Supporting Transactions for Bulk NFSv4 Compounds

13th ACM International Systems and Storage Conference (SYSTOR 2020)

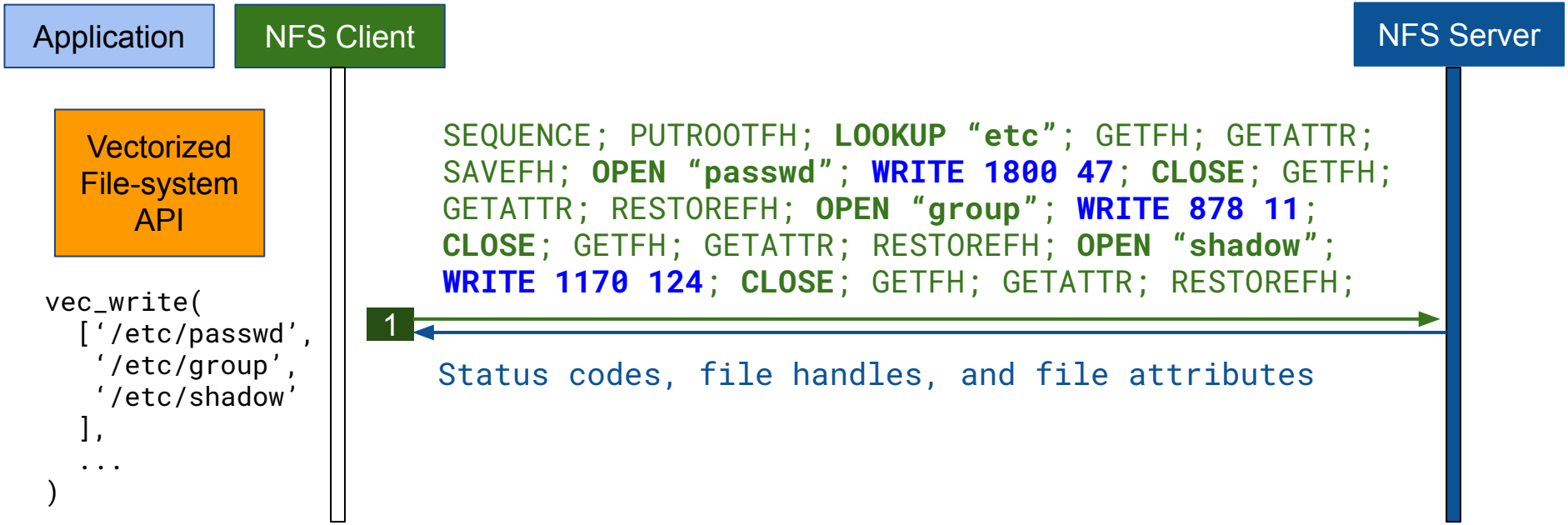
Wei Su[1], Akshay Aurora[1], Ming Chen[2], Erez Zadok[1]

