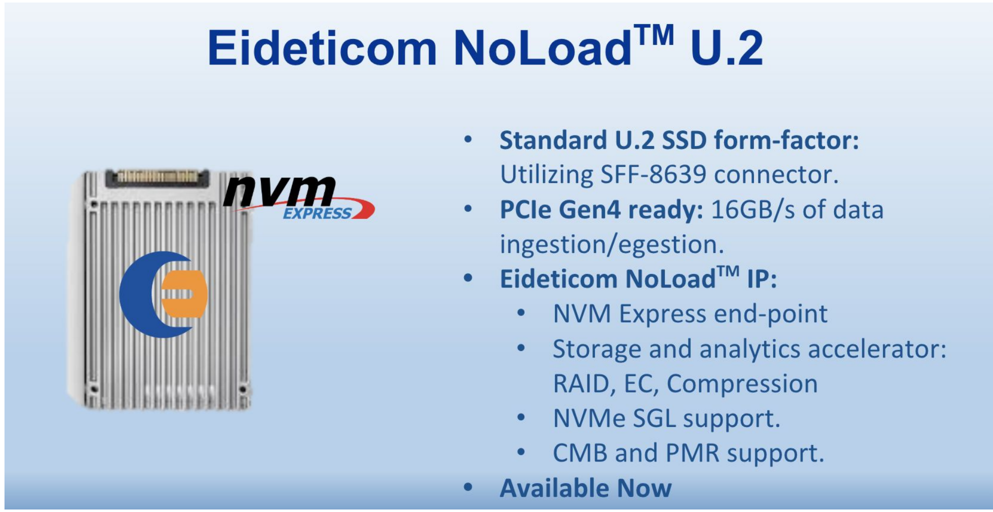
p2pdma: why, how, what?

Stephen Bates, Eideticom

p2pdma: why?

- PCIe devices getting very fast. DMAing all that traffic via system memory can be inefficient.
- PCIe devices are being populated with more and more memory (GPUs, NVMe SSDs etc).
- Existing schemes like GPUDirect are not upstreamable.
- Standards like NVMe allow devices to inform host of memory properties in a standards based way.



Eideticom NoLoad™ U.2

- **Standard U.2 SSD form-factor:** Utilizing SFF-8639 connector.
- **PCIe Gen4 ready:** 16GB/s of data ingestion/egestion.
- **Eideticom NoLoad™ IP:**
  - NVM Express end-point
  - Storage and analytics accelerator: RAID, EC, Compression
  - NVMe SGL support.
  - CMB and PMR support.
- **Available Now**

nvm EXPRESS

- High speed (PCIe Gen4 x4)
- NVMe CMB is a standards based BAR.
- Easily fit 24 in a 2U server.

p2pdma: how?

The latest p2pdma kernel patches are applied since 4.20-rc1 [1]. Note that we use an additional patch [2] which is not going upstream to expose p2pdma to userspace (we can discuss that more now). The p2pdma stuff only works on Intel and AMD but with additional hacky patches we can get it up on ARM64 [3] and are working to enable it on RISC-V too. We also have a simple p2p copy application which we use a lot for debug and performance testing [4] and fio also has support for using p2pdma memory via the iomap flag [5]. We don't have a QEMU model for NoLoad but we do use the NVMe model in QEMU for p2pdma testing. We added support for NVMe models with CMBs a while back to the upstream QEMU [6]. Finally I try and maintain a script that always builds the latest p2pdma kernel with sane .config options for x86_64 [7].

[1] https://lkml.org/lkml/2018/9/13/54

[2] https://github.com/sbates130272/linux-p2pmem/commit/9a5eccff0781f455ac6b2b146007f93c480166ff

