Jin-Soo Kim
(*jinsoo.kim@snu.ac.kr*)

Systems Software &
Architecture Lab.

Seoul National University

Fall 2022

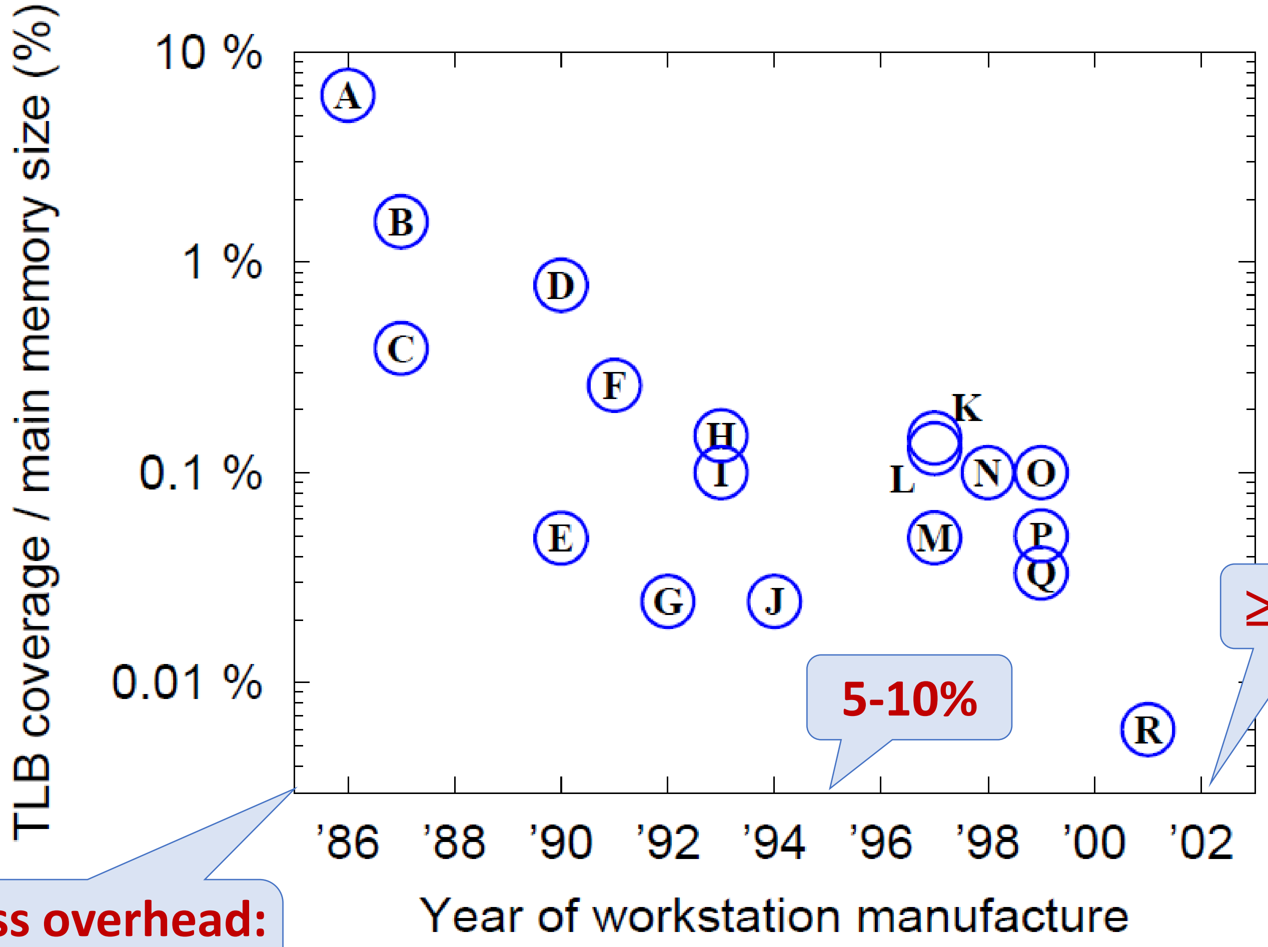
Superpages

(Juan Navarro et al., OSDI '02)

Some slides are borrowed from the authors'

Motivation

- TLB coverage
  - The amount of memory accessible through cached mappings in the TLB



