Journal of Digital Contents Society Vol. 17 No. 5 Oct. 2016(pp. 409-416)
http://dx.doi.org/10.9728/dcs.2016.17.5.409

# 병렬 분산파일시스템의 성능 분석을 통한 최적화 연구

윤준원*, 송의성**

## 요 약

최근 빅데이터 이슈가 화두가 됨에 따라 대학, 산업체, 연구소 등에서는 다양한 데이터들을 수집, 분석 하려는 노력이 활성화 되고 있다. 여기에는 과거부터 축적된 데이터, 현재에 바로 분석이 불가능하더라도 잠재적인 의미를 가지고 있는 데이터 등 대량의 데이터들이 수집되어 의미론적인 분석을 통해 가치 있는 분석결과를 얻게 된다. 이를 위해 전 세계적으로 대용량의 데이터 요구를 처리 할 수 있는 고성능 스토리지 시스템의 수요가 증가하고 있다. 또한, 여러 사용자들에게 축적된 대량의 데이터에 동시에 접속하여 다양한 분석을 수행할 수 있도록 안정성 있는 병렬 분산파일시스템을 제공해야 한다.

본 연구에서는 위와 같이 안정성 있는 파일시스템을 제공하기 위해 반드시 고려되어야 할 스토리지 시스템의 I/O 대역폭, 메타데이터의 성능 등을 파악하고 최적의 환경을 구성하기 위한 방법을 제시하고자 한다.

키워드 : 파일시스템, 빅데이터, 스토리지, 벤치마크, 성능분석

# Study of Optimization through Performance Analysis of Parallel Distributed Filesystem

JunWeon Yoon*, Ui-Sung Song**

## Abstract

Recently, Big Data issue has become a buzzword and universities, industries and research institutes have been efforts to collect, analyze various data enabled. These things includes accumulated data from the past, even if it is not possible to analysis at this present immediately a which has the potential means. And we are obtained a valuable result from the collected a large amount of data via the semantic analysis. The demand for high-performance storage system that can handle large amounts of data required is increasing around the world. In addition, it must provide a distributed parallel file system that stability to multiple users too perform a variety of analyzes at the same time by connecting a large amount of the accumulated data

In this study, we identify the I/O bandwidth of the storage system to be considered, and performance of the metadata in order to provide a file system in stability and propose a method for configuring the optimal environment.

Keywords : Filesystem, Big Data, Storage, Benchmark, Performance Evaluation

## 1. 서론

최근 산학연에서 기하급수적으로 늘어나는 통계데이터, 멀티미디어, 디지털 콘텐츠 등의 수요를 처리하기 위해 다양한 고성능 스토리지 시스템의 수요가 증가하고 있다. 이는 빅데이터 처리를 위한 필수적인 요소로 자리매김 하고 있으며 기존의 SAN(Storage Area Network) 스토리지의 경우 단일 스토리지 시스템의 I/O 한계와 파일시스템 용량 및 파일 공유 제한으로 인해 한계가 있다. NAS(Network Attached Storage)

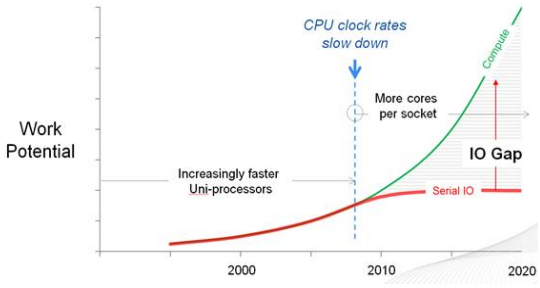
※ Corresponding Author: Ui-Sung Song
Received : October 10, 2016
Revised : October 25, 2016
Accepted : October 31, 2016
* Dept. of Supercomputing Center, KISTI
Tel: +82-42-869-0581, Fax: +82-42-869-0569
email: jwyoon@kisti.re.kr
** Dept. of Computer Education, Busan National University of Education / ussong@bnue.ac.kr

스토리지 또한 한정된 I/O 대역폭으로 고성능 구현이 어려우며 시스템 장애에 대한 안정성을 보장하기 어렵다. 이러한 문제점을 해결하기 위해 분산파일시스템이 시장에 본격적으로 정착되고 있는 실정이다[1].

