

# Storage Systems (StoSys)

## XM\_0092

## Lecture 1: Welcome and Introduction

Animesh Trivedi

<https://stonet-research.github.io/>

Autumn 2023, Period 1

# Welcome and who are we



**Animesh Trivedi**

<https://animeshtrivedi.github.io/>

Assistant Professor

**Interests:** Distributed Systems  
(storage, network, big data, OS)



**Zebin Ren**

Phd student

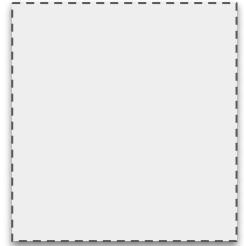
**Interests:** Storage systems,  
operating systems



**Krijn Doeckemeijer**

Phd student

**Interests:** Distributed storage,  
QoS, Networked systems



**You?**



**Matthijs Jansen**

Phd student

**Interests:** Edge computing,  
Machine learning, Data Centers



**Sudarsan Sivakumar**

MSc student

**Interests:** Systems



**Nick Tehrany**

Scientific programmer

**Interests:** (ZNS) Storage systems

# Storage-Network (Sto-Net) Research: Come join us

- (work-in-progress) <https://stonet-research.github.io/>
- **Storage research:** *How to build next generation of fast storage systems?*
  - *Persistent memories, flash storage, customized/programmable storage*
  - *Single machine, high-performance clusters (DAS-6), cloud computing*
  - *Computational storage and new APIs and abstractions*
  - *Workload/application specific storage*
- **Edge Computing :** *Storage management, resource management, and offloading*
- *Curious to know what is **CPU-free computing** ?!*
  - <https://dl.acm.org/doi/10.1145/3593856.3595906>

See more : <https://animeshtrivedi.github.io/>

**We meet every Tuesday from 2-3pm in 11A-37 (open invitation, see StoNet calendar)**

Opportunities for TAs, MSc theses, and research project courses (6 or 12 ECTS)

# Safety and diversity



The campus environment should be professional, welcoming and safe for everybody. If you see or experience something unsafe or harmful, or if you're having a difficult time:

- Reach out to [your teacher](#), or a [student diversity liaison](#).  
Even if it's not a serious issue. If we hear about it often enough, we can help.
- Visit the [student well-being point](#) or reach out to [a student psychologist](#).  
Main building: [vu.nl/en/student/student-wellbeing/student-wellbeing-point](http://vu.nl/en/student/student-wellbeing/student-wellbeing-point). Psychologists: [psychologists@vu.nl](mailto:psychologists@vu.nl)
- Department [diversity office hours](#), every 2nd Monday of the month, 12:45 (free lunch).  
Discuss problems you encounter and propose initiatives to improve things. Announced on Canvas community sites.
- Talk to your [study advisor](#).  
They will provide support, communicate problems to the department and tell you what resources are available.
- Talk to a [confidential councillor](#) (*vertrouwenspersoon*): [vertrouwenspersonen-studenten@vu.nl](mailto:vertrouwenspersonen-studenten@vu.nl).  
Guaranteed to be confidential. Usually for more serious problem, but **when in doubt, reach out**.

# Administration

This is a relatively new course - feel free to give your frank feedback!

**Project:** course has a project, read the project Handbook for details

- Group work: Start looking for your teammate now
- Attendance mandatory for the project grading sessions
  - **9-11 am Tue, 26th September 2023 and Tue, 17th Oct 2023 (lecture slots)**

**Lectures:** on related topics,

- Lectures are recorded, but the primary focus of teaching is in the class
  - *Recording is for refreshing*
- Attendance not mandatory (exceptions - project evaluation), but strongly encouraged

**Lab hours:** every Thursday from 3:30 - 5:15 pm (See canvas page for location)

See the Canvas page for more details (<https://canvas.vu.nl/courses/71163>)

# Grading and exam

## 50% exam and 50% project

- 5 milestones in the project, total worth of 50 points (+bonuses separate)
  - Max 20 points out of the bonuses
- Exam will be in-person (multiple choice questions), worth 50 points

### Project assessment

- [individual] Milestone 1 is code commit upload and a **canvas quiz**
- [Group] Milestone 2 and 4 - **code commit uploads** with tests that you should pass
- [Group] Milestone 3 and 5 are **interviews**

**Resit policy** : only available for the exam

**No Carry forward grades: Next year you will start from zero**

# Grade calculation

*You need to pass both, the project and the exam to pass the course*

- You need at least **25 / 50** to pass the project
- You need at least **25 / 50** to pass the exam

**The basic 50 marks out of 100 will be the grade of 6.0. After that it is a sliding scale**

```
Final grade = Low_ceiling {6.0 + ( (total_marks - 50) * 4 / 50)}  
// you need to cross the +0.5 stepping to claim the previous grade.
```

E.g., if you get 68/100 marks, then your grade : 7.5

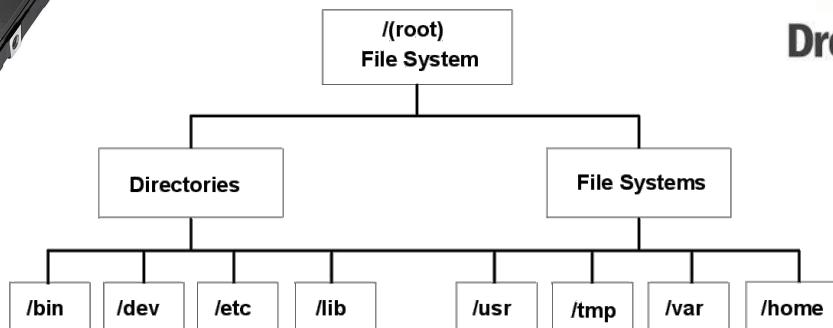
If you get 90 / 100 marks then your grade = 9.5

# Grade table (approx)

Your marks	Final grade
< 50	Fail
[ 50 - 56 )	6.0
[ 56 - 62 )	6.5
[ 62 - 68 )	7.0
[ 68 - 75 )	7.5
[ 75 - 81 )	8.0
[ 81 - 85 )	8.5
[ 85 - 90 )	9.0
[ 90 - 95 )	9.5
[ 95 - 100 ]	10.0

# **Questions about course administration?**

# What do you think of: Storage Systems



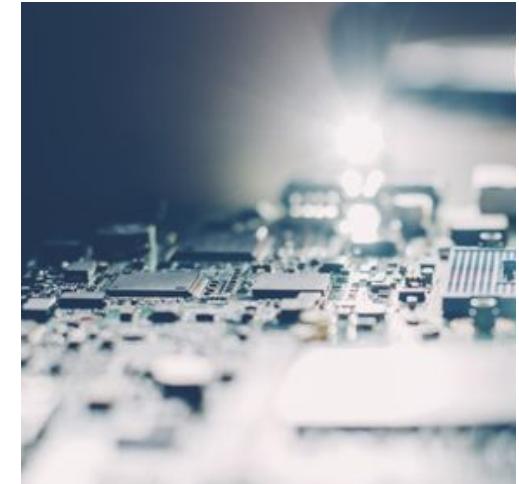
What else I am missing ?

# Storage Systems

**One of the most essential** components in the system

- *Can you imagine a computer that just does "calculations" - how exciting that would be*

**Probably** it is not the most exciting part of the system as we **were** not releasing headline-grabbing, super-duper, Gigahertz, a billion Multithreaded, hundreds of GB/s storage device on a daily basis



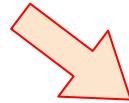
But something changed in mid-2000s that fundamentally re-shaped storage research : Rise of **Non-Volatile Storage** in mainstream/commodity computing

*At the end of this course I hope you will be as excited about it as I am*

# Data essential to our society



Financial



Warehouse



Social

Health



Mobility

ASTRON



# A minute on the Internet



# 200 Zettabytes *(by 2025)*

$200 \times 1,000,000,000,000,000,000$

A vertical scale diagram illustrating powers of ten. It features seven vertical dotted lines with upward-pointing arrows at the top. The labels are positioned as follows: 'sextillion' is at the bottom; 'quintillion' is above it; 'quadrillion' is further up; 'trillion' is near the middle; 'billion' is slightly above 'trillion'; 'million' is positioned between 'billion' and 'thousand'; and 'thousand' is at the very top.

*Scientists estimate that Earth contains 7.5 sextillion sand grains. That is 75 followed by 17 zeros. [See [here](#)]*

<https://localiq.com/blog/what-happens-in-an-internet-minute/>

<https://www.bondhighplus.com/2022/01/08/what-happen-in-an-internet-minute/>

<https://cybersecurityventures.com/the-world-will-store-200-zettabytes-of-data-by-2025/>

# If that Zettabytes does not resonate

Assume (*illustration purposes only*):

- 1 grain of rice is 1 byte of data



# If that Zettabytes does not resonate

Assume (*illustration purposes only*):

- 1 grain of rice is 1 byte of data
- **Kilobyte:** a cup of rice



# If that Zettabytes does not resonate

Assume (*illustration purposes only*):

- 1 grain of rice is 1 byte of data
- **Kilobyte**: a cup of rice
- **Megabyte**: 8 bags of rice



# If that Zettabytes does not resonate

Assume (*illustration purposes only*):

- 1 grain of rice is 1 byte of data
- **Kilobyte**: a cup of rice
- **Megabyte**: 8 bags of rice
- **Gigabyte**: 3 semi-truck



# If that Zettabytes does not resonate

Assume (*illustration purposes only*):

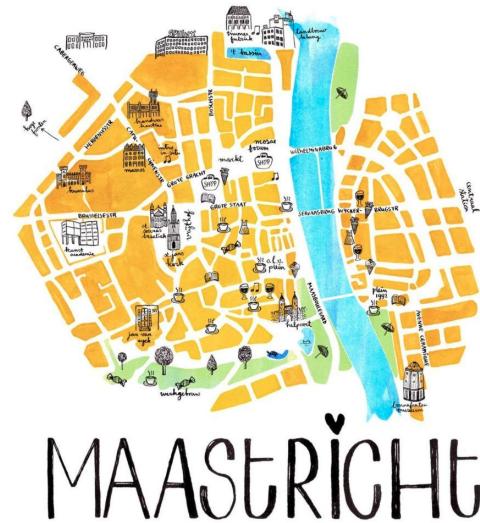
- 1 grain of rice is 1 byte of data
- **Kilobyte**: a cup of rice
- **Megabyte**: 8 bags of rice
- **Gigabyte**: 3 semi-truck
- **Terabyte**: 2 container ships



# If that Zettabytes does not resonate

Assume (*illustration purposes only*):

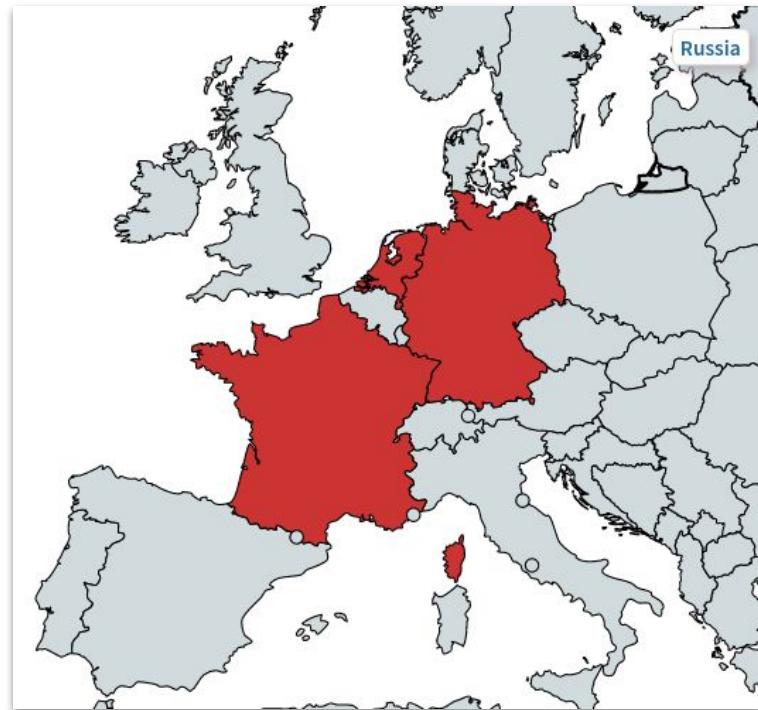
- 1 grain of rice is 1 byte of data
- **Kilobyte**: a cup of rice
- **Megabyte**: 8 bags of rice
- **Gigabyte**: 3 semi-truck
- **Terabyte**: 2 container ships
- **Petabyte**: Covers Maastricht



# If that Zettabytes does not resonate

Assume (*illustration purposes only*):

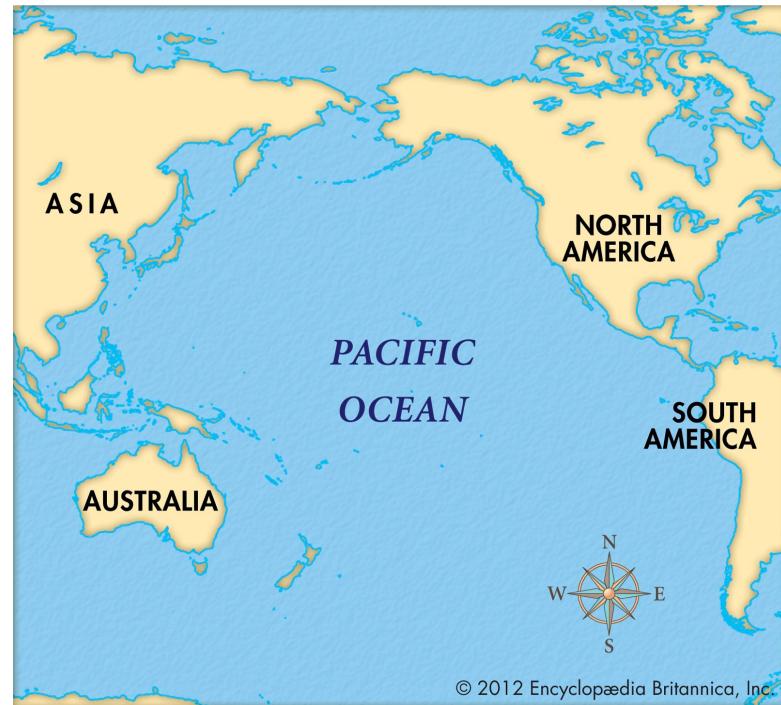
- 1 grain of rice is 1 byte of data
- **Kilobyte**: a cup of rice
- **Megabyte**: 8 bags of rice
- **Gigabyte**: 3 semi-truck
- **Terabyte**: 2 container ships
- **Petabyte**: Covers Maastricht
- **Exabyte**: Covers NL + DE + FR



# If that Zettabytes does not resonate

Assume (*illustration purposes only*):

- 1 grain of rice is 1 byte of data
- **Kilobyte**: a cup of rice
- **Megabyte**: 8 bags of rice
- **Gigabyte**: 3 semi-truck
- **Terabyte**: 2 container ships
- **Petabyte**: Covers Maastricht
- **Exabyte**: Covers NL + DE + FR
- **Zettabyte**: Fills up the Pacific Ocean



