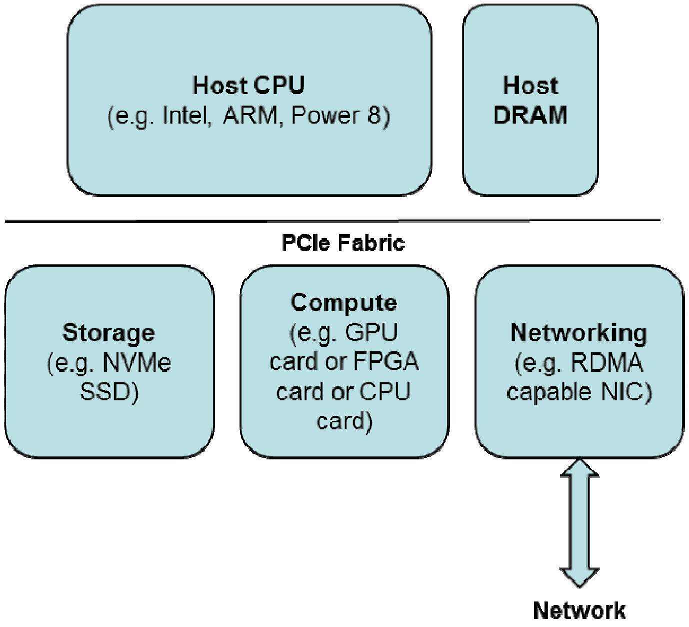
SDC
STORAGE DEVELOPER CONFERENCE
SNIA ■ SANTA CLARA, 2015

Donard: NVM Express for Peer-2-Peer between SSDs and other PCIe Devices

Stephen Bates
PMC

Donard Introduction

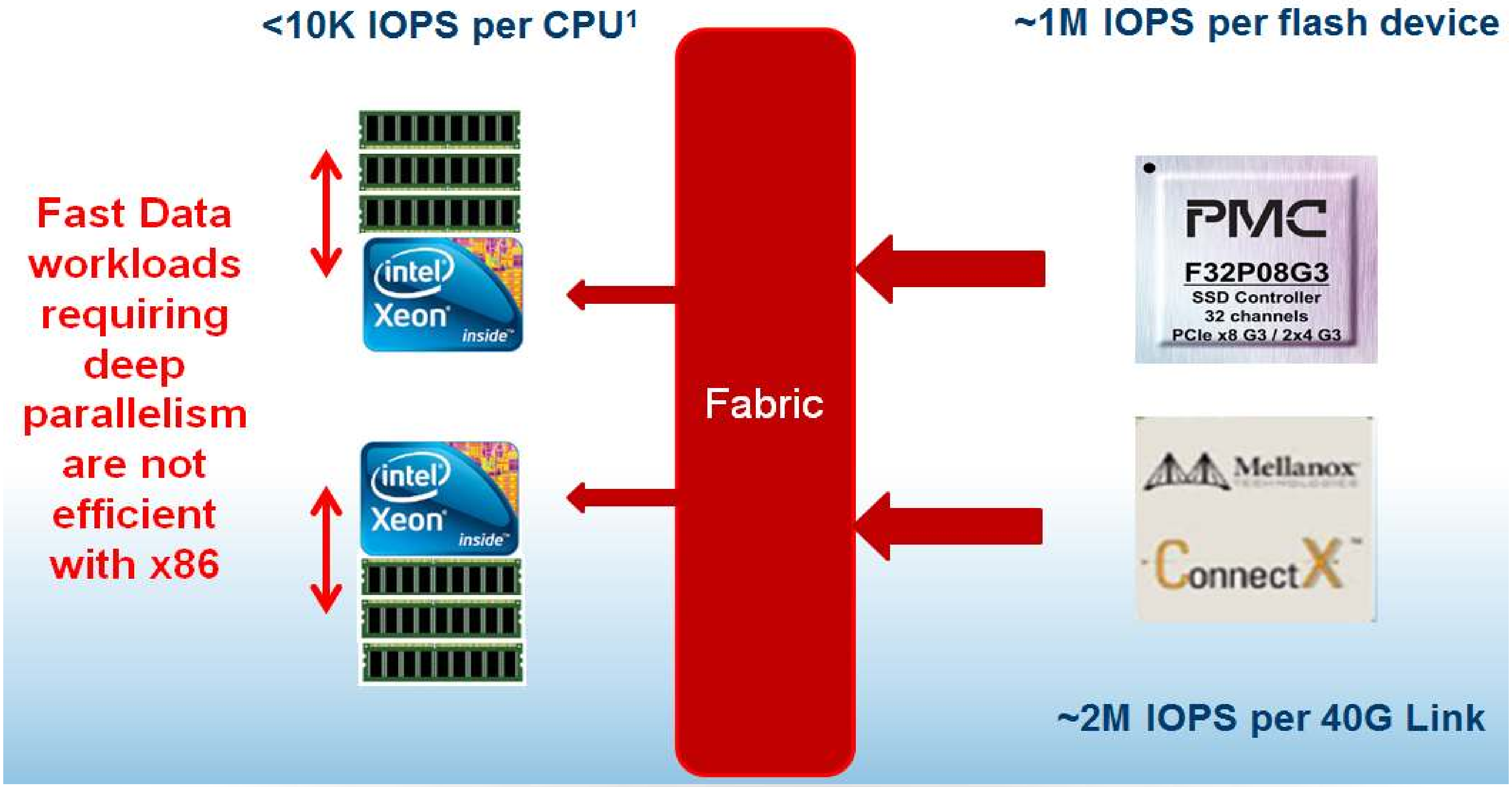
- Donard is a **CSTO program** at PMC.
- Builds on top of the **standard NVM Express (NVMe) Linux driver** to enable **p2p transfers** between PCIe SSDs and **3rd party PCIe devices** (such as GPUs or NICs).
- Consists of a **HW reference design**, **Linux kernel modifications** and a Donard library (**libDonard**).
- Providing **HW design, documents** and **source code**.



