

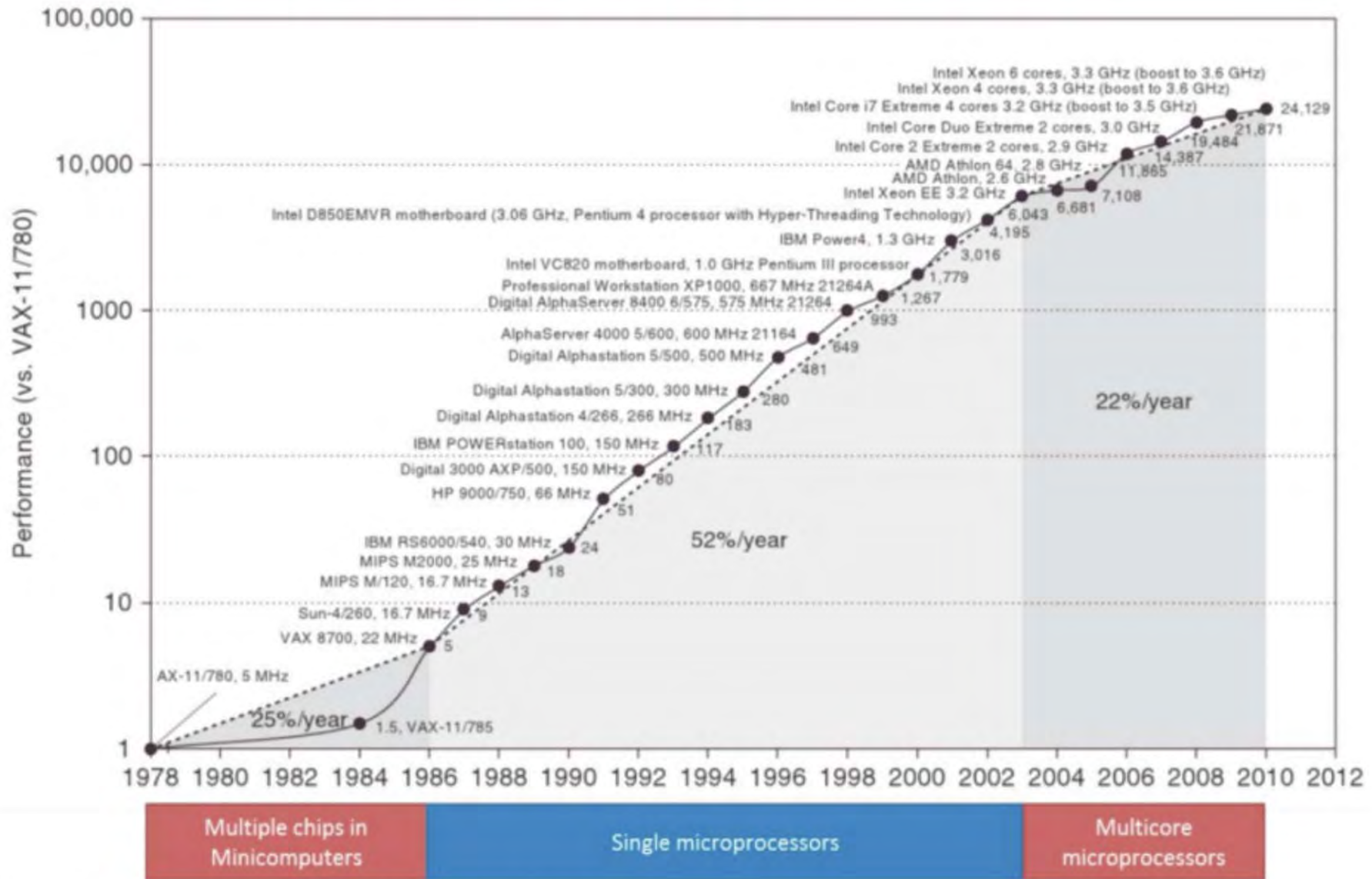
Scalable High Performance Flash Systems

Jeff Bonwick
Co-founder and CTO, DSSD / EMC

All computation is the same

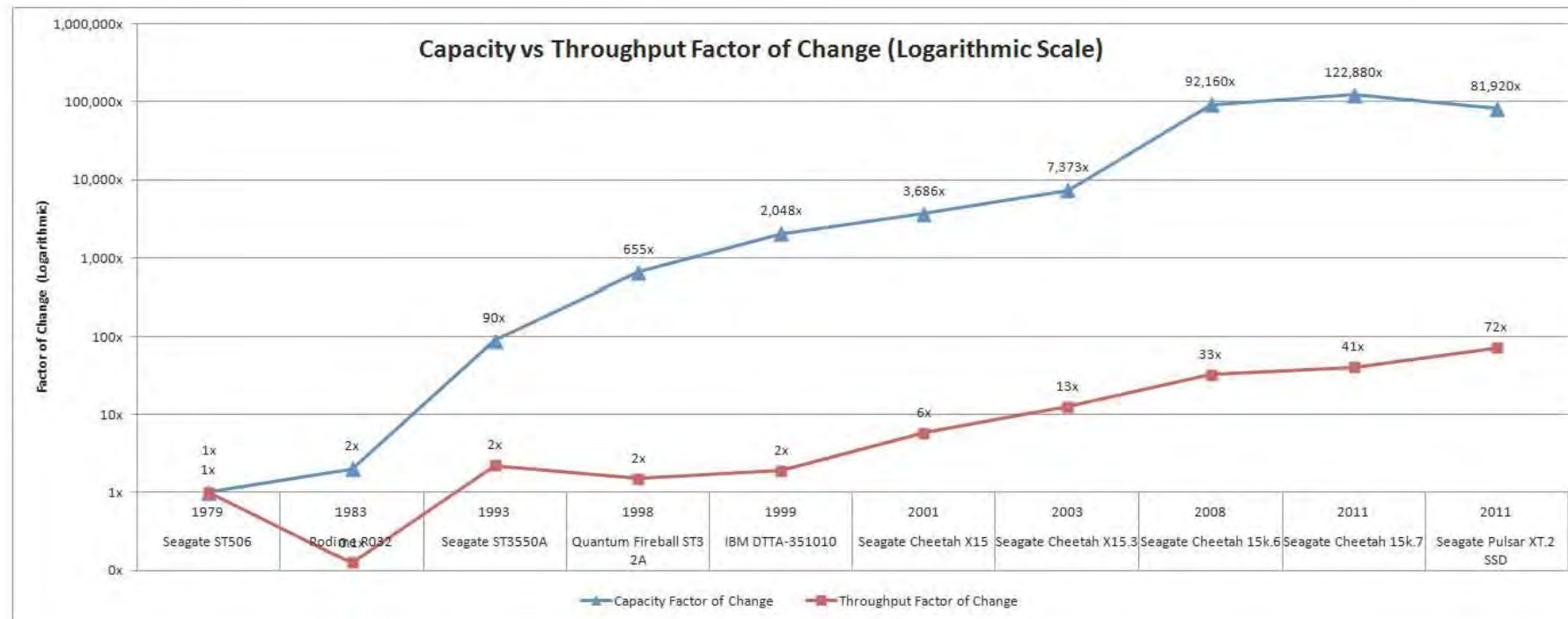
- `result = math(data);`
- To get results faster, both need to improve
 - faster math
 - faster data
- How's that going?

