2010 IEEE/ACM International Conference on Green Computing and Communications & 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing

Dynamic File System Migration for Energy Efficient Storage Management

Inuk You
youinuk@hanyang.ac.kr

Youjip Won
yjwon@ece.hanyang.ac.kr

Department of Electronics & Computer Engineering, Hanyang University
Seoul, Korea

Abstract— **Existing computer systems do not allow the user to access the system when turning off the disk containing the OS for saving energy. Thus existing studies have focused on devising effective disk spin up/down schemes. This paper proposes the scheme to keep on providing limited services as well as to improve the energy consumption in computer system. To do so, we employ the method that moves the root file system of the HDD to a RAM disk, in order to be able to access the system even in the case of a power-down of the disk containing the OS. Also, we realize the autonomic system migration by judging the system activity – the CPU and user activity. By doing so the power management system can reduce the power more remarkably than in existing systems, and in experimental evaluation we could save power consumption of 8.1 - 14.4%.**

Keywords: Energy-aware, File System Migration, Storage System

I. INTRODUCTION

A. Background

Of modern electronic products, computers occupy significant power consumption. According to [1], there are currently more than 180 million computers in use which consume approximately 2% of the annual power in the U.S. Servers and data centers consume about 61 billion kWh, that is, 1.5% annually and imaging products of 260 million to around 3% of annual power spent in the U.S. Though the power consumption that storages occupy varies with the computer system and the number of drives, general desktop computers in idle consume around 10.7% for hard disks [1]. However, the systems using plenty of storage like data centers consume approximately 26% of power only for the hard drives, and the percentage is expected to increase as we demand faster and larger capacity ones requiring more power [2-7]. Recent TPC-C benchmarks represent the disks consume the largest power – 63% of the total power – and the storage subsystem occupies 79.1% of all power in entire system [8].

While the capabilities of modern computer components are rapidly advancing, power consumption is also increasing accordingly. In the case of computer servers, in order to achieve higher performance more CPUs and memory are required. Because these server components (monitors excluded) consume the most power, their power management becomes more important than before and there are many related studies. Hard disks consume less power than CPUs and memory, however, in case of the storage servers equipped with multiple HDD, they can have the highest power consumption of all the components in the system. In this case, HDD power saving becomes an important factor. We can also contribute to power saving by effective management of hard disks in single HDD systems.

B. Related study

Modern HDD has a variety of states and power consumption can be reduced by changing the state to make the HDD inactive by spinning down. To access the hard disks, however, requires transition back to an active state which consumes too much time and energy. As such, existing disk power management methods focus on achieving the most effective disk spin up/down. In [9], the disk spin down algorithm was implemented by a Linux kernel, demonstrating reduced spin up latency.

In the case of mass storage systems, a technique of grouping disks and spinning down the areas responsible for lighter workloads is introduced. [10] suggests that spinning down many drive groups with reducing performance-down of system. In a similar study [11], extra drives are spun down while maintaining system performance. [5] introduces power management for RAID making it possible to process requests from the system while turning down part of disks and therefore reducing the power consumption of the HDD.

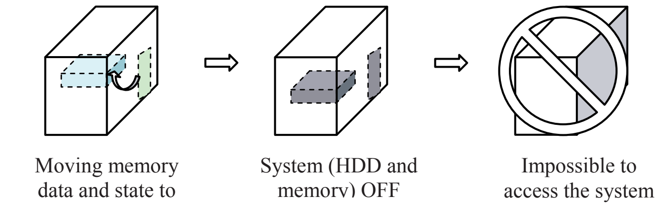
Recently, a new technology was introduced which enables the disk spin-down to last for a longer time by using non-volatile memory instead of disk to service I/O requests [12]. Called NV cache, it makes it possible to reduce disk activity to save HDD energy, and to ensure data reliability in spite of unexpected system shut downs. NV cache consists of read-cache and write-cache, and is similar to normal cache in terms of how it works; if the data exists in NV cache when a read request occurs, it will be serviced by the NV cache, otherwise it will be serviced by the disk. In which case, the hard disk should spin-up if in a spin-down state, that is, requires much power and latency. When a write request occurs during disk spin-down, it is simply redirected to NV cache, and disk spin-up would be required if the write buffer is full.

