# FlashVM:
# Virtual Memory Management on Flash

Mohit Saxena

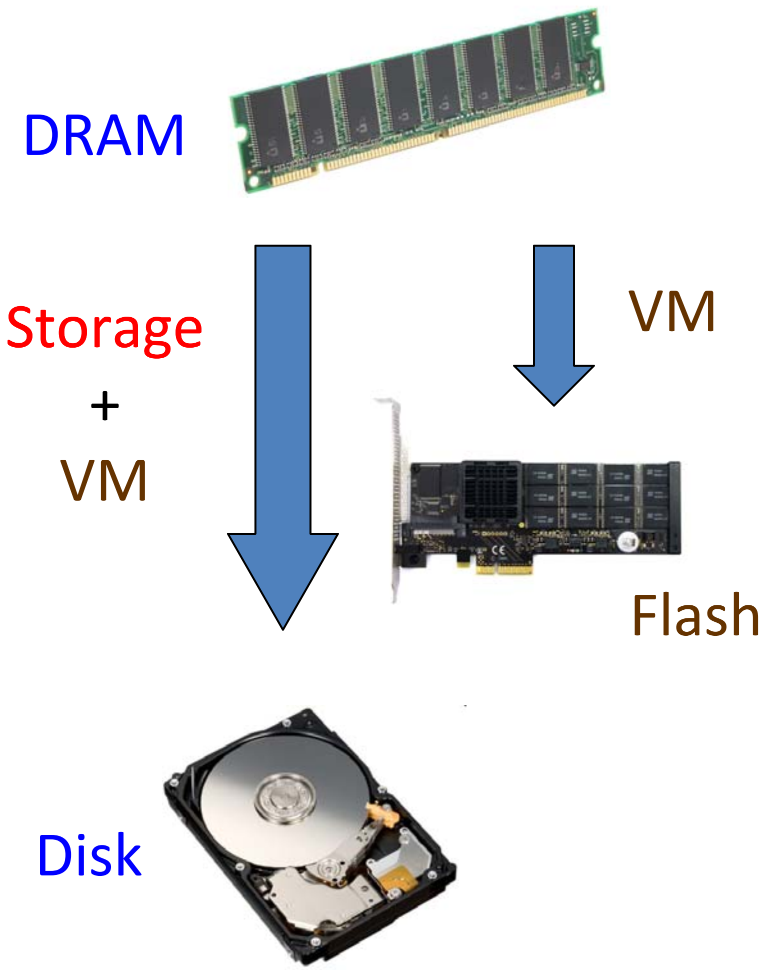
Michael M. Swift

University of Wisconsin-Madison

# Is Virtual Memory Relevant?

- There is never enough DRAM
  - Price, power and DIMM slots limit amount
  - Application memory footprints are ever-increasing
- VM is no longer DRAM+Disk
  - New memory technologies: Flash, PCM, Memristor ….

# Flash and Virtual Memory




DRAM is expensive

Disk is slow

Flash is cheap and fast

***Flash*** for ***Virtual Memory***

# In this talk

- Flash for Virtual Memory
  - Does it improve system price/performance?
  - What OS changes are required?
- FlashVM
  - System architecture using dedicated flash for VM
  - Extension to core VM subsystem in the Linux kernel
  - Improved performance, reliability and garbage collection

