Ghost Loads: What Is the Cost of Invisible Speculation?

Christos Sakalis
Mehdi Alipour
Alberto Ros (@ University of Murcia)
Stefanos Kaxiras
Alexandra Jimborean
Magnus Själander (@ NTNU Norway)
Speculative Out-of-Order Execution

➤ Try to execute any available instruction.

➤ Hide any “visible” side-effects until everything is fine.

➤ If something goes wrong, squash.

➤ Squashing will not undo any “invisible side-effects”, such as changes to the cache.
Speculative Out-of-Order Execution

➤ Try to execute any available instruction.
➤ Hide any “visible” side-effects until everything is fine.
➤ If something goes wrong, squash.
➤ Squashing will not undo any “invisible side-effects”, such as changes to the cache.
Spectre & Meltdown

➤ Spectre “guides” speculative execution by training the branch predictor.
➤ Meltdown uses speculative execution to leak memory addresses:
  ○ Speculative instructions bring cache lines into the cache.
  ○ Timing attacks can determine in which set cache lines are installed.
  ○ Address can be inferred based on the set.
➤ The addresses can be used to infer data:
  ○ Have the address determined based on the data.
➤ Lot’s of other attacks have been surfacing since…
Our Idea

➤ Speculative execution leaks information because it updates parts of the system in ways that can be measured:
  ○ Installs and evicts cache lines.
  ○ Updates the TLB.
  ○ Triggers the Prefetcher.
  ○ Changes the DRAM state.
  ○ Coherence.
  ○ …

➤ Our idea: Don’t do these things until the instruction is no longer speculative.
➤ We focus on the caches, specifically load accesses. **Not just for Spectre & Meltdown.**
No Speculation (Delay)

- Delay loads until they are no longer speculative.
- Essentially, **disable speculation** for loads.
- Baseline is a regular OoO processor.
- -40% performance, +30% energy

GMean is for all benchmarks, not just the ones displayed here.
Invisible Speculation: Ghosts

➤ Uncacheable Loads.
➤ Do not update the LRU, TLB, etc.
➤ Do not participate in coherence.
➤ Are only allowed to update fully associative or randomised structures.
➤ Prefetches triggered by Ghosts are also Ghosts (more in the next slides).
➤ Performance is even worse than delay.
➤ 18x DRAM reads (over baseline).
Ghost Buffer (GhB)

- Ghost Buffer: A small cache only for Ghosts.
- 8x64b = 512 bytes for the L1.
  - Bigger for L2, L3, etc.
- Read-only.
- Fully associative, or otherwise randomized.
- One per cache, attached.
- Stores Ghost prefetches.
- Slightly better than delay.
Materialization (Mtz)

➤ At commit, “replay” the load.
➤ Update the LRU.
➤ If possible, use the Ghost Buffer to install data into the cache.
➤ Etc…
➤ Quite often, by the time the Mtz packet reaches the cache, the data is already there.
Final Solution: Ghosts + GhB + Mtz

- Regular Mtz
  - Installs data from the GhB, otherwise goes to memory.

- No-Request Mtz
  - Only installs data from the GhB, never goes to memory.

- Final results: -12% performance loss, 8% energy increase.

- Main performance suspect: MLP
Full Results: Performance
Full Results: Performance

![Graph showing performance comparison between different models]

- Delay better than Ghosts

Baseline
Summary

➤ Speculative execution leaks information by changing the state.
➤ We can prevent that by using Ghosts + a Ghost Buffer + Materialization.
➤ Cost of security: only -12% IPC, +8% energy.
Summary

➤ Speculative execution leaks information by changing the state.
➤ We can prevent that by using Ghosts + a Ghost Buffer + Materialization.
➤ Cost of security: only -12% IPC, +8% energy.

Next Steps

➤ Do we need to secure all loads?
➤ How can we further improve performance?
➤ Predictor for Delay vs. Ghosts.
➤ Predictor for Materialization.
Summary

➤ Speculative execution leaks information by changing the state.
➤ We can prevent that by using Ghosts + a Ghost Buffer + Materialization.
➤ Cost of security: only -12% IPC, +8% energy.

Next Steps

➤ Do we need to secure all loads?
➤ How can we further improve performance?
➤ Predictor for Delay vs. Ghosts.
➤ Predictor for Materialization.
Ratio of Loads Executed Speculatively

[Bar chart showing the ratio of loads executed speculatively across different benchmarks and configurations. The y-axis represents the ratio of ghost loads, ranging from 0.0 to 1.0, and the x-axis lists various benchmarks and configurations. The chart includes various bars in different colors representing different configurations, such as baseline, delay, ghost, ghost+ghb, ghost+mtz, ghost+mtz+ghb, and ghost+mtz+ghb+noreq.]
L1 Miss Ratio

![L1 Miss Ratio graph showing various application benchmarks with different configurations]
L1 MSHR Hits & Misses

![L1 MSHR Hits & Misses Chart]

- Normal Misses
- Normal Hits
- Ghosts Misses
- Ghost Hit