
# **AsyncFS**: Metadata Updates Made Asynchronous for Distributed Filesystems with In-Network Coordination

Jingwei Xu, Mingkai Dong, Qiulin Tian, Ziyi Tian, Tong Xin, and Haibo Chen
IPADS, *Shanghai Jiao Tong University*

Presenter: Chongzhuo Yang, *USTC*

# 1.1. Large-Scale Distributed File System



File Management in Datacenters

1. **Numerous Files**:
   - Modern datacenters store numerous files (e.g., hundreds of billions).
2. **Frequent Access**:
   - Metadata constitute the majority of DFS operations (e.g., 67%-96% in Baidu).
3. Skew access pattern

*The **metadata performance** limits the scalability of distributed file systems.*

# 1.2. Metadata Review



## Tree Layout

(dir_id, name) → id

## File/Dir Attributes

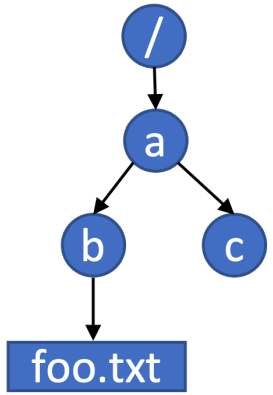
id → file attribute

id → directory attributes

E.g., the **`create`** under one directory will

1. insert one file attribute,
2. update the directory attribute.

| Key | | Value | |
|---|---|---|---|
| dir_id | name | id | type |
| 0 | 'a' | 1 | DIR |
| 1 | 'b' | 2 | DIR |
| 1 | 'c' | 3 | DIR |
| 2 | 'foo.txt' | 4 | FILE |

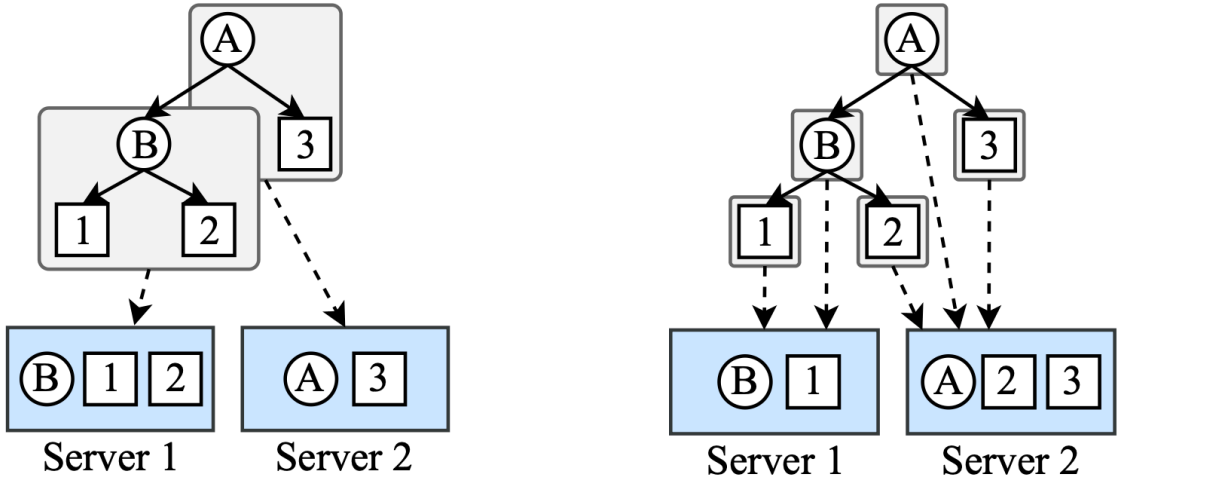



| Key | Value | | | | |
|---|---|---|---|---|---|
| id | type | name | children | length | ... |
| 0 | DIR | '/' | 1 | - | |
| 1 | DIR | 'a' | 2 | - | |
| 2 | DIR | 'b' | 1 | - | |
| 3 | DIR | 'c' | 0 | - | |
| 4 | FILE | 'foo.txt' | - | 4096 | |

# 1.3. Scaling DFS Metadata Performance



- Partitioning the metadata tree across multiple servers.



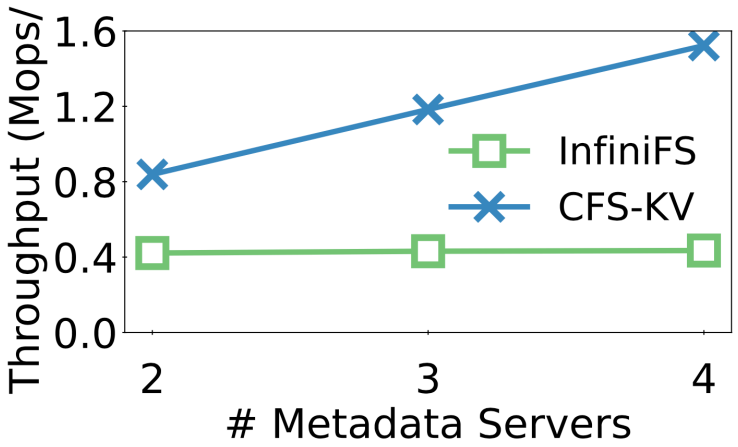

**(a)** Parent-children grouping **(b)** Parent-children separating

# 1.3. Scaling DFS Metadata Performance

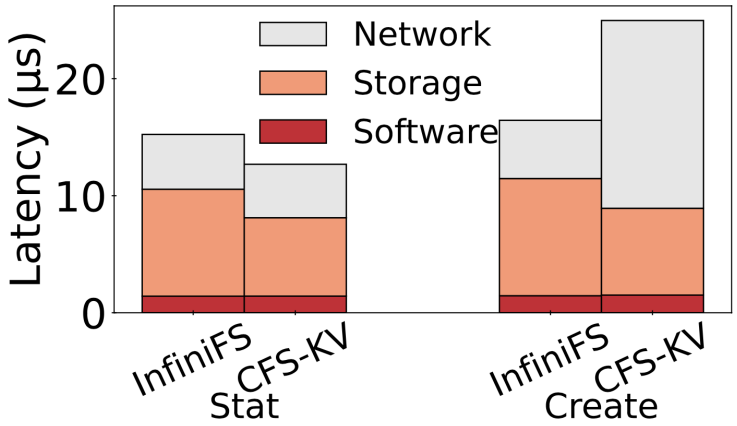


**Challenge 1**: The tradeoff between **load balance and locality** in system design.

| Strategy | Load Balance | Metadata Locality |
|---|---|---|
| P/C grouping (Ceph, InfiniFS) | ✗ | ✓ |
| P/C separating (CFS, 3FS) | ✓ | ✗ |




