Assignment #1 for Simulation (CAP 4800)

>>> SOLUTIONS <<<

This assignment covers material from the first week of class lecture and reading.

Problem #1 (5 points)

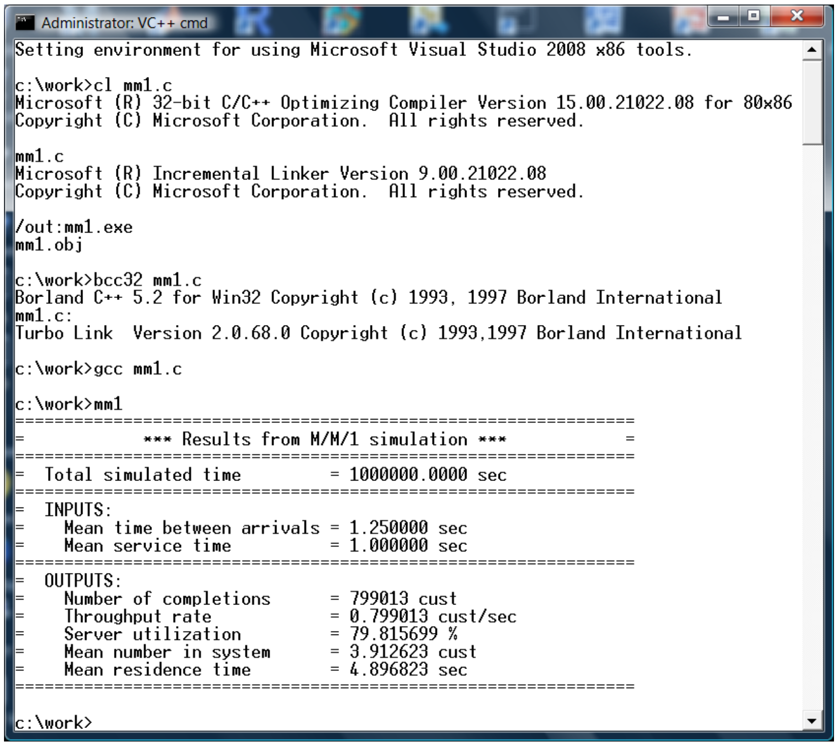
If is very important that you know what the class website contains. So, for this first fun problem you are to go on a frog hunt. There are images of frogs hidden on one, or more, page(s) that I have created found somewhere on, or linked to, the class website. So, go find the frogs! Give the URL of each page that contains a frog. It is a good idea to be aware of the content of the pages that contain the frogs.

> One frog can be found at the bottom of the page http://www.csee.usf.edu/~christen/cstyle.html. This page is found on the (top level) page http://www.csee.usf.edu/~christen/class3/misc3.html. A second frog can be found on the bottom of the reading page http://www.csee.usf.edu/~christen/class3/reading3.html. This page is linked-to from the course outline on http://www.csee.usf.edu/~christen/class3/outline3.html. The third frog is on the bottom of the miscellaneous page (http://www.csee.usf.edu/~christen/class3/misc3.html).

Problem #2 (5 points)

It is *absolutely critical* that you have a C programming environment that you are comfortable with. Download the program mm1.c from the Christensen tools page (http://www.csee.usf.edu/~christen/tools/toolpage.html). Compile it and run it. Take a screenshot of your compilation and execution. Submit the screenshot. Later in the semester you will need to use Visual C++ Express Edition 2008 as your development environment. So, if you do not yet have a C development environment on your PC I suggest this may be a good time to download and install VC++ Express 2008. You can find VC++ Express 2008 here: https://www.dreamspark.com/Products/product.aspx?productid=9. Why not VC++ Express 2010 or 2012? The CSIM software libraries that we will use later in the semester are compiled for VC++ Express 2008. If you are already committed to VC++ 2010 or 2012, see me and we will try to work-out a solution. Personally, I use command line for all of my C compilation – I do not use the IDE. You should use whatever you are comfortable with (that is, command line versus IDE).

> Here is a screenshot showing builds with cl, bcc32, and gcc, and then execution.



```
Setting environment for using Microsoft Visual Studio 2008 x86 tools.

c:\work>cl mm1.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

mm1.c
Microsoft (R) Incremental Linker Version 9.00.21022.08
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:mm1.exe
mm1.obj

c:\work>bcc32 mm1.c
Borland C++ 5.2 for Win32 Copyright (c) 1993, 1997 Borland International
mm1.c:
Turbo Link  Version 2.0.68.0 Copyright (c) 1993,1997 Borland International

c:\work>gcc mm1.c

c:\work>mm1
===============================================================
=            *** Results from M/M/1 simulation ***            =
===============================================================
=  Total simulated time         = 1000000.0000 sec
===============================================================
=  INPUTS:
=    Mean time between arrivals = 1.250000 sec
=    Mean service time          = 1.000000 sec
===============================================================
=  OUTPUTS:
=    Number of completions      = 799013 cust
=    Throughput rate            = 0.799013 cust/sec
=    Server utilization         = 79.815699 %
=    Mean number in system      = 3.912623 cust
=    Mean residence time        = 4.896823 sec
===============================================================

c:\work>
```

Problem #3 (50 points)

You are to find a paper in an ACM or IEEE conference that uses simulation methods to evaluate an ICT component or system. The paper must have been published in the last 10 years. For the paper, answer the following questions:

a) What is the problem or question being addressed?
b) Very briefly, what is the solution to the problem?
c) Describe the simulation model developed and used
d) Identify and describe the response variables
e) Identify and describe the factors and factor levels
f) Describe how the simulation results were presented

Include the paper in your submission package. Give the full and correct citation (in IEEE-CS style) for the paper. As you probably already know, the USF library gives you access to IEEE Xplore and the ACM Digital Library. I suggest using scholar.google.com (and not plain google.com) for searching for academic/research papers. Note that the paper must not be about simulation, but rather must be about something else (say, a new design or method) that uses simulation as the tool to evaluate the design or method being proposed and developed in the paper.