[1]Stony Brook University; [2]Google

Stony Brook University

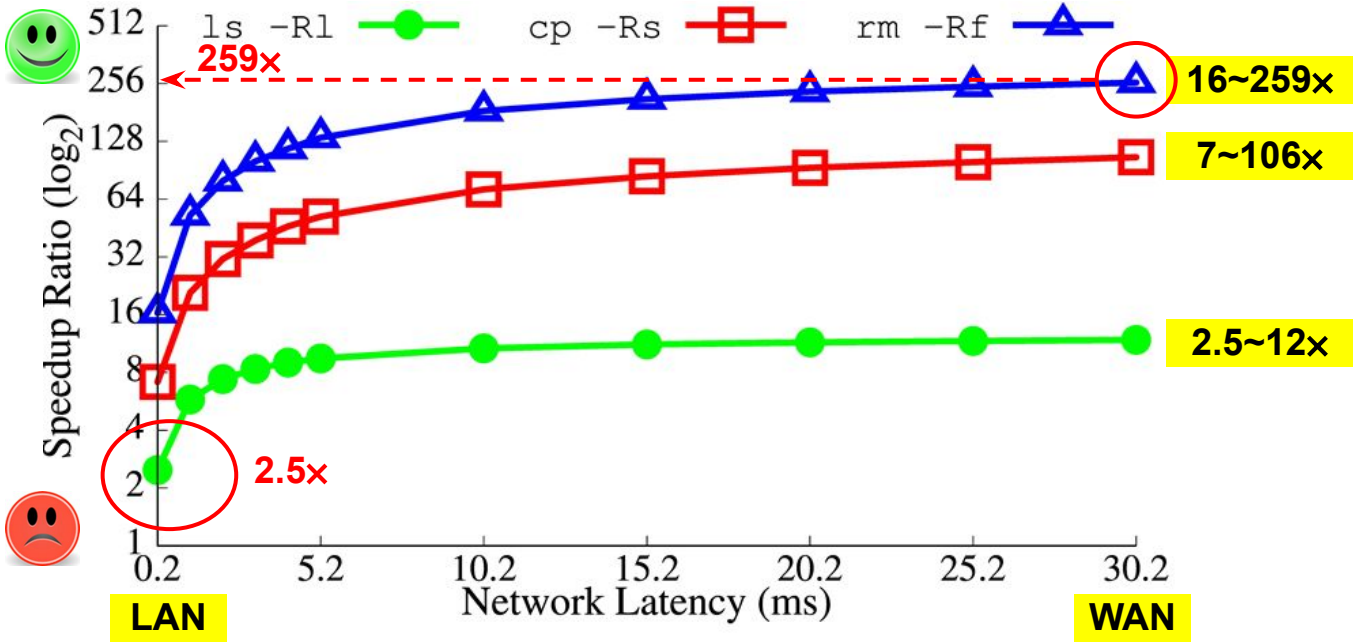
Background: Vectorized NFS

- Ideal utilization of compounding: Writing multiple files in one compound request



Background: Vectorized NFS

- Performance evaluation: Metadata intensive workload
  - Recursive listing, symlink, and removal



Motivation

- NFSv4 introduces “Compound” procedure
  - Clients can pack multiple NFS operations in one “compound”
  - This amortizes network latency and improves I/O throughput
  - Compounding speeds up NFS I/O by up to 2 orders of magnitude
- Challenge to client’s error handling
  - If an operation in a compound fails
    - NFS server only reports the error, but does not rollback
  - If the server crashed when executing a compound
    - Nothing will be done when it restarts
  - Difficulty for applications to handle errors
    - Any operation may fail, and crash may occur anytime
    - Hard to restore to initial state for a failed large compound

Design Overview

① Compound request reaches TCNFS server

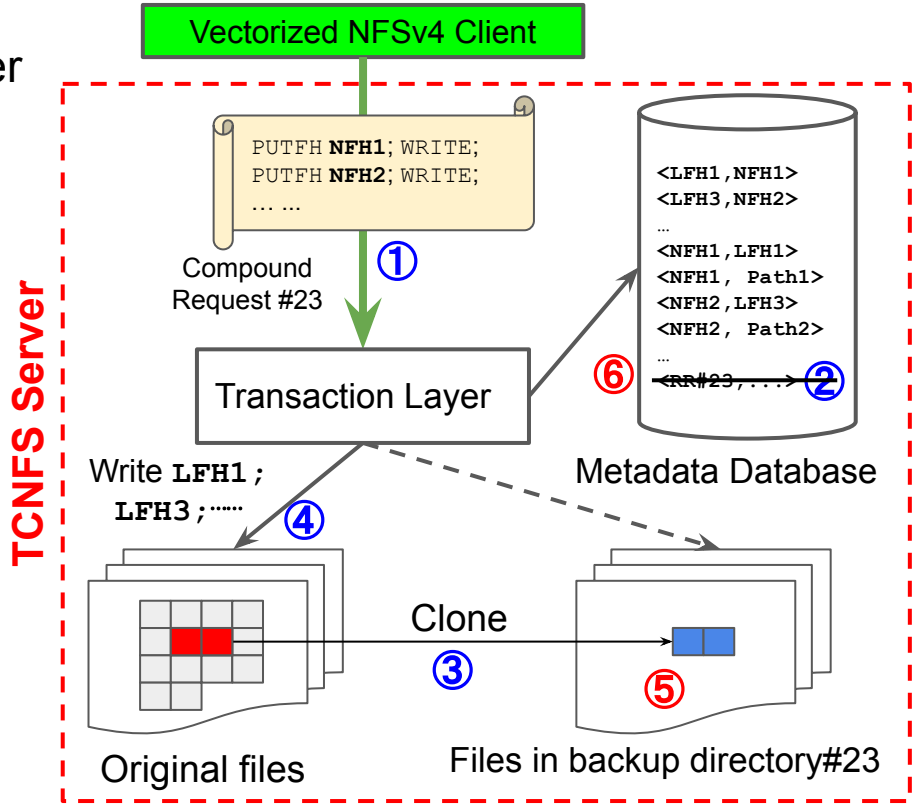
② TCNFS writes the compound request into metadata database as a Recovery Record (RR)

