4190.568: Advanced Operating Systems

Jin-Soo Kim
(*jinsoo.kim@snu.ac.kr*)

Systems Software & Architecture Lab.

Seoul National University

Spring 2026

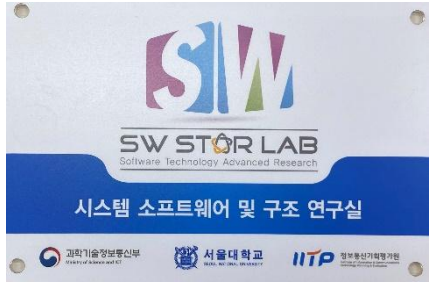
Course Information

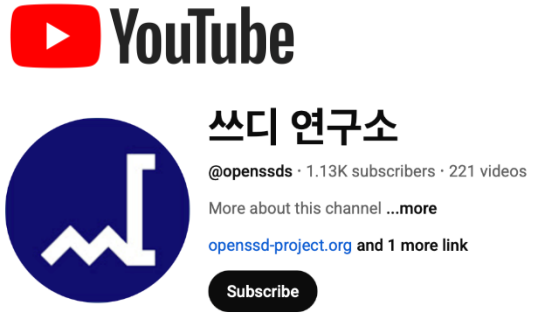
- Schedule
  - 11:00 – 12:15 (Tuesday & Thursday)
  - Engineering Bldg. #301-203
  - 3 credits
  - Official language: Korean

- Course homepage: https://csl.snu.ac.kr/courses/4190.568/2026-1/
- Lecture slides will be uploaded in the course homepage before the class

About Me

- Jin-Soo Kim (김진수)
  - Professor @ CSE Dept.
  - Systems Software & Architecture Laboratory
  - Operating systems, storage systems, parallel and distributed computing, embedded systems, …





- E-mail: jinsoo.kim@snu.ac.kr
- Tel: 02-880-7302
- Office: Engineering Bldg. #301-504
- The best way to contact me is by email

Prerequisites

- Prerequisites
  - M1522.000800 Undergraduate Systems Programming or equivalent
  - 4190.307 Undergraduate Operating Systems or equivalent
  - 4190.308 Undergraduate Computer Architecture or equivalent

- We will review some of fundamental operating system concepts to awaken the force within you

Image from https://pngimg.com/download/28396

Course Plan

- Lectures
  - Advanced topics on operating systems
  - Linux case study
- Invited talks
- Reading assignments
  - ~ 1 paper per week
  - ~~You need to fill out Google forms~~
- ~~Paper presentation~~
- Term project
- Final exam

Topics Planned

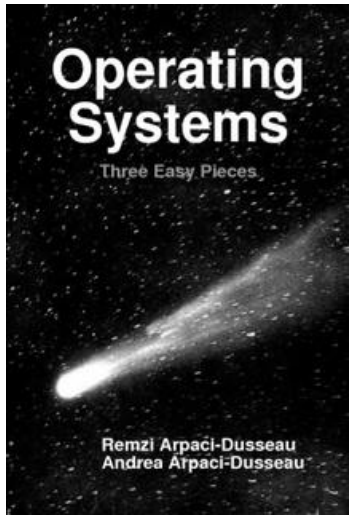
- Introduction to computer systems research
- Introduction to operating systems
- Storage
- File systems
- SSDs
- Processes and threads
- CPU scheduling
- ~~Synchronization~~
- Virtual memory
- Linux memory management
- Virtual machines
- OS structure and design

Class Materials

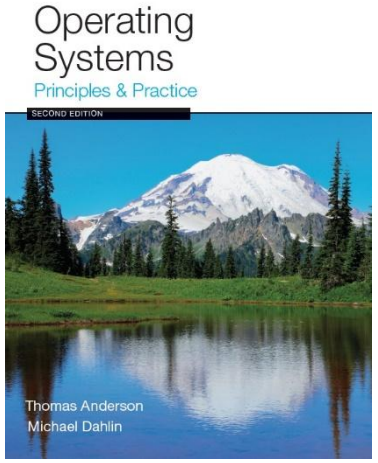
- Quality research papers from major conferences will be used:
  - SOSP (ACM Symposium on Operating Systems Principles)
  - OSDI (USENIX Symposium on Operating Systems Design and Implementation)
  - ASPLOS (ACM Conference on Architectural Support for Programming Languages and Operating Systems)
  - USENIX ATC (USENIX Annual Technical Conference)
  - FAST (USENIX Conference on File and Storage Technologies)
  - EuroSys (ACM European Systems Conference)
  - NSDI (USENIX Symposium on Networked Systems Design and Implementation)
  - …

