BATCHFS

Scaling the File System Control Plane with Client-Funded Metadata Servers

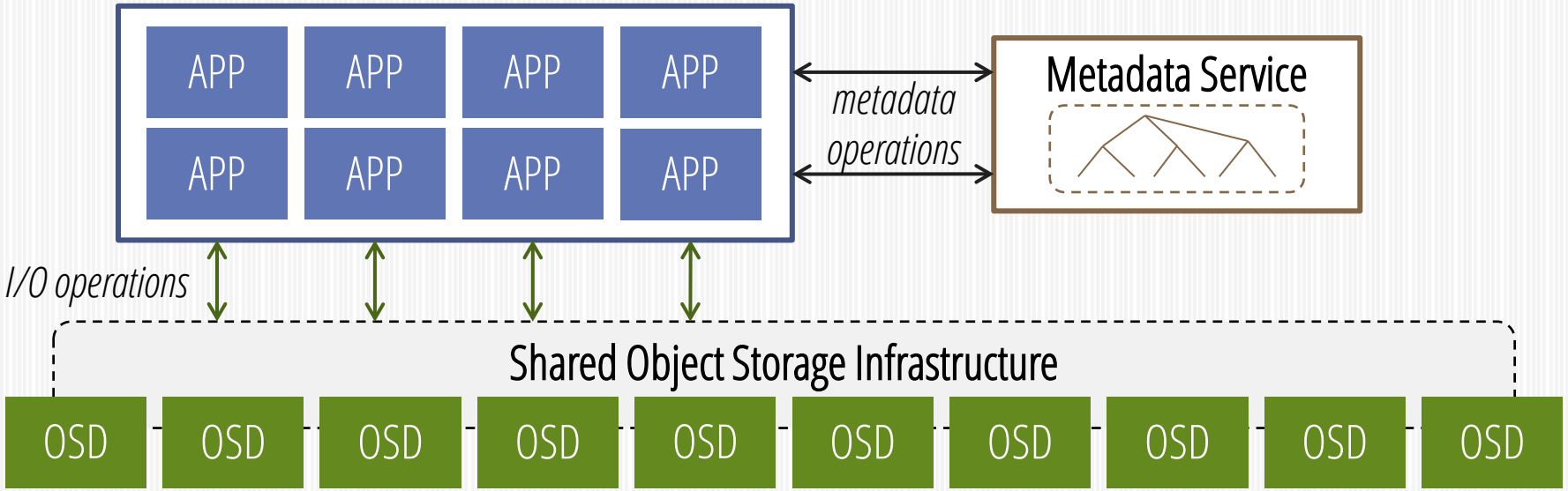
[vision-paper]

Qing Zheng, Kai Ren, Garth Gibson

Carnegie Mellon University

9th Parallel Data Storage Workshop/SC 2014

File System Architecture



Data path is parallel but metadata path is not necessarily.

Reality

Data scales; METADATA is hard to scale, especially in HPC data centers

Programmers like POSIX SEMANTICS, which limits linear scalability

How We Scale the Metadata

IndexFS

[SC14, Tue, 2:30pm, Room 393-94-95]

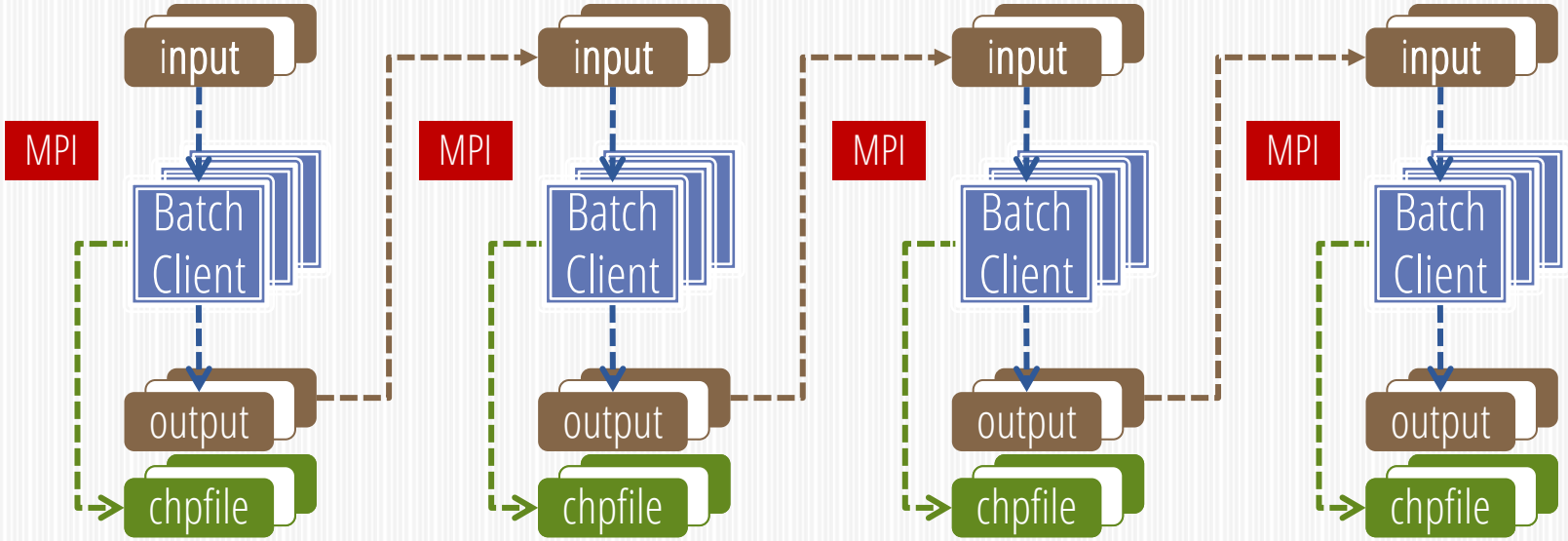
Two orders of magnitude faster than Lustre/PVFS

