You’ve Been Framed!
xPU, GPU and Computational Storage Programming Frameworks

Live Webcast
October 26, 2022
10:00 am PT / 1:00 pm ET
Today’s Presenters

John Kim
SNIA NSF Chair, NVIDIA

James Reinders
Engineer, Intel

Dr. Joseph L. White
Fellow, Dell Technologies

Ben Walker
Software Engineer, Intel

David McIntyre
Director, Product Planning, Samsung

Alex McDonald
Independent Consultant
SNIA - By the Numbers

Industry Leading Organizations: 180
Active Contributing Members: 2,500
IT End Users & Storage Pros Worldwide: 50,000
Technologies We Cover

- Ethernet, Fibre Channel, InfiniBand®
- iSCSI, NVMe-oF™, NFS, SMB
- Virtualized, HCI, Software-defined Storage
- Storage Protocols (block, file, object)
- Securing Data
SNIA Legal Notice

- The material contained in this presentation is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.
Today’s Agenda

- High-level overview:
  - AI/ML: OpenCL, CUDA, SYCL, oneAPI
  - xPU: DOCA, OPI, DASH, IPDK
  - Core data path frameworks: SPDK, DPDK
  - Computational Storage: SNIA Standard 0.8 (in public review), TP4091
- Stay tuned for future deeper dives
OpenCL and CUDA

John Kim, NVIDIA
What Are OpenCL and CUDA?

- Let general purpose computing use GPUs
- Libraries, APIs, compilers and drivers
- Extensions to programming languages

- OpenCL works with any GPU or CPU
- CUDA works only with NVIDIA GPUs
- ROCm works only with AMD GPUs
Choosing How to Accelerate Compute

Which Software Platform?
- Vendor-specific or cross-platform (and which cross-platform?)
- Which hardware do you need to support—GPU, CPU, FPGA, etc.
- Availability of suitable applications and libraries: AI, ML, HPC, etc.

Which Level?
- Applications—high level and easy, but limited to what’s been built
- Libraries and Middleware—more flexible but require more effort
- Drivers and APIs—highest flexibility, potentially fastest performance, but highest amount of work
SYCL and oneAPI

James Reinders, Intel
Q: What is SYCL?
A: C++ solution for heterogeneous programming

An open, multivendor, multiarchitecture approach.

Provides:
1. find a device,
2. manage memory, and
3. manage offloads.

Q: What is DPC++?
A: LLVM compiler (includes clang) implementation of SYCL.

The SYCL book

PDF >670K accesses direct from publisher and Consistently a top C++ Book on Amazon

Q: What is oneAPI?
A: Complementary open standards initiative to provide multivendor multiarchitecture support through libraries, profilers, debuggers, etc.

https://sycl.tech
1. ! Fortran loop
2. do i = 1, n
3.   z(i) = alpha * x(i) + y(i)
4. end do

1. // C++ loop
2. for (int i=0;i<n;i++) {
3.   z[i] = alpha * x[i] + y[i];
4. }

1. // SYCL kernel
2. myq.parallel_for(range{n}, [=](id<1> i) {
3.   z[i] = alpha * x[i] + y[i];
4. }) .wait();

"myq" is how SYCL supports gives us the ability to direct work to our device of choice at runtime (single source: any vendor, any architecture)
Why?
A New Golden Age for Computer Architecture

“The next decade will see a **Cambrian explosion of novel computer architectures**, meaning exciting times for computer architects in academia and industry.”

ACM Turing Award laureates
John Hennessy and David Patterson  
(CACM, Feb 2019, Vol 62, No 2, pp 48-60)
A Complete SYCL Program

```cpp
#include <CL/sycl.hpp>
constexpr int N=16;
using namespace sycl;
int main() {
    queue q;
    int *data = malloc_shared<int>(N, q);
    q.parallel_for(N, [=](auto i) {
        data[i] = i;
    }).wait();
    for (int i=0; i<N; i++) std::cout << data[i] << "\n";
    free(data, q);
    return 0;
}
```

Host code

Accelerator device code

Host code
C++ with SYCL and oneAPI…

...allows our application, in a single version, to identify and use any number of accelerators regardless of vendor or architecture – and do so with access to their best performance.

...are foundational efforts to ensure accelerated computing is open, multivendor, and multiarchitecture.

Higher layers of the software stack benefit without being forced to change.

