The Full Path to Full-Path Indexing

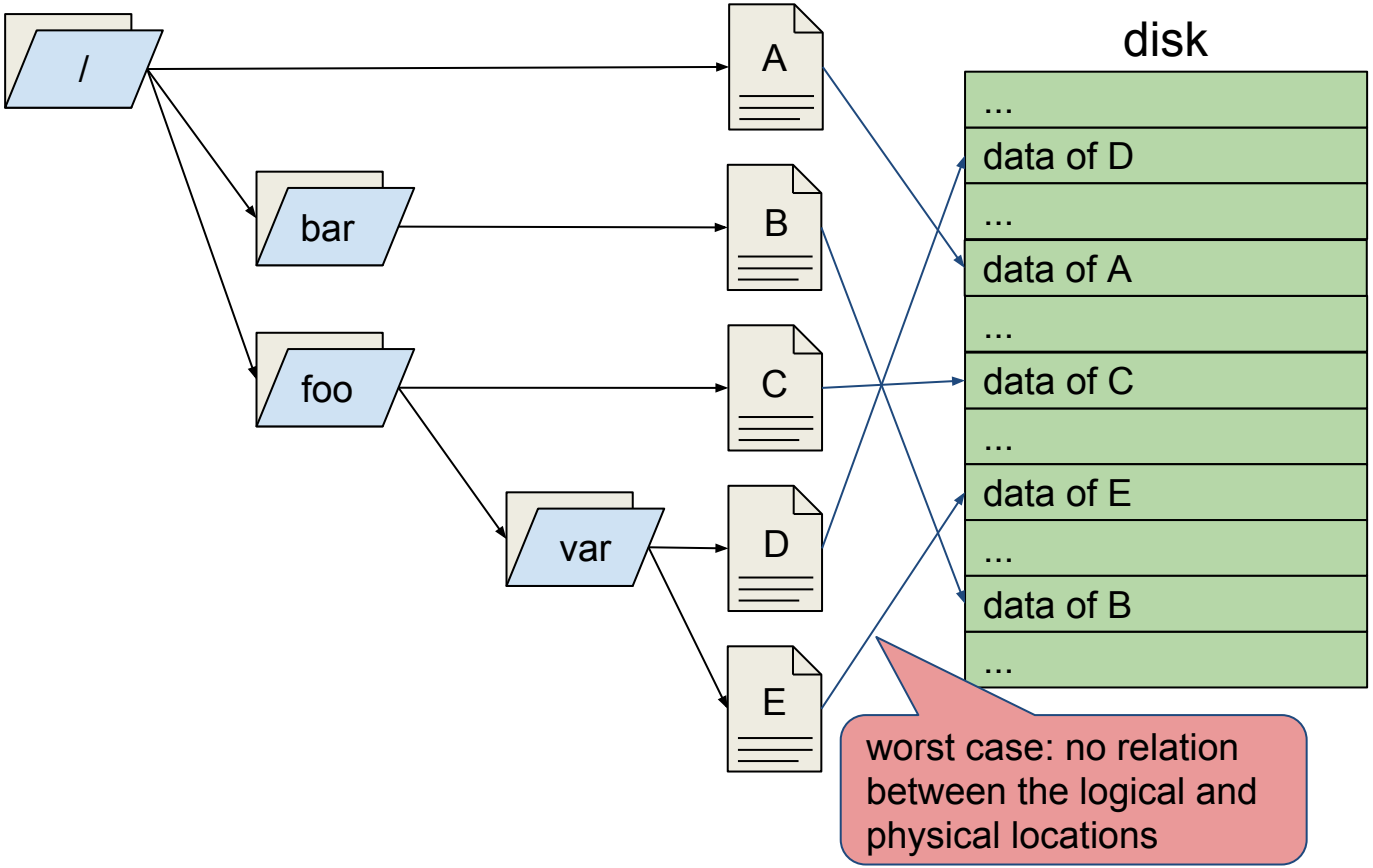
Yang Zhan, Alex Conway, Yizheng Jiao, Eric Knorr, Michael A. Bender, Martin Farach-Colton, William Jannen, Rob Johnson, Donald E. Porter, Jun Yuan

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL
RUTGERS
Stony Brook University
Williams
vmware

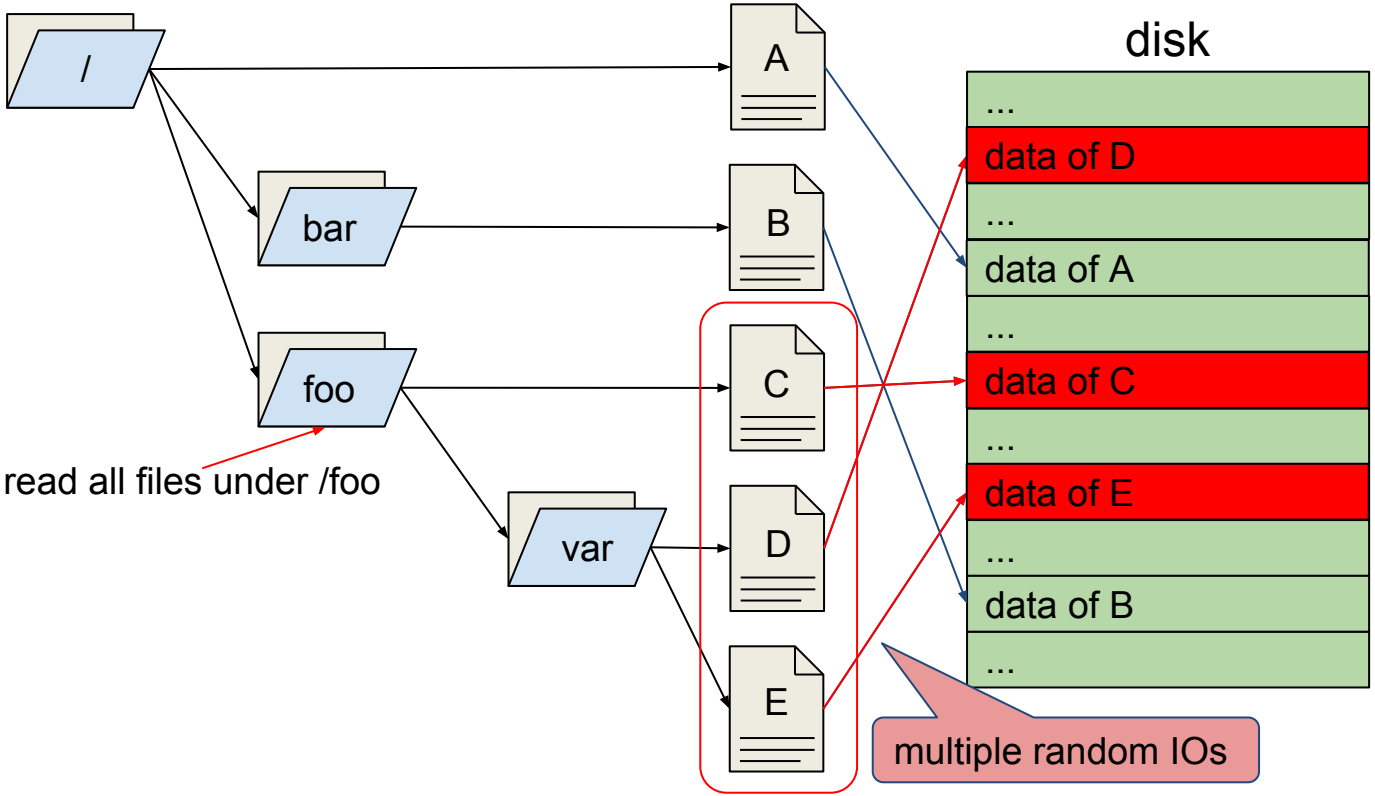
Talk Overview

- **What is full-path indexing and its benefits?**
  - **locality**
- What are the challenges?
  - renames
- How do we overcome them?
  - data structure techniques: tree surgery and lifting

Conventional file systems use inodes



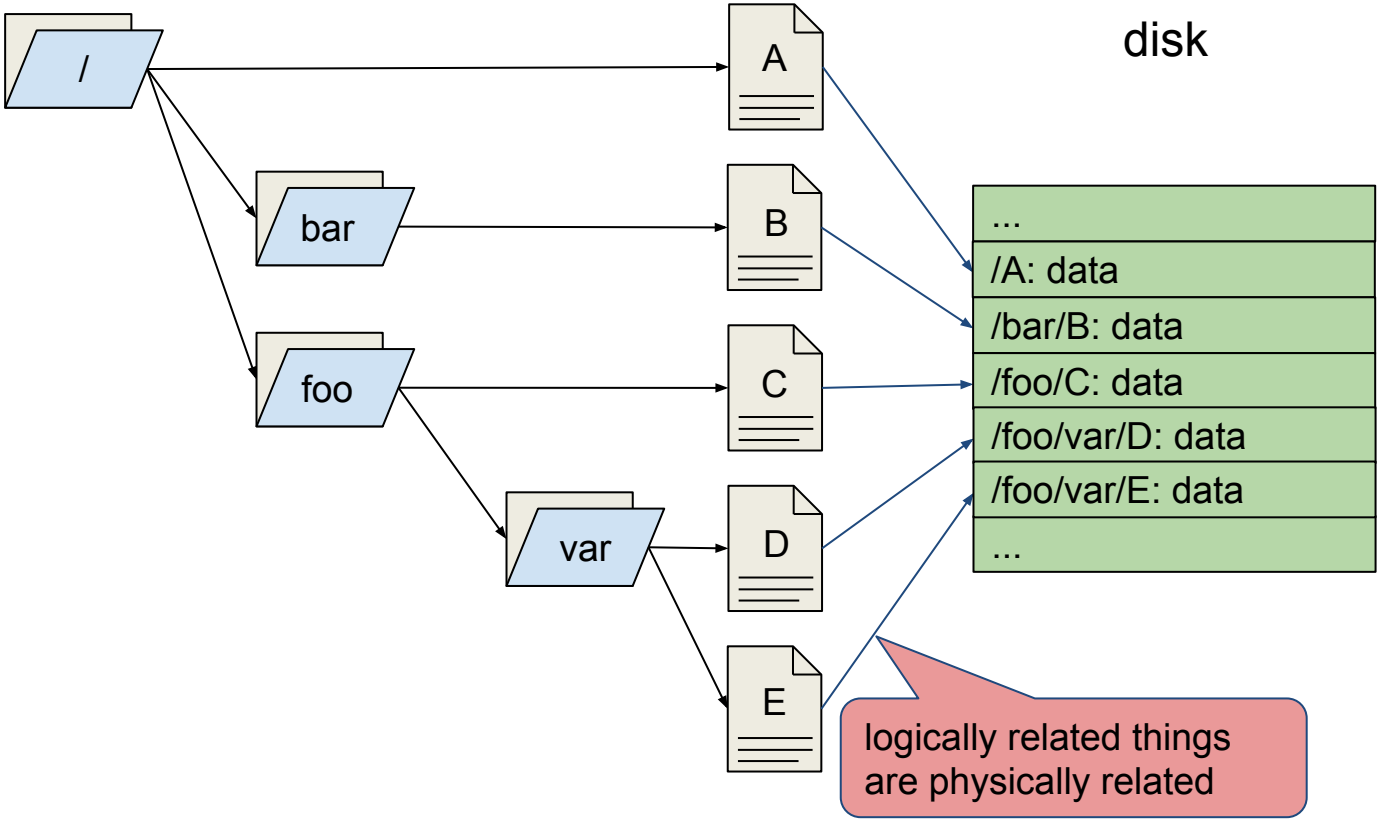
Inode file systems show no locality in the worst case



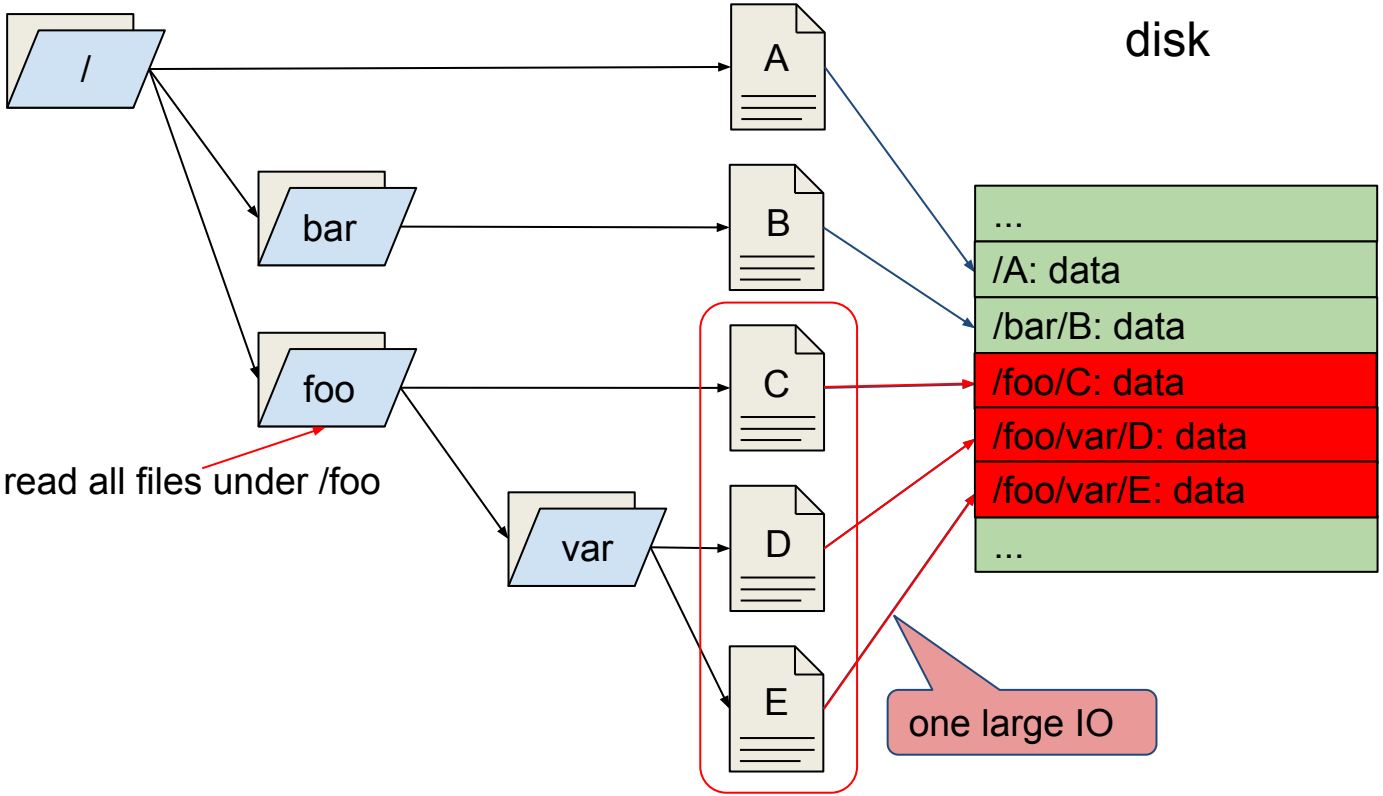
Full-path indexing file systems use full-paths

- Full-path indexing file systems index metadata and data in key-value stores using full-paths

