Potentiality of Power Management on Database Systems with Power Saving Function of Disk Drives

Norifumi Nishikawa[1,2], Miyuki Nakano[1], and Masaru Kitsuregawa[1]
[1]Institute of Industrial Science, the University of Tokyo
4-6-1 Komaba Meguro-ku, Tokyo 153-8505, Japan
[2]Systems Development Laboratory, Hitachi, Ltd.
292 Yoshida-cho, Totsuka-ku, Yokohama 244-0817, Japan

{norifumi,miyuki,kitsure}@tki.iis.u-tokyo.ac.jp, norifumi.nishikawa.mn@hitachi.com

Abstract

Power consumption of modern datacenters is increasing rapidly. Databases, especially OLTPs, have become a major application at datacenters. Therefore, power-saving management for OLTP applications has become an important task for user budgets and datacenter operations. A recent report described that disk drives consume about 70% of total power of IT equipment when such large OLTP applications are executed. As described in this paper, we specifically examine a novel power-saving management for multiple disk drives of OLTP applications. First, we constructed an experimental system with power meter, and measured basic power consumption and I/O behaviors of OLTP applications. Then we show that basic measurement results confirm there is still the potentiality of power-saving even though OLTP applications are running. We propose a new method that delays database writes based on this I/O behavior knowledge at run time. Experimental and simulation results obtained using our power-saving methods are explained. Our method provides dynamic power saving of disk drives running TPC–C applications, which issue many I/Os to the disk drives.

Keywords: Online transaction processing, OLTP, Disk drive, Power-saving

1 Introduction

Digital data produced by human beings are increasing every day. "*How Much Information?*", a report published by the University of California, San Diego states that the amount of digital data will be measurable in Yotta-Bytes (YB) worldwide by 2025 (Bohn, Short 2009). This explosion of digital data must be managed and used by data-intensive applications such as customer management systems and online transaction processing systems (OLTP). The rapid growth of digital data necessitates that these applications be readily extensible and that they provide high performance in processing digital data.

Now, these large data-intensive applications run at datacenters. The total management cost of datacenters increases annually. Especially, the cost for power and cooling has increased drastically. An IDC report (Eastwood, Bozman, Pucciarelli, and Perry 2009) describes that power and cooling costs were only 10% of hardware costs in 1997, but these costs will account for 75% of hardware costs in 2011.

IT equipment of datacenters includes servers, storage, and networks. Although the power consumption of storage is only approximately 30% of that of IT equipment (Rajecki 2008), our investigation specifically examines power saving for storage because main applications of datacenters are data-intensive applications which require many more disk drive spindles than other applications do. One report of the literature (Poess, and Nambiar 2008) describes that the power of disk drives for large online transaction processing systems (OLTPs, data-intensive applications) accounts for approximately 70% of the total power of IT equipment. Consequently, disk drive power-saving is important for maintaining good low power consumption of datacenters.

Many storage and disk drive power consumption methods have been proposed to date. Typical methods control the timing of the transition of the power status of disks or control the rotation speed of disks (Douglis, Krishnan, and Bershad 1995, Helmbold, Long, Sconyers, and Sherrod 2000, Gurumurthi, Sivasubramaniam, Kandemir, and Franke 2003, Zhu, Chen, Tan, Zhou, Keeton, and Wilkes 2005, Gniady, Hu, and Lu 2004, Son, Kandemi, and Choudhary 2005), extend the duration of the idle period to use a disk's power-saving functions by controlling the I/O interval (Papathanasiou, and Scott 2004, Li, and Wang 2004, Yao, and Wang 2006, Heath, Pinheiro, Hom, Kremer, and Bianchini 2002), and extend the idle period by replacing data to disks (Colarelli and Grunwald 2002, Li and Wang 2004, Pinheiro and Bianchini 2004, Weddle, Oldham, Qian, and Wang 2007, Verma, Koller, Useche, Rangaswami 2010). However, these methods are implemented inside storage, or use only storage access statistics gathered at storage.

Recently, other power-saving approaches based on applications such as DBMS have been proposed along with hardware-based approaches for storage and servers. One report (Harizopoulos, Shah, Meza, and Ranganathan 2009) presented a discussion of DBMS and its system configurations, which enable reduced power consumption of IT equipment based on experiments using a TPC–H database. An earlier study (Poess, Nambiar 2010) described the relation between performance and power consumption when varying the system configuration of a TPC–H database. Results obtained using these approaches suggest that the utilization of application knowledge offers power-saving potential for use in storage systems.

The application workload, however, changes according to the number of users and the data size; an adaptive power

consumption mechanism must therefore follow the dynamic variation of the application workload. A previous report (Poess and Nambia 2010) presented one answer to datacenter system design and construction problems, but no solution can be applied to a datacenter that is already in service. Another discussion (Harizopoulos, Shah, Meza, and Ranganathan 2009) of the power-saving approach of DBMS includes no quantitative evaluation of the effect of DBMS oriented power-saving approach and its side effects on DBMS performance.

As described in this paper, we investigate the potential of power saving of multiple disk drives used for OLTP application. Because OLTP applications constitute a major workload at datacenters, power management of OLTP application is expected to achieve great power savings at datacenters. However power management of OLTP is unlikely to be realized because OLTP is a high-I/O-load application of DBMS. It is quite difficult to realize power saving of OLTPs at run time. We have already reported preliminary experience results of power saving of TPC–C with two disk drives (Nishikawa, Nakano, and Kitsuregawa 2010).

The contribution of this paper is (i) to clarify the potential of run time power management of OLTP application based on their I/O behaviors, (ii) to propose a method that delays database writes based on this I/O behavior knowledge at run time, and (iii) to measure the actual power consumption with running TPC–C benchmark. Then we show that dynamic power saving of DBMS can be achieved without performance degradation.

Section 2 presents related works. Section 3 presents characteristics of disk drive power consumption. Section 4 presents a description of I/O behavior of TPC–C application on a small system configuration using one disk drive. Section 5 presents a proposed power-saving method using I/O behavior characteristics of TPC–C application. Section 6 explains the examination results; conclusions are presented in section 7.

2 Related Works

In this section, we describe works related to storage and the DBMS-based power-saving method. Power-saving methods for storage have been studied at storage fields. Therefore we first summarize storage- based related works. Then we discuss DBMS-based related works.

2.1 Storage-based Power Saving Method

Storage-based related works are categorized into those addressing disk drive rotation control, I/O interval control, and data placement control. Herein, these storage-based power-saving methods are discussed.

