File System Aging

Featuring slides modified from

Martín Farach-Colton
Rutgers University

Aging

- This video focus on ideas from two papers
  - Smith and Seltzer (1997)
  - Conway et al. (2017)

Outline

- I/O Models preview
- Verifying models through simulation and measurement
- Aging Problem
- Types of Fragmentation
- Quantifying and measuring aging

How do we model performance?

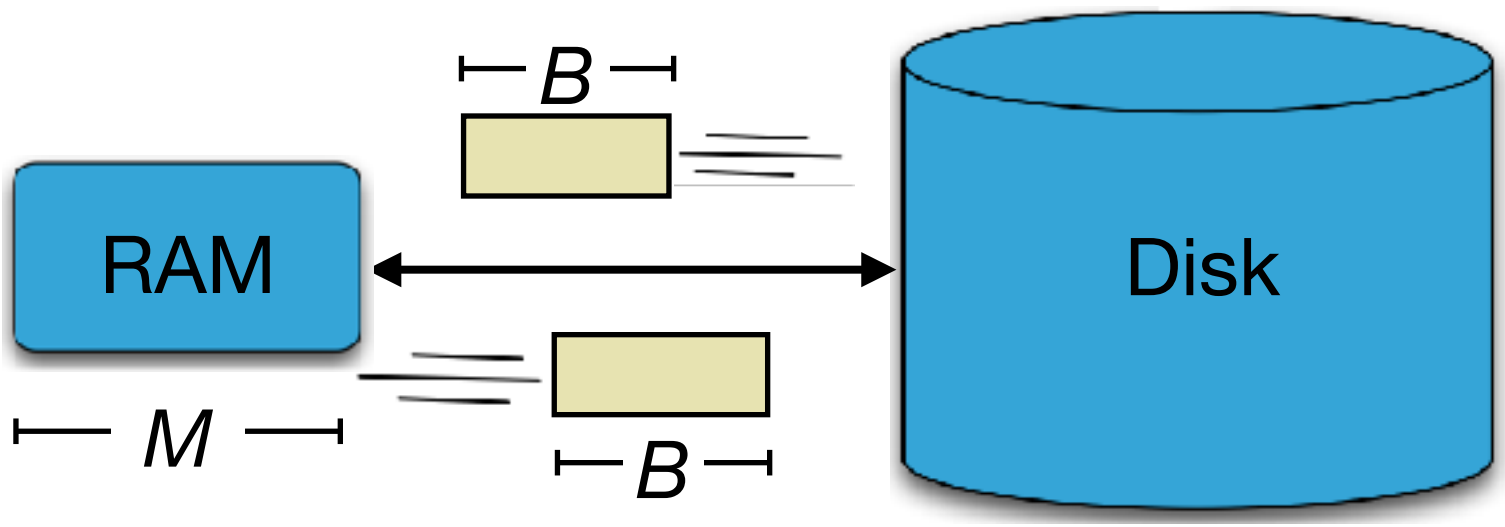
How do we account for disk I/O?

DAM model: One way that theorists think about *external memory* algorithms

- Data is transferred in blocks between RAM and disk.
- The number of block transfers dominates the running time.

Goal: Minimize # of I/Os

- Performance bounds are parameterized by block size ***B***, memory size ***M***, data size ***N***.



[Aggarwal+Vitter '88]

Is the DAM Model any good?

Short answer: Yes (2-competitive)

Long answer: …but not great (can’t tune parameters)

Affine model:

- Data is transferred in blocks between RAM and disk.
- If ***k*** blocks are transferred, the cost is

$$1 + \alpha k$$

- On hard disks, 1 is the normalized seek cost and α is the incremental bandwidth cost of transferring subsequent blocks
- On SSDs, it's more complicated, but the affine model still "fits" better than DAM model.
  - (And PDAM fits even better…)

Takeaway: the affine model captures the size of I/Os as well as the speed of the device itself (α). We'll refer to this

What does the DAM Model Say About Aging?

The DAM Model Says That Locality Matters:

- Transferring up to ***B*** contiguous bytes is free
  - Each seek incurs an additional DAM I/O
  - Ideal world (real): we read entire objects with a single seek
  - Ideal world (DAM): we always read **B** bytes per seek

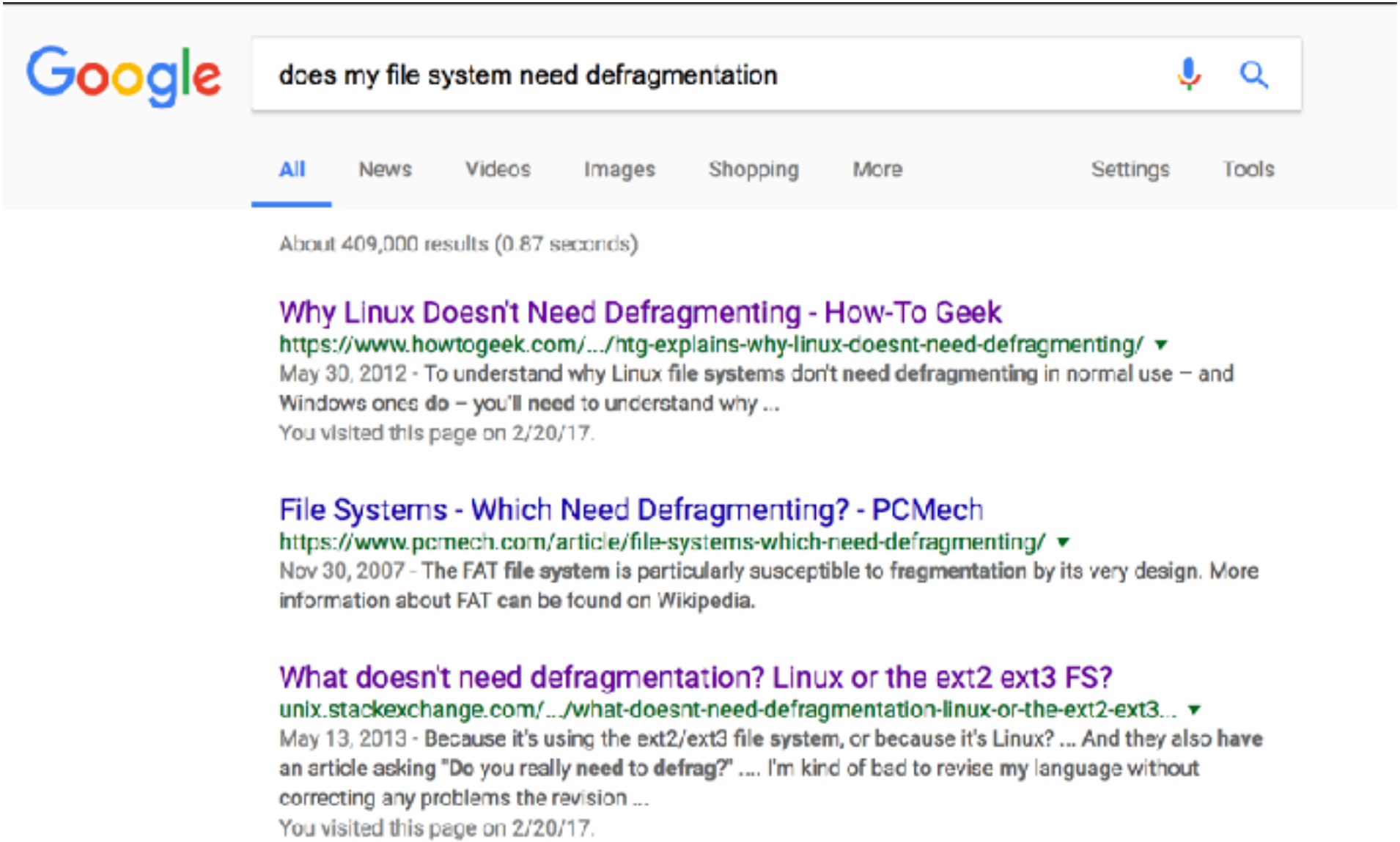
- What causes seeks in our file system?
  - Fragmentation!

Aging is the Accumulation of Fragmentation Over Time

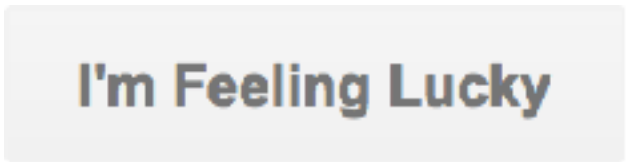
- Aging manifests as degrades performance
- Unless we intervene, “age” is typically monotonically increasing

Does aging happen in file systems?

Do file system age?



Do file system age?



Chris Hoffman at howtogeek.com says:

"Linux's ext2, ext3, and ext4 file systems… [are] designed to avoid fragmentation in normal use."

"If you do have problems with fragmentation on Linux, you probably need a larger hard disk."

Do file system age?

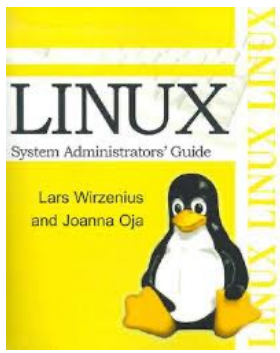
I'm Feeling Lucky

Chris Hoffman at howtogeek.com says:

"Linux's ext2, ext3, and ext4 file systems… [are] designed to avoid fragmentation in normal use."

"If you do have problems with fragmentation on Linux, you probably need a larger hard disk."

"Modern Linux filesystems keep fragmentation at a minimum…Therefore it is not necessary to worry about fragmentation in a Linux system."



So it’s all pointless, right?

Do file system age?

So: as of 1997, file systems aged.