The paper I selected to review was: T. Xie and Y. Sun, "PEARL: Performance, Energy, and Reliability Balanced Dynamic Data Redistribution for Next Generation Disk Arrays," *Proceedings of the 16th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Baltimore, Maryland, USA, September 8-10, 2008.

a) The paper explores a new hybrid storage architecture that combines hard disks and flash disks. Hard disks and flash disks have different performance, energy, and reliability (and cost) properties. For example, flash disks are well suited to read data (no seek time), but less so to write data (due to erasure time). The problem addressed is data placement (on hard disk or flash disk) to trade-off performance, energy use, and reliability.
b) The solution is a new algorithm with data structures to determine data placement.
c) The simulation model itself is not described in the paper beyond saying that the authors develop an "execution-driven simulator that models a hybrid disk array which has one hard disk array an done flash disk array." The simulator is parameterized from real (commercial) equipment and is trace driven.
d) The key response variables are mean response time and energy consumption.
e) Control variables for the system include storage capacity, disk access time, disk transfer rate, disk idle power, disk active power, flash read speed, flash write speed, flash idle power, flash active power. Control variables for the workload (which are traces) include number of reads, number of writes, trace length, average size, and average interarrival time.
f) The simulation results are presented in bar graphs showing mean response time and energy consumed as a function of storage size and number of disks.

The paper is attached to these solutions.

Problem #4 (40 points)

Consider a generic PC. What are the response variables of interest to performance? What are the factors (and factor levels) of interest to performance? What are possible workloads of interest? Identify factors (and factor levels) related to both the system itself and the workload.

The two key response variables for a PC are:

1) Job throughput – how many jobs per second can the PC complete
2) Response time – how long it takes to complete a job (this includes how long it takes for the PC to start a program and how long it takes to finish executing a program)

Factors of interest for a PC are many and they include for the system (listed here with level measures):

1) Processing (including width of data path, MIPS rating, and number of cores)
2) Memory (including amount and access speed)
3) Storage (including type, size, seek time, and access speed)
4) I/O (including data rate – wired and wireless)
5) Energy use (including battery lifetime if a laptop)

For the workload the following are key:

1) Type of jobs (e.g., spreadsheet, work processing, numerical intensive, I/O intensive, GUI or batch)
2) Number of simultaneous jobs

PEARL: Performance, Energy, and Reliability Balanced Dynamic Data Redistribution for Next Generation Disk Arrays

Tao Xie
San Diego State University
xie@cs.sdsu.edu

Yao Sun
Teradata Corporation
calvin.sun@teradata.com

Abstract

Contemporary disk arrays consist purely of hard disk drives, which normally provide huge storage capacities with low-cost and high-throughput for data-intensive applications. Nevertheless, they have some inherent disadvantages such as long access latencies, high annual disk replacement rates, fragile physical characteristics, and energy-inefficiency due to their build-in mechanical and electronic mechanisms. Flash-memory based solid state disks, on the other hand, although currently more expensive and inadequate in write cycles, offer much faster read accesses and are much more robust and energy efficient. To combine the complementary merits of hard disks and flash disks, in this paper we propose a hybrid disk array architecture named HIT (hybrid disk storage) for data-intensive applications. Next, a dynamic data redistribution strategy called PEARL (performance, energy, and reliability balanced), which can periodically redistribute data between flash and hard disks to adapt to the changing data access patterns, is developed on top of the HIT architecture.

1. Introduction

Data placement problem or file assignment problem (FAP), the problem of allocating data (e.g., a set of files) onto multiple disks prior to serving I/O requests so that some cost functions or performance metrics can be optimized, has been extensively investigated in the past years [6][13][23][24][25]. The principle idea of dynamic data placement algorithms is to use historic information of arrived files and their recorded access characteristics to make a good allocation for each arriving file so that load balancing among multiple disks can be maintained. However, data placement algorithms alone are insufficient to retain load balancing because the access pattern of a file system might change over a long-term period [19]. Consequently, an originally good data placement scheme under an initial workload may no longer be the case in a later scenario [20]. Thus, dynamic data redistribution algorithms, which can intelligently reallocate data across multiple disks to adapt to the changing workload pattern, become essential.

While high-performance is the only goal pursued by traditional data placement and data redistribution algorithms [6][13][20][23], modern data placement strategies like SEA [25] also takes energy-efficiency into account as hard disks contribute a significant percentage of total energy consumption in a computing infrastructure [17]. Although recent energy-aware data placement approaches can noticeably save energy [25], the improvement in energy conservation is limited due to the inherent energy inefficiency of the underlying hard disk drives. Unfortunately, contemporary disk arrays consist purely of energy-inefficient hard disk drives. Hence, a novel storage architecture that not only offers high-performance but also saves energy is needed.

Current flash memory assisted hard disk storage systems are mainly proposed to be applied in mobile platforms like personal laptops [5][12] or embedded systems [3]. Essentially, these flash memory and hard disk mixed storage systems only take flash memory as an extra layer of cache buffer [1][12]. However, we argue that flash memory is useful more than just an additional cache buffer. More precisely, flash memory based solid state disk (hereafter flash disk) is also well-suited for enterprise level applications, where performance, energy conservation, and disk reliability need to be taken into account simultaneously [3].

Flash disks have the following apparent advantages, which make them ideal storage devices for enterprise applications. First, they inherently consume much less energy than mechanical mechanism based hard disks [3]. Second, because of their solid state design they are free of mechanical movements, and thus, have enhanced reliability [18]. Third, they offer much faster random access by eliminating unnecessary seek time delays and rotation latencies [10][11][18]. The main concerns on current flash disks are their considerably higher prices, relatively small capacities, and limited erasure cycles

[*] This work was supported by the US National Science Foundation under Grant No. CCF-0742187.

