Understanding the Robustness of SSDs under Power Fault

Mai Zheng
Feng Qin
The Ohio State University

Joseph Tucek
Mark Lillibridge
HP Labs

Solid-State Drives (SSDs)

- a “truly revolutionary and disruptive” technology

- Great performance 🙂
- Low power consumption 🙂

Solid-State Drives (SSDs)

- a “truly revolutionary and disruptive” technology

- Great performance
- Low power consumption
- *** behavior in adverse conditions ?

Power Faults

- a threat never gone

Power Faults

- a threat never gone

Jan. 2013:“ A POWER OUTAGE at a key New Jersey data center ...”

Jul. 2012:“... human error was responsible for a data center POWER OUTAGE ...”

Jul. 2012:“POWER OUTAGE Hits London Data Center ...”

Jun. 2012:“Amazon Data Center LOSES POWER During Storm …”

Aug, 2011:“Colocation provider Colo4 experienced a POWER OUTAGE …”

Nov. 2010:“About 3,000 servers at Montreal web host iWeb experienced an OUTAGE …”

May 2010:“Car Crash Triggers Amazon POWER OUTAGE …”

Power Faults

- a threat never gone

Jan. 2013:" A POWER OUTAGE at a key New Jersey data center ..."

Jul. 2012:"... human error was responsible for a data center POWER OUTAGE ..."

Jul. 2012:"POWER OUTAGE Hits London Data Center ..."

Jun. 2012:"Amazon Data Center LOSES POWER During Storm ..."

Aug, 2011:"Colocation provider Colo4 experienced a POWER OUTAGE ..."

Nov. 2010:"About 3,000 servers at Montreal web host iWeb experienced an OUTAGE ..."

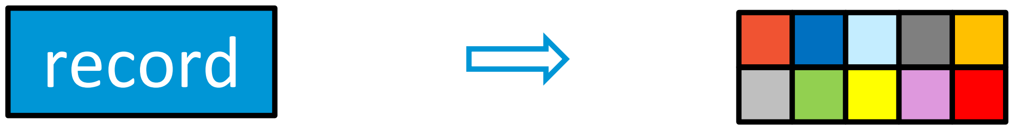
May 2010:"Car Crash Triggers Amazon POWER OUTAGE ..."