(그림 1)은 시스템 성능에 따른 파일시스템의 I/O 대역폭에 대한 요구사항으로, 최근 무어의 법칙(Moore's Law)의 예측과 달리 시스템의 성능이 일관성 있게 증가하지 않고 있으며, 그에 대비해 데이터 규모가 커짐에 따라 I/O 요구사항의 격차가 증가하고 있는 추세이다[2].

본 연구에서는 클러스터 규모의 분산파일시스템을 대상으로 오픈소스 벤치마크 도구를 통해 I/O 성능, 메타데이터 성능, 병렬파일시스템을 구성하는 인터커넥터의 성능 등을 측정하여 최적의 환경을 구성하는 연구를 수행하고자 한다.

(그림 1) 시스템 성능 증가와 파일시스템 I/O 격차




(Figure 1) Gap between Increase System Performance and Filesystem I/O (Source: Why Parallel I/O Software and Moore's Law Enable Virtualization and Software-Defined Data Centers to Achieve their Potential)

## 2. 관련 연구

분산 파일시스템은 기하급수적으로 증가하는 데이터를 저장하고 관리하는 것을 목적으로 하며 이를 구축하기 위해서는 거대한 IT 인프라가 요구된다. 대규모의 스토리지를 도입하여 운용하는 사이트에서는 병렬파일시스템 적용 후 사용자에게 서비스하기 위해서는 I/O 패턴에 따른 워크로드 분석이 반드시 필요하다.

본 논문에서는 Lustre 파일시스템으로 구축된 시스템 환경에서 오픈소스 기반의 I/O 벤치마크 도구인 IOR과 MDTEST를 이용하여 워크로드를 분석한다.

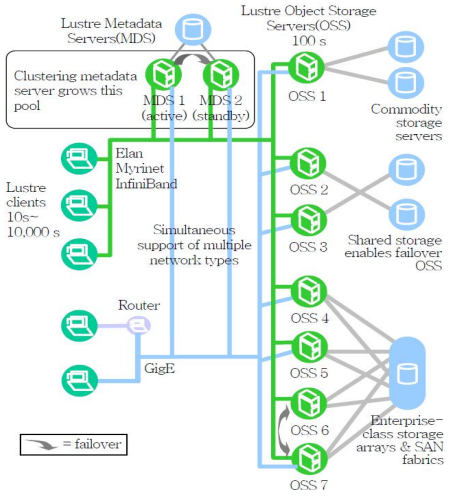
① Lustre 파일시스템

Lustre는 분산 병렬파일시스템으로 주로 HPC 분야의 대용량 파일시스템으로 사용되고 있다[3]. 또한 오픈소스 기반의 고성능 클러스터 파일시스템으로 기존의 NFS와 같은 분산 파일시스템에 객체 기반의 저장(Object-based Storage) 디스크 기술을 적용함으로써 성능, 가용성, 확장성의 문제를 개선하였다. Lustre는 대규모 클러스터 환경을 지원할 수 있는 객체 기반의 클러스터 파일시스템으로 (그림 2)와 같이 클라이언트 파일시스템, 메타데이터 서버, 객체 저장 서버들로 구성되며 각각은 고속 네트워크로 연결된다.

Lustre에서는 계층화된 모듈 구조로 TCP/IP, Infiniband, Myrinet, Quadrics Elan3, Quadrics Elan4 등과 같은 네트워크를 지원한다. 클라이언트 파일시스템은 리눅스 VFS에서 설치할 수 있는 파일시스템으로 메타데이터 서버와 객체 저장 서버들과 통신하면서 클라이언트 응용에 파일시스템 인터페이스를 제공한다. 메타데이터 서버는 파일시스템의 이름공간(Name Space)과 파일에 대한 메타데이터를 관리한다. 그리고 객체 저장 서버는 파일 데이터를 저장하고 클라이언트로부터의 객체 입출력 요청을 처리한다. 객체는 객체 저장 서버에 각각 스트라이핑(striping)되어 분산 저장된다.

