Optimizing Every Operation in a Write-Optimized File System

Jun Yuan, Yang Zhan, William Jannen, Prashant Pandey, Amogh Akshintala, Kanchan Chandnani, Pooja Deo, Zardosht Kasheff, Leif Walsh, Michael Bender, Martin Farach-Colton, **Rob Johnson**, Bradley C. Kuszmaul, Donald E. Porter

Stony Brook University, Facebook, Two Sigma, Rutgers University, Massachusetts Institute of Technology

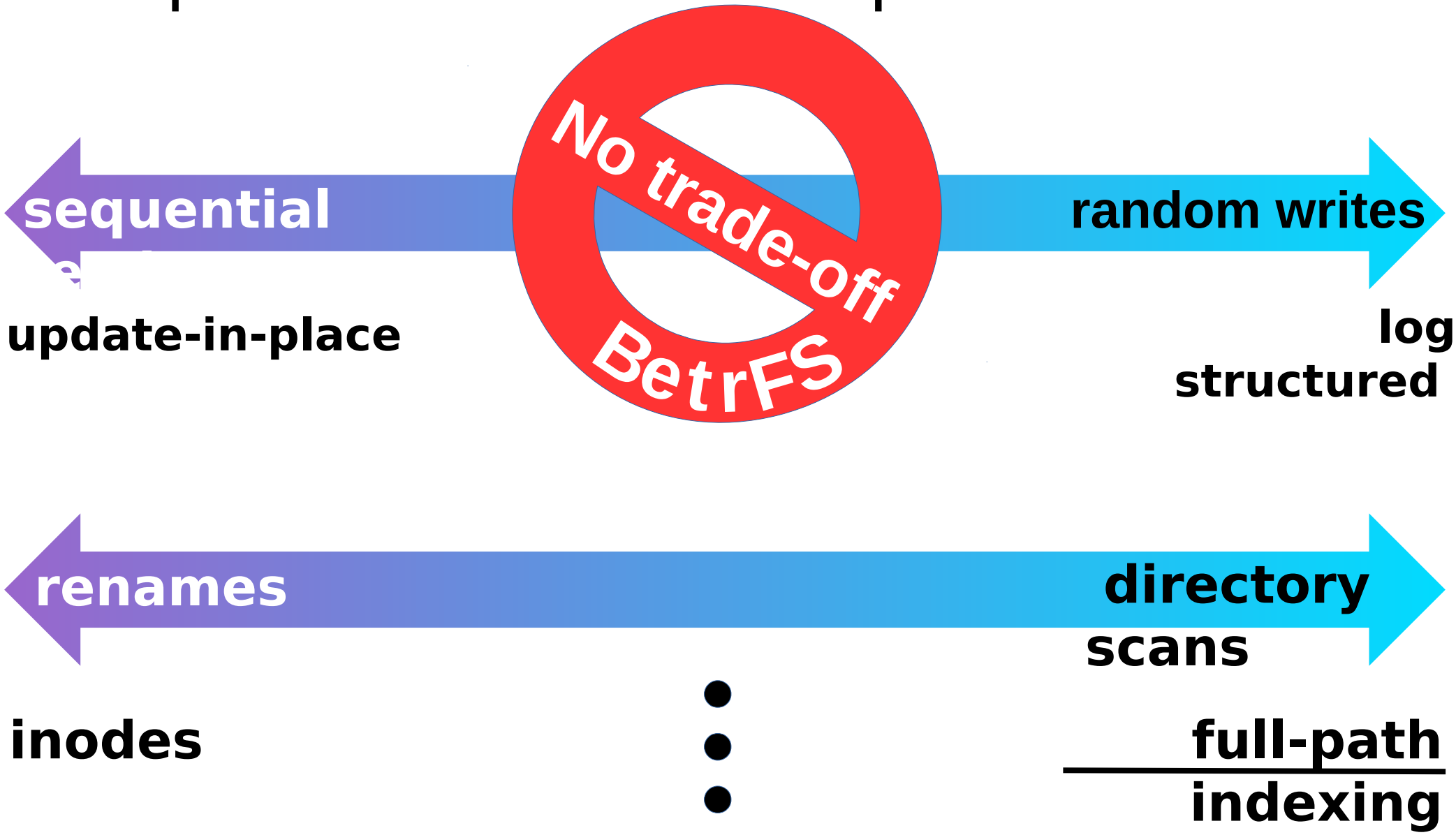
BetrFS goal:

High-performance, general-purpose file system

Need to perform well on many operations:

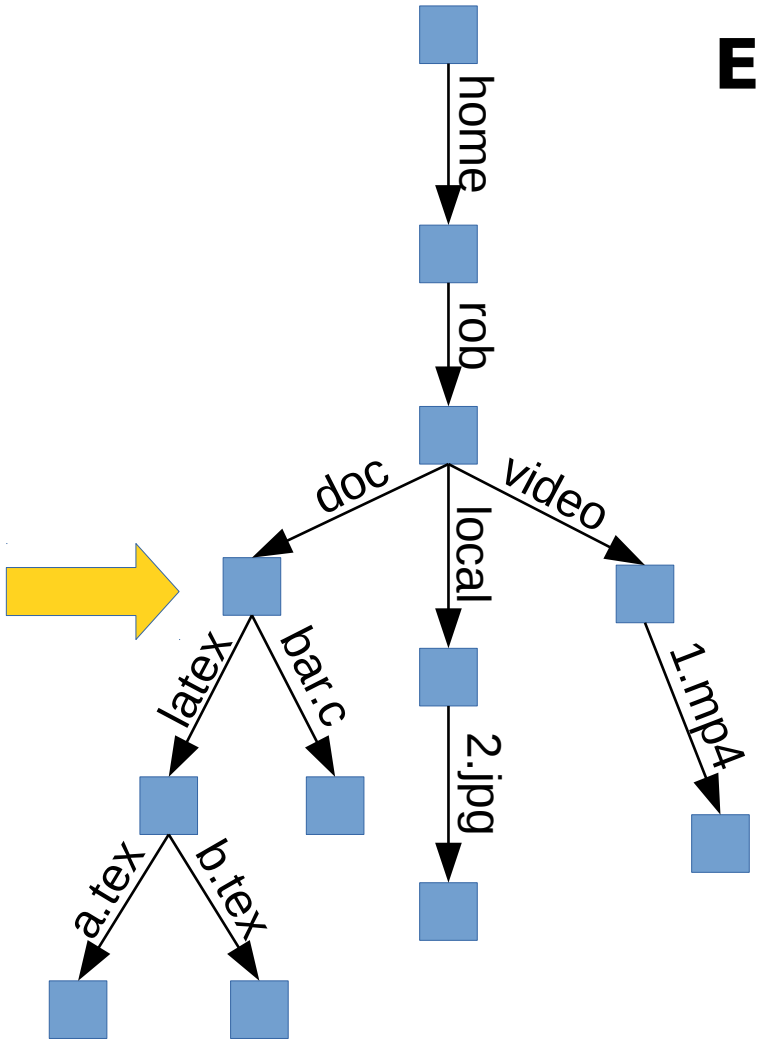
- Sequential reads
- Sequential writes
- Random writes
- File/directory renames
- File deletes
- Recursive scans
- Metadata updates

