1. *At any point of execution of the greedy algorithm, let $U$ be the set of unsaturated disks. Then for each file $i$, $r_{ij}\alpha_j$ is a constant over all unsaturated disks $j \in U$. Further, for each file $i$, $r_{ij}\alpha_j \geq r_{ij'}\alpha_{j'}$ for any disk $j \in U$ and $j' \notin U$.*

*Proof.* In the greedy algorithm, we always (partially) assign a file $i$ to disks so that $r_{ij}\alpha_j$ is the same for all unsaturated disks. Moreover, once a disk gets saturated, no more files are assigned to it. This implies that for a file $i$, $r_{ij}\alpha_j$ for saturated disks is no more than that for unsaturated disks. ■

Let $C_j$ denote the current load on disk $j$, i.e., the sum of all file sizes assigned to disk $j$. The above lemma gives us the following corollary.

COROLLARY 2.1. *At any point of execution of the greedy algorithm, let $U$ be the set of unsaturated disks. $C_j\alpha_j$ is a constant, for each $j \in U$. Further, $C_j\alpha_j \geq C_{j'}\alpha_{j'}$ for any disk $j \in U$ and $j' \notin U$.*

Next, we prove the optimality of the greedy solution.

THEOREM 2.1. *Let $\mathcal{S}$ be the assignment defined by the $r_{ij}$'s produced by the greedy algorithm. Let $\mathcal{S}'$ be any other assignment. Let $Z$ be the total transfer time incurred by $\mathcal{S}$ and let $Z'$ be the total transfer time incurred by $\mathcal{S}'$. Then $Z \leq Z'$.*

*Proof.* Given an instance $\mathcal{I}$, let $\phi$ denote the least frequency among the $n$ files in the instance. First consider the solution produced by the greedy algorithm. For each file $i$, let $z_i$ denote its transfer time weighted by its frequency, i.e., $z_i = \phi_i \cdot \max_j(\alpha_j r_{ij})$. Let $z_i = z_i^A + z_i^B$, where $z_i^A = (\phi_i - \phi) \cdot \max_j(\alpha_j r_{ij})$ and $z_i^B = \phi \cdot \max_j(\alpha_j r_{ij})$. Further, let the total transfer time over all files be $Z = Z^A + Z^B$, where $Z^A = \sum_i z_i^A$ and $Z^B = \sum_i z_i^B$. Now consider any other assignment $\mathcal{S}'$. Let $r'_{ij}$ be the file assignments corresponding to $\mathcal{S}'$. Define $z'_i = {z'}_i^A + {z'}_i^B$ in terms of the $r'_{ij}$, and $Z' = {Z'}^A + {Z'}^B$ as above.

We will show that $Z \leq Z'$ for all instances by induction on the number of files $n$. The hypothesis states that $Z \leq Z'$ for all instances with $k$ files. The base case with $k = 0$ is clearly true. For the induction hypothesis, we assume that $Z \leq Z'$ for $k$. Consider an instance $\mathcal{I}$ with $k+1$ files. Notice that $Z^A$ for instance $\mathcal{I}$ represents the total transfer time incurred by the greedy algorithm on an instance $\mathcal{I}'$ consisting of files 1 through $k$ in the original instance $\mathcal{I}$, with file $i$ having frequency $\phi_i - \phi$. This is because each file's frequency is reduced uniformly by $\phi$, so the order of the files remains unchanged. The greedy algorithm would consider these $k$ files in the same order as in instance $\mathcal{I}$, and give exactly the same assignment of these $k$ files to disks. Similarly, ${Z'}^A$ represents the total transfer time on instance $\mathcal{I}'$ under assignment $\mathcal{S}'$. By the inductive hypothesis, we know that $Z^A \leq {Z'}^A$. Now we concentrate on $Z^B$. This corresponds to the total transfer time incurred by the greedy algorithm on another instance $\mathcal{I}''$, consisting of all $k+1$ files in $\mathcal{I}$, each with frequency $\phi$. Similarly, ${Z'}^B$ corresponds to the total transfer time on the instance $\mathcal{I}''$ under assignment $\mathcal{S}'$.

Let $l$ be the last disk that received some file assignment according to the greedy algorithm running on instance $\mathcal{I}''$. By Lemma 2.1, we have that for any file $i$ and disk $j$, $r_{ij}\alpha_j \leq r_{il}\alpha_l$, implying $\max_j(r_{ij}\alpha_j) = r_{il}\alpha_l$. Since $C_l = \sum_i r_{il}$, we have that the overall transfer time $Z^B = \phi\alpha_l C_l$. By Corollary 2.1, in solution $\mathcal{S}$, every disk $j$ is either saturated, or has a current load $C_j$, such that $C_j\alpha_j = C_l\alpha_l$. Now consider the assignment $\mathcal{S}'$ that incurs ${Z'}^B$ on instance $\mathcal{I}''$. Let $C'_j$ denote the final load on disk $j$ in $\mathcal{S}'$. Then there exists a disk $d$, such that $C'_d \geq C_d$ for some disk $d$ which is unsaturated in $\mathcal{S}$. This is because $C'_d < C_d$ for all unsaturated disks would imply that the total amount assigned by $\mathcal{S}'$ is less than the total amount assigned by $\mathcal{S}$. Thus, $Z'^B \geq \phi\alpha_d C'_d \geq \phi\alpha_d C_d = \phi\alpha_l C_l = Z^B$. Therefore $Z' = Z'^A + Z'^B \geq Z^A + Z^B = Z$. This completes the induction. ■

This completes our discussion of single-object queries. In the rest of the paper, we will concentrate on the general model in which each query co-accesses two objects.

