FAST'26

Lockify: Understanding Linux Distributed Lock Management Overheads in Shared Storage

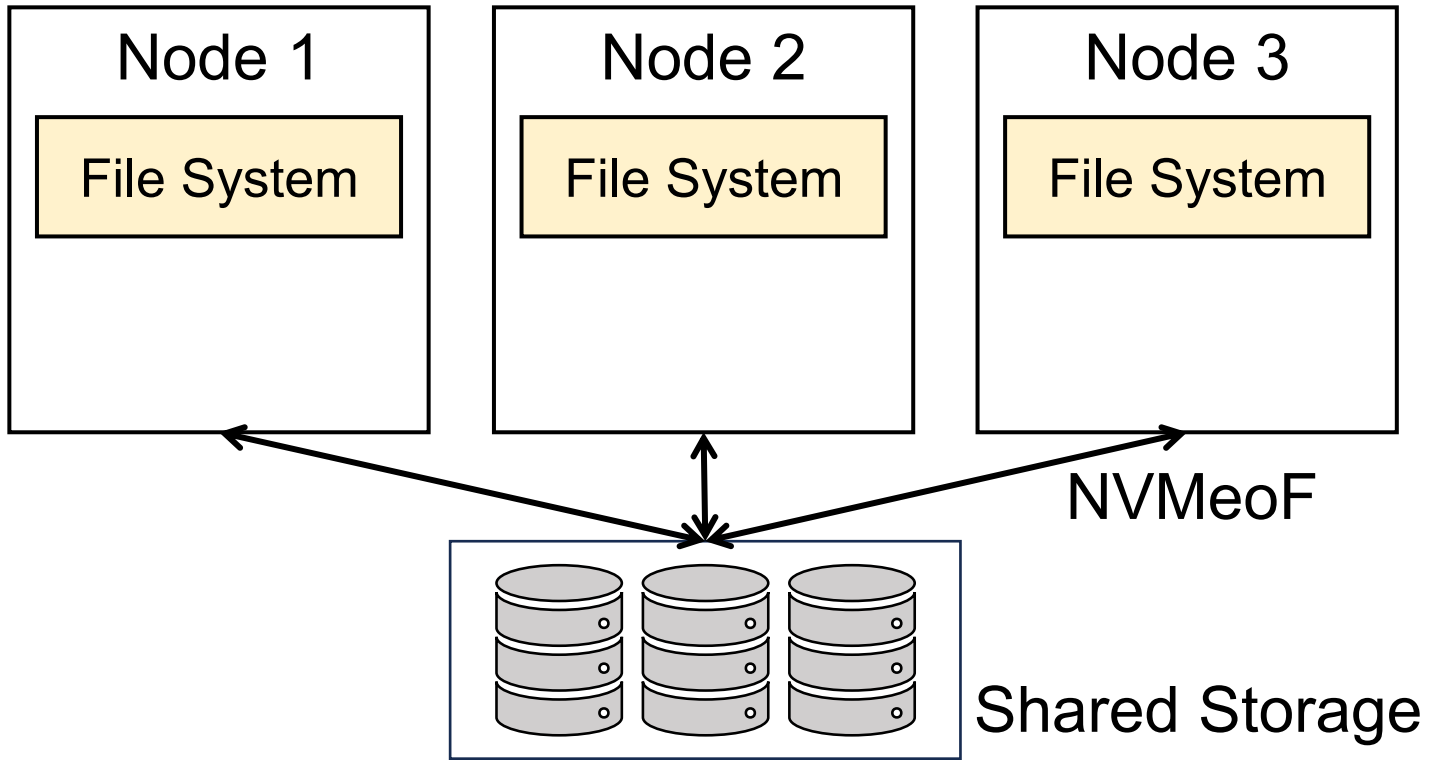
Taeyoung Park, Yunjae Jo, Daegyu Han, Beomseok Nam, and Jaehyun Hwang

SUNG KYUN KWAN UNIVERSITY

Background

Storage disaggregation in data centers

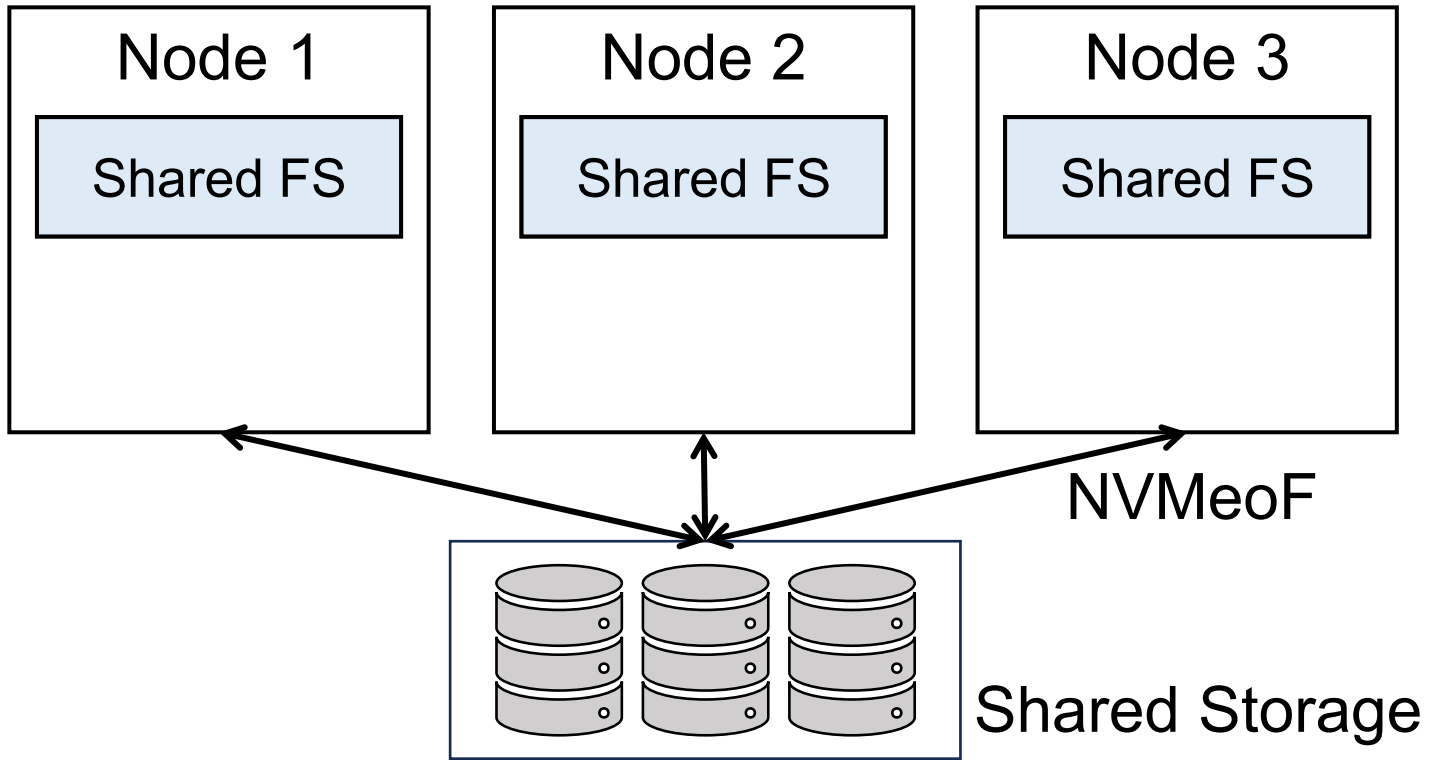
– Remote storage access via NVMe-over-Fabrics (NVMeoF)



Background

Concurrent access across multiple nodes

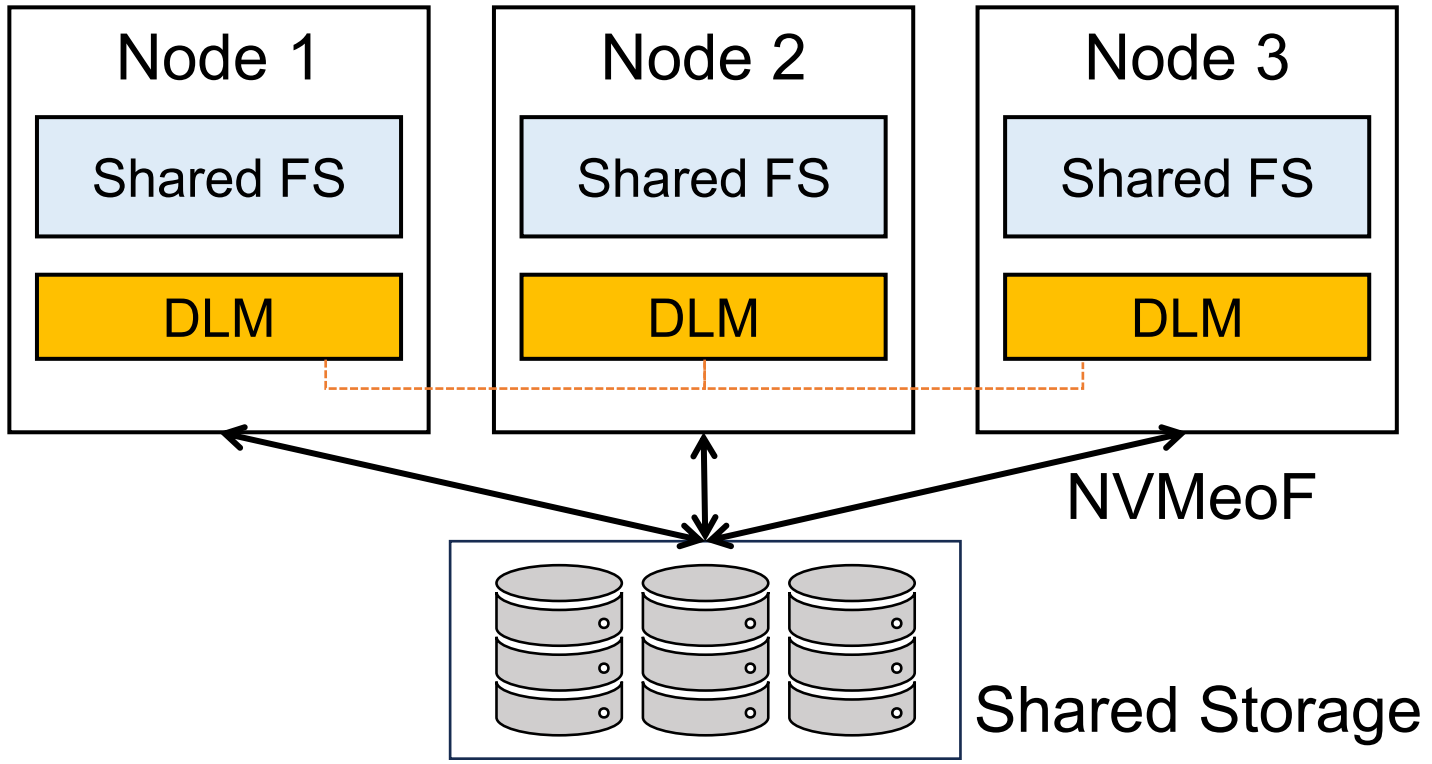
– Need shared-disk file systems (e.g., GFS2, OCFS2)



Background

Lock management becomes a critical component

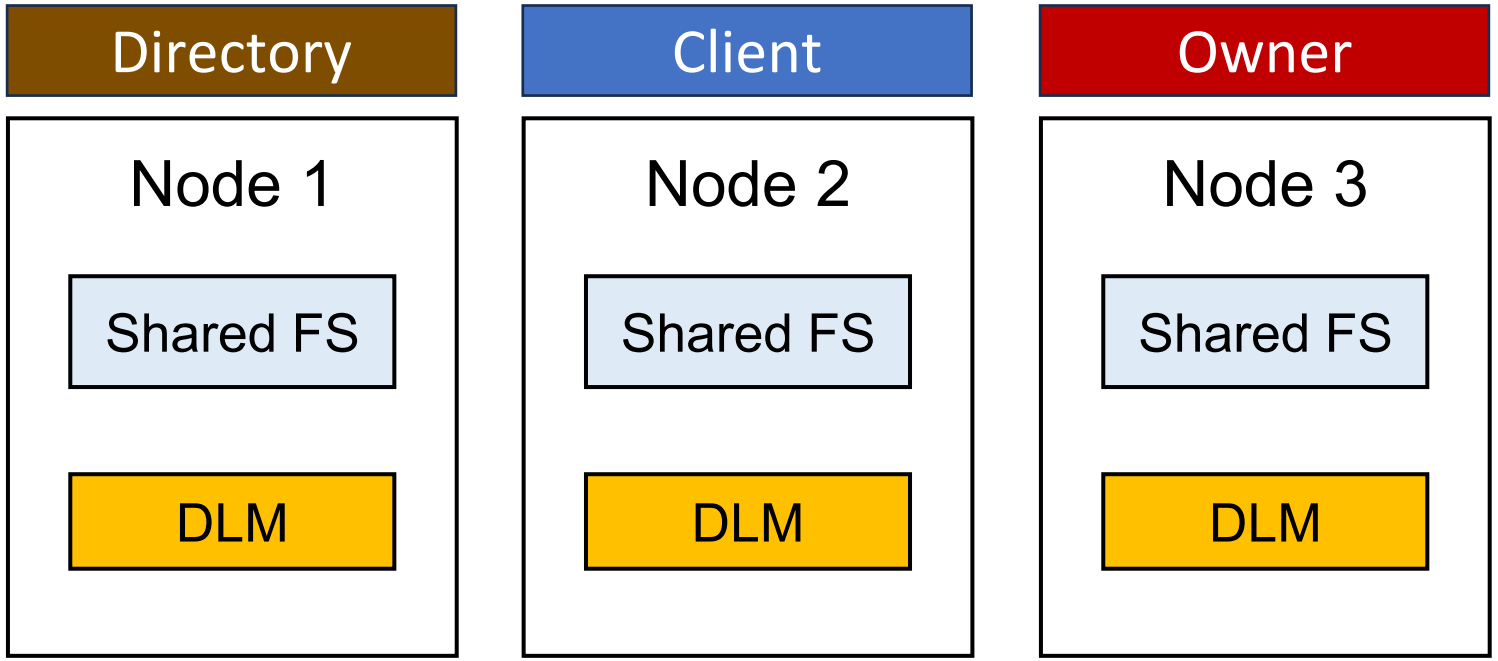
– Use a separate Distributed Lock Manager (DLM) layer in Linux



Distributed Lock Manager (DLM)

Two important roles in DLM

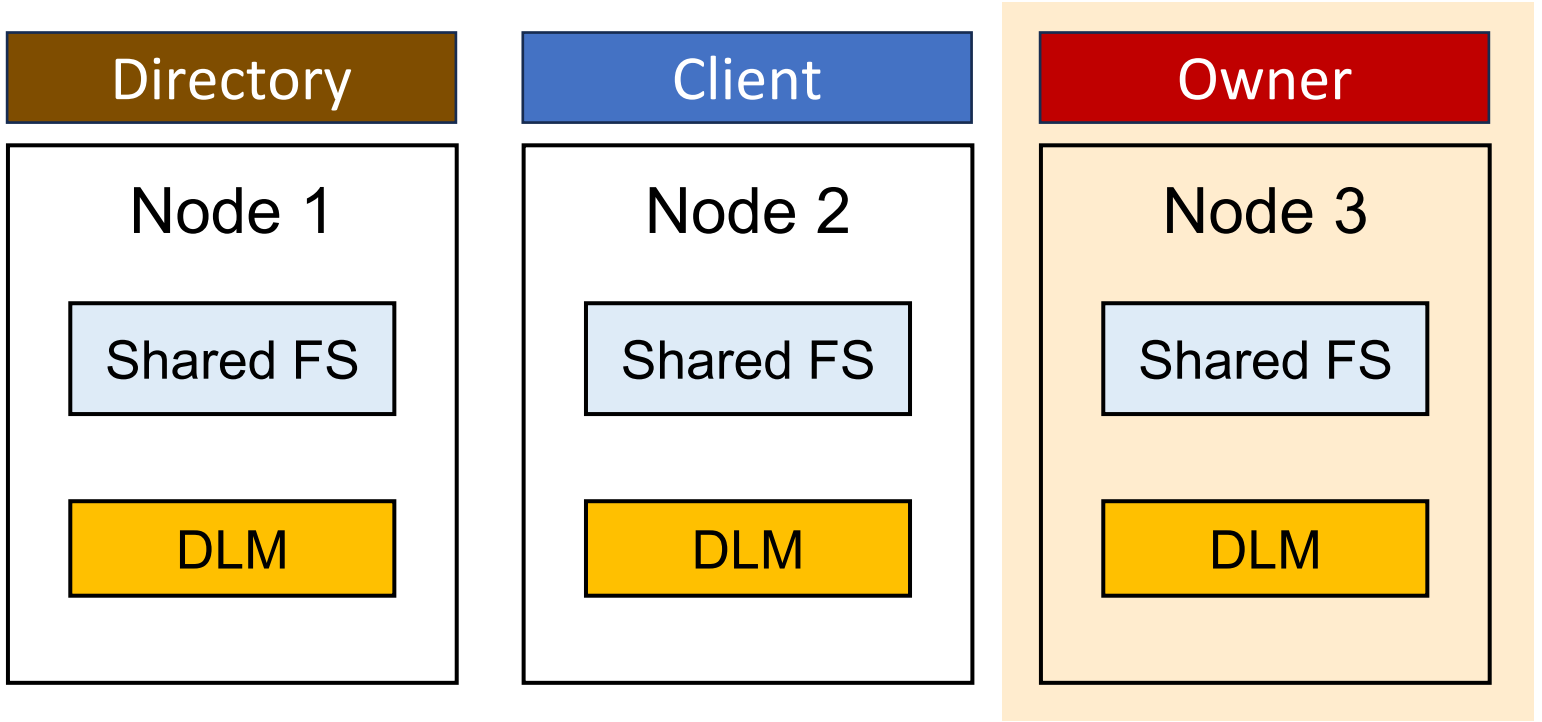
– Owner Node and Directory Node



Distributed Lock Manager (DLM)

Owner Node

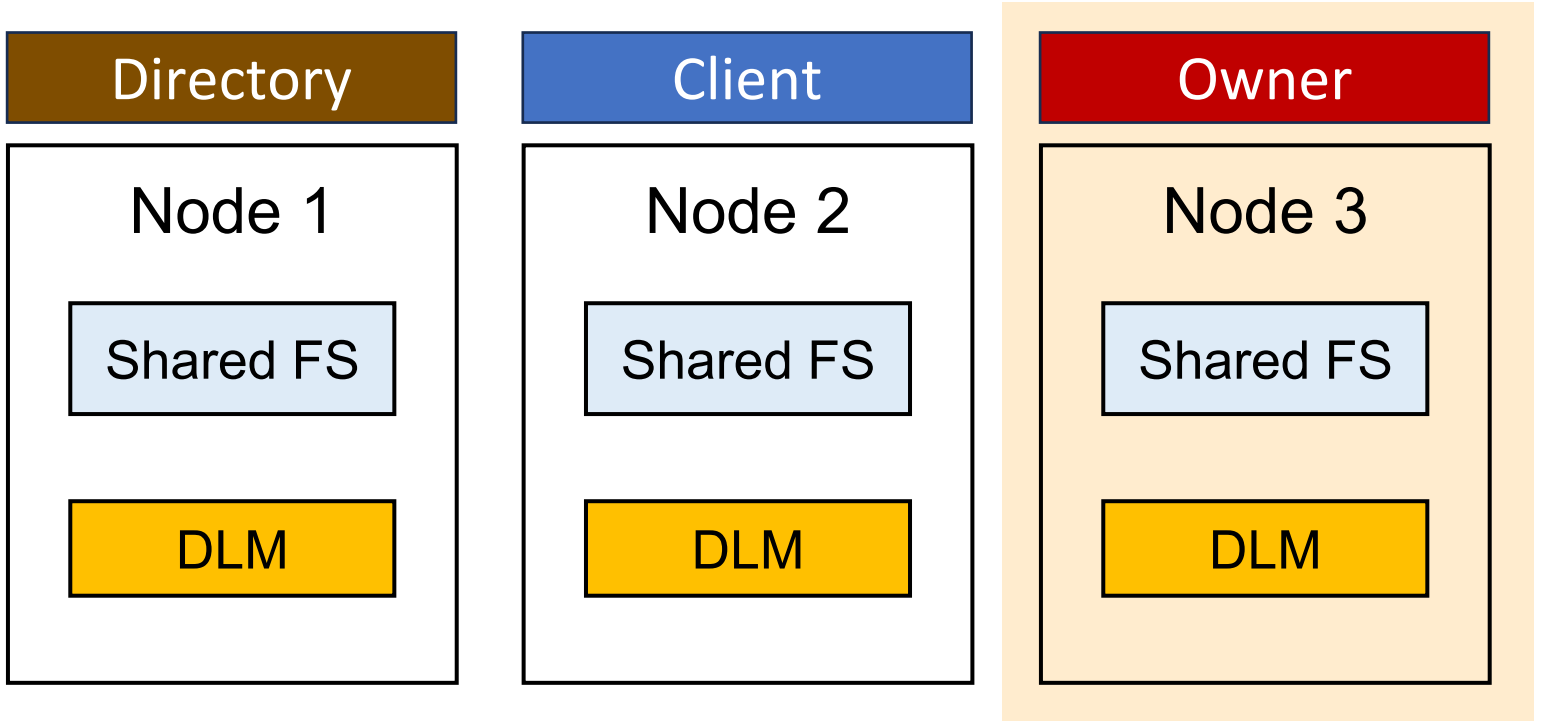
– Node that has ownership of the lock on a specific object



Distributed Lock Manager (DLM)

Owner Node

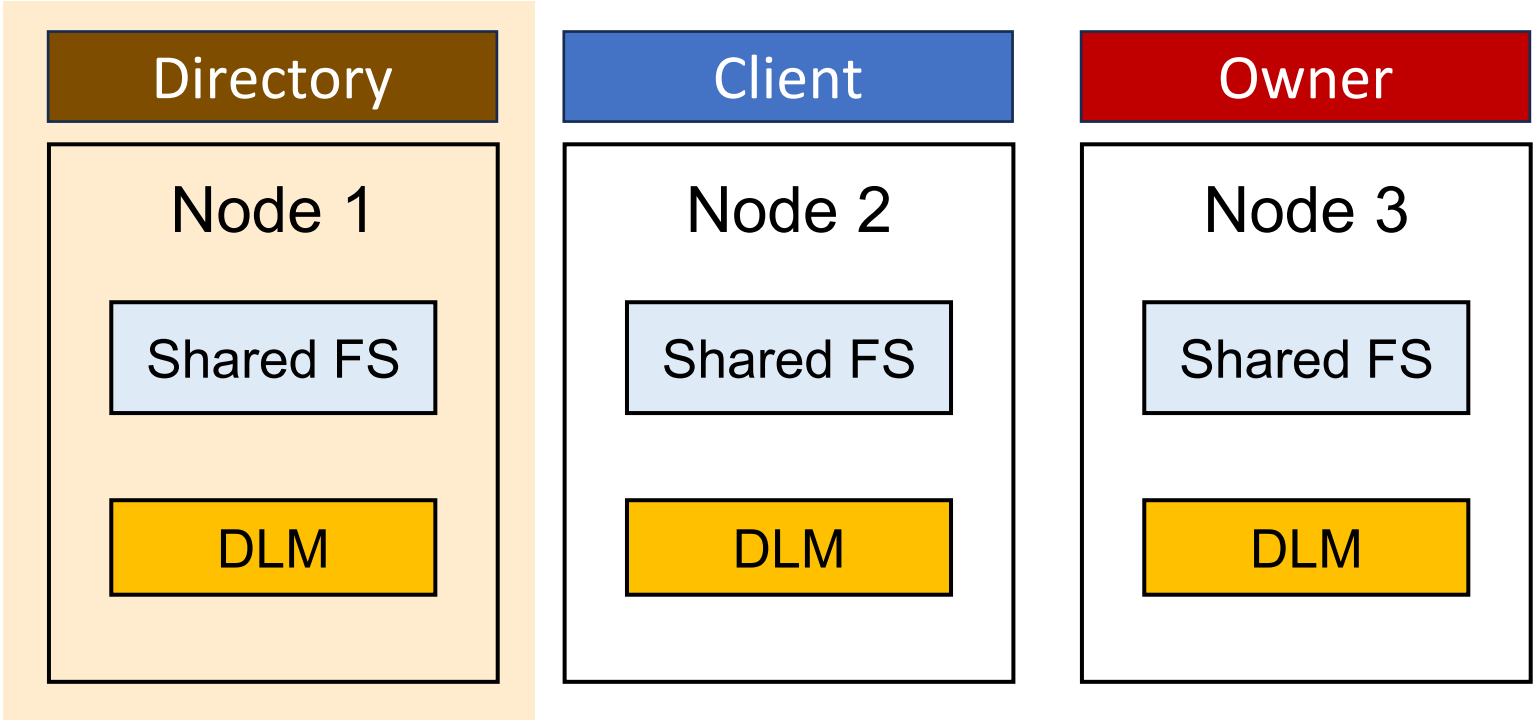
– Q: How do we identify Owner Node?



