7

DRAM 기반 메모리 저장장치 시스템을 위한 파일시스템 성능평가

백석대학교 ▌강 윤 희
한국전자통신연구원 ▌정 승 국

1. 서 론

최근 CPU 성능은 고속의 내부버스 및 다중 코어 프로세서의 등장으로 처리성능이 개선되고 있으며, FC (Fiber channel), iSCSI 기반의 저장 네트워크인 SAN은 기업내의 저장시스템의 TCO(Total Cost of Ownership)을 줄일 수 있도록 지원하고 있다[4]. 그러나 디스크 입출력 성능은 프로세스 처리속도 및 네트워크 전송률의 증가와 비교하면 상대적으로 느리며, 전체 저장시스템의 병목의 원인이 된다.

1980년대 등장한 SSD(Solid State Disk)는 일반 하드디스크와는 달리 반도체메모리를 이용하여 정보를 저장하는 장치를 말하며, 지금까지는 가격 경쟁력이 부족했기 때문에 고도의 안정성과 높은 데이터처리 속도가 요구되는 군수, 항공, 선박 등의 특수 분야에서 시장을 형성하고 있었다. SSD는 HDD의 기계적 동작부분이 전혀 없기 때문에 HDD보다 안정성이 있고 높은 데이터 전송속도를 가지고 있다[1,2].

SSD의 첫 번째 유형은 플래시 메모리(Flash memory) SSD이다. 플래시 메모리는 비휘발성 메모리의 일종으로 전기적인 방법으로 정보를 자유롭게 입출력할 수 있으며, 전력소모가 적고 고속프로그래밍이 가능하기 때문에 이미 노트북 등에 시장을 형성하고 있는 기술이다. 두 번째 유형인 메모리배열 SSD는 저장 매체로써 DRAM을 사용하는 휘발성의 저장장치이다. RAM 칩으로부터 직접 데이터를 저장하고 액세스하기 때문에 기존의 마그네틱 장비에 비하여 빠른 속도를 가질 수 있다. DRAM은 휘발성이지만 내부 배터리 시스템과 디스크 백업시스템의 통합에 의하여 스토리지 기능을 수행한다. DRAM 기반 메모리 저장장치의 특징은 빠른 액세스 타임, 광대역처리율, 높은 IOPS(I/O per second), 성능대비 저가격의 장점과 GB 당 높은가격, 휘발성이라는 단점이 있다.

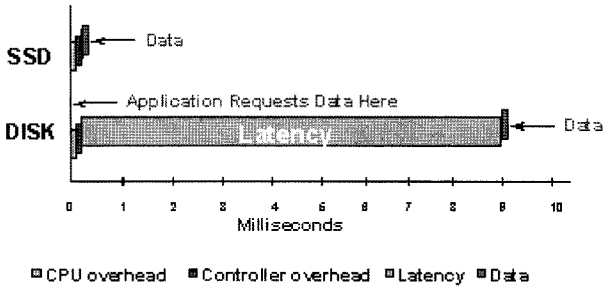


그림 1 SSD와 디스크와의 성능비교

이러한 SSD 기술의 등장은 낮은 지연시간(latency time), 0 회전시간(rotational delay)으로 이상적인 임의 데이터(random data) 접근이 가능하며, 높은 IOPS을 얻을 수 있다. 또한 물리적인 특성으로 인해 저전력 및 고내구성의 이점을 제공한다. 그림 1은 SSD와 디스크와의 성능 비교를 보인 것이다.

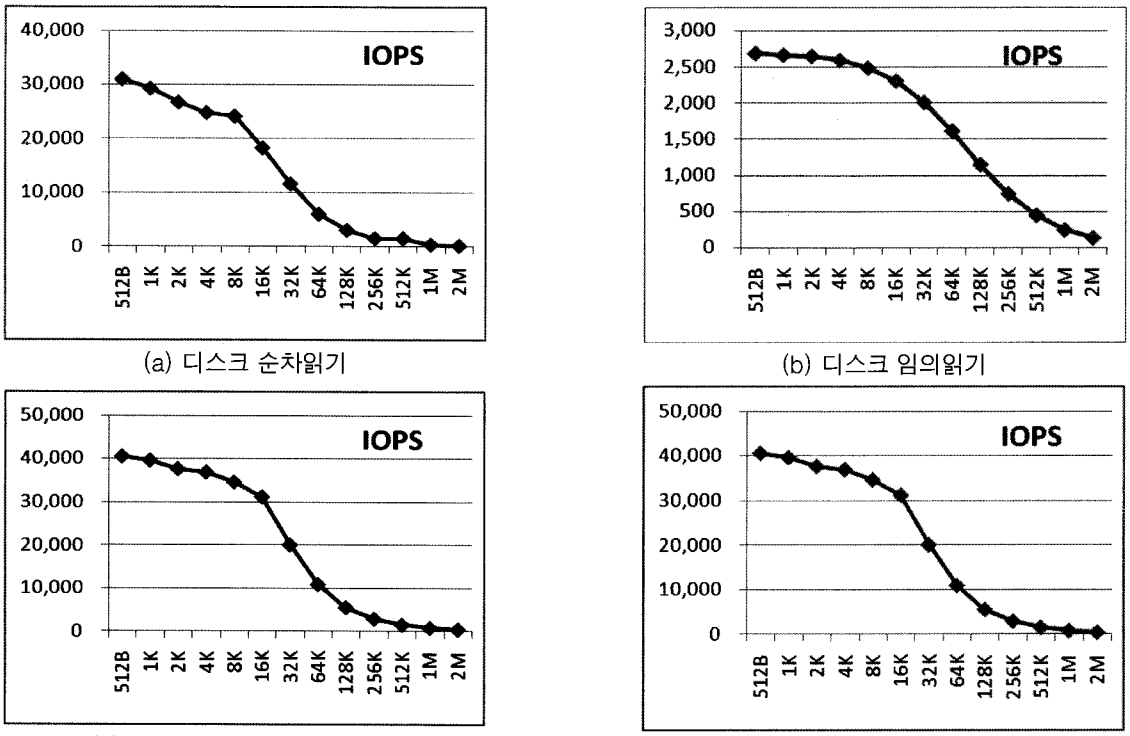
본 고에서는 한국전자통신연구원에서 개발중인 DRAM 기반 차세대 저장장치(Next Generation Storage, NGS) 시스템의 성능평가를 리눅스 운영체제 환경에서 수행한 후 결과를 기술한다. 본 실험은 크게 두 가지로, 첫 번째는 마이크로 벤치마크 프로그램인 Bonne++를 사용하여 HDD와 선행시스템의 성능을 평가하고 분석하였다[17]. 두 번째로는 마크로 벤치마크 도구인 PostMark 벤치마크 도구를 사용하여 주요 리눅스 파일시스템을 대상으로 하며, 해당 파일시스템은 커널 모듈의 형태로 구성한 후 성능평가를 수행한다[11].

