GreenFS: Making Enterprise Computers Greener by Protecting Them Better

Nikolai Joukov
IBM T.J.Watson Research Center
19 Skyline drive
Hawthorne, NY, USA 10532

Josef Sipek
Computer Science Department
Stony Brook University
Stony Brook, NY, USA 11794-4400

ABSTRACT

Hard disks contain data—frequently an irreplaceable asset of high monetary and non-monetary value. At the same time, hard disks are mechanical devices that consume power, are noisy, and fragile when their platters are rotating.

In this paper we demonstrate that hard disks cause different kinds of problems for different types of computer systems and demystify several common misconceptions. We show that solutions developed to date are incapable of solving the power consumption, noise, and data reliability problems without sacrificing hard disk life-time, data reliability, or user convenience.

We considered data reliability, recovery, performance, user convenience, and hard disk-caused problems together at the enterprise scale. We have designed GreenFS: a fan-out stackable file system that offers all-time all-data run-time data protection, improves performance under typical user workloads, and allows hard disks to be kept off most of the time. As a result, GreenFS improves enterprise data protection, minimizes disk drive-related power consumption and noise and increases the chances of disk drive survivability in case of unexpected external impacts.

Categories and Subject Descriptors

D.4.2 [**Software**]: Operating Systems—*Storage Management*

General Terms

Design

Keywords

Continuous Data Protection, backup, power efficiency

1. INTRODUCTION

Hard disks are mechanical devices. When their motors are spinning, they consume power, generate noise, and are sensitive to shocks and impacts.

Previous studies showed that hard drives consume from 5% to 31% of the desktop and notebook computer power [10, 28]. This not only increases the overall IT power consumption and reduces battery life of portable computers but also increases the amount of heat generated by the system. Excessive heat can cause discomfort and increase the power consumption of fans and air conditioners. In a densely packed server room heat removal can easily double the total electricity consumption [22]. In turn, electricity constitutes about 50% of the total cost of ownership of today's data centers. Office computers consume about 1% of the electricity produced in the United States with additional 1.2% consumed by the data centers [23]. Given the expected growth of the number of computers these fractions are expected to increase in the future.

Table 1 shows approximate retail prices of hard disks and the power they consume. One can see that during the expected hard disk life-time of five years, a disk consumes roughly the same amount worth of electricity as the initial hardware. The situation is especially critical in major cities where the electricity prices are the highest and the power supply increases may not be possible at all [25]. In these cases the only possibility to expand the businesses is to lower the existing power consumption.

Even low intensity office noise reduces productivity, motivation, ability to learn, and can even cause health risks [11]. Noise levels of the average desktop hard drives rotating at 5,400–7,200 RPM are typically between 30 and 50 dBA. High-performance hard disks with higher rotational speeds generate more noise. For comparison, CPU fans and case fans on commodity computers usually generate about 30 dBA of noise [7]. Disk head seek-related sounds distract the users even more due to their irregular nature. While one may argue that it is possible to manufacture computers with better sound-insulation we propose an approach that can reduce the level of office noise with existing hardware.

Spinning hard disk drives are fragile and sensitive to shocks. The speed of the disk heads moving over the disk platters is high and can exceed 150 miles per hour for 15 KRPM drives. In case of an abrupt impact or vibration, disk heads can touch and damage the platter surfaces. Gyroscopic effects exacerbate the problem for mobile devices. Even small disk platter damage can create a chain reaction of collisions of the particles scratched from the platters, the disk heads, and the disk platters. Therefore, even a single collision

	Unit cost ($)	Power (Watts)	Power cost ($/year)
Desktop HDD	100	10	18
Server HDD	250	15x2	53

Table 1: Approximate costs of the hard disks and related power costs. The server power cost includes the cost of cooling and UPS losses (x2 multiplier). Electricity price is assumed 20c/kWh (city commercial [9]).

EuroSys'08, April 1–4, 2008, Glasgow, Scotland, UK.
Copyright 2008 ACM 978-1-60558-013-5/08/04 ...$5.00.

of the disk drive head and the platters can result in the rapid loss of the drive and loss of the data [16].

Non-rotating disks in the stand-by mode consume an order of magnitude less power and generate less heat than busy disks and at least three times less than idle rotating disks [28]. Non-spinning disks are silent. They are also typically four to five times less sensitive to shocks and thus are more reliable (e.g., [13]). Therefore, it is desirable to minimize the time the disks are spinning.

Most solutions to the "rotating hard disk" problem that exist today target either portable devices, servers, or RAID controllers. "One media" solutions are trying to solve the problem with only one storage device. Multiple studies targeted one hard disk systems and designed policies to spin the disk up and down [10, 32]. Unfortunately, it is impossible to predict the future (including the disk access patterns), and delaying writes increases the risk of losing data. However, the biggest problem for 3.5" disks found in servers and desktops is the limited number of spin-up cycles that they can sustain. A typical hard disk can tolerate only about 50,000 cycles, which translates to about one cycle per hour if we assume that a disk's life-time is five years. This is why spinning the disks down is disabled or configured for very long intervals on most systems.

Completely disk-less clients [27] add inconvenience for the users and administrators. Desktop solutions have high latency and become unusable in case of network problems (that happen in real life). Also, such systems have considerably different administration process, which is not confined by the machine itself. That is why disk-less desktops and servers have limited adoption. With the recent increase of the sizes of flash memory it is expected that flash memory may replace the system disks. However, the sizes that are available today and at least in the near future are still much smaller and more expensive than users need.

Solutions that combine multiple possibly different disks were shown to be more effective for server-type workloads [6, 8, 51]. Unfortunately, servers and desktops have only one system disk. A combination of flash memory and hard disks partially solves the problem [5] but still can result in shortened life-time and long access latencies in case of the flash memory read misses. Previous attempts to augment the disk and flash with the network connectivity to store the data were shown to improve performance and prolong battery life on mobile systems [31]. However, they can shorten the disk life-times and increase power consumption on the server and, as a result, overall on the enterprise scale.

Better sound isolation and reaction to computer case acceleration are the common ways to address the problems of hard disk noise and fragility [16].

We designed GreenFS—a file system that provides hierarchical run-time data protection for all data and allows most enterprise hard disks to be kept in the stand-by state (without platters rotating) most of the time. Data is an expensive and frequently irreplaceable asset and data reliability is of paramount importance for most if not all enterprises. We observed that modern data backup mechanisms such as run-time replication or continuous data protection (CDP) and overall enterprise storage power consumption must be considered together. We also decided that our power saving mechanisms must not decrease data reliability in order to be used in real enterprises. GreenFS substantially increases data reliability, decreases overall power consumption, makes enterprises greener, and increases user convenience by improving performance, and decreasing office noise levels.

The rest of this paper is organized as follows: we describe our design in Section 2, implementation in Section 3, evaluation in Section 4, related work in Section 5, and conclude in Section 6.

2. DESIGN

Data reliability and availability are usually the most important requirements for storage systems. Traditional power optimization solutions frequently contradict these requirements and decrease user and administration convenience. For example, frequent spin-up and spin-down operations significantly decrease the life-time and thus reliability of the hard disk drives. As a result, these features are usually disabled or configured for about hour long time-outs on most servers and desktops. Notebook hard disks can survive about an order of magnitude more spin-up operations but will still wear out within a year if only the break-even balance of power is considered. Similarly, disk-less clients degrade performance and become nonoperational in case of network infrastructure problems.

In addition to power consumption, hard disks pose a set of other problems such as noise, fragility, and ease of being stolen or lost. However, servers, desktops, and mobile systems have different disks and different deployment scenarios, which makes some of the above problems important or completely unimportant. For example, a disk in a notebook consumes almost no power in the idle state and its power consumption optimization not only makes no sense at the enterprise scale but usually has negligible effect on the battery life. Similarly, a desktop in the enterprise is almost always reliably connected to the fast local network whereas a notebook can get disconnected at any time.

