Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Exploring optimal combination of a file system and an I/O scheduler for underlying solid state disks*

Hui SUN[1], Xiao QIN[2], Chang-sheng XIE‡[1]

([1]*National Laboratory for Optoelectronics and School of Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*)

([2]*Department of Computer Science and Software Engineering, Auburn University, Auburn, AL 36849, USA*)

E-mail: sunhuiworking@gmail.com; xqin@auburn.edu; cs_xie@hust.edu.cn

Received Oct. 29, 2013; Revision accepted Apr. 2, 2014; Crosschecked July 16, 2014

Abstract: Performance and energy consumption of a solid state disk (SSD) highly depend on file systems and I/O schedulers in operating systems. To find an optimal combination of a file system and an I/O scheduler for SSDs, we use a metric called the aggregative indicator (AI), which is the ratio of SSD performance value (e.g., data transfer rate in MB/s or throughput in IOPS) to that of energy consumption for an SSD. This metric aims to evaluate SSD performance per energy consumption and to study the SSD which delivers high performance at low energy consumption in a combination of a file system and an I/O scheduler. We also propose a metric called Cemp to study the changes of energy consumption and mean performance for an Intel SSD (SSD-I) when it provides the largest AI, lowest power, and highest performance, respectively. Using Cemp, we attempt to find the combination of a file system and an I/O scheduler to make SSD-I deliver a smooth change in energy consumption. We employ Filebench as a workload generator to simulate a wide range of workloads (i.e., varmail, fileserver, and webserver), and explore optimal combinations of file systems and I/O schedulers (i.e., optimal values of AI) for tested SSDs under different workloads. Experimental results reveal that the proposed aggregative indicator is comprehensive for exploring the optimal combination of a file system and an I/O scheduler for SSDs, compared with an individual metric.

Key words: Solid state disk (SSD), Performance, Energy consumption, File system, I/O scheduler
doi:10.1631/jzus.C1300314 **Document code:** A **CLC number:** TP316

1 Introduction

Solid state disks (SSDs) become popular storage devices and have advantages over hard disk drivers (HDDs) in terms of high performance, low energy consumption, shock resistance, and data reliability. Without moving components (e.g., magnetic head and spinning platters), an SSD has a lower latency of read or write than an HDD. SSD performance and energy consumption under random read-intensive workloads identify with those under sequential read-intensive workloads. However, a gap exists in write-intensive workloads. An SSD has lower performance and higher energy consumption under random write-intensive workloads than under sequential write-intensive ones.

Currently, few file systems (Rosenblum and Ousterhout, 1992) and I/O schedulers (Riska *et al.*, 2007) in an operating system concentrate on characteristics of non-moving components in an SSD. Studies focusing on SSD performance are mainly classified into two camps: (1) qualitatively study SSD

‡ Corresponding author

* Project supported by the National Basic Research Program (973) of China (No. 2011CB302303), the National Natural Science Foundation of China (No. 60933002), the National High-Tech R&D Program (863) of China (No. 2013AA013203), and the U.S. National Science Foundation under Grants CCF-0845257 (CAREER), CNS-0917137 (CSR), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), OCI-0753305 (CI-TEAM), DUE-0837341 (CCLI), and DUE-0830831 (SFS)

©Zhejiang University and Springer-Verlag Berlin Heidelberg 2014

performance and design new technologies to improve the SSD performance (Wang YK *et al.*, 2011; Kim *et al.*, 2012; Wang H *et al.*, 2013); (2) build SSD performance models (Dirik and Jacob, 2009; Hu and Haas, 2010; Maghraoui *et al.*, 2010). Few methods were applied to study the energy consumption of an SSD due to its low energy consumption compared with HDDs. Researchers studied energy consumption of SSDs on the basis of different levels of workloads (O'Brien *et al.*, 2008; Kim J *et al.*, 2009; Park *et al.*, 2009), e.g., micro-benchmarks at the block level and macro-benchmarks at the file system level. The energy consumption of SSDs has not been fully addressed in the prior studies. Park *et al.* (2011) studied the characteristics of performance and energy consumption in two file systems (i.e., ext2 and LFS). Sehgal *et al.* (2010) evaluated impacts of file systems on energy consumption and performance. SSD performance and energy consumption were closely related to the combination of file systems and I/O schedulers. Different combinations made diverse impacts on SSDs under various workloads.

We are motivated to apply a metric called the aggregative indicator (AI) to study aggregative impacts on performance and energy consumption of an SSD from different combinations of file systems and I/O schedulers. With AI in place, we not only extend work investigated by Sehgal *et al.* (2010) but also attempt to explore the optimal combination for an SSD under a workload condition. Five file systems (i.e., ext2, ext3, NILFS2, ReiserFS, and XFS) and three I/O schedulers (i.e., CFQ, Deadline, and NOOP) are configured in the operating system. Under a variety of workloads (i.e., varmail, fileserver, and webserver) simulated by Filebench, we explore performance and energy consumption of three real-world SSDs coupled with these file systems and I/O schedulers. We observe that the best combination of a file system and an I/O scheduler for SSDs could be configured in the operating system to deliver high performance with low energy consumption.

Note that there are two types of states for SSDs, i.e., initial state and steady state. The initial state, which is also called factory state, means that the SSD has just been initialized. The steady state indicates that the SSD has essentially invariant performance. In this paper, we focus on the right combination of a file system and an I/O scheduler to make initial state-based SSDs achieve high performance with low energy consumption. Steady state-based SSDs, which present invariant performance, are not satisfactory for selection of the right combination.

2 Background and related work

2.1 NAND flash and solid state disk

NAND flash (Luo and Zhao, 2007), used for data storage, was invented by Masuoka *et al.* (1987). NAND flash is composed of rows (connected to word-lines) and columns (connected to bit-lines). The intersection of a row and a column is named a floating gate transistor (FGT), which stores charges (Lee *et al.*, 2002). The structure of FGT is renamed a cell storing data bits. The number of data bits in a cell determines the type of NAND flash, e.g., single-level cell (SLC) NAND flash with one bit in a cell, multi-level cell (MLC) NAND flash including two data bits per cell, and triple-level cell (TLC) NAND flash having three data bits in a cell. Both MLC and TLC NAND flash memory have larger capacity, lower performance, and shorter endurance than SLC NAND flash.

NAND flash can execute three operations, namely, read, program (write), and erase. Each row in NAND flash corresponds to a page, which is the unit for a read or program operation. The block, which is the atomic unit for an erase operation, is composed of a fixed group of pages. Within a block, a page is read sequentially or randomly. Program operations can only access a page sequentially. For an updated page, only out-of-place and erase-before-write updates are allowed. Excessive updates can degrade the number of available program/erase cycles in NAND flash, thereby shortening SSD lifetime.

NAND flash has addresses multiplexed into eight I/Os, by which command, address, and data are communicated. NAND flash is arranged into a number of planes. One plane consists of a fixed number of blocks and a page register. The register buffers data transmission from (or to) cell memory during page read and program operations. NAND flash memory array space can be divided into many planes, in which two planes-based operations (e.g., read, program, and erase) can be supported to enhance SSD performance.

SSDs are designed as a block device and have been widely applied in embedded systems, data

center, and search engine companies. An SSD has higher performance and lower energy consumption over an HDD under random I/O-based workload conditions. An SSD is composed of one or more SSD controllers, NAND flash memory array, flash translation layer (FTL) (Gupta *et al.*, 2009; Wei *et al.*, 2011; Lu *et al.*, 2013), interface controller (i.e., SATA, SAS, fiber channel, and PCI-E), a board cache, and a printed circuit board. The SSD controller runs FTL and transforms commands between a host and NAND flash. FTL is an intermediate software layer and includes address mapping, garbage collection (Iliadis, 2010; Jung *et al.*, 2012), and wear-leveling (Jung *et al.*, 2007; Murugan and Du, 2011). The board cache can improve the performance of small writes and temporarily store a mapping table. FTL aims to map the logical address of a page into a physical one in flash memory. Three proposed mapping schemes include block-mapping (Ban, 1995), page-mapping (Gupta *et al.*, 2009), and hybrid mapping (Kim *et al.*, 2002; Kang *et al.*, 2006; Lee *et al.*, 2007; 2008).