Lustre는 병렬로 처리하여 여러 노드에서 동시에 하나의 파일을 읽고 쓰는 작업을 수행할 수 있으며, 응용 프로그램이 I/O 성능을 최대한 사용할 수 있도록 만든다. 또한, 여러 대의 스토리지를 하나의 파일시스템으로 구성하며, 하나의 파일을 여러 스토리지에 분산 및 병렬로 저장이 가능하다. Lustre는 고가용성 및 복원력을 가지고 있으며, 이것은 대규모의 HPC 환경을 위한 고가용성 및 장애 대응 구성을 지원한다[4].

(그림 2) Lustre 구성도




(Figure 2) Lustre Overview

② IOR 벤치마크

IOR은 병렬파일시스템의 I/O 성능을 측정하는 도구로 POSIX, MPIIO, HDF5와 같은 다양한 인터페이스와 액세스 패턴을 이용하여 병렬 파일시스템의 성능을 테스트한다[5]. MPI[6]를 사용하여 구성된 계산자원에서 파일시스템의 Read, Write 성능을 측정하게 되는데 테스트는 각 프로세스가 단일파일을 생성(Single-Shared File)하거나 또는 각 프로세스마다 각각 파일을 생성하는 방식(File per Process)으로 수행된다.

에너지연구국립계산과학센터(NERSC), 오크리지국립연구소(ORNL), 에너지국(DOE), 국립병렬컴퓨터공학연구센터(NRCPC) 등 대다수 HPC 운영사이트에서는 병렬파일시스템의 최적의 서비스 환경을 위해 다양한 워크로드 분석을 필수적으로 수행하고 있다. 이 결과를 활용하여 시스템 환경을 구축하고 설정하게 되는데, IOR은 대표적인 I/O 벤치마크 도구로 사용된다. 벤치마크 결과를 통해 사용자 어플리케이션 특성에 따라 요구되는 I/O 워크로드를 설정할 수 있으며, 최적의 작업수행 환경을 제시할 수 있다. 가령, <표 1>은 NERSC의 Bassi 시스템에서 단일파일과 공용파일에서의 I/O 성능을 측정한 값이다[7]. Bassi 시스템은 122노드로 구성된 Power5 (1.9GHz) 기반의 시스템으로 최대 I/O 대역폭 이론성능은 6.4GB/s이다. 이 외에 전송사이즈, 블록사이즈, 병렬수행에 따른 I/O 성능을 측정함으로서 사용자 어플리케이션을 최적화하기 위한 요구사항을 얻을 수 있다.

<표 1> NERSC Bassi 시스템에서 IOR 결과

| | Read Unique | Read Shared | Write Unique | Write Shared |
|---|---|---|---|---|
| IOR (MB/s) | 2592 | 2591 | 3404 | 3222 |

<Table 1> The Measured IOR results in NERSC Bassi system

③MDTEST

MDTEST는 MPI를 활용하여 디스크에 파일을 쓰고 지우는 메타데이터 동작에 대한 시스템 성능을 측정하는 도구로 파일시스템 Open/Stat /Close 동작을 측정하게 된다[8]. HPC 환경에서는 대량의 계산노드들이 공용의 파일시스템을 마운트 하여 사용하게 된다. 여러 사용자들이 동시에 글로벌 파일시스템에 접속해서 파일들을 생성하고 삭제하는 오퍼레이션을 수행하게 되는데, 이에 대해 파일시스템의 성능부하를 사전에 테스트할 수 있다. 빌드를 위해서 C 컴파일러와 MPI 라이브러리가 필요하며, 실행시에는 프로세스에서 실행될 파일의 개수, 공용 또는 개별 디렉터리 선택, 테스트 수행 옵션(Creat, Stat, Remove)을 설정해야 한다.