2.1.1 Disk Rotation Control

This approach controls the disk-drive rotation speed or power status. Proposed approaches of disk drive control are categorized into two groups: i) changing the length of the wait time to change the status of disk drive to standby or to sleep (Douglis, Krishnan, and Bershad 1995, Helmbold, Long, Sconyers, and Sherrod 2000); and ii) rotating a disk drive at multiple speeds, which specifically requires that the power consumption of a disk drive be lower when its rotation speed is low (Gurumurthi, Sivasubramaniam, Kandemir, and Franke 2003, Zhu, Chen, Tan, Zhou, Keeton, and Wilkes 2005).

Furthermore, another report in the literature (Zhu, Chen, Tan, Zhou, Keeton, and Wilkes 2005) proposes an I/O timing prediction method using a program counter at the OS level. Another report of a study (Gniady, Hu, and Lu 2004) proposes that compilers examine a scientific application source code and embed a disk control code to reduce unnecessary idle time.

Using disk power-saving functions and saving power requires an idle length sufficient to supplement the power loss during the power-on process of the disk drive (approximately 30 s). A typical OLTP processes tens to hundreds of transactions per second; the idle duration of disk drives is usually much shorter than 1 s. Applying these methods, which do not change the I/O timing of OLTP (a typical application of DBMS or DSS), is therefore difficult.

2.1.2 I/O Interval Control

This approach controls the I/O timing of an application to increase the probability that a disk drive is in a power-saving mode. A feature of this approach is to increase the idle period using hierarchical memory architecture such as cache memory (Papathanasiou and Scott 2004, Li and Wang 2004, Yao and Wang 2006), with varying application of I/O issue timing (Heath, Pinheiro, Hom, Kremer, and Bianchini 2002).

These simple cache control methods have difficulty increasing the idle period for I/O intensive application such as OLTP because many I/Os are issued to disks even if the cache hit rate is high. Consequently, it is difficult to apply this approach directly to data-intensive applications.

2.1.3 Data Placement Control

This approach is intended to reduce the disk-drive power consumption by controlling the data placement on disk drives. The concept of this approach is to concentrate frequently accessed data into a few disk drives, then to move the status of other disk drives to standby or to sleep mode (Colarelli and Grunwald 2002, Pinheiro and Bianchini 2004, Weddle, Oldham, Qian, and Wang 2007, Son, Chen, and Kandemir 2005). These approaches determine data placement based on the access frequency of blocks of storage. Recently, the data placement control function is implemented in the storage virtualization layer (Verma, Koller, Useche, Rangaswami 2010). This approach produces replicas of active data chunks into active RAID storage and cuts power applied to other inactive RAID storage.

The information used for these methods is obtained inside storage. Our approach, however, uses application information to calculate the data placement on storage. Applications (or their management software) know what data is accessed and its access timing. Therefore, we believe this application knowledge provides good hints for an appropriate data placement to reduce power consumption.

2.2 DBMS-based Power Saving Method

As described in the *Introduction*, DBMS-based power-saving methods are quite new. These approaches

demonstrate that utilization of application knowledge can support storage power saving. A few articles discuss the power efficiency datacenter system design and construction solutions (Harizopoulos, Shah, Meza, and Ranganathan 2009, Poess and Nambiar 2010).

However, no such proposal for power-saving methods is useful after the datacenter is in service. These methods have difficulty accommodating the workload variance of datacenters because such power-saving methods depend on hardware design.

3 Characteristics of Disk Drive Power Consumption

For developing the power-saving method for TPC–C, we first analyze disk drive power consumption characteristics. This section explains characteristics of disk-drive power consumption based on actual measurement results.

3.1 Measurement Environment

Fig. 1 presents an outline of equipment used in our power consumption measurement environment. A load-generating PC provides power to a measured disk drive using 4-pin power cables. The voltage of the red cable is 5 V; that of the yellow cable is 12 V. We connected a digital power meter (WT1600; Yokogawa Electric Corp.) to measure the electric current. We also measured voltages between the red cable and the black cable, and the yellow cable and black cable. The disk drive power consumption is their sum.

The load-generating PC CPUs are two Athlon 64 FX-74 3 GHz, 1 MB cache, 4-core processors (AMD, Advanced Micro Devices, Inc.). Main memory sizes of the load generating PCs are 8 GB. Measured disk drives are Barracuda ES ST3750640NS (750 GB, 7200 rpm; Seagate Technology LLC[*1]). The disk drive write caches are turned off to protect the database reliability because the DBMS uses no write cache.

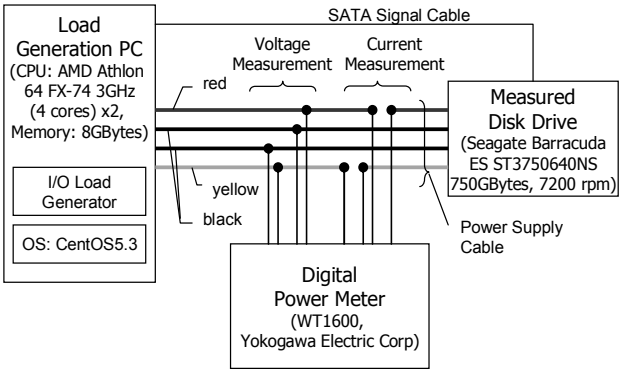

Fig. 1: Measurement Environment of Drive Disk Power Consumption.

3.2 Disk Drive Power Status

A disk drive we used for analyses has a power-saving function that changes the disk drive power status according to the following.

1. Active: I/Os are issued to the disk drive. The disk drive power consumption is highest.
2. Idle: No I/O is issued to the disk drive, but the disk drive serves I/Os immediately.
3. Standby: The disk drive reduces power to rotating disks by parking the head and stopping the disk drive rotation.
4. Sleep: The disk drive parks the head and stops the disk drive rotation. In this status, the disk drive also stops the power supply to the cache. Power consumption is the same as that in standby status, but a disk drive reset is necessary to change the status to active or idle.

We use the disk drive's active, idle, and standby statuses because the power consumption of sleep status is the same, but it requires an additional operation to change the disk drive status to active or idle states.

3.3 Power Consumption at Active/Idle States

Fig. 2 depicts a relation between the disk drive power consumption and I/Os per second (IOPS). The I/O size is 16 KB, which indicates that the power consumption of random I/O increases in accordance with an increase of IOPS, but it saturates the increase of power where IOPS is larger than 70–80 IOPS.

The power consumption of a sequential write is much less than that of random I/O because the disk drive head movements are far fewer than those of random I/O.

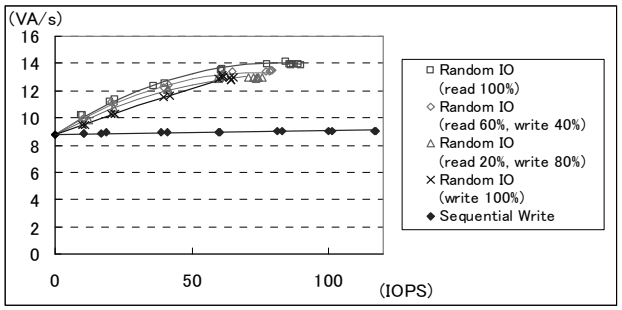

Fig. 2: IOPS and Power Consumption.

