NVMe over TCP
Storage with SPDK

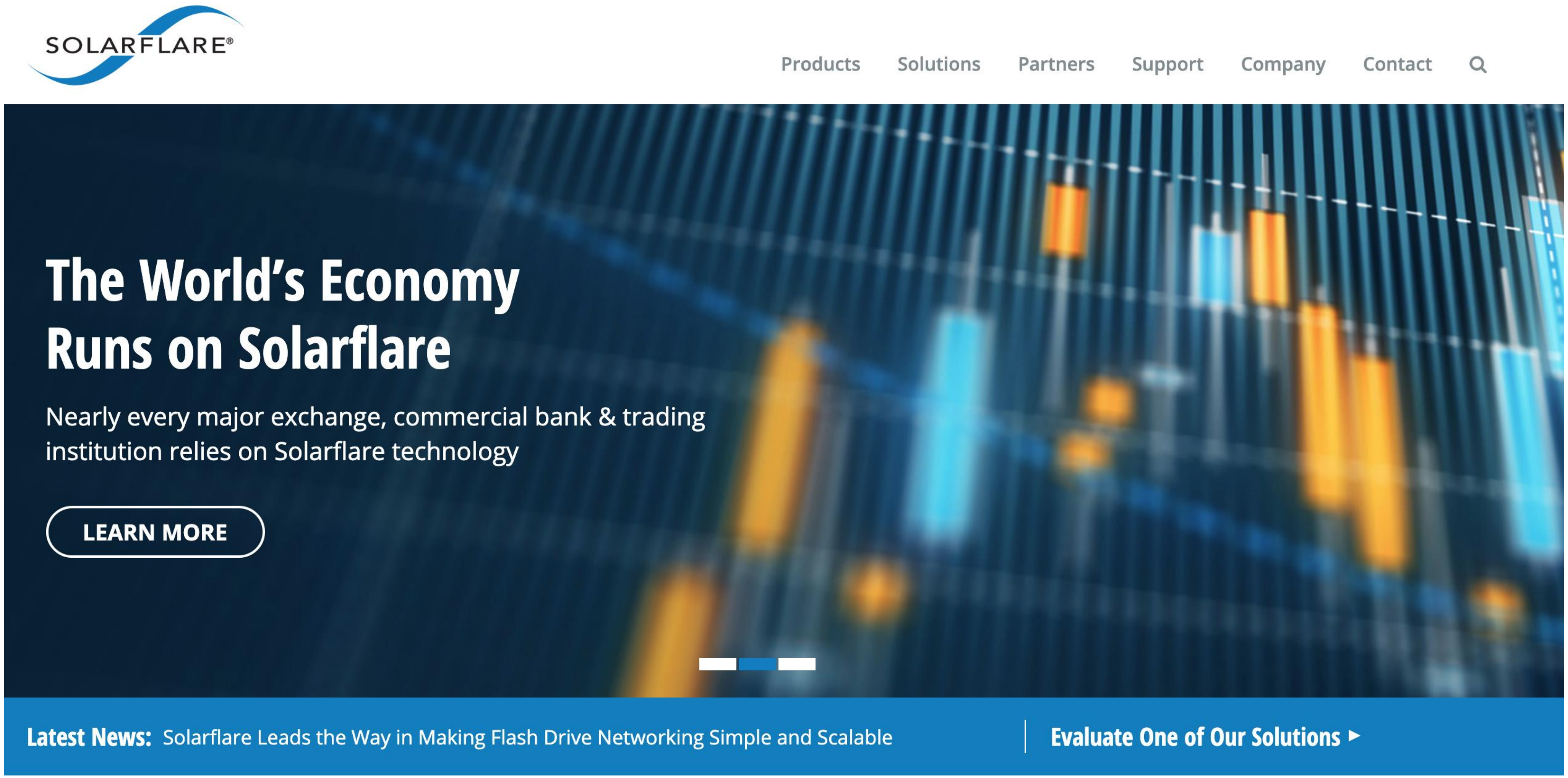
Solarflare has a mature and proven user space TCP/IP stack that has been integrated into every capital market, powering exchanges for over a decade. We recently ported this stack, called Onload®, onto SPDK. Solarflare has spent the past 24 months testing and proving the value of NVMe over TCP. We like to explore ways to work closer with the SPDK community and ecosystem.

Solarflare, Transforming the Way Applications Connect to Networks

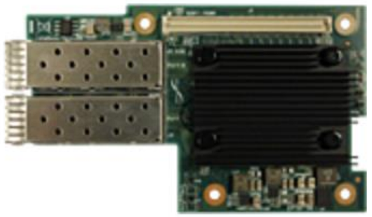
Patrick Dehkordi

About Solarflare

- Innovation since 2001
  - Founders still involved
  - 230+ Employees
  - HQ in Irvine CA, Engineering in Cambridge UK
  - 2000+ Customers
  - 82 countries
  - 90% market share in electronic trading
  - Onload powers nearly all the the major capital market exchanges.
  - Network adapters
  - Complementary intelligent network software
  - Packet accelerators, Packet capture, Packet filters



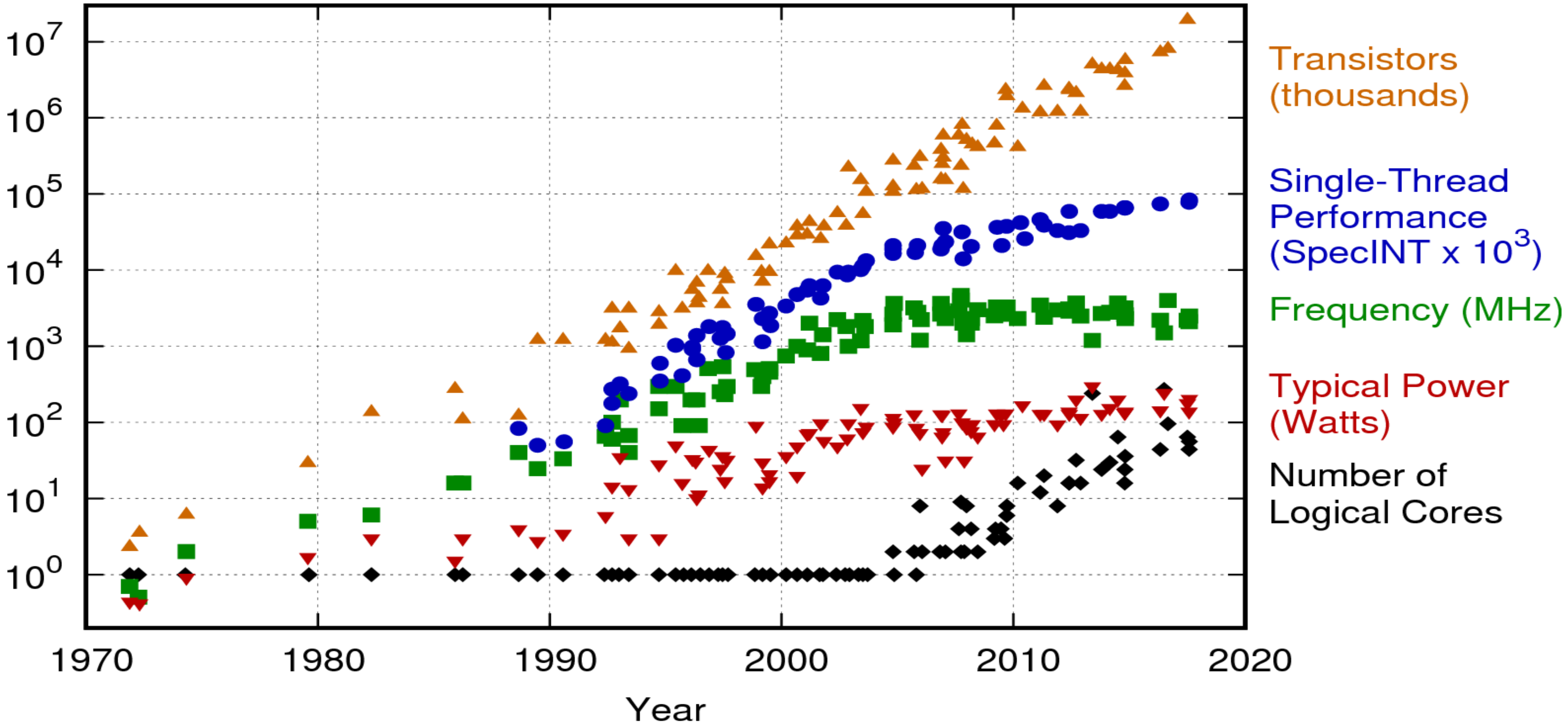
Solutions for Enterprise, Private & Public Cloud Data Centers



