Reducing Disk Power Consumption in Servers with DRPM

Using a dynamic-rotations-per-minute approach to speed control in server disk arrays can provide significant savings in I/O system power consumption without lessening performance.

Sudhanva Gurumurthi
Anand Sivasubramaniam
Mahmut Kandemir
Pennsylvania State University

Hubertus Franke
T.J. Watson Research Center

The growth of business enterprises and the Internet's emergence as a data processing medium have caused server-centric applications to proliferate. Data-centric services, such as file servers, Web portals, and transaction-processing systems have become commonplace in large corporations, smaller businesses, and academic institutions. Further, search engines such as Google and e-mail services like Hotmail sustain the Internet needs of millions each day.

Until now, performance has been the primary objective when designing hardware and software for these servers. Processor and memory designers constantly aim for higher clock frequencies and try to pack as many transistors as possible onto a chip. Disk drive designers attempt to manufacture faster hard disks with denser storage capacity, while software architects tune their applications for peak performance on a given hardware platform.

Recently, however, power dissipation has become a growing concern. Although traditionally considered a problem for mobile, battery-operated systems, power also poses challenges in terms of electricity costs and overall system design and reliability. Power can put a limit on how much designers can push performance, as power dissipation generates heat that affects component stability and reliability, especially for large server systems.

The input/output subsystem expends a significant percentage of this power. Servers use *disk arrays*, a large collection of disks that sustain the data requirements of these systems and, in many cases, exert a significant impact on overall server performance. Therefore, developers configure disk arrays to deliver extremely high performance. Although effective techniques exist for tackling disk power for laptops and workstations, applying them in a server environment presents a considerable challenge, especially under stringent performance requirements.

Using *dynamic rotations per minute (DRPM)* speed control for power management in server disk arrays can provide large savings in power consumption with very little perturbation in delivered performance. The DRPM technique dynamically modulates the hard-disk rotation speed so that the disk can service requests at different RPMs. Better still, existing technology can overcome the fundamental engineering challenges that fabricating such a device presents.

PERFORMANCE AND POWER

Typically, organizations house server machines in data centers, which are designed to supply high-quality electric power to those servers via multiple feed lines and extensive backup power supplies.[1] Further, to ensure the thermal stability of the servers, data centers employ powerful heating, ventilation, and air-conditioning systems. The power distribution in a typical server system allocates fully 40 percent of the overall power to the cooling system.

Nevertheless, among computer components, optimizing the I/O subsystem becomes a priority because it constitutes 26 percent of the overall power budget—primarily because servers use disk arrays to sustain the data storage and access

0018-9162/03/$17.00 © 2003 IEEE Published by the IEEE Computer Society

RAID Overview

The *redundant array of independent disks* approach[1] uses a bunch of disks to serve a request in parallel while providing the requestor a view of a single device. If there are n disks, with each disk having a capacity of B blocks, we can visualize the RAID address space as a linear address space from 0 to $nB - 1$. The unit of data distribution across the disks is a *stripe*. This stripe consists of user data arranged in a set of consecutive blocks.

Having multiple disks can decrease array reliability. Consequently, RAID configurations use either parity or mirroring for error detection and recovery. RAID configurations vary based on how they stripe the data and how they maintain redundancy. Currently used configurations include RAID-4, RAID-5, and RAID-10.

In a RAID-4 array of n disks, $n - 1$ disks store data, and the remaining disk stores parity. Logical stripe i thus falls on disk i mod $(n - 1)$, which assigns the data stripes cyclically to the disks storing data. The parity disk stores the parity information for stripes 1 through $n - 1$, n through $2(n - 1)$, and so on.

When a read for a logical block is presented, the array controller identifies the involved disk and blocks and issues requests to them. At the same time, the controller also will need to obtain the parity blocks for that data to confirm the data's validity. In a write operation, apart from involving the data disks, the controller may need to both read and write the parity to recalculate it. Thus, writes—especially small ones—usually become a problem.

RAID-5 generally works like RAID-4, except that it does not dedicate one disk to checking parity. Instead, each disk takes turns holding the parity for a given set of stripes. In this *rotating parity* approach, disk n serves as parity for data stripes 0 through $n - 1$, disk 1 serves as parity for data stripes n through $2(n - 1)$, disk 2 serves as parity for data stripes $2n - 1$ through $3(n - 1)$, and so on. Consequently, this approach can avoid the bottleneck of a single parity disk, especially for high levels of small-write traffic, evening out the load across the array.