Some operations seem to require a trade-off

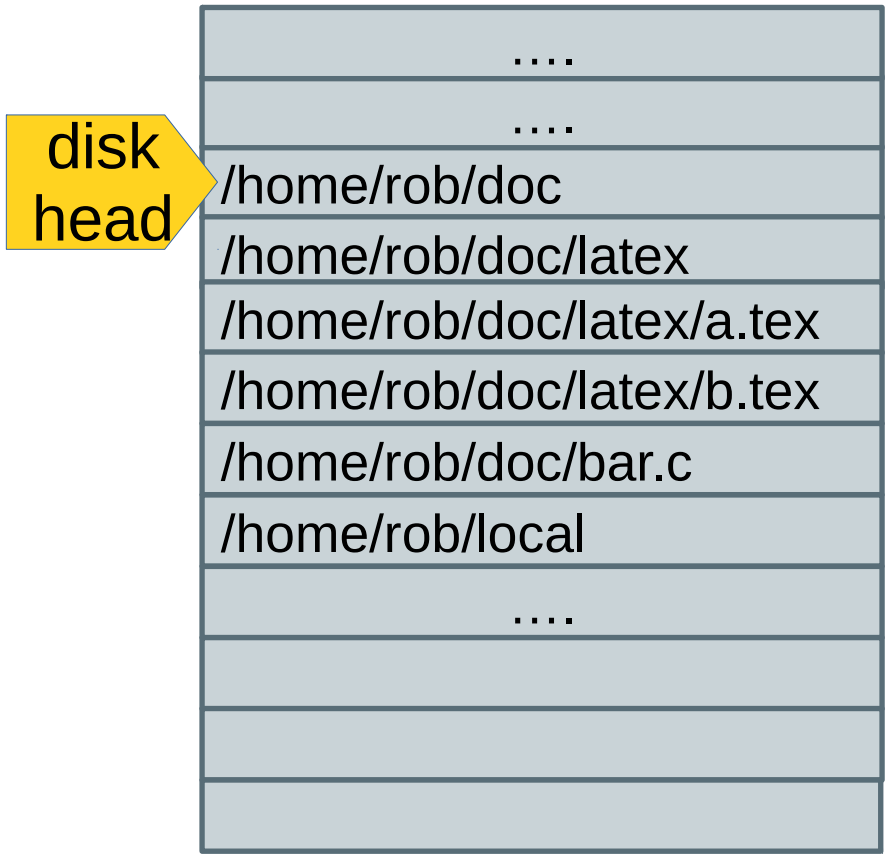


Full-path indexing yields fast directory scans

Example: grep -r “key” /home/rob/doc/



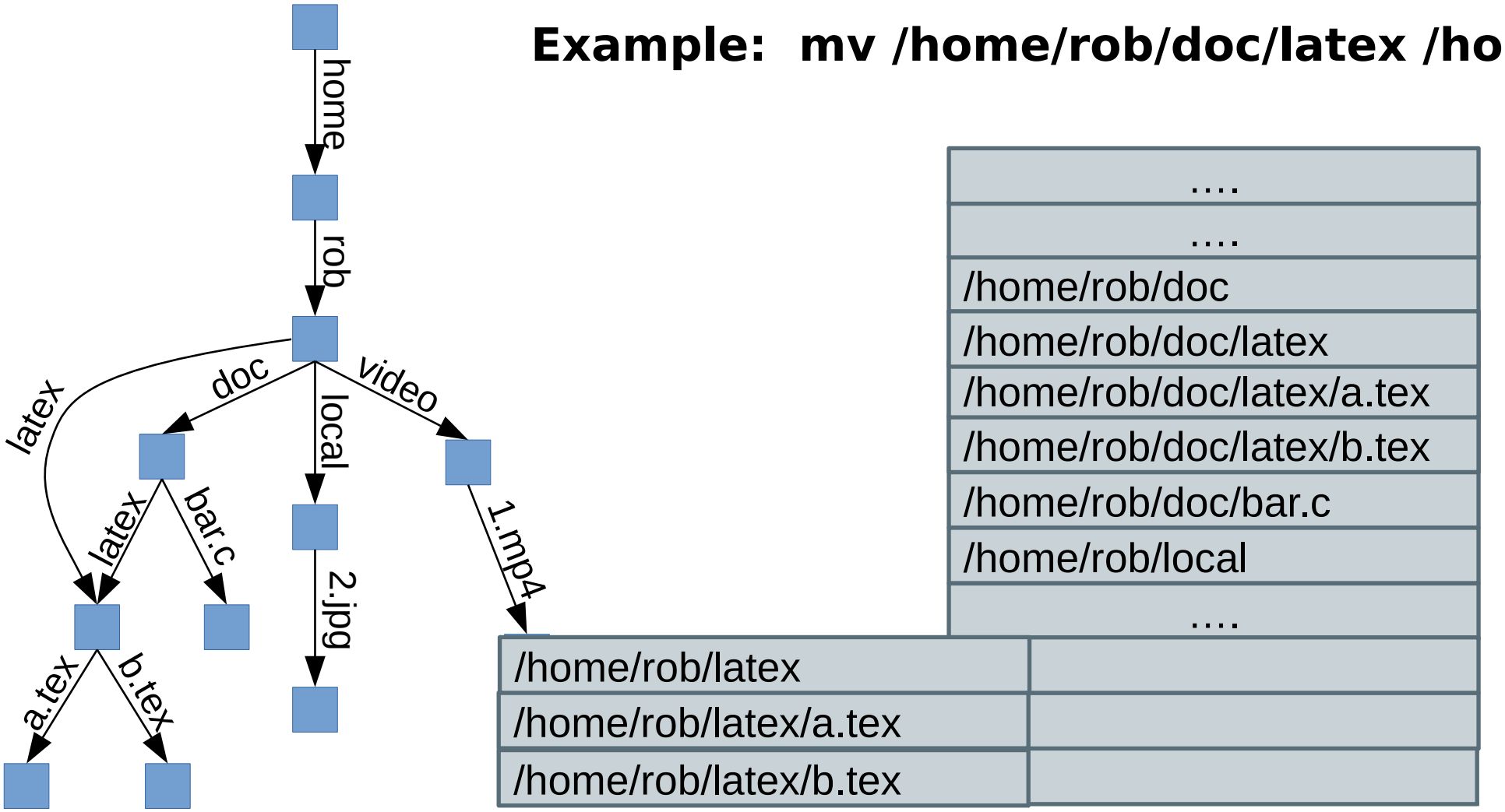
Directory Tree (logical)



Disk (physical)

Rename is expensive when using full-path indexing

Example: mv /home/rob/doc/latex /home/rob

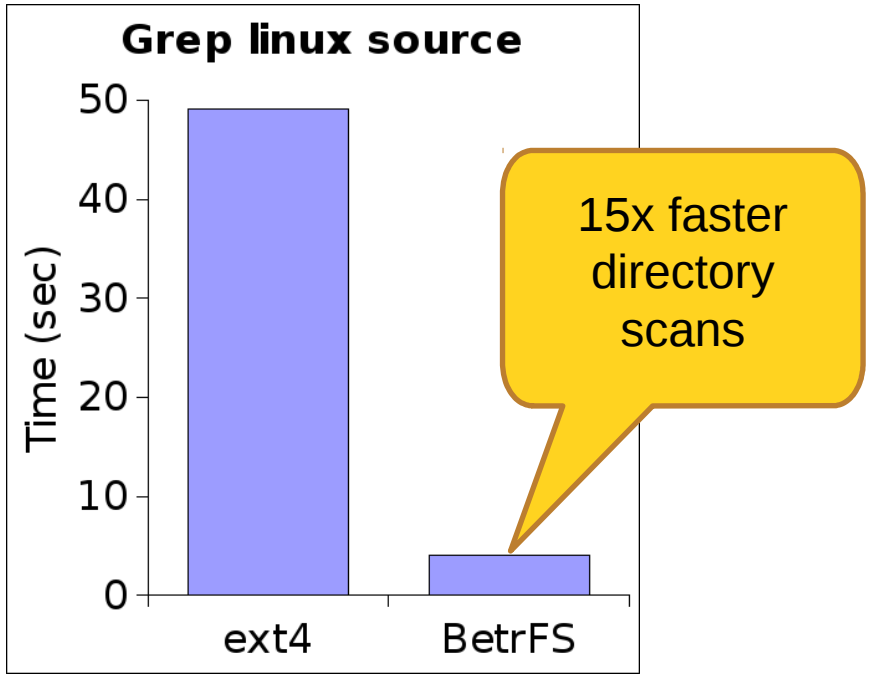


Directory Tree (logical)

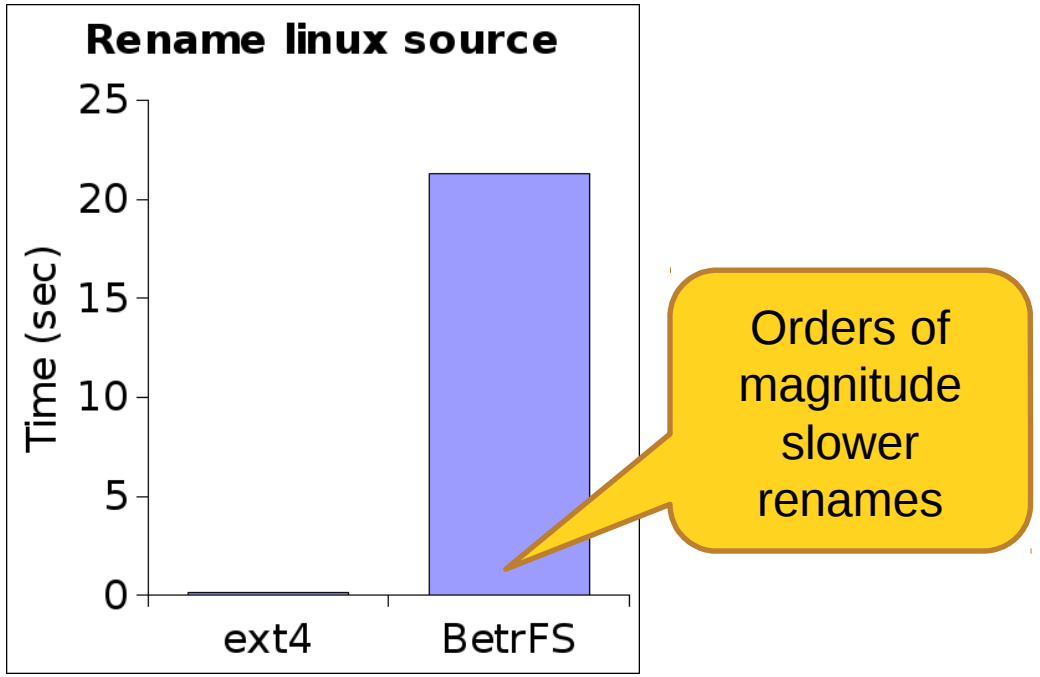
Disk (physical)