In Fig. 2, maximum values of IOPS of random I/O are shown to differ by the ratio of read and write percentages of I/Os: maximum random IOPS of 100% read I/O is about 90 IOPS; 60% read and 40% write I/O is about 80 IOPS; 20% read and 80% write I/O is about 70 IOPS; and 100% write I/O is about 60 IOPS. The maximum IOPS of the sequential write I/O is about 120 IOPS. Fig. 2 shows only 16 KB I/O size, but the trends of IOPS and power consumption of other I/O size (4 KB, 8 KB, 32 KB, 64 KB, and 128 KB) resemble the trend shown for 16 KB I/O.

3.4 Power Consumption of Standby Status and Break-even Time

Fig. 3 portrays the power consumption of a standby disk drive, along with the transition from active/idle status to standby status, and migration from standby status to active/idle status. Fig. 3 also shows the disk drive break-even time. The break-even time is the interval during which the reduced energy is equal to the added energy of spinning up disk drives.

Figure 3 also shows that the power consumption of a standby state is 1.5 VA/s, which is less than one-fifth of

*1: Seagate is a registered trademark of Seagate Technology LCC.

the power consumption of the idle status. The transition status from standby to an active/idle, however, requires 8 s and more than an average of 23.3 VA/s. The transition status from an active/idle to a standby requires 3.5 VA/s with 0.2 s. The break-even time calculated from these values is 23.9 s. Therefore, the idle time of more than 24 s is necessary to use this disk drive power-saving function.

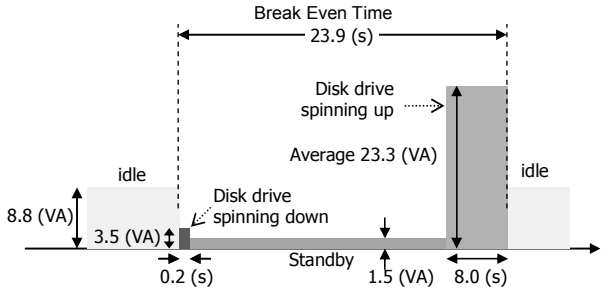

Fig. 3: Power Consumption of Standby Status and Break-Even Time.

4 I/O Behavior of OLTP Application

For developing a power-saving method using OLTP knowledge, we then measured the I/O behavior of tpcc-mysql (tpcc-mysql 2004) using our test bed environment. Here, tpcc-mysql is a simple implementation of the TPC–C benchmark.

4.1 Experimental Environment

The hardware is the same configuration presented in Fig. 1 of the previous section. The software configuration is the following: The OS is 32-bit version of CentOS; the DBMS is MySQL Communication Server 5.1.40 for Linux; and the OLTP application is tpcc-mysql. The file system cache and the disk drive are disabled. The DBMS buffer size is 2 GBytes maximum. Our first target is high-transaction throughput OLTP applications served at large datacenters. Therefore, we configured the DBMS buffer size as larger than the database size.

Our experimental environment for measuring I/O behavior and power consumption is a small one with only a few disk drives. The number of I/Os of disk drives is, however, several decades per second and is sufficiently high for a disk drive used by DBMS. A large RAID storage can serve more transactions than a system using a few disk drives. However, the high transaction throughput is achieved by striping I/Os to multiple disk drives, and the number of I/Os to one disk drive is the same as or fewer than that of the small environment. The disk drives' I/O load of our experimental environment is sufficiently high; achieving power saving of disk drives contributes to the power saving of the datacenter.

The database of TPC–C applications has nine tables named Customer, District, History, Item, NewOrders, OrderLine, Orders, Stock, and Warehouse. One or more indexes are defined against these tables. For this discussion, we call data of the table and indexes by the name of the table, e.g. we designate data of Customer tables and indexes as "Customer". The DBMS also has log data for recovering the database. We designate the log data as "Log" in this report. The database is approximately 1 GByte (number of Warehouses is 10), in which the Log data size is not included. We partition a disk into 10 volumes and format these volumes using the Ext2 file system. Log data and each file of the tables and indexes are placed separately into each volume. This file placement eases the receipt of I/O performance data of each database data (tables and indexes): we simply measure the OS level performance.

For measurements, we created a database, loaded data into the database, and restarted the DBMS to clarify the DBMS buffer. Then we started tpcc-mysql and measured the I/O behavior for 10 min after the throughput of DBMS was stable.

4.2 I/O Behavior Characteristics of TPC–C

Figure 4 shows the number of reads and writes per second and the average I/O intervals of each datum. As presented there, the characteristics of TPC–C I/Os were the following: i) write I/Os to Log data were dominant, ii) I/Os to tables and indexes were fewer than or equal to two I/Os per second, and iii) more than half these I/Os were writes. No I/O was measured to District, Item, or Warehouse data. The I/O intervals of these data (Log, tables and indexes) were shorter than the break-even time (23.9 s), except for NewOrder data.

Figure 5 portrays the intervals of I/Os of data. This figure reveals that the access intervals of OrderLine, Orders, and Stock data were skewed; access intervals were longer than the break-even time. Therefore, a power-saving method using I/O behavior characteristics enables stoppage of disk drives for a long time.

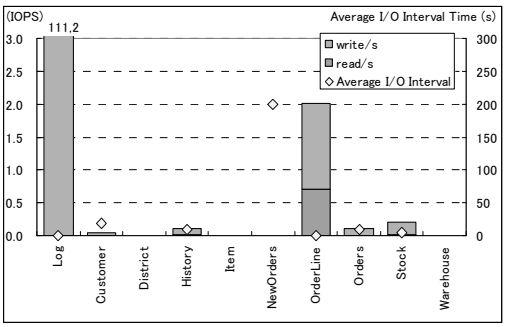

Fig. 4: Power Consumption of Standby Status and Break-even Time.

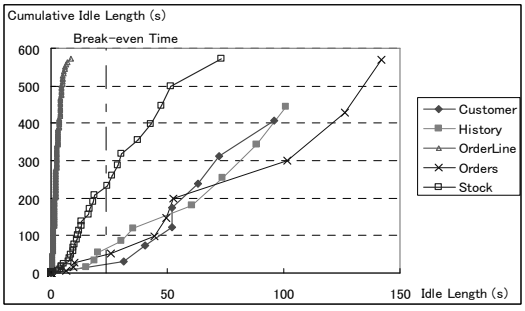

Fig. 5: Distribution of Access Intervals of TPC–C.

