Performance Analysis with Periscope

M. Gerndt, V. Petkov, Y. Oleynik
Technische Universität München

periscope@lrr.in.tum.de

October 2012
Outline

• Periscope overview
• Periscope performance analysis model
• Performance analysis automation
• Periscope GUI
Periscope

- **Distributed architecture**
  - Analysis performed by multiple distributed hierarchical agents

- **Iterative online analysis**
  - Measurements are configured, obtained and evaluated on the fly
  - No tracing files needed

- **Automatic bottlenecks search**
  - Based on performance optimization experts' knowledge

- **Enhanced GUI**
  - Eclipse based integrated development and performance analysis environment

- **Instrumentation**
  - Done by Score-P
Distributed Architecture

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

GUI

Start

Analysis Agents

Refinement

Candidate Properties

Location

Preciseness

Proven Properties

Analysis

Final Properties Report

Instrumented Application

Monitoring Requests

Performance Measurements

Raw Performance Data
Periscope Phases

- Periscope performs multiple iterative performance measurement experiments on the basis of *Phases*:
  - All measurements are performed inside phase
  - Begin and end of phase are global synchronization points
  - Automatic restart might be necessary

- Region needs to be marked as an Online Access Phase to use the Score-P Online Access Interface
  - Typically main loop of application → no need for restart, faster analysis
  - Unnecessary code parts are not measured → less measurements overhead
  - Severity value is normalized on the main loop iteration time → more precise performance impact estimation
Definition of Online Access Phases

```c
#include <scorep/SCOREP_User.h>
void foo()
{
    SCOREP_USER_REGION_DEFINE( my_region_handle )
    for(i=0;...
    {
        SCOREP_USER_OA_PHASE_BEGIN( my_region_handle, "OA_Phase",SCOREP_USER_REGION_TYPE_COMMON )
        // do something
        ...
        SCOREP_USER_OA_PHASE_END( my_region_handle )
    }
}
```
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
  – Westmere Single-node Analysis
  – Itanium2 Stall Cycle Analysis
  – IBM POWER6 Single Core Performance Analysis
  – MPI Communication Pattern Analysis
  – Generic Memory Analysis
  – 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 1.5
  – Available under: http://www.lrr.in.tum.de/periscope/Download

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