Potential Failures

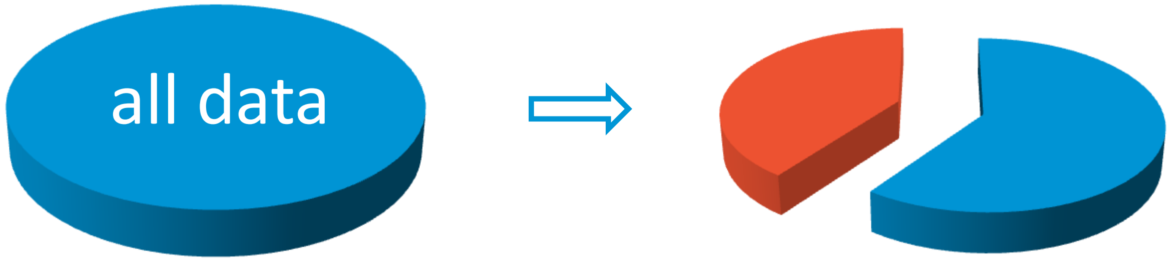
Simple Failures

before power fault after power fault

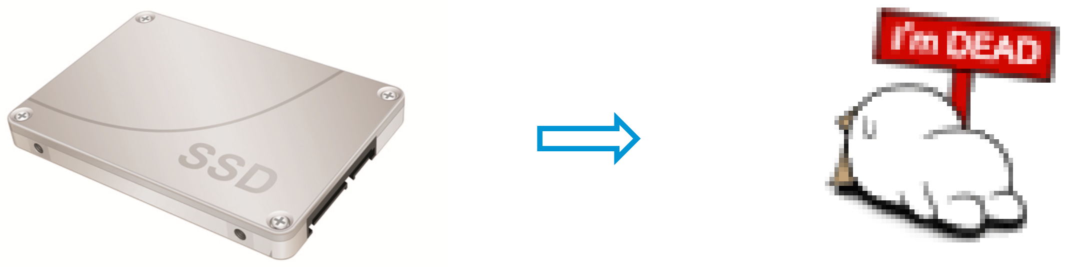
- Bit Corruption



- Metadata Corruption

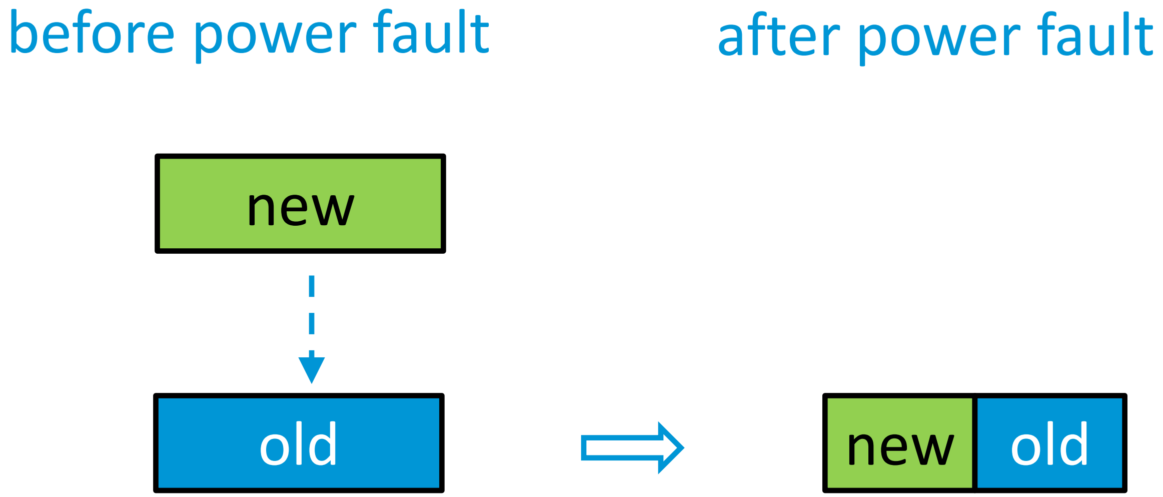


- Dead Device

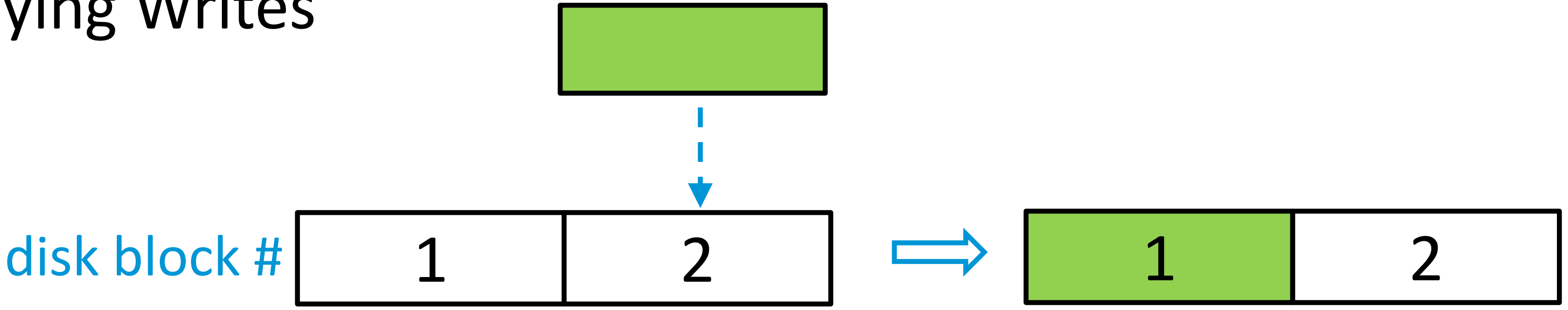


Simple Failures

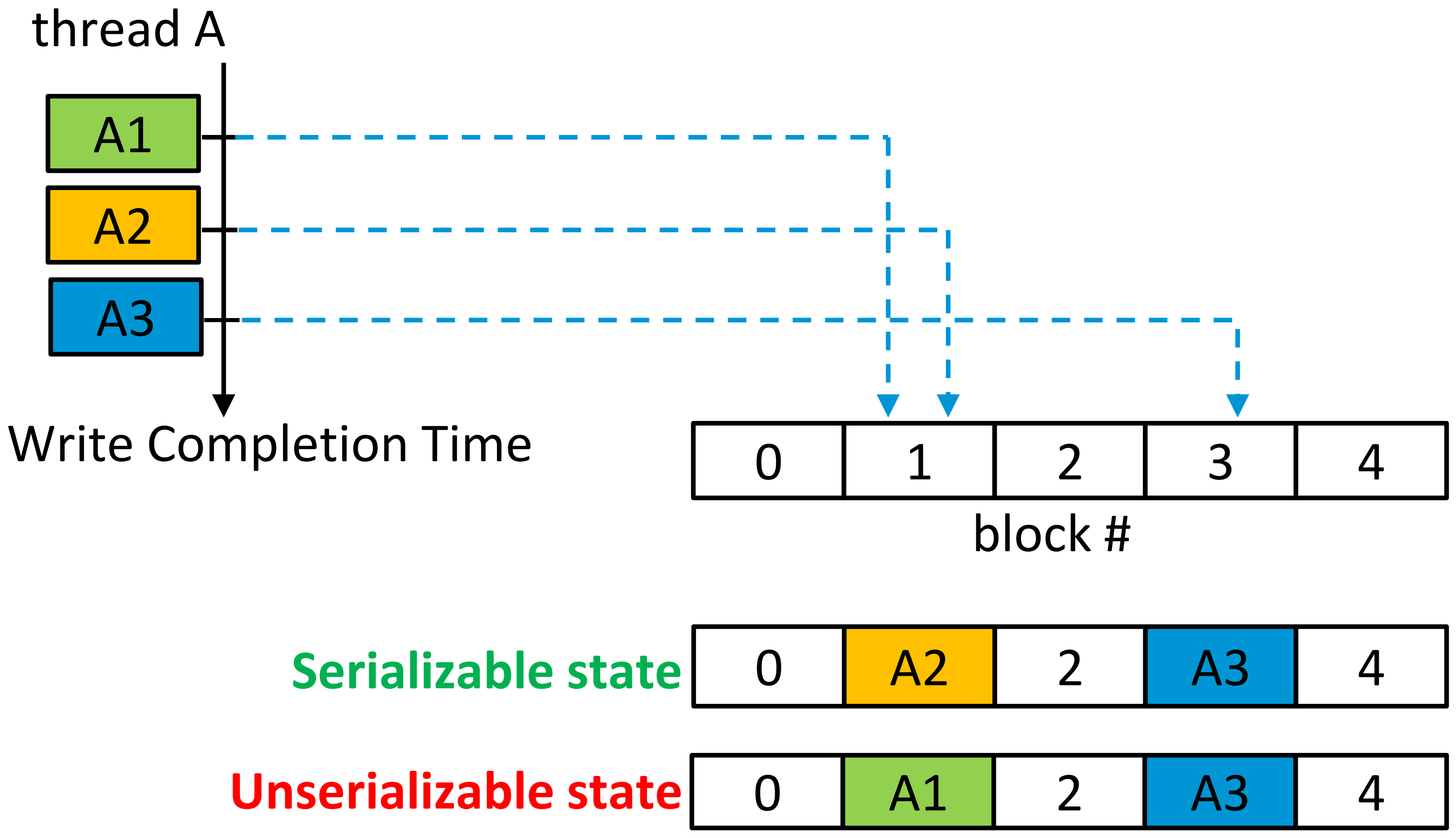
- Shorn Writes



- Flying Writes

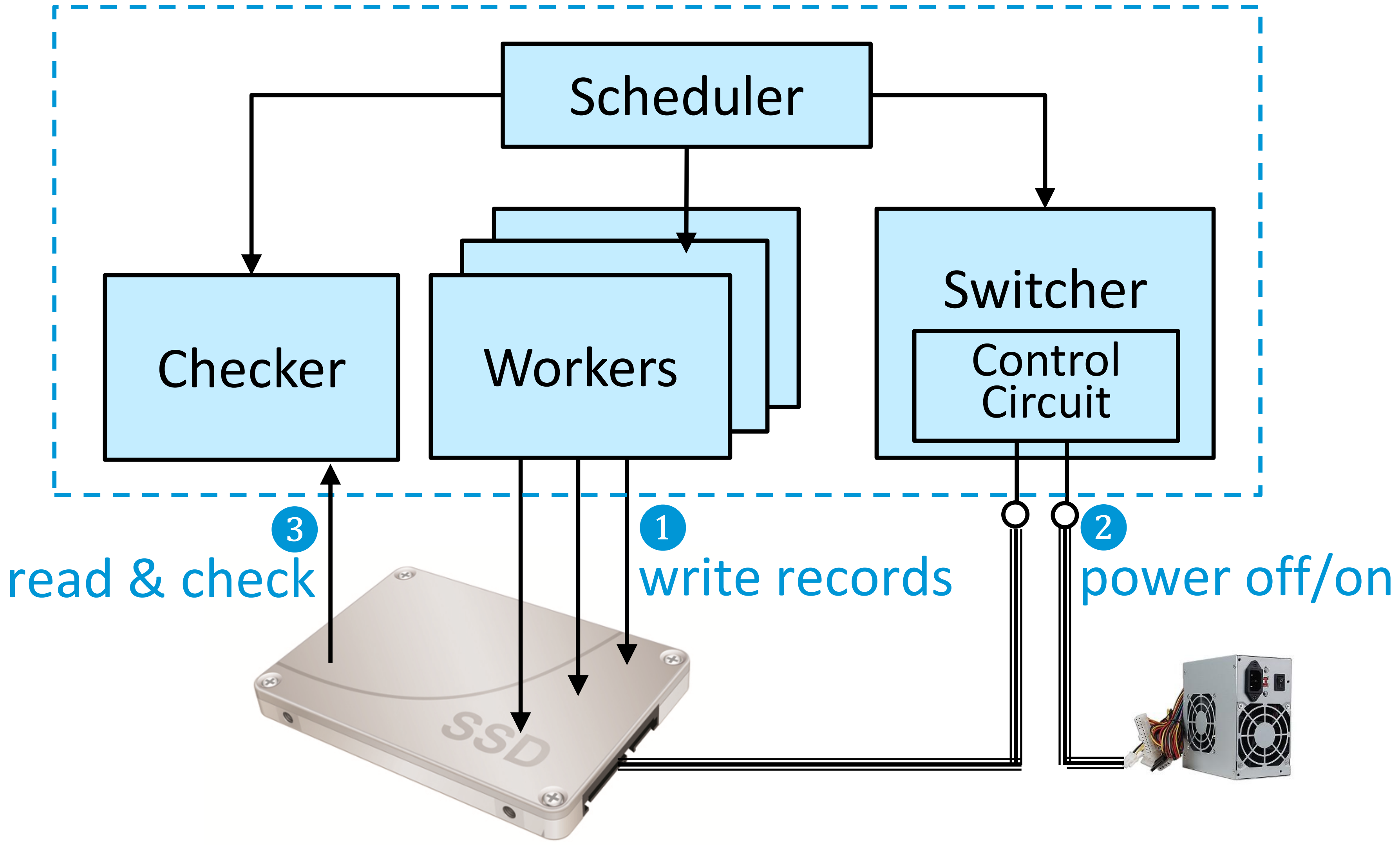


Complex Failure: Unserializable Writes

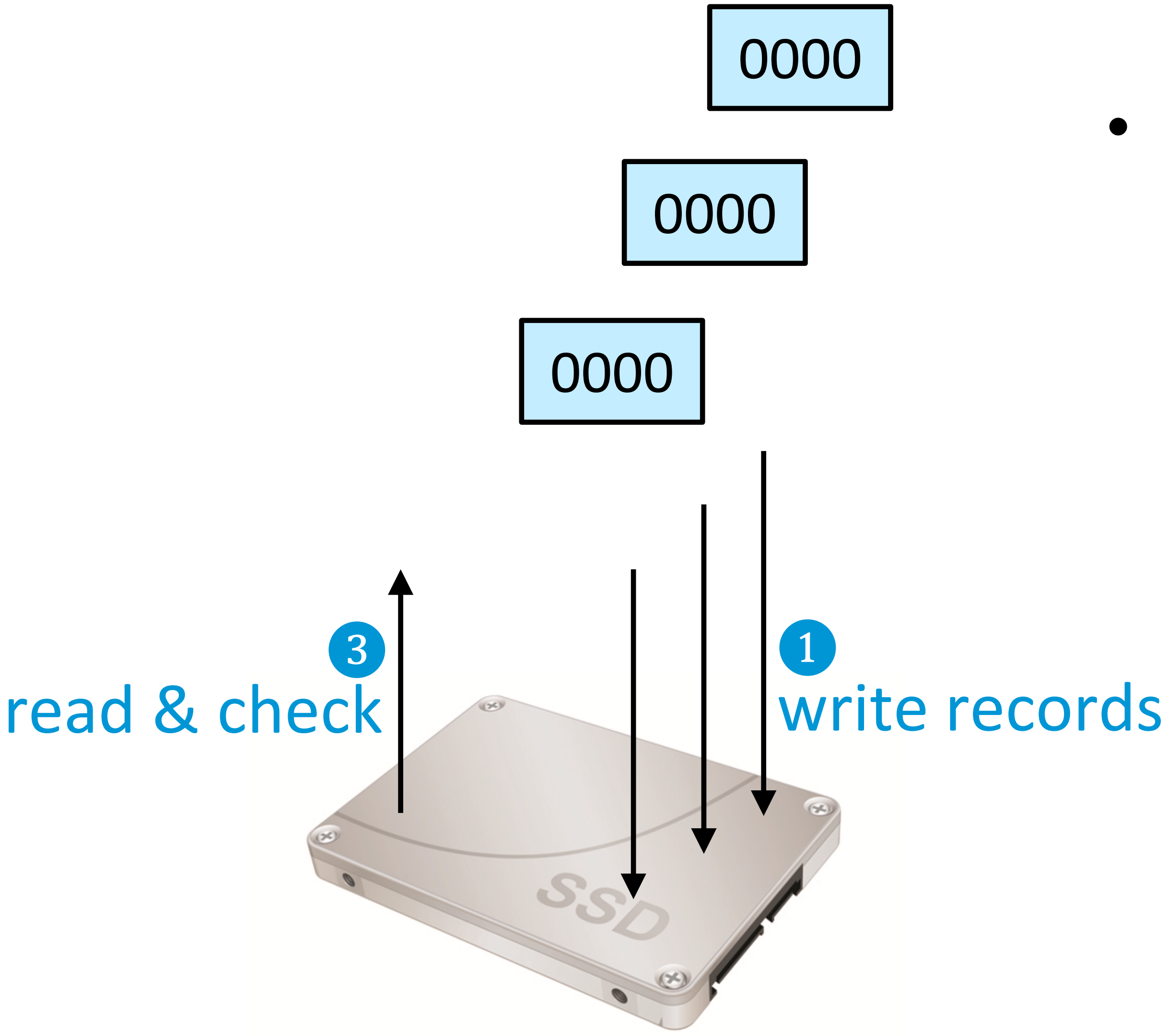


Testing Framework

Design

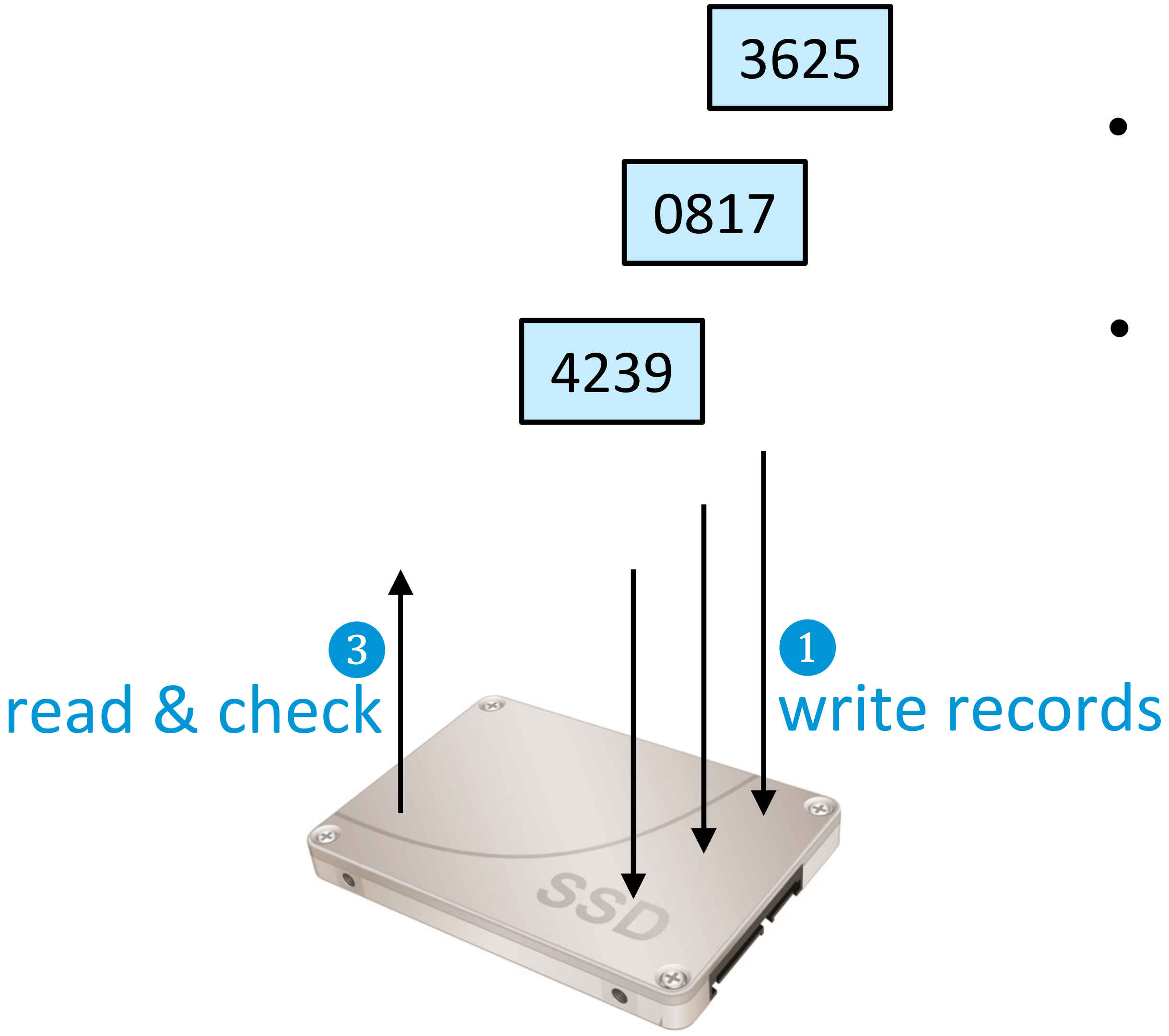

Scheduler
Checker
Workers
Switcher
Control Circuit
3
read & check
1
write records
2
power off/on
SSD

What to Write?



- all 0's ? ☹

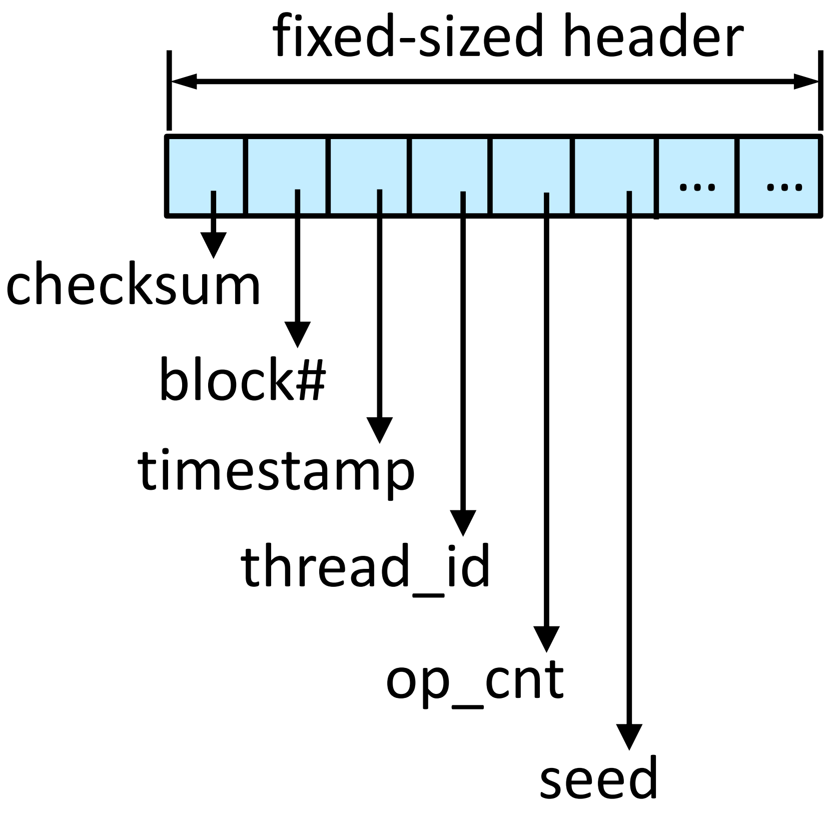
What to Write?



