2021 KoALICE Workshop

Research on EOS Disk Storage using Docker

2022.01.07

충북대학교

컴퓨터과학과

DCLab 이준영

2021 Timeline

Research Preparation

EOS Storage System on Docker Container

- Configure KISTI JBOD using Docker to install EOS storage
- Testing EOS storage using FUSE and FUSEx client
- Journal Paper (SCI)
- Conference Paper(Domestic)

Naver Cloud Intern

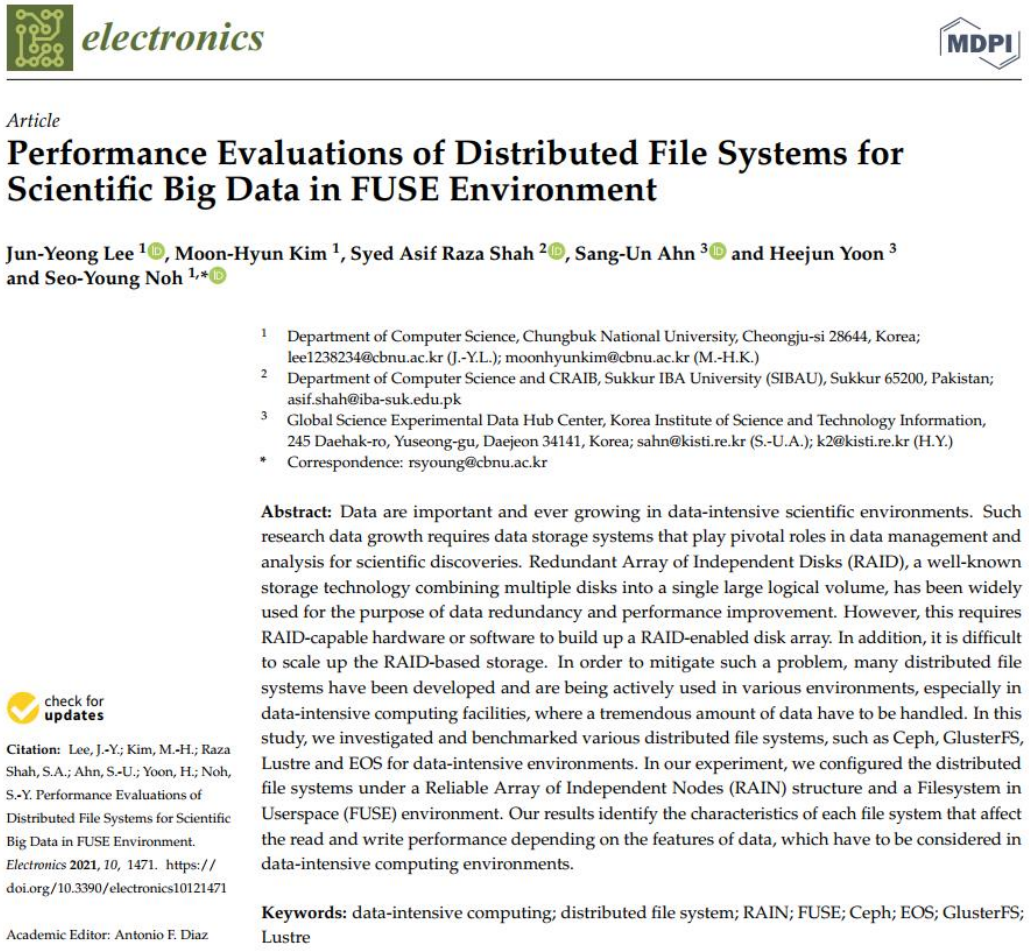
Finalizing Master Degree

- Graduation thesis
- Journal Paper(KCI)

Journal Paper – SCI(E)

Performance Evaluations of Distributed File Systems for Scientific Big Data in FUSE Environment

- 각 분산 파일 시스템을 Linux의 FUSE 클라이언트를 통해 동일한 조건 및 환경에서 성능 평가 진행
- 실험 결과를 통해 데이터의 특성에 따른 성능을 파악할 수 있는 결과를 도출함
- SCI(E) 저널 Electronics에 게재됨



electronics

MDPI

Article

Performance Evaluations of Distributed File Systems for Scientific Big Data in FUSE Environment

Jun-Yeong Lee [1], Moon-Hyun Kim [1], Syed Asif Raza Shah [2], Sang-Un Ahn [3] and Heejun Yoon [3] and Seo-Young Noh [1,*]

[1] Department of Computer Science, Chungbuk National University, Cheongju-si 28644, Korea; lee1238234@cbnu.ac.kr (J.-Y.L.); moonhyunkim@cbnu.ac.kr (M.-H.K.)
[2] Department of Computer Science and CRAIB, Sukkur IBA University (SIBAU), Sukkur 65200, Pakistan; asif.shah@iba-suk.edu.pk
[3] Global Science Experimental Data Hub Center, Korea Institute of Science and Technology Information, 245 Daehak-ro, Yuseong-gu, Daejeon 34141, Korea; sahn@kisti.re.kr (S.-U.A.); k2@kisti.re.kr (H.Y.)
* Correspondence: rsyoung@cbnu.ac.kr

Abstract: Data are important and ever growing in data-intensive scientific environments. Such research data growth requires data storage systems that play pivotal roles in data management and analysis for scientific discoveries. Redundant Array of Independent Disks (RAID), a well-known storage technology combining multiple disks into a single large logical volume, has been widely used for the purpose of data redundancy and performance improvement. However, this requires RAID-capable hardware or software to build up a RAID-enabled disk array. In addition, it is difficult to scale up the RAID-based storage. In order to mitigate such a problem, many distributed file systems have been developed and are being actively used in various environments, especially in data-intensive computing facilities, where a tremendous amount of data have to be handled. In this study, we investigated and benchmarked various distributed file systems, such as Ceph, GlusterFS, Lustre and EOS for data-intensive environments. In our experiment, we configured the distributed file systems under a Reliable Array of Independent Nodes (RAIN) structure and a Filesystem in Userspace (FUSE) environment. Our results identify the characteristics of each file system that affect the read and write performance depending on the features of data, which have to be considered in data-intensive computing environments.