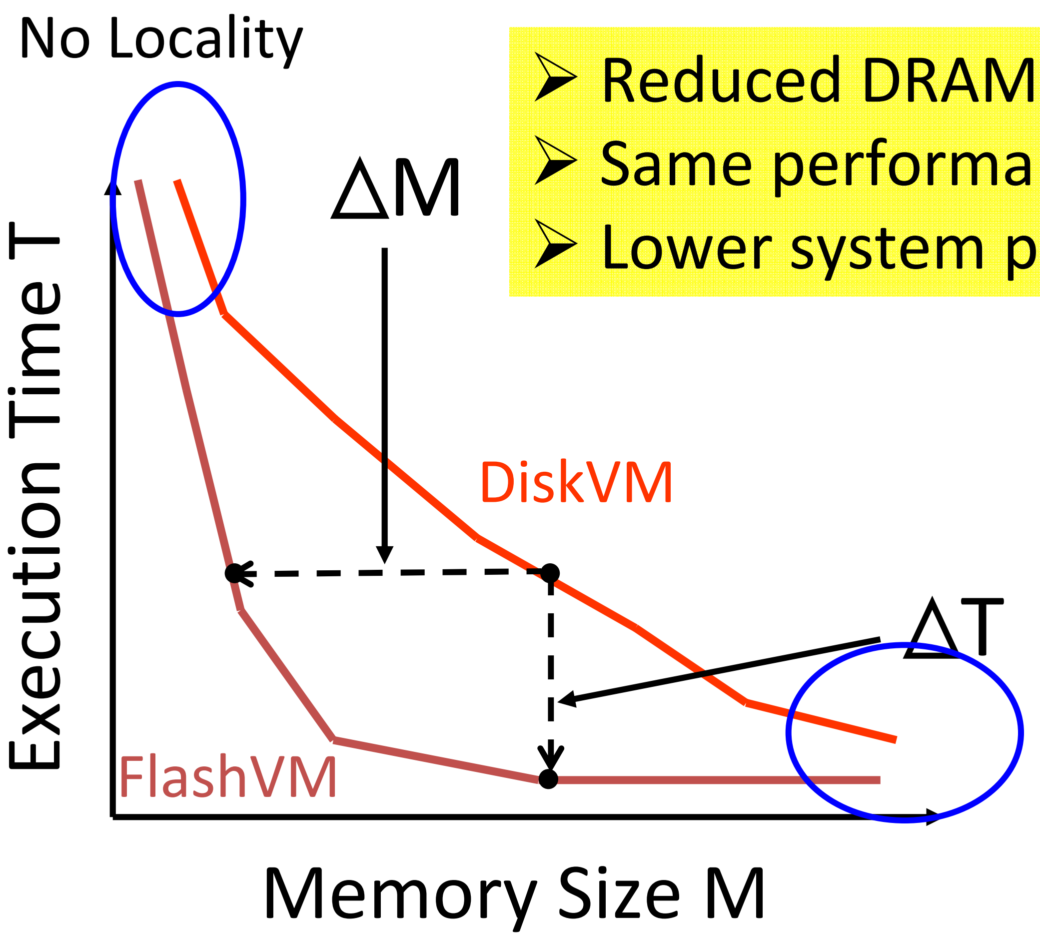
# Outline

- Introduction
- Background
  - **Flash and VM**
- Design
- Evaluation
- Conclusions

# Flash 101

- Flash is **not disk**
  - Faster random access performance: 0.1 vs. 2-3 ms for disk
  - No in-place modify: write only to erased location
- Flash blocks **wear out**
  - Erasures limited to 10,000-100,000 per block
  - Reliability dropping with increasing MLC flash density
- Flash devices **age**
  - Log-structured writes leave few clean blocks after extensive use
  - Performance drops by up to 85% on some SSDs
  - Requires garbage collection of free blocks

# Virtual Memory 101



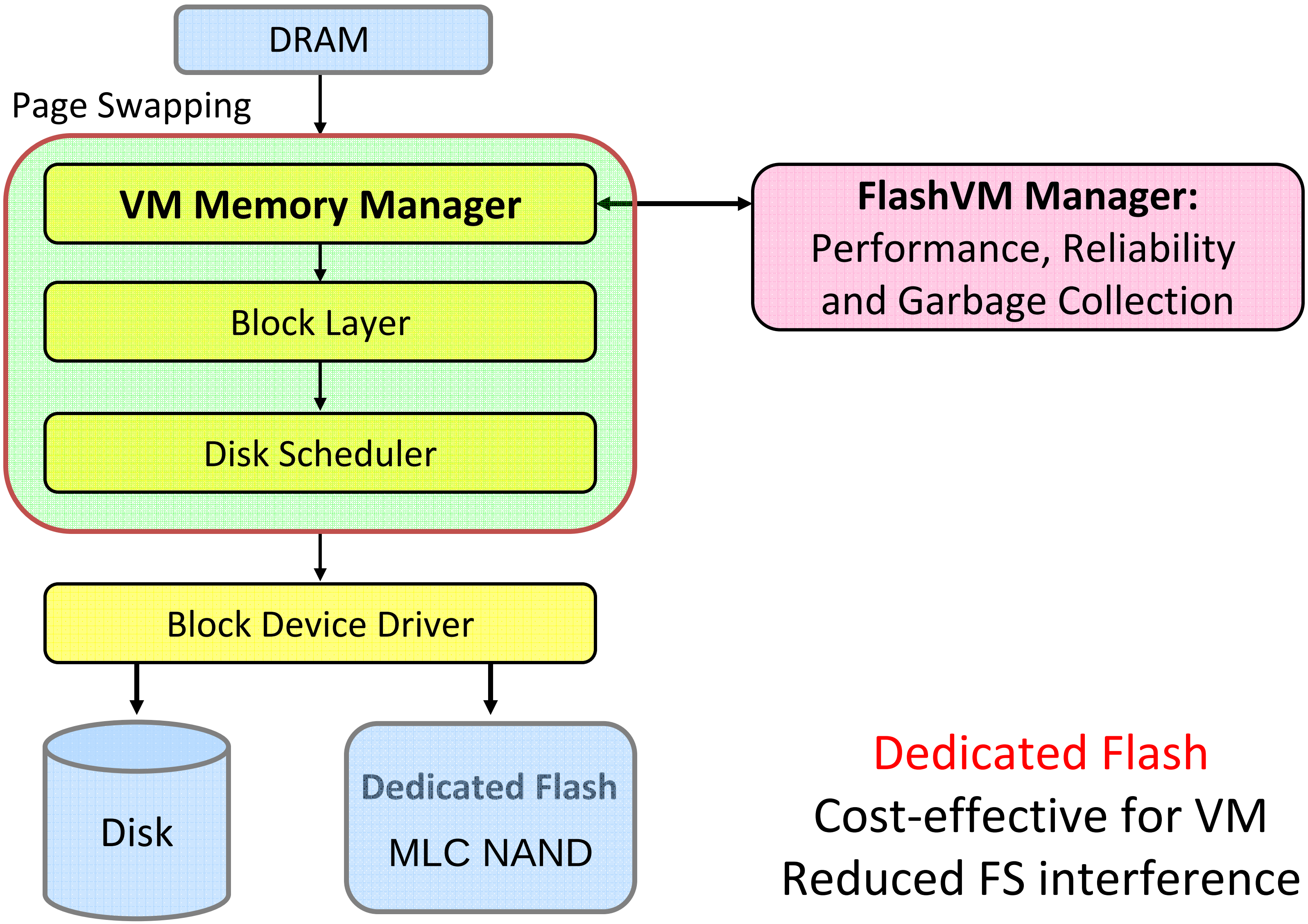

- Reduced DRAM
- Same performance
- Lower system price

- Faster execution
- No additional DRAM
- Similar system price

# Outline

- Introduction
- Background
- **Design**
  - Performance
  - Reliability
  - Garbage Collection
- Evaluation
- Conclusions