본고의 구성은 다음과 같다. 제 2장에서는 관련연구에 대해 설명하고 제 3장에서는 성능평가를 하고 제 4장에서는 결론을 기술한다.

2. 관련연구

2.1 DRAM 기반 메모리 저장 장치

디스크 기반 저장장치는 디스크가 갖는 매체 제약에



(a) 디스크 순차읽기 (b) 디스크 임의읽기

(c) NGS 선행시스템 순차읽기 (d) NGS 선행시스템 임의읽기

그림 2 디스크 장치와 메모리 저장장치의 성능 비교

따른 성능 저해 요소인 디스크 탐색 시간(seek time)을 가지며, 이는 전체 I/O 시간의 80%를 차지하고 있다. 기존의 유닉스 및 리눅스 기반의 디스크 기반 파일 시스템은 I/O성능 향상을 위해 버퍼캐쉬(Buffer cache)를 활용하여 성능을 개선하고 있다[3,5,6,7].

플래시 메모리는 비휘발성(non-volatile), 빠른 접근 속도, 저전력, 소형 및 경량의 특징으로 인해 다양한 분야에서 활용되고 있다. 이런 이유로 플래시 메모리는 게임기, 카메라, PDA, 휴대폰 및 셋탑박스 등 휴대용 전자기기에서 저장 매체로 활용되고 있으며, 최근에는 노트북 컴퓨터의 저장매체로 디스크를 대체하고 있다. 그러나 플래시 메모리는 매체의 특성상 쓰기 위해 기존의 내용을 갖는 블록을 지우고 써야 하는 등의 제약으로 인해 기록연산 지연 등의 문제점이 있다.

이의 대안으로 플래시 메모리 보다 성능 및 확장성이 뛰어난 DRAM 기반 메모리 저장시스템의 구성은 플래시 메모리의 연산지연의 제약 사항을 해결할 뿐만 아니라 읽기와 쓰기연산 속도향상도 기대할 수 있다. 그림 2는 SAN 환경에서 디스크 저장장치와 NGS 선행시스템을 대상으로 한 성능평가의 결과를 보인 것으로 디스크 장치의 경우 임의읽기의 경우 순차읽기에 비해 10%의 IOPS을 보인 반면 DRAM 기반 메모리 장치인 NGS는 순차읽기와 임의읽기의 차이가 없음을 알 수 있다.

해외의 DRAM 기반 메모리 저장장치는 TMS (Texas Memory Systems), SD(Solid Data Systems) 등 일부 업체에서 개발이 진행 중이다. SD사에서는 DRAM 기반 메모리 저장장치를 SAN(Storage Area Network)에서 RAID와 결합하여 파일 캐쉬에 활용하고자 하는 연구를 진행하고 있다[1,2].

국내에서도 기업 및 정부 부처에서는 DRAM 기반 메모리배열 기반 저장시스템인 NGS에 대한 개발을 진행하고 있다. DRAM은 휘발성(volatile)의 매체 특성으로 인해 저장된 정보의 유지를 위해서는 지속적인 전원 공급이 필수적으로 이루어져야 하며 백업을 위한 시스템이 요구된다. 현재 개발되어 운영중인 운영체제의 파일시스템은 디스크 기반의 저장장치를 기반으로 하고 있으므로 NGS를 위한 파일시스템의 개발을 위해서는 I/O 성능 평가가 필수적이다.

