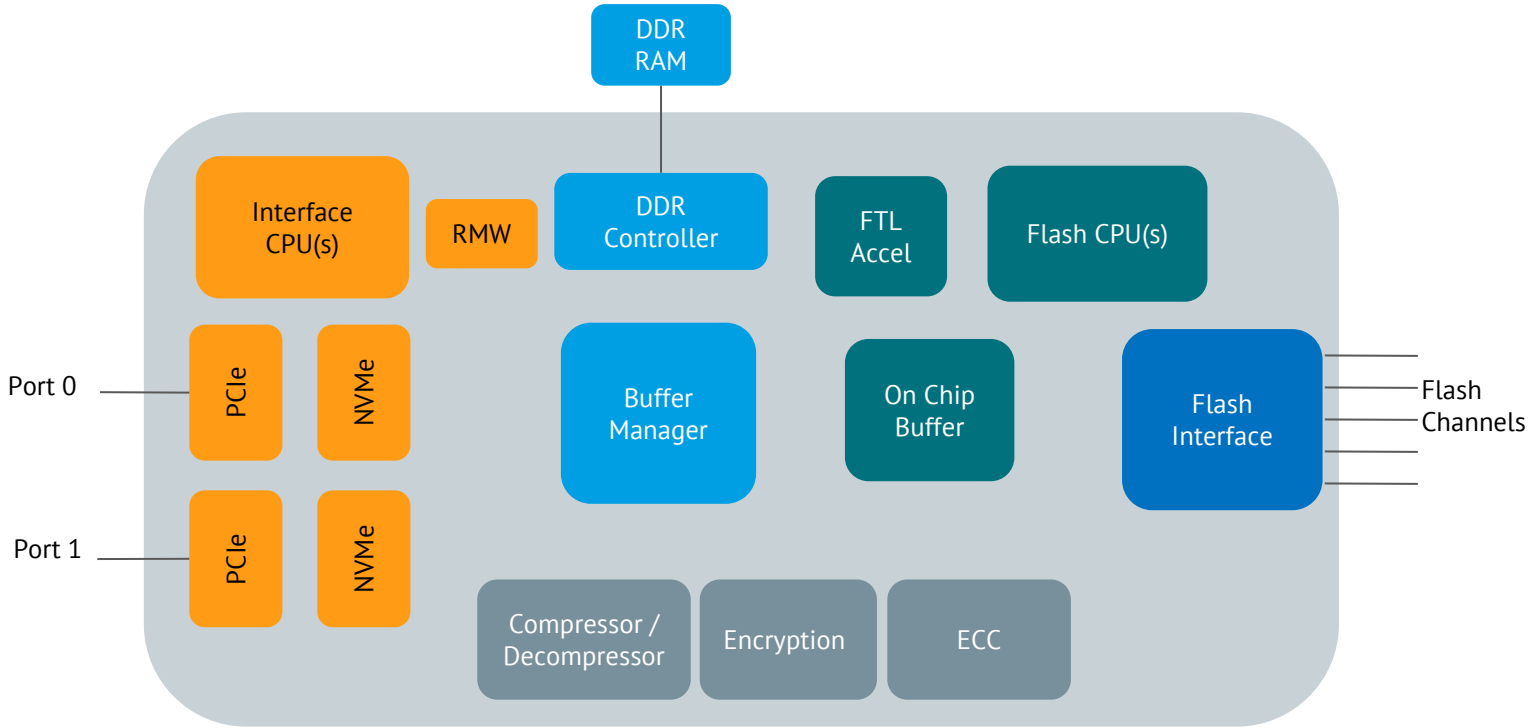
Flash Controller Technology

Andy Tomlin

NVMe SSD's

- NVMe is displacing SATA in applications which require performance
- NVMe has excellent programing model for host software
- Latency is becoming the key driving force for system performance, although IOPS always gets the marketing dollars
- Low latency is driving increasingly higher levels of HW automation for command processing
- One of key differentiations in future products will be how and what is HW automated by the controller
- Dividing line between FW and HW and interaction between will be key problem

