RDMA GPU Direct for ioMemory

GPU Technology Conference: S4265
Abstract

- Learn how to eliminate I/O bottlenecks by integrating Fusion-io ioMemory Flash storage into your GPU apps

1. Technical overview of Fusion-io’s PCIe attached ioMemory

2. Best practices and tuning for GPU applications using ioMemory based flash storage
   - Topics will cover threading, pipe-lining, and data path acceleration via RDMA GPU Direct
   - Sample code showing integration between RDMA GPU Direct and Fusion-io ioMemory will be presented
Outline

- Fusion-io ioMemory
- CUDA
  - Programming patterns
- GPU Direct
- RDMA GPU Direct for ioMemory
  - API
  - Sample code
  - Performance
- Conclusion
Fusion-io Introduction

Fusion-io delivers the world’s data faster. From workstations to datacenter servers, Fusion-io accelerates innovative leaders like:

Facebook  
Salesforce  
Xbox Live

and over 4000 other companies worldwide.
ioMemory Accelerates Applications

- Relational Databases: ORACLE, MySQL, Sybase, INGRES, DB2, PostgreSQL, INFORMIX
- Server Virtualization: VMware, Windows Server, Hyper-V, Citrix
- VDI: VMware, Citrix
- Big Data: Apache HBase, Cassandra, MongoDB, Oracle
- Search: fastAutonomy, Lucene, Oracle Text
- Analytics: AccessData, MarkLogic, LexisNexis
- HPC: FLUENT, ANSYS, TIBCO, IBM, Schlumberger, Lustre, IBM GPFS
- Messaging: IBM MQ, Autodesk, Adobe
- Workstation: Microsoft Exchange, SharePoint 2010, IBM Lotus
- Collaboration: ArcSight, Splunk, VARNISH
- Caching: LAMP, .NET
- Security/Logging: LAMP, .NET
Flash Results Are Dramatic

<table>
<thead>
<tr>
<th>FINANCIALS</th>
<th>WEB</th>
<th>TECH</th>
<th>RETAIL</th>
<th>GOV / MFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>NYSE Euronext</td>
<td>facebook</td>
<td>salesforce.com</td>
<td>alibris</td>
<td>Lawrence Livermore National Laboratory</td>
</tr>
<tr>
<td>Callcredit Information Group</td>
<td>lockerz</td>
<td>SevOne</td>
<td>Rakuten</td>
<td></td>
</tr>
<tr>
<td>First Rate</td>
<td>myspace</td>
<td>PIPKINS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TransUnion</td>
<td>ning</td>
<td>Cloudmark</td>
<td></td>
<td></td>
</tr>
<tr>
<td>craigslist</td>
<td>Coudmark</td>
<td>zenceeded</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kontera</td>
<td>Host Europe</td>
<td>MCX</td>
<td>cyberagent</td>
<td></td>
</tr>
<tr>
<td>dwango</td>
<td>Mixi</td>
<td>Pandora</td>
<td></td>
<td></td>
</tr>
<tr>
<td>skyscanner</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CREDIT SUISSE</th>
<th>Answers.com</th>
<th>datalogix</th>
<th>EQUINOX</th>
<th>tabulex</th>
</tr>
</thead>
<tbody>
<tr>
<td>5x FASTER DATA ANALYSIS</td>
<td>30x FASTER DATABASE REPLICATION</td>
<td>40x FASTER DATA WAREHOUSE QUERIES</td>
<td>15x QUERY PROCESSING THROUGHPUT</td>
<td>15x FASTER QUERIES</td>
</tr>
</tbody>
</table>

60+ case studies at http://fusionio.com/casestudies
ioMemory Purpose Built Products

**ENTERPRISE**
- ioDrive™
- SCALE UP

**HYPERSCALE**
- ioScale™
- SCALE OUT

**WORKSTATION**
- ioFX™
- SINGLE USER
Up to 3.0TB of capacity per PCIe slot

Up to 2.4TB of capacity per x8 PCIe slot

Up to 3.2TB of capacity per PCIe slot

Up to 1.6TB of workstation acceleration
Access Delay Time

Server/Workstation Based Flash

- Significantly faster than disk
- More scalable than memory
- Less expensive than memory
- Data is persistent like on disk

