FMS18 – Invited Session 101-B1 "Hardware Acceleration Techniques for NVMe-over-Fabric"

Paper Abstract: The move from direct-attach to Composable Infrastructure is being driven by large datacenters seeking increased business agility combined with lower **TCO**. This requires new remote-attached storage solutions which can deliver extremely high data rates with minimal latency overhead. Unfortunately, the industry-standard embedded processors used in controllers aren't fast enough to manage complex protocols at the required speed. For example, they cannot keep up with the work required to access NVMe SSDs efficiently over an **NVMe-oF** networked infrastructure. The solution is to add accelerators, typically built using an **ASIC**, **FPGA**, or other high-speed hardware. These accelerators offload the processing of protocols such as **RDMA**, **TCP**, and **NVMe**. The result is essentially the same performance for remote storage accessed over a network as for direct-attached storage. The combination provides an optimal blend of high performance, low power, and low cost to yield tremendous CAPEX and OPEX savings in the next-generation datacenter. The technology enables virtually limitless scalability, and will drive dramatically lower TCO for **hyperscale** and as-a-service datacenter applications.

Speakers' Biographies:

Bryan Cowger, with over 25 years of storage industry experience, is VP Sales/Marketing at **Kazan Networks**, a startup developing ASICs that target new ways of attaching and accessing flash storage in enterprise and hyperscale datacenters. Kazan Networks' products utilize emerging technologies such as NVMe and NVMe-oF. Bryan has spent his career defining and bringing to market successful high-performance storage networking ASICs for such protocols as Fibre Channel, SAS, SATA, Ethernet, PCIe, and NVMe. He has been awarded 4 patents in area of storage controller architecture. Before joining Kazan Networks, he was VP Sales/Marketing & Co-Founder at Sierra Logic, a developer of SATA-to-Fibre Channel controllers. He also spent over 10 years as a design engineer at Hewlett-Packard and Agilent Technologies. He holds a BS in Electrical Engineering from UC San Diego.

Hardware Acceleration of Storage for Composable Infrastructure

Bryan Cowger

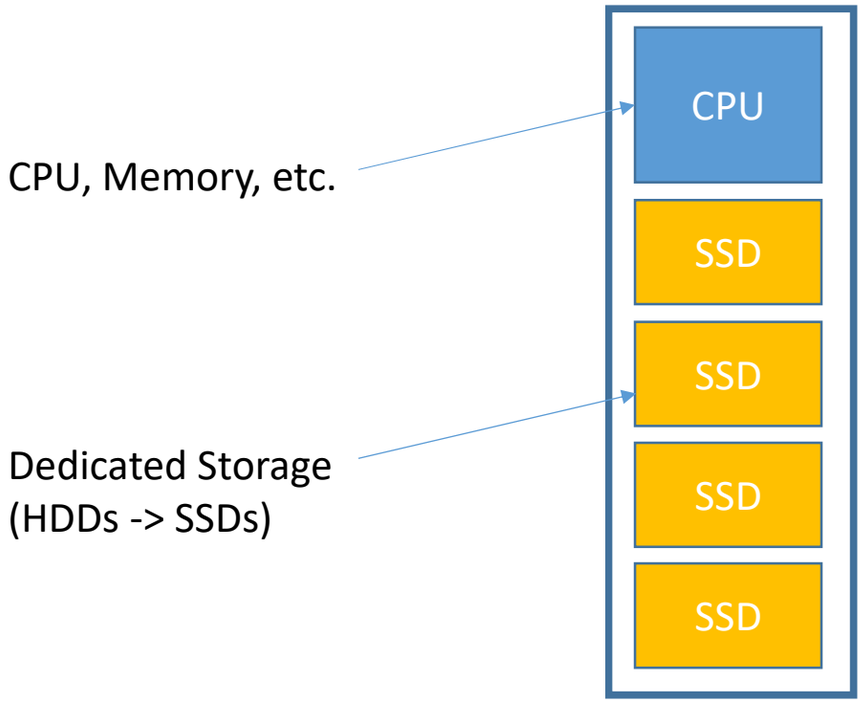
VP, Kazan Networks

Agenda

- What is Composable Infrastructure?
- Challenges for CI using existing technologies
- NVMe over Fabrics™ overview

- Sidebar example: Metal Working Machinery (?!)
- Implementation example: Network TCP Engine
- Architectural Comparisons of available solutions
- Practical examples and results/benefits of HW acceleration

Today's "Shared Nothing" Model
a.k.a. DAS



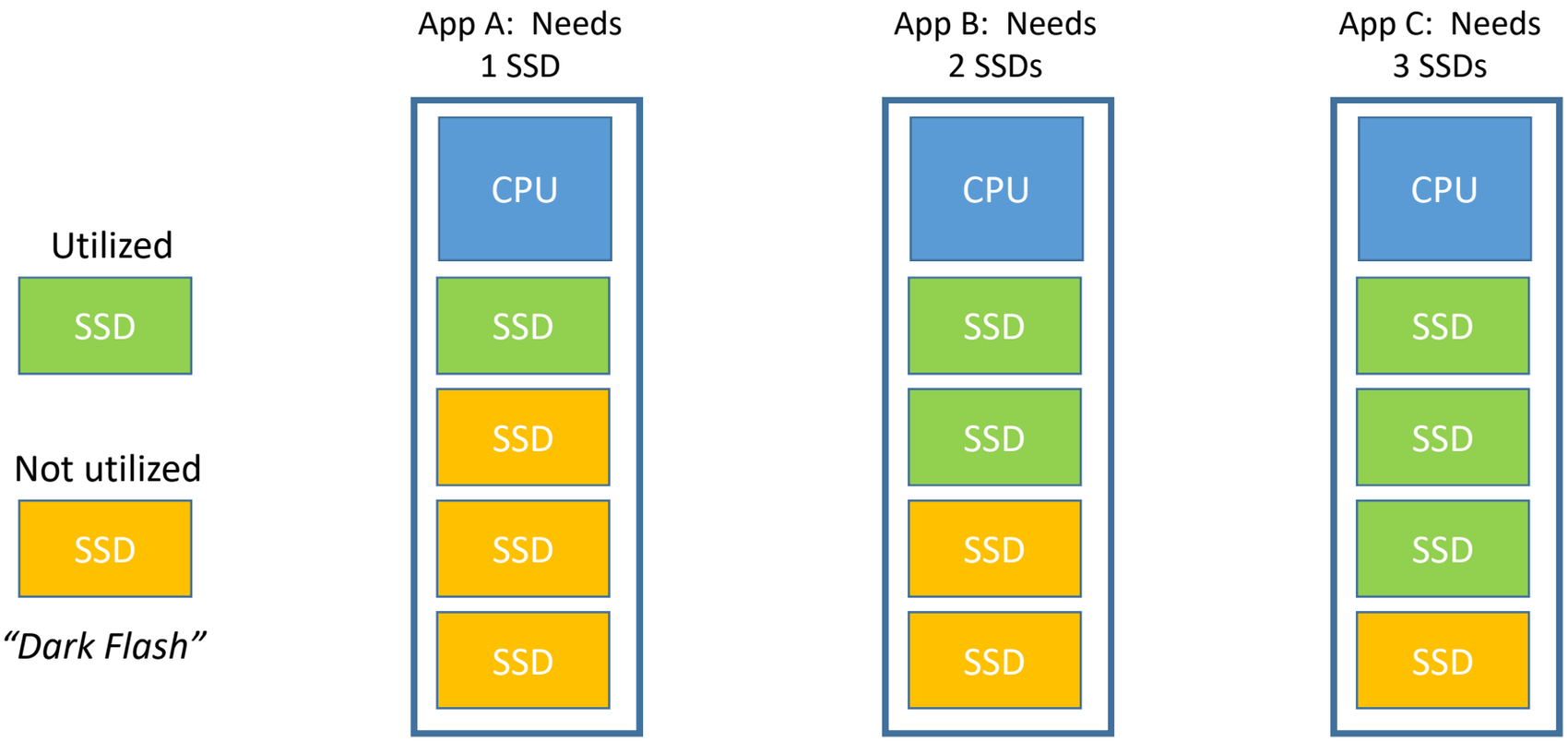
Challenges:

- Forces the up-front decision of how much storage to devote to each server.

- Locks in the compute:storage ratio.