**(a)** Throughput of *stat*.



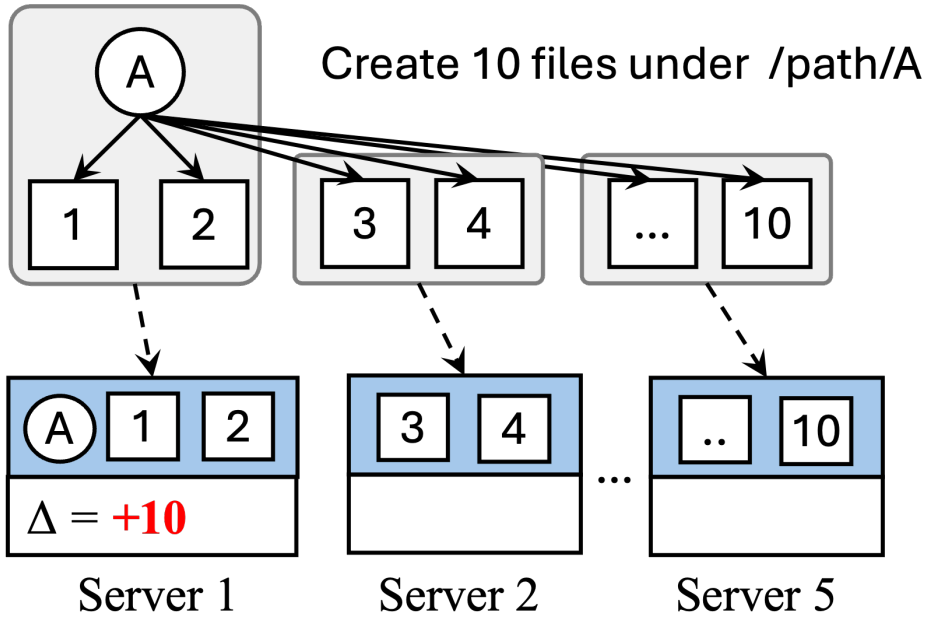

**(b)** Latency breakdown.

# 1.3. Scaling DFS Metadata Performance



**Challenge 2**: Parent-based operations will **incur contention** for a large directory.

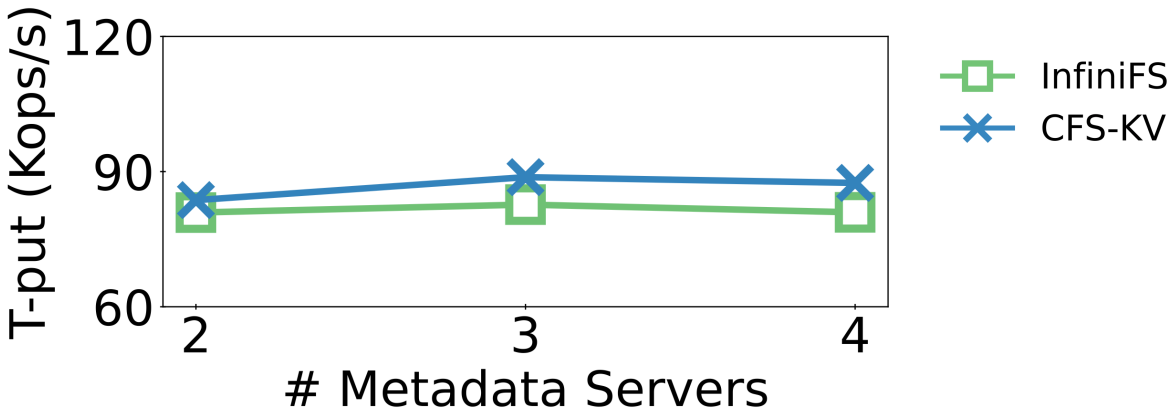
E.g., the directory attributes (children/links +1) of A will be updated +10 times.



# 1.3. Scaling DFS Metadata Performance



**Challenge 2**: Parent-based operations will **incur contention** for a large directory.



# 1.4. Workloads in Datacenters



**Insight 1**: Datacenter workloads are skewed

- Directory Size
- Directory Hotness
- Burst Updates

> Load balance is essential for DFS.

**Insight 2**: For directory attribute: #updates / #reads = 7

- Pigeonhole Principle

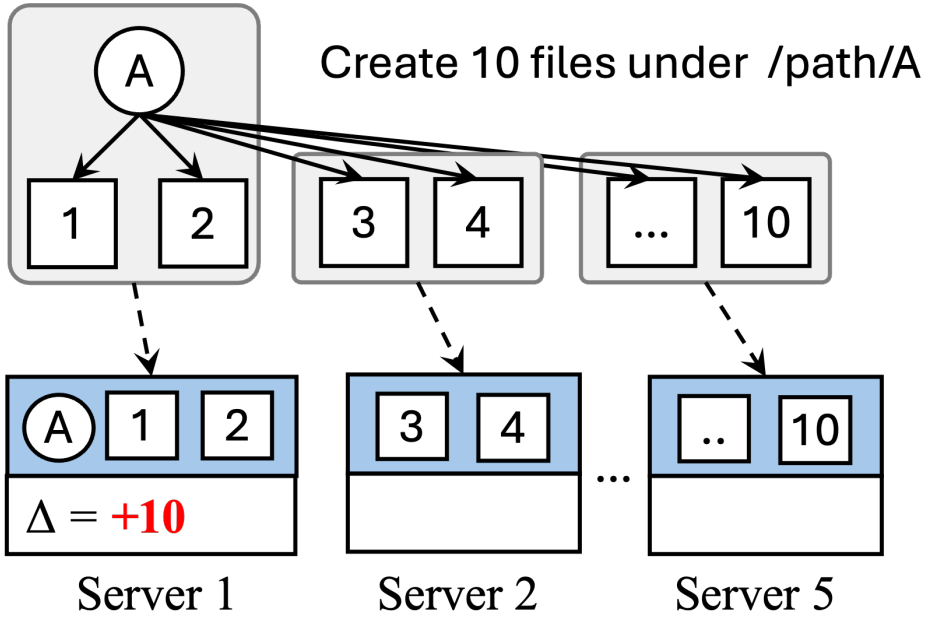
> The directory update is not immediately followed by directory reads.

# 2.1. Overview



**Main idea**: update directory attributes ***asynchronously***.

- Create a large number of files in the directory

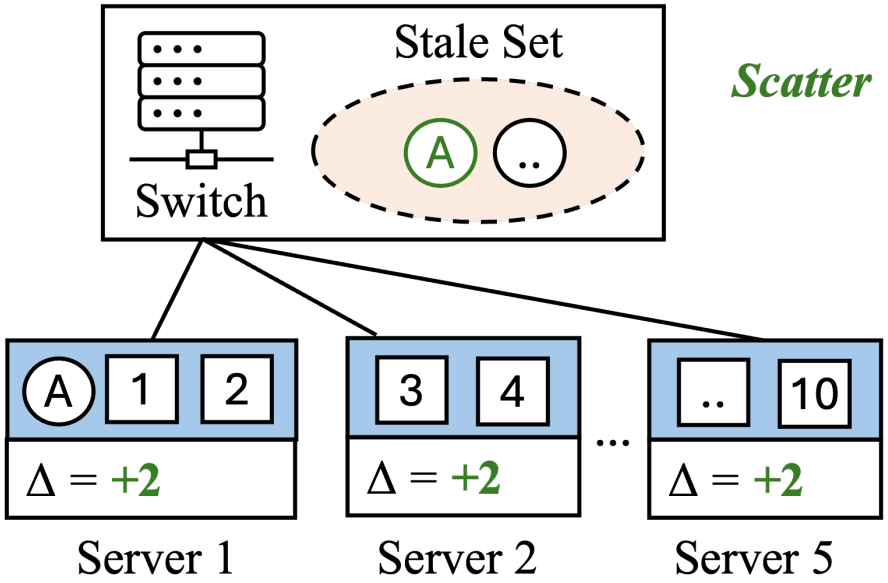


# 2.1. Overview



**Main idea**: update directory attributes ***asynchronously***.

- Scatter updates across multiple servers (no cross-server transactions).

