"An Evaluation of Directory Schemes for Cache Coherence"

Presented by Scott Weber

Outline

- Motivation and goals for directory schemes
- Directory schemes
- Schemes evaluated
  - Directory-based
  - Snoopy
- Insights from the evaluation
- Directory scheme alternatives
- Conclusions and Retrospective

Motivation and Goals

- Snooping does not scale past ~20 processors
  - Protocol depends on low-latency broadcasts
- Snooping interferes with the processor-cache connection
- Avoid broadcast nature of snooping
- Directory-based protocols should be competitive with snoopy protocols
- Access to a directory cannot be a bottleneck

Directory Schemes

- Tang scheme (Dir_nNB)
  - Multiple clean blocks, one dirty block
  - Copy of tags, dirty bits for each cache in directory

<u>Read miss</u>
check directory,
if dirty then write dirty back,
supply the data,
update directory.

<u>Write miss</u>
check directory,
if dirty then flush dirty back,
invalidate clean copies,
perform the write,
update directory.

<u>Write hit</u> (dirty)
if dirty bit set then write.

<u>Write hit</u> (clean)
if dirty bit not set,
notify directory,
invalidate clean copies,
update directory, update dirty bit.

Directory Schemes

- Modifications to Tang's scheme
  - Censier and Feautrier ($Dir_n NB$)
    - Vector of valid bits for each cache and dirty bit
    - Use the address of the data to access directory
  - Yen and Fu ($Dir_n NB$) refines C & F
    - Single bit in each cache to indicate only copy
    - When set, do not have to access directory
    - Requires more bandwidth to update single bits

Directory Schemes

- Archibald and Baer ($Dir_0 B$)
  - Four states:
    - block not cached
    - block clean in exactly one cache
    - block clean in an unknown number of caches
    - block dirty in exactly one cache
  - Requires broadcasts to do invalidations and write backs
  - Organization is still centralized
  - Easy to add more caches to the systems

Schemes Evaluated

- Classification
  - Dir(cache pointers)[Broadcast|No Broadcast]
- $Dir_1 NB$ – Tang (with n = 1) and variants
- $Dir_0 B$ – Archibald and Baer
- Alternatives attempt based on results
  - $Dir_i NB$, $Dir_n NB$, $Dir_1 B$, $Dir_i B$
- Write-Through-With-Invalidate (WTI)
- Dragon Update Protocol

Evaluation Methodology

- Trace-driven simulation
- Interested in memory traffic for CC (use ∞ cache)
- Machine independent metric

  Communication cost/memory reference
- Assume bus for comparison
- Measure event frequencies for various types of memory accesses (differ for each protocol)
- Weight event frequencies in terms of bus cycles
  - Non-pipelined shared bus model
  - Pipelined split address/data bus model

Evaluation of the Protocols

- Dir_1NB has a high read miss rate (5.18%)
  - Caused by read sharing among processes
  - Limitation of data only being in one cache
  - Dir_0B has a low read miss rate (0.62%)
- Dir_0B and WTI have same rates since they have same state changes on data in cache
- Dragon is dominated by write hits (updates)
- 36% of misses are coherency-related misses

Evaluation of the Protocols

- >85% writes to previously clean blocks cause invalidations in 0 or 1 caches
  - Motivates Dir_iNB, Dir_nNB, Dir_1B, Dir_iB
- Directory bandwidth similar to memory
  - Can distribute directory and memory to scale
- Estimating average memory access makes protocol bus cycles more equal
- Spin-locks on shared variables hurt Dir_1NB

Directory Scheme Alternatives

- Schemes introduced to decrease broadcasts
  - Dir_nNB Performs sequential invalidations
  - Dir_1B performs a single invalidation (common case) if broadcast bit is clear, otherwise broadcast
  - Dir_iNB and Dir_iB use limited number of ptrs
  - Limited broadcasts invalidate to cache subsets
    - 01XX01 encoding indicate subsets
- Schemes like these scale since new directory bits do not necessarily have to be added when scaling

Conclusions

- Bandwidth to directory is similar to bandwidth to memory
  - Distribute the directory and memory
  - Allows to scale with the number of processors
- Eliminates the inefficiency of broadcasts
  - Most blocks shared by 0 or 1 caches
  - Only need a few pointers in each directory entry
- Snoopy and broadcast protocols are competitive
  - Need to keep the number of spin-locks to a minimum

Retrospective

- Paper led to the development of DASH ($Dir_n NB$) prototype
- Concern at paper time was if snoopy and directory-based protocols were competitive
- Real issues
  - Scalability of coherence scheme
  - Implementation complexity
  - Overhead of coherence protocol
  - Performance with many processors
  - Implementing distributed directory coherence

Event Frequencies

Event Type	Schemes			
	$Dir_1 NB$	WTI	$Dir_0 B$	Dragn
instr	49.72	49.72	49.72	49.72
read	39.82	39.82	39.82	39.82
rd-hit	34.32	38.88	38.88	39.20
rd-miss(rm)	5.18	0.62	0.62	0.30
rm-blk-cln	4.78	-	0.23	0.14
rm-blk-drty	0.40	-	0.40	0.17
rm-first-ref	0.32	0.32	0.32	0.32
write	10.46	10.46	10.46	10.46
wrt-hit(wh)	10.19	10.25	10.25	10.36
wh-blk-cln	-	-	0.41	-
wh-blk-drty	-	-	9.84	-
wh-distrib	-	-	-	1.74
wh-local	-	-	-	8.62
wrt-miss(wm)	0.17	0.12	0.11	0.02
wm-blk-cln	0.08	-	0.02	0.01
wm-blk-drty	0.09	-	0.09	0.01
wm-first-ref	0.08	0.08	0.08	0.08

Bus Cycle Breakdown