On the effect of a configuration choice on the performance of a Mirrored storage system

Eitan Bachmat and Tao Kai Lam

E.Bachmat: Dept. of computer science Ben-Gurion U., Beer Sheva Israel, 84105.

email: ebachmat@cs.bgu.ac.il

T.K.Lam: EMC Corp., 228 south st., Hopkinton MA, USA, 01748.

email: tlam@emc.com

In this paper we study the effect of various configuration choices on the performance of a mirrored disk array. We introduce a large class of *semistructured* configurations which provide good seek minimization and bandwidth. We study their properties with respect to both online and offline mirroring algorithms. We also prove a theorem concerning the load balancing capabilities of random or semistructured configurations with several data copies. We show that almost the entire bandwidth of the storage system can be exploited once we increase the number of copies. In an appendix we explain a common error which appears in much of the literature concerning this subject

1 introduction

A *Mirrored* storage system is a storage system which contains more than one copy of each data element, different copies residing on different storage devices within the system. A *configuration* is a choice of data layout on the storage devices. In this paper we study the effects of a configuration choice on the performance behavior of a mirrored storage system. The performance of mirrored storage systems has been the subject of many pa-

pers. Papers which study seek minimization in mirrored systems, [12], [21], usually assume that the system consists of pairs of identical disks and the two data copies reside at identical locations on the two disks. Such a configuration is known as *physical mirroring* or *disk shadowing*. Other configurations which arrange the data copies at different locations or use more disks have not been explicitly explored in the context of seek minimization. In other studies which explore the bandwidth and load balancing capabilities of mirrored systems, [15], [31], [32], it is usually assumed that the configuration is *random*, meaning that the location of the two data copies is arbitrary and independent of each other, again other configurations have not been explicitly considered. As will be shown physical mirroring is indeed a good, but not optimal, configuration for seek minimization, on the other hand it is essentially the worst configuration at providing high bandwidth. Conversely, random configurations turn out to be optimal at providing high bandwidth but are essentially the worst configurations when it comes to seek minimization.

In this paper we introduce a large new class of configurations which we term *semistructured configurations*. A generic configuration in this class is a hybrid of the physically mirrored configuration and a random configuration. We will show that these configurations inherit the good properties of both the physically mirrored and random configurations. They are good at both seek minimization and load balancing. We will also consider some other configurations which have been suggested in the past such as interleaved declustering [33] and group rotate [15], as well as mirrored systems with more then two data copies. The latter have become increasingly feasible in recent years due to lower storage prices.

1.1 Organization of the paper and results

The paper is organized as follows,

In section 2 we we define some of the standard configurations which have been introduced in the literature and introduce the new class of semistructured configurations. We also consider briefly the seek function and user access pattern which we shall be using.

In section 3 we consider seek minimization as the target function. There are two types of algorithms which can be used to attain the goal of seek minimization, online algorithms and offline algorithms. We consider a very standard online algorithm, *the nearest server algorithm*. We present simulation evidence that the optimal configuration is obtained by exchanging the locations of the data in the outer and inner radial halves of the disk. We also show that semistructured configurations clearly outperform random configurations which seem to be the least desirable configurations from a seek minimization perspective.

We then consider offline algorithms. In this setting we show that semistructured configurations are precisely the class of configurations which allow optimal seek minimization. This result which is proved for any number of data copies is in fact the one which motivated the definition of this class of configurations and the entire paper. We also compare the performance of online and offline algorithms and show that they do not differ by much. We end the section with an explanation of why it is interesting to consider generic elements in the class of semistructured configurations and not just some well chosen examples which display good performance.

In section 4 we consider load balancing or bandwidth as the target function. Following the line of research initiated in [31] and [32], we introduce a new metric which measures the load balancing capabilities of a configuration. We show that random and semistructured configurations display optimal load balancing capabilities with respect to this metric. Another configuration, group rotate, which was introduced in [15] is also shown to be optimal. On the other hand Physical mirroring is shown to be very bad at load balancing. We then show that as the number of data copies

becomes large the entire potential bandwidth of the storage system can be exploited concurrently and efficiently provided that the configuration is random or semistructured. Experimental evidence suggests that 3 copies are in fact enough.

We conclude from all our results that semistructured configurations strike a good balance between the properties of physical mirroring and random configurations.

In an appendix to the paper we explain a common error which appears in many of the previous works on online algorithms for mirroring.

1.2 related work

The seek minimization properties of physical mirroring with 2 disks with respect to the nearest server algorithm have been studied extensively in [12] and [21]. As we noted there are many other papers which deal with this and related subjects which contain a basic error [5], [6], [7], [9], [10], [26], [25], [28], [29]. Among those whose results remain qualitatively correct despite the error one should note [25] which considers synchronous writes as well as reads and [26] which provides a comparison of online and offline algorithms all assuming physical mirroring.

Random configurations in the context of load balancing are explored in [32], [31] and [1]. The techniques which we use in our discussion of load balancing are similar to those which are presented in these references.

Load balancing and reliability properties of mirrored configurations have also been studied extensively in the specific context of multimedia applications, mostly using experimental methods [3], [14], [18], [19], [27]. Bandwidth is the metric of choice in these applications and seek minimization is not considered.

Chen and Towsley, [15], experimentally study the problem of

choosing a configuration for a mirrored system. Their results are summarized in Fig. 11 in loc. sit. one of the motivations of the current paper is to provide a theoretical framework which will help explain some of the phenomenon observed in [15]. In addition our results can be used to analyze some of the configurations they have not studied.

2 Configurations for mirrored systems

In this section we introduce the configurations which will be considered in the paper. Along with some well known configurations we introduce the notion of a semistructured configuration. We will also briefly discuss the seek function and user access pattern which will be used in our experiments and results.

2.1 configurations

Let n, h, d be integers such that nh is divisible by d. Let $L = nh/d$ A *configuration* with parameters (N, h, d), consists of an h by n matrix M whose entries are integers in the range $0, ..., L-1$ subject to the following constraints:

A) Each number $0 \in \{1, ..., L\}$ appears exactly d times in the configuration matrix.

B) The entries in any given column are distinct.

It will be convenient to index the rows and columns of the configuration matrix by $0, ..., h-1$ and $0, ..., n-1$.

The configuration should be interpreted as follows. The system is comprised of n disks, indexed $0, ..., n-1$. The system also has L *files* (data units) which are indexed $0, ..., L-1$. and there are d copies of each file on different disks.

Each disk is divided into h ring shaped radial zones, which can

each hold a single file. We index these radial zones from the outer most to the inner most by indexes $0, ..., h-1$. Each column of the configuration matrix describes a single disk. The k'th entry in the column is the index of the file whose data resides in the k'th radial zone of the disk.

To summarize, each column states which files reside on a given disk and in what order, starting from the outside circumference. this order is compatible with the standard indexing of physical blocks in disk drives.

We introduce some of the configurations which we will study in this paper When d is not explicitly noted we assume that $d = 2$.

1) Physical mirroring: (Folklore)

We define the h by d matrix $P = P^{h,d}$ by the formula

$P_{i,j}^{h,d} = i$

In this configuration all d disks contain identical data at identical locations.

2) Interleaved declustering: (introduced in [33])

Let h be even. We define the h by $(h+1)$ matrix $I = I^h$ by the formula

$I_{ij}^h = i + hj/2$ for $i < h/2$ and

$I_{ij}^h = I_{i-h/2,j-((i+1-h/2) \, mod \, h)+1}^h$ for $i \geq h/2$.

For example, when $h = 4$ we obtain the configuration

0 2 4 6 8

1 3 5 7 9

8 0 2 4 6

7 9 1 3 5

We note that each pair of disks share a single file in common in this configuration.