CPU Cache
DRAM

Disk Drives & SSDs

Nanoseconds $10^{-9}$
Microseconds $10^{-6}$
Milliseconds $10^{-3}$
Cut-Through Architecture

Generic SSD Disk Approach

Fusion Cut-Through Approach

Host CPU
PCIe
SAS
DRAM
RAID Controller
Super Capacitors
NAND
SSD

Host CPU
PCIe
DRAM
App
OS
Fusion Cut-Through Architecture

Data path Controller
High Bandwidth Scalability

ioMemory bandwidth scales linearly while SSDs and PCIe SSDs max out at around 2GB/s (when adding multiple devices)

Example: Eight ioMemory devices in RAID0 provide a bandwidth of 12GB/s in a single system!
Latency

With an average response time of 50µs ioMemory will service 20 I/O’s for every 1ms of disk access latency!
Consistent Application Performance

ioMemory balances read/write performance for consistent throughput

Queuing behind slow writes causes SSD latency spikes
ioDrive2 – ioMemory for Enterprise

Pushing Performance Density Envelope with ioMemory Platform

- Combines VSL and ioMemory into an ioMemory platform that takes enterprise applications and databases to the next level
  - Consistent Low Latency Performance
  - Industry-leading Capacity
  - Lowest CPU Utilization Per Unit of Application Work
  - Unrivaled Endurance
  - Enterprise Reliability via Self-Healing, Wear Management, and Predictive Monitoring
  - Extensive OS Support and Flexibility
ioDrive2 Duo – Heaviest Workloads

> Based on the same ioMemory platform as the ioDrive2
> The ioDrive2 Duo provides 2.4TB of ioMemory per x4 and x8 PCI Express slot

Organizations can:

- Handle twice the transactions, queries and requests
- Add capacity for future growth
- Ensure applications seamlessly handle larger traffic spikes
- Consolidate more infrastructure
Drivers for Hyperscale Data Centers

<table>
<thead>
<tr>
<th>MARKET SEGMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Large-scale Transaction Processing</td>
</tr>
<tr>
<td>▶ Content Distribution On-Demand Streaming</td>
</tr>
<tr>
<td>▶ Big Data Management and Meta Data Indexing</td>
</tr>
<tr>
<td>▶ Cloud Computing and Multi-tenancy</td>
</tr>
<tr>
<td>▶ Business Intelligence: Data Warehousing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KEY DRIVERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Increase Transaction Throughput</td>
</tr>
<tr>
<td>▶ Lower Latency and Command Completion Times</td>
</tr>
<tr>
<td>▶ Lower Flash TCO</td>
</tr>
</tbody>
</table>

Source: HGST Flash Memory Summit Aug 2012
ioScale – ioMemory for Hyperscale

Purpose-built for the Largest Web Scale and Cloud Environments

▸ Consistent low latency and high performance
▸ Reliability through Simplicity
▸ Unrivaled endurance
▸ Tools for Flash-Smart Applications
▸ Industry-Leading performance density
  • 410GB, 825GB, 1650GB and 3.2TB
ioFX – ioMemory for Workstations

Bridging the gap between creative potential and application performance

▸ Tuned for sustained performance in multithreaded applications
▸ Work on 2K, 4K and 5K content interactively, in full resolution
▸ Manipulate stereoscopic content in real-time
▸ Accelerate video and image editing and compositing
▸ Speed video playback
▸ Powerful throughput to maximize GPU processing
▸ Simplify and accelerate encoding and transcoding as well as other data-intensive activities in contemporary digital production
## ioDrive2 Specifications

