Improving File System Performance of Mobile Storage Systems Using a Decoupled Defragmenter

Sangwook Shane Hahn[*], Sungjin Lee[†], Cheng Ji[‡], Li-Pin Chang[+],
Inhyuk Yee[*], Liang Shi[#], Chun Jason Xue[‡] and Jihong Kim[*]

[*]Seoul National University
[†]Daegu Gyeongbuk Institute of Science and Technology (DGIST)
[‡]City University of Hong Kong
[+]National Chiao-Tung University
[#]Chongqing University

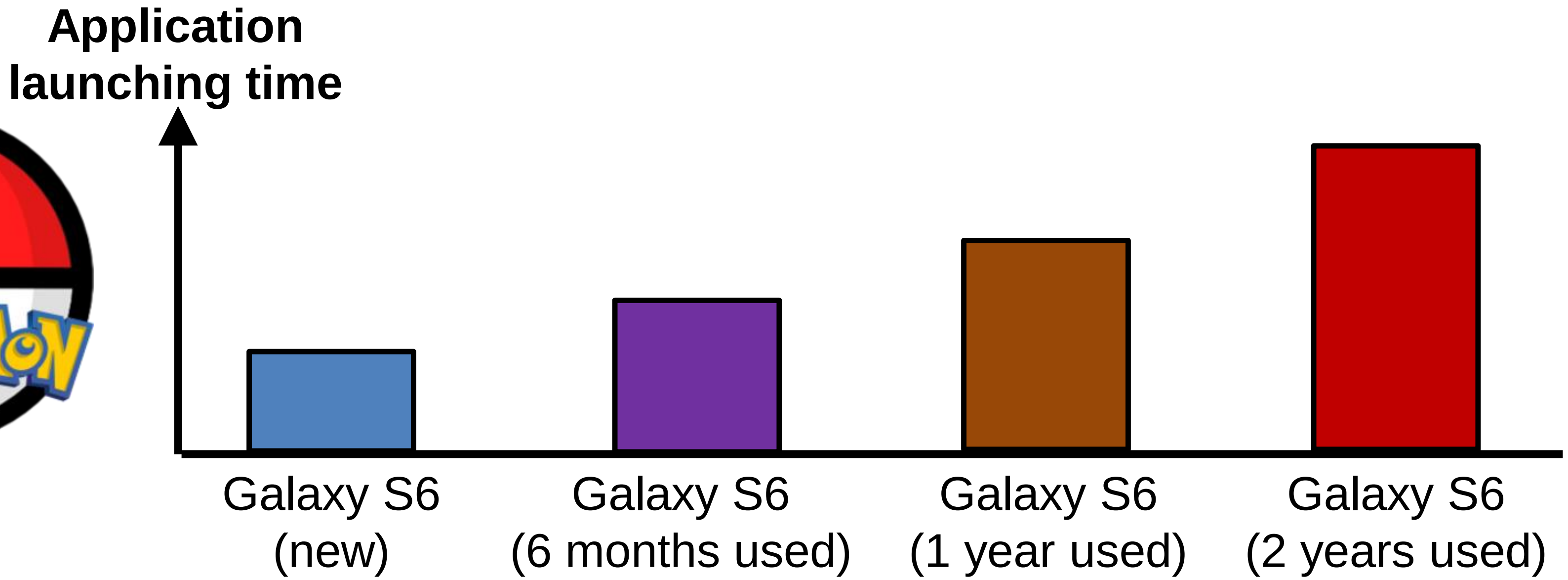
USENIX Annual Technical Conference, 2017

Outline

- Impact of File Fragmentation/Defragmentation
- Key Observations on Flash-based File Fragmentation
  - ◆ Decoupled Fragmentation
  - ◆ Dominant Impact of Logical Fragmentation
- Janusd: a Decoupled Defragmenter
- Experimental Results
- Conclusions

Gradual Performance Degradation on Smartphones

Performance of Android smartphones ***gradually degrades*** **as smartphones** ***age***

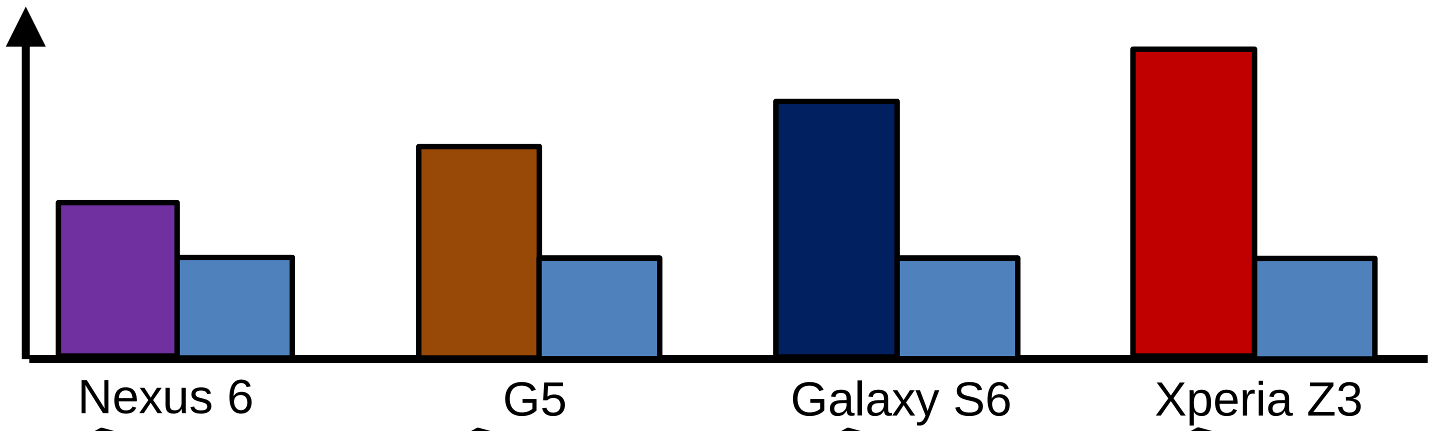


Application launching times increase up to 3 times on 2-year used smartphones

Root Cause: File Fragmentation

Q: **Why** does performance degrade ?

A: File fragmentation



16% files fragmented

22% files fragmented

27% files fragmented

34% files fragmented

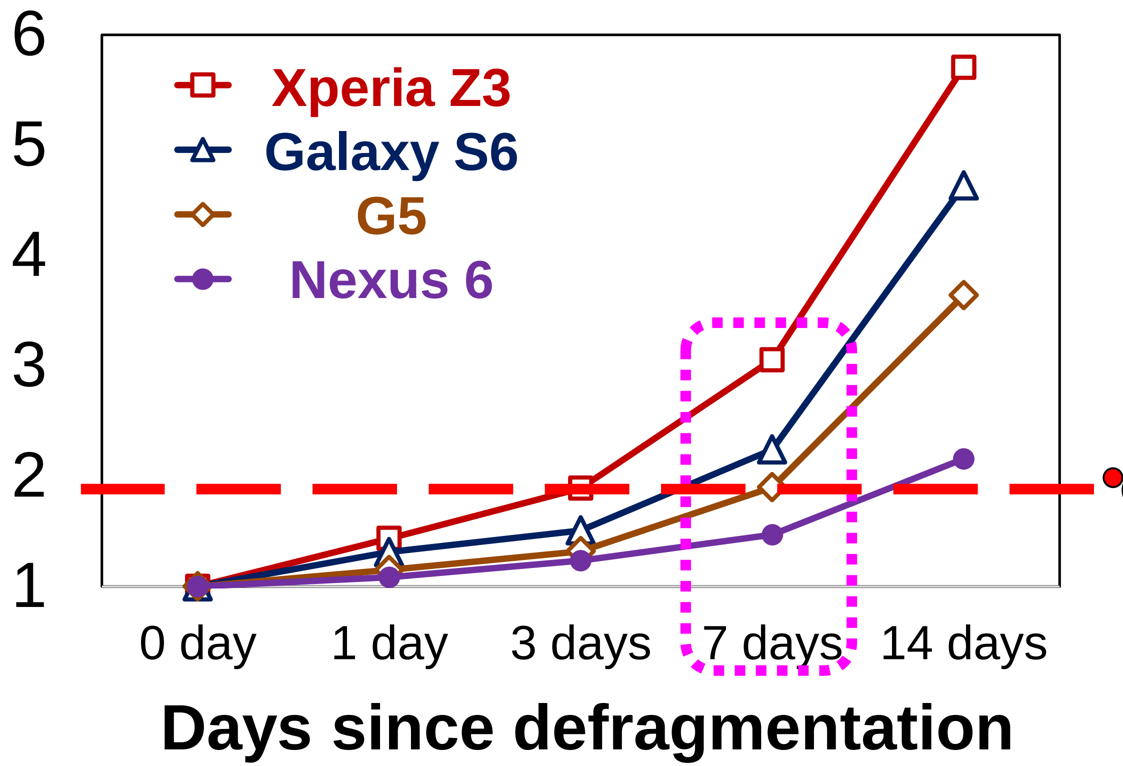
Defragmentation ***can*** ***improve*** ***the degraded performance*** ***by fragmentation***

