Falcon: Scaling IO Performance in Multi-SSD Volumes

Pradeep Kumar **H Howie Huang**

The George Washington University

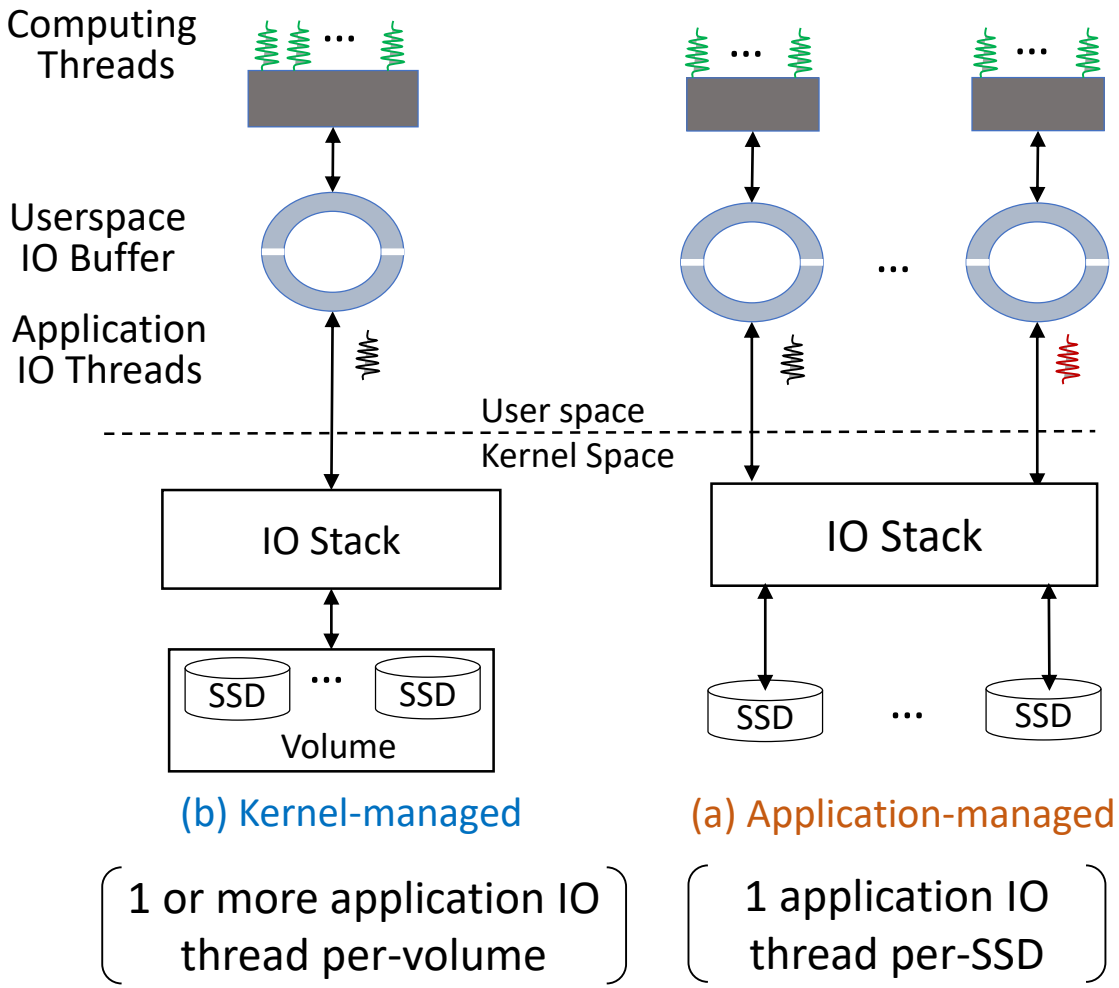
SSDs in Big Data Applications

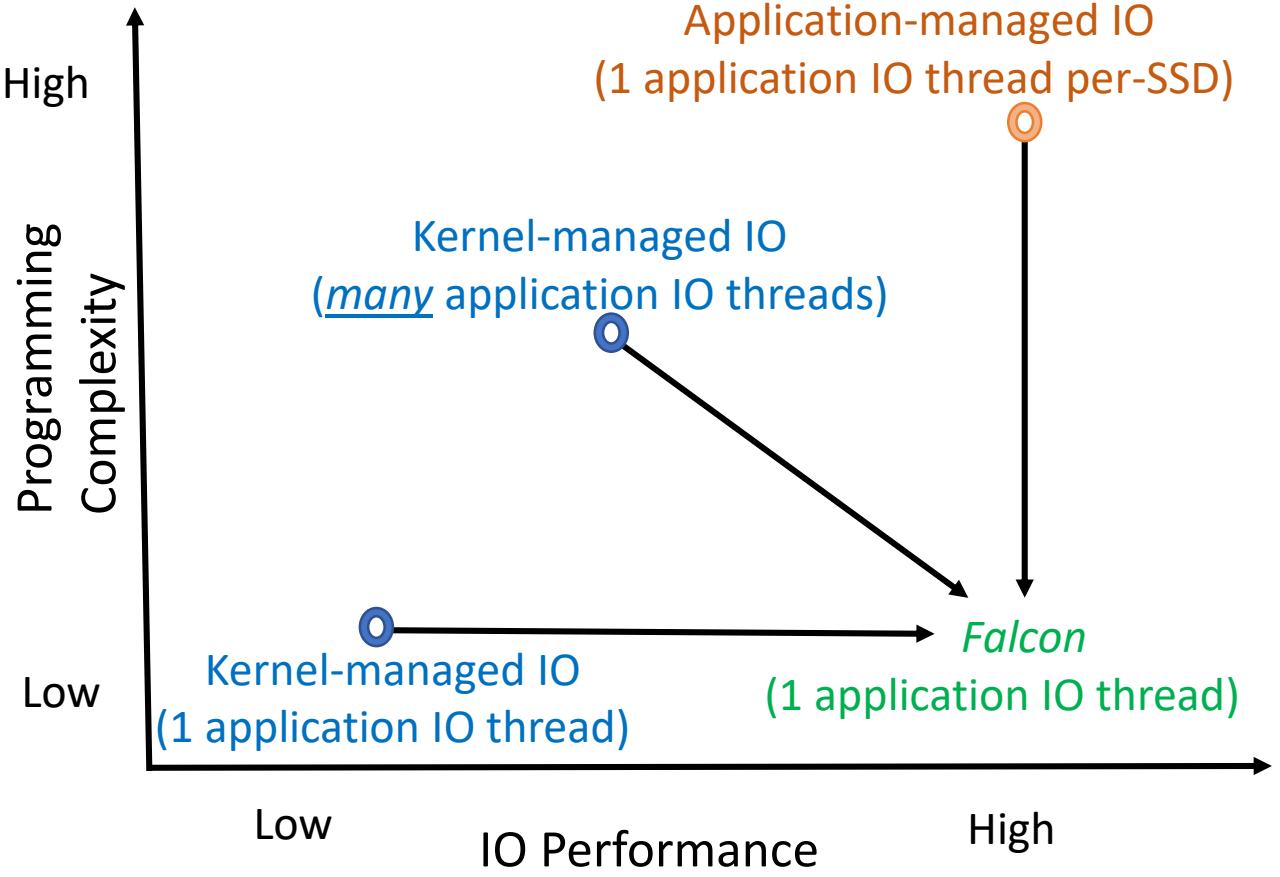
- Recent trends advocate using many SSDs for higher throughput in
  - Graph Analytics
  - Machine Learning
  - Key-Value stores, etc.

