Disk Performance Enhancement through Markov-based Cylinder Remapping *

Robert Geist
Darrell Suggs Robert Reynolds
Shardul Divatia Fred Harris
Evan Foster Priyadarshan Kolte

Department of Computer Science
Clemson University
Clemson, SC 29634-1906
phone: 803-656-2258
email: rmg@cs.clemson.edu

Abstract

A scheme for disk subsystem performance enhancement that is based on (virtual) cylinder remapping is proposed. A natural workload on a real system is measured, and statistical tests are used to determine that disk accesses are appropriately modeled by a first order Markov chain. Maximum likelihood estimators of the Markov model parameters are used in a simulated annealing algorithm to find a permutation of the (virtual) cylinders that substantially reduces expected seek distance. This permutation is then installed in a real system and tested under a workload that is stochastically generated from the Markov model. The proposed scheme is seen to offer a 25.6% reduction in mean service time when compared to the original (unmapped) cylinder arrangement.

keywords: Disk subsystem, Cylinder mapping, Markov models, Simulated annealing, UNIX^{TM}, DG/UX^{TM}

1. INTRODUCTION

Although CPU speeds of workstations and minicomputers have increased by at least an order of magnitude in the last five years, most of us in the university environment have not witnessed an accompanying order of magnitude increase in overall computer system performance. The reality of using heavily loaded systems is that we compute at disk speed, not CPU speed, and the performance of disk hardware has simply not kept pace. As a result, the possibility of performance-enhancing modifications to disk device drivers and file system structures continues to receive substantial attention from operating system designers.

We have previously investigated *disk scheduling algorithms* [1, 2, 7, 8, 9, 10] which are used to decide which of a queue of pending disk requests should be served next. The conclusions of those studies can be briefly summarized as follows:

- On heavily loaded systems, we can expect good scheduling to reduce average request waiting time by as much as 50% from that experienced under first-come, first-served scheduling.
- The best overall performance appears to be provided by the WSCAN algorithm [10]:
  1. Maintain a preferred read/write head direction, in (towards hub) or out (towards edge).
  2. Serve next the physically closest request in the preferred direction, unless a request falls in a very small "window" behind the head, in the opposite direction. In this case, serve the request in the window, but do not change the window location or the preferred direction.
  3. If no requests exist in the preferred direction, change the preferred direction.

The WSCAN algorithm has been independently tested by Data General, and will be incorporated in a forthcoming release of their DG/UX operating system.

Another approach to disk subsystem performance improvement has been suggested recently by Vongsathorn and Carson [12]. They conjecture that seek times could be substantially reduced if the arrangement of information on the disk cylinders "matched" the natural arrangement of requests generated by the workload. They observe that on a disk with N cylinders the sequence of cylinders visited can be approximately

*This work was supported in part by an equipment donation from Data General Corp.
^{TM} UNIX is a trademark of AT&T.
^{TM} DG/UX is a trademark of Data General Corp.

modeled by a discrete-time Markov chain with N states. Using their notation, we let π_i denote the steady-state probability that the read/write head is on cylinder i, and let ρ_{ij} denote the conditional probability that the head moves next to cylinder j, given that it is on cylinder i. Minimizing expected seek distance then amounts to finding that permutation, P, of {1,2,...,N} that minimizes

$$E[P] = \sum_{i=1}^{N} \sum_{j=1}^{N} |P(i) - P(j)| \pi_i \rho_{ij} \,, \qquad (1)$$

and then remapping the cylinders according to that permutation. Note that any such permutation remapping can be accomplished by a sequence of simple pairwise exchanges of cylinder contents.

Nevertheless, the number of permutations to be considered in minimizing (1) is $N!$, where a reasonable value of N is 2000. Since $2000! \approx 10^{5736}$, which dwarfs the number of atoms in the universe, the problem of finding this optimum permutation is clearly intractable. Vongsathorn and Carson resort to an approximation: they assume independent cylinder access, i.e. $\rho_{ij} = \pi_j$, and therefore they need only minimize

$$E[P\,|independent] = \sum_{i=1}^{N} \sum_{j=1}^{N} |P(i) - P(j)| \pi_i \pi_j \,. \quad (2)$$

From [6] the permutation P that minimizes (2) is the so-called *pipe organ arrangement* in which the most frequently accessed cylinder is in the middle, the next two most frequently accessed cylinders are adjacent to the first and on opposite sides of it, and so on. They proceed to implement a fault-tolerant cylinder remapping scheme, based on pairwise cylinder exchange, which provides this pipe organ arrangement.

