ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 43, No. 4, pp. 411-418, 2016. 4
http://dx.doi.org/10.5626/JOK.2016.43.4.411

시스템 환경이 Filebench 벤치마크에 미치는 영향 분석

(Analyses of the Effect of System Environment on Filebench Benchmark)

송 용 주 † 김 정 훈 †† 강 동 현 † 이 민 호 † 엄 영 익 †††
(Yongju Song) (Junghoon Kim) (Dong Hyun Kang) (Minho Lee) (Young Ik Eom)

요 약 최근 낸드 플래시 메모리가 널리 보급됨에 따라 기존 파일 시스템의 한계를 보완하고 낸드 플래시 메모리의 장점을 활용하기 위한 파일 시스템 연구가 활발히 진행되고 있다. 이렇게 제안된 파일 시스템들에 대해서는, 일반적으로 벤치마크를 통해 성능 측정이 이루어진다. 서버나 모바일 환경에서 실제 시스템의 성능 측정이 어려울 경우, 벤치마크는 측정하고자 하는 실제 시스템에 대한 직접적인 성능 측정 대신 워크로드를 통해 재현된 환경에서 소프트웨어적 성능 측정을 가능하게 한다. 이 때, 성능 측정 환경이 실제 시스템이 아니기 때문에 측정하는 시스템 환경에 따라서 일정하지 않은 성능 측정 결과를 보인다. 이에 본 논문에서는 파일 시스템의 성능을 측정하는데 흔히 사용되는 벤치마크 중에서 Filebench를 이용하여 여러 가지 시스템 환경에 따른 성능 측정 결과를 살펴보고 측정 결과의 변동이 생기는 원인을 알아본다. 실험 결과, 캐시 내부에 벤치마크 I/O 외의 성능 측정에 불필요한 I/O가 많이 발생할수록 벤치마크의 성능 측정 결과가 떨어지는 것을 확인하였다. 또한 fsync 동작이 포함된 백그라운드 I/O를 동작시키는 경우에는 최대 98.2%의 성능 저하가 발생하는 것을 확인하였다.

키워드: SSD, HDD, 벤치마크, Filebench, fsync, 디스크 I/O

Abstract In recent times, NAND flash memory has become widely used as secondary storage for computing devices. Accordingly, to take advantage of NAND flash memory, new file systems have been actively studied and proposed. The performance of these file systems is generally measured with benchmark tools. However, since benchmark tools are executed by software simulation methods, many researchers get non-uniform benchmark results depending on the system environments. In this paper, we use Filebench, one of the most popular and representative benchmark tools, to analyze benchmark results and study the reasons why the benchmark result variations occur. Our experimental results show the differences in benchmark results depending on the system environments. In addition, this study substantiates the fact that system performance is affected mainly by background I/O requests and fsync operations.

Keywords: SSD, HDD, benchmark, Filebench, fsync, disk I/O

· 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학 ICT연구센터육성지원사업의 연구결과로 수행되었음(IITP-2015-(H8501-15-1015))
· 이 논문은 2015 한국컴퓨터종합학술대회에서 'Filebench 벤치마크 I/O 특성 분석'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 성균관대학교 전자전기컴퓨터공학과
eluphany@skku.edu
kkangsu@skku.edu
minhozx@skku.edu
†† 비 회 원 : 삼성전자 소프트웨어센터 선임 연구원
jhoon20.kim@samsung.com
††† 종신회원 : 성균관대학교 소프트웨어대학 교수(Sungkyunkwan Univ.)
yieom@skku.edu
(Corresponding author임)

논문접수 : 2015년 8월 24일
(Received 24 October 2015)
논문수정 : 2015년 10월 30일
(Revised 30 October 2015)
심사완료 : 2015년 12월 15일
(Accepted 15 December 2015)

정보과학회논문지 제43권 제4호(2016. 4)

1. 서 론

최근 하드웨어 기술의 발전으로 차세대 메모리가 등장하고 있으며 이를 컴퓨터 시스템에 적용하기 위한 연구가 활발하다. 특히 기존에 저장장치로 주로 사용되던 HDD(Hard-Disk Drive)를 대체할 수 있는 낸드 플래시 메모리 기반의 SSD(Solid State Drive)가 주목받고 있다. HDD는 물리적인 탐색시간(Seek Time)과 느린 읽기/쓰기 속도 등의 한계로 높은 성능을 요구하는 시스템에 사용되기 힘들기 때문이다. 반면, SSD는 다수의 낸드 플래시 메모리가 병렬적으로 구성되어 있는 저장장치로 강한 내구성과 저전력을 가진다는 장점이 있다. 또한, SSD는 HDD와 달리 탐색시간이 존재하지 않기 때문에 높은 디스크 I/O 성능을 보이며 일관된 성능을 보장한다. 하지만 SSD는 덮어쓰기가 불가능하며 데이터를 수정하기 위해 erase-before-write 작업이 필요하다는 점과 쓰기 속도가 읽기 속도에 비해 느리다는 단점을 가지고 있다[1,2]. 이런 이유로 최근 SSD의 특징을 효율적으로 활용할 수 있는 YAFFS2[3], JFFS2[4]와 같은 파일 시스템들이 제안된 바 있다.