- 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]

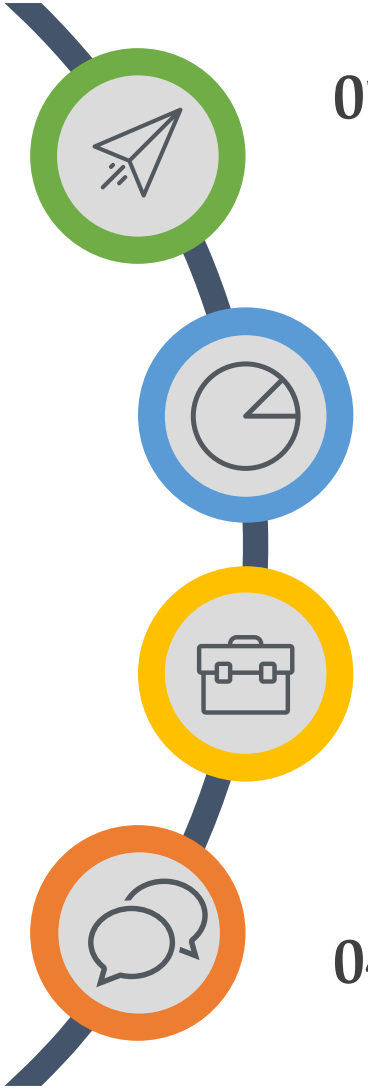
- Increasing use of batched IO interfaces such as libaio in Linux