The simplifying assumption of independent cylinder access is a point of contention among system designers. The designers of Data General's DG/UX operating system argue that cylinder sequence dependencies are extremely important [4]. They find that, in typical user workloads, access to a file index node (inode) is often immediately followed by access to the blocks of the file itself. For this reason they partition file systems into Disk Allocation Regions (DAR's) which contain both inodes and the associated contiguous file blocks. This design might be regarded as a heuristic attempt to minimize (1).

The validity of the independent cylinder access assumption is clearly workload and architecture dependent, but, for any particular system, it is still best examined in two phases:

1. Do we have real data from our system that allows us to reject the independence assumption in favor of a specific Markov model?
2. Does 1 matter, i.e., even if the answer is yes, can we improve significantly upon the pipe organ arrangement?

In this paper we present a case study of a real workload on a real system for which the answer to both questions is a qualified "yes." We describe a workload measurement and testing technique that allows us to determine when to reject independence in favor of a Markov model, and we describe an approach to the optimization problem, (1), that allows significant performance gains.

The remainder of the paper is organized as follows. In section 2 we describe the experimental platform, the workload, and the measurement facility. In section 3 we describe the Markov model derived from our measurement, the test for model order, and the results of its application to our measured workload. In section 4 we describe a technique for finding cylinder permutations, P, that "match" the measured workload in the sense of minimizing expected seek distance, (1). Section 5 contains a description of the installation of our cylinder remapping facility and test results from its use in a real system. It also enumerates some implementation issues that must be addressed in a further study. Conclusions follow in section 6.

2. WORKLOAD

Our experiment was performed on a Data General AViiON^{TM} AV 4000 running the DG/UX 4.30 operating system, a symmetric multiprocessor implementation of UNIX. The test disk was a 179 Mbyte SCSI drive with an Adaptec AIC-6250 Protocol Chip as bus interface. This architecture presented two obstacles to remapping physical cylinders. First, SCSI drives attempt to hide physical cylinder layout from the operating system, and so extensive testing was necessary to determine the layout. Second, the extensive testing revealed a most inconvenient physical structure: on this 179 Mbyte disk, the number of sectors per track is variable, ranging from 31 to 53. With 5 surfaces, the cylinders then vary in size from 155 to 265 sectors.

Since working directly with this unusual and inconvenient physical structure would have restricted portability and added to implementation overhead, we chose instead to focus on *virtual cylinders*. A virtual cylinder is a group of consecutive physical sectors which may cross physical cylinder boundaries. The size (in sectors) of a virtual cylinder was dictated by several considerations. It had to be small enough to capture real cylinder request dependencies, but not so small that the total number of virtual cylinders, which is the number

^{TM} AViiON is a trademark of Data General Corp.

of states in our workload model, would render the model unmanageable. Further, as discussed in section 5, implementation is facilitated by choosing a virtual cylinder size that is an integer multiple of the operating system's default transfer size. In DG/UX 4.30 this is 16 sectors (8 Kbytes). Thus we chose 160 sectors (32 sectors $\times$ 5 surfaces) as the virtual cylinder size, which gave us 2,181 virtual cylinders.

The AViiON 4000 and its 179 Mbyte disk are reserved for special projects, such as this one, and thus its natural workload is not representative of the general programming mix we wished to study. Fortunately, this system is attached to (boots from) a Data General AViiON AV 200 with a 330 Mbyte SCSI disk drive which is in general use by students and faculty, and thus does support the desired workload mix.

We chose to monitor the latter system and adapt its measured workload to the former. We built a new DG/UX kernel for the AV 200 which contained a global memory region to record disk requests. For each request we recorded the sector number, the time the request entered the WSCAN queue, and the time the request left the WSCAN queue to enter service. To extract the recorded information from kernel memory, we included a new system call that simply read the kernel variables and wrote them to user address space. An independent user process periodically exercised the system call and wrote results to a remote file via NFS in order to minimize interference with the natural load on the 330 Mbyte disk.

We monitored the system for a period of three days, which yielded a stream of 163,681 disk accesses. Since WSCAN was in effect, and not FCFS, the stream leaving the queue differed from the stream arriving to the queue. The effects of the scheduling algorithm cannot be ignored, and thus it is the departure stream that is of interest. Finally, we partitioned the sectors of the 330 Mbyte drive to yield the desired 2,181 virtual cylinders, and then re-expressed the departure stream in terms of these virtual cylinders. This stream of virtual cylinder requests was then used to produce a Markov model representation of the workload that could be transferred to the test system.

