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

A fast and simple system performance emulator for enhanced solid state disks: a case study of long read operations*

Do Yeun KIM[1], Chanik PARK[2], Eui-Young CHUNG[3], Sung Woo CHUNG†‡1
([1]*Division of Computer and Communication Engineering, Korea University, Seoul 13-713, Korea*)
([2]*Samsung Electronics, Yongin-City, Kyunggi-Do 446-701, Korea*)
([3]*School of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, Korea*)
†E-mail: swchung@korea.ac.kr

Received Aug. 16, 2009; Revision accepted Mar. 8, 2010; Crosschecked May 10, 2010

Abstract: In this paper, we propose a fast and simple system emulator, called a system performance emulator (SPE), to evaluate long read operations. The SPE estimates how much system-wide performance is enhanced by using a faster solid state disk (SSD). By suspending a CPU for a certain time during direct memory access (DMA) transfer and subtracting this suspended time from the total DMA time, the SPE estimates the improvement in system performance expected from an enhanced SSD prior to its manufacture. We also examine the relation between storage performance and system performance using the SPE.

Key words: Solid state disk (SSD), Performance emulator, System performance, Direct memory access (DMA), Operating system
doi:10.1631/jzus.C0910505 **Document code:** A **CLC number:** TP333

1 Introduction

The performance of a system relies on the performance of its components such as its CPU, memory systems, and storage devices. Recently, the development of a solid state disk (SSD) based on NAND flash memories has led to a dramatic improvement in the performance of storage devices, especially for read operations (Samsung Electronics, 2007; Mtron Storage Technology, 2008). Table 1 shows the advantages of an SSD over a hard disk drive (HDD) including less weight, remarkable robustness, high reliability, lower power consumption, and outstanding performance.

There have been many studies aimed at enhancing SSDs. A recent survey (Gal and Toledo, 2005a) discussed studies carried out to evaluate algorithms and data structures specially suited for SSDs. To outperform existing flash file systems both in booting time and garbage collection overheads, efficient NAND flash file systems have been proposed (Lim and Park, 2006). For embedded devices, a transactional flash file system has been developed

Table 1 Comparison between SSD and HDD

Parameter	Value/Description	
	2.5″ SATA SSD*	2.5″ SATA HDD
Mechanism type	NAND flash memories	Magnetic rotating platters
Weight	75 g	98 g
Performance	Read: 100 MB/s Write: 80 MB/s	Read: 59 MB/s Write: 60 MB/s
Shock resistance	1500*g*/0.5 ms	325*g*/2.0 ms
Operating temperature	0–70 °C	5–55 °C
Active power consumption	0.41 W	2.0 W
Price	≈\$200/32 GB**	<\$30/32 GB**

* Samsung Electronics, 2007; ** In November 2009. SSD: solid state disk; HDD: hard disk drive

‡ Corresponding author
* Project supported by the Second Brain Korea 21 Project and Samsung Electronics

(Gal and Toledo, 2005b). Hybrid disks are another area of research (Bisson and Brandt, 2007). The techniques proposed by these studies have led to enhanced SSDs.

However, though storage performance is enhanced, the overall system performance may not increase proportionally. For this reason, a system performance emulator (SPE) is necessary for enhanced storage devices prior to their manufacture. To estimate the system performance, conventional full-system simulation frameworks such as Simics (Magnusson *et al.*, 2002) and SIMFLEX (Hardavellas *et al.*, 2004) can be used. However, the complexity of the simulators is so high that they cannot reflect the detailed hardware specification and architecture synchronously. Furthermore, these simulators usually have a long evaluation time (Griffin *et al.*, 2002). Another technique for estimating system performance is storage emulation. Storage emulation offers a solution to the problems of full-system simulators. Timing-accurate storage emulators have been proposed (Noble *et al.*, 1997; Fall, 1999; Griffin *et al.*, 2002), but they are difficult to adopt because they are inaccurate or relatively slow. (In local emulation, the emulator is set on a target system itself, thus requiring some modifications of the target system. In remote emulation, the emulator is set on a second system attached to the target system via a storage interconnect; thus, the target system is not modified. When the emulator runs locally, the emulator uses extra CPU time and memory for emulation, which reduces accuracy. When the emulator runs remotely, it is relatively slow to represent real storage components. Also, it takes a substantial time for the emulator to control the target system, resulting in inevitable inaccuracy.)

