

FlashLogging:

*Exploiting Flash Devices for
Synchronous Logging Performance*

Shimin Chen

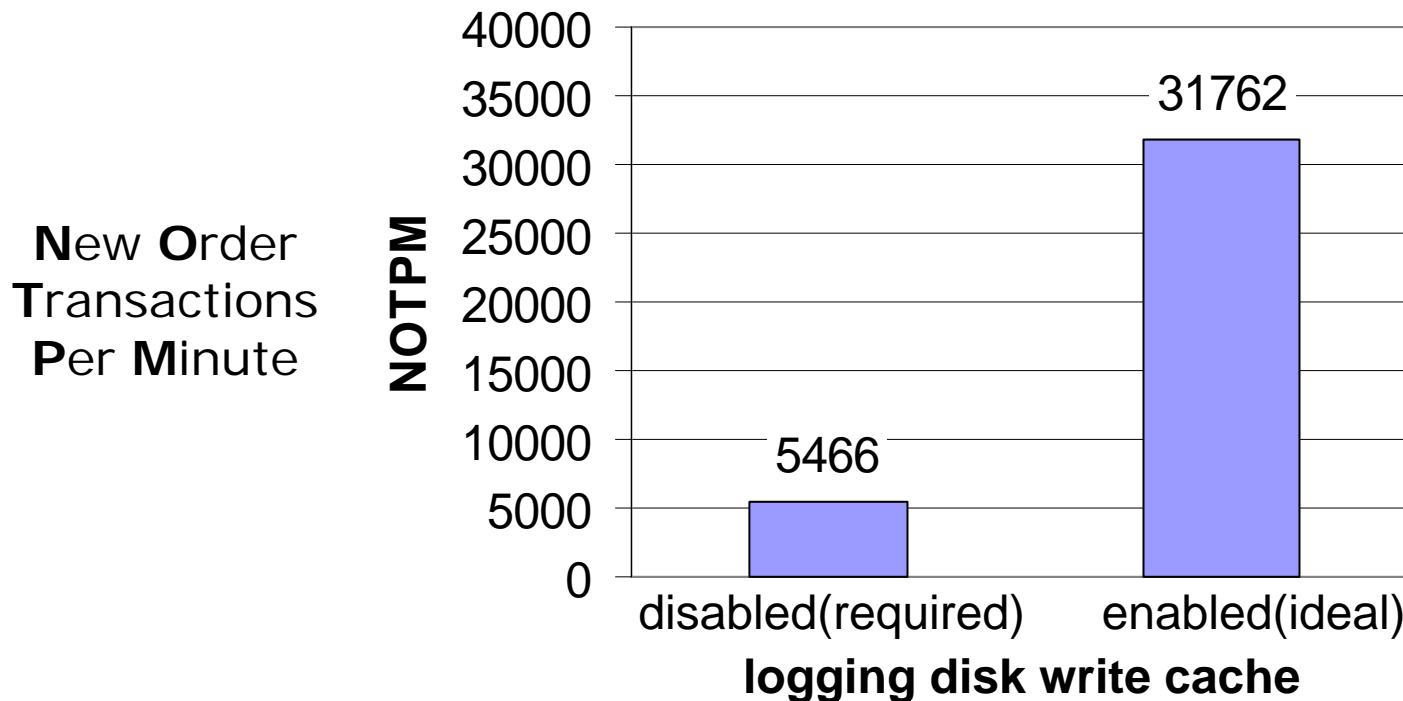
Intel Labs Pittsburgh

Synchronous Logging

- Central mechanism to ensure transaction durability
 - At transaction commit time, flush redo log records
- OLTP: DB Size < (aggregate) Memory Size
 - e.g., TPCC, 30 million users require < 100GB space
 - Exploited in previous studies: [Stonebraker et al. 2007], [Harizopoulos et al. 2008], ...
- In contrast, synchronous logging requires writing to stable media

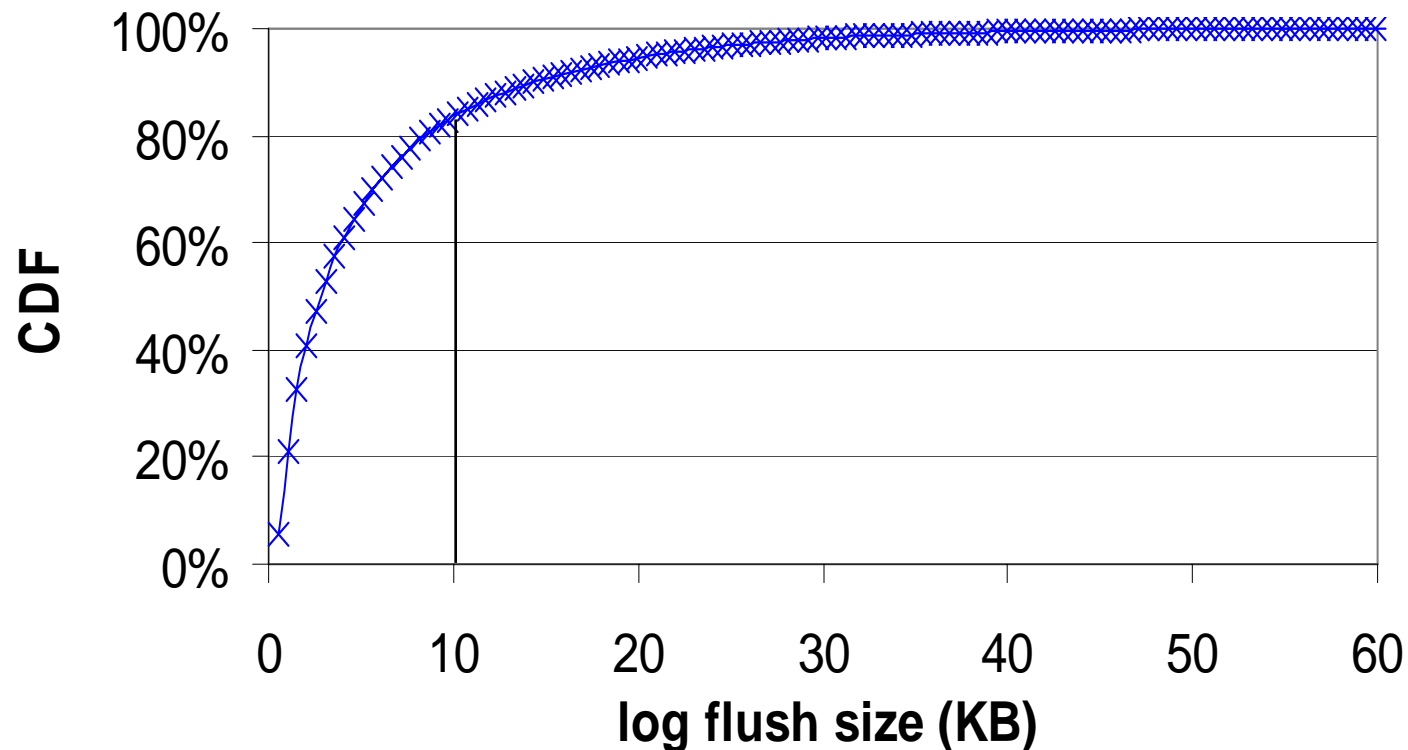
 Synchronous logging plays an important role in OLTP performance

Using Magnetic Disk as Logging Device



- TPCC Experiments:
 - One blade runs MySQL+InnoDB, the other runs TPCC driver
 - 20-warehouse TPCC DB
 - 10k rpm disk as logging device
- Magnetic-disk-based logging is a major performance bottleneck

Magnetic Disks Suffer From Small Sequential Writes



- Cumulative distribution of log flush sizes
- Rotating disks ill-suited for the pattern

Our Approach: Exploiting Flash Devices

- **Efficient:**
 - Solid state, no moving parts
 - Flash devices handle small sequential writes well
- **Inexpensive:**
 - \$/GB drops quickly
 - Small-capacity package
- **Simple:**
 - Changes to DB code are limited to the logging subsystem