③ TCNFS backs up data blocks of files that will be changed by the compound request

④ TCNFS executes the operations

⑤ TCNFS removes backup data

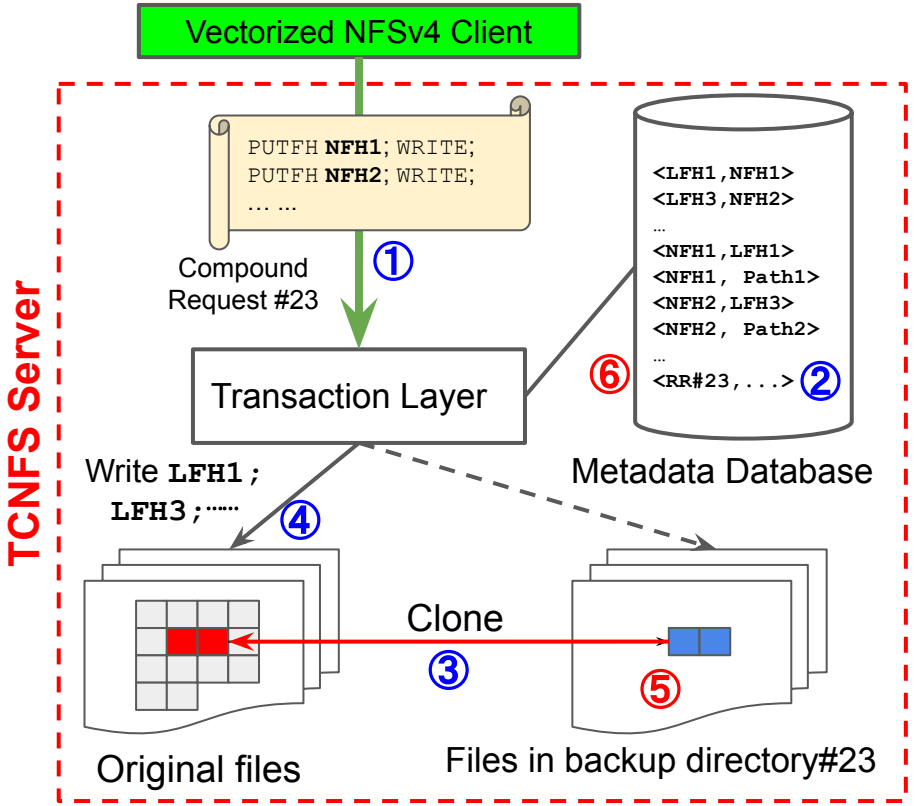
⑥ TCNFS removes the recovery record



Stony Brook University

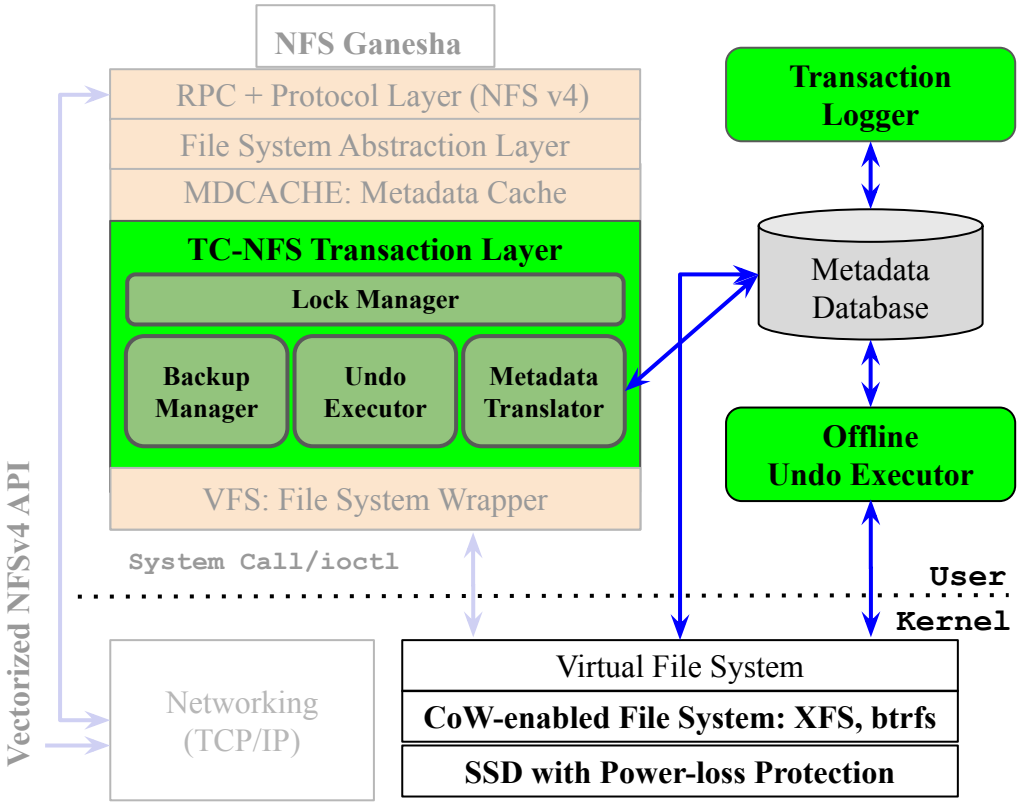
Design: Error Handling

- In case of an error…