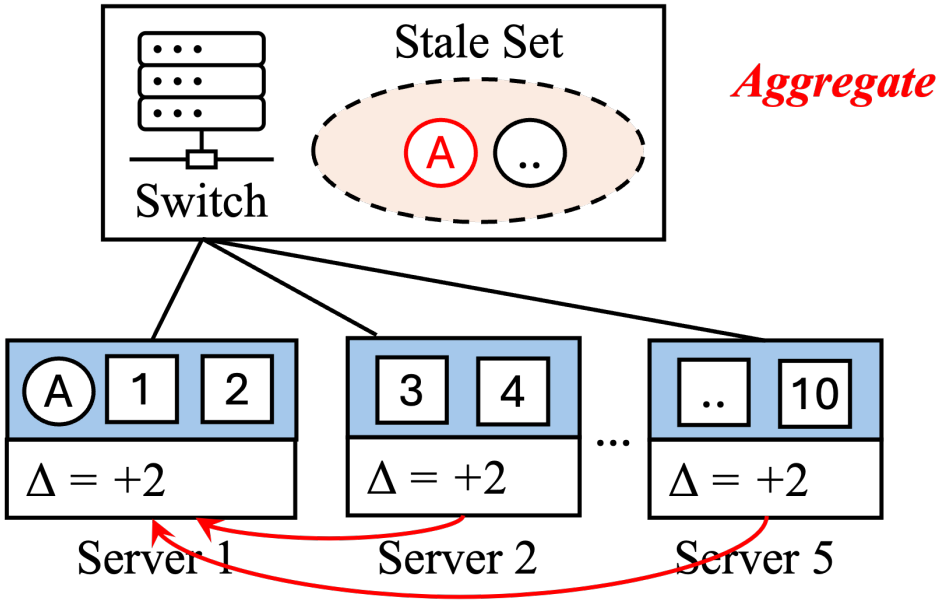


# 2.1. Overview



**Main idea**: update directory attributes ***asynchronously***.

- Aggregate updates from other servers when directory reads occur.

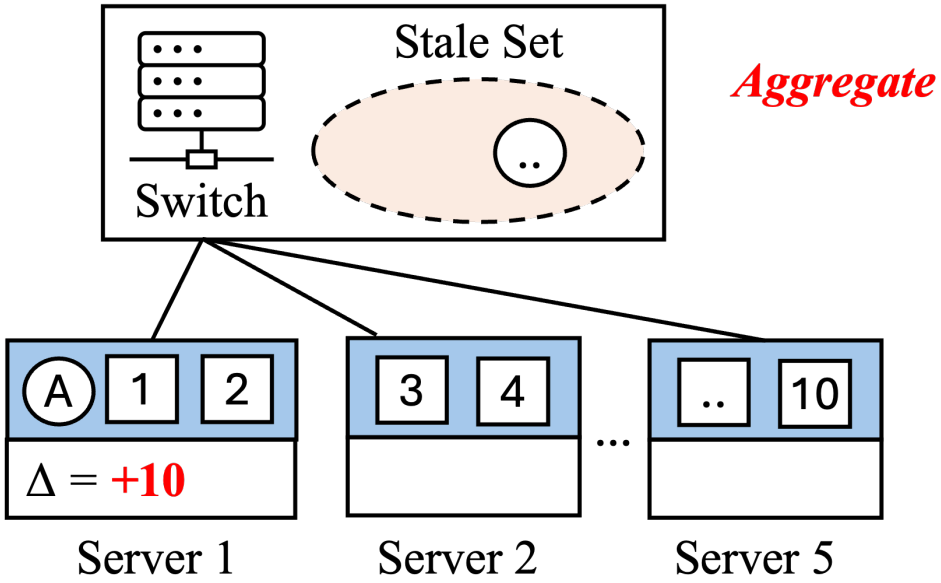


# 2.1. Overview



**Main idea**: update directory attributes ***asynchronously***.

- Aggregate updates from other servers when directory reads occur.



# 2.1. Overview



**Insights**:

1. *Load balance is essential for DFS.*
2. *The directory update is not immediately followed by directory reads.*

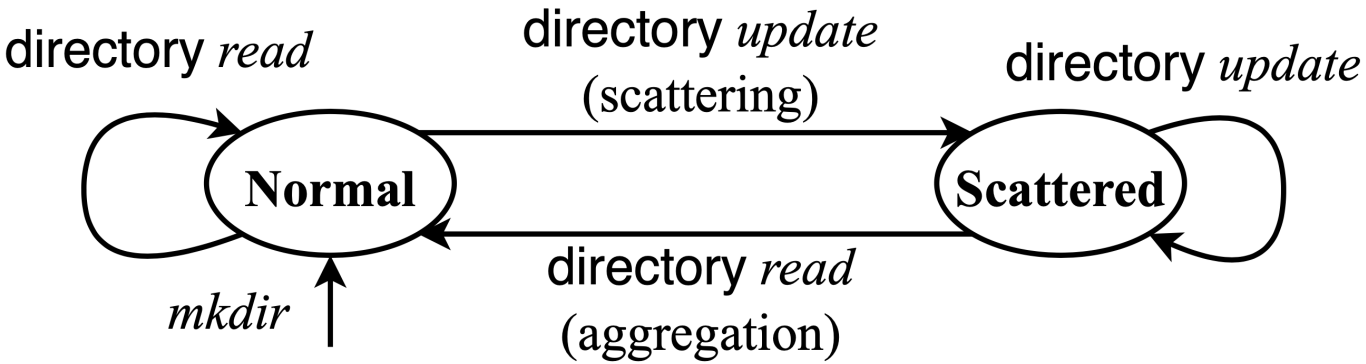
| **Design Goal** | **Key Design** |
|---|---|
| Load Balance | Parent/Children Separating |
| Low Overhead | Asynchronous Metadata Operations |
| Avoid Contention | Change-Log Recast and Applying |

