PAM: An Efficient Power-Aware Multi-level Cache Policy to Reduce Energy Consumption of Software Defined Network

Xiaodong Meng*, Long Zheng†, Li Li‡ and Jie Li‡
Shanghai Key Laboratory of Scalable Computing and Systems,
Department of Computer Science & Engineering, Shanghai Jiao Tong University, Shanghai, China 200240
*xiaodongmeng1985@sjtu.edu.cn, ‡{lilijp, lijie}@cs.sjtu.edu.cn, †Corresponding Author: longzheng@sjtu.edu.cn

***Abstract*—Nowadays energy consumption is one of the most significant aspects in Internet operations, where multi-level routing is widely used. In a typical hierarchical router cache structure, the upper level storage serves as a cache for the lower level, which forms a distributed multi-level cache system. In the past two decades, several classic LRU-based multi-level cache policies were proposed to improve the overall I/O performance of storage systems. However, few power-aware multi-level cache policies focus on the storage devices in the bottom level, which consume more than 27% energy of the whole system [20].**

To address this problem, in this paper, we propose a novel Power-Aware Multi-level cache (PAM) policy, which can reduce the energy consumption of storage devices with both high performance and high I/O bandwidth. In our PAM policy, a proper number of cold dirty blocks in the upper level cache are identified and selected to flush directly to the storage devices, which provides high probability to extend the duration time of data disks with standby status. Thus the energy consumption can be reduced. Simulation results show that, compared to the existing popular cache schemes such as PA-LRU, PB-LRU and Demote, PAM saves the power consumption by up to 15% under different I/O workloads, which improves the energy efficiency by up to 50.5%.

***Index Terms*—Storage System, Multi-level Cache, Energy Consumption, I/O Performance, Hint**

I. Introduction

Multi-level cache is one of the most significant techniques to bridge heterogenous network devices to provide high aggregate and concurrent I/O performance, which is widely used for network services such as cloud computing. In a typical hierarchical router caching structure, the storage devices in the upper level serve as caches to accelerate the I/O processing for the lower level, which forms a distributed multi-level cache system. Based on the access patterns of various I/O workloads, a multi-level cache system typically uses hints to identify the status of data blocks in different levels, which provide a global view of the storage system to enhance the I/O performance.

Research on multi-level cache becomes popular in the last two decades. Several classic policies are proposed to improve the overall performance, which make data blocks exclusively in different levels by using various types of hints [21], [24]. If a data block is demoted from upper level to the lower level, a flag can be applied with this block, which is called "demote hint". Meanwhile, if a data block is promoted from the lower level to the upper level, the corresponding flag is called "promote hint". If the status of data blocks are identified by an equation or an experienced function according to the I/O workloads of different applications, the corresponding flags are "application hints". According to the differences among hints, multi-level cache policies can be divided into three categories, demote-based multi-level policies [27], [28], promote-based multi-level schemes [8], [29], [31] and application-based multi-level algorithms [14], [16].

However, existing approaches focus on enhancing the I/O performance [27], [8], [12], but ignore the energy efficiency of storage systems, which is a critical issue in tremendous literatures [36], [25]. In particular, in a multi-level cache system, energy consumption of storage devices in the bottom level is ill-considered. For example, in Figure 1, we compare the optimal and typical cases[1] by using write-back and flush policies in a multi-level cache. It is clear that the optimal write-back and flush policies can save the energy consumption of storage devices by up to 80%. Hence an efficient energy management policy in multi-level cache is necessary.

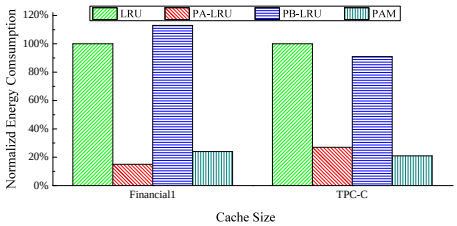


Fig. 1. A normalized energy consumption of different policies and corresponding optimal model under two real traces. (cache size is 64MBytes, the time window is 5 second). The Write-Back sends only the last dirty block to disk when cache is full, while the Flush writes all the dirty blocks. The optimal result of each policy is implemented by re-organising all write in a certain time window to maximum the standby duration of disks.

