File System Aging

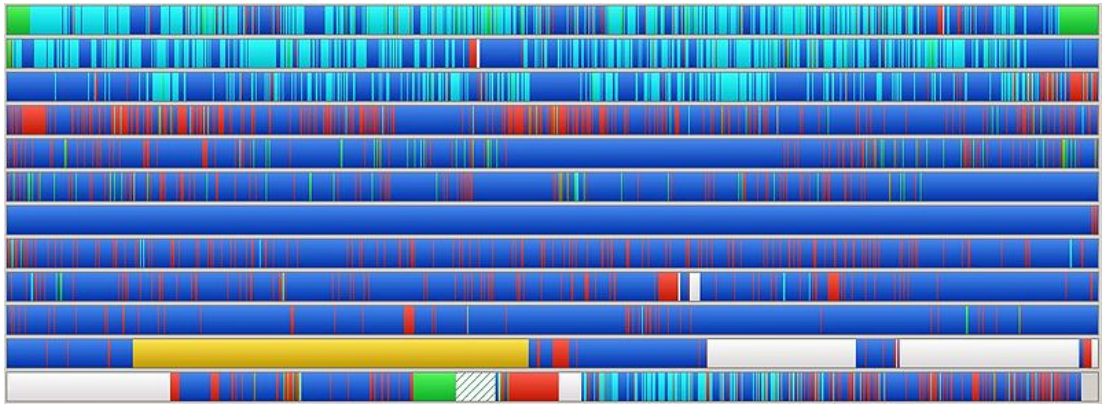
January 20, 2020

Fabian Hofbeck

Friedrich-Alexander-Universität Erlangen-Nürnberg

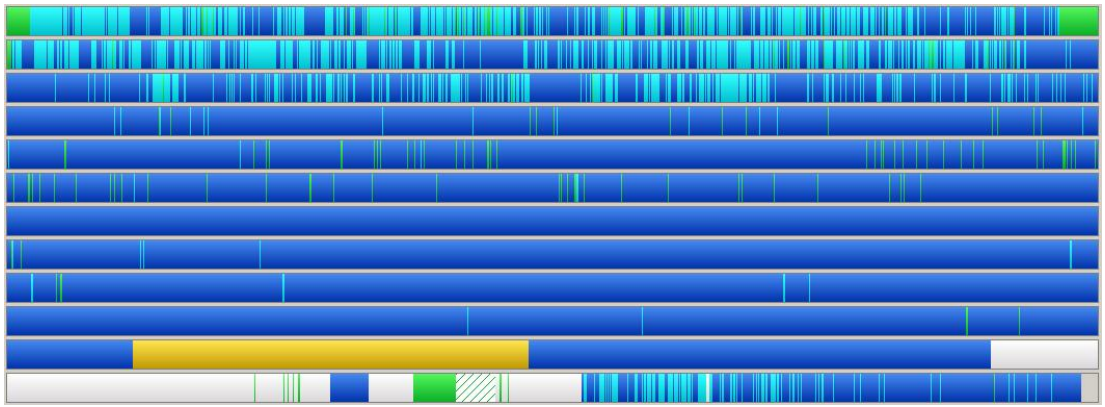
Motivation

Motivation



Blue - defragmented files
Red - fragmented files and folders
Yellow - paging file
Green - system files
White - unused space
Green/White - reserved system space
Light Blue - defragmented folders

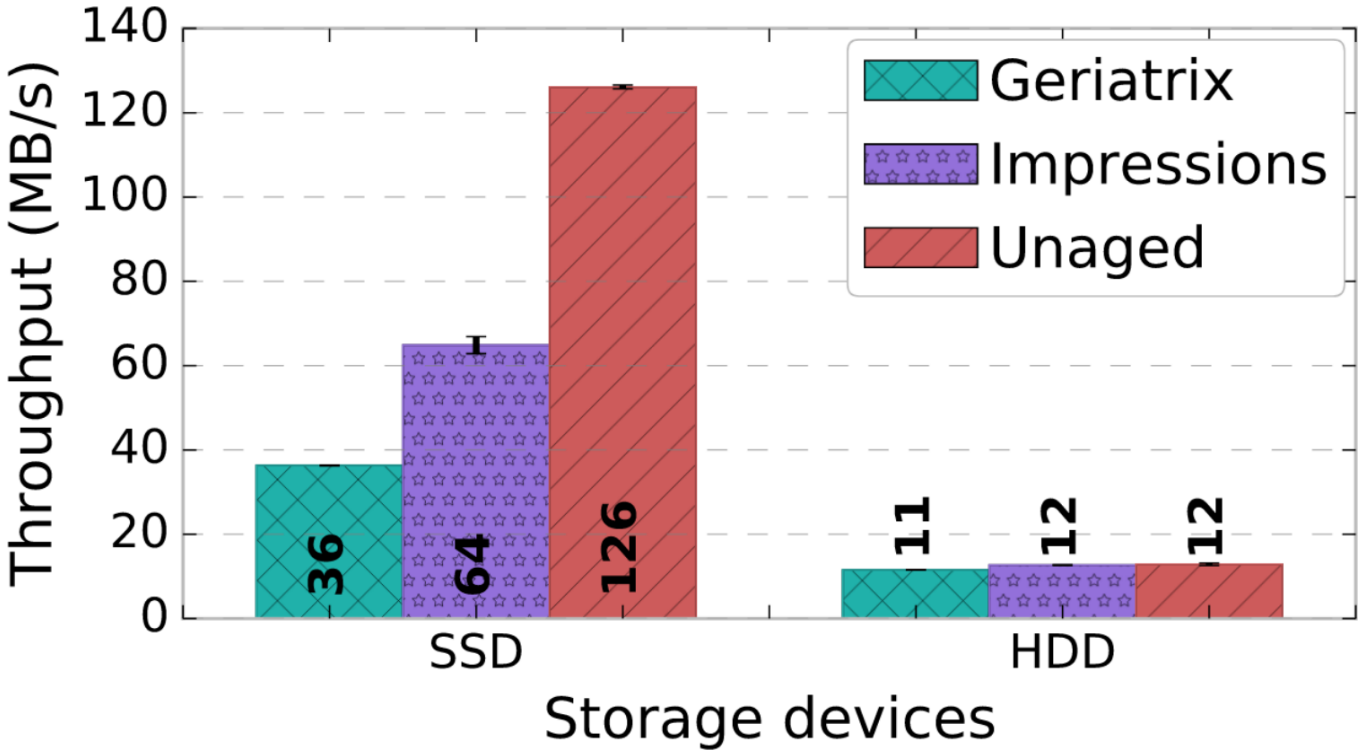
↓ Defragmentation



Blue - defragmented files
Red - fragmented files and folders
Yellow - paging file
Green - system files
White - unused space
Green/White - reserved system space
Light Blue - defragmented folders

Is this still relevant?