[14]. Therefore, it is wise to integrate small capacity flash disks with high capacity hard disk drives to form a hybrid disk array so that their complementary merits can be benefited by enterprise applications. To this end, we propose a novel, hybrid disk storage architecture (HIT) for next generation server-class disk arrays (see Fig. 1). Clearly, the HIT architecture can readily outperform traditional hard disk arrays in energy conservation. As for performance and reliability, novel data management software are needed to judiciously utilize the complementary merits of flash disk and hard disk so that storage systems for data intensive-applications like OLTP (online transaction processing) can achieve a high performance and reliability level.

In this paper we restudy dynamic data redistribution problem in the context of the new HIT architecture. An innovative dynamic data redistribution strategy called PEARL (performance, energy, and reliability balanced), which periodically redistributes a set of data based on their access characteristics and the distinct features of hard disk and flash disk, is developed on top of the HIT architecture. Considering data access characteristics and features of different types of disks (flash or hard), the PEARL strategy intelligently redistributes data to its right place (a flash disk or a hard disk) where the requests targeting on it can be most efficiently served.

The rest of the paper is organized as follows. In the next section we briefly introduce the related work and the motivation of this study. Section 3 presents the HIT architecture. The PEARL strategy and its overhead analysis are provided in Section 4. In Section 5 we evaluate performance of PEARL based on three real-world traces. Section 6 concludes the paper with summary and future directions.

2. Related work and our idea

Compared with numerous static data placement algorithms [6][13][24][25], only very few investigations on dynamic data allocation and redistribution (or reallocation) problem [2][20] have been accomplished. Scheuermann et al. proposed an array of heuristic algorithms for dynamic data redistribution by taking access pattern changes into consideration [20]. The basic idea of their algorithms is to minimize the queuing delays by distributing the load across the disks as evenly as possible and by selectively redistributing the load dynamically by "disk-cooling". However, all of their algorithms bear the following two major limitations [20]. First, they assume that all of the sub-requests are uniformly distributed among the disks, which obviously contradicts the fact that real workloads generally exhibit skewed access frequencies [13][19]. Second, their approaches just assume that the relevant workload parameters a priori can be estimated with sufficient accuracy without actually implementing any dynamic file access monitoring mechanisms. Even worse, the "known as a priori" assumption about workload parameters is against the spirit of dynamic data allocation and reallocation, where such workload characteristics can not be obtained in advance. Therefore, a new data allocation and reallocation (redistribution) strategy without the limitations mentioned above is needed to fully address the challenging dynamic data reorganization problem.

Presently flash memory is only used as extra cache buffer [1][5][12]. For example, a hybrid hard disk model, which embeds flash memory chips into a hard disk to make a hybrid disk, was proposed by Microsoft [15]. It takes flash memory as on-board memory buffers. Another typical example is SmartSaver, a disk energy-saving scheme for a mobile computer proposed by Chen et al. [5]. Their scheme uses the flash drive as a standby buffer for caching and prefetching data. Kim et al. extended the usage of flash memory device by developing an energy-efficient file placement technique named PB-PDC (pattern-based PDC) [12], which adapts the existing PDC (Popular Data Concentration) algorithm [16] by separating read and write I/O requests. PB-PDC locates all read-only data upon a flash drive while puts all rest data on a hard disk. Still, the PB-PDC technique only concentrated on one flash drive with a single hard disk in a mobile laptop computing environment. In addition, it did not take changing workload patterns into account.

The PEARL strategy first divides the hard disk array into multiple zones. Each zone contains the same number of blocks and each block is 512 bytes. It then monitors the access patterns of each zone. Data can be dynamically created or deleted. In addition, the access pattern of each zone could vary over time. Initially, all data including newly created ones are distributed across the hard disk array in RAID-X manner (e.g., X could be 0 or 5, see Fig. 1). At the end of each epoch, after obtaining statistics of each zone's access pattern, PEARL first separates all zones into three categories: write-excessive, read-exclusive, and read-write. If the write frequency of a zone exceeds the flash disk write cycle threshold value, it will be defined as a write-excessive zone and will stay on the hard disk array. All zones that do not belong to the write-excessive category will be further divided into two groups: read-exclusive and read-write. Zones with both read and write accesses are in the read-write group, whereas zones with read only accesses go into the read-exclusive group. Next, PEARL selects a set of zones that are appropriate for being allocated on the flash disk array from the read-exclusive and the read-write groups based on each zone's popularity and performance-energy trade-off parameter (Eq. 5). And then it reallocates these zones onto the flash disks. When data access pattern changes, PEARL redistributes zones between the flash disk array and the hard disk array accordingly.

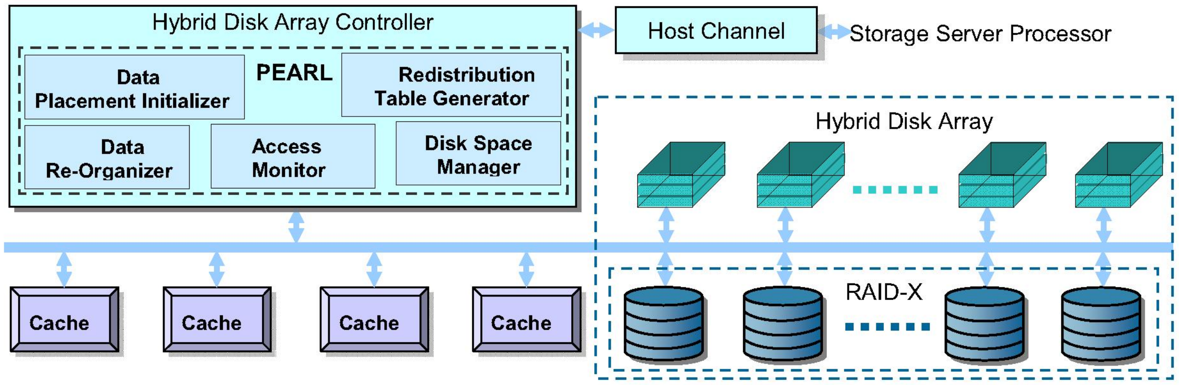


Figure 1. The HIT (*h*ybrid d*i*sk s*t*orage) architecture.