# FlashVM Hierarchy



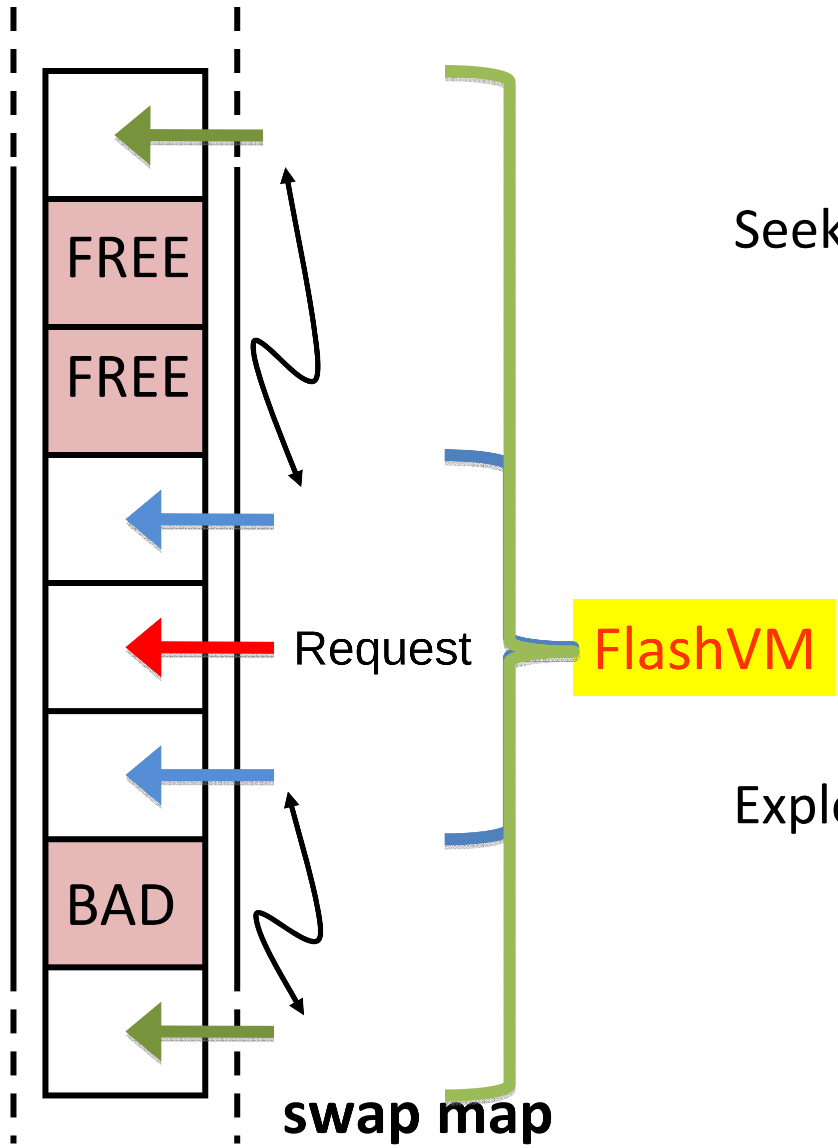

Dedicated Flash
Cost-effective for VM
Reduced FS interference

# VM Performance

- Challenge
  - VM systems optimized for *disk performance*
  - Slow random reads, high access and seek costs, symmetrical read/write performance
- FlashVM de-diskifies VM:
  - Page write back
  - Page scanning
  - Disk scheduling
  - Page prefetching

