Introduction to Superpages

Background

- Virtual Memory systems divide memory into pages
- Pages are allotted by the operating system for each process
- Every memory access that is paged will go through a virtual to physical address translation
- For quicker translation, page tables are cached in TLBs.
- TLBs cannot be too big as they are in the critical path so multi-level TLBs are used

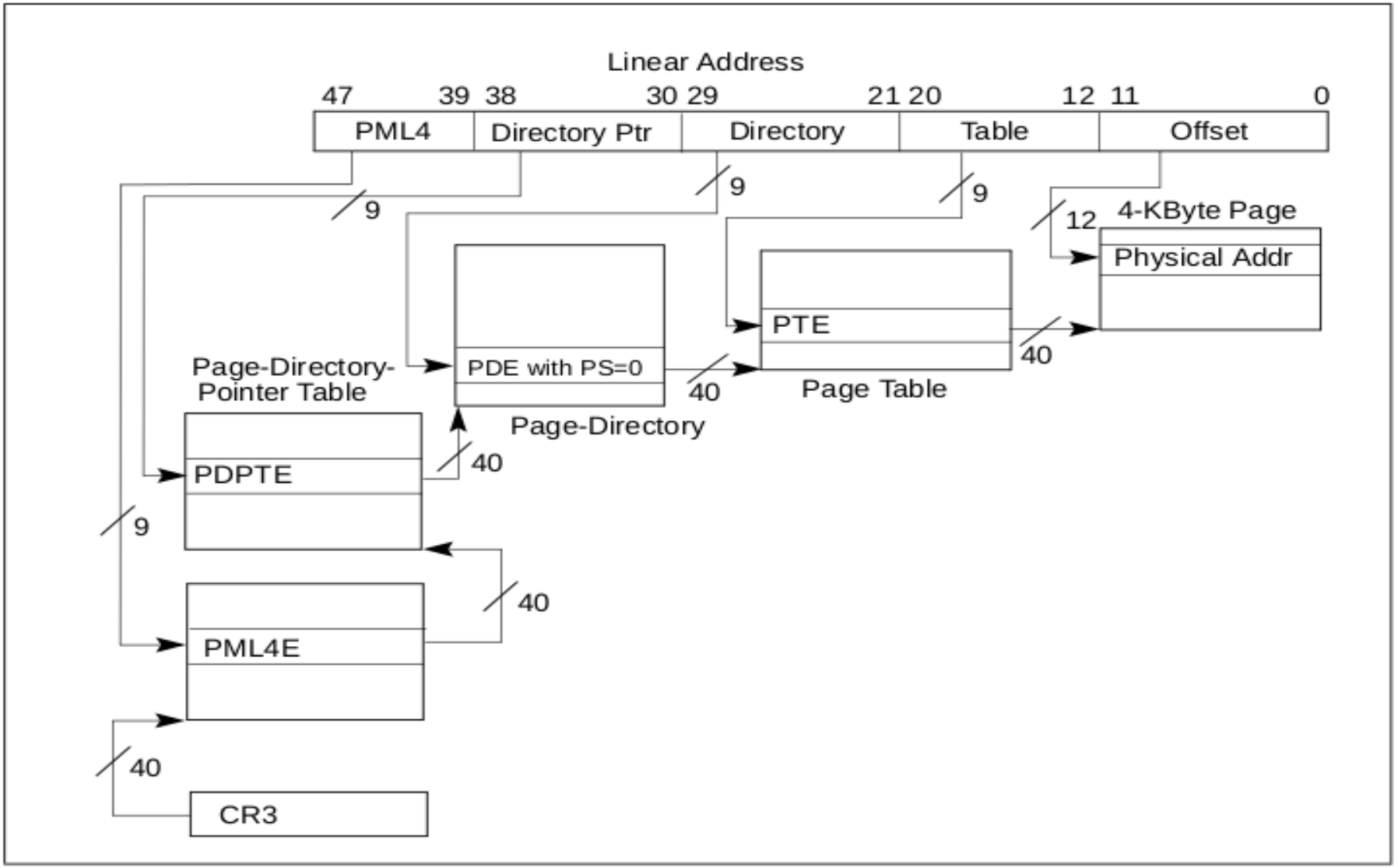
Pros and Cons of Page Sizes

- Smaller pages reduce internal fragmentation but occupy more memory (more entries) compared to a larger one
- When swapping from memory, access times dominate so larger pages are better.
  - Recent trends : 100 times improvement in transfer rate
  - 3 times improvement in access times [1]
- 128 entry TLB for a 4KB page size = ½ MB
  So code reach is very little for modern day applications
- Larger pages also cause coherency issues