This trade-off affects real performance

- Ext4 uses inodes
- BetrFS uses full-path indexing



Lower is better



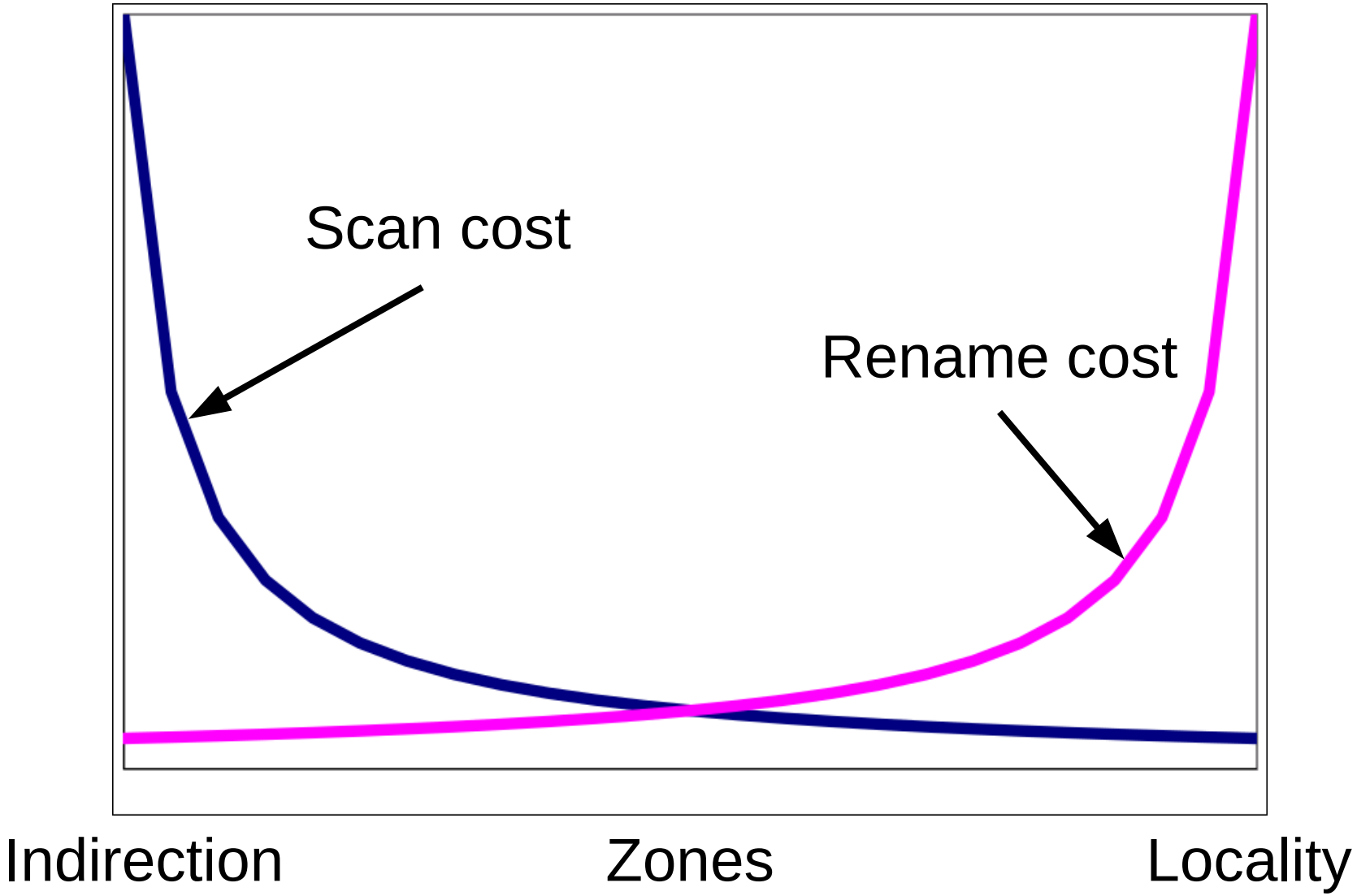
Lower is better

BetrFS and ext4 represent different trade-offs between directory scan and rename performance

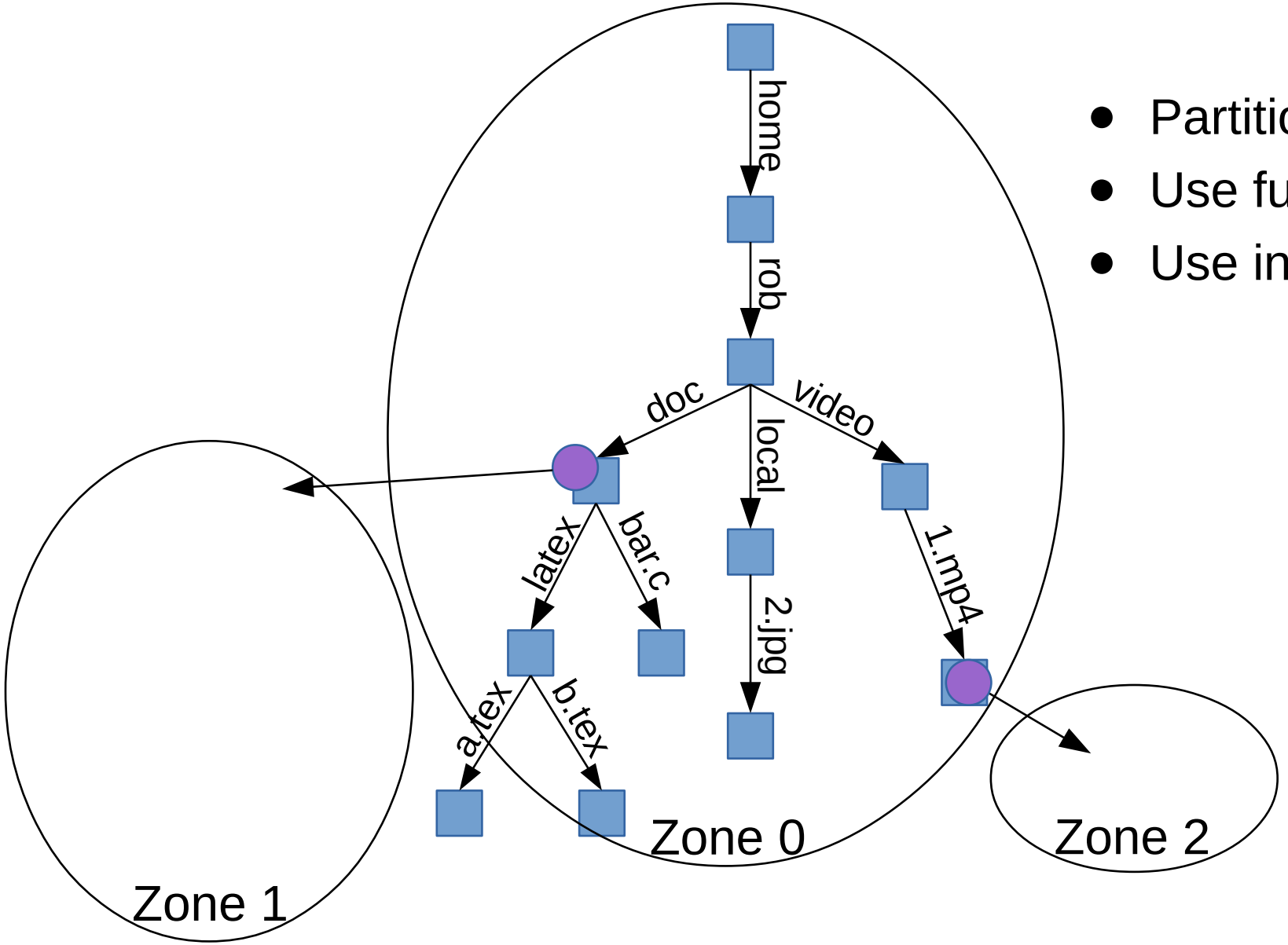
Outline

- Zoning: a technique for fast renames + scans
- Other contributions (sketch)
- Evaluation