  - TCNFS reverses previously executed operations
- In case of a server crash…
  - The recovery record will be present in the metadata database
  - TCNFS will parse the recovery record to retrieve the failed compound request
  - TCNFS reverses the partially done compound request



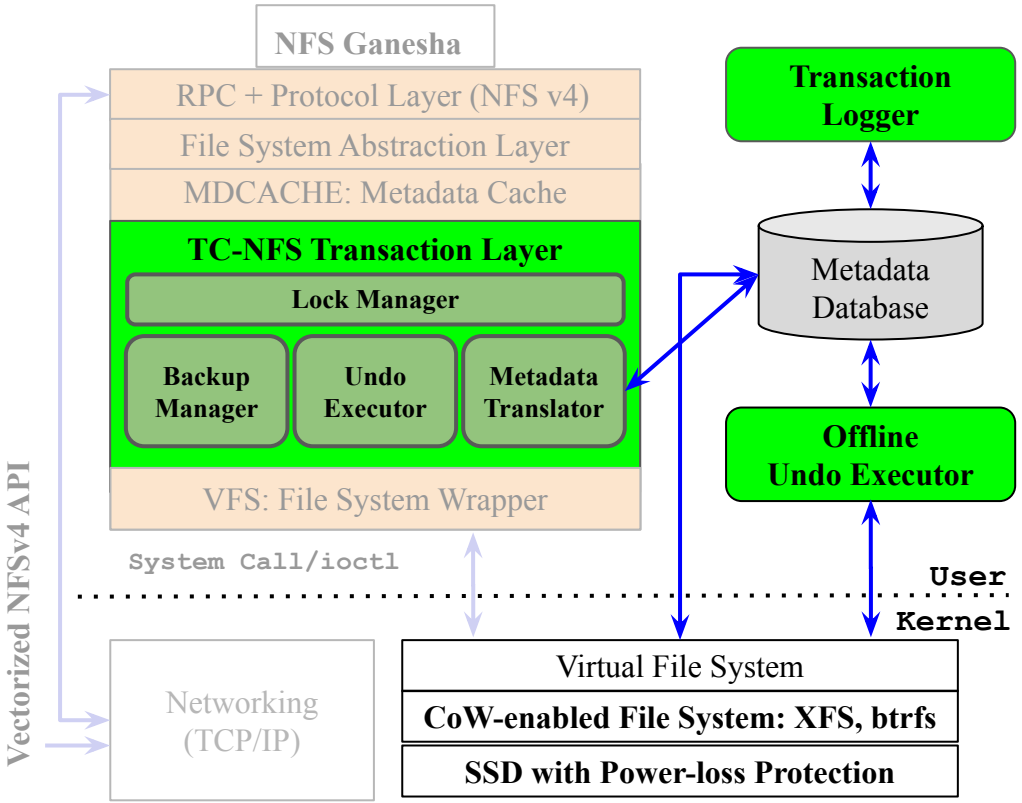
Prototype Architecture (1)

