Journal of KIIT. Vol. 18, No. 5, pp. 13-19, May 31, 2020. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2020.18.5.13

익스텐트 맵을 이용한 온라인 스토리지 마이그레이션 방법

김진수*, 송석일**

Online Storage Migration Method using Extent Map

Jinsu Kim*, Seokil Song**

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2019R1A2C1005052)
또한, 2018년 한국교통대학교 지원을 받아 수행하였음.

요 약

온라인 스토리지 마이그레이션이란 서비스를 중단하지 않고 스토리지를 원격에 복제(Replication)하기 위한 연산이다. 온라인 스토리지 마이그레이션은 스토리지 교체, 복제, 데이터 센터 이전 등 다양한 응용에서 꼭 필요한 연산이다. 이 논문에서 제안하는 온라인 스토리지 마이그레이션 방법은 전송속도를 높이기 위해서 전송 대상 파일들의 익스텐트 맵을 생성하고 물리주소에 따라 순차 IO를 수행하여 파일들을 읽고 이를 타겟(target)으로 전송한다. 또한, 전송 중에 발생하는 소스(source) 측의 변경에 대해서 추적을 하고 이를 로그 레코드로 기록하여 전송 중의 변경을 타겟측에 반영할 수 있다. 이 논문에서는 제안하는 온라인 마이그레이션 방법을 실제로 구현하고 소스와 타겟의 일치성과 온라인 마이그레이션 수행 속도를 측정하는 실험을 수행하였다. 실험 결과 단순 원격복사 명령어인 scp 에 비해서 2배 이상 빠르게 마이그레이션을 수행함을 확인하였다.

Abstract

Online storage migration is an operation to replicate a storage remotely without interrupting service. This operation is essential for various applications such as storage replacement, replication, and data center relocation. The proposed online storage migration method in this paper creates an extent map of files to be transferred, performs sequential IO according to the physical address, reads the files, and transmits them to the target to increase the transfer speed. In addition, it is possible to track changes on the source side occurring during transmission and record them in log records to reflect the changes during transmission on the target side. Finally, in this paper, we implement the proposed online storage migration method and perform an experiment to measure the speed of online storage migration. As a result of the experiment, we show that the proposed migration method is performed twice or more faster than the simple remote copy command, scp.

Keywords
data migration, online, replication, storage system

* ㈜ 데이터커맨드 주임연구원
- ORCID: https://orcid.org/0000-0002-8205-4859
** 한국교통대학교 컴퓨터공학전공 교수(교신저자)
- ORCID: https://orcid.org/0000-0002-0110-7155

· Received: Mar. 31, 2020, Revised: Apr. 24, 2020, Accepted: Apr. 27, 2020
· Corresponding Author: Seokil Song
School of Computer Engineering & Information Technology, Korea National University of Transportation, Daehakro 50, Chungju, Chungbuk 27469, Korea
Tel.: +82-43-841-5349, Email: sisong@ut.ac.kr

Ⅰ. 서 론

온라인 스토리지 마이그레이션(Online storage migration)은 서비스를 중단하지 않고 스토리지를 원격에 복제(Replication)하거나 스토리지 교체, 부하균등(Load balancing) 등을 수행하는 데 있어 중요한 연산이다[1]-[3]. 온라인 스토리지 마이그레이션을 수행하기 위해서는 소스(Source) 스토리지의 내용이 지속적으로 변경하는 상황에서 소스 스토리지의 데이터를 일관성 있게 타겟(Target) 스토리지에 전송해야 한다.

대표적인 온라인 마이그레이션 도구로 RSync[4]가 있다. RSync는 리눅스 운영체제에서 기본으로 제공하는 도구로 소스 디렉토리를 타켓 디렉토리에 동기화시키는 기능을 수행한다. 동기화를 위해 RSync는 소스 디렉토리와 타겟 디렉토리의 파일들을 읽어서 시그니쳐(Signature)를 생성하고 시그니쳐를 비교하여 델타(Delta)를 생성한다. 최종적으로 델타를 타겟 디렉토리에 적용하여 소스 디렉토리와 타겟 디렉토리를 일치시킨다.