3. A MARKOV MODEL

A *first-order* Markov chain, such as described in section 1, is one in which next state probabilities depend only upon the current state. If next state probabilities depend upon the last 2, 3, or, more generally, n states visited, then the Markov chain is said to be of *order* 2, 3, or n. In this same terminology, an independent access model is an order 0 Markov chain, since next state probabilities do not depend upon any states visited.

We assume that the measured sequence of virtual cylinders visited represents a Markov chain of some order, perhaps order 0. Following Haring [3], we can test for this order. Let $vc(t) \in \{1, 2, ..., 2181\}$ denote the virtual cylinder accessed on the t^{th} request, where the total number of requests is T = 163,681. Let

$$N_j(t) = \begin{cases} 1 & \text{if } vc(t)=j \\ 0 & \text{otherwise} \end{cases}$$

$$N_{ij}(t) = \begin{cases} 1 & \text{if } vc(t)=j \text{ and } vc(t-1)=i \\ 0 & \text{otherwise} \end{cases}$$

Maximum likelihood estimators of π_j and ρ_{ij} are then

$$\hat{\pi}_j = \sum_{t=1}^{T} N_j(t)/T$$

and

$$\hat{\rho}_{ij} = \sum_{t=2}^{T} N_{ij}(t) / \sum_{t=2}^{T} N_i(t-1)$$

We first tested the hypothesis

$$H_0 : \rho_{ij} = \pi_j \quad \forall i, j \quad \text{(Markov chain is order 0)}$$

versus the alternative

$$H_1 : \rho_{ij} \neq \pi_j \quad \forall i, j \quad \text{(Markov chain has order} \geq 1\text{)}$$

From [3],

$$\chi_0^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{t=2}^{T} N_i(t-1) \, [\hat{\rho}_{ij} - \hat{\pi}_j]^2 \, / \, \hat{\pi}_j$$

has χ^2 limiting distribution with $N(N-1)$ degrees of freedom. For such large degrees of freedom, we can regard $\sqrt{2\chi_0^2} - \sqrt{2N(N-1)-1}$ as a sample from a standard normal [11]. Our measured virtual cylinder stream yielded χ_0^2 = 41.49 $\times 10^6$, which allowed us to reject H_0 at significance level less than 0.001.

We tested for higher order using a similar procedure. Let γ_{ijk} denote the conditional probability that the read/write head moves next to cylinder k, given that it is now on cylinder j and arrived at cylinder j from cylinder i. If

$$N_{ijk}(t) = \begin{cases} 1 & \text{if } vc(t)=k,\ vc(t-1)=j, \\ & \quad \text{and } vc(t-2)=i \\ 0 & \text{otherwise} \end{cases}$$

then a maximum likelihood estimator of γ_{ijk} is

$$\hat{\gamma}_{ijk} = \sum_{t=3}^{T} N_{ijk}(t) / \sum_{t=3}^{T} N_{ij}(t-1)$$

and we can test

$$H_0 : \gamma_{ijk} = \rho_{jk} \quad \forall i,j,k \quad \text{(Markov chain is order 1)}$$

versus the alternative

$$H_1 : \gamma_{ijk} \neq \rho_{jk} \quad \forall i,j,k \quad (\text{order} \geq 2)$$

by using

$$\chi_0^2 = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{t=3}^{T} N_{ij}(t-1)\,[\hat{\gamma}_{ijk} - \hat{\rho}_{jk}]^2/\hat{\rho}_{jk}$$

which has χ^2 limiting distribution with $N(N-1)^2$ degrees of freedom. In this case we were unable to reject H_0 (even at the 0.1 significance level), and so we conclude that a first order Markov chain best describes the measured workload. Nevertheless, we must qualify this conclusion with the observation that T = 163,681 may be insufficient for this second test, and additional measurement may be useful.

4. OPTIMIZATION

Simulated annealing [5] is a probabilistic algorithm that has been applied to many optimization problems in which the set of feasible solutions is so large that an exhaustive search for the optimum solution is out of the question. The optimal placement of 2,181 virtual cylinders on the disk drive yields a feasible set of 2181! $\approx 10^{6336}$ possible permutations, and so the problem would appear to be a good candidate for this approach. Although simulated annealing does not necessarily provide the optimum solution, it usually does provide a good solution in reasonable time.

