



Summary

- Interference in shared resources (e.g., LLC) harms performance (up to 94% overhead)
 - Mitigation: **page coloring** reduces interference (down to 16%)
- Increasing address translation overheads due to page table walks
 - Mitigation: **huge pages** improve TLB performance (by up to 42%)
- Problem:** page coloring and huge pages are fundamentally incompatible!

Colored Huge Pages

- HW/SW approach to combine both techniques efficiently and robustly
- Enhances performance through LLC isolation & accelerated address translation

Our proposal

1 Background and Motivation

Page Coloring

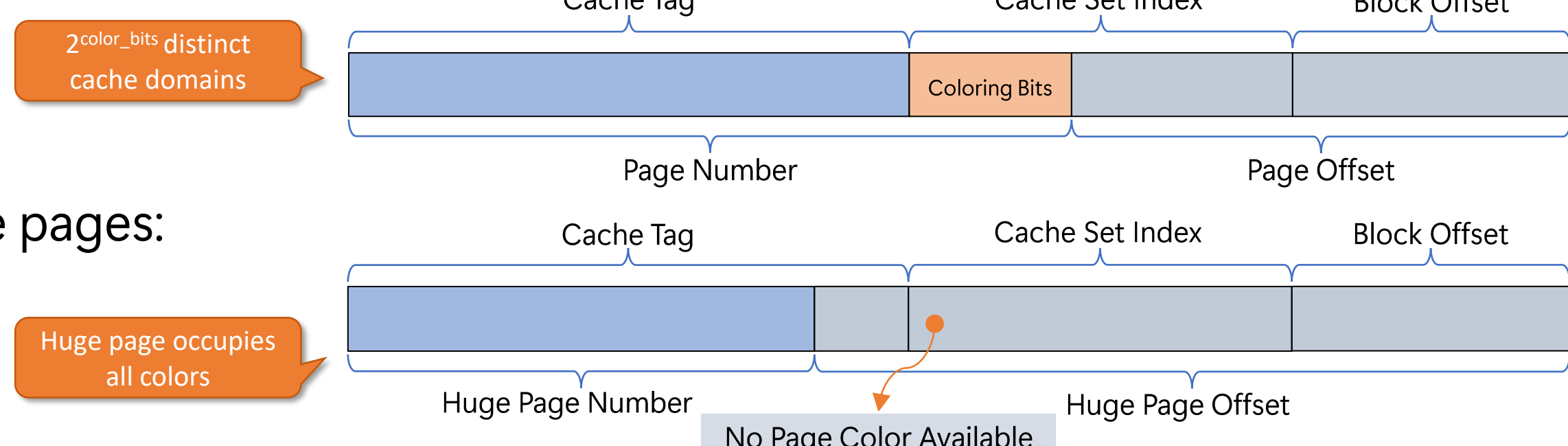
- Software approach for partitioning shared hardware components
- Bits of PFN can influence which portions of a shared resource a process can access
⇒ OS steers frame allocation to achieve partitioning

Huge Pages

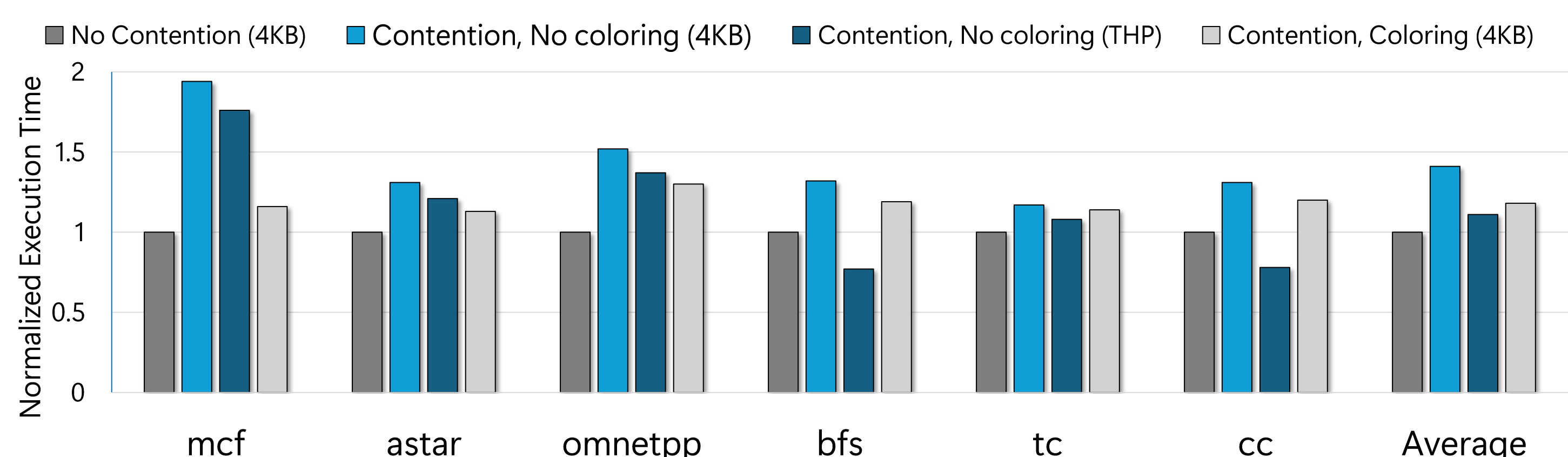
- HW/SW support for improving virtual memory through increased TLB reach
⇒ OS allocates physical memory at larger granularity