Parameter Tuning

# Page Prefetching




VM assumption

Seek and rotational delays are longer than the transfer cost of extra blocks

Linux sequential prefetching

Minimize costly disk seeks

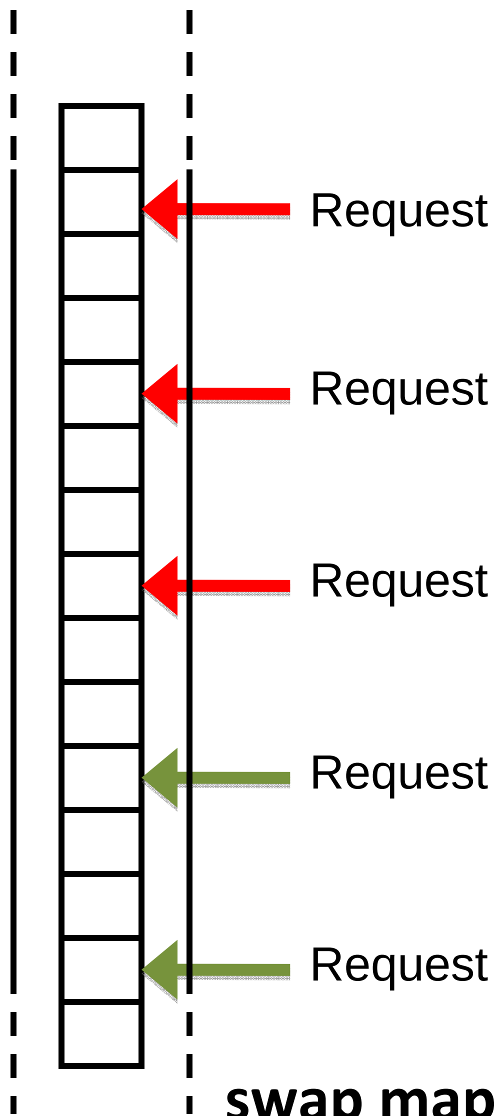
Delimited by free and bad blocks

FlashVM prefetching

Exploit fast flash random reads and spatial locality in reference pattern

Seek over free and bad blocks

# Stride Prefetching

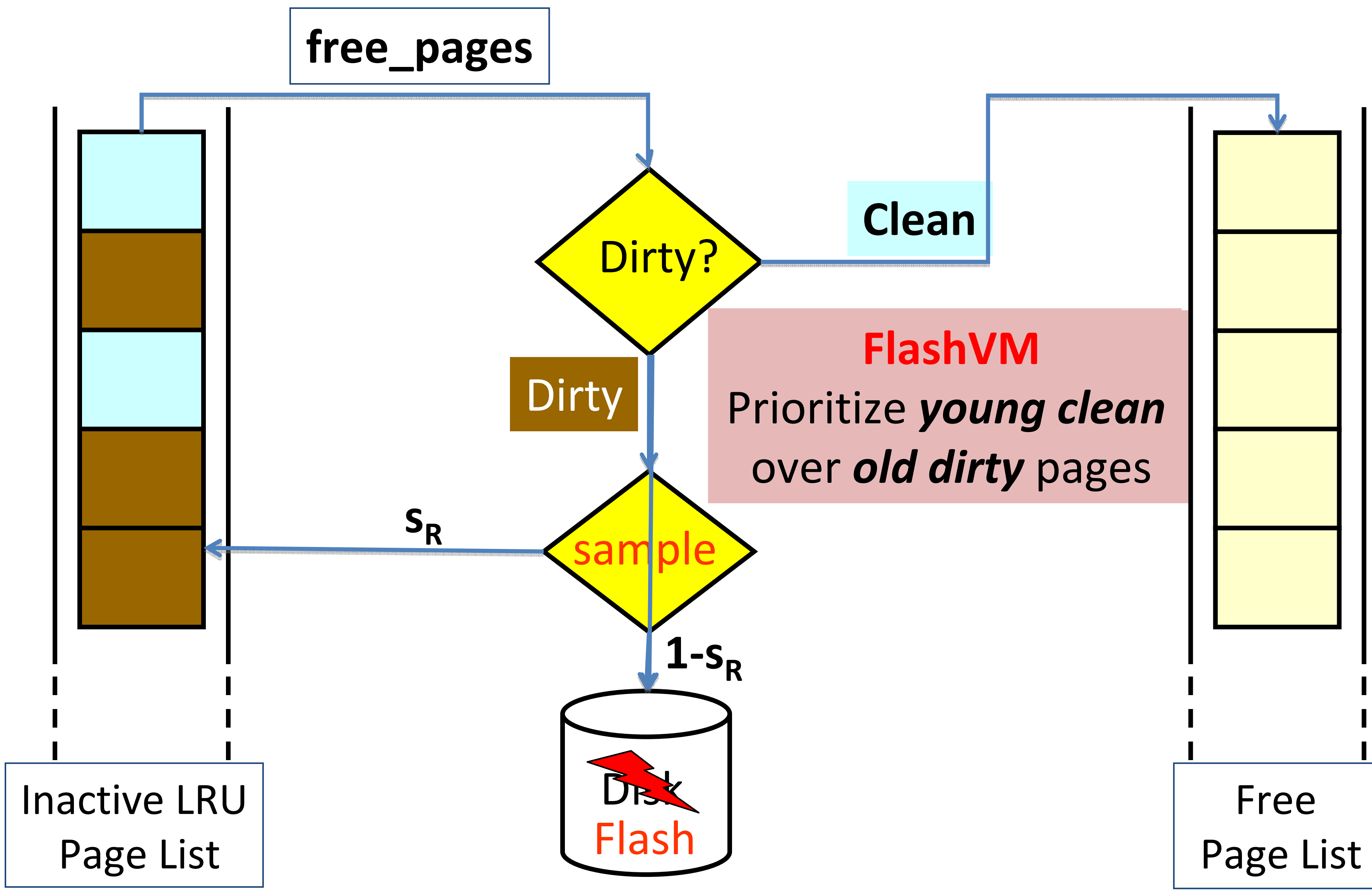



- FlashVM uses stride prefetching
  - Exploit temporal locality in the reference pattern
  - Exploit cheap seeks for fast random access
  - Fetch two extra blocks in the stride

# The Reliability Problem

- Challenge: Reduce the number of writes
  - Flash chips lose durability after 10,000 – 100,000 writes
  - Actual write-lifetime can be two orders of magnitude less
  - Past solutions:
    - Disk-based write caches for streamed I/O
    - De-duplication and compression for storage
- FlashVM uses knowledge of page *content* and *state*
  - Dirty Page sampling
  - Zero Page sharing

# Page Sampling



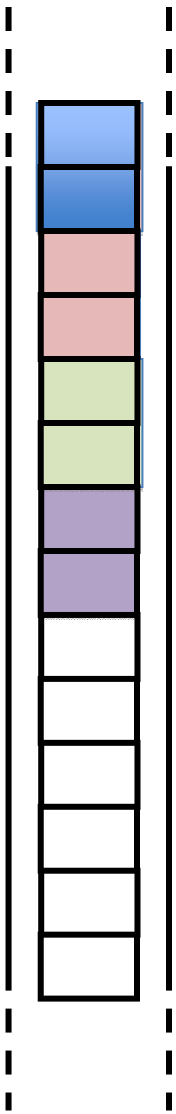
# Adaptive Sampling

- Challenge: Reference pattern variations
  - Write-mostly: Many dirty pages
  - Read-mostly: Many clean pages
- FlashVM adapts sampling rate
  - Maintain a moving average for the write rate
  - Low write rate → Increase $s_R$
    - Aggressively skip dirty pages
  - High write rate → Converge to native Linux
    - Evict dirty pages to relieve memory pressure