Existing IO Model





Outline



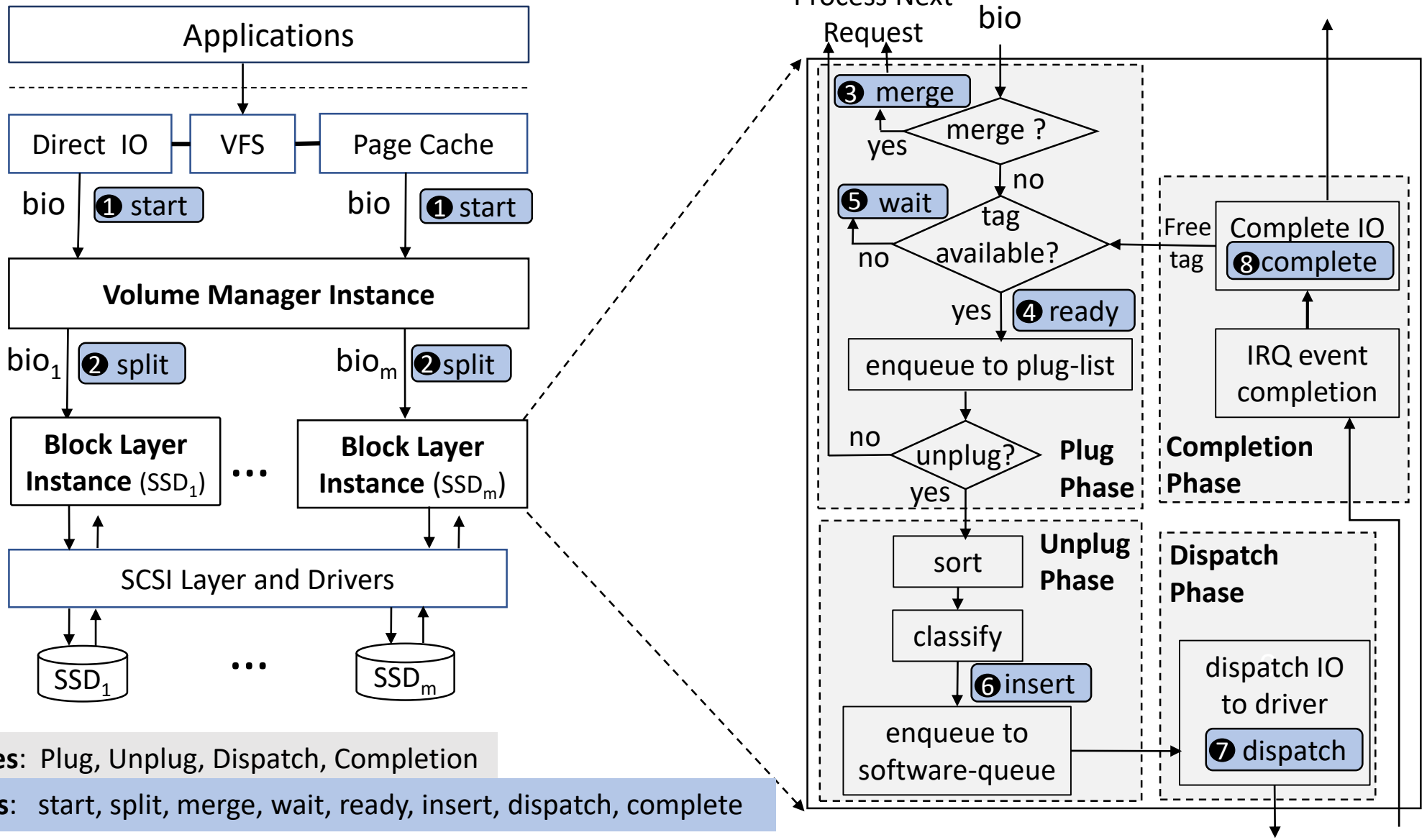
01 – Overview and Problem Statement ✓

02 – Background and Block Layer Insufficiency

03 – Falcon Architecture

04 – Evaluation

Linux: IO Flow and IO States

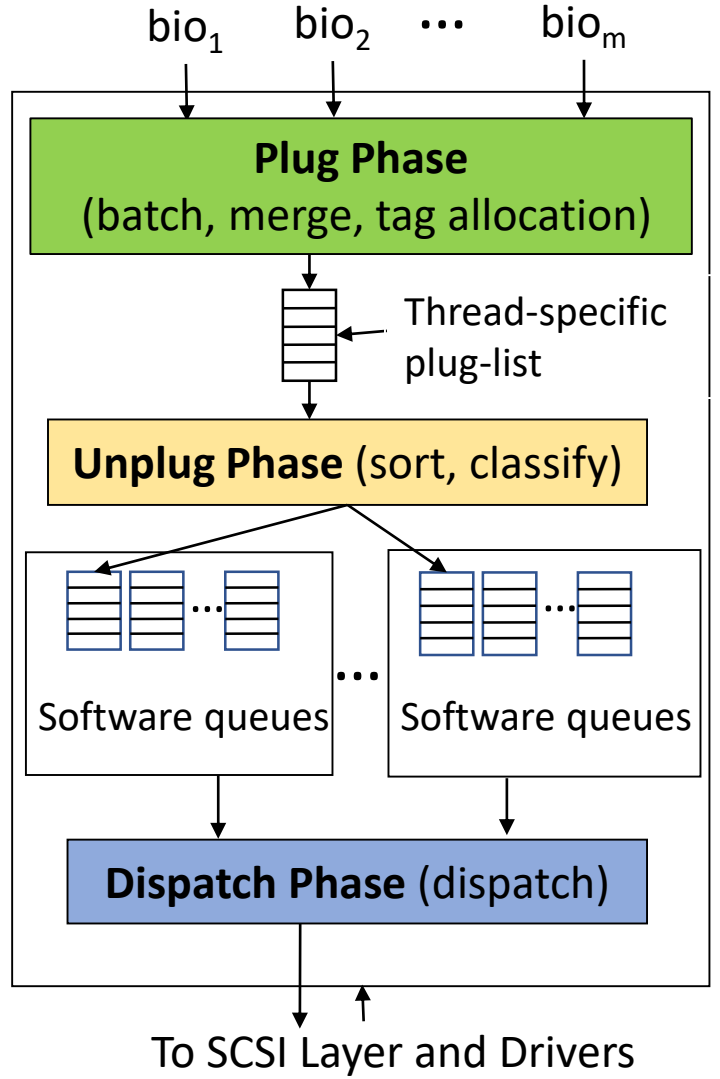


IO Phases: Plug, Unplug, Dispatch, Completion

IO States: start, split, merge, wait, ready, insert, dispatch, complete

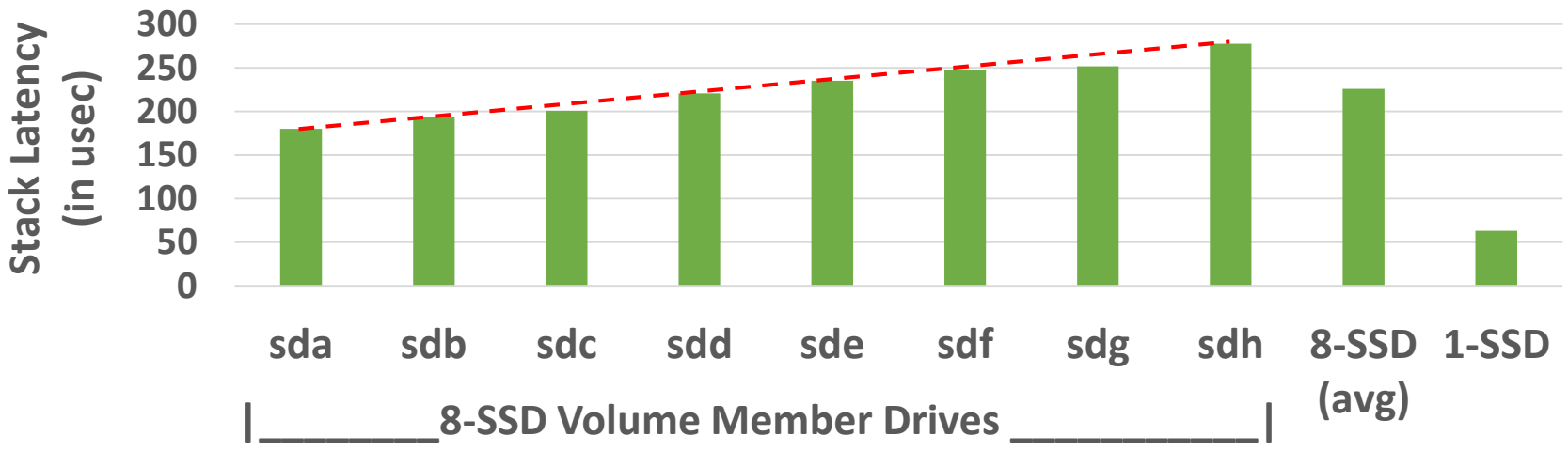
Linux: Mixes IO Batching and IO Serving Tasks

