A Profile-Driven Statistical Analysis Framework for the Design Optimization of Soft Real-Time Applications

Tushar Kumar[1], Jaswanth Sreeram[2], Romain Cledat[2], Santosh Pande[2]
[1]School of Electrical and Computer Engineering, [2]College of Computing, Georgia Institute of Technology, Atlanta, GA, USA
tushark@ece.gatech.edu, jaswanth@cc.gatech.edu, romain@cc.gatech.edu, santosh@cc.gatech.edu

Motivation

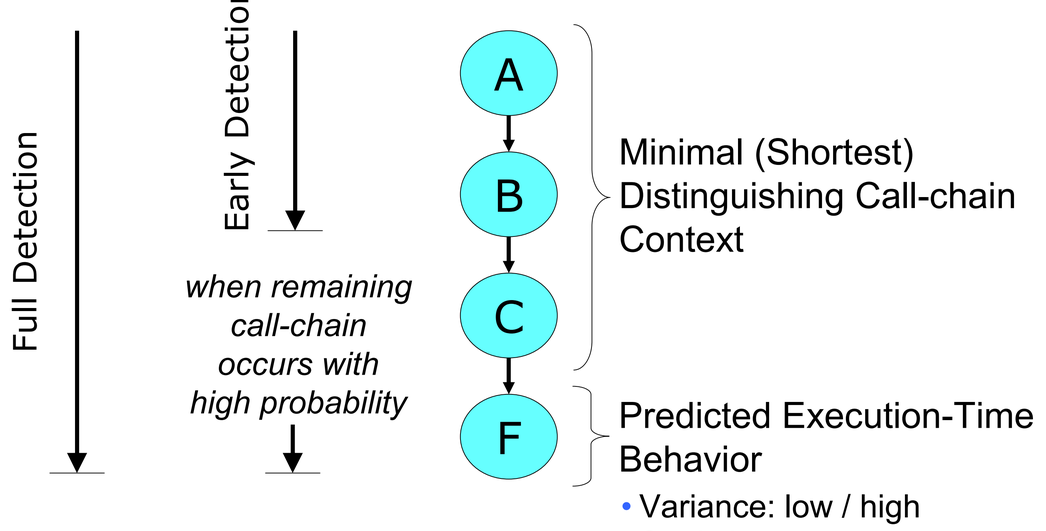
- Lack of a formal methodology for the design and optimization of Soft Real-time applications
  - Monolithic C/C++ applications: Games, multimedia
- Difficulty in capturing Soft Real-time requirements using formalisms from Hard Real-time (*Deadlines*, *Tasks*)

  "Achieve 30 frames-per-second on average, and stay within 5 fps of average frame-rate with > 95% probability"
- Soft Real-time applications need to meet statistical execution-time requirements under common or representative data and usage scenarios

Analysis and Optimization Methodology

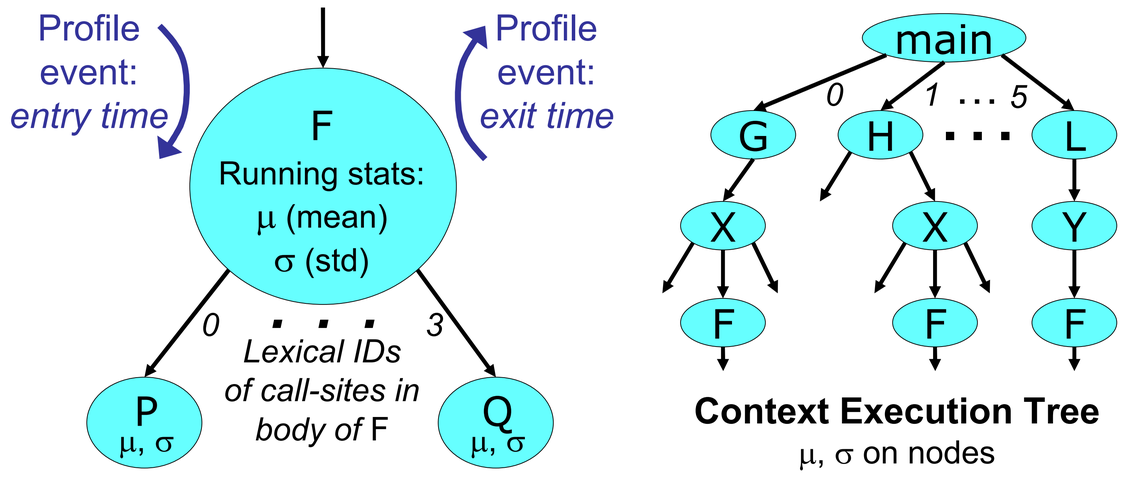
- Automatically *characterize the variability* in the execution of an application, and *construct patterns that predict* the variable behavior in a *statistically reliable manner*
  - Applications driven with *representative data sets* (profiling)
  - Determine selective occurrence of variation behavior based on *calling-context*
  - Summarize the most significant *application-wide variation behavior* using few patterns
- Patterns direct programmer's attention to the functions and corresponding calling-contexts that *most significantly* affect and predict the application's Soft Real-time characteristics
  - **High-variance**: execution of function F in given call-context *contributes significant variability* to the execution time of enclosing scopes
  - **Low-variance**: F under given call-context *significantly dampens variations* in enclosing scopes
  - Many other interesting behaviors (such as, correlated execution times) can be similarly detected as patterns
- Programmers can even detect call-contexts of patterns at runtime, and explicitly adjust program activity based on statistics associated with corresponding patterns