Distributed Lock Manager (DLM)

Directory Node

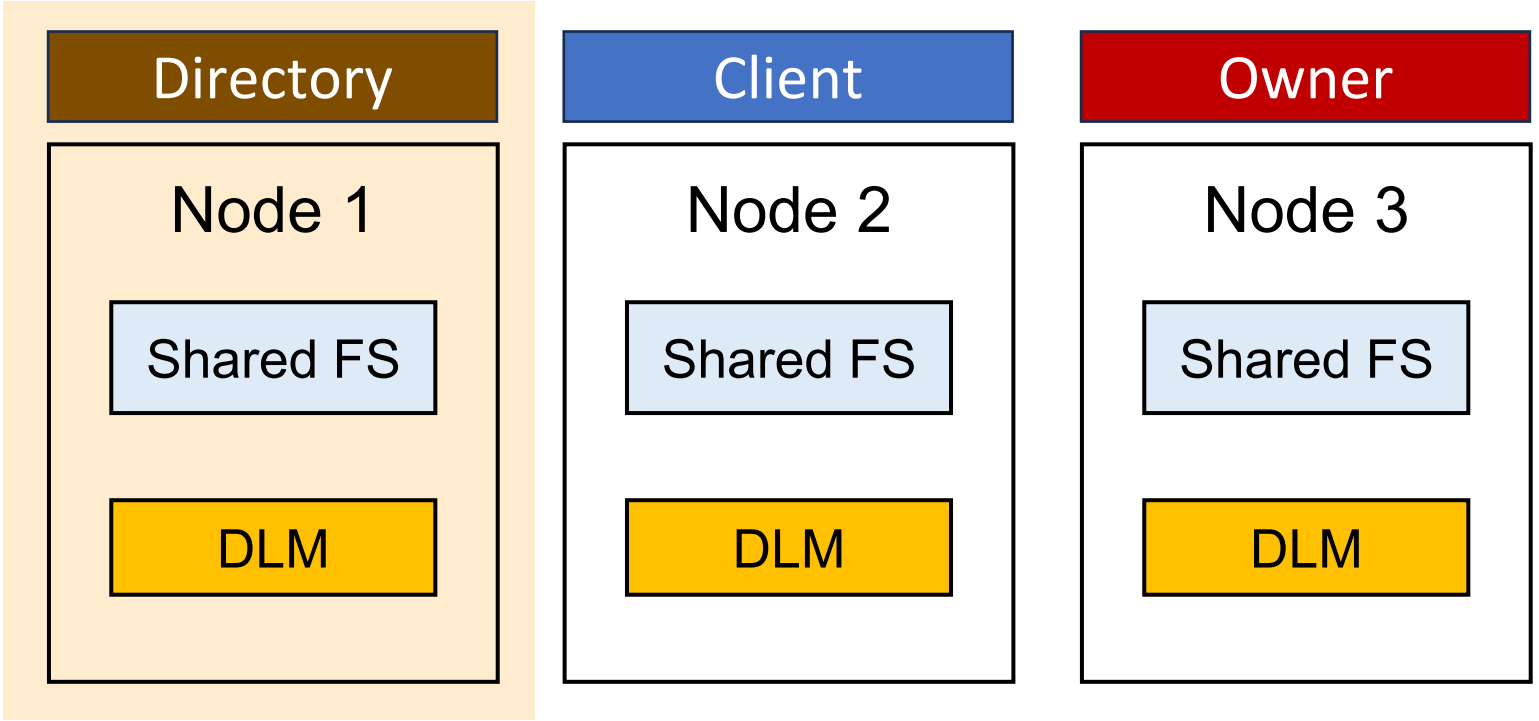
– A: Directory Node maintains the ownership mapping



Distributed Lock Manager (DLM)

Directory Node

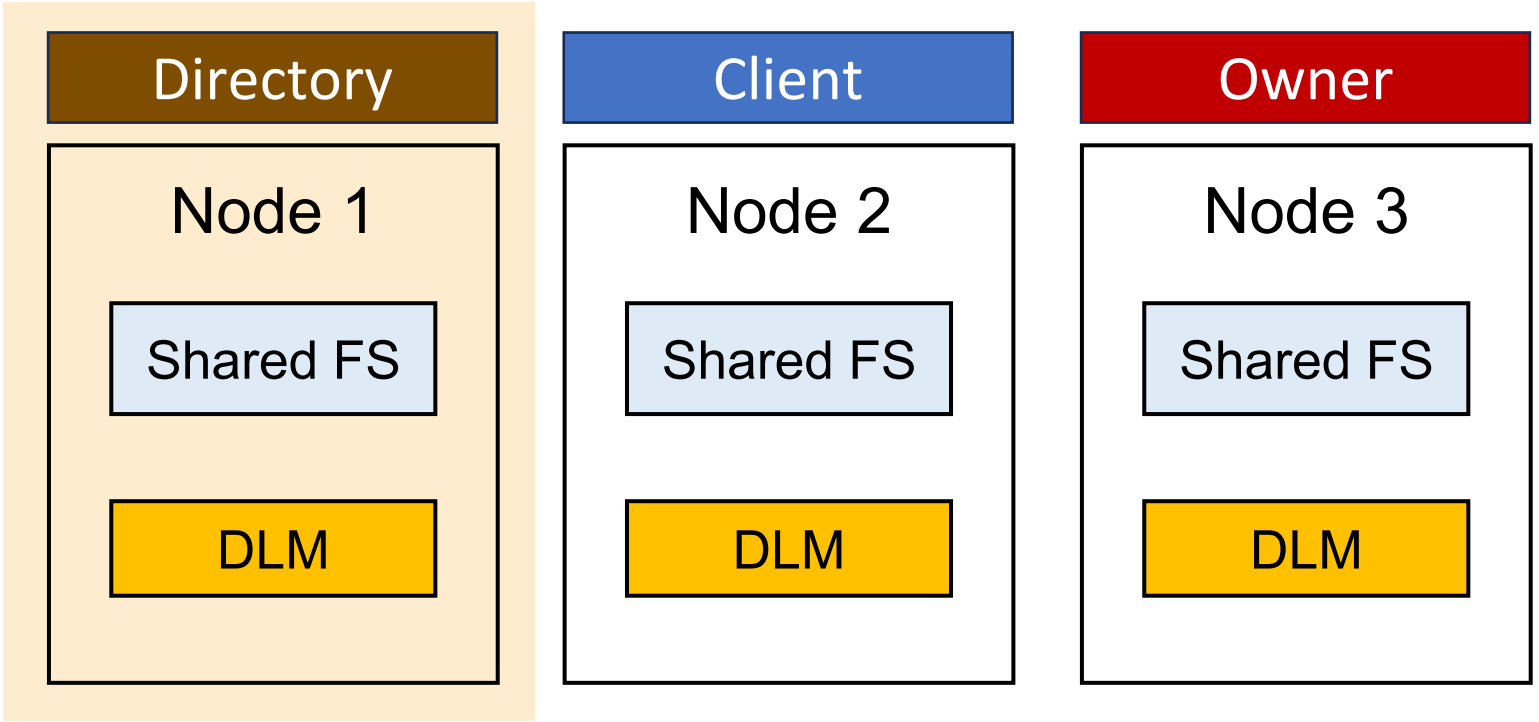
– Q: How do we identify Directory Node?



Distributed Lock Manager (DLM)

Directory Node

– A: Using the object's hash value!



Distributed Lock Manager (DLM)

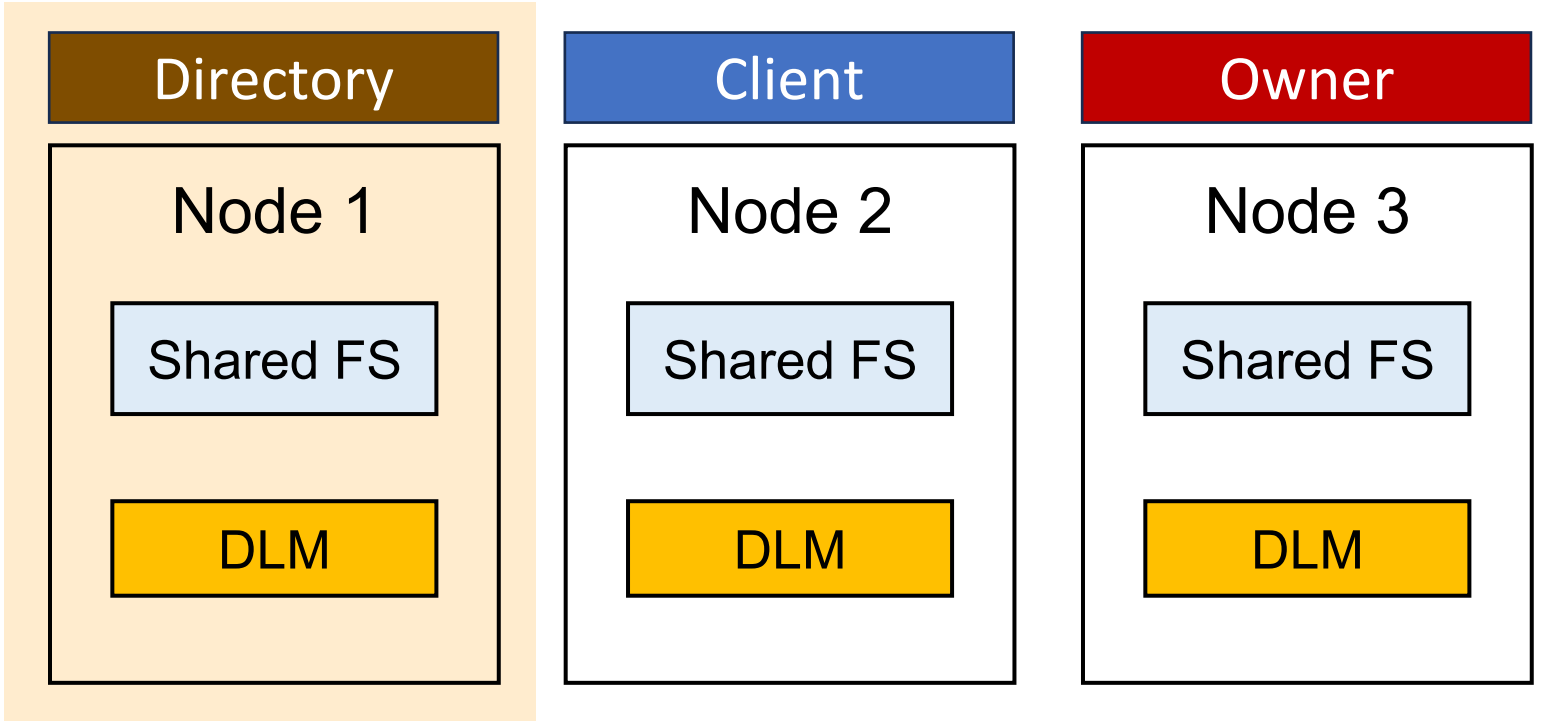
Directory Node

– A: Using the object's hash value!

Hash(object) % #nodes

If Hash(object) = 10
#nodes = 3

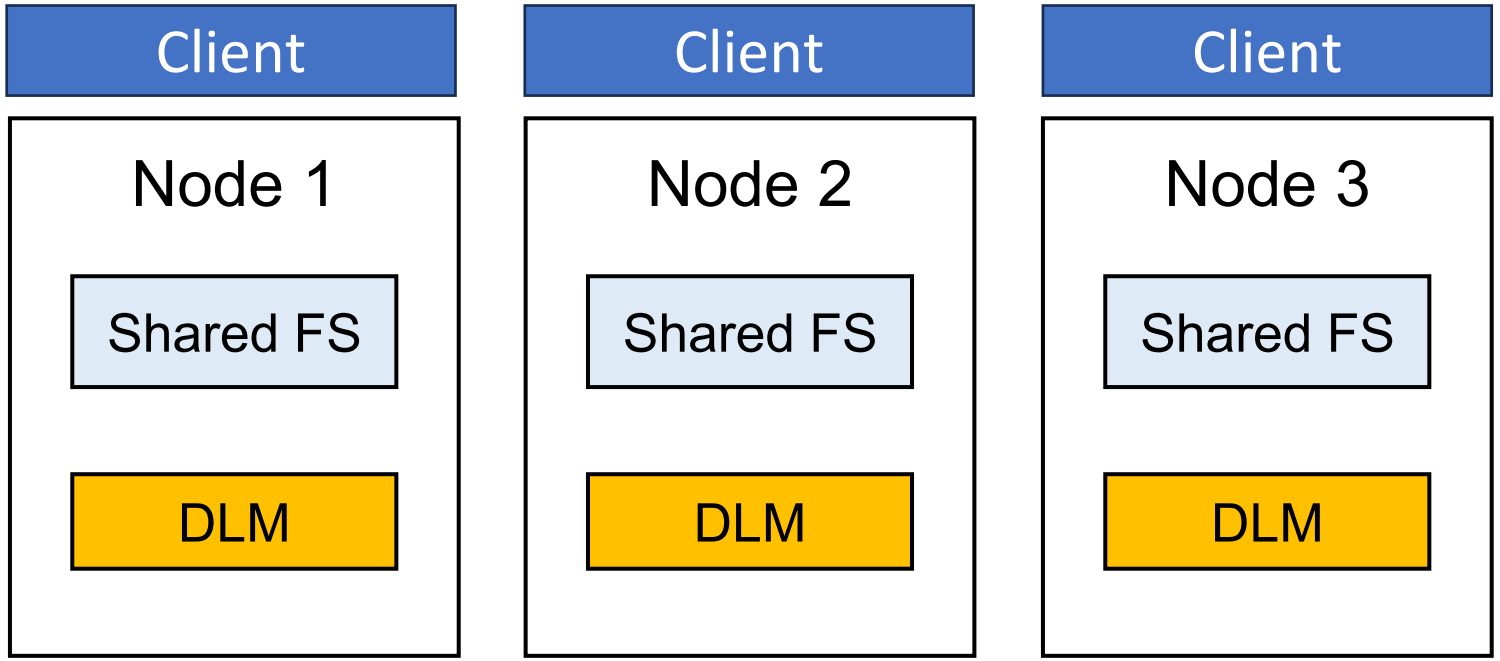
Directory = 10%3 = Node 1



Distributed Lock Manager (DLM)

Fully distributed structure

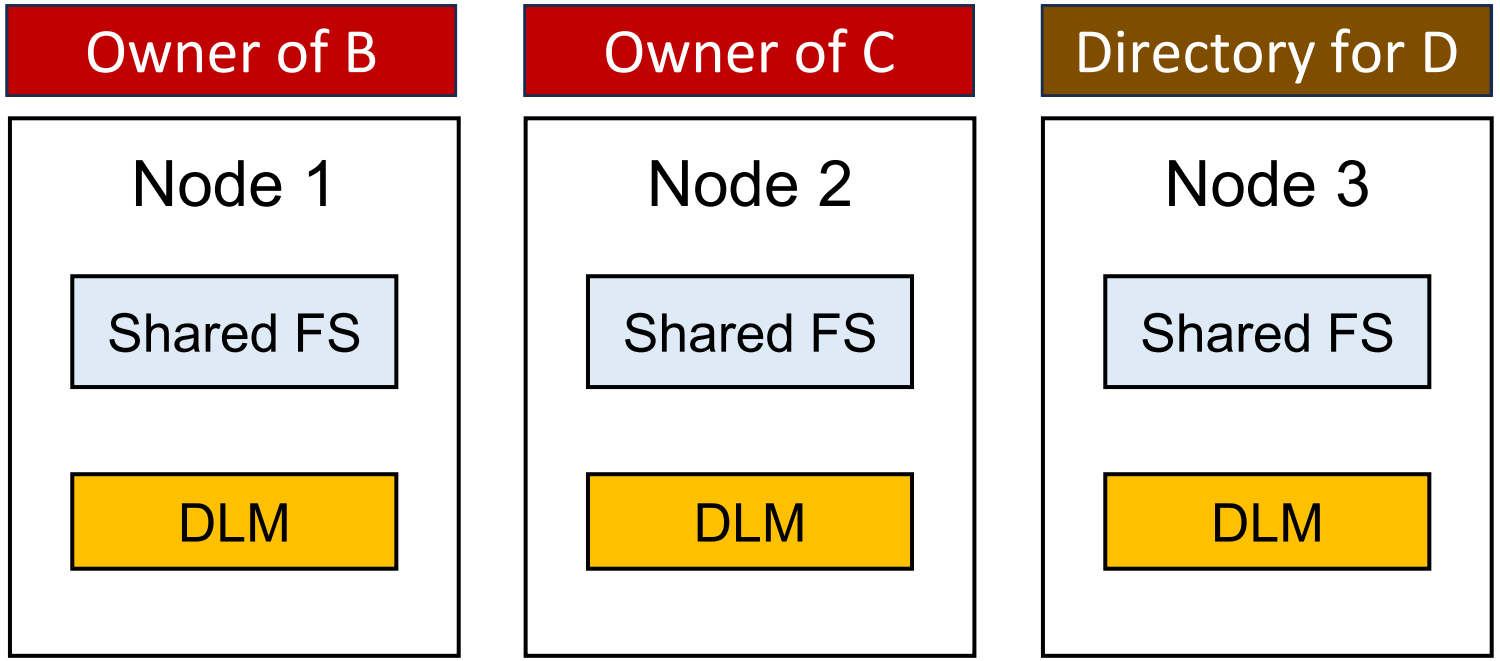
– Every node acts as a client



Distributed Lock Manager (DLM)

