# Page Overlays

## An Enhanced Virtual Memory Framework to Enable Fine-grained Memory Management

**Vivek Seshadri**

Gennady Pekhimenko, Olatunji Ruwase,

Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch,

Todd C. Mowry, Trishul Chilimbi

SAFARI @CMU

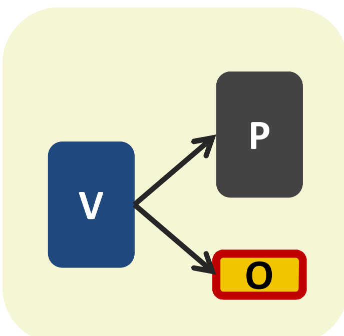
Carnegie Mellon University

intel

Microsoft Research

# Executive Summary

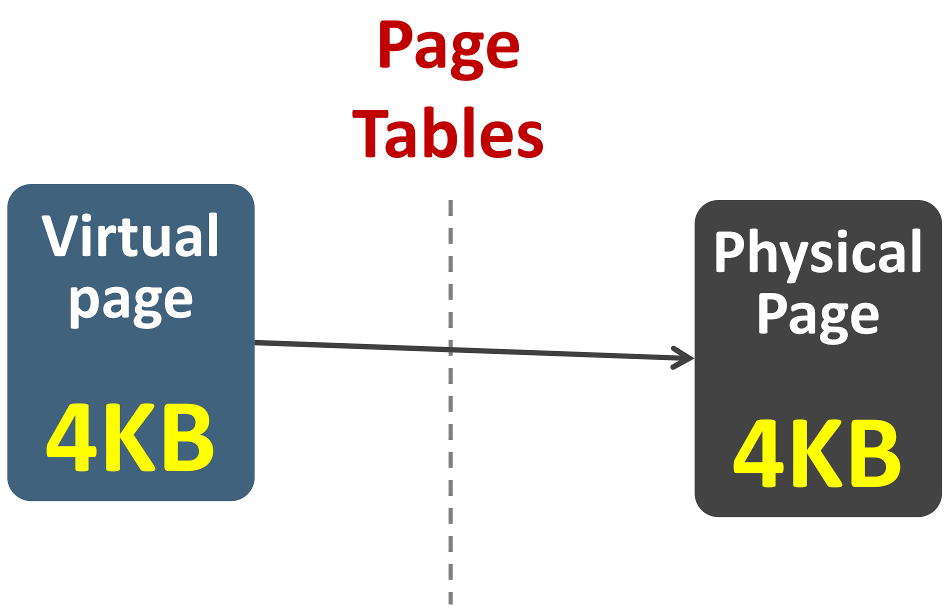
- Sub-page memory management has several applications
  - More efficient capacity management, protection, metadata, …
- Page-granularity virtual memory → inefficient implementations
  - Low performance and high memory redundancy



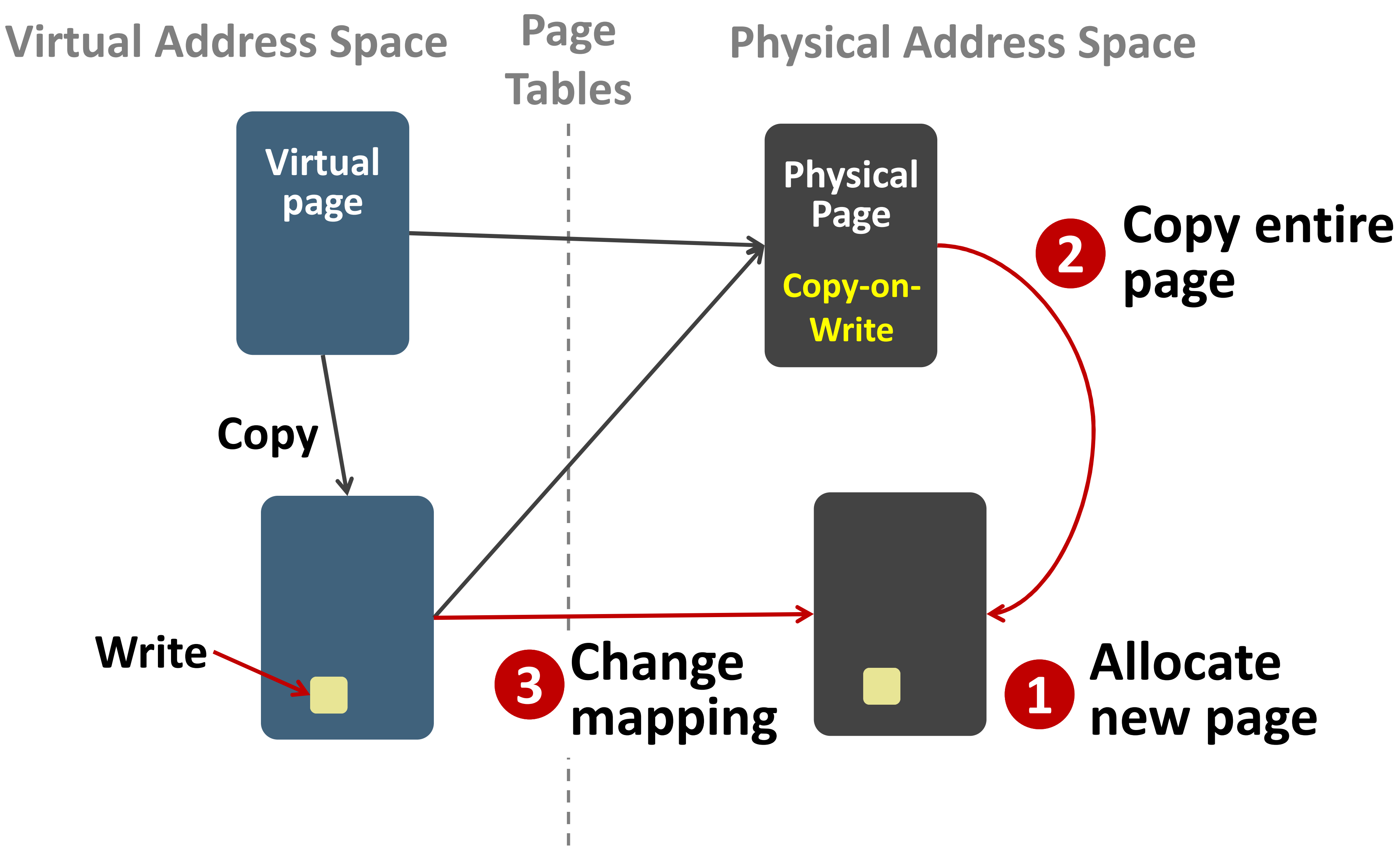

- Page Overlays: New Virtual Memory Framework
- Virtual Page → (physical page, overlay)
  - Overlay contains new versions of subset of cache lines
  - Efficiently store pages with mostly similar data
- Largely retains existing virtual memory structure
  - Low cost implementation over existing frameworks
- Powerful access semantics – Enables many applications
  - E.g., overlay-on-write, efficient sparse data structure representation
- Improves performance and reduces memory redundancy

# Existing Virtual Memory Systems

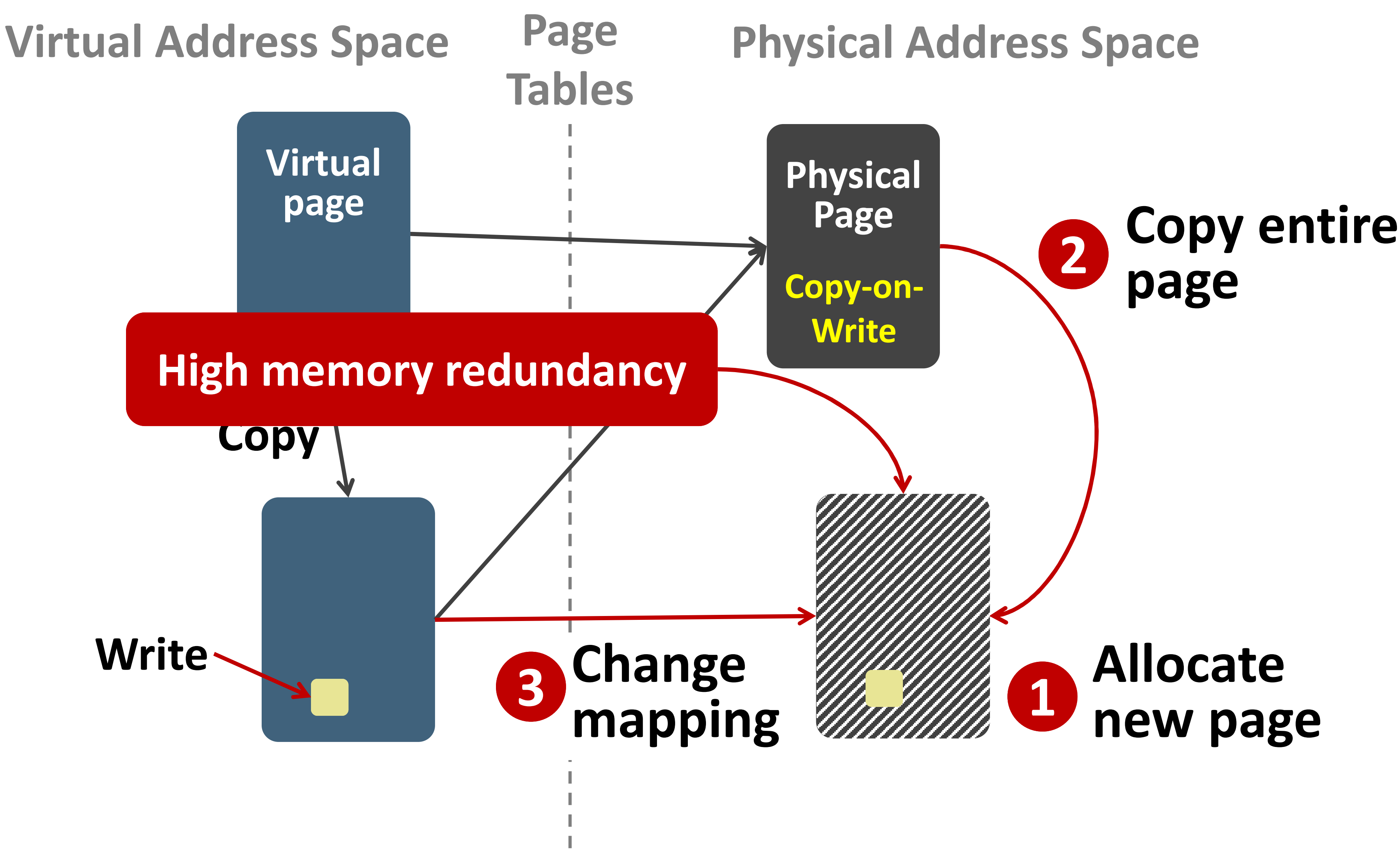
- Virtual memory enables many OS functionalities
  - Flexible capacity management
  - Inter-process data protection, sharing
  - Copy-on-write, page flipping