File Fragmentation Recurrence on Smartphones

Q: How often should we defrag smartphones ?

Average
# of fragments
per file

Degree of
file fragmentation

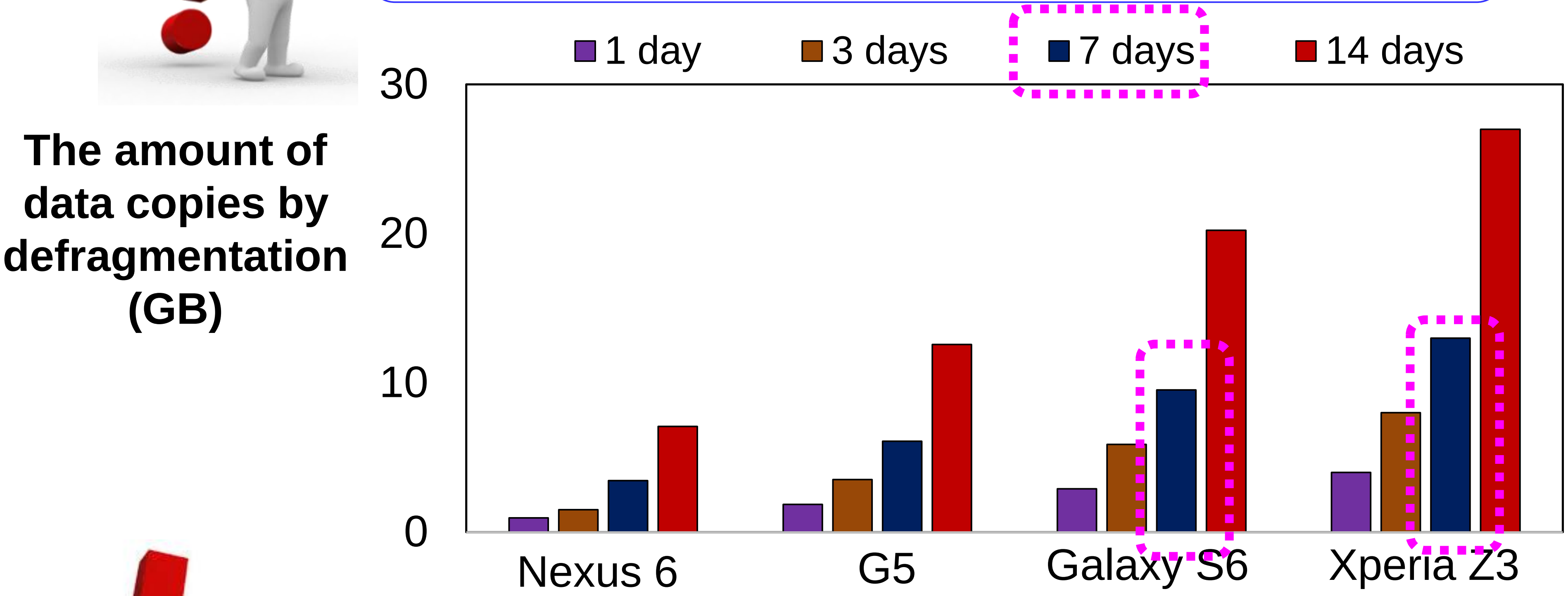


User begins to experience performance degradation

File fragmentation recurs even in a week

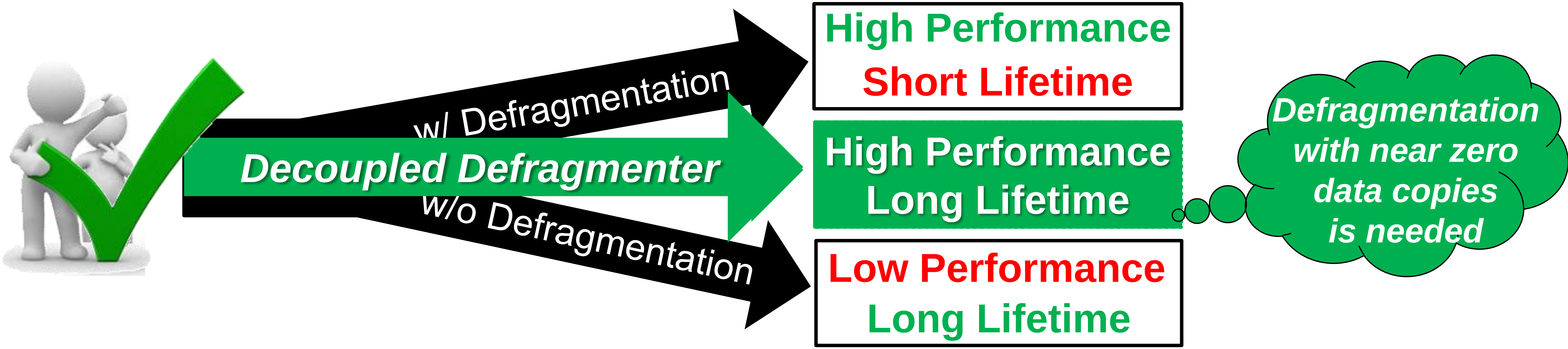
Lifetime Impact of Frequent Defragmentation

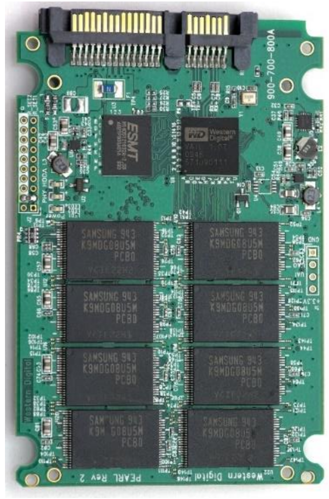
Q: Are there any **side effects** from frequent defragmentation?