Fully distributed structure

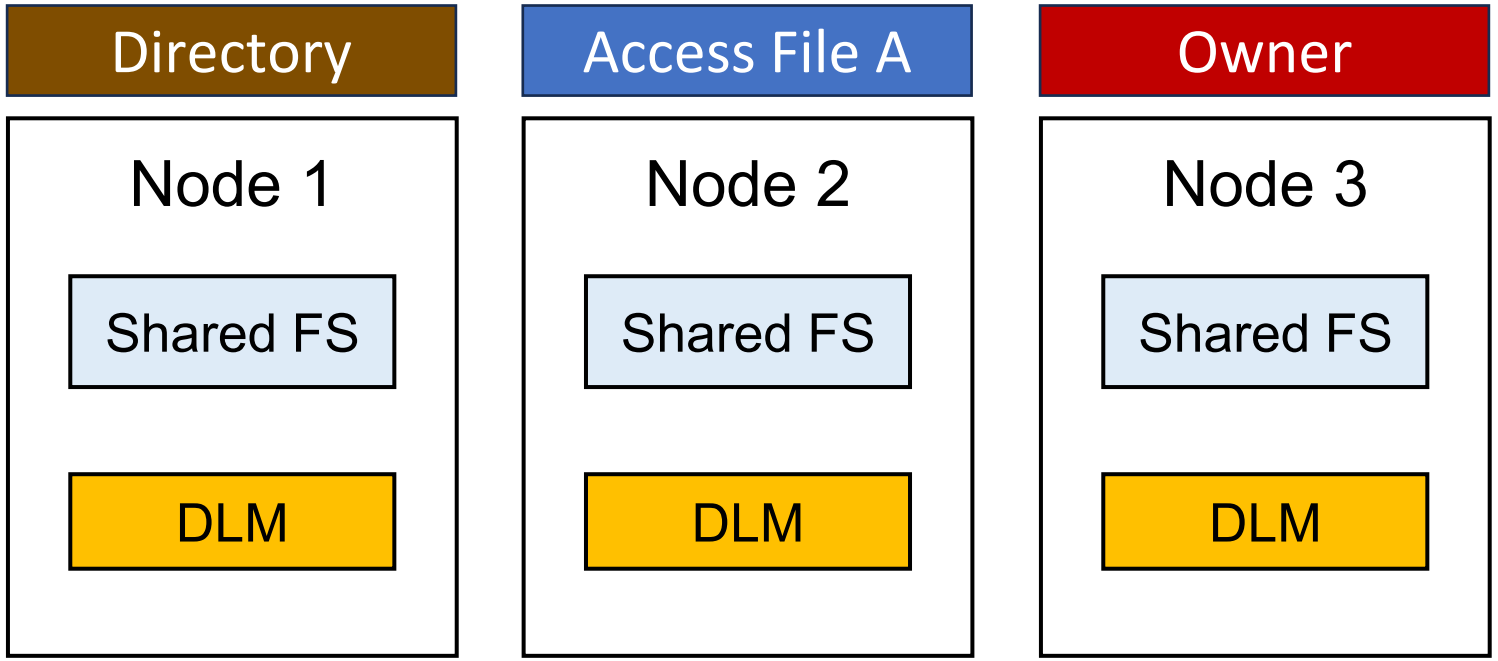
– Any node can act as Owner or Directory for an object



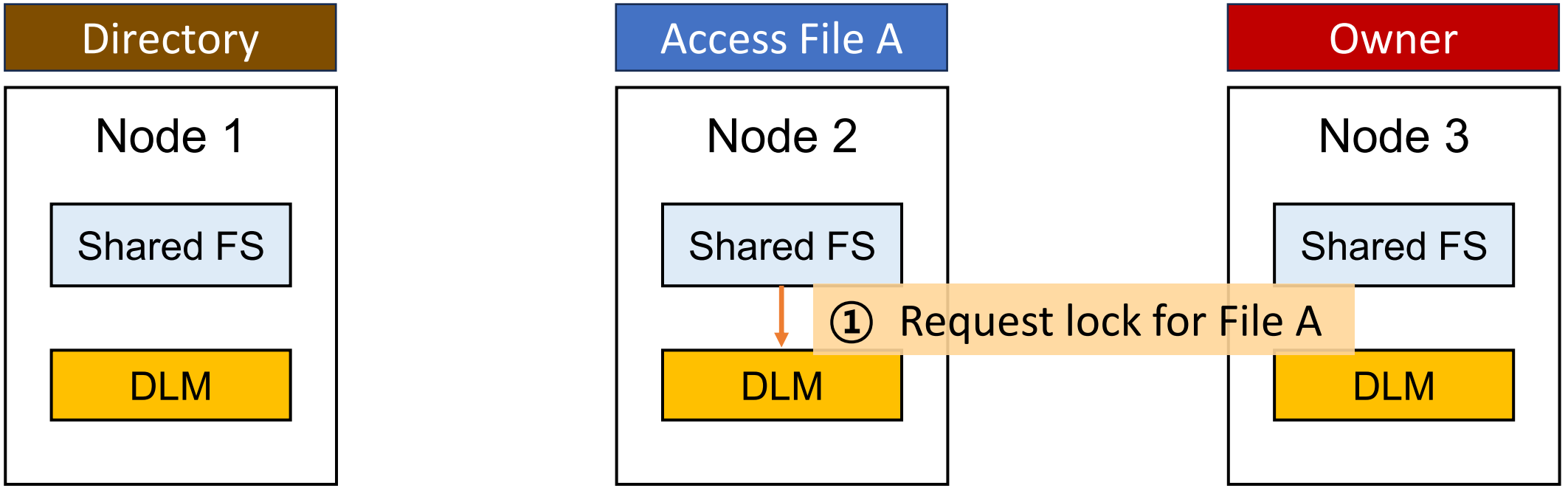
DLM Example

Example of Lock Acquisition

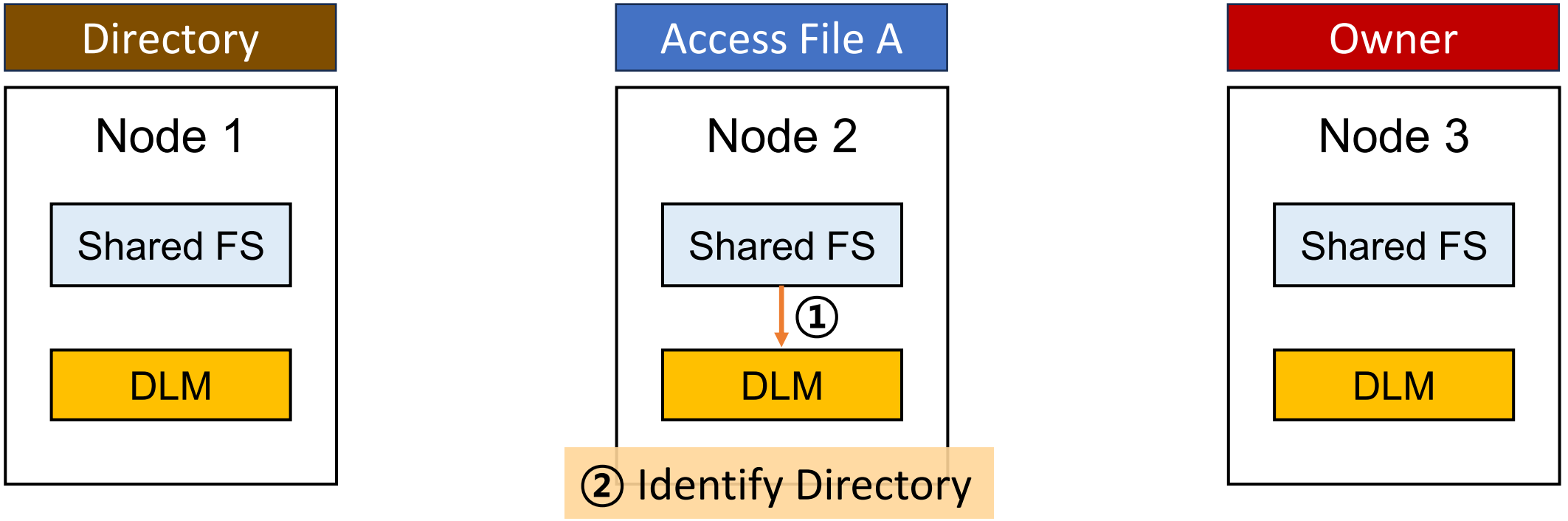
– Node 2 requests a lock on File A



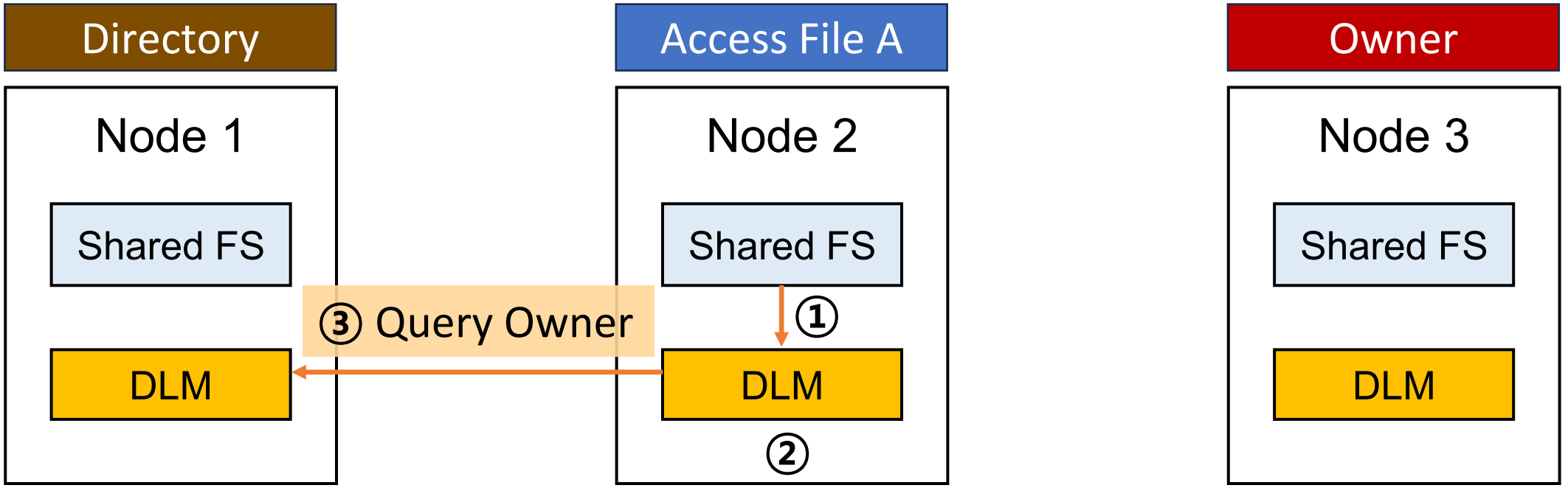
DLM Example


Directory
Access File A
Owner
Node 1
Shared FS
DLM
Node 2
Shared FS
DLM
Node 3
Shared FS
DLM
① Request lock for File A

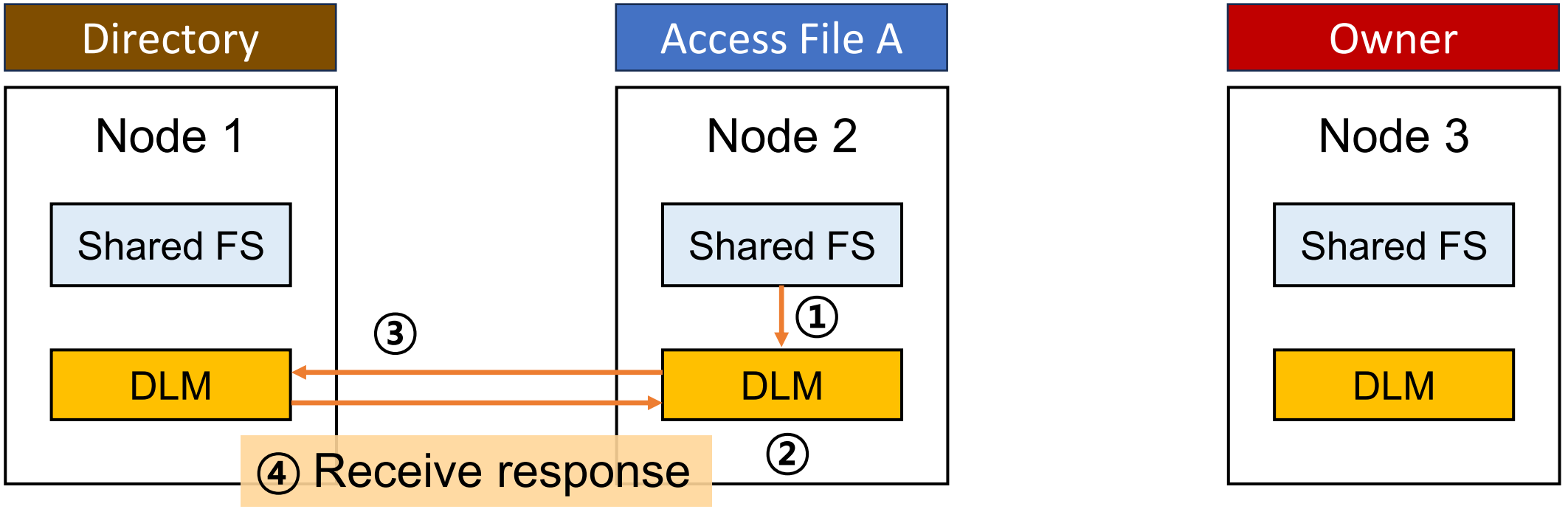
DLM Example


Directory
Access File A
Owner
Node 1
Shared FS
DLM
Node 2
Shared FS
①
DLM
② Identify Directory
Node 3
Shared FS
DLM

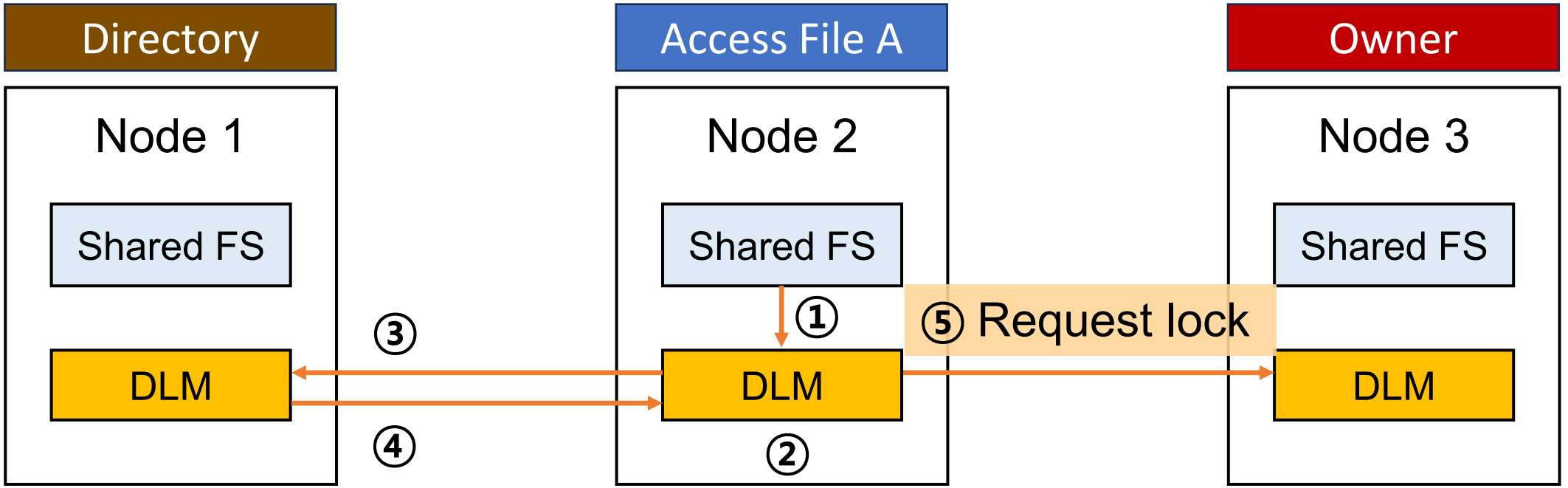
DLM Example


Directory
Access File A
Owner
Node 1
Node 2
Node 3
Shared FS
Shared FS
Shared FS
③ Query Owner
①
DLM
DLM
DLM
②

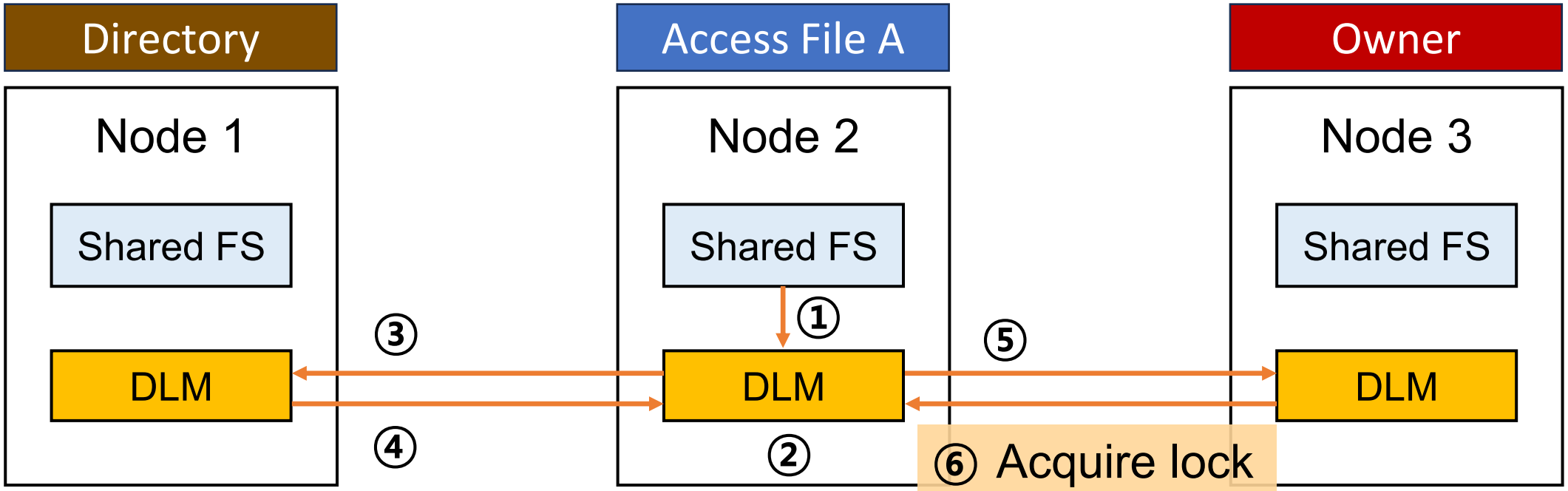
DLM Example


Directory
Access File A
Owner
Node 1
Node 2
Node 3
Shared FS
Shared FS
Shared FS
DLM
DLM
DLM
①
②
③
④ Receive response

DLM Example


Directory
Access File A
Owner
Node 1
Node 2
Node 3
Shared FS
Shared FS
Shared FS
①
⑤ Request lock
③
DLM
DLM
DLM
④
②

DLM Example


Directory
Access File A
Owner
Node 1
Node 2
Node 3
Shared FS
Shared FS
Shared FS
DLM
DLM
DLM
①
②
③
④
⑤
⑥ Acquire lock

Motivation

High Contention = Low Performance

Motivation

Low Contention = High Performance?

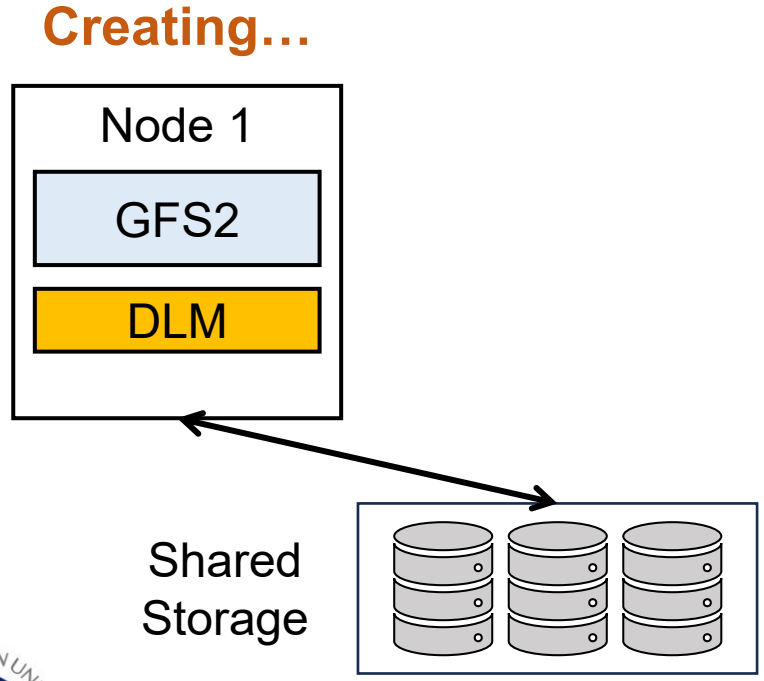
Motivation

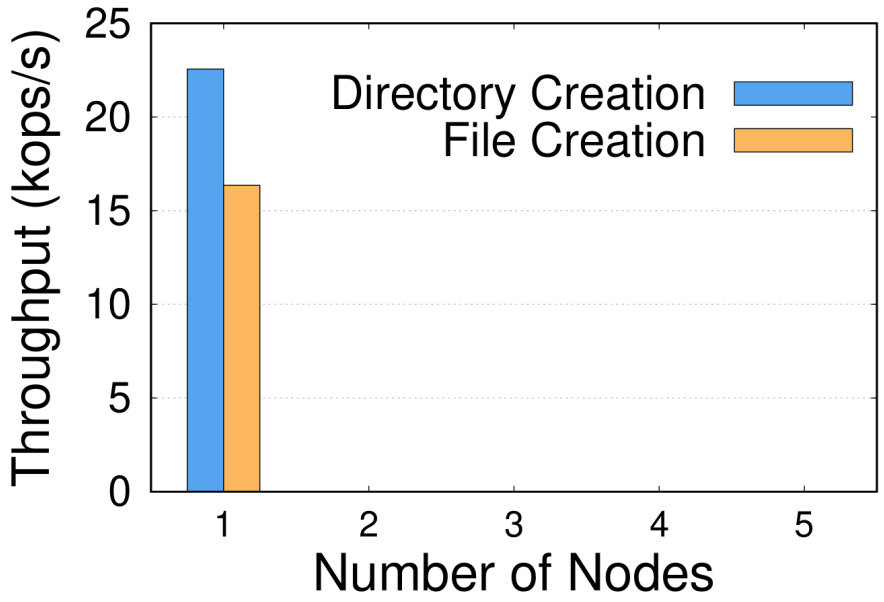
Problem: Low Contention ≠ High Performance