Full-path indexing file systems use full-paths

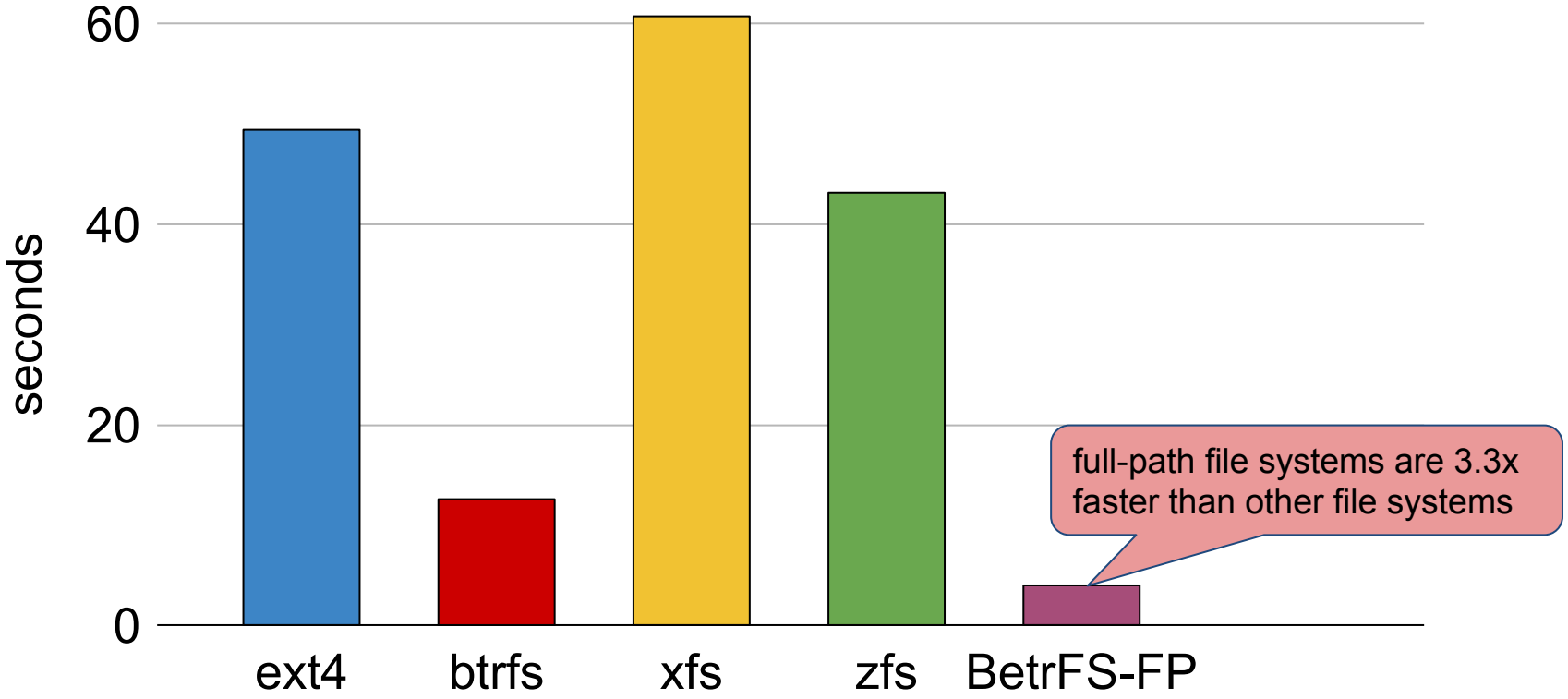


Full-path file systems ensure locality



Scans are fast in full-path file systems

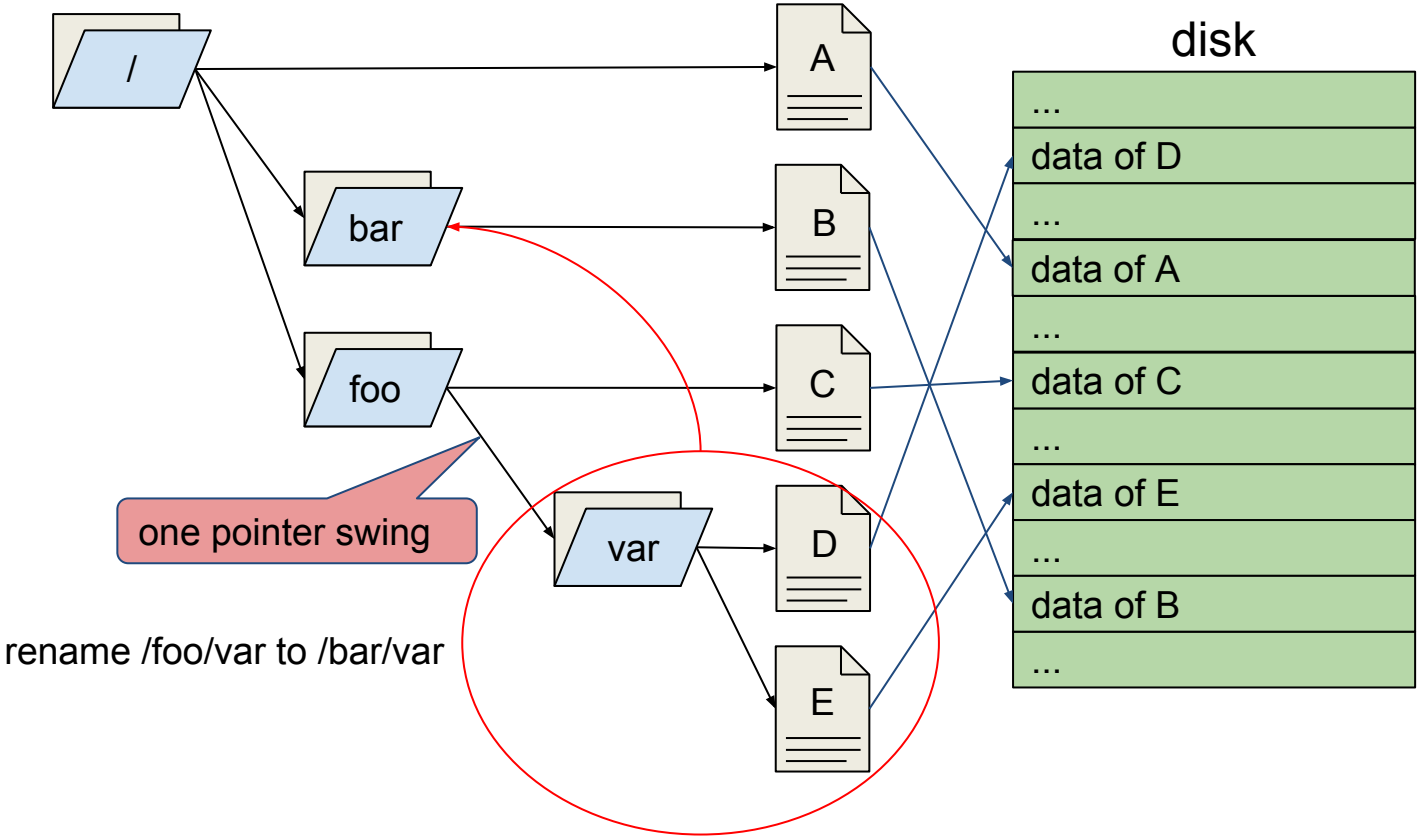
Time to grep the linux source directory (lower is better)



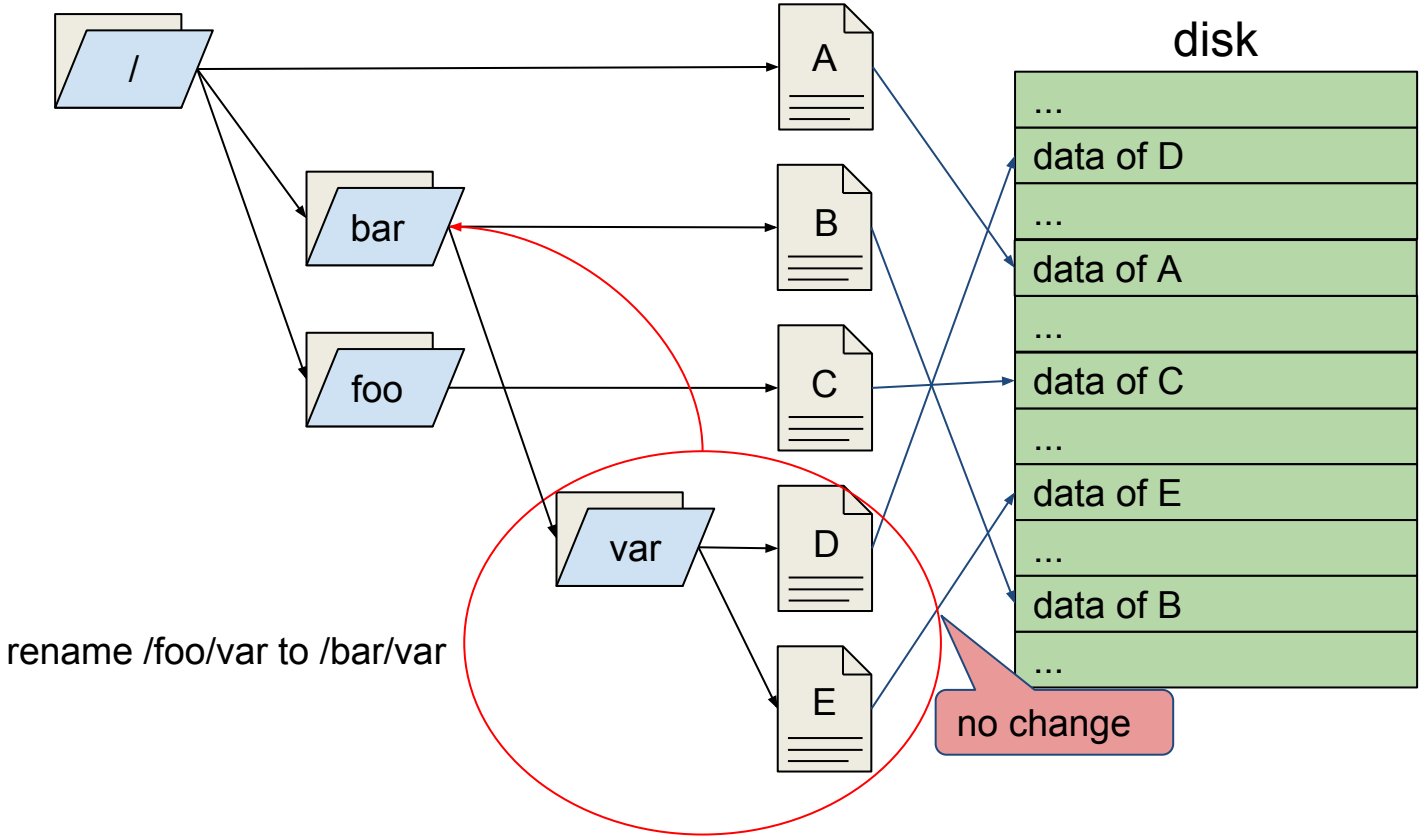
Talk Overview

- What is full-path indexing and its benefits?
  - locality
- **What are the challenges?**
  - **renames**
- How do we overcome them?
  - data structure techniques: tree surgery and lifting