A block-level mapping uses the minimal cache space for a mapping table but can incur more data updates and rewrite operations, which shorten the lifetime of NAND flash memory. A page-level mapping reduces the frequency of rewrite operations, but consumes much cache space for a mapping table. A hybrid mapping scheme combines advantages in block- and page-level mappings. It decreases occupied cache space and reduces the number of data updates. These methods extend the lifetime of NAND flash memory. Garbage collection reclaims invalid pages in blocks to provide free space for the page program. In the process of garbage collection, there can be an excessive number of page-rewrite and block-erase operations, which deteriorates SSD performance and raise energy consumption of SSDs. Wear-leveling guarantees an even erase count distribution during the process of writing data across NAND flash memory.

2.2 SSD performance and energy consumption

There are two approaches for investigating SSD performance. One is the qualitative study, in which researchers gained performance parameters and applied these parameters to understand behaviors of an SSD (Bux, 2009). Afterwards, new hardware and software were designed to improve SSD performance. The results of mean latency on a variety of SSDs were presented in Desnoyers (2010). Read, program, and erase latency were an approximation of specified values from manufacturer datasheets in most cases. Park and Shen (2009) designed a trace-driven method for evaluating SSD performance under scientific workload conditions. Experiments (Kim JH *et al.*, 2009; Kim J *et al.*, 2012) showed that SSDs slightly outperform HDDs in performance under a sequential write-dominant workload condition. When multiple processes simultaneously accessed a single SSD, the performance significantly reduced. Lee *et al.* (2008) evaluated the applicability and potential impact on SSDs under workload conditions including sequential write and random read operations. They concluded that significant improvement was achieved in transaction processing by replacing magnetic disks with SSDs. Chen *et al.* (2009) conducted intensive experiments and measurements on different types of state-of-the-art SSDs and achieved an insight into the unique performance characteristics of SSDs. They observed a handful of performance issues and SSD behaviors to improve SSD performance.

The second approach is to build a performance model (Huang *et al.*, 2011) according to application workloads. An SSD simulator (Agrawal *et al.*, 2008) based on DiskSim was used to study impacts of hardware and software components on SSD performance under a targeted workload condition. Another SSD model based on the Disksim simulator (Kim Y. *et al.*, 2009) was developed to evaluate SSD performance and analyze the energy consumption of SSDs by employing different flash translation layer schemes. Maghraoui *et al.* (2010) provided a linear model for SSDs. Parameters of the model were presented along with micro-benchmarks and used to build a simulator. Dirik and Jacob (2009) proposed a simulator based on DiskSim v2.0 to study SSD architectures and management techniques, which quantified an SSD performance under user-driven/PC applications in a multi-task environment. An analytic approach to modeling operations of an SSD (Bux, 2009) was described by a Markov chain to evaluate performance of write operations. An empirical model (Hu and Haas, 2010) was developed to study SSD performance in the case of greedy garbage collection policy under random write workload

conditions. This model aims to quantitatively analyze the fundamental limit of sustained random write performance of SSDs.

For SSD energy consumption, O'Brien *et al.* (2008) provided energy consumption measurements based on USB devices, which were composed of NAND flash memory. Experimental results discussed energy consumption and performance in USB devices. Seo *et al.* (2008) evaluated the efficiency of SSD energy consumption at the block I/O level and file system level. Park *et al.* (2009) designed an energy consumption model for an SSD. They collected traces from a real-world workload condition and replayed them on this model to study the SSD energy consumption. FlashPower, an NAND flash energy consumption model (Mohan *et al.*, 2010) integrated with CATI5.3, was demonstrated based on SLC-NAND flash. Parameters of device and micro-architecture were taken as input in this model to estimate the energy consumption of SSDs during various operation modes. Yoo *et al.* (2011) studied energy consumption of write operations to explore key technical characteristics inside the SSD.

2.3 File system and I/O scheduler

Fig. 1 depicts the structure of a storage system. Our study concentrates on the impacts of file systems and I/O schedulers on SSDs. We investigate the combination of file systems and I/O schedulers in SSD-based storage systems to achieve high performance and low energy consumption. Five mainstream file systems (i.e., ext2, ext3, ReiserFS, XFS, and NILFS2) and three I/O schedulers (i.e., CFQ, Deadline, and NOOP) are configured in an operating system to measure performance and energy consumption of SSDs. I/O schedulers sort and merge written requests in SSDs to optimize performance at the block level. The Linux kernel version used in our experiment is 2.6.35. We employ default parameters of a file system in the process of formatting and mounting SSDs.

2.3.1 File system

Ext3, Reiserfs, and XFS are typical journaling file systems, which keep track of changes in a journal and store circular logs in a dedicated area in storage devices. The journaling file systems have three modes, namely journal, ordered, and write-back.

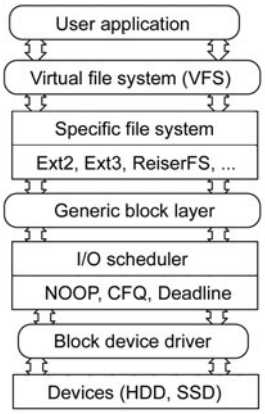

Fig. 1 The storage level based on SSDs and HDDs. We concentrate on the specific file system layer and the I/O scheduler layer in this study

Ext2 (i.e., the second extended file system) (Appleton, 1997) and ext3 (i.e., the third extended file system) (Tweedie, 2000): ext2 and ext3 belong to extended file systems. Ext2 inherits ideas from the Berkeley Fast File System (McKusick *et al.*, 1984). They have the same table-based structure of inodes. From the table, we can obtain the address of an inode. Ext3, extended from ext2, has all advantages of the journal-structured file system. Ext3 guarantees the system recovery from a crash, for example. In ext3, the default mode is ordered, which is a kind of physical journal. In addition, it costs much time to recover the crashed file system. Under the same workload condition, ext3 performs more extra writes than ext2.

ReiserFS (Agrawal *et al.*, 2007): ReiserFS, the logical journaling file system designed based on object-oriented paradigm, is composed of the semantic layer and storage layer. The semantic layer manages object namespace and defines object interfaces. Disk space is managed by the storage layer. The mapping item (i.e., from file name to file) is implemented by the B+ tree structured inode. The data in a file system is stored by the pattern of B+ tree, which is different from the mapping table in ext2 or ext3. The data is stored in inodes of leaves in the B+ tree, by which the performance of searching data is better than that in extended file systems under a workload with small file size (e.g., smaller than 4 KB).

XFS (Wang and Thomas, 1993): The 64-bit high-performance journal file system partitions disk space into allocation groups (also called chucks or linear regions). Each allocation group manages its

own inodes and free space, respectively. In the interior of an allocation group, the data is managed by the pattern of B+ tree. The structure improves parallel I/O performance in multi-processor systems, because metadata updates are processed in parallel.

NILFS2 (Konishi *et al.*, 2006) (i.e., new implementation of a log-structured file system): Data in NILFS2 is recorded by the pattern of the log structure (i.e., append-write mode). Data and metadata updates are sequentially written in log space. This can reduce the seek time and minimize the data loss, which occurs after a crash. NILFS2 supports continuous snapshotting. The management for data and inodes is implemented by the pattern of B-tree. Append-write mode makes SSDs have better performance compared with random write patterns. NILFS2 reduces the random writes and improves SSD lifetime.

2.3.2 I/O scheduler

NOOP (Pratt and Heger, 2004; Moallem, 2008; Heger and Quinn, 2010): the NOOP scheduler only inserts incoming I/Os into a simple FIFO queue and implements I/Os merging, meaning that it cannot sort but only merge requests. This scheduler is useful when it has been determined that the host should not attempt to re-order requests based on the sector numbers contained therein.

Deadline (Pratt and Heger, 2004; Moallem, 2008; Heger and Quinn, 2010): Deadline has two advantages over NOOP. The sorted queue sorts requests from the upper layer to minimize seek times and allocates a deadline to each request, thereby ensuring that each request can be eventually serviced. Therefore, two couples of sorted queues and one deadline queue are applied for read and write I/Os, respectively.

CFQ (Pratt and Heger, 2004; Moallem, 2008; Heger and Quinn, 2010): CFQ sets a queue for a process, allocates a time slice to requests in the queue, and handles requests in a round robin manner when the time slice on the request expires. CFQ sorts requests using their addresses to reduce time of disk pads rotation and improve HDD performance.

