The 14th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage)

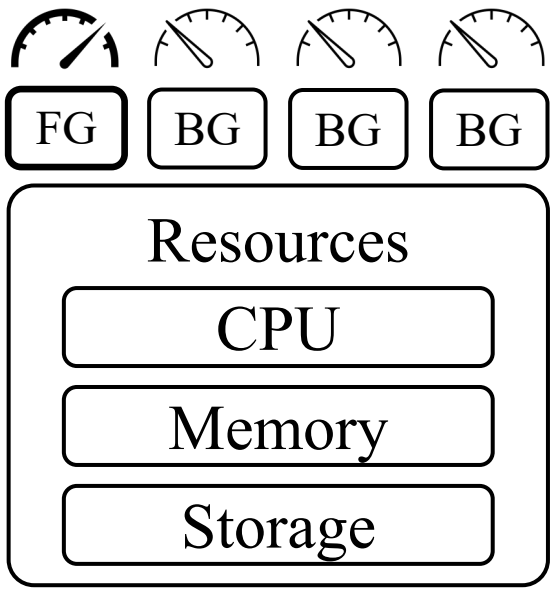
File Fragmentation from the Perspective of I/O Control

Jonggyu Park and Young Ik Eom
Sungkyunkwan University

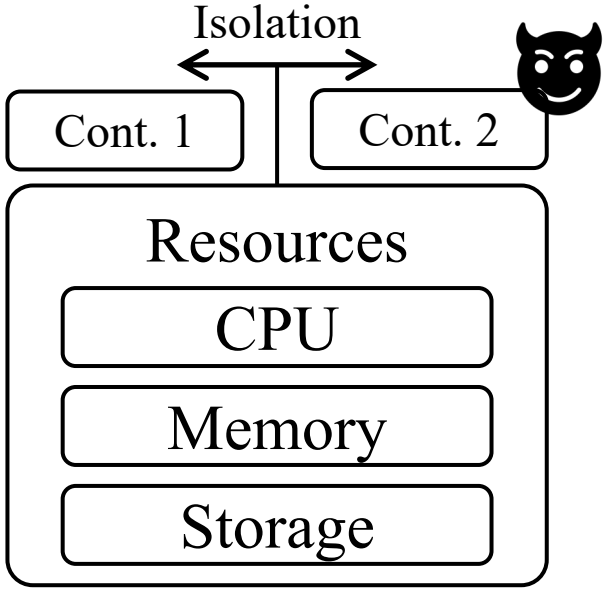
File Fragmentation
from the Perspective of I/O Control

Linux I/O control mechanisms

- Multiple apps/containers co-run together
  - Resource control for better responsiveness and performance isolation
  - Weight-based I/O control with Cgroup



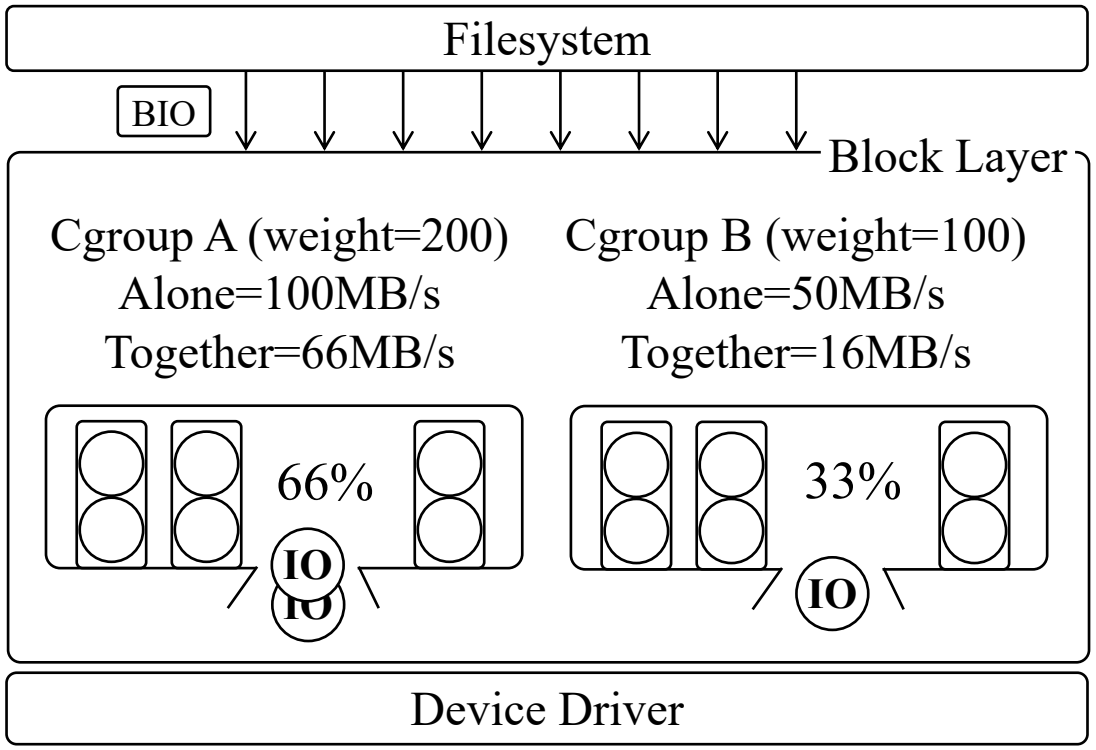
< Mobile/Desktop >



< Server >

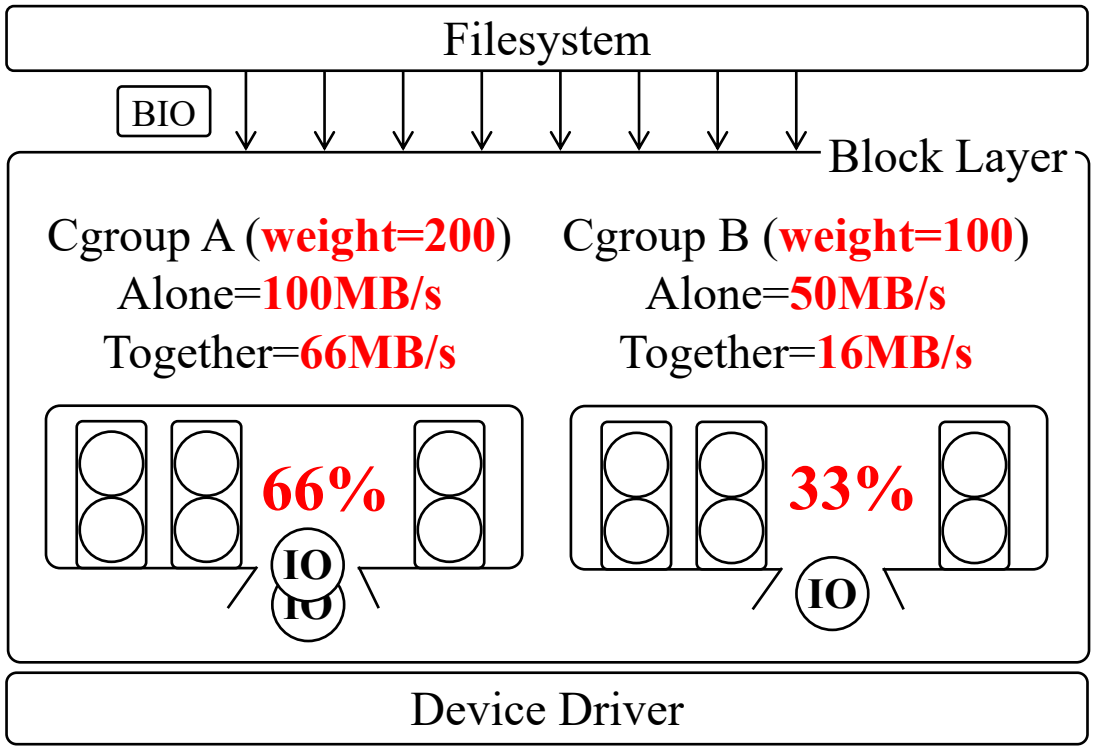
Linux I/O control mechanisms

- Weight-based I/O control mechanisms distribute a fair amount of …
  - CFQ: IOPS-based
  - BFQ: sector-based
  - IOCost: device occupation-based
    1. Modeling performance of SSDs
    2. Estimate device occupation of each I/O