RSync는 소스와 타겟 디렉토리간 동기화를 수행할 수 있지만, 이를 이용해서 온라인 마이그레이션을 수행하기 위해서는 지속적인 동기화를 수행해야 한다. 또한, RSync는 변경되는 파일의 시맨틱을 파악하지 못하기 때문에 이름이 변경된 파일이나 이동된 파일을 신규 파일로 인식할 수 있고, 이에 따라 재전송 동작을 수행하게 되므로 비효율성이 존재한다.

Aspera Sync[5]는 소스 디렉토리에 대한 스냅샷을 생성하고 소스 스냅샷 대비 타겟의 변경 내용을 추적하여 타겟 디렉토리에 대한 동기화를 수행할 수 있다. Aspera Sync는 스토리지의 변경이 발생할 때마다 동작하여 동기화를 수행할 수 있다.

Lsyncd[6]는 소스 노드와 타겟 노드사이의 특정 디렉토리를 동기화시켜주는 오픈소스 프로젝트로 rsync 와 inotify[7]를 이용해서 구현되었다. 소스 노드의 inotify에서 제공되는 파일 시스템 이벤트를 일정시간 동안 모아서 rsync를 통해 타겟 노드로 전송하여 동기화를 수행한다.

기존에 제안된 방법들은 모두 타겟 디렉토리에 파일이 이미 있을 때 이를 동기화 하는데는 적합하지만 최초에 소스 디렉토리의 파일들을 타겟 디렉토리로 마이그레이션하는데는 적합하지 않다. 최초에 소스 디렉토리의 파일들을 타겟 디렉토리로 이동시키기 위해서는 모든 파일에 대한 원격 파일 전송이 수행되어야 하며 전송시 소스 디렉토리에 가해지는 변경을 타겟에 반영해야 한다.

기존 방법들을 이용해서 이를 수행하기 위해서는 먼저 소스 디렉토리의 파일들을 어떤 방법을 통해서 타겟으로 전송한 후 타겟과 소스의 차이를 찾아내어 다시 동기화를 시키는 과정이 수행되어야 한다. 변경이 지속되는 상황에서는 소스와 타겟을 완전히 일치시키기 위해서는 반복적인 동기화 과정이 필요하며 상황에 따라서는 끝나지 않을 수도 있다.

이 논문에서는 파일 시스템의 변경을 실시간으로 추적하는 델타 기술[2]과 파일의 물리적 주소인 익스텐트 맵(Extent map)을 이용하여 효과적으로 소스의 파일들을 타겟으로 전송하는 온라인 스토리지 마이그레이션 기법을 제안한다. 제안하는 방법은 파일의 익스텐트 맵을 통해서 연속 읽기를 가능하게 하며 파일의 변경을 변경 연산 단위로 추적하여 변경분을 빠르게 추적하여 전송이 가능하다.

파일을 읽기 위해서는 내부적으로 파일의 논리 주소를 물리 주소로 바꾼 이후, 저장소에서 읽어야 하는데, 파일 단편화가 많은 경우 파일을 논리적 연속으로 읽기 위해서 임의(Random) I/O가 발생하게 되어 디스크 접근 시간이 늘어나게 된다. 파일 전송 중에 전송중인 파일의 내용을 변경하게 되면, 경우에 따라 로컬 노드의 데이터 내용과 원격 노드의 데이터 내용이 다르게 되는 경우가 발생할 수 있다. 파일 단위로 데이터를 전송하기 때문에 만약 데이터 전송 중, 데이터 전송을 중단하게 되면, 다시 이 파일을 보내기 위해서 파일 전체를 다시 보내는 경우가 발생할 수 있다.