- Lock Manager
  - Coordinates multi-client conflicting access
- Backup Manager
  - Creates and cleans up backups
- Undo Executor
  - Reverts partially executed compounds due to failure
- Metadata Translator
  - Mappings between NFS file handle and local file handle



Prototype Architecture (2)

- Transaction Logger
  - Creates and cleans up the Recovery Records
- Offline Undo Executor
  - Reverts partially executed compounds due to server crash
- CoW-enabled File System
  - Use CoW to create backups to reduce I/O overhead
- SSD with Power Protection
  - Ensures endurance and reduces the latency of fsync()



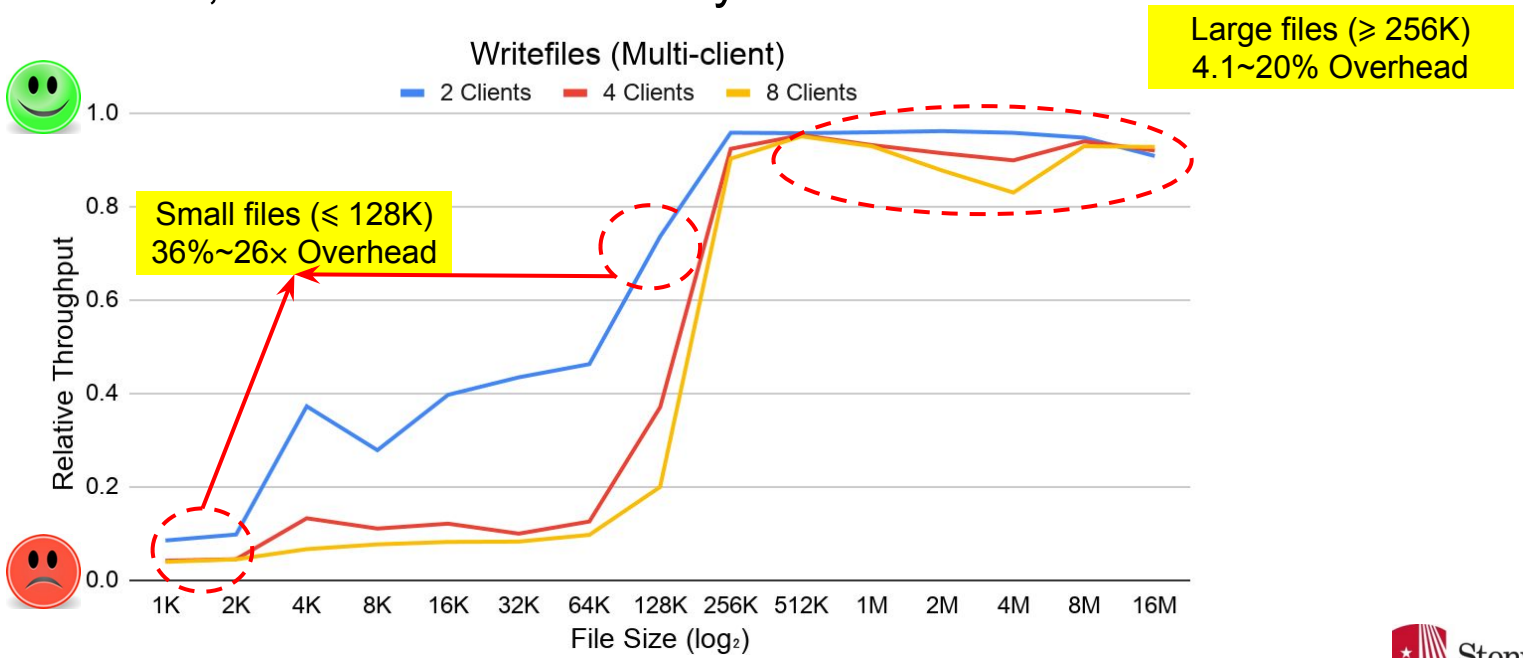
Stony Brook University

Experimental Setup

- 3 identical machines, 1 Server + 2 Clients
  - Each client machine runs 4 KVM virtual machines
  - Each VM runs Ubuntu 18.04 and one vNFS/NFSv4 client
- CPU: Intel Xeon X5650
- RAM: 64GB
- Storage
  - 147GB hard drive for system disk (ext4)
  - 200GB Intel DC-S3700 SSD for server's TC-NFS backend storage (XFS)
- Network
  - 10GbE NIC connected via 10GbE switch
  - average RTT = 0.2ms
- OS: Ubuntu 18.04 with Linux Kernel v4.15

