JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 24, No. 11, pp. 2953-2961, Nov. 2023

Check for updates

데이터의 I/O 특성에 따른 병렬 분산파일시스템 구축 및 성능 검증

윤 준 원[1] · 송 의 성[2*]
[1]한국과학기술정보연구원 슈퍼컴퓨팅인프라센터 책임연구원
[2*]부산교육대학교 컴퓨터교육과 교수

Construction and Performance Verification of a Parallel Distributed File System According to Data I/O Characteristics

Junweon Yoon[1] · Ui-Sung Song[2*]

[1]Principal Researcher, Department of Supercomputing Center, KISTI, Daejeon 34141, Korea
[2*]Professor, Department of Computer Education, Busan National University of Education, Busan 47503, Korea

[요 약]

사용자 어플리케이션의 요구 성능은 머신러닝, 딥러닝 등과 같은 AI 활용 연구가 적용되면서 더욱 높아지고 복잡해지고 있다. 특히 AI 기반의 작업 환경은 전통적인 CPU 아키텍처에서 병렬처리 연산에 강화된 수천 개의 산술연산장치(ALU)를 사용하는 GPU와 같은 이기종 아키텍처로 의존성이 높아지고 있다. 이렇게 HPC에 다양한 연산 장치를 추상화하여 고성능 연산을 지원하는 이기종 컴퓨팅 환경이 접목되면서 대량의 다양한 패턴의 데이터 I/O 처리 요구사항 또한 높아지고 있다. 본 논문에서는 HPC에서 다양한 사용자의 어플리케이션 수행 시에 요구되는 스토리지의 병렬 I/O 처리에 대한 특성과 지원하기 위한 기법을 제시한다. 이를 위해 최근 HPC, AI 환경에서 요구되는 다양한 I/O 패턴과 High- throughput, Random Small I/O 데이터 처리를 위한 최적 환경을 병렬 파일시스템에 실제 적용하고 성능을 검증한다. 나아가 I/O 패턴에 따라 고속의 저장 매체를 정책적으로 사용할 수 있는 계층형 스토리지(Tiering Storage)를 구축하여 기능과 성능을 확인한다. 본 연구 결과는 최근 다양한 사이트에 도입되는 고성능 스토리지와 병렬파일시스템 구축 사례 연구로 활용될 수 있다. 또한 향후에는 확장성을 높인 성능 검증이 요구된다.

[Abstract]

The required performance of user applications is becoming higher and more complex as AI-driven research such as machine learning(ML), deep learning(DL) is applied. In particular, AI-based system configurations are increasingly dependent on heterogeneous architectures such as GPUs that use thousands of arithmetic logic units(ALUs) specialized in parallel processing operations from traditional CPU architectures. As a heterogeneous computing environment that supports high-performance computation by abstracting various computing devices into HPC, the requirements for processing large amounts of various patterns of data I/O are also increasing.

This paper presents the characteristics and technologies for parallel I/O processing of storage required when performing applications by various users in HPC. This analyzes the various patterns of I/O requirements required in recent HPC, AI-based studies. In addition, the optimal configuration for high-throughput, Random Small I/O data processing is applied to the parallel file system, and its performance is verified. Furthermore, verify the function and performance by configuring tiering storage that can systematically use high-speed storage media according to I/O patterns. The results of this research can be used as a case study for the deployment of high-performance storage and parallel file systems that have recently been deployed at various sites. Furthermore, future work will require performance validation with increased scalability.

색인어 : 병렬파일시스템, 스토리지, 이기종 컴퓨팅, 인공지능, 벤치마크, GPU
Keyword : Parallel Filesystem, Storage, Heterogeneous Computing, AI, Benchmark, GPU

http://dx.doi.org/10.9728/dcs.2023.24.11.2953

Received 27 September 2023; **Revised** 19 October 2023
Accepted 24 October 2023

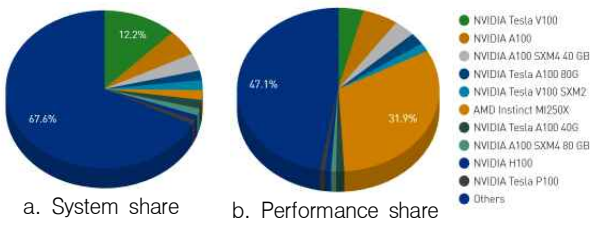
***Corresponding Author; Ui-Sung Song**

Tel: +82-51-500-7326
E-mail: ussong@bnue.ac.kr

Ⅰ. 서 론

HPC (High Performance Computing) 환경은 거대 규모 어플리케이션 수행에 대표적인 도구로 계산 자원의 성능과 규모, 고속의 인터커넥트, 고성능 스토리지 등이 밀결합 되어 최대 성능을 도출할 수 있다. 대표적인 HPC 환경에서 주로 수행되는 구조분석/해석, 양자역학, 분자동역학, 유체역학 등의 응용 소프트웨어에서부터 최근 부각되는 빅데이터와 AI 분야의 머신러닝, 딥러닝과 같은 대규모 데이터 처리 환경까지 확대되고 있다. 특히 AI 모델은 의미 분석이 필요한 방대한 데이터 세트내의 연관성을 단순한 규칙에 따라 해석하는 것이 아니라 인간의 습득 과정과 유사하게 경험과 학습을 통해 의사 결정을 수행한다[1]. 최근 AI 알고리즘(모델)의 개선과 더불어 양질의 데이터를 수집, 저장하고 구축하기 위한 노력이 지속적으로 진행되고 있다[2].

