Scaling Metadata Performance for POSIX Applications

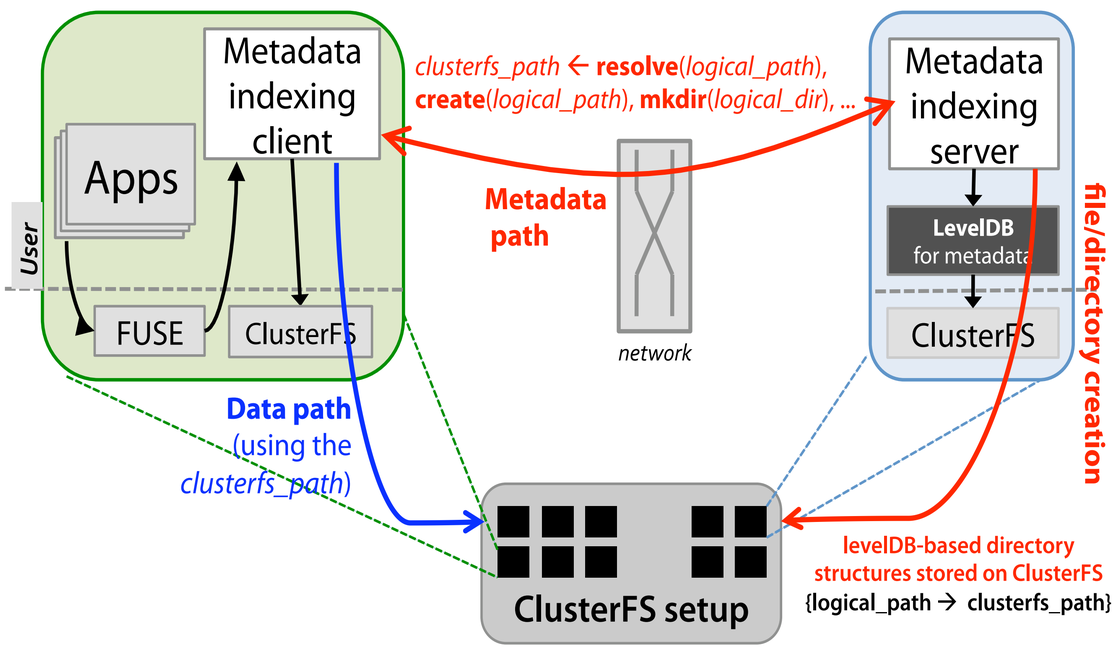
Kai Ren, Kartik Kulkarni, Swapnil Patil, Garth Gibson (Carnegie Mellon University)

Overview

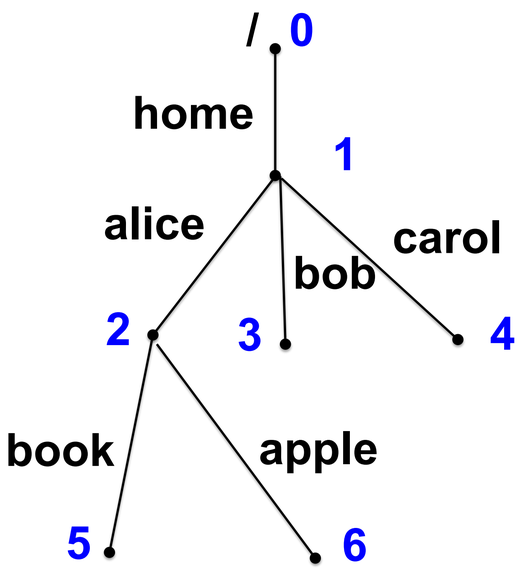
- **Problem**: Scalable metadata service for existing cluster FSs.
- **Approach**: Parallel directory indexing (GIGA+) and high performance metadata layout (TableFS)
  - Layer on existing cluster FSs without any modifications
  - De-specialization approach that uses one representation for all metadata (e.g. directory entries, attributes and etc.)

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 extensions pass metadata sets via LevelDB bulk insert



