Perils of the One-Size-Fits-All Kernel: A Fast, Secure Search for File System Metadata

Prajwal Challa

Mentors: Dr. Brad Settlemyer & Dominic Manno

LA-UR-20-26136

MOTIVATION

Consider the recent events at LANL where it was observed that SSDs in production had a breakdown after being up for certain number of hours. If you are a affected user, you are now left with plethora of questions like

- Is the data already written somewhere in the storage medium?
- Do I already have a copy?
- How do I effectively search entire LANL storage which has ~80PB of scratch, ~60PB of campaign storage and ~60PB of archival storage ?

GUFI

- GUFI- Grand Unified File Index
- GUFI performs fast and parallel queries while maintaining security considerations (using threads)
- Querying is done on copy of metadata on seperate machine
- A separate copy of metadata is kept to avoid resource contention on storage systems
- Place multiple indexes under parent index
- Uses SQLite for database processing

GUFI design

- Two phases (1) Indexing (2) Querying
- Indexing
  - Create GUFI index
  - copy permissions of source tree to index
  - Uses a breadth first manner
  - Store directory metadata in a single database
  - Database has entries table for file information, summary table for summary for the directory

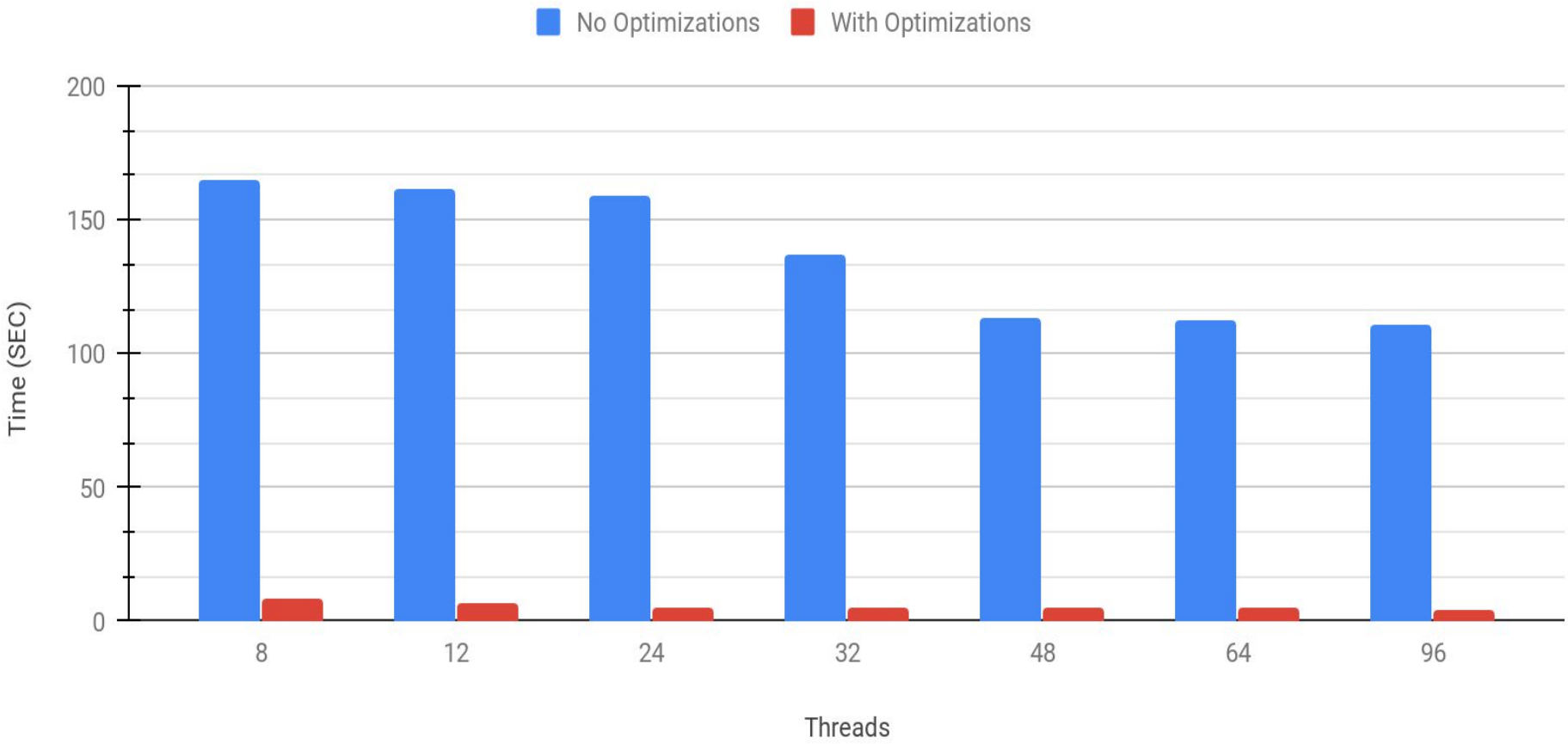
GUFI design

1. On the GUFI index querying is done as follows
    a. Open directory
    b. distribute subdirectories to threads
    c. Open, query and close database
    d. Close directory

