Storage Interfaces

Introduction

Tom Gardner
IEEE Silicon Valley Technology History Committee

- Part of IEEE Santa Clara Valley Section
- Webinars on history of Silicon Valley history

www.SiliconValleyHistory.com
## Storage Interfaces

### Agenda

<table>
<thead>
<tr>
<th>Speaker</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom Gardner</td>
<td>Introduction &amp; History</td>
</tr>
<tr>
<td>Grant Saviers</td>
<td>Evolution of Storage Interfaces</td>
</tr>
<tr>
<td>Jai Menon</td>
<td>Evolution of Block Storage System Interfaces</td>
</tr>
<tr>
<td>Amber Huffman</td>
<td>Fast &amp; Efficient Interfaces for the SSD Era</td>
</tr>
<tr>
<td>Questions &amp; Answers</td>
<td></td>
</tr>
</tbody>
</table>
Layers of memory and storage

In the beginning

Never enough memory — UNOBTANIUM
Early memory hierarchy – IBM 650

Tape Storage

Immediate Access Storage

Disk Storage

Input Card

Drum

Output Card

Inquiry & Reply

Storage Interfaces

Silicon Valley Technology History Committee
JBODs with “dumb” drive interfaces
(1960s – 1980s)
Early defacto “dumb” drive interfaces
“Dumb” drive interface characteristics

Typically three cables:
- Power
- **Control**
  - Selection
  - Movement
  - Status
  - Start/Stop
  - Power control
- **Data**
  - Unidirectional
  - Synchronization
  - Safety

Maximum individual control cable (A) lengths = 100 ft

Maximum individual data cable (B) length = 50 ft
## “Dumb” drive interfaces

### Dumb because drive specific functions in controller

- Serializing and deserializing
- ECC/EDC generation and correction/detection
- Blocking and unblocking
- Write encoding and data separation

### Barrier to technical innovation

<table>
<thead>
<tr>
<th>Date</th>
<th>Command Interface</th>
<th>Step/direction Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>1961</td>
<td>7361 SCU/1301</td>
<td></td>
</tr>
<tr>
<td>1965</td>
<td>2841 SCU 2302,2303,2311,2321</td>
<td></td>
</tr>
<tr>
<td>1969</td>
<td>DEC Controllers+  MRX 630-1 DEC RP01-3</td>
<td></td>
</tr>
<tr>
<td>1973</td>
<td></td>
<td>SA900</td>
</tr>
<tr>
<td>1975</td>
<td>CDC SMD</td>
<td></td>
</tr>
<tr>
<td>1978</td>
<td></td>
<td>SA4000</td>
</tr>
<tr>
<td>1979</td>
<td></td>
<td>ST506/ST412</td>
</tr>
<tr>
<td>1984</td>
<td>ESDI</td>
<td></td>
</tr>
</tbody>
</table>
Parallel intelligent drive interfaces
(1972 – 2000s)
Move the intelligence into the drives
three examples

IBM Proprietary DCI
DEC Proprietary Massbus
Sun “Standard” IPI

Sort of standard
SASI/SCSI
Bridge controller

ANSI Standard
SCSI
ATA
1979 | SA SASI Paper
SA and NCR at ANSI

Beginning 1981? | Bridge controllers
### Parallel SCSI

<table>
<thead>
<tr>
<th>Year</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1979</td>
<td>SA SASI Paper SA and NCR at ANSI</td>
</tr>
<tr>
<td>Beginning 1981?</td>
<td>Bridge controllers</td>
</tr>
<tr>
<td>Beginning 1984</td>
<td>Embedded SCSI</td>
</tr>
</tbody>
</table>

A 10-Megabyte 5¼-Inch Mass Storage Device With Embedded Controller.
<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1979</td>
<td>SA SASI Paper</td>
</tr>
<tr>
<td></td>
<td>SA and NCR at ANSI</td>
</tr>
<tr>
<td></td>
<td><strong>Beginning 1981?</strong></td>
</tr>
<tr>
<td></td>
<td>Bridge controllers</td>
</tr>
<tr>
<td></td>
<td><strong>Beginning 1984</strong></td>
</tr>
<tr>
<td></td>
<td>Embedded SCSI</td>
</tr>
<tr>
<td>1986</td>
<td>Common Command Set</td>
</tr>
<tr>
<td></td>
<td>Rev 4b</td>
</tr>
<tr>
<td>(1994!)</td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td>CAM Committee</td>
</tr>
<tr>
<td></td>
<td>ASPI (Adv. SCSI Programming Intf.)</td>
</tr>
<tr>
<td>(1995!)</td>
<td></td>
</tr>
</tbody>
</table>

Your SCSI nightmare is over
Parallel SCSI
SAM - SCSI Architectural Model

X3T10/99D revision 18
November 27, 1995

Figure 2: Functional Scope of SCSI-3 Standards
Parallel SCSI
SCSI-1 (1986) into this century

Logical Block Addressing

- \(2^{24}\) Blocks of up to \(2^{24}\) bytes \(\rightarrow\) \(2^{64}\) Blocks of up to \(2^{64}\) bytes
- 8 bit bus, 8 devices \(\rightarrow\) 16 bit bus, 16 devices
- 8 LUNs/device \(\rightarrow\) 8 byte LUN structure
- 5MB/sec \(\rightarrow\) 640 MB/sec?
- 22 HDD commands (\(\sim62\) pages) \(\rightarrow\) 87 HDD commands (440 pages)
- Single ended cable: 6m \(\rightarrow\) 3 \(\rightarrow\) 1.5 \(\rightarrow\) 0
- HV differential cable: 25m \(\rightarrow\) 0
- LV differential cable: 12m \(\rightarrow\) 10
- 2 connectors \(\rightarrow\) 8 connectors
Parallel ATA