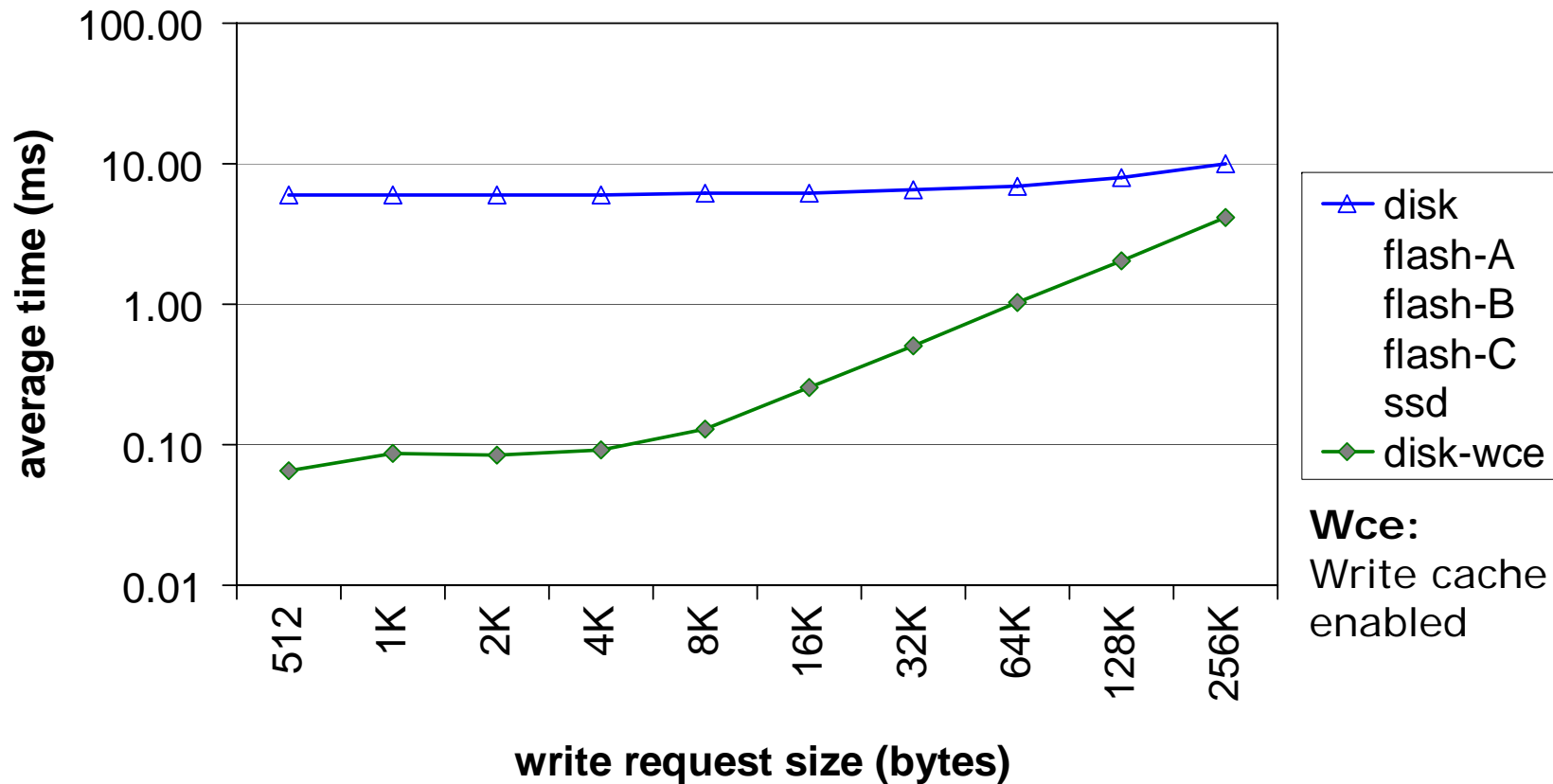
Contributions

- Find that even low-end flash devices can work well
 - USB flash drives are a good match for synchronous logging
- Address four key challenges:
 - Efficiently exploiting an array of flash drives
 - Coping with large variance of write latencies
 - Efficient recovery processing
 - Combining USB flash drives with disks for better performance
- Achieve up to 5.7X improvements over magnetic-disk-based logging, 98.6% of the ideal performance

Outline

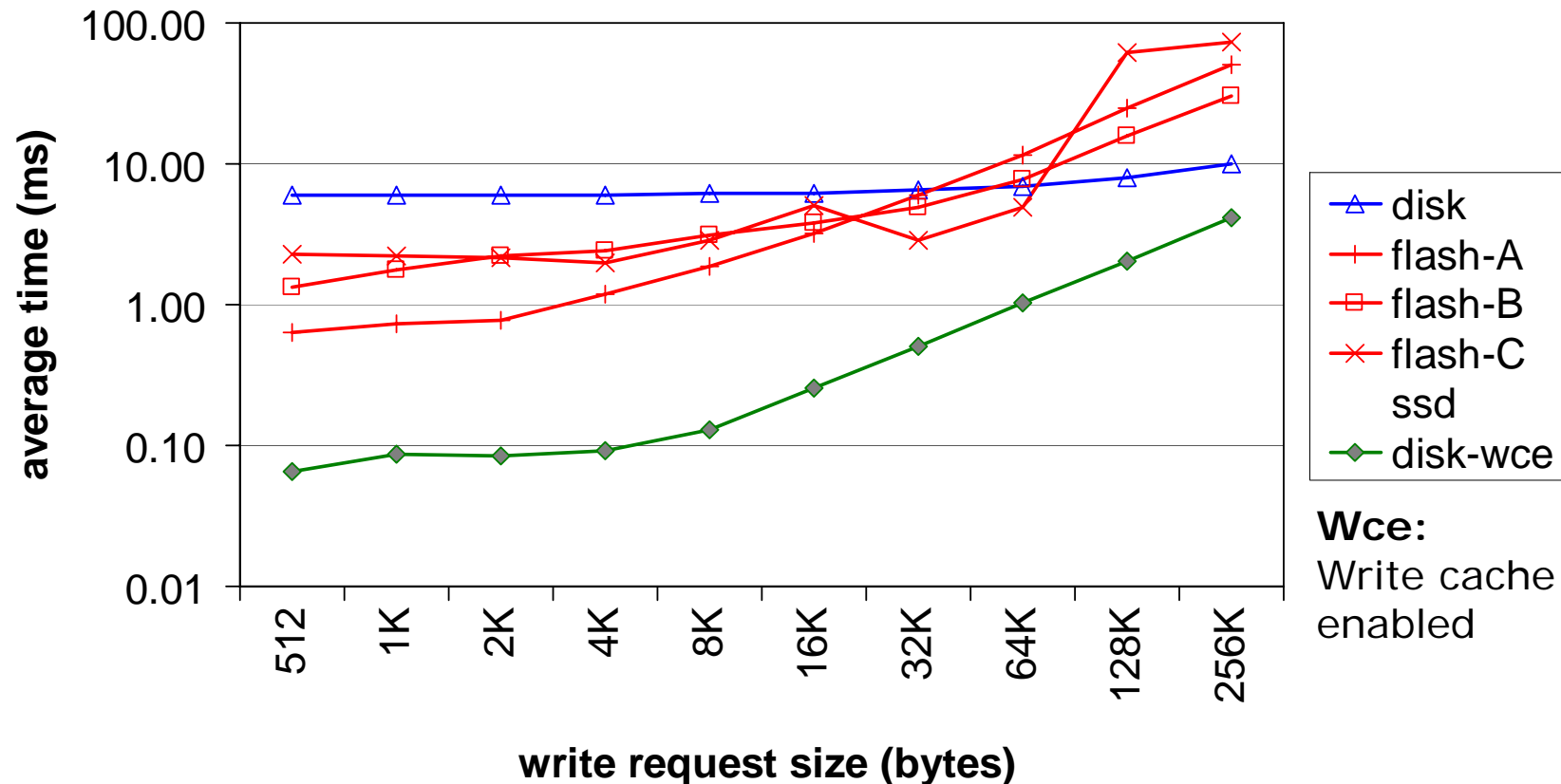
- Introduction
- The Case for exploiting USB Flash Drives for Synchronous Logging
- FlashLogging Design
- Performance Evaluations
- Related Works
- Conclusions