- 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



Linux block layer control flow

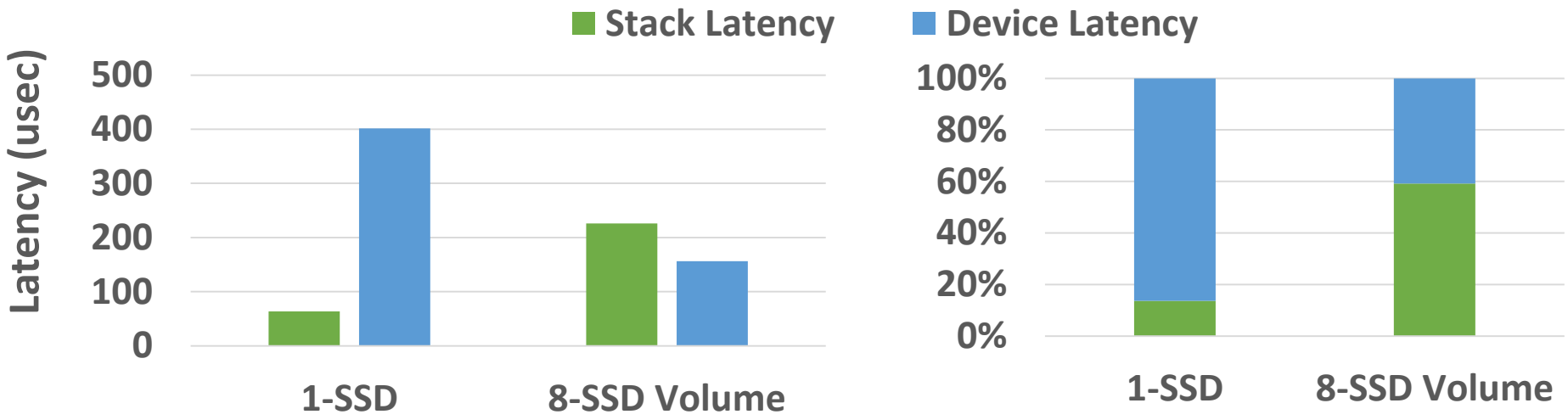
Insufficiency #1: Lack of Parallelism



- 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

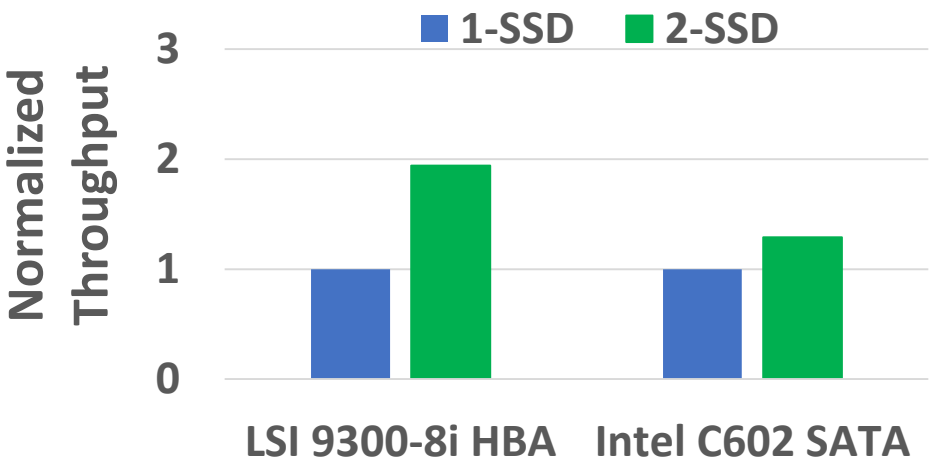


(a) Absolute latency **(b) Percentage**

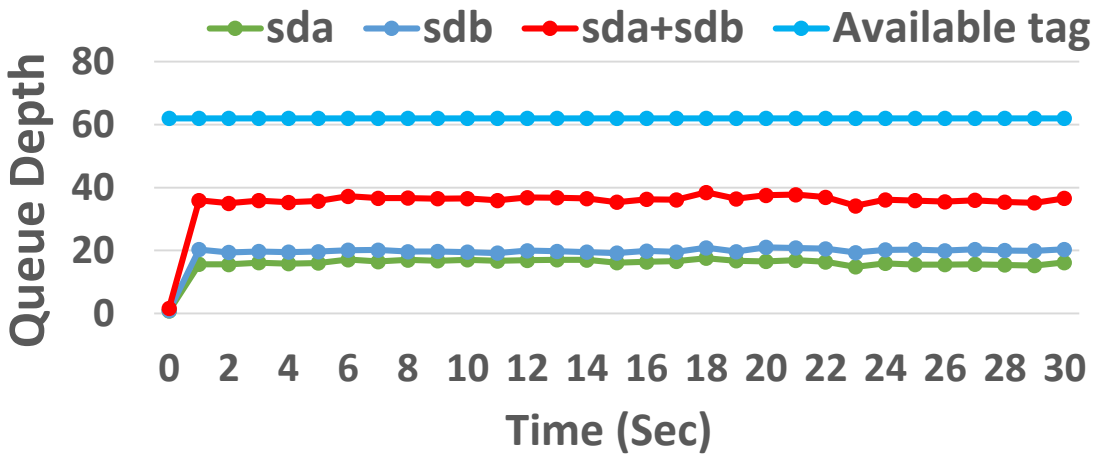
- Stack Latency in 8-SSD volume *is greater than* 1-SSD system
- Stack latency *is greater than* device latency in 8-SSD volume