References

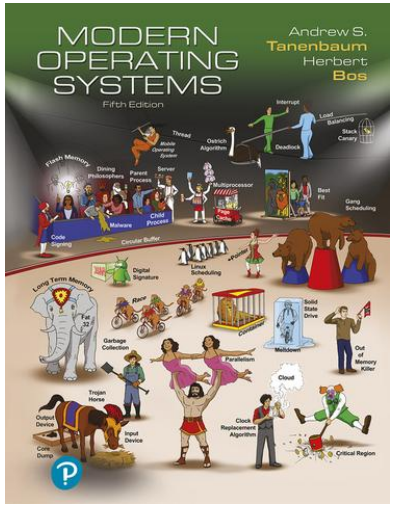
- Operating Systems: Three Easy Pieces
  - By Remzi & Andrea Arpaci-Dusseau
  - Freely available at http://ostep.org



- Operating Systems: Principles and Practice
  - By Tom Anderson & Michael Dahlin
  - 2nd Edition, Recursive Books, 2014



- Modern Operating Systems
  - By Andrew Tanenbaum & Herbert Bos
  - 5th Edition, Pearson Education, 2022



Reading Assignments

- Critical reading of technical papers is a must skill to have for your research
- The reading list will be posted in the course home page

Reading List

Historical Perspective

- (H1) D. Ritchie and K. Thompson, "The UNIX Time-Sharing System," CACM, 1974. (The SIGOPS Hall of Fame Award '05)
- (H2) ⭐ D. Ritchie, "The Evolution of the Unix Time-sharing System," AT&T Bell Laboratories Technical Journal 63, No. 6, October 1984, pp.1577-93.

Computer Systems Research

- (I1) ⭐ Butler W. Lampson, "Hints for Computer System Design," SOSP, 1983. (The SIGOPS Hall of Fame Award '05)
- (I2) Roy Levin and David D. Redell, "An Evaluation of the Ninth SOSP Submissions or How (and How Not) to Write a Good Systems Paper," ACM Operating Systems Review, 1983.

Projects

- Term projects should be done in teams of three students
- Each project should be completed within this semester with some tangible results
  - New ideas without any evaluation will not be considered for grading, no matter how novel they are
- Project topics need to be related to operating systems (especially to storage and file systems with NVMeVirt ☺)

NVMeVirt [FAST '23]

- A versatile software-defined virtual NVMe device

- Source code available at https://github.com/snu-csl/nvmevirt

NVMeVirt: A Versatile Software-defined Virtual NVMe Device

Sang-Hoon Kim, *Ajou University*
Jaehoon Shim, *Seoul National University*
Euidong Lee, *Seoul National University*
Seongyeop Jeong, *Seoul National University*
Ilkueon Kang, *Seoul National University*
Jin-Soo Kim, *Seoul National University*

Abstract

There have been drastic changes in the storage device landscape recently. At the center of the diverse storage landscape lies the NVMe interface, which allows high-performance and flexible communication models required by these next-generation device types. However, its hardware-oriented definition and specification are bottlenecking the development and evaluation cycle for new revolutionary storage devices.

In this paper, we present NVMeVirt, a novel approach to facilitate software-defined NVMe devices. A user can define any NVMe device type with custom features, and NVMeVirt allows it to bridge the gap between the host I/O stack and the virtual NVMe device in software. We demonstrate the advantages and features of NVMeVirt by realizing various storage types and configurations, such as conventional SSDs, low-latency high-bandwidth NVM SSDs, zoned namespace SSDs, and key-value SSDs with the support of PCI peer-to-peer DMA and NVMe-oF target offloading. We also make cases for storage research with NVMeVirt, such as studying the performance characteristics of database engines and extending the NVMe specification for the improved key-value SSD performance.

