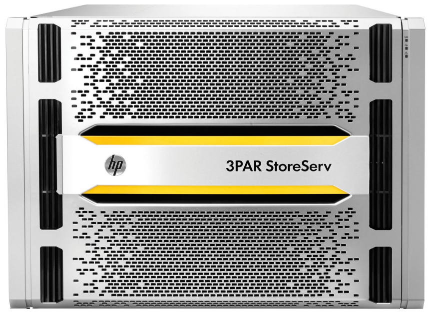
Alleviating Garbage Collection Interference through Spatial Separation in All Flash Arrays

Jaeho Kim, Kwanghyun Lim*, Youngdon Jung,
Sungjin Lee, Changwoo Min, Sam H. Noh

*Currently with Cornell Univ.

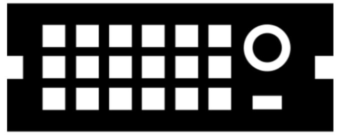
All Flash Array (AFA)

- Storage infrastructure that contains only flash memory drives
  - Also called Solid-State Array (SSA)



https://images.google.com/
https://www.purestorage.com/resources/glossary/all-flash-array.html

Example of All Flash Array Products (1 brick or node)

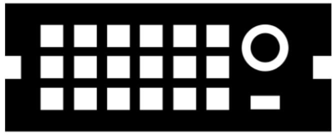


	EMC XtremIO	HPE 3PAR	SKHynix AFA
Capacity	36 ~ 144 TB	750 TB	552 TB
Number of SSDs	18 ~ 72	120	576
Network Ports	4~8 x 10Gb iSCSI	4~12 x 16Gb FC	3 x Gen3 PCIe
Aggregate Network Throughput	5 ~ 10 GB/s	8 ~ 24 GB/s	48 GB/s

A: EMC XtremIO X2 Specification
B: HPE 3PAR StoreServ Specification
C: Performance Analysis of NVMe SSD-Based All-flash Array Systems. *[ISPASS'18]*

https://www.flaticon.com/

Example of
All Flash Array Products (1 brick or node)



	EMC XtremIO	HPE 3PAR	SKHynix AFA
Capacity	36 ~ 144 TB	750 TB	552 TB
Number of SSDs	18 ~ 72	120	576
Network Ports	4~8 x 10Gb iSCSI	4~12 x 16Gb FC	3 x Gen3 PCIe
Aggregate Network Throughput	5 ~ 10 GB/s	8 ~ 24 GB/s	48 GB/s

A: EMC XtremIO X2 Specification
B: HPE 3PAR StoreServ Specification
C: Performance Analysis of NVMe SSD-Based All-flash Array Systems. *[ISPASS'18]*

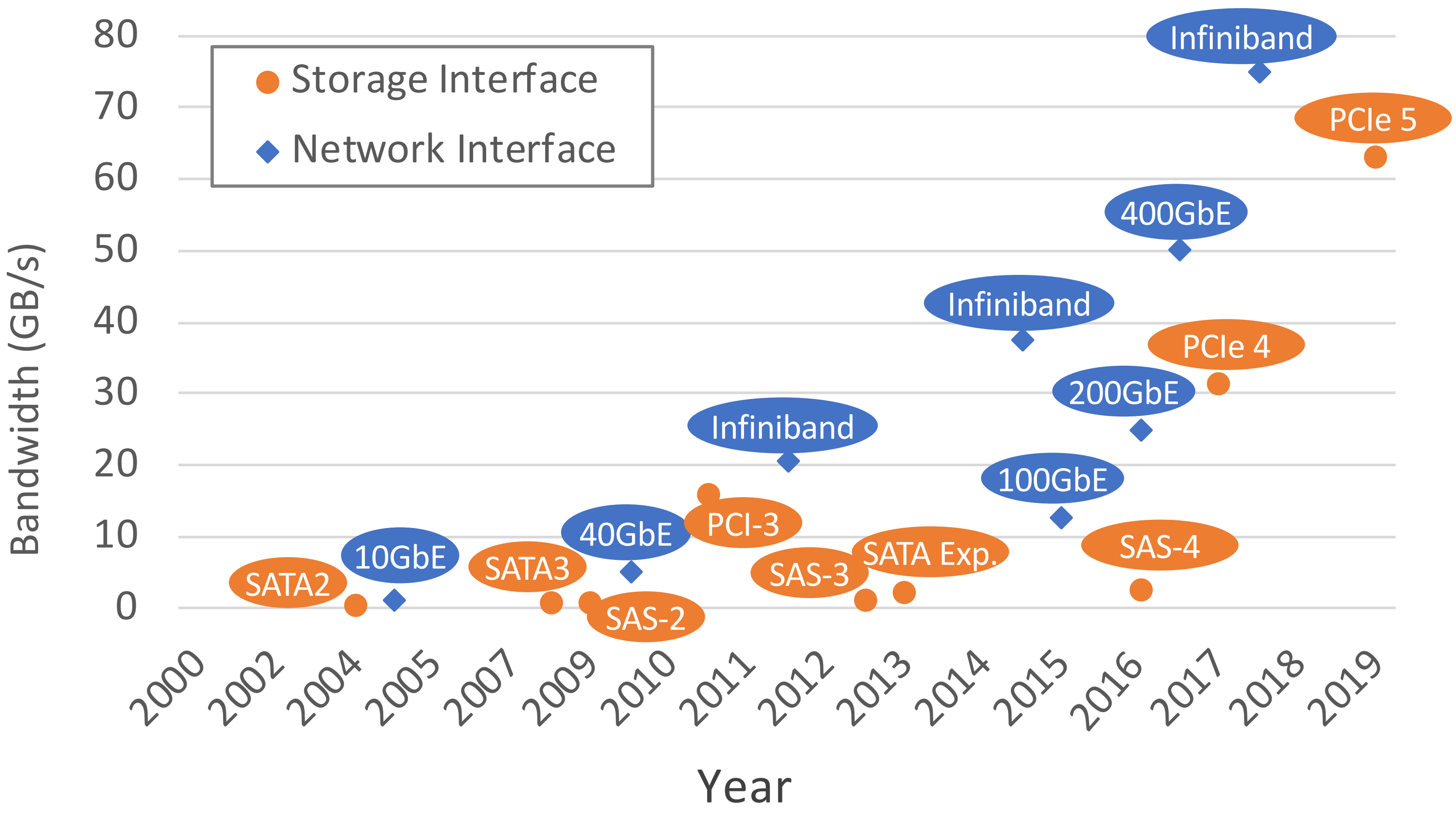
https://www.flaticon.com/

SSDs for Enterprise

Manufacturer	Product Name	Seq. Read Throughput	Seq. Write Throughput	Capacity
Intel	DC P4800X	2.5 GB/s	2.2 GB/s	1.5 TB
	DC D3700	2.1 GB/s	1.5 GB/s	1.6 TB
	DC P3608	5 GB/s	3 GB/s	4 TB
Samsung	PM1725b	6.3 GB/s	3.3 GB/s	12.8 TB
	PM983	3.2 GB/s	2 GB/s	3.8 TB

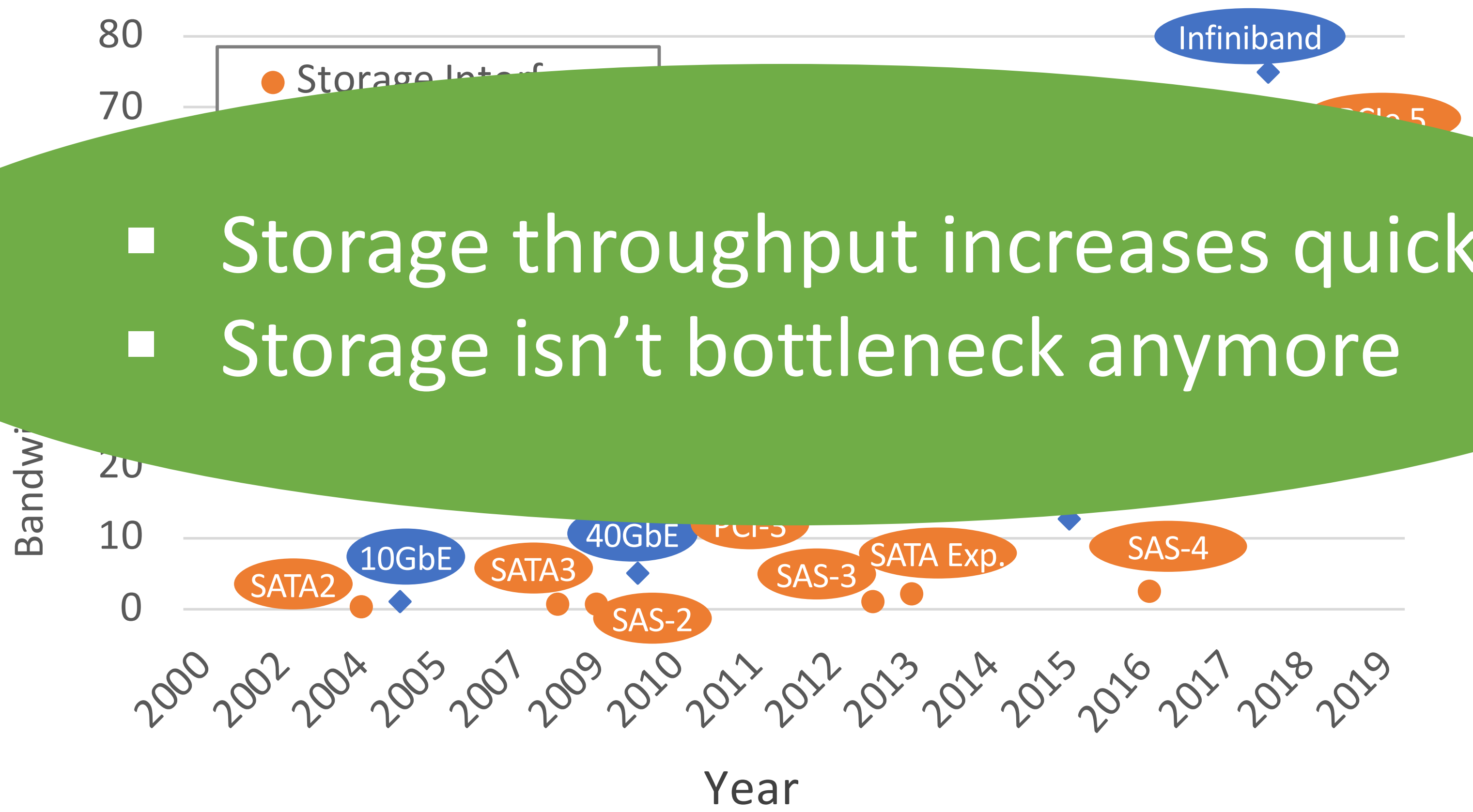
Intel: https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/data-center-ssds.html
Samsung: https://www.samsung.com/semiconductor/ssd/enterprise-ssd/

Bandwidth Trends for Network and Storage Interfaces



Interfaces: https://en.wikipedia.org/wiki/List_of_interface_bit_rates#Local_area_networks
SATA: https://en.wikipedia.org/wiki/Serial_ATA
PCIe: https://en.wikipedia.org/wiki/PCI_Express

Bandwidth Trends for Network and Storage Interfaces



- Storage throughput increases quickly
- Storage isn't bottleneck anymore

Interfaces: https://en.wikipedia.org/wiki/List_of_interface_bit_rates#Local_area_networks
SATA: https://en.wikipedia.org/wiki/Serial_ATA
PCIe: https://en.wikipedia.org/wiki/PCI_Express

Example of All Flash Array Products (1 brick or node)

	EMC XtremIO	HPE 3PAR	SKHynix AFA
Capacity	36 ~ 144 TB	750 TB	552 TB
Number of SSDs	18 ~ 72	120	576
Network Ports	4~8 x 10Gb iSCSI	4~12 x 16Gb FC	3 x Gen3 PCIe
Aggregate Network Throughput	5 ~ 10 GB/s	8 ~ 24 GB/s	48 GB/s

A: EMC XtremIO X2 Specification
B: HPE 3PAR StoreServ Specification
C: Performance Analysis of NVMe SSD-Based All-flash Array Systems. *[ISPASS'18]*

Example of
All Flash Array Products (1 brick or node)

Throughput of a few high-end SSDs can easily saturate the network throughput

[illegible]	[illegible]	[illegible]	AFA [illegible]
[illegible]	[illegible]	[illegible]	[illegible]
Network Ports	4 [illegible]	4 ~ 12 x 16Gb FC	3 x Gen3 PCIe
Aggregate Network Throughput	5 ~ 10 GB/s	8 ~ 24 GB/s	48 GB/s

A: EMC XtremIO X2 Specification
B: HPE 3PAR StoreServ Specification
C: Performance Analysis of NVMe SSD-Based All-flash Array Systems. *[ISPASS'18]*

Current Trends and Challenges

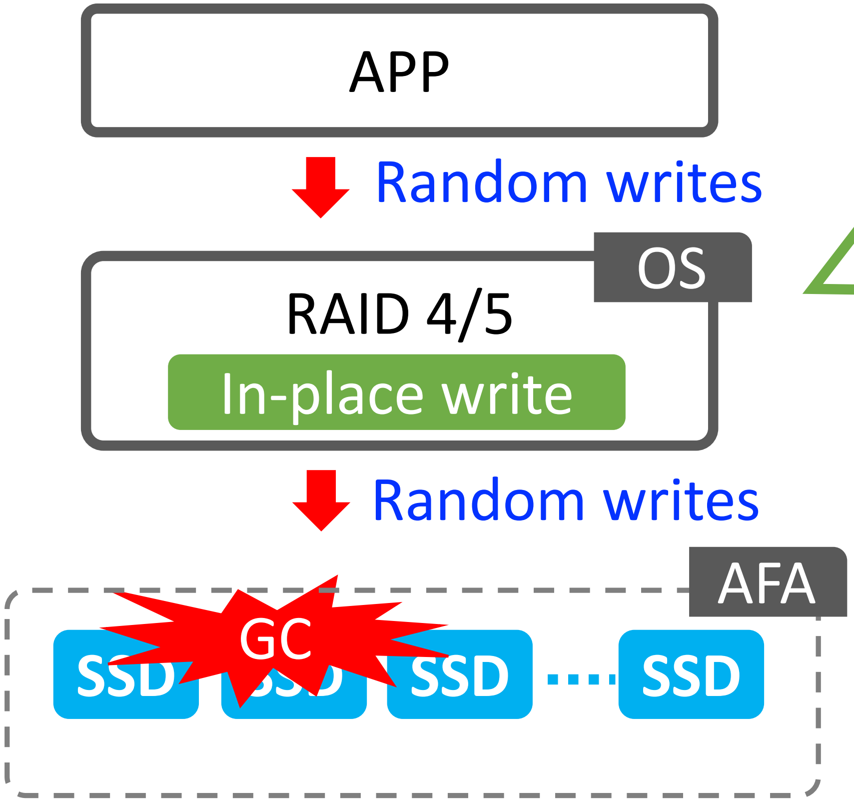
Trends

- Performance of SSDs is fairly high
- Throughput of a few SSDs easily saturates network bandwidth of a AFA node

Challenges

- Garbage Collection (GC) of SSD is still performance bottleneck in AFA
- What is an ideal way to manage an array of SSDs with the current trends?

Traditional RAID Approaches

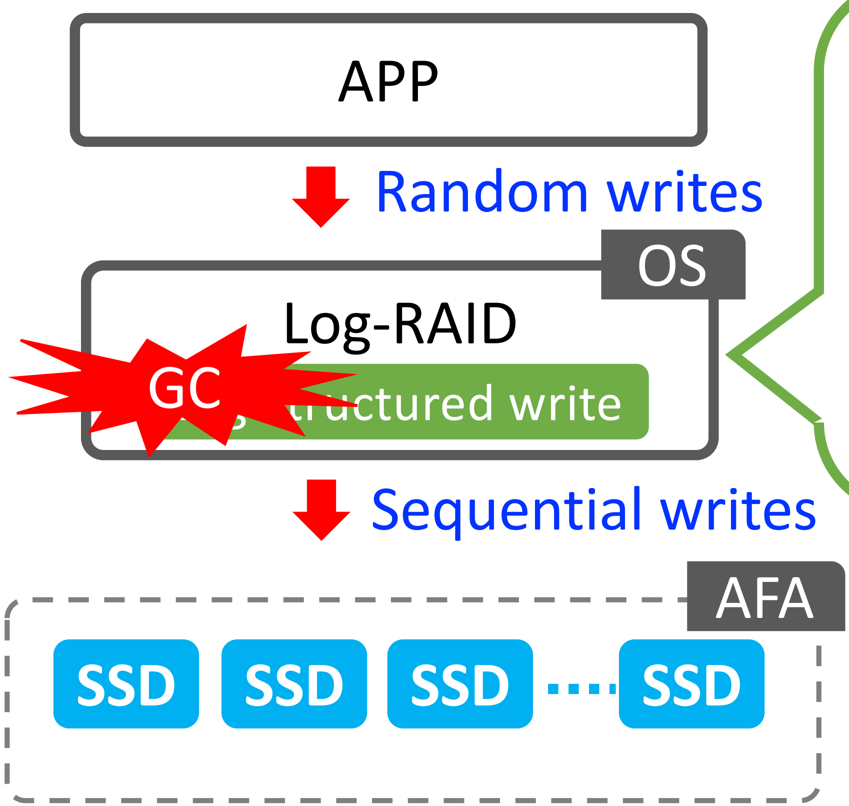


- Traditional RAID employs in-place update for serving write requests
- High GC overhead inside SSD due to random write from the host

Limitations

- Previous solutions
  1) Harmonia [MSST’11]
  2) HPDA [TOS’12]
  3) GC-Steering [IPDPS’18]

Log-(based) RAID Approaches



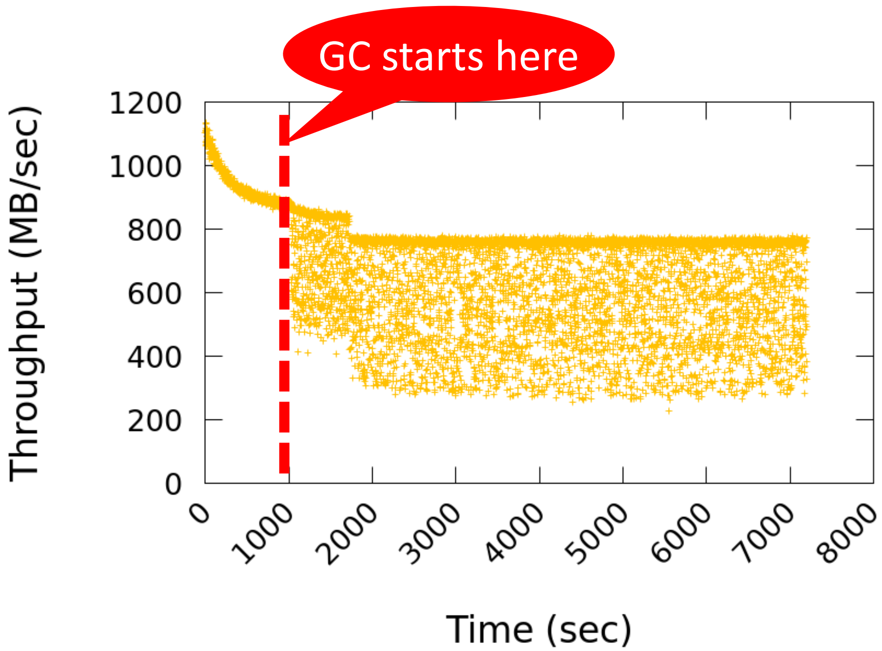
- Log-based RAID employs log-structured writes to reduce GC overhead inside SSD
- Log-structured writes involve host-level GC, which relies on idle time
- If no idle time, GC will cause performance drop

Limitations

- Previous solutions
  1) SOFA [SYSTOR’14]
  2) SRC [Middleware’15]
  3) SALSA [MASCOTS’18]

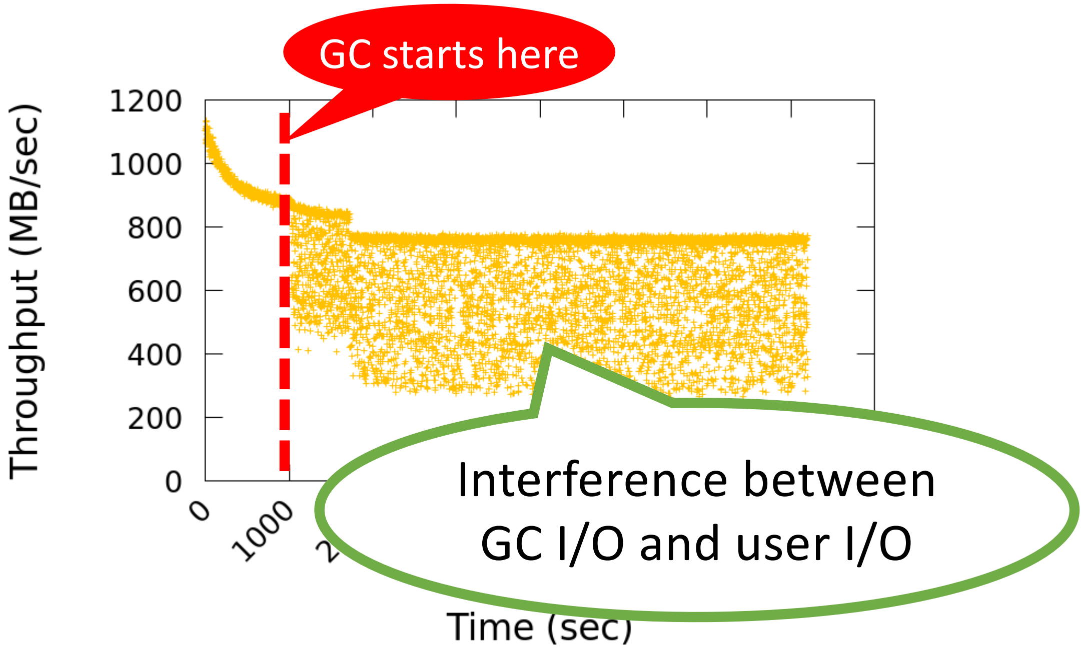
Performance of a Log-based RAID

- Configuration
  - Consist of 8 SSDs (roughly 1TB capacity)
- Workload
  - Random write requests continuously for 2 hours



Performance of a Log-based RAID

- Configuration
  - Consist of 8 SSDs (roughly 1TB capacity)
- Workload
  - Random write requests continuously for 2 hours

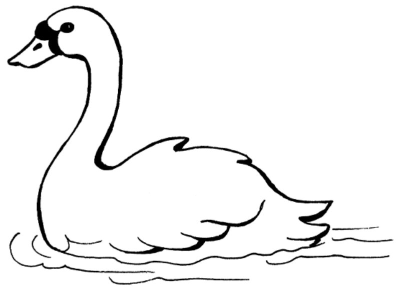


Performance of a Log-based RAID

- Configuration
  - Consist of 8 SSDs (roughly 1TB capacity)
- Workload
  - Random write requests continuously for 2 hours

How can we avoid this performance variation due to GC in All Flash Array?

