Impacts of Indirect Blocks on Buffer Cache Energy Efficiency

Jianhui Yue, Yifeng Zhu*, Zhao Cai
University of Maine
Department of Electrical and Computer Engineering
Orono, USA
{ jyue, zhu*, zcai}@eece.maine.edu

Abstract

Indirect blocks, part of a file's metadata used for locating this file's data blocks, are typically treated indistinguishably from file's data blocks in buffer cache. This paper shows that this conventional approach will significantly detriment the overall energy efficiency of memory systems. Scattering small but frequently accessed indirected blocks over all memory chips reduce the energy saving opportunities. We propose a new energy-efficient buffer cache management scheme, named MEEP, which separates indirect and data blocks into different memory chips. Our trace-driven simulation results show that our new scheme can save memory energy up to 16.8% and 15.4% in the I/O-intensive server workloads TPC-R and TPC-H, respectively.

1 Introduction

As the computing capacity increases rapidly in large-scale cluster computing platforms, power management becomes an increasingly important concern. For example, the power density of Google clusters with low-tech commodity PCs exceeds 700 W/ft^2, while the cooling capability in typical data servers lies between 70 and 120 W/ft^2 [2], [17]. A large power consumption in a cluster not only increases its running cost, but also raises its components' temperature through rapid heat dissipation, accordingly reducing the reliability and increasing the maintenance cost. The recent trend towards very-large-scale clusters, with tens of thousands of nodes [15], will only exacerbate the power consumption issue.

Many previous studies [11], [12], [18] have shown that main memory is one of major sources of power consumption. The energy breakdown measured on a real server shows that the memory consumes 41% of the total energy and is 50% more than the processors [12]. As the memory capacity continues to increase rapidly in order to bridge the ever-widening gap between disk and processor speeds, memory energy efficiency becomes an increasingly important concern.

Specifically, the paper makes the following two contributions. Our previous study [21] indicates that the replacement algorithm's ability to capture temporal locality of data blocks is one of important factors to influence buffer cache energy consumption behavior. In this study we investigate the impacts of indirect blocks on the memory energy. In a file system, a indirect block contains pointers to file data blocks or to another indirect block. Through this hierarchical structure, disk storage space can be organized into logic units, i.e., files. Thus, there are two types of buffer cache traffic: data blocks and indirect blocks. However, from the traces collected on real-world servers, we find that the total working set and the total traffic volume of indirect blocks can be 7.62%% and 18.6% of data blocks, respectively. In spite of generating such large volume of traffic, indirect blocks have received few attentions in the studies of memory energy optimization.

Specifically, the paper makes the following contributions.

- Through simulations based on three I/O-intensive server traces, we find that indirect block traffic can lead to inferior memory energy efficiency. Compared with data blocks, indirect blocks are relatively smaller but are accessed much more frequently. In conventional management scheme in most operating systems, data blocks and indirect blocks are placed interleaved in the buffer cache. The large volume of indirect accesses results in fewer energy saving opportunities through memory chip powerdown and DMA overlapping.

- This paper proposes a new energy-efficient buffer management scheme named MEEP. The key idea of MEEP is to separate indirect and data blocks and place them into disjointed sets of memory chips. Our simulation results show that with only one memory chip dedicated for storing indirect blocks, the buffer cache can have

16.8% memory energy saving, with little degradation in hit rates.

The rest of the paper is organized as follows. Section 2 briefly describes the background, including file indirect blocks, power-aware memory chips, and DMA transfers. Section 3 describes our new energy-efficient buffer management placement scheme MEEP. Section 4 presents our evaluation methodology and simulation results. Section 5 discusses prior related work. Section 6 concludes the paper.

2 Background

2.1 File Indirect Block

File systems use indirect blocks to organize disk sectors into logic files. The addresses of data blocks of a file are stored in the inode or the indirect blocks of this file. In order to efficiently utilize disk space, all indirect blocks of a file are organized in a multi-level hierarchical tree as illustrated in Fig. 1. In this paper, we assume that the level of a child indirect block is higher than its parent indirect block. While block addresses in the first level are directly stored in the inode itself, addresses in the other levels are organized into blocks and stored on disk in a similar way as file data blocks. In order to obtain a block's disk address, one or more related indirect block(s) are needed to access. For example, in the worst case, four indirect blocks may be accessed in many Linux file systems.

