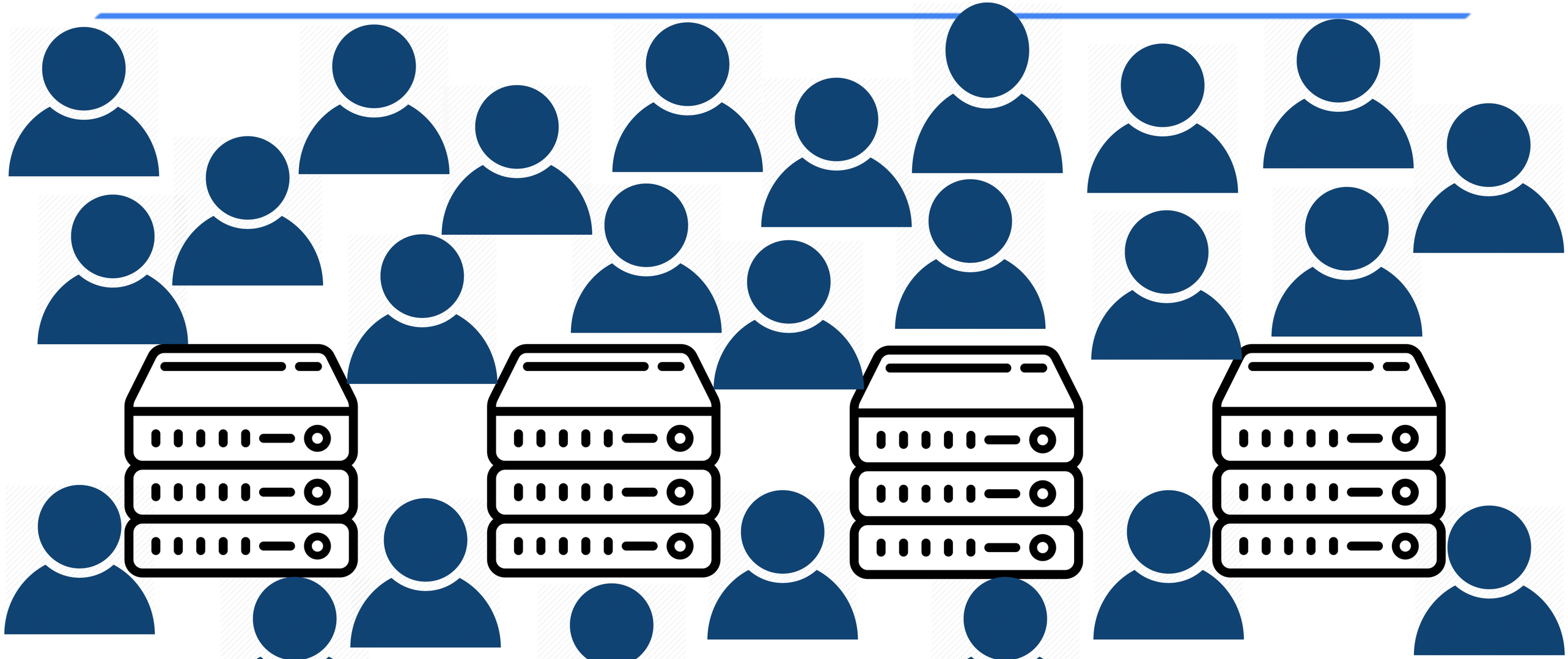
PageForge: A Near-Memory Content-Aware Page-Merging Architecture

Dimitrios Skarlatos, Nam Sung Kim, and Josep Torrellas

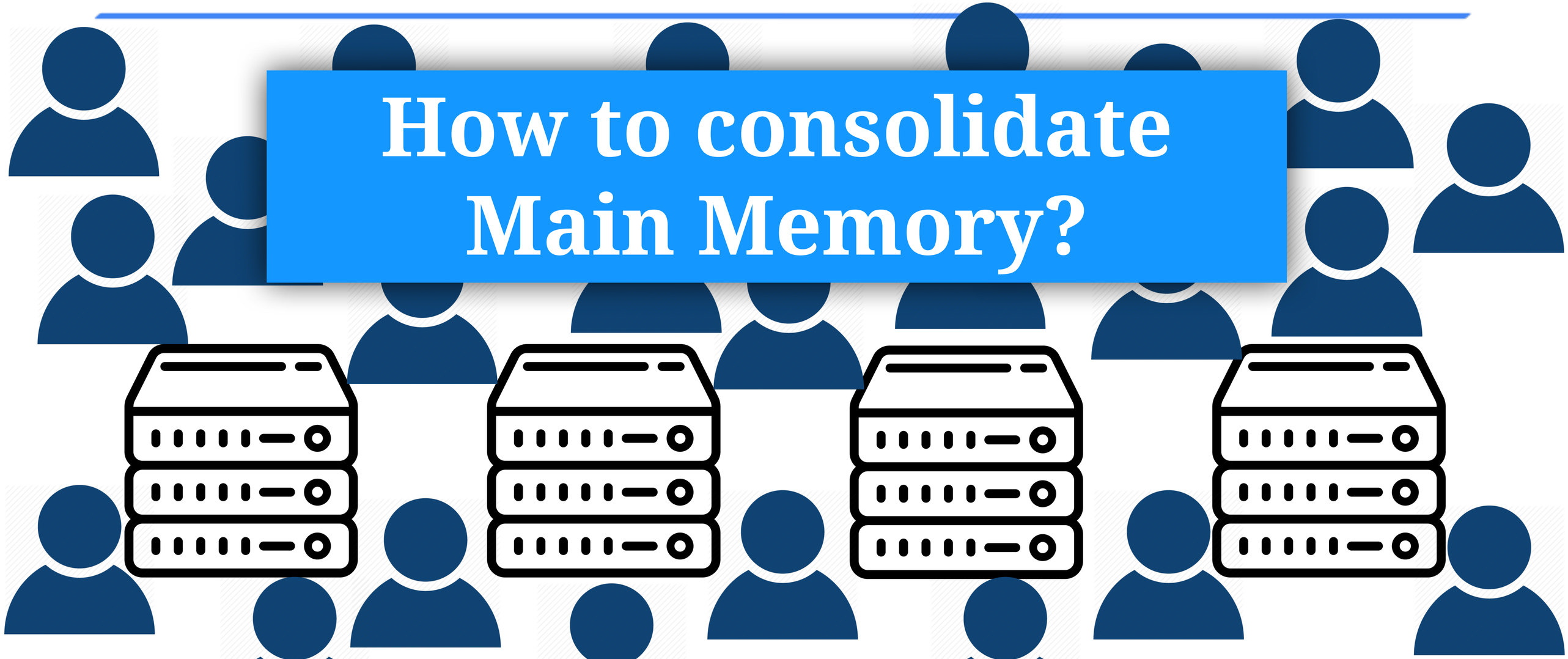
University of Illinois at Urbana-Champaign

MICRO-50 @ Boston

Motivation: Server Consolidation in the Cloud



Motivation: Server Consolidation in the Cloud



How to consolidate Main Memory?

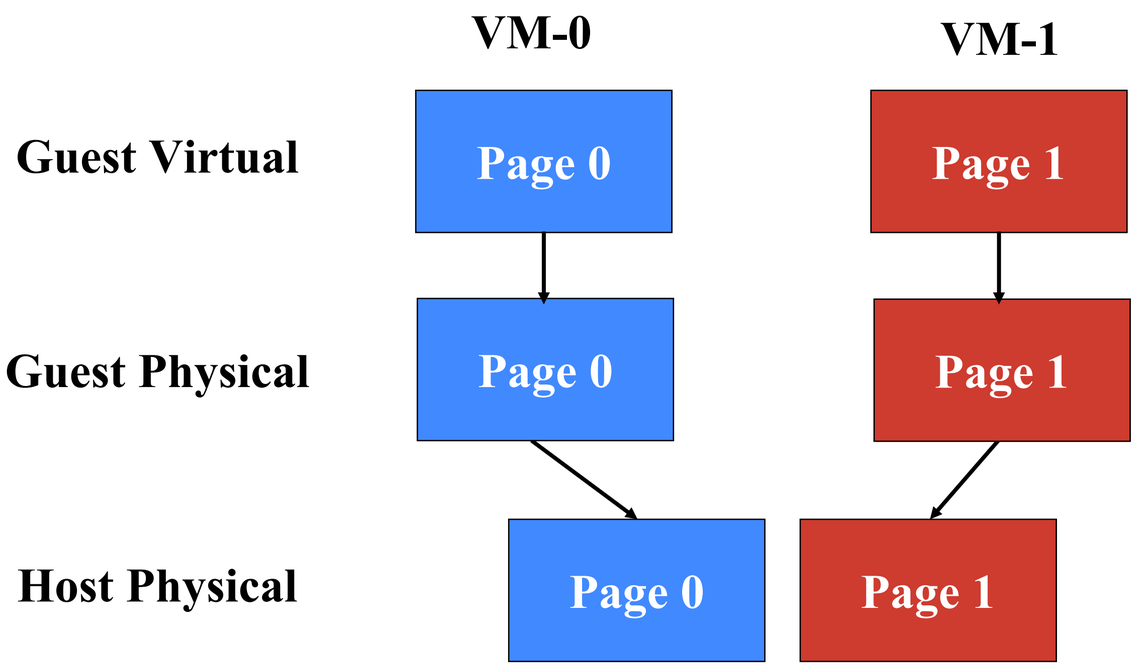
Content-Aware Page Merging or Page Deduplication

Content-Aware Page Merging or Page Deduplication

- Hypervisors search the address space and merge identical pages

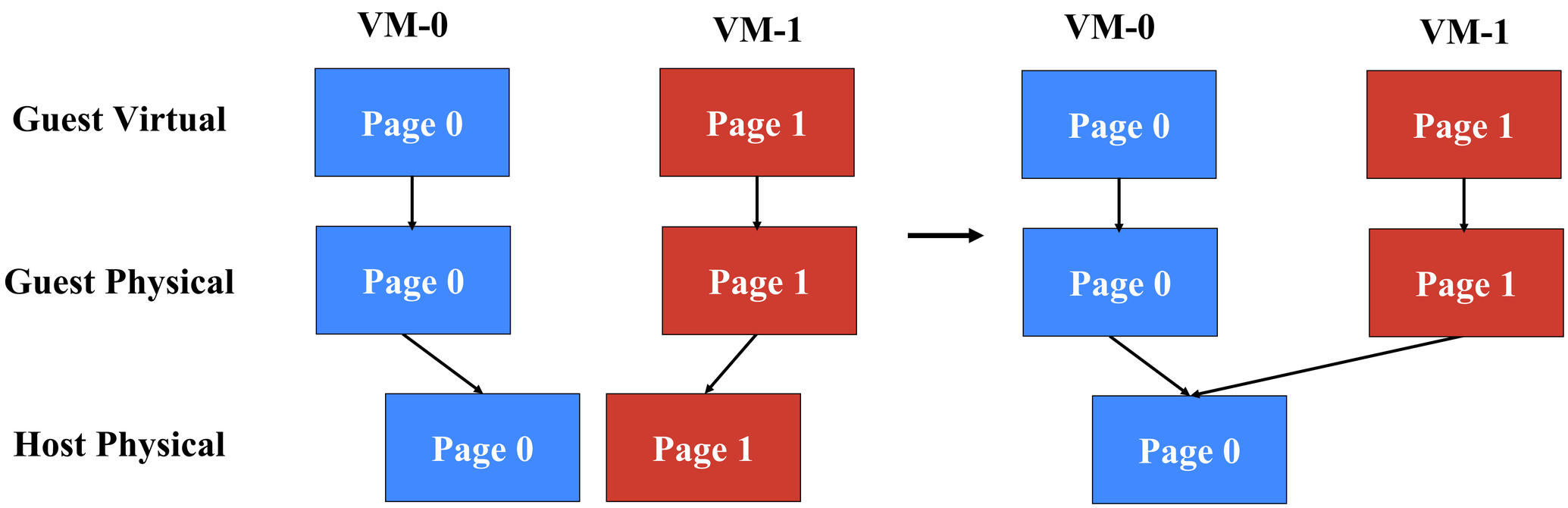
Content-Aware Page Merging or Page Deduplication

- Hypervisors search the address space and merge identical pages



Content-Aware Page Merging or Page Deduplication

- Hypervisors search the address space and merge identical pages



Problem: Overhead of Software Page Merging

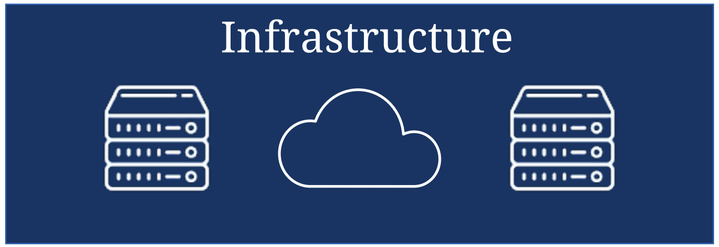
- Search hundreds of millions of pages
- Latency sensitive applications get disrupted
- RedHat's SW page merging (KSM) has execution overhead:
  - Average mean latency overhead: 68%
  - Average tail latency overhead: 136%

Contribution: **PageForge**

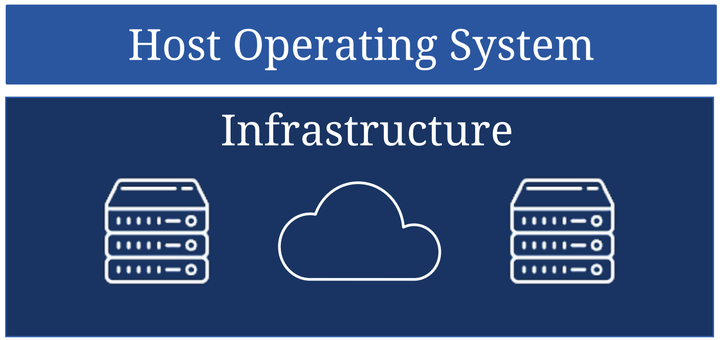
- First solution for hardware-assisted content-aware page merging
- General, effective, minimal hypervisor involvement & hardware mods
- Reduced overhead vs state-of-the-art software:
  - Mean latency 68% → 10%
  - Tail latency 136% → 11%
- Same memory savings as software: 48% → Twice #VMs
- Novel use of ECC for page content characterization

