VI Prof: A Vertically Integrated Full-System Profiler

NGS Workshop, April 2007

Hussam Mousa
Chandra Krintz
Lamia Youseff
Rich Wolski
RACELab Research

• Dynamic software adaptation
  ■ As program behavior or resource conditions change
  ■ Dynamically change the program (via re-compilation) or the runtime services to account for and exploit these changes
  ■ To improve performance and energy efficiency

• For high-end systems
  ■ Workstations, desksides, clusters, servers, …

• Three key components of adaptive software optimization
  1. Extraction of performance metrics: Program profiling
  2. Behavior characterization and prediction
  3. Program/system modification to exploit future behavior
     ‣ Via dynamic compilation or runtime optimization
**VI VA: Vertically Integrated Virtualization**
Full system specialization & dynamic adaptation

- **Key**: Single application execution model: server systems, batched clusters
- VMMs - emerging software technology that enables isolation, improved server utilization, migration, portability

UCSB Laboratory for Research on Adaptive Compilation Environments
**VI VA: Vertically Integrated Virtualization**

Full system specialization & dynamic adaptation

- Application specific
- Resource-aware
- High performance

**Key:** Single application execution model: server systems, batched clusters

- Current system layers and boundaries available to programmer
- VI VA automatically eliminates, integrates, and customizes layers during compilation and runtime to extract new levels of high performance
**VI VA: Vertically Integrated Virtualization**

Full system specialization & dynamic adaptation

- Application specific
- Resource-aware
- High performance

- **Dynamic Software Adaptation**
  - Profiling
  - Prediction
  - Compiler and runtime optimization
Full System Profiling: VI Prof

- Vertically integrated profiler (and post-processing toolkit)
  - Based on OProfile -- (Linux kernel module that exports HPM data)
  - *Full-system* HPM sampling system

- Collects HPM stats across all functions/methods in system
  - Control sampling rate: trade off accuracy for performance
  - Single unified system
  - OS-level so no application-level perturbation

- Maps and tracks dynamically changing code regions
VI Prof Implementation

• Runtime profiler
  ■ Attributes performance data (HPM values) to code addresses
    ▸ Which are later mapped to methods/functions offline
  ■ Daemon that periodically samples the system
    ▸ Extended to enable registration of
      ♦ Dynamically generated code (due to dynamic (re-)compilation)
      ♦ Code bodies that are moved via a copying garbage collection (GC)

• VM Agent
  ■ Virtual machine module that tracks dynamic compilation and GC
    ▸ Creates code maps (method signatures to addresses)
    ▸ We handle GC as a cascade of epochs
    ▸ Portable
  ■ Asynchronously logs registration details
  ■ Highly optimized for minimal application interruption
VI Prof Post-Processing Toolkit and API

- Set of tools that categorize, sort, and display sample information in a variety of ways
  - Handle the map files from the VM agent
  - Search the cascade from most recent to earliest epoch
    - If the code body for a particular sample is not found in current epoch
      - The previous epoch is searched
      - This continues until the code body is found

- Clean API available that enable integration of any system that generates or moves code dynamically
  - VI Prof is currently integrated into
    - Mono (.Net), Hotspot, JikesRVM, and soon Microsoft Phoenix
  - Any Linux 2.6 system
Experimental Methodology

• OProfile 0.9.2
• JikesRVM 2.4.5
• Linux Kernel 2.6.20.16
• Single core Intel 3.4 MHZ Xeon with 2GB of RAM
• Benchmarks:
  ■ SpecJ VM98, Dacapo, SpecJ BB
  ■ Repeated runs, averaged
  ■ Average runtime without profiling: 33s
VI Prof Overhead

Benchmarks from SpecJ VM98, Dacapo, SpecJ BB; Averaged over 10 runs (max removed)

Sampling rates: 1/N cycles
Oprof 90K -> sample once every 90000 cycles
Related and Ongoing Work

• Related work
  - OProfile Linux profiler (http://oprofile.sourceforge.net)
  - Other HPM-based sampling systems (*non-integrated*)
    - Virtual machines [Hauswirth05]
    - Performance and event monitoring (PEM) [IBM04]
  - Instrumentation systems (complementary to VIProf)
    - JVM [Arnold01, Sastry01, Newhall99]
    - OS [Mirgorodskiy03, Tamches99]

• Currently, we are working on
  - Integrating VIProf into Xen
  - Supporting multiple OS instances concurrently
  - Performance analysis of VI Profiles
    - When is instrumentation required? Profile-guided profiling
    - Capture phase, threading, I/O, memory management behavior
RACELab VIVA-Related Projects

• Automatic deployment systems for Xen images
  - Batched clusters for scientific computing
  - Distributed systems

• XEN performance evaluation for HPC
  - File I/O, MPI communication, computationally-bound
  - Automatic installation of OS images over Xen
    ▸ Integrated with development environment

• Customization of Linux & integration with higher-level services
  - Specialization of Linux modules for application-specific behaviors
  - Virtual machines, Grid and web services
Conclusions

- Traditional static compiler techniques have difficulty extracting high-performance from programs in modern PLs given increasing complexity in hardware and software
  - **Our work:** novel dynamic compiler and runtime techniques that adapt the software stack to changes in the execution environment
- Key first step toward this goal
  - Accurate and low overhead full-system profiling: VIProf
  - Tracks hardware performance counters across all code in system
    - Kernel, library, application
    - Handles dynamism efficiently (dynamic compilation, moving GC)
  - For efficient generation of online performance data
    - That can be used to guide optimization, specialization of the application or runtime

For more info: [http://www.cs.ucsb.edu/~racelab](http://www.cs.ucsb.edu/~racelab)