WD Controller w/ MiniScribe drive
Compaq Portable II
1986

Conner CP342
1987
Parallel ATA

“The ATA bus (a.k.a. IDE bus) is a disk drive interface originally designed for the ISA Bus of the IBM PC/AT”

1987  Conner publishes “Task File Interface”
1989  First presentation at CAM
Parallel ATA

“The ATA bus (a.k.a. IDE bus) is a disk drive interface originally designed for the ISA Bus of the IBM PC/AT”

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
<td>Conner publishes “Task File Interface”</td>
</tr>
<tr>
<td>1989</td>
<td>First presentation at CAM</td>
</tr>
<tr>
<td>1994</td>
<td>ATA-1 Issued</td>
</tr>
<tr>
<td>1996</td>
<td>ATA-2 Issued</td>
</tr>
</tbody>
</table>

- ATA-1 76 pages
  - 28 bit – LBA added
  - 35 Commands
  - PIO/DMA Up to 8.3 MB/sec

1994 – 1996 Spec Wars

- ATA-2 76 pages
  - 48 Commands
  - Up to 16.7 MB/sec
**Parallel ATA**

*The ATA bus (a.k.a. IDE bus) is a disk drive interface originally designed for the ISA Bus of the IBM PC/AT*

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
<td>Conner publishes “Task File Interface”</td>
</tr>
<tr>
<td>1989</td>
<td>First presentation at CAM</td>
</tr>
<tr>
<td>1994</td>
<td>ATA-1 Issued</td>
</tr>
<tr>
<td>1996</td>
<td>ATA-2 Issued</td>
</tr>
<tr>
<td>1998</td>
<td>ATA/ATAPI-4 Issued</td>
</tr>
</tbody>
</table>

- SFF Committee did the work ~1996
- ATA/ATAPI-4 337 pages
  - SCSI transported over ATA
  - 49 Commands
  - Up to 33.3 MB/sec
Parallel ATA

“The ATA bus (a.k.a. IDE bus) is a disk drive interface originally designed for the ISA Bus of the IBM PC/AT”

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
<td>Conner publishes “Task File Interface”</td>
</tr>
<tr>
<td>1989</td>
<td>First presentation at CAM</td>
</tr>
<tr>
<td>1994</td>
<td>ATA-1 Issued</td>
</tr>
<tr>
<td>1996</td>
<td>ATA-2 Issued</td>
</tr>
<tr>
<td>1998</td>
<td>ATA/ATAPI-4 Issued</td>
</tr>
<tr>
<td>2002</td>
<td>ATA-6 obsoletes CHS addressing</td>
</tr>
</tbody>
</table>

- ATA/ATAPI-6 496 pages
  - 71 non Packet Commands
  - 29 Packet Commands
  - Up to 100 MB/sec
- Last parallel spec
How the world turned serial

- 1981 – DEC SI/SDB (RA80 disk drive)

- As systems’ interconnects
  - 1983 - DEC CI
  - 1990 – IBM ESCON

- 2000-3: SATA
- 2002-4: SCSI over Fabrics (SAS)
- 2014-6: NVMe_oF
Into the 80s
JBODs

2021
Universal Storage Server

• Fibre Channel
• FC-NVMe
• iSCSI
• FICON (CKD)
• 4.5 PB
## Subsystem progression 1983 into this century

<table>
<thead>
<tr>
<th>Year</th>
<th>Product</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>1983</td>
<td>DEC HSC 500</td>
<td>Mirror</td>
</tr>
<tr>
<td>1988</td>
<td>AT SS5000</td>
<td>Virtualization, Distributed Parity</td>
</tr>
<tr>
<td>1989</td>
<td>RAID Paper</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Auspex 5000</td>
<td>NAS</td>
</tr>
<tr>
<td>1990</td>
<td>EMC 4200</td>
<td>Emulated CKD</td>
</tr>
<tr>
<td>1997</td>
<td>EMC Celerra</td>
<td>NAS/SAN Server</td>
</tr>
<tr>
<td>2001</td>
<td>IBM</td>
<td>iSCSI</td>
</tr>
</tbody>
</table>
The next speaker
Grant Saviers

Evolution of Storage Interfaces
Evolution of Storage Interfaces

Grant Saviers
May 2021
IEEE Storage Interfaces webinar
Which “Storage Interfaces”? 

• What real time control and data functions are where?
  • Recording technology dependent? Bit modulation, clocking, detection, checks
  • Where is DMA done? CPU? Controller?
  • Error detection & recovery? Competence? Complexity?
  • RAS and failure prediction? None? 365 x 24?
• What wires & protocols at various levels? du jour? Multi-gen?
• O/S driver to controller or channel?
  • Where is logical to physical mapping done?
  • What error recovery where?
  • What performance optimization & where?
  • Application access to bits or files? TP & DB systems bits vs files
My view of the generations

- Gen 1- Logic is expensive – cost & reliability
  - “fire hose” coaxial cables & CPU dedicated to I/O
- Gen 2 – Physically big disks on big systems: > fan out, distance needs
  - Separate storage controllers, strings, channels
  - Sequential logic & state machines

- Microprocessors + ASICs

- Gen 3 – Technology independent bus + protocols
  - Multi generation subsystem HW/SW architectures
- Gen 4 – Intelligent controllers & serial interconnects
- Gen next – what should “storage interfaces” be?
Gen 1 – IBM 1401 + 729 tape drive ~ 1957