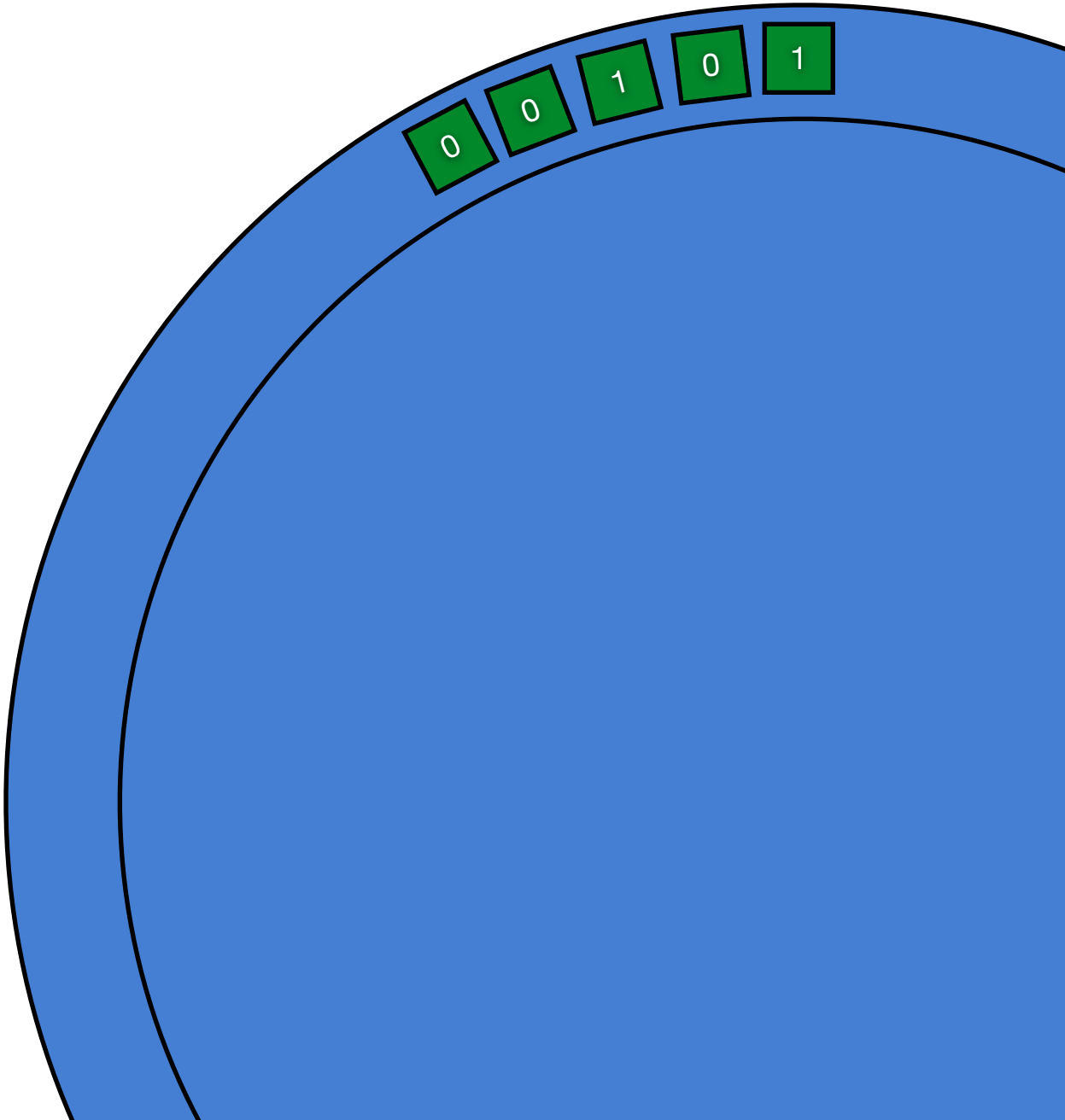
Then file systems got better, and sys admins say they don't age.

What's the actual story?