In this paper, we propose a fast and simple SPE for long read operations. An SPE does not simulate complex target systems (instead, it runs on a real system) and does not need extra CPU time or memory for emulation itself. In addition, an SPE runs on a target system and does not require remote emulation. For these reasons, an SPE is relatively fast and accurate. SPEs use a novel approach that suspends a CPU for a certain time (depending on the improvement in SSD performance) during direct memory access (DMA) transfer. To obtain the effective execution time, the suspended time is subtracted from the measured execution time. Using an SPE, we examined the relation between storage performance and system performance in several applications such as file read, boot-up, and Web server benchmark.

The proposed technique requires estimating DMA time (the time required for transferring data from the storage device to main memory) to determine the amount of suspended time, when the DMA request is issued. Note that an SSD, based on NAND flash memories, uses the same mechanical parts as an HDD. Hence, the variation in the DMA time is negligible and the data transfer time is proportional to the data size (Chung *et al.*, 2008). Based on this characteristic of the SSD, we can precisely estimate DMA time by looking at the DMA size when the DMA request is issued. This technique is applicable to any other random access based storage device in which the variation in DMA time is quite small.

SPEs work only for long read operations. It is too complex to evaluate the write performance, since there is no way to detect when the DMA write operation is completed; in fact, we know of no method to emulate write operations without dedicated hardware. Fortunately, the main benefit from using an SSD is to improve the read performance rather than the write performance, which implies that applications in which write operations are dominant are not a main target for an SSD. Many techniques (e.g., using a small cache to temporarily store write data) have been proposed to improve the write performance of SSDs (Lee *et al.*, 2007; Yoon *et al.*, 2008), but some concerns remain.

2 System performance emulator

The emulation process of an SPE is accomplished in two steps: (1) DMA traces are collected; (2) improvement of system performance is estimated based on the emulation results.

Step 1: Storage device enhancement generally means that DMA time is reduced by the enhanced storage device. To emulate the enhanced storage, an SPE suspends the CPU for a certain time during DMA transfer: this suspended time depends on the storage performance improvement rate. We ensure that all the cores are halted. Since we know which tasks are target applications on the operating system (OS) by modifying the OS code, we prevent those tasks from running during the suspended time (a meaningless while loop is appended in the OS code that mimics suspension during the suspended time). At that time, the program, which counts elapsed time and compares it to the predefined time, is running. When the elapsed time is the same as the predefined time, the original applications are resumed. Fig. 1 describes the proposed technique by comparing the emulation mode to the normal mode.

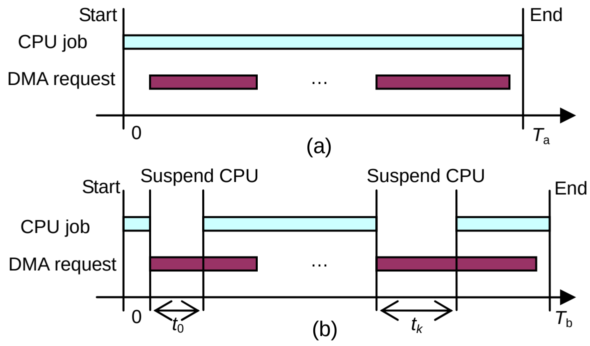

Fig. 1 SPE technique for emulating an enhanced SSD
(a) Normal mode: execution time=T_a; (b) Emulation mode: execution time=$T_b-\sum t_n$

To suspend a CPU during DMA transfer, an SPE should accurately estimate each DMA time when every DMA request starts. An SPE refers to the data size of a DMA request when the DMA request is issued, and then estimates DMA time using the data size. Note that since the SSD is based on random access memory, the variation in DMA time is negligible and the data transfer time is proportional to the data size (Chung *et al.*, 2008). Table 2 shows DMA time in relation to the data sizes of DMA requests measured from a real SSD (Mtron Storage Technology, 2008). The maximum variation rate of the DMA time is less than 2%.

Table 2 DMA time according to the data size of the DMA request

DMA request sector size*	DMA time (μs)
32	235
56	335
160	801
200	981

* One sector: 512 bytes

The CPU suspended time for each DMA request is calculated as follows:

$$T_{\text{suspended}} = T_{\text{DMA}}\left(1-\frac{1}{R_{\text{perf}}}\right), \tag{1}$$

where T_{DMA} is the DMA time for a DMA request and R_{perf} is the storage performance improvement rate.