Basic block diagram of NVMe Controller



Controller Types

Market	Client	Data Center	Enterprise
Dies	8-32	16-64	16-256
Capacity	128GB-1TB	512GB-2TB	512GB-8TB
NAND Channels	4-8	8-16	8-16
NVMe Ports/(Lanes)	1/(4)	1/(4-8)	1-2(4-16)
Memory	MLC/TLC	MLC/TLC	SLC/MLC/(TLC)
Mapping	4K	4K/8K	4K/8K/16K

Compression

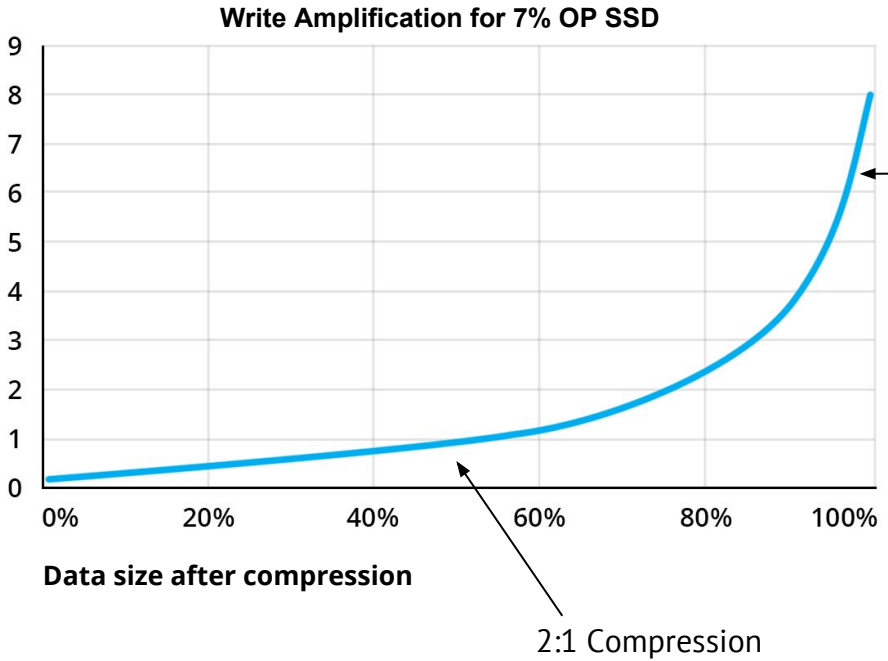
- No mainstream adoption in SSD's since Sandforce series of controllers
- Heavy adoption in Storage systems
- Compression has non-linear effect on write performance, as compression changes not just the size of what you are writing but the effective overprovisioning of the drive/system that you are writing into

Downsides of compression:

- Complexity of the system. Dealing with variable length data.
- Latency implications - need fast compression/decompression

Write amplification and compression

Typical laptop data is ~2:1 compressible



Modeling data, does not include WA from metadata
https://arxiv.org/pdf/1110.4245.pdf