3. The HIT architecture

The HIT architecture is depicted in Fig. 1. Each flash disk cooperates with a hard disk through a dedicated high-bandwidth connection to compose a flash-hard disk pair, where load balancing can be achieved. Meanwhile, both the hard disk and the flash disk are directly attached to the system bus. All hard disks are organized in a RAID structure like RAID-0. The hard disk array plus its associated flash disks construct a hybrid disk array. Note that the number of flash disks and the number of hard disks are not necessarily equal. In our implementation, we adopt a one-to-one configuration because it makes data redistribution between the hard disk array and the flash disks simple (see Fig. 2).

Since all flash disks are emulated as hard disks, from the hybrid disk array controller point of view, there exist two groups of same type disks in the hybrid disk array. Within the hybrid disk array controller, some data management modules like PEARL are implemented to manage data across the hybrid disk array and the controller caches. The hybrid disk array controller is connected to the storage server host through the host channel. Note that multiple hybrid disk arrays each with its own disk array controller can be connected with the storage server processor simultaneously. The PEARL strategy consists of five modules, Data Placement Initializer (DPI), Redistribution Table Generator (RTG), Data Re-Organizer (DRO), Access Monitor (AM), and Disk Space Manager (DSM).

4. The PEARL strategy

4.1. How it works?

The PEARL strategy judiciously yet dynamically designates each zone as either *flash-favorite* or *hard-favorite* based on its I/O access characteristics. Each zone is then allocated onto its favorite disk array so that the complementary merits of flash disks and hard disks can be mostly utilized while their respective disadvantages can be avoided. It is understood that read-exclusive data is suitable for flash disks as they don't contribute any erasure cycles. Further, accessing the read-exclusive data on flash disk can significantly save energy and gain potential performance enhancement due to no seek time and rotation latency any more. Similarly, write-excessive data is more appropriate for hard disks where erasure cycle limitation doesn't apply. The most difficult task for a data redistribution strategy is to decide where some read-write popular zones should go. Unlike existing conservative algorithms such as PB-PDC [12], which immediately puts all read-write data onto hard disks to avoid any write cycles on flash disk, PEARL adopts a more open attitude and makes a smart decision based on a good trade-off between performance and energy saving.

Fig. 2 shows an illustrative example of data redistribution between hard disks and flash disks using the PEARL strategy. Assume that there are only two hard disks and two flash disks in the hybrid disk array (see Fig. 1). Further, assume that the capacity of a flash disk is only half of that of a hard disk. Initially, data was allocated across the hard disk array in some RAID structure and the flash disks are empty. PEARL divides the hard disk array into 4 zones and the size of each zone is 12 blocks. Each block is 512 bytes and has its LBA (Logical Block Address). Note that the extremely small size of a zone and the unrealistic small capacity of a disk in this example are for illustration. While the hard disk array was logically divided into 4 equal size zones, the flash disk array was partitioned into 2 same size zones as well. Immediately after the time instance 0, the access monitor (AM) starts to monitor the popularity (in terms of number of accesses) for each zone on the hard disk array. Assume that Zone 2 and Zone 3 are the hottest zones during the first epoch and they are all *flash-favorite*. Thus, at the end of the first epoch, the reorganization table generator (RTG) module generates a data reorganization table (DRT) and hands it to the data re-organizer (DRO), which

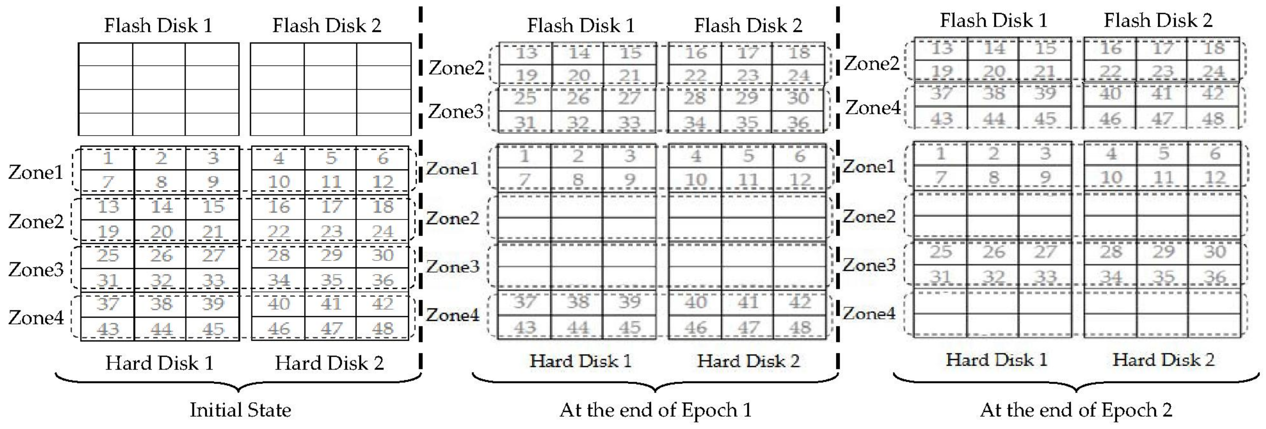


Figure 2. An example of data redistribution by PEARL during the first two epochs.

in turn transfers data of Zone2 and Zone3 from the hard disk array to the flash disk array. Therefore, all subsequent accesses targeting on the data of the two migrated zones will be directed to the flash disk array during the second epoch. Similarly, at the end of the second epoch, AM discovers that the hottest 2 zones change from (Zone2, Zone3) to (Zone2, Zone4). As a result, DRO first dumps data in Zone3 back to the hard disk array, and then, transfers data in Zone 4 to the flash disk array. By periodically updating popular flash-favorite data in the flash disk array, PEARL dynamically separates data onto two disk arrays so that requests can be served efficiently on distinct types of disks.