Intel Support for Multiple Size Pages

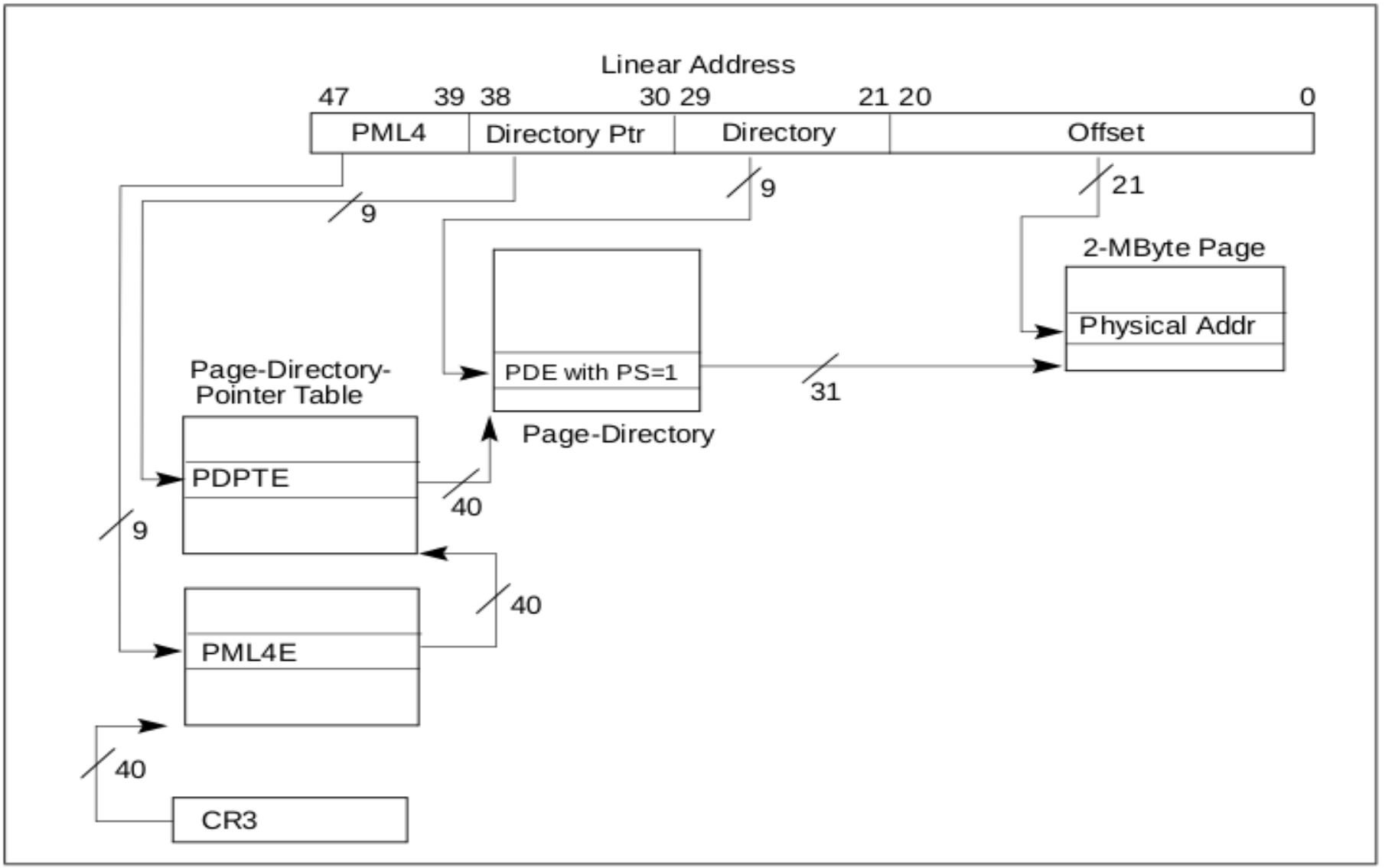
- Hierarchical page tables
- Each level is 4KB in size
  - 32 bit, 1024 entries (4B per entry)
  - 64bit, 512 entries (8B per entry)
- 64 bit architecture -- 4 levels
  - Nomenclature : Page map level4, page directory pointers, page directories and page tables
  - Supported sizes 4KB, 2MB and 1GB