한편, CPU 아키텍처는 고든 무어(Gordon Moore)가 제시한 법칙에 따라 반도체 집적회로의 성능이 증가하였으나 2010년도에 이르러 트랜지스터의 물리적 직접도 한계에 봉착하였고 멀티/매니 코어(Multi/Many core)와 같은 병렬 형태의 프로세서로 진화하였다[3]. AI 기반의 연구 환경은 대량의 데이터를 분석하고 연관성을 학습, 추론하는 환경으로 계산량에 대한 요구사항과 의존성이 높아지면서 CPU 기반의 시스템에서 병렬 연산을 더욱 강화할 수 있는 이기종 아키텍처(Heterogeneous Architecture) 형태로 수요가 늘어나고 있다[4]. 이기종 시스템은 GPU와 같은 보조 프로세서(Coprocessor)를 활용하여 연산 처리량을 가속 시킬 수 있으며, CPU와 이기종 아키텍처 간의 데이터 처리를 추상화하여 수행할 수 있는 소프트웨어 환경도 포함된다. 고성능컴퓨팅의 세계적인 추세를 파악할 수 있는 슈퍼컴퓨터 TOP500을 살펴보면 세계 1위의 슈퍼컴퓨터 연산 능력은 10년간 60배가 넘게 증가하였다. 특히 미국 에너지부(DoE) 산하 오크리지 국립연구소(Oak Ridge National Laboratory)의 Frontier 시스템은 Linpack ExaFlops(초당 100경,[1018] 계산속도) 임계값을 넘은 최초 슈퍼컴퓨터로 '22년 상반기 Top500에서 실측성능(Rmax) 1.194 EFlops 성능을 기록하였다. 이 시스템은 AMD GPU가 장착된 시스템이며 지난 수년간 NVIDIA GPU를 주축으로 이기종 시스템들이 지속적으로 발전해왔다.



a. System share b. Performance share

그림 1. TOP500 가속기/코프로세서 시장 점유율(ISC23)
Fig. 1. TOP500 share of Accelerator/Co-Processor (ISC23)

ISC23 ('23.6) 리스트의 TOP10 중 7개 사이트가 GPU 기반 슈퍼컴퓨터가 등재되어 있으며 그림 1은 이기종 시스템에 장착되어 있는 제품군의 점유율을 나타내고 있다[5].

본 논문에서는 HPC 환경에서의 계산자원의 특징 및 데이터 I/O 특성에 따른 병렬 분산파일시스템을 분석한다. HPC, AI 등의 다양한 I/O 패턴을 동적으로 유연하게 처리하기 위한 파일시스템의 기술들을 논하고 실제 시스템에 적용함으로써 유효성과 성능 향상을 검증한다. 본 연구는 다양한 사용자 어플리케이션 I/O 패턴에 대응하는 파일시스템 도입시 지침이 될 수 있다.

Ⅱ. 연구 배경

수년간 AI 발전을 이끈 딥러닝(Deep Learning) 분야의 연구는 인간 뇌 신경망(Neural Network)의 뉴런(Neuron)이라는 연속적인 작은 세포 단위의 연결된 구조를 묘사했으며 시냅스(Synapse)라는 세포를 통해 다수의 전기 자극을 입력으로 받아 저장 또는 새로운 자극을 생성하여 넘긴다. 인간의 뇌에는 뉴런 뇌세포가 1,000억 개 정도가 있고 시냅스의 연결이 100조 개나 존재한다[6]. 이런 방대한 규모의 빅데이터를 스스로 학습하는 머신러닝, 딥러닝 기법은 결과의 정밀도와 정확도를 높이기 위해 병렬 연산(Parallel Computing) 기법이 반드시 필요하며 계산 자원의 요구사항도 CPU 클럭과 같은 집적도에 따른 성능에서 분산 처리 환경으로 변화하였다[7].

GPU는 초기 대량의 픽셀(Pixel)을 처리하기 위한 목적에서 시작되었으나 데이터 처리량이 기하급수적으로 증가되면서 CPU보다 클럭 수는 낮지만 GPU 내 수천에서 수만개의 코어를 배치, 활용함으로써 병렬 연산 능력을 최대한 이끌어 낼 수 있다. GPU에 장착된 수많은 산술연산장치(ALU)는 하나의 명령어로 복수 개의 데이터를 처리(SIMT, Single Instruction, Multiple Threads)하게 되며 기본 단위인 SP (Scala 또는 Stream Processor)가 모여 SM (Streaming Multiprocessor)으로 구성된다. 특히 한 번에 병렬로 처리되는 쓰레드 그룹을 워프(Warp)라고 하며 SM 단위로 수행된다. 또한 GPU 연산의 워크로드가 항상 높은 정밀도를 필요로 하지 않기 때문에 GPU에서 제공하는 여러 형태의 정밀도 (FP8~64, BF16, INT4~8 등)를 사용할 수 있다. 모델의 규모 데이터 처리방식이 각각 다양하기 때문에 정밀도 전달에 많은 비트를 사용하는 것은 처리속도와 메모리 사용량을 낭비할 수 있다. NVIDIA 볼타(Volta), 튜링(Turing), 암페어(Ampere), 호퍼(Hopper) GPU 아키텍처에서 지원되는 다중 정밀도(Multi Precision), 혼합 정밀도(Mixed Precision) 처리는 응용프로그램 수행시 다른 정밀도 수준을 사용하여 정확도를 유지하면서 계산 처리속도의 효율성을 얻을 수 있어 AI 학습 속도를 가속화 시킬 수 있다[8]. HPC 환경은 전통적인 CPU 기반 아키텍처에서 이기종 시스템 환경 그리고 데이터

처리 방법 다변성 등을 고려한 I/O 처리 환경이 응용 프로그램의 성능을 좌우한다. 병렬 파일시스템 환경도 동적으로 데이터를 처리하고 가속화를 지원하는 기술들이 적용되고 있다.

Ⅲ. 관련 연구

3-1 고성능 병렬 파일시스템

병렬 파일 시스템(Parallel File System)은 대규모 데이터를 효과적으로 저장하고 고속의 I/O를 제공하기 위한 파일 시스템으로 데이터를 여러 서버 노드에 분산하여 저장하고 여러 클라이언트가 동시에 파일을 읽고 쓸 수 있는 고성능 환경을 제공한다. 대표적인 병렬 파일 시스템으로는 Lustre, GPFS (IBM Spectrum Scale), BeeGFS, Ceph 등이 있으며 대규모의 계산 과학 및 공학 연구, 데이터 분석, 기상 모델링 및 다양한 고성능 컴퓨팅 애플리케이션에서 널리 사용된다. 본 연구에 사용할 파 Lustre는 대표적인 고성능 병렬 파일 시스템으로 대규모 클러스터 컴퓨팅 환경에서 데이터를 저장하고 관리하기 위해 설계되었다. Lustre는 파일 시스템 데이터를 여러 서버에 분산하여 저장하므로 대용량 데이터의 고성능 액세스를 지원하며 메타데이터 서버와 데이터 서버를 분리하여 고성능 메타데이터 액세스를 제공한다. 또한 수천 대의 노드와 페타바이트(PB) 단위로 확장 가능한 디자인을 가지고 있어 대규모 클러스터 컴퓨팅 환경에서 요구하는 성능과 용량을 제공한다. Lustre는 데이터를 여러 디스크에 나누어 저장하고 병렬 I/O를 지원하기 위해 데이터를 스트라이핑(Striping) 하는데 이로써 데이터의 분산과 고성능 처리가 가능하다. 특히 Lustre 클라이언트는 데이터를 로컬 캐시에 저장하여 반복적인 액세스를 최적화하고 전체 시스템 부하를 줄이는 기능도 제공한다. 또한 Lustre 관련 제품군에서는 관리 및 모니터링을 위한 도구를 제공하여 파일 시스템 상태를 모니터링하고 즉각적으로 성능 상태와 장애를 인지할 수 있게 제공해준다. Lustre는 객체 기반 파일시스템으로 그림 3과 같이 메타데이터 서버(MDS), 객체 저장 서버(OSS), 클라이언트 노드로 구성되며 각각은 TCP/IP, InfiniBand[9], OPA 등[10]과 같은 고속의 인터커넥트를 이용하여 연결된다. 또한 Lustre는 메타데이터 서버와 객체 저장 서버의 이중화를 통해 고가용성(High Availability) 및 결함 포용성(Fault Tolerance)을 제공하여 장애 대응력을 지원한다[11].