Zoning: balancing indirection and locality



Zone: a subtree of the directory hierarchy

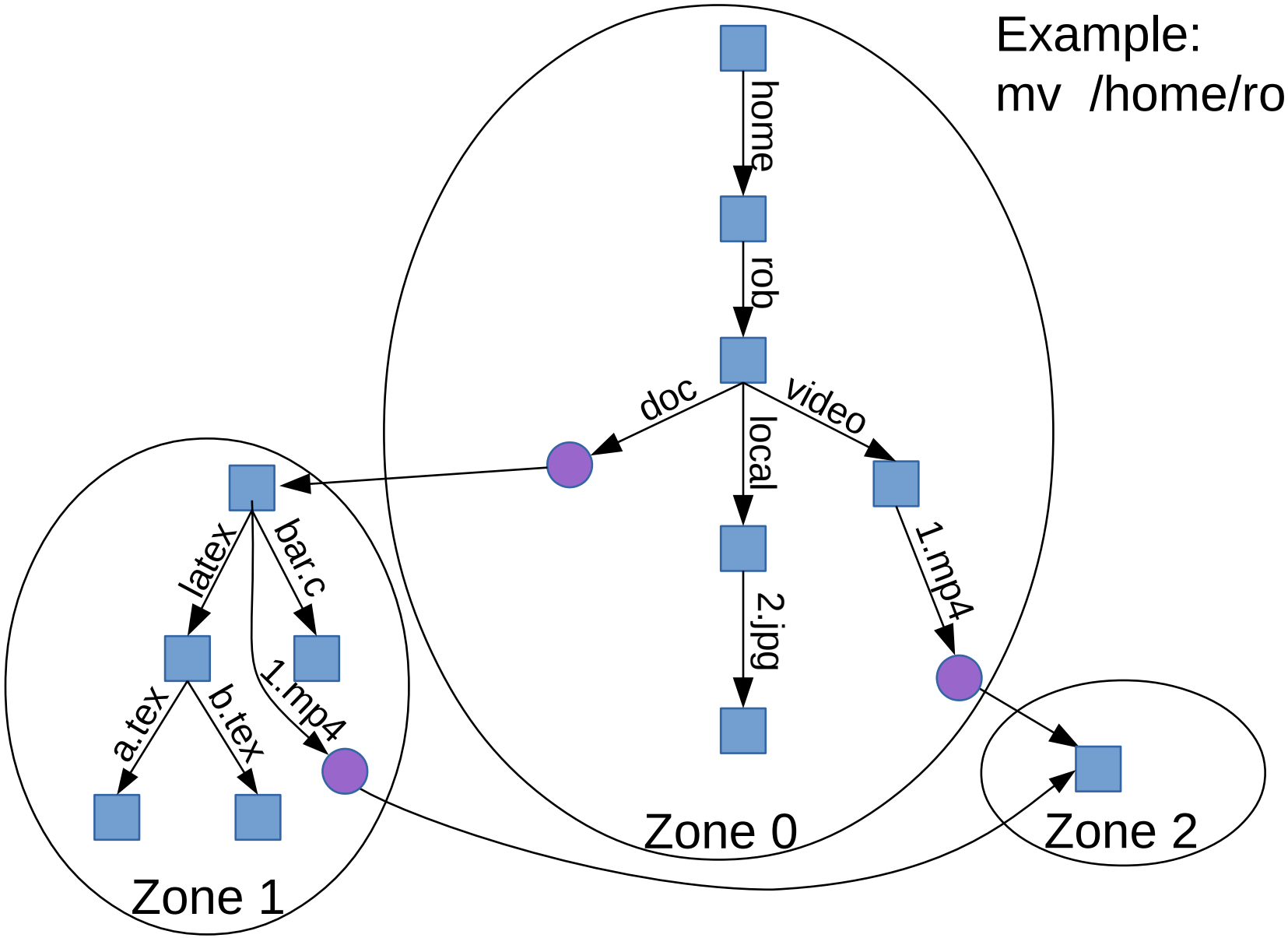


- Partition file system into zones
- Use full-path indexing within zones
- Use inodes between zones