In modern operating systems, indirect blocks share buffer cache with data blocks. During a cache miss for an indirect block, the requested indirect block will be fetched from the disk and then placed in the buffer cache. When a read miss happens to a data block, the corresponding indirect blocks needs to be accessed first to acquire the disk addresses of the missed data block.

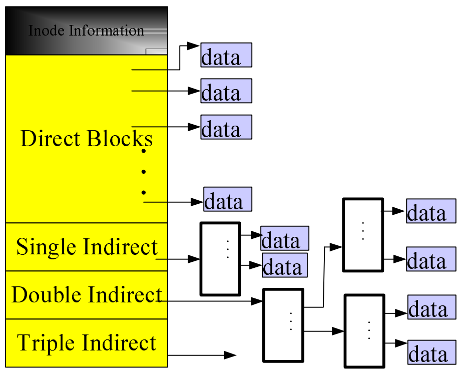


Figure 1. File Data Indirect Block.

Table 1. Power states and transition delay of a RDRAM chip

Power State/Transition	Power (mW)	Delay
Active	300	-
Standby	180	-
Nap	30	-
Powerdown	3	-
Active → Standby	240	1 memory cycle
Active → Nap	160	8 memory cycles
Active → Powerdown	15	8 memory cycles
Standby → Active	240	+6 ns
Nap → Active	160	+60 ns
Powerdown → Active	15	+6000 ns
Standby → Nap	160	+4 ns
Nap → Powerdown	15	~0 ns

2.2 RDRAM Memory Chips

In the RDRAM technology, each memory chip can be independently set to a proper state: active, nap, standby and powerdown. In the active state, a chip can perform reading or writing and consumes full power. In the other states, the chip powers off different components to conserve energy. In these states, the chip can not service any read/write requests before it becomes active. The transition from a lower power state to a higher one requires some time delay. Table 1 summarizes the power consumption rate of each state and the time delay needed to transition among these states.

There are two classes of techniques to control the power state of a memory chip: static and dynamic. Static techniques always set a chip to a fixed low-power state. The chip is transitioned back to full-power state only when it needs to service a request. After the request is serviced, the chip immediately goes back to the original state, unless there is another request waiting. In contrast, dynamic techniques change current power state to the next lower power state only after being idle for a threshold amount of time. The thresholds are dynamically adjusted according to the variation of memory I/O workload. In this paper, we focus on dynamic techniques in our energy evaluation.

2.3 Network and Disk DMA Operations

Direct Memory Access (DMA) has been widely used to transfer data blocks between main memory and I/O devices including disks and network. Fig. 2 gives an example of disk-network datapath for two cache misses A and B, following steps from 0 to 3. When a read request arrives through a network interface (NIC), the server first performs data address translation and then checks whether desired data blocks are stored in the main-memory buffer cache. If they are cached, the host processor on the storage server initiates a network DMA operation to transfer the data out directly from the main memory through NIC. If they are not, the processor first performs a disk DMA transfer to copy the data from disks to the main-memory buffer cache, and then the processor conducts a network DMA transfer to send the data out. For write requests, the datapaths are similar but flow in the reverse direction.

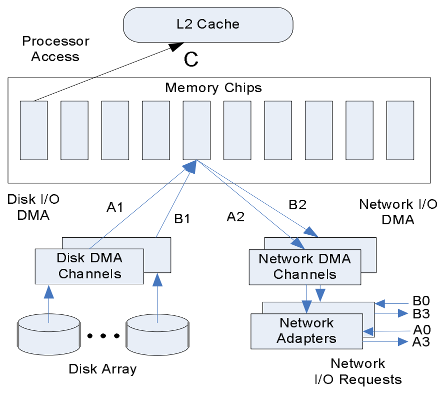


Figure 2. I/O path for two cache read misses in A and B typical storage server, following steps from 0 to 3. C is a cache line access from CPU.