CPU performance is 20,000x faster than 1980...



... but disk performance hasn't kept up

- Capacity has grown ~100,000x
- Bandwidth has grown ~100x
- IOPS has grown ~10x



Source: <https://tylermuth.files.wordpress.com/2011/11/capacity-vs-throughput-lo.png>

We need to get I/O back on track

	1980 (HDD)	Today (DSSD)	Improvement
CPU	1	20,000	20,000x
Capacity	5 MB	100 TB	20,000,000x
Bandwidth	5 MB/s	100 GB/s	20,000x
IOPS	50	10M	200,000x

But how?

What can a single flash chip do?

- Media access time <100 us
- Implies >10,000 IOPS
- If doing 32K reads, implies >320 MB/s
- So a single SSD containing 512 NAND die (say 64 placements of 8 die stack packages) *could* deliver 5M IOPS and 160 GB/s
- Why doesn't it?

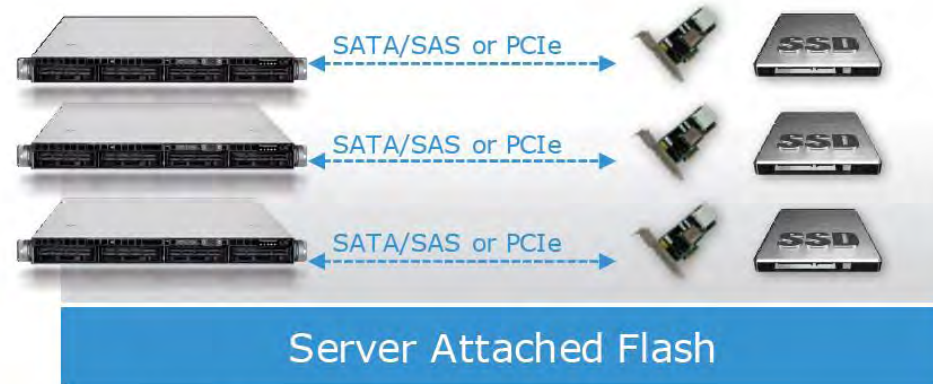
What limits flash performance?

- In general:
 - power and cooling
 - multiple NAND die per flash channel
 - each ONFI / Toggle flash channel 200-800 MB/s
- In an SSD form factor:
 - 12 Gb/s SAS / SATA interface
 - software overhead
- In a server-attached PCIe card form factor:
 - limited scale
 - inability to share

Two main approaches to flash thus far



- Enterprise storage features
- BUT**
- Traditional network latencies & I/O stack bottlenecks



