An Empirical Study of NVM-based File System

Hongwei Duan, Liang Shi, Qingfeng Zhuge,
Edwin Hsing-Mean Sha, Changlong Li, Yujiong Liang

East China Normal University

Contents

Background

- NVM-based File Systems
  - PMFS uses journaling for metadata updates.
  - NOVA adopts the log-structured file system techniques.
  - SIMFS fully utilizes the memory mapping hardware at the file access path.
- SSDPlayer is a graphical tool for visualizing the various processes that cause data movement on SSDs.

Motivation

- The increasing complexity of state-of-the-art NVM management justifies the adoption of new research and analysis techniques.
- There is no systematic performance study across different emerging file systems mentioned above.
- SSDPlayer is unsuitable for the study of NVM-based file system.

Evaluation: System Configuration

- CPU: a 2.4 GHz Intel E5-2640 10-core processor
- DRAM :
  - 48GB
  - 16GB acting as volatile memory for volatile data structures of the system
- NVM : 32GB acting as persistent memory for in-memory file systems
- OS : Ubuntu 4.4.30

Evaluation

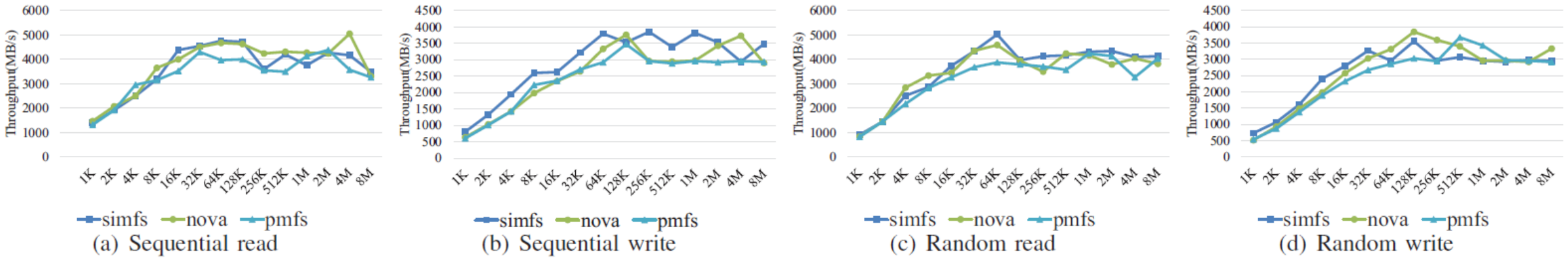

Fig. 1. Throughput comparison of three file systems with single thread.

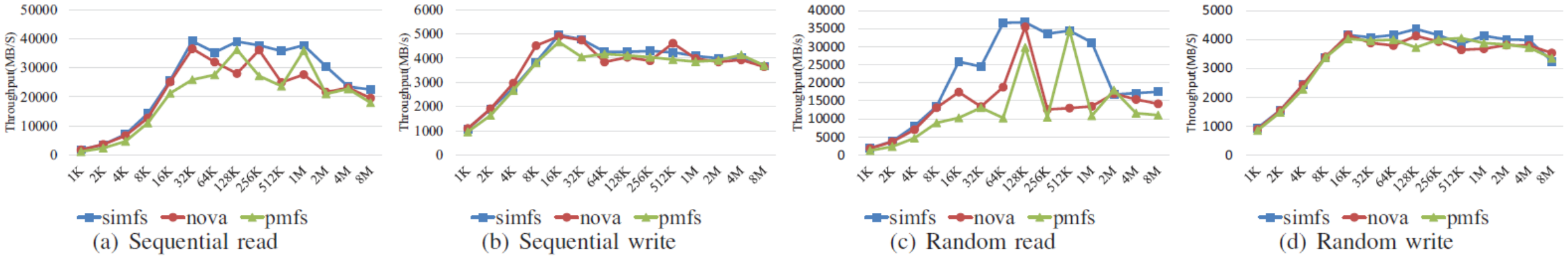

Fig. 2. Throughput comparison of three file systems with 16 threads.

Evaluation

SIMFS obtains better performance on the majority of cases except the openfiles.

