PathFS: A Streaming File System for Edge Environments

Kianoosh Abbasi

Motivation

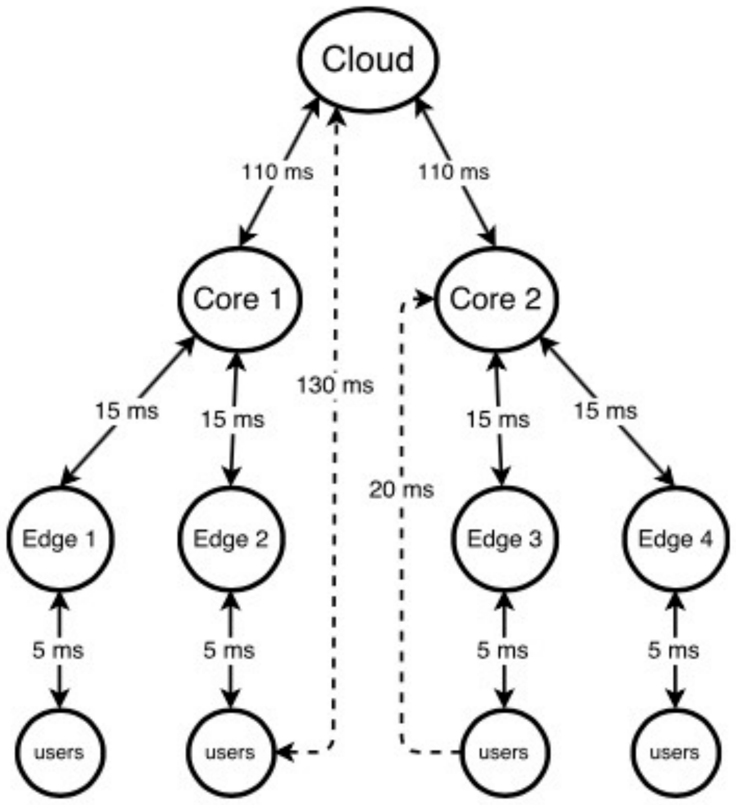
- Most of existing file systems only address cloud systems
  - NFS, IPFS, Ceph
- Edge scenario
  - IoT cameras and sensors continuously producing data
  - Operators want to store and observe the stream of data with low latency

Motivation

- Solution:
  - A file system that addresses the heterogeneity of resources
  - Provides fast data propagation to support streaming
- Building on top of existing systems
  - CloudPath paper: an edge computing framework with routing, execution, and storage facilities
  - PathStore: CloudPath’s storage system
    - Based on Cassandra
    - Data represented by CQL queries

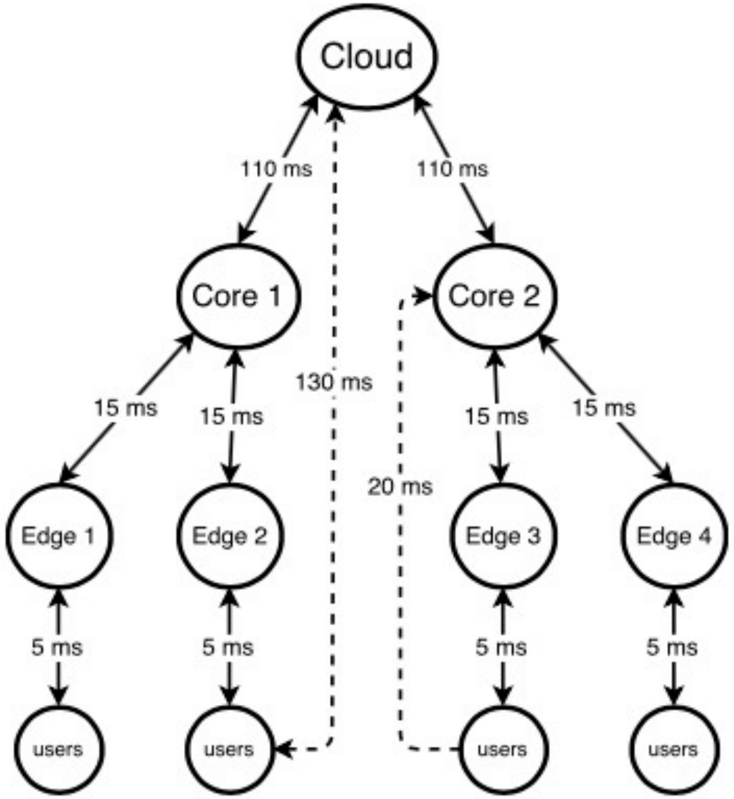
PathStore Architecture

- Hierarchy of nodes
- Each node store a subset of parent's data
- Moving down the hierarchy
  - Resources are more limited
  - Delay and bandwidth usage decrease
- Data replication policy
  - Lazy propagation of updates
  - Push and Pull servers
    - Interest subscription