TLB miss overhead: ≤ 5%

5-10%

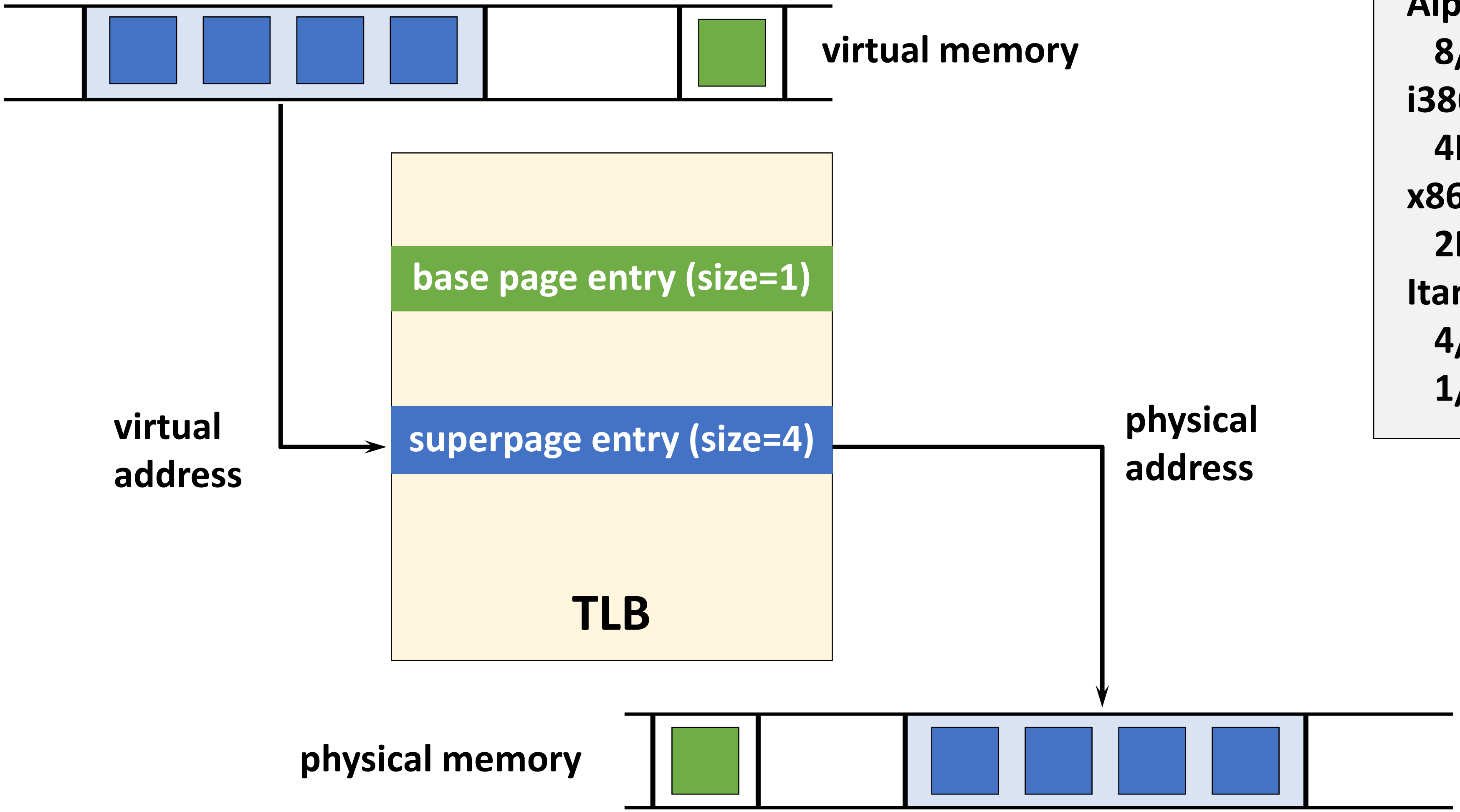
≥ 30%

Superpages

- Memory pages of larger sizes than base pages
  - Supported by most modern CPUs

- Otherwise, same as normal pages
  - Power-of-2 size
  - Use only one TLB entry
  - Contiguous (physically and virtually)
  - Aligned on superpage boundary
  - Uniform protection attributes
  - One reference bit, one dirty bit

TLB with Superpages



virtual memory

virtual address

base page entry (size=1)

superpage entry (size=4)

TLB

physical address

physical memory

Alpha:
8/64/512KB, 4MB
i386:
4KB, 4MB
x86_64:
2MB, 1GB
Itanium:
4/8/16/64/256KB,
1/4/16/64/256MB

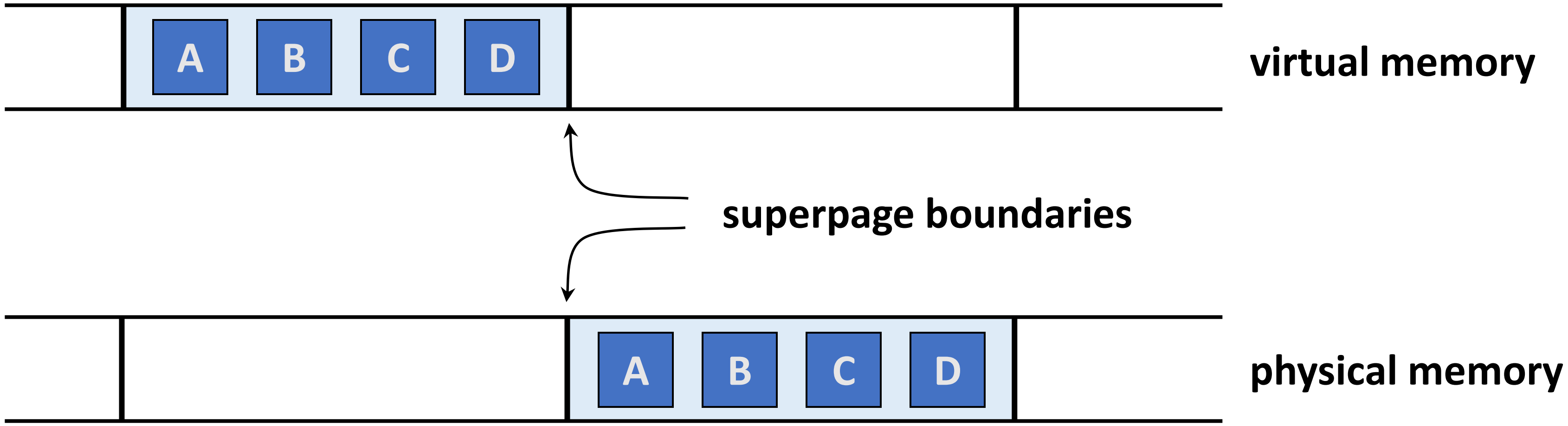
Using Superpages for Base Pages

- Why?
  - Increased TLB coverage without enlarging the TLB size

- Why not?
  - Enlarged application footprint
  - Increased internal fragmentation due to partly used pages
  - Premature onset of memory pressure
  - Higher I/O demands due to increased paging granularity