Keywords: data-intensive computing; distributed file system; RAIN; FUSE; Ceph; EOS; GlusterFS; Lustre

check for updates

Citation: Lee, J.-Y.; Kim, M.-H.; Raza Shah, S.A.; Ahn, S.-U.; Yoon, H.; Noh, S.-Y. Performance Evaluations of Distributed File Systems for Scientific Big Data in FUSE Environment. *Electronics* **2021**, *10*, 1471. https://doi.org/10.3390/electronics10121471

Academic Editor: Antonio F. Diaz

Conference Paper - Domestic

CERN EOS 분산 파일 시스템의 배포 환경에 따른 I/O 성능 비교

- **서버를 통한 물리적인 클러스터와 Linux의 KVM 기반 가상머신을 통한 가상 클러스터에서의 EOS 분산 파일 시스템 성능을 비교**

- **최우수논문으로 선정되어 현재 정보과학회논문지(KCI)에 게재 예정**

제21-1853호

최 우 수 논 문 상

제 목 : CERN EOS 분산파일시스템의 배포환경에 따른 I/O성능비교

저 자 : 이준영·김문현·김경준·노서영(충북대학교)

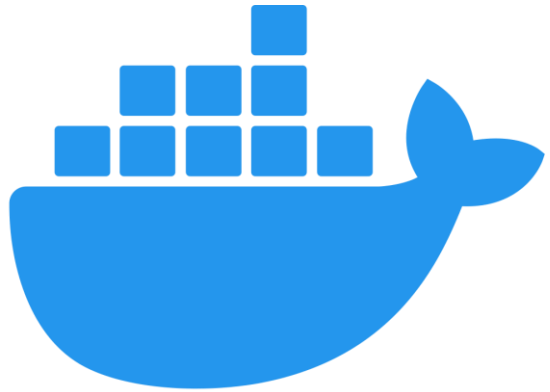
위 논문은 2021 한국컴퓨터종합학술대회에서 컴퓨터시스템 분야의 최우수논문으로 선정되었기에 이 상을 수여함.

2021년 6월 24일

한국정보과학회
회 장 나 연 묵

EOS on Docker @ KISTI JBOD

- 기존의 EOS 스토리지는 서버 OS에 직접 설치하여 구현되는 베어메탈 방식으로 구현되었음
- EOS는 실험적으로 사용할 수 있는 Docker 이미지를 배포중
- 이번 실험에서는 OS에 직접 설치하지 않고, Docker을 통해 EOS의 구성 요소를 컨테이너화하여 Docker상에서 EOS가 어떻게 동작하는지 분석하였음
- 작년과 달리, KISTI에서 고성능의 테스트 서버를 제공하여 실제 프로덕션 레벨의 하드웨어를 통해 실험을 진행할 수 있었음.



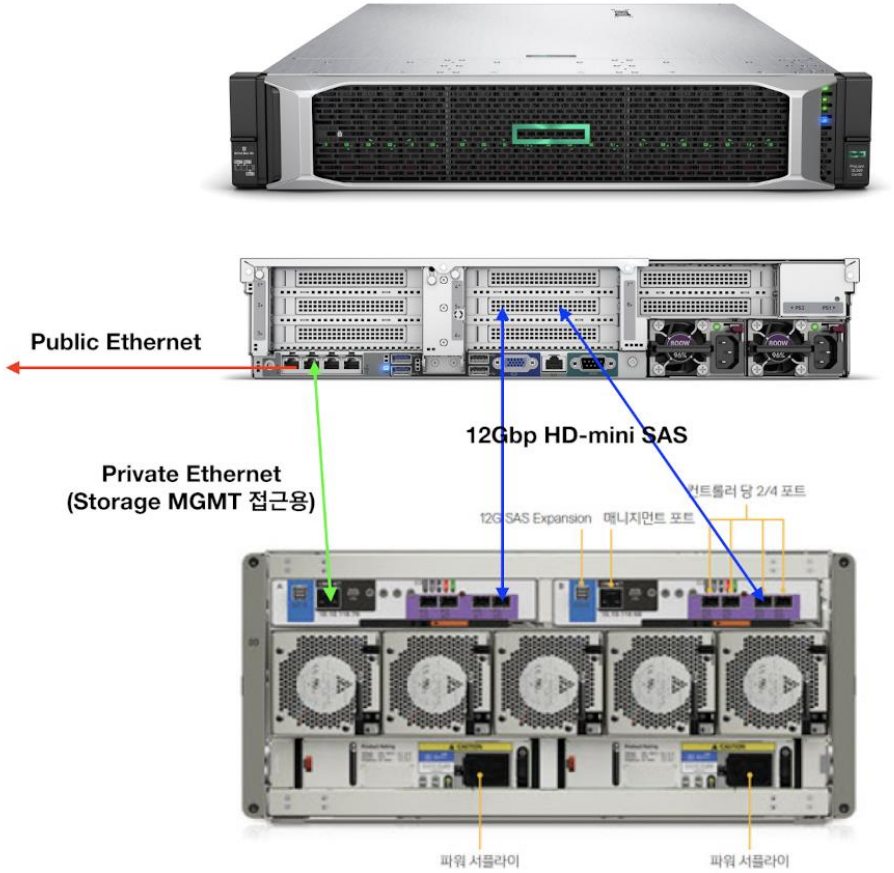
docker

EOS

KISTI JBOD

- **KISTI JBOD**
  - **CPU: Xeon Gold 6230 20Core * 4**
  - **RAM: 768GB**
  - **HDD: 12TB * 70EA**
  - **Storage Enclosure: Dell ME4084**

- **8 Separate RAID 5 volume with 5 HDDs were created for EOS storage.**



KISTI JBOD

- **KISTI JBOD**
  - **CPU: Xeon Gold 6230 20Core * 4**
  - **RAM: 768GB**
  - **HDD: 12TB * 70EA**
  - **Storage Enclosure: Dell ME4084**