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

Insufficiency #3: Unpredictable Blocking



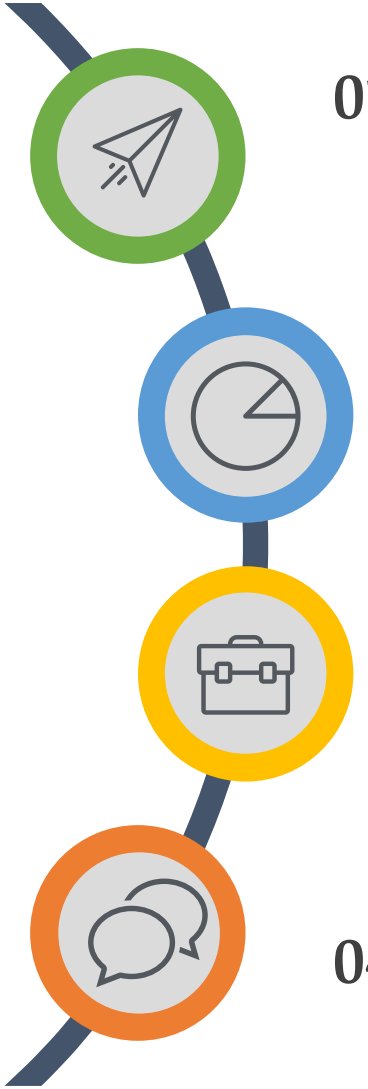
(a) IO performance Scaling



(b) Tag usage in 2-SSD SATA volume

- 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

Outline



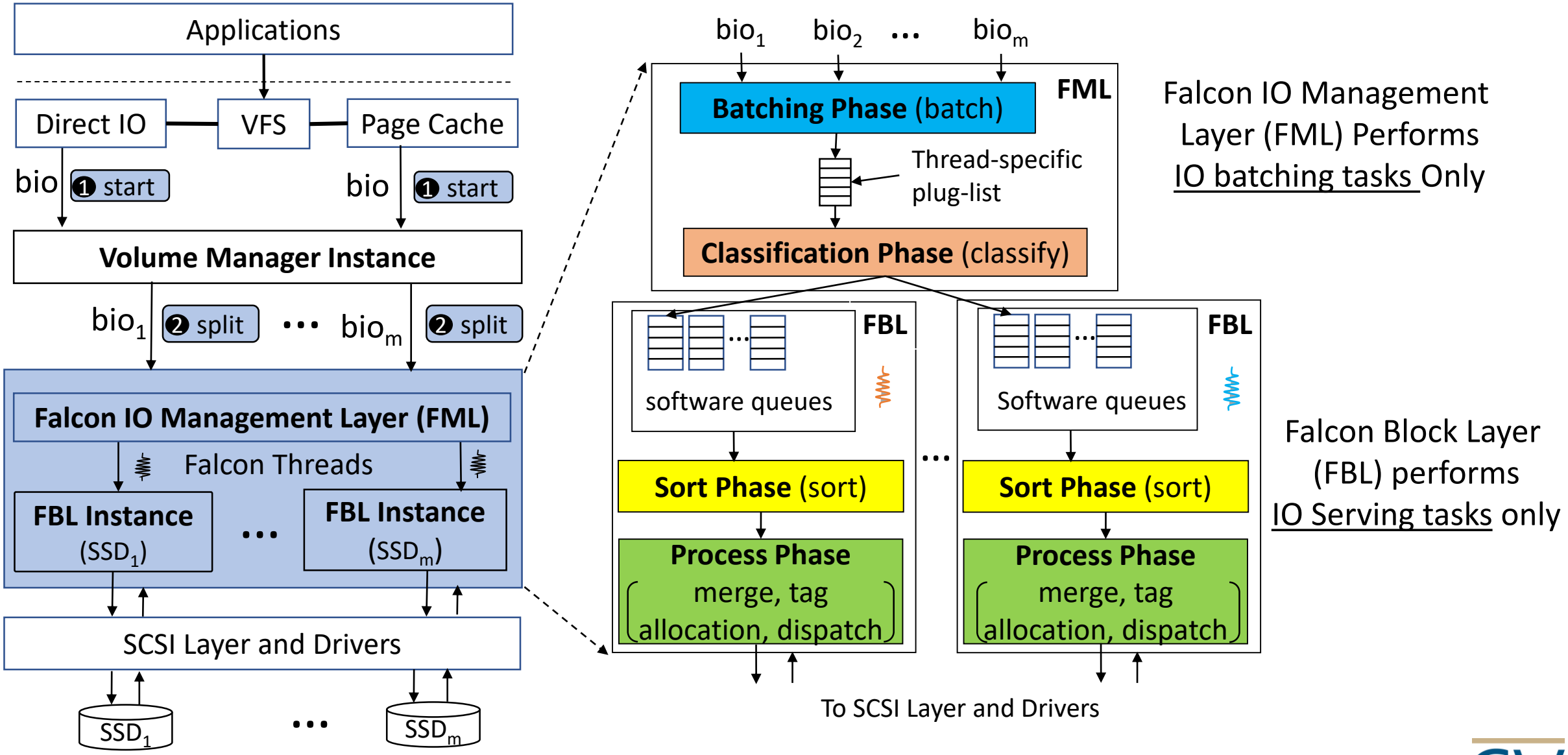
01 – Overview and Problem Statement ✓

02 – Background and Block Layer Insufficiency ✓

03 – Falcon Architecture

04 – Evaluation

Falcon: Separates IO Batching from IO Serving Tasks



Falcon IO Management Layer (FML) Performs IO batching tasks Only

Falcon Block Layer (FBL) performs IO Serving tasks only