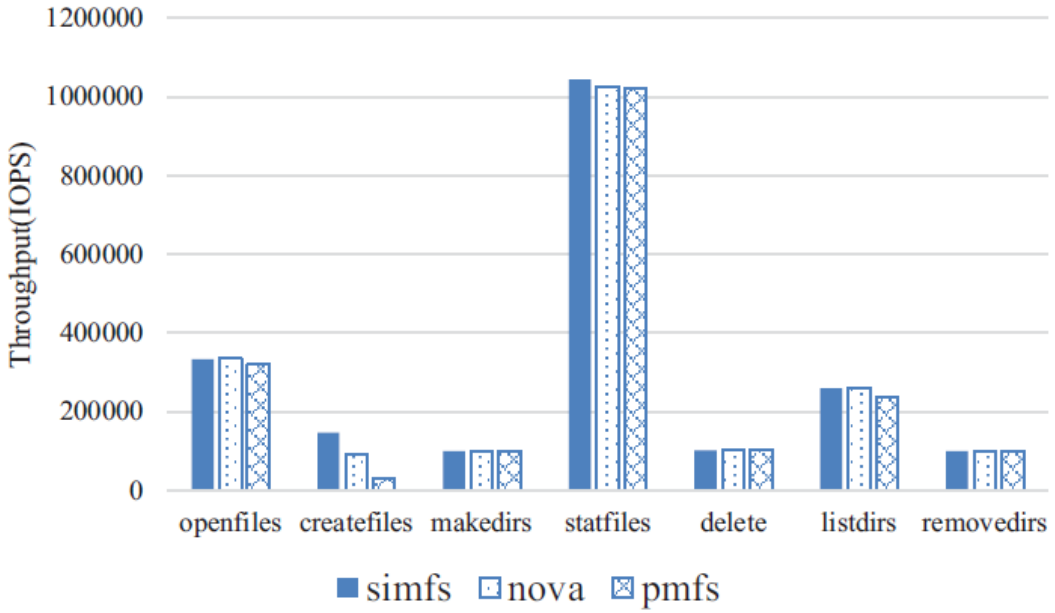

Fig. 3. Metadata operations .

SIMFS performs slightly worse than NOVA under webproxy loads and is the best under other loads.

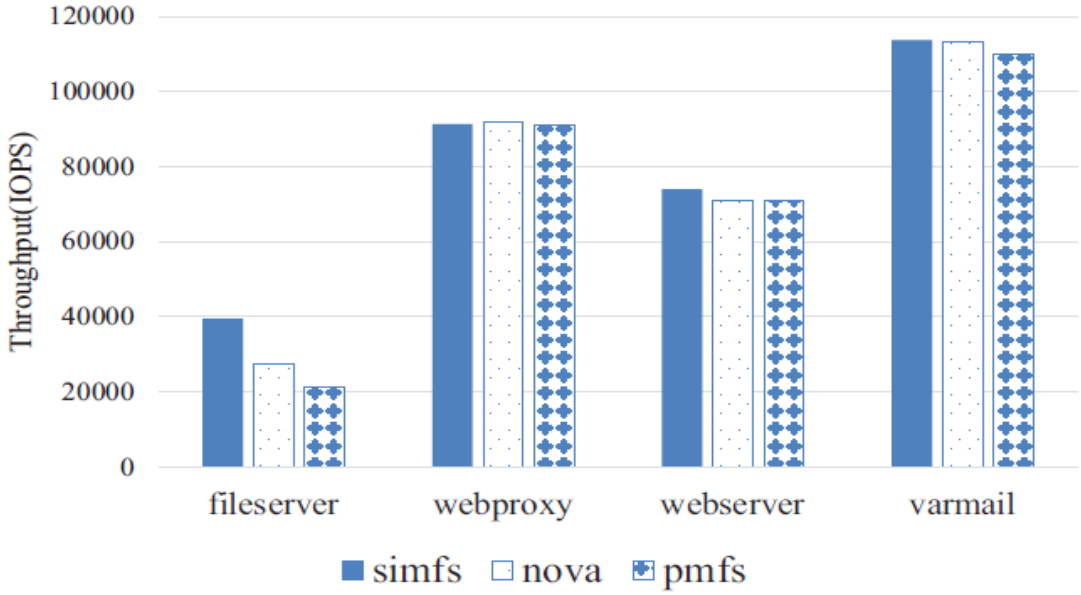

Fig. 4. Filebench application load .

Evaluation

Some NVM-based file systems that use journaling for metadata updates may suffer performance degration with the directory width increasing.