Shared Nothing Model

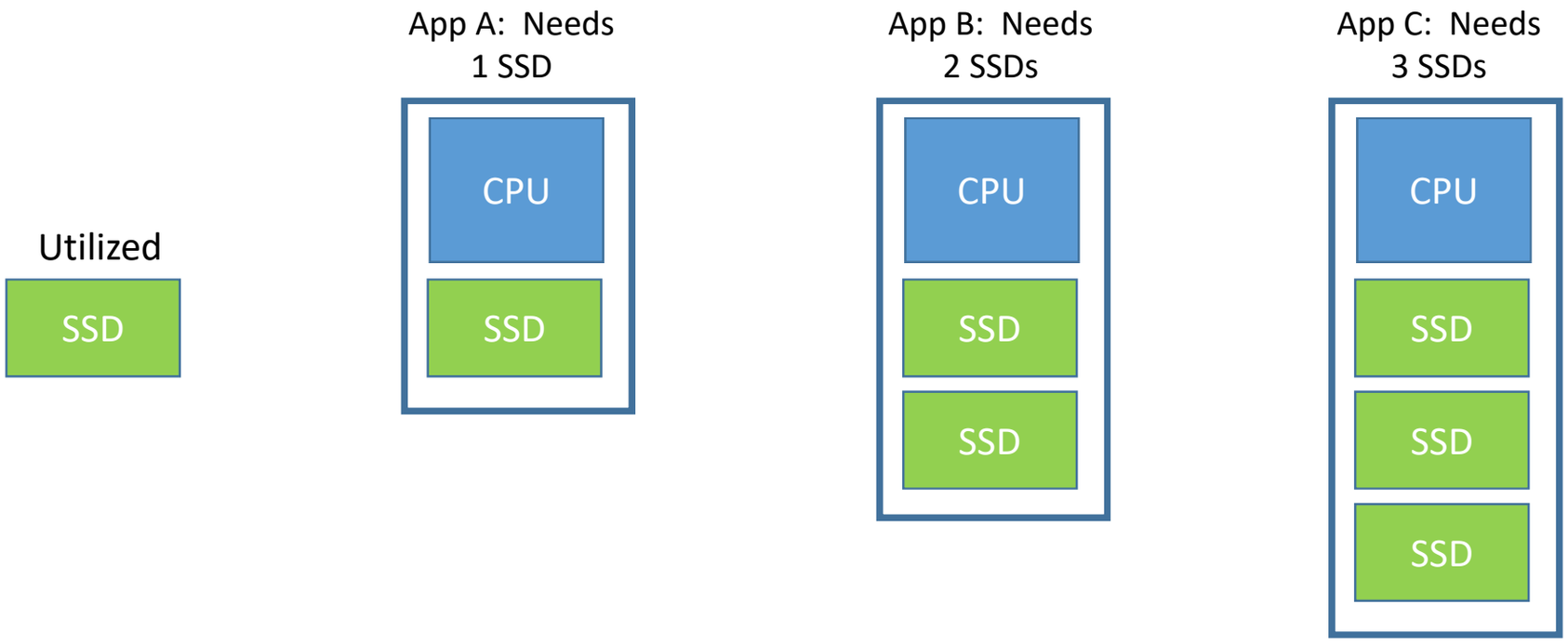
Option A: One Model Serves All Apps



Net utilization: 6 SSDs out of 12 = 50%

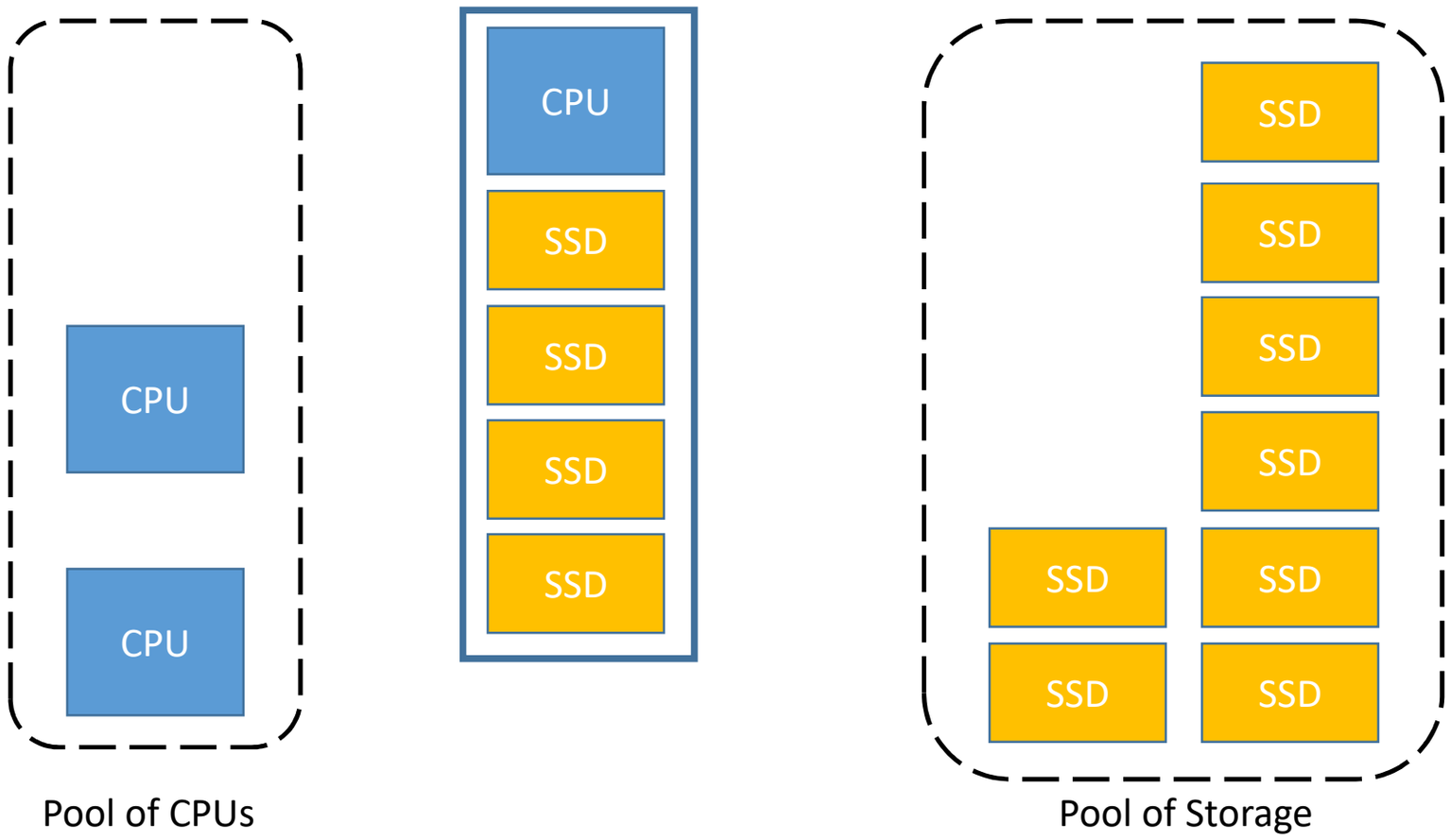
Shared Nothing Model

Option B: Specialized Server Configurations

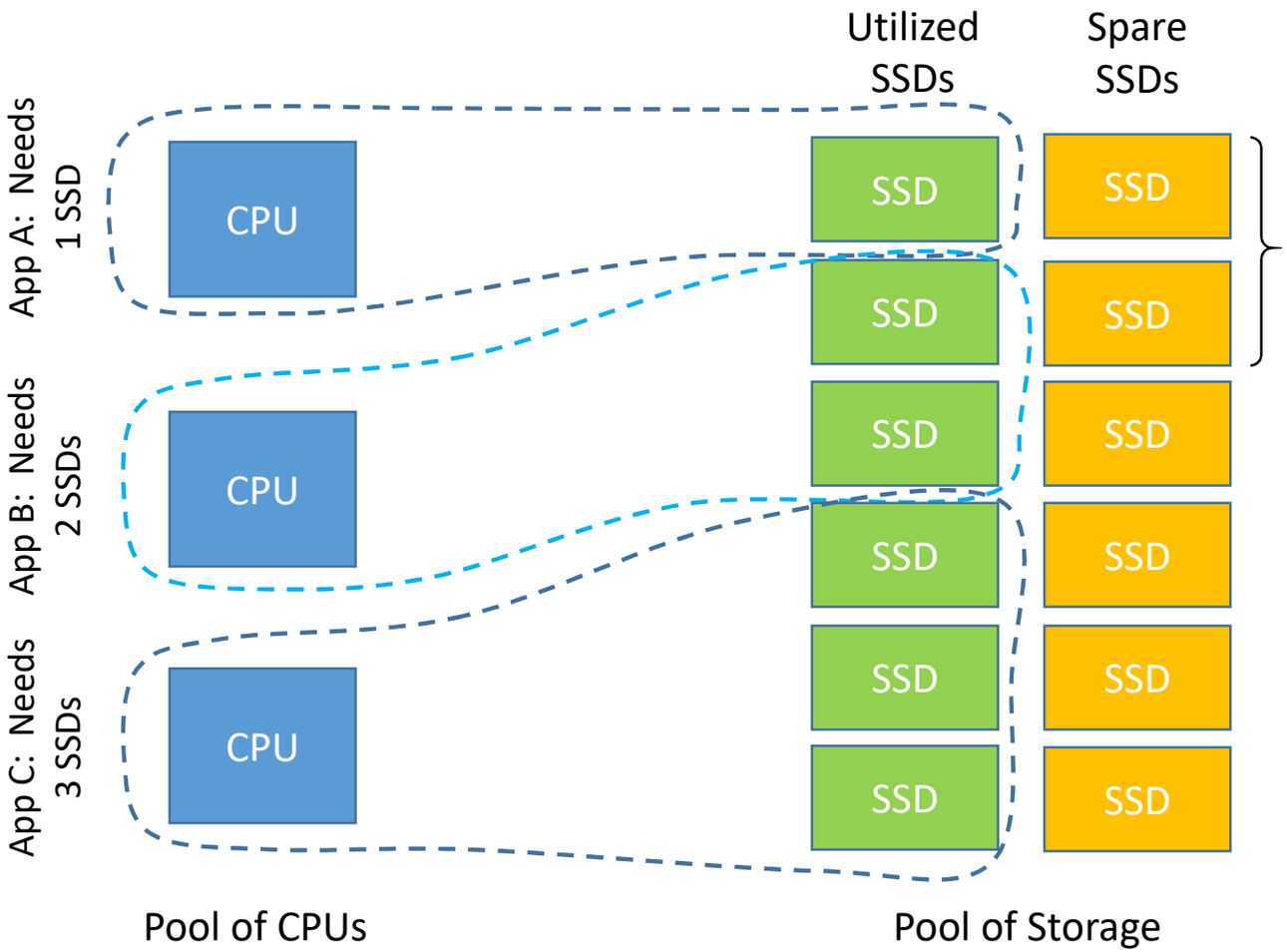


Dark Flash eliminated, but limits ability and future app deployments

Disaggregated Datacenter



The Composable Datacenter



Spares / Expansion Pool

- Minimize *Dark Flash*!
- Buy them only as needed
- Power them only as needed

The Composable Datacenter

Real Savings?