In this work, we explore the method of minimizing power consumption in the commodity server. There are two main constraints we aim to address in this study. First, the idea should be able to be realized on the existing server organization without any hardware modification. We do not introduce new hardware layer, e.g. NAND flash, SSD, or other types of large scale NVRAM. Second, users should be able to access the system even in the case of a power-down of the disk containing the OS. In addressing these constraints, we exploit characteristics of the modern server system: main memory is abundant resource and storage subsystem constitutes dominant fraction of power consumption. We propose to turn off the hard disk drive when server is idle. This idea is not new. More importantly, to keep on providing limited services, we dynamically migrate the frequently accessed objects at the storage system, e.g. objects in the root file system in UNIX, to memory. To maintain consistency in accessing objects independent of whether the object is in memory or in storage, we establish a block device on main memory and move the objects from the storage system to main memory.

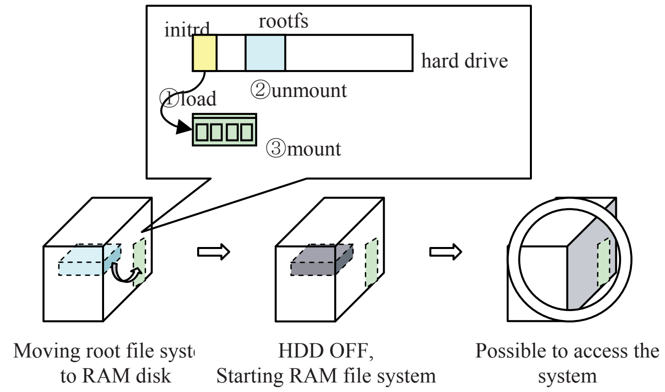
The following Chapter describes the concept of power management using file system migration utilizing a RAM disk. Chapter 3 depicts the autonomic scheduling for file system transition. Lastly this system is evaluated in Chapter 4.

II. FILE SYSTEM MIGRATION FOR ENERGY EFFICIENT STORAGE MANAGEMENT

This study aims to reduce power consumption of hard disks in system using HDD. We propose a method of transferring the storage state in order to conserve energy. This is achieved by turning the HDD off and starting the RAM disk during times of no disk access. We evaluate the possible reduction in power consumption by comparing this system with others that operate hard disks for 24 hours.


Moving memory data and state to disk partition
System (HDD and memory) OFF
Impossible to access the system

(a) Power management using hibernation


initrd
rootfs
hard drive
①load
②unmount
③mount
Moving root file syst to RAM disk
HDD OFF, Starting RAM file system
Possible to access the system

(b) Power management using variable file system

Figure 1. System access according to power management method

A. System overview

If a user turns a disk off for power saving in a single HDD system, user is therefore unable to access the system (see figure 1.(a)). It is the disk spin-down that offers a method to reduce power consumption without turning off the hard disk. Despite this, if there is no access to the hard disk for a long time, disk power-down becomes more effective for power management. We can save disk power and also keep the system active by moving the root file system from the HDD to RAM disk. This keeps the system running while the HDD is in power-down mode (see figure 1.(b)).

This paper proposes a method which moves the root file system to a RAM disk. This method provides a solution of a problem being unable to access the system when turning HDD off in order to reduce power consumption. This method consists of the following four components; (i) Power manager module: performs the processes preparing the system to turn the HDD off, move the real root file system to the RAM disk, and start the root file system on the RAM disk. (ii) Real root file system: a root file system operating on a hard disk. (iii) RAM root file system: a root file system operating on a RAM disk after HDD power-down. This can be created by copying the real root file system dynamically, or can be built in advance as a single tiny root file system such as initrd. and (iv) Kernel and each driver: same as with the general OS.