Linux I/O control mechanisms

- Weight-based I/O control mechanisms distribute a fair amount of …
  - CFQ: IOPS-based
  - BFQ: sector-based
  - IOCost: device occupation-based
    1. Modeling performance of SSDs
    2. Estimate device occupation of each I/O

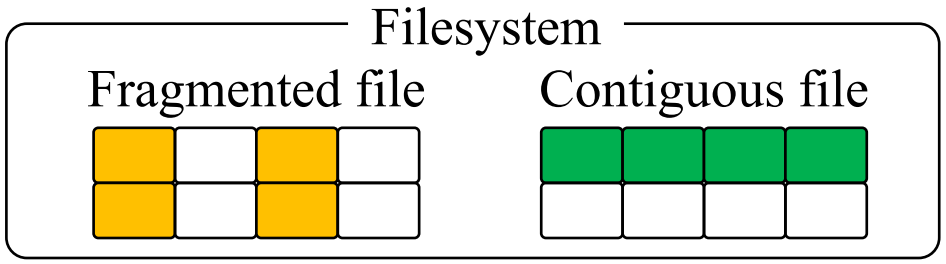


From the Perspective of I/O Control

File Fragmentation
from the Perspective of I/O Control

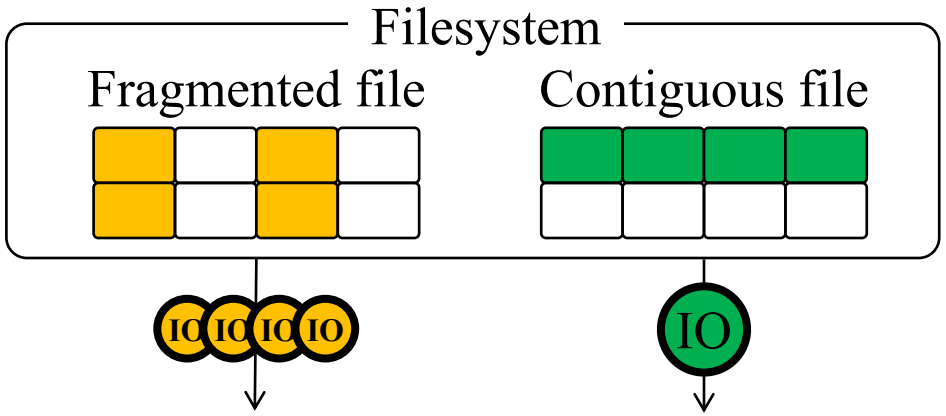
What is (intra-file) fragmentation?

- A file is scattered into multiple fragments



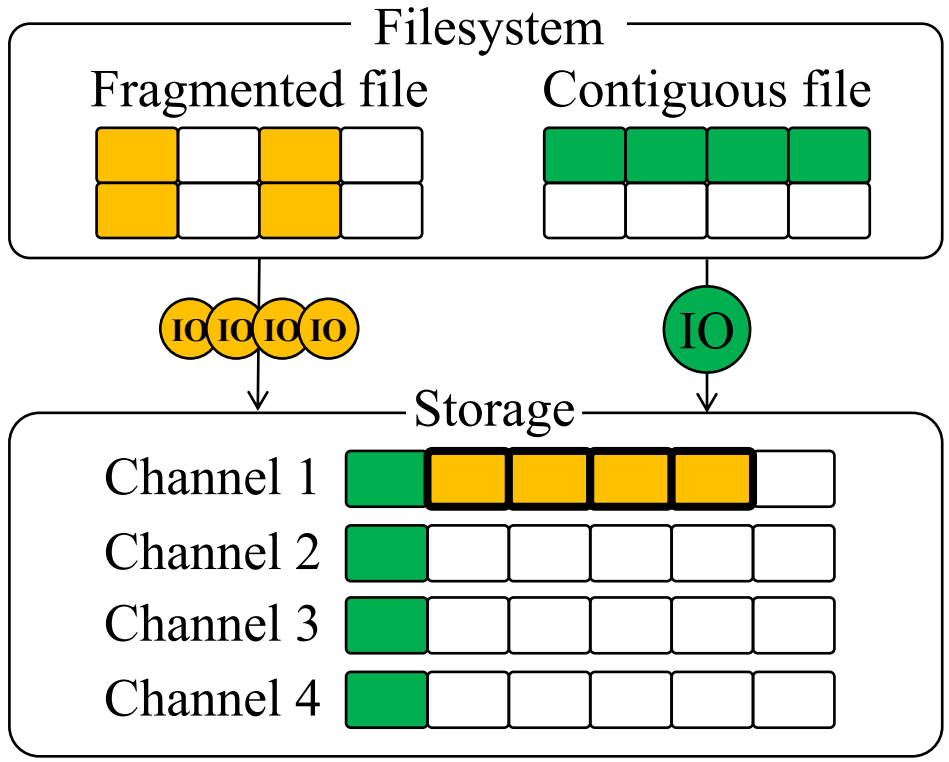
What is (intra-file) fragmentation?

- A file is scattered into multiple fragments
- Problems
  - Increases **the number of I/Os**
  - Decreases **the size of I/Os**



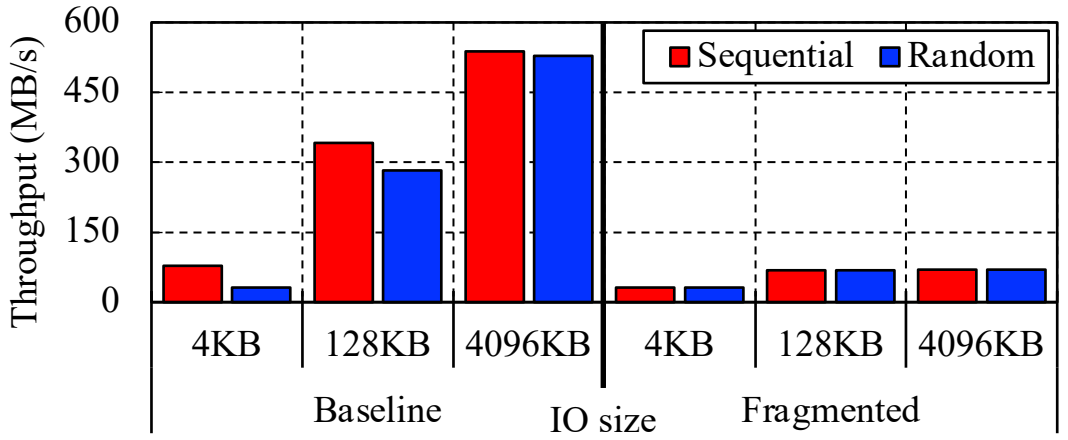
What is (intra-file) fragmentation?

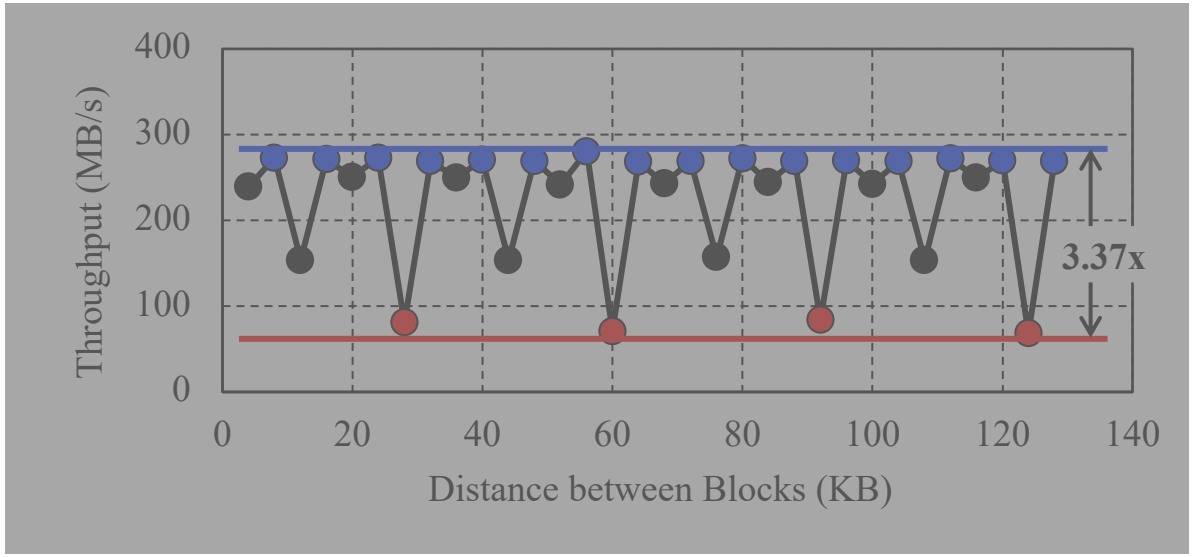
- A file is scattered into multiple fragments
- Problems
  - Increases the number of I/Os
  - Decreases the size of I/Os
  - Increases **I/O randomness**