Incompatibility

- LLC partitioning



Performance Effect



LLC contention: 1.41x slowdown Huge Pages: 1.11x slowdown Page coloring: 1.18x slowdown

2 Related Work

1. Scattered Superpages [ICCD'13]

Breaks down physical huge page into individual base pages of the desired color

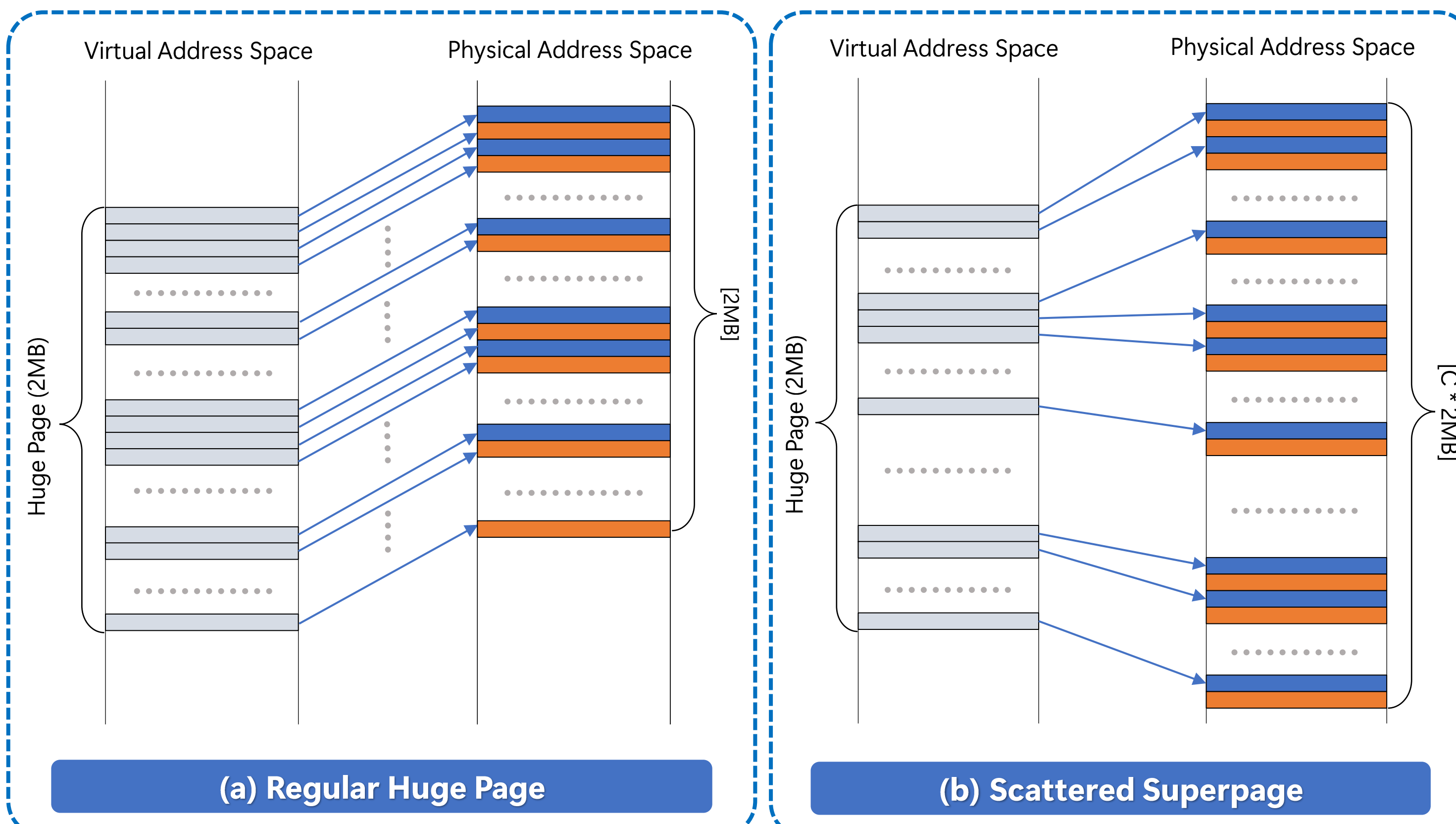
2. SWAP [DAC'14]