Our Solution (SWAN)



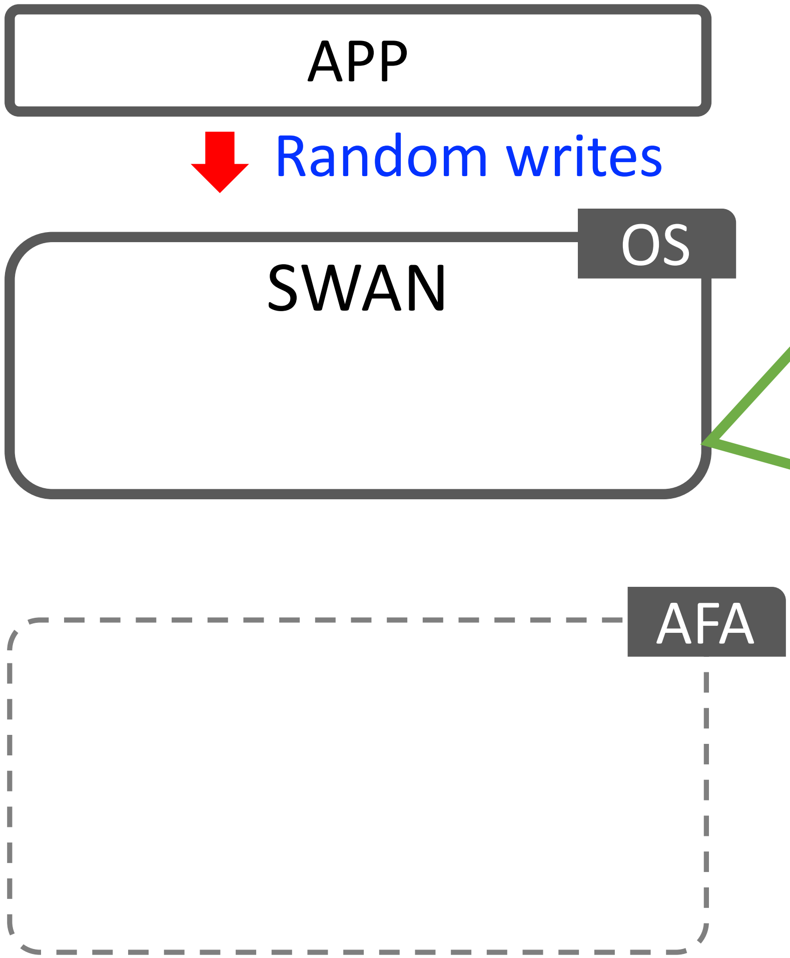
- SWAN (**S**patial separation **W**ithin an **A**rray of SSDs on a **N**etwork)

- Goals
  - Provide sustainable performance up to network bandwidth of AFA
  - Alleviate GC interference between user I/O and GC I/O
  - Find an efficient way to manage an array of SSDs in AFA

- Approach
  - Minimize GC interference through SPATIAL separation

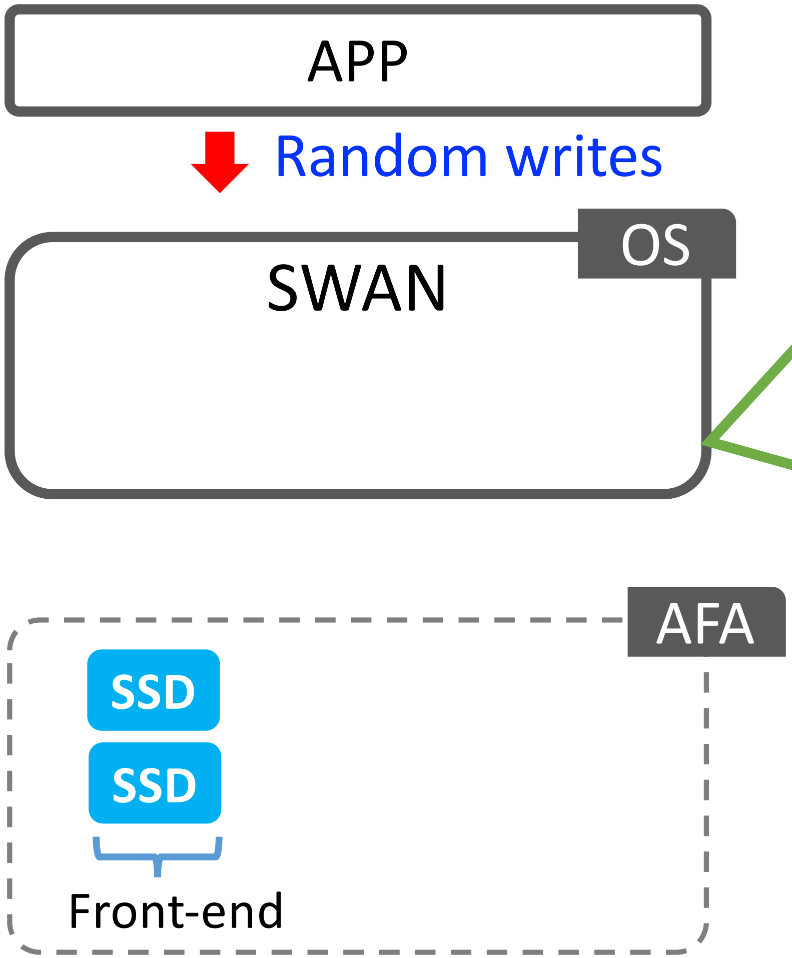
Image: https://clipartix.com/swan-clipart-image-44906/

Our Solution: Brief Architecture of SWAN



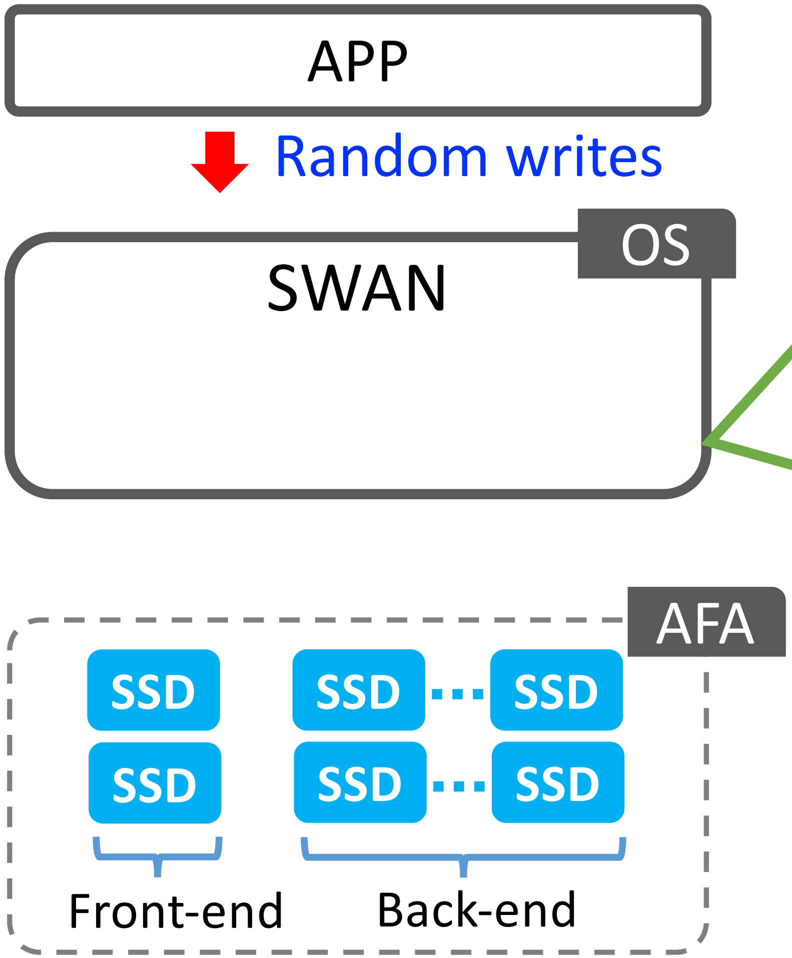
- Divide an array of SSDs into front-end and back-end like 2-D array
  - Called, SPATIAL separation
- Employ log-structured writes
- GC effect is minimized by spatial separation

Our Solution: Brief Architecture of SWAN



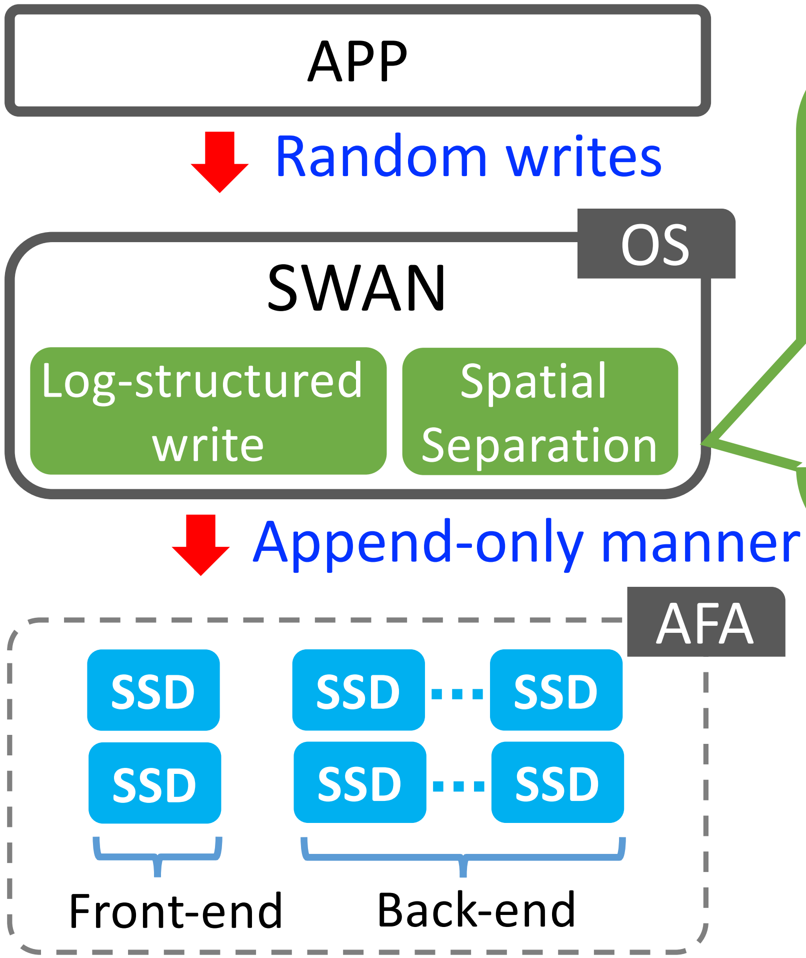
- Divide an array of SSDs into front-end and back-end like 2-D array
  - Called, SPATIAL separation
- Employ log-structured writes
- GC effect is minimized by spatial separation

Our Solution: Brief Architecture of SWAN



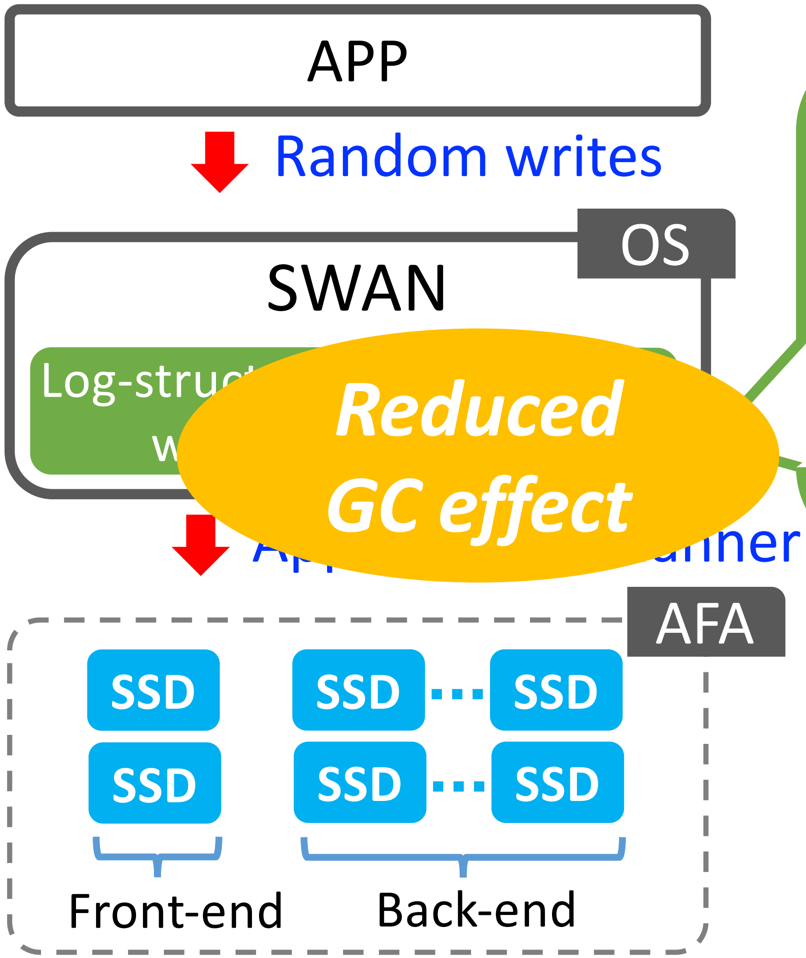
- Divide an array of SSDs into front-end and back-end like 2-D array
  - Called, SPATIAL separation
- Employ log-structured writes
- GC effect is minimized by spatial separation

Our Solution: Brief Architecture of SWAN



- Divide an array of SSDs into front-end and back-end like 2-D array
  - Called, SPATIAL separation
- Employ log-structured writes
- GC effect is minimized by spatial separation

Our Solution: Brief Architecture of SWAN



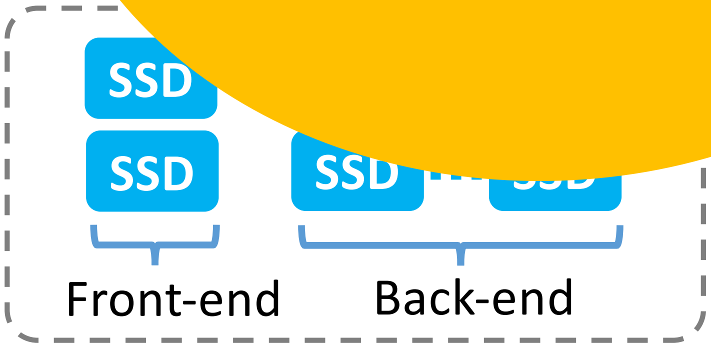
- Divide an array of SSDs into front-end and back-end like 2-D array
  - Called, SPATIAL separation
- Employ log-structured writes
- GC effect is minimized by spatial separation

Our Solution: Brief Architecture of SWAN

***Log-based RAID*:**
Temporal separation
between GC and
user I/O

VS.