With the development of energy-efficient architectures [32] and green storage [3], nowadays energy consumption receives more attention than ever. Although a few power-aware LRU-based policies [35], [36] are designed to make sharp energy reduction for the storage systems, they are insufficient to maintain high performance for distributed multi-level cache systems [27], [8], particularly in terms of low aggregate hit ratio. On the other hand, typical multi-level cache algorithms [27], [8], [14], [16] can achieve high performance, while the energy consumption cannot be saved. These algorithms pro-

[1]This case is typical in multi-level cache algorithms such as Demote [27], Promote [8]

INISCom 2015, March 02-04, Tokyo, Japan

DOI 10.4108/icst.iniscom.2015.258322

vide a global management among all cache levels via various types of hints, which make caches exclusively to increase the usage of cache space. Therefore, the I/O performance can be improved. However, in these algorithms, the energy-saving information are neither involved in hints, nor in global management strategies. Thus the energy consumption cannot be guaranteed at a low level. In summary, existing cache policies cannot provide an efficient management policy to provide both high performance and low energy consumption for multi-level cache.

A possible solution against the energy efficiency issue is to employ Software Defined Networking (SDN) [19]. A SDN based network offers a central control for overall network topology while it guarantees the communication among difference network devices. SDN decouples the system into control and data planes. In control plane, a flow table is designed to record the routing information for specific flows in either centralized or distributed manners. For different traffic pattern, the control plane can compute a optimal data path for either high performance or green energy networking.

In this paper, we propose a novel Power-Aware Multi-level cache (PAM) scheme, which can increase the energy efficiency of a distributed multi-level cache system. PAM assembles small I/O write requests in a same disk by writing only once, and the status of the disk can be set to energy-saving mode (e.g., sleep or standby).

Our contributions include,

- We propose a novel multi-level cache policy (PAM) using power-aware information of both data blocks and disks, which efficiently reduces energy consumption of the whole storage systems.
- We develop a model to mathematically describe and analyze our PAM scheme. This model can easily compare the accumulative energy efficiency of all data blocks in each disk.
- We implement our PAM policy which shows higher energy efficiency compared to other popular power-aware cache algorithms.

The rest of this paper continues as follows, Section II briefly overviews related work and our motivation. In Section III we present the design, model and replacement policy of PAM. In Section IV, we present the simulation results by using various multi-level cache approaches. Finally we conclude the paper in Section V.

II. Related Work and Our Motivation

There are three options to reduce energy consumption of Software Defined Network. Reduction of redundant network traffic by flow control or network redesign, replacement of components by energy efficient devices, and using local/global data control policy to optimize resource utilization of SDN [23]. The first two methods are one-time implementation and static. The scalability of the system is restricted. Many researches on router power focuses on changing the traffic pattern or data flow path locally or globally, creating more opportunity for network devices to sleep during the idle time. Gupta et al. [9] consider the problem of aggregating traffic along a few routes in inactive period, making several idle routers more idle. Lun et al. [22] propose to establish minimum-cost multicasting connections over coded packet networks. These methods reaches green routing through routing control at link level. They cannot reduce the amount of traffic on the network radically. While our algorithm optimizes the data cache of routers on the passway of dataflow, which forms a multi-level cache model. The optimization reduces the data amount at routing while can be scalable to any network topology with a minimal cost.

