Libpubl: Exploiting Persistent User Buffers as Logs for Write Atomicity

Jaewon Jung, Jungsik Choi, Hwansoo Han

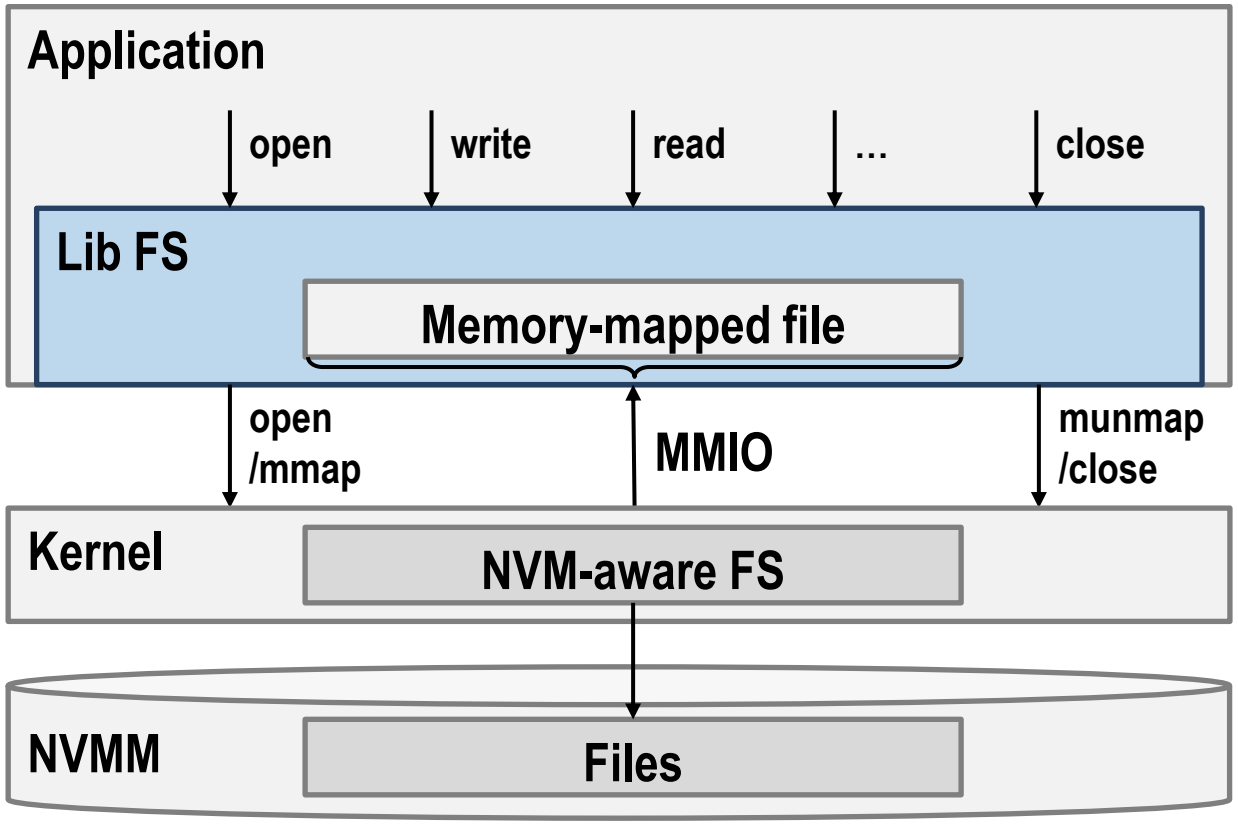
SUNG KYUN KWAN UNIVERSITY(SKKU)

HotStorage '21

Research for NVMM System

- Numerous NVM-aware file systems
  - Promising systems for large scale data centers
  - Many optimization opportunities
- Library FS
  - Runs on POSIX FS (DAX-mmap)
  - Memory-mapped IO for low latency access to NVMM
- Need to guarantee write atomicity
  - CoW or Logging
  - How to reduce write amplification