Motivation



Aging impact on Ext4 atop SSD and HDD. [8]

Overview

Background

Natural Transfer Size (NTS) [4]

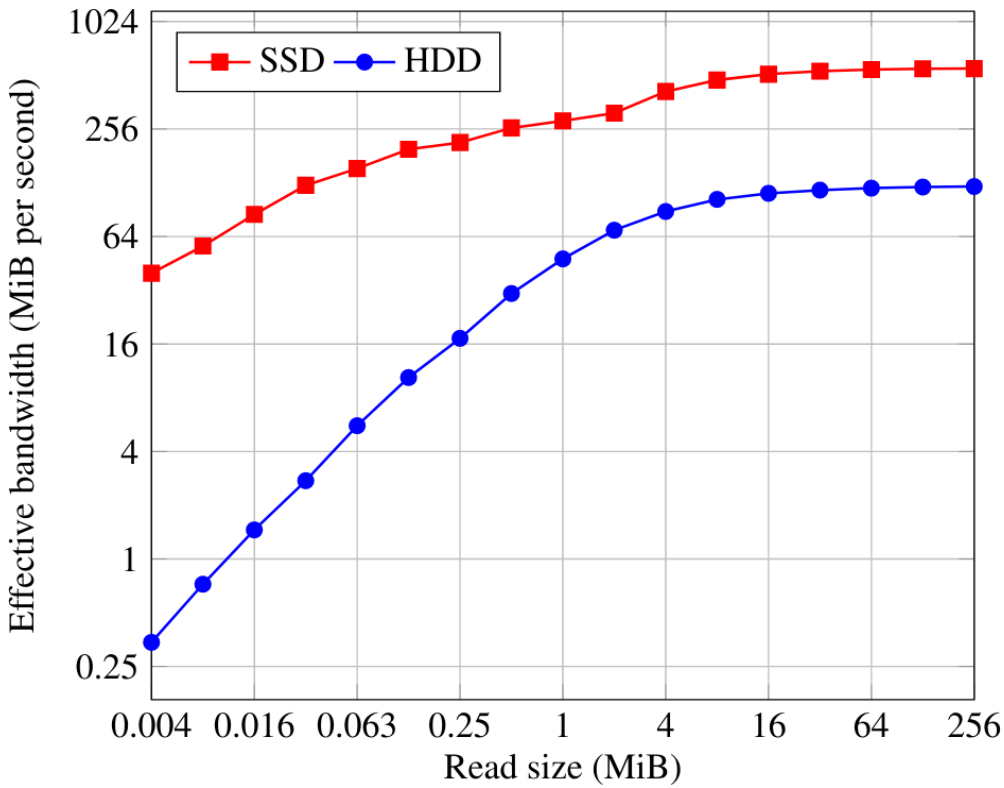
- Maximizing read throughput of a device by reading:
  (a) sufficiently **large** chunks of
  (b) **sequential** data

Natural Transfer Size (NTS) [4]

- Maximizing read throughput of a device by reading:
  (a) sufficiently **large** chunks of
  (b) **sequential** data
- How large is sufficient?

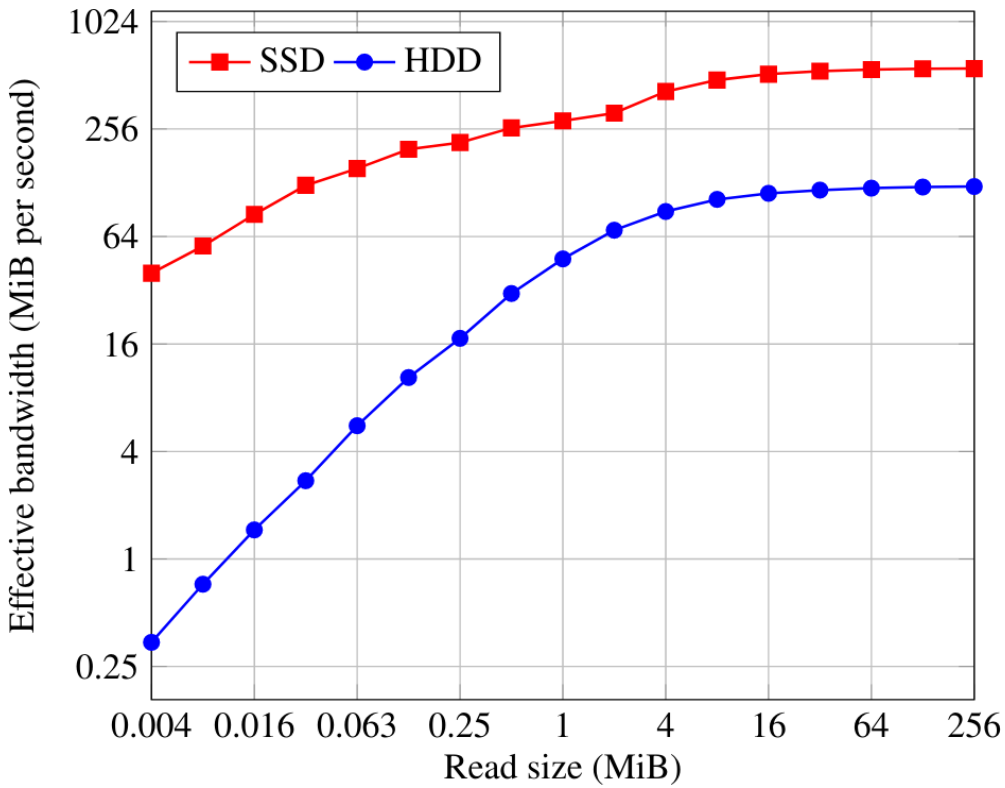
Natural Transfer Size (NTS) [4]

- Maximizing read throughput of a device by reading:
  (a) sufficiently **large** chunks of
  (b) **sequential** data
- How large is sufficient?



Natural Transfer Size (NTS) [4]

- Maximizing read throughput of a device by reading:
  (a) sufficiently **large** chunks of
  (b) **sequential** data
- How large is sufficient?
  - 4MiB $\rightarrow$ Natural Transfer Size (NTS)



Update-In-Place File Systems

Allocation

- Allocation of space for new files
- Modification of files *updates* them *in-place*