# If that Zettabytes does not resonate

Assume (*illustration purposes only*):

- 1 grain of rice is 1 byte of data
- **Kilobyte**: a cup of rice
- **Megabyte**: 8 bags of rice
- **Gigabyte**: 3 semi-truck
- **Terabyte**: 2 container ships
- **Petabyte**: Covers Maastricht
- **Exabyte**: Covers NL + DE + FR
- **Zettabyte**: Fills up the Pacific Ocean
- **Yottabytes**: An Earth size rice ball

We are here



# At the same time

Our needs for timely data analysis are increasing

- **Micro**-seconds and **Milli**-seconds analysis
- Gigabytes, Terabytes, Petabytes, Exabytes per query
- Scalability, distributed systems
- Energy and efficiency needs

*μsec-scale data center and clouds*

At the end of the day : **how fast can you read and write your data from storage devices** (you won't be able to process faster than this)

This is what we will be studying in this course

# Scope of this course

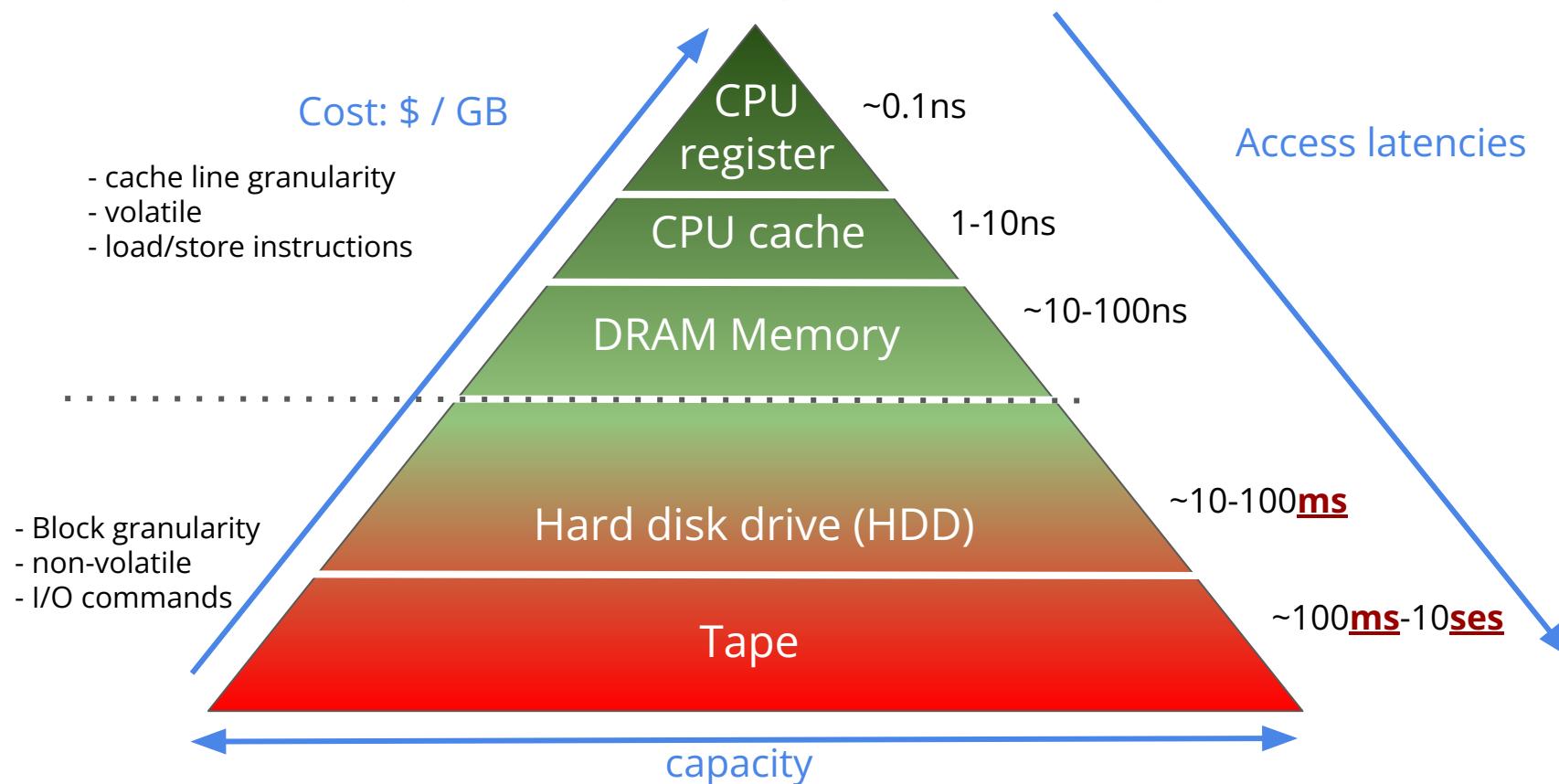
Storage systems is a **really really large topic**, covering ...

- Physics : DNA, electrons, glass, magnetism - how to encode 0s and 1s
- Architecture and devices
- Algorithms and theory (encoding, compression, error corrections)
- Distributed systems
- Fault-tolerance and failure management
- Application API and abstractions
- Quality of service
- Modeling, cost, and energy
- ...

In this course, we focus on the *major innovations* (*a biased selection by me, naturally*) that are happening in the area of data and storage management due to the advent of NVM storage

# Questions?

# The triangle of storage hierarchy



# Recap: The hard disk drive (HDD) world

Invented by IBM in 1956, in the market in the 1960s

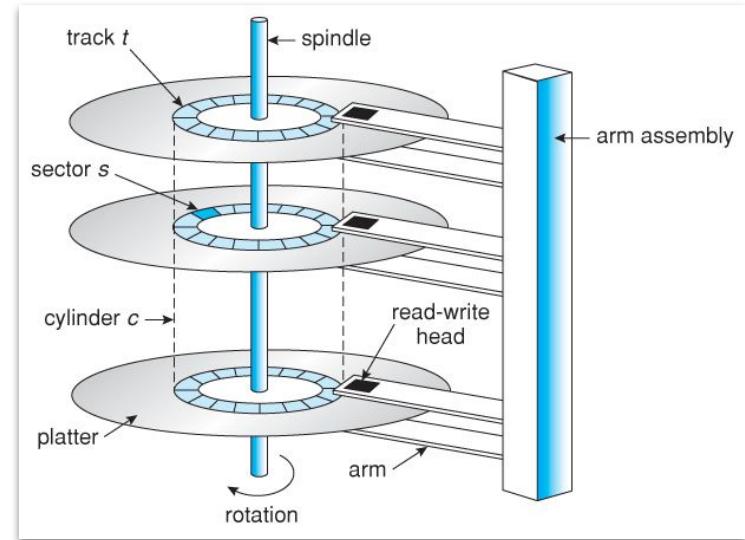
Uses ferromagnetic material to store bits

There are **sectors** (512 bytes) arranged within a **track**, on **platters**

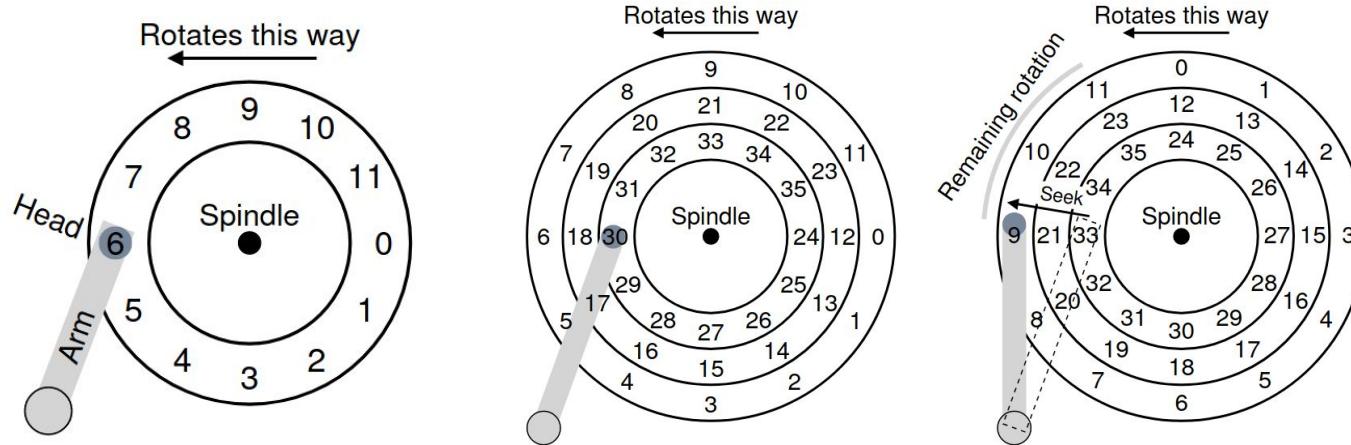
A rotational head positions the sensor on the right sector to read data (7,000 - 15,000 RPM)

On paper, a sector can be read/written many times

Has been the primary technology to persistently store data for almost 50 years



# HDD disk I/O basics



**Transfer time = rotation + seek + read/write time**

**Typically:**

- Sequential I/O fast, adjacent I/O fast
- Milliseconds for latency
- ~100s MB/sec bandwidth (sequential)

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects via	SCSI	SATA

# HDD design mantras

1. I/O happens in a sector granularity (512 bytes)
2. Read and write are symmetric - they both have the same performance
3. Random performance (both read/write) are worse than the sequential performance
4. Small I/O performance is bad - cannot amortize the seek and rotational time
5. [Not so well known] Outer vs inner track performance
  - a. Outer tracks rotate with a faster linear speed than Inner tracks
  - b. Hence, faster bandwidth (30+) and lower latencies
  - c. See section 4.3 <http://cseweb.ucsd.edu/~gporter/papers/tritonsort-nsdi11.pdf>
6. (Theoretically) A sector can be read/written infinitely many times
  - a. Disregarding heating, damages, demagnetization, etc.
  - b. For more details, *An Analysis of Data Corruption in the Storage Stack* (USENIX FAST 2008) and *Failure Trends in a Large Disk Drive Population* (USENIX FAST 2007)

# Linux Tools: HDD information

```
atr@evelyn:~$ lsblk -I 8
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda        8:0    0 238.5G  0 disk 
└─sda1     8:1    0   731M  0 part /boot
└─sda2     8:2    0    1K  0 part 
└─sda5     8:5    0 237.8G  0 part 
  └─sda5_crypt 253:0  0 237.8G  0 crypt 
    ├─ubuntu--vg-root 253:1  0 236.8G  0 lvm   /
    └─ubuntu--vg-swap_1 253:2  0   980M  0 lvm   [SWAP]
```

```
atr@evelyn:~$ sudo hdparm -g /dev/sda
/dev/sda:
geometry      = 31130/255/63, sectors = 500118192, start = 0 .
```

Cylinders / heads / sectors, the size (in sectors)

```
atr@evelyn:~$ sudo hdparm -I /dev/sda
[sudo] password for atr:
/dev/sda:

ATA device, with non-removable media
  Model Number: SAMSUNG MZ7TE256HMHP-000L7
  Serial Number: S1K7NSAG234139
  Firmware Revision: EXT09L6Q
  Transport: Serial, ATA8-AST, SATA 1.0a, SATA II Ext
Standards:
  Used: unknown (minor revision code 0x0039)
  Supported: 9 8 7 6 5
  Likely used: 9
Configuration:
  Logical      max      current
  cylinders    16383    16383
  heads        16       16
  sectors/track 63       63
  --
  CHS current addressable sectors:   16514064
  LBA   user addressable sectors:   268435455
  LBA48 user addressable sectors: 500118192
  Logical Sector size:                 512 bytes
  Physical Sector size:                512 bytes
  Logical Sector-0 offset:              0 bytes
  device size with M = 1024*1024:    244198 MBytes
  device size with M = 1000*1000:    256060 MBytes (256 GB)
  cache/buffer size = unknown
```

<https://www.kernel.org/doc/Documentation/admin-guide/devices.txt>

<https://linux.die.net/man/8/hdparm>

# HDD: Performance improvements over the years

	1983	2011	Improved
CPU Speed	1x10Mhz	4x3GHz	> 1,000x
Memory Size	$\leq$ 2MB	8GB	$\geq$ 4,000x
Disk Capacity	$\leq$ 30MB	2TB	> 60,000x
Net Bwidth	3Mbps	10Gbps	> 3,000x

Improvements in systems performance

- **CPU** - yes
- **Memory** - yes
- **Network** - yes
- **Disk** - yes

# HDD: Performance improvements over the years

	1983	2011	Improved
CPU Speed	1x10Mhz	4x3GHz	> 1,000x
Memory Size	$\leq$ 2MB	8GB	$\geq$ 4,000x
Disk Capacity	$\leq$ 30MB	2TB	> 60,000x
Net Bwidth	3Mbps	10Gbps	> 3,000x

Improvements in systems performance

- CPU - yes
- Memory - yes
- Network - yes
- Disk - yes (*is it?*)

	Mid-1980s	2009	Improvement
Disk capacity	30 MB	500 GB	16667x
Maximum transfer rate	2 MB/s	100 MB/s	50x
Latency (seek + rotate)	20 ms	10 ms	2x
Capacity/bandwidth (large blocks)	15 s	5000 s	333x worse
Capacity/bandwidth (1KB blocks)	600 s	58 days	8333x worse
Jim Gray's Rule [11] (1KB blocks)	5 min.	30 hours	360x worse

Bw bound

Latency bound

Cost/access bound

<https://web.stanford.edu/~ouster/cgi-bin/papers/ramcloud.pdf>

[http://www.scs.stanford.edu/~rumble/papers/latency\\_hotos11.pdf](http://www.scs.stanford.edu/~rumble/papers/latency_hotos11.pdf)

Jim Gray's Five Minute Rule: [https://en.wikipedia.org/wiki/Five-minute\\_rule](https://en.wikipedia.org/wiki/Five-minute_rule)

# HDD : Achilles heels of systems building

**Latency limits** : can not rotate faster than 10s thousands / sec

- Cannot drop latency below a few milliseconds
- Various caching, prefetching techniques in DRAM



**Random performance** very poor (a few IO (~100s) operations/sec, 10 MB/sec)

- Buffering operation and write them in a single large request to amortize rotational and seek latencies

**Fundamentally:** all these limitations are physical limits (cannot be improved)

- *There is an arm sensor movement*
- *There is a rotation of spindle*

***Imagine a storage without any movements***

# Enter: USB drive (*what is inside a USB drive*)

✓ Voor 23.59 uur besteld, morgen gratis bezorgd ✓ Gratis ruilen ✓ 14 échte winkels ✓ Onze app ✓ Beste webwinkel 2022

Extern geheugen USB sticks Top 10 Samsung Sandisk Kingston Geheugenkaarten Externe harde schijven

Terug Geheugen & opslag / USB sticks / Sandisk USB sticks / iXpand GO Flash drive 3.0 128GB

## SanDisk iXpand GO Flash drive 3.0 128GB

★★★★★ 8,9/10 (50 reviews) Toon alle accessoires Sandisk USB sticks



Onze keuze voor een usb stick met Lightning aansluiting

Comparing Keuze Toon meer



Vergelijkbare en betere producten Productinformatie Aanbevolen combinatie Accessoires Reviews Specificaties