# 2.2. Asynchronous Metadata Operations



## 2.2.1. System State

- Normal
- Scattered



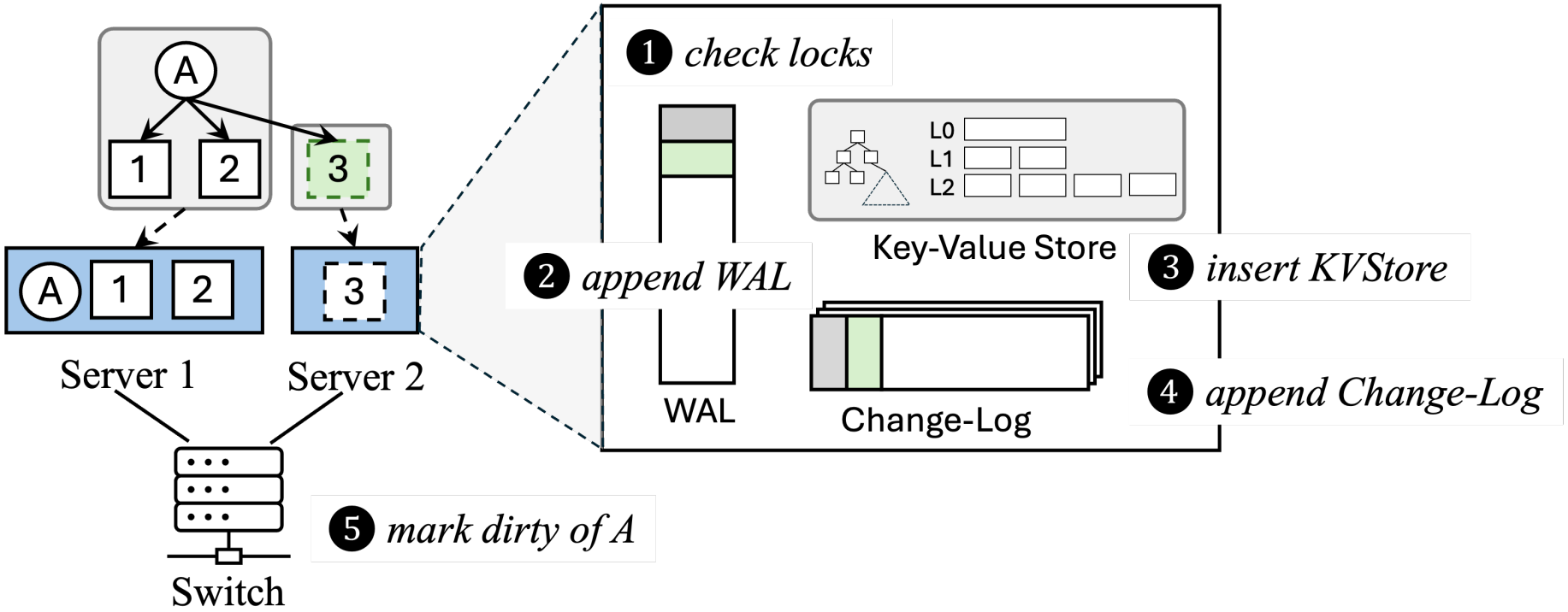

* *Transition granularity is the fingerprint group.*

# 2.2. Asynchronous Metadata Operations



## 2.2.2. Workflow of mkdir/create/delete

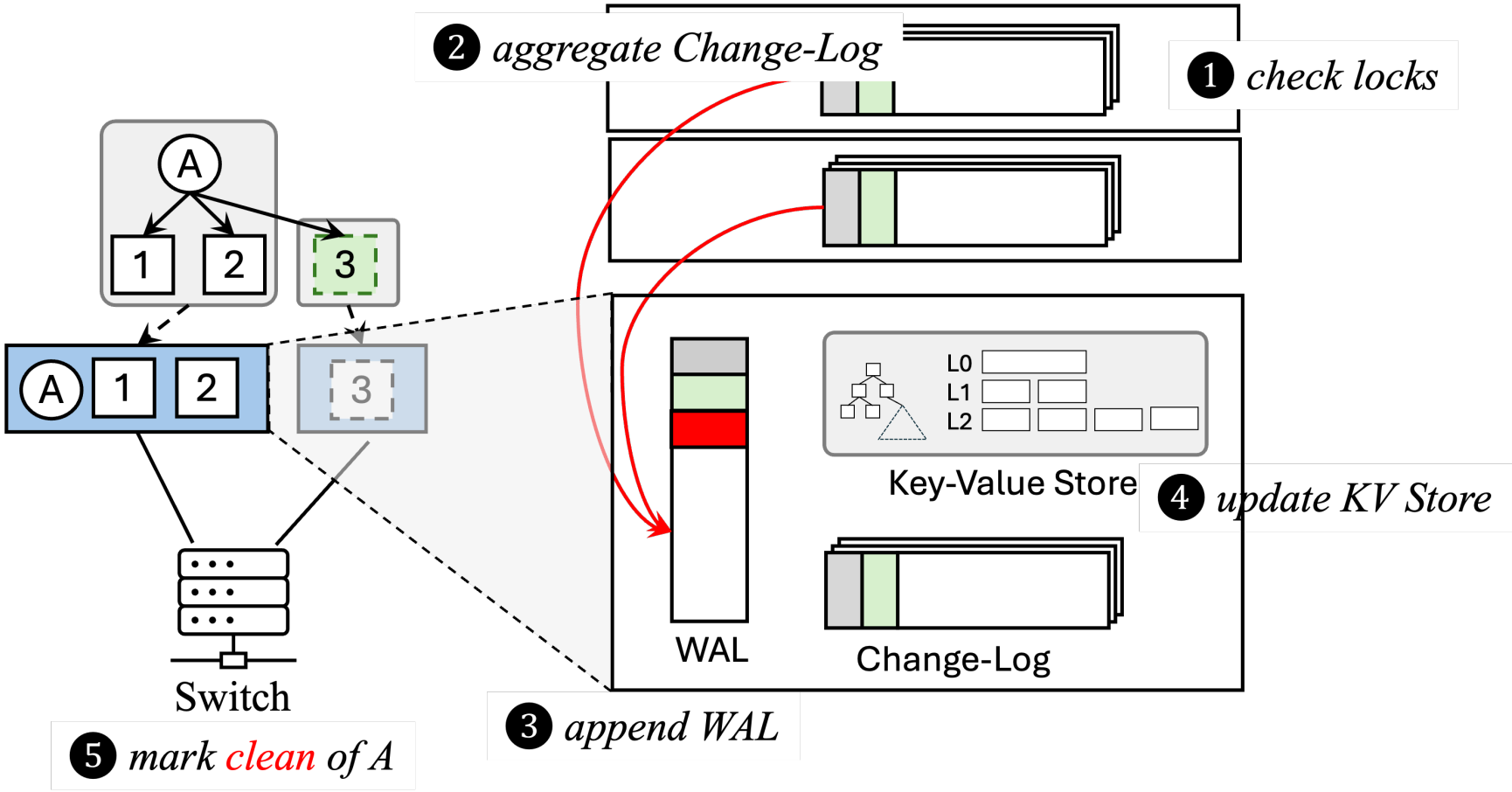
- One operation only updates key-value store in one server.



