KIPS Trans. Comp. and Comm. Sys.
Vol.7, No.11 pp.275~280 pISSN: 2287-5891

https://doi.org/10.3745/KTCCS.2018.7.11.275

Performance Enhancement and Evaluation of Distributed File System for Cloud

Jong Hyuk Lee†

ABSTRACT

The choice of a suitable distributed file system is required for loading large data and high-speed processing through subsequent applications in a cloud environment. In this paper, we propose a write performance improvement method based on GlusterFS and evaluate the performance of MapRFS, CephFS and GlusterFS among existing distributed file systems in cloud environment. The write performance improvement method proposed in this paper enhances the response time by changing the synchronization level used by the synchronous replication method from disk to memory. Experimental results show that the distributed file system to which the proposed method is applied is superior to other distributed file systems in the case of sequential write, random write and random read.

Keywords : Distributed File System, MapRFS, CephFS, GlusterFS, Write Performance, Big Data, Cloud Computing

클라우드 분산 파일 시스템 성능 개선 및 평가

이 종 혁†

요 약

클라우드 환경에서 빅데이터 적재와 이후 애플리케이션을 통한 고속 처리를 위해서는 적합한 분산 파일 시스템의 선택이 요구된다. 본 논문에서는 GlusterFS 기반 쓰기 성능 향상 방법을 제안하고 클라우드 환경에서 기존 분산 파일 시스템 중 MapRFS, CephFS, GlusterFS와 성능을 비교 평가한다. 본 논문에서 제안한 쓰기 성능 향상 방법은 동기식 스토리지 복제 방식에서 사용하는 동기화 수준을 디스크에서 메모리로 변경함으로써 응답 시간을 향상 시킨다. 실험 결과는 본 논문의 제안 방법이 적용된 분산 파일 시스템이 순차 쓰기의 경우와 랜덤 쓰기와 랜덤 읽기가 혼합된 경우에서 다른 분산 파일 시스템 대비 성능이 우수함을 보인다.

키워드 : 분산 파일 시스템, MapRFS, CephFS, GlusterFS, 쓰기 성능, 빅데이터, 클라우드 컴퓨팅

1. 서 론

모바일 인터넷 시대를 기점으로 전 세계적으로 디지털 환경에서 생성 및 유통되는 문자, 영상 등의 데이터양이 폭발적으로 증가함에 따라 방대한 규모와 정형화 되지 않은 데이터를 포함하는 빅데이터 시대가 도래했다. 빅데이터는 대량(High-Volume), 고속(High-Velocity), 그리고(또는) 다양성(High-Variety)의 속성을 지닌 정보 자산으로 이를 이용한 통찰 강화, 의사 결정 및 프로세스 자동화를 위해서는 비용 효율적이고 혁신적인 정보처리가 요구된다[1]. 빅데이터 정보처리는 일반적으로 수집, 적재, 처리, 탐색, 분석 및 응용의 단계로 구성된다. 이 중 적재 단계는 수집한 데이터를 분산 스토리지에 영구 또는 임시로 저장하는 단계로 수집 데이터 형태에 따라 분산 스토리지 유형이 파일 시스템, NoSQL, In-Memory Cache, Message Oriented Middleware 등과 같이 달라진다. 그리고 그 다음 단계인 처리 단계 방법도 달라지므로 적절한 분산 스토리지 선택이 중요하다.

대규모 화학 및 생물학적 문제를 풀기 위한 과학 애플리케이션뿐만 아니라 텍스트, 음성, 영상 인식 애플리케이션 등과 같은 기존 레거시 애플리케이션은 일반적으로 POSIX (Portable Operating System Interface) 방식으로 파일 시스템을 접근한다. 그래서 분산 스토리지는 일반적으로 NFS (Network File System)와 같은 표준화된 I/O 인터페이스를 제공한다. 이는 고도 병렬화된 대부분의 애플리케이션에서 노드 간 파일 공유를 가능하게 하여 한 노드에서 처리한 파일을 다른 노드에서 접근하도록 하는 공유 파일 시스템의 필요성을 의미한다. 대규모 데이터를 저장하기 위한 공유 파일 시스템은 두 가지 방법으로 구축된다. 첫 번째 방법은 대규모 데이터를 저장하기에 충분히 큰 자원이 있는 대형 시스템을 사용하는 것으로

※ 이 결과물은 2017년도 대구가톨릭대학교 학술연구비 지원에 의한 것임.
† 정 회 원 : 대구가톨릭대학교 빅데이터공학과 조교수
Manuscript Received : August 29, 2018
Accepted : September 27, 2018
* Corresponding Author : Jong Hyuk Lee(jonghyuk@cu.ac.kr)