These schedulers initially aim to reduce seek time and improve HDD performance. SSDs have no moving component; reducing seek time is not applicable for SSDs but can improve random writes inside SSDs. To find the optimal scheduler for SSDs, we perform experiments on SSDs based on different file systems and I/O schedulers under various workload conditions (see Section 5).

3 Evaluation model

We describe an evaluation model to explore an optimal combination of a file system and an I/O scheduler to improve the performance and energy efficiency of SSDs. In this section, we present the evaluation model based on voltage, current, and performance in an SSD.

The energy consumption of an SSD can be expressed as

$$P = UI, \tag{1}$$

where U and I are the voltage and current on an SSD, respectively. U is a constant, whereas I varies based on workload conditions. We divide the period of measurement (i.e., T) into a handful of intervals. The energy consumption P_i measured during the ith interval T_i can be computed as:

$$P_i = U_i I_i,$$

where P_i, U_i, and I_i denote the energy consumption, voltage, and current measured during interval T_i, respectively. Therefore, the average value of energy consumption (i.e., $\overline{P}$) of the SSD in the process of measurement under a workload condition is expressed as

$$\overline{P} = \frac{\sum_{i=1}^{N}(P_i T_i)}{T} = \frac{\sum_{i=1}^{N}(U_i I_i T_i)}{\sum_{i=1}^{N} T_i}, \tag{2}$$

where N is the number of interval T_i during the period of measurement, i.e., $T = \sum_{i=1}^{N} T_i$. As the voltage of SSD is considered to be a constant, Eq. (2) may be rewritten as

$$\overline{P} = U\frac{\sum_{i=1}^{N}(I_i T_i)}{\sum_{i=1}^{N} T_i}. \tag{3}$$

SSD performance can be evaluated in terms of data transfer rate (i.e., MB/s) and throughput (i.e., IOPS). Given a workload condition, performance metrics, i.e., IOPS and MBPS, for SSDs have the

same proportion of changes. Therefore, we choose IOPS as a performance metric of SSDs.

To assess SSD performance per unit of energy consumption, we employ an aggregative indicator AI to explore the optimal combination of a file system and an I/O scheduler for an SSD:

$$\mathrm{AI}=\frac{\mathrm{IOPS}}{\overline{P}}=\mathrm{IOPS}\frac{\sum_{i=1}^{N}T_i}{U\sum_{i=1}^{N}(I_iT_i)}. \quad (4)$$

We employ AI to evaluate performance and energy consumption for an SSD based on different combinations of file systems and I/O schedulers. The larger the value of AI, the more optimal the combination of a file system and an I/O scheduler is for the SSD. In the following section, we apply our evaluation model and method to real-world SSDs to optimize the performance and energy consumption of SSDs.

4 Measurement environment and methodology

4.1 Experiment setup

Our measurement system (Fig. 2) is mainly composed of a workload generator system, a power analyzer system, and tested SSDs. The workload generator system hosts on a desktop PC, which has an Intel Dual-Core 2.50 GHz CPU, 2 GB DRAM, and 1 TB HDD with 64 MB cache. Experimental results of performance and energy consumption for tested SSDs are stored in the HDD. In the workload generator system, Filebench running on Ubuntu 10.0 (Linux-2.6.35) generates three simulated workloads, namely varmail, fileserver, and webserver. The power analyzer system samples current at a frequency of 10 kHz from tested-SSDs by Hall-effect measurement on the 5 V line. These values of current are transmitted to the HDD of the host by a local area network at an interval of 1 s. Tested SSDs include Intel X25-M 80 GB G2 (SSD-I for short), SoliWare 64 GB (SSD-S for short), and Kingston 80 GB SSD (SSD-K for short). Characteristics of the tested SSDs are listed in Table 1. MLC NAND flash presents higher energy consumption but lower performance than SLC NAND flash. Therefore, we select MLC NAND flash-based SSDs to explore the right combination of a file system and an I/O scheduler to guarantee SSDs high performance at low energy consumption. Two types of SSD controller are applied in our test. SSD controllers for SSD-I and SSD-K are implemented in an ASIC design while SSD-S has an FPGA-based controller. The former consumes lower energy than the latter. To study the impact of DRAM capacity on performance and energy consumption of SSDs, SSD-I with a smaller DRAM capacity (e.g., 16 MB) and SSD-S or SSD-K with a larger DRAM capacity (e.g., 128 MB) are employed in our test. SSD-S outperforms SSD-I or SSD-K occasionally when temperatures fall in a wider range.

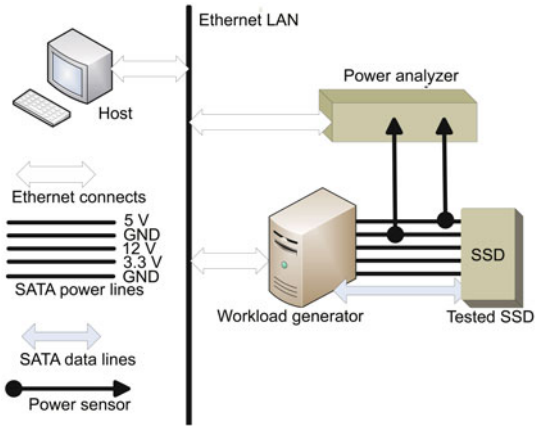

Fig. 2 The system for measuring the performance and energy consumption of tested SSDs

4.2 Measurement workload

Filebench, a file system and storage benchmark, can resemble real-world workload scenarios. We

Table 1 Characteristics of tested SSDs

Tested SSDs	Capaciity (GB)	Flash type	SSD controller	DRAM size (MB)	Performance (MB/s)		Power (W)		Secure erase
					Read	Write	Work	Idle	
SSD-I	80	MLC	Intel PC29AS21AA0	16	201.3	73.8	2.6	0.15	Support
SSD-S	64	MLC	Xilinx XC3S1600E	128	98.8	103.6	4.1	1.50	Support
SSD-K	64	MLC	Toshiba T6UG1XBG	128	220.0	140.0	5.9	3.30	Support

employ three simulated workloads (i.e., varmail, fileserver, and webserver) in our experiment. The workload generator issues macro-benchmarks and records performance of tested SSDs. Varmail emulates file read and random append-write operations in a mail server; fileserver emulates reads, writes, and deletes on a fileset of random sizes in a file server; webserver emulates reads on a fileset of random sizes and a random append log-file write in a web server. Characteristics of workloads are listed in Table 2.

4.3 Measurement methodology

We evaluate SSD performance and energy consumption using the measurement system depicted in Fig. 2. Three workloads, five file systems, and three I/O schedulers are tested in our experiment.

In the experiment, we perform a secure erase command on a tested SSD under Ubuntu. By doing this, the data in tested SSDs is erased, and then SSDs can be put into the initial state before every test.

In each experiment, we initially configure a pair of file system and I/O scheduler in the Linux system. We run the workloads on an SSD for 600 s and measure performance and energy consumption for the tested SSD. In the process of measurement, we obtain results of energy consumption of the SSD and store them in the HDD of the workload generator system. When the workload is completed, we record the mean value of SSD performance. Afterwards, we change the combination types (e.g., file system types and I/O scheduler types), and then measure the tested SSD again.

We mainly explore initial state-based SSDs to find the right combination of a file system and an I/O scheduler, which makes it optimal for SSDs, meaning that SSDs provide high performance with low energy consumption. Long run-time can make tested SSDs deviate from the initial state; therefore we configure 600 s as the run-time in our experimentation.

We apply IOPS (i.e., performance metric) and power (i.e., energy consumption metric) to evaluate tested SSDs based on various combinations of file systems and I/O schedulers. The aggregative indicator (see AI in Eq. (4)) represents the value of IOPS/P for tested SSDs.

When all tests under a workload condition are completed, we obtain and analyze AI based on performance and energy consumption for the tested SSD. From experimental results, we choose the optimal combination of a file system and an I/O scheduler, which can optimize the performance and energy efficiency of the tested SSD. Using the Intel SSD based on CFQ and five file systems, we summarize an example of experimental results (including power, IOPS, speed, and AI) for SSD-I under fileserver in Table 3. We will present all results in the following section. In addition, we present an analysis of the optimal combination of a file system and an I/O scheduler for SSDs under three workload conditions.