Fragmentation from the perspective of performance

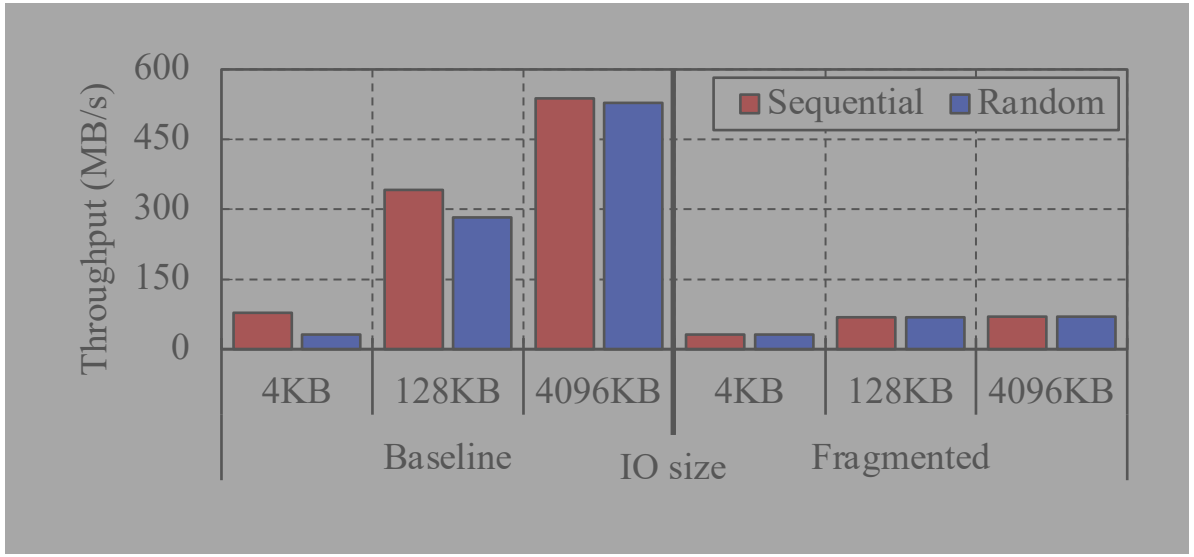
- Causes
  - Increases the number of I/Os (32x)
  - Decreases the size of I/Os (4KB)
  - Increases I/O randomness
- Effects
  - **Lower performance**
  - Irregular performance

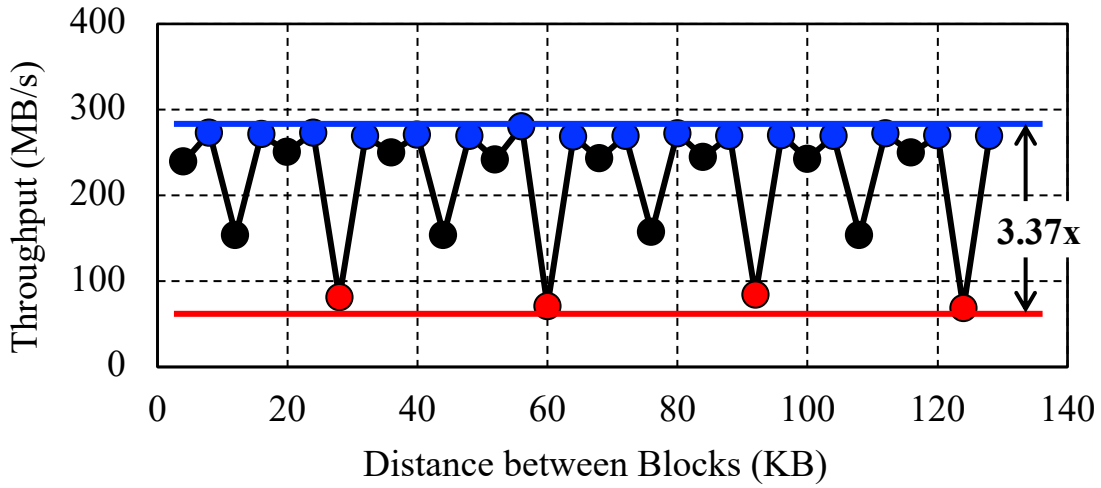




Fragmentation from the perspective of performance

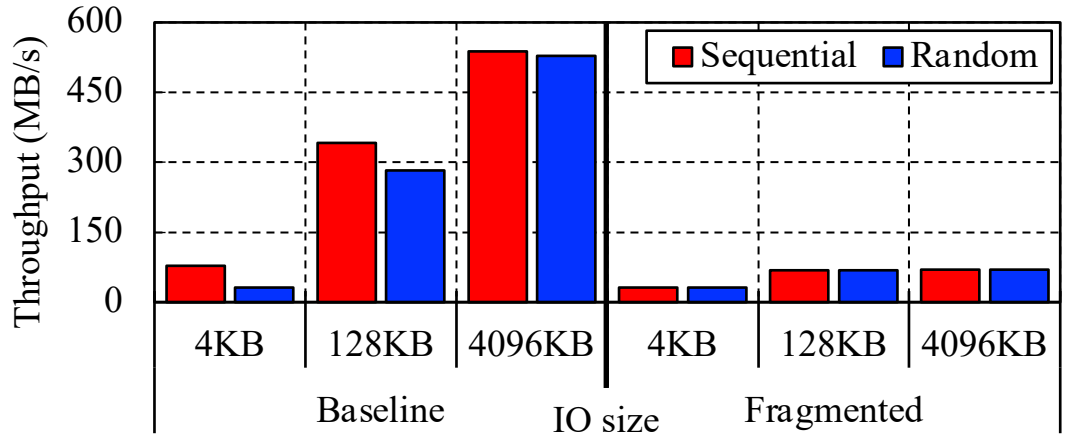
- Causes
  - Increases the number of I/Os (32x)
  - Decreases the size of I/Os (4KB)
  - Increases I/O randomness
- Effects
  - Lower performance
  - **Irregular performance**