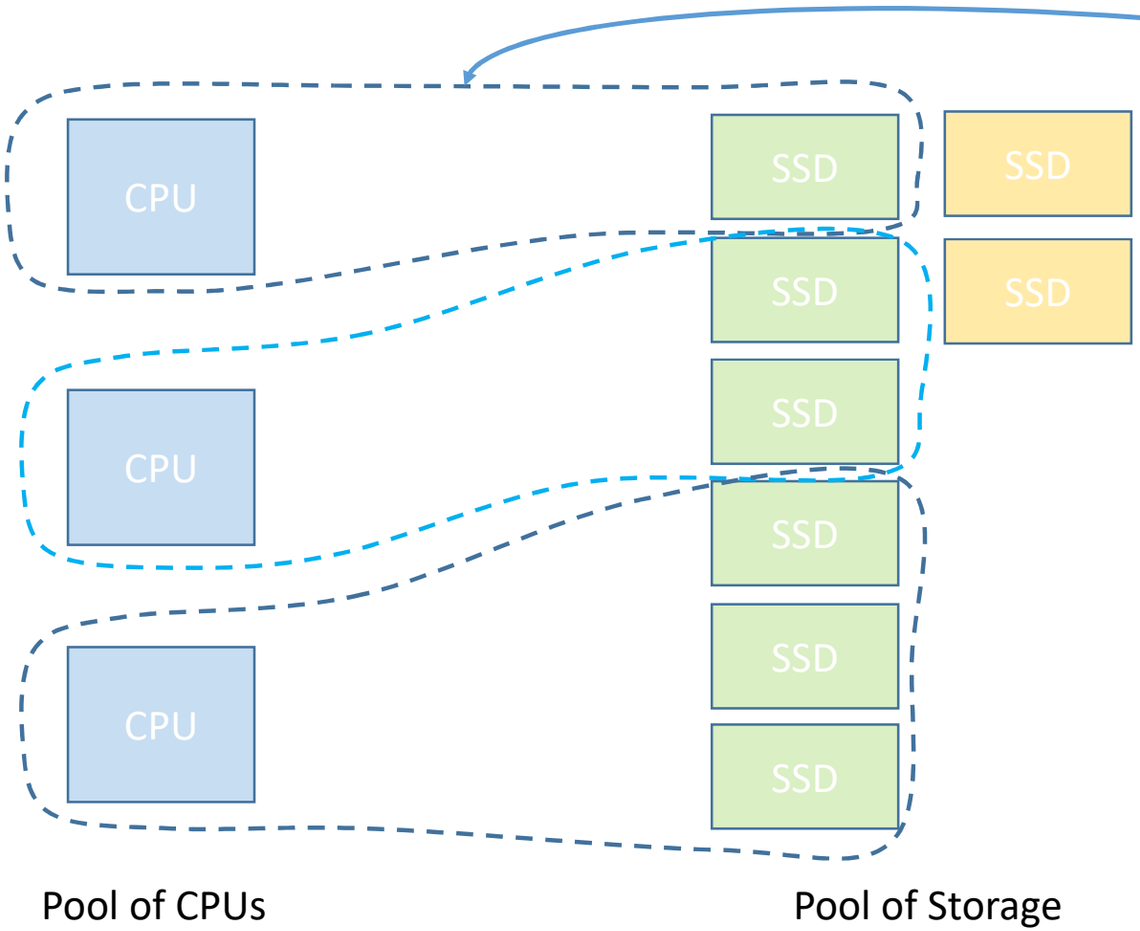
- Example:
  - 100k servers
  - 1M SSDs

Assume 40% increase in storage utilization

- <u>CapEx:</u>
  - 1M SSDs * 60% = 600k SSDs
  - Savings of 400k SSDs
  - $300 per SSD
  - $120M savings
- <u>OpEx:</u>
  - Not powering 400k SSDs
  - Assume ~10W per SSD
  - Assume $0.10 KWH
  - $10k savings per day
  - $3.5M savings per year

The Composable Datacenter

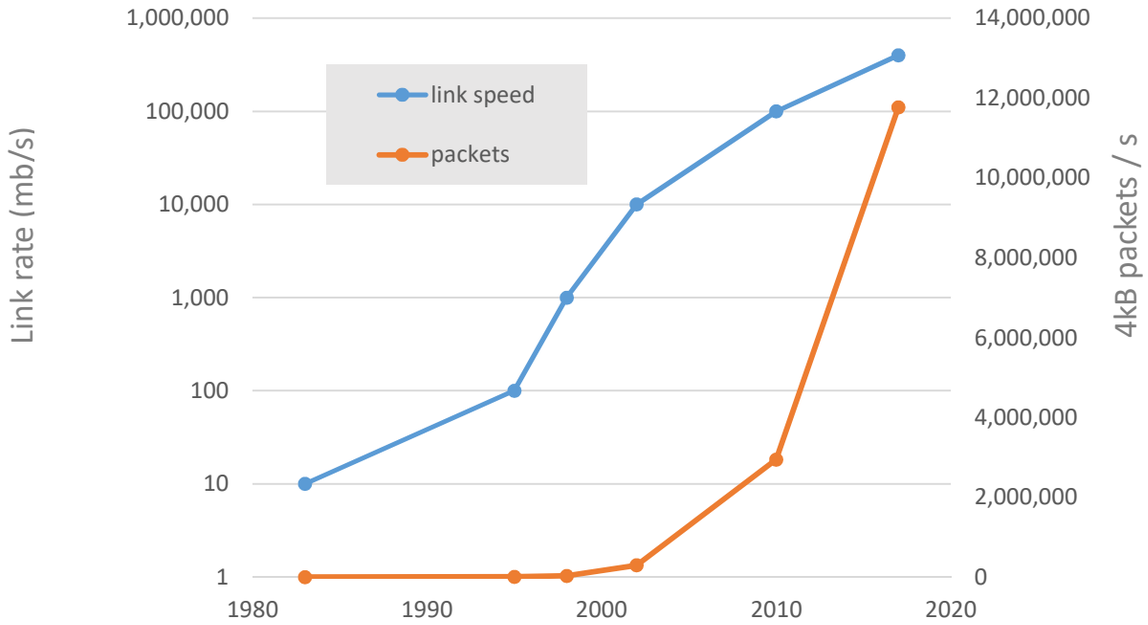
Based on NVMe-oF



- Software-defined DataCenter
- Primary application for NVMe-oF™
- *“Infrastructure as Code”*
- Could be any “fabric”
  - Ethernet
  - Fibre Channel
  - Infiniband
  - Next-gen…
- Hyperscalers focusing on Ethernet

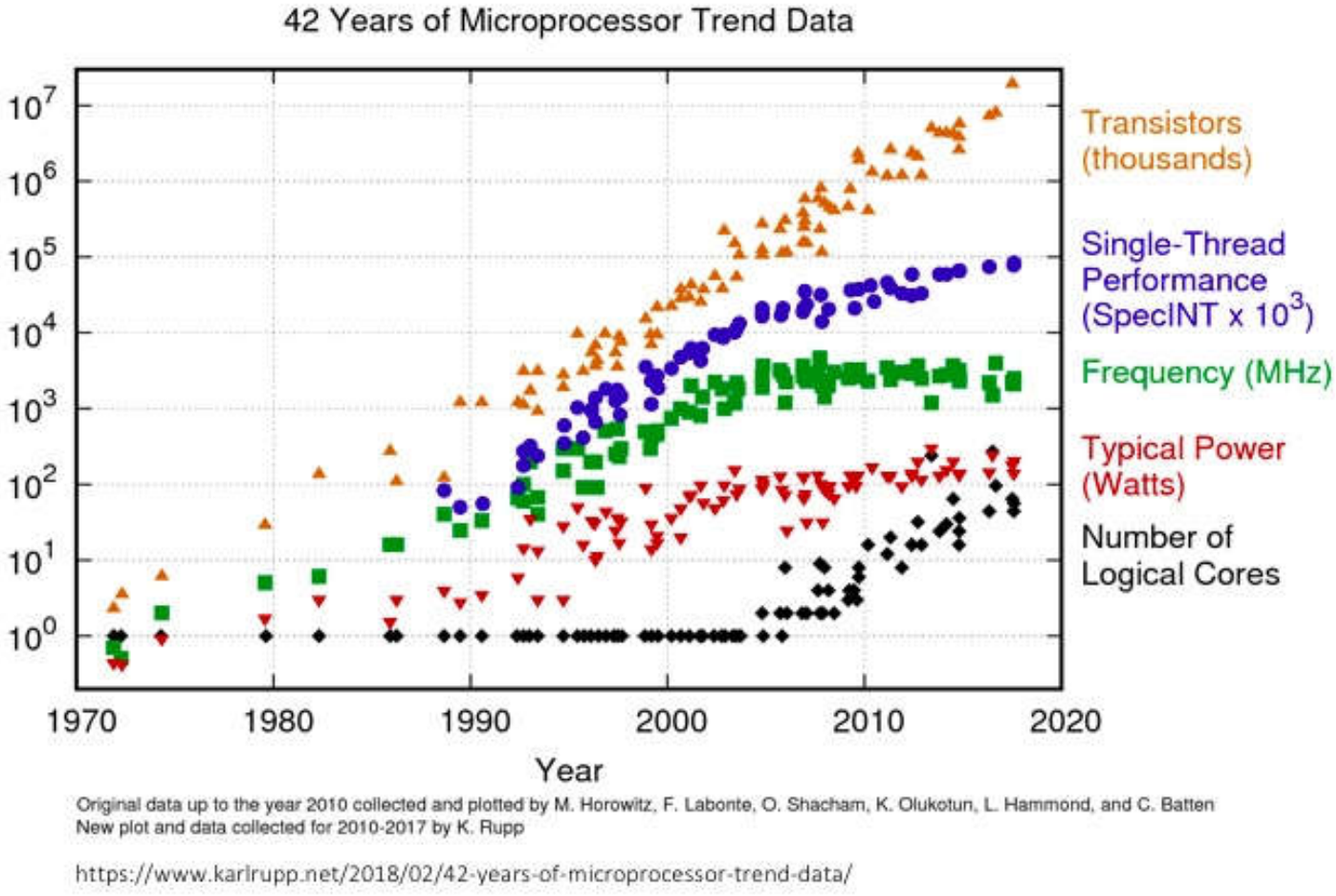
Ethernet Roadmap

"Initial Standard Completed"



Link rate (mb/s)	4kB packets / sec
10	294
100	2,941
1,000	29,412
10,000	294,118
100,000	2,941,176
400,000	11,764,706

The Processor Challenge



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

Summary of Challenges

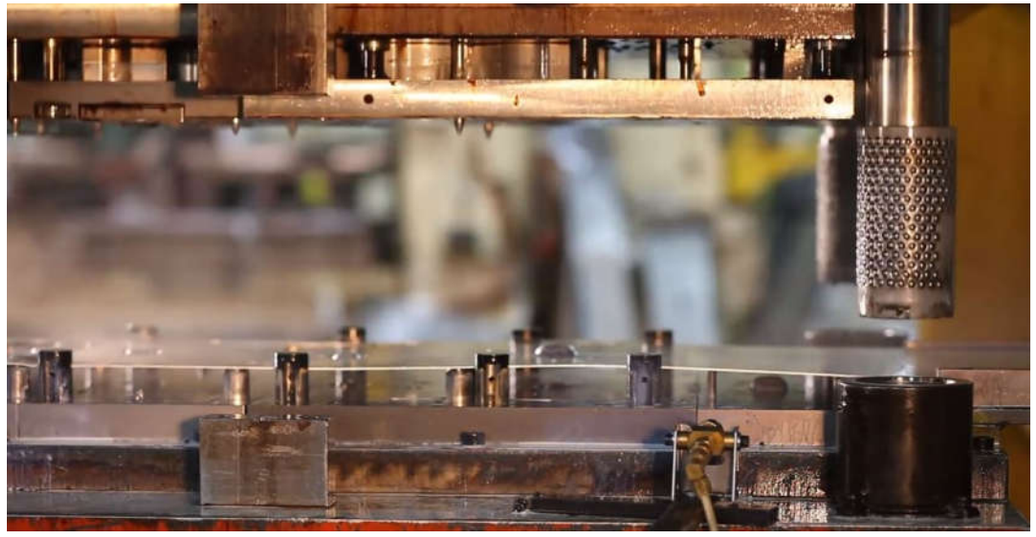
- Ethernet speeds going up…
- Embedded processing capabilities plateauing…
- Compute and storage disaggregating…
- … while storage media latencies are decreasing

- One solution: Speed up how networking controllers are built by basing them on more specialized, dedicated hardware.