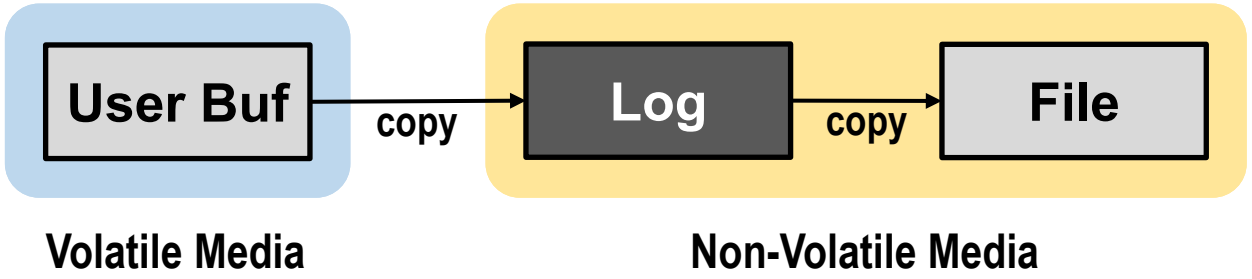
Lib FS



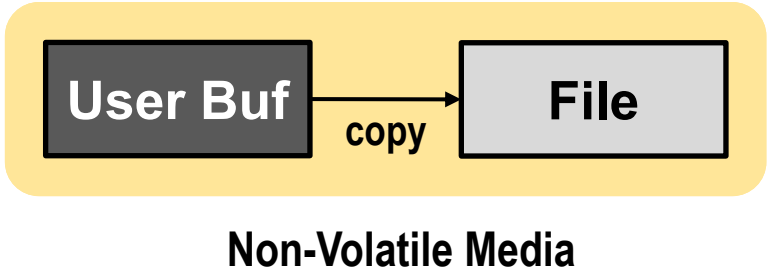
Design of Libpubl

- **Persistent user buffers as logs**
  - Persistent user buffers as logs
  - To minimize write amplification
- Background checkpoint
- Fine-grained range lock
- Recovery

