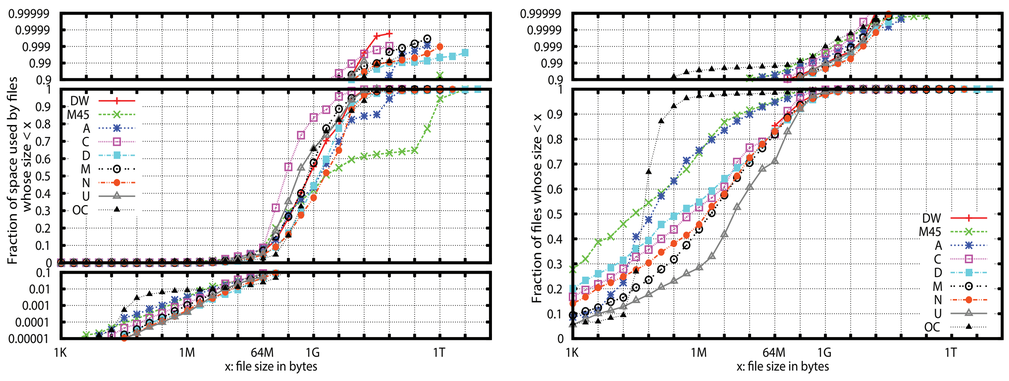
ShardFS: Scalable Metadata for HDFS

Lin Xiao, Alex Degtiar, Ge Gao, Yu Su, Chaomin Yu, Garth Gibson (CMU)

Need for Scalable Metadata

- Big Data is lots of data and lots of files too
  - Lots of files means lots of metadata operations



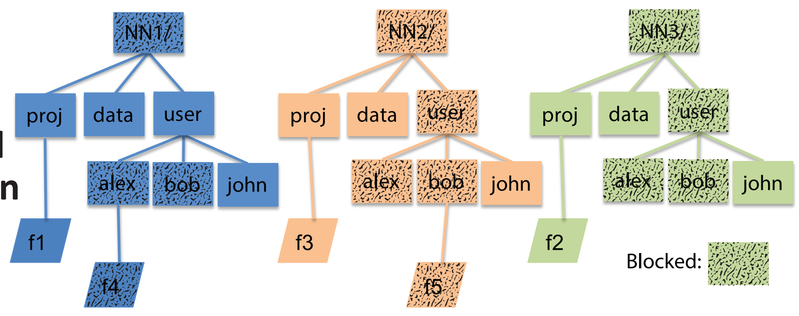
- Scaling metadata service
  - Middleware layer on HDFS federation
  - Deployable with current production
  - Lock-free fast and scalable for common operations
  - Fully replicated state for other, slow, operations
  - Best for “weak scaling” of metadata op frequency

Failure Recovery

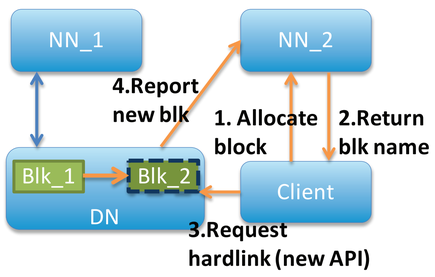
- Distributed transaction for directory inode operations
  - Relies on high availability for underlying namenodes
    - Only worry about recovery for client failures
  - Read-Write locking with write-ahead log (WAL)
  - Read-Repair: Client repairs inconsistency (after failure declared) before completing new operation
- Implemented in client library
  - Use Zookeeper as lock server and WAL store
  - Client may do recovery when encountering old WAL
  - Assumes clients have privilege for recovery operation
- Preliminary results
  - No degradation for file creates
  - mkdir latency increases from 0.5-1ms to 4-8ms
  - Need to improve efficiency of lock mechanisms
    - Add lock API in Zookeeper servers
- Should move the recovery to server for appropriate authorization

Incremental Server Growth

- Migrate metadata on ranges
  - Cur range: [/user/alex, /user/bob]
  - Next range: [/user/john, /user/john]
- Within each range
  - Replicate dirs to new servers
  - Move files based on Shard function