# 2.2. Asynchronous Metadata Operations

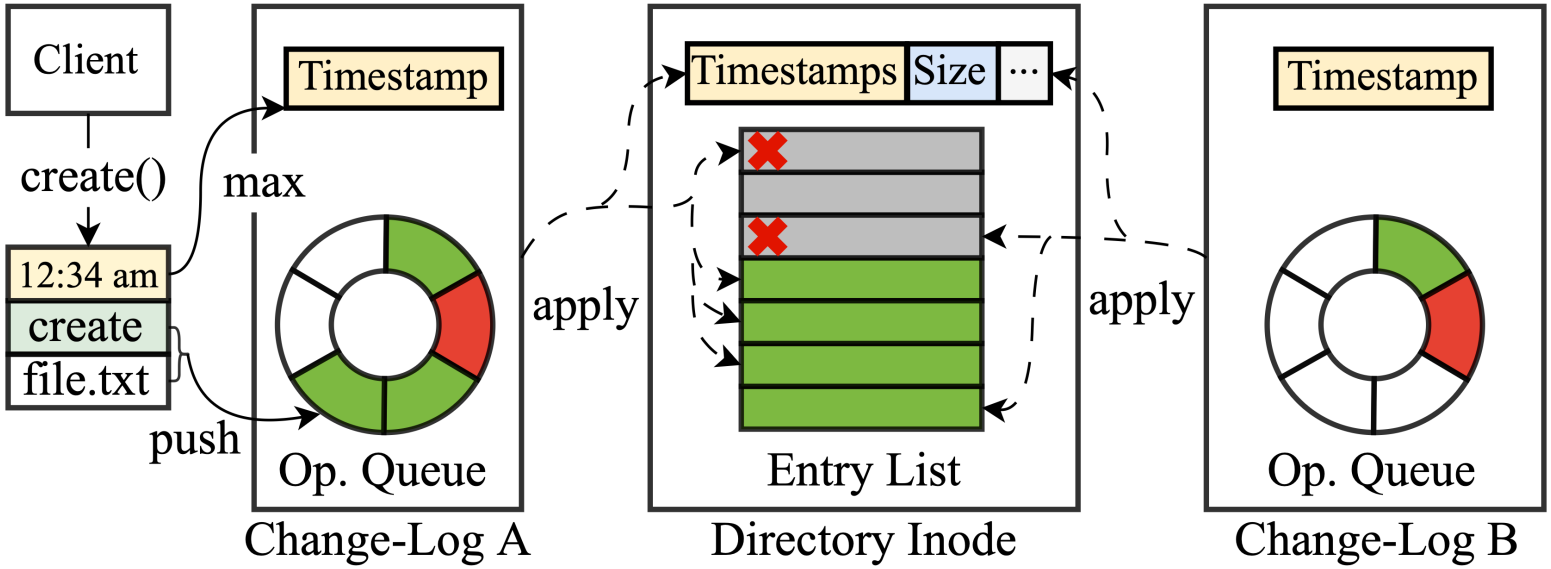


## 2.2.3. Workflow of statdir and readdir



# 2.3. Change-Log Recast and Applying






- Modified Time
- Entry List
- Directory Size

Apply change-log when the Op. Queue is full or timeout.

# 2.4. Crash Recovery



- How to recover when async is not applied?
- How to recover when switch is down?

**Server failure**: *recover by WAL*

Rebuild change-logs and key-value stores.

**Switch failure**: *recover by aggregations*

Transfer to normal state by aggregating all change-logs.

# 3.1. Experiment Configuration



**Table 4:** Hardware configuration of clusters.

| | | |
|---|---|---|
| Server Cluster | CPU | 2 × Intel Xeon Gold 5317 3.00GHz, 12 cores |
| | Memory | 16 × DDR4 2933MHz 16GB |
| | Storage | Intel Optane Persistent Memory |
| | Network | 2 × ConnectX-5 Single-Port 100GbE |
| Client Cluster | CPU | 2 × Intel Xeon E5-2650 v4 2.20GHz, 12 cores |
| | Memory | 16 × DDR4 2933MHz 16GB |
| | Network | 2 × ConnectX-4 Single-Port 100GbE |