Figure 2 overviews the system structure. Figure 2.(a) depicts the internal flow of normal tasks and power management tasks (PM task) in the normal state (disk power-on state). Normal tasks are operated as processes of the root file system - same as with the general OS. When a PM task is executed, the hard disk is turned off after the following tasks are completed by the power manager

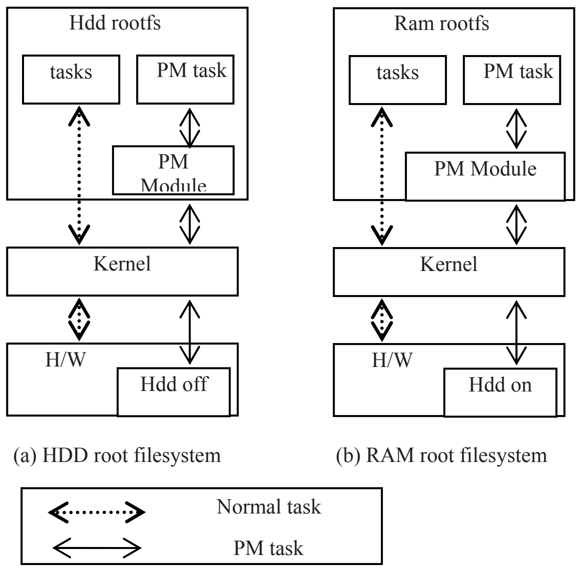


Figure 2. Overview of system structure

module (PM module) - unmounting each file system on the disk, moving the root file system to the RAM disk, and starting the RAM disk as root file system. Figure 2.(b) depicts the internal flow of normal and PM tasks in the RAM disk when in the power-off state.

B. Storage State Transition

Since the root file system is operating on a hard disk in a system based on a HDD, it is prohibited to turn off the hard disk while running the root file system. As such, it is not possible to transfer the storage state while the HDD is running. If the root file system is first copied to the RAM disk and then started, it is possible to turn off the hard disk. However, because the capacity of a root file system is generally large and that programs requiring disk access cannot be used after HDD power-off, it is logical to use Initrd (or, if not Linux, the RAM disk image corresponding to the particular OS) rather than to copy the whole real root file system. Most Linux distributions provide a single generic kernel image for booting in a variety of H/W. Device drivers of the kernel image consist of loadable kernel modules. The kernel image is minimized so it is possible to boot from limited capacity storage such as embedded systems. The required module in the root file system should be detected and loaded in advance for mounting the root file system at boot.

To achieve this, code is not added to the kernel but the temporary root file system, which is Initrd, is utilized at the initial boot stage. It detects the H/W, loads modules, and discovers the devices necessary to mount the real root file system [13]. The Initrd image forms a single tiny integrated root file system for those jobs, and many embedded systems employ it as a real root file system. We can place an Initrd image within a specific partition in a hard disk, e.g. boot partition, or flash memory if a system provides it.

C. File System Migration

In preparation for hard disk power-off for power saving, it is necessary to first copy the Initrd image to the RAM disk. In legacy UNIX Operating System, after a boot loader, e.g. GRUB in Linux, copies Initrd and the kernel image to memory the following tasks are performed; the kernel is invoked, several kernel initializing tasks are performed, and init process is executed. Since this study does not consider a system boot stage, it is only necessary to mount the RAM disk area loaded to memory. Specifically, this entails the preparation of a namespace to mount a device file system like /dev/ram/, and checks the magic number to determine a format of the image (i.e. if it is ext2, gzip or romfs etc). Also, it allocates the area for the RAM disk, and uncompresses and loads RAM disk image to memory after CRC checks. At this point the Initrd image becomes a mountable block device.