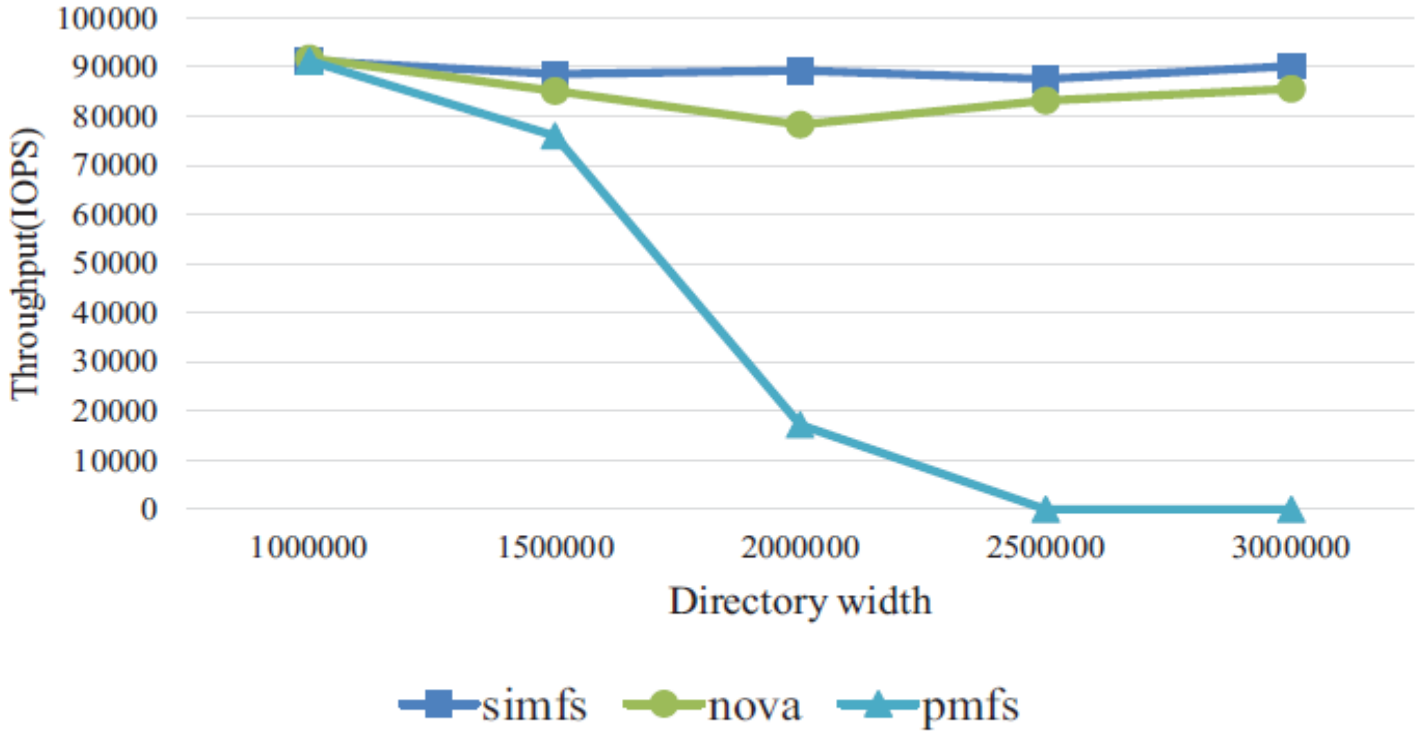

Fig. 5. Webproxy .

Evaluation

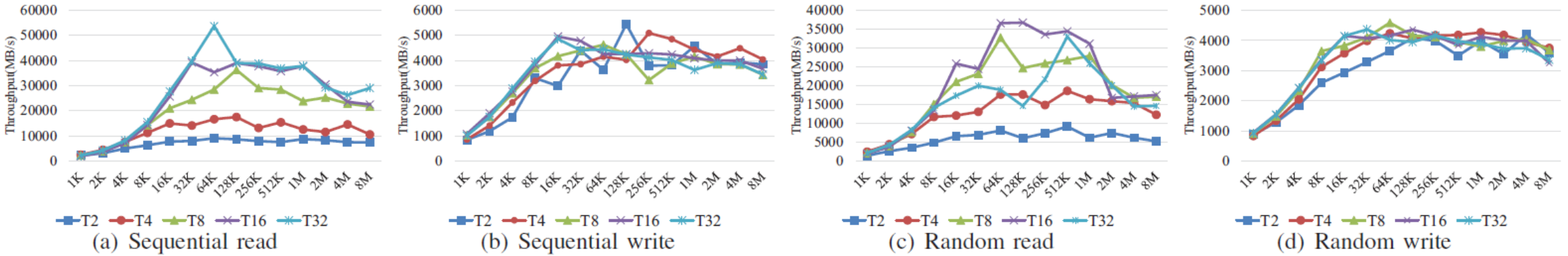

Fig. 6. Throughput comparison of SIMFS with varying number of threads.