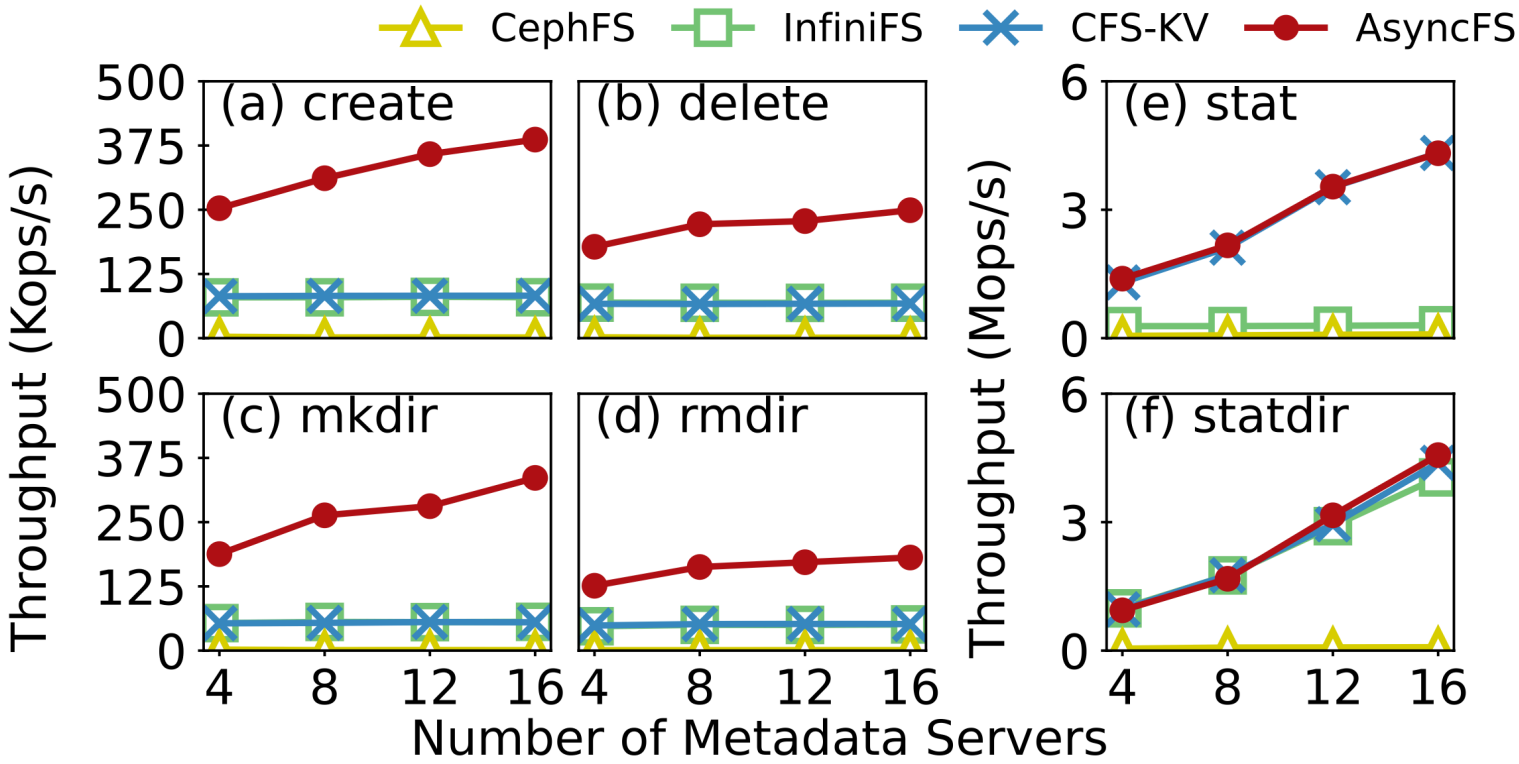
## Baselines

1. CephFS
2. InfiniFS (P/C grouping)
3. CFS-KV (P/C separating, based on RocksDB instead of TafDB)

# 3.2. Overall Performance



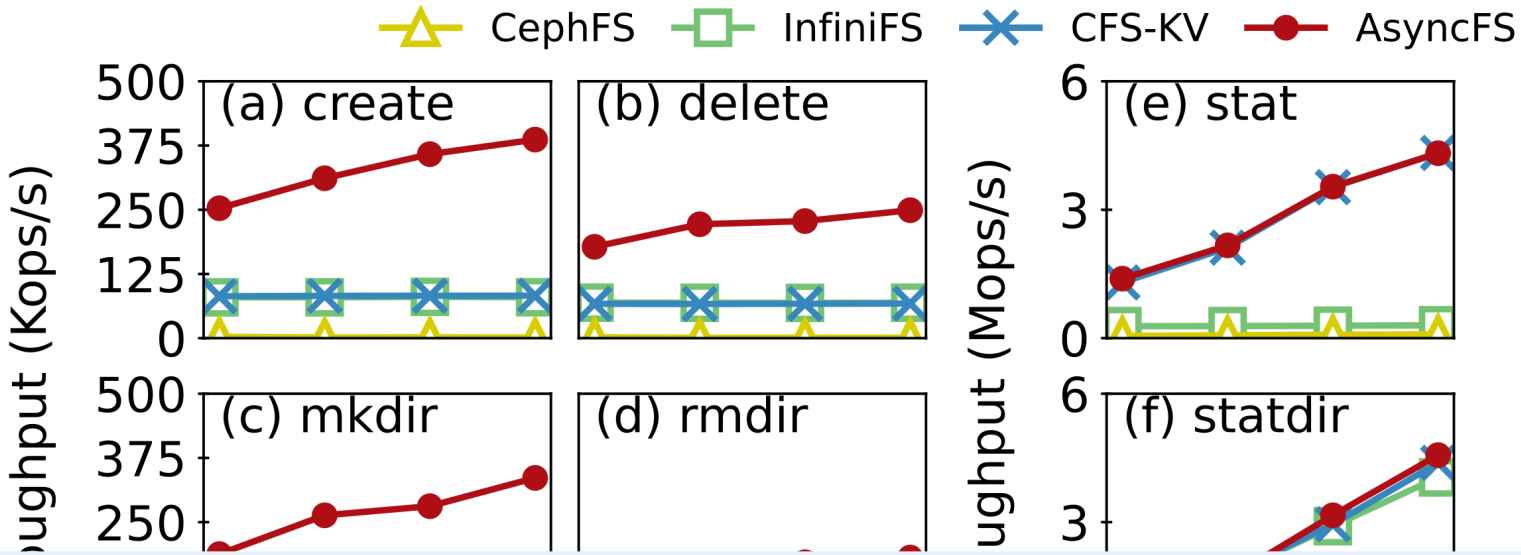
- [**Load Balance**] Operations under a large directory (10 million = 10,000,000).



# 3.2. Overall Performance



- [**Load Balance**] Operations under a large directory (10 million = 10,000,000).