<https://www.coolblue.nl/product/856964/sandisk-ixpand-go-flash-drive-3-0-128gb.html>

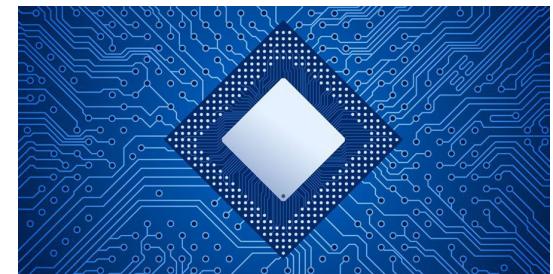
# Non-Volatile Memory (NVM) Technology

An umbrella term for many related technologies

(Solid-State Storage, Storage-Class Memory - *we are rubbish with names* 😞)

The idea is to use physical properties of a media to store bits and data

- [HDD] Magnetic state
- **Electrical charge** ←
- Crystalline / amorphous state
- Resistance
- Optical
  - Project Silica, <https://www.microsoft.com/en-us/research/project/project-silica/>



**No moving parts !**

# Caution / Confusion

What is a **memory** and what is a **storage**

Definition is a bit more fluid these days



- There are technologies which might have “memory” as a keyword in their name
  - You can use them to make a RAM as well (but will be slow, not efficient)
- Then there is how they are packed and used with operating systems
  - **Memory** - something that can be directly addressed from the CPU (load/store)
  - **Storage** - integrated with a block-level data transfer protocol to DRAM

*Focus on understanding the key concepts, without getting dragged with names*

# Flash Memory

Invented by Toshiba

- Dr. Fujio Masuoka (1984)
- Since the mid-1980s commercially available

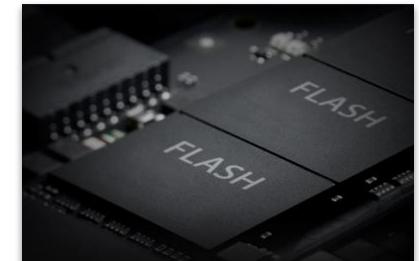
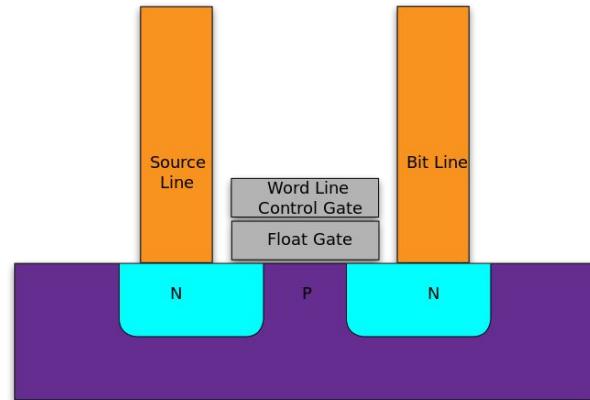
Has been around for almost 40 years

First became popular with embedded devices

In the mainstream computing (storage) from the mid-2000s

One of the most popular technologies to store data today

A very fundamental shift how to store and manage data



<https://www.eweek.com/storage/why-flash-storage-will-remain-dominant-for-several-years>

[https://en.wikipedia.org/wiki/File:Flash\\_cell\\_structure.svg](https://en.wikipedia.org/wiki/File:Flash_cell_structure.svg)

# Flash Storage - Storing data, bits by bits

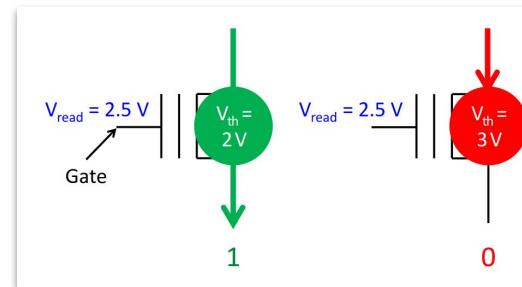
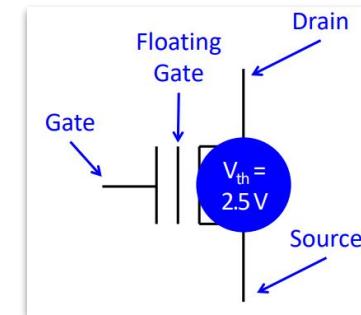
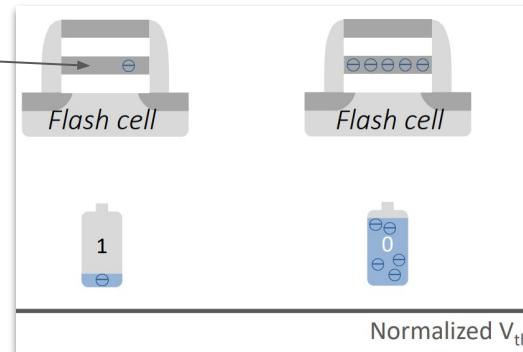
The *isolated* Floating Gate controls current flowing through the flash cell transistor

Electrons are trapped in the FG - giving it a threshold voltage ( $V_{th}$ )

Apply a known reference voltage ( $V_{read}$ ) at the gate to find out if there are electrons trapped in the FG

- Pass through = 1
- Blocked = 0

Hence, a single cell can store a zero or one



# Flash Storage - Storing data, bits by bits

Flash storage cell can be Single-Level (**SLC**), Multi-level (**MLC**) or Triple-Level (**TLC**)

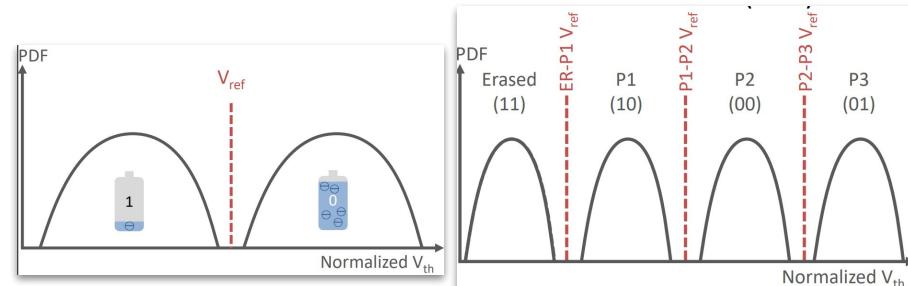
- SLC = 0, presence = 1 (2 values)
- MLC = {00, 01, 10, 11} (4 values)
- TLC = {000, 001...111} (8 values)
- QLC, PLC... (density improvements)

SLC cells have the highest performance

Multi-bit cells have the highest density, lower \$/byte cost (mass storage)

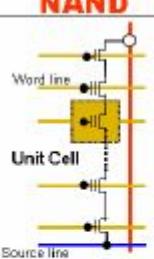
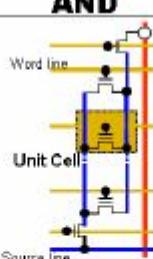
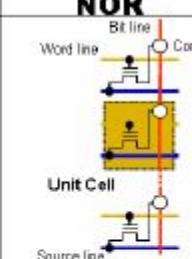
A FG must be re-programmed / erase (P/E) continuously, leading to an erosion of the isolated oxide layer - **finite P/E life cycle**

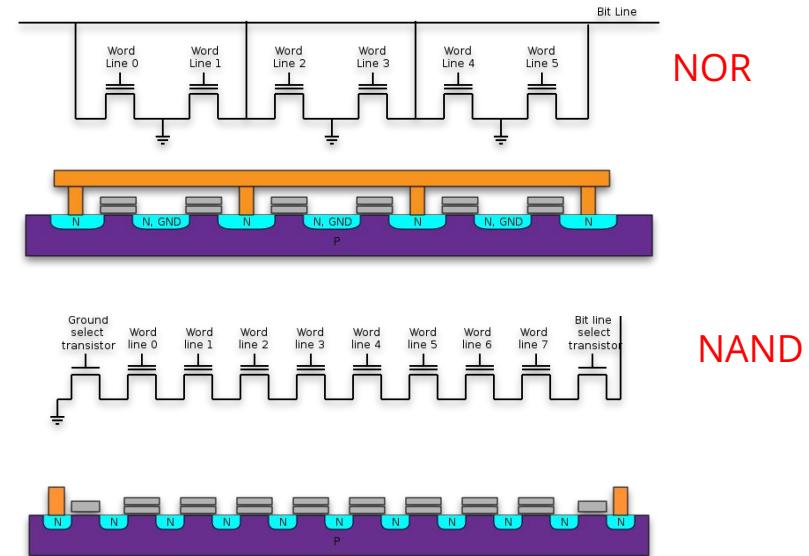
Multiple reads over the time can also erode the charge : **read disturbance**



# NAND, NOR, and AND Flash cells

There are multiple ways in which you can pack your cell calling the right trade-off between density, addressability, read and write performance, no. of connections, etc.

	<b>NAND</b>	<b>AND</b>	<b>NOR</b>
Cell Array			
Layout			
Cross-section			
Cell size	<b>4F<sup>2</sup></b>	<b>8F<sup>2</sup></b>	<b>10F<sup>2</sup></b>



<https://www.edn.com/flash-memory-101-an-introduction-to-nand-flash/>

NAND Flash 101: <https://www.simms.co.uk/Uploads/Resources/50/f4366381-b992-425a-bec4-cca409b51a6c.pdf>

# NAND and NOR

Historically there was a debate which type will win: **who won?**

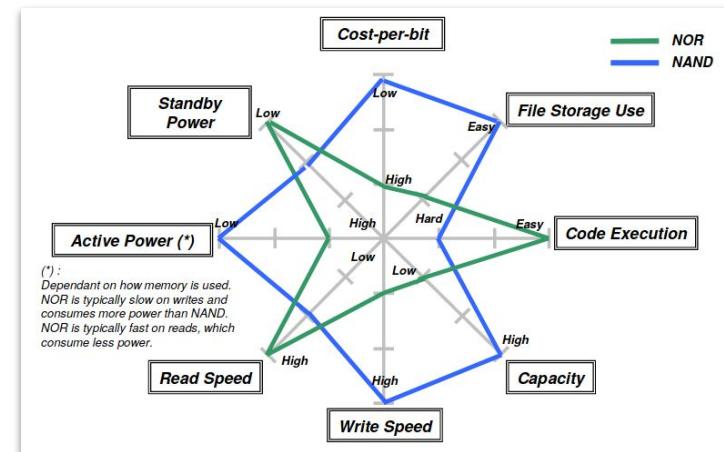
**NOR Flash** cells are often used in BIOS ROMs, firmwares for code execution

- High erase cost, but byte-addressable

**NAND flash** cells are packed in a high density manner, used for mass storage

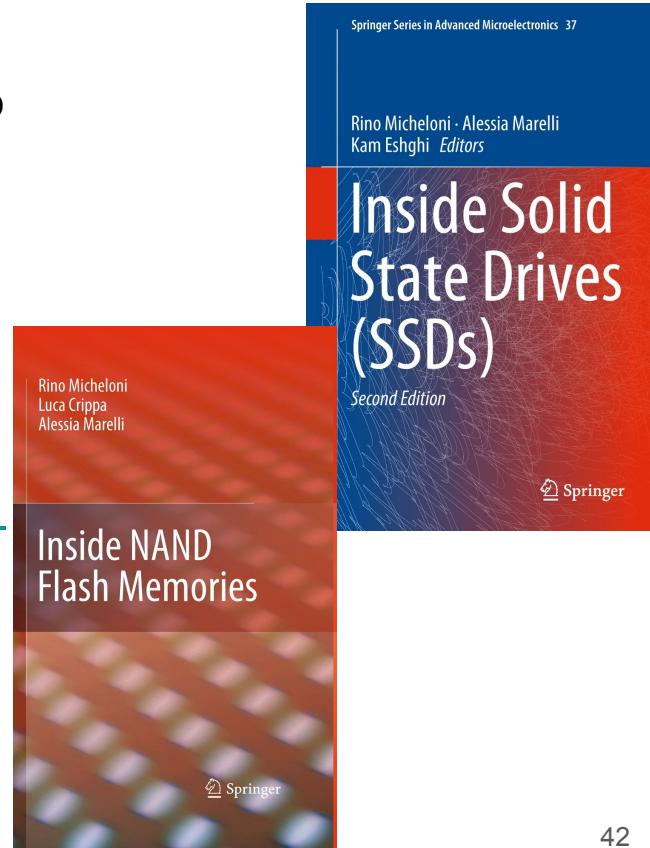
- High density, but block addressable

	SLC NAND Flash (x8)	MLC NAND Flash (x8)	MLC NOR Flash (x16)
Density	512 Mbits <sup>1</sup> – 4 Gbits <sup>2</sup>	1Gbit to 16Gbit	16Mbit to 1Gbit
Read Speed	24 MB/s <sup>3</sup>	18.6 MB/s	103MB/s
Write Speed	8.0 MB/s	2.4 MB/s	0.47 MB/s
Erase Time	2.0 mSec	2.0mSec	900mSec
Interface	I/O – indirect access	I/O – indirect access	Random access
Application	Program/Data mass storage	Program/Data mass storage	eXecuteInPlace



# For further low-level details...

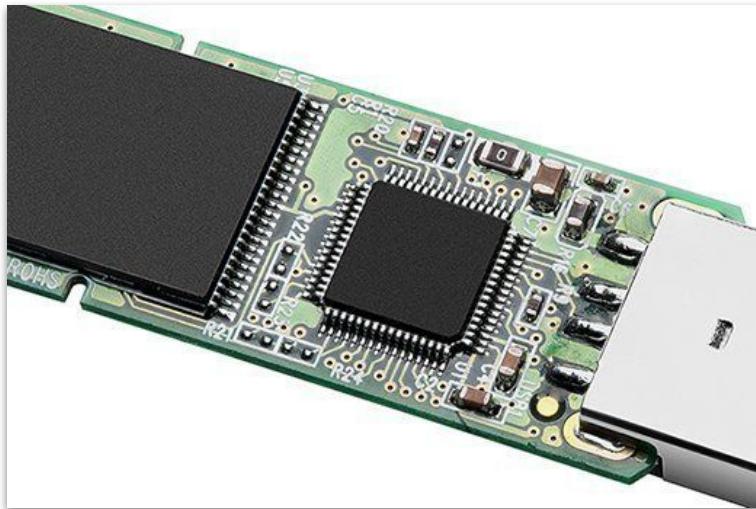
- Springer, Inside Solid State Drives (SSDs), Editors: Rino Micheloni, Alessia Marelli, Kam Eshghi, 2018,  
<https://link.springer.com/book/10.1007/978-981-13-0599-3>
- Springer, Inside NAND Flash Memories, Authors: Rino Micheloni , Luca Crippa , Alessia Marelli, 2010,  
<https://link-springer-com.vu-nl.idm.oclc.org/book/10.1007/978-90-481-9431-5>



*These should be accessible within the VU network*

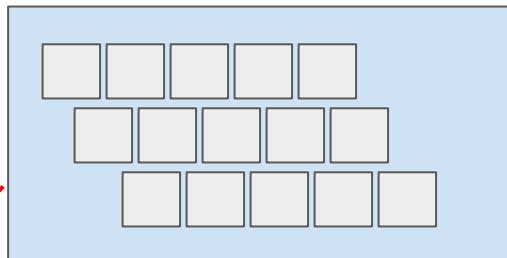
# Questions?