파일 시스템들의 성능은 공인된 벤치마크를 통해 측정이 가능하다. 벤치마크는 측정 기준을 설정하고 소프트웨어 시뮬레이션을 통해 시스템의 성능을 평가하는 도구이다. 파일 시스템의 디스크 I/O 성능을 평가하기 위해 대표적으로 Filebench[5], IOzone[6], Postmark[7] 등이 사용된다. 이런 벤치마크를 통해 기존의 파일 시스템과의 성능 측정 결과를 비교하여 제안된 파일 시스템이 더 좋은 성능을 가진다는 것을 증명할 수 있기 때문이다. 또한, 벤치마크가 지원하는 다양한 기능을 통해 측정하고자 하는 시스템의 특징을 정확히 파악할 수 있으며 이를 통해 효율적인 유지보수가 가능하고 기존 파일 시스템의 개선을 위한 연구 방향을 정할 수도 있다. 하지만 파일 시스템의 성능을 측정하고 비교 대상과 성능 차이를 확인하기 위해서는 신뢰성 있는 측정 결과가 필요하다. 일반적으로 비교 대상과 최대한 같은 환경에서 실험을 진행하려고 노력하지만 시스템의 상태와 저장장치의 특성 등에 따라 벤치마크의 성능이 일정하지 않을 수 있기 때문이다. 따라서 본 논문에서는 벤치마크를 통해 시스템의 성능을 측정하고자 할 때 고려해야하는 사항과 측정 결과에 영향을 미치는 요인을 실험을 통해 알아본다. 실험에 사용되는 벤치마크는 파일 시스템의 디스크 I/O 성능을 측정하기 위해 흔히 사용되는 Filebench이다. 이는 다양한 워크로드를 제공하는데 본 논문에서는 쓰기 위주 워크로드인 Fileserver, Varmail과 읽기 위주 워크로드인 Webserver, Webproxy를 사용한다. 또한, 시스템의 상태와 저장장치의 특성이 벤치마크의 성능 측정 결과에 어떤 영향을 끼치는지 알아보기 위해 다양한 시스템 상황을 설정하였으며 기존에 저장장치로 흔히 사용되는 HDD와 낸드 플래시 메모리 기반의 SSD를 사용하여 실험을 진행하였다. 실험 결과를 보면 메인 메모리 내부에 벤치마크와 상관없는 페이지가 많이 할당되어 있을수록 벤치마크의 성능 측정 결과가 낮고 측정 결과의 오차가 커진다는 사실을 확인하였다. 또한, 백그라운드 I/O 동작이 벤치마크의 성능 측정 동작을 간섭함으로써 성능 측정 결과에 차이가 발생하는 것과 백그라운드 I/O 동작에 fsync 동작이 포함되어있을 경우 성능 측정 결과의 차이가 더 심하게 발생하는 것을 확인하였다. 그리고 HDD 기반의 시스템보다 SSD 기반의 시스템이 I/O 처리 성능에 대한 측정 결과의 오차가 적게 발생한다는 사실을 확인하였다.

본 논문의 구성은 다음과 같다. 2장에서는 Filebench의 특성과 Filebench에서 제공하는 워크로드, fsync 명령 및 Drop Caches에 대해 알아본다. 3장에서는 시스템 설정과 진행하는 실험에 대해 간단하게 파악한다. 4장에서는 실험 결과를 분석해보며 마지막으로 5장에서는 결론을 내며 논문을 마무리한다.

2. 배경 지식

2.1 Filebench

Sun Microsystems에서 개발된 Filebench는 파일 시스템의 데이터 입출력 성능을 측정하기 위한 벤치마크로 초기에는 솔라리스 운영체제의 성능 분석에 사용되었다. Filebench는 기존에 쓰이던 벤치마크들과 다르게 매우 유연하여 특정 어플리케이션의 행동을 하나하나 지정할 수 있다는 특징이 있다. 파일 크기, 개수, 쓰레드의 개수, I/O 크기, 실행 시간 등 여러 가지 요소들의 설정이 가능해 측정 환경에 따라 적합한 워크로드와 환경을 설정하여 성능 측정이 가능하다. Filebench를 통해 성능을 측정하면 처리량, 지연 시간, 시스템 호출마다 사용되는 CPU 자원 등을 확인할 수 있다. 또한, Filebench는 다수의 워크로드들을 지원하는데 메일, 웹, 파일 서버나 데이터베이스 서버와 같은 복잡한 응용 프로그램을 쉽게 테스트 할 수 있다. Filebench는 기존 배포된 다른 벤치마크들과 비교해 볼 때 쉽게 사용이 가능하고 설치도 빠르게 할 수 있으며 리눅스, FreeBSD와 솔라리스 플랫폼에서 테스트가 가능하다는 장점이 있다[5]. 본 장에서는 Filebench가 제공하는 워크로드 중에서 대표적인 Fileserver, Varmail, Webproxy, Webserver 워크로드에 대해 알아본다.