5 Power-Saving Method using I/O Behavior Characteristics of OLTP Application

This section presents a description of our power-saving method. The feature of our approach is to use

characteristics of I/O behavior of a TPC–C application. As Fig. 3 shows, writing Log data is a dominant task of TPC–C I/O, the remaining I/Os issued to database are also write dominant.

We propose a power-saving method using characteristics of the I/O behaviors of TPC–C application, especially the DB write operation behavior of DBMS. Write operations from the DBMS buffer to disk drives are done asynchronously with processing of transactions. Our method delays the asynchronous write I/Os to disk drives to generate idle intervals. Then our method is designed to use the power-saving function of disk drives.

The delay of time is determined dynamically according to the number of dirty pages in a DBMS buffer. That number is changed by the amount of transaction requests per second. This write delay causes little degradation of the transaction throughput because transactions write updated data only into the DBMS buffer and committed before the updated data are written to disk drives.

The features of the proposed method are: i) to generate inactive disk drives by gathering data of a few I/Os into the same drives, ii) to delay writing I/Os to database data until the database data are read on the same disk drives

5.1 Power-Saving Method using Data Placement

As portrayed in Fig. 4, the access frequency of each datum apparently differs. Writes to Log data are dominant, and tables and indexes are less frequently accessed. For the table and index data, the access frequency of OrderLine is high, and Orders and Stocks are follows. No I/O is issued at District, Item, or Warehouse. This method gathers frequently accessed data to a few disk drives, and then gathers infrequently accessed data into other disk drives to locate disk drives that can be stopped. This method reduces the disk-drive power consumption when the access frequency of data is low. However, from Fig. 4, we can observe long I/O intervals for Orders, History, Customer, and Stock data.

We expect that this long I/O interval will enable us to use the power-saving function of disk drives aggressively. This prediction of long I/O intervals cannot be achieved solely in consideration of the I/O disk drive frequency of storage-level knowledge.

Basically, we determine data placement from statistics of Idle length of applications' files. However, each application has runtime requirements specified by applications' nature. For TPC-C, the delay of transaction response may not be allowed. The transaction response time is affected by a read I/O. Therefore, we consider immediately read data as active data. Data that are placed in active disks is different in each application. Thus, we evaluate this method by using some patterns of data placement.

The I/O pattern of data might change after the system runs because the amounts of data or number of users are increased. Previous work based on access frequency blocks therefore required the calculation of data placement. The dynamic data replacement described herein is a future mode of work for storage power saving.

5.2 Power-Saving Method using Delayed Write I/Os

We propose a delayed write I/O method to use long I/O intervals for power-saving of disk drives. This method is based on the DBMS behavior of writing operations.

A DBMS writes database data in the following cases: i) a checkpoint write writes-back dirty DBMS buffer pages to disk drives, and ii) a dirty page flush is issued to acquire a free page into the DBMS buffer. The dirty page flush does not occur when the DBMS buffer has sufficient free space. A checkpoint write is asynchronous to query the execution of DBMS; DBMS enables timing control. We use an asynchronous checkpoint write to hold sufficient time to stop the disk-drive spindle.

Figure 6 presents our proposed method. Our approach produces a long idle period by delaying write I/O database data until the DBMS reads database data on the same disk drive. As shown there, our approach causes no delay of query processing threads. We can spin down the disk drive at this long idle period without degradation of OLTP throughput. The read I/Os are issued to disk drives synchronously because a delay of the read I/O causes a delay of query execution.

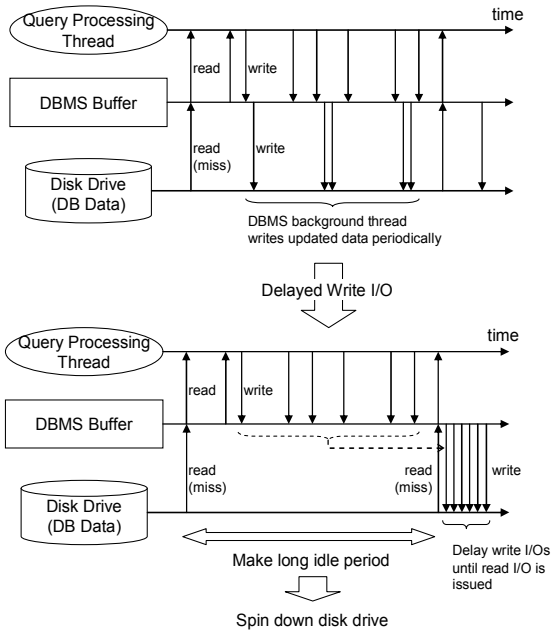

Fig. 6: Delayed Write I/O.

The I/O delay of the proposed method is the following:

1. Step 1. Keep write I/O into memory irrespective of the disk drive power status.
2. Step 2. Read I/Os of DBMS are executed immediately. Write I/Os delayed more than x min are written to the disk drive after a read I/O request. Here, $x = (1 - i/f)C$, where i denotes the number of Write I/Os to disk drives, f is a flush rate to disk drives, and C signifies interval length between two check points. This step also ensures that the number of write I/Os does not exceed 60 I/Os per second. The value of 60 I/Os per second is a maximum number of random write I/Os of disk drives (see section 3.3).

3. Step 3. If write I/Os are delayed more than the interval of DBMS checkpoint, then they must be written to a disk drive. The maximum throughput is 60 I/Os per second, as in step 2.

Our proposed method applies write delay to the DBMS buffer. The delay length is determined dynamically according to the number of dirty pages in the DBMS buffer. The write is issued asynchronously with transaction processing. It is assumed that little impact is given to the transaction response because i) this delayed write method is applied only to inactive disk drives, and ii) there are few data read operations to the inactive disk drives. Consequently, the data flush to these inactive disk drives has little impact to the data read operations issued by transactions. Therefore, these methods are useful for power saving of datacenters after the datacenters are run.

6 Evaluation

We evaluated our power-saving method. First, we present results of our method obtained using a small database on two disk drives. Then we present results from a larger database on five disk drives

6.1 Evaluation of Two Disk Drives

We evaluated our data placement method based on access frequency using two disk drives. First, we show the actual measurement with a spin down function, which is provided by the Seagate disk drive. Then we evaluated our write I/O delay method using the data placement method during a simulation.

6.1.1 Evaluation of Data Placement Method Based on Access Frequency

6.1.1.1 System Configuration

Figure 7 presents the system configuration used for this evaluation; it is based on the configuration described in sections 3 and 4. We add a SATA disk drive to the configuration (Barracuda ES ST3750640NS; Seagate Technology LLC) and put database data and Log data into two SATA disk drives. The added disk drive is the same model as those described in sections 3 and 4 (Seagate). We connected a digital power meter to the added drive and measured the disk drive power consumption.