## 3. 시스템 구성

본 연구를 위해 우선 <표 2>과 같이 클러스터와 파일시스템의 구성을 선정하였다. KISTI 슈퍼컴퓨터 Tachyon은 SUN Blade 6275 기반으로 구성된 초병렬 컴퓨팅 시스템으로 이론최고성능(Rpeak) 300TFlops를 보이고 있으며, 3,200대의 컴퓨팅 노드와 인프라 노드(Provisioning, Scheduler, Storage, Interconnector, Archiver 등)로 구분된다. (그림 3)은 Tachyon 시스템의 개요도를 나타내고 있다. Tachyon은 RedHat 5.3기반의 리눅스로 운영되고 있으며, 인터커넥션으로는 인피니밴드(Infiniband) QDR 네트워크의 2-layer, Fat-Tree 토폴로지로 구성이 되어 있다. IB는 계산노드와 병렬파일시스템의 I/O로 사용하고 있으며, 병렬 파일시스템으로는 오픈소스 Lustre가 설치되어 있다.

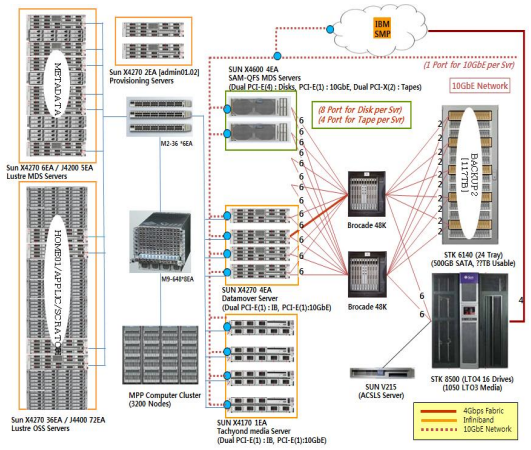
<표 2> 시스템 개요

| Scope | Specification | 비 고 |
|---|---|---|
| Architecture | Cluster | Blade Type |
| Processor | Intel Xeon X5570 2.93GHz(Nehalem) | System Bus: Intel QPI Speed (6.4GT/s) L1/L2/L3:128k/1M/8MB |
| Memory | DDR3/1333MHz | 24GB/node, 3GB/core |
| Disk Storage | DDN SFA12K-40/SS8460 | 2,520TB(after RAID) |
| Interconnection Network | Infiniband 40G 4X QDR | Sun Datacenter 648 Switch |
| OS | RedHat Enterprise Linux 5.3 | Kernel 2.6.18-194.17.1 |
| Filesystem | Lustre 1.8.6(client) /Lustre 2.4.2(server) | /home01, /scratch2 |

<Table 2> System Overview

병렬파일시스템을 구성하고 있는 스토리지는 DDN SFA12K-40 모델로 10개의 Enclosure SS8460(Disk Box)와 10대의 ORACLE X3-2L 서버(MDS 2대, OSS 8대)로 2.4PB의 글로벌 스크래치 디렉터리를 제공하고 있다. 대역폭은 이론성능 약 25.6GB/s이다.

(그림 3) Tachyon 시스템 개요도




(Figure 3) Tachyon System Overview

# 4. 병렬파일시스템 성능분석

본 논문에서는 병렬파일시스템의 I/O 성능을 측정하기위해 IOR과 MDTEST 벤치마크 도구를 사용하였다. 실험에서는 Tachyon 시스템의 I/O 성능을 다양한 패턴을 이용하여 테스트를 수행하였고, 각 실험마다 스토리지와 계산노드의 입력 값을 변경하면서 최적의 수행환경을 도출하였다. 계산노드의 캐시성능, 인터커넥션 네트워크인 인피니밴드의 대역폭, 스토리지 서브시스템의 성능 한계로 읽기와 쓰기 성능은 선형적으로 증가할 수 없다. 따라서 파일사이즈, 전송사이즈, 병렬수행을 위한 프로세스 개수 등 다양한 값을 이용하여 테스트를 수행하고 최적 성능환경을 확인할 수 있다.

## 4.1 캐시 성능 테스트

캐시 기능은 매번 디스크에서 데이터를 읽어오는 것이 아니라 메모리 버퍼에 캐싱함으로서 파일시스템의 읽기 성능을 향상시킬 수 있다. 정해진 크기의 메모리의 버퍼에 데이터를 저장하여 매번 디스크로부터 읽어오는 성능의 부하를 줄일 수 있다(Block Buffer Cache라 함). 캐싱성능을 측정하기 위해 단일 노드에서 파일사이의 크기를 늘여가면서 테스트를 하였다[9].