Event-driven System

- Updates will be propagated eagerly
  - Upward propagation
  - Downward propagation
- Downward propagation based on interest subscription in PathStore
  - Use a private table to check query matching



File System Interface

- Entry vs Filedata tables
- Implemented FUSE interface
  - File System for User Space
  - FUSE Syscalls
    - Read
    - Write
    - Readdir
    - Open
    - Create
    - Symlink

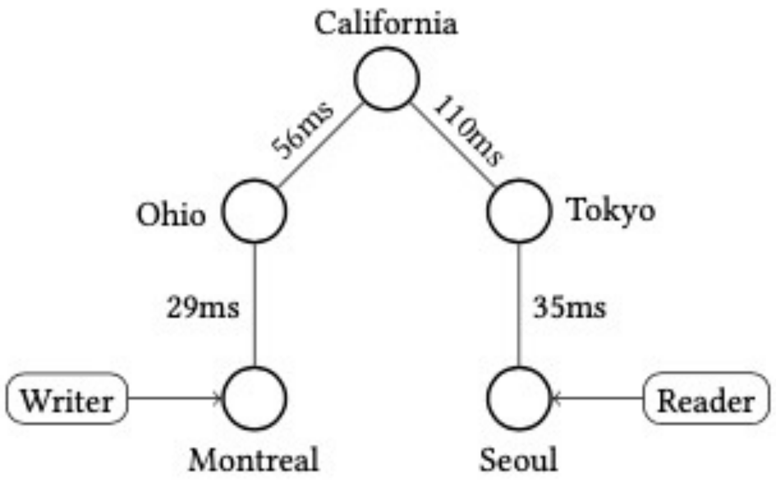
Entry (parent_id, name):
    parent_id: uuid
    name: varchar
    id: uuid
    mode: int (type & permissions)
    userId: int
    groupId: int
    size: bigint
    created: timestamp
    modified: timestamp
    changed: timestamp
    blocks: int
    follow: varchar

Filedata (file_id, insert_time)
    file_id: uuid
    insert_time: timestamp
    raw_data: blob