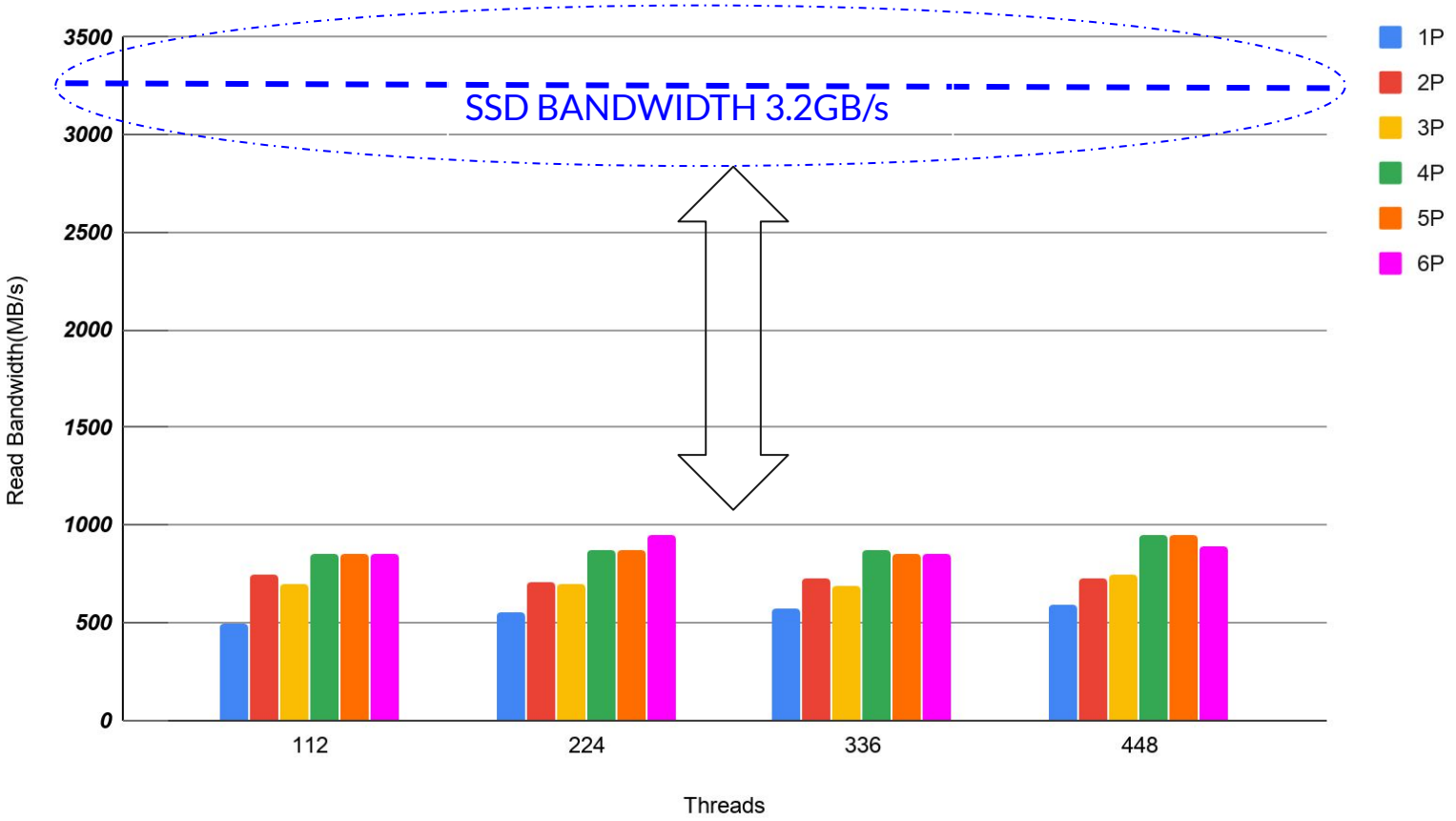
Optimizations

- Custom compile options
- Removed Sqlite protections in code
- Removed Sqlite runtime protections
  - stats .wal and .journal files
- Sqlite3_open_v2 VFS - unix none
- Jemalloc

