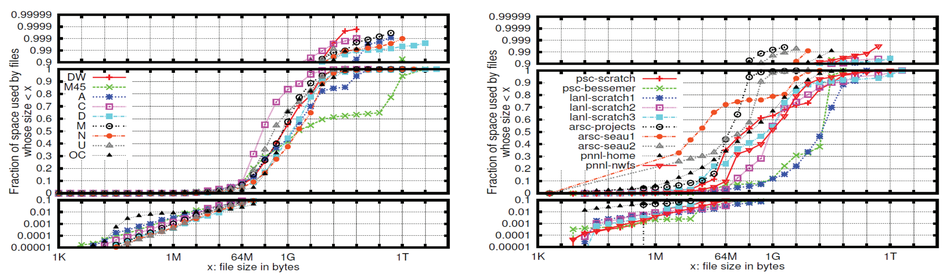
FILESYSTEM METADATA MANAGEMENT USING FAST KEY-VALUE STORE

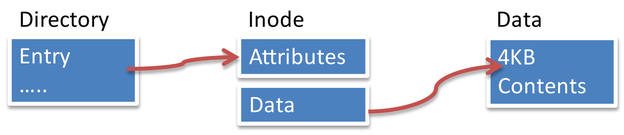
Kai Ren, Garth Gibson (CMU)

OVERVIEW

- Today's local filesystems are filled with large files but overwhelmed by more small files
- Small files widely-spread on disk needing more disk seeks to get attributes and data
- Key idea: develop new directory representation to embed file attributes and small file data
- Approach: use LevelDB with one table for all directories



Old:

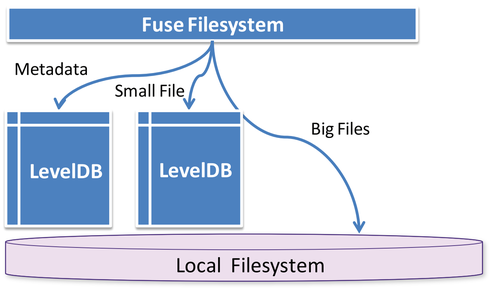


New: Directory table as a key-value store

Path Name	Attributes	Data
foo	rwxr-wr--	....
bar	rwxr-wr-x	....

FILESYSTEM DIRECTORY IMPLEMENTATION

- Built on top of Fuse file system, storing metadata and small files into LevelDB

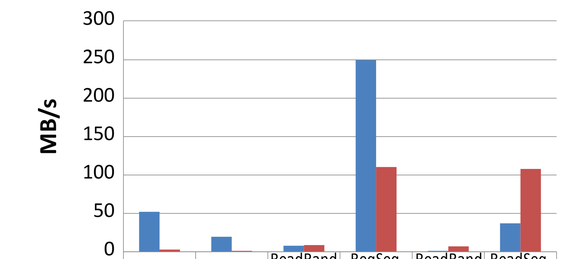


DATA LAYOUT

- Schema of Metadata:
  - Key: [Inode ID of Parent Directory, CRC Hash(filename)]
  - Value: [Filename, Attributes]
- Schema of Data:
  - Key: Inode ID, Chunk ID
  - Value: Chunk content

EVALUATION OF LEVELDB VS. B-TREES

- 1 Million entries (key: 16bytes, value: 128bytes



	FillSeq	FillRandom	ReadRand (warm)	ReqSeq (warm)	ReadRand (cold)	ReadSeq (cold)
LevelDB	52.1	19.4	7.9	248.9	0.7	36.7
Kyoto Cabinet	2.9	0.9	8.5	110.3	7.3	107.5

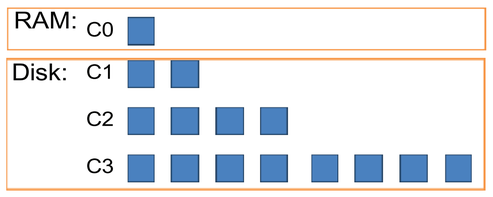
LOG-STRUCTURED MERGE TREE

- In today's Big-Data world, high ingestion workloads are prevalent and overwhelm traditional B-Trees
- Write optimized data structures like LSM (Log Structured Merge) Tree and its variants are proposed:

	Read	Update	Range Query
CoW B-Tree	$O(Log_B N)$	$O(Log_B N)$	O(LogN+L/B) random I/Os
LSM Tree	O(LogN)	O(LogN/B)	O(LogN+L/B) Sequential I/Os

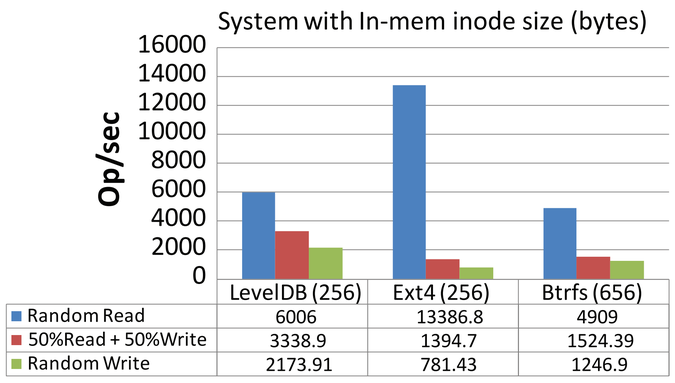
LSM DATA STRUCTURE OVERVIEW

- LSM, R-COLA, LevelDB and other variants:
  - C[i] (i > 0): an immutable stable containing sorted (k, v) pairs, and a b-tree index
  - Size of C[i+1] = R * size of C[i]
- Insertion: write to in-ram tablet, later will be written to disk
- Compaction: merging C[0]…C[i] into C[i+1] when C[0]…C[i] are full
- Range query: search sstables with appropriate range in each C[i]



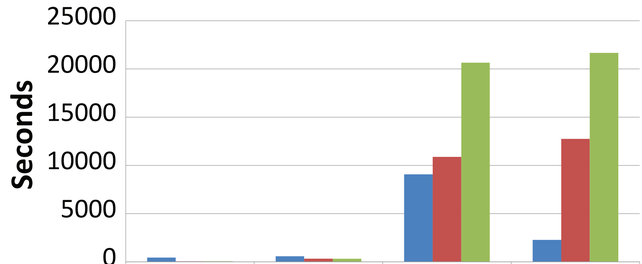
EVALUATION OF FILESYSTEM METADATA

- Workload: 1M empty files, only attributes in LevelDB
- Random Read: stat a file randomly
- Random Write: chmod / utime a file randomly
- Memory limit: 600MB for LevelDB, 1000MB for Ext4 and Btrfs
  - Limiting memory tries to control different internal cache sizes.



	LevelDB (256)	Ext4 (256)	Btrfs (656)
Random Read	6006	13386.8	4909
50%Read + 50%Write	3338.9	1394.7	1524.39
Random Write	2173.91	781.43	1246.9

- Workload: Write 1M 1KB files sequential, overwrite 1M 1KB files. Compare sequential overwrite, random overwrite, and overwrite with read operations
- Memory Limit: 512MB, Disk Partition: 5GB



	Sequential Write	Sequential Overwrite	50%Read+50% Write	Random Overwrite
LevelDB	444	581	9071	2279
Ext4	52	324	10886	12754.5
Btrfs	59	318	20655	21669

Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University · Georgia Tech · intel · PRINCETON UNIVERSITY · UC Berkeley