WCET analysis of multi-level set-associative data caches

Benjamin LESAGE, Damien HARDY & Isabelle PUAUT

University of Rennes1

30th June 2009
1. Issues

2. Method

3. Results

4. Conclusion & future works
Issues

• Instruction caches:
  • Static analysis [Ferdinand & al. - 2000]
  • Extension to many replacement policies [Heckmann - 2003]
  • Multi-level analysis [Hardy & Puaut - 2008]

• Data caches:
  • Single level LRU data cache analysis
Issues

- Path indeterminism
  - Gathering information for all possible incoming paths
- Access indeterminism
  - Keep coherent information when precise access target remains unknown
- Data cache hierarchy
  - Estimate accessed cache levels upon an access
Method overview

Data address analysis: attach address intervals to each memory reference

Multi-level data Cache Analysis: focus of this speech

Worst Case Analysis: see the paper for details
Method overview - Multi-level data Cache analysis

- Statically analyse data caches one by one (from top to bottom)
- Compute accesses occurrences on cache level $L$ according to:
  - Occurrences on cache level $L - 1$
  - Access classifications on cache level $L - 1$
Assumptions

- **Load policy**
  - Search in cache level $L$ iff, a miss occurred on $L - 1$
  - Miss on $L \implies$ entire missing cache line inserted in $L$
- **Write policy**
  - **Write-through** : updating the all memory hierarchy
  - **No-write-allocate** : no insertion upon miss on store
- **No other actions on cache contents**

\( \implies \) Set-associative data caches not enforcing inclusion
Cache analysis - Path indeterminism

- Safe estimation of cache contents $\Rightarrow$ all paths considered altogether
- Based on abstract interpretation and fixpoint computation [Ferdinand & al. - 2000], 3 analyses:
  - **Must**: memory blocks always present in the cache
  - **May**: memory blocks that may be present in the cache
  - **Persistence**: memory blocks which once loaded will not be evicted from the cache
- **Cache Hit/Miss Classification based on abstract cache contents**
  $\rightarrow$ compute WCET contribution of references w.r.t. caches
Cache analysis - Path indeterminism

Example (Must analysis)

- Considered LRU data cache:

- Access to memory block a:
Cache analysis - Path indeterminism

Example (Must analysis)

- Considered LRU data cache:

- Access to memory block a: Update the cache state upon an access
Cache analysis - Path indeterminism

Example (Must analysis)

- Considered LRU data cache:

- Access to memory block a:

  Update the cache state upon an access
Cache analysis - Path indeterminism

Example (Must analysis)

- Considered LRU data cache: 

- Access to memory block a: Update the cache state upon an access
Cache analysis - Path indeterminism

Example (Must analysis)

- Considered LRU data cache: 
- Access to memory block a: 
  Update the cache state upon an access
Cache analysis - Path indeterminism

Example (Must analysis)

- Considered LRU data cache:

- Access to memory block a: Update the cache state upon an access
Cache analysis - Path indeterminism

Example (Must analysis)

- Considered LRU data cache:

- Access to memory block a: Update the cache state upon an access
  Join incoming cache states on branch convergence
Considered LRU data cache: \[ \text{age}^+ \]

Access to memory block a: \( [a] \)

Update the cache state upon an access

Join incoming cache states on branch convergence

Join\text{Must}: intersection + maximal age
Cache analysis - Access indeterminism

Problem: Exact access address statically unknown (only an upper bound).
⇒ Modify Must, May and Persistence analyses Update functions

Must analysis

- Considered LRU data cache:
- Access to memory block a:
- Address range attached by the data address analysis
Cache analysis - Access indeterminism

Problem: Exact access address statically unknown (only an upper bound).

⇒ Modify Must, May and Persistence analyses Update functions

Must analysis

- Each t[i] is accessible
Cache analysis - Access indeterminism

Problem: Exact access address statically unknown (only an upper bound).

⇒ Modify Must, May and Persistence analyses Update functions

Must analysis

- Each t[i] is accessible
- Only one is accessed in fact
Cache analysis - Access indeterminism

Problem: Exact access address statically unknown (only an upper bound).
⇒ Modify Must, May and Persistence analyses Update functions

**Must analysis**

- Each t[i] is accessible
- Only one is accessed in fact
⇒ We combine Update and Join functions
Cache analysis - Hierarchy

Problem: Which access occurs and should be considered in analysis?

Pitfall: systematic access not the worst case (see [HP - 2008])
Cache analysis - Hierarchy

Problem: Which access occurs and should be considered in analysis?

⇒ Cache Access Classifications:

- **Always (A)**: reference $r$ always accesses cache level $L$
- **Never (N)**: reference $r$ never accesses cache level $L$
- **Uncertain-Never (U-N)**: no guarantee on whether or not $r$ accesses cache level $L$ considering the first occurrences of $r$, then $r$ never accesses $L$
- **Uncertain (U)**: no guarantee on whether or not $r$ accesses cache level $L$
Cache analysis - Hierarchy

A : Analysed, as before
N : Not analysed, no need to update the cache
U/U-N : Union of both the cases (cache is updated or not)

Works because of assumptions
Experimental setup & Benchmarks

- 2-levels cache hierarchy:
  - **L1**: 4-way 1KB data cache, 32B line size, 1 cycle access latency
  - **L2**: 8-way 4KB data cache, 32B line size, 10 cycles access latency

  **Memory**: 100 cycles access latency
  ▶ 150 cycle store latency
  ▶ perfect instruction cache

\[ \text{WCET}_{L1} \] : only L1 considered, all L2 accesses are misses
\[ \text{WCET}_{L1 \& L2} \] : both L1 and L2 analysed
Metric : ratio \( \frac{\text{WCET}_{L1 \& L2}}{\text{WCET}_{L1}} \)
Results

Tasks working set sizes (bytes):
- fft : 528
- qurt : 284
- statemate : 410

Very interesting with:
- few indeterministic accesses (just the hierarchy)
- or working set size fitting in the L2-cache
Less interesting with:

- indeterministic accesses to large ranges
- or unused L2-cache

Tasks working set sizes (bytes):

- insertsort : 60
- matmult : 4900
- minver : 690
- ns : 5048
Directions for less pessimistic estimates

Direct extensions:
- Better address analysis
- Locking or use of scratchpad memories
- Alter assumptions
- Other cache configurations

Other ongoing work:
- Considering Multi-core architectures