RAID-10, which has recently become popular, employs a combination of striping and *data mirroring*, which duplicates data on two different disks. This method splits the disk array into two mirrors, each with an equal number of disks. Each mirror simply stripes the data across the disks cyclically, without any parity. A write must update both mirrors. The array controller sends read requests to the mirror that has the shortest request queue at that instant. Selecting the mirror with the shortest seek time can break ties. RAID-10 provides greater reliability because it can tolerate multiple disk failures.

Reference

1. D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *Proc. ACM SIGMOD Conf. Management of Data*, ACM Press, 1988, pp. 109-116.

requirements for the applications they run. Data-intensive applications, such as those used for transaction processing and decision support, require the I/O subsystem to be performance efficient. Further, in addition to many such server machines, a data center also includes several storage components, such as Network-Attached Storage boxes, that further contribute to the overall I/O power consumption.

One way to optimize performance is to use hard disks that have a faster RPM. Higher speeds reduce access times for disk reads and writes. However, a faster disk also consumes more power. For example, tracking the power consumption of different generations of IBM hard disks for various RPMs reveals that the spindle motor rotating the disk platters uses more than 81 percent of this power.[2]

Another performance-enhancing technique, typically used in conjunction with faster disks, organizes the data over the constituent drives in different *redundant arrays of independent disks (RAID)* configurations, an approach that the "RAID Overview" sidebar describes in detail. Although a powerful tool for improving I/O subsystem performance, RAID uses many potentially high-speed disks and thus increases power consumption. Further, the standard configuration parameters for a RAID array—the number of disks, stripe size, and RAID level—demonstrate quite different behavior in terms of power and performance across various server workloads.[3]

DISK POWER MANAGEMENT

Managing power in the context of single-disk systems such as laptops and desktops has been a topic of extensive research.[4,5] Although traditional approaches have worked well, implementing them still presents challenges that require further research.

Traditional power management

In most single-disk applications, *traditional power management* involves two steps. TPM first detects suitable idle periods, then spins down the disk to a low-power standby mode when the power management algorithm predicts that doing so will save energy. However, spinning the disk back up from this standby mode when the system receives an I/O request incurs additional latency and power costs.

Predicting how long the next idle period will last usually involves tracking some kind of history of the previous idle periods. If this period is likely to be long enough to outweigh the spindown and spinup costs, the system saves energy by turning off the spindle motor driving the platters, which spins down the disk to standby mode. TPM offers an effective energy-saving technique, especially in laptops, primarily because of the long idle periods and relatively low latencies for disk spindown and spinup.

The "ABCs of Disk Spindle Motors" sidebar provides a more detailed explanation of these issues.

TPM challenges

Although TPM potentially can provide significant energy savings for laptops and desktops, several factors make it difficult to apply this technique to servers. First, server workloads differ significantly from those on smaller platforms. In addition to accessing a multiplicity of disks for each request because of striping, these workloads also are intrinsically more I/O intensive. Server workloads typically create a continuous request stream that must be serviced, instead of the relatively intermittent activity that is characteristic of the more interactive desktop and laptop environments. This makes it difficult to obtain idle periods sufficiently long enough to profitably employ TPM.

Figure 1 plots the breakdown of the total I/O subsystem energy for three RAID configurations against two common transaction-processing benchmarks:

- TPC-C, an online transaction-processing benchmark, simulates a set of users who perform transactions such as placing an order and checking its status.
- TPC-H, an online application-processing benchmark, captures decision-support transactions on a database.

Figure 1a shows the breakdown for the energy consumed when the disks become *active* to perform reads and writes, when they remain *idle* but still spinning, and when they change their *position* during head movements. Clearly, the I/O subsystem expends the most energy when the disk is idle. This, then, should be the main mode of operation optimization, possibly with TPM.

To show how much energy power-mode transitions can save without degrading performance, an oracle idle-predictor is used to calculate each idle period. Applying TPM for a long-enough disk idle

ABCs of Disk Spindle Motors

Disk-spindle motors are permanent-magnet DC brushless motors. To operate as a brushless motor, Hall-Effect or back-EMF sensors inside the motor are required to provide the pulses necessary for proper commutation (rotation).

To make the disks start spinning, a high-acceleration torque is required to overcome the forces caused by the heads sticking to the disk platter's surface. Technologies such as disk drive load/unload[1] can ameliorate this problem by lifting the disk arm from the platter's surface. These technologies, which also provide power benefits, have been used in IBM hard-disk drives, for example, to implement the IDLE-3 mode. In addition to providing the starting torque, the spindle motor also must sustain its RPM once it reaches the intended speed.

One traditional approach used over the years to improve disk performance is to simply increase the RPM, which reduces rotational latencies and transfer times and can prove beneficial in bandwidth-bound applications. However, such increases can cause additional concerns such as noise and *nonrepeatable run-outs*—off-track errors that occur at higher RPMs, especially at high track densities. High-RPM disk designers have used advanced motor-bearing technologies such as fluid and air bearings to address these design considerations.