XtremeScale X2522 Family		XtremeScale X2552	XtremeScale X2541
Product Brief		Product Brief	Product Brief
X2522-10G / X2522-10G-PLUS	X2522-25G / X2522-25G-PLUS	X2552	X2541
10GbE	**10/25GbE**	**10/25GbE**	**10/25/40/50/100GbE**
Dual Port	Dual Port	Dual Port	Single Port
Low Profile, MD2	Low Profile, MD2	OCP v2.0	Low Profile, MD2
PCIe 3.1 x8	PCIe 3.1 x8	PCIe 3.1 x8	PCIe 3.1 x16
SFP28	SFP28	SFP28	QSFP28

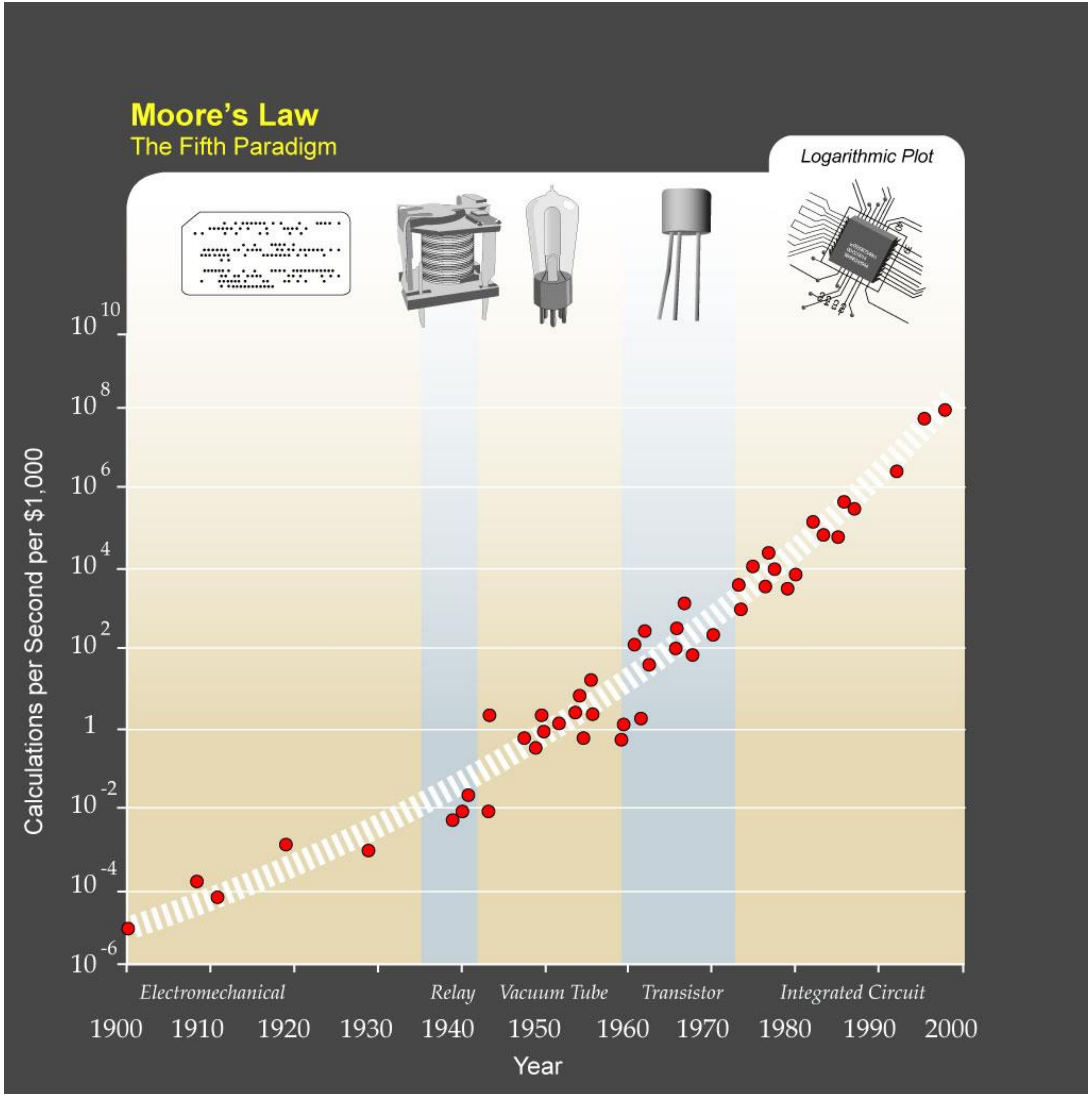
PLUS SKU includes Onload, TCP Direct, and PTP

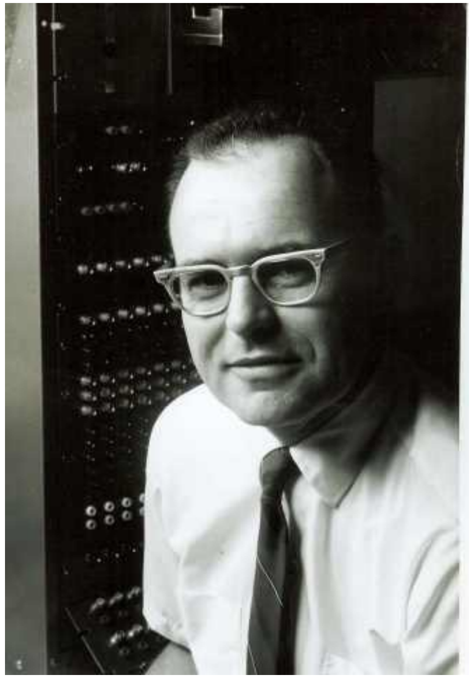
Hardware Advances - 42 Years of Microprocessor Trend Data



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

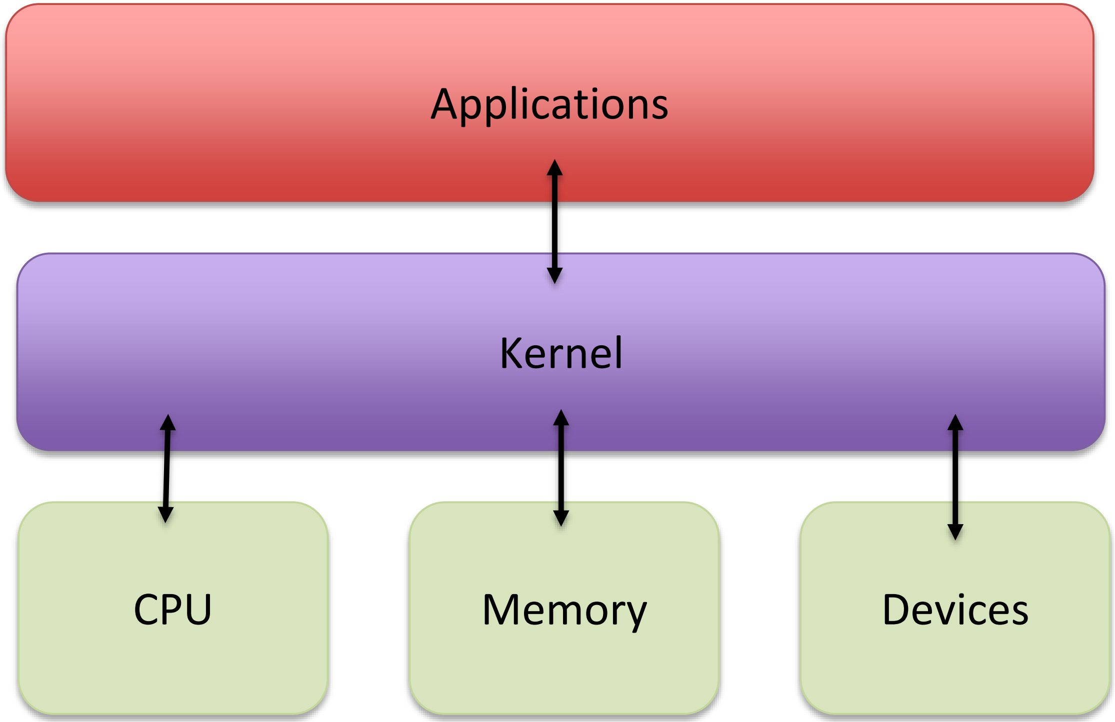
According to Kurzweil, since the beginning of [evolution](), more complex life forms have been evolving exponentially faster, with shorter and shorter intervals between the emergence of radically new life forms, such as human beings, who have the capacity to engineer