## 3 Algorithms for Transfer Time

In this section, we developing algorithms that minimize the total transfer time only while ignoring the seek time, for queries co-accessing two files. We will show the optimal assignment will spread files uniformly across all disks, in the case of disks have large enough storage capacity. We will also present algorithms to solve the more general case where disks have limited capacities. The following definitions will be used throughout the rest of the paper.

DEFINITION 3.1. *Let $f_{ij}$ denote the fraction of file $i$ assigned to disk $j$, i.e, $f_{ij} = r_{ij}/r_i$. Let $T(q)$ and $S(q)$ denote the transfer and seek time respectively for query $q = (i, i')$, i.e., $T(q) = \max_{j\in D}(r_{ij}\alpha_j + r_{i'j}\alpha_j)$ and $S(q) = \max_{j\in D}(\min\{r_{ij}, r_{i'j}\} \cdot \beta_j)$.*

Clearly, $\phi(i, i') \cdot \max\{T(q), S(q)\} \leq P(q) \leq \phi(i, i')(T(q) + S(q))$. We use superscripts to distinguish between different assignments, for instance, $S^{\mathcal{A}}(q)$ denotes the seek time of query $q$ in assignment $\mathcal{A}$. When the context is clear, the superscripts are omitted.

THEOREM 3.1. *If disks have unlimited storage capacity, then each file should be so split across all $m$ disks such that for each file $i$ $r_{ij}\alpha_j$ is a constant over all disks $j$.*

*Proof.* For each file $i$, let $\kappa_i$ denote a constant such that setting $r_{ij} = \frac{\kappa_i}{\alpha_j}$ for all $j$ gives $\sum_j r_{ij} = r_i$. We first show that for a query $q = (i, i')$, $T(q) \geq \kappa_i + \kappa_{i'}$ in any assignment. The two files have a combined size of $r_i + r_{i'}$. Since we are only concerned with transfer time, it is equivalent to spreading a single file $f$ of size $r_f = r_i + r_{i'}$ across disks. The best way to spread the total mass $r_i + r_{i'}$ across disks is to choose $r_{fj}$ such that $r_{fj}\alpha_j$ is constant, say $\kappa_f$, over all disks $j$, and $\sum_j r_{fj} = r_f$. This implies that $\kappa_f = \kappa_i + \kappa_{i'}$, hence $T(q) \geq \kappa_i + \kappa_{i'}$. The uniform assignment achieves $T(q) = \kappa_i + \kappa_{i'}$ for each query $q = (i, i')$, which implies its optimality. ■

The above theorem applies as along as the disk capacity $L_j \geq \sum_i r_{ij}$. In case some disks have smaller capacities, we can use linear programming to solve the instance optimally. Besides the variables $r_{ij}$, we introduce a new variable $x_{ii'}$ for each query $q = (i, i')$. Intuitively, $x_{ii'}$ represents $T(q)$ expressed with a set of linear constraints.

$$
\begin{aligned}
\min \quad &: \quad \sum_{q=(i,i')\in Q} \phi(i,i') x_{ii'} \\
\text{subject to} \quad &: \quad x_{ii'} \geq (r_{ij} + r_{i'j})\alpha_j, \ \forall\ q = (i,i') \in Q,\ j \in D \\
& \quad\ \ L_j \geq \sum_{i\in F} r_{ij}, \ \forall\ j \in D \\
& \quad\ \ r_i = \sum_{j\in D} r_{ij}, \ \forall\ i \in F \\
& \quad\ \ r_{ij} \geq 0, \ \forall\ i \in F,\ j \in D
\end{aligned}
$$

Clearly, the values $r_{ij}$ obtained after solving the LP will be an optimal assignment of files to disks.

## 4 Algorithms for Seek Time

Next we consider the problem of minimizing only the seek time, while ignoring the transfer time. The phrase "the DATABASE LAYOUT PROBLEM" (or "the DL problem") in this section refers exclusively to the problem of minimizing the total seek time. Our goal is to establish an equivalence relation between the problem of minimizing the total seek time and MINIMUM EDGE DELETION $k$-PARTITION problem, a well-studied NP-complete problem. As shown at the end of this section, minimizing the total seek time with unlimited disk capacities is very hard to approximate, which implies that the more general case of limited capacity disks is even harder to approximate. The hardness result from this section will aid us in deriving a hardness result for the case of minimizing the combined access time.

We first investigate the combinatorial structure of the optimal solution for minimizing the total seek time for disks with unlimited capacities. We distinguish between two types of assignment: *integral* and *fractional.*

DEFINITION 4.1. *An* integral *assignment of files to disks is the one that assigns each file completely to one of the disks. A* fractional *assignment allows a file to be split across multiple disks.*

Obviously, we are interested in the optimal *fractional* assignment, as it allows more flexibility and possibly solutions with lower total transfer time. However, we first show that the gap between the optimal *fractional* assignment and the optimal *integral* assignment is bounded by a constant.