Why Dedicate Hardware?

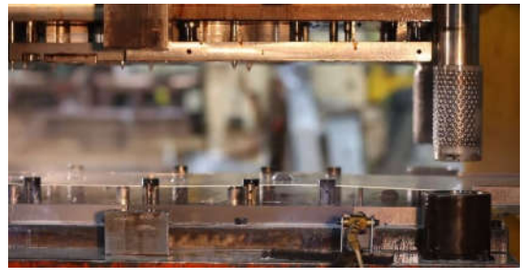
- Metal-stamping machine examples

Programmable



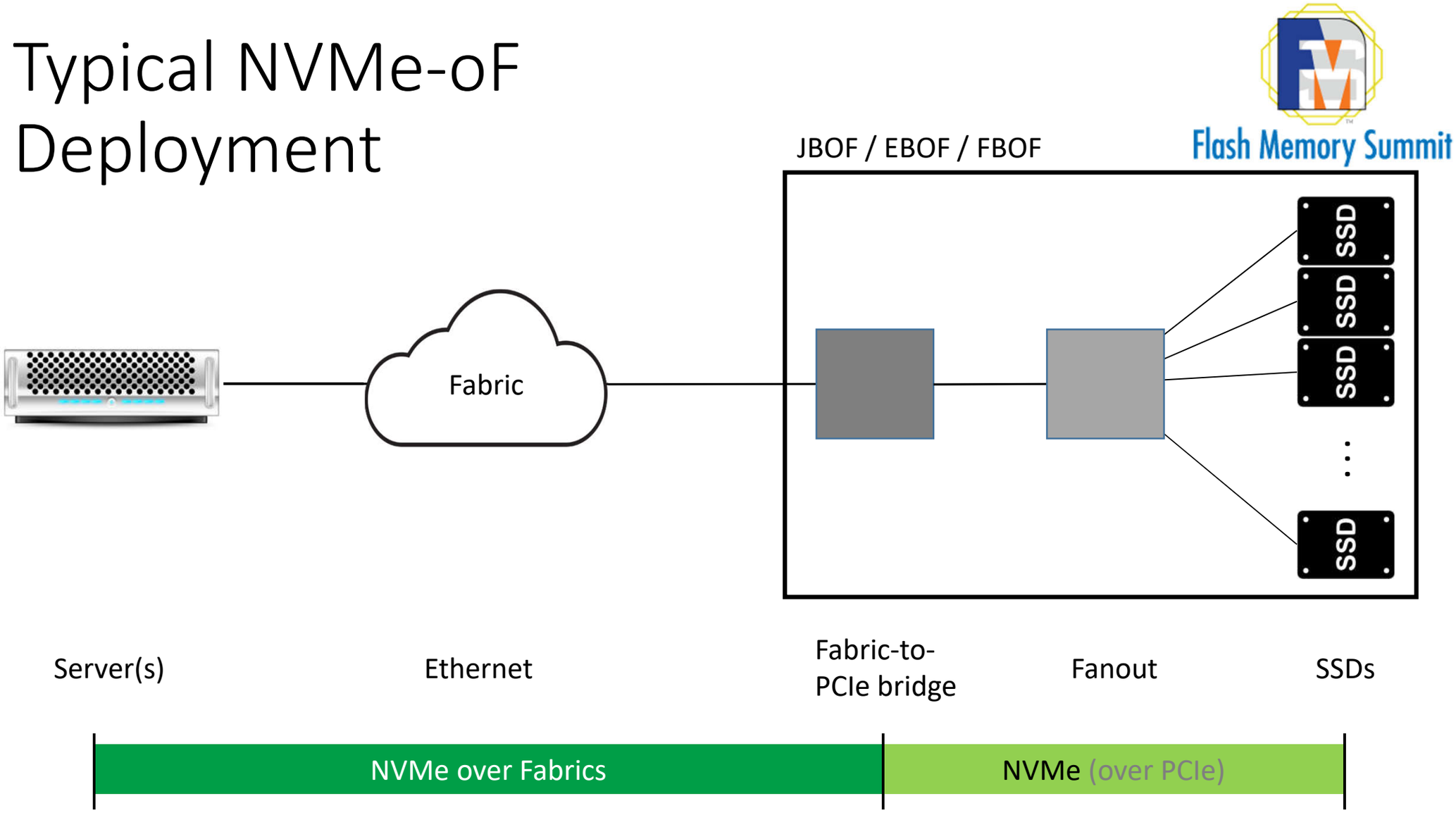
Progressive Stamping

Two Machines: Side-by-Side

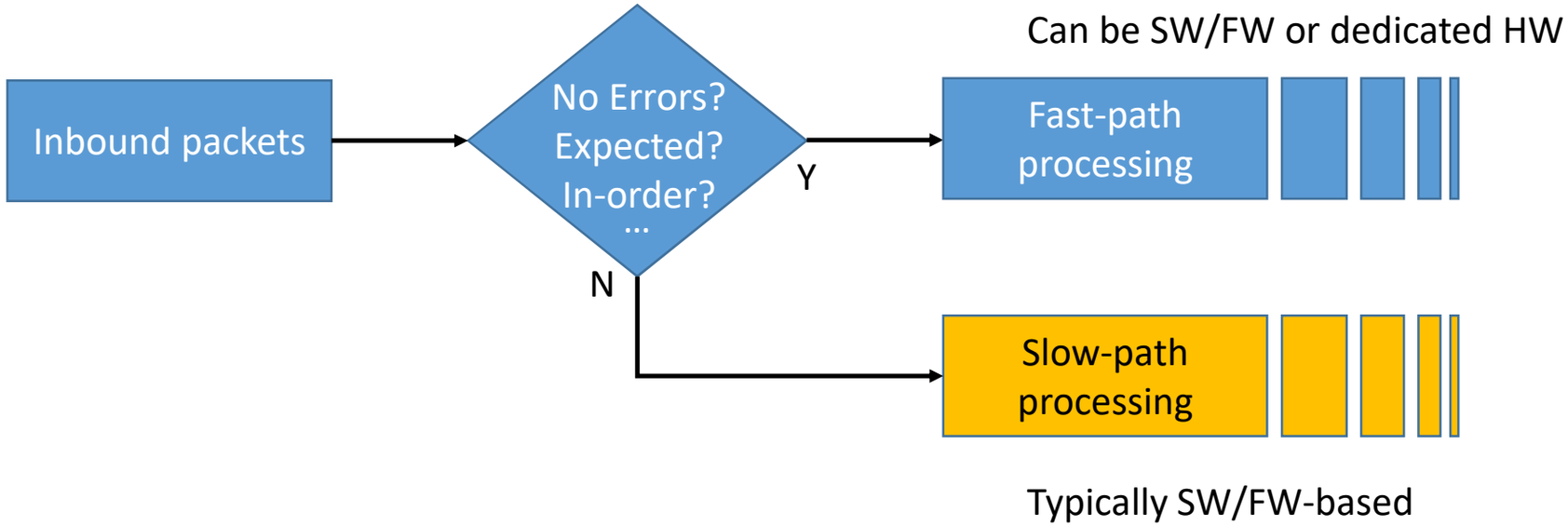


• Versatility:	✓ Programmable	Hard-coded
• Bandwidth:	12 parts / 8 minutes	✓ 1 part / 2 seconds
• Latency:	8 minutes	✓ 20 seconds
• Size:	18’ x 17’	✓ 6’ x 3’
• Weight:	14 tons	✓ 2 tons
• Power:	9kW	✓ 2kW

Typical NVMe-oF Deployment



Networking Controller
Typical Architecture of Inbound Path



Inbound TCP Control Engine Example

- Data structure at right used to track status of each TCP frame
- Each inbound frame header must be compared against 320 bits of control information
- Fast-path / slow-path decision to be made
- Various fields must be updated and written back

DWord	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	TCP_State				SA	RC	port		dupack_cnt				retx_cnt				snd_scale				EE	SD	RT	UL	WF	KP	CP	R1	LS	PL	tmr_t	
1	snd_wnd																rsvd											ES	flags			
2	rcv_nxt																															
3	snd_una																															
4	snd_nxt																															
5	snd_wl1																															
6	rsvd																															
7	snd_max																															
8	snd_cwnd																															
9	t_rtseq																															
10	recovery																															
11	last_ack_sent																															
12	rcv_wnd																snd_ssthresh															
13	max_sndwnd																mss															
14	rto								t_rtt								t_srtt							t_rttvar								
15	t_idle																tsd_tail															
16	tsd_una																tsd_nxt															
17	WinUpdate Link																Port Pause Link															
18	rcv_lost																															
19	rcv_nxt2																															

TCP Network Control Block Data Structure

Inbound TCP Parsing Example

Software-based algorithm:

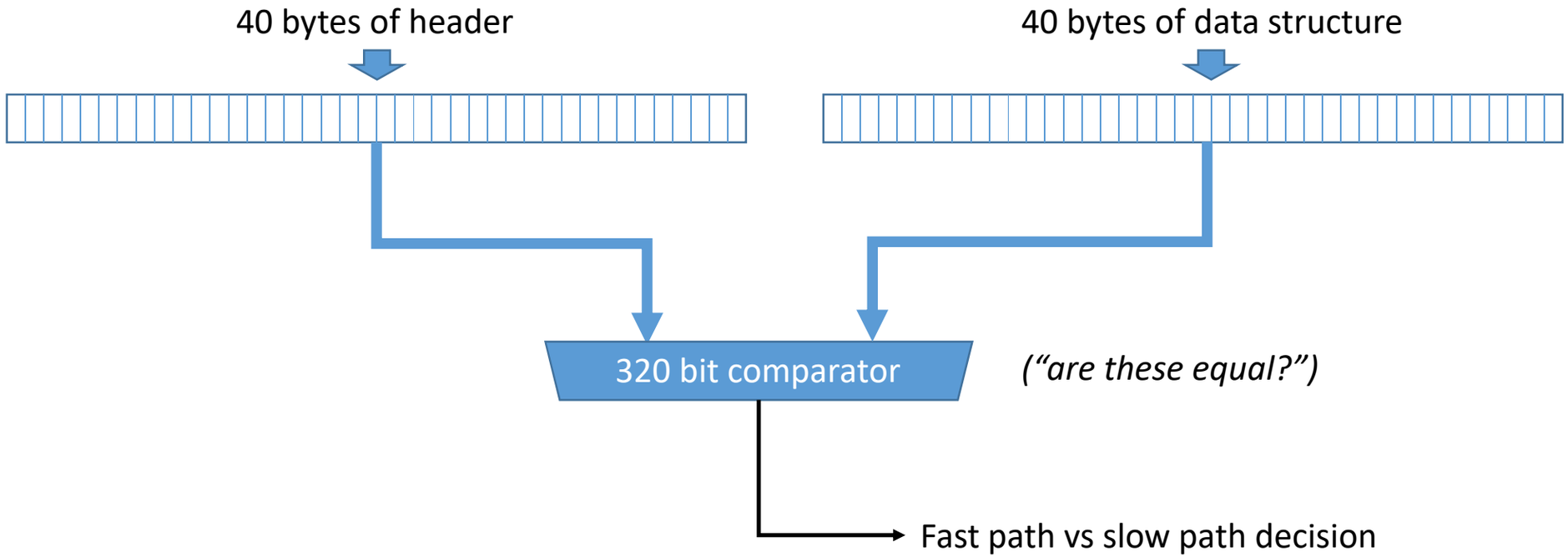
- Loop 5 times:
  - Read in 64 bits from header (1 clock)
  - Read in 64 bits from data structure (1 clock if in TCM; 10+ clocks if in DRAM)
  - Make decision on multiple fields (2 or more clocks)
- Once entire data structure / header is processed, write back necessary information (5+ clocks)
- Take action to move header information to next part of protocol processing (e.g. NVMe) (5+ clocks)

- Total: 30-50 clock cycles
- (2X higher if a 32-bit processor)

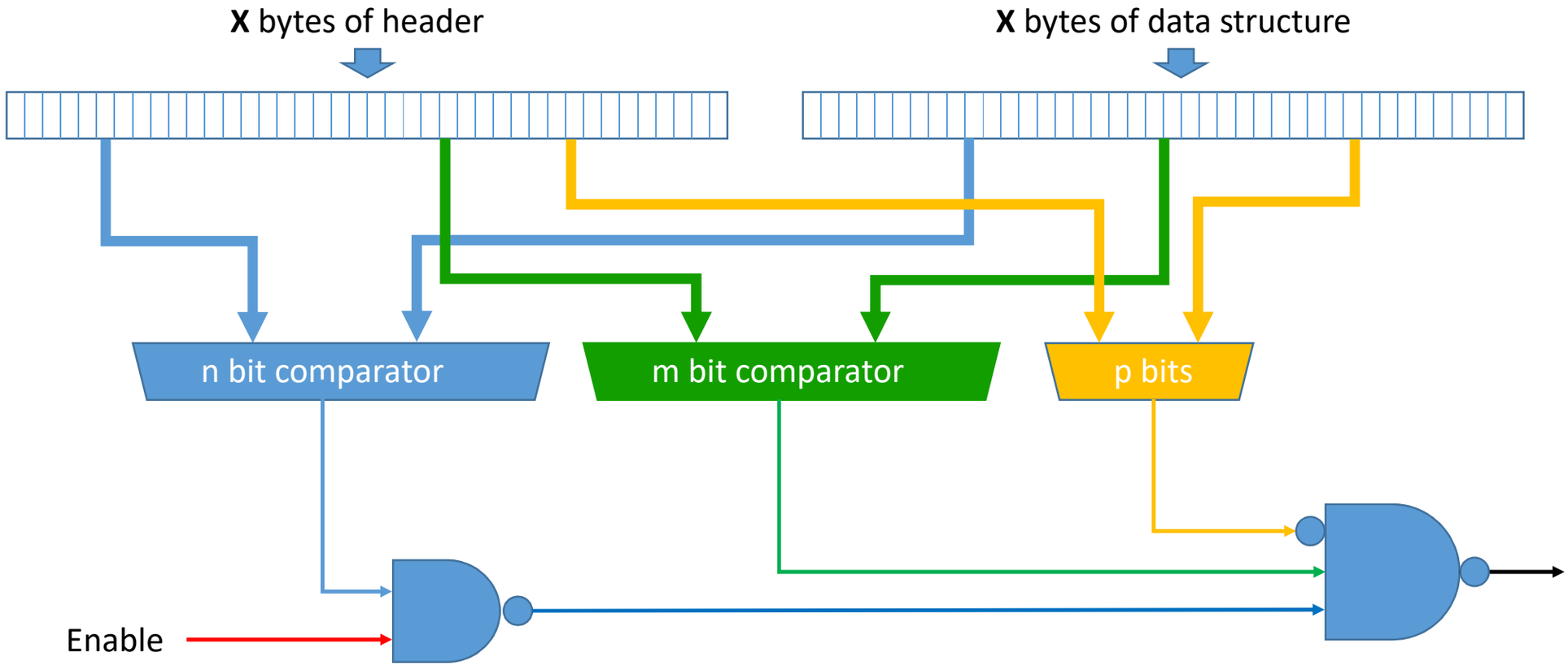
Hardware-based algorithm:

DWord	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	TCP_State				SA	RC	port		dupack_cnt				retx_cnt				snd_scale				EE	SD	RT	UL	WF	KP	CP	R1	LS	PL	tmr_t	
1	snd_wnd																rsvd											ES	flags			
2	rcv_nxt																															
3	snd_una																															
4	snd_nxt																															
5	snd_wl1																															
6	rsvd																															
7	snd_max																															
8	snd_cwnd																															
9	t_rtseq																															
10	recovery																															
11	last_ack_sent																															
12	rcv_wnd																snd_ssthresh															
13	max_sndwnd																mss															
14	rto								t_rtt								t_srtt				t_rttvar											
15	t_idle																tsd_tail															
16	tsd_una																tsd_nxt															
17	WinUpdate Link																Port Pause Link															
18	rcv_lost																															
19	rcv_nxt2																															

TCP Header Processing Hardware



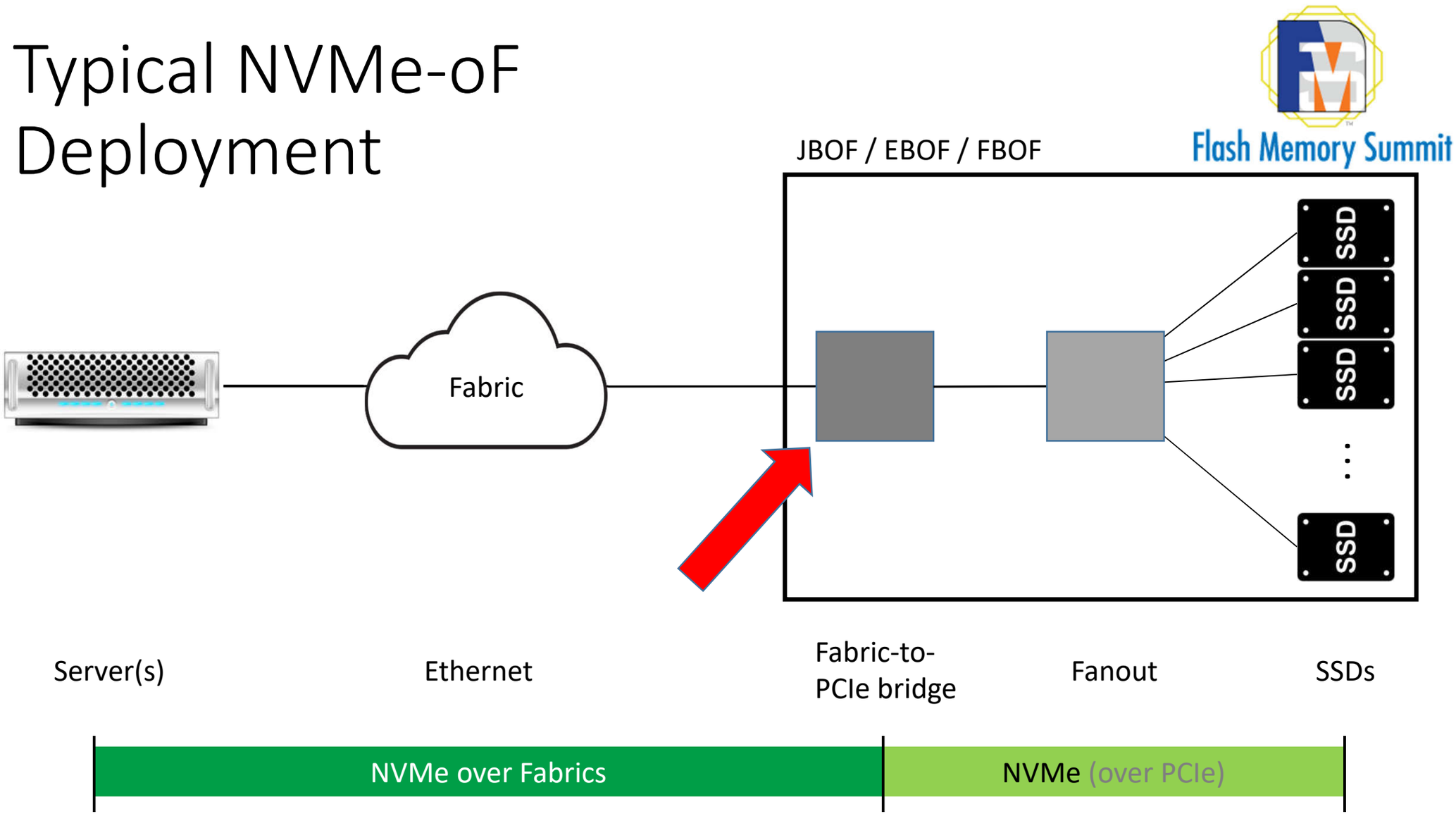
More Complex Decisions



HW-based Finite State Machines (FSM)

