Design of Parallel and High Performance Computing
HS 2017
Torsten Hoefler, Markus Püschel
Department of Computer Science
ETH Zurich

Homework 2

Cache addressing

In the following we will assume we are working with a bus-based multiprocessor machine with four processors. Each processor has its own direct-mapped cache. Each cache is 32 Bytes in size, organized in four sets. The memory in the machine is byte-addressable and the address width of the machine is eight bit.

Show how cache-addressing works on this machine. Which part of an address is used as tag, set and offset?

Solution

In a direct mapped cache each set contains one cache line. (In general k cache lines in a k-way associative cache, "direct mapped" is just special terminology for a 1-way associative cache.) Since we have four sets, there are also four cache lines in this case. As our cache is 32 Byte in total, each cache line therefore is 8 Bytes in size. To address each byte in such a cache line we therefore need three bits in the offset. To distinguish between the four sets, two bits are needed. The rest of the address is stored in the tag field.

MESI State Transitions

Describe what happens in the MESI protocol (bus traffic, state changes) if a processor experiences

1. a local read miss, while another cache holds a copy in exclusive state
2. a local read miss, while another cache has a copy in modified state
3. a local write hit, while the cacheline is in modified state
4. a local write hit, while the cacheline is in exclusive state
5. a local write hit, while the cacheline is in shared state in several caches

Solution

1. Processor requests data from memory, other processor sees the request. Both processors set the state of the cached line to shared.
2. Processor requests data from memory, other processor sees the request, The modified value is written back to memory. Both processors set the state of the cached line to shared.
3. locally cached value is updated, state remains modified.
4. locally cached value is updated and its state changes from exclusive to modified.
5. Modyfying processor broadcasts an invalidate message on the bus, other processors set their state to invalid, local value is updated and its state is set to modified.

Cache coherence

The machine described above uses the MESI protocol to maintain cache coherence. In the following $R_p(a)$ means that processor p reads one byte from the address a. $W_p(a) = v$ means processor p writes v to the memory location a. Addresses are represented in binary. Initially all cache lines are invalid and all memory locations contain 0.

Show the transitions of cache lines in the table below.

Action	P_0	P_1	P_2	P_3	Latency
$R_0(00000000)$					
$R_1(00000010)$					
$R_2(00000011)$					
$W_3(00000100) = 1$					
$W_3(00000101) = 2$					
$R_0(00000100)$					
$W_0(00000101) = 3$					

Assume a cache hit takes one cycle (both for read and write operations), transfering a cache line (either fetch or write back) takes 8 cycles, and a request to use a cache line exclusively (transition I - M) resp. a request to uprade to exclusive use (transition S - M) takes 2 cycles. What is the penalty of this code sequence over the best alternative execution sequence that you can think of?

Solution

All addresses in the given sequence concern the same cacheline.

Action	P_0	P_1	P_2	P_3	Latency
$R_0(00000000)$	E	I	I	I	8
$R_1(00000010)$	S	S	I	I	8
$R_2(00000011)$	S	S	S	I	8
$W_3(00000100) = 1$	I	I	I	M	2+8
$W_3(00000101) = 2$	I	I	I	M	1
$R_0(00000100)$	S	I	I	S	8+8
$W_0(00000101) = 3$	M	I	I	I	2+1

In total this takes 54 cycles.

If we reorder this sequence into $R_0(00000000)$, $R_0(00000100)$, $W_0(00000101) = 3$, $R_1(00000010)$, $R_2(00000011)$, $W_3(00000100) = 1$, $W_3(00000101) = 2$ it takes only 45 cycles.

False sharing

Write a program which demonstrates *false sharing*. Provide two versions of the same algorithm, one version where false sharing is likely to occur, one where it does not. Meassure the execution times of both versions to check if they differ significantly.