Extending the Path Analysis Technique to Obtain a Soft WCET

Paul Keim, Amanda Noyes,
Drew Ferguson, Josh Neal,
Chris Healy

Department of Computer Science, Furman University

Overview

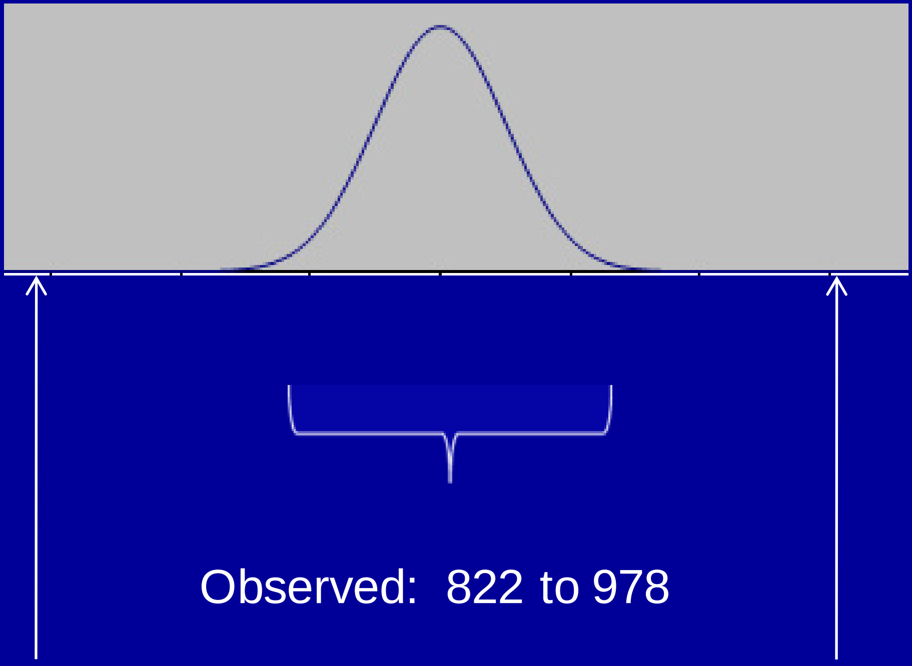
- Hard versus soft WCET
- Incorporating soft WCET in loop analysis algorithm
- Study of benchmarks
- Comparison of static timing analysis
- Ongoing & future work
- Conclusions

Motivation

- Need to bound WCET of tasks.
  - Static timing analysis tools
- Can we tolerate an occasional missed deadline?
  - Hard RT system: NO. WCET is absolute
  - Soft RT system: YES. A WCET estimate that almost always bounds the actual WCET is acceptable.
- Traditional WCET analysis has been for hard RT
  - Estimates can be quite loose due to input data

For example …

- Consider a simple distribution of possible execution times, compared to “hard” WCET



Observed: 822 to 978

BCET = 600

WCET = 1200

Goal

- We want to provide a tighter WCET bound, which may underestimate the actual execution time in rare cases (< 1%).

- Extend earlier work in timing analysis
  - Statically determine the distribution of execution times.

- Need to also update hardware simulator so it also produces time distributions: interesting case studies.

Loop analysis

- In our “path analysis” approach, we do the analysis bottom up. Instruction → path → loop → function → task.
  - The loop’s execution time distribution depends on its paths.
  - Multiple paths result from conditional control flow.
  - Choice of path typically depends on input data, unknown at compile time.
  - 1 path : trivial
  - 2 paths : we use binomial probability technique
  - 3+ paths : repeated application of (2)

2 path case

Let A = longer path and B = shorter path

compute A_{time} and B_{time} as well as A_{prob} and B_{prob}

```
total_prob = 0.0
for i = 0 to n
    p = probability that A is taken on i of the n iterations and
            B is taken on (n – i) iterations
    for j = 100*total_prob   to  100* (total_prob + p)
            time_dist[ j ] = $A_{time} * i + B_{time} * (n – i)$
    total_prob += p
```

More paths?

- Consider case of 4 paths (A,B,C,D) with equal probability of being taken.
  - Use 2 case model to find time distributions TD_{A+B} and TD_{C+D}.
  - Concatenate the two TD's. Sort the values, and remove every other element (because they were the same size) to normalize size of the TD.