Content Duplication in the Cloud

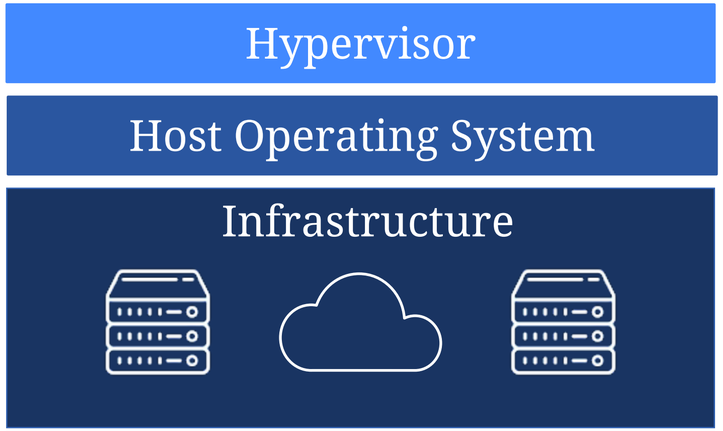
Background: Cloud



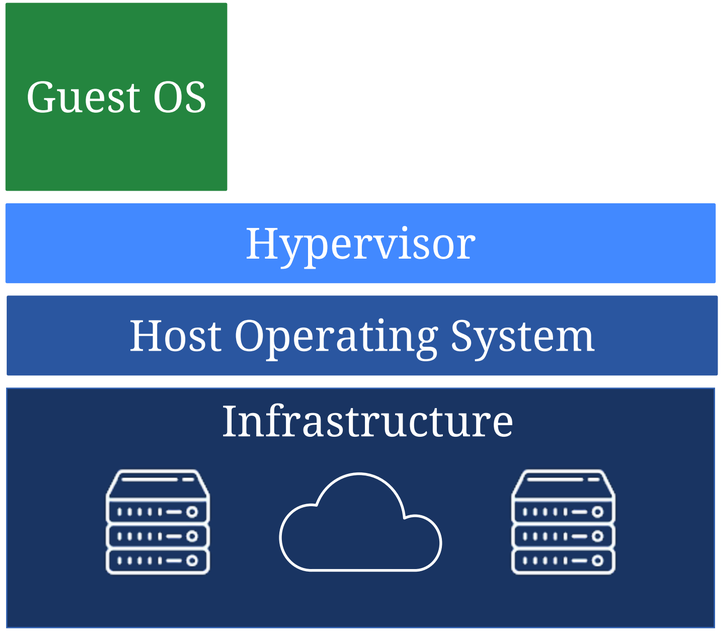
Background: Cloud



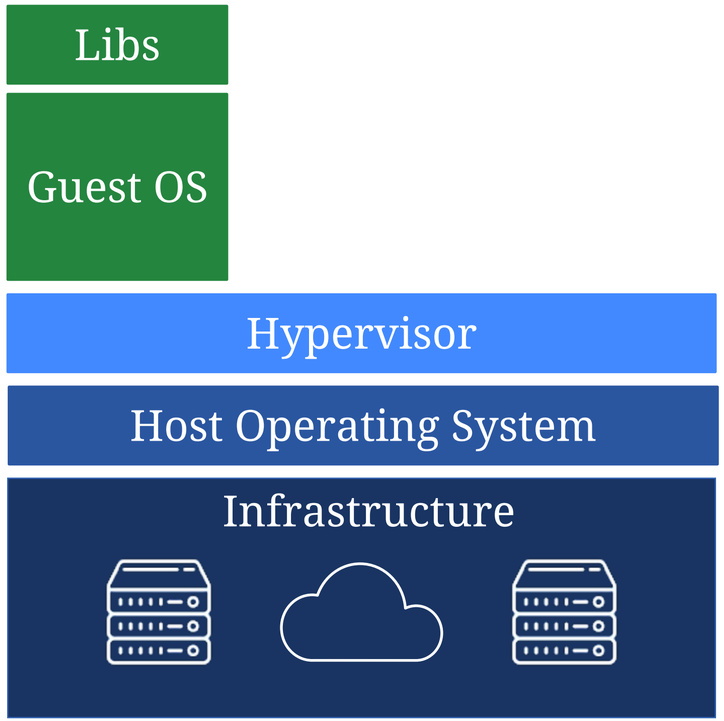
Background: Cloud



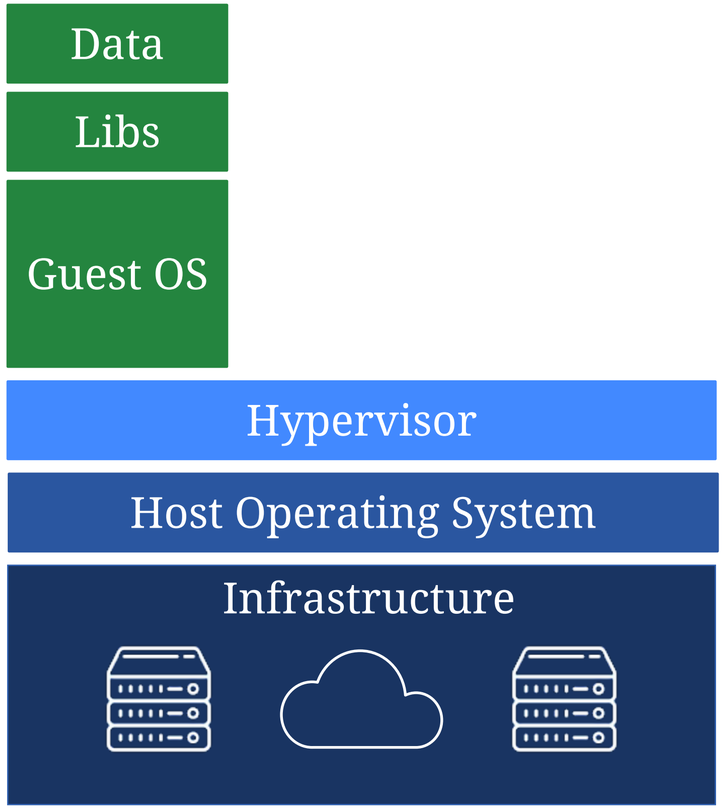
Background: Cloud



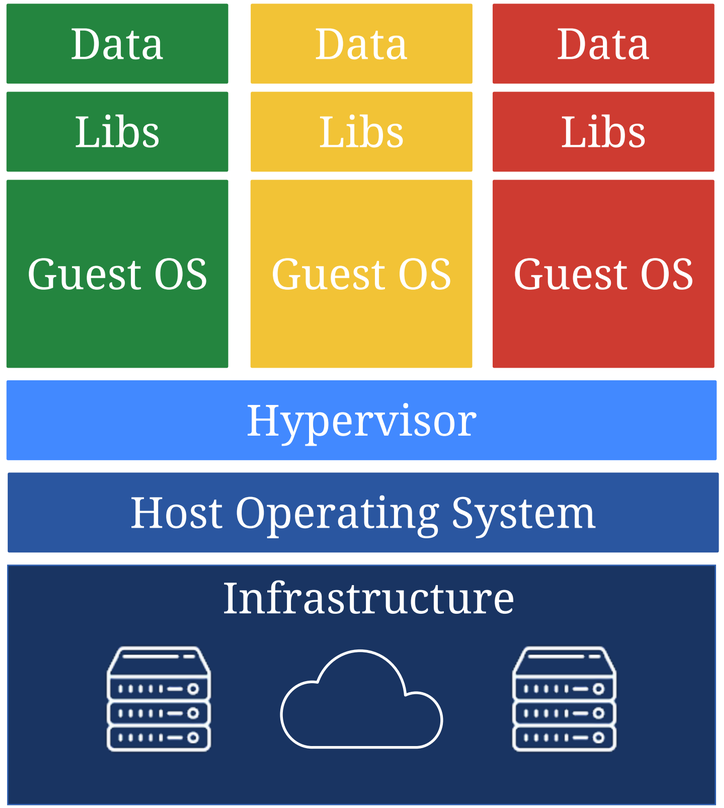
Background: Cloud



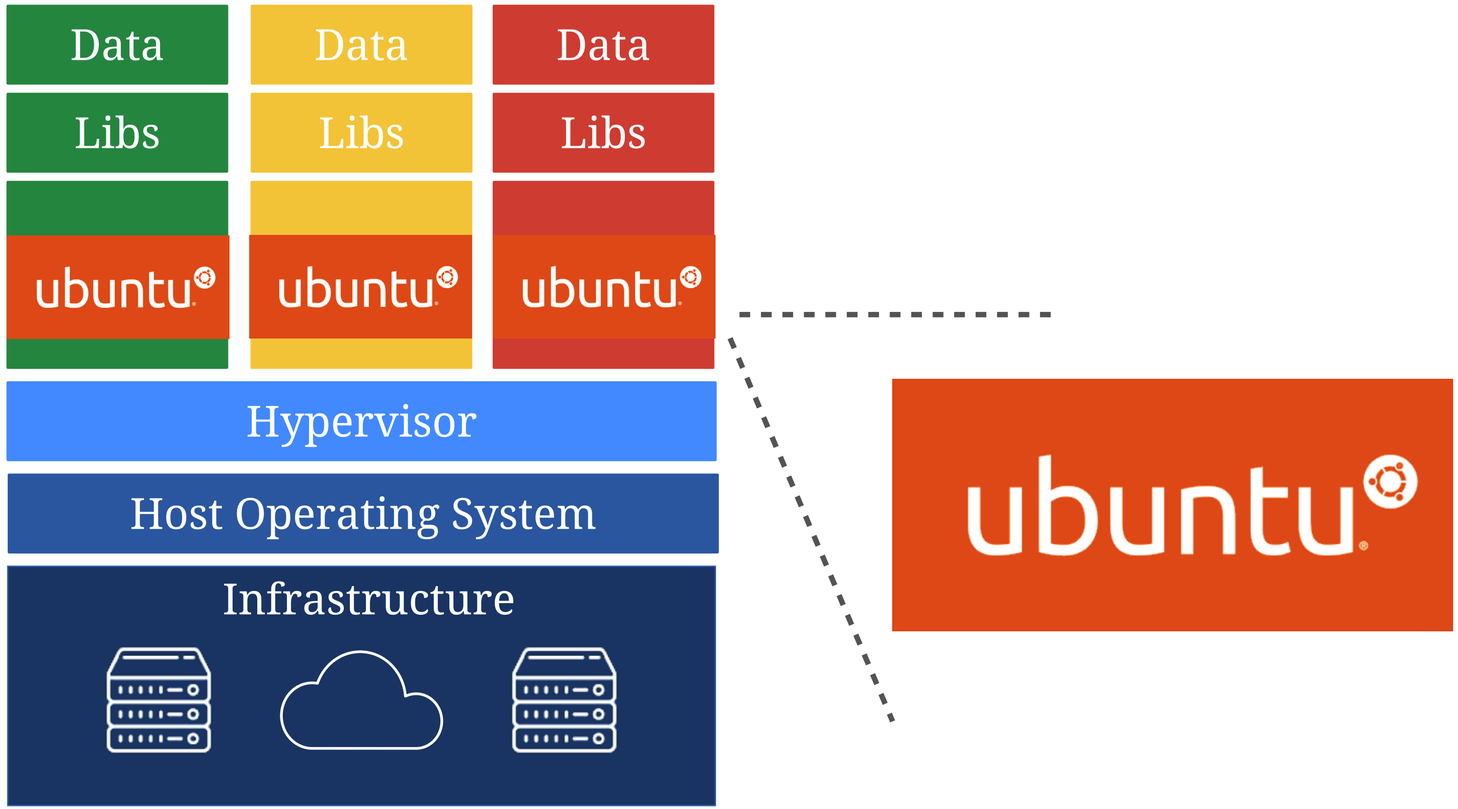
Background: Cloud



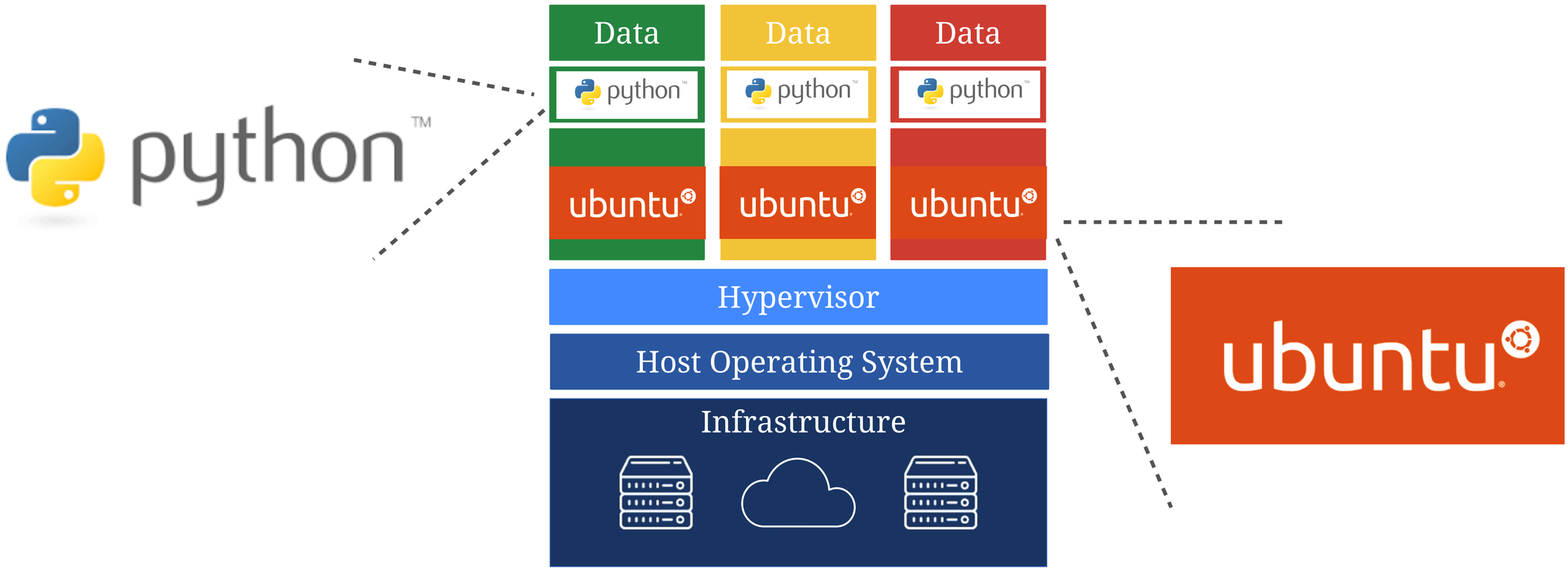
Background: Cloud



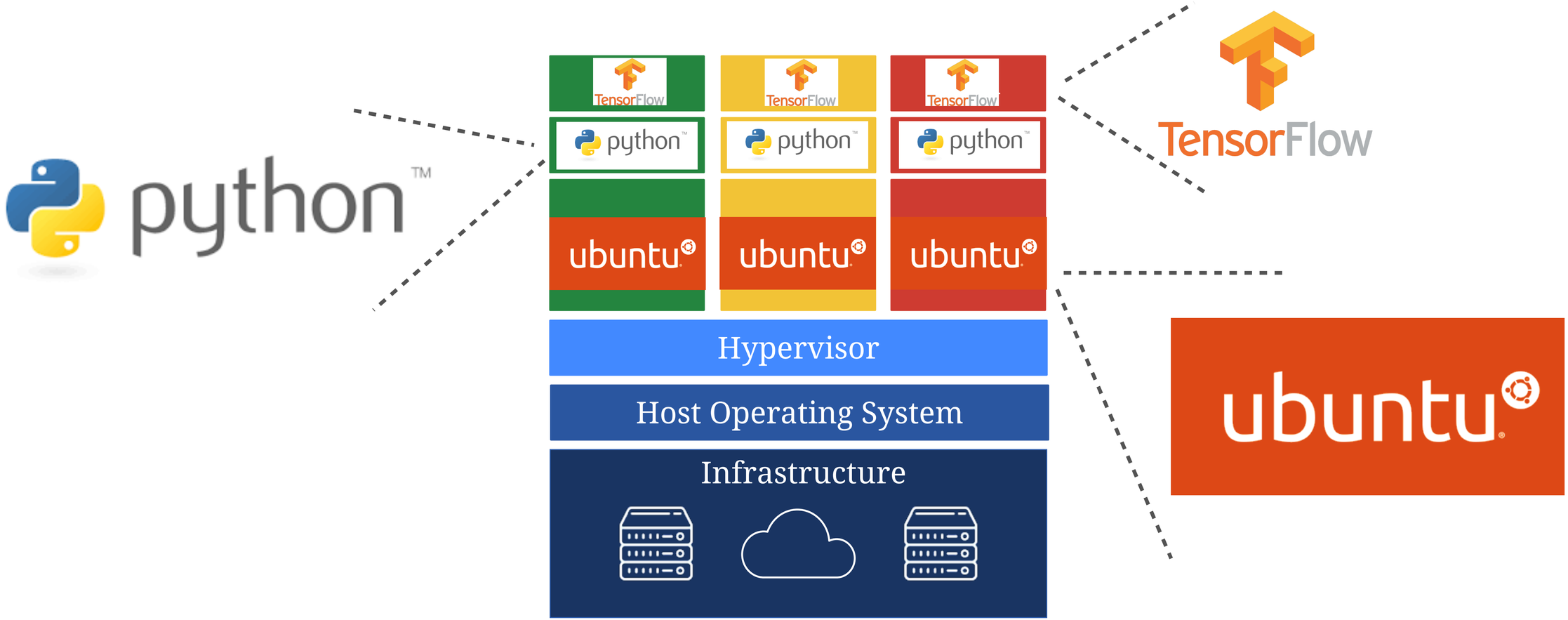
Background: Content Duplication



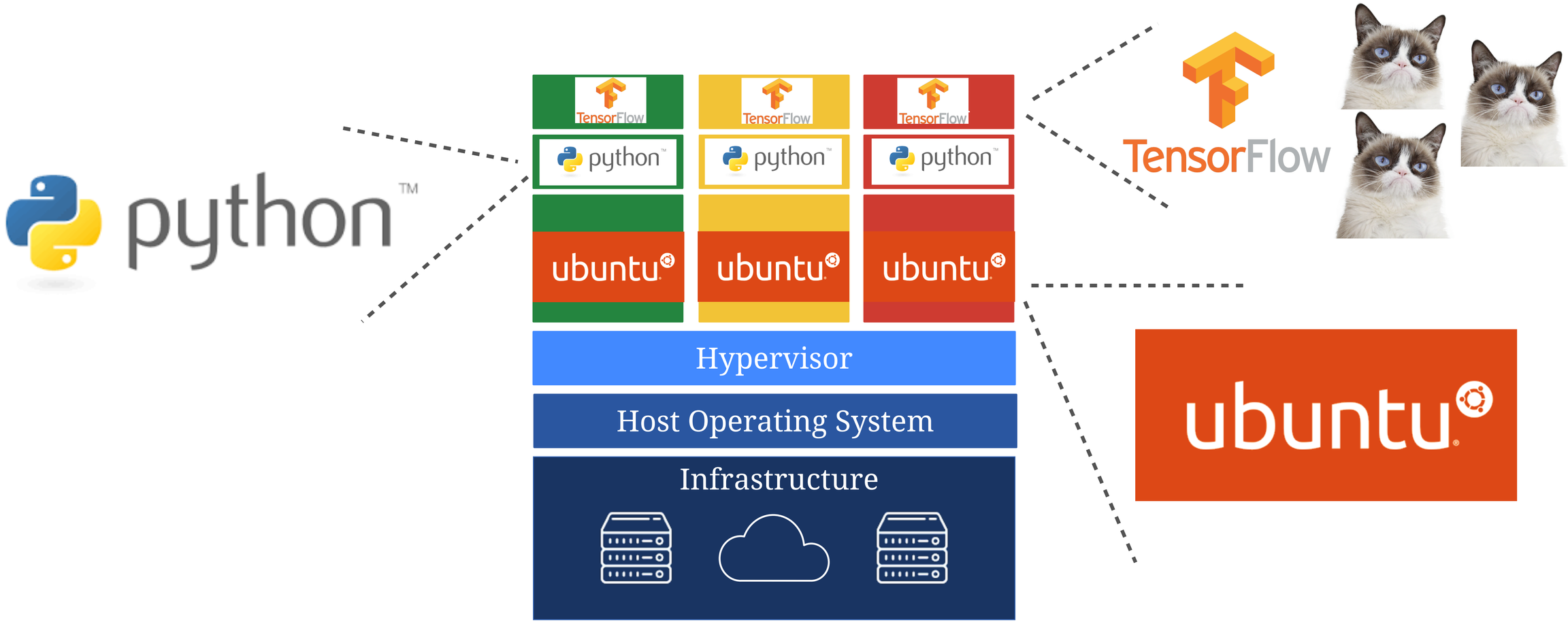
Background: Content Duplication



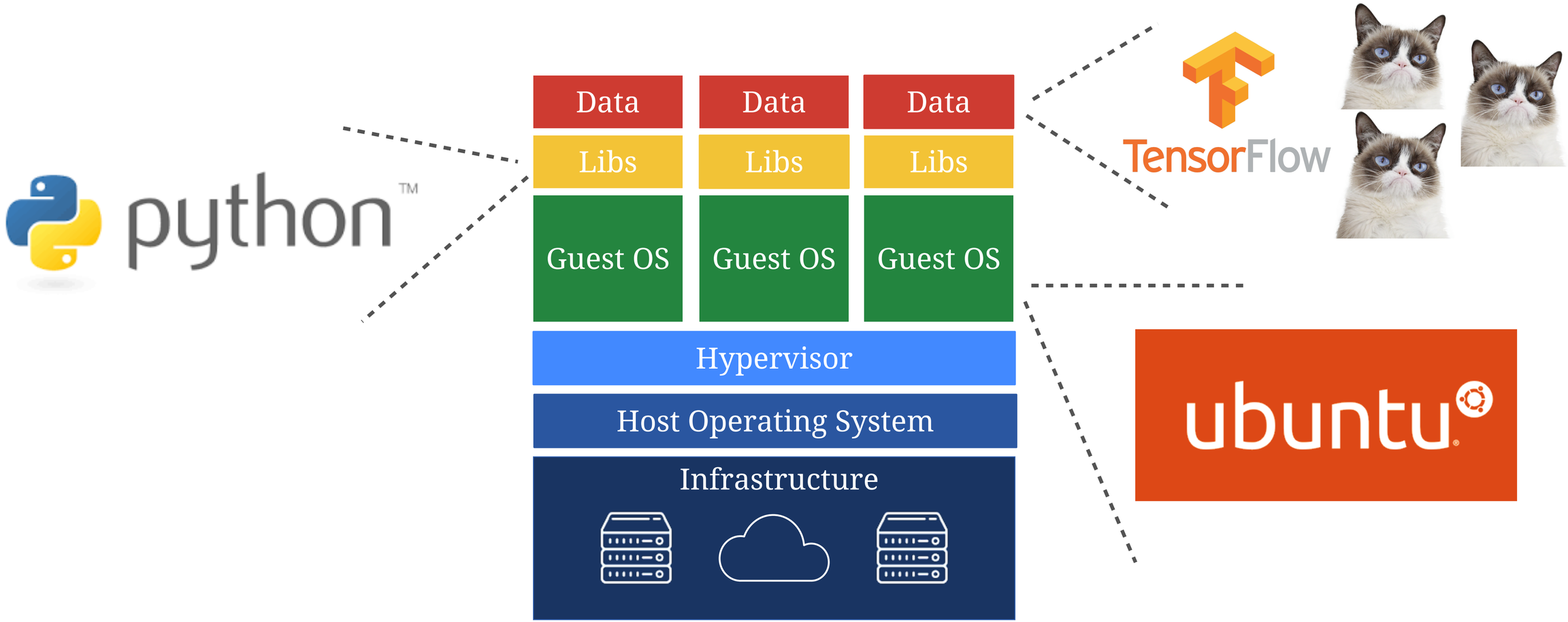
Background: Content Duplication