- **8 Separate RAID 5 volume with 5 HDDs were created for EOS storage.**

- **Volume was mapped to server using multipath configuration.**

EOS on Docker @ KISTI JBOD

- **EOS storage was deployed using Docker and RAID volumes.**
- **Data layout was set to RAID 6 to mimic O^2 storage configuration.**

```
[root@cbnu-ui ~]# docker ps
CONTAINER ID        IMAGE               COMMAND
8cbfc483d5eb        centos:7            "/bin/bash"
de9b23c5fab3        971a0225            "/startup_script.sh"
d80b279d199e        971a0225            "/startup_script.sh"
2a12d4fba804        3f9f4ac355ea        "/bin/bash"
8aaf8fa38049        3f9f4ac355ea        "/bin/bash"
1bbbe48112a6        3f9f4ac355ea        "/bin/bash"
94594b6850aa        3f9f4ac355ea        "/bin/bash"
70fd3e63aac6        3f9f4ac355ea        "/bin/bash"
cb50e388e49e        3f9f4ac355ea        "/bin/bash"
f12e9f334477        3f9f4ac355ea        "/bin/bash"
d958c5a0d476        3f9f4ac355ea        "/bin/bash"
402c613ee0be        3f9f4ac355ea        "/bin/bash"
1590b4db687f        3f9f4ac355ea        "/bin/bash"
e96703072d5f        3f9f4ac355ea        "/bin/bash"
3b2e3d4b461f        3f9f4ac355ea        "/bin/bash"
```

```
EOS Console [root://localhost] |/eos/plain/> space ls
```

type	name	groupsize	groupmod	N(fs)	N(fs-rw)	sum(usedbytes)	sum(capacity)	capacity(rw)	nom.capacity	quota	balancing	threshold	converter
spaceview	default	8	24	8	8	76.71 GB	375.81 TB	375.81 TB	0 B	off	off	20	off

```
EOS Console [root://localhost] |/eos/plain/> fs ls
```

host	port	id	path	schedgroup	geotag	boot	configstatus	drain	active	health
fst1.eos.docker	1095	1	/jbod1	default.0	docker::test	booted	rw	nodrain	online	no mdstat
fst2.eos.docker	1095	2	/jbod2	default.0	docker::test	booted	rw	nodrain	online	no mdstat
fst3.eos.docker	1095	3	/jbod3	default.0	docker::test	booted	rw	nodrain	online	no mdstat
fst4.eos.docker	1095	4	/jbod4	default.0	docker::test	booted	rw	nodrain	online	no mdstat
fst5.eos.docker	1095	5	/jbod5	default.0	docker::test	booted	rw	nodrain	online	no mdstat
fst6.eos.docker	1095	6	/jbod6	default.0	docker::test	booted	rw	nodrain	online	no mdstat
fst7.eos.docker	1095	7	/jbod7	default.0	docker::test	booted	rw	nodrain	online	no mdstat
fst8.eos.docker	1095	8	/jbod8	default.0	docker::test	booted	rw	nodrain	online	no mdstat

EOS Benchmark Methods

- **FUSE**

  Pros

  - Easy to benchmark.

  Cons

  - Access performance is lower due to translation layer.

- **FUSEx**

  Pros

  - Easy to benchmark.

  - Has improved performance than FUSE

  Cons

  - Same as FUSE.

- **XRootD**

  Pros

  - Native protocol for EOS.

  - Has high performance compared to other methods.

  Cons

  - No tools for benchmarking

- **WebDAV(HTTP)**

  Pros

  - Based on HTTP protocol

  - Has many ways to benchmark.