Weekly defragmentation can reduce the storage lifetime by more than 10%

Summary: Impact of File Fragmentation/Defragmentation



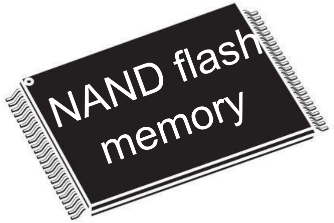


NAND Flash-based Storage

File fragmentation in NAND flash-based storage is *quite different* from conventional one in HDD

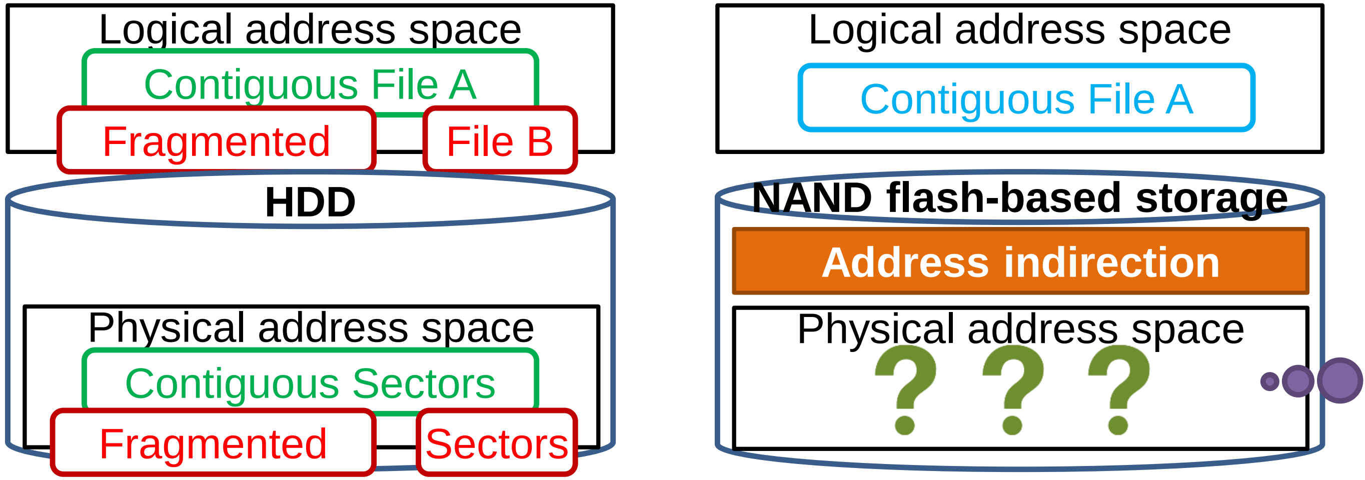
1. *Decoupled fragmentation*
2. *High overhead of logical fragmentation*

Observation 1: Decoupled Fragmentation



All data are stored using *address indirection*

Fragmentation at ***logical space*** and ***physical space***



Logical address space

Contiguous File A

Fragmented

File B

HDD

Physical address space

Contiguous Sectors

Fragmented

Sectors

Logical address space

Contiguous File A

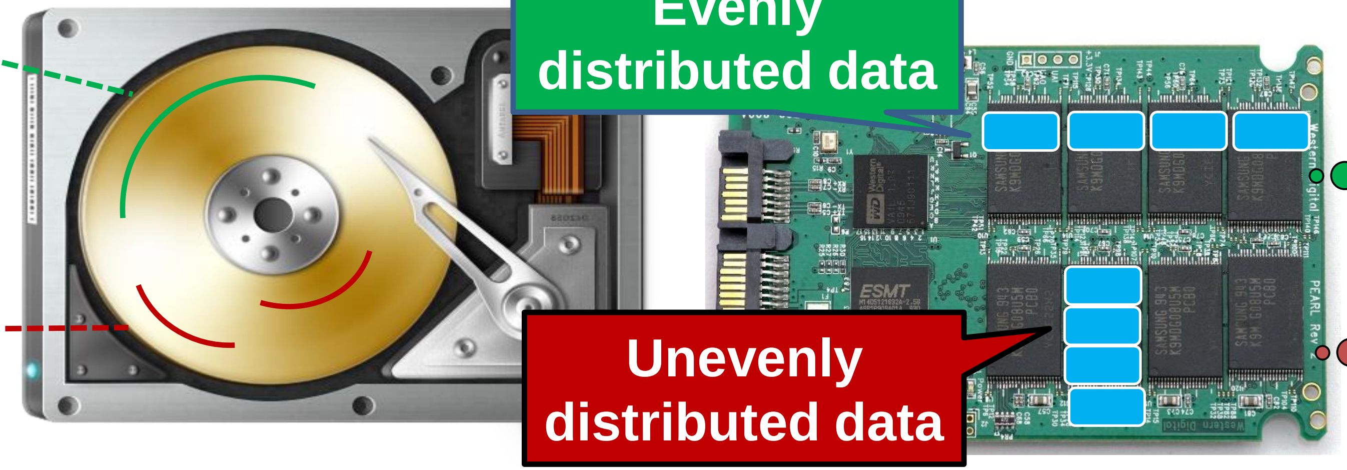
NAND flash-based storage

Address indirection

Physical address space

? ? ?

