MPSoC'24

Will 4K pages last forever?

Eduardo Ribeiro[†‡], César Fuget[‡],
Christian Fabre[‡] and **Frédéric Pétrot**[†]
†Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA,
‡Univ. Grenoble Alpes, CEA, LIST,
F-38000 Grenoble, France
frederic.petrot@univ-grenoble-alpes.fr

Constants in the Universe

Find the intruder, ...

- Circle $\frac{Perimeter}{Diameter}$: $\pi = 3.14159265358979323846264338327950288419716939 9...$
- Euler's number : $e = 2.71828182845904523536028747135266249775724709...$
- Planck's constant : $h = 6.62607015 \times 10^{-34}\ J \times Hz^{-1}$
- Charge of the electron : $e = 1.602176634 \times 10^{-19}\ C$
- Speed of light : $c = 299792458\ m \times s^{-1}$
- ...
- Gravitational constant : $G = 6.6743015 \times 10^{-11}\ N \times m^2 \times kg^{-2}$
- Page size : $P_s = 4096\ B$
- Boltzmann constant : $1.380649 \times 10^{-23}\ J \times K^{-1}$
- ...

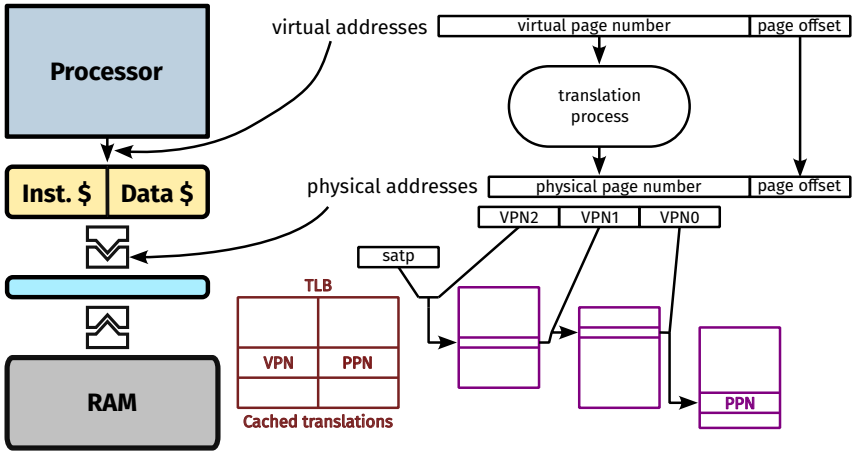
Constants in the Universe

Find the intruder, ...

- Circle $\frac{Perimeter}{Diameter}$: $\pi = 3.14159265358979323846264338327950288419716939 9...$
- Euler's number : $e = 2.71828182845904523536028747135266249775724709...$
- Planck's constant : $h = 6.62607015 \times 10^{-34}\ J \times Hz^{-1}$
- Charge of the electron : $e = 1.602176634 \times 10^{-19}\ C$
- Speed of light : $c = 299792458\ m \times s^{-1}$
- ...
- Gravitational constant : $G = 6.6743015 \times 10^{-11}\ N \times m^2 \times kg^{-2}$
- Page size : $P_s = 4096\ B$
- Boltzmann constant : $1.380649 \times 10^{-23}\ J \times K^{-1}$
- ...

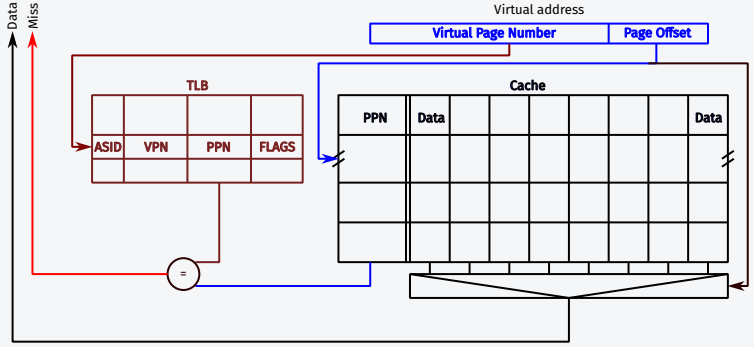
Address Translation Primer

CPUs produce virtual addresses, memories (IPs) consume (generally) physical addresses



Address Translation Primer

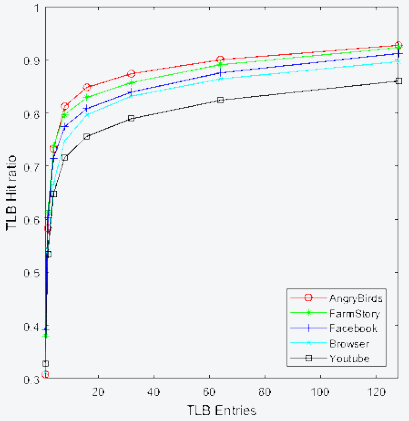
TLB in critical path of memory hierarchy



- TLB is a cache : size and associativity matters
- Can be neither too large nor too associative
- TLB misses are costly

Address Translation Primer

Performance impact of address translation



source : *Issues in File Caching and Virtual Memory Paging with Fast SCM Storage*. Yunjoo Park and Hyokyung Bahn. Advances in Science, Technology and Engineering Systems Journal, vol 5, 2020.