# So how things are arranged inside a flash drive?



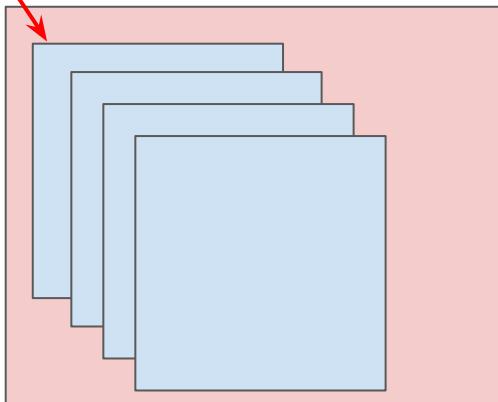
*So, why is this question important?*

# Making flash pages and blocks from flash cells



A bunch of flash cells are packed in a **page** : typically 4kB

- Typically, a unit of I/O - the page (similar to a sector size in HDD)
  - Note: different vendors/systems might use different names like sector or even block (in the project Handbook: page)



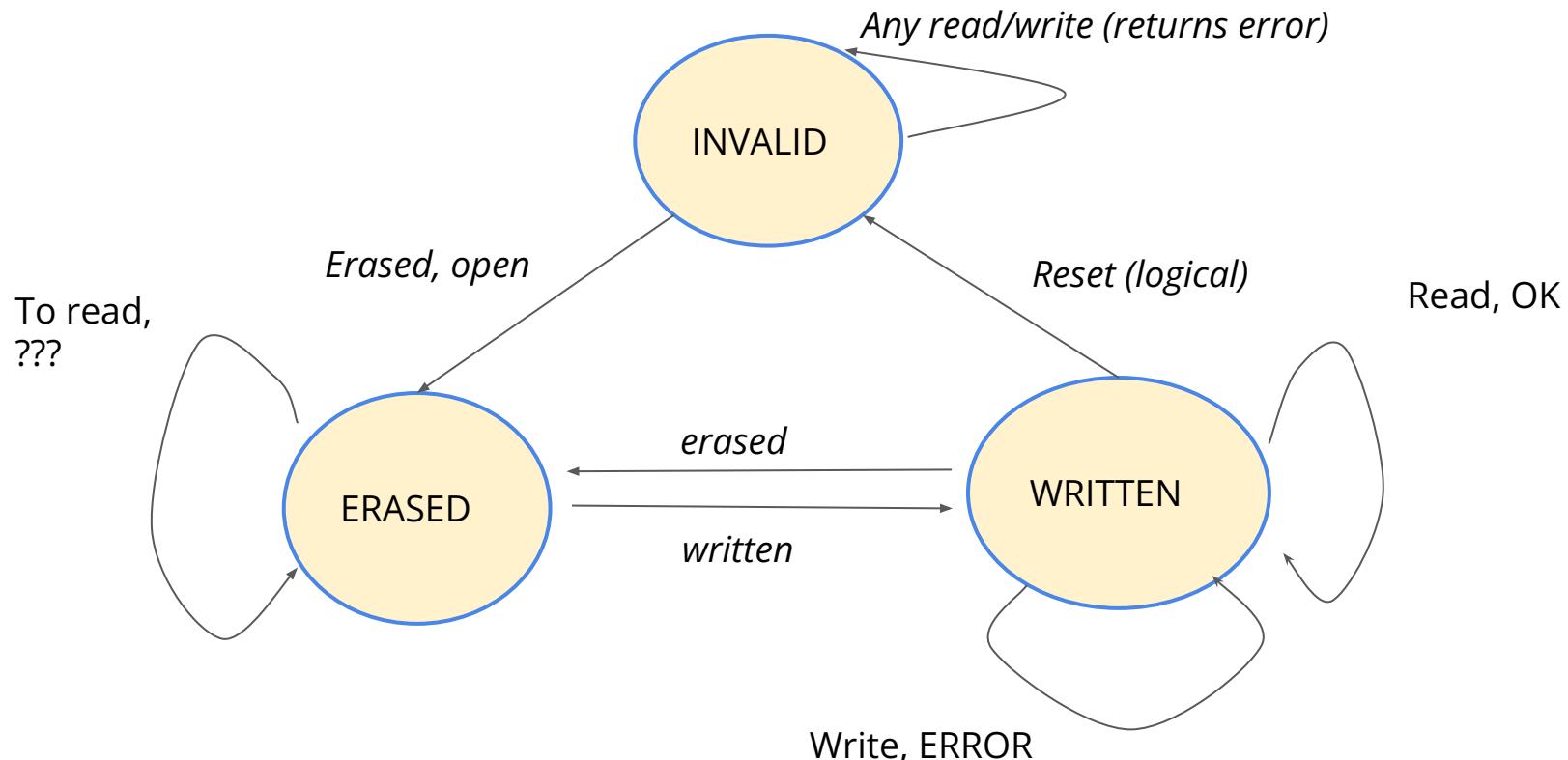
A bunch of flash pages packed together as a **block**: typically 64-128-256 pages (hence, a few Megabytes)

- Typically, a unit of erase (bulk erasure)
- **Recall**: you need to erase before you can write on a flash cell
  - Note: different vendors/systems might use different names like chunk (in the project Handbook: Zone)

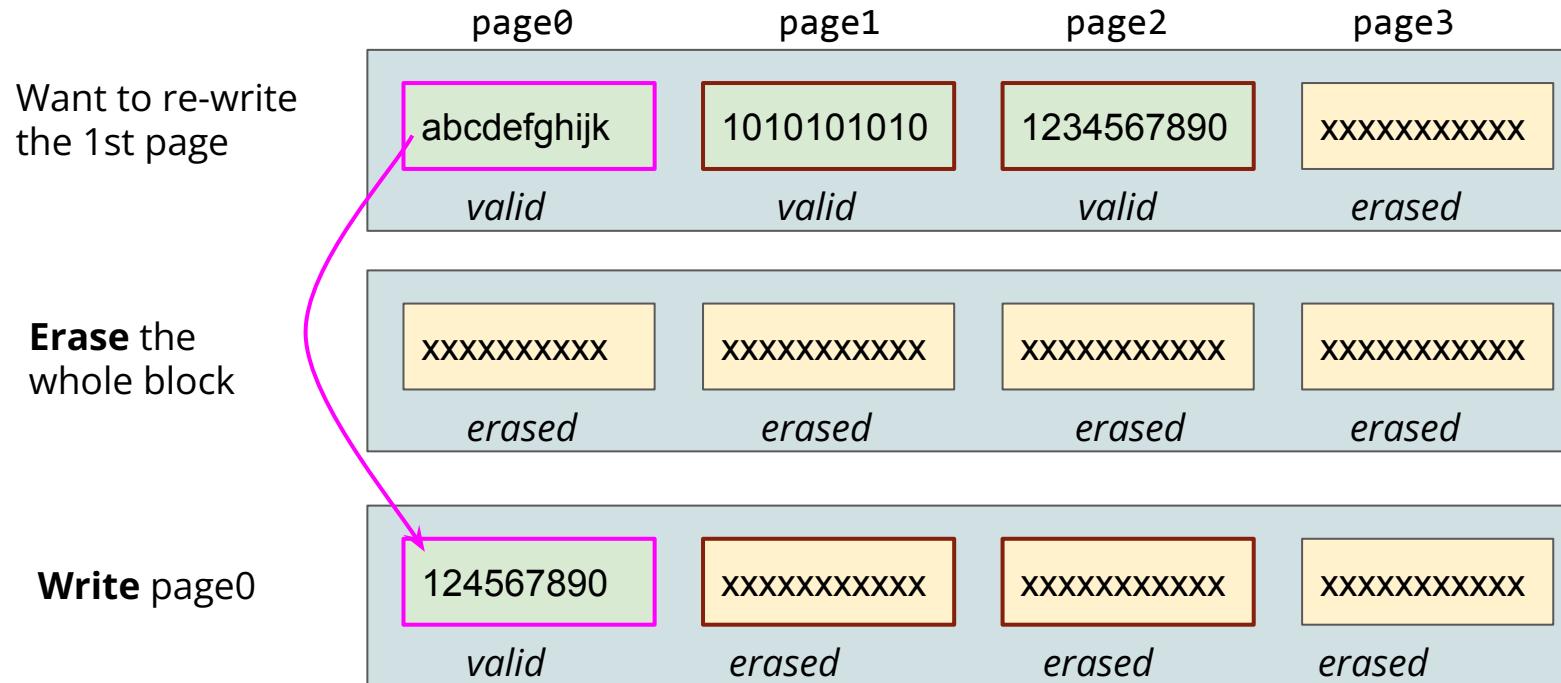
# Flash page: Basic operations

1. **Read (a page):** logically the same as reading a sector, `read(page_number)`
  - a. Internally just applying voltage and reading values
  - b. No moving parts, hence, very fast 10s of microseconds
  - c. Does it matter you are reading 1st or 1024th page? NO: hence, *uniform fast, random reads performance*
2. **Erase (a block):** reset a block (a few Megabytes)
  - a. Copy the content (if any) that you want to save to another location
  - b. Erasure the whole block (Program-Erase cycle)
  - c. Typically slow, a few to 10-100s of milliseconds
3. **Write (a page):** Once a block has been erased, `write(page_number)`
  - a. Can only write in an incremental page order inside a block
  - b. Less expensive than a erase, but more expensive than a read - typically, 50-100 microseconds
  - c. Once written, can not overwrite before an erase
  - d. Can have different read/write granularity based on the packaging

# Simplified flash page state machine (impl. dependent)



# Write - Erase Operation in Detail



Note: the content of page1 and page2 are lost, hence, if you want to save them then you have to copy out the whole block - we will come back to this problem later again

# Typical Values of Operations

*The latest and the greatest devices will do better*

	<b>Read (usec)</b>	<b>Write (usec)</b>	<b>Erase (usec)</b>	<b>P/E cycles</b>
<b>SLC</b>	25	200-300	1,500-2,000	~100,000
<b>MLC</b>	50	~600-900	~3,000	~10,000
<b>TLC</b>	75	~900-1,350	~4,500	~5,000

These are typical values for a **flash physical chips**. Newer chips have better performance. So, a value that you might see in your operating system (**flash storage device** level) depends upon :

- The technical generation of the device
- What cell technology it is using
- How old is it (?)
- How much data you have written to it (*? yes you read it right*)
- and a few more parameters...

# So far: Flash vs HDD (at the cell level)

1. No moving parts inside a flash storage
2. Flash read latencies are much lower than HDDs (10s of msecs vs 10s of usecs)
3. Asymmetric read and write performances, 10 (r) vs 100 (w) useconds
  - a. HDD have a symmetric performance for operations
4. No overwriting or in-place updates on the same page again (HDDs can do that)
5. [new operation] Erase operation, must erase a whole block before writing
6. [new activity] If there are valid data pages, must copy data before erase
7. Finite number of times one can erase and program flash cells
  - a. Nonetheless, like with any electron based storage, they leak and data will be eventually lost
8. Read performance is uniform - no difference with random or sequential **at the cell level** (*what you get at the device level, we will see*)
  - a. Write - **what do you think?**
9. Anything else that you could think of ?

# How do we make a device out of Flash cells?

We have seen how an individual cells/pages behave

How do we pack them as single “device” so that:

- They have sectors
- They have addressable sectors
- We can issue a read or write command
- We can attach to our computers / servers
- We can install a file system on it
- Who will do erase
- How do we rewrite the same “location” again



# Milestone work - a must read

Most details from this lecture come from this paper. Though values might have changed the concepts, concerns, and trade-offs remain true still today.

**Design Tradeoffs for SSD Performance**

Nitin Agrawal\*, Vijayan Prabhakaran, Ted Wobber,  
John D. Davis, Mark Manasse, Rina Panigrahy  
*Microsoft Research, Silicon Valley*  
*\*University of Wisconsin-Madison*

**Abstract**  
*Solid-state disks (SSDs) have the potential to revolutionize the storage system landscape. However, there is little published work about their internal organization or the design choices that SSD manufacturers face in pursuit of optimal performance. This paper presents a taxonomy of such design choices and analyzes the likely performance of various configurations using a trace-driven simulator and workload traces extracted from real systems. We find that SSD performance and lifetime is highly workload-sensitive, and that complex systems problems that normally appear higher in the storage stack, or even in distributed systems, are relevant to device firmware.*

**1 Introduction**  
The advent of the NAND-flash based solid-state storage device (SSD) is certain to represent a sea change in the architecture of computer storage subsystems. These devices are capable of producing not only exceptional

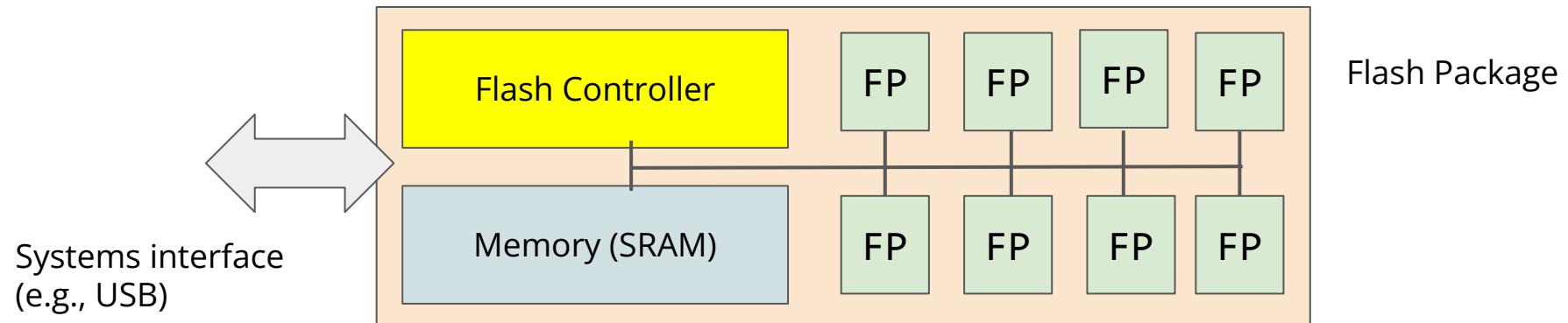
their design. Where such knowledge exists, it typically remains the intellectual property of SSD manufacturers. As a consequence, it is difficult to understand the architecture of a given device, and harder still to interpret its performance characteristics.

In this paper, we lay out a range of design tradeoffs that are relevant to NAND-flash solid-state storage. We then analyze several of these tradeoffs using a trace-based disk simulator that we have customized to characterize different SSD organizations. Since we can only speculate about the detailed internals of existing SSDs, we base our simulator on the specified properties of NAND-flash chips. Our analysis is driven by various traces captured from running systems such as a full-scale TPC-C benchmark, an Exchange server workload, and various standard file system benchmarks.