[3] https://github.com/sbates130272/linux-p2pmem/commits/pci-p2p-v5-plus-ioremap

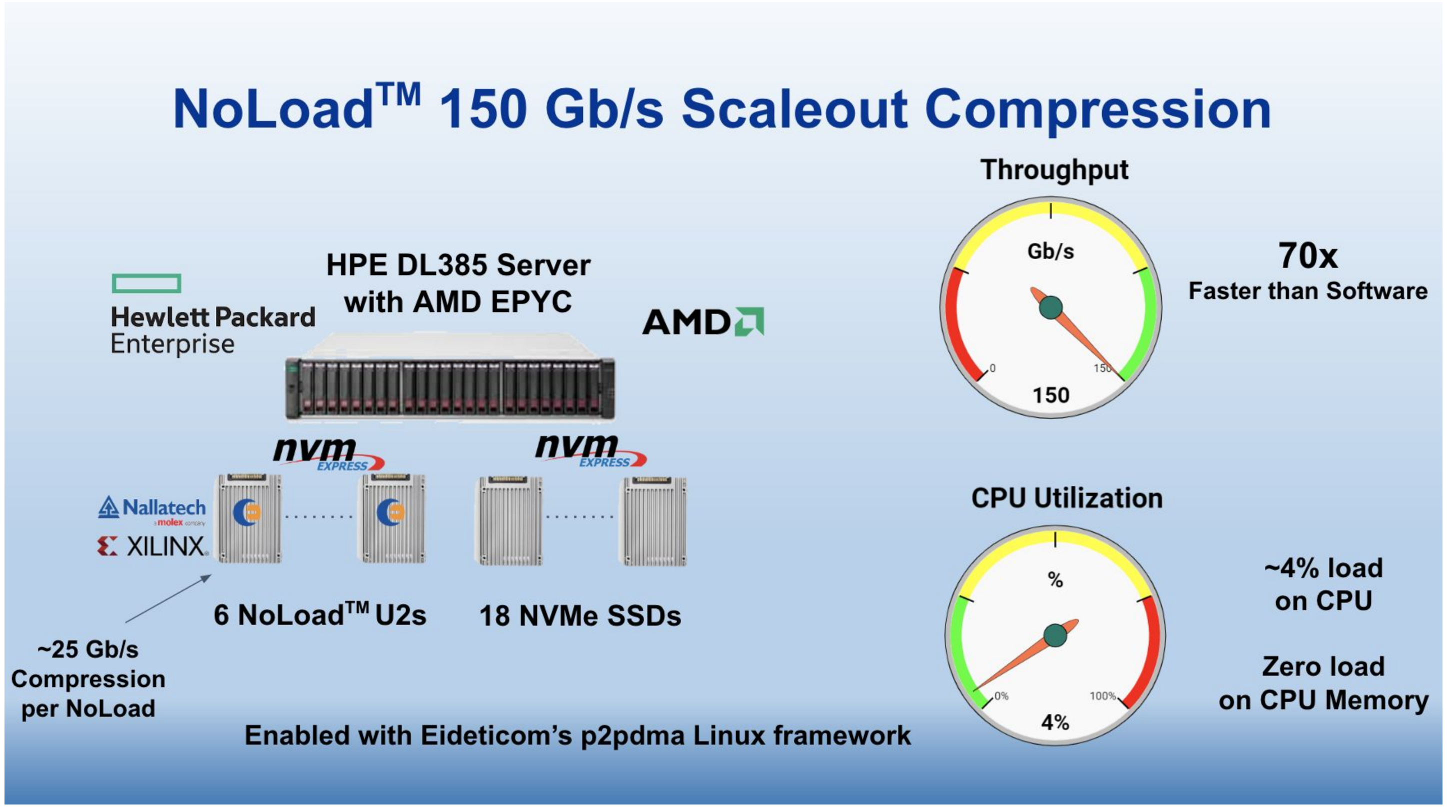
[4] https://github.com/sbates130272/p2pmem-test

[5] https://github.com/axboe/fio/blob/master/HOWTO [see iomap mapshared option]

[6] https://github.com/qemu/qemu/commit/b2b2b67a0057407e19cfa3fdd9002db21ced8b01

[7] https://github.com/sbates130272/kernel-tools/blob/master/build-latest-p2pdma-kernel [pull the whole kernel-tools repo and run this script, only supports Debian and derivates right now].

p2pdma: what (can we do with it)?


NoLoad™ 150 Gb/s Scaleout Compression
HPE DL385 Server with AMD EPYC
Hewlett Packard Enterprise
AMD
nvm EXPRESS
nvm EXPRESS
Nallatech
XILINX
6 NoLoad™ U2s
18 NVMe SSDs
~25 Gb/s Compression per NoLoad
Enabled with Eideticom's p2pdma Linux framework
Throughput
Gb/s
0
150
150
70x
Faster than Software
CPU Utilization
%
0%
100%
4%
~4% load on CPU
Zero load on CPU Memory

p2pdma: what next?

- How do we expose p2pdma memory to userspace?
  - HMM?
  - /dev/p2pmemX?
  - System calls?
- How do we augment p2pdma to accomodate other PCI resources?
  - Doorbells?
  - Config space?
  - NVMe-oF target offload?
- How we enable other interesting use-copies?
  - NVMe to NVMe copies
  - NVMe to/from GPGPU
- Can GPUDirect leverage this to become legitimate?