BatchFS Scale another order of magnitude

Section One

BATCH APPLICATION

Batch Applications



Batch apps are self-coordinated by MPI and workflow engines

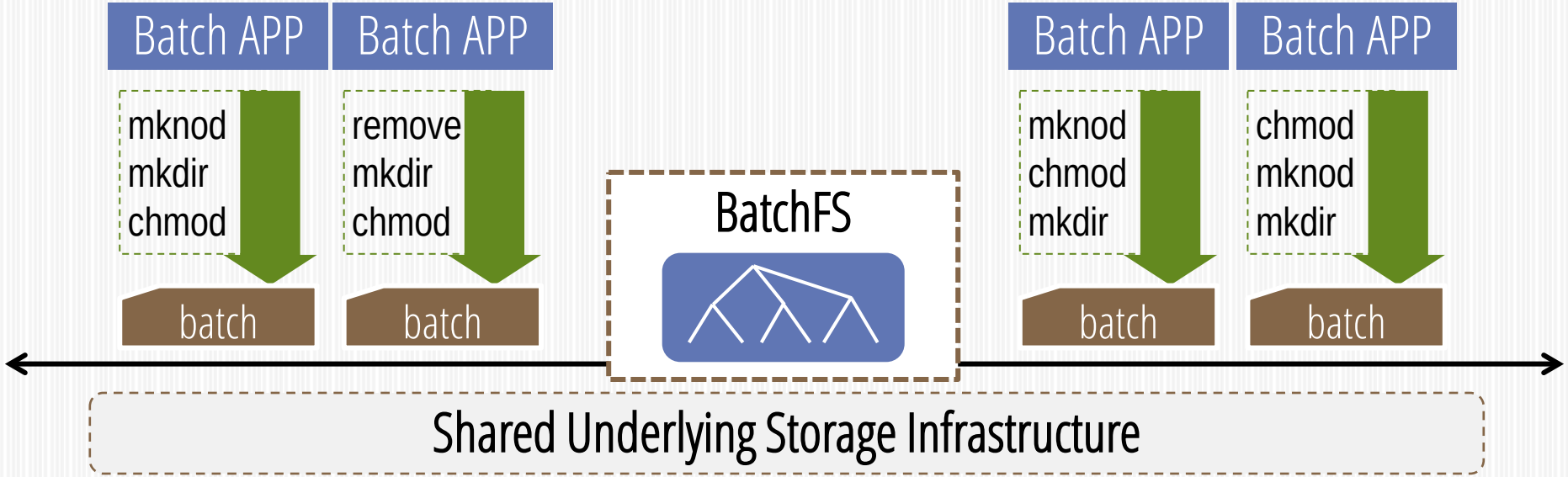
Key Observation

Batch apps DON'T need FS to communicate

SYNCHRONOUS and SERIALIZED metadata management is OVERKILL for batch apps

Introducing BatchFS

Deep batching for high throughput



BatchFS Philosophy

From per-op *to* per-batch *synchronization*

From server-side *to mostly* client-side *processing*

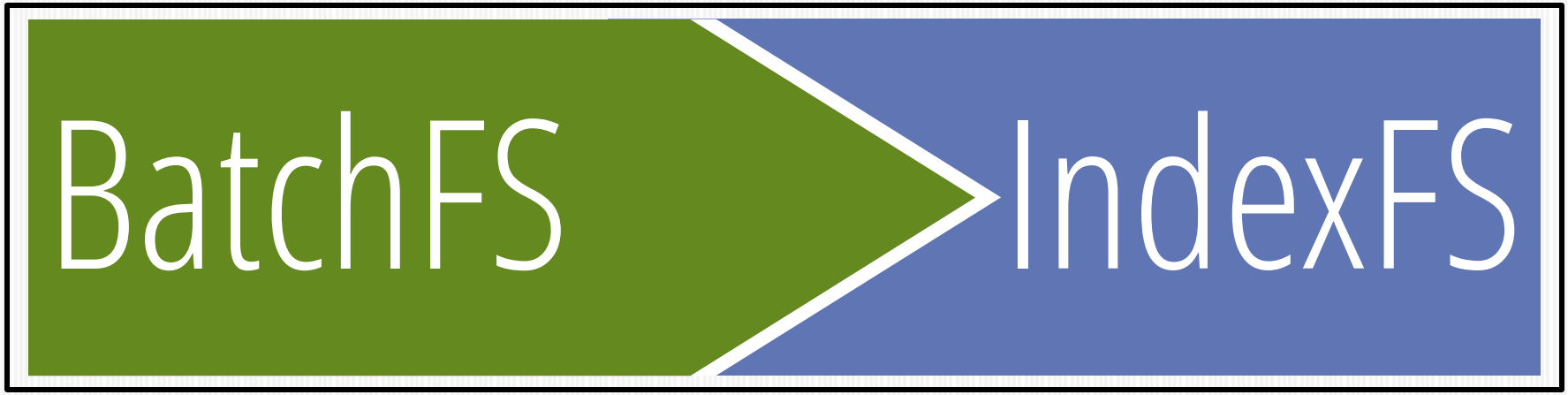
→ CLIENT-FUNDED metadata architecture

Section Two

BATCHFS

BACKGROUND

Background



BatchFS is designed as an extension of IndexFS

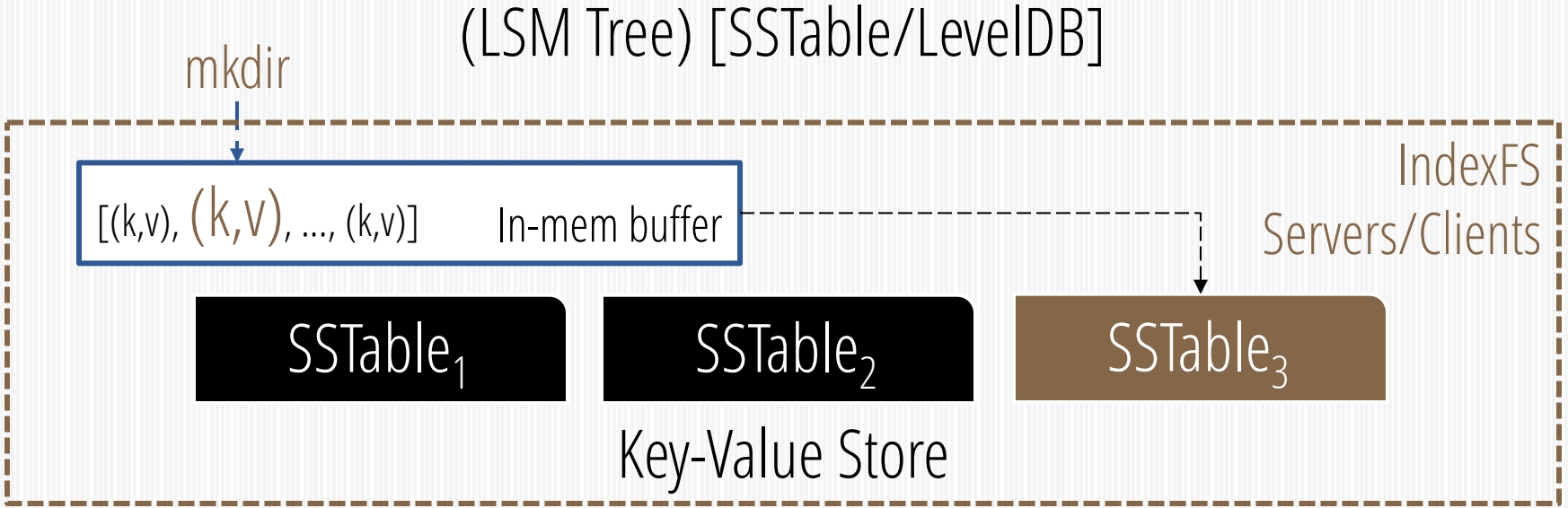
[SC14, Tue, 2:30pm, Room 393-94-95]

inheriting its metadata representation to enable high-performance metadata processing