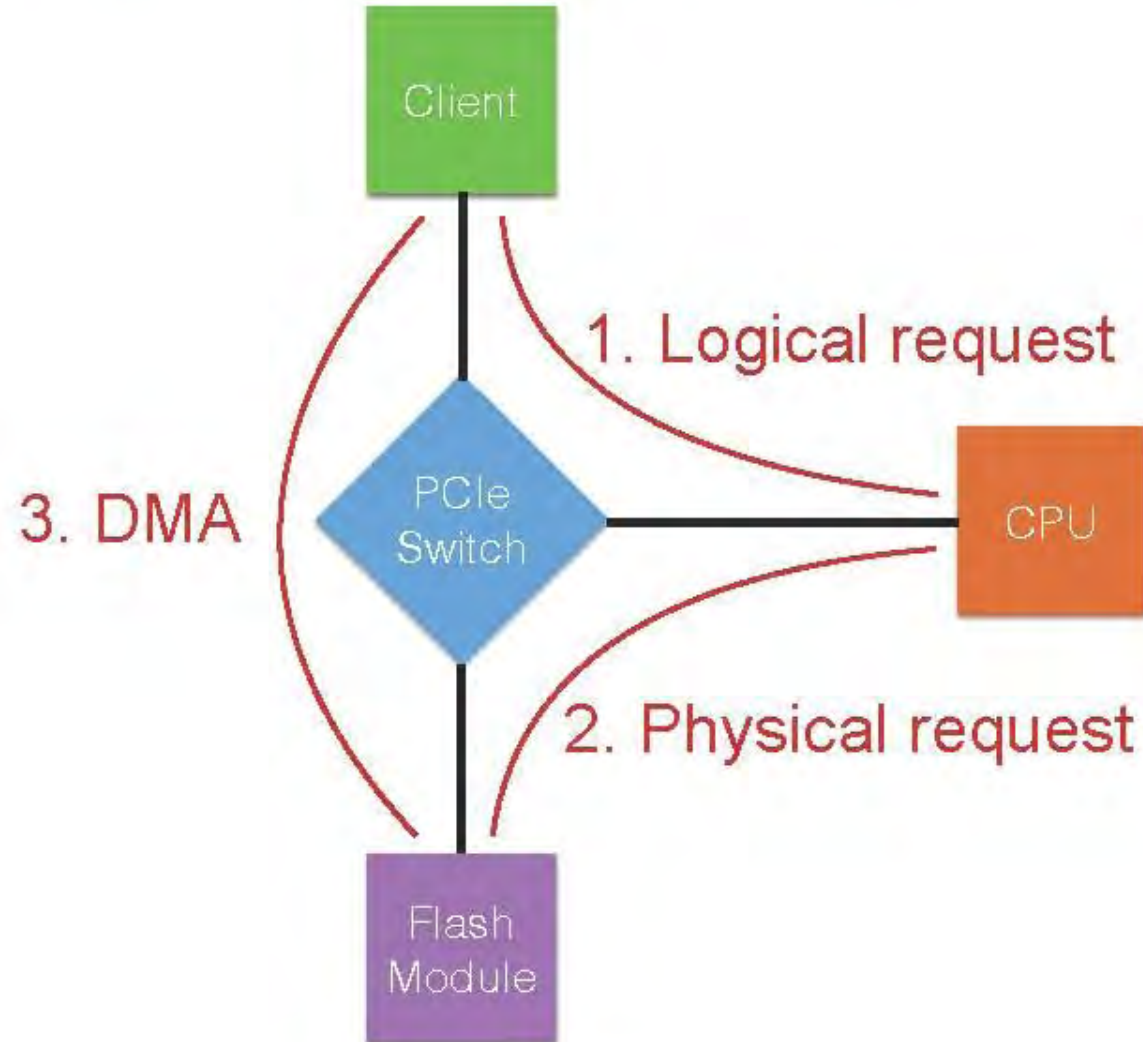
- Flash hostage to individual servers
- Stranded storage and data shuffling among servers
- No enterprise storage features
- Limited capacity

Is there some way to get the best of both?
Without the limitations of either?

Rack-Scale Flash

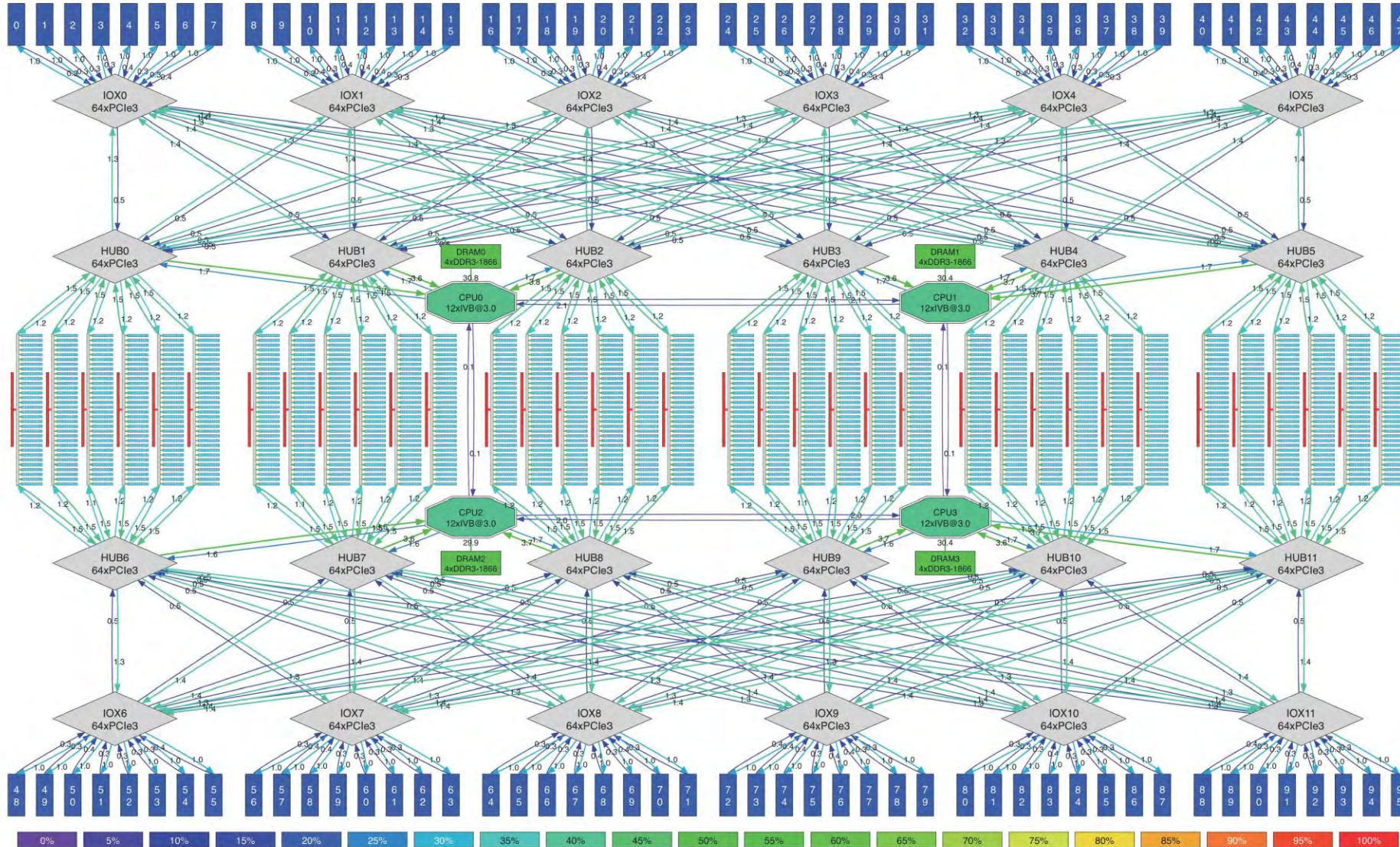
- Move the flash out of servers (like AFA)
- Support RAID, dual-porting, etc (like AFA)
- Allow all clients to see all data (like AFA)
- Abandon legacy SCSI stack (like PCIe card)
- Provide much stronger data protection (new)
- Exploit massive NAND concurrency (new)
- Move smarts out of individual FMs (new)