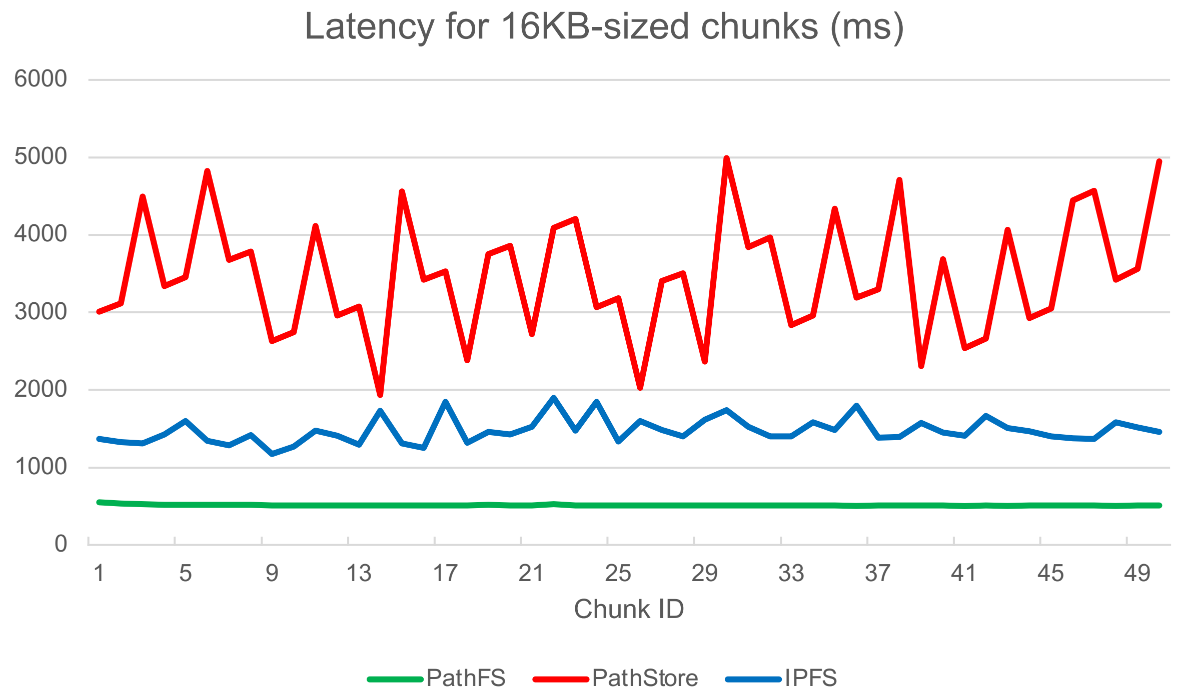
EVALUATION

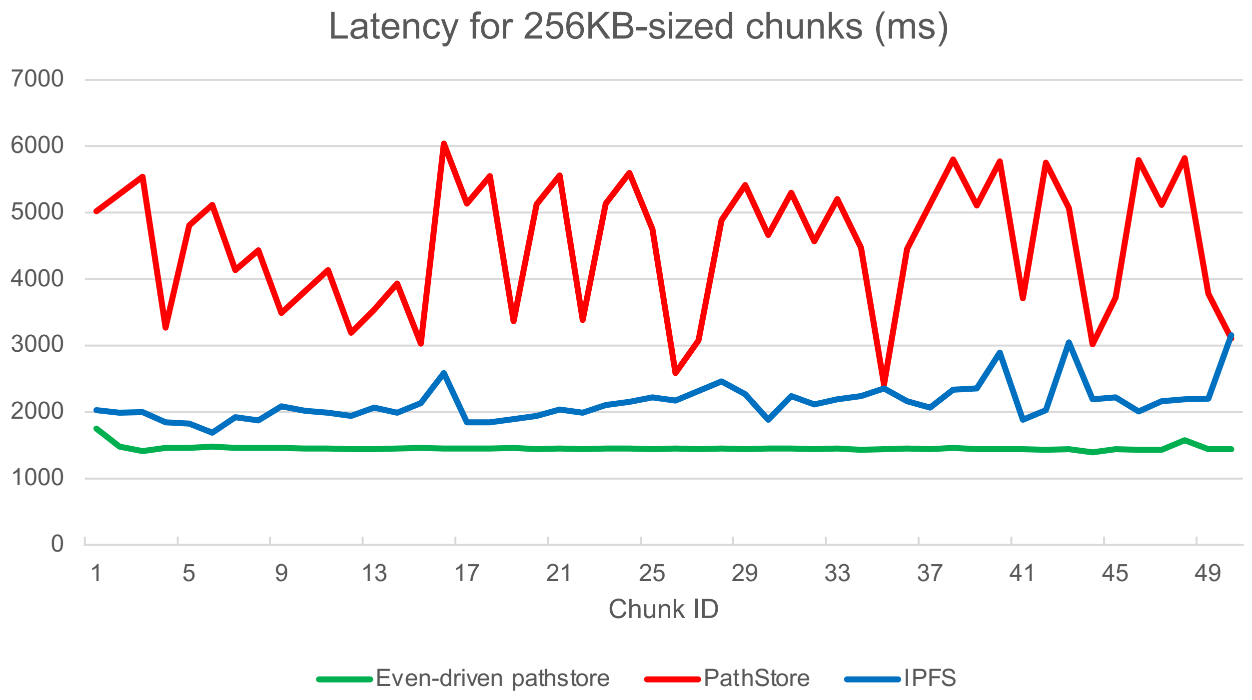
Evaluation

- Evaluating the event-driven system
- 5-node topology with simulated latency
- Writer repeatedly inserts 16KB and 256KB-sized data chunks
- Measured how long it takes to reach reader
- Compared with PathStore and IPFS



Event-driven system results





FUSE Evaluation

- Meaursing single-node performance of the schema and FUSE implementation
- Used FIO benchmark
  - Repeatedly reading/writing data with 4K-sized chunks
- Compared with the native file system, nfs, and Cassandra

FUSE Performance

FIO Benchmark (random read/ writes)	PathFS	Cassandra pathfs	native	nfs (1 host)
read	629 KB/s	1.2 MB/s	1.8 MB/s	621 KB/s
write	230 KB/s	444 KB/s	666 KB/s	227 KB/s

