How Processor-Memory Integration Affects the Design of DSMs

Liuxi Yang, Anthony Nguyen, and Josep Torrellas

University of Illinois

http://iacoma.cs.uiuc.edu/iacoma/

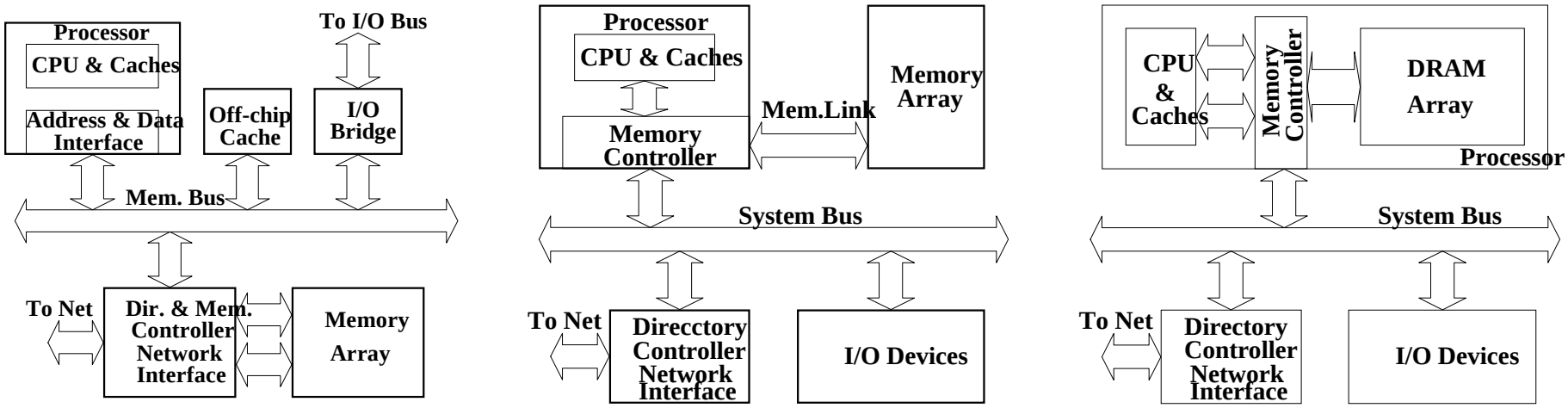
l-yang3, anguyen, torrella@cs.uiuc.edu

The Point

- Implications of processor-memory integration on DSMs
  - → Redesign the nodes
  - → Reorganize the whole machine
- How?
  - → On-chip memories configured as caches
  - → Move directory controller off-chip
  - → Computing / non-computing nodes
  - → Non-computing nodes
    - Support cache coh protocol
    - Manage memory

    〉 In software

Trend in DSM Node Organization



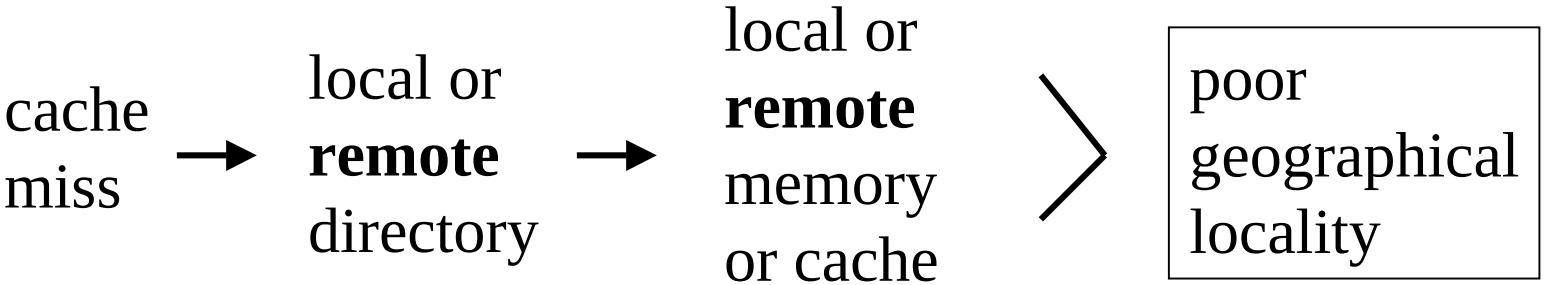
•Progressive processor-memory integration

How to Benefit from the Integration?