4.2. System models

The set of zones is represented as $Z = \{z_1, ..., z_i, ..., z_m\}$. The flash disk array is modeled as $FD = \{fd_1, ..., fd_j, ..., fd_n\}$, whereas the hard disk array is denoted by $HD = \{hd_1, ..., hd_j, ..., hd_n\}$. Let bl denote the size of a block in Mbyte and it is assumed to be a constant in the system. A zone z_i ($z_i \in Z$) is modeled as a set of rational parameters, i.e., $z_i = (s_i, r_i, w_i)$, where s_i is the zone's size in terms of number of blocks, r_i is the zone's read access rate (1/second), and w_i is the zone's write access rate (1/second). Each hard disk's transfer rate is t^h (Mbyte/second). Its active energy consumption rate and idle energy consumption rate are p^h (Watts) and i^h (Watts), respectively. Similarly, each flash disk is modeled as $fd_j = (r^f, w^f, p^f, i^f)$, where r^f is its read rate (Mbyte/second), w^f is its write rate (Mbyte/second), p^f is its active energy consumption rate (Watts), and i^f is its idle energy consumption rate (Watts). In addition, SK denotes average seeking time of a hard disk and RT represents average rotation latency of a hard disk. The time span of one epoch is denoted by T_e (second). Thus, the mean service time of a block of data in zone z_i served by a hard disk is

$$mst_i^h = SK + RT + bl / t^h \, . \qquad (1)$$

However, if the block is served by a flash disk, its mean service time becomes

$$\begin{aligned} mst_i^f &= [(r_i T_e / s_i) * (bl / r^f) + (w_i T_e / s_i) * (bl / w^f)] / [(r_i + w_i) T_e / s_i] \\ &= bl(r_i / r^f + w_i / w^f) / (r_i + w_i) \qquad (2) \end{aligned}$$

Hence, the performance gain pg_i in terms of mean service time reduction ratio of zone z_i is defined in Eq. 3.

$$\begin{aligned} pg_i &= mst_i^h / mst_i^f \\ &= (SK + RT + bl / t^h)(r_i + w_i) / bl(r_i / r^f + w_i / w^f) \end{aligned} \qquad (3)$$

For each read-write zone, we need to decide where to store it. Thus, we need to calculate its energy gain eg_i in one epoch in Eq. 4, where ec_i^h is the energy consumption of a block in zone z_i in one epoch if it is stored in the hard disk array, and ec_i^f is the energy consumption of the block in zone z_i in one epoch if it is in the flash disk array.

$$\begin{aligned} eg_i &= ec_i^h / ec_i^f \\ &= [mst_i^h * p^h * (r_i T_e + w_i T_e) / s_i] / [mst_i^f * p^f * (r_i T_e + w_i T_e) / s_i] \qquad (4) \\ &= (mst_i^h * p^h) / (mst_i^f * p^f) \end{aligned}$$

The performance energy ratio of zone z_i is defined as

$$per_i = (eg_i - 1) / (1 - pg_i) \, . \qquad (5)$$

PER represents the performance energy ratio threshold value set by administrator. Similarly, *PDA* is the *performance degradation allowed*, which is also a constant value set by administrator. The total number of write cycles of a flash disk is a constant *WC*, which is assumed to be 1 million in our simulation experiments. Besides, *DY* represents the duration years of a flash disk and we set *DY* as 5 years in Section 5. As a result, *WCPS* (write cycles per second) that is allowed by a flash disk is defined in Eq. 6 as below.

$$WCPS = (WC / DY) / (365 * 24 * 60 * 60) \qquad (6)$$

For instance, the value of *WCPS* in our simulations is around 0.0063 (1/second). Therefore, the reliability loss

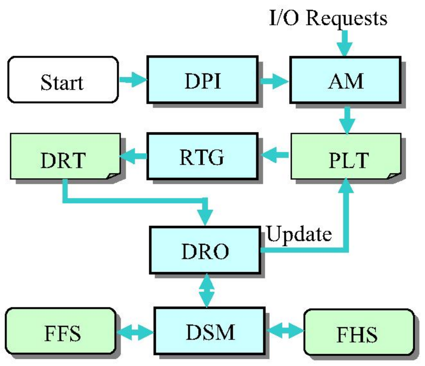


Figure 3. The PEARL strategy.

rl_i of zone z_i if it is stored on the flash disk array can be computed by

$$rl_i = \begin{cases} 1, \text{ if } w_i \geq WCPS \\ 0, \text{ otherwise} \end{cases} \tag{7}$$

The request set is $R = \{r_1, ..., r_k, ..., r_x\}$. Each request is modeled as $r_k = (lba_k, len_k, a_k, t_k)$, where lba_k is the starting logical block address, len_k is the number of bytes that the request accesses, a_k is the arrival time of request r_k, t_k is the type of the request r_k and it can be "*r*", "*w*", "*c*", and "*d*" representing "read", "write", "create", and "delete", respectively.

4.3. Implementations

The PEARL strategy consists of five modules that coordinate with each other via four data structures: popularity and location table (PLT), data redistribution table (DRT), free flash-disk space array (FFS), and free hard-disk space array (FHS) (see Fig. 3).

At the beginning, all data is striped across the hard disk array in some RAID fashion. Dynamically created files are also distributed initially across the hard disk array. PEARL first starts the DPI (data placement initializer) module, which creates PLT table for later use. After the hybrid disk array begins to serve I/O requests, PEARL launches the AM process to record each zone's popularity in terms of number of accesses within one epoch in the PLT table (see Fig. 3). The PLT table, which contains the latest popularity information of each zone and its location (flash or hard), will be used later by the RTG module (see Fig. 4) to generate the DRT table. A sample table of PLT is given in Table 1. For example, zone3 has 452 read accesses and 37 write accesses during one epoch and its current location is on the flash disk array. After labeling all popular zones, RTG generates the DRT table, which lists all zones that need to be reallocated between the hard disk array and the flash disk array. Guided by the DRT table, the DRO module reallocates all zones in the DRT table to their favorite

Table 1. A sample PLT table.