- all 0’s ? ☹
- random numbers? ☹

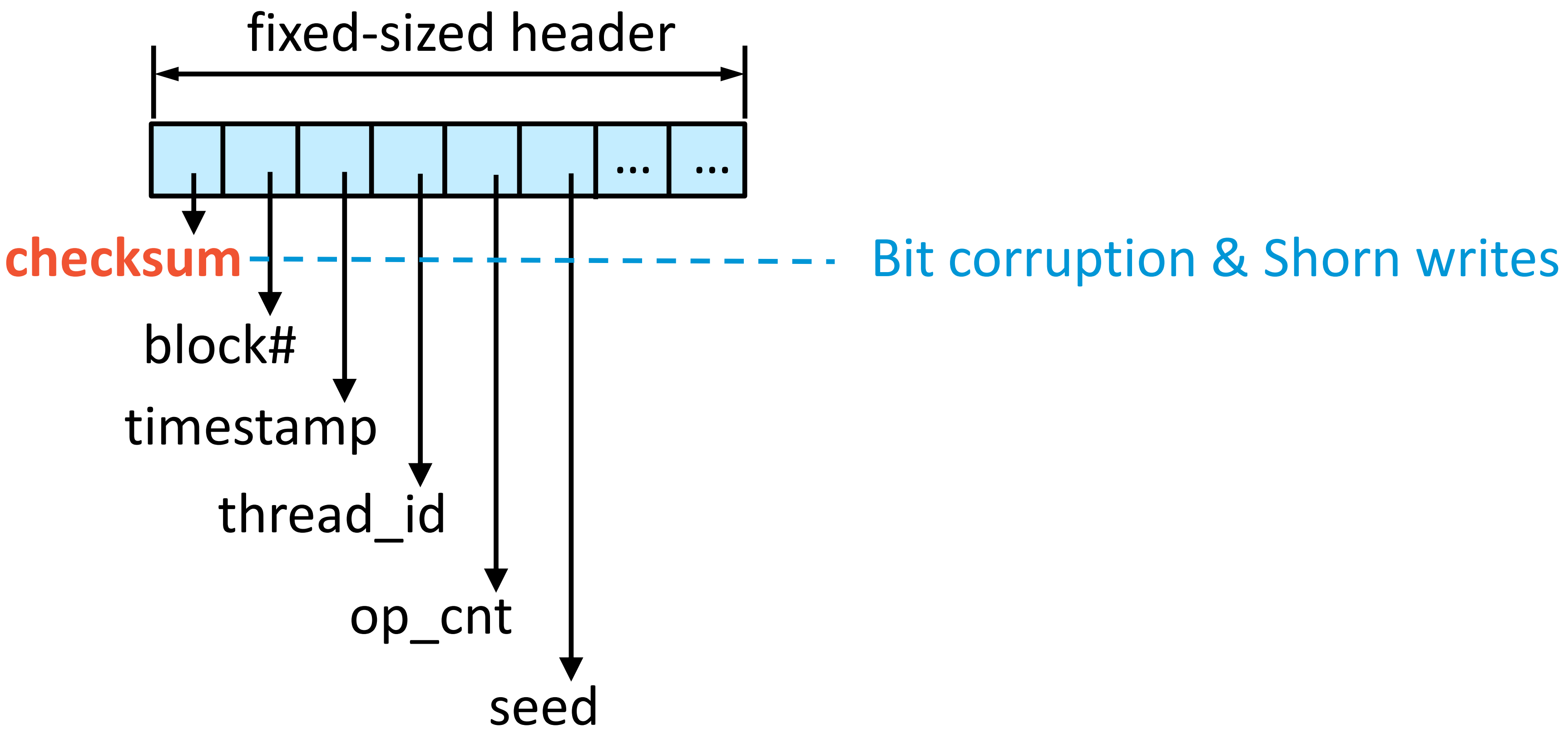
Special Record Format

- allows detecting all 6 types of failures



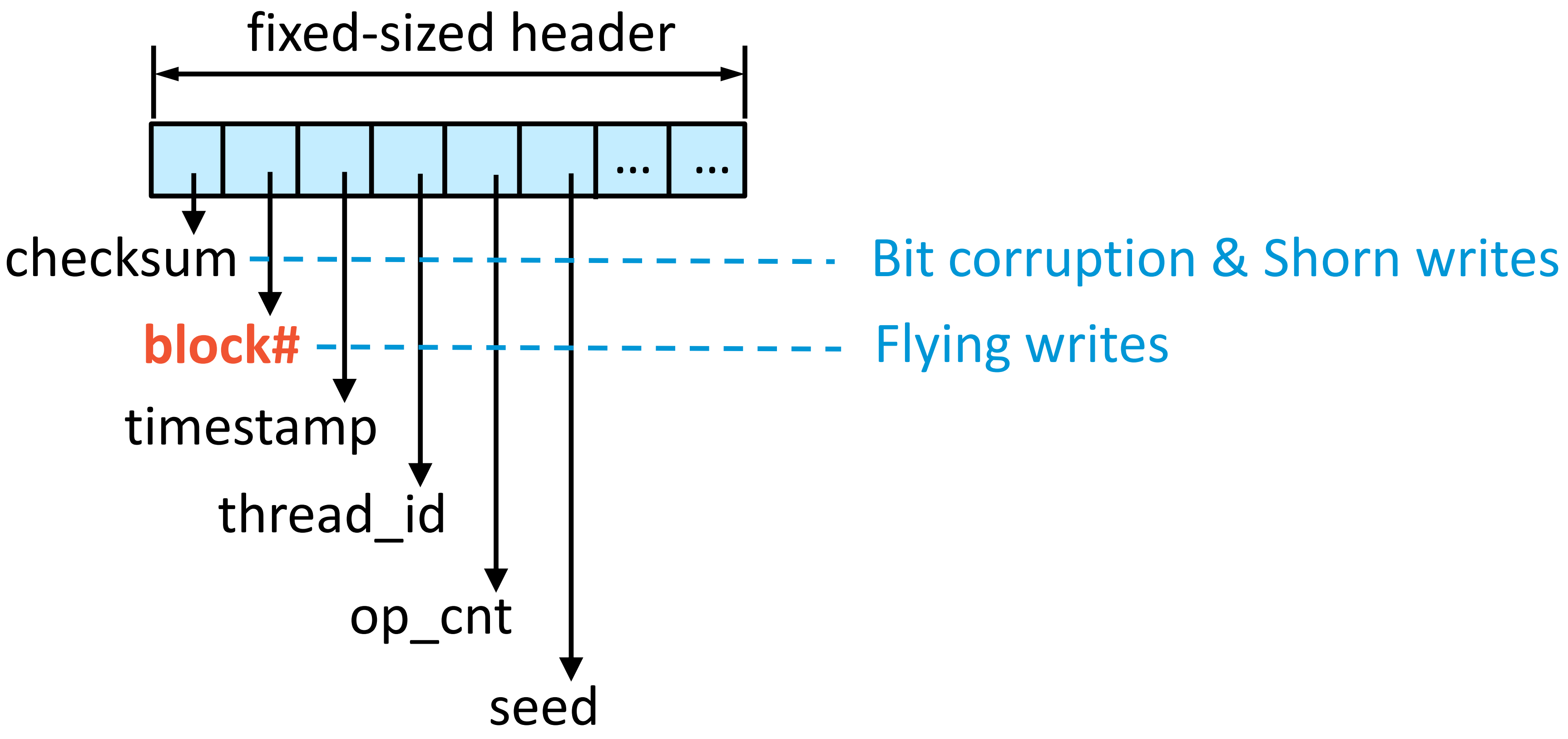
Special Record Format

- allows detecting all 6 types of failures



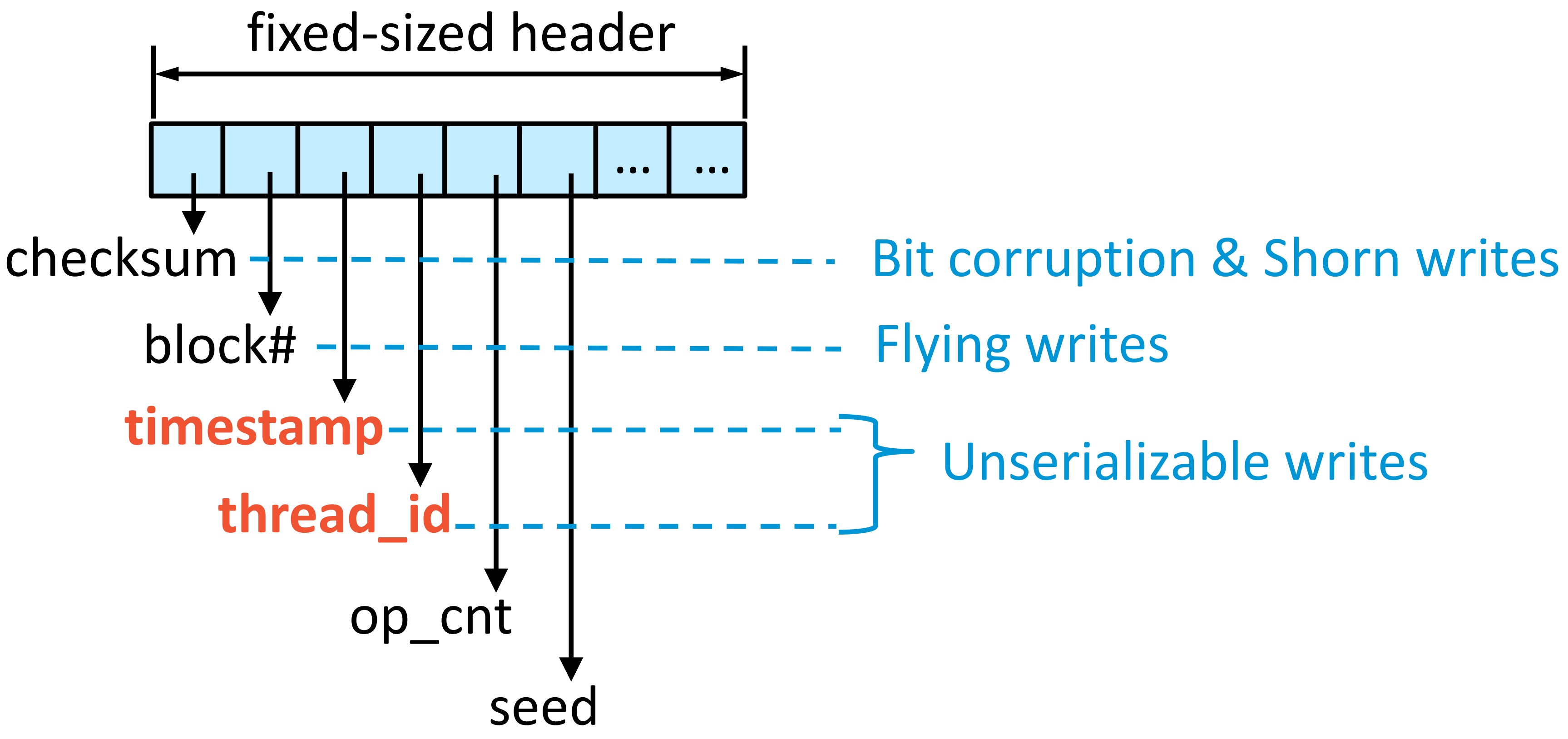
Special Record Format

- allows detecting all 6 types of failures



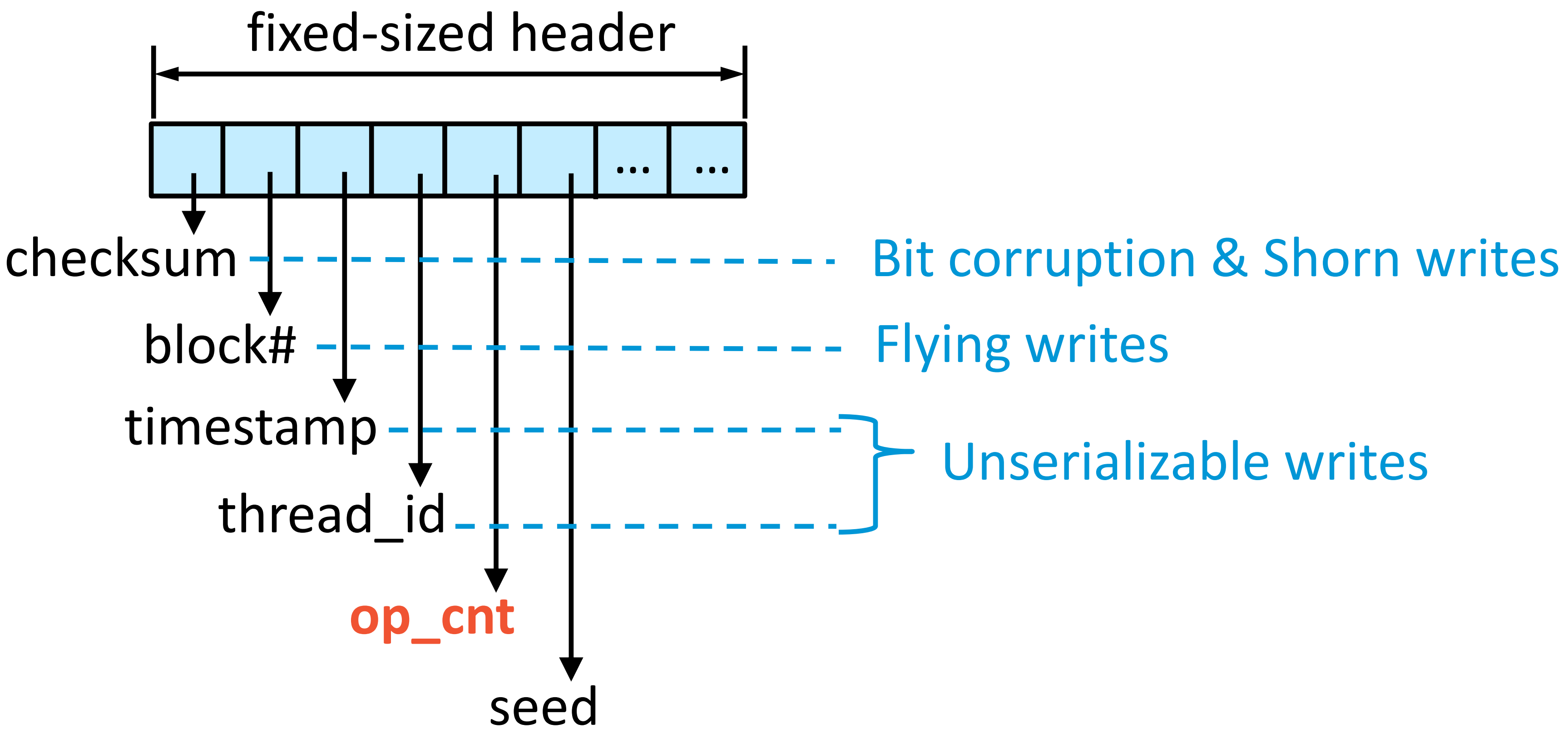
Special Record Format

- allows detecting all 6 types of failures



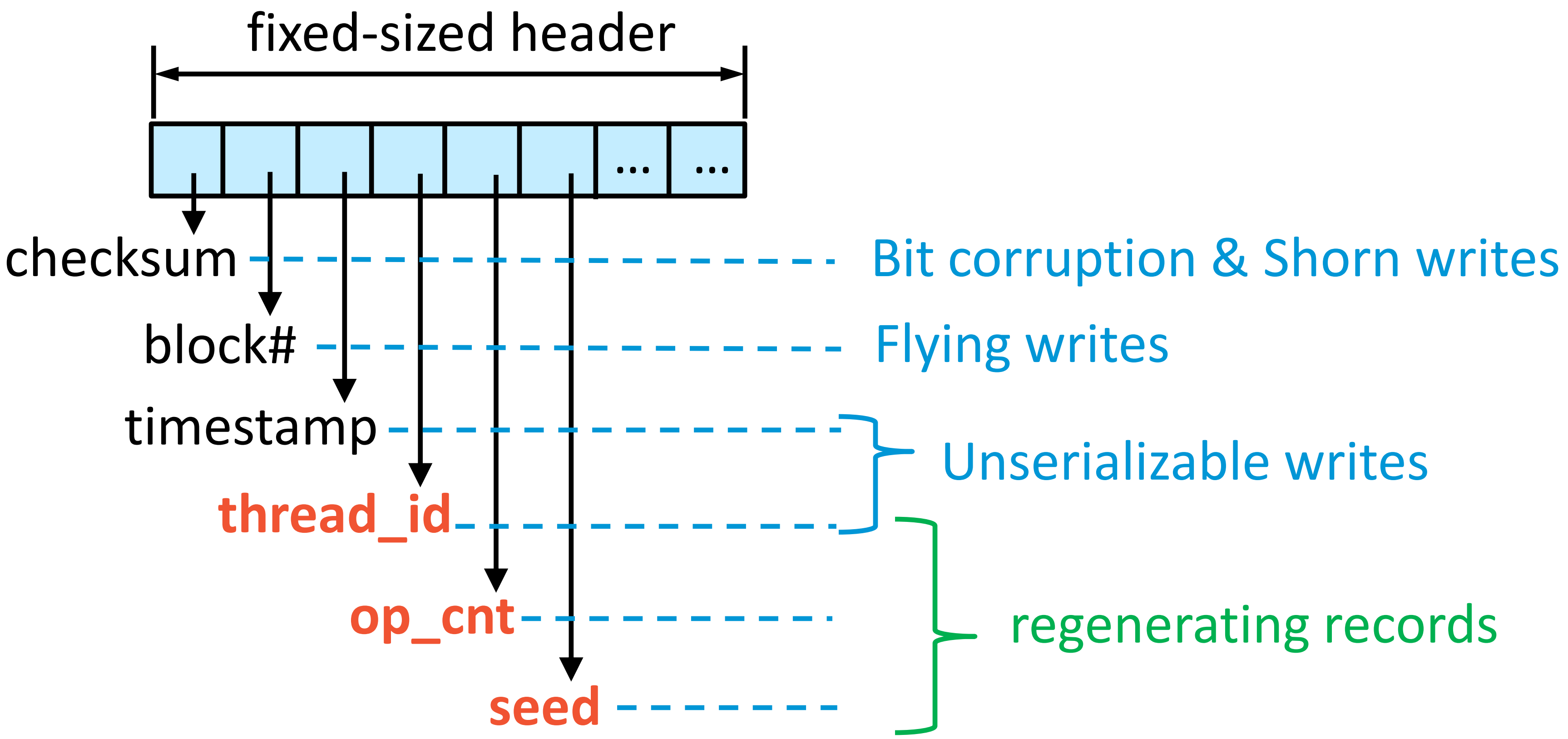
Special Record Format

- allows detecting all 6 types of failures



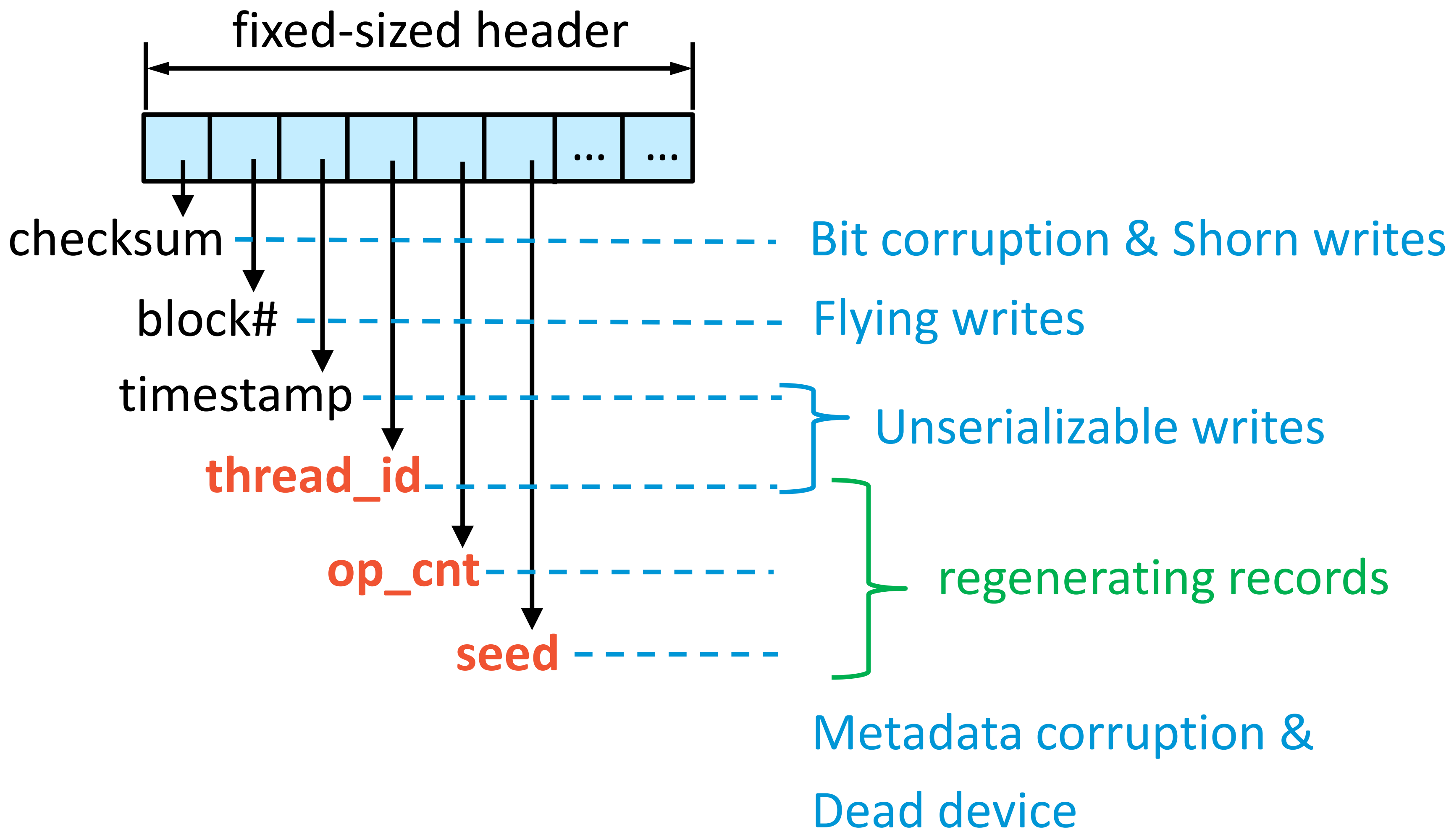
Special Record Format