The Basic Design Element

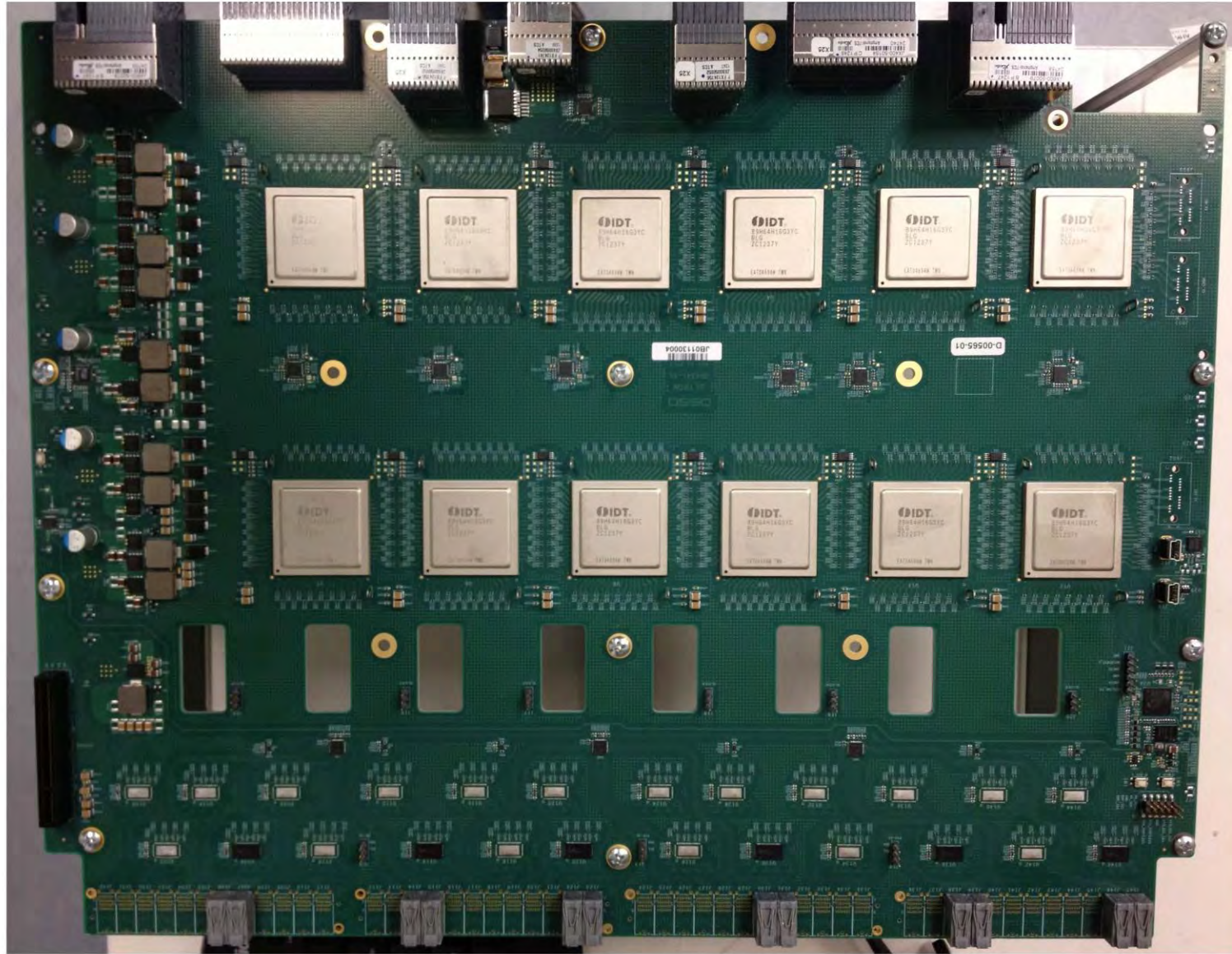


... highly replicated to make D5

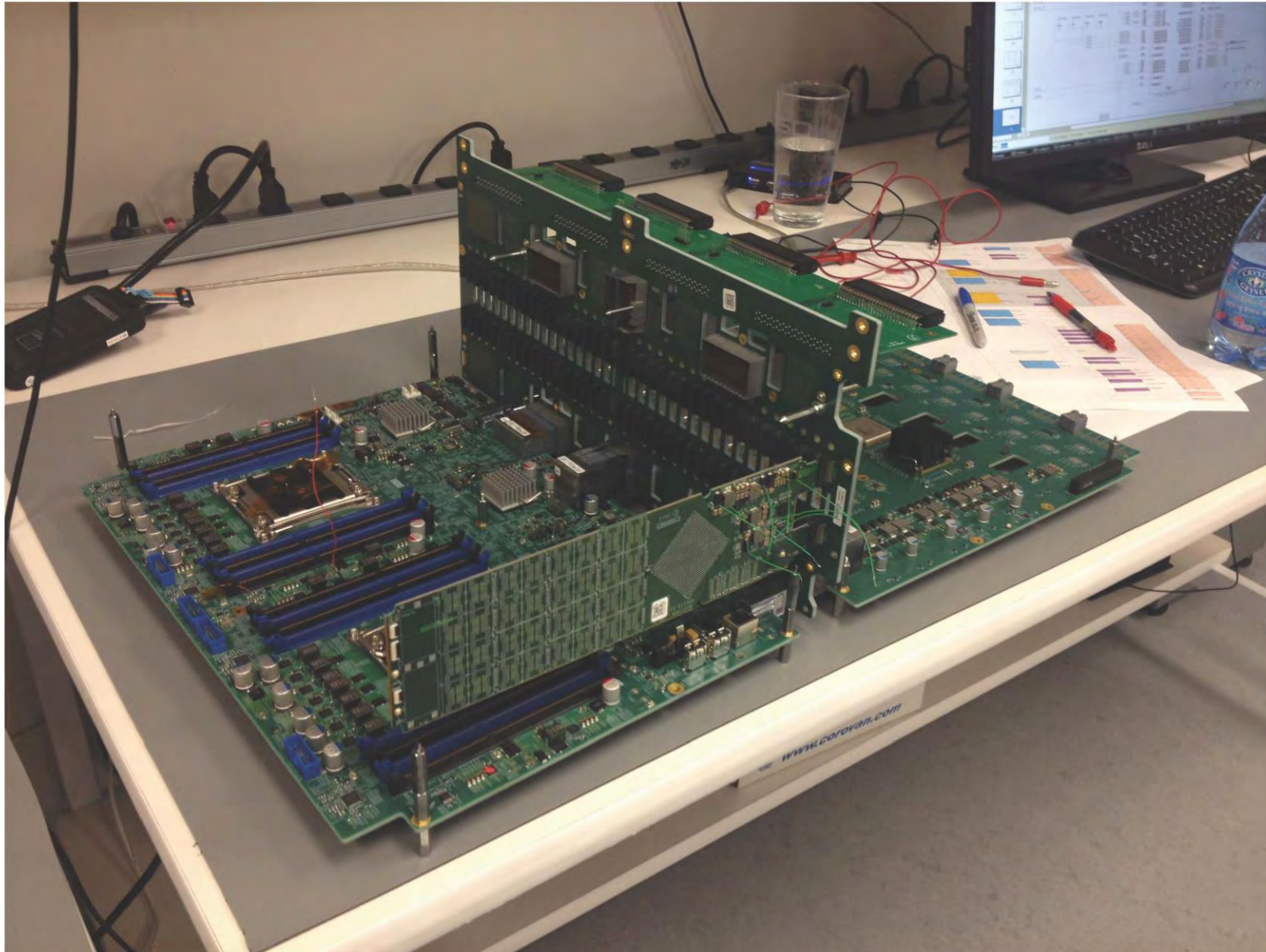
D5 1.0: 36 x 4T FMs, 96 ports RW mix 75/25 @ 8k, Amp 1.33, Cache 0% 129 GB/s (97R/32W) 16M IOPS (12R/4W) Life: 95TB @ 30k P/E = 3.2EB / 2.3y