<table>
<thead>
<tr>
<th>ioDrive2 Capacity</th>
<th>400GB SLC</th>
<th>600GB SLC</th>
<th>365GB MLC</th>
<th>785GB MLC*</th>
<th>1.2TB MLC*</th>
<th>3.0TB MLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Bandwidth - 1MB</td>
<td>1.4 GB/s</td>
<td>1.5 GB/s</td>
<td>910 MB/s</td>
<td>1.5 GB/s</td>
<td>1.5 GB/s</td>
<td>1.5 GB/s</td>
</tr>
<tr>
<td>Write Bandwidth - 1MB</td>
<td>1.3 GB/s</td>
<td>1.3 GB/s</td>
<td>590 MB/s</td>
<td>1.1 GB/s</td>
<td>1.3 GB/s</td>
<td>1.3 GB/s</td>
</tr>
<tr>
<td>Ran. Read IOPS - 512B</td>
<td>360,000</td>
<td>365,000</td>
<td>137,000</td>
<td>270,000</td>
<td>275,000</td>
<td>143,000</td>
</tr>
<tr>
<td>Ran. Write IOPS - 512B</td>
<td>800,000</td>
<td>800,000</td>
<td>535,000</td>
<td>800,000</td>
<td>800,000</td>
<td>535,000</td>
</tr>
<tr>
<td>Ran. Read IOPS - 4K</td>
<td>270,000</td>
<td>290,000</td>
<td>110,000</td>
<td>215,000</td>
<td>245,000</td>
<td>136,000</td>
</tr>
<tr>
<td>Ran. Write IOPS - 4K</td>
<td>270,000</td>
<td>270,000</td>
<td>140,000</td>
<td>230,000</td>
<td>250,000</td>
<td>242,000</td>
</tr>
<tr>
<td>Read Access Latency</td>
<td>47μs</td>
<td>47μs</td>
<td>68μs</td>
<td>68μs</td>
<td>68μs</td>
<td>68μs</td>
</tr>
<tr>
<td>Write Access Latency</td>
<td>15μs</td>
<td>15μs</td>
<td>15μs</td>
<td>15μs</td>
<td>15μs</td>
<td>15μs</td>
</tr>
</tbody>
</table>

Bus Interface: PCI Express 2.0

- **Weight**: 6.6 ounces (9.5 ounces)
- **Form Factor**: HH x HL (FH x HL)
- **Warranty**: 5 years or maximum endurance used

### Supported Operating Systems

- **Linux**: RHEL 5/6; SLES 10/11; OEL 5/6; CentOS 5/6; Debian Squeeze; Fedora 16/17; openSUSE 12; Ubuntu 10/11/12
- **UNIX**: Solaris 10/11 x64; OpenSolaris 2009.06 x64; OSX 10.6/10.7/10.8
- **Hypervisors**: VMware ESX 4.0/4.1/ESXi 4.1/5.0/5.1, Windows 2008 R2 with Hyper-V, Hyper-V Server 2008 R2

[www.fusionio.com](http://www.fusionio.com) August 2013
# ioDrive2 Duo Specifications

<table>
<thead>
<tr>
<th>ioDrive2 Duo Capacity</th>
<th>2.4TB MLC*</th>
<th>1.2TB SLC*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Bandwidth (1 MB)</td>
<td>3.0 GB/s</td>
<td>3.0 GB/s</td>
</tr>
<tr>
<td>Write Bandwidth (1 MB)</td>
<td>2.5 GB/s</td>
<td>2.5 GB/s</td>
</tr>
<tr>
<td>Ran. Read IOPS (512B)</td>
<td>540,000</td>
<td>700,000</td>
</tr>
<tr>
<td>Ran. Write IOPS (512B)</td>
<td>1,100,000</td>
<td>1,100,000</td>
</tr>
<tr>
<td>Ran. Read IOPS (4K)</td>
<td>480,000</td>
<td>580,000</td>
</tr>
<tr>
<td>Ran. Write IOPS (4K)</td>
<td>490,000</td>
<td>535,000</td>
</tr>
<tr>
<td>Read Access Latency</td>
<td>68µs</td>
<td>47µs</td>
</tr>
<tr>
<td>Write Access Latency</td>
<td>15µs</td>
<td>15µs</td>
</tr>
<tr>
<td>Bus Interface</td>
<td>PCI Express 2.0 x8 electrical x8 physical</td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td>Less than 11 ounces</td>
<td></td>
</tr>
<tr>
<td>Form Factor</td>
<td>Full-height, half-length</td>
<td></td>
</tr>
<tr>
<td>Warranty</td>
<td>5 years or maximum endurance used</td>
<td></td>
</tr>
</tbody>
</table>