Swaps bits from huge page number with cache set index bits prior to LLC access

Approach	Fragmentation Robustness	No Extra LLC Conflict Misses	HW/SW Compatibility	OS Design & Evaluation
Scattered Superpages	✗	✓	✓	✗
SWAP	✓	✗	✗	✗
Colored Huge Pages (this work)	✓	✓	✓	✓

Scattered Superpages [ICCD'13]

Limitation: massive physical contiguity required (i.e., #colors **contiguous** huge pages per Scattered Superpage)

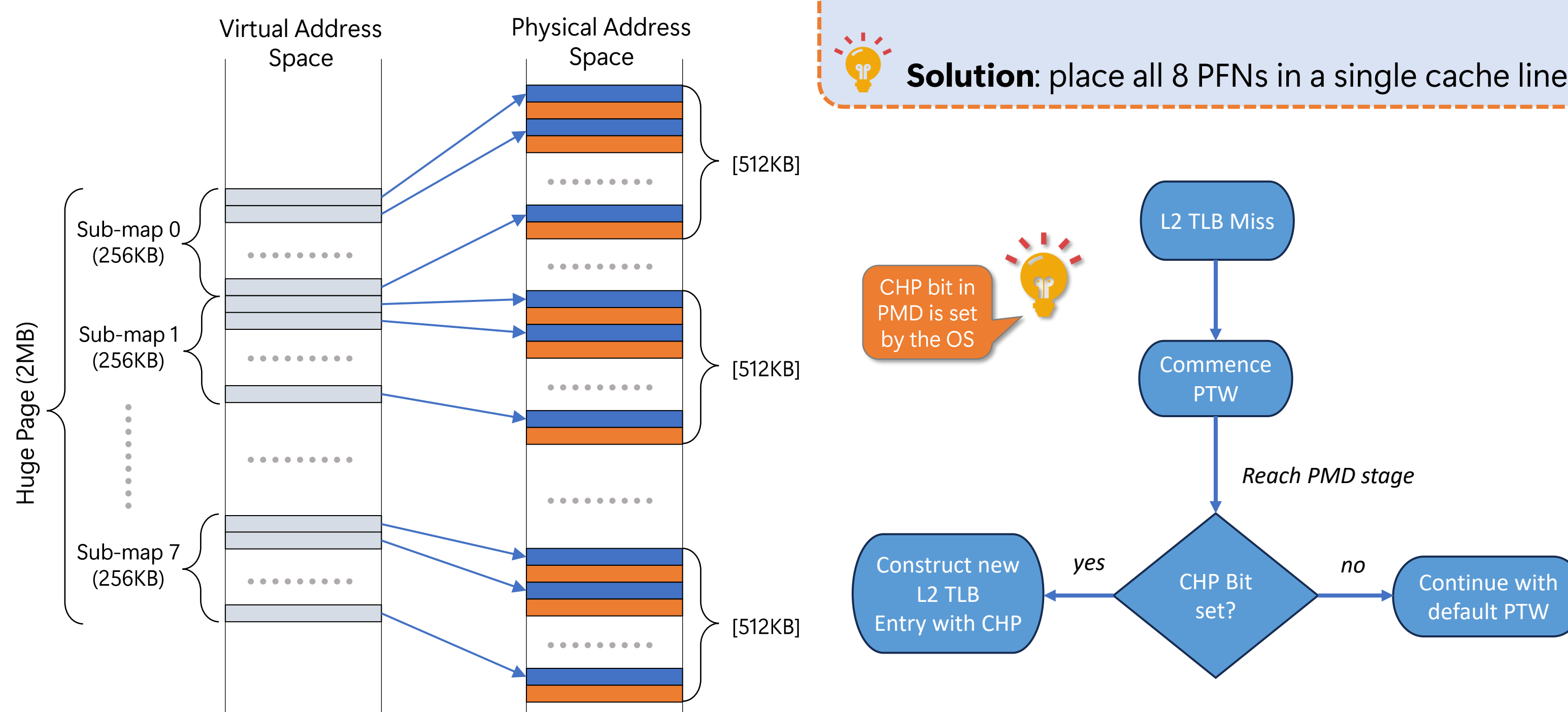


3 Colored Huge Pages