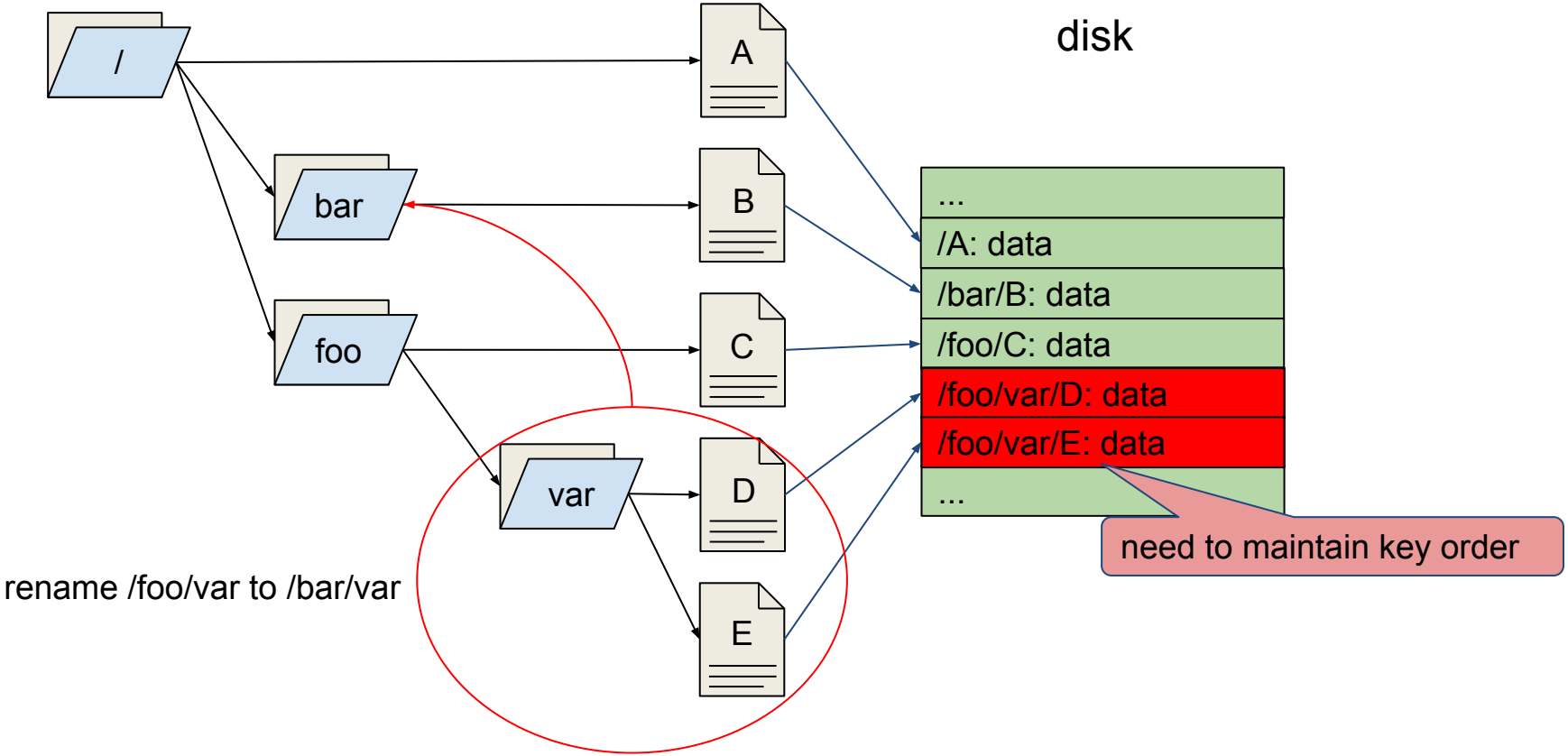
Renames are cheap in inode file systems



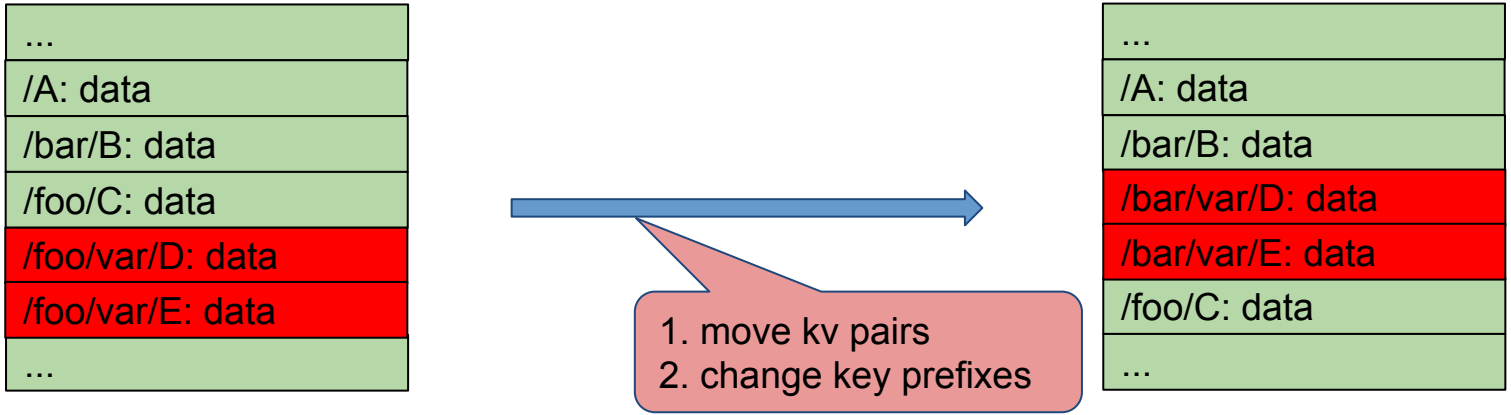
Renames are cheap in inode file systems



Renames seem expensive with full-path indexing

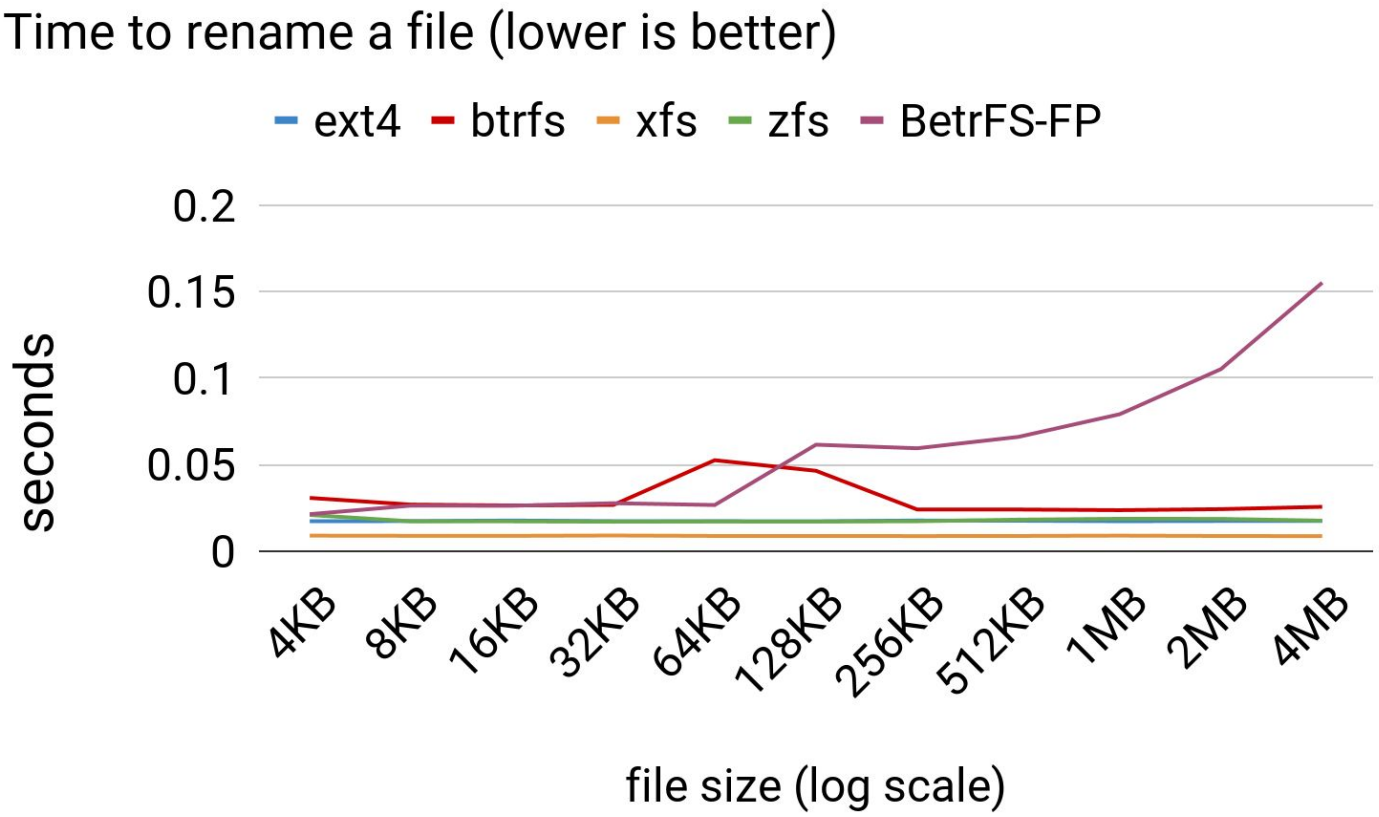


Renames seem expensive with full-path indexing

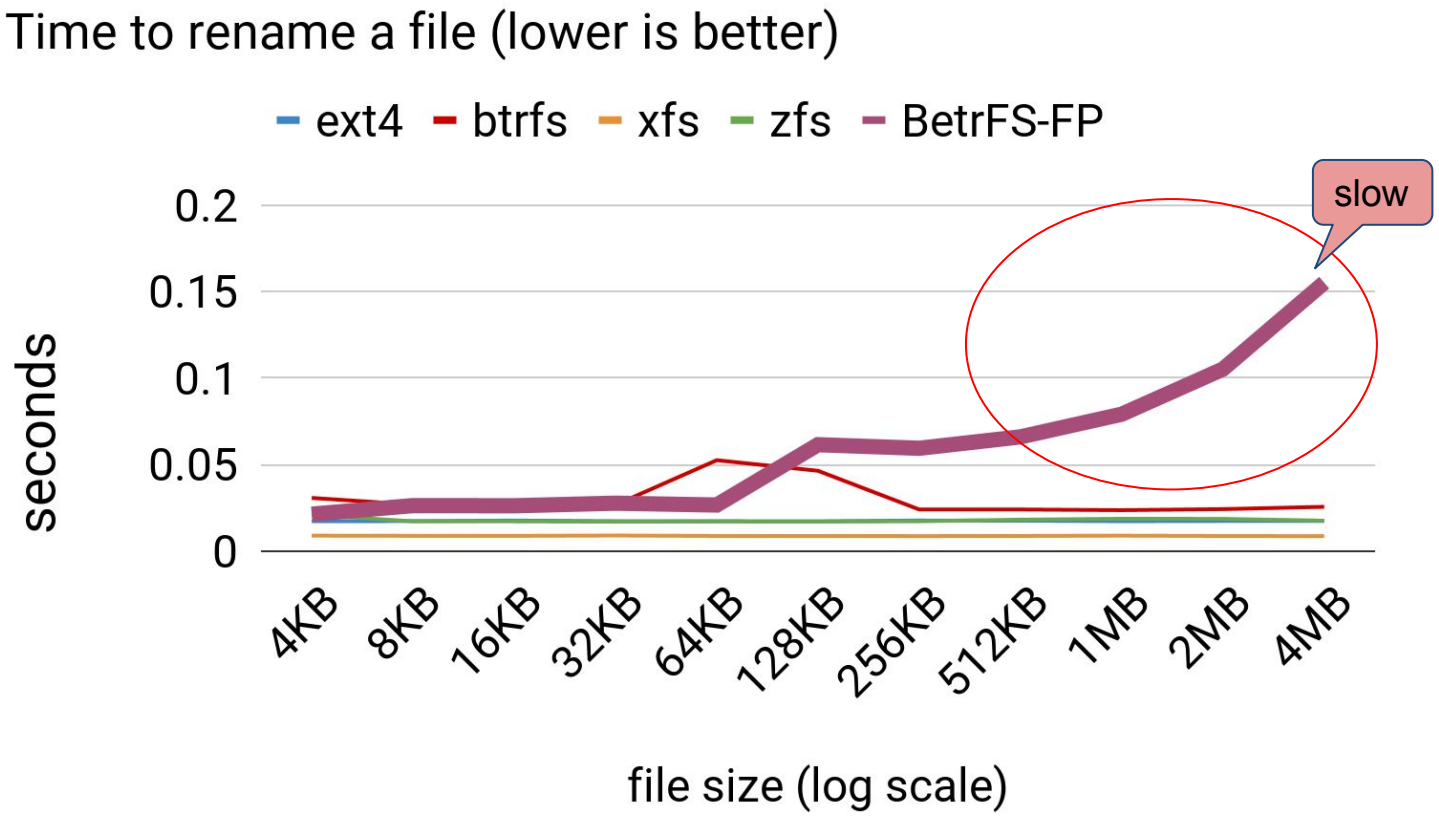


Expensive when rename size is large

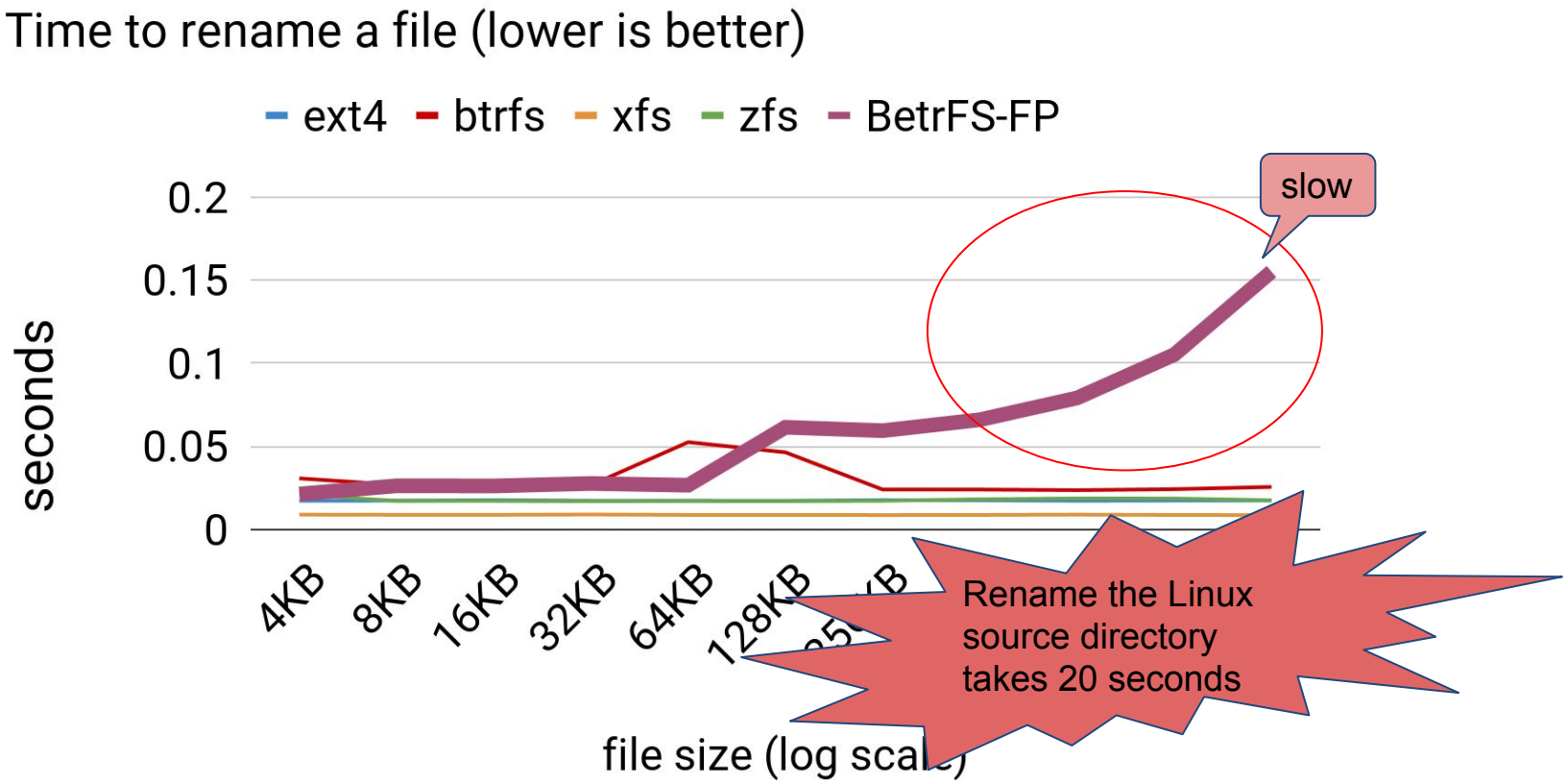
Renaming big files are slow in full-path file systems



Renaming big files are slow in full-path file systems



Renaming big files are slow in full-path file systems



Inode vs. Full-path indexing

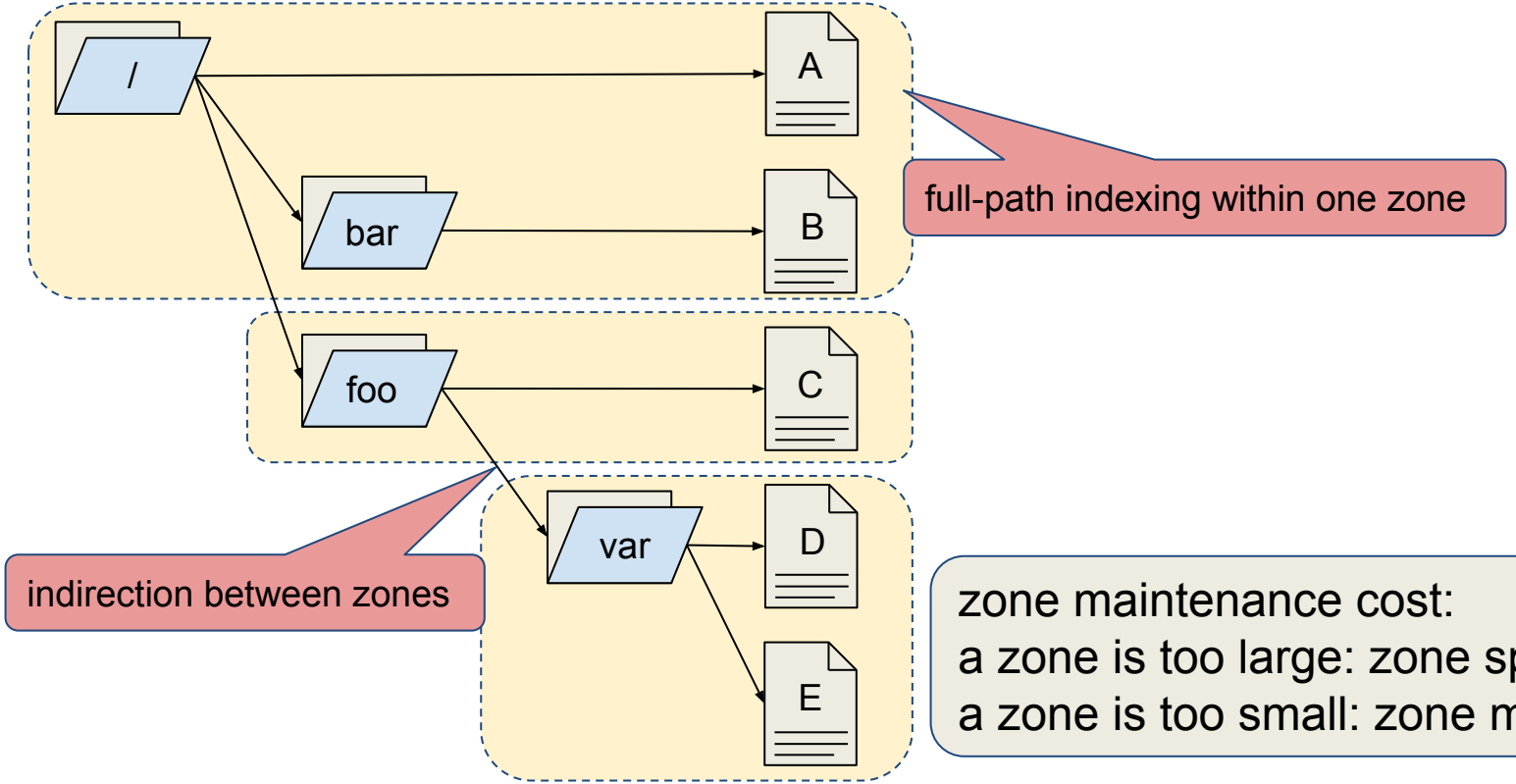
	rename	locality
inode file systems	😁	☹️
full-path file systems	☹️	😁

We want to get decent renames with good locality

Zoning tries to solve the rename problem

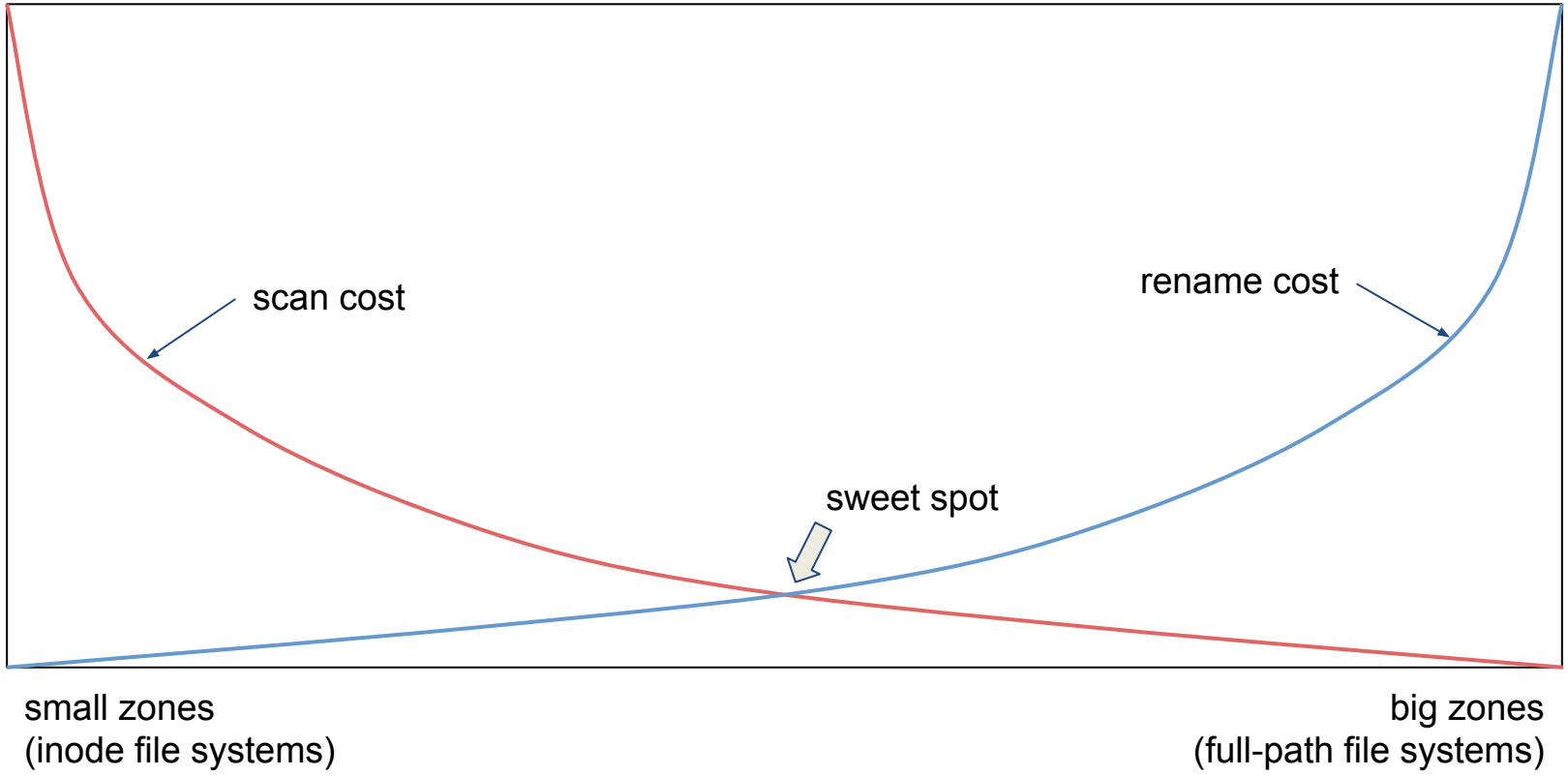
- In FAST 2016, zoning was introduced to BetrFS
- Zoning tries to get both locality and fast renames

Zoning Example



zone maintenance cost:
a zone is too large: zone split
a zone is too small: zone merge