Intel Support for Multiple Size Pages



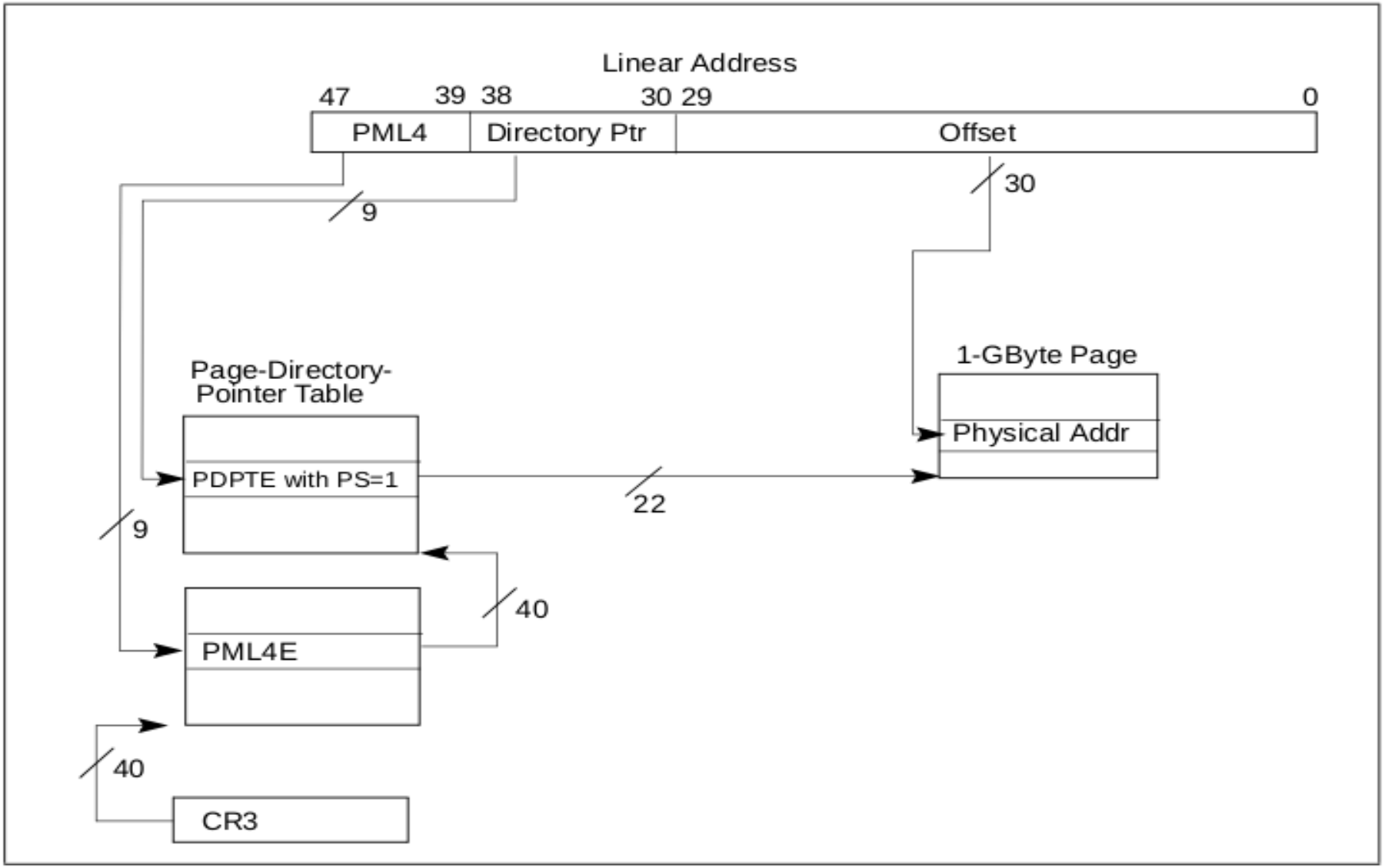
Paging Structure for 4KB Pages [2]