TABLE I
Symbols in This Paper

Parameters	Definition
γ	Flush ratio of dirty blocks
n	Total number of cache levels
L_i	The i^th cache level
N_d	The number of dirty blocks in L_n
x	A random number of dirty blocks
$D(x)$	A value which indicates the current disk status
$I(x)$	History demotion/promotion information of block x
$U(x)$	Update frequency of block x
$H(x)$	Hotness value of block x
B_i	The i^{th} block
t_i	An interval which between two adjacent disk requests
W_i/R_i	The i_{th} write/read request.
T_{pw}	Life time of a sequential write request to data blocks
P_a	Disk power in the active mode
P_{dl}	Disk power in the idle mode
P_s	Disk power in the standby mode
P_{up}	Power required to spin a disk up
P_{down}	Power required to spin a disk down
T_s	Duration time of a disk in the standby mode
T_{min-s}	Minimum value of T_s to achieve energy-saving of storage devices
T_{up}	Time consumed to spin a disk up
T_{down}	Time consumed to spin a disk down
C	positive write count, which is the total number of dirty blocks aggregated in a request
C_{temp}	A temporary value of C

A. Disk Power Modes (DPM)

A typical enterprise Hard Disk Drive (HDD) has three modes, Active, Idle and Standby. Meanwhile, several HDDs has a sleep state. In the active mode, the drive is executing a Read or a Write command. In the idle mode, the drive is not actively working but the heads are flying over the disk. Therefore, the energy consumption in the idle mode is slightly lower than the active mode, but the disk is out of service. In the standby mode, the drive is not spinning while the electronic interface still accepts system calls. Various approaches are presented to analyze different modes of various disks [11], [10]. If we want to switch between the active mode and the idle mode, a small amount of energy need to be consumed. For another scenario, if we want to change the status by spinning up from the standby mode, a huge amount of time and energy consumption are required.

The principle task of an energy conscious disk management policy forces transition to standby mode when there is a potential energy saving. It consists of two steps, detecting suitable idle periods and spinning down the disk to a low power mode. Generally, history behavior is traced in the detection step to make a prediction on how long the next idle period last. If the period is long enough to overweigh the spin-up/down cost, the disk shifts into standby mode.

To make an accurate prediction on the duration of an upcoming idle period, Disk Power Modes (DPM) [17] is one of the most significant ways to guarantee efficient spin-up/spin-down in storage devices.

Research on DPM has appeared in many literatures. A fixed threshold is used in [7], wherein if the idle period lasts over two seconds, the disk is switched to standby mode, and is re-spun up only when the next request arrives. Irani *et al.* [13] describe a scheme to calculate the minimal length of an idle period to justify the spin-up/spin-down costs for multiple power modes. Gurumuthi *et al.* [10]. have proposed an approach, in which the disk can spin to several lower speed modes dynamically to expand the energy saving opportunities. Du *et al.* [3] introduce a three-state disk model to conserve energy of streaming media server with QoS guaranteed.

B. Cache Replacement Algorithm

Multi-level cache is a topic in recent years. MQ [34] identifies three properties for a good second level buffer cache: minimal lifetime, frequency-based priority and temporal frequency to efficiently manage the second level buffer cache. With the development of hints [21], [24], multi-level cache algorithms enter a new era. Typically, they can be divided into three categories, demote, promote and application hint based algorithms. Several popular cache policies are presented in many conferences and journals, such as Karma [29], MC^2 [30], CLIC [16], Hint-K [28] etc.

Many researchers have attempted to decrease the power consumption by software approaches. Previous literatures [26], [6] aim to control cache power states adaptively at runtime. Matthew *et al.* [26] purpose a memory management infrastructure to save energy. Cai *et al.* [2] present a scheme to adjust the cache size and disk time out value for reducing the average energy consumption. Li *et al.* [15] propose a memory-disk joint power management policy to constraint energy with guaranteed overall performance. Previous studies [35], [36] investigate the role of storage cache management algorithms to decrease disk energy consumption. These approaches reduces the frequency of spin-up, which is caused by miss when cache is full.

C. Our Motivation

Nowadays most commercial storage servers use a large storage cache to speed up I/O processing, where many cache replacement algorithms are proposed to cache hot data as much as possible to improve performance. However, according to an investigation on the energy consumption of disk drives [35], the performance-oriented cache algorithms is not energy-optimal. Typical cache policies maximize the usage of cache space by adjusting the request queues, which directly affect the energy consumption of data disks. These cache algorithms are difficult to tradeoff between the performance and energy consumption, which motivates us to propose a new cache policy in this paper.