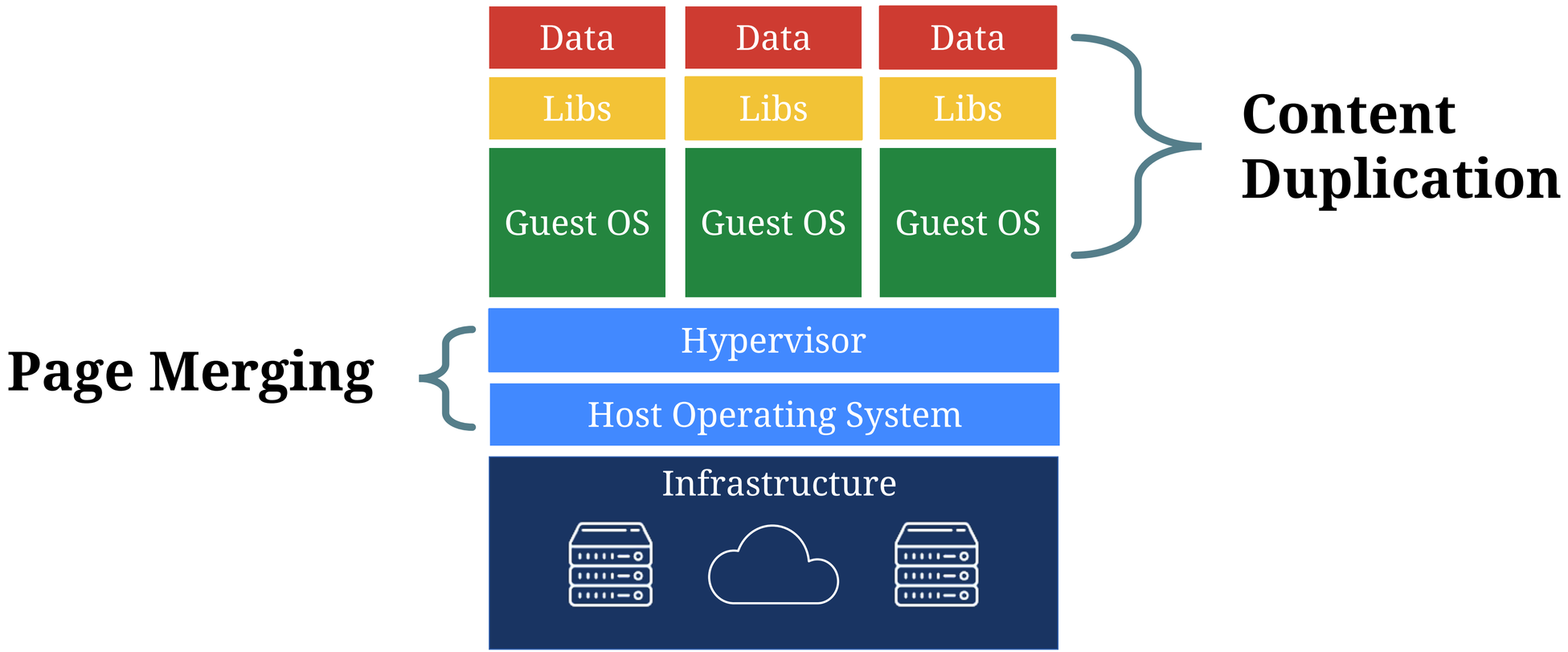
Background: Content Duplication



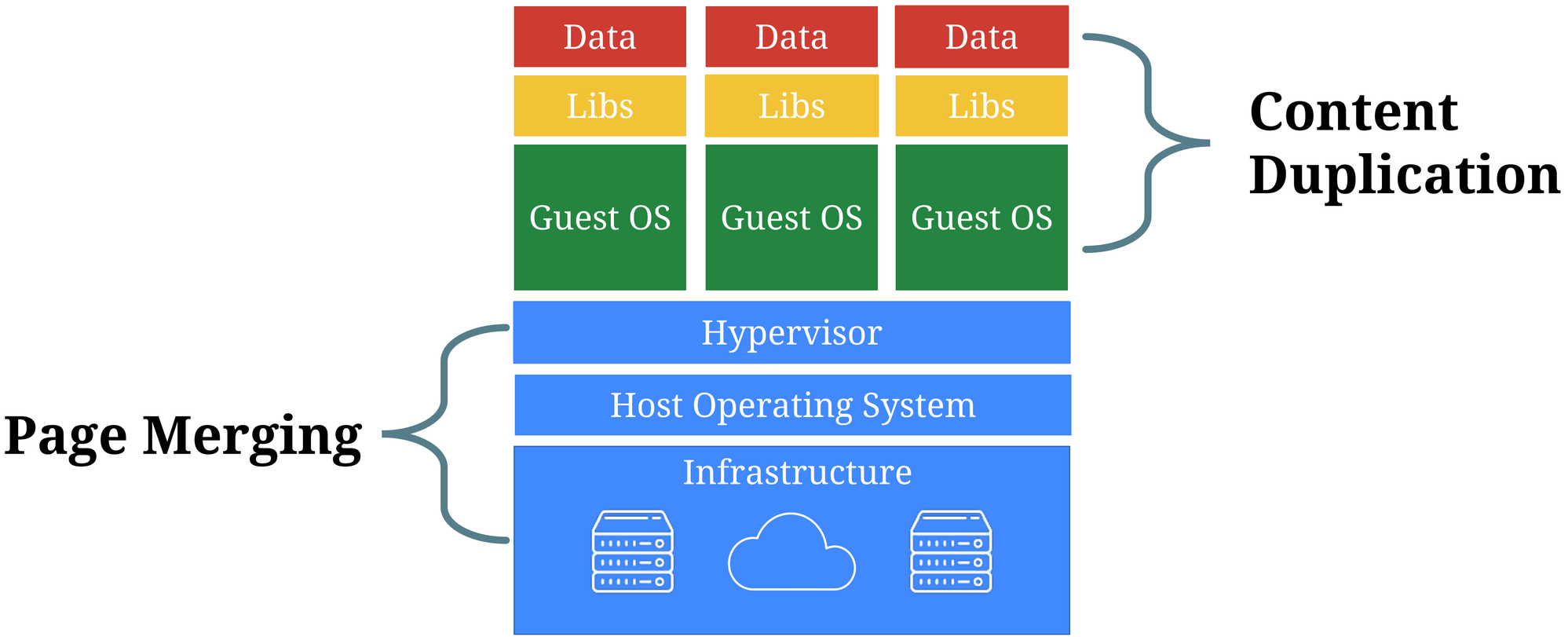
Background: Content Duplication



Background: Content Duplication



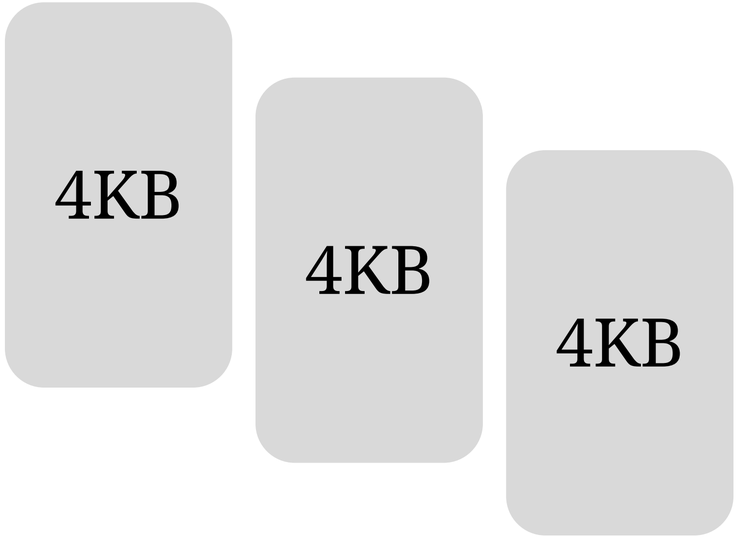
Our Proposal: Content Deduplication in HW



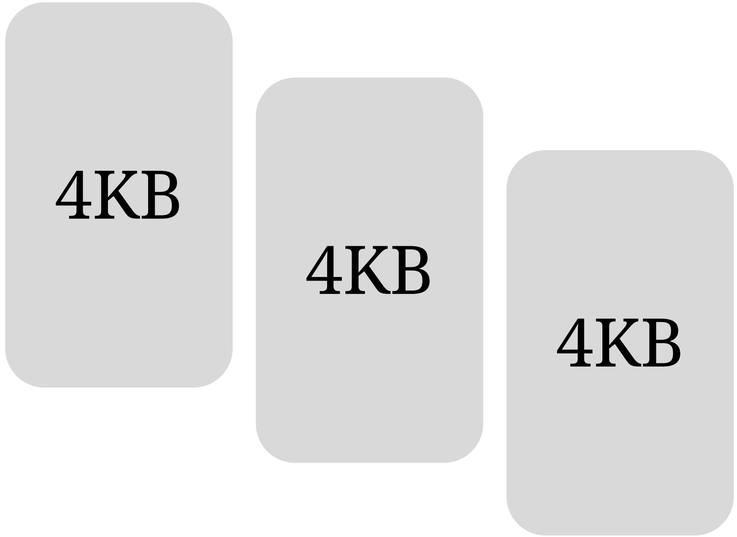
Software-based Content-Aware Page Merging

Software-Based Content-Aware Page Merging

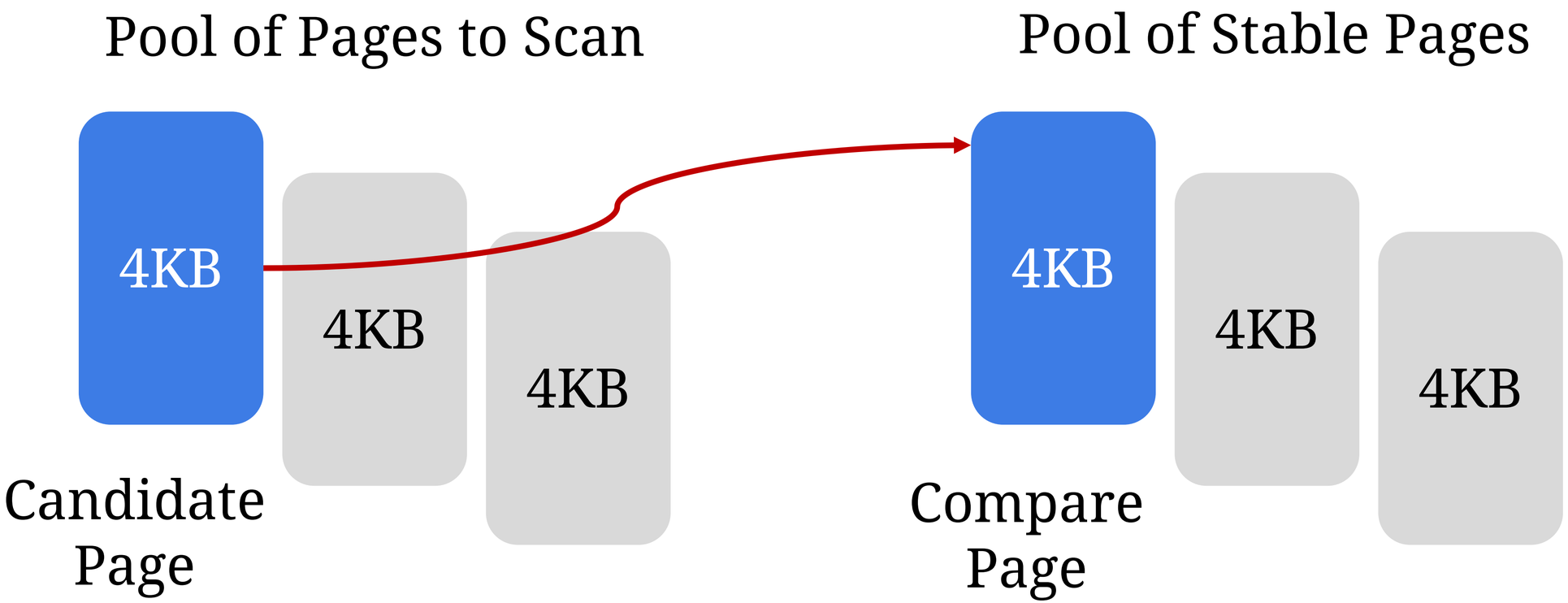
Pool of Pages to Scan



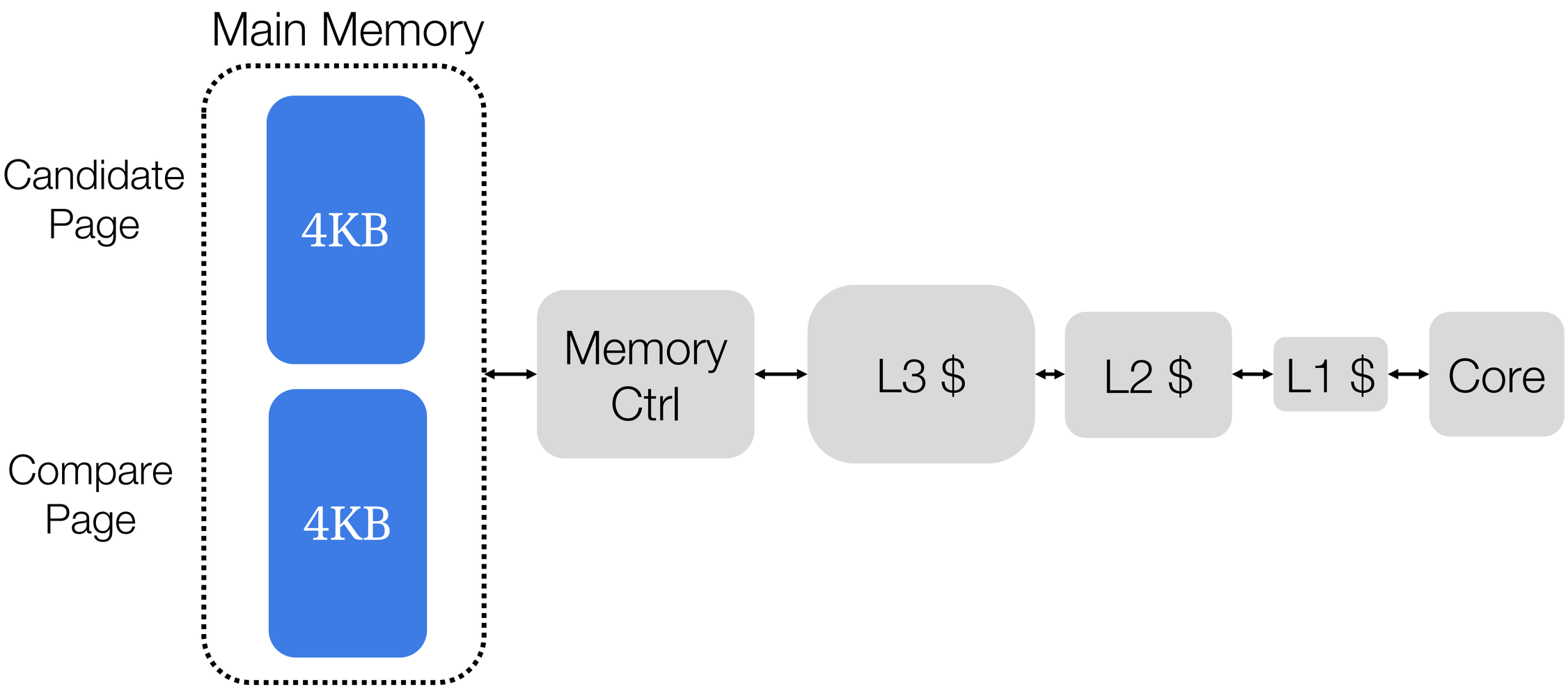
Pool of Stable Pages



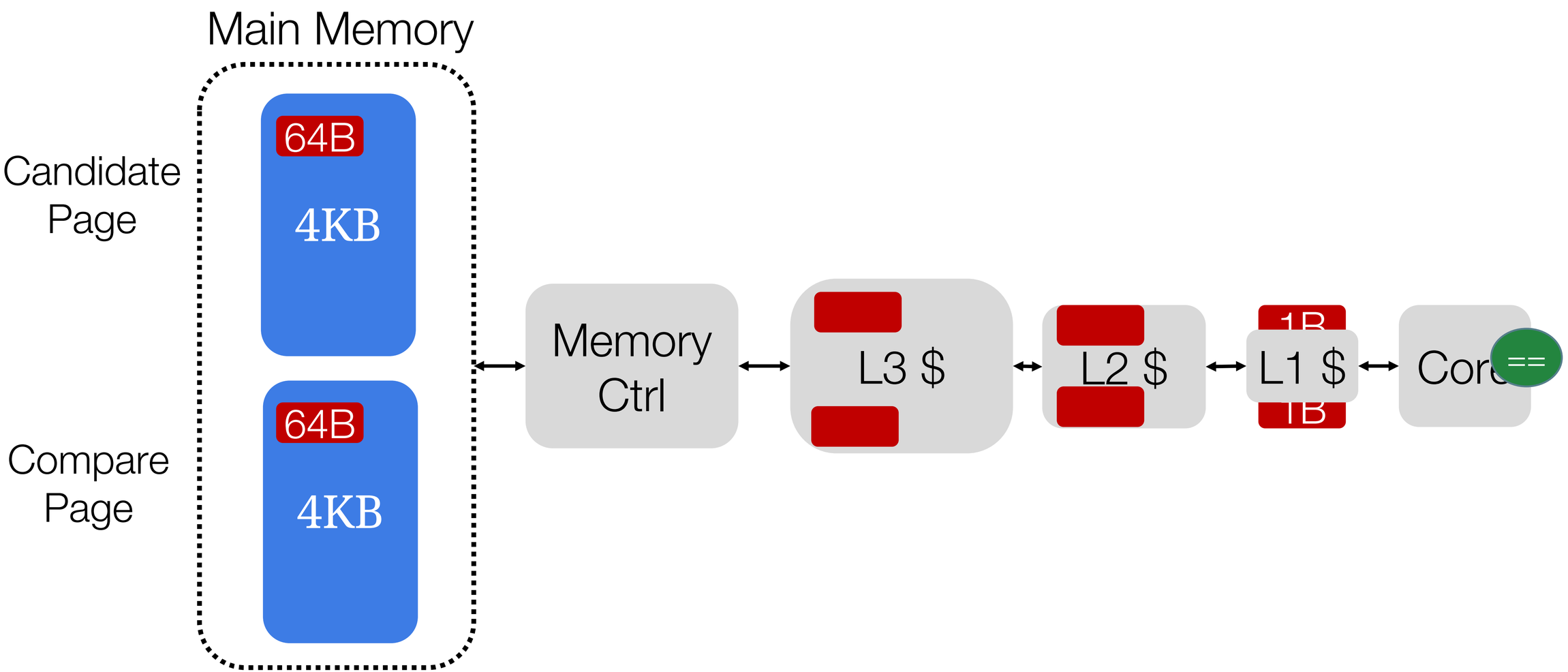
Software-Based Content-Aware Page Merging



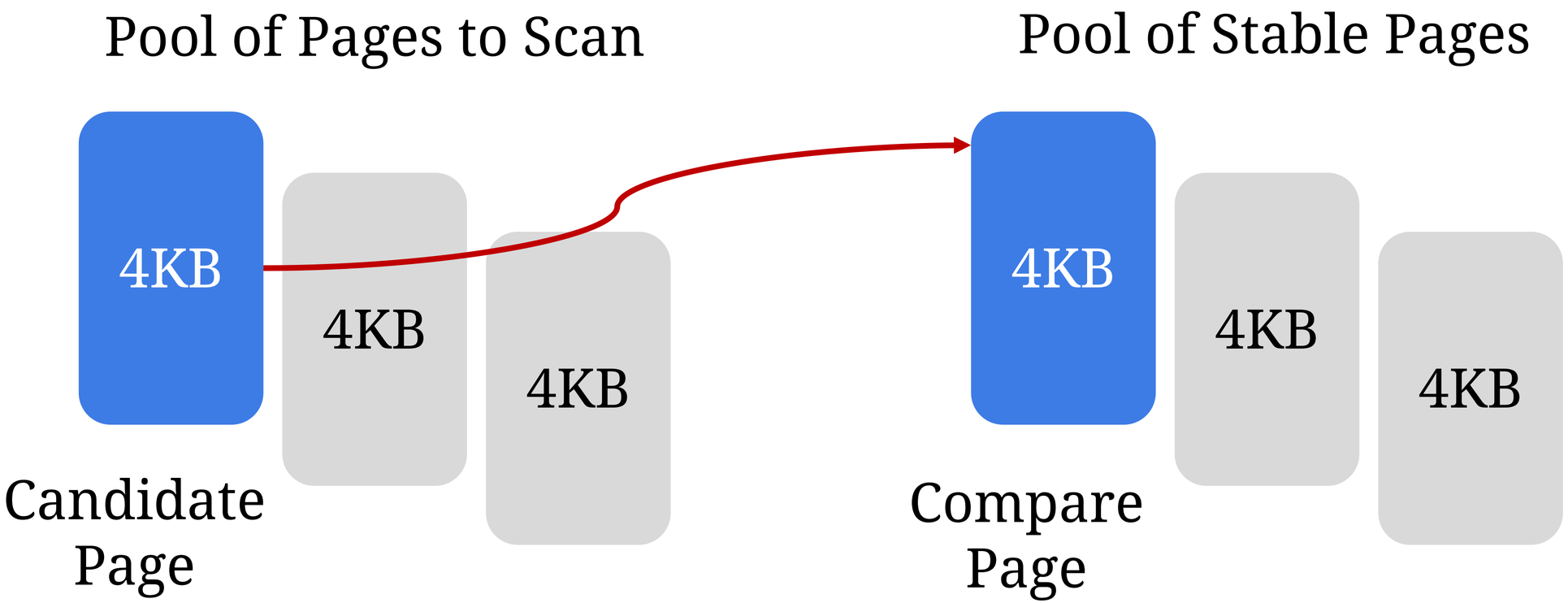
Software-Based Content-Aware Page Merging