On a storage server, recent DMA controllers, such as Intel's chipset E8870 and E7500 [9], allow multiple DMA transfers on different buses to access the same memory module simultaneously in a time multiplexing fashion. Typically, the peak transfer rate of a memory chip can be a multiple factor of the bandwidth of the PCI bus. For example, the transfer rate of most recent RDRAM chips [8] and DDR SDRAM are up to 3.2GB/s and 2.1GB/s respectively, while a typical PCI-X bus only gives a maximum rate of 1.064GB/s and the second-generation SATA disk DMA throughput is only 300 MB/s.

Multiplexing various slow disk and network I/Os to the same memory chip can reduce the waste of active memory cycles and hence save memory energy. Most DMAs move a large amount of data, usually containing multiple 512-byte disk sectors or 4-KByte memory pages. Without multiplexing, a memory chip is periodically touched during a DMA transfer and such access period is too short to justify the transition to a low-power mode [11], [13], [18]. As a result, significant amount of active energy is wasted. However, when DMAs on different I/O buses are coordinated to access the same memory chip, such energy waste can be reduced. For example, when the concurrent requests A and B in Fig. 2 are directed to the same memory chip, the DMA transfers $A1$ and $B1$ can overlap with each other in time and accordingly one of them takes a "free ride" and consumes zero energy, without causing any performance penalty. Similarly, $A2$ and $B2$ can also overlap with each other.

As mentioned previously, the processor accesses indirect block from memory with size of L2 cache line. For example, the transfer C from memory to processor is response to the indirect block request illustrated in Fig. 2. In the context of data server, the size of processes images are much smaller than size of file data working set and they can be stored at very smaller set of memory chips. As result, energy energy incured by processes images accesses is ingored in our paper, which is similarly treated in Ref. [18].

3 Memory Energy Efficient Placement Scheme

Placing the data blocks with temporal locality at one memory chip is a widely used approach to achieve memory energy saving. The missed data block cause corresponding indirect blocks accesses. In the case of large file, the missed data block with large LBN incurs up to four indirect blocks at different levels. Such indirect blocks have very strong temporal locality. However, they often reside at different memory chips and activate extra memory chips, resulting more energy consumption. This is because that the indirect blocks at difference levels are spread over several chips due to the differences in their access frequencies.

In order to address the above issue, we propose a new data placement scheme. In this scheme, indirect blocks are artificially assigned a smaller set of memory chips, one or two. In this way, the indirect block traffic is limited to a smaller set of memory chips. In order to limit the indirect block traffic to a smaller set of memory chips, the block allocation method before cache full and eviction method are needed to be modified. The function *getFree-Block_Chip(type)* selects a free block from a specific memory chip set determined by parameter type and thus keeps blocks residing at related chips sets. Additionally, two LRU stacks are needed and each type of block has its independent LUR stack. When replacement happens, the victim block is selected from the bottom of related LUR stack and thus the property of blocks residing at related chips sets is maintained. We refer this algorithm as MEEP.

4 Energy Evaluation

This section compares the energy consumption of three schemes: MEEP, conventional buffer cache with only data block traffic, and conventional buffer cache with both data block and indirect block traffic.

Algorithm 1 MEEP Algorithm

```
$LBN$ is the block id of missed block
$type$ is either data block or indirect block
$stacks$ data block or indirect block Stacks
$freeBlk$ the number of free data block or free indirect blocks
$entry$ block entry in the stack
**if** a cache miss occurs **then**
  **if** $freeBlks[type]$ **then**
    $entry \Leftarrow newEntry()$
    { allocate block from corresponding chips}
    $entry.chip \Leftarrow getFreeBlock\_Chip(type)$
    $entry.lbn \Leftarrow LBN$
    $freeBlks[type] - -$
  **else**
    { evict block from corresponding chips}
    $entry \Leftarrow getBottomEntry(stacks[type])$
    $entry.lbn \Leftarrow LBN$
  **end if**
  return $entry$
**else**
  $entry \Leftarrow getEntry(stacks[type], LBN)$
  putTop( $stacks[type], entry)$
  return $entry$
**end if**
```

4.1 Traces

In our experiments, we chose two real-life data server traces: *TPC-R* and *TPC-H* [1]. Both *TPC-R* and *TPC-H* are transaction processing work load. Fig. 3 shows data block and indirect block virtual address diagrams as a function of the virtual time that is defined as the number of references issued so far and is incremented for each request. Since the indirect block traffic is affected by the data blocks hit rate, we collect indirect block traces with specified size of buffer cache 512MB and 512MB respectively for *TPC-R*, *TPC-H*. In order to make these diagrams readable, the traces are plotted with a sampling rate of 1000 for data blocks and 100 for indirect blocks. In the term of working set, the ratio of indirect blocks to data blocks are 7.57% and 7.62% respectively for *TPC-R* and *TPC-H*. In the term of traffic volume, the ratios of indirect blocks to data blocks are 18.6% and 18.1% for *TPC-R* and *TPC-H*, respectively. The above statistical results further show that the impact of indirect data blocks on the energy efficiency of buffer cache can not be ignored, which in fact motivates our study in this paper.