TABLE II
RELATIONSHIP BETWEEN UPDATE FREQUENCY OF DATA BLOCKS AND DISK OPERATIONS (THE DATA IS COLLECTED IN $Financial1$ TRACE)

Write policy	Disk operations
Typical write-back	1
Low update frequency first	0.95
High update frequency first	1.08

III. POWER-AWARE MULTI-LEVEL CACHE (PAM) POLICY

In this section, we give the details on the Power-Aware Multi-level Cache (PAM) Policy, which involves the identification of hot blocks, demotion/promotion policies, write mechanism, etc.

A. Design Purpose

Existing power-aware storage cache methods save energy by extending the idle periods of storage devices. The major limitation is that they are not comprehensive solutions for both cache and storage devices. Based on this key insight, we introduce a novel multi-level cache scheme (PAM) by providing both high performance and low energy consumption for storage systems.

Regarding to the write requests, PAM uses a specific scheme to save energy consumption. First, according to the data blocks in each level, PAM defines a value to identify hot data blocks, which includes the history hint information, the corresponding disk status, update frequencies, etc. Second, a proper demotion/promotion policy are designed to improve the performance. Third, based on the hotness value of data blocks, an efficient write mechanism are proposed to flush write requests to standby/sleep data disks. In addition, PAM also has strategy to handle data consistency. In the following subsections, we give the design details of these techniques.

B. Identification of Hot Blocks

In this subsection, we illustrate the identification of hot blocks in PAM.

PAM exploits the *hotness* of a dirty block to decide whether the block is suitable to be updated on storage devices. Typically, the hotness value of a data block involves three elements as below,

- Disk Status (delegated by "D"): This element records the Disk Power Mode (DPM) information. When a disk is in the low power mode (standby or sleep), the disk status is set to "1". Otherwise, the value is set to zero. Basically, the value is updated when a disk switches between high and low power modes.

- Inter-cache Communication (delegated by "I"): This element indicates the blocks' activities among different cache levels. If a block is demoted from the upper to the current level, the value of inter-cache communication is set to "0". Otherwise, it is set to "1".
- Updating frequency (delegated by "U"): This element saves the update frequency of data blocks. PAM defines a threshold for each cache level. If the update frequency of a data block is larger than the corresponding threshold, the value of update frequency is set to 1. Otherwise, it is set to zero. In Table II, we summarize the relationship between update frequency and disk operations. It is clear that update frequency has a significant impact on both I/O performance and energy consumption.

In PAM, to quantitatively measure the elements, we set various priorities for them. Generally, disk status has the highest priority and update frequency has the lowest priority. It is reasonable that PAM trades energy-saving element as the most important for storage devices. In our implementation, we assign one bit for each element to categorize different hotness values, which is organized as a mixed hint. Thus, the hotness of a random block x is calculated by,

$$H(x) = D_x * 2^2 + I_x * 2^1 + U_x \quad (D_x, I_x, U_x \in (0,1)) \quad (1)$$

C. Demotion/Promotion Policy

When a data block is written and the cache are full, PAM has demotion and promotion policies to handle the corresponding dirty blocks.

1) ***Demotion Policy:*** Dirty blocks can be demoted to a lower level only when the cache is full. PAM demote the coldest block instead of the tail block. The related algorithm is shown Algorithm 1.

2) ***Promotion Policy:*** PAM promotes hot dirty blocks to the upper level at a fixed rate. The corresponding Algorithm 1 shows the procedure of promoting a block from L_{i+1} to L_i. At the end of a time window, the hotness H_x of the coldest block in L_i is sent to L_{i+1} to initiate promotion. L_{i+1} will promote the hottest block if its hotness is larger than H_x.