Implication: Recursive directory scans only perform seeks when crossing zones

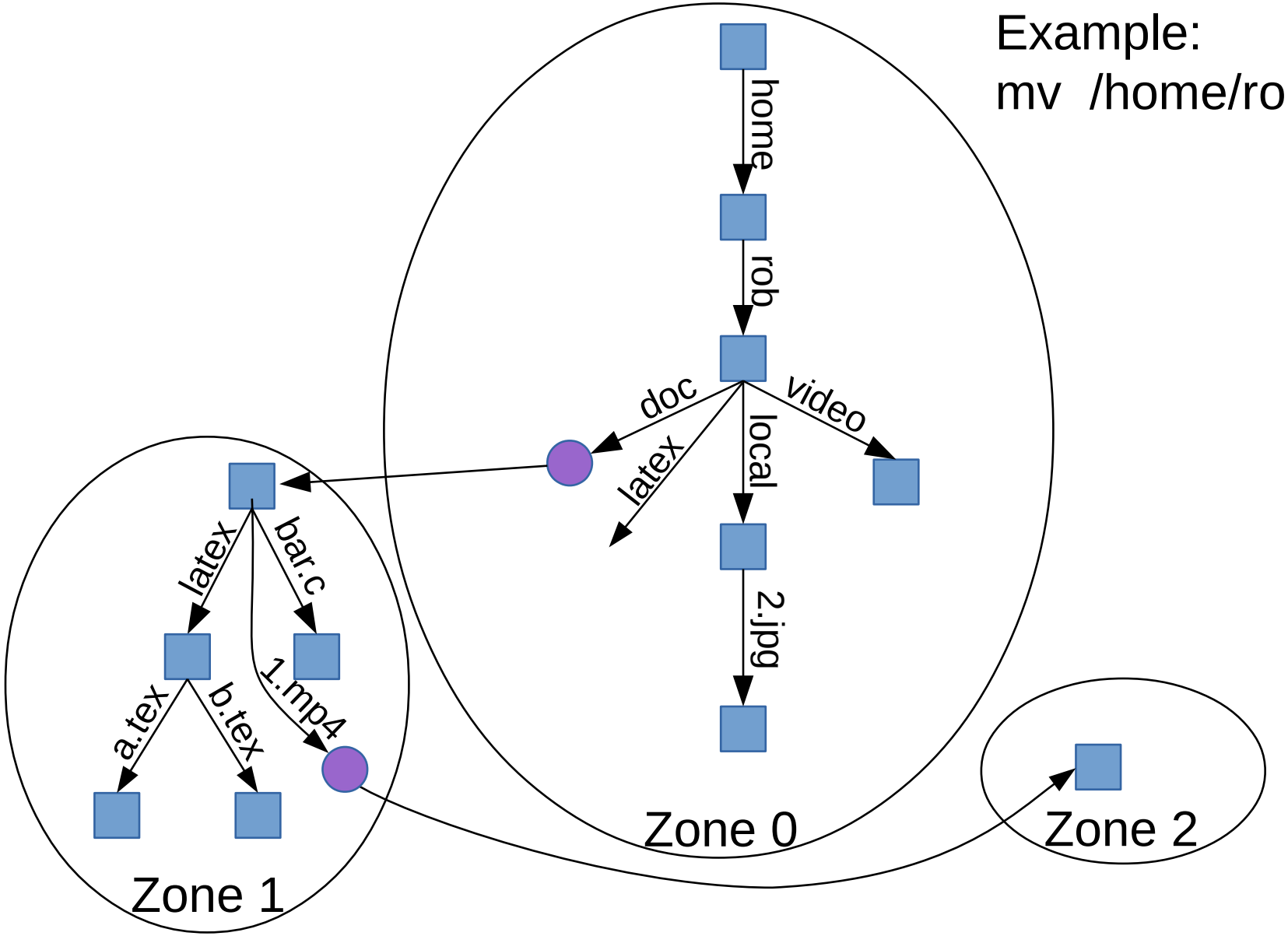
Moving the root of a zone is cheap

Example:
mv /home/rob/video/1.mp4 /home/rob/doc

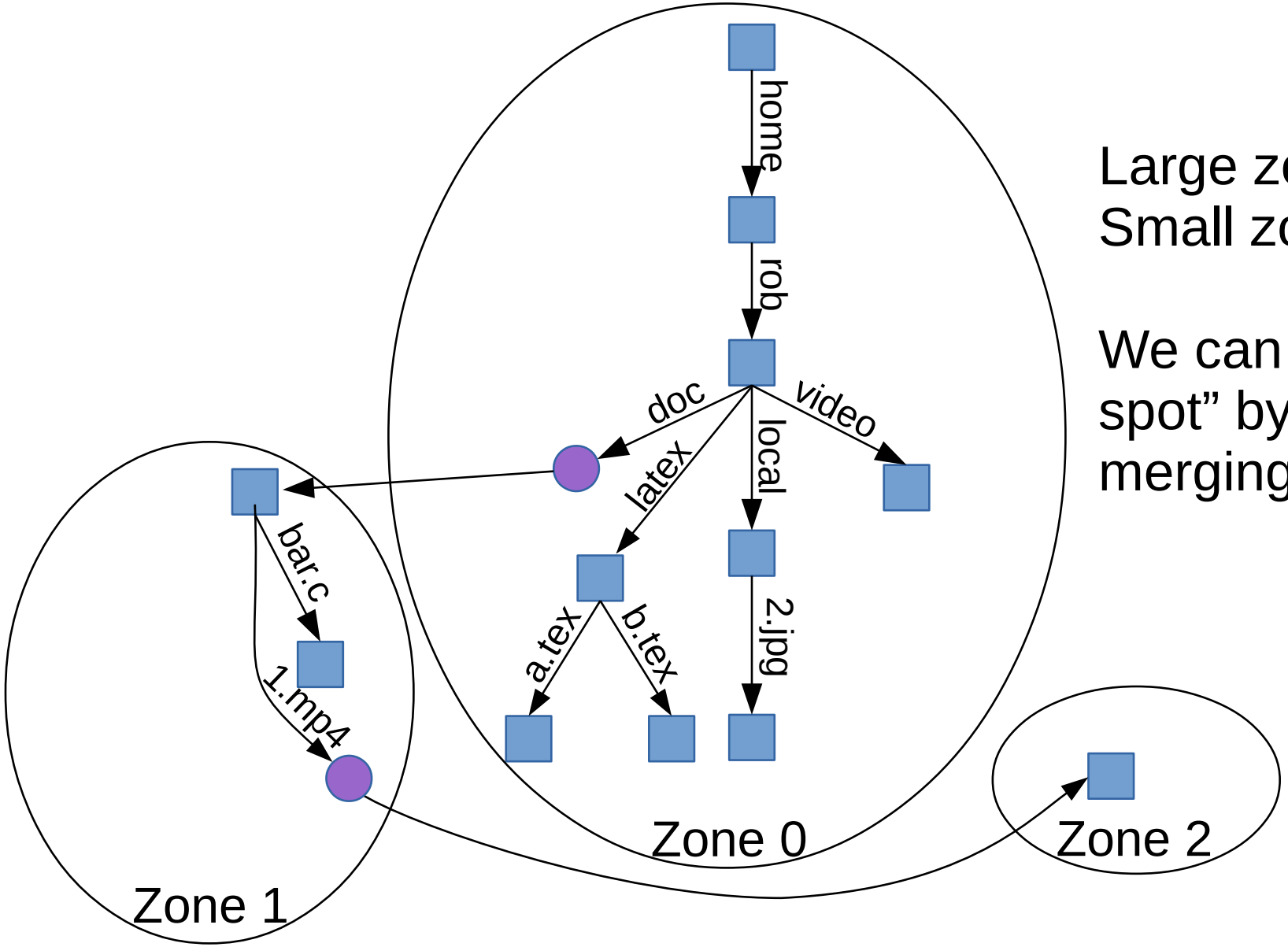


Renaming a subtree of a zone requires copying

Example:
mv /home/rob/doc/latex /home/rob/latex



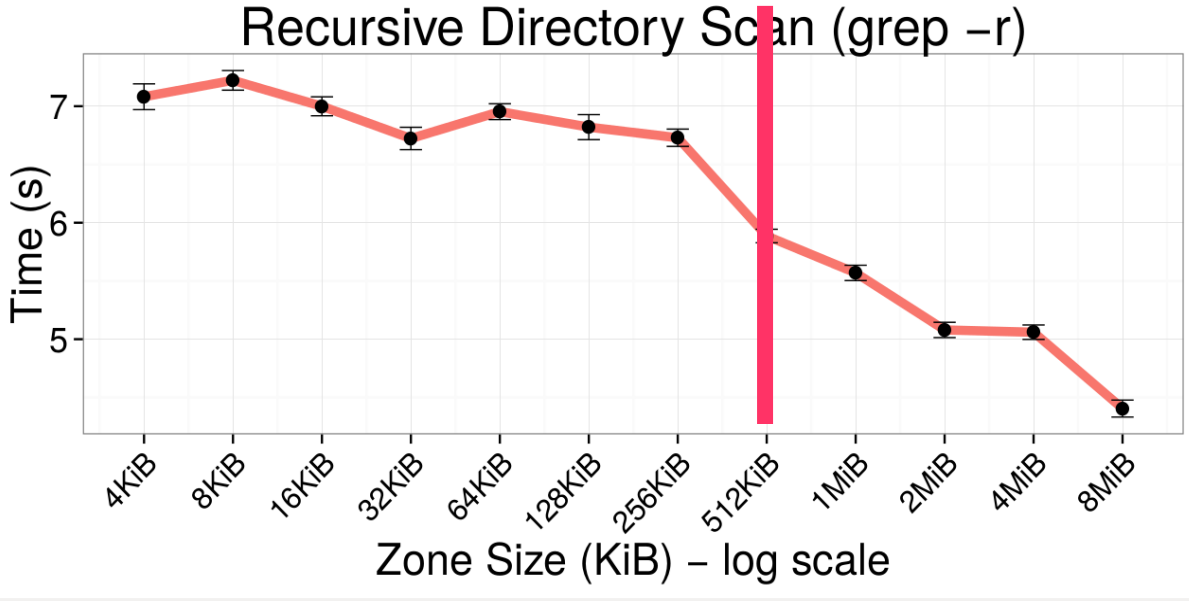
Managing zone sizes

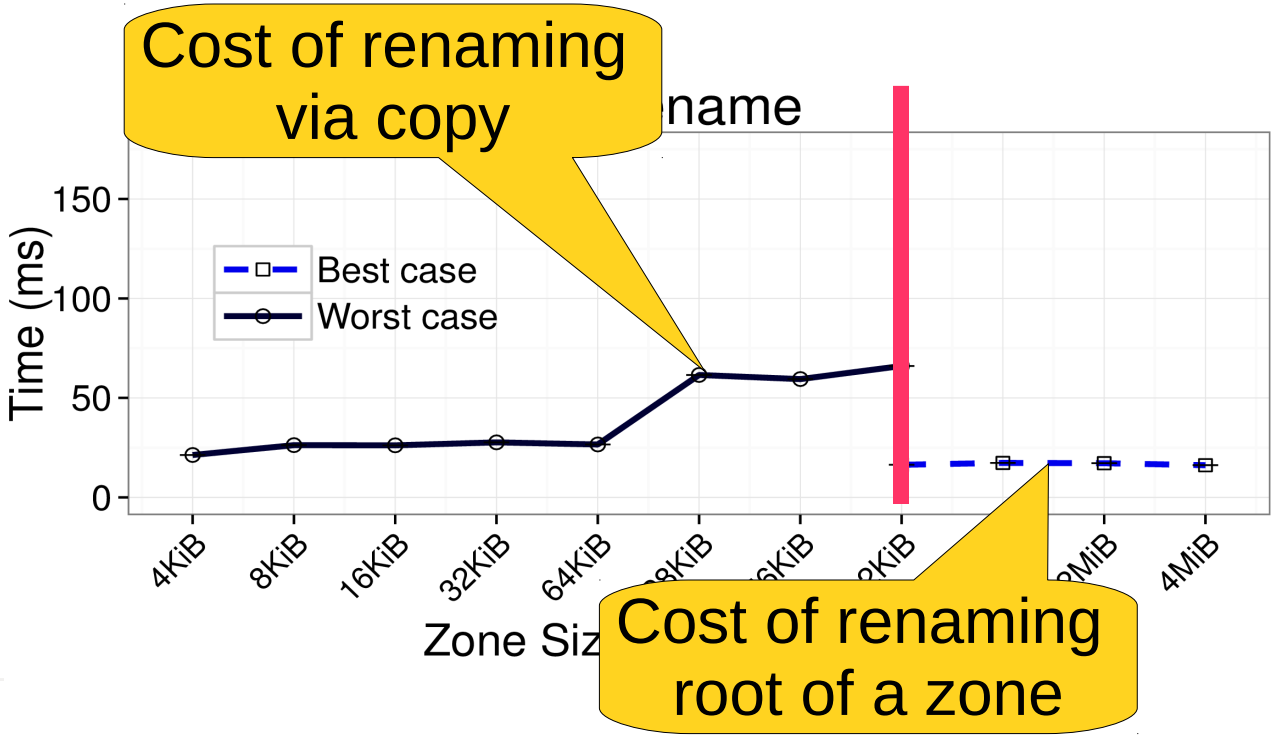


Large zones → fast directory scans
Small zones → fast renames

We can keep zone sizes in a “sweet spot” by splitting large zones and merging small zones

How big should zones be?