5 Exploring optimal combination for SSD based on AI

We analyze three real-world SSDs under varmail, fileserver, and webserver workload conditions. For SSDs, we reveal performance (i.e., IOPS) and energy consumption, and then study the AI of different combinations of file systems and I/O schedulers. For

Table 2 Characteristics of macro-benchmarks in Filebench

Workload	File size (KB)	Number of files	Number of threads	I/O size	Time (s)	$N_{\text{read}} : N_{\text{write}}$
Varmail	16	1000	16	16 KB (M), 1 MB (R)	600	1:1
Fileserver	128	10 000	50	16 KB (M), 1 MB (F)	600	1:2
Webserver	16	1000	100	1 MB	600	10:1

F: fixed size of I/O; R: read size of I/O; M: mean size of I/O

Table 3 An example of experimental results based on the CFQ I/O scheduler for SSD-I under fileserver

File system	Power (W)	IOPS	Speed (MB/s)	AI
Ext2	1.8042	4598.63	110.2	2548.85
Ext3	1.8397	4220.50	101.1	2294.13
NILFS2	1.5689	3055.57	73.1	1947.60
ReiserFS	1.7290	3147.33	75.3	1820.36
XFS	1.9963	4351.04	104.2	2179.50

SSD-I having the largest AI value, we plot changes of energy consumption and mean performance (Cemp for short) under three conditions (i.e., the highest performance, the lowest energy consumption, and the largest AI), respectively.

5.1 Varmail workload

5.1.1 AI results under varmail

Varmail workload resembles I/O behaviors (i.e., random read and append-write or sequential write I/Os) of real-world email servers. File size in this workload is relatively small (Table 2). Figs. 3–5 show experimental results of tested SSD-I, SSD-S, and SSD-K under varmail workload.

In most cases, SSDs coupled with extended file systems (e.g., ext2 or ext3) have higher performance (i.e., IOPS) at lower energy consumption (Fig. 3) than that using other file systems under varmail, meaning that the ext2- and ext3-based SSDs have better AI performance in the case of three I/O schedulers (Fig. 3c). The results plotted in Figs. 4a and 5a reveal that when it comes to SSD-K and SSD-S, the extended file systems with I/O schedulers achieve better performance than other file systems.

Ext2- and ext3-based SSDs have high energy efficiency. For example, the energy consumptions of SSD-K and SSD-S under CFQ are 1.1374 W and 0.4094 W, respectively; the energy consumptions of SSD-K and SSD-S under Deadline are 0.1955 W and 1.0584 W, respectively; the energy consumptions of SSD-K and SSD-S under NOOP are 0.1921 W and 1.2577 W, respectively. Integrated with NOOP, ext2- or ext3-based SSDs have higher AI values than other file systems. The SSDs coupled with ext2 or ext3 exhibit high AI values. The outstanding performance and energy efficiency are attributed to workload types, the combination of a file system and an I/O scheduler, and SSDs. SSDs achieve high performance for varmail workload containing random reads and append writes. Therefore, SSDs coupled with the extended file system ext2 or ext3 can provide high performance and low energy consumption under varmail workload. File size in varmail is relatively small and the extended file systems deal well with small size files by virtue of inodes. Ext2-based SSDs do not store journey of data in disks and cost less energy than ext3-based SSDs, where SSDs should store journey of data. SSDs based on ReiserFS express low energy consumption, which results from the B+ tree structure used in data searching and processing. These advantages in extended file systems make SSDs have large values of AI. In our experiment, SSD-K has the largest AI value when the ext2 file system is integrated with NOOP under varmail workload.

5.1.2 Cemp for SSD-I under varmail

The changes of energy consumption and mean performance (Cemp) for SSD-I under three conditions (the largest AI, the lowest power, and the highest performance) are shown in Fig. 6.

Ext2-CFQ-based SSD-I has the largest AI and lowest power (Figs. 6b and 6c), meaning that the SSD-I based on the combination of ext2 and CFQ can achieve the right high performance with low energy consumption. Under these two conditions, the energy consumption of SSD-I ranges from 0.525 W to 0.535 W for read I/Os in the workload; the energy consumption is anywhere between 0.355 W and 0.555 W for append-write operations. Because read- and append-write-based I/Os result in low energy consumption, the change of energy consumption is relatively smooth. In addition, write I/Os in ext2 less than those in ext3 are loaded on SSD-I; therefore, the SSD-I based on ext2 and CFQ performs with invariant energy consumption compared to that based on ext3 and NOOP. Ext3-NOOP-based SSD-I offers higher performance than the ext2-CFQ-based counterpart; however, ext3-NOOP-based SSD-I has high energy efficiency because ext3 can load more write I/Os on SSD-I than ext2. Therefore, ext3-NOOP-based SSD-I has sharper energy consumption changes (from 1.5 W to 1.8 W in Fig. 6a) than ext2-CFQ-based SSD-I (from 0.525 W to 0.553 W in Fig. 6a or Fig. 6c). We conclude from these results that ext2-CFQ-based SSDs offer more optimal AI (i.e., better performance at lower energy consumption) under varmail workload.

5.2 Fileserver workload

5.2.1 AI results under fileserver

Fileserver workload resembles a real-world multi-threading workload composed of random reads and writes. File size in fileserver workload is large (Table 2). The percentages of random writes and reads are 67% and 33%, respectively. Figs. 7–9

