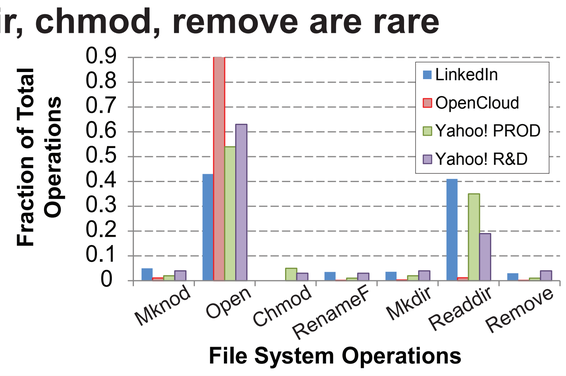
ShardFS vs. IndexFS: Replication vs. Caching Strategies for Distributed Metadata Management in Cloud Storage Systems

Lin Xiao, Kai Ren, Qing Zheng, Garth Gibson
Carnegie Mellon University

Scaling Metadata Service

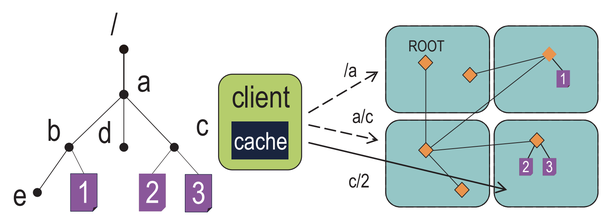
- Big Data is lots of data and lots of files too
  - › Lots of files means lots of metadata operations
- Operation distribution from HDFS clusters
  - › open is the most common operation
  - › mkdir, chmod, remove are rare



System Designs

IndexFS

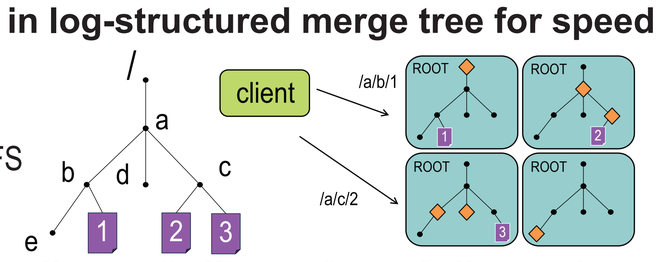
code available:
http://www.pdl.cmu.edu/indexfs



- Dynamically partitioned namespace
  - › Newly created directory is randomly assigned to a server
  - › Binary splitting a directory partition using GIGA+ [FAST11]
- Use client caching of directory entries to mitigate hotspots
  - › Don't want storms of cache invalidation callbacks
  - › Use leases with only expiration deadlines per directory
  - › Affect only rmdir, rename and chmod directory
- Represent metadata in log-structured merge tree for speed

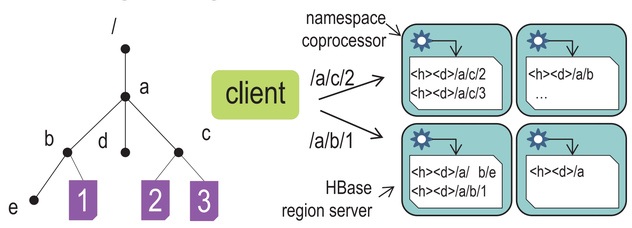
ShardFS

code available:
http://www.pdl.cmu.edu/ShardFS



- Replicate directory attributes & dirents for subdirectories
  - › Any MDS can resolve pathname locally
  - › Client only talks to one MDS for file operations
  - › Slower directory mutations, e.g. mkdir
- Shard files: by hash on pathname (or part of it)
  - › File metadata is only stored in one server
- Distributed transactions for directory metadata mutations
  - › Optimistic concurrency control
  - › Optimized monotonic mutations: reduce blocking
  - › Single RPC operations may retry when fail