**Supported Operating Systems**

- **Linux**: RHEL 5/6; SLES 10/11; OEL 5/6; CentOS 5/6; Debian Squeeze; Fedora 16/17; openSUSE 12; Ubuntu 10/11/12
- **UNIX**: Solaris 10/11 x64; OpenSolaris 2009.06 x64; OSX 10.6/10.7/10.8
- **Hypervisors**: VMware ESX 4.0/4.1/ESXi 4.1/5.0/5.1, Windows 2008 R2 with Hyper-V, Hyper-V Server 2008 R2

[www.fusionio.com](http://www.fusionio.com) August 2013
# ioScale Specifications

<table>
<thead>
<tr>
<th>ioScale 410GB</th>
<th>ioScale 825GB</th>
<th>ioScale 1650GB</th>
<th>ioScale 3.2TB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Factor</td>
<td>HHxHL</td>
<td>HHxHL</td>
<td>FHxHL</td>
</tr>
<tr>
<td>Read Bandwidth (1MB)</td>
<td>1.4 GB/s</td>
<td>1.4 GB/s</td>
<td>1.4 GB/s</td>
</tr>
<tr>
<td>Write Bandwidth (1MB)</td>
<td>700 MB/s</td>
<td>1.1 GB/s</td>
<td>1.1 GB/s</td>
</tr>
<tr>
<td>Random Read IOPS (512 bytes)</td>
<td>78K</td>
<td>135K</td>
<td>144K</td>
</tr>
<tr>
<td>Random Write IOPS (512 bytes)</td>
<td>535K</td>
<td>535K</td>
<td>535K</td>
</tr>
<tr>
<td>Read Access Latency</td>
<td>77µs</td>
<td>77µs</td>
<td>77µs</td>
</tr>
<tr>
<td>Write Access Latency</td>
<td>19µs</td>
<td>19µs</td>
<td>19µs</td>
</tr>
<tr>
<td>Power Profile</td>
<td>&lt; 25W</td>
<td>&lt; 25W</td>
<td>&lt; 25W</td>
</tr>
<tr>
<td>PBW (typical)</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

**Supported Operating Systems**

- **Linux**: RHEL 5/6; SLES 10/11; OEL 5/6; CentOS 5/6; Debian Squeeze; Fedora 16/17; openSUSE 12; Ubuntu 10/11/12
- **UNIX**: Solaris 10/11 x64; OpenSolaris 2009.06 x64; OSX 10.6/10.7/10.8
- **Hypervisors**: VMware ESX 4.0/4.1/ESXi 4.1/5.0/5.1, Windows 2008 R2 with Hyper-V, Hyper-V Server 2008 R2
## ioFX Specifications

<table>
<thead>
<tr>
<th>ioFX Capacity</th>
<th>420GB</th>
<th>1650GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND Type</td>
<td>MLC (Multi Level Cell)</td>
<td></td>
</tr>
<tr>
<td>Read Bandwidth (1MB)</td>
<td>1.4 GB/s</td>
<td>1.4GB/s</td>
</tr>
<tr>
<td>Write Bandwidth (1MB)</td>
<td>700MB/s</td>
<td>1.1GB/s</td>
</tr>
<tr>
<td>Read Access Latency (4K)</td>
<td>77µs</td>
<td>77µs</td>
</tr>
<tr>
<td>Write Access Latency (4K)</td>
<td>19µs</td>
<td>19µs</td>
</tr>
<tr>
<td>Bus Interface</td>
<td>PCI Express 2.0 x4 (x8 physical)</td>
<td>PCI Express 2.0 x4 (x4 physical)</td>
</tr>
<tr>
<td>Bundled Mgmt Software</td>
<td>ioSphere</td>
<td></td>
</tr>
<tr>
<td>Operating Systems</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Microsoft Windows</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Linux</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OSX</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hypervisors</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warranty</td>
<td>3 years or maximum endurance used</td>
<td></td>
</tr>
</tbody>
</table>