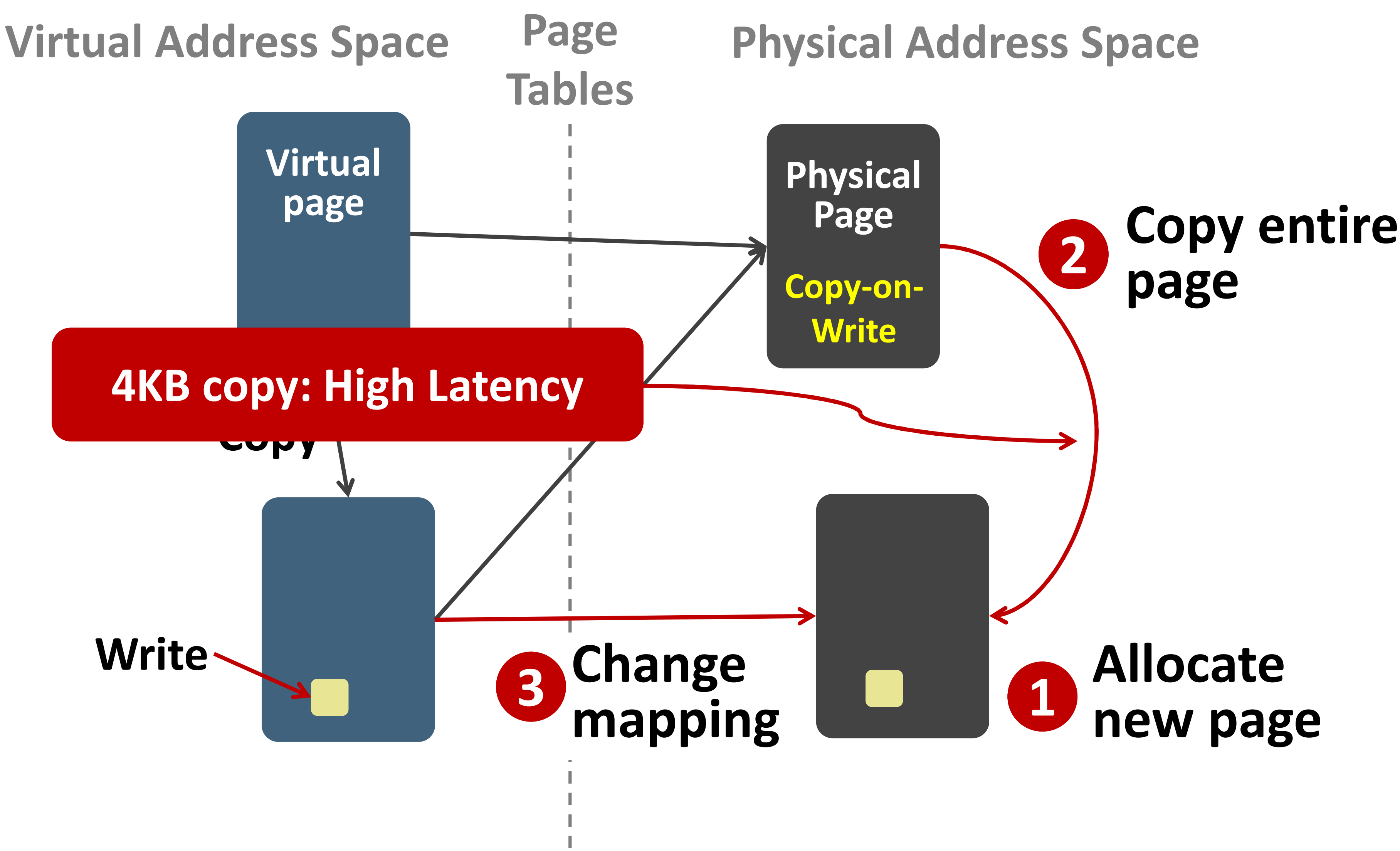
# Case Study: Copy-on-Write



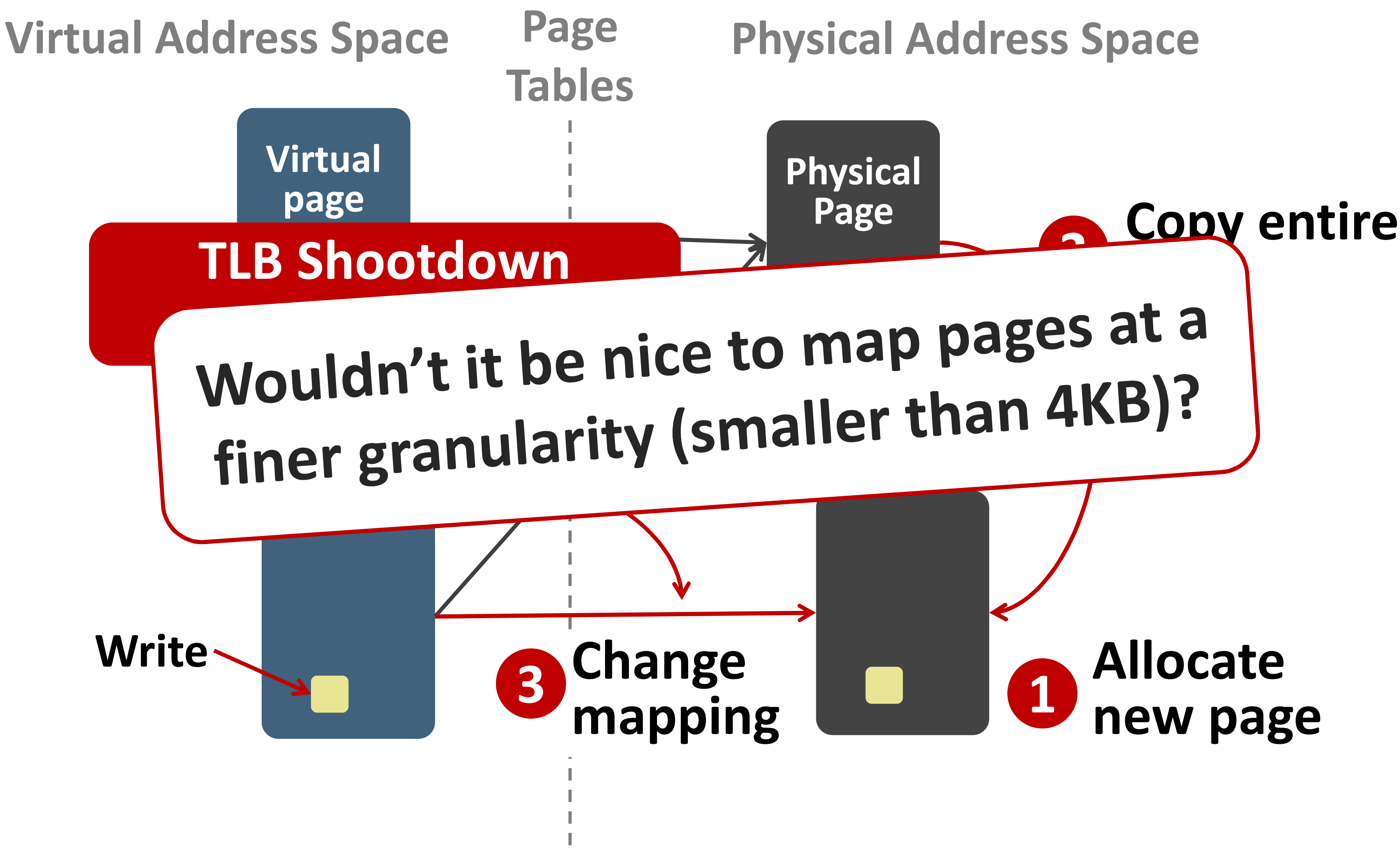
# Shortcomings of Page-granularity Management



# Shortcomings of Page-granularity Management



# Shortcomings of Page-granularity Management




Wouldn't it be nice to map pages at a finer granularity (smaller than 4KB)?