• Processor or imbedded processor partial controller is the controller
  • CPU ALU committed to device to memory transfers
  • Program stops during I/O (except Univac I, 5 years prior)
  • Read analog signals error checked, detected, clock recovery in CPU

• Coaxial cable bus - “fire hose” size cable
  • Daisy chain, up to 8 tape drives
  • Analog read signal in same cable
  • CPU “TAU” controls real time tape motion, checks analog
  • Huge connector – “shoe” in IBM terminology, 6x6”, 10#

• Architecture and bus the same for 6 gens of 729 drives (I > VI)
Gen 2 – Disk performance ~1965 - 83
2311, 2314, 3330, 2315, 5440 and clones

• Controller “de-jour” I/F per drive maker, size, & technology generation
  • Market leader often set the “standard” IBM, Diablo, CDC

• Motion control moves to disk drive
  • Incremental or absolute track position commands
  • Overlapped simultaneous seeks

• Read peak detectors usually in device - some areal density independence
  • BUT Clock recovery in controller & write clock & encoded data from controller
  • SERDES, error detection in controller, 1 data transfer at a time

• Single ended digital daisy chain bus; typically 8 devices per controller

• String controllers and channels emerge (high perf tape similar)
  • Large fan-outs to dozens of drives, simultaneous parallel data paths
Gen 3 – Technology independent storage bus

- DEC controller and driver proliferation was a big problem/mess/expensive
  - 16, 18, 32, & 36 bit CPU architectures
  - Many O/S – 16b 5x, 18b 2x, 32b 2x, 36b 2x = cost, complexity, & incomplete
  - Goal “universal storage bus”

- ~1973 Massbus – tape, disk, bulk RAM mixed on single storage device bus
  - Multi tech & multi generation – any mix per controller
    - 2314, 3330, HPT, 3350, 9370/SMD and “double density”
    - NRZI & PE & GCR tape drives
  - Daisy chain, 8 devices, twisted pair differential, 16+P control + 18+P data + 4 timing
  - One controller per CPU implementation – eg RH11, RH70, RH780 (up to 4 per)

- Mixed 2314, 2314-2, 3330, 3330-2, SMD, & 3350 on a single controller were common
DEC Massbus (MB) Implementation

• SERDES, peak detector, clock recovery moved into disk drive
• IBM PCM drive I/F’s modified to DEC spec. 2314s & 3330s & SMDs
  • “Bustle” cabinet bolted to side of drive (from MRX, CDC, ISS)
• Seek, block access, in drive (not CKD capable)
  • Dual port drives for redundancy and performance
  • 1 at a time data transfer, but overlapped control
  • Tape subsystem had head of string controller
• Tape remained “du-jour” at device with MB head of string control
• DEC HPT disk and 3350 had native MB
120v 1/2HP
700 watts

240v 3 phase 3 to 5 HP
10KW start/2.5KW run
40 Mbytes
25msec

5v 0.5 watts
1 Terabyte
200usec
Gen 4 – Intelligence & serial ~ 1981

- DEC Digital Storage Architecture (DSA) & Clustering
- Transactions/sec, I/Os per second leadership goals
  - More devices, further away, increased security & reliability
  - RAS including failure prediction, real time monitoring, & automatic failure recovery & redundancy
- Serial interconnects (DEC “SDI/STI”) are feasible
  - Silicon ASICs are cheaper than copper wires
  - Radial connections to devices & dual port
  - Small cables & connectors for higher fan-ins
  - Transformer coupled for EMC “bombproof”
DEC DSA 1981 – Multi tech & gen/high perf.

• Disks as perfect logical block storage (like SSD now)
  • Logical to physical mapping moved to subsystem
  • Revector bad blocks and tracks into spare space
  • Standardized “on disk” format: spares, error logs, diagnostics (0.4% data loss)

• Robust architectural definition –
  • Mass Storage Control Protocol - MSCP, Class & Port drivers
  • CPU architecture independent
  • Maximum performance optimization
DSA - Increased performance and function

• HSC = Hierarchical Storage Controller – 5 levels
  • Cache, disk, tape, bulk ram, tape libraries
• VAX Cluster Controllers (HSC40 thru HSC95)
  • N-way Mirroring, inline auto restore, rebuild, & catch-up
  • Online mirror volume backup/restore to tape
  • Storage device diagnostics, error logging, revector
  • Multi drive I/O queues, multi level access optimization
  • High fan in/out: up to 56 devices; multiple HSC per cluster
  • Multiple simultaneous data paths of 4 or 8 devices per path
  • Later gen HSCx5 incorporated SCSI interfaces

• Many storage generations – at all levels of hierarchy
• Also “simpler” controllers – Unibus, Qbus, BI bus
And there was/is SCSI

• Somewhat technology independent
  • Disk, tape, CD, & DVD storage
  • RAS - None to limited
  • Many connector “standards”

• Slogged through many generations –
  • 8, 16 or 32 bit wide bus running at 5, 10 or 20 MHz
  • PIO evolved to DMA controllers
  • Narrow evolved to wider daisy chain busses 8 to 16 devices
  • Single ended > LVDS signals

• SAS, SSA, iSCSI architectural melding

• Dumb bits/blocks, & evolved versions for “server” grade RAS & 10e6 MTBF
What should disk storage look like?