최근 HPC 뿐만 아니라 AI 기반 ML/DL 연구가 활발해지면서 다양한 I/O 패턴과 대규모의 파일 처리를 위한 요구사항이 늘어나고 있다. NVMe SSD와 같은 비휘발성의 플래시메모리 기반의 미디어를 이용한 고성능 스토리지 및 병렬 I/O 성능을 향상시키기 위한 파일시스템의 새로운 기술들이 적용되고 있다.

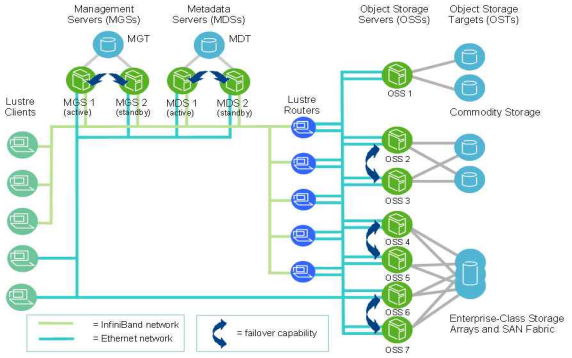


그림 2. Lustre 개요
Fig 2. Lustre overview

1) DNE (Distributed NamEspace)

Lustre에서 데이터의 메타 정보는 중앙화된 메타데이터 서버(MDS: MetaData Server)를 통해 관리된다. DNE는 분산 파일 시스템에서 파일 및 디렉터리의 이름 공간을 관리하는 기술로 그림 2와 같이 복수 개의 MDS를 설정할 수 있으며 Lustre 파일 시스템에서는 DNE를 통해 파일과 디렉터리의 이름 공간을 각각 관리한다. 복수 개의 MDS를 사용하는 Lustre 파일 시스템에서는 다양한 이점을 갖는다. 첫째, 분산된 메타데이터는 파일 시스템의 확장성을 향상시킨다. 여러 MDS가 동시에 작업을 처리하므로 시스템의 처리량(IOPS)이 증가하고 병목 현상이 감소한다. 둘째, 복제된 메타데이터를 통해 고가용성(HA: High Availability)을 제공한다. 하나의 MDS가 고장 나더라도 다른 MDS가 해당파일 및 디렉터리의 메타데이터를 처리할 수 있어 시스템의 가용성을 보장할 수 있다. 셋째, 복수 개의 MDS는 메타데이터의 부하를 분산시켜 시스템의 성능을 향상시킨다. 복수 개의 MDS를 사용하는 경우 메타데이터 정보를 동기화하는 메커니즘이 필요하다. 이를 위해 Lustre는 메타데이터 간 동기화를 위한 특정 프로토콜과 알고리즘을 사용하여 MDS 간에 메타데이터 변경 사항을 상호 동기화하여 일관성을 유지한다.

2) Progressive File Layout (PFL)

Lustre 파일시스템은 스트라이핑(Striping)을 통해 하나의 파일을 물리적으로 분산된 여러 디스크(OST: Object Storage Target)에 분할하여 저장함으로써 병목 현상을 제거 하고 I/O 성능 향상시킬 수 있다. PFL은 기존에 파일을 고정으로 분할하여 저장하는 방식이 아니라 파일의 크기에 따라 복합 레이아웃을 설정하여 동적으로 분할 개수(Stripe count)와 크기(Stripe size)를 조정한다.

3) Data on Metdata (DoM)

DoM은 Lustre에서 OST에 분산된 작은 파일 검색 및 무작위 I/O(Small random file I/O)의 부하를 개선하기 위한 기법이다. 작은 파일은 MDT에 직접 배치함으로써 실제 OST에 발생하는 부하를 줄여 전체 파일시스템의 성능향상을 가

져온다. 위에 언급한 PFL의 복합 레이아웃 설정을 통해 MDT 블록 영역을 지정하고 파일의 크기를 설정하면 해당되는 작은 파일은 MDT에 직접 저장된다. 나머지 크기의 파일들은 PFL에서 지정한 레이아웃에 맞게 OST에 저장된다. 이로써 작은 파일에 대해 더 빠르게 액세스가 가능하다[12]. 그림 3은 Lustre DoM 설정을 보여주는 그림으로 레이아웃의 첫 번째 컴포넌트인 MDT 영역에는 1M 이하 사이즈의 작은 데이터가 저장되며 나머지 파일 크기의 데이터는 OST 영역에 저장된다.

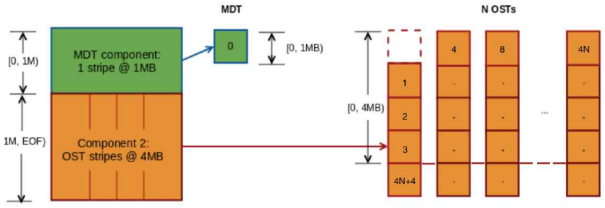


그림 3. DoM의 파일 배치 <Lustre S/W 2.x Manual>
Fig. 3. DoM file layout

3-2 파일 시스템 성능 벤치마크

1) IOR 벤치마크

IOR은 병렬 파일 시스템의 성능을 측정하는 벤치마크 도구로 POSIX, MPIIO, HDF5와 같은 다양한 인터페이스와 액세스 패턴을 이용하여 스토리지의 대역폭, 지연, I/O 처리량 등을 측정할 수 있다. MPI 라이브러리를 사용 병렬 파일 I/O를 수행하여 여러 노드 또는 프로세서에서 동시에 파일을 읽고 쓸 수 있다. MPI(Message Passing Interface)는 병렬 컴퓨팅에서 메시지 패싱을 지원하기 위한 표준 프로토콜 및 라이브러리이다[13]. 테스트는 각 프로세스가 단일파일을 생성(Single-Shared File)하거나 또는 각 프로세스마다 각각 파일을 생성하는 방식(File per Process)으로 수행된다[14].