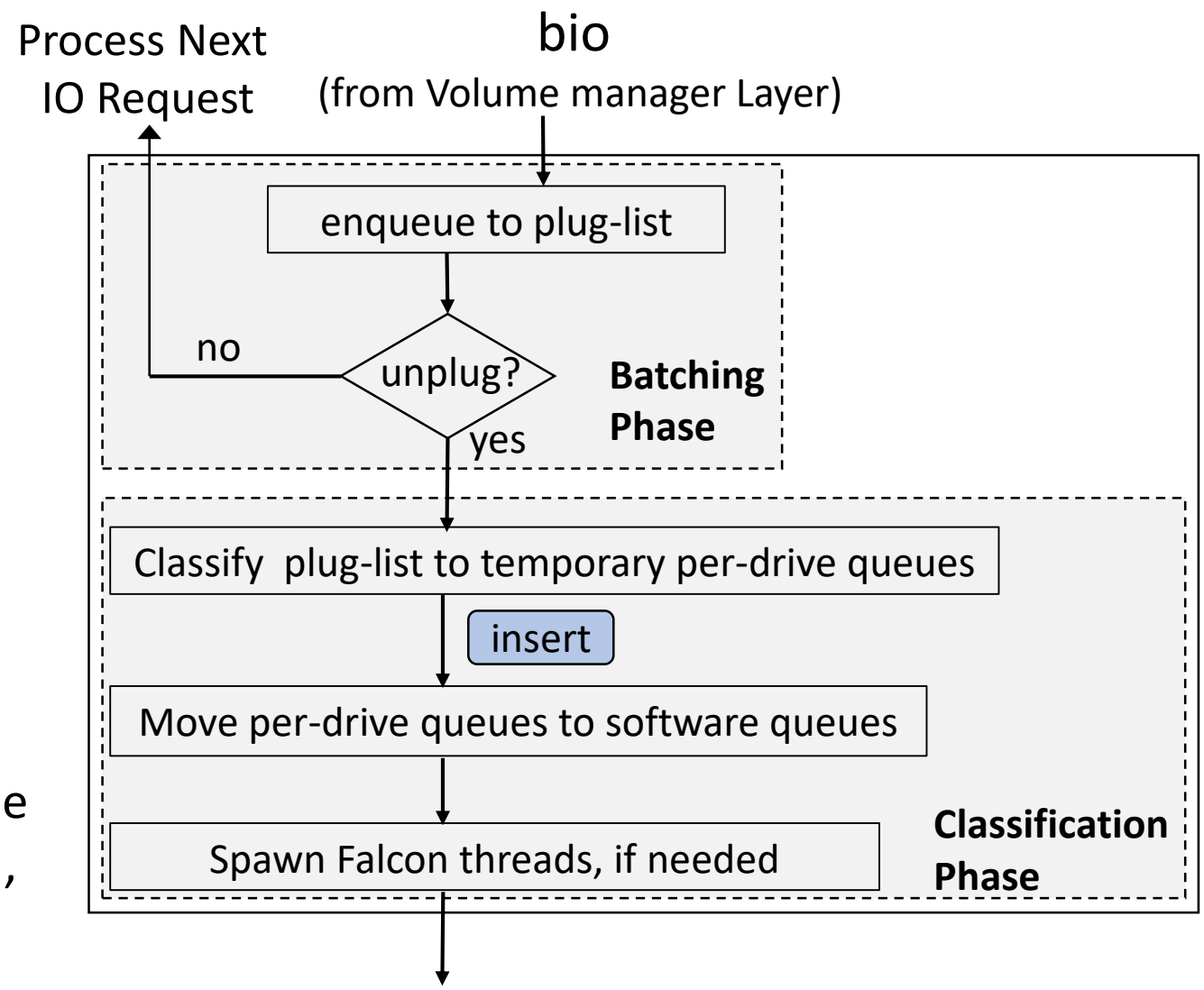
Falcon: Feature Comparison with Linux

Block Layer Features	Linux 1-SSD	Linux Multi-SSD	Falcon Volume
Parallel Processing	NA	✗	✓
Per-Drive Sort	✓	✗	✓
Neighbor Merge	✗	✗	✓
Dynamic Tag Management	✗	✗	✓

- Well-suited for multi-SSD volume
- Improvements are applicable to 1-SSD system also

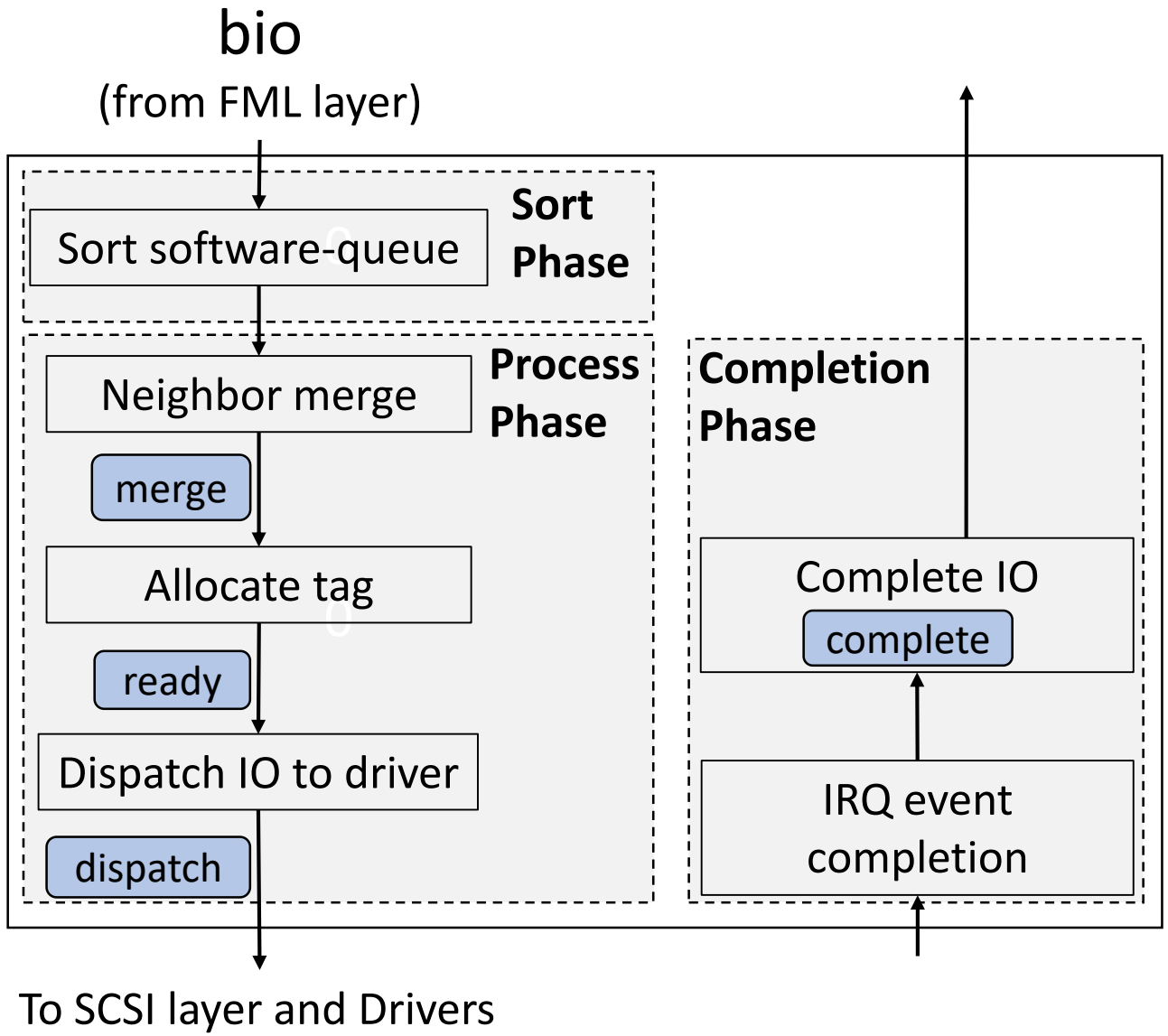
Falcon IO Management Layer

- 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



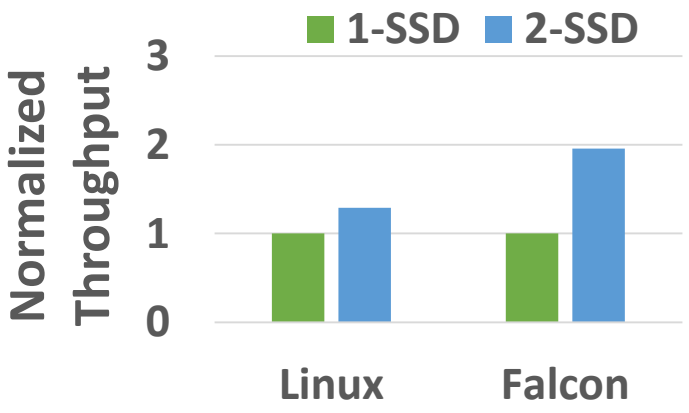
Falcon Block Layer

- 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)