  Cons

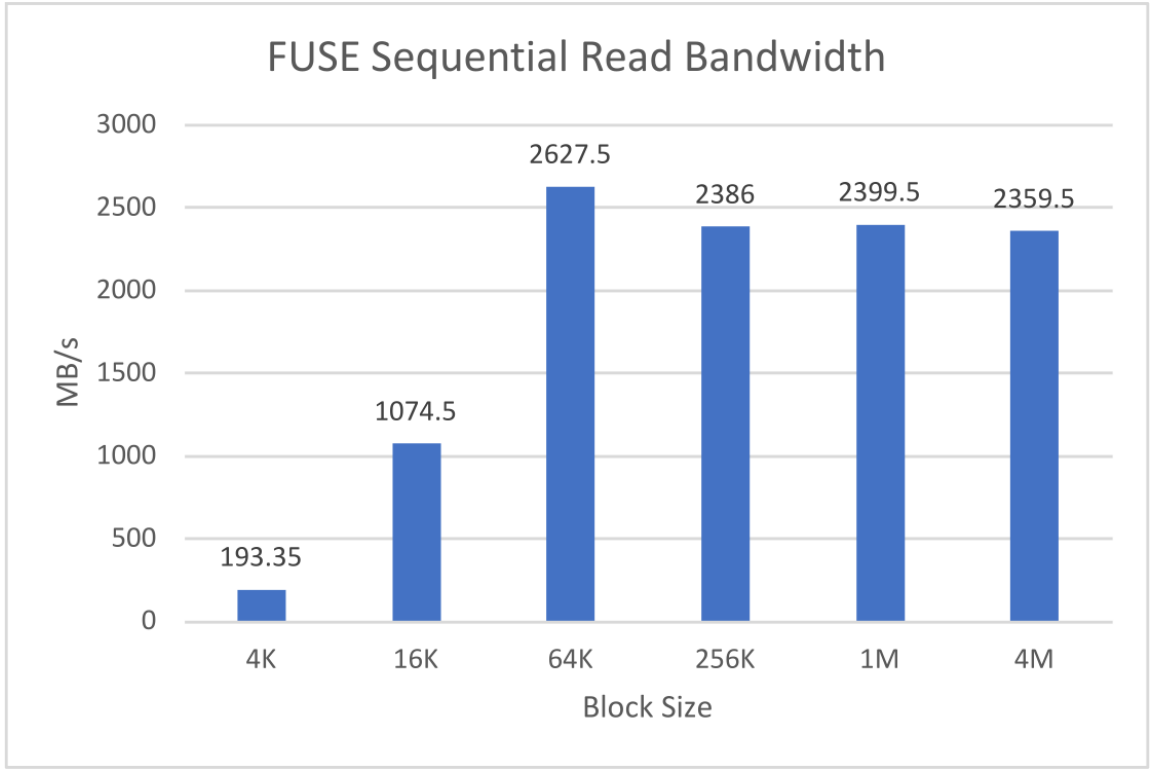
  - HTTP access is restricted

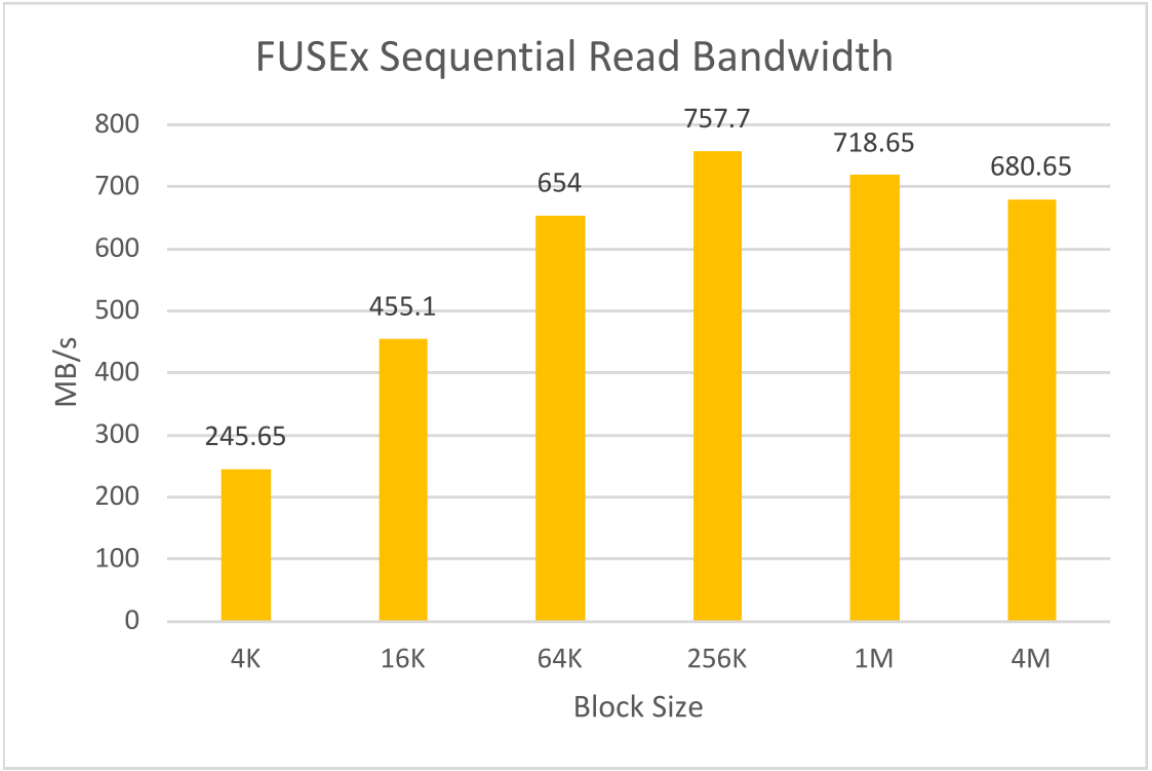
VDBench Benchmark

- Developed by Oracle.
- Used to benchmark large-scale storage system.
- Able to direct access Linux userspace and many distributed file system's own API.
- Benchmark was performed with FUSE and FUSEx client increasing the size of transfer blocks from 4 to 4096K.
- XRootD cannot be used because it does not support XRootD protocol.

Result – Sequential Read

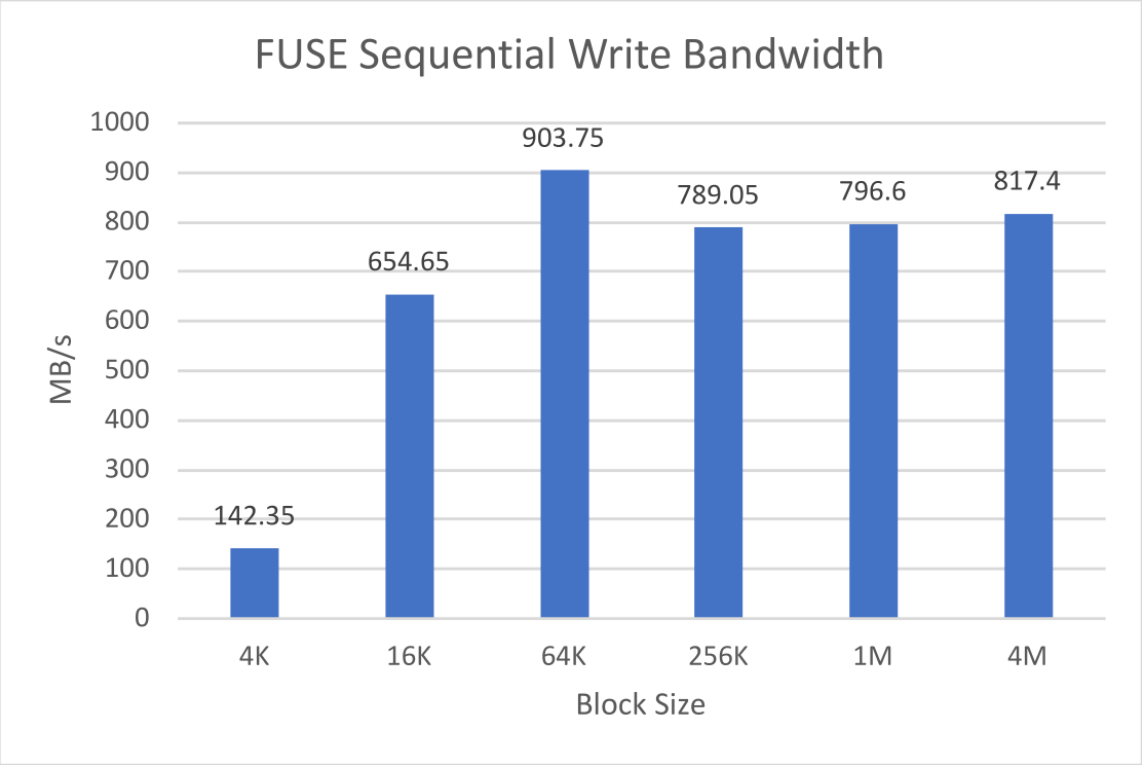
- **FUSE client shows constant bandwidth after 64K block.**
- **4K performance of FUSEx client was higher than FUSE client, however other block result shows much lower performance than FUSE client.**