2) MDTEST

MDTEST는 MPI를 활용하여 메타데이터의 성능을 측정하는 도구로 파일시스템 Open/Stat/Close 동작을 측정하게 된다[8]. HPC 환경에서는 대량의 계산노드들이 공용의 파일시스템을 마운트 하여 사용하게 된다. 여러 사용자들이 동시에 공유 파일시스템에 접속해서 파일들을 생성하고 삭제하는 오퍼레이션을 수행하게 되는데, 이에 대해 파일시스템의 성능 부하를 사전에 테스트할 수 있다. 빌드를 위해서 C 컴파일러와 MPI 라이브러리가 필요하며, 실행 시에는 프로세스에서 실행될 파일의 개수, 공용 또는 개별 디렉터리 선택, 테스트 수행 옵션(Creat, Stat, Remove)을 설정해서 각각의 메타데이터 성능을 검증한다.

위와 같은 벤치마크 수행을 통해 사용자 애플리케이션 특성에 따라 요구되는 I/O 워크로드를 측정할 수 있으며 최적의 작업수행 환경을 설계할 수 있다[15].

Ⅳ. 시스템 환경

본 연구는 HPC의 고성능 스토리지 파일시스템에 대규모의 고성능 I/O 관련 기술을 적용하고 BMT 도구를 활용하여 데이터 특성에 따른 다양한 I/O 패턴 성능을 분석한다. 이를 위해 고 대역폭의 인터커넥트를 통해 구성된 클러스터 시스템과 고성능의 병렬파일시스템을 구축하였고 최근 NVMe 기반의 플래시 디스크가 적용되면서 기존 SATA, SAS 디스크 간의 데이터 계층형 스토리지(Tiering Storage) 환경을 구성하고 자동 적재 기술을 적용하여 기능 및 성능 시험을 수행한다. 이 실험은 기존 소규모의 테스트 환경과 달리 고성능 컴퓨팅 환경(HPC)에서 파일시스템의 I/O 성능 향상을 위한 다양한 기법들을 실제 적용했을 때의 실제 성능의 변화와 이점을 살펴본다. 이는 최근 AI 기반의 연구 확산으로 산학연 여러 곳에 구축되고 있는 대규모 클러스터 환경의 고성능 스토리지 구축 사례로 참조될 수 있다.

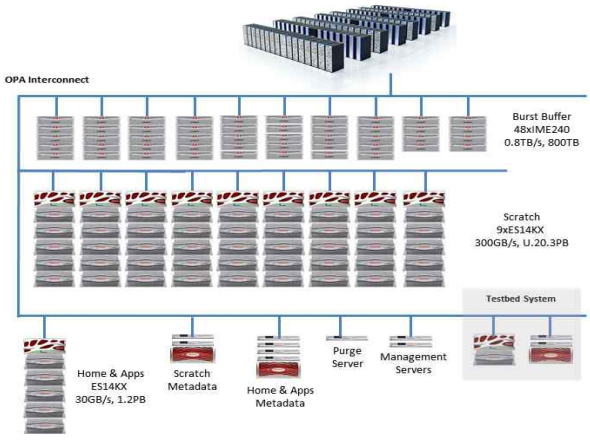


그림 4. 슈퍼컴퓨터 5호기 누리온 스토리지 구성
Fig. 4. 5th supercomputer NURION storage configuration

4-1 버스트버퍼(Burst Buffer)

고성능 컴퓨팅(High-Performance Computing, HPC) 및 대규모 데이터 처리 환경에서 사용되는 기술로, 데이터 스토리지와 컴퓨팅 리소스 간의 효율적인 데이터 이동 및 관리를 위해 설계된 시스템이다. 대용량 데이터를 빠르게 읽고 쓰기 위해 높은 처리량 및 낮은 지연 시간을 가진 NVMe 기반의 플래시 디스크 캐시 계층을 제공한다. 그림 4의 KISTI 슈퍼컴퓨터 5호기 누리온에 도입된 버스트버퍼(Busrt Buffer) IME는 계산노드와 스토리지 중간에 I/O 가속을 위해 NVMe 기반으로 구축된 파일시스템이다[16]. 사용자는 기존 병렬파일시스템에서 데이터를 버스트버퍼에 스테이징(Staging)한 후 연산 결과를 캐싱(Cashing)하고 다시 병렬파일시스템으로 플러싱(Flushing)한다.

4-2 자동 적재 계층형 스토리지(Automated Tiering Storage)

계층형 스토리지는 저장 매체에 따라 다양한 스토리지 티어를 구성하고 정책에 따라 데이터를 이주 관리하면서 가용성, 성능 및 비용을 최적화하는 방법이다. 이 기술은 대용량 데이터를 효율적으로 관리하고, 스토리지 비용을 절감하며, 데이터 성능을 향상시키는 데 사용된다. 최근 NVMe (Non-Volatile Memory Express) SSD는 기존의 SATA SSD와 비교하여 더 높은 성능을 제공하며 낮은 지연 시간과 빠른 데이터 읽기/쓰기 속도로, 데이터 액세스 속도를 향상시키고 애플리케이션 성능을 향상시킬 수 있다. 하지만 고가의 저장매체로 All-Flash 기반의 스토리지의 사례가 있으나 구축에 많은 비용이 투입된다. 계층형 스토리지 기술은 서로 다른 NVMe 기반 플래시와 기존 SATA, SAS 기반의 디스크 매체를 계층으로 구성하는 기술이다. 버스트버퍼와 달리 자동 티어링 스토리지 기술(Automated tiering storage)은 데이터의 액세스 패턴을 분석하고, 자주 액세스되는 데이터를 정책에 따라 NVMe SSD와 같은 빠른 스토리지에 배치하여 성능 향상과 비용 절감을 동시에 충족할 수 있다[17].