# Outline

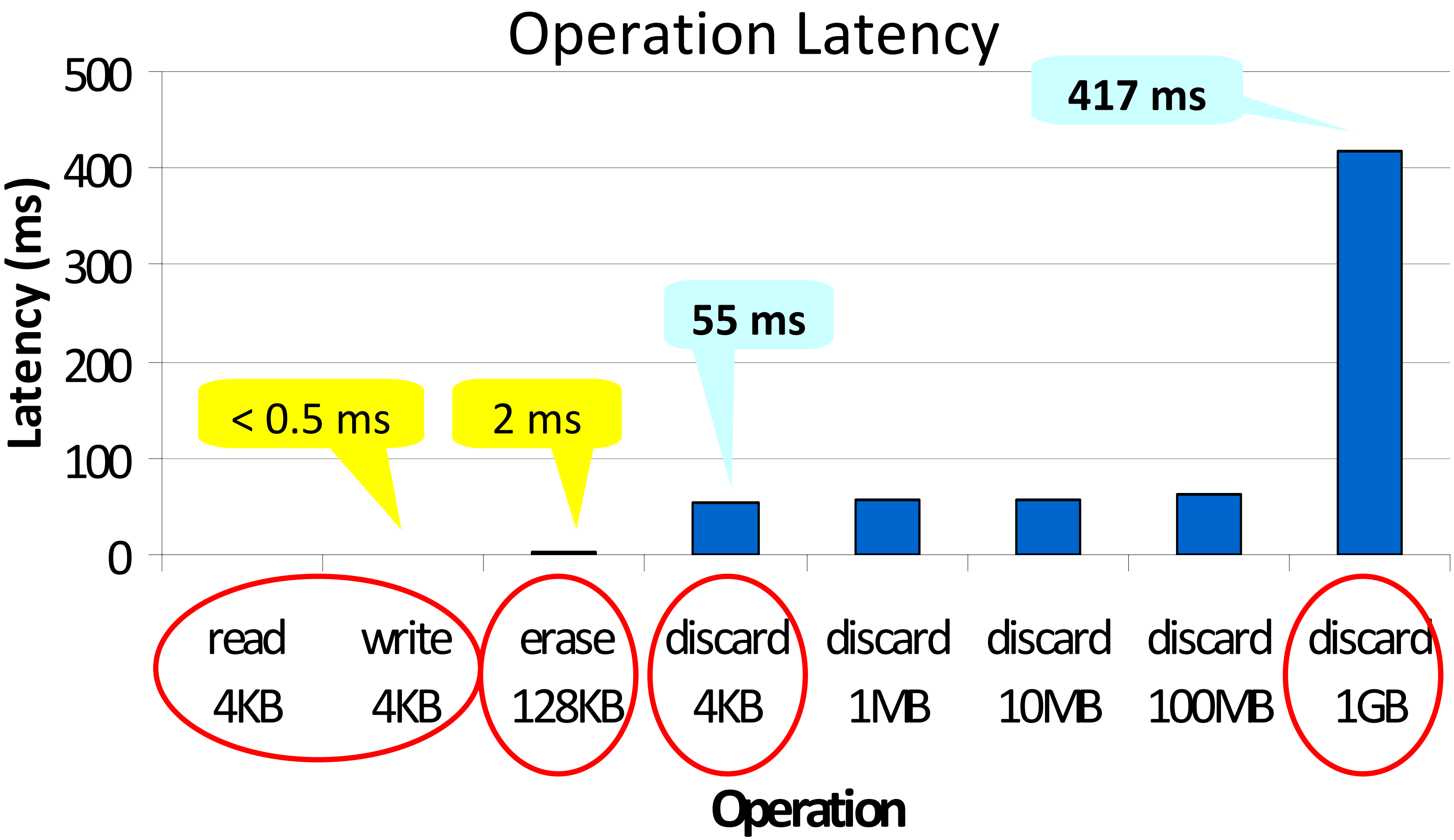
- Introduction
- Why FlashVM?
- Design
  - Performance
  - Reliability
  - **Garbage Collection**
- Evaluation
- Conclusions

# Flash Cleaning




- All writes to flash go to a new location
- Discard command notifies SSD that blocks are unused
- Benefits:
  - More free blocks for writing
  - Avoids copying data for partial over-writes