Algorithm 1: Demotion and Promotion Policies in PAM

Demotion Policy:
When a new block is inserted into L_i,
if L_i *is full & the block at the tail of the queue is dirty* **then**
  Find the dirty block with the maximum hotness value (assume Block x).
  Update $H(x)$.
  Demote x from L_i.
end
Promotion Policy:
Find the dirty block x_i with the lowest hotness hotness value in L_i.
Send $H(x_i)$ to L_{i+1}.
Get the largest hotness value in L_{i+1} (assume $H(x_{i+1})$ and the corresponding block is Block x_{i+1})
if $H(x_{i+1}) > H(x_i)$ **then**
  Update $H(x_{i+1})$.
  Promote x_{i+1} from L_{i+1}.
end

D. Write Mechanism

PAM takes a write mechanism at the last level cache to optimize the access pattern of data disks. It captures the historical statistics of read and write operations and select a small portion of dirty blocks to be flushed to data disks, which is decided by a well defined ratio. It is calculated by a positive write count (C) divided the total number of dirty blocks in L_n (N_d),

$$\gamma = \frac{C}{N_d} \quad (2)$$

In a fixed time window, if the sum of all write requests intervals is larger than the related threshold, C is equal to the total number of write requests. After C is set, PAM reorganizes the request queue, and selects C blocks out of N_d dirty blocks in L_n, which are aggregated as a request on a large data block. Then the request is written directly to the disks with the standby/sleep mode. To make PAM energy-efficient, we should trade-off the performance degradation and the reduction on energy consumption. To achieve this goal, PAM sets a threshold, which is the minimal duration time that a disk can maintain in the standby mode. Even if disk mode is switched at an inappropriate time, such as spinning up a disk immediately after spinning down, extra energy consumption is required. Therefore, with a given DPM scheme and a request queue, the threshold (minimum duration time in the standby mode for a disk) is decided by the parameters of a disk and update frequency , which can be computed by the following inequality,

$$\begin{aligned} & (T_s + T_{down} + T_{up}) * P_{dl} \\ \geq \quad & T_s * P_s + T_{down} * P_{down} + T_{up} * P_{up} \end{aligned} \quad (3)$$

The threshold (minimum duration time of a disk in the standby mode) can be calculated by Equation 3,

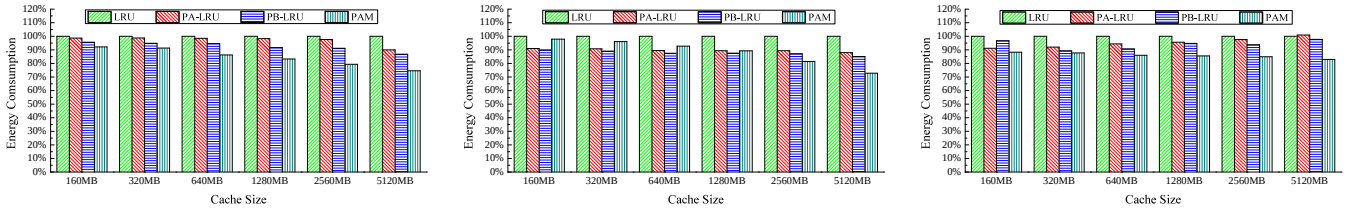
$$T_{min-s} \quad = \quad \frac{(P_s - P_{dl}) * T_s + (P_{up} - P_{dl}) * T_{up}}{P_{dl} - P_s} \quad (4)$$

IV. Simulation Methodology And Analysis

To verify the effectiveness of PAM, we use trace driven simulation to evaluate the PAM scheme and compare it with several popular cache approaches, such as LRU [5], PA-LRU [35], PB-LRU [35], Demote [27] and ARC [18].