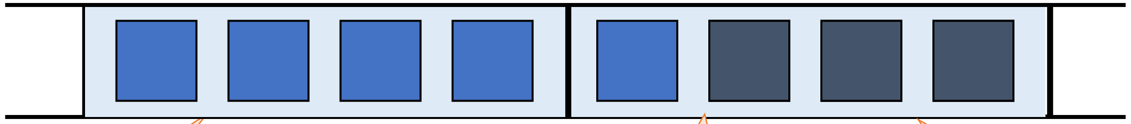
Issue 1: Superpage Allocation

- How / when / what size to allocate?
- Relocation-based: requires memory copy
- Reservation-based: superpage size to reserve?



Issue 2: Promotion

- Create a superpage out of a set of smaller pages
- Promotion can be performed incrementally
- When to promote?



Create small superpage? May incur overhead

Forcibly populate pages? May incur I/O cost or increase internal fragmentation

Wait for app to touch pages? May lose opportunity to increase TLB coverage

Issue 3: Demotion

- Convert a superpage into smaller pages
- When page attributes of base pages of a superpage become non-uniform
- During partial pageouts
  - All portions of a superpage not actively used

- Problem:
  - Hardware only maintains a single reference bit for the superpage
  - Which portions of a superpage are actively used?

Issue 4: Eviction

- Inactive superpages evicted from physical memory on memory pressure

- Problem: dirty pages
  - Hardware maintains a single dirty bit for the superpage
  - Which base pages should be flushed?

Issue 5: Fragmentation

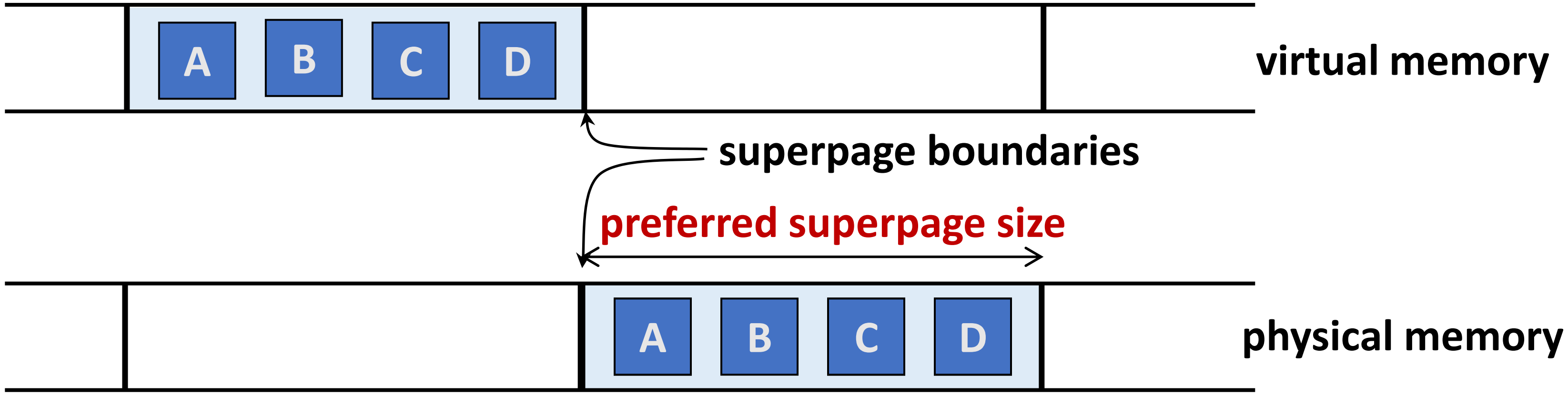
- Memory becomes fragmented due to
  - Use of multiple page sizes
  - Scattered wired (non-pageable) pages
- Contiguity: contended resource
- OS must
  - Use contiguity restoration techniques
  - Trade off impact of contiguity restoration against superpage benefits

Overall Design

- Observation: Once an application touches the first page of a memory object then it is likely that it will quickly touch every page of that object
  - Superpages as large and as soon as possible
  - As long as no penalty if wrong decision
- Reservation-based superpage management
- Support for multiple superpage sizes
- Scalability to very large superpages
- Demotion of sparsely referenced superpages
- Effective preservation of contiguity without the need for compaction
- Efficient disk I/O for partially modified superpages

Superpage Allocation

- Reservation-based (preemptible) allocation
  - On a page fault, determine a preferred superpage size
  - Only the mapping for the faulting page is inserted into the page table
  - The rest of frames are tentatively reserved for potential future use