Sequential Write Performance



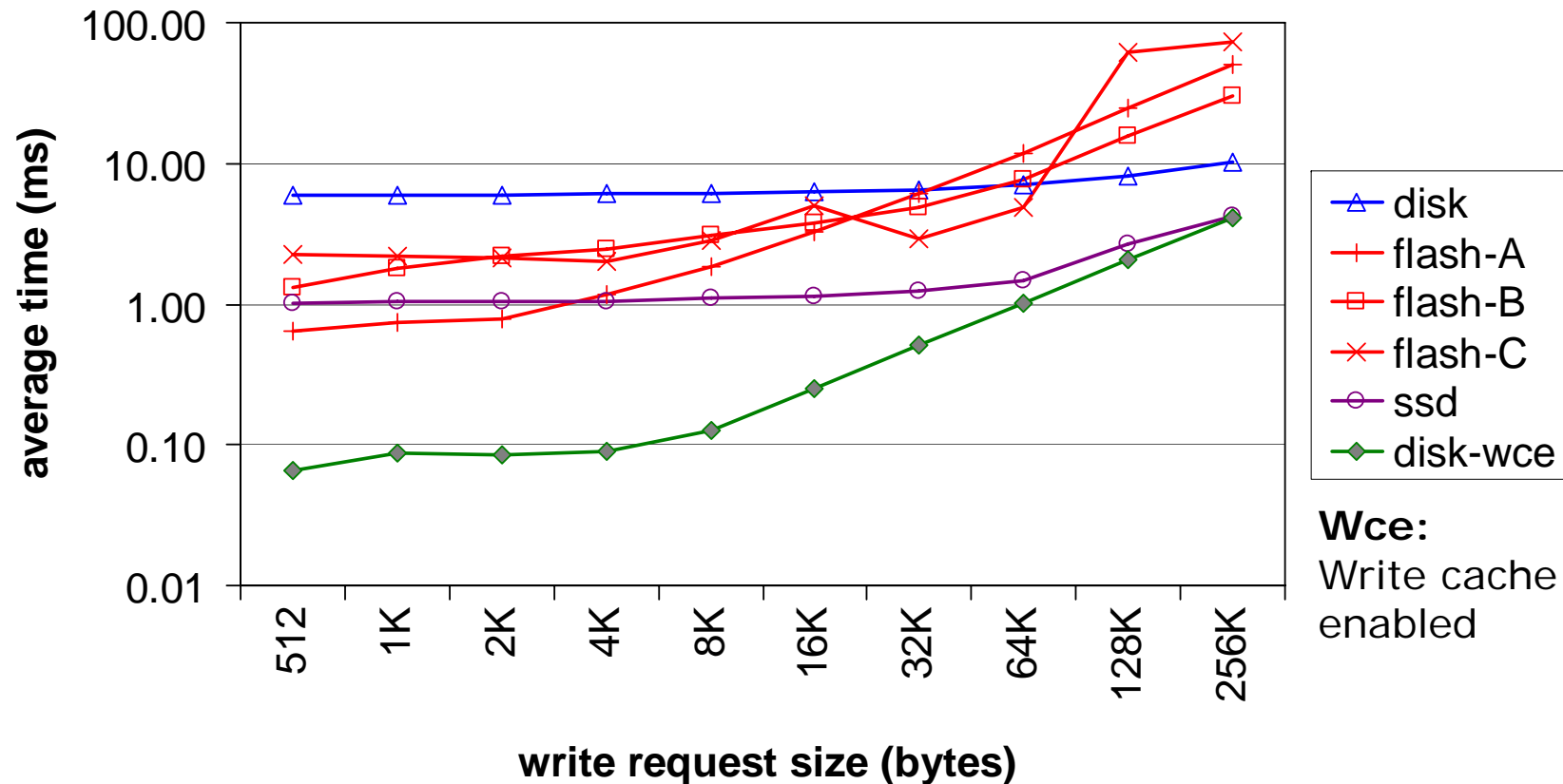
- Disk suffers from small sequential writes

Sequential Write Performance



- Low-end flash devices: three types of USB flash drives
- Much better performance for small writes (< 32KiB)

Sequential Write Performance



- High-end flash device: SSD
- USB flash drives have comparable performance for small writes

USB Flash Drives are a Good Match for Synchronous Logging



Compared to flash memory cards and SSDs

- USB ports are widely available
- USB bus bandwidth good for logging:
 - USB 2.0: 60MB/s; USB 3.0: 600MB/s
- Price of individual drive is low:
 - We can use multiple devices for better performance
- Hot-plug capability allows easy replacement
 - An 8GB drive can last ~462 days

Outline

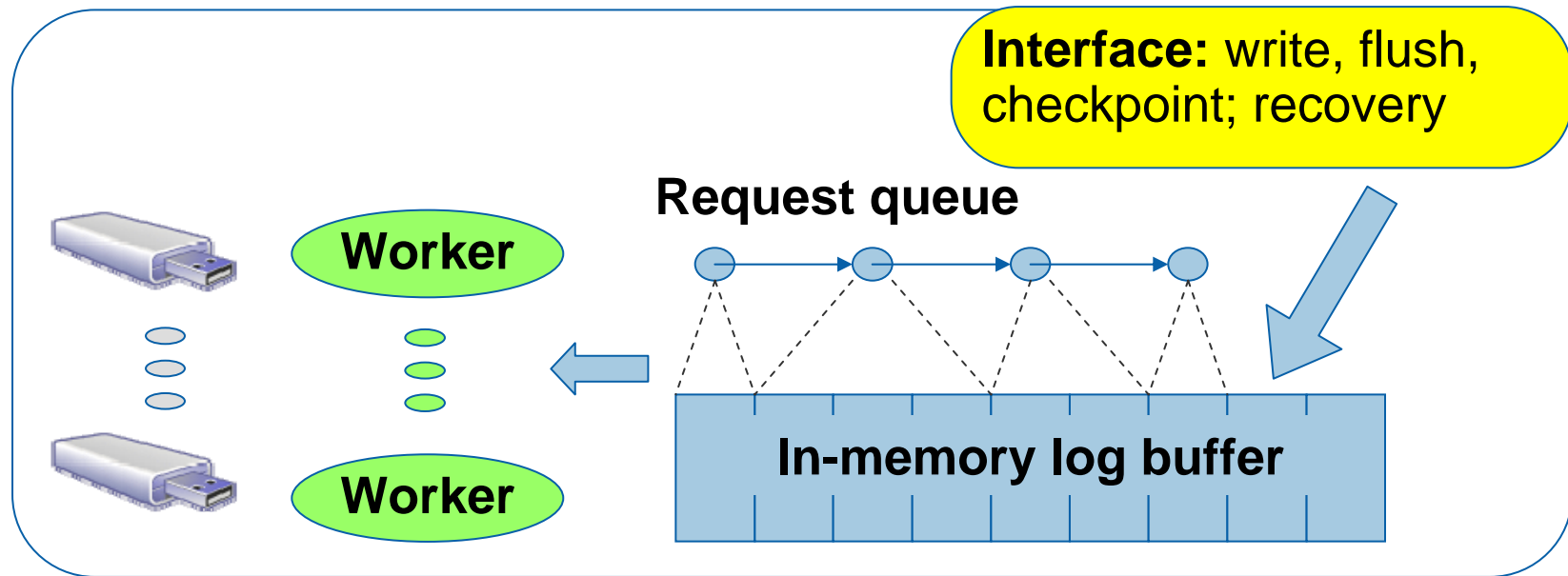
- Introduction
- The Case for exploiting USB Flash Drives for Synchronous Logging
- FlashLogging Design
- Performance Evaluations
- Related Works
- Conclusions