When designing GreenFS our goal was to take care of different real computer system-disk caused problems. We believe that the only way to design a practical system like that is to improve data reliability, convenience for the users and administrators, and at the same time decrease the power consumption and noise. These design priorities are driven by the market: data and human labor are much more expensive than electricity and hard disks.

Hard disks fail, fail inevitably and unexpectedly [1, 38]. People make mistakes and overwrite or delete useful data. Hard disks or whole computers get lost or stolen. Data backup systems try to minimize the consequences of these harmful events. Traditional backup systems create snapshots of a subset of files on a periodic basis. This poses two problems:

1. Some important data may be left unprotected due to subset of files selection mistakes (which is usually realized when it is already too late) and

2. The most recent (and thus frequently most important) data updates are not protected.

The first problem could be solved by backing up whole hard disks. However, it is usually time consuming and considered prohibitively expensive because of the expensive storage systems used for backups. Also, increasing the amount of backup storage increases the enterprise power consumption. The second problem is partially solved by the run-time data replication. In addition, reverting to some earlier version of the file is frequently desirable. For example, if a user deletes a portion of the document by mistake. Continuous Data Protection (CDP) [26] preserves backup copies for every data update on-the-fly. This allows users to roll-back any file to any previous state in time. Unfortunately, mobile users are still left unprotected when not connected to a reliable network link.

Summarizing the above observations, GreenFS must:

- Provide run-time data protection (CDP or at least replication) of each and whole hard disk in the enterprise;

- Do so even when a desktop loses connectivity due to temporary network problems or when mobile clients are away from the network infrastructure;

- Avoid significantly increasing the cost of required backup storage;
- Spin the local hard disks up for short periods of time and only several times a day;
- Provide data access latency and bandwidth similar to the operation with the local hard disks at least under typical user workloads;
- Require minimal hardware and software modifications in existing infrastructure.

To solve the above challenges we employed three design solutions: (1) buffered all-time protection; (2) reversed backup operation; and (3) all-data protection. In addition, GreenFS is modular to fit any existing enterprise IT infrastructure.

2.1 Buffered All-Time Protection

Figure 1 shows the architecture of GreenFS. We will describe it starting from the clients (desktops, notebooks, servers) and then describe the backup server architecture.

The GreenFS client is a file system that performs run-time backup. It is designed in a hierarchical way: flash memory serves as a buffer that keeps data updates (potentially with their versions) when the remote server is unavailable. The combination of the flash layer and the remote server contain all the data necessary to recover any file or any file version at any time. Once network connectivity is reestablished collected data is submitted to the remote server.

2.2 Reversed Backup Operation

GreenFS reverses the use of backup and local hard disk storage under the normal conditions: all read and write requests are normally sent to the backup target and not to the disk. As described above, GreenFS client file system uses a flash layer between itself and a remote server. The flash is used for both: keeping data updates and most recent version of the frequently accessed files to improve performance. Therefore, the read access latency is usually defined by the fast flash layer and not the network. In case of CDP all versions except the latest one get discarded from the flash layer after they are stored on the server to free up space for caching. If a client is connected to the server we can use the whole flash layer for caching and replace least frequently accessed data with more frequently accessed data for improved performance.

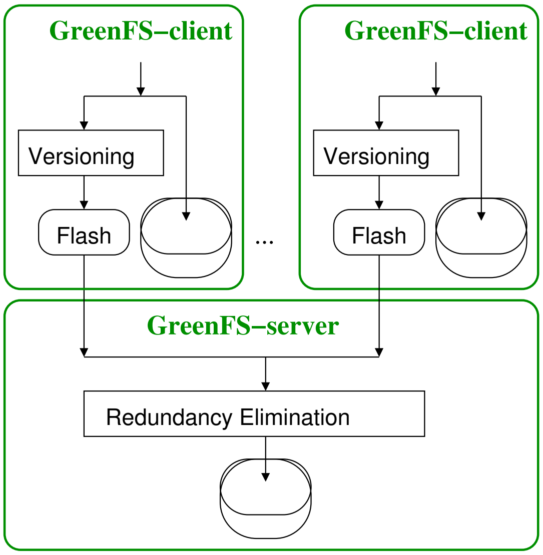

Figure 1: GreenFS hierarchical all-data all-time Continuous Data Protection with redundancy elimination.

GreenFS reverses the roles of the backup server and the local hard disk: the local hard disk becomes a backup of the data stored remotely. It is used when the local network infrastructure is having problems (which happens even in enterprise environments), is not sufficient for high-bandwidth workloads (which rarely happens on user workstations and even server system disks), and when mobile clients operate in the disconnected mode or with poor network connectivity. The reverse mode of operation allows GreenFS to keep the system and other local workstation hard disks off most of the time. In a sense, the local hard disk becomes a recovery mechanism in case of server connectivity problems or when remote storage bandwidth or latency would cause noticeable inconvenience for the users.

The data updates are synchronized with the local disk several times a day based on several conditions:

1. GreenFS synchronizes the data on system shut down. This guarantees that even if the network connectivity to the backup server is not available upon next boot up operation the system will still have all the data locally to operate autonomously.

2. GreenFS marks page cache pages that were modified and not committed to the local disks (even if they were committed to the remote storage system). When the system runs out of memory and needs to discard such pages GreenFS can be configured to: 1) spin up the disk and commit them to the local disk and 2) drop the pages. The first configuration is necessary for mobile computers that may become disconnected from the server at any time. They are usually equipped with 2.5” hard disks and can sustain relatively frequent spin up operations. Also, this mode may be necessary in the office environments with unstable network connectivity. The second mode is suitable for permanent servers and workstations. Even if they are rebooted or their software crashes their local hard disks can be resynchronized during the following boot time.

3. We spin up the local hard disk if the bandwidth or latency to the remote storage system cause noticeable inconvenience for the users. Thus, we do not spin up the disk just because we detect that we can save power by using the local storage (which is a rare case as described in Section 2.5) but we spin the disk up if there is a long period of high bandwidth data read activity. We do the same for writes if the system page cache gets filled with outstanding write requests. Again, this is necessary to avoid user inconvenience due to excessive cache purging and improve data reliability.

4. In any case, GreenFS periodically synchronizes the copies at configurable frequency. For example, at least once a day or at particular times of low system activity.

5. A user may want to manually turn on the disk for a limited amount of time for some specific reason.

GreenFS keeps information about the rate of spin up operations and balances the user convenience accordingly. Thus, GreenFS will not spin up the local hard disk even in case of high bandwidth utilization if the disk was spun up too many times within the last several days.

It is commonly believed that flash memory can extend the local memory cache size and therefore improve performance on the client [36, 42]. In case of GreenFS, a flash layer can also significantly reduce the load on the server. The information contained on the server belongs to a particular client and is not supposed to be changed by any other clients. This allows GreenFS to avoid costly data revalidation and assume that the information contained in the client flash layer is authoritative. Reduced load on the server keeps its price low and only the minimal number of disks spinning.

2.3 All-Data Protection

GreenFS uses run-time data protection for all data on all local disks because: (1) people make mistakes when deciding which files to backup and (2) reversed backup mode of operation (described in Section 2.2) assumes that all the local data can be found on the remote server.

This creates two potential problems: (1) reliable storage used for backups is expensive and dramatic increase of its size is prohibitively expensive; (2) backup storage consumes power and generates heat, which we are trying to avoid. We propose two solutions to these problems: (1) redundant data elimination and (2) hierarchical backup.

2.3.1 Redundant Data Elimination

In large enterprises there is a significant similarity of the data stored on the local disks. First, there is a small set of installed operating systems with many files that do not differ from installation to installation. In fact, with automated patching procedures most computers are running exactly the same versions of the operating systems and other software. People working on related projects have similar documents and emails that were distributed and stored locally. It was shown experimentally that data redundancy elimination in the enterprises can reduce data sizes by about three times [14, 24].

A related enterprise desktop user characteristic is relatively small data sets. Unlike at home, users store less multimedia content on their computers at work. Combination of small data sets and efficient and low-cost compression (duplicate elimination) allows us to store all data of many users on a much smaller number of server hard disks.