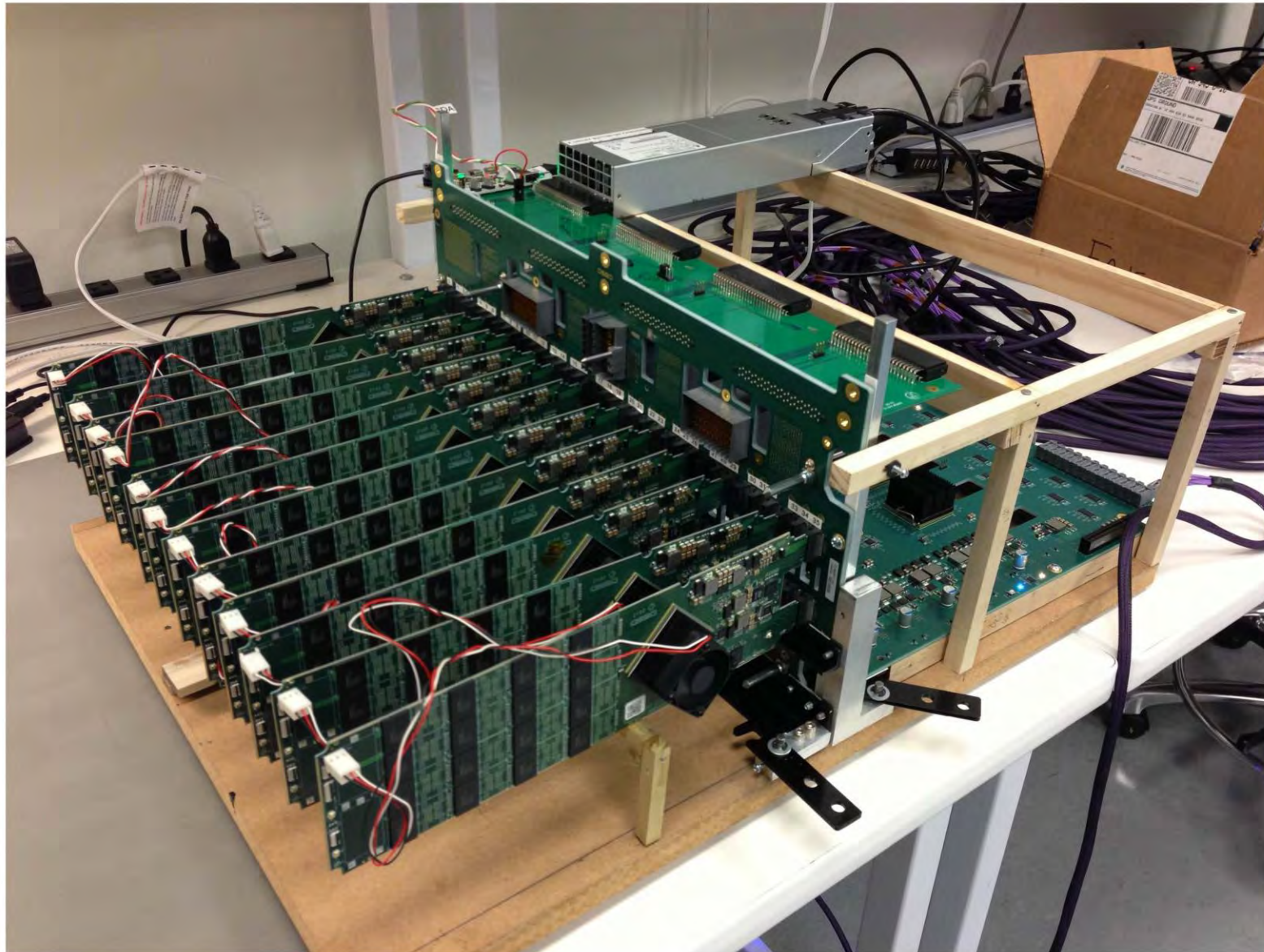
D5's full-mesh PCIe fabric



Anatomy of the D5



With a few more flash modules...