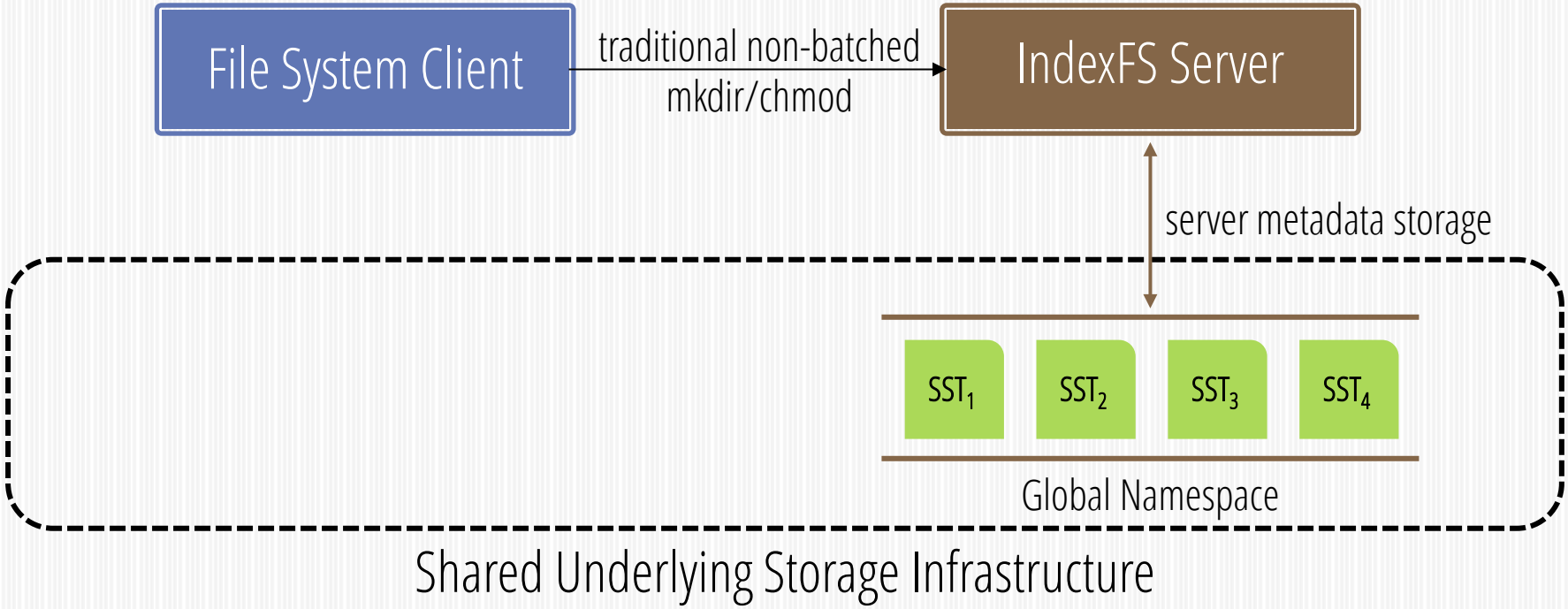
Metadata Representation

Log-structured and indexed data structure

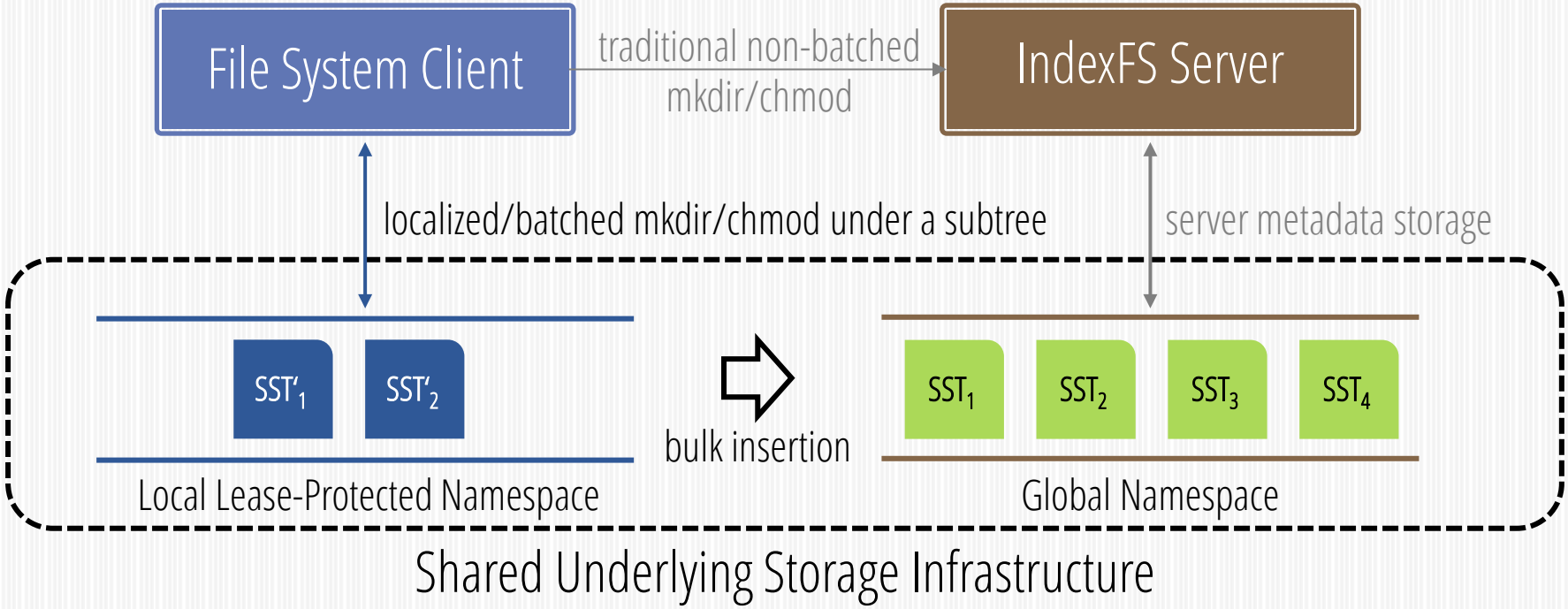
(LSM Tree) [SSTable/LevelDB]



Client-Server Interaction



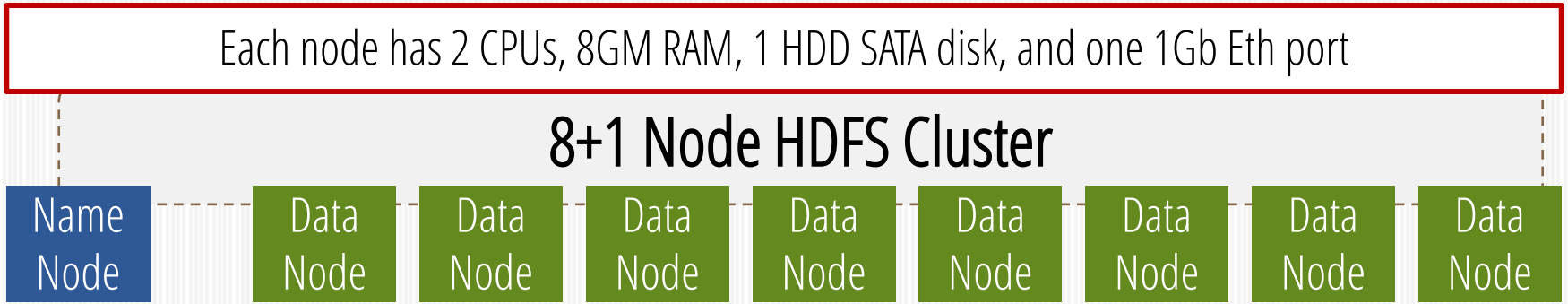
Metadata Bulk Insertion



Preliminary Results

A prototype of BatchFS as an IndexFS [SC14] feature

metadata bulk insertion (batching)