3) Group rotate: (introduced in [15])

We define the h by $2h$ matrix $G = G^h$ by the formula

$G^h_{i,j} = j + ih$ for $j < h$

$G^h_{i,j} = G^h_{i,j-h-i \bmod h}$ for $j \geq h$

When $h = 3$ we obtain

0 3 6 0 3 6

1 4 7 7 1 4

2 5 8 5 8 2

4) Chained declustering: (introduced in [22])

Let h be even. We define the h by n matrix $C = C^{h,n}$ by

$C^{h,n}_{i,j} = i + jh/2$ for $i < h/2$

$C^{h,n}_{i,j} = C^{h,n}_{i-h/2,j+1 \bmod n}$ for $i \geq h/2$

When $h = 4$ and $n = 4$ we get

0 2 4 6

1 3 5 7

2 4 6 0

3 5 7 1

2.1.1 Semistructured configurations

We introduce a large family of configurations which we call *Semistructured* We assume strictly for the purpose of simplicity that d divides h.

Definition : A configuration M is said to be semistructured if there exists a vector $Q = (q_0, ..., q_{N-1})$ with $0 \leq q_i \leq h - h/d$ such that the set $D_Q = \cup_{i=1}^{N} \left\{ M_{i,q_i}, M_{i,q_i+1}, ..., M_{i,q_i+(h/d)-1} \right\}$ equals the set $\{0, ..., L-1\}$.

If M is semistructured with respect to Q we let

$$B_Q = \cup_{i=1}^{N} \{(i, q_i), ..., (i, q_i + (h/d) - 1)\}$$

By definition the set of locations B_Q contains a single copy of each file.

We provide some examples of the newly defined semistructured configurations.

A) $P^{h,2}$ is semistructured with respect to two symmetrically related Q vectors, $Q_1 = (1, h/2+1)$, $Q_2 = (h/2+1, 1)$. More generally $P^{h,d}$ is semistructured.

B) I^h and $C^{h,n}$ are both seen to be semistructured with the choice $Q = 0$.

C) G^h is semistructured with the choice $Q_i = 0$ for $i = 0, ..., h-1$ and $Q_i = h/2$ otherwise.

The configuration

7 5 7

2 9 9

1 6 2

5 0 8

6 3 0

3 8 5

is an example of a "generic" semistructured configuration with a unique Q vector $(2, 3, 0)$.

while, The configuration

0 0

1 2

2 1

3 3

provides a very simple example of a non semistructured configuration.

2.2 Seek function and user access pattern

In order to study seek minimization we need to specify the amount of time it takes the disk to seek between files. A seek function has the general form $f(|i-j|)$ where i and j refer to the radial location indices of the files on the disk. f is obviously an increasing function since longer seeks take more time. In our simulations we use the linear seek function model $f(|i-j|) = \frac{|i-j|}{(h-1)}$. This model is by far the most popular model in analytic studies of storage system performance, [4], [12], [13], [21], [34] and others. Using other non linear functions will only change the numerical values of the computed average seek, but not the relative performance of different configurations. The normalization factor $\frac{1}{h-1}$ allows us to compare systems with different values of h since a full disk seek from the outer most file to the inner most file always takes 1 unit of time.

Since our goal is to study the effect of a configuration choice we will use the simplest and most popular model of a user's access pattern. We assume that the user sends requests independently of each other to uniformly distributed random locations in the system. This is a homogeneous activity model. Only reads profit from a mirroring algorithm, therefore we assume a read only access pattern.

3 Seek Minimization

In this section we consider the effect of a configuration choice on seek minimization. there are two types of seek minimization algorithms which may be employed. The first type is the family of *online* algorithms. These algorithms decide which data copy will be used to service a read I/O request based on the current position of the disk heads in the system. The prototypical example of an online algorithm is the *Nearest server* algorithm, NS for short, which at any given moment chooses the head closest to a data copy, to service the request. Such algorithms may be very efficient but they require knowledge of the head location of several disks in the system (those containing copies of the required data). Such information may be difficult to obtain in real time especially in distributed systems. This problem leads to the consideration of a second type of mirroring algorithms which are *offline* . In such algorithms, one chooses for each copy $0 \leq i \leq d-1$ of each file $0 \leq j \leq L-1$ a probability $p_{i,j}$ that a read request from file j will be serviced by the i'th copy of the file. When a read request to file j enters the system it is serviced by the i'th copy of the file with probability $p_{i,j}$ regardless of the current head positions in the system. The choice of probabilities can change periodically when new data on the user access pattern is received. Such algorithms are very easy to implement in distributed systems and require minimal control, so they can even be implemented outside the system. Online algorithms by their real time nature are very difficult to implement using external controls. In the next two subsections we consider both types of algorithms and how the choice of a configuration affects their performance. We note that both types have been commercially implemented in disk arrays.

3.1 Online algorithms

We describe the average seek of the NS algorithm with respect to a configuration M. We assume that $d = 2$. With the exception

of the work of Calderbank, Coffman and Flatto, [12], [13], all studies of online mirroring algorithms have been incorrect. This is explained in the appendix.

Recalling that the user access pattern we employ is an independent request model, the nearest server algorithm implemented on a system with configuration M is described by a Markov process, which we denote NS_M. The states of NS_M are given by the file positions of all disk heads. Such states correspond to vectors of length n with values in the range $0, ..., h-1$. The system is in state $\bar{S} = (s_0, ..., s_{h-1})$ if the head of the i'th disk is currently located at file s_i on the disk, where the files on the disk are indexed from the outermost to the innermost radially.

We refer the reader to [17] for the basic theory of Markov chains, including the notions of periodicity, ergodicity, stationary distributions and the like.

It is easy to show that for any configuration M, NS_M has no periodicities. For some configurations such as $P^{h,d}$ it is not ergodic. We will comment on this phenomenon shortly but for now assume that M is a configuration for which NS_M is ergodic. In this case there will be a unique stationary probability distribution μ_M on the finite set of all possible states S.

For $l = 0, ..., L-1$ and $k = 0, ..., d-1$ let $i_{l,k}, j_{l,k}$ be the locations of the different copies of the l'th file, that is, $M_{i_{l,k},j_{l,k}} = l$ for all $0 \leq k \leq d-1$. Let $S = (s_0, ..., s_{n-1})$ be the state of the system at some given time and let the new request be to file l. Since we assumed that the access probability is uniform, the probability that l will be chosen is $\frac{1}{L}$. let $r(S, l) = Min_k \ |s_{i_{l,k}} - j_{l,k}|$ be the seek distance, given the current state S, that the head chosen by the nearest server algorithm will cover to service the request to file l. the average seek of the NS algorithm with configuration M will be

$$E(M) = \sum_S \sum_l \frac{1}{L} \mu_M(S) r(S, l)$$

Since the number of states, h^N is usually very large it is usually very difficult to compute μ_M and hence this formula is not used for computations. Instead we resort to simulations of NS_M in which we record the average observed seek distance.

We still must address the issue of ergodicity. In non ergodic Markov processes there are several basic stationary distributions $\mu_1, ..., \mu_f$, corresponding to the different recurrent sets of states. In such cases we get f different formulas for the average seek, one for each stationary distribution. In terms of the performance of the system this means that the average seek may depend (probabilistically) on the initial state of the system. The Markov chains arising from various possible configurations are not necessarily ergodic.

For example if $h \geq d$ then $NS_{P^{h,d}}$ has $d!$ ergodic components, however they all turn out to be equivalent and to yield the same average seek distance hence the average seek is well defined and independent of the initial state of the system.

The following facts regarding ergodicity of the Markov chain can be proved:

1) It is easy to produce examples with $d = 4$ of configurations M which do not have a well defined average seek distance.

2) All explicit configurations appearing in this paper do have a well defined average seek distance.

3) It can be shown that a randomly chosen configuration has a well defined average seek with very high probability (as h tends to infinity).

4) We conjecture that any configuration with $d = 2$ has a well defined average seek. This conjecture can be proved for systems with 2 or 3 disks.

Given the evidence in items 2-4 this issue has no effect on the

results reported in this paper.

3.1.1 Results for 2 disk systems

We consider systems containing only two disks which contain the same files but not necessarily in the same order. Larger systems can be built from 2 disk systems by repeating the configuration, as in large shadowed systems in which disks are paired and physically mirrored. Such systems are commercially common.

Since we assume a uniform access pattern, permuting the names of the entries in a configuration M does not change the behavior of the system, hence we may assume that $M_{i,0} = i$. We denote by π the permutation that satisfies $M_{i,1} = \pi(i)$, hence the choice of a configuration amounts to choosing a permutation. If the chosen permutation is the identity we obtain physically mirrored disks.