The DBMS and DB configurations are as described in section 4. Disk drives are configured to transition to the standby status when the idle period (x) is longer than 5 s, and move to active state when an I/O arrives. We use the minimum standby timeout length (5 s) that is selectable at the operating system. For evaluation, we started tpcc-mysql; then we measured the disk-drive power consumption and transaction throughput for 10 min after the transaction throughput was stable.

In Fig. 5, we can observe that almost all idle lengths of Customer, History, and Orders data are longer than the break-even time. Therefore, once the data are accessed, the probability of the next access to the data within a short interval length is low. Consequently, the disk drive should be stopped immediately after the disk drive is accessed. Therefore, we use the minimum standby timeout value (5 s) of the operating system.

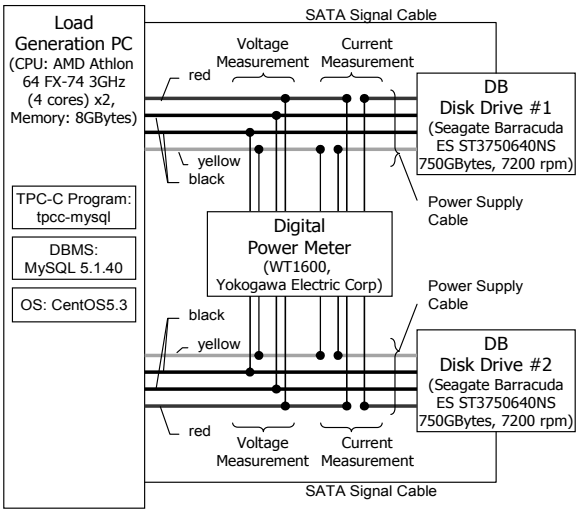

Fig. 7: System Configuration. (Two Disk Drives)

6.1.1.2 Data Placement Variation

We assessed four data placements of two disk drives shown in Table 1 to verify data placement method effects.

As described section 5, we use some patterns for data placement. The disk drives are of two types: active and non-active. Using this approach, it is likely that non-active disk drives will be put into standby status when no I/O is issued. It is also likely that the standby status will be maintained for a long time. Table 1 shows that disk drive #1 is active and disk drive #2 is not active. We put active data into disk drive #1 and non-active data into disk drive #2. Numbers of I/Os to the disk drive #2 are reduced in accordance with the increase of the case number.

Case	Data on Disk Drive #1	Data on Disk Drive #2
Case #1	Log	Customer, District, History, Item, NewOrders, Orders, OrderLine, Stock, Warehouse
Case #2	Log, OrderLine	Customer, District, History, Item, NewOrders, Orders, Stock, Warehouse
Case #3	Log, NewOrders, OrderLine, Orders, Stock	Customer, District, History, Item, Warehouse
Case #4	Log, Customer, NewOrders, OrderLine, Orders, Stock	District, History, Item, Warehouse

Table 1: Data Placement

6.1.1.3 Evaluation Results of Data Placement Method

Figure 8 presents the actual power consumption of the two disk drives measured using the data placement, i.e., data are distributed into two disk drives in Table 1. Here, we call the results obtained using a spin down function of disk drive as “with power-saving”; the results which the spin down function are turned off are those “without power-saving”. The spin down function is applied only to the non-active disk drive (drive #2 in Table 1).

As shown there, the power consumption without power-saving is nearly equal to 18 VA for all cases. In contrast, the power consumption with the power-saving method differs greatly among the four cases. In case #1, the value is equal to the value without the power-saving method. Therefore, in case #1, the data placement method

does not reduce the disk-drive power consumption. In case #2, the power consumption is increased to 20.2 VA (+10.2% compared with the power consumption without power-saving method). In cases #3 and #4, however, the disk-drive power consumption values (16.6 VA and 14.0 VA respectively) are smaller than those without the power-saving method. In the most efficient case, case #4, the disk drive power consumption is about 23.1% less than that without the power-saving method.

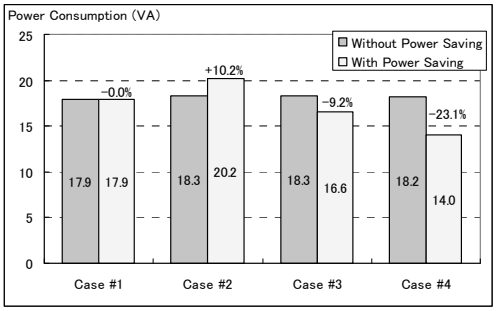

Fig. 8: Power Consumption (Measured Values; Two Disk Drives for DBMS).

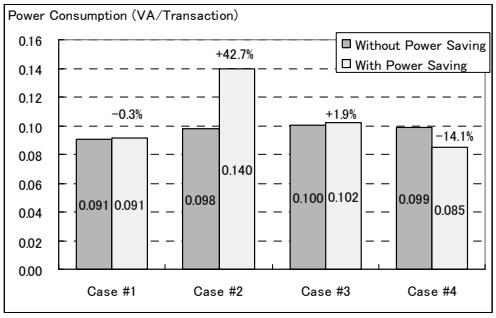

Fig. 9: Power Consumption per Transaction (Measured Values; Two Disk Drives for DBMS).

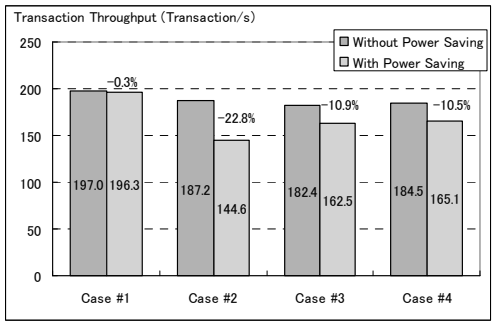

Fig. 10: Transaction Throughput (Measured Values; Two Disk Drives for DBMS).

Figure 9 presents the power consumption necessary for executing one transaction. The data are calculated by dividing the power consumption per second by the number of transactions executed per second. Data are normalized for each case by the power consumption without disk-drive power-saving. The power consumption per transaction of case #2 with power-saving is 0.140 VA/transaction, which is more than 40% higher than the value without power-saving. In case #3, the power consumption with power-saving is 0.102 VA/transaction, which is nearly equal to the power consumption without power-saving. In case #4, the power consumption per transaction with power-saving is 0.085 VA/transaction. Our approach therefore enables reduction of the power consumption by approximately 14%.

Figure 10 presents the transaction throughput of each case. In Fig. 10, the transaction throughput with power-saving of case #1 is 196.3 transactions per second, which is nearly equal to the transaction throughput without power-saving. In case #2, the transaction throughput with power-saving is 144.6 transactions per second, which is more than 22% lower than case #2 without power-saving. However, in cases #3 and #4 with power-saving, reduction of the transaction throughput maintains approximately 10% degradation.