Micro-Benchmark: Writefiles

- Writefiles (Multi-client)
  - Write 1,000 fixed-size files from 1K to 16M in parallel
  - 1~8 clients, 0.2ms network latency

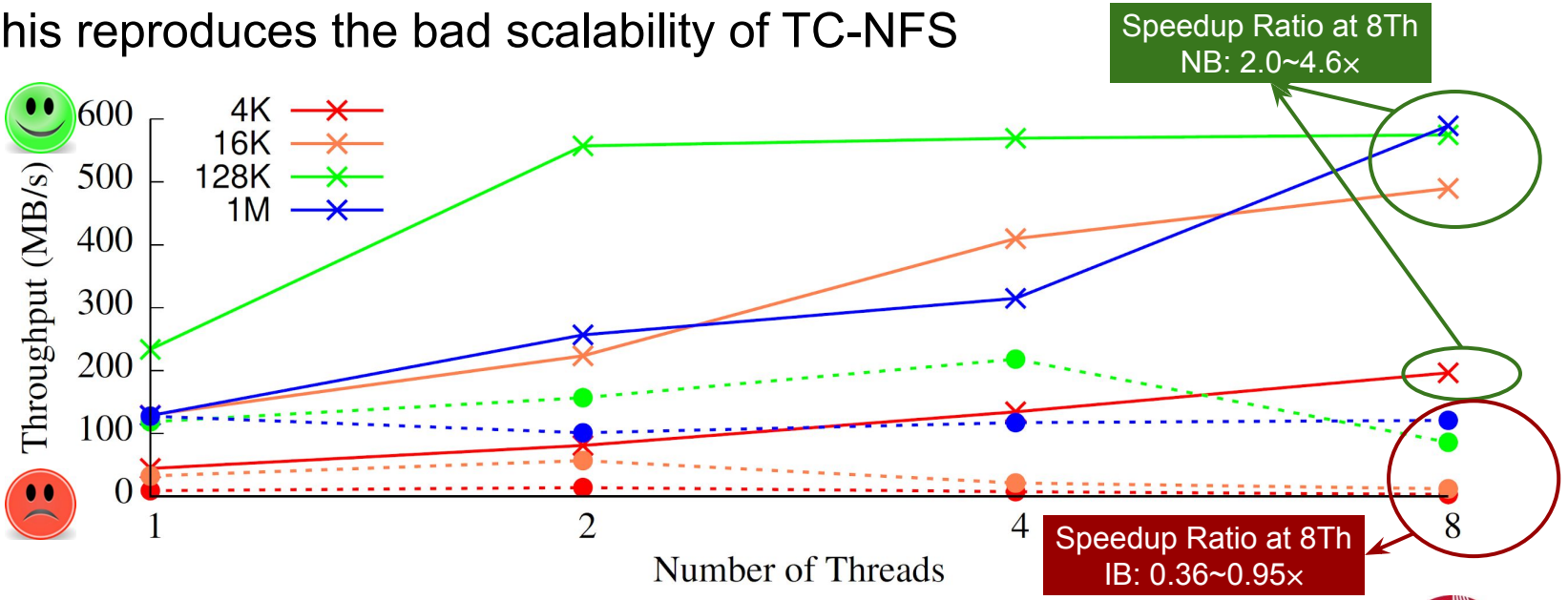


Explore the Bottleneck

- Local “Writefiles” Workload Simulation
  - Concurrently writes 1,000 equal-size files locally to the SSD using 1~8 threads repeatedly for 30s
  - `fsync()` is called after writing each file to simulate the behavior of the NFSv4 server (NFS-Ganesha)
  - Two types of workload
    - Interleaving-backup: Create backup for the target file before each `write()` operation using Copy-on-Write cloning
    - No-backup: Only do `write()` and `fsync()`, no backups