- allows detecting all 6 types of failures



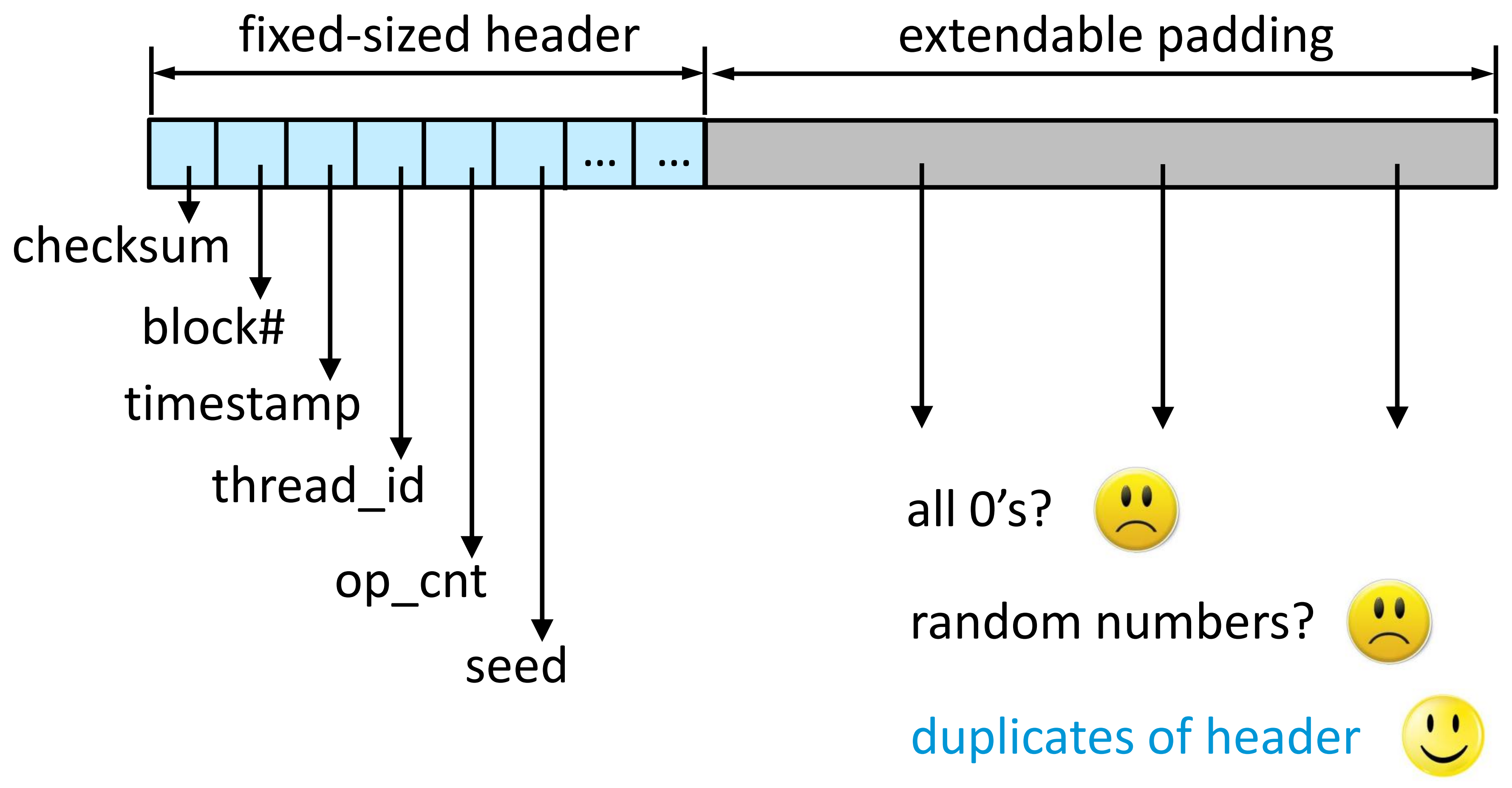
Special Record Format

- allows detecting all 6 types of failures

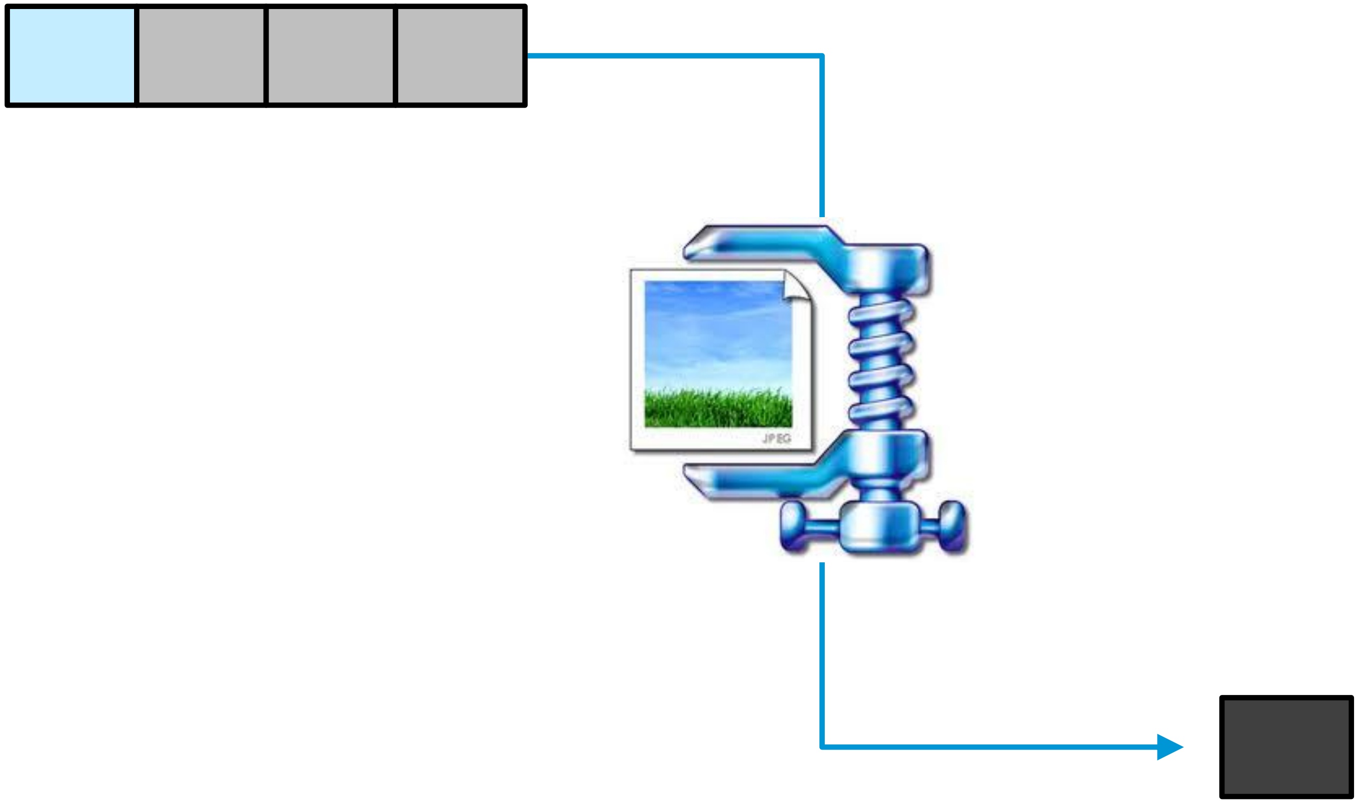


Special Record Format

- allows detecting all 6 types of failures

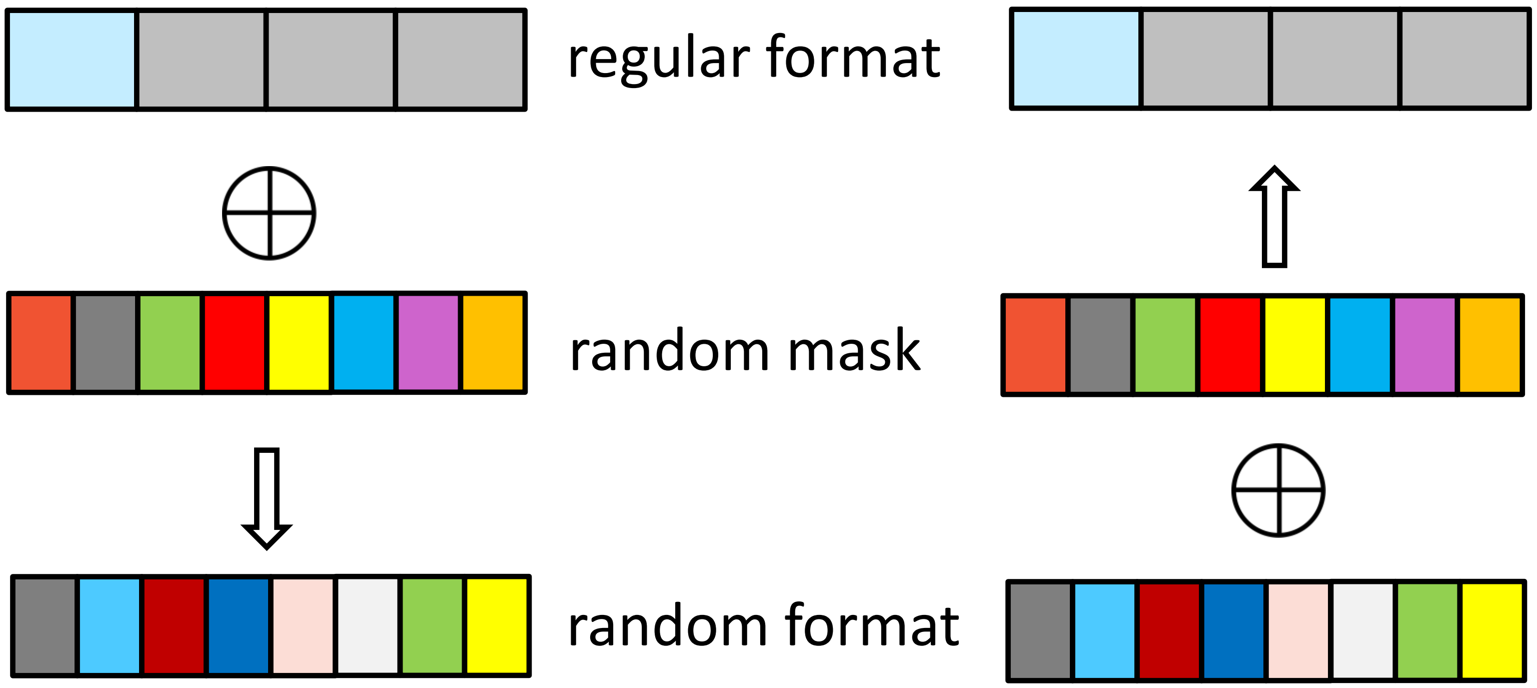


Advanced FTL: Compression



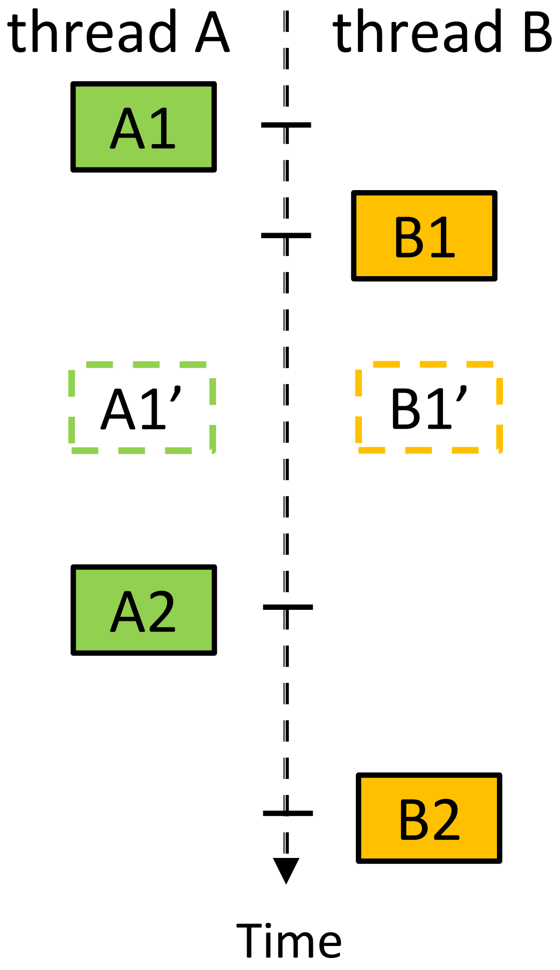
Randomization of Record Content

- avoid interference of compression



Deriving Completion-time Partial Order

- a key step of unserializable writes detection

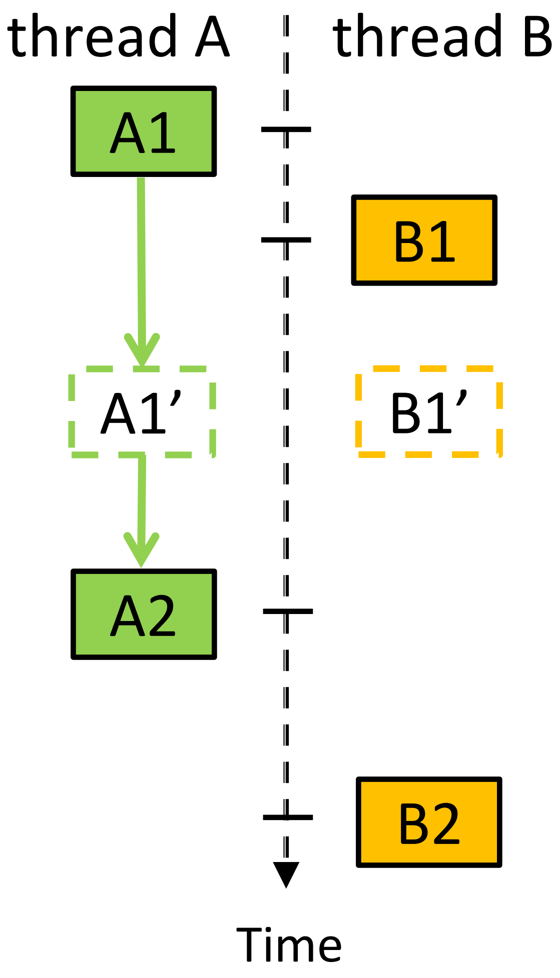


🟩 A1 : generating time of 1st record of A