4.2 Simulation Environment

We have developed a detailed trace-driven simulator that faithfully emulates network DMA and disk DMA operations and accurately records energy consumed by each memory chip. On storage servers, both network and disk DMAs are heavily involved in most cases. Through disk DMAs, data missed in the cache or dirty blocks are exchanged between memory chips and disk drives. Through network DMAs, requested data are sent to clients from the memory through network interfaces. With new emerging technology, multiple DMA requests from different bus channels but targeting to the same chip can be simultaneously serviced in a multiplexing way. The data sever simulated in this paper is configured with 6 network adaptors and 12 disks. Each device, either disk or network adaptor, is attached to one 133 MHz PCI-X bus. A DMA request is performed on the corresponding PCI-X bus whose device is the source or destination. Disksim [3], a well validated disk array simulator, is incorporated into our simulator to precisely emulate the timing of I/O traffic.

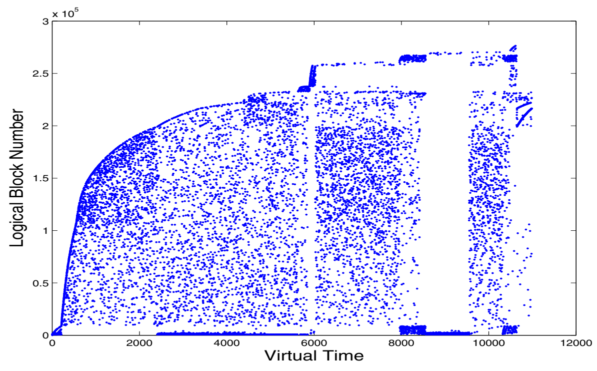
The simulator emulates RDRAM memory chips, whose parameters are given in Table 1. Each chip has a capacity of 32MB and a peak performance of 3.2GB/second. The simulator precisely models power state transitions, DMA contentions and queuing processes. The default sector size is 512 Bytes, data block pointer size is 4 bytes and L2 cache line size is 128 bytes. In the MEEP algorithm, our experiment results show that MEEP energy consumption is not sensitive to variation of size of indirect blocks memory chips set and we only present data results with one memory chip.

We simulate the traces by replaying all I/O events at predetermined times specified in the traces, independent of the performance of memory hierarchy. This approach is used mainly because all traces that we have access to do not record the dependence among request completion and subsequent I/O arrivals.

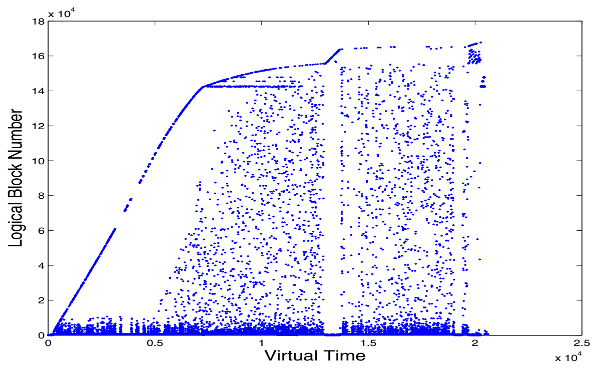
In the following energy efficiency evaluation, the *conventional method* represent the scheme in which data blocks and indirect blocks are placed in buffer cache interchangeably. This method is widely used in most modern operating systems. In order to investigate the impact of indirect blocks, we present the buffer cache energy consumption under only data block traffic and compare it against the case with only data blocks. We also evaluate MEEP's energy consumption, indirect block memory chip energy consumption and data block memory chips energy consumption. Lastly we present the indirect blocks hit rate and data blocks hit rate.