Theory of Aging over the Ages

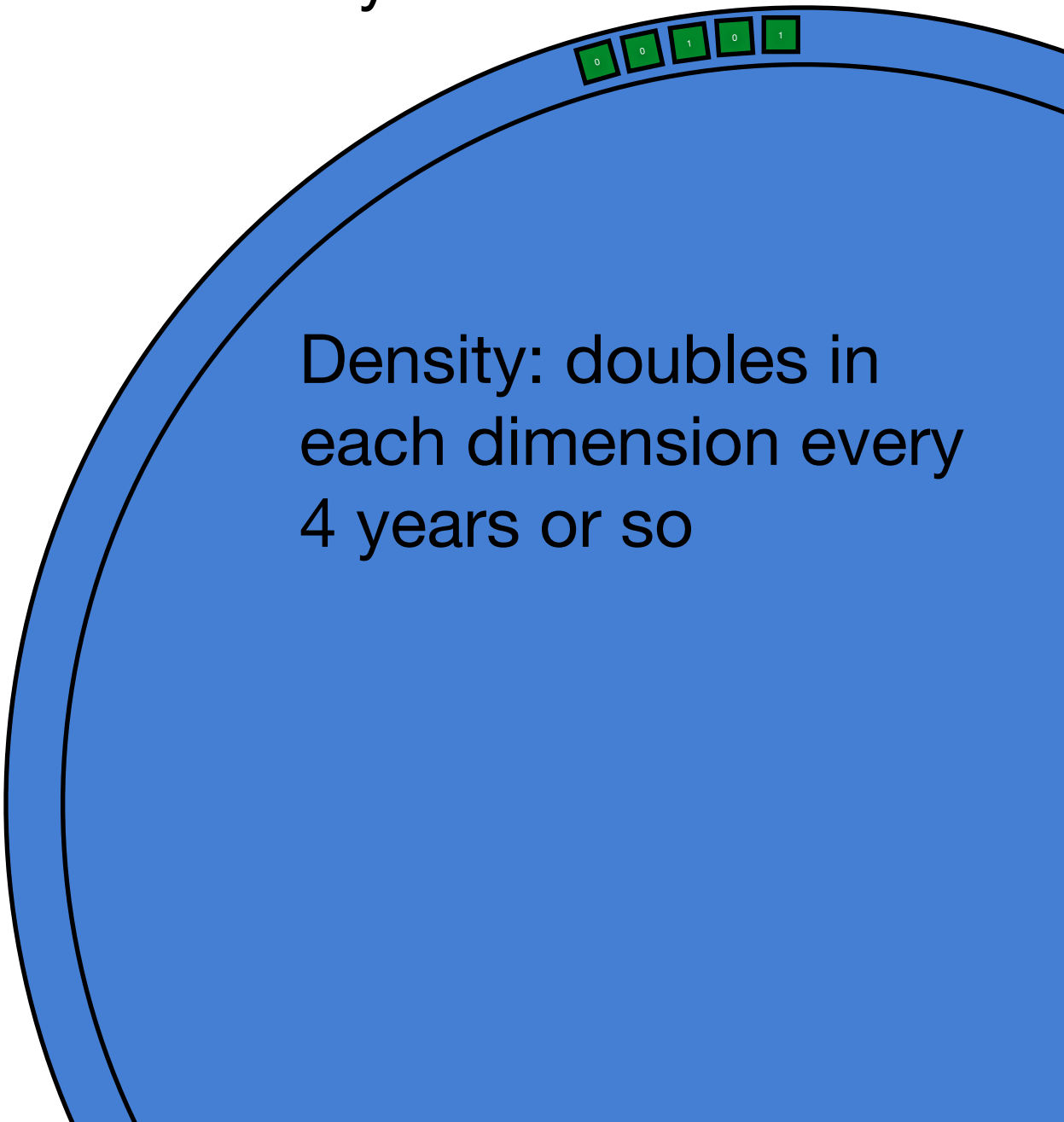
Euclid's view of hard disks

Year: X



Euclid's view of hard disks

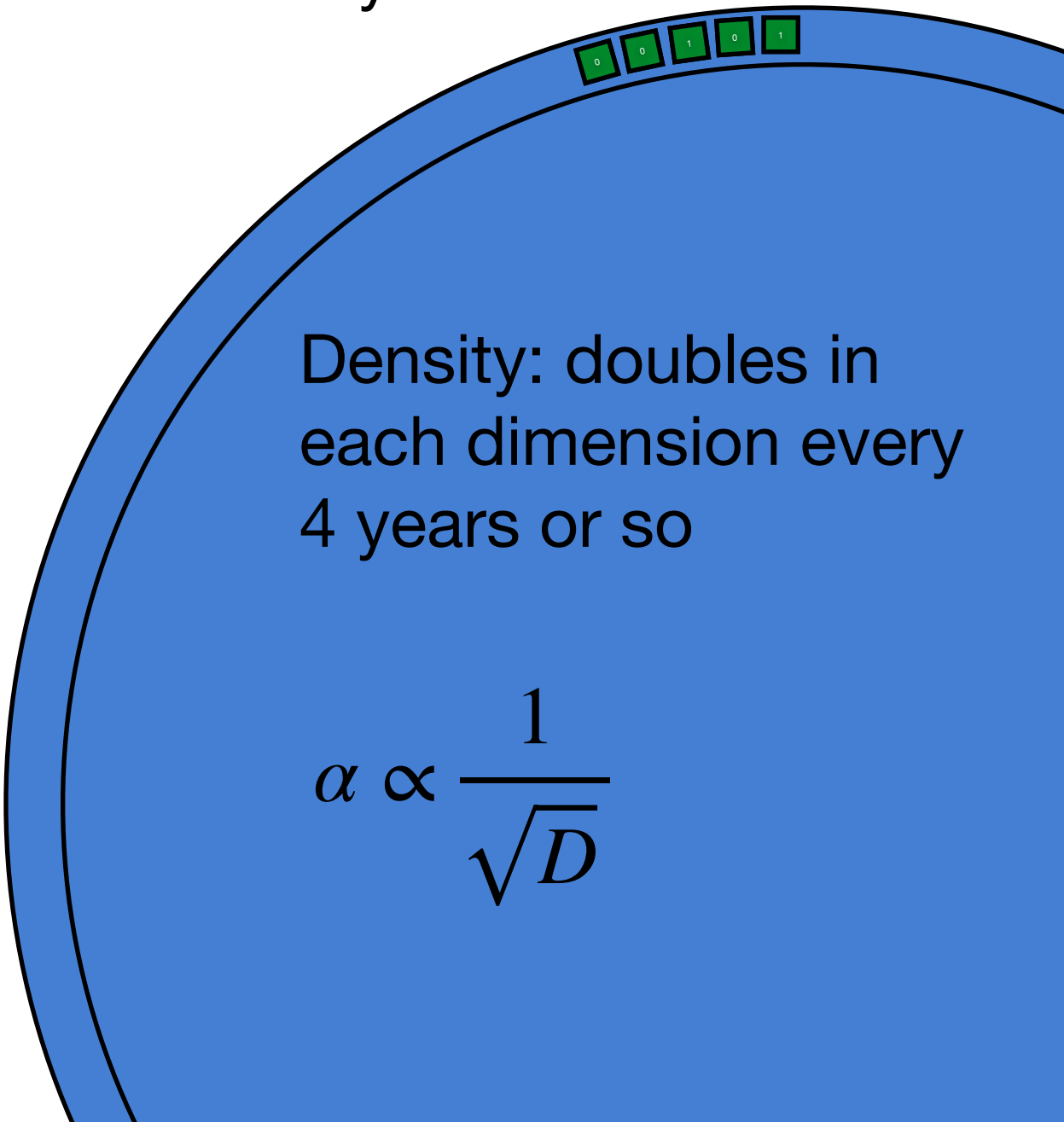
Year: X+~4 years



Density: doubles in each dimension every 4 years or so

Euclid's view of hard disks

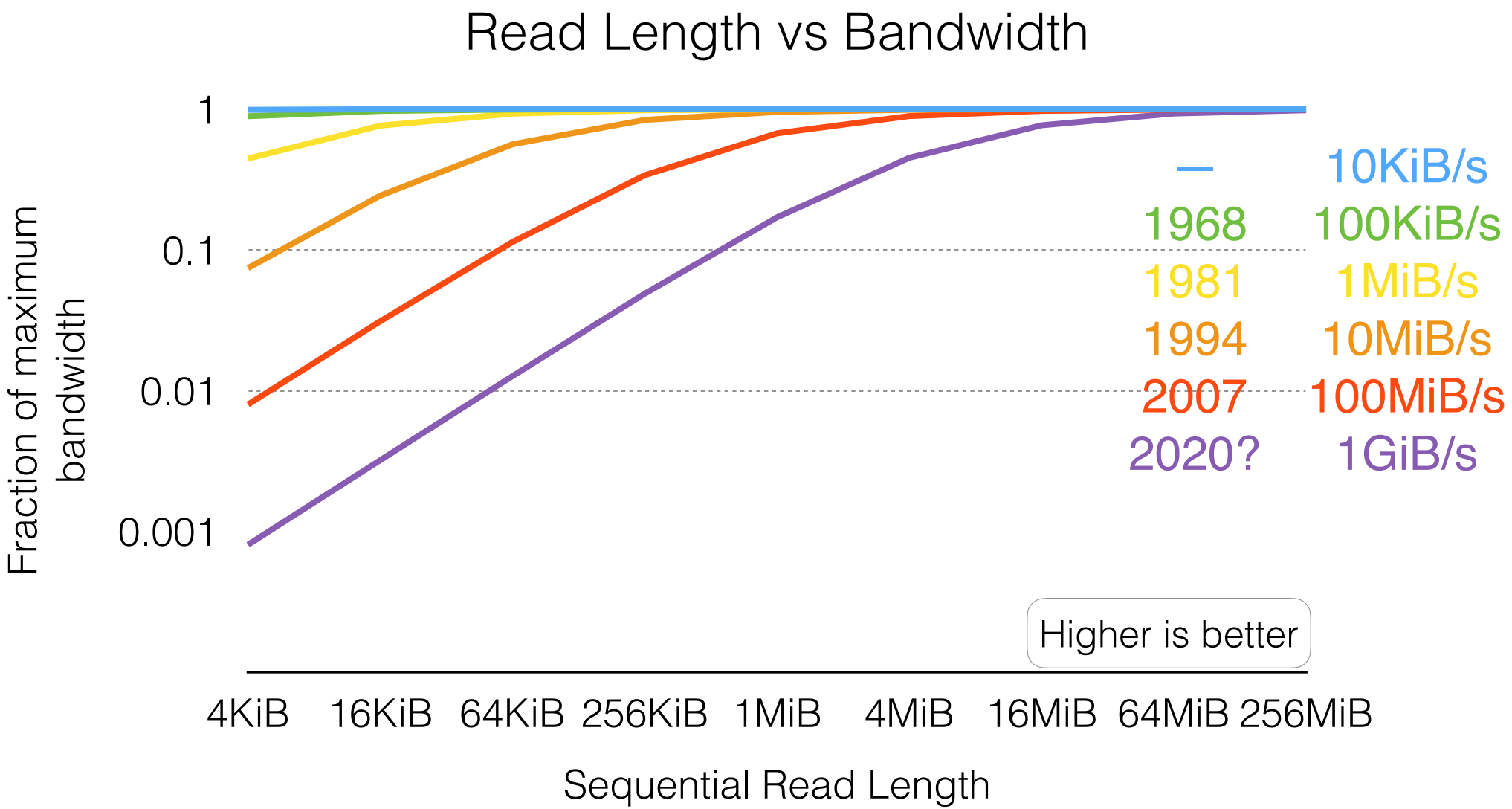
Year: X+~4 years



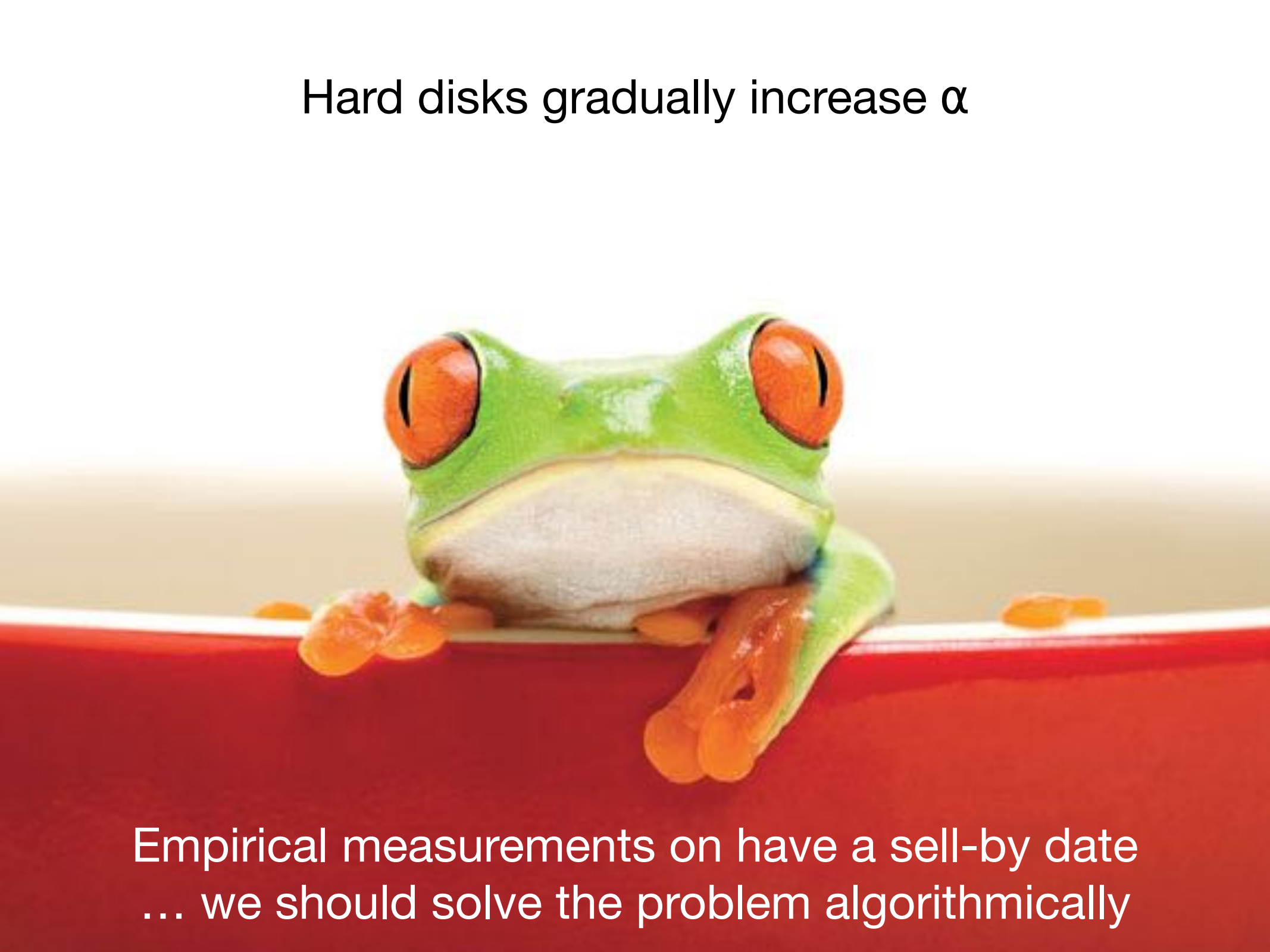
Density: doubles in each dimension every 4 years or so

$$\alpha \propto \frac{1}{\sqrt{D}}$$

Read length/bandwidth over time



Hard disks gradually increase α



Empirical measurements on have a sell-by date
… we should solve the problem algorithmically

Assumption

- Random seek is 100x slower than sequential
- 1% of blocks are non-sequential in the file system

Conclusion

- That's enough to limit IO to 50%

So, for people who think that file systems don't age, are you sure that modern file systems keep fragmentation to under 1%?