From Figs. 8, 9, and 10, it is apparent that the considerable data placement can achieve large power saving. In case #1, the power-saving function does not reduce power consumption because all of the idle lengths of the disk drives #1 and #2 are less than the standby timeout (5 s). In case #2, the power consumption is increased and transaction throughput is decreased because the idle period length of disk drive #2 is greater than the standby timeout, but shorter than the break-even time (23.9 s). This difference causes an energy loss to spin up disk drive #2 and a transaction delay until the disk drive is spun up. Furthermore, this transaction delay halts I/Os to disk drive #1, causing another energy loss.

In cases #3 and #4, the disk drive power consumption is reduced when using our proposed method because the idle periods are longer than the break-even time (23.9 s). The degradation of transaction throughput results from the wait that is necessary for disk drive #2 to spin up.

6.1.2 Evaluation of Delayed I/O

6.1.2.1 Simulation setup

We evaluated our proposed method with a delayed write I/O function. No implementation of a delayed write I/O function exists on commercial DBMS. Therefore, we generate the I/O trace of the write I/O function from I/O traces obtained during experiments of data placement method (Fig. 6). We reorder these I/O traces along with the delayed I/O function policy. We then apply our power-saving function to the generated I/O traces during a certain period described in subsection 5.2. In this simulation, we use *f* of 60 IOPS and *C* of 600 s. The value of *i* is 1.7 IOPS for case #1, 0.5 IOPS for case #2, 0.2 IOPS for case #3, and 0.1 IOPS for case #4.

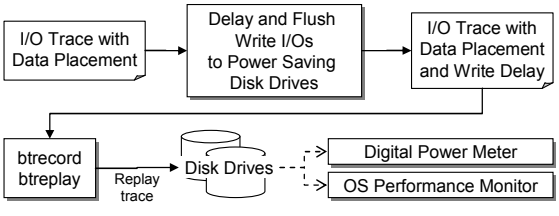

Fig. 11: Simulation Method for Write Delay.

We measured disk-drive power consumption by replaying the I/O trace on these two disk drives using the *btrecord* and *btreplay* programs (Brunelle 2007). Figure 11 portrays the power simulation method. The evaluation environment is identical to the measurement environment described in subsection 6.1.1.1 and Fig. 7.

This simulation setup measures only the power consumption, I/O response time, and throughput of the disk drives. Therefore we calculate transaction throughput according to the duration from when the transaction is interrupted until the status of disk drives changes from standby to active/idle state using I/O trace and transaction throughput measured in the previous subsection (6.1.1).

6.1.2.2 Simulation Results

Figure 12 portrays results of disk-drive power consumption for each case. Results show the measured power consumption without power saving and with power saving (case #1 – case 4#). Here, the power-saving method includes both the data placement method and the delayed write I/O method. Figure 12 shows that the disk drive power consumption is decreased, except in case #1. The maximum reduction of power consumption was 36% for cases #3 and #4; their power consumption levels were 11.4 VA and 11.3 VA, respectively.

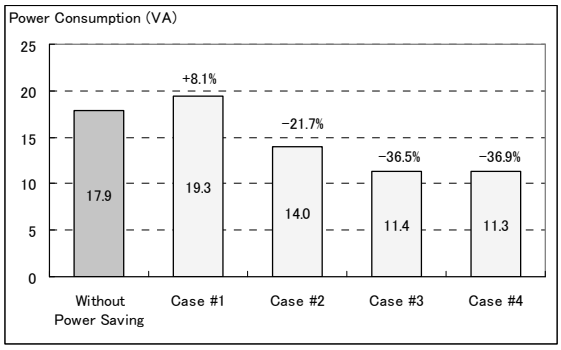

Fig. 12: Power Consumption (Two Disk Drives for DBMS with Delayed Write I/O).

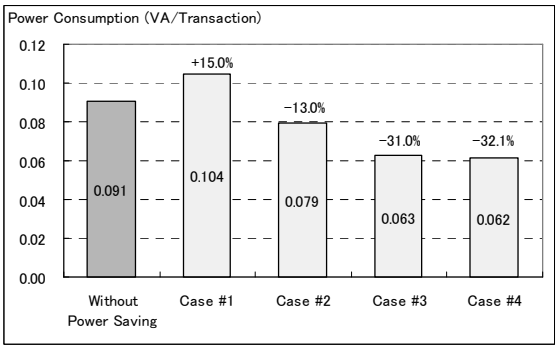

Fig. 13: Power Consumption per Transaction (Two Disk Drives for DBMS with Delayed Write I/O).

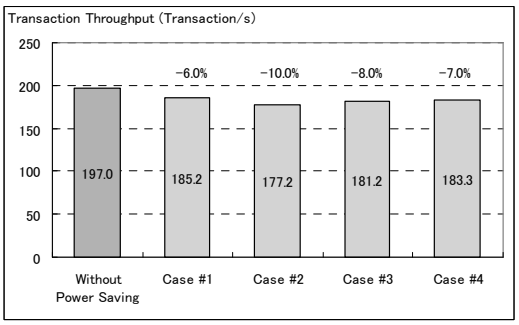

Fig. 14: Transaction Throughput (Two Disk Drives for DBMS with Delayed Write I/O).

Figure 13 presents the power consumption per transaction with and without power-saving methods. As Fig. 13 shows, power consumption is higher in case #1, but the power consumption is lower in other cases (#2–#4). Especially, the power consumption is decreased more than 30% in cases #3 and #4.

Figure 14 presents results of transaction throughput for each case. Transaction throughput was reduced by 6% in case #1, 10.0% in case #2, 8.0% in case #3, and 7.0% in case #4. Results show that our power-saving method only slightly affects the transaction throughput (less than 10%).

The power consumption was increased and transaction throughput was decreased in case #1 for reasons described in the preceding subsection. These results demonstrate that our power-saving methods—the data placement method based on TPC–C I/O behavior characteristics, and the delayed write I/O—radically reduce disk drive power consumption in a small DBMS configuration.

6.1.3 Impacts to Transaction Throughput

The read delay attributable to our power-saving method might affect the transaction response time. Therefore we investigate the read response time of disk drives with application of our proposed power-saving method.

We analyzed the relation between read response time and the number of reads (used trace files gathered at 6.1.1.3) to verify the impact of disk status changes. Figure 15 shows read response times and cumulative distribution function of the number of reads to disk drives when our power-saving functions is applied. As Fig. 15 shows, the number of reads for which the response time is longer than 1 s (which might strongly impact the transaction response time) is only 1.8% of all read I/Os. Moreover, the numbers of reads per transaction to tables and indexes on the disk drives with the applied power-saving function are less than 0.001 (calculated from Table 1 and Fig. 4). Therefore, the number of read I/Os per transaction affected by changing disk drive power status is less than 1/50,000: our proposed method has little impact on the transaction throughput.

