NVRAMOS 2025

Understanding Linux Distributed Lock Management Overheads

Jaehyun Hwang
(jh.hwang@skku.edu)

with Taeyoung Park, Yunjae Jo, Daegyu Han, Beomseok Nam
Sungkyunkwan University

Confessions

- **This talk includes our FAST'26 paper**
  - which has not yet been presented at the conference

    Slides are not yet polished..

  - which is a short paper (7 pages)

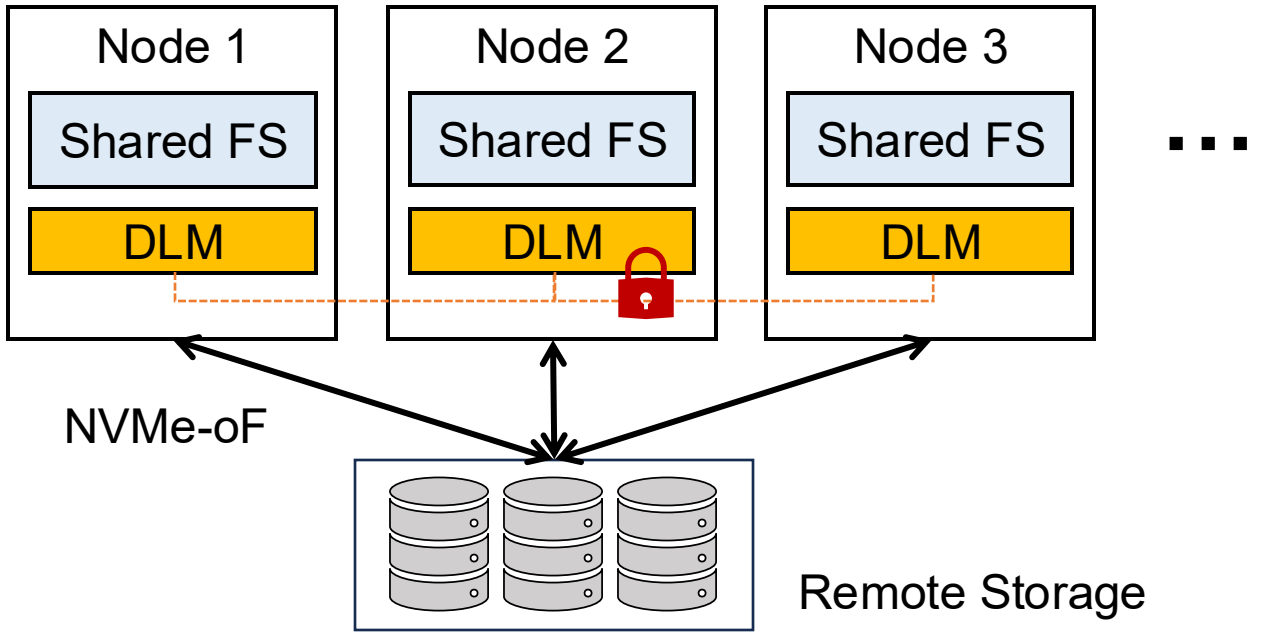
    Not enough for a 40-min talk

- **Initially, planned to cover two topics**

- **Finally, "FAST'26 paper + a few side stories" seems fine**
  - which led to a title change..

Target environment

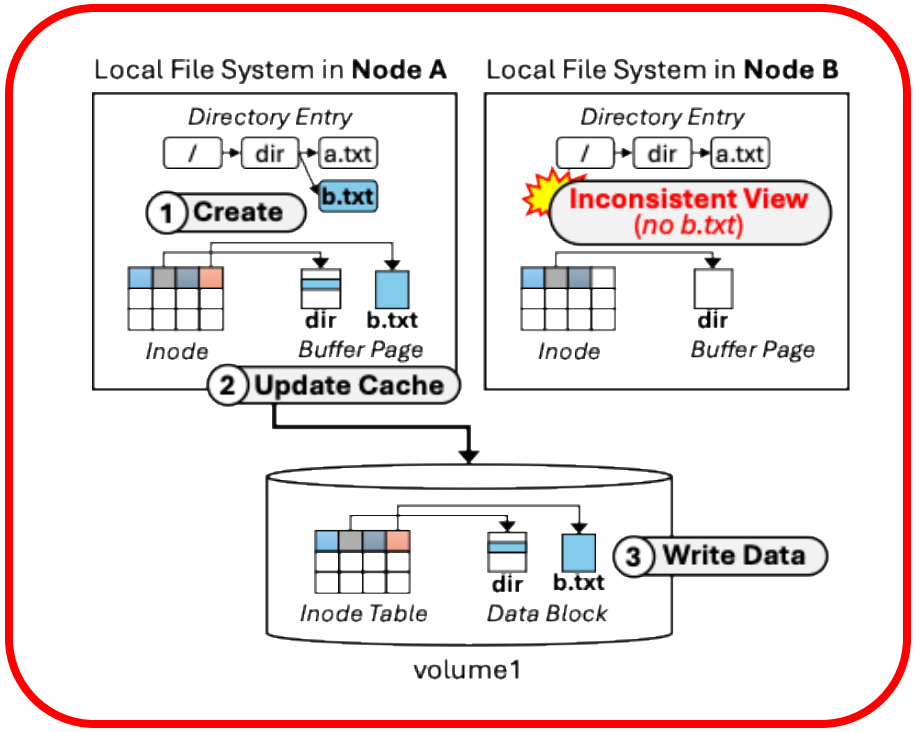
- **Storage disaggregation in data centers**
  - Remote storage access via NVMe-over-Fabrics (NVMe-oF)
  - Shared-disk file systems (e.g., GFS2, OCFS2) or local file systems (e.g., EXT4)
  - Distributed Lock Manager (DLM) for file operations

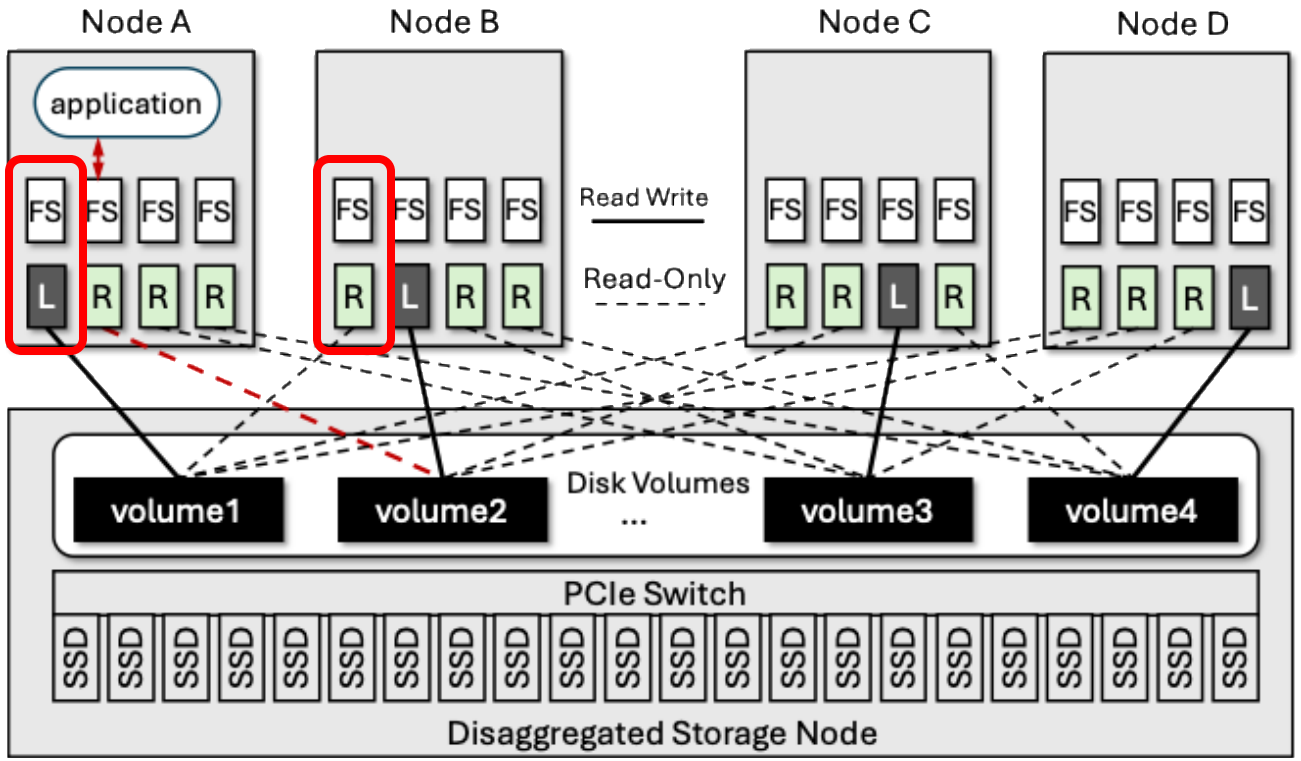


Initial motivation

- **[CLOUD'23] EXT4-oF**
  - Maintains cache coherence for **EXT4** metadata across nodes
  - Extends **EXT4** into a *shared file system* for immutable data
  - Currently its functionality is limited to "`open()`"

Can we overcome these limitations?