먼저, Fileserver 워크로드는 간단한 파일-서버 I/O 동작의 성능을 측정할 수 있다. 이 워크로드는 순차적으로 파일 생성-데이터 추가-쓰기/읽기 작업-파일 삭제를

실행한다. 또한, 다수의 사용자가 홈 디렉토리의 파일과 디렉토리에 접근하는 상황을 테스트한다. Fileserver 워크로드는 쓰기 작업이 읽기 작업에 비해 2배 더 많이 발생하는 쓰기 위주의 워크로드이다[5]. 두 번째, Varmail 워크로드는 간단한 메일 서버의 I/O 동작을 시험해볼 수 있다. Varmail 워크로드는 메일 서버에서 파일 안에 각각의 메일이 저장되어 있는 상황을 가정한다. Varmail 워크로드는 멀티 쓰레드 작업을 지원하고, 데이터 생성 및 추가, 그리고 동기화 작업을 지원하며 한 폴더 안에서 읽기/쓰기 작업을 진행한다. 한 가지 특징으로는 평균적인 작업 파일 크기가 16KB로 가장 작은 작업 크기로 디스크 I/O가 동작한다[8]. Varmail 워크로드는 Fileserver와 같이 쓰기 위주의 워크로드이며, fsync 명령을 통해 다른 워크로드들보다 더 많은 디스크 I/O를 발생시키는 특징을 가지고 있다. 세 번째, Webproxy 워크로드는 웹 프록시 서버의 디스크 I/O의 성능을 측정하는데 사용된다. Webproxy 워크로드의 구성은 다음과 같다. 파일 생성-쓰기-종료, 파일 열기-읽기-종료, 디렉토리 트리 내부의 다수 파일 삭제 및 프록시 서버의 로그 데이터 추가 등이 있다. Webproxy 워크로드는 읽기/쓰기 작업의 비율이 5:1로 읽기 위주의 워크로드이다[5]. 마지막으로, Webserver 워크로드는 웹 서버 I/O 동작에 대한 성능을 시험할 수 있다. 읽기/쓰기 작업의 비율이 10:1로 읽기 위주의 워크로드이다. 동시에 여러 파일에 대해 파일 열기-읽기-쓰기를 순차적으로 수행한다[5]. 멀티 쓰레드에 의해 웹 페이지를 읽는 것과 같이 모든 파일을 연속적으로 읽는 방식이다. 이 워크로드는 작은 크기의 파일들에 대해 찾고 읽는 작업뿐 아니라 웹 로그가 증가함에 따라 데이터 및 메타 데이터 또한 동시에 업데이트되는 상황까지 고려하여 성능을 측정할 수 있다[8].

2.2 Drop Caches

Drop Caches 동작은 메인 메모리의 여유 공간을 확보하기 위해 사용된다. Drop Caches 동작을 실행하면 시스템은 데이터가 수정이 되지 않은 페이지, 덴트리와 아이노드처럼 회수 가능한 슬랩 오브젝트와 관련된 페이지를 메모리에서 해제되며 커널 2.6.16 이상의 버전부터 제공되고 있다. 이 동작은 '/proc/sys/vm/drop_caches'의 변수에 입력되는 값에 따라서 달라진다. 변수에 '1'을 입력하면 데이터의 갱신이 없는 페이지 캐시만 해제되고 '2'을 입력하면 덴트리와 아이노드에 관련된 페이지 캐시가 해제된다. '3'을 입력하면 데이터의 갱신이 없는 페이지와 덴트리, 아이노드와 관련된 페이지 캐시들이 모두 해제되어 메모리 공간을 확보할 수 있다[9]. 한편, Drop Caches 동작은 성능의 문제를 일으킬 수 있다. 해제된 페이지들이 사용 중이거나 빈번하게 사용되던 것이라면 이들을 다시 저장장치에서 읽어 들어야 하는데 이는 상당히 많은 I/O와 CPU 작업을 발생시키기 때문이다.

2.3 *fsync* 동작

컴퓨터 시스템의 운영체제는 사용자가 쓰기 작업을 할 때 크게 'Write through'와 'Write back' 방식을 지원한다. 첫 번째, 'Write through' 방식은 쓰기작업이 발생하면 수정된 데이터를 메모리 버퍼뿐 아니라 저장장치에도 반영하는 방식으로 메모리와 저장장치 사이의 데이터 일관성을 유지할 수 있는 장점이 있다. 하지만 저장장치에 쓰기작업이 완료될 될 때까지 CPU가 대기하는 시간이 필요하기 때문에 성능 저하가 생기는 단점이 있다. 이와 달리 'Write back' 방식은 수정된 데이터를 저장장치에 바로 반영하지 않는다. 대신 운영체제는 메모리 버퍼에 쓰기 작업이 반영된 데이터를 캐싱한다. 저장장치에 수정된 데이터를 바로 반영하지 않음으로써 저장장치의 디스크 I/O 량을 감소시킬 수 있으며 시스템의 반응성 역시 향상시킬 수 있다. 'Write back'방식은 일정 시간이 흐른 뒤, 메모리 버퍼에 캐싱되어 있던 데이터를 저장장치에 반영한다[10].