User workloads in the enterprises typically exhibit high degree of spatial and temporal locality as well as characterized by low bandwidth consumption and low I/O rates. This makes it possible to efficiently use multi-disk technologies to reduce the server storage cost and power consumption [6, 8, 48, 51]. For example, a gear-shifting RAID [44] can use D (two or more) 1TB disks to provide redundancy and store data of about $50 \times (D-1)$ users assuming that the average user data set size is 60GB and it contains 2/3 of redundant data. At the times of higher server load such systems can increase the number of involved disks and performance but also increase their power consumption.

2.3.2 Hierarchical Backup

The cost of the storage systems skyrockets with the increase in their reliability. If even after the data sets are compressed the resulting size is deemed to be too big and expensive to store we can use hierarchical data backup. The first backup level is used to store all the data from all computers and is big and may be not very reliable (e.g., an entry-level RAID system). The second backup level is a reliable, potentially expensive, and small storage system used to store the most important user data. Normally, people use only the second layer. The second backup system may be slow. For example, an outsourced service with low access bandwidth and high latency or a tape. In this configuration the primary backup system is used not just as a cache (e.g., as in [8]) to the secondary storage system but also keeps and protects a superset of the secondary storage data.

2.4 Reliability

GreenFS stores data on three different types of storage with different reliability characteristics and thus provides different data reliability in different modes of operation.

2.4.1 Connected Operation

First, lets take a look at the connected mode of operation when GreenFS stores data updates remotely, uses flash for caching, and sometimes synchronizes the data with the local disk. In this mode data pages and inodes are never released from the memory cache before an acknowledgment of successful commit to the server is received. Even after that point the data is still kept in memory until the page is reclaimed by the system.

As we described earlier, to evict a page GreenFS can be configured to either save the page to the local disk or drop it. In both cases the data is reliably stored on the remote storage with redundancy (e.g., on a RAID). This means that even if the system crashes before the data is saved to the local disk it is still saved with redundancy and thus higher reliability remotely. After the system is restarted it fetches the data from the server and increases the data reliability even further by replicating it locally.

2.4.2 Disconnected Operation

Second, let us take a look at the disconnected mode of GreenFS operation when GreenFS is not connected to the server for example due to a network unavailability. This mode of operation is supposed to be rare for server and desktop systems (e.g., a disaster, a server maintenance, or a network outage). Therefore, desktops and servers spin up their disks and use flash as a cache to improve performance of small reads (this operation is similar to hybrid drives [43]). Keeping the disks spinning allows us to minimize the number of spin up operations and provide high bandwidth and latency for the users.

On mobile systems disconnected from the server GreenFS operates similarly to desktops and servers except that it tries to prevent the disk from spinning most of the time. GreenFS saves updates to the flash memory and synchronizes updates with the disk from time to time. In this mode GreenFS also spins up the disk in case of flash cache read misses. 2.5 inch hard disks found in notebooks have relatively small spin up times and consume insignificant amounts of power (just a few percent of the system power as we will see later). However, they are likely to experience hits and shocks potentially leading to the data corruption. Therefore, GreenFS is not trying to aggressively optimize the disk energy in this mode but rather minimizes the time the disk is spinning without increasing the number of spin up operations beyond safe limits. This is accomplished by keeping the disk rotating for some time after it is spun up even if it is not energy efficient.

Similar to the disconnected operation, GreenFS can keep the pages in memory even after they are committed to the flash layer and write them to the local disk later. This may be necessary in cases when flash cannot be considered reliable or can be unexpectedly disconnected. Flash reliability is discussed next.

2.4.3 Flash Layer Reliability

Flash memory is known to have a limited number of write cycles. Single Level Cell (SLC) flash memory is faster and can sustain about 100,000 overwrites. Multi Level Cell (MLC) flash memory is slower and can sustain about 10,000 overwrites. However, MLC flash memory is cheaper and is usually bigger. USB flash drives and most other flash devices today have built-in support of the wear-leveling. This means that even if a file system overwrites the same location of the flash memory the writes reach different physical locations. As a result, it should take about a month of constant writing at full available bandwidth to wear out a 4GB MLC flash drive and about a year for the same SLC drive [40]. Fortunately, system disks are not subject to constant write workloads lasting months. In fact, Under typical user workloads SLC flash is expected to last decades. Nevertheless, in the rare cases of write-intensive workloads MLC-type flash memory use with GreenFS should be avoided.

Note that even if the flash layer gets worn out the original (correct) data is still sent to the remote server or/and saved to the local disk. The data is not stored remotely or on a local disk only if the system crashes or gets ungracefully shut down before the data is stored on the local disk and only during the disconnected operation. Also, data could be corrupted by a worn out flash cache if it fails to detect data corruption during a partial page read. In this case this corrupted data may be submitted to the remote server and the local disk during a later partial write. Fortunately, flash memory devices usually have check sum or other mechanisms to verify the data integrity. Similarly, a data integrity verification stackable file system can do the same with higher guarantees if mounted between GreenFS and a flash disk [21]. Even if the corrupted data propagates to the server an earlier unaffected version could be recovered by the means of CDP.

We believe that it is safe for GreenFS to use even the cheap MLC-type flash as the data read and write caching layer. Even under the worst case workloads, if data corruption occurs, it is unlikely to propagate and its effects can be efficiently minimized by the use of CDP. For unusual constant write-intensive workloads the use of SLC-type flash may be required.

2.5 Power Efficiency

GreenFS saves energy under various types of workloads: idle, small reads/writes, and for most large reads/writes:

2.5.1 Idle

Desktop and most server system disks are idle most of the time. During these periods of no data transfer activity GreenFS saves the power normally consumed by the disk motors for idle platter rotation as well as disk electronics. Flash memory used by GreenFS consumes negligible amounts of power. The network interfaces usually consume less power, are usually powered up most of the time in any case, and can be quickly powered up and down with minimal power cost. Enterprises are typically running the backup servers in any case so they consume power even with no GreenFS. Therefore, GreenFS saves all the power otherwise consumed by the idle disks.

However, even if we include the backup power consumption into the overall picture (or assume that the backup server power consumption increases two times) GreenFS still saves most of the storage-related power. For example, a small backup server consumes 100–200 Watts, which, when divided by the number of clients ($\geq$ 50), adds only 1–4 Watts per client. This is substantially less than the savings for desktops and servers but can be slightly higher than the savings for the notebooks. Fortunately, we are talking about fractions of a Watt per notebook in the worst case, which is negligible. Also, this extra power consumption is spent on the server and not affecting the notebook's battery. In fact, the battery lifetime gets slightly improved [3].

2.5.2 Small Reads and Writes

Hard disks consume significantly more power when they reposition their head (seeking) [15]. GreenFS fetches most data from the flash layer with minimal power consumption. Even in the rare case of the flash cache miss it is more efficient to read the data from the remote server's cache than the local disk. This is because even wireless network interfaces consume less energy than disk drives for small transfer sizes: disks consume energy to spin up, seek with the head, and have small aggregate bandwidth when seeking for data. Small writes are sent to the remote backup server, which is more efficient than spinning up the local disk. At a later time, GreenFS saves the accumulated updates to the local disk, which is more efficient due to fewer seek operations and only one instead of many spin up operations.

2.5.3 Large Reads and Writes

Hard disks may become more power efficient compared to networks when transferring large amounts of data. This is because their bandwidth is sometimes higher, which results in smaller transmission times. The total amount of energy necessary to transfer a file consists of the sum of power necessary to setup the transfer we call U_{up} (e.g., switch the network interface from PSM to CAM mode or spin the disk up) and transfer the data. The energy necessary to transfer the data is approximately equal to the amount of power (P) spent during the transfer multiplied by the transfer time. Equation 1 shows the total energy dependency on the transfer bandwidth (B) and file size (S). This linear estimation is consistent with our experiments and [3].

$$U = U_{up} + P \times \frac{S}{B} \tag{1}$$

Table 2 shows typical values of U_{up}, P, and B for server, desktop, and notebook systems that we observed in the real enterprise environment (their specifications will be provided in Section 4).

