

Adaptive Distributed Software Virtual Memory for Raw

Csaba Andras Moritz
andras@lcs.mit.edu

Motivation

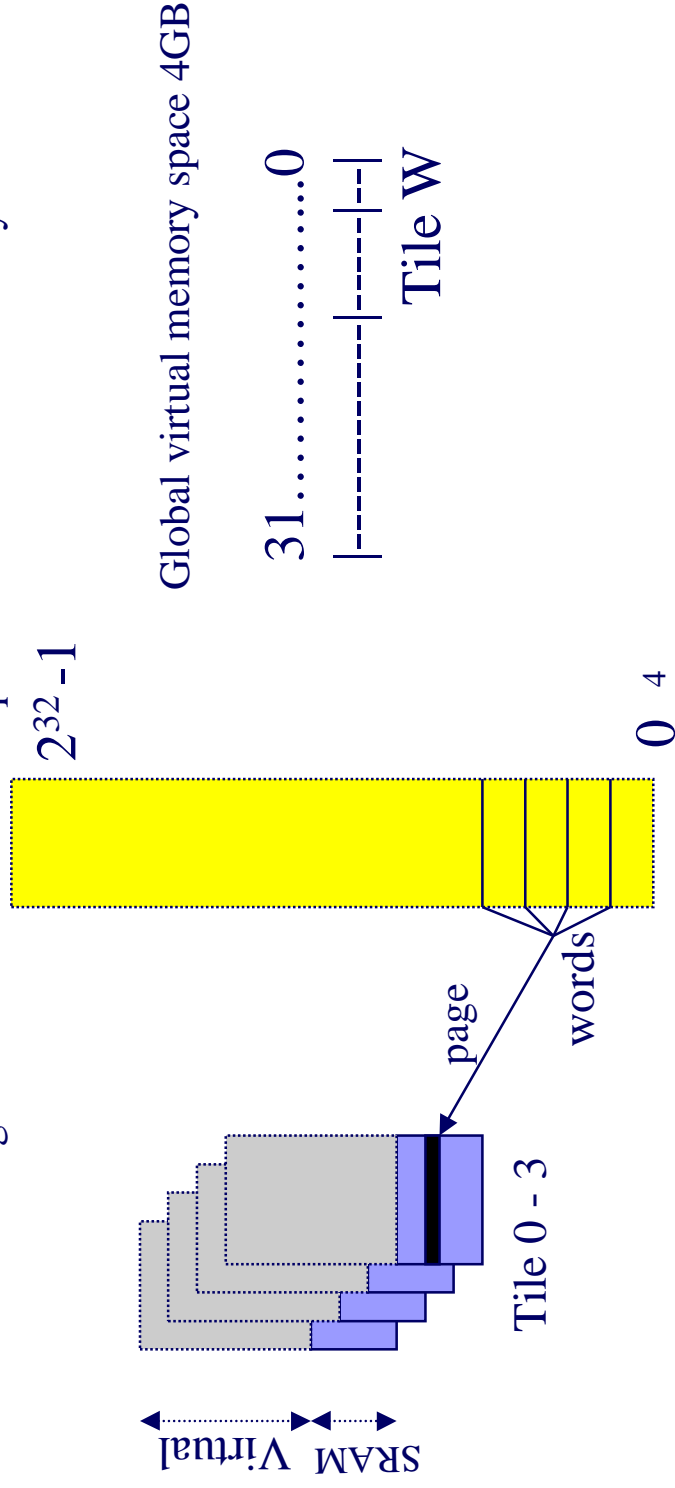
- ◆ Solution for finite on-chip memory limitation on Raw
 - **Data** (run applications with large data-sizes)
 - Instructions