We find that many of the issues that arise in SSD design appear to mimic problems that have previously appeared higher in the storage stack. In solving these hard problems, there is considerable latitude for design

[https://www.usenix.org/legacy/events/usenix08/tech/full\\_papers/agrawal/agrawal.pdf](https://www.usenix.org/legacy/events/usenix08/tech/full_papers/agrawal/agrawal.pdf)

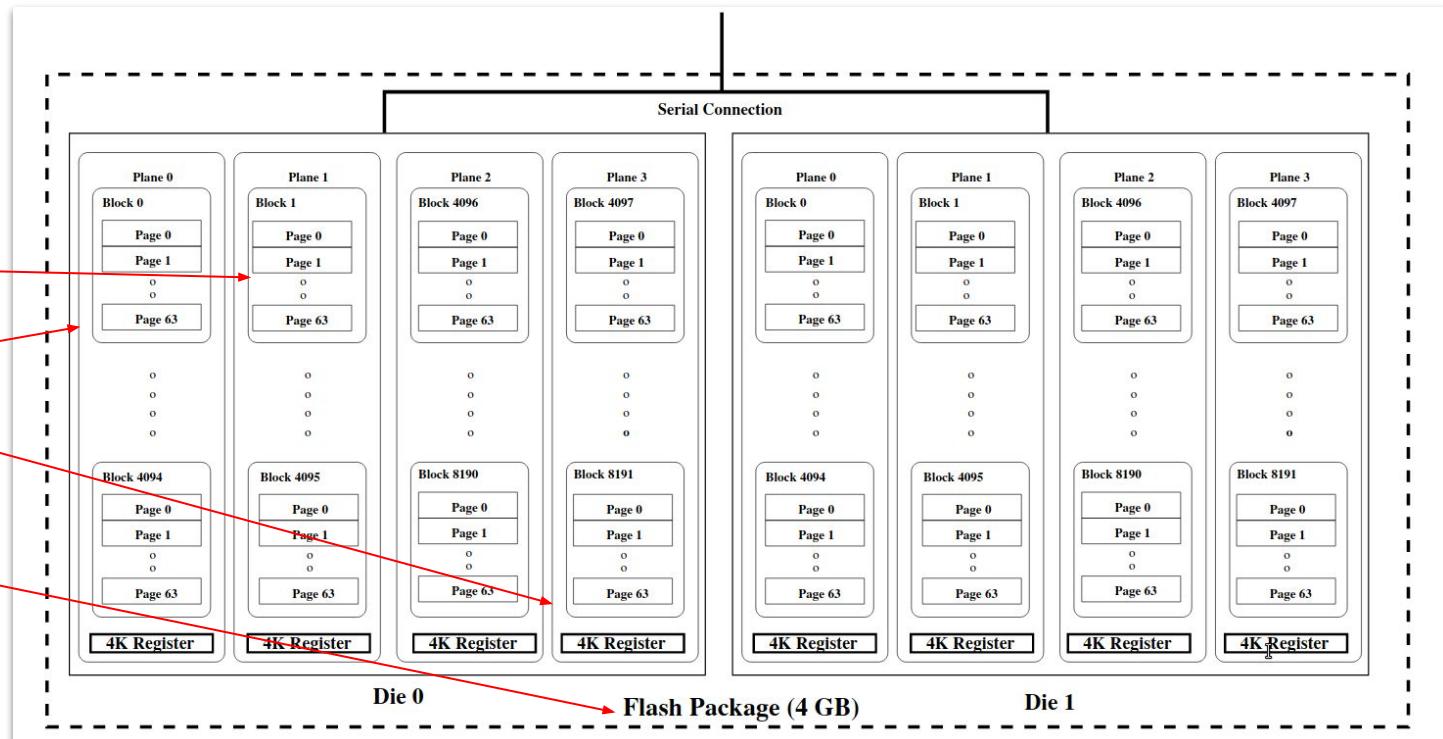
# What is inside the device



- Flash chips are arranged with a connector (could be any topology, but here a line)
- Flash controller: a microcontroller, programs the flash chips
- Memory : fast, buffer memory to hold data in transit
- System interface: can support, SATA, SAS, USB, PCIe (we will see)

# Flash Package (FP) Layout

Pages in blocks  
Blocks in planes  
Planes in dies  
Dies in packages



# Flash Package (FP) layout

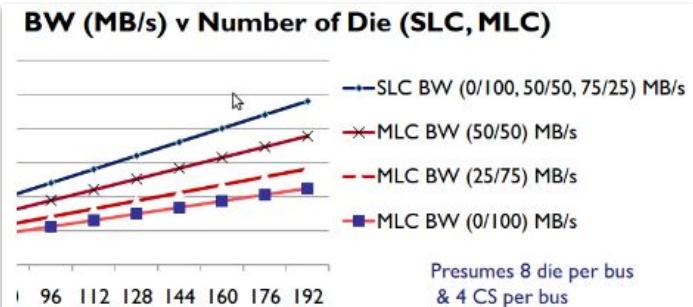
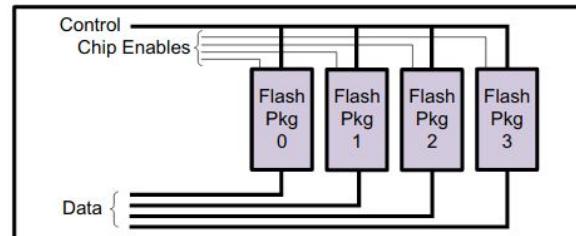
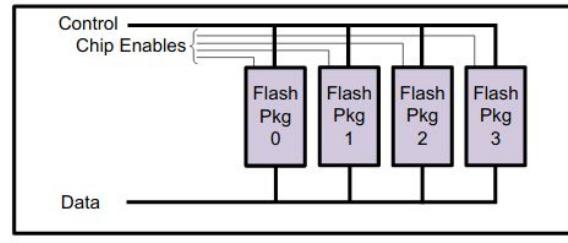
Different packages might serve I/O in parallel

Different packages might serve control in parallel

Different planes may work in parallel with a single command addressability

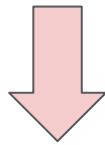
- Can move from plane 0 to 1, but not 2 (without buffering in between)
- Depends upon in the internal circuitry
  - Cost function

Multiple dies can be packaged together to push bandwidth



# The Read/Write interface?

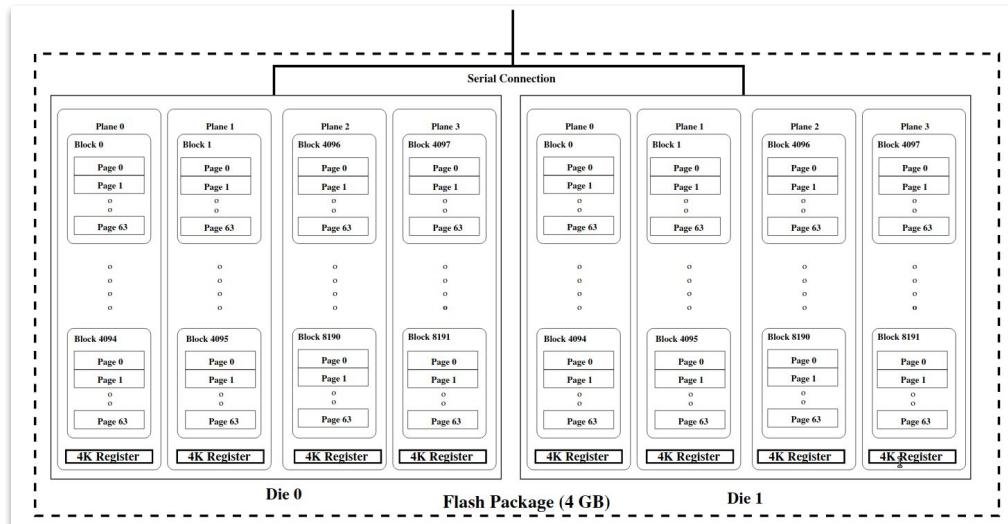
Write ( $p1, d0, b52, p211$ )?



Read ( $p1, d42, b13, p521$ )?

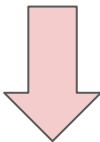


/dev/ssd0



# The Read/Write interface?

Write ( $p1, d0, b52, p211$ )?



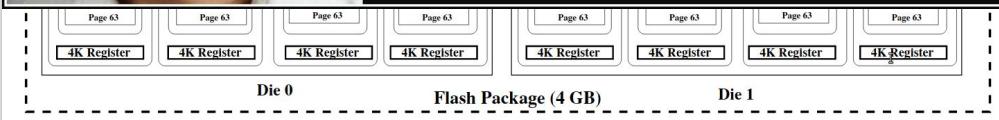
Read ( $p1, d42, b13, p521$ )?



All problems in Computer Science  
can be solved by another level of  
indirection.

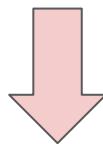
— *Butler Lampson* —

AZ QUOTES



# The Read/Write interface?

Write ( $p1, d0, b52, p211$ )?



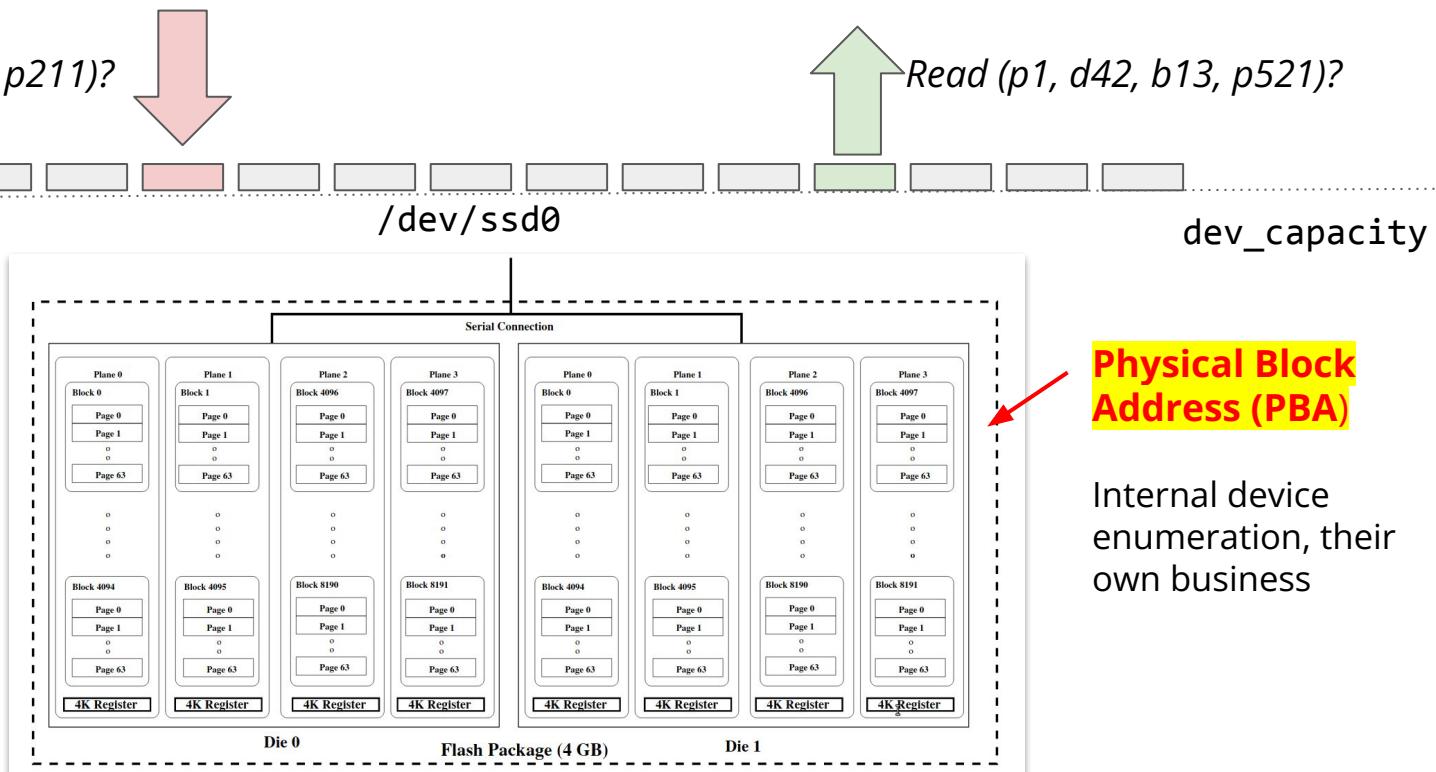
Read ( $p1, d42, b13, p521$ )?

0

**Logical Block Address (LBA)**

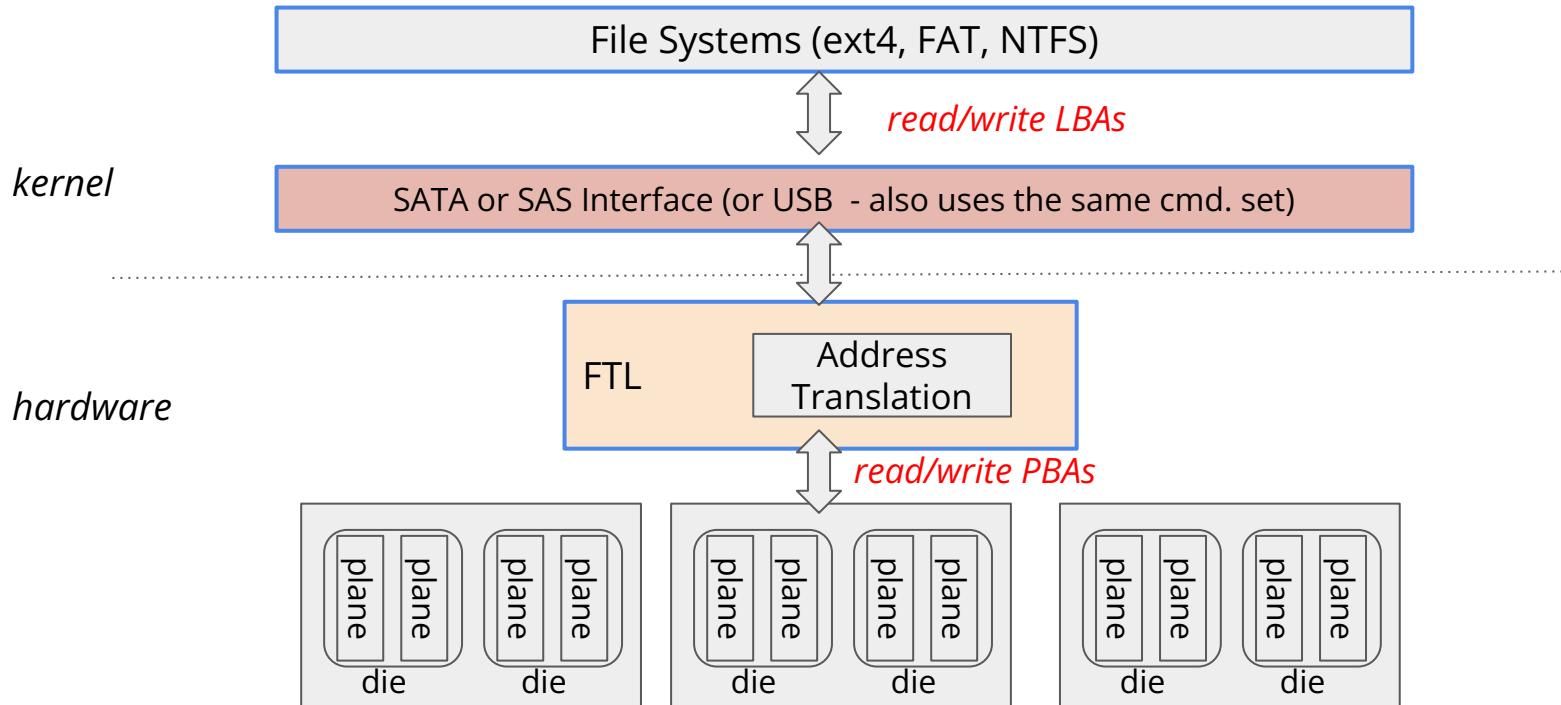
(almost the same as the sector, but different size - page size typically)