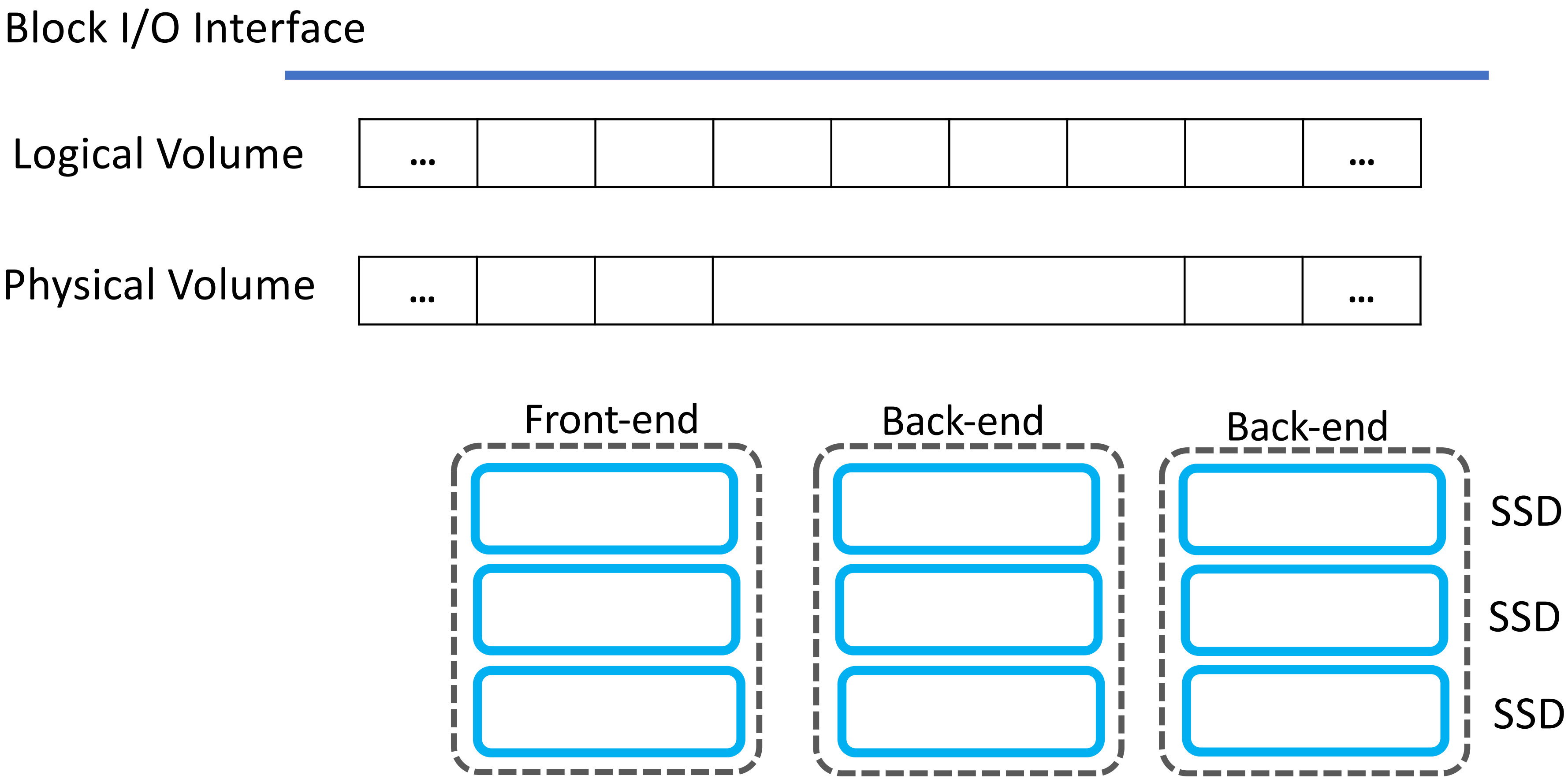
***SWAN*:**
Spatial separation
between GC and
user I/O



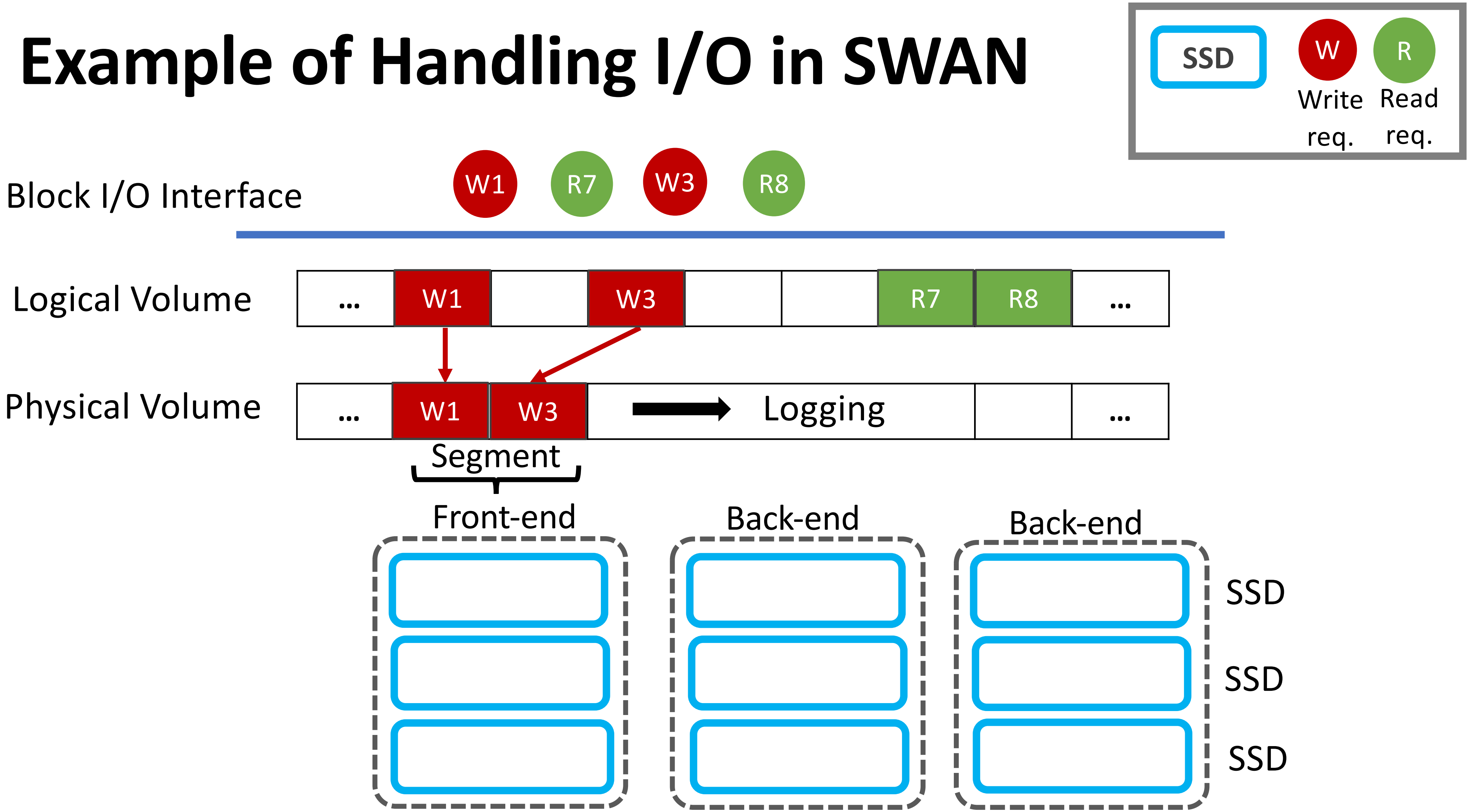
Architecture of SWAN

- Spatial separation
  - Front-end: serve all write requests
  - Back-end: perform SWAN's GC
- Log-structured write
  - Segment based append only writes, which is flash friendly
  - Mapping table: 4KB granularity mapping table
- Implemented in block I/O layer
  - where I/O requests are redirected from the host to the storage

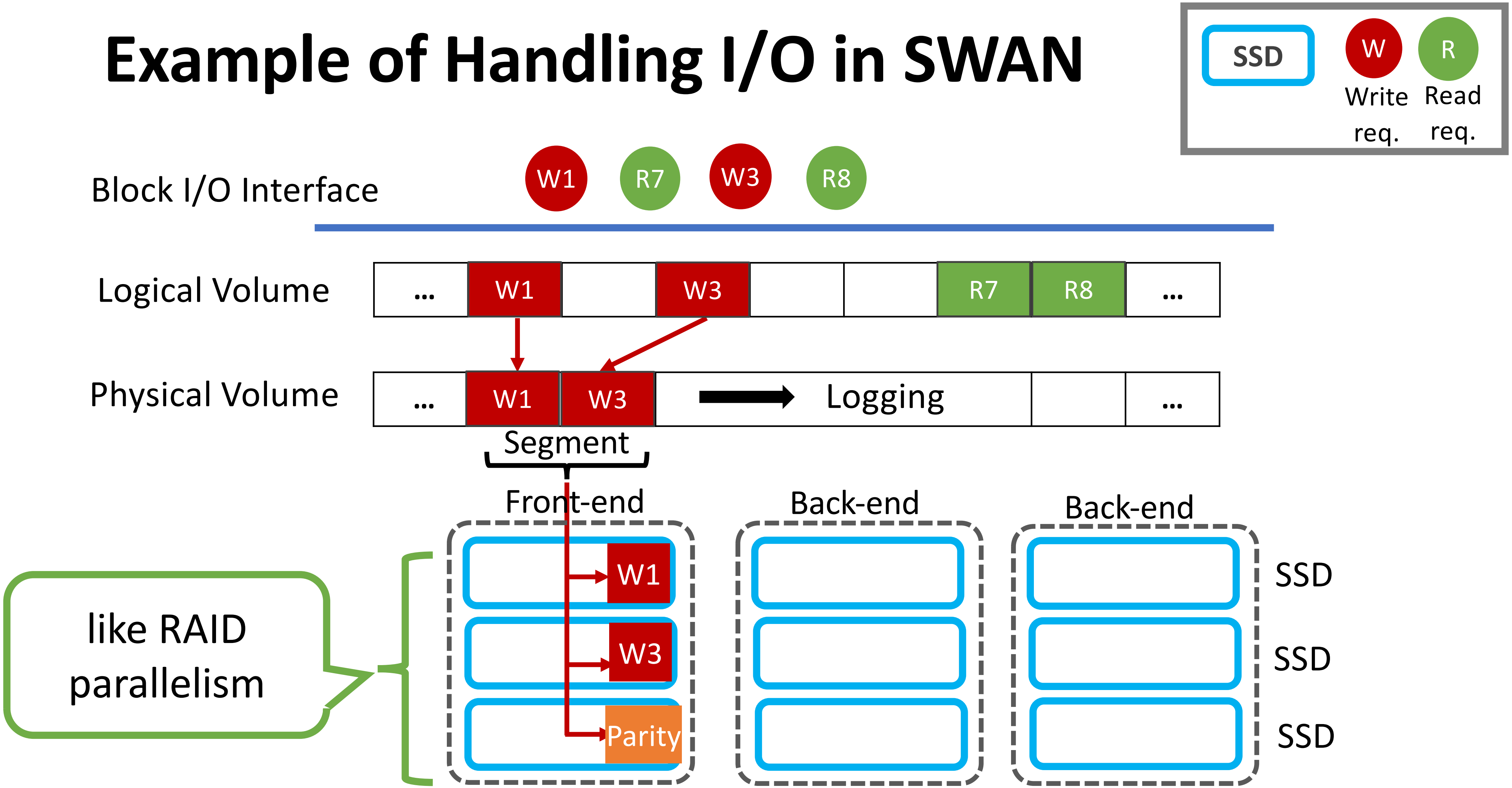
Example of Handling I/O in SWAN



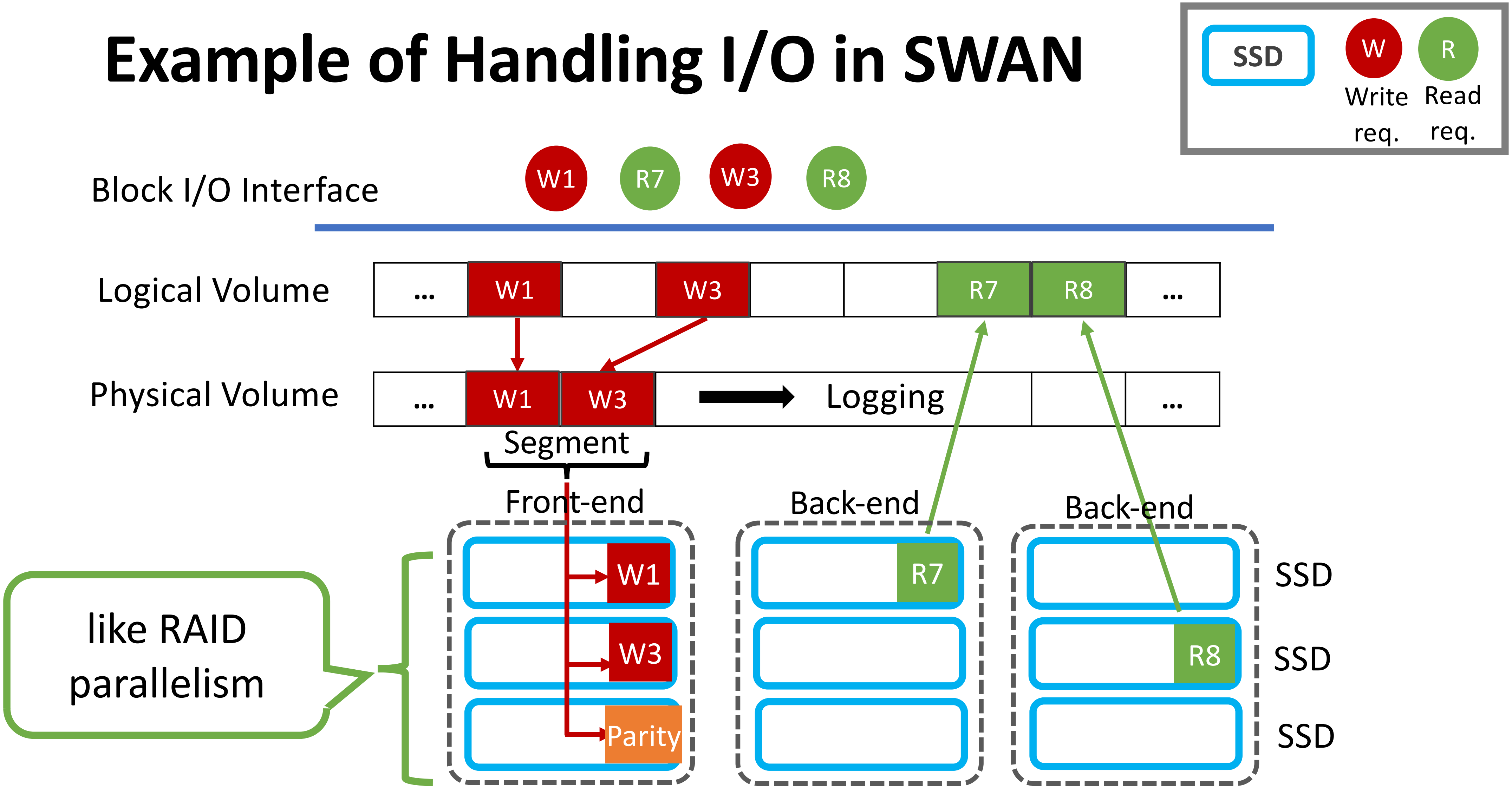
Example of Handling I/O in SWAN



Example of Handling I/O in SWAN

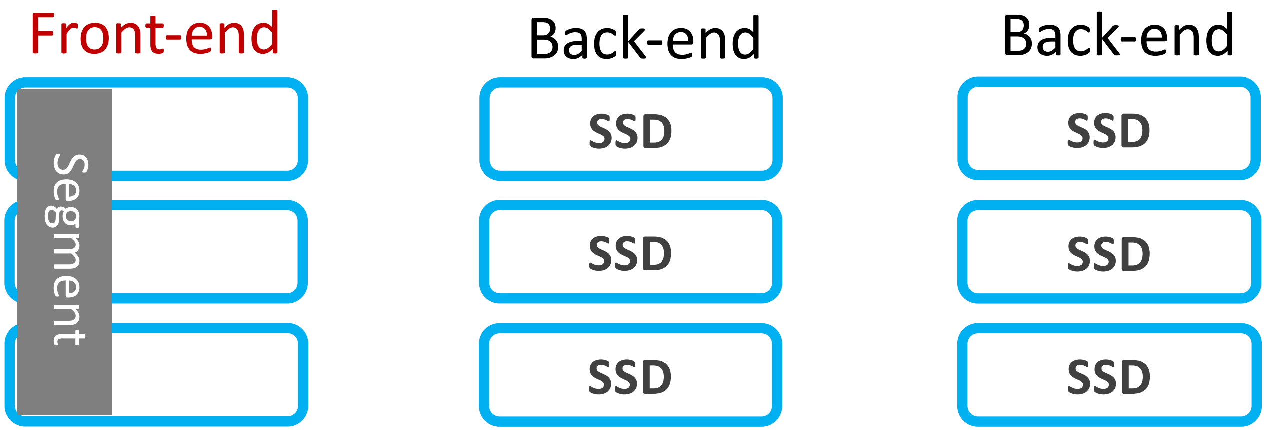


Example of Handling I/O in SWAN



Procedure of I/O Handling (1/3)

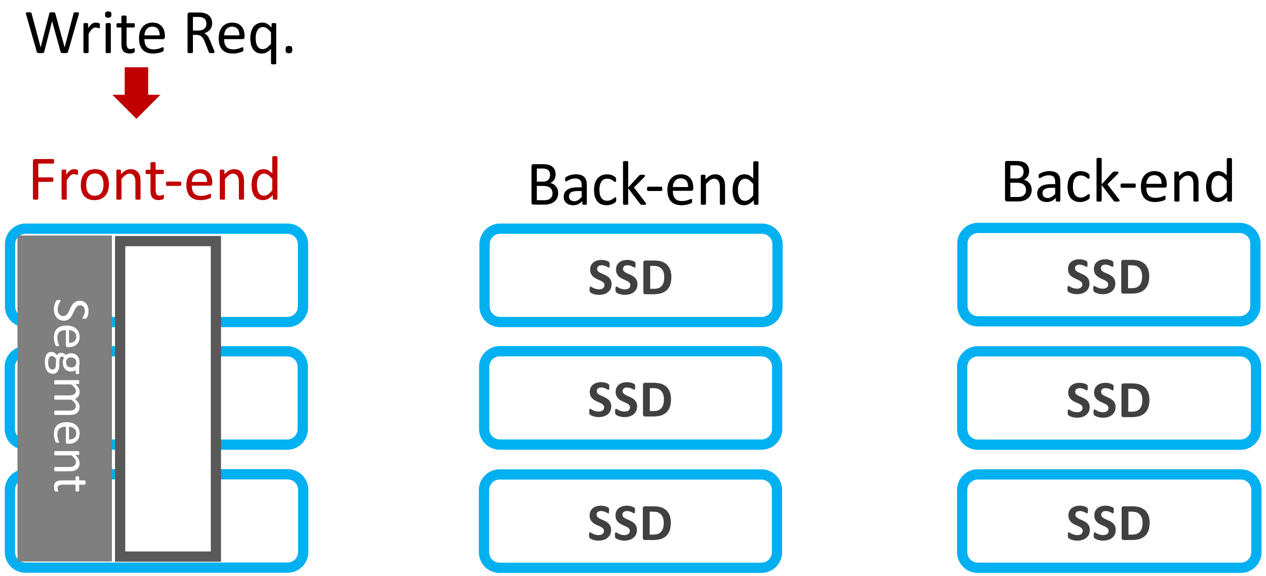
- Front-end absorbs all write requests in append-only manner
  - To exploit full performance of SSDs



(a) First - phase

Procedure of I/O Handling (1/3)

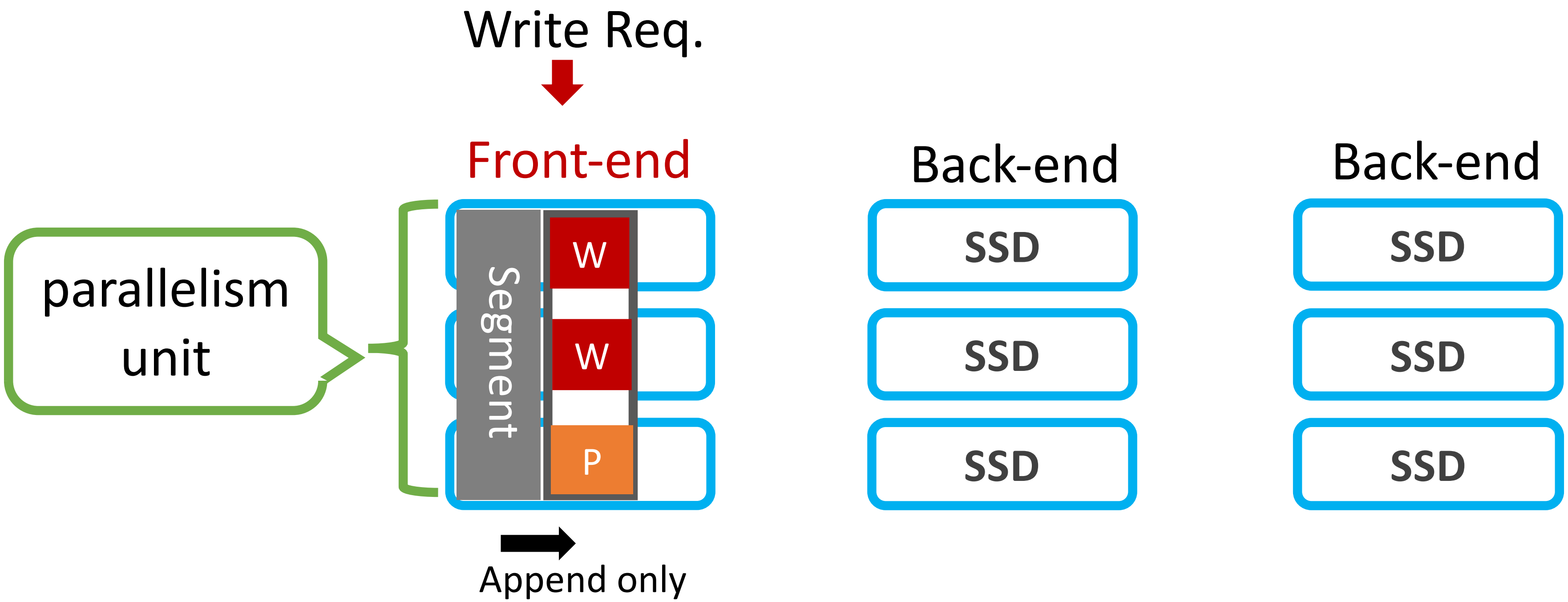
- Front-end absorbs all write requests in append-only manner
  - To exploit full performance of SSDs



(a) First - phase

Procedure of I/O Handling (1/3)