Reference

1. IBM Hard Disk Drive Load/Unload Technology; www.storage.ibm.com/hdd/library/whitepap/load/load.htm.

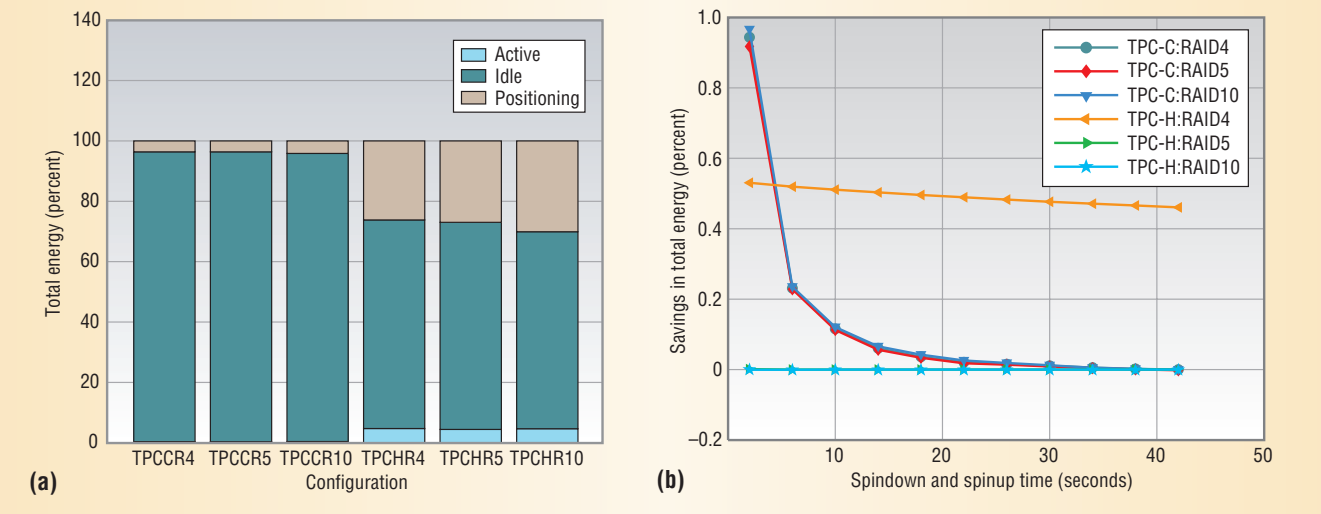

Figure 1. Energy behavior and optimization for transaction-processing workloads. (a) The breakdown of total energy consumption differs across disk modes. R-4, R-5, and R-10 refer to RAID-4, RAID-5, and RAID-10, respectively. (b) The percentage savings in total energy consumption for different spindown and spinup values.