Logging

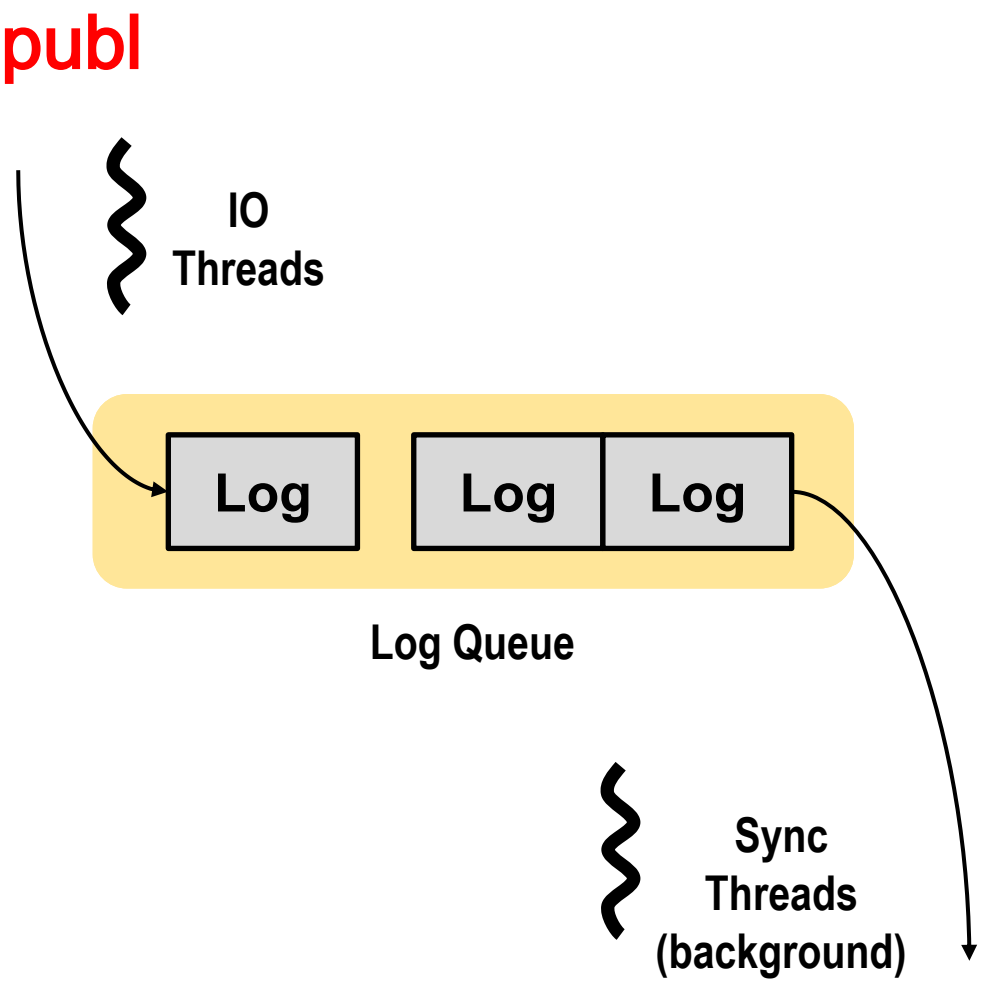


Libpubl



Design of Libpubl

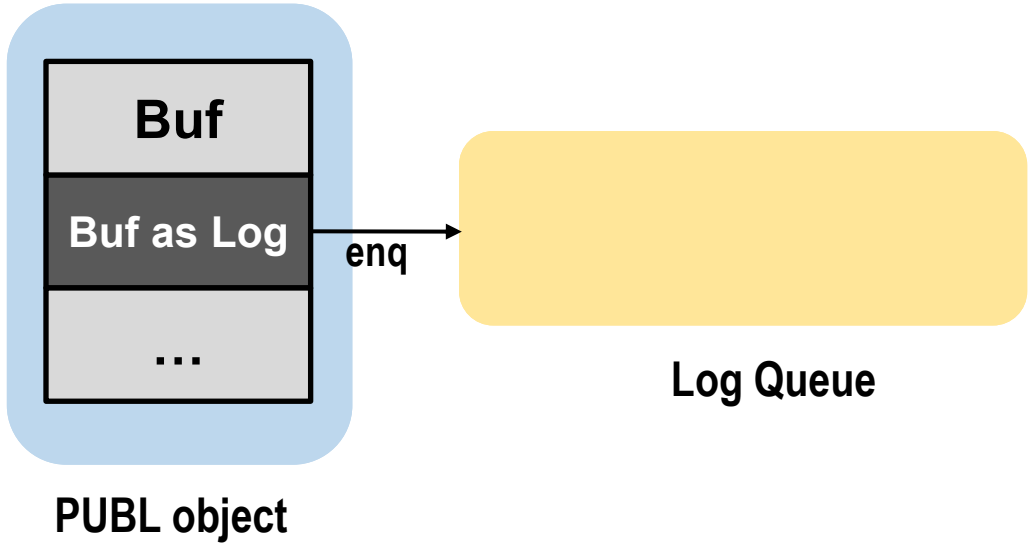
- Persistent user buffers as logs
- **Background checkpoint**
  - To hide checkpoint time
  - Only logging in write()
  - Wait early checkpoint or background checkpoint done
- Fine-grained range lock
- Recovery



Design of Libpubl

- Persistent user buffers as logs
- **Background checkpoint**
  - **PUBL object**
  - **# preliminary buffers**
- Fine-grained range lock
- Recovery

Libpubl



Design of Libpubl

- Persistent user buffers as logs
- **Background checkpoint**
  - **PUBL object**
  - **# preliminary buffers**
- Fine-grained range lock
- Recovery

Throughput improvement from various numbers of preliminary buffers

# Preliminary buffers	1	2	4	8	16
Early checkpoint (%)	0.02	0.01	0.00	0.00	0.00
Background checkpoint (%)	99.94	6.54	0.04	0.02	0.01
TOTAL wait (%)	99.96	6.55	0.04	0.02	0.01
Throughput (MB/s) *	1,872	3,337	3,348	3,399	3,394

*** Fio performance with sequential write to NVDIMM-N**

Design of Libpubl

- Persistent user buffers as logs
- **Background checkpoint**
  - **Logging and checkpointing**
- Fine-grained range lock
- Recovery