Step 2: Finally, to measure the improvement in system performance of an enhanced SSD, the SPE calculates the execution time of workload applications. The execution time of an enhanced SSD by emulation is expressed as follows:

$$T_{\text{enhanced}} = T_{\text{end}} - T_{\text{start}} - \sum T_{\text{suspended}}, \tag{2}$$

where $T_{\text{suspended}}$ is the CPU suspended time for each DMA request, and T_{end} and T_{start} are the end time and start time, respectively, of workload applications.

For example, when evaluating the system performance of an SSD that is 50% faster than those currently available, R_{perf} in Eq. (1) is 1.5. In this case, when the sector size of a DMA is 56 sectors (one sector is 512 B), DMA time will be 335 μs (Table 2). According to Eq. (1), the suspended time for the DMA should be 112 μs.

Note that short read operations of less than 10 ms cannot be evaluated using the proposed emulator, because OS level timing granularity is 10 ms, which is greater than short read (e.g., 4 KB) operations. However, as computer technology advances, the OS timing granularity will decrease. The proposed technique will then be applicable to the shorter read. Write operations also cannot be captured in this emulator since on the OS level there is no way of knowing exactly when the write operation is completed in the SSD (the DMA ack. is sent to OS only after data is sent to the buffer in the SSD controller). In addition, if the performance of the SSD is considerably improved (e.g., read throughput is over 1 GB/s), a performance bottleneck may be moved to other parts, such as memory access time and north bridge delay time. In that case, Eq. (1) needs to be modified. However, for now, the proposed technique can be applied effectively to evaluate long read performance.

3 Validation of the system performance emulator

In this section, we validate the accuracy of the SPE, which is implemented with the proposed technique as explained in Section 2. First, we show that the variation in DMA time is negligible on SSDs. Second, we verify the accuracy of the SPE by comparing the emulation results on the SPE with the results on a real enhanced SSD.

3.1 Experimental environments

We used two real SSDs which have different performance. The performance of one SSD (throughput 92.6 MB/s) was 1.98 times faster than that of the other (throughput 46.7 MB/s) in terms of the file read speed (Table 3).

Table 3 Experimental environments

	Specification
CPU	Intel® Core™ 2 CPU 6400 2.13 GHz
Memory	Samsung DDR2-SDRAM 2 GB
OS	Fedora Core 6 (kernel 2.6.18; ext3)
SSD	Samsung PATA 32 GB (throughput 46.7 MB/s)*
	Mtron SATA 16 GB (throughput 92.6 MB/s)**

* Samsung Electronics, 2007: MCAQE32GMPP-0XA; ** Mtron Storage Technology, 2008: MSD-SATA3025

3.2 Variation in DMA time on SSDs

The technique proposed in Section 2 was implemented in the SPE to estimate the DMA time. The SPE checks the DMA request data size when the DMA request is issued. Using the data size, the SPE estimates the DMA time and calculates the time for a CPU suspension. Therefore, to adopt this technique, it must be confirmed that the variation in DMA time is negligible among DMA requests that are issued with the same data size on the same SSD. Fig. 2 describes the DMA time of each DMA request issued by the file read operation for a 500 MB file on an SSD (throughput 92.6 MB/s).

When the sector size is 56, the average DMA time is 335 μs (Table 2) and its standard deviation is 3.94. As the sector size is 200, the average DMA time is 981 μs and its standard deviation is 6.47. These results verify that the standard deviation of DMA time is remarkably small.

Fig. 3 confirms that the variation in large file read time is negligible on SSDs. The maximum variation of file read time on SSDs is less than 1% of the mean file read time. Moreover, we examined boot-up time and also found negligible variation (Fig. 4).

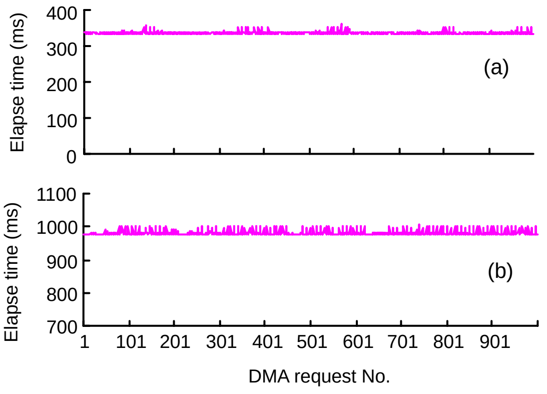

Fig. 2 Variation in DMA time on SSDs according to the sector size of DMA requests
(a) Sector size: 56; (b) Sector size: 200