Internally we can map from LBA to physical locations

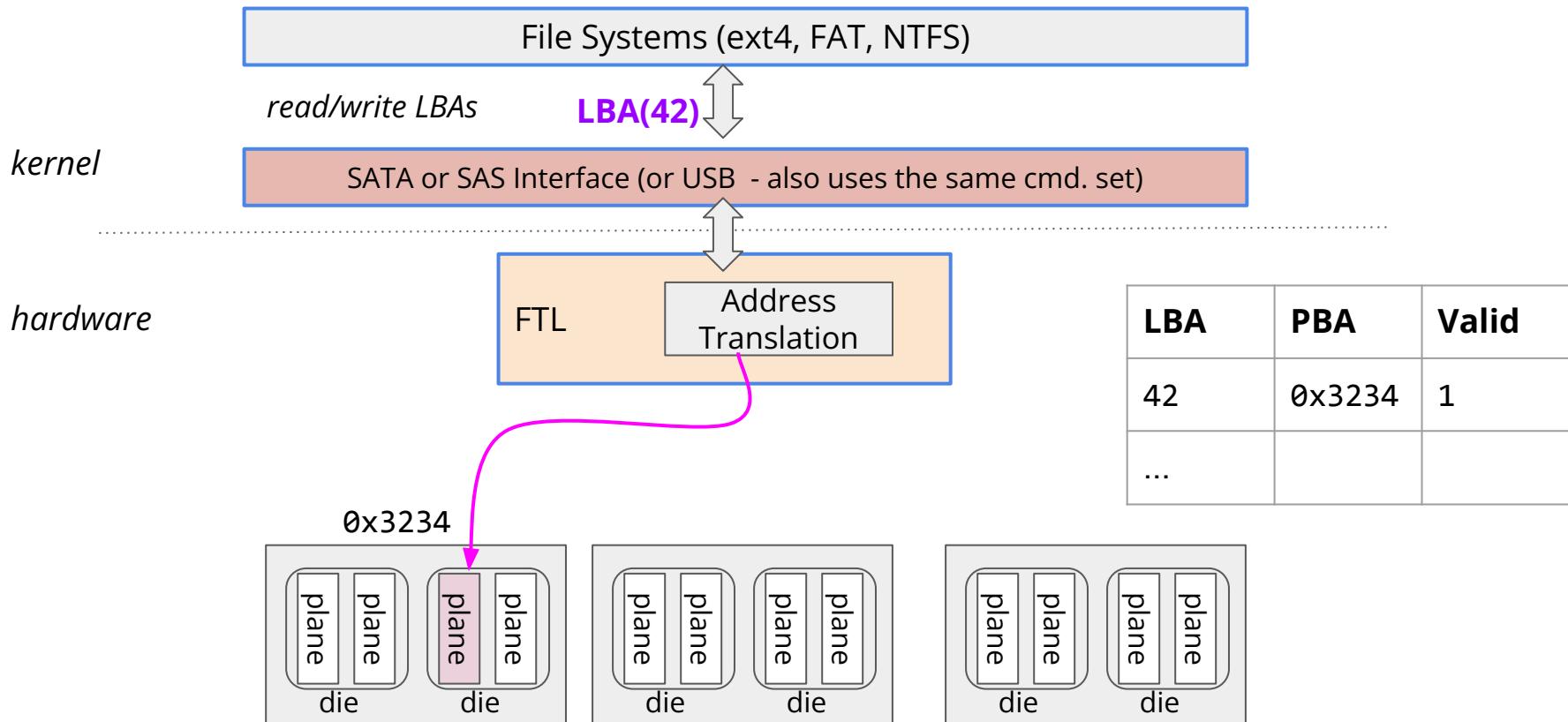


# Flash Translation Layer (FTL)

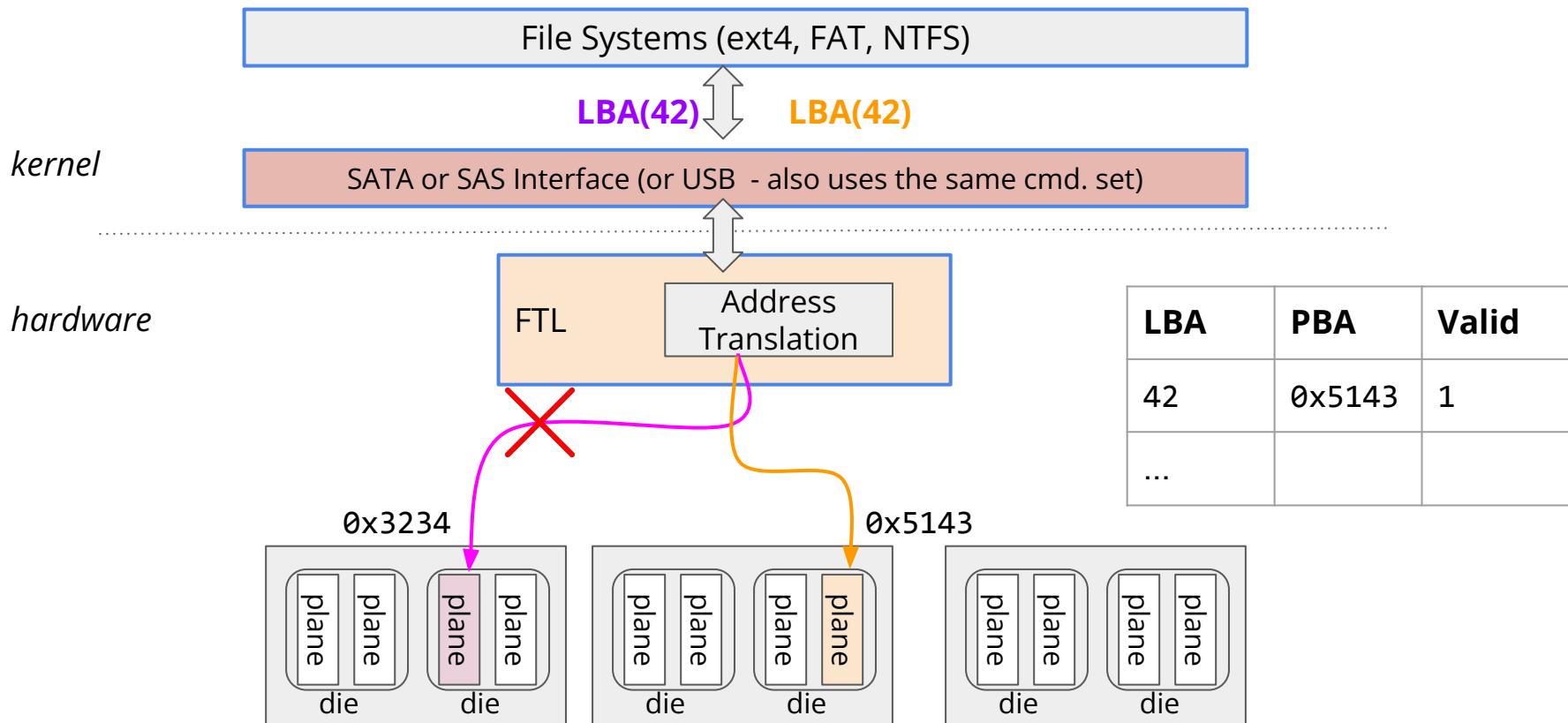
A piece of logic that runs on the controller and is responsible for : **Address Translation**



