Scalable Metadata Service in HDFS

Lin Xiao

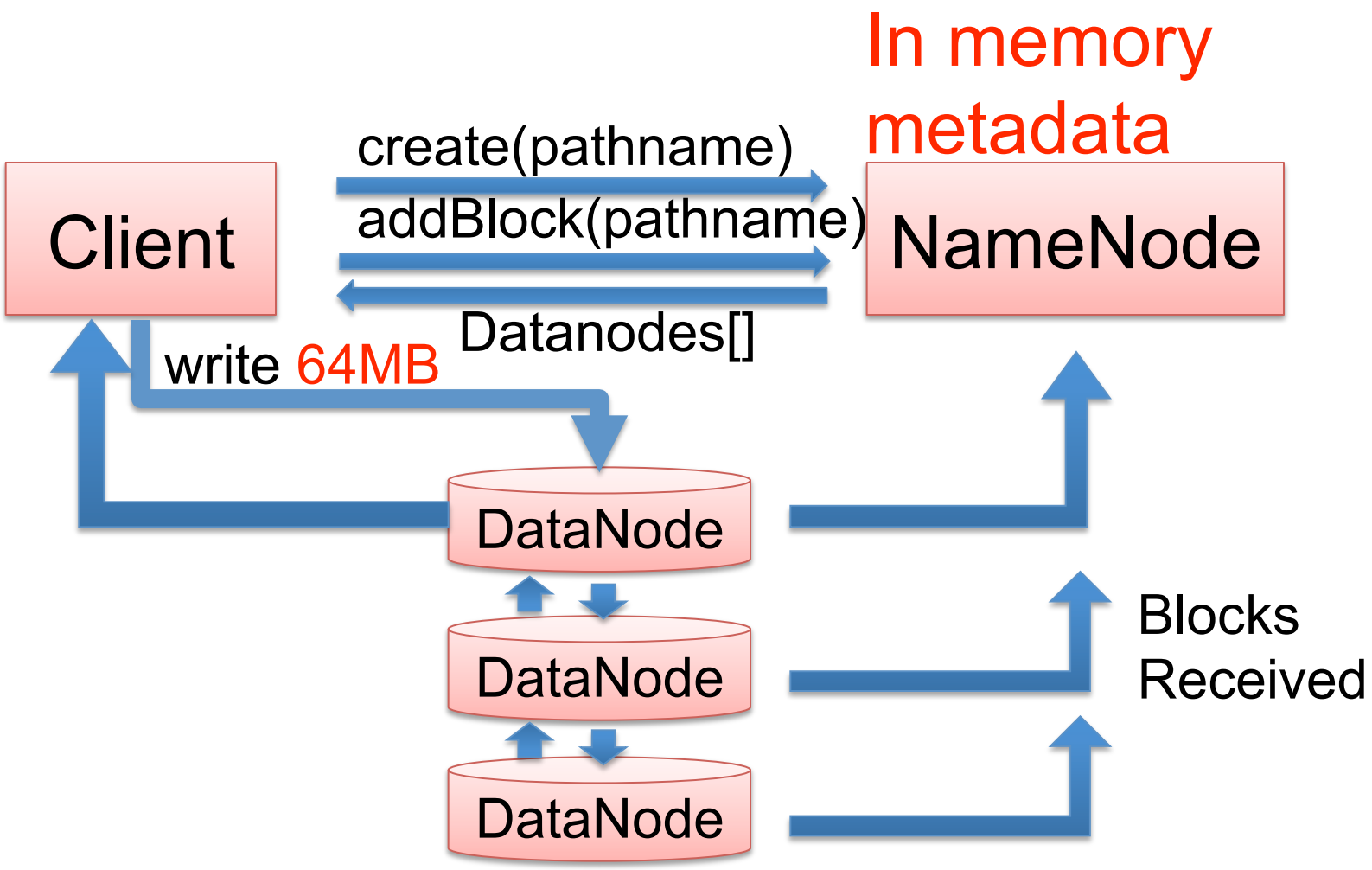
Wittawat Tantisiriroj, Garth Gibson

PARALLEL DATA LABORATORY

Carnegie Mellon University

Carnegie Mellon
Parallel Data Laboratory

HDFS architecture



Single metadata server limitation

- One node is throughput limited
  - Especially problematic for small files
- Total number of files is limited by memory size
  - In memory necessary for fast access

Our goals

- Remove the limit on number of files & blocks
  - Store metadata in memory & disk
- Distribute metadata service
  - Increase the throughput without hot spot
- Keep low latency
  - When metadata fit in memory, achieve good perf
- High availability and fault tolerance