Motivation

Problem: Low Contention ≠ High Performance

– Only one node is creating files and directories

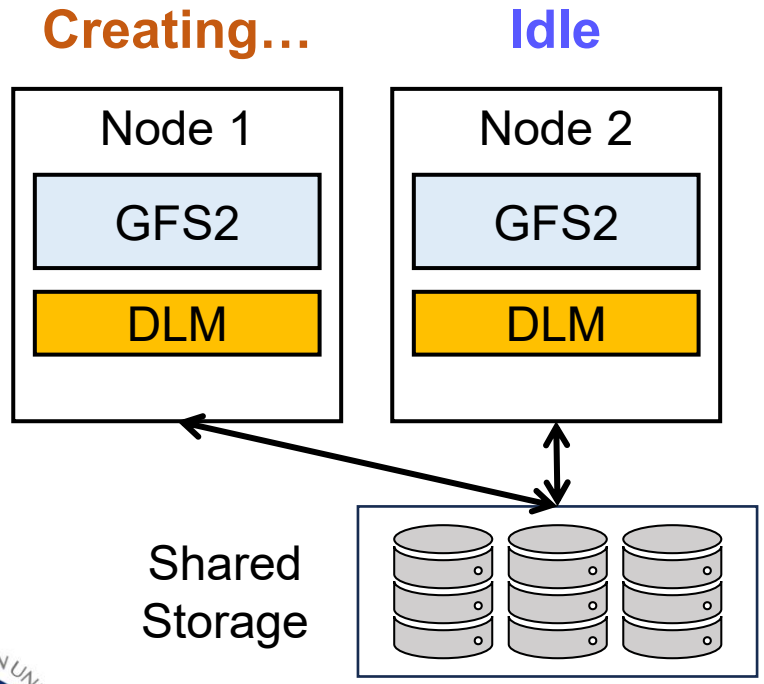


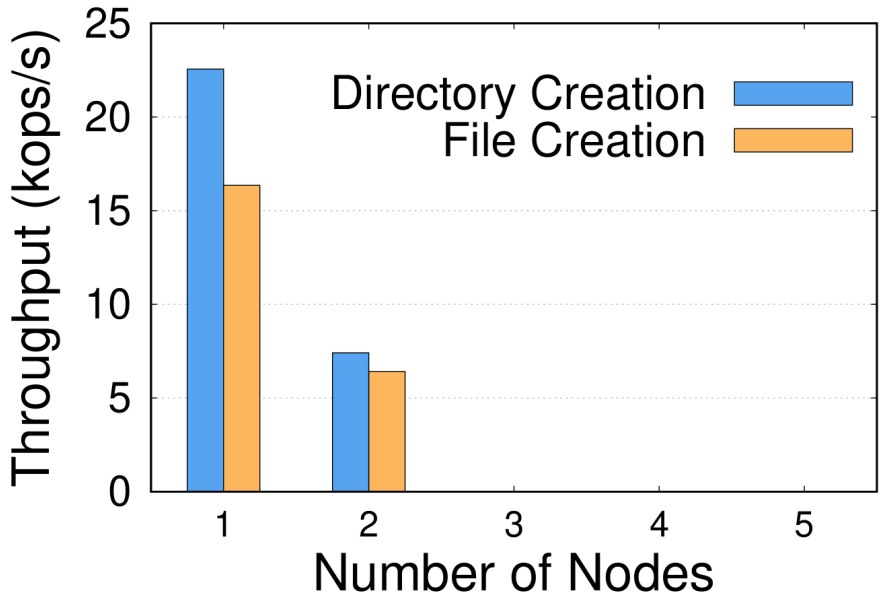


Motivation

Problem: Low Contention ≠ High Performance

– Performance degradation with additional idle nodes

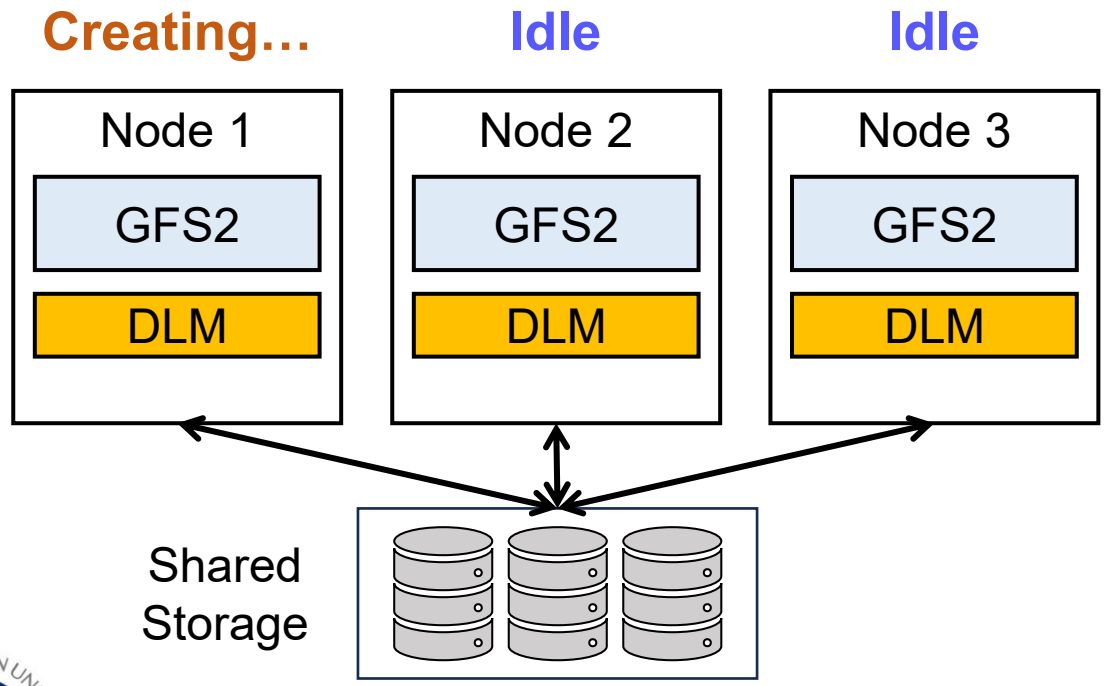


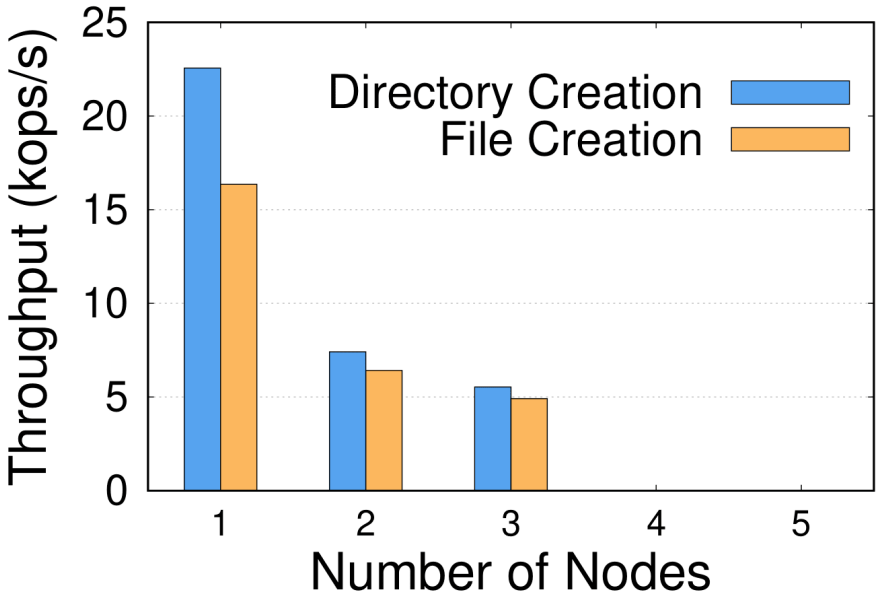


Motivation

Problem: Low Contention ≠ High Performance

– Performance degradation with additional idle nodes



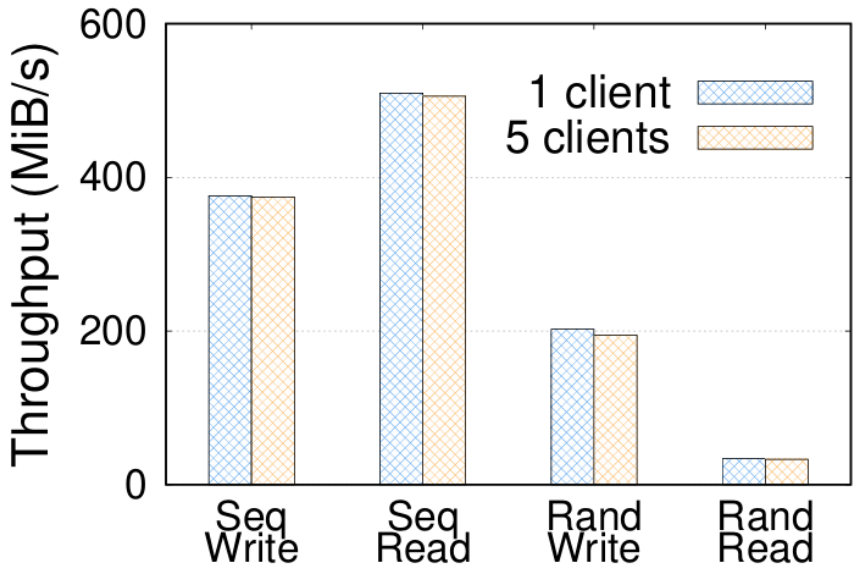


Motivation

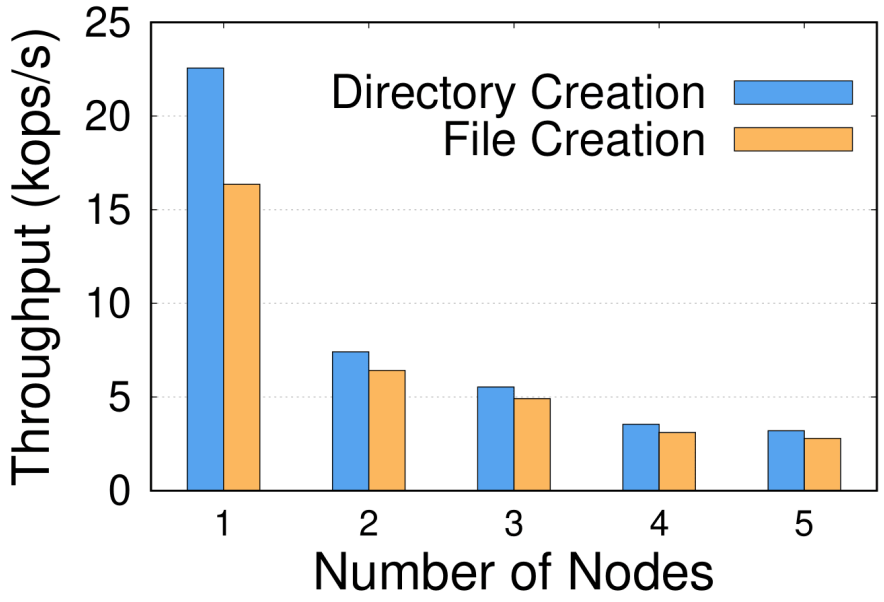
Problem: Low Contention ≠ High Performance

– Especially in creation

Bottleneck: IO



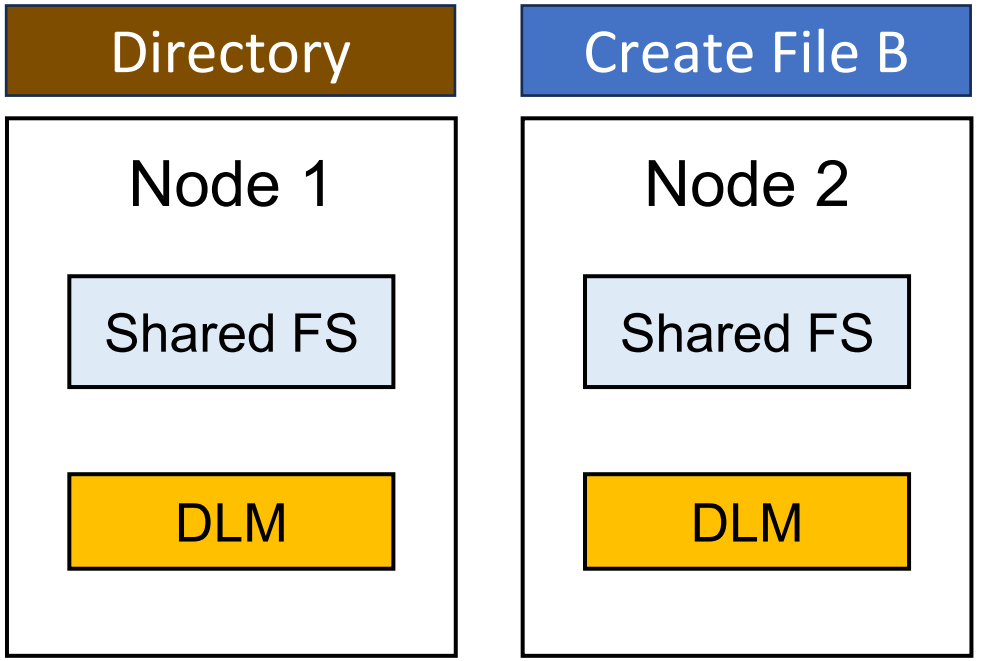
Bottleneck: DLM



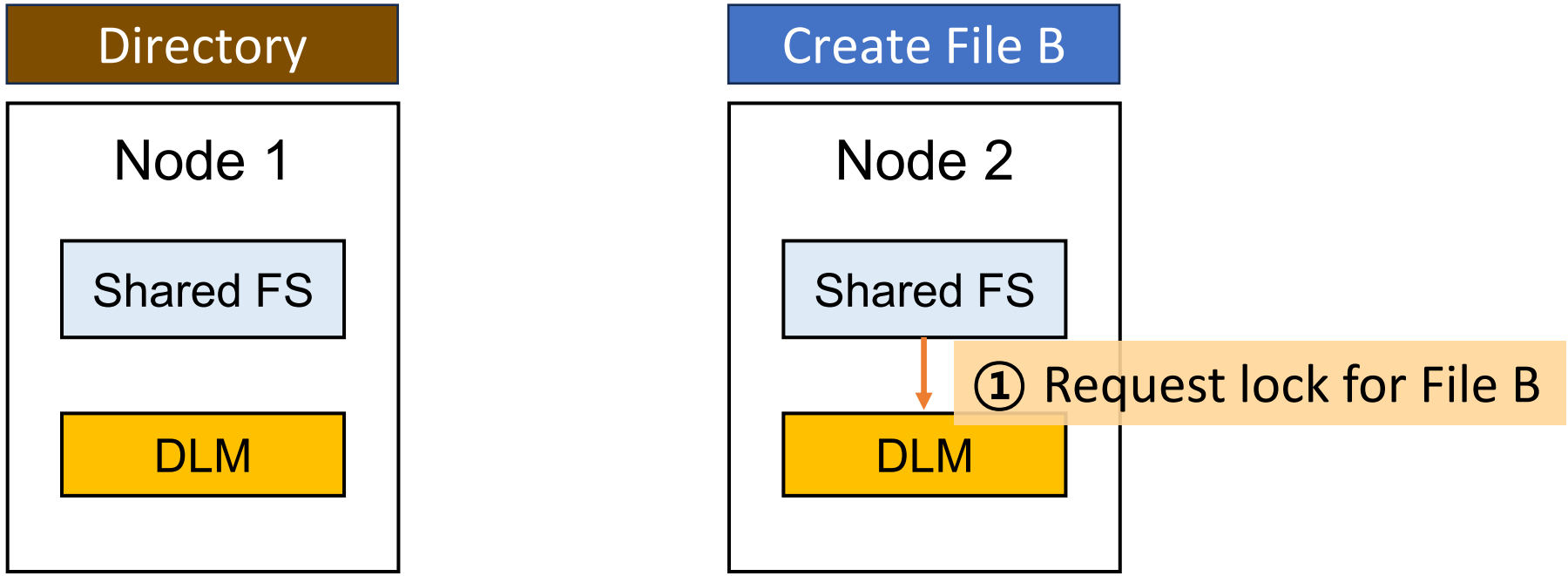
DLM Operation in Creation

Example of creation

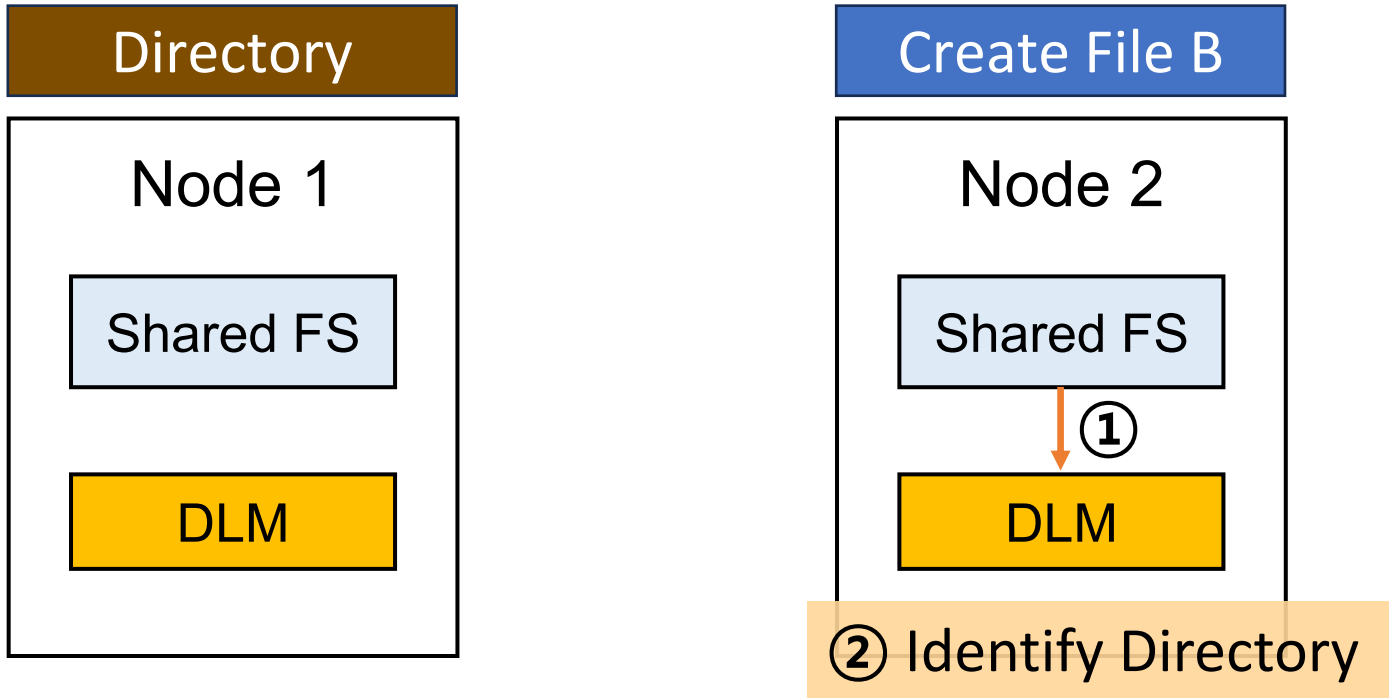
– Node 2 requests lock on newly created File B



