A Case for Scaling HPC Metadata Performance through De-specialization

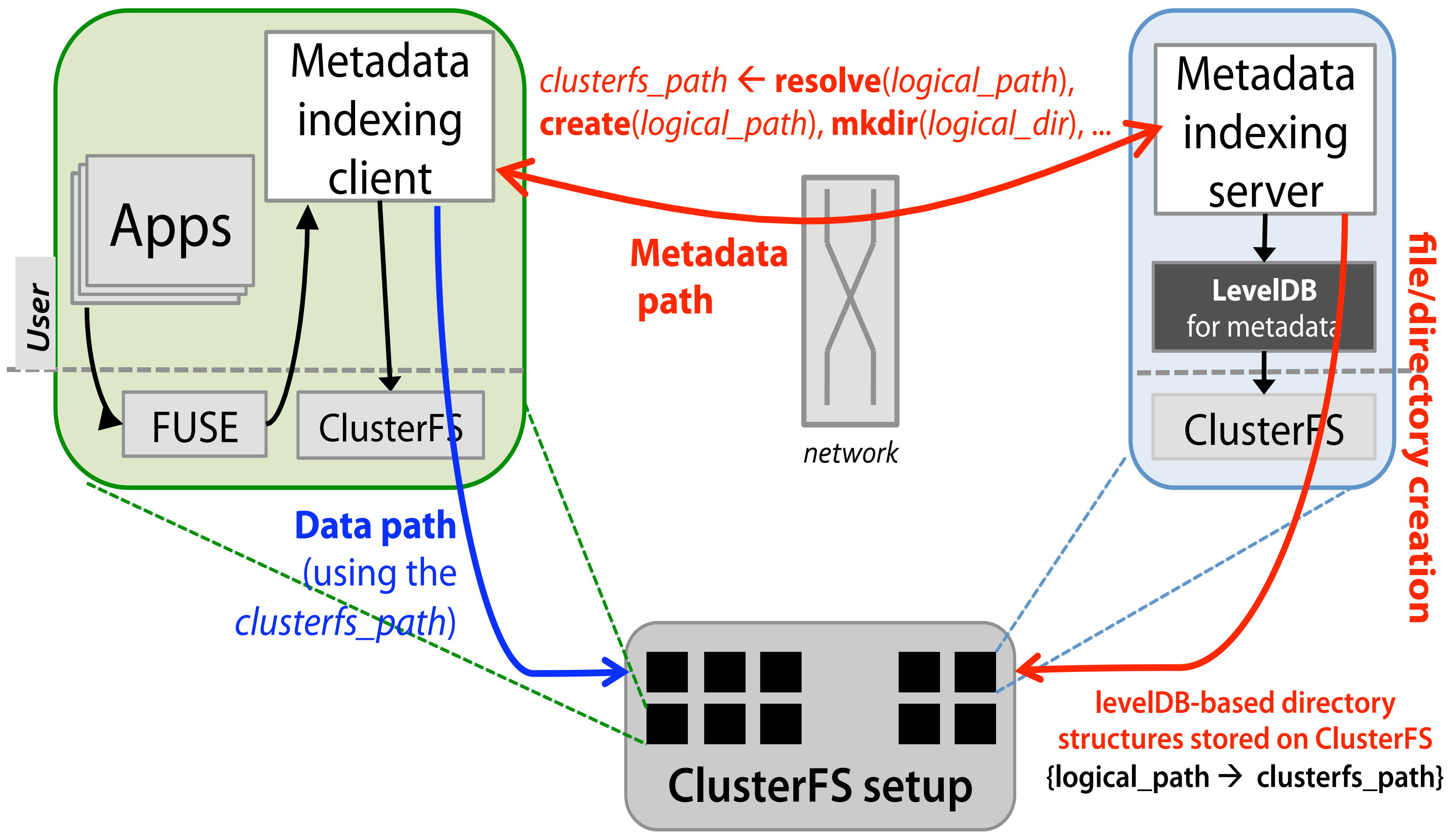
Swapnil Patil, Kai Ren, Kartik Kulkarni, Garth Gibson (Carnegie Mellon University)