As an example consider the case $h = 6$. In this case there are 720 possible permutations. Inverting the order of appearance of file copies on the second disk has no effect on the results since it is a distance preserving operation, thus we need to consider only 360 permutations. We simulated all 360 permutations, the permutation that produces the lowest average seek was $\pi = (3, 4, 5, 0, 1, 2)$ which gives $E(M) = 0.151$ Note that π corresponds to the chain declustered configuration $C^{6,2}$. $C^{8,2}$ was also the best among configurations with $h = 8$. The analogue for larger h is $C^{h,2}$, for $h = 20$ the average seek was 0.146. The identity for h=20 produced 0.162. We conjecture that $C^{h,2}$ is optimal for all values of h.

We note that the result for $C^{20,2}$ is lower then the average seek for an optimal online mirroring algorithm (not NS) which uses the physically mirrored configuration. The average seek for the optimal algorithm which was computed in [5] is 0.159.

We compared physical mirroring with generic semistructured configurations and completely random configurations. The generic

semistructured configurations were drawn using a random Q vector, writing the numbers $0, ..., L-1$ in consecutive order in B_Q and then randomly placing $0, ..., L-1$ at the other positions. Completely random configurations were drawn by choosing a completely random permutation. We chose for our initial experiments $h = 30$. We chose 1000 random configurations. For each configuration we simulated NS until the value of the average seek seemed to settle on a given value up to a 0.005 shift. The following list summarizes the results. The left column represents a range for the average seek and the right column counts the number of random configurations whose average seek fell within that range.

$0.175 - 0.180$ 3

$0.180 - 0.185$ 37

$0.185 - 0.190$ 145

$0.190 - 0.195$ 317

$0.195 - 0.200$ 350

$0.200 - 0.205$ 140

$0.205 - 0.210$ 8

We then chose 1000 generic semistructured configurations. The corresponding table looks as follows

$0.160 - 0.165$ 114

$0.165 - 0.170$ 793

$0.170 - 0.175$ 93

Physical mirroring yielded an average seek of 0.161. The computation for physical mirroring was initially performed in [12]. We cannot use directly the value of 0.1625 given in [12] since that is the value for very large h and since our NS procedure differs slightly from that of loc.sit. The average seek in physical mirror-

ing was incorrectly computed in [9] and other papers to be in the range 0.2-0.21. The reason for the error in [9] is explained in the appendix.

Our main observation is that each semistructured configuration yielded a lower average seek than each random configuration. We also observes that physical mirroring and $C^{h,2}$ are among the best semistructured configurations.

Repeating this experiment with $h = 100$ The range for the average seek of random configurations was 0.188-0.205. The range for generic semistructured configurations was 0.166-0.173.

We now try to qualitatively explain the data. We claim that as h tends towards infinity the average seek distance for almost all configurations will tend to a single limit, hence the performance of all randomly chosen configurations will tend to converge. In fact as h tends to infinity the Markov chain associated with a random configuration will converge to a unique Markov chain M_{rand}. The states of M_{rand} are points (x, y) in the unit square. The state transition in M_{rand} is as follows. A point (x', y') is uniformly chosen in the unit square. If $|x - x'| \leq |y - y'|$ we move from (x, y) to (x', y). Otherwise we transition to (x, y'). The transition cost is $MIN(|x - x'|, |y - y'|)$. The points x', y' are the continuous limit for the pair $\frac{i}{h-1}, \frac{\pi(i)}{h-1}$, where i is chosen uniformly and π is a random permutation.

M_{rand} is closely related to the CNN problem which was studied in [23] in the context of competitive algorithms. Simulations indicate that the average seek of M_{rand} is about 0.193, in line with our results on the finite approximation random configurations.

For generic semistructured configurations the limiting processes are different, since there is a relation between i and $\pi(i)$. More precisely if i is contained in B_Q then it's copy on D_1 is not in B_Q and vice versa. If we consider a family of generic semistructured configurations, with $h \longrightarrow \infty$ and $a = q_0/h$, $b = q_1/h$ fixed, the limiting process will be $M_{a,b}$ whose transition rule is the following.

Given a state (x, y) choose x' uniformly. If $x' \in [a, a+1/2]$ choose y' uniformly outside the range $[b, b+1/2]$, otherwise choose y' uniformly within $[b, b+1/2]$. Proceed as in M_{rand}. We did not study the dependency of the average seek of $M_{a,b}$ on a and b, but our previously stated empirical results suggested that the dependence is not too great and is within the rather narrow range of 0.166-0.173.

3.1.2 Results for systems with more disks

We conducted some experiments on systems with more than two disks. The results convey the same spirit as those of the two disk experiments, however the average seek numbers tend to be higher. The configuration I^8 which has 9 disks had an average seek of 0.175 and G^8, with 16 disks, yielded a result of 0.193.

We also performed empirical tests to measure the average seek of configurations on 10 disks with $h = 30$. Random configurations were in the range 0.197-0.214, with the majority around 0.205. As in the case of two disks we have a limit for the average seek of a random configuration which depends only on the number of disks in the configuration.

Generic semistructured configurations again performed better in the range 0.170-0.177 with the majority around 0.174. These numbers show that for both 2 and 10 disks semistructured configurations lie in terms of average seek about a third of the way between physical mirroring and random configurations. We do not expect the numbers to vary materially beyond 10 disks.

Finally we note that there are other seek minimization algorithms for which it would be interesting to repeat such experiments. The prime example being the *double cover* algorithm or DC for short, see [11] for a full description. In a worst case setting (competitive analysis), it is far superior to NS, however in the distributional setting which we consider, it is not as good. On a physically mirrored configuration it yielded an average seek of 0.18...

Offline mirroring algorithms are algorithms which do not require knowledge of the system's current state to make a choice of the copy which will service a request. We do allow the choice to be probabilistic. This leads to the following definition.

Definition: An *offline mirroring algorithm* consists of the choice of a matrix W, called the *weight matrix*, of size $n \times h$. W has the following two properties.

1) $w_{i,j} \geq 0$.

2) Let $(i_{l,1}, j_{l,1}), ..., (i_{l,d}, j_{l,d})$ be the locations of file l, that is, $M_{i_{l,k},j_{l,k}} = l$ for all $1 \leq k \leq d$, then $\sum_{k=1}^{d} W_{i_{l,k},j_{l,k}} = 1$, for all l.

Given W, a read request for data from file l will be directed towards the k'th copy of the file, which resides on disk $i_{l,k}$ in position $j_{l,k}$, with probability $W_{i_{l,k},j_{l,k}}$. Condition (2) ensures that the sum of the probabilities is 1.

If M is semistructured with respect to a vector Q, we can define a particular weight matrix by the rule $W_{i,j} = 1$ if and only if $(i,j) \in B_Q$. The offline mirroring algorithm which this weight matrix describes will be called a *partition* algorithm. When the semistructured configuration is physical mirroring then this algorithm coincides with the classical partition algorithm.

Since we assume that the activity of all files is equal we may normalize it to be 1. When h is fixed, all partition algorithms have the same average seek which is

$$E_h^{Part} = \frac{2d^2}{h^2(h-1)} \sum_{k=1}^{(h/d)-1} ((h/d) - k)k$$

regardless of the specific semistructured configuration used. It is easy to verify that as h tends to infinity the average seek tends

to $1/(3d)$.

Given a configuration M and an offline mirroring algorithm W it is easy to show [34] that the average seek over all disks in the system is

$$E(M,W) = \sum_{i=1}^{n} E_{D_i} = 2\sum_{i=1}^{n}(\sum_{m>j} W_{i,m}W_{i,j}f(m-j))/a_{D_i}$$

Here D_i refers to the i'th disk, E_{D_i} is the contribution of the i'th disk, f refers to the seek function and $a_{D_i} = \sum_{j=1}^{h} W_{i,j}/L$ is the activity of the i'th disk. Let $S(M,W) = E(M,W)/(2L)$ and correspondingly $S_{D_i} = E_{D_i}/(2L)$. For convenience We will use $S(M,W)$ to compare configurations and offline algorithms instead of the average seek $E(M,W)$, since they are proportional the comparison will not be affected. Let f be a seek function. All we will require is that f is increasing. Given f the seek time between the i'th and j'th files on the disk is $f(|i-j|)$. In the previous section we used the function $f(x) = x/(h-1)$.