PCIe Devices include

- NVMe SSDs
- RDMA capable NICs
- GPU cards
- NVMe compliant NVRAM cards

Why Donard? Goals and Objectives



<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

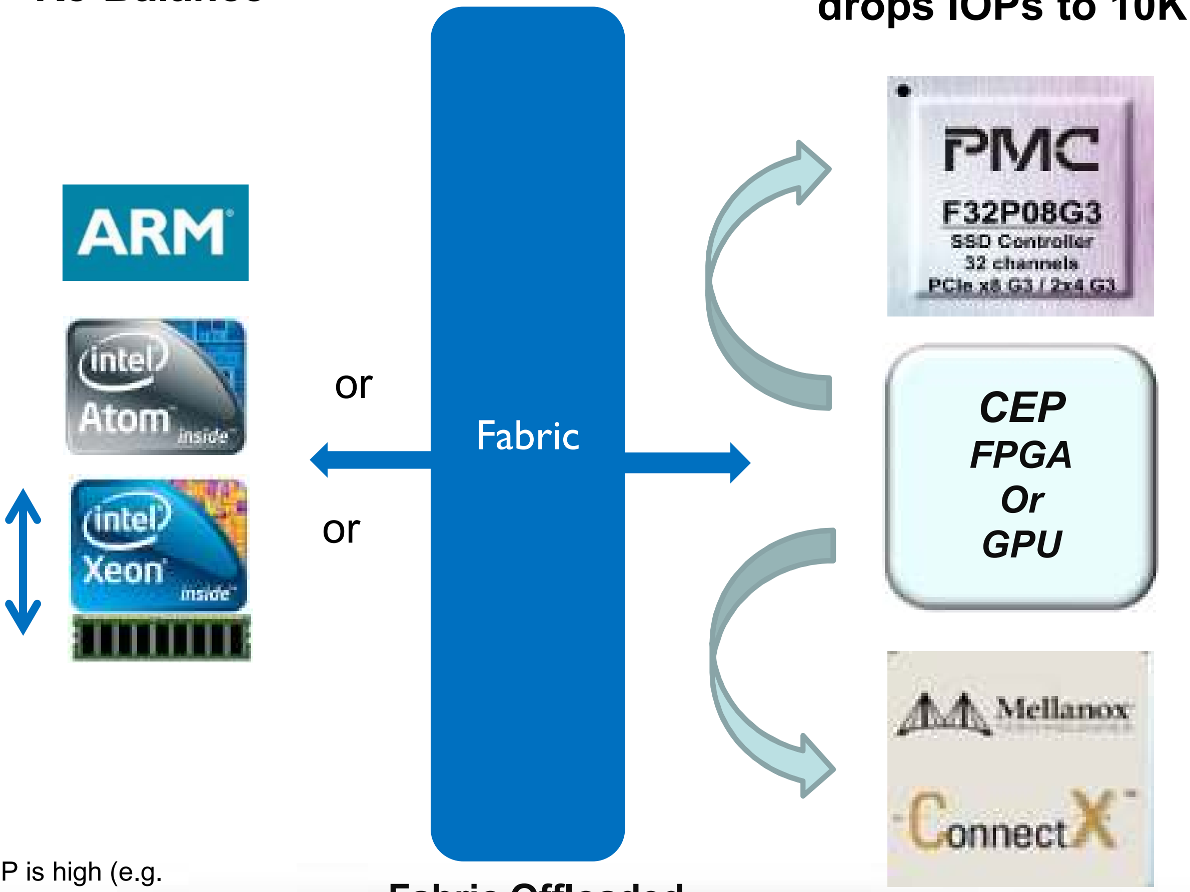
The Solution: Complex Event Processing

Pre-process Fast Data Algo’s in the data path

Re-Balance

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

Simple Control And Management



or

or

Fabric

CEP FPGA Or GPU

Re-locate Parallel Application pre-processing needs into Data Path with high-speed I/O’s

Fabric Offloaded

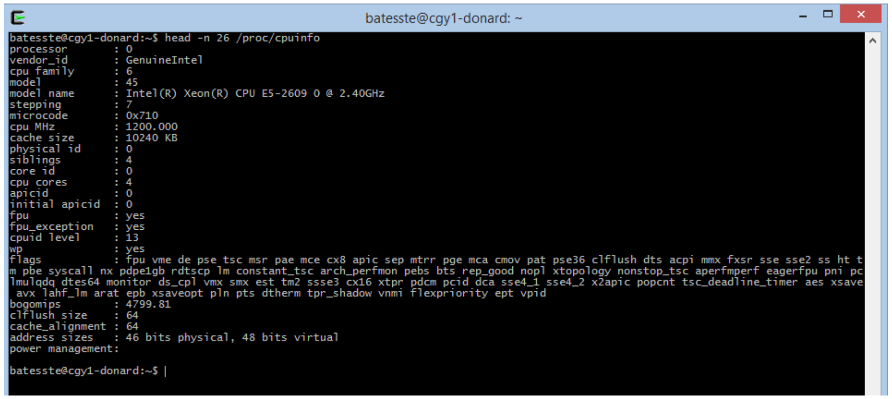
[1] Assuming computation per IOP is high (e.g. image search, encryption, audio processing).

Donard Hardware

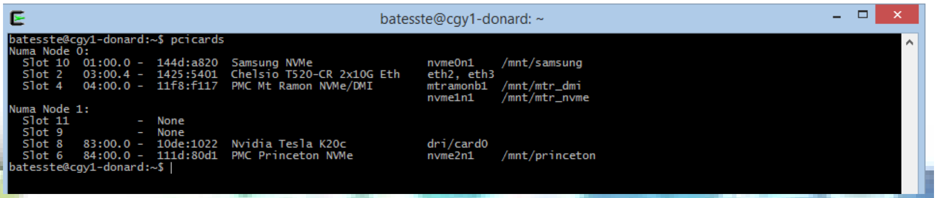
- Our current Donard HW is based on a **SuperMicro** server.
- Both CPU sockets were populated with quad-core **Intel(R) Xeon(R) CPU** E5-2609 0 @ 2.40GHz (SandyBridge).
- PCIe cards were always directly attached to the CPU PCIe lanes (not the PCH nor a PCIe switch). Wrote a simple tool to show physical mapping.
- PCIe Devices included:
  - PMC SSD eval cards.
  - PMC Mt Ramon (NVRAM) card.
  - Samsung XS1715 SSD
  - Chelsio T520-CR 2x10Gbe iWARP NIC
  - Mellanox MT27600 56G IB NIC
  - Nvidia Telsla K20c GPU card

About This Motherboard

The Super X9DRG-QF motherboard supports dual Intel E5-2600(v2) Series Processors (Socket R LGA 2011) that offer QPI (Intel QuickPath Interface) Technology (V.1.1), providing point-to-point connection with a transfer speed of up to 8.0 TG/s. With the C602 chipset built in, the X9DRG-QF motherboard supports Intel® Management Engine (ME), Rapid Storage Technology, Digital Media Interface (DMI), PCI-E Gen. 3.0, and up to 1866 MHz DDR3 memory. This motherboard is ideal for application 4U/4GU server platforms. Please refer to our website (http://www.supermicro.com) for CPU and memory support updates.