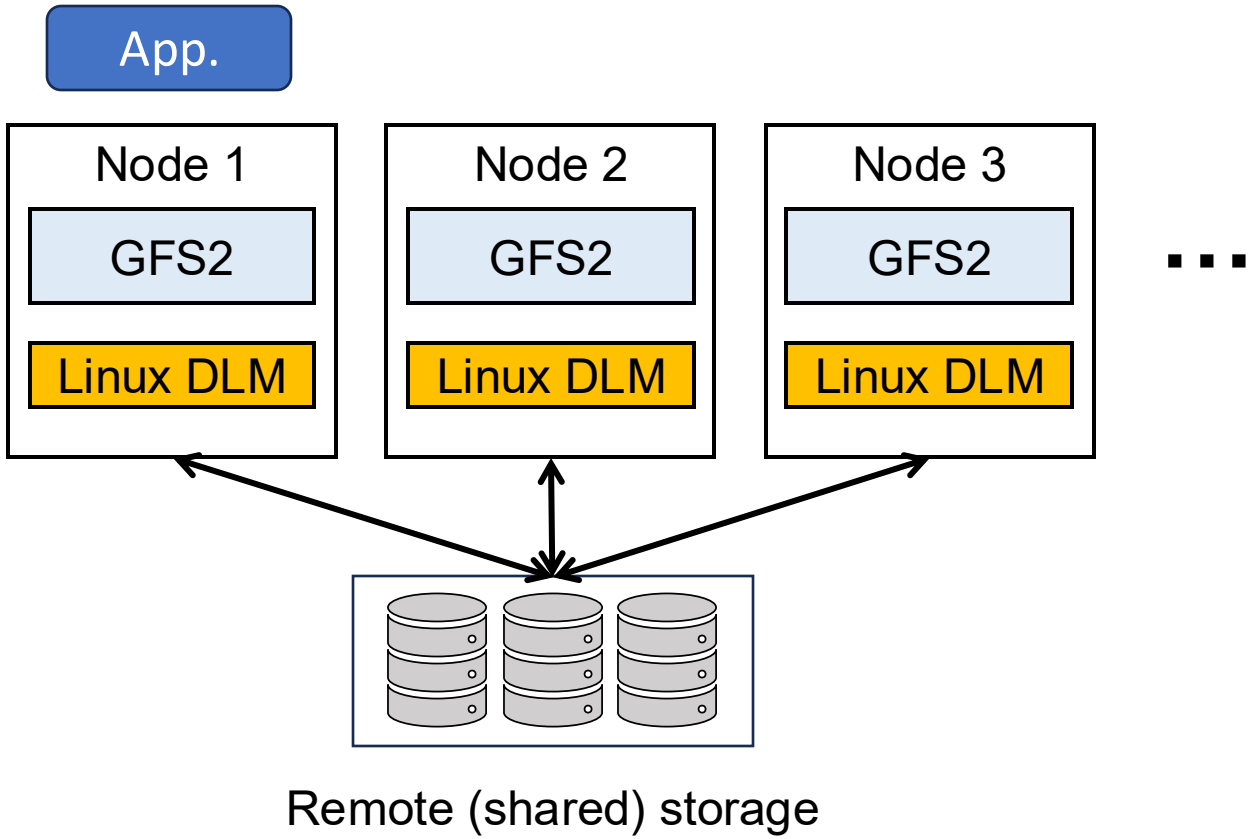


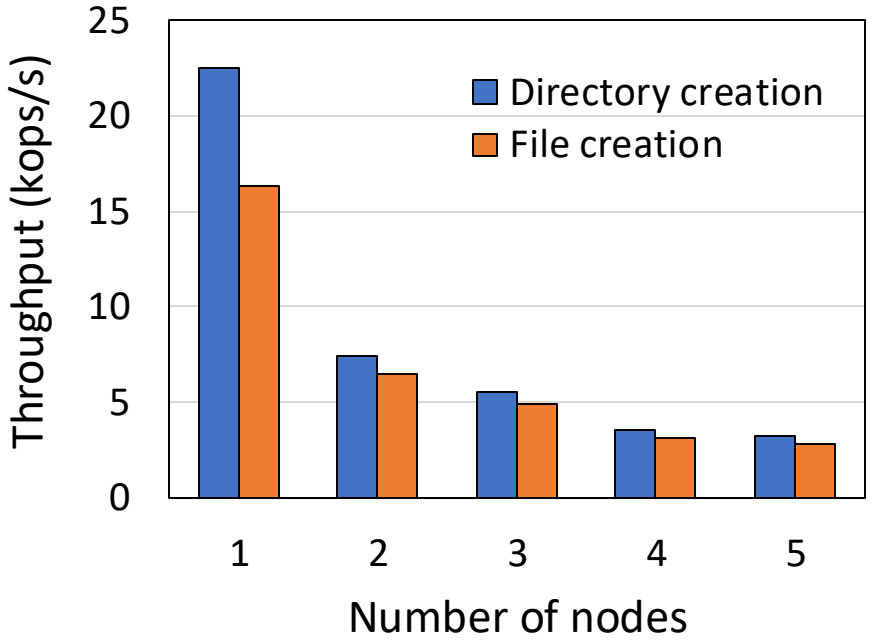


Real motivation

- **Scalability issue of shared-disk file systems**

  even when only a single node is active!

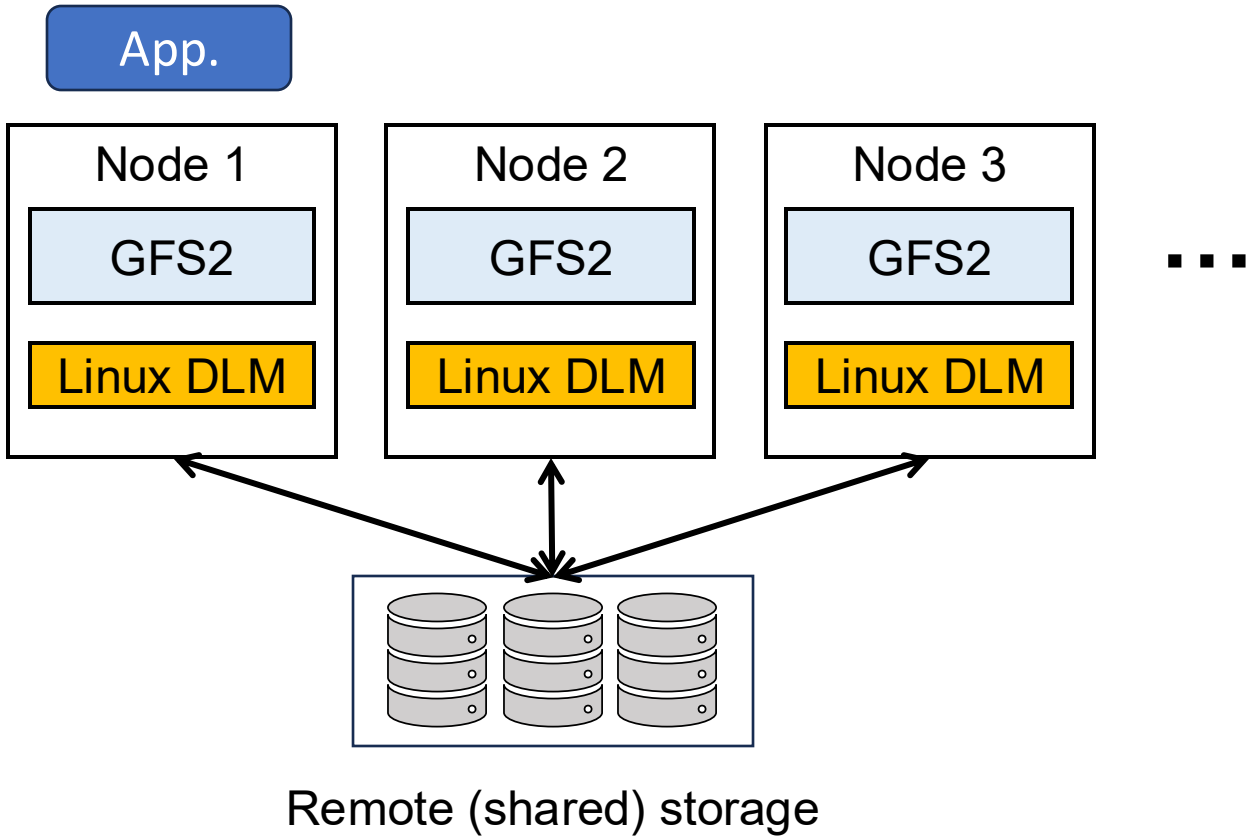


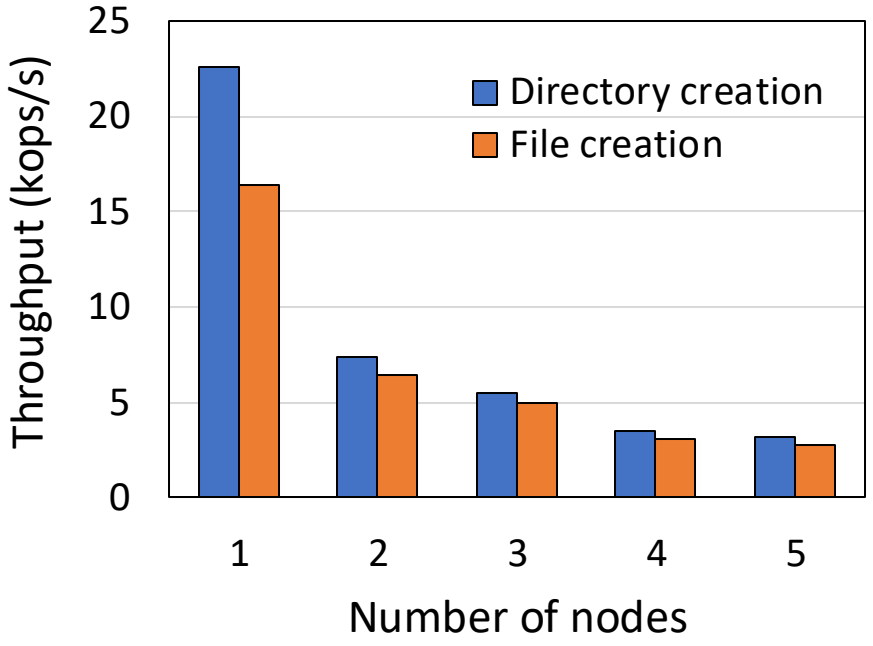


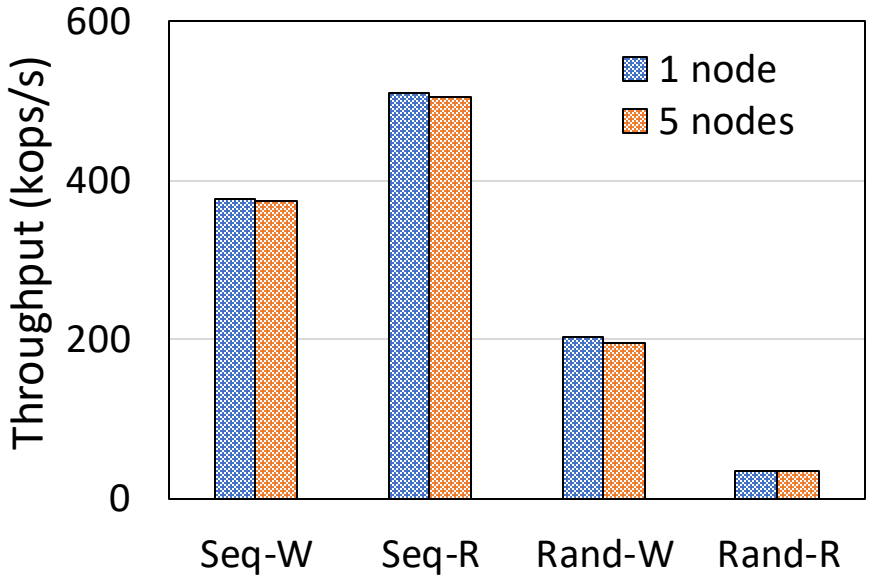
Real motivation

- **Scalability issue of shared-disk file systems**

  even when only a single node is active!

