Continuous Constant-Memory Monitoring of Embedded Software Timing

Johan Kraft and Thomas Nolte
Mälardalen University
Västerås, Sweden
{johan.kraft, thomas.nolte}@mdh.se
ABB Industrial robot controller

- Complex
  - 3 million lines of code
  - About 50 tasks
  - Highly event triggered
- Failures very expensive
- Real-time and Performance
- Timing analysis?
  - Trial-and-error
Timing Analysis by Simulation

- RTOS-level simulation on PC
  - Application code + CPU usage annotations
  - Run many simulations with random variations
- Applicable to complex systems
  - No design assumptions
- Finds problems and extreme cases
  - But no guarantees – like testing
- Our simulator: RTSSim
Challenge

- Modeling an existing complex embedded software system for simulation-based timing analysis?
  - Manual modeling not realistic
Simulation Model Extraction

- Extract functional model from source code
  - Earlier work, using program slicing
- Generate timing profile from measurements
  - Execution times
  - Inputs
  - Response times – for validation
Paper Contribution

- Continuous Online Timing Profile Generation
  - Allows for very long monitoring sessions
  - Uses a constant, moderate amount of RAM
  - No extra hardware – monitor deployed systems
Timing Profile

- Sample distribution: complex, multimodal
  - Does not fit theoretical distributions
- Represent as N intervals
  - Min: Lowest sample value of interval
  - Max: Highest sample value of interval
  - Count: Number of samples in interval
- Usage during simulation
  - Select interval by probability Count / TotalSamples
  - Sample from uniform distribution [Min, Max]
Timing Model Generation

\[ X.\text{count} = X.\text{count} + 1 \]

\[ x = \text{next input} \]

Any existing interval \( X \) enclosing \( x \)?

Yes
Interval Merge Heuristics

- Calculate merge fitness for neighbour intervals
  - Proximity
    - Closer intervals are more suitable for merge
  - Density
    - The more similar Count/(Max-Min), the better fitness
    - To avoid merging a "spike" with a "plateu"
  - Count
    - If few samples, disregard "Density" in fitness value
Characteristics

- Processes one sample at a time
  - Process directly – no sample buffer needed
  - Or, use a small buffer and process on idle time
- RAM needed per property: 3\text{wn}
  - \text{w}: Width of interval properties (e.g., 2 or 4 bytes)
  - \text{n}: Number of intervals allowed
- At \text{w} = 4, \text{n} = 10:
  - 120 bytes per property
  - Allow for 4,294,967,295 (2^{32} -1) samples per interval
Prototype Evaluation Setup

- **RTXC on ARM7** → **Trace** → **Tracealyzer** → **Timing Profile Generation**
- **RTSSim Simulator** → **Trace** → **Tracealyzer** → **Timing Profile Generation**
“Timing Profile Viewer”
COMODRV, Execution Time, Intervals: 8/8, Data points: 839/839
Future Work

- Implement for online use on embedded HW
  - FreeRTOS on Atmel AT91SAM7 (ARM7)

- Design, implement and evaluate other interval merging heuristics
Thank you for your time

Questions or comments?