Realism and Simplicity: Disk Simulation for Instructional OS Performance Evaluation

Peter DeRosa Kai Shen Christopher Stewart Jonathan Pearson
Department of Computer Science, University of Rochester
peter.david.derosa@gmail.com, {kshen, stewart, jpearson}@cs.rochester.edu

ABSTRACT

Operating system laboratory assignments based on bare hardware or detailed machine simulators can be excessively challenging for many students. In the most often used approach, students develop kernels on virtual machines with a much simplified hardware interface. Traditionally this simplification goes so far as to make realistic performance measurement impossible. We propose *Vesper*, an instructional disk drive simulator with a high degree of performance realism. Vesper retains simplicity while providing timing statistics close to that of real disk drives. The key to our approach is to provide hardware abstractions that are simple but yet capable of capturing device interactions with major performance impacts. Vesper laboratory assignments allow students to realistically explore the performance consequences of various system designs without the cumbersome aspects of the real hardware interface. This paper describes the design and implementation of the Vesper disk drive simulator. We evaluate the effectiveness of Vesper-based laboratory assignments in terms of operating system performance evaluation. Student experience and feedback are also reported.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer Science Education; D.4.7 [**Operating Systems**]: Organization and Design

General Terms

Measurement, Performance, Experimentation

Keywords

Operating systems, virtual machine, disk simulation, performance evaluation

1. INTRODUCTION

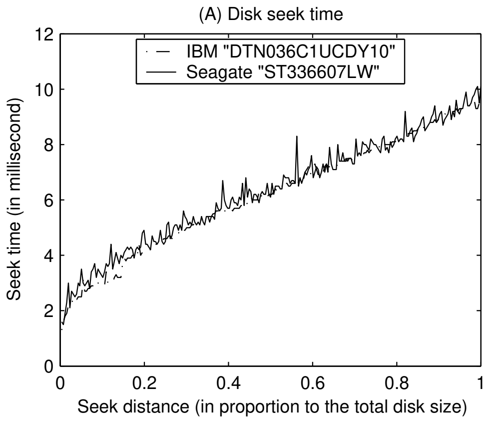
For undergraduate operating system courses, student kernel development upon real machines [11, 12] requires immense time and dedicated hardware. The more often used approach for OS laboratory assignments is based on virtual machines, such as Nachos [2]. Traditional instructional VMs oversimplify the hardware model such that they cannot provide realistic performance statistics. Therefore assignments based on such platforms can only examine students' ability to produce *correct* system designs and implementations. While the *efficiency* of various OS designs are often discussed in class, students typically do not have the opportunity to realistically explore performance issues in practice. The use of detailed machine simulation to study computer system performance has been investigated in the operating systems research community [8], however, such simulations expose the raw and uninviting interface of the real hardware. This approach is prone to create time-consuming assignments in which students fixate on details rather than concepts. As a result, existing student assignments based on detailed machine simulators (*e.g.*, VMware [13] and bochs [1]) are often limited to small modifications to an existing OS such as Linux.

In this paper, we argue for an instructional virtual environment that simulates hardware devices with a very simple abstraction and yet maintains realistic timing statistics. While this design concept can be applied to a plethora of hardware devices, our discussion in this paper focuses on the disk I/O subsystem. We choose disks since they are the performance-determining factor for many real-world applications, and appear likely to remain so in the foreseeable future. Our disk simulator, called *Vesper*, is profile-driven. The Vesper profiler automatically observes the I/O performance at a large number of sampled input parameters. This information is then fed into the simulator which mimics the performance characteristics of the profiled drive. The accurate disk-timing model in Vesper is presented to student OS programmers through a simple interface (similar to the disk drive interface in Nachos).