Let’s test the hypothesis!
How?

Smith and Seltzer ‘97

Keith Smith started grad school in ’92

- He decided to take snapshots of a bunch of computers
  - Every day
    - For years
- (A snapshot is a fixed state of some disk/FS)

Logistical Challenges of this approach:

- Data collection process is very time consuming
- The cost of storing the snapshots could be huge
- How to figure out set of operations that transition between consecutive snapshot states?
  - Necessary if want to recreate same process on multiple different file systems
  - There are operations that could occur between Snapshot i and Snapshot i+1, but that wouldn’t up in either snapshot (short lived files, sets of updates, etc.)
- Reproducibility?
  - A single workstation/lab may not be representative, and the problems above prohibit using this approach on a global scale

Despite challenges, many important takeaways

He and Seltzer found that:

- If you replay the changes implied by the snapshots
- File system performance degrades significantly

Layout score for measuring a FS age:

- The layout score of a single file is the percentage of blocks in the file that are *optimally allocated*
  - No gap between a block and the previous block of the same file.
  - File’s first block is ignored because it has no “previous block”

Layout score limitations?

- Layout score only captures data blocks of a single file
  - What about metadata accessed along with the file (e.g., inode)
  - What about related files?
    - Files in the same directory are likely read together
  - What about free space?
    - Fragmentation of free space could affect allocation of *future* files
- Layout score is a static measure
  - Does not necessarily reflect the access patterns of a real system

Question: How else could we measure fragmentation?

How else could we measure fragmentation?

```
time grep -r random_string /path/to/fs
```

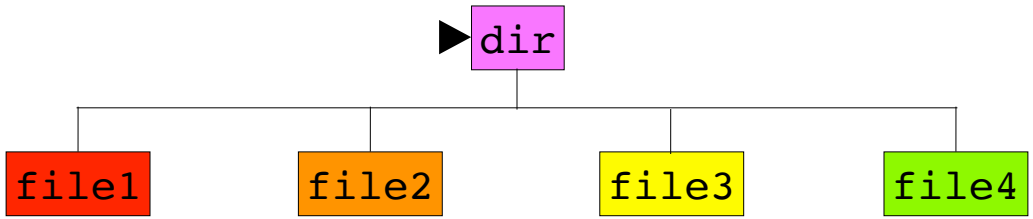
Like timing a preorder traversal of FS tree…

Should measure variety of fragmentation types

- Why?

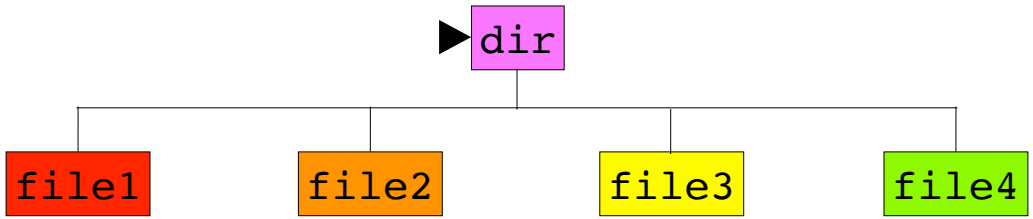
Measuring Aging

```
time grep -r random_string /path/to/filesystem
```



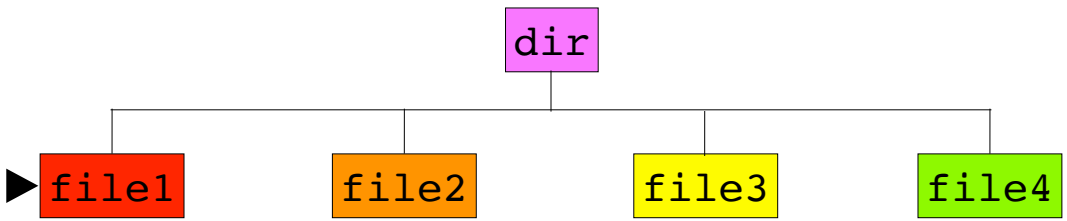
Measuring Aging

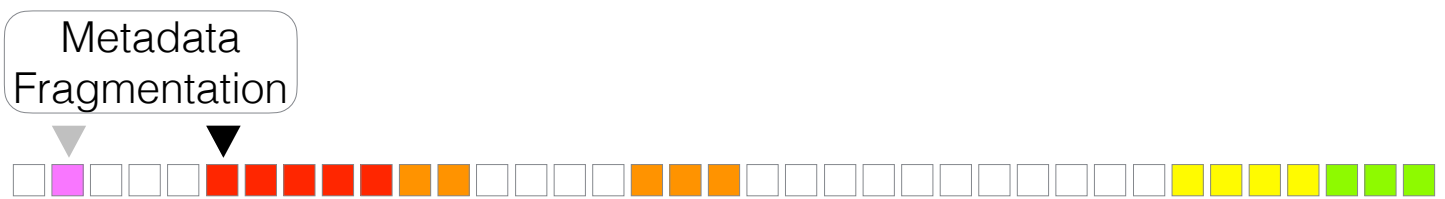
```
time grep -r random_string /path/to/filesystem
```



Measuring Aging

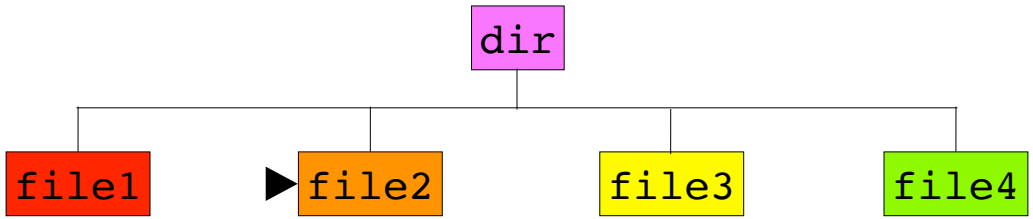
```
time grep -r random_string /path/to/filesystem
```

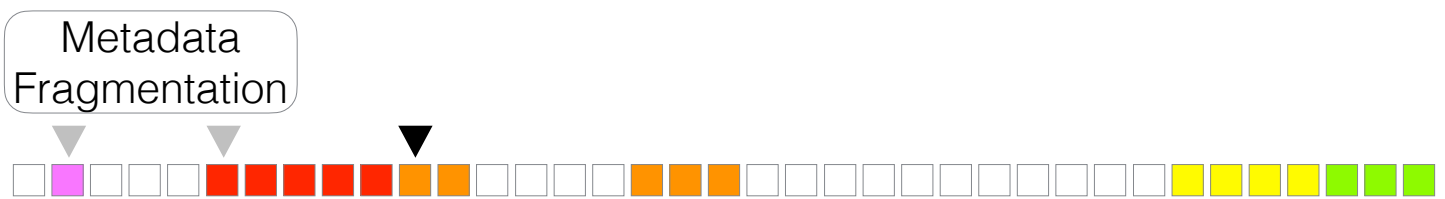




Measuring Aging

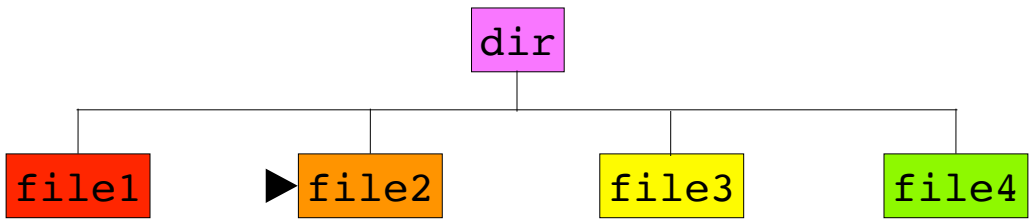
```
time grep -r random_string /path/to/filesystem
```

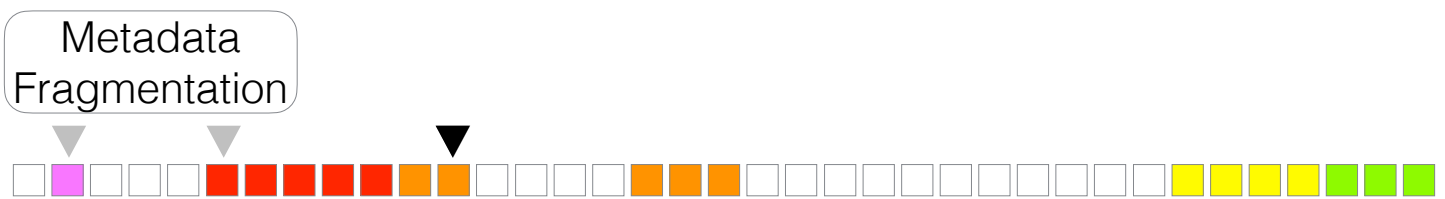




Measuring Aging

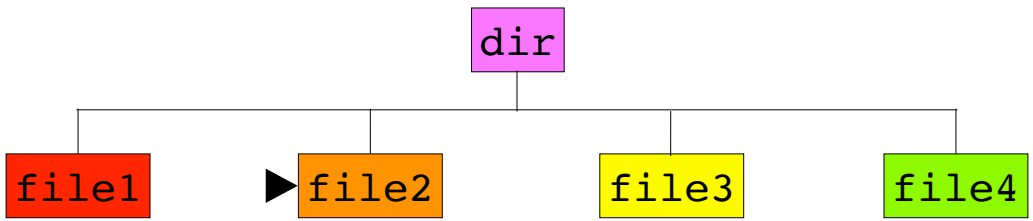
```
time grep -r random_string /path/to/filesystem
```