# Discard is Expensive

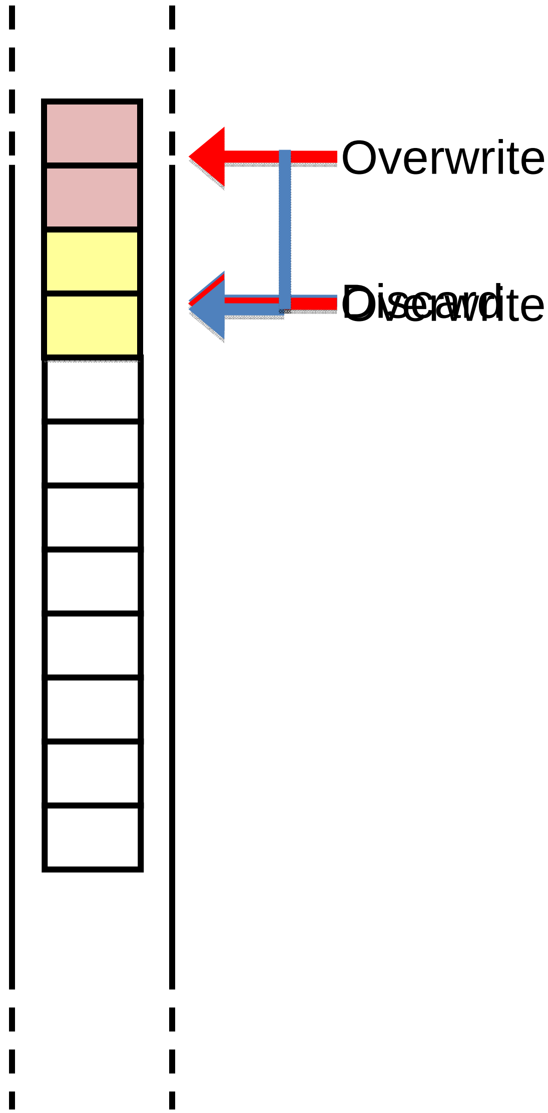



OCZ-Vertex, Indilinx controller

# Discard and VM

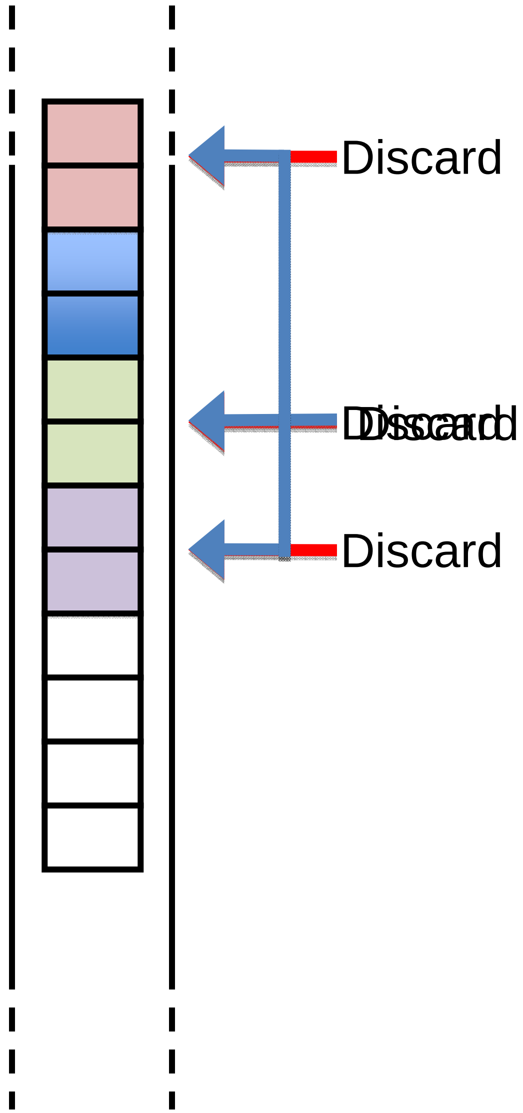
- Native Linux VM has limited discard support
  - Invokes discard before reusing free page clusters
  - Pays high fixed cost for small sets of pages
- FlashVM optimizes to reduce discard cost
  - Avoid unnecessary discards: dummy discard
  - Discard larger sizes to amortize cost: merged discard

# Dummy Discard




- Observation: Overwriting a block
  - notifies SSD it is empty
  - after discarding it, uses the free space made available by discard
- FlashVM implements *dummy discard*
  - Monitors rate of allocation
  - Virtualize discard by reusing blocks likely to be overwritten soon

# Merged Discard




- Native Linux invokes discard once per *page cluster*
  - Result: 55 ms latency for freeing 32 pages (128K)
- FlashVM batch many free pages
  - Defer discard until 100 MB of free pages available
  - Pages discarded may be non-contiguous

# Design Summary

- Performance improvements
  - Parameter Tuning: page write back, page scanning, disk scheduling
  - Improved/stride prefetching
- Reliability improvements
  - Reduced writes: page sampling and sharing
- Garbage collection improvements
  - Merged and Dummy discard