Performance improvement from background checkpointing

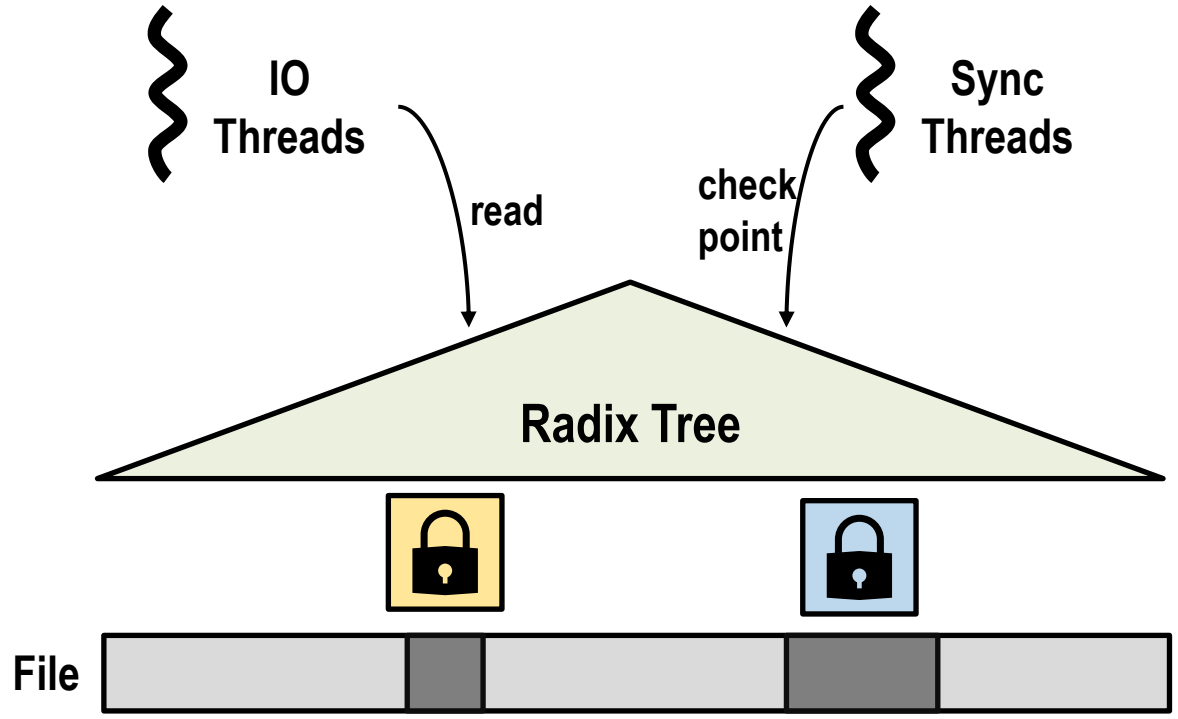
	Logging		Checkpointing (Libpubl)	
	No	Yes	Foreground-only	Background
Throughput (MB/s) *	2,755	1,749	2,754	3,363

*** Fio performance with sequential write to NVDIMM-N**

Design of Libpubl

- Persistent user buffers as logs
- Background checkpoint
- **Fine-grained range lock**
  - To provide scalability
  - Lock sizes ranging from 4KB to 2MB
- Recovery