V. 성능 분석

5-1 시스템 성능 검증

1) 분산 메타데이터 (DNE) 성능 검증

DNE를 테스트를 위해 복수 개의 MDS가 필요하다. 이를 위해 기존 MDS에 메타데이터 정보가 저장되는 단일 Metadata Target (MDT)를 백업 후 두 개의 볼륨으로 나누어 복수 개의 MDT로 그림 5와 같이 구성하였다. Lustre 2.x 버전에서는 최대 256개의 분산 메타데이터를 구성할 수 있다. 하지만 분산 메타데이터의 개수가 많아질수록 항상 성능이 향상되지 않는다. 메타 정보가 여러 곳에 분산될 경우 파일의 정보를 찾아야 하므로 메타데이터 부하가 증가할 수 있기 때문이다. 따라서 스토리지 용량과 파일의 크기에 따라 분산 메타데이터의 설정을 결정해야 한다.

HW 구성 후 Lustre 파일시스템은 DNE를 표 1과 같이 각 디렉터리에 대해 분산 메타데이터 개수를 설정할 수 있다. 두 개의 디렉터리를 생성한 후 하나의 디렉터리는 단일 메타데이터를 또 하나의 디렉터리는 복수 개로 구성하여 대역폭과 메타데이터 성능을 비교하였다.

표 1. DNE 설정에 따른 디렉터리 속성 정의
Table 1. Defining directory properties according to DNE configuration

```
[ DNE configurations ]
# lfs getdirstripe noDNE_cpool/
lmv_stripe_count:0 lmv_stripe_offset:0 lmv_hash_type: none
# lfs mkdir --mdt-count 2 -D ./DNE_hpool/
# lfs getdirstripe DNE_cpool/
lmv_stripe_count:2 lmv_stripe_offset:1 lmv_hash_type: fnv_1a_64
mdtidx                FID[seq:oid:ver]
     1                [0x2400163ca:0x1354a:0x0]
     0                [0x200074b9f:0x13548:0x0]
```

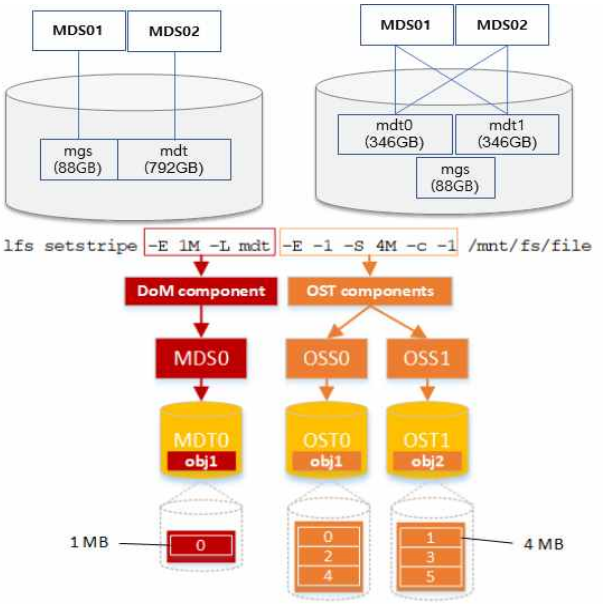


그림 5. DNE 및 DoM 구성에 따른 데이터 분할
Fig. 5. Data striping by DNE and DoM configuration

표 2와 같이 MPI를 이용하여 각 프로세스마다 2GB 파일을 생성하는 IOR과 표 3의 명령어를 통해 1,000개의 파일을 생성하여 메타데이터 성능을 측정하는 MDTEST 벤치마크를 수행하였다. 그림 6과 7에서 보듯이 단일 MDT에 비하여 대역폭(MiB/s)과 메타데이터(IOPS) 성능이 향상됨을 확인할 수 있다.

표 2. IOR 벤치마크 옵션
Table 2. IOR benchmark option

```
# mpirun ior -a MPIIO -b 2G -o DNE_dir -t 1m -w -r -F
```

표 3. MDTEST 벤치마크 옵션
Table 3. MDTEST benchmark option

```
# mpirun mdtest -d DNE_dir -n 1000 -F -T -E -C -r
```

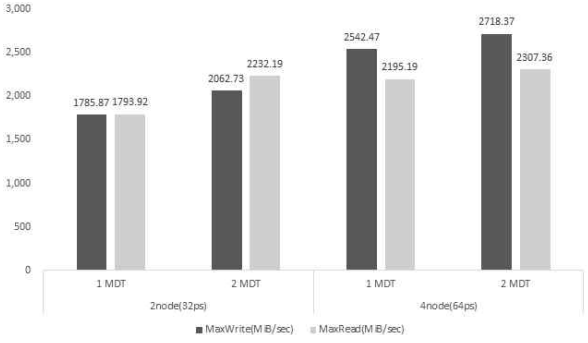


그림 6. DNE 구성에 따른 대역폭 확장성 테스트
Fig. 6. Bandwidth scaling performance according to DNE

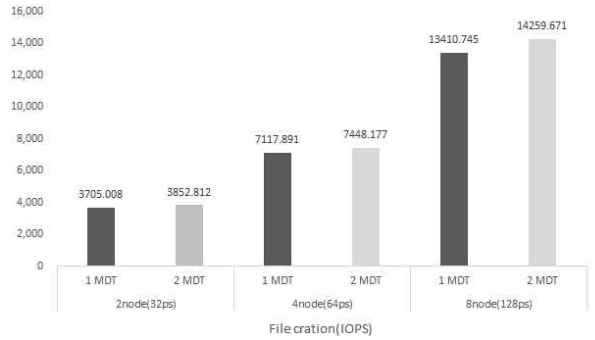


그림 7. DNE 구성에 따른 IOPS 확장성 테스트
Fig. 7. IOPS scaling performance according to DNE

2) PFL 성능 검증

그림 8과 9는 1~4노드(16~64ppn)로 확장하면서 단일 파일을 생성(Single-Shared File)하는 성능 시험을 PFL이 미적용(no_PFL)된 디렉터리와 설정 된 디렉터리에서 각각 수행하였다. 표 4는 PFL에 적용된 분할(striping) 설정이며 여기에서 보듯이 4M부터 파일 사이즈에 따라 2배씩 분할 수를 증가시켰다. PFL 설정으로 분할 개수(Stripe count)가 많아져 데이터가 여러 저장소에 있는 경우 Read 성능에서 각 파일의 인덱스 정보를 수집하는 부하가 더 발생할 수 있다. 따라서 스토리지의 규모와 클라이언트의 수에 따라 적절한 분할 개수를 설정해야 한다.