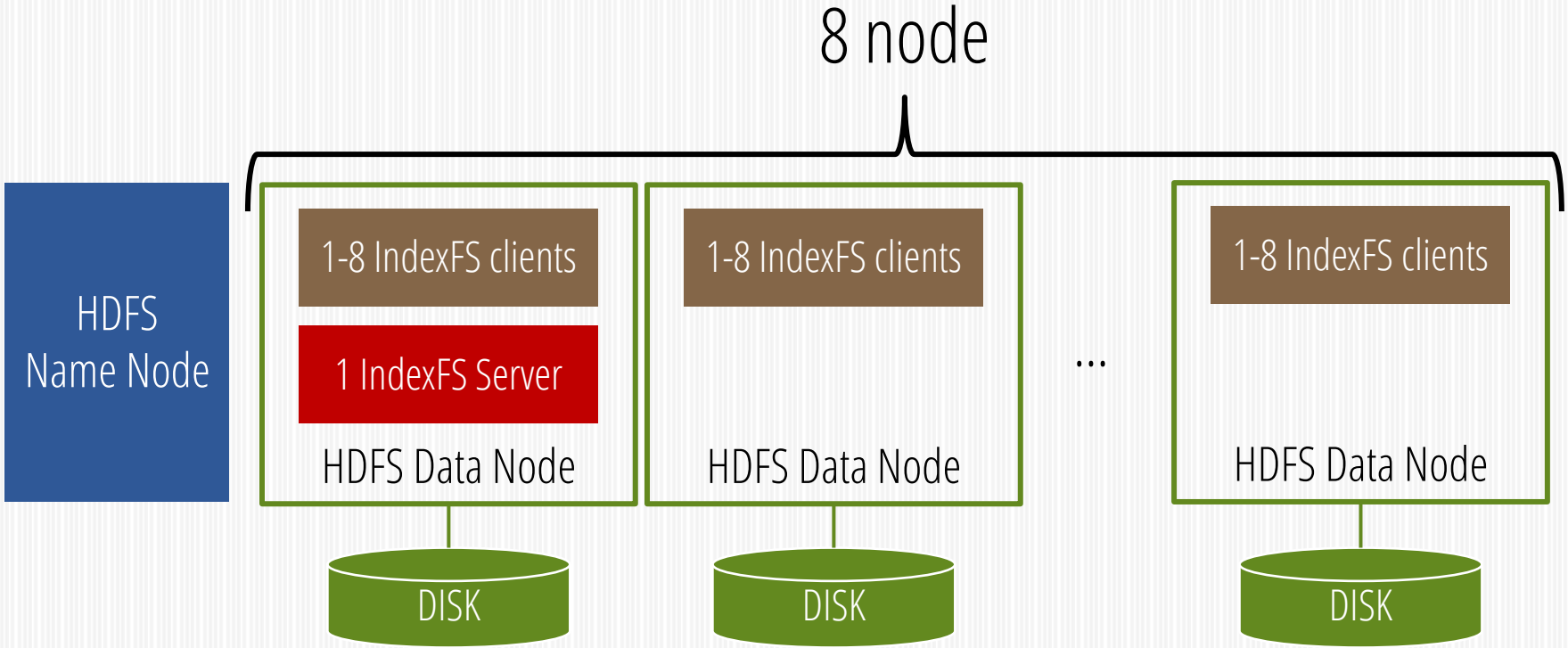


Workload

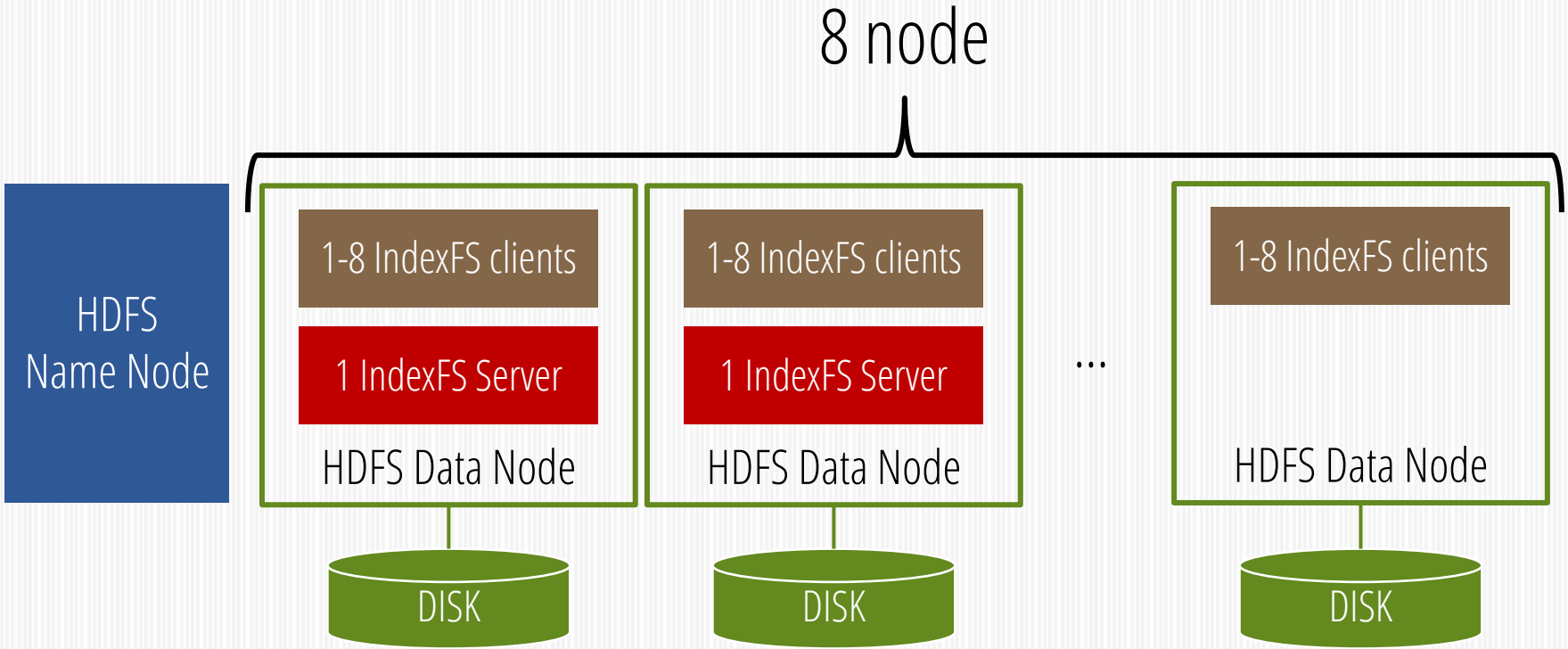
Each client process creates 1 private directory *(8-64 client processes on fixed 8 nodes)*

Clients insert empty files into their own directories *(in total 1 million * #servers files)*