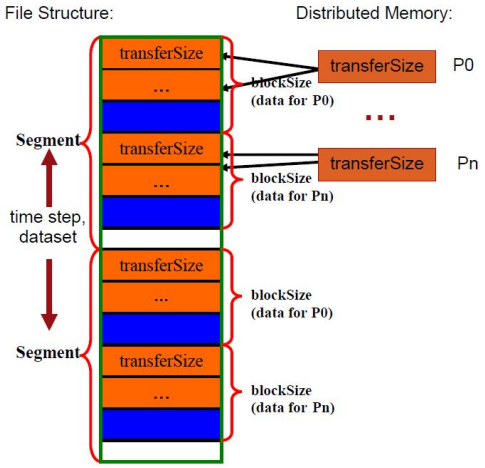
파일사이즈(FileSize)는 (그림 4)와 같이 블록사이즈*프로세스의 수*세그먼트의 수로 결정된다.

전송사이즈는 프로세스가 한번의 task당 전송하는 데이터 크기이며 정한 블록사이즈 만큼 전송하게 된다. 따라서 블록사이즈는 전송사이즈의 배수가 된다[10]. Tachyon 시스템의 계산노드는 8개의 코어로 구성되어 있으며 세그먼트의 수는 1로 지정하였다.

- $FileSize = BlockSize \times NumTasks(8) \times SegmentCount(1)$

실험은 Tachyon 단일노드에 구성된 8개의 프로세서 모두를 이용해 POSIX 인터페이스를 사용, 전송사이즈는 2MB로 고정하고 블록사이즈를 <표 3>과 같이 변경하면서 수행하였다.

(그림 4) 공유 파일 타입 구성-세그먼트, 블록, 전송사이즈




(Figure 4) Design of Shared File Type -Segment, Block, TransferSize
(Source: Using IOR to analyze the I/O performance for HPC platforms, LLNL)

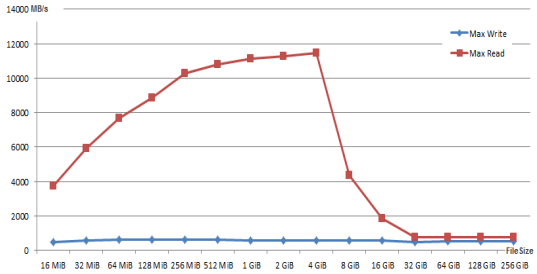
(그림 5)와 같이 파일사이즈가 4GB가 넘는 경우 캐시의 효과가 사라져 성능이 현저하게 저하됨을 확인할 수 있다.

<표 3> 블록사이즈 변경 테스트

```
for k in 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192 16384 32768 // Block Size
do
 mpirun -NP IOR.posix -a POSIX -b ${k}m -o /PFS/iorDatat-$JOB_ID -t 2m -v -w -r
done
```

<Table 3> Changing the block size test

(그림 5) 캐시 성능 테스트




(Figure 5) Cache Effect Test

## 4.2 전송사이즈 테스트

전송사이즈(TransferSize)는 앞서 언급했듯이 각 프로세스에서 한번에 전송하는 데이터 크기로 (그림 4)에서 보듯이 전체 파일을 쓰고 읽을 때 전송하는 사이즈를 의미한다. 블록사이즈는 전송사이즈의 배수가 되어야하며 전송 사이즈가 작을수록 시스템의 부하가 커져 성능이 저하된다. <표 4>는 전송사이즈를 16KB, 2MB 단위로 전송하였을 때 성능을 측정한 결과이다. 실험 결과 16KB와 같이 전송사이즈 너무 작은 경우 각 프로세스마다 전송부하 발생해 2MB 전송사이즈에 비해 읽기 속도의 경우 2배 이상의 성능차이가 나는 것을 확인할 수 있다.