Intel Support for Multiple Size Pages



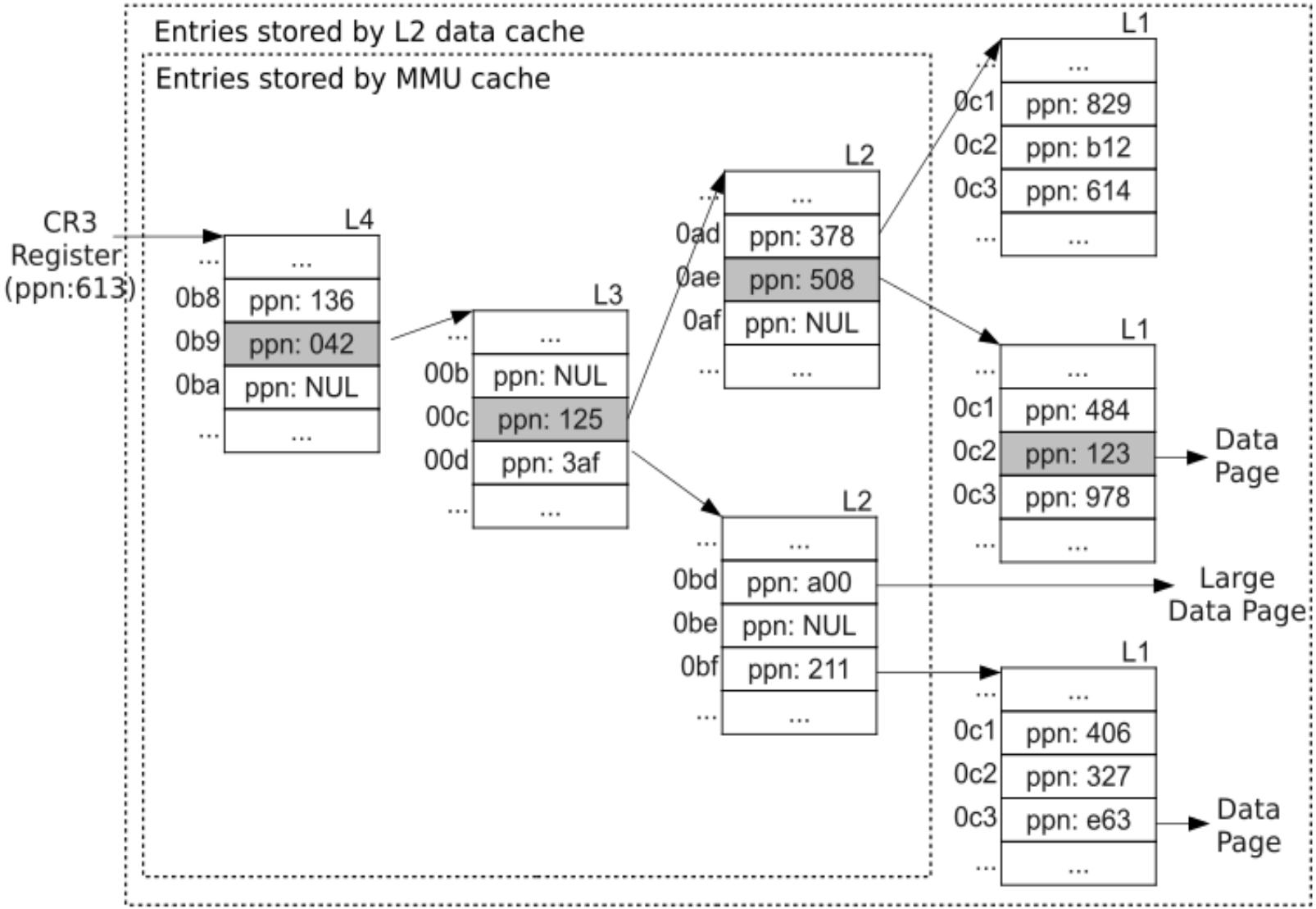
Paging Structure for 2MB Pages [2]

Intel Support for Multiple Size Pages



Paging Structure for 1GB Pages [2]

Intel Support for Multiple Size Pages



Overall Diagram [3]

Implementation of Superpages

- Allocation [6]
  - Superpage is allocated and loaded at the first page fault. All subpages are loaded and TLB entry is made
  - Size of superpages is either supplied by the user or detected by the operating system based on the size of the request
  - Further TLB misses are avoided once pages are loaded
  - Better to transfer large page from disk than many shorter ones

Implementation of Superpages

- Reservation [6]
  - Reservation for superpage is made but not loaded into memory
  - Loading of pages is performed at the base page size one at a time
  - All entries of the superpage are loaded when a particular threshold of loading base pages is met.
  - Advantage of ensuring that superpage will be used effectively before loading it
  - Also reduces start-up time of a process

Implementation of Superpages

- Relocation [6]
  - Base pages are loaded by default but memory usage is monitored
  - Decision to promote to superpage taken by operating system
  - Contiguous locations are found and pages are copied or brought in from secondary storage