Input for this problem consisted of the measured estimators, $\hat{\pi}_i$ and $\hat{\rho_{ij}}$. For any permutation, P, the *energy* of the permutation was defined to be the estimated average seek distance,

$$energy[P] = \sum_{i=1}^{N}\sum_{j=1}^{N} |P(i) - P(j)|\hat{\pi}_i\hat{\rho_{ij}}, \qquad (3)$$

The simulated annealing algorithm can then be described as a non-deterministic walk on an energy surface that is changing shape under the control of a "cooling" schedule. Specifically,

```
#define EQUILIBRIUM
        (accepts≥2000 AND rejects≥5000)
            OR (accepts+rejects > 50000)
#define FROZEN ((temperature < 0.5) OR
        ((temperature < 1.0) AND (accepts==0)))
```

```
while(not(FROZEN)){
    accepts = rejects = 0;
    old_energy = energy();
    while(not(EQUILIBRIUM)){
        cylinder1 = choose_random_cylinder();
        cylinder2 = choose_random_cylinder();
        swap_cylinders(cylinder1,cylinder2);
        new_energy = energy();
        E = new_energy - old_energy;
        if(rand(0,1) < e^min{0.0,-E/temperature}){
            accepts++;
            old_energy = new_energy;
            }
        else {
            /* put them back */
            swap_cylinders(cylinder1,cylinder2);
            rejects++;
            }
        }
    temperature = temperature*0.8;
    }
```

The energy of the identity permutation, which describes the original cylinder arrangement, was 390.647935. Upon completion, the simulated annealing algorithm had arrived at a permutation with an energy of 32.3543347, and this permutation was used in the subsequent system test.

5. INSTALLATION AND TESTING

The installation of virtual cylinder remapping in an operating system requires implementation of both *relocation* and *remapping*. First, the virtual cylinders must be physically relocated on disk to match a desired permutation, and the operating system must be made aware that the permutation is in effect. Subsequently, the operating system must remap each disk sector request so that a request falling into virtual cylinder i is translated into a request for virtual cylinder $P(i)$, where the desired information actually resides.

To accomplish remapping, we installed in the kernel address space an array to hold the translation map, and we added a system call, *set_disk_map()*, to initialize the array so that $disk_map[i] = P(i)$. We also added a system call, *disk_map()*, to enable or disable remapping. When mapping is enabled, all requests for disk access are translated via the mapping array. To verify that mapping occurred correctly, we wrote unique identifiers at the beginning of each virtual cylinder, installed and activated a fictitious map, and read from each of the virtual cylinders.

	Mean serv. time (ms)	Improvement
Unmapped	24.423321	—
Pipe organ	19.217960	21.31%
Markov	18.158880	25.65%

Table 1: Mean service times

To measure the effects of remapping, we instrumented the disk request subsystem to record, for each disk access, the requested sector number, the time the request entered the queue, the time the request left the queue to begin service, and the time at which service was completed. Using this data, we computed mean service time under each of three disk arrangements: the initial unmapped configuration, the pipe organ arrangement, and the permutation produced by the annealing algorithm discussed in the previous section. For each test, we used the same request stream of 250,000 disk accesses, which was stochastically generated from the first order Markov model workload description.

The measured mean service times for each map configuration are shown in Table 1. We see that the pipe organ configuration provided a significant improvement over the original unmapped arrangement. However, when first order dependencies are taken into account, we see that an additional improvement can be realized. These results indicate that, if the sequence of disk requests can be correctly characterized by a first order Markov model, as the tests of section 3 would suggest, then effective disk rearrangement can provide a noticeable performance improvement over both the unmapped disk and a rearrangement that assumes independent accesses.

These results were obtained from a fairly simple prototype implementation of virtual cylinder remapping. A production implementation would require that we handle many details not addressed in the prototype. At present the operating system stores the translation map only in kernel memory, so that a system crash causes loss of ability to address the disk correctly. A production system would have to store the map on disk as well as in memory, so that it could recover from a crash. Furthermore, the map must be stored in a disk area which is not remapped, since otherwise the crash recovery process would face the recursive problem of needing the disk map to find the disk map. Fortunately, there is usually an area of the disk reserved for just such information (e.g. bad block maps).

Crash recovery is not the only problem we face; even normal machine boot-up presents some challenges. Booting often proceeds in two stages. First, control is transferred to simple loader code stored in hardware PROM. This code reads a second-stage loader from a fixed location on disk, and transfers control to it. The second-stage loader contains enough file system information to find a file containing the operating system kernel, which it loads and executes. This boot process has two major implications for cylinder remapping. First, since we cannot change the boot PROM, the second stage bootstrap must remain in a fixed location, and hence cannot be remapped. Second, the second stage bootstrap must be modified to perform remapping so that the kernel file can be located. This means that the bootstrap code becomes somewhat larger and that we must implement remapping in two places: the bootstrap and the kernel.