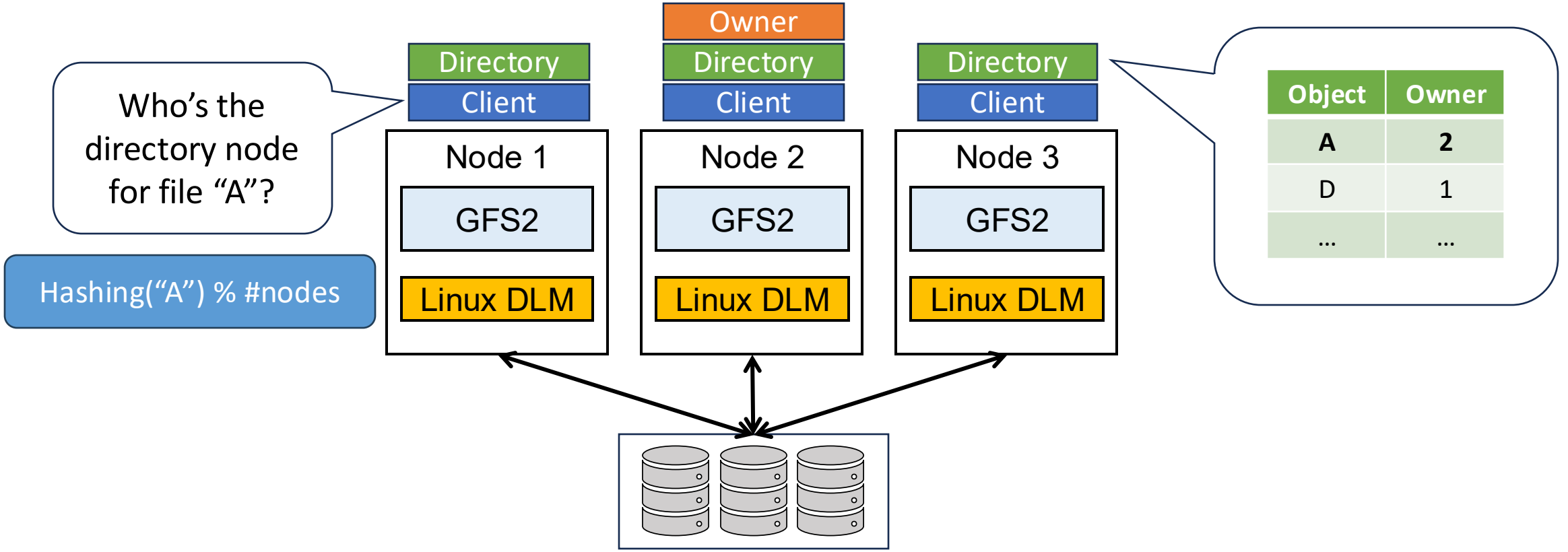




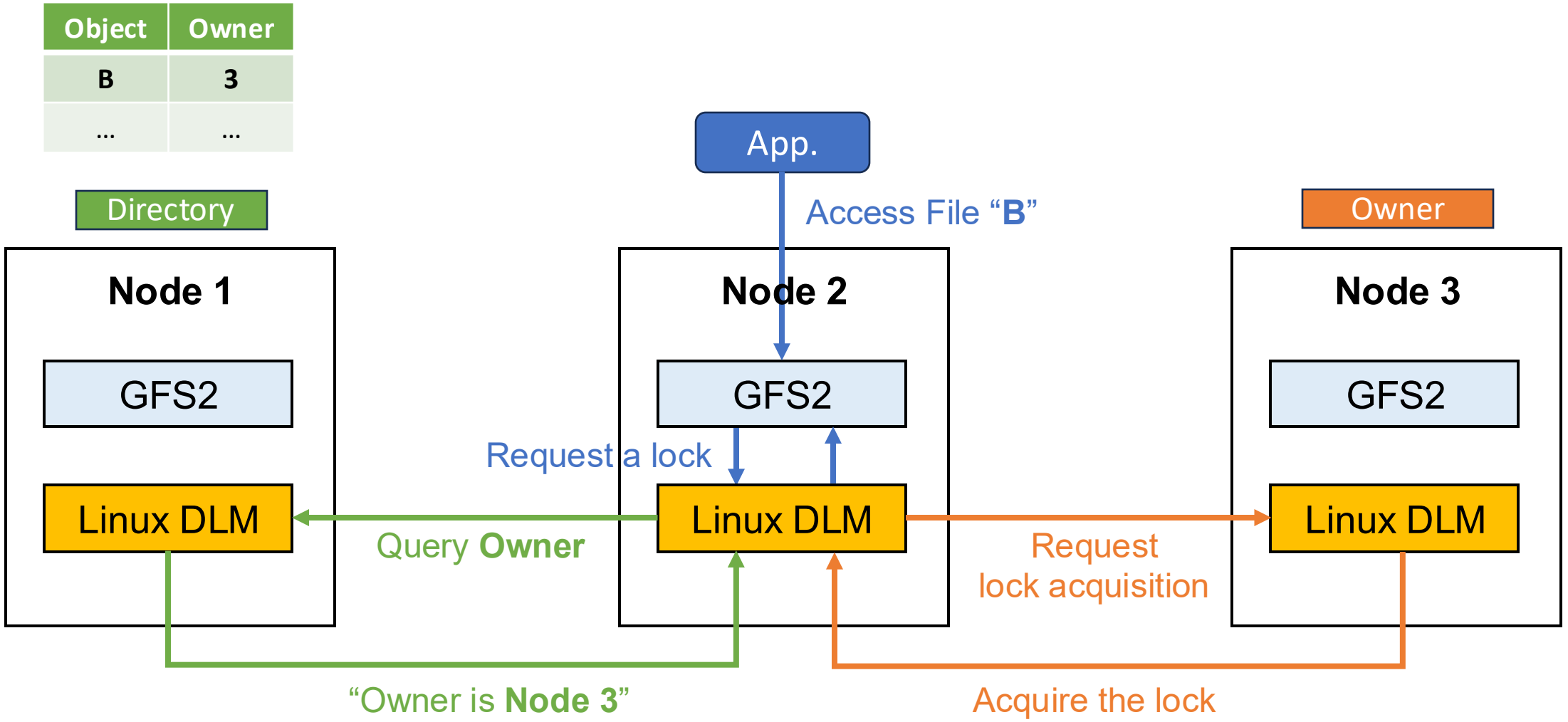


Linux distributed lock management

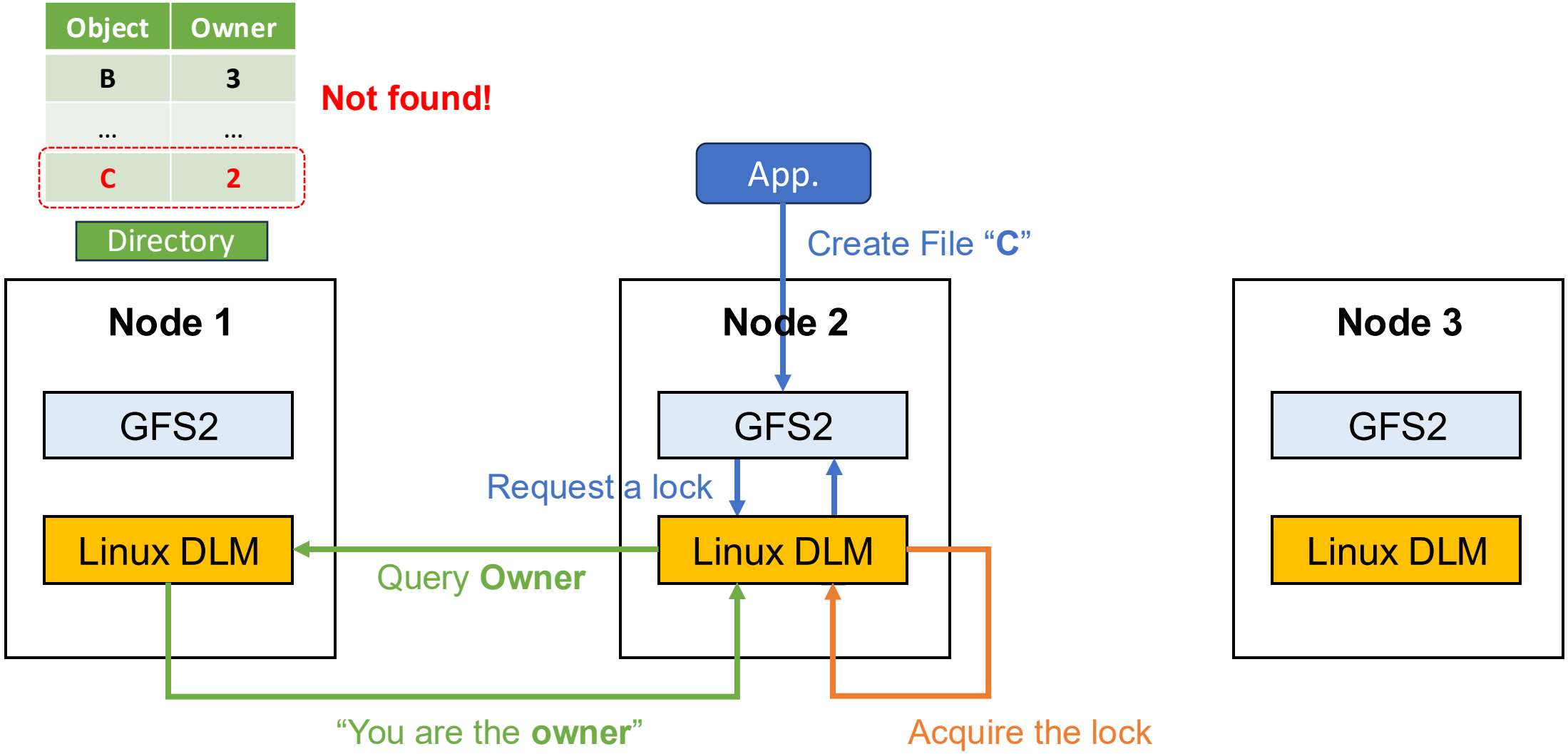
- **Directory:** finds the owner node of the requested object (file or directory)
- **Owner:** is the node who has the ownership of the requested object