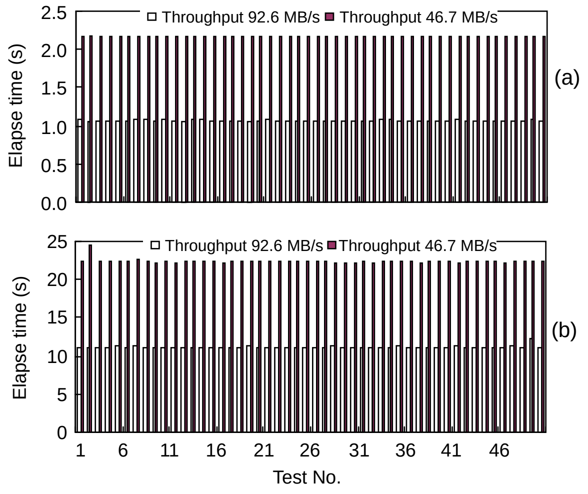

Fig. 3 Variation in file read time on SSDs
(a) File read: 100 MB; (b) File read: 1 GB

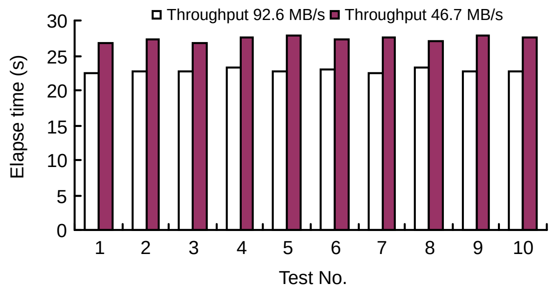

Fig. 4 Variation in boot-up time on SSDs

3.3 Accuracy of the SPE

In this section, we verify the accuracy of the SPE. The enhanced SSD (throughput 92.6 MB/s) is 1.98 times faster than another SSD (throughput 46.7 MB/s) in terms of the file read. Hence, for emulating an enhanced SSD on the low-performance SSD, we should suspend a CPU during 49.6% of each DMA time (49.6% is calculated from Eq. (1)). Fig. 5 shows the results for file read on real SSDs and the emulation results for file read on the SPE.

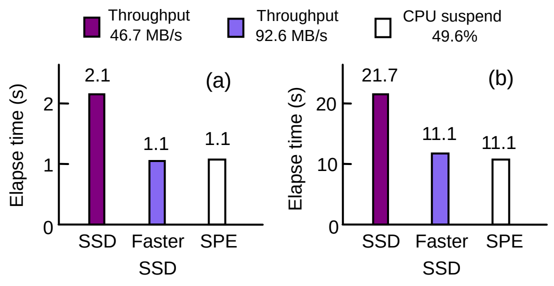

Fig. 5 The results for file read on SSDs and the SPE
(a) File read: 100 MB; (b) File read: 1 GB

The file read time of a 100 MB file on the low-performance SSD (throughput 46.7 MB/s) was 2.114 s and that of a 1 GB file was 21.714 s. On the enhanced SSD (throughput 92.6 MB/s), the file read time of a 100 MB file was 1.072 s and that of a 1 GB file was 11.116 s. The results from the SPE, which emulated an enhanced SSD on the low-performance SSD, showed that the file read time of a 100 MB file and a 1 GB file was 1.077 s and 11.063 s, respectively. Thus, the maximum error between the results from the real SSDs and the emulation results from the SPE was less than 0.5%, which verifies that the accuracy of the SPE is reliable.

4 Estimation of system performance

We estimate the improvement of system performance with various applications, based on the improvement of storage device performance. In this paper, we examine the relation between storage performance and system performance on several applications: file read, boot-up, and Web server benchmark.

4.1 File read

The throughput of the selected SSD was 46.7 MB/s. We ran a file read operation and measured the file read time on four files: 10 MB, 50 MB, 100 MB, and 500 MB. At the same time, the SPE emulated the enhanced storage devices by suspending a CPU during 0%, 50%, and 80% of each DMA time. In this case, the ‘0% (no) suspend’ means that the file read was executed with no CPU suspension. The ‘50% suspend’ means the performance of the SSD was improved by a factor of 2 compared with the performance of the standard SSD.