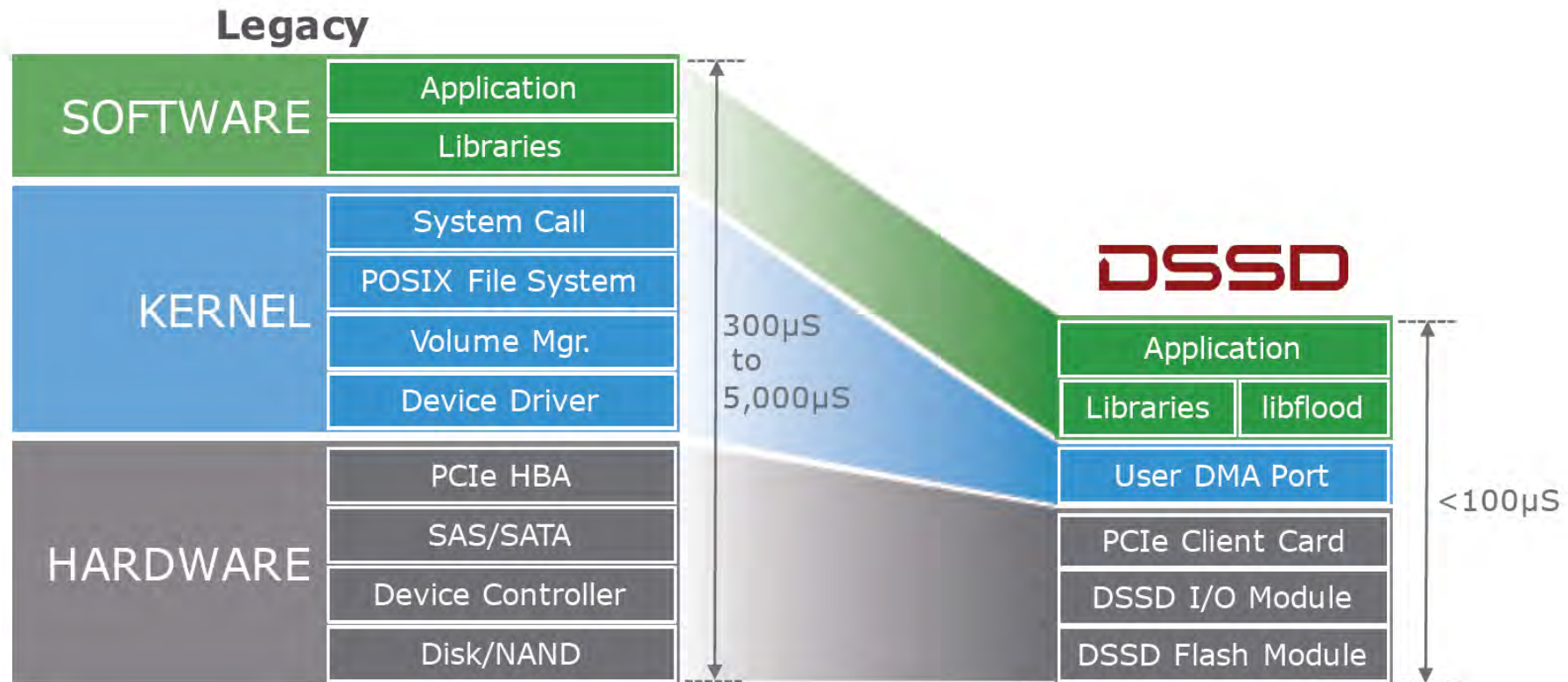


Software has to be radically different

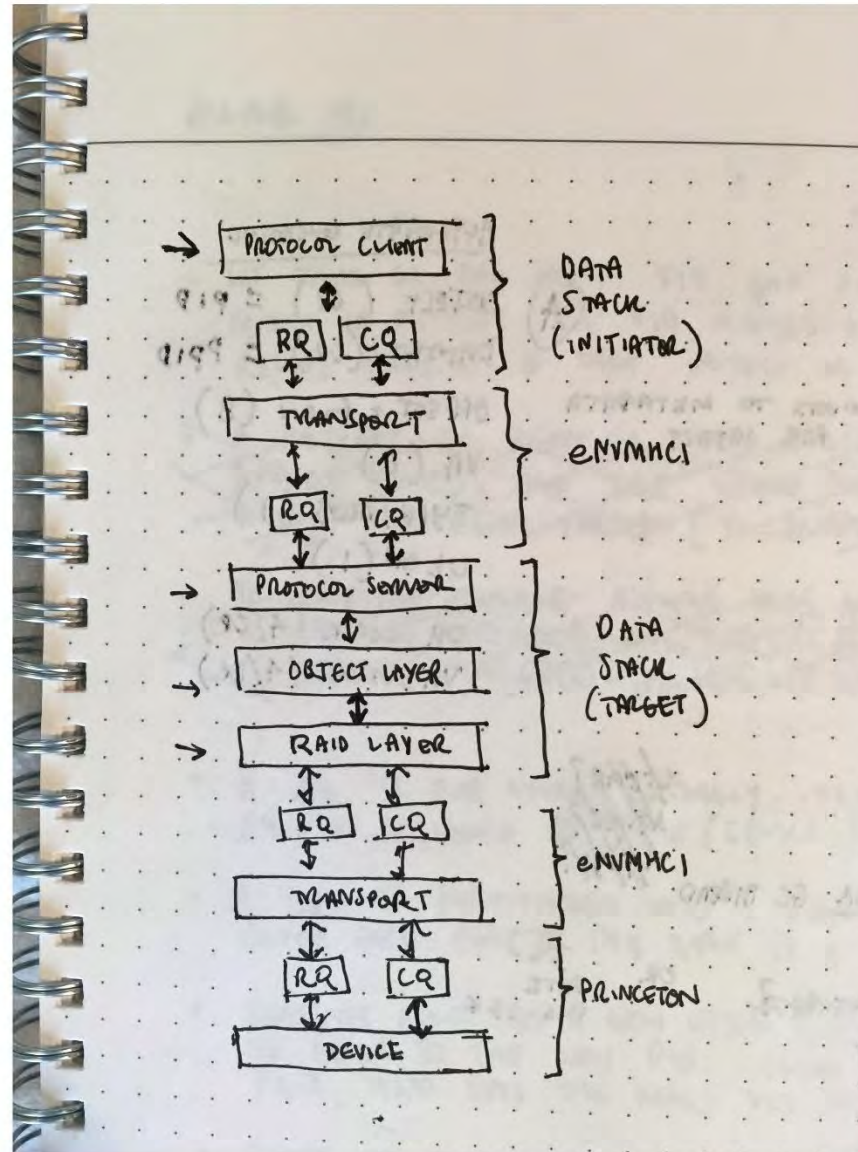
- D5 has 32 CPU cores running I/O
- Those 32 cores drive 10M IOPS
- That's 300k IOPS per core, or
- 3 microseconds per IOP
 - a few thousand instructions,
 - a dozen cache misses,
 - ... time's up!
- How can we do an IOP in 3 us?

Eliminate every layer possible

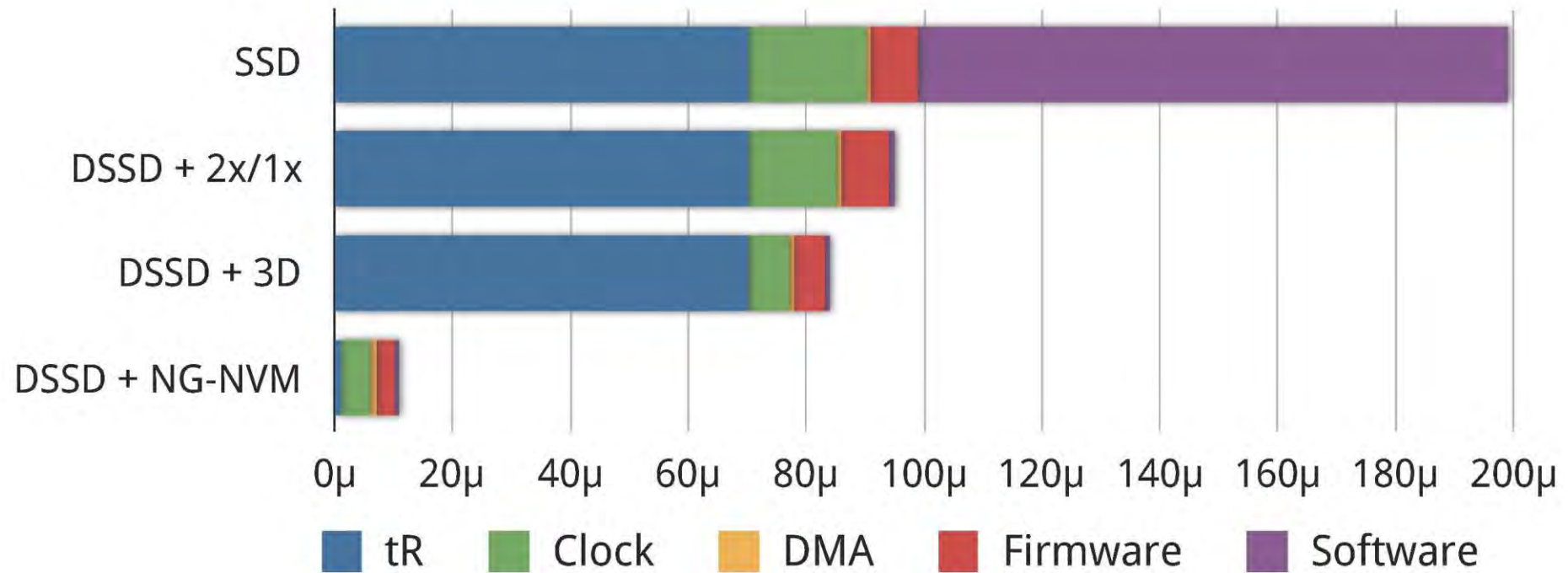
- NVMe SQ/CQ model is a key enabler



DSSD software stack, November 2010



Components of I/O latency



Flash failure modes

- Disk drives fail catastrophically
 - and the whole drive fails
- Flash degrades slowly over time
 - bit error rate eventually exceeds ECC capacity
 - individual cells fail randomly
- Vastly more discrete devices
 - 18,000 NAND die vs. array of 25 SSDs
 - enables new data protection model