Preferred Superpage Size

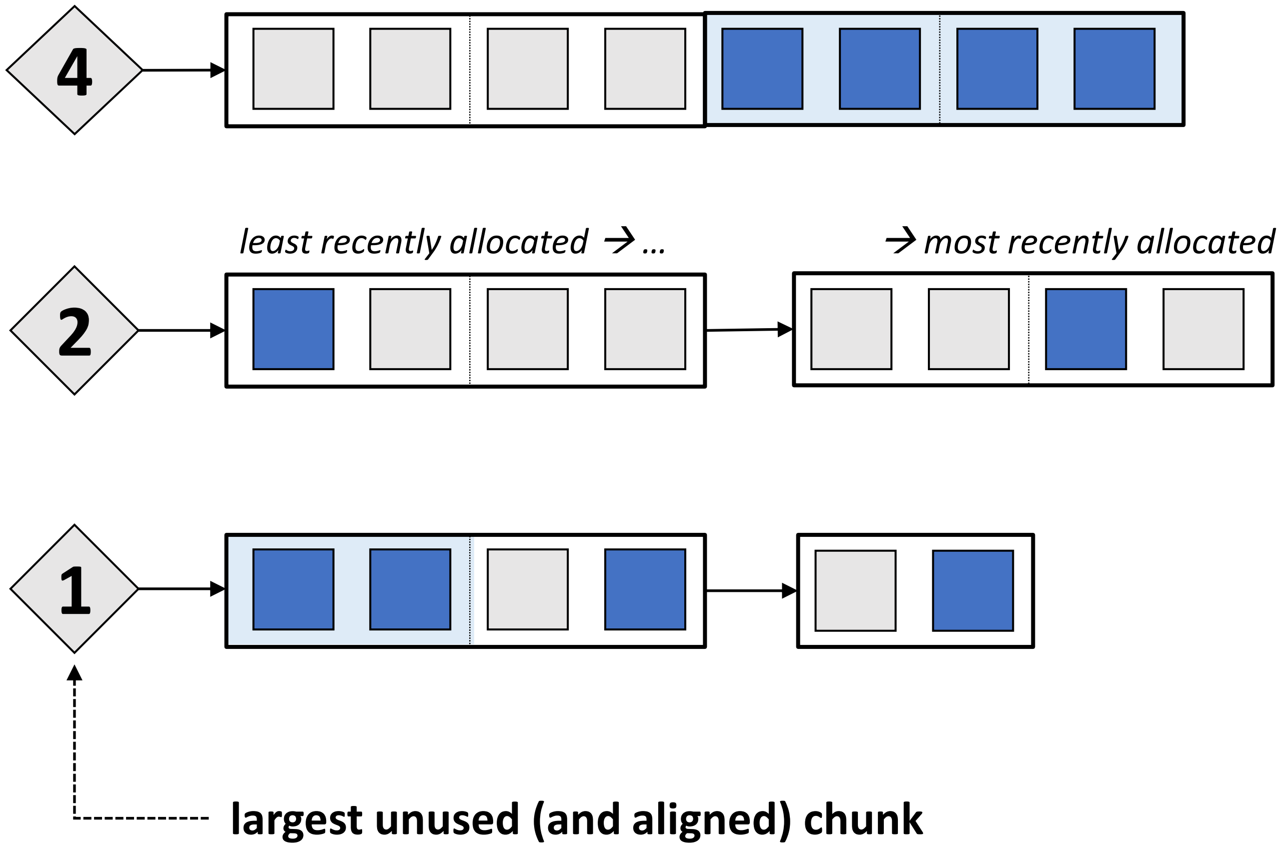
- Observation
  - Too large superpage → Can be preempted later
  - Too small superpage → Need relocation

- Opportunistic policy
  - The largest, aligned superpage that contains the faulting page, not overlapped with existing reservations or allocated pages
  - For fixed size memory objects (e.g., code, data, memory-mapped files):
    No larger than the memory object
  - For dynamically sized memory objects (e.g., stack, heap):
    The superpage size is limited to the current object size

