Lecture 25: Security, VM, Multiproc

Today’s topics:

- Security wrap-up
- Virtual memory
- Multiprocessors, cache coherence
Meltdown
Spectre: Variant 1

if (x < array1_size)
  y = array2[array1[x]];

x is controlled by attacker

Access pattern of array2[] betrays the secret

array1[] is the secret

Thanks to bpred, x can be anything

Victim Code
Spectre: Variant 2

Attacker code
Label0: if (1)
Label1: ...

Victim code
Label0: if (...)
... ...

R1 ← (from attacker)
R2 ← some secret

Label1:

Iw [R2]
Virtual Memory

- Processes deal with virtual memory – they have the illusion that a very large address space is available to them.

- There is only a limited amount of physical memory that is shared by all processes – a process places part of its virtual memory in this physical memory and the rest is stored on disk (called swap space).

- Thanks to locality, disk access is likely to be uncommon.

- The hardware ensures that one process cannot access the memory of a different process.
Virtual Memory
Address Translation

- The virtual and physical memory are broken up into pages

8KB page size

Virtual address
- virtual page number
- page offset 13

Translated to physical page number

Physical address
Memory Hierarchy Properties

- A virtual memory page can be placed anywhere in physical memory (fully-associative)

- Replacement is usually LRU (since the miss penalty is huge, we can invest some effort to minimize misses)

- A page table (indexed by virtual page number) is used for translating virtual to physical page number

- The page table is itself in memory
Since the number of pages is very high, the page table capacity is too large to fit on chip.

A translation lookaside buffer (TLB) caches the virtual to physical page number translation for recent accesses.

A TLB miss requires us to access the page table, which may not even be found in the cache – two expensive memory look-ups to access one word of data!

A large page size can increase the coverage of the TLB and reduce the capacity of the page table, but also increases memory waste.
**TLB and Cache**

- Is the cache indexed with virtual or physical address?
  - To index with a physical address, we will have to first look up the TLB, then the cache → longer access time
  - Multiple virtual addresses can map to the same physical address – must ensure that these different virtual addresses will map to the same location in cache – else, there will be two different copies of the same physical memory word

- Does the tag array store virtual or physical addresses?
  - Since multiple virtual addresses can map to the same physical address, a virtual tag comparison can flag a miss even if the correct physical memory word is present
Cache and TLB Pipeline

Virtually Indexed; Physically Tagged Cache
Bad Events

- Consider the longest latency possible for a load instruction:
  - TLB miss: must look up page table to find translation for v.page P
  - Calculate the virtual memory address for the page table entry that has the translation for page P – let’s say, this is v.page Q
  - TLB miss for v.page Q: will require navigation of a hierarchical page table (let’s ignore this case for now and assume we have succeeded in finding the physical memory location (R) for page Q)
  - Access memory location R (find this either in L1, L2, or memory)
  - We now have the translation for v.page P – put this into the TLB
  - We now have a TLB hit and know the physical page number – this allows us to do tag comparison and check the L1 cache for a hit
  - If there’s a miss in L1, check L2 – if that misses, check in memory
  - At any point, if the page table entry claims that the page is on disk, flag a page fault – the OS then copies the page from disk to memory and the hardware resumes what it was doing before the page fault
  ... phew!
Multiprocessor Taxonomy

- SISD: single instruction and single data stream: uniprocessor

- MISD: no commercial multiprocessor: imagine data going through a pipeline of execution engines

- SIMD: vector architectures: lower flexibility

- MIMD: most multiprocessors today: easy to construct with off-the-shelf computers, most flexibility
Memory Organization - I

• Centralized shared-memory multiprocessor or Symmetric shared-memory multiprocessor (SMP)

• Multiple processors connected to a single centralized memory – since all processors see the same memory organization → uniform memory access (UMA)

• Shared-memory because all processors can access the entire memory address space

• Can centralized memory emerge as a bandwidth bottleneck? – not if you have large caches and employ fewer than a dozen processors
Snooping-Based Protocols

- Three states for a block: invalid, shared, modified
- A write is placed on the bus and sharers invalidate themselves
- The protocols are referred to as MSI, MESI, etc.
Example

- P1 reads X: not found in cache-1, request sent on bus, memory responds, X is placed in cache-1 in shared state
- P2 reads X: not found in cache-2, request sent on bus, everyone snoops this request, cache-1 does nothing because this is just a read request, memory responds, X is placed in cache-2 in shared state

- P1 writes X: cache-1 has data in shared state (shared only provides read perms), request sent on bus, cache-2 snoops and then invalidates its copy of X, cache-1 moves its state to modified
- P2 reads X: cache-2 has data in invalid state, request sent on bus, cache-1 snoops and realizes it has the only valid copy, so it downgrades itself to shared state and responds with data, X is placed in cache-2 in shared state, memory is also updated
### Example

<table>
<thead>
<tr>
<th>Request</th>
<th>Cache Hit/Miss</th>
<th>Request on the bus</th>
<th>Who responds</th>
<th>State in Cache 1</th>
<th>State in Cache 2</th>
<th>State in Cache 3</th>
<th>State in Cache 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: Rd X</td>
<td>Rd Miss</td>
<td>Rd X</td>
<td>Memory</td>
<td>Inv</td>
<td>Inv</td>
<td>Inv</td>
<td>Inv</td>
</tr>
<tr>
<td>P2: Rd X</td>
<td>Rd Miss</td>
<td>Rd X</td>
<td>Memory</td>
<td>S</td>
<td>Inv</td>
<td>Inv</td>
<td>Inv</td>
</tr>
<tr>
<td>P2: Wr X</td>
<td>Perms Miss</td>
<td>Upgrade X</td>
<td>No response. Other caches invalidate.</td>
<td>Inv</td>
<td>M</td>
<td>Inv</td>
<td>Inv</td>
</tr>
<tr>
<td>P3: Wr X</td>
<td>Wr Miss</td>
<td>Wr X</td>
<td>P2 responds</td>
<td>Inv</td>
<td>Inv</td>
<td>M</td>
<td>Inv</td>
</tr>
<tr>
<td>P3: Rd X</td>
<td>Rd Hit</td>
<td>-</td>
<td>-</td>
<td>Inv</td>
<td>Inv</td>
<td>M</td>
<td>Inv</td>
</tr>
<tr>
<td>P4: Rd X</td>
<td>Rd Miss</td>
<td>Rd X</td>
<td>P3 responds. Mem wrtbk</td>
<td>Inv</td>
<td>Inv</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>