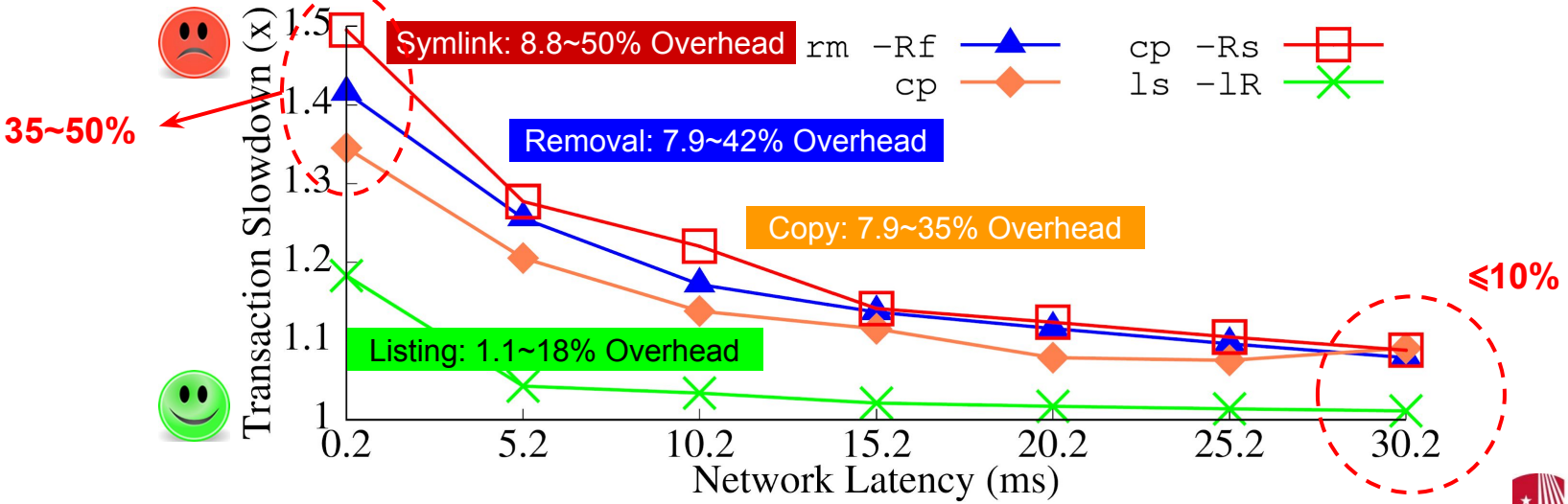
Exploring the Bottleneck

- Solid lines: No-backup; Dashed lines: Interleaving-backup
- No-backup (NB) workload scales well with number of threads
- Interleaving-backup (IB) workload does not scale or become worse
- This reproduces the bad scalability of TC-NFS



Macro-Benchmark: Coreutils

- Test target: Linux kernel 4.20.7 source tree
  - 62,447 regular files (Average size: 14.9 KB)
  - 4,148 directories (Average 15 children per directory
- Single-client, varied network latency between 0.2ms to 30.2ms
- Baseline: vNFS Client + Vanilla NFSv4 Server; Measured total runtime



Conclusions

- Provides transaction support for NFSv4 compounds
  - Makes error handling easier for applications
  - Currently supports the following operations: OPEN, CREATE, WRITE, LINK, REMOVE and simple RENAME (non-directories)
- Introduces modest overhead to single-client workloads and real-world applications
  - Considering the improvement vNFS provides, vNFS Client + TC-NFS is still much faster than traditional NFSv4 system
- Higher overhead in multi-client workloads
  - This is because CoW cloning + synced writes are slow on XFS
  - Will be resolved once the CoW feature is optimized

Supporting Transactions for Bulk NFSv4 Compounds

13th ACM International Systems and Storage Conference (SYSTOR 2020)

Wei Su[1], Akshay Aurora[1], Ming Chen[2], Erez Zadok[1]

[1]Stony Brook University; [2]Google

Paper: https://www.fsl.cs.sunysb.edu/docs/nfs4perf/tcnfs-systor2020.pdf
Project Source Code: https://github.com/sbu-fsl/fsl-tc-server

Q&A