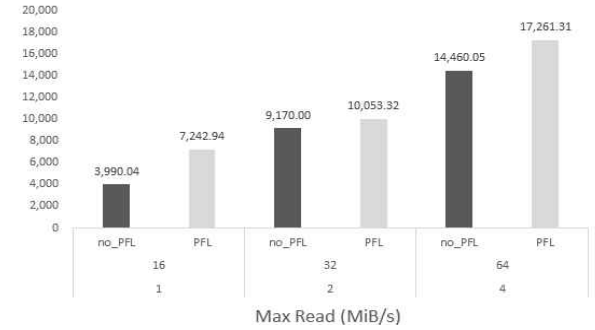


그림 8. PFL 설정에 따른 Read 성능 향상
Fig. 8. Read performance benchmark according to PFL

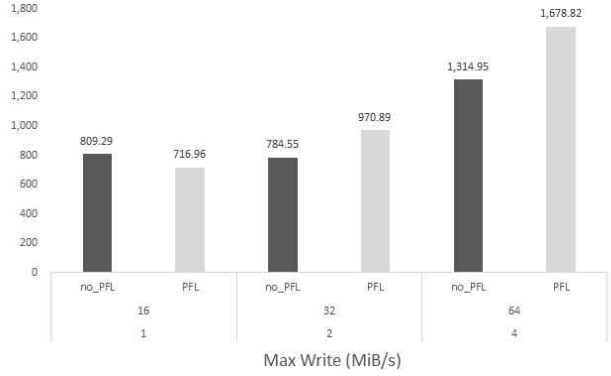


그림 9. PFL 설정에 따른 Write 성능
Fig. 9. Write performance benchmark according to PFL

표 4. PFL의 동적 분할(Striping) 설정
Table 4. Dynamic striping configuration for PFL

File size (FS)	Stripe-count	Stripe-size
FS ≤ 4MB	1	1MB
4MB < FS ≤ 512MB	2	1MB
512MB < FS ≤ 1GB	4	1MB
1G < FS ≤10GB	8	1MB
10G< FS ≤100G	16	1MB
FS > 100G	32	1MB

3) Data on Metdata (DoM) 성능 검증

Lustre는 상대적으로 작은 파일의 I/O 처리 성능에 취약하다. DoM은 메타데이터에 작은 파일을 직접 저장하여 I/O 성능을 향상시킬 수 있는 기법이다. 표 5와 같이 메타데이터 타깃(MDT)과 파일사이즈를 지정하여 DoM을 설정한다.

표 6은 DoM 성능 측정을 위한 IOR 벤치마크 명령어로 표 5의 설정에서와 같이 data_dom 디렉터리에 사이즈가 64KB 이하의 파일들은 메타데이터에 저장되어 I/O가 발생한다. 그림 10의 실험은 블록사이즈 1KB와 1,000KB 사이즈의 파일들을 64개의 프로세서에서 생성하여 수행한다. 상대적으로 작은 블록사이즈(1KB)의 파일들이 많은 경우 메타데이터에서 처리되어 더 높은 성능을 얻게 된다.

표 5. DoM 설정 및 디렉터리 속성
Table 5. DoM configuration and directory properties

```
[ DoM Configuration ]
$lfs setstripe -E 64K -L mdt
  -E -1 —stripe-count=4 —stripe-size=1M  IOR_dom
$ lfs getstripe -d IOR_dom    // DoM is set
stripe_count:0 stripe_size:65536 pattern: mdt stripe_offset:-1
stripe_count:0 stripe_size:1048576 pattern: raid0 stripe_offset:-1
$ lfs getstripe -d IOR_data   // DoM not set
stripe_count:1 stripe_size:1048576 pattern:0 stripe_offset: -1
```

표 6. IOR 벤치마크 수행 옵션
Table 6. IOR benchmark options and execution

```
# mpirun ior -a MPIIO -o data_dom -b [1,1000k] -t
  [1k,100k] -d 10 -C -Q 25 -e -w -r -k -F -i 1
```

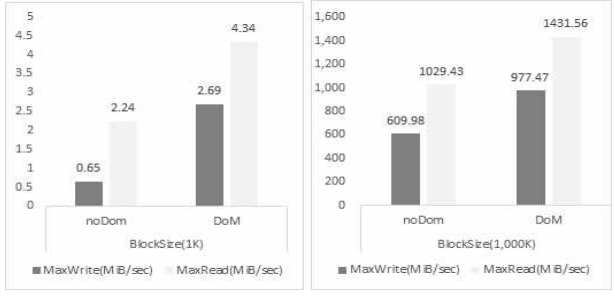


그림 10. DoM 구성에 따른 대역폭 성능 비교
Fig. 10. Bandwidth performance comparison according to DoM configuration

4) 자동 적재 계층형 스토리지 기능 및 성능 검증

계층형 스토리지는 저장 매체에 따라 저장소를 분류하고 각 계층에 대한 액세스 패턴과 요구 사항에 따라 데이터를 배치하여 비용 효율적인 방식으로 파일을 관리한다. Hot/Cool 영역은 각 저장소 계층을 이르며 Hot 영역에는 빠른 응답 시간과 고성능의 대역폭을 제공하는 NVMe SSD와 같은 매체를 사용하여 빈번하게 액세스되거나 작은 용량의 데이터를 저장한다. 하지만 높은 매체 비용으로 잦은 I/O가 발생하는 데이터를 대상으로 저장하며 일정 기간 동안 액세스가 발생하지 않으면 비용이 낮은 Cool 영역에 데이터를 복사 후 공간을 다시 확보하게 된다. Cool 영역은 상대적으로 액세스 빈도가 낮은 데이터를 저장하게 되며 보관 기간이 더 길고 데이터 액세스가 더 느릴 수 있지만, 비용이 상대적으로 낮다. 일반적으로 저장소 비용을 절감하면서 데이터를 안전하게 보관하는데 사용된다.

본 실험을 위해 슈퍼컴퓨터 5호기 누리온 Burst Buffer의 NVMe SSD를 장착 노드를 Hot 영역으로 기존 NSAS 기반의 디스크는 Cool 영역으로 그림 11과 같이 계층형 스토리지를 재구성하였다.