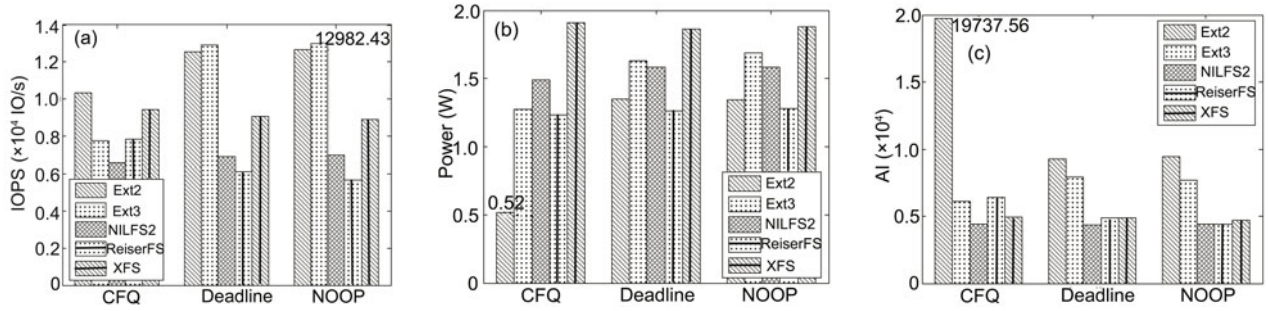

Fig. 3 Experimental results of SSD-I under varmail: (a) IOPS; (b) Power; (c) AI

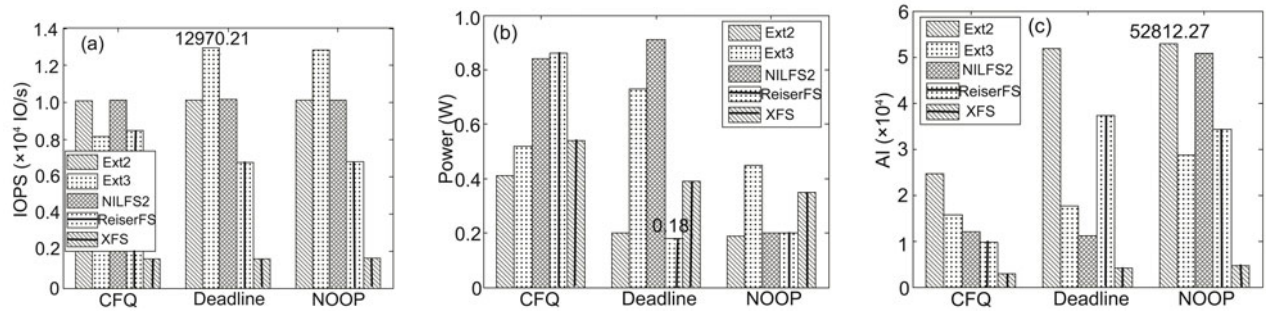

Fig. 4 Experimental results of SSD-K under varmail: (a) IOPS; (b) Power; (c) AI

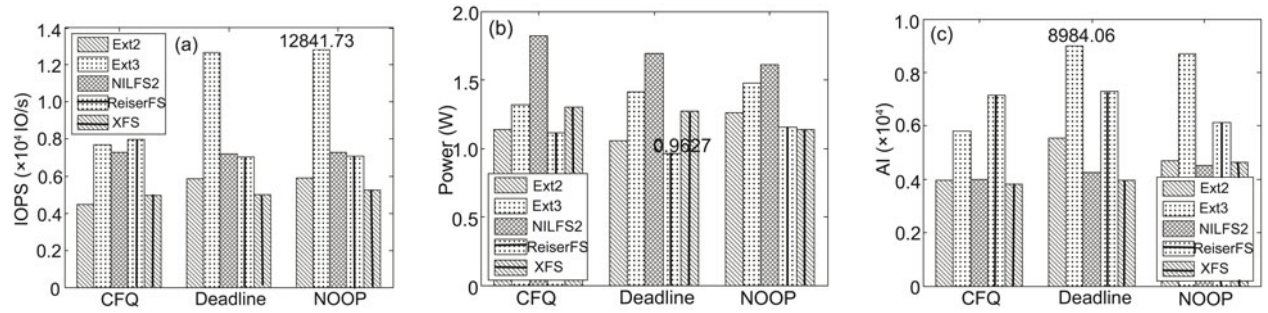

Fig. 5 Experimental results of SSD-S under varmail: (a) IOPS; (b) Power; (c) AI

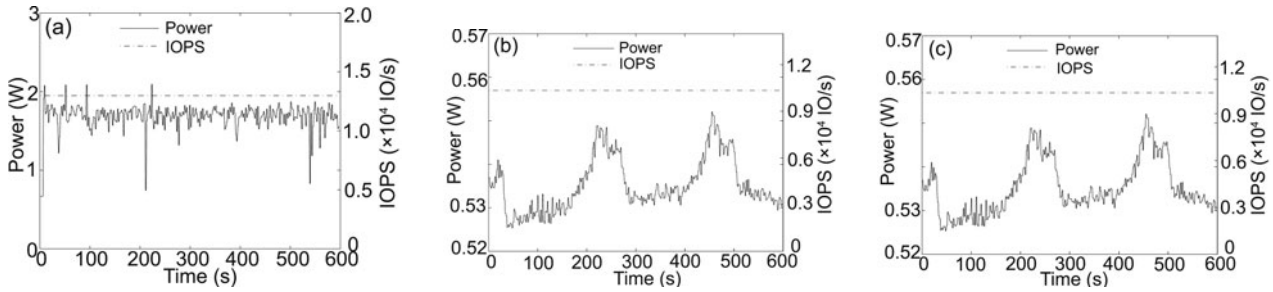

Fig. 6 Cemp of SSD-I under varmail: (a) ext3 and NOOP based on the largest IOPS; (b) ext2 and CFQ based on the lowest power; (c) ext2 and CFQ based on the largest AI

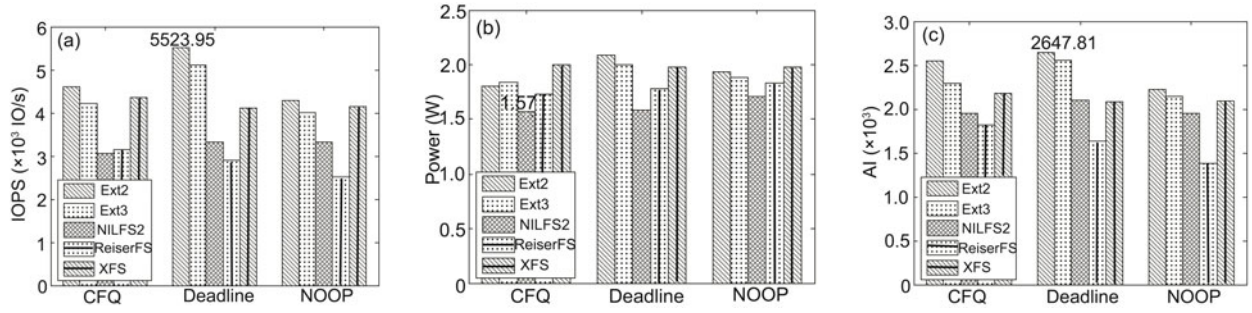

Fig. 7 Experimental results of SSD-I under fileserver: (a) IOPS; (b) Power; (c) AI

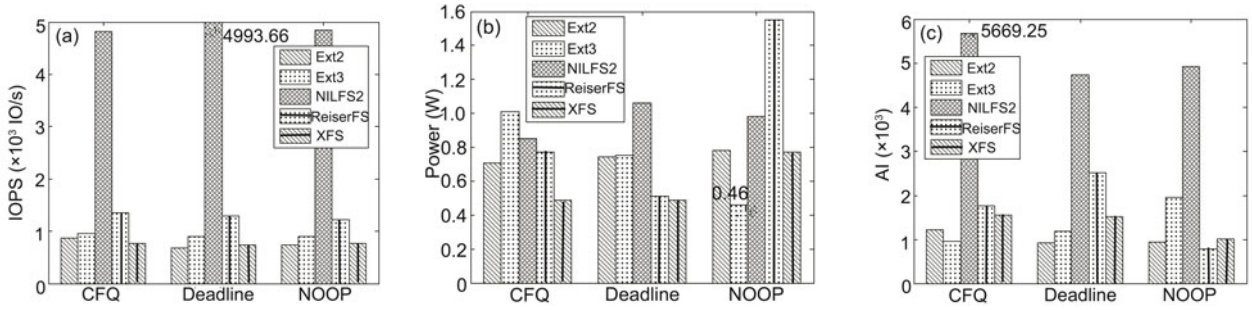

Fig. 8 Experimental results of SSD-K under fileserver: (a) IOPS; (b) Power; (c) AI

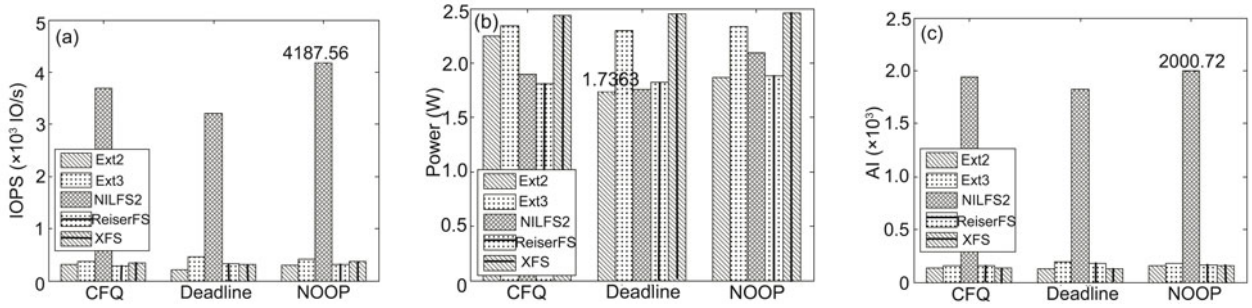

Fig. 9 Experimental results of SSD-S under fileserver: (a) IOPS; (b) Power; (c) AI

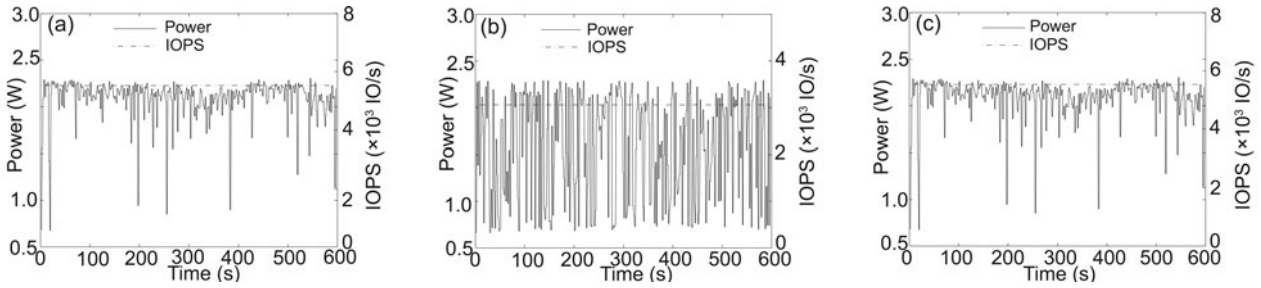

Fig. 10 Cemp of SSD-I under fileserver: (a) ext2 and Deadline based on the largest IOPS; (b) NILFS2 and CFQ based on the lowest power; (c) ext2 and Deadline based on the largest AI

reveal experimental results for three tested SSDs.