수직 확장 모델이다. 두 번째 방법은 다수의 노드를 사용하여 각 노드가 데이터의 일부분을 저장하는 수평 확장 모델이다. 수직 확장 모델은 대규모 데이터를 저장하기에는 물리적 한계에 도달하여 분산 파일 시스템과 같은 수평 확장 모델이 스토리지 솔루션으로 주목 받고 있다. 그래서 최근 클라우드 컴퓨팅은 빅데이터를 분산 파일 시스템에 저장하여 다수의 클라이언트가 데이터에 접근 및 조작(생성, 수정, 삭제, 읽기, 쓰기)하도록 지원하고 있다. 분산 파일 시스템은 각 데이터 파일이 일정한 크기 단위(즉, 청크)로 여러 노드에 분할 저장될 수 있어 병렬 실행을 요구하는 응용에 적합하다.

이와 같이 빅데이터 적재와 이후 애플리케이션을 통한 고속 처리에 적합한 분산 파일 시스템의 선택을 위해서는 그에 관한 조사와 성능 평가가 요구된다. 본 논문은 POSIX 방식으로 접근 가능한 분산 파일 시스템 중 MapRFS, CephFS, GlusterFS의 성능을 평가한다. 그리고 기존 GlusterFS에 기반을 둔 쓰기 성능 향상 방법을 제안하고 기존 성능과 비교 평가한다. 본 논문은 분산 파일 시스템 성능 평가 면에서 기존 연구와 달리 다음과 같은 차별성이 있다.

- 실제 퍼블릭 클라우드 인프라 환경인 아마존 EC2에서 테스트베드를 구축 후 성능 평가함
- 최근 저장 매체인 일반 SSD와 NVMe에 대해 성능 평가함
- 순차적 쓰기 및 읽기에 대한 성능뿐만 아니라 랜덤 쓰기 및 읽기에 해당하는 MySQL OLTP에 대해 성능 평가함
- GlusterFS의 NFS를 통한 쓰기 성능 향상 방법을 제안하고 평가함

본 논문의 구성은 다음과 같다. 2절에서는 분산 파일 시스템에 대한 관련 연구를 살펴본다. 3절에서는 기존 MapRFS, CephFS, GlusterFS의 아키텍처와 쓰기 성능 향상 방법을 살펴본다. 4절에서는 실험 환경과 결과를 나타낸다. 마지막으로 5절에서 결론을 맺는다.

2. 관련 연구

화학 및 생물학적 문제를 풀기 위한 다양한 이론이 등장하면서 이를 계산하는 고성능 컴퓨팅 인프라 및 관련 미들웨어 기술 설계에 관한 연구가 진행되었다. Roch et al.[2]은 전산 화학 프로그램을 위한 소프트웨어인 GAMESS(General Atomic and Molecular Electronic Structure System)에 적합한 고효율 스토리지 솔루션을 알아보고자 GlusterFS와 CephFS와 같은 분산 파일 시스템 솔루션의 성능을 평가하였다. 이 연구는 GAMESS의 워크로드의 I/O 패턴을 분석하여 실험에서 사용할 워크로드 테스트 케이스를 만들었고 평가 척도로 읽기/쓰기 속도 및 비율, 확장성, 블록 크기에 따른 영향 정도를 사용하였다. 실험 결과는 GlusterFS가 일반적으로 Ceph 및 로컬 디스크 파일 시스템 보다 더 좋은 성능을 보여주어 로컬 저장소에 대한 가장 합리적인 대안이라고 판단하였다. Donivito et al.[3]은 고에너지 물리학(High-Energy Physics, HEP) 데이터 분석을 위해 HDFS, Ceph, GlusterFS와 같은 분산 파일 시스템의 성능을 평가하였다. Gudu et al.[4]은 Ceph가 이론적인 성능과 실제 성능의 차이가 없으며 스토리지 노드와 클라이언트의 수의 변경에 따라 확장성이 보장됨을 실험을 통해 증명하였다. 그리고 Ceph의 저널을 메모리와 같이 좀 더 빠른 매체를 사용하게 함으로써 쓰기 처리량이 크게 증가함을 밝혀냈다.