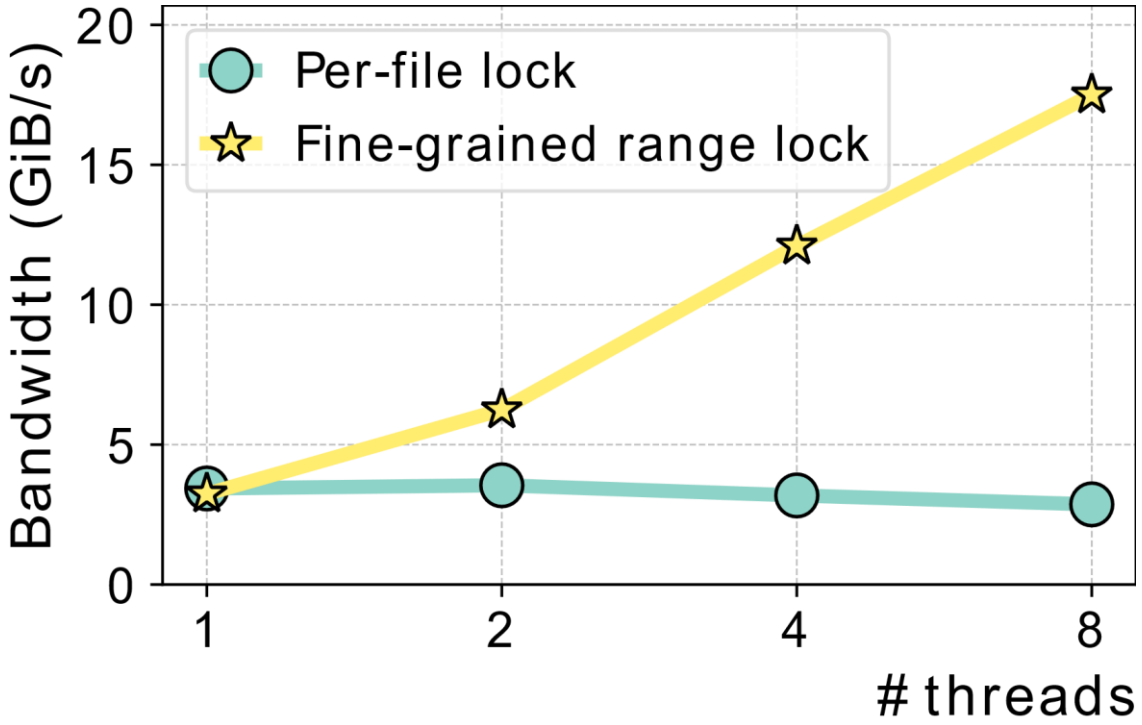
Libpubl



Design of Libpubl

- Persistent user buffers as logs
- Background checkpoint
- **Fine-grained range lock**
  - **To provide scalability**
  - **Lock sizes ranging from 4KB to 2MB**
- Recovery

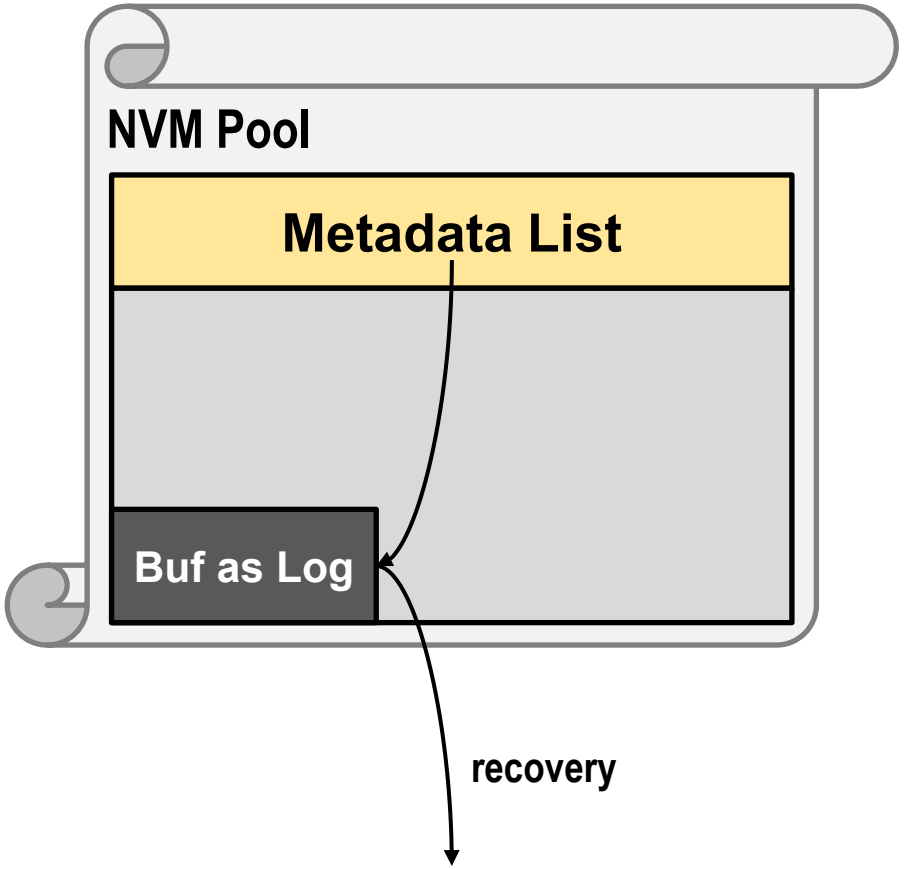
Scalability improvement from fine-grained range lock



Design of Libpubl

- Persistent user buffers as logs
- Background checkpoint
- Fine-grained range lock
- Recovery
  - Access the logs via their relative offsets
  - At every library initialization time

Libpubl



Experiments

- **Experiment environment**
  - **NVDIMM-N system**
    - **20 cores, 96GB DRAM**
    - **96GB NVDIMM-N**
  - **Optane system**
    - **72 cores, 768GB DRAM**
    - **3024GB Optanes**
    - **App Direct mode with interleaving enabled**
- **Microbenchmark, Fio**

