Dynamic Analysis from the Bottom Up

Markus Mock

University of Pittsburgh

Dynamic Analysis

- Dynamic
  - Continuous & productive [Encyclopedia Britannica]
- Analysis
  - Separation of a whole into its parts

Making DA Productive

- Observation of program properties at run time
- Observation must be efficient to be productive
  - Need efficient profiling techniques

Profiling

- Instrumentation
  - Precise and detailed information
  - Significant program slowdown
- Sampling
  - Approximate information
  - Smaller impact on performance

Overhead Reduction

- Transition from quantity to quality
- Minimally-invasive observation enables
  - observation of new properties
  - Finer-grained observation
  - Pervasive deployment of Dynamic Analysis

Example: Debugging

- Data watchpoints
  - Expensive in software
    - Based on traps for every (memory access) instruction
    - Slowdown ~100X
  - Simple hardware support makes them feasible for whole programs
    - watchpoint registers
    - Special & simple processor hardware monitors memory operations and traps to software only when accesses occur
    - Available, e.g., on Pentium

Vision: Bottom-Up Dynamic Analysis

- Design DA infrastructure from the hardware-level to the application-level
- Exploit hardware features for fast data collection
- Design compositional primitives that support multiple dynamic analyses
- Explore what hardware features are required

Analysis Counters

- Modern hardware has performance counters
  - Simple form for monitoring system behavior, examples:
  - Typically oriented towards / used for performance profiling / improvement
  - Examples
    - Cache hits & misses
    - IPC
- Analysis counters
  - Monitor software properties
  - Oriented towards analysis, i.e., understanding of program parts and whole
  - Examples
    - Call graph counter ⇒ dynamic call graph
    - Alias counter ⇒ dynamic points-to analysis

Call Graph Counters

- Processor records for every call instruction
  - Current procedure address (CA)
  - Target address (CE)
  - Call instruction address (CS)
  - In a fixed size rotating hardware history list for <CA,CE, CS> tuples
  - Periodic transfer of the hardware structure contents to program memory (monitoring process or application code), a la DCPI

Why this will fail

- Hardware development is performance-driven
  - Little interest in software engineer's concerns
- Useful dynamic properties are complex
  - Too expensive to realize in silicon
  - Too narrowly applicable (just one client per feature)
- Hardware-realization too inflexible
  - Many different rapidly involving dynamic analyses that all require different support mechanisms

Why this will work

- Hardware development is performance-driven
  - Some dynamic analyses help improve performance
  - Enough transistors available for off critical path structures (counters, watch registers etc)
- Useful dynamic properties are complex - but
  - Can be synthesized from simple primitives
  - Primitives can be shared across multiple analyses
  - Primitives can be combined in different ways ⇒ flexibility