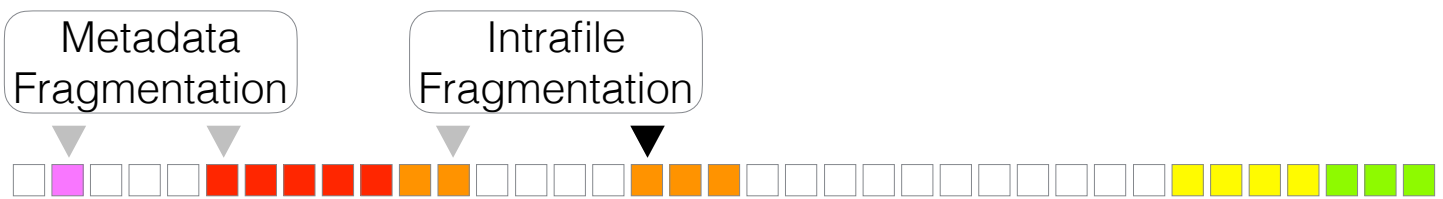




Measuring Aging

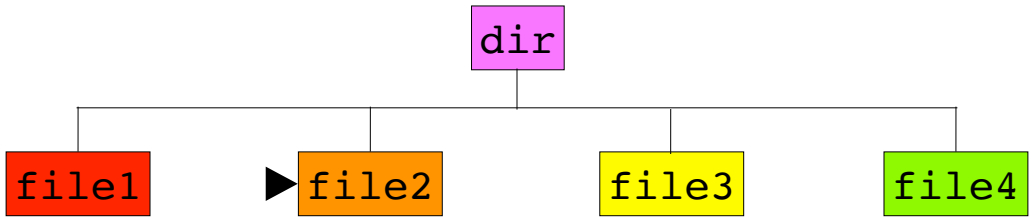
```
time grep -r random_string /path/to/filesystem
```

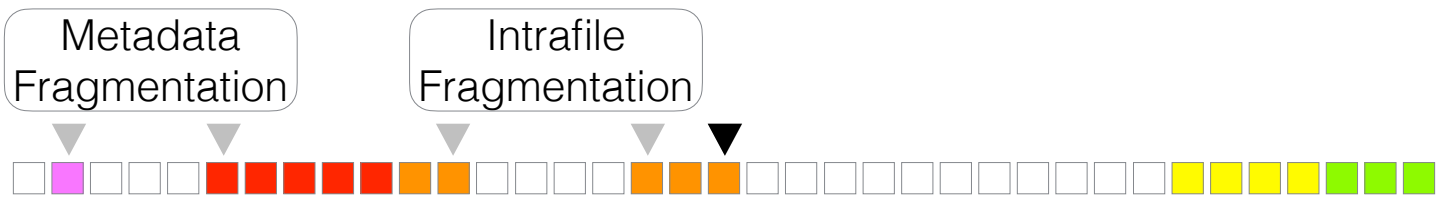




Measuring Aging

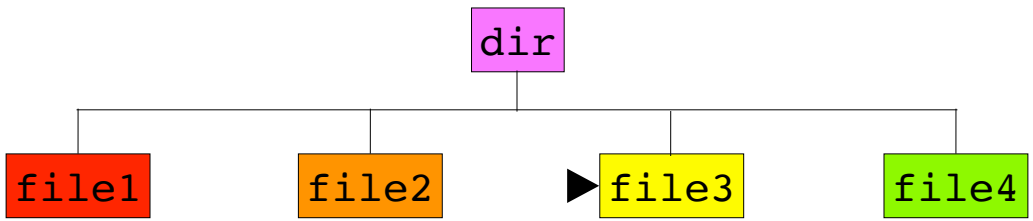
```
time grep -r random_string /path/to/filesystem
```

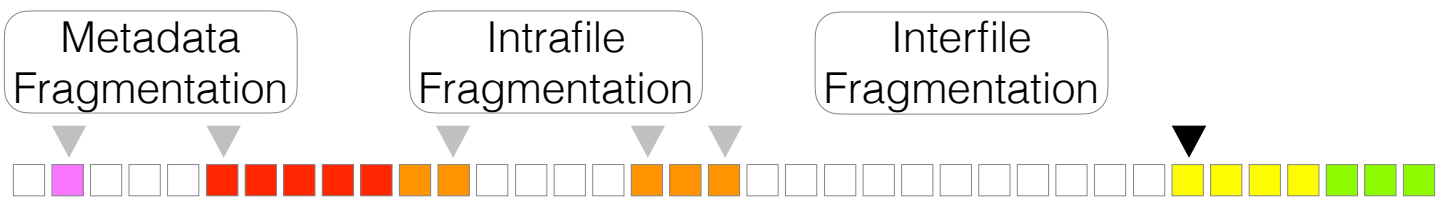




Measuring Aging

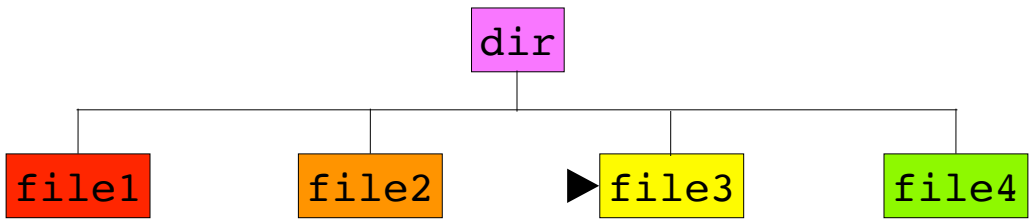
```
time grep -r random_string /path/to/filesystem
```

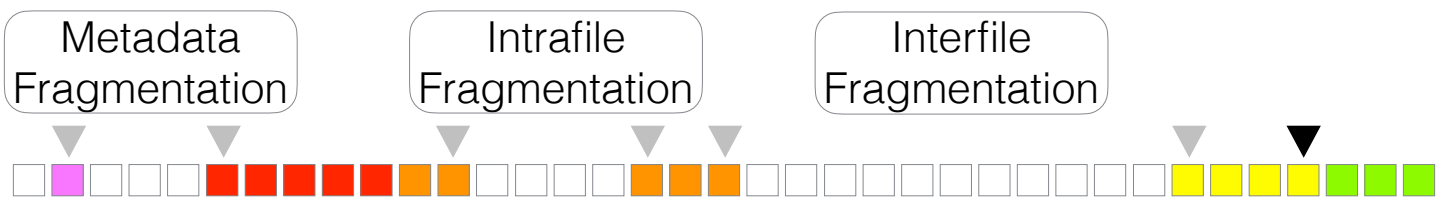




Measuring Aging

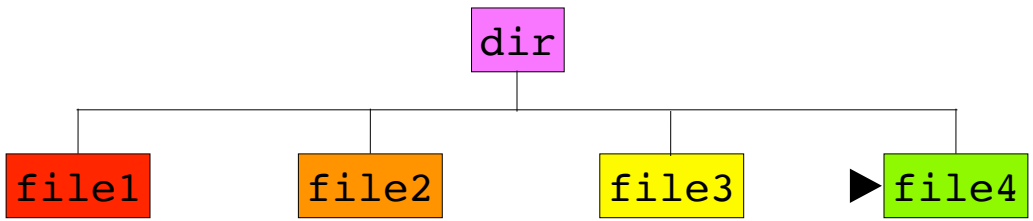
```
time grep -r random_string /path/to/filesystem
```

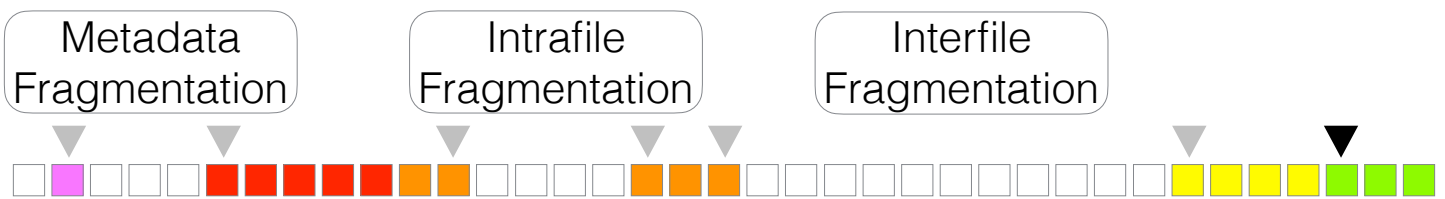




Measuring Aging

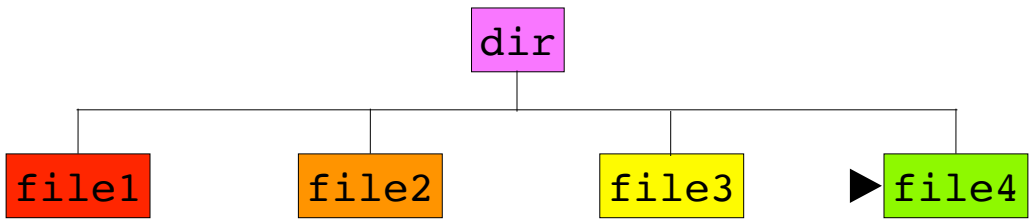
```
time grep -r random_string /path/to/filesystem
```

