CS533: Cache Coherence (II)

Josep Torrellas

University of Illinois in Urbana-Champaign

January 29, 2015
Sparse Directories

Since total \# of cache blocks in machine is much less than total \# of memory blocks, most directory entries are idle most of the time.

Example: 2 MB cache, 256 MB memory/PE \rightarrow 99\% idle

Sparse directories reduce memory requirements by:
1. Using single directory entry for multiple memory blocks
2. Dir-entry can be freed by invalidating cached copies of a block
3. Main problem is the potential for excessive dir-entry conflicts
4. Conflicts can be reduced by using associative sparse directories
Performance of Sparse Directories

- Figure 11 in Gupta et al. paper
- Figure 12 in Gupta et al. paper

Note: size factor is: Dir Entries / Cache Entries
Linked-list Schemes

- Keep track of PEs caching a block by linking cache entries together
- Scalable Coherent Interface (SCI) is the most developed protocol: uses doubly-linked list for chaining cache entries together
Implementation

- For each line in memory: a tag identifies the first processor in the sharing list
- Tags in the processor caches identify others in the sharing list
- Lists are unbounded in length, dynamically created, pruned, and destroyed
Normal operation: line is in memory or in cache

Head administers the return of dirty data to memory. Differentiates between clean, dirty, and number of sharers

Hard to verify

There is a base protocol and a set of performance enhancements

Case for a singly-linked list:
  - Advantage: space
  - Disadvantage: traversal
Summary of Linked-List Protocols

Advantages
- Directory memory needed scales with the number of PEs
- Distribute traffic/communication more

Disadvantages
- Requires directory memory to be built from SRAM (same as cache)
- To perform invalidation on write, need to serially traverse caches of sharing PEs (long latency and complex)
- Cache replacements are complex as both forward and backward pointers need to be updated
- In base protocol, read to clean data requires 4 messages (first to memory and then to the head cache) as compared to 2 messages in other protocols
Hypothesis: On a write to a shared location, with high probability only a small number of caches need to be invalidated

- If the above were not true, directory schemes would offer little advantage over snoopy schemes
- Experience tends to validate this hypothesis
Types of Shared Data

1. Code and read-only objects
   - No problems as rarely written

2. Migratory objects
   - Even as the number of PEs scale, only 1-2 invalidations

3. Most-read objects
   - Invalidations are large but infrequent, so little impact on performance

4. Frequently read/written objects (e.g. task queue structs)
   - Invalidations usually remain small, though frequent

5. Synchronization objects
   - low-contention locks result in few invalidations
   - high-contention locks need special support (SW trees, queuing locks)

6. Badly-behaved objects
Misses on Shared Data

- Single-word cache blocks
  - Cold effects
  - True sharing
- Multi-word cache blocks:
  - Cold effects
  - True sharing
  - False sharing

Difference? Caused by the prefetching effect of lines
# Effects of Multi-Word Cache Blocks

## Single-word blocks

<table>
<thead>
<tr>
<th></th>
<th>Misses</th>
<th>Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold</td>
<td></td>
<td></td>
</tr>
<tr>
<td>True Sh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>False Sh</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Multi-word blocks

<table>
<thead>
<tr>
<th></th>
<th>Misses</th>
<th>Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Sharing and Traffic

Miss Rate

Traffic

Words/Block

Infinite cache size

Words/Block

Josep Torrellas (UIUC)
Measurements

Miss Rate

Words/Block

Infinite cache size

Total misses
False sharing misses
Cold+True sharing misses

Josep Torrellas (UIUC)
How to Improve

Reduce false sharing:
- Synchronization variables
- Scalar with false sharing
- Heap allocation
- Record expansion (padding with blanks)
- Record alignment

Improve the spatial locality of true sharing:
- Scalars protected by locks
- Record alignment
Summary of Directory-Based Coherence

Directories offer scalable cache coherence

- No broadcasts
- Arbitrary network topology
- Tolerable hardware overheads
- Software understood (invalidations, false sharing)
Summary of Software Coherence

A very tough problem due to the fact that precise usage information is needed in the presence of:

- Memory aliasing
- Explicit parallelism