<표 4> 전송사이즈 테스트 결과

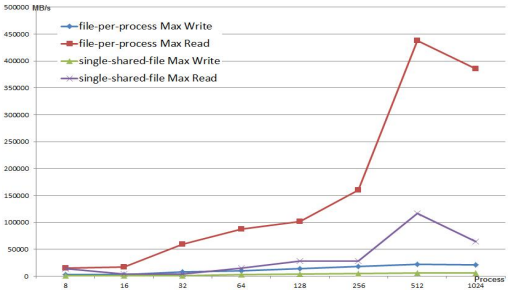
| | 16kb | 2mb |
|---|---|---|
| Max Write(MB/s) | 419.26 | 537.59 |
| Max Read(MB/s) | 4393.16 | 10084.25 |

<Table 4> Transfer Size Test

## 4.3 병렬 I/O 성능 측정

본 실험에서는 노드의 수(프로세스 개수)를 증가하면서 병렬성을 테스트하였다. Tachyon에 단일노드에 구성된 모든 코어(8개)를 사용하여 테스트하였다. 본 테스트에서는 두가지 형태로 테스트하게 된다. 모든 프로세스가 공유파일시스템에 단일 파일을 생성(single-shared-file)하는 것과 모든 프로세스가 각각 유일한 파일을 생성(file-per-process)할 때의 성능을 비교하였다.

(그림 6) 병렬 I/O 성능 테스트



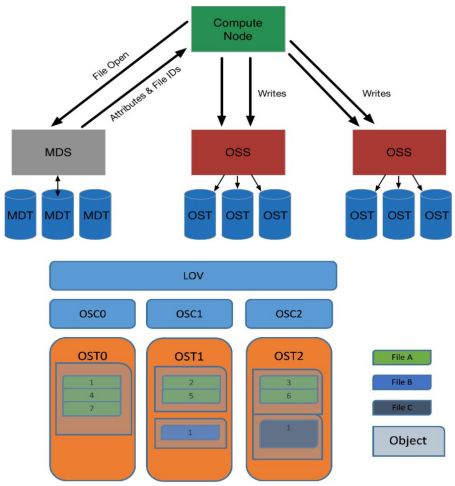

(Figure 6) Parallel I/O Test

4.1절 캐시 성능 테스트에서 확인 하였듯이 Tachyon 단일 계산노드에서 파일사이즈가 4GB 이상으로 커지는 경우 캐시의 효과가 사라져 성능이 저하됨을 확인하였다. 따라서 블록사이즈(BockSize)를 256MB로 제한하여 POSIX 인터페이스를 통하여 실험한 결과이며 (그림 6)의 그래프를 보면 512프로세스(64노드) 까지는 I/O의 총 성능이 향상됨을 볼 수 있으나 그 이후 성능이 감소됨을 확인할 수 있다. 이는 스토리지 디스크 서브시스템에 연결된 커넥터의 대역폭(실제 스토리지가 지원하는 성능)이 포화되면서 발생하는 현상으로 볼 수 있다.

### 4.4 스트라이핑(Striping) 성능 측정

Lustre 파일시스템은 객체 기반 오브젝트(Object-based) 분산 파일시스템으로 네트워크 상에 분산된 클라이언트-서버 형태로 구성된다. 클라이언트 파일시스템은 클라이언트 응용에 파일 접근 인터페이스를 제공하고, 메타데이터 서버는 저장된 파일의 오브젝트에 대한 메타데이터를 관리하면서 클라이언트에 대한 인증과 접근제어를 담당한다. 클라이언트는 파일 접근 인터페이스를 통하여 메타데이터 서버로부터 인증과 파일에 대한 메타데이터를 얻고 객체 저장 서버로부터 직접 파일 객체를 입출력 할 수 있다[11].

