Opportunities for Global Optimization: breaking the boundaries across system components

Ramon Bertran  Marisa Gil  Javier Cabezas  Victor Jiménez  Luis Vilanova  Enric Moranchon  Nacho Navarro

Module Optimization View

- Each component is compiled separately
  - Intra-module optimizations
  - Several traditional techniques: Constant propagation, dead & unused code elimination, ...
- Components will interact at run-time
  - Must follow ABIs conventions
  - Calling conventions expose properties but optimizations still conservative at component boundaries
- Convenient for General Purpose Systems
  - Need to support several hardware and software components

Global System Optimization View

- All components are analyzed together
  - Global view enables the propagation of code and data properties
  - Optimizations can be applied across components
  - Possibility to specialize modules to work together
- For Specialized Systems
  - Reduced and fixed set of components
  - New inter-module opportunities arise, for example:
    - Constant propagation across modules
    - Optimize-Specialize calls to library functions and system calls for specific parameters
    - Branch optimization, code reordering
    - Unused code becomes dead code on the Global View
    - Delete unused library functions, system calls and handlers

Building a Global System View

1. Build a control flow graph (CFG) for each system component
   - Applications, libraries, servers, and kernel
2. Identify disconnected edges
   - Library calls, system calls, IPC, handlers, software interrupts and exception handlers
   - Component entry/exit points
3. Merge component’s CFGs in a single WCFG
   - Using symbolic information
   - Analyzing the code
   - Feedback from an expert developer
4. Characterize the connections
   - How the data flows through the connections
   - Considering address spaces, privilege level, ...

Use of Diablo binary relinker framework to build the CFGs and perform analysis and optimizations

Opportunities on Linux

- Monolithic kernel
  - Entirely in privileged mode
    - Several system calls
      - Entry/Exit points
        - System call handlers
        - Entry/Exit points
- Shared library model
  - Several functionalities provided
    - Entry/Exit points
    - Exported function definitions and data
    - Inter-module calls

Opportunities on L4 : Pistachio

-µKernel design
  - Privileged module:
    - System call, IPC, interrupt and exception handlers
    - Minimal abstractions and system calls
  - Kernel interface and KIP:
    - L4 Library
    - KIP page mapped in user address space contains system call code
    - Entry/Exit points:
      - Functions and data exported by L4 Library
      - System calls stubs exported by the kip

- IPC / Server application model
  - Servers at user level provide
    - Functionalities to the applications
  - Communication through applications rely on IPC messages

- Optimization/Specialization opportunities
  - Few dead code elimination opportunities
  - Minimal and already optimized µKernel functionalities
  - Across layer connections
    - Application to server connections
      - Difficult to characterize the path: user -> lib -> kip -> kernel
    - Inter-component connections
      - Several unused functions in shared libraries
      - Kernel unused system calls removed using KIdiablo

Case of study: of a Linux system

- Embedded shell and web server on top of Linux 2.4
- Use of Diablo framework to specialize the system applying global dead code elimination
  - User level
    - Static configuration: few dead code elimination
    - Dynamic configuration: extensive dead code elimination
      - Several unused functions in shared libraries
      - Kernel unused system calls removed using KIdiablo

Future Work

- Evaluation of new optimization opportunities like constant propagation across system components
- Server specialization on the L4 environment
- Automatize WCFG building process
  - Continue developing tools for automatic analysis
    - Minimize the need of developer feedback

http://www.ats.ugent.be/diablo/