• Gordon Bell circa 1983 – “Why aren’t you putting the file system in the disk drive?”
  • VMS file system engineer – “you will get the code only from my dead hands”.
  • Auspex, then NetApp and now the cloud as “perfect” file storage
• SATA proliferation – du jour incrementalism
• Many RAID flavors (DEC sponsored research@ Berkeley)
  • Why when redundancy in cloud and mirroring are so cheap?
  • No on disk data standard means SOL for some failures.
Jai Menon

Evolution of Block Storage System Interfaces
Evolution of Block Storage System Interfaces
(From mainframes to industry-standard servers)

Jai Menon
5/11/21
Talk Outline

• Mainframe block storage system interfaces
  • CKD, ECKD
  • CKD-Emulation on FB disks

• Block Storage systems interfaces for industry-standard servers
  • FC, iSCSI, NVMeOF

No coverage of file or object interfaces
Mainframes – CKD

CKD refers to both disk format and commands (CCWs)

- CKD Disks
- Control unit
- Channels
- Mainframe: MVS, zOS
- Parallel Bus & tag
- CKD commands
  - Seek
  - Set sector
  - Search Key Equal
  - Tic *-8
  - Read Data

- Introduced in 1964 with S/360
- Synchronous operation
- Gaps between fields must be large enough to allow channel to read, match, and turn around to issue next command
- As disk track density increased, more and more space wasted in gaps
- Each CKD disk cost IBM approx. $100M to develop
- Last CDK disk 3390-9 in 1993
  - 11”, 4.2 MB/s, 4200 rpm, 22.7 GB
Mainframes – CKD-Emulation
Possible because processors and memory became cheaper

Key challenges:
• Performance
  • Preserving sector positioning
  • RMW for CKD writes (PTT)
• Reliability
  • Data fields span FB sectors

Emulation Approaches:
• Channel to CU interface
• CU to disk interface

History:
• First done on 9370 (1987)
• EMC (1990)
• IBM RAMAC Array (1994)
  • Emulates 3390 disk
• Still done to this day (DS8900)

Physical track
Logical track

Data fields span FB sectors

Seek
Set sector
Search Key
Equal Tic *-8
Read Data

Parallel Bus & tag

Possible because processors and memory became cheaper
Mainframes – ECKD
processors and memory became cheaper

- Channels
  - Parallel Bus & tag
  - ECKD commands
  - CKD commands

- Control unit
  - Caching
  - NVS
  - Define Extent
  - Locate
  - Read Data

- CKD Disks

- Introduced in 1985 with IBM 3880 Control Unit
- Channel and CU/disk operate asynchronously
- Longer distance possible from mainframe to disks
- Better performance on hits (with caching)
- Some access methods still used CKD

Storage Interfaces
Mainframes – Escon
Introduced in 1990 as part of S/390

- Escon
  (optical fiber, serial)
- ECKD and CKD commands
- Operated originally at 10 MB/sec
- Allowed greater distance
- Allowed switching
- Single CU to 8 mainframes

Control unit
- Cache
- NVS
- Industry-standard FB disks
Mainframes – FICON, zHPF

FICON Introduced in 1998 as part of 5th gen S/390

FICON
- 1, 2, 4, 8, 16 Gbps
- FC switches and Directors
- Up to 100 kms

zHPF (high Performance FICON)
- Introduced in 2009
- Use FCP approach to transfer cmds, data & status
- Optional feature
- TCWs (Transfer Control Words) instead of CCWs
- 90% typical perf improvement
- Fewer channels needed

FICON = ECKD/CKD commands on FC
zHPF = TCWs on FC
Mainframes today – DS8900 controller

Mainframe z15 server

channels

FC 16 Gbps (optical fiber, serial)

FICON = ECKD/CKD commands on FC
zHPF = TCWs on FC
zHyperlink (PCIe connection)

DS 8900 Control unit

Cache

NVS

Industry-standard FB disks/SSDs

Still does CKD-Emulation
Emulates 3390 and 3380 CKD disks
Supports FICON and zHPF
zHyperlink announced in 2017 with z14 & DS8880
- Upto 32 ports per z14
- Synchronous I/O, point to point
- PCIe based
- 18 usecs response time with zHyperlink point to point <150m
Industry-standard servers – SCSI over FC

- **FC HBA**
- **Storage Controller**
  - Cache
  - NVS
  - FB Disks/SSDs

3 FC topologies (Pt to Pt, FCAL, switched)

**FCP = SCSI commands on FC (ANSI standard approval in 1994)**

1st FCP controller - the Sun SPARCStorageArray (March 1994)
- FC point-point to the Sun server and SCSI disks
- 31.5 GB RAID Array, RAID 0, 1, 0+1, 5
- 25 MB/s interface
- 2000 iops
- $1.62/MB
Industry-standard servers – iSCSI = SCSI on TCP/IP on Ethernet

- 1st draft IBM/Cisco 2000; ratified in 2003
- Lower cost than FC
- Longer distance than FC
- Runs on existing network infrastructure
- IBM Total Storage IP Storage 200i, which shipped in 2001, is the earliest iSCSI product in the industry.
Industry-standard servers – NVMe over Fabrics (NVMeoF)
Focus on All Flash Storage Systems
(Extends NVMe over PCIe to other transports)

- Designed to support SSD storage
- Higher performance (64K queues, 64K cmds per queue)
- Lower complexity

Examples
- Pure Storage (NVMe/RoCE) – 2019
- NVMe/TCP
  - Fungible Storage cluster (NVMe/TCP) - 2020
    - 2U, 24 SSDs
    - 15M IOPS in 2U, 120 usecs latency
    - 8x100 Gbps ports
- Lightbits Labs - 2020
Performance Comparison: Random Read