The following result shows that semistructured configurations equipped with the partition algorithm are optimal among all pairs consisting of a configuration and an offline mirroring algorithm.

Theorem 1 *Fix h. Given a configuration M with parameter h and an offline algorithm W then $E(M,W) \geq E(Part)_h$, equality holds if and only if M is semistructured and W is a partition algorithm with respect to a Q vector.*

Proof: We consider the restriction of the configuration M and the weight matrix W to a single disk D_k in the system. The weight matrix restricted to the files on the particular disk can be described by a weight vector $\vec{w} = (w_1, \ldots, w_h)$ where $w_i = W_{k,i}$. The contribution of the disk to the seek estimate $S(M,W)$ will be denoted by $S(\vec{w})$, to emphasize the dependence on the vector w. We have

$$S(\vec{w}) = \frac{\sum_{i<j} w_i w_j}{\sum_i w_i} f(j-i)$$

We need the following simple lemma

Lemma 2 *For $0 < x \leq h$ consider $V(x)$, the set of all vectors $\vec{w}$ with entries $0 \leq w_j \leq 1$ and satisfying $\sum_i w_i = x$. A weight vector $\vec{w}$ that minimizes S among all weight vectors in $V(x)$ must satisfy the following conditions:*

There exists an index q, $1 \leq q \leq h - [x]$ such that

(1) $w_j = 1$ for $q \leq j \leq q + [x] - 1$
(2) Either w_{q-1} or $w_{q+[x]}$ equals $x - [x]$, and hence all other weights are zero.

That is, $\vec{w} = (0, \ldots, 0, x - [x], 1, 1, \ldots, 1, 0, 0, \ldots, 0)$ or $(0, \ldots, 0, 1, 1, \ldots, 1, x - [x], 0, 0, \ldots, 0)$.

Proof: Let $\vec{w}$ be any weight vector in $V(x)$ that does not satisfy the conditions in the lemma. Let r be the largest index for which the corresponding entry w_r is nonzero. Then there exists q, $0 < q < r$ such that $w_q < 1$ and $w_{q+1} = w_{q+2} = \ldots = w_{r-1} = 1$. Let $y = \min(1 - w_q, w_r)$ and denote by $\vec{v}$ the following perturbed weight vector

$$v_i = \begin{cases} w_q + y & \text{if } i = q \\ w_r - y & \text{if } i = r \\ w_i & \text{otherwise.} \end{cases}$$

One now easily verifies the following formula

$$\begin{aligned} & S(\vec{w}) - S(\vec{v}) \\ = & \frac{y}{x} \left(\sum_{j=1}^{q-1} w_j \left(f(r-j) - f(q-j) \right) + f(r-q) \min(w_q, 1 - w_r) \right) \\ \geq & 0. \end{aligned}$$

Equality holds if $w_j = 0$ for all $j < q$ and either $w_r = 1$ or $w_q = 0$. Hence the weight vector which attains the minimum must have the required form. *q.e.d*

Using Lemma 2, we obtain easily that the minimum seek for

vectors in $V(x)$ is

$$S(x) = \sum_{k=1}^{[x]} \left(1 - \frac{k}{x}\right) f(k)$$

Since the weight vector $\vec{w}$ of partition restricted to a single disk is of the form given in the lemma we see that the contribution of a single disk to the seek estimate of partition is $S(h/d)$, h/d being the weight of each disk in partition.

We wish to study the dependence of $S(x)$ on x.
Let $g(k) = f(k) - f(k-1)$ then

$$\begin{aligned} S(x) &= \sum_{k=1}^{[x]} \left(1 - \frac{k}{x}\right) \sum_{j=1}^{k} g(j) \\ &= \sum_{j=1}^{[x]} \left(\sum_{k=j}^{[x]} \left(1 - \frac{k}{x}\right) \right) g(j). \end{aligned}$$

we wish to find the minimum of $\sum_i S(a_i)$ over all sequences $a_1, a_2, \ldots, a_n$, where a_i are nonnegative real numbers such that $\sum_i a_i = L$. Let $H_j(x) = \sum_{k=j}^{[x]}(1 - k/x)$. It is easy to verify the following properties:

(1) $H_j(x)$ is an increasing function;
(2) $H_j(x) = 0$ if $x < j$;
(3) $H_j(x) = [x] - j + 1 - [x] - j + 1)([x] + j)/(2x)$ if $x \geq j$;

The following lemma deals with the restriction of the system to two disks.

Lemma 3 *Let $a < b$ be two nonnegative numbers and consider the sum $H_j(a) + H_j(b)$.*

(1) If a, b are both not integers, then there exists a', b' such that

$$H_j(a) + H_j(b) \geq H_j(a') + H_j(b')$$

a' or b' is an integer and $a + b = a' + b'$.

(2) If a, b are both integers, then

$$H_j(a) + H_j(b) \geq H_j(a+1) + H_j(b-1)$$

Proof:

(1) Case 1: $a < b < j$

$$H_j(a)+H_j(b) = \begin{cases} H_j(0) + H_j(a+b) & \text{if } a+b < 1; \\ H_j([a]+1) + H_j(a+b-[a]-1) & \text{otherwise.} \end{cases}$$

Case 2: $a < j < b$

$$H_j(a) + H_j(b) > H_j([a]+1) + H_j(b+a-([a]+1))$$

since $H_j(a) = H_j([a]+1) = 0$ and $H_j(x)$ is an increasing function.

Case 3: $j < a < b$ Let $x_1 = \min(a-[a], [b]+1-b)$ and $x_2 = \min([a]+1-a, b-[b])$. We consider the function $F(x) = H_j(a+x) + H_j(b-x)$ in the interval $[x_1, x_2]$. The second derivative of F is

$$-\frac{([a]-j+1)([a]+j)}{(a+x)^3} - \frac{([b]-j+1)([b]+j)}{(b-x)^3}$$

which is non-positive in the interval $[x_1, x_2]$. Hence, the function $F(x)$ attains its minimum when $x = x_1$ or $x = x_2$. In both cases, either $a+x$ or $b-x$ is an integer.

(2) Suppose $a \leq b$ are integers. The expression $H_j(a) + H_j(b) - H_j(a+1) - H_j(b-1)$ is always nonnegative when $a < j$. When $j > a$,

$$H_j(a) + H_j(b) - H_j(a+1) - H_j(b-1) = \frac{(b-a-1)(a+b)}{2a(a+1)b(b-1)}$$

This shows that $H_j(a) + H_j(b)$ is a minimum when $a = b$ or $a = b-1$.

q.e.d

To complete the proof of the theorem, consider a pair M, W consisting of an offline algorithm with weights W on a configuration M with n disks. Let $\sigma_0, \ldots, \sigma_{n-1}$ be the sum of weights on each of the n physical disks with respect to W. We have $\sum_i \sigma_i = L$. Fix j and consider the sum $\sum_i H_j(\sigma_i)$. By repeated application of Lemma 3(1) to pairs of σ_i's which are not integers, we can find $\eta_{i,j}$ such that all $\eta_{i,j}$ are integers, $\sum_i \eta_{i,j} = L$ and $\sum_i H_j(\eta_{i,j}) \leq \sum_i H_j(\sigma_i)$. Now, apply Lemma 3(2) repeatedly, to the smallest and largest of $\eta_{i,j}$'s. Each step lowers $\sum_i \eta_{i,j}^2$, thus, after a finite number of steps we obtain $\tau_{i,j}$, integers, such that $\sum_i \tau_i = L$ and for any i, l, $|\tau_{i,j} - \tau_{l,j}| \leq 1$, thus $\tau_{i,j} = L/N = h/d$ for all i. By lemma 3(2) we also have $\sum_i H_j(h/d) \leq \sum_i H_j(\sigma_i)$ for all j and consequently $nS(h/d) = \sum_i S(h/d) \leq \sum_i S(\sigma_i)$. By lemma 2 we also have $S(\sigma_i) \leq S_i(M, W)$. On the other hand $nS(h/d)$ is the seek achieved by the partition algorithm. From lemmas 3 and 2 it is also easy to see that equality holds only if M is semistructured and W is a partition algorithm. *q.e.d*

The theorem shows the advantages of a semistructured configuration over a random configuration. Homogeneous workloads in which the activity of all files is equal are commercially common. One example is given by, Teradata [33], a database application which uses hash functions to achieve complete homogeneity. In a random configuration the random workload will most likely be divided evenly across the entire disk while in a semistructured configuration all the activity will be handled by a more localized radial zone, say, the outer radial half of the disk. The performance comparison is thus the same as that of a uniformly random workload across an entire disk, compared with a uniformly random access workload restricted to half the disk.