본 연구에서는 위와 같은 문제를 해결하기 위해서 파일의 FIEMAP[8]을 사용해 논리 주소를 물리 주소로 매핑한 후 물리적 주소순으로 디스크에 접근하여 순차(Sequential) I/O를 이용해 처리하도록 한다. 또한, FIEMAP을 통해 생성된 매핑 데이터를 유지하여 데이터 전송이 중간에 중단되더라도 중단된 지점부터 데이터를 보낼 수 있도록 한다. 또한, 델타 기술을 사용하여 온라인으로 데이터를 전송할 수 있도록 한다.

II. 제안하는 온라인 스토리지 마이그레이션 방법

이 논문에서는 소스 노드의 데이터 파일들을 타겟 노드에 온라인으로 동기화하는 방법을 제안한다. 동기화의 일반적인 절차는 소스 노드의 데이터 파일들을 읽는 단계와, 읽은 데이터를 네트워크를 통해 타겟 노드로 전송하는 단계, 타겟 노드가 전송된 데이터를 기록하는 단계로 구성된다.

온라인 동기화는 위와 같은 절차로 동기화를 수행하는 과정 중에 소스 노드가 변경될 때 변경사항을 타겟 노드에 반영하여 소스 노드와 타겟 노드 사이에 데이터의 일관성을 보장하기 위한 것이다. 동기화를 위해서 소스 노드의 데이터 파일들을 읽을 때 파일 단위로 읽게 되면 대량의 임의 IO가 발생할 수 있다. 임의 IO는 동기화 속도를 크게 저하시킬 수 있다. 본 논문에서는 임의 IO 문제를 동기화 대상 데이터 파일들을 구성하는 익스텐트들에 대한 물리적 맵(디스크 및 페이지 캐시 주소)을 생성하고 소스의 데이터 파일들을 익스텐트들의 물리 주소 순서에 따라 순차적으로 접근하여 완화한다.

델타 기술을 이용하여 온라인 동기화가 수행되는 중에 소스 노드에 발생된 변경을 캡처하여 저장하고 동기화 완료 후 타겟에 반영하여 소스 노드와 타겟 노드 간의 데이터 일관성을 보장한다. 소스 노드의 데이터가 대용량일 때는 온라인 동기화에 소요되는 시간이 수 시간 ~ 수 일이 될 수도 있다. 온라인 동기화에 소요되는 시간이 길어지는 경우 소스 노드가 원활하게 스토리지 서비스를 제공할 수 있도록 하기 위해서 온라인 동기화를 일시 중지시키는 경우도 있다. 이때 소스 노드에 델타 기술에 의해 생성되는 로그의 양이 매우 커질 수 있어 공간 및 로그 반영 시간 측면에서 비효율적이다. 이를 해결하기 위해서 동기화 중에 소스 노드에서 변경되는 사항을 로그로 생성하지 않고 단지 익스텐트 맵에만 기록하고 1차 동기화(InitialCopy)가 완료되면 익스텐트 맵을 참조하여 변경사항을 다시 동기화(ReCopy)하는 방법을 사용한다. ReCopy 대상의 크기가 어느 이하가 되면 ReCopy 도중 로그를 생성하고 동기화를 수행한 후 로그를 반영하여 동기화를 완료한다.

마지막으로, 타겟 노드의 운영체제와 파일시스템에 독립적인 마이그레이션을 수행할수 있도록 하기 위해서 운영체제와 파일시스템에 중립적인 파일 시스템 연산 목록(FSOP, File System Operations)을 정의하고 FSOP와 각 타겟 파일 시스템의 연산과의 매핑 테이블을 만들어서 FSOP를 해당 파일시스템에 맞도록 변환하여 수행하는 방법을 취한다.

그림 1은 이 논문에서 제안하는 온라인 동기화 시스템 구성도를 보여준다. Snapshot Manager는 소스 노드의 사용자가 지정한 디렉토리에 한 개 이상의 스냅샷을 생성한다. Delta Manager는 델타 기술을 이용해 소스 노드의 사용자가 지정한 특정 디렉토리에 행해지는 모든 변경 연산을 기록한다. EXTENT MAP Manager는 스냅샷을 통해 생성된 디렉토리 내의 각 파일들을 구성하는 익스텐트 테이블을 생성한다. 생성한 이후에는 Delta Manager가 추적한 변경연산을 분석하여 익스텐트 테이블을 업데이트한다.