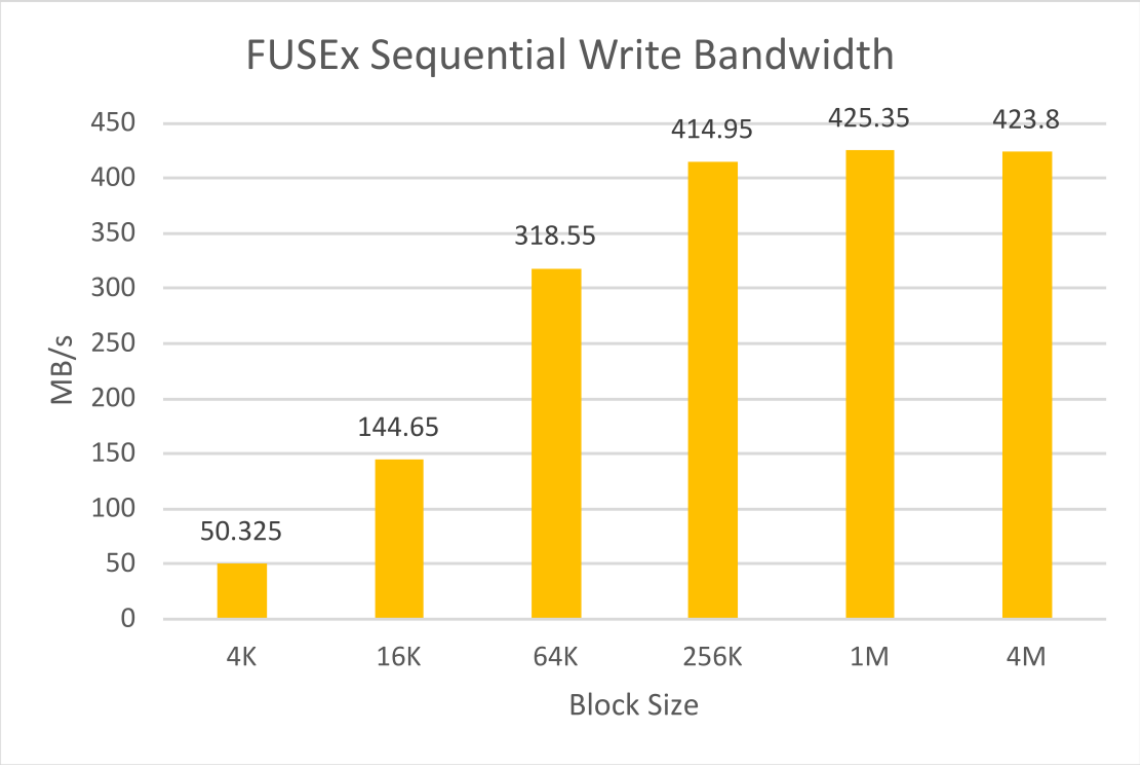




Result – Sequential Write

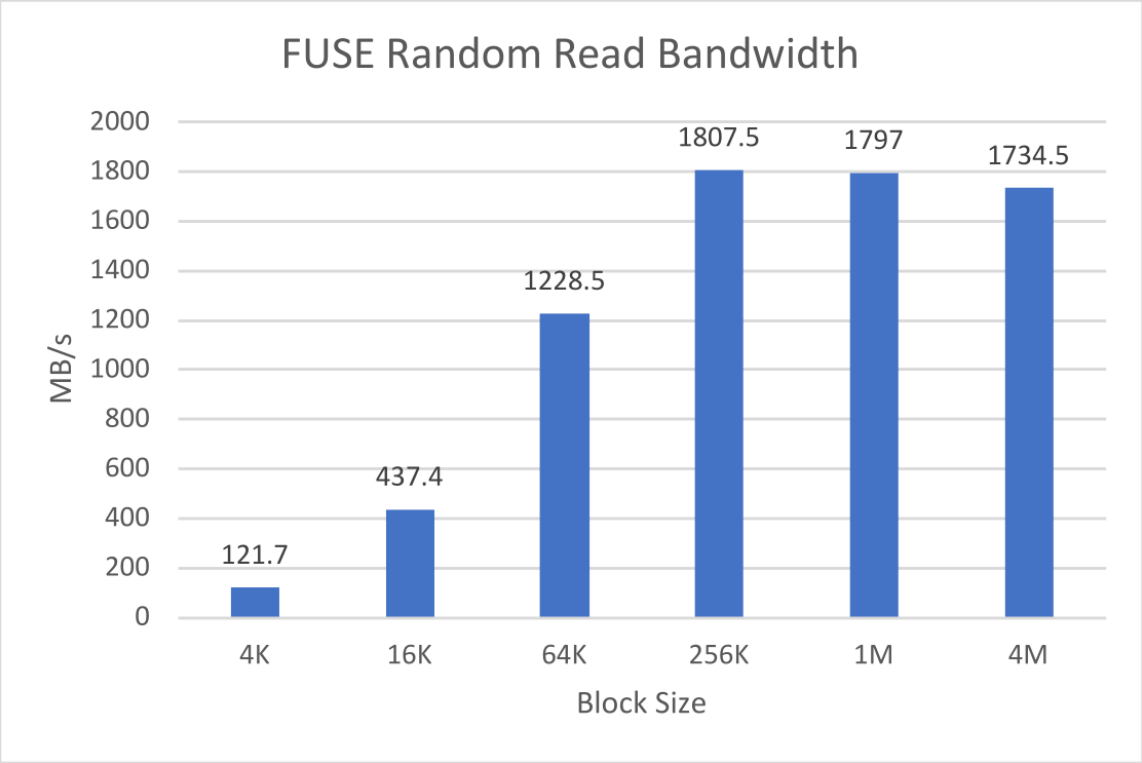
- **FUSE client shows similar pattern to sequential read.**
- **FUSEx client shows stable bandwidth increase, but lower performance compared to FUSE.**