Synchronous Logging Requirements

We analyze MySQL-InnoDB logging system:

- **Online log:** circular and kept small for fast recovery
- **Normal processing:**
 - Sequential log flushes
 - Checkpoint: records LSN of oldest dirty page in buffer pool
 - **Mainly sequential writes**
- **Recovery processing:**
 - Locate last correct checkpoint
 - Given recorded LSN, find the corresponding log records
 - Scan log till crash point
 - **Mainly sequential reads**

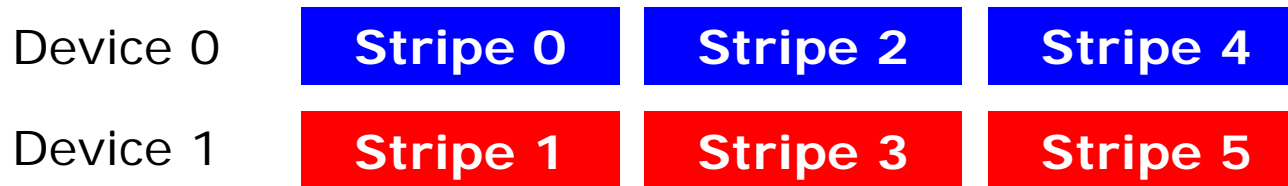
FlashLogging Architecture



- Simple producer-consumer structure

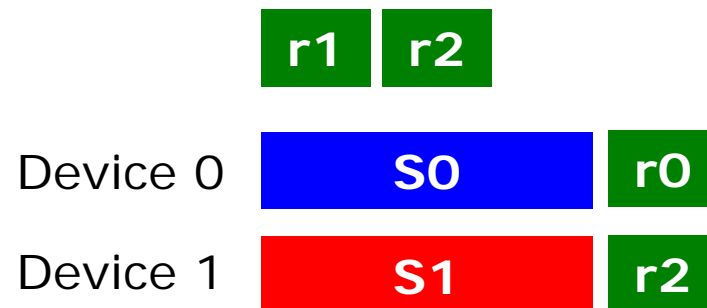
Exploiting an Array of Flash Drives

- Conventional array design: round robin, striped



- Pro: fast random access
 - Given LSN, directly compute device ID and address
 - But not very useful for synchronous logging

- Con:

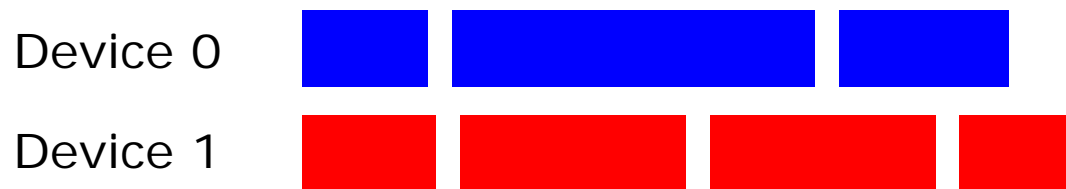


Request skipping
if stripe > log flush size

Request splitting
if stripe < log flush size

Our Solution: Unconventional Array

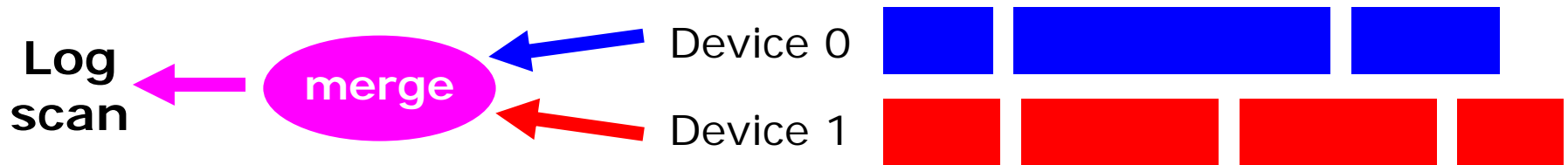
- Invariant: **LSNs on a device are non-decreasing**
- Log flush:
 - An entire flush request can always go to a single device
 - Avoid request splitting or skipping



- Rare Random accesses handled by binary search
 - Improvement: build an index structure to narrow search range (please see paper)

Recovery Processing

- Locate starting point (checkpoint LSN):
 - Random access to the log
- Log scan:
 - Non-decreasing LSNs on every log device
 - Scan all the devices in parallel
 - Merge data from all devices based on LSN

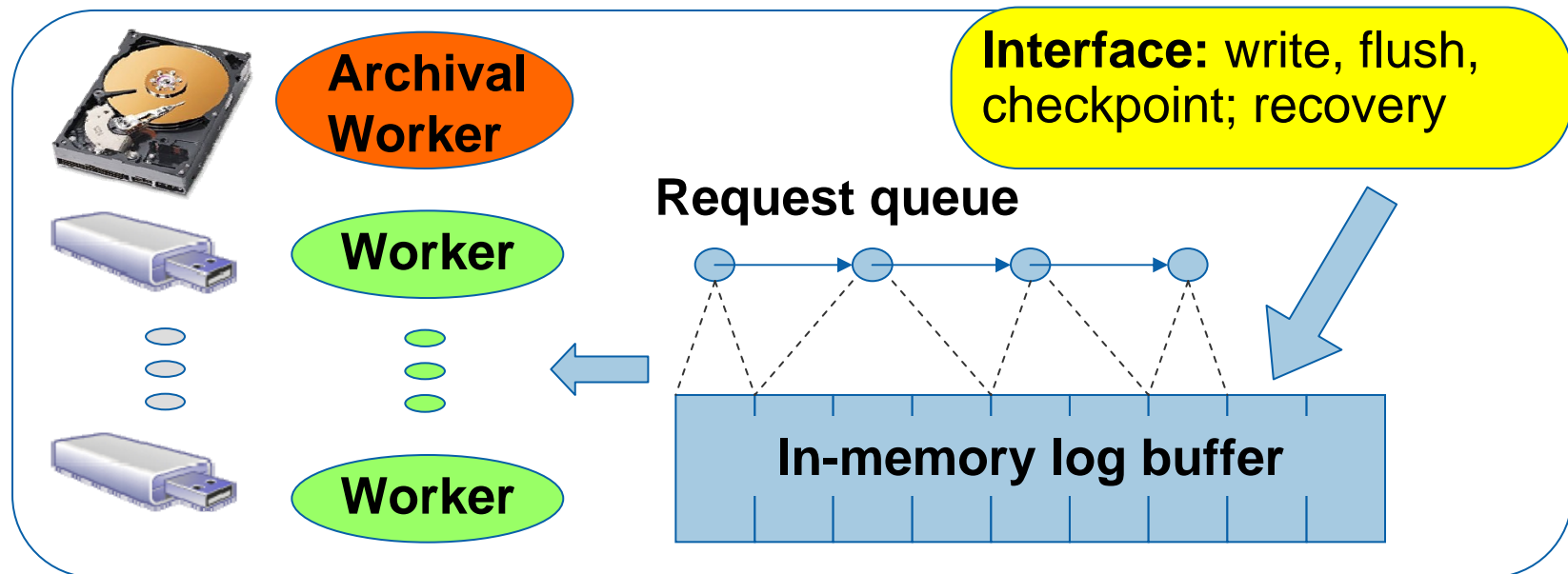


Coping with Outlier Writes

- Outlier writes:
 - elapsed time of outlier write \gg average time (size)
 - Flash device management operations (such as erasure)
 - Difference can be up to two order of magnitudes
- Predict outliers? difficult
 - Cannot predict the patterns using combinations of write request size and number of requests
 - Different devices have different patterns
- Our solution: outlier detection and hiding
 - Detection: write time $\geq 2 \times$ average time (size)
 - Hiding: re-issue the write to another ready logging device

Combining USB Flash Drives and Disks

- Why?
 - Logging: disk is competitive when write size $\geq 32\text{KiB}$
 - Recovery: disk has larger read bandwidth than USB flash drives