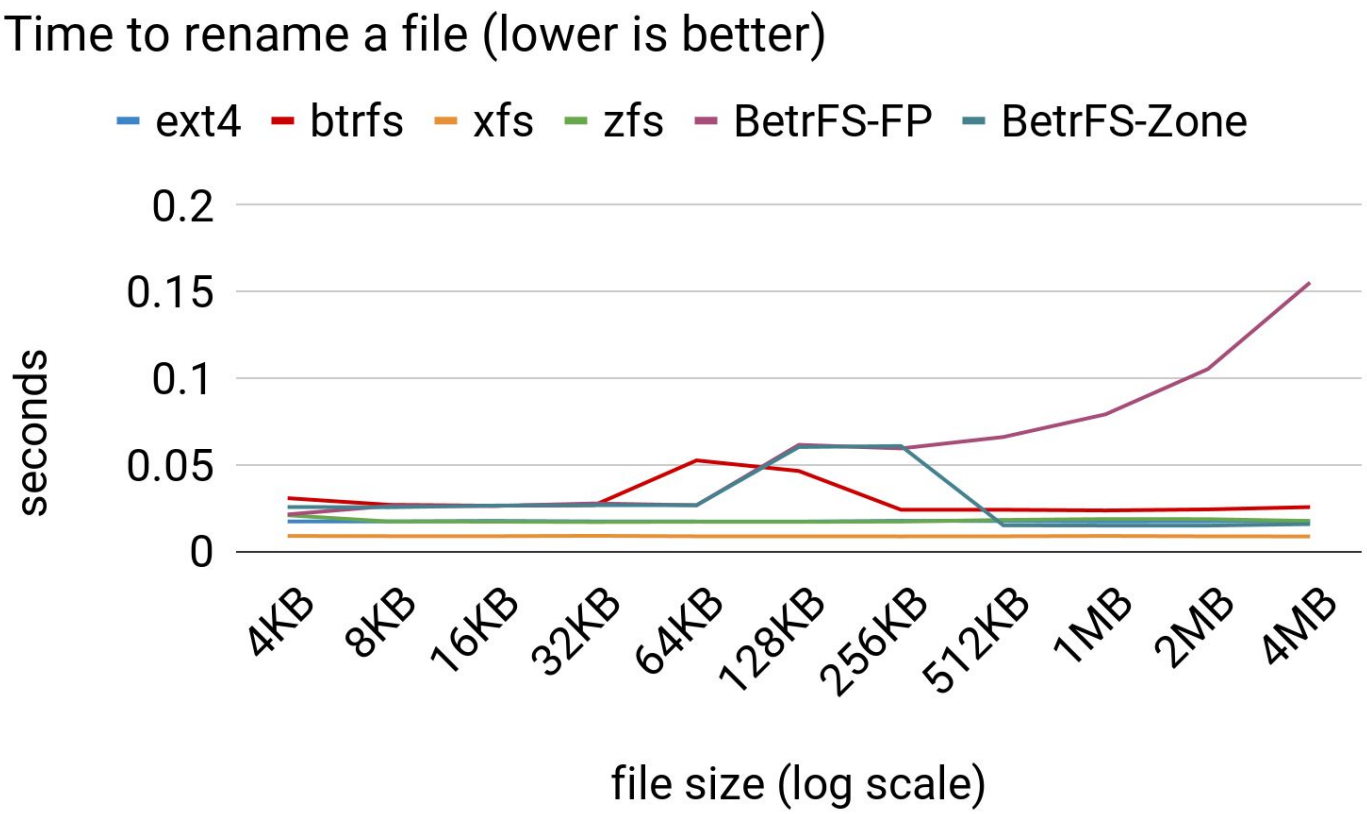
Zoning tries to achieve both fast renames and locality



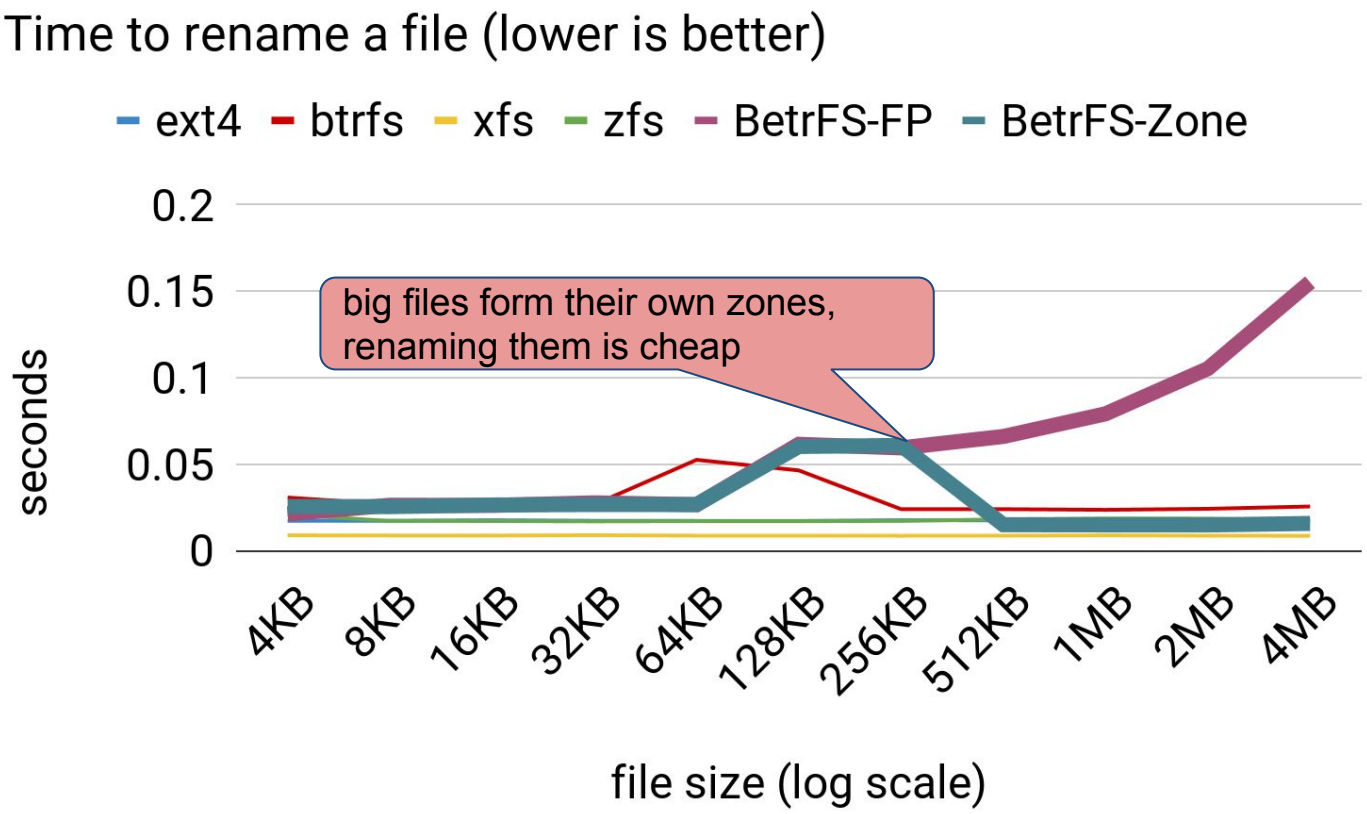
Zoning performance

	rename	locality	other operations
zoning	🤔	🤔	🤔

Zoning achieves cheap renames



Zoning achieves cheap renames

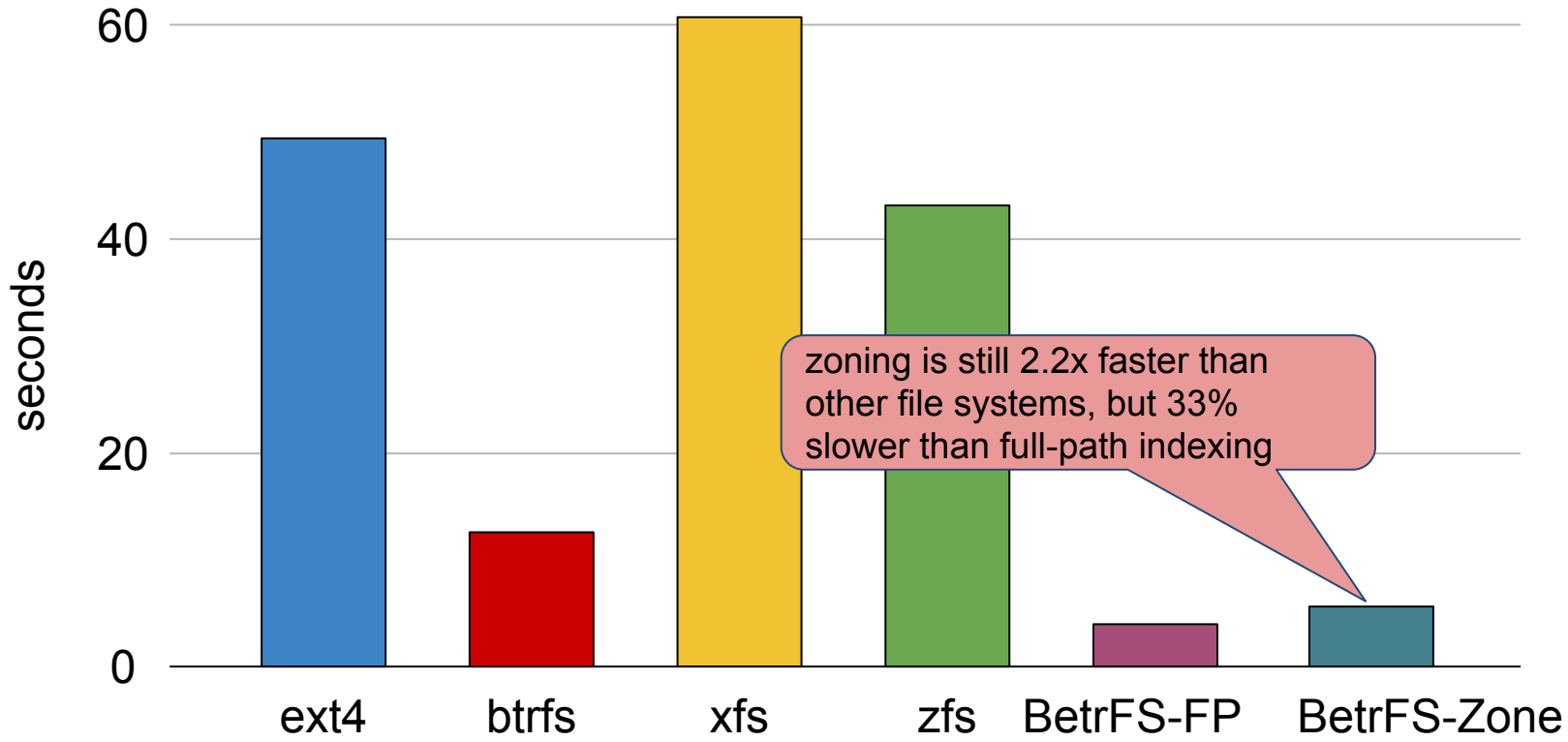


Zoning performance

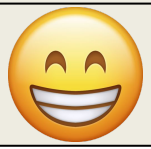
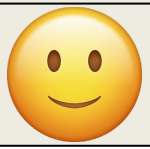
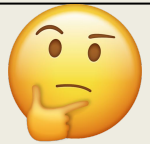
	rename	locality	other operations
zoning	😁	🤔	🤔

Zoning has relatively good locality

Time to grep the linux source directory (lower is better)

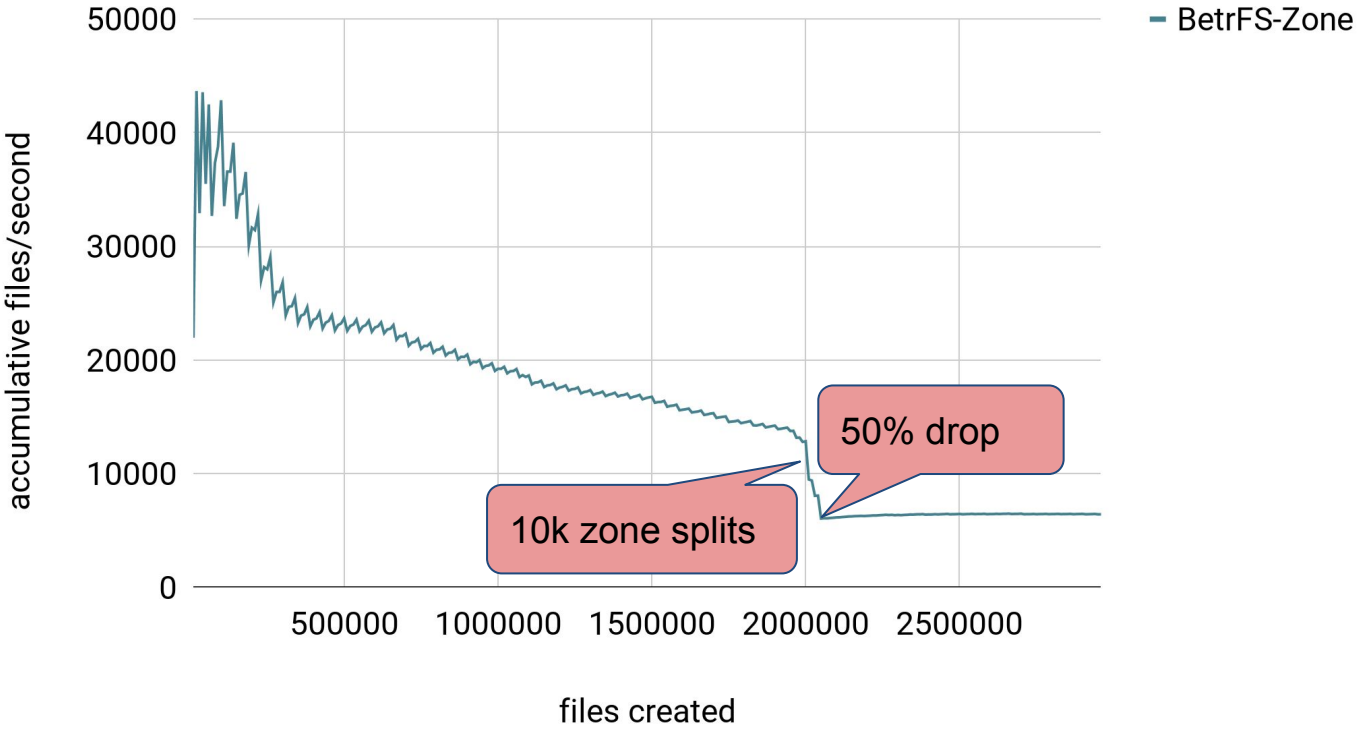


Zoning performance

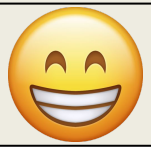
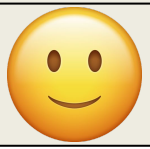
	rename	locality	other operations
zoning			

Zone maintenance can be expensive

Tokubench: create 3 million 200-byte files in a balanced directory tree (higher is better)



Zoning performance

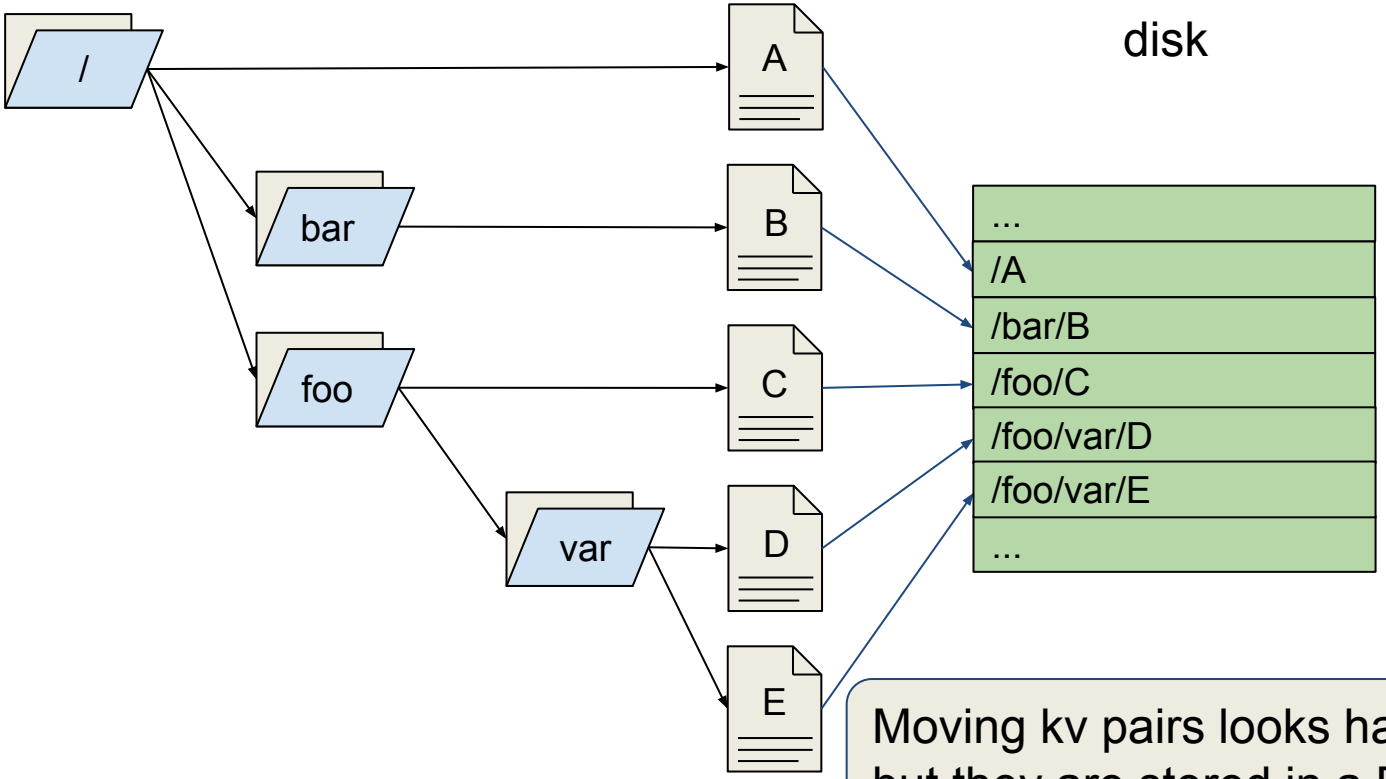
	rename	locality	other operations
zoning			