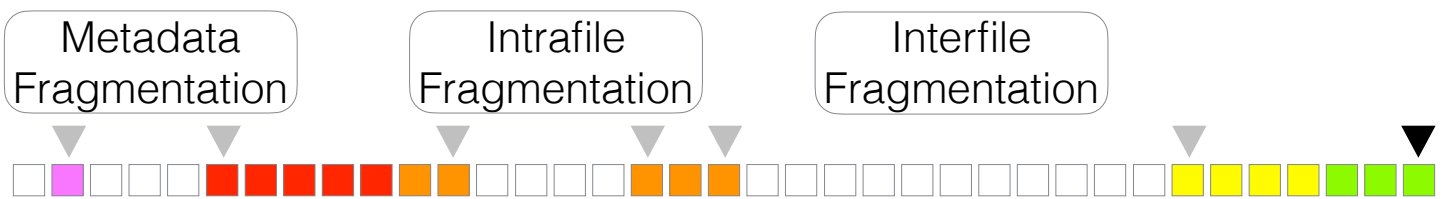




Measuring Aging

```
time grep -r random_string /path/to/filesystem
```





Then normalize per gigabyte read

Used normalized cost of a recursive grep as a *dynamic* age measurement

Used normalized cost of a recursive grep as a *dynamic* age measurement

Captures three categories of fragmentation:

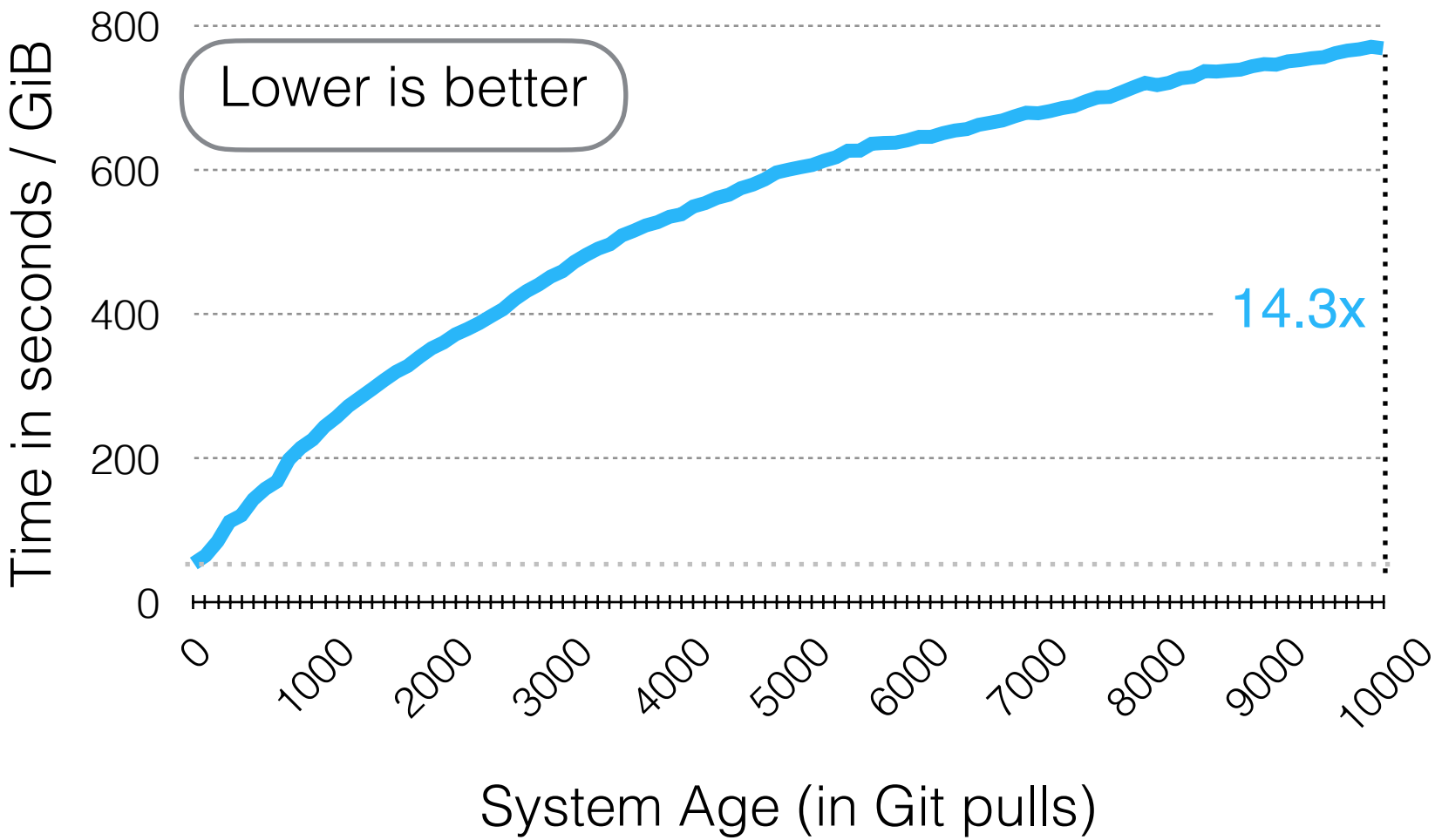
- Intrafile fragmentation is fragmentation involving blocks from the same file.
- Interfile fragmentation is fragmentation involving blocks from two different files.
- Metadata fragmentation is fragmentation involving at least one metadata block.

Limitations of recursive grep:

- Only captures costs of reads
  - Does not capture free space fragmentation, which affects write aging

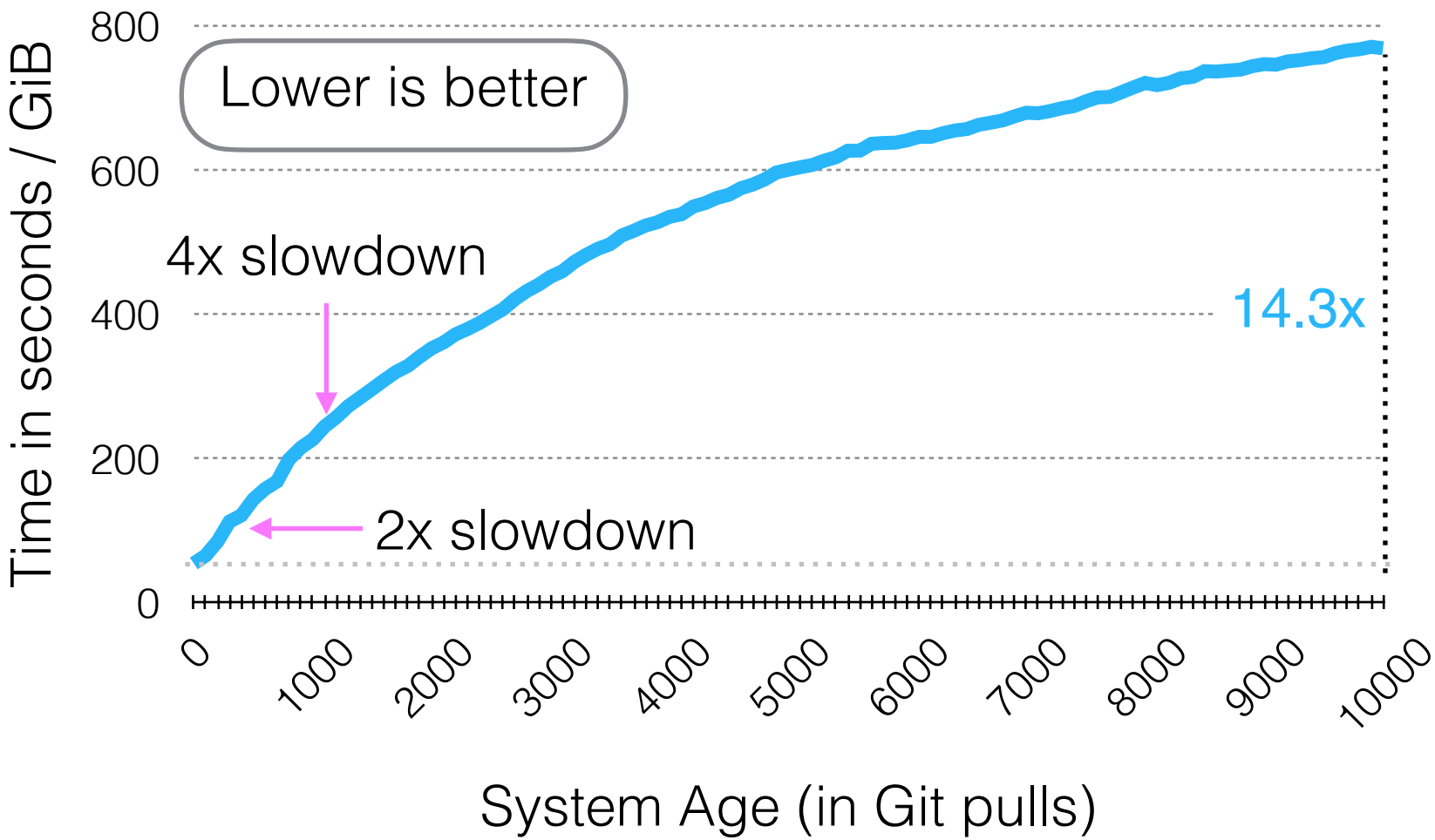
Do modern file systems really age?