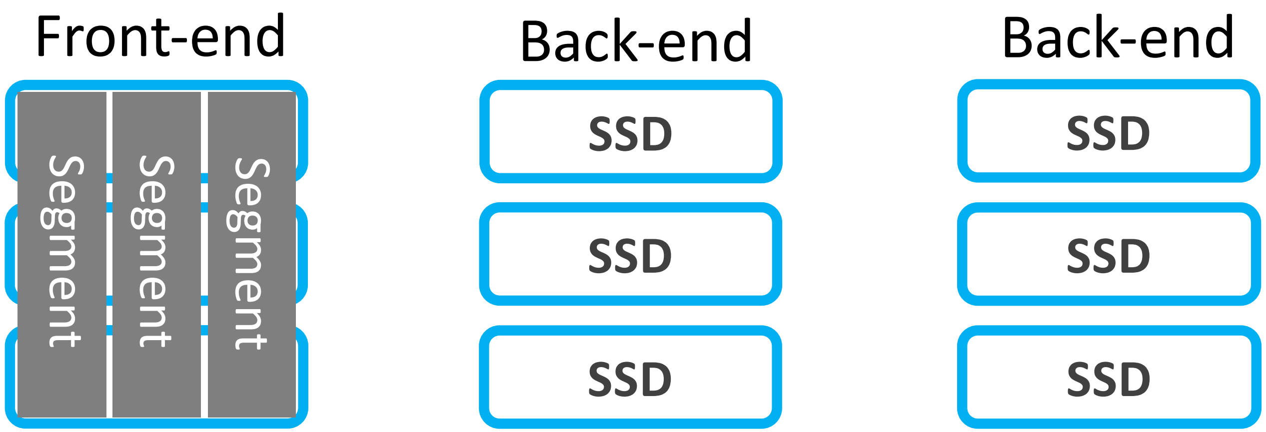
- Front-end absorbs all write requests in append-only manner
  - To exploit full performance of SSDs



(a) First - phase

Procedure of I/O Handling (2/3)

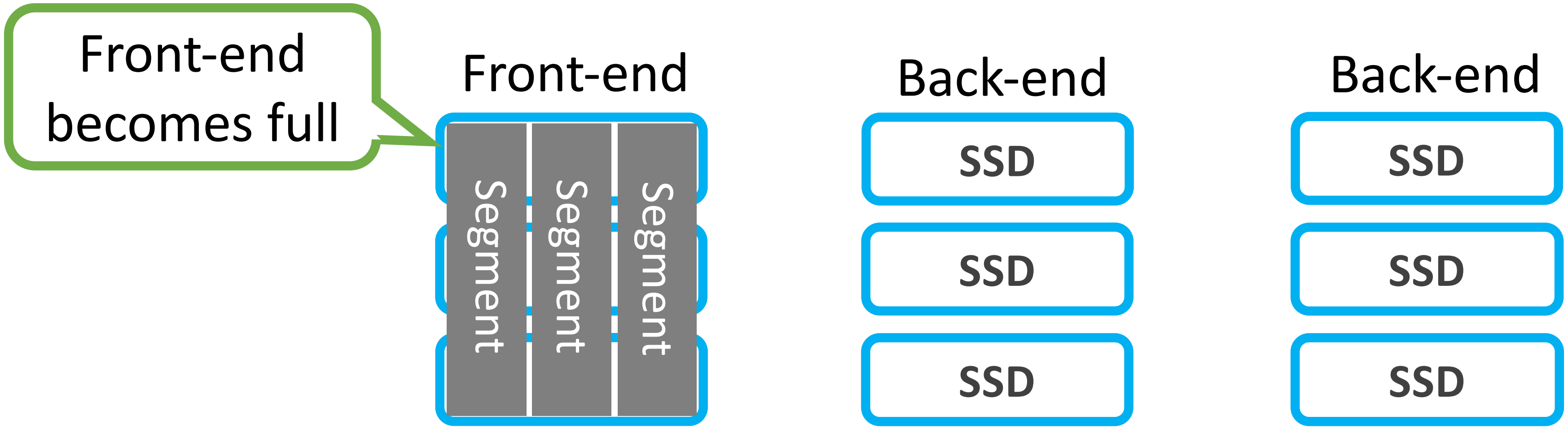
- When the front-end becomes full
  - Empty back-end becomes front-end to serve write requests
  - Full front-end becomes back-end
  - Again, new front-end serves write requests



(a) Second - phase

Procedure of I/O Handling (2/3)

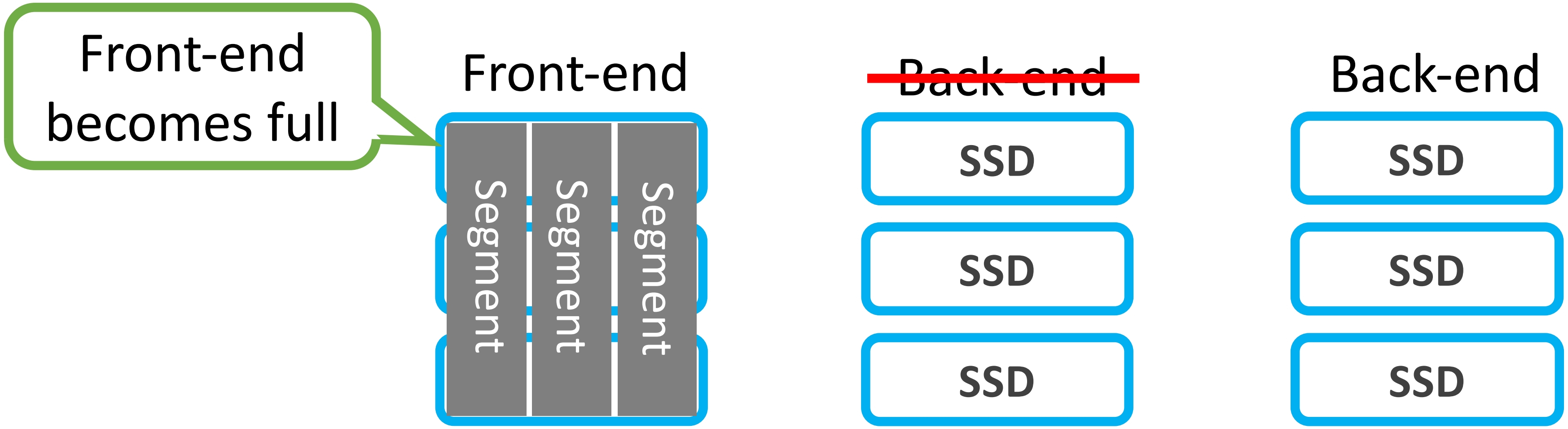
- When the front-end becomes full
  - Empty back-end becomes front-end to serve write requests
  - Full front-end becomes back-end
  - Again, new front-end serves write requests



(a) Second - phase

Procedure of I/O Handling (2/3)

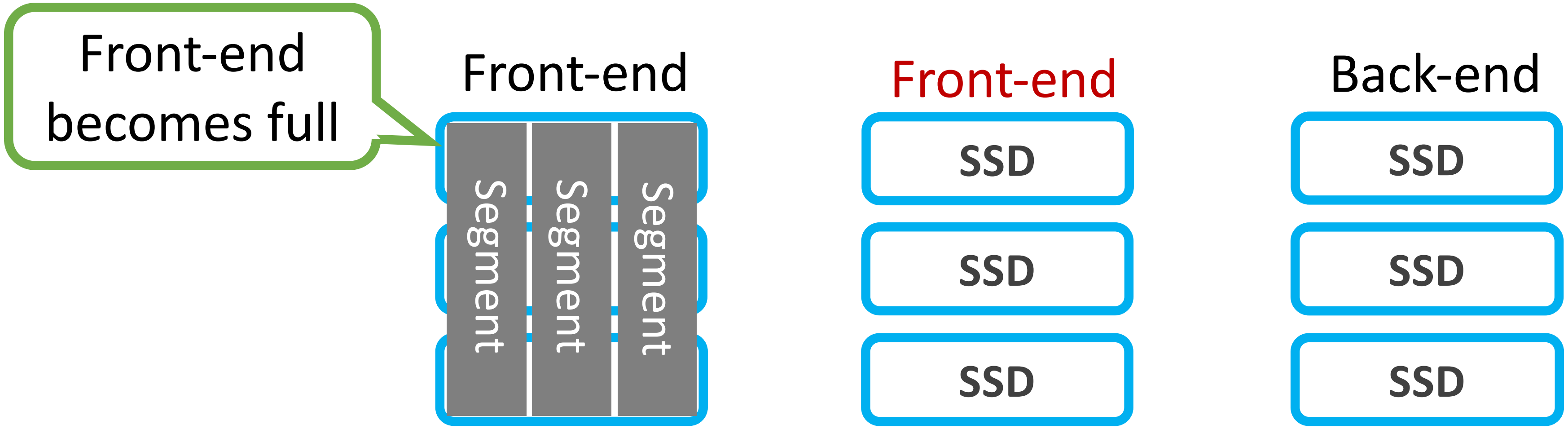
- When the front-end becomes full
  - Empty back-end becomes front-end to serve write requests
  - Full front-end becomes back-end
  - Again, new front-end serves write requests



(a) Second - phase

Procedure of I/O Handling (2/3)

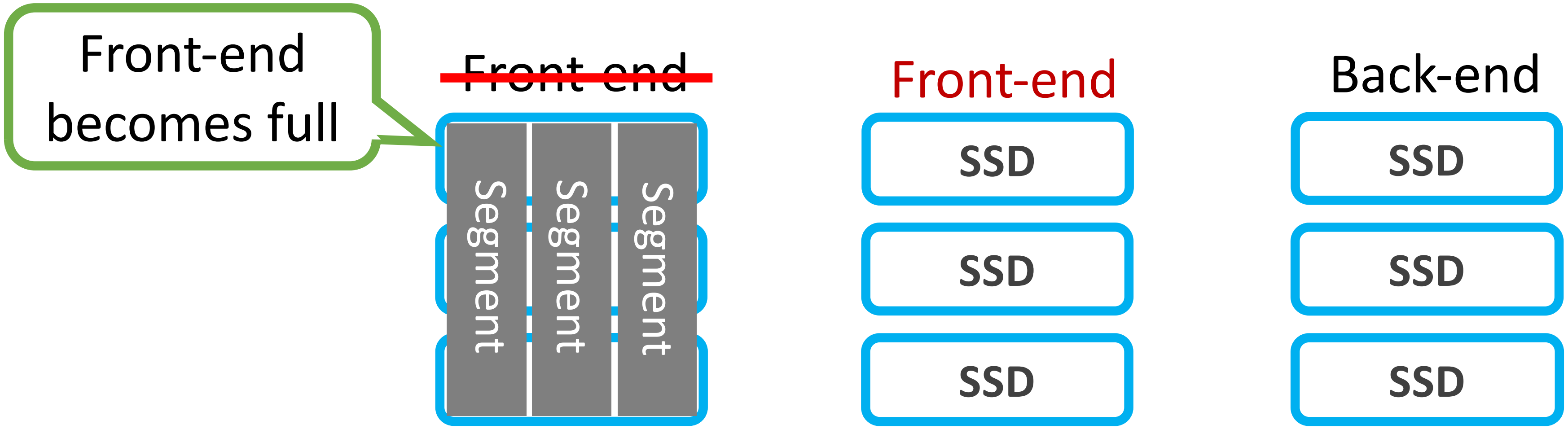
- When the front-end becomes full
  - Empty back-end becomes front-end to serve write requests
  - Full front-end becomes back-end
  - Again, new front-end serves write requests



(a) Second - phase

Procedure of I/O Handling (2/3)

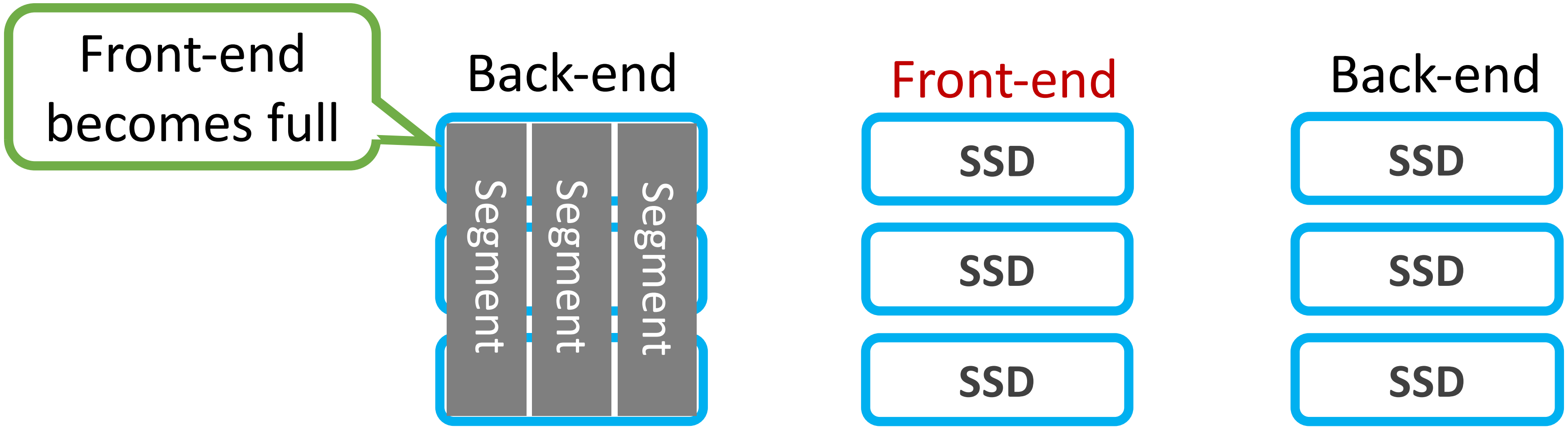
- When the front-end becomes full
  - Empty back-end becomes front-end to serve write requests
  - Full front-end becomes back-end
  - Again, new front-end serves write requests



(a) Second - phase

Procedure of I/O Handling (2/3)

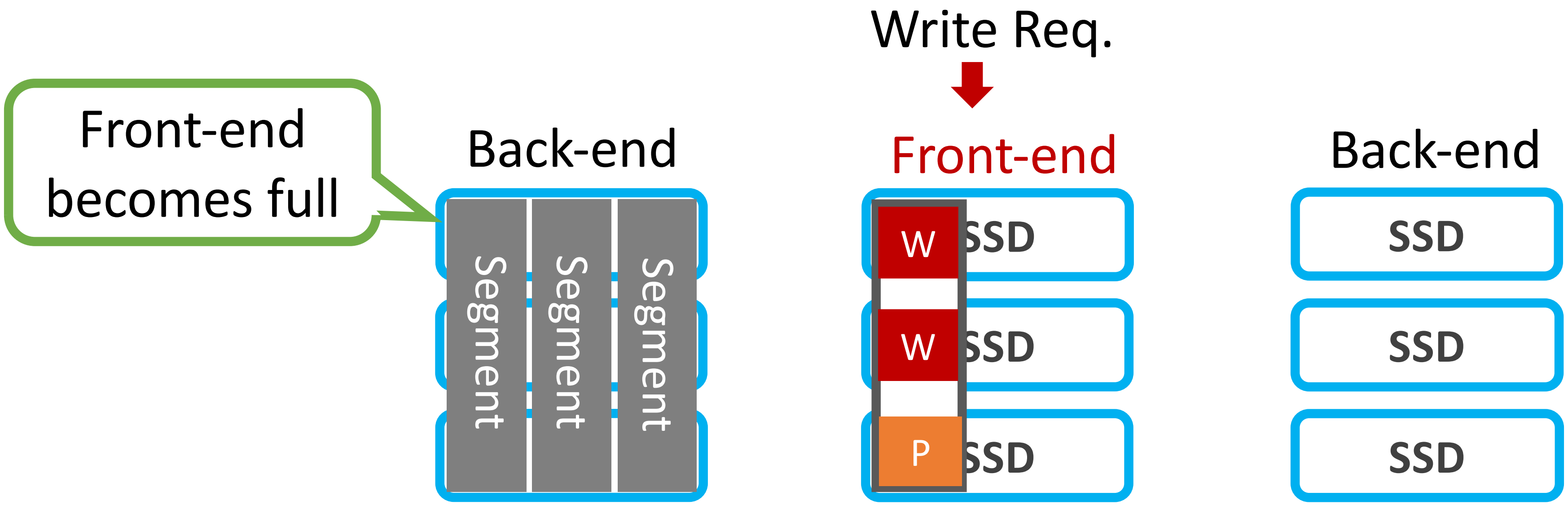
- When the front-end becomes full
  - Empty back-end becomes front-end to serve write requests
  - Full front-end becomes back-end
  - Again, new front-end serves write requests



(a) Second - phase

Procedure of I/O Handling (2/3)

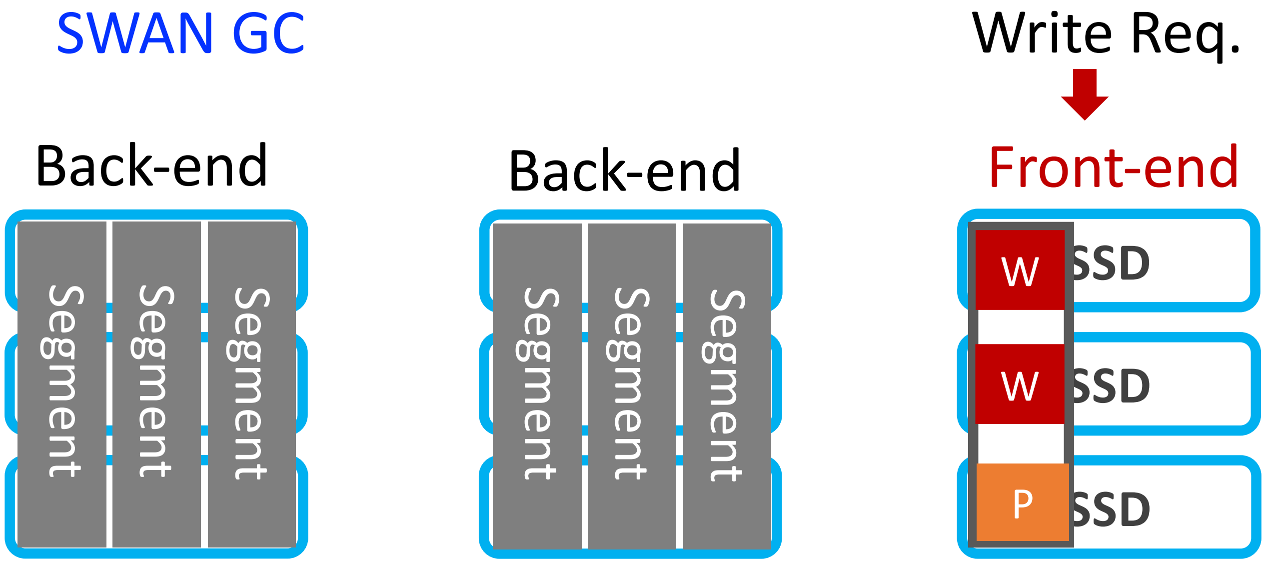
- When the front-end becomes full
  - Empty back-end becomes front-end to serve write requests
  - Full front-end becomes back-end
  - Again, new front-end serves write requests



(a) Second - phase

Procedure of I/O Handling (3/3)

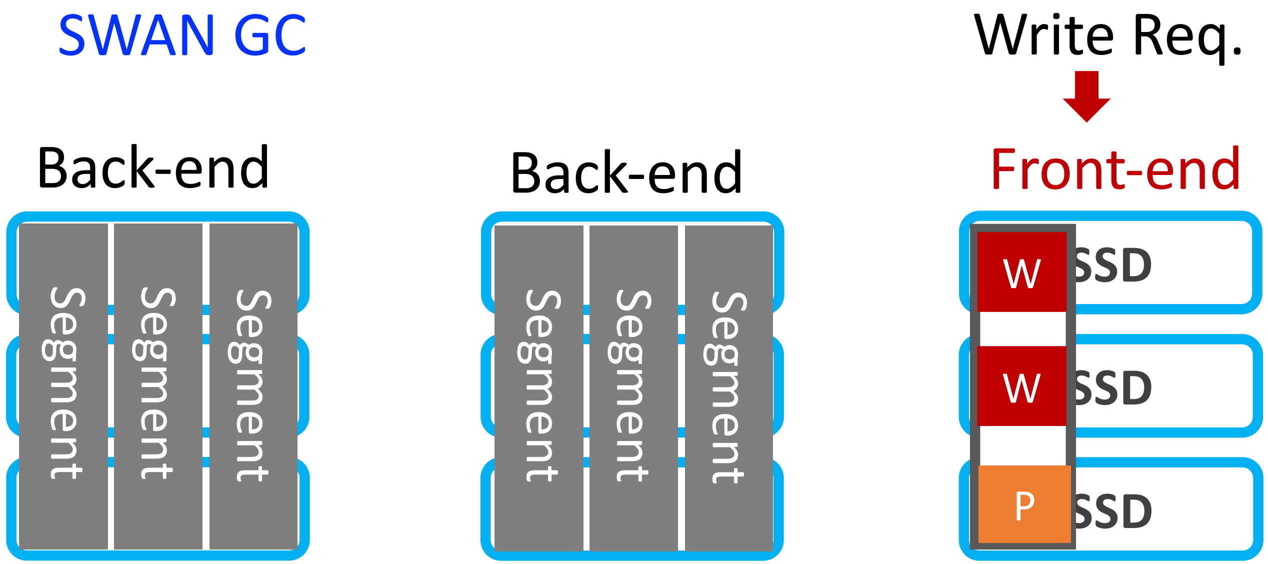
- When there is no more empty back-end
  - SWAN's GC is triggered to make free space
  - SWAN chooses a victim segment from one of the back-ends
  - SWAN writes valid blocks within the chosen back-end
  - Finally, the victim segment is trimmed



