Performance Analysis with Periscope

M. Gerndt, V. Petkov, 
Y. Oleynik 
Technische Universität München 
periscope@lrr.in.tum.de 
March 2012
Outline

- Motivation
- Periscope overview
- Periscope performance analysis model
- Performance analysis automation
- Periscope GUI
Motivation

- Performance analysis procedure on POWER6 as an example:
  - Use Tprof to pinpoint time consuming subroutines
  - Use Xprofiler (GUI for gprof) to understand call graph
  - Use hpmcount (libhpm) to measure Hardware Counters
  - Use mpitrace to investigate mpi communication

- Problems:
  - Time consuming
  - Error prone
  - Not scalable
  - Requires deep hardware knowledge

- Solution:
  - Performance analysis automation
Periscope

- **Distributed architecture**
  - Analysis performed by multiple distributed hierarchical agents
- **Iterative online analysis**
  - Measurements are configured, obtained and evaluated on the fly
  - no tracing!
- **Automatic bottlenecks search**
  - Based on performance optimization experts' knowledge
- **Enhanced GUI**
  - Eclipse based integrated development and performance analysis environment
- **Instrumentation**
  - Fortran, C/C++
  - Automatic overhead control
Distributed Architecture

- Graphical User Interface
  - Eclipse-based GUI
- Interactive frontend
  - Analysis control
  - Agents network
- Monitoring Request Interface
  - MRIMonitor/Score-P
  - Application
Iterative Online Analysis

Start → Candidate Properties → Raw Performance Data

Refinement

Location → Monitoring Requests → Performance Measurements

Precision

Proven Properties → Analysis

Final Properties Report

Analysis Agents

Instrumented Application
Automatic search for bottlenecks

• Automation based on formalized expert knowledge
  – Potential performance problems → properties
  – Efficient search algorithm → search strategies

• Performance property
  – Condition
  – Confidence
  – Severity

• Performance analysis strategies
  – Itanium2 Stall Cycle Analysis
  – IBM POWER6 Single Core Performance Analysis
  – MPI Communication Pattern Analysis
  – Generic Memory Strategy
  – OpenMP-based Performance Analysis
  – Scalability Analysis – OpenMP codes
Example Properties

- **StallCycles** (Region, Rank, Thread, Metric, Phase)
  - Condition
    - Percentage of lost cycles >30%
  - Severity
    - Percentage of lost cycles

- **MPI Late Sender**
  - Automatic detection of wait patterns
  - Measurement on the fly
  - No tracing required!

- **OpenMP Synchronization properties**
  - Critical section overhead property
  - Frequent atomic property
Scalability Analysis – OpenMP codes

- Identifies the OpenMP code regions that do not scale well
- Scalability Analysis is done by the frontend / restarts the application /
- No need to manually configure the runs and find the speedup!

Frontend initialization

Frontend.run()
  i. Starts application
  ii. Starts analysis agents
  iii. Receives found properties

Configuration 1, 2, ..., 2^n

After n runs

Extracts information from the found properties
Does Scalability Analysis
Exports the Properties
GUI-based Analysis
Thank you for your attention!

- **Current version 1.4**
  - Available under: http://www.lrr.in.tum.de/periscope/Download

- **Supported architectures**
  - SGI Altix 4700 Itanium2
  - IBM Power575 POWER6
  - IBM BlueGene/P
  - x86-based architectures

- **Further information:**
  - Periscope web page: http://www.lrr.in.tum.de/periscope
  - Contact us directly at: periscope@lrr.in.tum.de