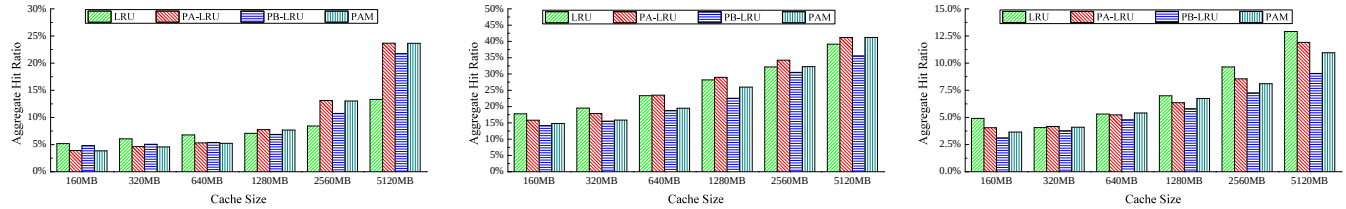
A. Simulation Methodology

We use a modified *fscachesim* as the simulator to evaluate various cache approaches, which appears in several literatures [27], [28]. The length of profiling period is set to 10 percent of the runtime module. The specification of the data disks in our simulation is similar to the IBM Ultrastar 36Z15 [1], which is shown in Table III. SSD is excluded in our simulation,,



(a) *MSN-SFS* (b) *Microsoft Exchange* (c) *TPC-C*

Fig. 2. Energy consumption under different traces with different aggregate cache sizes (Energy consumption of LRU is normalized to 100%)



(a) *MSN-SFS* (b) *Microsoft Exchange* (c) *TPC-C*

Fig. 3. Aggregate hit ratio under different traces with different aggregate cache sizes.

TABLE III
TYPICAL DISK PARAMETERS (IBM ULTRASTAR 36Z15)

Description	Values
Active Power (Read/Write/Seek)	13.5W
Idle Power	10.2W
Standby Power	2.5W
Spin-up Power	12.39W
Spin-up Time	10.9s
Spin-down Power	8.67W
Spin-down Time	1.5s
Standard Interface	SCSI
Disk Rotation Speed	15000RPM
Disk Average Seek time	3.4ms
Disk Average Rotate time	2.0ms
Disk Transfer Rate	55MB/s

because it performs better due to lower power off/up delay and lower energy consuming in state switching. So simulations with typical hard disk drives can illustrate the robustness of PAM.

(1) ***MSN Storage File System (MSN-SFS)***: The traces were collected for MSN Storage file server for a duration of six hours, which include thirty-six 10-minute trace files. They trace the primarily disk I/O events at block level as well as file I/O events.

(2) ***Microsoft Exchange***: The Microsoft Exchange traces are collected from an Exchange 2007 SP1 server, which is a mail server for 5,000 corporate users.

(3) ***TPC-C***: The traces were collected at a server running TPC-C benchmark.

Unlike the simulation of other cache algorithms which have to wait for enough blocks flooding into all cache levels, PAM ignores the warm up time in *fscachesim*.

Unless otherwise mentioned, the following default parameters are used: two cache levels ($n = 2$) and block size 4KB. Default ratio of cache size between an upper cache level and the next lower level is $1 : 4$. Aggregate cache size is the sum of all cache level sizes. Based on the default settings, the average access time of L1 cache, L2 cache and disks are 0.2ms, 2ms and 10ms, respectively.

B. Numerical Results

In this section, we give the numerical results of energy efficiency using different cache algorithms.

1) Energy Consumption The first experiment is to compare the energy consumption of PAM to other algorithms under different traces. Typical multi-level cache algorithms don't contain any power-aware policy, so they are excluded in our comparison. We use the energy consumption of the LRU replacement algorithm as the baseline (100%). In some write intensive traces, such as *TPC-C* (in Figure 2(c)) and *Microsoft Exchange* (in Figure 2(b)), compared to the power-aware cache algorithms (PA-LRU and PB-LRU), PAM saves the energy consumption by up to 15%.

2) Cache Performance Next, we measure the cache performance under different workloads. In this experiment, we use two traditional metrics (*Aggregate Hit ratio* and *Average Response Time*) to evaluate the I/O performance. Because PA-LRU and PB-LRU have lower performance than typical LRU algorithm [35], [36], they are not included in our comparison. Nowadays most multi-level cache algorithms are based on demote hints [27], [4], [33], [28], so we select Demote algorithm [27] in our comparison[2].