Detailed disk simulators with accurate timing models have been produced in the past [7, 9, 10, 14]. Some (*e.g.*, DiskSim [3]) also support automated disk characteristic analysis. However, these efforts have often focused on the minutiae of disk simulation and even emulation with the exact hardware interface [4]. The accuracy and detail afforded by such simulators is admirable, but the cost is the simplicity which is paramount to any instructional platform. Vesper seeks to fill the void between overly simplified systems like Nachos, and overly complex systems such as DiskSim, by balancing the needs for simplicity and realism. In particular, Vesper focuses on capturing the aggregate timing of a sequence of requests, rather than that of each individual request. This relaxation allows us to significantly reduce the simulation complexity without losing much overall performance realism.

Previous simulated instructional platforms were not designed for performance realism. The disk simulators for Nachos [2] and System/161 [5] employ a seek time model that is proportional to the disk seek distance. This model does not reflect the fixed initiation cost of disk seeks (such as the head positioning overhead) on real drives. Additionally, they typically employ arbitrarily chosen disk parameters and ignore the seek and transfer timing variation due to disk zoning and other disk characteristics. The recent GeekOS [6] runs on the detailed machine simulator bochs [1]. However, bochs concerns itself mostly with fast simulation speed, and it does not offer realistic timing statistics. As far as we know, no existing instructional platform is suitable for our goal of providing a high degree of performance realism through a simple simulated hardware abstraction.

The rest of this paper is organized as follows. Section 2 presents the design and implementation of the Vesper disk drive simulator. Section 3 describes Vesper-based laboratory assignments that explore the performance of various disk I/O scheduling algorithms and examine the benefit of I/O prefetching. Section 4 concludes the paper.



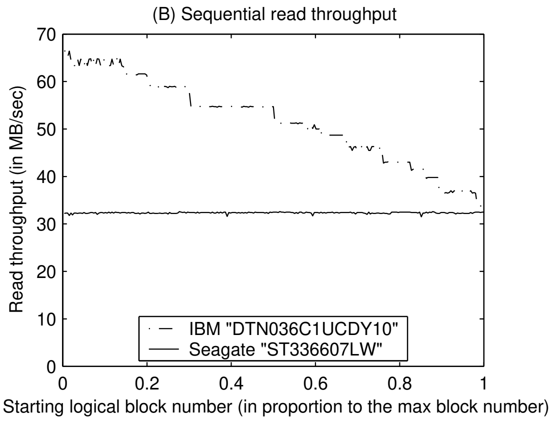


Figure 1: Sequential seek time and read throughput profiling results for two SCSI drives.

2. VESPER DISK SIMULATOR

Unlike detailed disk simulators such as DiskSim, Vesper does not differentiate the effects which the controller, bus, and drive have upon the aggregate timing of the whole subsystem. Instead, Vesper views the controller, bus, and drive as a black box which accepts a simple set of commands. The Vesper profiler accumulates timing statistics for these commands with varying input parameters and records them into a profile. All the characteristics and quirks of the profiled drive can then be incorporated into the Vesper disk simulator. Ideally, instructors will profile a variety of drives and distribute the profiles with student assignments. This variety is a key advantage of Vesper over previous instructional platforms, because it allows students to examine seek and transfer timing characteristics of different disks as well as experience the effects that drives and controllers of varying quality can have on real-world performance.

The rest of this section describes the Vesper disk drive simulation model in detail. We also illustrate the realism of its timing statistics and the simplicity of its device interface.

2.1 Vesper Disk Drive Simulation Model

The main objective of the disk profiling is to acquire the functional mapping between parameters of an I/O command with its estimated runtime. The total time of a disk operation mainly includes the seek time, rotational delay, and data transfer time:

$$T_{total} = T_{seek} + T_{rotation} + T_{transfer} \qquad (1)$$

Many factors affect the individual component times, but in each case it is not hard to establish what the major factors are. Seek time depends almost entirely on the number of cylinders (we approximate it with the number of blocks) being traversed. Data transfer time depends on the starting cylinder[1] (again we approximate it using the starting block number) and the number of blocks which are to be read. Rotational delay depends on the disk head location after seeking and the location of desired data on the track, which requires accurate disk geometry information. It may be further complicated by out-of-order transfer intended for reducing the rotational delay on modern disks. Since our objective is to capture the aggregate timing of a sequence of requests, rather than that of each individual request, our model simply uses the average rotational delay between two random track locations (*i.e.*, the time it takes the disk to spin half a revolution).