# Fine-grained Memory Management

**Higher Performance**
(e.g., more efficient copy-on-write)

**Fine-grained data protection**
(simpler programs)

**Fine-grained Memory Management**

**More efficient capacity management**
(avoid internal fragmentation, deduplication)

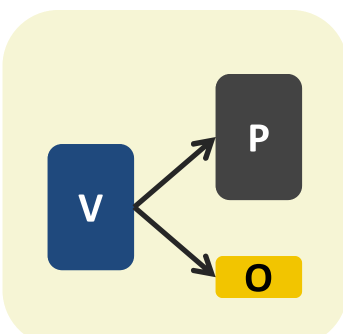
**Fine-grained metadata management**
(better security, efficient software debugging)

# Goal: Efficient Fine-grained Memory Management

## Existing Virtual Memory Framework

- Low performance
- High memory redundancy

## New Virtual Memory Framework

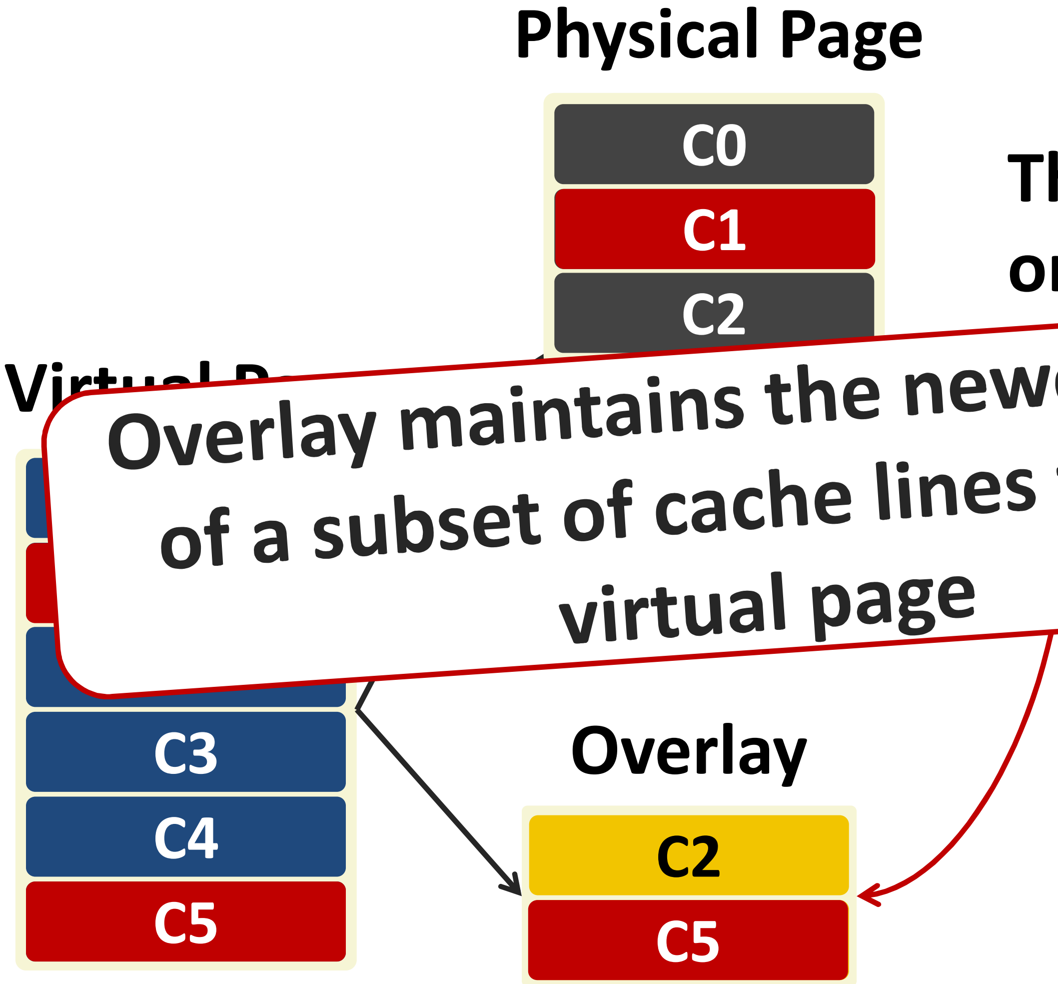



- Enable efficient fine-grained management
- Low implementation cost

# Outline

- Shortcomings of Existing Framework
- Page Overlays – Overview
- Implementation
  - Challenges and solutions
- Applications and Evaluation
- Conclusion