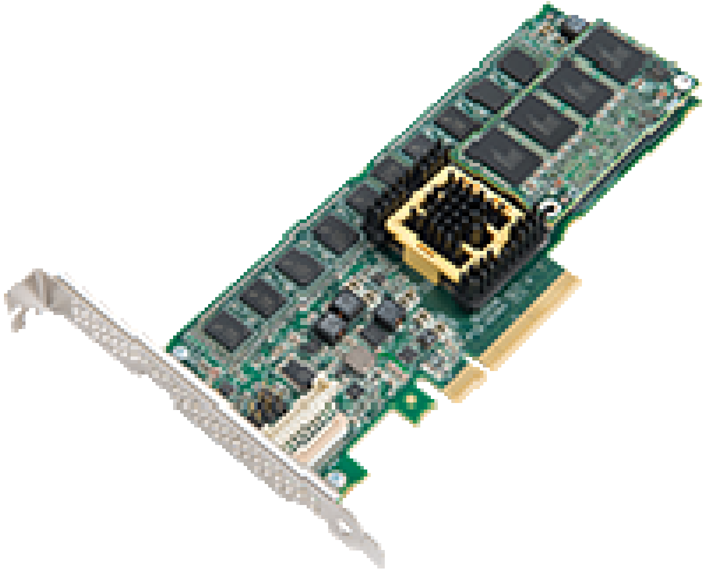
```
batesste@cgy1-donard:~$ head -n 26 /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz
stepping        : 7
microcode       : 0x710
cpu MHz         : 1200.000
cache size      : 10240 KB
physical id     : 0
siblings        : 4
core id         : 0
cpu cores       : 4
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht t
m pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pc
lmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave
 avx lahf_lm arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid
bogomips        : 4799.81
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

batesste@cgy1-donard:~$ |
```



```
batesste@cgy1-donard:~$ pcicards
Numa Node 0:
  Slot 10  01:00.0 -  144d:a820  Samsung NVMe                nvme0n1    /mnt/samsung
  Slot 2   03:00.4 -  1425:5401  Chelsio T520-CR 2x10G Eth   eth2, eth3
  Slot 4   04:00.0 -  11f8:f117  PMC Mt Ramon NVMe/DMI       mtramonb1  /mnt/mtr_dmi
                                                             nvme1n1    /mnt/mtr_nvme
Numa Node 1:
  Slot 11          -  None
  Slot 9           -  None
  Slot 8   83:00.0 -  10de:1022  Nvidia Tesla K20c           dri/card0
  Slot 6   84:00.0 -  111d:80d1  PMC Princeton NVMe          nvme2n1    /mnt/princeton
batesste@cgy1-donard:~$ |
```

PMC FlashTec NVRAM card (Mt Ramon)

- DRAM cache accessed using of NVM express SSD controller (Princeton).
- Can access DRAM via block driver (NVMe) or proprietary character based drive.
- Note NVM Express 1.2 was ratified Nov 2014 and standardizes controller memory exposure via NVMe (driver work in progress).
- Almost 1 million 4KB IOPs. Low latency. 10 million 64B IOPs.
- In production in late Q1 2015. 4GB, 8GB and 16GB SKUs.

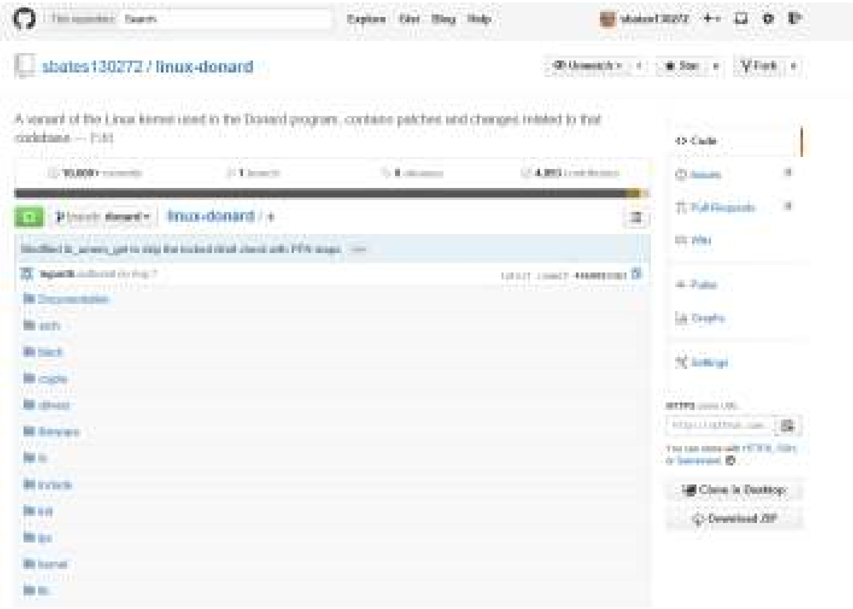


This card has the distinct advantage of working with both the NVMe driver and a character based driver.

A super-capacitor ensures DRAM contents flushed to NAND on sudden power loss.