빅데이터 기술의 출현과 함께 빅데이터 시스템의 성능을 평가하기 위해 다양한 벤치마크가 개발되고 있다. FIO (Flexible I/O Tester)[5]는 파일 I/O 성능을 측정하는 벤치마킹 유틸리티이며 SysBench[6]는 파일 I/O 성능뿐만 아니라 DBMS의 성능을 측정하는 벤치마킹 유틸리티이다. Yahoo! Cloud Serving Benchmark (YCSB)[7]는 Cassandra, HBase, MySQL 등과 같은 트랜잭션 처리 시스템 성능을 측정하기 위해 개발되었다. YCSB는 DB 테이블에 대한 insert, read, update, scan 등으로 조합된 워크로드를 생성한다. 이 외에도 CloudSuite[8], BigBench [9] 등과 같이 클라우드 서비스 및 빅데이터 분석 성능을 측정하는 다양한 벤치마크가 있다. 클라우드 환경에서 I/O 워크로드 특성은 매우 이질적이고 복잡하기 때문에 클라우드 파일 시스템의 성능을 정확하고 효율적으로 평가하기는 어렵다. Ren et al.[10]은 이를 극복하기 위해 AliCloud 클라우드 파일 시스템인 Pangu의 I/O 워크로드 트레이스를 수집 및 분석하여 요청 도달 패턴, 요청 크기, 데이터 수 등과 같은 여러 가지 속성으로 특징화하고 이를 반영한 Porcupine 벤치마킹 제품군을 개발하였다. 이 제품군은 과거 트레이스를 재생하여 워크로드를 생성할 뿐만 아니라 사용자 정의 구성에 따라 파일 I/O 요청 스트림을 합성하는 기능을 포함한다. 분산 파일 시스템은 클라우드 규모의 데이터 처리 미들웨어의 핵심 구성 요소이다. 이에 따라 벤치마크 프로그램 또는 어플리케이션을 실행하여 분산 파일 시스템의 성능을 측정하는 연구가 주로 진행되었다. 그러나 Wu et al.[11]은 모델 중심 성능 분석 프레임워크를 제안하였다. 이 프레임워크를 통해 아키텍처 설계, 구성, 배치 과정에서도 성능 결과에 대해 빠른 피드백이 가능하여 결과적으로 아키텍처 재설계 및 재배치 비용을 줄일 수 있었다.

최근 클라우드 파일 시스템이 널리 이용되고 있는 가운데 랜덤 쓰기 연산이 시스템 성능에 많은 영향을 주고 있다는 조사 결과가 있다[12, 13]. Gong et al.[14]은 고정 크기의 객체(또는 청크)를 기반으로 기존 분산 파일 시스템과 달리 Ceph를 기반으로 가변 크기의 객체를 지원하는 분산 파일 시스템(VarFS)을 개발하였다. VarFS는 가변 객체 인덱싱을 통합하고 랜덤 쓰기 인터페이스를 지원하며 POSIX 호환을 유지한다. 특히, VarFS는 불필요한 읽기 및 쓰기 데이터의 양을 줄여 결과적으로 전체 전송 데이터양을 줄이도록 설계되었다. 그 결과, 랜덤 쓰기 시 지연 시간이 기존 Ceph에 비해 약 1~2 배 짧았다.

3. 분산 파일 시스템 아키텍처 및 쓰기 성능 향상

3.1 분산 파일 시스템 아키텍처

대부분의 분산 파일 시스템은 클라이언트 서버 아키텍처 기반으로 구축되었지만 최근에는 클러스터 기반 아키텍처와

같은 분산된 형태 또한 존재한다.

클라이언트 서버 아키텍처의 대표적인 예는 Network File System (NFS)이다. NFS는 네트워크상의 여러 노드 간 파일이 공유 되어 원격에 있는 파일이 마치 로컬에 있는 파일처럼 접근되도록 표준화된 인터페이스를 제공한다. 이는 NFS가 로컬의 가상 파일 시스템과 호환되는 파일 시스템 중 하나임을 의미한다. 그러나 NFS는 단일 서버 환경에서는 단일 장애점(single point of failure)이 있어 가용성(availability) 측면에서 한계가 있다. 이를 극복하기 위해 최근의 분산 파일 시스템은 다중 서버 환경에서 한 서버가 중단되더라도 정상 동작하는 다른 서버로 제어권이 자동적으로 넘겨지는 장애극복(failover) 기법을 사용한다.

클러스터 기반 아키텍처는 파일 스트라이핑(file-striping) 기술을 사용하여 클라이언트 서버 아키텍처의 한계를 극복하고 병렬 처리를 요구하는 응용의 성능을 향상시킨다. 파일 스트라이핑은 파일을 여러 청크로 분리하고 이를 여러 서버에 분할하여 저장하는 것을 의미한다. 이를 통해 파일의 다른 부분을 병렬적으로 접근 가능하도록 하여 병렬성을 높인다. 하지만 많은 수의 하드웨어를 사용할수록 더 많은 장애가 발생할 수 있다. 이를 위해 같은 청크가 여러 서버에 저장되도록 하는 데이터 복제 기법을 사용하여 데이터 접근에 대한 단일 장애점 문제를 해결한다. 클러스터 기반 아키텍처를 이용한 대표적인 분산 파일 시스템 예로 Google File System (GFS) [15], Hadoop File System (HDFS)[16], MapR File System (MapRFS)[17], Ceph File System(CephFS)[18], Gluster File System(GlusterFS) [19] 등이 있다. 본 논문은 이 중 성능 집약적 응용을 위해 설계된 MapRFS, CephFS, GlusterFS의 아키텍처를 살펴보고 GlusterFS의 쓰기 성능 개선 방법을 제안한다.