LEMMA 4.1. *For any assignment $\mathcal{A}$ with total seek time $C_A$, there exists an* integral *assignment $\mathcal{A}'$ with total seek time $C_{A'}$, where $C_{A'} \leq 2C_A$.*

*Proof.* As defined earlier, let $f_{ij}$ denote the fraction of file $i$ assigned to disk $j$ in $\mathcal{A}$. Note that $\sum_j f_{ij} = 1$ for all $i$. We create the *integral* assignment $\mathcal{A}'$ via a randomized algorithm. In $\mathcal{A}'$, we assign file $i$ to disk $j$ with probability $f_{ij}$. Next, we bound the total seek time incurred in $\mathcal{A}'$. Consider any query $q = (i, i')$. In $\mathcal{A}$, the seek time incurred in co-accessing files $i$ and $i'$ is $S^{\mathcal{A}}(q) = \max_j(\min\{r_i f_{ij}\beta_j, r_{i'} f_{i'j}\beta_j\})$. Let $g = \max_j(\min\{f_{ij}\beta_j, f_{i'j}\beta_j\})$, then $S^{\mathcal{A}}(q) \geq g \cdot \min\{r_i,\ r_{i'}\}$. In the assignment $\mathcal{A}'$, if files $i$ and $i'$ get assigned to different disks, then there is no seek time incurred in co-accessing files $i$ and $i'$. If files $i$ and $i'$ get assigned to the same disk $j$, then the seek time becomes $\beta_j \cdot \min\{r_i, r_{i'}\}$. The probability of both the files $i$ and $i'$ get assigned to disks $j$ is $f_{ij} \cdot f_{i'j}$. Observe that $f_{ij} \cdot f_{i'j} \cdot \beta_j \leq g \cdot (f_{ij} + f_{i'j})$. This implies the expected total seek time for $\mathcal{A}'$ is given by

$$\begin{aligned} S^{\mathcal{A}'}(q) &= \min\{r_i,\ r_{i'}\} \cdot \sum_j f_{ij} \cdot f_{i'j} \cdot \beta_j \\ &\leq \min\{r_i,\ r_{i'}\} \cdot \sum_j g \cdot (f_{ij} + f_{i'j}) \\ &= \min\{r_i,\ r_{i'}\} \cdot g \cdot (\sum_j f_{ij} + \sum_j f_{i'j}) \\ &= 2g \cdot \min\{r_i,\ r_{i'}\} \\ &\leq 2S^{\mathcal{A}}(q) \end{aligned}$$

Since $C_A = \sum_{q=(i,i')\in Q} \phi(i,i') S^{\mathcal{A}}(q)$ and $C_{A'} = \sum_{q=(i,i')\in Q} \phi(i,i') S^{\mathcal{A}'}(q)$, we conclude that $C_{A'} \leq 2C_A$. We just showed that the expected seek time for the integral solutions returned by the randomized algorithm is no more than $2C_A$. This implies the existence of an integral assignment with seek time no more than $2C_A$. ■

We note that the factor 2 is tight for this randomized rounding procedure. Consider the following fractional assignment for two unit size files and two identical disks with seek rate 1: $r_{11} = r_{22} = \epsilon$ and

$r_{12} = r_{21} = 1 - \epsilon$. Thus the seek time $S^{\mathcal{A}}$ for the query $q = (1, 2)$ is $\epsilon$. After randomized rounding of this fractional assignment, the two files collide with probability $2\epsilon(1-\epsilon)$, for an expected seek time $S^{\mathcal{A}'}$ of $2\epsilon(1-\epsilon)$. The ratio of $S^{\mathcal{A}}$ to $S^{\mathcal{A}'}$ is $2(1-\epsilon)$ which approaches 2 as $\epsilon \to 0$.

Further, we will show that there exists an integral assignment which is as good as the optimal *fractional* assignment. We first prove a weaker property for which we need to introduce the following notation.

DEFINITION 4.2. *Define $c_{ij} = r_{ij}\beta_j$, the potential seek time of file $i$ on disk $j$. For each file $i$, define $c_i^{\min} = \min_j\{c_{ij} : c_{ij} \neq 0\}$ and $c_i^{\max} = \max_j\{c_{ij}\}$. Let $U$ denote the set of files $i$ such that $c_i^{\max} \neq c_i^{\min}$. Define $\Delta = \min_{i \in U} c_i^{\min}$. Let $V = \{z_1, z_2, \ldots, z_k\} \subseteq U$ denote the set of files such that $c_i^{\min} = \Delta$, for $i \in V$. For an assignment $\mathcal{A}$, define the* stretch *of $\mathcal{A}$ as $R(A) = \max_{z_i \in V}(c_{z_i}^{\max} - c_{z_i}^{\min})$.*

LEMMA 4.2. *There exists an optimal* fractional *solution, such that each file is split uniformly in terms of potential seek time across a subset of disks, i.e., for each file $i$ there exists a constant $c_i$, such that $c_{ij} \in \{0, c_i\}$ for all disks $j$.*