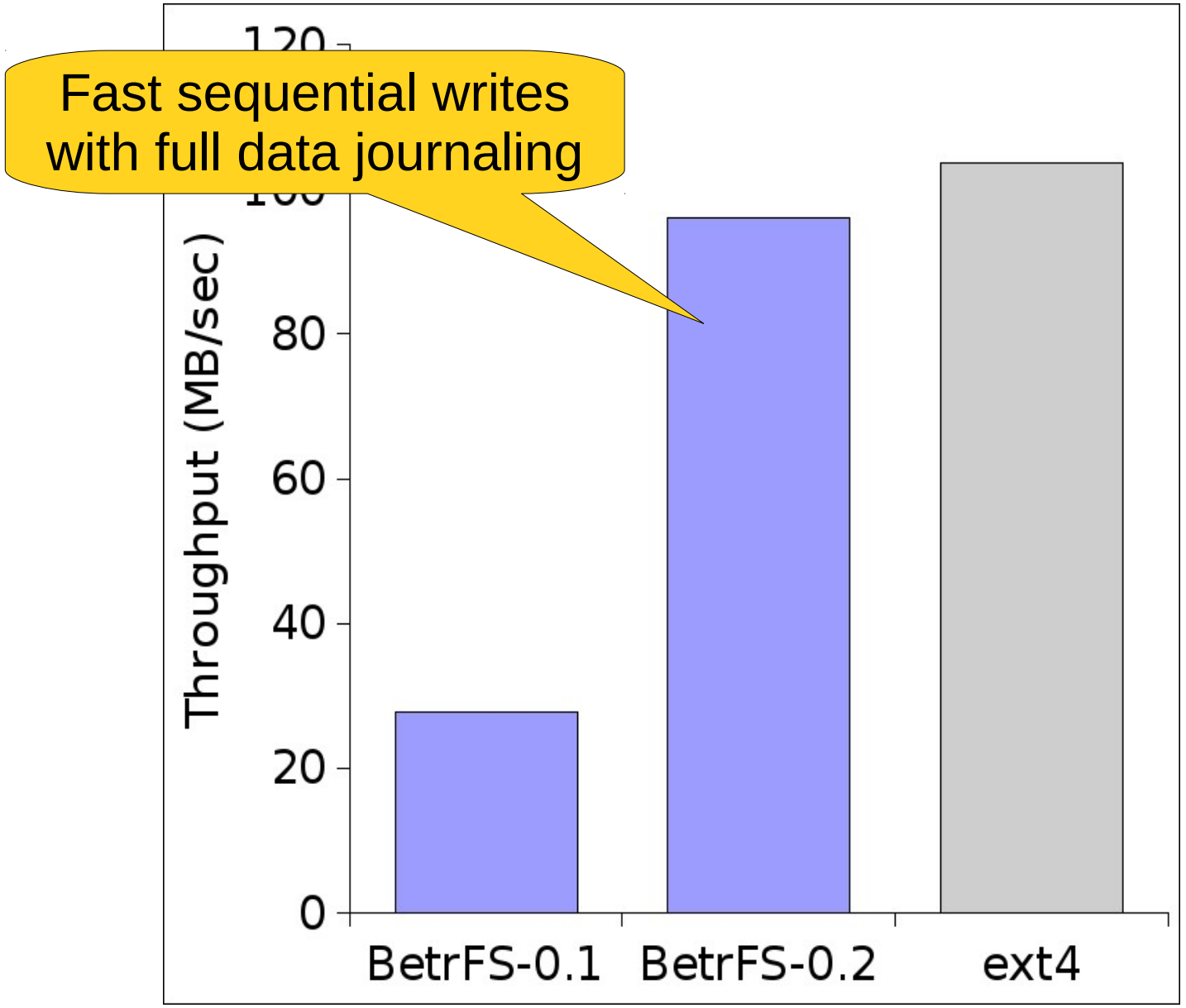


BetrFS-0.2 uses 512KB zones to balance rename and scan performance

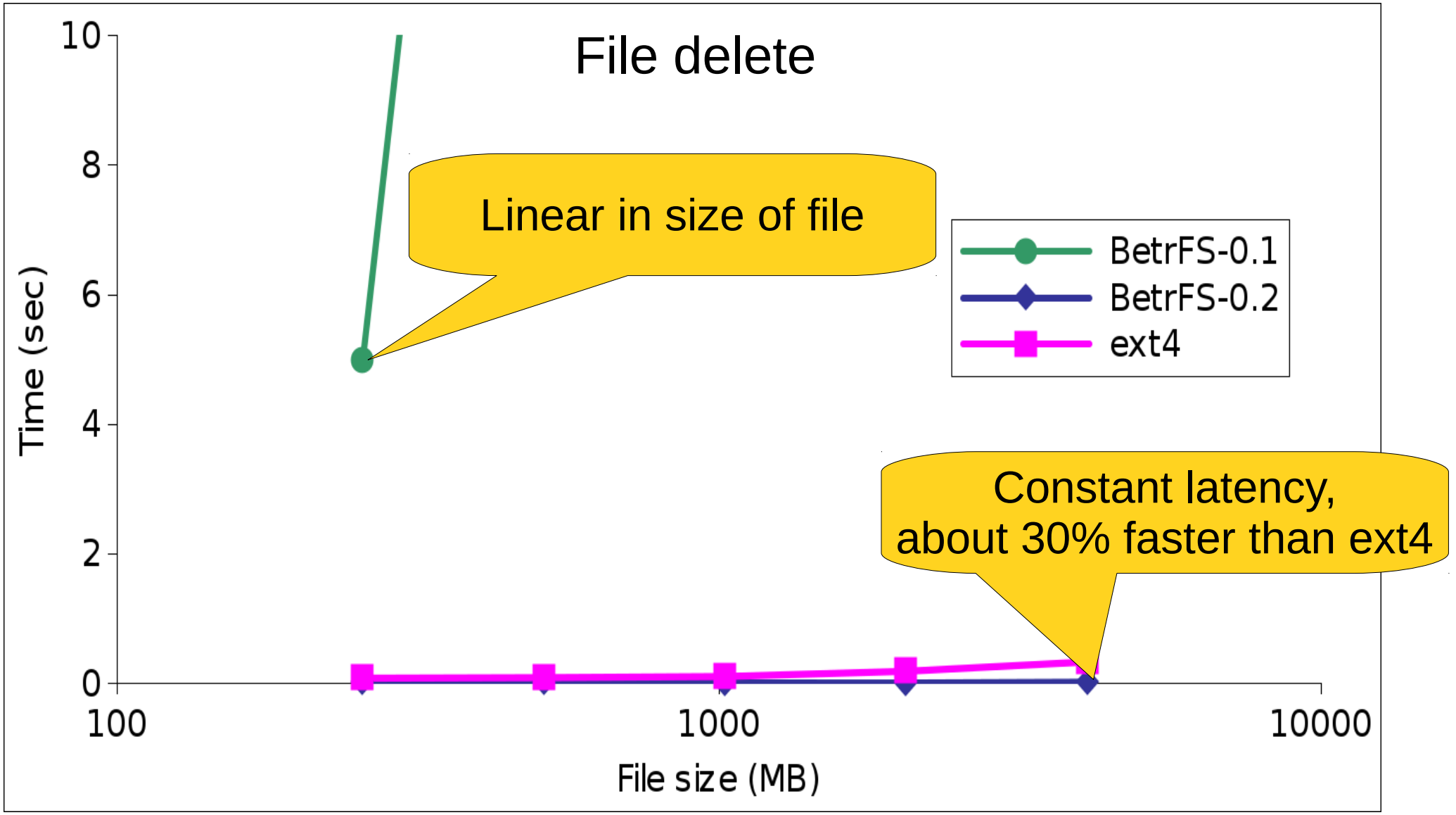
Other contributions

Late-binding journal