Multi-Dimensional RAID

- RAID-6 across channels within each FM
 - Survives channel failure, random page failure
 - Most page-level errors fixed here, inside the FM, so that RAID reconstruction does not involve CM
- RAID-6 across FMs
 - Survives whole-FM failure and field upgrade
 - Repairs errors that channel-level RAID could not
- Layered RAID-of-RAID is pretty good, but we can do much better

1D RAID Reconstruction

- Normal RAID-6 recovers from 1-2 bad blocks

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
D1	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
D2	2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
P3	3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
Q4	4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7

1D RAID Reconstruction

- Read of <2, 5> fails.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

1D RAID Reconstruction

- Read rest of row or column. We'll try row D2.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
D1	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
D2	2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
P3	3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
Q4	4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7

1D RAID Reconstruction

- Fix <2, 5> using double parity. Done.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
D1	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
D2	2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
P3	3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
Q4	4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7

1D RAID Reconstruction

- Normal RAID-6 cannot handle triple failure

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

1D RAID Reconstruction

- Can't recover <2, 5> using row parity.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

1D RAID Reconstruction

- Can't recover <2, 5> using column parity.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
D1	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7
D2	2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
P3	3,0	3,1	3,2	3,3	3,4	3,5	3,6	3,7
Q4	4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7

2D RAID Reconstruction

- 2D RAID can recover $\langle 2, 5 \rangle$. First, get row D0.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

2D RAID Reconstruction

- Fix row D0 using row parity.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

2D RAID Reconstruction

- Fix column D5. We now have $\langle 2, 5 \rangle$. Done.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

2D RAID Reconstruction

- Incredibly powerful. This is recoverable:

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

2D RAID Reconstruction

- Read row D0.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

2D RAID Reconstruction

- Fix row D0. No other row can be fixed.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

2D RAID Reconstruction

- Read column D1.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

2D RAID Reconstruction

- Fix column D1.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

2D RAID Reconstruction

- Read and fix column D2.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

2D RAID Reconstruction

- Read and fix column D3.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

2D RAID Reconstruction

- Read and fix columns D5, P6, Q7.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

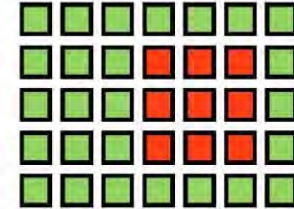
2D RAID Reconstruction

- Fix rows D1, D2, D3, Q4.

	D0	D1	D2	D3	D4	D5	P6	Q7
D0	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	0, 6	0, 7
D1	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7
D2	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7
P3	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7
Q4	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7

Multi-Dimensional RAID Properties

- For D-dimensional RAID with M-way parity in each dimension, the smallest unsolvable failure is a perfect grid of $(M+1)^D$ points



- Much larger asymmetric failures are solvable
- By contrast, linear RAID with the same space overhead ($M \cdot D$) cannot solve anything $> M \cdot D$
- Multi-dimensional RAID takes advantage of the inherent sparseness of higher-dimensional space
- Algorithm must be explicitly multi-dimensional: naive RAID-of-RAID layering can only solve $M \cdot D$

2D RAID Mathematical Properties

Parity equations:

$$P = \sum_x D_x \quad Q = \sum_x D_x \cdot g^x \quad R = \sum_x D_x \cdot g^{2x}$$

More generally,

$$P_d = \sum_x D_x \cdot g^{dx}$$

In an xy grid,

$$\text{Row } y \text{ } P_d = \sum_x D_{xy} \cdot g^{dx}$$

$$\text{Col } x \text{ } P_e = \sum_y D_{xy} \cdot g^{ey}$$

Parity of parity:

$$\text{Row } P_d \text{ of Col } P_e = \sum_x (\sum_y D_{xy} \cdot g^{ey}) \cdot g^{dx}$$

$$\text{Col } P_e \text{ of Row } P_d = \sum_y (\sum_x D_{xy} \cdot g^{dx}) \cdot g^{ey}$$

Same answer either way!

$$P_{de} = \sum_{x,y} D_{xy} \cdot g^{dx+ey}$$

Therefore, P_{de} can be used to solve both rows and columns

2D RAID Parity Grid

D_{00}	D_{10}	D_{20}	$\sum_x D_{x0}$	$\sum_x D_{x0} \cdot g^x$
D_{01}	D_{11}	D_{21}	$\sum_x D_{x1}$	$\sum_x D_{x1} \cdot g^x$
D_{02}	D_{12}	D_{22}	$\sum_x D_{x2}$	$\sum_x D_{x2} \cdot g^x$
D_{03}	D_{13}	D_{23}	$\sum_x D_{x3}$	$\sum_x D_{x3} \cdot g^x$
$\sum_y D_{0y}$	$\sum_y D_{1y}$	$\sum_y D_{2y}$	$\sum_{x,y} D_{xy}$	$\sum_{x,y} D_{xy} \cdot g^x$
$\sum_y D_{0y} \cdot g^y$	$\sum_y D_{1y} \cdot g^y$	$\sum_y D_{2y} \cdot g^y$	$\sum_{x,y} D_{xy} \cdot g^y$	$\sum_{x,y} D_{xy} \cdot g^{x+y}$

2D Parity Space: $1 - (16/18) \cdot (30/32) = 17\%$