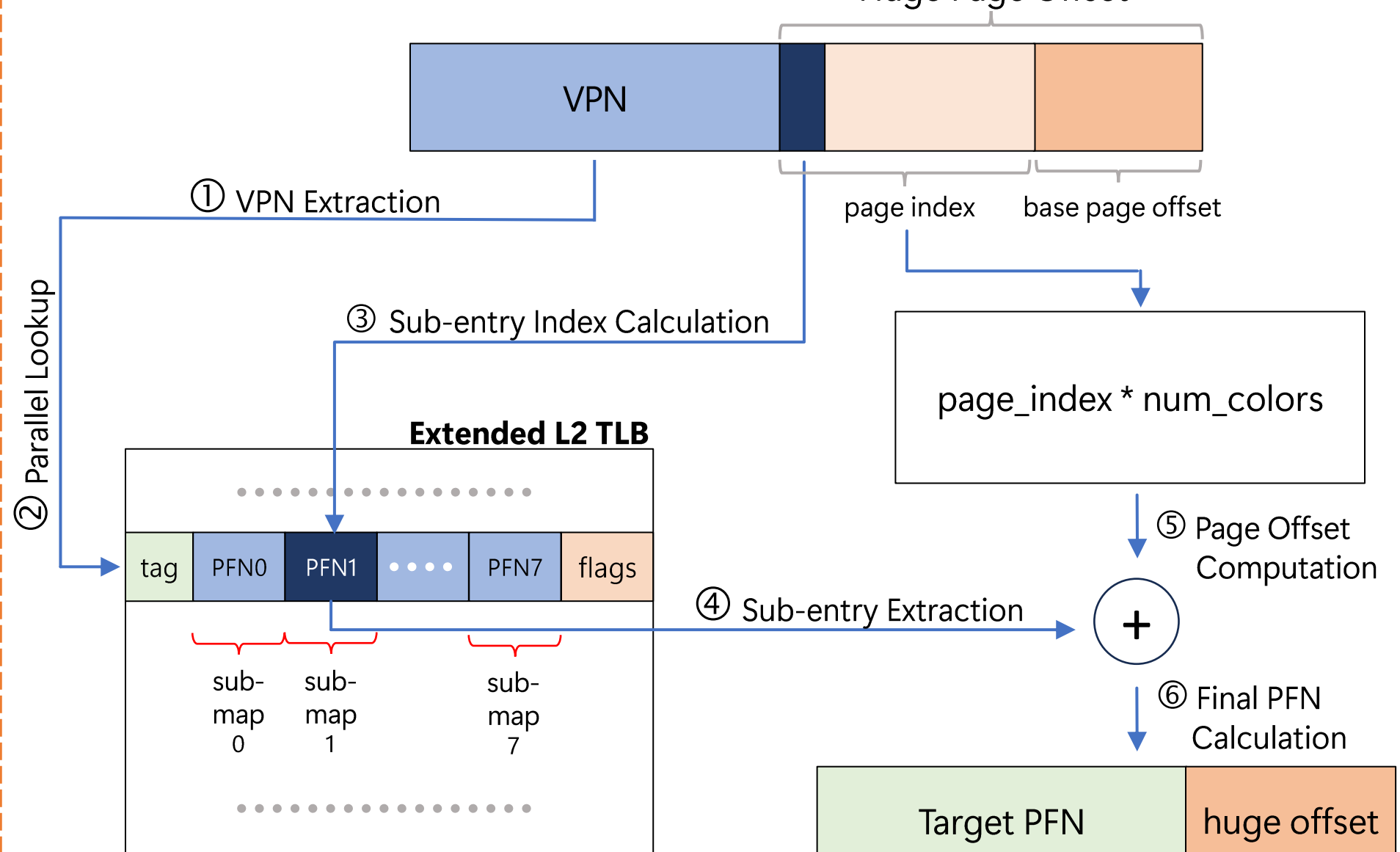
Key Idea

Lower contiguity requirements by allowing each 2MB strided mapping to break down into **multiple independent strided** sub-mappings

- Hardware:** efficient translation with simple modifications in L2 TLB and Page Table Walker
- Software:** OS constructs and installs these mappings in process address space



Address Translation with CHP



OS Implementation

Intercept page faults in large VMAs

- Best-effort to map faulting 2MB region with CHP