▨ A1' : completion time of writing 1st record of A

Deriving Completion-time Partial Order

- a key step of unserializable writes detection

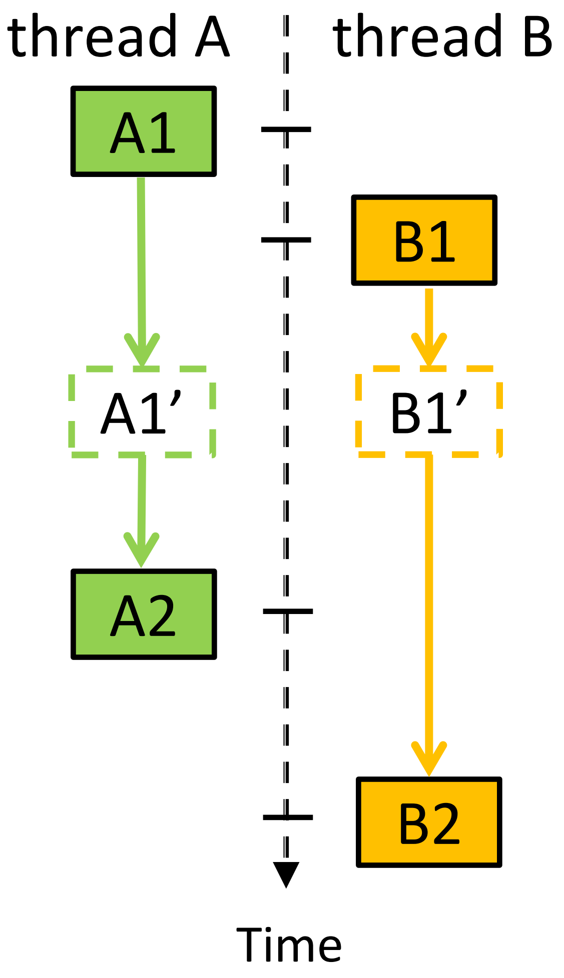


Intra-thread:

A1 -> A1’ -> A2

Deriving Completion-time Partial Order

- a key step of unserializable writes detection



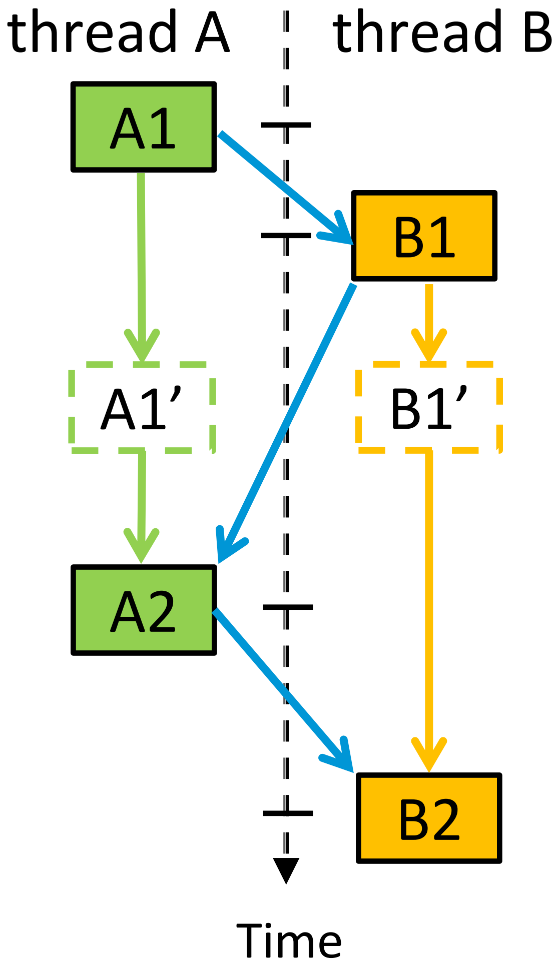
Intra-thread:

A1 -> A1’ -> A2

B1 -> B1’ -> B2

Deriving Completion-time Partial Order

- a key step of unserializable writes detection



Intra-thread:

A1 -> A1’ -> A2

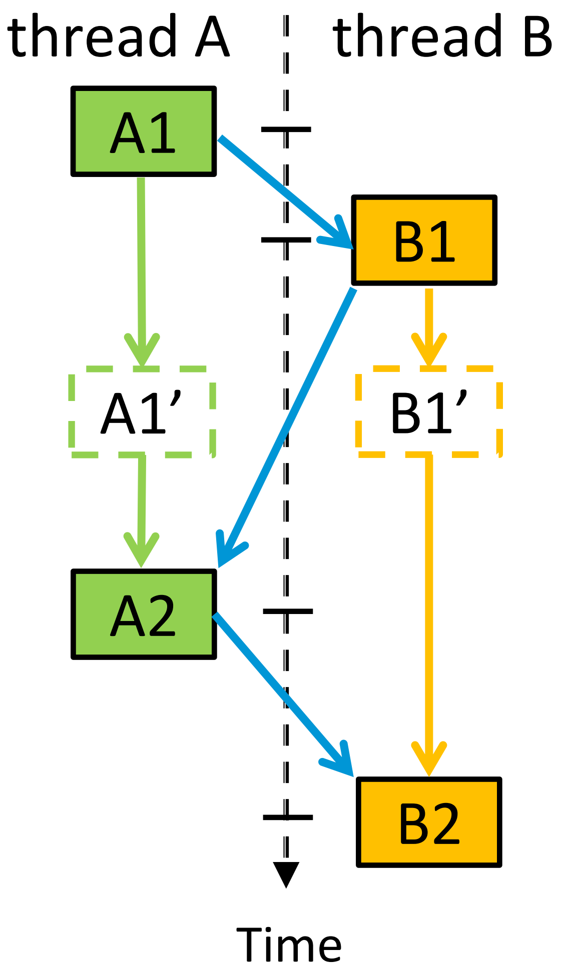
B1 -> B1’ -> B2

Inter-thread:

A1 -> B1 -> A2 -> B2

Deriving Completion-time Partial Order

