Specialization Tools & Techniques

Synthetix Project

Oregon Graduate Institute

Black, Consel, Pu, Walpole

Specialization in OS

- The Problem
  - General algorithms very complex
  - Most of the time the situation is simple
- Case studies in OS
  - Synthesis (Columbia, 87-92)
  - Mach IPC (Mach 4 project)
  - HP-UX Unix File System (Synthetix)

Concrete Example: UFS

- Specialize read/write, buffer cache
  - HP-UX 9.04 (dual processor)
- Simple common case:
  - Sequential read with exclusive access
  - Up to two order-of-magnitude improvement
- What about the uncommon cases?
  - Open (same file) violates exclusive access
  - Lseek violates sequential read

Making Common Cases Fast

- Principle
  - Don’t check for the common case
  - Check for the negation!
- Conditional constants
  - Predicates that are almost always true
  - Allow specialization for common conditions

Making Other Cases Correct

- Conditional constants may change
- Guards detect change
  - Repair action is to *unspecialize* by replugging with generic code
- Extension of partial evaluation
  - PE uses constants that do not change

Impact on System Modularity

- Replugging with a specialized implementation is a form of OI/MOP
  - System/module interface preserved
  - Performance is improved
- Conditional constants destroy locality
  - Correctness of a module depends on tests in other modules
  - **Research:to keep track of dependencies**

Language Abstractions

- Encapsulation enables safe replugging
- Specification of specialization condition
  - Tells PE to specialize on conditional const.
- Controlled use of aliases
  - Simplifies finding code that changes conditional constants
- Guard construct
  - Supports guard checker

Generating Executable Code

- Runtime code generation
  - PE: on-line as well as off-line
  - Binding time analysis: choose optimal time
- Incremental dynamic linking
  - Dynamically inserting new specialized modules

Need New Language & Compiler Technology

- New “middle-end” technology
  - Extend compiler technology into runtime domain
  - New language abstractions are required to automate incremental specialization
- Complement language & compiler work
  - ML, SPIN, other panelists....