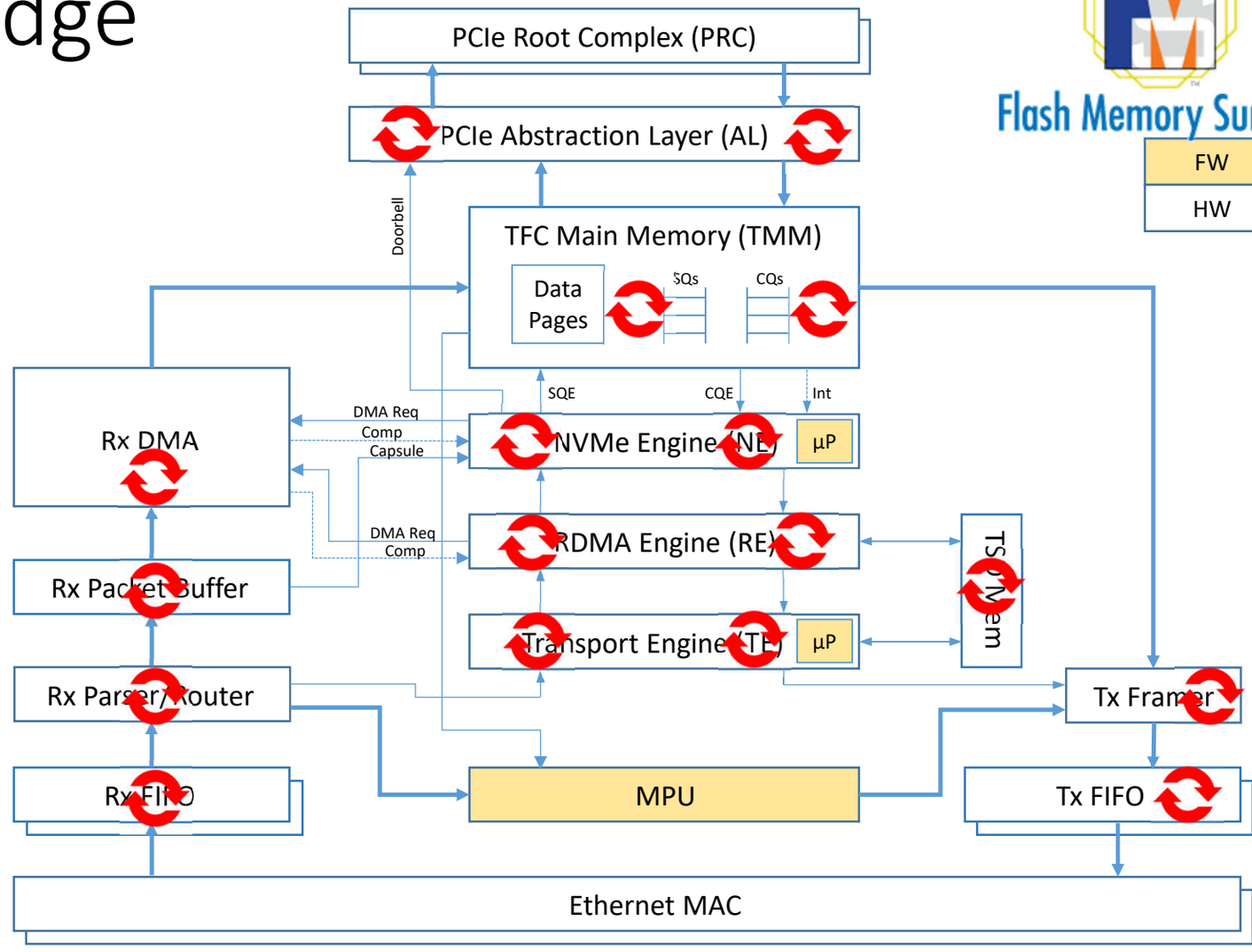
- Essentially just bespoke “very long instruction word” processors
- From Wikipedia: **Very long instruction word** (**VLIW**) refers to instruction set architectures designed to exploit instruction level parallelism (ILP).
  - Whereas conventional central processing units (CPU, processor) mostly allow programs to specify instructions to execute in sequence only, a VLIW processor allows programs to explicitly specify instructions to execute in parallel.
  - This design is intended to allow **higher performance** without the complexity inherent in some other designs.
- Hard-coded state transitions instead of instruction set-based

Typical NVMe-oF Deployment



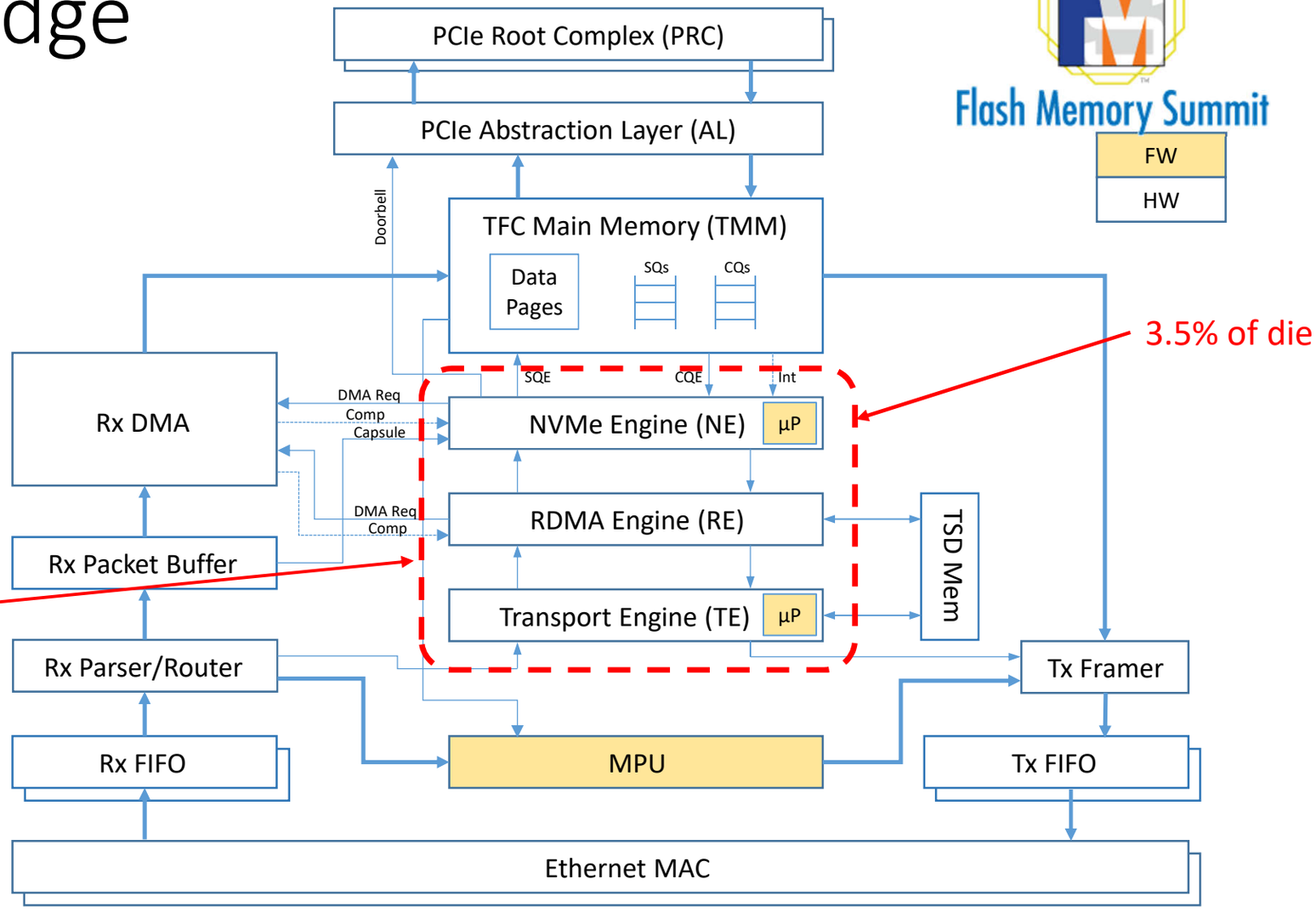
NVMe-oF Bridge Architecture

- Approximately 150 FSMs running in parallel
- Some simple, e.g. buffer management
- Some quite complex:
  - RoCE v1, v2
  - iWARP /TCP
  - TCP
  - NVMe
  - NVMe-oF
- Slow-path implemented in FW