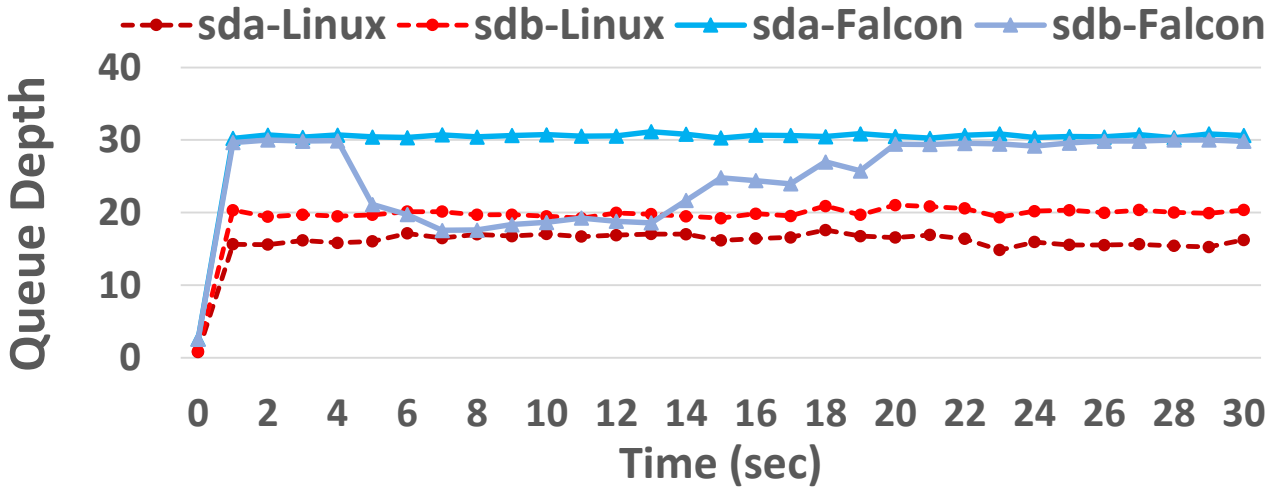


Dynamic Tag Management

- 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

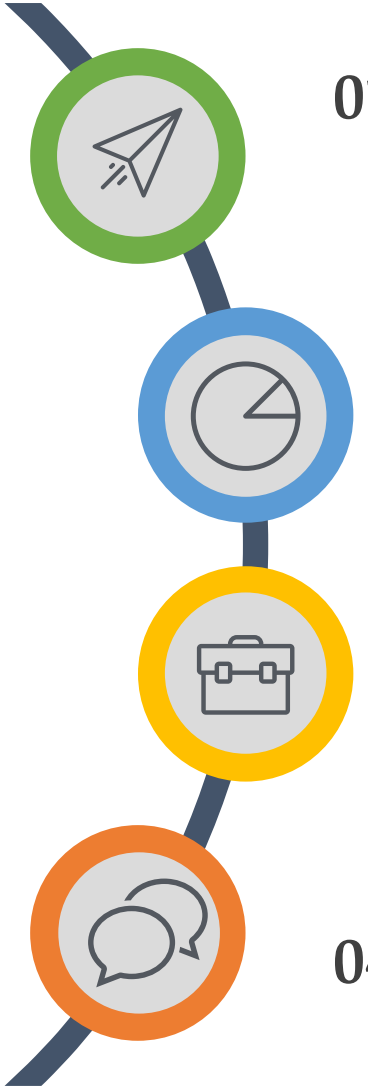


(a) IO Throughput Scaling for SATA controller



(b) Tag usage in 2-SSD SATA volume

Outline



01 – Overview and Problem Statement ✓

02 – Background and Block Layer Insufficiency ✓

03 – Falcon Architecture ✓

04 – Evaluation

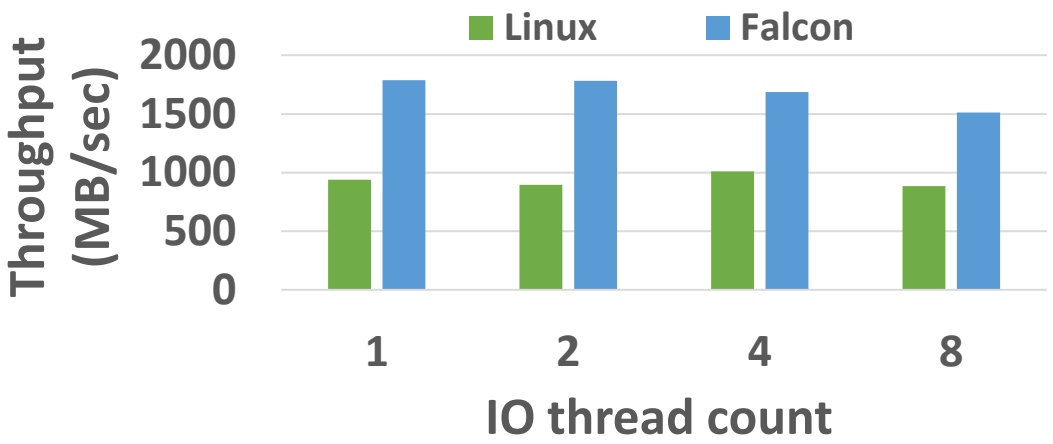
Evaluation Setup

- Falcon: 600 lines of C kernel code add
- Ubuntu 16.04 version with Kernel version 4.4.0, Blk-mq block layer

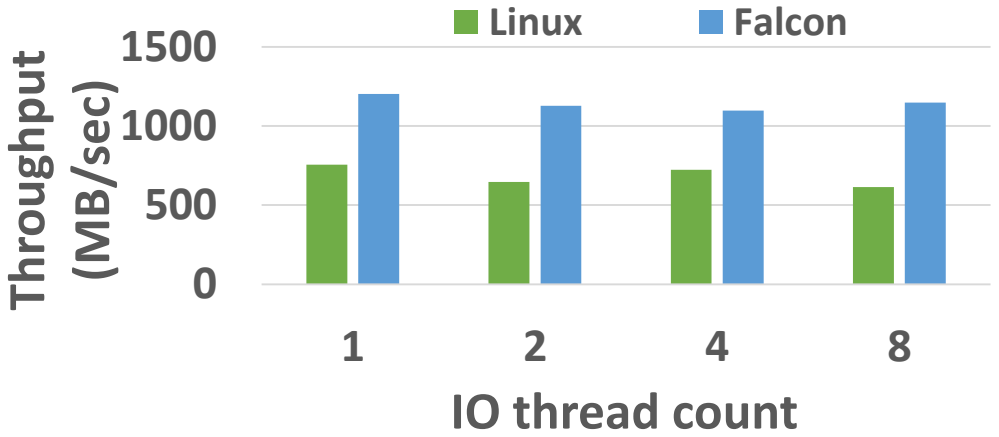
- 2 Intel Xeon CPU E5-2620 2GHz with 6 cores each
- 32GB DRAM
- 8 Samsung EVO 850 500 GB SSDs, connected using LSI SAS9300-8i HBA