Physical fragmentation



Contiguous File A

Fragmented File B

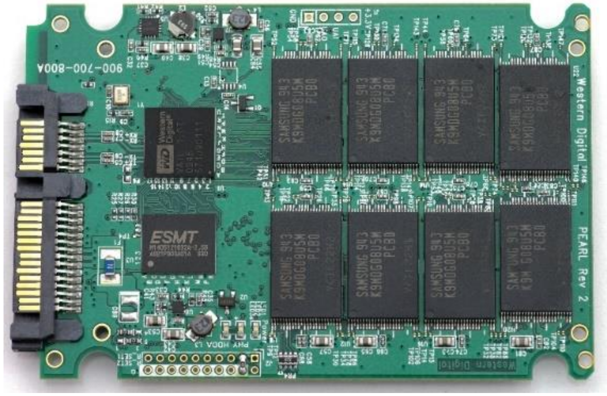
Evenly distributed data

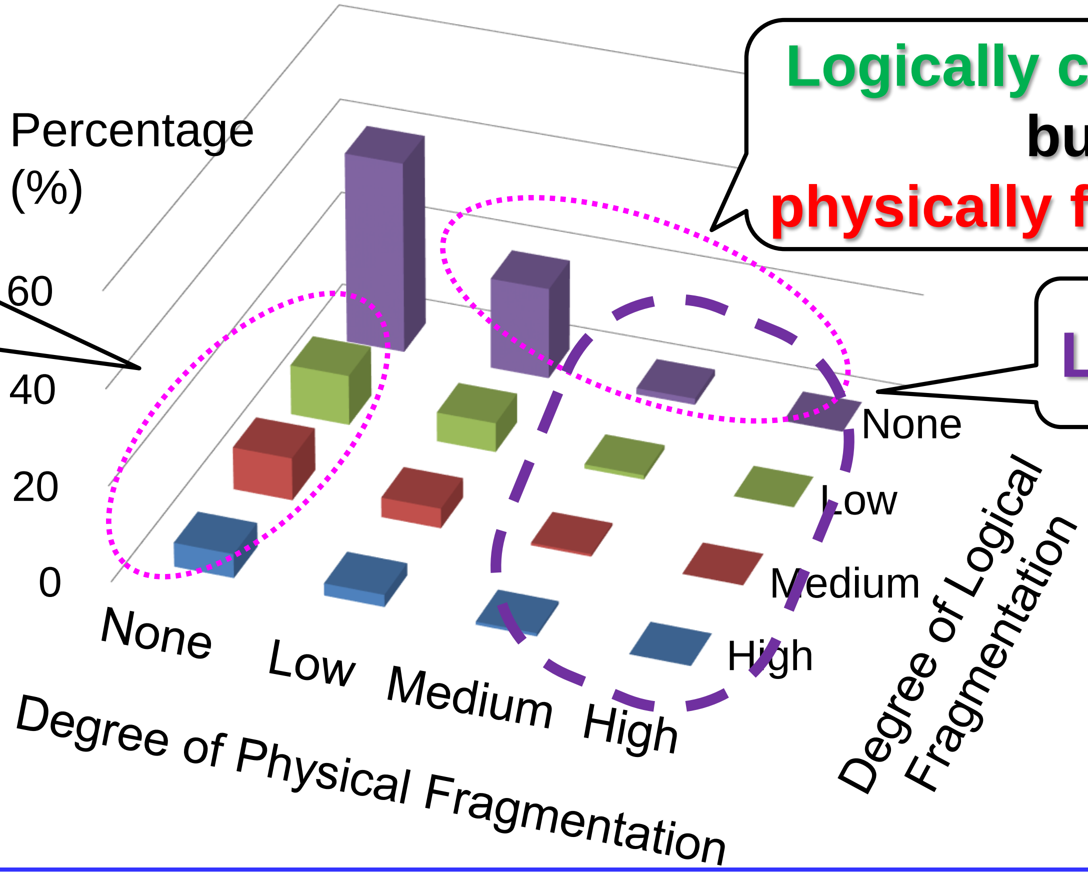
Unevenly distributed data

High degree of I/O parallelism

Low degree of I/O parallelism

Observation 1: Decoupled Fragmentation




Percentage (%)
60
40
20
0
None
Low
Medium
High
Degree of Physical Fragmentation
None
Low
Medium
High
Degree of Logical Fragmentation

Logically fragmented but, physically contiguous

Logically contiguous but, physically fragmented

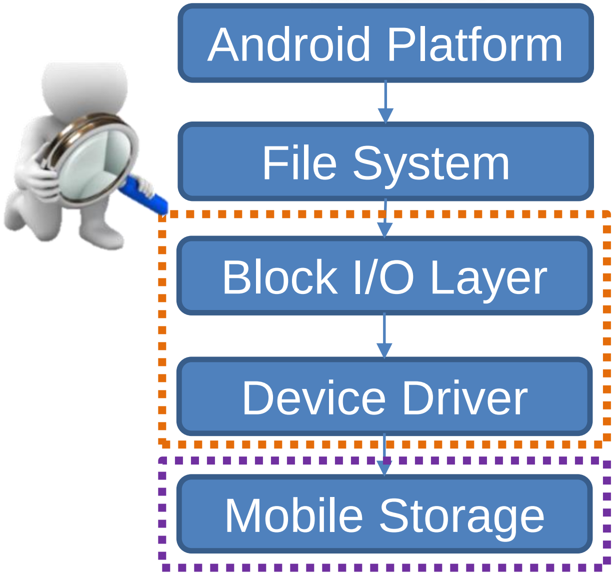
Less than 1%

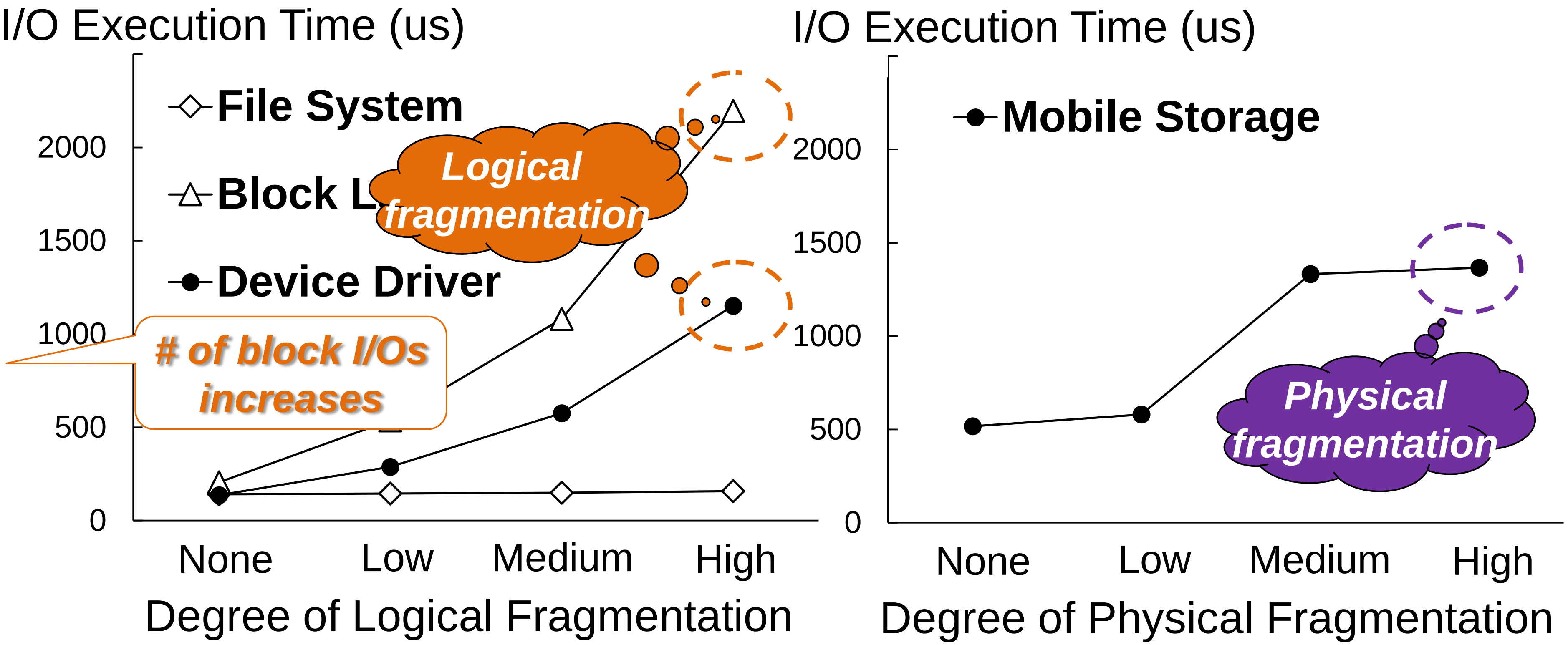
1. There is no correlation between logical/physical fragmentation

2. Physical fragmentation rarely occurs

Observation 2: High Overhead of Logical Fragmentation

Q: How much the impact of logical/physical fragmentation on performance?