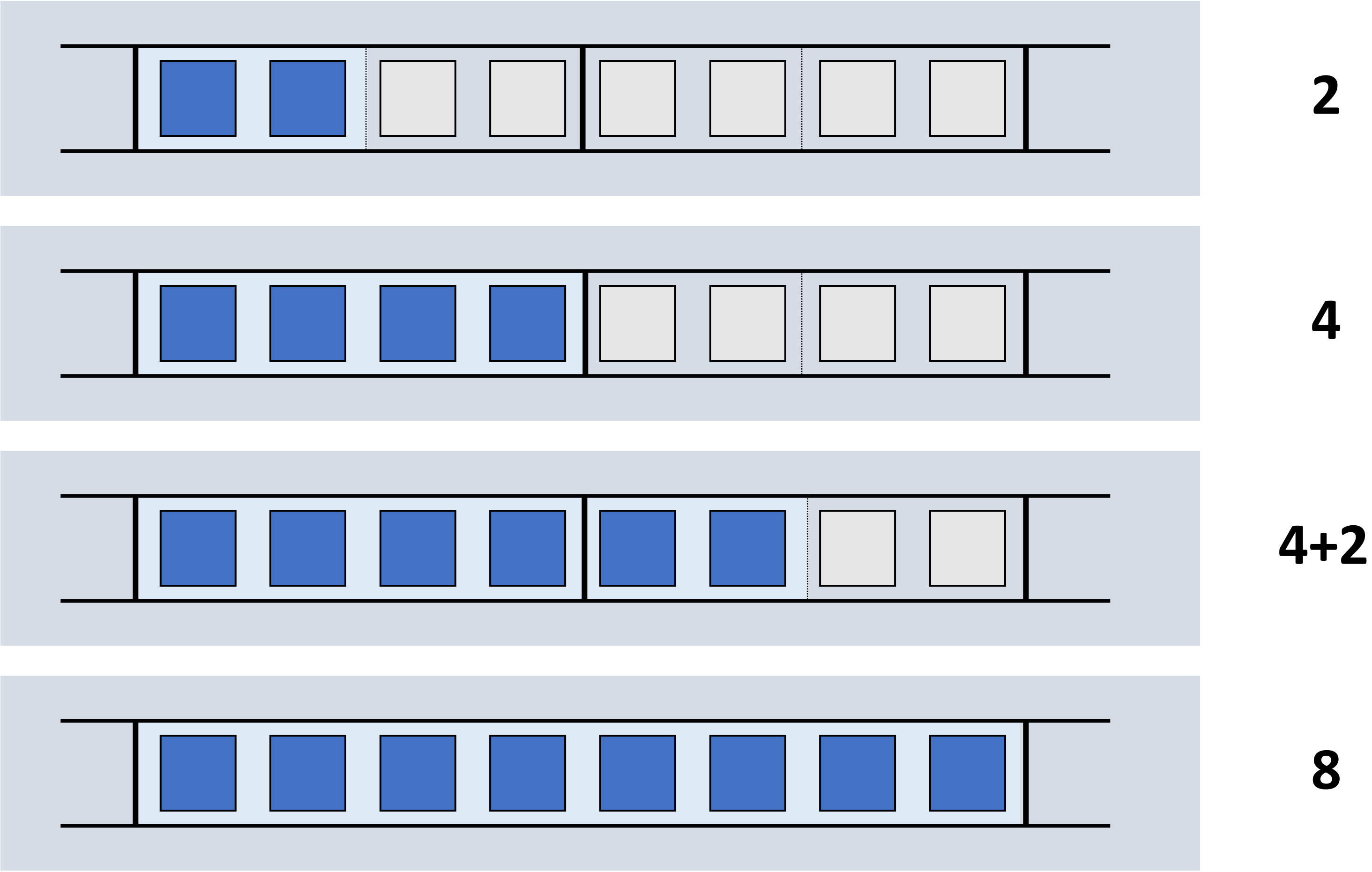
Preempting Reservations

- When free physical memory becomes scarce or excessively fragmented

- Victim selection: Reservation that the most recent population was done least recently


4
least recently allocated → …
→ most recently allocated
2
1
largest unused (and aligned) chunk

Incremental Promotions



Speculative Demotions

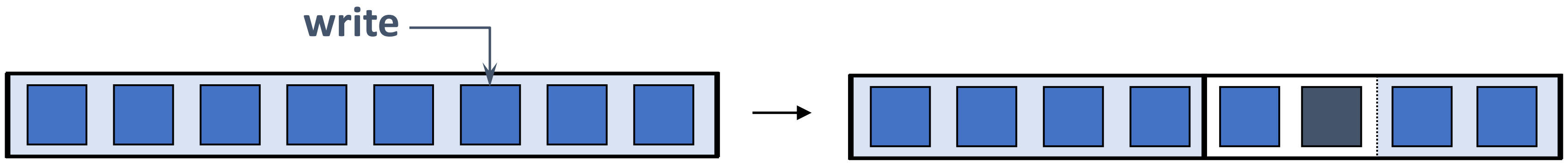
- Incremental demotion
  - When a base page is selected for eviction
  - When the protection attributes are changed on part of a superpage
  - Demoted incrementally to the smaller superpage sizes

- Speculative demotion
  - How to detect portions of a superpage not referenced anymore?
  - On memory pressure, demote superpages when resetting reference bit
  - Re-promote (incrementally) as pages are referenced

Evicting Dirty Superpages

- One dirty bit per superpage
  - What's dirty and what's not?

- Demote on first write to clean superpage
- Re-promote (incrementally) as other pages are dirtied