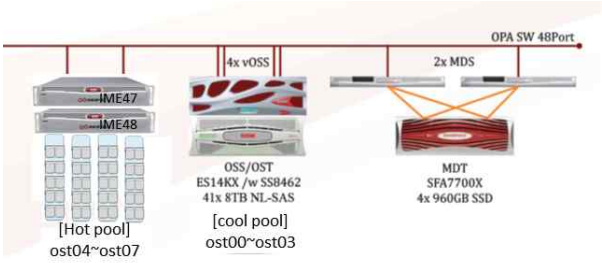


그림 11. 누리온 Burst Buffer를 활용한 계층형 스토리지 구성
Fig. 11. Tiered storage configuration using Nurion burst buffer

표 7. 계층형 스토리지의 자동 적재 설정 옵션
Table 7. Auto staging configuration for tiering storage

```
# lfs setstripe -E 1G -c 1 -p ddn_ssd --comp-flags=prefer
  -E 16G -c 4 -E eof -c 40 -p ddn_hdd /scratch
```

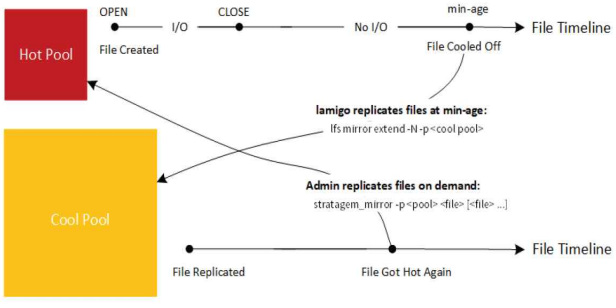


그림 12. 누리온 Burst Buffer를 활용한 계층형 스토리지 구성
Fig. 12. Tiered storage configuration using Nurion burst buffer

Hot/Cool 영역의 자동화된 데이터 관리(Automated Storage Tiering)를 위해 DDN의 Lustre 어플라이언스로 제공되는 EXAScaler 소프트웨어를 설치하였다. 계층형 스토리지 간의 자동 적재를 위해 EXAScaler에서는 Hot에서 Cool 영역으로 데이터를 미러링하는 과정과 조건에 따라 Hot 영역에 용량 조건에 따라 퍼지하는 기능을 가지고 있다. 표 7과 같이 파일의 사이즈 범위에 따라 지정된 영역에 저장되며 플래그 옵션을 통해 Hot 영역에 저장된 데이터는 동일하게 Cool 영역으로 미러링 되고 일정시간이 지나면 Hot 영역에서 제거된다. 읽기시에는 Hot 영역에 데이터가 있는 경우 바로 가져오지만 제거된 파일의 경우 Cool 영역에서 데이터를 가져오고 Hot 영역으로 다시 적재시킨다. 그림 12는 Hot/Cool 영역의 데이터 관리 흐름을 보여주며 Hot 영역에 저장된 데이터는 Cool 영역에 동일하게 미러링 및 동기화가 발생하며 일정 시간동안 사용되지 않으면 Hot 영역에서 제거된다. 이 후 해당 파일이 다시 사용되면 Hot 영역에 자동으로 적재(staging) 되어 해당 영역에서 I/O가 발생한다.

그림 13은 Hot/Cool 영역에서 모든 프로세서가 단일파일을 생성(SSF, Single-Shared File)하거나 또는 각 프로세서가 각각 파일을 생성(FPP, File per Process)하는 방식에서 읽기 성능을 네 가지 방식으로 측정하였다. 첫 번째와 두 번째는 Hot/Cool 각 영역에서의 읽기 대역폭의 차이를 보여준다. 세 번째는 플래시기반의 NVMe SSD로 구성된 Hot 영역과 N-SAS 기반의 Cool 영역으로 구성된 자동 적재 환경에서 발생하는 I/O 패턴에 따라 일정 시간 후 Hot 영역에 데이터가 제거되었을 때의 성능을 보여준다. 이 경우(Clear Hot-Pool)는 Cool 영역과 동일 성능을 보이며, 이후 다시 데이터를 읽을 때는 Hot 영역으로 데이터가 자동 적재되어 되어 네 번째 Hot Pool (Auto Staging)과 같이 Hot 영역에서 읽기를 성능을 보이게 된다. 즉, Hot 영역에서 제거된 데이터를 다시 읽을 때는 처음 Cool 영역에서 가져온 후 Hot 영역으로 적재된다.

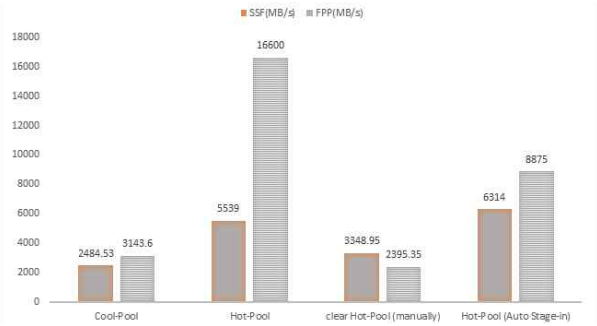


그림 13. Hot/Cool 및 스테이징 인/아웃을 통한 SSF, FPP 읽기 성능
Fig. 13. SSF, FFP read performance with Hot/Cool and staging In/Out

Ⅵ. 결 론

HPC 환경에 AI가 접목되면서 고성능의 이기종 가속기 기반 시스템 도입이 일반화되고 있다. 즉, 계산 자원의 성능 집약도가 높아짐에 따라 스토리지의 I/O 성능과 다양한 패턴들에 대한 요구사항이 동시에 급격히 증가하고 있다. 관련하여 계층형 자동적재 스토리지 기술(Automated Storage Tiering), 메타데이터의 IOPS 성능 향상을 위한 분산 스토리지 기술, 네트워크(RDMA, RoCE 등) 분야, GPU 직접통신 기술(GPU-Direct Storage) 등과 같은 I/O 성능향상 기술과 NVMe (Non-Volatile Memory Express)와 PMem (Persistent Memory) 등과 같은 매체들이 스토리지에 적용되어 저지연 시간, 높은 처리량, 병렬 액세스를 지원하는 고성능 스토리지 환경을 제공한다. 이렇게 다양한 분야에서 높은 I/O 처리량에 대한 수요를 극복하기 위해 지속해서 새로운 연구와 기술이 제시되고 있다.