# The Page Overlay Framework

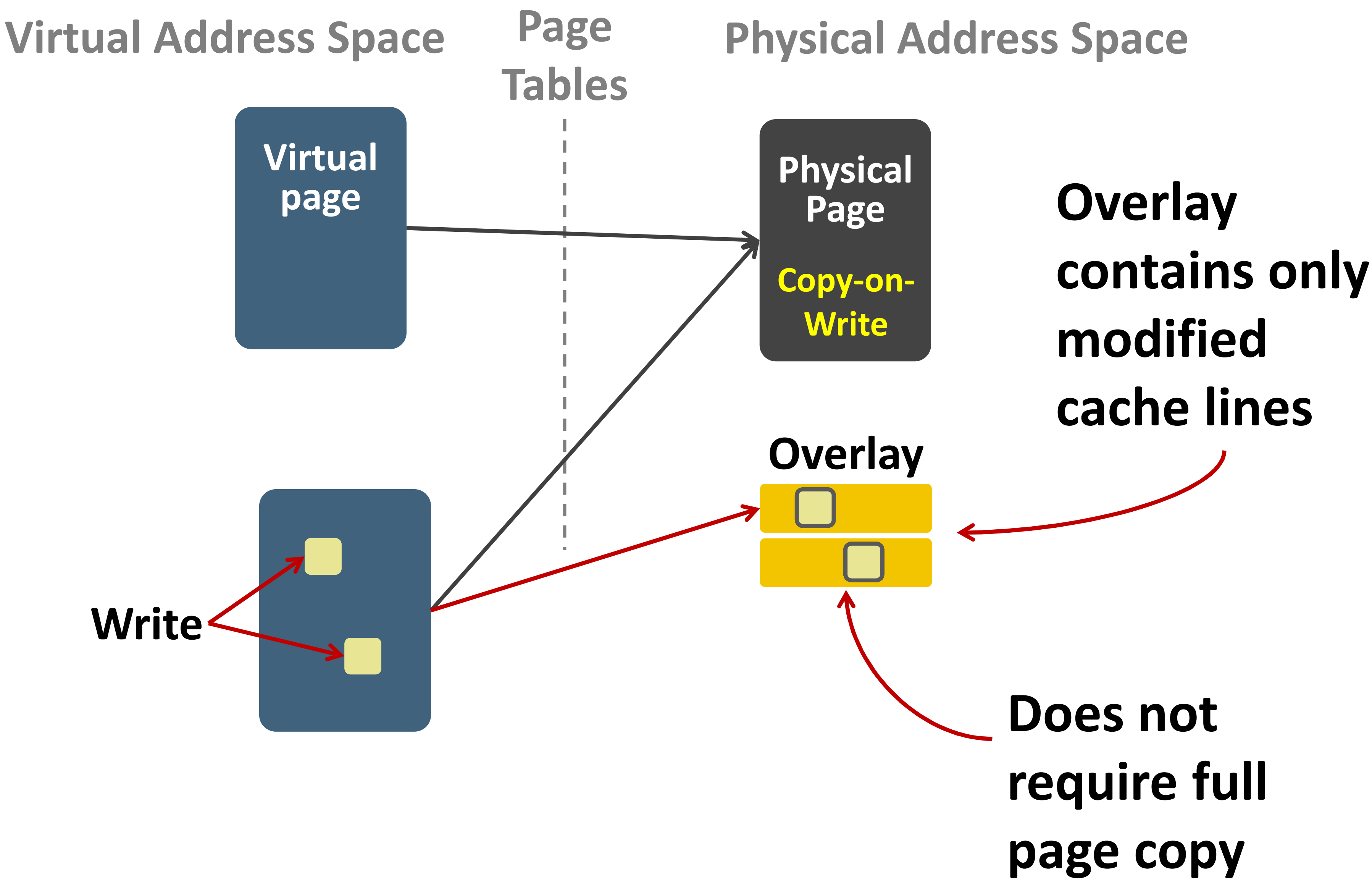



The overlay contains only a subset of cache ... virtual

**Overlay maintains the newer version of a subset of cache lines from the virtual page**

**Access Semantics:** Only cache lines not present in the overlay are accessed from the physical page

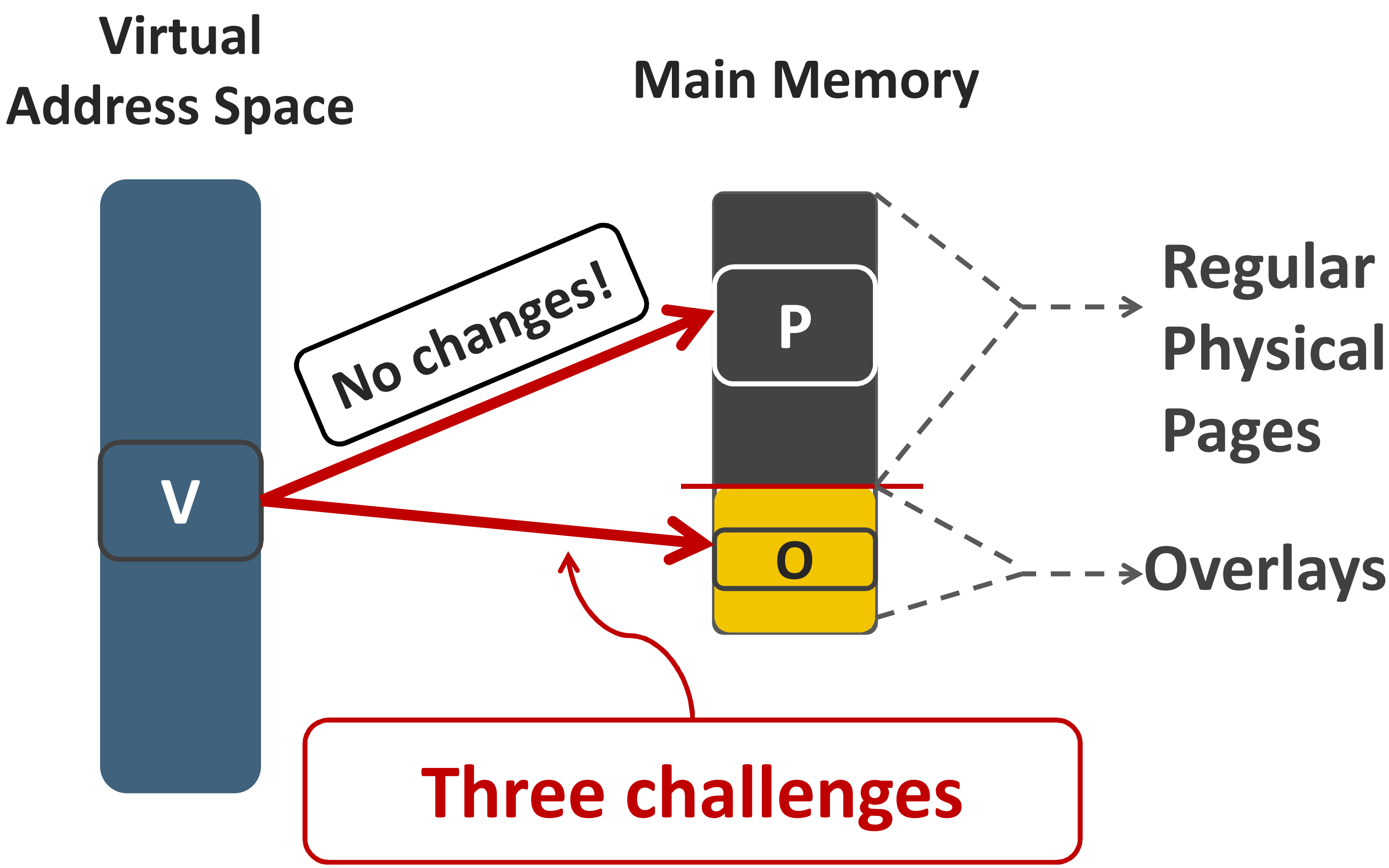
# Overlay-on-Write: An Efficient Copy-on-Write



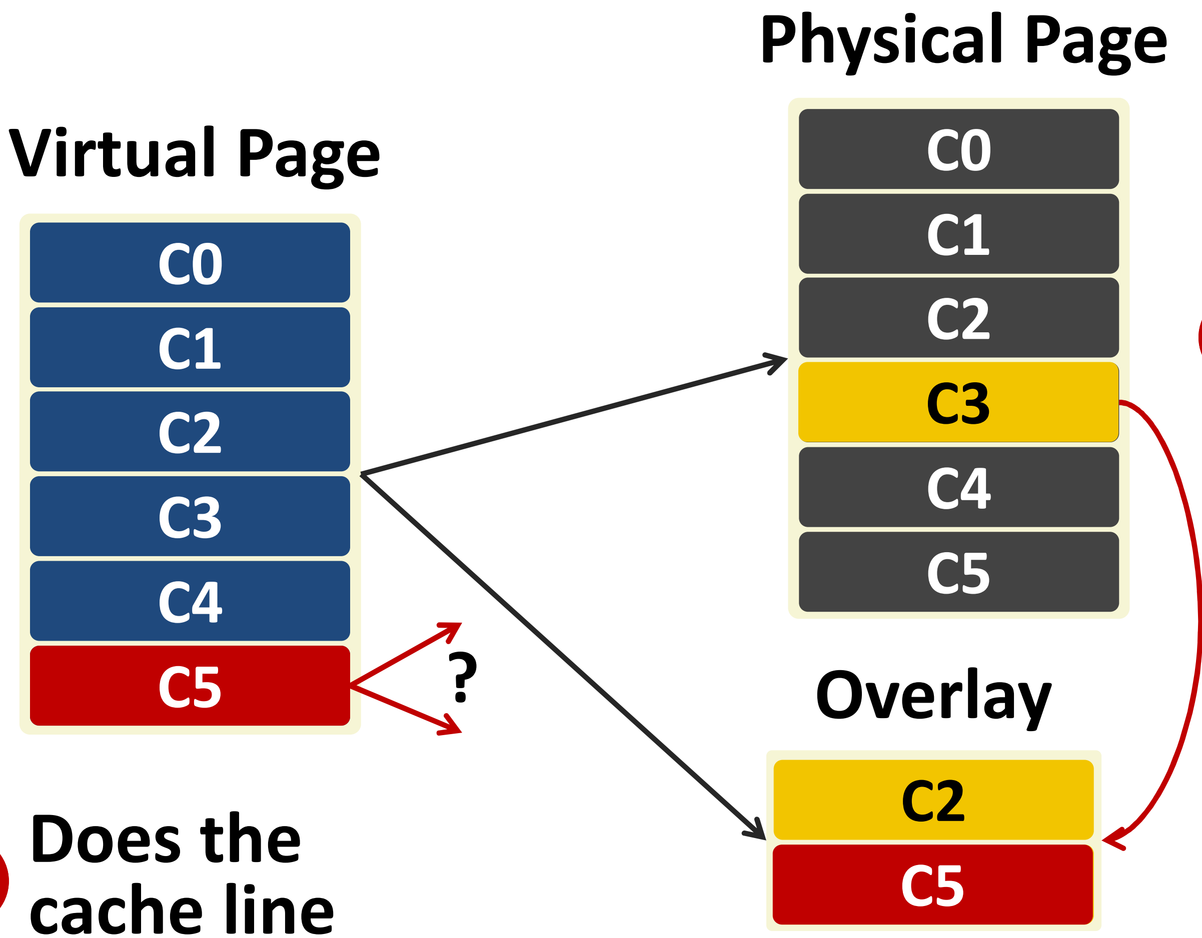
# Outline

- Shortcomings of Existing Framework
- Page Overlays – Overview
- Implementation
  - Challenges and solutions
- Applications and Evaluation
- Conclusion