1 Introduction

NAND flash memory gains significant popularity for consumer devices and enterprise servers, and the fast advancement of semiconductor technologies fosters the non-volatile memory (NVM) to build storage devices, enlightening high-density low-latency storage devices. Nowadays, we can purchase off-the-shelf storage devices, which feature tens of microsecond latency and several GiB/s of bandwidth [16, 47].

Along with the performance and data density improvement, there has been an active trend toward making storage devices smarter and more capable. For efficient and effective data processing and management, many innovative device concepts have been proposed, including but not limited to Open-Channel SSD (OCSSD) [5, 34, 41], zoned namespace SSD (ZNS SSD) [4, 12], key-value SSD (KVSSD) [14, 19, 23, 45], and computational storage [8, 11, 20, 29, 31, 33, 52]. These new types of devices are significantly diversifying the storage device landscape. In this trend, software-based storage emulators are becoming more important than ever. For instance, when academia and/or industry propose an innovative storage device concept, fully developing an actual product from the conceptual idea takes a while. Meanwhile, we can implement a new concept in an emulator and see its benefits and pitfalls while running real workloads. This can provide us invaluable insights, facilitating rapid design space exploration. Moreover, by collecting various performance metrics from the emulator, we can understand the I/O characteristics of operating systems and the applications. This information can be used to optimize both the software and hardware of the target system. Finally, each emulator has a sophisticated performance model along with many knobs that can control a certain performance characteristic of the emulated device. This can help us predict the application performance on future storage devices that exhibit different performance characteristics.

However, to the authors' best knowledge, none of the previously proposed emulators fully satisfy the requirements to be used in the modern storage environment. Many emerging device types are often optimized to their primary targeting workloads and require a customized communication model between the host and device. This requirement makes the *NVMe interface* the most preferred interface for the emerging device types due to its flexibility and extendibility. This implies that a proper storage emulator should provide a comprehensive method to customize at the NVMe interface level. However, emulating the full NVMe interface in software is challenging as the NVMe interface inherently involves the protocol defined at the hardware level. Previous work proposes to circumvent the difficulty of emulating the NVMe interface by interposing hooks in the host NVMe device driver or leveraging virtualization technologies [12, 32, 35, 55]. However, these approaches fail to present a suitable NVMe device instance that is fully functional in the diverse modern storage environments such as when the kernel is being bypassed [24, 54] or when a

