Accelerating Data-Centers using NVMe and CUDA

Stephen Bates, PhD
Technical Director, CSTO,
PMC-Sierra

Project Donard @ PMC-Sierra

- Donard is a PMC-Sierra CTO project that leverages NVM Express (NVMe) to accelerate data-center applications.
- It uses NVMe and (Remote) Direct Memory Access to enable the “Trifecta” of compute, network and storage PCIe devices.
- Builds on work done by Nvidia, Mellanox and others.

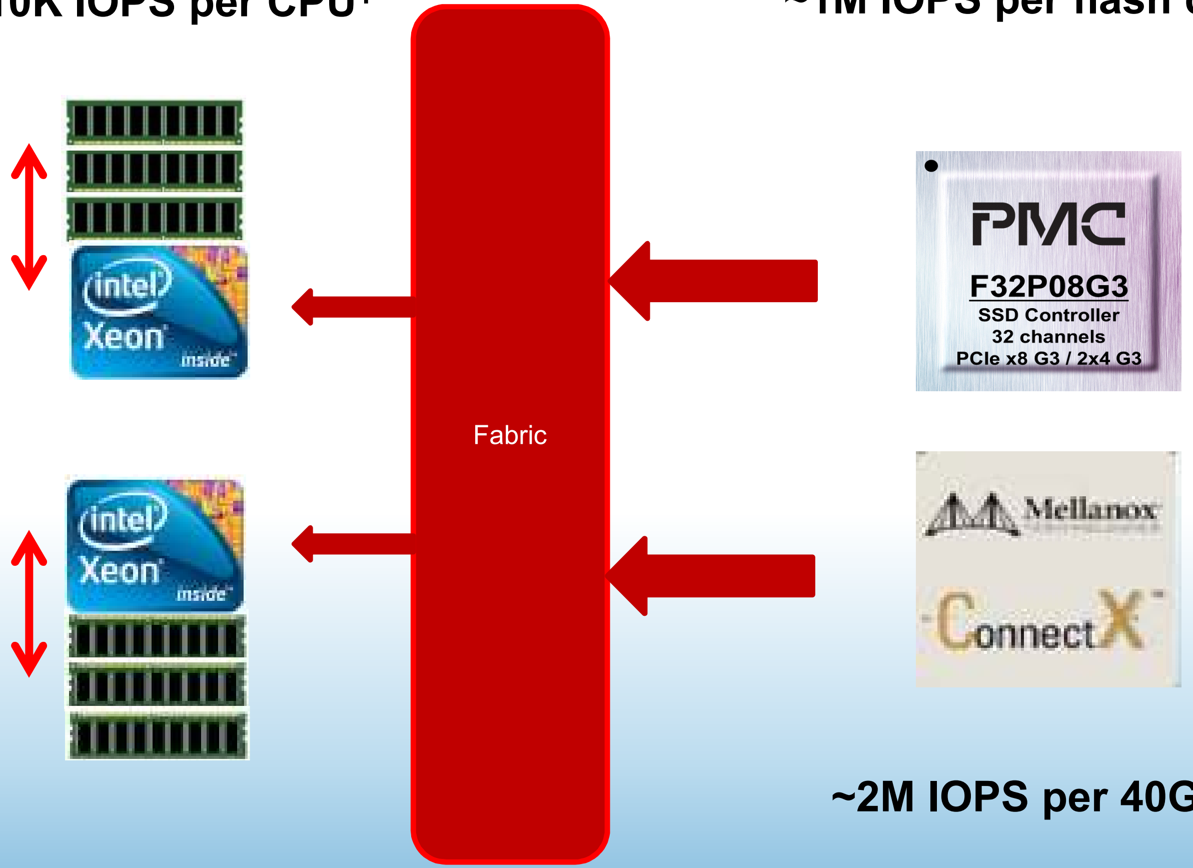
The Problem: x86 Status Quo

Low parallelism = CPU / Memory / fabric saturation

<10K IOPS per CPU[1]

~1M IOPS per flash device

Fast Data workloads requiring deep parallelism are not efficient with x86



Fabric

~2M IOPS per 40G Link

Fabric becomes overloaded

[1] Assuming computation per IOP is high (e.g. image search, encryption, audio processing).