*iSCSI and NVMe/TCP AFA targets built from Intel Gold Servers*

(Manoj Wadekar, FB and Anjaneya Chagam, Intel - 2019)

**Average Latency (μs) of 9 RocksDB Instances**

<table>
<thead>
<tr>
<th>Method</th>
<th>Latency (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSCSI</td>
<td>1,363</td>
</tr>
<tr>
<td>NVMe TCP/IP</td>
<td>118</td>
</tr>
<tr>
<td>Local</td>
<td>112</td>
</tr>
</tbody>
</table>

**Average Throughput (Kops/s) of 9 RocksDB Instances**

<table>
<thead>
<tr>
<th>Method</th>
<th>Throughput (Kops/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSCSI</td>
<td>12</td>
</tr>
<tr>
<td>NVMe TCP/IP</td>
<td>135</td>
</tr>
<tr>
<td>Local</td>
<td>142</td>
</tr>
</tbody>
</table>

Minimal performance overhead with NVMe over TCP/IP
The Next Speaker

- Amber Huffman
- Fast and efficient interfaces for the SSD Era
NVMe® Technology
Fast & Efficient Interfaces for the SSD Era

Amber Huffman
Fellow & Chief Technologist of IP Engineering Group, Intel Corporation
President, NVM Express, Inc.
Memory and Storage Hierarchy Gaps

- Cost-Performance Gap
- Capacity Gap
- Storage Performance Gap

Memory
- Compute Cache
  - 100s MB, ~1ns
- In-Package Memory
  - 1s GB, ~10ns
- Memory
  - 10s GB, <100ns
- Secondary Storage
  - 100s GB, <1usec
- Tertiary Storage
  - 1s TB, <10usecs

Storage
- Primary Storage
  - 10s TB, <100usecs
- Secondary Storage
  - 1s TB, <10usecs
- Tertiary Storage
  - 10s TB, <10 msecs
Memory and Storage Hierarchy Gaps

For illustrative purposes only.
The Evolution of NVMe® Technology

- SCALE OVER FABRICS
- UNIFY PCIE* SSDs
- ENABLE INNOVATION

Deployments

2010 → 2020
Framing the Need
Why Standard Interface for PCIe* SSDs

Enabling Broad Adoption of PCIe SSDs: Enterprise NVMHCI

- PCIe SSDs are attractive and popping out of the woodwork
  - Eliminates SAS infrastructure, plenty of PCIe lanes, bandwidth can be concentrated, lower latency

- PCIe SSDs lack standard OS support & driver infrastructure, since there is no standard host controller interface

- Impact of no standard infrastructure:
  - Requires SSDs vendor to provide drivers for every OS
  - OEM features, like error logs, are implemented in an inconsistent fashion
  - OEMs have to validate multiple drivers

- Enable broad adoption by extending NVMHCI (Non-Volatile Memory Host Controller Interface) to Enterprise
  - Leverage the client NVMHCI i/f, software infrastructure, and Workgroup to fill gap quickly with streamlined solution
  - http://www.intel.com/standards/nvmhci
Building a Coalition
Pull Together Key Stakeholders to Drive Change

The NVMHCI Workgroup includes 55+ members, focused on delivering streamlined NVM solutions.

*Other names and brands may be claimed as the property of others
Deliver the Baseline Specification
And Select a Much Better Name!

**NVM Express** Overview

- NVM Express is a scalable host controller interface designed for Enterprise and Client systems that use PCI Express SSDs
  - Includes optimized register interface and command set
- NVMe was developed by industry consortium of 80+ members and is directed by a 10 company Promoter Group
- NVMe 1.0 published on March 1st, available at nvmexpress.org

*NVMe standardization effort is complete & stable
Product introductions coming in 2012*
Paint the Future

Why It’s Worth Crossing the Adoption Chasm

NVMe*: Efficient SSD Performance

| NVMe* designed for high parallelism and low latency |

<table>
<thead>
<tr>
<th>AHCI1</th>
<th>NVMe EXPRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncacheable Register Reads</td>
<td></td>
</tr>
<tr>
<td>Each consumes 2000 CPU cycles</td>
<td>4 per command</td>
</tr>
<tr>
<td></td>
<td>8000 cycles, ~ 2.5 µs</td>
</tr>
<tr>
<td>0 per command</td>
<td></td>
</tr>
<tr>
<td>MSI-X and Interrupt Steering</td>
<td></td>
</tr>
<tr>
<td>Ensures one core not IOPs bottleneck</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Parallelism &amp; Multiple Threads</td>
<td></td>
</tr>
<tr>
<td>Requires synchronization lock to issue command</td>
<td>No locking, doorbell register per Queue</td>
</tr>
<tr>
<td>Maximum QueueDepth</td>
<td></td>
</tr>
<tr>
<td>Ensures one core not IOPs bottleneck</td>
<td>32</td>
</tr>
<tr>
<td>64K Queues</td>
<td></td>
</tr>
<tr>
<td>64K Commands per Q</td>
<td></td>
</tr>
<tr>
<td>Efficiency for 4KB Commands</td>
<td></td>
</tr>
<tr>
<td>4KB critical in Client and Enterprise</td>
<td>Command parameters require two serialized host DRAM fetches</td>
</tr>
<tr>
<td>Command parameters in one 64B fetch</td>
<td></td>
</tr>
</tbody>
</table>