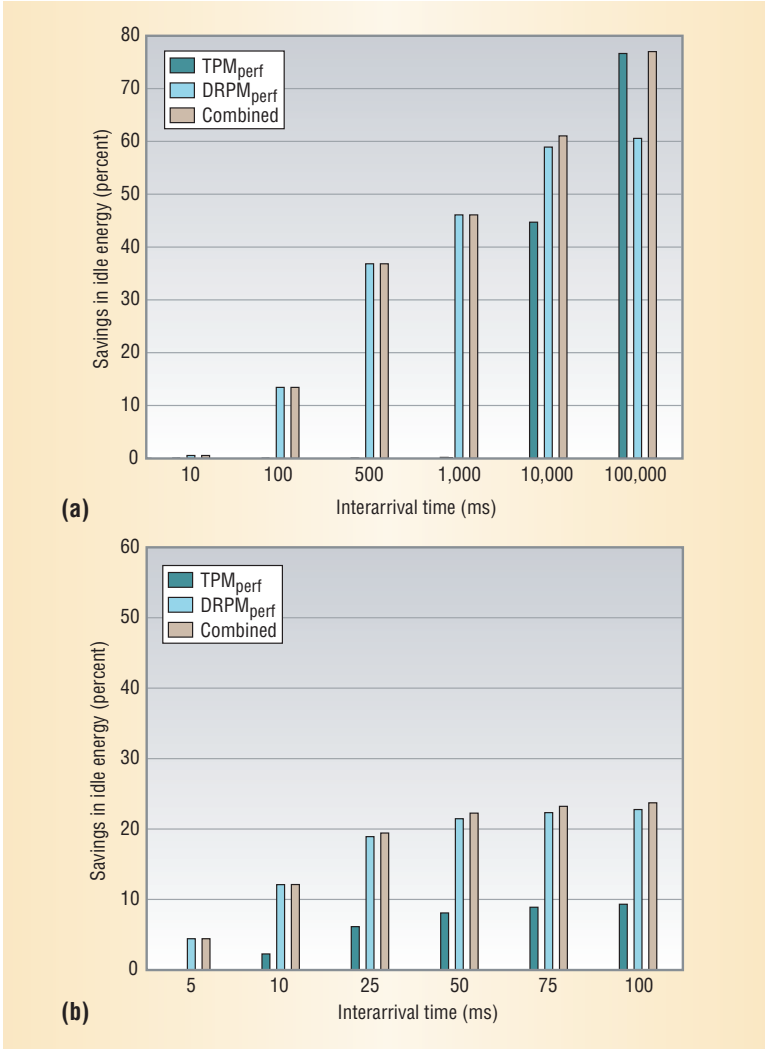

Figure 2. Savings in idle energy using TPM_{perf}, $DRPM_{perf}$, and combined savings for (a) the regular, exponential method and (b) bursty, Pareto method traffic. For each type, the mean interarrival time varies on the x-axis.

period spins the disk down and back up again just in time to service the next request, thereby saving energy while achieving the same performance generated without applying TPM.

Figure 1b plots the energy savings for different values of spindown and spinup latencies. A typical laptop disk has a spindown and spinup latency of 4 to 5 seconds, whereas a server disk has a combined latency of around 30 to 40 seconds. Thus, a combined latency of 2 seconds produces less than a 1 percent improvement in energy consumption. Similar trends have also been observed for network servers.[6]

TPM offers a less attractive option for another reason: The predictability of the idle periods tends to be low for server workloads. Detailed statistical analysis of transaction-processing workload predictability[3] has shown very little correlation between a given idle period's duration and the duration of previous idle periods. This variability makes it difficult to develop effective prediction mechanisms.

Further, server disks differ physically from their laptop and desktop counterparts. In general, manufacturers do not design them to use these low-power modes, and, even if they have this capability, the heavy platter assemblies incur lengthy spindown and spinup times, making this option less attractive for periods of active use. Finally, performance remains the primary concern in servers, and efforts to save power must avoid degrading performance.

All of these issues warrant rethinking how to tackle the disk power problem in servers. The spindle motor consumes 81.34 percent of disk power. The higher a server disk's RPM, the greater its power. However, for a given disk, its power consumption P relates to the angular velocity ω as

$$P = \frac{K_e^2 \omega^2}{R}$$

where K_e is a motor voltage constant and R is the motor resistance. This equation, similar to the equation relating power and voltage for CMOS circuits, indicates that a change in disk rotation speed has a quadratic effect on power consumption. This is the core idea behind the *dynamic RPM* technique.

RETHINKING DISK DRIVE DESIGN

The DRPM approach offers an alternative to current technology, which allows only two options for operating the disk: full-speed and standby. In DRPM, the disk can operate between these two extremes and move closer to either based on whether performance or energy takes precedence at any given time. Dynamically modulating the disk's RPM can reduce the energy consumption the spindle motor causes. Using DRPM exploits much shorter idle periods than TPM can handle and also permits servicing requests at a lower speed, allowing for greater flexibility in choosing operating points for a desired performance or energy level.

We have developed analytical models for both the performance and power behavior of a DRPM-capable hard disk and studied its sensitivity to a variety of physical parameters and storage organizations.[2] To analyze DRPM's potential under a variety of I/O conditions, we use a synthetic workload generator that lets us control parameters such as the request arrival pattern, the mix of reads and writes, and the degree of sequentiality in disk block accesses.

To determine the nature of the I/O requests' arrival pattern at the storage subsystem, we use two traffic generators. A generator based on exponential probability distribution produces more or less regular request interarrival times. The other generator, based on a Pareto distribution, produces requests in bursts. We derived our results for this

study from a 12-disk RAID-5 array with a 16-Kbyte stripe size. The disks have a maximum rotation speed of 12,000 RPM and can operate as low as 3,600 RPM when using DRPM.

To compare the potential of both DRPM and TPM across the workloads, we assume that we can match the performance of a non-power-managed storage system exactly. To achieve this, we use an idle-time prediction oracle that can exactly predict when the next request will arrive after serving each request. Based on this prediction, our idealized DRPM scheme, which we refer to as $DRPM_{perf}$, figures out how low an RPM it can go down to and yet still come back up to full speed just in time to service the next request.

We apply a similar policy for the corresponding TPM scheme (called TPM_{perf}), in which we spin the disk down and spin it back up completely, provided the duration of the idle interval can accommodate a spindown and spinup. We also consider a combination of these two schemes in which we choose either $DRPM_{perf}$ or TPM_{perf} for every idle interval based on which scheme saves the most energy. Figure 2 shows the savings during these idle periods for the two arrival patterns across a spectrum of interarrival times.

