Practical, transparent operating system support for superpages

이인조
육준성, 이한얼

Contents

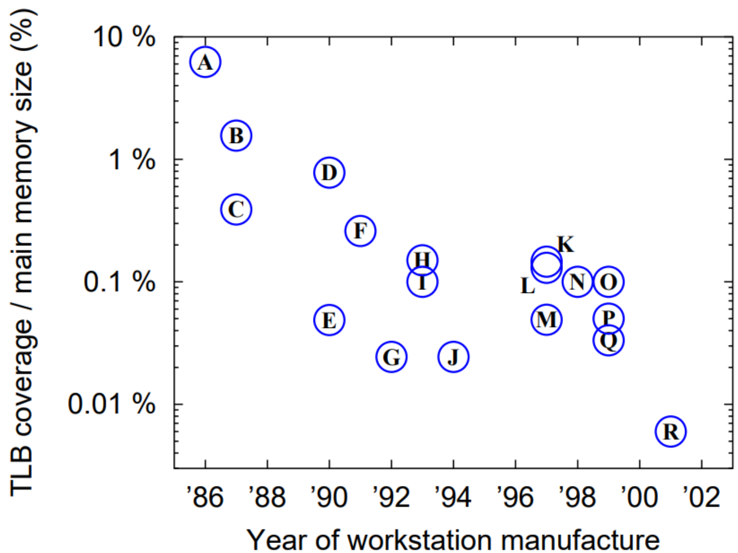
Background

- TLB coverage is decreasing with growing memory size over years
- Use of large page can lead to memory pressure
- Superpages are able to alleviate this problem
- Terminology
  - TLB Coverage
    - The amount of memory accessible through TLB entries, without TLB misses
  - Superpage
    - Memory page for larger size than a Base-page (ordinary memory page)
    - Multiple sizes
    - Single entry in TLB per superpage
  - Memory objects
    - Occupy some contiguous space in virtual memory
    - E.g.: memory mapped files, code, data

- But, Superpages may cause wasted memory and fragmentation
- Managing superpages become a complex

Motivation

- TLB coverage trend
  - Factor of 1000 decrease in 15 years



- To increase TLB coverage
  - More TLB entries is expensive
  - Larger page size leads to internal fragmentation and increased I/O
  - Solution: use multiple page size
  - But, also, leads to external fragmentation and imposes several challenges

Constraints for Superpages

- Page size should be supported by processor.
  - Alpha processor provides 8KB base pages and 64KB, 512KB and 4MB superpages
- Contiguous.
  - physical and virtual address space
- Aligned.
  - starting address in physical & virtual address space must be a multiple of its size.
- Single TLB entry for a superpage. Meaning single set of protection attributes and single reference, dirty bit for a superpage.
  - due to the coarse granularity, cannot distinguish between base pages

Issues and tradeoffs

- Allocation
  - relocation-based allocation
    - Incurs overhead of moving pages when superpages are created to satisfy contiguity and alignment
  - reservation-based allocation
    - when a page is initially loaded decide what the final superpage size will be and reserve contiguous frames
    - Choose a superpage size without foreknowledge.(largest size that is smaller or equal to the size of memory object)
    - Don’t know if adjoining pages will ever be needed by the program

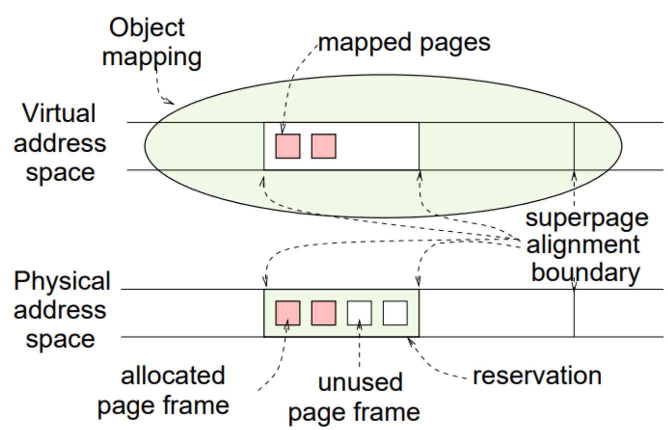


Figure 2: Reservation-based allocation.