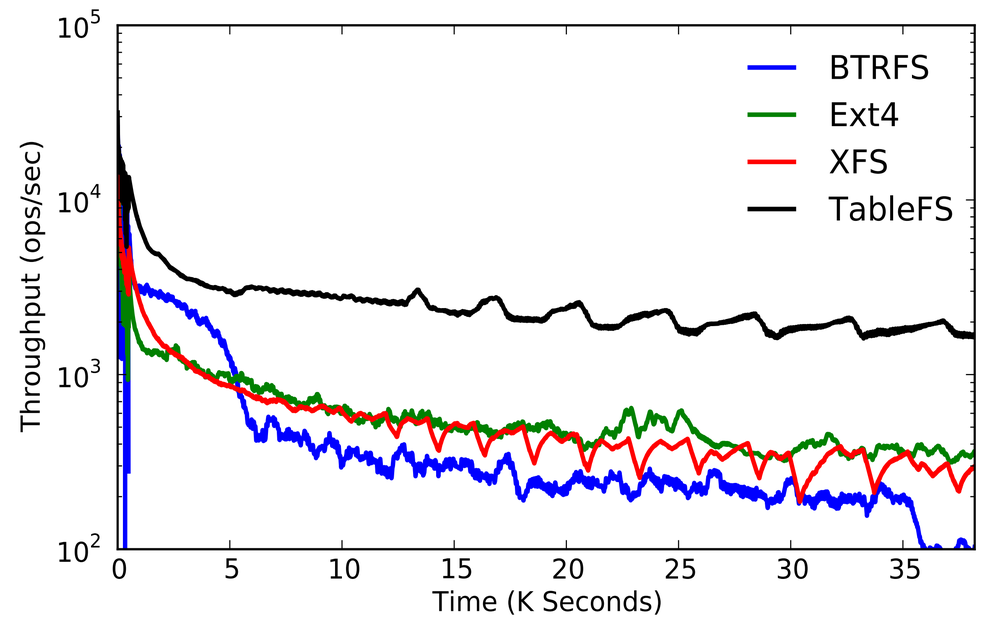
- **TableFS**: stores files and directories as key-value pairs in LevelDB (a variant of log-structure merge tree).
- **Key**: <parent inode number, hash(filename)
- **Value**: filename, inode attributes, inline file data (for small files) or symbolic link (for large files)



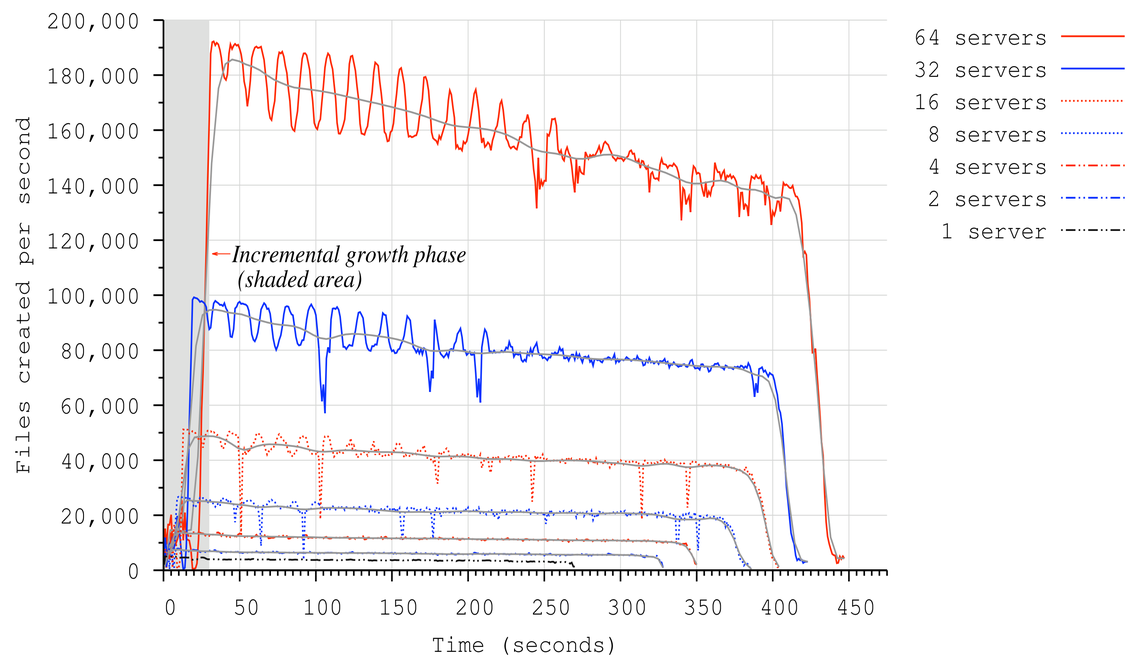
Key	Value
<0,hash(*home*)>	1, "home", *stat*
<1,hash(*alice*)>	2, "alice", *stat*
<1,hash(*bob*)>	3, "bob", *stat*
<1,hash(*carol*)>	4, "carol", *stat*
<2,hash(*book*)>	5, "book", *stat,* Inline file data
<2,hash(*apple*)>	6, "apple", stat, File pointer

Preliminary Evaluation

- **Single Node Performance**: For zeror-bye file creation workload, TableFS outperforms local file systems for metadata-intensive workloads by up to ten times.
- **Machine Setup**: Dual Core, 16GB RAM, and 2TB hard disk.



- **Setup**: 64-node cluster with 1 GigE NIC. Initially "cluster FS" is local disk, and uses 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

Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University · Georgia Tech · intel · Princeton University · UC Berkeley