In the case of the file read, the DMA operations are independent of CPU jobs, and consequently suspending a CPU did not have any effect on the file read time. Fig. 6 shows that the emulation results reflect only on the effect of subtraction of total suspended time: when the suspension rate was 50% (i.e., the SPE emulates a storage device with a two-fold enhancement), the file read time decreased by a factor of about 1/2. Similarly, when the suspension rate was 80% (i.e., the SPE emulates a storage device enhanced by 500%), the file read time decreased by a factor of about 1/5.

The total DMA time accounted for 90%–95% of the execution time of the file read. Thus, file I/O has a large effect on system performance.

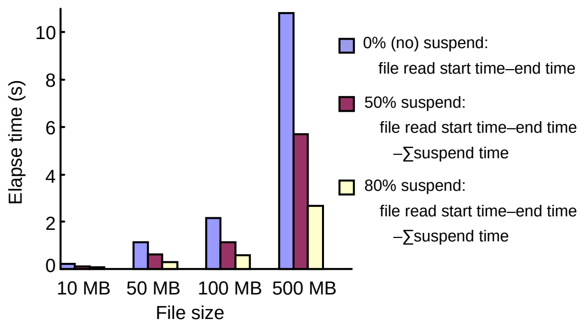

Fig. 6 File read results in relation to the file size and CPU suspend rates

4.2 Boot-up

The accuracy of the SPE for boot-up is shown in Fig. 7. The boot-up time on the low-performance SSD (throughput 46.7 MB/s) was 27.341 s. The SPE emulated the enhanced storage device with the CPU suspend rate of 49.6%. The emulation result was 23.642 s for the boot-up process. The boot-up time was 22.8 s on the real enhanced SSD (throughput 92.6 MB/s). Hence, we also confirm that the SPE is reliable for boot-up (the accuracy was 96.3%).

The total boot-up time was 27.3 s of which 8.1 s was accounted for by the total DMA time. The DMA time accounted for 29.5% of the total boot-up time. Thus, there was less of an improvement in system performance in the case of boot-up than in the case of file read.

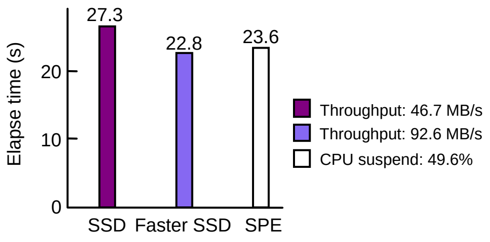

Fig. 7 Boot-up test results on SSDs and the SPE

4.3 Web server benchmark

Table 4 shows experimental environments for a Web server benchmark. Since the maximum network upload bandwidth of our Web server is around 11 MB/s, the throughput of the benchmark was 11.734 MB/s on the low-performance SSD (throughput 46.7 MB/s). The emulation result on the SPE was 12.691 MB/s (Fig. 8). If there is no limit on the network upload bandwidth, the emulation result on the SPE, a throughput of 12.691 MB/s, is expected to be close to the result on the real enhanced SSD. Thus, we should be careful about performance bottlenecks in other system components, when we use an SPE.

Table 4 Experimental environments for a Web server benchmark

	Specification
CPU	Intel® Core™ 2 CPU 6400 2.13 GHz
Memory	Samsung DDR2-SDRAM 2 GB
OS	Fedora Core 6 (kernel 2.6.18)
SSD	Samsung PATA 32 GB (throughput: 46.7 MB/s)*
	Mtron SATA 16 GB (throughput: 92.6 MB/s)**
NIC	Realtek rtl8111 (10/100/1000 M Gigabit Ethernet)
Web server	Apache 2.2.3
Benchmark tool	WebBench 5.0***

* Samsung Electronics, 2007; ** Mtron Storage Technology, 2008; *** VeriTest, 2005

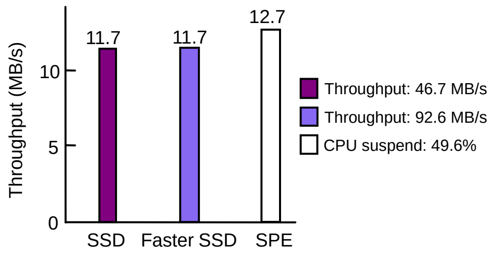

Fig. 8 Web server benchmark results on SSDs and the SPE

The execution time of the Web server benchmark was 30 s of which 4.8 s (16.0%) was the DMA time. This rate (16.0%) was lower than the DMA time rate of boot-up. Hence, the improvement of system performance by the enhanced SSD was lower for the Web server benchmark than for the file read and boot-up.