Time to open 1.6 M dbs using sqlite3_open_v2()



Bandwidth performance for multiple processes for 9.6 M databases on 6 SSDs



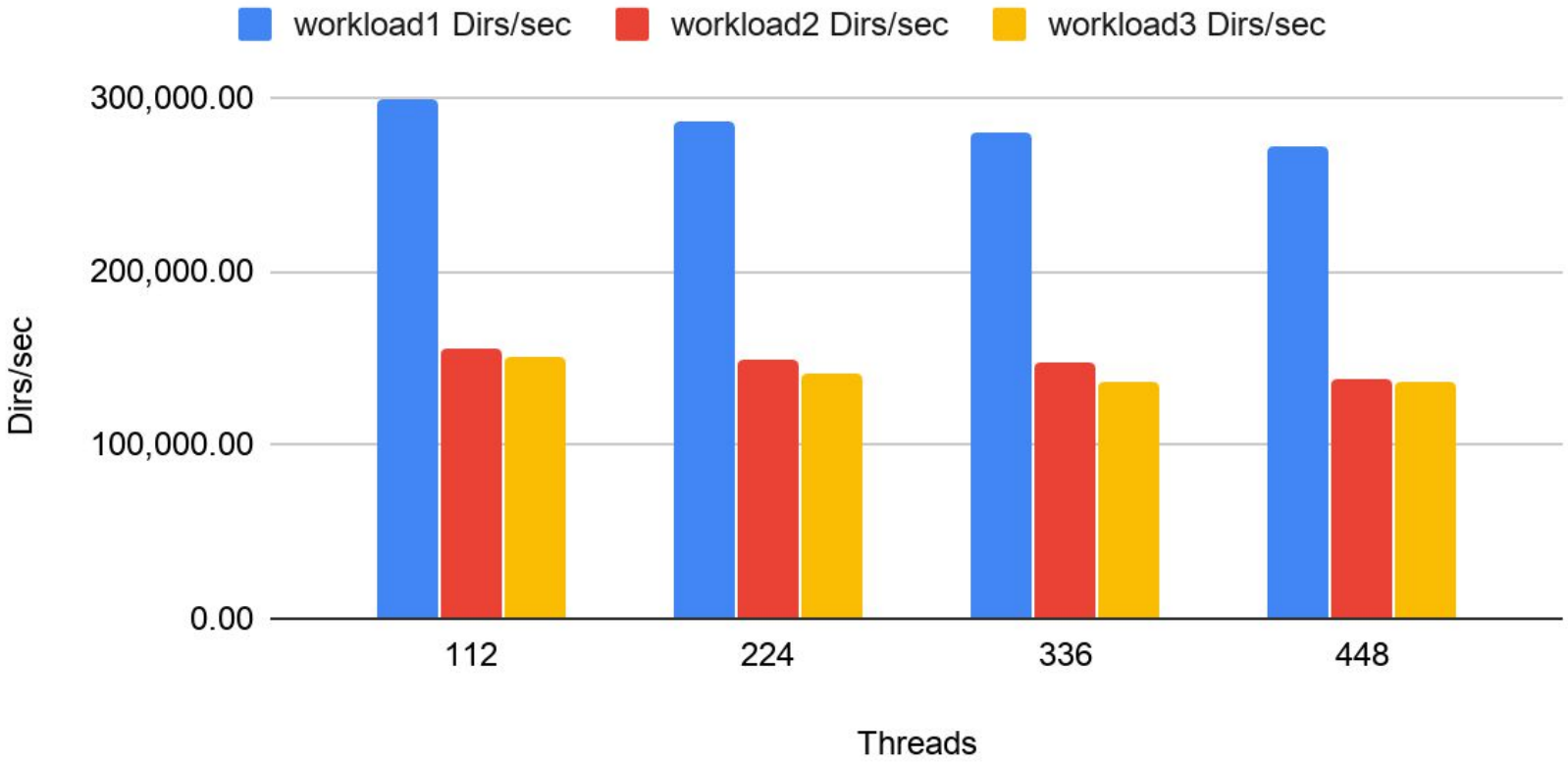
Workload characterization

- Great control over workload
- Number of directories, database sizes, depth were tuned
- Indexes ranged from few thousand directories to ~10 Mil directories with varying depth and database sizes