리눅스와 같은 운영체제 경우 'Write back' 방식을 기본적으로 제공한다. 하지만, 'Write back' 방식은 사용자가 데이터를 저장장치에 반영하지 않은 상태에서 전원이 갑자기 꺼지거나 시스템 충돌이 발생하면 메모리에 캐싱되어 있던 데이터는 저장장치에 저장되지 못하고 사라지게 된다. 이와 같은 상황을 피하기 위해 리눅스는 *fsync* 명령을 제공한다. *fsync* 명령은 메모리에 할당된 데이터 중 쓰기작업이 반영된 데이터를 명령이 내려진 즉시 저장장치로 반영한다. 실제로 *fsync* 명령은 모바일 환경에서 주로 사용되는 DBMS(Database Management System)인 SQLite에서 데이터 일관성을 유지하지 위해 사용된다. 저널링 기법을 제공하는 EXT4 파일 시스템에서 SQLite가 명령하는 *fsync*는 시스템의 디스크 I/O 성능을 저하시키는 요인이 된다. 이는 *fsync* 동작이 추가적인 쓰기 작업뿐만 아니라 데이터의 지역성을 떨어트려 디스크 I/O 성능을 저하시키기 때문이다[11]. 예를 들어, EXT4 파일 시스템에서 제공하는 저널링 기법 중에 하나인 Ordered 모드에서 *fsync* 동작을 통해 4KB 쓰기 작업을 수행하게 되면 한 번의 쓰기 요청으로 인해 세 번의 쓰기 작업이 수행된다. Ordered 모드는 메모리 버퍼 안의 페이지가 수정될 때 데이터를 먼저 수정한 뒤, 수정된 페이지와 관련된 메타데이터를 파일 시스템 저널 구역에 기록한다. 이런 동작을 *fsync*와 함께 동작할 경우, 저장장치에는 LBA(Logical Block Address)에 데이터를 업데이트하는 작업이 이뤄지고 수정된 메타데이터를 파일 시스템 저널 구역에 기록하고

최종적으로 저장장치에 수정된 데이터를 반영함으로써 세 번의 쓰기 작업이 발생한다[12].

3. 실험 환경

3.1 시스템 환경

본 논문에서는 다양한 시스템 환경에 따른 성능을 분석하기 위해 아래 표 1과 같이 실험 환경을 구축하였다.

표 1 시스템 환경
Table 1 System Environment

CPU	Intel Core i5-3570K@3.40GHz
Memory	4GB of DRAM
OS	Ubuntu 12.04 LTS
Kernel	Linux Kernel ver. 3.13.0
HDD	Seagate Barracuda ST31000524AS 1TB (Spindle Speed: 7200 RPM, Seek Time: 8.5 ms)
SSD	SAMSUNG SSD 840 Series 120GB (Seq. Write: 470 MB/s, Ran. Write: 90,000 IOPS)

실험에 사용된 파일 시스템으로는 EXT4 파일 시스템을 사용하였으며 EXT4 파일 시스템이 제공하는 저널링 기법 중 Ordered 모드를 기반으로 모든 실험을 진행하였다. 또한, 저장장치의 특성이 벤치마크 성능 측정 결과에 어떤 영향을 미치는지 확인하기 위해 HDD와 SSD를 저장장치로 사용하였으며 아래 그림 1과 같이 각각 디바이스 별로 마운트하여 실험하였다. 각각 마운트되어 있는 경로를 통해 HDD와 SSD로 통하는 디스크 I/O 과정을 분석할 수 있었다.

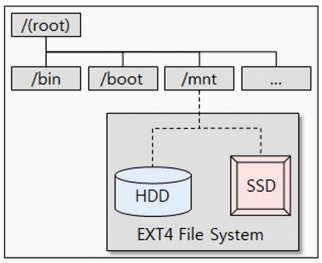

그림 1 실험 환경 구축
Fig. 1 Experimental Environment

3.2 워크로드 설정

본 논문에서는 공인된 벤치마크 중 하나인 Filebench를 사용하였으며 Filebench가 지원하는 워크로드 중 Fileserver, Varmail, Webserver, Webproxy 워크로드를 사용하였다. 표 4에서 확인할 수 있듯이 Fileserver와 Varmail은 쓰기 위주의 워크로드이고, Webserver와 Webproxy는 읽기 위주의 워크로드이다. 실험에 사용한 워크로드 설정은 표 4와 같이 기본 설정으로 하였다[13].

표 2 HDD에서의 쓰기 작업
Table 2 Write Operation Count with HDD

	Fileserver	Varmail	Webserver	Webproxy
Scenario A	9301	453	3316	21657
	448256	40615	76259	103517
	2.07%	**1.12%**	**4.35%**	**20.92%**
Scenario B	8210	467	1907	19366
	409344	43532	76511	103120
	2.01%	**1.07%**	**2.49%**	**18.78%**
Scenario C	4627	279	950	8665
	427008	32611	74746	90143
	1.08%	**0.86%**	**1.27%**	**9.61%**

표 3 SSD에서의 쓰기 작업
Table 3 Write Operation Count with SSD

	Fileserver	Varmail	Webserver	Webproxy
Scenario A	613264	3120	3281	62137
	2648576	219108	76879	177043
	23.15%	**1.42%**	**4.27%**	**35.10%**
Scenario B	631689	1653	2756	60527
	2743808	214164	74931	177925
	23.02%	**0.77%**	**3.68%**	**34.02%**
Scenario C	625277	273	2752	60686
	2718208	216692	76571	179546
	23.00%	**0.13%**	**3.59%**	**33.80%**