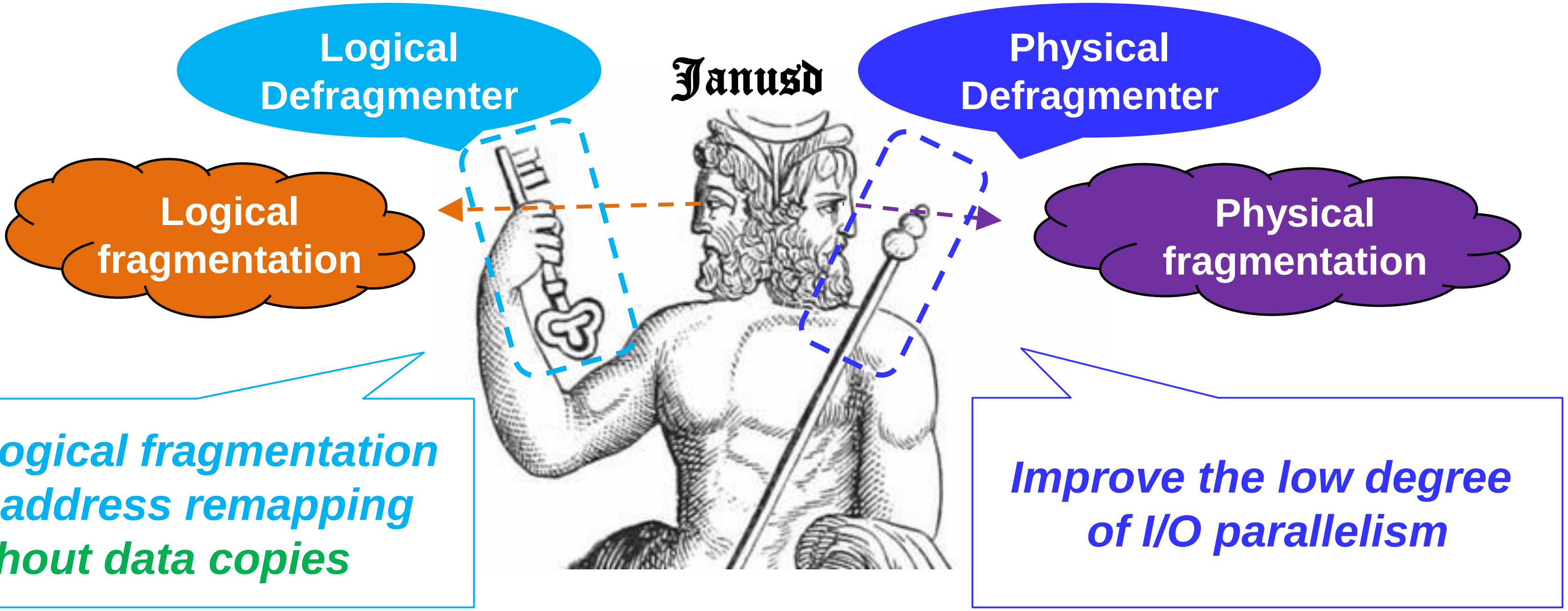
Logical fragmentation

# of block I/Os increases

Physical fragmentation

Logical fragmentation overhead overwhelms physical fragmentation overhead

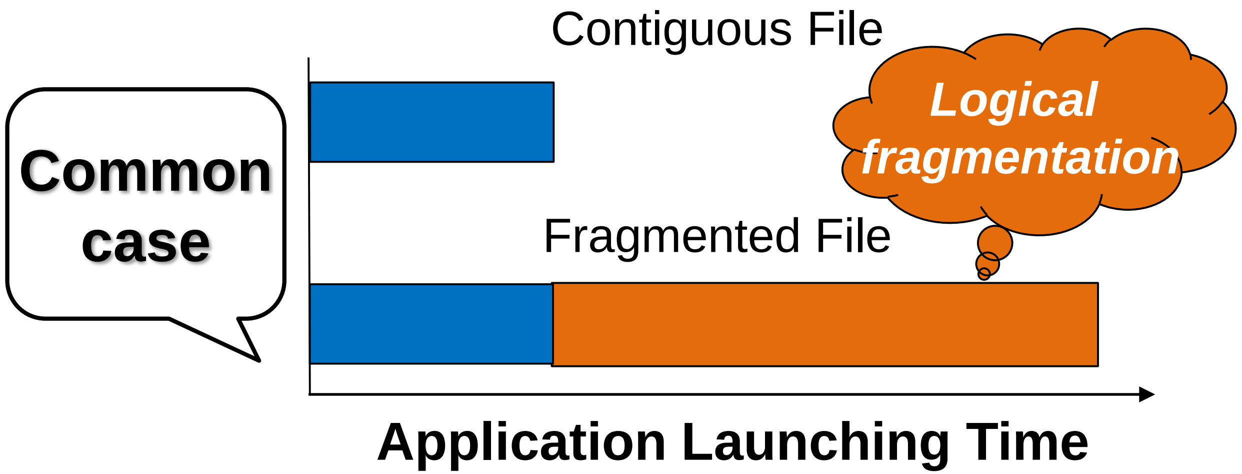
Solution for Decoupled Fragmentation

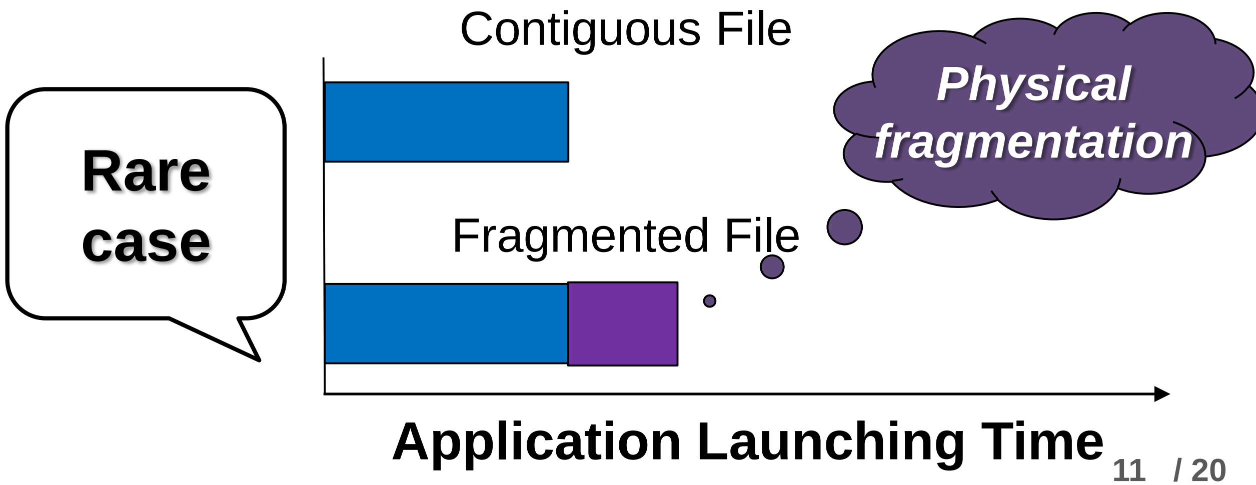


Defrag logical fragmentation using address remapping without data copies

Improve the low degree of I/O parallelism

Janus Defragmenter



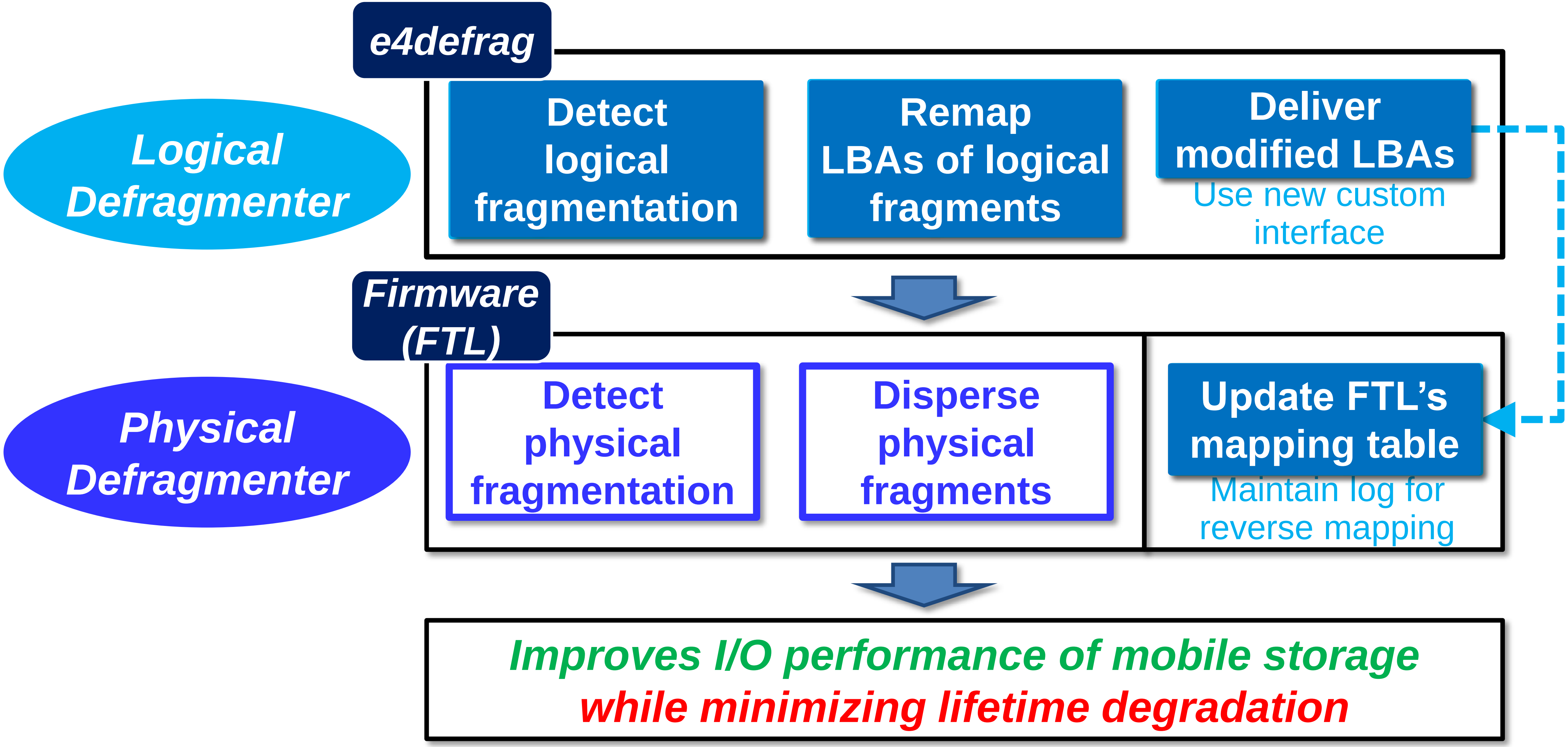


Outline

- Impact of File Fragmentation/Defragmentation
- Key Observations on Flash-based File Fragmentation
  - ◆ Decoupled Fragmentation