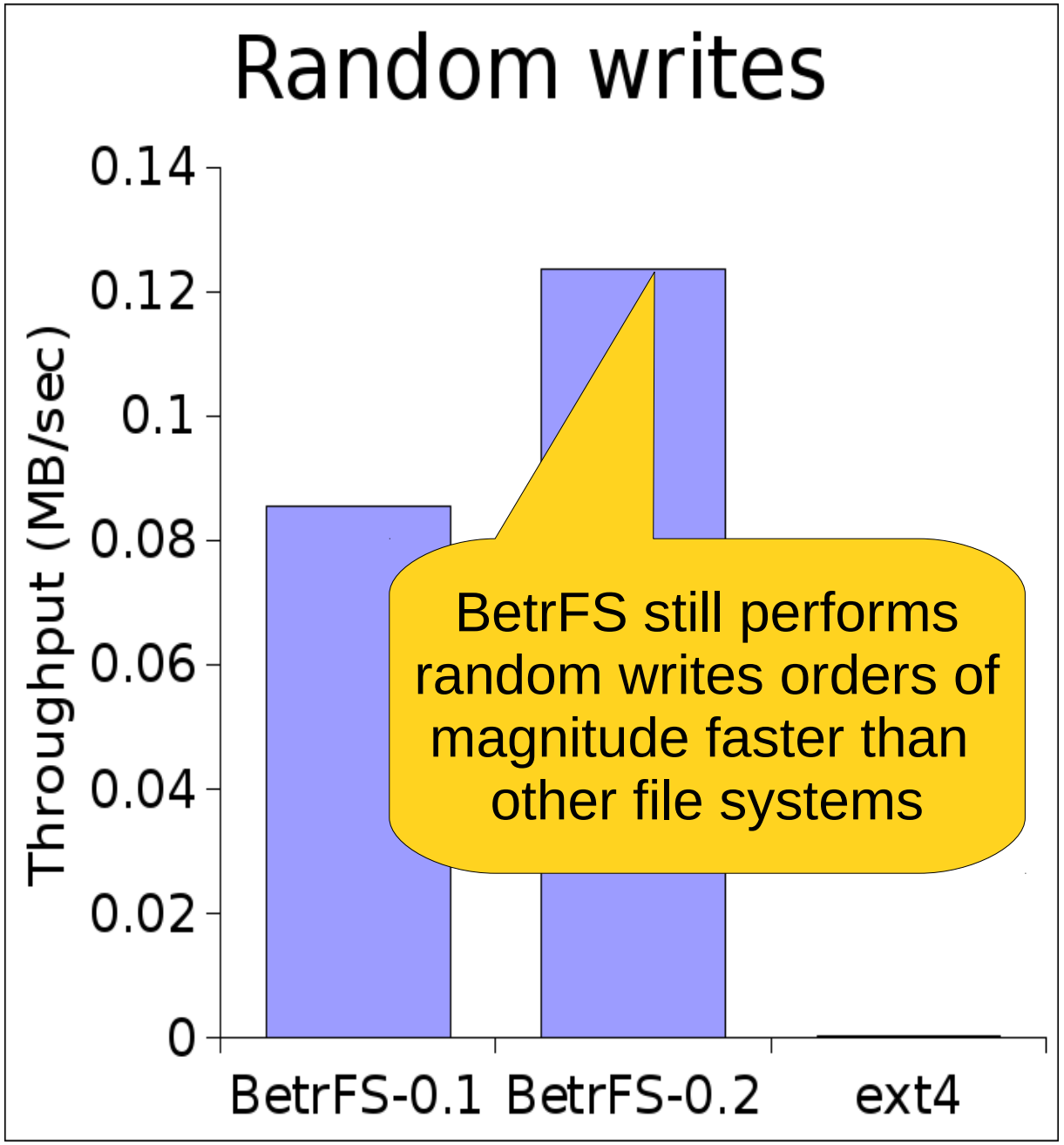
See paper for details

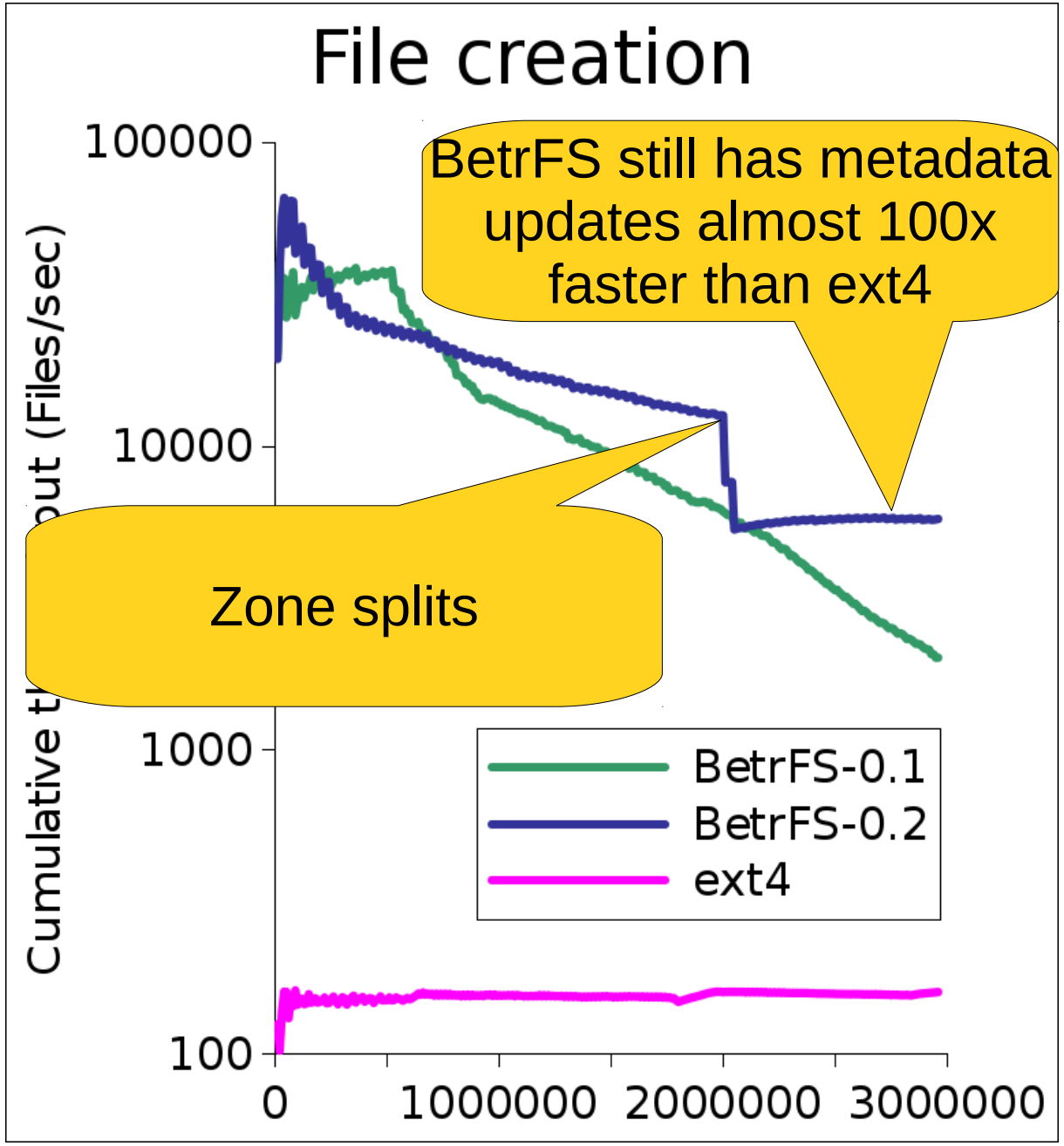


Rangecast delete



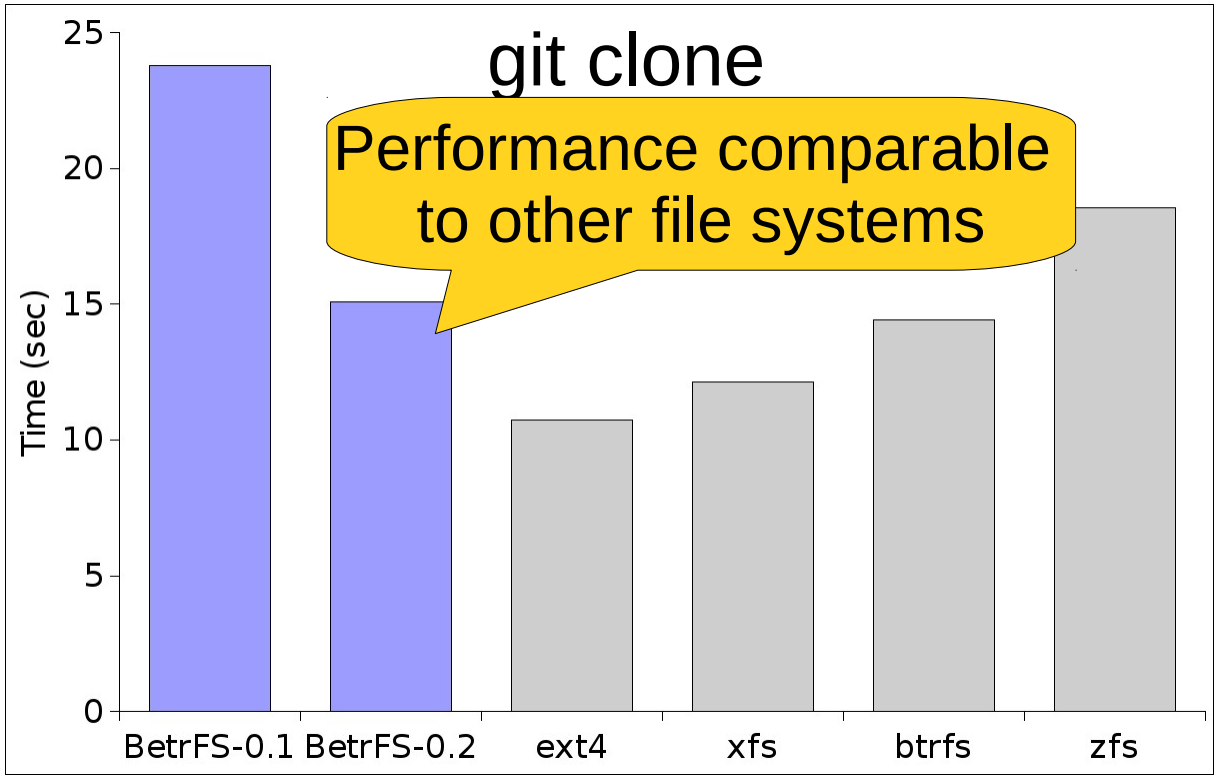
Is BetrFS still fast at other operations?