Experimental results reveal that performance of tested SSD under fileserver workload is lower than that under varmail or webserver workload. Fileserver contains more random write I/Os than read ones; in contrast, there are fewer random write I/Os in varmail (i.e., read and append-write I/Os) and webserver (i.e., 90% read I/Os). Garbage collection is easily triggered by random write I/Os loaded on SSDs. Therefore, the values of SSD performance under fileserver workload are smaller than those under varmail and webserver, while the values of energy consumption of SSDs are larger than those under other workload conditions.

SSD-I's garbage collection triggered by random I/O causes SSD-I to have much higher and sharper changes in energy consumption under fileserver than under the other workload (Fig. 10).

NILFS2 makes use of a log-structure file system to handle random write operations. Such a sequential write pattern allows performance improvement of SSDs. Therefore, NILFS2 improves SSDs in terms of AI, IOPS, and energy consumption.

Fig. 7 shows that the values of performance of extended-file-system-based SSDs are higher than those of the same SSDs using other file systems. NILFS2-based SSDs offer high energy efficiency. The AI values of the extended-file-system-based SSD-I are very large. Figs. 8 and 9 show that NILFS2-based SSDs provide higher performance and lower energy consumption than the same SSDs that employ other file systems. For example, the IOPS values of SSD-K and SSD-S are 4993.66 and 4187.56, respectively. Figs. 7a, 7b, and 7c reveal that ext2- and ext3-based SSDs have high energy efficiency, because of there being no journal operations in the table structure of inodes. NILFS2-based SSDs have higher energy consumption than the SSDs using the extended file system. NILFS2-based SSDs have higher AI values than those based on others with an I/O scheduler. NILFS2-CFQ-based SSD-K offers high AI values. The detailed energy consumption information for the SSDs can be found in Fig. 10.

5.2.2 Cemp for SSD-I under fileserver

In fileserver workload, the write ratio is higher than those in other workloads. We depict the Cemp for SSD-I under three conditions, i.e., the largest AI, lowest power, and highest performance, in Fig. 10. Experimental results plotted in Figs. 10a and 10c confirm that the ext2-Deadline-based SSD-I offers the largest AI and highest performance, respectively. However, the energy consumption of the ext2-Deadline-based SSD-I is much higher than that of the NILFS2-CFQ-based SSD-I. The ext2-Deadline-based SSD-I provides high overall performance. The energy consumption of the ext2-Deadline-based SSD-I is anywhere between 1.8 W and 2.4 W; the energy consumption of the NILFS2-CFQ-based SSD-I ranges from 0.7 W to 2.2 W. We find that the change in energy consumption in Fig. 10b is much sharper than that in Fig. 10a or Fig. 10c. The NILFS2-CFQ-based SSD-I has low performance; SSD-I has lower AI than that plotted in Fig. 10c. Fig. 10 shows that the ext2-Deadline-based SSD-I has higher AI (i.e., better performance at lower energy consumption) than other types of SSD-I under fileserver workload.

5.3 Webserver workload

5.3.1 AI results under webserver

Webserver workload simulates workload in a web server. The percentage of read operations in this workload is around 90%. Operations include random read and append log data writes. SSDs provide high performance under a workload with read and append-write operations. File size in this workload is much smaller than those in other workloads (Table 2). Therefore, SSDs exhibit high performance and low energy consumption under this workload. Experimental results are shown in Figs. 11–13.

SSD performance is higher than those under other workloads (Fig. 11a). For example, IOPS of SSD-K under webserver, SSD-I under fileserver, and SSD-I under varmail are 18 373.87, 5523.95, and 12 982.43, respectively. Ext2 is very beneficial to workload conditions with small file sizes and does not store journal data in disk space. Hence, performance of ext2-based SSD is better than that based on other file systems (Figs. 11a, 12a, and 13a, in which the highest SSD performance is achieved based on different I/O schedulers). Ext2 is superior to journal file systems in the use case of SSDs (refer to Section 5.1.1 for the reasons why ext2 has these advantages over journal file systems).

With a large percentage of reads and append-write log data in this workload, SSD energy

consumption is low (Figs. 11b, 12b, and 13b). Energy consumption of the XFS-based SSDs becomes small, because XFS can efficiently process large size files. The AI values are higher than those under other workloads. The results (Figs. 11c, 12c, and 13c) show high AI values for extended-file-system-CFQ-based SSDs. The ext3-CFQ-based SSDs can achieve the optimal performance. According to Figs. 11c, 12c, and 13c, the largest value of AI for any tested SSD exists in the CFQ-based combination. In most cases, CFQ can help improve the overall SSD performance and energy consumption under this workload.

5.3.2 Cemp for SSD-I under webserver

Under different conditions (i.e., the largest AI, lowest power, and highest performance), three kinds of Cemps exist in SSD-I based on three types of combinations (Fig. 14). The ext3-CFQ-based SSD-I has the largest AI (Fig. 14c), meaning that SSD-I offers high performance and consumes low energy. Fig. 14a shows that the ext2-NOOP-based SSD-I has high performance with low energy consumption ranging from 0.655 W to 2.260 W. The energy consumption of the ext2-NOOP-based SSD-I changes much dramatically than that of the ext3-CFQ-based SSD-I, in the interval (0.6563, 2.2897) W. The mean value of AI for the ext2-NOOP-based SSD-I is smaller than that of SSD-I based on the optimal combination (Fig. 14c). The energy consumption of the NILFS2-CFQ-based SSD-I (Fig. 14b) is anywhere between 0.6574 W and 2.1737 W, which is smaller than that for SSD-I based on the optimal combination. However, SSD-I has two deficiencies, lower performance and dynamically changing energy consumption, which lower the AI value of NILFS2-CFQ-based SSD-I. Fig. 14 shows that SSD-I based on the optimal combination has lower energy efficiency than NILFS2-CFQ-based SSD-I, and lower performance than ext2-NOOP-based SSD-I. The ext3-CFQ-based SSD provides the best overall performance and energy consumption among the SSDs under webserver workload.

6 Conclusions

In this paper, we aim at optimizing performance and energy efficiency of SSDs by choosing the most appropriate combination of a file system and an I/O scheduler. We demonstrate a holistic way of evaluating performance and energy efficiency of SSDs using the AI rather than a single metric under various workload conditions, which are generated by filebench. We draw the following conclusions:

1. Experiment results confirm that SSD performance and energy consumption largely depend on the combination of a file system and an I/O scheduler under a workload condition.

2. In most cases, tested SSDs coupled with the extended file systems (e.g., ext2 and ext3) and any I/O scheduler provide better AI than other SSD counterparts under varmail workload with read and append-write I/Os.

3. Tested SSDs based on NILFS2 and any I/O scheduler offer high AI under fileserver workload containing a large number of writes.

4. Tested SSDs using the combination of the extent file systems (i.e., ext2 and ext3) and CFQ exhibit high AI values under the read-intensive webserver workload.

We also choose a typical SSD (i.e., SSD-I) to study changes of energy consumption and mean performance (i.e., Cemp) in SSDs under three conditions (i.e., the largest AI, lowest power, and highest performance), and draw the following conclusions:

1. SSD-I based on the extended file systems (e.g., ext2 and ext3) delivers high performance at low energy efficiency in most cases. The ext2-based SSD-I has no journaling-write operations; therefore, ext2-based SSD-I reveals lower energy consumption and less sharper changes in energy consumption values than ext3-based SSD-I. However, data stability and security in ext2 are not guaranteed. Ext3 including characteristics of ext2 maintains the journal-structure, which guarantees the security of this file system.

2. NILFS2 is beneficial for write operations, especially random writes. Compared with other file systems, NILFS2-based SSD-I performs with lower energy consumption and smoother changes in energy consumption.

3. Currently, I/O schedulers are designed for HDDs to optimize HDD performance. Our experiment confirms that the CFQ and NOOP schedulers can improve SSD-I performance and energy consumption under workload conditions with most read I/Os while Deadline-based SSD-I displays high performance at low energy consumption.