Next, some file systems in the HDD need to be unmounted. Since it is not possible to unmount if there are processes running in the current hard disk, all running processes should be terminated. This can be performed by a batch file like rcDown in Linux. rcDown performs the tasks for

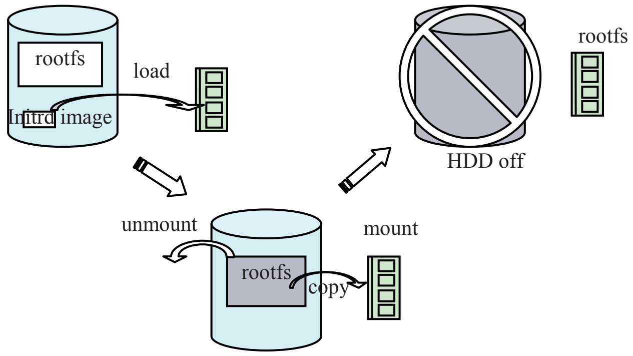


Figure 3. Process which moves the root file system from HDD to RAM disk

system shutdown in batch processing; terminating system processes, user processes and daemons, and unmounting each file system.

Furthermore, the user should be given the choice whether to force termination or terminate after the process is finished. When the unmount process completes, the system unmounts each file system on the disk, and also swaps off after unmounting proc file system. The root file system which has been written to RAM should then be mounted to make a new root file system. The block device can be mounted as root file system directly, or by changing Initrd to a root file system using a program such as pivot_root or chroot after the RAM disk has been mounted to a certain point (e.g. /mnt/ram/).

Between mounting the RAM disk and changing it to a root file system, if configuration files are being copied in the real root file system to corresponding directories in the RAM root file system, major user and system tasks can be performed (excluding those requiring HDD access). This can be achieved through a scheduling program like cron periodically. Of course, necessary tasks may be performed by executing commands directly in sleep state (i.e. HDD power-off state). Figure 3 shows a series of processes which power-off the hard disk after moving the root file system from the HDD to a RAM disk.

III. AUTONOMIC SCHEDULING OF FILE SYSTEM MIGRATION

The point of time for switching hard disk states can be determined by two ways, that is, static state transition and dynamic state transition. Static state transition is performed according to user settings, where a user sets the times for HDD power-off (and starting of the RAM disk) and power-on of the HDD (and the resumption of the actual root file system). Monitoring the system clock periodically, if the time/date a user sets coincides with that of the system, the corresponding command will be executed. In this paper, the monitoring is performed as a default every minute. This can be realized by a scheduling program like cron.

Also, a user may switch the disk state immediately by directly running a command. Dynamic state transition is performed according to current system activity; based on monitoring periodically whether users are using the system or not, file system will be migrated automatically if no one uses the system. The system's activities monitored in this paper are User activity and System resource activity.

The user information of the kernel can be used to determine user activity. The kernel periodically puts the information for current and other users of the system in specific locations as binary files (i.e. /var/run/utmp etc). It is possible to get this information by commands such as *w* and *who*. The required information is the user IDs of those connected to the current system and the idle time for each user. Considering this information periodically, if the idle time for all users exceeds a certain value, we regard the user activity as inactive. This means that there is no input (of keyboards and mouse) for all users including root during the certain term. If tasks executed by a user (for example, copying mass files, playing multimedia files, etc.) are running without user input, it is not desirable to regard the system as inactive. Therefore it is necessary to examine the activity of each system resource as well. The CPU activity can be monitored, and state information of the resource can be obtained by monitoring the corresponding directory (i.e. /proc/stat) in the proc file system. Alternatively, it may be more convenient to monitor the activity state for each resource by using system monitoring tools such `as top`, `collectl`, `sar`, and so on.

It would be better to set a period of examining the CPU activity more shortly than of examining the user activity, because usage of the system resource is fluctuated much more frequently and significantly than rate of the user connection is. The CPU is considered to be in an idle state when its usage is less than a minimum value. This represents the CPU's maximum usage for performing tasks when in the idle state. This may be determined by the user or at system design. The CPU activity can be monitored periodically and if CPU idle state exceeds a certain term, the system is regarded as inactive. Table 1 shows the algorithm used to determine the point of time for switching the hard disk state automatically. The certain term of system inactivity necessary for system transition is defined as a default 45 minutes.