If the interarrival times are small—as shown in the leftmost bars in each of Figure 2's graphs—none of the power management schemes work effectively. On the other hand, if disk idleness times stretch into hundreds of seconds—as in the rightmost bar for Figure 2's exponential traffic section—TPM_{perf} does a better job of saving idle energy. However, between these two extremes, $DRPM_{perf}$ saves much more energy than TPM_{perf} for a range of operating conditions. As expected, the combined scheme approaches the better of the two methods across the spectrum of workloads.

PRACTICAL DRPM CONTROL POLICY

Although $DRPM_{perf}$ can provide considerable energy savings, applying it in practice is difficult, especially since perfect prediction of the idle periods might not always be possible. To address this, we have developed a more practical control policy that can exploit the DRPM capability to save energy without significantly degrading performance. This control policy uses a two-level feedback-driven algorithm that involves both the disk-array controller and the constituent disks. The algorithm is partitioned such that the disks that form the lower level try to maximize the energy savings as much as possible. On the other hand, the array controller seeks to optimize the storage system's performance by measuring the I/O requests response time and limiting the degree of freedom with which disks can optimize energy to maintain performance.

We have developed a control policy that exploits DRPM to save energy without significantly degrading performance.

In our control policy, the array controller specifies a low watermark for each disk, which indicates how low an RPM the disk can slow to. Periodically, each disk inspects its request queue to see if any requests are pending. If not, the disk lowers its RPM. The disk can keep lowering its RPM, but it cannot go lower than the low watermark that the array controller specifies for it.

At the higher level, the array controller tracks response times for I/O requests it handles during certain request periods. At the end of each period, the controller calculates the percentage change in the response time over the past two periods, as follows:

- If the percentage change (ΔT_{perf}) grows *larger* than the upper-tolerance level, the controller immediately issues a command to all disks operating at lower RPMs to ramp up to full speed. To do this, the controller sets the low watermark at each disk to full RPM, which instructs the disks not to operate below this value.
- If the percentage change lies *between* an upper- and lower-tolerance level, the controller keeps the low watermark where it is, since the response time lies within tolerance levels.
- If the percentage change is *less* than the lower tolerance level, the controller can lower the low watermark even further. The controller calculates the specific RPM used for the low watermark proportionally, based on how much the response time change is lower than the lower tolerance level.

Figure 3 shows these scenarios for an upper tolerance of 15 percent and a lower tolerance of 5 percent. The dotted line shows the low watermark's position before the application of heuristics, while the solid line shows the result of applying these heuristics. The controller calculates diff, the *percentage difference* in response times t1 and t2 between successive *n*-request windows. Figure 3a shows that if the difference is greater than the upper tolerance (UT), the controller sets the low watermark (LOW_WM) to the maximum RPM for the next *n* requests. Figure 3b shows that if the difference lies between the upper- and lower-tolerance limits, the controller retains the current low-watermark value. Figure 3c shows that if the difference is greater