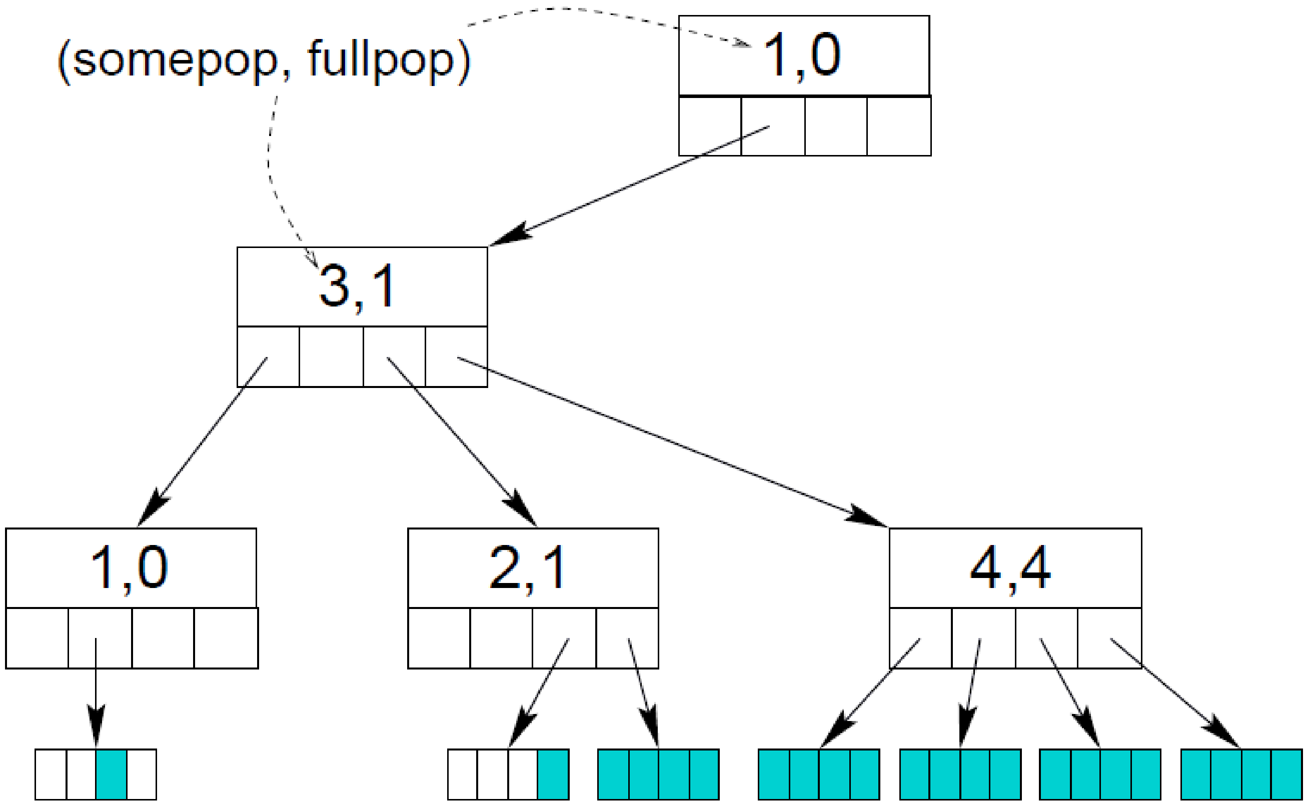


- Inferring dirty pages using hash digests?

Population Map

- Use hash table + radix tree
- Each level corresponds to a page size

- Reserved frame lookup
- Overlap avoidance
- Promotion decision
- Preemption assistance



FreeBSD Implementation

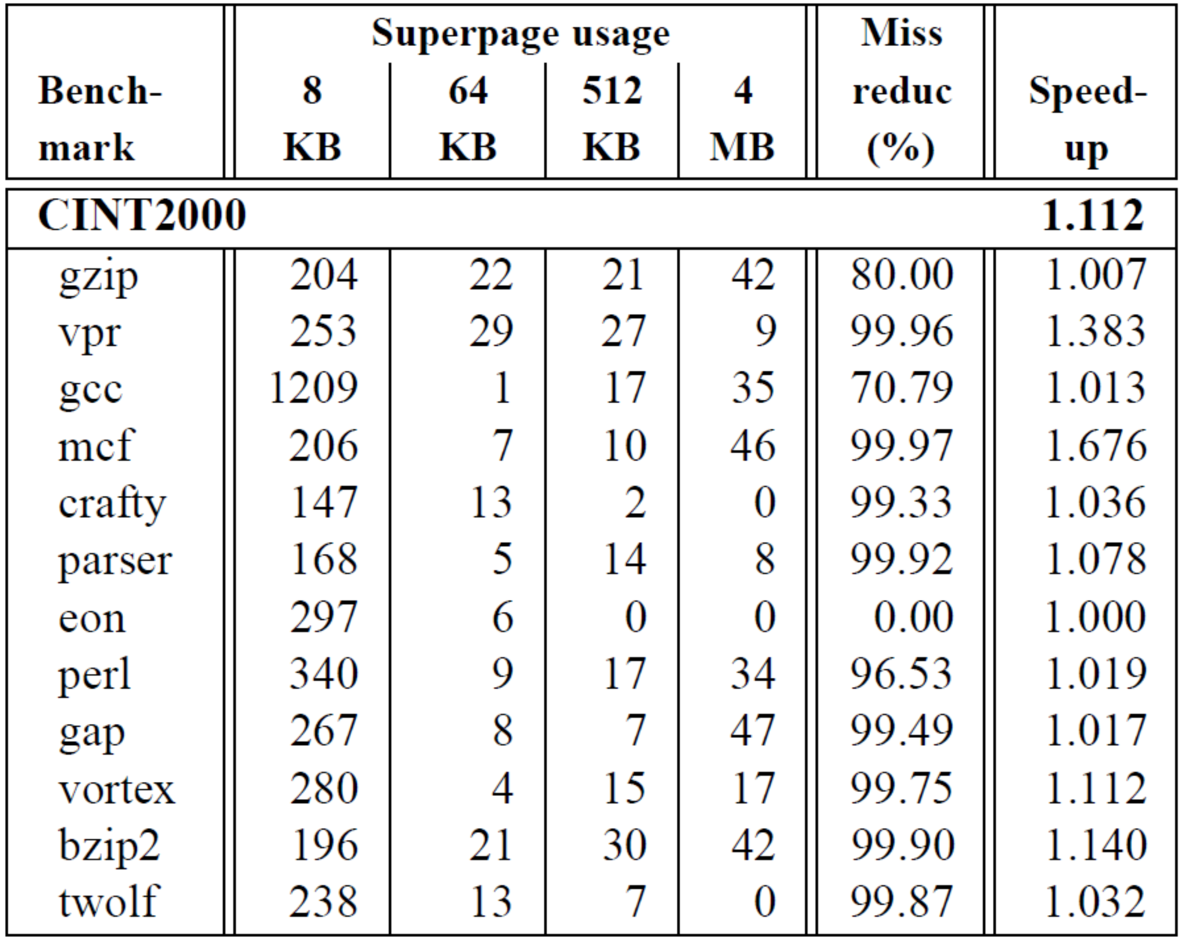
- FreeBSD lists of pages
  - Active: access recently (reference bit can be either 0 or 1)
  - Inactive: mapped, not referenced for a long time
  - Cache: clean and unmapped
- Contiguity-aware page daemon
  - Use cache pages for reservations
    - If a cache page is referenced, the associated reservation is preempted
  - On low contiguity, move clean, inactive pages to the cache list
    - Prefer pages that contribute the most to contiguity
  - Clean file pages moved to the inactive list when the file is closed
- Cluster wired pages