Summary

- PathFS: File system for edge environments
  - Heterogeneity of resources
  - Support for streaming
- Built on top of PathStore: A key-value edge storage using Cassandra
- Contributions:
  - Event-driven update propagation
  - FUSE interface support
- 60% Improvement in propagation latency

Future Work

- More rigorous testing of FUSE
  - Finding out performance bottlenecks
- PathFS writes are append-only
  - Implement support for edits
- Conflict handling

PathFS Bottlenecks

Template from: https://www.free-powerpoint-templates-design.com/mobile-control-drone-powerpoint-templates/

System Overview

Experiments

- Measure each layer’s overhead
- Experiments with:
  - Cassandra simple insert/update queries
  - Cassandra + our schema (fuseless)
  - Cassandra + our schema + fuse
  - Cassandra + our schema + fuse + FIO (the benchmark)

New Swift Nodes!

- Infrastructural changes:
  - Moved the swift nodes to SSD drives
  - Cool and fast nodes!
  - Unit tests for PathFS

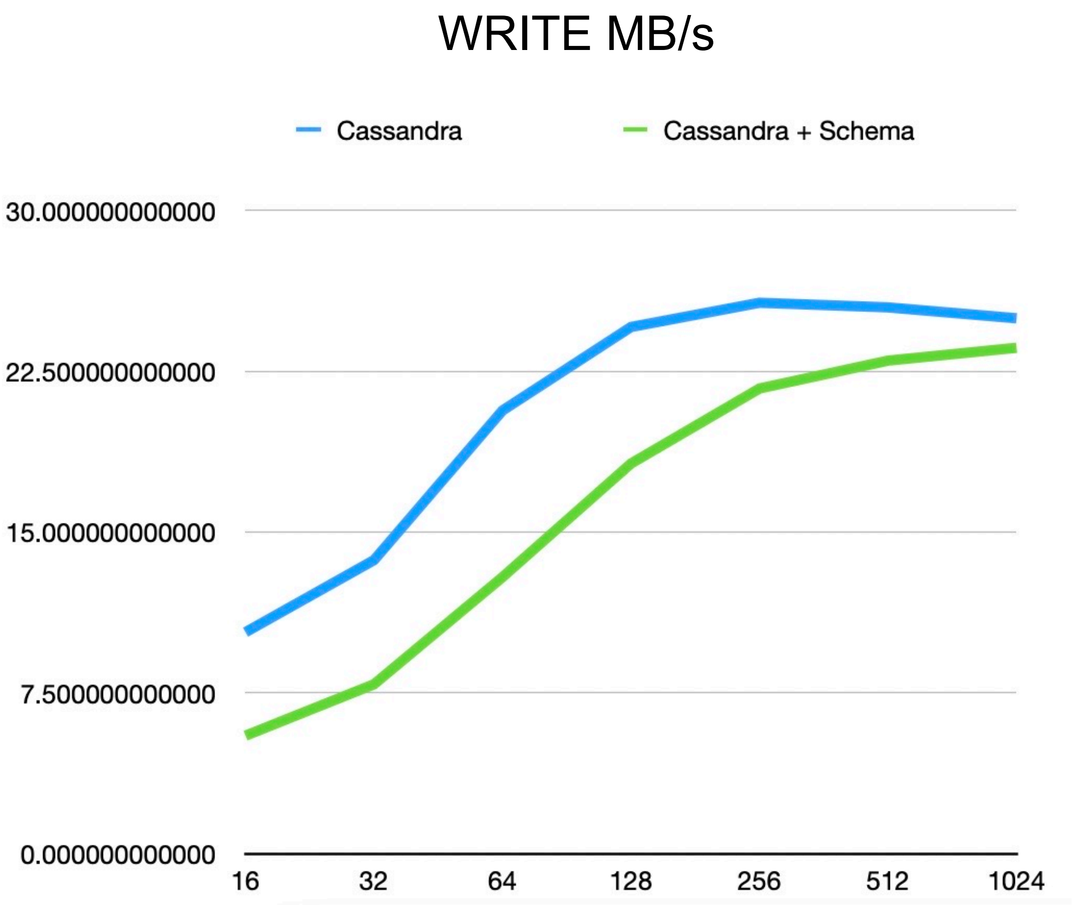
Experiments

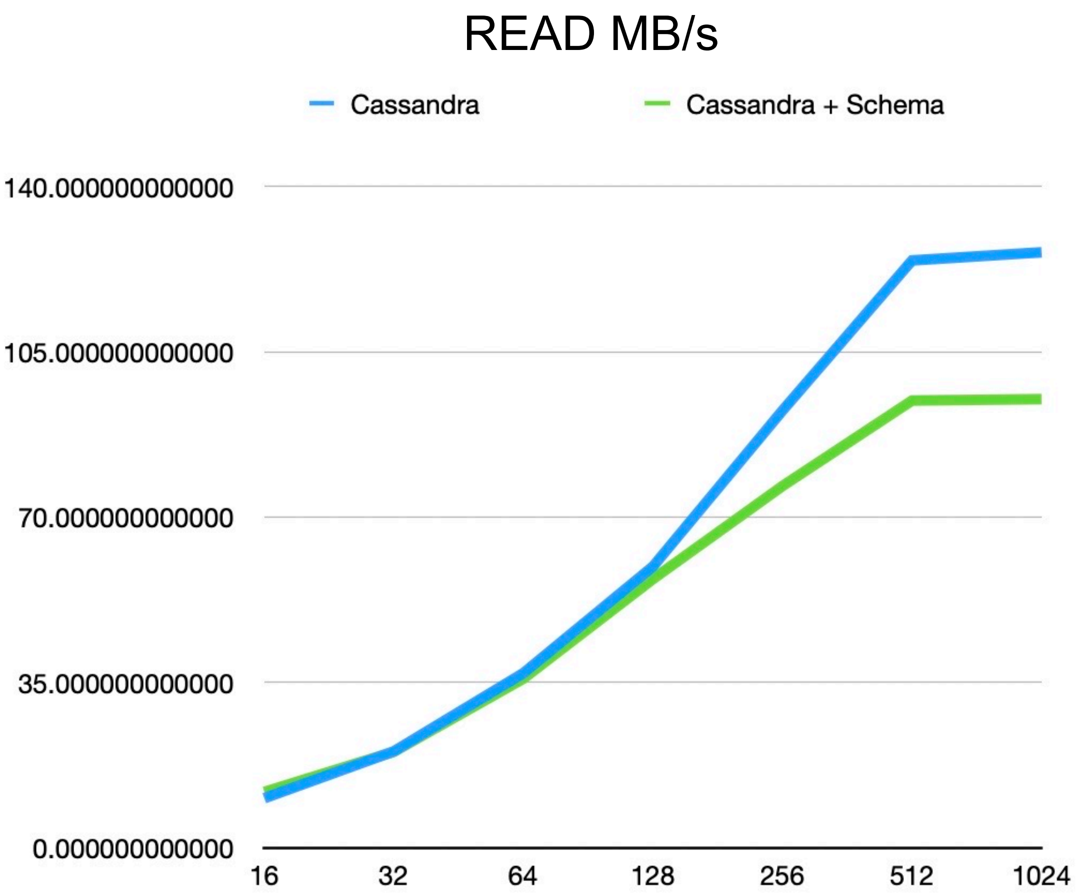
- PathFS now assumes fixed chunk sizes
- In all experiments:
  - We create a 128MB-sized file
  - Add chunks one by one
  - Close and and re-open
  - Read chunks one by one
- Chunk sizes vary!

Wow!