PCIe

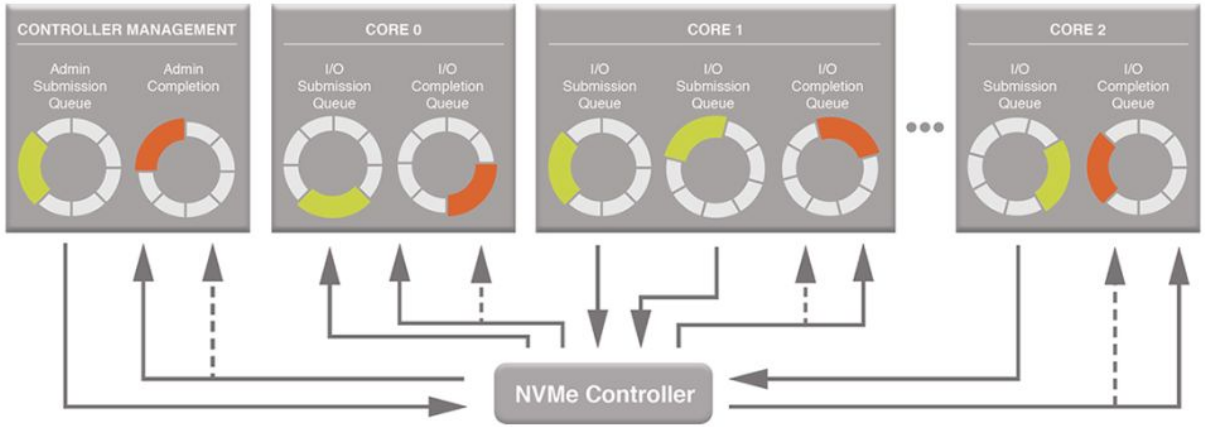
- Gen3 (8GT/s), Gen4 coming (16GT/s)
- Typical SSD Controller configured as PCIe endpoint
- Client and Data Center drives usually single port, some high end Enterprise controllers dual port for storage system applications (often a key marketing differentiation)
- Typical implementations utilize 3rd party controller IP
- SRIOV for VM support maybe requirement for Enterprise & Data Center applications

NVMe queuing model

Highly efficient programming model compared to other storage protocols such as SATA

Designed for modern multi core CPU's

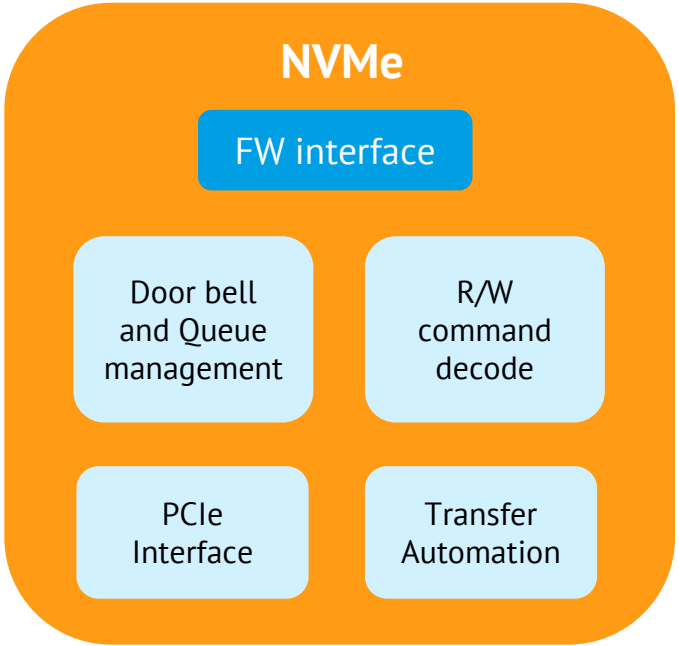
Queue in each direction, with doorbell notification mechanism