Experimental Setup

- FreeBSD 4.3
- Alpha 21264 @ 500MHz, 512MB RAM
- 8KB, 64KB, 512KB, 4MB pages
- 128-entry DTLB, 128-entry ITLB
- Unmodified applications from SPEC CPU2000 benchmark and others

Best-case Performance

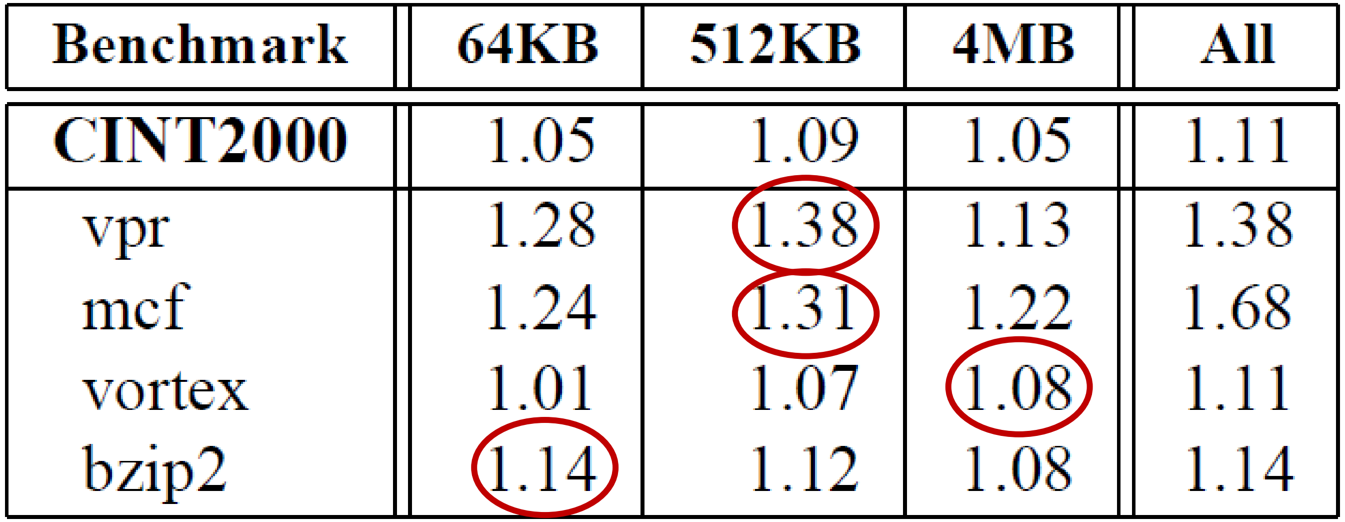
- 30%+ in 8 out of 35 benchmarks



Bench-mark	Superpage usage 8 KB	64 KB	512 KB	4 MB	Miss reduc (%)	Speed-up
CINT2000						**1.112**
gzip	204	22	21	42	80.00	1.007
vpr	253	29	27	9	99.96	1.383
gcc	1209	1	17	35	70.79	1.013
mcf	206	7	10	46	99.97	1.676
crafty	147	13	2	0	99.33	1.036
parser	168	5	14	8	99.92	1.078
eon	297	6	0	0	0.00	1.000
perl	340	9	17	34	96.53	1.019
gap	267	8	7	47	99.49	1.017
vortex	280	4	15	17	99.75	1.112
bzip2	196	21	30	42	99.90	1.140
twolf	238	13	7	0	99.87	1.032