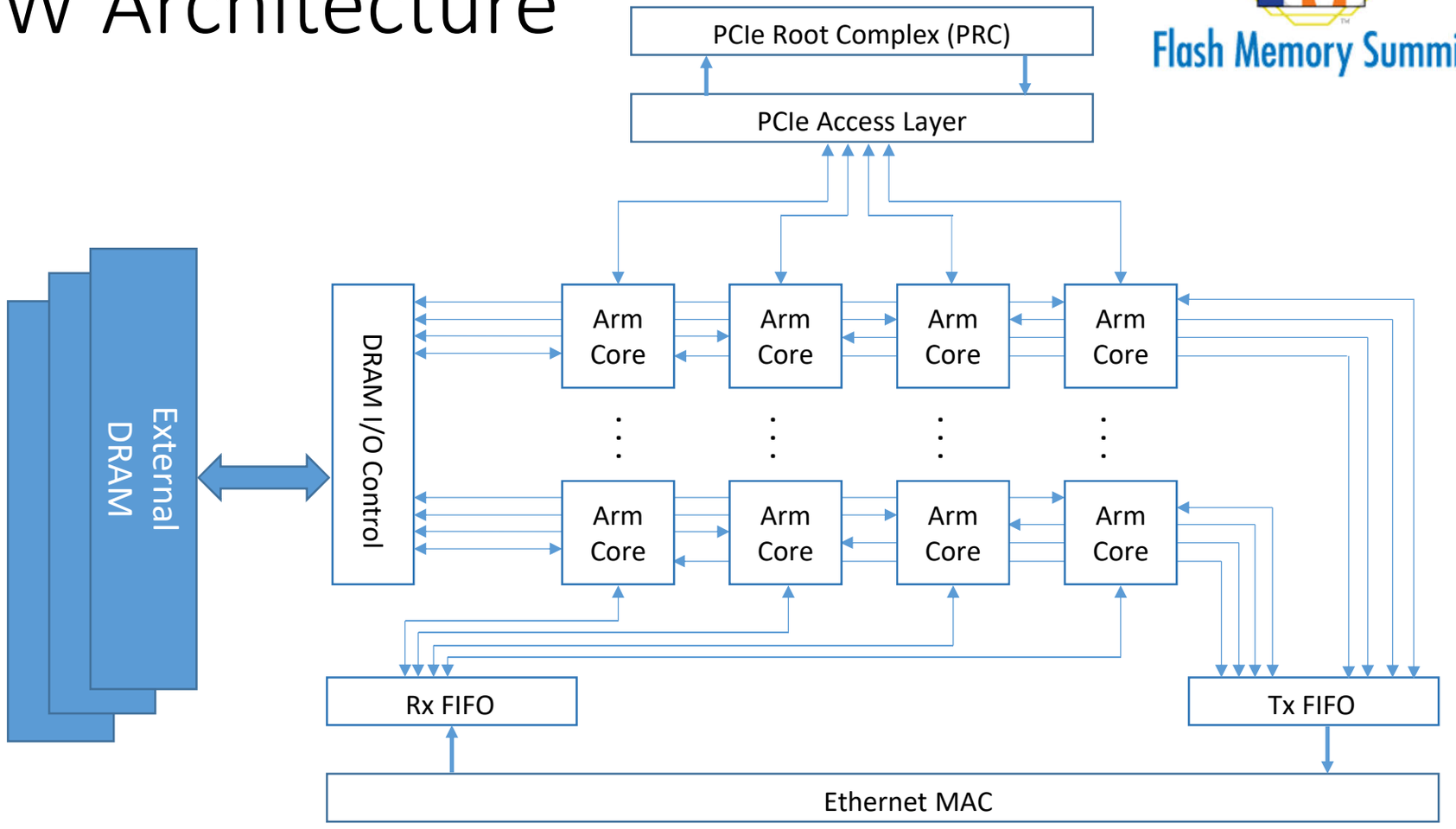


NVMe-oF Bridge Architecture

- Approximately 150 FSMs running in parallel
- Some simple, e.g. buffer management
- Some quite complex:
  - RoCE v1, v2
  - iWARP /TCP
  - TCP
  - NVMe
  - NVMe-oF
- Slow-path implemented in FW



Purely SW Architecture



NVMe-oF Options

High-End

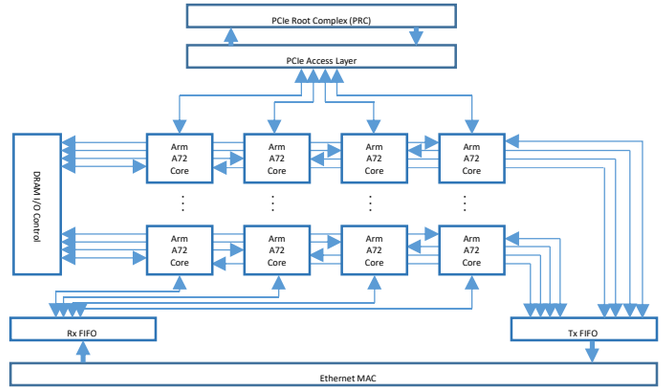
- x86 server-based
- Up to 3.3GHz
- 200W+ / 100Gb

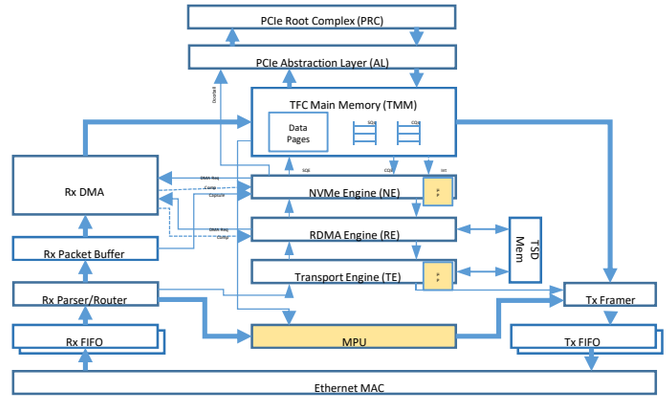
Mid-range

- Network Processors
- Up to 3.0GHz
- 15-20W / 100Gb

Low-power, low-cost

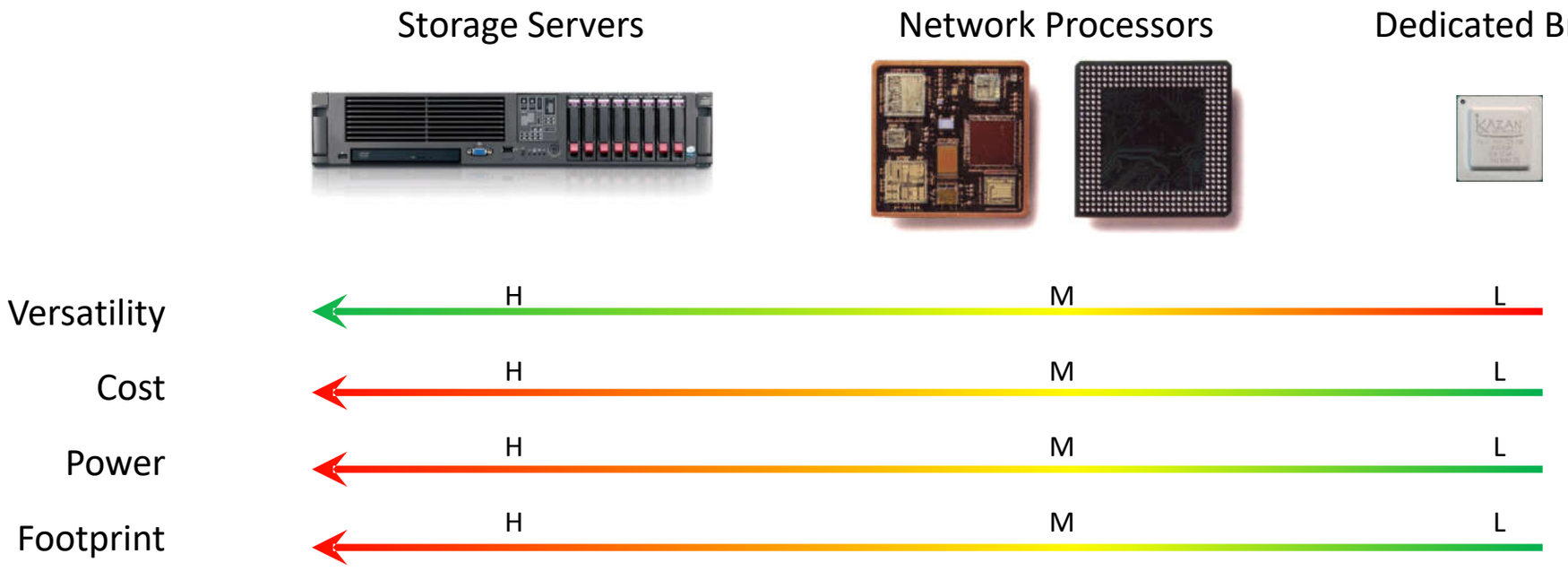
- Dedicated hardware
- ~0.5GHz
- 7W / 100Gb





A Spectrum of Options

- Myriad choices for NVMe-oF / Composable Infrastructure target deployments



Composable Infrastructure

At Hyperscale

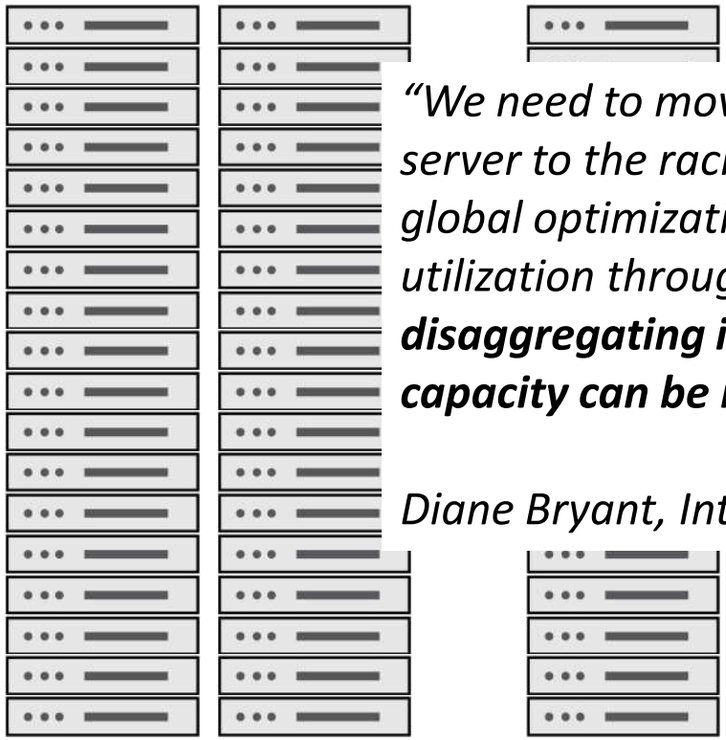
Units of Compute

- Processor
- Memory
- I/O (RDMA, TCP/IP)

Units of Storage

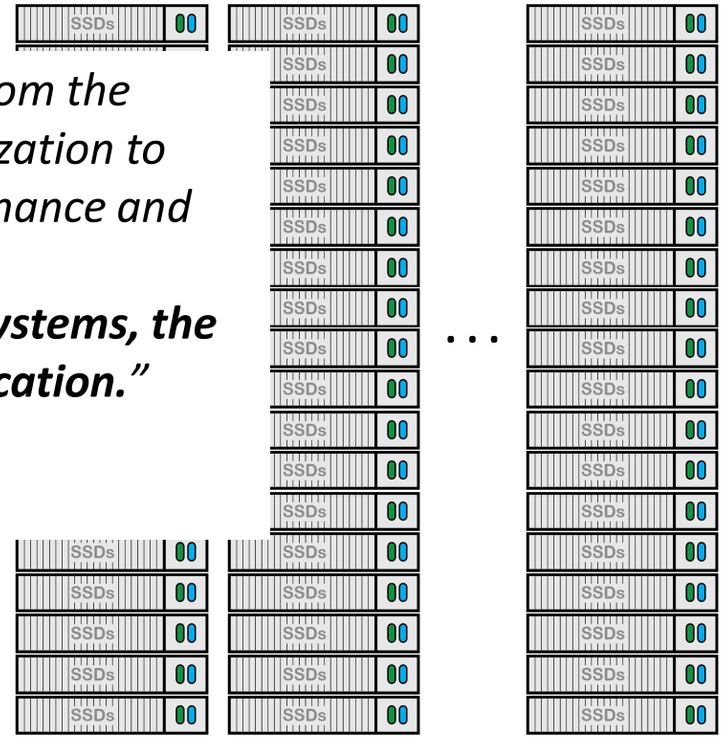
- SSDs
- Fanout
- I/O (RDMA, TCP/IP)
- Manageable

Composable Infrastructure
At Hyperscale



"We need to move the standard unit of compute from the server to the rack; we need to go from local optimization to global optimization. We can deliver higher performance and utilization through the pooling of resources, so ***by disaggregating independent server and storage systems, the capacity can be more finely allocated to the application.***"