표 4 워크로드 특성
Table 4 Characteristics of Workloads

	Fileserver	Varmail	Webserver	Webproxy
Avg. filesize	128KB	16KB	16KB	16KB
# of files	10,000	1,000	1,000	10.000
# of threads	50	16	100	100
Avg. iosize	1024KB	1024KB	1024KB	1024KB
R : W ratio	1:2	1:1	10:1	5:1

4. 실험 결과 분석

벤치마크에 대한 시스템 환경의 영향을 분석하기 위해 다음과 같이 세 가지 실험을 진행하였고, 각각 다섯 번씩 반복하여 결과를 확인하였다. 1) 다양한 시스템 상황에 따라 벤치마크 측정 결과를 확인한다. 2) *fsync* 없이 백그라운드 I/O가 동작하는 상황에서 벤치마크 측정 결과를 확인한다. 3) *fsync*가 포함된 백그라운드 I/O가 동작하는 상황에서 벤치마크 측정 결과를 확인한다.

4.1 시스템 상태에 따른 실험 결과 분석

시스템 상태에 따른 실험 결과를 분석하기 위해 시스템 상태를 세 가지 시나리오로 나누었다. 먼저 'Scenario A'는 파일 시스템을 마운트하고 난 뒤, 메인 메모리의 페이지 캐시를 비운 상태에서 성능을 측정한 결과이다.

Drop Caches는 앞서 2장에서 설명하였듯이 수정되지 않은 페이지를 메모리에서 해제시키는 동작이다. Drop Caches만을 동작했을 경우, 데이터가 수정된 페이지는 해제되지 않는다. 따라서 메인 메모리에 할당되어 있는 모든 페이지들을 해제하기 위해 Drop Caches를 사용하기에 앞서 sync 명령을 사용한다. sync 명령이 동작하면 메모리에서 데이터가 수정된 페이지가 저장장치로 반영되어 메모리 버퍼와 저장장치의 데이터가 동기화된다. 이후 Drop Caches를 실행시켜 메모리에 할당된 모든 페이지들을 해제시킨다. 이를 통해 별도의 프로세스를 실행하지 않은 상태에서 벤치마크를 통해 파일 시스템의 디스크 I/O 성능을 측정한다. ‘Scenario B’는 파일 시스템이 마운트된 직후의 벤치마크 성능 측정 결과를 보여준다. 파일 시스템을 마운트 하는데 필요한 데이터들이 메모리 버퍼에 할당되어 있는 상태이다. 마지막으로 ‘Scenario C’는 Filebench를 통한 I/O 작업을 4번 반복 동작시켜 메인 메모리상에 많은 페이지들을 할당시킨 뒤에 시스템 성능을 측정한다. 이 때 할당되어있는 페이지들은 I/O 성능 측정에 사용되는 데이터와는 전혀 관련이 없어 데이터 캐싱에 대한 효과는 없다. Filebench의 경우, 파일 시스템의 디스크 I/O 성능을 측정할 때 초기 작업으로 파일을 생성하여 쓰기/읽기 등의 작업을 한 뒤 생성된 파일을 모두 삭제하기 때문이다.

실험에 사용한 저장장치는 그림 2와 그림 3과 같이 HDD와 SSD를 사용하였으며 시스템의 성능 측정 결과가 얼마나 변동이 생기는지 앞서 소개한 네 가지 워크로드를 통해 살펴보았다. 또한 표 3과 표 4를 통해 Filebench가 성능을 측정하는 동안 시스템에서 발생하는 쓰기 작업 횟수와 Filebench에서 발생하는 쓰기 작업 횟수를 확인할 수 있다. 시스템 상태에 따라 그림 2와 그림 3에서 보는 것처럼 ‘Scenario C’에서 성능을 측정한 결과가 가장 낮은 것과 측정 결과의 오차가 가장 크게 발생하는 것을 확인할 수 있다. 특히, ‘Scenario C’ 상황에서 HDD 기반으로 실험한 경우 측정 결과들 간의 오차가 최대 9.7%의 오차가 발생하였으며 SSD 기반의 실험에서는 최대 3.0%의 오차가 발생하였다. 이는 메인 메모리에 할당된 페이지 중 Filebench와 상관없는 페이지가 많을수록 디스크 I/O 성능이 낮게 측정된다는 것이다. ‘Scenario B’의 경우 시스템을 부팅한 후에 파일 시스템을 마운트하는 과정에서 필요한 슈퍼블록, 아이노드, 덴트리 등을 구성하기 위한 데이터들을 메인 메모리로 할당된다. 이는 벤치마크가 I/O를 수행하는 과정에서 메모리 버퍼에 할당된 불필요한 페이지들로 추가적인 디스크 I/O가 발생하여 시스템의 성능이 낮게 측정되는 요인이 된다. ‘Scenario C’의 경우 이전 4번 반복된 벤치마크 동작으로 ‘Scenario B’의 마운트 과정에서 메모리에 할당되는 페이지보다 훨씬 많은 양의 불필요한 페이지가 할당된다. 따라서 ‘Scenario B’보다 더 큰 성능 저하를 확인할 수 있다. 실제로 표 3과 표 4에서 보듯 ‘Scenario C’의 경우에서 시스템의 I/O 중 Filebench I/O의 비율이 가장 낮은 것을 확인하였다. 그림 2를 보면 ‘Scenario A’ 대비 ‘Scenario C’의 성능 저하가 Fileserver와 Varmail의 경우 최대 22.8%, 21.6%까지 발생하였고 Web server와 Webproxy의 경우 최대 7.3%, 56.4%의 성능 저하를 보였다. 또한, 그림 3에서 보듯 플래시 메모리 기반의 SSD를 저장 장치로 사용하는 시스템에서는 기계적인 탐색시간이 없기 때문에 쓰기 위주의 워크로드와 읽기 위주의 워크로드의 구분 없이 3%의 오차범위 내에서 일관된 성능을 보였다.

