Embedded systems engineering
Distributed real-time systems

David Kendall

Time is Central to Real-time and Embedded Systems

Several timing analysis problems:

- Worst-case execution time (WCET) estimation
- Estimating distribution of execution times
- Threshold property: can you produce a test case that causes a program to violate its deadline?
- Software-in-the-loop simulation: predict execution time of particular program path

ALL involve predicting an execution time property!

WCET and BCET

- Remember the simple formula for response time analysis (no blocking or jitter).

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- Each C_i represents the *worst-case computation time* of its task.
- Worst-case computation time (WCET) : the longest time taken by a program code to complete its execution (assuming no blocking, jitter or interference)
- Best-case computation time (BCET) : the shortest time taken by a program code to complete its execution (assuming no blocking, jitter or interference)
- How to obtain values for C_i?

Calculating execution times

- Measurement
  - Need to exercise great care in obtaining measurements
  - Need to take care in interpreting results
  - How to know if you've measured the worst (best) case?
- Analysis
  - Intended to guarantee that the worst (best) case execution time is reported
  - Difficult to take account of all architectural effects: pipelines, caches, speculative execution etc.
- Let's consider a diagram that illustrates the relationship between these approaches

Timing analysis of systems

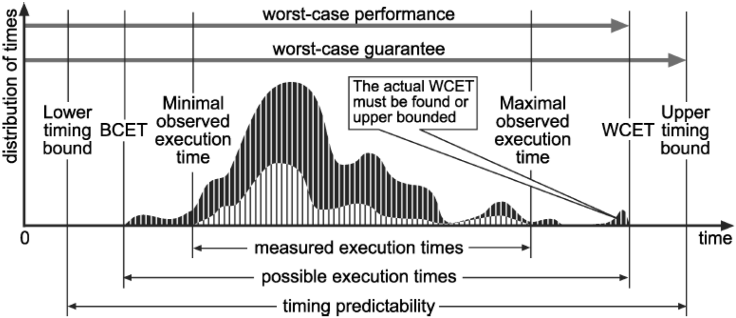


Fig. 1. Basic notions concerning timing analysis of systems. The lower curve represents a subset of measured executions. Its minimum and maximum are the *minimal* and *maximal observed execution times*, respectively. The darker curve, an envelope of the former, represents the times of all executions. Its minimum and maximum are the *best-* and *worst-case execution times*, respectively, abbreviated BCET and WCET.

Figure from R.Wilhelm et al., *ACM Transactions on Embedded Computing Systems*, Vol. 7:3, pp 36:1 – 36:53, April 2008.

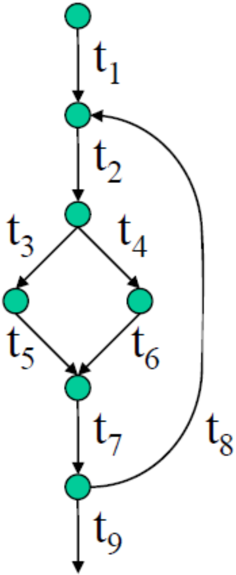
Worst-case execution time analysis

- If measurement is not guaranteed to reveal the worst-case execution time of software, is there a different approach that is guaranteed?
- Worst-case execution time analysis (WCET).
  - Static analysis of the program code, ie don't run the program but analyse its text to discover its possible behaviours.
- WCET analysis computes upper bounds for the execution time:
  - of a given piece of code
  - running on a given machine
  - starting in a given state (considers low-level hardware details: state of the pipeline, caches, registers, etc.)

WCET requirements

- Calculation of all feasible paths through the program code
- Calculation of the execution time of each feasible path when executed on a particular hardware platform

Calculating feasible paths



- Construct the control flow graph (CFG) of the program
  - CFG: nodes are basic blocks, edges show program flow between basic blocks
  - Basic block: sequence of instructions with a single point of entry at the beginning and a single point of exit at the end
- Identify the feasible paths through the CFG
- Calculate the execution times of each basic block and its transfer of control to the next basic block (t_i).
- Find the path that gives that maximum sum of execution times

Components of execution time analysis

- Program path (control flow) analysis
  - Want to find longest path through the program
  - Identify feasible paths
  - Find loop bounds (may require user annotations)
  - Identify dependencies between different code fragments
- Processor behaviour analysis
  - For small code fragments, generate bounds on run-times on the given hardware
  - Model details of architecture, including cache behaviour, pipeline stalls, branch prediction, etc.
- Outputs of both analyses feed into each other

Common current approach

- Manually construct processor behaviour model
- Use model to find “worst-case” starting processor states for each basic block then calculate execution times of the blocks from these states
- Use these times as upper bounds on the time of each basic block
- Formulate an integer linear program to find the maximum sum of these bounds along any program path

Example (from Y.T. Li and S. Malik)

xi → # times Bi is executed

dj → # times edge is executed

C_i → measured upper bound on time taken by Bi

Want to

maximize $\sum_i C_i \, xi$

subject to constraints

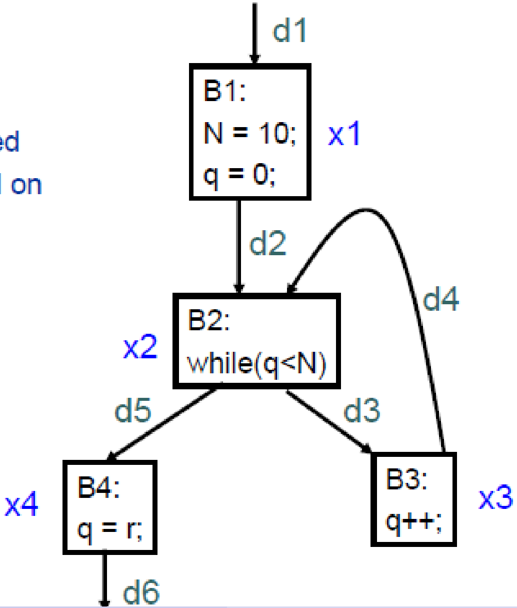
x1 = d1 = d2

d1 = 1

x2 = d2+d4 = d3+d5

x3 = d3 = d4 = 10

x4 = d5 = d6



WCET analysis is difficult

- Complex for modern processors with pipelines and caches.
- Difficult to get precise timing model of processor (simplifications and errors in data sheets)
- High implementation effort to port tool to new target
- Subject of current research
    - Possible interesting way forward: combine analysis and measurement
    - “The best model of the processor is the processor itself”
    - Analysis applied to develop a systematic search for input data that yield the worst case
    - Execute the program with the identified worst-case data and measure its execution time on the processor (or a cycle-accurate simulator)

Current WCET estimation tools

- Commercial
  - aiT from AbsInt (used to analyse the flight control software of the Airbus A380)
  - boundT from Tidorum Ltd.
  - RapiTime from Rapita Systems Ltd.
- Academic
  - Gametime from University of California at Berkeley
  - Chronos from the National University of Singapore

Acknowledgements

- Raimund Kirner, Y.T. Li, S. Malik, Edward Lee, Sanjit Seshia, R. Wilhelm, *The Determination of Worst-Case Execution Times: Overview of Methods and Survey of Tools*, ACM Transactions on Embedded Computing Systems (TECS) 7:3, 2008