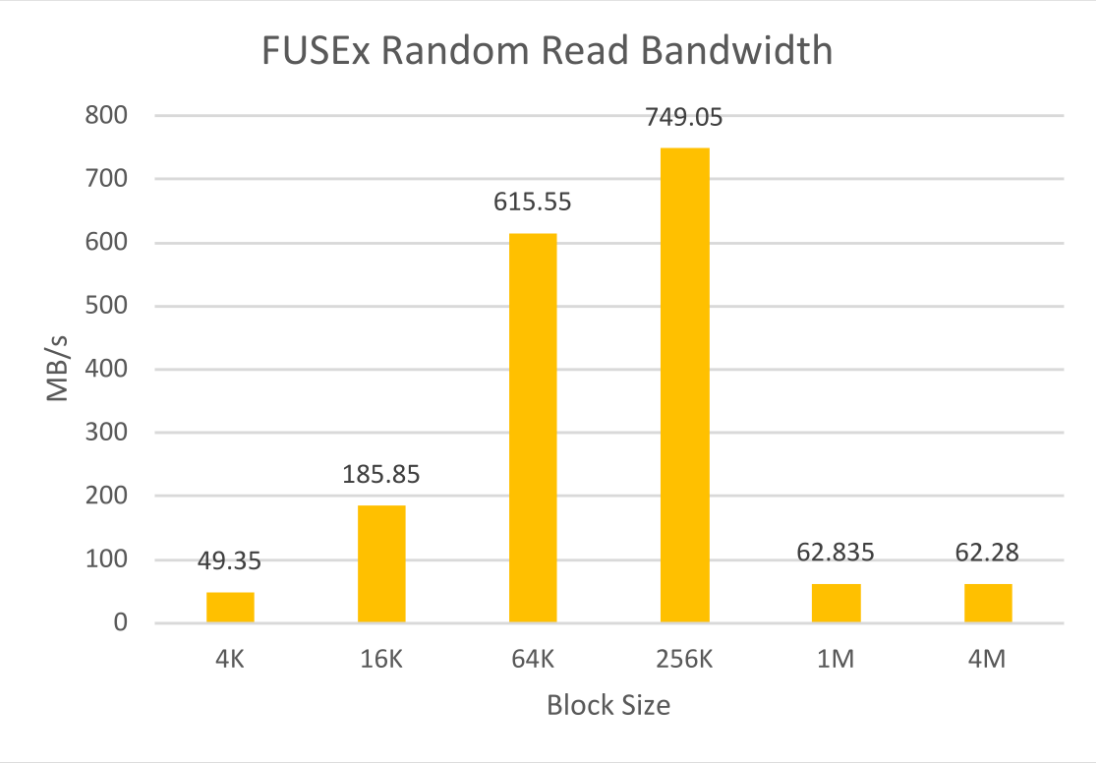




Result – Random Read

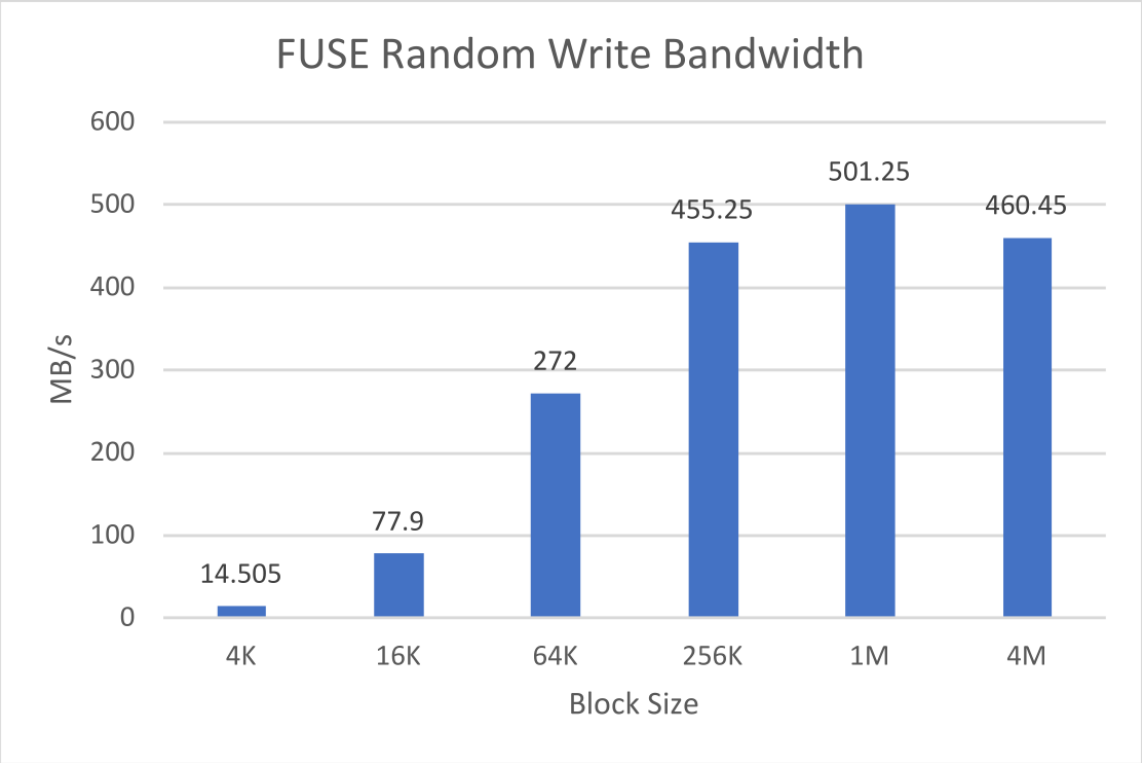
- **Unlike sequential result, FUSE shows bandwidth increase up to 256K block.**
- **FUSEx shows high bottleneck after 256K block.**