1) MapRFS

MapRFS는 가용성, 확장성, 성능, 안정성 측면에서 최적화된 분산 파일 시스템이다. MapRFS는 HDFS와 달리 데이터 위치와 복제본(replica)에 대한 정보인 클러스터 메타데이터를 네임노드 없이 관리하는 분산 메타데이터 아키텍처를 이용한다. 이 아키텍처를 통해 메타데이터 접근에 대한 고가용성(High Availability, HA)을 제공하고 클러스터에 저장할 수 있는 파일의 수를 거의 제한하지 않음으로써 파일 개수 및 용량에 대한 확장성을 제공한다. 그리고 MapRFS는 클러스터 기반 아키텍처 기반의 다른 분산 파일 시스템처럼 데이터를 자동 분할하고 복제한다. 특히 MapRFS는 청크 사이즈 단위로 데이터 접근을 하는 HDFS와 달리 청크 사이즈와 더불어 조금 더 작은 단위인 블록 사이즈 단위로 데이터 접근을 하게 함으로써 랜덤 접근에 유리하도록 설계되었다. MapRFS는 데이터 저장소 기능 측면에서 HDFS API와 호환될 뿐만 아니라 NFS 인터페이스를 통해 Network Attached Storage (NAS) 방식의 접근이 가능하다.

2) CephFS

CephFS는 성능과 신뢰성 측면에서 최적화된 분산 파일 시스템이다. Ceph는 단일 분산 컴퓨터 클러스터에 오브젝트 스토리지를 구현한 스토리지 플랫폼이다. 이 오브젝트 스토리지를 통해 오브젝트 레벨(즉, Reliable Autonomic Distributed Object Storage, RADOS), 블록 레벨(즉, RADOS Block Device), 파일 레벨(즉, CephFS)의 스토리지 인터페이스를 제공한다. Ceph 메타데이터 서버 클러스터는 파일 시스템의 파일명과 디렉토리를 RADOS 클러스터 내 저장된 오브젝트로 매핑하는 서비스이다. 이 메타데이터 서버 클러스터는 확장 또는 축소가 가능하며 파일 시스템을 동적으로 리밸런스하여 데이터가 클러스터 노드들 간에 균등하게 분산되도록 한다. 이를 통해 고성능을 보장하고 특정 노드에 과중한 로드가 발생하지 않도록 한다. Ceph도 다른 분산 파일 시스템과 같이 더 높은 처리량을 달성하기 위해 여러 노드에 개별 파일을 스트라이핑한다.

3) GlusterFS

GlusterFS는 클라우드 컴퓨팅, 스트리밍 미디어 서비스, CDN(Content Delivery Networks) 등과 같은 응용에서 NAS 방식의 접근이 가능한 분산 파일 시스템이다. GlusterFS는 스토리지의 기본 단위로서 브릭(brick)과 볼륨(volume)을 사용한다. 브릭은 각각의 서버에서 분산 파일 시스템 용도로 익스포트하려는 디렉토리를 의미하며 볼륨은 이 브릭들의 집합이다. GlusterFS는 사용자의 요구사항(스토리지 크기 확장성, 신뢰성, 성능 등)에 따라 여러 가지 볼륨 타입을 지원한다. Distributed Volume의 경우 한 파일이 같은 브릭에 저장되지 않아 볼륨 크기를 쉽게 확장할 수 있지만 브릭에 장애가 발생하면 데이터 손실이 발생한다. Replicated Volume의 경우 한 파일은 복수 개의 브릭에 저장되어 한쪽 브릭에 장애가 발생하더라도 다른 복제된 브릭을 통해 데이터 접근이 가능하여 Distributed Volume에 비해 좀 더 향상된 신뢰성과 데이터 중복성(redundancy)을 제공한다. Distributed Replicated Volume의 경우 한 파일이 서로 다른 서버의 브릭에 복제되어 Distributed Volume과 Replicated Volume의 장점을 모두 갖는다. 그런데 한 브릭에 저장된 매우 큰 파일이 많은 클라이언트로부터 동시에 접근된다면 성능이 감소될 소지가 있다. 그래서 Striped Volume의 경우 데이터를 일정한 단위(즉, 청크)로 분할하여 여러 브릭에 저장시켜 데이터 접근에 대한 로드를 분산시킨다. 하지만 이 볼륨 타입은 데이터 중복성을 제공하지 않는다. Distributed Striped Volume의 경우 여러 개의 Striped Volume을 사용하여 각각의 파일을 분산되게 저장함으로써 볼륨 크기를 쉽게 확장할 수 있도록 한다.