Zoning is not the answer

Talk Overview

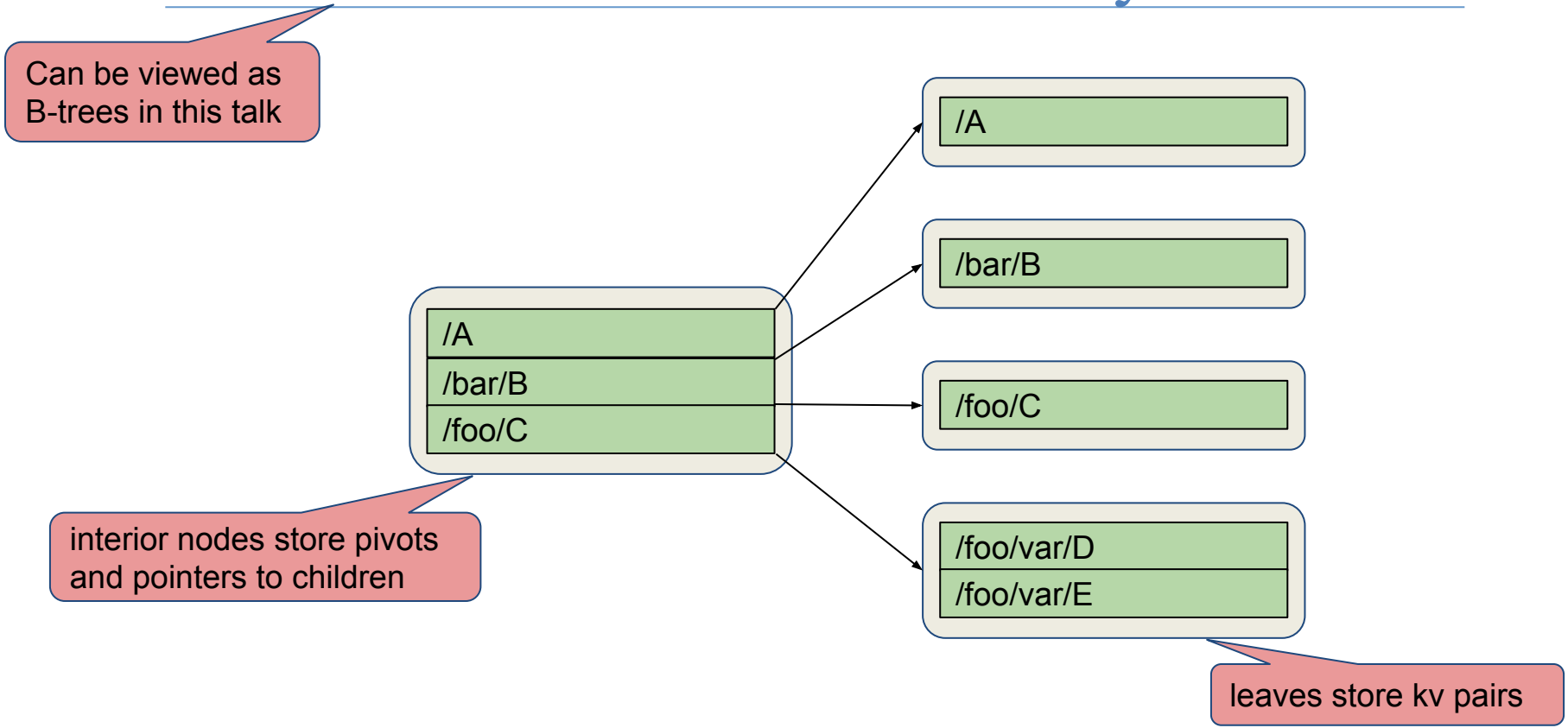
- What is full-path indexing and its benefits?
  - locality
- What are the challenges?
  - renames
- **How do we overcome them?**
  - **data structure techniques: tree surgery and lifting**

Moving is expensive in an array

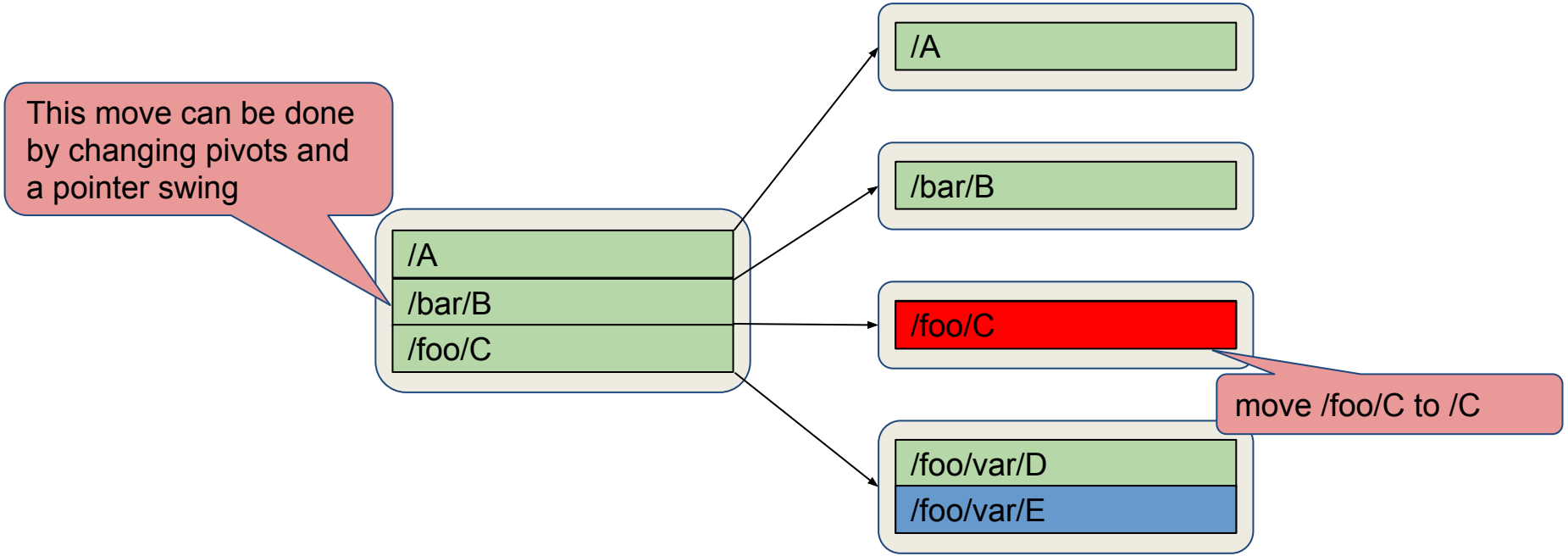


Moving kv pairs looks hard in an array, but they are stored in a B^{ε}-tree in BetrFS

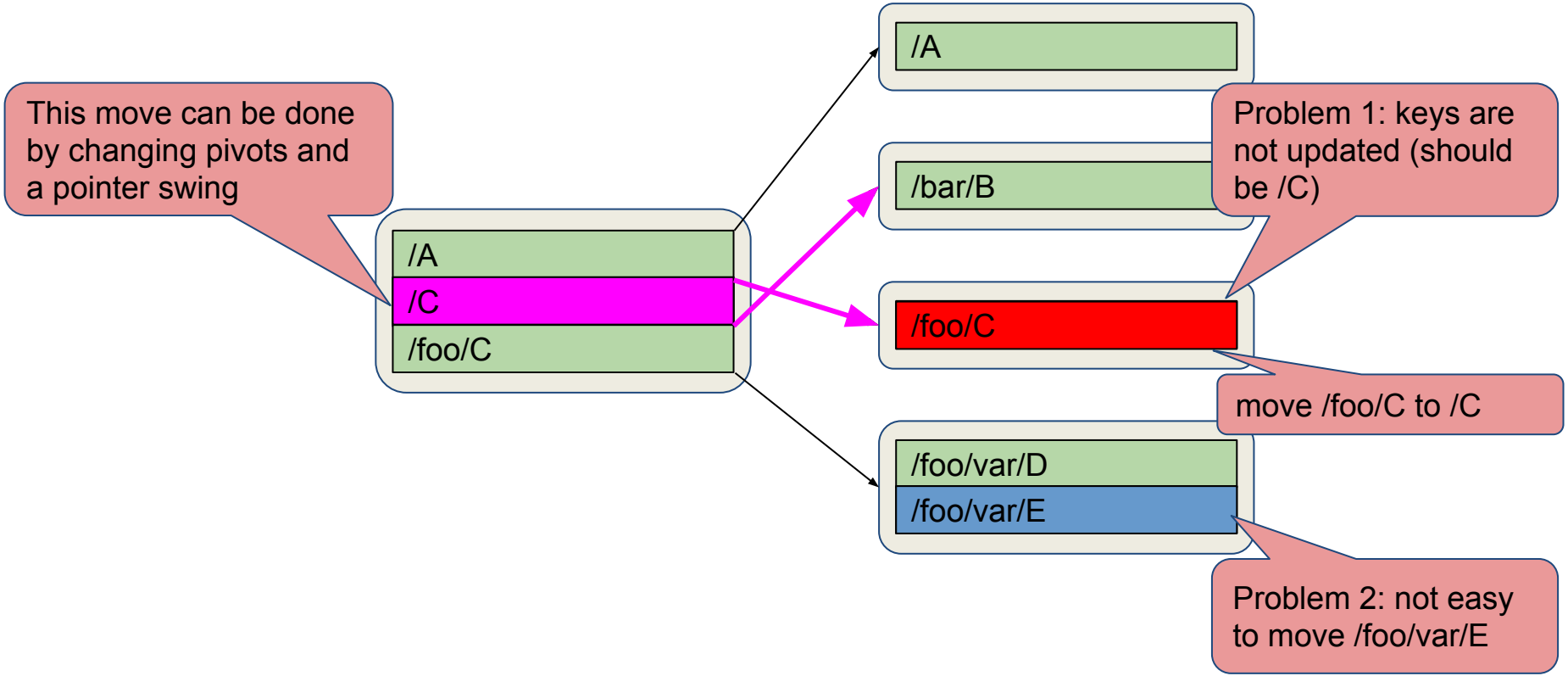
B^{ε}-trees sometimes allow easy moves



B^{ε}-trees sometimes allow easy moves



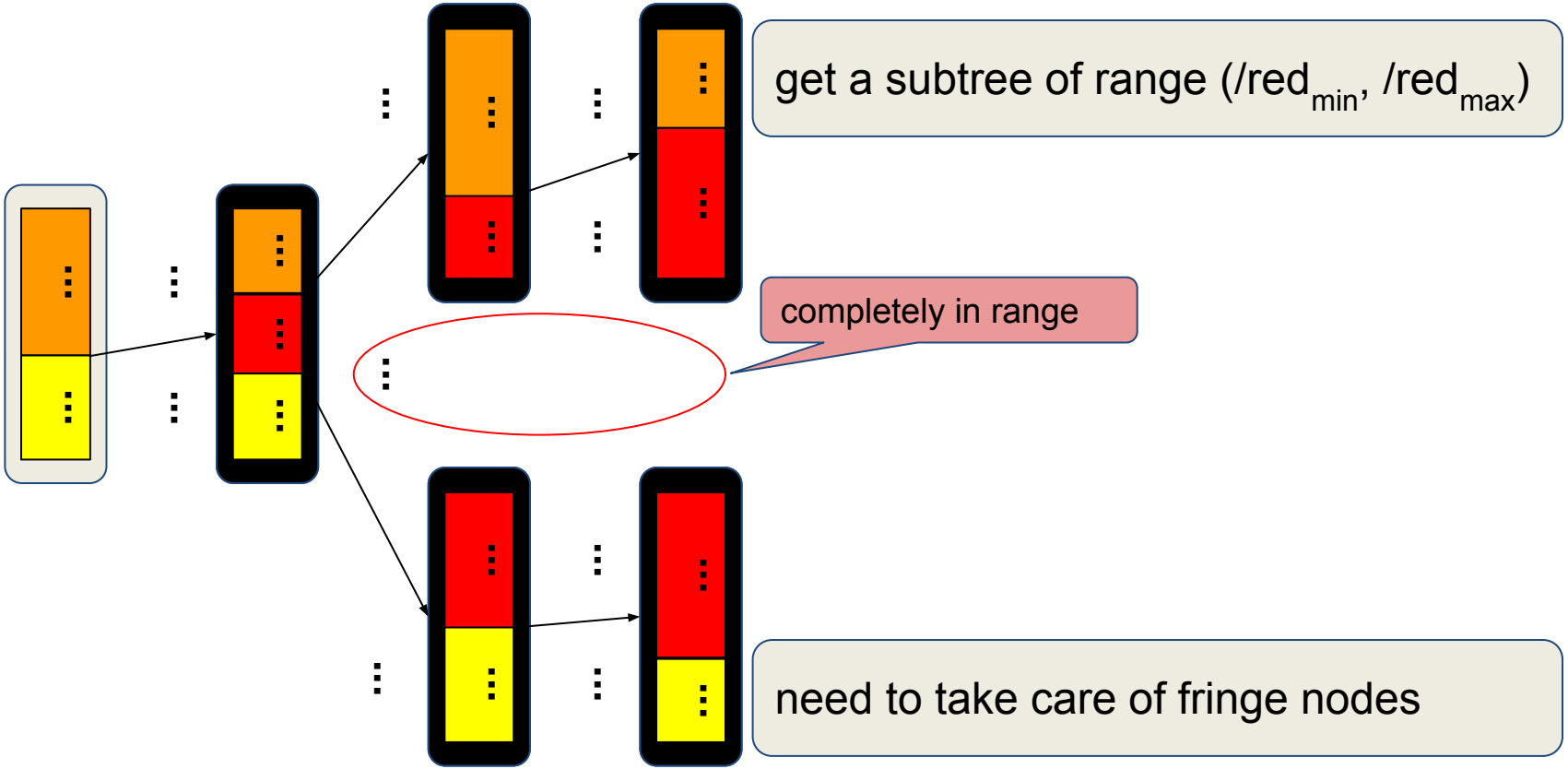
B^{ε}-trees sometimes allow easy moves



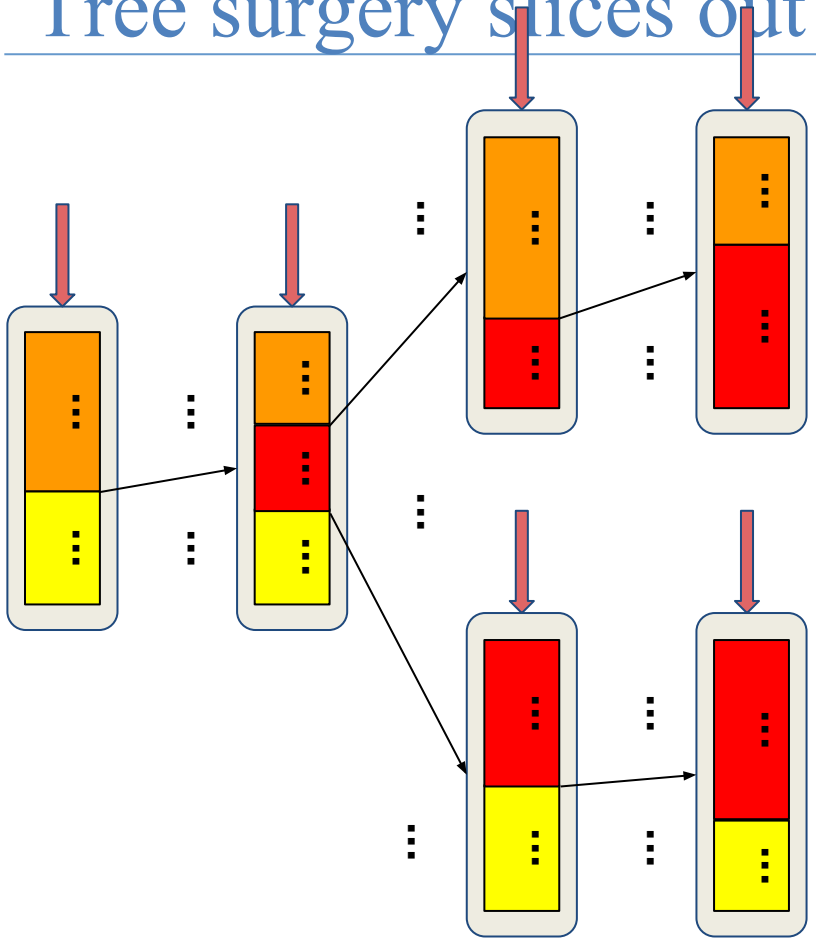
A rename can be done by moving a subtree

- Two problems:
  - need to get an isolated subtree
    - **tree surgery** in $O(B^{\varepsilon}$-tree height) IOs
  - need to update keys
    - **lifting**, no additional IO cost
- The whole solution is called **range-rename**