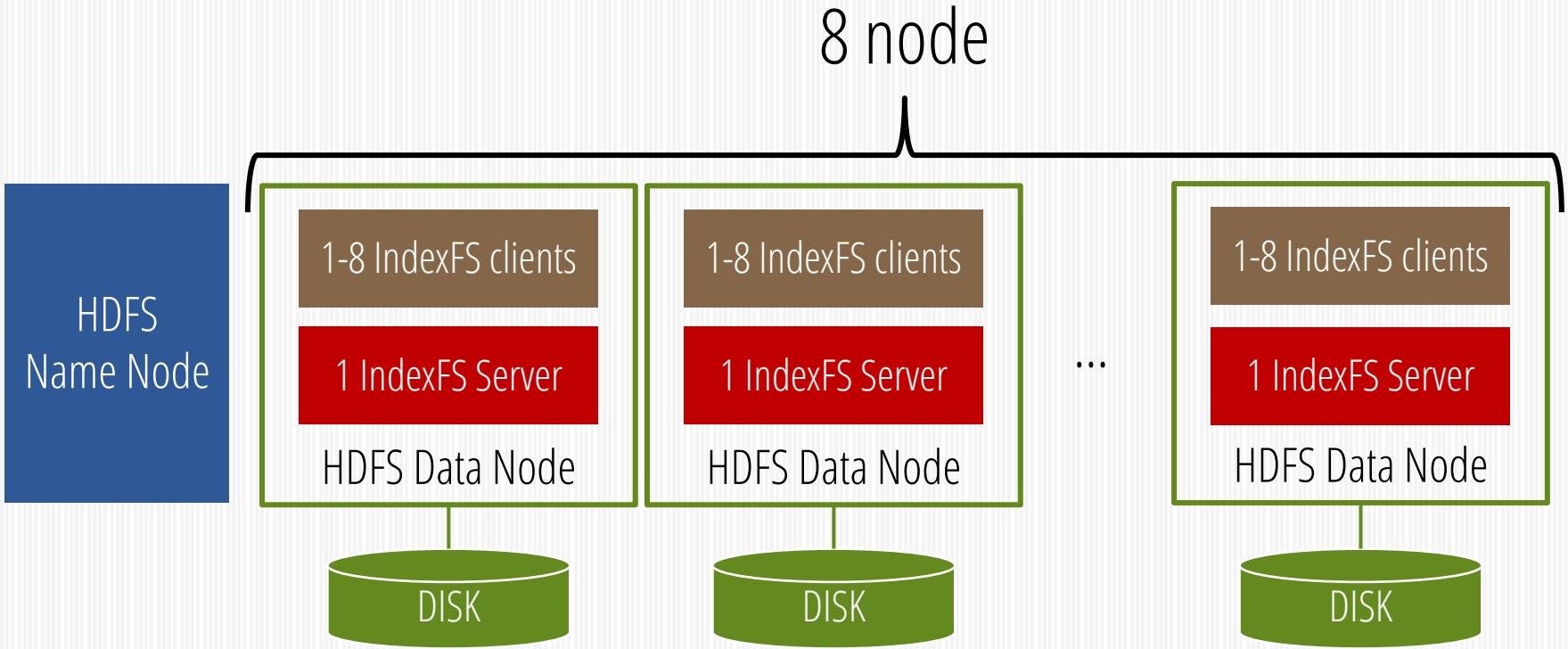
Experiment Setup #1



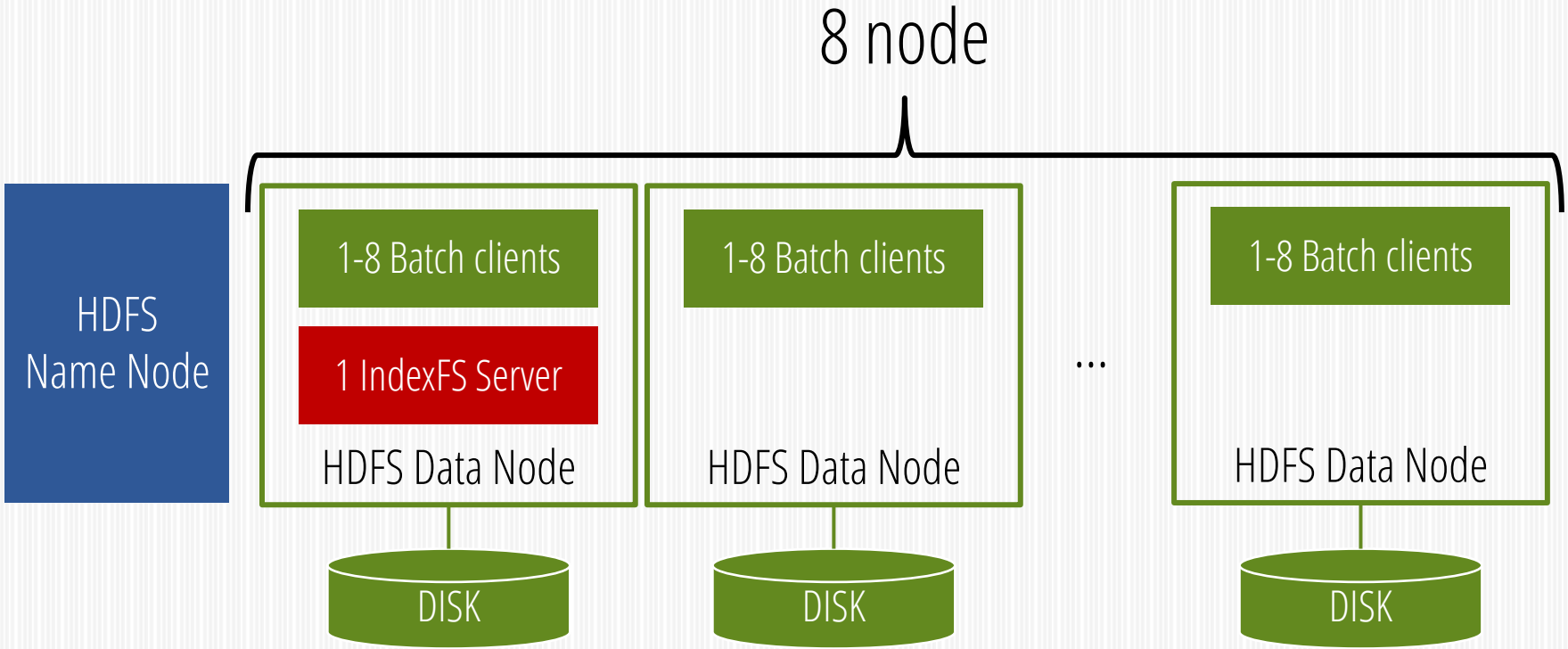
Experiment Setup #2



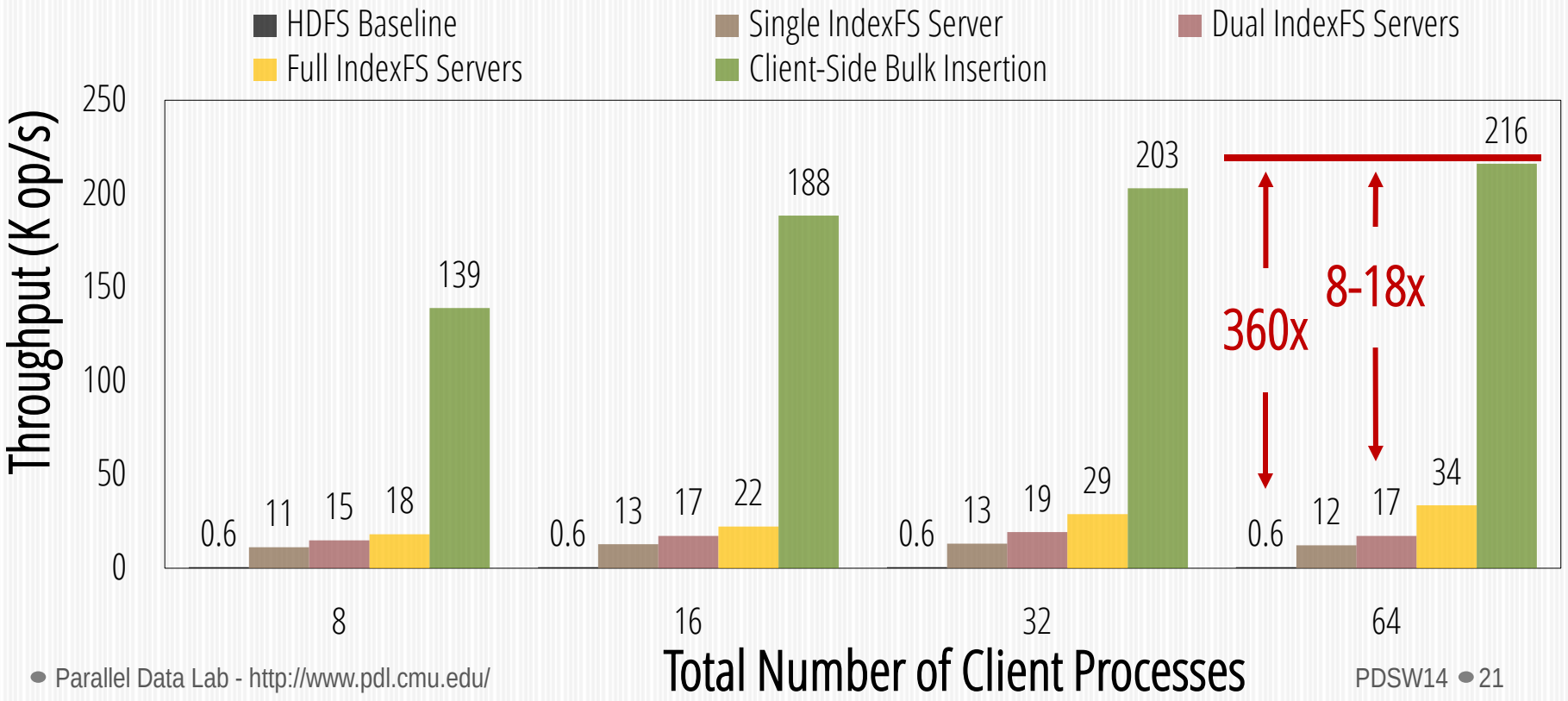
Experiment Setup #3



Experiment Setup #4



8x-360x Perf. Improvements