Issues

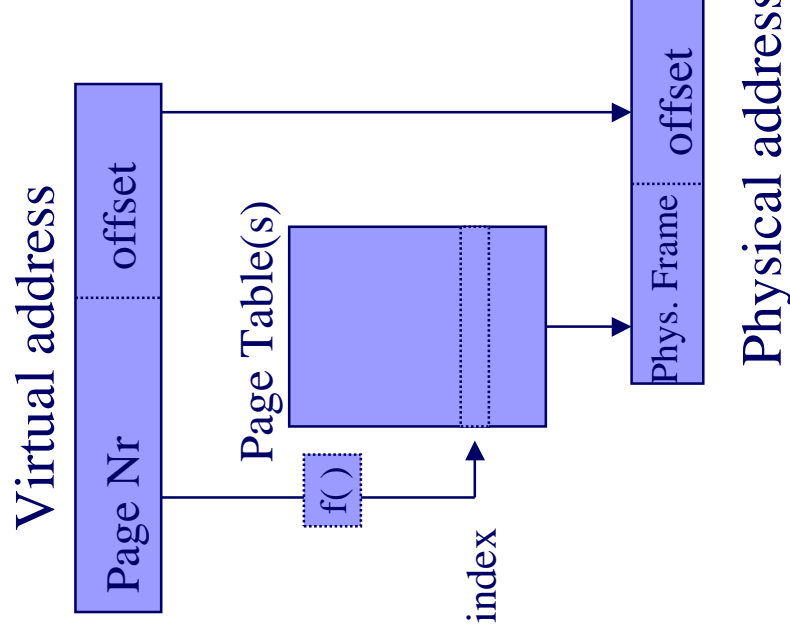
- ◆ What is a page? How is the memory partitioned?
- ◆ Static, dynamic, speculative references
- ◆ Address translation
 - Page Table (PT) organization, PT size, page size
 - Run-time VM check for load/store
- ◆ Page-fault(s), external memory, paging
- ◆ Compiler optimizations

A possible approach

- A page is a *contiguous* address-range on a tile, but it is interleaved in the global virtual address space
- Each tile gets its *private* fraction of the global virtual memory to check on.
- Translation happens at the *destination* tile and is local.
- No more global communication required for the virtual memory.



Address translation overview



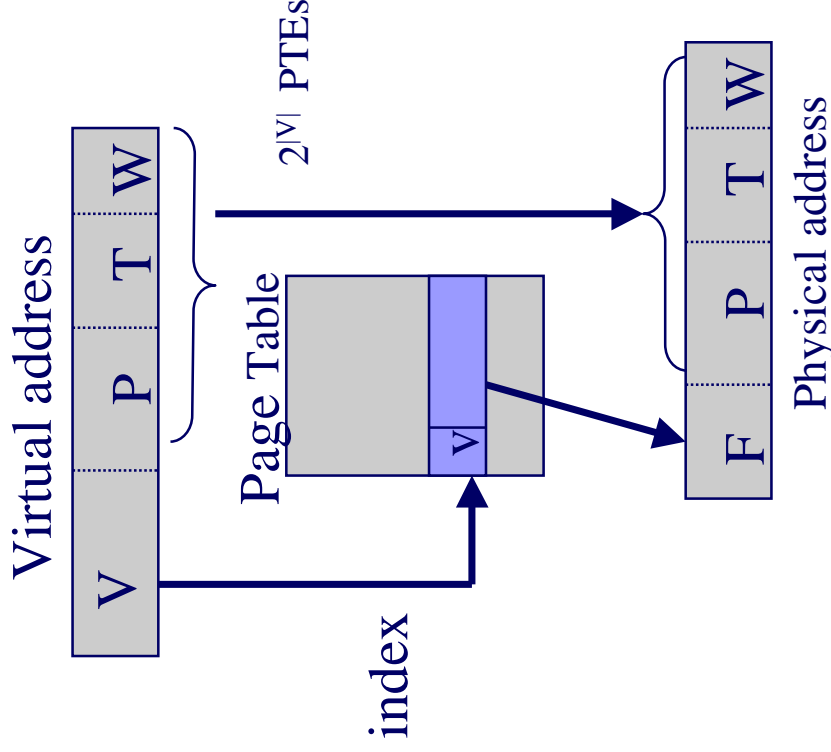
Page table (PT)/mapping types:

- Fully mapped
- Direct mapped
- Direct mapped inverted PT
- Two-way set associative with Inverted PT
- Multi-level