Donard OS: linux-donard

- Our current Donard SW is based on a **Debian Linux**. However it *should* work with most Linux distros.
- Baseline of the kernel is pulled from **main kernel** repo (git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git). Our last rebase was against **3.19.1**
- The kernel is updated to provide some non-mainline functionality. Use git log on the git repo to review those changes.
    - Patch for fallocate() to allow for remote writes.
    - PMEM+DAX for persistent memory devices (NVRAM and NVDIMM).
- Our version of the kernel is online at https://github.com/sbates130272/linux-donard

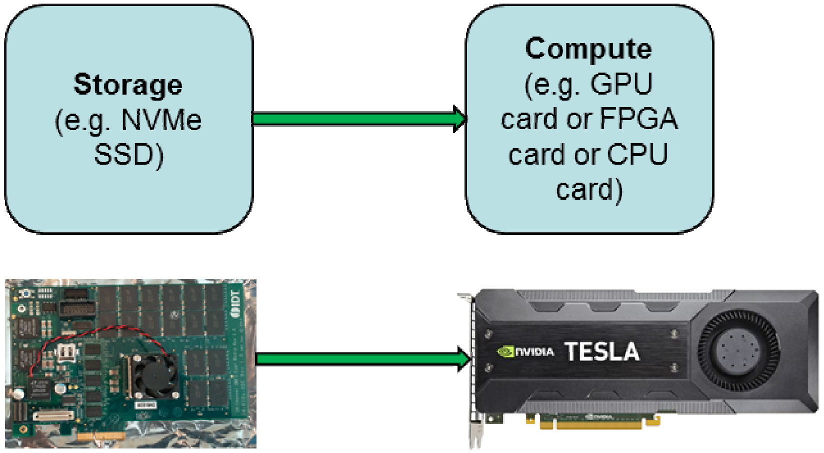


sbates130272 / linux-donard

A variant of the Linux kernel used in the Donard program, contains patches and changes related to that codebase — Edit

Donard Performance Example

- Modified the **NVMe module** in the kernel to add a new IOCTL that uses **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**
- **Donard** increases bandwidth and reduces host memory requirements.



Storage (e.g. NVMe SSD)

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

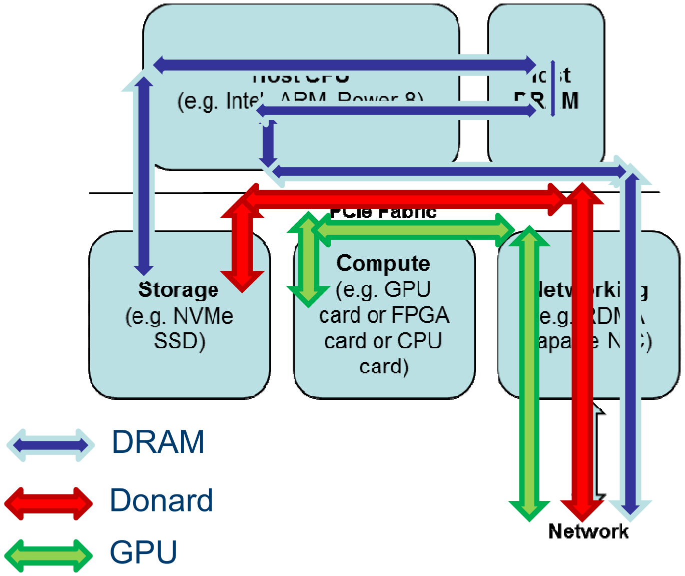
Technique	Bandwidth[1] (GB/s)	DRAM Volume (GB)[2]
Classical	1.9	3.85
Donard (DMA)	2.5	0.56

[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 likwid-perfctr utility

Donard Use-Case: RDMA Write Cache

- Combine **Donard** with **Mt Ramon** and **RDMA capable NIC** to off-load write caches in data-centers.
- Implemented using Donard environment using **Chelsio** NIC (to be repeated ASAP with **Mellanox**).
- Saw **CPU and DRAM off-load** using **Donard**. Also expect a latency reduction but this has not been measured.
- Several **Intel peer-2-peer** issues still need to be **root-caused**. Expect this will improve **Bandwidth**.



	Technique	Bandwidth (MB/s)	DRAM Utilization[1]
Reads	DRAM	1170	36,000,000
	GPU	800	4,000,000
Writes	DRAM	1170	34,000,000
	Donard	**1170**	**250,000**
	GPU	1170	4,000,000

[1] Measured DRAM accesses using likwid-perfctr

DONARD Use-Case: Image Search

	Technique	Bandwidth (GB/s)	DRAM Utilization[1]
Reads	Classical	1.90	5230
	Donard	2.50	1
Writes	Classical	1.51	6012
	Donard	0.65	1

- Donard offloads host DRAM and can improve throughput (for reads).
- Still working on a write issue. PCIe switch helps to remedy this.
- Approximately 6 MM and $5,000 in equipment invested to date.

Applications

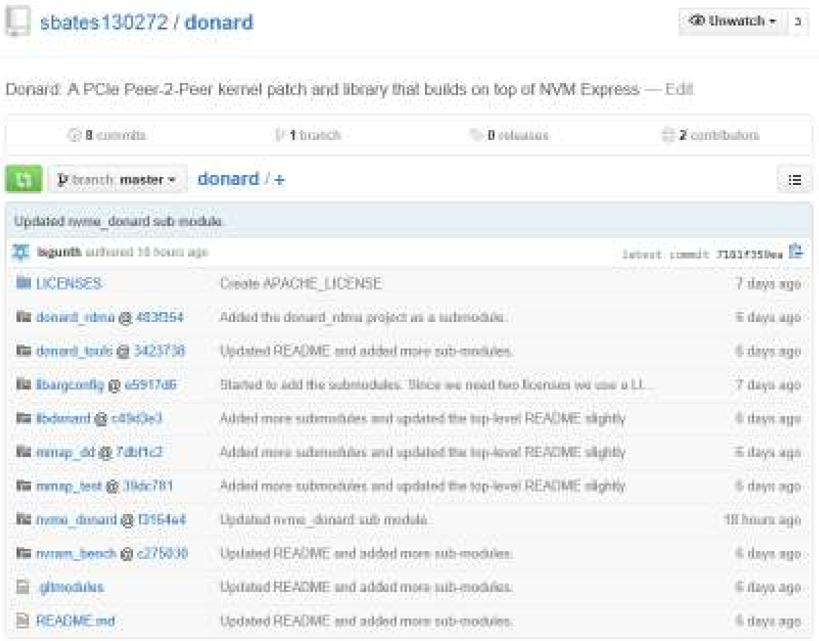
- We wrote a program to search for the PMC logo in a large (10,000+) image database.
- Performance improved as we migrated to DMA on a SSD+GPU compared to a traditional solution.
- Note it also moves the bottleneck from the host DRAM interface to the GPU.
- Other applications might include sorting and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