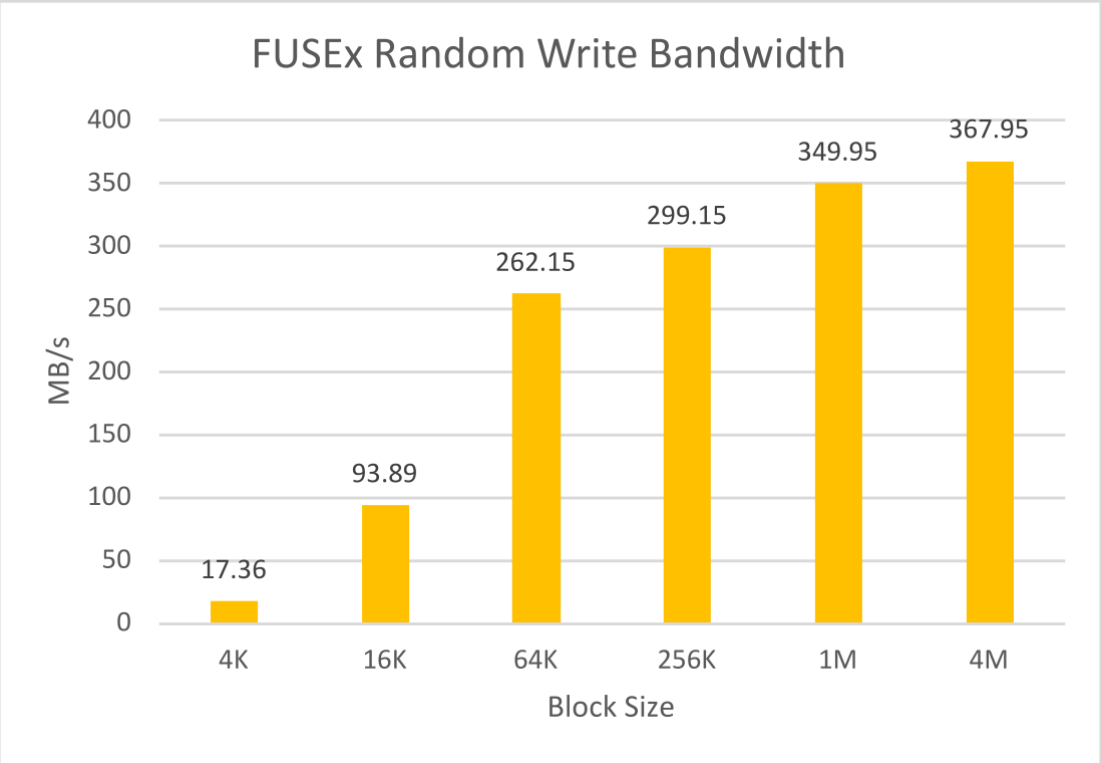




Result – Random Write

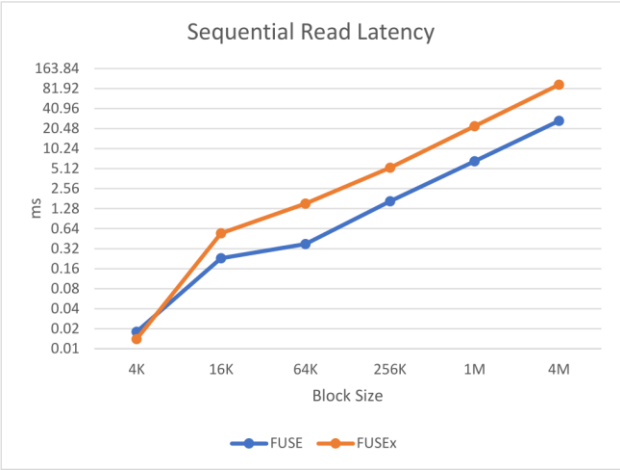
- **FUSEx client shows higher performance at 4K and 16K block.**
- **After 256K block, FUSE client performance is higher than FUSEx client.**