Tradeoffs:

- hit-time vs PT size
- hit-time vs hit-rate

Fully mapped



Notation:

W = Word offset

T = Tile

P = Page offset in words

V = Virtual Page Number

F = Physical Page Number

v = valid bit

|V| = bits in **V**

PTE = Page Table Entry

Page size = $2^{|P|}$

PT size = $2^{|V|}$ words

Example:

1.) 32 bit addr, 1KB page size,

16 tiles => PT size = 1 MB

2.) 24 bit addr (16 Mbyte),

PT size = 4KB

Adv: impl in software =>

fastest load hit-time=

4(6) extra cycles

Direct mapped

Notation:

W = Word offset

T = Tile

P = Page offset in words

V = Virtual Page Number

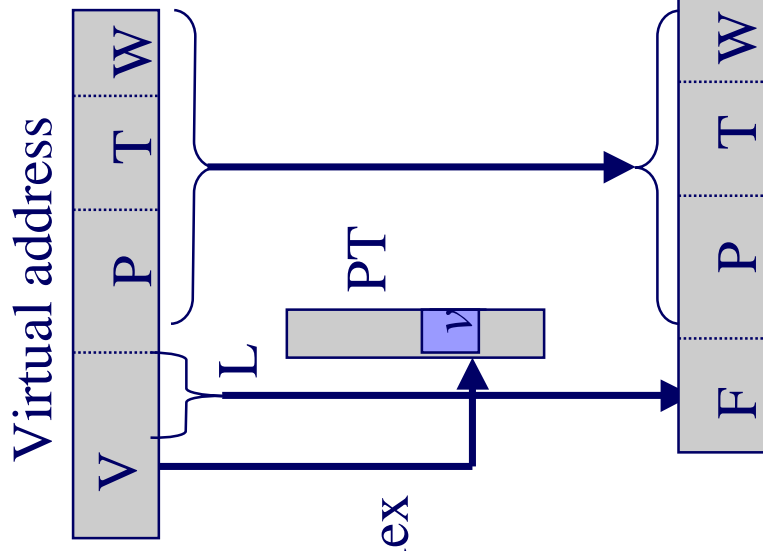
L = Low order V bits in SRAM space

F = Physical Page Number

v = valid bit

|V| = bits in V

index



Physical address

Page size = $2^{|P|}$ words, $2^{|V|}$ PTEs

PT size = $2^{|V|}$ bits,

Example:

32 bit addr, 32KB/tile SRAM

1KB page size,

16 tiles => PT size = 32 KB

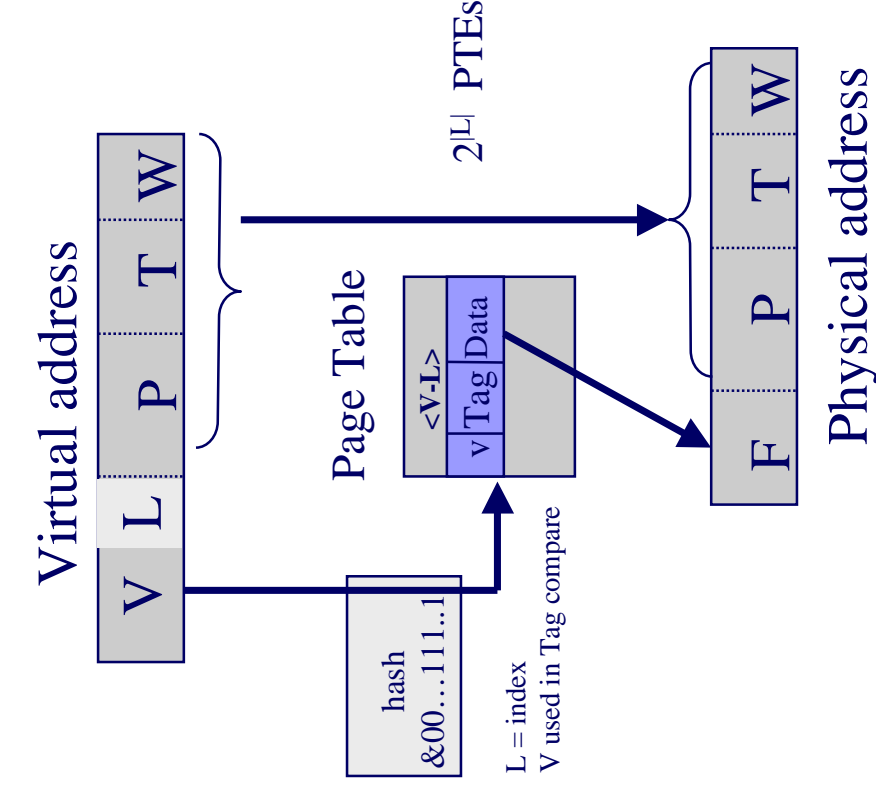
4KB page, PT size = 8 KB

Adv: reduced PT size, but still require larger page size

Disadv: PT size reduction only by const.

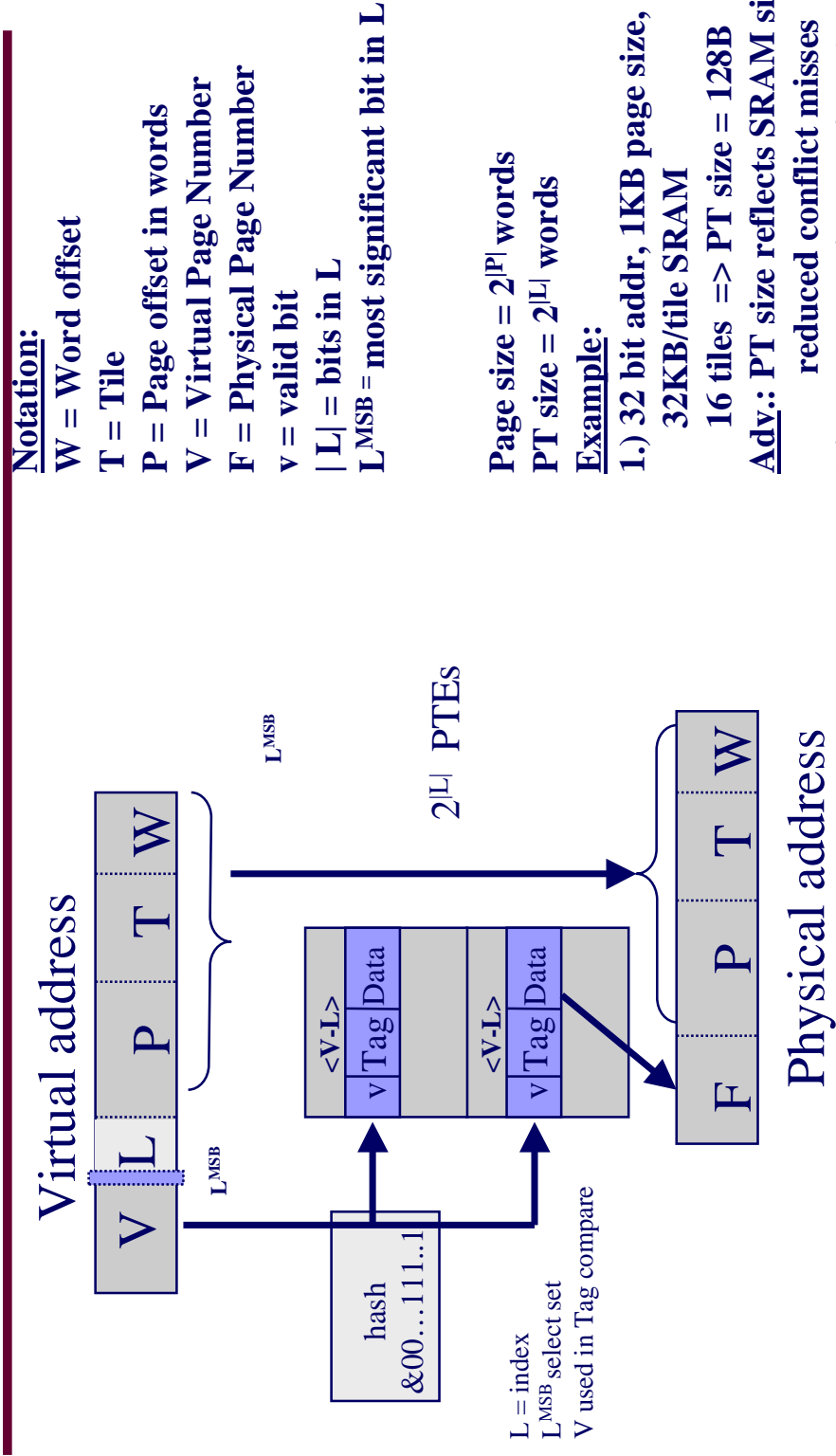
2 extra instructions to extract valid bit, conflict misses