Latencies :

- L1-TLB miss, often highly associative :
  $\approx$ 5 cycles
- L2-TLB miss :
  $\approx$ 50 cycles

High miss ratio
$\Rightarrow$ quickly decreasing performance

Address Translation Primer

Address Translation Archaeology

- Introduction of 4 KiB pages in 1964 :
  G. Amdahl, G. A. Blaauw, and F. P. Brooks, Jr., *Architecture of the IBM System/360*. IBM Journal, pages 87–101, April 1964.
- Translation Lookaside Buffer Introduction in 1965 :
  Webb T. Comfort. *A Computing System Design for User Service*. In Proceedings of the Fall Joint Computer Conference, pages 619–626, 1965.

Address Translation History Glitches

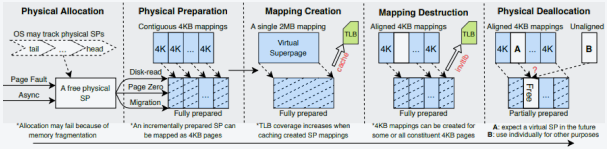
- PowerPC 601 (1993), Alpha 21264 (1998) : 8 KiB pages
- ARM based Apple Mx (2022) : 16 KiB pages

Virtually all ISA have a minimal page size of 4 KiB

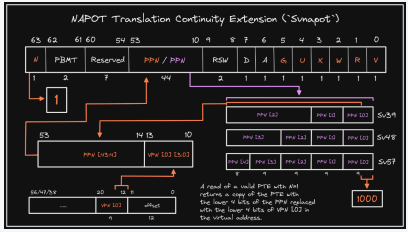
Address Translation Primer

Tricks to deal with small pages : Burden on the OS!

- Traverse only part of the page-table hierarchy :
  4 KiB, 2 MiB, 1 GiB page sizes, typically



- Merge TLB entries for contiguous 4 KiB pages :



Empirical approach to determining appropriate page size

Reference TLB Architecture : TOP500 recent leader Fugaku Supercomputer

Table 1: Fujitsu A64FX TLB specifications

		Association method	Number of entries	Replacement algorithm
Data TLB	L1	Fully associative	16	FIFO
	L2	4-way	1 024	LRU
Instruction TLB	L1	Fully associative	16	FIFO
	L2	4-way	1 024	LRU

Functional Simulation

- QEMU cross-ISA simulator (RISC-V on x86-64) tracing all memory accesses
- TLB simulator fed by QEMU traces

Empirical approach to determining appropriate page size

Multicore Benchmarks

- NPB (HPC), PARSEC, SPLASH3 (Parallel) and SPEC (Sequential) (b)

Parameters

- Number of cores (n) 32, 64 and 96 (HPC and Parallel)
- 4 KiB to 256 KiB page size (p)

Metrics

- TLB miss rate (mr)
- Memory bloat (mb) :
  Amount of memory reclaimed by the OS over actually used memory (normalized)
- Mixing of both metrics is a wild measure of «optimality»

Empirical approach to determining appropriate page size

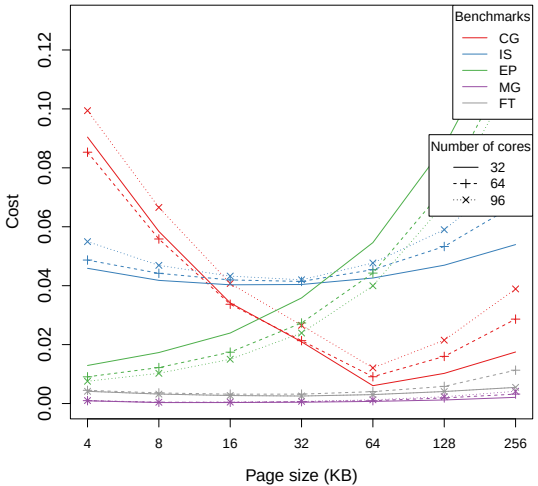
Tradeoff between miss rate and memory overhead ($0 \leq w \leq 1$)

For one program b on n cores :
Cost function (w) : $J_{n,b}(p) = w \cdot mr_{n,b}(p) + (1 - w) \cdot mb_{n,b}(p)$
«Optimal» page size : $p^k_{n,b} = \text{argmin}_{p \in P} J_{n,b}(p) = \{p \mid J_{n,b}(p) = \min_{\pi \in P} J_{n,b}(\pi)\}$