We take a large number of measurement samples for seek and transfer times at different input parameters (*i.e.*, the seek distance for seek time modeling and the starting location for transfer time modeling). We conduct the measurements by issuing direct SCSI commands through the Linux generic SCSI interface, which allows us to bypass the OS memory cache and selectively disable the disk controller cache. The number of measurement samples can be adjusted for the Vesper profiler. Results in this paper represent 256 samples evenly distributed across each profiled disk. Each

[1] The data transfer speed on modern disks is not constant due to *disk zoning*, under which tracks on different cylinders may contain different amount of data.

Disks	random seek	random transfer
IBM	10.1%	1.1%
Seagate	8.9%	2.6%

Table 1: Validation error for the Vesper disk simulation model (on two SCSI drives using two synthetic workloads).

sampled seek time is determined as the average time of seeking from the very beginning of the disk to a particular sampling location. The transfer time is measured by reading 16 MB at each sampling location, as a large amount of data is needed to determine the maximum disk throughput.

Figure 1 illustrates the seek time and sequential read throughput for two 36 GB SCSI drives that we profiled during the development of Vesper. The two drives have similar seek time curve. We note that the seek time is not proportional to the seek distance. Even very short distance seeks must induce significant seek initiation cost (around 2 ms). As for the sequential read throughput, the IBM drive supports higher throughput for data at outer disk cylinders (those with small block numbers) due to zoning on modern disk drives. The Seagate drive also supports zoning, however, we observe a flat curve for the Seagate drive sequential read throughput in Figure 1(B). We attribute this to speed constraint of the disk controller, although other causes are also possible. Despite this oddity and the difference between the two drives, Vesper's sampling-based model was able to capture their characteristics quite well as demonstrated in the next section.

2.2 Realism of Timing Statistics

The accuracy of the Vesper disk simulation model has been validated using two synthetic workloads: *random seek* and *random transfer*. Random seek consists of 1000 seeks both starting and ending at random blocks. Random transfer consists of 1000 reads starting at random blocks, with random read sizes between 512 bytes (1 block) and 16 megabytes. Before each transfer is timed the disk head is positioned at the starting block to eliminate any seeking time, which might cause error. For each workload, we compare the predicted runtime of the Vesper model with the measured result on the real system. The difference, divided by the measured result, is considered as the validation error. The complete validation results are shown in Table 1. Results demonstrate the performance realism of the Vesper disk simulator. The validation error is around 10% for the random seek workload while the error is below 3% for the random transfer workload. Such level of accuracy makes it a realistic platform for students to explore performance issues of OS design and implementation.

2.3 Simplicity of Device Interface

Student OSes interact with Vesper's simulated drive by means of *only* three functions (shown in Figure 2). The initialization routine takes the disk profile and an interrupt handler function as parameters. The interrupt handler is invoked upon the completion of each I/O request. For the I/O read and write functions, the only addition to the Nachos disk interface is an extension to support multi-block operations — the second parameter of each function specifies the number of blocks in the I/O operation.

```
/* Disk initialization with a specified profile
   and an interrupt handler function */
void Init(char *profile, FuncPtr interrupt_hdr);

/* Multi-block disk read request */
void Read(int start_block, int len, char *data);

/* Multi-block disk write request */
void Write(int start_block, int len, char *data);
```

Figure 2: Vesper disk driver simulator APIs.