Issues and tradeoffs

- Fragmentation Control
  - Release contiguous chunks of inactive memory from previous allocations
  - Preempt an existing partially is used reservation
- Promotion
  - Set of base pages from a potential super page satisfying constraints is promoted to a Superpage
  - Incrementally promote up to size of the original reservation
- Demotion
  - Reduce size of a superpage
    - A smaller super page
    - Base pages
  - A single reference bit make it difficult to detect which portion are used
- Eviction
  - At a demand of memory preassure, inactive superpage is evicted from physical memory
  - A single dirty bit can cause an entire flush

Related approaches

- Partial-subblock TLB

Page 1: VPN = 110100 PPN = 10000 Attr = α
Page 2: VPN = 110101 PPN = 11011 Attr = α
Page 3: VPN = 110110 PPN = 00010 Attr = α
Page 4: VPN = 110111 PPN = 00011 Attr = α

✓ => Valid

✗ => Invalid

Single-Page-Size TLB

110100
110101
110111
110110

10000	α	Mod	✓
11011	α	Mod	✓
00011	α	Mod	✓
00010	α	Mod	✓

4K/16K Superpage TLB

110100	4KB
110101	4KB
110111	4KB
110110	4KB

10000	α	Mod	✓
11011	α	Mod	✓
00011	α	Mod	✓
00010	α	Mod	✓

Complete-subblock TLB (subblock factor = 4)

1101
Unused
Unused
Unused

10000	α	M0	✓	11011	α	M1	✓	00010	α	M2	✓	00011	α	M3	✓
			✗				✗				✗				✗
			✗				✗				✗				✗
			✗				✗				✗				✗

Partial-subblock TLB (subblock factor = 4)

1101	✓	✗	✗	✗
1101	✗	✗	✓	✓
1101	✗	✓	✗	✗
Unused	✗	✗	✗	✗

10000	α	0	M0			
00000	α	0			M2	M3
11011	α	1		M1		

Design

- Reservation-based allocation
  - Set of frames initially is reserved at page fault
  - The system determines a preferred superpage size
  - A mapping is entered into the page table for the base page

- Preferred Superpage Size Policy
  - Decision is made early at process execution
  - Looks only at attributes of memory object
    - For fixed-size memory objects(e.g. code sefments)
    - For Dynamically sized memory objects(e.g. stacks, heaps)
  - Decided size
    - Too large: decision overridden by preempting init reservation
    - Too small: cannot be reverted
  - Policy: Pick the maximum

Design

- Preempting reservations
  - Free physical memory is scarce/ excessively fragmented
  - Preempt reserved but unused frames
  - Choice between
    - Refusing allocation; reserving smaller extent than desired for new allocation
    - Preempting existing reservation which has enough unallocated frames
  - Whenever possible preempt & give space for allocation

- Fragmentation control
  - buddy allocator performs coalescing of available memory regions whenever possible
  - Contiguity-aware page daemon
    - Controlling fragmentation comes at a price.

Design

- Incremental Promotions
  - Multiple superpage sizes exist
  - Superpage gets created when any size supported gets fully populated within a reservation
  - Increment smaller superpage to next larger size superpage

- Speculative Demotions
  - A base page of a superpage targetted to be evicted by page daemon, causes a demotion
  - Incrementally demote to next smaller superpage size
  - Demote when the protection attributes arechanged on part of a superpage
  - Demote in order to determine if the superpage is still being actively used in its entirety