*Proof.* We prove this by contradiction. Suppose every optimal *fractional* assignment violates this property. Let $\mathcal{A}$ be the optimal fractional assignment with the minimum stretch $R(A)$. In $\mathcal{A}$, we count the total frequency $\Phi$ of queries of the form $q = (z_i, \star)$ with $z_i \in V, \star \in U$, and $S^{\mathcal{A}}(q) = c_{z_i}^{\min} = \Delta$, or of the form $q = (z_i, \star)$ with $z_i \in V, \star \notin U$, $S^{\mathcal{A}}(q) = \Delta$, and $c_\star^{\min} = c_\star^{\max} > \Delta$. Since queries are unordered pairs of files, in the case of a query of the form $q = (z_i, z_{i'})$ with $z_i, z_{i'} \in V$, we only increment $\Phi$ once if $S^{\mathcal{A}}(q) = \Delta$. We also count the total frequencies $\Psi_i$ of queries of the form $q = (z_i, \star)$ with $\star \notin V$ and $S^{\mathcal{A}}(q) = c_{z_i}^{\max}$, or of the form $q = (z_i, \star)$ with $\star \in V$, $S^{\mathcal{A}}(q) = c_{z_i}^{\max}$, and $c_\star^{\max} > c_{z_i}^{\max}$, for each file $z_i \in V$. In addition, for each query of the form $q = (z_i, z_{i'})$, we keep a counter $\Psi_{ii'}$, which is set to $\Psi_{ii'} = \phi(i, i')$ if $S^{\mathcal{A}}(q) = c_{z_i}^{\max} = c_{z_i'}^{\max}$, and is set to 0 otherwise.

Notice that for any query $q = (i, i')$, we have $S^{\mathcal{A}}(q) \leq c_i^{\max}$ and $S^{\mathcal{A}}(q) \leq c_{i'}^{\max}$. Another useful property is that for any query $q = (i, i')$, if we increase $r_{ij}$ by $\epsilon/\beta_j$ for all $j$ with $c_{ij} = S^{\mathcal{A}}(q)$, then the new seek time for $q$ is no more than $S^{\mathcal{A}}(q) + \epsilon$. Consider the following two possible modifications to assignment $\mathcal{A}$:

1. Increase the value $\Delta = c_{z_i}^{\min}$ by a small amount $\epsilon > 0$. We achieve this by increasing the value $r_{z_i j}$ by $\epsilon/\beta_j$ on disks $j$ with $c_{z_i j} = c_{z_i}^{\min}$, for each $z_i \in V$. To balance out the assignment for files in $V$, we decrease $c_{z_i j}$ by an appropriate amount $\epsilon_i > 0$, on disks $j$ with $c_{z_i j} = c_{z_i}^{\max}$, for each $z_i \in V$. The choice of $\epsilon$ determines each value of $\epsilon_i$ and vice versa. Let the assignment obtained after this modification be $\mathcal{A}'$. If a query $q$ was contributed to $\Phi$, i.e., $q = (z_i, \star)$ with $S^{\mathcal{A}}(q) = \Delta$, then $S^{\mathcal{A}'}(q) \leq \Delta + \epsilon$.
   *Claim:* There exists an $\epsilon$ small enough such that if the query $q = (z_i, \star)$ contributed to $\Psi_i$ in assignment $\mathcal{A}$, then $S^{\mathcal{A}'}(p) \leq c_{z_i}^{\max} - \epsilon_i$.
   We argue this as follows. In order to ensure that $c_{z_i}^{\max} - \epsilon_i$ is still the maximum potential seek time for file $z_i$ in $\mathcal{A}'$, an $\epsilon_i$ equaling the difference between the maximum and second maximum potential seek time of file $z_i$ in $\mathcal{A}$ suffices. This choice of $\epsilon_i$ determines a value for $\epsilon$. We determine the value of *epsilon* through all files $z_i \in V$ and use the smallest value of $\epsilon$ found. With this choice of $\epsilon$, we also ensure that for the query $q = (z_i, z_{i'})$ with $S^{\mathcal{A}}(q) = c_{z_i}^{\max} = c_{z_{i'}}^{\max}$, $S^{\mathcal{A}'}(q) \leq c_{z_i}^{\max} - \max\{\epsilon_i, \epsilon_{i'}\}$. There exists a small enough $\epsilon$ so that besides the conditions above, the seek time of all other queries[1] does not increase. For a query $q = (z_i, \star)$ with $S^{\mathcal{A}}(q) < \Delta$, since we didn't change the assignment of any file with potential seek time less than $\Delta$ on any

[1] Queries that are not counted towards $\Phi$, $\Psi_i$, or $\Psi_{i,i'}$. A query $q = (z_i, z_i')$ is counted towards $\Psi_{i,i'}$ only if $\Psi_{i,i'} = \phi(i, i')$.

disk, $S^{\mathcal{A}'}(q) \le S^{\mathcal{A}}(q)$. For a query $q = (z_i, \star)$ with $S^{\mathcal{A}}(q) = \Delta$ and not counted towards $\Phi$, we know that $\star \notin U$ and $c_\star^{\min} = c_\star^{\max} = \Delta$, implying that $S^{\mathcal{A}'}(q) \le \Delta$. For a query $q = (z_i, \star)$ with $S^{\mathcal{A}}(q) > \Delta$, we just need to make sure that $\epsilon \le S^{\mathcal{A}}(q) - \Delta$. By the choice of such an $\epsilon$, the total seek time for $\mathcal{A}'$ compared to that of $\mathcal{A}$ is increased by at most $\epsilon\Phi$, and decreased by at least $\sum_{1 \le i \le k} \epsilon_i \Psi_i + \sum_{q=(i,i') \in Q} \max\{\epsilon_i, \epsilon_{i'}\} \Psi_{ii'}$.