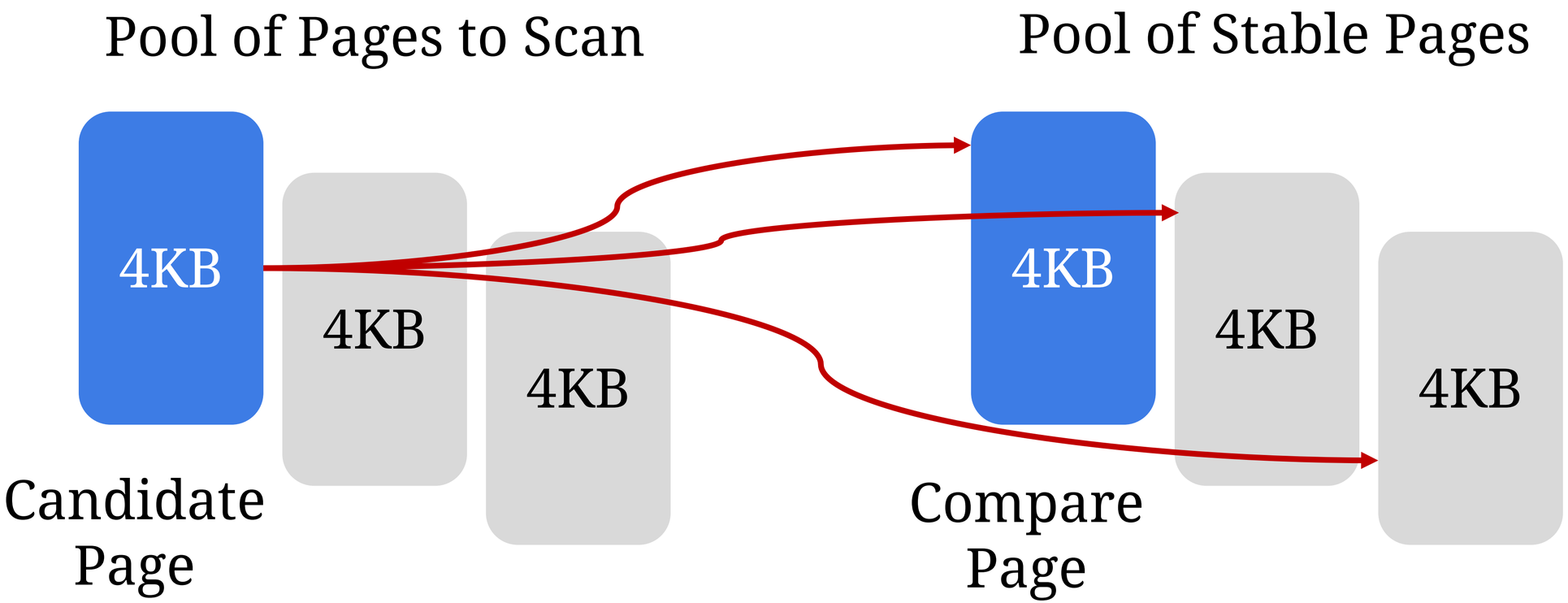
Software-Based Content-Aware Page Merging



Software-Based Content-Aware Page Merging



Software-Based Content-Aware Page Merging

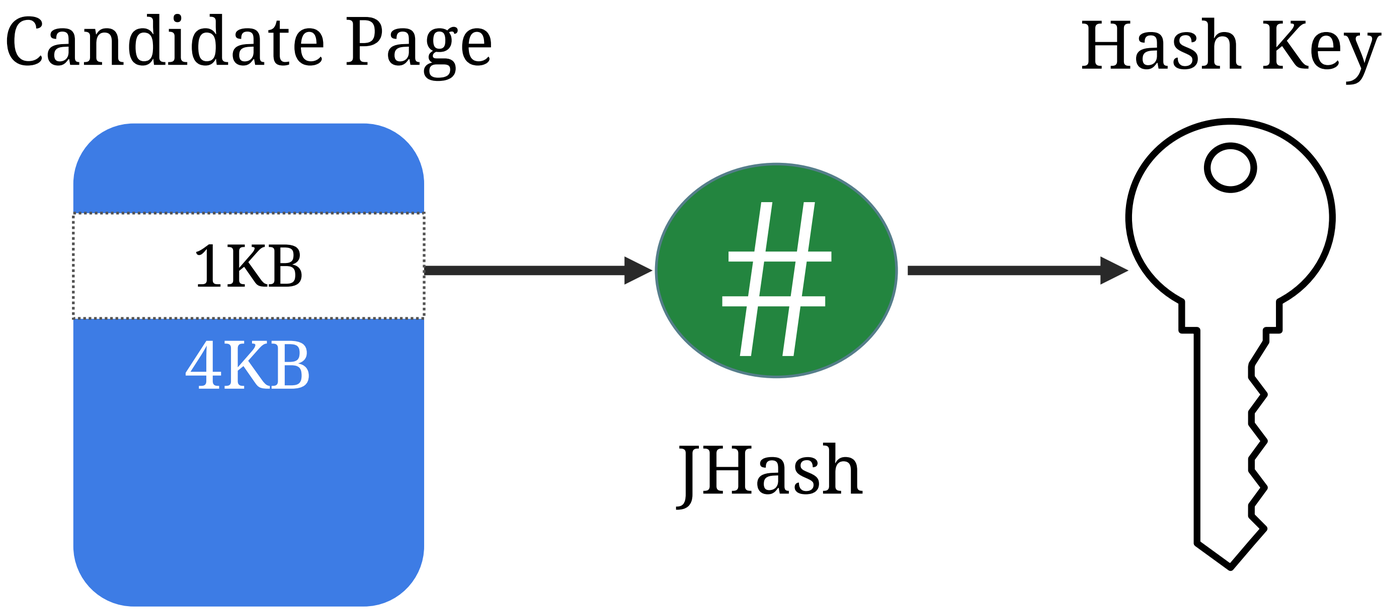


Why Page-Merging is expensive?

- Search hundreds of millions of pages
  - Many core cycles consumed
  - Caches get polluted
  - Latency sensitive applications get disrupted

Software-Based Content-Aware Page Merging

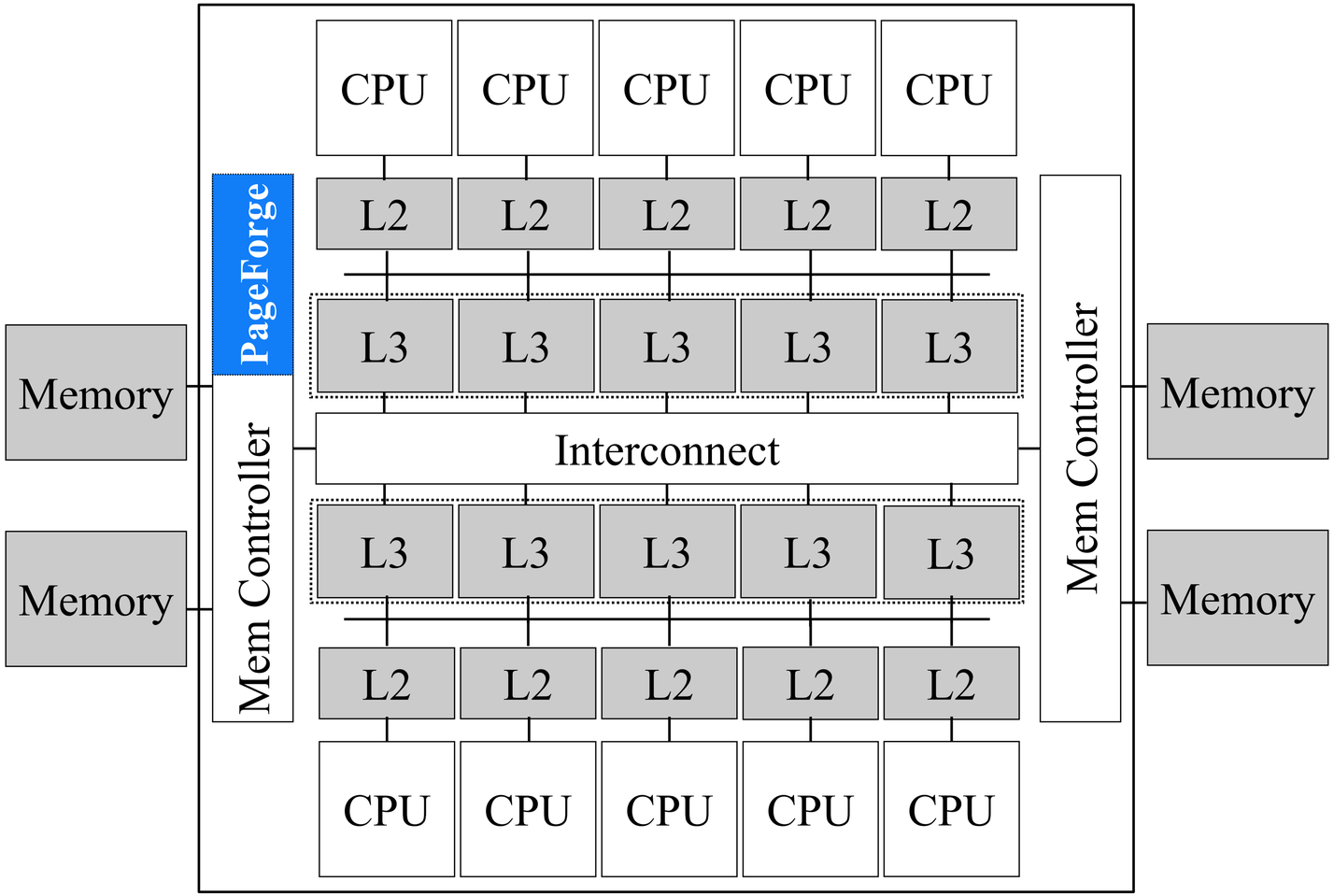
- Optimization: Identify if the candidate page has recently changed
- If a page changes too often → Not a good candidate



PageForge: Hardware Assisted Page-Merging

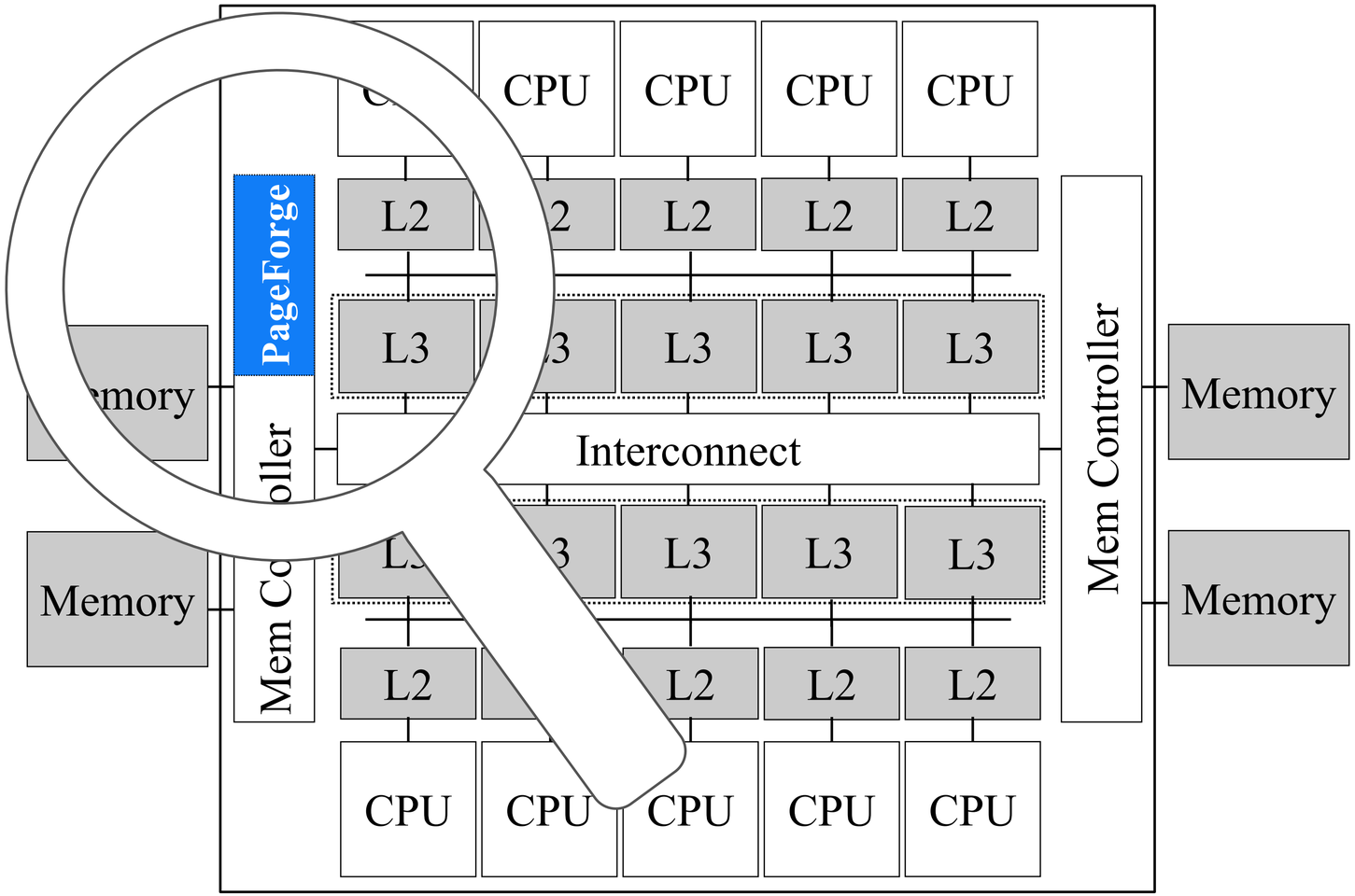
PageForge Main Idea

- Hardware engine in the memory controller
  - Compares pages
  - Generates hashes
- Advantages:
  - ✓ No core utilization
  - ✓ No cache pollution

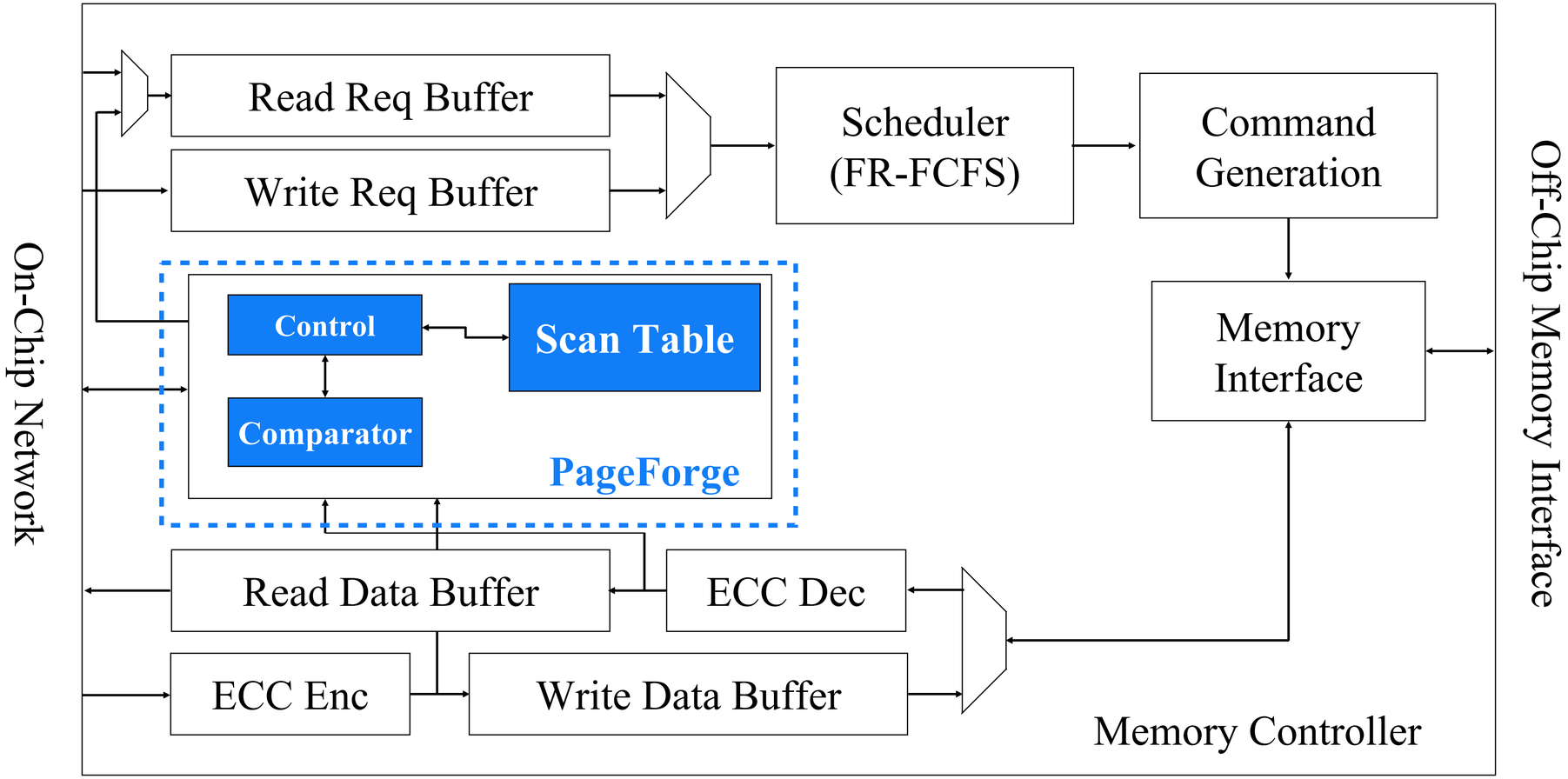


PageForge Main Idea

- Hardware engine in the memory controller
  - Compares pages
  - Generates hashes
- Advantages:
  - ✓ No core utilization
  - ✓ No cache pollution



PageForge @ The Memory Controller



PageForge Operation

PageForge Operation

Operating System

- Loads in Scan Table:
  - Candidate page PPN
  - PPNs of pages to compare to candidate

PageForge Operation

Operating System

- Loads in Scan Table:
  - Candidate page PPN
  - PPNs of pages to compare to candidate

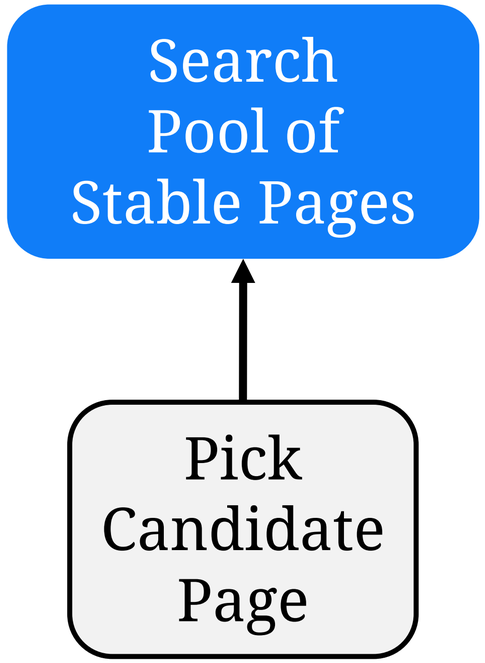
PageForge Hardware

- Autonomously performs :
  - Sequence of comparisons
  - Generation of the hash key for candidate page