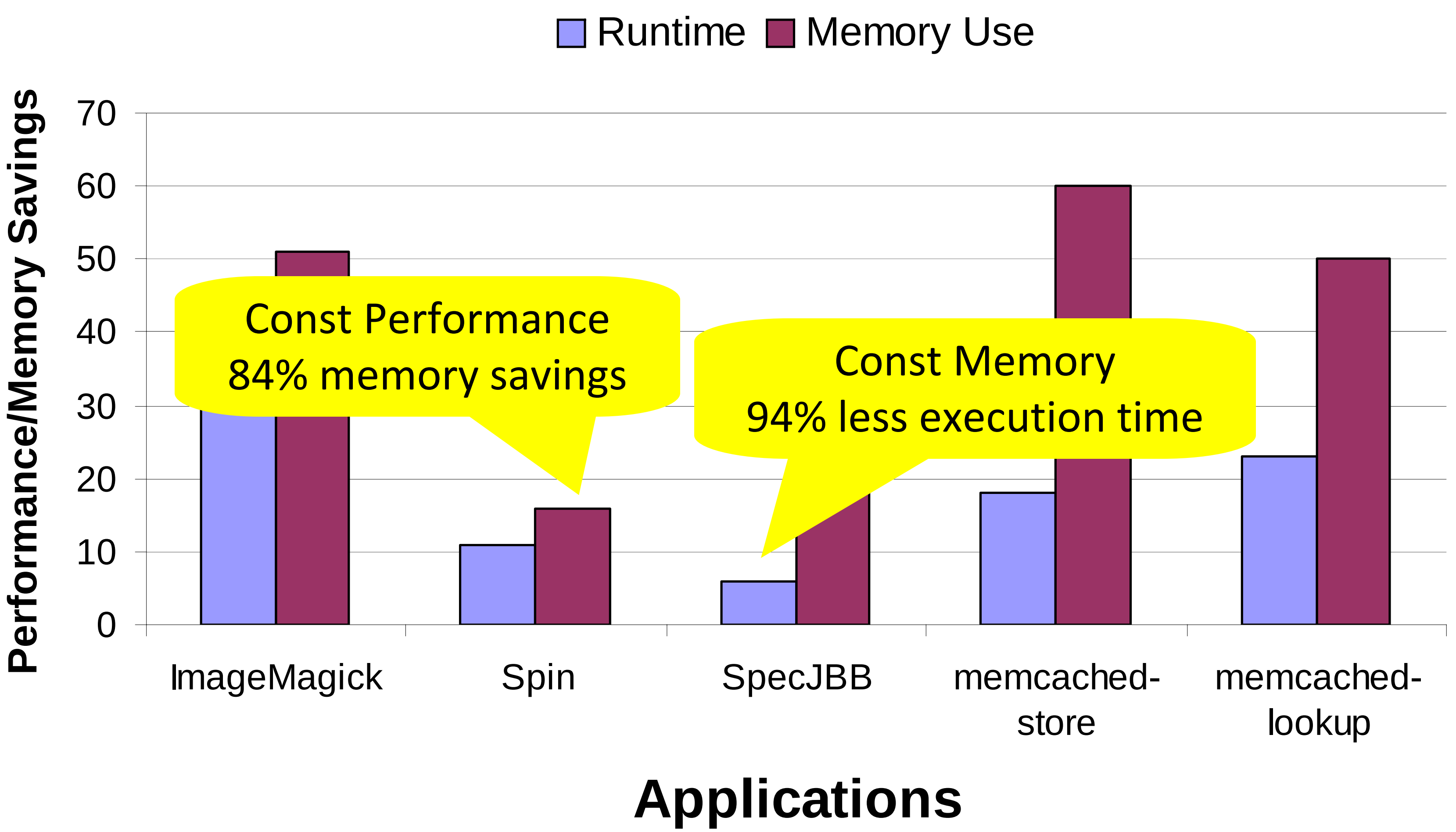
# Outline

- Introduction
- Motivation
- Design
- **Evaluation**
  – Performance and memory savings
  – Reliability and garbage collection
- Conclusions