Section Three

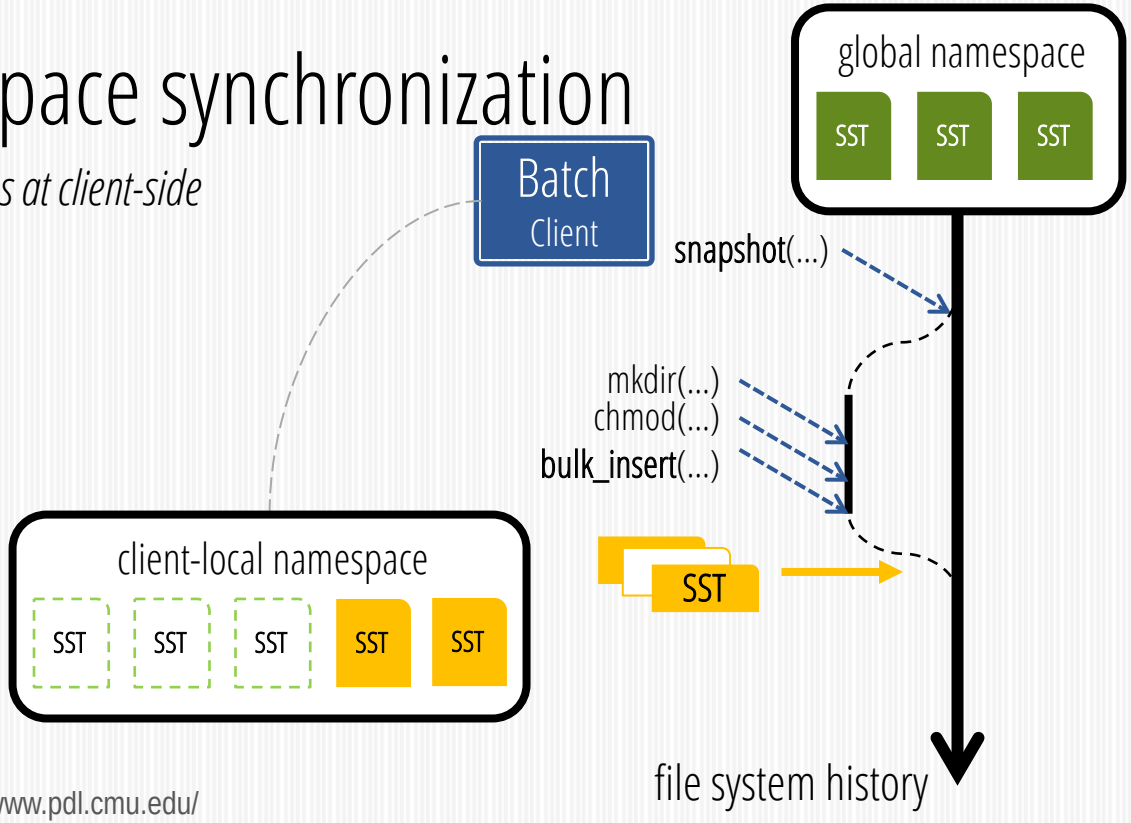
BATCHFS

DESIGN

Deep Metadata Batch

Lazy namespace synchronization

Pre-execute metadata ops at client-side



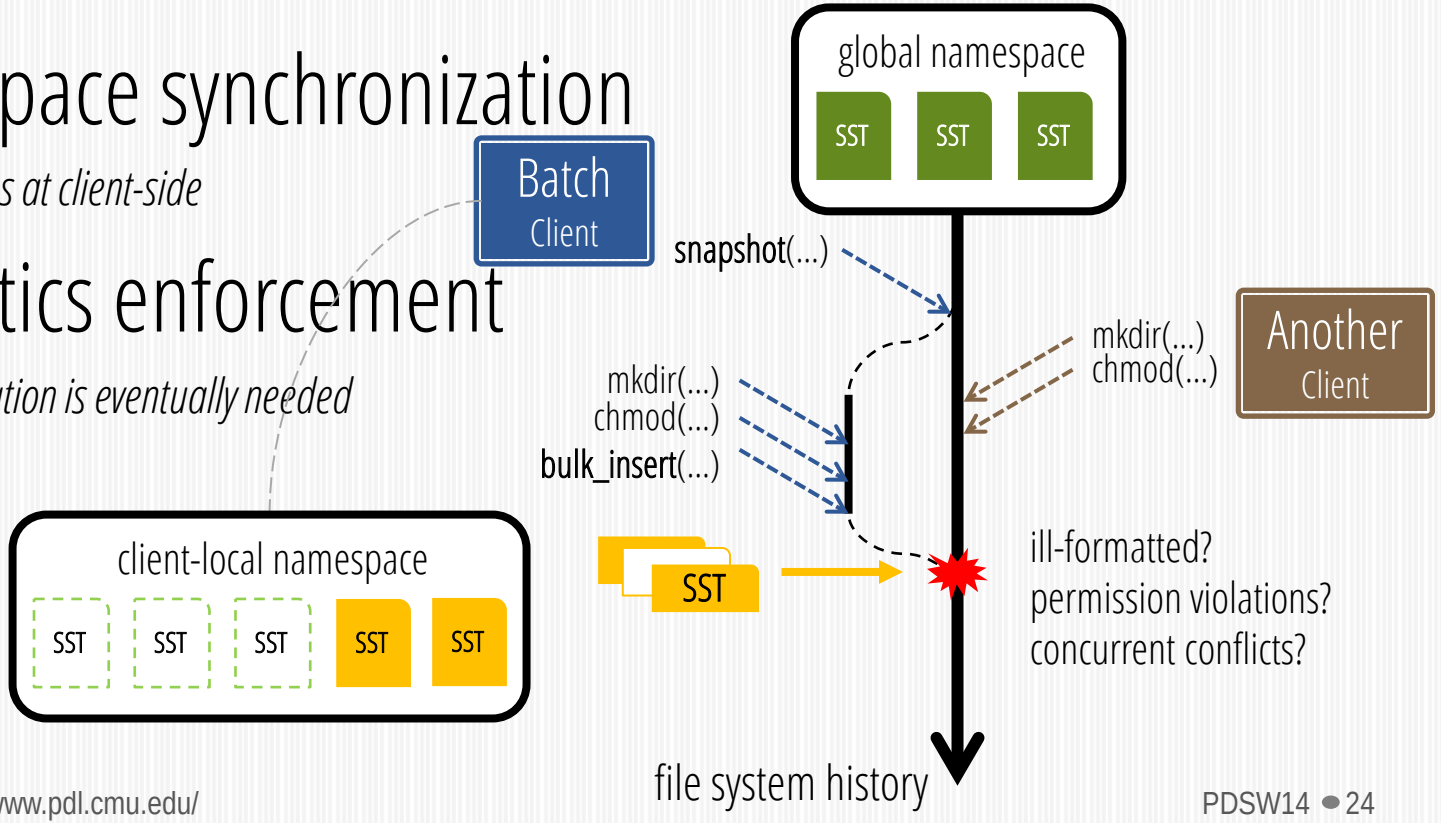
Deep Metadata Batch

Lazy namespace synchronization