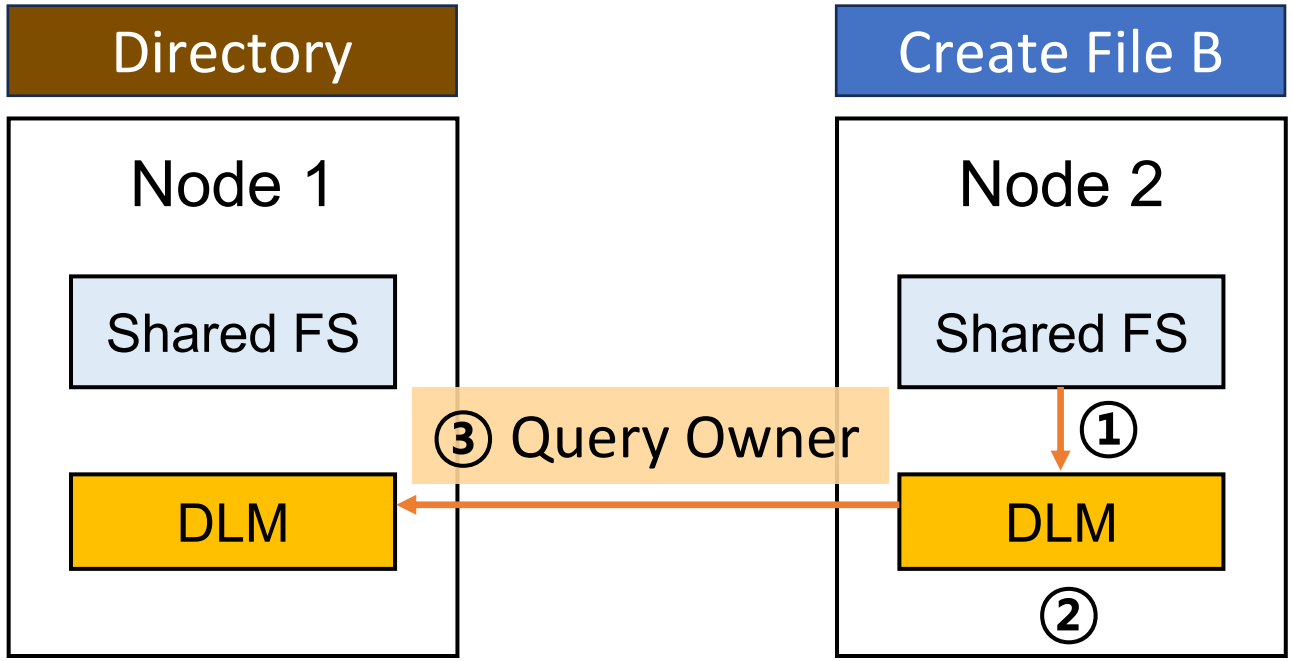
DLM Operation in Creation



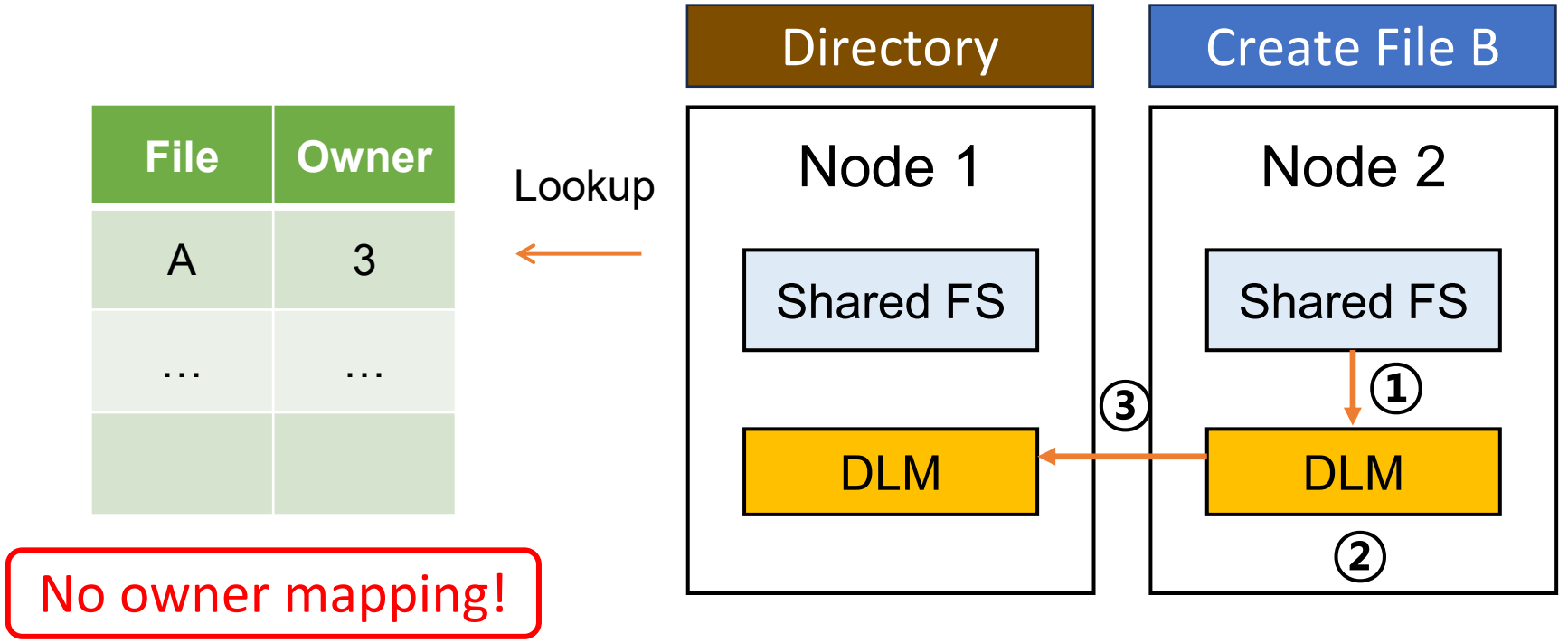
DLM Operation in Creation



DLM Operation in Creation



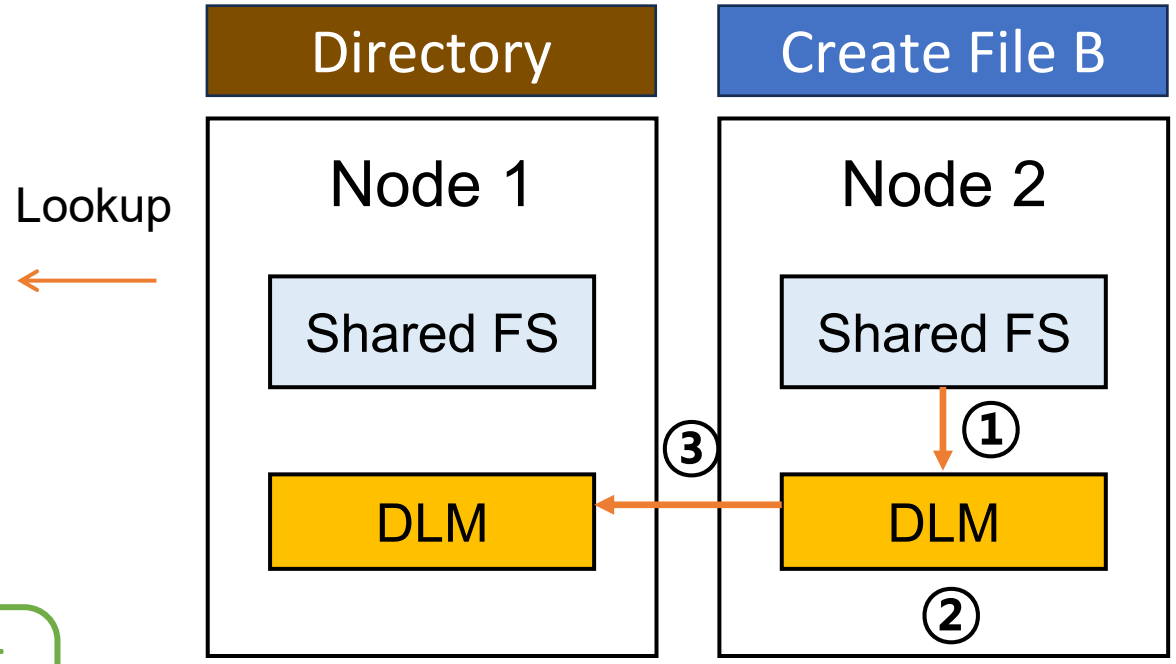
DLM Operation in Creation



DLM Operation in Creation

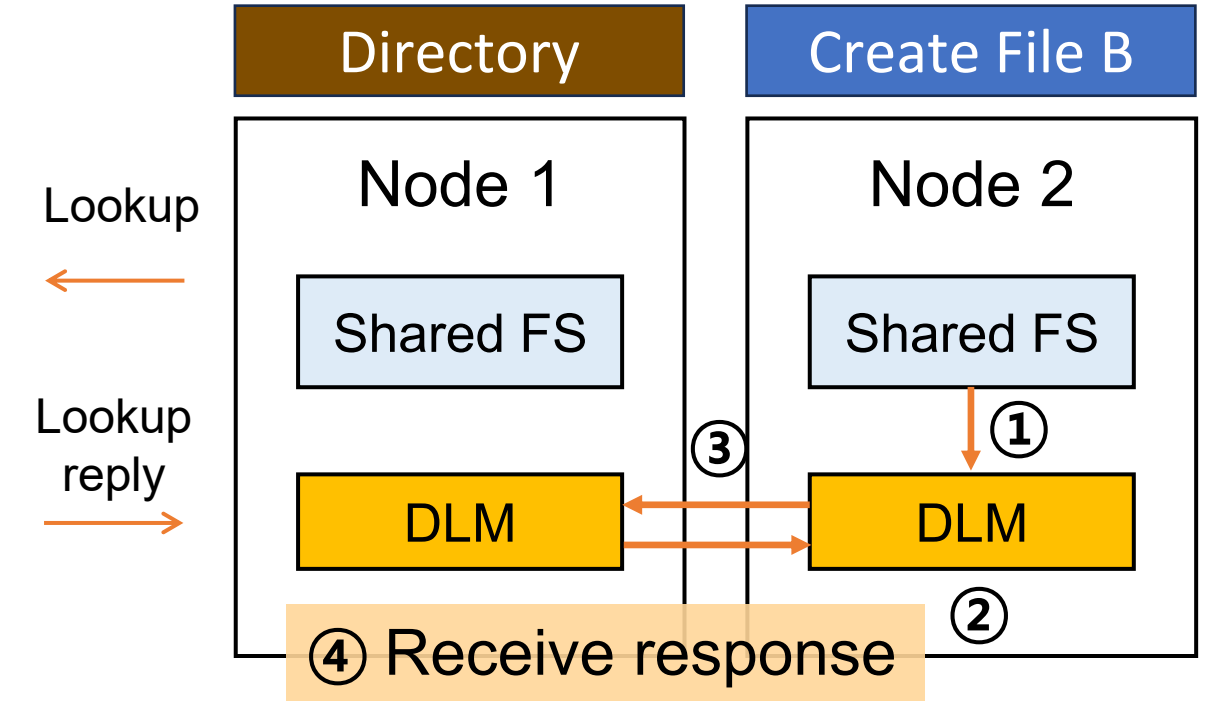
File	Owner
A	3
…	…
B	2

Assign requesting node as owner!



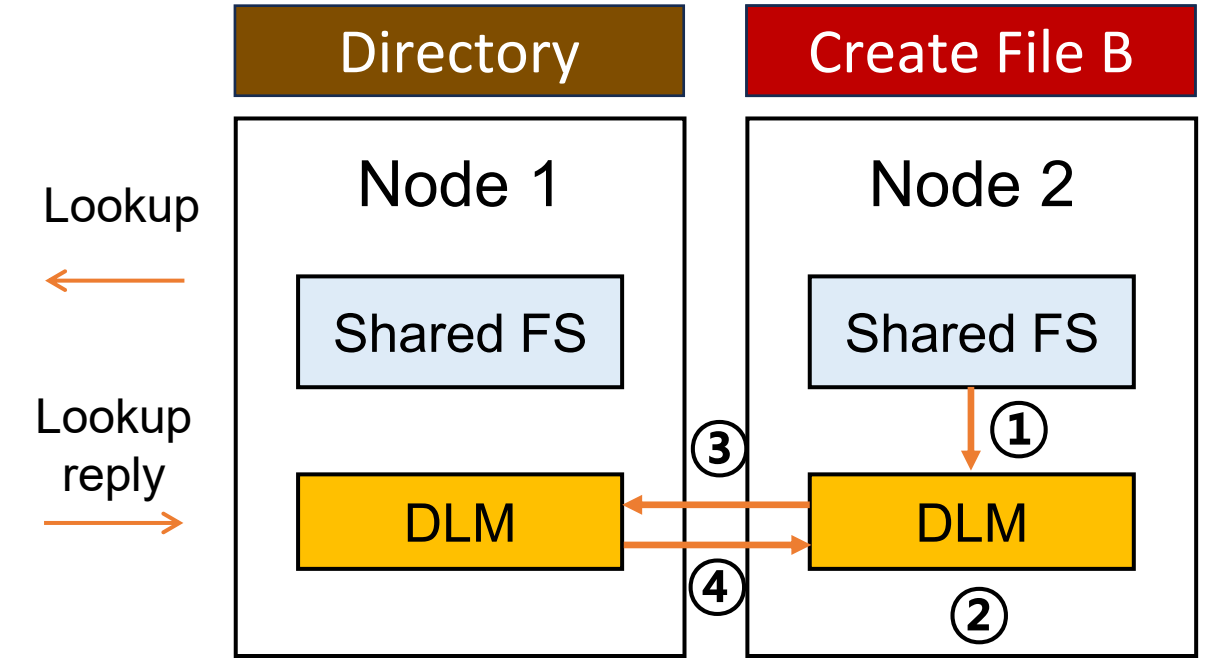
DLM Operation in Creation

File	Owner
A	3
…	…
B	2



DLM Operation in Creation

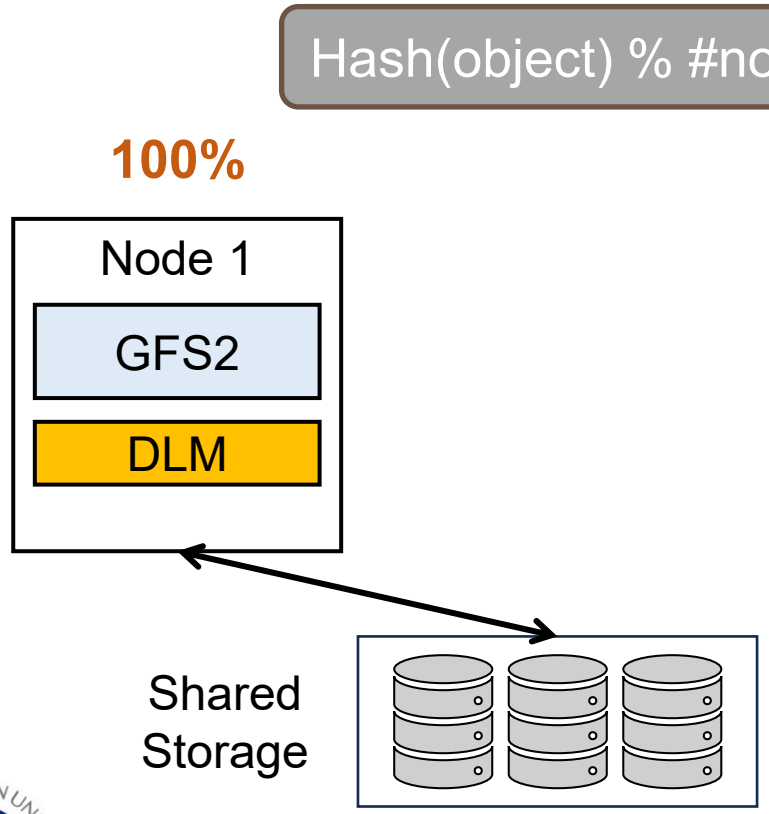
File	Owner
A	3
…	…
B	2



⑤, ⑥ Request and acquire lock **in local**

DLM Operation in Creation

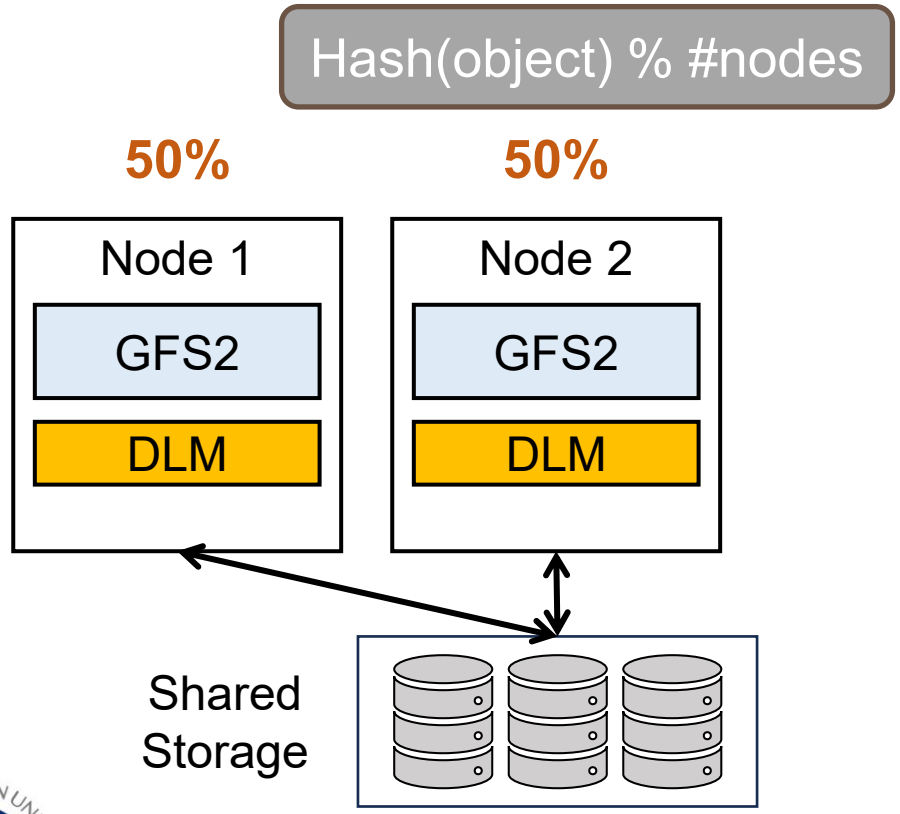
Root Cause: Accessing remote directory node overhead



Remote directory
0%

DLM Operation in Creation

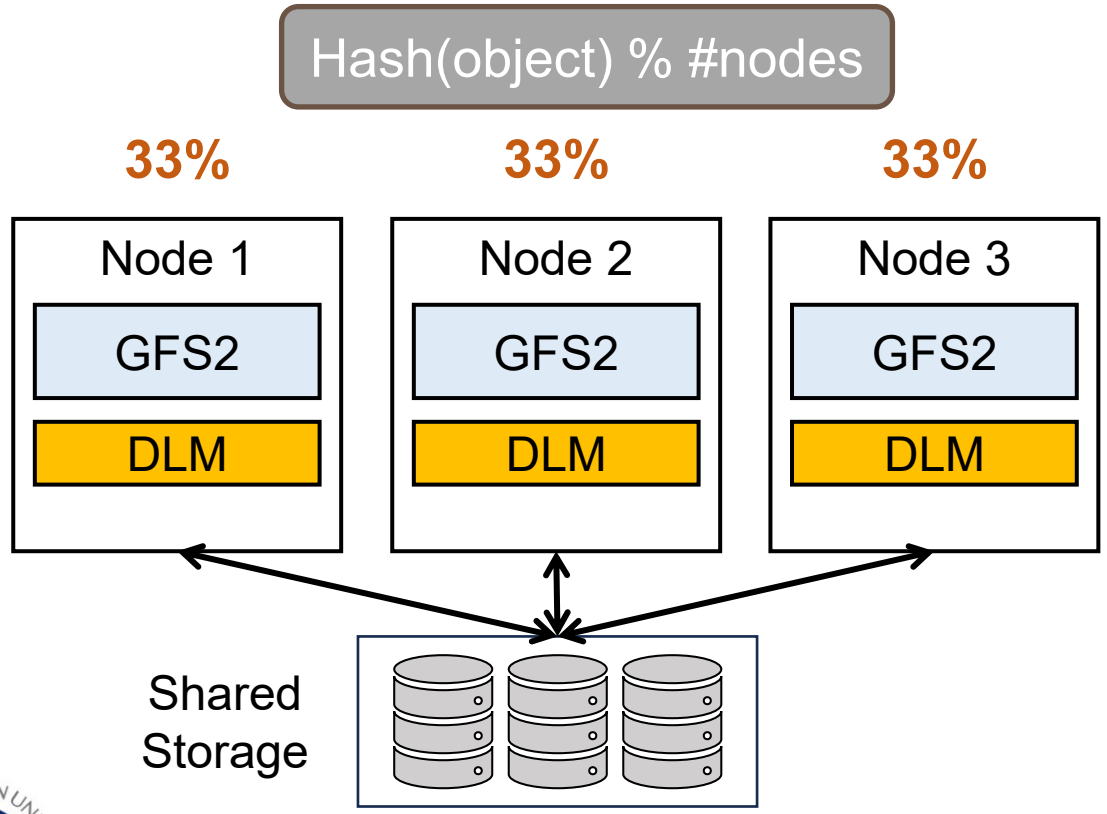
Root Cause: Accessing remote directory node overhead



Remote directory
50%

DLM Operation in Creation

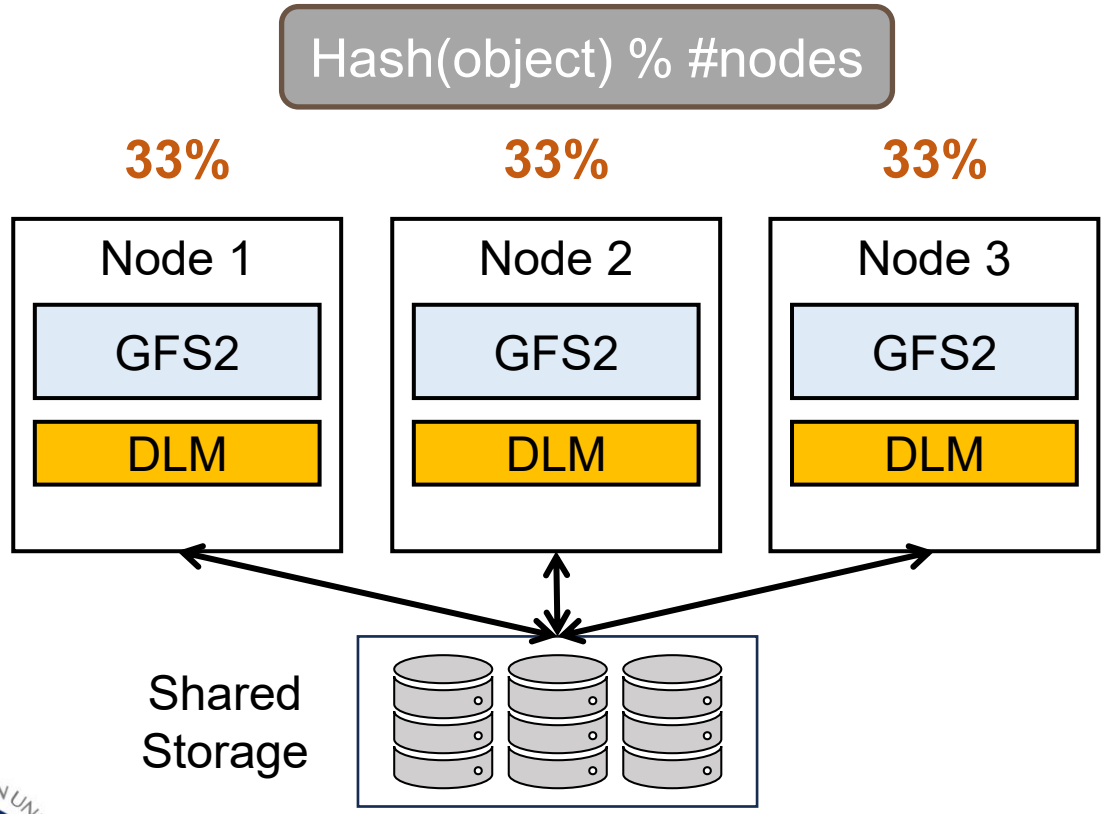
Root Cause: Accessing remote directory node overhead