Lock acquisition example



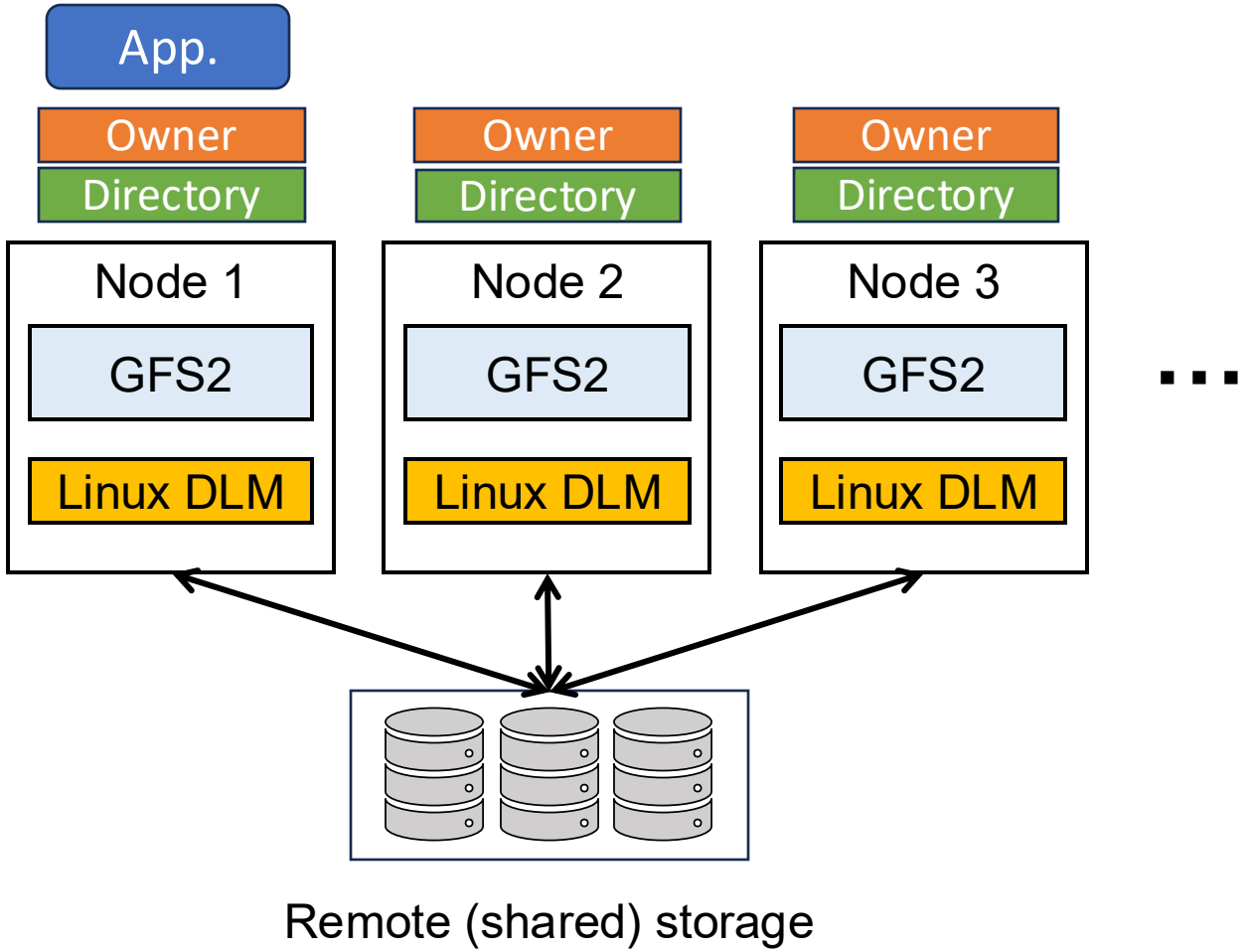
How about creation?



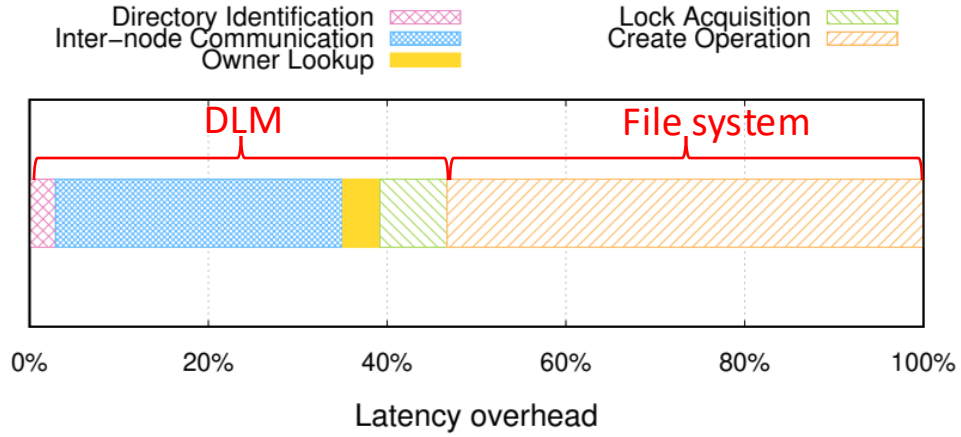
Why scalability issue for creation?

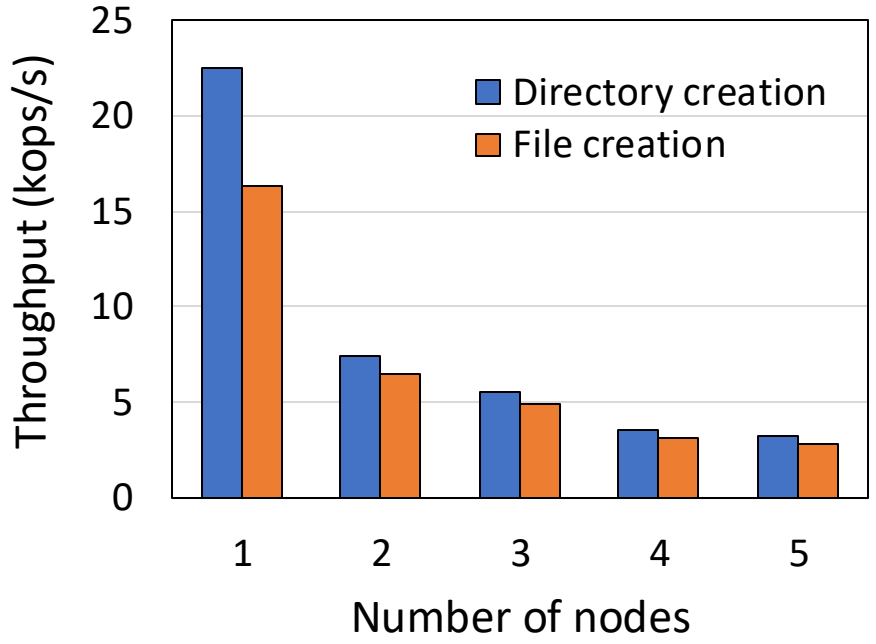
Single-node case: locking is entirely local!

vs.



[A multi-node latency breakdown]

