Falcon: Scaling IO Performance in Multi-SSD Volumes

Pradeep Kumar       H Howie Huang
The George Washington University
SSDs in Big Data Applications

- Recent trends advocate using many SSDs for higher throughput in:
  - Graph Analytics
  - Machine Learning
  - Key-Value stores, etc.

- New techniques are taking advantage of high random IOPS of SSDs:
  - Fine grained IOs in graph processing [FAST’17]
  - Doing random IOs in graph processing [ATC’16]
  - Range scan in WiscKey is many parallel random IOs [FAST16]

- Increasing use of batched IO interfaces such as libaio in Linux
Existing IO Model

- **Computing Threads**
  - Userspace IO Buffer
  - Application IO Threads

- **IO Stack**
  - SSD
  - Volume

- **Programming Complexity**
  - High
  - Low

- **IO Performance**
  - Low
  - High

- **Application-managed IO**
  - (1 application IO thread per-SSD)

- **Kernel-managed IO**
  - (many application IO threads)

- **Falcon**
  - (1 application IO thread)

- **(b) Kernel-managed**
  - 1 or more application IO thread per-volume

- **(a) Application-managed**
  - 1 application IO thread per-SSD
Outline

01 – Overview and Problem Statement

02 – Background and Block Layer Insufficiency

03 – Falcon Architecture

04 – Evaluation
Linux: IO Flow and IO States

**IO Phases:** Plug, Unplug, Dispatch, Completion

**IO States:** start, split, merge, wait, ready, insert, dispatch, complete
Linux: Mixes IO Batching and IO Serving Tasks

➢ Examples:
  ▪ Mixing batching with merge and tag allocation in plug phase
  ▪ Mixing classify with sort in unplug phase

➢ Root cause:
  ▪ Many tasks are tied to plug-list
  ▪ Not designed for multi-SSD volume

➢ Creates many Insufficiencies
  ▪ Lack of parallelism in IO processing
  ▪ Inefficient Merge and Sort
  ▪ Unpredictable blocking

Diagram:
- **Plug Phase** (batch, merge, tag allocation)
- **Unplug Phase** (sort, classify)
- **Dispatch Phase** (dispatch)
- Software queues
- Thread-specific plug-list
- To SCSI Layer and Drivers

Linux block layer control flow
Insufficiency #1: Lack of Parallelism

- Increasing stack latency of member SSDs
  - E.g., Stack Latency of **sda** is less than **sdb**

- Effect of sequential IO serving and round-robin dispatch
  - IOs of last drive will acquire *insert* after IOs of every other drive gets dispatched
Insufficiency #2: Inefficient Merge and Sort

➢ Stack latency is greater than device latency in 8-SSD volume

➢ Plug-list intermixes IOs destined to all member drives
  ▪ Search for a merge candidate even in unrelated IOs
  ▪ Larger sorting workload across SSDs

Stack Latency in 8-SSD volume is greater than 1-SSD system
Insufficiency #3: Unpredictable Blocking

- If tag allocation fails, the IO thread blocks waiting for a free tag.
- Uncertainty about active IO count in the pipeline:
  - Storage controller dependent
  - Compromises the IO scalability in SATA controller connected SSD volume.

(a) IO performance Scaling

(b) Tag usage in 2-SSD SATA volume
Falcon: Separates IO Batching from IO Serving Tasks

**Falcon** separates IO batching from IO serving tasks. The **Falcon IO Management Layer (FML)** performs IO batching tasks only. The **Falcon Block Layer (FBL)** performs IO serving tasks only.

The diagram illustrates the process phases:

- **Classification Phase (classify)**
- **Batching Phase (batch)**
- **Sort Phase (sort)**
- **Process Phase** (merge, tag, allocation, dispatch)

**Applications**

**Direct IO**

**VFS**

**Page Cache**

**Volume Manager Instance**

**Falcon IO Management Layer (FML)**

**Falcon Block Layer (FBL)**

**SCSI Layer and Drivers**

**SSD_1**

**SSD_m**
## Falcon: Feature Comparison with Linux

<table>
<thead>
<tr>
<th>Block Layer Features</th>
<th>Linux 1-SSD</th>
<th>Linux Multi-SSD</th>
<th>Falcon Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel Processing</td>
<td>NA</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Per-Drive Sort</td>
<td>✔</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Neighbor Merge</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Dynamic Tag Management</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

- ✔ Well-suited for multi-SSD volume
- ✗ Improvements are applicable to 1-SSD system also
Falcon IO Management Layer

