Concurrent Support of Multiple Page Sizes on a Skewed Associative TLB

- Andre Seznec, Member IEEE
IRISA/INRIA, Campus de Beaulieu, 35042 Rennes, France

A review by:
Hareesh Nagarajan
hnagaraj@cs.uic.edu
Let us dissect the title

“Concurrent Support of Multiple Page Sizes on a Skewed Associative TLB”

- What is a TLB?
  - Translation Lookaside Buffers (more in the next slide)
- Why multiple page sizes?
  - Different applications work best with different page sizes.
  - Small page size v/s Large page size
- Why concurrent?
  - Processes can also benefit from different page sizes (E.g. Code segment in one page, Data segment in another)
An overview of TLB

- The TLB is a **small, specialized cache** that **speeds up memory accesses** by storing recently used address translations.
- A TLB can be viewed as a **hash table** (that only has the capacity for holding a subset of the actively used address translations.)
- To increase the performance of a TLB (other than making it larger) is to:
  - **Increase associativity**: typically performing multiple comparisons in parallel to avoid slowing down lookups; however, this is expensive in terms of chip area and energy consumption.
  - **Skewed associativity**: using several different hash functions for parallel lookups, has been demonstrated to yield good results with less parallelism and therefore at a lower cost.
More on the TLB...
And some more...

Example: TLB with 64 (100 octal) slots. Suppose the following virtual pages are referenced (octal): 621, 2145, 621, 2145, ... 321, 2145, 321, 621.
Why not just use associative TLB for multiple page sizes?

- When a single page size is used there is no ambiguity in defining the page number for a virtual address (Since, all pages are the same size).

(TAGtobechecked & MASKmapped) is compared with TAGmapped
Example

- If 2 page sizes are allowed 8KB($2^{13}$) and 4MB($2^{22}$)
- If 4GB($2^{32}$) of memory is addressable
- Number of 8KB pages: $2^{32}/2^{13} = 2^{19}$ pages
- Number of 4MB pages: $2^{32}/2^{22} = 2^{10}$ pages

- TAGtobechecked = say $2^{12}$ (TAGtobechecked is 19 bits long)
- MASKmapped = say 19 (As we are referring to a 8KB page)
- TAGmapped = $2^{12}$
- TAGtobecheck & $2^{(MASKmapped+1)}-1$ compared with TAGmapped
  - 0000 0001 0000 0000 0000
  - & 1111 1111 1111 1111 1111
  - Yes! It is equal to TAGmapped. But would not be equal to TAGmapped if the page was of size 4MB!
Downsides

- As you can see by the picture below - 512 small 8KB pages will conflict with 1 - 4MB page.
- Example:
The idea behind the skewed associative cache was extended by the author. An X-way set associative cache is built with X distinct banks. The memory block at address D may be physically mapped onto physical line f(D) of any of the distinct banks.

Figure 1: 3 data blocks conflicting for a single set on a two-way set-associative cache. A, B and C compete for only two locations.
Skewed Associative Cache

(... a different mapping is used for each data bank)

Figure 2: A, B and C compete for the same location in bank 0, but can be present at the same time, as they do not map to the same location in bank 1
Properties of a skewed associative TLB

- **# 1**: For a given 'l' way, for a given virtual address \( V(A) \), there exists entry 'E' in way l and there exists one possible page size 's', such that: \( F(V(A), s) \rightarrow \{E, \text{ in way } l\} \)
- **# 2**: Distinct ways of the caches are indexed with different hash functions \( F(x, y) \)
- **# 3**: Any TLB location can map pages of any possible size

The **difficulty** with property 1 is that the virtual page number associated with a virtual address depends on page size
The page size function

- With the page size function - S - if a virtual page V(A) is mapped to a given way I from the TLB then its page size is known.
- S depends on the chain X(A) of the 3 bits 21, 22, 23 from the address A (LSB: Bit 0)
- The ALPHA ISA supports 4 page sizes: 8K, 64K, 512K & 4MB.

- ways 0 and 4: 8K for X(A)=0,4; 64K for X(A)= 1,5; 512K for X(A)= 2,3; 4M for X(A) = 6,7.
- ways 1 and 5: 8K for X(A)=1,5; 64K for X(A)= 0,4; 512K for X(A) = 6,7; 4M for X(A) = 2,3.
- ways 2 and 6: 8K for X(A)=2,6; 64K for X(A)= 3,7; 512K for X(A)= 0,1; 4M for X(A) = 4,5.
- ways 3 and 7: 8K for X(A)=3,7; 64K for X(A)= 2,6; 512K for X(A) = 4,5; 4M for X(A) = 0,1.

Bits 21, 22 are used for 8K and 64K page sizes
Bits 22, 23 are used for 512K and 4M page sizes
• **Properties of the page function:**
  • In each way $\frac{1}{4}$ of virtual address space is mapped with 8KB Pages, another $\frac{1}{4}$ with 64KB and so on.
  • For a page size s, each way maps $\frac{1}{4}$ of total virtual space
Indexing Function

Now, since we have 8 ways we need 8 indexing functions for the skewed associative TLB

- For $0 \leq i < 4$; $f_i(A) = g(A, S(A, i))$
- For $4 \leq i < 8$, $f_i(A) = G(A, S(A, i))$

The idea:
- If $f(i<4) - 4$ fixed 7 Bit strings are extracted from the virtual address and are presented to the 4-4 Mux. Control to mux is from bits 21, 22, 23
- If $f(i\geq4) - We need to perform an extra XOR
Conclusion

- Concurrently supporting multiple page sizes with a single process allows a **better management of the VM** than supporting single page size.
- Currently – If architecture supports multiple page sizes
  - Processor implements either Fully associative TLB or distinct TLB for each of the page sizes
- It is shown that a 2-way skewed associative TLB in practice **behaves close** to the 8-way set associative TLB
- Index computation **overhead** is minimal.
In skewed-associative models, the sets of possible placements for two entries may only partially overlap. Thus, the current placement of entries will limit future replacement possibilities.

This is an inherent inflexibility in traditional skewed-associative models, since we cannot predict which placements will enable the most desirable future replacement choices.

Thorild Selén from Uppala Universitet demonstrates how the performance of skewed-associative TLB models can be enhanced further by reorganisation (moving old entries around to allow for more efficient replacements.)
Any questions?

Thank you.