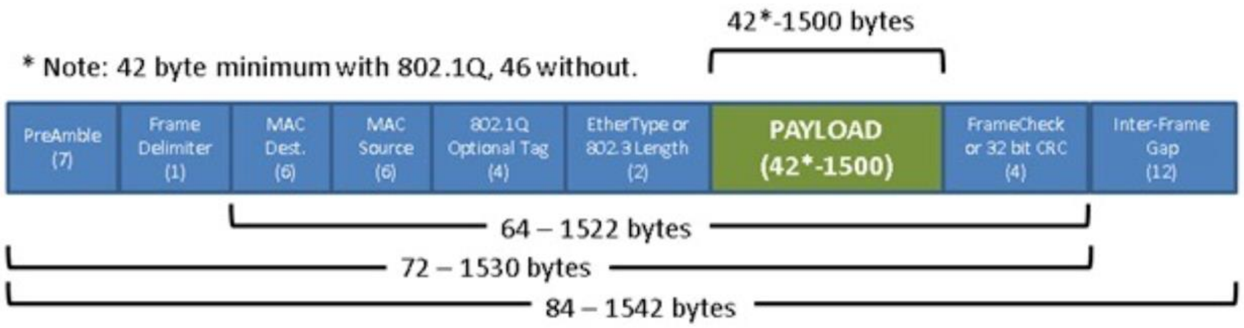
System Architecture and performance

- Application
- Kernel
- CPU
- Memory
- Storage IO
- Network IO



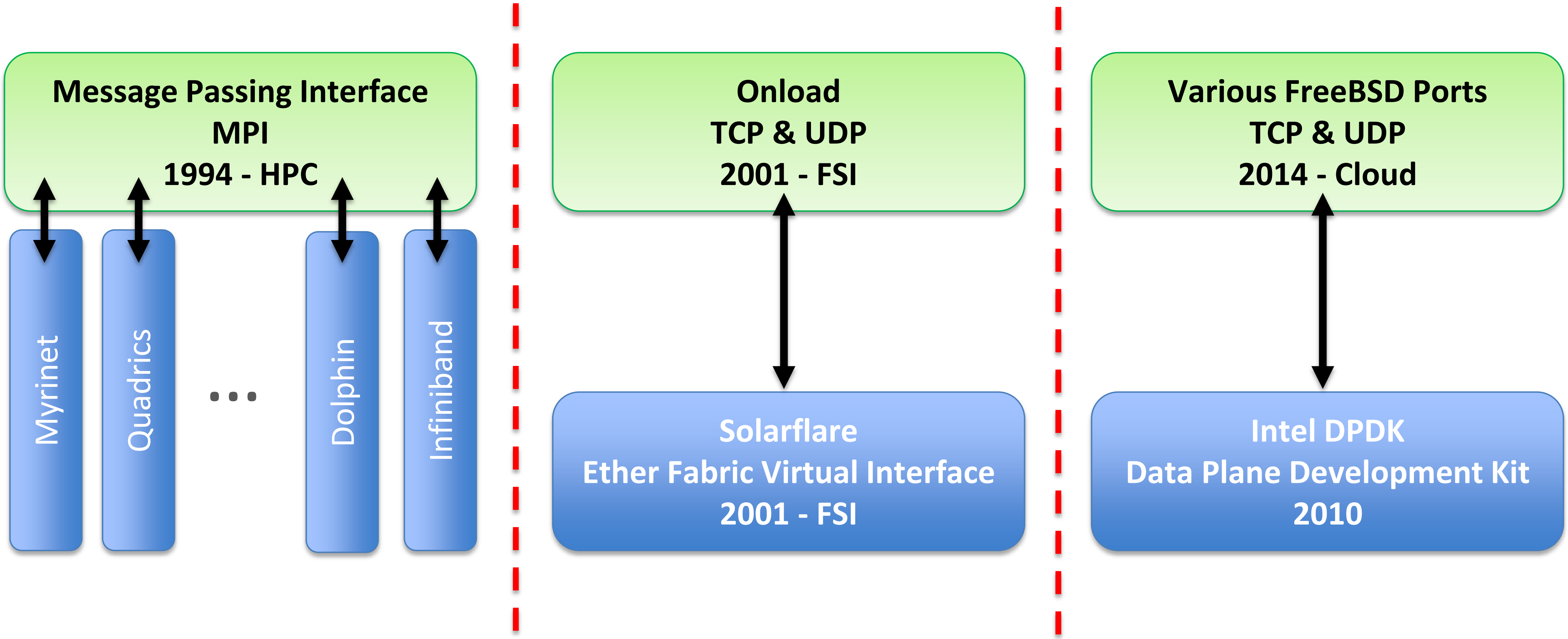
Metrics for IO performance: Latency vs Packet Rate

Bandwidth (Gbs) = Packet Rate (Mpps) * Packet Size



Speed	bits/second	bytes/second	maximum PPS
10Mbps	10,000,000	1,250,000	14,881
100Mbps	100,000,000	12,500,000	148,810
1Gbps	1,000,000,000	125,000,000	1,488,095
10Gbps	10,000,000,000	1,250,000,000	14,880,952
100Gbps	100,000,000,000	12,500,000,000	148,809,524

User Layer Networking has Evolved to Address the Problem

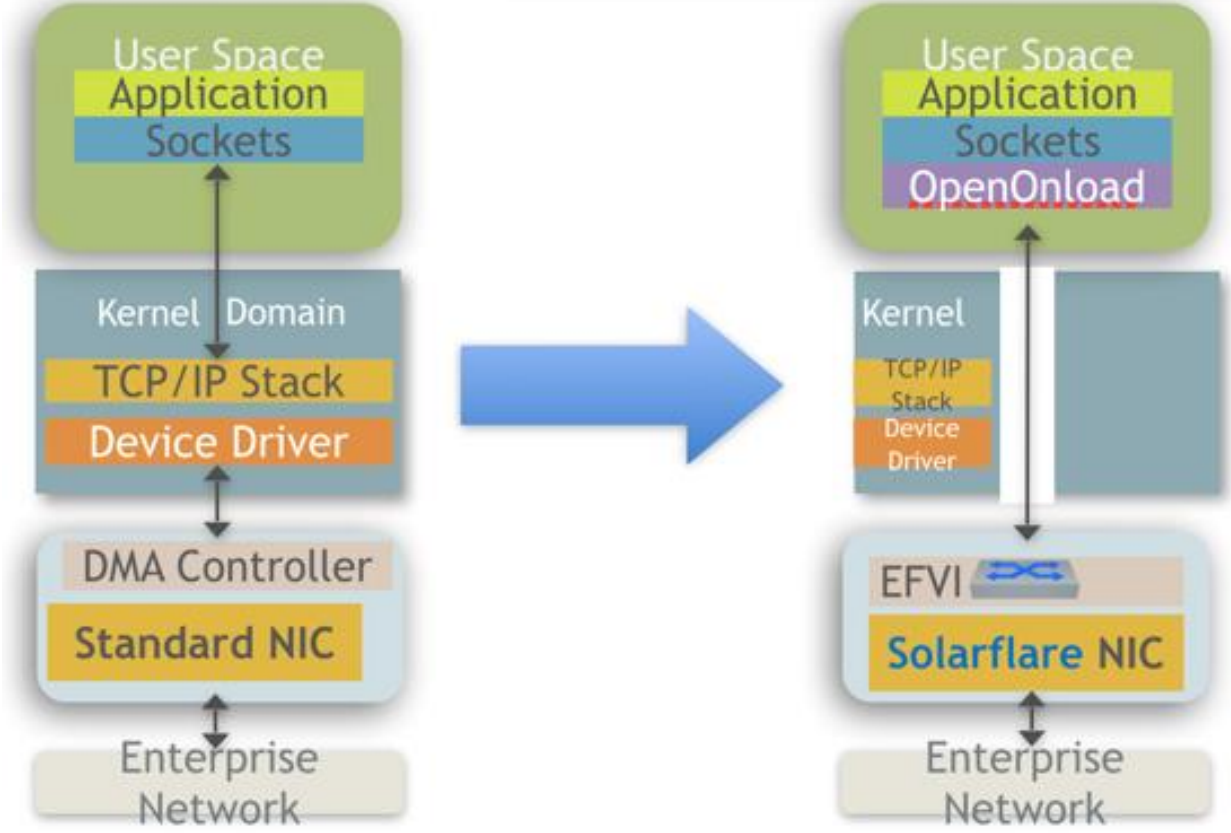


User Space Networking with Onload

- Direct connection between Applications and Networks

- Reduced latency (ns)
- Improved packet rate (pps)
- Efficient CPU utilization (W)
- Near zero jitter (sigma)
- TCP/IP (R&D)

Onload



How Onload Works:

- Applications makes socket calls unaware of existence of Onload.
- Onload is activated at execution by preloading **libonload.so**
- Onload Library is a ***dynamically linked shared object***.
- **LD_PRELOAD** environment variable is used at runtime.
- Onload sits between the application executable and **libc.so**
- **libc.so** is the library that implements normal sockets call
- **libonload.so** intercepts:
  - *poll(), select(), epoll_wait()*
  - *recv(), read(), recvfrom(),*
  - *recvmsg() send(), sendto()*
  - and others…
- If they operate on these sockets, **libonload.so** will take over.
- If not system calls are passed to **libc.so**
- A list of accelerated file descriptors/sockets is kept to distinguish between user space and kernel space.
- For Rx , filters are inserted for each accelerated socket.
- Filters enable the adapter to DMA a particular packet onto the Onload vNIC.
- Traffic that is not filtered is passed to the Kernel vNIC.