5 Conclusions

In this paper, we propose a fast and simple system performance emulator (SPE) for evaluating long read operations. The proposed SPE accurately estimates the performance improvement of systems without an existing enhanced SSD. Moreover, the SPE reflects real system environments, as it runs on a real system. The SPE was validated against real SSDs and the accuracy was 99.5% for file read and 96.3% for boot-up. The proposed SPE is expected to provide a cost-effective method for analyzing the effects of enhanced SSDs that are not yet available.

References

Bisson, T., Brandt, S.A., 2007. Reducing Hybrid Disk Write Latency with Flash-Backed I/O Requests. 15th IEEE Int. Symp. on Modeling, Analysis, and Simulation, p.402-409. [doi:10.1109/MASCOTS.2007.57]

Chung, S.W., Lee, J.S., Kim, D.Y., 2008. Utilizing Solid State Disk for Aggressive Processor Power Saving Techniques in Real-Time Applications. Euromicro Conf. on Real-Time Systems, p.5-7.

Fall, K., 1999. Network Emulation in the Vint/NS Simulator. IEEE Symp. on Computers and Communications, p.244-250. [doi:10.1109/ISCC.1999.780820]

Gal, E., Toledo, S., 2005a. Algorithms and data structures for flash memories. *ACM Comput. Surv.*, **37**(2):138-163. [doi:10.1145/1089733.1089735]

Gal, E., Toledo, S., 2005b. A Transactional Flash File System for Microcontrollers. The USENIX Annual Technical Conf., p.89-104.

Griffin, J.L., Schindler, J., Schlosser, S.W., Bucy, J.S., Ganger, G.R., 2002. Timing-Accurate Storage Emulation. USENIX Conf. on File and Storage Technologies (FAST), p.75-88.

Hardavellas, N., Somogyi, S., Wenisch, T.F., Wunderlich, R.E., Chen, S., Kim, J., Falsafi, B., Hoe, J.C., Nowatzyk, A.G., 2004. SIMFLEX: a fast, accurate, flexible full-system simulation framework for performance evaluation of server architecture. *ACM SIGMETRICS Perform. Eval. Rev.*, **31**(4):31-34. [doi:10.1145/1054907.1054914]

Lee, S.W., Park, D.J., Chung, T.S., Lee, D.H., Park, S., Song, H.J., 2007. A log buffer-based flash translation layer using fully associative sector translation. *ACM Trans. Embed. Comput. Syst.*, **6**(3), No. 18. [doi:10.1145/1275986.1275990]

Lim, S.H., Park, K.H., 2006. An efficient NAND flash file system for flash memory storage. *IEEE Trans. Comput.*, **55**(7):906-912. [doi:10.1109/TC.2006.96]

Magnusson, P.S., Christensson, M., Eskilson, J., Forsgren, D., Hallberg, G., Hogberg, J., Larsson, F., Moestedt, A., Werner, B., 2002. Simics: a full system simulation platform. *IEEE Comput.*, **35**(2):50-58. [doi:10.1109/2.982916]

Mtron Storage Technology, 2008. Mtron SSD, Data Sheet. Available from http://www.mtron.net/English/Customer/downloads.asp?sid=download [Accessed on Feb. 20, 2009].

Noble, B.D., Satyanarayanan, M., Nguyen, G.T., Katz, R.H., 1997. Trace-Based Mobile Network Emulation. ACM SIGCOMM Conf., p.51-61. [doi:10.1145/263105.263140]

Samsung Electronics, 2007. Samsung SSD, Data Sheet. Available from http://www.samsung.com/global/business/semiconductor/products/flash/ssd/2008/home/home.html [Accessed on Feb. 20, 2009].

VeriTest, 2005. Benchmark Tool WebBench 5.0. Available from http://cs.uccs.edu/~cs526/webbench/webbench.htm or http://dictionary.zdnet.com/definition/PC+Magazine+benchmarks.html?tag=col1;inner [Accessed on Feb. 20, 2009].

Yoon, J.H., Nam, E.H., Seoug, Y.J., Kim, H., Kim, B., Min, S.L., Cho, Y., 2008. Chameleon: a high performance flash/FRAM hybrid solid state disk architecture. *IEEE Comput. Archit. Lett.*, **7**(1):17-20. [doi:10.1109/L-CA.2007.17]