3.2 eGlusterFS: GlusterFS 기반 쓰기 성능 향상 방법 적용

스토리지 복제 방식은 동기식과 비동기식으로 나뉜다. 동기식 복제는 클라이언트로부터 쓰기 I/O 요청을 받을 경우 데이터가 복제되어, 참여하는 모든 서버에서 복제 데이터가 쓰기 완료될 때까지 I/O 요청을 완료하지 않는 것을 의미한다. 따라서 네트워크 지연이 증가할수록 응답 시간이 증가하여 성능 감소의 원인이 되지만 특정 서버에서 장애가 발생하더라도 손실 없이 데이터를 복구할 수 있는 특징이 있다. 반

면 비동기식 복제는 클라이언트로부터 쓰기 I/O 요청을 받을 경우 일단 마스터 서버에 쓰기 완료를 한 후 클라이언트에 바로 응답하고, 복제 데이터는 일정 시간 후 또는 일정 간격으로 슬레이브 서버에 쓰게 하여 실제 완료를 미루는 것을 의미한다. 따라서 응답 시간은 클라이언트와 마스터 서버 간 네트워크 지연 및 마스터 서버의 쓰기 I/O 성능에 대부분 영향을 받아 동기식 복제에 비해 짧다. 하지만 복제 중 마스터 서버에 장애가 발생하면 데이터가 손실될 수 있다. GlusterFS는 동기식을 Automatic File Replication (AFR)이라는 명명으로 지원하며 비동기식을 Geo-replication이라는 명명으로 지원한다. 본 논문은 GlusterFS 기반 AFR 사용 시 쓰기 성능 향상 방법이 적용된 eGlusterFS를 제안한다. 파일 수정 작업 시 AFR은 잠금 설정, xattr 설정, 쓰기, xattr 해제, 잠금 해제의 단계를 거치는데 본 논문은 Fig. 1과 같이 쓰기 시 동기화 정도를 기존 디스크에서 메모리로 변경한다. 즉, Fig. 1에서 File2가 3개의 서버의 디스크에 쓰기 완료된 후 응답하는 것이 아니라 File1이 메모리에 쓰기 완료된 후 응답하는 것처럼 변경한다. 본 논문의 메모리에 쓰는 방식은 기존 동기식에서 사용한 디스크에 쓰는 방식 보다 훨씬 빠른 프로세스이기 때문에 성능 향상의 방법이 된다. 그리고 모든 복제 참여 서버에서 동시에 장애가 발생하지 않는 이상 데이터 손실 가능성은 없다. 하지만 본 논문의 방법이 특정 조건에서 데이터 손실이 발생할 수 있으므로 데이터 중요도에 따라 위 세 가지 방법 중 선택적 사용이 필요하다. eGlusterFS에 대한 구현 결과는 [20]에 공개되어 있다.

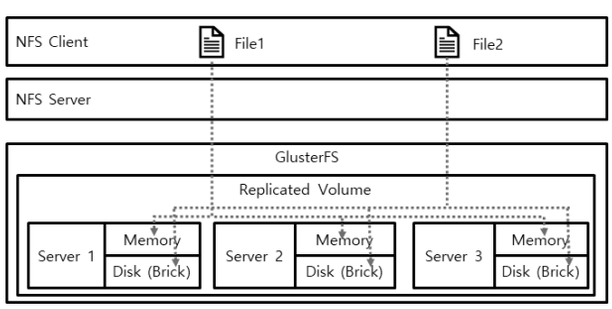

Fig. 1. File Synchronization in eGlusterFS

4. 실 험

4.1 실험 환경

본 논문은 네 가지 분산 파일 시스템의 성능을 실험하기 위하여 Table 1과 같이 퍼블릭 클라우드 중 하나인 아마존 EC2를 이용하여 테스트베드를 구축하였다. 저장 매체(일반 SSD와 NVMe SSD)의 차이로 인한 각 분산 파일 시스템의 성능 차이를 살펴보기 위해 저장 매체에 적합한 인스턴스 타입을 다르게 선택하였으나 CPU 개수, 메모리 크기, 네트워크 대역폭은 모두 동일하게 하였다. 각 분산 파일 시스템 설정은 기본 설정을 사용하되 복제 개수는 3개로 모두 동일하게 하였고 클라이언트 인스턴스 타입 또한 모두 동일하게 사용하였다.