4.3 *TPC-R* Workload

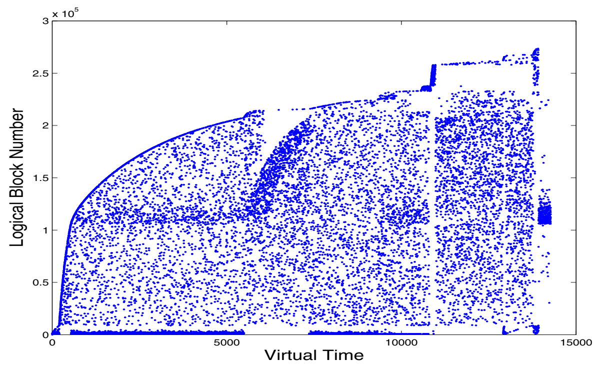
We have the following observations. Firstly, the energy gap is large between the following two workloads (1) only data blocks and (2) combined indirect and data blocks. For example, at cache size of 128MB the difference is up to 37J



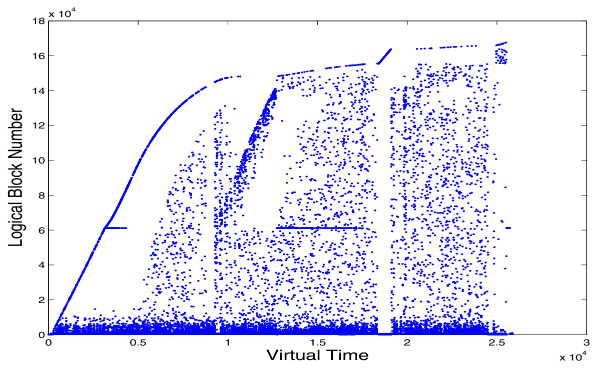
(a) *TPC-R* Data Block Trace (sampling period: 1000)



(b) *TPC-R* Indirect Block Trace (sampling period: 100)



(c) *TPC-H* Data Block Trace (sampling period: 1000)



(d) *TPC-H* Indirect Block (sampling period: 100)

Figure 3. Trace Data Block and Indirect Block Traffic.

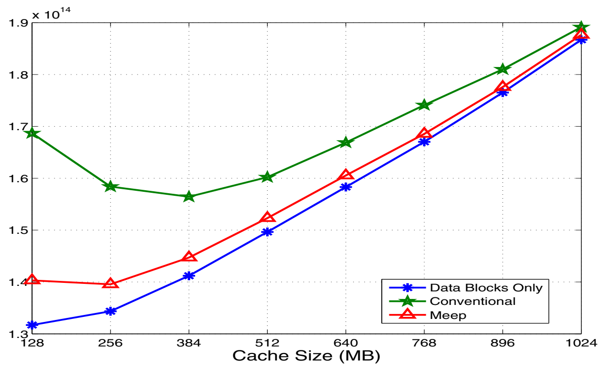
and the average difference is 13.68J with the corresponding rated values of 29.1% and 8.8%. Hence, the indirect block traffic can not be ignored in designing energy efficient buffer cache. Secondly, MEEP can greatly reduce buffer cache energy consumption. In Fig. 4(a), the energy difference between MEEP and the conventional method can reach 28.4J, with an average energy saving 10.6J. The corresponding rated values are 16.8% and 6.28%. We also observe that MEEP's energy saving decreases when the cache size increases. This is because when the cache is large enough to hold almost all data in buffer cache and replacement seldom happens, a data block and its related indirect blocks most likely reside in the memory chip. For example, at the cache size of 1GB, in Fig. 4(b), the conventional method hit rate approaches 95%. In Fig. 5(c) and Fig. 4(d), we compare their energy and hit rate. The indirect blocks' memory consumption is almost independent with cache size changes. This is because a fixed number of memory chips are used for indirect blocks. It is also noticed that indirect block memory energy is not proportional to its working set. This is because the total access to indirect block is one L2 cache line and one disk sector during a cache hit and a cache miss,respectively. Both are smaller than the size of a data block. However, the hit rate shows a different trend. When the size of data block cache is increased, the data hit rate is increased, resulting in less cache misses. Accordingly the indirect block traffic is also reduced and this potentially leads to the hit rate decrease for indirect blocks.