TABLE 1. ALGORITHM FOR MONITORING THE SYSTEM RESOURCE ACTIVITY AND THE USER ACTIVITY STATE

```
while (TRUE)
{
/* watching system resource activity */
    for (user activity period/PERIOD) { /* user activity period is 1 min. so looping 12 times */
        Reading the current state information of CPU
        if (CPU usage < CPU_IDLE_LIMIT) { /* CPU_IDLE_LIMIT is 5 by default */
            COUNT++
            if (COUNT*PERIOD > INACT_TIME) { /* PERIOD is 5 sec. by default */
                system_inact=1 /* CPU is inactive */
                break
            } else
                system_inact=0
        } else
            COUNT=0
        sleep PERIOD
    }
/* watching user activity */
    Reading the user IDs being connected to current system and the idle time for each user
    if (Idle time for all user IDs > INACT_TIME)
        user_inact=1 /* All users are inactive */
    else
        user_inact=0

    if (system_inact && user_inact)
        return /* State transition of hard disk */
}
```

IV. PERFORMANCE EVALUATION

This Chapter evaluates the reduction in power consumption by using the power management system, compared with the system operating the hard disk 24 hours a day. The experiments cover two scenarios. In the first scenario, backup is run everyday, i.e. heavy disk workloads for a certain period of time, and the second scenario only runs default system tasks. In each scenario, we measure the power consumption for the system which transitions to sleep state for 6 hours everyday and for otherwise system.

A. Experimental environments

This work has been implemented to the Kurobox/pro [14] and the power consumption of the system itself has been measured by using a power measurement tool [15]. For convenience, we denote the system using power management as PMS (Power Management System), and the normal system as NS (Normal System). Both systems each use a 3.5 inch, 7200 rpm, SATA II, 250 GB hard disk.

B. Power consumption when running only default system tasks

Figure 4 shows the power consumption for the PMS and NS over a week duration. The PMS is set to be in the sleep state from 12:00 AM to 6:00 AM everyday. No user tasks runs in either system.

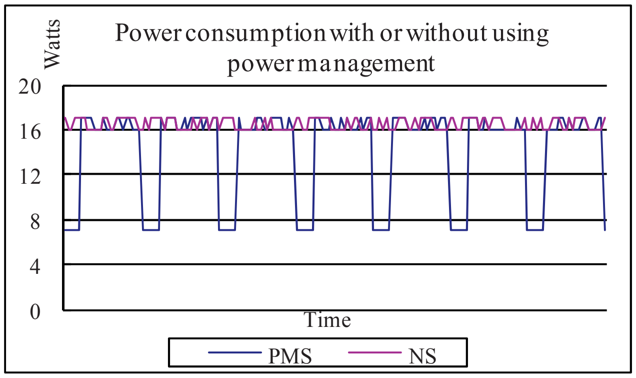
As can be observed in Figure 4, while in the sleep state, the system consumes only 41.2~43.7% of the power of the normal system. Using these results, the average power consumption rate for the PMS can be determined. If the power consumption rate of NS is equal to 100, the average Power Consumption Rate (PCR) of the PMS for a day is given by the following formula.

$$PCR_{PMS} = \frac{24-t}{24} \cdot 100 + \frac{t}{24} \cdot 42.5 \qquad (1)$$

where t is the time per day (in hours) spent in the sleep state. In these experiments t = 6, therefore

$$PCR_{PMS} = \frac{24-6}{24} \cdot 100 + \frac{6}{24} \cdot 42.5 = 85.6$$

These results demonstrate that when compared with the NS m, the PMS saves an average of 14.4% of power a day.