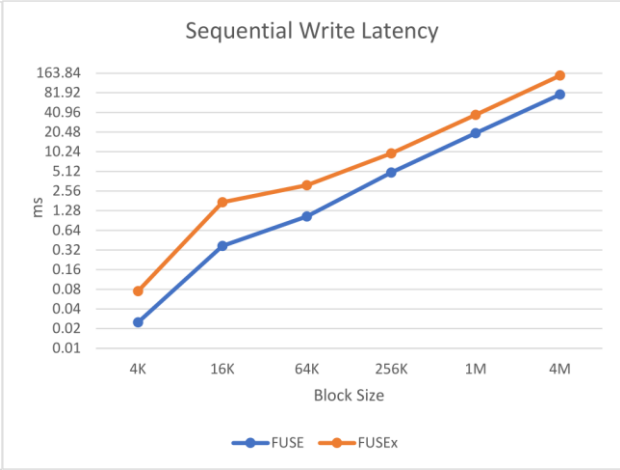


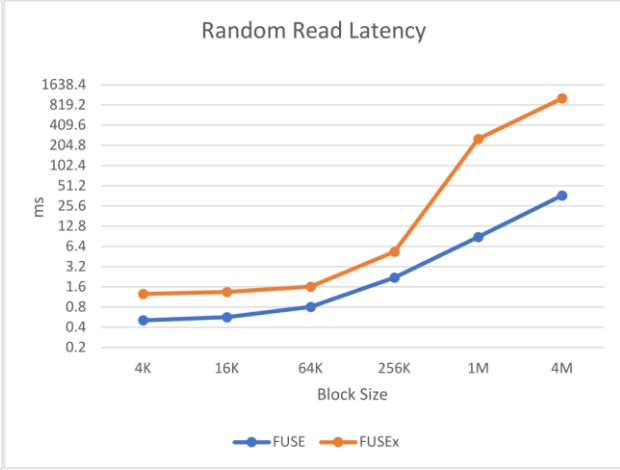


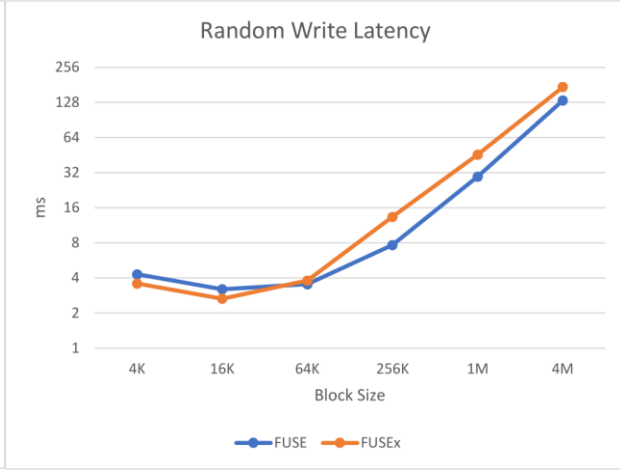
Result – Sequential/Random Latency

- **Similar to bandwidth result, FUSE client's latency was lower than FUSEx client.**
- **FUSEx has lower latency when small block size were used.**
- **But large block shows long latency time then FUSE client.**





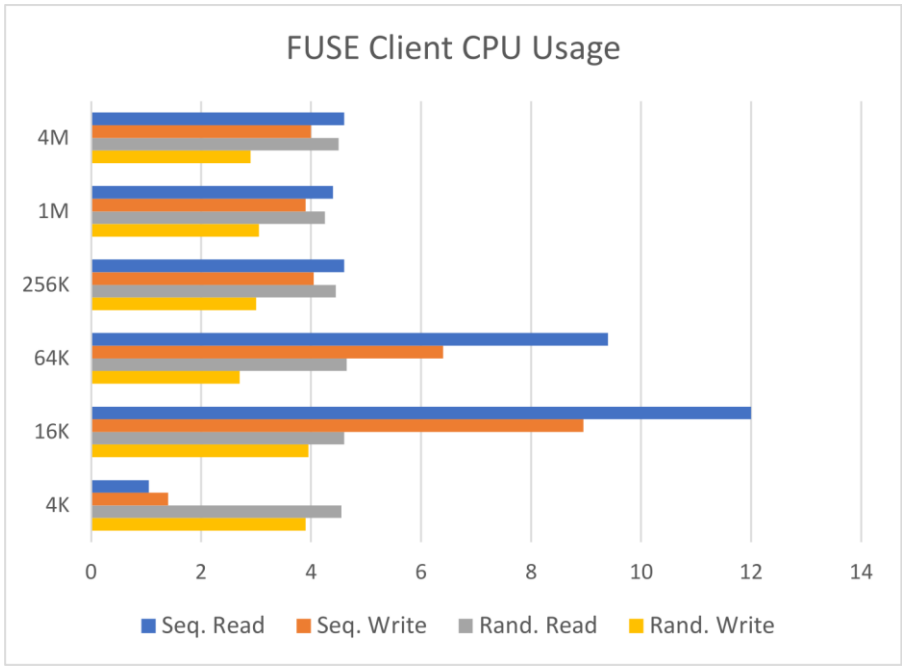


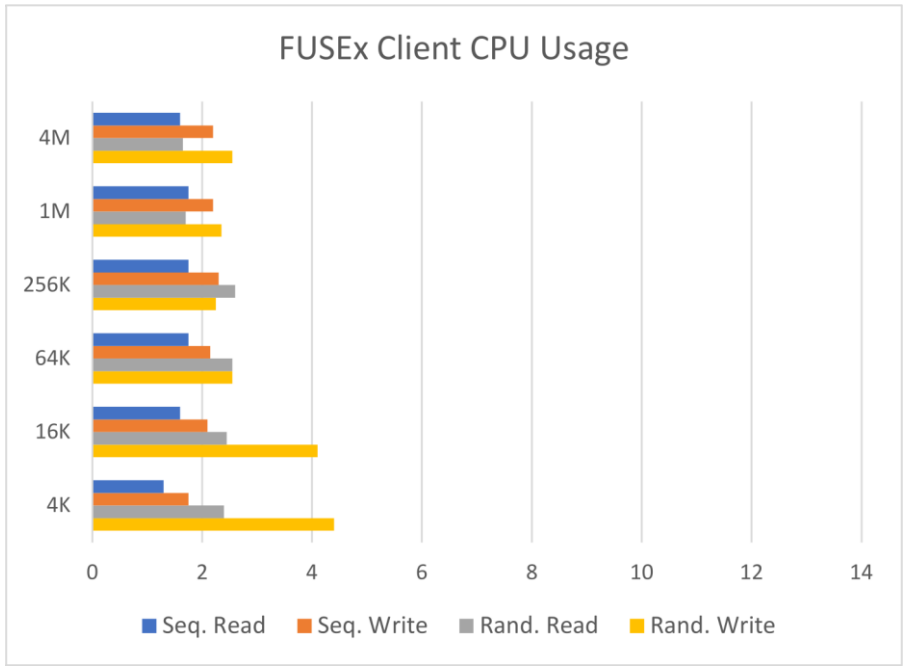


Result – CPU Usage

- **CPU utilization of FUSEx client was lower than FUSE client.**

- **Maybe client performance is bound to CPU utilization, but not sure about it.(might be bug)**





Summary

- Performance of FUSEx client was poorer than legacy FUSE client.
- Also, latency result shows same result – FUSEx client had higher latency than FUSE Client except small block size.
  - This can be bug because production environment needs small size block performance.
- Same phenomenon was observed when multiple clients were used.
- Although CPU utilization for FUSEx client was way lower than FUSE client, we cannot analyze the result because it needs more low-level inspection.
- Have to think about benchmark with XRootD.

Thank You