Model Predictive Control for Memory Profiling

Sean Callanan, Radu Grosu, Justin Seyster, Scott A. Smolka, Scott D. Stoller, and Erez Zadok
Department of Computer Science
Stony Brook University
http://www.fsl.cs.sunysb.edu/hcos
The HCOS Project: High-Confidence Systems Software

Theme
Control Theory + Runtime Verification: high-accuracy monitoring with controllable overhead

NSF CNS-0509230, Runtime Monitoring & Model Checking for High-Confidence System Software
The HCOS Project: Tools and Techniques

- **Compiler-Based Instrumentation**: Plug-In architecture for GCC 4

- **GCC Plug-Ins for**:
  - Bounds checking
  - Monte Carlo software model checking
  - Kernel refcount usage checking
  - GIMPLE simulation and visualization

- **Memcov** memory profiler & leak detector
Memcov: Memory Profiling + Leak Detection

- **Goal:** detect *memory leaks* (or infrequently accessed areas) while regulating runtime monitoring overhead

- **Memcov** samples memory accesses to determine access frequencies and patterns, not just check for `free()`s and GC-style reachability

- **Model Predictive Control (MPC)** for *adaptive sampling*: maintain low constant overhead + high accuracy
Model Predictive Control

**Goal:** compute an optimal input by minimizing given cost function over a certain prediction horizon using model of the system
Model Predictive Control for Memory Profiling

Instrumentation requires no reccompilation; dynamic loading only

Instrumented Proc.
allocated areas
set r/w permission bits

MPC controller
Dynamic Optimizer
System Model + Cost Fcn

page interrupts

optimal sensing rates
MPC for Memory Profiling (cont.)

- **Sensor** = MMU + signal handler
- **Actuator** = an allocated area’s memory protection bits
- **Taking a sample** = setting area's protection bits
- **Controller** adjusts sampling rate to be inversely proportional to area's access rate
- Thus, infrequently accessed areas (i.e. potential leaks) considered highly *critical* and monitored more frequently
Control Loop as a “Timed Automaton”

\[ \text{x is a clock variable that is reset after every transition} \]

\[ \text{monitoring-off} \]

\[ x \leq f(\text{rate, nreg}) \]

\[ \langle \text{access} \rangle / \text{rate}=x^{-1} ; x=0 \]

\[ \text{monitoring-on} \]

\[ x \geq f(\text{rate, nreg}) / \langle \text{enable monitoring} \rangle ; x=0 \]

\[ f(\text{rate, nreg}) = \frac{p}{o_{\text{global}} / \text{nreg}} - \frac{1}{\text{rate}} - p \]
Memcov Instrumentation Architecture

Without Profiling

User code
- Target
  - malloc()
  - brk()
Libary and system code
- libc
  -INVLPG
  - Kernel
- MMU
  - Memory

With Profiling

User code
- Target
  - Sens/Act
  - Controller
  - mmap()
  - mprotect()
Libary and system code
- libc
  - brk()
  - Kernel
- MMU
  - Memory

Monitor fault
Page faults
Memcov Multi-Process Architecture

- **Commands**: Activate region
- **Events**: malloc, free, realloc, page fault
We wrote benchmark for vim that creates a file, populates it, saves it, then deletes all its data.

Heavy allocator traffic causes page faults.

Most areas are monitored.
Memory efficiency for vim

Observation: Most areas just get older!
Conclusions

- **Model Predictive Control** helps achieve accurate monitoring with low constant overhead

- **Current and Future Research**
  
  - Applying `memcov` to Firefox and Apache
  
  - MPC-based techniques for IDS *packet sampling*, detecting kernel `refcount` mis-uses, bounds checking, etc.
Questions?

- Thank you!
- For more information, please visit:
  http://www.fsl.cs.sunysb.edu/~hcos

- Bullet: Apply control theory to software sensors to control Heisenberg effect!