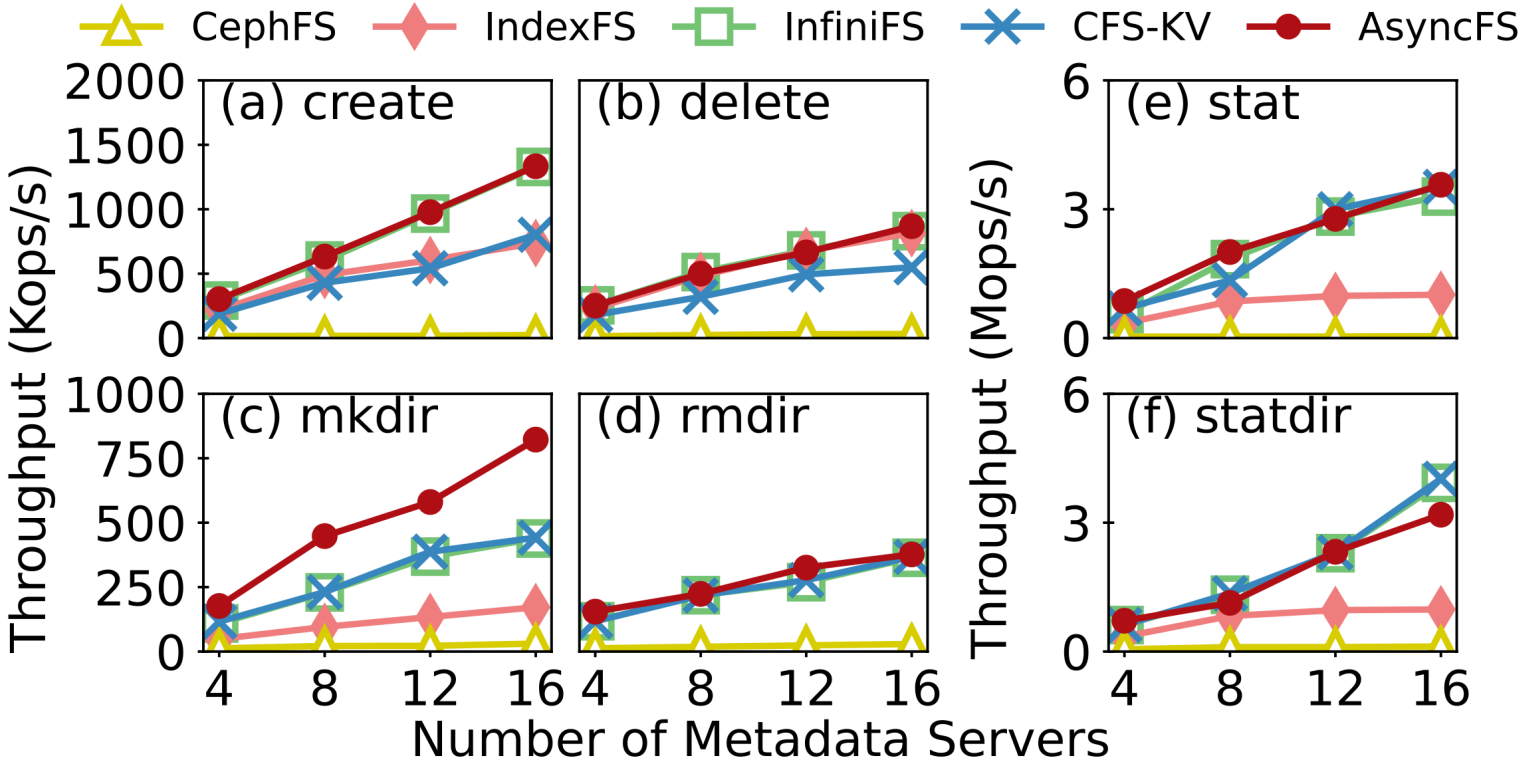



In stat, AsyncFS and CFS-KV can scale out for their **P/C separating** design!
In create/delete, only AsyncFS can scale out for its **scatter/gather** design!

# 3.2. Overall Performance



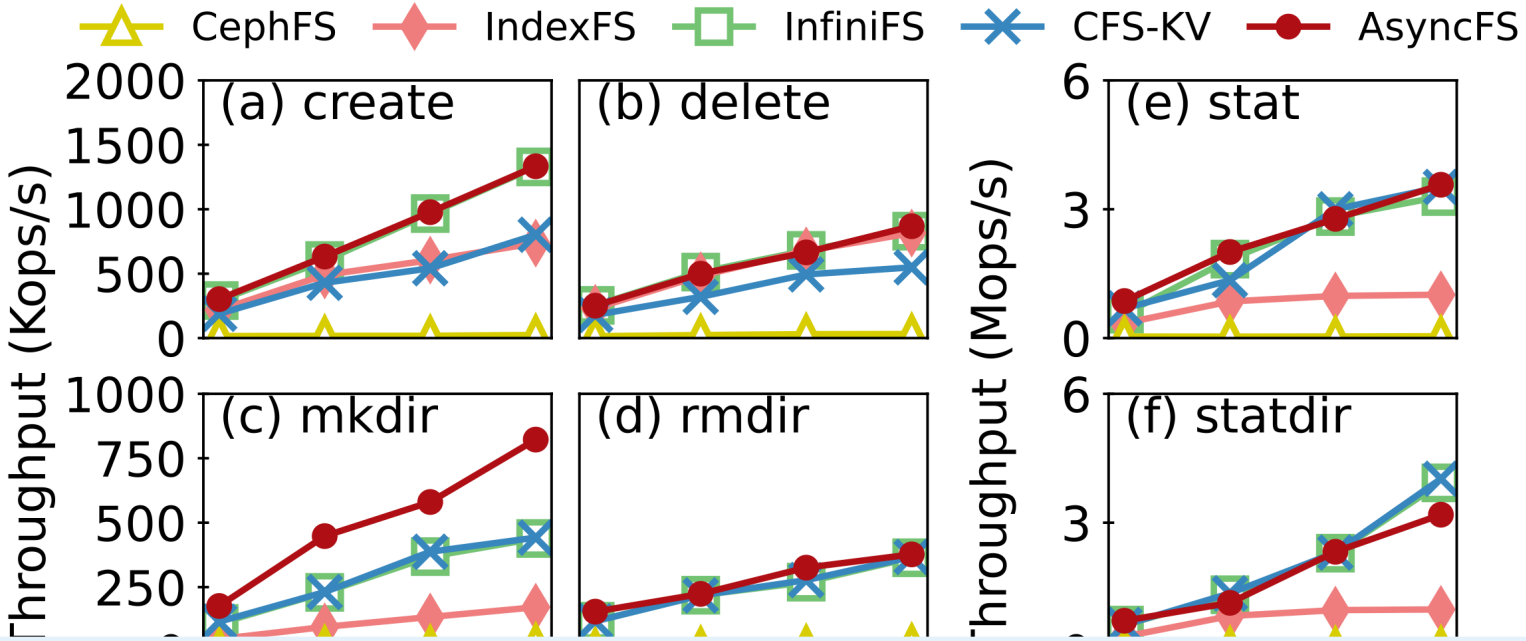
- [**Contention**] Operations under 1024 directories (1024 × 0.1 million).



# 3.2. Overall Performance



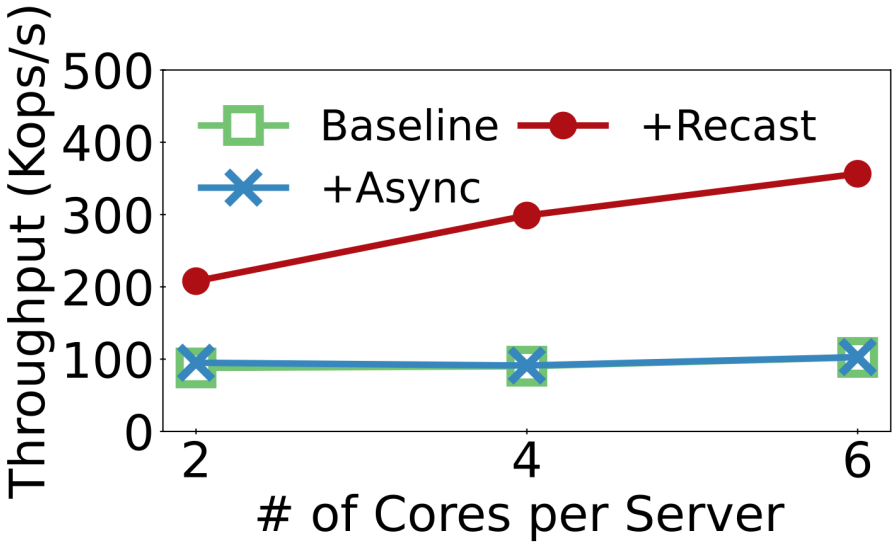
- [**Contention**] Operations under 1024 directories (1024 × 0.1 million).

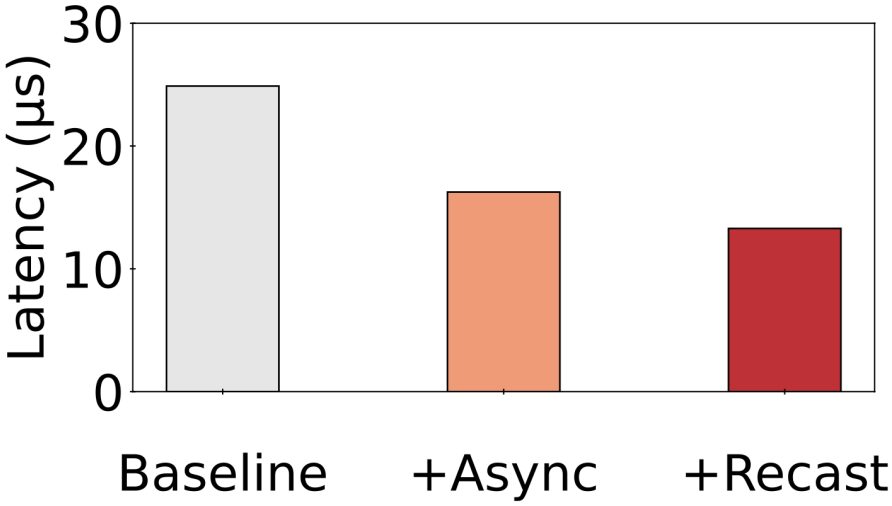



The **Change-Log Recast** in AsyncFS and **locality** in InfiniFS can alleviate contention.