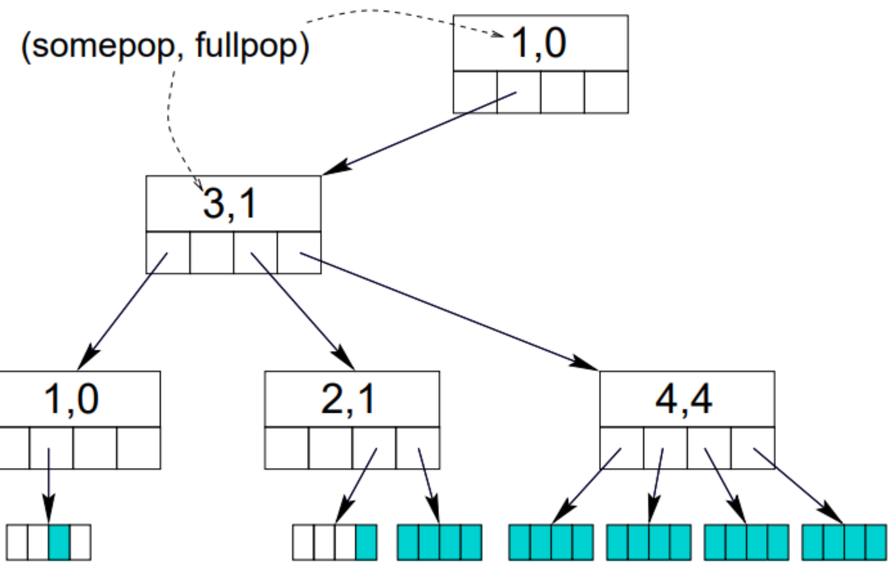
Design

- Paging Out Dirty Superpages
  - OS keeps only single dirty bit for the whole superpage.
  - Problem: only one base page modified; cost of writing full superpage to disk (high cost on I/O)
  - Demote clean superpages whenever process attempts to write to them and can repromote later

- Multi-list reservation scheme
  - Keep track of reserved page extent that are not fully populated
  - One list per each page size supported.
  - Kept sorted by the time of their most recent page frame allocation
  - Preempt the extent whose most recent allocation, occurred least recently among all reservations in the list (head)

Design

- Population Map
  - Keep track of allocated base pages within each memory object
  - Queried on every page fault, should support efficient lookups.
  - Used for various decisions
    - Reserved frame lookup
    - Overlap avoidance
    - Promotion decisions
    - Preemption assistance



Evaluation

- Platform
  - FreeBSD-4.3 kernel
  - 3500 lines of C code
  - Compaq XP-1000 machine
    - Alpha 21264 processor at 500 MHz
    - Four page sizes: 8KB base pages, 64KB, 512KB and 4MB superpages
    - Fully associative TLB with 128 entries for data and 128 for instructions
    - Software page tables, with firmware-based TLB loader

  - Page table entry replication
    - 4MB : page size field of 512 page table entries

Evaluation

- Best-case benefits due to superpages

Benchmark	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
CFP2000						**1.110**
wupw	219	14	6	43	96.77	1.009
swim	226	16	11	46	98.97	1.034
mgrid	282	15	5	13	98.39	1.000
applu	1927	1647	90	5	93.53	1.020
mesa	246	13	8	1	99.14	0.985
galgel	957	172	68	2	99.80	1.289
art	163	4	7	0	99.55	1.122
equake	236	2	19	9	97.56	1.015
facerec	376	8	13	2	98.65	1.062
ammp	237	7	21	7	98.53	1.080
lucas	314	4	36	31	99.90	1.280
fma3d	500	17	27	22	96.77	1.000
sixtr	793	81	29	1	87.50	1.043
apsi	333	5	5	47	99.98	1.827
Web	30623	5	143	1	16.67	1.019
Image	163	1	17	7	75.00	1.228
Povray	136	6	17	14	97.44	1.042
Linker	6317	12	29	7	85.71	1.326
C4	76	2	9	0	95.65	1.360
Tree	207	6	14	1	97.14	1.503
SP	151	103	15	0	99.55	1.193
FFTW	160	5	7	60	99.59	1.549
Matrix	198	12	5	3	99.47	7.546

Evaluation

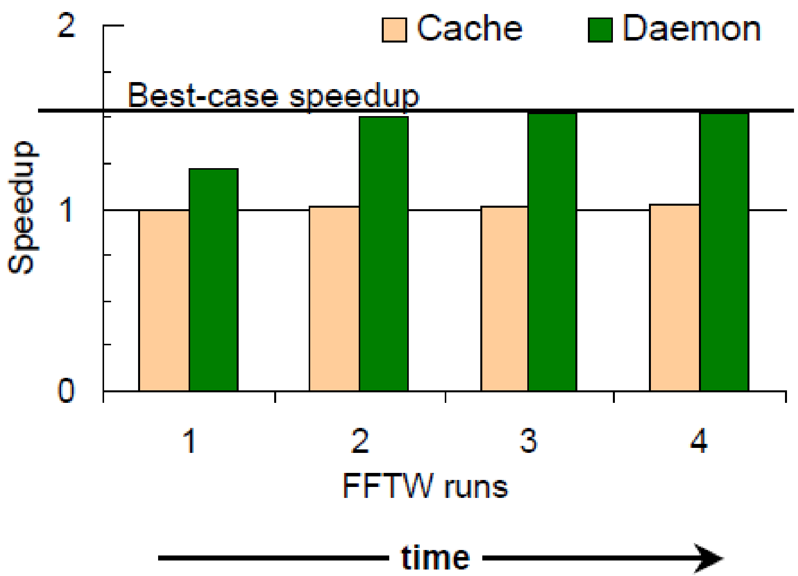
- Benefits from multiple superpage sizes