Comparison of NVMM File Systems

Filesystem	File IO	Atomicity	NVDIMM	Optane
Ext4-DAX	Kernel	X	O	O
NOVA	Kernel	O	O	X
Libnvmmio	User	O	O	O
Libpubl*	User	O	O	O

Experiments

- **Experiment environment**
  - **NVDIMM-N system**
    - **20 cores, 96GB DRAM**
    - **96GB NVDIMM-N**
  - **Optane system**
    - **72 cores, 768GB DRAM**
    - **3024GB Optanes**
    - **App Direct mode with interleaving enabled**
- **Microbenchmark, Fio**

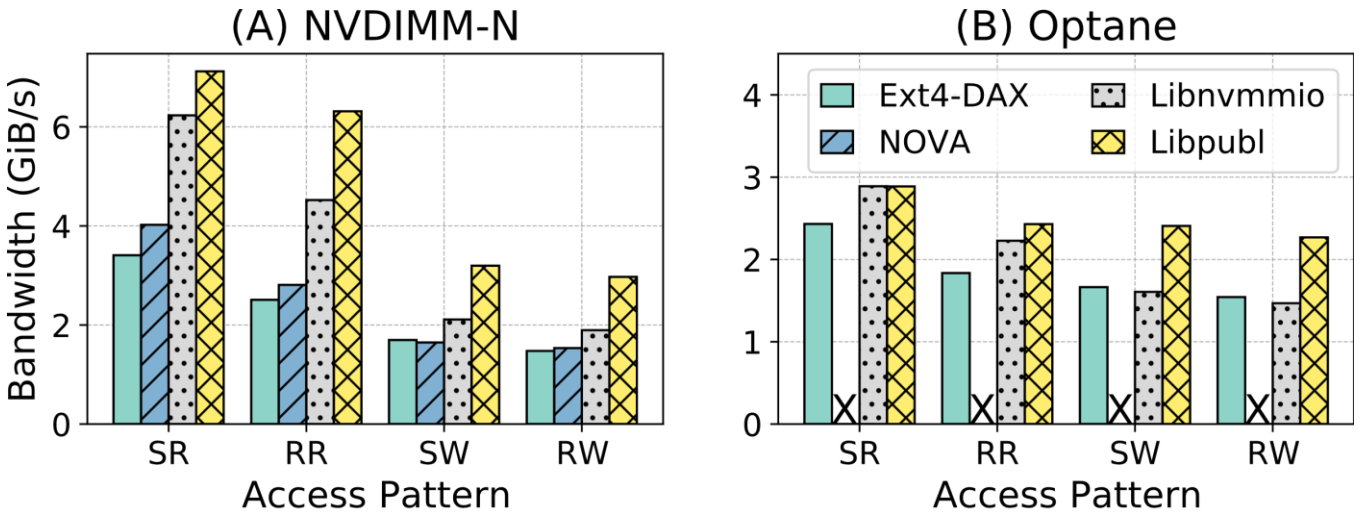
Comparison of NVMM File Systems

Filesystem	File IO	Atomicity	NVDIMM	Optane
Ext4-DAX	Kernel	X	O	O
NOVA	Kernel	O	O	X
Libnvmmio	User	O	O	O
Libpubl*	User	O	O	O

Experiments

- Experiment environment
- Microbenchmark, Fio
  - Various Access Patterns
  - Various Write Sizes
  - Multi-Threaded Write