# 3.3. Contribution Breakdown





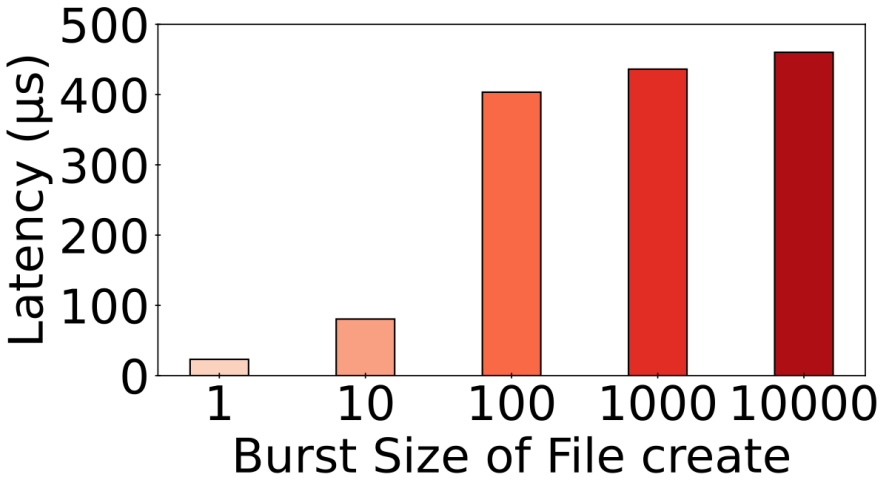



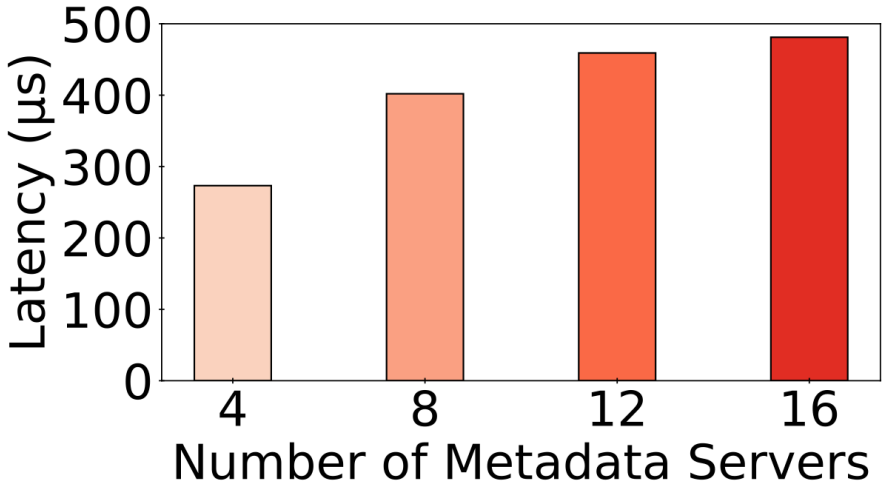

- **+async** can reduce latency.
- **+recast** can improve the throughput.

# 3.4. Directory Aggregation Overhead



- Perform **`statdir`** after burst creating files.



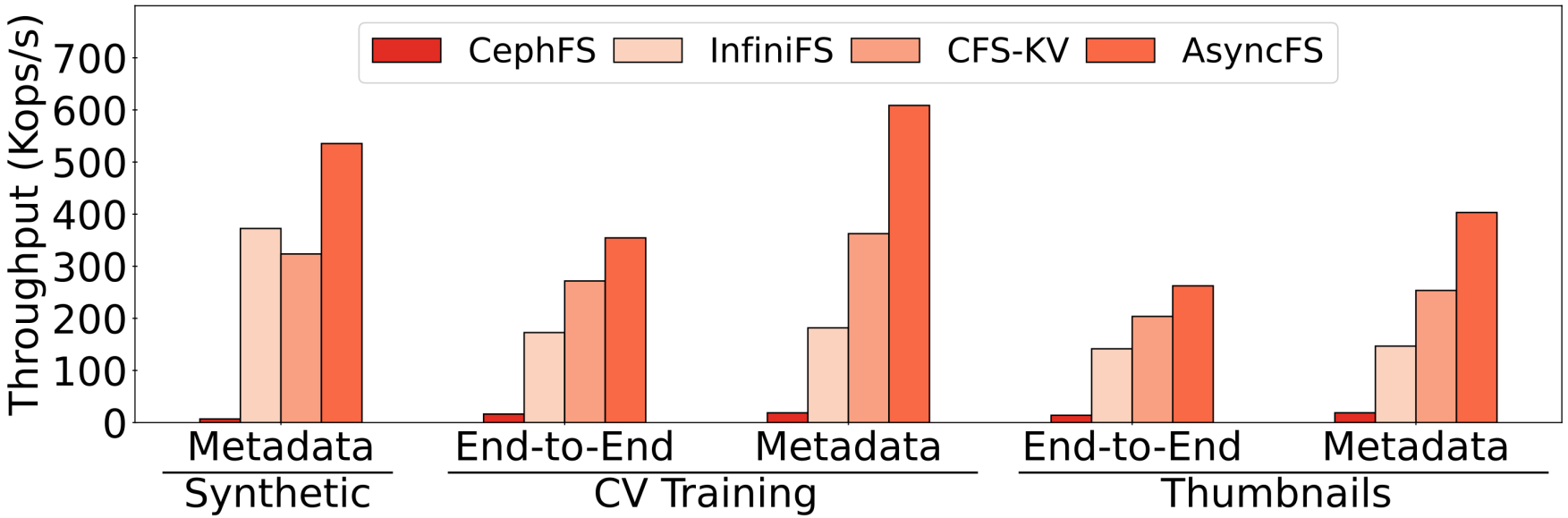




The number of entries in Change-Log to apply:

1. Increase burst size
2. Increase metadata server

# 3.5. End-to-End Performance






CV training and Thumbnails

- processing small images (i.e., 256KiB)

Synthetic workload

- generated based on operation ratios from PanguFS.

# 4. Discussion



This paper proposes AsyncFS, including a distributed metadata service with asynchronous metadata updates.

**Pros.**

- Achieve both load balance and low update overhead.
- Give an important insight: directory attributes updated can be delayed.

**Cons.**

- Increased overhead of statdir.
- Based on a centralized server to track dirty/clean state of directories.

# Thanks