Zone ID	No. of Reads (R)	No. of Writes (W)	Place (P)
1	206	0	1
2	0	119	0
3	452	37	1
…		…	0
m	*x*	*y*	0

destinations. During the data redistribution process, DRO consults to the DSM (disk space manager), which is responsible for managing both disk space for hard disk array and flash disk array.

Based on the observations from real traces [19], the popularity of a piece of data either gradually changes or almost keeps constant. Therefore, it is feasible for

```
1.  Sort all zones in PLT into a list Z in descending order of
    their no. of reads R at the end of each epoch
2.  Create a blank DRT (Data Re-Distribution Table) table
3.  Initialize flash_room as the flash disk array capacity
4.  for each zone $z_i$ starting from the first one in Z do
5.      case (R = = 0 ) and (W = = 0)
6.          Tag $z_i$ as a hard-favorite zone
7.      case (R = = 0 ) and (W > 0)
8.          Tag $z_i$ as a hard- favorite zone
9.      case (W = = 0 ) and (R > 0)
10.         Tag $z_i$ as a flash- favorite zone
11.     case (R > 0 ) and (W > 0)
12.         Use Eq. 7 to compute its reliability loss $rl_i$
13.         if $rl_i$ = = 1
14.             Tag $z_i$ as a hard- favorite zone
15.         else
16.             Use Eq. 3 to compute its $pg_i$
17.             case $pg_i$ > 1
18.                 Tag $z_i$ as a flash- favorite zone
19.             case (1- PDA) < = $pg_i$ < = 1
20.                 Use Eq. 4 to compute its $eg_i$
21.                 if $eg_i$ > 1
22.                     Use Eq. 5 to calculate its $per_i$
23.                     if $per_i$ > = PER
24.                         Tag $z_i$ as a flash- favorite zone
25.                     else
26.                         Tag $z_i$ as a hard- favorite zone
27.                     end if
28.                 else
29.                     Tag $z_i$ as a hard- favorite zone
30.                 end if
31.             case $pg_i$ < (1-PDA)
32.                 Tag $z_i$ as a hard- favorite zone
33.         end if
34.     if (flash_room – $s_i$) > = 0
35.         flash_room = flash_room – $s_i$
36.     else
37.         Exit
38.     end if
39. end for
35. Generate DRT based on PLT and the new tags
```

Figure 4. The RTG module.

PEARL to use the most recent access statistics of a zone to predict its next epoch data access pattern in a dynamic I/O workload scenario. Obviously, data redistribution is achieved at the cost of both performance degradation and extra energy consumption. Fortunately, PEARL only needs to re-distribute a small portion of popular zones at the end of each epoch due to the smooth changes in data access pattern. Also, to reduce the overhead associated with data redistribution, PEARL confines the time span of each epoch so that frequent data reallocation can be avoided. Due to the space limit, we only present the RTG module (see Fig. 4). PEARL has to pay extra data reallocation time and energy consumption caused by data redistribution at the end of each epoch. In the worst-case of our trace-driven simulations when using the Financial2 trace, we observed that when a flash disk capacity is 4 GB, the total number of disks in each array is 6, and the length of an epoch is 1000 seconds, the total size of data that swaps between the hard disk array and the flash disk array is 40 MB. Hence, the file redistribution time in each epoch is around 2.06 seconds and the energy overhead caused by reallocation is only around 22.34 joules.

5. Performance evaluation

5.1. Experimental setup

We developed an execution-driven simulator that models a hybrid disk array, which has one hard disk array and one flash disk array. For hard disk, we use the parameters of the Seagate Cheetah 15K.4 73.4 GB [4]. For flash disk, we adopt the specifications of the Adtron A25FB-20 Flashpak with capacity varying from 4 GB (default value) to 32 GB [22]. The average access time for the hard disk is 6.2 msec. Its transfer rate is 77 MB/sec. The hard disk active power and idle power are 17 watts and 11.9 watts, respectively. The read speed of the flash disk is 78 MB/sec and its write speed is 47 MB/sec. The flash disk active power and idle power are 3.43 watts and 1.91 watts, respectively. The number of flash disk is always equal to the number of hard disk and it varies in the range (6, 8, 10, 12, 14, 16) with 8 as the default value. The default length of an epoch is set to 1000 seconds. The block size *bl* is set to 512 bytes. The default zone size is 10 Mbytes. The performance metrics include *mean response time* (average response time of all file access requests submitted to the hybrid disk array), *energy consumption* (energy consumed by the hybrid disk array during the process of serving the entire request set), and *write cycles per block* (the number of writes per block on one flash disk during one day).

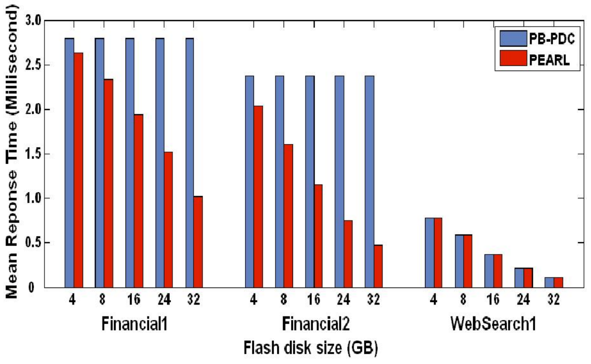
We evaluate the PEARL and the PB-PDC algorithms by running trace-driven simulations over three real-life traces: Financial 1, Financial 2, and WebSearch1, which have been widely used in the literature [21]. Financial1 and Financial2 were collected from requests to OLTP applications at two large financial institutions. WebSearch1 is an I/O trace from a popular search engine [21]. Since the simulation times in our experiments are much shorter compared with the time spans of the traces, we truncate each trace such that only the first 500,000 I/O requests are included. The statistics of each trace are listed in Table 2.