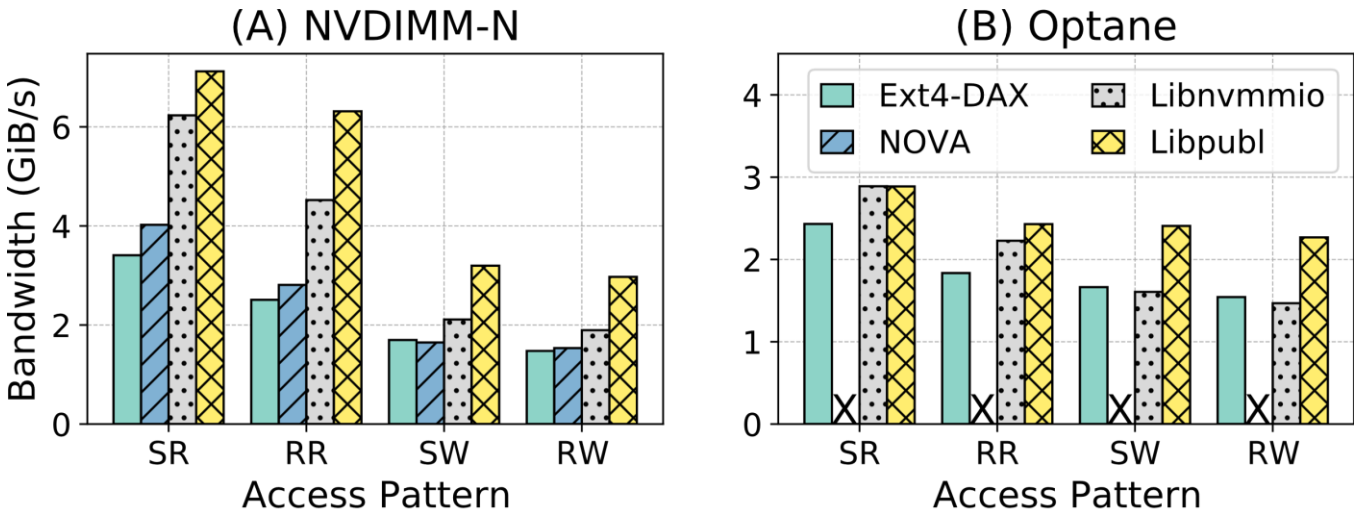
•A single thread, FS=1GB, BS=4KB, T=15s, refill_buffers



Experiments

- Experiment environment
- Microbenchmark, Fio
  - Various Access Patterns
  - Various Write Sizes
  - Multi-Threaded Write

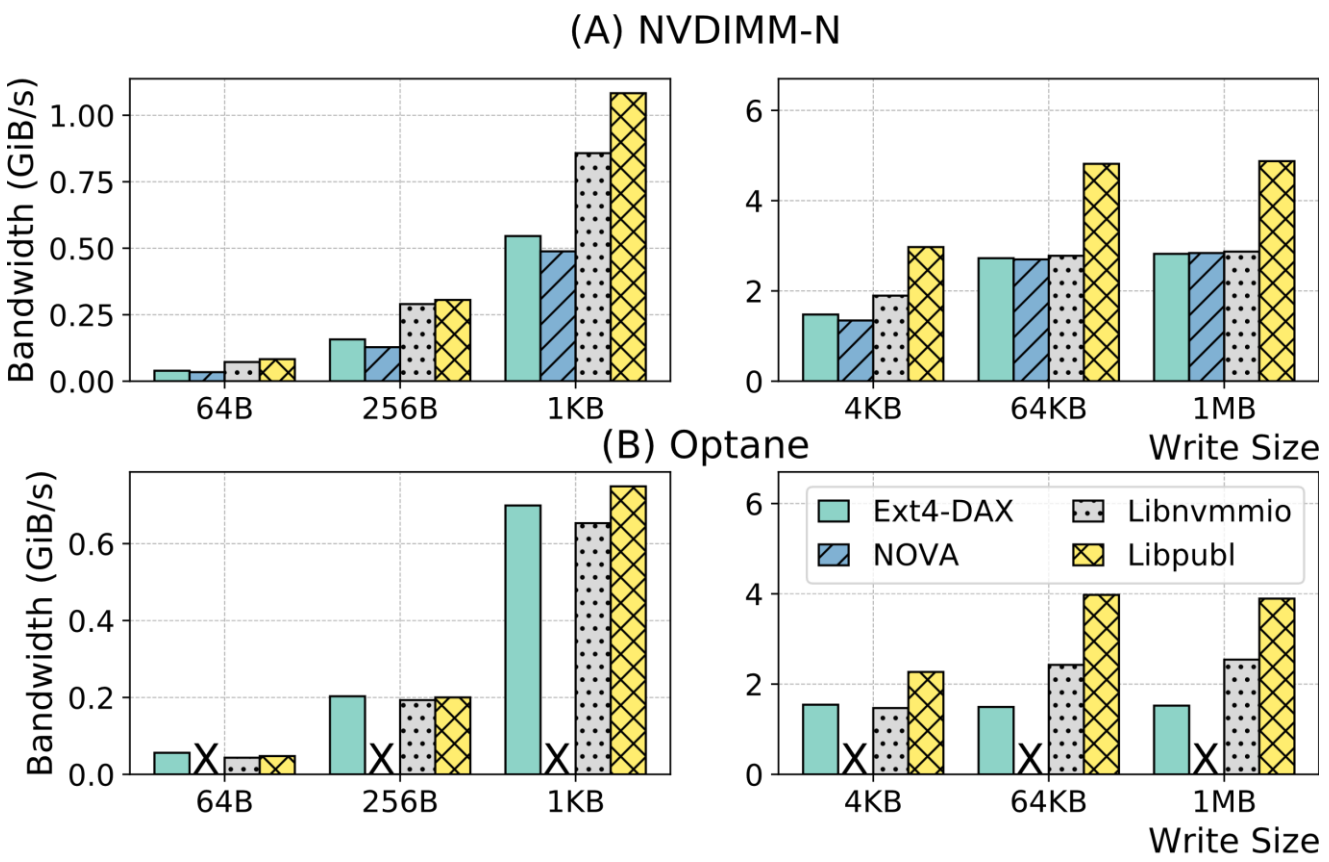
•A single thread, FS=1GB, BS=4KB, T=15s, refill_buffers