Overview

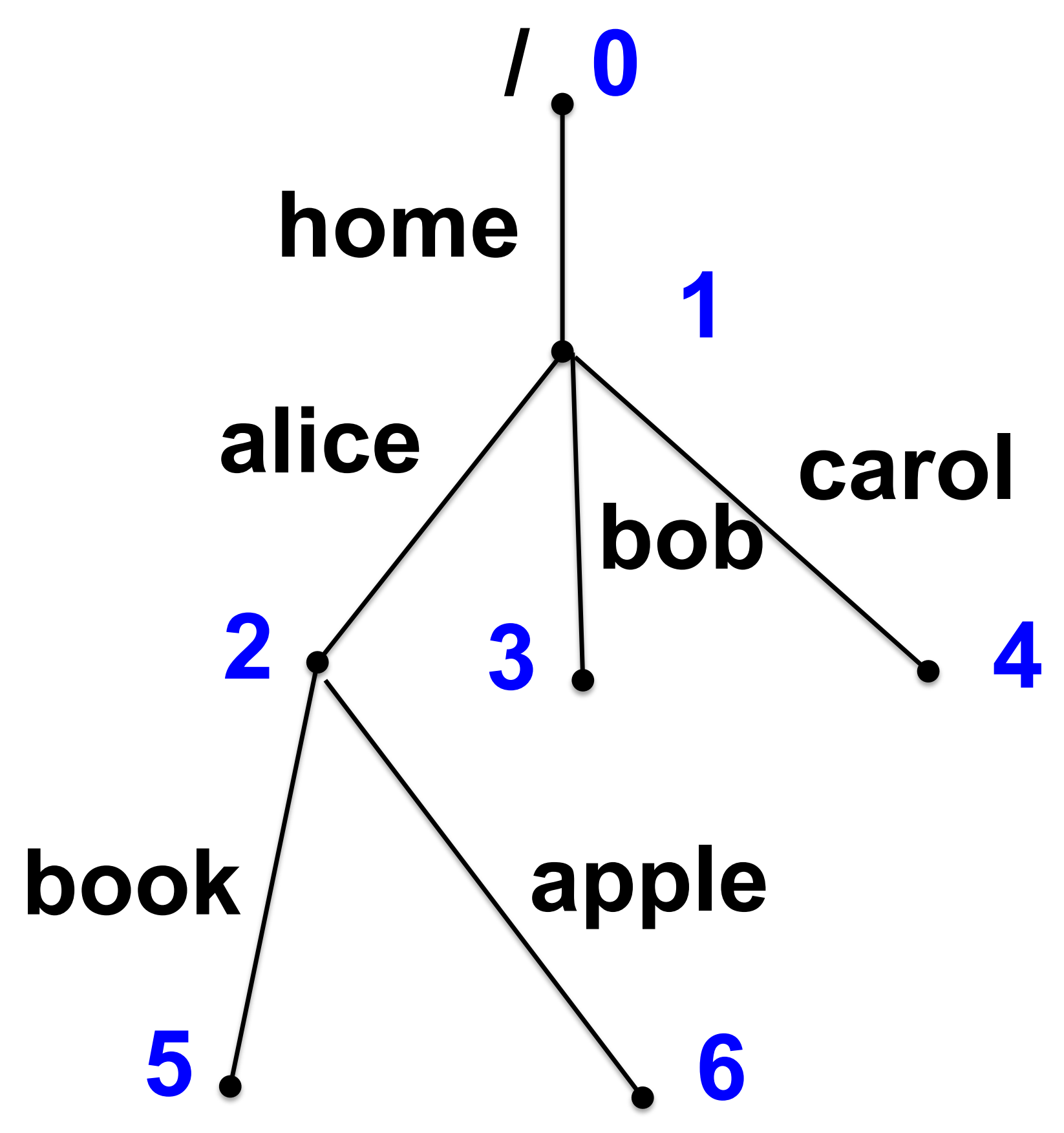
- **Problem**: Prevalence of non-scalable metadata servers
- **Approach**: Parallel directory indexing (GIGA+) for distribution + packed metadata data server (TableFS)
  - Layer on existing cluster file systems without any modifications
  - De-specialization: hide a lot of metadata (dir ents, inodes, etc) from cluster file system

Design and Implementation

- **GIGA+:** partitions, indexes, and distributes directories over multiple servers [Patil11]
- **TableFS** uses LevelDB to pack and order directory entries & inode info on-disk [Ren12]
- FUSE-based Giga+TablesFS shards metadata over servers and TableFS to pack it into cluster file system
- Giga+ splits shards to load balance: TableFS extentions pass metadata sets via LevelDB bulk insert


Metadata indexing client
Apps
User
FUSE
ClusterFS
clusterfs_path ← resolve(logical_path),
create(logical_path), mkdir(logical_dir), ...
Metadata path
network
Metadata indexing server
LevelDB for metadata
ClusterFS
file/directory creation
Data path (using the clusterfs_path)
ClusterFS setup
levelDB-based directory structures stored on ClusterFS
{logical_path → clusterfs_path}