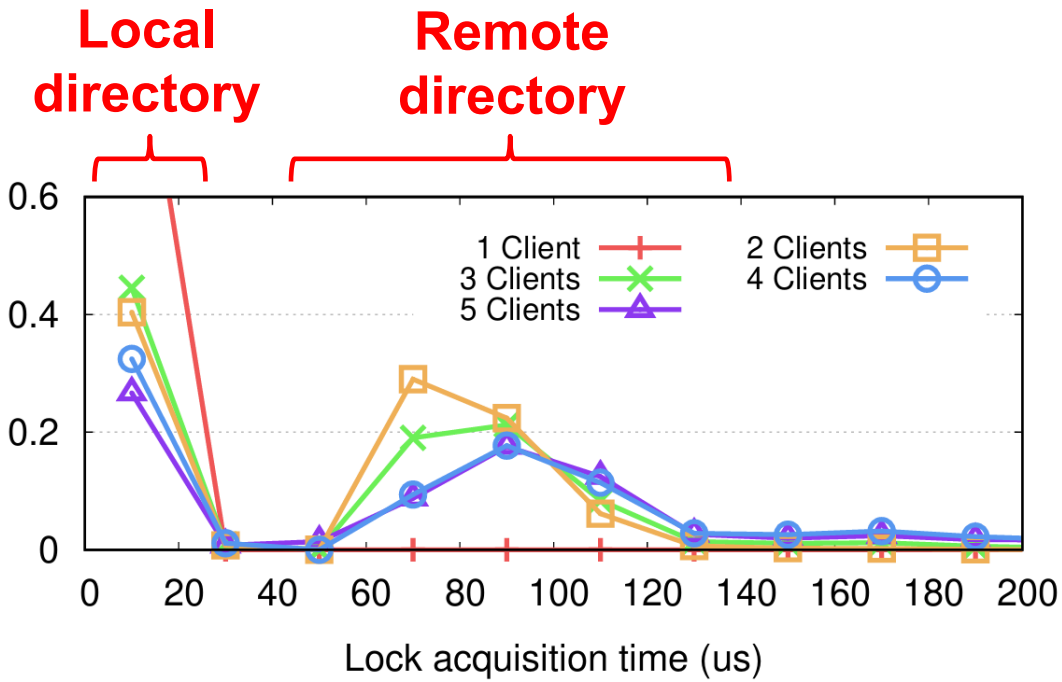


Remote directory
66%

DLM Operation in Creation

Root Cause: Accessing remote directory node overhead

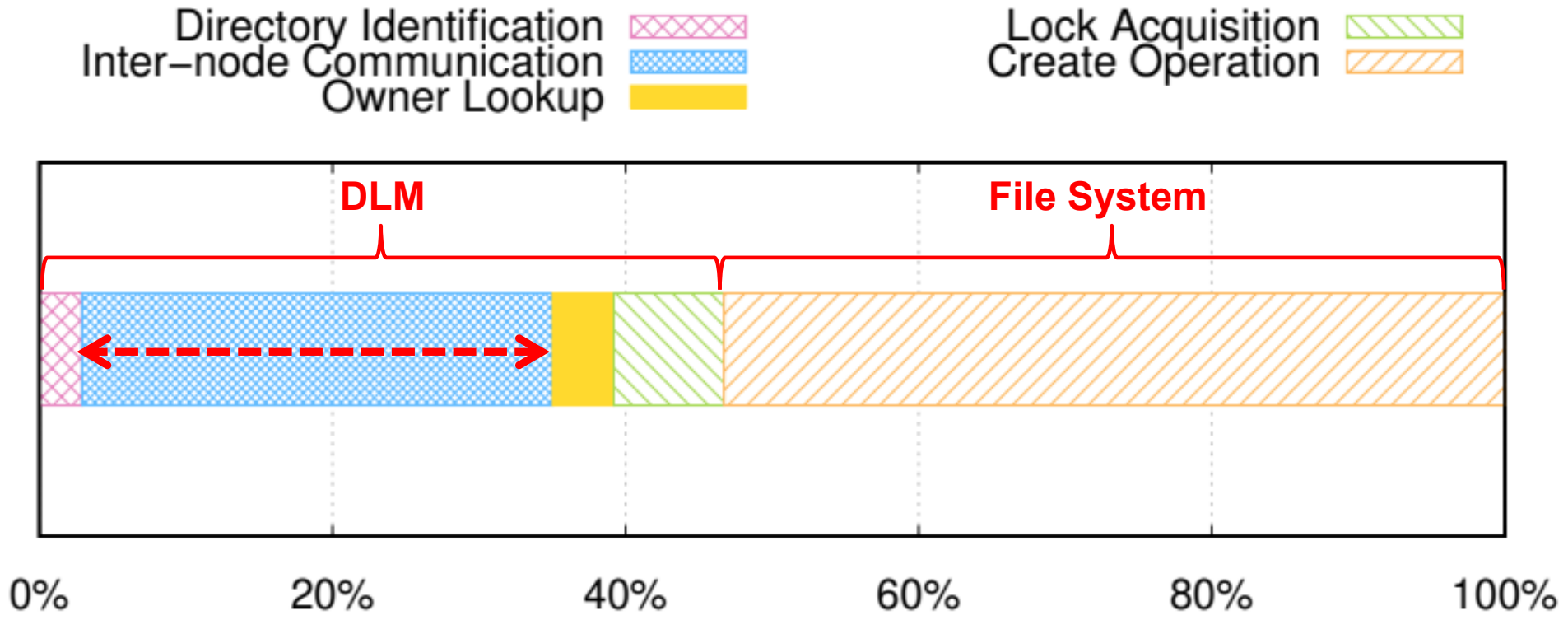




DLM Operation in Creation

Root Cause: Accessing remote directory node overhead

– 5 clients latency breakdown



Motivation

Review motivation: In creation scenario,

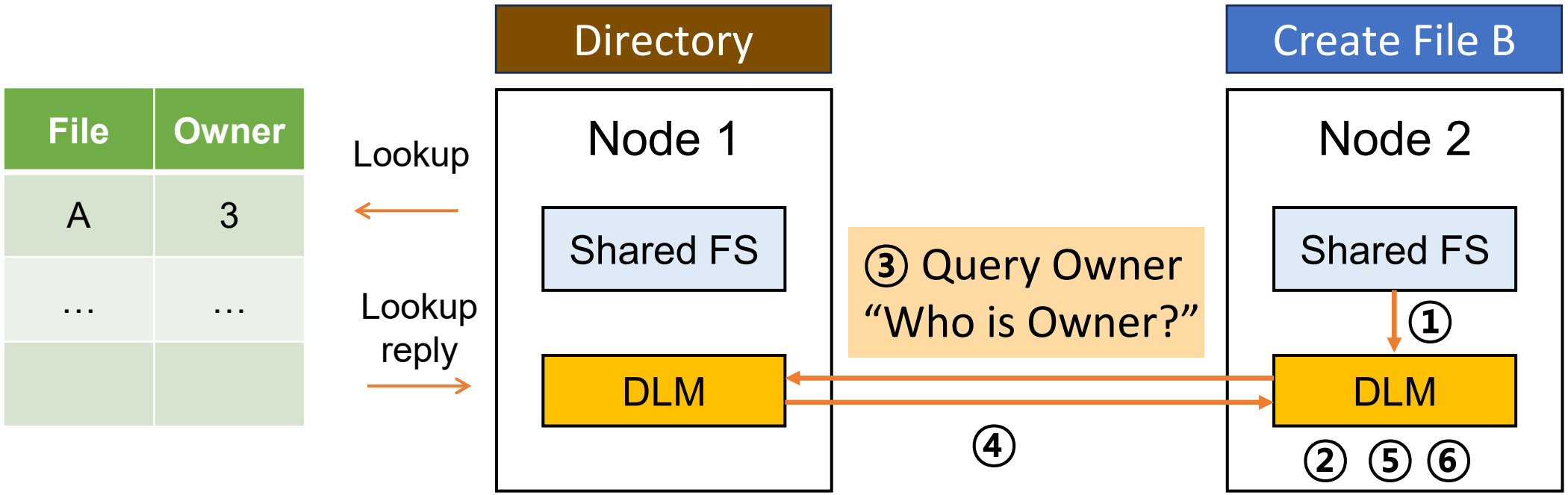
1. Assign requesting node as owner
2. Accessing remote directory node is main overhead