- What if $(A+B)_{prob}$ > $(C+D)_{prob}$?
  - Before you concatenate, stretch TD_{A+B} by a factor of $(A+B)_{prob}$ / $(C+D)_{prob}$

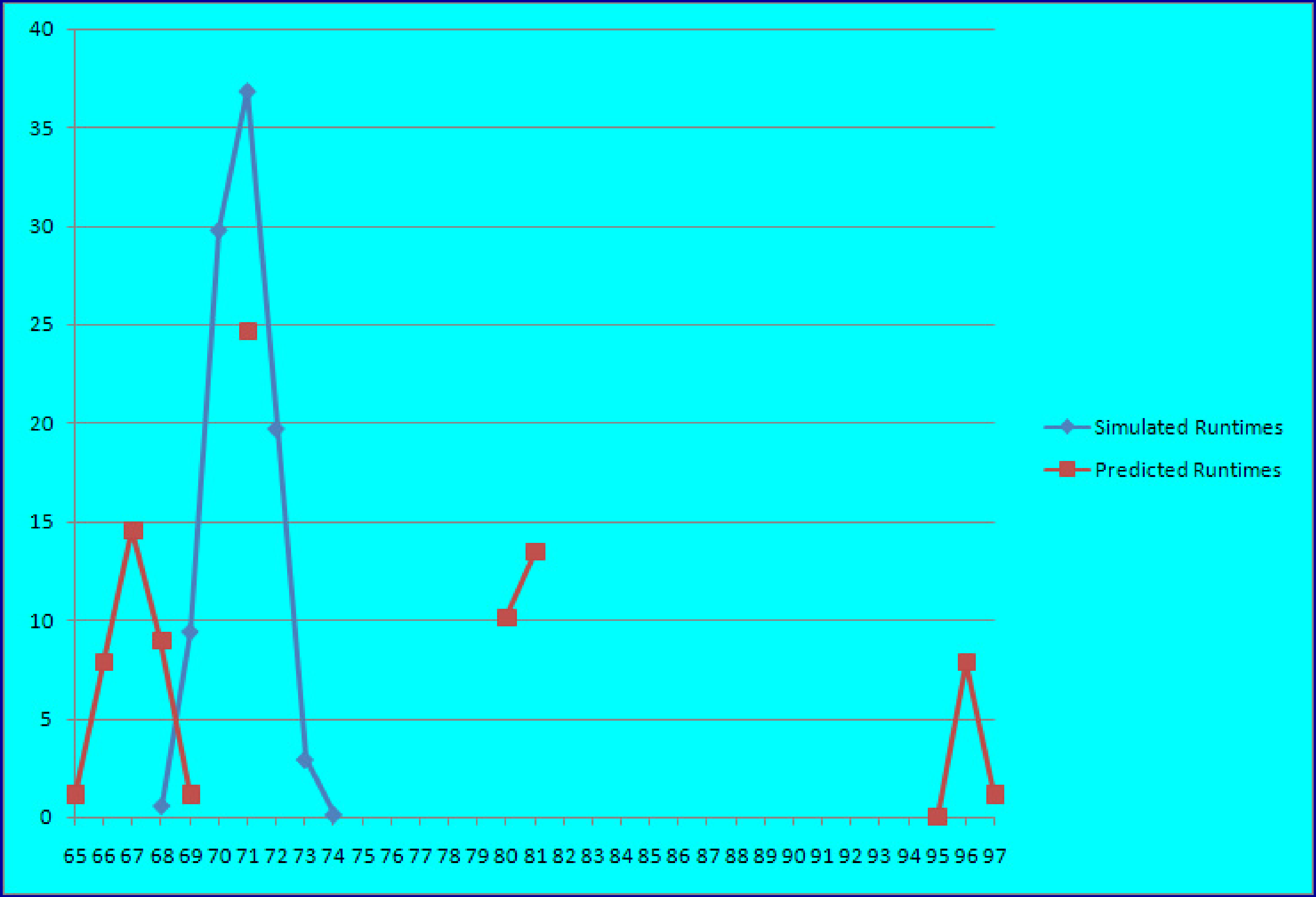
Methodology

- Benchmark tasks with random input data
- Timing analyzer only needs to be run once, because it ignores the input data.
- Using hardware simulator
  - 1000 trials of benchmark program
  - Automatically generate next version of benchmark, compile and run.
  - Testing is easier for soft WCET than for hard WCET because we don’t need to figure out the WC input data. ☺

Simulated observations

- Each time we ran a randomized version of a benchmark, we computed: observed execution time / predicted WCET. (As a percent)
- We took note of:
  - Width of the observed distribution
  - Skewness
  - The observed soft WCET. In one case this was as low as 2%.
  - Different loops in the same function
  - Effect of changing the distribution of input values. ("fair" versus "unfair")

Result with 7 paths



Ongoing & Future

- Combining multiple loops
- More realistic estimate of probability of taking a branch
- Open questions:
  - Consider other probability distributions?
  - How to measure “how close” static and dynamic distributions are?

- GUI and Website `cs.furman.edu/~sparta`
- Combine with work on parametric timing analysis to produce distribution of polynomial coefficients.

Conclusion

- Extend our existing framework for bounding hard WCET to generate:
  - static prediction for soft WCET,
  - as well as entire execution time distribution.
- Incorporated into existing timing analyzer, giving rapid result for all loops in program.
- Scales well with ↑ number of paths.
- Potential benefit for system developers if hard and soft WCET differ substantially.