### Operating Systems

- **Linux**: RHEL 5/6, SLES 10/11, CentOS 5/6, Debian Squeeze, Fedora 16/17, Ubuntu 10/11/12, OELv5/6, OpenSUSE 12
- **OSX**: MAC OSX 10.6.7/10.7/10.8
- **Hypervisors**: VMware ESX 4.0/4.1, VMware ESXi 4.1/5.0/5.1, Windows 2008 R2 with Hyper-V, Hyper-V Server 2008 R2

[www.fusionio.com](http://www.fusionio.com) August 2013
Datasheets

http://www.fusionio.com/
CUDA and the ioMemory VSL

Applications/Databases

File System

Kernel

Virtual Storage Layer (VSL)

ioMemory

Host

DRAM / Memory / Operating System and Application Memory

ioMemory Virtualization Tables

CPU and cores

Virtual Storage Layer (VSL)

Commands

ioDrive

ioMemory Data-Path Controller

Banks

Channels Wide

Host

DATA TRANSFERS

PCIe

Commands

Applications/Databases

File System

Kernel

Virtual Storage Layer (VSL)

ioMemory
ioMemory -> GPU

- **OS-pinned CUDA buffer**
  
  ```c
  // Alloc OS-pinned memory
  cudaHostAlloc((void**)&h_odata, memSize,
  (wc) ? cudaHostAllocWriteCombined : 0);
  ```

- **Read from ioMemory**
  ```c
  fd = open("/mnt/ioMemoryFile", O_RDWR | O_DIRECT);
  rc = read(fd, h_odata, memSize);
  ```

- **Copy (DMA) to GPU**
  ```c
  cudaMemcpyAsync(d_idata, h_odata, memSize,
  cudaMemcpyHostToDevice, stream);
  ```
### ioMemory -> GPU

<table>
<thead>
<tr>
<th>File System</th>
<th>CUDA RT / Driver API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>Kernel</td>
</tr>
<tr>
<td>Virtual Storage Layer</td>
<td>Nvidia.ko</td>
</tr>
</tbody>
</table>

**Application**

- **READ**
- **WRITE**

- DMA
- DMA
GPU -> ioMemory

- OS-pinned CUDA buffer
  ```c
  // Alloc OS-pinned memory
  cudaHostAlloc((void**)&h_odata, memSize,
                 (wc) ? cudaHostAllocWriteCombined : 0);
  ```

- Copy (DMA) from GPU
  ```c
  cudaMemcpyAsync(h_odata, d_idata, memSize,
                  cudaMemcpyDeviceToHost, stream);
  ```

- Write to ioMemory
  ```c
  fd = open("/mnt/ioMemoryFile", O_RDWR | O_DIRECT);
  rc = write(fd, h_odata, memSize);
  ```
Programming Patterns

- Pipelines
  - CPU threads
  - CUDA streams
- Ring buffers
  - Parallel DMA
- Direct I/O
GPU Direct
GPU Direct
https://developer.nvidia.com/gpudirect

- Copy data directly to/from CUDA pinned host memory
  - Avoid one copy but not a host memory bypass
  - Currently supported by ioDrive (as described previously)

- Peer to peer transfers
  - P2P DMA
    - between PCIe GPUs

- Peer to peer memory access between GPUs
  - NUMA
    - between CUDA kernels
RDMA GPU Direct
http://docs.nvidia.com/cuda/gpudirect-rdma/index.html