One can see that reads as large as hundreds of megabytes can be more efficient when done from the server over the network compared to the desktop and server hard disks. This is because spinning up the local disk is expensive. Large writes over GreenFS are generally more expensive than without it because we need to write the data to the server and to the local disk in any case and the cost of spinning up the disks becomes small compared to the write process.

However, we assume (and later validate) that large write operations are rare. GreenFS is not trying to optimize power for large read and write workloads. Instead we spin up the disk when GreenFS detects a constant reading or writing process to minimize the user inconvenience. The exact amount of activity time that triggers a spin-up gets adjusted based on the past history of the frequency of spin-up operations to keep their total number low. Detection of the bandwidth-sensitive user activity can further improve GreenFS's decision making about spinning up the local disk [3].

Device	U_{up} (J)	P (W)	B (MB/s)
Server HDD	75	15	71
Desktop HDD	41	11	56
Notebook HDD	5	2.5	44
gigabit ethernet	0.12	1.5	86
100Mbps ethernet	0.12	1.2	11
802.11g	0.14	1.4	2

Table 2: Typical values of start up energy (U_{up}), power (P), and bandwidth (B) while transferring the data.

2.6 Noise reduction

GreenFS keeps the hard disks off most of the time, which eliminates the noise related to hard disk motors (spinning and seeking). In addition, hard disks produce high level of noise during spin up operations. This is especially noticeable and distracting due to significant change from low noise to high noise levels.

To address this problem GreenFS attempts to spin the disk up at times when the user is unlikely to be around or during the times the disk is expected to produce noise. Section 2.2 describes the cases when GreenFS turns the disk on.

1. On shut down users normally expect their computers to produce noise and therefore do not get distracted by the sound of the disk spinning up.
2. On servers and desktops GreenFS is frequently configured to purge the data from the cache when committed to the backup server under the memory pressure. Therefore, the disk is not spun up in that case. On mobile computers and computers configured to synchronize with the disk before dropping the memory pages GreenFS have to spin up the local disk. However, high memory pressure is usually associated with a heavy I/O activity and is usually caused and thus expected by the user.
3. Similar to the above, high levels of I/O activity are usually associated with some rare user-initiated operations and are therefore expected by the users.
4. GreenFS attempts to perform periodic synchronization at the times of no user activity when the user is likely to be away.
5. A user is not expected to get disturbed by the disk sound after requesting the disk to spin up.

2.7 Shock Susceptibility

Hard disks are about four to five times more fragile when their platters are spinning and especially fragile when the disk head is positioned over the rotating disk surface. Traditional disk shock protection systems move the disk head away from the disk surface with the data or spin the disk down when they detect that a computer system is subject to an acceleration (suspected fall) [16]. There are two problems associated with this approach: First, the system is not protected against external hits. For example, somebody may unexpectedly open a door and hit a non-moving notebook in somebody else's hands. Second, it takes 0.3–0.5 seconds to unload a disk head, which is more or equal to the typical notebook free-fall time [16]. This makes it necessary to predict free-falls before they happen and keep the hard disk inaccessible for at least a second each time a fall is suspected.

GreenFS keeps the disks off even when accessing the data on-the-go given that the network connectivity is available. When the network connectivity is unavailable, existing shock protection systems protect the disk as usually. Therefore, GreenFS provides much better protection when the network connectivity is available and allows for the same level of protection as without GreenFS otherwise.

2.8 Modular Design and Legacy Systems

Various enterprises have different software installation, update and security enforcement policies as well as various hardware running various operating systems. It is not uncommon in the enterprises to have legacy systems that run decades old hardware and software. Therefore, for a practical enterprise-level system it is important to support a variety of hardware and software systems as well as legacy systems that cannot be modified. Because of this we decided to use stackable file systems as building blocks to add desired functionality and standard network file system interfaces to communicate between clients and servers.

Stackable file systems are layers that intercept VFS-level requests, possibly modify the associated data or metadata and forward them further [50]. They can be mounted over file systems, directories, or files. Figure 2 shows a simple pass-through stackable file system called WrapFS that passes all requests unmodified. Stackable file systems can be stacked on top of each other to add multiple new features such as encryption [12, 47], anti-virus scanning (e.g., AVFS) [29], or versioning [30] at the same time. Stackable file systems allow addition of only the desired code to standard WrapFS-like templates on a number of operating systems including Linux, Windows, FreeBSD, and Solaris [50]. Another important advantage of stackable file systems is the ability to reuse well-maintained code of other file systems. For example, a stackable file system can be mounted over NFS or CIFS file systems and thus reuse their network protocols and already deployed servers (which is important in real deployment scenarios).

A different class of stackable file systems called fan-out stackable file systems forward requests to a number of file systems (in fact directories) mounted below [18, 34]. GreenFS is a stackable fan-out file system. It is mounted over three lower directories:

Local Disk local disk mount point,

Flash directory on a flash media, and

LAN a directory exported by a remote backup server (e.g., over NFS or CIFS).

An example GreenFS configuration is shown in Figure 3. In this example, GreenFS is mounted over a local disk mounted using Ext3; an encrypted flash memory device mounted using Ext3cow; and a remote NFS server mounted below VersionFS for continuous data protection. Here, GreenFS uses a standard NFS server that already existed in the organization. If it is possible to modify the backup server it should be configured to perform redundant data elimination/compression and store data with versions. This can be done by using existing versioning file systems on the server (e.g., VersionFS or Ext3cow [33]). Many storage systems support compression. If such support is not available one may use a stackable compression file system [49] or disk-based file systems that support compression. Similar reasoning is applicable to the local flash disk. For example, for performance reasons it can be mounted with no versioning over Ext2 (this will provide no CDP in the discon-

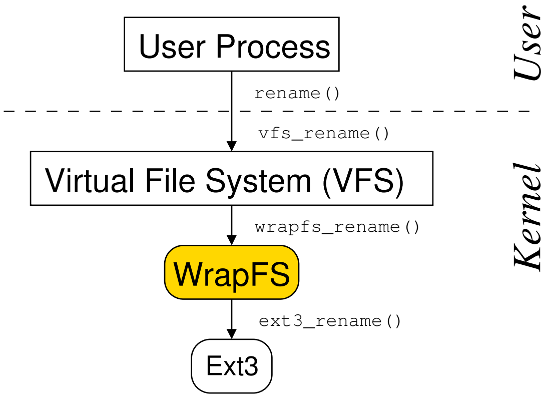

Figure 2: WrapFS is a pass-through stackable file system that forwards requests to the lower file system.

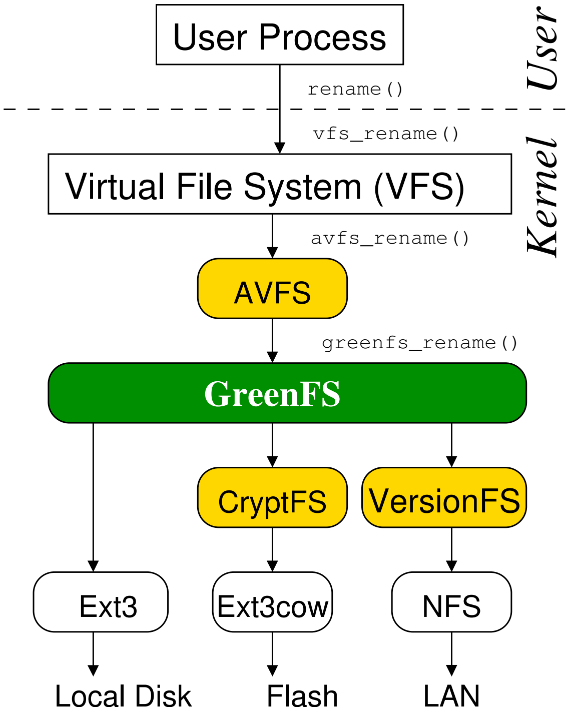

Figure 3: A possible GreenFS configuration.

nected mode of operation) or can be mounted over a versioning file system such as Ext3cow.

As we can see, modular design allows creating various configurations including legacy systems (clients or backup servers) and balance data protection, security, and convenience according to the specific needs.