The results of aggregate hit ratio are shown as Figure 3. We can see that the aggregate hit ratio of PAM is acceptable. In

[2]LRU and ARC cache policies can be applied with demote algorithm, and we use "LRU+Demote" and "ARC+Demote" to delegate them, respectively.

many cases, PAM has slight performance degradation (by up to 2.2%) compared to other popular performance-oriented cache algorithms. This is reasonable because some cache space are used to store the metadata of flush write blocks.

C. Analysis

From the results in previous section, compared to LRU, PA-LRU, PB-LRU and Demote, it is clear that PAM has many advantages on power reduction and power efficiency. There are several reasons to achieve these gains. First, PAM is a power-aware policy based on the quantitatively analysis of data disks, which selects a proper number of dirty blocks in the upper level cache to flush to the storage devices. It can sharply reduce the power consumption of data disks. Second, PAM is a global cache management scheme considering both read and write requests. On one hand, the write requests are flushed to save energy. On the other hand, PAM retains demote hints to process read requests, which can maintain high performance. Therefore, PAM achieve high energy efficiency.

V. Conclusions

In this paper, we propose a novel multi-level cache algorithm, which can sharply decrease the energy consumption via extending the period of standby mode of data disks. In our simulation, compared to PA-LRU and PB-LRU, PAM saves the power consumption by up to 15% while the performance is still competitive with the existing popular cache schemes such as LRU and Demote under different I/O workloads.

VI. Acknowledgement

This work is partially sponsored by the National Basic Research 973 Program of China (No. 2015CB352403), the National Natural Science Foundation of China (NSFC) (No. 61261160502, No. 61272099, No. 61303012, No.61332001), the Program for Changjiang Scholars and Innovative Research Team in University (IRT1158, PCSIRT), the Scientific Innovation Act of STCSM (No. 13511504200), the Shanghai Natural Science Foundation (No. 13ZR1421900), the Scientific Research Foundation for the Returned Overseas Chinese Scholars, and the EU FP7 CLIMBER project (No. PIRSES-GA-2012-318939).

References

[1] Hitachi, ultrastar 36z15 datasheet. http://www1.hitachigst.com/hdd/ultra/ul36z15.htm, 2011.

[2] L. Cai and Y. Lu. Joint power management of memory and disk. In *Proc. of the DATE'05*, Munich, Germany, 2005.

[3] Y. Chai and W. Fan. Three-state disk model for high quality and energy efficient streaming media servers. In *Proc. of the IEEE ISADS'13*, Beijing, China, 2013.

[4] Z. Chen, Y. Zhou, and K. Li. Eviction based cache placement for storage caches. In *Proc. of the USENIX ATC'03*, San Antonio, TX, 2003.

[5] P. Denning. The working set model for program behavior. *Communications of the ACM*, 11(5):323–333, 1968.

[6] B. Diniz, D. Neto, et al. Limiting the power consumption of main memory. In *Proc. of the ISCA'07*, San Diego, CA, 2007.

[7] F. Douglis, P. Krishnan, and B. Marsh. Thwarting the power-hungry disk. In *Proc. of the Winter USENIX'94*, San Francisco, CA, 1994.

[8] B. Gill. On multi-level exclusive caching: Offline optimality and why promotions are better than demotions. In *Proc. of the USENIX FAST'08*, San Jose, CA, 2008.

[9] M. Gupta and S. Singh. Greening of the internet. In *in Proceedings of ACM SIGCOMM*, pages 19–26, August 2003.

[10] S. Gurumurthi, A. Sivasubramaniam, et al. DRPM: Dynamic speed control for power management in server class disks. In *Proc. of the ISCA'03*, San Diego, CA, 2003.

[11] D. Helmbold, D. Long, et al. Adaptive disk spin-down for mobile computers. *Mobile Networks and Applications*, pages 285–297, 2000.

[12] Huang et al. An analysis of facebook photo caching. SOSP '13, pages 167–181.

[13] S. Irani, S. Shukla, and R. Gupta. Competitive analysis of dynamic power management strategies for systems with multiple power saving states. In *Proc. of the DATE'02*, Paris, France, 2002.