PageForge Operation

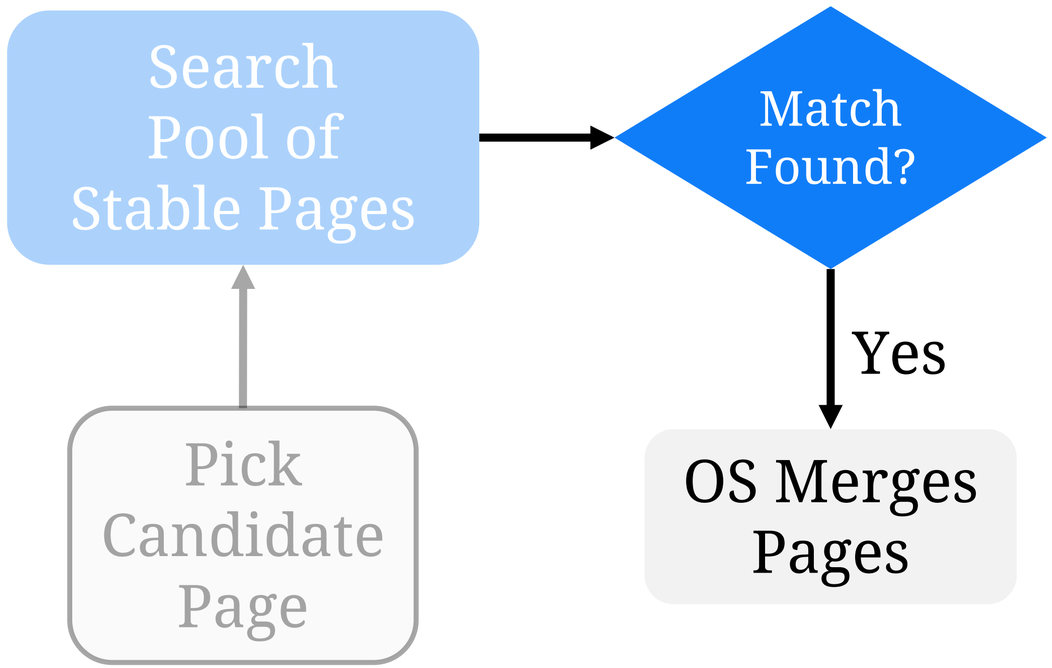
In Hardware

PageForge Operation



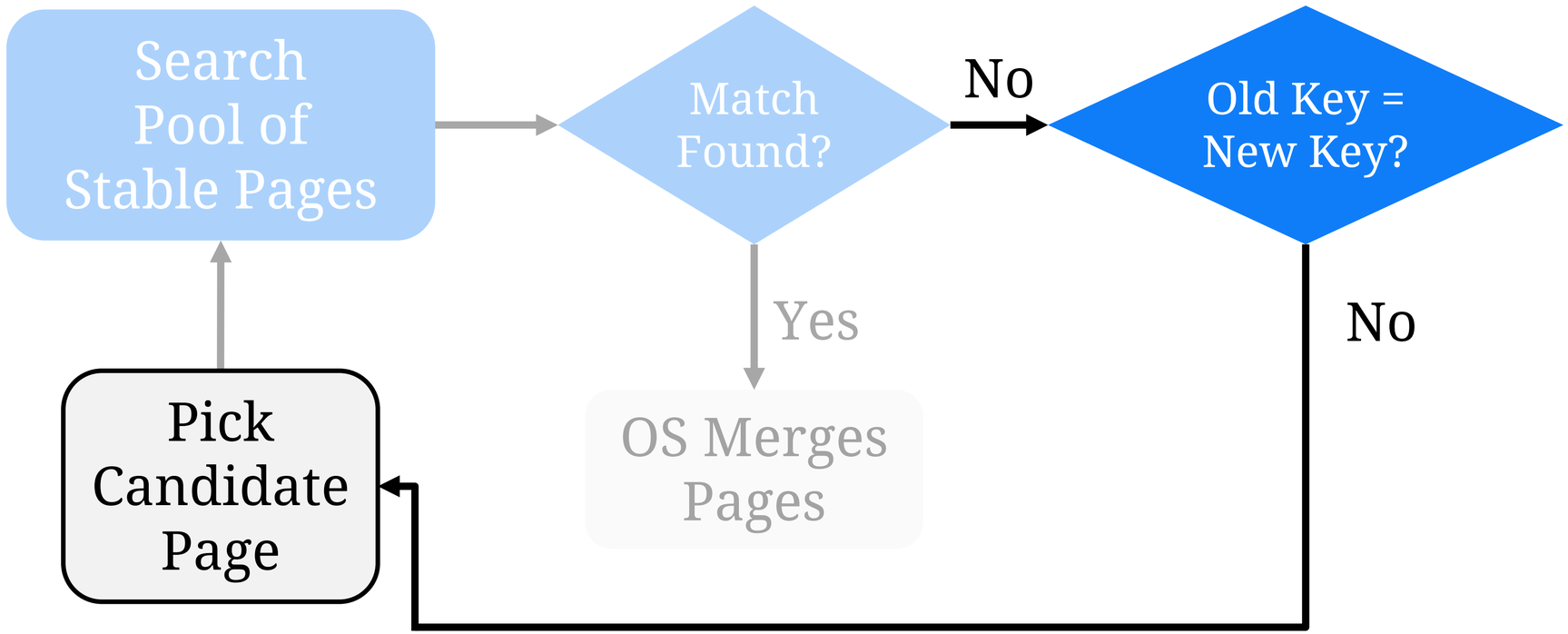
In Hardware

PageForge Operation



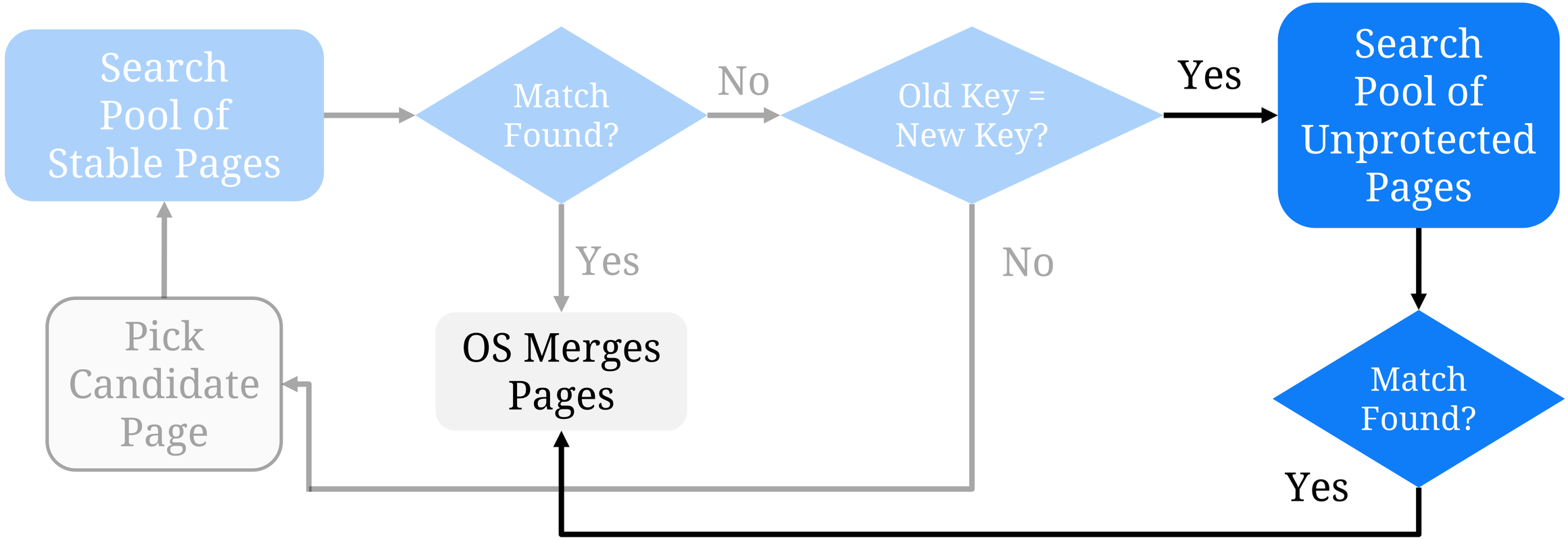
In Hardware

PageForge Operation



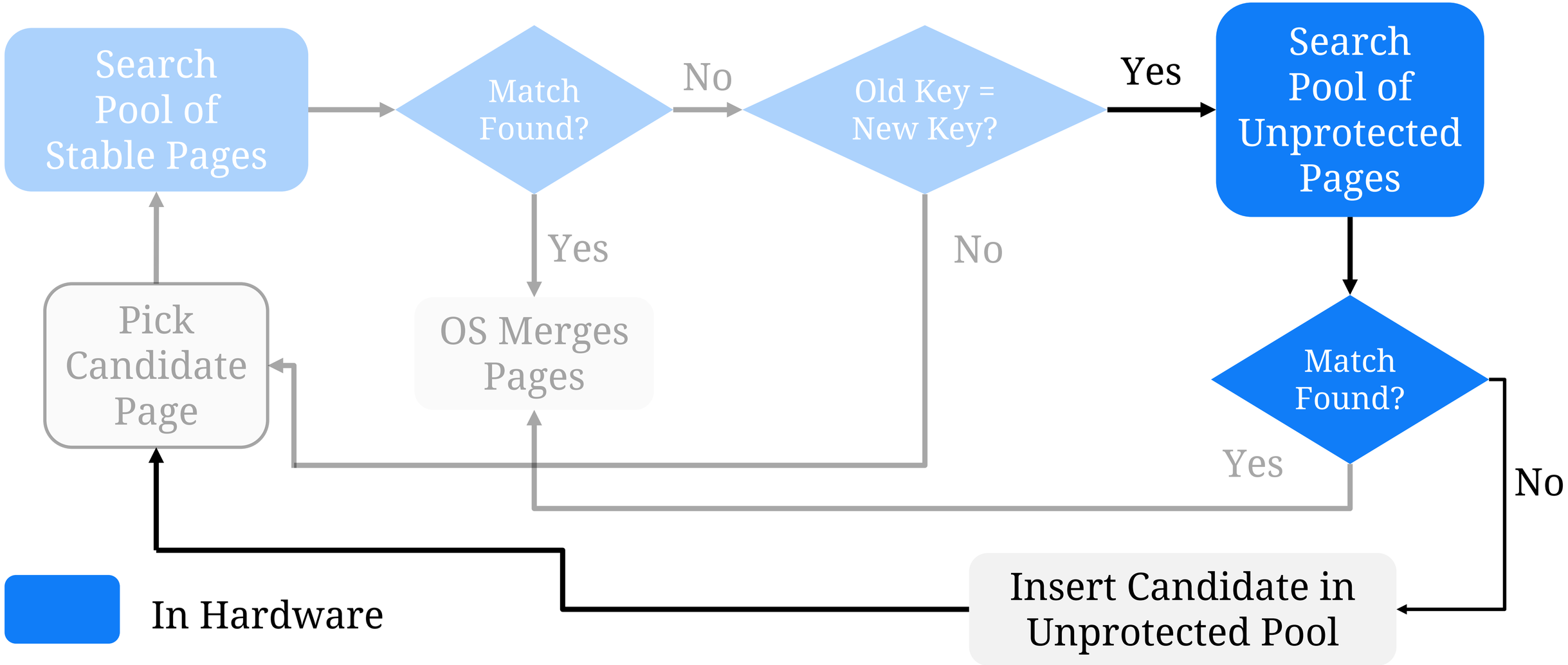
In Hardware

PageForge Operation

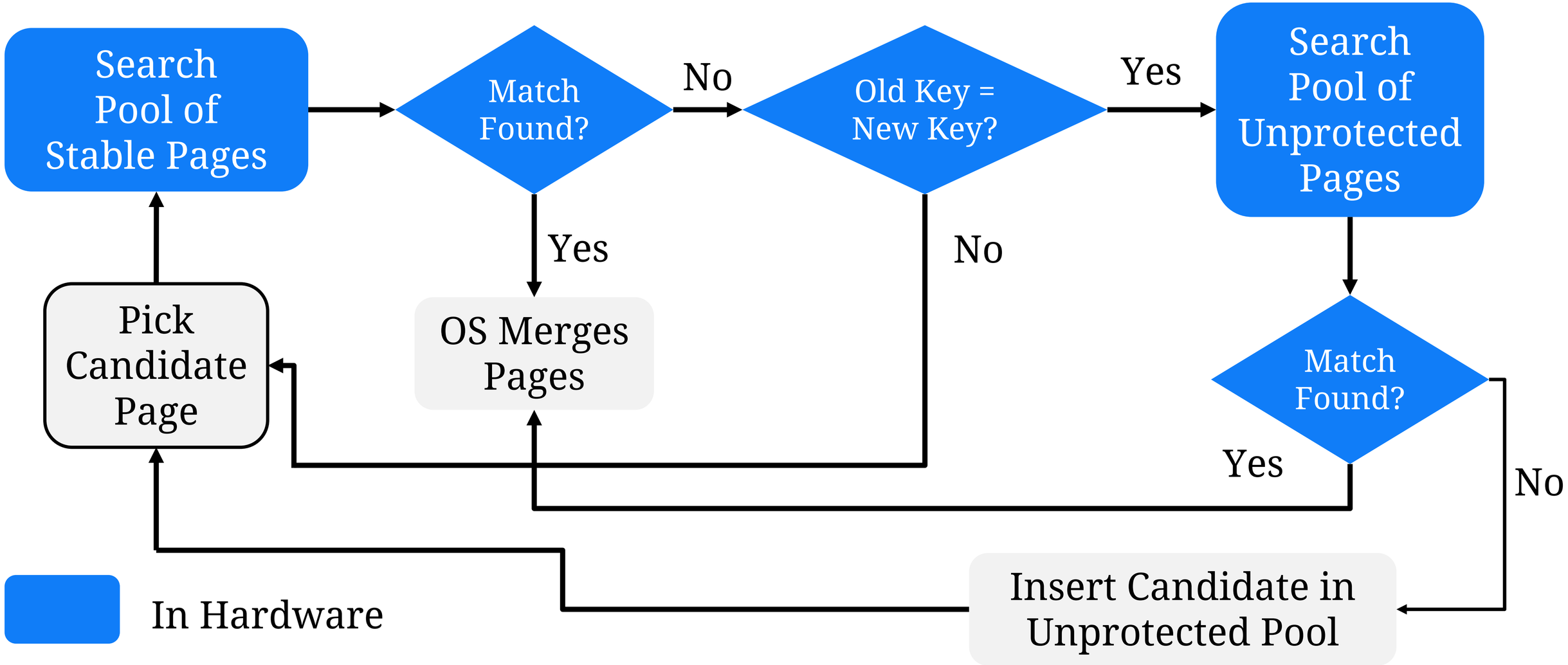


In Hardware

PageForge Operation



PageForge Operation



Eliminating The Cost of Hash Keys

Hash Key for Page Content Characterization

Hash Key for Page Content Characterization

- If page changed recently, key *may* be different

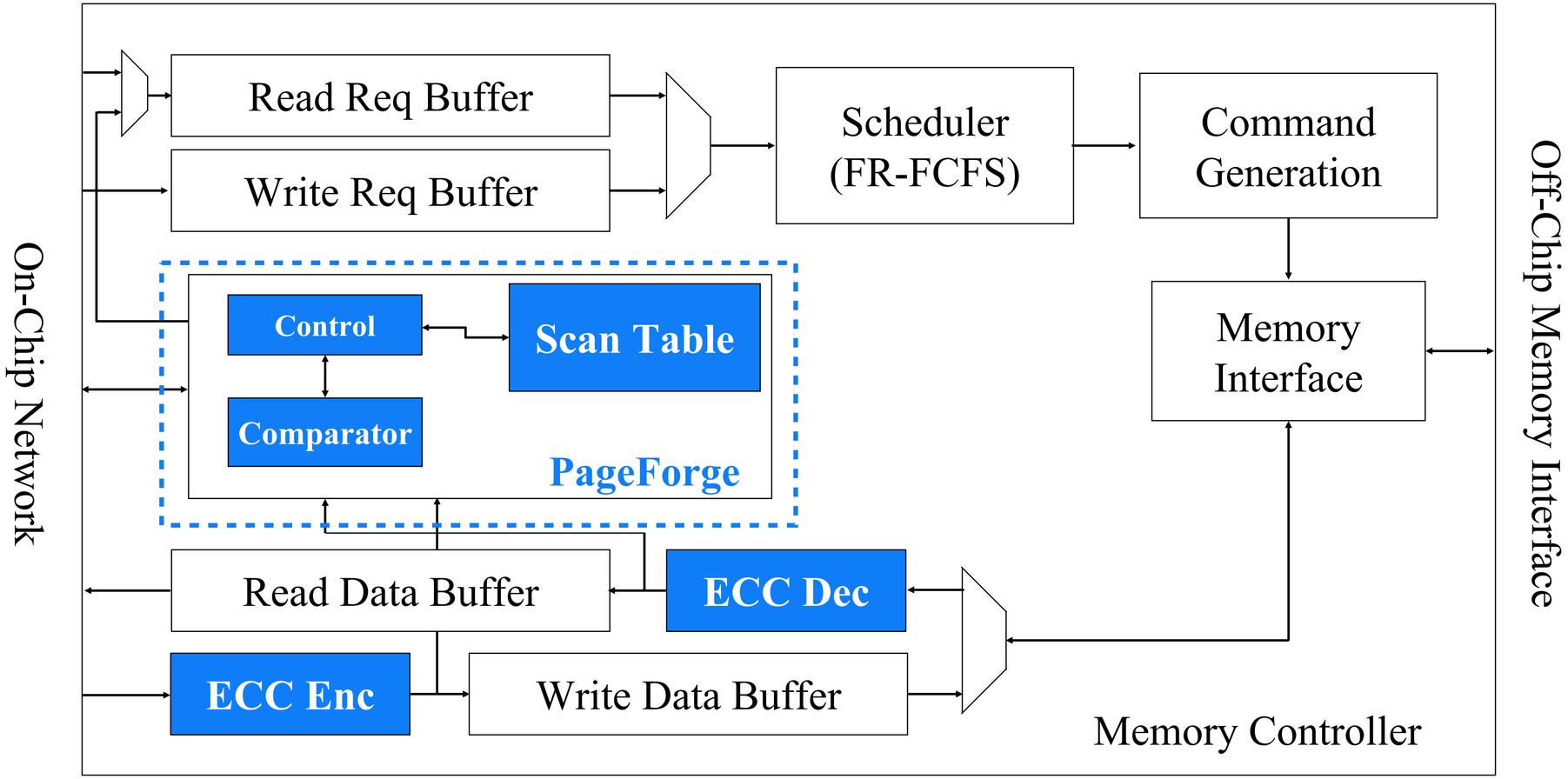
Hash Key for Page Content Characterization

- If page changed recently, key *may* be different
- KSM: JHash → Serial computation + 1KB of data

Hash Key for Page Content Characterization

- If page changed recently, key *may* be different
- KSM: JHash → Serial computation + 1KB of data
- PageForge: ECC → Parallel computation + 256B of data