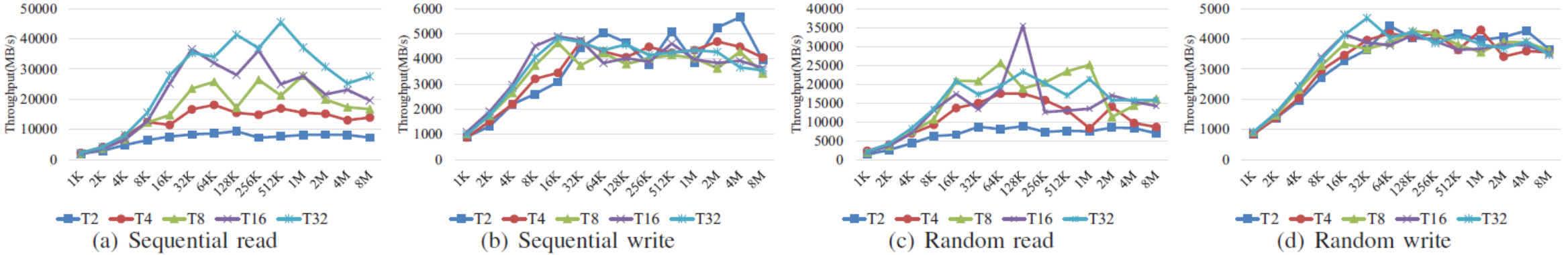

Fig. 7. Throughput comparison of NOVA with varying number of threads.

Evaluation: Observation

- Continuously increasing the number of threads, the performance will be worse, so we should limit the number of threads.
- Different access pattens have different thread saturation point. Usually sequential read has the highest saturation point, followed by random read.

Evaluation

When access granularity is too small, usually when access granularity is lower than 256B, the latency of NVM-based file systems increases sharply.

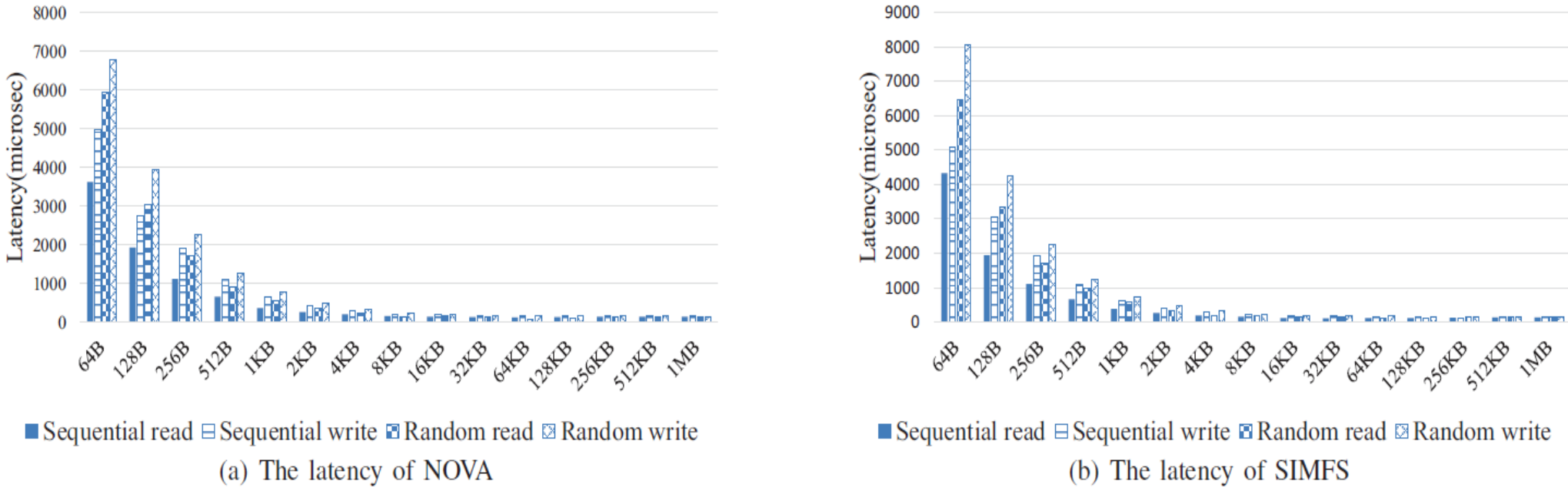

Fig. 8. Read/write latency under different access granularity.