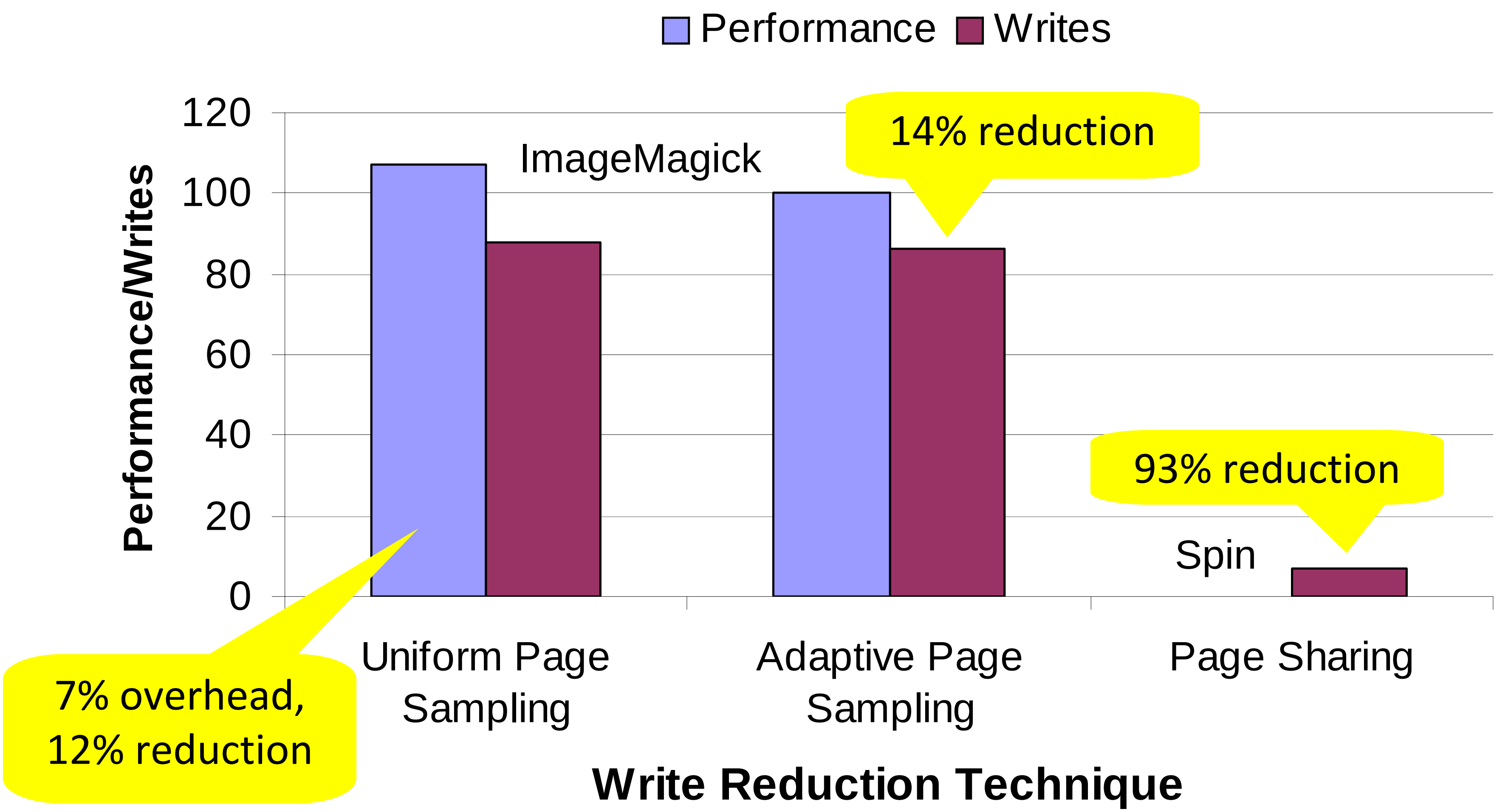
# Methodology

- System and Devices
  - 2.5 GHz Intel Core 2 Quad, Linux 2.6.28 kernel
  - IBM, Intel X-25M, OCZ-Vertex trim-capable SSDs
- Application Workloads
  - ImageMagick - resizing a large JPEG image by 500%
  - Spin – model checking for 10 million states
  - SpecJBB – 16 concurrent warehouses
  - memcached server – key-value store for 1 million keys

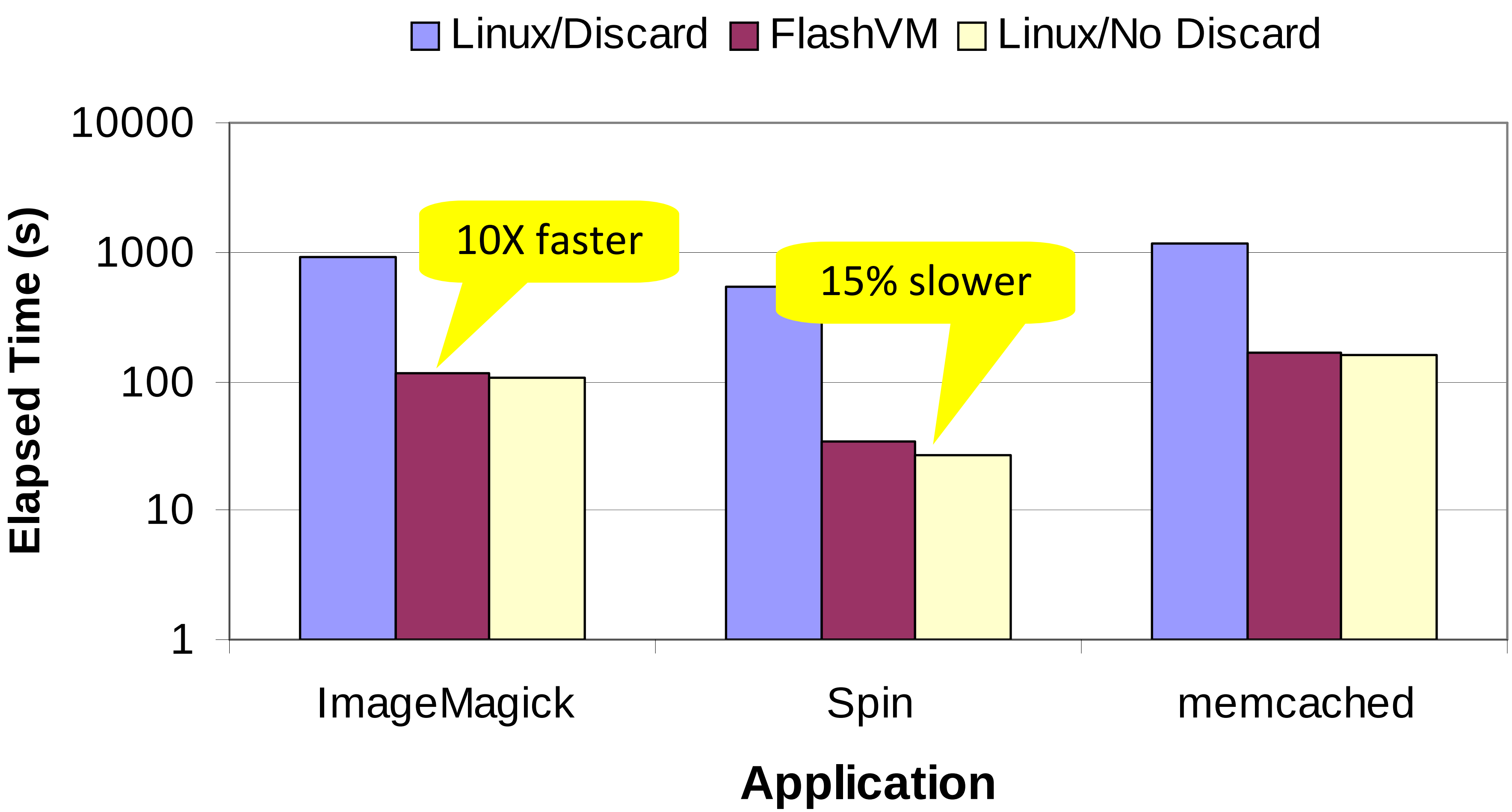
# Application Performance and Memory Savings



# Write Reduction



# Garbage Collection



# Conclusions

- FlashVM: Virtual Memory Management on Flash
  - Dedicated flash for paging
  - Improved performance, reliability and garbage collection
- More opportunities and challenges for OS design
  - Scaling FlashVM to massive memory capacities (terabytes!)
  - Future memory technologies: PCM and Memristors

# Thanks!

FlashVM: Virtual Memory Management on Flash


Mohit Saxena
Michael M. Swift

University of Wisconsin-Madison
*http://pages.cs.wisc.edu/~msaxena/FlashVM.html*