Sample Workloads

- Workload 1 is a GUFI index of production tree
- Workload 2 has twice the directories per level compared to workload 1 (index2 is twice wider than index 1)
- Workload 3 has twice the depth compared to workload 1 (index3 is twice deeper than index 1)

Directories per second for different workloads using GUFI on 1 SSD



Bandwidth performance of GUFI_FIND on 3 workloads (1 SSD)

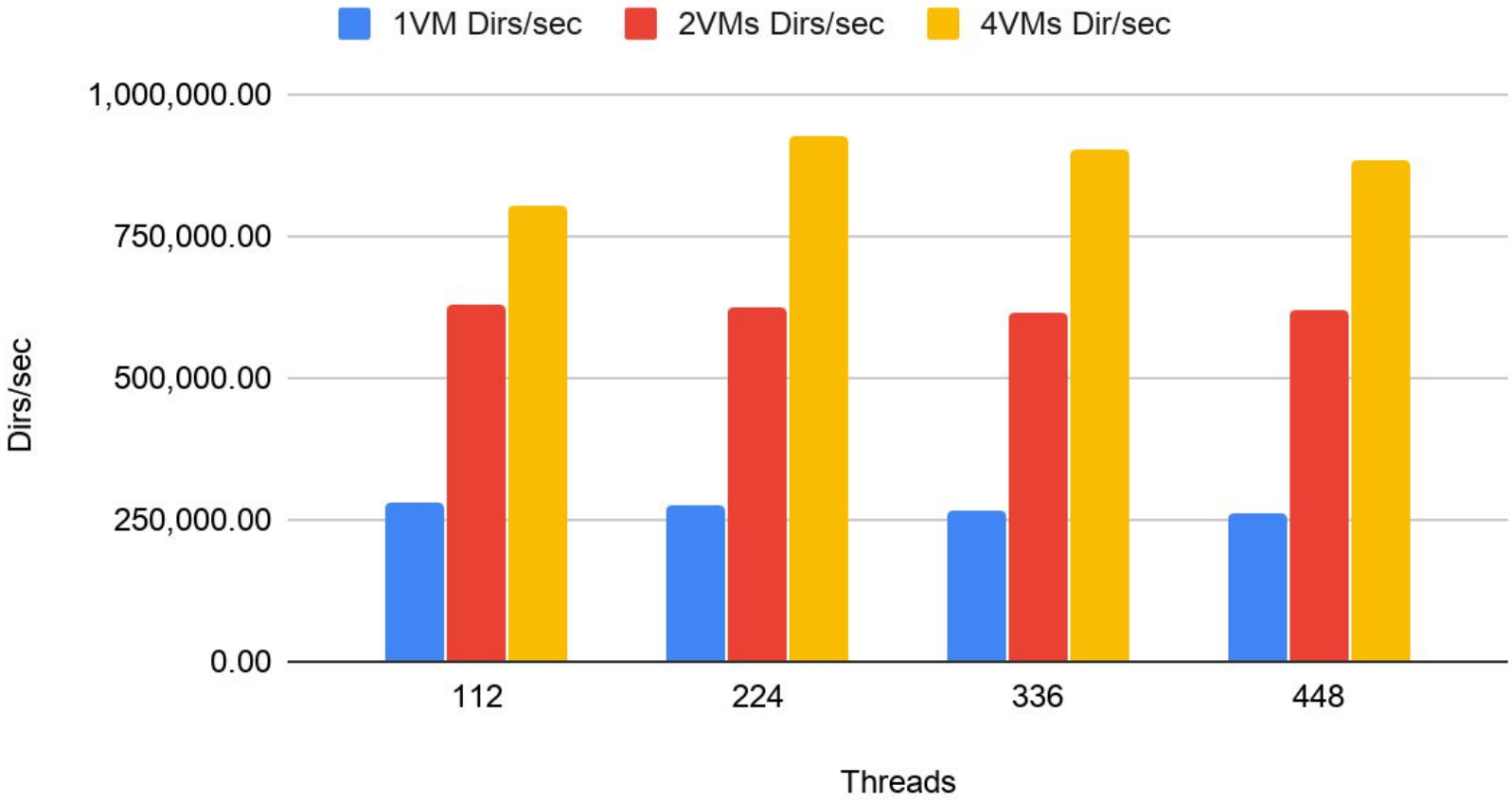
One size fits all?