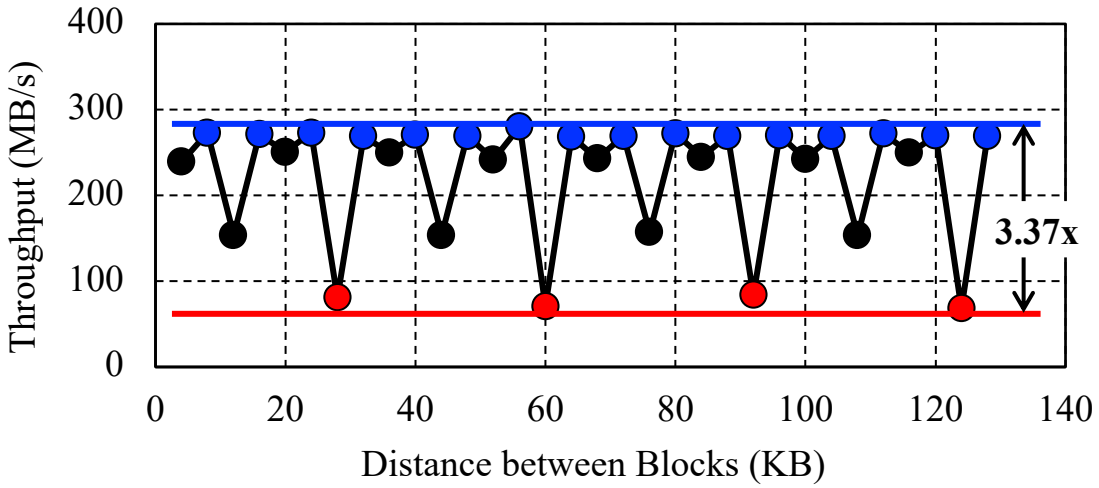




Fragmentation from the perspective of I/O control

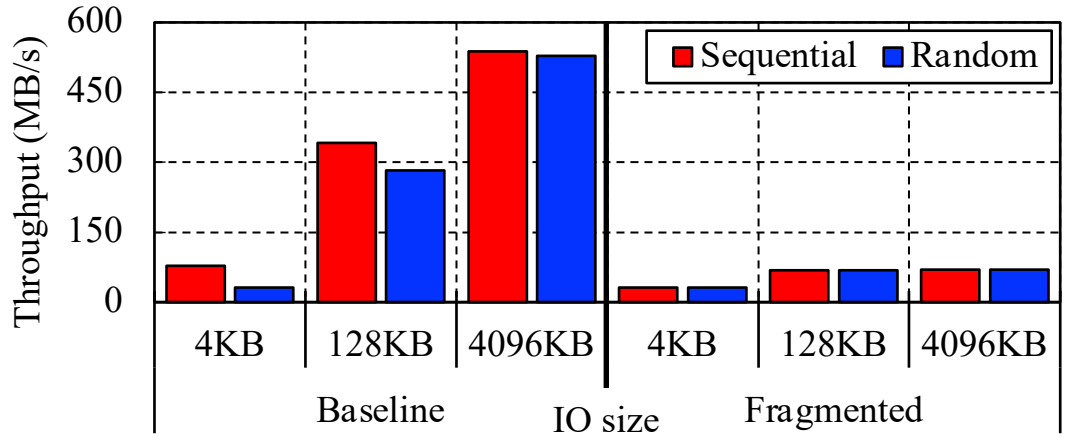
- Causes
  - **Increases the number of I/Os (32x)**
  - Decreases the size of I/Os (4KB)
  - Increases I/O randomness
- Effects
  - **CFQ: more I/Os for the same data**
  - BFQ: more time for the same data
  - IOCost:
    fail to estimate device occupation

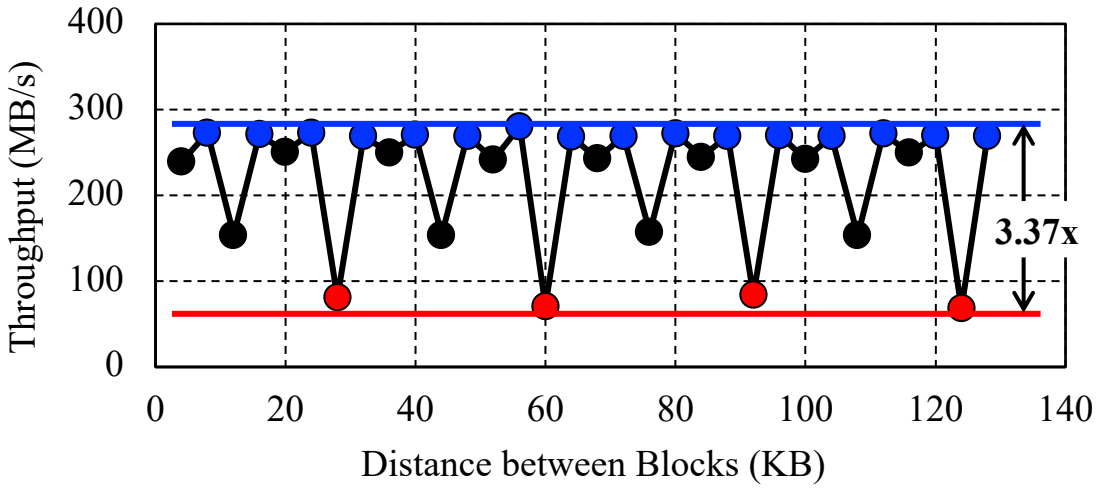




Fragmentation from the perspective of I/O control

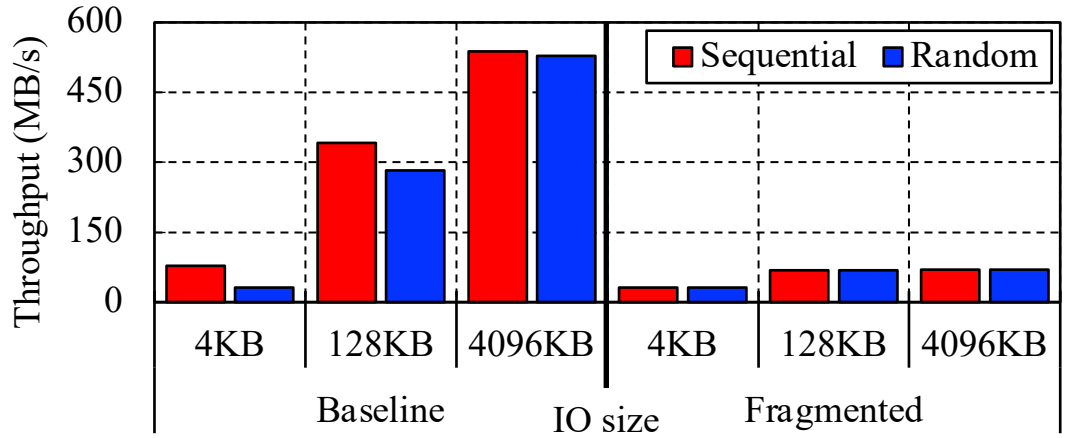
- Causes
  - Increases the number of I/Os (32x)
  - **Decreases the size of I/Os (4KB)**
  - Increases I/O randomness
- Effects
  - CFQ: more I/Os for the same data
  - **BFQ: more time for the same data**
  - IOCost:
    fail to estimate device occupation

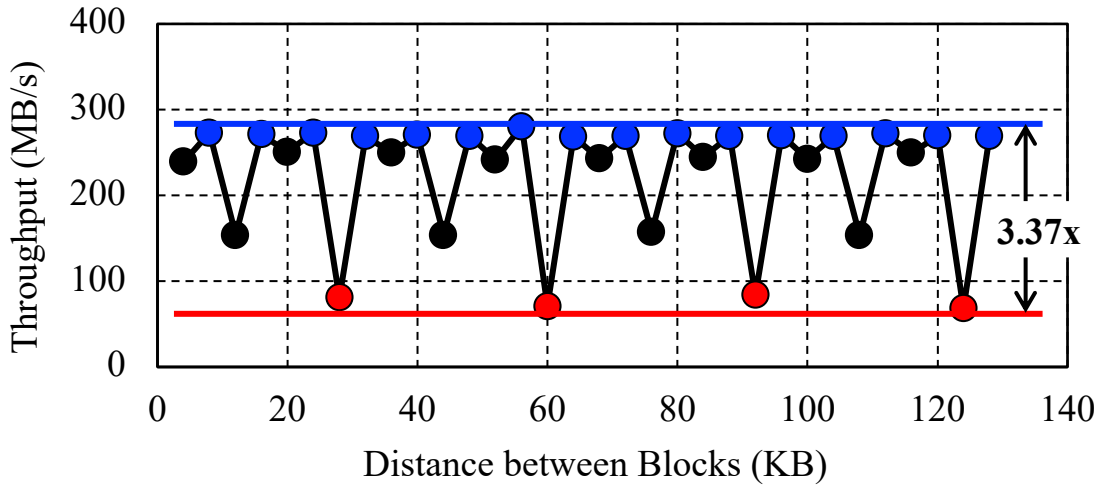




Fragmentation from the perspective of I/O control

- Causes
  - Increases the number of I/Os (32x)
  - Decreases the size of I/Os (4KB)
  - **Increases I/O randomness**
- Effects
  - CFQ: more I/Os for the same data
  - BFQ: more time for the same data
  - **IOCost:**
    fail to estimate device occupation