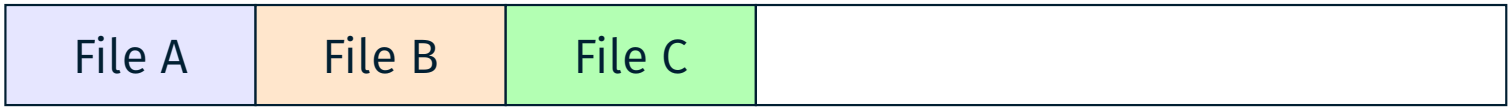
Update-In-Place File Systems

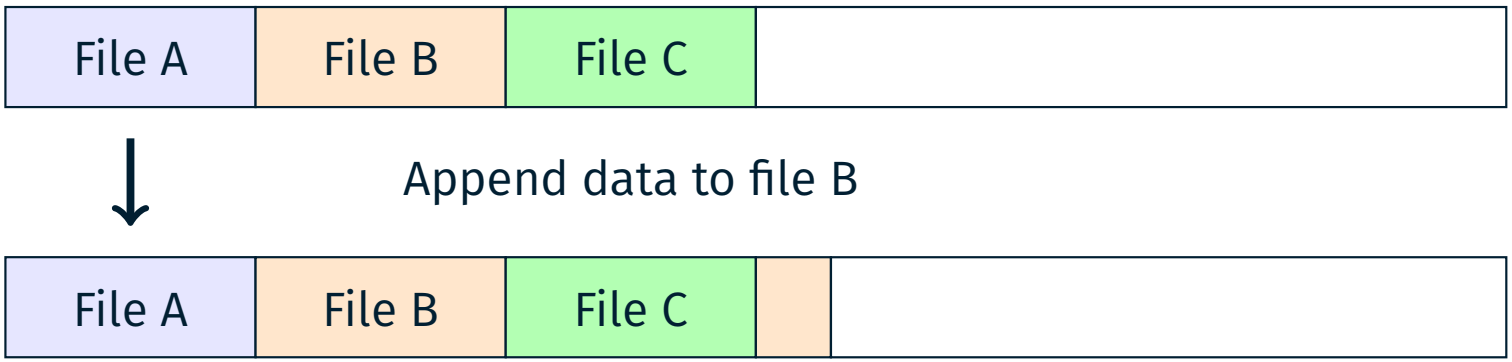
Allocation

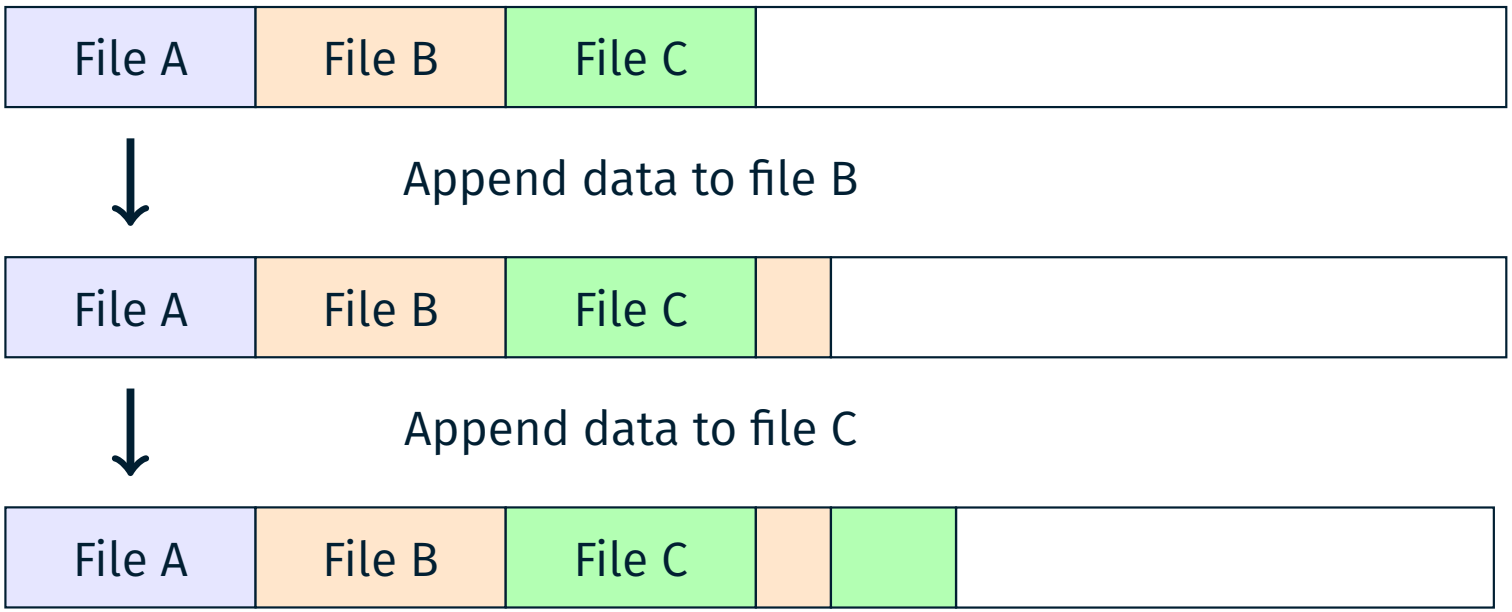
- Allocation of space for new files
- Modification of files *updates* them *in-place*

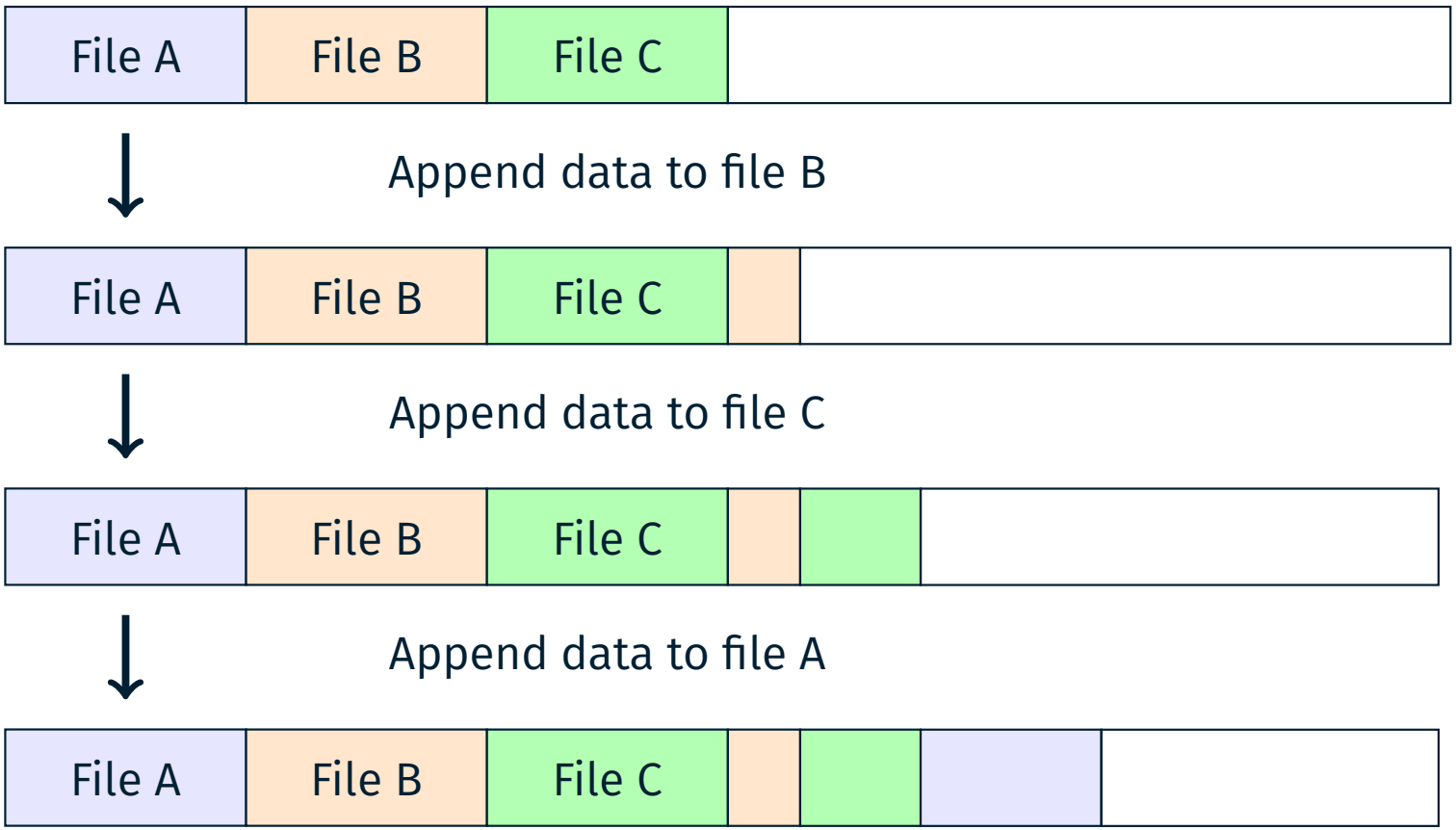
Causes for fragmentation:

- Growth beyond intial allocation
  $\rightarrow$ **intra-file** fragmentation
- Related files are not stored adjacent
  $\rightarrow$ **inter-file** fragmentation


File A
File B
File C


File A
File B
File C
Append data to file B
File A
File B
File C

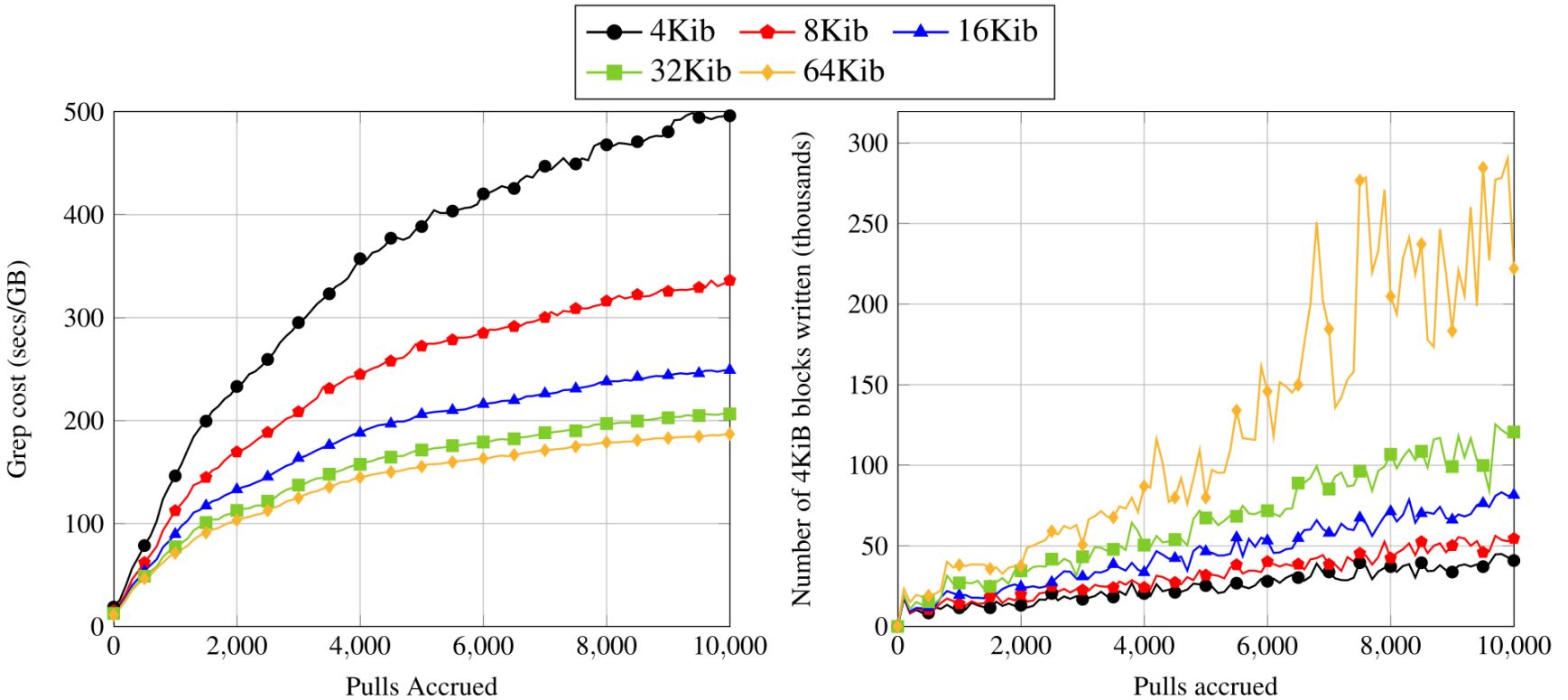

File A
File B
File C
Append data to file B
File A
File B
File C
Append data to file C
File A
File B
File C


File A
File B
File C
Append data to file B
File A
File B
File C
Append data to file C
File A
File B
File C
Append data to file A
File A
File B
File C

B-Tree based File Systems

Organize some parts or the entire file system in B-Trees

- File insertions and deletetions modify the trees structure and cause fragmentation
- Aging depends on the size of the tree nodes
  - NTS sized nodes reduce aging
  - Smaller nodes reduce write amplification



(a) Grep cost at different node sizes (lower is better).

(b) Write amplification at different node sizes (lower is better).

Aging and write amplification on Btrfs. [4]

Measuring age

Conventional file system benchmarks:

- Operation speed [8]
- Search times [4]

Measuring age

Conventional file system benchmarks:

- Operation speed [8]
- Search times [4]

Measuring fragmentation:

- Layout Score [9]
- Dynamic Layout Score [4]
- Free Space Extents [8]
- Write latency [5]

Actual use:

- Accurate representation
- Long term use required (months or years)

Application based aging:

- Aging specific to one workload
- Cannot replicate all effects of real long term aging [8]

Artificial Aging

Generates an artificial workload from file system snapshots

+ Can closely match months of real aging in hours

- Fails to replicate fragmentation for long aging durations
- Regular snapshots are required

Generates a realistic file system image

+ Can be configured to match characteristics and distributions of aged file systems
- No realistic free space fragmentation [8]

Geriatrix [8]

Ages a file system until it matches an aging profile

+ Creates realistic free space fragmentation
+ Can approximate a realistic aged file system in hours

- Requires aging profiles specific to the file system [8]

Known Countermeasures

Grouping Files

Reduces inter-file fragmentation by

- Storing logically related files physically close
- Reserving space for additional files to be allocated

Which files are related?

Grouping Files

Reduces inter-file fragmentation by

- Storing logically related files physically close
- Reserving space for additional files to be allocated

Which files are related?

- Same directory
- Created simultaneously

Grouping Files

Reduces inter-file fragmentation by

- Storing logically related files physically close
- Reserving space for additional files to be allocated

Which files are related?

- Same directory
- Created simultaneously

Limitations

- Directories can grow to exceed the reserved space
- Files can be falsely classified as related

Allocating Extents

Extent: Group of sequential blocks

Reduce intra-file fragmentation by

- Allowing files to grow inside the extent
- Ensuring a minimum size for file fragments

What is a good size for extents?

Allocating Extents

Extent: Group of sequential blocks

Reduce intra-file fragmentation by

- Allowing files to grow inside the extent
- Ensuring a minimum size for file fragments

What is a good size for extents?