2.2 저널링

파일시스템은 저장공간 상에 파일에 대한 내용과 별도로 다양한 내부적인 자료구조를 저장한다. 이러한 메타데이터는 파일의 내용에 대한 접근 및 처리를 필요로 하는 파일시스템에 대한 정보를 제공한다[6,14,15]. 처리 성능과 디스크 접근 시간 간의 차이점이 커지고 있는 반면 주메모리의 크기의 증가에 따라 파일 시

스템은 디스크 접근을 피하고 디스크 지연을 은익하기 위해 캐싱 기술을 도전적으로 채용하고 있다[14]. 일 예로, write() 호출에 의해 디스크에 해당 변경 또는 추가 시의 데이터 접근 경로를 보인 것이다. 사용자 응용에서의 write() 호출에 의해 파일시스템은 변경 부분을 캐쉬 내에 반영하게 된다. 그러나 캐쉬 상의 변경 블록은 즉시 디스크에 쓰지 않고 현재 캐쉬 내의 블록이 변경되었음을 변경(dirty) 상태로 표시하게 된다. 이와 같은 쓰기 지연(delayed write)는 빈번한 쓰기 연산의 일괄쓰기를 통해 성능을 향상시킨다.

그러나 버퍼의 내용을 디스크에 쓰기 전에 갑작스러운 정전 등과 같은 장애로 인해 파일시스템에 일관성을 잃게 되며, 재부팅 후에는 파일시스템의 비정상적인 상태에 놓이게 된다. 즉, 캐싱 된 데이터 블록에 대한 정보가 디스크 상의 파일시스템에 반영되었으나 이후 시스템의 장애로 인해 i-node의 내용과 해당 데이터블록이 디스크 상의 파일시스템에 반영되지 못하는 경우가 발생하며, 시스템은 일관성을 유지하지 못하게 된다.

파일시스템의 일관성을 유지하기 위해 여러 방법이 개발되었다. Fsck 를 사용한 파일시스템 일관성 유지 기법은 모든 메타데이터의 일관성을 검사하기 위해 최소한 수분에서 수시간이 소요되게 되며, 이 과정 동안에는 파일시스템을 사용할 수 없게 된다[15]. Fsck 작동은 시스템의 가동시간(uptime)이 중요시되는 데이터센터와 같은 환경에서는 활용하기 어려운 문제점을 갖는다. 이를 해결하기 위한 저널링 파일시스템(journaling file system)은 데이터베이스의 트랜잭션 개념을 이용하여 파일에 대한 연산을 로깅한 후 결함 발생시 빠르게 회복시킬 수 있는 메커니즘을 제공한다.

많은 파일시스템에서는 다양한 형태의 저널링 모드를 제공한다. 일례로 Ext3에서는 저널링 처리를 위해 3가지 종류의 모드가 제공되며 모드 간에 성능차이점을 보인다. 상세한 실험 결과는 3장에서 내용을 기술하도록 하였다.

3. 선행 시스템 성능평가

3.1 성능평가 개요

NGS 전용 파일시스템의 설계를 위한 성능평가는 대용량의 저장 시스템을 지원하며, 빠른 회복이 가능한 리눅스 기반의 저널링 파일시스템으로 한정한다. 본 실험에서는 파일시스템의 성능평가를 위해 두 종류의 벤치마크도구를 사용하여 실험을 수행한다.

성능평가를 위한 파일시스템은 선행시스템 상에 개별 파티션으로 구성한다. 개별 파트션은 16Gbyte로 설정하였으며, 파티션 내에 파일시스템을 생성하고 마운트한 후 성능 실험을 수행하였다. 대상 저장장치는 태진인포텍의 Jetspeed로 SCSI 기반의 전용 장치 드라이버를 통해 입출력이 이루어진다. 표 1은 성능 평가를 위한 대상시스템의 하드웨어 플랫폼 및 운영체제 구성을 기술한 것이다.

3.2 벤치마크 도구를 사용한 성능 평가 실험

3.2.1 파일 시스템 마이크로 벤치마크

마이크로 벤치마크는 순차적인 읽기/쓰기, 다중스트림의 읽기/쓰기, 임의 읽기/쓰기, 파일 생성/삭제, 파일 메타데이터 연산 등의 성능측정을 목적으로 한다. 마이크로 벤치마크 프로그램의 설계는 파일시스템 실행의 단일측면의 값을 명확하게 측정하여야 한다[11,16,17]. 마이크로 벤치마크는 파일복사 또는 반복적인 주소록 목록과 같은 파일체계 기능의 작은 부분을 반복적으로 수행한다.

마이크로 벤치마크의 측정값은 파일시스템 운영시의 파일 시스템 작업부하의 변화에 따른 측정값에 비해 민감하지 않은 특징이 있다. 이 벤치마크는 파라

표 1 대상 시스템

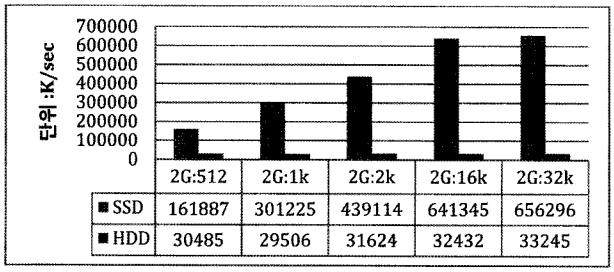
구성요소	사양	특징
CPU	Intel(R) Xeon(R) CPUE5472@ 3.00GHz 64 bit Xeon 쿼드코어X 2	L1 I cache: 32K L1 D cache: 32K L2 cache: 6144K X 2
메모리	FBDIMM DDR2 1GB(PC2-6400) X 8	
프로세서 버스	FSB 1600MHz	
OS 디스크	Seagate S-ATA2 500GB	7200 rpm 16 MB disk buffer
SSD 장치	Jetspeed 16G	
외부 데이터 버스	PCI Express	2.5Gb/s:Width x4
운영체제	RHEL 4_update 2버전	Linux kernel 2.6.20