2. Decrease the value $\Delta = c_{z_i}^{\min}$ by the same amount $\epsilon$ determined in the first modification. We do this by decreasing the value $r_{z_i j}$ by $\epsilon/\beta_j$ on disks $j$ with $c_{z_i j} = c_{z_i}^{\min}$, for each $z_i \in V$. Correspondingly, we increase $c_{z_i j}$ by the appropriate amount $\epsilon_i$, on those disks $j$ with $c_{\{z_i\}j} = c_{z_i}^{\max}$, for each $z_i \in V$. Let $\mathcal{A}''$ be the assignment obtained after this modification.
*Claim:* If a query $q$ contributed towards $\Phi$, i.e., $q = (z_i, \star)$ with $S^{\mathcal{A}}(q) = \Delta$, then $S^{\mathcal{A}''}(q) = \Delta - \epsilon$. We prove this claim by considering three situations. If $\star \notin U$, then by definition, $c_\star^{\max} = c_\star^{\min} > \Delta$. The fact that $S^{\mathcal{A}}(q) = \Delta$ implies that $\star$ resides only on those disks $j$ where $c_{z_i j}^{\mathcal{A}} \in \{0, \Delta\}$. We know that for these disks $j$, $c_{z_i j}^{\mathcal{A}'} \in \{0, \Delta - \epsilon\}$, and hence the claim holds. Else if $\star \in U$ but $\star \notin V$, then $c_\star^{min} > \Delta$. In this case, the claim holds by a reasoning similar to the one above. Otherwise, $\star = z_{i'} \in V$. In this case, $S^{\mathcal{A}}(q) = \Delta$ implies that in $\mathcal{A}$, there is no disk $j$ for which both $c_{z_i j}$ and $c_{z_{i'} j}$ exceed $\Delta$, and again the claim holds with a similar reasoning. Next, notice that if $q = (z_i, \star)$ was counted towards $\Psi_i$ in $\mathcal{A}$, then $S^{\mathcal{A}''}(q) \le c_{z_i}^{\max} + \epsilon_i$. And if $\Psi_{ii'} = \phi(i, i')$, then $S^{\mathcal{A}''}(q) = c_{z_i}^{\max} + \min\{\epsilon_i, \epsilon_{i'}\}$. Regardless of the above choice of $\epsilon$, this modification does not increase the seek time of any of the queries which does not contribute towards $\Phi$, $\Psi_i$, or $\Psi_{i,i'}$, since only the $c_{z_i}^{\max}$'s are increased. Thus the total seek time is decreased by at least $\epsilon\Phi$, and increased by at most $\sum_{1 \le i \le k} \epsilon_i \Psi_i + \sum_{i \ne i'} \min\{\epsilon_i, \epsilon_{i'}\} \Psi_{ii'}$.

If the first modification does not increase the total seek time, then we have found an optimal assignment $\mathcal{A}'$ with a smaller stretch value than $\mathcal{A}$, contradicting the choice of $\mathcal{A}$. Otherwise, the second assignment $\mathcal{A}''$ must have smaller total seek time compared to $\mathcal{A}$, since $\min\{\epsilon_i, \epsilon_{i'}\} \le \max\{\epsilon_i, \epsilon_{i'}\}$. In this case, we have found an assignment with a smaller total seek time than $\mathcal{A}$, contradicting the optimality of $\mathcal{A}$. ■

With the previous lemma, we now show the following.

THEOREM 4.1. *There exists an optimal* fractional *assignment that is also* integral.

*Proof.* We prove this by contradiction. Suppose every optimal *fractional* assignment splits at least one file. Among all optimal *fractional* assignments that spread the files uniformly in terms of potential seek time (by the previous lemma, there exists at least one such assignment), let $\mathcal{A}$ be the assignment which minimizes the number of split files. Let $c_i$ be the uniform potential seek time for file $i$. Consider the file $z$ with the largest $c_i$ among all the split files $i$. For convenience, reorder the disks so that $c_{zj} = c_z$ for $1 \le j \le k$, where $k \in \mathbb{N}$. Let $\Phi_j$ denote the total frequency of queries of the form $q = (z, \star)$ with $c_{\star j} > c_z$, for $1 \le j \le k$. Note that $S^{\mathcal{A}}(q) = c_z$ for these queries. Since $z$ is the file with the largest $c_z$ among all split files, $c_{\star j} > c_z$ implies that $\star$ must be completely assigned to disk $j$. Thus, queries which contribute to different $\Phi_j$'s are disjoint, implying that the total seek time for queries which contribute towards the $\Phi_j$'s is $\sum_{1 \le j \le k} \Phi_j c_z = \sum_{1 \le j \le k} \Phi_j \beta_j r_{zj}$. We modify $\mathcal{A}$ by completely assigning $z$ to disk $y$ with the smallest $\Phi_j \beta_j$, for $1 \le j \le k$. Let $\mathcal{A}'$ be the assignment obtained after the modification. Under this assignment, the total seek time for the queries which contributed towards $\Phi_i$'s is $\Phi_y \beta_y r_z = \Phi_y \beta_y \sum_{1 \le j \le k} r_{zj} \le \sum_{1 \le j \le k} \Phi_j \beta_j r_{zj}$. For any other query $q = (z, \star)$ with $S^{\mathcal{A}}(q) \le c_z$, we stipulate that $S^{\mathcal{A}}(q) = c_\star$ whenever $S^{\mathcal{A}}(q) \ne 0$. These queries' seek costs do not increase in $\mathcal{A}'$, since $c_\star$ remains unchanged. Thus, overall $\mathcal{A}'$ has no more total seek time than the optimal assignment $\mathcal{A}$.