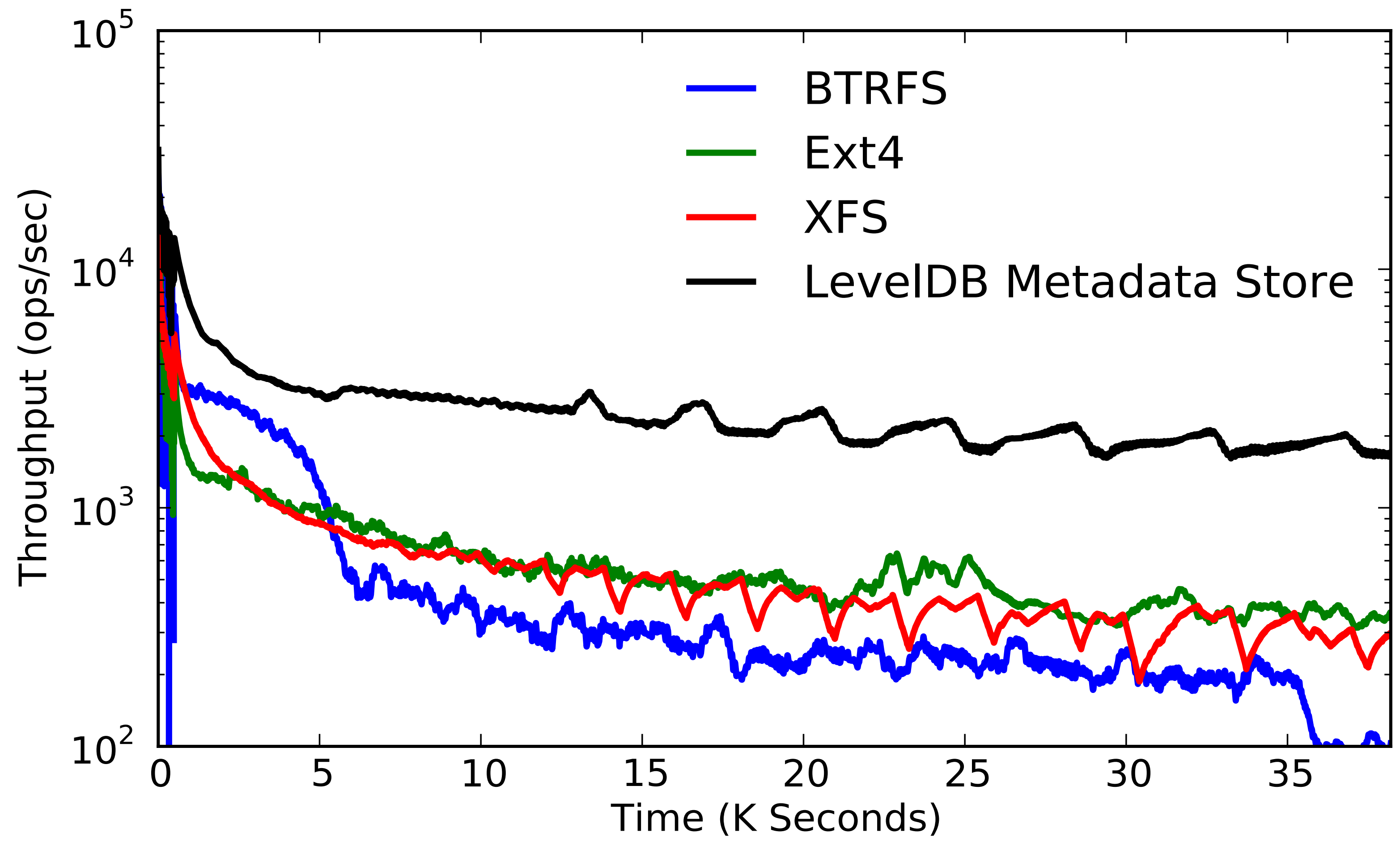
- **TableFS** stores files and directories as key-value pairs in LevelDB (a variant of log-structure merge tree).
- Key is <parent inode number, hash(filename)
- Value: filename, inode attributes, symbolic link



Key	Value
<0,hash(*home*)>	1, "home", *stat*
<1,hash(*alice*)>	2, "alice", *stat*
<1,hash(*bob*)>	3, "bob", *stat*
<1,hash(*carol*)>	4, "carol", *stat*
<2,hash(*book*)>	5, "book", *stat,* File pointer
<2,hash(*apple*)>	6, "apple", stat, File pointer