# Implementation Overview



# Implementation Challenges

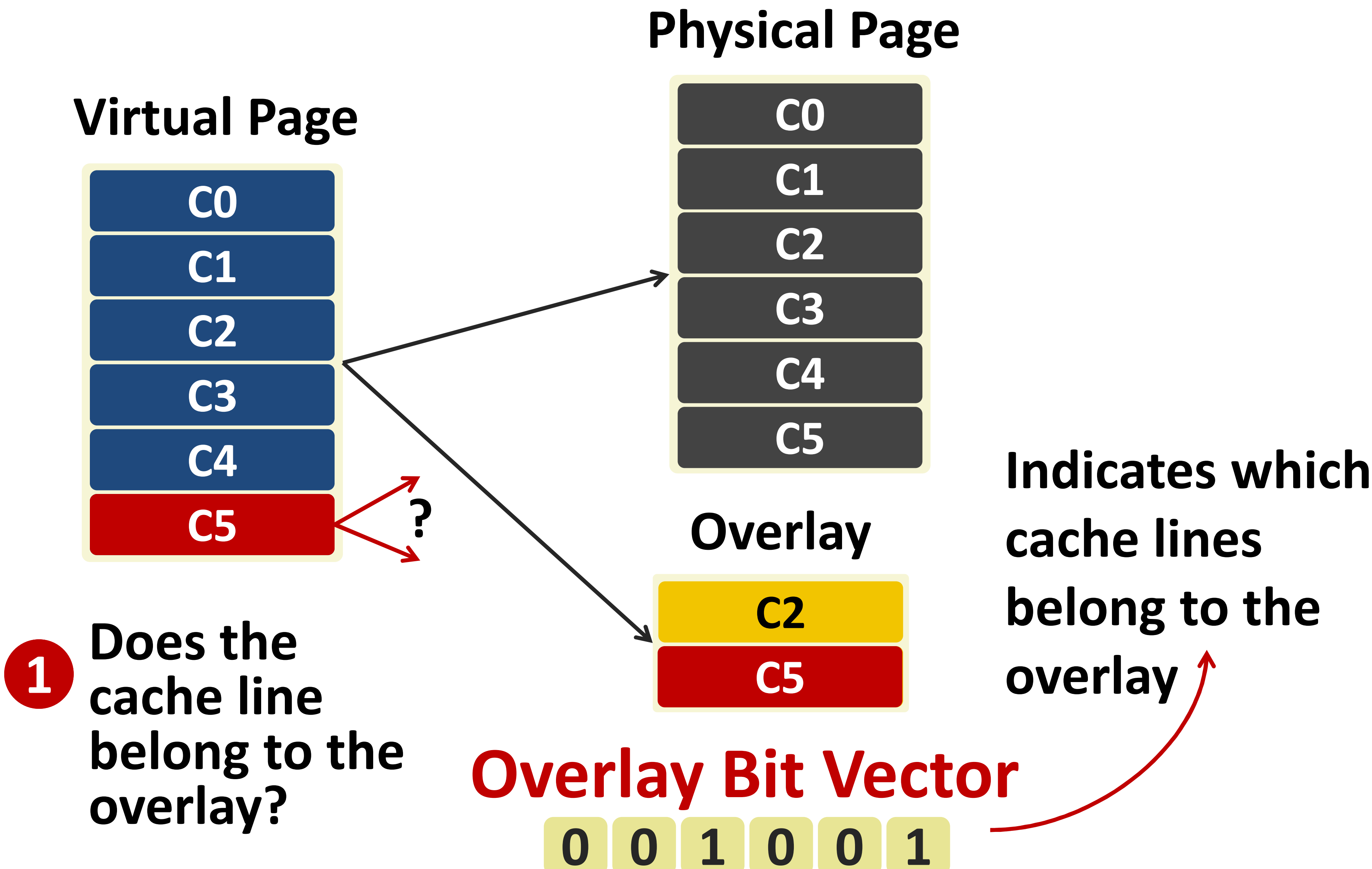



3 How to keep the TLBs coherent?

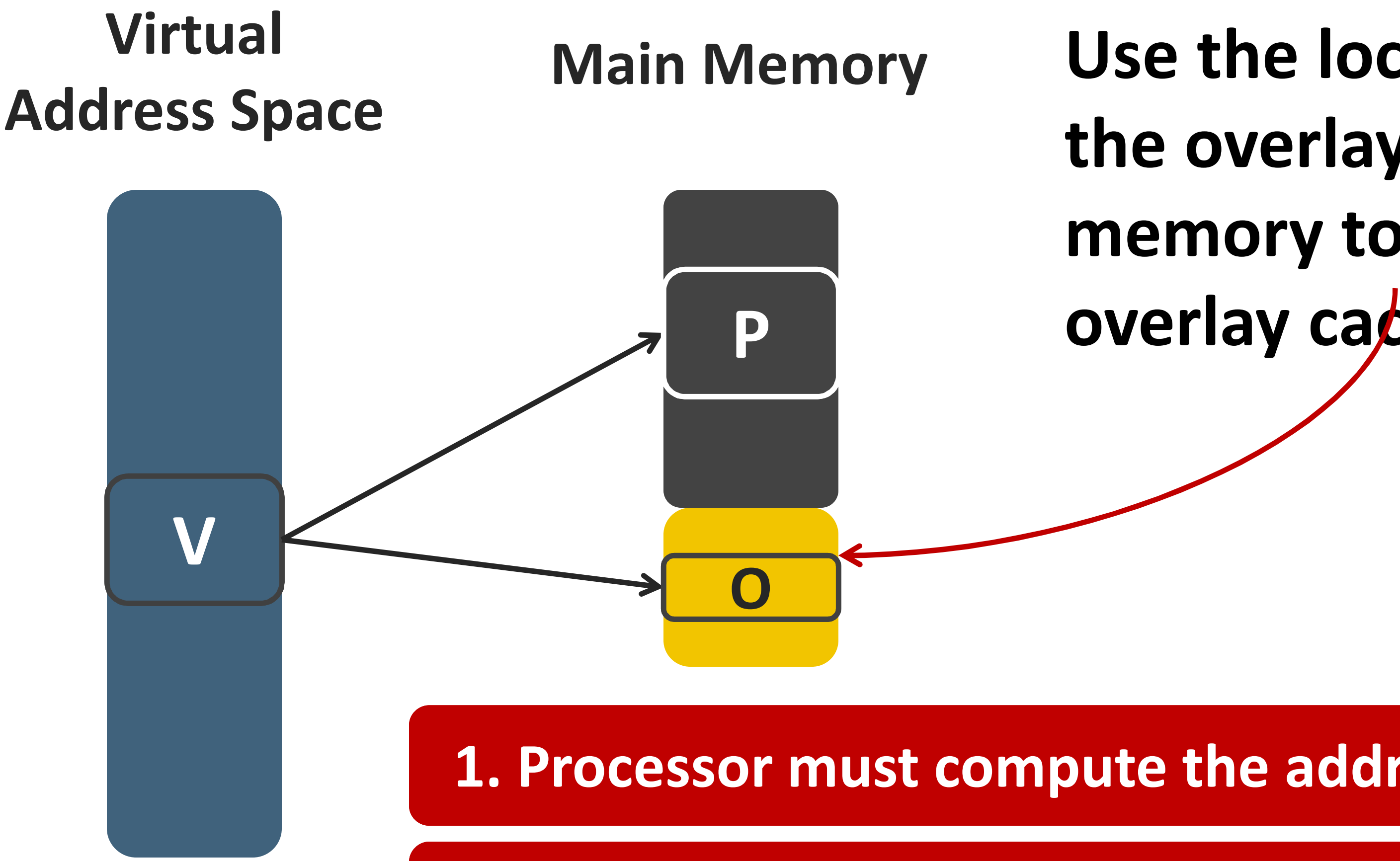
1 Does the cache line belong to the overlay?