Offering an essential alternative to proprietary foundations.

https://sycl.tech
https://sycl.tech

SYCL Academy

Online training,
Book download,
Sample codes,
Articles,
and more.

Data Parallel C++

Mastering DPC++ for Programming of Heterogeneous Systems using C++ and SYCL

—
James Reinders
Ben Ashbaugh
James Brodman
Michael Kinsner
John Pennycook
Xinmin Tian

https://sycl.tech

PDF for H&P paper
xPU: DOCA, OPI, DASH, IPDK

Joe White, Dell
A Perspective on DPU Programming and Frameworks

**DPU - Data Processing Unit (aka xPU)**

Effectively a micro-server optimized for dataflow and packet processing providing accelerators, offload engines, & local services

Presents virtual functions to a host (looks like a NIC, GPU, etc)

**Protocol Acceleration**
- NVMe-oF, TCP, RoCE
- P4 Pipeline
- Inline micro-processors

**Encryption**
- Secure Keys
- RegEx
- Hashing

**Embedded Switching**

**Network Packet Processing**

**NIC Functions**

**Host resident software can directly take advantage of DPU offloads and accelerators**

**“Uses”**

**DPU can provide a fixed Services and Functions in an ‘appliance’ model configured through APIs**

**“Uses”**

**Security Enforcement Boundary**

**Independent Lifecycle and Provisioning**

**Multi-tenancy partitioning & policies**

**DPU resident software and services can run independent of the Host OS**

**“On A”**

**Internal “HAL” APIs**
The objective of the Open Programmable Infrastructure Project is to foster a community-driven standards-based open ecosystem for next generation architectures and frameworks based on DPU/IPU-like technologies.

https://opiproject.org  https://github.com/opiproject

https://lists.opiproject.org/g/opi
OPI Project Goals

- Create community-driven standards-based open ecosystem for DPU/IPU-like technologies
- Create vendor agnostic framework and architecture for DPU/IPU-based software stacks
- Reuse existing or define a set of new common APIs for DPU/IPU-like technologies when required
- Provide implementation examples to validate the architectures/ APIs

OPI Technical Deliverables
- Open-Source Projects
- Specifications/Standards
- Reference Platforms
- Test Suites & Cases
- POC/Prototypes
Open source abstraction layer
• Runs across multiple platforms
• Standards Based Accelerations
  – P4 to program the network
  – SPDK for customized storage protocols
  – DPDK or eBPF to accelerate packet flow
  – OVS, SONIC, INT with Deep Insight

A sub-project of OPI
An Implementation of OPI
Across CPU, DPU, IPU & Switch
SONiC-DASH is an open-source project that will "deliver enterprise network performance to critical cloud applications".

- Develop APIs + object models describing network services for the cloud
- The goal of DASH is to be specific enough for SMART Programmable Technologies to optimize network performance and leverage commodity hardware technology to achieve 10x or even 100x stateful connection performance.

https://github.com/Azure/DASH
NVIDIA DOCA
DPU Software Development Kit

DOCA is for DPUs what CUDA is for GPUs

Built on open, industry-standard APIs: DPDK, SPDK, TLS and P4

DOCA libraries provide higher abstraction for enhanced developer experience

Preserves developer investment as DPUs evolve

Support for multiple OS

https://developer.nvidia.com/networking/doca
Core Data Path Frameworks: SPDK, DPDK

Ben Walker, Intel
Overview

The previous section covered xPU-centric frameworks

These xPU frameworks are built on top of existing software projects

These projects handle the data path and contain the device drivers
Use Case: Virtual Switch Offload

- When running many VMs on a single system, they often talk to each other over a virtual network with a virtual, software-based switch.
- This virtual switch is often implemented using DPDK and OVS
- This virtual switch can be offloaded to an xPU.
Data Plane Development Kit (DPDK)

- Long-standing packet processing framework widely deployed in switches and on servers
- Open source, BSD Licensed
- Runs on x86, POWER, and ARM on top of Linux
- Serves as the core data-path framework for software-defined networking across the industry
Virtual Switch Offload Implementation

- The xPU may present multiple PCIe (virtual) functions to be direct assigned to VMs/containers. Switching is done inside the xPU instead of in host system software.
- May be implemented as xPU hardware or as xPU software (firmware?) or as a hybrid
  - When implemented as xPU software, typically based on DPDK and OVS – the same code that ran on the host!
  - But most commonly, xPUs have extra accelerators or a full hardware path for switching and use software only has a fall back for corner cases
Use Case: Block Device Virtualization