http://www.nvmexpress.org/nvm-express-overview/

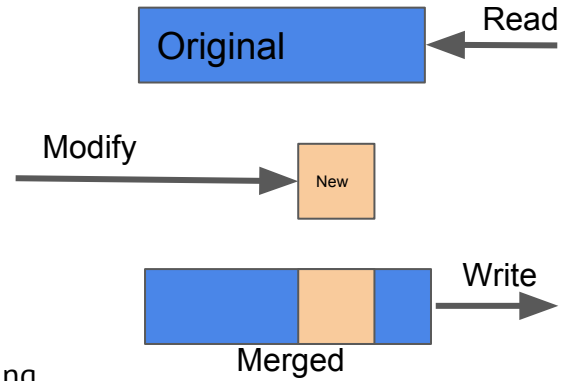
NVMe controller

- 64K queues possible by spec, most controllers 8-64 queue pairs
- Buffers can be defined by PRP - 4K chunks or SGL - Scatter/Gather list of segments
- SGL more flexible - especially for storage applications, but typical OS driver usage - 4K PRP is fine. SGL is optional and typically only implemented on leading edge or Enterprise controllers.
- With increasing performance the division of work between HW and FW increasingly complex and blurry. BIC solutions will fully automate basic R/W commands including trim without FW involvement.
- Atomic commands may be HW or FW
- Admin commands, other commands and exception conditions typically in FW



Unaligned writes (Read/Modify/Write)

- Interface block size is still most typically 512 bytes. Flash management mapping granularity typically 4K or higher. For writes smaller than the mapping granularity (unaligned) this leads to a Read/Modify/Write sequence
- Depending on market segment this is either done via FW or through HW automation
- This operation must take place on uncompressed, unencrypted data



Complications

- End to end protection scheme (if used) needs special handling
- Coherency is a natural problem for RMW as the the read creates different timing scenarios vs normal writes
- Coherency of overlapped writes (solvable with mapping)
- Coherency with other, non overlapping RMW commands

FTL Accelerator

At minimum this will perform L2P map lookup

Can possibly perform many other actions depending on implementation

- Map update
- Journalling

Map is typically used to solve various coherency issues in system

- GC vs host write
- RMW
- Delay between data allocation and meta data write

Map update will have two types of update methods

- Normal update - host write
- Conditional update - coherency checking

FTL Accelerator is center point for critical controller IP and so detail here are intentionally vague

Front end CPU(s) manage NVMe interface

As performance demands increase, Front End CPU load is increasing to the point that likely more than 1 core or higher end core needed to provide necessary performance

On client controllers this can cost from a power perspective

HW automation key here from performance and power perspective

With advanced levels of HW automation the interface between HW & FW can become critical bottleneck

- Register type interfaces carry heavy cycle count cost penalties
- Memory Queue based interfaces are preferred mechanism with registers mainly used for configuration or rare operations.
- Debug mechanisms must be built into HW

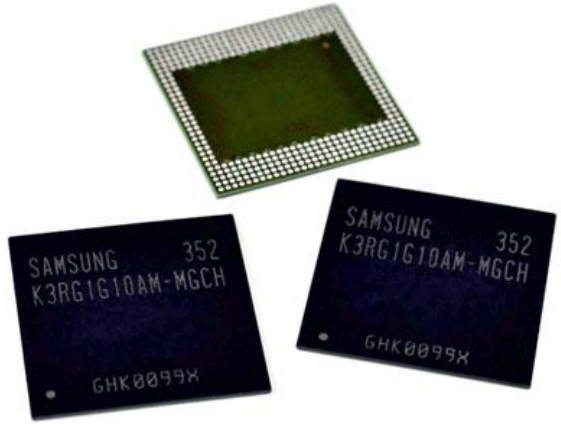
DDR interface

DDR is essential in all but very lowest end controllers. DDR4 typical today

DDR Memory will store:

- Map and other large data structures
- Read data
- Write data
- FW code

In many cases all read and write data will pass through DDR, making DDR bandwidth an important architectural factor



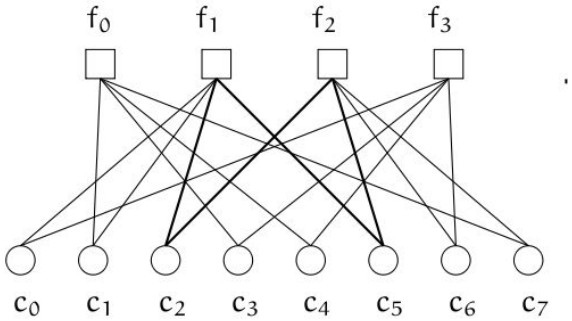
ECC engine

Flash Error rates continue to degrade with geometry

BCH typical on previous NAND geometries, but efficiency drops at higher correction capabilities. LDPC is taking over. Critical challenge is designing for higher throughputs. Likely requires multiple engines to meet throughput.