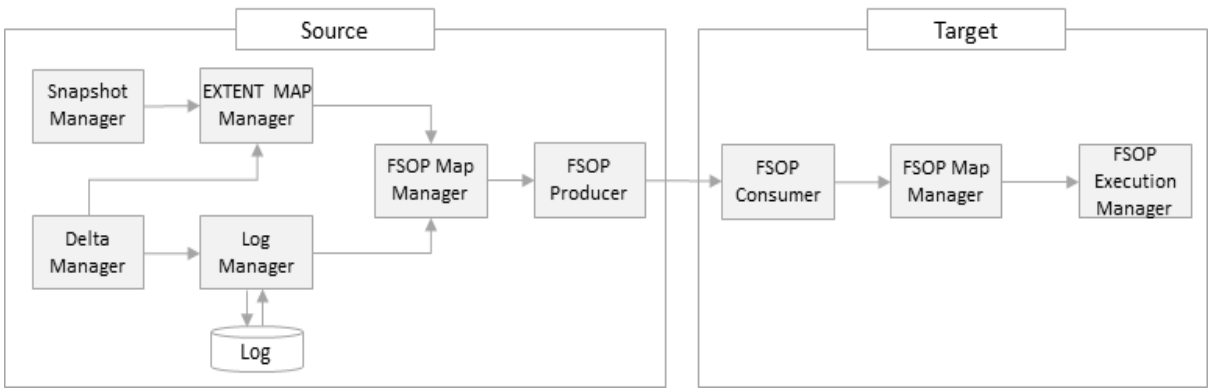

그림 1. 온라인 스토리지 마이그레이션 시스템 구성도
Fig. 1. System architecture of online storage migration system

Log Manager는 Delta Manager가 추적한 변경 연산에 대해서 로그 레코드를 생성한다. 구조변경 및 내용 변경에 대한 로그를 생성하며 상황에 따라 구조변경 로그 레코드만 생성 할 수도 있다. 소스 노드 FSOP Map Manager는 Log Manager가 생성한 로그레코드 및 FIEMAP을 사전에 정의된 FSOP 매핑 테이블을 참조하여 로그 레코드에 포함된 소스 노드측 파일시스템 변경 연산과 FIEMAP의 익스텐트를 중립 FSOP로 변환한다. 타겟 노드측에서는 소스 노드가 전송한 중립 FSOP를 타겟측 FSOP로 변환한다.

FSOP Producer는 FSOP를 네트워크를 통해서 타겟 노드 측에 전송한다. FSOP Consumer는 소스 노드 측에서 제공하는 FSOP를 네트워크를 통해 수신한다. FSOP Execution Manager는 타겟 노드의 FSOP Map Manager가 타겟측 FSOP 로 변환 한 것을 실행하여 동기화 수행한다.

표 1은 하둡(Hadoop) 파일시스템인 HDFS[9]와의 매핑을 수행하는 FSOP 매핑 테이블이다. FSOP는 다양한 운영체제 혹은 파일시스템에서 같은 목적으로 제공하는 함수들을 하나의 통일된 형식으로 바꾸는 것을 말한다. 예를 들어, 파일 쓰기를 하기 위해서 리눅스에서 write, writev, pwrite, pwritev, mmap, msync 시스템 호출[10]을 제공하며, HDFS에서는 append, copyFromLocalFile 등을 제공하는데 이들 간의 매핑을 FSOP_WRITE로 정의한다.

온라인 동기화를 시작하면 가장 먼저 디렉토리 스냅샷을 생성한다. 디렉토리 스냅샷은 소스 노드의 데이터 파일들에 대한 디렉토리 구조와 각 디렉토리에 포함된 파일들의 하드링크들로 구성된다.