- We explored different tree patterns using mdtest
- We observed near-linear scaling when mdtest experiments were run on multiple kernels
- Splitting permission checks and disk access doesn’t work well with kernel design where inode and dentry cache are required for permissions

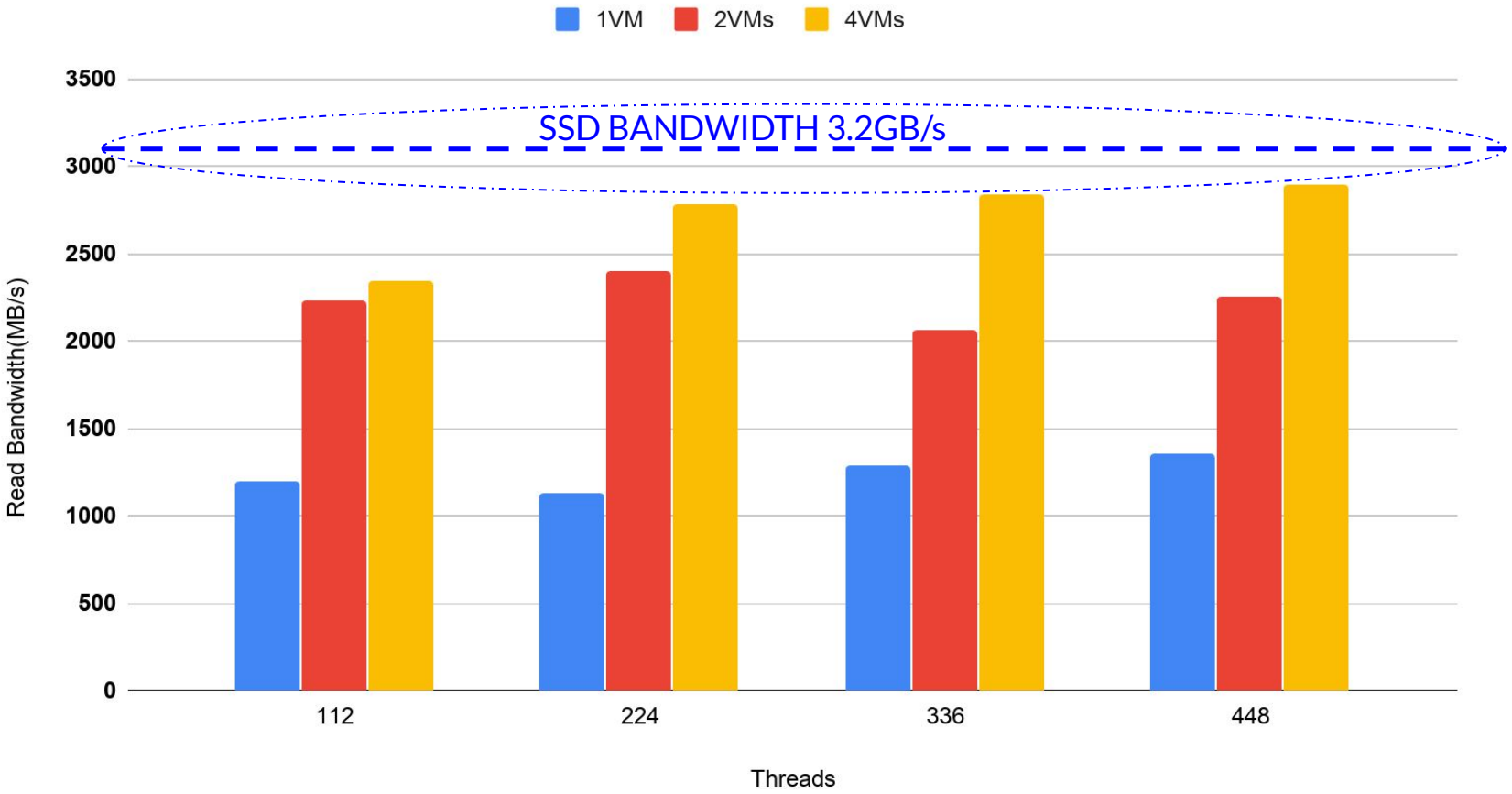
How to saturate NVMe SSD?

- Virtual Machines
- Easier to reuse verified kernel code than writing userspace code
- GUFI experiments were performed on multiple VMs sharing single NVMe ssd
- NVME-MDev for sharing ssd among VMs

Directories per second using GUFI for workload1 on 1 SSD



Bandwidth performance of SSD for workload1 using GUFI



Future work

- Where exactly is the bottleneck in kernel?

Thank you

- GUFI is available at https://github.com/mar-file-system/GUFI

QUESTIONS?