Thus, we have found another optimal assignment $\mathcal{A}'$ with one less split file than $\mathcal{A}$, contradicting the choice of $\mathcal{A}$. ■

The preceding theorem proves the existence of an *integral* optimal assignment. Our next theorem gives a polynomial time algorithm converting any *fractional* assignment to an *integral* one with equal or less total seek time, improving Lemma 4.1.

THEOREM 4.2. *There exists a polynomial time algorithm, which for any given* fractional *assignment $\mathcal{A}$, finds an* integral *assignment $\mathcal{A}'$ with equal or less total seek time.*

*Proof.* We utilize the proof of Lemma 4.2. At each step, we either perform Modification 1 or Modification 2, whichever doesn't increase the total seek time. The first modification requires us to pick an appropriate $\epsilon$. For this, the two sufficient conditions are:

1. We need to ensure that for each file $z_i$, $c_{z_i}^{\max} - \epsilon_i$ is still the maximum potential seek time for file $z_i$ in $\mathcal{A}'$. For this, each $\epsilon_i$ should be no more than the difference between the maximum and second maximum potential seek time of file $z_i$ in $\mathcal{A}$. Enumerating through all such $z_i$'s, we can pick the smallest $\epsilon$ corresponding to the $\epsilon_i$'s determined by the above condition. Note that this choice of $\epsilon$ decreases at each one file $z_i$'s maximum potential seek time to equal to its second maximum potential seek time in $\mathcal{A}$, thus reducing the total number of different potential seek costs for all files by at least 1.

2. We need to make sure that $\Delta + \epsilon$ does not exceed $S^{\mathcal{A}}(q)$, for every query $q$ with $S^{\mathcal{A}}(q) > \Delta$. For this, we can pick $\epsilon = \min_{q:S^{\mathcal{A}}(q)>\Delta} S^{\mathcal{A}}(q) - \Delta$. Note that this choice of $\epsilon$ also decreases the number of different potential seek costs by 1.

We pick the smaller of the two $\epsilon$'s found by the conditions above, and each execution of the first modification reduces the number of different potential seek costs by one.

If the second modification is chosen instead, then reducing $\Delta$ by $\epsilon$ does not increase the total seek time. We notice the following. By an argument similar to the one in Lemma 4.2, reducing $c_{z_i}^{\min}$ from $\Delta$ all the way down to 0 instead[2] would not increase the total seek time. This variant of the second modification also reduces the total different potential seek costs by one.

We have a total of $n$ files, each file could have at most $m$ different potential seek costs in $\mathcal{A}$. This implies at most $nm$ modifications are needed to produce a solution with uniform potential seek costs for all files.

Next we utilize Theorem 4.1 to reassign each split file to a unique disk. We need at most $n$ such reassignment, since each reassignment decreases the number of split files by one. Overall, the total number of operations is polynomial. ■

The above theorem indicates that the problem of finding an assignment that minimizes the total seek time is as hard as the problem of finding an *integral* assignment. Next we show that minimizing the total seek time for homogeneous disks, i.e., disks with uniform seek rate, is equivalent to the MINIMUM EDGE DELETION $k$-PARTITION problem (MEDP-$k$).

DEFINITION 4.3. [MINIMUM EDGE DELETION $k$-PARTITION PROBLEM]

INPUT: *A graph $G = (V, E)$, with weighted edges.*

[2] We set $c_{z_i j}^{\mathcal{A}'}$ to 0 for all $z_i$'s and $j$ such that $c_{z_i j}^{\mathcal{A}} = c_{z_i}^{\min} = \Delta$.

OBJECTIVE: *Find a coloring of vertices with $k$ colors $\mathcal{C} : V \to \{k\}$ that minimizes the total weight of monochromatic edges (an edge is monochromatic iff its end points have the same color in $\mathcal{C}$).*

THEOREM 4.3. *The* DATABASE LAYOUT *problem with $m$ homogeneous disks is equivalent to the* MINIMUM EDGE DELETION $m$-PARTITION *problem.*

*Proof.* We first reduce the DL problem with homogenous disks to the MEDP-$m$ problem. Given files $F$, disks $D$ and a query workload $Q$ with query frequency $\phi(i, i')$'s, we create a graph $G = (V, E)$, where $V = F$ and $E = Q$. And the weight of an edge $e = (i, i')$, $w(e)$, is set to $\phi(i, i') \min\{r_i, r_{i'}\}$. Given a solution $\mathcal{S}$ to this instance of the MEDP-$m$ problem, we produce an *integral* assignment $\mathcal{A}$ for the DL problem as follows: a file $i$ is assigned to disk $j$ if and only if it was assigned color $j$ in the MEDP-$m$ solution. It is clear that the total weight of monochromatic edges in $\mathcal{S}$ is exactly the total seek time incurred by the *integral* assignment $\mathcal{A}$. By Theorem 4.1, any $\gamma$-approximation algorithm for the MEDP-$m$ problem achieves the same approximation ratio $\gamma$ for the DL problem with $m$ homogeneous disks.