(그림 7) Lustre 구조 및 스트라이핑 개념



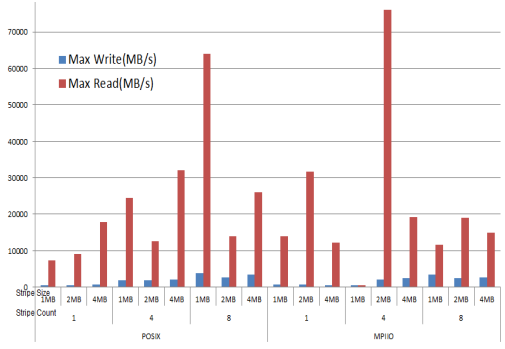

(Figure 7) Lustre Structure and Striping Concept

(그림 7)은 Lustre 파일시스템 구조와 파일 스트라이핑(Striping)의 개념을 나타내고 있다. 파일 스트라이핑은 Lustre의 대표적인 특징이라 할 수 있으며 OST(Object Storage Target), 즉 물리적으로 분산되어있는 디스크에 하나의 파일을 분산시켜 저장함으로써 병목현상을 제거하고 I/O 성능을 향상시킬 수 있다[12]. Lustre 기본 설정으로 파일이 분산 저장되는 개수를 나타내는 스트라이프 카운트(stripe count)는 1개, 각 분산 저장시 저장되는 파일의 사이지를 나타내는 스트라이프 사이즈(stripe size)는 1MB로 되어있다. 스토리지의 성능과 파일시스템의 구성 또는 사용자의 작업 데이터 특성에 따라 아래와 같이 설정하여 스트라이프 카운트와 사이즈를 설정할 수 있다.

- *lfs setstripe [-s stripe_size] [-c stripe_count] <filename|dirname>*

본 실험에서는 POSIX와 MPIIO 인터페이스를 이용하여 (그림 8)과 같이 스트라이프 카운트와 사이즈를 증가시키며 16개의 노드에서 2개의 프로세스를 사용하여 테스트하였다. (그림 8)에서 보듯이 스트라이프 카운트와 사이즈가 커질수록 전반적으로 성능이 증가하는 것을 볼 수 있으며, Max Read에서 POSIX의 경우 스트라이프 카운트가 8개 사이즈가 1MB 일 때 가능 좋은 성능을 MPIIO의 경우 스트라이프 카운트가 4개 사이즈가 2MB 일 때 가장 좋은 성능이 나타남을 확인할 수 있었다.

(그림 8) 스트라이핑 성능 테스트



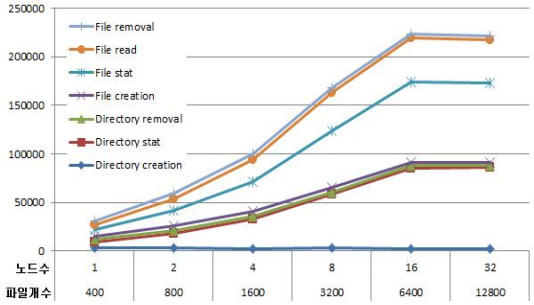

(Figure 8) Striping Performance Test

### 4.5 메타데이터 성능 측정

메타데이터 성능 측정을 위해서 오픈소스 도구인 MDTEST를 사용하였다. 이 벤치마크 도구는 MPI를 활용하여 글로벌 파일시스템의 메타데이터 동작에 대한 성능을 측정하게 된다.

빌드를 위해서 C 컴파일러와 MPI 라이브러리가 필요하며 실행시에는 프로세스에서 실행될 파일의 개수, 공용 또는 개별 디렉터리 선택, 테스트 수행 옵션(Create, Stat, Remove)을 설정해야 한다.

(그림 9) MDTEST 수행 결과




(Figure 9) MDTEST Test Result

공용 파일시스템의 단일 디렉터리에서 테스트를 수행하였으며, (그림 9)와 같이 Creation, Removal 동작의 경우에는 여러 프로세스가 단일 디렉터리 액세스로 인한 Locking 메커니즘 영향으로 성능이 증가하지 않고 일정하게 나타남을 볼 수 있다[13][14]. 디렉터리 파일의 초당 I/O 오퍼레이션(IOPS)이 3000~4000인 반면에 Stat의 초당 I/O오퍼레이션 16노드까지는 비례적으로 증가하였다.