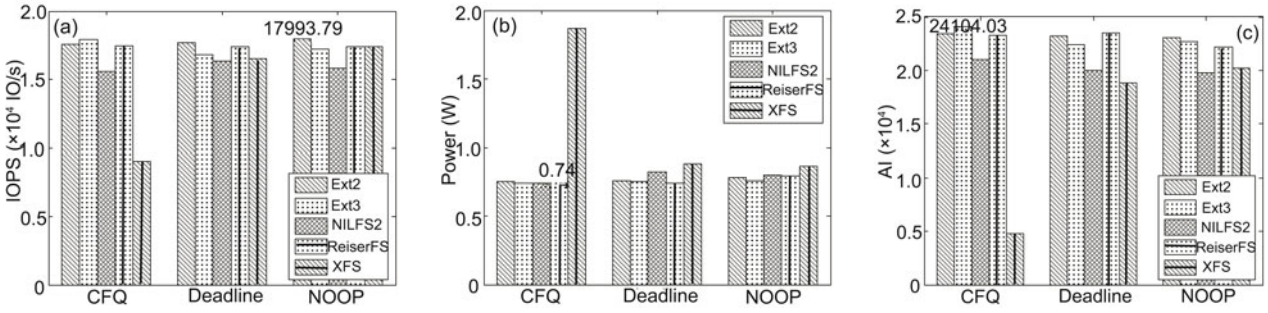

Fig. 11 Experimental results of SSD-I under webserver: (a) IOPS; (b) Power; (c) AI

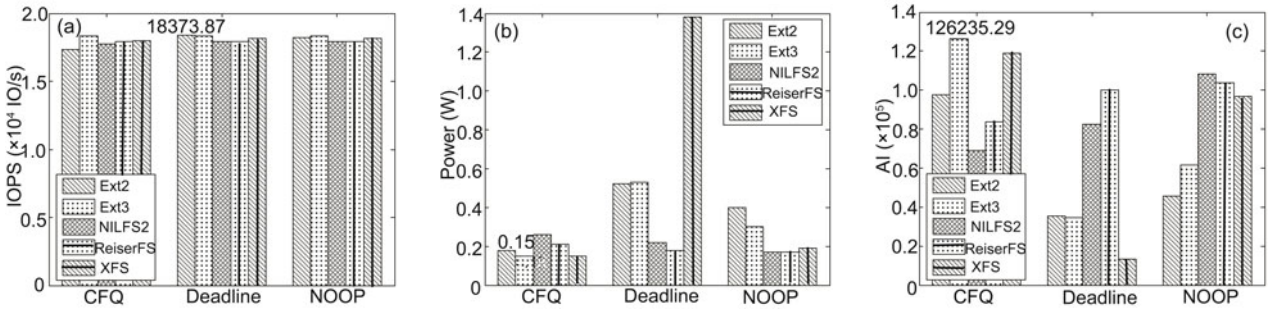

Fig. 12 Experimental results of SSD-K under webserver: (a) IOPS; (b) Power; (c) AI

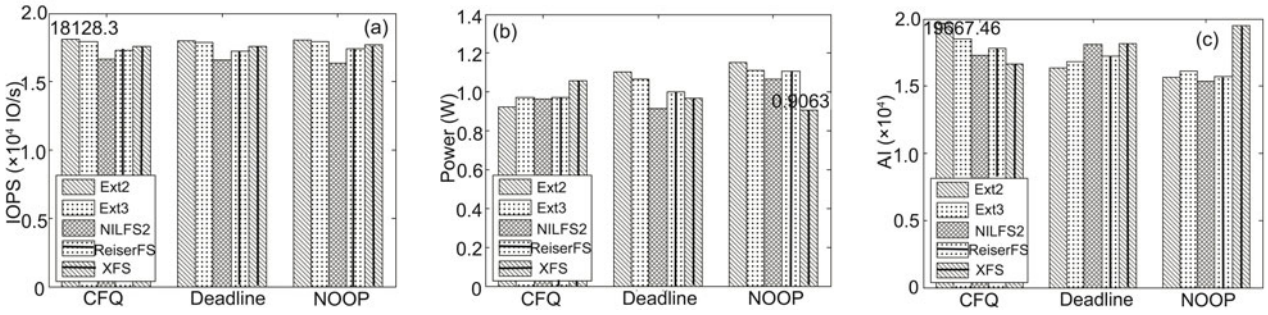

Fig. 13 Experimental results of SSD-S under webserver: (a) IOPS; (b) Power; (c) AI

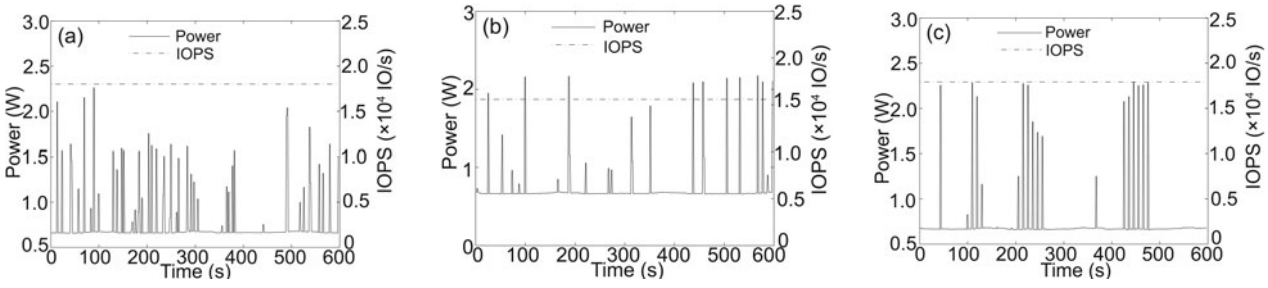

Fig. 14 Cemp of SSD-I under webserver: (a) Ext2 and NOOP based on the largest IOPS; (b) NILFS2 and CFQ based on the lowest power; (c) Ext3 and CFQ based on the largest AI

Acknowledgement

We thank Jian ZHOU, Rui ZENG, Jin CHENG, and Lei LIU for their help in previous experimentation.

References

Agrawal, N., Bolosky, W.J., Douceur, J.R., *et al.*, 2007. A five-year study of file-system metadata. *ACM Trans. Storage*, **3**(3), Article 9. [doi:10.1145/1288783.1288788]

Agrawal, N., Prabhakaran, V., Wobber, T., *et al.*, 2008. Design tradeoffs for SSD performance. USENIX Annual Technical Conf., p.57-70.

Appleton, R., 1997. Kernel korner: a non-technical look inside the EXT2 file system. *Linux J.*, **1997**(40es), Article 19.

Ban, A., 1995. Flash File System. US Patent 5 404 485.

Bux, W., 2009. Performance Evaluation of the Write Operation in Flash-Based Solid-State Drives. Technical Report No. RZ3757, IBM Research, Zurich, Rschlikon.

Chen, F., Koufaty, D.A., Zhang, X.D., 2009. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. Proc. 11th Int. Joint Conf. on Measurement and Modeling of Computer Systems, p.181-192. [doi:10.1145/1555349.1555371]

Desnoyers, P., 2010. Empirical evaluation of NAND flash memory performance. *ACM SIGOPS Oper. Syst. Rev.*, **44**(1):50-54. [doi:10.1145/1740390.1740402]

Dirik, C., Jacob, B., 2009. The performance of PC solid-state disks (SSDs) as a function of bandwidth, concurrency, device architecture, and system organization. *ACM SIGARCH Comput. Archit. News*, **37**(3):279-289. [doi:10.1145/1555815.1555790]

Gupta, A., Kim, Y., Urgaonkar, B., 2009. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. Proc. 14th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, p.229-240. [doi:10.1145/1508244.1508271]

Heger, D.A., Quinn, R., 2010. Linux 2.6 IO performance analysis, quantification, and optimization. Proc. Int. Conf. for Performance and Capacity Management-CMG.

Hu, X.Y., Haas, R., 2010. The Fundamental Limit of Flash Random Write Performance: Understanding, Analysis and Performance Modelling. Technical Report No. RZ3771, IBM Research, Zurich, Rschlikon.

Huang, H.H., Li, S., Szalay, A., *et al.*, 2011. Performance modeling and analysis of flash-based storage devices. IEEE 27th Symp. on Mass Storage Systems and Technologies, p.1-11. [doi:10.1109/MSST.2011.5937213]

Iliadis, I., 2010. Performance of the Greedy Garbage-Collection Scheme in Flash-Based Solid-State Drives. Technical Report No. RZ3769, IBM Research, Zurich, Rschlikon.