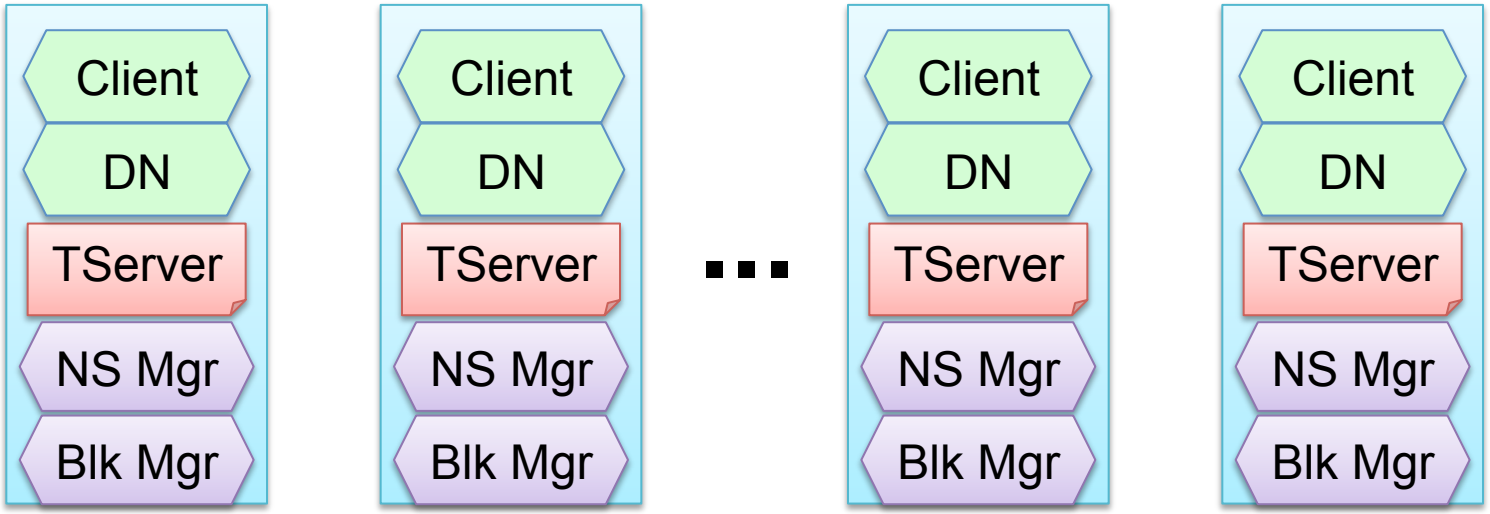
- Apply techniques to scale HDFS namenode

Related work

- Shared-disk with distributed lock: GPFS
- Static namespace partition
  - Sub-tree: Yahoo! Federation HDFS, PanFS
  - Round robin: PVFS
- Dynamic partition
  - Sub-tree: Ceph (on recent workload)
  - Directories as objects : Ursa Minor
  - Hash: Giga+
- Table for metadata: Colossus

What shall we do?

- Table stores removes HDFS limitation with parallel storage, name and block managers

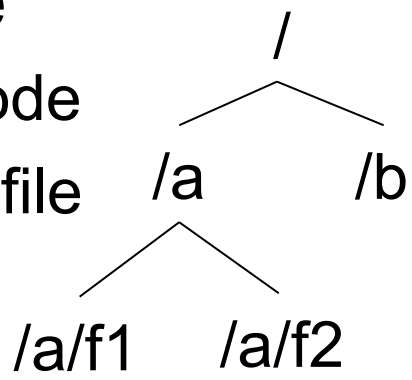


What to keep in a scalable DB?

- Namespace and file metadata table
  - Pathname to row key mappings
    - May cause multiple RPCs to open a file
  - Column families: attributes, data locations
- Block metadata and location table
  - Various granularity of managing blocks
- Other tables
  - Datanode status
  - Quota
  - Both are constantly changing

Row key choices (1)

- Parent Inode + file name
  - + Rename only changes inode
  - - Multiple lookups for each file
    - Multiple RPCs

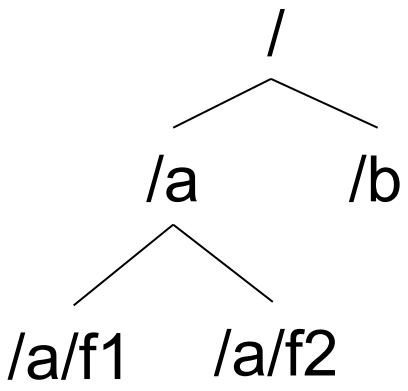


Path	Key
/	0
/a	0+a
/b	0+b
/a/f1	1+f1
/a/f2	1+f2

- Hash (path name)
  - + Load balanced
  - - No Locality

Row key choices (2)

