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.

PCle Devices include
- NVMe SSDs
- RDMA capable NICs
- GPU cards
- NVMe compliant NVRAM cards
Why Donard? Goals and Objectives

<10K IOPS per CPU

~1M IOPS per flash device

Fast Data workloads requiring deep parallelism are not efficient with x86

~2M IOPS per 40G Link
The Solution: Complex Event Processing
Pre-process Fast Data Algo’s in the data path

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 Tesla K20c GPU card
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
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.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Bandwidth(^1) (GB/s)</th>
<th>DRAM Volume (GB)(^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>1.9</td>
<td>3.85</td>
</tr>
<tr>
<td>Donard (DMA)</td>
<td>2.5</td>
<td>0.56</td>
</tr>
</tbody>
</table>

\(^1\) Bandwidth was measured on our server which had a very standard PCIe fabric using a total transfer size of 1GB. 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**.

<table>
<thead>
<tr>
<th></th>
<th>Technique</th>
<th>Bandwidth (MB/s)</th>
<th>DRAM Utilization¹</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reads</strong></td>
<td>DRAM</td>
<td>1170</td>
<td>36,000,000</td>
</tr>
<tr>
<td></td>
<td>GPU</td>
<td>800</td>
<td>4,000,000</td>
</tr>
<tr>
<td><strong>Writes</strong></td>
<td>DRAM</td>
<td>1170</td>
<td>34,000,000</td>
</tr>
<tr>
<td></td>
<td><strong>Donard</strong></td>
<td>1170</td>
<td>250,000</td>
</tr>
<tr>
<td></td>
<td>GPU</td>
<td>1170</td>
<td>4,000,000</td>
</tr>
</tbody>
</table>

¹ Measured DRAM accesses using likwid-perfctr

---

2015 Storage Developer Conference. © PMC. All Rights Reserved.
DONARD Use-Case: Image Search

<table>
<thead>
<tr>
<th>Technique</th>
<th>Bandwidth (GB/s)</th>
<th>DRAM Utilization$^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reads</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classical</td>
<td>1.90</td>
<td>5230</td>
</tr>
<tr>
<td>Donard</td>
<td>2.50</td>
<td>1</td>
</tr>
<tr>
<td><strong>Writes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classical</td>
<td>1.51</td>
<td>6012</td>
</tr>
<tr>
<td>Donard</td>
<td>0.65</td>
<td>1</td>
</tr>
</tbody>
</table>

- 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.

<table>
<thead>
<tr>
<th></th>
<th>HDD</th>
<th>SDD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mpix/s</td>
<td>Mpix/s</td>
</tr>
<tr>
<td>CPU</td>
<td>77.0</td>
<td>122.8</td>
</tr>
<tr>
<td>CUDA$^1$</td>
<td>95.1</td>
<td>312.5</td>
</tr>
<tr>
<td><strong>DONARD</strong></td>
<td>N/A</td>
<td><strong>534.2</strong></td>
</tr>
</tbody>
</table>

$^1$ 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.

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

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.
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.

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.

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

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)

io_peer_mem module (https://github.com/sbates130272/io_peer_mem)

PMC patches (only one needed, linux-donard e50a659)
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
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

SDC 15

2015 Storage Developer Conference. © PMC. All Rights Reserved.
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.