- **IO batching in plug-list**
  - No processing, just batching

- **Classification**
  - Single pass operation
  - No sorting

- **Enabling parallel processing**
  - Creates Falcon threads per FBL

- **Uniform unplug criteria**
  - 32 per-SSD, 256 for 8-SSD volume
  - Lower criteria for low IO demand, and latency sensitive applications
  - See the paper for more details

```
bio
Process Next IO Request
(from Volume manager Layer)

enqueue to plug-list

unplug?

no

yes

Batching Phase

Classify plug-list to temporary per-drive queues

insert

Move per-drive queues to software queues

Spawn Falcon threads, if needed

Classification Phase
```
Falcon Block Layer

- **Sort Phase**
  - Per-Drive Sort

- **Process Phase**
  - Neighbor Merge
  - Dynamic tag allocation
  - Dispatch

- **Completion Phase**

- Able to saturate a Samsung 950 Pro 512GB NVMe SSD
  - 1375 MB/s (13% better than Linux)
Dynamic Tag Management

- Allocate a tag only if a dispatch is required
- Bio-queue keeps bio objects yet to be dispatched
- Pressure point controls the active IO count in the pipeline

(a) IO Throughput Scaling for SATA controller
(b) Tag usage in 2-SSD SATA volume
Evaluation Setup

- Falcon: 600 lines of C kernel code add
- Ubuntu 16.04 version with Kernel version 4.4.0, Blk-mq block layer
- 2 Intel Xeon CPU E5-2620 2GHz with 6 cores each
- 32GB DRAM
- 8 Samsung EVO 850 500 GB SSDs, connected using LSI SAS9300-8i HBA
- 4KB Stripe size is used by default
- Raw volume, Ext4 and XFS file systems are evaluated
- Revised FIO is used as micro-benchmark
Ext4 File Random IO

- Ext4 has file inode lock issue
- 1.77x speedup for random read
- 1.59x speedup for random write

(a) Single file read throughput

(b) Single file write throughput
Buffered Random Write

- Buffer cache has just 1 thread per volume for flushing dirty pages
- 1.39x speedup for 4-SSD volume
- 1.59x speedup for 8-SSD volume
Graph Processing

➢ G-Store[SC’16] is used
  ▪ Semi-external graph analytics engine
  ▪ Configurable number of IO threads
  ▪ Linux setup, 1 and 8 IO threads
  ▪ Falcon : 1 IO thread

➢ 8-SSD volume, XFS filesystem
➢ Kronecker graph scale 28, edge factor 16 is used

➢ BFS, kCore: High random IO
➢ Connected component (CC), PageRank (PR): Sequential IO

4.12x speedup over Linux 1 IO thread setup
1.78x speedup over Linux 8 IO thread
## IO Trace Replay

<table>
<thead>
<tr>
<th>Trace Name</th>
<th>Read (%)</th>
<th>IO size range</th>
<th>Size (GB)</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>UM-Financial1</td>
<td>23.16</td>
<td>512B - 16715KB</td>
<td>17.22</td>
<td>Online transaction processing</td>
</tr>
<tr>
<td>UM-Financial2</td>
<td>82.34</td>
<td>512B - 256.5KB</td>
<td>8.44</td>
<td>Online transaction processing</td>
</tr>
<tr>
<td>UM-Websearch1</td>
<td>99.98</td>
<td>512B - 1111KB</td>
<td>15.24</td>
<td>Web Search</td>
</tr>
<tr>
<td>UM-Websearch2</td>
<td>99.98</td>
<td>8KB - 32KB</td>
<td>65.82</td>
<td>Web Search</td>
</tr>
<tr>
<td>FIU-Home</td>
<td>1</td>
<td>512B - 512KB</td>
<td>34.58</td>
<td>Research group activities</td>
</tr>
<tr>
<td>FIU-Mail</td>
<td>8.58</td>
<td>4KB - 4KB</td>
<td>86.64</td>
<td>Mail Server</td>
</tr>
<tr>
<td>FIU-Webuser</td>
<td>10.33</td>
<td>4KB - 128KB</td>
<td>30.94</td>
<td>Web User</td>
</tr>
<tr>
<td>FIU-Web-vm</td>
<td>21.8</td>
<td>4KB - 4KB</td>
<td>54.52</td>
<td>Webmail proxy/online course management</td>
</tr>
</tbody>
</table>

### Throughput Comparison

- **Linux**
- **Falcon**

1.67x better IO throughput on average
Conclusion

➢ Separating batching from IO serving tasks is the key for IO scalability in multi-SSD volume

➢ Falcon enforces per-drive processing
   - Improves the IO stack performance
   - Parallelizes the IO serving tasks across member SSDs

➢ Falcon improves the performance by 1.69x for various applications on 8-SSD volume
➢ Falcon achieves 1.13x throughput for an NVMe SSD
Thank You

- Falcon is open source now
  - https://github.com/iHeartGraph/falcon