- Proposal: near-zero-delay archival disk
 - If new data in log buffer $\geq 32\text{KiB}$, flush data to disk
 - Wake up any frontend threads associated with the flushed data

Outline

- Introduction
- The Case for exploiting USB Flash Drives for Synchronous Logging
- FlashLogging Design
- Performance Evaluations
- Related Works
- Conclusions

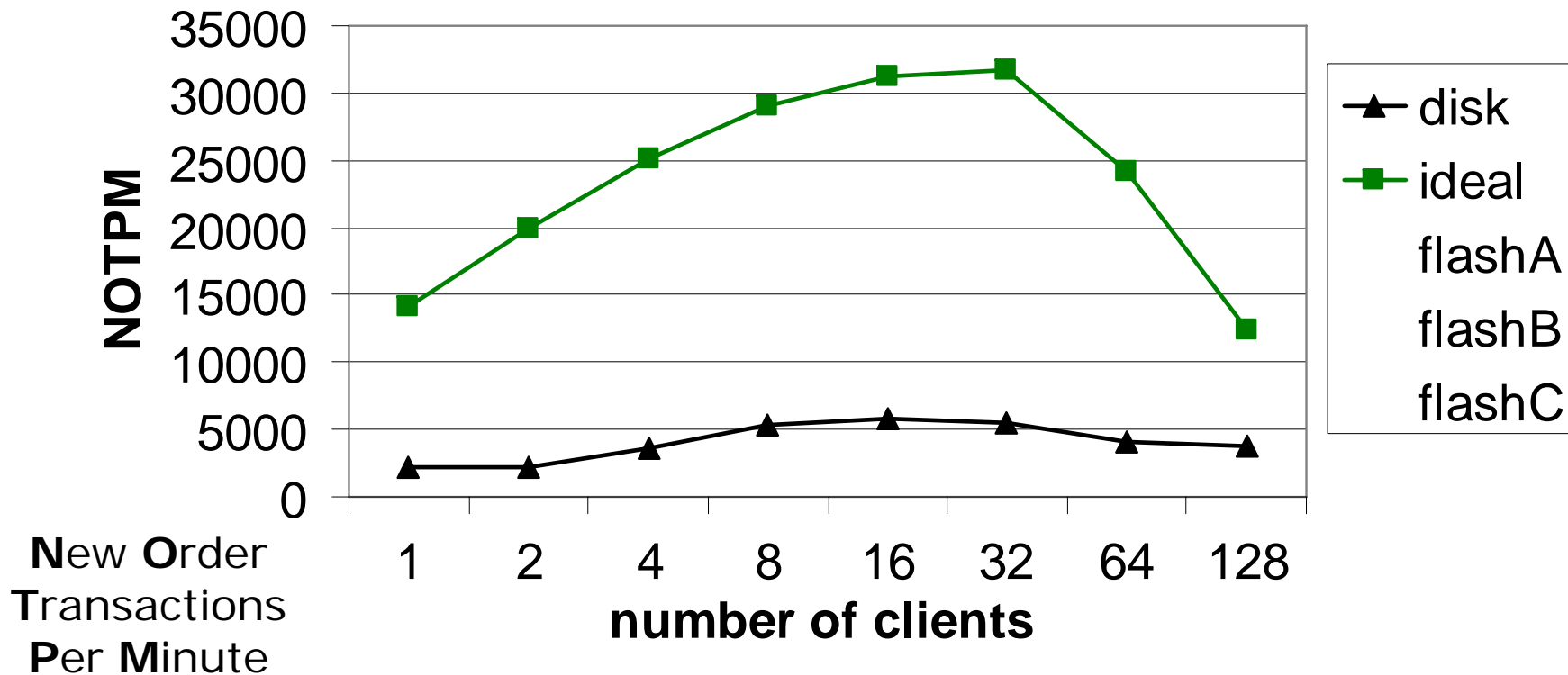
Implementation

- MySQL 5.0.24a with InnoDB as storage engine
- Implemented FlashLogging as logging subsystem
 - Supports raw devices
 - Avoid overlapping writes to the same blocks
- To be fair, for disk-based logging, we report the better of:
 - Original MySQL+InnoDB with log files on the log disk
 - Note: raw device is not supported
 - Flashlogging with raw log disk

Experimental Setup

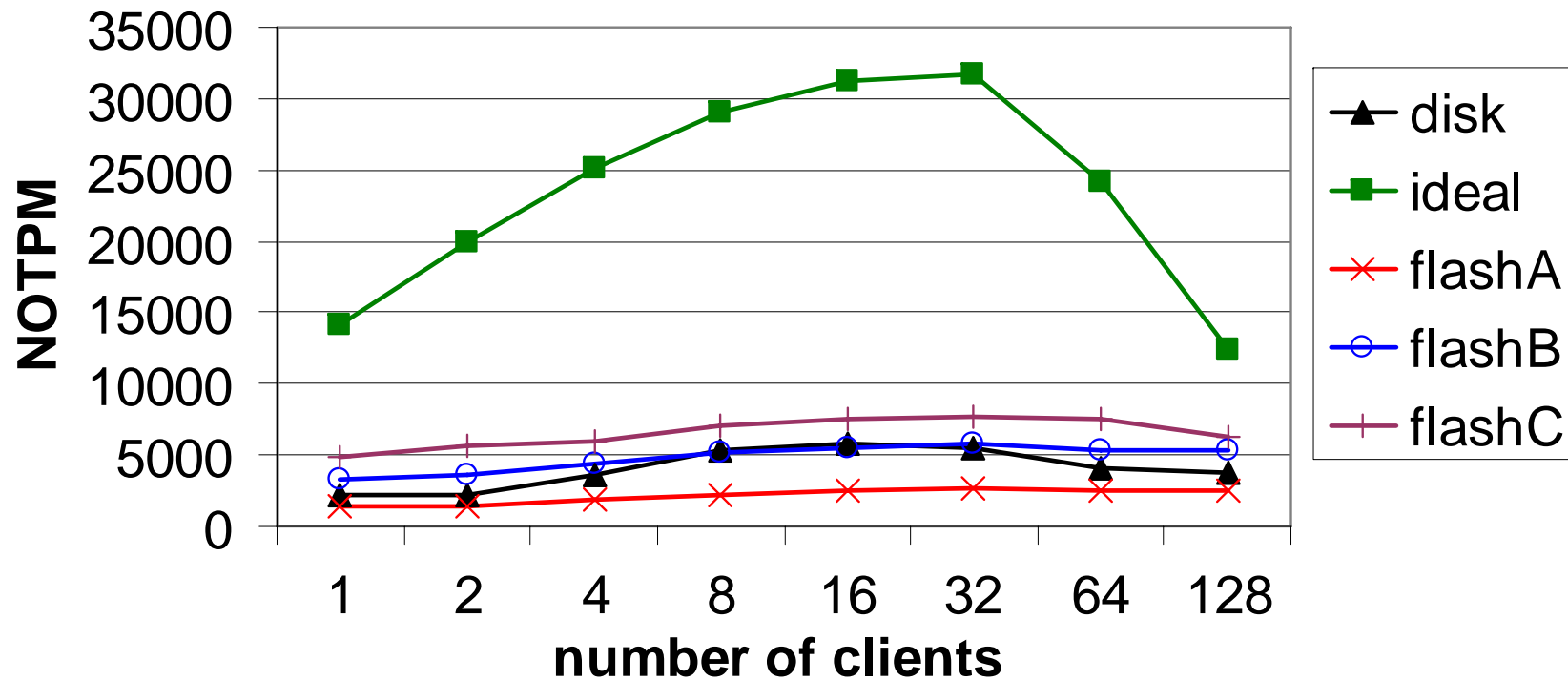
- Two Dell PowerEdge 1955 blade servers
 - Intel 5160 CPU, 3GHz, dual core, two threads per core
 - 4MB L2 cache, 4GB DRAM, 1Gbps Ethernet connection
 - Linux 2.6.17
- Logging device:
 - 10K rpm SAS disk
 - Three types of USB flash drives: flash-A, flash-B, flash-C
 - Up to two USB flash drives
 - SSD
 - Write cache is disabled unless for the “ideal” device
- TPCC:
 - MySQL server runs on one blade, 20 warehouses
 - Data + indices fit into main memory
 - TPCC driver (based on OSDL DBT2 0.39) repeatedly issue TPCC transaction requests with zero think time