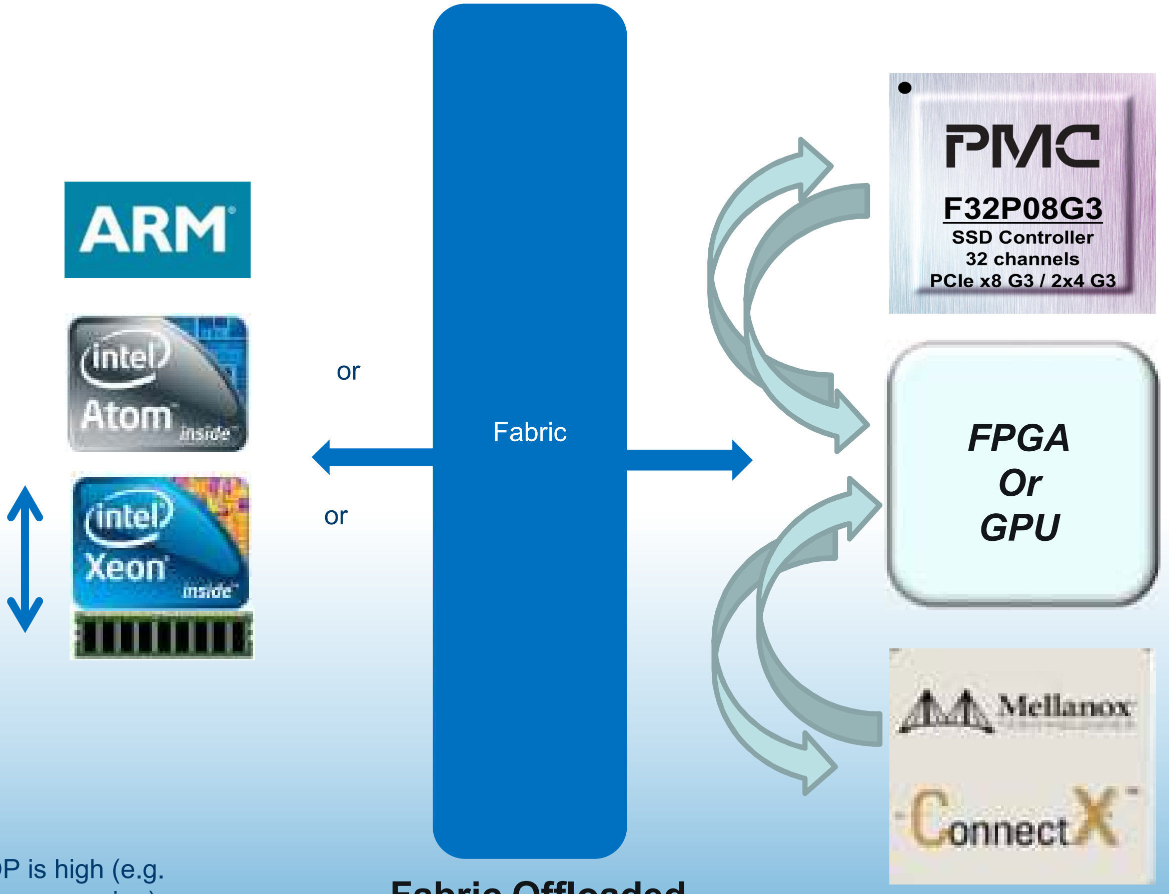
The Solution: Donard

Pre-process algorithms in the data path

A >99% filter rate drops IOPs to 10K.