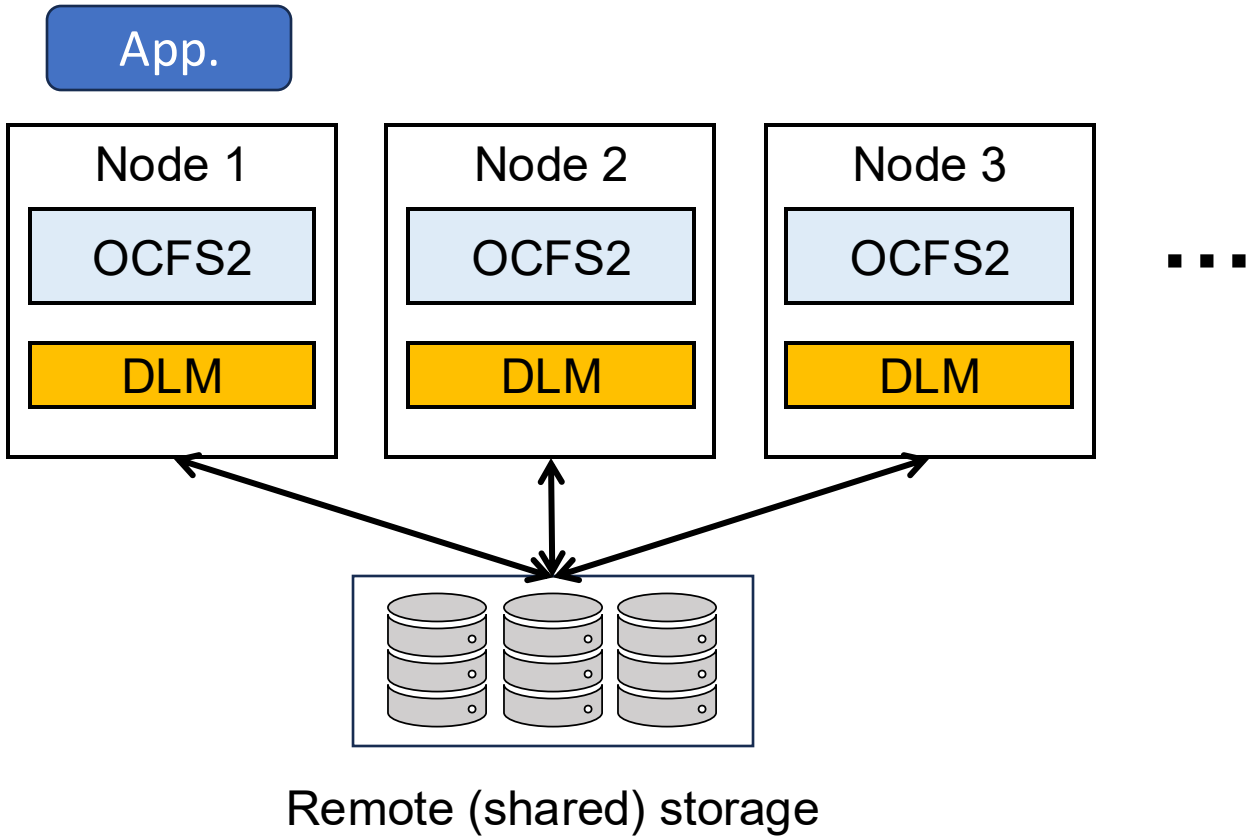


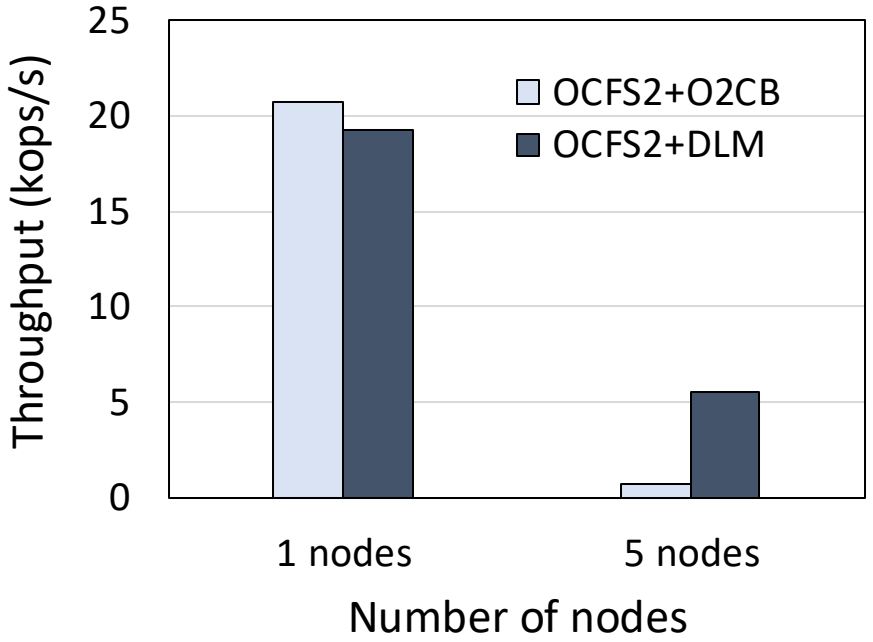


What about other combinations?

- **OCFS2 + (O2CB or Linux DLM)**

A similar trend!





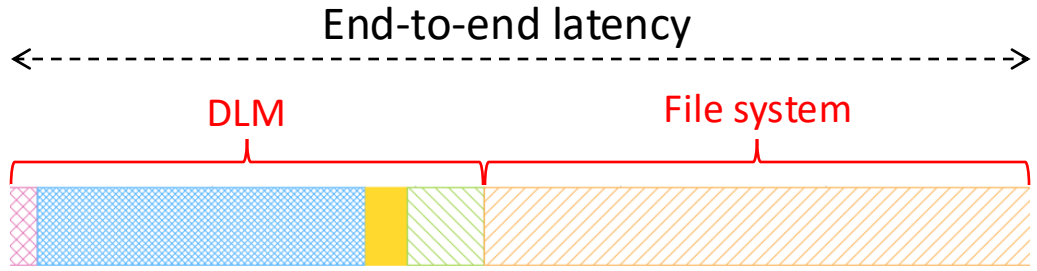
Our solution: Lockify (notifies lock ownership)

- **Self-owner notifications**
  - Observation: DLM doesn't care about the lock owner of non-existent objects!
  - Let's notify the directory node of ***self-ownership*** upon object creation

- **Extended lock acquisition interface**
  - File systems decide whether to enable Lockify through the extended interface
  - Minimize modifications to existing file systems

- **Asynchronous ownership management**
  - Introduce a wait-list to track and resend unconfirmed notifications
  - Ensure DLM-level consistency across nodes

Lockify: self-owner notifications



Object	Owner
B	3
...	...
C	2

Directory

Add “C’s owner is Node 2” **without lookup**

App.

Create File “**C**”

Node 1

Shared FS

Lockify

Node 2

Shared FS

Lockify

Node 3

Shared FS

Lockify

Request a lock with Lockify enabled

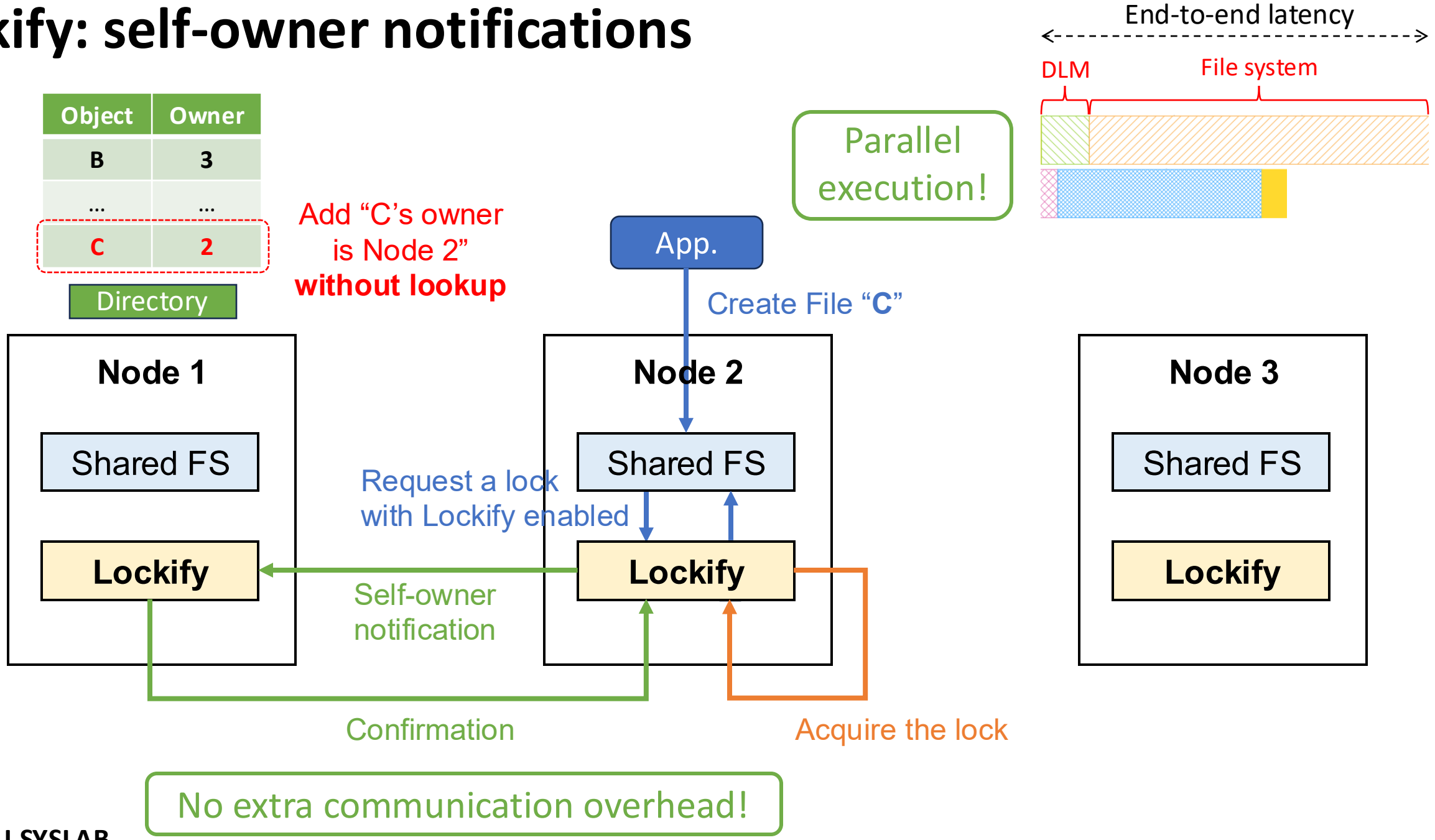
Self-owner notification

Confirmation

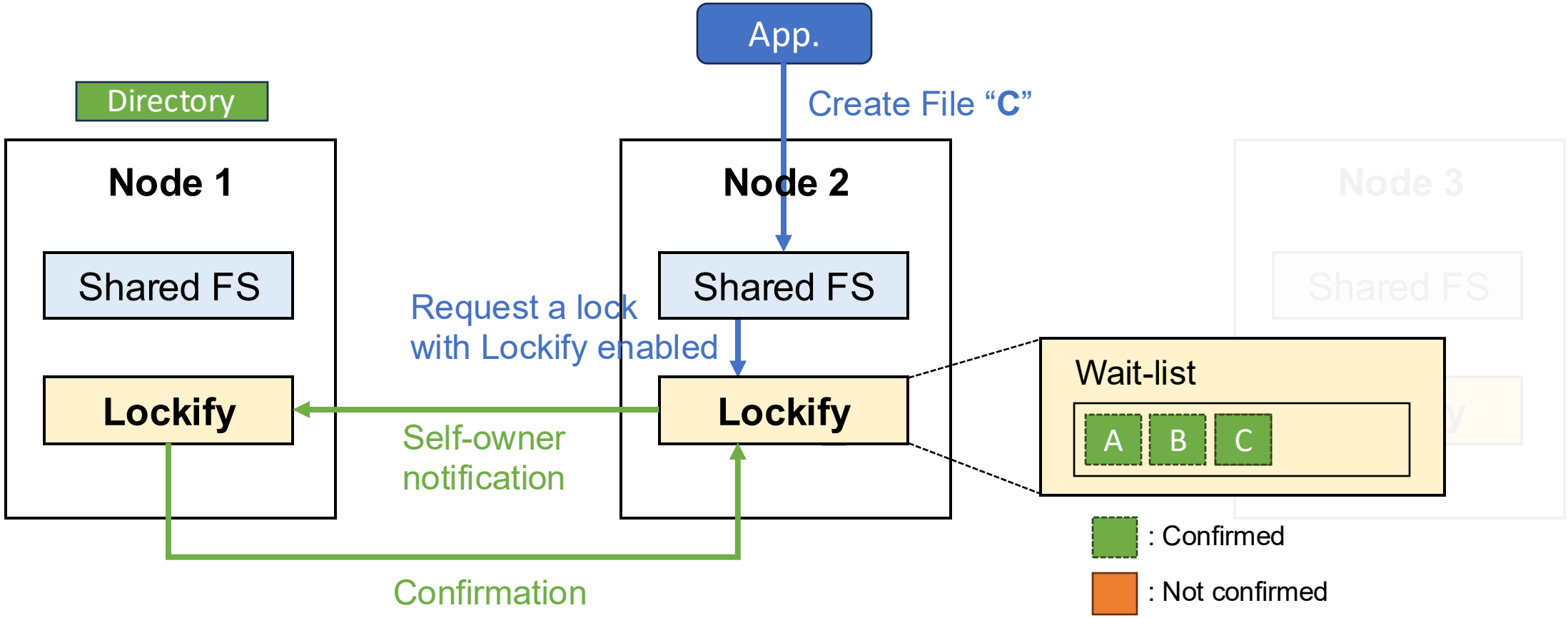
Acquire the lock

No extra communication overhead!