Logging Performance: Disk-Based-Logging vs. Ideal



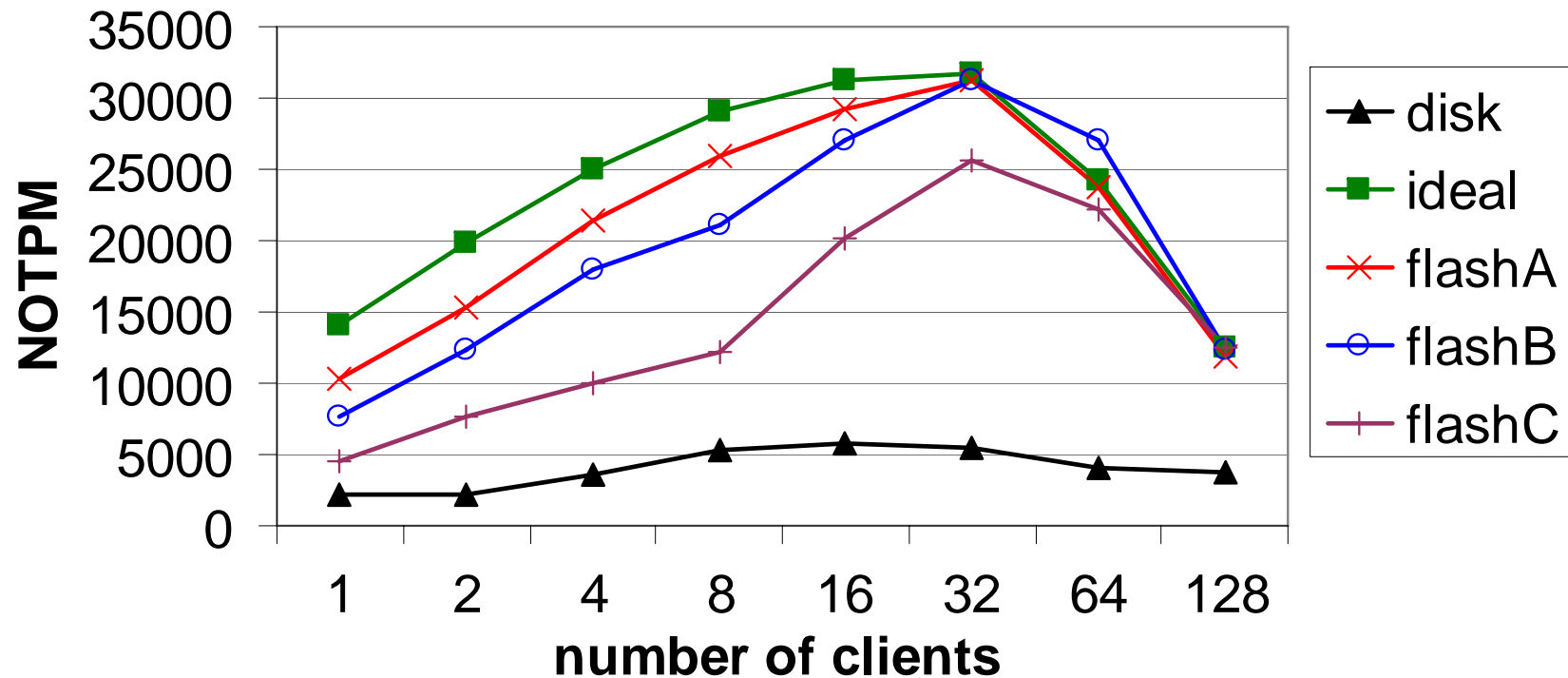
- **Disk-based-logging is a performance bottleneck.**

Logging Performance: Employing USB flash drives Naively



- Little improvements

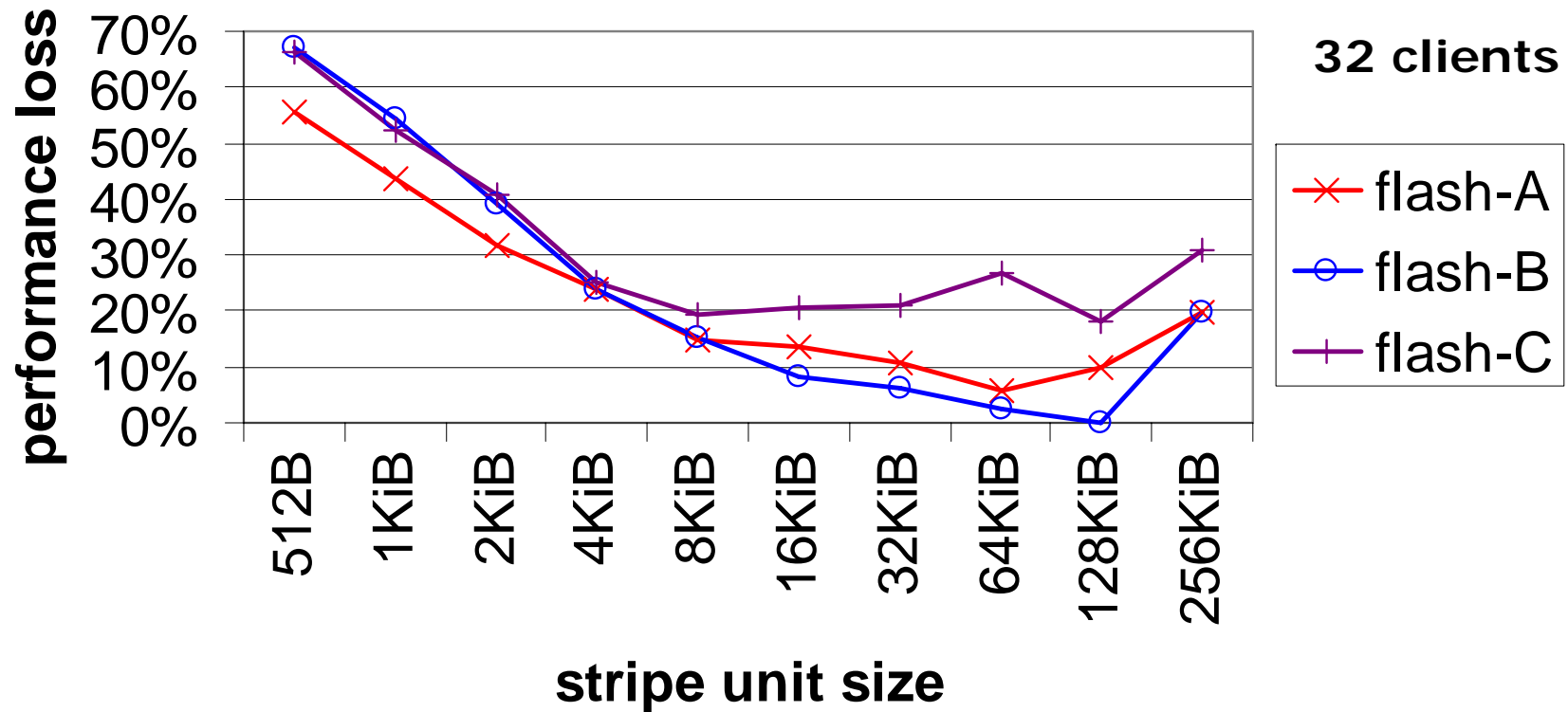
Logging Performance: FlashLogging with Best Configuration