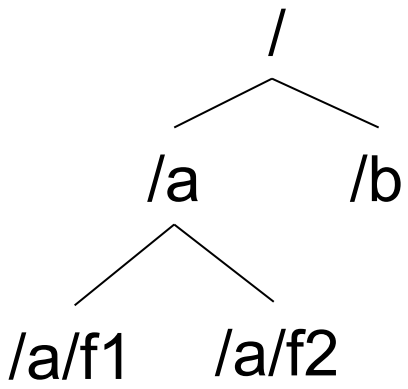
- Full pathname
  - + One table lookup for each file
  - - Cannot just scan for just children
  - - Rename a directory needs to rename every file



Pathanme	Key
/	/
/a	/a
/a/f1	/a/f1
/a/f2	/a/f2
/b	/b

Row key choices (3)

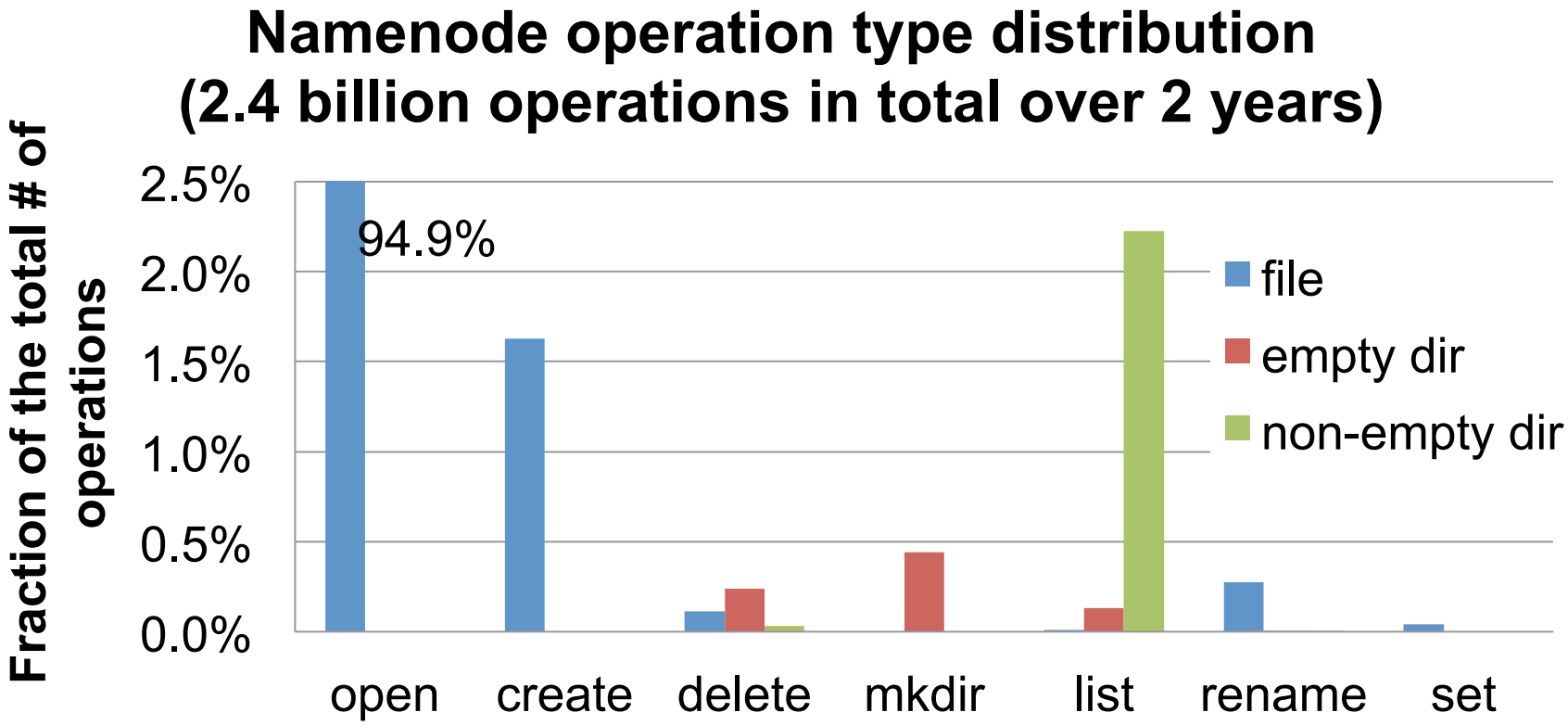
- Directory depth + full path name
  - \+ Locality for every directory