Lockify: self-owner notifications



Lockify: asynchronous ownership management

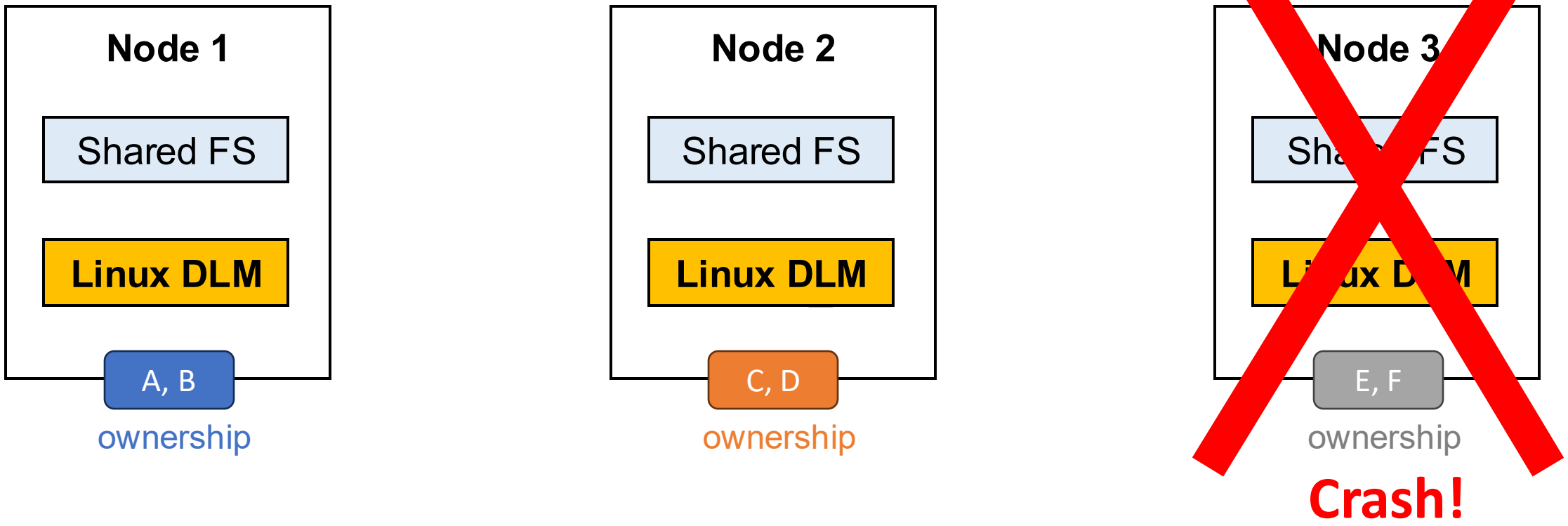


How to deal with node crashes: Linux DLM's recovery

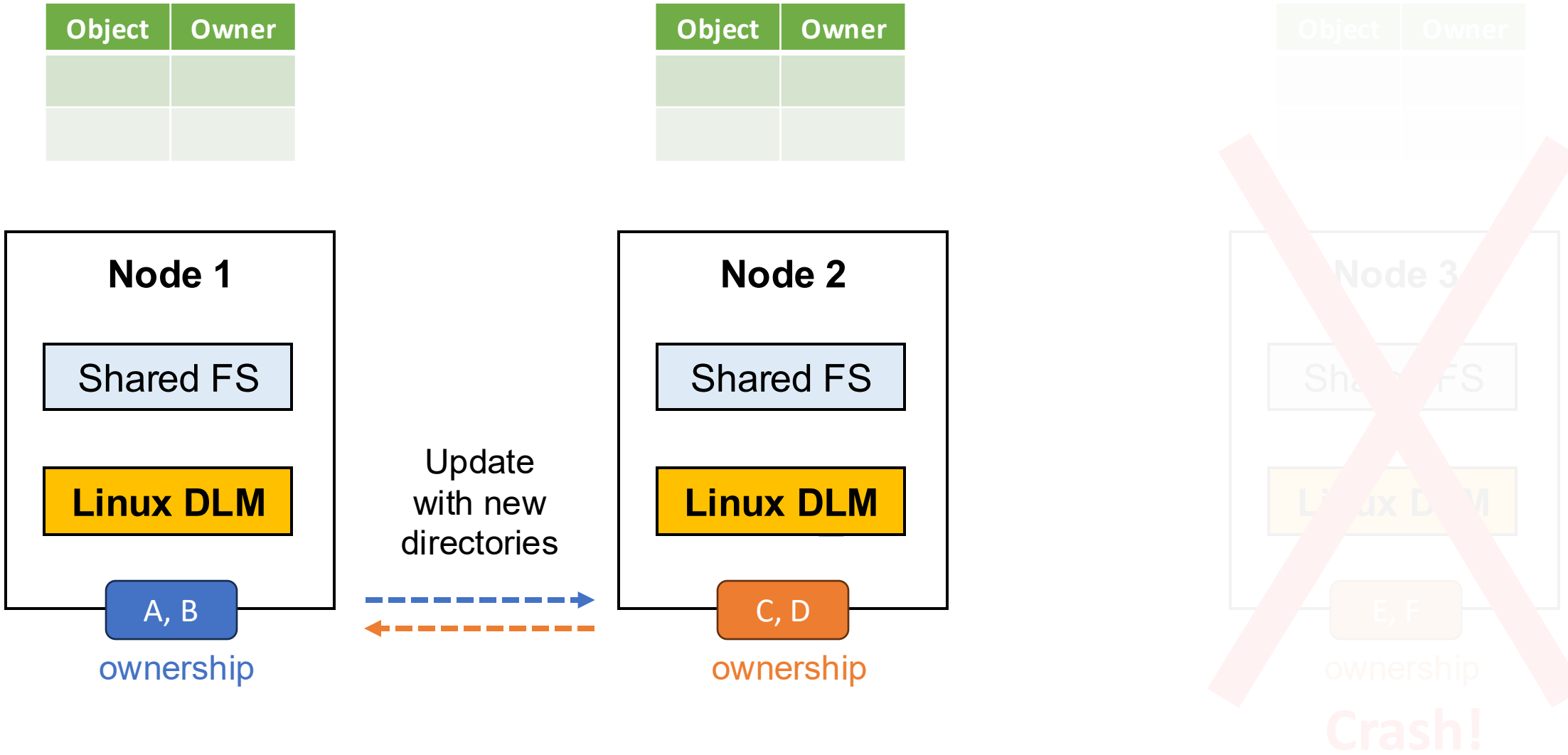
Object	Owner
C	2
E	3

Object	Owner
A	1
F	3

Object	Owner
B	1
D	2



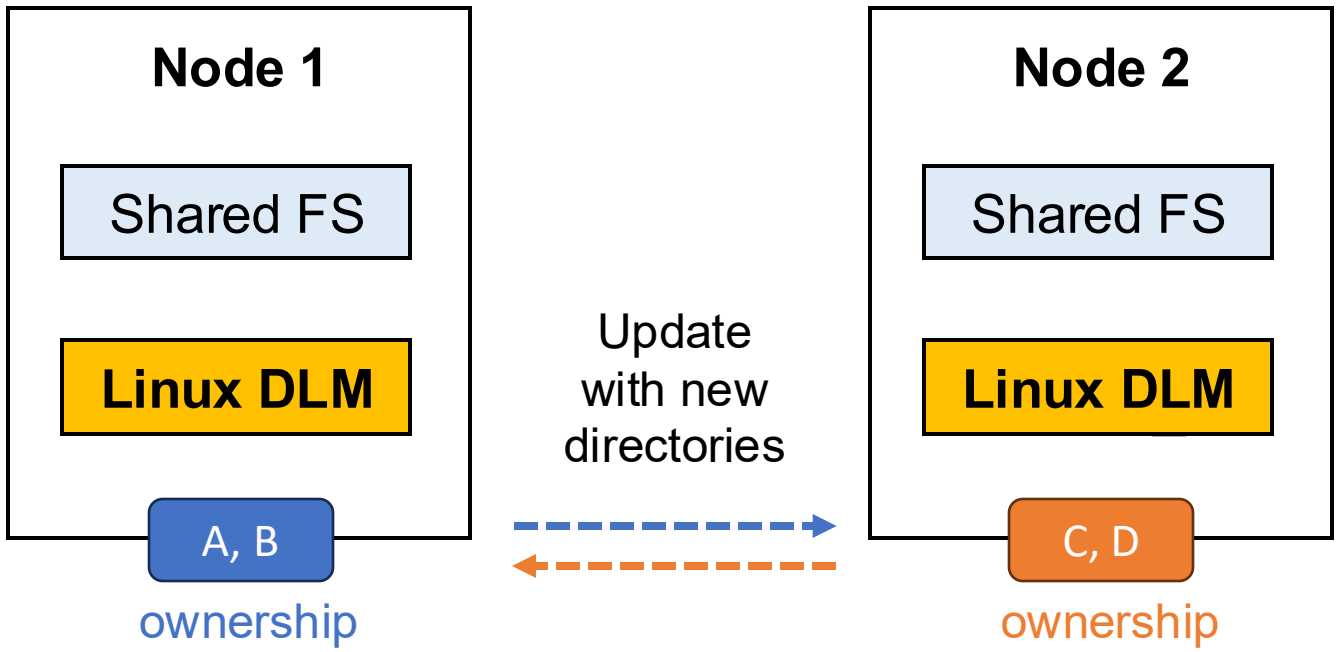
How to deal with node crashes: Linux DLM's recovery