(a) Third - phase

Procedure of I/O Handling (3/3)

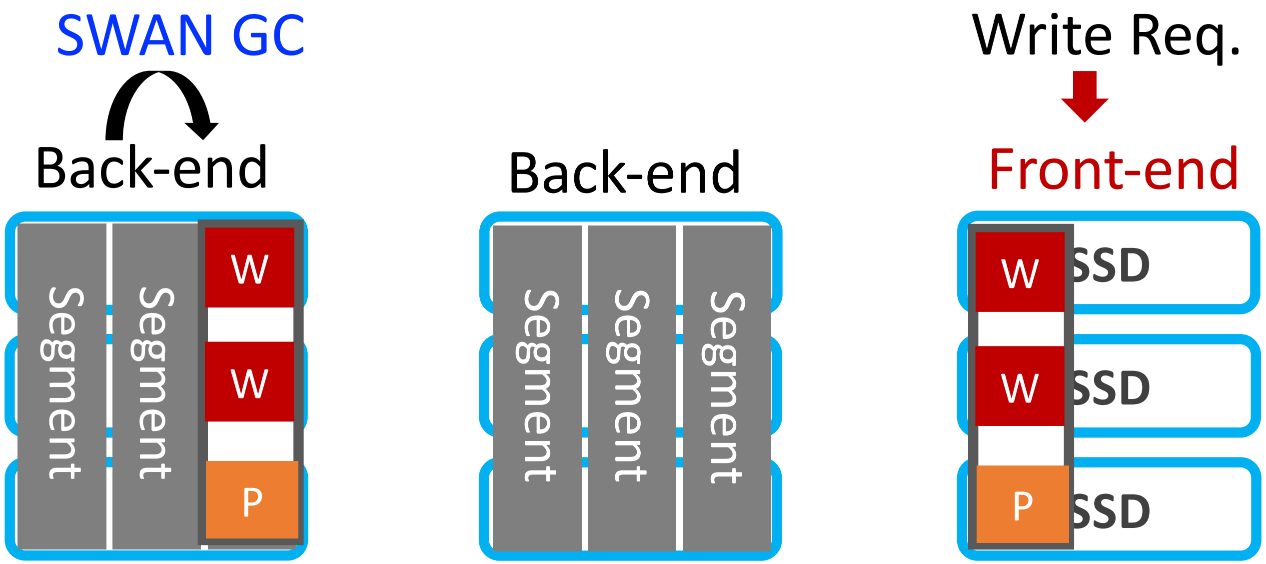
- When there is no more empty back-end
  - SWAN's GC is triggered to make free space
  - SWAN chooses a victim segment from one of the back-ends
  - SWAN writes valid blocks within the chosen back-end
  - Finally, the victim segment is trimmed



(a) Third - phase

Procedure of I/O Handling (3/3)

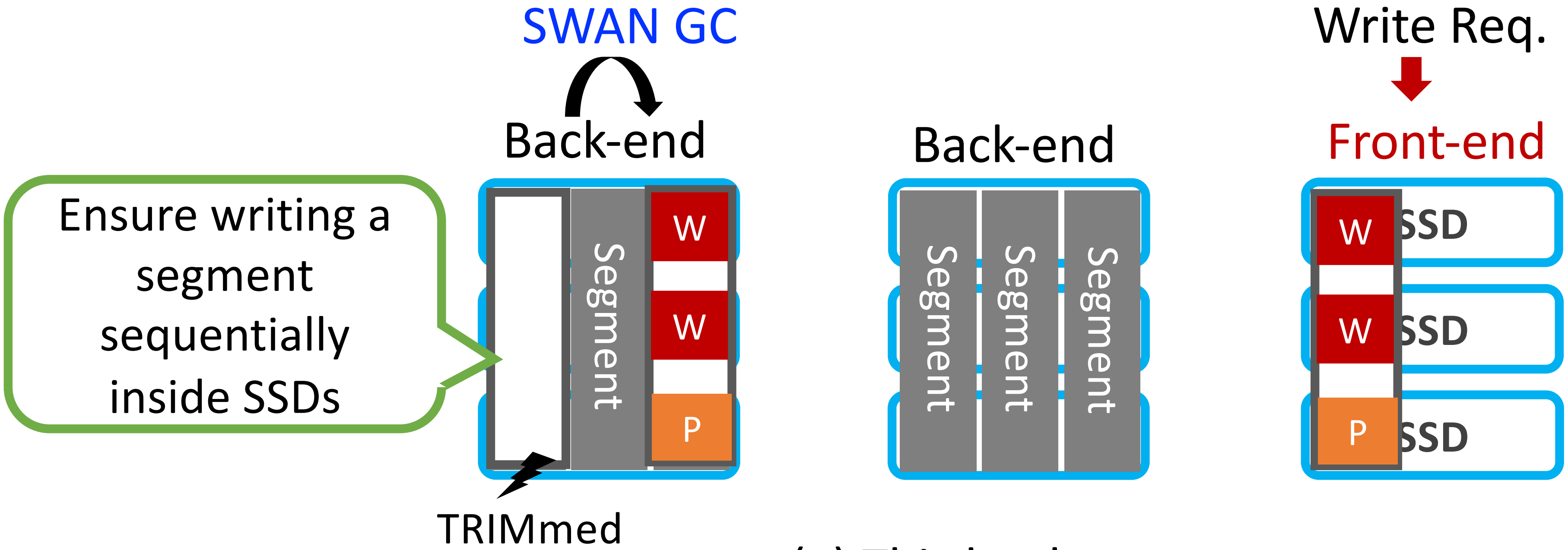
- When there is no more empty back-end
  - SWAN's GC is triggered to make free space
  - SWAN chooses a victim segment from one of the back-ends
  - SWAN writes valid blocks within the chosen back-end
  - Finally, the victim segment is trimmed



(a) Third - phase

Procedure of I/O Handling (3/3)

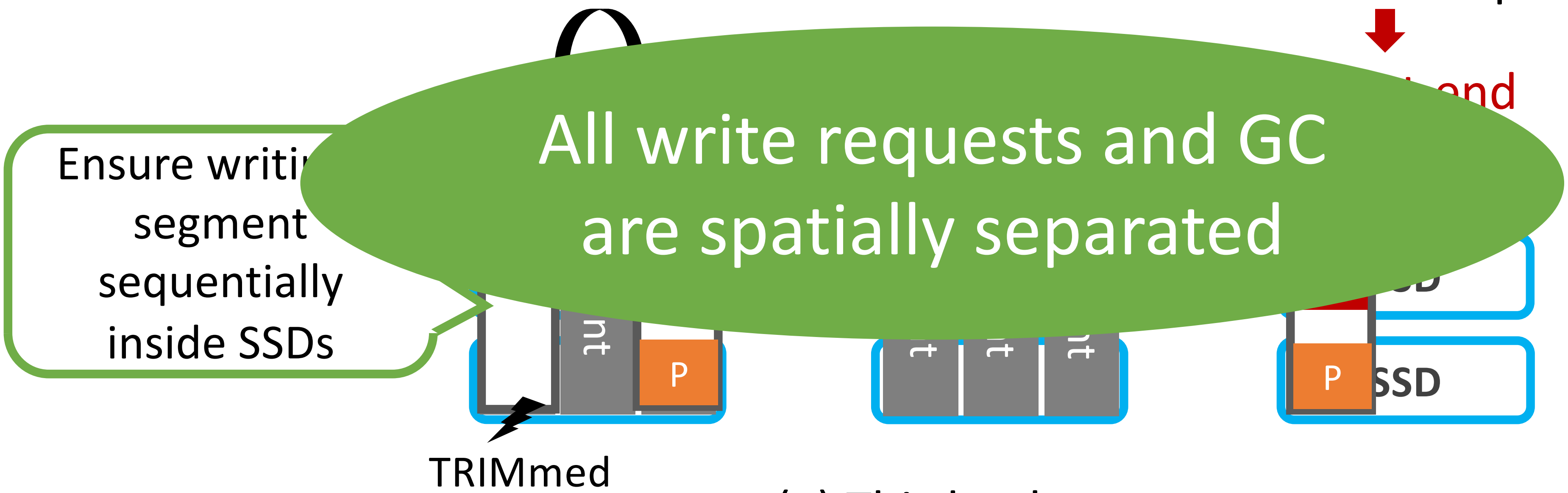
- When there is no more empty back-end
  - SWAN's GC is triggered to make free space
  - SWAN chooses a victim segment from one of the back-ends
  - SWAN writes valid blocks within the chosen back-end
  - Finally, the victim segment is trimmed



(a) Third - phase

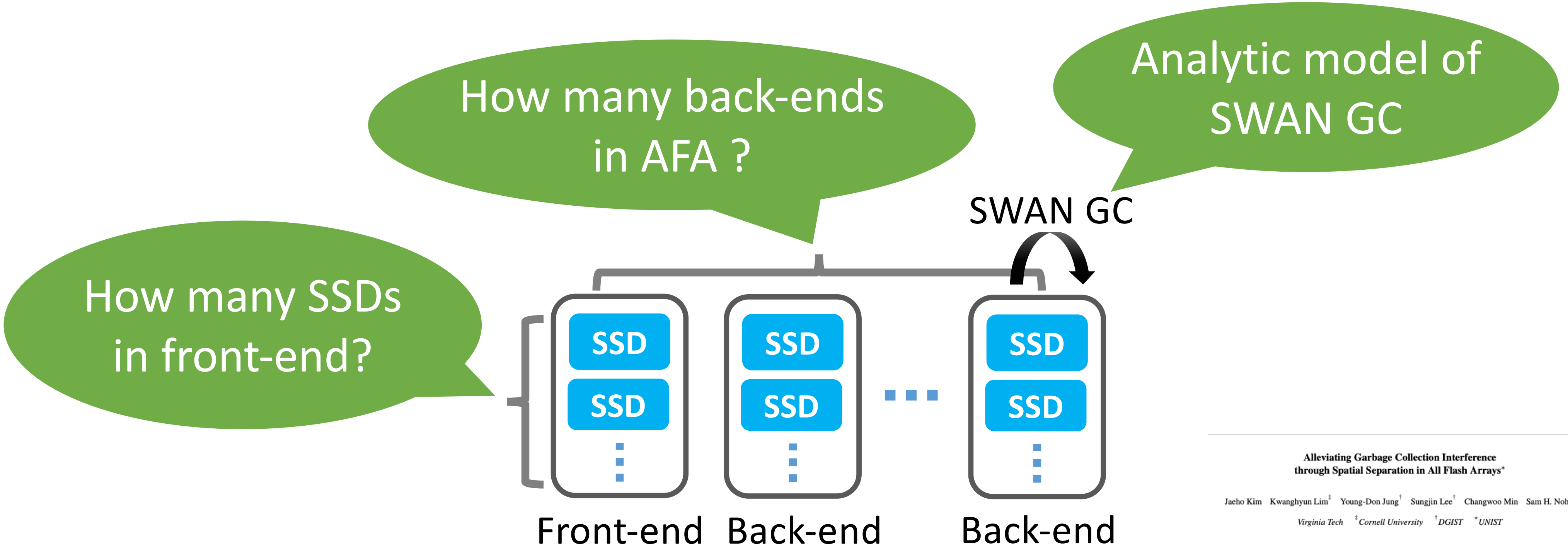
Procedure of I/O Handling (3/3)

- When there is no more empty back-end
  - SWAN's GC is triggered to make free space
  - SWAN chooses a victim segment from one of the back-ends
  - SWAN writes valid blocks within the chosen back-end
  - Finally, the victim segment is trimmed

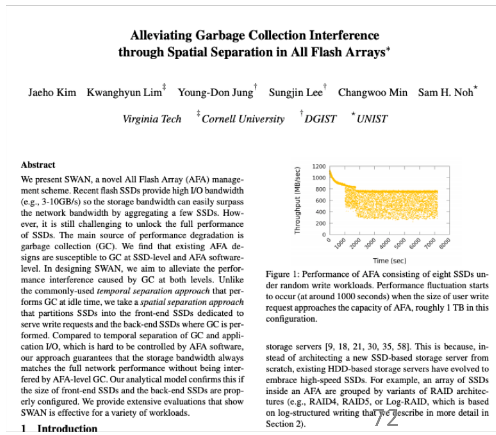


(a) Third - phase

Feasibility Analysis of SWAN



Please refer to our paper for details!



Evaluation Setup

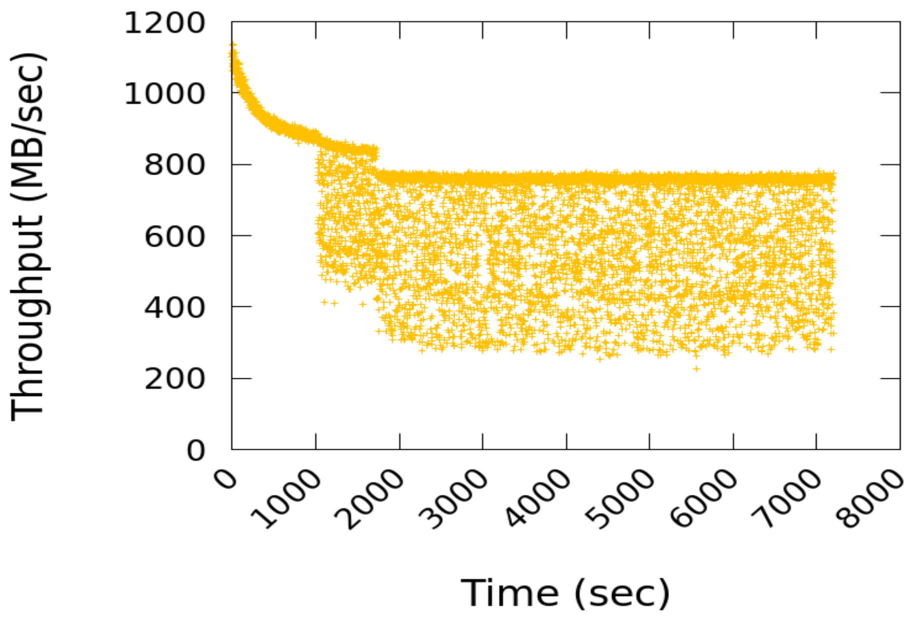
- Environment
  - Dell R730 server with Xeon CPUs and 64GB DRAM
  - Up to 9 SATA SSDs are used (up to 1TB capacity)
  - Open channel SSD for monitoring internal activity of an SSD
- Target Configurations
  - RAID0/4: Traditional RAID
  - Log-RAID0/4: Log-based RAID
  - SWAN0/4: Our solution

No parity	1 parity per stripe
RAID0	RAID4
Log-RAID0	Log-RAID4
SWAN0	SWAN4

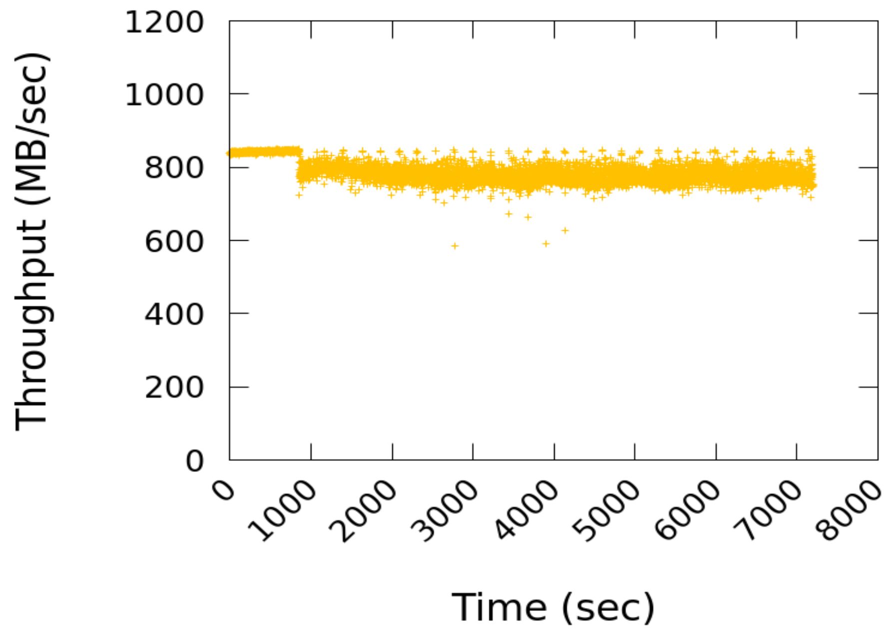
- Workloads
  - Micro-benchmark: Random write request
  - YCSB C benchmark

Please refer to paper for more results!

Random Write Requests for 2 Hours (8KB Sized Req.)

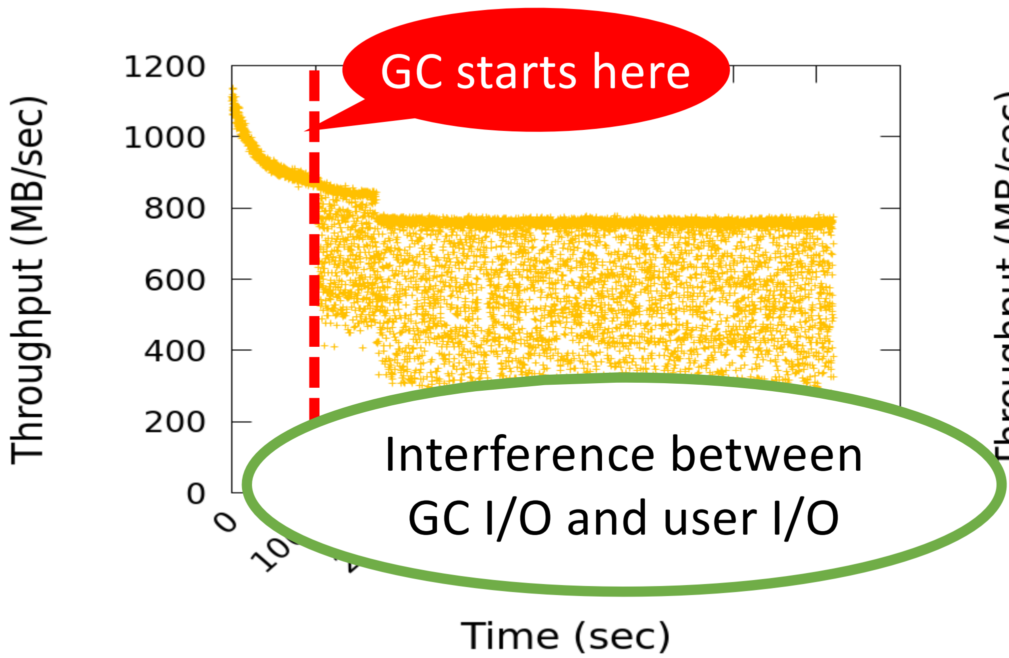


Log-RAID0

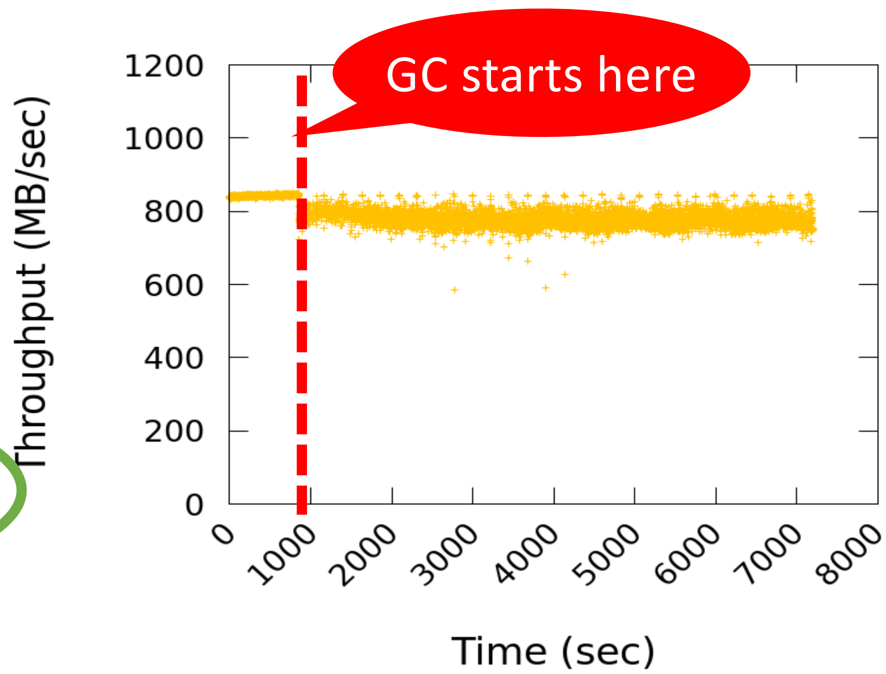


SWAN0

Random Write Requests for 2 Hours (8KB Sized Req.)