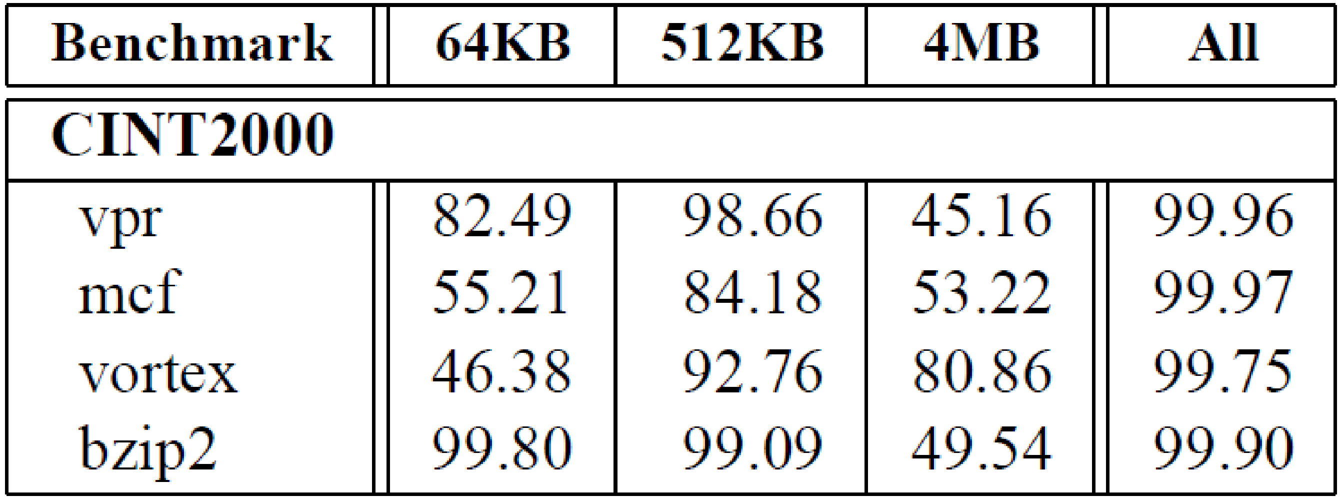
Multiple Superpage Sizes

Speedups



Benchmark	64KB	512KB	4MB	All
CINT2000	1.05	1.09	1.05	1.11
vpr	1.28	1.38	1.13	1.38
mcf	1.24	1.31	1.22	1.68
vortex	1.01	1.07	1.08	1.11
bzip2	1.14	1.12	1.08	1.14

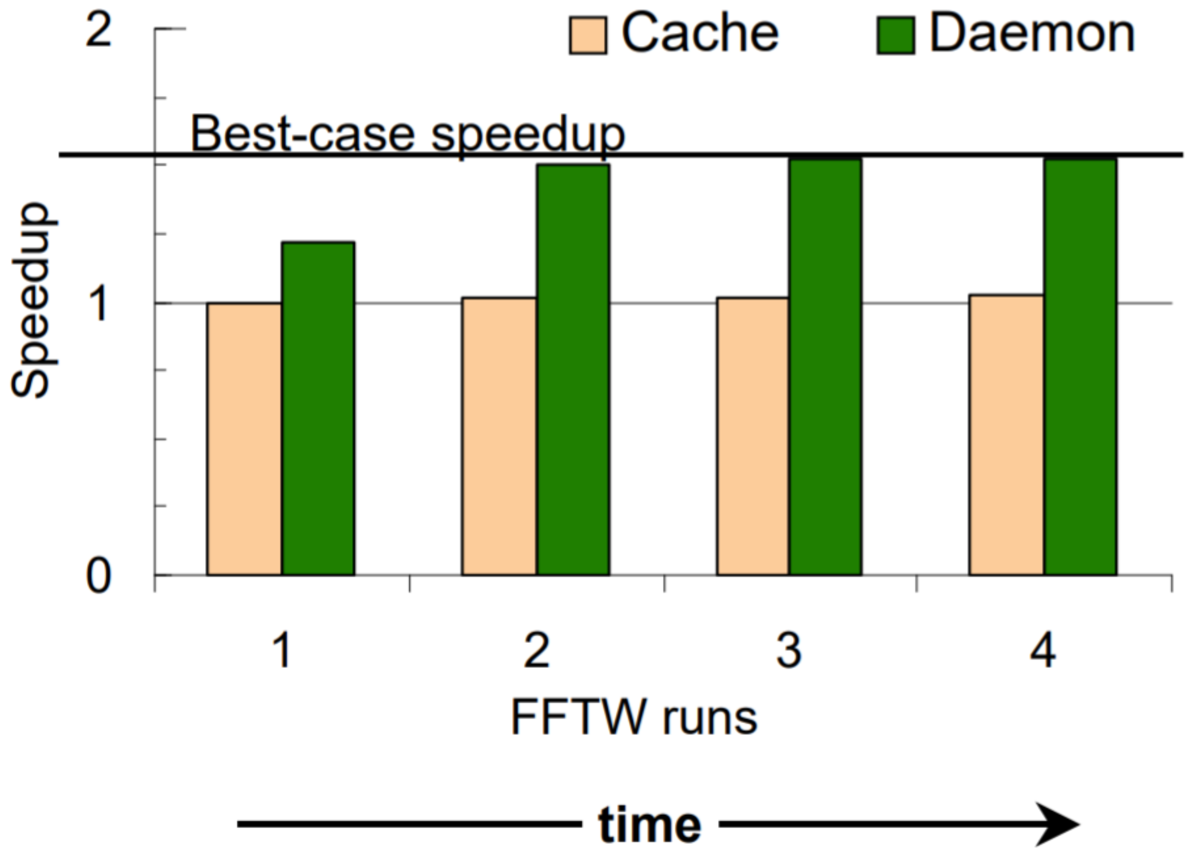
TLB miss reduction (%)



Benchmark	64KB	512KB	4MB	All
CINT2000				
vpr	82.49	98.66	45.16	99.96
mcf	55.21	84.18	53.22	99.97
vortex	46.38	92.76	80.86	99.75
bzip2	99.80	99.09	49.54	99.90

Fragmentation Control

- Web server to create memory fragmentation + four runs of FFTW
  - Cache: all cached pages are used for superpages
  - Daemon: contiguity-aware page replacement daemon



Summary

- Superpages: 30%+ improvement
  - Transparently realized, low overhead

- Contiguity restoration is necessary
  - Sustains benefits, low impact

- Multiple page sizes are important
  - Scales to very large superpages

Follow-ups

- Ingens
  - "Coordinated and Efficient Huge Page Management with Ingens", OSDI, 2016
  - For modern Intel CPU-based servers with hypervisors
  - Linux's transparent huge page support is greedy and aggressive

- Quicksilver
  - "A Comprehensive Analysis of Superpage Management Mechanisms and Policies", USENIX ATC, 2020
  - A framework proposed to understand various superpage management schemes: FreeBSD, Linux, Ingens, HawkEye
  - Sync vs. async allocation, incremental vs. full preparation, in-place vs. out-of-place mapping, etc.