Scalability for Future NVM

- NVMe* is defined to scale for future NVM
  - Host controller standards live for 10+ years
  - Future NVM may have sub microsecond latencies
- 1M IOPS needs highly efficient driver approach
  - Benefits from removing OS queues, IO scheduler, and SCSI layer while optimizing for NVMe
- Block layer attach reduces overhead > 50%
  - Block layer: 2.8 µs, 9100 cycles
  - Traditional: 6.0 µs, 19500 cycles

Measurement taken on Intel® Core™ i5-2500K 3.3GHz 6MB L3 Cache Quad-Core Desktop Processor using Linux RedHat EL6.0 2.6.32-71 Kernel
Provide Physical Infrastructure
Optimized Form Factors for SSDs

SSD Form Factor Working Group
Driving Enterprise SSD Infrastructure

- Form Factor
  - Benefit from current 2.5" HDD form factor
  - Expand power envelope

- Connector
  - Support Multiple Protocols
    - PCI Express* 3.0, SAS 3.0, SATA 3.0
    - Management Bus
    - Dual port (PCI)
    - Multi-lane capability (PCIe/SAS)
    - SAS & SATA Backward Compatibility

- Hot-Plug
  - Hot-Plug Connector
  - Identify desired drive behavior
  - Define required system behavior

An Optimized Caseless Form Factor

- Attributes to strive for in a new standardized caseless SSD FF:
  - Scalable from small to large capacity points
  - Support SATA Gen3 and two lanes of PCI Express* Gen3
  - Optimize for Z height (e.g. board edge connector, reduce PCB thickness)
  - Mounting strategy will limit board area and reduce fasteners
  - Optimize board size based on BGA NAND package & ensure efficient tiling

Example Solution

mSATA vs Example

Watch for new standard caseless form factor effort
Enabling the Ecosystem
Drivers & BIOS Enabling Required

Reference Drivers for Key OSs

- **Linux**
  - Already accepted into the mainline kernel on kernel.org
  - Open source with GPL license
  - Refer to http://git.infradead.org/users/willy/linux-nvme.git

- **Windows**
  - Baseline developed in collaboration by IDT*, Intel, and LSI*
  - Open source with BSD license
  - Maintenance is collaboration by NVMe WG and Open Fabrics Alliance
  - Refer to https://www.openfabrics.org/resources/developer-tools/nvme-windows-development.html

- **VMware**
  - Initial driver developed by Intel
  - Based on VMware advice, “vmk linux” driver based on Linux version
  - NVMe WG will collaborate with VMware on delivery/maintenance

Reference Drivers for Key OSs (cont.)

- **Solaris**
  - There is a working driver prototype
  - Planned features include:
    - Fully implement and conform to 1.0c spec
    - Efficient block interfaces bypassing complex SCSI code path
    - NUMA optimized queue/interrupt allocation
    - Reliable with error detect and recovery fitting into Solaris’ FMA
    - Build ZFS with multiple sector sizes (512B, 1KB, 2KB, 4KB) on namespaces
    - Fit into all Solaris disk utilities and fwflash(1M) for firmware
    - Boot & install on SPARC and X86
    - Surprise removal support
  - Plan to validate against Oracle* SSD partners
  - Plan to integration into S12 and a future S11 Update Release

- **UEFI**
  - The driver is under development
  - Plan to open source the driver in Q1 ‘13, including bug/patch process
  - Beta quality in Q1’13, production quality Q2’13

IDF2012
INTEL DEVELOPER FORUM

NVMe = NVM Express
Industry Thought Leaders Show What’s Possible…

Designing with the Right Pieces

- multi-core parallelism + PCI Express* Gen3 bandwidth
- NUMA-aware software + NVMe flash
- = millions of IOPs...
- ...at microsecond latencies

new math for storage platforms

Proof Points in EMC* Products

Project Thunder: A networked non-volatile memory appliance

- Building block design center:
  - 10-20TB of flash capacity
  - Consistent 2.5M IOP throughput @ 150us

- NVMe ready:
  - Today: improved latency, reduced processor overhead
  - Tomorrow: ready for nano-second class NVM
  - On the showcase floor until 2pm

NVMe = NVM Express
Industry Thought Leaders
Inbox Driver Support by Microsoft

Microsoft’s Support of NVM Express

- The Natural Progression from SATA for NVM
  - Standardized PCI Express* Storage
  - First devices are enterprise-class
  - High-Density / High-Performance
  - Closing the latency gap with RAM

- Windows* Inbox Driver (StorNVMe.sys)
  - Windows Server 2012 R2 (enterprise)
  - Windows 8.1 (client)
  - Stable Base Driver

- The Storport Model
  - Reduced development cost
    - Offloads Basics: PnP, Power, Setup, Crash, Boot*
  - Mature / Hardened Model
  - Storport optimized for performance
  - RAM-backed NVMe device
    - > 1 million IOPS with < 20µs latencies

StorNVMe Delivers a Great Solution

- StorNVMe Implementation Highlights
  - Uses hardened Enterprise Storage Stack
  - Strives for 1:1 mapping of queues to processors
  - NUMA optimized
  - Asynchronous notification supported
  - Interrupt coalescing supported
  - Rigorous testing on Windows*
  - Firmware Update/Download (via IOCTL)

- With great IOPs
- And low latency

System Configuration:
2 Socket Romley-based Server, 16 physical processors (32), Random Read Workload, Tool: Iometer
Crossing the Chasm
Requires Patience and Perseverance

NVM Express* Deployment is Starting

- First plugfest held May 2013 with 11 companies participating
  - Three devices on Integrator’s List
  - Next plugfest planned for Q4

