Date : 2020-11-01

시뮬레이션. 제품을 온전히 만들지 않고, 특정 기능, 요구사항을 구현해서 생산 비용을 줄이는 것.

스토리지 시스템. 컴퓨터 구조 부터 시작된다.

boot! ↗

레지스터	Layer1	shut down. ↓	Operating System.
캐시 메모리	Layer2		File Systems.
주 기억장치	Layer3		Firmware, device. driver
보조 기억 장치.	Layer 4.		✓NV - memories. (비휘발 메모리) ↔ NV - RAM.
			SLC → TLC, PCM. 3D xpoint.

HDD → SSD로 변경시 문제점 (low-level)

기존. sata 규격에 맞게 운영체제, 하드웨어가 출시 되어 있기에, SSD는 다른 규격이 필요하였음.

기존에 운영체제. 하드웨어 에서 작동하기위해. ✓(Firmware. Controller를 만들어야함) ex) SSD controller 에서. SMART 정보. F2FS가 아닌 NTFS. EXT4 에서도 작동

application layer의 중요성

그리고 Low-level I/O. Controller를 이용 상 가능한 회피 목표, 속도차.를 가진다 함.

1. 결국. Software의 I/O가 느려짐. // SSD를 기반으로 만든 소프트웨어의 설계를 해야 하기에.

(application layer)

2. 컴퓨터 아키텍처의 구조로 인해 느려짐 등. // ~~가장 낮은 Layer인 보조 기억장치인 빠르고. 속도의 병목이 생긴다?~~ 비현실적!

보조기억장치의 I/O가 줄어들어서 해결이 안된다. ex) 병목현상.

TLC의 지연시간이 더 늘었는데, 왜 사용하나 // Controller 발전. Program cycle limit : 쓰기 횟수

SLC처럼 한번에 3bit read. 지연시간 ↓. (TLC의 성능 향상)

Fusion memory : SLC cache + TLC. 데이터 삽입시 SLC에 임시.

NV-RAM. // 비휘발성 메모리.. 즉. 보조의 경계 사라짐

데이터 크기 변화 고려. 속도 ↑. // 결국 NV RAM의 장점을 응용?, 그리고, FTL, 등 어떻게 적용?

DRAM. < PRAM < NVM.

레지스터 메모리 같은 것

(★) 파일 크기 ≠ 디스크 할당크기의 이유!

↳ 결국 파일 I/O 최소단위에 맞게 데이터의 저장이 되지 않기에.

그리고 결국. 기존의 파일 시스템이 비효율적이기에!