  - ◆ Dominant Impact of Logical Fragmentation
- **Janusd: a Decoupled Defragmenter**
- **Experimental Results**
- **Conclusions**

Overview of Decoupled Defragmenter

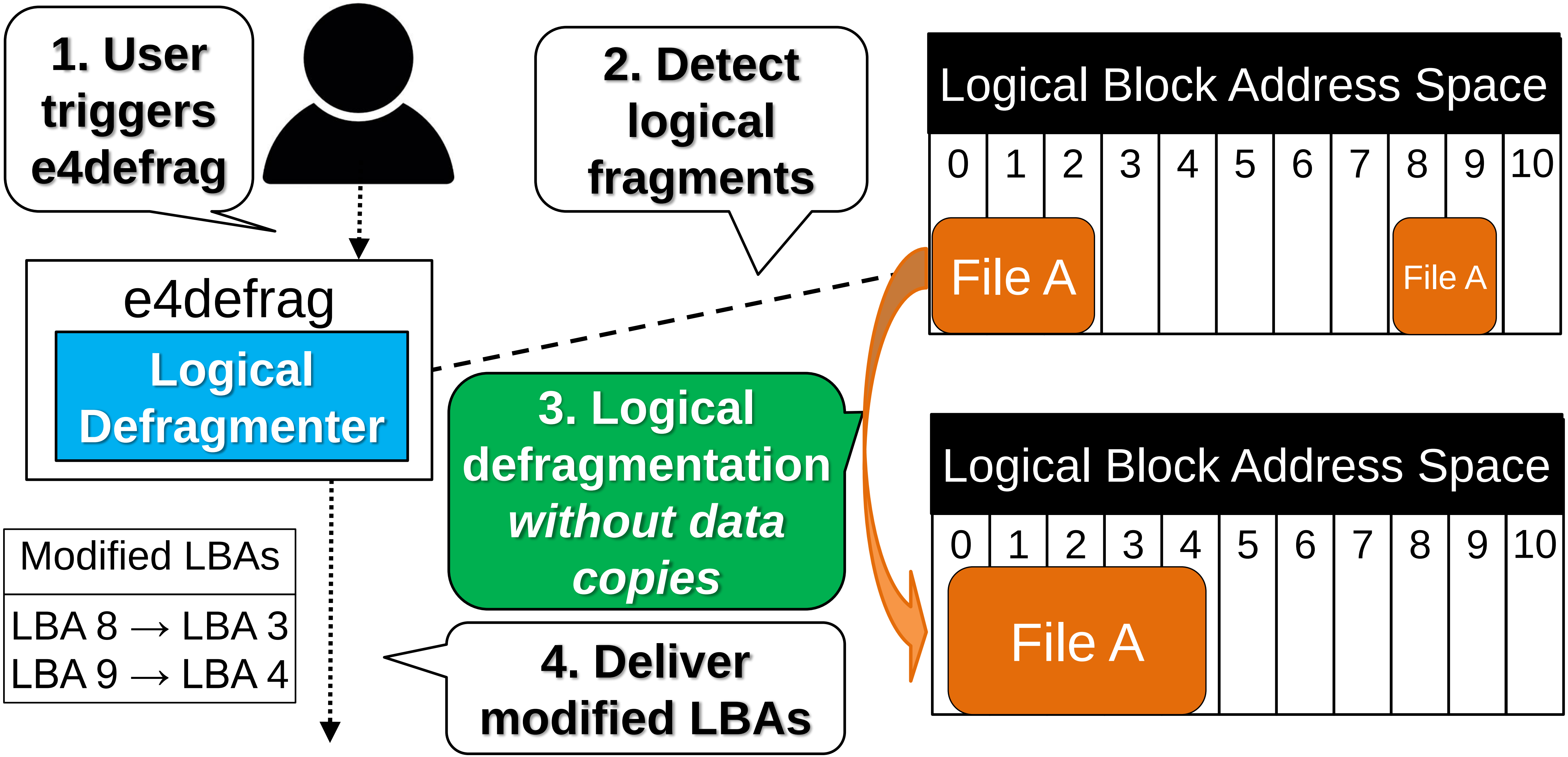
Decoupled Defragmenter (Janusd)



e4defrag
Logical Defragmenter
Detect logical fragmentation
Remap LBAs of logical fragments
Deliver modified LBAs
Use new custom interface
Firmware (FTL)
Physical Defragmenter
Detect physical fragmentation
Disperse physical fragments
Update FTL's mapping table
Maintain log for reverse mapping

Improves I/O performance of mobile storage while minimizing lifetime degradation

Logical Defragmenter (JanusdL)