2.9 Privacy and Security

GreenFS stores the data on a flash media and transmits it to and from a remote server, which requires special measures to provide privacy and security of the data.

2.9.1 Flash Layer

It is believed that future motherboards will have flash memory built-in [42]. This will make it as hard to steal or access the data on the flash as on the hard disk. In addition, removable flash devices (and future built-in flash memory) can be efficiently protected by encryption. As described above, GreenFS can be mounted over an encryption stackable file system mounted over the flash memory [12]. Similarly, some modern flash USB drives support built-in encryption of the data. Therefore, even if a removable flash drive with the cached data and data versions gets lost or stolen the data will be hard or impossible to recover.

In fact, the ability to remove the flash drive is important to protect the data stored on mobile systems if these systems are lost or stolen. A removable flash drive should be detached from the portable computer and kept separately (e.g., in a pocket). Even if the portable computer is lost or stolen the data contained on the flash and the remote backup server combined together will allow users to recover all their data including the most recent updates. In fact, the hierarchical CDP mode will allow users to recover any past version of any file.

2.9.2 Remote Backup Server

Enterprises commonly use encrypted virtual private networks to connect their clients to the server. GreenFS can simply reuse these protected channels. If such channels are not available GreenFS can be mounted over the network file systems that support encryption [41].

In case the client system is compromised an attacker can attempt to delete or modify the data [45]. This threat can be resolved by disallowing users to delete data versions on the backup server. In fact, modern regulations frequently require the same [37].

3. IMPLEMENTATION

As a stackable file system, GreenFS can be implemented for most OSs [50]. We implemented GreenFS as a loadable Linux kernel module for Linux 2.6.23. We based our implementation on UnionFS because it already supports the unioning functionality that we reuse in several modes of operation [34]. UnionFS consists of 8,987 lines of C code. We added/modified 1,458 lines of the kernel code. In addition, we wrote about 500 lines of the user mode code.

Kernel-mode operation is more efficient and secure but is harder to implement, maintain, and debug. Therefore, our objective was to make the kernel module changes simple but sufficient. GreenFS operates in the kernel during the normal operation over three or a subset of the lower file systems. GreenFS reads and writes data to and from the appropriate lower file systems. When it needs to release the memory pages or inodes it either writes them to the local disk or simply discards them (this is configurable at mount time). As we will describe below, GreenFS also maintains a number of status files on the lower file systems for the data recovery, performance, and synchronization purposes

The recursive file system synchronization happens entirely in the user mode. Thus, if during the mount process GreenFS detects that the mount directory has a status file indicating unclean unmount it asks to run an `fsck` process. A situation like this may be caused by a system crash, which could mean that some data was committed to the server or saved on the flash but not saved on the disk. GreenFS's `fsck` uses `rsync` to synchronize the file systems.

NFS by default waits for the remote host to respond even if the network connectivity becomes unavailable. This would cause GreenFS to wait on a related operation instead of switching to the disconnected mode of operation. To solve this problem NFS below GreenFS must be mounted with the `-o soft` option. This instructs it to timeout if the server is not responding for some time. GreenFS probes the remote server to detect when it becomes available again. Once, it happens GreenFS initiates a remounting process which involves the user-mode file system synchronization.

In addition to caching files on the flash GreenFS also caches negative directory entries meaning that the file does not exist. This allows GreenFS to avoid looking up non-existing files on the remote server. GreenFS also maintains status files to indicate that only a portion of some big file is cached on the local flash. Both of these functionalities are inherited from UnionFS and required only minimal modifications.

We focused our implementation on the server, desktop, and notebook systems. In the future it can be easily extended to support media players with hard disks and wireless connectivity. It was reported that on such systems hard disks consume significant share of the total energy [31]. Unfortunately, at the time of this writing we were unable to identify any such players supporting Linux. We also did not implement the redundancy elimination or compression server part because this was addressed elsewhere [24, 14].

4. EVALUATION

We have evaluated GreenFS using the following four computers to conduct our benchmarks:

BLADE An IBM LS20 blade in an IBM BladeCenter H Chassis with a 2.2 GHz dual core AMD Opteron 275 (2 MB cache) and 1 GB RAM. It had two Seagate Savvio Ultra320 SCSI 10 KRPM disks. This computer was used as an NFS server. It was located in the data center and connected to the enterprise 1 Gbps network.

SERVER A stock IBM xSeries 225 server with two 2.8 GHz dual core Intel Xeon (512 KB cache) CPUs and 2 GB of RAM equipped with LSI Logic Dual Ultra320 SCSI card and a 40 GB 10 KRPM Fujitsu SCSI disk. It was located in the same data center as the BLADE.

DESKTOP A stock IBM ThinkCentre 8187 desktop with a 3 GHz Pentium 4 (512 KB cache) and 512 MB of RAM. It had a single 100 GB IDE disk and was located in the same building as the servers.

NOTEBOOK A stock IBM ThinkPad T42 notebook with a 2.0 GHz CPU and 2 GB of RAM.

All machines ran Debian Linux with a vanilla 2.6.23-rc1 Linux kernel. Test machines were connected via a 1 Gbps network link via 3 routers in the real enterprise environment. We used an inexpensive MLC 1GB USB 2.0 flash drive.

We used the Auto-pilot benchmarking suite [46] to run all of the benchmarks. The lower-level file systems were remounted before every benchmark run to purge the page cache. We ran each test at least ten times and used the Student-t distribution to compute 95% confidence intervals for the mean elapsed, system, user, and wait times. Wait time is the elapsed time less CPU time used and consists mostly of I/O, but process scheduling can also affect it. In each case the half-widths of the confidence intervals were less than 5% of the mean.

4.1 Performance

We ran two benchmarks to evaluate GreenFS's performance on the DESKTOP connected with BLADE over NFS.

4.1.1 OpenSSH Compile

First, we compiled OpenSSH v.4.0p1 assuming that it represents one of the typical types of the user workloads. It is well known that compile benchmarks are usually CPU-bound [50]. However, they generate relatively large number of writes, which require special handling by GreenFS. We compiled OpenSSH over Ext2 and Ext3 file systems mounted over the local hard disk and flash; UnionFS mounted over one, two, and three lower directories created on the local disk, and GreenFS mounted over the local disk, flash drive, and an NFS file system.

Figure 4 shows the running times of the above experiments. We can see that compilation over Ext2 mounted over flash is only a fraction of 1% slower than compilation over the local disk. This is because the running time is dominated by the CPU. However, Ext3 mounted over flash is 6% slower because of the slow flash writes to the journal. UnionFS (the base of GreenFS) adds less than 1% overheads of CPU time per added branch. As a result, UnionFS mounted over three lower directories is only 2.3% slower than the lower Ext file systems running alone. Despite of the high bandwidth between the desktop and the NFS server, network introduces significant latency. As a result, compilation over NFS takes 2.8 times longer than over the local disk. No wonder, disk-less clients are not very popular. GreenFS uses local flash for caching and is only 9% slower than the local file systems mounted over the local disk and is only 2.4% slower than Ext3 mounted over flash.

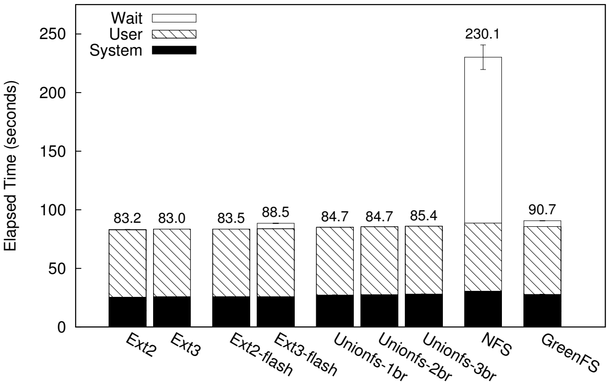

Figure 4: OPENSSH benchmark results.