매터의 범위를 추가하여 특정한 환경 또는 구성요소의 의존성을 평가할 수 있다. 예를 들어서 파일을 읽을 때는 버퍼캐쉬와 읽기방식에 관한 두 가지 파라매터에 의존한다. 첫 번째는 버퍼캐쉬에 데이터의 적재 유무이고 두 번째로는 프로그램이 파일을 순차적으로 읽을 것 인지 랜덤하게 읽을 것 인지 하는 것이다. 두 가지 측정값은 다른 파라매터의 벤치마크 결과에 영향을 준다. 또한 파일 읽기는 디렉터리 계층 어디에 위치했는지 뿐만 아니라 파일의 수와 크기에도 의존한다.

선행시스템과 HDD의 성능평가는 성능평가를 위한 파일시스템은 ext3를 대상으로 하였다. 그림 3에서 그림 5의 SSD는 선행시스템을 표현한다.

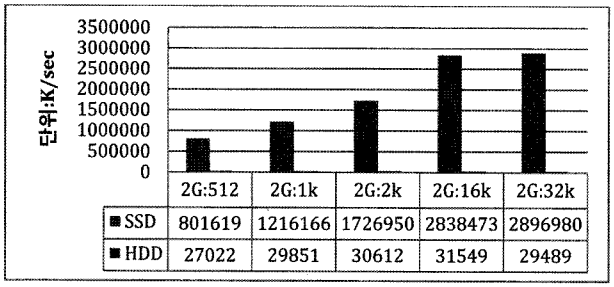
그림 3은 디스크에 데이터를 블록 단위로 순차적으로 쓰는 실험에 대한 결과이다. 그래프에서와 같이 입출력의 크기가 클수록 속도도 향상 되었다. 그 이유는 입출력 이후 CPU에게 인터럽트 요청 수가 입출력의 크기가 작을수록 증가하기 때문이다. 즉, 같은 파일 크기(2G)에 대해 많은 입출력 요청을 수행하게 됨으로 입출력 할 때마다 인터럽트를 처리하기 위해 성능이 저하된다. 실험 결과를 보면 SSD가 HDD에 비해 최소 5.7배에서 14.51배까지 성능 차이를 보였다.

그림 4는 디스크에서 데이터 블록을 순차적으로 읽어오는 실험에 대한 결과이다. 입출력 크기가 커짐에 따라 속도도 향상 되었다. 실험 결과를 보면 SSD가 HDD에 비해 최소 30배 에서 최대 82배의 성능차이를 보였음을 알 수 있다.



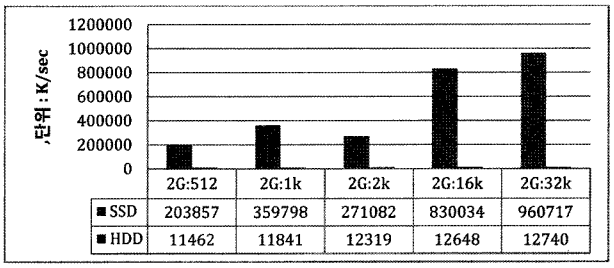
	2G:512	2G:1k	2G:2k	2G:16k	2G:32k
SSD	161887	301225	439114	641345	656296
HDD	30485	29506	31624	32432	33245

그림 3 순차쓰기



	2G:512	2G:1k	2G:2k	2G:16k	2G:32k
SSD	801619	1216166	1726950	2838473	2896980
HDD	27022	29851	30612	31549	29489

그림 4 순차 읽기



	2G:512	2G:1k	2G:2k	2G:16k	2G:32k
SSD	203857	359798	271082	830034	960717
HDD	11462	11841	12319	12648	12740

그림 5 Re-write

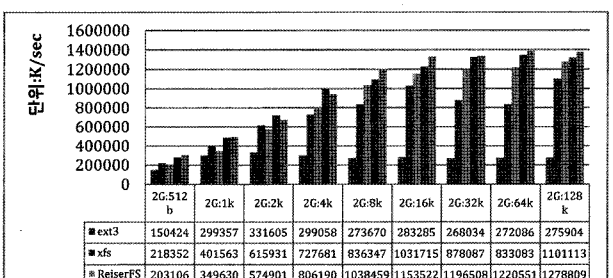
그림 5는 데이터를 읽어와서 블록의 내용을 수정한 후 이를 다음 쓰기를 수행하기 위한 실험결과이다. 이것은 보통 데이터베이스에서 자주 일어나는 트랜잭션이다. 실험 결과를 보면 SSD가 HDD에 비해 최소 17배에서 최대 30배의 성능 차이를 보였다.

또한 모든 실험항목에서 SSD가 HDD보다 성능이 떨어지는 곳이 한 군데도 없다는 사실을 발견할 수 있다. 이를 기반으로 SSD가 HDD보다 성능이 좋으며 Output일 때는 평균 11배, Input일 때는 평균 60배, Re-write일 때는 평균 23배 성능이 우수함을 알 수 있다.