How to deal with node crashes: Linux DLM's recovery

Object	Owner
C	2
D	2

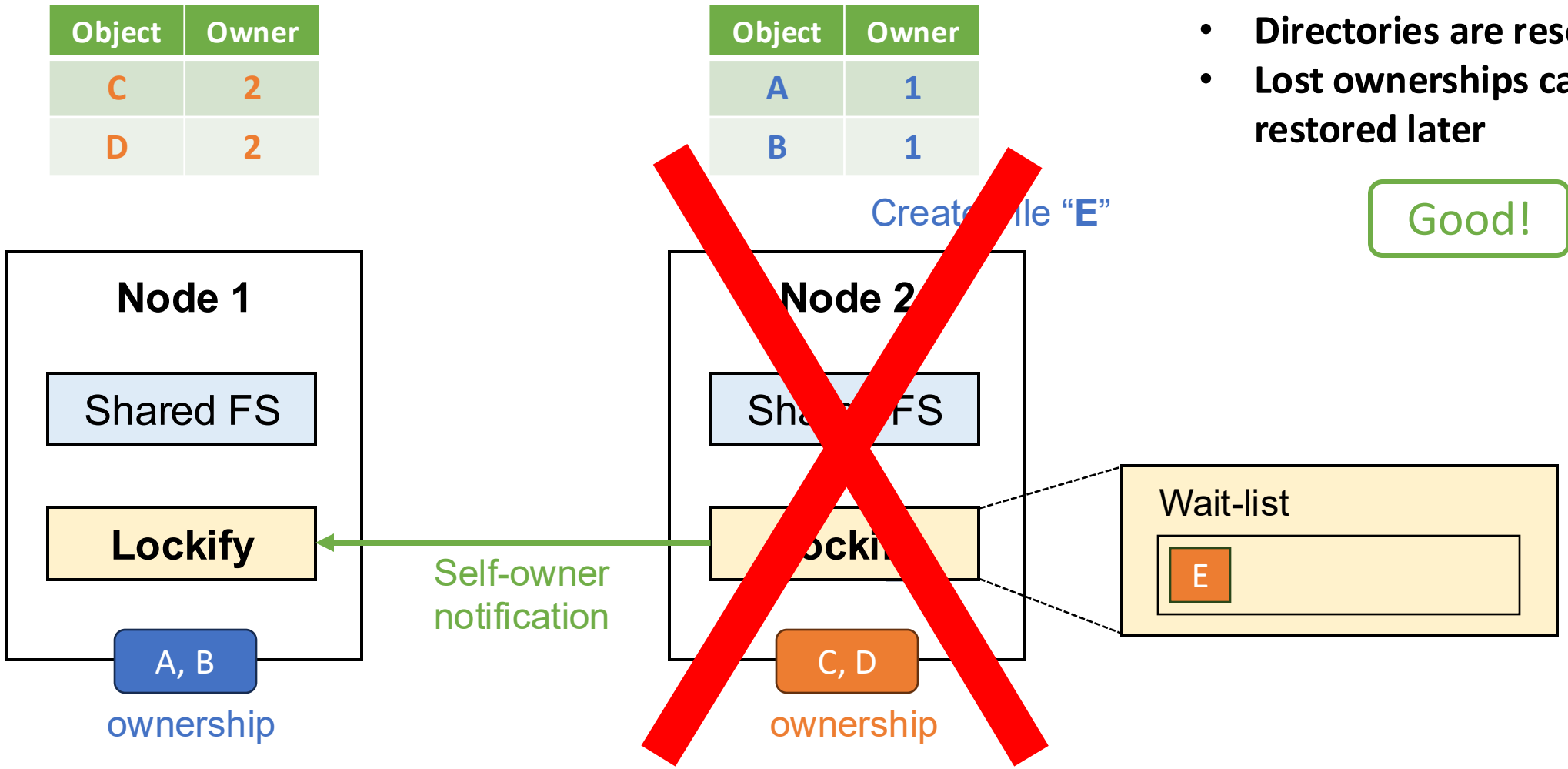
Object	Owner
A	1
B	1



What if some node wants to access E or F?

=> That node becomes the new owner!

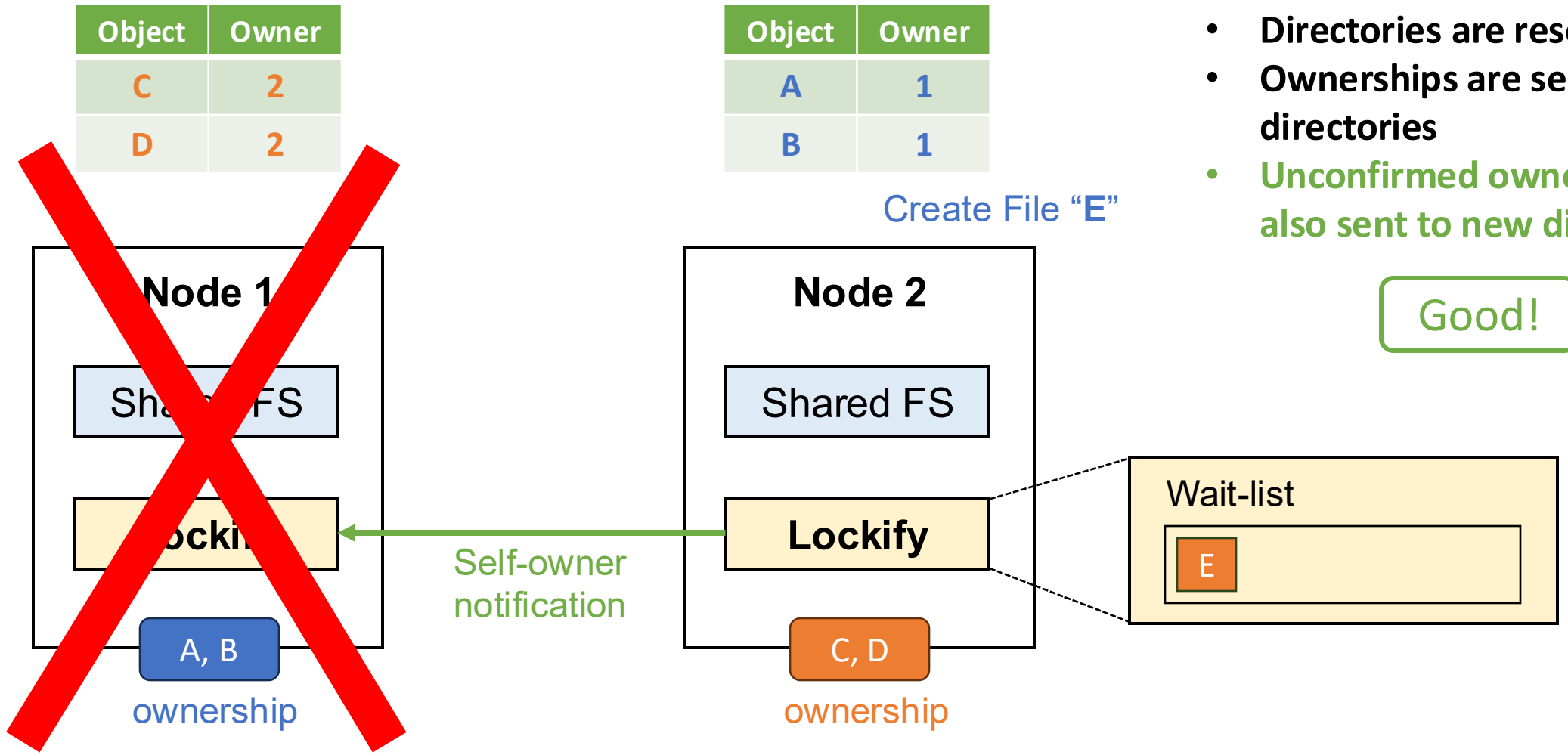
How to deal with node crashes: Lockify's recovery



- Directories are reset
- Lost ownerships can be restored later

Good!

How to deal with node crashes: Lockify's recovery



- Directories are reset
- Ownerships are sent to new directories
- Unconfirmed ownerships are also sent to new directories

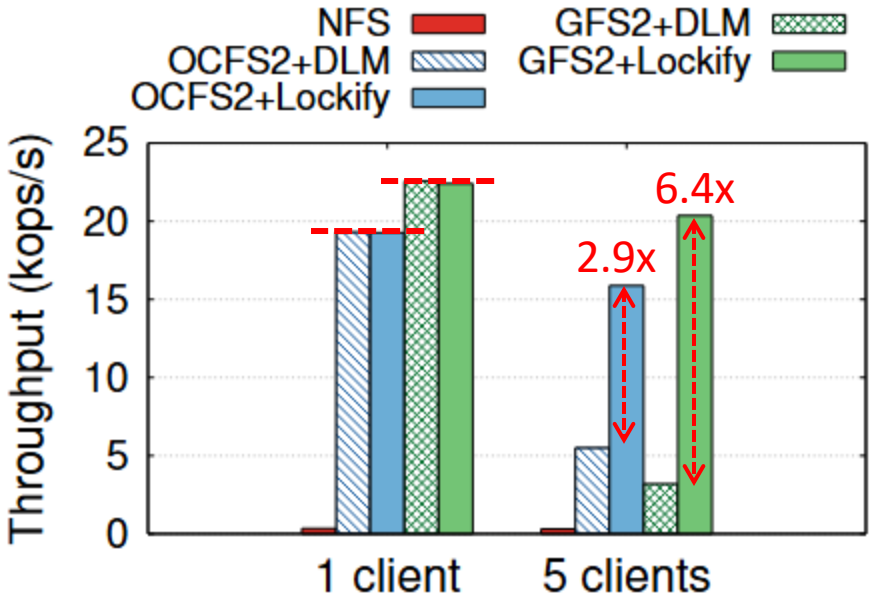
Good!