Our solution: Lockify

1. Self-Owner Notifications
2. Extended Lock Acquisition Interface
3. Asynchronous Ownership Management

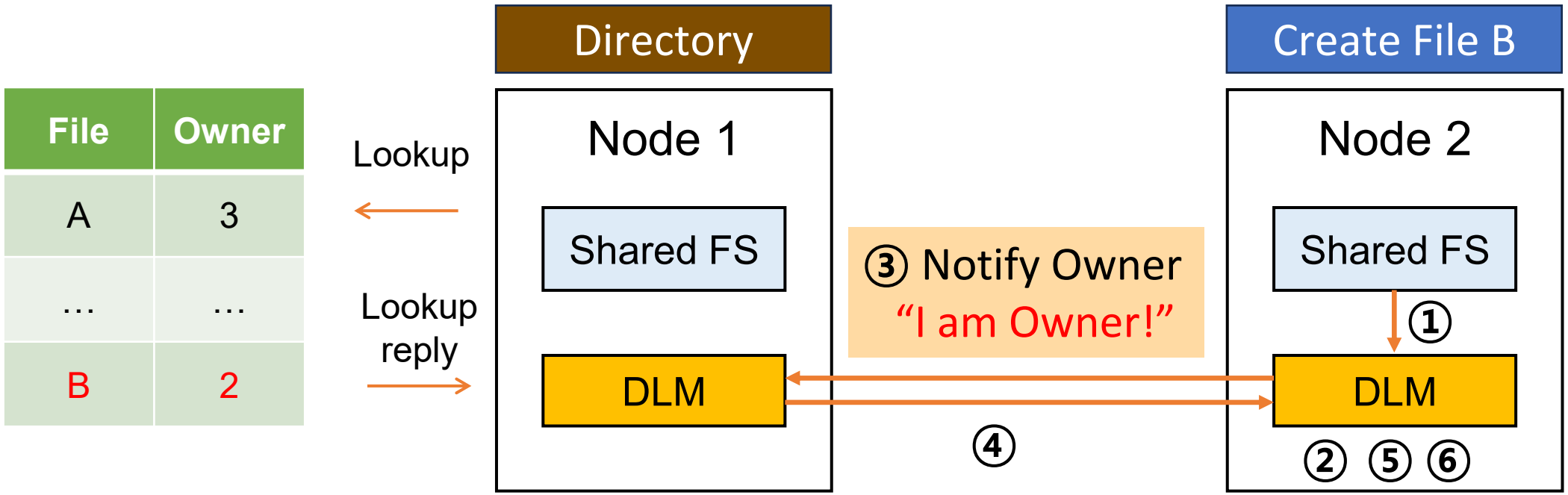
Lockify Design

1. Self-Owner Notification



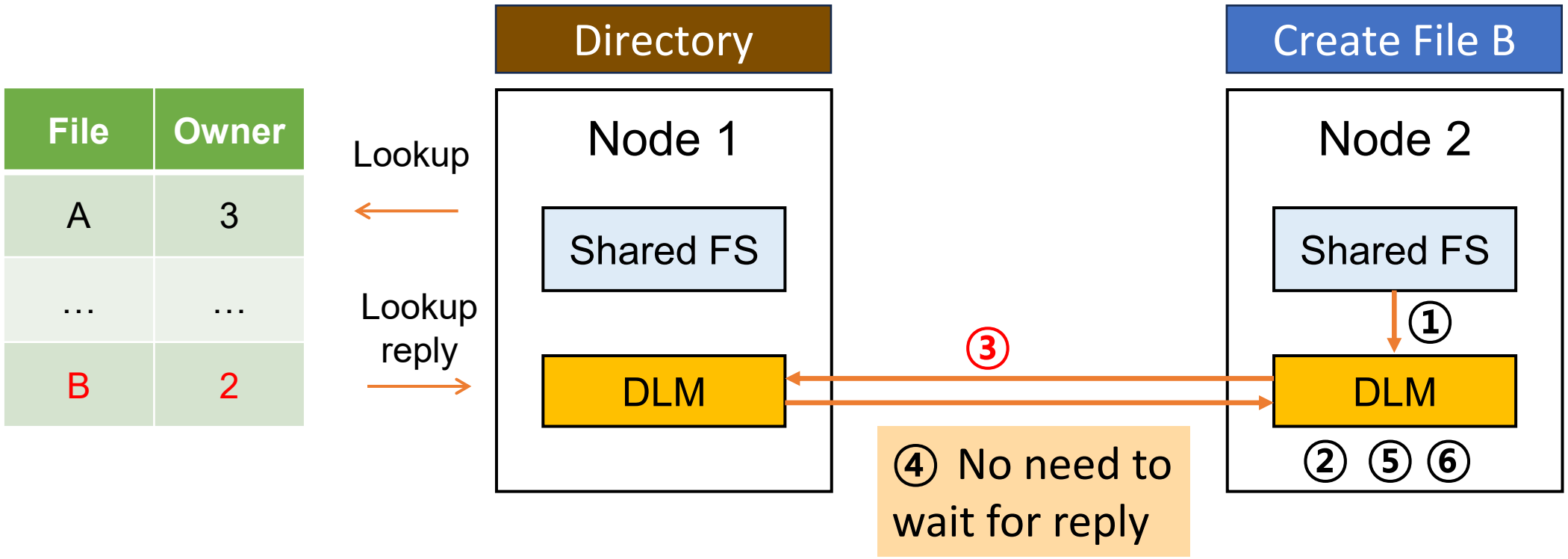
Lockify Design

1. Self-Owner Notification



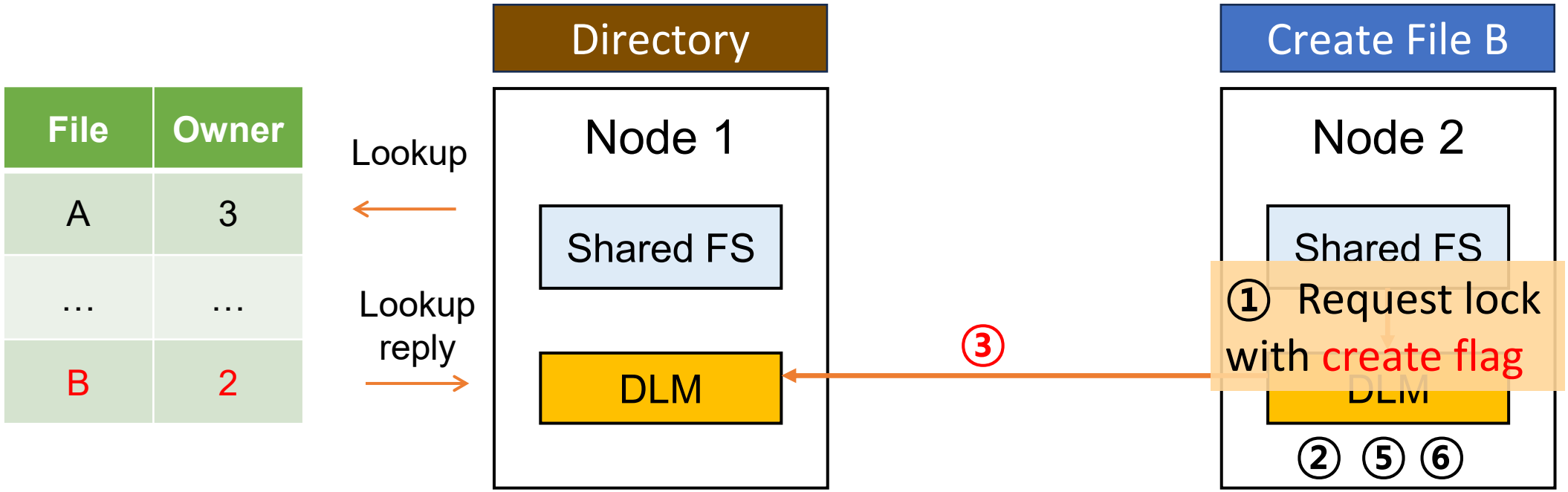
Lockify Design

1. Self-Owner Notification



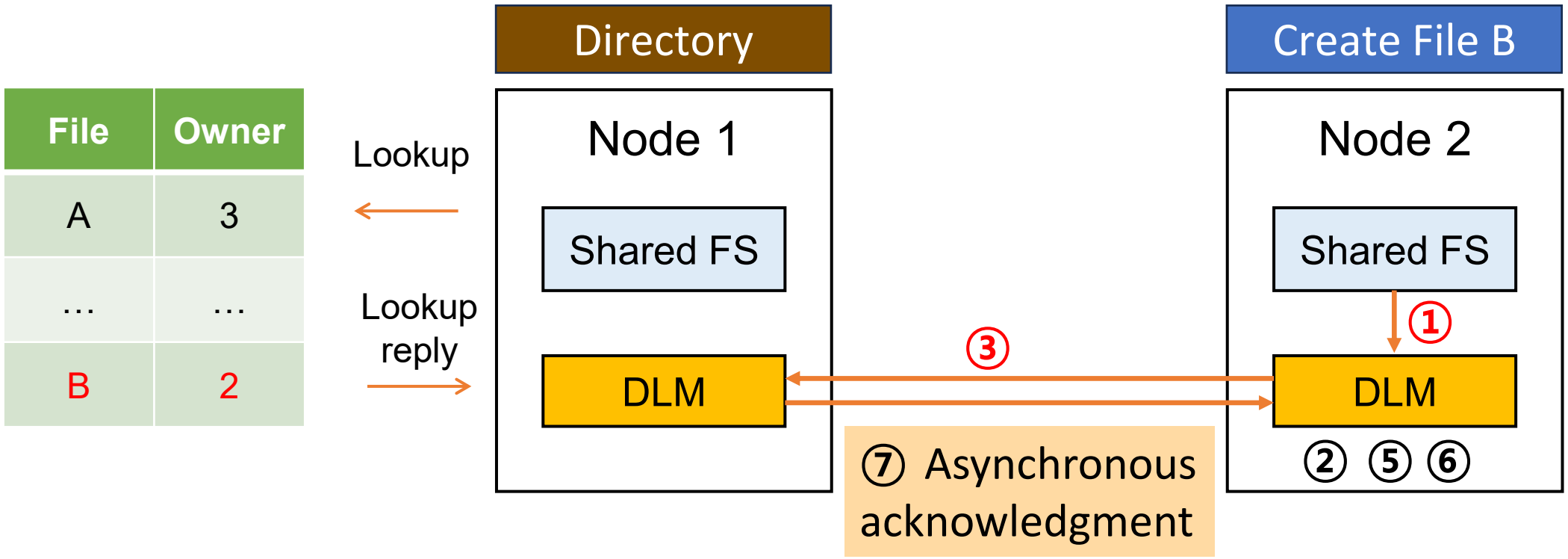
Lockify Design

2. Extended Lock Acquisition Interface



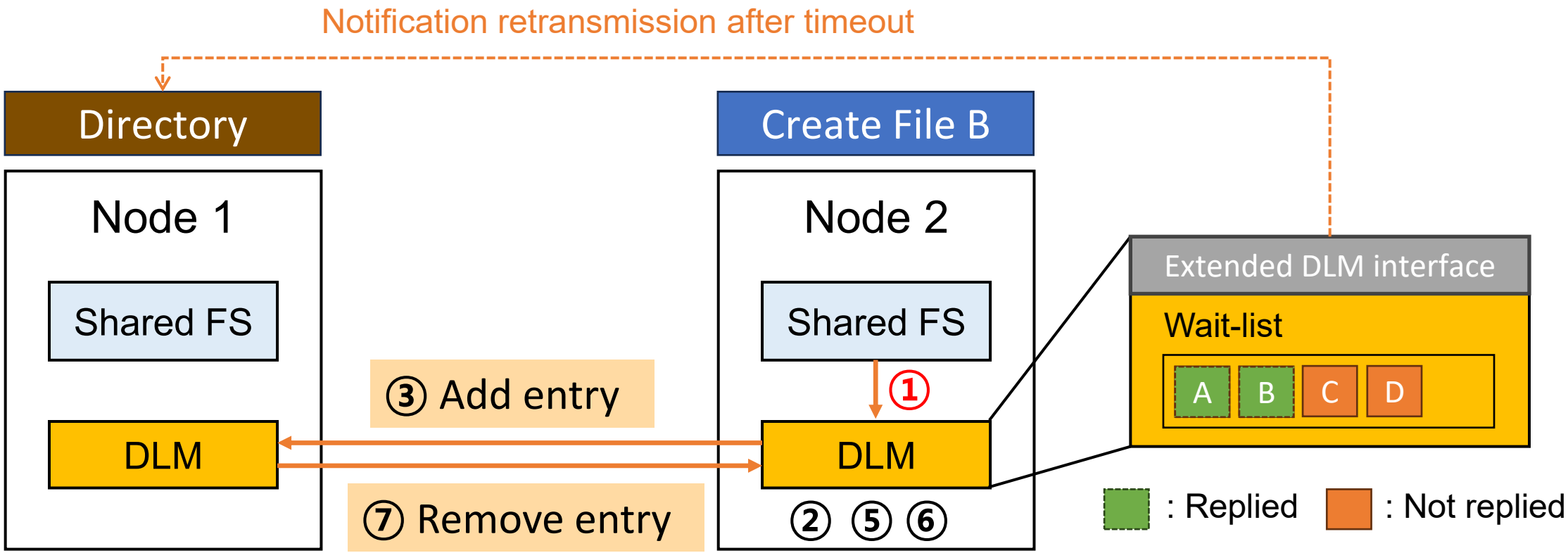
Lockify Design

3. Asynchronous Ownership Management



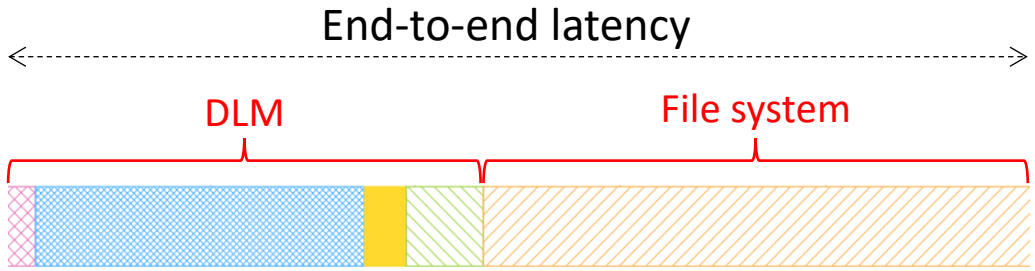
Lockify Design

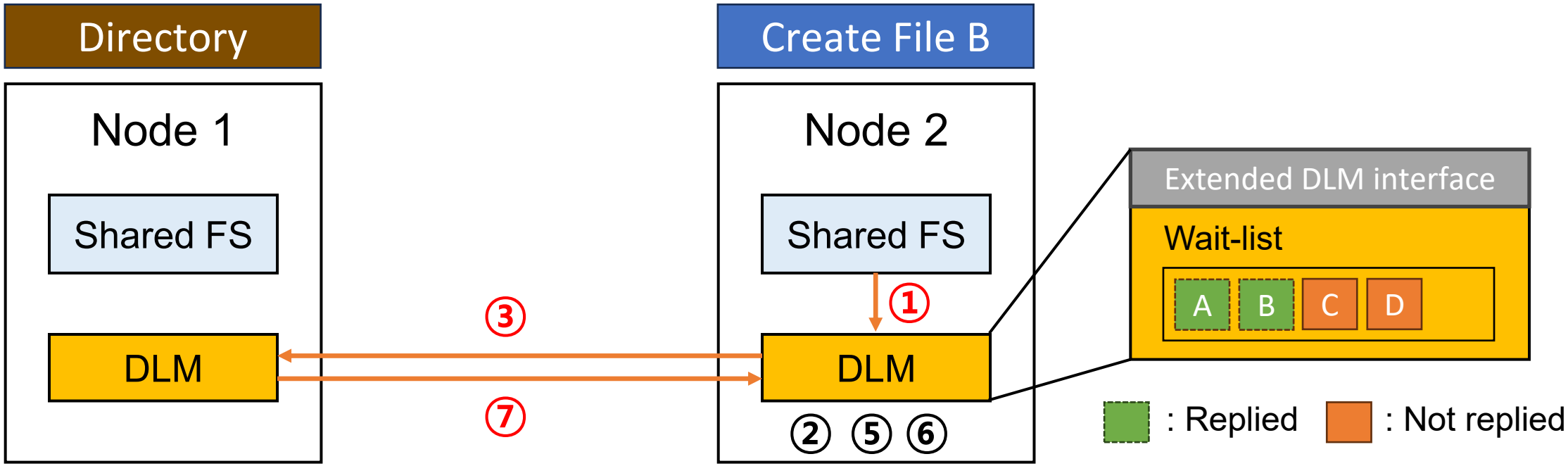
3. Asynchronous Ownership Management



Lockify Design

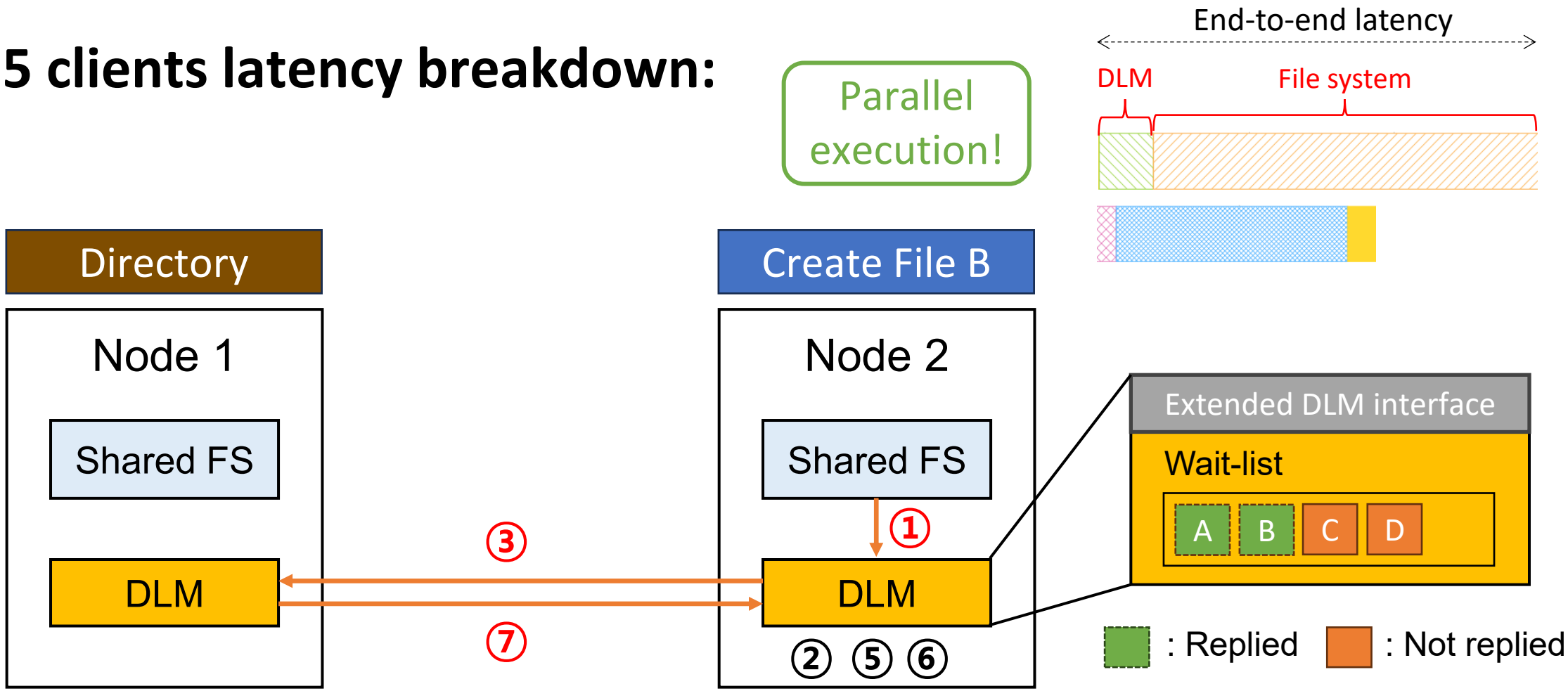
5 clients latency breakdown:





Lockify Design

5 clients latency breakdown:



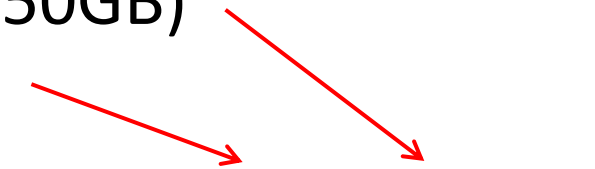
Evaluation

Implemented in the Linux kernel 6.6.23

- On top of the kernel DLM
- Also modified GFS2 and OCFS2 with the extended lock interface

Testbed Setup

- CPU: Intel Xeon Gold 5115 (2.40GHz, 20 cores)
- RAM: 64GB
- Storage: Samsung 970 EVO Plus NVMe SSD (250GB)
- Network: Mellanox ConnectX-4 NIC (56Gbps)



Evaluation

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

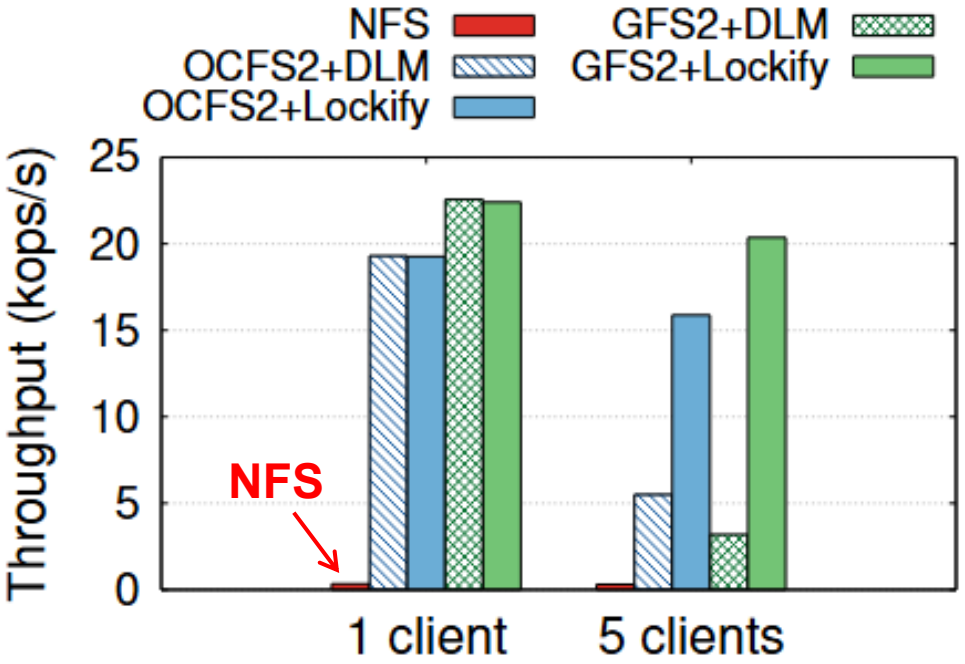
Target Systems

- GFS2 + DLM, GFS2 + Lockify
- OCFS2 + DLM, OCFS2 + Lockify
- NFS

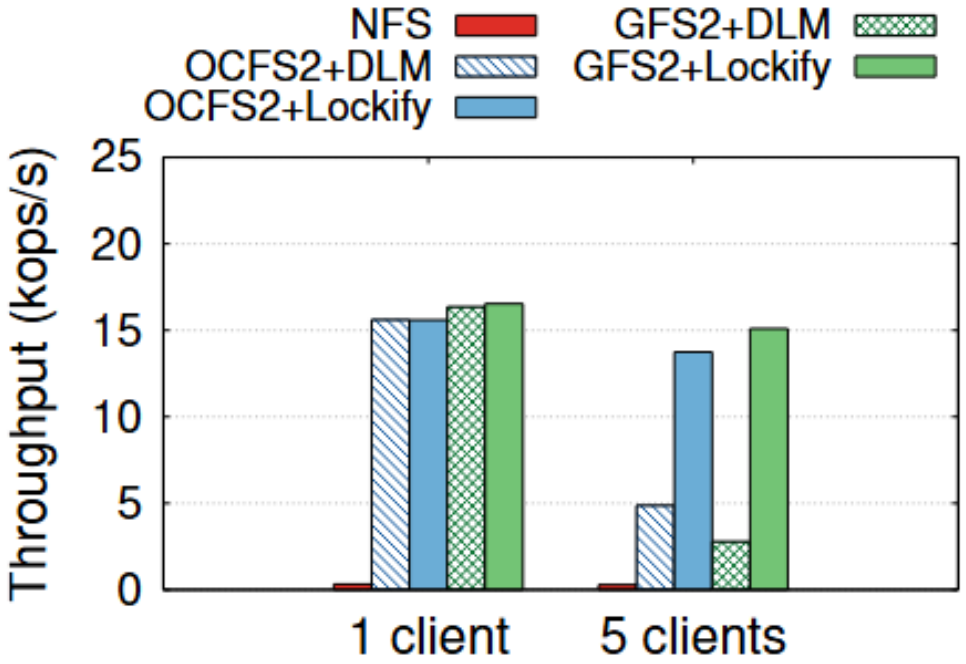
Microbenchmark (no contention)

mdtest

– Single client creates files and directories



(a) Directory creation.

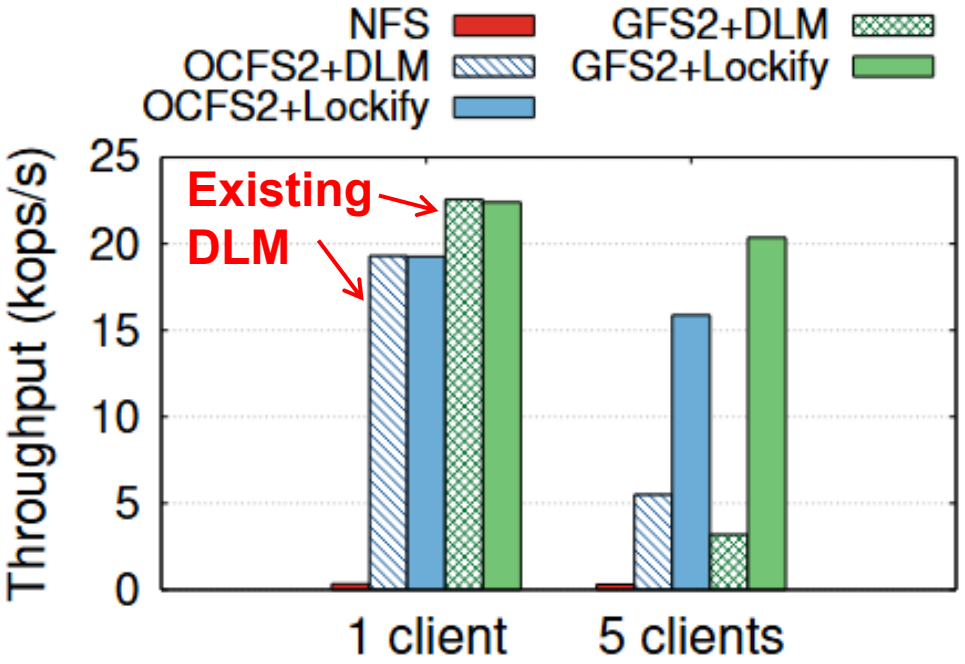


(b) File creation.

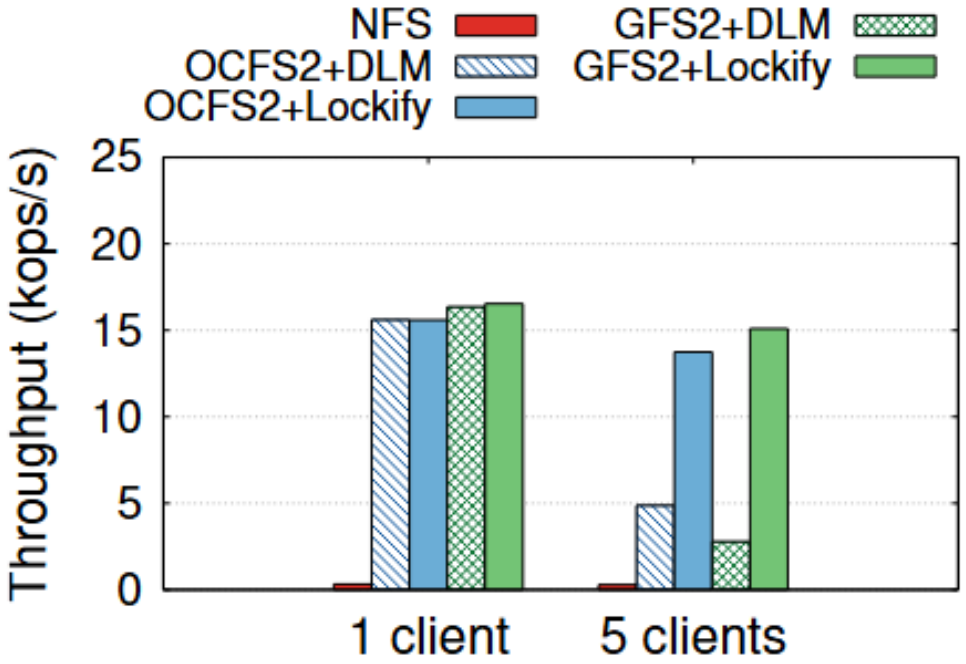
Microbenchmark (no contention)

mdtest

– Single client creates files and directories



(a) Directory creation.

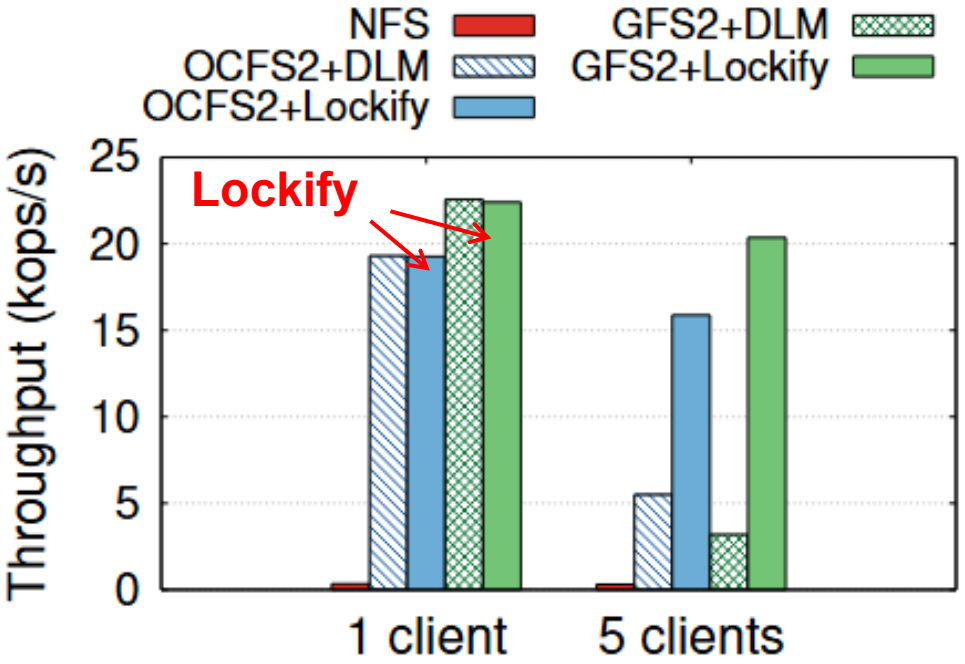


(b) File creation.

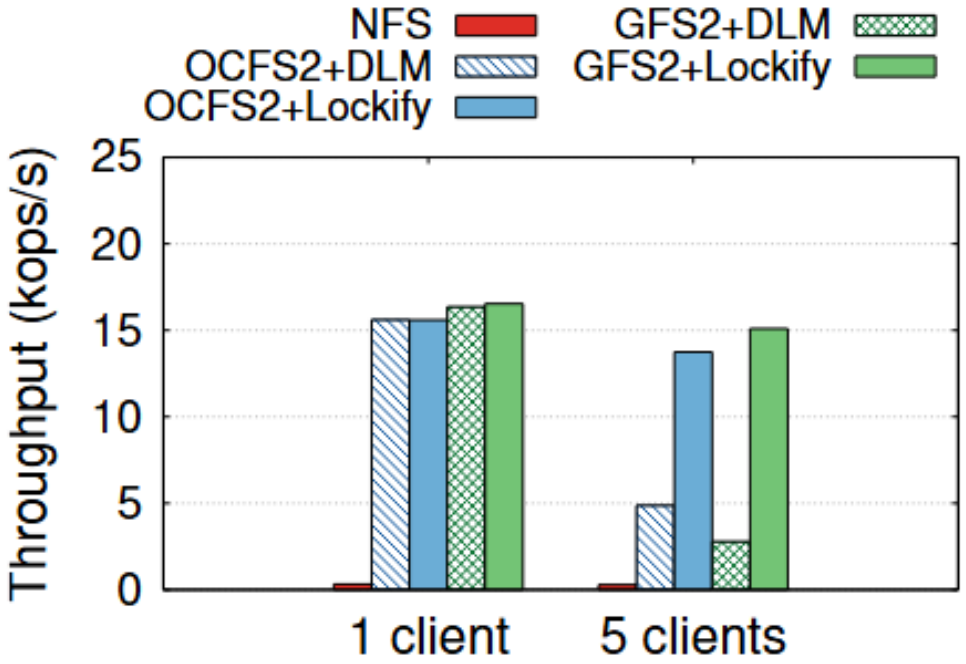
Microbenchmark (no contention)

mdtest

– Single client creates files and directories



(a) Directory creation.

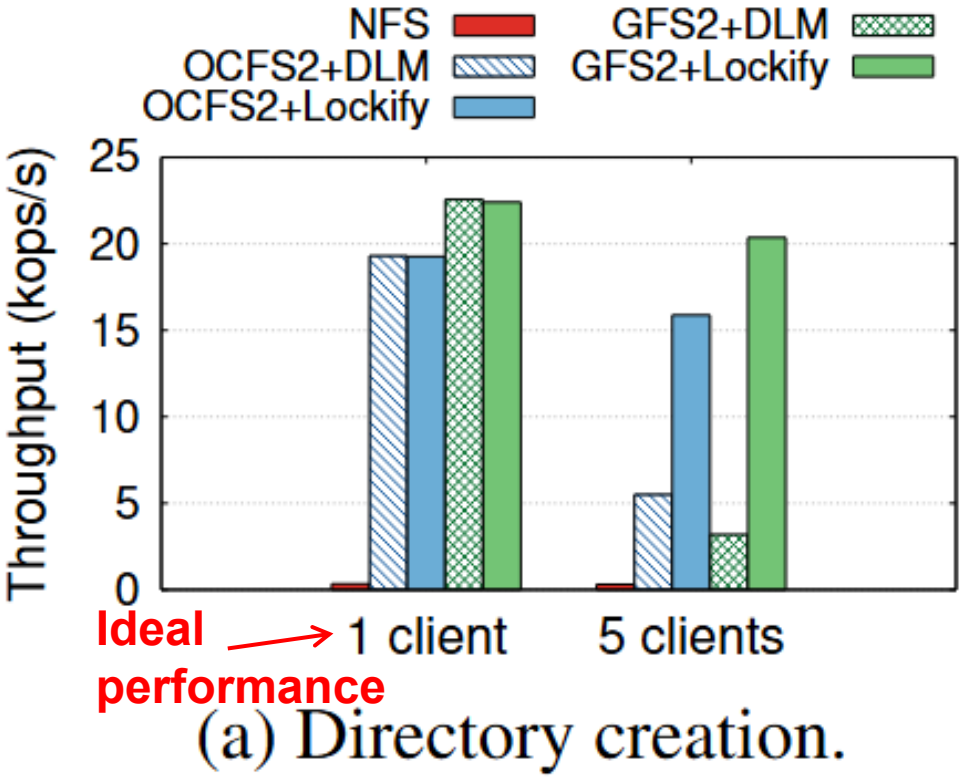


(b) File creation.