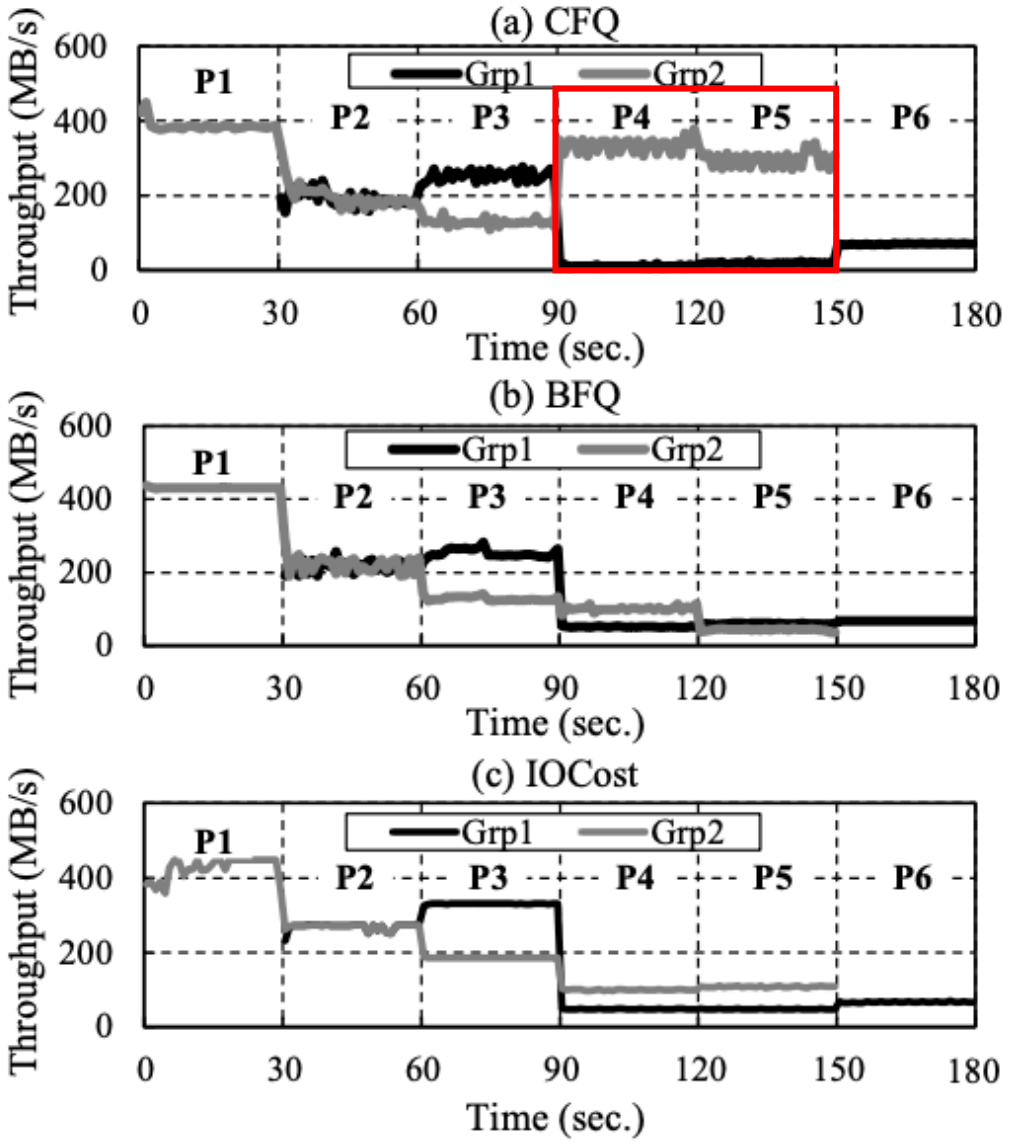
Fragmentation from the perspective of I/O control

[P1]: grp2 (100) - non-fragmented files
[P2]: grp1 (100), grp2 (100) - non-fragmented files
[P3]: grp1 (200), grp2 (100) - non-fragmented files
[P4]: grp1 (100) - fragmented files, grp2 (100) - non-fragmented files
[P5]: grp1 (200) - fragmented files, grp2 (100) - non-fragmented files
[P6]: grp1 (100) - fragmented files

CFQ (IOPS-based):
Problem: Unfair I/O sharing
Why?: more number of I/Os due to fragmentation

BFQ (Sector-based):
Problem: performance interference
Why?: smaller I/O size due to fragmentation

IOCost (Device occupation-based):
Problem: performance interference
Why?: irregular latency per I/O



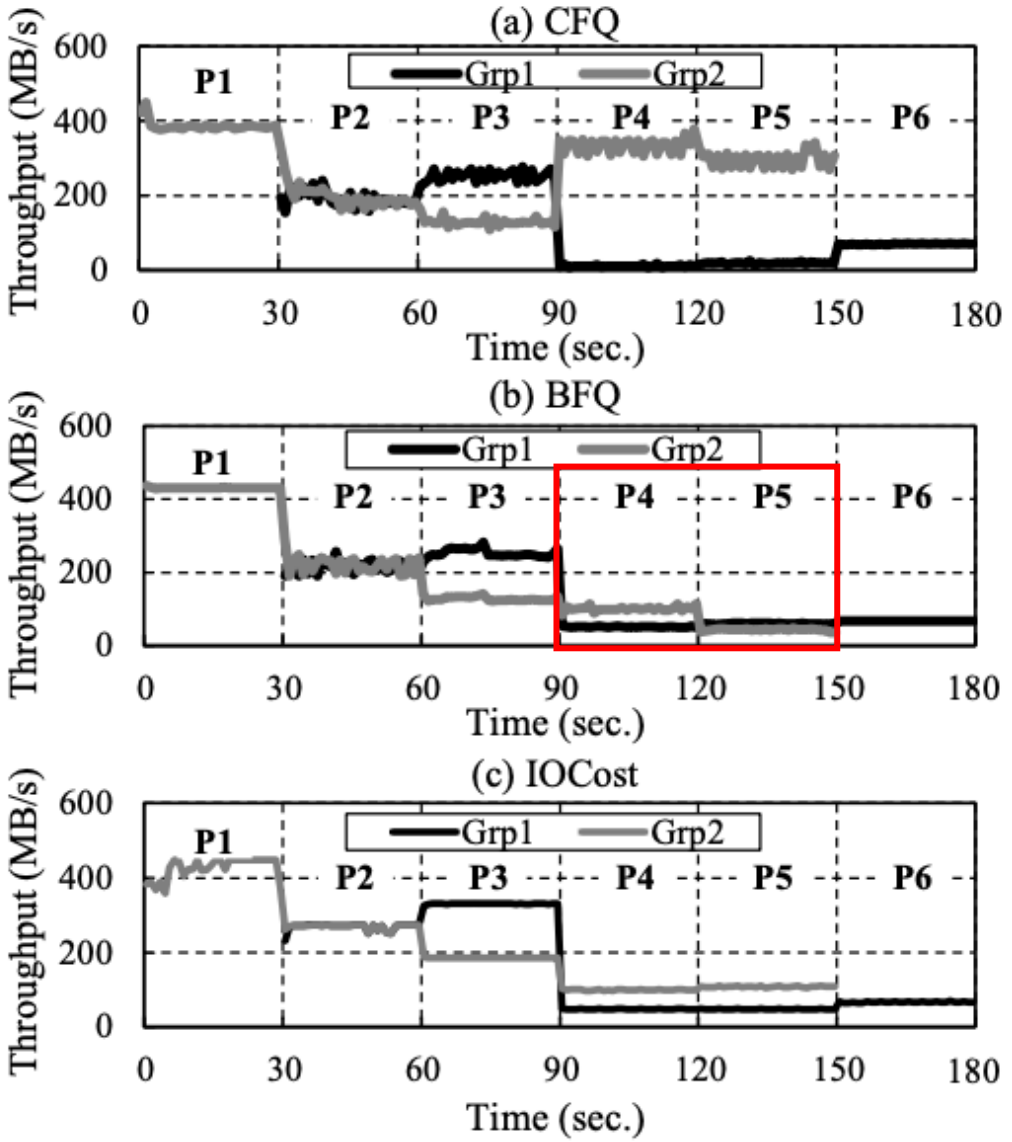
Fragmentation from the perspective of I/O control

[P1]: grp2 (100) - non-fragmented files
[P2]: grp1 (100), grp2 (100) - non-fragmented files
[P3]: grp1 (200), grp2 (100) - non-fragmented files
[P4]: grp1 (100) - fragmented files, grp2 (100) - non-fragmented files
[P5]: grp1 (200) - fragmented files, grp2 (100) - non-fragmented files
[P6]: grp1 (100) - fragmented files

CFQ (IOPS-based):
Problem: Unfair I/O sharing
Why?: more number of I/Os due to fragmentation

BFQ (Sector-based):
Problem: performance interference
Why?: smaller I/O size due to fragmentation