LDPC can perform soft and hard correction. Soft correction allows multiple reads to be combined for superior correction capabilities. This is only used when hard correction fails.

Depending on implementation the soft decoder may be separate engine



Buffer management

As more controller element move to HW, buffer management also may move to HW

Algorithms used to manage buffers in HW are typically the same as used in FW

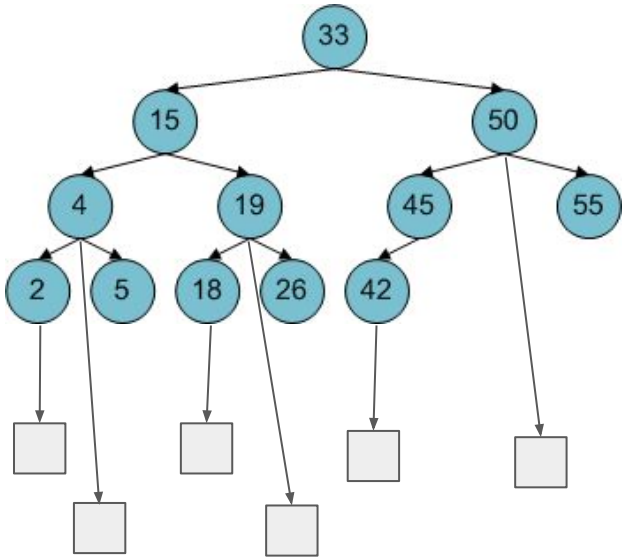
Two main problems are solved by buffer manager

- Allocation/Deallocation
- Address lookup

Address lookup typically done through binary search tree such as AVL or Red/Black tree

Allocation/Deallocation through simple list structure

Buffer manager also has to keep track of ECC state (RAW or corrected)



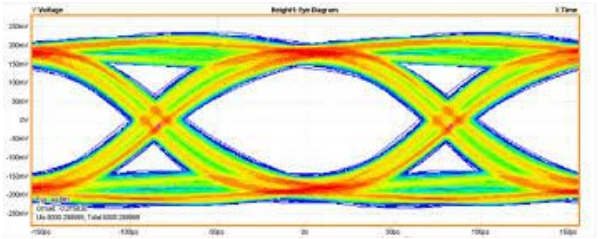
Flash interface

Each channel allows data transfer to set of dies. ONFI 3.x -> ONFI 4, or Toggle 2.0 allow up to 400MT/s transfer rates.

Speed is dependant on channel loading.

The NAND interface is typically managed by either very simple programmable HW sequencer or very small CPU that does nothing other than send command sequences.

Scrambler/Encryption engine is used to reduce probability of undesirable bit line patterns. This is either part of flash interface or ECC engine.



ONFi
OPEN NAND
FLASH INTERFACE

Flash CPU(s)

This is main FW work horse and runs the flash management system. Highly likely to be more than one core.

Main Tasks:

- Meta data management
- Error handling
- Block picking, Wear leveling
- Bad block management
- Scheduling
- Internal filesystem

Devicepros

Straight Forward

- **Straight Forward.** Our company motto embodies simplicity, efficiency, and transparency. We adopt “best practices” ruthlessly enabling speed, security and control as needed;
- Led by storage industry veterans, Derrill Sturgeon and Andy Tomlin. Based in Bay Area with development center in Minsk, Belarus providing expertise in Test Automation and Flash based systems to the Storage industry;
- Passionate about helping our customers achieve their goals! We aim to delight our customers with our services, not merely satisfy;
- Excellence in communication is one of our top priorities. Engineering Services requires robust and frank conversations from inception through execution.