Pre-execute metadata ops at client-side

Lazy semantics enforcement

Delayed until synchronization is eventually needed



BATCHFS

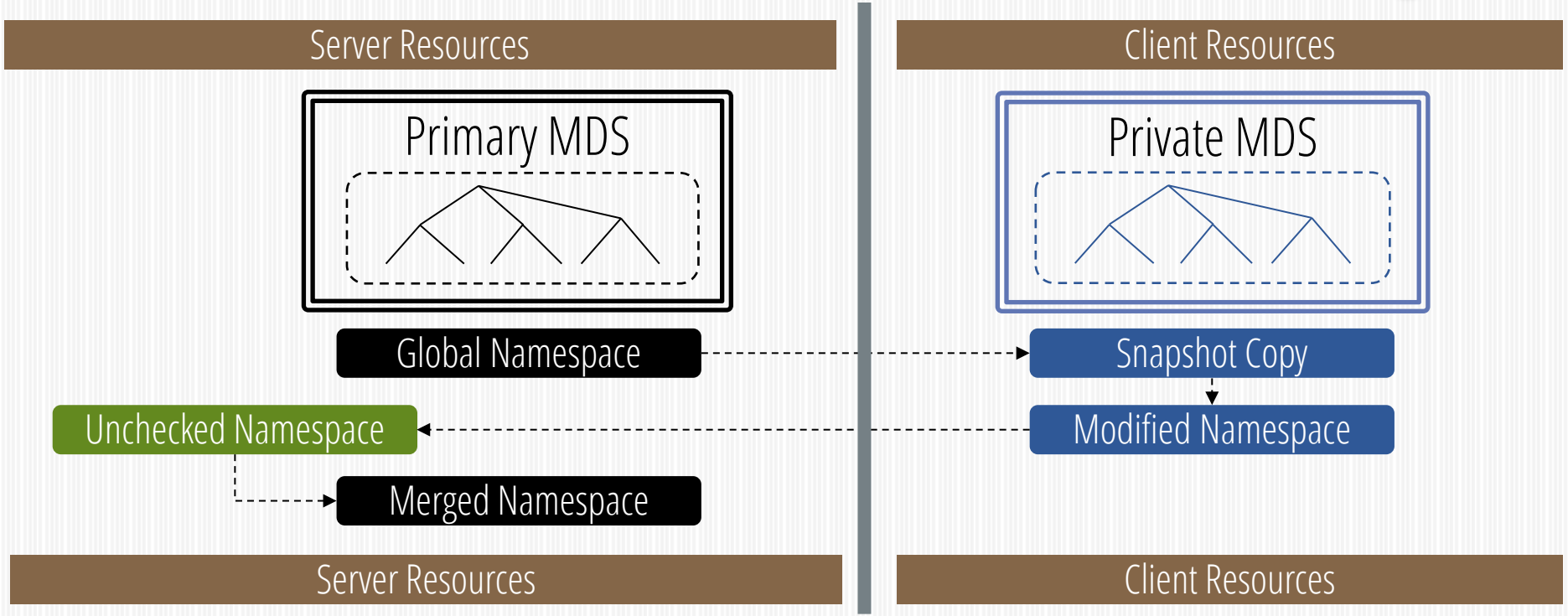
[PDSW14]

INDEXFS

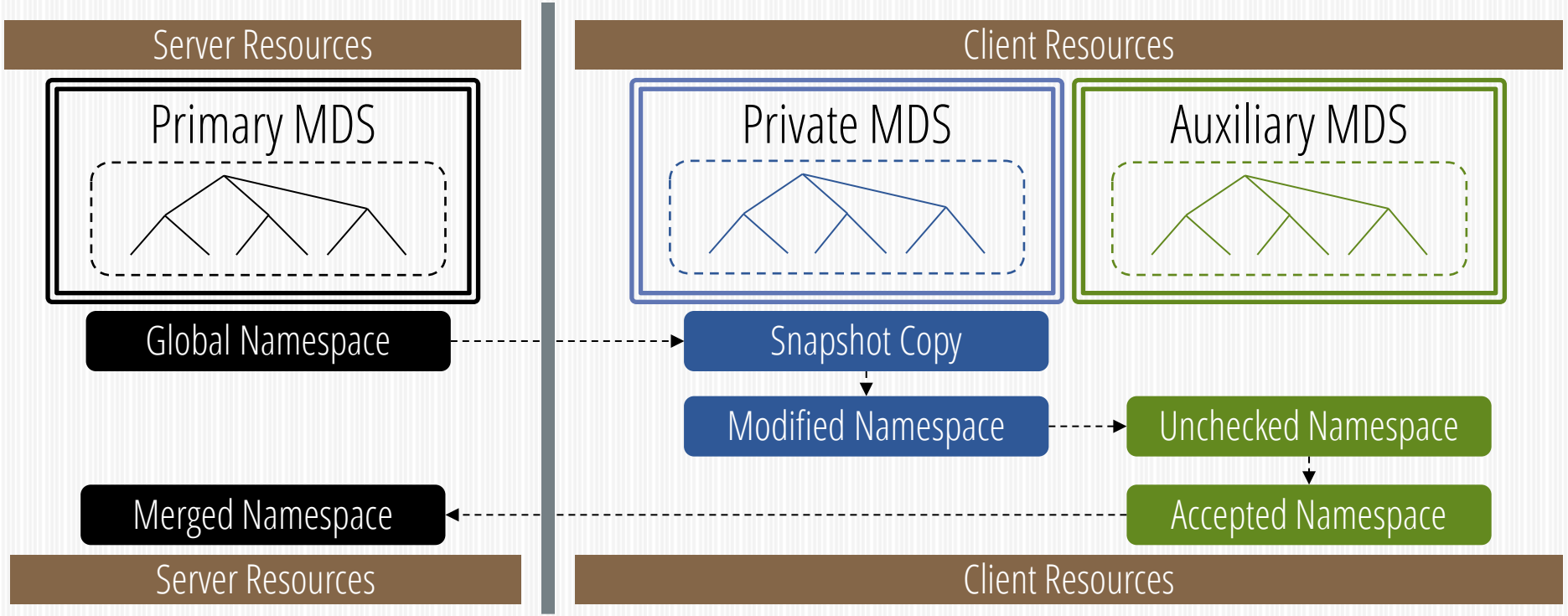
[SC14]