- Extent size $>=$ NTS
- More than 90% of files are 1MiB or smaller [3, 6]

Allocating Extents

Extent: Group of sequential blocks

Reduce intra-file fragmentation by

- Allowing files to grow inside the extent
- Ensuring a minimum size for file fragments

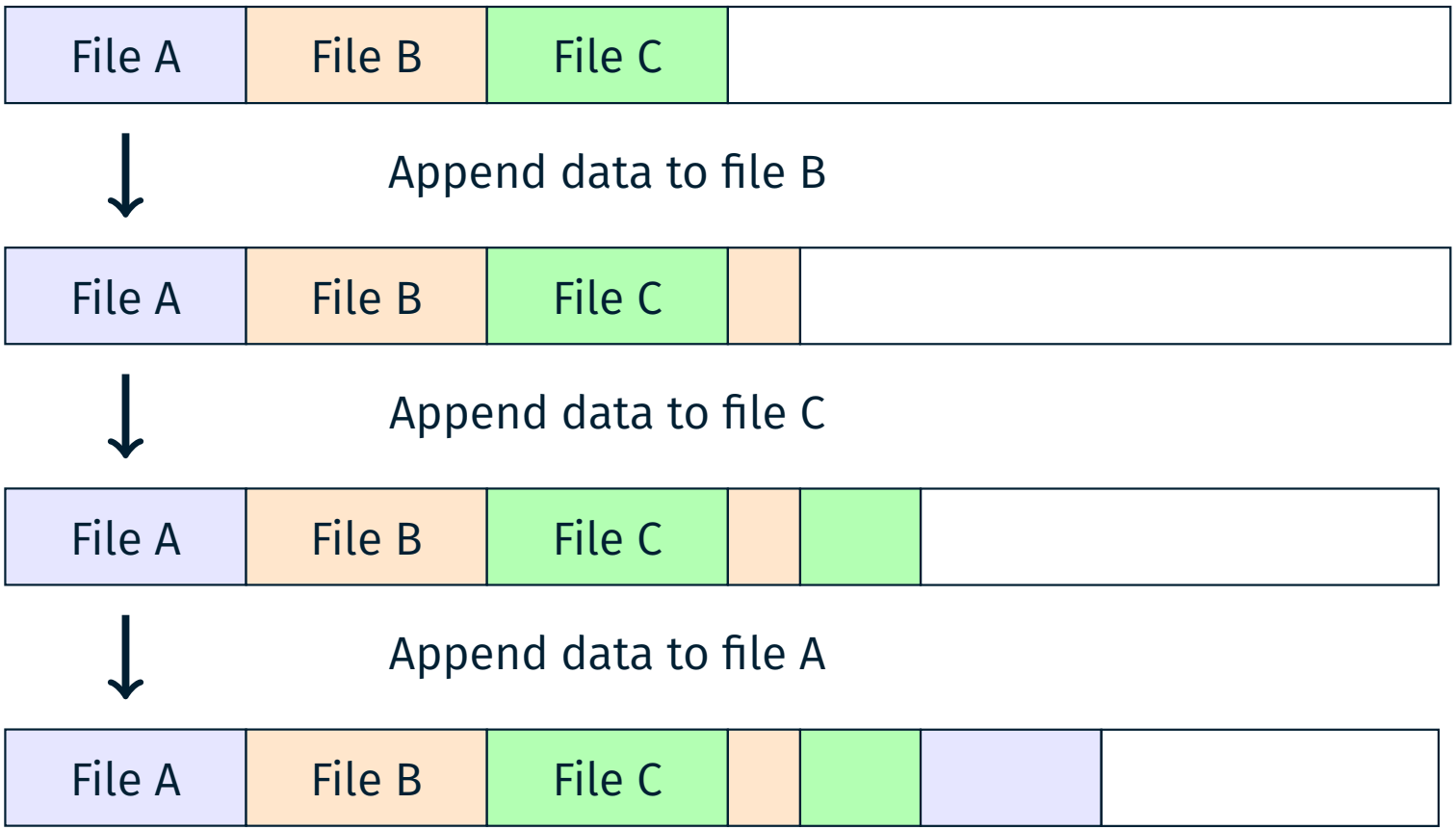
What is a good size for extents?

- Extent size $>=$ NTS
- More than 90% of files are 1MiB or smaller [3, 6]

Limitations

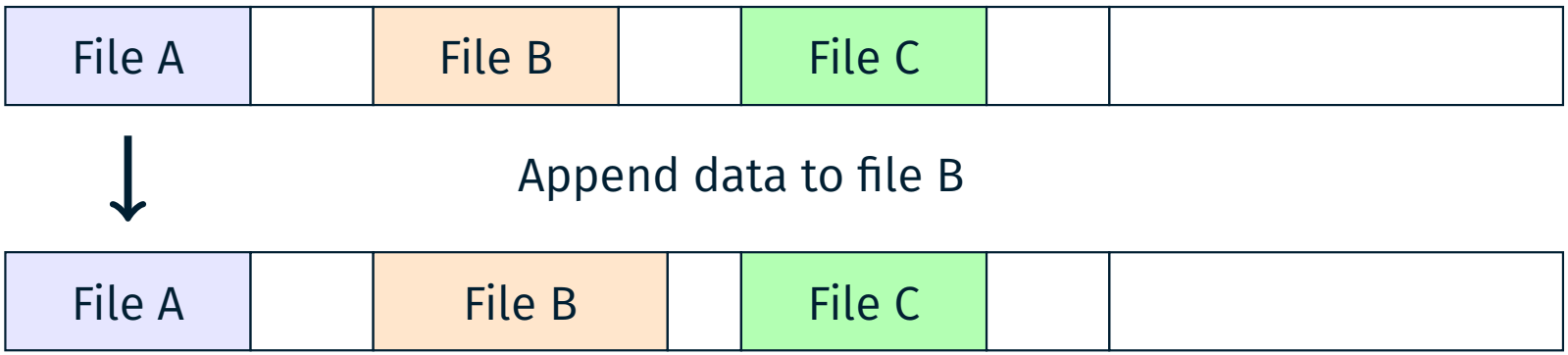
- Files growing beyond the extent size fragment