Inverted PT, Direct mapped



- Notation:**
- W** = Word offset
 - T** = Tile
 - P** = Page offset in words
 - V** = Virtual Page Number
 - F** = Physical Page Number
 - v** = valid bit
 - |L|** = bits in L
 - PTE** = Page Table Entry
- Page size = $2^{|P|}$ words
 PT size = $2^{|L|}$ words
- Example:**
- 32 bit addr, 1KB page size, 32KB/tile SRAM
- 16 tiles => PT size = 128B
- Adv.:** PT size reflects SRAM size
- Disadv.:** conflict misses, extra instructions
- hashing
 - Tag comparison
- Total: 7(9) instructions**

Inverted PT, 2way s.a. mapped



Notation:

- W = Word offset
- T = Tile
- P = Page offset in words
- V = Virtual Page Number
- F = Physical Page Number
- v = valid bit
- |L| = bits in L
- L^{MSB} = most significant bit in L

Page size = $2^{|P|}$ words
 PT size = $2^{|L|}$ words

Example:

- 1.) 32 bit addr, 1KB page size, 32KB/file SRAM
- 16 tiles => PT size = 128B

Adv.: PT size reflects SRAM size

reduced conflict misses

Disadv.: extra instructions (total 8.5)

- hashing

- worst case 2 Tag comparisons

CheckVM&load() Fully mapped case

INPUT: r1 = PT base, r2 = virtual address

- 1: r3 = r2 >> (|P| + |T| + |W|); /* get virtual page number = PTE index */
- 2: r4 = [r1 + r3] /* load PTE, maybe two instructions on MIPS*/
- 3: r5 = r2 & VMASK /* get offset in page */
- 4: blz r4, HIT /* check valid bit */
- 5: continue with page-fault case

HIT:

- 5b: r6 = r4 & FMASK /* get physical page number from entry*/
- 6b: r7 = [r6 + r5] /* load, maybe two instructions */

Using invalid bit (branch on page-fault) instead of valid bit eliminates
line 5b. TOTAL HIT-COST 4(6) instructions.

Summary software translation

- Optimize hit-case:
 - » Fully-mapped: 4(6) extra instructions for hit-case
 - » Direct-mapped w reduced PT size: 6(8)
 - » Inverted PT, direct-mapped, reduced PT size: 7(9)
 - » Inverted PT 2-way, better hit-rate, reduced PT size: 8.5(10.5)
- *Adaptive* address translation idea:
 - » Use the first scheme for small problems and if compiler can estimate memory space required by the application.
 - » Use scheme like the Inverted PT with 2-way set associative mapping for large problem sizes

Page transfer

- DRAM partitioned in logical blocks each for a tile virtual address space, each block with several memory banks for fast access.
- Dynamic messages: general approach
- Statically scheduled for some cases:
 - » idea of ghost memory(tiles) for completely static programs

–

Compiler optimizations

- Batch-checking (affine array accesses)
- Bulk-paging (reduce latency by multiple-pages)
- Footprint pre-fetch (pre-fetch if footprint is known)
- Page-size estimation
- Reference type based:
 - » Stack, Modulo Unrolled, Scalars
- Cache optimizations may apply, e.g., loop fusion, loop interchange,...
- Other Software: ILP, CSE