4.1.2 Emacs Invocation

Invocation of programs takes time and requires users to wait. Our second performance benchmark was the time to invoke Emacs 22.1 in the standard configuration. We installed Emacs and its dependent libraries on a file system that was in use for two years before the experiments and thus aged. During the loading process Emacs reads several libraries, fetches and executes scripts, and writes some status information.

Figure 5 shows invocation times for the local hard disk, flash, NFS, and GreenFS. We can see that local hard disk corresponds to the longest time. This is because starting a program commonly requires accessing data at various locations, which requires seeking with the disk head. Flash memory has lower bandwidth but requires no seeking. As a result, the invocation process was 45% faster. We purged local caches before the runs but did not purge the cache on the NFS server. This is because we normally expect the server cache to be large and contain frequently used data. Recall that if the same binary is stored on multiple systems redundancy elimination

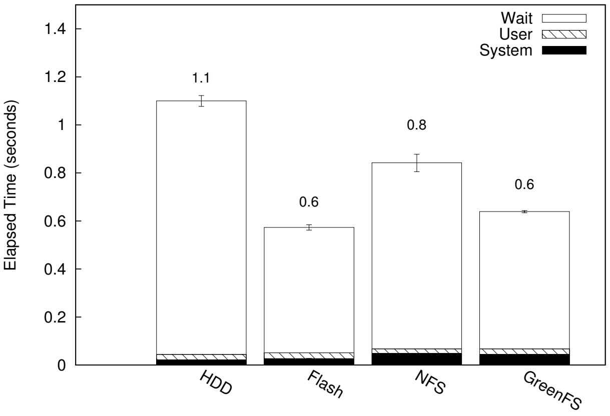

Figure 5: EMACS benchmark results.

will keep only one shared copy at the server. Our network link had a much higher bandwidth than flash but higher latency. Emacs invocation over NFS was 27% faster than from the local disk but 33% slower than from flash. GreenFS fetches the data from flash and sends a few writes caused by Emacs over NFS. As a result its running time is indistinguishable from the flash time.

4.2 Backup Server Load

We used OSprof to collect statistics of the user activity at the file system level [19]. This allowed us to accurately capture memory mapped operations that are frequently missed by the system call tracers. Table 3 shows some statistics about a representative trace of the normal kernel developer activity that we captured (JEFF). As we can see, the workload is heavily read-biased and its data footprint can fit in a modest flash memory device. For comparison we also included the same information about a trace used in a related project for mobile systems (PURCELL [31]).

Trace	JEFF	PURCELL
Number of ops	1,987,982	87,739
Duration (Hours)	26.48	27.66
Update ops	5.6%	6%
Working set (MB)	649	252

Table 3: File system traces details.

GreenFS's caching in memory and flash on the client side and no data revalidation-related activity significantly reduces the server load. We observed that the average rate of requests sent to the server was 1.3 requests per second. During day-long period of tracing the server experienced only 3 spikes of 10,000–15,000 requests per 30 seconds; 17 spikes of 1,000–10,000 requests per 30 seconds; and 505 spikes of 1,000–100 requests per 30 seconds sent from our client. The largest request was less than 1 MB. This type of load can be easily handled by an entry level storage system like our BLADE with no noticeable performance degradation for at least 50 GreenFS clients [39].

While collecting statistics about user workloads on several enterprise systems we noticed that under both Windows and Linux OSs background write operations happen pretty frequently. In fact, the longest disk idle interval we observed was slightly longer than a minute. These writes are caused by various processes usually related to the enterprise system management such as security monitoring. We attempted to disable these processes but quickly gave up because of their number, variations on different systems, and importance. This leaves no room for power optimizations using time-out based disk power management even if they can predict the future access patterns.

4.3 Power Efficiency

We measured overall system power consumption because we believe that optimizations can potentially increase the power consumption of CPUs, network interfaces, and other system components. To measure the power consumption of SERVER, DESKTOP, and NOTEBOOK systems we used a Diego Systems Series 200 power monitor. We inserted it directly between the test systems and the wall power outlet. We measured the power consumption of the BLADE system using the power monitoring infrastructure built into the BladeCenter.

Table 4 shows the idle power consumption of our test systems. In the idle state our systems (except BLADE) were running X and a window manager in the default configurations. NOTEBOOK' configuration is the NOTEBOOK system with the power saving features turned on: the screen is set to the minimal brightness and the CPU clock speed is scaled down. Idle state corresponds to the typical power consumption observed on desktops and notebooks most of the time. Even most servers today have utilization of only about 15% and their power consumption is not much different from the idle state [4, 22].

System	Original Power (W)	GreenFS Power (W)	Savings (%)
BLADE	90	N/A	N/A
SERVER	113	101	12
DESKTOP	54.1	46.8	14
NOTEBOOK	20.1	19.6	2.5
NOTEBOOK'	13.9	13.4	3.6

Table 4: Power consumption of the idle test systems.

The maximum power consumption of the BLADE system is 130 W. Assuming that it serves 50 GreenFS clients concurrently it adds at most 2.6 W per client. During the JEFF workload GreenFS spins up the disk only once—at night. Measuring the related power consumption and the power consumed for the data transfers we estimated that the average reduction of the total energy consumption per GreenFS client under the JEFF workload is 7%. This translates into saving 60% of storage-related power.

We found it interesting that today's notebook hard disks consume so little energy that its optimization becomes a nonsense. Nevertheless, this seems to be one of the well kept secrets today and we can still see that this problem attracts significant attention. We believe that disk drive survivability and data reliability are the most important optimization goals for the notebook drives. And that is what our all-data hierarchical CDP and keeping the disk off efforts are aimed to achieve.

4.4 Shock Protection

To provide an example showing how GreenFS protects hard disks against shocks in the real environments we performed the following experiment. Authors' offices are located in the same building on floors two and four. Table 5 shows hard disk protection related metrics observed while going between the offices using the stairs and the elevator.

As we can see, the network was available all the time while going using the stairs and was unavailable during just a few seconds in the elevator. The active protection system (APS) was turning the disk off for several seconds upon detecting the beginning of a possible notebook fall. This way, APS was protecting the notebook's hard disk against possible falls. Unfortunately, it is impossible to predict an accidental external hit in advance by measuring the notebook acceleration. For example, somebody else may hit our notebook by suddenly opening one of the four doors on our stairs. GreenFS kept the disk off all the time and thus maximally protected it from the accidental hits.

	elevator	stairs
trip time (sec)	85	58
network available (%)	94	100
disk is shock protected APS (%)	29	63
disk is shock protected GreenFS (%)	100	100
disk stop operations APS	4	6
disk stop operations GreenFS	0	0

Table 5: Comparison of GreenFS and Active Protection System as mechanisms to protect the disk against shocks.

5. RELATED WORK

5.1 Hard disk power saving solutions

Numerous existing solutions for single-drive configurations advocate prefetching the data and delaying writes to create intervals of disk inactivity when the disk can be put into the stand-by mode [10, 32]. However, delaying the writes increases the chances of the data loss. Data prefetching cannot be accurate in practice and can, in fact, increase the power consumption [51]. Worse yet, frequent disk spin-up operations can drastically decrease the disk life-time and increase the chances of the data loss [17]. In particular, solutions that power down the disk propose frequent disk spin-up operations based on the break-even balance between power consumption to spin the disk up and idle disk power savings. For example, for 16 second break-even intervals a notebook drive is expected to last less than a year under the worst case but possible workload [32].

Disk drives consume significantly more power when they are accessing the data or seeking. File system layouts that minimize the seek distances can reduce the seek-caused power consumption [15]. However, user workloads keep hard disks idle most of the time, which minimizes the aggregate effect since most of the time there are no seeks to optimize.

5.1.1 Solutions with multiple storage devices

Multi-disk solutions were shown to be more successful in conserving the energy and stopping the disks [8]. Multi-disk solutions based on the multi-speed disks were shown to be especially successful [6, 48, 51]. Unfortunately, these solutions are not readily applicable to desktops, notebooks, and handheld devices with only one disk drive.