Student assignments often use very small simulated disks for simplicity. A small simulated disk also makes it easy to explore issues such as the disk geometry and space allocation. For example, the simulated disk drive in Nachos is 128 KB. We scale the disk properties in the Vesper profile such that timing information obtained for large real drives still applies to small simulated drives. Specifically, we scale the seek distance for the seek time model and the starting location for data transfer time model according to the ratio between the real disk size and the simulated disk size. We also scale the disk rotation speed such that it takes the same amount of time to rotate one revolution for both the real disk and the simulated disk. Finally, we scale the sequential read throughput so that it takes similar amount of time for sequentially reading one track of data on both the real disk and the simulated disk. Particularly for a simulated disk with 4 KB track size and a typical real disk track size of 400 KB, we use 1/100 as the scaling ratio for the sequential read throughput. These scaling parameters can be adjusted by the instructor if necessary but students do not need to be aware of them.

3. ASSIGNMENTS AND EVALUATION

3.1 Vesper-based OS Assignments

We augment the Nachos virtual machine (version 3.4) by replacing its original disk simulator with Vesper. Note that the Vesper disk simulator can also be used with other instructional platforms. We choose Nachos because many instructors are already comfortable with it, and both preexisting public and home-brew Nachos assignments may easily take advantage of Vesper's features in this context.

We enhance the Nachos file system assignments with the emphasis on realistic performance evaluation. In the first assignment, we ask students to implement several disk scheduling algorithms (First-Come-First-Serve, Shortest-Seek-Time-First, and Cyclic-Look) into the operating system and compare their performance on simple user programs. Students are required to experiment with multiple disk drives (or multiple Vesper disk profiles, to be more precise) and explain the disk characteristics that affect the performance. Students are aware of the key characteristics of the experimented drives but they do not need to understand how the profiles are acquired or how they are incorporated into the instructional virtual machine.

In the second assignment, we ask students to implement I/O caching and prefetching in the file system and measure their performance impact. As for prefetching, we also ask

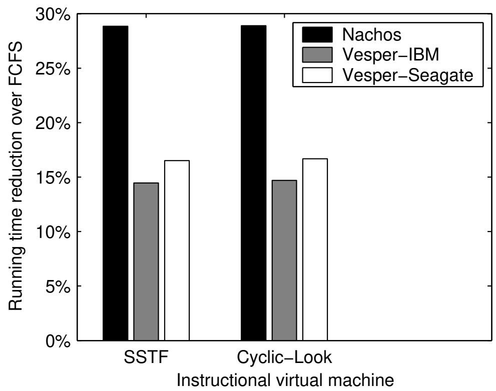


Figure 3: Performance enhancement of seek-reduction disk I/O schedulers under different instructional virtual machines. "Vesper-IBM" represents the Vesper disk simulator running on the IBM drive profile on Section 2.1. "Vesper-Seagate" represents the Vesper disk simulator running on the Seagate drive profile.

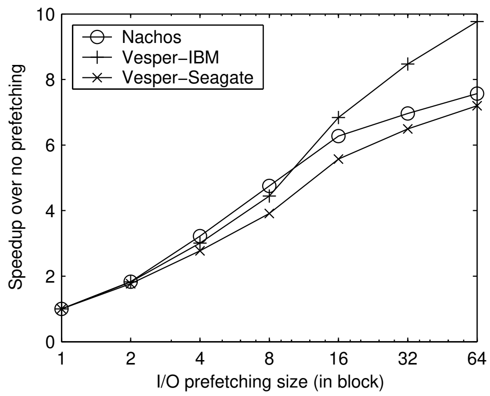


Figure 4: Benchmark speedup due to I/O prefetching. The base performance is that of "no prefetching" or a prefetching size of 1 block.

students to quantify the performance when using different prefetching sizes. Similar to the first assignment, students are required to experiment with multiple disk drives and identify the disk characteristics that affect the performance of disk scheduling algorithms.

For both assignments, we provide a *concurrent read* user program for performance evaluation. The program initializes the empty disk with 10 files of 10 KB each (so the 10 files would occupy most of the space on the 128 KB simulated drive). It then launches 10 concurrent threads to read these 10 files. Each thread reads 10 bytes at a time until it finishes reading the file it is assigned to. The program reports the elapsed time for all concurrent read threads to complete. In addition to the given program, we also encourage students to design more test programs to identify the impact of disk drive characteristics on file system performance.