Giraffa

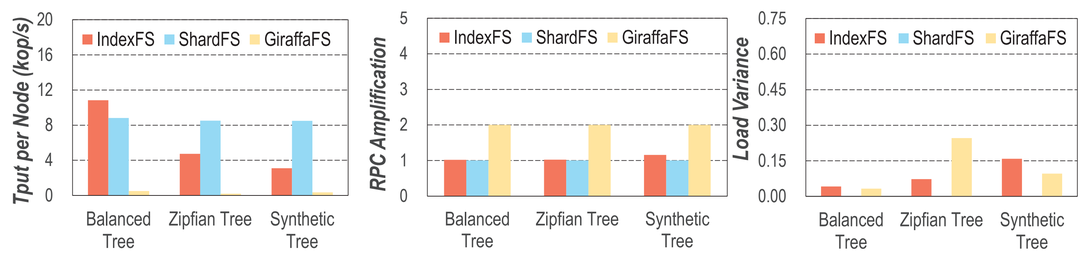


- Table partitioned namespace: metadata is stored in HBase
  - › Each file and directory is mapped to one row with a hash string and full path as the key
- Metadata operations implemented as coprocessor
  - › No hierarchical permission checks
- Related: CalvinFS stores permissions from root, dir content as values for readdir, WAN replication

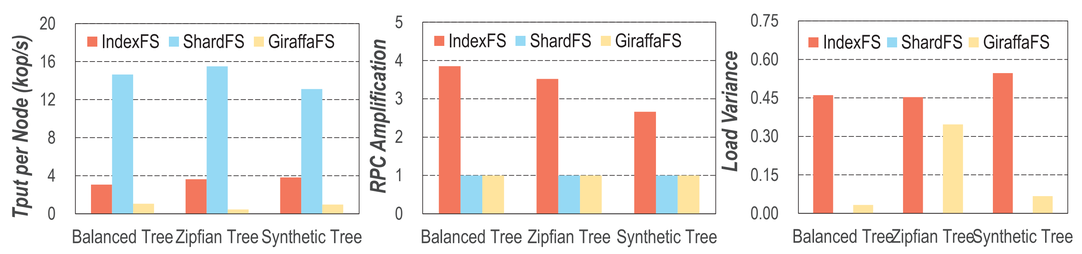
Experiments

- 64 servers nodes & 64 clients nodes on Kodiak
- Balanced: 10 subdirs/internal dir, 1280 files/leaf dir
- Zipfian: same dirs, leaf dir size follows Zipfian distr
- Synthetic: generated based on Yahoo! trace by Mimesis
- Three phase benchmark
  - › Directory creation: create all directories
  - › File creation: create all files
  - › Query: stat on files with various distribution

Microbenchmark Results



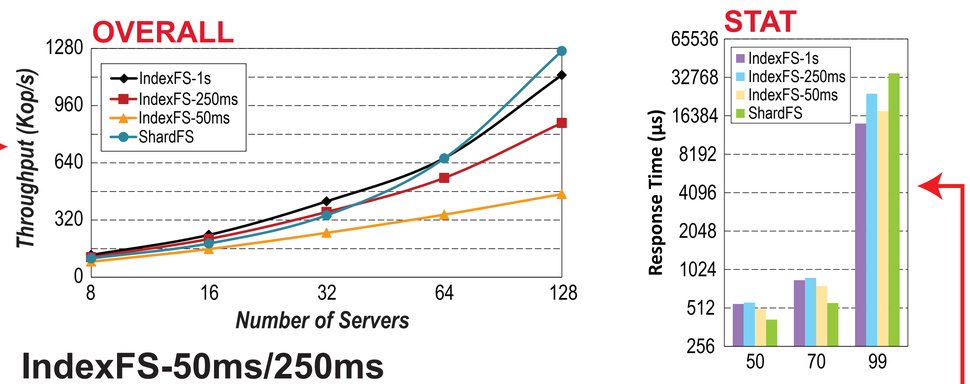
- Clients create files in leaf directories
  - › ShardFS performs similar for all trees & load balanced
  - › IndexFS hurt by dir splitting and imbalanced load
  - › Giraffa rarely splits individual directories



- Stat on files with uniform distribution
  - › ShardFS benefits from load balance and one RPC
  - › IndexFS prefix cache not effective
    - More server lookups and load imbalanced

Weak Scaling Workload

- Not all metadata operations scale as the system grows
  - › E.g. HPC checkpoint: one file per core
  - › Larger systems have more files in each directory
- Weak scaling workload
  - › File metadata ops scale while dir ops remain the same
  - › Replay LinkedIn trace with scaling file operations



- IndexFS-50ms/250ms
  - › IndexFS w/cache expire time as 50ms and 250ms
  - › Not scale when cache miss ratio is high
- ShardFS outperforms IndexFS with 128 servers
- ShardFS sees better stat latency at 70 percentile