(a) time series comparison (for a week)

	Normal mode	Sleep mode
Voltage RMS value	100.0 Vrms	100.0 Vrms
Current RMS value	0.31-0.34 Arms	0.16 Arms
Effective power	16-17 W	7 W
Apparent power	32-34 VA	16 VA
Reactive power	27-29 var	14 var
Load power factor	0.51-0.52	-
Load crest factor	3.77-4.02	-

(b) numerical comparisons

Figure 4. Power consumption with or without power management

C. Power consumption when running backup tasks daily

The backup for both systems was scheduled at 1:00 AM daily. The backup size is 10 GB consisting of more than 50,000 files. Aside from the backup tasks, other experimental configurations are the same as in the previous section. The PMS transitioned to the sleep state at 12:00 AM, and returned to the normal state at 12:45 AM for daily back up at 1:00 AM. Then, once the backup had finished (around 2:30 AM) it transitioned to the sleep state again. The NS also performs backup staying in the normal state from 1:00 AM everyday.

Figure 5 shows the power consumption when running backup. The usage is 100~105.9% of the power consumption of the normal state without backup. As discussed in the previous section the average power consumption rate for a day using the PMS (with running backup) can be calculated. If the power consumption rate of NS is set to 100, then

$$PCR_{PMS} = \frac{n}{24} \cdot 100 + \frac{s}{24} \cdot 42.5 + \frac{b}{24} \cdot 103 \qquad (2)$$

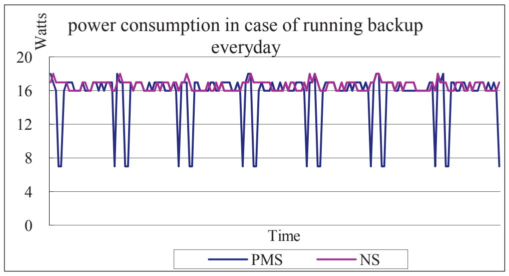
where n is the time spent in normal state with no heavy disk workload, s is the time in sleep state, and b is the time spent running the daily backup, and each unit is in hours. As n = 15.5, s = 6, and b = 2.5 from the results of experiments, then

$$PCR_{PMS} = \frac{18}{24} \cdot 100 + \frac{3.5}{24} \cdot 42.5 + \frac{2.5}{24} \cdot 103 = 91.9$$

Therefore, using the PMS while running backup saves an average 8.1% of power daily compared to the NS.

D. FS Migration and other power saving schemes

FS Migration is the scheme for users to be able to use



(a) time series comparison (for a week)

	Backup running
Voltage RMS value	100.0 Vrms
Current RMS value	0.31-0.35 Arms
Effective power	16-18 W
Apparent power	32-35 VA
Reactive power	27-30 var
Load power factor	0.51-0.53
Load crest factor	3.77-4.32

(b) power consumption when running backup

Figure 5. Power consumption with or without using power management when running backup everyday

the system in spite of disk power-down other than existing standby or hibernation, with saving the energy consumption due to disk power-off. Available services after FS Migration are as follows:

- ✓ Each device related services besides hard disk (Network, USB, PCMCIA, etc), so that USB Portable Storage also available
- ✓ Basic system utilities (mount, cp, rm, ls, ln, ps, etc)
- ✓ Network utilities (telnet, ping, wget, netstat, etc)
- ✓ Editor utilities (vi, etc)
- ✓ Shell programming (bash, sh, etc)
- ✓ Compress programs (gzip, gunzip, tar, etc)
- ✓ Other Core Utility programs (file utilities, text utilities, shell utilities, etc)

Table 2.(a) compares the power on/off state in each hardware component for standby, hibernation, shutdown, FS Migration each. Also, table 2.(b) compares the service availability, boot up or resume time, and energy saving for each of them.

V. CONCLUSION

In systems using multiple disks, e.g. storage servers, significant energy savings can be achieved by reducing the power consumption of hard disks. This paper proposes a power management system which saves system energy by turning off disks and starting a RAM disk while the system is inactive. The achievable reduction in power consumption is evaluated compared with a system operating a hard disk all day long.