3.2 Evaluation Results

Using a typical student implementation, we show the performance evaluation results of the I/O prefetching and scheduling assignments. We show the results when using the original Nachos disk simulator and Vesper with two disk profiles (for the IBM and Seagate drives described in Section 2.1). Our purpose is to demonstrate Vesper's ability in supporting realistic performance evaluation and uncovering the performance impact of varying drive characteristics found on real disks.

Seek reduction I/O scheduling algorithms such as SSTF and C-Look (or the elevator scheduler) outperform FCFS by reordering disk I/O requests to reduce the disk seek overhead. Figure 3 shows the performance improvement of seek reduction schedulers under different instructional virtual machines. We observe that the original Nachos disk simulator exhibits much higher performance improvement. This is because Nachos's proportional seek time model does not consider the seek initiation overhead such as the head positioning cost. Such fixed seek cost makes seek-reduction schedulers less effective in practice. The two disks used in the Vesper simulator exhibit similar performance characteristics in this test because they have very similar seek time curve. The sequential read throughput does not have much impact on the performance differentiation of I/O scheduling algorithms.

I/O prefetching can improve the performance of concurrent reads by amortizing the cost of disk seeks over large granularity read operations. In other words, the disk transfers more data between consecutive seeks due to I/O prefetching. More aggressive prefetching (*i.e.*, larger prefetching size) tends to improve the performance further. Figure 4 shows the benchmark speedup due to I/O prefetching. We observe that the Vesper simulator with the IBM drive profile exhibits higher performance improvement at large prefetching sizes. This is because it has much higher sequential read throughput than the Seagate drive and the original Nachos disk simulator.

In summary, the original Nachos disk drive simulator is oversimplified such that its performance results can differ significantly from those of realistic disk drives. Further, different real disk drives also have varying characteristics that may affect system performance. Under such context, Vesper-based laboratory assignments allow students to realistically explore operating system performance issues at no cost of abstraction complexity.

3.3 Experience and Student Feedback

We describe our experience with 9 student groups (each comprised of two or three students) working on the Vesper-based assignment in an operating systems course. Figure 5 illustrates the grade distributions of the Vesper-based assignment (on file system and I/O) and a traditional Nachos assignment (on process management). Despite the enhanced performance realism, the Vesper-based assignment is not substantially more challenging and we attribute this to the Vesper disk drive's simple interface. Informal interviews with students revealed their interests in performance-based assignments. Students were especially surprised by the performance impact of prefetching and disk scheduling algorithms, and were motivated to evaluate the performance of other disk-bound operating system components (*i.e.*, virtual memory). Some students recommended a class-wide performance contest, which would not be meaningful without a realistic performance measurement platform.

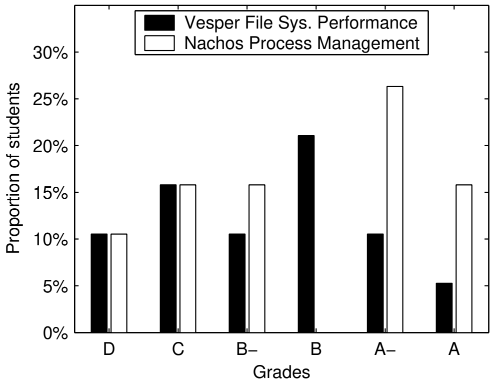


Figure 5: Grade distribution for the Vesper-based assignment and a traditional Nachos assignment.

4. CONCLUSION

In this paper, we propose Vesper, an instructional disk simulator with a high degree of performance realism. Vesper balances the need for simplicity in an instructional platform with the objective of offering students the opportunity of realistic performance measurements. The Vesper disk simulation model is based on observing the I/O performance at a large number of sampled input parameters. Validated with two synthetic workloads, we show that the Vesper model has a worst-case error of around 10%.