- a key step of unserializable writes detection



Intra-thread:

A1 -> A1’ -> A2

B1 -> B1’ -> B2

Inter-thread:

A1 -> B1 -> A2 -> B2

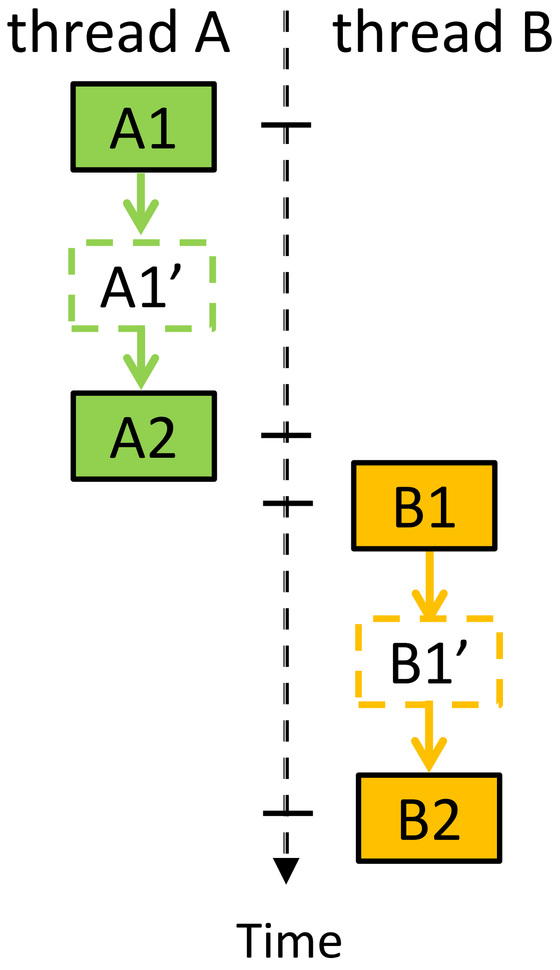
$\not\Rightarrow$ A1’ -> B1’ or

B1’ -> A1’

Conservatively report no errors

Deriving Completion-time Partial Order

- a key step of unserializable writes detection



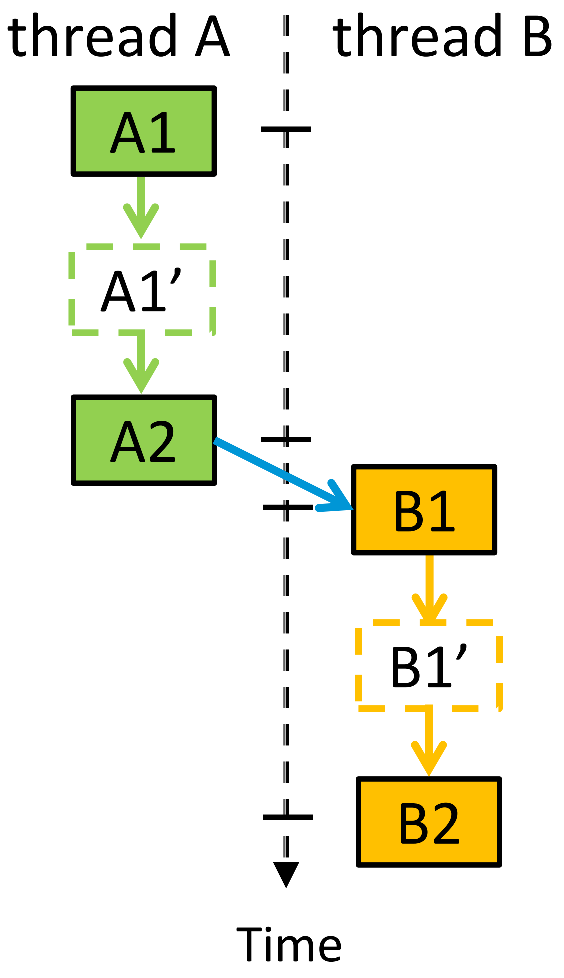
Intra-thread:

A1 -> A1' -> A2

B1 -> B1' -> B2

Deriving Completion-time Partial Order

- a key step of unserializable writes detection



Intra-thread:

A1 -> A1’ -> A2

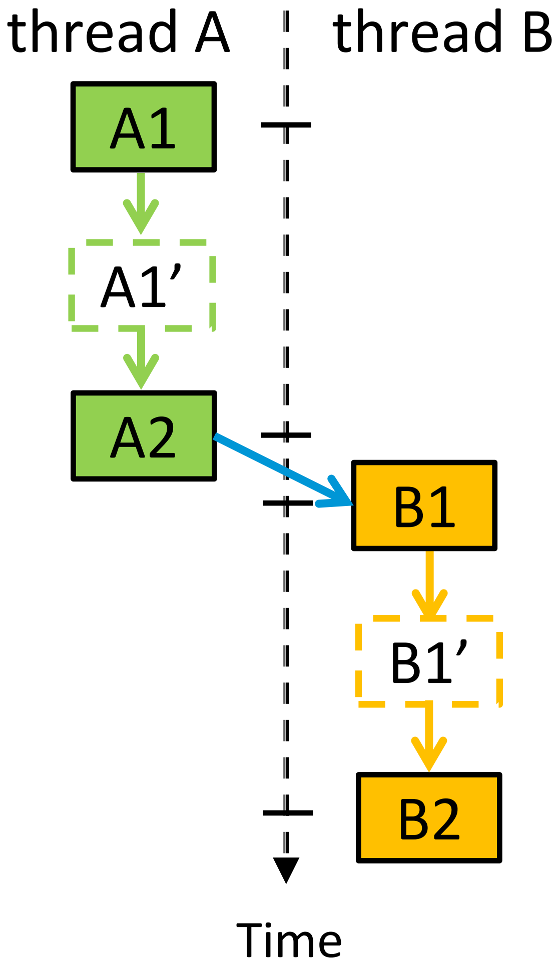
B1 -> B1’ -> B2

Inter-thread:

A2 -> B1

Deriving Completion-time Partial Order

- a key step of unserializable writes detection



Intra-thread:

A1 -> A1’ -> A2

B1 -> B1’ -> B2

Inter-thread:

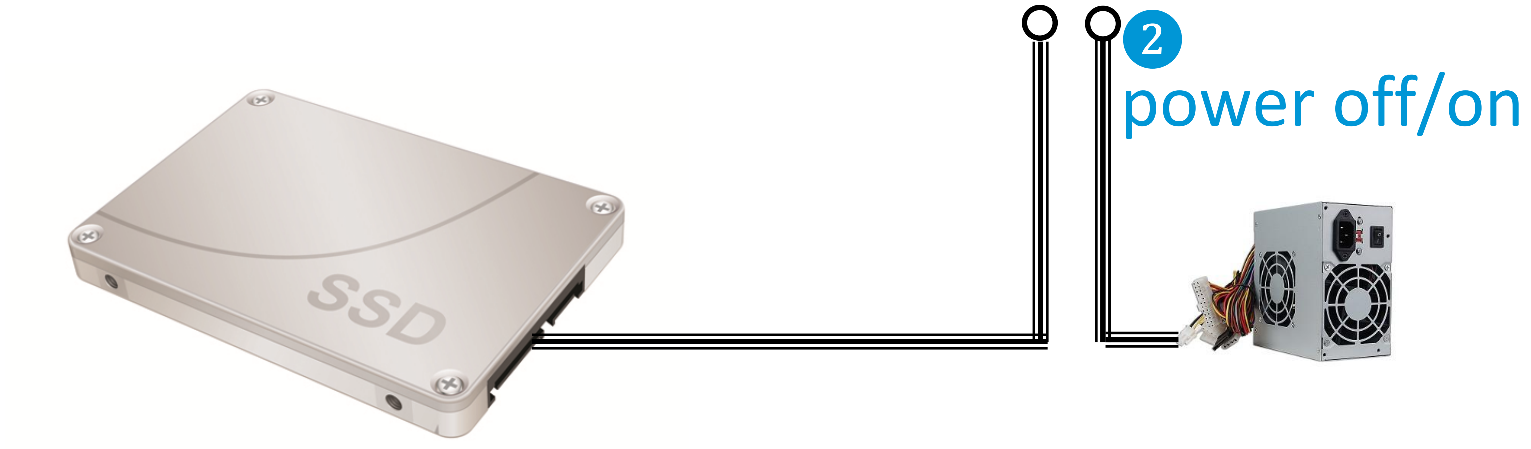
A2 -> B1

$\Rightarrow$ A1’ -> B1’

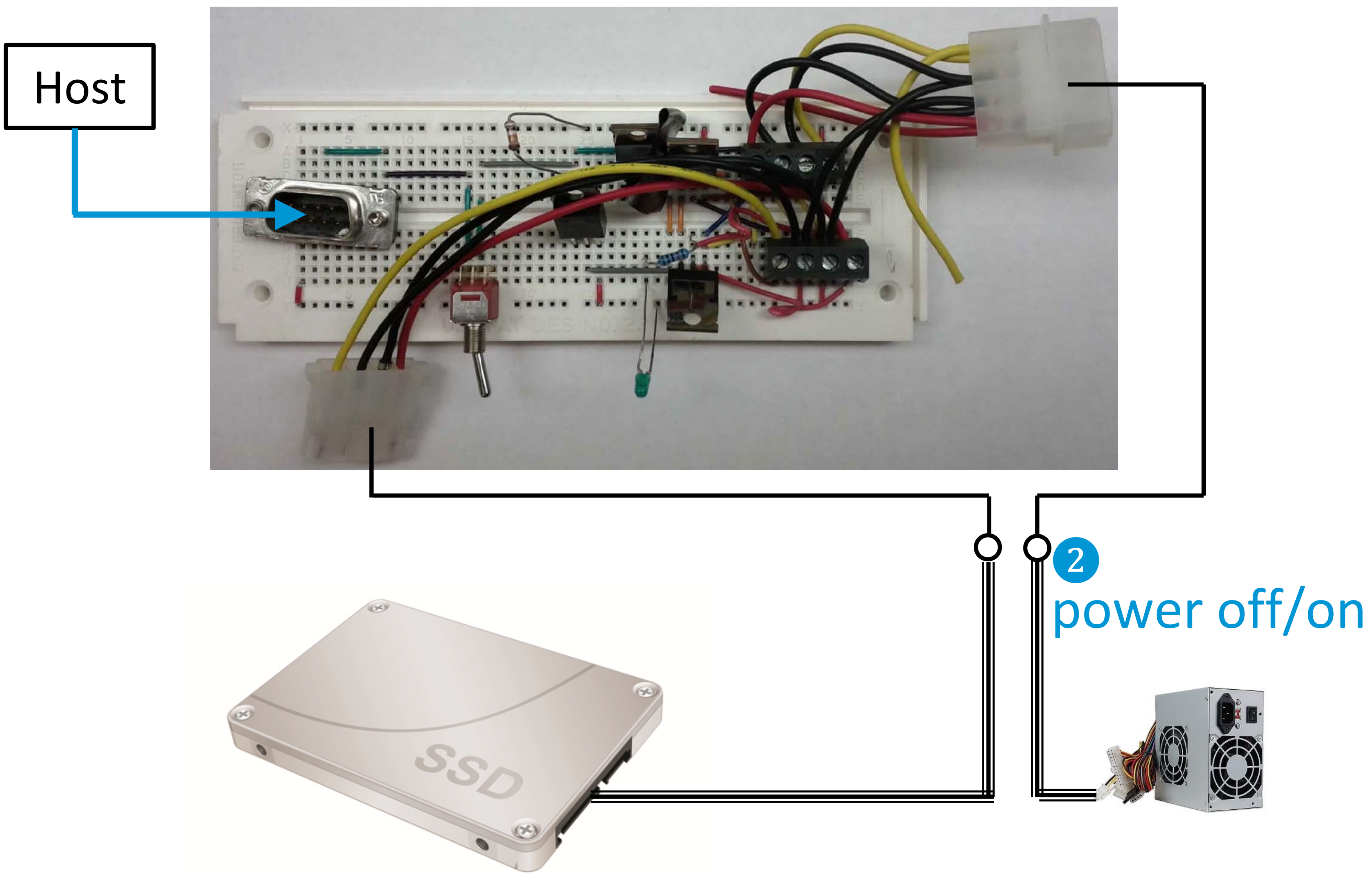
More details in our paper

& Golab *et al.* PODC’11

Power Fault Injection



Power Fault Injection



Results

Experimental Environment

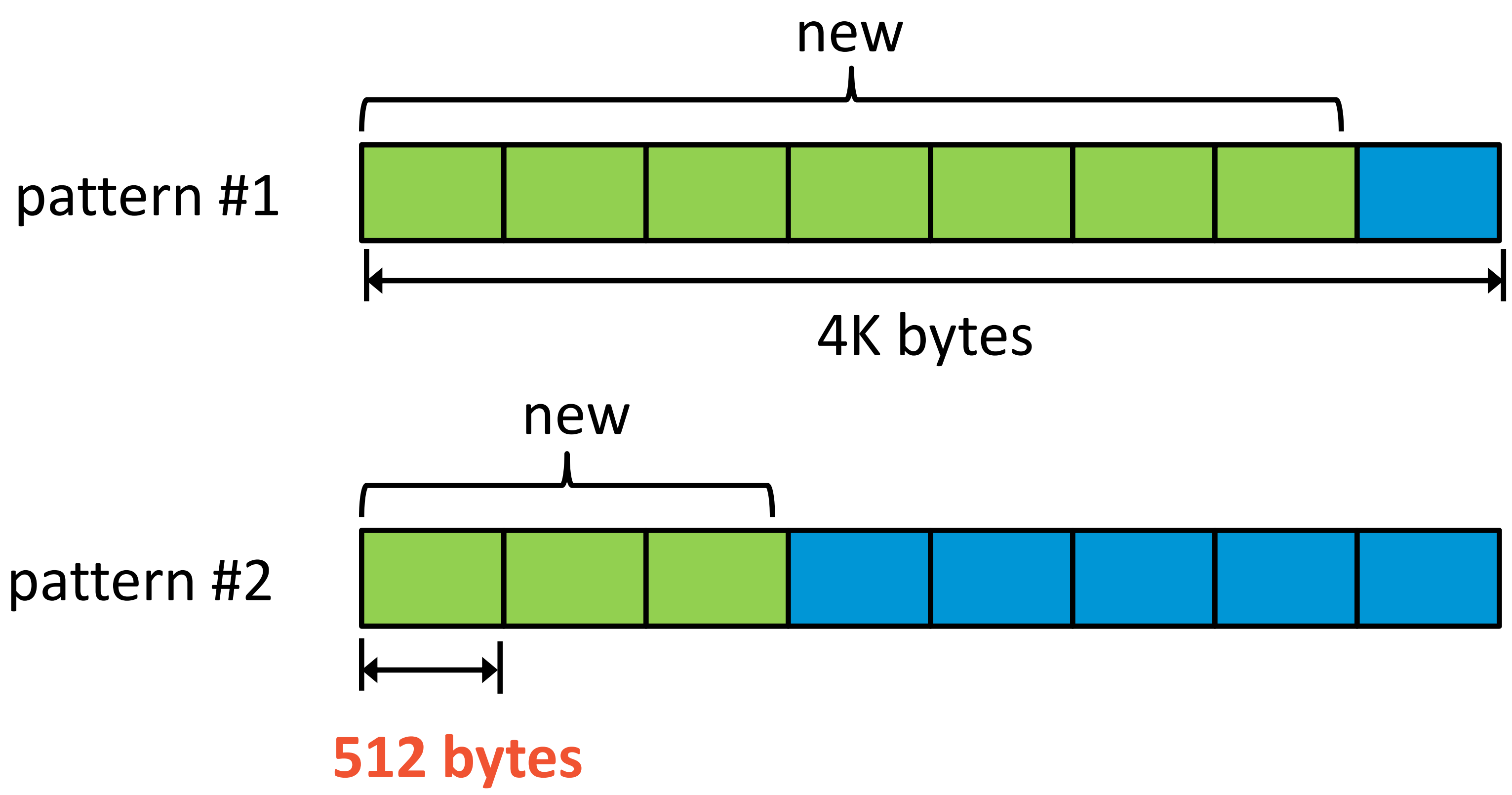
- Block Devices
  - 15 SSDs and 2 hard drives
  - SLC & MLC
  - Manufactured in 2009 – 2012
  - 4 have power-loss protection
  - Low-end to high-end ($0.63/GB - $6.50/GB)
- Host System
  - Debian 6.0 w/ 2.6.32 kernel
  - LSI Logic SAS controller
  - no filesystem on devices
  - Synchronized & Direct I/O (O_SYNC | O_DIRECT)