Linux Support for Multipage Sizes

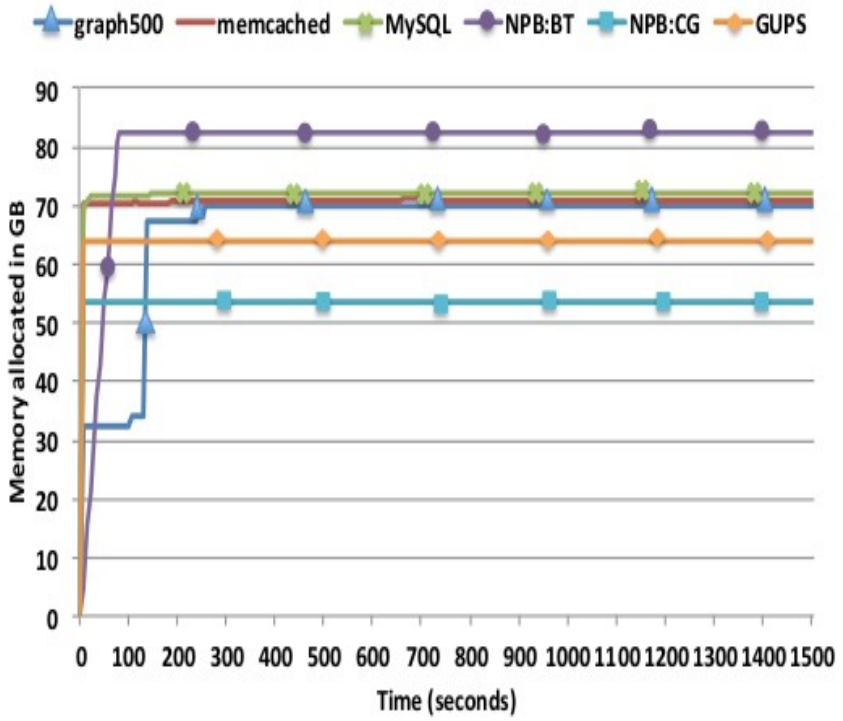
- Hugetlbfs [5] library supports hugepages (linux 2.6.16 onwards)
  - Function calls for using of huge pages
    - Application developer driven
  - Linker scripts for dynamically linked programs
    - Requires program to be relinked but no source code modification is necessary
    - Compiler directives for linking are available
- Environment variables needs to be set for enabling hugepages, enabling sharing of huge pages, etc
- If the hardware supports multiple page sizes, heap page size can be chosen as per program needs
- Information on current allocation can be found at /proc/meminfo

Case Study : Big Memory Servers

	Percentage of execution cycles servicing TLB misses			
	Base Pages (4KB)		Large Pages (2MB)	Huge Pages (1GB)
	D-TLB	I-TLB	D-TLB	D-TLB
graph500	51.1	0	9.9	1.5
memcached	10.3	0.1	6.4	4.1
MySQL	6.0	2.5	4.9	4.3
NPB:BT	5.1	0.0	1.2	0.06
NPB:CG	30.2	0.0	1.4	7.1
GUPS	83.1	0.0	53.2	18.3

TLB miss rates for server benchmarks for different page sizes [4]

Case Study : Big Memory Servers



Memory allocation with time [4]

	Percentage of allocated memory with read-write permission
graph500	99.96%
memcached	99.38%
MySQL	99.94%
NPB/BT	99.97%
NPB/CG	99.97%
GUPS	99.98%

Percentages of pages with read-write permissions [4]

Big Memory Servers

- Inference
  - Dynamic memory allocation is predictable
  - Accesses permissions for pages is predictable (RW, R, W, etc)
  - Swapping is almost non-existent due to careful optimizations and sufficient memory
  - Fragmentation problem managed internally
  - Scaling not possible with pages
- Conclusion
  - Makes a case for segmentation rather than pages
  - Fixed size of memory allocation not suitable for these applications

References

- [1] Tom’s Hardware. 15 Years Of Hard Drive History:Capacities Outran Performance. http://www.tomshardware.com/reviews/15-years-of-hard-drive-history,1368.html
- [2]http://www.intel.com/technology/intel64/index.htm
- [3] Thomas W. Barr, Alan L. Cox, and Scott Rixner. 2010. Translation caching: skip, don't walk (the page table). In Proceedings of the 37th annual international symposium on Computer architecture (ISCA '10). ACM, New York, NY, USA, 48-59.
- [4] Arkaprava Basu, Jayneel Gandhi, Jichuan Chang, Mark D. Hill, and Michael M. Swift. 2013. Efficient virtual memory for big memory servers. In Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA '13). ACM, New York, NY, USA
- [5] http://www.csn.ul.ie/~mel/projects/deb-libhugetlbfs/package/libhugetlbfs-1.2/HOWTO
- [6] P. Weisberg and Y. Wiseman. 2009. Using 4KB page size for virtual memory is obsolete. In Proceedings of the 10th IEEE international conference on Information Reuse \& Integration (IRI'09), Kang Zhang and Reda Alhajj (Eds.). IEEE Press, Piscataway, NJ, USA, 262-265