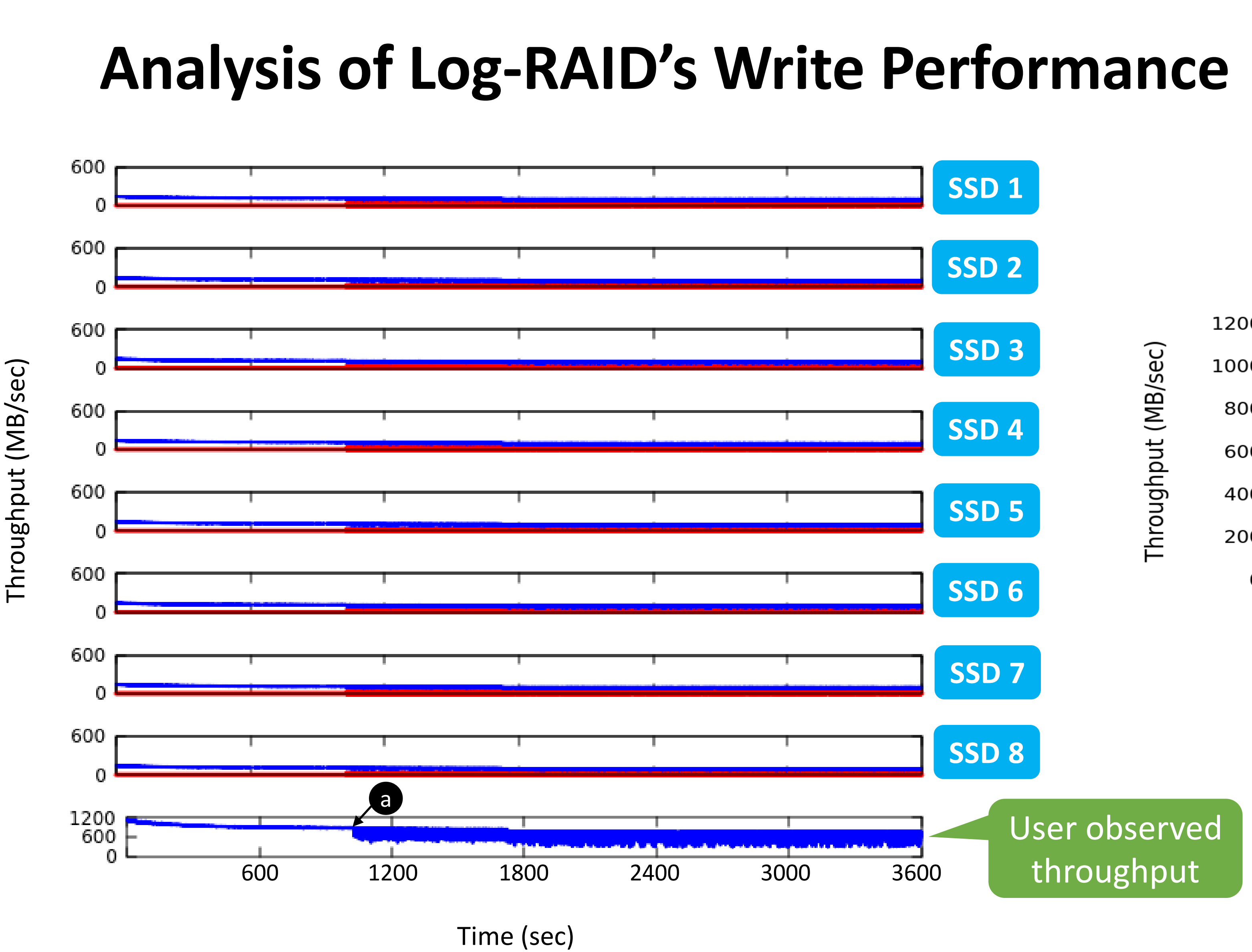


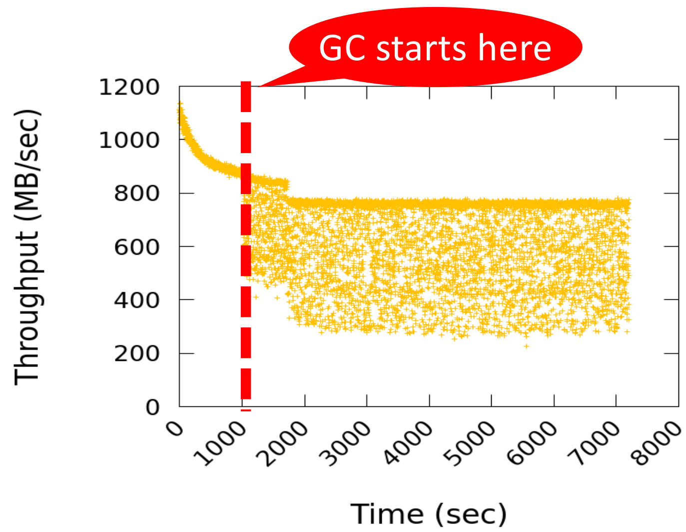
Log-RAID0



SWAN0

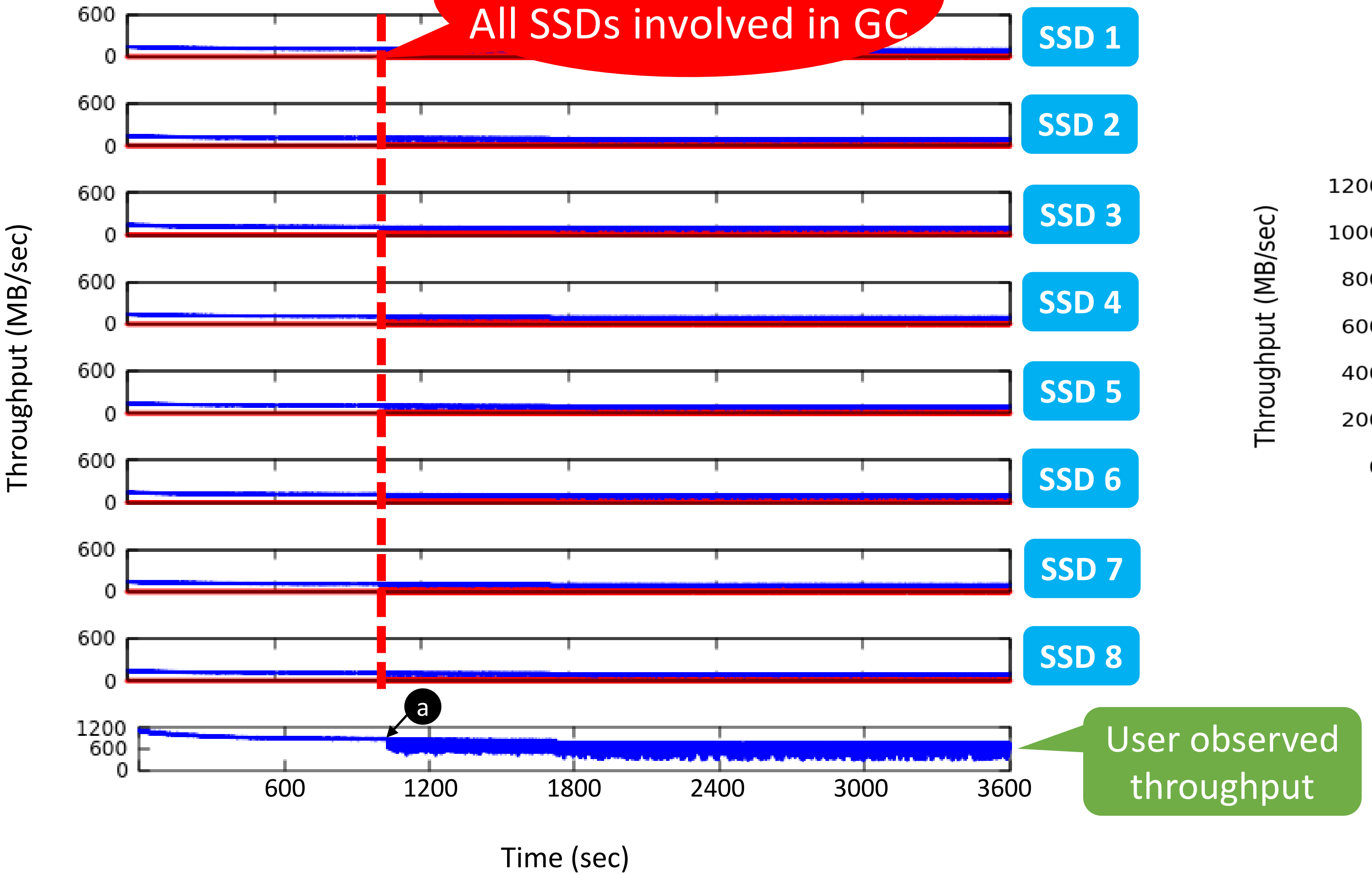
Analysis of Log-RAID's Write Performance

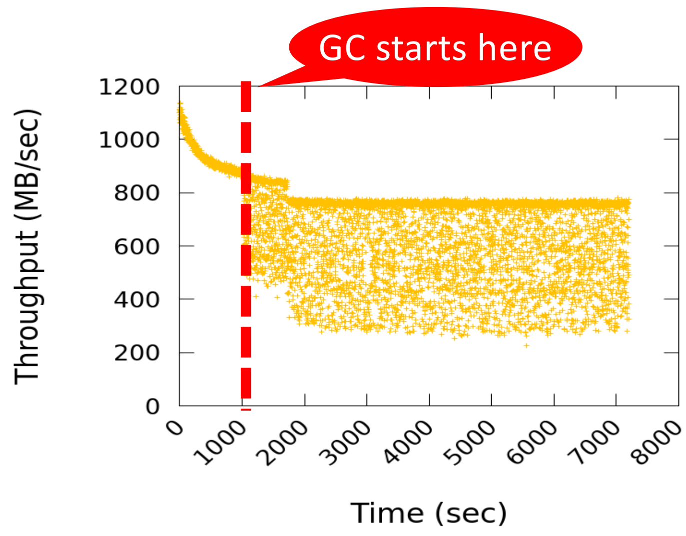




Log-RAID0

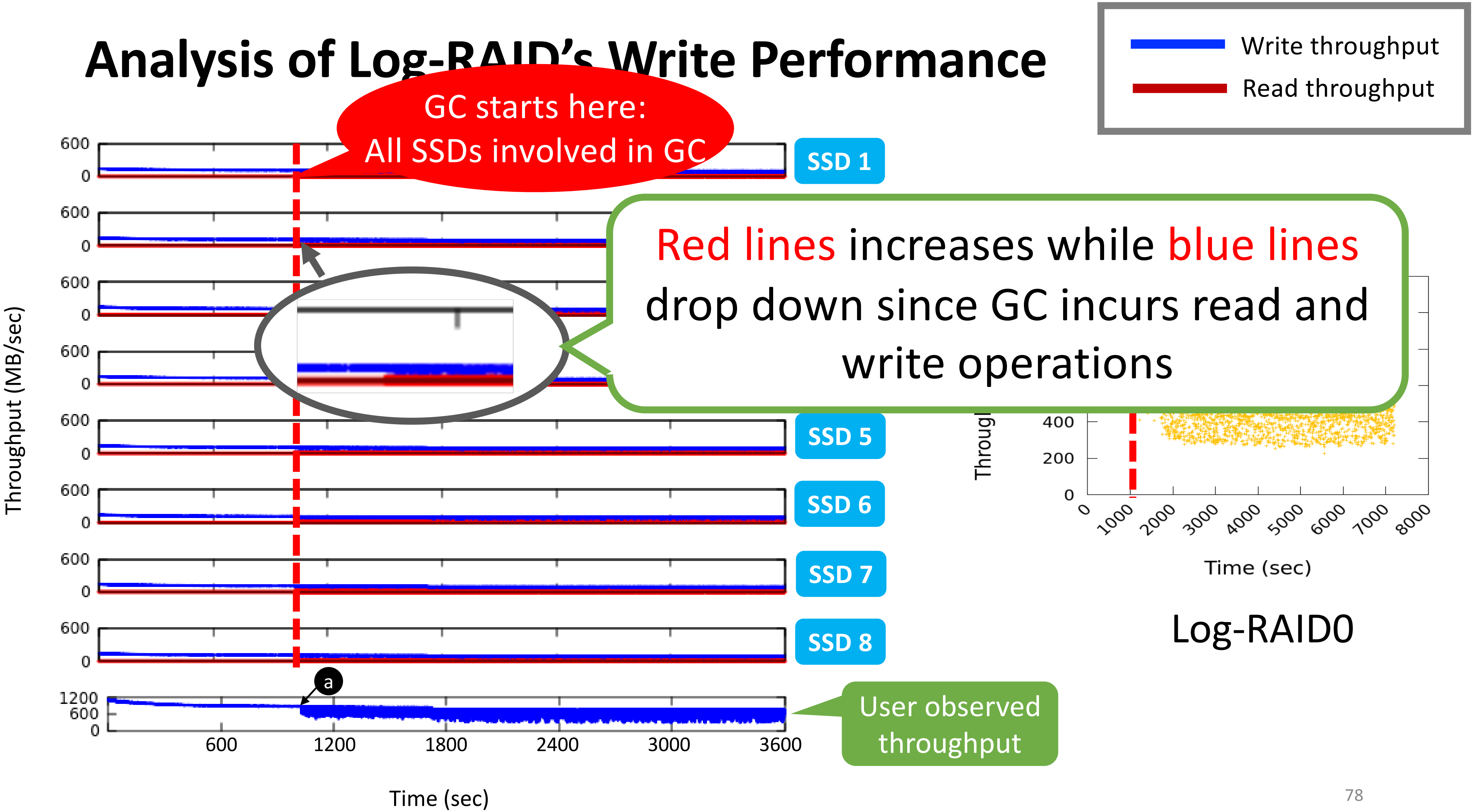
Analysis of Log-RAID's Write Performance



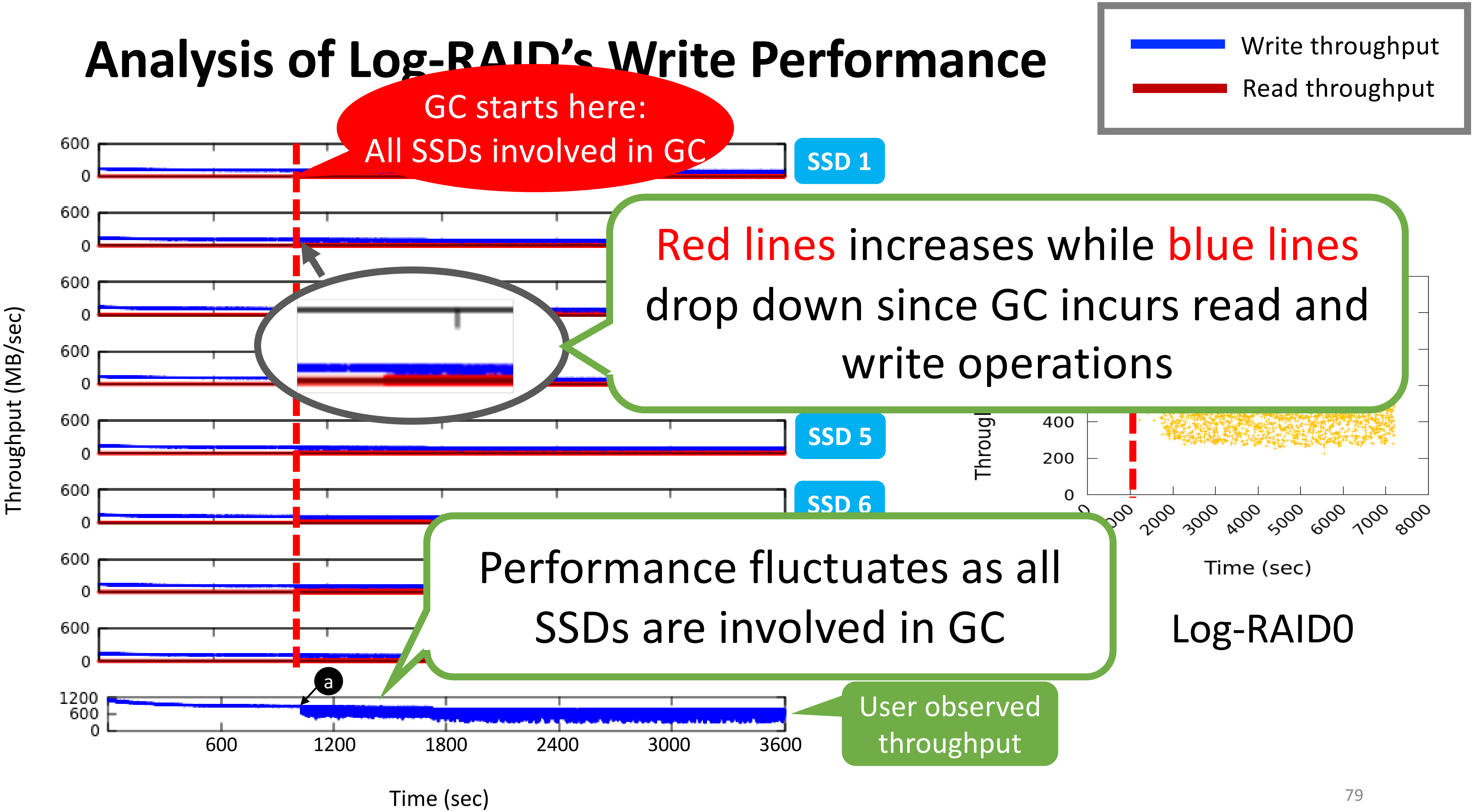


Log-RAID0

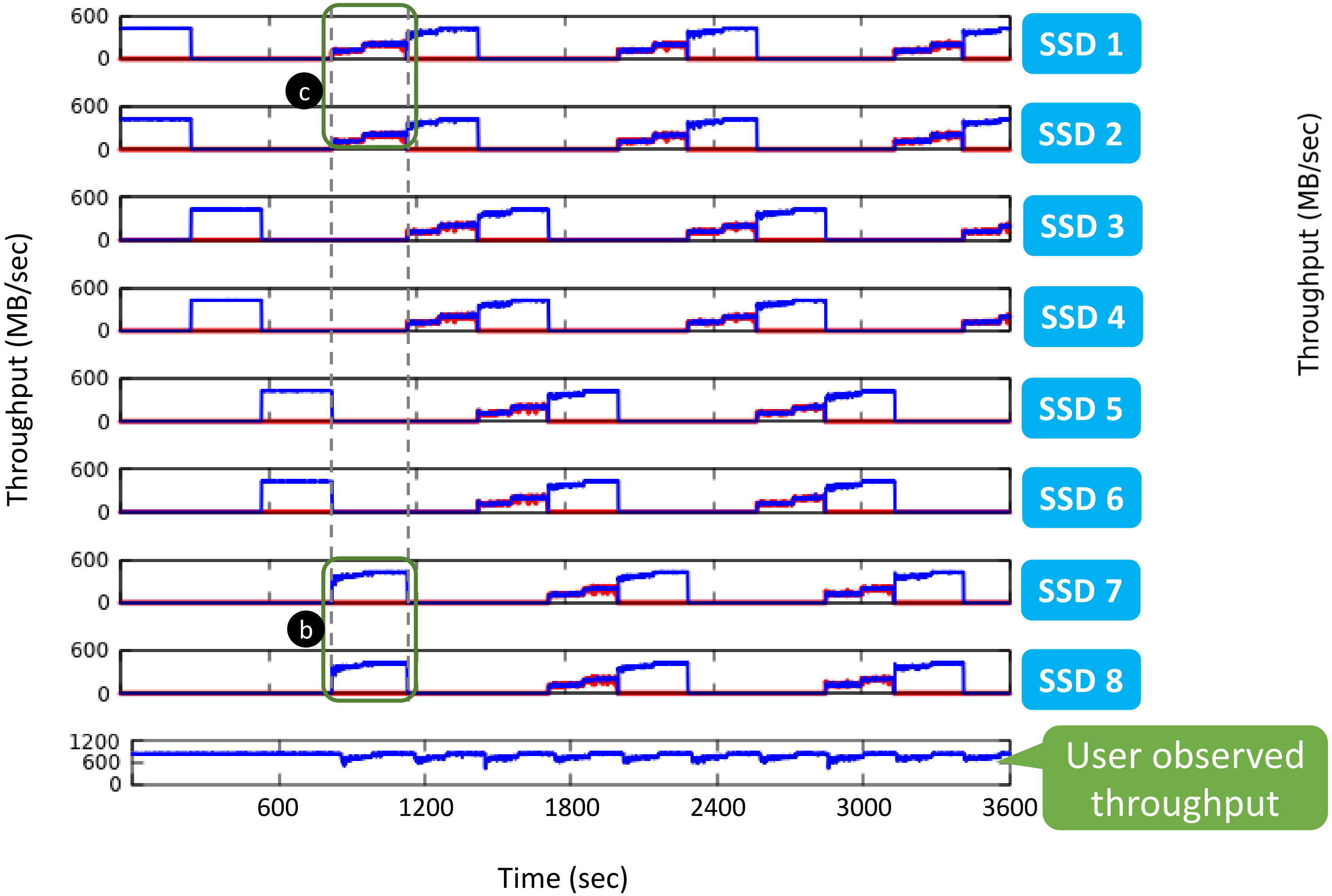
Analysis of Log-RAID's Write Performance