	HDD	SDD	
	Mpix/s	Mpix/s	Bottleneck
CPU	77.0	122.8	CPU
CUDA[1]	95.1	312.5	DRAM
DONARD	**N/A**	**534.2**	**GPU**

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

Donard Codebase

- PMC has released the Donard code-base under a mix of Apache and GPL licensing:
  - All PMC developed code is released under Apache – use and abuse as you see fit.
  - Any code that is based off Linux kernel is released under GPL 2.0 (as per kernel requirements).
- The code is soft-released at GitHub (https://github.com/sbates130272/donard).
- Code is released without any assumption of support or liability.



sbates130272 / donard

Donard: A PCIe Peer-2-Peer kernel patch and library that builds on top of NVM Express — Edit

8 commits · 1 branch · 0 releases · 2 contributors

branch: master · donard / +

Updated nvme_donard sub module.

logunth authored 18 hours ago · latest commit 7181f350ea

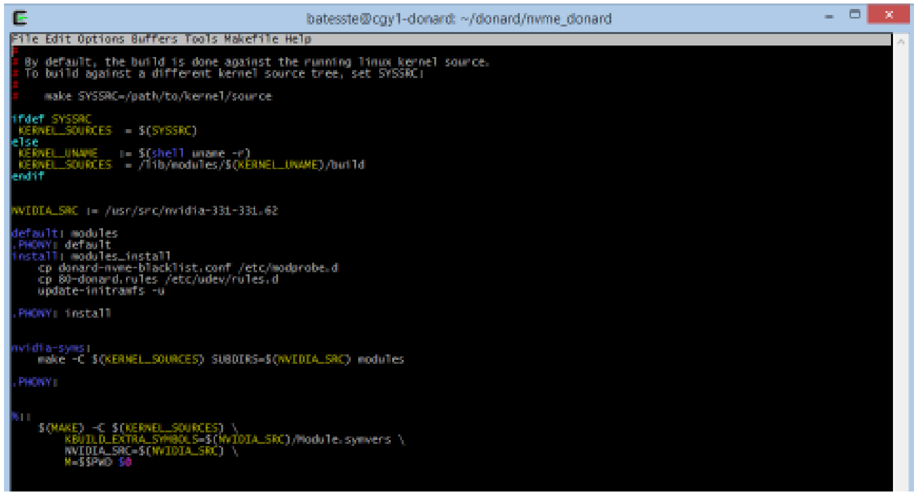
LICENSES	Create APACHE_LICENSE	7 days ago
donard_rdma @ 483f354	Added the donard_rdma project as a submodule.	5 days ago
donard_tools @ 3423738	Updated README and added more sub-modules.	6 days ago
libargconfig @ e5917d6	Started to add the submodules. Since we need two licenses we use a LI…	7 days ago
libdonard @ c49d3e3	Added more submodules and updated the top-level README slightly	6 days ago
mmap_dd @ 7dbf1c2	Added more submodules and updated the top-level README slightly	5 days ago
mmap_test @ 39dc781	Added more submodules and updated the top-level README slightly	6 days ago
nvme_donard @ f3154e4	Updated nvme_donard sub module.	18 hours ago
nvram_bench @ c275030	Updated README and added more sub-modules.	6 days ago
.gitmodules	Updated README and added more sub-modules.	6 days ago
README.md	Updated README and added more sub-modules.	6 days ago

The main GitHub repo calls other repos as submodules.

There is also a separate repo containing the kernel mods we applied to get this to work.

Donard Codebase – nvme_donard

- The nvme_donard repo contains the code needed to build the nvme_donard.ko kernel module.
- Since it enables p2p with the Nvidia card you need to know where the Nvidia driver code is installed (see Makefile)
- By default builds against running kernel. However you can build against any other dev installed kernel.
- Includes a install rule to replace the standard nvme module with nvme_donard (nvme is blacklisted in /etc/mod


batesste@cgy1-donard: ~/donard/nvme_donard
File Edit Options Buffers Tools Makefile Help

```
#
# By default, the build is done against the running linux kernel source.
# To build against a different kernel source tree, set SYSSRC:
#
#    make SYSSRC=/path/to/kernel/source

ifdef SYSSRC
 KERNEL_SOURCES  = $(SYSSRC)
else
 KERNEL_UNAME    := $(shell uname -r)
 KERNEL_SOURCES  = /lib/modules/$(KERNEL_UNAME)/build
endif


NVIDIA_SRC := /usr/src/nvidia-331-331.62

default: modules
.PHONY: default
install: modules_install
	cp donard-nvme-blacklist.conf /etc/modprobe.d
	cp 80-donard.rules /etc/udev/rules.d
	update-initramfs -u

.PHONY: install


nvidia-syms:
	make -C $(KERNEL_SOURCES) SUBDIRS=$(NVIDIA_SRC) modules

.PHONY:


%::
	$(MAKE) -C $(KERNEL_SOURCES) \
		KBUILD_EXTRA_SYMBOLS=$(NVIDIA_SRC)/Module.symvers \
		NVIDIA_SRC=$(NVIDIA_SRC) \
		M=$$PWD $@
```

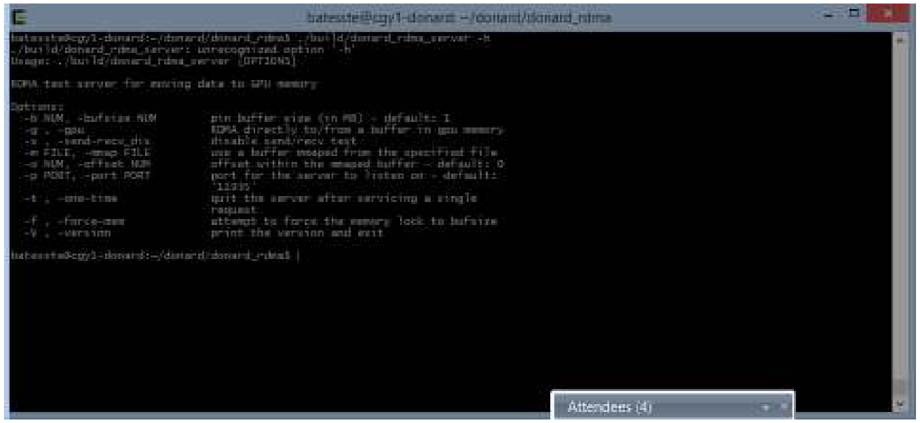
Once the new nvme_donard module is modprobe'd into the kernel a new IOCTL is available:

#define NVME_IOCTL_SUBMIT_GPU_IO _IOW('N', 0x45, struct nvme_gpu_io)

This IOCTL allows us to move data directly to/from the GPU BAR from/to any NVMe device.

