Death Match ’92: NUMA v. COMA

CS258
By Nemanja Isailovic

In this corner: NUMA

- Each node has a portion of main memory and a directory corresponding to that portion.
- Each memory address has a home node. (It can exist in any cache, but not in any other memory.)
- Requests for data are sent to the home node.
- Reassignment of home node can be done by OS or user code in page-sized chunks only.
- Examples: DASH and Alewife

In this corner: COMA

- Each node has a portion of main memory.
- Memory acts as a cache, so there is no home node.
- Data can be transferred or replicated between memories in cache-line-sized chunks.
- With no home node, a hierarchical directory structure is used to locate data.
- Replacements are so easy to do that they don’t need to be explained in any COMA paper.
- Examples: DDM and KSR1

Basic Comparison

- Advantages of COMA
  – Could reduce average cache miss latency due to improved locality
- Disadvantages of COMA
  – Could increase average cache miss latency due to hierarchical directory structure
  – Replacements can be tricky
Let’s Get Theoretical

- How will each perform on different types of cache misses?
- Cold Misses: COMA is likely to do worse, since the remote access latency is worse.
- Coherence Misses: Data is guaranteed to be on a remote node, so COMA will certainly do worse. However, if combining happens to work well, COMA may do better.
- Capacity and Conflict Misses: In COMA, data is likely to be in local attraction memory due to migration and replication. In NUMA, it may not be. So COMA will likely perform better.

Let’s Get Theoretical II

- How should application miss rate affect things?
- Low Miss Rate: NUMA and COMA should be similar.
- High Miss Rate with Mainly Coherence Misses: COMA will do worse since we know that NUMA performs far better on coherence misses.
- High Miss Rate with Mainly Capacity Misses and Fine-Grained Data Access: With many nodes accessing a single page, NUMA will have a large number of remote requests, so COMA should do better.
- High Miss Rate with Mainly Capacity Misses and Coarse-Grained Data Access: NUMA may be able to migrate pages effectively and perform almost as well as COMA.

The Simulator

- 16 nodes
- one processor per node
- cache line size of 16B
- processor cache size of 4KB
- NUMA network is a 4x4 wormhole-routed synchronous mesh (16 bits wide), clocked at 100MHZ
- COMA network (for the directory hierarchy) has a branching factor of 4, with 32-bit links and synchronous transfers, clocked at 50MHZ

Issues with the Simulator

- Tiny processor cache (4KB): Their claim is that they had to use smaller working sets than normal, so they used a smaller cache. (Note: Capacity misses favor COMA.)
- Infinite attraction memories are used for COMA. Again, replacement is ignored.
- Only simulates references to shared data (not instructions or private data).
- Combining is modeled, but contention at hierarchical directories is not.
Results

Other Notes

Page migration helps in a select few algorithms that use coarse-grained data accesses (Figure 5).

Page size affects page migration effectiveness (Table 4).

Effective initial placement of pages in NUMA can result in large speedups (Table 5).

NUMA starts beating COMA across the board as the processor cache size is increased, since capacity misses are reduced (Table 6).

COMA-FLAT

- Each block has a home node. The directory at the home node keeps track of all copies of the block. (NUMA)
- The attraction memory on a node is not reserved only for data in that directory. Other blocks can migrate there (like in a cache). (COMA)
- Data is transferred between attraction memories in cache-lines, not in pages. (COMA)

More COMA-FLAT

- In addition to the home node, each block has a current Master node, which is much like the Owner in MOESI.
- A request for a block is sent to the home node. The home node redirects it to the current Master. The Master replies to the requester, while the home node updates its own sharing list as necessary.
- Blocks can be Invalid, Shared, Master-Shared or Exclusive. Master-Shared and Exclusive can exist only on the current Master node.
Conclusions

- COMA handles capacity misses more efficiently due to migration and replication of small blocks of data.
- NUMA handles coherence misses more efficiently due to latency through COMA’s directory hierarchy.
- Good initial placement and smart migration of pages can give NUMA the edge in all but the most fine-grained data accesses.
- COMA is down! 1...2...3...4...5...6...7...8...9...10
- The victory goes to NUMA! The crowd goes wild.