[14] S. Jiang and X. Zhang. ULC: A file block placement and replacement protocol to effectively exploit hierarchical locality in multi-level buffer caches. In *Proc. of the ICDCS'04*, Tokyo, Japan, 2004.

[15] X. Li, Z. Li, et al. Performance directed energy management for main memory and disks. In *Proc. of the ASPLOS'04*, Boston, MA, 2004.

[16] X. Liu, A. Aboulnaga, et al. CLIC: Client-informed caching for storage servers. In *Proc. of the USENIX FAST'09*, San Francisco, CA, 2009.

[17] Y. Lu and G. Micheli. Comparing system-level power management policies. *IEEE Design & Test*, 18(2):10–19, 2001.

[18] N. Megiddo and D. Modha. ARC: A self-tuning, low overhead replacement cache. In *Proc. of the USENIX FAST'03*, San Franciso, CA, 2003.

[19] K. Panos, H. Perros, et al. Sdn-based solutions for moving target defense network protection. 2014.

[20] L. Parolini, B. Sinopoli, and B. Krogh. Reducing data center energy consumption via coordinated cooling and load management. In *Proc. of the HotPower'08*, San Diego, CA, 2008.

[21] R. Patterson, G. Gibson, et al. Informed prefetching and caching. In *Proc. of the ACM SOSP'95*, Cooper Mountain, CO, 1995.

[22] M. Rossi, N. Bui, et al. Synapse++: Code dissemination in wireless sensor networks using fountain codes. volume 9, pages 1749–1765, Dec 2010.

[23] W. Rui, J. Zhipeng, et al. Energy-aware routing algorithms in software-defined networks. In *WoWMoM'14*, pages 1–6, June 2014.

[24] P. Sarkar and J. Hartman. Hint-based cooperative caching. *ACM Transactions on Computer Systems*, 18(4):387–419, 2000.

[25] Minseok Song et al. Saving disk energy in video servers by combining caching and prefetching. *TOMCCAP*, 10(1s):15, 2014.

[26] M. Tolentino, J. Turner, and K. Cameron. Memory-miser: a performance-constrained runtime system for power-scalable clusters. In *Proc. of the ACM CF'07*, Ischia, Italy, 2007.

[27] T. Wong and J. Wilkes. My cache or yours? Making storage more exclusive. In *Proc. of the USENIX ATC'02*, Monterey, CA, 2002.

[28] C. Wu, X. He, et al. Hint-K: An efficient multi-level cache using k-step hints. http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.49.

[29] G. Yadgar et al. Karma: Know-it-all replacement for a multilevel cache. In *Proc. of the USENIX FAST'07*, San Jose, CA, 2007.

[30] G. Yadgar, M. Factor, et al. MC^2: Multiple clients on a multilevel cache. In *Proc. of the ICDCS'08*, Beijing, China, 2008.

[31] G. Yadgar, M. Factor, et al. Management of multilevel, multiclient cache hierarchies with application hints. *ACM Transactions on Computer Systems*, 29(2):Article 5, 2011.

[32] J. Yue, Y. Zhu, and Z. Cai. An energy-oriented evaluation of buffer cache algorithms using parallel I/O workloads. *IEEE Transactions on Parallel and Distributed Systems*, 19(11):1565–1578, 2008.

[33] Y. Zhou, Z. Chen, and K. Li. Second-level buffer cache management. *IEEE Trans. on Parallel and Distributed Sys.*, 15(6):505–519, 2004.

[34] Y. Zhou et al. The multi-queue replacement algorithm for second level buffer caches. In *Proc. of the USENIX ATC'01*, Boston, MA, 2001.

[35] Q. Zhu, F. David, et al. Reducing energy consumption of disk storage using power-aware cache management. In *Proc. of the HPCA'04*, Madrid, Spain, 2004.

[36] Q. Zhu, A. Shankar, and Y. Zhou. PB-LRU: a self-tuning power aware storage cache replacement algorithm for conserving disk energy. In *Proc. of the ICS'04*, Saint Malo, France, 2004.