Jung, D., Chae, Y.H., Jo, H., *et al.*, 2007. A group-based wear-leveling algorithm for large-capacity flash memory storage systems. Proc. Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems, p.160-164. [doi:10.1145/1289881.1289911]

Jung, M., Prabhakar, R., Kandemir, M.T., 2012. Taking garbage collection overheads off the critical path in SSDs. Proc. 13th Int. Middleware Conf., p.164-186. [doi:10.1007/978-3-642-35170-9_9]

Kang, J.U., Jo, H., Kim, J.S., *et al.*, 2006. A superblock-based flash translation layer for NAND flash memory. Proc. 6th ACM & IEEE Int. Conf. on Embedded Software, p.161-170. [doi:10.1145/1176887.1176911]

Kim, J., Kim, J.M., Noh, S.H., *et al.*, 2002. A space-efficient flash translation layer for CompactFlash systems. *IEEE Trans. Consum. Electron.*, **48**(2):366-375. [doi:10.1109/TCE.2002.1010143]

Kim, J., Oh, Y., Kim, E., *et al.*, 2009. Disk schedulers for solid state drivers. Proc. 7th ACM Int. Conf. on Embedded Software, p.295-304. [doi:10.1145/1629335.1629375]

Kim, J., Seo, S., Jung, D., *et al.*, 2012. Parameter-aware I/O management for solid state disks (SSDs). *IEEE Trans. Comput.*, **61**(5):636-649. [doi:10.1109/TC.2011.76]

Kim, J.H., Jung, D., Kim, J.S., *et al.*, 2009. A methodology for extracting performance parameters in solid state disks (SSDs). IEEE Int. Symp. on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, p.1-10. [doi:10.1109/MASCOT.2009.5366154]

Kim, Y., Tauras, B., Gupta, A., *et al.*, 2009. FlashSim: a simulator for NAND flash-based solid-state drives. 1st Int. Conf. on Advances in System Simulation, p.125-131. [doi:10.1109/SIMUL.2009.17]

Konishi, R., Amagai, Y., Sato, K., *et al.*, 2006. The Linux implementation of a log-structured file system. *ACM SIGOPS Oper. Syst. Rev.*, **40**(3):102-107. [doi:10.1145/1151374.1151375]

Lee, J.D., Hur, S.H., Choi, J.D., 2002. Effects of floating-gate interference on NAND flash memory cell operation. *IEEE Electron Dev. Lett.*, **23**(5):264-266. [doi:10.1109/55.998871]

Lee, S., Shin, D., Kim, Y.J., *et al.*, 2008. LAST: locality-aware sector translation for NAND flash memory-based storage systems. *ACM SIGOPS Oper. Syst. Rev.*, **42**(6):36-42. [doi:10.1145/1453775.1453783]

Lee, S.W., Park, D.J., Chung, T.S., *et al.*, 2007. A log buffer-based flash translation layer using fully-associative sector translation. *ACM TECS*, **6**(3), Article 18. [doi:10.1145/1275986.1275990]

Lee, S.W., Moon, B., Park, C., *et al.*, 2008. A case for flash memory SSD in enterprise database applications. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.1075-1086. [doi:10.1145/1376616.1376723]

Lu, Y., Shu, J., Zheng, W., *et al.*, 2013. Extending the lifetime of flash-based storage through reducing write amplification from file systems. Proc. 11th USENIX Conf. on File and Storage Technologies, p.257-270.

Luo, J., Zhao, G., 2007. Solid State Hard Disk. US Patent 764 231.

Maghraoui, K.E., Kandiraju, G., Jann, J., *et al.*, 2010. Modeling and simulating flash based solid-state disks for operating systems. Proc. 1st Joint WOSP/SIPEW Int. Conf. on Performance Engineering, p.15-26. [doi:10.1145/1712605.1712611]

Masuoka, F., Momodomi, M., Iwata, Y., *et al.*, 1987. New ultra high density EPROM and flash EEPROM with NAND structure cell. Int. Electron Devices Meeting, p.552-555. [doi:10.1109/IEDM.1987.191485]

McKusick, M.K., Joy, W.N., Leffler, S.J., *et al.*, 1984. A fast file system for UNIX. *ACM Trans. Comput. Syst.*, **2**(3):181-197. [doi:10.1145/989.990]

Moallem, M., 2008. A Study on the Performance Evaluation of Linux I/O Schedulers. MS Thesis, University of Toronto, Canada.

Mohan, V., Gurumurthi, S., Stan, M.R., 2010. FlashPower: a detailed power model for NAND flash memory. Proc. Conf. & Exhibition on Design, Automation & Test in Europe, p.502-507. [doi:10.1109/DATE.2010.5457154]

Murugan, M., Du, D.H.C, 2011. Rejuvenator: a static wear leveling algorithm for NAND flash memory with minimized overhead. IEEE 27th Symp. on Mass Storage Systems and Technologies, p.1-12. [doi:10.1109/MSST.2011.5937225]

O'Brien, K., Salyers, D.C., Striegel, A.D., *et al.*, 2008. Power and performance characteristics of USB flash drives. Int. Symp. on a World of Wireless, Mobile and Multimedia Networks, p.1-4. [doi:10.1109/WOWMOM.2008.4594868]

Park, J., Yoo, S., Lee, S., *et al.*, 2009. Power modeling of solid state disk for dynamic power management policy design in embedded systems. Proc. 7th IFIP Int. Workshop on Software Technologies for Embedded and Ubiquitous Systems, p.24-35. [doi:10.1007/978-3-642-10265-3_3]

Park, S., Shen, K., 2009. A performance evaluation of scientific I/O workloads on flash-based SSDs. IEEE Int. Conf. on Cluster Computing and Workshops, p.1-5. [doi:10.1109/CLUSTR.2009.5289148]

Park, S., Kim, Y., Urgaonkar, B., *et al.*, 2011. A comprehensive study of energy efficiency and performance of flash-based SSD. *J. Syst. Archit.*, **57**(4):354-365. [doi:10.1016/j.sysarc.2011.01.005]

Pratt, S.L., Heger, D.A., 2004. Workload dependent performance evaluation of the Linux 2.6 I/O schedulers. Linux Symp.

Riska, A., Larkby-Lahet, J., Riedel, E., 2007. Evaluating block-level optimization through the IO path. USENIX Annual Technical Conf., p.247-260.

Rosenblum, M., Ousterhout, J.K., 1992. The design and implementation of a log-structured file system. *ACM Trans. Comput. Syst.*, **10**(1):26-52. [doi:10.1145/146941. 146943]

Sehgal, P., Tarasov, V., Zadok, E., 2010. Evaluating performance and energy in file system server workloads. 8th USENIX Conf. on File and Storage Technologies, p.253-266.

Seo, E., Park, S.Y., Urgaonkar, B., 2008. Empirical analysis on energy efficiency of flash-based SSDs. Proc. Conf. on Power Aware Computing and Systems, p.1-5.

Tweedie, S., 2000. Ext3, journaling filesystem. Ottawa Linux Symp., p.24-29.

Wang, H., Huang, P., He, S., *et al.*, 2013. A novel I/O scheduler for SSD with improved performance and lifetime. IEEE 29th Symp. on Mass Storage Systems and Technologies, p.1-5. [doi:10.1109/MSST.2013.6558426]

Wang, R.Y., Anderson, T.E., 1993. xFS: a wide area mass storage file system. Proc. 4th Workshop on Workstation Operating Systems, p.71-78. [doi:10.1109/WWOS.1993.348169]

Wang, Y.K., Goda, K., Nakano, M., *et al.*, 2011. Performance evaluation of flash SSDs in a transaction processing system. *IEICE Trans. Inform. Syst.*, **94**(3):602-611. [doi:10.1587/transinf.E94.D.602]

Wei, Q.S., Gong, B., Pathak, S., *et al.*, 2011. WAFTL: a workload adaptive flash translation layer with data partition. IEEE 27th Symp. on Mass Storage Systems and Technologies, p.1-12. [doi:10.1109/MSST.2011.5937217]

Yoo, B., Won, Y., Choi, S., *et al.*, 2011. SSD characterization: from energy consumption's perspective. 3rd USENIX Workshop on Hot Topics in Storage and File Systems.