Pattern



Pattern Generation Steps

CET Construction



Total dynamic instruction count used for time-stamps

Context Execution Tree
μ, σ on nodes

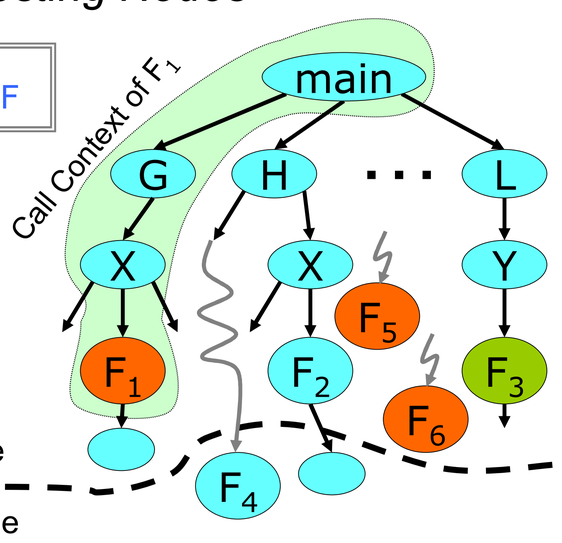
Time complexity = $\theta(N_{Prof})$
N_{Prof} = # of profile events

Tagging Interesting Nodes

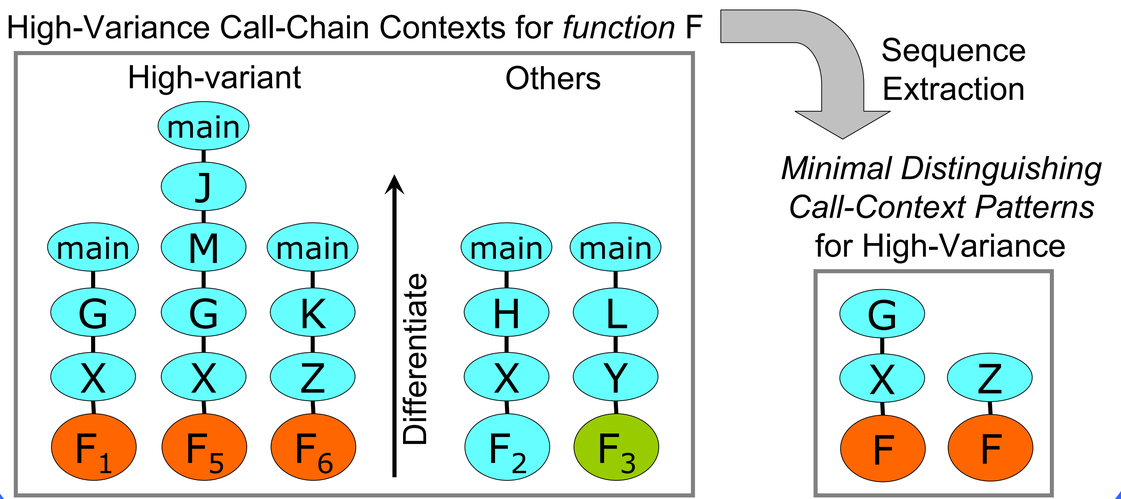
For any node F: $cov_F = \sigma_F / \mu_F$

cov_F > high_threshold
➔ Tag F as **high-variant**

cov_F < low_threshold
➔ Tag F as **low-variant**



Generating Patterns



Results

Benchmark	Profile Steps (*Pattern Extraction*)	Regression Steps (*Testing Patterns*)	Low-Variance Patterns			High-Variance Patterns		
				Regression			*Regression*	
			# Extracted	Remain Low-Var	μ not affected	# Extracted	σ not affected	μ not affected
mpeg2enc	10 Million	30 Million	33	33	31	24	12	21
mpeg2dec	10 Million	30 Million	17	13	14	31	30	31
h263dec	5 Million	25 Million	32	26	29	28	26	26

Key Contribution

Combination of Profiling, Program Representation, Statistical Metrics, and Distinguishing Sequence Extraction techniques allow for a *fast, automatic determination of the program contexts* that most significantly affect and predict an application's Soft Real-time behavior.