Snapshot of a subtree	Empty subtree
Concurrent access	Exclusive access
Optimistic concurrency control	Protected by server-issued leases
No timeout	Lease expires
Snapshot reads w/ access control	Empty subtree

Cliend-Funded Metadata Processing



Cliend-Funded Metadata Verification



Section Four

FUTURE WORK

Conflict Resolution

Who is responsible? What's the semantics?

A) DB-like, read/write sets, transactional

B) Bayou-like, auto resolution, domain rules

C) Coda-like, resolved by human

Self-probable Metadata

For clients to generate proofs of the correctness of their namespace mutations

A) operation log (possibly compressed)

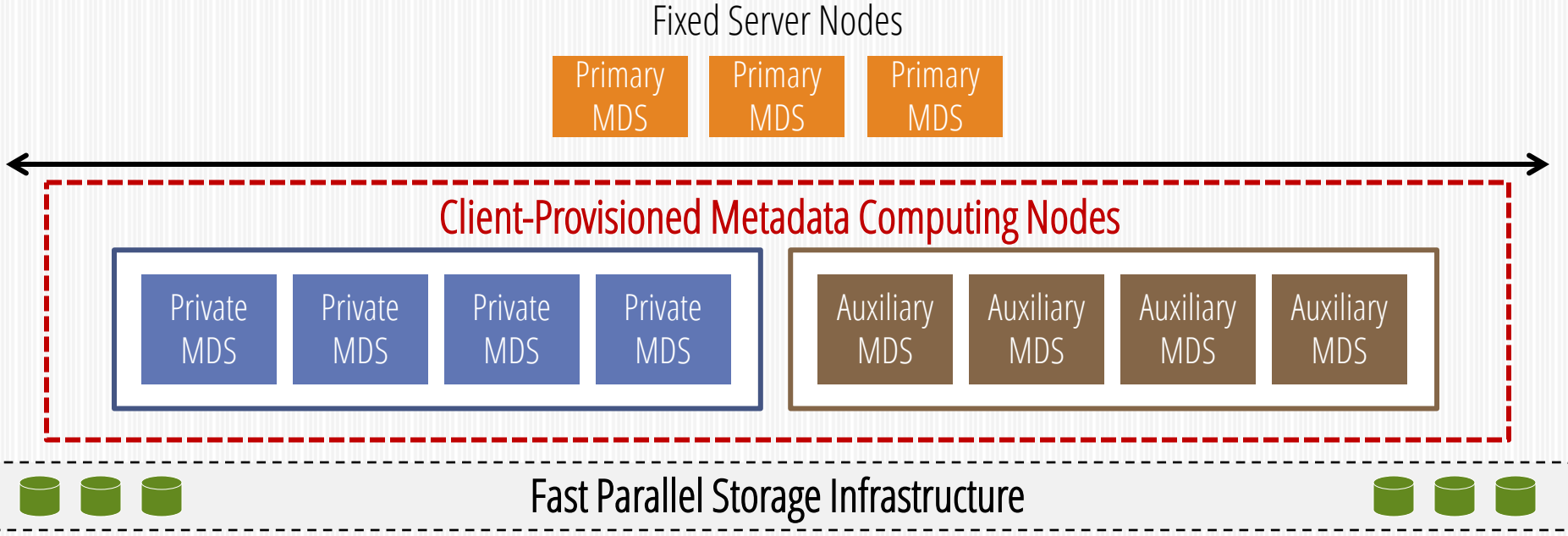
B) logic-based proof (proof-carrying-code)

Conclusion

BATCHFS

At least one RPC per operation	✖
Inefficient metadata representation	✖
Pessimistic concurrency control	✖
Synchronous metadata interface	✖
Dedicated authorization service	✖

BatchFS Architecture



BatchFS scales with the number of client nodes.

Reference

BATCHFS | Scaling the File System Control Plane with Client-Funded Metadata Servers (PDSW14)

INDEXFS | Scaling File System Metadata Performance with Stateless Caching and Bulk Insertion (SC14)

QUESTIONS

IDEAS AND FEEDBACK

BACKUP SLIDES

Access & Quota Control

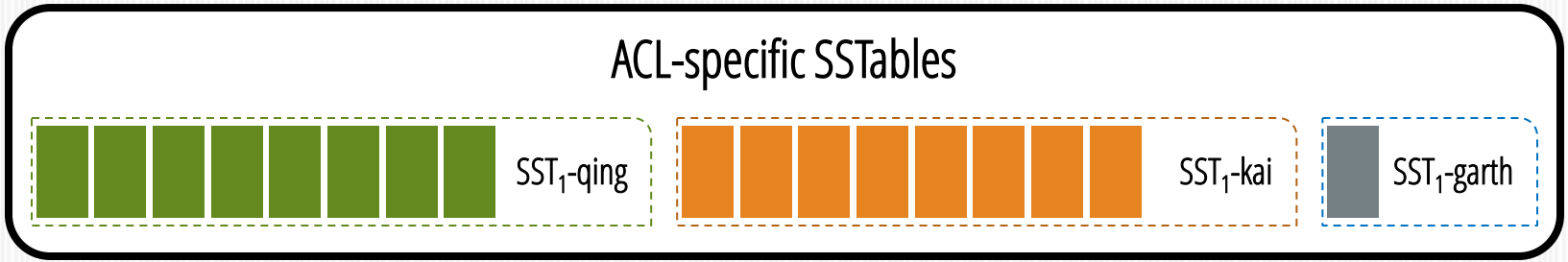
Access enforced by OSD

No quota control for metadata

Quota control on data provided by OSD

Snapshot Access Control

Underlying Parallel File System // Access Ctrl // Quota Mng



Trustworthiness

How to trust auxiliary metadata servers?

A) just trust these servers

B) trusted VM running the right code

C) hardware co-processors such as TPM