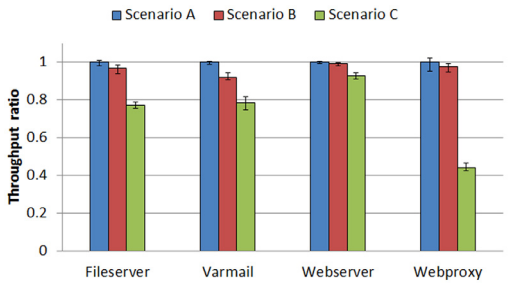

그림 2 HDD에서의 벤치마크 측정 결과

Fig. 2 I/O Performance of Filebench with HDD

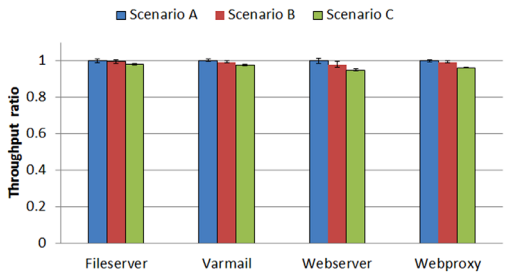

그림 3 SSD에서의 벤치마크 측정 결과

Fig. 3 I/O Performance of Filebench with SSD

4.2 백그라운드 I/O 동작에 따른 실험 결과 분석

본 장에서는 백그라운드 I/O 동작이 벤치마크의 성능 측정 결과에 어떤 영향을 미치는지 확인하기 위해 별도의 프로세스를 실행시키지 않는 상태에서 측정한 벤치마크의 성능 측정 결과와 백그라운드 I/O가 동작하는 상황에서 벤치마크의 성능 측정 결과를 비교하였다. 별도의 프로세스를 실행시키지 않은 상태의 벤치마크 성능 결과는 4.1장에서 측정한 ‘Scenario A’의 결과를 사용하였으며 백그라운드 I/O는 Filebench 수행 중 FIO 벤치마크[14]를 통해 100KB 파일을 계속해서 생성하도록 하였다.

그림 4에서는 HDD 기반의 시스템에서 백그라운드 I/O 동작의 유무에 따른 측정 결과를 보인다. 쓰기 위주의 워크로드인 Fileserver와 Varmail의 경우 각각 17.5%와 90.2% 성능 저하가 발생하였고, 읽기 위주의 워크로드인 Webserver와 Webproxy의 경우 각각 11.7%, 49.7%의 성능 저하를 보였다. 또한, SSD 기반의 시스템에서 백그라운드 I/O가 동작하는 상황에서의 벤치마크 성능 결과를 보면, 그림 5에서 보는 것처럼 Fileserver와 Varmail 워크로드 경우 각각 8.7%, 98.2% 성능 저하가 발생하였고, Webserver와 Webproxy 워크로드의 경우 각각 19.2%, 19.5% 성능 저하를 보였다. 이 실험을 통해 백그라운드 I/O가 동작하는 상황에서 벤치마크를 실행하면 시스템의 성능을 정확히 측정하기 힘들다는 사실을 알 수 있다. 이는 백그라운드 I/O 동작이 벤치마크의 I/O 동작을 간섭을 하기 때문이다.

실험 결과를 보면 워크로드의 특성에 따라 벤치마크의 성능 측정 결과가 달라진다는 것을 확인할 수 있다. 특히 Varmail의 경우는 다른 워크로드들과 다르게 워크로드 자체에 *fsync* 동작이 포함되어 있다. *fsync*는 디스크 버퍼 안에 저장된 데이터를 디스크에 모두 반영되도록 하는데 Varmail 워크로드의 *fsync*가 백그라운드 I/O 동작으로 인해 디스크 버퍼에 저장된 데이터까지 처리함으로써 'Scenario A'에서 보다 더 많은 데이터를 처리한다. 이 때문에 다른 워크로드들에 비해 Varmail 워크로드가 더 큰 성능 저하를 보이게 된다. 읽기 위주의 워크로드 중에서 Webserver에 비해 Webproxy의 성능 측정 결과의 차이가 더 큰 이유는 Webserver는 읽기/쓰기 비율이 10:1인 반면 Webproxy의 경우는 읽기/쓰기 비율이 5:1로 Webserver보다 Webproxy가 더 많은 쓰기 작업을 하기 때문이다. 그림 5에서 보듯 높은 성능의 SSD를 기반으로 하는 시스템은 백그라운드 I/O도 동시에 빠르게 처리하기 때문에 HDD 기반의 시스템보다 백그라운드 I/O 동작이 벤치마크 측정 결과에 더 적은 영향을 미치는 것을 확인할 수 있다.