생성된 디렉토리 스냅샷의 모든 하드링크들을 스캔하여 파일들에 대한 익스텐트 맵(FIEMAP)을 생성한다. FIEMAP은 그림 2의 페이지 캐시에 대한 FIEMAP과 그림 3의 HDD에 대한 FIEMAP으로 구성된다. 페이지 캐시에 대한 FIEMAP은 각 파일을 구성하는 페이지가 페이지 캐시에 있는 것을 나타내며 HDD에 대한 FIEMAP은 각 파일을 구성하는 익스텐트들의 하드디스크 상의 물리 주소를 나타낸다.

생성한 디스크 FIEMAP의 엔트리들을 물리적 주소를 기준으로 정렬한다. 동시에 델타 모듈을 이용해서 소스 노드의 파일시스템의 변경을 추적하고 파일 업데이트, 생성, 삭제에 대해서는 변경 사항을 FIEMAP에 반영한다. 그리고 구조 변경(디렉토리 및 파일 생성/삭제/이름변경) 연산은 로그 레코드를 생성하여 저장한다.

표 1. Linux VFS - HDFS 매핑 FSOP 리스트
Table 1. FSOP list with mapping Linux VFS operations to HDFS operations

FSOP	LINUX VFS	HDFS
WRITE	WRITE WRITEV PWRITE PWRITEV MMAPMSYNC	append UD_Update moveFromLocalFile
OPEN	OPEN CREAT OPENAT	open create createNewFile
UNLINK	UNLINK UNLINKAT	delete
MKDIR	MKDIR MKDIRAT MKNODE MKNODEAT	mkdirs
RMDIR	RMDIR	delete
RENAME	RENAME RENAMEAT	Rename
TRUNCATE	TRUNCATE	UD_Update moveFromLocalFile
CLOSE	CLOSE	close
LINK	LINK LINKAT	createSymlink
SYMLINK	SYMLINK SYMLINKAT	createSymlink

Fullpath	Cache Bitmap
a.txt	0010...0010
b.txt	1000...1000

그림 2. 페이지 캐시 FIEMAP
Fig. 2. Page cache FIEMAP

Fullpath	Extent#	Logical Addr	Physical Addr	Length	Flags
a.txt	0	0	7918845952	134217728	0
a.txt	1	134217728	8053063680	134217728	1
b.txt	0	0	7247757312	134217728	0
b.txt	1	134217728	7381975040	134217728	1

그림 3. HDD FIEMAP
Fig. 3. HDD FIEMAP

이후 InitialCopy 절차를 진행한다. InitialCopy는 FIEMAP에 따라 물리적 주소로 정렬된 익스텐트들을 순차적으로 읽어서 타겟으로 전송하는 절차이다. InitialCopy 에서는 구조 변경을 추적하여 로그레코드로 기록한다. 구조 변경에 대한 로깅은 마이그레이션 연산이 일시정지된 상태에서도 계속된다.

동시에 변경연산 (write, truncate)에 의해 파일이 변경(Update, append, shrink) 되면 이를 그림 3의 FIEMAP 테이블의 Update 컬럼에 반영한다. FIEMAP 테이블의 Update 컬럼은 U/A/S 중 하나의 플래그로 변경된다. U(Update)는 해당 익스텐트 전체 또는 일부가 변경될 경우에 해당하고, A(Append)는 익스텐트가 추가되었을 때 FIEMAP 테이블에 새로운 엔트리를 추가하고 Update 컬럼에 A를 기록한다. S(Shrink)는 엔트리의 익스텐트가 삭제되는 경우에 해당하는 플래그 이다. 만일 익스텐트의 일부만 삭제된다면 Length 컬럼을 조정한다. 이름변경 연산(rename)에 의해 파일의 경로가 변경될 경우에는 이를 그림 4의 RENAME 테이블에 기록한다.