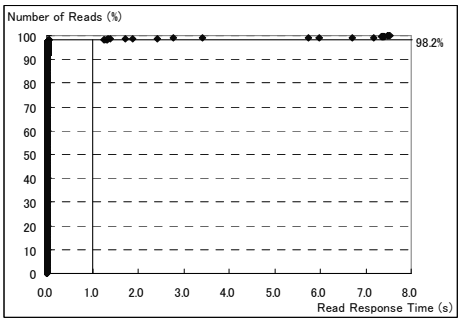

Fig. 15: Read response time and Number of Read I/Os (With Power Savings).

6.2 Evaluation of Five Disk Drives

The data allocation variations are limited in the two-disk-drive case. Therefore, we increased the number of disk drives from two to five. We simulated the power consumption of a five-disk-drive DBMS environment to confirm the effect of our method under a larger database configuration.

6.2.1 Simulation Condition

We placed Log data on one disk drive, and Database data on four disk drives. The data size differences among these data are minimized.

Table 2 shows the data placement and the percentage of the data size. Log data were sent to disk drive #1. Customer data were sent to disk drive #2. OrderLines data were sent to disk drive #3. Stock data were sent to disk drive #4. Other data were sent to disk drive #5. Disk drive power consumption characteristics were as described in section 3. The I/O trace data used for the simulation were collected in the environment described in section 4.

Disk Drive	Data	Capacity [%]
Disk #1	Log	20
Disk #2	Customer	17
Disk #3	OrderLine	25
Disk #4	Stock	30
Disk #5	District, History, Item, NewOrders, Orders, Warehouse	8

Table 2: Data Placement and Number of Disk Drives (Five Disks)

6.2.2 Simulation setup

In this simulation, we calculate the disk-drive power consumption using disk-drive power consumption characteristics presented in section 3, and the I/O trace data collected in the environment presented in section 4.

We first generate I/O traces for each disk drive with and without power-saving methods. The I/O trace is gathered by each data (Log and Tables). For I/O traces without power-saving methods, we merge these traces using each disk drive. We generate the I/O trace with power-saving methods as described in subsection 6.1.2.

For calculating the power consumption of disk drives, we approximate the measured power consumption using a quadratic function. The equation of power consumption and number of I/Os to disk drive per second is $y = -0.0005x^2+0.1016x+8.8025$, where x is the number of I/Os per second and y is the disk-drive power consumption (VA). The value of 8.8025 is the power consumption when the disk drive is in an idle state. We used the power consumption and I/O delay of the standby status of the disk drive in Fig. 3 presented in subsection 3.4.

6.2.3 Simulation Results

Figure 16 presents simulation results of the disk-drive power consumption per second. The data are normalized using the results without the power-saving method. As Fig. 16 shows, our data placement method based on TPC–C I/O behavior characteristics and delayed write I/O reduces the disk-drive power consumption by 41.2%.

Figure 17 shows the power consumption per transaction. These data are normalized using the results without the power-saving method. The power consumption per transaction without the power-saving method with data placement method is nearly equal to the power consumption per transaction without the power-saving method. The power consumption using all the proposed power-saving methods is decreased 40%.

Figure 18 depicts transaction-throughput simulation results. The data are normalized using the results without power-saving method. As Figure 18 shows, the transaction throughput using our proposed method is almost equal to the throughput without the power-saving method. The transaction throughput is reduced only with data placement method because a transaction waits for a disk drive spin up when the disk drive is in standby status. The transaction throughput with delayed write I/O is higher than that without it because the delayed write I/O gathers I/O into a short period of time and the frequency of a waiting disk drive's spin up is reduced.

These evaluation results demonstrate that our power-saving method saves more than 40% of disk-drive power consumption with little transaction throughput degradation despite very hard conditions of running TPC–C applications with very high transaction traffic. This result implies that our method can save more disk-drive power consumption when the I/O traffic becomes lower. Consequently, the proposed method is expected to follow dynamic workload changes of various applications at datacenters easily.

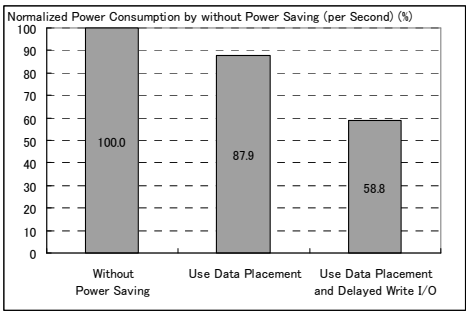

Fig. 16: Power Consumption (Five Disk Drives for DBMS with Delayed Write I/O).

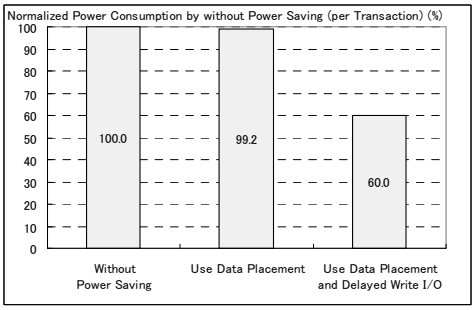

Fig. 17: Power Comparison per Transaction (Five Disk Drives for DBMS with Delayed Write I/O).

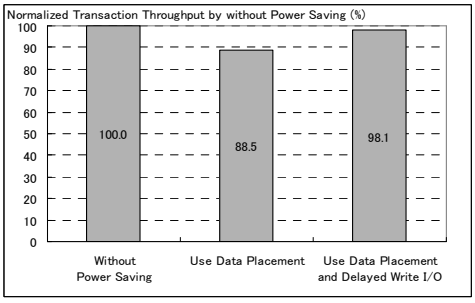

Fig. 18: Transaction Throughput (Five Disk Drives for DBMS with Delayed Write I/O).

7 Conclusion and Future Works

Business applications using DBMS are the major applications that are served at datacenters. Traditional power saving approache for these business applications is static because such power-saving methods depend on the hardware design. As described in this paper, we proposed a novel power-saving method combining application knowledge and a disk power-saving function. We then ran a TPC–C application and measured the disk-drive power consumption and the transaction throughput. Results show the potential of the dynamic power saving approach for business applications.

Our power-saving method enables dynamic reduction of disk-drive power consumption for TPC–C applications running with high throughput. The salient feature of our approach is the DBMS control of its I/O and power status of disk drives to reduce the disk drive power consumption. Our proposed method extends idle periods of disk drives using i) a data placement method based on I/O behavior of Log and Tables of database, and ii) delayed writing I/O of the DBMS buffer flush.