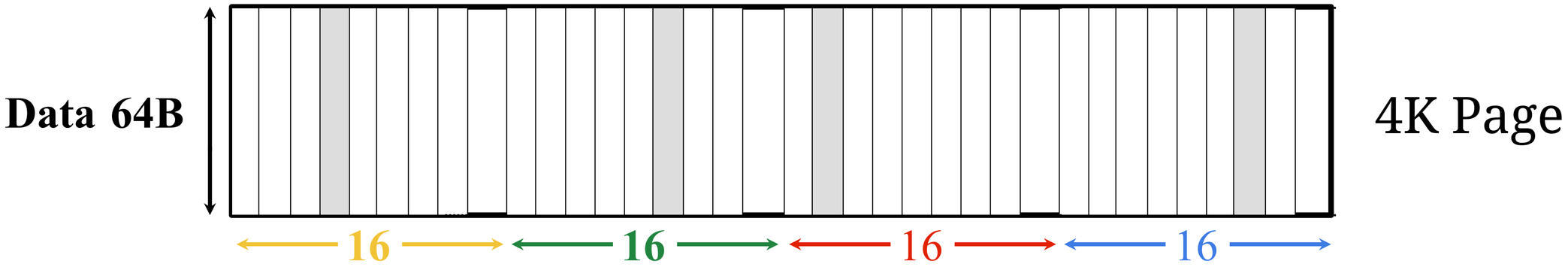
PageForge @ The Memory Controller



Proposal: ECC for Page Content Characterization

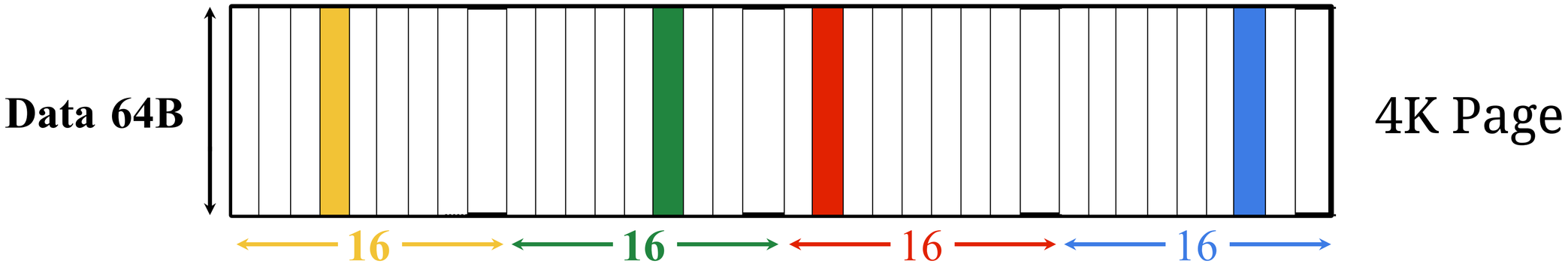
Proposal: ECC for Page Content Characterization

- Break a 4KB page into 4 segments



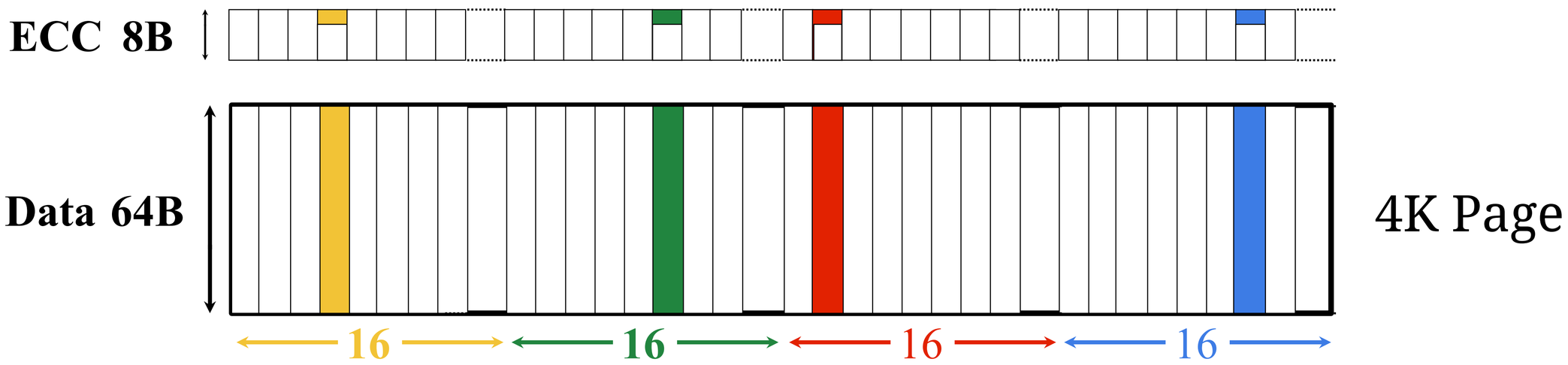
Proposal: ECC for Page Content Characterization

- Break a 4KB page into 4 segments
- Pick a random cache line within each segment



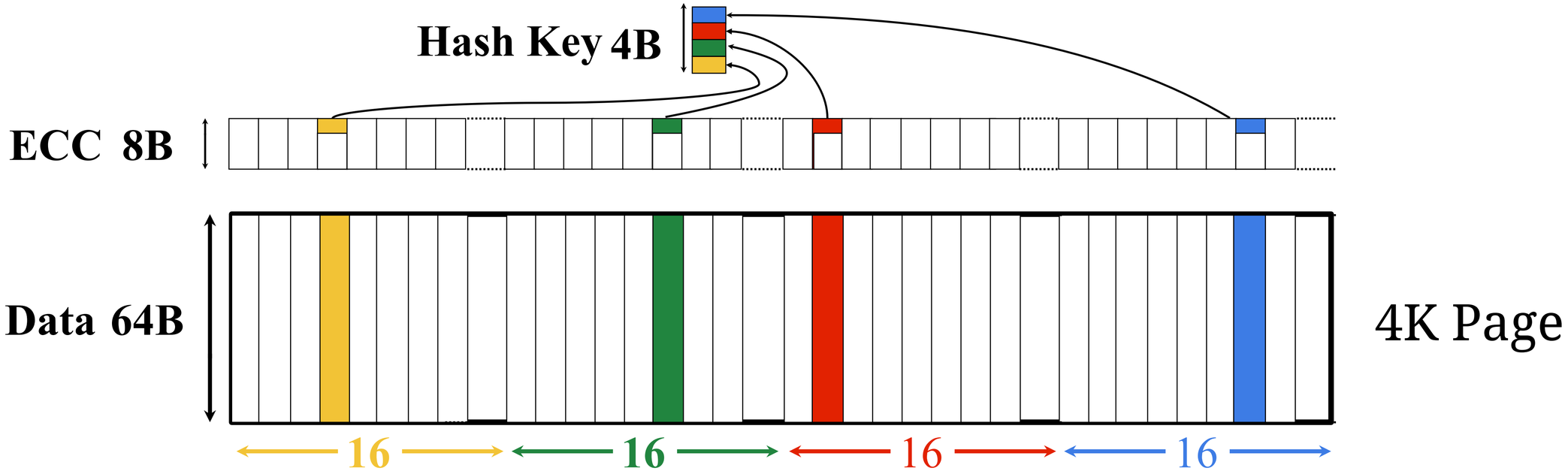
Proposal: ECC for Page Content Characterization

- Break a 4KB page into 4 segments
- Pick a random cache line within each segment
- Select the 8-bit ECC of the least significant QWORD



Proposal: ECC for Page Content Characterization

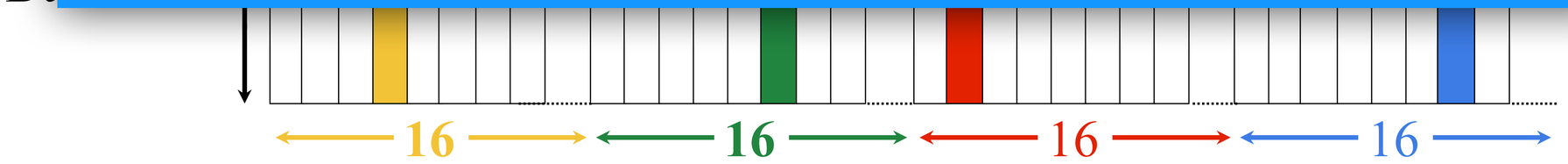
- Break a 4KB page into 4 segments
- Pick a random cache line within each segment
- Select the 8-bit ECC of the least significant QWORD



Proposal: ECC for Page Content Characterization

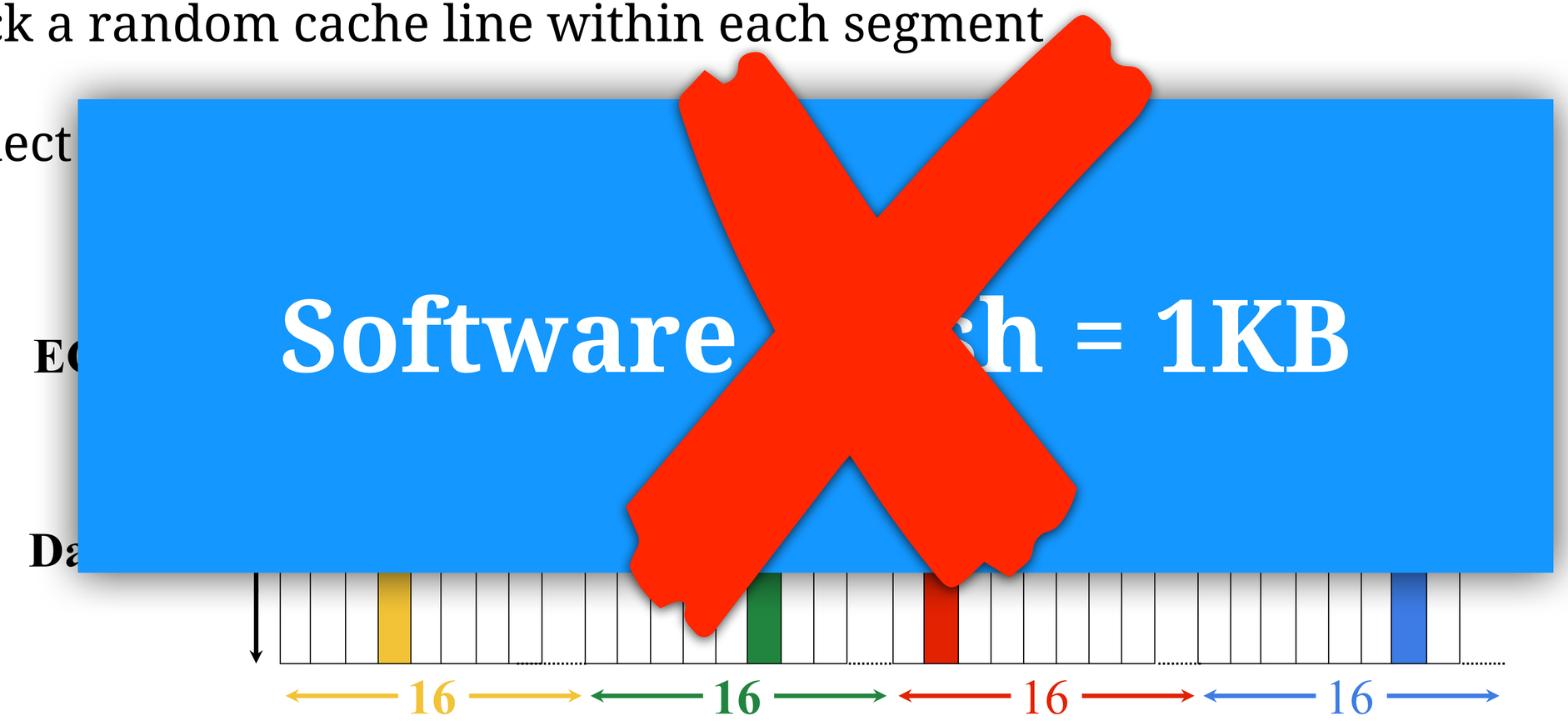
- Break a 4KB page into 4 segments
- Pick a random cache line within each segment
- Select

Software JHash = 1KB



Proposal: ECC for Page Content Characterization

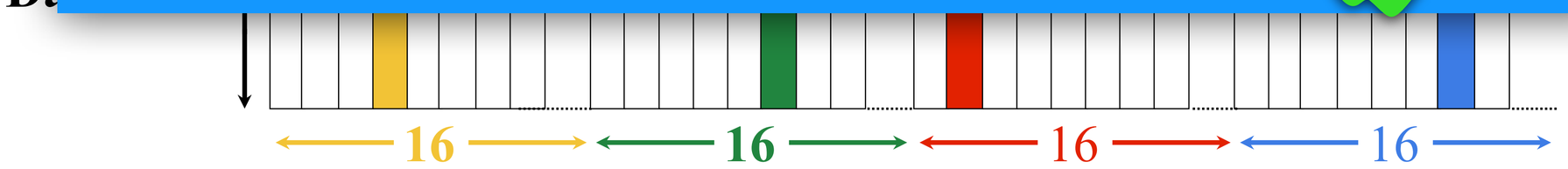
- Break a 4KB page into 4 segments
- Pick a random cache line within each segment
- Select



Proposal: ECC for Page Content Characterization

- Break a 4KB page into 4 segments
- Pick a random cache line within each segment
- Select

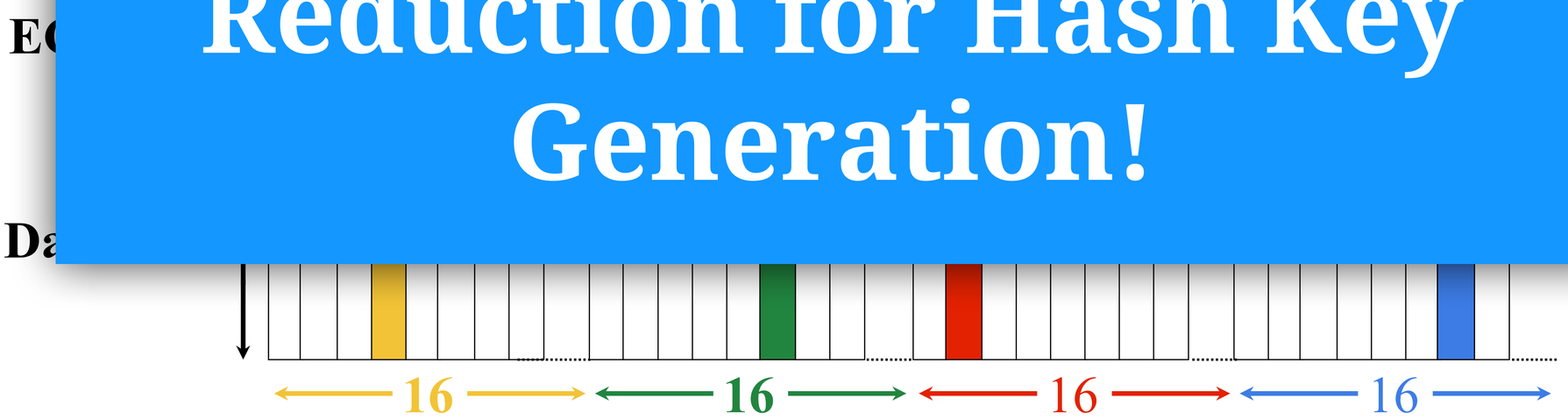
ECC Hash = 256 Bytes



Proposal: ECC for Page Content Characterization

- Break a 4KB page into 4 segments
- Pick a random cache line within each segment
- Select

75% Memory Footprint Reduction for Hash Key Generation!



Evaluation

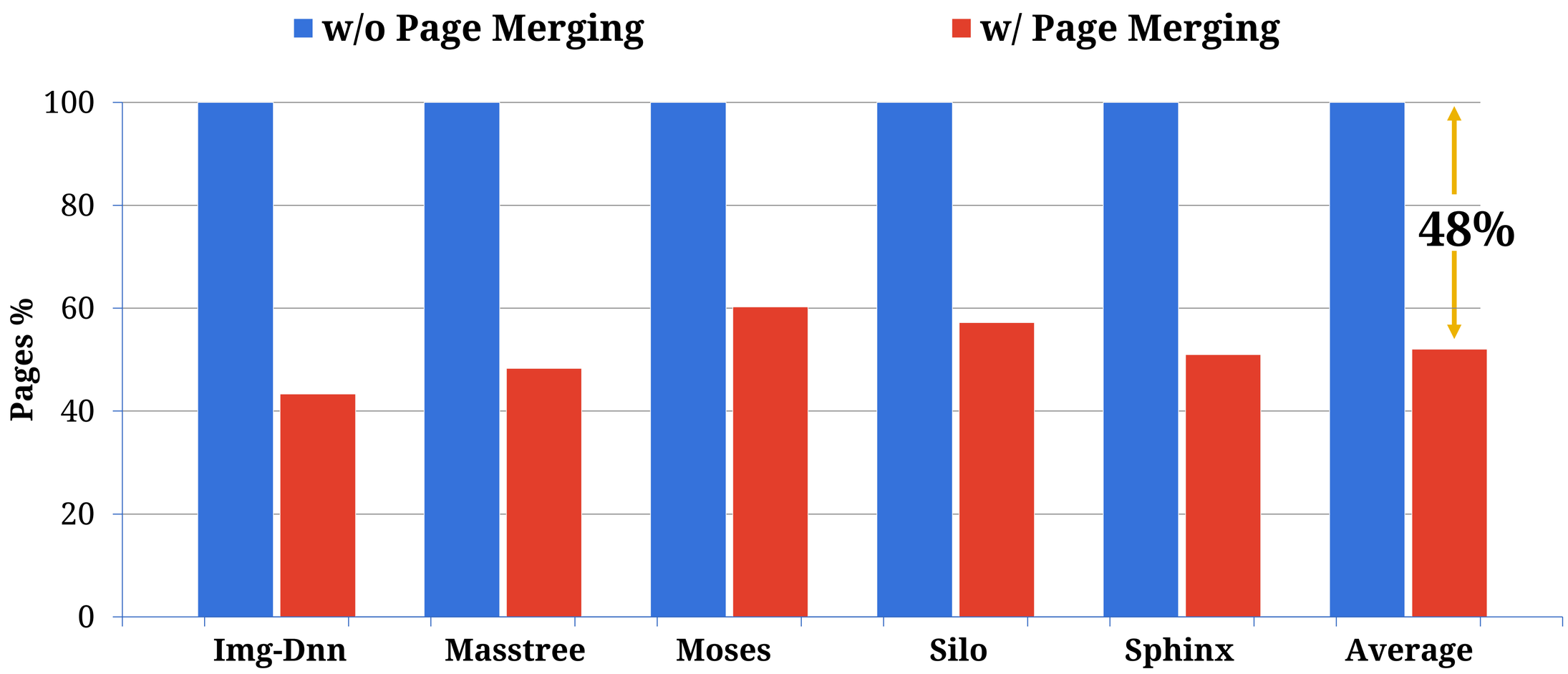
Simulation Setup

- 2GHz, 10-core, 16GB DRAM
- One VM per core
- Ubuntu 16.04 Host, Ubuntu Cloud Guest
- InHouse simulator: Simics + SST + DRAMSim2
- TailBench Suite benchmarks

