Evaluation of A Flash Storage Filesystem on the Cray XE-6

Jay Srinivasan and Shane Canon
CUG 2013
Napa, California
May 2013
Outline

• Motivation
• Flash Characteristics and Approaches
• Testbed architecture
• Results
• Future work & Conclusions
Disk Bandwidths failing to Keep Pace

Source: IBM/Violin Memory White Paper
Disk Bandwidths failing to Keep Pace

Source: IBM/Violin Memory White Paper
Characteristics of Flash

**Good**
- Random Read Performance/IOPS
- Bandwidth
- Power

**Bad**
- Erase Cycle for Writes
- Wear/Endurance
- Grooming Cycle
Characteristics of Flash

**Good**
- Random Reads
- Performance
- Bandwidth
- Power

**Bad**
- Erase Cycle for Writes
- Wear/Endurance
- Grooming Cycle
SLC, MLC, and TLC

- **SLC** (Single-Level): 1-bit/cell, 2 states
- **MLC** (Multi-Level): 2-bits/cell, 4 states
- **TLC** (Three-Level): 3-bits/cell, 8 states

Diagram:

- **P/E Cycle**
- **SLC**: 100,000 cycles
- **5x nm MLC**: 10,000 cycles
- **3x nm MLC**: 5,000 cycles
- **2x nm MLC**: 3,000 cycles
- **3-bit per cell**: 1,000 cycles

ECC Requirements:
- 4-bit ECC
- 8-bit ECC
- 15-bit ECC
- 24-bit ECC

Source: JMicron, Western Digital, Morgan Stanley Research
Methods of Integration

On-Node
• Scalable BW
• Use as Memory
Integrated Hierarchy
• Transparent to upper layers

On-Edge/Shared
• Works in systems without local-node support
Testbed Architecture

Cray XE-6

DISK File System
- OSS
- OST

Flash File System
- OSS
- OST

Infiniband

LNET Router
Compute
Compute
Compute
Compute
Compute
Compute
Compute
Compute
Compute
Compute
We used a pool of high-performing storage along with a pool of lower-performing storage and migrated files between the two.

Both Flash and Disk filesystems are mounted on the compute/service/login nodes as “external” Lustre filesystems.

The “Migrator” program runs in the background looking for specific “checkpoint” files to move – this is done on a login node that has the Flash and Disk filesystems mounted.
**Configuration Details**

**Disk File System**
- Single Object Storage Server
- Four Object Storage Targets
- Single Dell R710
- Two LSI 8600 storage arrays.
  - (Very Small, just for testing)

**Flash File System**
- Two Object Storage Servers
- Four Object Storage Targets
- Two IBM x3650 M2
- Four Virident tachION cards
  - 400 GB of SLC-class NAND
  - ~1.1 GB/s bandwidth
  - 160k Read IO operations per second.

**Common to Both**
- 12 Node TDS System (288 cores)
- Two Lustre Network (LNET) Routers
- QDR InfiniBand Network
Benchmark code

• **IOR** is a standard parallel filesystem benchmark – we use POSIX I/O and a file-per-process

• **flashio** is a benchmark code that mimics checkpoint I/O
  
  – Computation (matrix-multiply) followed by “checkpoint” I/O (bursty, short duration)
  
  – Compute and I/O time can be tuned to ensure I/O time is a small fraction of the overall compute time
  
  – Code tracks time for compute, I/O and overall run time.
Migrator

- Flash storage is a scarce resource and cannot be used for long-term storage, or even for much longer than the duration of the job
- I/O acceleration can be explicitly requested by the user or be transparent to the user
- I/O path is complicated and some user interaction will be required to ensure it is effective.
- We use a migrator task that moves data from Flash to Disk storage automatically
  - Moves only specific “named” checkpoint files and depends on a semaphore file to determine which one to move.
Results

IOR Write Bandwidth

- IOR Write B/W on Flash and Disk Storage as a function of the number of tasks
- 4MiB transfer size, 1GiB block size, file-per-process
Results

Flash vs. Disk run times

Number of tasks

Run time (secs)

Flash Time (s)  Disk Time (s)
Results

Flash vs. Disk I/O times (Low concurrency)

- Time (secs)
- Number of tasks
- Flash Time (s) vs. Scratch Time (s)
- Data points showing increasing time as number of tasks increases.
## Cost (Enterprise Class)

<table>
<thead>
<tr>
<th>Storage</th>
<th>Bandwidth Cost ($ per GB/s)</th>
<th>Capacity Cost ($ per TB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash Storage (Enterprise PCI-e)</td>
<td>$6,000</td>
<td>$6,000</td>
</tr>
<tr>
<td>Disk Storage (Enterprise Array)</td>
<td>$22,000</td>
<td>$400</td>
</tr>
</tbody>
</table>
## Cost Comparison – Strawman Config

<table>
<thead>
<tr>
<th></th>
<th>Hybrid</th>
<th>Disk Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash Storage BW (TB/s)</td>
<td>2.25</td>
<td>-</td>
</tr>
<tr>
<td>Disk Storage BW (TB/s)</td>
<td>0.39</td>
<td>1.00</td>
</tr>
<tr>
<td>Flash Capacity (PB)</td>
<td>2.25</td>
<td>-</td>
</tr>
<tr>
<td>Disk Capacity (PB)</td>
<td>20.9</td>
<td>53.3</td>
</tr>
</tbody>
</table>

### Example Application

<table>
<thead>
<tr>
<th></th>
<th>Hybrid</th>
<th>Disk Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checkpoint Volume (TB)</td>
<td>1200</td>
<td>1200</td>
</tr>
<tr>
<td>Checkpoint Iteration (s)</td>
<td>3600</td>
<td>3600</td>
</tr>
<tr>
<td>Time for Checkpoint (s)</td>
<td>533</td>
<td>1200</td>
</tr>
<tr>
<td>Time for Compute (s)</td>
<td>3067</td>
<td>2400</td>
</tr>
<tr>
<td>Percentage of Time in I/O</td>
<td>15%</td>
<td>33%</td>
</tr>
<tr>
<td>Improvement</td>
<td>28%</td>
<td>-</td>
</tr>
</tbody>
</table>

### Cost

<table>
<thead>
<tr>
<th></th>
<th>Hybrid</th>
<th>Disk Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Cost</td>
<td>$22,100,000</td>
<td>$22,000,000</td>
</tr>
</tbody>
</table>
Beyond NAND

**Phase Change Memory**
- Changes a material to/from a amorphous/crystalline structure
- O(100ns) switching time
- 100M write cycle endurance
- Limited production at lower capacities

**Memristor**
- Resistance can be changed which stores the state
- < 100ns switching time
- O(1M) write cycle endurance
- Production pushed back beyond 2015
Future Work

• Performance at scale
  – Hundreds or thousands of (filesystem) clients
  – Larger pool of Flash storage

• Improvements to the “migrator” – allow for “job asynchronous” migration or even staging (for reads)

• Evaluate ways to make performance more predictable to users at scale – private storage pools.

• Explore ways to expose control of and manage the migrator
Conclusions

• Flash and Solid State technologies are promising ways of accelerating I/O on HPC systems today.
• I/O acceleration can be done by using a storage hierarchy, and can be achieved all along the I/O path – from the compute element to the storage unit.
• I/O acceleration is primarily a Software problem – and there are a number of ways to solve the problem.
• We believe an optimal solution should not be hidden from user input and control of its use.
Questions?