Applications Seeing 50-400% Performance Gains with Onload

1) Webservers
2) Proxies / Load Balancers
3) Databases
4) Message Brokers
5) DNS / Routers
6) In-memory Data Stores
7) In-memory Compute Grids

Evolution of IO bus

1) ISA (Industry Standard Architecture) bus: ISA bus was created by IBM in 1981. The ISA bus can transfer 8 or 16 bits at one time. ISA's 8 bit bus ran at 4.77 MHz *(the clock speed of the IBM PC's 8088 CPU)* with a data transfer of just over 2 MByte/s. The 16 bit (2 byte) IBM AT's 80286 CPU ran originally at 8 MHz and about 8 MByte/s. The ISA is still in use with parallel printers.

2) PCI (Peripheral Component Interconnect) bus: PCI bus was created by Intel in 1993. PCI bus can transfer 32 or 64 bits at one time. PCI bus ran originally at 33 Mhz, with a data transfer of 250 Mbyte/s. PCI Express is used with modern graphics processor cards at 1 GByte/s (or more), also network cards.

With regards to the actual display screens, the new HDMI version 2.0 video display controller supports 18 GBit/s, and the newer HDMI version 2.1 controller supports 48 GBit/s. With the complementary display standard known as DisplayPort, their version 1.2 supports 17 GBit/s and versions 1.3 and 1.4 support 32 GBit/s.

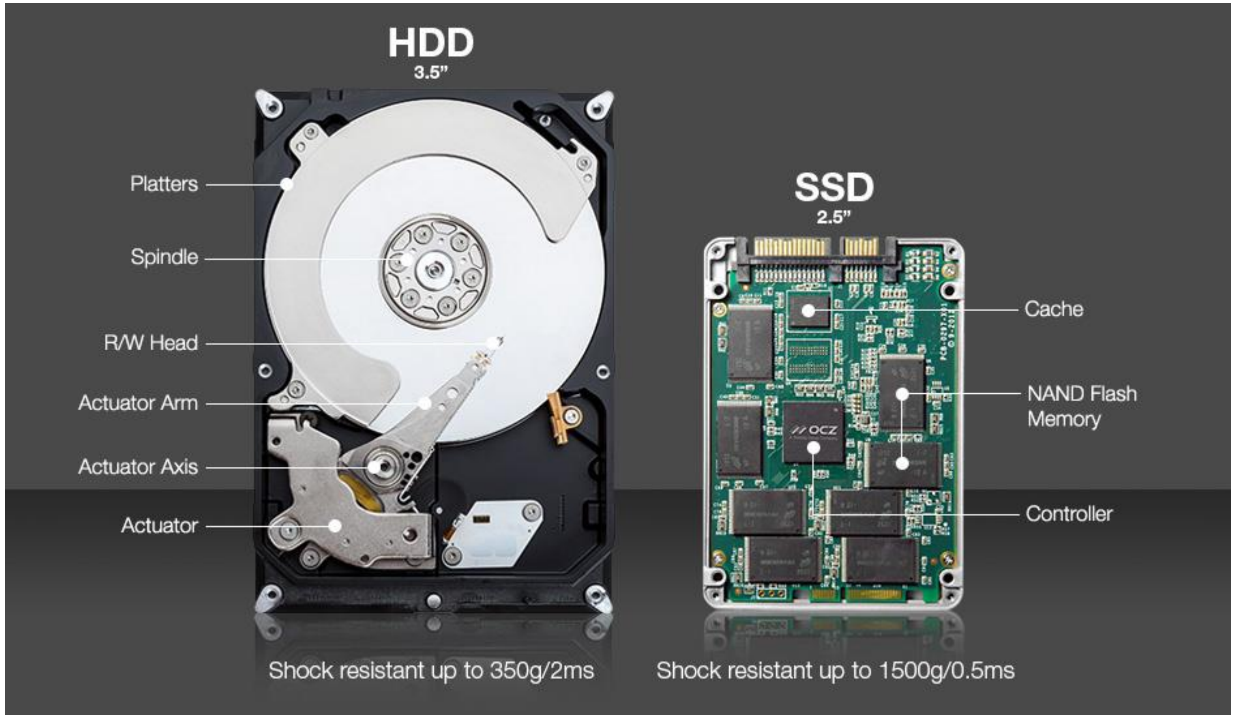
3) IDE (Integrated Drive Electronics) bus: IDE bus is used for connecting disks and CDROMs to the computer. Retroactively termed Parallel ATA (AT attachment) with the first such 16-bit drives appearing in Compaq PCs in 1986. A short-lived, 8-bit implementation of ATA was created for the IBM XT and similar machines. The latest versions of Parallel ATA support up to 133 MByte/s.
Since 2003 PATA has been replaced by SATA (Serial ATA), which uses the same basic command set but is able to operate at a much higher speed needing fewer support and control signals. Their revision 3.2 release in 2013 supported 2 GByte/s.

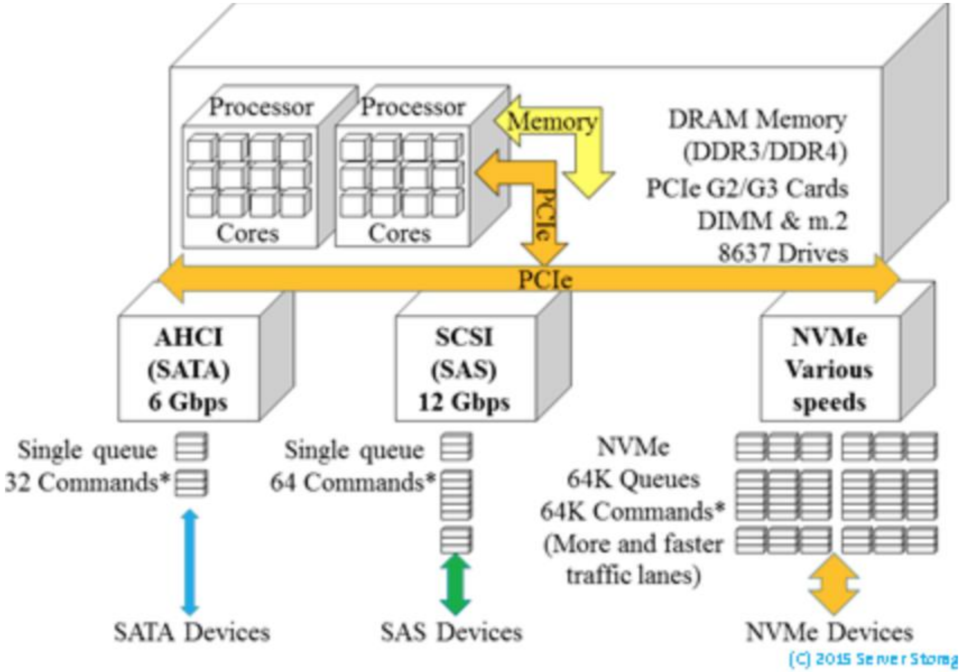
4) SCSI (Small Computer System Interface) bus: It is a high performance 16-bit bus which was used for fast disks, scanners, and for devices which require high bandwidth. It has a data rate of 640 MByte/s.

5) USB (Universal Serial Bus), a single bit bus: It is used for connecting keyboard mouse and printer and other USB devices such as wireless network adapters to the computer. A USB bus has a connector with four wires. Two wires are used for supplying electrical power to the USB devices. USB 1.0 had a data rate of 1½ MByte/s and USB 2.0 has a data rate of 60 MByte/s. There is now a USB 3.0, that can travel at 640 MByte/s, though interference issues have been reported with wireless devices.