We now reduce the MEDP-$m$ problem to the DL problem. Given $G = (V, E)$ and $w(e)$, we create a DL instance, where $F = V$, $Q = E$, and $\phi(i, i') = w(e)$ for each edge $e = (i, i')$. Each file $i$ has unit size, and each of the $m$ disks has seek rate 1. Given any assignment $\mathcal{A}$ for the DL problem, we first produce an *integral* assignment $\mathcal{A}'$ according to Theorem 4.2. We then produce a solution $\mathcal{S}$ to the MEDP-$m$ problem as follows: vertex $i$ is assigned color $j$ if and only if it was assigned to disk $j$ in $\mathcal{A}'$. Clearly, the total seek time in $\mathcal{A}'$ is the same as the total weight of monochromatic edges in $\mathcal{S}$. This reduction shows that any hardness result for the MEDP-$m$ problem implies the same hardness for the DL problem. ∎

The next two theorems are direct implication of the previous theorem, using results for the MEDP-$k$ problem.

THEOREM 4.4. *For every fixed $\epsilon > 0$ and every $2 - \epsilon < \alpha \leq 2$, it is NP-hard to approximate the total seek time within a factor of $O(n^{2-\epsilon})$, over the DL instances with $n$ files, $|Q| = \Theta(n^\alpha)$ queries, and $m \geq 3$ disks.*

*Proof.* This follows immediately from the result of Kann, Khanna, Lagergren, and Panconesi [6], which states that it is NP-hard to approximate the MEDP-$k$ problem within a factor of $O(n^{2-\epsilon})$, with $n$ vertices, $\Theta(n^\alpha)$ edges and $k \geq 3$. ∎

THEOREM 4.5. *For the DL problem with two disks, one can approximate the total seek time within a factor of $O(\log n)$, where $n$ is the number of files. For the DL problem with three disks, one can approximate the total seek time within a factor of $\epsilon n^2$, for any constant $\epsilon > 0$.*

*Proof.* The result for two disks follows immediately by the approximation result for MEDP-2 in a paper by Garg, Vazirani, and Yannakakis [5]. The result for three disks follows immediately from Lemma 9 of the paper by Kann, Khanna, Lagergren, and Panconesi [6]. ∎

## 5 Combining Transfer and Seek Time

In this section, we consider the problem of minimizing the combined transfer and seek time. Recall that $\gamma = \max_{j \in D} \frac{\beta_j}{\alpha_j}$.

THEOREM 5.1. *The optimal assignment $\mathcal{A}$ that minimizes only the total transfer time, is an $(1 + \gamma/2)$-approximation to the problem of minimizing the total combined transfer and seek time.*

*Proof.* For any query $q = (i, i')$ let

$$T(q) = \max_j (r_{ij}\alpha_j + r_{i'j}\alpha_j)$$
$$S(q) = \max_j (\min\{r_{ij}\beta_j, r_{i'j}\beta_j\})$$

Notice that

$$\min\{r_{ij}\beta_j, r_{i'j}\beta_j\} \leq (r_{ij} + r_{i'j})/2 \cdot \beta_j \leq \gamma/2(r_{ij}\alpha_j + r_{i'j}\alpha_j).$$

We thus conclude that $S(q) \leq \gamma/2 \cdot T(q)$ for any assignment. Comparing $\mathcal{A}$ with any other assignment $\mathcal{B}$, we see that $\sum_{q\in Q} T^{\mathcal{A}}(q) \leq \sum_{q\in Q} T^{\mathcal{B}}(q)$, by the definition of $\mathcal{A}$. Also $\sum_{q\in Q} S^{\mathcal{A}}(q) \leq \gamma/2 \sum_{q\in Q} T^{\mathcal{A}}(q) \leq \gamma/2 \sum_{q\in Q} T^{\mathcal{B}}(q)$. Since $\sum_{q\in Q} P^{\mathcal{A}}(q) \leq \phi(i,i')(\sum_{q\in Q} T^{\mathcal{A}}(q) + \sum_{q\in Q} S^{\mathcal{A}}(q))$, we get that $\sum_{q\in Q} P^{\mathcal{A}}(q) \leq \phi(i,i')(1+\gamma/2)\sum_{q\in Q} T^{\mathcal{B}}(q) \leq (1+\gamma/2)\sum_{q\in Q} P^{\mathcal{B}}(q)$. ∎

THEOREM 5.2. *The* DATABASE LAYOUT *problem is NP-hard to approximate within $\rho \cdot \gamma$, for some constant* $0 < \rho < 1$.

*Proof.* We use a hardness result for a special case of the MEDP-$k$ problem shown in [9], which states for the class of graphs with unit-weight edges, it is NP-hard to decide whether the number of the monochromatic edges is 0 or at least $\rho|E|$, for some constant $0 < \rho < 1$. We reduce this case of the MEDP-$k$ problem to the problem of minimizing the combined transfer and seek time. Similar to Theorem 4.3, given a graph $G = (V, E)$, we introduce a DL instance with $|V|$ unit-size files, $k$ homogeneous disks, and $|E|$ queries with $\phi(i,i') = w(e) = 1$, for every edge $e = (i, i')$. And for each disk $j$, we set $\alpha_j = 1$ and $\beta_j = \gamma$. If there exists a coloring of $G$ for which the number of the monochromatic edges is 0, then there exists a corresponding assignment $\mathcal{A}$ of files to disks, such that $S(q) = 0$ for all queries $q$. Notice that for this assignment $\mathcal{A}$, $T(q) \leq \phi(i,i') \cdot \max\{r_i, r_{i'}\} = 1$ for every query $q = (i, i')$. Thus, the overall access time of $\mathcal{A}$ is no more than $|E|$. On the other hand, if for any coloring of $G$, the number of the monochromatic edges is at least $\rho|E|$, then by Theorem 4.3, for any assignment of the files to disks, the total seek time is at least $\gamma\rho|E|$. Thus, it is NP-hard to decide whether the total access time of the derived DL instance is at most $|E|$, or at least $\gamma\rho|E|$. The ratio of these two values implies the hardness result. ∎