Reminder: Update-In-Place Fragmentation

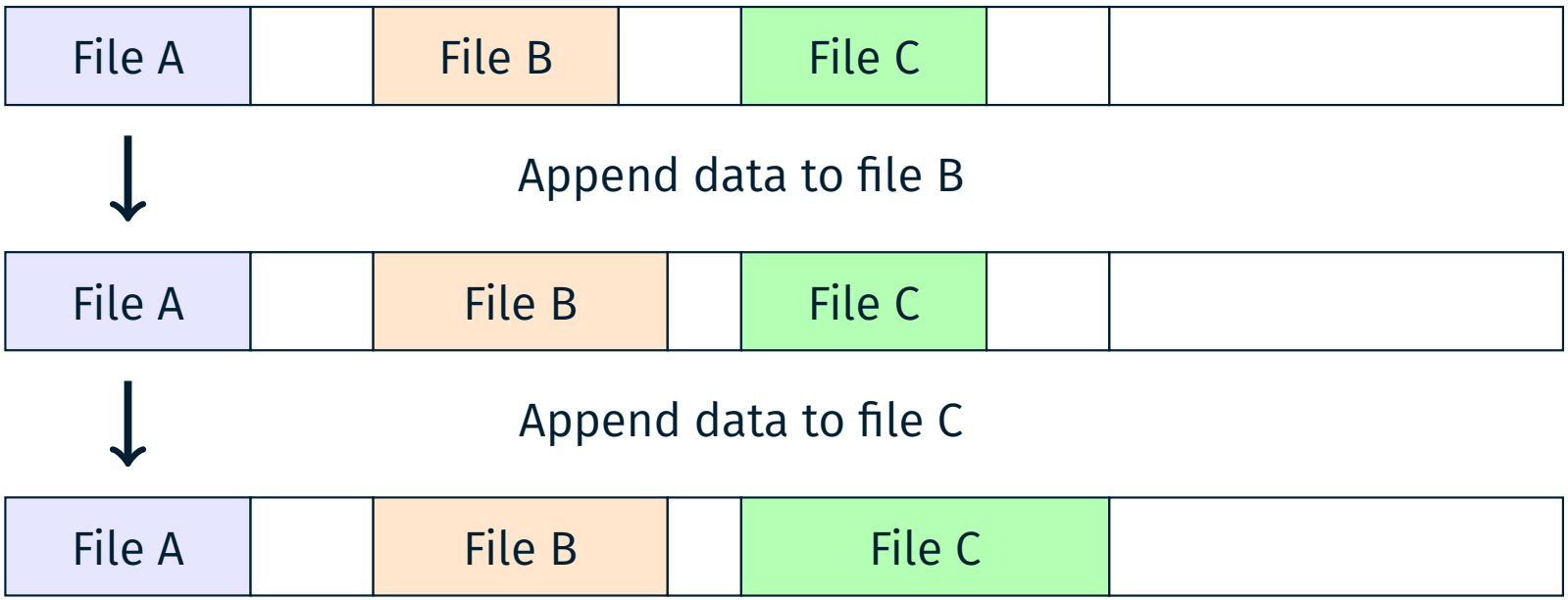


Example: Allocating Extents

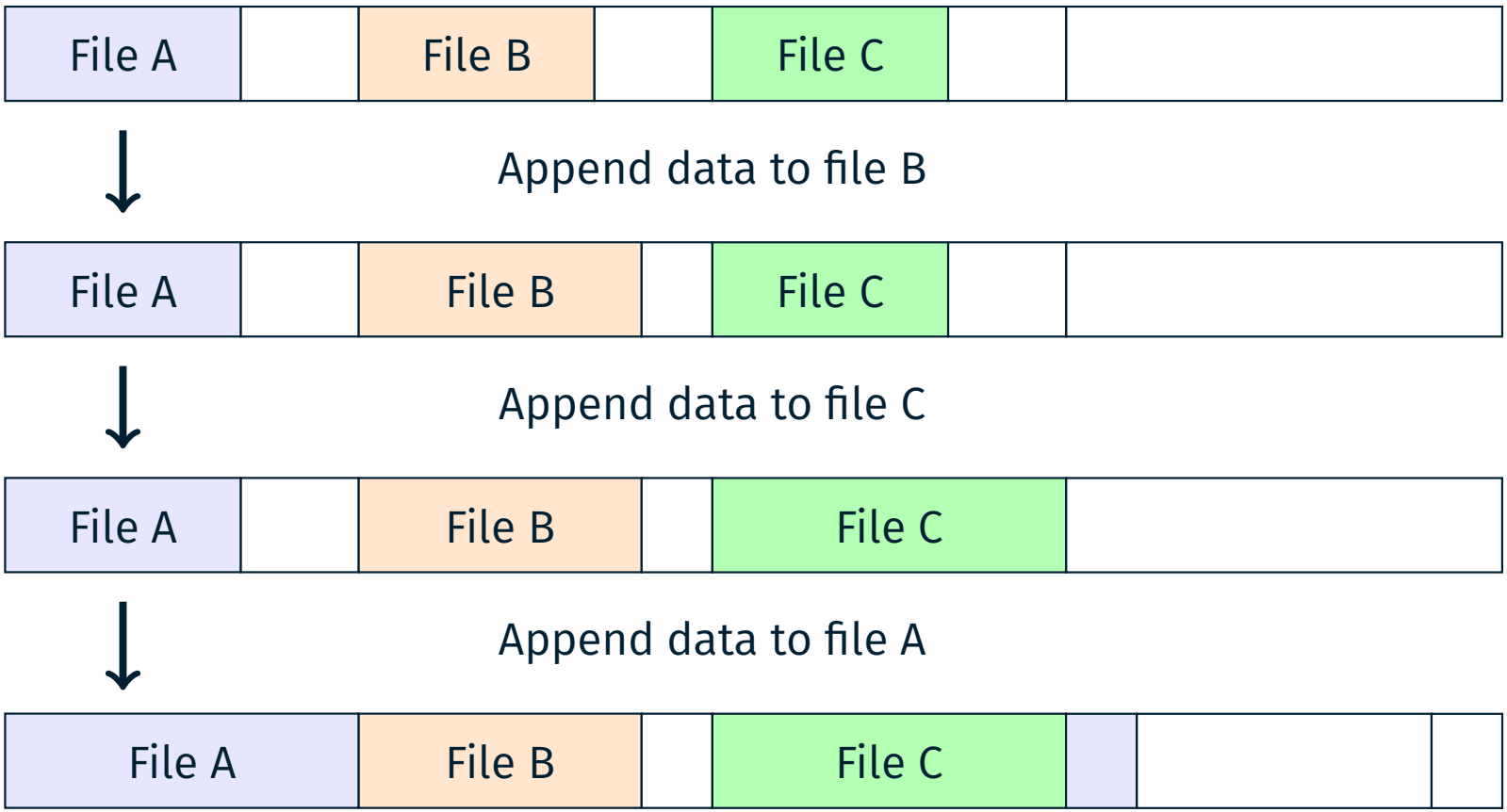
Example: Allocating Extents



Example: Allocating Extents



Example: Allocating Extents



Delayed Allocation

Reduces intra-file fragmentation by

- Delaying allocation after file creation and
- Writing data to a buffer initially and
- Choosing an appropriate extent after the file has reached final size

Delayed Allocation

Reduces intra-file fragmentation by

- Delaying allocation after file creation and
- Writing data to a buffer initially and
- Choosing an appropriate extent after the file has reached final size

Limitations:

- Delay can only be a few seconds [4]
- Files growing over a long period of time will still fragment

Packing

Reduces inter-file fragmentation by

- Packing multiple small files into one extent
- Storing metadata in the same extent
- Storing small files directly in the metadata structures

Packing

Reduces inter-file fragmentation by

- Packing multiple small files into one extent
- Storing metadata in the same extent
- Storing small files directly in the metadata structures

Limitations:

- Can lead to heavy fragmentation on file growth

Modern File Systems and Aging

Ext4 [1]

Type: Update-In-Place

Countermeasures:

- Grouping
- Packing
- Extents (up to 128MiB)
- Delayed allocation

BtrFS [4]