- If not available, fall back to 4KB page coloring

How to allocate a strided 256KB sub-mapping (repeat 8x to fill entire CHP):

- Request a #color * 256KB contiguous chunk from buddy allocator
- Carve out the individual 4KB pages of the desired color
- Install pages in process address space
- Cache left-over mappings in **Allocator Cache** for future use

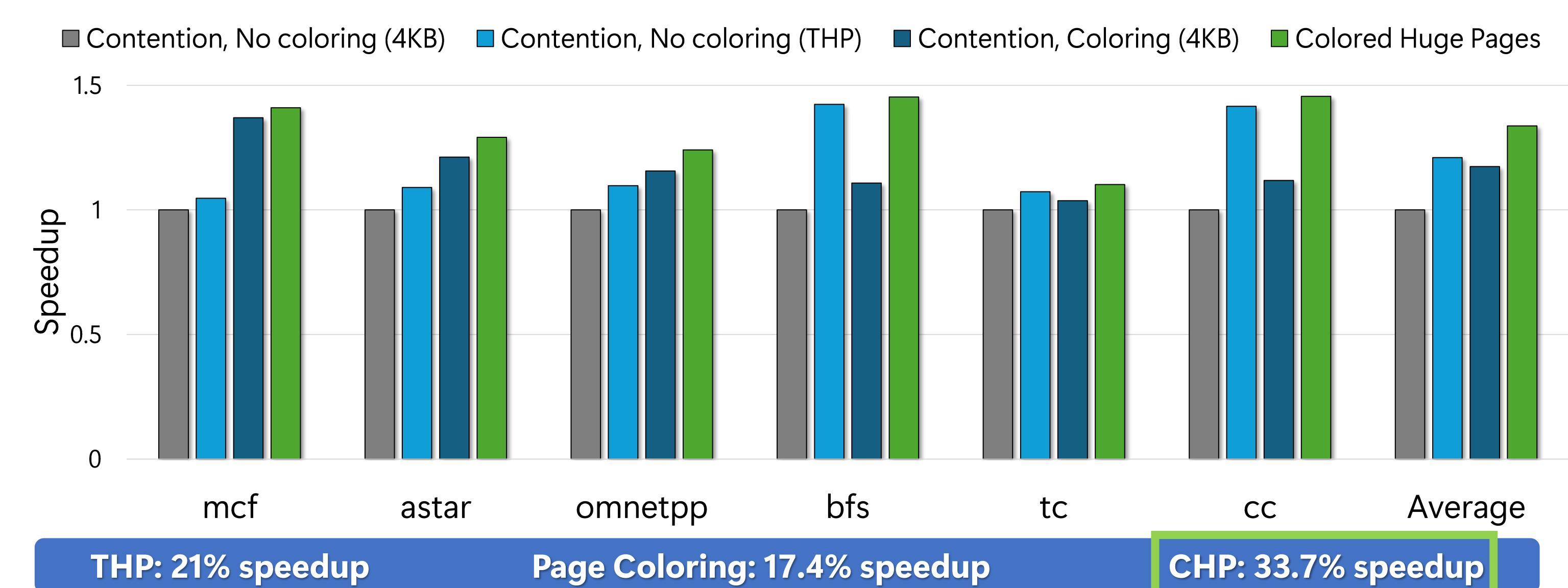
4 Evaluation

Methodology

Implemented in Linux v6.6 – Analytical performance modeling based on hardware counters