본 논문은 고성능 스토리지 환경에 적용된 PFL, DoM, DNE 등과 같은 병렬 I/O 성능 향상 기술을 HW부터 설계하고 구축하여 각 기법들에 대한 성능 검증을 수행하였다. 또한 최근 지속적으로 적용되고 있는 계층형 자동적재 스토리지 기술(Automated Storage Tiering)을 실제 구축하여 관련 SW를 설치하고 각 기능과 성능을 확인하였다. 본 연구는 스토리지 시스템을 구축, 설계하는 인프라 연구자와 실제 애플리케이션을 수행하는 연구자에게 HW, SW 관련 기술 및 I/O 패턴에 대한 고려사항을 종합적으로 제시할 수 있고 성능 개선에 대한 예측을 할 수 있게 한다.

향후, 본 실험에 적용된 환경을 확장 구성하여 스케일링에 따른 성능 비교와 실제 HPC, AI의 현업 애플리케이션을 수행하면서 I/O 패턴에 대한 특성을 분류하여 동적으로 최적의 파일시스템 환경을 제공할 수 있는 연구를 진행하고자 한다. 이런 사례와 성능결과는 고성능 스토리지 구축 사례로 필요한 지침이 될 수 있다.

참고문헌

[1] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, “Survey and Benchmarking of Machine Learning Accelerators,” in *Proceedings of IEEE high performance extreme computing conference (HPEC),* Waltham: MA, pp. 1-9, September 2019. https://doi.org/10.1109/HPEC.2019.8916327

[2] D. Milojicic, P. Faraboschi, N. Dube, and D. Roweth, “Future of HPC: Diversifying Heterogeneity,” in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, pp. 276-281, February 2021. https://doi.org/10.23919/DATE51398.2021.9474063

[3] H. N. Khan, D. A. Hounshell, and E. R. H. Fuchs, “Science and Research Policy at the End of Moore’s Law,” *Nature Electronics,* Vol. 1, pp. 14-21, January 2018. https://doi.org/10.1038/s41928-017-0005-9

[4] M. A. Dávila Guzmán, R. Nozal, R. Gran Tejero, M. Villarroya-Gaudó, D. Suárez Gracia, and J. L. Bosque, “Cooperative CPU, GPU, and FPGA heterogeneous execution with EngineCL,” *The Journal of Supercomputing*, Vol. 75, No. 3, pp. 1732-1746, March 2019. https://doi.org/10.1007/s11227-019-02768-y

[5] Top500. Lists: JUNE 2023 [Internet]. Available: http://top500.org/.

[6] Y. Roh, G. Heo, and S. E. Whang, “A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 33, No. 4, pp. 1328-1347, April 2021. https://doi.org/10.1109/TKDE.2019.2946162

[7] A. Elhassouny and F. Smarandache, “Trends in Deep Convolutional Neural Networks Architectures: A Review,” in *Proceedings of International Conference of Computer Science and Renewable Energies (ICCSRE)*, Agadir, Morocco, pp. 1-8, July 2019. https://doi.org/10.1109/ICCSRE.2019.8807741

[8] K. Fujita, T. Yamaguchi, Y. Kikuchi, T. Ichimura, M. Hori, and L. Maddegedara, “Calculation of Cross-Correlation Function Accelerated by TensorFloat-32 Tensor Core Operations on NVIDIA’s Ampere and Hopper GPUs,” *Journal of Computational Science*, Vol. 68, 101986, April 2023. https://doi.org/10.1016/j.jocs.2023.101986

[9] J. Liu, J. Wu, S. P. Kini, D. Buntinas, W. Yu, B. Chandrasekaran, ... and D. K. Panda, MPI over InfiniBand: Early Experiences, The Ohio State University, Columbus: OH, OSU-CISRC-10/02-TR25, 2003.

[10] M. S. Birrittella, M. Debbage, R. Huggahalli, J. Kunz, T. Lovett, T. Rimmer, ... and R. C. Zak, “Intel® Omni-Path Architecture: Enabling Scalable, High Performance Fabrics,” in *Proceedings of the 23rd Annual Symposium on High-Performance Interconnects*, Santa Clara: CA, pp. 1-9, August 2015. https://doi.org/10.1109/HOTI.2015.22

[11] A. George, R. Mohr, J. Simmons, and S. Oral, Understanding Lustre Filesystem Internals - Second Edition, Oak Ridge National Laboratory, Oak Ridge: TN, ORNL/TM-2021/2212, September 2021.

[12] J. Fragalla, B. Loewe, and T. K. Petersen, “New Lustre Features to Improve Lustre Metadata and Small-File Performance,” *Concurrency and Computation: Practice and Experience,* Vol. 32, No. 20, e5649, October 2020.

https://doi.org/10.1002/cpe.5649

[13] B. Barker, “Message Passing Interface (MPI),” *Workshop: High Performance Computing on Stampede*, Vol. 262, January 2015.

[14] H. Shan and J. Shalf, “Using IOR to Analyze the I/O Performance for HPC Platforms,” in *Proceedings of the 49th Cray User Group Meeting (CUG 2007)*, Seattle: WA, May 2007.

[15] P. Luszczek, J. J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, ... and D. Takahashi, Introduction to the HPC Challenge Benchmark Suite, Lawrence Berkeley National Laboratory, Berkeley: CA, LBNL-57493, March 2005.

[16] The KISTI National Supercomputing Center [Internet]. Available: http://www.ksc.re.kr/.

[17] H. Herodotou and E. Kakoulli, “Automating Distributed Tiered Storage Management in Cluster Computing,” arXiv: 1907.02394, July 2019. https://doi.org/10.48550/arXiv.1907.02394

윤준원(Junweon Yoon)

2004년 : 고려대학교 대학원 컴퓨터학과 (이학석사)

2018년 : 고려대학교 대학원 컴퓨터학과 (공학박사)

2005년 ~ 현 재: KISTI 국가슈퍼컴퓨팅본부 책임연구원

※관심분야 : 분산컴퓨팅, 슈퍼컴퓨팅, 인공지능(ML/DL), 병렬파일시스템, 배치스케줄링, 벤치마크(BMT) 등

송의성(Ui-Sung Song)

1997년 : 고려대학교 컴퓨터학과(학사)

1999년 : 고려대학교 대학원 컴퓨터학과 (이학석사)

2005년 : 고려대학교 대학원 컴퓨터학과 (이학박사)

2006년 ~ 현 재: 부산교육대학교 컴퓨터교육과 교수

※관심분야 : 컴퓨터교육, 교육용로봇교육, 컴퓨터네트워크, 스마트러닝 등