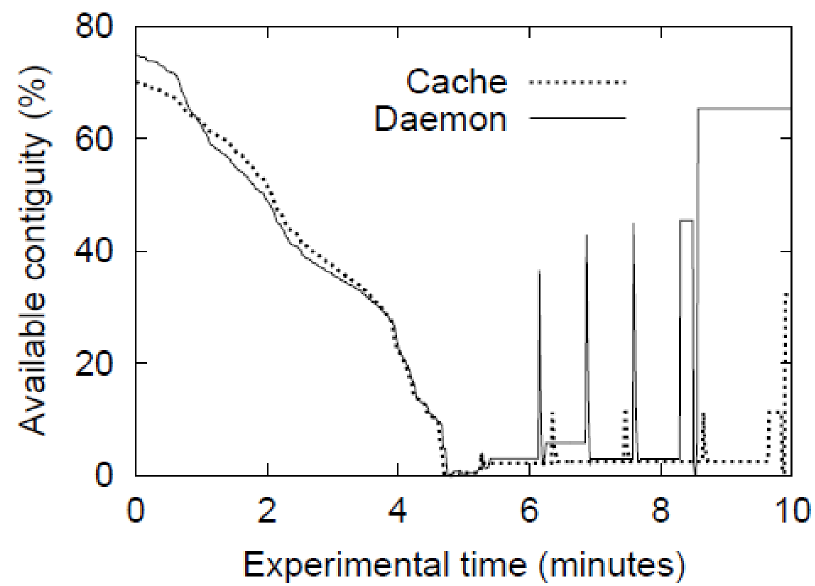
Benchmark	64KB	512KB	4MB	All
CINT2000	1.05	1.09	1.05	1.11
vpr	1.28	1.38	1.13	1.38
mcf	1.24	1.31	1.22	1.68
vortex	1.01	1.07	1.08	1.11
bzip2	1.14	1.12	1.08	1.14
CFP2000	1.02	1.08	1.06	1.12
galgel	1.28	1.28	1.01	1.29
lucas	1.04	1.28	1.24	1.28
apsi	1.04	1.79	1.83	1.83
Image	1.19	1.19	1.16	1.23
Linker	1.16	1.26	1.19	1.32
C4	1.30	1.34	0.98	1.36
SP	1.19	1.17	0.98	1.19
FFTW	1.01	1.00	1.55	1.55
Matrix	3.83	7.17	6.86	7.54

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
CFP2000				
galgel	98.51	98.71	0.00	99.80
lucas	12.79	96.98	87.61	99.90
apsi	9.69	98.70	99.98	99.98
Image	50.00	50.00	50.00	75.00
Linker	57.14	85.71	57.14	85.71
C4	95.65	95.65	0.00	95.65
SP	99.11	93.75	0.00	99.55
FFTW	7.41	7.41	99.59	99.59
Matrix	90.43	99.47	99.47	99.47

Evaluation

- Sustained benefits in the long term
  - Sequential execution





  - Concurrent execution

Evaluation

- Overheads
  - Incremental promotion overhead
  - Sequential access overhead
  - Preemption overhead
  - Overhead in practice

- Dirty superpages
  - The performance penalty : 20x

Evaluation

- Scalability
  - Promotion and demotions
  - Dirty/reference bit emulation

Conclusion

- Superpages are physical pages of large size, which may be used to increase TLB coverage, reduce TLB misses, and thus improve application performance.
- They describe a practical design and demonstrate that it can be integrated into an existing general-purpose operating system.
- They observe performance benefits of 30% to 60% in several cases, and show that the system is robust.

Limitation

- Dirty superpages
- Contiguity control
- Fairness