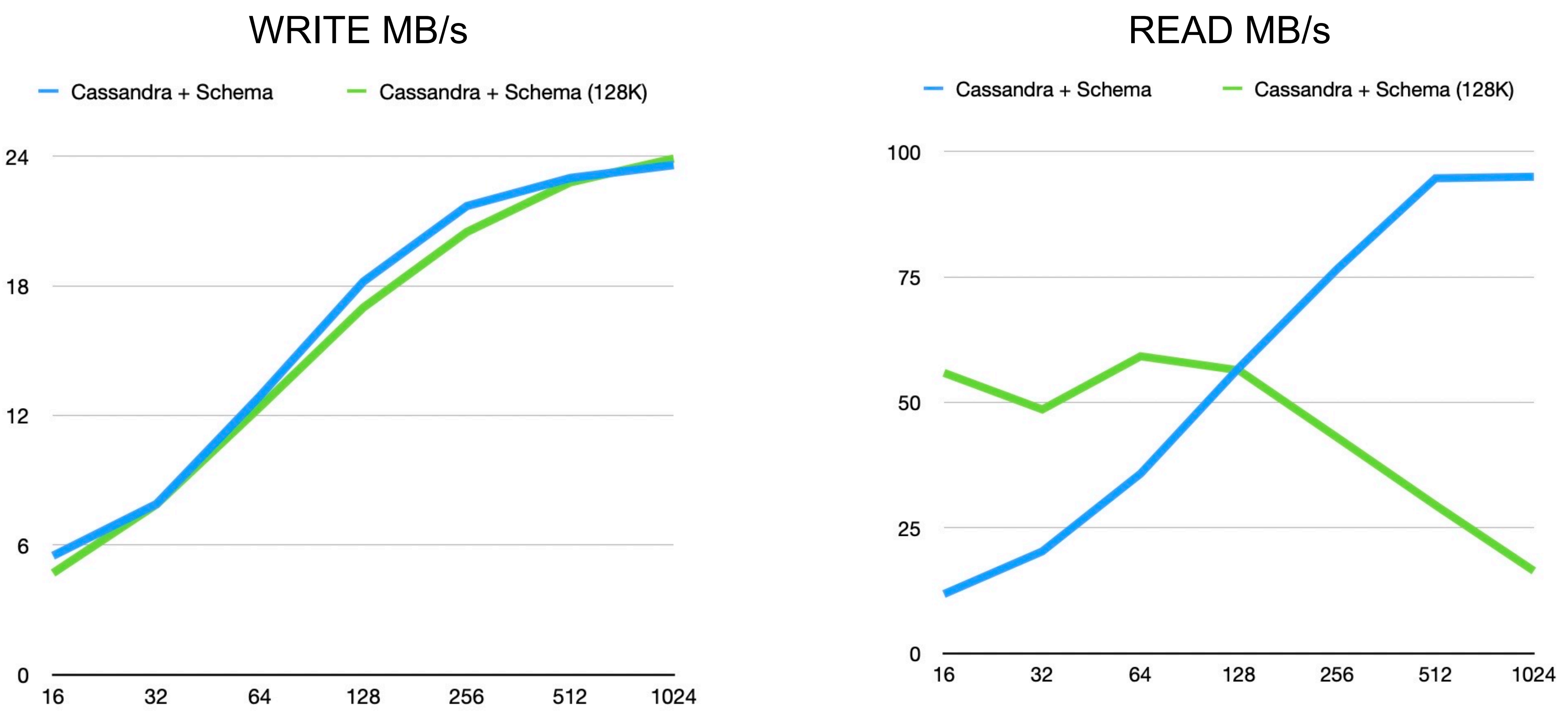
- Always seen performance degradation in fuse for large chunks
- The reason: Fuse modifies the requested chunk size!
  - 0 - 4K -> 4K
  - 4 - 16K -> 16K
  - 16 - 64K -> 64K
  - 64 - infinity -> 128
- Also, Fuse is multi-threaded