Lockify evaluation setup

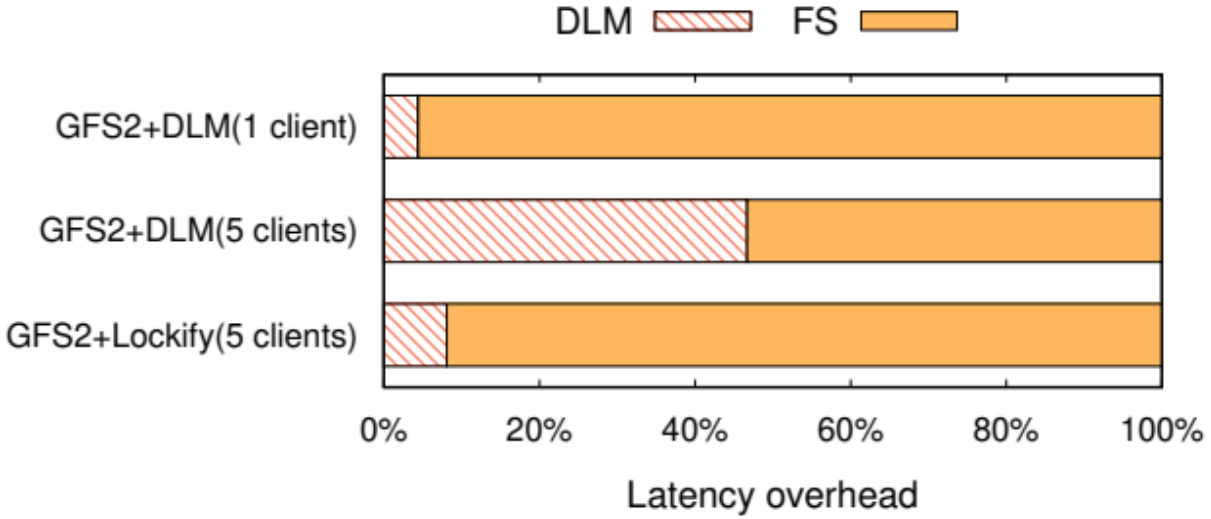
- **Implemented in the Linux kernel 6.6.23**
  - On top of the kernel DLM
  - Also modified GFS2 and OCFS2 with the extended lock acquisition interface
- **Five 20-core servers connected via 56Gbps links**
  - Connected to a shared storage server using NVMe-over-TCP
  - 250GB Samsung 970 EVO Plus NVMe SSD
- **Workloads**
  - Microbenchmarks: **mdtest**
    - Low-contention scenario: 1 active node with varying #nodes from 1 to 5
    - High-contention scenario: 5 active nodes
  - Real-world workloads: **Postmark** and **Filebench**

Microbenchmarks (mdtest)

- **Low-contention scenario with a single active node**



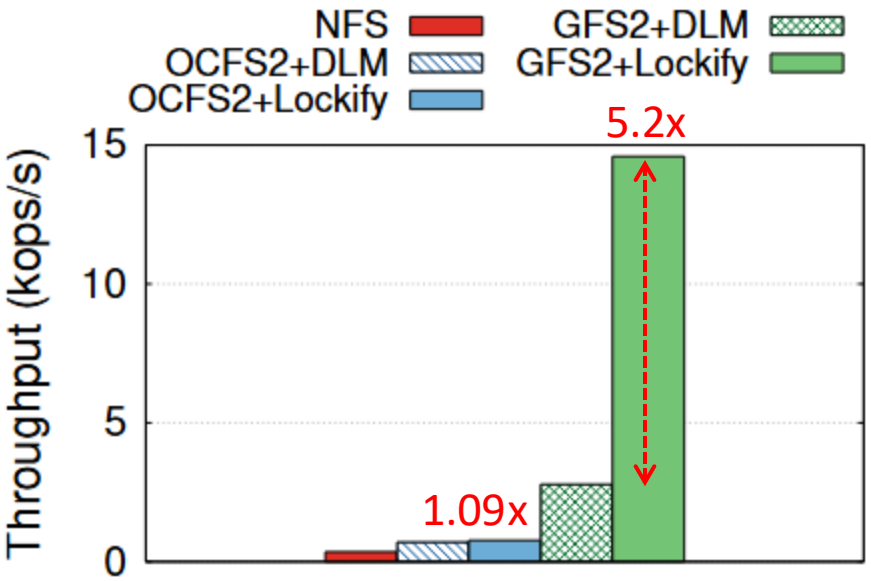
(a) Directory creation.



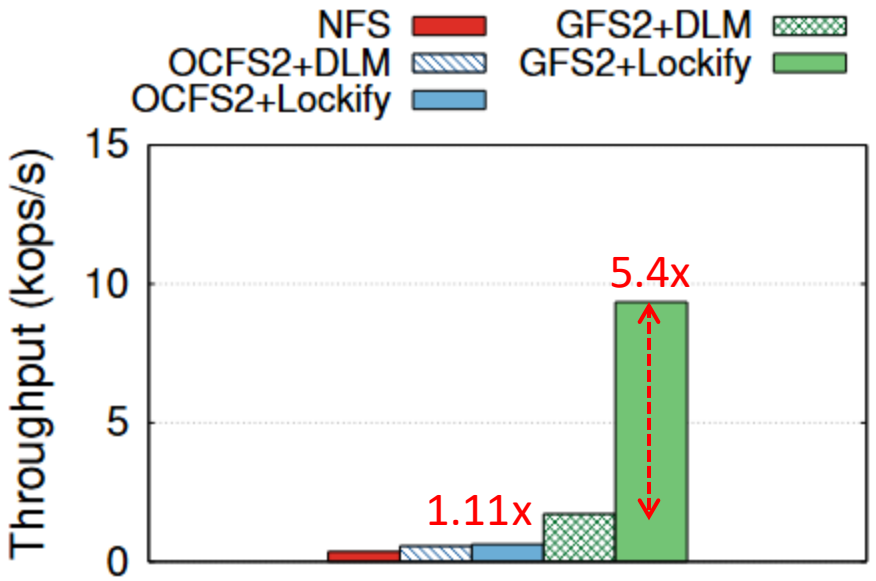
Lockify significantly reduces DLM-side latency with low overhead, achieving higher throughput compared to existing solutions

Microbenchmarks (mdtest)

- **High-contention scenario with 5 active nodes**



(a) Directory creation.

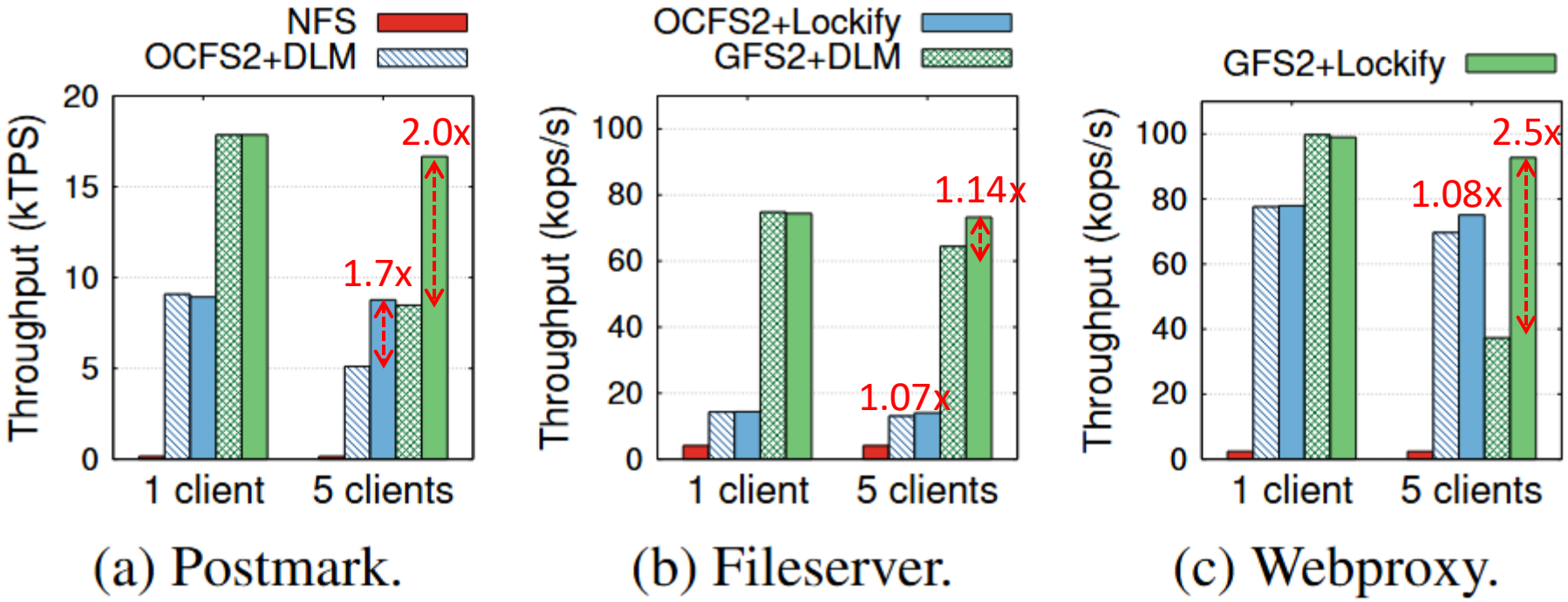


(b) File creation.

GFS2's internal optimization (regarding parent directory locks) helps Lockfy improve performance even in high-contention scenarios

Real-world workloads

- **Low-contention scenario with a single active node**



Lockify achieves better performance across diverse workloads!

Summary

- **For object creations, shared-disk file systems:**
  - Suffer from high lock-acquisition latency with multiple nodes
  - Face this scalability challenge even under low-contention scenarios
- **With self-owner notifications, Lockify:**
  - Minimizes lock-acquisition latency through execution pipelining
  - Achieves up to 6.4x higher throughput compared to kernel DLM and O2CB

https://github.com/skku-syslab/lockify

Thank you!
Q&A