4.4 *TPC-H* Workload

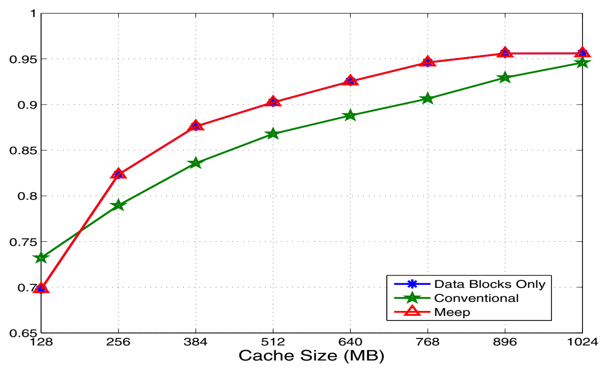
The results of *TPC-H* are very similar to *TPC-R* workload. Firstly, at the cache size of 128MB, the difference of energy consumption between the I/O traffic with and without indirect blocks can reach up to 42J, with an average of 16J . Secondly, in Fig. 5(a), the energy difference between MEEP and the conventional method is up to 32 J with an average energy saving 12.2J. The rated average energy saving of MEEP is 15.4%.

5 Related Work

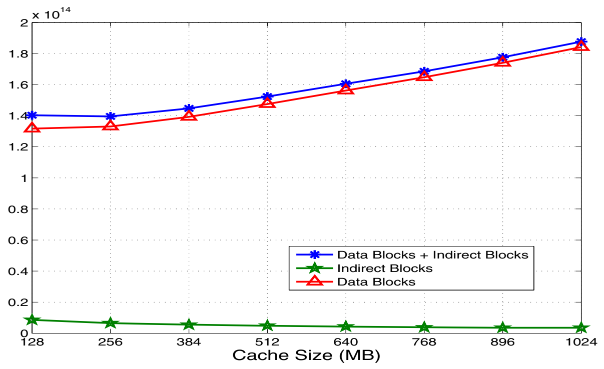
Power consumption has been an issue primarily in embedded or portable computer systems. Until recently, energy efficiency is becoming an increasingly important concern in high-end servers. On individual servers, many studies have been conducted to save memory energy. The most important principle to conserve the memory energy is to minimize the number of simultaneously accessed memory chips. In order to achieve it, previous works [11, 7, 10, 16]



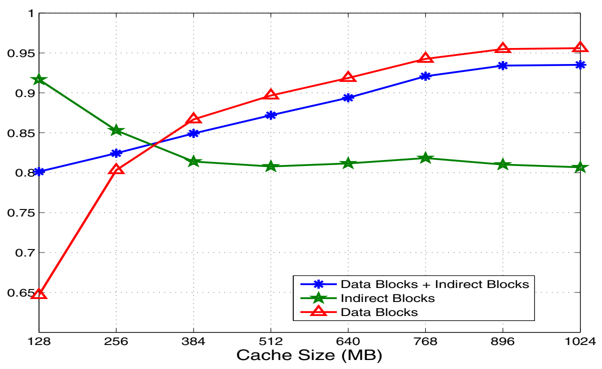
(a) Energy (unit ns*mW)



(b) Hit Rate



(c) Energy Decompositon (unit ns*mW)



(d) Hit Rate Decomposition

Figure 4. Workload *TPC-R*.

propose to place data blocks with temporal locality at the same memory chip by exploiting data block semantics such as process, file and database table. Ref. [11, 18, 19, 16, 14] proposes to dynamically migrate hot data blocks to a smaller set of memory chips. Typically these approaches unavoidably pay fairly large performance and energy overhead during both the bookmarking process for identifying hot blocks and the migration course.

Ref. [5, 7, 20, 6] propose to adaptively control the memory power states, instead of relying on reactive threshold mechanisms. Ref. [13, 4] aim to optimize the overall energy efficiency of both memory chips and disk drives. While these research work is designed for virtual memory, very little has been done for buffer cache. Ref. [18] proposes two schemes to save energy in data servers: temporally aligning DAM transfers to the same memory chips through buffering and migrating data among chips to minimize the number of active chips. Ref. [22] proposes a new buffer cache replacement algorithm to reduce the disk energy consumption.