1. User triggers e4defrag

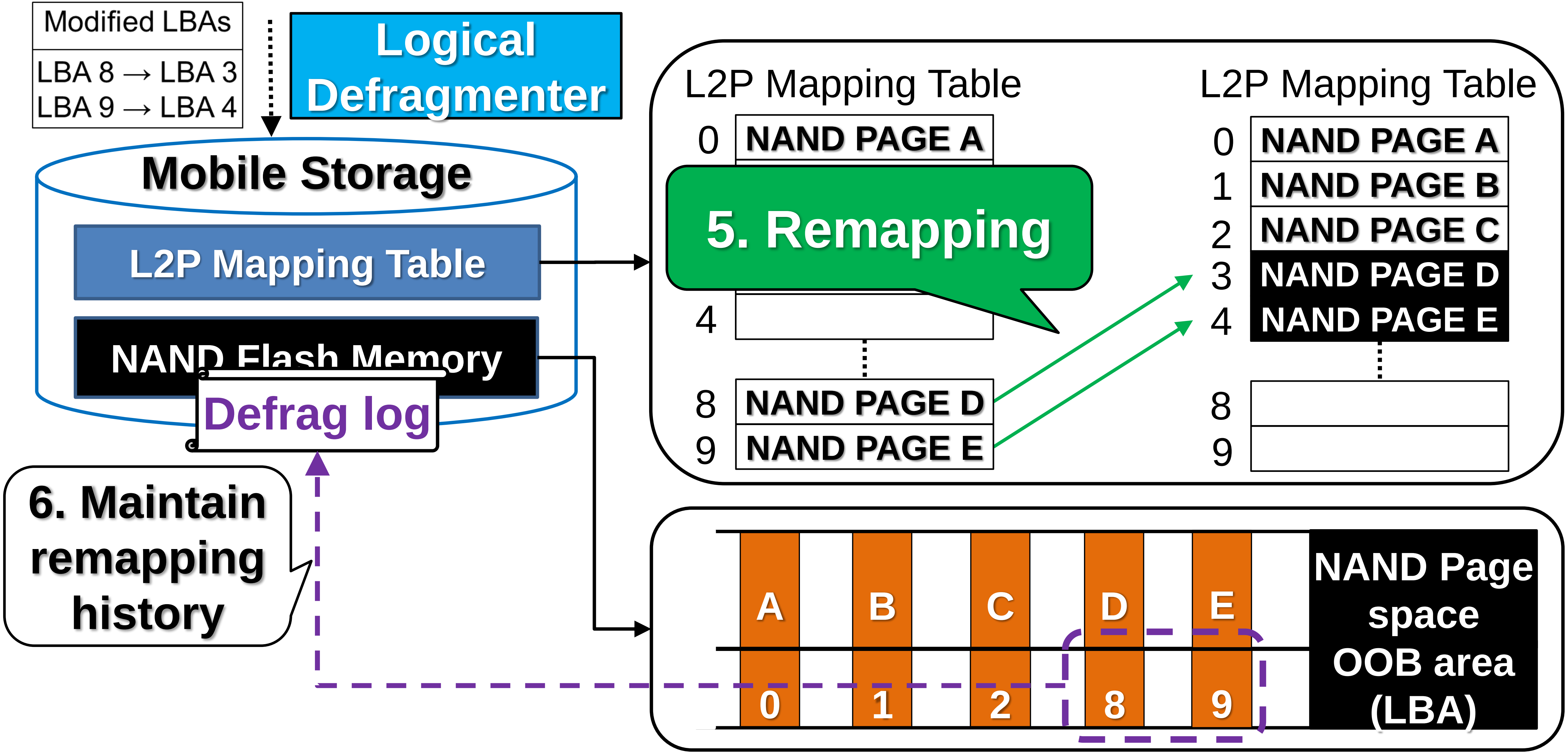
2. Detect logical fragments

3. Logical defragmentation *without data copies*

4. Deliver modified LBAs

Modified LBAs
LBA 8 → LBA 3 LBA 9 → LBA 4

Logical Defragmenter (JanusdL)



Modified LBAs
LBA 8 → LBA 3
LBA 9 → LBA 4

Logical Defragmenter

Mobile Storage

L2P Mapping Table

NAND Flash Memory

Defrag log

6. Maintain remapping history

5. Remapping

L2P Mapping Table

0	NAND PAGE A
4	
⋮	
8	NAND PAGE D
9	NAND PAGE E

L2P Mapping Table

0	NAND PAGE A
1	NAND PAGE B
2	NAND PAGE C
3	NAND PAGE D
4	NAND PAGE E
⋮	
8	
9	

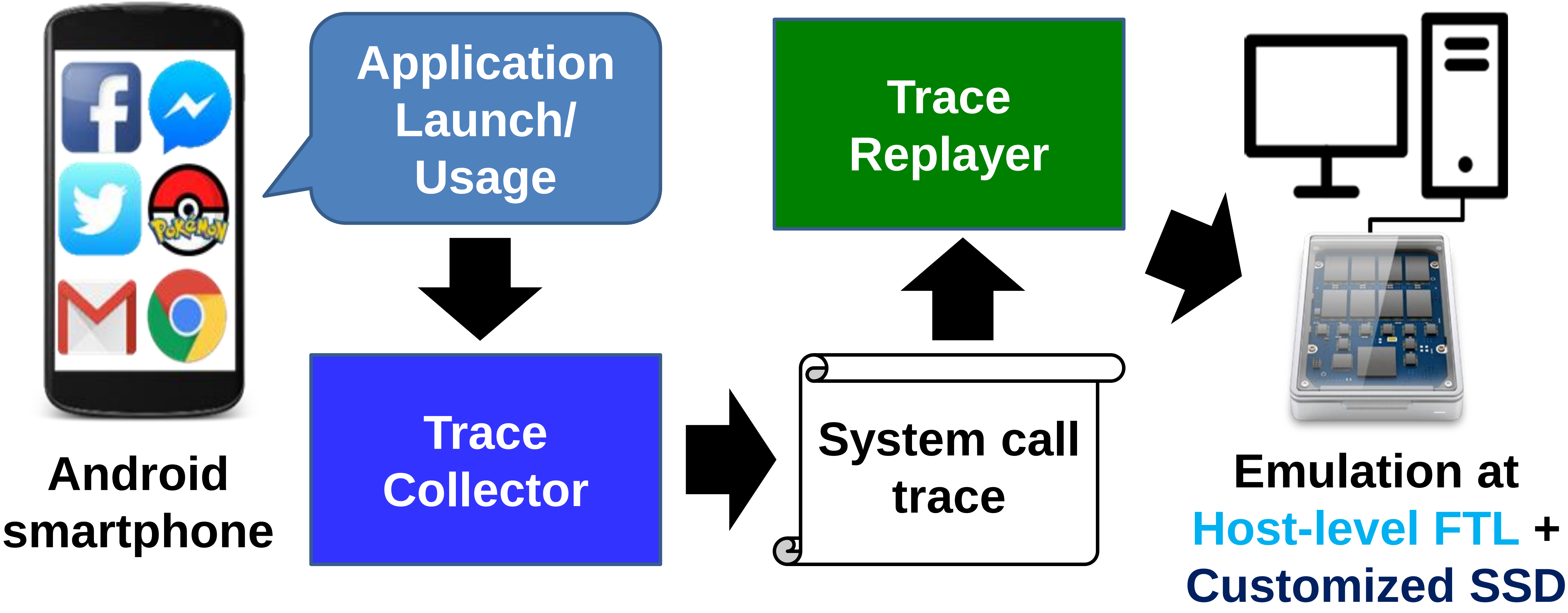
A	B	C	D	E	NAND Page space
0	1	2	8	9	OOB area (LBA)

Evaluation Scenarios

- **We collected six different application usage traces**
  - ➢ **Application launching scenarios**
  - ➢ **Interactive application usage scenario (10 minutes)**

Scenario	Scenario Description
Chrome	Launching app → Viewing webpages
Messenger	Launching app → Viewing chat records
Gmail	Launching app → Viewing emails
Facebook	Launching app → Viewing online news
Twitter	Launching app → Viewing online news
Game	Launching Pokemon Go → Playing game