Table 1. Experimental Environments

	General SSD	NVMe SSD
Server instance and storage types	r3.8xlarge (32 vCPU, 244GiB Memory, 2 x 320 SSD, 10Gbit Network), io1 (Provisioned IOPS SSD, 100GiB, 10,000 IOPS) x 3	i3.8xlarge (32 vCPU, 244GiB Memory, 4 x 1,900 NVMe SSD, 10Gbit Network)
No. of servers	3	
Configuration	Use default settings (Replica: 3) Use three identical type EBS volumes as data volumes	Use default settings (Replica: 3) Use three local NVMe SSDs as data volumes
Client instance type	m4.10xlarge (40 vCPUs, 160GiB Memory, 10Gbit Network), gp2 (General Purpose SSD, 100GiB) x 1	
Operating system	CentOS 7.4	

4.2 벤치마크

본 논문은 파일 I/O와 DBMS 관점에서 네 가지 분산 파일 시스템의 성능을 측정하기 위해 FIO와 SysBench를 사용하였다. 본 논문에서 분산 파일 시스템 내 파일 순차 쓰기 및 순차 읽기 시 사용한 FIO 설정은 다음과 같다.

```
--size=6144M --numjobs=32 --rw={write|read}
--bs=1024k --direct=1 --sync=1 --filesize=1M
```

이 설정은 6,144개의 1MB 크기의 파일(총용량 6GB)을 대상으로 32개의 클라이언트가 1,024KB 크기의 블록과 Direct I/O, Sync 옵션을 사용하여 병렬적으로 읽거나 쓰는 것을 의미한다.

본 논문에서 분산 파일 시스템 내 DB 관련 파일을 저장 후 쿼리 시 사용한 SysBench 설정은 다음과 같다.

```
--db-driver=mysql --oltp-table-size=10000000
--oltp-tables-count=32 --threads=250
```

이 설정은 MySQL DBMS에 레코드 개수가 천만 개인 32개의 테스트 테이블을 대상으로 250개의 클라이언트가 다양한 쿼리를 병렬적으로 실행하는 것을 의미한다.

4.3 실험 결과

Fig. 2는 FIO 벤치마크 프로그램으로 측정한 일반 SSD 테스트 환경에서의 순차 쓰기 성능 결과 그래프이다. 네 가지 분산 파일 시스템 중 본 논문에서 제안한 NFS 쓰기 성능 향상 방법이 적용된 eGlusterFS가 가장 높은 성능을 보이며 GlusterFS 대비 약 11%의 성능 향상이 있었다.

Fig. 3은 FIO 벤치마크 프로그램으로 측정한 NVMe SSD 테스트 환경에서의 순차 쓰기 성능 결과 그래프이다. 마찬가지로 eGlusterFS가 가장 높은 성능을 보이며 GlusterFS 대비 약 7%의 성능 향상이 있었다. GlusterFS 대비 eGlusterFS의 순차적 쓰기 성능 정도는 일반 SSD가 NVMe SSD 테스트 환경보다 약 4% 정도 높은데 이는 본 논문에서 제안하는 NFS

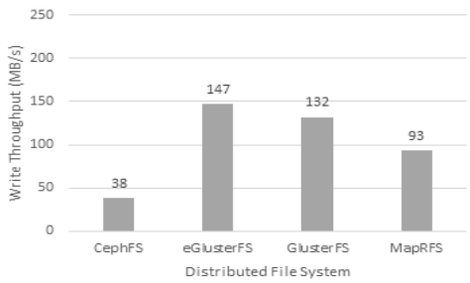

Fig. 2. Sequential Write Throughput (MB/s) in a General SSD Test Environment

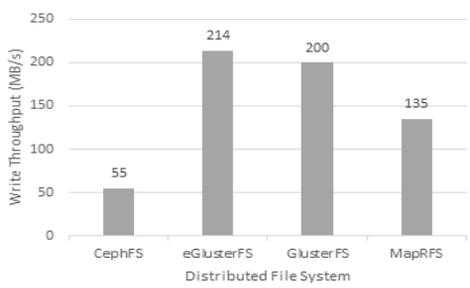

Fig. 3. Sequential Write Throughput (MB/s) in a NVMe SSD Test Environment

쓰기 성능 향상 방법이 쓰기 성능이 좋지 않은 저장 매체에서 좀 더 효과적으로 작용한다는 것을 의미한다.

Fig. 4는 FIO 벤치마크 프로그램으로 측정한 일반 SSD 테스트 환경에서의 순차 읽기 성능 결과 그래프이다. 네 가지 분산 파일 시스템 중 CephFS가 가장 높은 성능을 보인다.