Donard Codebase –donard_RDMA

- The donard_rdma repo contains code to implement both a RDMA server and client that can also perform p2p on the server side.
- Note this is not NVMe based per se because the target has to be a memory space, not a mailbox.
- Builds on top of Open Fabric to implement the hooks to the RDMA NIC.
- Validated with Chelsio iWARP NIC. Should work with other RDMA devices (iWARP, CoE, Infiniband).
- We were able to saturate the 10GbE link. Need to test at 40GbE.



```
batesste@cgy1-donard:~/donard/donard_rdma$ ./build/donard_rdma_server -h
./build/donard_rdma_server: unrecognized option '-h'
Usage: ./build/donard_rdma_server [OPTIONS]

RDMA test server for moving data to GPU memory

Options:
  -b NUM, --bufsize NUM       pin buffer size (in MB) - default: 1
  -g , --gpu                  RDMA directly to/from a buffer in gpu memory
  -s , --send-recv_dis        disable send/recv test
  -m FILE, --mmap FILE        use a buffer mmaped from the specified file
  -o NUM, --offset NUM        offset within the mmaped buffer - default: 0
  -p PORT, --port PORT        port for the server to listen on - default:
                              '11235'
  -t , --one-time             quit the server after servicing a single
                              request
  -f , --force-mem            attempt to force the memory lock to bufsize
  -V , --version              print the version and exit

batesste@cgy1-donard:~/donard/donard_rdma$ |
```

Once the new nvme_donard module is modprobe’d into the kernel a new IOCTL is available:

#define NVME_IOCTL_SUBMIT_GPU_IO _IOW('N', 0x45, struct nvme_gpu_io)

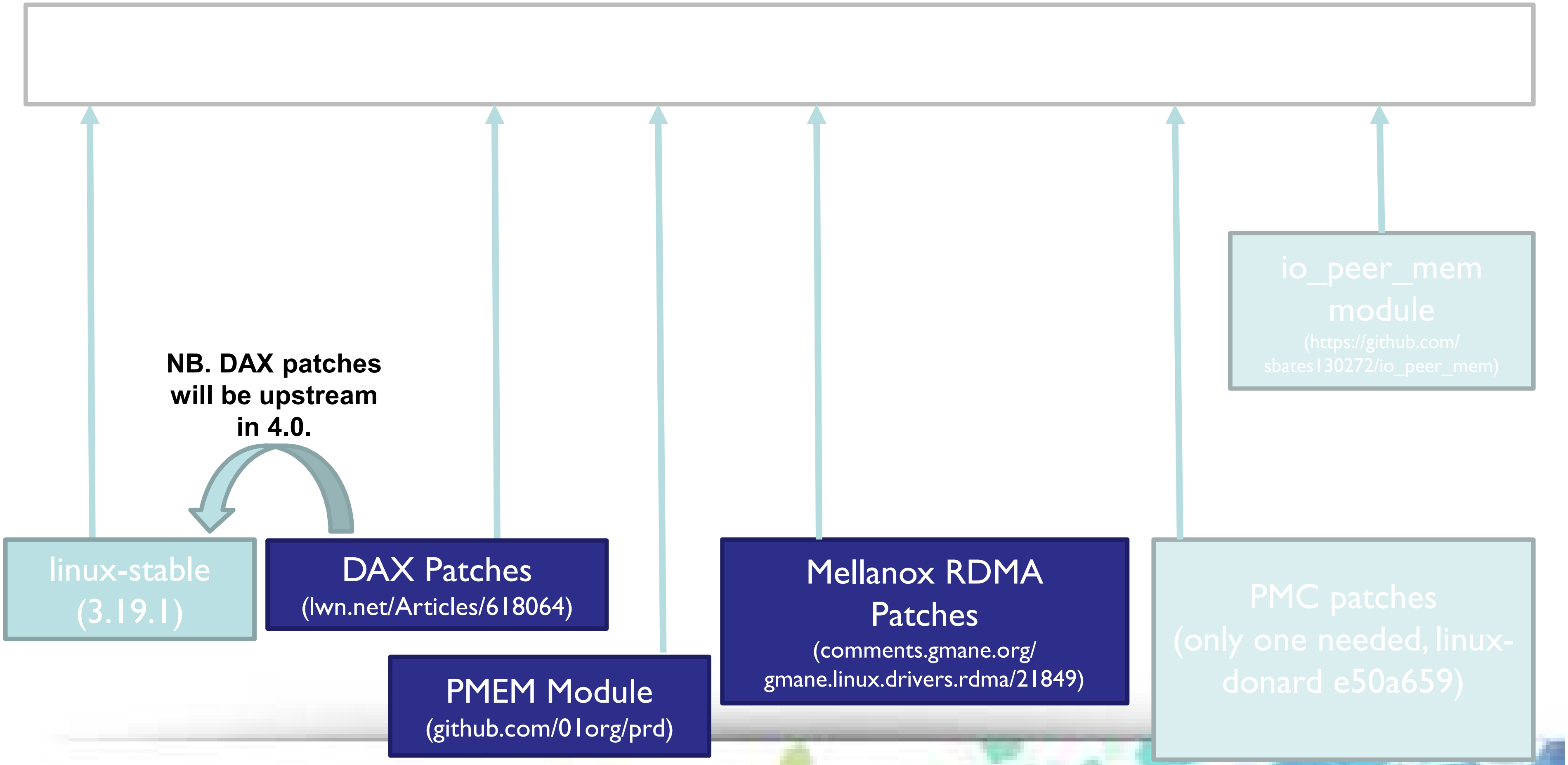
This IOCTL allows us to move data directly to/from the GPU BAR from/to any NVMe device.

