Integrating Parallel Application Development with Performance Analysis in Periscope

V. Petkov, M. Gerndt
Technische Universität München

19 April 2010
Atlanta, GA, USA
Motivation

Common performance analysis procedure on Power6 systems
- Use Tprof to pinpoint time-consuming subroutines
- Use Xprofiler to understand call graph
- Use hpmcount (libhpm) to measure HW Counters

Problem
- Mostly post-development process
  → Learning new tools required
  → Hard to map bottlenecks to their source code location
- Routine, error-prone and time-consuming

Solution
- Automate performance analysis
- Integrate parallel application development and performance analysis within the same IDE
Related Work

• **Tools having separate user interfaces**
  - Tailored to gain maximum flexibility when presenting the collected data
  - Often hard to map the detected bottlenecks to their exact source location
  - External to user's development environment
    → impose greater learning overhead
    → require switching of applications (development/analysis tools)
  - Examples are Vampir, SCALASCA, IBM HPCS Toolkit, etc.

• **Tools being integrated in existing IDEs**
  - Provide smooth transition between the analysis results and their source code regions
  - Tend to be easier to use as the developers do not have to learn new user interfaces and/or different tools
  - Examples are VTune, TAU, HPCToolkit, PPW, etc.
Periscope performance analysis toolkit

On-line
- no need to store trace files

Distributed
- reduced network overhead
- based on autonomous cooperating agents

Analyzes:
- Single-node Performance
  - Intel Itanium2
  - IBM Power6
  - x86-based Systems
- MPI Communication
- OpenMP Performance

Supports: Fortran, C/C++
Periscope GUI Overview

Integrates with the Eclipse Development Platform
- Open-source, extensible and very popular IDE
- Supports different programming languages: C/C++, Fortran, etc.
- Uses the Eclipse Parallel Tools Platform (PTP) which provides a higher-level abstraction of the underlying parallel system

Designed to combine
- Performance measurement functionality of Periscope
- Advanced IDE functions like code indexing, refactoring, etc.

Features
- Multi-functional table to display the detected bottlenecks
- Outline of the instrumented code regions
- Clustering techniques to get classes of similarly behaving processes
- Supports both local and remote projects
- Higher-level configuration and execution of performance experiments
Periscope GUI Overview

- Source code editor
- Project explorer
- Instrumentation outline view
- Periscope properties view
Multi-functional table based on the **OSEE XViewer**

- Simple and clean tree-based overview
- Multi-level grouping
- Complex data filtering
- Multiple criteria sorting algorithm
- Navigation from the properties to their source code location
Standard Intermediate Representation (SIR) View

- Resembles the code outline view of the Eclipse C/C++ Development Tooling
- Outlines the instrumented code regions and their nesting
- Shows the number of properties in each region
- Assists code navigation
- Filters the displayed properties
Eclipse File System (EFS)
- Abstracts the underlying file system details
  → *Any supported file system can be used: Remote projects using SSH/FTP/DStore, Local, Zip, etc.*

- Source files of the analyzed application reside only on the remote
  → *no need for synchronization*

Remote Development Tools (RDT)
- Part of Eclipse Parallel Tools Platform (PTP) Project
- Remote Compilation
- Remote Indexing
- Currently supports only C/C++ applications
External Tools Framework (ETFw)

- Part of Eclipse Parallel Tools Platform (PTP) Project
- More convenient environment using ETFw's Profile launch configuration
  → no terminal access needed
  → higher level configuration and automation possible
Clustering support

Properties summarization
- Metaproperties

Needed for peta-scale PA

Identify *hidden* behavior

Based on the Weka workbench:
- *Waikato Environment for Knowledge Analysis*
- Uses K-Means algorithm
- Groups properties based on CPU distribution and code region

Results shown in a table view similar to the properties view
Future Work

Management and comparison of multiple experiments

Enhancing the clustering functionality

- Add pre- and post-processing steps to improve the quality of the results
- Use attribute selection techniques to highlight the most variable data points

Sharing the collected data with other performance tools

- Integrate with a generic performance database, e.g. PerfDMF (TAU)
- Allow the developer to easily apply more than one tool on the same project
Thank you for your attention!

Further information:
http://www.lrr.in.tum.de/periscope