Design: NVMPlayer

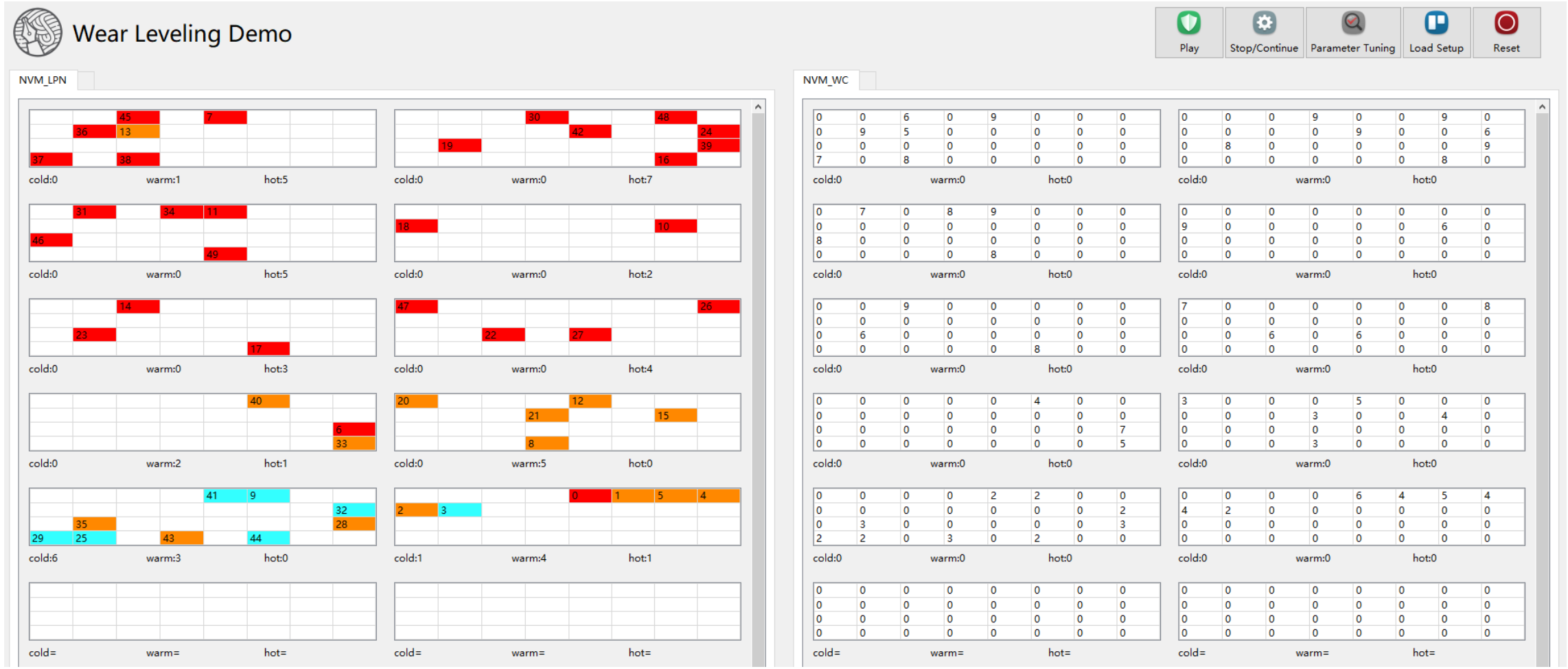

Fig. 9. Wear leveling demo

Conclusion

- Some NVM-based file systems may suffer performance degration with the directory width increasing.
- Continuously increasing the number of threads, the performance will be worse.
- Different access pattens have different thread saturation point.
- When access granularity is too small, the latency of NVM-based file systems increases sharply.
- Visualization should be a standard mechanism in the tool box of every NVM oriented research or development team.

Thanks for listening

Please feel free to ask any questions.