Kernel: Linux-Donard

□ Upstream code

■ Third party patchset

□ PMC patchset



io_peer_mem module (https://github.com/sbates130272/io_peer_mem)
NB. DAX patches will be upstream in 4.0.
linux-stable (3.19.1)
DAX Patches (lwn.net/Articles/618064)
PMEM Module (github.com/01org/prd)
Mellanox RDMA Patches (comments.gmane.org/gmane.linux.drivers.rdma/21849)
PMC patches (only one needed, linux-donard e50a659)

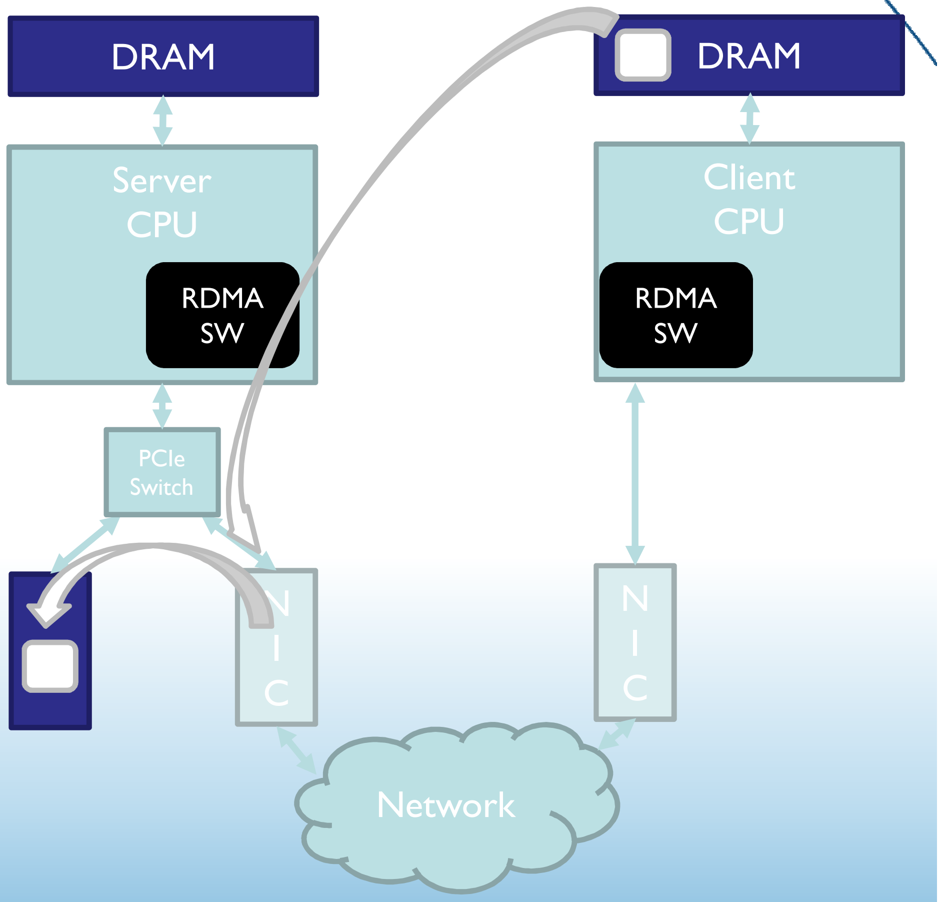
https://github.com/sbates130272/io_peer_mem/blob/master/README.md

Peer-Direct NVRAM over RDMA Fabrics

Proof of Concept

Demoed at Flash Memory Summit 2015

- Development platform to enable testing of remote memory transactions over RDMA fabrics to non-volatile storage
  - Mellanox RDMA HCA
  - PMCS NVRAM Card
  - PMCS PCIe Switch
- IO transactions bypass host CPU on server using Peer-Direct
  - Reduced server load and DRAM bandwidth
- 7us latency for 4KB IO from client to server non-volatile memory over RDMA connection
  - Network latency no longer a don't-care for remote block IO transactions