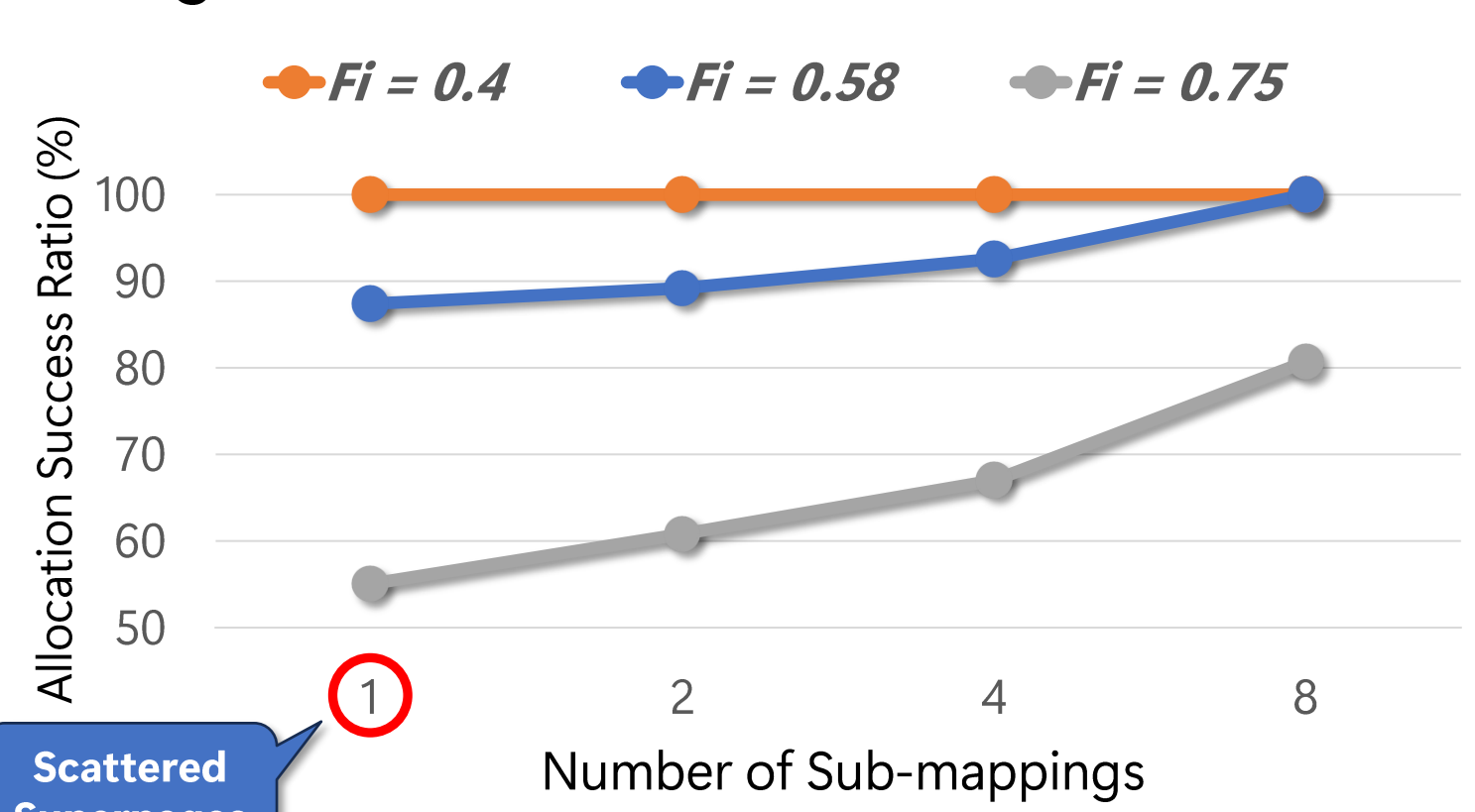
Performance

CHP successfully combines the benefits of both page coloring and huge pages, achieving better results than either technique in isolation



Impact of the #Sub-mappings

- Decreased allocation effectiveness when fewer sub-mappings are used
- Difference more pronounced under higher fragmentation



Comparison to SWAP

- SWAP restricts an entire huge page to a single cache subset (color)
- Up to 12% increase in LLC misses and 13.1% slowdown