Schema Overhead



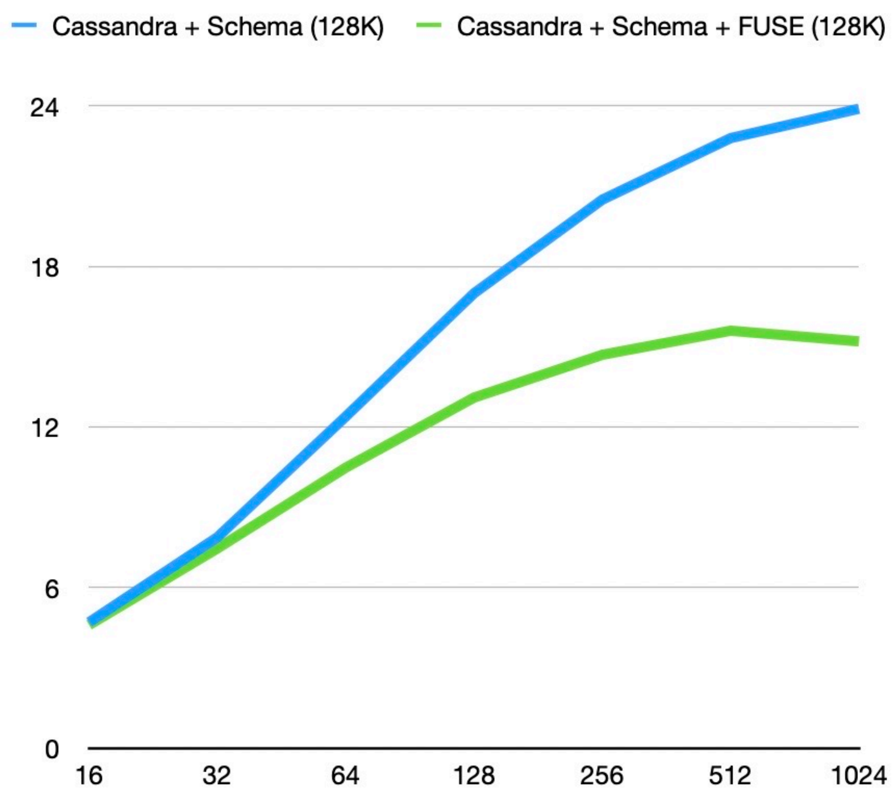


The Impact of Fixed Sizes

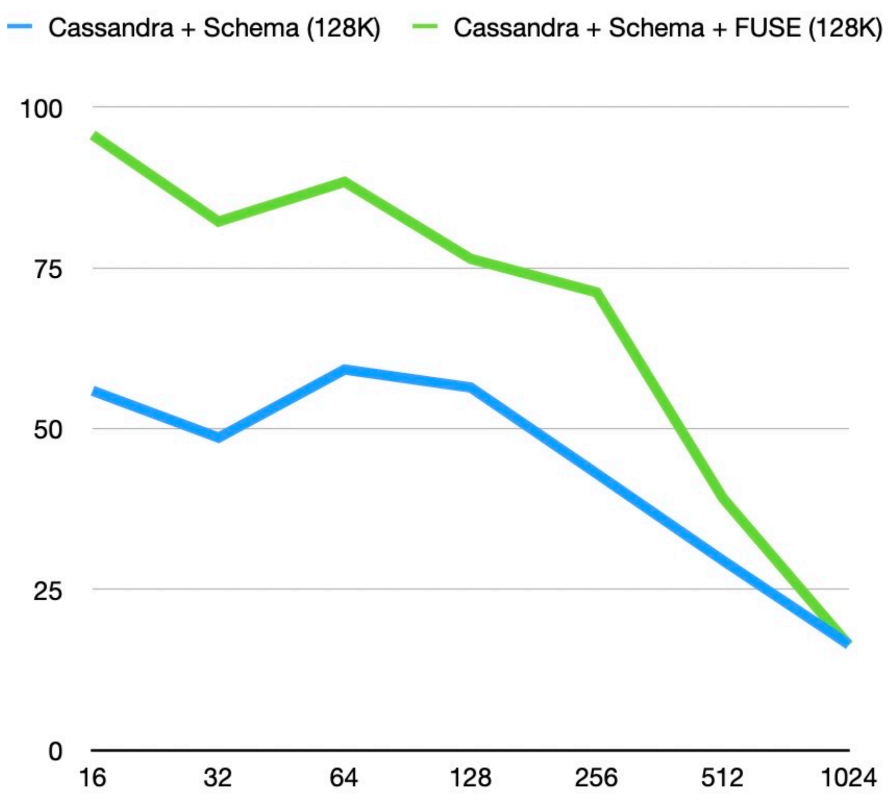


FUSE Overhead!