2 What is the address/tag of the overlay cache line?

# Identifying Overlay Cache Lines: Overlay Bit Vector



# Addressing Overlay Cache Lines: Naïve Approach



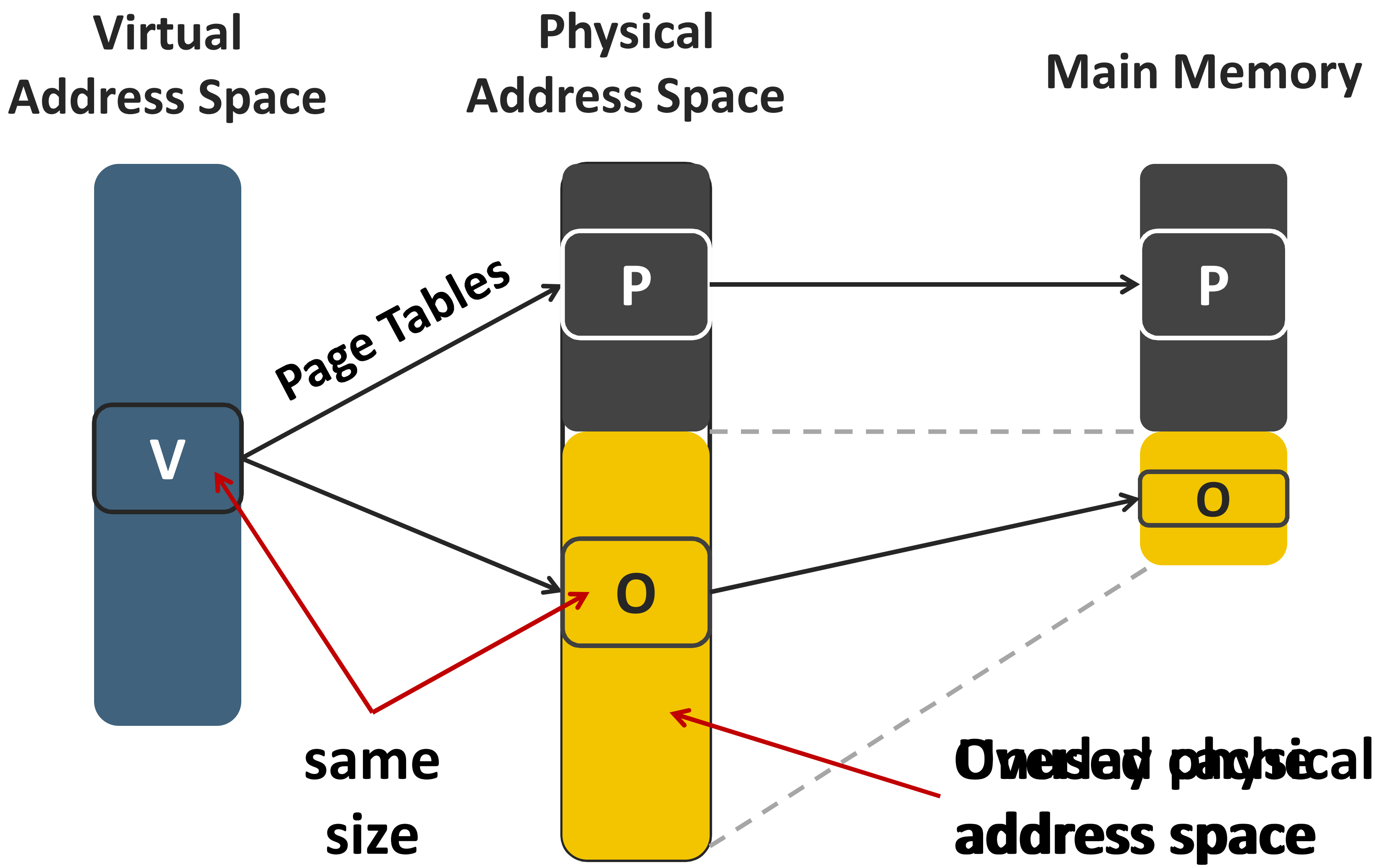

Use the location of the overlay in main memory to tag overlay cache lines

1. Processor must compute the address

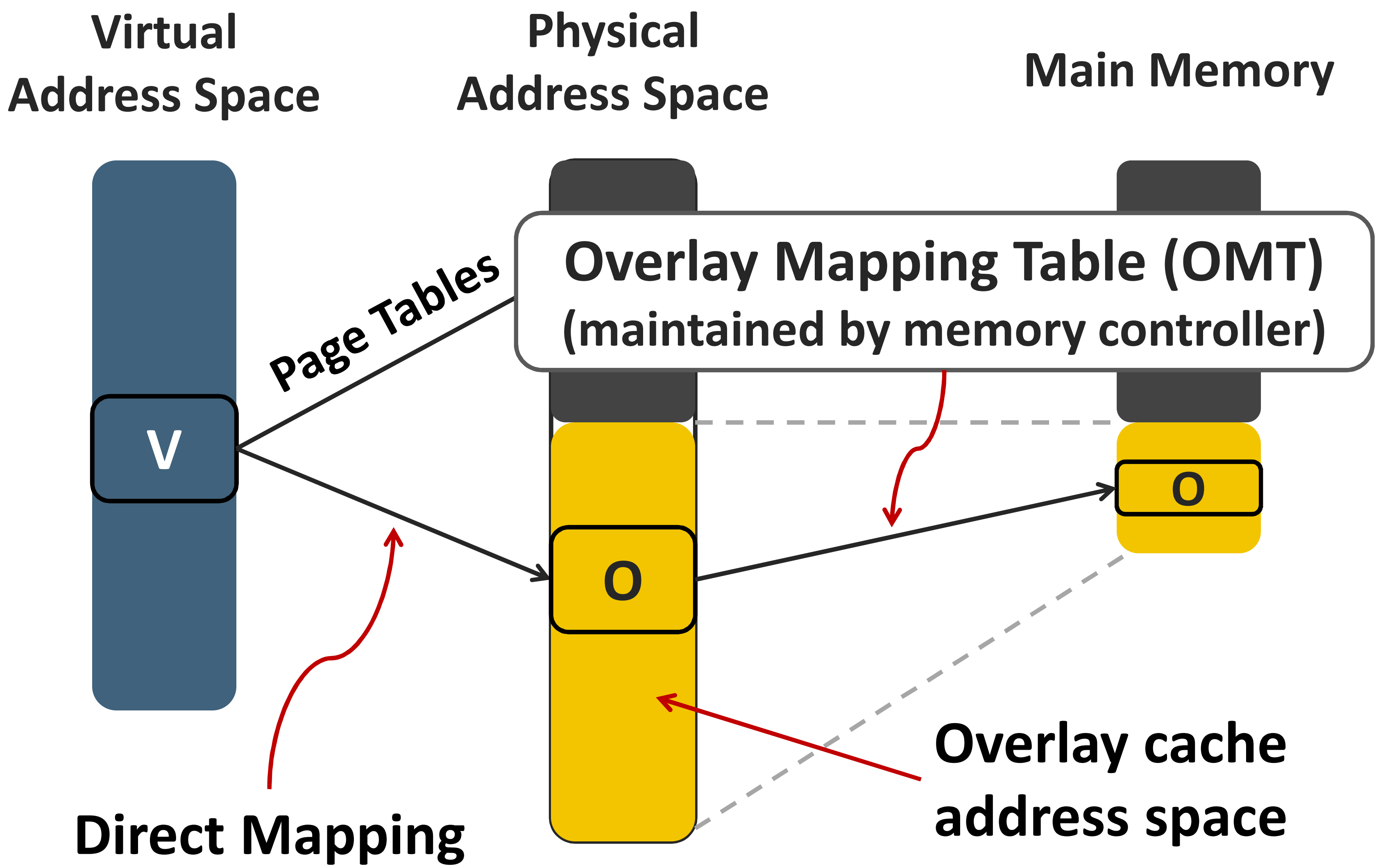
2. Does not work with virtually-indexed caches

3. Complicates overlay cache line insertion

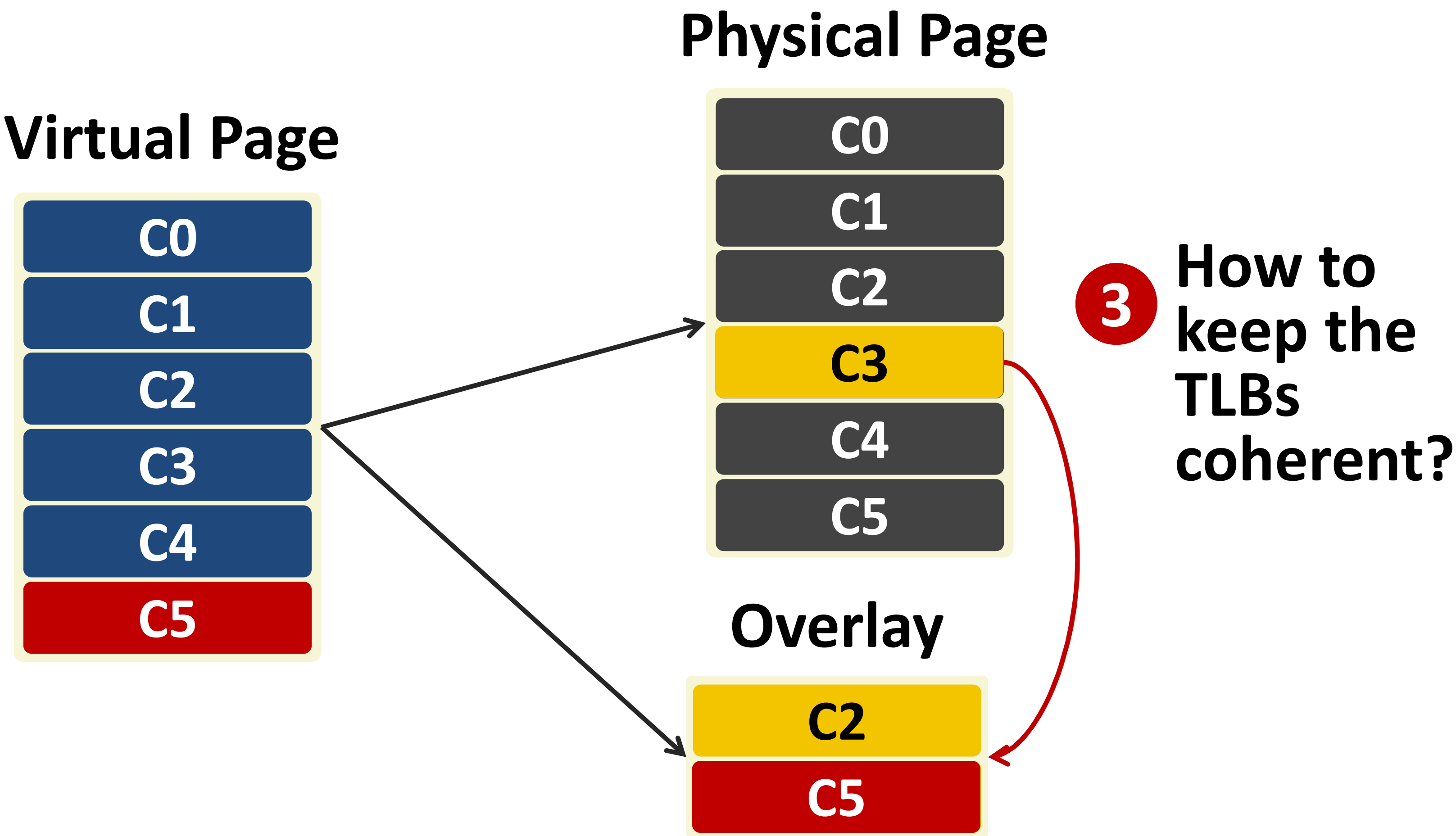
# Addressing Overlay Cache Lines: Dual Address Design