4.3 *fsync* 명령에 따른 실험 결과 분석

본 실험에서는 *fsync* 명령이 포함된 백그라운드 I/O 동작이 벤치마크의 성능 측정 결과에 어떤 영향을 미치는지 확인해본다. 앞서 언급한 실험과 마찬가지로 'Scenario A'의 성능 측정 결과와 함께 비교 및 분석한다. 백그라운드 I/O 동작을 수행하고 있는 FIO 벤치마크에서 지원하는 옵션을 통해 *fsync* 명령을 추가하였다.

그림 6과 그림 7에서 보는 것처럼 백그라운드 I/O 동작에 *fsync* 명령이 포함되어 있을 때 *fsync* 명령이 없을 때보다 더 큰 성능 저하가 발생하는 것을 확인하였다. HDD 기반의 시스템과 SSD 기반의 시스템에서 File server

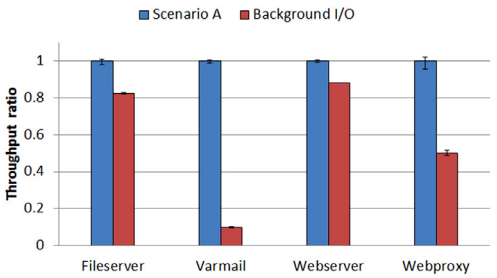

그림 4 HDD에서의 백그라운드 I/O에 따른 측정 결과
Fig. 4 I/O Performance of Filebench with Background I/O in HDD systems

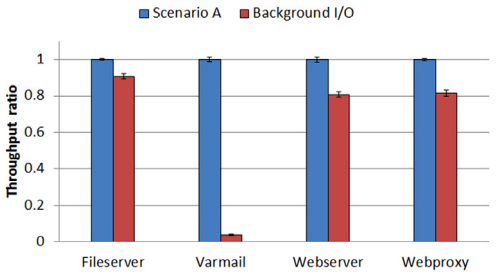

그림 5 SSD에서의 백그라운드 I/O에 따른 측정 결과
Fig. 5 I/O Performance of Filebench with Background I/O in SSD systems

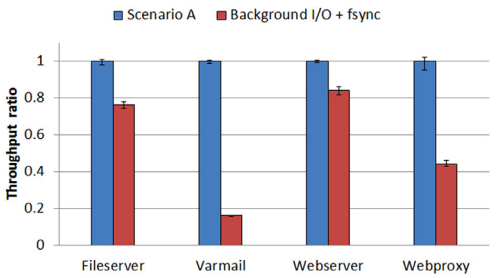

그림 6 HDD에서의 fsync 명령에 따른 측정 결과
Fig. 6 I/O Performance of Filebench with fsync in HDD systems

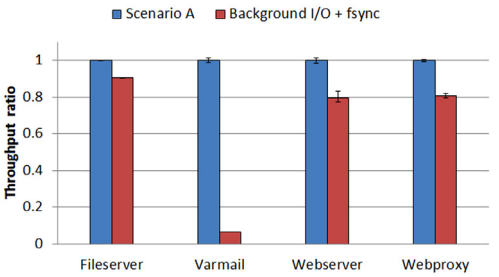

그림 7 SSD에서의 fsync 명령에 따른 측정 결과
Fig. 7 I/O Performance of Filebench with fsync in SSD

의 측정 결과는 각각 5.8%, 0.8% 성능 저하가 발생하였으며 Webserver의 측정 결과는 4.0%, 1.5%의 성능 저하가 발생하였다. Webproxy 역시 5.9%, 1.1%의 성능 저하가 발생한 것을 확인할 수 있었다. 하지만, Varmail의 경우 HDD 기반의 시스템에서는 6.5%, SSD 기반의 시스템에서는 2.8% 가량을 성능 향상이 있는 것을 확인할 수 있었다. 이는 Varmail과 백그라운드 I/O를 동작하는 FIO 벤치마크 모두 *fsync*가 동작하면서 나타는 결과이다. 백그라운드 I/O를 동작하는 FIO의 *fsync* 명령이 Varmail의 *fsync*가 처리해야하는 데이터 중 일부를 저장장치에 반영하여 Varmail의 *fsync*가 단독으로 디스크 버퍼의 데이터를 저장장치에 반영할 때와 달리 FIO 벤치마크의 *fsync*와 함께 동작하면서 Varmail의 *fsync*가 처리하는 디스크 버퍼의 데이터가 더 적기 때문이다.