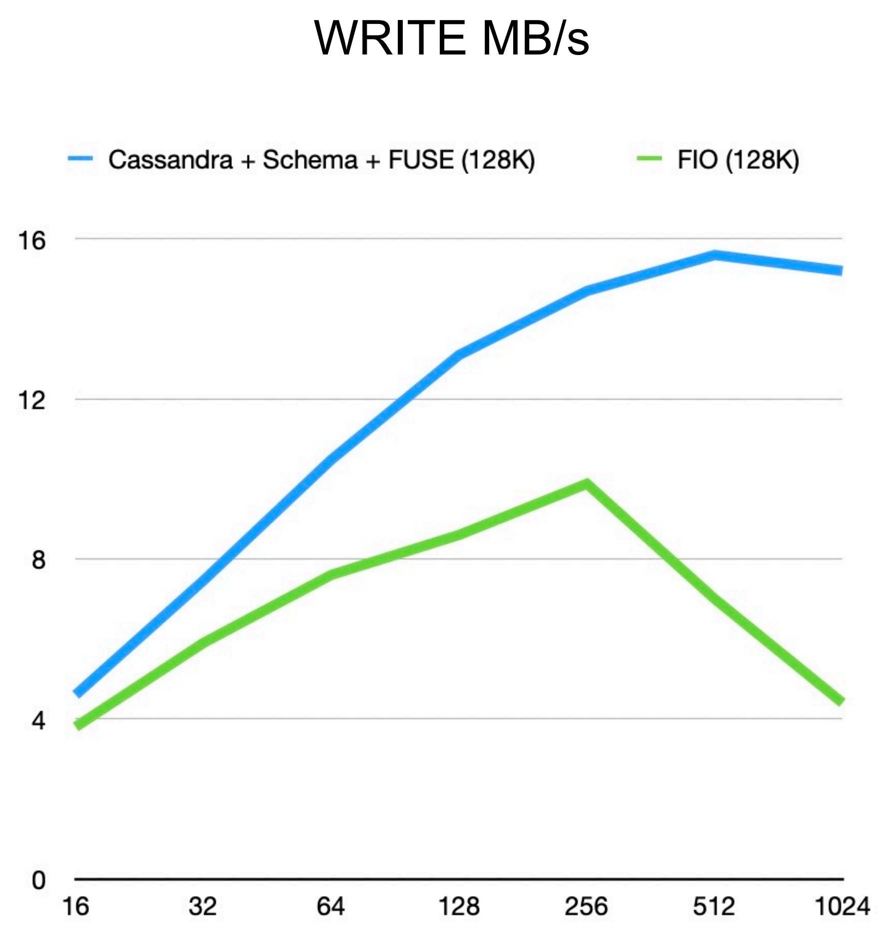
WRITE MB/s

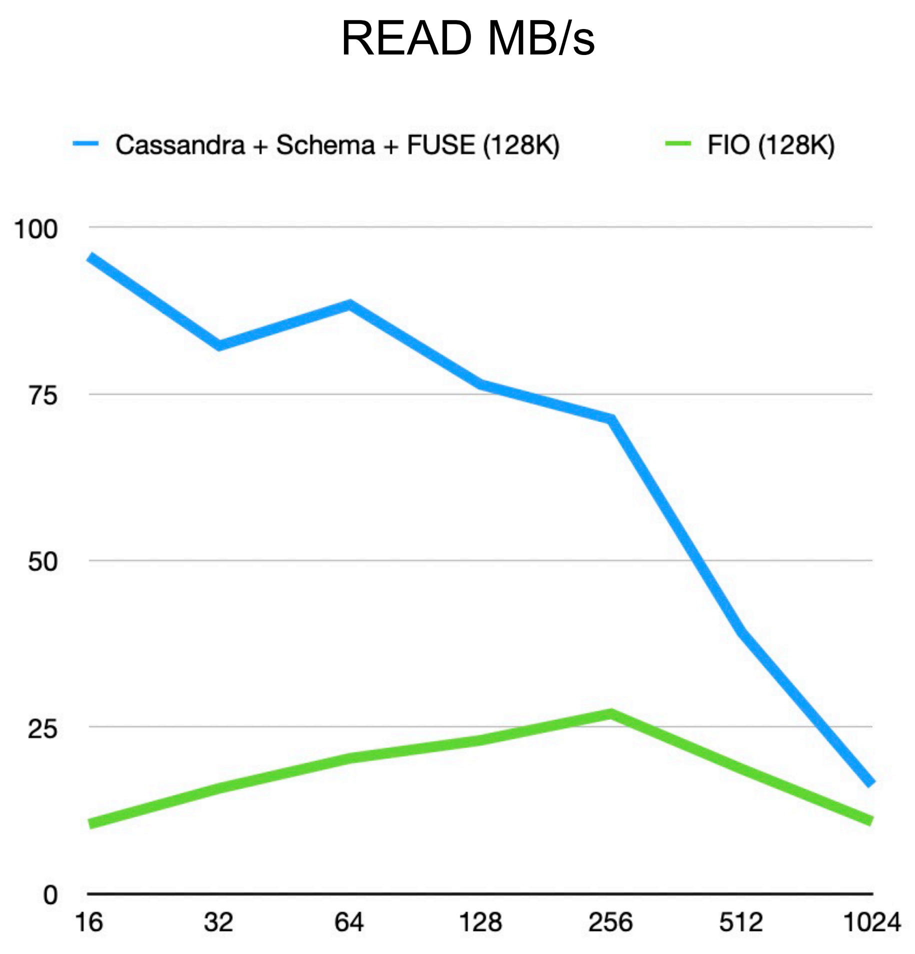


READ MB/s



FIO Overhead





Thanks for your attention!

Template from: https://www.free-powerpoint-templates-design.com/mobile-control-drone-powerpoint-templates/