Table 2. The statistics of the traces.

Parameter	Financial 1	Financial 2	WebSearch1
Reads	370658	411344	499900
Writes	129342	88656	100
Length (sec.)	5450	3950	1500
Ave. size (bytes)	5359.2	2314.8	15402
Ave. inter-arrival time (ms)	10.9	7.9	3

5.2. Experimental results

The first group of experiments was conducted to study the impact of flash disk capacity on the performance and energy consumption of the two algorithms (Fig. 5). In the



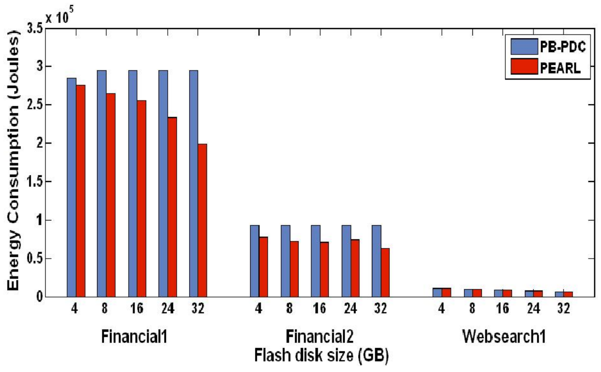


Figure 5. A comparison between the two algorithms in terms of flash disk capacity.

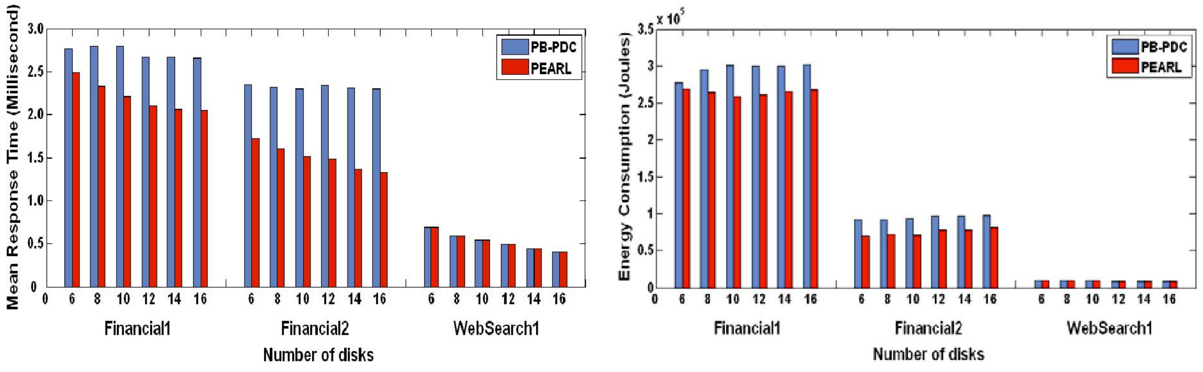


Figure 6. A comparison between the two algorithms in terms of disk number.

Financial 1 trace simulations, an average improvement of 29.1% in mean response time and 22.3% in energy consumption were observed by PEARL over PB-PDC (Fig. 5). In the Financial 2 trace experiments, PEARL achieves an average improvement of 41.2% in mean response time and 23.8% in energy consumption. However, when using the WebSearch1 trace, there is no obvious difference between PEARL and PB-PDC because the I/O requests are extremely read-dominant (99.98%). As a result, PEARL and PB-PDC allocate the same data sets onto flash disks. The second group of experiments tests the scalability of the two algorithms by varying disk number from 6 to 16 (see Fig. 6). As we can see that PEARL consistently outperforms PB-PDC in both mean response time and energy consumption. More importantly, PEARL scales well in terms of performance. It is understood that energy consumption becomes higher as the number of disks increases (see Fig. 6). The last group of experiments examines the reliability impact caused by the PEARL strategy. Since PB-PDC does not allocate any data that has write requests onto the flash disk, it has no negative effect on flash disk write cycles. Therefore, we only record maximal, mean, and standard deviation of write cycles per block in one day for PEARL. In the worst-case (Financial 2), PEARL results in only less than 20 write cycles per block within one day (Fig. 7), which is far below 544 write cycles per day, the threshold write cycle value of a flash disk with total 1 million write cycles and 5 years warranty. Thus, the impact of PEARL on flash disk reliability can be safely omitted.

6. Conclusions

Dynamic data allocation and reallocation (redistribution) problem has been largely investigated in the past years [2][13][20][23]. The only goal for existing algorithms such as HP (Hybrid Partition) [13], C-V (Cool Vanilla) [23], and HB (Simple Heat Balancing) [23] is to improve performance in terms of mean response time. Nowadays, however, energy consumption becomes a severe concern in data-intensive applications like OLTP. Thus, modern dynamic data allocation and redistribution strategies need to be both performance-driven and energy-aware. Unfortunately, traditional pure hard disk based disk arrays provide little room for researchers to amend current algorithms to be energy-efficient. Therefore, a new energy-efficient disk storage system is greatly needed. To this purpose, we propose a hybrid disk array architecture named HIT (_h_ybrid d_i_sk s_t_orage), which combines the complementary merits of hard disks