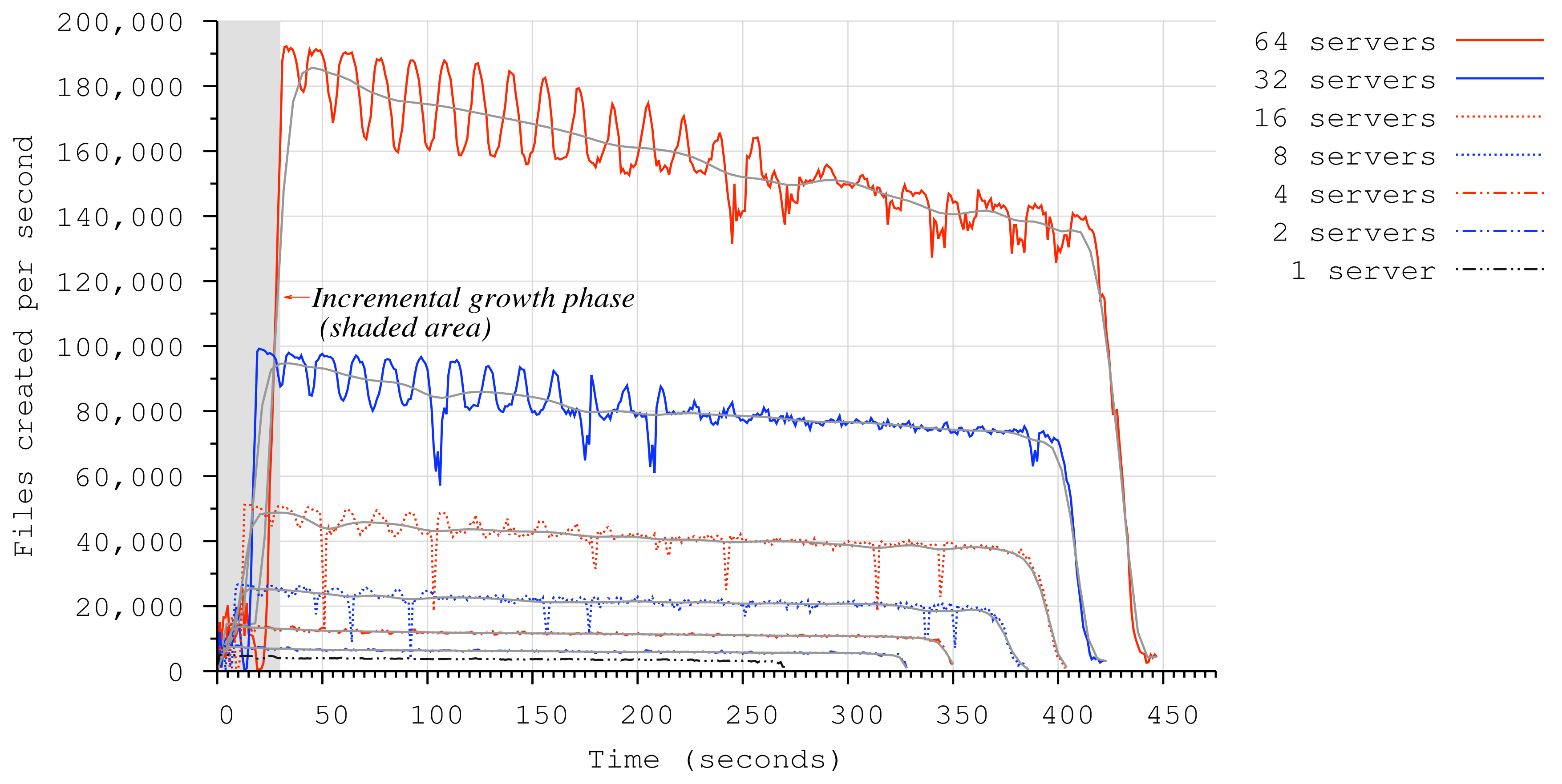
Preliminary Evaluation

- **Single Node Performance**: TableFS outperforms local file systems for metadata-intensive workloads by up to ten times.



Setup: 64-node cluster where each machine has 16GB RAM, one 2TB disk with 1 GigE NIC. Initially "cluster FS" is local disk, mostly, and NFS for splitting shards

Scalability: For a zero-byte file creation workload, Giga+TableFS prototype scales up to 64 servers delivering ~160,000 file creates per second



Ongoing work – PanFS layering

Decoupling data and metadata paths

-- non-open file ops follow FUSE to Giga+TableFS

-- open big file sym links to PanFS for bandwidth

-- bypass implies Giga+TableFS metadata gets stale

-- one goal is to modify FUSE kernel module to always do redirection (not just return sym link) and replicate at least file close syscall

Sustaining high creation rates for large files

-- Delay file creates on PanFS until file is large

-- hide the latency of file creation during writing