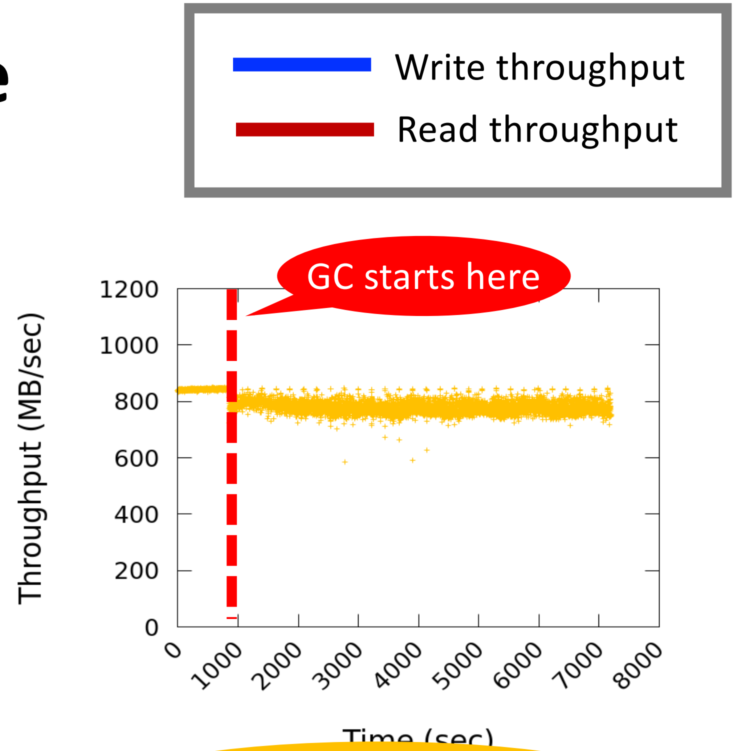


Analysis of Log-RAID's Write Performance



Analysis of SWAN's Write Performance

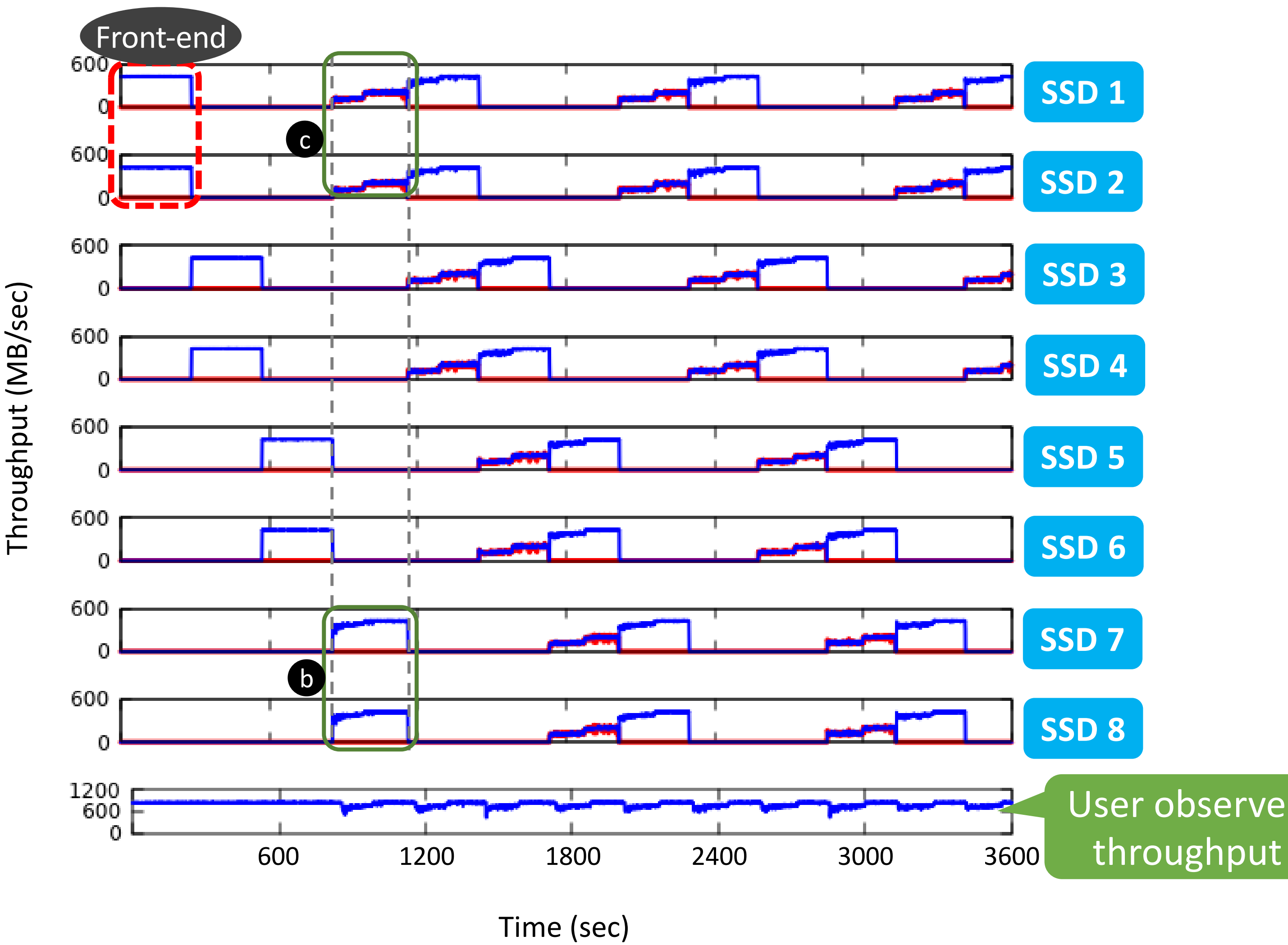


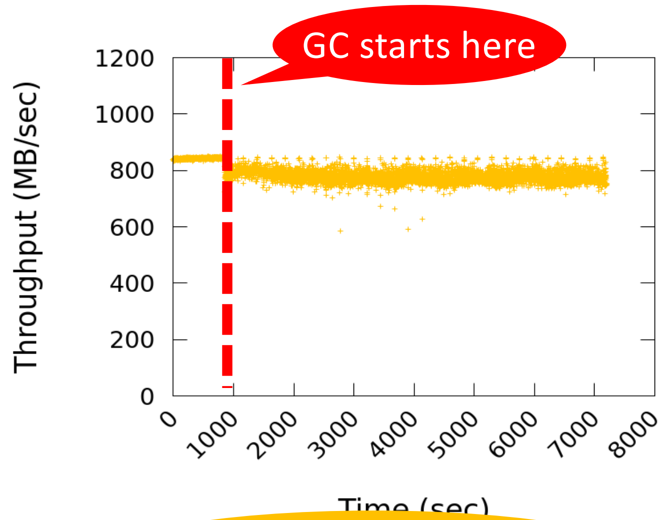


Configuration

- SWAN has 1 front-end and 4 back-ends
- Front/back-ends consists of 2 SSDs

Analysis of SWAN's Write Performance

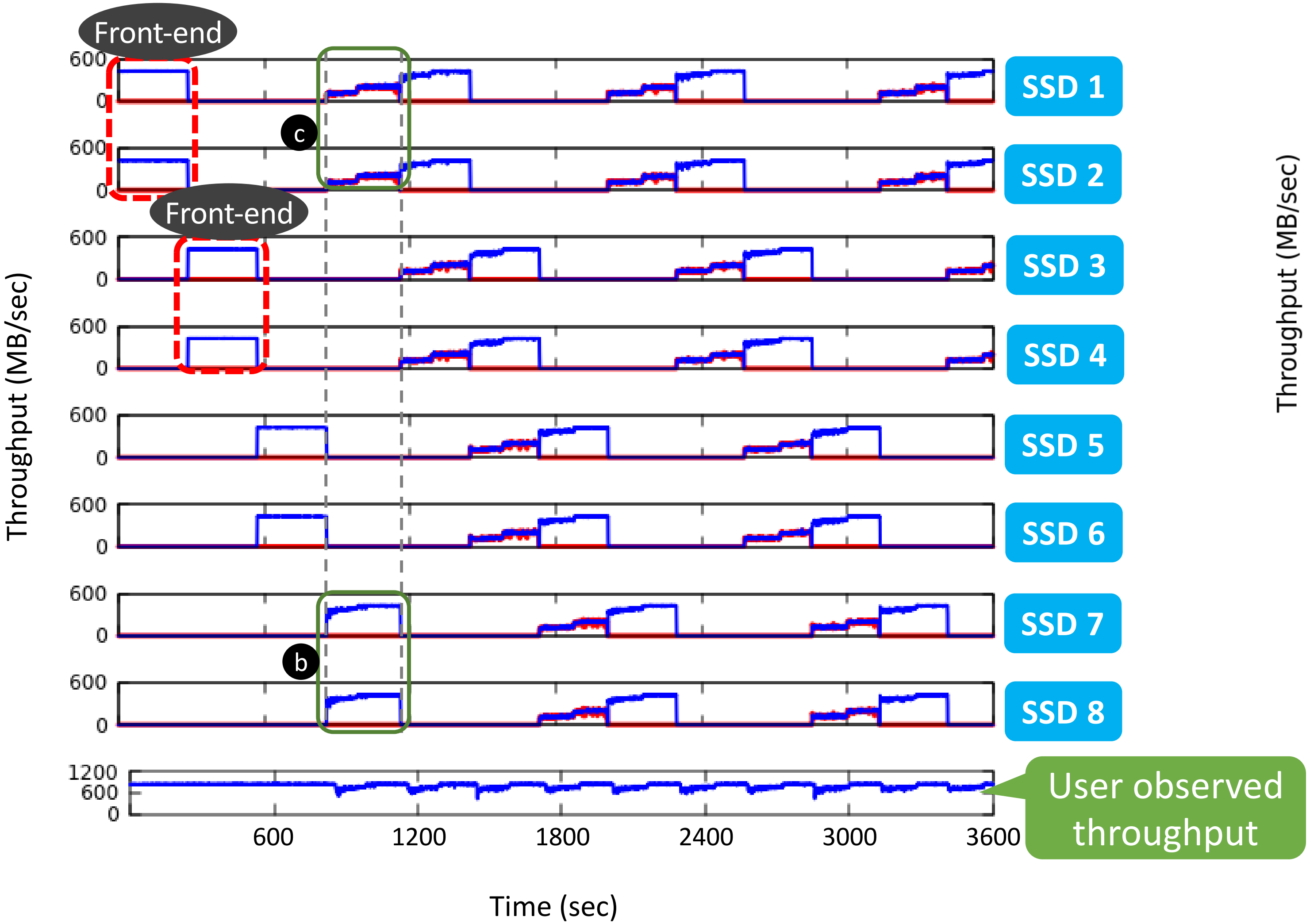


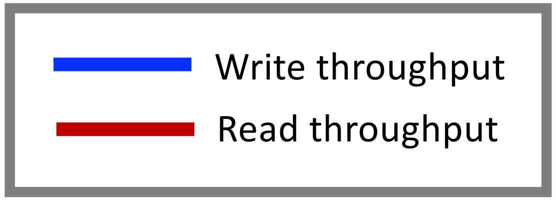


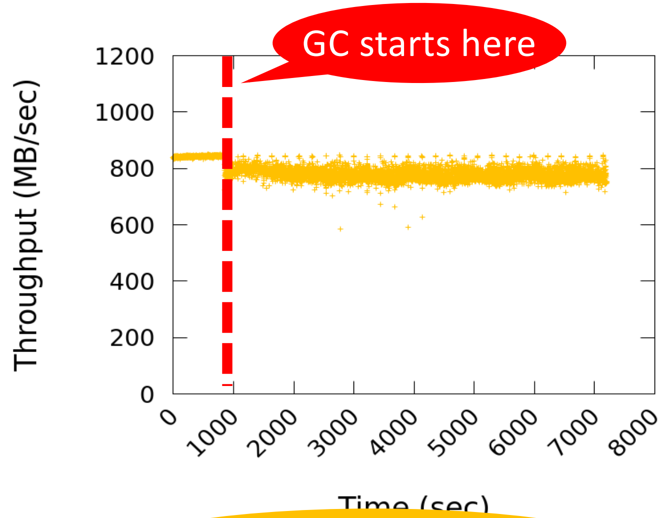
Configuration

- SWAN has 1 front-end and 4 back-ends
- Front/back-ends consists of 2 SSDs

Analysis of SWAN's Write Performance



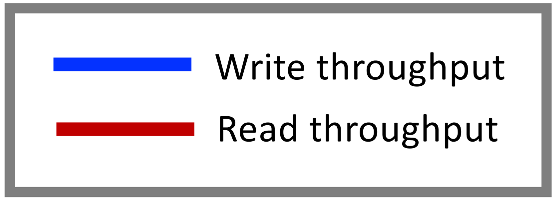


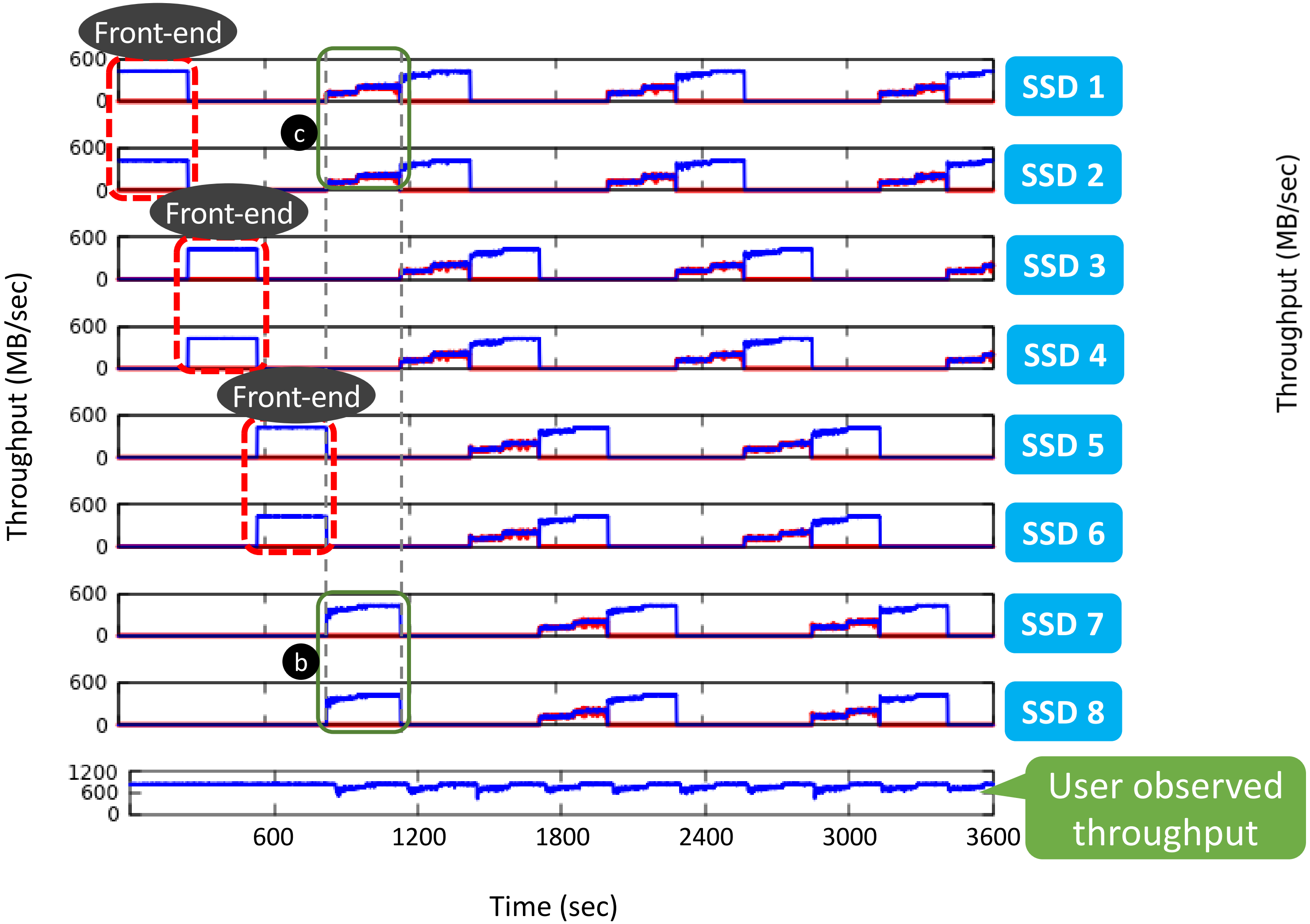


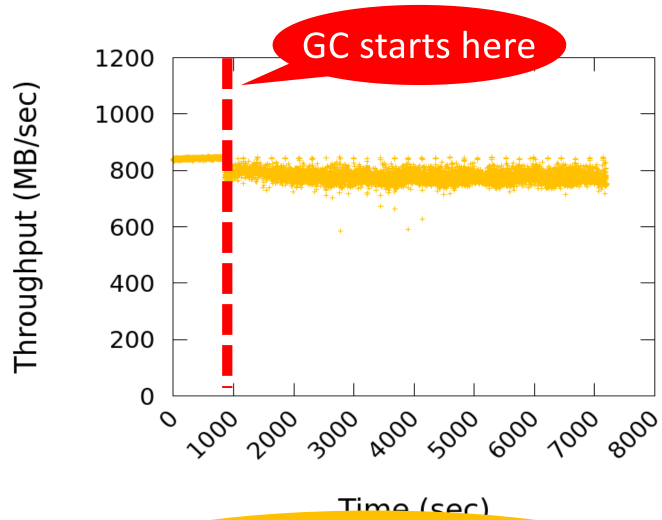
Configuration

- SWAN has 1 front-end and 4 back-ends
- Front/back-ends consists of 2 SSDs

Analysis of SWAN's Write Performance



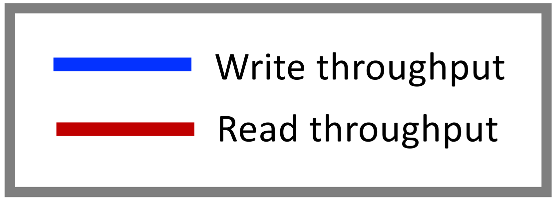


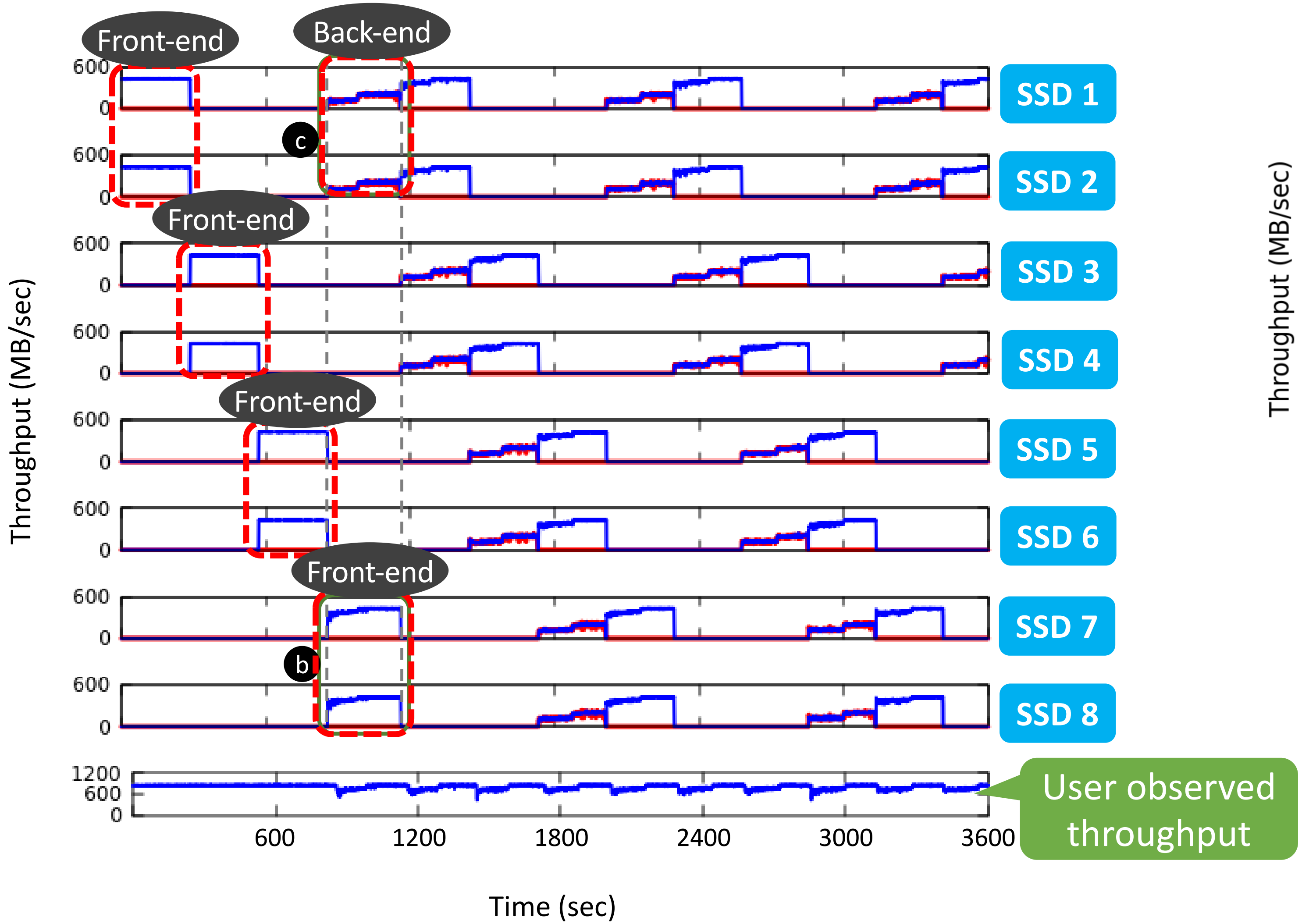


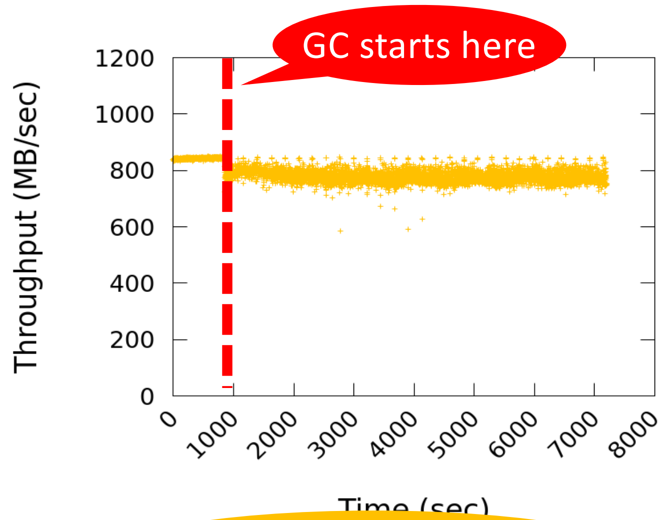
Configuration

- SWAN has 1 front-end and 4 back-ends
- Front/back-ends consists of 2 SSDs

Analysis of SWAN's Write Performance



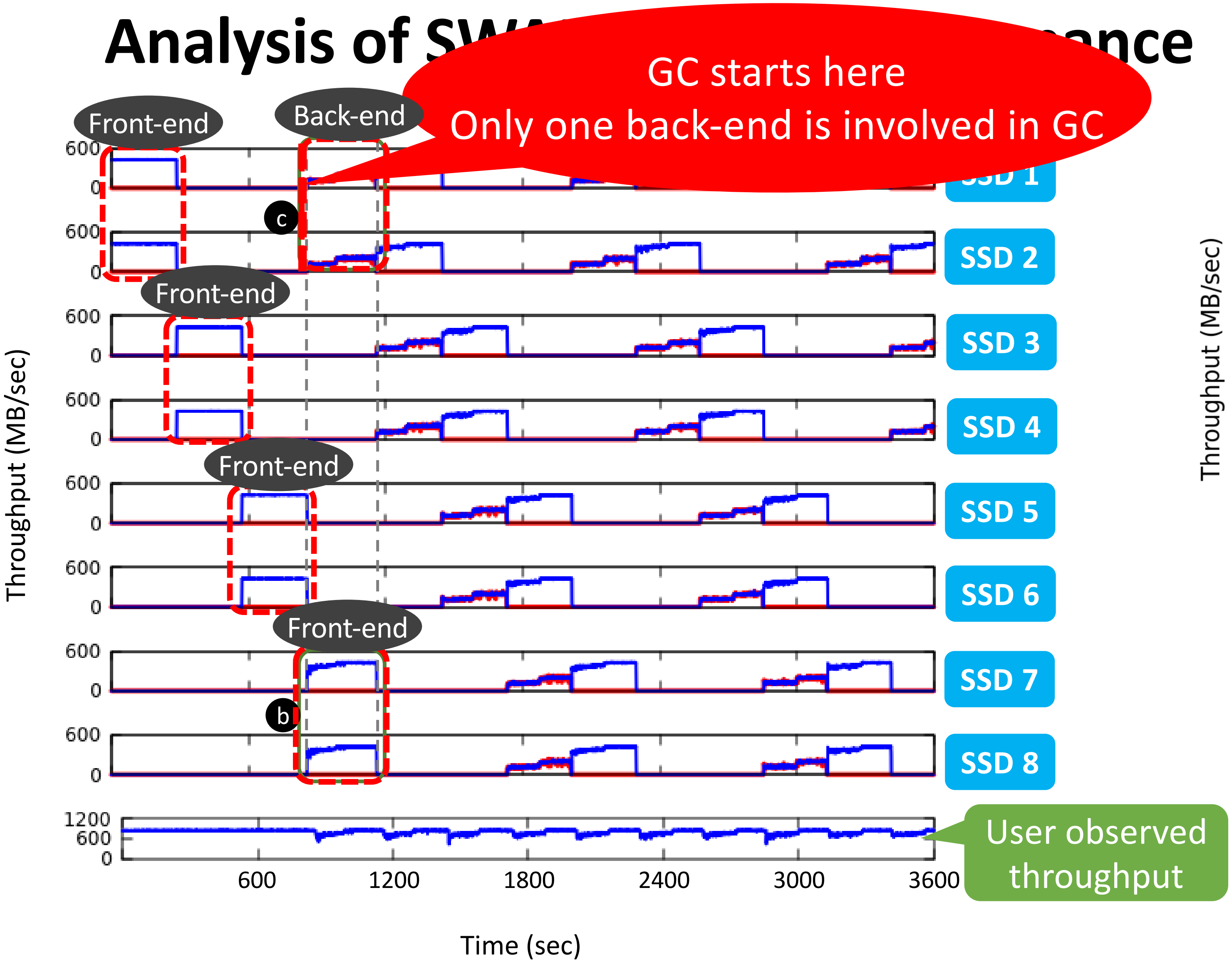


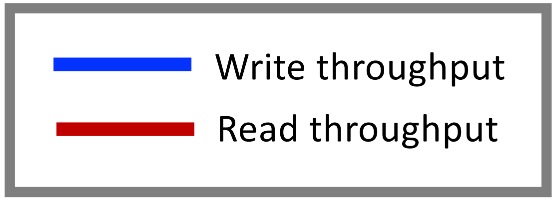


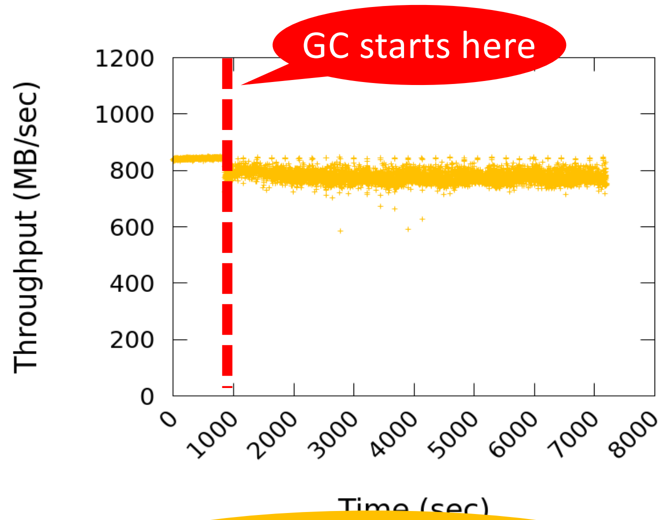
Configuration

- SWAN has 1 front-end and 4 back-ends
- Front/back-ends consists of 2 SSDs

Analysis of SWA... ...ance



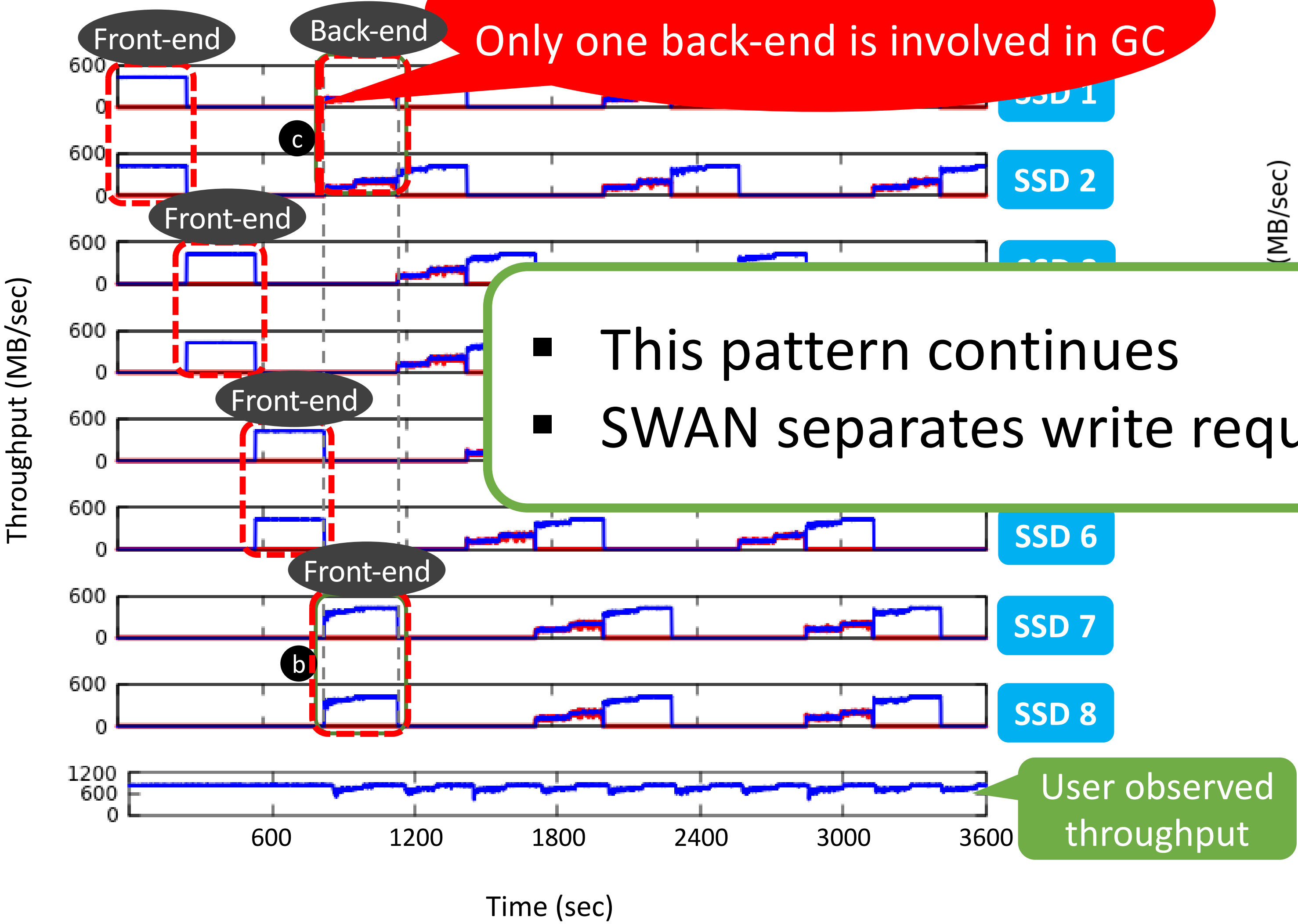


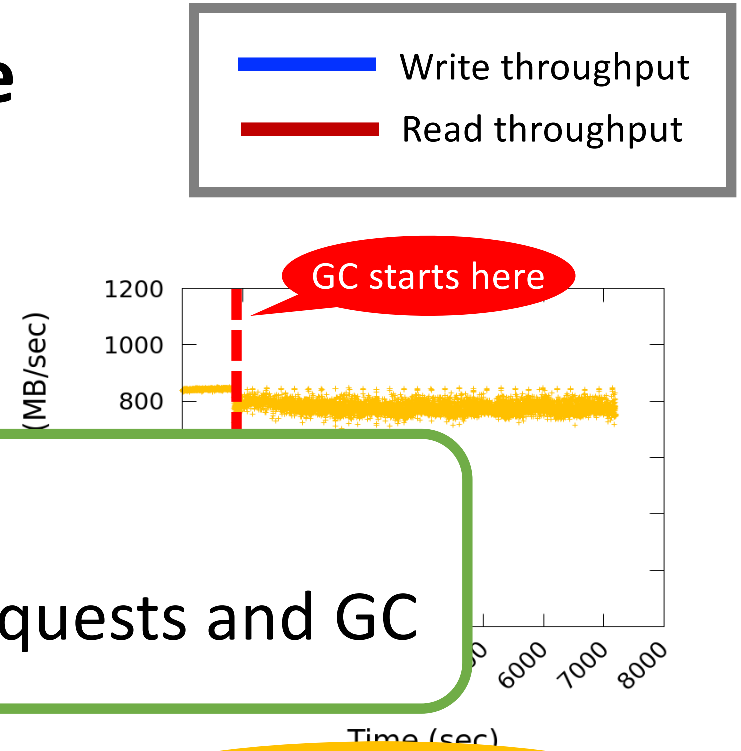


Configuration

- SWAN has 1 front-end and 4 back-ends
- Front/back-ends consists of 2 SSDs

Analysis of SW… ance





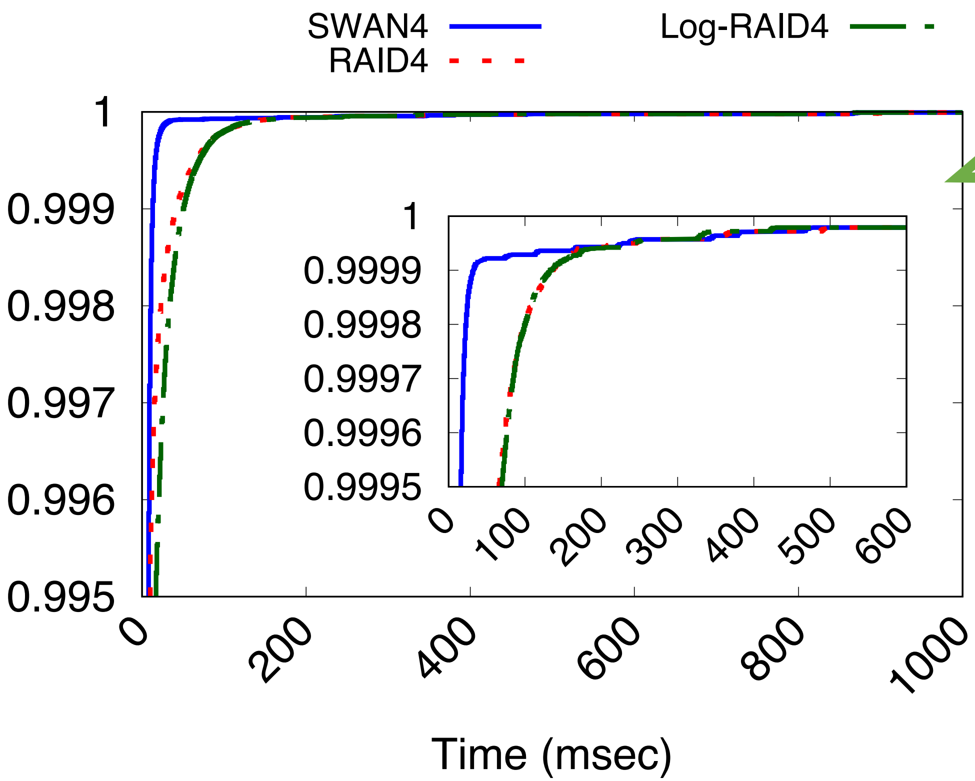
- This pattern continues
- SWAN separates write requests and GC

Configuration

- SWAN has 1 front-end and 4 back-ends
- Front/back-ends consists of 2 SSDs

Read Tail Latency for YCSB-C

