Motivation

- Timing analysis (WCET estimation) required for
  - Scheduling of real-time tasks
  - Reducing power consumption with DVS
- Data Caches
  - Introduce unpredictability
  - Add to pessimism of WCET estimates
  - Are difficult to characterize
- Getting rid of data caches causes performance loss…
- Solution: Try to predict data cache behavior

Cache Miss Equations (CMEs)

- One popular approach to characterize data caches
- Set of linear Diophantine equations, relates
  1. iteration space,
  2. cache parameters and
  3. memory references
- Works on loop nest oriented code
- Solutions to equations give potential miss points

CME implementation framework

- Framework used – Coyote by Vera et al.
- Output – est. # misses for each reference in loop nest
- Results slightly pessimistic
- Works only for perfect loop nests

Methodology

- Loop nests transformed to single loop nest by fusion
  - Conditionals based on loop induction variables introduced
- Exact data cache reference patterns generated
- Analyze all iteration points
- Re-analyze “compulsory misses”
  - to eliminate pessimistic misses
- Verify correctness of “hits”

A simple example

- Cache configuration
  - 32 byte cache line, 1KB cache, direct mapped
- Input
  - A[1..5][1..5] – base address = 151944
  - B[1..5][1..5] – base address = 153068
- Loop nest:
  - for(r = 0; r < 5; r++)
    - for(c = 0; c < 5; c++)
- Original Coyote framework output
  - Misses for Ref1 = 8, Ref2 = 9 and Ref3 = 0
- Our output
  - Ref 1: MhMhMhMhMhMhMhMhMhMhMhMh – total misses = 6
  - Ref 2: MhMhMhMhMhMhMhMhMhMhMh – total misses = 6
  - Ref 3: hhhhhhhhhhhhhhhhhhhhhhhhh – total misses = 0

Conclusions

- Exact miss counts obtained
- WCET estimates can be made significantly tighter
- Helps embedded systems
- low-end processors to very high-end processors!
- Future work
  - Relax assumptions of CME framework

More about CMEs

- Types of CMEs
  - Cold miss equations
    - Capture misses on first access to memory line
  - Replacement miss equations
    - Capture interference between two references
  - Solving CMEs
    - Direct solutions not practical due to complexity
    - Complexity reduced in implementations

Contributions of our approach

- We produce a data cache analyzer which:
  - Transforms arbitrary loop nests to single loop nest
  - Same reuse & iteration representations
  - Same CME implementation reused
  - Produces exact # of misses for every reference
  - Gives actual hit/miss pattern
- Applicability
  - Static timing analyzer → tighten WCET bounds