- Samsung announced first NVM Express* (NVMe) product in July

Learn More in the NVMe Community

Check out the NVMe Community in the Showcase to see NVMe products in action

<table>
<thead>
<tr>
<th>Company</th>
<th>Booth #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dell</td>
<td>726</td>
</tr>
<tr>
<td>Intel</td>
<td>727 &amp; 734</td>
</tr>
<tr>
<td>EMC</td>
<td>728</td>
</tr>
<tr>
<td>Micron</td>
<td>729</td>
</tr>
<tr>
<td>SanDisk</td>
<td>730</td>
</tr>
<tr>
<td>LSI</td>
<td>731</td>
</tr>
<tr>
<td>SNIA</td>
<td>732</td>
</tr>
<tr>
<td>PMC-Sierra</td>
<td>733</td>
</tr>
<tr>
<td>Agilent</td>
<td>735</td>
</tr>
<tr>
<td>Western Digital</td>
<td>736</td>
</tr>
<tr>
<td>Teledyne LeCroy</td>
<td>737</td>
</tr>
<tr>
<td>Viking Technology</td>
<td>738</td>
</tr>
<tr>
<td>Tektronix</td>
<td>739</td>
</tr>
</tbody>
</table>
Evangelize the Value Proposition
Fast, Efficient, Simple

**NVM Express* (NVMe) Delivers Best in Class IOPs**

- 100% random reads: NVMe has >3X better IOPs than SAS 12Gbps
- 70% random reads: NVMe has >2X better IOPs than SAS 12Gbps
- 100% random writes: NVMe has ~ 1.5X better IOPs than SAS 12Gbps

**The Efficiency of NVM Express* (NVMe)**

- CPU cycles in a Data Center are precious
- And, each CPU cycle required for an IO adds latency
- NVMe Express* (NVMe) takes less than half the CPU cycles per IO as SAS

**4K Random Workloads**

- PCIe/NVMe
- SAS 12Gbps
- SATA 6Gbps

**Relative Efficiency**

- Higher is Better

**CPU Clocks per IO**

- Lower is Better

---

Note: PCI Express* (PCIe*)/ NVM Express* (NVMe) Measurements made on Intel® Core™ i7-3770S system @ 3.1G Hz and 4G B Mem running Windows* Server 2012 Standard O/S. PCIe/NVMe SSD is under development. SAS Measurements from HGST Ultrastar* SSD800M/1000M (SAS) Solid State Drive Specification. SATA Measurements from Intel Solid State Drive DC P3700 Series Product Specification. Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of these factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.
Memory and Storage Hierarchy Gaps

For illustrative purposes only.
Memory and Storage Hierarchy Gaps Solutions

For illustrative purposes only
The Evolution of NVMe® Technology

Deployments

 SCALE OVER FABRICS

UNIFY PCIE SSDs

2010  2020
Enabling More Use Cases
NVMe* over Fabrics

Why NVM Express* (NVMe) over Fabrics?

- Simplicity, Efficiency and End-to-End NVM Express* (NVMe) Model
  - NVMe has a single Admin queue pair with 10 required commands
  - NVMe supports up to 64K I/O Queues with 3 required commands
  - Simplicity of protocol enables hardware automated I/O Queues – transport bridge
  - No translation to or from another protocol like SCSI (in firmware/software)
  - Inherent parallelism of multiple I/O Queues is exposed
  - NVMe commands and structures are transferred end-to-end

Goal: Make remote NVMe equivalent to local NVMe, within ~ 10 µs latency.

Architectural Approach

- The NVM Express* (NVMe) Workgroup is starting the definition of NVMe over Fabrics
- The first fabric definition is the RDMA protocol – used with Ethernet and InfiniBand™
- A flexible transport abstraction layer is under definition, useful for many different fabrics
Enabling More Use Cases
NVMe* over Fabrics

Commonality Between PCI Express® and Fabrics

- The vast majority of NVM Express™ (NVMe) is leveraged as-is for Fabrics
  - NVM Subsystem, Namespaces, Commands, Registers/Properties, Power States, Asynchronous Events, Reservations, etc.

- Primary differences reside in enumeration and queuing mechanism

<table>
<thead>
<tr>
<th>Differences</th>
<th>PCI Express® (PCIe)</th>
<th>Fabrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>Bus/Device/Function</td>
<td>NVMe Qualified Name (NQN)</td>
</tr>
<tr>
<td>Discovery</td>
<td>Bus Enumeration</td>
<td>Discovery and Connect commands</td>
</tr>
<tr>
<td>Queuing</td>
<td>Memory-based</td>
<td>Message-based</td>
</tr>
<tr>
<td>Data Transfers</td>
<td>PRPs or SGLs</td>
<td>SGLs only, added Key</td>
</tr>
</tbody>
</table>
Enabling More Use Cases
NVMe* over Fabrics

- Use NVMe* end-to-end to get the simplicity, efficiency and low latency benefits

- NVMe over Fabrics is a thin encapsulation of the base NVMe protocol across a fabric
  - No translation to another protocol (e.g., SCSI)

- NVMe over Fabrics 1.0 includes RDMA binding enabling Ethernet and InfiniBand™
  - INCITS T11 defining Fibre Channel binding

*Other names and brands may be claimed as the property of others.
NVMe® Technology
Roadmap Evolution

NVMe® Development Timeline

2011
- Multi-Path IO
- Namespace Sharing
- Reservations
- Autonomous Power Transition
- Scatter Gather Lists

NVMe 1.1 – Oct ‘12
- Base spec for PCIe®
- Queuing Interface
- Command Set
- E2E Data Protection
- Security

2012
- Namespace Management
- Controller Memory Buffer
- Temperature Thresholds
- Active/Idle Power & RTD3
- Host Memory Buffer
- Live Firmware Update

NVMe 1.2 – Nov ‘14
- Out-of-band management
- Device discovery
- Health & temp monitoring
- Firmware Update
- And more...