Hybrid-drives with built-in flash memory caches have recently emerged on the market and some OSs already support them. The flash-based caches allow safely delaying the writes to the disk platters. On systems with little system memory hybrid-drives can also reduce the number of reads from the disk platters and thus increase the intervals of the disk inactivity and performance [5, 43]. Unfortunately, in case of flash memory misses hybrid-drives can significantly increase the latency of the disk requests. Similar to the single-disk solutions, hybrid drives experience excessive (albeit smaller) number of spin-up operations shortening the disk life-time. In addition, flash memory can sustain a limited number of writes. Because flash is an integral part of the drive the life-time of flash memory defines the life-time of the whole hybrid-disk. In addition, hybrid-drives are expensive, have limited availability, and their use requires replacement of the existing disk drives.

Disk-less servers and clients keep all the data at the remote storage systems and have no hard disks [27]. However, network latency makes this solution inconvenient for users. Also, network connectivity is not always available in practice and it adds frustration for the users when they cannot use even their own workstation. Anecdotal evidence suggests that many clients refrain from using disk-less servers because of the inconvenience with their administration.

Mobile systems frequently rely on the wireless network use for backups. Modern personal area networks when combined with traditional WiFi networks can further decrease the associated power consumption [2].

BlueFS is a file system to improve performance and battery life-time for small mobile computers [31]. However, it is not well suitable for other computer systems. Thus, its use can result in the accelerated disk wear (especially if tried on desktop or server hard disks), and increases overall power consumption if deployed in the enterprise environments.

5.2 Hard disk data protection

Periodic back up of data and run-time replication are standard ways to minimize the amount of data lost due to a hard disk failure. Unfortunately, run-time replication overwrites previously saved versions of the data, which are frequently required later. To solve this problem modern back up systems save versions of the data, which is called continuous data protection [26].

Modern notebooks support acceleration sensors attempting to detect the moment the notebook is falling down and stop the disk drive [16]. Unfortunately, such measures cannot react to abrupt accidental hits of a notebook or a handheld device when the notebook is at rest before the impact.

5.3 Stackable file systems

Stackable file systems were originally introduced in the early '90s [35]. They augment the functionality of existing file systems by transparently mounting a layer above them. Stackable file systems can be stacked on top of each other to add multiple new features such as encryption [12, 47], anti-virus scanning (e.g., AVFS) [29], data consistency verification [21], secure deletion [20], or versioning [30] at the same time. Stackable file systems are commonly used on Linux, Windows, FreeBSD, and Solaris [50]. A different class of stackable file systems called fan-out stackable file systems forward requests to a number of file systems mounted below (e.g., RAIF [18] and UnionFS [34]). GreenFS is a stackable fan-out file system.

6. CONCLUSIONS

We believe that the right solution to the hard disk related problems in the enterprises is to consider storage components together. We designed GreenFS—a client file system and a method to construct data backup systems and keep most disks off. It offers all-time all-data run-time data protection, and at the same time increases convenience for the users.

Power consumption of modern computer systems is a sum of consumptions of several not well related components. CPU, memory, video controllers, power supplies, displays, and storage all consume their not dramatically different slices of power. Usually, every such sub-component consumes 5–30% of the total power. Therefore, we believe that optimizing all these components is the right way to reduce the total power consumption of modern computer systems. Despite of the fact that we backup and protect all data we decrease the overall IT power consumption. We demonstrated that under typical user workloads we can save 60% of the overall storage-related power.

GreenFS is a solution to a variety of different disk drive-related problems on different types of systems in a unified way. The main benefits for different classes of systems are summarized in Table 6:

	Servers	Desktops	Notebooks
hierarchical CDP			Y
all-data CDP	Y	Y	Y
Small seek time	Y	Y	Y
Power	Y	Y	
Noise		Y	Y
Shocks			Y

Table 6: Classes of systems that benefit the most from all-time all-data CDP, flash caching, and keeping the disk off (power savings, noise reduction, shocks survivability)

Servers.

System hard disks in servers typically consume 10% or less of the total power. However, most data centers are running close to their maximum capacity today and cooling process increases the associated expenses twice. Therefore, even a relatively small decrease in the power consumption can allow otherwise impossible expansion of the data center and delay migration to a new one.

Desktops.

Hard disks in desktop computers consume about 15% of the total power and substantially increase the office noise levels. Keeping the disk off can save IT money and increase human productivity otherwise reduced by noise. Desktop users significantly benefit from GreenFS's all-data continuous data protection that provides protection for all their data.

Notebooks.

Unlike commonly believed, hard disks in modern notebooks consume insignificant 4% or less of the total power even when all the other power-hungry components as CPU and screen are configured to consume as little as possible. However, notebook hard disks are frequently subject to external impacts or theft. The ability to keep hard disks off allows to minimize the probability of data loss due to hard disk impacts without the need to interrupt normal user activity even on-the-go. Hierarchical continuous data protection is most beneficial for notebook users because it protects the data even when no network connectivity is available. Separately kept flash media is also an efficient way to recover the data after a notebook is lost or stolen.

Acknowledgments

Murthy V. Devarakonda provided guidance and supported us. We would like to thank Alain Azagury, Fred Douglis, Hai Huang, Hui Lei, Marcel C. Rosu, and Erez Zadok for fruitful discussions. We would also like to thank Charles P. Wright for his help with the Auto-pilot setup and Bulent Abali and James V. Norris for their help with the BladeCenter power measurements. Anonymous reviewers provided interesting and valuable comments.

7. REFERENCES

[1] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch. A five-year study of file-system metadata. In *Proceedings of the Fifth USENIX Conference on File and Storage Technologies (FAST '07)*, pages 31–45, San Jose, CA, February 2007. USENIX Association.

[2] M. Anand and J. Flinn. PAN-on-demand: Building self-organizing PANs for better power management. Technical Report CSE-TR-524-06, Computer Science and Engineering Division, University of Michigan, August 2006.

[3] M. Anand, E. B. Nightingale, and J. Flinn. Ghosts in the machine: Interfaces for better power management. In *Proceedings of MobiSys 2004*, pages 23–35, Boston, MA, June 2004. ACM.

[4] L. A. Barroso and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, December 2007.

[5] T. Bisson, S. A. Brandt, and D. Long. A hybrid disk-aware spin-down algorithm with I/O subsystem support. In *Proceedings of the International Performance Conference on Computers and Communication (IPCCC '07)*, New Orleans, April 2007.

[6] E. V. Carrera, E. Pinheiro, and R. Bianchini. Conserving disk energy in network servers. In *Proceedings of the International Conference on Supercomputers (ICS '03)*, San Francisco, CA, June 2003. ACM.

[7] Noise Pollution Clearinghouse. Noise control in PCs—reduction of noise in PCs, May 2004. `http://www.nonoise.org/resource/pcnoise/poweroid/poweroid.htm`.

[8] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *Proceedings of the Supercomputing Conference 2002 (SC2002)*, pages 1–11, Baltimore, MD, November 2002.

[9] Tokyo Electric Power Company. Service guide.

[10] F. Douglis, P. Krishnan, and B. Marsh. Thwarting the power-hungry disk. In *Proceedings of the Winter USENIX Technical Conference*, pages 293–306. USENIX Association, January 1994.

[11] G. W. Evans and D. Johnson. Stress and open-office noise. *Journal of Applied Psychology*, 85(5):779–783, 2000.

[12] M. A. Halcrow. eCryptfs: An Enterprise-class Encrypted Filesystem for Linux. In *Proceedings of the 2005 Linux Symposium*, pages 201–218, Ottawa, Canada, July 2005. Linux Symposium.

[13] Hitachi CinemaStar Hard Disk Drive Specifications Hitachi Global Storage Technologies.

[14] B. Hong, D. Plantenberg, D. D. E. Long, and M. Sivan-Zimet. Duplicate data elimination in a san file system. In *Proceedings of the 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies (MSST 2004)*, pages 301–314, College Park, MD, April 2004. IEEE.

[15] H. Huang, W. Hung, and K. Shin. FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP '05)*, pages 263–276, Brighton, UK, October 2005. ACM Press.

[16] IBM. Active Protection System whitepaper, October 2003.