Re-Balance

Simple Control And Management



or

or

Fabric

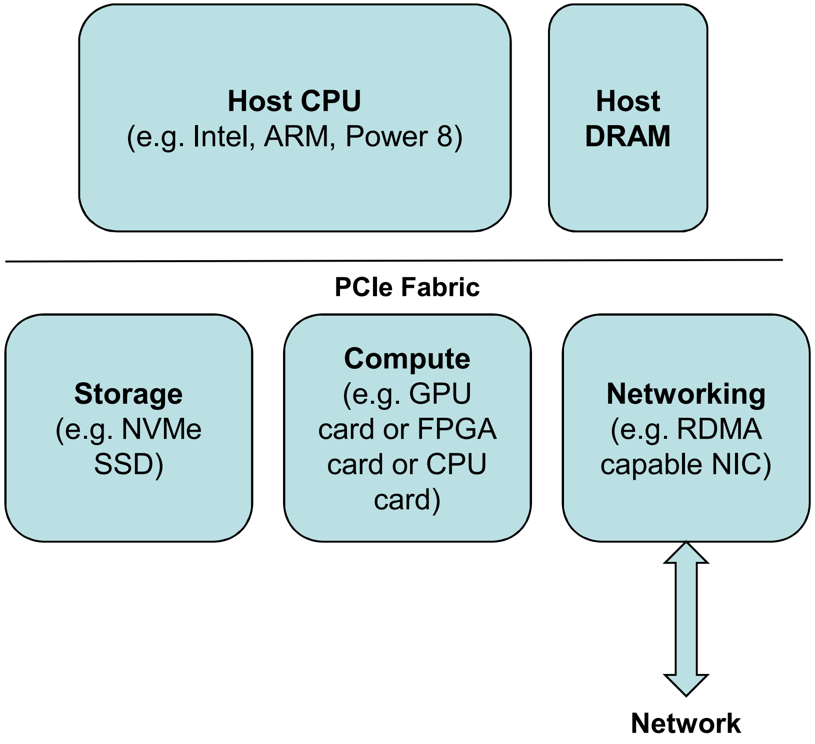
FPGA Or GPU

Fabric Offloaded

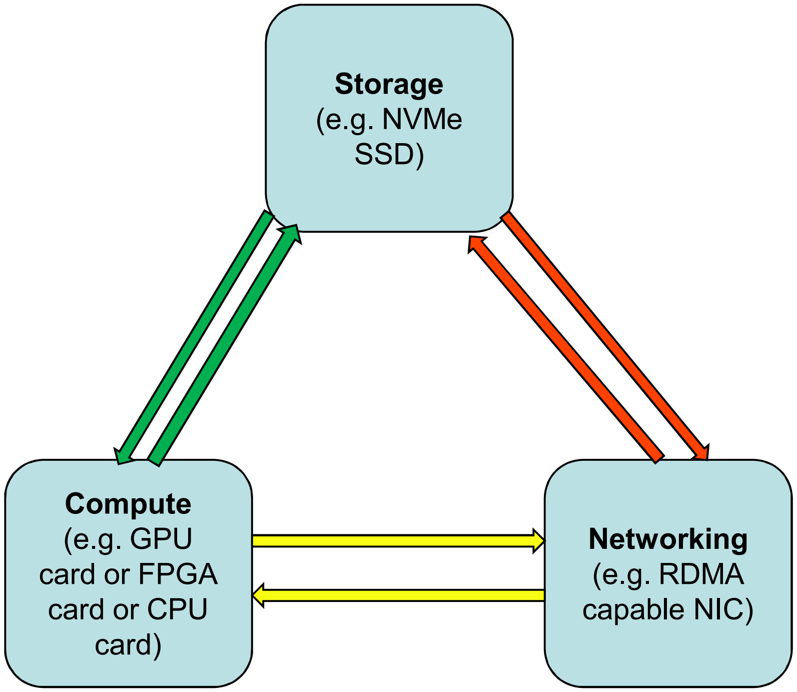
[1] Assuming computation per IOP is high (e.g. image search, encryption, audio processing).

The PCIe “Trifecta”

- We want to enable the Trifecta on a PCIe fabric.
- This work leverages NVMe to enable two edges of this Trifecta.
- Others have worked on other arms (see next slide).



The PCIe “Trifecta”



Storage
(e.g. NVMe SSD)

Compute
(e.g. GPU card or FPGA card or CPU card)

Networking
(e.g. RDMA capable NIC)

🟩 **This work.**

🟨 **Work done by others (e.g. GpuDirect).**

🟥 **Future work.**

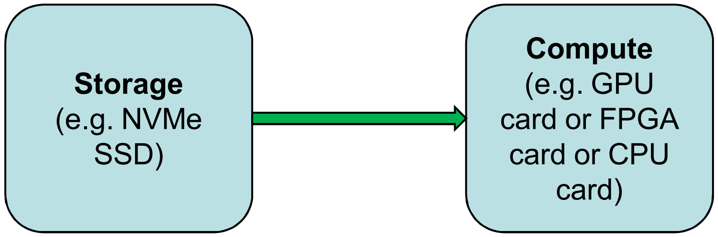
Why NVM Express?

- NVM Express (NVMe) provides a consistent, open-source interface into PCIe storage devices.
- An NVMe driver has been part of both Linux and Windows Server for quite some time now.
- NVMe is extendible and is the focus of optimization within kernel.org[1].
- Using NVMe makes Donard much more scalable and amenable to community development.
- The work in this paper should be applicable to any NVMe compliant drive.

[1] http://kernelnewbies.org/Linux_3.13#head-3e5f0c2bcebc98efd197e3036dd814eadd62839c

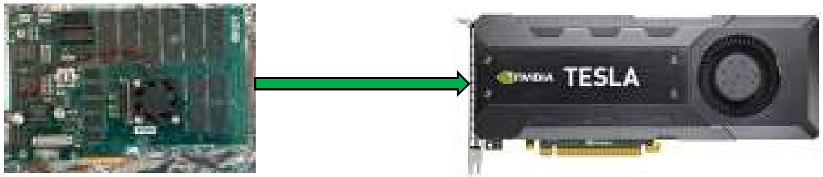
Storage->Compute

- Built a Linux server running kernel 3.13.
- Installed an NVMe SSD and a Nvidia Tesla K20c.
- Modified the NVMe module in the kernel to add a new IOCTL that use DMA between SSD and the GPU card.
- Used CUDA 6.0 Peer-To-Peer (p2p) APIs to enable the DMA.
- Measured the impact of the new IOCTL on bandwidth and host DRAM utilization.