The power management system makes it possible to perform major tasks regularly by using a scheduling

TABLE 2. COMPARISON FOR EACH POWER MANAGEMENT SCHEME

	standby	hibernation	shutdown	FS Migration
CPU	Off	Off	Off	On
Memory	On	Off	Off	On
HDD	Off	Off	Off	Off
NIC	Off	Off	Off	On
USB	Off	Off	Off	On

(a) Power on/off state

	standby	hibernation	shutdown	FS Migration
Service	Unavailable	Unavailable	Unavailable	Available
Boot up time	Short	Normal	Long	Normal
Power saving	Good	Very good	Very good	Normal

(b) Service availability, boot up or resume time, and energy saving

program even after entering the sleep state. The user can also perform necessary tasks by interacting with the shell directly. These characteristics make it distinctly different from the sleep state of existing power management systems.

The experimental performance evaluation demonstrates a saving of around 14.4% of power consumption when using power management for 6 hours a day in a one HDD system. Power reductions of around 8.1% a day can achieved even when running backup daily. Moreover, we can see that the backup management enables transition back to the sleep state after waking up at the scheduled time to perform backup. This occurs even if the backup task is scheduled while in the sleep state.

ACKNOWLEDGMENT

This work is based on the project of developing the NAS (Network Attached Storage) firmware in Buffalo, Inc. We would like to thank the NAS Development Group in Buffalo, Inc. Also, This work is funded by Center for Large Scale Hyper-MLC SSD Technology Development(Grant No. 1003520) at Hanyang University by KEIT and Ministry of Knowledge and Economy of Korea.

REFERENCES

[1] (2009). *ENERGY STAR* Version 5.0 System Implementation*. Available: http://www.intel.com/Assets/PDF/whitepaper/321556.pdf

[2] E. Otoo, *et al.*, "Analysis of trade-off between power saving and response time in disk storage systems," *IPDPS,* 2009.

[3] K. Greenan, *et al.*, "A Spin-Up Saved is Energy Earned: Achieving Power-Efficient, Erasure-Coded Storage," HotDep08, 2008.

[4] R. Garg, *et al.*, "Markov Model Based Disk Power Management for Data Intensive Workloads," in *CCGRID*, 2009, pp. 76-83.

[5] C. Weddle, *et al.*, "PARAID: A gear-shifting power-aware RAID," *ACM Transactions on Storage (TOS),* vol. 3, p. 13, 2007.

[6] X. Li, *et al.*, "Performance directed energy management for main memory and disks," *ACM Transactions on Storage (TOS),* vol. 1, p. 380, 2005.

[7] Q. Zhu, *et al.*, "Reducing energy consumption of disk storage using power-aware cache management," *HPCA,* 2004.

[8] M. Poess and R. Nambiar, "Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results," *Proceedings of the VLDB Endowment,* vol. 1, pp. 1229-1240, 2008.

[9] T. Bisson and S. Brandt, "Adaptive disk spin-down algorithms in practice," 2004.

[10] L. Ganesh, *et al.*, "Optimizing Power Consumption in Large Scale Storage Systems," *In Proceedings of the 11th Workshop on Hot Topics in Operating Systems (HotOS XI),* 2007.

[11] E. Pinheiro, *et al.*, "Exploiting redundancy to conserve energy in storage systems," *ACM SIGMETRICS Performance Evaluation Review,* vol. 34, p. 26, 2006.

[12] (2007). *Overcoming disk drive access bottlenecks with Intel Turbo Memory*. Available: http://download.intel.com/design/flash/nand/turbomemory/article.pdf

[13] W. Almesberger, "Booting linux: The history and the future," in *Proceedings of the Ottawa Linux Symposium*, 2000.

[14] *KUROBOX/PRO Product Specifications*. Available: http://downloads.buffalo.nas-central.org/KBPro_ARM9/GPL/English-Documentation/Product%20Specifications.pdf

[15] Available: http://www.elgar.com/products/EC1000S/downloads/EC1000S_User_Manual_4994-974.pdf