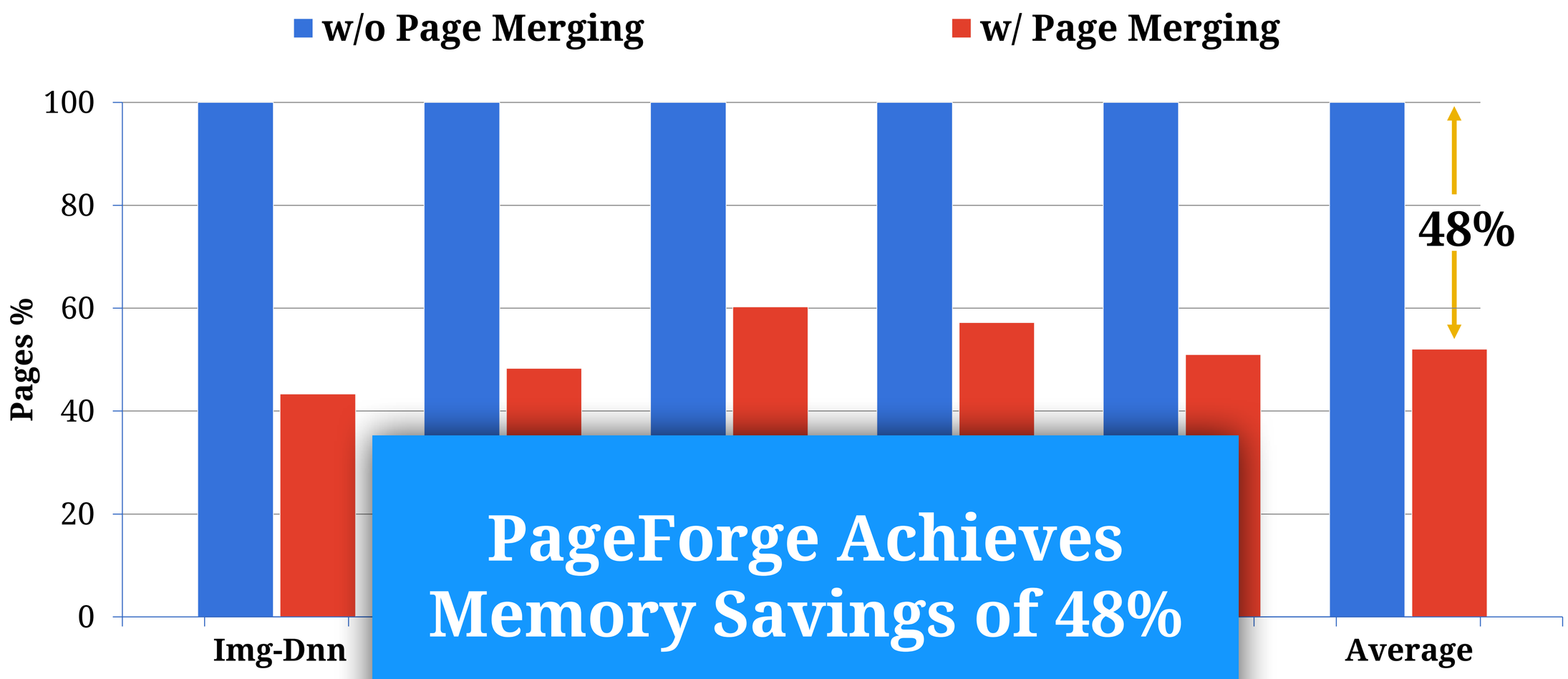
Configurations

- ***Baseline***: Page-merging is disabled
- ***KSM***: RedHat's implementations shipped with the current Linux
- ***PageForge***: Hardware-assisted page-merging

Memory Savings – w/o vs w/ Page Merging

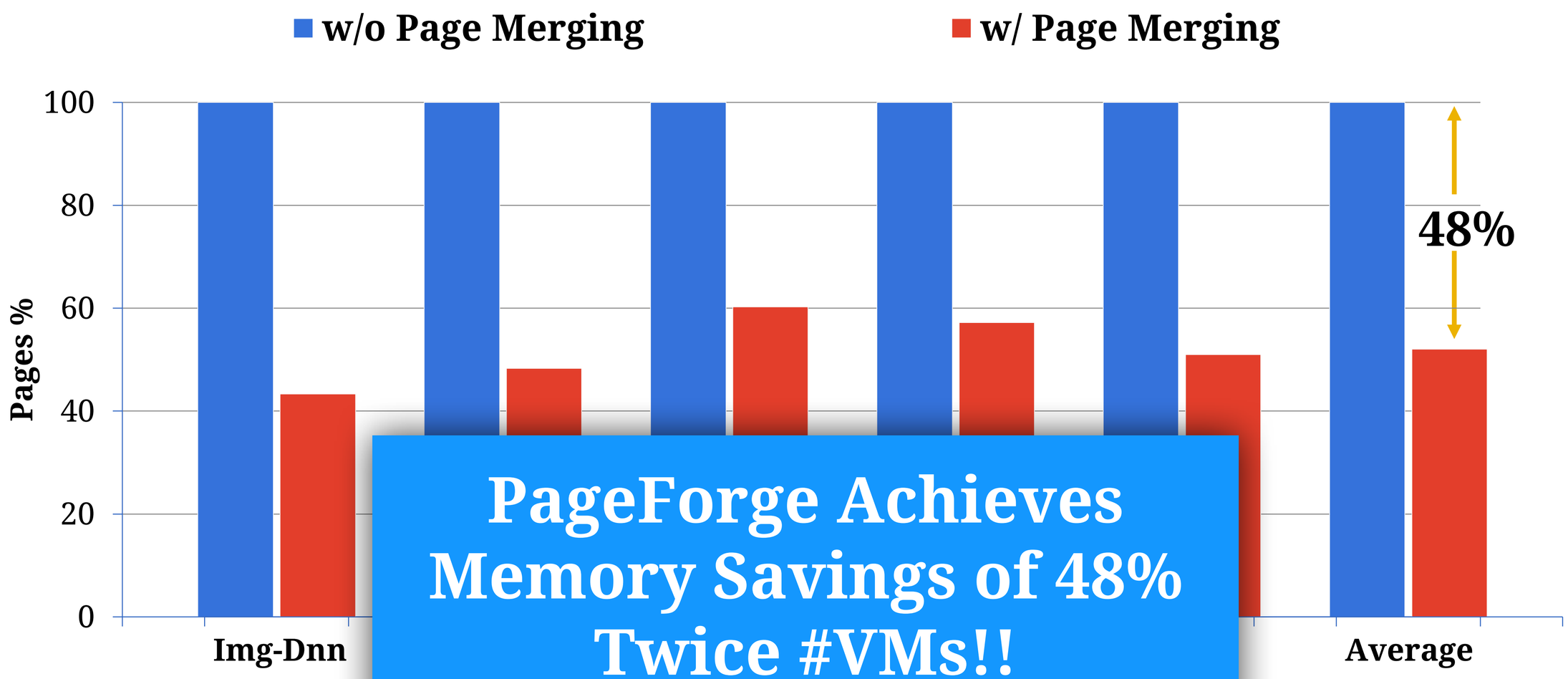


Memory Savings – w/o vs w/ Page Merging



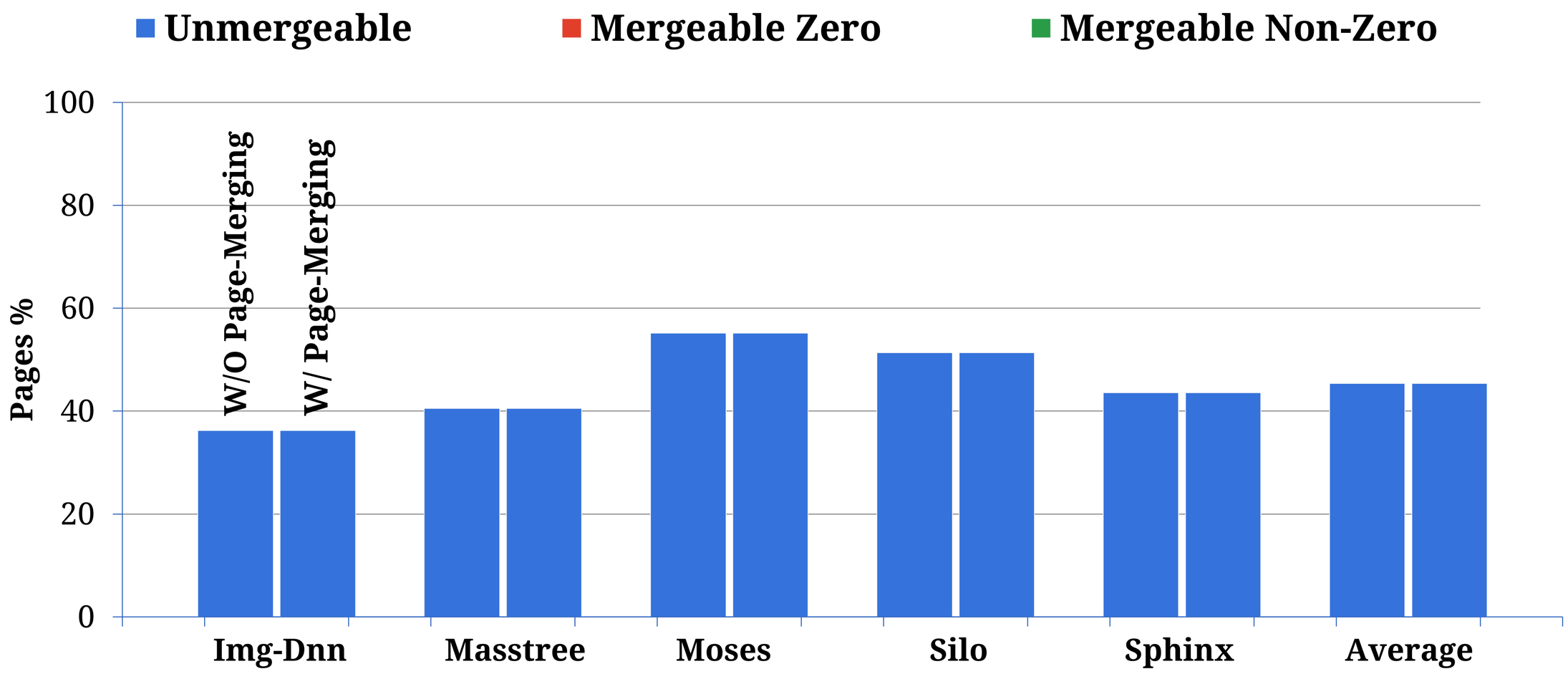
PageForge Achieves Memory Savings of 48%

Memory Savings – w/o vs w/ Page Merging

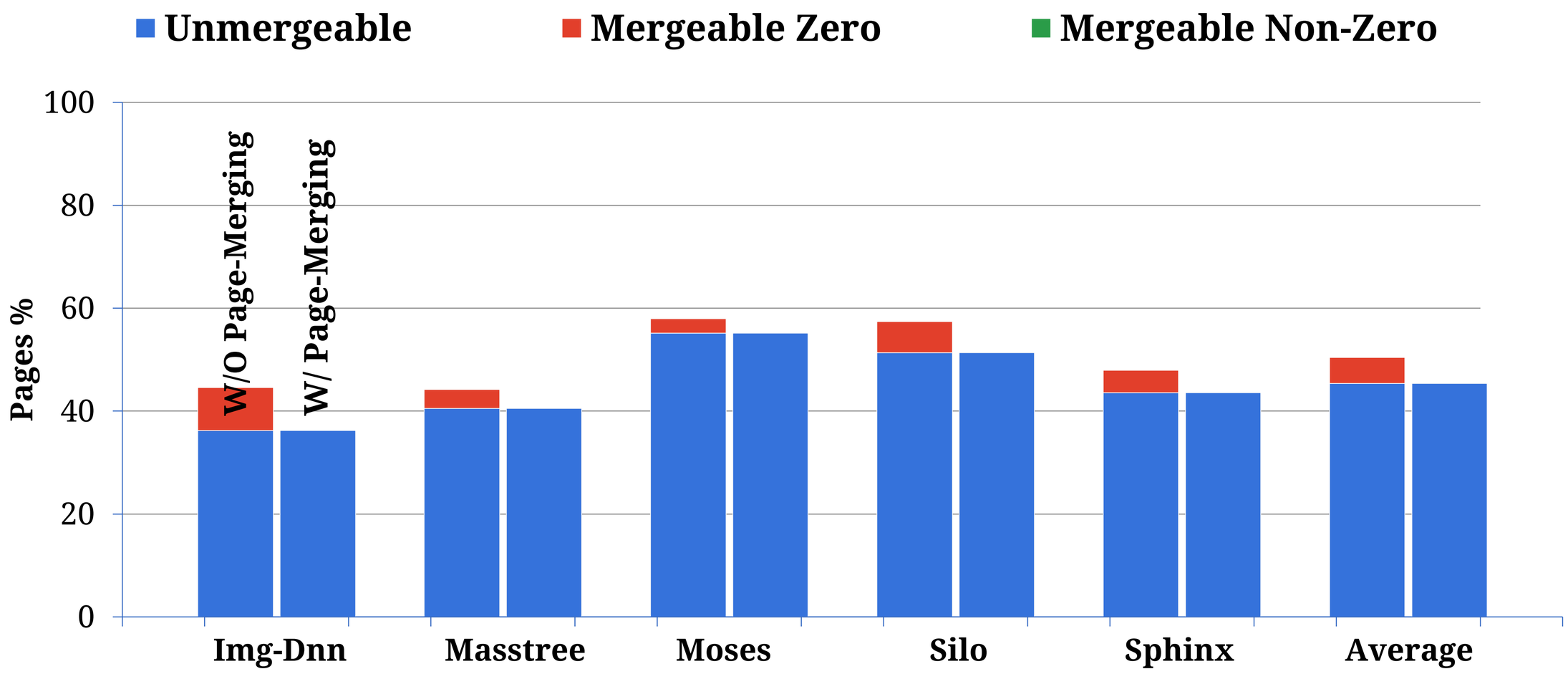


PageForge Achieves Memory Savings of 48% Twice #VMs!!