Exploit the close P-M coupling as much as possible

- Easy in uniprocessors:

- Hard in traditional NUMA DSMs:



How Often Remote?

- Remote directory: given **N** nodes, **(N-1)/N** addresses are in remote directories

  → Improved data layout?

- Remote memory/cache: start-up accesses
  cache overflows
  coherence activity

Our Solution

- Each block in memory has state (I,S,Ex) + address tag

  → skip dir check

- Memory treated as cache (data replication + migration). Not COMA

  → no remote mem due to overflow

- Move directory controller off-chip

Part of DRAM addressed as plain memory for private data

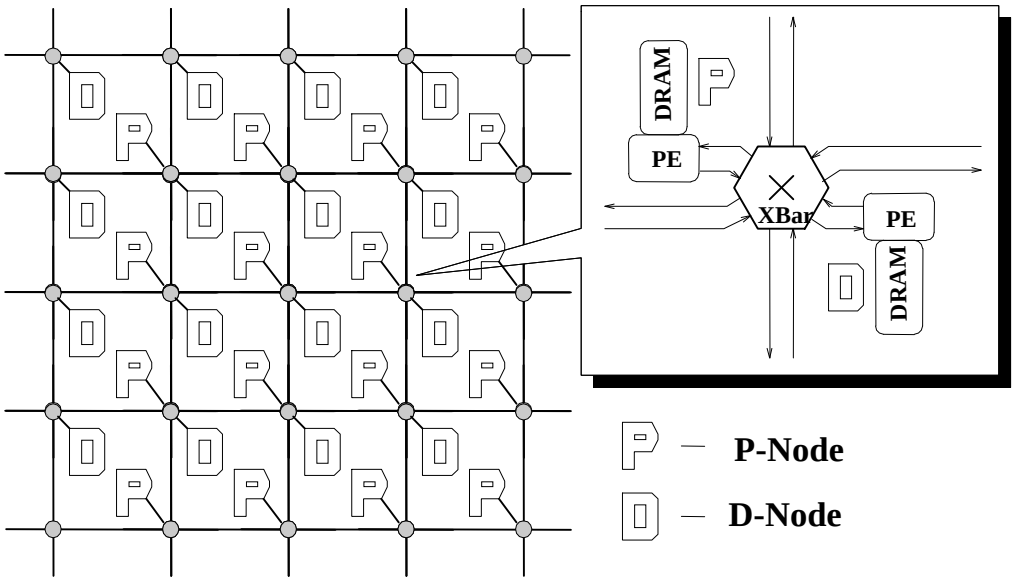
Memory as cache is useful for uniprocessors too:
for off-chip memory expansion

Moving Directory Controller Off-Chip

- Why?
  - Local directory is usually accessed by remote requests
  - On-chip directories are useless for uniprocessors
  - Allows varying the # of directories depending on appl
- Little is lost → DRAM cache: few dir accesses
  → Most dir accesses due to coh: remote
- Want to use off-the-shelf

  Directory controller as a separate node with same P+M hardware as the computing nodes

Resulting Machine



Usually several P per D

- Processing Node (P): Compute; Use memory as cache
- Directory Node (D):
  - Keep distributed directory
  - Execute coherence protocol (keep Ps coherent)
  - Memory used as the only mem for sh data in machine

Resulting Machine

D-nodes have relatively little to do beyond startup

- Modest # dir accesses:

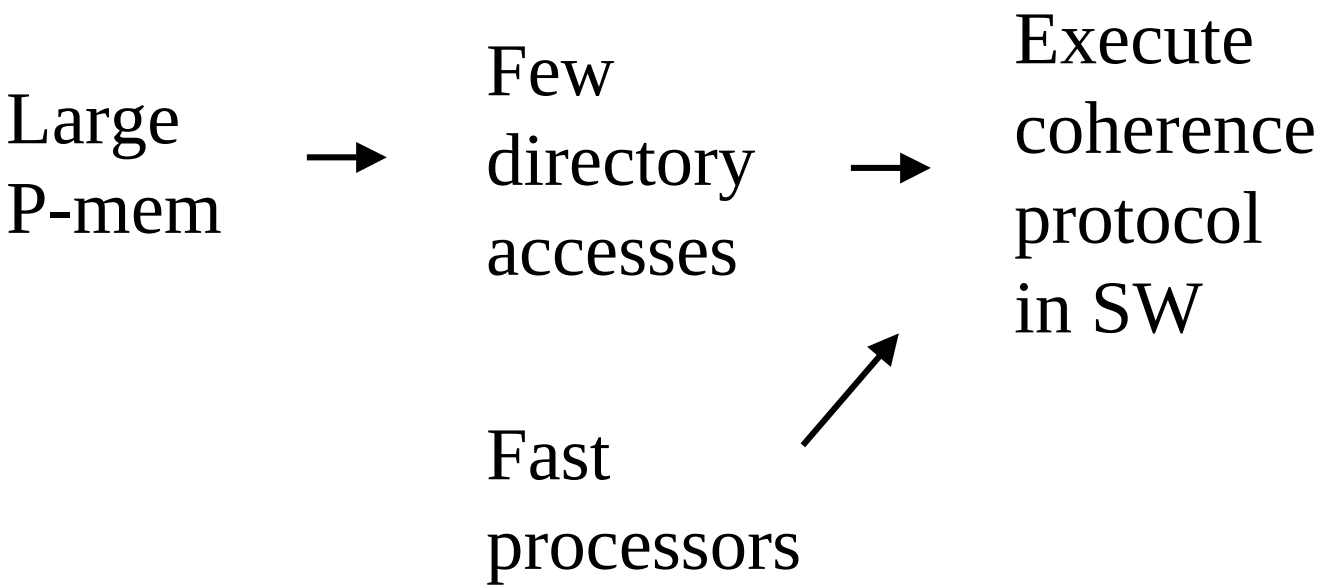
  → Little P overflow; only coherence

- Modest # of line insertions/extractions from memory

  → Few exclusive lines displaced from P; only coherence

Function of D-Nodes (I)

•Directory handling:



Function of D-Nodes (II)

•Manage the D-memory:

Only overflow lines in D-memory → Sparse storage → Save line in any location in D-node
Keep linked-lists
SW-managed memory

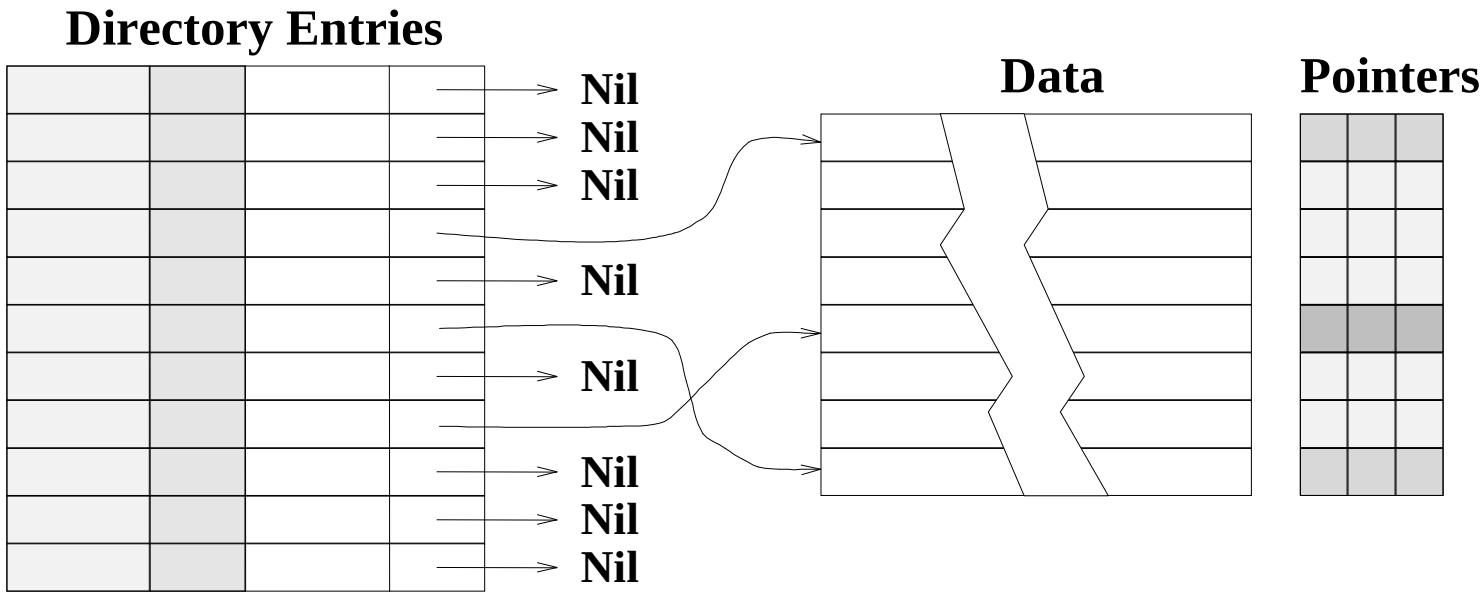
Little P-mem overflow ↑

Fast processors ↑

↓

Compacted storage: several P-nodes/D-node

Resulting D-Node Directory/Memory



- Directory + memory state is cached in the caches of D-node

Further Optimization: Row Tag Buffer

- Goal: speedup access to large on-chip DRAM
- How? Eliminate redundant RAS cycles