A and B: 2 USB flash + 1 disk; C: 2 USB flash + 1 disk + outlier hiding

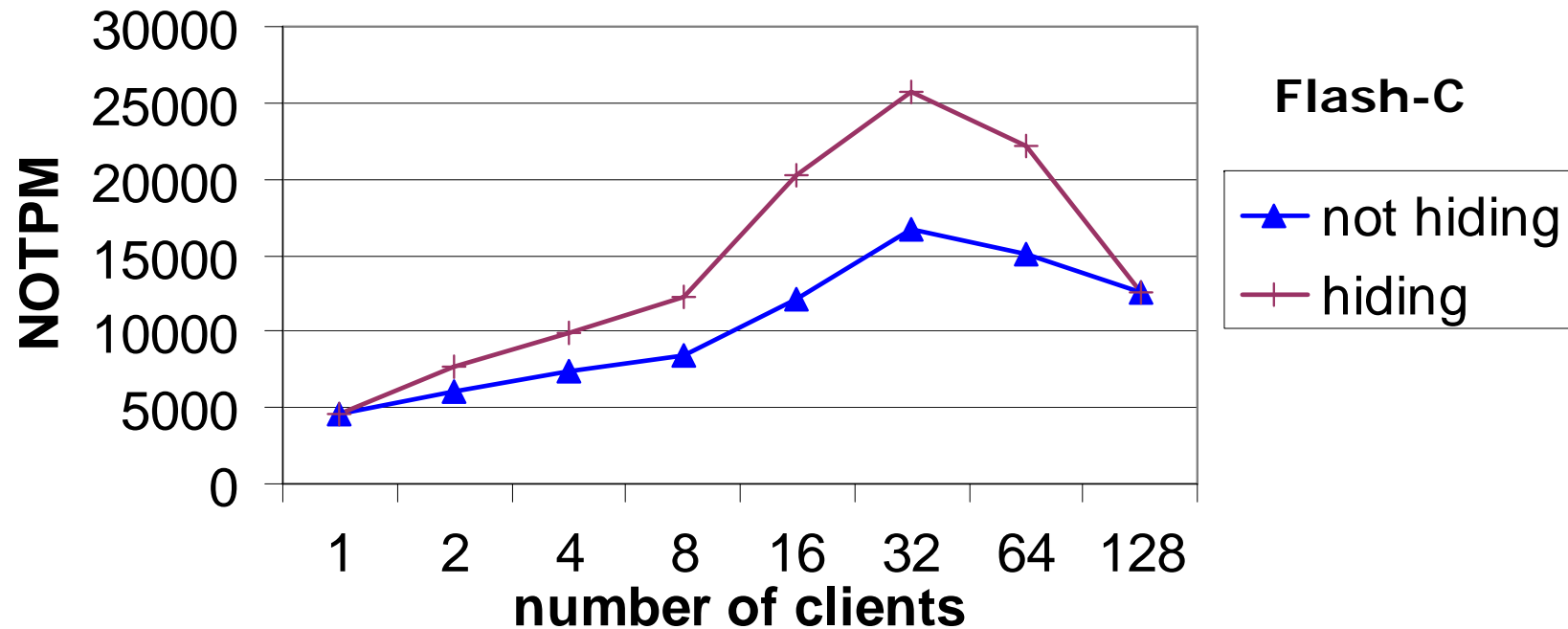
- Achieve up to 5.7X improvements over disk-based logging, 98.6% of the ideal performance

FlashLogging vs. Conventional Array



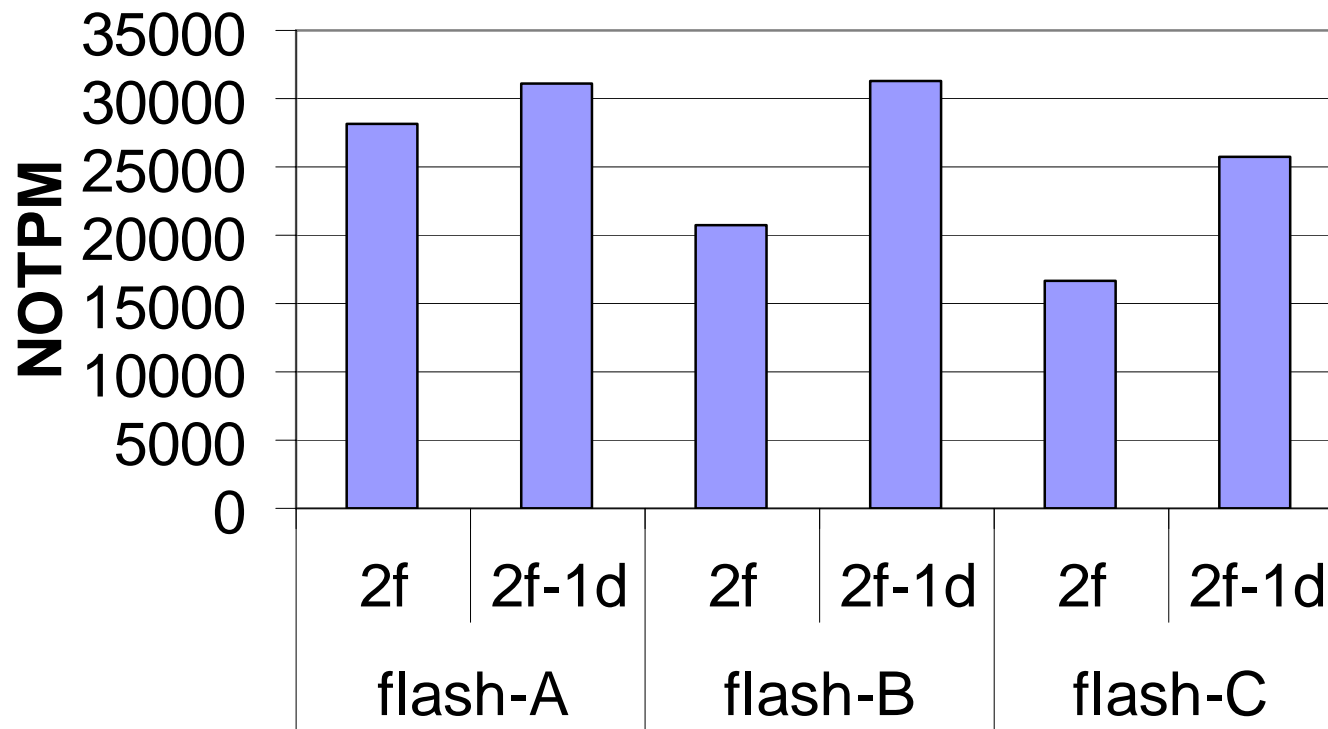
- Request splitting and skipping incur up to 67% of performance loss