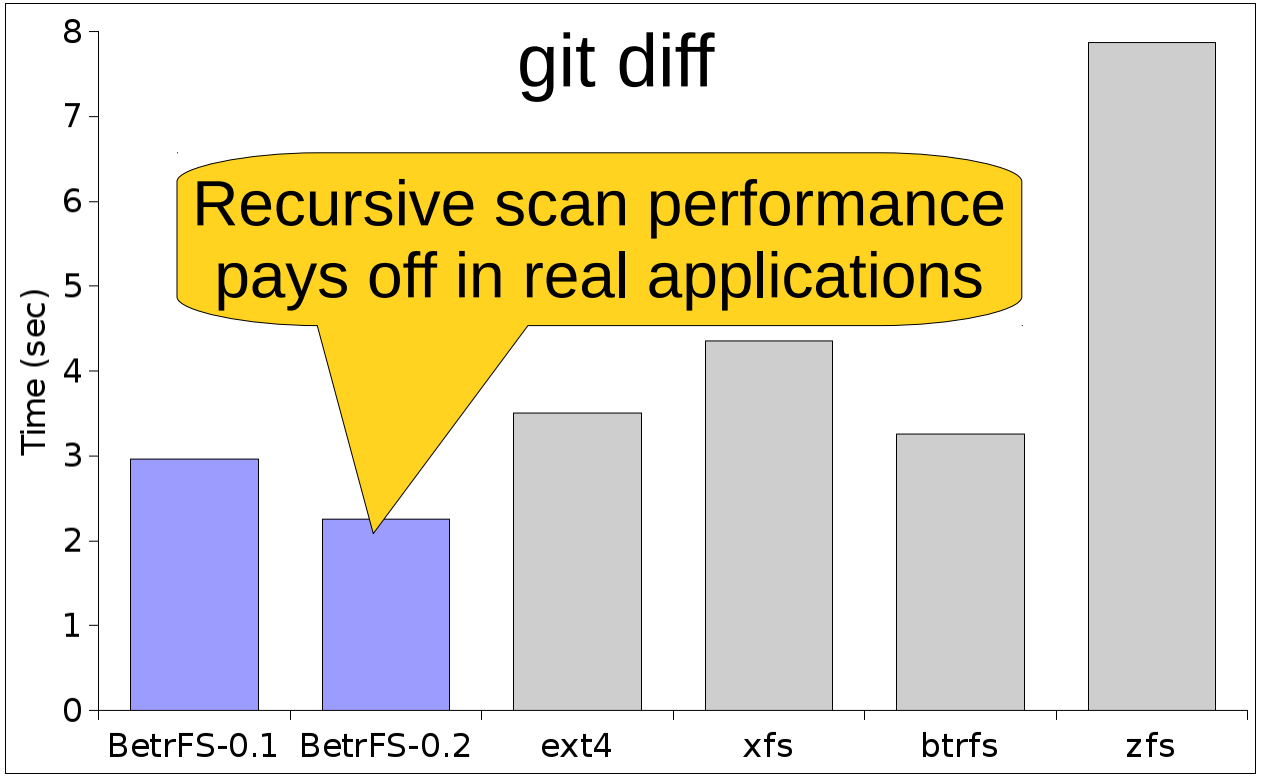

Random writes
Throughput (MB/sec)
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
BetrFS-0.1
BetrFS-0.2
ext4
BetrFS still performs random writes orders of magnitude faster than other file systems


File creation
Cumulative t
100000
10000
1000
100
0
1000000
2000000
3000000
BetrFS still has metadata updates almost 100x faster than ext4
Zone splits
BetrFS-0.1
BetrFS-0.2
ext4

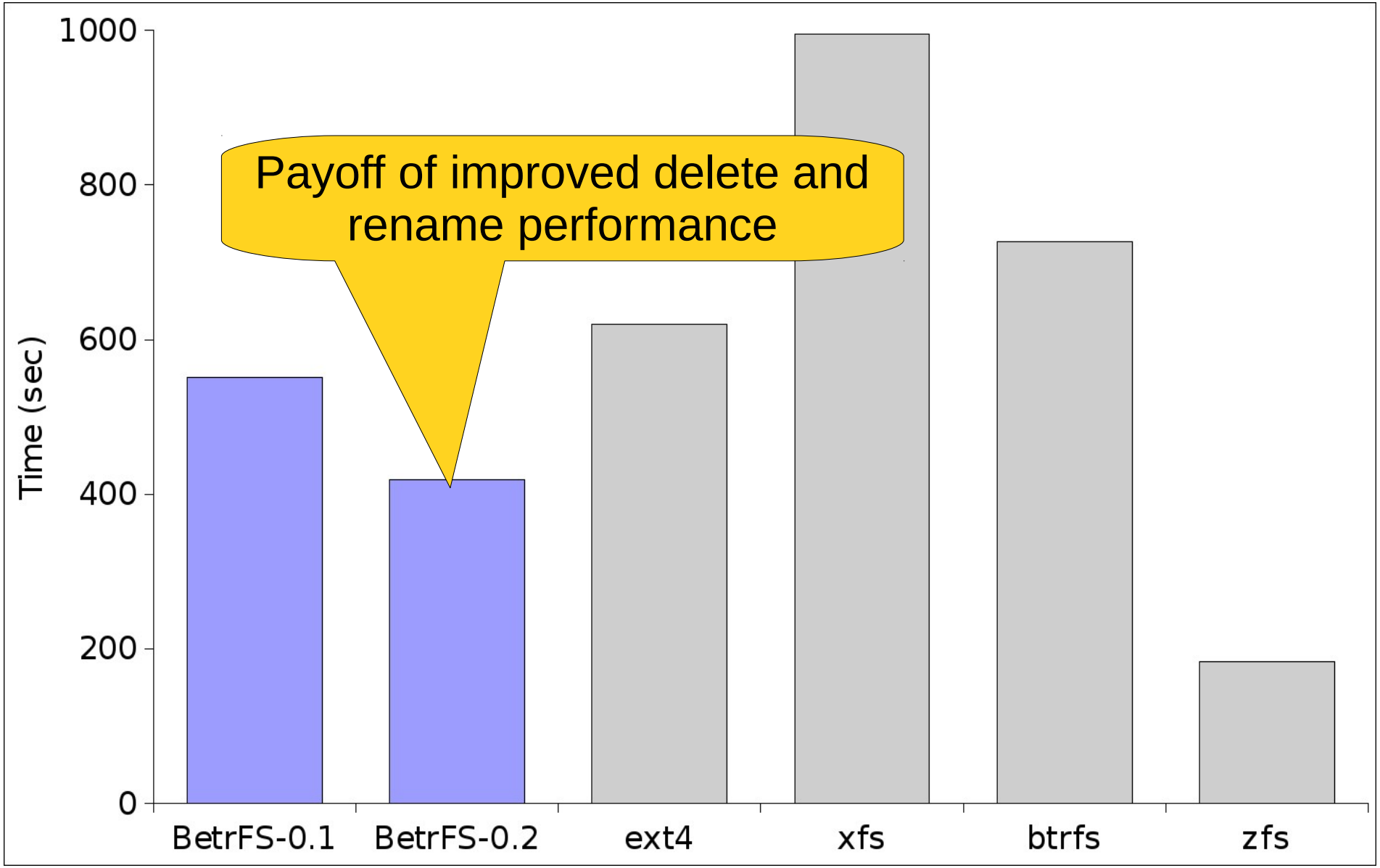
What about real application performance?

Macrobenchmark: git





Macrobenchmark: dovecot imap maildir workload



Conclusion

- A write-optimized file system can be general purpose
  - Write optimization is not a trade-off

- BetrFS has strong performance across many operations
- And across many applications

- Opportunity to re-examine file system trade-offs in light of new data structures

Code available at
betrfs.org

SSD performance preview