Diane Bryant, Intel, IDF16



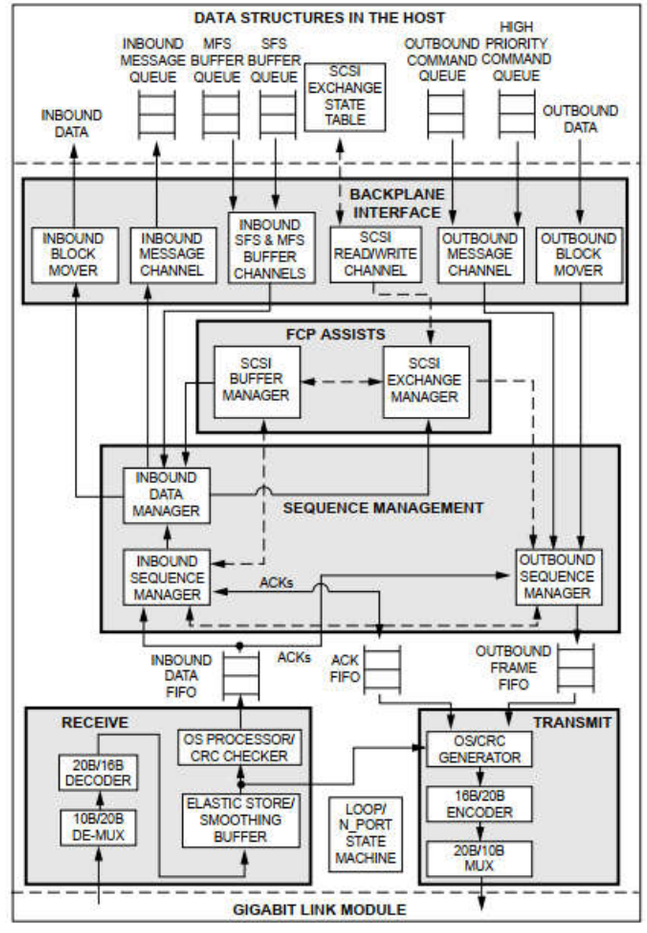
Cost Optimized Compute

Cost Optimized Storage
Storage Utilization > 80%!

Industry Examples

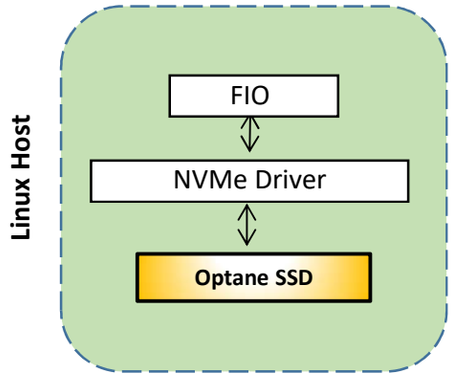
Industry Example: Tachyon™

- Fibre Channel controller family first introduced in the 1990s
- Entirely HW FSM-based
  - No embedded processors!
- High-level protocol (SCSI) in HW
- Complex algorithms (e.g. FC-AL initialization) in HW
- Generated ~$1B in revenue during the family's ~20 year lifecycle



Industry Example: Fuji + Optane™

Latency



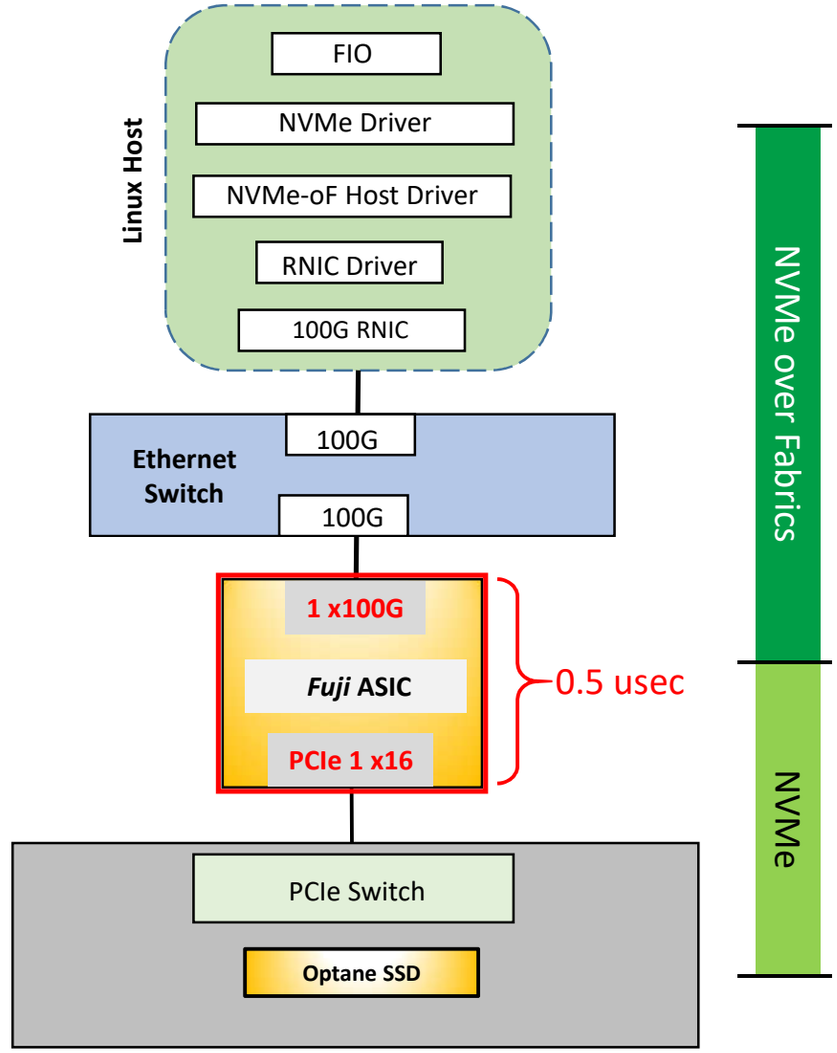
	Read Latency (usec)	
I/O Size	512B	4kB
Native	8.03	8.98

DAS Mode

	Read Latency (usec)	
I/O Size	512B	4kB
NVMe-oF	12.20	13.83

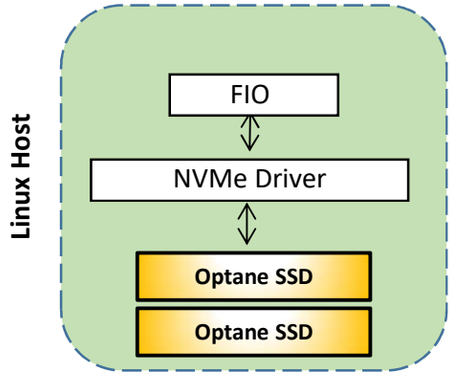
	NVMe-oF Incremental Latency (usec)	
I/O Size	512B	4kB
NVMe-oF	4.18	4.85

Disaggregated Mode



Industry Example: Fuji + Optane™

IOPS / Bandwidth

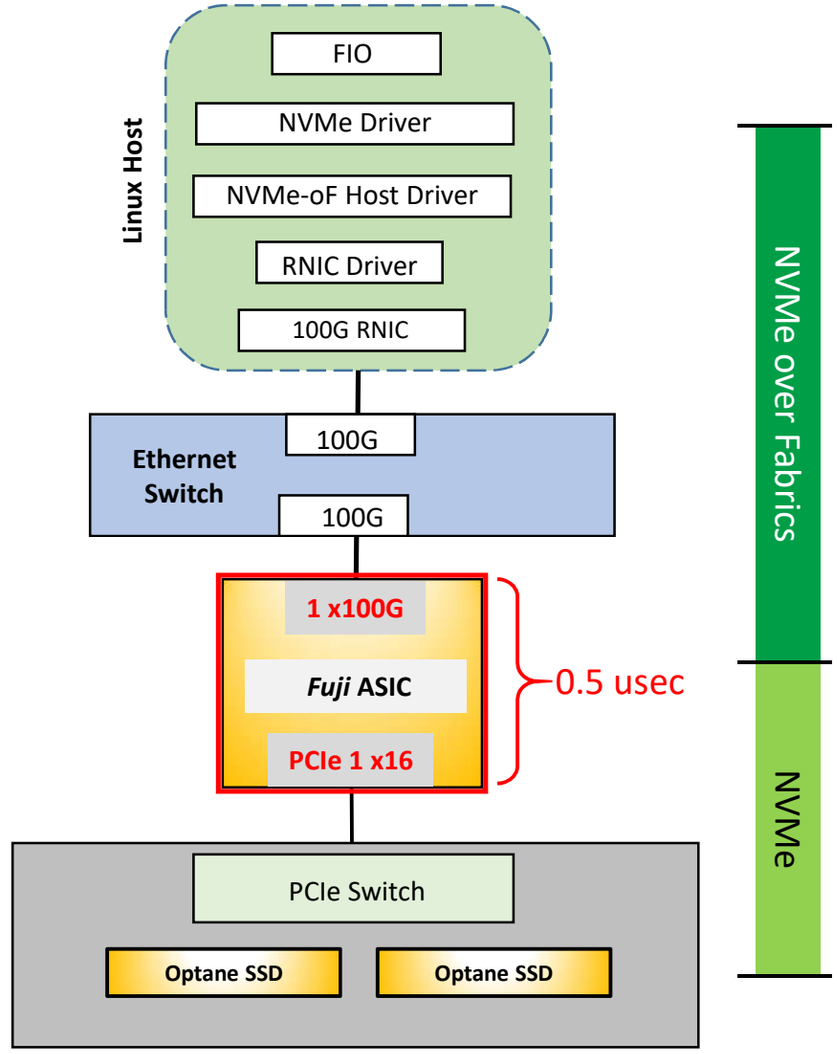


	Optane Native	
	4kB Rnd	128k Seq
Reads	571k IOPS	2.58 GB/s
Writes	546k IOPS	2.16 GB/s

DAS Mode

	Optane NVMe-oF	
	4kB Rnd	128k Seq
Reads	571k IOPS	2.28 GB/s
Writes	543k IOPS	2.18 GB/s

Disaggregated Mode



Takeaways

- Multiple solutions being delivered to the market this year
- Decision to make: Versatility vs optimized hardware
- Composable Infrastructure is nearing reality
- No need to sacrifice performance… while reaping upsides
- 2019 will be the year of initial NVMe-oF deployments

Q & A

Flash Memory Summit

Thank You!