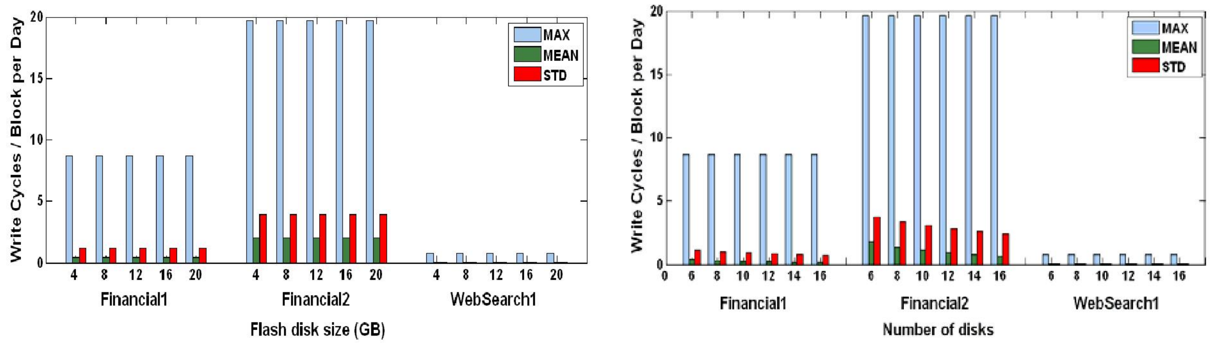


Figure 7. The write cycles per block within one day caused by the PEARL strategy.

and flash disks. More importantly, a novel dynamic data redistribution PEARL built on top of the HIT architecture is designed and implemented. PEARL noticeably improves performance while reducing energy consumption without compromising flash disk reliability. Comprehensive simulation experiments using real-world traces demonstrate that PEARL consistently outperforms an existing dynamic file assignment algorithm PB-PDC.

We will extend our scheme by considering a dramatically dynamic environment, where data access patterns can suddenly change. As a result, a high data redistribution cost may arise as the number of data zone migrations increases substantially. One possible solution is to use data replication technique.

References

[1] M. Arlitt and C. Williamson, "Web server workload characterization: the search for invariants," *Proc. ACM SIGMETRICS Conf.*, pp. 126-137, May 1996.

[2] R. Arnan, E. Bachmat, T.K. Lam, and R. Michel, "Dynamic data reallocation in disk arrays," *ACM Transactions on Storage*, Vol. 3, Issue 1, Article 2, March 2007.

[3] K. Cash, "Flash Solid State Disks - Inferior Technology or Closet Superstar?", BiTMICRO Networks.

[4] Cheetah 15K.4 Mainstream enterprise disc drive storage, Seagate Cheetah ST3146854LW.pdf.

[5] F. Chen, S. Jiang and X. Zhang, "SmartSaver: Turning Flash Drive into a Disk Energy Saver for Mobile Computers," *Proc. International Symposium on Low Power Electronics and Design*, pp. 412-417, 2006.

[6] W. Dowdy and D. Foster, "Comparative Models of the File Assignment Problem," *ACM Computing Surveys*, Vol. 14, No. 2, pp. 287 - 313, 1982.

[7] D. Dumitru, Understanding Flash SSD Performance, EasyCo LLC, August 2007.

[8] A. Fitzgerald, "Flash Disk Reliability Begins at the IC Level," *COTS Journal*, http://www.cotsjournalonline.com/home/article.php?id=100053, Jan. 2004

[9] J.W. Hsieh, T.W. Kuo, L.P. Chang, "Efficient Identification of Hot Data for Flash Memory Storage Systems," *ACM Transactions on Storage*, Vol. 2, Issue 1, pp. 22-40, February 2006.

[10] A. Kawaguchi, S. Nishioka, and H. Andmotoda, "A flash-memory-based file system," *Proc. USENIX Technical Conference*, pp. 155-164, 1995.

[11] H. Kim and S.G. Lee, "A new flash-memory management for flash storage system," *Proc. The 23rd International Computer Software and Applications Conference*, pp. 284-289, 1999.

[12] Y.J. Kim, K.T. Kwon, and J. Kim, "Energy-efficient file placement techniques for heterogeneous mobile storage systems," *Proc. of the 6th ACM & IEEE International conference on Embedded software*, pp. 171-177, 2006.

[13] L.W. Lee, P. Scheuermann and R. Vingralek, "File assignment in parallel I/O systems with Minimal Variance of Service Time," *IEEE Transactions on Computers*, Vol. 49, No. 2, pp. 127-140, Feb. 2000.

[14] B. Levine, Dell and Alienware Offer Samsung 64-GB SSDs, September 2007, http://www.newsfactor.com/story.xhtml?story_id=55235.

[15] R. Panabaker, "Hybrid Hard Disk and ReadyDrive™ Technology: Improving Performance and Power For Windows Vista Mobile PCs," Microsoft WinHEC 2006.

[16] E. Pinheiro and R. Bianchini, "Energy Conservation Techniques for Disk Array-Based Servers," *Proc. Conference on Supercomputing*, pp. 88-95, June 2004.

[17] Power, Heat and Sledgehammer, White paper, Maximum Institution Inc., Apr. 2002.

[18] Product Specification, Adtron A25FB-20 Flashpak Data Storage, http://www.adtron.com/pdf/A25FB-20-sum100807.pdf.

[19] D. Roselli, J.R. Lorch, and T.E. Anderson, "A Comparison of File System Workloads," *Proc. USENIX Technical Conference*, pp. 44-54, June, 2000.

[20] P. Scheuermann, G. Weikum, and P. Zabback, "Data partitioning and load balancing in parallel disk systems," *VLDB,* Vol. 7, Issue 1, pp. 48-66, 1998.

[21] SPC TRACE FILE FORMAT SPECIFICATION, http://traces.cs.umass.edu/index.php/Storage/Storage

[22] Storage Products, A25FB-20-R2spec101507.pdf.

[23] G. Weikum, P. Zabback, and P. Scheuermann, "Dynamic file allocation in disk arrays," *ACM SIGMOD,* Volume 20, Issue 2, pp. 406-415, June 1991.

[24] T. Xie, "SOR: A Static File Assignment Strategy Immune to Workload Characteristic Assumptions in Parallel I/O Systems," *Proc. 36th International Conference on Parallel Processing (ICPP)*, September 10-14, 2007.

[25] T. Xie, "SEA: A Striping-based Energy-aware Strategy for Data Placement in RAID-Structured Storage Systems," *IEEE Transactions on. Computers*, Vol. 57, No. 6, pp. 748-761, June 2008.