Remark: We restricted ourselves to the case in which d divides h simply in order to simplify the definition of a semistructured configuration and the theorem. Both the definition and the theorem can be extended to the general case. Note that by the second lemma if d does not divide h the optimal seek minimization solution is not load balanced. For example for a pair of physically

mirrored disks with $h = 3$ the offline algorithm with weight vector $(1, 1, 0)$ has an average seek of $1/6$ while the load balanced weight vector $(1, 1/2, 0)$ yields an average seek of $2/9$.

We also note that the result in the theorem can be generalized to include synchronous writes. The inclusion of writes places some restrictions on the Q vector of optimal semistructured configurations, it must be symmetric with respect to the radial center of the disk.

3.3 Comparison of online and offline seek minimization

It is interesting to compare the seek minimization qualities of offline and online algorithms. For $h = 4$ we have the following simple claim which shows that surprisingly offline algorithms are at least as good if not better then online algorithms, at least when the access pattern is uniform.

Proposition 4 *If $h = 4$ and the access pattern is uniform then partition with respect to a semistructured configuration is optimal among all pairs consisting of a choice of a configuration and a mirroring algorithm, either online or offline*

Proof: Since partition only uses locations in B_Q to service requests all heads in the system point to different files, since no two copies of the same file are in B_Q. Since $h = 4$ and all activity levels are equal we see that the cumulative access probability of all the files to which the heads are pointing at any given moment is $1/2$. We see that with probability $1/2$ a read request will result in a head movement. For partition this head movement is always minimal, to a neighboring file. For any other algorithm the probability of head movement in any given state is at least $1/2$ and the head movement is at least one file, hence, no algorithm can have smaller average seek than partition. *q.e.d*

Remark: We note that the normalized average seek for partition is approximately $1/6$ when $h = 4$. For NS with physical mirroring

as the configuration, a simple Markov chain calculation gives an average seek value of approximately 0.188 when $h = 4$.

When $h > 4$ this result is no longer true as was shown in our computations of the average seek for $h = 20$. This has already been observed in [12]. As h becomes large, Physically mirrored disks and C_2^h yield better results than partition. We note however that both these configurations are for two disks and their generalization to many disks will be shown in the next section to suffer from poor load balancing. Partition on the other hand can be generalized to any number of disks with semistructured configurations which provide good load balancing.

The main advantage of online algorithms over offline algorithms lies in the former's better ability to adjust to rapidly changing workloads. On the other hand offline algorithms are much easier to implement since they do not require real time communication between different disks and processors.

3.4 Generic semistructured configurations and virtual storage

We saw that semistructured configurations are good at seek minimization in the presence of both online and offline algorithms. We will show in the next section that some particular semistructured configurations like I or G also provide very good load balancing. naturally one might wonder as to why we should bother with generic semistructured configurations as well.

Virtual storage disk arrays in which data is constantly being moved about the system are very common commercially. Such systems employ a garbage collection mechanism which continuously provides available disk spaces by eliminating obsolete data in the system. For example, one very popular garbage collection mechanism which optimizes write performance is known as LSF [30]. LSF and similar garbage collection mechanisms change

the configuration, so even if a system begins with an I or G configuration it will not remain in that state.

It is easy to insure that the garbage collector maintains only semistructured configurations by employing independent garbage collection in B_Q and it's complement. Since B_Q and it's complement consist of contiguous disk regions this will not hamper the work of LFS, for which the notion of contiguous disk space is important.

Our results and those of the next section show that the resulting generic semistructured configurations all share good performance which is superior to that of random configurations, hence system performance will be maintained throughout the lifetime of the system despite the constant configuration changes.

4 Load balancing

We turn our attention to a second performance metric, bandwidth. Since high bandwidth is achieved by exploiting the parallelism inherent in disk arrays, we equate high bandwidth and load balancing with the ability of the system to handle concurrently requests to many files. This ability does not depend on the internal order of the files in the disk. We will discuss the cases $d = 2$ and $d > 2$ separately

4.1 Load balancing with d=2

A *multi graph* is a graph in which more than one edge between vertices is allowed.

Definition: Given a configuration M, the *configuration graph (multi graph)* G_M is a graph whose vertices correspond to the disks of the system and whose edges correspond to files. A file connects the two disks on which it's copies reside.

Examples:

1) The configuration graph of I^h is the complete graph on $h+1$ vertices since every two disks share a single file in common.

2) The configuration graph of G^h is the complete bipartite graph on (h,h), i.e. the vertices of the configuration graph are divided into two disjoint sets of size h. There are no edges between members of the same set but there is an edge between any two members which are not in the same set.

3) The configuration graph of C_N^h consists of a cycle of length N, with each edge multiplied $h/2$ times.

4) The configuration graph of P^h consists of two vertices joined by h edges.

If a configuration M has parameters N, h then N is obviously the number of vertices of G_M and h will be the degree of the vertices in the graph. For any graph G we denote it's vertex set by $V(G)$ and it's edge set by $E(G)$.

The load balancing capabilities of random graphs have been studied in [31] and [32] assuming the complete graph as the configuration graph and for h fixed in [1]. Following [32] we consider the following.

Let G be a multi graph. A *schedule* f is a mapping $f : E(G) \longrightarrow V(G)$ such that $f(e)$ is incident to e for all edges $e \in E(G)$. The *load* of a schedule f, $L(f)$, is defined as $Max_{v \in V(G)} |f^{-1}(v)|$. The load of G, $L(G)$ is defined to be $Min_f\ L(f)$, where the minimum is taken over all schedules of G. To explain the terminology, consider the state of the system at some given time instance. Assume that there are outstanding read requests to a subset A of the files in the system. We may form the subgraph G_A of G_M which has the same vertex set but whose edges correspond only to files in A. A schedule f of G_A is simply a rule which decides which copy of each file in A will service the request to that file. The load is

the maximal number of file requests that any disk in the system will have to support.

For example the system will be able to concurrently support requests to a subset of files A if G_A has a schedule with load 1, or in other words $L(G_A) = 1$.

If Δ is a set of vertices of a graph G, we let $e(\Delta)$ denote the number of edges of G both of whose endpoints are in Δ. The *unavoidable load* of Δ, $L(\Delta)$ is defined to be $e(\Delta)/|\Delta|$ rounded up to the next integer value. We let $L^*(G) = Max_\Delta\ L(\Delta)$ where the maximum is taken over all subsets of $V(G)$. It is proved in [32] that $L(G) = L^*(G)$. As a special case this result means that $L(G) = 1$ if and only if all the connected components of G are either trees or contain a single cycle (*unicyclic*).

As a first application of these ideas we have the following

Proposition 5 *If h is even then, every configuration M can be converted into a semistructured configuration M' by permuting the internal order of files in the various disks. In particular M and M' share the same configuration graph.*

Proof: To construct a semistructured configuration via internal order permutations in the disks we need to find for each disk i a subset T_i of $h/2$ files from the disk such that the T_i are disjoint. Then the union $T = \cup T_i$ will consist of all files. We then shuffle the order of appearance of files in each disk so that the files in T_i appear contiguously and thus obtain a semistructured configuration. To prove the existence of such T_i we note that in terms of the configuration graph the choice of T_i corresponds to a schedule of G_M with load $h/2$. Let Δ be any set of vertices in G_M. Since there are h edges incident to any vertex in the graph and since the edges which are counted in $e(\Delta)$ have both endpoints in Δ we have $e(\Delta) \leq |\Delta|(h/2)$, the proposition now follows immediately from the equation $L(G_M) = L^*(G_M)$. *q.e.d*

The proposition essentially states that generic semistructured

configurations and random configurations have similar load balancing capabilities.