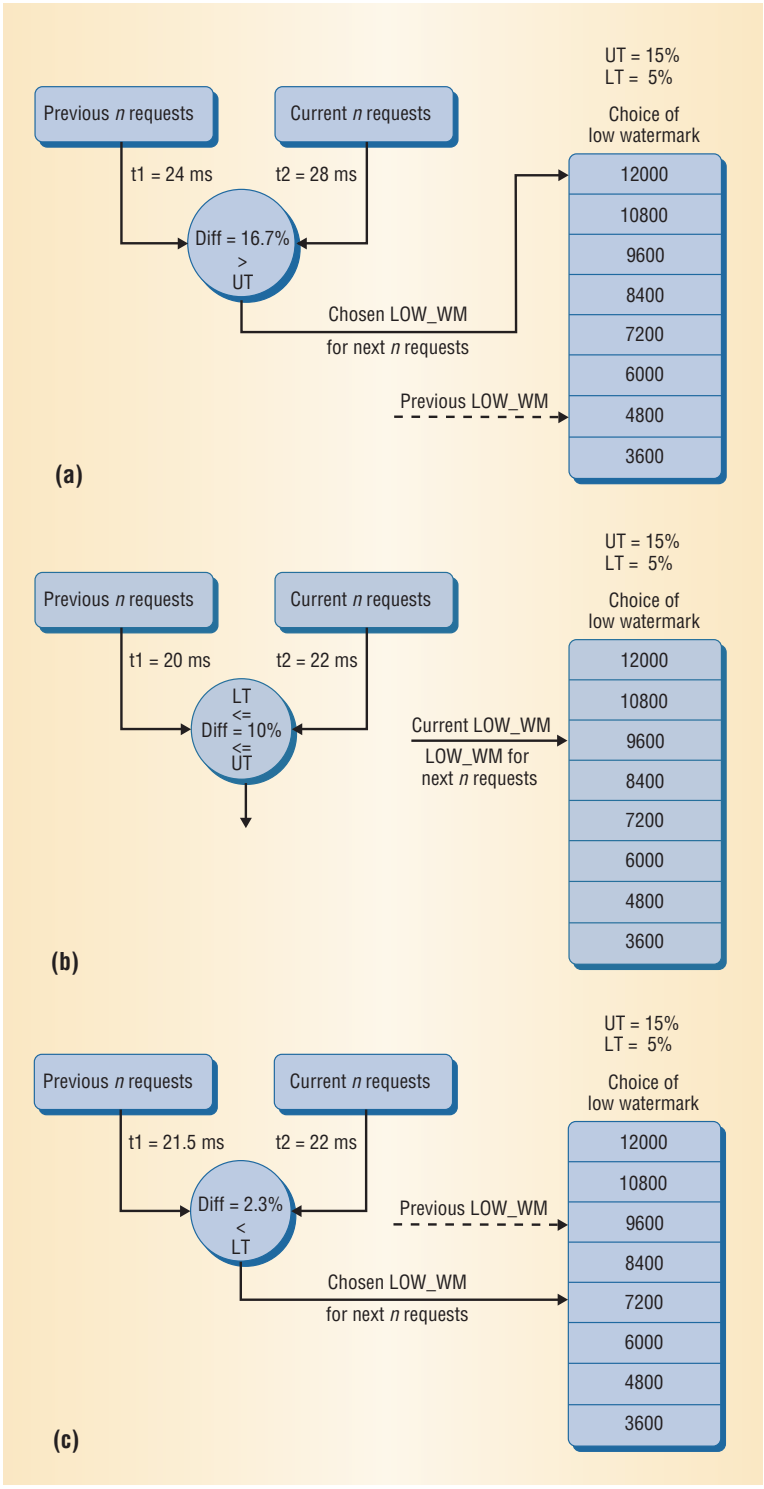

Figure 3. DRPM control-policy operation for an upper tolerance of 15 percent and a lower tolerance of 5 percent. The controller calculates the percentage change over the last two periods and adjusts the RPM's tolerances as appropriate for (a) an RPM that exceeds the upper tolerance, (b) an RPM that falls between the upper and lower tolerances, and (c) an RPM that lies below the lower tolerance.

than the low-watermark value, the controller sets the value to less than the maximum RPM.

Given that the difference value is higher than 50 percent but less than 75 percent of the low tolerance in this example, the controller sets it two levels lower than the previous low watermark. Had the tolerance been between 75 percent and 87.5 percent, the controller would have set the value three levels lower, and so on.

Figure 4 shows the effectiveness of this control policy. Figure 4a shows the savings in the total energy consumption of the storage system that we modeled, while Figures 4b and 4c show the performance results for the exponential and Pareto methods, respectively.

To quantify the performance of our policy against a baseline scheme in which DRPM induces no performance degradation, we plot the *cumulative density function* of the response times. Although applying our DRPM policy might inordinately delay a few requests, most requests incur very little delay. A response-time CDF shows the fraction of requests with response times below a given value on the x axis. The closeness of the control policy's CDF plots to the baseline curve indicates how well the policy limits response-time degradation.

In some cases, because our control policy allows handling requests at lower RPMs, it produces even greater energy savings than $\text{DRPM}_{\text{perf,}}$ which services requests at the highest RPM to preserve performance. Having larger request windows provides greater energy savings because the disks have more opportunity to move to lower RPMs as the array controller makes performance checks less frequently.

From a performance viewpoint, the control policy can track the baseline system's response time behavior quite closely. For example, 90 percent of the requests for the exponential workload incur a response-time degradation of less than 5 percent, while the response time for the Pareto method is so minuscule, it is difficult to discern the different curves in the plot in Figure 4c.[2]

ENGINEERING A DRPM HARD DISK

Building a multispeed hard disk makes the design process more complex. Because conventional disks operate only at a particular speed, the manufacturer can design most of the drive's components for that specific RPM. However, for a DRPM disk, ensuring correct and reliable operation requires designing some components to accommodate speed changes. Doing so involves overcoming several design hurdles.

First, the hard disk's spindle motor requires variable-speed rotation. The *pulse-width modulation*

technique controls the motor's speed. PWM achieves speed control by switching the power supply to the motor on and off at a certain frequency, called the *duty cycle*. Although conventional disks employ only a single duty cycle, a DRPM disk requires multiple duty cycles. Fortunately, operating a DC brushless motor at multiple speeds is fairly straightforward.