SSD에서 파일시스템 별 성능평가는 ext3, XFS, JFS, ReiserFS, spadFS을 대상으로 하였다. 본 고에서는 캐쉬의 성능을 평가할 수 있는 Re-write에 대해서만 한정하여 기술하도록 한다. 그림 6은 Re-write의 실험결과를 보인 것으로 실험에서는 캐쉬 의존성을 확인할 수 있다. ext3 파일 시스템은 I/O 단위가 2Kb일 때 가장 우수한 성능을 보였으며, 다른 파일 시스템 들은 64Kb(JFS, SpadFS) 및 128Kb(ResierFs, XFS)에서 가장 좋은 성능을 보였다. 64Kb까지는 성능이 지속적으로 향상되었는데 그 이유는 최근 입출력 순으로 캐쉬에 쓴 후 데이터를 읽어올 때 원하는 데이터가 캐쉬에서 읽혀짐으로써 성능을 향상시켜 주기 때문으로 IO 파일의 크기가 클수록 캐싱 효과는 크게 나타난다.

3.2.2 파일 시스템 마크로 벤치마크

마크로 벤치마크는 소형 데이터베이스, 대형 데이



	2G:512b	2G:1k	2G:2k	2G:4k	2G:8k	2G:16k	2G:32k	2G:64k	2G:128k
ext3	150424	299357	331605	299058	273670	283285	268034	272086	275904
xfs	218352	401563	615931	727681	836347	1031715	878087	833083	1101113
ReiserFS	203106	349630	574901	806190	1038459	1153522	1196508	1220551	1278809
SpadFS	281525	487044	718732	996717	1095687	1226662	1325315	1346839	1317699
jfs	302210	498033	670643	944017	1205848	1328684	1333366	1386631	1377997

그림 6 파일 시스템 별 Re-write

터베이스, 다중쓰레드 기반의 웹서버, 메일 서버, 비디오 서버 등과 같은 실제 응용환경에서 발생하는 작업부하의 특징에 따른 성능측정을 목적으로 한다[11]. 벤치마크 프로그램의 목적은 실제로 작업부하를 주었을 때 파일시스템이 어떻게 실행되는지에 대한 평가를 위해 사용된다. 마이크로 벤치마크는 이를 위한 평가하고자 하는 환경에 따라 특별한 파라매터를 설정한 후 수행한다.

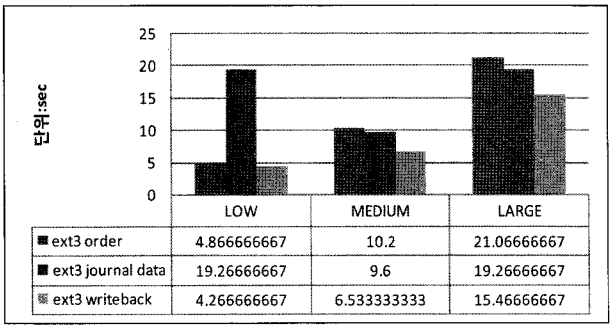
파일 시스템 매크로벤치마크인 Andrew벤치마크는 파일의 생성, 읽기 및 쓰기를 혼합하여 평가를 수행한다. 이를 위해 디렉터리 계층을 구성한 후 파일을 복사하거나 복사본에 대한 파일정보를 얻도록 한다. Andrew벤치마크를 수정하여 작업부하가 동시에 일어나는 복잡한 실행을 평가할 수 있으며, 분산 파일 시스템에서 파일 서버의 확장성을 측정하기 위한 실험에 사용한다.

NFS 서버 실행을 측정하는데 대부분 널리 사용되는 벤치마크로는 SPEC SFS가 사용된다. SPEC SFS 벤치마크는 NFS 서버에 먼저 파일 계층을 생성한 후 다수의 클라이언트 장치들이 서버에게 무작위로 NFS 요구를 수행하도록 한다. SFS의 결과값은 시간당 NFS의 오퍼레이션 값으로 결정한다.

마크로 벤치마크 도구인 PostMark는 작은 크기의 파일들을 생성하고, 읽기, 변경, 삭제 등의 트랜잭션을 수행한다. 파일의 개수와 수행할 트랜잭션 개수는 사용자에 의해 설정할 수 있다.

PostMark 벤치마크 도구를 사용한 성능평가는 ext3, XFS[8], ReiserFS[9,13], SpadFS[10]의 파일시스템을 대상으로 하며, 해당 파일시스템은 커널 모듈의 형태로 구성한 후 실험을 수행한다.

- 성능평가를 위해 작업부하에 대해 위에서 설정한 3가지 종류의 데이터 입출력 버퍼의 크기(512, 4096, 8192) 별로 5회의 수행을 한 후 이들 결과의 평균값을 사용한다.
- 성능평가 파라매터는 형상파일에 저장한 후 수행한다. 성능평가를 위해 논리적으로 SCSI 디스크 형태로 이뮬레이션하여 접근되므로 대상 파일시스템의 생성 및 마운트 전에 단일의 파티션으로 구성하여 실험을 수행한다.
- PostMark를 사용한 성능평가에서는 커널의 부분적인 장애로 인한 부분 결함(partial failure)에 대한 부분을 고려하지 않는다. 단, 저널링을 통한 시스템 회복에 대한 부하를 포함하도록 하며, 성능평가는 메타데이터에 대한 저널링을 기준으로 성능평가를 수행한다.