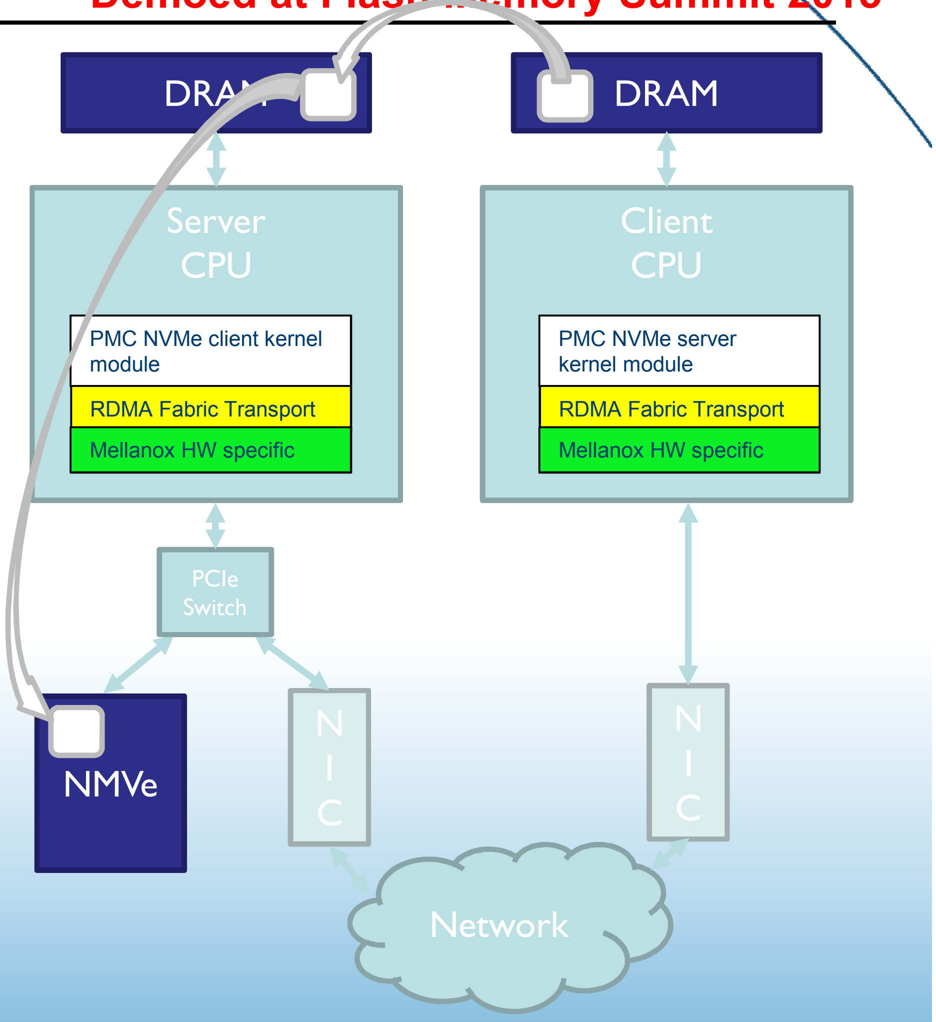


NVMe over RDMA

Proof of Concept

- Development platform to enable testing of NVMe with RDMA
  - Mellanox RDMA NIC
  - PMCS high performance NVMe device
- IO performance for remote NVMe transactions similar to local device
  - No impact to IO throughput
    - Fully utilizing RDMA bandwidth with 4K IO
  - Latency impact is currently <6us on 4KB random Read/Write
    - Further improvements expected
- Next step - Peer-Direct with PMCS PCIe switch in server
  - Further reduce latency and fully offload data plane from host CPU

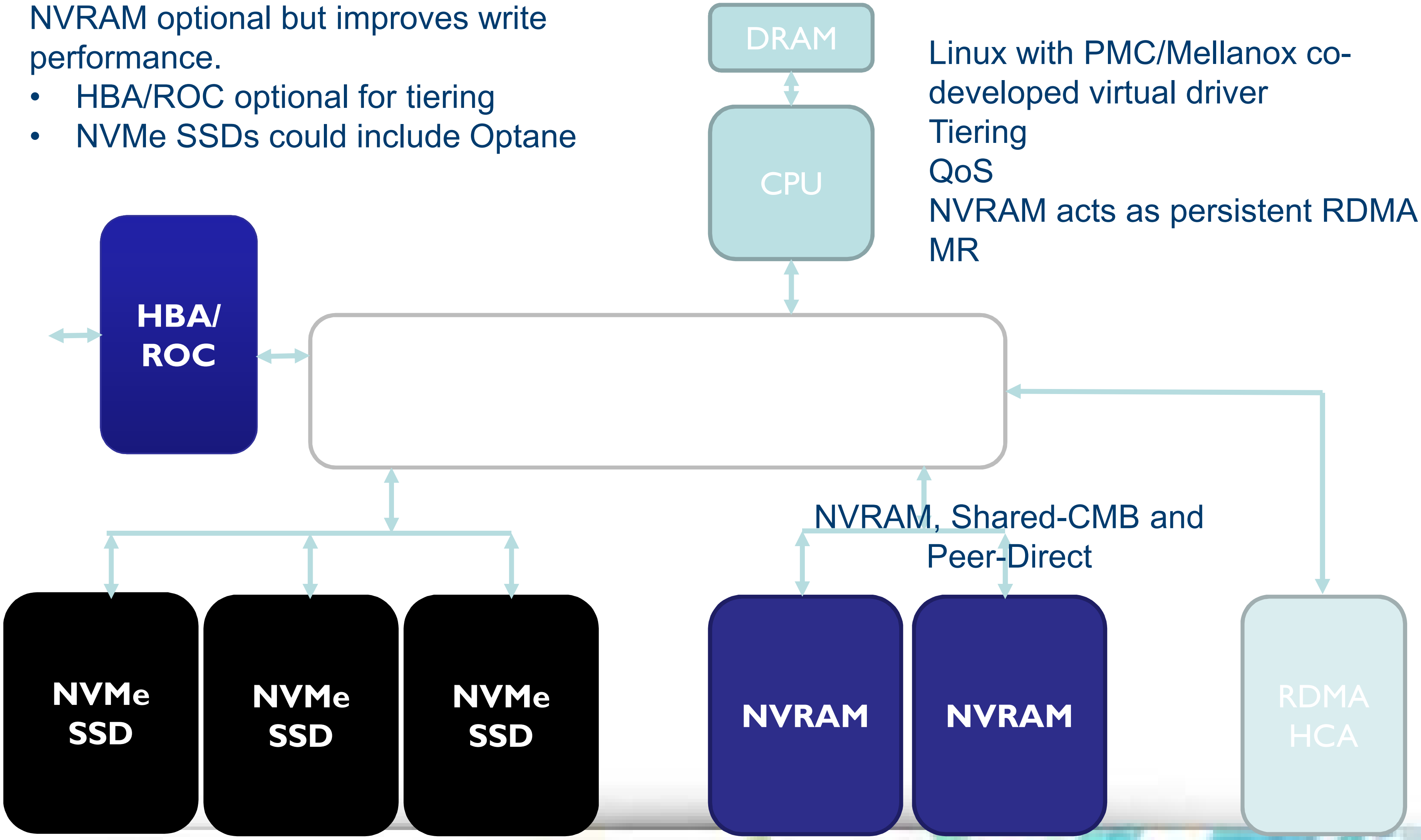
Demoed at Flash Memory Summit 2015



NVMe over RDMA Reference Design

NVRAM optional but improves write performance.

- HBA/ROC optional for tiering
- NVMe SSDs could include Optane



Linux with PMC/Mellanox co-developed virtual driver
Tiering
QoS
NVRAM acts as persistent RDMA MR

NVRAM, Shared-CMB and Peer-Direct

Conclusions

- Project Donard has developed a framework that allows PCIe devices to communicate in a Peer-2-Peer fashion.
- We have presented results for GPU<->NVMe and RDMA<->NVRAM/NVMe.
- The RDMA work ties into NVMe over Fabrics and Controller Memory Buffers.