IOCost (Device occupation-based):
Problem: performance interference
Why?: irregular latency per I/O



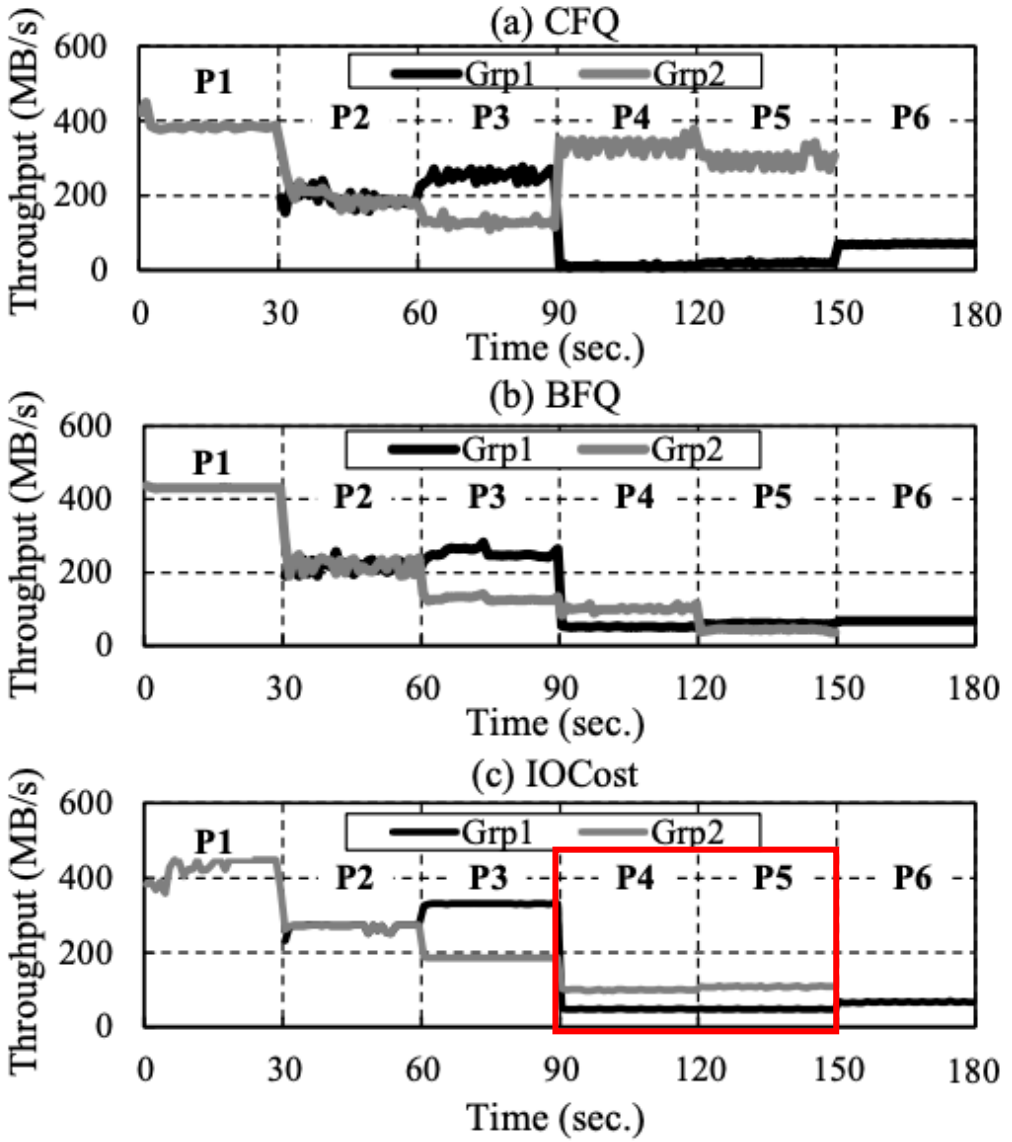
Fragmentation from the perspective of I/O control

[P1]: grp2 (100) - non-fragmented files
[P2]: grp1 (100), grp2 (100) - non-fragmented files
[P3]: grp1 (200), grp2 (100) - non-fragmented files
[P4]: grp1 (100) - fragmented files, grp2 (100) - non-fragmented files
[P5]: grp1 (200) - fragmented files, grp2 (100) - non-fragmented files
[P6]: grp1 (100) - fragmented files

CFQ (IOPS-based):
Problem: Unfair I/O sharing
Why?: more number of I/Os due to fragmentation

BFQ (Sector-based):
Problem: performance interference
Why?: smaller I/O size due to fragmentation

IOCost (Device occupation-based):
Problem: performance interference
Why?: irregular latency per I/O



Possible solutions

- Better I/O control mechanisms
  - More accurate performance model of IOCost
  - New I/O control mechanism considering wall-clock time
- Scheduling at different layers
  - System call layer
  - Inside storage
  - Multiple layers
- **Defragmentation**

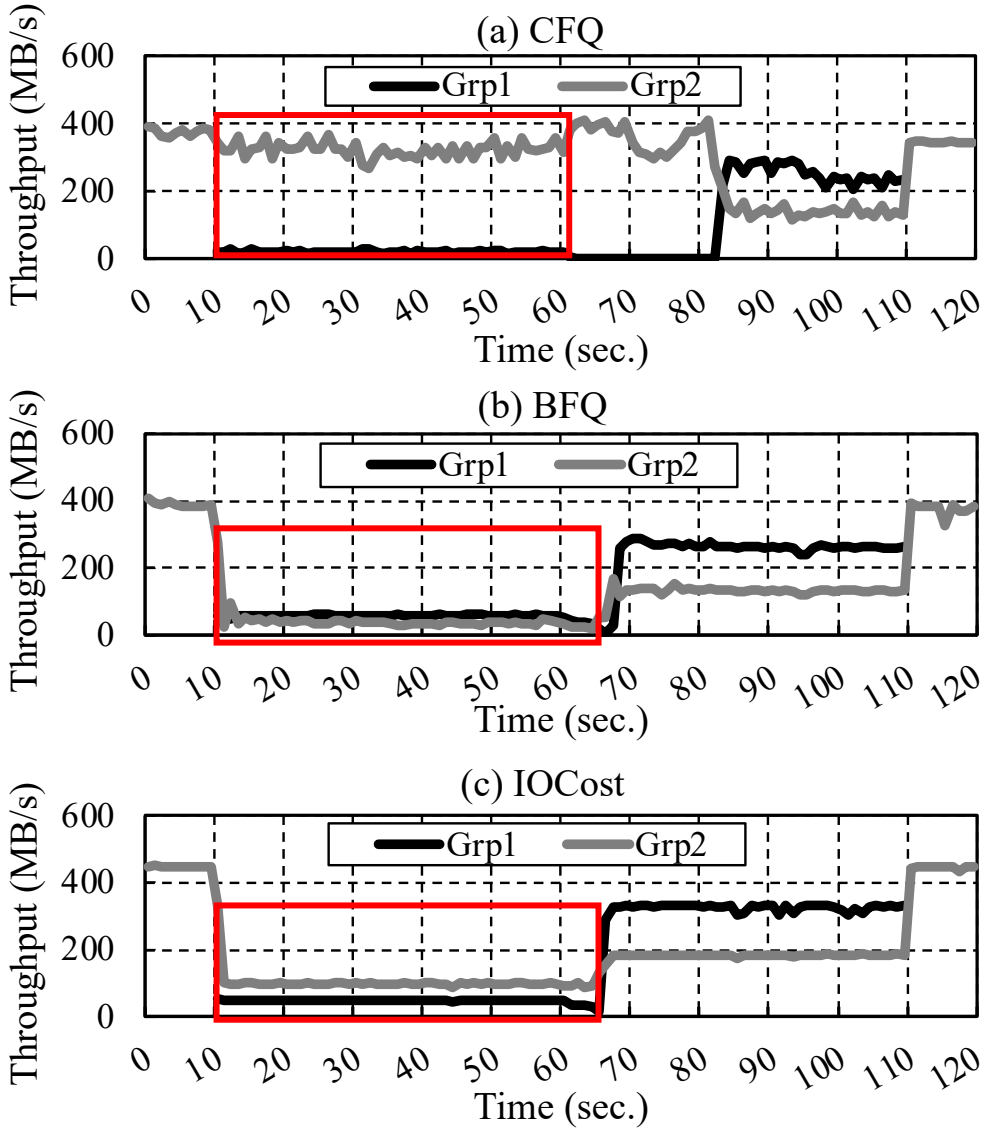
Defragmentation as an antidote

- Evaluation Setup
  - Storage: Samsung 850 Pro
  - Workload-A: two 128KB sequential read workload
  - Workload-B: SQLite SELECT query while running 64KB rand. read (qd: 16)
  - Kernel: Linux kernel 4.19.176 for CFQ and 5.4.0 for BFQ/IOCost
  - Filesystem: F2FS
  - Defragmentation tool: FragPicker