Git workload on ext4 on HDD



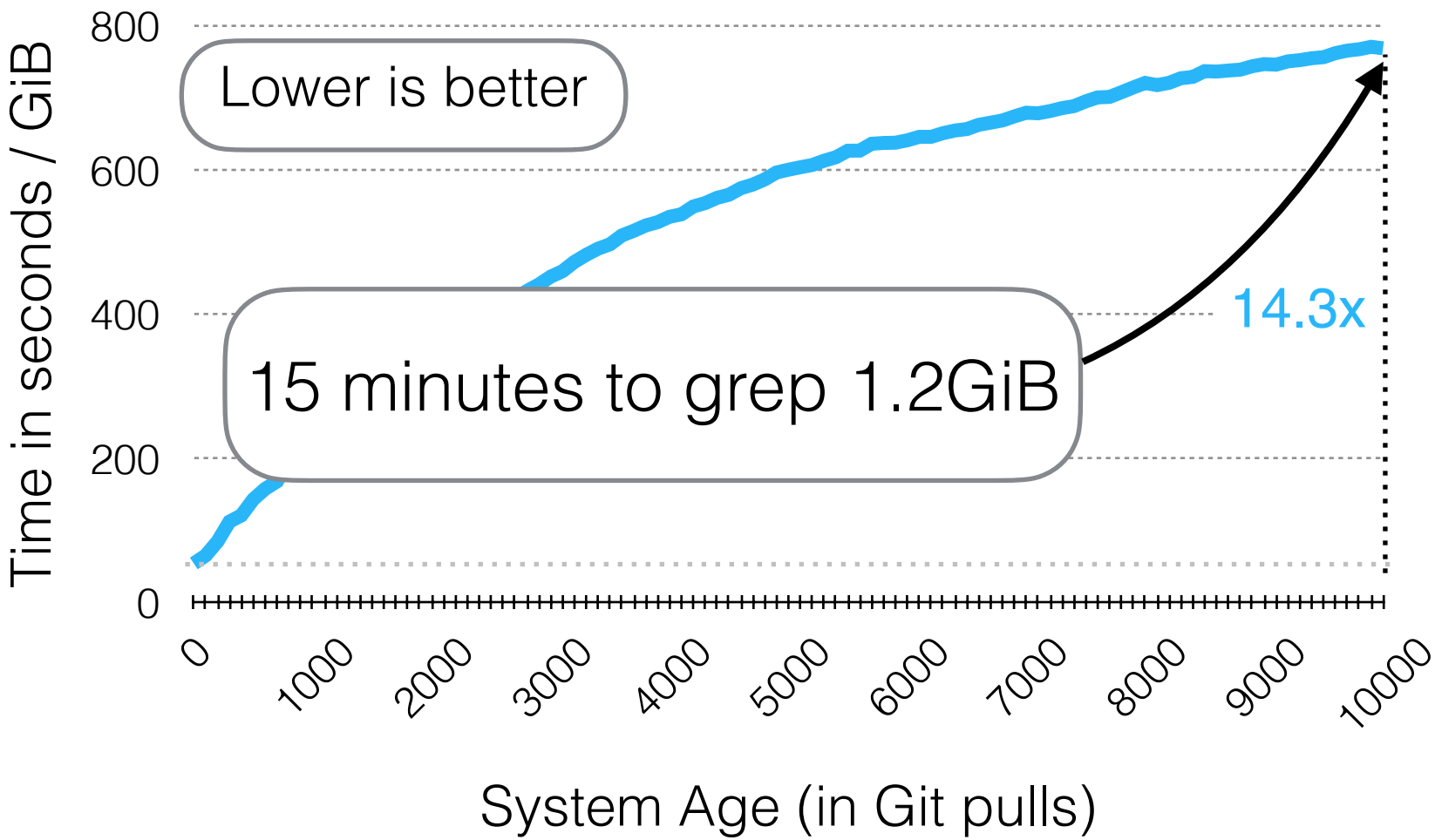
Our Setup: Cold Cache, 3.4 GHz Quad Core, 4GiB RAM,
20 GiB HDD partition - SATA 7200 RPM

Git workload on ext4 on HDD



Our Setup: Cold Cache, 3.4 GHz Quad Core, 4GiB RAM,
20 GiB HDD partition - SATA 7200 RPM

Git workload on ext4 on HDD



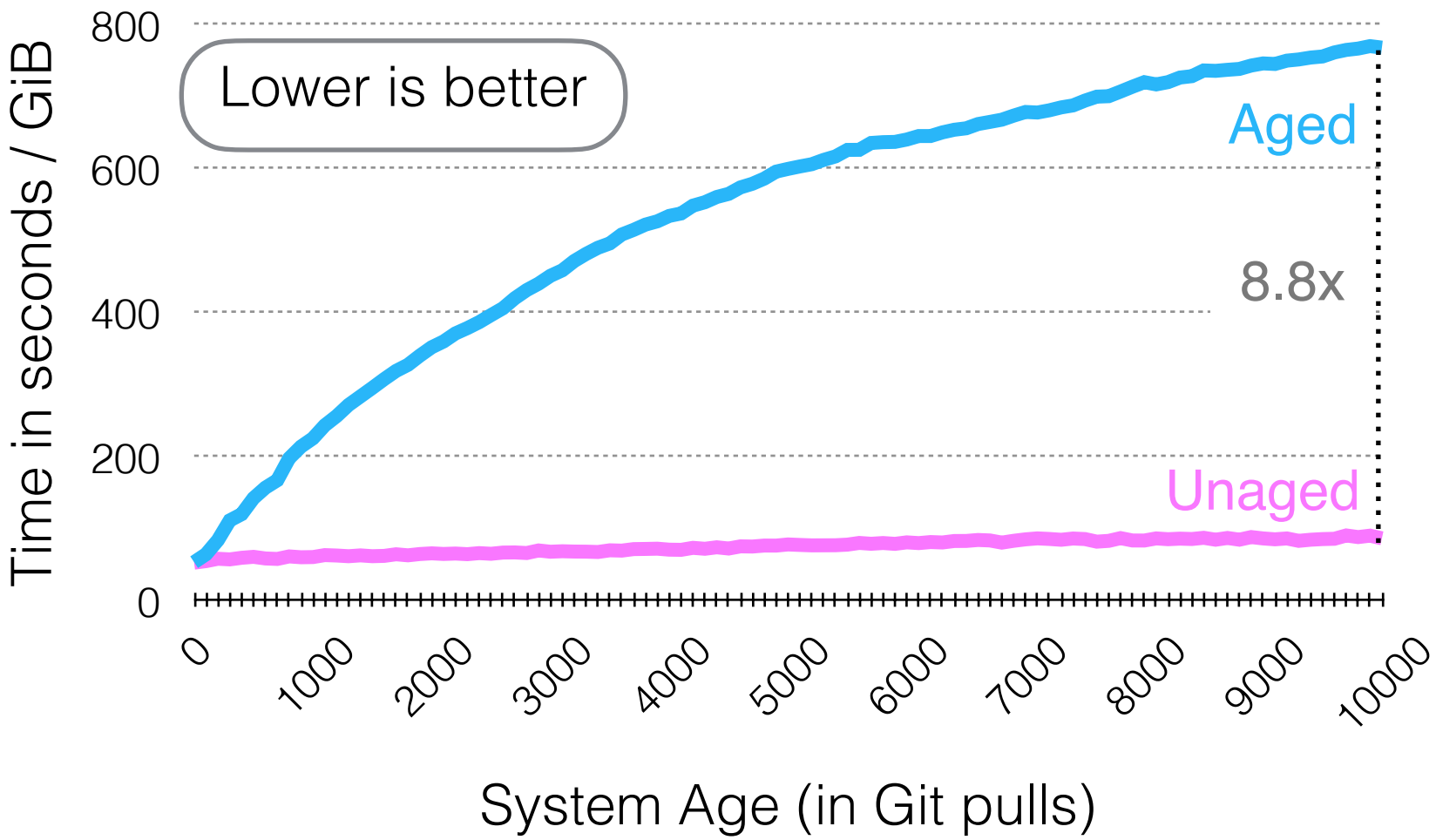
Our Setup: Cold Cache, 3.4 GHz Quad Core, 4GiB RAM,
20 GiB HDD partition - SATA 7200 RPM

Ruling out alternative explanations

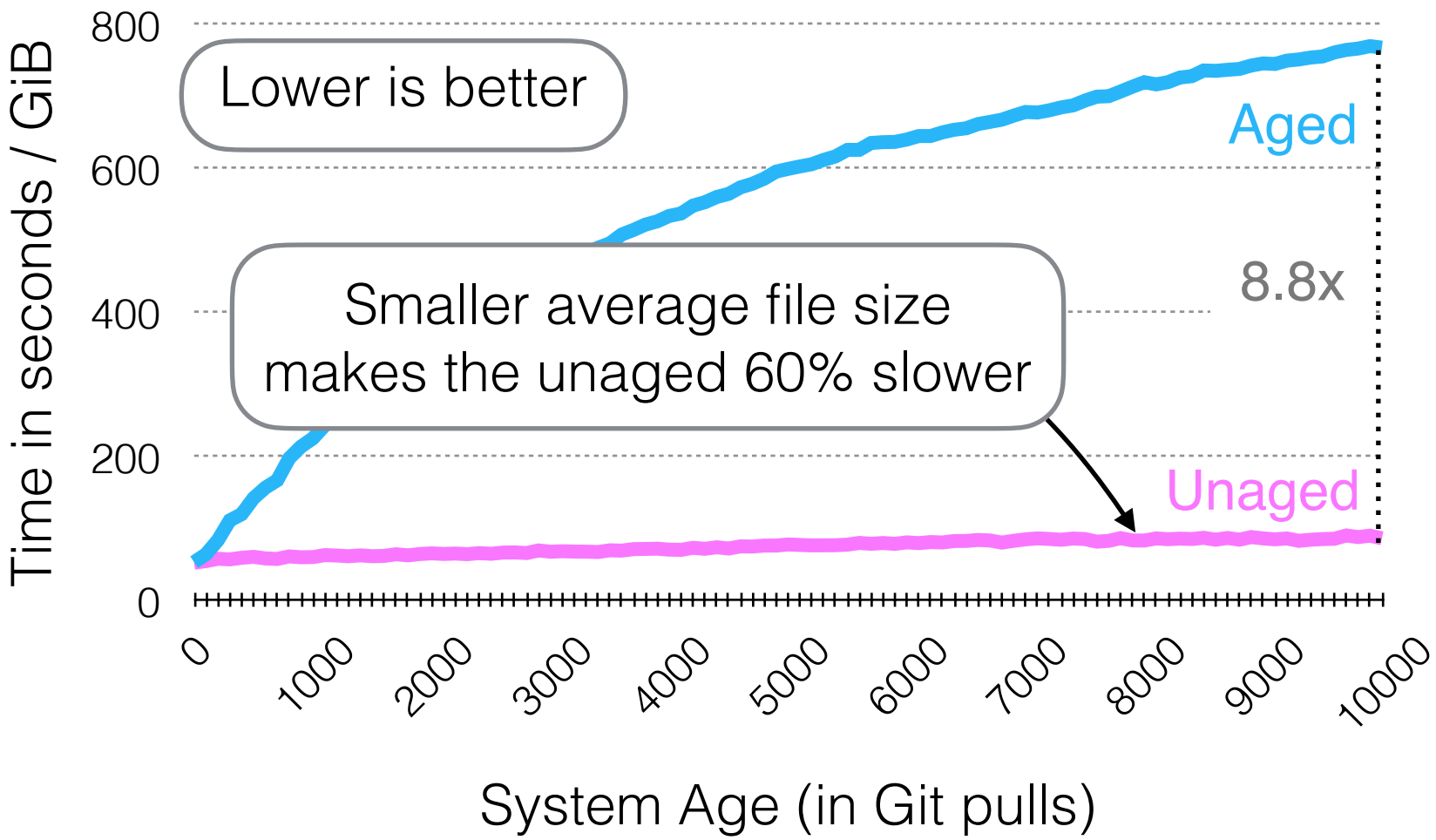
Is it a change in the file system?