Tree surgery slices out an isolated subtree



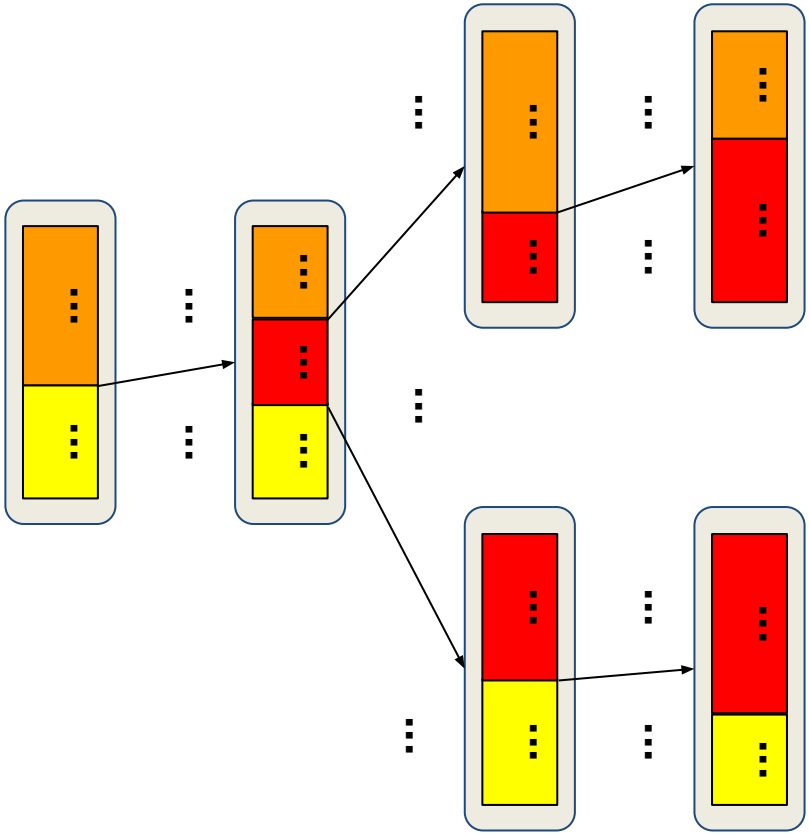
Tree surgery slices out an isolated subtree



get a subtree of range (/red$_{min}$, /red$_{max}$)

1. walk down the tree with min and max key

Tree surgery slices out an isolated subtree

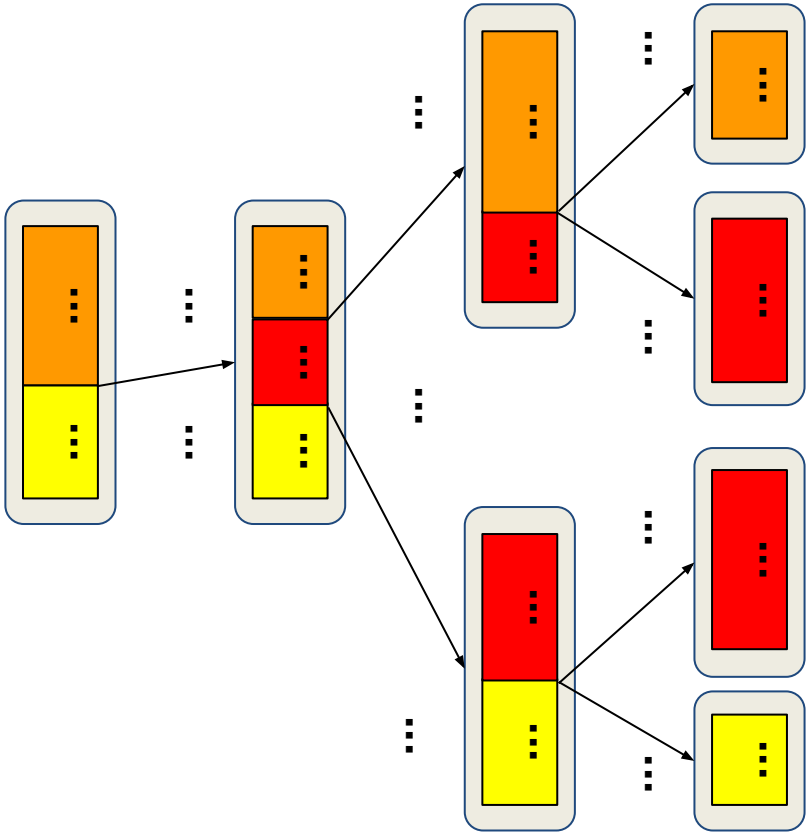


get a subtree of range (/red$_{min}$, /red$_{max}$)

1. walk down the tree with min and max key

2. start node splits from leaves

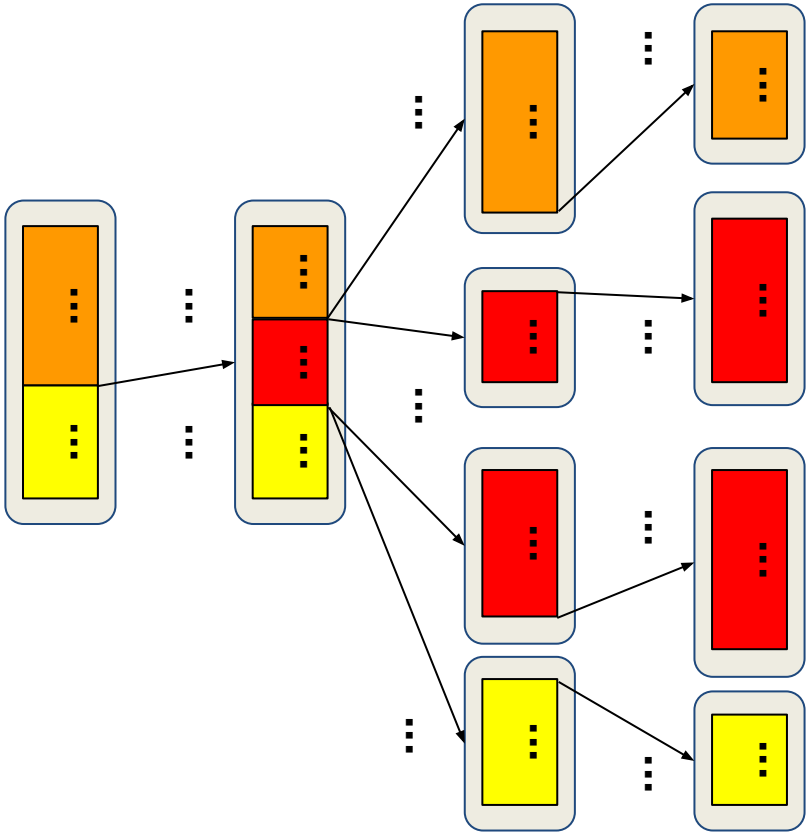
Tree surgery slices out an isolated subtree



get a subtree of range (/red$_{min}$, /red$_{max}$)

1. walk down the tree with min and max key
2. start node splits from leaves
3. keep splitting until two splits converge

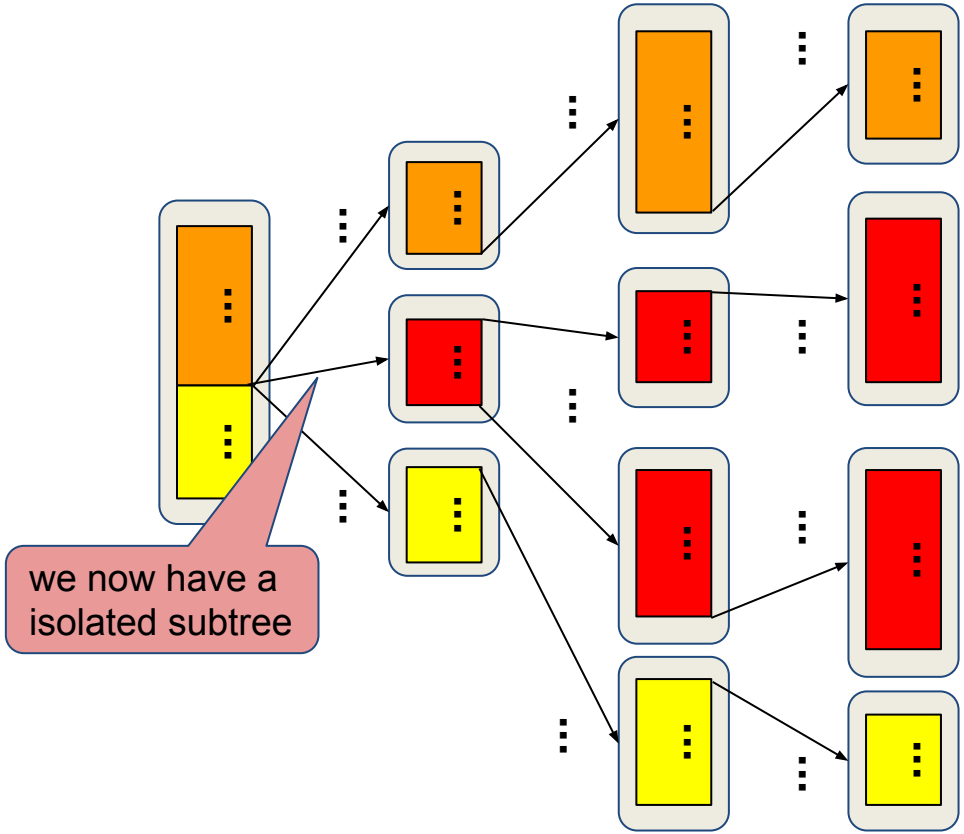
Tree surgery slices out an isolated subtree



get a subtree of range (/red$_{min}$, /red$_{max}$)

1. walk down the tree with min and max key
2. start node splits from leaves
3. keep splitting until two splits converge

Tree surgery slices out an isolated subtree



get a subtree of range (/red$_{min}$, /red$_{max}$)

1. walk down the tree with min and max key

2. start node splits from leaves

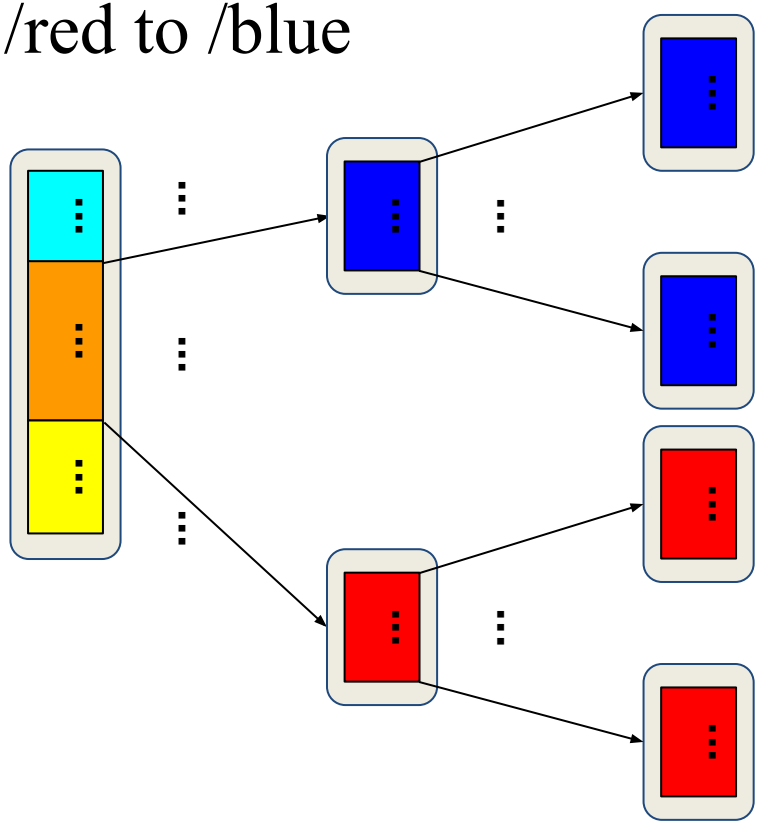
3. keep splitting until two splits converge

Tree surgery also slices at the destination

- Reasons:
  - to setup pivots for the source tree
  - POSIX allows renames to overwrite files

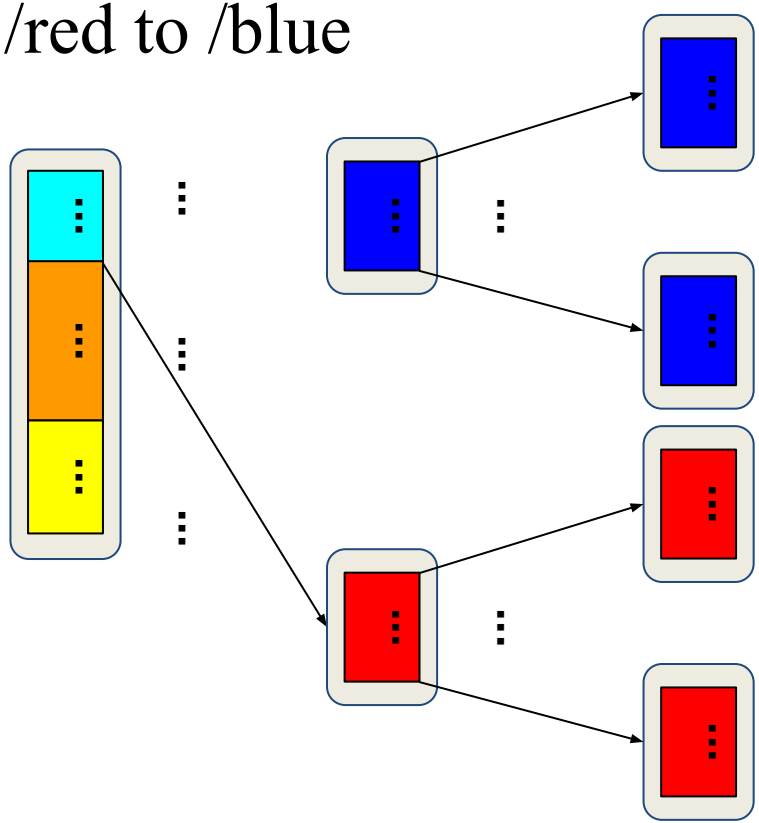
Tree surgery finishes with a pointer swing