For all programs of a benchmark on all numbers of cores :
Cost function (w) : $\hat{J}_{n,b}(p) = \frac{J_{n,b}(p)}{J^{\max}_{n,b}}$
«Optimal» page size : $\hat{p}^k = \text{argmin}_{p \in P} \sum_{b \in \{\text{benchmarks}\}} \sum_{n \in \{\text{cores}\}} \hat{J}_{n,b}(p)$

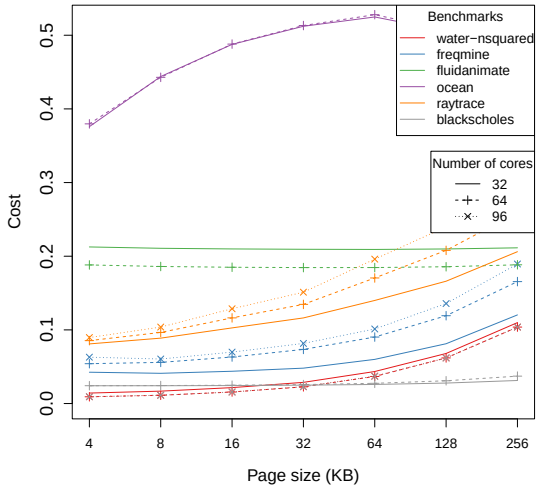
Experimental results

With $w = 0.5$
NAS Parallel Benchmarks (HPC)



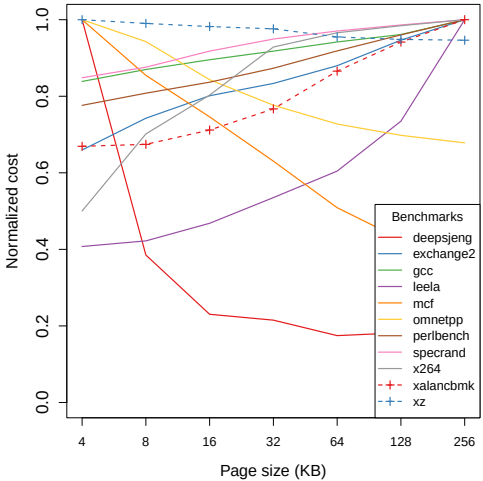
Experimental results

With $w = 0.5$
Parsec (Parallel programming)



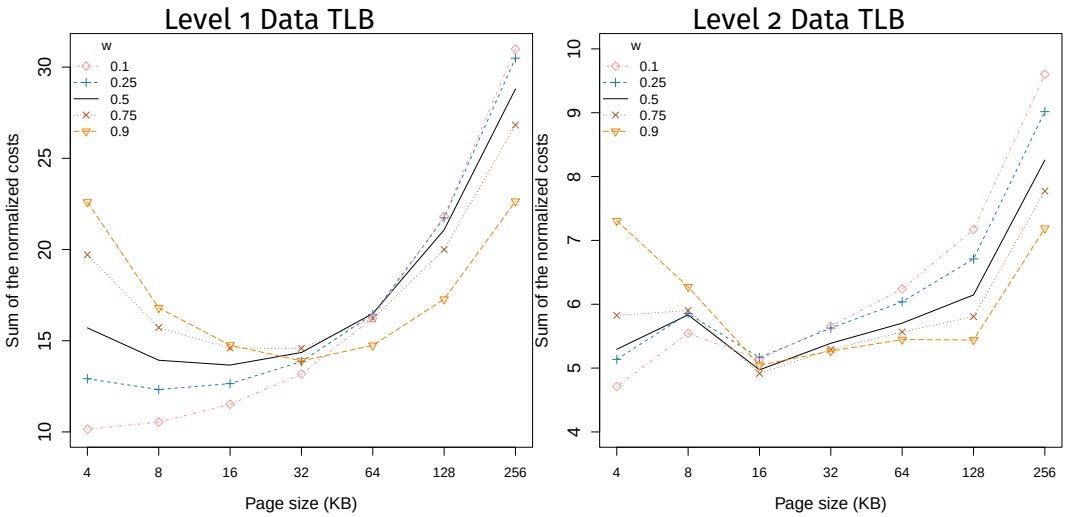
Experimental results

With $w = 0.5$
SPEC CPU 2017 (Sequential programs)



Experimental results

For all Parallel Benchmark programs, varying *w*



Takeaway

One (page) size does not fit all !

«Optimal» page size is application class (w) dependent

- 4 KiB : useful for embedded, limited memory, devices, $w < 0.5$
- 16 KiB : right choice for general purpose computing , $w \approx 0.5$
- 32 KiB : appropriate for high performance computing , $w \gg 0.5$
- Going over 32 KiB only very rarely useful

Thanks for listening

Q&A

Questions, feedback and discussions welcome!

Thanks to :

- The ANR for financing in part that research though grant N° ANR-21-CE25-0016
- The members of the Maplurinium Project