## 6 Related Work

To the best of our knowledge, there is no earlier work providing a formal analysis of models that take into consideration the cost of co-accessing data objects in determining a database layout. Our results indicate that no algorithm can achieve an approximation ratio better than $\rho\gamma$ on all instances, unless $P = NP$. However, this does not exclude the search for heuristics that perform well in practice. Indeed, our work was motivated by that of Agrawal, Chaudhuri, Das, and Narasayya [2] who studied some greedy heuristics and showed empirically they out-perform full striping. To this extent, our formal analysis complements the earlier work.

On the other hand, our work indicates that striping files produce solutions that are close to the theoretical lower bound. Scheuermann, Weikum, and Zabback [10] studied the issue of the granularity of striping, i.e., the smallest fraction of a file to be considered when performing full striping.

There has been a significant amount of work in the area of storage administration and management. Since workload and query plans change over time, dynamic load balancing has been the focus of numerous earlier papers [1, 3, 4, 7, 8, 10, 11]. Due to space constraints, we refer the reader to [2] for a detailed survey of this line of work.

## References


[1] AutoAdmin Project at Microsoft Research. *research.microsoft.com/dmx/AutoAdmin.*

[2] S. Agrawal, S. Chaudhuri, A. Das, and V. Narasayya. Automating Layout of Relational Databases. In *Proceedings of 19th International Conference on Data Engineering*, 2003, pp. 607–618.

[3] G. Copeland, W. Alexander, E. Boughter, and T. Keller. Data Placement in Bubba. In *Proceedings of SIGMOD Conference*, 1988, pp. .99–108.

[4] H. Dewan, M. Hernandez, K. Mok, and S. Stolfo. Predictive Dynamic Load Balancing of Parallel Hash-Joins Over Heterogeneous Processors in the Presence of Data Skew. In *Proceedings of PDIS*, 1994, pp. 40–49.

[5] N. Garg,V.V. Vazirani, and M. Yannakakis. Multiway cuts in directed and node weighted graphs. In *Proceedings of 21st International Colloquium on Automata, Languages and Programming*, 1994, pp. 487–498.

[6] V. Kann, S. Khanna, J. Lagergren, and A. Panconesi. On the Hardness of Approximating Max k-Cut and its dual. *Chicago Journal of Theoretical Computer Science*, 1997.

[7] M. Lee, M. Kitsuregawa, B. Ooi, K. Tan, and A. Mondal. Towards Self-Tuning Data Placement in Parallel Database Systems. In *Proceedings of SIGMOD Conference*, 2000, pp. 225–236.

[8] L. Lee, P. Scheuermann, and R. Vingralek. File Assignment in Parallel I/O Systems with Minimum Variance and Service Time. *IEEE Transactions on Computers*, 1998.

[9] E. Petrank. The Hardness Of Approximation: Gap Location. In *Israel Symposium on Theory of Computing Systems*, 1993, pp. 275–284.

[10] P. Scheuermann, G. Weikum, and P. Zabback. Data Partitioning and Load Balancing in Parallel Disk Systems. *The VLDB Journal*, 7(1998): 48–66.

[11] R. Vingralek, Y. Breitbart, and G. Weikum. SNOWBALL: Scalable Storage on Networks of Workstations with Balanced Load. *Distributed and Parallel Databases*, 6(1998): .


## Appendix

### Pseudo-Code for the Greedy Algorithm

**Algorithm 1** GREEDY

Sort files in decreasing order of their frequency $\phi_i$.
Let $R_j = L_j$ for each disk $j$. $\{R_j$ denotes the available storage on disk $j$.$\}$
Initialize each $r_{ij}$ to be 0.
**while** There are files left unassigned **do**
  Consider the first file $i$ in the list
  Set $\theta_i = r_i(\sum_j 1/\alpha_j)^{-1}$.
  Set $q_{ij} = \theta_i/\alpha_j$. $\{$So $\sum_j q_{ij} = r_i$ and $q_{ij}\alpha_j = \theta_i$ for all $j$.$\}$
  **if** $\forall\ j,\ q_{ij} \leq R_j$ **then** $\{$In this case, file $i$ is completely assigned.$\}$
    $r_{ij} = r_{ij} + q_{ij}$
    Update $R_j = R_j - q_{ij},\ \forall\ j$
    Delete file $i$ from the list.
  **else** $\{$In this case, there exists a disk with $R_j = 0$, i.e., saturated.$\}$
    Find $\lambda = \min_j \frac{R_j}{q_{ij}}$
    Set $q_{ij} = \lambda q_{ij}$
    $r_{ij} = r_{ij} + q_{ij}$
    Update $R_j = R_j - q_{ij},\ \forall\ j$. $\{$Note one of the disks must be saturated.$\}$
    Update file $i$, so $r_i = r_i - \sum_j q_{ij}$. $\{$File $i$ will be considered in the next step again.$\}$
    Remove any saturated disks from consideration.
  **end if**
**end while**