- rename /red to /blue

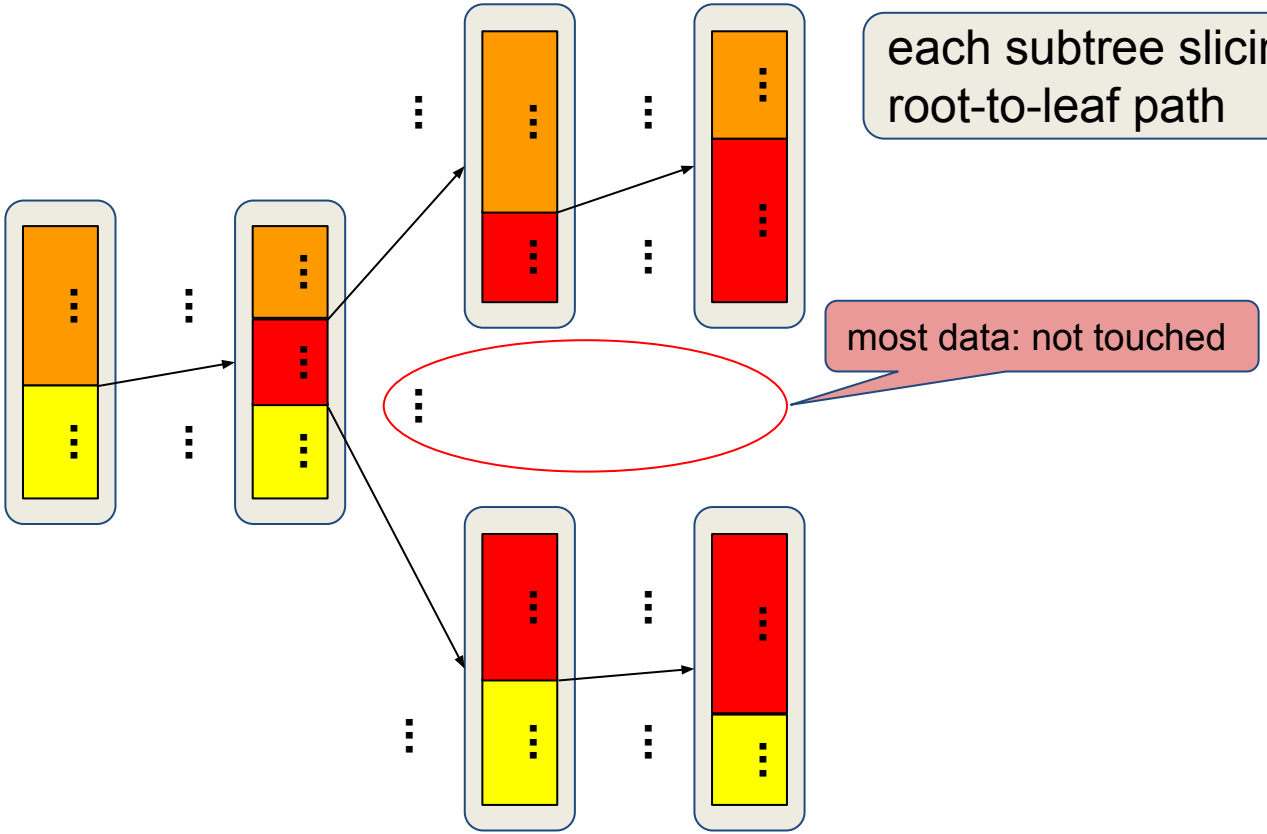


Tree surgery finishes with a pointer swing

- rename /red to /blue



Tree surgery completes in O(B^{ε}-tree height) IOs

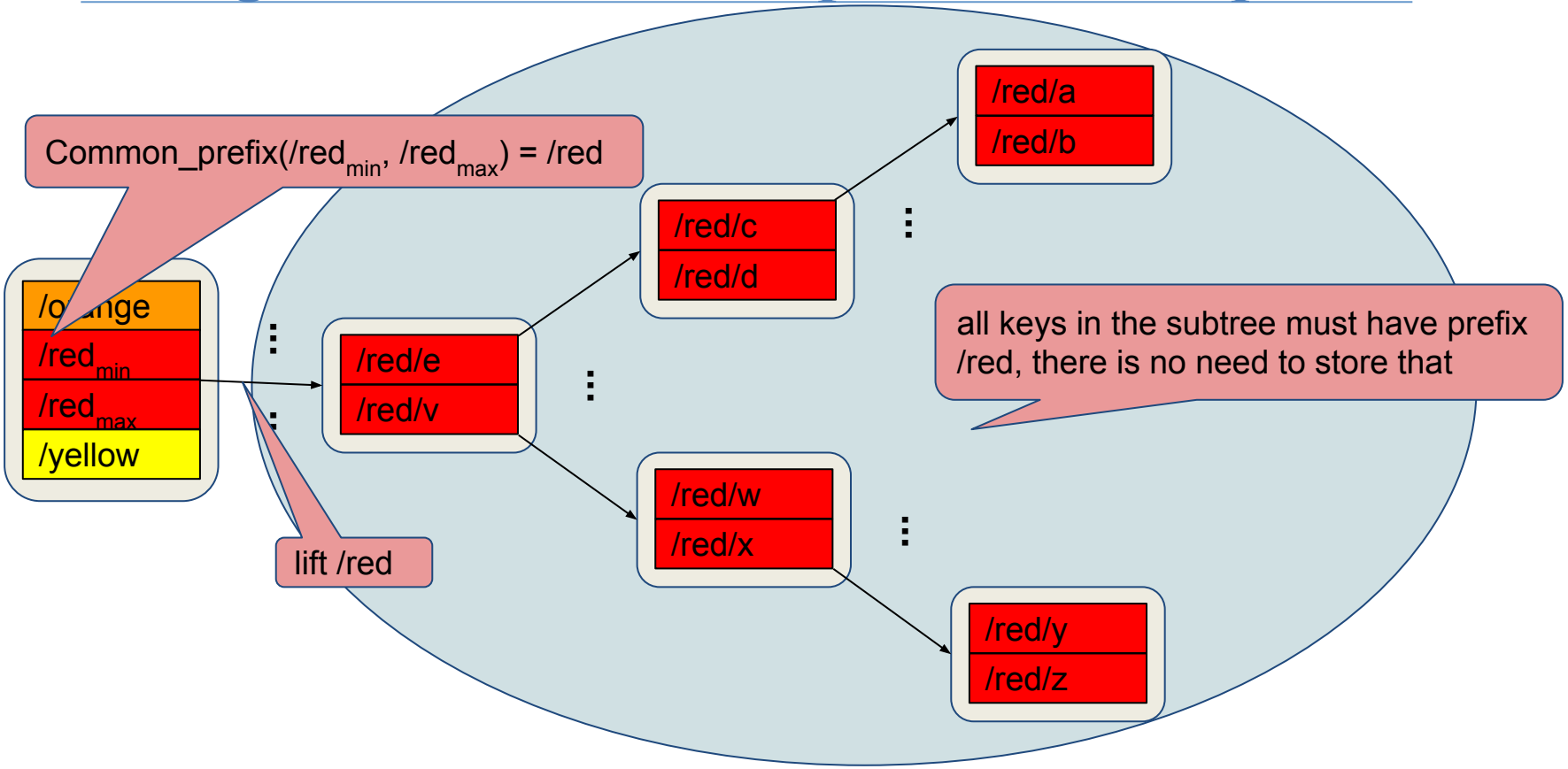


Updating all keys in the subtree is expensive

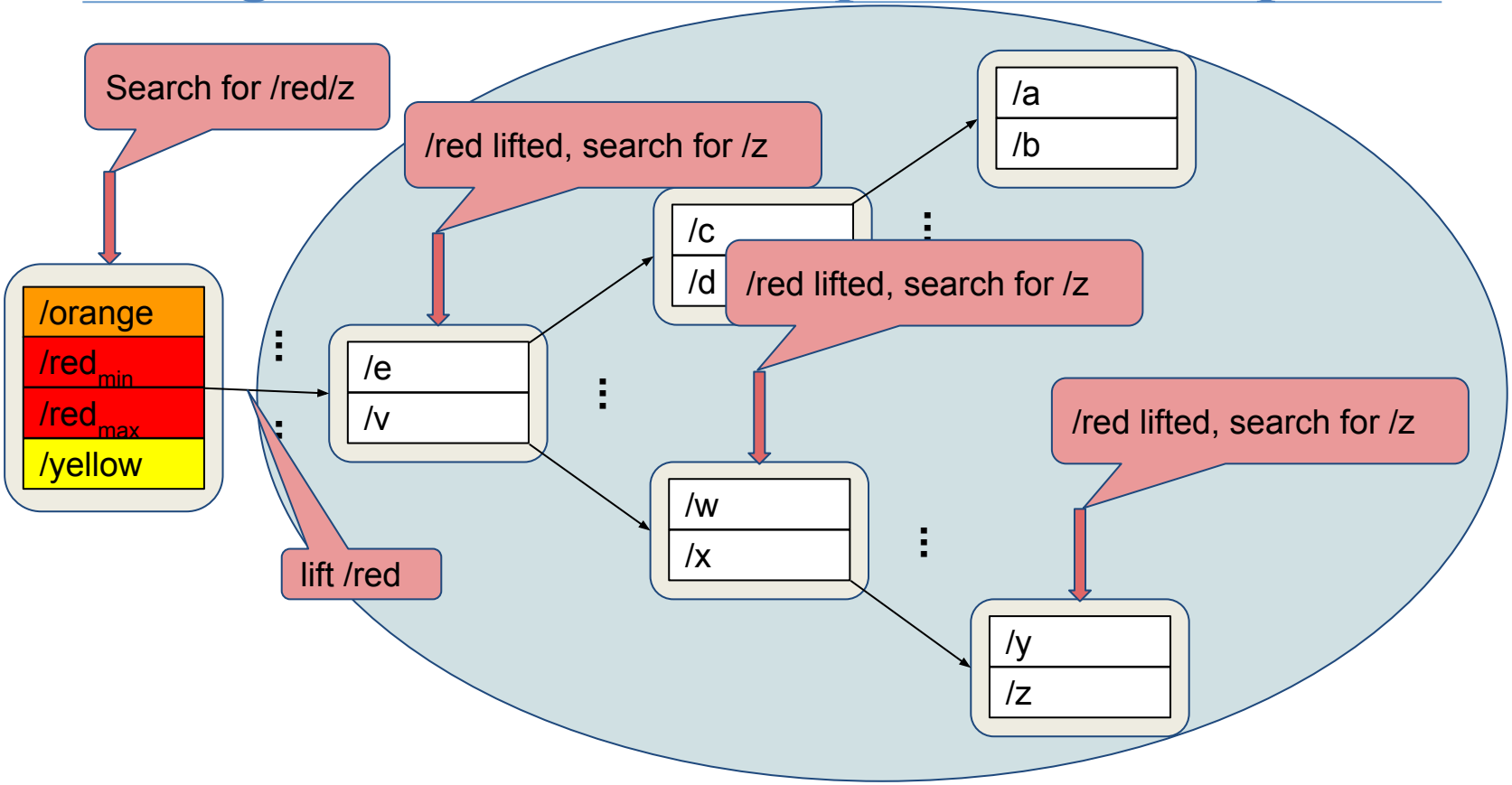
- Revert the IO cost to O(subtree size)

- Solution: **lifting** to convert B^{ε}-trees to lifted B^{ε}-trees
  - prefix updates are free

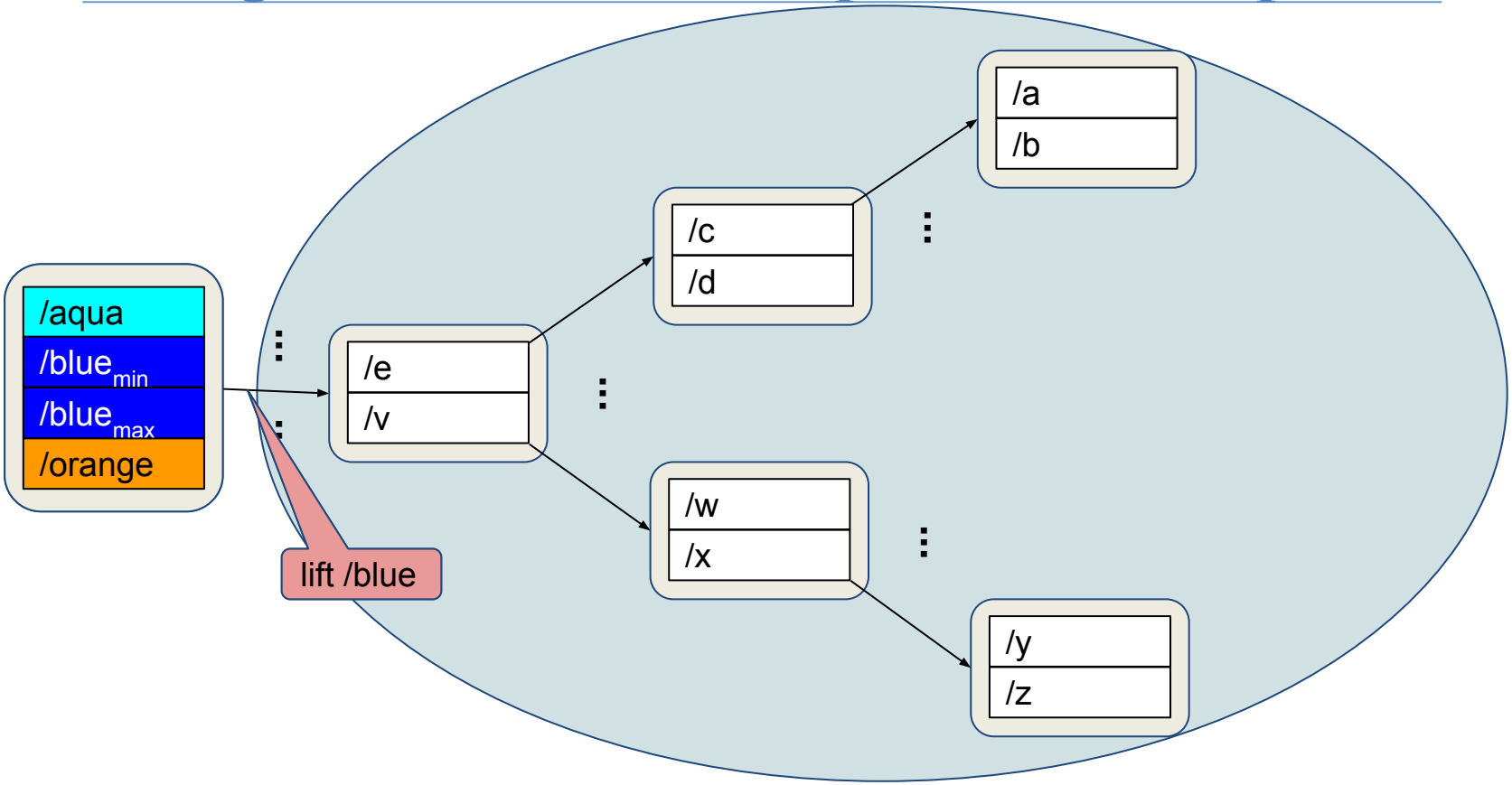
Lifting lifts the common prefix of two pivots



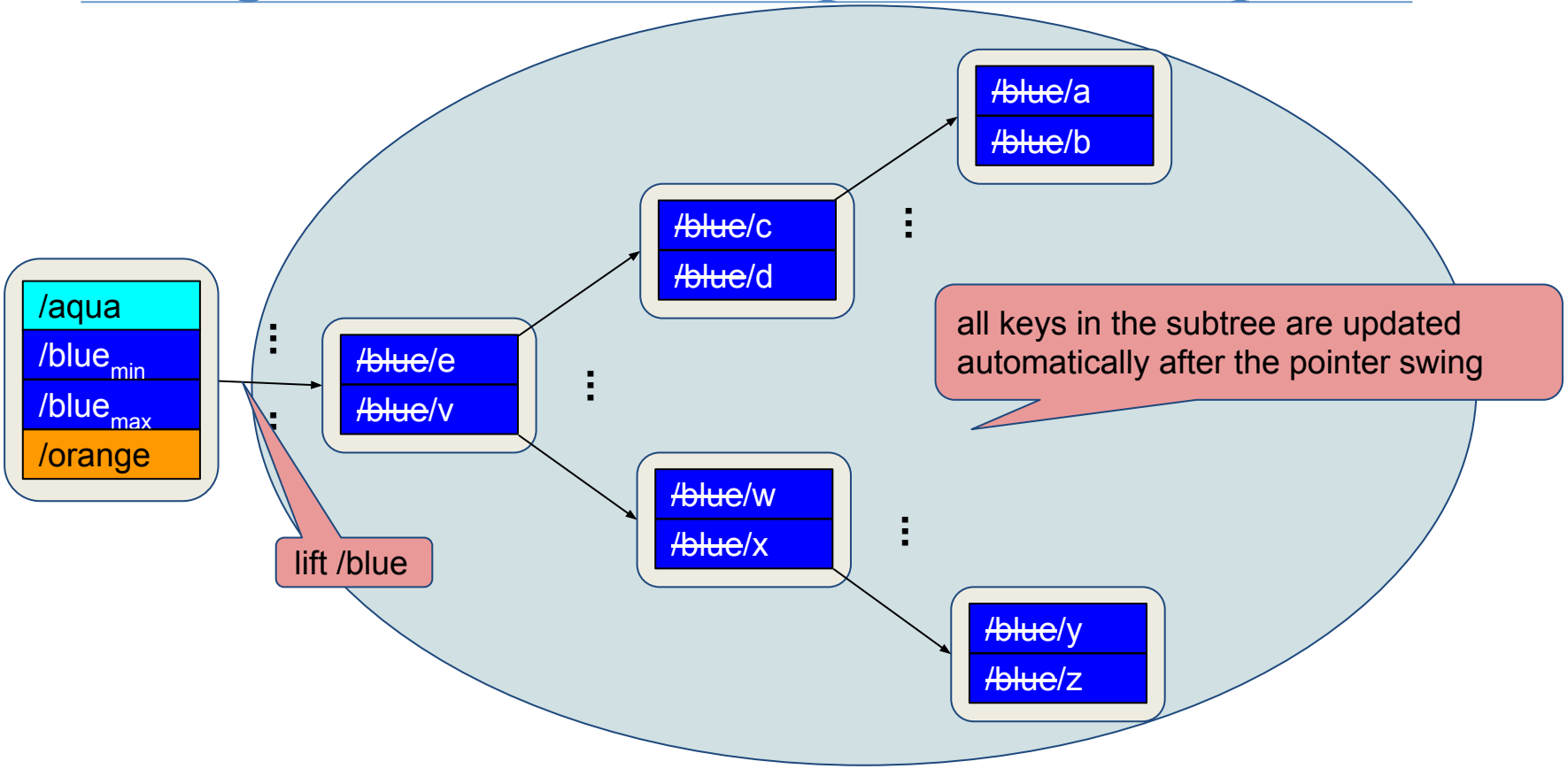
Lifting lifts the common prefix of two pivots



Lifting lifts the common prefix of two pivots



Lifting lifts the common prefix of two pivots



Lifting does not introduce additional IOs

- Lifting happens at all times
- Cost of other operations:
  - collect lifted parts along the root-to-leaf path
  - no additional IO
- Cost of maintaining key lifting
  - key lifting can only change in node splits/merges
  - no additional IO

Range-rename completes in O(B^ε-tree height) IOs

- Range-rename performs tree surgery
  - O(B^ε-tree height) IOs
- Key/value pairs are stored in lifted B^ε-trees
  - keys are updated after tree surgery without cost