# What Happens when you overwrite a LBA



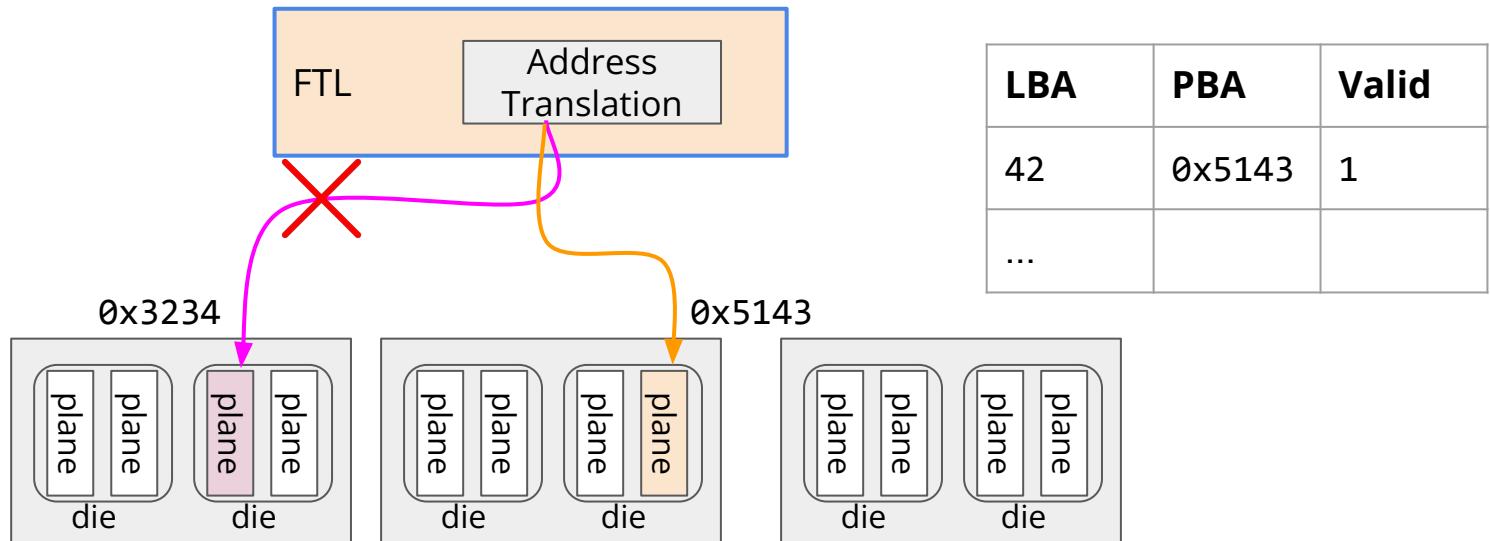
# What Happens when you overwrite a LBA



# The basic FTL address translation

Can give a way to rewrite the same address again and again

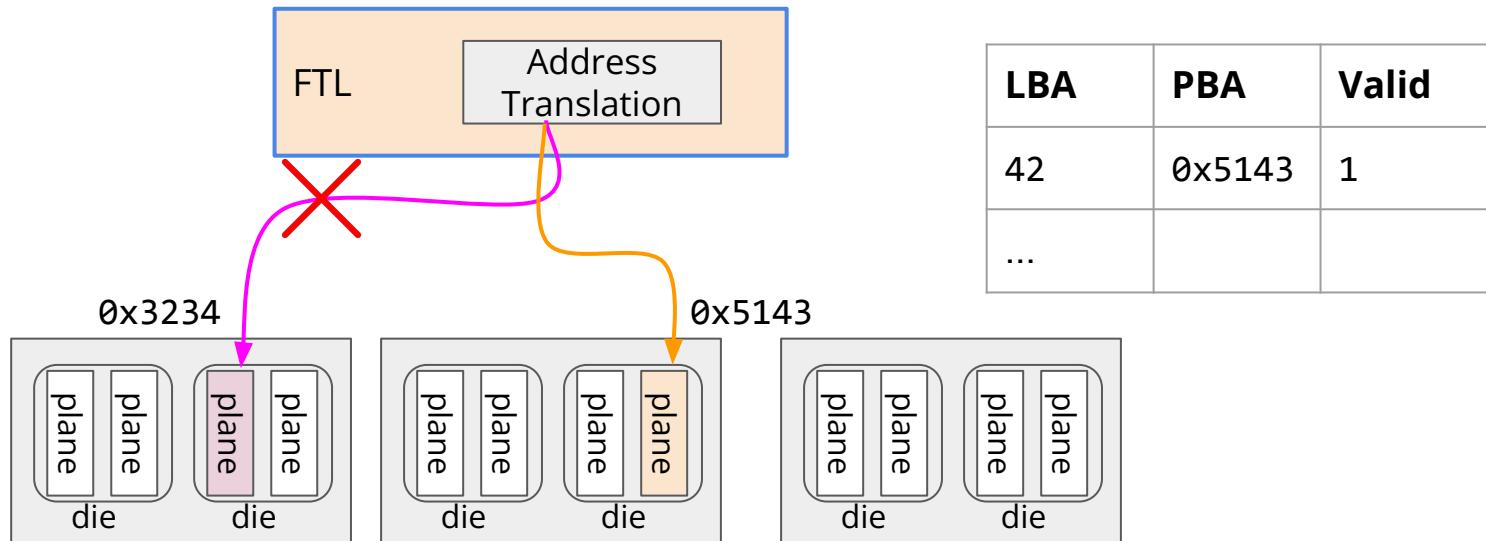
- *Is that all?*



# The basic FTL address translation

Can give a way to rewrite the same address again and again

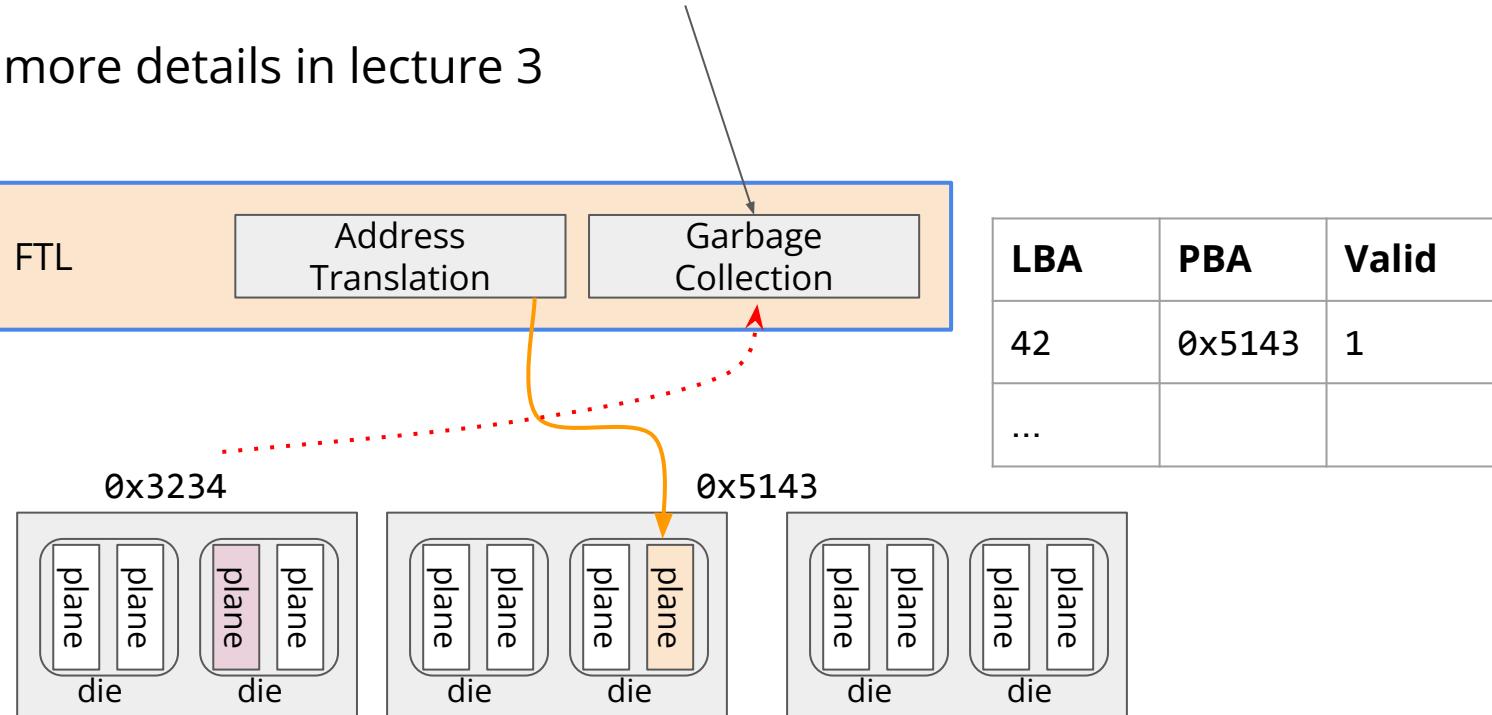
- *What happened with the 0x3234 page?*
- *What if there are no free ready blocks available?*



# Garbage collection (GC)

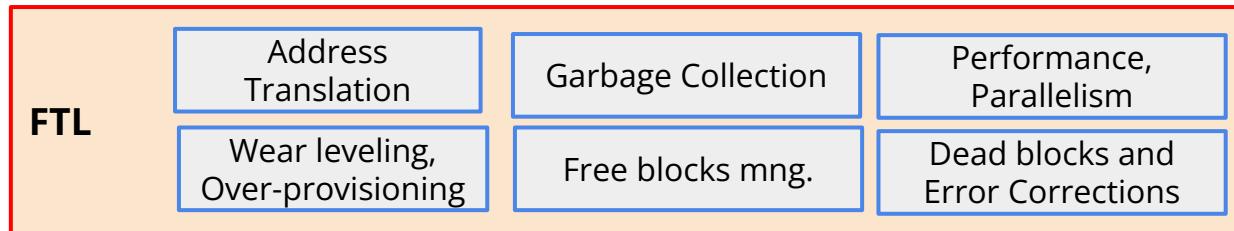
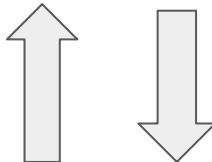
The old blocks need to keep track of and **garbage collected** for erasing

We will see more details in lecture 3

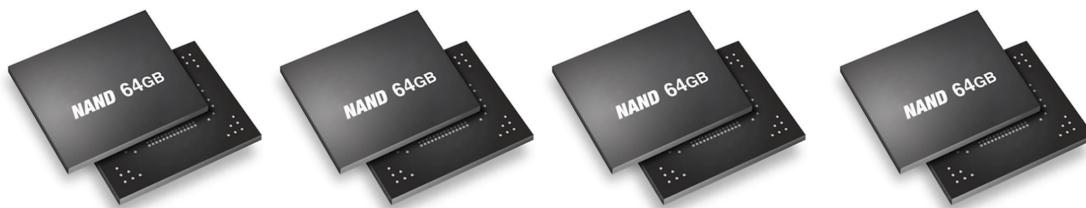


# Flash Translation Layer does a lot more ...

*Reads writes from the software (OS, application, fs)*



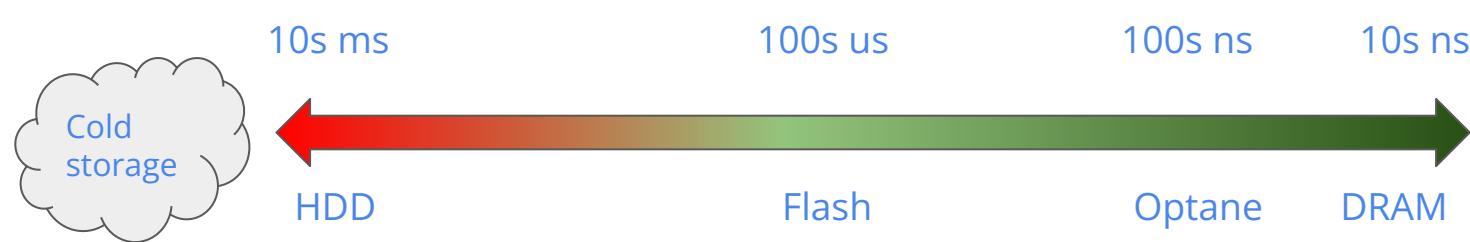
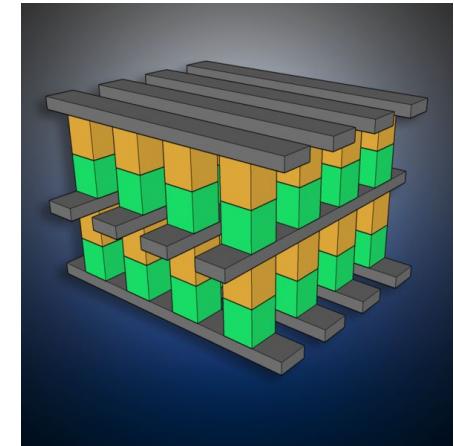
***FTL is the secret sauce***



# Questions?

# Optane: A new class of NVM technology

- Jointly developed by Intel and Micron
- Uses a new type of material
  - Not publicly disclosed - but thought to be a resistive bulk material
  - Hard to guess internals, but we are trying ;)
- Much faster than flash, and can be packaged as byte-addressable memory
  - Latencies in 100s of nanoseconds
  - Bandwidths in 10s of GBytes/sec



# Optane: Memory and storage form factors

~~Today available~~ in storage as well as memory form factor

- Optane DIMMs (memory/cache) : 128GB, 256GB and 512GB DIMMs
  - Do direct load/store from the CPU
  - Treat it like a persistent memory
- Optane SSDs (as storage /dev/optane)
  - Make a file system, and do file I/O, mmap, read, and write



# Though financial realities can be harsh ...

SIGN IN / UP

The Register

STORAGE

This article is more than 1 year old

# Why Intel killed its Optane memory business

32

Effort to create a new tier of memory flopped as rivals offered faster and more open alternatives

 Tobias Mann

Fri 29 Jul 2022 01:30 UTC

**ANALYSIS** Intel CEO Pat Gelsinger has confirmed that Intel will quit its Optane business, ending its attempt to create and promote a tier of memory that's a little slower than RAM but had the virtues of persistence and high IOPS.

The news should not, however, come as a surprise. The division has been on life support for some time following Micron's 2018 decision to terminate its joint venture with Intel, selling the fab in which the 3D XPoint chips that go into Optane drives and modules were made. While Intel has signaled it is open to using third-party foundries, without the means to make its own Optane silicon, the writing was on the wall.

As our sister site *Blocks and Files* reported in May, the sale only came after Micron had saddled Intel with a glut of 3D XPoint memory modules – more than the chipmaker could sell. Estimates put Intel's inventories at roughly two years worth of supply.

In its poor earnings report for Q2, Intel said quitting Optane will result in a \$559 million inventory impairment. In other words, the company is giving up on the project and writing off the inventory as a loss.

The deal also signals the end of Intel's SSD business. Two years ago Intel sold its NAND flash business and manufacturing plans to SK hynix to focus its efforts on the Optane business.

Announced in 2015, 3D XPoint memory arrived in the form of Intel's Optane SSDs two years later. However, unlike SSDs from rivals, Optane SSDs couldn't compete on capacity or speed. The devices instead offered some of the strongest I/O performance on the market – a quality that made them particularly attractive in latency sensitive applications where sheer IOPS were more important than throughput. Intel claimed its PCIe 4.0-based P5800X SSDs could reach up to 1.6 million IOPS

Intel also used 3D XPoint in its Optane persistent memory DIMMs, particularly around the launch of its second- and third-gen Xeon Scalable processors.

[https://www.theregister.com/2022/07/29/intel\\_optane\\_memory\\_dead/](https://www.theregister.com/2022/07/29/intel_optane_memory_dead/)

**SYSTEMS**

This article is more than 1 year old

# How CXL may change the datacenter as we know it

6 

Bye-bye bottlenecks. Hello composable infrastructure?

By  Tobias Mann Mon 16 May 2022 11:24 UTC



**INTERVIEW** Compute Express Link (CXL) has the potential to radically change the way systems and datacenters are built and operated. And after years of joint development spanning more than 190 companies, the open standard is nearly ready for prime time.

For those that aren't familiar, CXL defines a common, cache-coherent interface for connecting CPUs, memory, accelerators, and other peripherals. And its implications for the datacenter are wide ranging. Jim Pappas, CXL chairman and Intel director of technology initiatives, tells *The Register*.

So with the first CXL-compatible systems expected to launch later this year alongside Intel's Sapphire Rapids Xeon Scalables and AMD's Genoa forth-gen Epycs, we ask Pappas how he expects CXL will change the industry in the near term.

### Composable memory infrastructure

According to Pappas, one of the first implementations for CXL will likely involve system memory. Until now, there've only been two ways to attach more memory to an accelerator, he explains. Either you added more DDR memory channels to support more modules, or it had to be integrated directly onto the accelerator or CPU package.

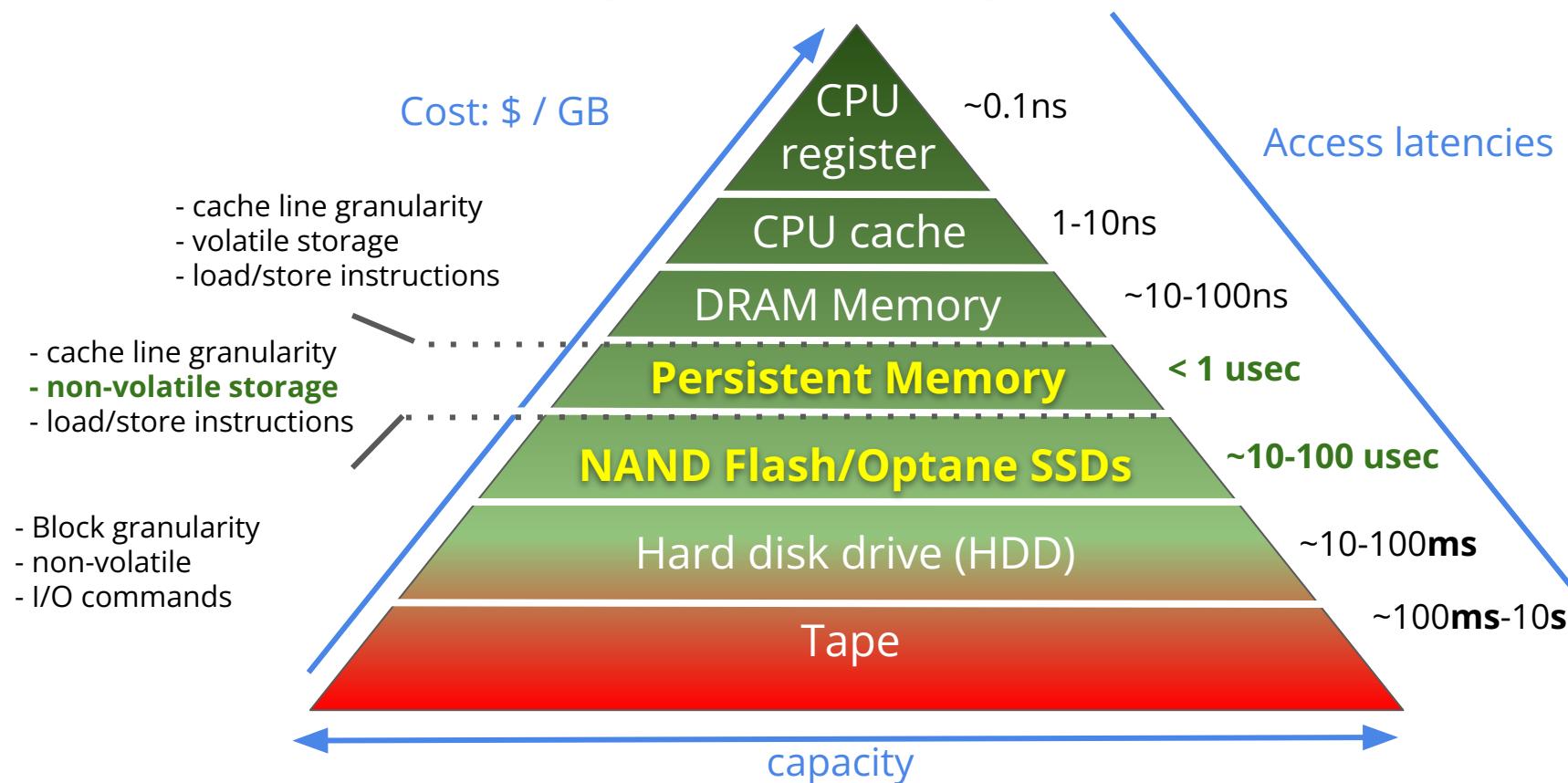
"You can't put memory on the PCIe bus," but with CXL you can, Pappas says. "CXL was designed for accelerators, but it was also designed to have a memory interface. We all knew from the very beginning that this could be used as a different port for memory."

Instead of populating a system with more or larger memory modules, additional memory could be installed via a card using a common interface for PCIe and CXL. And thanks to the simple-switching systems introduced with the CXL 2.0 spec, it became possible for resources, including memory, to be pooled and accessed by multiple systems simultaneously.