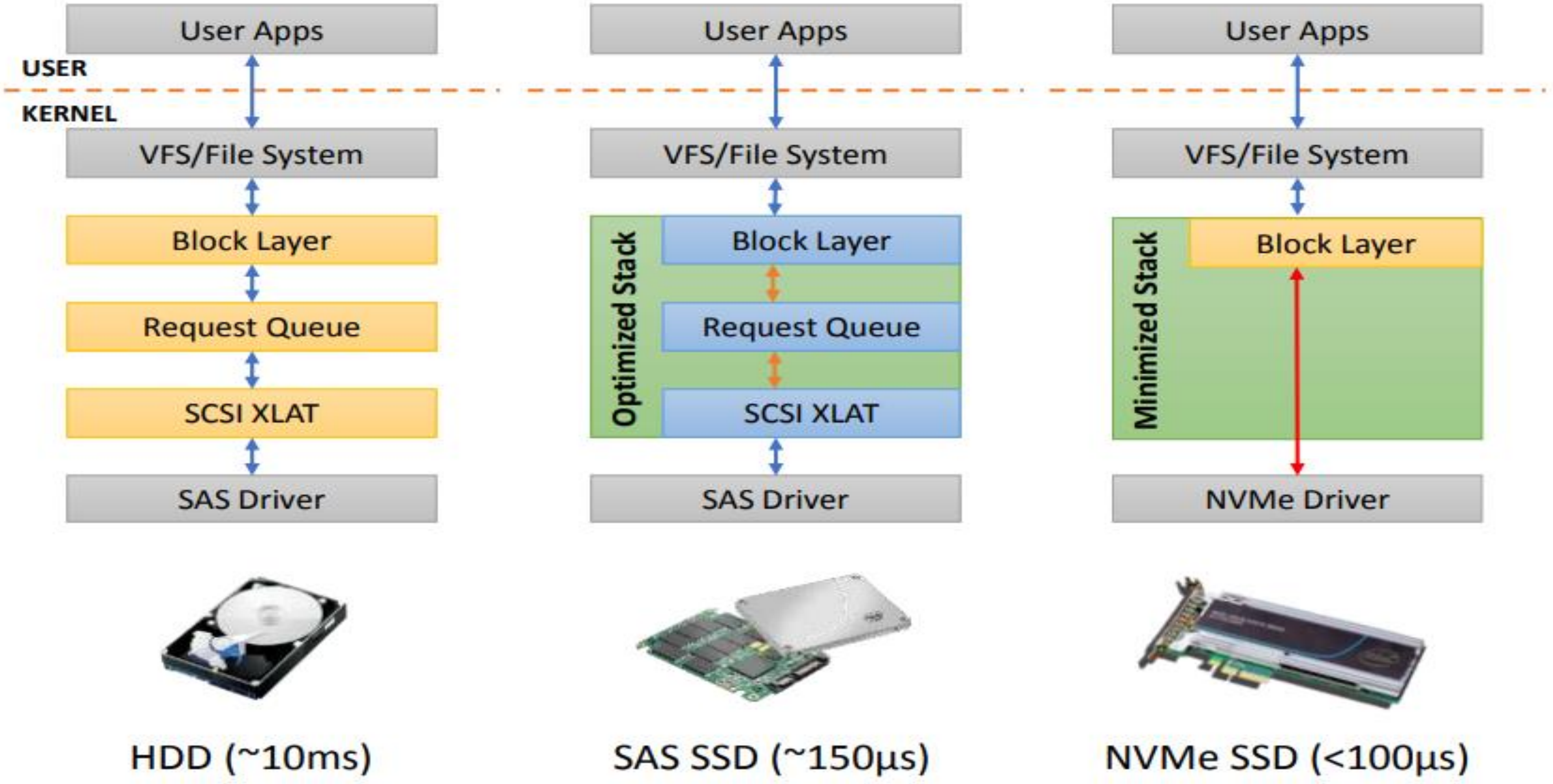
6) IEEE 1394 or FireWire: IEEE 1394 is used for high speed data transfer. It was built by Apple, though Apple have moved away from it. It can transfer data at a rate of up to 400 MByte/s. It is a single bit serial bus which is used for connecting cameras, and other multimedia devices.

Need for a solid state storage protocol

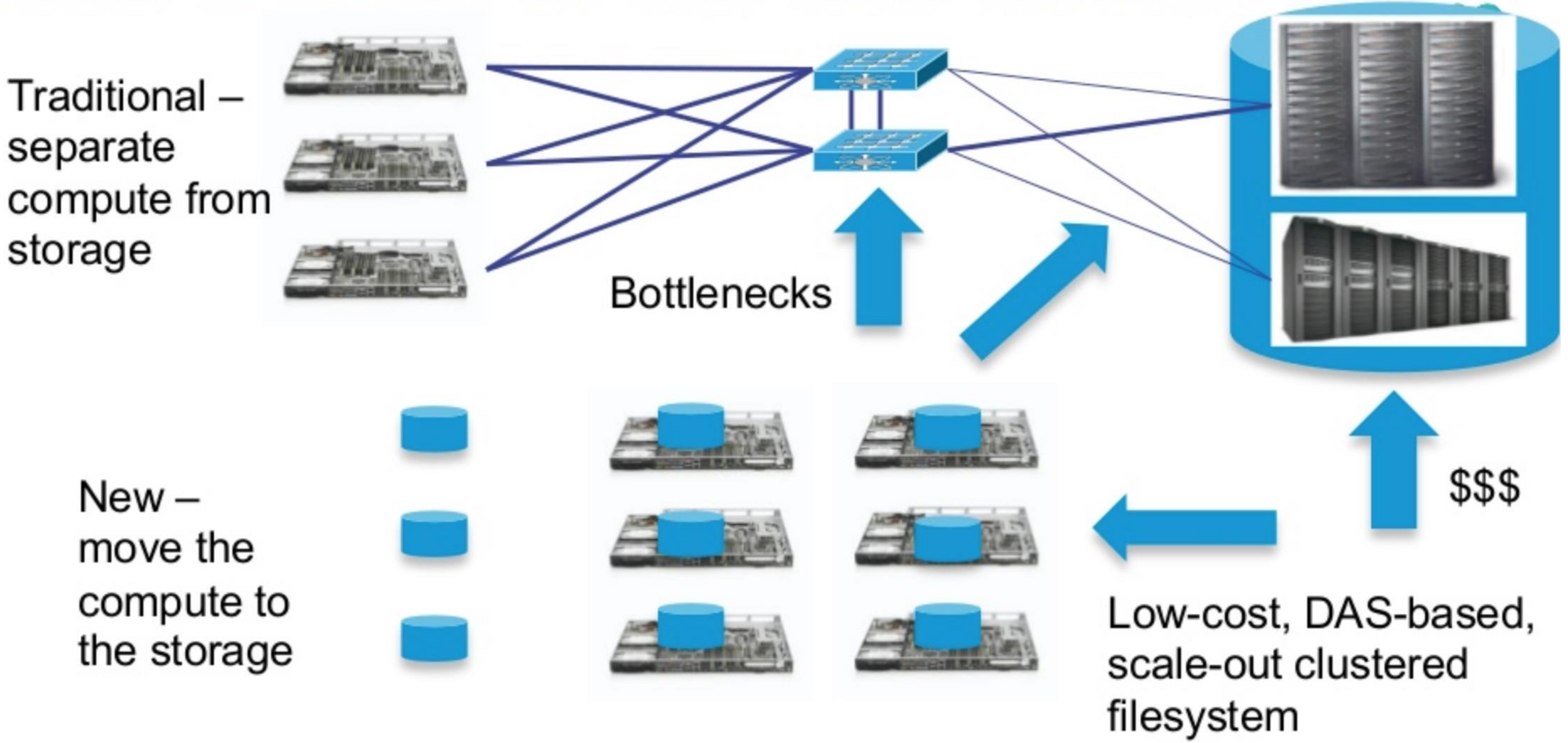




Evolution of the Storage Stack



Classic NAS/SAN vs. New Scale-out DAS



Cost Model

$$C_{direct} = \max\left(\frac{\mathrm{GB}_t}{\mathrm{GB}_s}, \frac{\mathrm{IOPS}_t}{\mathrm{IOPS}_s}, \frac{\mathrm{QPS}_t}{\mathrm{QPS}_s}\right) \cdot (f + c)$$

$$C_{disagg} = \max\left(\frac{\mathrm{GB}_t}{\mathrm{GB}_s}, \frac{\mathrm{IOPS}_t}{\mathrm{IOPS}_s}\right) \cdot (f + \delta) + \left(\frac{\mathrm{QPS}_t}{\mathrm{QPS}_s}\right) c$$

where:

f: cost of Flash on a server

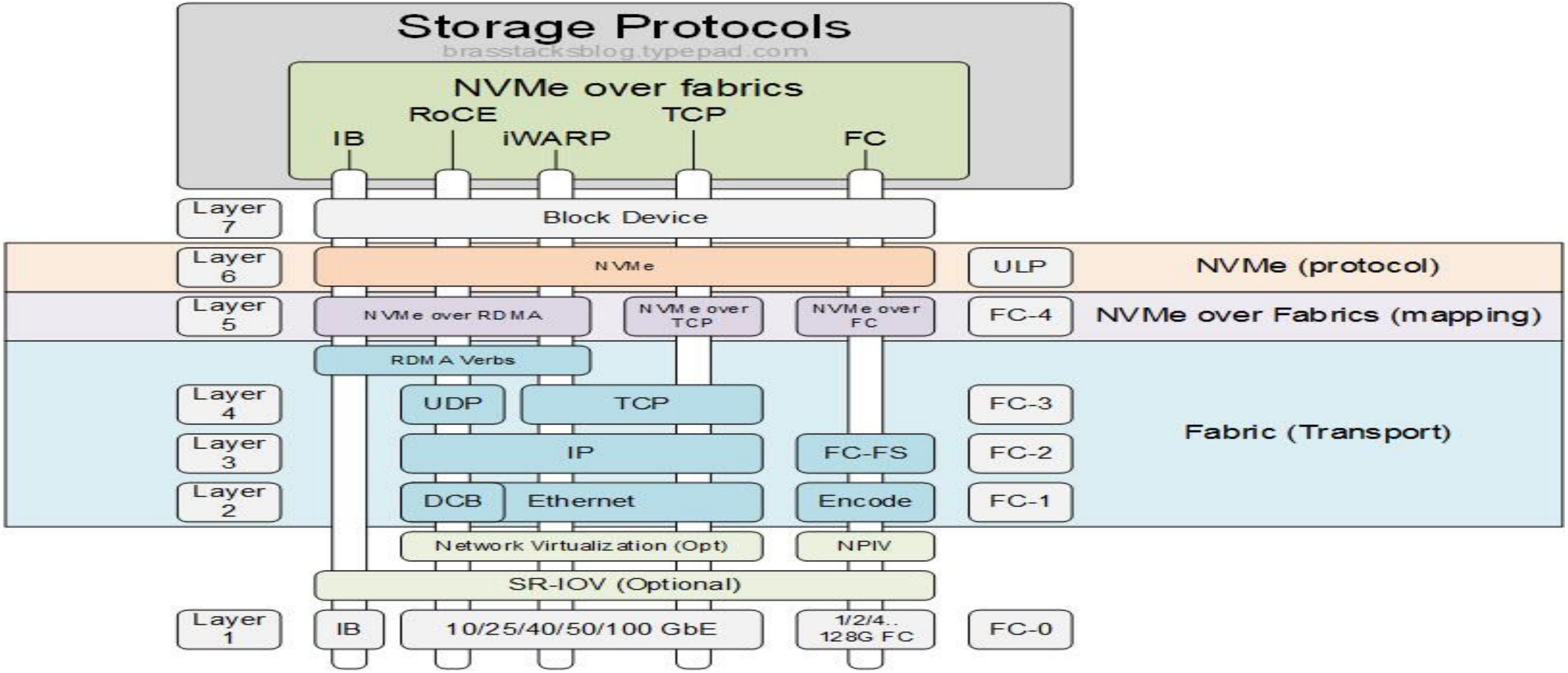
c: cost of CPU, RAM and NIC on datastore server

δ: cost of CPU, RAM and NIC on Flash tier server, i.e. resource "tax" for disaggregation

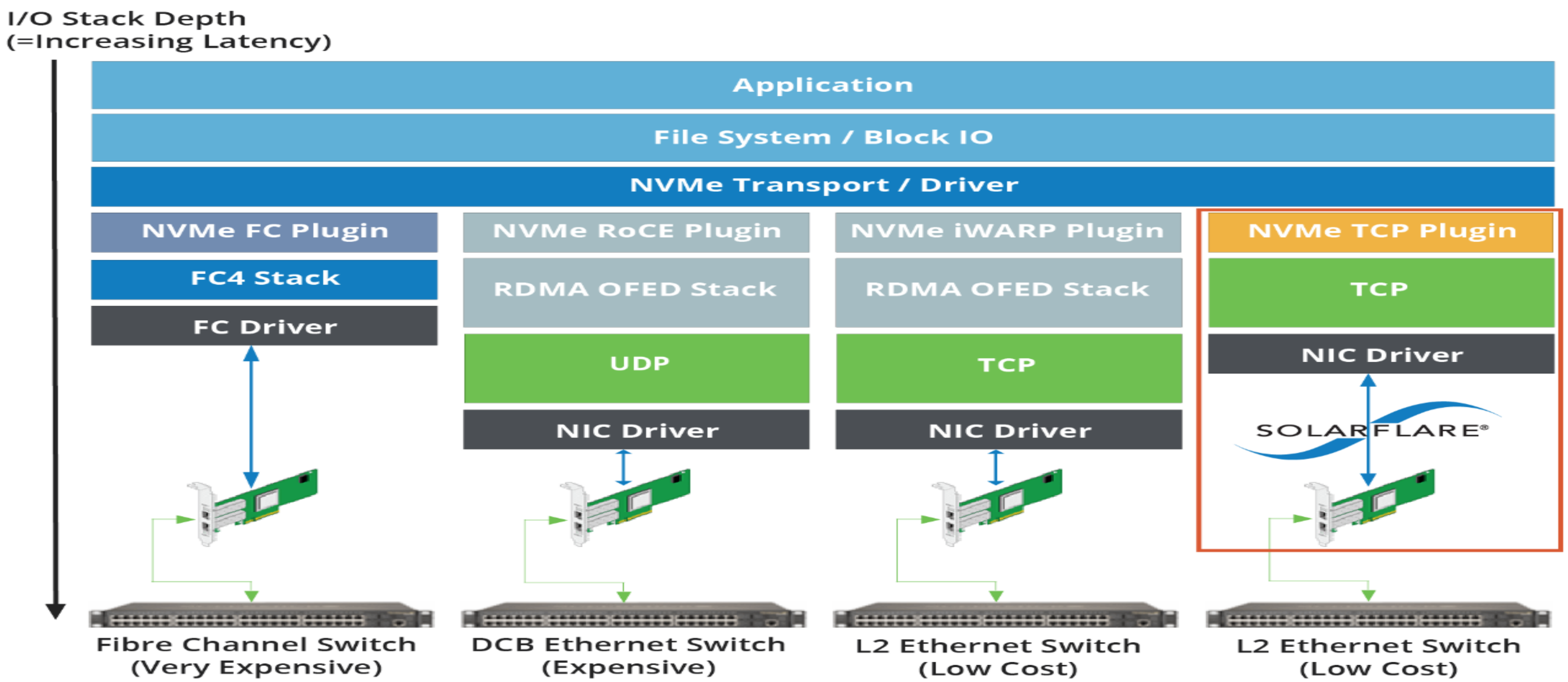
x_s: x provided by a single server, $x = \{\mathrm{GB}, \mathrm{IOPS}, \mathrm{QPS}\}$