Defragmentation as an antidote (Synthetic)

Before defragmentation,

- CFQ: Improper scheduling
- BFQ: performance interference
- IOCost: performance interference

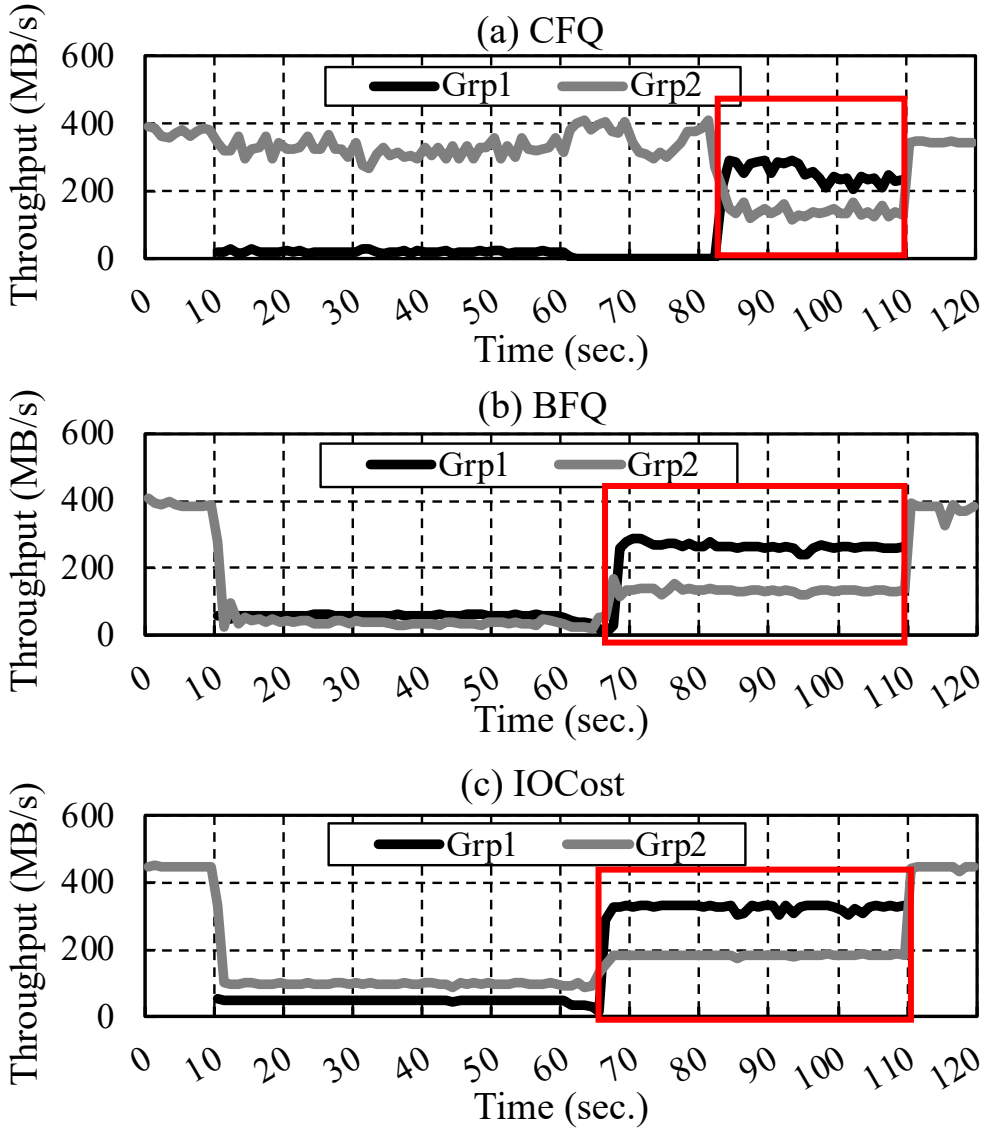


Defragmentation as an antidote (Synthetic)

Before defragmentation,

- CFQ: Improper scheduling
- BFQ: performance interference
- IOCost: performance interference

Defragmentation relieves the problem



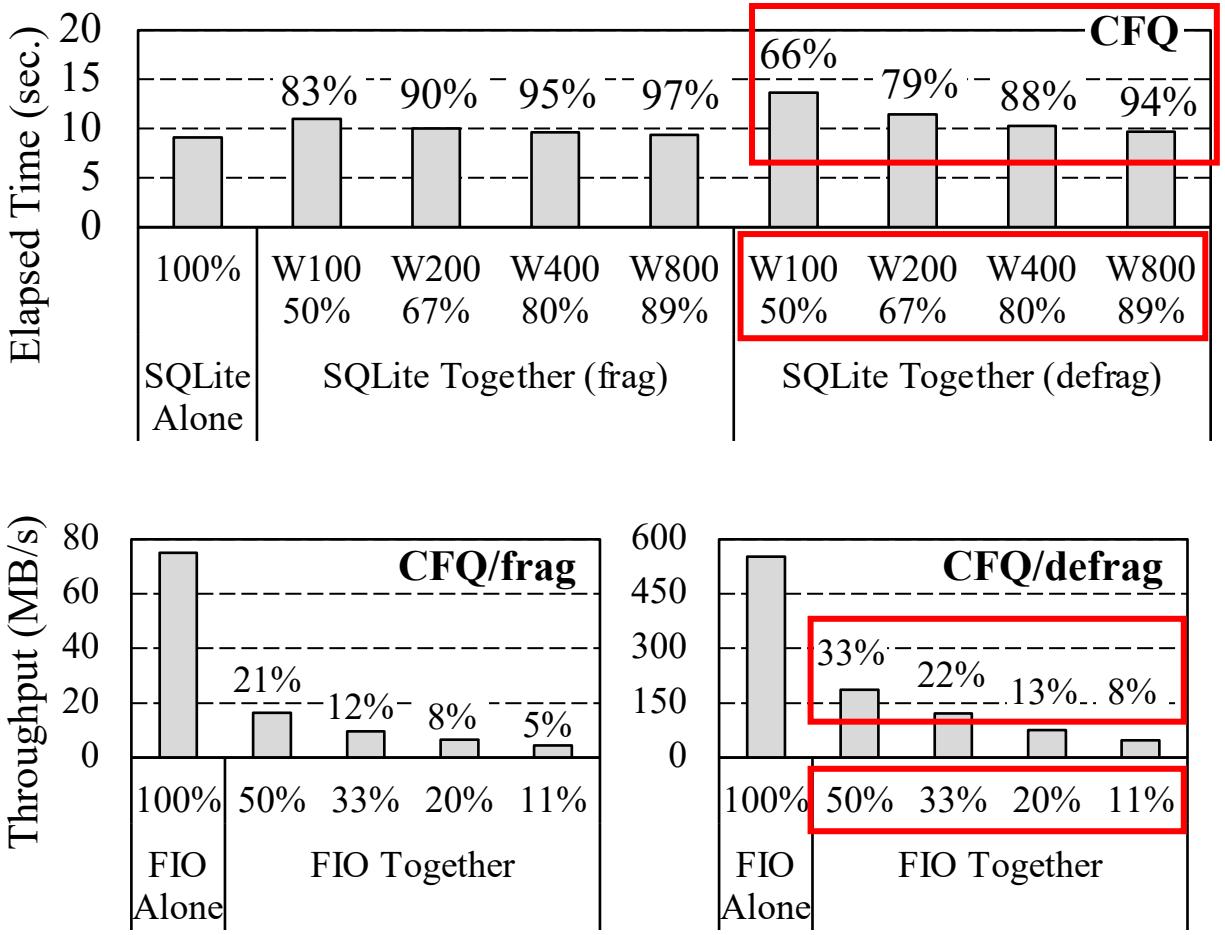
Defragmentation as an antidote (SQLite/FIO)