Fig. 5는 FIO 벤치마크 프로그램으로 측정한 NVMe SSD 테스트 환경에서의 순차 읽기 성능 결과 그래프이다. 마찬가지로 CephFS가 가장 높은 성능을 보인다. 하지만 별도의 실험에서 CephFS의 순차 읽기 성능이 로컬 디스크의 순차 읽기 성능과 거의 같은 것으로 보아 클라이언트 CephFS에서 캐시 히트가 발생한 것으로 보인다. 즉, Direct I/O 옵션 지정에 따라 파일 읽기 시 클라이언트 운영체제의 읽기 캐시를 우회하여 서버의 CephFS 내 데이터를 읽어야 하지만 클라이언트 CephFS의 캐시를 이용하는 것으로 보인다. Direct I/O 옵션이 정상적으로 동작했다면 GlusterFS와 MapRFS와 같이 유사한 성능을 보였을 것이다.

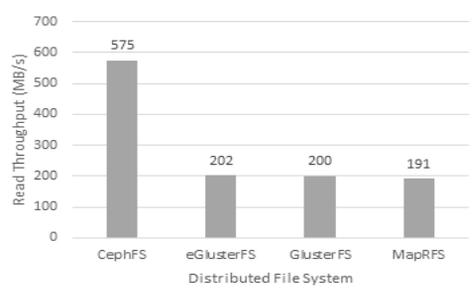

Fig. 4. Sequential Read Throughput (MB/s) in a General SSD Test Environment

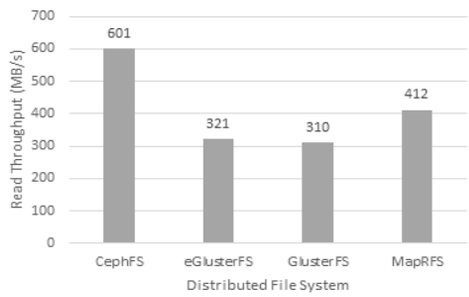

Fig. 5. Sequential Read Throughput (MB/s) in a NVMe SSD Test Environment

Fig. 6은 SysBench 벤치마크 프로그램으로 측정한 일반 SSD 테스트 환경에서의 OLTP 성능 결과 그래프이다. 네 가지 분산 파일 시스템 중 본 논문에서 제안한 쓰기 성능 향상 방법이 적용된 eGlusterFS가 가장 높은 성능을 보이며 GlusterFS 대비 약 6%의 성능 향상이 있었다.

Fig. 7은 SysBench 벤치마크 프로그램으로 측정한 NVMe SSD 테스트 환경에서의 OLTP 성능 결과 그래프이다. 마찬가지로 eGlusterFS가 가장 높은 성능을 보이며 GlusterFS 대비 약 20%의 성능 향상이 있었다. 본 논문에서 사용한 SysBench OLTP 테스트는 5개의 SELECT 쿼리, 2개의 UPDATE 쿼리, 1개의 DELETE 쿼리, 1개의 INSERT 쿼리가 랜덤으로 실행되어 결과적으로 읽기와 쓰기 비율이 약 75 : 25 이다. 이 실험의 결과로 보아 랜덤 읽기와 랜덤 쓰기가 섞여 있는 I/O 요청에 대해서 eGlusterFS의 성능이 다른 분산 파일 시스템보다 우수함을 알 수 있다.

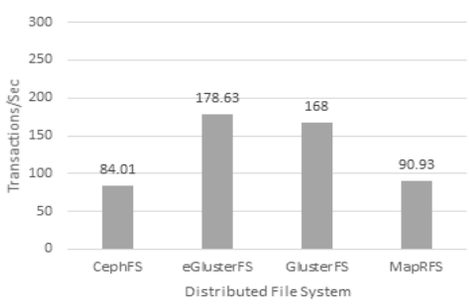

Fig. 6. OLTP Throughput (Transactions/Sec) in a General SSD Test Environment

Fig. 7. OLTP Throughput (Transactions/Sec) in a NVMe SSD Test Environment

5. 결 론

본 논문에서는 GlusterFS 기반 쓰기 성능 향상 방법을 제안하고 기존 분산 파일 시스템 중 MapRFS, CephFS, GlusterFS와 성능을 클라우드 환경에서 비교 평가하였다. 본 논문에서 제안한 쓰기 성능 향상 방법은 기존 동기식에서 사용한 디스크에 쓰는 방식을 메모리에 쓰는 방식으로 변경함으로써 응답 시간을 향상 시켰다. 실험 결과를 보면 본 논문에서 제안한 방법이 적용된 eGlusterFS는 순차 쓰기의 경우 일반 SSD 사용 환경에서는 GlusterFS 대비 약 11%, NVMe SSD 사용 환경에서는 약 7%의 성능 향상이 있었다. 그리고 OLTP를 통한 랜덤 쓰기 및 랜덤 읽기의 경우 일반 SSD 사용 환경에서는 GlusterFS 대비 약 6%, NVMe SSD 사용 환경에서는 약 20%의 성능 향상이 있었다. 이 결과를 통해 클라우드 환경에서 애플리케이션 특성에 따라 어떤 분산 파일 시스템이 적합한지 알 수 있다.