We also describe Vesper-based file system assignments and compare the student performance evaluation results of Vesper with those of the original Nachos disk simulator. Our study shows that Nachos can provide quantitatively inaccurate performance results on seek-reduction disk scheduling algorithms. In comparison, Vesper lets students realistically explore operating system performance issues with minimal changes to the convenient and popular Nachos interface. With Vesper it is possible to introduce performance-centric assignments into the OS classroom and to provide students with an outlook about how different hardware characteristics can affect the performance. Tuning an OS to meet the needs of different hardware devices is very important in practice, but it is often ignored in operating systems instructions.

The philosophy of binding realistic timing to simplified interfaces can be extended to other devices in the future. For example, combining a realistic system memory simulator with the existing disk work will result in a comprehensive testbed for students to experiment with virtual memory and cache-replacement policies. We expect that similar observation-based modeling techniques can be devised for other hardware devices in our future work.

Acknowledgment

This work was supported in part by an NSF CAREER Award CCF-0448413.

Resources on the Web

The Vesper disk profiler and Vesper-enhanced Nachos simulator can be acquired by visiting `www.cs.rochester.edu/u/kshen/research/vesper`. The performance-oriented file system assignment described in this paper (as used at the University of Rochester) can also be found there.

5. REFERENCES

[1] bochs: The Open Source IA-32 Emulation Project. http://bochs.sourceforge.net.

[2] W. A. Christopher, S. J. Procter, and T. E. Anderson. The Nachos Instructional Operating System. In *Proc. of the USENIX Winter Conference*, San Diego, CA, January 1993.

[3] The DiskSim Simulation Environment. http://www.pdl.cmu.edu/DiskSim.

[4] J. L. Griffin, J. Schindler, S. W. Schlosser, J. S. Bucy, and G. R. Ganger. Timing-accurate Storage Emulation. In *Proc. of the First USENIX Conference on File and Storage Technologies*, Monterey, CA, January 2002.

[5] D. A. Holland, A. T. Lim, and M. I. Seltzer. A New Instructional Operating System. In *Proc. of the 33rd ACM Symposium on Computer Science Education*, pages 111–115, Cincinnati, KY, February 2002.

[6] D. Hovemeyer, J. K. Hollingsworth, and B. Bhattacharjee. Running on the Bare Metal with GeekOS. In *Proc. of the 35th ACM Symposium on Computer Science Education*, pages 315–319, Norfolk, VA, March 2004.

[7] D. Kotz, S. B. Toh, and S. Radhakrishnan. A Detailed Simulation Model of the HP 97560 Disk Drive. Technical Report PCS-TR94-220, Department of Computer Science, Dartmouth College, July 1994.

[8] M. Rosenblum, E. Bugnion, S. Devine, and S. A. Herrod. Using the SimOS Machine Simulator to Study Complex Computer Systems. *ACM Transactions on Modeling and Computer Simulation*, 7(1):78–103, January 1997.

[9] C. Ruemmler and J. Wilkes. An Introduction to Disk Drive Modeling. *IEEE Computer*, 27(3):17–28, March 1994.

[10] E. Shriver, A. Merchant, and J. Wilkes. An Analytical Behavior Model for Disk Drives with Readahead Caches and Request Reordering. In *Proc. of the ACM SIGMETRICS*, pages 182–192, Madison, WI, June 1998.

[11] A. S. Tanenbaum and A. S. Woodhull. *Operating Systems: Design and Implementation*. Prentice Hall, second edition, 1997.

[12] C. Vaill and J. Nieh. Experiences Teaching Operating Systems Using Virtual Platforms and Linux. In *Proc. of the 36th ACM Symposium on Computer Science Education*, pages 520–524, St. Louis, MO, February 2005.

[13] VMware. http://www.vmware.com.

[14] B. L. Worthington, G. R. Ganger, and Y. N. Patt. Scheduling Algorithms for Modern Disk Drives. In *Proc. of the ACM SIGMETRICS*, pages 241–251, Santa Clara, CA, May 1994.