NVMe Management Interface 1.0 – Nov ‘15
- Sanitize
- Virtualization
- Directives
- Self-Test & Telemetry
- Boot Partitions
- And more...

2016
- NVMe over Fabrics 1.0 – June ‘16
- Enable NVMe SSDs to efficiently connect via fabrics like: Ethernet, Fibre Channel, InfiniBand™, etc

2017

NVMe is the place for innovation in SSDs.

Flash Memory Summit 2016
Santa Clara, CA

*Other names and brands may be claimed as the property of others.
NVMe® Technology Powers the Connected Universe

<table>
<thead>
<tr>
<th>Units (Ku)</th>
<th>2016</th>
<th>2017</th>
<th>2018</th>
<th>2019</th>
<th>2020*</th>
<th>2021*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise</td>
<td>364</td>
<td>749</td>
<td>1,069</td>
<td>2,045</td>
<td>4,067</td>
<td>5,554</td>
</tr>
<tr>
<td>Cloud</td>
<td>2,051</td>
<td>3,861</td>
<td>10,369</td>
<td>12,276</td>
<td>18,982</td>
<td>21,999</td>
</tr>
<tr>
<td>Client</td>
<td>33,128</td>
<td>48,951</td>
<td>82,587</td>
<td>143,236</td>
<td>202,348</td>
<td>258,791</td>
</tr>
</tbody>
</table>

* Projections provided by Forward Insights Q2’20

- NVMe technology grew from 3 Petabytes to 29 PB shipped per year from 2016 to 2019
- For 2020, the projection is 54 PB
- NVMe technology demand projected to remain strong in a post COVID world
Driving Simplicity in a World of Complexity

- Zoned Namespaces
- Original NVMe
- More expansion
- NVMe/New Technology
- NVMe/RDMA

Too Complex

Too Dependent on Human Glue
Driving Simplicity in a World of Complexity

- Focused on core values… Fast, Simple, Scalable
- Foster areas of innovation AND avoid impact to broadly deployed solutions
- Create an extensible infrastructure that will take us through the next decade of growth
Specification Families

- The core of NVMe and NVMe over Fabrics integrated into a base specification
- Modular command set specifications (Block, Zoned Namespaces, Key Value, etc)
- Modular transport layer specifications (PCI Express*, RDMA, TCP)
- Maintain Management Interface as separate modular specification
# NVM Express Technology Specification Roadmap

<table>
<thead>
<tr>
<th>NVMe Spec</th>
<th>NVMe-oF Spec</th>
<th>NVMe-MI Spec</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2015</strong></td>
<td><strong>2015</strong></td>
<td><strong>2015</strong></td>
</tr>
<tr>
<td>Q1</td>
<td>Q1</td>
<td>Q1</td>
</tr>
<tr>
<td></td>
<td>Q2</td>
<td>Q2</td>
</tr>
<tr>
<td></td>
<td>Q3</td>
<td>Q3</td>
</tr>
<tr>
<td></td>
<td>Q4</td>
<td>Q4</td>
</tr>
<tr>
<td>Q2</td>
<td>Q2</td>
<td>Q2</td>
</tr>
<tr>
<td></td>
<td>Q3</td>
<td>Q3</td>
</tr>
<tr>
<td></td>
<td>Q4</td>
<td>Q4</td>
</tr>
<tr>
<td>Q3</td>
<td>Q3</td>
<td>Q3</td>
</tr>
<tr>
<td></td>
<td>Q4</td>
<td>Q4</td>
</tr>
<tr>
<td>Q4</td>
<td>Q4</td>
<td>Q4</td>
</tr>
</tbody>
</table>

- **2016**
  - NVMe Base Spec
  - NVMe Transport Spec(s)
  - NVMe Command Set Spec(s)
- **2017**
  - NVMe 1.3 May’17
  - NVMe-oF 1.1 July’19
  - NVMe-MI 1.1 May’19
- **2018**
  - IO Determinism (NVM Sets)
  - Persistent Event Log, Rebuild Assist
  - Persistent Memory Region (PMR)
  - Asymmetric Namespace Access (ANA)
- **2019**
  - Enclosure Management
  - In-band Mechanism
  - Storage Device Extension
- **2020**
- **2021**

- NVMe 1.2.1 May’16
  - Transport and protocol
  - RDMA binding
- NVMe 1.3 May’17
  - Sanitize
  - Streams
  - Virtualization
- NVMe 1.4 June’19
  - IO Determinism (NVM Sets)
  - Persistent Event Log, Rebuild Assist
  - Persistent Memory Region (PMR)
  - Asymmetric Namespace Access (ANA)
- NVMe-MI 1.0 Nov’15
  - Out-of-band management
  - Device discovery
  - Health & temp monitoring
  - Firmware Update
- NVMe-MI 1.1 May’19
  - Enclosure Management
  - In-band Mechanism
  - Storage Device Extension
- NVMe-oF 1.0 May’16
  - Transport and protocol
  - RDMA binding
- NVMe-oF 1.1 July’19
  - Enhanced Discovery
  - TCP Transport Binding
- NVMe-MI 1.2 Spec

- Released NVMe specification
- Planned release
Q&A answers by:

I Dal Allan  Tom Gardner  Amber Huffman  Jai Menon  Grant Saviers