Type: B-Tree based

Node size: 4KiB (default) to 64KiB (maximum)

Countermeasures:

- Extents
- Delayed allocation
- Packing

BetrFS [7]

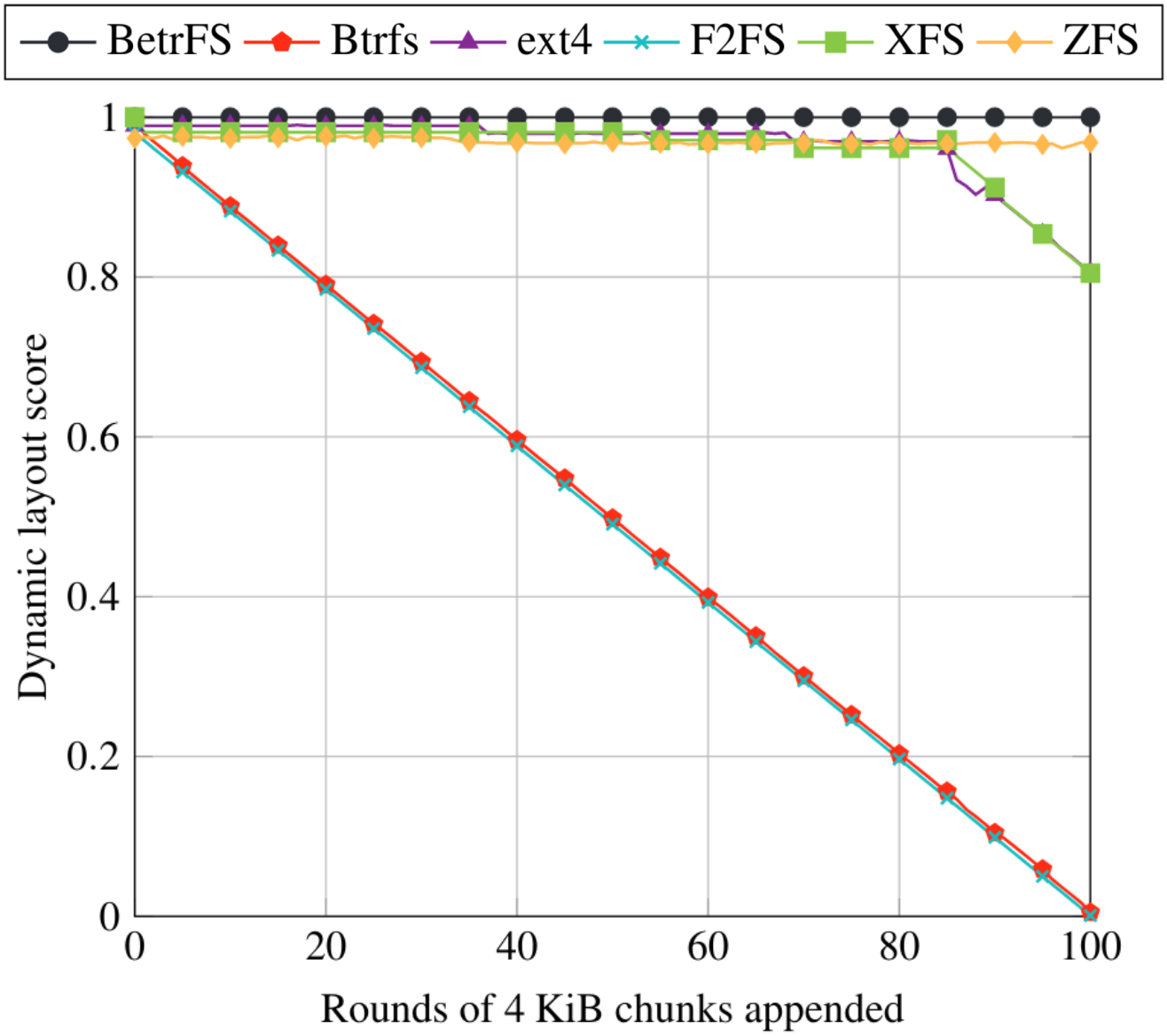
Type: B^{ε}-Tree based

Node size: 4MiB

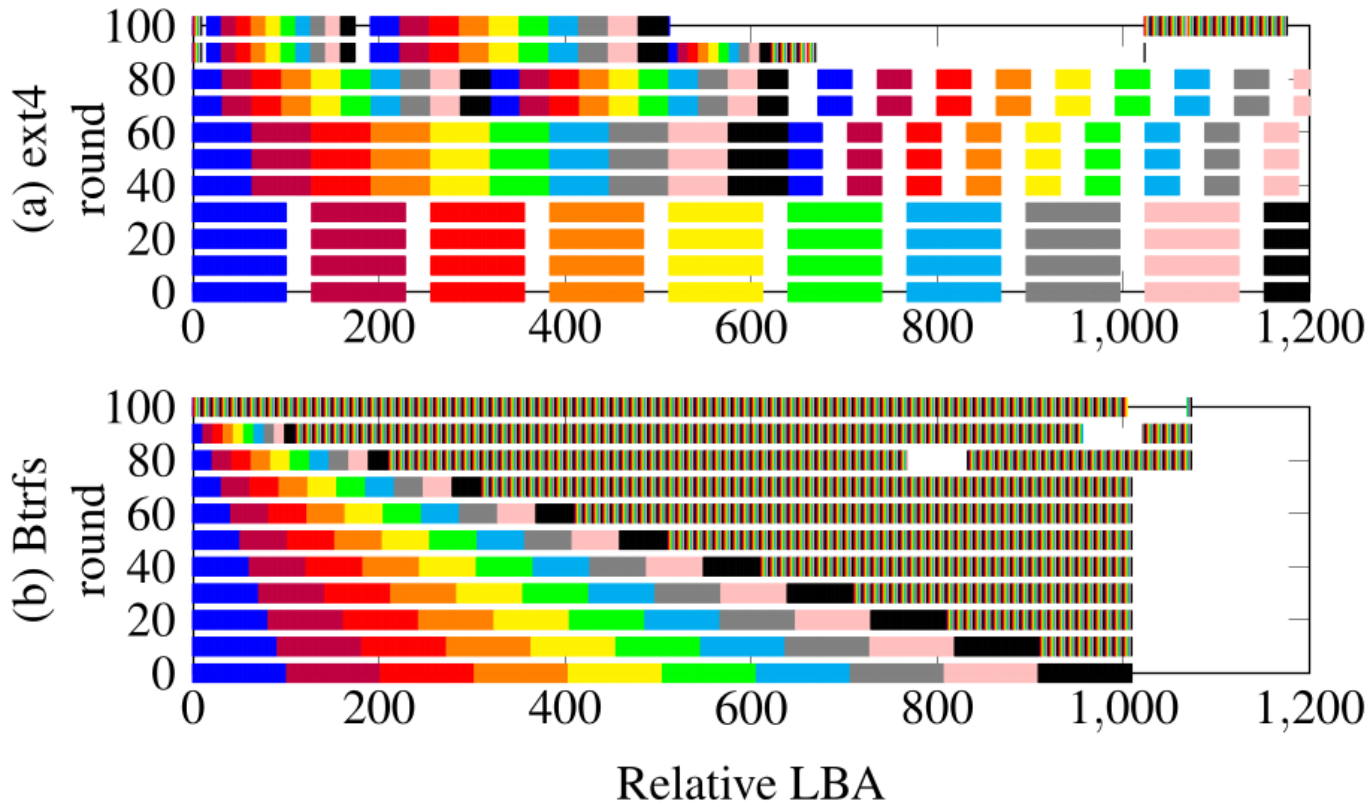
Countermeasures:

- B^{ε}-Tree
  - Modifications to nodes are buffered
  - Reduced write amplification
- Data is also stored in a B^{ε}-Tree

Comparison: Intra-File Fragmentation [4]

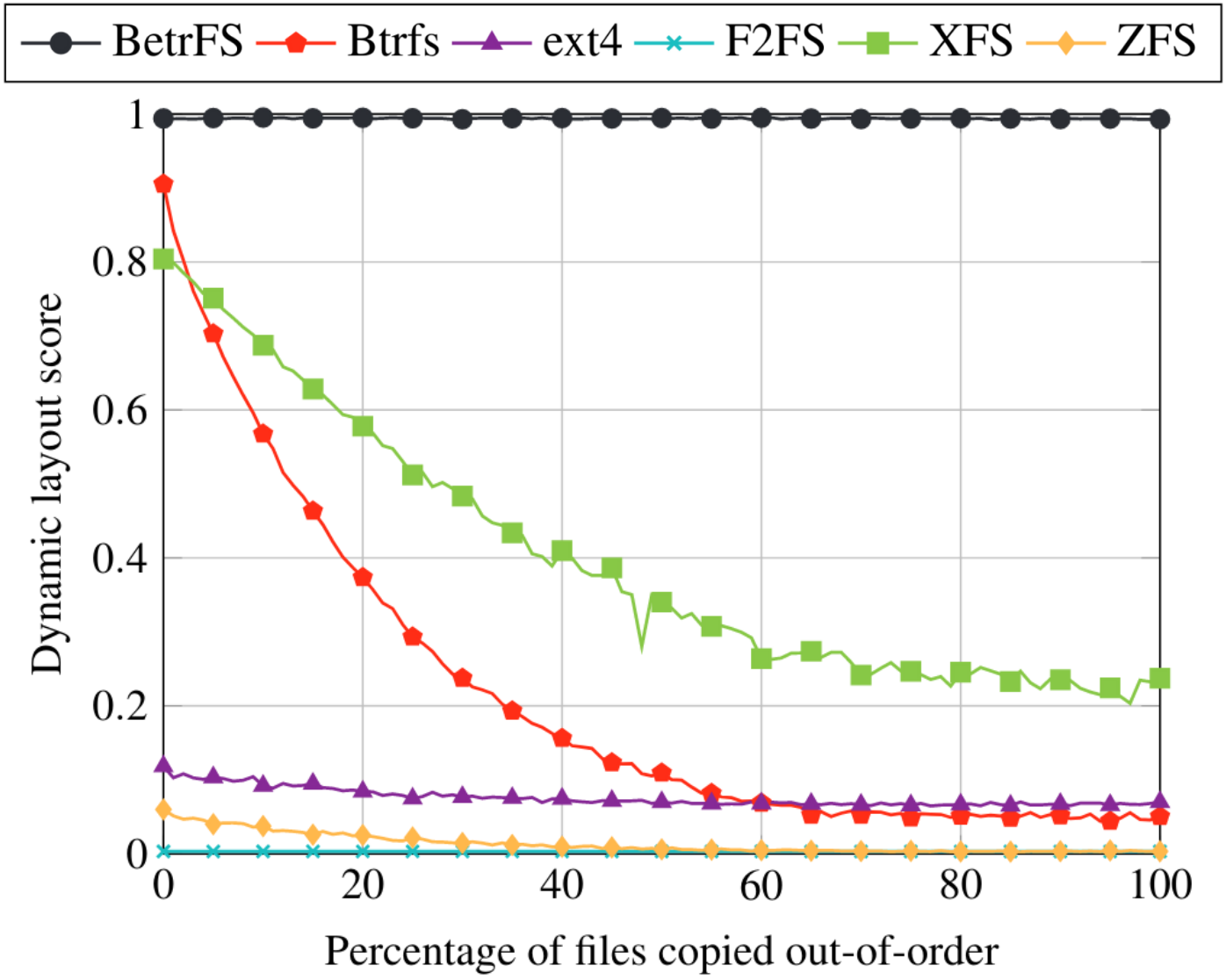


Comparison: Intra-File Fragmentation



Intrafile benchmark layout visualization. [4]

Comparison: Inter-File Fragmentation [4]



Conclusion

Conclusion

File System Aging is not a solved problem:

- Ext4 and BtrFS age measurably
- Known countermeasures help but have limitations

Current and future research:

- Aging on non-mechanical drives
- New Filesystems

Questions?

References (1)

[1] The linux kernel documentation: ext4 data structures and algorithms.

`https://www.kernel.org/doc/html/latest/filesystems/ext4/index.html`.

Accessed: 10.01.2020.

[2] Nitin Agrawal, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau.

Generating realistic impressions for file-system benchmarking.

ACM Transactions on Storage (TOS), 5(4):16, 2009.

[3] Nitin Agrawal, William J Bolosky, John R Douceur, and Jacob R Lorch.

A five-year study of file-system metadata.

ACM Transactions on Storage (TOS), 3(3):9, 2007.

References (2)

[4] Alex Conway, Ainesh Bakshi, Yizheng Jiao, William Jannen, Yang Zhan, Jun Yuan, Michael A. Bender, Rob Johnson, Bradley C. Kuszmaul, Donald E. Porter, and Martin Farach-Colton.

File systems fated for senescence? nonsense, says science!

In *15th USENIX Conference on File and Storage Technologies (FAST 17)*, pages 45–58, Santa Clara, CA, February 2017. USENIX Association.

[5] Alex Conway, Eric Knorr, Yizheng Jiao, Michael A. Bender, William Jannen, Rob Johnson, Donald Porter, and Martin Farach-Colton.

Filesystem aging: It's more usage than fullness.

In *11th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 19)*, Renton, WA, July 2019. USENIX Association.

References (3)

[6] John JD Douceur and Bill Bolosky.

A large-scale study of file-system contents.

1999.

[7] William Jannen, Jun Yuan, Yang Zhan, Amogh Akshintala, John Esmet, Yizheng Jiao, Ankur Mittal, Prashant Pandey, Phaneendra Reddy, Leif Walsh, et al.

Betrfs: A right-optimized write-optimized file system.

In *13th {USENIX} Conference on File and Storage Technologies ({FAST} 15)*, pages 301–315, 2015.

[8] Saurabh Kadekodi, Vaishnavh Nagarajan, and Gregory R Ganger.

Geriatrix: Aging what you see and what you don't see. a file system aging approach for modern storage systems.

In *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, pages 691–704, 2018.

References (4)

[9] Keith Smith and Margo Seltzer.

File system aging - increasing the relevance of file system benchmarks.

ACM SIGMETRICS Performance Evaluation Review, 25, 07 1997.