Pathanme	Key
/	0+/
/a	1+/a
/b	1+/b
/a/f1	2+/a/f1
/a/f2	2+/a/f2

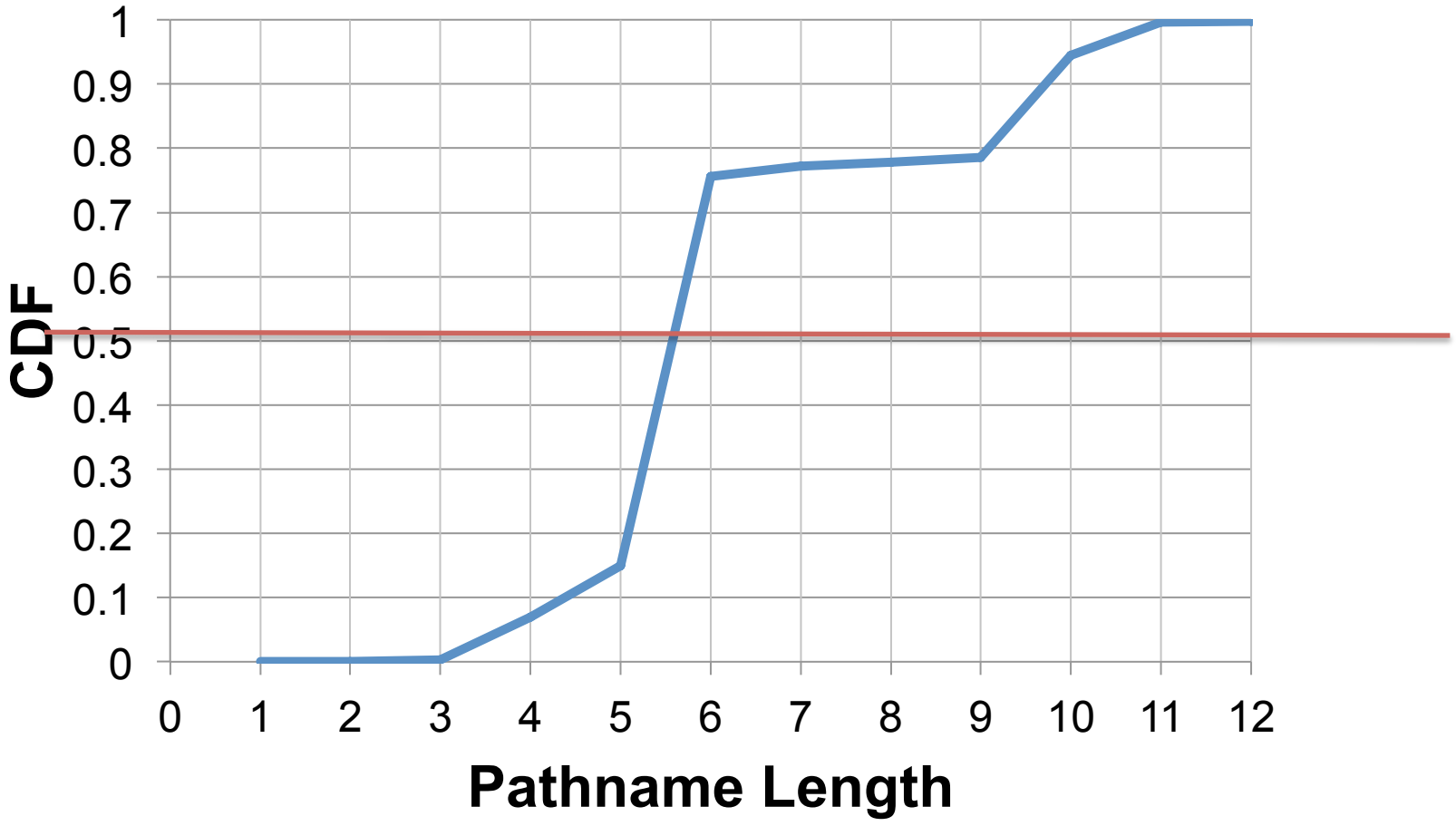
- Which one is better?

Opencloud operation stats



- Most operations are open

Pathnames tend to be deep



Challenges

- Bootstrapping
  - Original HDFS for the scalable DB?
- Reduce latency caused by distributing metadata
  - Collocate processes with tablet server
  - Send requests in parallel
- Efficient use of memory
  - Memory overhead compared to customized service
- Constantly changing table
  - Datanode status, Quota
- Failure handling