- SWAN4 shows the shortest read tail latency
- RAID4 and Log-RAID4 suffers long tail latency



Spatial separation is effective for handling read requests as well

Benefits with Simpler SSDs

- SWAN can saves cost and power consumption w/o compromising performance by adopting simpler SSDs
  1) Smaller DRAM size
  2) Smaller over-provisioning space (OPS)
  3) Block or segment level FTL instead of page-level FTL

SWAN sequentially writes data to segments and TRIMs a large chunk of data in the same segment at once

Conclusion

- Provide full write performance of an array of SSDs up to network bandwidth limit
- Alleviate GC interference through separation of I/O induced by application and GC of All Flash Array
- Introduce an efficient way to manage SSDs in All Flash Array

Thanks for attention!

Q&A

Backup slides

Handling Read Requests in SWAN

- Recent updated data might be served at page cache or buffer

- Falling in front-end
  - Give the highest priority to read requests

- Falling in GC back-end
  - Preempt GC then serve read requests

- Falling in idle back-ends
  - Serve immediately read requests

GC overhead inside SSDs

- GC overhead should be very low inside SSDs
  - SWAN writes all the data in a segment-based append-only manner
  - Then, SWAN gives TRIMs to ensure writing a segment sequentially inside SSDs

Previous Solutions

Solutions	Write Strategy	How Separate User & GC I/O	Disk Organization
Harmonia [MSST'11]	In-place write	Temporal (Idle time)	RAID-0
HPDA [IPDPS'10]	In-place write	Temporal	RAID-4
GC-Steering [IPDPS'18]	In-place write	Temporal	RAID-4/5
SOFA [SYSTOR'14]	Log write	Temporal	Log-RAID
SALSA [MASCOTS'18]	Log write	Temporal	Log-RAID
Purity [SIGMOD'15]	Log write	Temporal	Log-RAID
SWAN (Proposed)	**Log write**	**Spatial**	**2D Array**