A more complicated issue for DRPM pertains to the head's *fly height*—the height at which the disk-head slider flies above the platter's surface. The fly height is sensitive to the speed at which the platters spin. If the fly height drops too low, a head crash can occur. On the other hand, if the height becomes too great, improper magnetization can cause data errors. Therefore the fly height must remain more or less constant over the entire range of linear velocities the spindle system supports. Some slider designs[7] can maintain this optimal fly height over a wide range of RPMs.

Another important design concern involves the head-positioning servo and the data channel for multispeed operation. In hard disks, positioning the head requires accurate information about the tracks' location. This information is encoded by the manufacturer as servo signals on special servo sectors that are inaccessible through normal read or write operations. The actuator uses this servo information to accurately position the head over the tracks' center.

The controller must sample the servo information at a specific frequency to position the head properly. As the storage density increases, the number of *tracks per inch* increases, requiring higher sampling frequencies. The TPI sampling frequency is directly proportional to the disk's RPM, which might make properly sampling the servo information at lower RPMs impossible. Some researchers[8] have addressed this problem by designing a servo system that operates at both low and high RPMs. This approach uses a data channel that can operate over the entire range of data rates for the different RPMs—an effective strategy given that a channel's data rate is directly proportional to the RPM.

DRPM can help strike a balance between performance and power tradeoffs while recognizing that disk requests in server workloads can present relatively shorter idle times. This area presents opportunities for further innovations in disk manufacturing, such as the currently available non-server-class disk from Sony,[9] which can be configured for variable RPMs. Other opportunities include