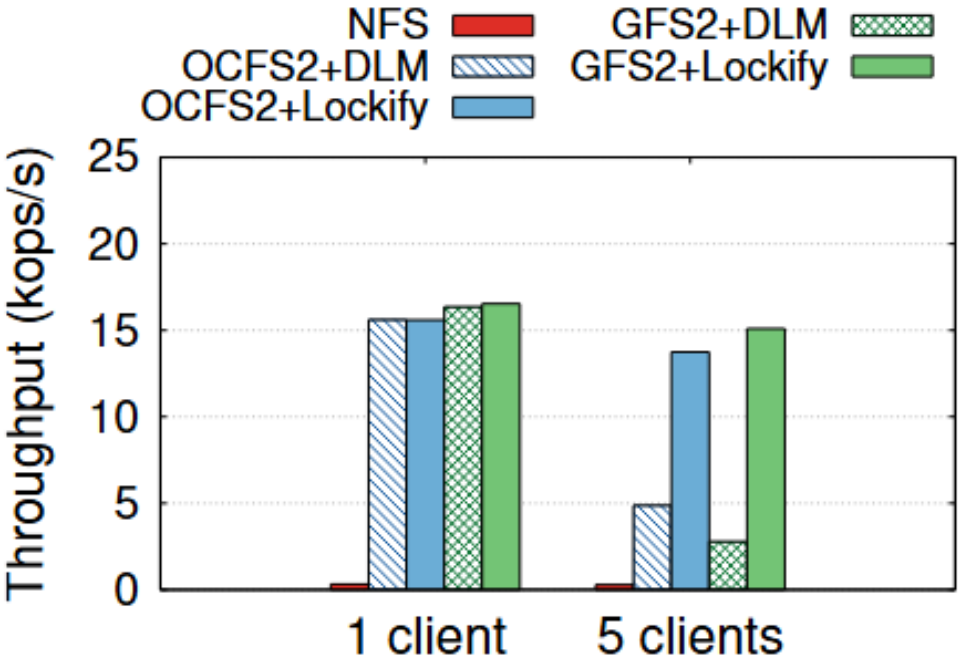
Microbenchmark (no contention)

mdtest

– Single client creates files and directories



(a) Directory creation.

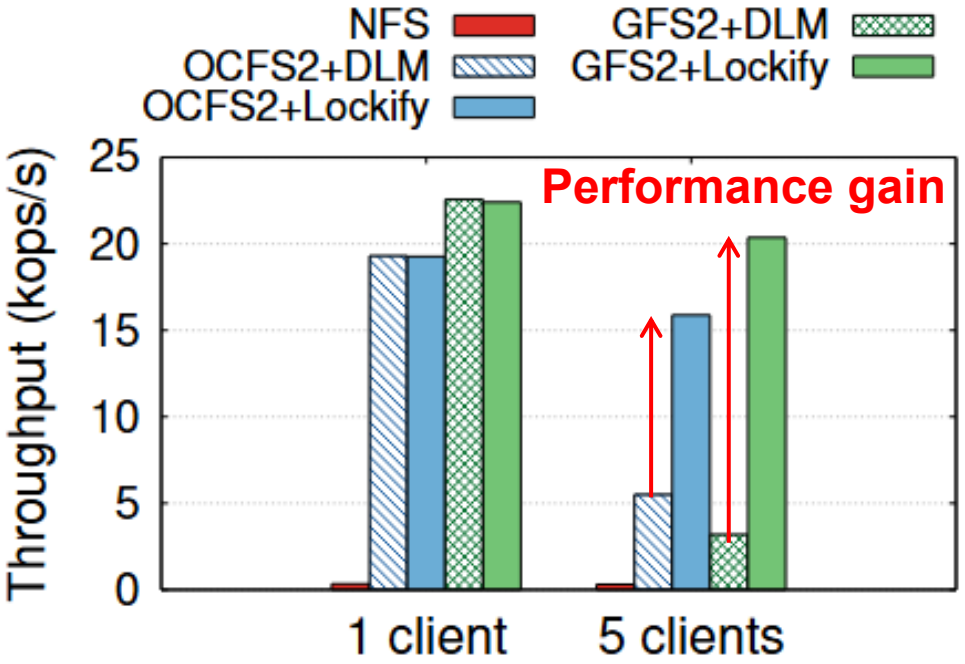


(b) File creation.

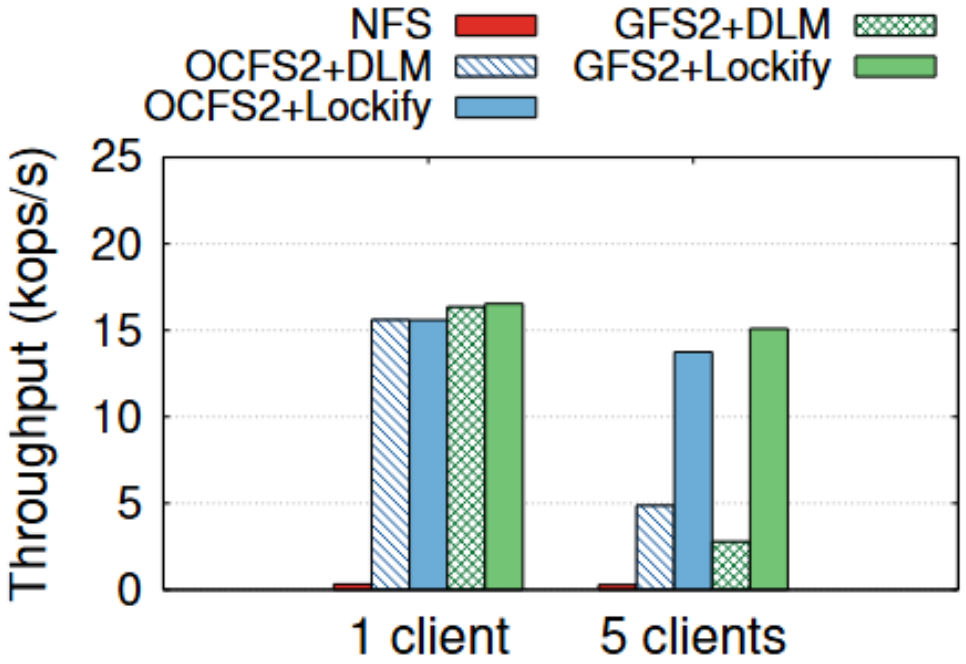
Microbenchmark (no contention)

mdtest

– Single client creates files and directories



(a) Directory creation.

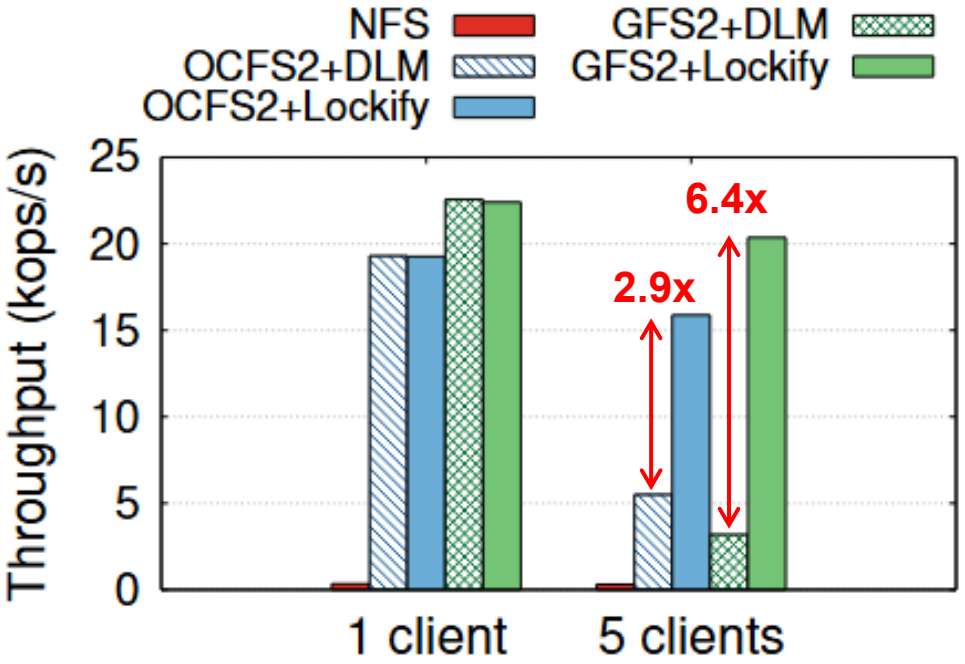


(b) File creation.

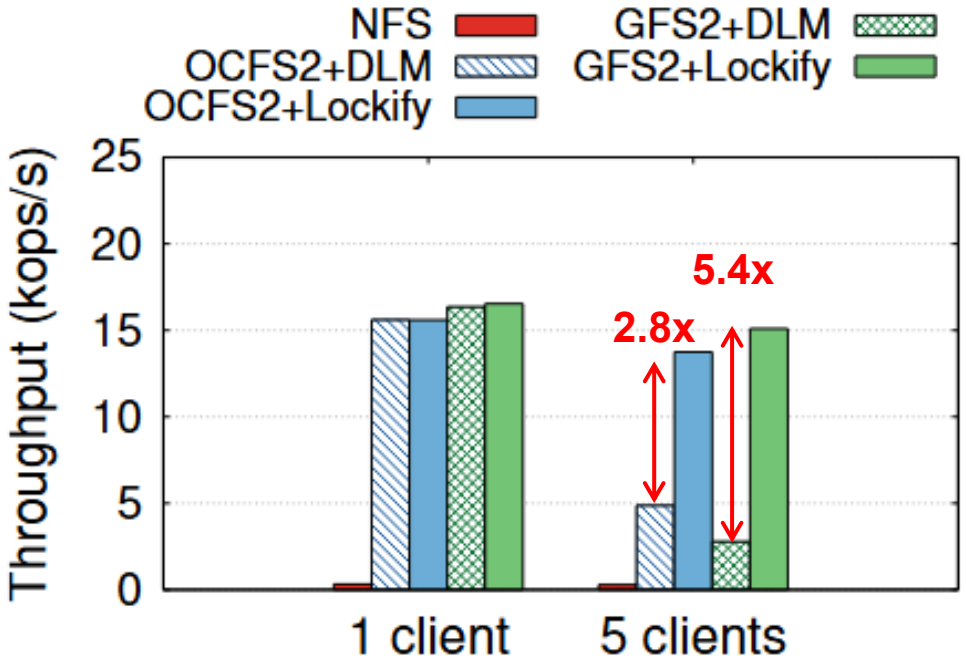
Microbenchmark (no contention)

mdtest

– Single client creates files and directories



(a) Directory creation.

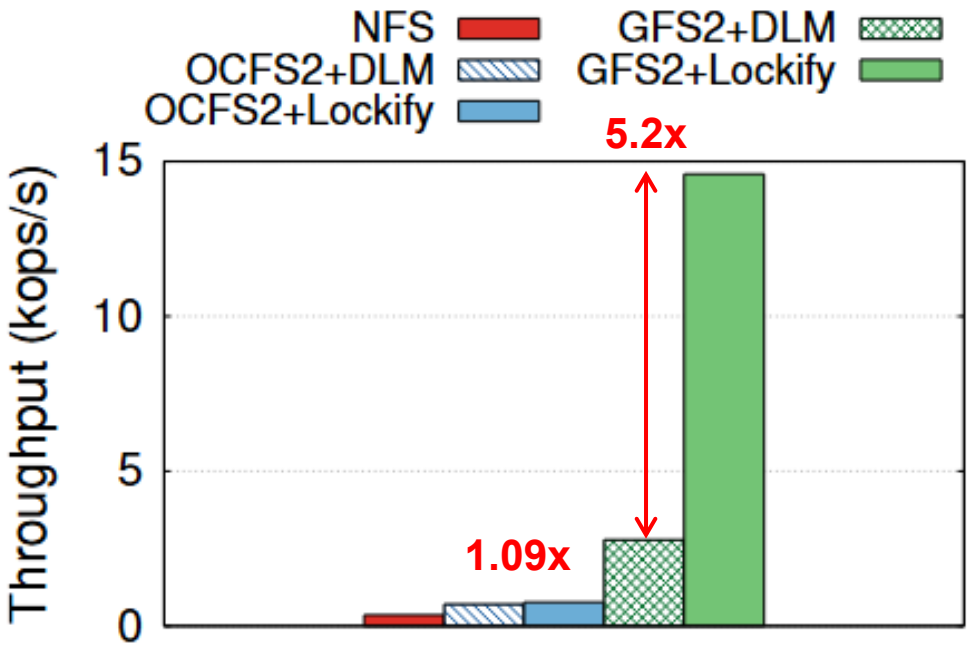


(b) File creation.

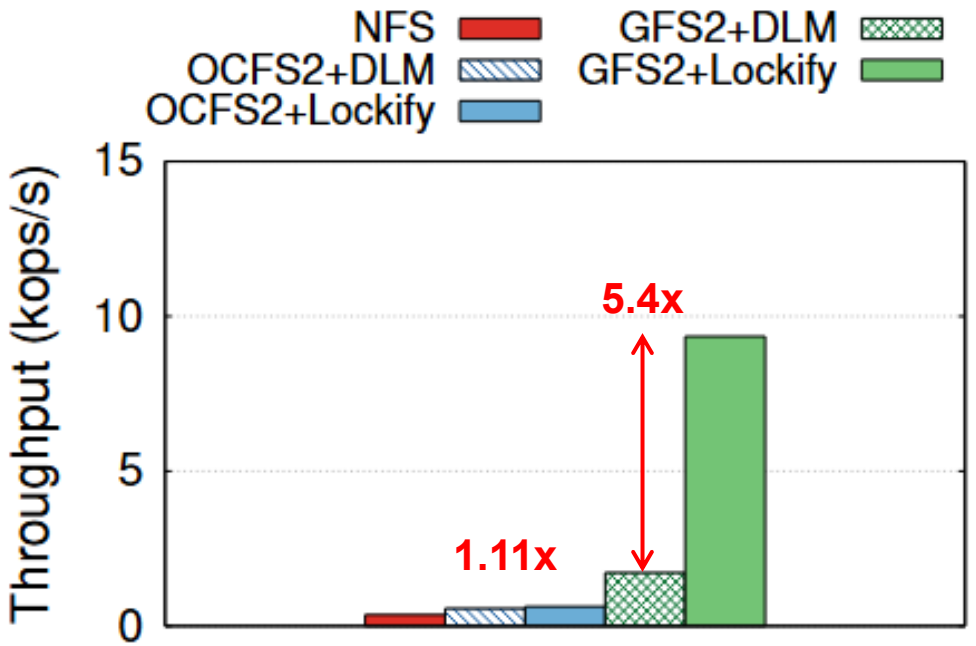
Microbenchmark (high contention)

mdtest

– 5 clients create files and directories simultaneously



(a) Directory creation.

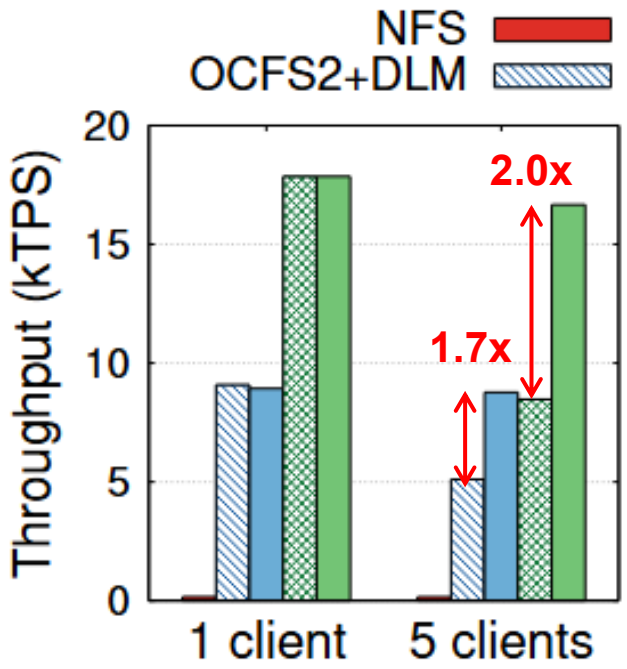


(b) File creation.

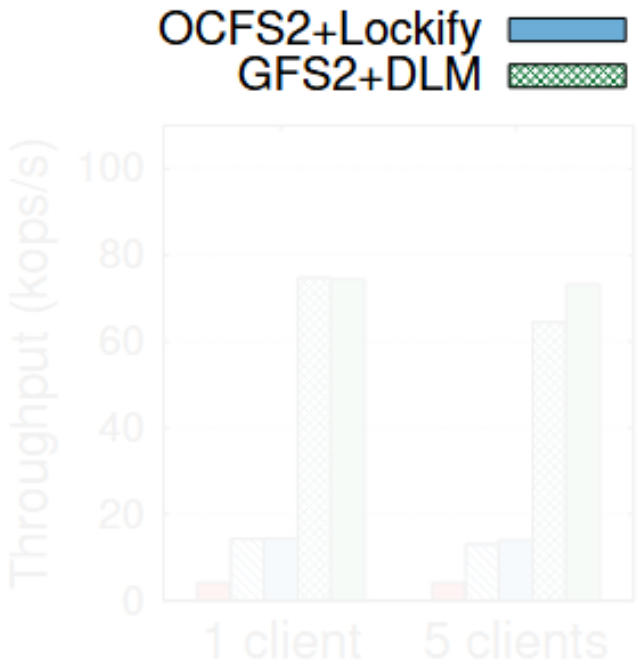
Real-World Benchmark

Postmark

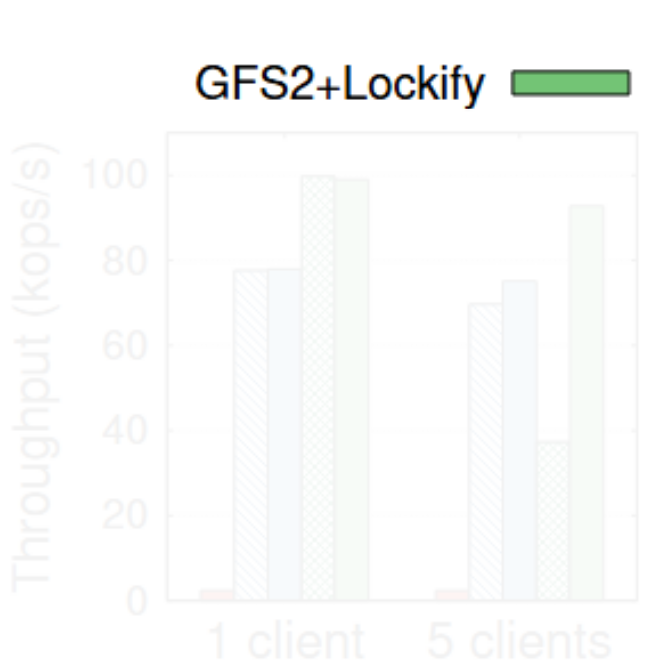
– Simulating real-world email and web service environments



(a) Postmark.



(b) Fileserver.

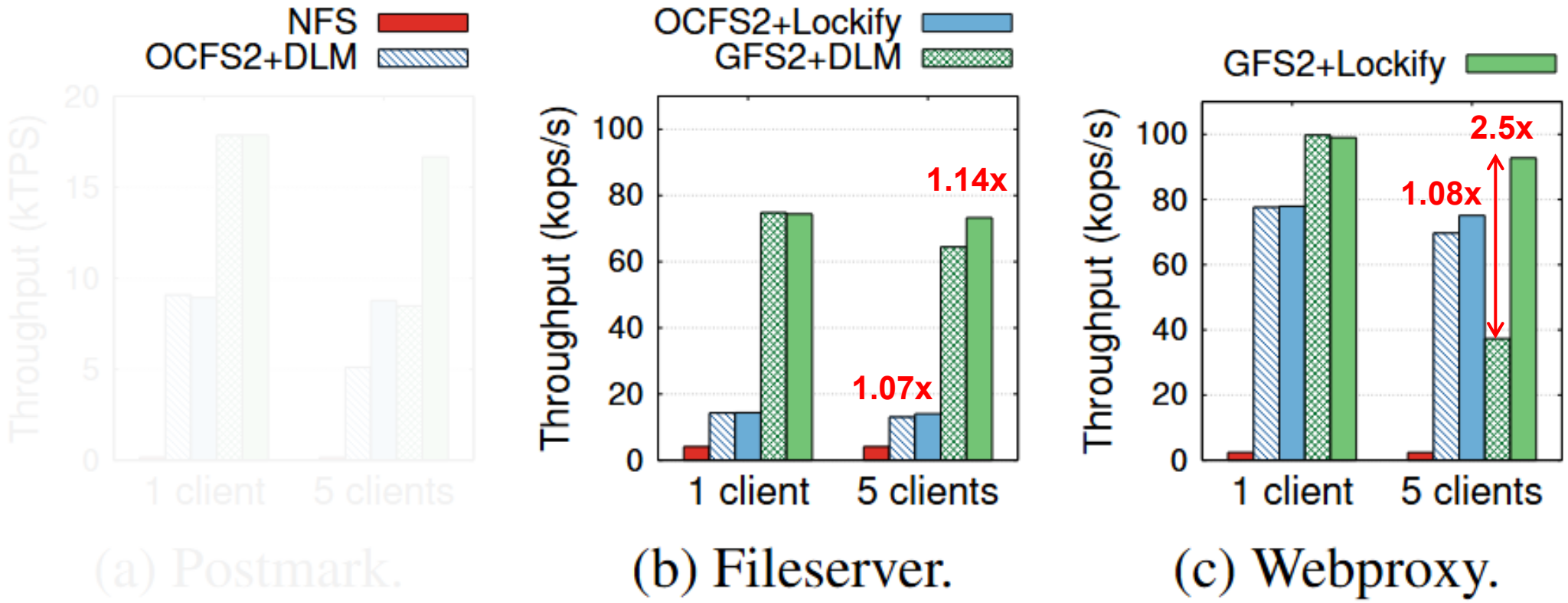


(c) Webproxy.

Real-World Benchmark

Filebench

– Workloads: Fileserver, Webproxy



Evaluation

Crash consistency test (xfstests)

– 75 generic tests for NFS

System		Score
GFS2	DLM	70
	Lockify	
OCFS2	DLM	67
	Lockify	

Evaluation

Crash consistency test (xfstests)

- 75 generic tests for NFS

System		Score
GFS2	DLM	70
	Lockify	70
OCFS2	DLM	67
	Lockify	67

No negative impact!

Conclusion

- Introduced novel Distributed Lock Manager
- Optimized create operations by hiding network latency
- Achieved up to 6.4x higher throughput compared to kernel DLM

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