References

[1] M. A. Beyer and D. Laney, The Importance of 'Big Data': A Definition [Internet], https://www.gartner.com/doc/2057415/importance-big-data-definition.

[2] L. M. Roch, T. Aleksiev, R. Murri, and K. K. Baldridge. "Performance analysis of open source distributed file systems for practical large scale molecular ab initio, density functional theory, and GW+ BSE calculations," *International Journal of Quantum Chemistry*, Vol.118, No.1, 2018.

[3] G. Donvito, G. Marzulli, and D. Diacono, "Testing of several distributed file-systems (HDFS, Ceph and GlusterFS) for supporting the HEP experiments analysis," *Journal of Physics: Conference Series*, Vol.513, No.4, 2014.

[4] D. Gudu, M. Hardt, and A. Streit, "Evaluating the performance and scalability of the ceph distributed storage system," in *Proceedings of IEEE International Conference on Big Data (Big Data)*, 2014.

[5] FIO [Internet], https://github.com/axboe/fio

[6] SysBench [Internet], https://github.com/akopytov/sysbench

[7] Cooper, Brian F., Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears, "Benchmarking cloud serving systems with YCSB," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, pp.143-154. ACM, 2010.

[8] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," *ACM SIGPLAN Notices*, Vol.47, No.4, pp.37-48, 2012.

[9] A. Ghazal, T. Ivanov, P. Kostamaa, A. Crolotte, R. Voong, M. Al-Kateb, W. Ghazal, and R. V. Zicari, "BigBench V2: The New and Improved BigBench," in *Proceedings of the 33rd International Conference on Data Engineering (ICDE)*, pp.1225-1236, 2017.

[10] Z. Ren, W. Shi, J. Wan, F. Cao, and J. Lin, "Realistic and scalable benchmarking cloud file systems: Practices and lessons from AliCloud," *IEEE Transactions on Parallel & Distributed Systems*, Vol.28, No.1, pp.3272-3285, 2017.

[11] Y. Wu, F. Ye, K. Chen, and W. Zheng, "Modeling of distributed file systems for practical performance analysis," *IEEE Transactions on Parallel and Distributed Systems*, Vol.25, No.1, pp. 156-166, 2014.

[12] T. Harter, D. Borthakur, S. Dong, A. S. Aiyer, L. Tang, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Analysis of HDFS under HBase: a facebook messages case study," *in Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST)*, pp.199-212, 2014.

[13] M. Shamma, D. T. Meyer, J. Wires, M. Ivanova, N. C. Hutchinson, and A. Warfield, "Capo: Recapitulating Storage for Virtual Desktops," in *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST)*, pp. 31-45. 2011.

[14] Y. Gong, C. Hu, Y. Xu, and W. Wang, "A Distributed File System with Variable Sized Objects for Enhanced Random Writes," *The Computer Journal*, Vol.59, No.10, pp.1536-1550, 2016.

[15] S. Ghemawat, H. Gobioff, and S. Leung, "The Google File System," *SIGOPS Oper. Syst. Rev.*, Vol.37, No.5, pp.29-43, 2003.

[16] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the IEEE 26th Symposium on Mass Storage systems and Technologies (MSST)*, 2010.

[17] MapRFS [Internet], https://mapr.com/products/mapr-fs.

[18] S. A. Weil, S. A. Brandt, E. L. Miller, D. DE Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp.307-320, 2006.

[19] GlusterFS [Internet], https://docs.gluster.org.

[20] NFS-Ganesha [Internet], https://github.com/seoultower/nfs-ganesha

이 종 혁

https://orcid.org/0000-0002-8163-9388

e-mail : jonghyuk@cu.ac.kr

2004년 고려대학교 컴퓨터교육과(학사)

2006년 고려대학교 컴퓨터교육학과(석사)

2011년 고려대학교 컴퓨터교육학과(박사)

2011년~2012년 University of Houston, Post-Doc.

2012년~2017년 삼성전자 클라우드플랫폼그룹 책임연구원

2017년~현 재 대구가톨릭대학교 빅데이터공학과 조교수

관심분야 : 빅데이터, 클라우드컴퓨팅, 분산시스템, 인공지능