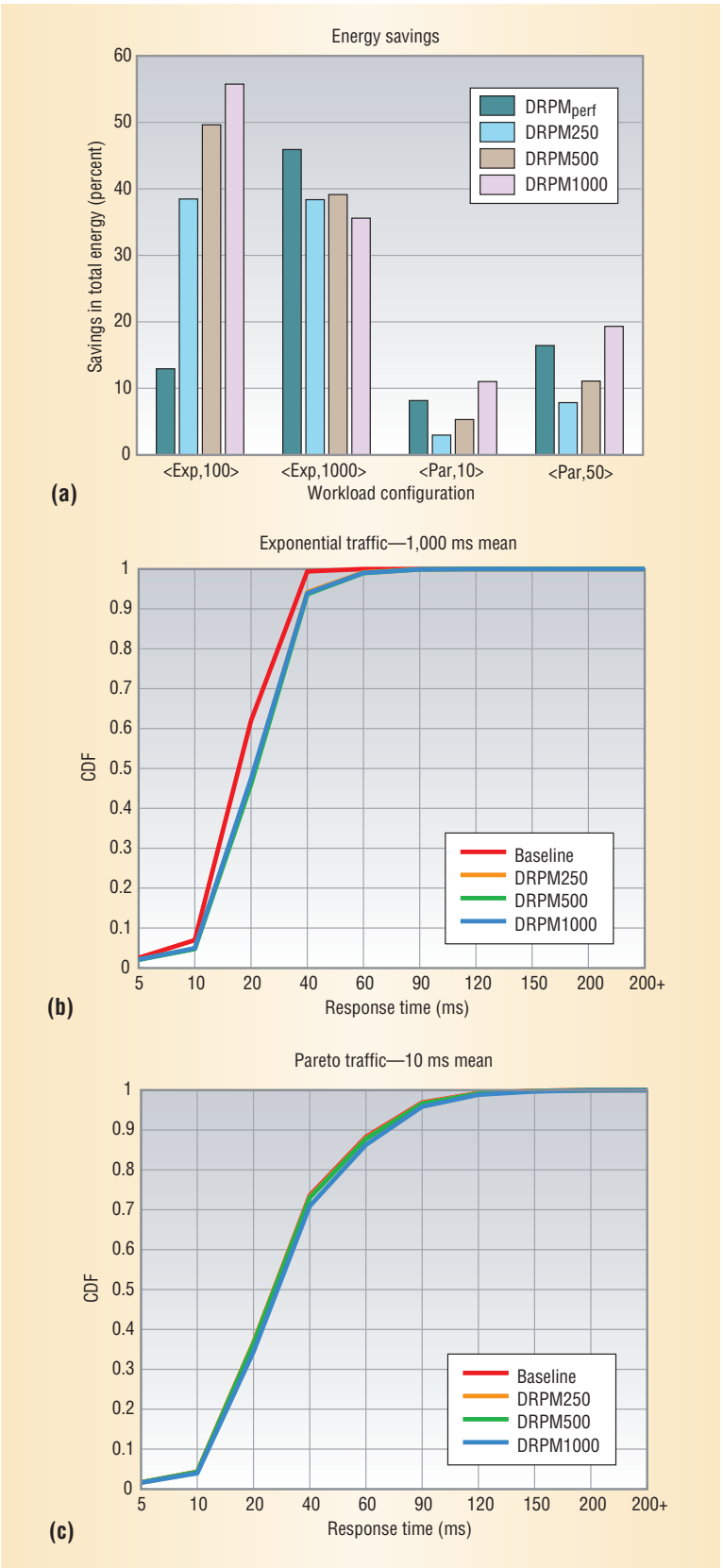

Figure 4. DRPM control policy results for an upper tolerance of 15 percent and a lower tolerance of 5 percent. (a) The savings in total energy consumption for the exponential and Pareto methods, (b) exponential method performance results for request window sizes of 250, 500, and 1,000 I/O requests, and (c) the same results and request window sizes for the Pareto method.

extending systems software support[10] to amplify these approaches' savings, such as buffering and prefetching mechanisms to increase disk idleness. Conceivably, researchers could even use a mix of disks, either with DRPM or with constant but different RPMs,[6] to investigate these tradeoffs. ■

References

1. F. Beck, "Energy Smart Data Centers: Applying Energy Efficient Design and Technology to the Digital Information Sector," Renewable Energy Policy Project, Research Report no. 14, Nov. 2001; www.repp.org.
2. S. Gurumurthi et al., "DRPM: Dynamic Speed Control for Power Management in Server Class Disks," *Proc. Int'l Symp. Computer Architecture*, IEEE CS Press, 2003, pp. 169-179.
3. S. Gurumurthi et al., "Interplay of Energy and Performance for Disk Arrays Running Transaction Processing Workloads," *Proc. Int'l Symp. Performance Analysis of Systems and Software*, IEEE Press, 2003, pp. 123-132.
4. T. Heath et al., "Application Transformations for Energy and Performance-Aware Device Management," *Proc. Int'l Conf. Parallel Architectures and Compilation Techniques* (PACT 2002), IEEE CS Press, 2002, pp. 121-130.
5. Y-H. Lu et al., "Quantitative Comparison of Power Management Algorithms," *Proc. Design Automation and Test in Europe*, ACM Press, 2000, pp. 20-26.
6. E.V. Carrera, E. Pinheiro, and R. Bianchini, "Conserving Disk Energy in Network Servers," *Proc. Int'l Conf. Supercomputing*, ACM Press, 2003, pp. 86-97.
7. N. Kojima et al., "Flying Characteristics of Novel Negative Pressure Slider 'Papillon,'" *J. Applied Physics*, vol. 81, no. 8, 1997, pp. 5399-5401.
8. H. Yada et al., "Head Positioning Servo and Data Channel for HDDs with Multiple Spindle Speeds," *IEEE Trans. Magnetics*, vol. 36, no. 5, 2000, pp. 2213-2215.
9. K. Okada, N. Kojima, and K. Yamashita, "A Novel Drive Architecture of HDD: 'Multimode Hard Disc Drive,'" *Proc. Int'l Conf. Consumer Electronics*, ACM Press, 2000, pp. 92-93.
10. A.E. Papathanasiou and M.L. Scott, *Increasing Disk Burstiness for Energy Efficiency*, tech. report 792, Univ. of Rochester, 2002.

Sudhanva Gurumurthi *is a PhD candidate in the Department of Computer Science and Engineering at Pennsylvania State University. His research interests include computer architecture, low-power design, and the evaluation of computer systems. Gurumurthi received a BE in computer science and engineering from the College of Engineering Guindy, Anna University, Chennai, India. Contact him at gurumurt@cse.psu.edu.*

Anand Sivasubramaniam *is an associate professor in the Department of Computer Science and Engineering at Pennsylvania State University. His research interests include operating systems, computer architecture, and performance evaluation. Sivasubramaniam received a PhD in computer science from Georgia Tech. Contact him at anand@cse.psu.edu.*

Mahmut Kandemir *is an assistant professor in the Department of Computer Science and Engineering at Pennsylvania State University. His research interests include embedded systems, power-aware computing, and optimizing compilers. Kandemir received a PhD in electrical engineering and computer science from Syracuse University. Contact him at kandemir@cse.psu.edu.*

Hubertus Franke *is a research staff member at IBM's T.J. Watson Research Center. His research interests include parallel and distributed computing, especially message-passing systems, operating environments, and architectures. Franke re- ceived a PhD in electrical engineering from Vanderbilt University. Contact him at frankeh@watson.ibm.com.*