[17] D. D.E. Long J. Rybczynski and A. Amer. Expecting the unexpected: Adaptation for predictive energy conservation. In *Proceedings of the First ACM Workshop on Storage Security and Survivability (StorageSS 2005)*, pages 130–134, FairFax, VA, November 2005. ACM.

[18] N. Joukov, A. M. Krishnakumar, C. Patti, A. Rai, S. Satnur, A. Traeger, and Erez Zadok. Raif: Redundant array of independent filesystems. In *Proceedings of the 24th International IEEE Symposium on Mass Storage Systems and Technologies*, pages 199–212, San Diego, CA, September 2007. IEEE.

[19] N. Joukov, A. Traeger, R. Iyer, C. P. Wright, and E. Zadok. Operating system profiling via latency analysis. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI 2006)*, pages 89–102, Seattle, WA, November 2006. ACM SIGOPS.

[20] N. Joukov and E. Zadok. Adding Secure Deletion to Your Favorite File System. In *Proceedings of the third international IEEE Security In Storage Workshop (SISW 2005)*, pages 63–70, San Francisco, CA, December 2005. IEEE Computer Society.

[21] A. Kashyap, S. Patil, G. Sivathanu, and E. Zadok. I3FS: An In-Kernel Integrity Checker and Intrusion Detection File System. In *Proceedings of the 18th USENIX Large Installation System Administration Conference (LISA 2004)*,

pages 69–79, Atlanta, GA, November 2004. USENIX Association.

[22] P. Khanna. Operation consolidation: reducing the number of servers can offer dramatic cost savings, but experts warn that trying to cram too much onto one box can backfire. *Computing Canada*, March 12 2004. by quoting G. Haff, senior analyst at Illuminata, Inc.

[23] J. Koomey. Estimating total power consumption by servers in the U.S. and the world. Technical Report Final Report, Lawrence Berkeley National Laboratory, February 2007.

[24] P. Kulkarni, F. Douglis, J. LaVoie, and J. M. Tracey. Redundancy elimination within large collections of files. In *Proceedings of the Annual USENIX Technical Conference*, pages 59–72, Boston, MA, June 2004. USENIX Association.

[25] Rakesh Kumar. Gartner: A message from data center managers to CIOs: Floor space, power and cooling will limit our growth, August 2006.

[26] G. Laden, P. Ta-Shma, E. Yaffe, M. Factor, and S. Fienblit. Architectures for controller based CDP. In *Proceedings of the Fifth USENIX Conference on File and Storage Technologies (FAST '07)*, pages 107–121, San Jose, CA, February 2007. USENIX Association.

[27] J. Layton. The coming of diskless clusters. *Linux Magazine*, October 2005.

[28] A. Mahesri and V. Vardhan. Power consumption breakdown on a modern laptop. In *Proceedings of the Workshop on Power-Aware Computer Systems (PACS 2004)*, Portland, OR, December 2004. IEEE Computer Society.

[29] Y. Miretskiy, A. Das, C. P. Wright, and E. Zadok. Avfs: An On-Access Anti-Virus File System. In *Proceedings of the 13th USENIX Security Symposium (Security 2004)*, pages 73–88, San Diego, CA, August 2004. USENIX Association.

[30] K. Muniswamy-Reddy, C. P. Wright, A. Himmer, and E. Zadok. A Versatile and User-Oriented Versioning File System. In *Proceedings of the Third USENIX Conference on File and Storage Technologies (FAST 2004)*, pages 115–128, San Francisco, CA, March/April 2004. USENIX Association.

[31] E. Nightingale and J. Flinn. Energy-efficiency and storage flexibility in the blue file system. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI 2004)*, pages 363–378, San Francisco, CA, December 2004. ACM SIGOPS.

[32] A. E. Papathanasiou and M. L. Scott. Energy efficient prefetching and caching. In *Proceedings of the Annual USENIX Technical Conference*, pages 255–268, Boston, MA, June 2004. USENIX Association.

[33] Z. N. J. Peterson and R. C. Burns. Ext3cow: The design, Implementation, and Analysis of Metadata for a Time-Shifting File System. Technical Report HSSL-2003-03, Computer Science Department, The Johns Hopkins University, 2003. `http://hssl.cs.jhu.edu/papers/peterson-ext3cow03.pdf`.

[34] D. Quigley, J. Sipek, C. P. Wright, and E. Zadok. UnionFS: User- and Community-oriented Development of a Unification Filesystem. In *Proceedings of the 2006 Linux Symposium*, volume 2, pages 349–362, Ottawa, Canada, July 2006.

[35] D. S. H. Rosenthal. Evolving the Vnode interface. In *Proceedings of the Summer USENIX Technical Conference*, pages 107–118, Anaheim, CA, June 1990. USENIX Association.

[36] M. Russinovich. Inside the windows vista kernel: Part 2. *Microsoft TechNet Magazine*, 2007.

[37] P. Sarbanes and M. G. Oxley. *Sarbanes-Oxley Act of 2002*. U.S. Government Printing Office, July 2002.

[38] B. Schroeder and G. A. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *Proceedings of the Fifth USENIX Conference on File and Storage Technologies (FAST '07)*, pages 1–16, San Jose, CA, February 2007. USENIX Association.

[39] SPEC. SPEC SFS97_R1 V3.0. `www.spec.org/sfs97r1`, September 2001.

[40] Silicon Systems. Increasing flash solid state disk reliability, April 2005.

[41] A. Traeger, K. Thangavelu, and E. Zadok. Round-trip privacy with NFSv4. In *Proceedings of the Third ACM Workshop on Storage Security and Survivability (StorageSS 2007)*, pages 1–7, Alexandria, VA, October 2007. ACM.

[42] M. Trainor. Overcoming disk drive access bottlenecks with intel robson technology. *Technology@Intel Magazine*, December 2006.

[43] A. A. Wang, P. Reiher, G. J. Popek, and G. H. Kuenning. Conquest: Better Performance Through A Disk/Persistent-RAM Hybrid File System. In *Proceedings of the Annual USENIX Technical Conference*, pages 15–28, Monterey, CA, June 2002. USENIX Association.

[44] C. Weddle, M. Oldham, J. Qian, A. A. Wang, P. Reiher, and G. Kuenning. PARAID: A gear-shifting power-aware RAID. In *Proceedings of the Fifth USENIX Conference on File and Storage Technologies (FAST '07)*, pages 245–260, San Jose, CA, February 2007. USENIX Association.

[45] J. Wires and M. J. Feeley. Secure file system versioning at the block level. In *Proceedings of the EuroSys 2007 Conference*, pages 203–215, Lisboa, Portugal, March 2007. ACM.

[46] C. P. Wright, N. Joukov, D. Kulkarni, Y. Miretskiy, and E. Zadok. Auto-pilot: A platform for system software benchmarking. In *Proceedings of the Annual USENIX Technical Conference, FREENIX Track*, pages 175–187, Anaheim, CA, April 2005. USENIX Association.

[47] C. P. Wright, M. Martino, and E. Zadok. NCryptfs: A secure and convenient cryptographic file system. In *Proceedings of the Annual USENIX Technical Conference*, pages 197–210, San Antonio, TX, June 2003. USENIX Association.

[48] X. Yao and J. Wang. RIMAC: A novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems. In *Proceedings of the EuroSys 2006 Conference*, pages 249–262, Leuven, Belgium, April 2006. ACM.

[49] E. Zadok, J. M. Anderson, I. Bădulescu, and J. Nieh. Fast Indexing: Support for size-changing algorithms in stackable file systems. In *Proceedings of the Annual USENIX Technical Conference*, pages 289–304, Boston, MA, June 2001. USENIX Association.

[50] E. Zadok, R. Iyer, N. Joukov, G. Sivathanu, and C. P. Wright. On incremental file system development. *ACM Transactions on Storage (TOS)*, 2(2):161–196, May 2006.

[51] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: Helping Disk Arrays Sleep through the Winter. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP '05)*, pages 177–190, Brighton, UK, October 2005. ACM Press.