The operating system transfer size and protocol are also a concern. Requests arrive to the SCSI controller in the form: (starting sector, number of sectors to transfer). Care must be taken that such requests do not extend across virtual cylinder boundaries, since contiguous sectors may no longer be contiguous after remapping. In response to a user request for sector n, the DG/UX transfer protocol requests from the SCSI drive 16 consecutive sectors starting at sector $16 \times \lfloor n/16 \rfloor$. Thus the choice of 160-sector virtual cylinders precludes the spanning problem. This was not an issue during the tests we have described, since driving the system from the Markov workload model allowed us to bypass the protocol and access the disk directly through the raw interface.

The final major detail to be addressed is the time of the actual rearrangement of disk data. This rearrangement can be performed either online while the machine is in normal operation, or offline. While taking the machine out of normal operation makes rearrangement easy, machine availability is decreased. Online rearrangement keeps availability high but requires more effort to ensure disk consistency at all times. In either case, two major criteria must be satisfied: the translation map must be valid at all times during normal operation and rearrangement, and if there is a machine failure during the rearrangement process, the map must be valid after recovery and the rearrangement process must continue.

6. CONCLUSIONS

We have suggested a scheme for disk subsystem performance enhancement that is based on (virtual) cylinder remapping. We measured workload on a real UNIX system and found that disk accesses could be reasonably characterized by a first order Markov model. We used simulated annealing, together with maximum likelihood estimators of the Markov model parameters, to find a permutation of the (virtual) cylinders that substantially reduced expected seek distance. We then

installed this permutation in a real system and tested it under a workload that was stochastically generated from the Markov model. We found this procedure offered a 25.6% reduction in mean service time when compared to the original (unmapped) cylinder arrangement.

Disk remapping provides an opportunity to optimize performance for multiple workloads, such as different shifts or days (e.g., many installations have a primarily interactive load during the day, but shift to batch and database processing in the evening). Normally a workload change would mean that a disk would have to be rearranged to accommodate a new disk map matching the new workload. Consider, however, a fault tolerant system maintaining multiple copies of physical disks. There is nothing to prevent each copy from having a unique translation map, tuned for a specific workload. In this case, a workload change could be met simply by a change in designation of the primary disk in a duplicated group.

Finally, we should note that the realized 25.6% improvement may be a conservative estimate of the potential for remapping. The distribution of requests across the virtual cylinders in our measured workload happened to be highly non-uniform. Should the distribution of requests become uniform, a pipe-organ arrangement would offer no improvement, and yet the Markov-based approach should still be effective. Further, the annealing algorithm strives to minimize average seek distance, but our interest is average seek time. Seek time is a nonlinear function of seek distance, and so the two averages are not equivalent with respect to optimization. A carefully measured seek time function should replace the more naive distance function, $|P(i) - P(j)|$, in future studies.

References

[1] R. Geist and S. Daniel. A continuum of disk scheduling algorithms. *ACM TOCS*, 5, 1987.

[2] R. Geist, R. Reynolds, and E. Pittard. Disk scheduling in System V. In *Proc. of the ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 55–68, Banff, Alberta, May 1987.

[3] G. Haring. On stochastic models of interactive workloads. *Proc. of PERFORMANCE '83*, pages 133–152, May, 1983.

[4] M.H. Kelley. A look at the DG/UX file system. Technical Report 3, Data General Corporation, 1990.

[5] S. Kirkpatrick. Optimization by simulated annealing: quantitative studies. *Journal of Statistical Physics*, 34(5/6):pp. 975–986, March 1984.

[6] D.E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, 1973. pp. 404-405.

[7] R. Reynolds. Sector-based disk scheduling in the UNIX System V environment. In *Proc. of the ACM Southeastern Regional Conf.*, pages 648–652, Mobile, Alabama, April 1988.

[8] R. Reynolds and M. Beede. An analytical comparison of disk scheduling algorithms. In *Proc. of the ACM Southeastern Regional Conf.*, pages 56–61, Birmingham, Alabama, April 1987.

[9] D. Suggs. Disk scheduling with future knowledge. In *Proc. of the ACM Southeastern Regional Conf.*, Greenville, South Carolina, April 1990.

[10] D. Suggs. The use of future knowledge in the design of a disk scheduling algorithm. Master's thesis, Clemson University, 1990.

[11] K. Trivedi. *Probability and Statistics with Reliability, Queueing and Computer Science Applications*. Prentice-Hall, 1982.

[12] P. Vongsathorn and S.D. Carson. A system for adaptive disk rearrangement. *Sofware Practice and Experience*, 20(3):pp. 225–242, March 1990.