x_t: x required in total for the application

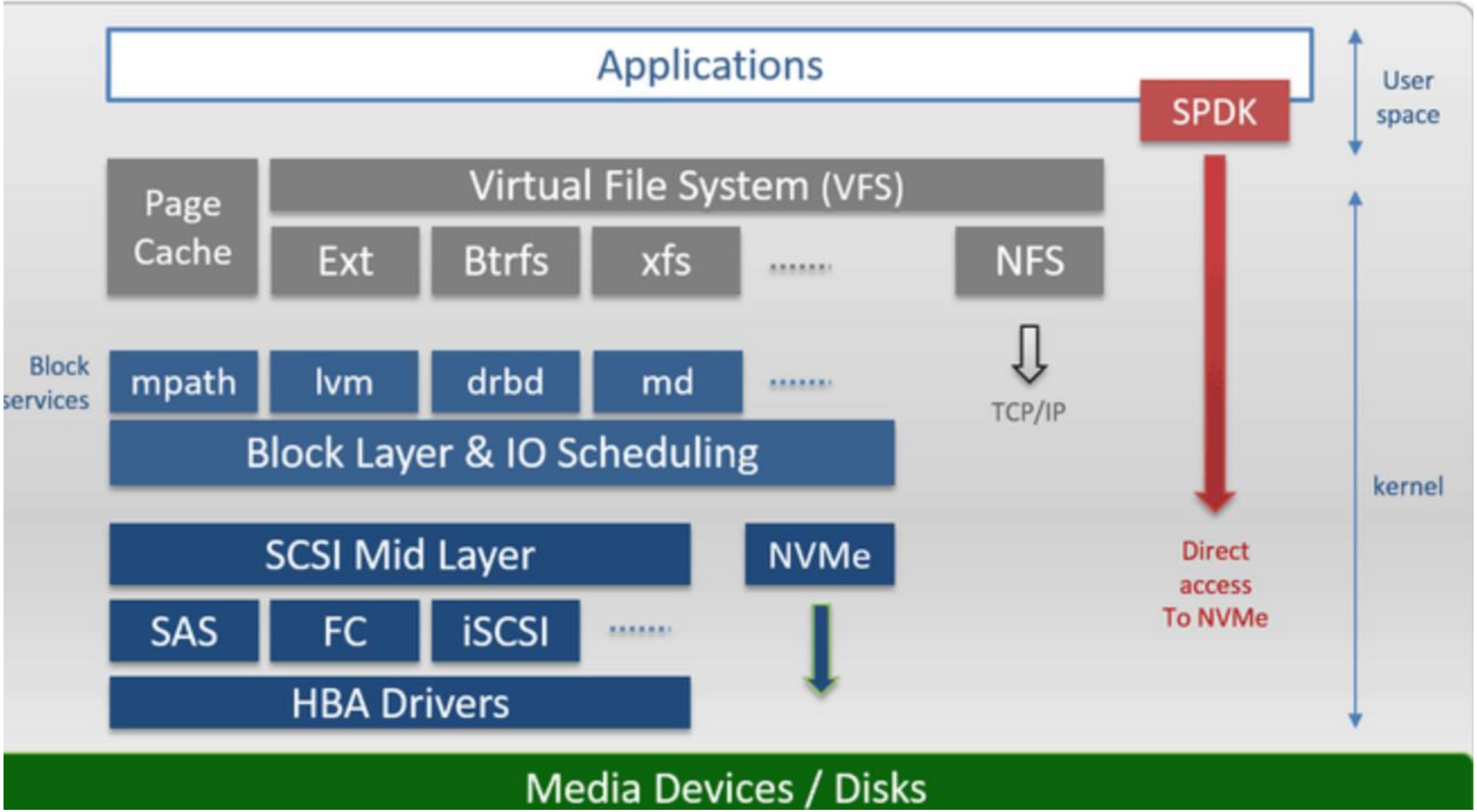
Network Storage NVMe



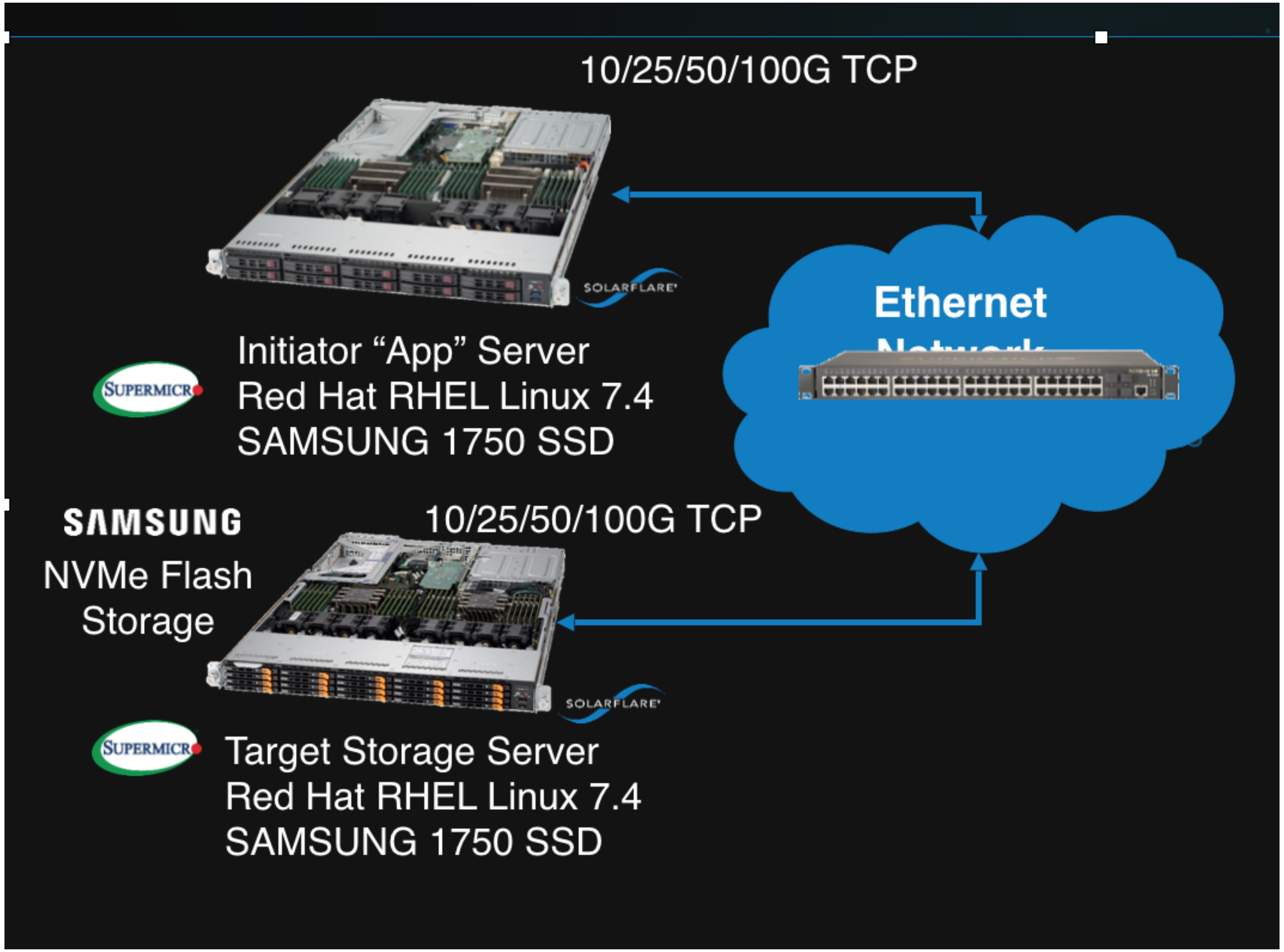
NVMe/TCP



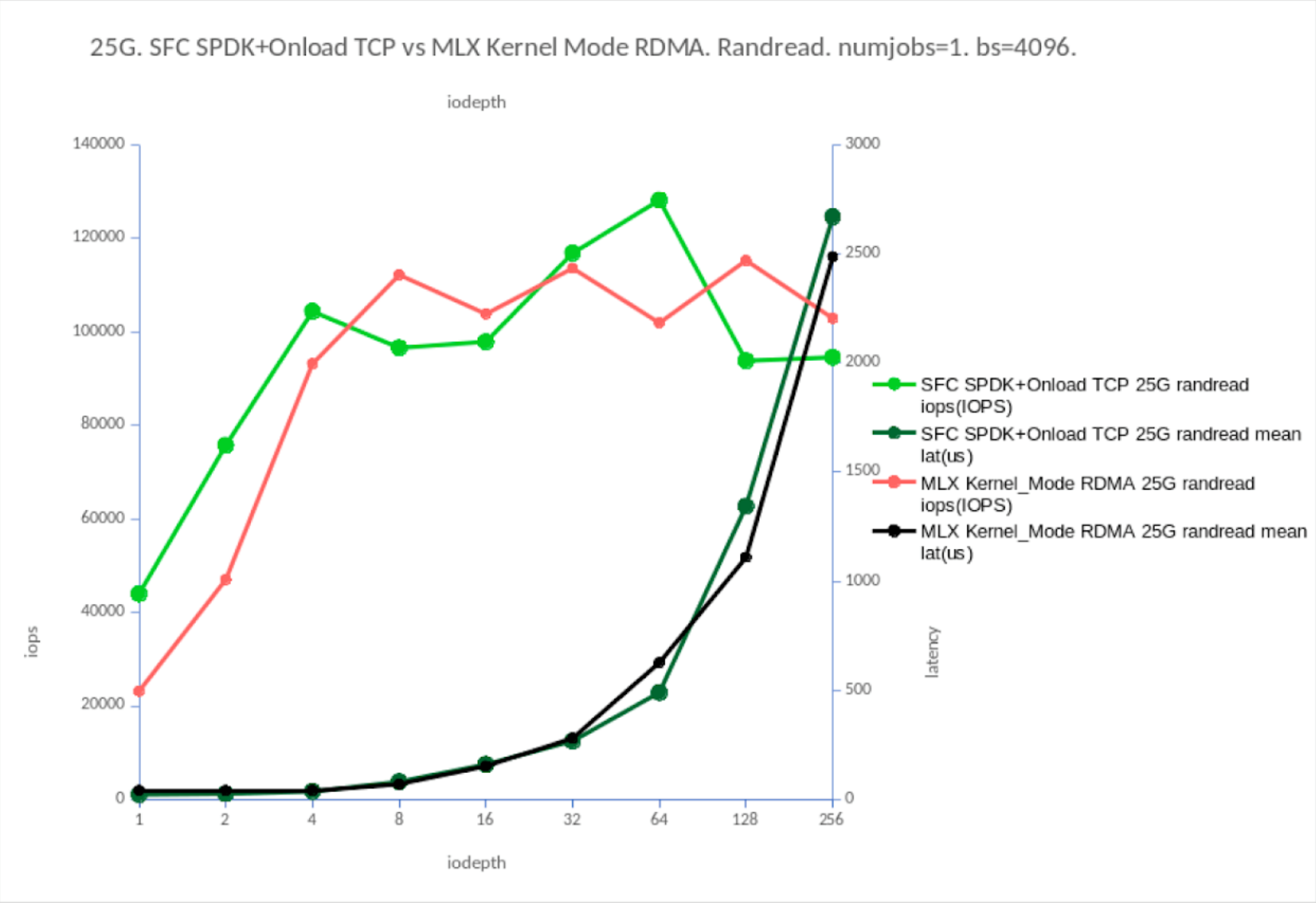
I/O Stack Depth
(=Increasing Latency)
Application
File System / Block IO
NVMe Transport / Driver
NVMe FC Plugin
FC4 Stack
FC Driver
NVMe RoCE Plugin
RDMA OFED Stack
UDP
NIC Driver
NVMe iWARP Plugin
RDMA OFED Stack
TCP
NIC Driver
NVMe TCP Plugin
TCP
NIC Driver
SOLARFLARE
Fibre Channel Switch
(Very Expensive)
DCB Ethernet Switch
(Expensive)
L2 Ethernet Switch
(Low Cost)
L2 Ethernet Switch
(Low Cost)