- VMs can directly use NVMe or virtio-(blk/scsi) disks. The guest only has drivers for those.
- The real disk may be network attached using some other protocol.
- A storage virtualization target presents VMs with emulated NVMe/virtio disks and forwards I/O using the necessary protocols.
- This storage virtualization can be offloaded to an xPU.
### Storage Performance Development Kit (SPDK)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long-standing storage framework widely deployed in the cloud, NAS, and SAN systems</td>
<td></td>
</tr>
<tr>
<td>Open source, BSD Licensed</td>
<td></td>
</tr>
<tr>
<td>Runs on x86, POWER, and ARM on top of Linux</td>
<td></td>
</tr>
<tr>
<td>Serves as the core data-path framework for software-defined storage across the industry</td>
<td></td>
</tr>
</tbody>
</table>
Storage Virtualization Implementation

- The xPU may present multiple PCIe (virtual) functions to be direct assigned to VMs/containers. Ideally, these appear as NVMe devices.
- May be implemented as xPU hardware or as xPU software (firmware?) or as a hybrid (most common)
  - When implemented as xPU software, typically based on SPDK– the same code that ran on the host!
  - But most commonly, xPUs have extra accelerators (crypto, compression, RDMA) to accelerate data movement
Computational Storage

David McIntyre, Samsung
Problem Statement: Data Processing Resources Can Be Misbalanced

Computational Storage Benefits
- Host Offload (Data Analytics and Mgt)
- Reduce Data Movement
- Application Performance Focus
- Security
Computational Storage Explained

**Classic Architecture**
- DRAM
- CPU
- SSD
- SSD Controller
- NAND Flash

**Computational Storage Architecture**
- DRAM
- CPU
- PCIe
- SSD
- Compute Engine
- SSD Controller
- NAND Flash

**Key Points**
- CPU overloaded
- Large data transfers
- Bring Compute to the Data
- Offloaded processes run near the data high, scalable total internal bandwidth processing and bandwidth scales with data

**Performance**
- Performance Ceiling
- Performance Scales with Data

# SSDs / Server
- Scalable Acceleration with SmartSSD
- 2x — 10x
Standards based Computational Storage

- **SNIA CS API Specification (Host)**
  - Defines methodology for Host Applications to initiate commands to computational storage resource (e.g. scan query)

- **NVMe TP4091 Specification (Device)**
  - Defines command environment from host to target (e.g. Compute Engine)
  - Target supports NVMe 2.0 specification

![Diagram of Computational Storage Programming Methodology]

- Application
- SNIA CS API
- NVMe Driver
- Computational Storage Device
- Execute Command (Scan Query)
- Result (Scan Query)
Computational Storage Scan Acceleration Programming Build Steps

1. **Build Open Sourced Software Modules**
   - Custom scan extension, scan server, CS API lib, etc.

2. **Enable PostgreSQL to use custom scan extension**
   - Modify just one line of postgresql.conf
   - No PostgreSQL recompilation required

3. **Load Samsung NVMe device driver**
   - TP4091 enabled

4. **Run application**
Today We Covered…

- High-level overview of significant & important developments:
  - AI/ML: OpenCL, CUDA, SYCL, oneAPI
  - xPU: DOCA, OPI, DASH, IPDK
  - Core data path frameworks: SPDK, DPDK
  - Computational Storage: SNIA Standard 0.8 (in public review), TP4091

- What specific topics would you like us to cover in future deeper dives?
  - Use the feedback text area as you leave the presentation
After this Webcast

- Please rate this webcast and provide us with your feedback
- This webcast and a copy of the slides are available at the SNIA Educational Library [https://www.snia.org/educational-library](https://www.snia.org/educational-library)
- A Q&A from this webcast, including answers to questions we couldn’t get to today, will be posted on our blog at [https://sniansfblog.org/](https://sniansfblog.org/)
- Follow us on Twitter [@SNIANSF](https://twitter.com/SNIANSF)
After this Webcast

- Please rate this webcast and provide us with your feedback
- This webcast and a copy of the slides are available at the SNIA Educational Library [https://www.snia.org/educational-library](https://www.snia.org/educational-library)
- A Q&A from this webcast, including answers to questions we couldn’t get to today, will be posted on our blog at [https://sniansfblog.org/](https://sniansfblog.org/)
- Follow us on Twitter [@SNIANSF](https://twitter.com/SNIANSF)
Thank You