Before defragmentation, the victim is

- CFQ: FIO
- BFQ: SQLite
- IOCost: SQLite

After defragmentation

- CFQ: reduces the number of I/Os
- BFQ: increase the average I/O size
- IOCost: stabilize the I/O performance



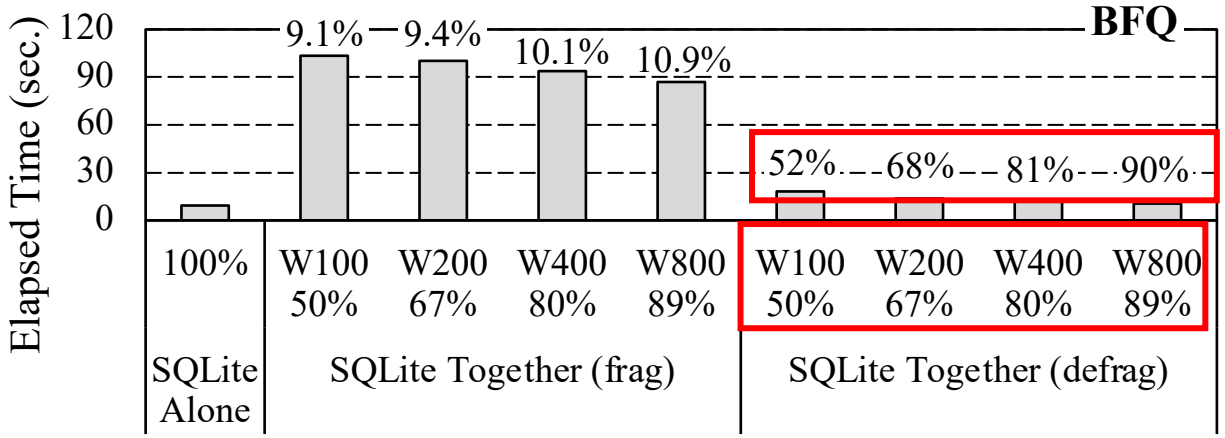
Defragmentation as an antidote (SQLite/FIO)

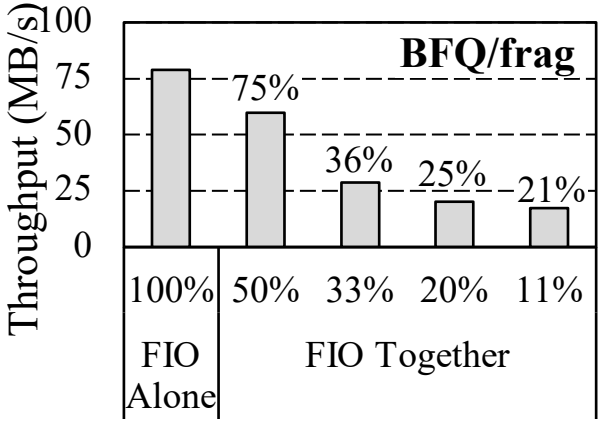
Before defragmentation, the victim is

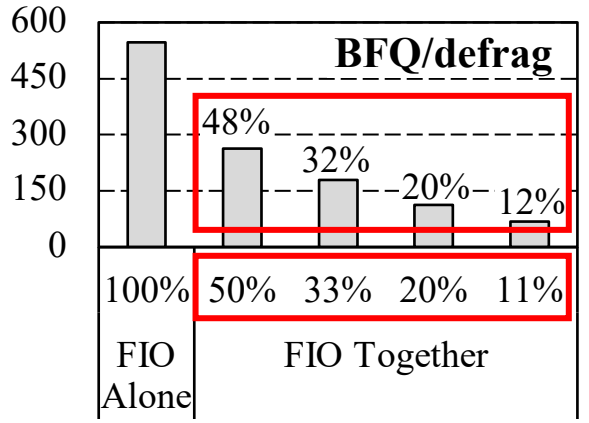
- CFQ: FIO
- BFQ: SQLite
- IOCost: SQLite

After defragmentation

- CFQ: reduces the number of I/Os
- BFQ: increase the average I/O size
- IOCost: stabilize the I/O performance







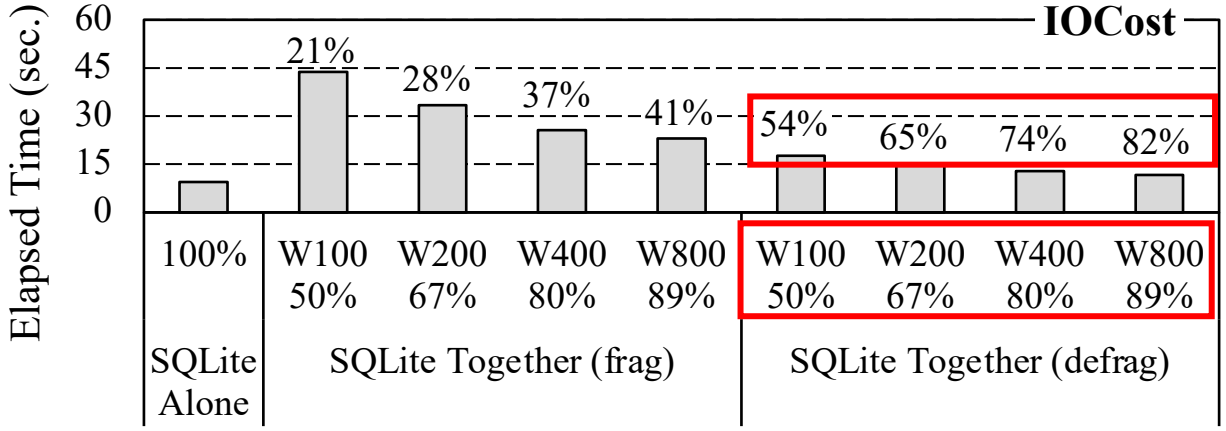
Defragmentation as an antidote (SQLite/FIO)

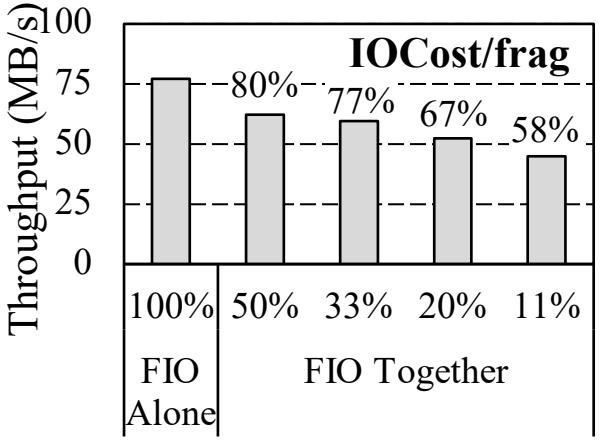
Before defragmentation, the victim is

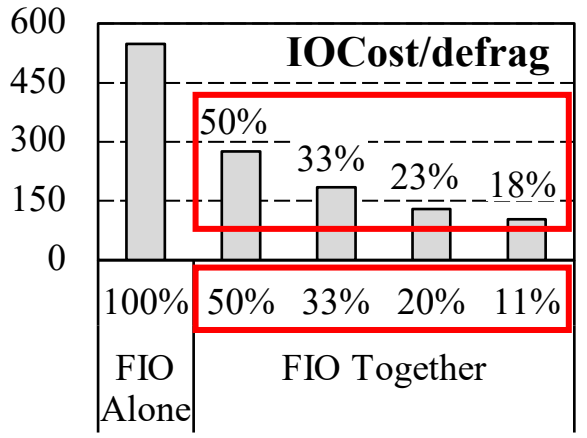
- CFQ: FIO
- BFQ: SQLite
- IOCost: SQLite

After defragmentation

- CFQ: reduces the number of I/Os
- BFQ: increase the average I/O size
- IOCost: stabilize the I/O performance







Takeaways

- Fragmentation is toxic to I/O control
  - CFQ: toxic to the corresponding cgroup that uses fragmented data
  - BFQ/IOCost: toxic to other cgroups (performance interference)
- Defragmentation as an antidote
  - CFQ: defrag the data of the cgroup that experiences performance degradation
  - BFQ/IOCost: defrag the data of other cgroups

Thank you

Contact: jonggyu@skku.edu