# Virtual-to-Overlay Mappings

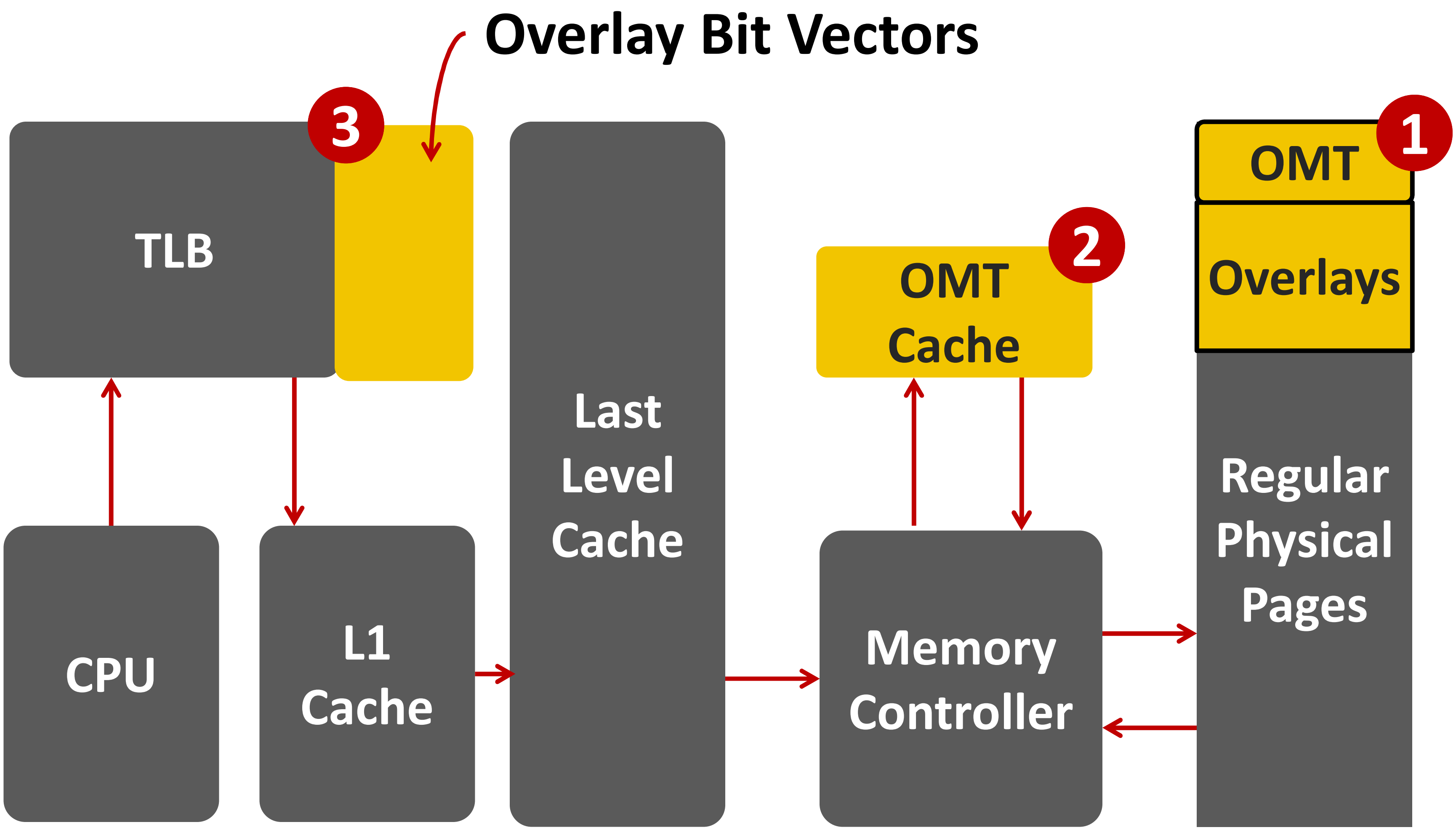


# Keeping TLBs Coherent




**Use the cache coherence protocol to keep TLBs coherent!**

# Final Implementation



# Other Details in the Paper

- Virtual-to-overlay mapping
- TLB and cache coherence
- OMT management (by the memory controller)
- Hardware cost
  - 94.5 KB of storage
- OS Support

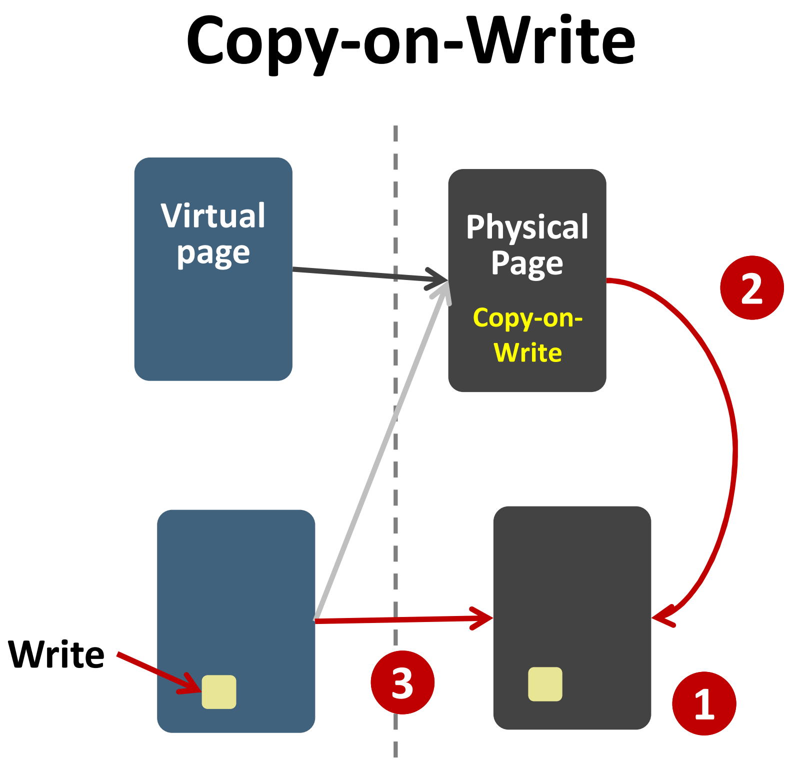
# Outline

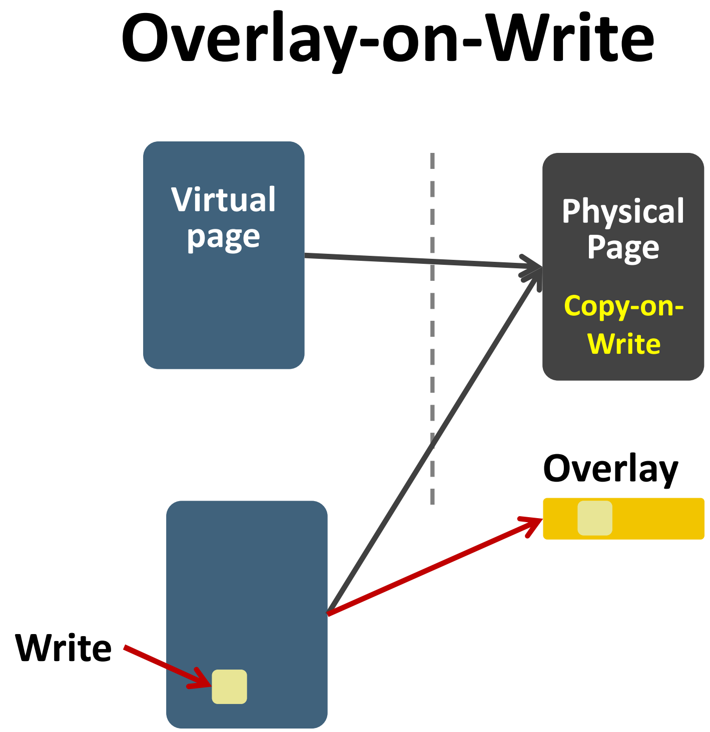
- Shortcomings of existing frameworks
- Page Overlays – Overview
- Implementation
  – Challenges and solutions
- Applications and Evaluation
- Conclusion