To measure the load balancing capabilities of families of configurations we introduce, following [1], the notion of the *load 1 bandwidth* of a family of configurations. Let G be a graph. Recall that in terms of the configuration graph outstanding read requests correspond to edges. Our model for picking a random set of read requests, or equivalently a random edge subset A will be the standard random subgraph model in which each edge of G is chosen independently with some fixed probability p. We let $G(p)$ denote the probability space on the subgraphs of G induced by this model.

Consider a family $\bar{G} = (G_n)$ of regular graphs, with degrees $h_n = h(G_n)$. Consider the random variable $\chi_{n,p} : G_n(p) \rightarrow \{0, 1\}$ whose value on A is 1 iff the subgraph of G induced by the edge set A is of load 1.

We define the *cycle threshold* of the family $\bar{G}$ as

$$CT(\bar{G}) = Sup\left\{\frac{1}{2}h_n p \mid \liminf_{n\rightarrow\infty} E(\chi_{n,p}) = 1\right\}$$

Note that $\frac{1}{2}h_n p|V(G_n)|$ is the expected number of edges in a subgraph in $G_n(p)$, and therefore $CT(\bar{G})$ represents the asymptotic ratio of edges to vertices for the maximum p for which the subgraph G_A in $G(p)$ will be of load 1 with probability approaching 1. Equivalently, the load 1 bandwidth describes the maximal portion of the system's disks which can be concurrently active serving request without queuing requests. it is our measure of the load balancing capabilities of families of configurations.

It is shown in [1] that for any family of configurations with h_n tending to infinity, we have $LB(\bar{G}) \leq 1/2$, which means that asymptotically no mirrored system with n disks can concurrently support more than $n/2$ randomly chosen read requests with high probability. This is still much better than the $O(\sqrt{n})$ requests that a non mirrored system can support. We will consider a family

of configurations to be good at load balancing if $LB(\bar{G}) = 1/2$.

4.1.1 Examples

We examine the load balancing capabilities of the previously introduced families of configurations. We note that the case of h fixed is treated in detail in [1] where explicit optimal families of configurations arising from number theory are presented. Thus, we will consider families in which h tends to infinity.

1) Let G_n be the configuration graph of a system with n disks composed of $n/2$ pairs of physically mirrored disks. For a subset A of the files the load is 1 only if at most 2 of the files in A reside on a given pair of physically mirrored drives. It can be easily shown that this criterion leads to the conclusion that a physically mirrored system with n disks can concurrently support no more than $O(n^{2/3})$ requests leading to to $LB(\bar{G}) = 0$.

2) Let $G_n = G(C_n^h)$, for some h. G_A will have load 1 only if at most 3 active files reside on any given disk This leads to the conclusion that an system interleaved declustering system with n disks can support only $O(n^{3/4})$ random read I/O concurrently and hence $LB(\bar{G}) = 0$.

3) Let $G_h = G(I^h)$, then G_n is the complete graph on $h + 1$ vertices. By the results of [16] (see also chapter 10 of [2]) on the phase transition for random graphs we have $LM(\bar{G}) = 1/2$.

4) Let $G_h = G^h$ be the family of group rotate configurations. By arguments which are very similar to the case of complete graphs we have $LB(\bar{G}) = 1/2$. For the sake of completeness we provide a sketch of the proof. Assume $p = c/h$ with $c < 1$. Let $B_{h,h}$ be the complete bipartite graph which as noted above is the configuration graph of G^h. Let D and E be the two sets of vertices with h elements each, that is vertices are numbered $1, ..., 2h$ and each vertex in the range $D = \{1, ..., h\}$ is connected to each vertex in the range $E = \{h + 1, ..., 2h\}$. We first compute the average

number of cycles in a random subgraph G_A of $B_{h,h}(p)$.

In a bipartite graph all cycles have even length. To specify a cycle of length $2k$ we choose an ordered k tuple $x_1, ..., x_k$ from D and an ordered k tuple $y_1, ..., y_k$ from E and consider the cycle $(x_1, y_1, x_2, y_2, ..., x_k, y_k)$, this representation is unique up to a cyclic permutation on both the x's and y's and orientation reversal, hence the number of cycles in $B_{h,h}$ is
$(h(h-1)...(h-k+1))^2/2k < h^{2k}/2k$.

The probability of existence of the specified cycle in G_A is $p^{2k} = c^{2k}/h^{2k}$ since it has $2k$ edges, so the expected number of cycles of size $2k$ in H is bounded by $c^{2k}/2k$ and hence if $c < 1$ the total expected number of cycles is dominated by a converging geometric series and hence is bounded independently of h.

In our situation bad components of G_A with load greater then 1 are those containing at least two cycles. The component must therefore include either two cycles with a common vertex or edge or a pair of disjoint cycles connected by a path. We will show that the probability of such elements emerging inside a component of G_A tends to zero as h tends to infinity. We concentrate on the case of two cycles with a common vertex or edge, the other case being nearly identical. We can view a pair of cycles with a common edge as consisting of a cycle and two points, not necessarily distinct, on the cycle with a path consisting of l vertices not in the circle, also connecting them. The description is not unique. but that will not affect our calculations. For a cycle of size $2k$ there are $4k^2$ choices for pairs of vertices. Given a pair (z_1, z_2) and assuming z_1 is in D and z_2 in E (the other cases are again very similar) we choose a path with $l = 2j$ (l must be even) vertices by specifying ordered sets $x_1, ..., x_j$ in D and $y_1, ..., y_j$ in E, the path is then given by $z_1, y_1, x_1, y_2, x_2, ..., ..., y_j, x_j, z_j$ and has $l+1$ edges. The number of paths is then $(h(h-1)...(h-j+1))^2 < h^{2j} = h^l$ and the path probability is c^{l+1}/h^{l+1} hence the expected number of paths of length l between pairs of points on a cycle of length $2k$ is bounded by $(2k)c^{2k+l}/h$. If $c < 1$ then summing over all k's

and l's we obtain a bound of a/h for some constant a, hence the expected number of components with at least two cycles tends to zero. This shows that $LB(\bar{G}) > (1/2)h(c/h) = (1/2)c$ for all $c < 1$ which completes the argument.

5) Families of random and semistructured configurations. It is easy to show using methods similar to those of the previous item that families of random configuration graphs with h tending to infinity have load 1 balance equal to $1/2$ and by proposition 5 the same holds for families of generic semistructured configurations.

To Summarize, the families I^h, G^h, families of random configurations and generic semistructured configurations are all asymptotically optimal load balancers while P^h and C_n^h configurations are not.

4.2 Load balancing for $d > 2$

When $d > 2$ the configuration graph is replaced by a similarly defined *configuration hyper graph*. An alternative description is via a bipartite graph G with vertex set (D, F), with the set D having n vertices representing the disks in the system and F having nh/d vertices representing the files. A vertex $i \in D$ is connected via an edge to an element $j \in F$ if file j has a copy on disk i. At a given time instance we assume that we have a subset A of files for which there are read requests and consider accordingly the subgraph G_A of G. In this setting all requests in A can be supported concurrently if and only if the graph G_A has a *set of distinct representatives*, SDR for short, which is a matching between all elements of F and elements of D. For a subset B of A let $N(B)$ denote the subset of D consisting of the neighbors of B. By Hall's matching theorem, see [20], an SDR exists iff for all subsets B we have $|B| \leq |N(B)|$.

One application is the following proposition

Proposition 6 *If $d = h$ then for any given configuration M the*

set of all files can be supported

Proof: If $d = h$ then the graph G is a bipartite d regular graph and hence by Hall's Theorem [20], there exists a perfect matching for G. *q.e.d*

We may define the load 1 bandwidth of M exactly as before. Our main result states that for random configurations, as the number of copies d grows the load 1 bandwidth approaches 1, which means that the system can exploit it's entire potential bandwidth.

