Ideas for Efficient Hardware-Assisted Data Breakpoints

Jonathan Cook   Mayur Palankar
New Mexico State University
WODA 2004
Watching Variables

• Typical instrumentation is control-oriented, but watching data is often useful and interesting
• E.g., Tcl “trace variable” command
• Historically, debuggers have been very poor at watching data
  – general locality problem is hard
  – so just break after “every” instruction
  – e.g., 10,400 slowdown on a simple program
Specialized CPU Hardware

• i386+ has four breakpoint registers (other popular CPUs have one)
• Each will watch one word of memory at CPU speeds
• Enables limited data breakpoints with no slowdown -- unless breakpoint occurs
• Used only naively so far, by debuggers
Using BP Registers for DA/RM

- Why not use these four registers for many other dynamic analysis/runtime monitoring purposes?
- To consider this, we must not be limited by the number of registers

- How to watch 100 variables with just four registers?
Overall Process

Program Source → Static Analysis

Watched Variables → Dynamic Analysis

Dynamic Analysis → Desired Output

Static Analysis → BP Reg Schedule
Scheduling BP Registers

- Given: program, set of variables to watch
- Produce: schedule of BP register usage
- Simple variables only – easy
- Arrays, pointers make everything hard
- When to change schedule?
Ideas

- Static analysis informs/creates schedule
- Hierarchy of points at which to change schedule
  - BP triggers themselves (def-def chains?)
  - function call/returns (scoping)
  - basic block entry/exit (scoping)
- Points to analysis to handle pointers
- Backpedal to high coverage but < 100%
Why this will succeed

• Data watching is useful, and has been hard to support
• It’s a shame not to use hardware support if it is available
• Points-to analyses show few offending pointers
• May enable other interesting ideas
  – data-based joinpoints for AOP?
  – security-oriented monitoring
Why this will fail

• Context switching!
  – BP registers trigger kernel-level trap
  – programming support only allows parent process to catch the trap

• Previous work (Wahbe et al., 1993 PLDI) set a high bar using direct instrumentation

• Maybe not as high a need for data watching as we think?