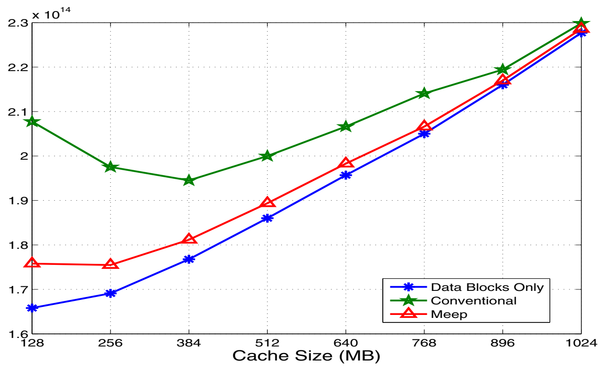
6 Conclusion

Indirect blocks, used for locating file data blocks on disks, are equally treated in buffer cache in conventional buffer management systems in many modern operating systems. Our research results show this conventional scheme, placing indirect blocks and data blocks interchangeably into the same set of memory chips, is detrimental to the memory energy efficiency. When the indirect blocks is scattered all over the memory chips, these memory chips then have little chance for entering lower-power modes and for DMA overlapping, since the access frequencies of indirect blocks are typically much higher data blocks.

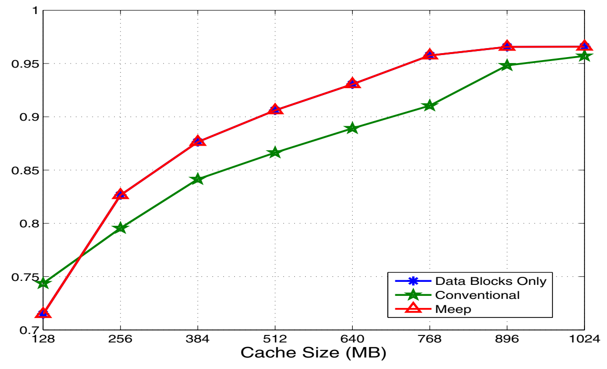
In this paper, we propose a new buffer management scheme, named MEEP, which clusters indirect and data blocks into separate sets of memory chips. Our simulation results indicate that using one dedicated memory chip for indirect blocks can result in 16.8% of energy saving when compared against the conventional buffer management scheme. Our immediate future work is to implement MEEP in Linux systems and measure the energy efficiency by running real application.

Acknowledgements

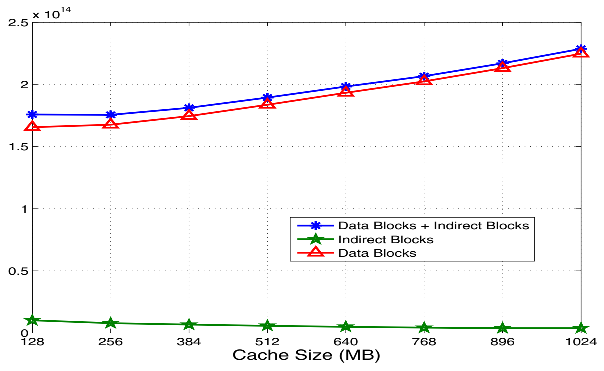
This work is supported by a UMaine Startup Grant, NSF EPS-0091900, NSF CCF-0621526/0621493, NSF CNS 0723093, NSF CNS 0619430, NASA Maine Space Grant and a Chinese NSF 973 Project Grant (No. 2004cb318201), and equipment donations from SUN.



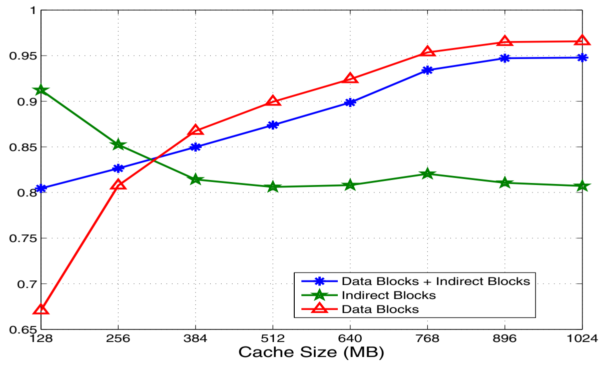
(a) Energy (unit ns*mW)



(b) Hit Rate



(c) Energy Decompositon(unit ns*mW)



(d) Hit Rate Decomposition

Figure 5. Workload *TPC-H*.