CPU

PCI Express

Third Party Device

GPUDirect RDMA

GPU
RDMA GPU Direct

- Available in Kepler-class GPUs and CUDA 5.0+
- Fastest possible inter-device communication
  - Utilizes PCIe peer-to-peer DMA transfer
- Avoids host memory altogether
  - No bounce buffer nor copy-in/copy-out
- Third party device can directly access GPU memory
  - Tech preview (proof of concept) for ioMemory
TECH PREVIEW

RDMA GPU Direct for ioMemory
ioMemory <-> GPU

Application

File System
- Kernel
- Virtual Storage Layer

CUDA RT / Driver API
- Kernel
- Nvidia.ko

nvidia_get/put_p2p_pages()
Design Considerations

- **Supported systems**
  - Kepler-class GPUs
  - Intel Sandy/Ivy Bridge

- **User space tokens**
  - Query CUDA and pass to VSL

- **Lazy unpinning**

- **Kernel driver linkage**
  - VSL uses nvidia_get/put_pages() APIs
  - Adopt the Mellanox GPU Direct approach
CUDA API

- Data structure
  typedef struct CUDA_POINTER_ATTRIBUTE_P2P_TOKENS_st {
    unsigned long long p2pToken;
    unsigned int vaSpaceToken;
  } CUDA_POINTER_ATTRIBUTE_P2P_TOKENS;

- Function
  CUresult CUDAAPI cuPointerGetAttribute(void *data,
  CUpointer_attribute attribute, CUdeviceptr pointer);

- Usage
  ptr = cudaMalloc();
  cuPointerGetAttribute(&tokens,
    CU_POINTER_ATTRIBUTE_P2P_TOKENS, ptr);
VSL user space API

typedef struct fusion_iovec {
    fio_block_range_t iov_range; ///< Range of sectors
    uint32_t iov_op;            ///< FIOV_TRIM/WRITE/READ
    uint32_t iov_flags;         ///< FIOV_FLAGS_* (GPU)
    uint64_t iov_base;          ///< Mem addr (CPU or GPU)
    ////< Special tokens for p2p GPU transfers
    unsigned long long iov_p2p_token;
    unsigned int iov_va_space_token;
    unsigned int iov_padding_reserved;
} fusion_iovec_t;

int ufct_iovec_transfer(int fd,
                        const struct fusion_iovec *iov,
                        int iovcnt, uint32_t *sectors_transferred);
Sample Code

cuda_err = cudaMalloc((void**)&buf, allocate_size);
cu_err = cuPointerGetAttribute(&cuda_tokens,
   CU_POINTER_ATTRIBUTE_P2P_TOKENS,
   (Cudeviceptr)buf);

iov[0].iov_range.base     = sector;  // ioMemory block addr
iov[0].iov_range.length   = individual_buf_size;
iov[0].iov_op             = is_write  ? FIOV_WRITE : FIOV_READ;
iov[0].iov_flags          = is_gpu    ? FIOV_FLAGS_GPU : 0;
iov[0].iov_base           = buf;     // GPU memory addr
iov[0].iov_p2p_token      = cuda_tokens.p2pToken;
iov[0].iov_va_space_token = cuda_tokens.vaSpaceToken;

ufct_iovec_transfer(fd, iov, iovcnt, &sectors_transferred);
Benchmark

- System details
  - Intel Sandy Bridge
  - NVIDIA Tesla K20
- Modified CUDA bandwidth test
- Modified fio-iovec-transfer test
Modified CUDA bandwidth test

Bandwidth (GB/s) Single Thread

- HARD DRIVE
- IOMEMORY
- IOM W/ GDR
- IOM W/ GDR W/ LAZY UNPINNING

- Read (ioM->GPU)
- Write (GPU->ioM)
Modified fio-iovect-transfer test

Bandwidth (GB/s) with Multiple Threads

![Graph showing bandwidth with multiple threads](image-url)
Conclusion

- ioMemory for high performance GPU I/O
  - Available now!
- Tech preview of RDMA GPU Direct for ioMemory
  - Bypass host memory to further increase I/O performance
- ioMemory can accelerate your GPU applications
- Invite feedback
  - vbrisebois@fusionio.com
Thank You