## 5. 결 론

HPC 환경에서 파일시스템의 성능은 연구자가 어플리케이션을 수행할 때 반드시 고려할 상황이다. 파일시스템의 성능은 계산노드 요소 구성, 인터커넥션 네트워크 대역폭, 스토리지의 용량 및 구성에 따라 크게 좌우된다.

본 연구에서는 Tachyon 시스템에서 파일사이즈, 단일노드 I/O, 병렬노드 I/O, 스트라이핑 설정 및 메타데이터 성능을 각각 테스트하고 각 테스트에서 최적의 설정 환경을 확인 할 수 있었다. 이를 활용하여 사용자의 작업 코드를 최적화하여 작업 수행시간을 최소화하고 전체 시스템 효율성을 높일 수 있다. 나아가 본 실험을 활용하여 차후 도입될 시스템에서 또한 파일시스템 최적 성능을 구할 수 있다.

향후, 다양한 파일시스템 성능요소를 측정하기 위한 추가로 필요한 벤치마크 도구 선정하고 적용하면 좀 더 면밀하고 다양한 측정을 수행할 수 있다. 나아가 자동화 도구를 통해 시스템의 주기적인 성능 점검과 로그들을 파악함으로써 파일시스템의 가용성을 더욱 높일 수 있다.

## References

[1] Zhao, D., Shou, C., Zhang, Z., Sadooghi, I., Zhou, X., Li, T., & Raicu, I.“ FusionFS: a distributed file system for large scale data-intensive computing”, In Greater Chicago Area System Research Workshop pp. 10-11, 2013

[2] CEO & President George Teixeira“Why Parallel I/O Software and Moore’s Law Enable Virtualization and Software-Defined Data Centers to Achieve their Potential”, DataCore Opinion Paper, Aug, 2015

[3] Sun Microsystems, Inc., “LUSTRETM FILE SYSTEM”, Oct. 2008.

[4] Oral, Sarp, et al. "OLCF's 1 TB/s, next-generation lustre file system." Proceedings of Cray User Group Conference (CUG 2013), 2013.

[5] LOEWE, William; MCLARTY, T.; MORRONE, C., “Ior benchmar”, 2012, https://sourceforge.net/projects/ior-sio.

[6] MPI-2: Extensions to the Message Passing Interface. http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html

[7] Shan, Hongzhang, and John Shalf. "Analysis of Parallel I/O for NERSC HPC Platforms: Application Requirements, Benchmarks, and Delivered System Performance.“

[8] Mdtest benchmark. sourceforge.net/projects/mdtest.

[9] Eshel, Marc, et al. "Panache: A Parallel File System Cache for Global File Access." FAST. pp.155-168. 2010.

[10] SHAN, Hongzhang; SHALF, John. Using IOR to Analyze the I/O performance for HPC Platforms. La wrence Berkeley National Laboratory,2007.

[11] University of Colorado Research Computing, "Parall el IO on Janus Lustre"

[12] DEVENDRAN, Dharshi, et al." Collective I/O Optim izations for Adaptive Mesh Refinement Data Writes on Lustre File System", 2016.

[13] Oertel, René. "Benchmarking the Chemnitz High Pe rformance Linux Cluster (CHiC).", Jan, 2008.

[14] Alexander Oltu UniBCCS, "Performance Analysis-Synthetic benchmarks: IOR, bonnie++, mdtest", 201 0.

**윤 준 원**

2004년 : 고려대학교 대학원 컴퓨터학과(이학석사)
2011년 : 고려대학교 대학원 컴퓨터학과 박사 수료
2005년~현재 : KISTI 국가슈퍼컴퓨팅연구소 선임연구원
관심분야: 분산 컴퓨팅, 결함포용시스템, 슈퍼컴퓨팅, 병렬파일시스템, 배치스케줄링

**송 의 성**

1997년 : 고려대학교 컴퓨터학과(학사)
1999년: 고려대학교 대학원 컴퓨터학과(이학석사)
2005년: 고려대학교 대학원 컴퓨터학과(이학박사)
2006년~현재: 부산교육대학교 컴퓨터교육과 교수
관심분야: 컴퓨터교육, 교육용로봇교육, 컴퓨터네트워크, 스마트러닝, 분산컴퓨팅