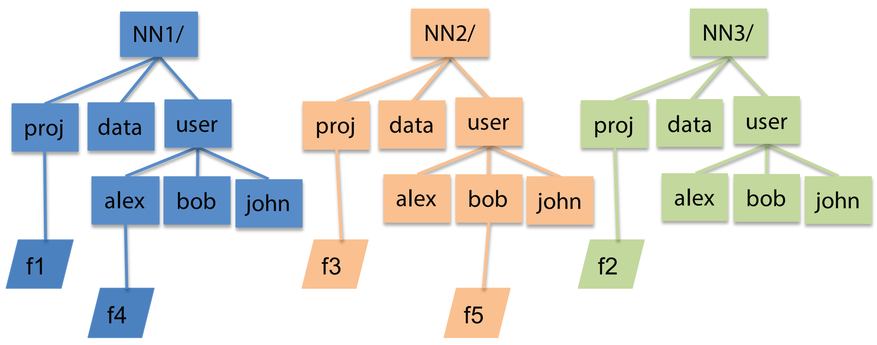


- No data movement
  - Assign new block_id
  - Use old block location
  - Hard link block file



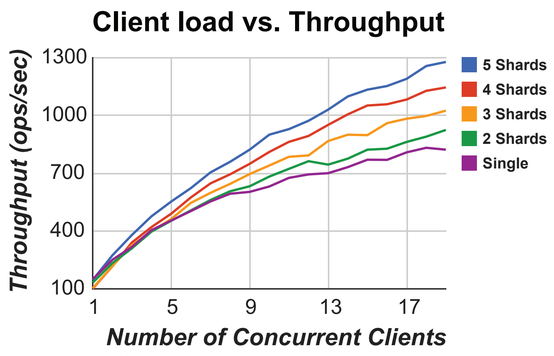
ShardFS Overview

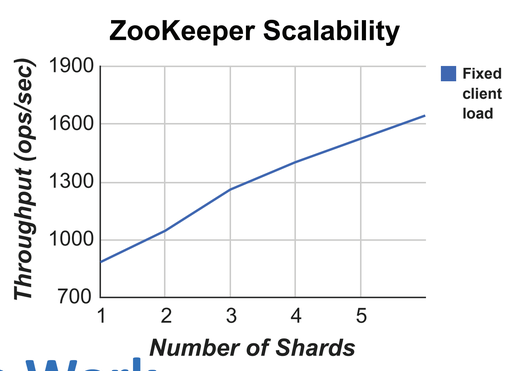
- ▪Each namenode only sees its own namespace
- Replicate namespace (directory entries for directories)
- Shard files: by hash(fullpath) or hash(filename)
- Optimistic for file inode operations
  - No locks at all: if create /proj/f1 succeeds
  - Take locks and wait if create fails
- Fine-grain zookeeper locking for directory ”inode” ops
  - Read lock from root to inode parent
  - Write lock for inode effected



Scalable Lock Service

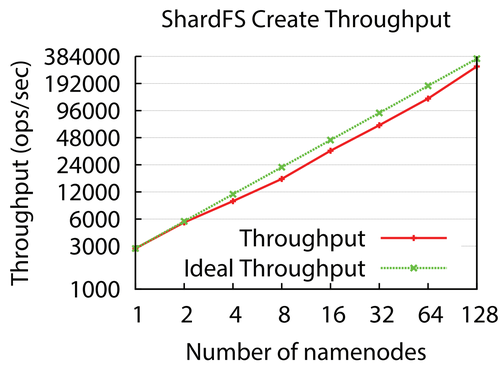
- Use DHT approach to scale ZooKeeper as lock service
- ShardFS client library for lock service:
  - Flatten directory namespace
  - Delegate lock requests to ZooKeeper shards using flattened path as shard key
- Each ZK shard relies on existing high availability of ZK
- 1-node ZK saturates at ~900 ops/sec
- Sharding scales near-linear of throughput, w/small overhead cost





Performance & Future Work

- Experiments run on Kodiak cluster
- Demonstrate significant scalability
  - Create tput almost increases linearly with more



- High latency for directory operations
  - Latency of mkdir as distributed transaction grows faster
  - Multiple ZK calls for locks hurts ShardFS latency
- Future work:
  - Workload optimization to avoid replicating small dirs

Intel Science & Technology Center for Cloud Computing

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