<https://www.computeexpresslink.org/resource-library>

[https://www.theregister.com/2022/05/16/cxl\\_datacenter\\_memory/](https://www.theregister.com/2022/05/16/cxl_datacenter_memory/)

# The (new) triangle of storage hierarchy



# Funny enough ...in the Architecture Bible

## ***Computer Architecture: A Quantitative Approach*** **John Hennessy and David Patterson**

5.4	Protection: Virtual Memory and Virtual Machines	315
5.5	Crosscutting Issues: The Design of Memory Hierarchies	324
5.6	Putting It All Together: AMD Opteron Memory Hierarchy	326
5.7	Fallacies and Pitfalls	335
5.8	Concluding Remarks	341
5.9	Historical Perspective and References	342
	Case Studies with Exercises by Norman P. Jouppi	342

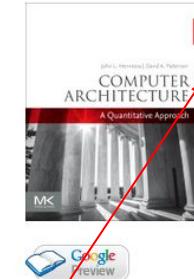
### Chapter 6 Storage Systems

6.1	Introduction	358
6.2	Advanced Topics in Disk Storage	358
6.3	Definition and Examples of Real Faults and Failures	366
6.4	I/O Performance, Reliability Measures, and Benchmarks	371
6.5	A Little Queuing Theory	379
6.6	Crosscutting Issues	390
6.7	Designing and Evaluating an I/O System—The Internet Archive Cluster	392
6.8	Putting It All Together: NetApp FAS6000 Filer	397
6.9	Fallacies and Pitfalls	399
6.10	Concluding Remarks	403
6.11	Historical Perspective and References	404
	Case Studies with Exercises by Andrea C. Arpaci-Dusseau and Remzi H. Arpaci-Dusseau	404

### Appendix A Pipelining: Basic and Intermediate Concepts

A.1	Introduction
-----	--------------

5th edition



Computer Architecture

6th Edition

A Quantitative Approach

Write a review

Authors: John Hennessy, David Patterson

eBook ISBN: 9780128119068

Paperback ISBN: 9780128119051

Imprint: Morgan Kaufmann

Published Date: 23rd November 2017

Page Count: 936

[View all volumes in this series: The Morgan Kaufmann Series ...](#)

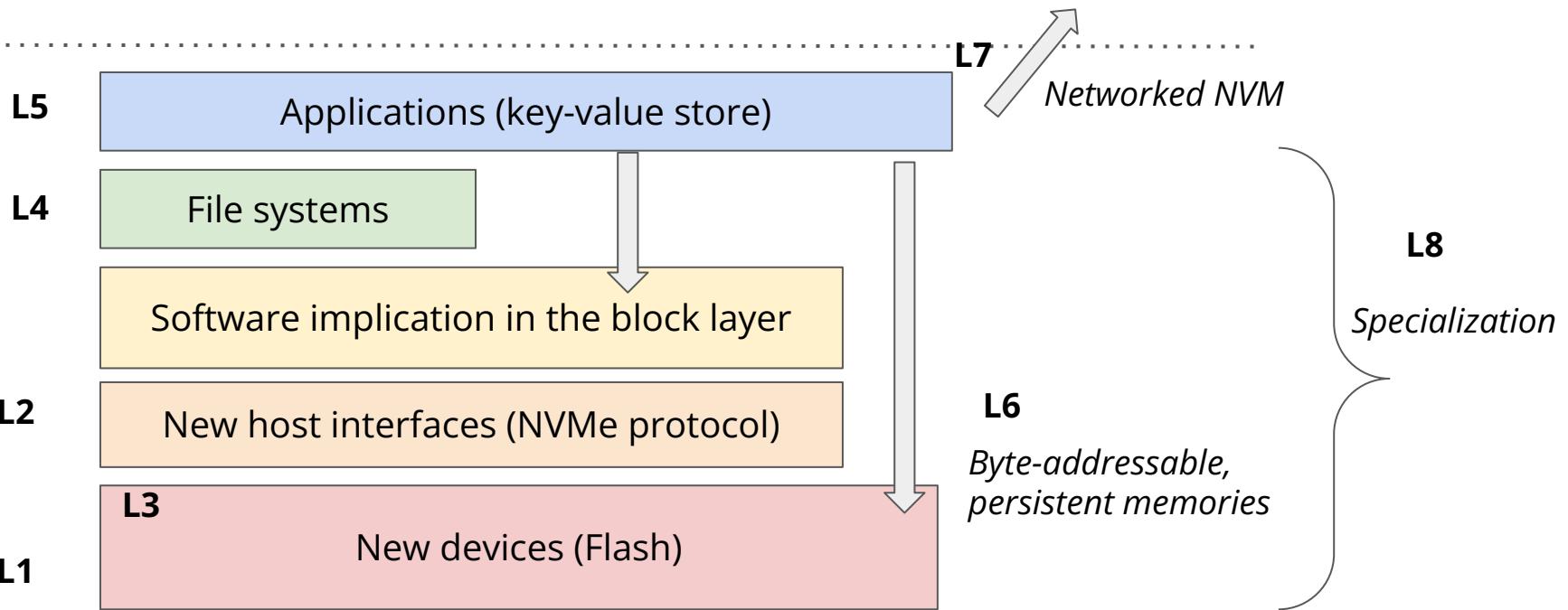
### Table of Contents

#### Printed Text

1. Fundamentals of Quantitative Design and Analysis
2. Memory Hierarchy Design
3. Instruction-Level Parallelism and Its Exploitation
4. Data-Level Parallelism in Vector, SIMD, and GPU Architectures
5. Multiprocessors and Thread-Level Parallelism
- 6. The Warehouse-Scale Computer**
7. Domain Specific Architectures
  - A. Instruction Set Principles
  - B. Review of Memory Hierarchy
  - C. Pipelining: Basic and Intermediate Concepts

# The layered approach in the lectures

Distributed Systems L9-L10

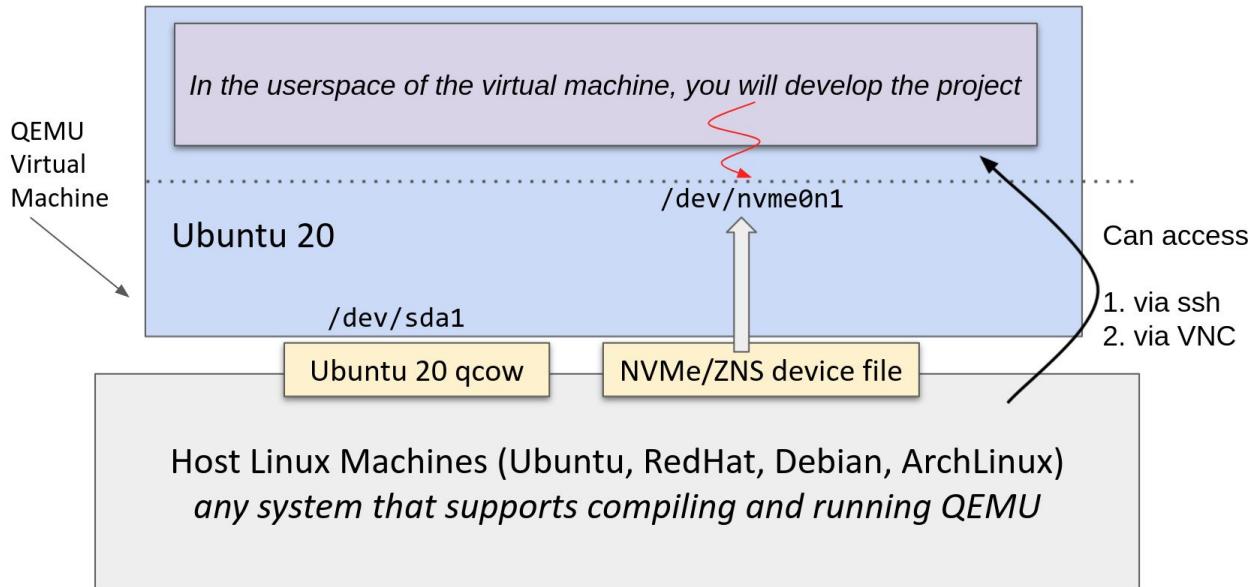


# Syllabus outline

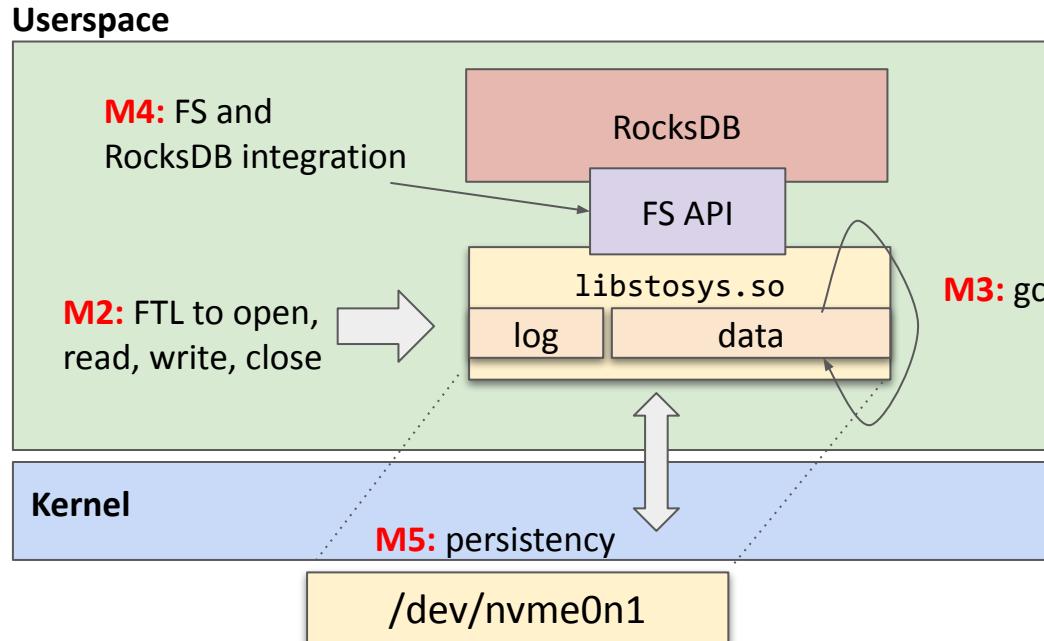
1. **Welcome and introduction to NVM (today)** 
2. Host interfacing and software implications
3. Flash Translation Layer (FTL) and Garbage Collection (GC)
4. NVM Block Storage File systems
5. NVM Block Storage Key-Value Stores
6. Emerging Byte-addressable Storage
7. Networked NVM Storage
8. Trends: Specialization and Programmability
9. Distributed Storage / Systems - I
10. Distributed Storage / Systems - II
11. Emerging Topics

# Project overview

You will be doing coding to understand internals and challenges associated with modern storage stacks (for now just “functional” not “performance”)



# Project milestones



# On Canvas - read the project handbook

Stosys System Project Handbook (2023 - 2024)

**VU**

**Storage Systems (StoS) Course Project Handbook**  
(XM\_0092)  
P1 (September - October), 2023  
Version: 4.0

**Animesh Trivedi** ([a.trivedi@vu.nl](mailto:a.trivedi@vu.nl))

Matthijs Jansen ([m.s.jansen@vu.nl](mailto:m.s.jansen@vu.nl)), Zebin Ren ([z.ren@vu.nl](mailto:z.ren@vu.nl)),  
Krijn Doeckemeijer ([k.doeckemeijer@vu.nl](mailto:k.doeckemeijer@vu.nl)), Nick Tehrany ([n.a.tehrany@vu.nl](mailto:n.a.tehrany@vu.nl)) and  
Sudarsan Sivakumar ([s.sivakumar@student.vu.nl](mailto:s.sivakumar@student.vu.nl))

**Storage-Networking Research Group**  
<https://stonet-research.github.io/>



Course mascot (courtesy: Gilles Magalhaes via DALL-E)



# Warning

This is an **experimental** course

- You are a part of the experiment
- We will build upon your feedback

This is a **coding-heavy** course

- If you have not done C/C++ programming this may be a tough course
  - *Use of structs, pointers, file I/O, thread synchronization, locks*
- Start coding early, there will be plenty of surprises (but fun as well)

*There's a relatively little flexibility with the deadlines (as they all dependent on each other)*



# Plagiarism in StoSys



- Copy (part of) a solution from another team
- Copy (part of) a solution from the Internet
- **Buy a solution from any other source/outsource assignment**
- Copy + make minor changes to any of the above

## Why is plagiarism really bad?

- **Bad** for you: hurts your learning process
- **Bad** for others: takes away from the work of others and disrespectful !
- **Bad** for us: a lot of extra unthankful work
- **Bad** for everybody: hurts reputation of the course and VU



<https://vu.nl/en/student/examinations/academic-integrity>  
<https://vu.nl/en/student/examinations/generative-ai-your-use-our-expectations>

# Plagiarism in StoSys



## How is plagiarism detected in StoSys?

- We have automatic plagiarism detection tools (code pattern matching)
- Internet
- Manual checks

"I did not know" or "I was not told not to do this" - are not excuses that will be accepted

## What happens if you commit plagiarism?

- You will be reported to the exam committee
- It is up to them to decide on disciplinary actions
- **Good news:** you get exactly 1 warning first
- **Bad news:** that warning is today!

# What is acceptable, what is not

We believe in open-collaboration, and learning - the whole field of Computer Science is built on it ...

- Acceptable
  - Discussing a potential design with others
  - Discussing a debugging problem with others
  - Looking on internet / Linux kernel source code for clarification
- Not acceptable
  - Under no circumstances copying code is acceptable
  - No transfer/sharing of code from one team to another

This is not an exhaustive list. We will deal such issues on a case by case basis

# Recap: From this lecture

- **Understand: The difference between storage and memory is blurring**
- You should know what is NVM
- You should know flash internals, and how they differ from HDDs
- What are SLC, MLC and TLC flash cells, what do they offer
- What is an FTL (the idea of an SSD-internal mapping)
- What are LBAs and PBAs
- What is Optane memory
- General performance ballpark numbers of NVM technologies

# Lecture Reading List

1. Michael Cornwell. 2012. Anatomy of a Solid-state Drive: While the ubiquitous SSD shares many features with the hard-disk drive, under the surface they are completely different. Queue 10, 10 (October 2012), 30–36. DOI:<https://doi.org/10.1145/2381996.2385276>
2. Design Tradeoffs for SSD Performance,  
[https://www.usenix.org/legacy/events/usenix08/tech/full\\_papers/agrawal/agrawal.pdf](https://www.usenix.org/legacy/events/usenix08/tech/full_papers/agrawal/agrawal.pdf)
3. Operating System Implications of Fast, Cheap, Non-Volatile Memory,  
[https://www.usenix.org/legacy/events/hotos11/tech/final\\_files/Bailey.pdf](https://www.usenix.org/legacy/events/hotos11/tech/final_files/Bailey.pdf)
4. Flash-based SSDs, Operating Systems: Three Easy Pieces,  
<http://pages.cs.wisc.edu/~remzi/OSTEP/file-ssd.pdf>
5. Nand flash solid state storage: Performance and capability, an in-depth look  
[https://www.snia.org/sites/default/education/tutorials/2009/spring/solid/JonathanThatcher\\_NandFlash\\_SSS\\_PerformanceV10-nc.pdf](https://www.snia.org/sites/default/education/tutorials/2009/spring/solid/JonathanThatcher_NandFlash_SSS_PerformanceV10-nc.pdf)
6. Data Storage Research Vision 2025, <https://dl.acm.org/doi/book/10.5555/3316807>  
<https://par.nsf.gov/servlets/purl/10086429>