	LOW	MEDIUM	LARGE
■ ext3 order	4.866666667	10.2	21.06666667
■ ext3 journal data	19.26666667	9.6	19.26666667
▒ ext3 writeback	4.266666667	6.533333333	15.46666667

그림 7 트랜잭션 처리시간

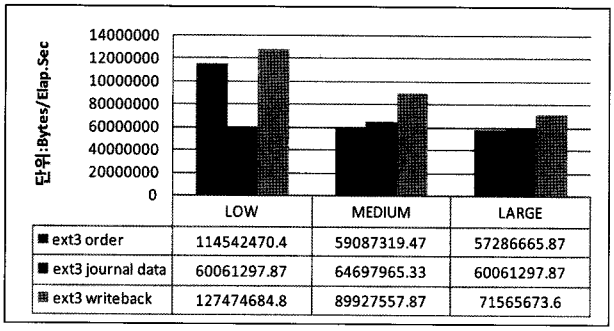
- 전체 시스템에 대한 체크포인트(checkpoint)를 통한 이미지 구성 및 롤백(rollback)에 대한 부분을 고려하지 않는다.

NGS 전용파일시스템 설계 시 전원결함 및 시스템 장애의 복구를 위해 저널링은 필수적인 항목으로서 Ext3를 대상으로 저널형태별 성능 분석을 수행한다. Ext3에서는 저널링 처리를 위해 지연쓰기(writeback), 순서(ordered), 저널데이터(journal data) 3가지 종류의 모드가 제공된다.

- 지연쓰기 모드는 데이터 블록에 대한 특별한 처리를 하지 않으며, 메타데이터에 대한 저널유지도 메모리에 유지한다. 이에 따라 데이터의 일관성을 보증하지 못하는 문제점을 갖는다.
- 순서모드는 커밋 처리를 수행하기 전에 변경이 있는 데이터 블록의 쓰기를 완료한다.
- 저널 데이터 모드는 데이터 블록도 저널링 대상으로 유지한다. 이에 따라 데이터블록도 일단 저널 영역에 쓰게 되고 데이터의 일관성도 보증한다.

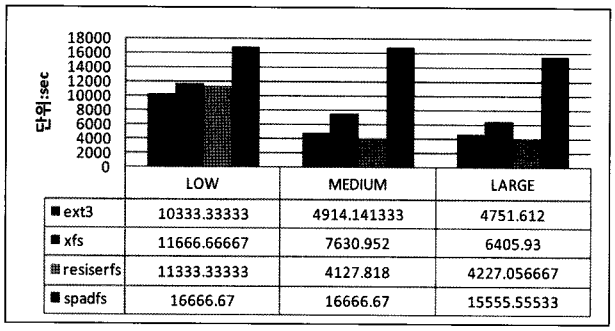
저널링의 종류에 따른 트랜잰션 처리시간은 그림 7과 같다. 전체적으로 지연쓰기 방식을 채용하는 것이 전체 작업부하에서 성능이 우수함을 보였다. 특히 소형 작업부하와 같이 파일의 수에 비해 트랜잭션의 수가 큰 경우에는 지연쓰기를 통한 처리시간의 단축이 가능함을 보였다. 소형 작업 부하의 경우 저널데이터 방식에 비해 지연쓰기는 4.5배의 처리시간의 차이가 있음을 알 수 있다. 그러나 처리시간은 파일의 수가 증가됨에 따라 순서 모드와 저널데이터 모드의 트랜잭션 처리 시간과 차이가 없음을 보인다. 이는 대상 시스템의 L2 캐쉬의 크기에 따른 캐싱 효과에 기인한 것으로 예상할 수 있다.

그림 8은 저널링 종류에 따른 쓰기 대역폭을 보인 것으로 전체적으로 지연쓰기 방식을 채용하는 것이 전체 작업부하에서 성능이 우수함을 보였다. 또한 파일의 수에 비해 트랜잭션의 수가 큰 대형 작업부하에서는 세가지 저널링 모드의 쓰기 대역폭이 유사함을



	LOW	MEDIUM	LARGE
ext3 order	114542470.4	59087319.47	57286665.87
ext3 journal data	60061297.87	64697965.33	60061297.87
ext3 writeback	127474684.8	89927557.87	71565673.6

그림 8 쓰기대역폭



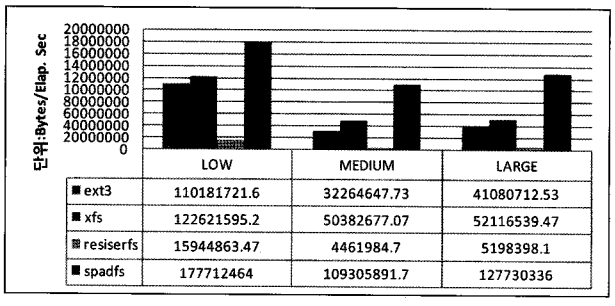
	LOW	MEDIUM	LARGE
ext3	10333.33333	4914.141333	4751.612
xfs	11666.66667	7630.952	6405.93
resiserfs	11333.33333	4127.818	4227.056667
spadfs	16666.67	16666.67	15555.55533

그림 9 초당 트랜잭션 처리

볼 수 있었다. 이는 L2 캐쉬의 크기에 따른 캐싱 효과에 기인한 것으로 예상된다.