Outlier Hiding



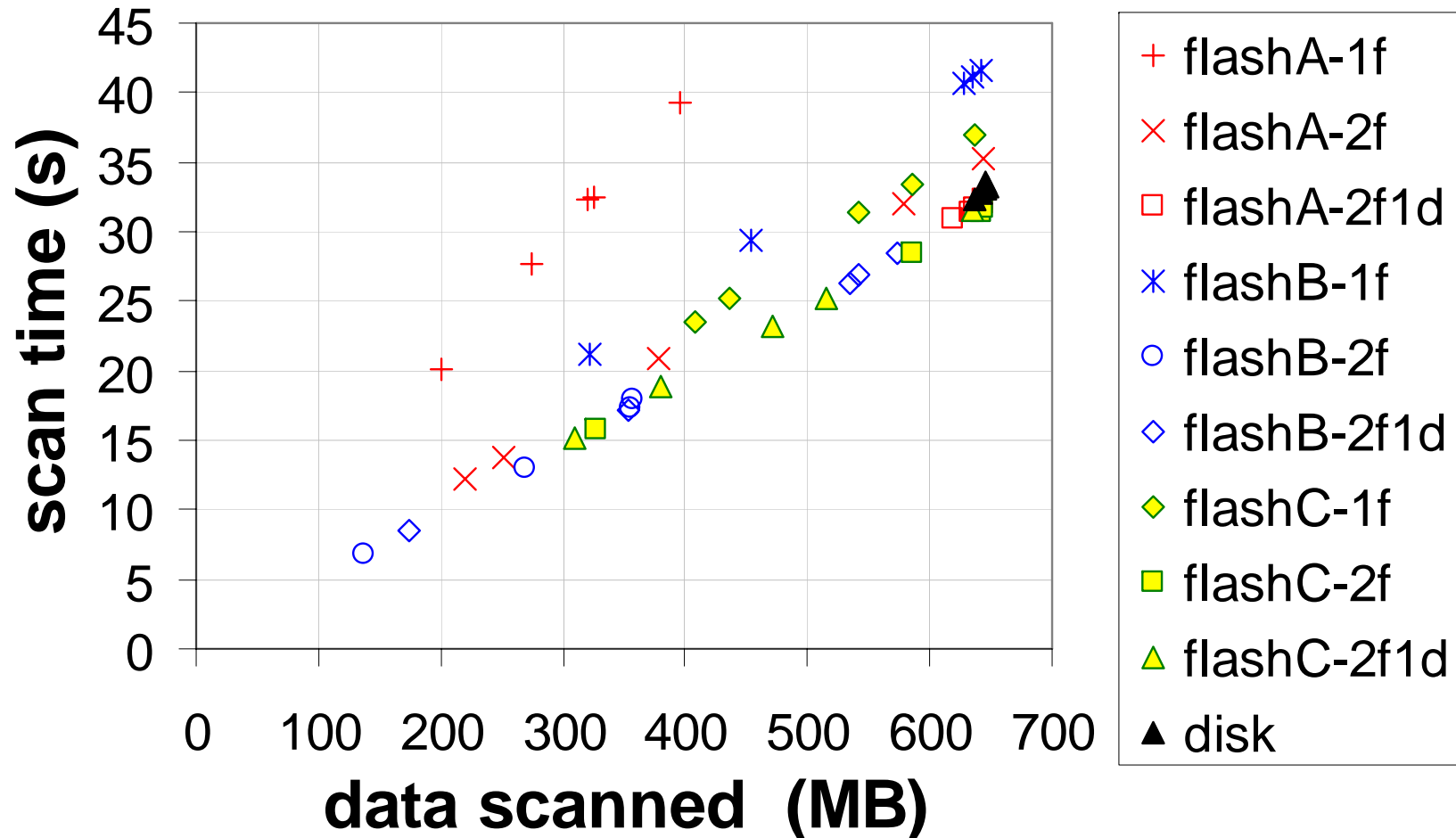
- Flash-C sees the longest outlier time
- Outlier hiding achieves up to 1.66X improvements

Near-Zero-Delay Archival Disk



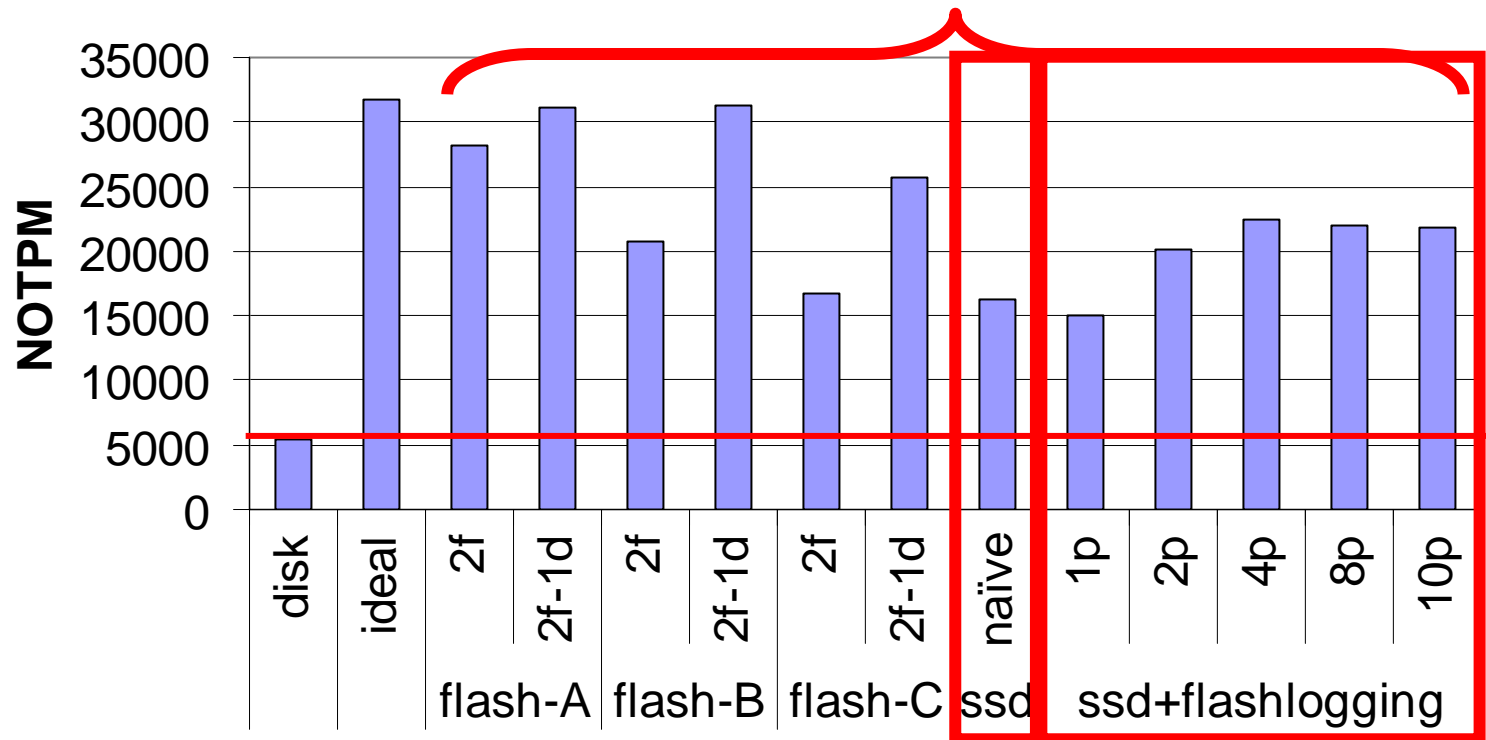
- Achieves significant improvements
- More significant for lower-performing USB flash drives

Recovery Performance



- 5 tests per configuration
- **Comparable performance with original**

Compared to SSD



- Using SSD naively: 2.98X over disk-based logging
- Exploit multiple parallel channels in SSDs: 4.12X over disk
 - Use K partitions as virtual devices with FlashLogging
- Multiple USB flash drives achieve similar performance with lower \$

Related Works

- Recent works exploiting flash devices for various aspects of data management systems
 - [Lee & Moon, SIGMOD'07]
 - [Ross, DaMoN'08], [Shar et al. DaMoN'08], [Lee et al., SIGMOD'08], [Koltsidas & Viglas, VLDB'08], [Nath & Gibbons, VLDB'08]
 - [Bouganim et al. CIDR'09], [Caulfield et al. ASPLOS'09], [Do & Patel, DaMoN'09], [Stoica et al. DaMoN'09], [Ou et al. DaMoN'09], [Tsirogiannis et al. SIGMOD'09]
- [Lee et al. SIGMOD'08]
 - Shows that SSDs (as drop-in replacement of HDDs) can significantly improve OLTP performance
 - Our paper focuses on transactional logging of OLTP and shows that a FlashLogging design with low-end flash devices can achieve good performance

Conclusions

- Flash devices provide an *efficient, inexpensive, and simple* solution to synchronous logging
- Can achieve good performance even with low-end, USB flash devices
- We propose and evaluate a FlashLogging design:
 - Unconventional array organization
 - Outlier detection and hiding
 - Efficient recovery processing
 - Near-zero-delay archival disk
- Exploiting parallelism is important
 - Multiple flash devices
 - Multiple channels in SSDs

Thank you!

shimin.chen@intel.com