Experiments

- Experiment environment
- **Microbenchmark, Fio**
  - Various Access Patterns
  - **Various Write Sizes**
  - Multi-Threaded Write

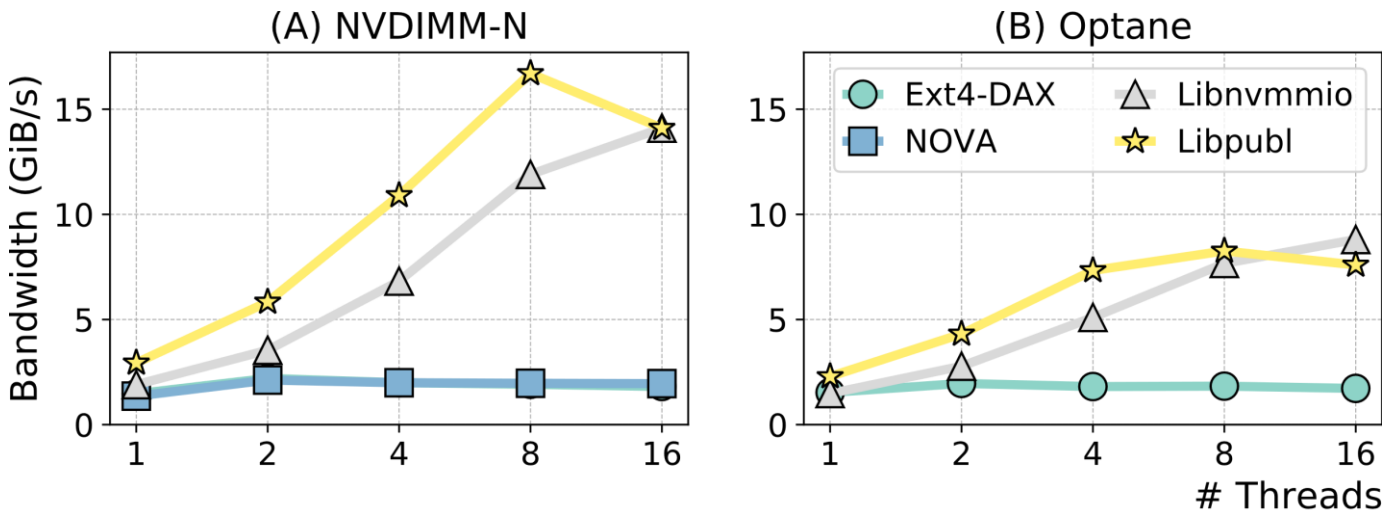
•A single thread, RW, FS=1GB, T=15s, refill_buffers



Experiments

- Experiment environment
- **Microbenchmark, Fio**
  - Various Access Patterns
  - Various Write Sizes
  - **Multi-Threaded Write**

•Multi threads, RW, FS=1GB, BS=4KB, T=15s, refill_buffers



Conclusion

- **Low latency & Scalable IO while guaranteeing write atomicity**
  - **Managing persistent user buffers as logs to reduce write amplification**
  - **User-level library filesystem to enable IO without system call**
  - **Background checkpointing to hiding checkpointing time**
  - **Fine-grained range locking using radix tree**

- **Performance**
  - **2.08x better throughput**
  - **8.7x better scalability**