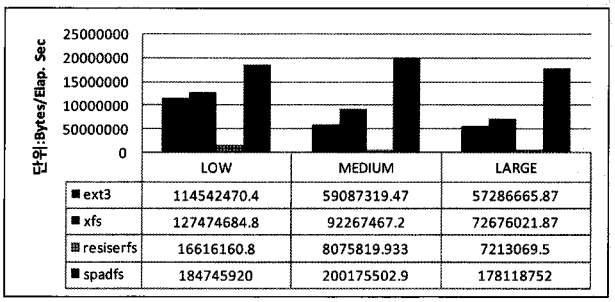
그림 9는 파일시스템 별 초당 트랜잭션 처리수에 대한 성능을 보인 것이다. 전체적으로 spadfs 파일시스템이 다른 파일시스템에 비해 월등히 처리되는 트랜잭션의 수에서 우수함을 보인다. 이는 저널링을 하지 않고 결함횟수(crash count)를 사용하고 있는 것으로 예상할 수 있다. Ext3, XFS, ReiserFS의 비교 결과에서는 XFS가 소형, 중형, 대형 작업부하에 대해 다른 파일시스템에 비해 초당 처리 트랜잭션의 수에서 높을 성능을 보임을 알 수 있다. 소형 작업부하의 경우에서는 ReiserFS이 Ext3 보다 우수함을 보인다.

그림 10은 읽기 대역폭의 실험결과를 보인 것으로 ext3, XFS, reiserFS의 비교 결과에서 XFS가 소형, 중형, 대형 작업부하에 대해 다른 파일시스템에 비해 초당 처리 트랜잭션의 수에서 높을 성능을 보임을 알 수



	LOW	MEDIUM	LARGE
ext3	110181721.6	32264647.73	41080712.53
xfs	122621595.2	50382677.07	52116539.47
resiserfs	15944863.47	4461984.7	5198398.1
spadfs	177712464	109305891.7	127730336

그림 10 읽기 대역폭



	LOW	MEDIUM	LARGE
ext3	114542470.4	59087319.47	57286665.87
xfs	127474684.8	92267467.2	72676021.87
resiserfs	16616160.8	8075819.933	7213069.5
spadfs	184745920	200175502.9	178118752

그림 11 쓰기 대역폭

있다. 소형작업부하에서는 ext3에 비해 XFS가 1.1배 우수하고, ReiserFS에 비해서는 7.7배 우수함을 보인다. 그림 11은 쓰기 대역폭을 보인 것으로 읽기 대역폭과 동일한 성능을 보인다.

3.3 성능평가 분석

파일시스템은 운영체제의 주요한 구성요소이며 파일시스템의 구현 시 품질은 응용형태 및 운영환경에 영향을 준다. 실험을 통해 파일 시스템의 설계 시에는 성능과 안전성을 고려하여 하며 NGS전용파일 시스템 설계를 위해서는 저널링에 따른 성능 파라메타에 대한 영향평가 및 개선사항에 대한 도출이 요구됨을 알 수 있다. 저널링에 따른 파일시스템 성능 오버헤드를 분석한 결과 쓰기지연을 통한 파일시스템에 저장매체 접근 횟수를 줄이는 것이 성능개선에 활용할 수 있다.

파일시스템 별 성능분석을 통해서는 저널링의 오버헤드를 spadfs과 비교함으로써 전체 파일 처리에 있어서 많은 부분을 차지하고 있음을 알 수 있었다. 그러나 spadfs 시스템의 경우 대용량의 데이터 처리에 대한 신뢰성 보장을 위한 추가실험이 필요하다. 저널링의 효율성을 높일 수 있는 경량화가 필요하다. 경량화를 위한 지연쓰기(delayed write)는 저널영역을 별도의 파일시스템으로 유지하여 다중 갱신과 결합시켜 메타데이터 갱신 성능을 상당히 얻을 수 있다. 작은 크기에 대한 디스크접근은 백그라운드 디스크 쓰기 등의 디스크 스케줄링 조정을 하여 병렬성을 높일 수 있도록 개선할 수 있다.

XFS 파일 시스템은 다른 파일시스템에 비해 병렬 I/O를 위한 구성요소를 포함하고 있다. XFS 파일시스템은 내부적으로 할당그룹(allocation group)으로 파티션이 구분된다. 개별적인 할당 그룹은 파일과 디렉터리를 유지함으로써 병렬성을 높일 수 있다. 전용 파일시스템 설계 시에는 단일 파일시스템에 대한 물리적인 분할을 통해 입출력의 성능 개선을 이룰 수 있다.

앞서의 실험에서는 트랜잭션 수의 증가 시 동일한 파일에 대한 캐쉬적중율(cache hit ratio)이 높아짐에

따라 성능이 개선됨을 볼 수 있었다. 특히 NGS의 설계 시 대용량 L2 캐쉬는 저장매체에 대한 접근 횟수를 줄일 수 있으므로 성능개선 효과가 있다.

4. 결 론

본 고에서는 고속의 입출력을 지원하는 메모리배열을 저장매체로 사용하는 선행 NGS 시스템의 성능평가를 기술하였으며, 이를 기반으로 NGS 전용 파일시스템 개발 시의 고려사항을 제안하였다. 이를 위해 마크로벤치마크 프로그램인 Bonnie++와 매크로벤치마크PostMark를 사용하여 실험을 수행하였다. 실험결과 분석을 통해 시스템 장애시 발생되는 결함 처리를 처리하는 기법에 따른 성능영향을 평가하였으며, 저널링 기법에 대한 효율성을 보장하는 것이 필요함을 알 수 있었다.