Smaller files, shallower tree, …

Aging ext4 with Git on HDD

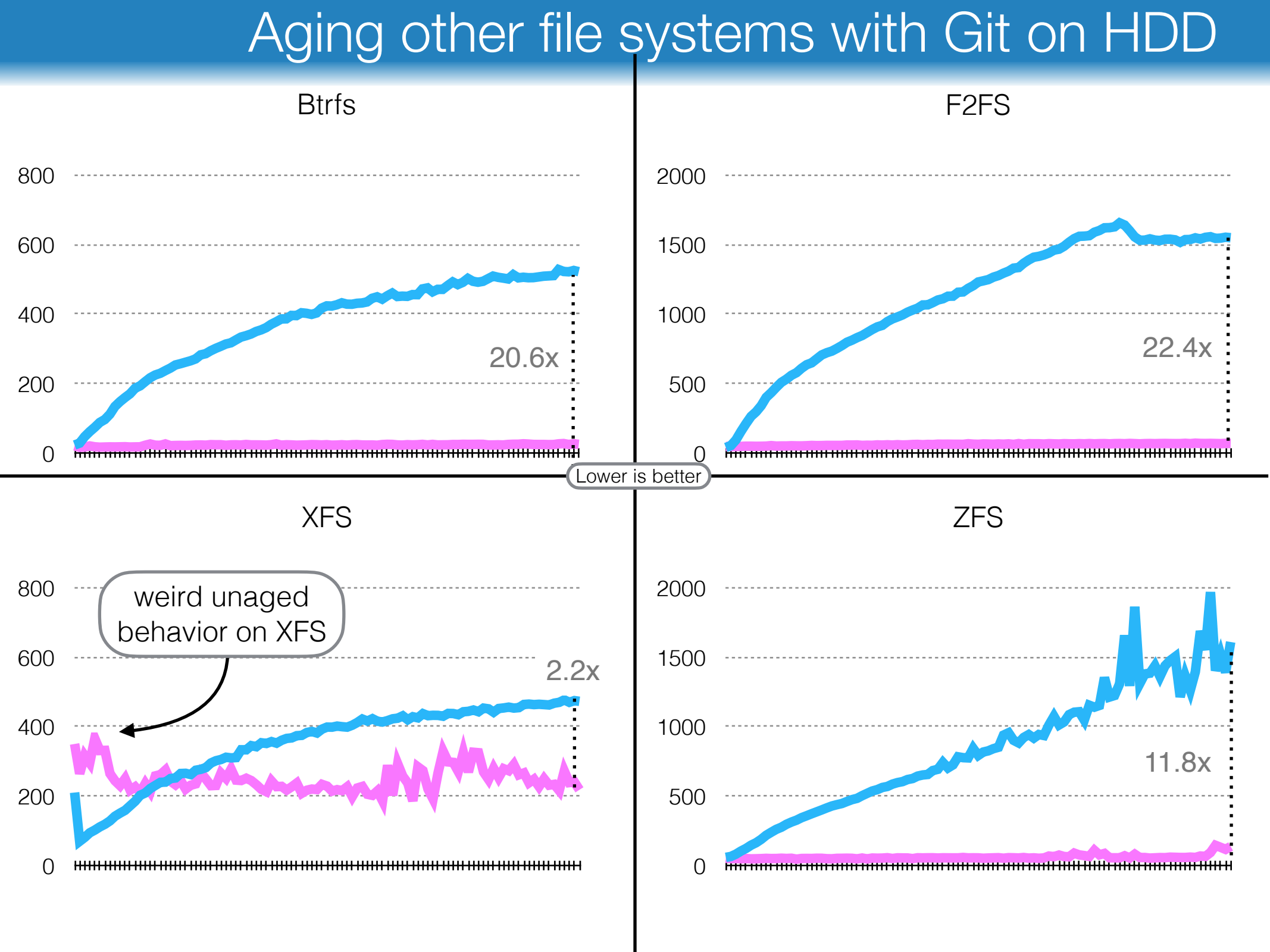


Aging ext4 with Git on HDD



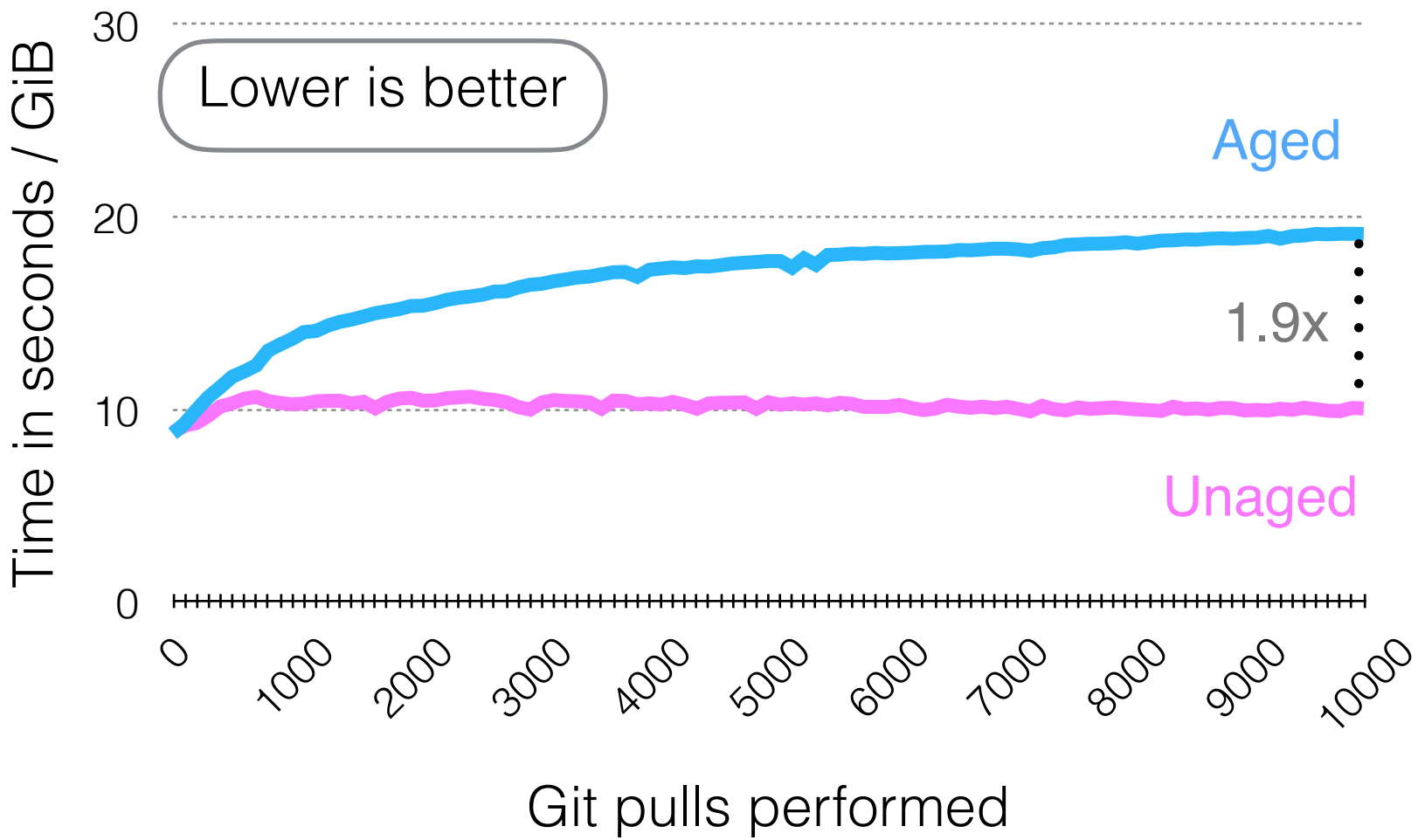
Is it just ext4?

Aging other file systems with Git on HDD



Will SSDs save us?

Git Workload on XFS on SSD



Other file systems give similar results (~2x slowdown)

Key Takeaways

Aging is real

- It's easy to age standard file systems

Aging hides in plain sight

- Aging so fast that FSs are always aged
- Old disks would have shown little aging
  - Because they had a smaller random-vs-sequential gap

If we want to describe a system's performance, we must think carefully about a "representative state" of the system

- We must age our file systems
  - Measuring from a "clean slate" is not realistic
  - Conway et al. proposed *one way* to age a file system that is not file-system-specific, but there are other options
    - However, *unaged* should not be one of them!