We also demonstrated that our method achieves an approximately 40% power reduction in a five-disk-drive case. Although we focus on power saving of OLTP applications in this paper, our method is likely to be applicable to other DBMS applications such as DSS or Web Services. The I/O traffic of these applications is not so heavy compared with the I/O traffic of TPC–C. Therefore, the proposed method offers great potential to save power consumption of datacenters.

We plan to extend the proposed power-saving method to a large RAID storage system and other workloads such as TPC–H and TPC–W. We then intend to develop a storage power saving system that offers very little degradation of transaction throughput.

8 References

Bohn, R. and Short, J. (2009): *How Much Information? 2009*. Report on American Consumers, http://hmi.u.csd.edu/pdf/HMI_2009_ConsumerReport_Dec9_2009.pdf, Accessed 9 Dec 2009.

Eastwood, M., Bozman, J.S., Pucciarelli, J.C. and Perry, R. (2009): *The Business Value of Consolidating on Power-Efficient Servers*. Customer Findings, IDC White Paper #218185, http://uk.logicalis.com/pdf/IDC_ White_paper_energy.pdf, Accessed 6 Jul 2009.

Yoder, A. G. (2010): *Green Storage Technologies and Your Bottom Line*, Storage Networking World Spring 2010 Tutorial, http://www.snia.org/education/tutorials/2010/spring/green/Alan%20Yoder_Green_Storage_Technologies.pdf, Accessed 20 Apr 2010

Rajecki, K. (2008): *Tiered Storage Architectures*. http://www.sun.com/solutions/landing/industry/education/pdf/webinar_050708.pdf, Accessed 12 Jul 2010.

Poess, M. and Nambiar, R.O. (2008): Power cost, the key challenge of today's data centers: a power consumption analysis of TPC–C results. *In International Conference on Very Large Data Base*, Auckland, New Zealand, 1229-1240, ACM Press.

Douglis, F., Krishnan, P. and Bershad, B. (1995): Adaptive Disk Spin-Down Policies for Mobile Computers. *Proc. of 2nd USENIX Symposium on Mobile and Location Independent Computing*, Ann Arbor, USA, 121-137, USENIX Association

Helmbold, D.P., Long, D.D.E., Sconyers, T.L. and Sherrod, B. (2000): Adaptive Disk Spin Down for Mobile Computers. *Mobile Networks and Applications* **5**(4): 285-297.

Gurumurthi, S., Sivasubramaniam, A., Kandemir, M. and Franke, H. (2003): DRPM: Dynamic Speed Control for Power Management in Server Class Disks, *Proc. 30th Annual International Symp. on Computer Architecture*, San Diego, USA, 169-181 IEEE Computer Society.

Zhu, Q., Chen, A., Tan, L., Zhou, Y., Keeton, K. and Wilkes, J. (2005): Hibernator: Helping Disk Arrays Sleep through the Winter. *Proc. 20th ACM Symposium on Operating Systems Principles*, Stevenson, USA, **39**(5): 177-190, ACM New York.

Gniady, C., Hu, Y.C. and Lu, Y.H. (2004): Program Counter Based Techniques for Dynamic Power Management. *Proc. 10th International Symposium on High Performance Computer Architecture*, Madrid, Spain, 24-35, IEEE.

Son, S.W., Kandemir, M. and Choudhary, A. (2005): Software-Directed Disk Power Management for Scientific Applications. *Proc. 19th IEEE International Parallel and Distributed Processing Symposium*, Denver, USA, 4, IEEE Computer Society.

Papathanasiou, A.E. and Scott, M.L. (2004): Power Efficient Prefetching and Caching. *Proc. USENIX 2004 Annual Technical Conference*, Boston, USA, 22, USENIX Association Berkeley.

Li, D. and Wang, J. (2004): EERAID: Power Efficient Redundant and Inexpensive Disk Arrays. *Proc. 11th Workshop on ACM SIGOPS European Workshop*, Leuven, Belgium.

Yao, X. and Wang, J. (2006): RIMAC: A Novel Redundancy based Hierarchical Cache Architecture for Power Efficient. *High Performance Storage System Proc. 2006 EuroSys Conference*, Leuven, Belgium.

Heath, T., Pinheiro, E., Hom, J., Kremer, U. and Bianchini, R. (2002): Application Transformations for Power and Performance-Aware Device Management. *11th International Conference on Parallel Architectures and Compilation Techniques*. Virginia, USA, 121-130

Colarelli, D. and Grunwald, D. (2002): Massive Arrays of Idle Disks for Storage Archives. *Supercomputing, ACM /IEEE 2002 Conference*.

Pinheiro, E. and Bianchini, R. (2004): Energy Conservation Techniques for Disk Array Based Servers. *Proc. 18th Annual International Conference on Supercomputing*, Saint-Malo, France, 68-78, ACM.

Weddle, C., Oldham, M., Qian, J. and Wang, A.A. (2007): PARAID: A Gear-Shifting Power-Aware RAID. *5th USENIX Conference on File and Storage*, San Jose, USA, USENIX Association.

Harizopoulos, S., Shah, M.A., Meza, J. and Ranganathan, P. (2009): Power Efficiency: The New Holy Grail of Data Management Systems Research. *4th Biennial Conf. on Innovative Data Systems*, Asilomar, USA.

TPC–H: Available: http://www.tpc.org/tpch/, Accessed 15 Jul 2010.

Poess, M. and Nambiar, R.O. (2010): Tuning Servers, Storage and Database for Power Efficient Data Warehouse. *26th IEEE International Conf. on Data Engineering*, Long Beach, USA, 1006-1017, IEEE Computer Society.

Son, S.W., Chen, G. and Kandemir, M. (2005): Disk Layout Optimization for Reducing Power Consumption. *Proc. 19th Annual International Conference on Supercomputing*, Cambridge, USA, 274-283, ACM.

Nishikawa, N., Nakano, M. and Kitsuregaw, M. (2010): Low power Management of OLTP Applications Considering Disk Drive Power Saving Function. *21st International Conference on Database and Expert Systems Applications (DEXA)*, Bilbao, Spain, Part I, LNCS 6261, LNCS.

Brunelle, A.D. (2007): btrecord and btreplay User Guide, http://www.cse.unsw.edu.au/~aaronc/iosched/doc/btreplay.html, Accessed 1 Apr 2010

tpcc-mysql (2004): https://code.launchpad.net/~percona-dev/perconatools/tpcc-mysql, Accessed 10 Oct 2009.

Verma, A., Koller, R., Useche, L., and Rangaswami, R. (2010): SRCMap: Energy Proportional Storage using Dynamic Consolidation. *8th USENIX Conference on File and Storage Technologies*, San Jose, USA, 267-280, USENIX Association.