- 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



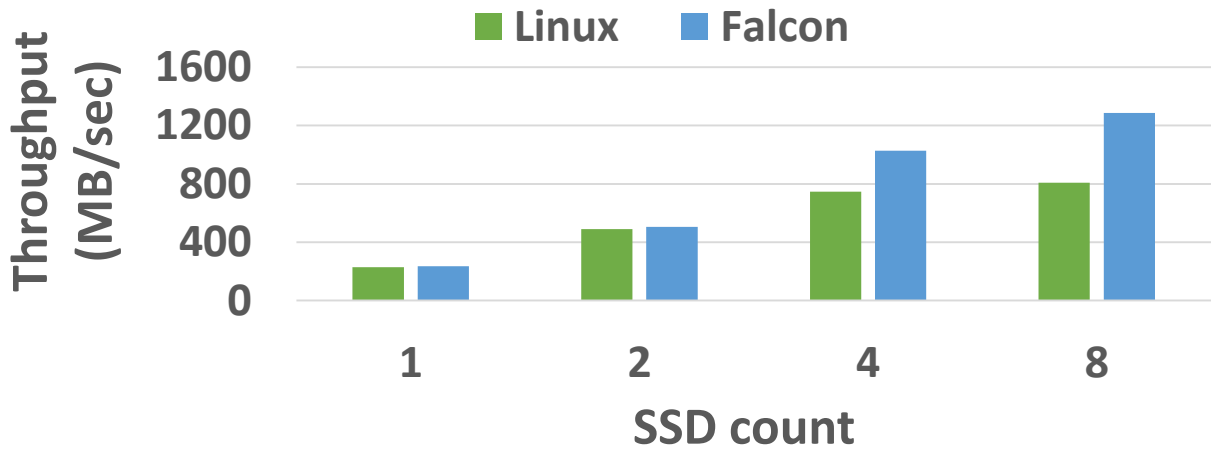
(a) Single file read throughput



(b) Single file write throughput

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

Buffered Random Write

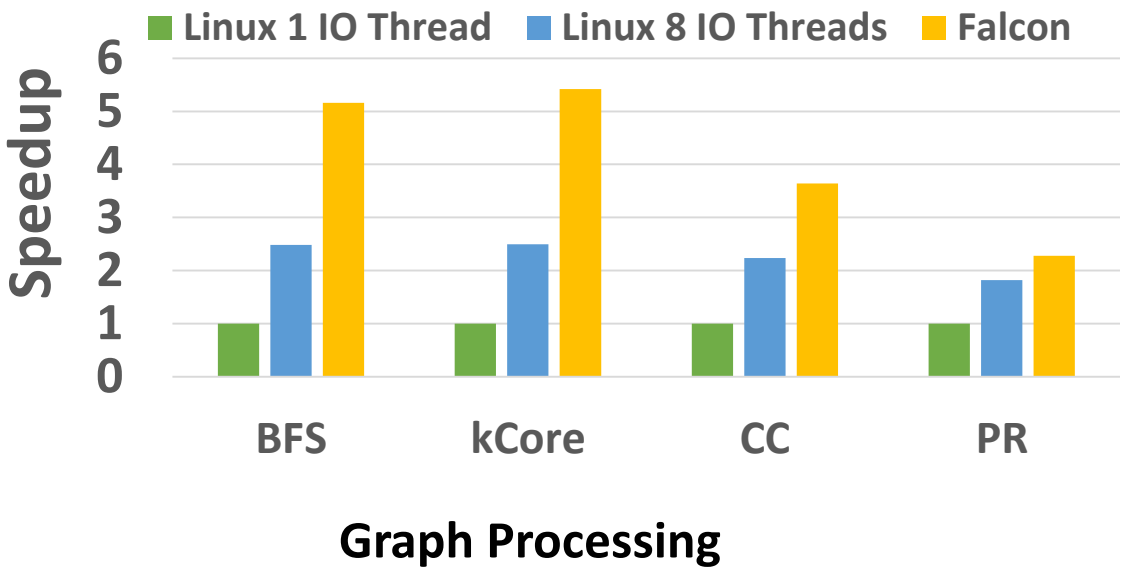


- Buffer cache has just 1 thread per volume for flushing dirty pages

- 1.39x speedup for 4-SSD volume
- 1.59x speedup for 8-SSD volume

Graph Processing

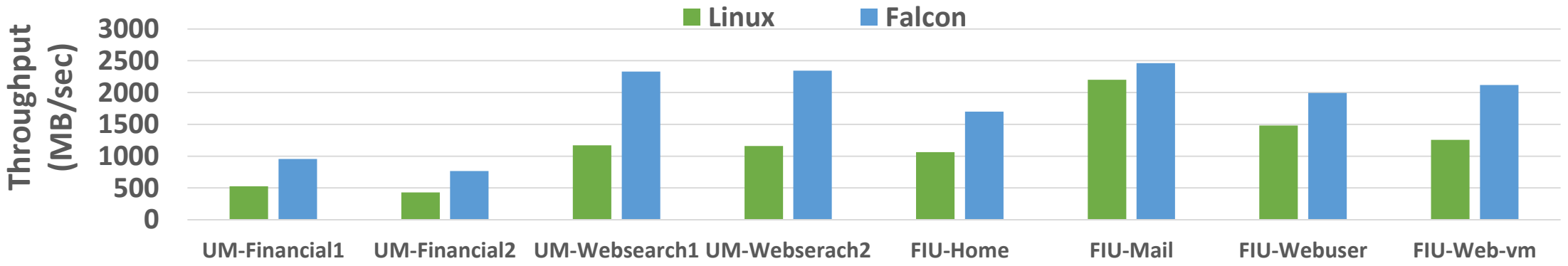
- G-Store[SC'16] is used
  - Semi-external graph analytics engine
  - Configurable number of IO threads
  - Linux setup, 1 and 8 IO threads
  - Falcon : 1 IO thread
- 8-SSD volume, XFS filesystem
- Kronecker graph scale 28, edge factor 16 is used
- BFS, kCore: High random IO
- Connected component (CC), PageRank (PR): Sequential IO



- 4.12x speedup over Linux 1 IO thread setup
- 1.78x speedup over Linux 8 IO thread

IO Trace Replay

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



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