Experimental Settings



Android smartphone

Application Launch/ Usage

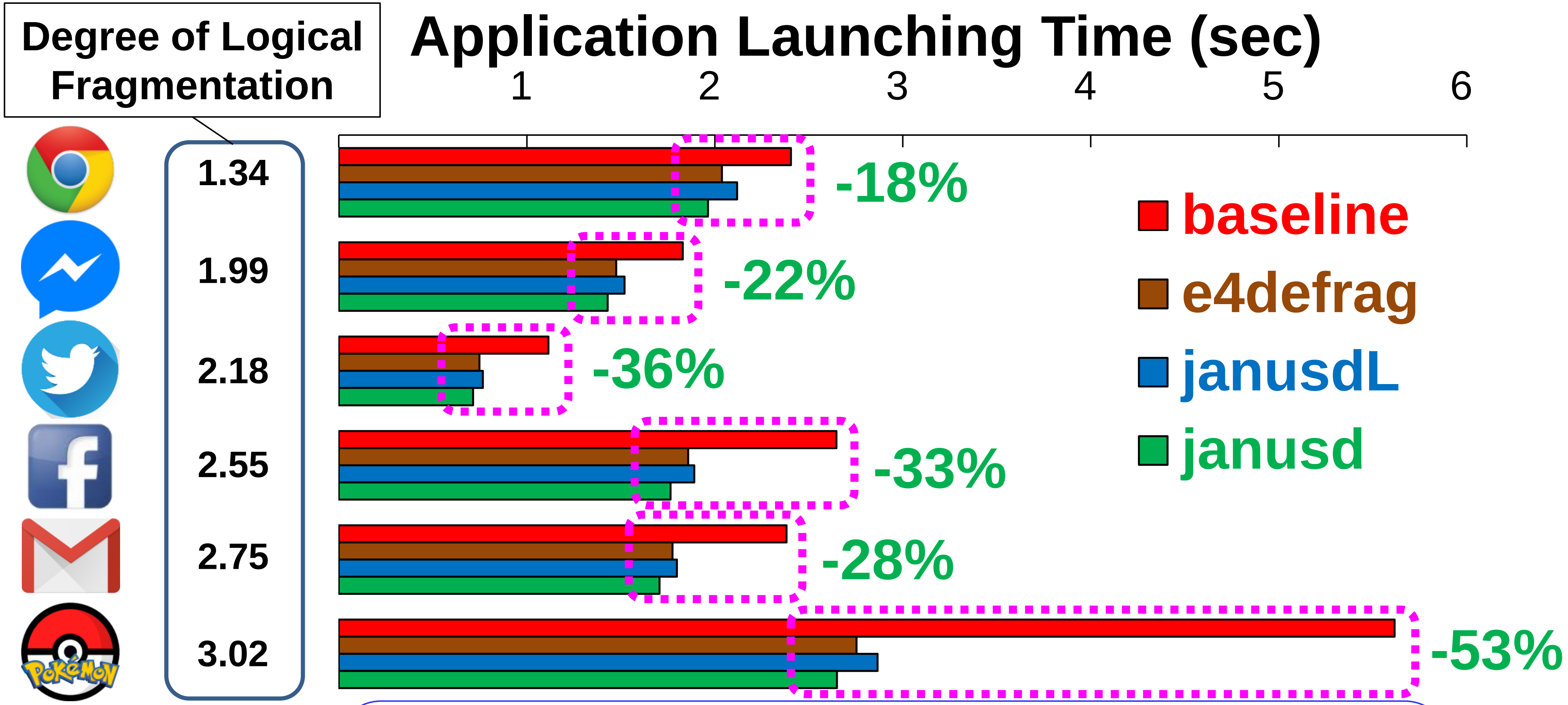
Trace Collector

System call trace

Trace Replayer

Emulation at Host-level FTL + Customized SSD

Result 1: Application Launching Time

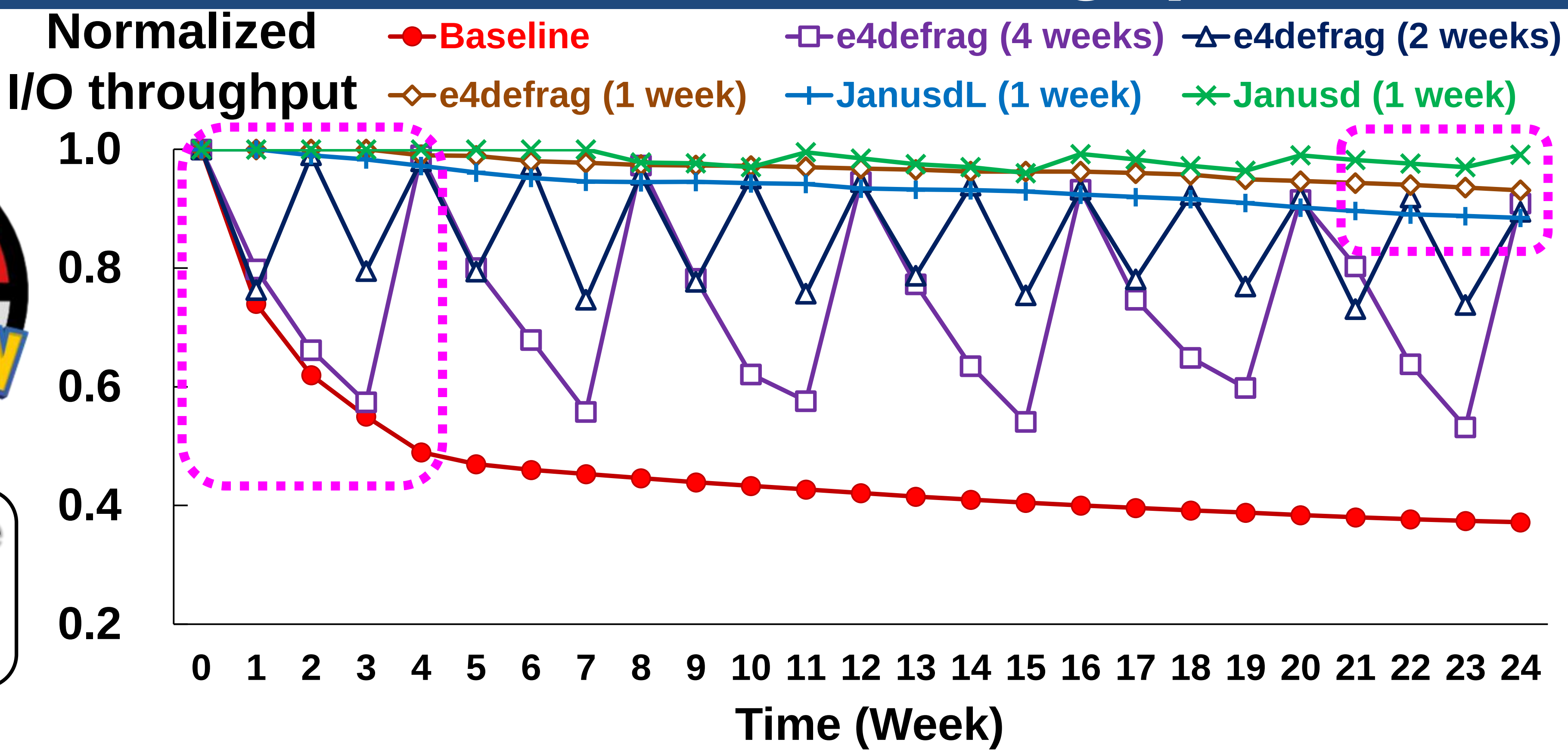


1. The more file fragmentation, the greater the performance improvement

2. Janusd achieves better performance than conventional defragmenter (e4defrag)

Result 2: I/O Throughput

10 minute usage scenario



1. Performance degradation *occurs* even when we defrag smartphone every 2 weeks

2. Conventional defragmenter has limitations in solving physical fragmentation

Conclusion

- **We have presented a decoupled defragmenter for improving the file system performance**
  - **JanusdL defrags logical fragmentation without data copies by remapping LBAs with FTL's mapping table**
  - **JanusdP defrags physical fragmentation by improving I/O parallelism of files**
  - **Improved application launching times by 32% on average**
  - **Reduced the amount of data copies by 99.99% on average**

- **Future expends**
  - **Free space defragmentation tool**
  - **Defrag-on-write() which triggers JanusdL right before write()**

감사합니다
Natick
Grazie
Danke
Ευχαριστίες
Dalu
Obrigado
Thank You
Köszönöm
Tack
Спасибо
Dank
Gracias
谢谢
Merci
Seé
ありがとう