# Methodology

- Memsim memory system simulator [Seshadri+ PACT 2012]
- 2.67 GHz, single core, out-of-order, 64 entry instruction window
- 64-entry L1 TLB, 1024-entry L2 TLB
- 64KB L1 cache, 512KB L2 cache, 2MB L3 cache
- Multi-entry Stream Prefetcher [Srinath+ HPCA 2007]
- Open row, FR-FCFS, 64 entry write buffer, drain when full
- 64-entry OMT cache
- DDR3 1066 MHz, 1 channel, 1 rank, 8 banks

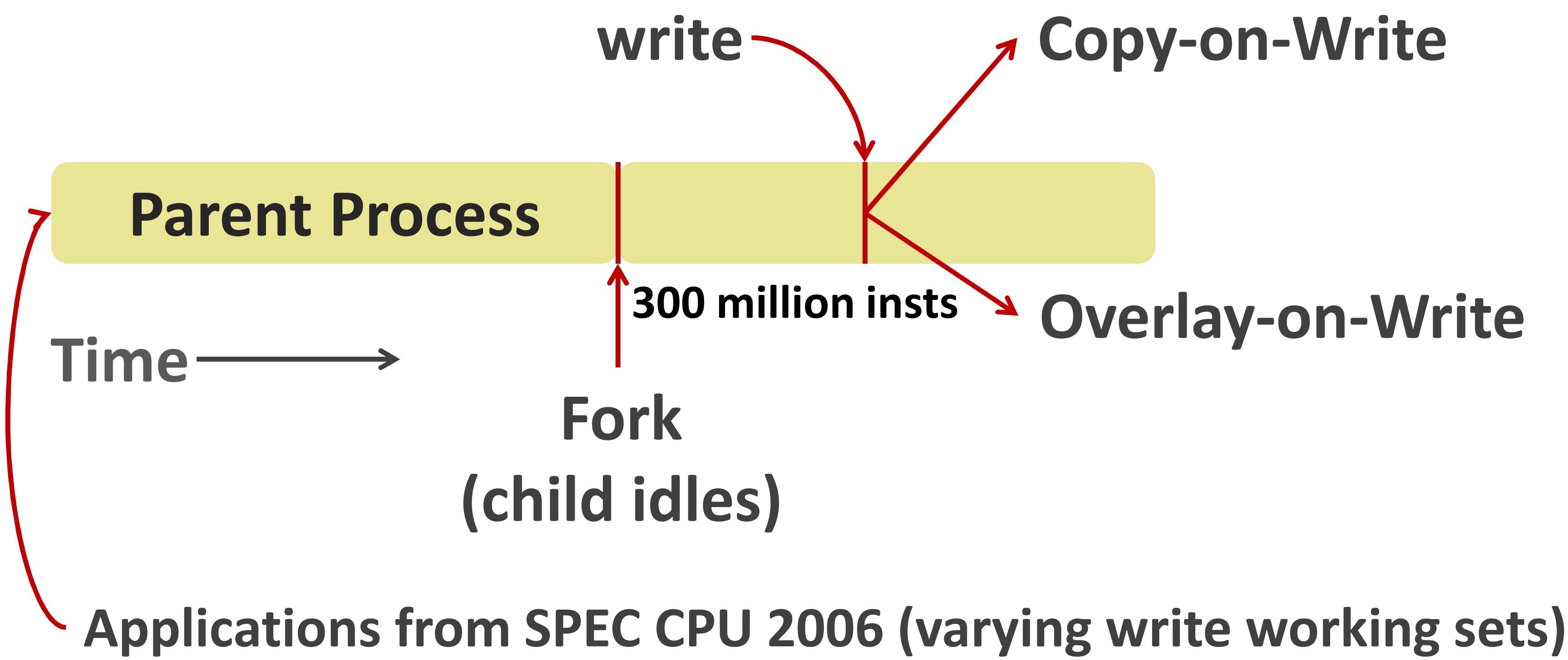
# Overlay-on-Write



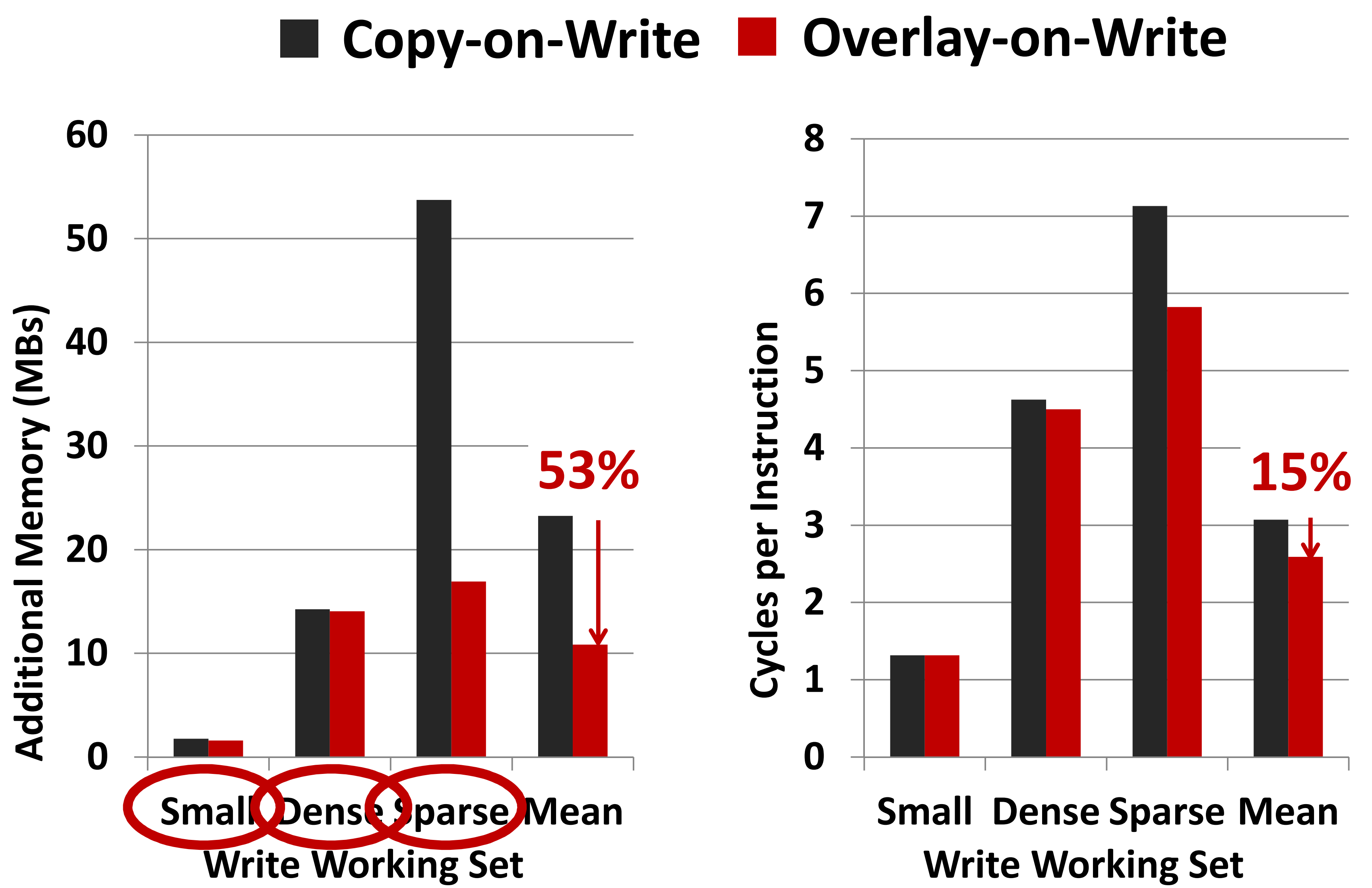

- Lower memory redundancy
- Lower latency

# Fork Benchmark



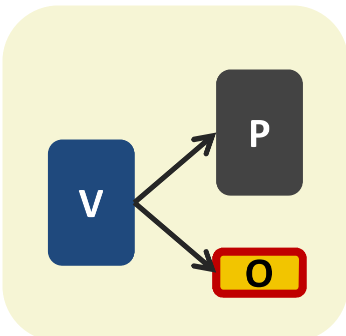

- Additional memory consumption
- Performance (cycles per instruction)

# Overlay-on-Write vs. Copy-on-Write on Fork



# Conclusion

- Sub-page memory management has several applications
  - More efficient capacity management, protection, metadata, …
- Page-granularity virtual memory → inefficient implementations
  - Low performance and high memory redundancy




- Page Overlays: New Virtual Memory Framework
- Virtual Page → (physical page, overlay)
  - Overlay contains new versions of subset of cache lines
  - Efficiently store pages with mostly similar data
- Largely retains existing virtual memory structure
  - Low cost implementation over existing frameworks
- Powerful access semantics – Enables many applications
  - E.g., overlay-on-write, efficient sparse data structure representation
- Improves performance and reduces memory redundancy

# Page Overlays

## An Enhanced Virtual Memory Framework to Enable Fine-grained Memory Management

**Vivek Seshadri**

Gennady Pekhimenko, Olatunji Ruwase,
Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch,
Todd C. Mowry, Trishul Chilimbi

SAFARI @CMU

Carnegie Mellon University

intel

Microsoft Research