We use the following notation Let n be an integer, c some constant satisfying $c \leq 1$ and define $k = cn$. We will assume that k is an integer.

Theorem 7 *(1) If* $c < e^{-\frac{2}{d-1}}$ *then*

$$\lim_{n\to\infty} Pr(SDR \ \ exists) = 1$$

(2) If $c > 1 - e^{-dc}$ *then*

$$\lim_{n\to\infty} Pr(SDR \ \ exists) = 0$$

Proof: We shall first prove statement 1. By Hall's basic matching theorem [20], we can find a matching of all elements of A with elements of D if for any subset $I \subset M$ we have $|N(I)| \geq |I|$. Therefore the probability that an SDR matching will not be found is bounded by

$$\begin{aligned} Pr(\text{no SDR matching}) &\leq \sum_{I\subset A} Pr(|N(I)| < |I|) \\ &= \sum_{m=2}^{k} \binom{k}{m} Pr(|I| = m \text{ and } |N(I)| < |I|) \end{aligned}$$

If $|I| = m$ we can think of $N(I)$ as the image of a map $f : \{1, \ldots, dm\} \longrightarrow D$. We have

$$
\begin{aligned}
Pr(\text{no SDR matching}) &\leq \sum_{m=2}^{k} \binom{k}{m} \frac{|\{f \ : \ |image \ \ f| \leq m-1\}}{n^{dm}} \\
&= \sum_{m=2}^{k} \binom{k}{m} \sum_{l=1}^{m-1} \frac{|\{f : \ \ |image \ \ f| = l\}|}{n^{dm}} \\
&\leq \sum_{m=2}^{k} \binom{k}{m} \sum_{l=1}^{m-1} \binom{n}{l} \left(\frac{l}{n}\right)^{dm}. \qquad (1)
\end{aligned}
$$

We first show that for $c < e^{-\frac{2}{d-1}}$

$$
\lim_{n\to\infty} \sum_{m=2}^{k} \binom{k}{m} \binom{n}{m-1} \left(\frac{m-1}{n}\right)^{dm} \longrightarrow 0
$$

Let us compare successive terms in the sum. Denote by e_m the summand corresponding to the index m. One easily verifies that $e_2 < 1/n$. We have

$$
\frac{e_{m+1}}{e_m} = \frac{[cn]}{m+1} \frac{n-m+1}{m} \left(\frac{m}{m-1}\right)^{dm} \left(\frac{m}{n}\right)^{d}
$$

Since e_2 tends to zero and the successive ratios in the cases of interest to us will be bounded by some fixed constant we may assume that m is large hence we may replace with arbitrary accuracy the expression $\frac{e_{m+1}}{e_m}$ by

$$
h_m = \frac{cn-m}{m} \frac{n-m}{m} e^d \left(\frac{m}{n}\right)^{d}
$$

We then have

$$
h_m < \frac{cn^2}{m^2} e^d \left(\frac{m}{n}\right)^{d} = e^d c \left(\frac{m}{n}\right)^{d-2}
$$

We shall denote the right hand side by j_m consider now

$$
r_m = e_2 \prod_{l=2}^{m-1} j_l
$$

Let us examine r_{cn}.

$$
\begin{aligned}
r_{cn} &\equiv \prod_{l=2}^{cn-1} j_l \\
&\equiv e^{dcn} c^{cn} (cn!)^{d-2} / n^{cn(d-2)} \\
&\equiv e^{dcn} c^{cn} (cn)^{cn(d-2)} / (n^{cn(d-2)} e^{cn(d-2)}) \\
&= (e^2 c^{d-1})^{cn}
\end{aligned}
$$

where $\equiv$ stands for equality up to a rational function (quotient of polynomials) of n. Thus, if $c < e^{-2/(d-1)}$ then r_{cn} is exponentially small. from the definition one can easily verify that we can find a constant b such that for $n >> 0$ $j_l < 1/2$ for all $l < bn$, therefore the sum $\sum_{m=2}^{bn} e_m$ is dominated by a convergent geometric series whose first term e_2 tends to zero. We note that the sequence j_l is increasing and hence the sequence $r_{bn}, \ldots, r_{cN}$ is either decreasing ($j_{cn} < 1$) or decreases and then increases. In either case we get that for all l satisfying $bn < l < cn$ we have $r_l < \max(r_{bn}, r_{cn})$. Since both are exponentially small the sum of terms tends to zero. We now show that under the condition $c < e^{-\frac{2}{d-1}}$ we have

$$
(1+\epsilon)^{(m-1)-l} \binom{n}{l} \left(\frac{l}{n}\right)^{dm} < \binom{n}{m-1} \left(\frac{m-1}{n}\right)^{dm}
$$

for $l < m-1$ and some $\epsilon > 0$. Let g_l be the left hand side of the last inequality. Consider successive quotients $t_i = g_{i+1}/g_i$ for $l < i < m$. Then

$$
t_i \equiv e^{dm/i}(N-i)/(i+1) > e^d \frac{1 - e^{-\frac{2}{d-1}}}{e^{-\frac{2}{d-2}}}
$$

where $\equiv$ denotes equality up to an arbitrarily close to 1 constant. It is easy to verify for small d that the right hand side is greater then one, while asymptotically

$$
\frac{1 - e^{-\frac{2}{d-2}}}{e^{-\frac{2}{d-1}}} \equiv \frac{2}{d-1}
$$

hence $t_i \equiv 2e^d/(d-1) > 1$ which establishes the claim. Thus, the

sum

$$\sum_{l=1}^{m-1} \binom{n}{l} \left(\frac{l}{n}\right)^{dm}$$

is dominated by an increasing geometric series whose top element is e_m hence we only have to consider the sum of the e_m and we are done proving statement 1.

We now sketch the proof of statement 2. We note that the probability as n tends to infinity that an element in D will be in $N(A)$ is $1 - e^{-dc}$ and hence the expected size of $N(A)$ is $(1 - e^{-dc})n$ which should be more then cn for an SDR matching to exist. We also have to compute the variance of $N(A)$ to show that with probability tending to 1 the size of $N(A)$ is close to the average. The size of $N(A)$ is the sum of the random variables X_i, where X_i takes the value 1 if i is in the image and zero otherwise. Since the covariance of pairs X_i, X_j is easily shown to be negative the variance of the sum is smaller then in the case of independent variables by [17] theorem 9.5.2. An application of Chebyshev's inequality ends the argument. *q.e.d*

The lower bound $e^{-2/(d-1)}$ is certainly not best possible. A simple inspection of inequality (1) shows that we can improve the bound to $\frac{\exp(-2/(d-1))}{1-(d+1)\exp(-d)}$ using essentially the same argument and noting that one can consider only maps f which are onto and in fact each element has an inverse image of size at least 2. This is still far from the correct value which seems to be much closer to the upper bound. Experiments show that already for $d = 3$ the threshold c is about 0.91 so almost all disks are concurrently usable.

5 Conclusions

We have introduced a large class of configurations, the semistructured configurations. We proved that configurations in this class are precisely those which allow optimal offline seek minimiza-

tion. We showed that generic semistructured configurations and certain explicit families within this class such as group rotate and interleaved declustering allow good seek minimization, both offline and online, and also allow good load balancing. Generic semistructured configurations can also be easily maintained in the context of virtual storage systems. These results suggest that mirrored storage systems should be configured using These configurations.

We also showed that by considering more than two data copies the storage array can use all it's bandwidth at maximum efficiency, experiments suggest that 3 data copies are sufficient.

the present work considers both bandwidth and seek minimization which is related to response time. Optimizing bandwidth and optimizing response time are not the same. In fact, in many cases optimization procedures like striping which improve bandwidth also increase response time. This paper can be viewed as part of a larger study of the relation between bandwidth and response time optimization with the goal of improving both. We hope to return to this theme in future work.

Acknowledgments : We would like to heartily thank D.Berend and D.Arnon for very helpful conversations. We also thank Oren Weiss, Oded Glinanski, Alon Ozer, Nir Levin and the entire storage systems class of spring 99 at BGU for providing the actual simulation data used throughout the paper.

References

[1] N. Alon and E. Bachmat, regular graphs whose subgraphs tend to be acyclic, *Proceedings of Eurocomb*, 2003.