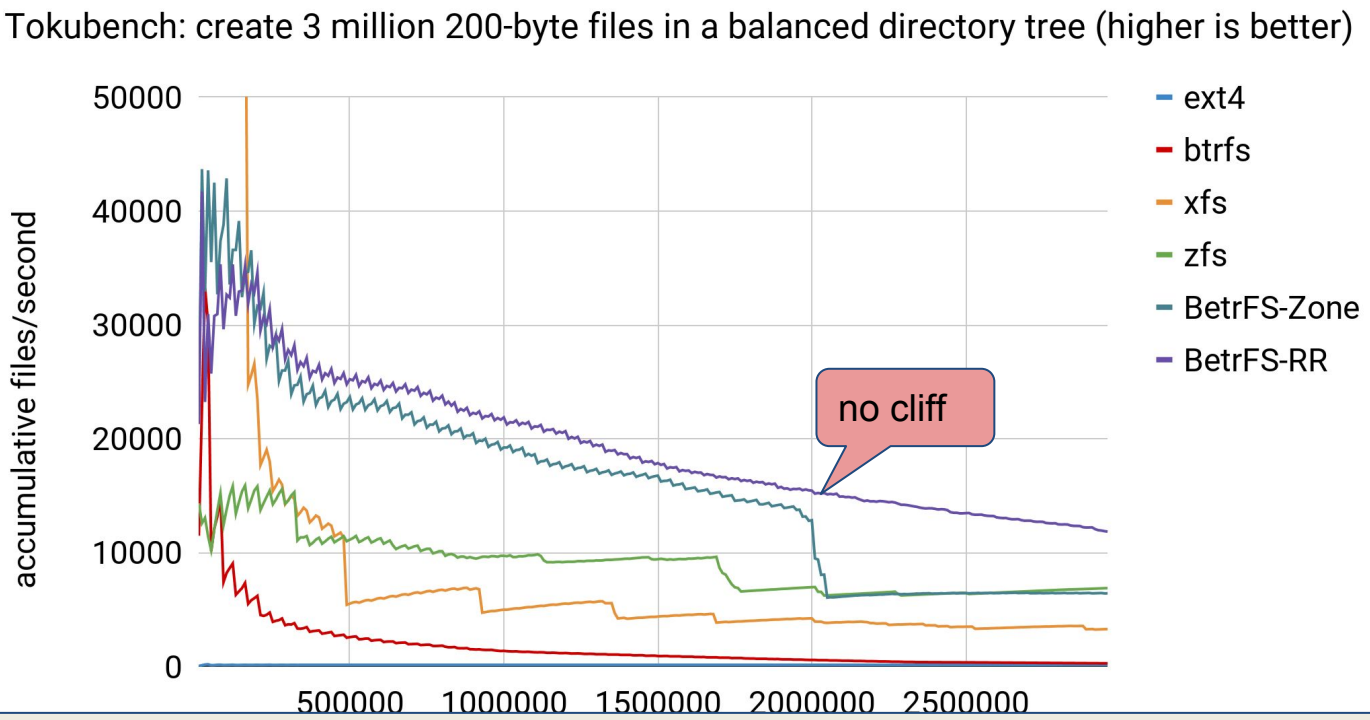
Evaluation

	other operations	rename	applications
range-rename	🤔	🤔	🤔

Experimental Setup

- Dell optilex destop
  - 4-core 3.4 GHz i7, 4 GB RAM
  - 7200 RPM 500 GB Seagate Barrcuda

Tokubench



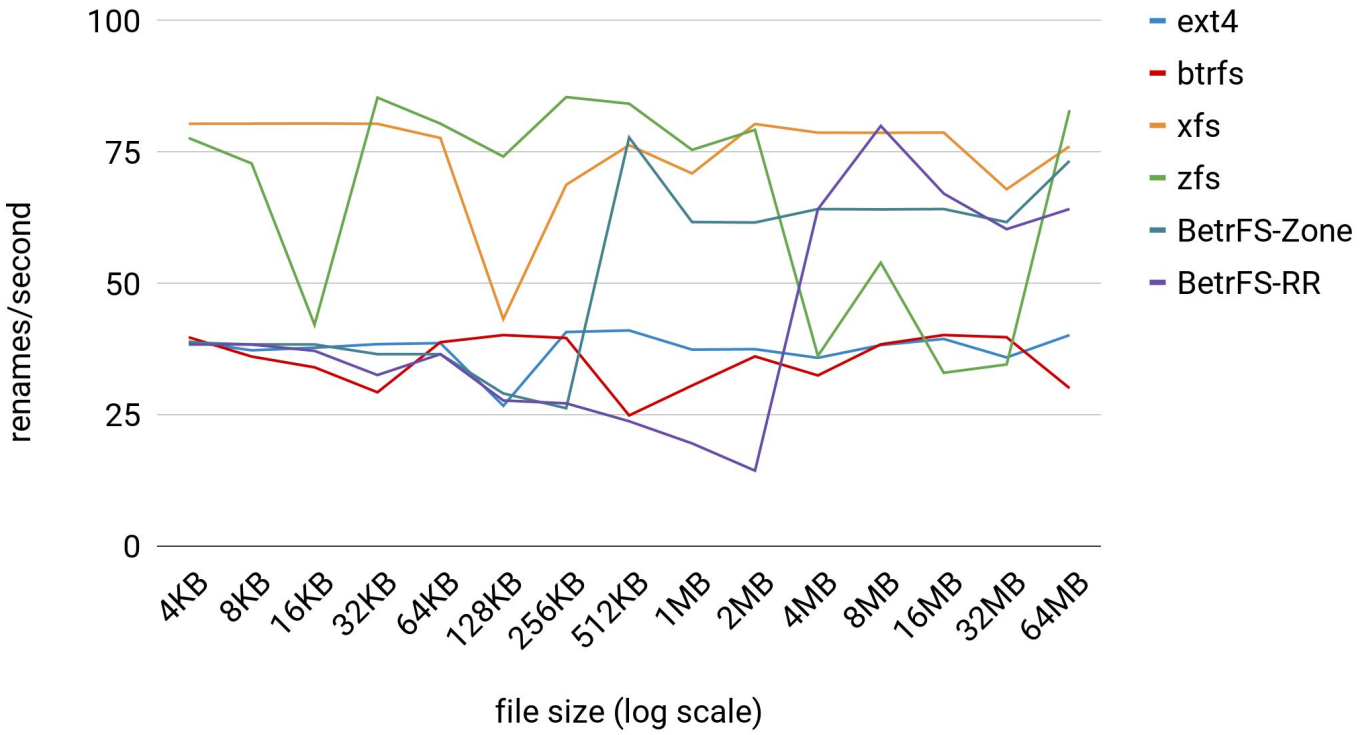
Range-rename doesn't charge other operations as much as zoning

Evaluation

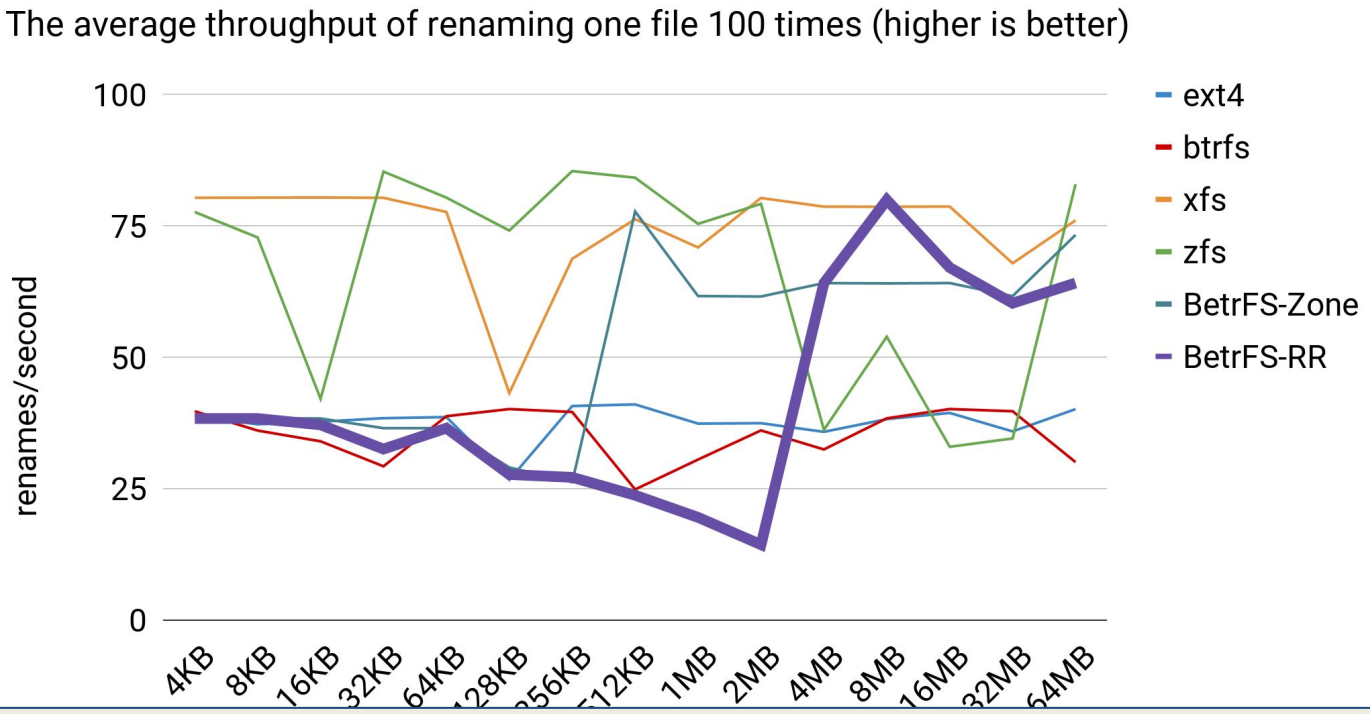
	other operations	rename	applications
range-rename	😃	🤔	🤔

Rename Throughput

The average throughput of renaming one file 100 times (higher is better)



Rename Throughput



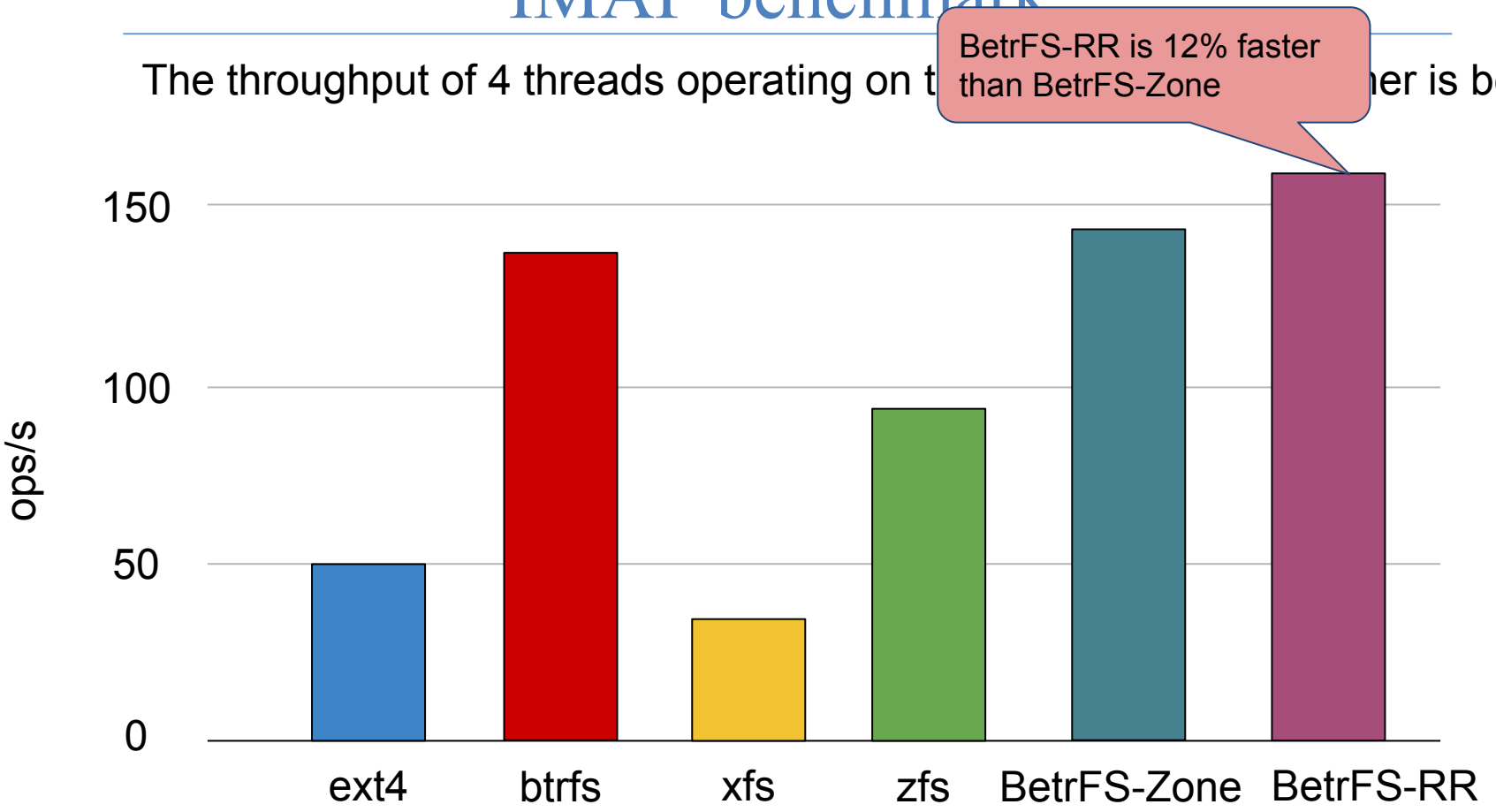
In normal cases, range-rename can rename as fast as other file systems

Evaluation

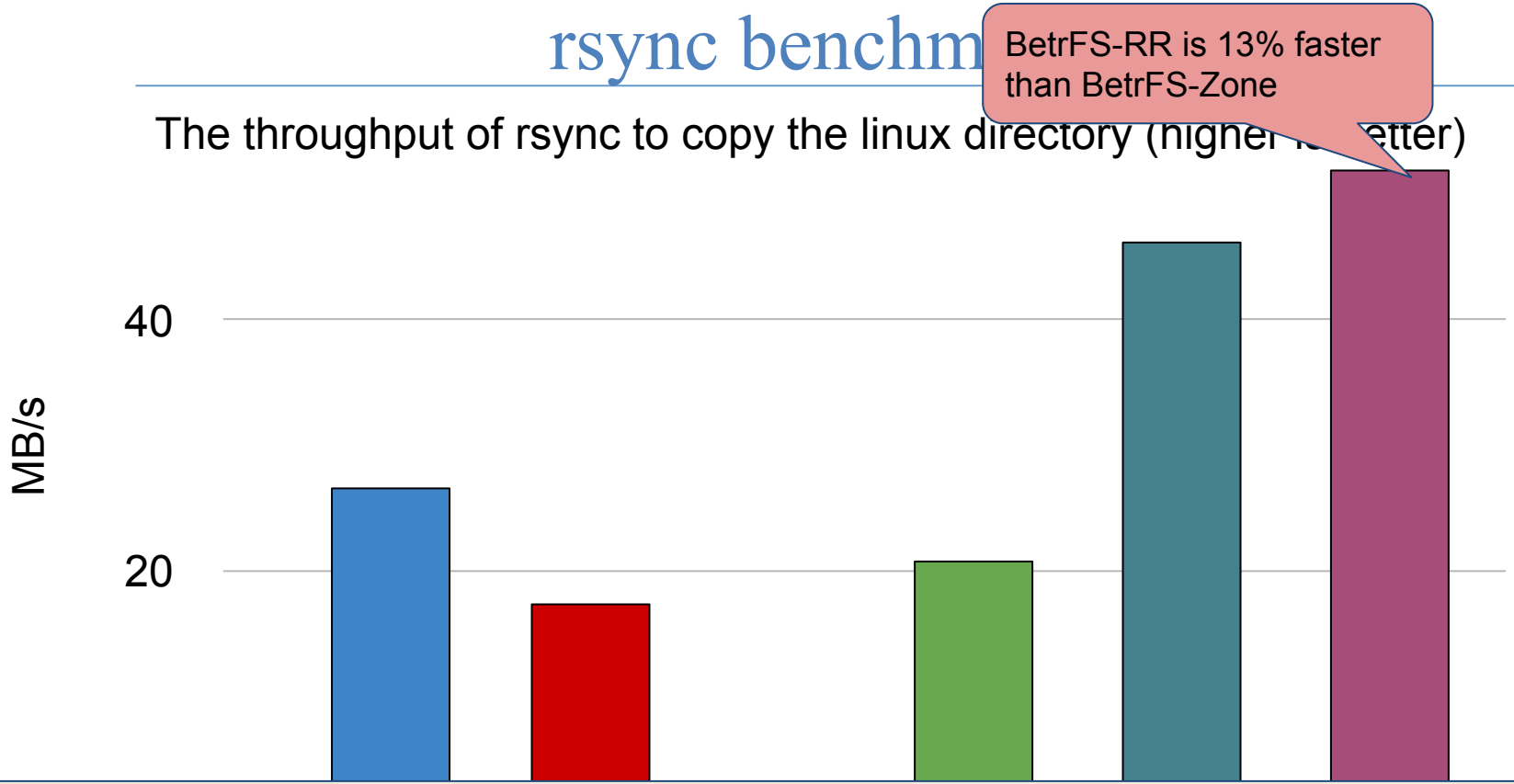
	other operations	rename	applications
range-rename	😃	😃	🤔

IMAP benchmark

The throughput of 4 threads operating on t ner is better)



rsync benchm



BetrFS-RR is faster than BetrFS-Zone in application benchmarks

Evaluation

	other operations	rename	applications
range-rename	😃	😃	😄

Evaluation

	other operations	rename	applications
range-rename	😃	😃	😄

- Lower taxes on non-rename operations
  - A few regressions
- Worst case rename costs: logarithmic in size
- Additional opportunities for range rename

in paper

Conclusion

- BetrFS with range-rename
  - maintain full-path indexing
  - decent rename performance
  - no tradeoff: locality, rename and other operations

Web: betrfs.org
Code: https://github.com/oscarlab/betrfs
Email: betrfs@googlegroups.com