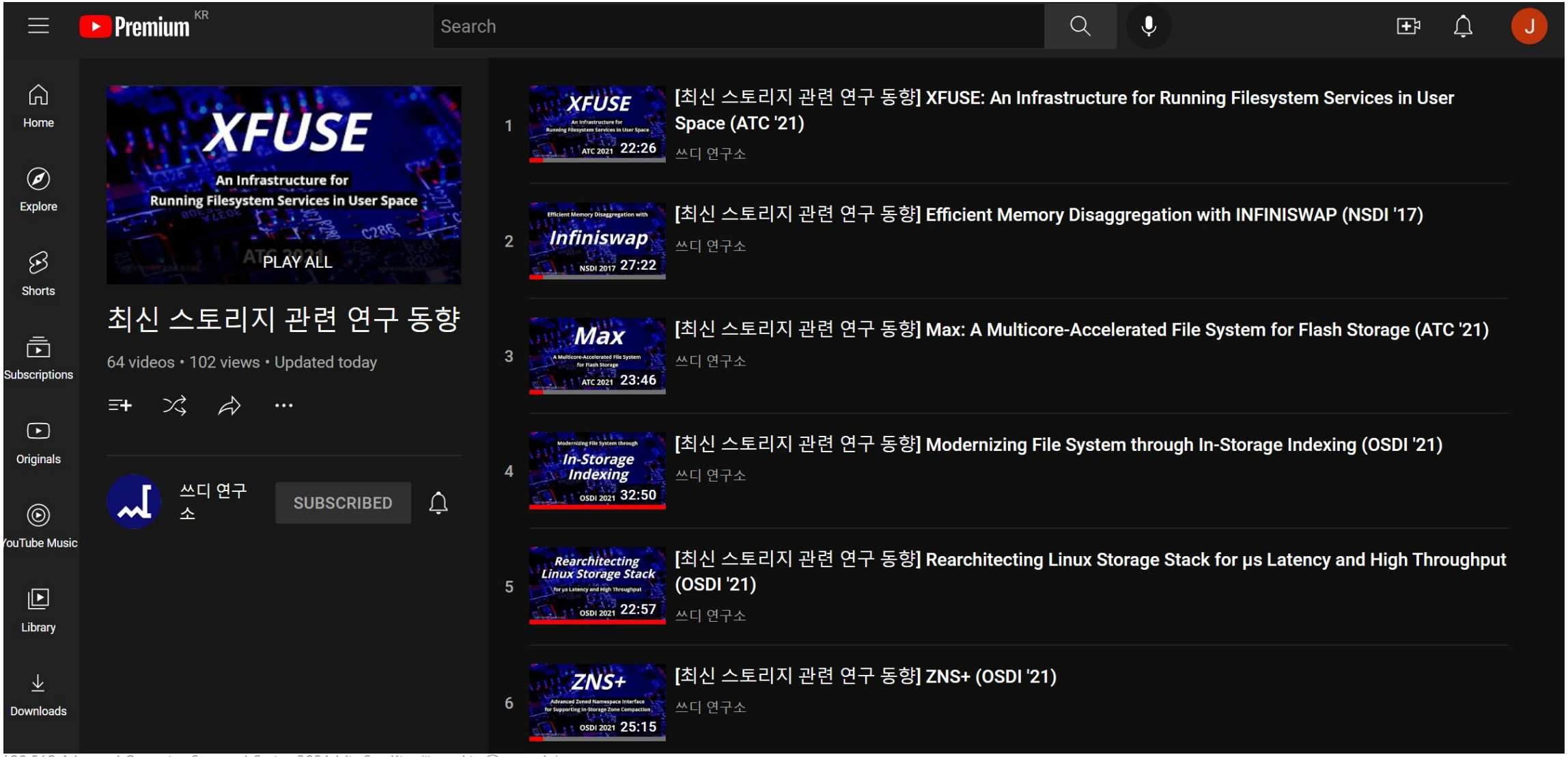
Some Project Topics

- Port NVMeVirt to other platforms (e.g., Apple M1/M2)
- Tune the performance model for real SSDs
  - Currently, NVMeVirt has performance models for Intel P4800X Optane SSD, Samsung 970 Pro SSD, Samsung KVSSD, and WD ZN640 ZNS SSD
- Propose a new NVMe interface and show its benefit
- Study application's storage access pattern
- Study applications' behavior when the storage gets faster/slower
- Propose a new storage stack for high-performance storage device

Past Project Topics

- Spring 2024 http://csl.snu.ac.kr/courses/4190.568/2024-1/#conf
- Spring 2023 http://csl.snu.ac.kr/courses/4190.568/2023-1/#conf
- Fall 2022 http://csl.snu.ac.kr/courses/4190.568/2022-2/#conf
- Fall 2021 http://csl.snu.ac.kr/courses/4190.568/2021-2/#projects
- Fall 2020 http://csl.snu.ac.kr/courses/4190.568/2020-2/#projects
- Spring 2019 http://csl.snu.ac.kr/courses/4190.568/2019-1/#conf

Get Some Idea Here, Too!



Projects: Proposal

- Due: April 17th (tentative)
- Format: 1 page, free writing
- Project proposal should include the following:
  - The motivation and the goal of your work
  - The problem you would like to solve (define clearly)
  - Brief summary of related work
  - Your ideas to solve the problem
  - Research plan for the project

- Project proposals will be reviewed by the instructor

Projects: Term Paper

- Due: June 19th (tentative)
- In ACM/IEEE conference proceedings format (two columns)
- Up to 6-page long in English

Projects: Evaluation

- Your term paper will be evaluated using the following criteria:

1. Brightness: on your motivation and idea
2. Comprehensiveness: on the survey of existing work
3. Soundness: on your methodology
4. Impressiveness: on your results
5. Your time and efforts spent on this project

Grading Policy

- Midterm exam: 30%
- Final exam: 30%
- Term project: 40%

- *Subject to change*

Reading Assignment #1

Computer Systems Research

- (I1) ⭐ Butler W. Lampson, "Hints for Computer System Design," SOSP, 1983. (The SIGOPS Hall of Fame Award '05)

- Due: Before the class on March 10th