Storage (e.g. NVMe SSD)

Compute (e.g. GPU card or FPGA card or CPU card)



Technique	Bandwidth[1] (GB/s)	DRAM Utilization[2]
Classical	1.9	5230.0
Donard (DMA)	2.5	1.0

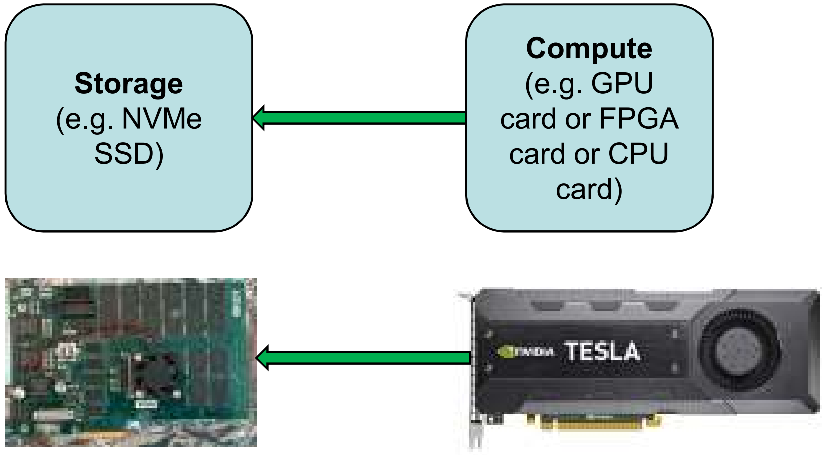
[1] Bandwidth was measured on our server which had a very standard PCIe fabric using a total transfer size of GB. Tests run 10 times. Results may vary depending on your PCIe architecture.

[2] DRAM utilization estimated using the page fault counters in the x86 CPU. Normalized to Donard performance.

Compute->Storage

- Similar IOCTL as previous slide with direction changed.
- Had to resolve the file extents issue. Several options:
  1. Overwrite an existing file.
  2. Create a file using existing OS constructs (slow).
  3. Modify the file stats properties to allow uninitialized extents – possible security issue[1].
- Measured the impact of the new IOCTL on bandwidth and host DRAM utilization.
- Still trying to determine why DMA method is slower. Suspect issue with PCIe architecture in server.

[1]https://lwn.net/Articles/492959/



Storage (e.g. NVMe SSD)

Compute (e.g. GPU card or FPGA card or CPU card)

Technique	Bandwidth[1] (GB/s)	DRAM Utilization[2]
Classical	1.51	6012.0
Donard (DMA)	0.65	1.0

[1] Bandwidth was measured on our server which had a very standard PCIe fabric using a total transfer size of GB. Tests run 10 times. Results may vary depending on your PCIe architecture.

[2] DRAM utilization estimated using the page fault counters in the x86 CPU. Normalized to Donard performance.

Donard in the Data-Center

- Data-Centers (DC) are deploying flash in a direct attach model to aid in application acceleration.
- Although SATA is prevalent today many DCs see a shift to PCIe attached flash and like the idea of using NVMe.
- Some DC customers are offloading applications to heterogeneous compute platforms such as GPU cards and FPGA cards[1].
- Donard assists in this offload and also reduces the burden on the host processor and host DRAM.

[1] **A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services**, *41st Annual International Symposium on Computer Architecture (ISCA),* June 2014.

Donard DC Application - Haystack

- We wrote a program to search for the PMC logo in a large (10,000+) image database.
- Performance improved as we migrated to DMA on a SSD+GPU compared to a traditional solution.
- Note it also moves the bottleneck from the host DRAM interface to the GPU.
- Other applications might include sorting and write-caching.

	HDD	SDD	
	Mpix/s	Mpix/s	Bottleneck
CPU	77.0	122.8	CPU
CUDA	95.1	312.5	DRAM
DONARD	**N/A**	**534.2**	**GPU**

Next Steps

- Determine reason for slow write performance from GPU into SSD.
- Add RDMA capable NICs to the Donard platform and complete the “Trifecta”, this is work in progress.
- Enable Donard within the community so others can benefit and contribute (GitHub?). Already working with Steve Swanson’s team at UCSD.
- Work with the NVM Express standards body to incorporate some of the Donard ideas into NVMe.

Thank You