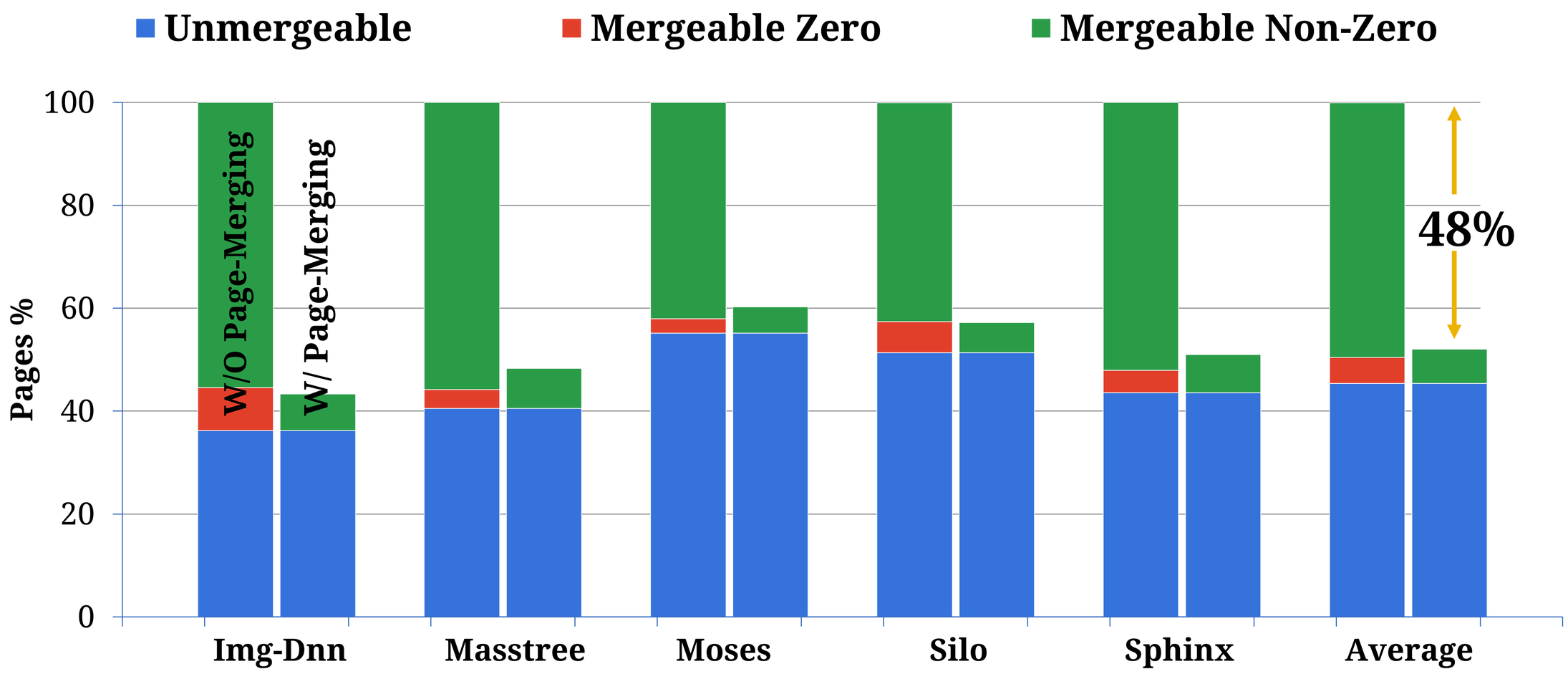
Memory Savings – w/o vs w/ Page Merging



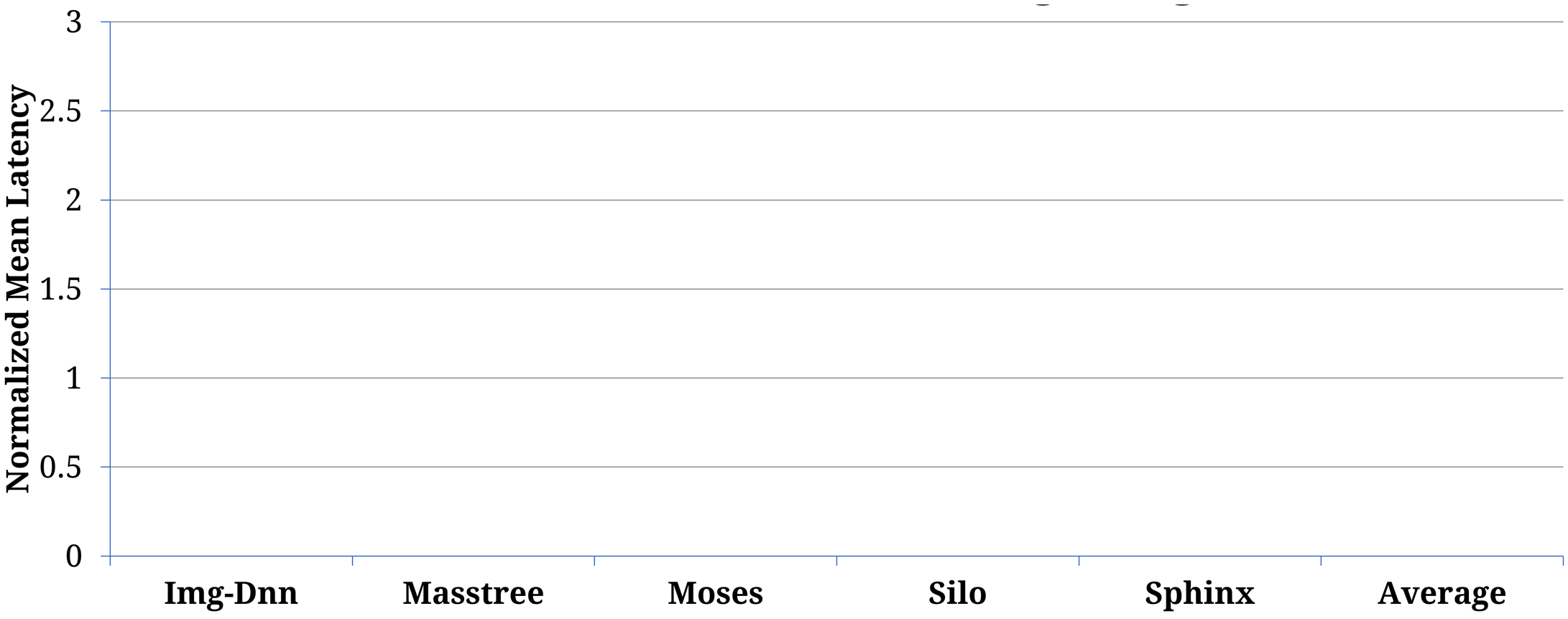
Memory Savings – w/o vs w/ Page Merging



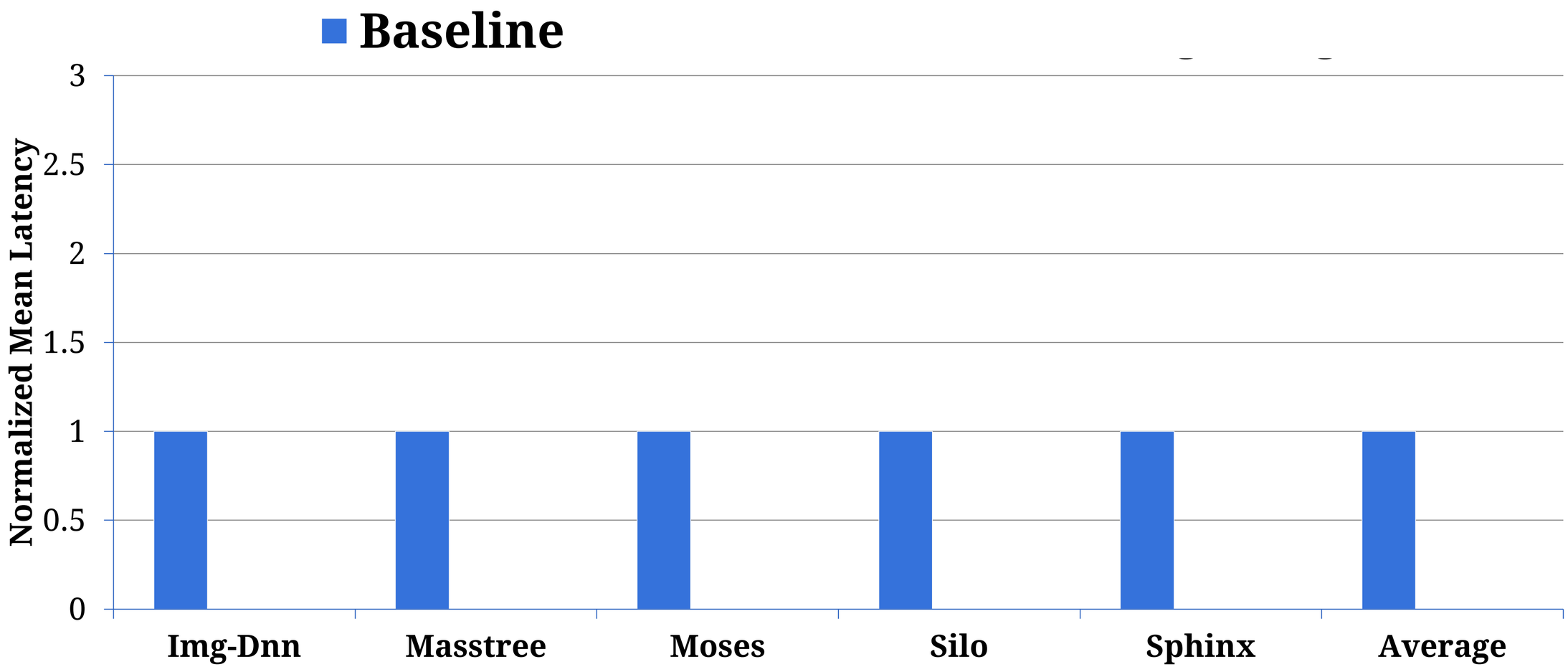
Memory Savings – w/o vs w/ Page Merging



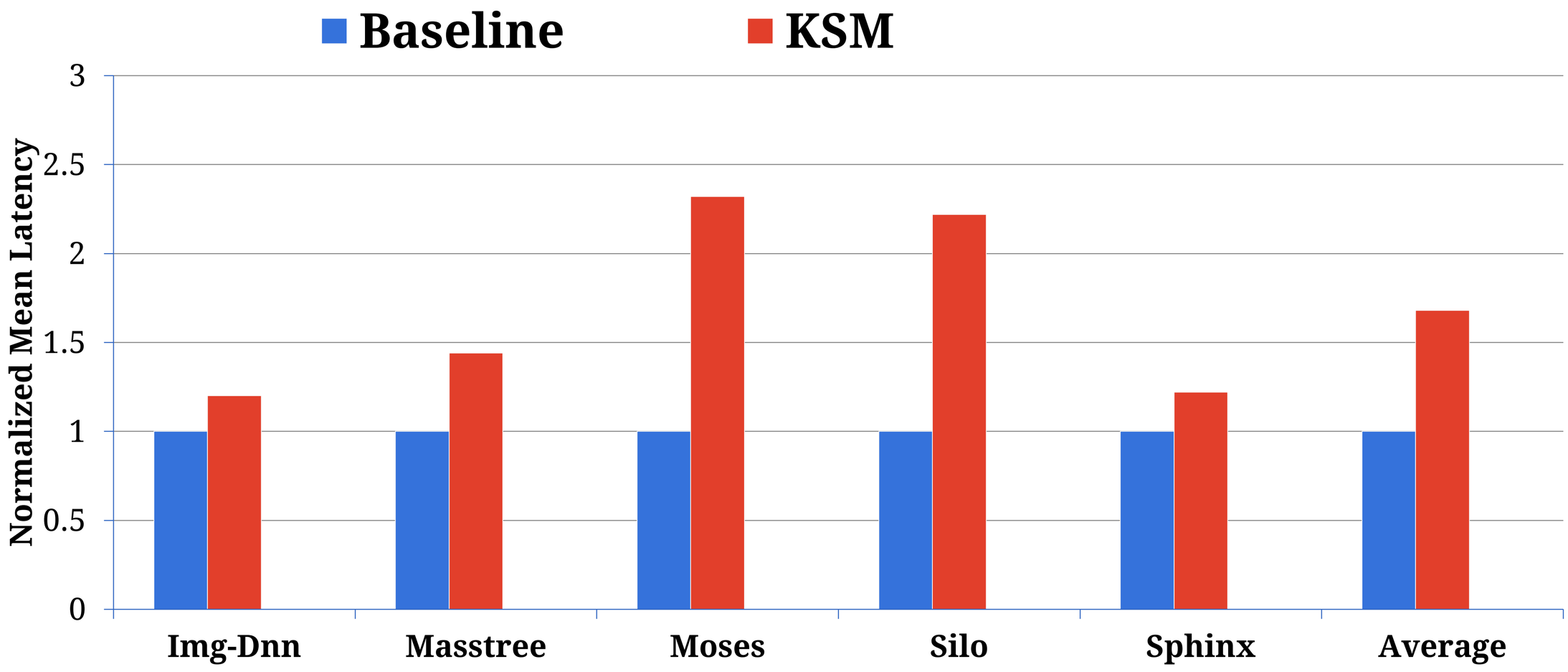
Mean Latency of Requests



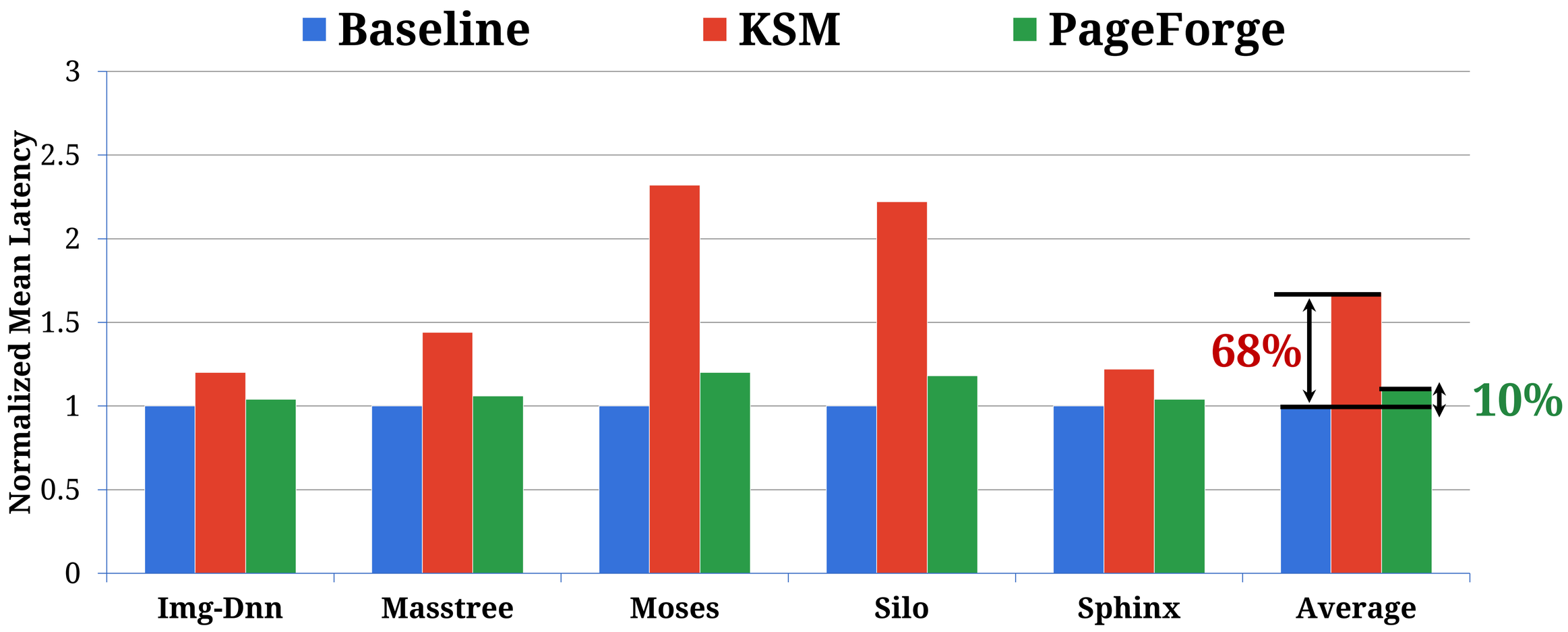
Mean Latency of Requests



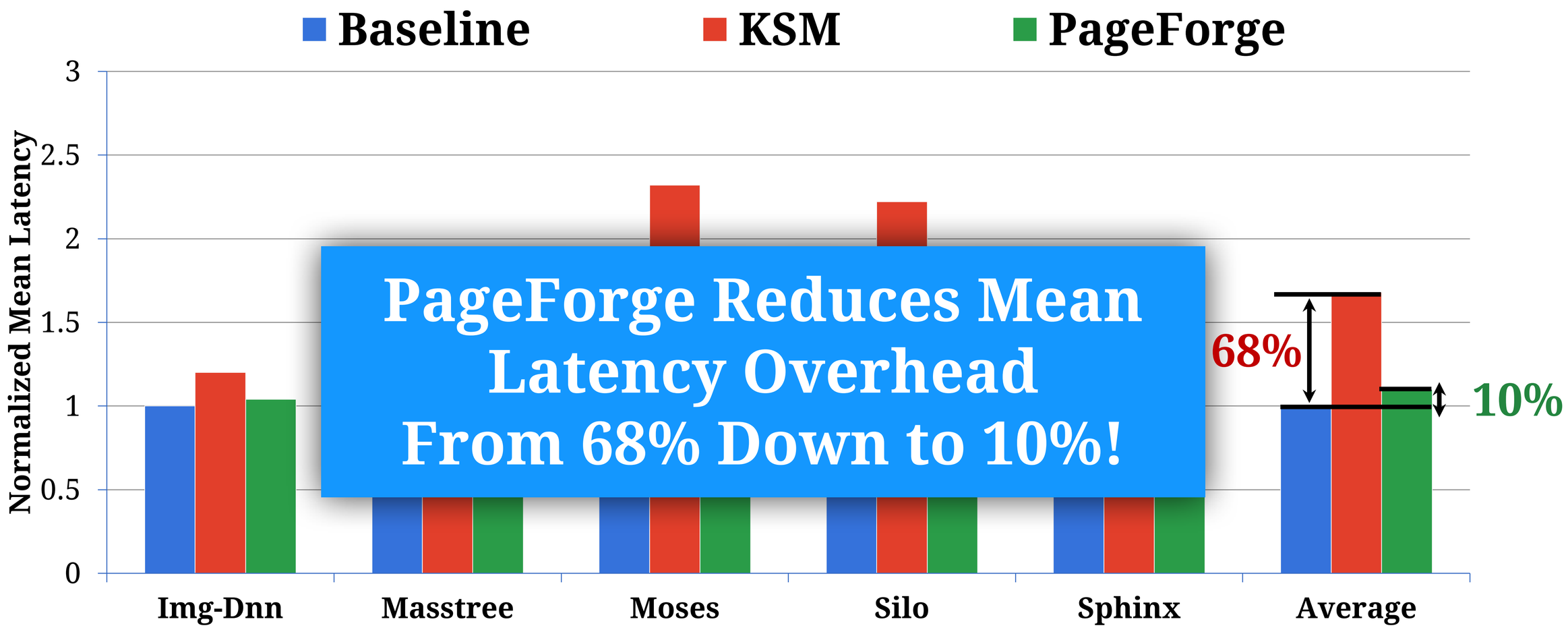
Mean Latency of Requests



Mean Latency of Requests

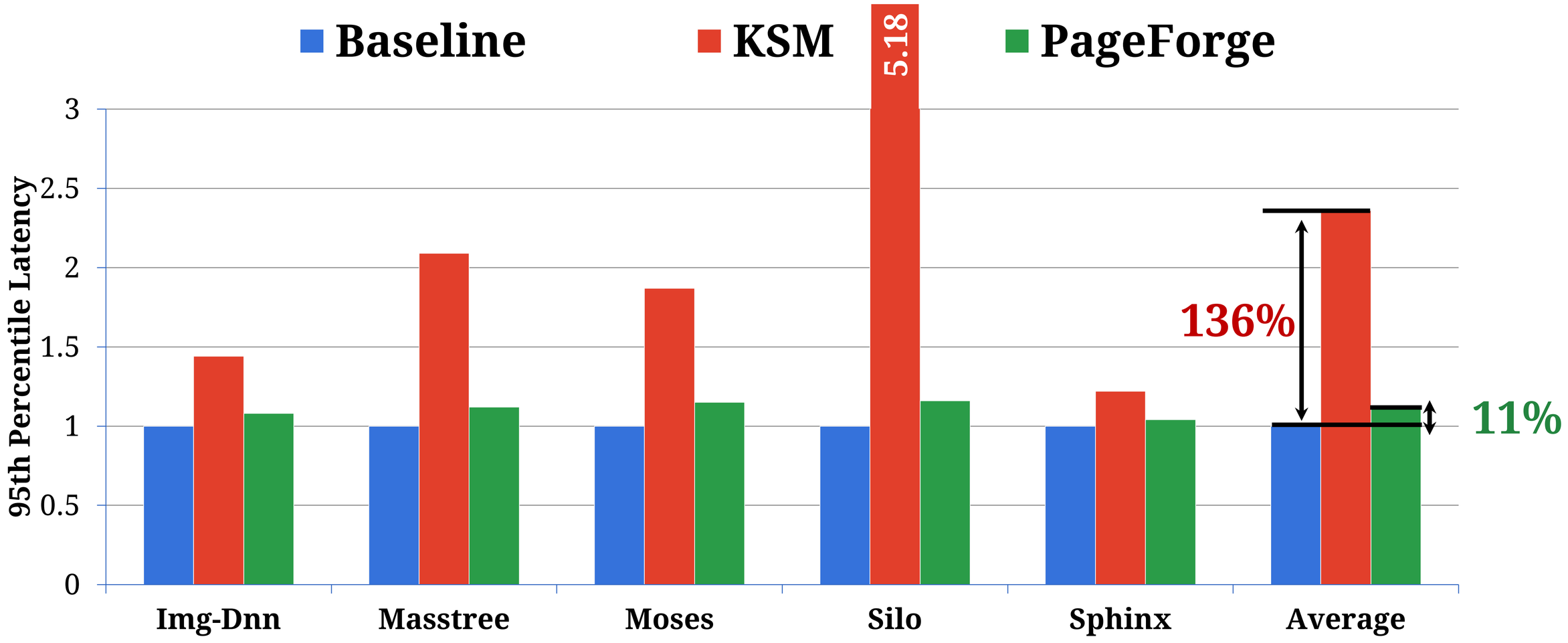


Mean Latency of Requests

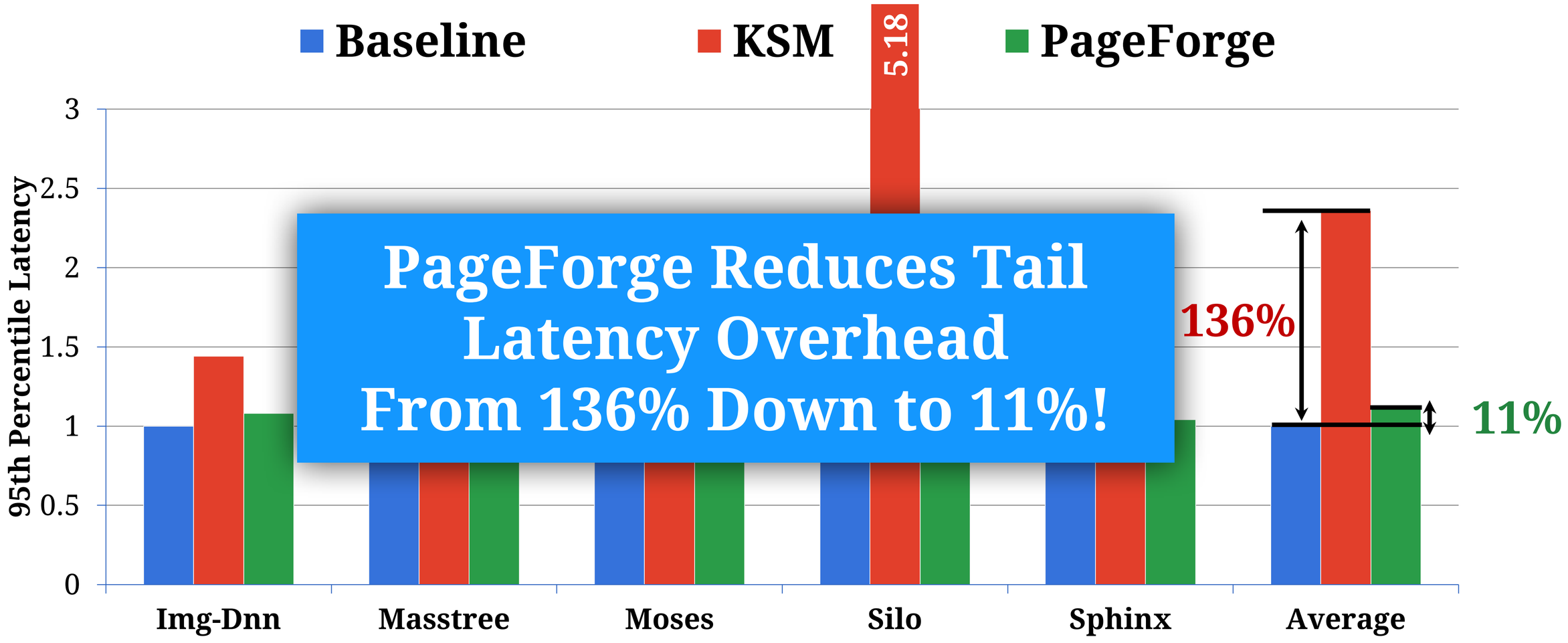


PageForge Reduces Mean Latency Overhead From 68% Down to 10%!

Tail Latency of Requests



Tail Latency of Requests



PageForge Reduces Tail Latency Overhead From 136% Down to 11%!

Also in the Paper

- Software interface to PageForge
- Interaction with the cache coherence protocol
- Alternative designs
- Bandwidth analysis
- ECC vs Jhash → ECC keys add negligible collisions
- Power and area at 22nm is negligible

Takeaway: **PageForge**

- First solution for hardware-assisted content-aware page merging
- General, effective, minimal hypervisor involvement & hardware mods
- Reduced overhead vs state-of-the-art software:
  - Mean latency 68% → 10%
  - Tail latency 136% → 11%
- Same memory savings as software: 48% → Twice #VMs
- Novel use of ECC for content characterization → 75% less memory footprint