또한, 반복적으로 정렬된 FIEMAP의 각 엔트리의 익스텐트들 중 Update 컬럼의 플래그에 C가 설정되지 않은 것들을 타겟에 복사하고 해당 엔트리의 Update 컬럼에 C 플래그를 추가한다.

마지막으로 FIEMAP의 익스텐트들을 모두 복사하면 그림 2의 페이지 캐시 FIEMAP에 대한 FIE(B)MAP을 통해서 어떤 페이지가 캐시에 있는지 확인한다. 페이지 캐시에 있는 데이터는 다시 타겟 노드에 복사하InitialCopy 절차가 완료되면 ReCopy 절차가 진행된다. ReCopy 절차에서는 HDD FIEMAP 엔트리들 중 Update 컬럼에 U, A 플래그가 설정된 엔트리들의 비율을 계산한다. 해당 비율이 사전에 주어진 비율보다 크면 InitialCopy 절차를 다시 진행한다. 고 InitialCopy 절차를 종료한다. 비율이 사전에 주어진 비율보다 작으면 구조변경 및 콘텐츠 변경에 대한 로그레코드를 기록하기 시작한다.

OrigPath	NewPath
a.txt	a1.Txt
b.txt	a1.txt

그림 4. Rename 테이블
Fig. 4. Rename table

이후 FIEMAP 테이블의 Update 컬럼중 U, N 플래그가 설정된 엔트리들의 익스텐트들을 타겟에 복사한다. 모두 복사하면 그동안 생성한 구조 변경 및 콘텐츠 변경 로그레코드들을 타겟으로 전송하여 반영하고 마이그레이션 절차를 마친다.

III. 성능 평가

이 논문에서는 제안하는 온라인 마이그레이션 방법의 성능을 입증하기 위해서 리눅스 운영체제에서 구현 하고 실험을 수행하였다. 실험은 온라인 마이그레이션의 수행시간과 온라인 마이그레이션 도중 소스에 변경 연산이 수행되더라도 이를 타겟에 일관성 있게 반영하는지를 측정하였다. 실험에 사용된 HW 및 SW 환경은 표 2에 나타나 있다. 실험에는 최대 500GB의 임의로 생성한 파일 및 디렉토리들을 사용하였다.

표 2. 실험에 사용된 HW 및 SW 환경
Table 2. Experimental environment of HW and SW

Category		Description
HW · SW Spec.	Source node	Intel(R) Xeon(R) CPU E31220@3.10GHz, Memory 32G, Ubuntu 18.04.1 LTS
	Target node	Intel(R) Xeon(R) CPU E31220@3.10GHz, Memory 32G, centos7
	Network	Intel 82574L Gigabit Network (Bandwidth 1G)
Data set	Max 500GBytes Randomly generated files and directories	

먼저, 온라인 마이그레이션 도중에 소스에 일정량의 변경을 가했으며 변경한 내용이 타겟에 모두 반영되었는지 측정하였다. 실험결과 소스와 타겟이 정확이 일치함을 알 수 있었다. 또한, 전송 시간의 효율성을 보기 위해서 동일한 데이터에 대한 scp 명령어 수행 결과와 제안하는 방법을 비교하였다. 기존의 마이그레이션 방법들은 동기화를 위한 사전처리 등으로 인해 타겟이 완전히 비어있는 상황에서는 초기 동기화에 수행되는 시간이 scp에 비해 훨씬 오래 걸리게 된다.

이런 이유로 이 논문에서는 동기화 시간을 scp와 비교하였다. 표 3은 실험결과를 보여준다. 실험에 사용된 데이터를 200GB에서 500GB로 변화시키면서 마이그레이션에 소요되는 시간을 측정하였다. 표 3에서 보는 것처럼 제안하는 방법이 2배 이상 빠른 것을 볼 수 있다.