[2] N. Alon, J.H. Spencer and P. Erdos, *The probabilistic method*, John Wiley and sons, 1992.

[3] S.V. Anastasiades, K.C. Sevcik and M. Stumm, Maximizing throughput in replicated disk striping of variable bit-rate streams, *Usenix Tech. conference*, 2002.

[4] O.I. Aven, E.G. Coffman and Y.A. Kogan, *Stochastic analysis of storage systems*, D.Reidel Publishing, 1987.

[5] A. Bestavros, IDA-based redundant arrays of inexpensive disks, *Proceedings of the IEEE conference on parallel and distributed information systems*, 1991.

[6] A. Bestavros D. Chen and W. Wong, The reliability and performance of parallel disks, *Bell labs technical report, 45312-891206-01TM*, 1989.

[7] R. Barve, E. Shriver, P.B. Gibbons, B.K. Hillyer, Y. Matias and J.S. Vitter, Modeling and optimizing throughput of multiple disks on a bus, *Proceedings of SIGMETRICS*, 1999.

[8] M. Buddhikot and G. Parkular, Distributed data layout, scheduling and playout control in a large scale multimedia storage server, *technical report WUCS-94-33 Dept. of CS, Washington U., St. Louis MO*, 1994.

[9] D. Bitton and J. Gray, Disk shadowing, *Proceedings of the 14th VLDB conference* , pages 331-338, Los Angeles, Aug. 1988.

[10] D. Bitton, Arm scheduling in shadowed disks, *Proceedings of the IEEE spring Compcon* , pages 132-136, Feb. 1989.

[11] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*, Cambridge university press, 1998.

[12] A.R. Calderbank, E.G. Coffman and L. Flatto, sequencing problems in two-server systems, *Math. Oper. research* , Vol. 10(4), pages 585-598, Nov. 1985.

[13] A.R. Calderbank, E.G. Coffman and L. Flatto, Optimum separation in a disk system with two read/write heads, *Journal of the ACM*, Vol. 31, pages 826-838, 1984.

[14] M.S. Chen, H.I. Hsiao, C.S. Li and P.S. Yu, Using rotational mirrored declustering for replica placement in a disk array based video server, *In Proceedings of the 3rd Intl. Conf. on multimedia,* 121-130, 1995.

[15] S. Chen and D. Towsley, A performance evaluation of RAID architecture, *IEEE transactions on computers* , Vol. 45, No. 10, 1116-1130, Oct. 1996.

[16] P. Erdos and A. Renyi, On the evolution of random graphs, *Magyar Tud. Akad. Mat. Kut. Int. Kozl*, Vol. 5, 17-61, 1960.

[17] W. Feller, *Introduction to probability theory and applications Vol.1*, Wiley and sons, 1950.

[18] J. Gafsi and E.W. Biersack, Modeling and performance comparison of reliability strategies for distributed video servers, *IEEE Trans. on parallel and distributed systems*, vol. 11(4), 412-430, 2000.

[19] L. Golubchik, J.C. Lui and R.R. Muntz, Chained declustering: Load balancing and robustness to skew and failures, *In proceedings of the 2nd Intl. workshop on research issues in data Engineering*, 88-95, 1992.

[20] P. Hall, On representatives of subsets, *Journal of London Mat. Soc.*, Vol. 10, pages 26-30, 1935.

[21] M. Hofri, Should the two headed disk be greedy?-Yes, it should, *Infor. Proc. Letters*, Vol. 16, pages 83-85, 1983.

[22] H. Hsiao H. and D. Dewitt, Chained declustering: A new availability strategy for multiprocessor database machines, *In Proc. 6th international data engineering conference* , Los Angeles, Feb 1990.

[23] E. Koutsoupias and D.S. Taylor, The CNN problem and other k-server variants, *Proceedings of STACS 2000*, pages 581-592, 2000.

[24] T. Kwon and S. Lee, Data placement for continuous media in multimedia DBMS, *In proceedings of the Intl. Workshop on multimedia database management systems*, 1995.

[25] R.W. Lo and N.S. Matloff, A probabilistic limit on the virtual size of replicated disk systems, *IEEE transactions on knowledge and data engineering*, Vol. 4(1), pages 99-102, 1992.

[26] M. Manzur-Murshed and M. Kaykobad, Seek distances in two headed disk systems, *Info. Proc. Letters*, Vol. 57(1), pages 205-209, 1996.

[27] A. Mourad, Doubly striped disk mirroring: Reliable storage for video servers, *Multimedia, tools and applications*, vol.2, 253-272, 1996.

[28] Y. Manolopoulos and A. Vakali, Seek distances in disks with two independent heads per surface, *Information processing letters*, vol 35, 37-42, 1991.

[29] Y. Manolopoulos Y. and A. Vakali, Exact analysis of expected seeks in mirrored disks, *information processing letters*, vol 61, 325-329, 1997.

[30] M. Rosenblum and J. Ousterhout, The design and implementation of a log structured file system, *ACM transactions on computer systems*, Vol. 10(1), pages 26-52, 1992.

[31] P. Sanders, Reconciling simplicity and realism in parallel disk models, *Proceedings of SODA 2001*, pages 67-76, 2001.

[32] P. Sanders, S. Egner and J. Korst, Fast concurrent access to parallel disks, *Proceedings of SODA 2000*, pages 849-858, 2000.

[33] Teradata, DBC/1012 database computer system manual release 2.0, *Doc. No. c10-0001-02*, Teradata Corp., Nov 1985.

[34] C.K. Wong, *Algorithmic studies in mass storage systems*, computer science press, 1983.

6 Appendix

We consider the problem of calculating the average seek of NS_M, the nearest server algorithm with respect to a configuration M. Our main observation is that the stationary distribution of NS_M is not uniform even though the access pattern is.

Consider for example a physically mirrored system with two disks and $h = 3$. It is easy to verify that the probability of the states $(0, 1)$ and $(1, 2)$ in the stationary distribution is $1/4$ while the probability of $(0, 2)$ is $1/2$. The average seek can then be calculated to be $1/6$. This distribution though is not typical and is strongly influenced by the fact that h is very small.

For a physically mirrored systems with $d = 2$ and large h the stationary probability and hence the average seek have been computed in [12]. In particular the stationary distribution is plotted in that paper. It is obvious from the graph that the stationary distribution is highly skewed though the distribution of the requests is uniform.

This may seem at first counter intuitive but may be easily verified either via simulations or simply by calculating small examples. The heads tend to hover around the center of the disk more frequently then around the edges.

In [9] a computation of the average seek was made assuming that the stationary distribution is uniform. The computation yielded an average seek of $1/(2d + 1)$. This result is unfortunately incorrect. In the case of $d = 2$ the correct result as computed in [12] is about 0.161, not 0.2.

Looking back at our computations of the average seek of a random configuration we may notice that the value $1/(2d + 1)$ provides a good approximation of the average seek for these configurations. The reason is that the Markov chains associated with such random configurations tend to have a much more uniform stationary

distribution and hence the calculation of [9] nearly applies.

Since the appearance of [9], the computation for physically mirrored systems, has also been carried out (sometimes independently) with the same error in [5], [6], [7], [9], [10], [26], [25], [28], [29]. The error is fatal only to a minority of the papers, but some revision of some of the results is needed in all of them. In these papers the error has been "generalized" to settings which include writes, non linear seek functions and others. The error has been validated experimentally in at least 3 papers. In addition other papers use the computation to compare NS with partition. Since partition has an average seek of $1/3d$ they conclude that partition is better. Following simulations, the correct average seek for physically mirrored systems with NS and $d \leq 10$ is always a little less then $1/3d$, hence slightly better than partition. We conjecture that this is the case for all d but that NS and partition have asymptotically (as d becomes large) the same performance.

References **To date there appears to be no correct analysis, analytic or experimental, of average seek time in physically mirrored storage systems with more than two disks which employ the NS algorithm**

Anyone attempting to tackle this problem should read (and not just cite) [12] and [13] first.

acknowledgments : The authors would like to thank Jim Gray for the encouragement he provided us in writing this appendix. Despite the mathematical error [9] is an inspiring paper.