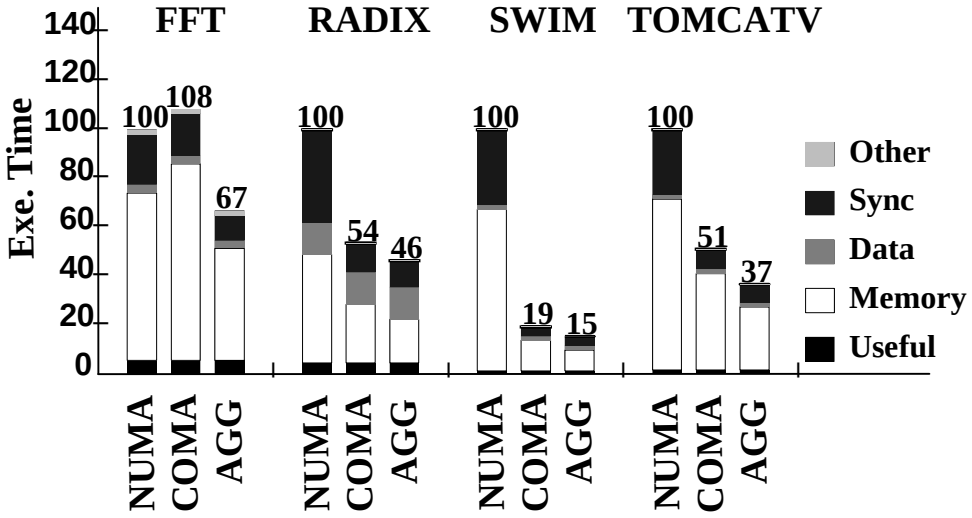
(1) Add buffer to memory controller w/ tags of rows currently in the memory row buffers

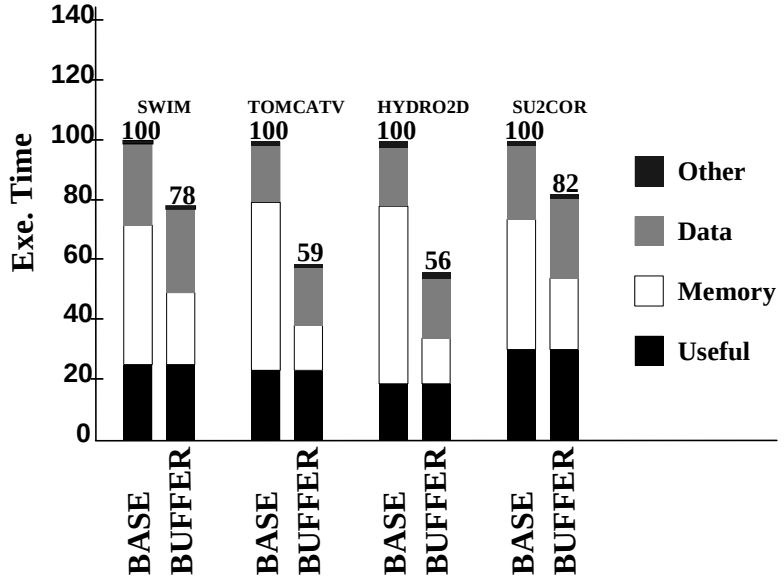
→ Checked during the cache access

→ In miss: no issue RAS if hit in Row Tag Buffer

(2) Sequential pretouch:
bring next row to the row buffer too

Evaluation





- 42% average faster than futuristic NUMA/COMA

- Row Tag Buffer: average speedup > 32%

Conclusions

- Proposed design
  - → On-chip memories configured as caches
  - → Move directory controller off-chip
  - → Computing / non-computing nodes
  - → Non-computing nodes
    - Support cache coh protocol
    - Manage memory

    } In software

- Memory as cache is useful for uniprocessors too: for off-chip memory expansion