References

[1] C. G. Ali R. Butt and Y. C. Hu. The performance impact of kernel prefetching on buffer cache replacement algorithms. In *Proceedings of the ACM International Conference on Measurement & Modeling of Computer Systems (SIGMETRICS)*, Banff, Canada, June 2005.

[2] L. A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The Google cluster architecture. *IEEE Micro*, 23(2):22–28, 2003.

[3] J. S. Bucy, G. R. Ganger, and et al. The disksim simulation environment version 3.0 reference manual. `http://www.pdl.cmu.edu/DiskSim`.

[4] L. Cai and Y.-H. Lu. Joint power management of memory and disk. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 86–91, Washington, DC, USA, 2005. IEEE Computer Society.

[5] V. Delaluz, A. Sivasubramaniam, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. Scheduler-based dram energy management. In *DAC '02: Proceedings of the 39th conference on Design automation*, pages 697–702, New York, NY, USA, 2002. ACM Press.

[6] B. Diniz, D. Guedes, W. M. Jr., and R. Bianchini. Limiting the power consumption of main memory. In *Proceedings of the International Symposium on Computer Architecture ISCA*, pages 290–301. ACM Press, June 2007.

[7] H. Huang, P. Pillai, and K. G. Shin. Design and implementation of power-aware virtual memory. In *USENIX Annual Technical Conference*, pages 57–70, 2003.

[8] R. Inc. Rambus memory chips. `http://www.rambus.com`.

[9] Intel. Server and workstation chipsets. `http://www.intel.com/products/server/chipsets/`.

[10] A. C. Jayaprakash Pisharath and M. Kandemir. Energy management schemes for memory-resident database systems. In *ACM Thirteenth Conference on Information and Knowledge Management (CIKM'05)*, Washington, DC, USA, Nov 2004. ACM Press.

[11] A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis. Power aware page allocation. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 105–116, New York, NY, USA, 2000. ACM Press.

[12] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller. Energy management for commercial servers. *Computer*, 36(12):39–48, 2003.

[13] X. Li, Z. Li, Y. Zhou, and S. Adve. Performance directed energy management for main memory and disks. *Trans. Storage*, 1(3):346–380, 2005.

[14] V. D. L. Luz, M. Kandemir, and I. Kolcu. Automatic data migration for reducing energy consumption in multi-bank memory systems. In *DAC '02: Proceedings of the 39th conference on Design automation*, pages 213–218, New York, NY, USA, 2002. ACM Press.

[15] H. Meuer, E. Strohmaier, J. Dongarra, and H. D. Simon. Top 500 supercomputers. Website, 2005. `http://www.top500.org`.

[16] J. L. Min Lee, Euiseong Seo and J. Kim. Pabc: Power-aware buffer cache management for low power consumption. *IEEE Transactions on Computers*, 56(4), 2007.

[17] B. Moore. Take the data center power and cooling challenge. Energy User News, Aug. 2002.

[18] V. Pandey, W. Jiang, Y. Zhou, and R. Bianchini. DMA-aware memory energy management for data servers. In *The Proceedings of the 10th International Symposium on High-Performance Computer Architecture (HPCA'06)*, 2006.

[19] P. Ramamurthy and R. Palaniappan. Performance-directed energy management using bos. *SIGOPS Oper. Syst. Rev.*, 41(1):66–77, 2007.

[20] M. E. Tolentino, J. Turner, and K. W. Cameron. Memory-miser: a performance-constrained runtime system for power-scalable clusters. In *CF '07: Proceedings of the 4th international conference on Computing frontiers*, pages 237–246, New York, NY, USA, 2007. ACM Press.

[21] J. Yue, Y. Zhu, and Z. Cai. Evaluating memory energy efficiency in parallel i/o workloads. In *Proceedings of 2007 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 21 – 30, Austin, TX, USA, Sept. 2007.

[22] Q. Zhu and Y. Zhou. Power aware storage cache management. *IEEE Transactions on Computers*, 54(5):587–602, May 2005.