표 3. scp와 제안하는 방법의 마이그레이션 속도 비교
Table 3. Comparison of migration speed between scp and the proposed method

Division	500GB	400GB	300GB	200GB
scp (min)	131	113	92	69
Proposed method (min)	56	47	35	21

IV. 결 론

이 논문에서는 스토리지 교체, 복제, 데이터 센터 이전 등 다양한 응용에서 꼭 필요한 온라인 스토리지 마이그레이션 방법을 제안하였다. 제안하는 온라인 스토리지 마이그레이션 방법은 전송속도를 높이기 위해서 전송 대상 파일들의 익스텐트 맵을 생성하고 물리주소에 따라 순차 IO를 수행하여 파일들을 읽고 이를 타겟으로 전송하도록 하였다.

또한, 전송 중에 발생하는 소스 측의 변경에 대해서 추적을 하고 이를 로그레코드로 기록하여 전송중의 변경을 타겟측에 반영할 수 있게 하였다. 실험을 통해서 온라인 마이그레이션이 수행된 후 소스와 타겟 스토리지의 일치성을 비교한 결과 정확히 일치함을 볼수 있었다. 또한, 온라인 마이그레이션 수행 속도를 측정하는 실험에서도 기존의 리눅스 명령어인 scp에 비해서 2배 이상 수행속도가 더 빠름을 확인하였다.

References

[1] A. Modi, A. Raghavendra, and P. S. Thilagam, "Live migration of virtual machines with their local persistent storage in a data intensive cloud", International Journal of High Performance Computing and Networking, Vol. 10, No. 1-2, pp. 134-147, Jan. 2017.

[2] J. Kim and S. Song, "Online migration method for storage based on delta technology", In Proceedings of the Korea Contents Association Conference, Mokpo, South Korea, pp. 13-14, May 2018.

[3] Y. Yang, B. Mao, H. Jiang, Y. Yang, H. Luo, and S. Wu, "SnapMig: Accelerating VM live storage migration by leveraging the existing VM snapshots in the cloud", IEEE Transactions on Parallel and Distributed Systems, Vol. 29, No. 6, pp. 1416-1427, Jun. 2018.

[4] F. Purnama, U. Tsuyoshi, and M. I. Royyana, "Rsync and Rdiff implementation on Moodle's backup and restore feature for course synchronization over the network", In Proceedings of 2016 IEEE Region 10 Symposium (TENSYMP), Bali, Indonesia, pp. 24-29, May 2016.

[5] https://www.ibm.com/products/aspera/sync. [accessed; May 17, 2020]

[6] A. Kittenberger, "Lsyncd - Live Syncing (Mirror) Daemon", [online] Available: https:// axkibe.github.io/lsyncd/. [accessed; May 17, 2020]

[7] C. Fisher, "Linux filesystem events with inotify", Linux Journal, Article No. 2, Jan. 2018.

[8] https://www.kernel.org/doc/Documentation/filesystems/fiemap.txt [accessed; May 17, 2020]

[9] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system", In Proceedings of 2010 IEEE 26th symposium on mass storage systems and technologies (MSST) IEEE, Incline Village, NV, USA, pp. 1-10, May 2010.

[10] http://man7.org/linux/man-pages/man2/syscalls.2.html [accessed; May 17, 2020]

저자소개

김 진 수 (Jinsu Kim)

2015년 8월 : 한국교통대학교 컴퓨터공학과(학사)
2017년 2월 : 한국교통대학교 컴퓨터공학과(석사)
2020년 2월 : 한국교통대학교 컴퓨터공학과(박사수료)
2020년 3월 ~ 현재 : 데이터커맨드 주임 연구원
관심분야 : 데이터베이스, 빅데이터, 스토리지 시스템 등

송 석 일 (Seokil Song)

1998년 2월 : 충북대학교 정보통신공학과(공학사)
2000년 2월 : 충북대학교 정보통신공학과(공학석사)
2003년 2월 : 충북대학교 정보통신공학과(공학박사)
2003년 7월 ~ 현재 : 한국교통대학교 컴퓨터공학과 교수
관심분야 : 데이터베이스, 센서 네트워크, 스토리지 시스템 등