Applications
SPDK
User space
Page Cache
Virtual File System (VFS)
Ext
Btrfs
xfs
....
NFS
Block services
mpath
lvm
drbd
md
....
TCP/IP
Block Layer & IO Scheduling
kernel
SCSI Mid Layer
NVMe
Direct access To NVMe
SAS
FC
iSCSI
....
HBA Drivers
Media Devices / Disks

Demo Setup


10/25/50/100G TCP
Ethernet
Initiator “App” Server
Red Hat RHEL Linux 7.4
SAMSUNG 1750 SSD
SAMSUNG
NVMe Flash
Storage
10/25/50/100G TCP
Target Storage Server
Red Hat RHEL Linux 7.4
SAMSUNG 1750 SSD

1. SFC SPDK+Onload Target TCP (18.2 Release) vs MLX RDMA @25G.



Basic Comparison/Observations.

1. SFC SPDK+Onload Target TCP vs MLX RDMA @25G.

Latency.

+ SFC SPDK+Onload min latency @ FIO numjobs=1 slightly higher than that of MLX RDMA.
  - randread: 22ns vs 19ns
  - randwrite: 24ns vs 17ns

Throughput.

+ SFC SPDK+Onload max throughput @ FIO numjobs=40 marginally better than that of MLX RDMA.
  - randread: 719kIOPS (2811MB/s) vs 715kIOPS (2794MB/s)
  - randwrite: 705kIOPS (2757MB/s) vs 703kIOPS (2746MB/s)

Misc.

At 25G, the I/O throughput seems network bandwidth limited

- with both SFC SPDK+Onload & MLX RDMA hitting approx. line rate. @iodepth > 2.
- Theoretically, line rate limit @25G is approx. 3000MB/s

Intel has Highlighted the Benefits of Using SPDK

SOLARFLARE®

SPDK
more performance from CPUs, non-volatile media, and networking

Up to **10X MORE** IOPS/core for NVMe-oF* vs. Linux kernel

Up to **8X MORE** IOPS/core for NVMe vs. Linux kernel

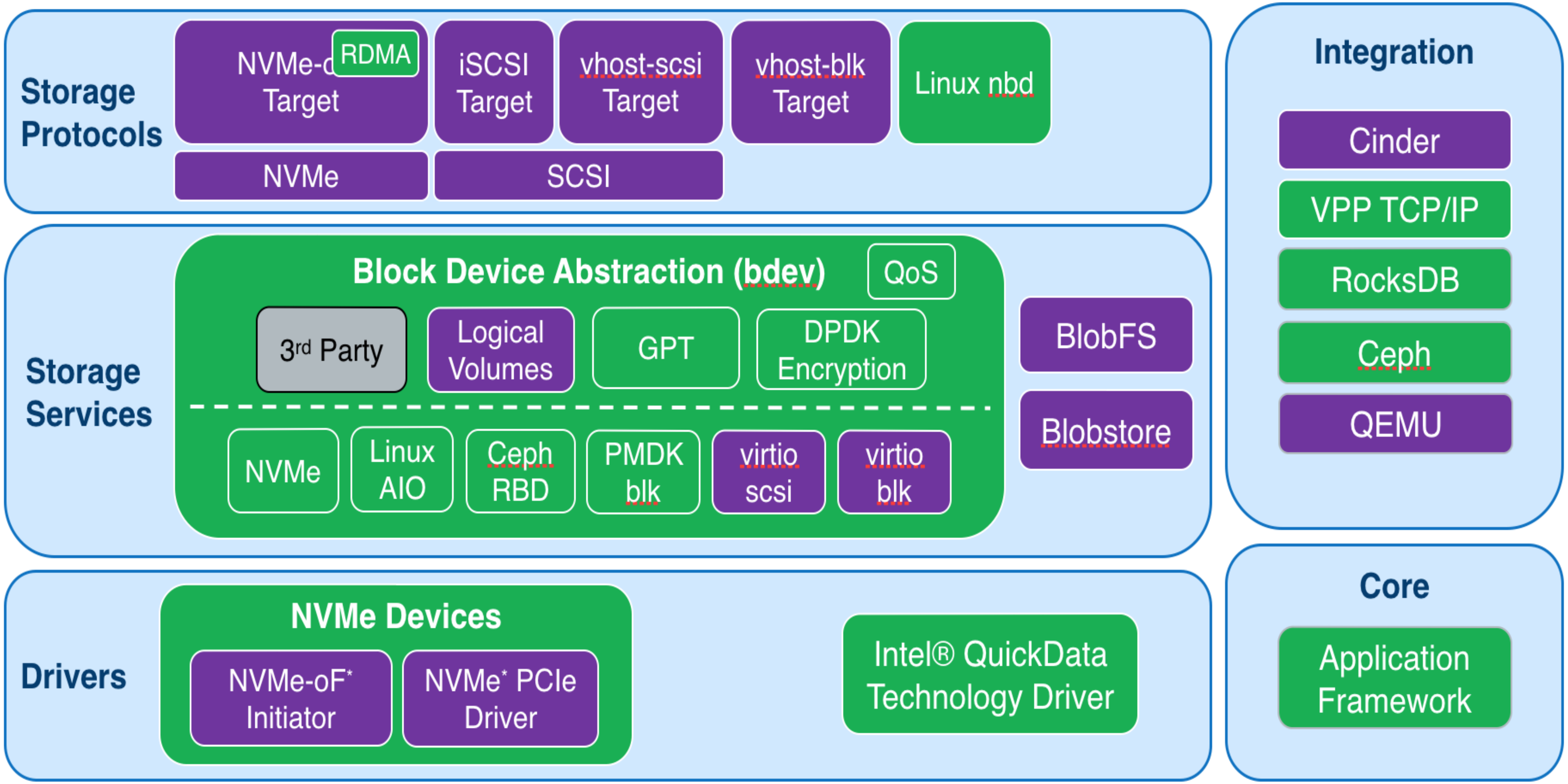
Up to **50% BETTER** Tail Latency for RocksDB workloads

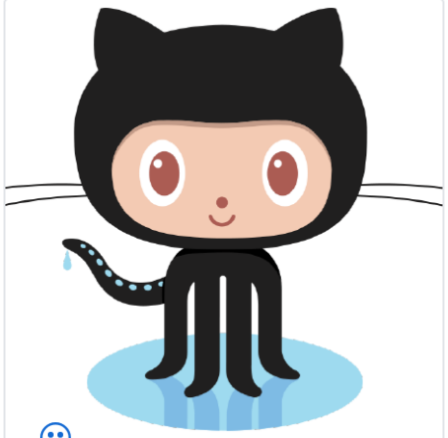
FASTER TTM/ **LESS** RESOURCES than developing components from scratch

Provides Future Proofing as NVM technologies increase in performance

http://www.intel.com/performance

SPDK Architecture





Set your status

Patrick Dehkordi

PatrickDehkordi

The Solarflare repos are examples of integrating SolarFlare products with 3rd party software. They require obtaining the Solarflare software to be functional.

Solarflare
USA
patrick.dehkordi@gmail.com
https://www.linkedin.com/in/ql...

Overview | Repositories 21 | Projects 0 | Stars 23 | Followers 4 | Following 21

Pinned

Customize your pins

nvme-of-tcp
Forked from solarflarecommunications/nvme-of-tcp
Linux kernel source tree
C

patrickdehkordi.github.io
githost
HTML

TiffCoin
Forked from cryptonotefoundation/cryptonote
TiffCoin, a cyrpto currency based on the CryptoNote protocol
C++

OnloadDocker
Open Onload in Docker
★ 3

libmemcached
Forked from membase/libmemcached
Where I do my development for libmemcached
C 1

memcached
Forked from memcached/memcached
memcached development tree
C