Summary of Observations

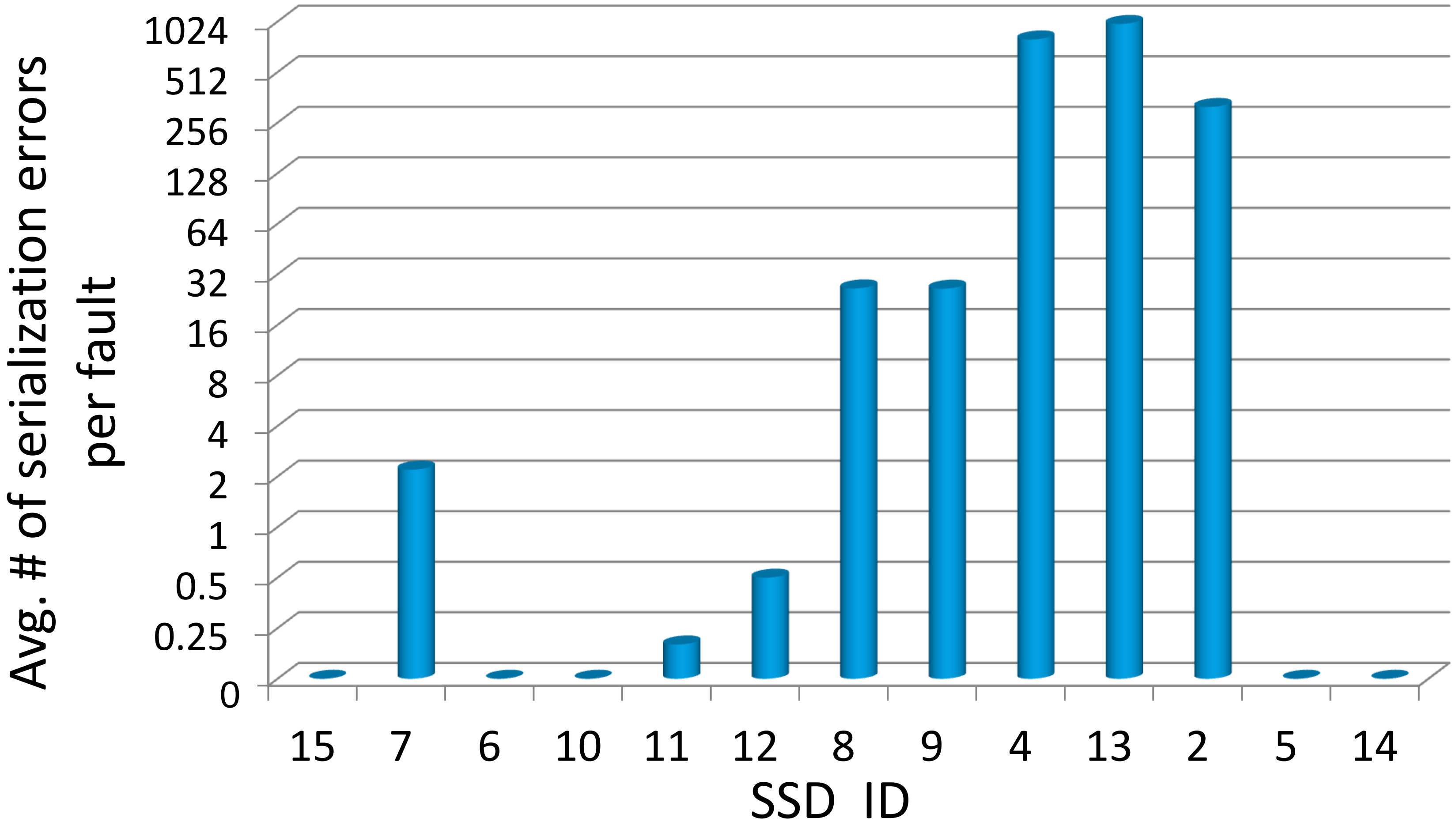
Failures	# of SSDs
Bit Corruption	3
Metadata Corruption	1
Dead Device	1
Shorn Writes	3
Flying Writes	0
Unserializable Writes	8
None	2

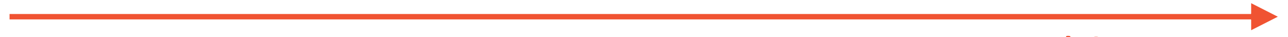
- 13 of 15 SSDs exhibit failure(s)
- 2 perfect SSDs
- 5 of 6 failures observed

Shorn Writes: Subpage Programming

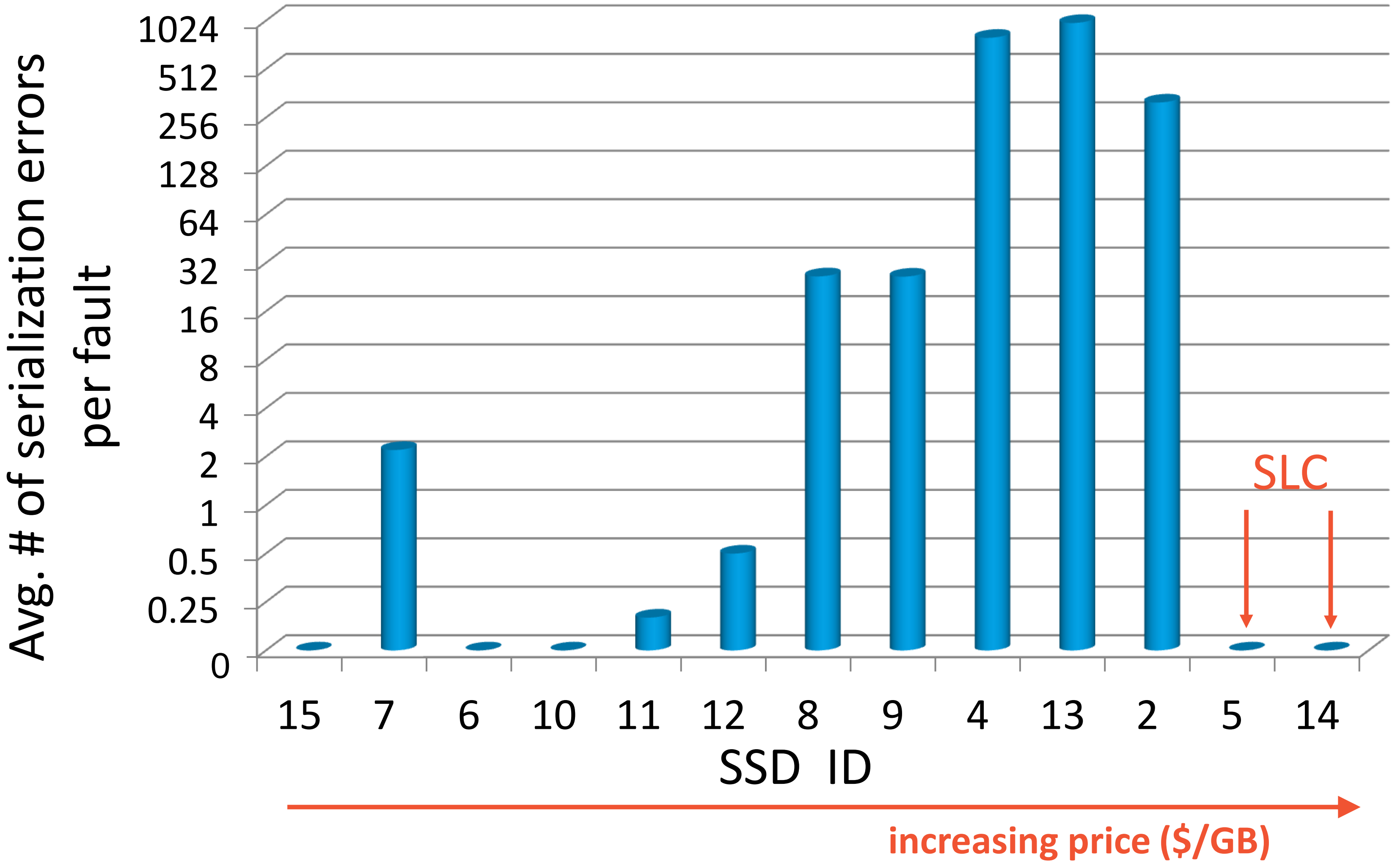


Serialization Errors: Avg. Numbers Per Fault

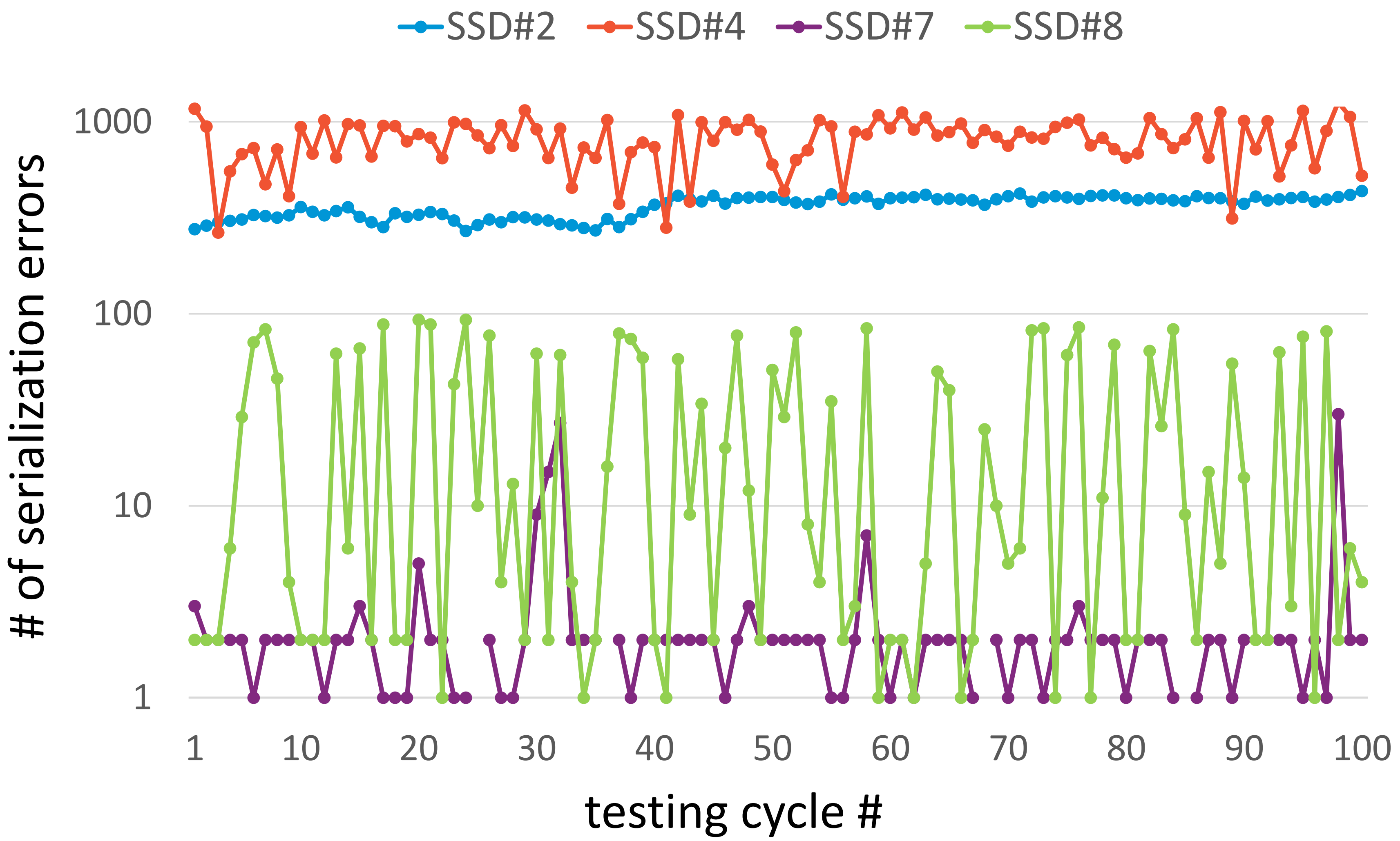




Serialization Errors: Avg. Numbers Per Fault

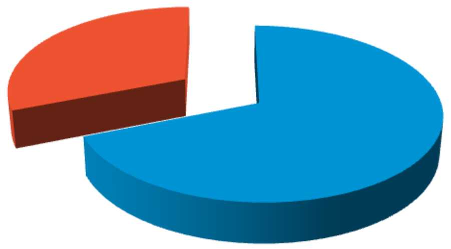


Serialization Errors: Patterns Over Time



Metadata Corruption

- 1 SSD
- 8 injected power faults
- lost 31% (72 GB) data



Dead Device

- 1 SSD
- 136 injected power faults
- can no long be detected by host

Conclusion

- An effective methodology to expose bugs in block devices under power fault
- Important implications to storage design
  - e.g. write ahead logging V.S. unserializable writes

Thank you!

Pristine Version of Our Paper can be Found at:

http://www.cse.ohio-state.edu/~zhengm/