5. 결 론

본 논문에서는 다양한 상황의 시스템에서 Filebench를 통해 디스크 I/O 성능을 측정하고 그 결과를 분석해 보았다. 또한, 저장장치의 특성이 벤치마크 성능 측정 결과에 어떤 영향을 미치는지 확인하기 위해 HDD와 SSD 두 가지 저장장치를 사용하여 실험을 진행하였다. 실험 결과를 보면 시스템의 상태에 따라 벤치마크의 성능 측정 결과가 달라지는 것을 확인할 수 있었으며 백그라운드 I/O가 동작하는 상황에서 벤치마크를 통해 디스크 I/O의 성능을 측정하였을 때 백그라운드 I/O 동작의 간섭으로 인해 벤치마크의 성능 측정 결과가 더 낮을 것을 확인하였다. 특히 *fsync* 동작이 포함된 백그라운드 I/O가 동작이 벤치마크의 성능 측정 결과에 더 큰 영향을 미치는 것을 확인하였다. 한편, 모든 측정 결과에서 보듯 HDD와 달리 물리적인 탐색 시간이 없는 플래시 메모리 기반의 SSD를 저장장치로 사용하는 시스템에서 측정한 벤치마크 측정 결과가 HDD 기반 시스템보다 더 낮다는 사실을 확인하였다. 실험 결과들을 통해 저장장치의 특성과 시스템 상황에 따라 측정되는 벤치마크 결과가 달라지는 것을 확인하였다. 따라서 파일 시스템의 디스크 I/O 성능을 측정할 때 신뢰성을 지닌 결과를 추출하기 위해서는 저장장치 특성과 시스템의 상황을 고려하는 자세가 필요하다.

References

[1] S. Park and K. Shen, "FIOS: A Fair, Efficient Flash I/O Scheduler," *Proc. of the USENIX Conference on File and Storage Technologies*, pp. 155-170, 2012.

[2] C. Feng, D. Koufaty, and X. Zhang, "Understanding Intrinsic Characteristics and System Implications of Flash Memory based Solid State Drives," *Proc. of the ACM SIGMETRICS*, pp. 181-192, 2009.

[3] (2012, March 28). How Yaffs Works [Online]. Available: http://www.yaffs.net (downloaded 2015, AUG. 1)

[4] D. Woodhouse, "JFFS: The Journaling Flash File System," *Proc. of the Ottawa Linux Symposium*, 2001.

[5] (2014, July 17). Filebench [Online]. Available: http://filebench.sourceforge.net/wiki/index.php/Filebench (downloaded 2015, April. 15)

[6] (2006, October 28). Iozone Filesystem Benchmark [Online]. Available: http://www.iozone.org/

[7] (1997, August 10). Postmark: A New File System Benchmark [Online]. Available: http://www.netapp.com/tech_library/3022.html

[8] P. Sehgal, V. Tarasov, and E. Zadok, "Evaluating Performance and Energy in File System Server Workloads," *Proc. of the USENIX Conference on File and Storage Technologies*, pp. 253-266, 2010.

[9] S. Sridhar, D. Kondamudi, H. Kotha, and N. Krishna, "A Study on The Effect of Cache Memory in Computing Environment with Performance Analysis," *Journal of Scientific & Engineering Research*, Vol. 4, No. 7, pp. 1077-1083, 2013.

[10] A. Hatzieleftheriou and S. V. Anastasiadis, "Host-side Filesystem Journaling for Durable Shared Storage," *Proc. of the USENIX Conference on File and Storage Technologies*, pp. 59-66, 2015.

[11] D. Jeong, Y. Lee, and J. S. Kim, "Boosting Quasi-Asynchronous I/O for Better Responsiveness in Mobile Devices," *Proc. of the USENIX Conference on File and Storage Technologies*, pp. 191-202, 2015.

[12] S. Jeong, K. Lee, S. Lee, S. Son, and Y. Won, "I/O Stack Optimization for Smartphones," *Proc. of the USENIX Conference on Annual Technical Conference*, pp. 309-320, 2013.

[13] E. Lee, H. Bahn, and S. H. Noh, "Unioning of The Buffer Cache and Journaling Layers with Non-volatile Memory," *Proc. of the USENIX Conference on File and Storage Technologies*, pp. 73-80, 2013.

[14] (2006, September 05). Flexible I/O Tester [Online]. Available: https://github.com/axboe/fio

송 용 주

2015년 한국산업기술대학교 컴퓨터공학과 학사. 2015년~현재 성균관대학교 전자전기컴퓨터공학과 석사과정. 관심분야는 스토리지 시스템, 운영체제

김 정 훈

2010년 성균관대학교 컴퓨터공학과 학사
2012년 성균관대학교 휴대폰학과 석사
2016년 성균관대학교 IT융합학과 박사
2016년~현재 삼성전자 소프트웨어센터 선임 연구원 재직. 관심분야는 시스템 소프트웨어, 운영체제, 가상화 기술, 파일 시스템

강 동 현

정보과학회논문지
제 43 권 제 1 호 참조

이 민 호

2013년 성균관대학교 전자전기공학부 학사. 2015년 성균관대학교 전자전기컴퓨터공학과 석사. 2015년~현재 성균관대학교 전자전기컴퓨터공학과 박사과정. 관심분야는 운영체제, 가상화 기술, 파일 시스템

엄 영 익

정보과학회논문지
제 43 권 제 1 호 참조