기존의 파일시스템의 개선 연구는 디스크 기반의 매체를 기반으로 하고 있으므로 NGS의 파일 시스템을 설계하기 위해서는 고려하기 어려운 점이 있다. 이를 위해 주요 파일시스템인 Ext3, XFS, ReiserFS, JFS에 대한 성능 및 안정성 관련 평가를 수행하였다. 또한 이들 파일시스템의 주요 자료구조와 알고리즘을 분석한 후 NGS 전용 파일시스템 설계를 위한 개선사항을 도출하였다.

CPU, 메모리 및 L2 캐쉬의 고성능 및 대용량화에 따라 파일시스템의 I/O 성능 보장을 위해서는 커널 내의 효율적인 자료구조 및 알고리즘을 통한 최적화가 필요하다. NGS 전용파일시스템 설계는 응용, 커널 및 디바이스 드라이버(device driver)에 대한 IO 성능 평가가 필수적이다.

향후 NGS 전용파일시스템은 네트워크 접속 저장 장치(Network-attached storage devices)로 활용하기 위해 여러 네트워크 노드 간의 데이터 및 처리기능의 분산을 위한 방대한 확장가능 파일 시스템을 구성할 수 있는 방법을 제공할 수 있도록 구성되어야 한다. 특히, 인터넷 기반의 멀티미디어 스트리밍 서비스를 위한 IPTV, 데이터웨어하우스 등의 기업 내의 저장 시스템으로 활용될 수 있도록 하기 위해서는 응용에 대한 데이터 접근패턴에 대해 추가적으로 고려되어야 한다.

참고문헌

[1] Solid Data systems, "Impact of Solid-state disk on high-transaction rate databases", Solid data systems, Inc. White paper, 2005 Feb

[2] TMS, "Increase Application Performance with Solid State Disks", TMS white paper, 2008.Feb

[3] Vnodes: An Architecture for Multiple File System Types in Sun UNIX

[4] Nava Aizikowitz, Alex Glikson, COMPONENT-BASED PERFORMANCE MODELING OF A STORAGE AREA NETWORK, Proceedings of the 2005 Winter Simulation Conference

[5] http://tldp.org/LDP/tlk/fs/filesystem.html

[6] Dominic Giampaolo, Practical File System Design with the Be File System, Morgan Kaufmann Publishers, 1999.

[7] Elizabeth Shriver, Christopher Small and Keith A. Smith, Why does file system prefetching work?, Proceedings of the USENIX Annual Technical Conference Monterey, California, USA, June 6-11, 1999.

[8] A. Sweeney - Scalability in the XFS File System Proceedings of the Usenix 1996 Technical Conference, pp. 1-14.

[9] Namesys, ReiserFS

[10] Patočka,, Spadfs, http://artax.karlin.mff.cuni.cz/~mikulas/spadfs/

[11] J. Katcher, "PostMark: A New File System Benchmark," Technical Report 3022, Network Appliance, 1997.

[12] McKusick & Ganger, 1999. M. McKusick & G. Ganger, "Soft Updates: A Technique for Eliminating Most Synchronous Writes in the Fast Filesystem," Proceedings of the Freenix Track at the 1999 Usenix Annual Technical Conference, p. 1-17 (June 1999).

[13] Reiser, 2001. H. Reiser, "The Reiser File System," http://www.namesys.com/res_whol.shtml (January 2001).

[14] Seltzer et al, 2000. M. Seltzer, G. Ganger, M. McKusick, K. Smith, C. Soules, & C. Stein, "Journaling versus Soft Updates: Asynchronous Meta-data Protection in File Systems," Proceedings of the San Diego Usenix Conference, p. 71-84 (June 2000).

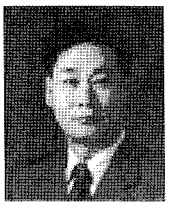
[15] Best & Kleikamp, 2003. S. Best & D. Kleikamp, "How the Journaled File System handles the on-disk layout," http://www-106.ibm.com/developerworks/linux/ library/l-jfslayout/ (2003).

[16] D. Capps and W.D. Norcott, "Iozone Filesystem Benchmark," http:/www.iozone.org/, 2004.

[17] R. Coker, "Bonnie++ Benchmark Tool," http://www.coker.com. au/bonnie++/, 2004.

강 윤 희

1989 동국대학교 컴퓨터공학과 (학사)
1991 동국대학교 컴퓨터공학과 (석사)
2002 고려대학교 컴퓨터과학과(박사)
1991～1994 한국전자통신연구원(연구원)
1994～1997 한국문화예술진흥원(선임연구원)
2000～현재 백석대학교 정보통신학부 조교수
관심분야 : 그리드 컴퓨팅, 분산시스템, 결함포용
E-mail : yhkang@bu.ac.kr



정 승 국

2004 한남대학교 전자정보통신 공학과(박사)
1985～현재 한국전자통신연구원(책임연구원)
관심분야 : Grid Computing, Utility Computing, Solid State Disk, Storage&Server Virtualization
E-mail : skjeong@etri.re.kr

강원지부 총회 및 제3회 학술대회

- ❑ 일　　자 : 2009년 6월 12일
- ❑ 장　　소 : 연세대학교 원주캠퍼스
- ❑ 주　　관 : 강원지부
- ❑ 주　　최 : 한국정보과학회
- ❑ 문　　의 : 연세대학교 윤상균 교수 033-760-2267