Recent Developments in Score-P and Scalasca V2

Aug 2015 | Bernd Mohr

9th Scalable Tools Workshop
Lake Tahoe
YOU KNOW YOU MADE IT …
… IF LARGE COMPANIES “STEAL” YOUR STUFF
Introducing the Intel® Trace Analyzer and Collector Performance Assistant

Motivation: Improve method of performance analysis via the GUI

Solution:

- Define common/known performance problems
- Automate detection via the Intel® Trace Analyzer

Example: A “Late Broadcast” is not easy to identify with existing views

Source:
New “Performance Assistant” Chart Added

List of all performance issues found

Duration: Runtime wasted due to perf problem

Source:
Which Performance Issues are automatically identified?

Point-to-point exchange problems:

- Late Sender

Problems with global collective operation performance:

- Wait at Barrier
- Early Reduce
- Late Broadcast

Scalasca

- **Scalable Analysis of Large Scale Applications**
- **Approach**
  - **Instrument** C, C++, and Fortran parallel applications
    - Based on MPI, OpenMP, SHMEM, or hybrid
  - **Option 1: scalable call-path profiling**
  - **Option 2: scalable event trace analysis**
    - **Collect** event traces
    - **Search** trace for event patterns representing inefficiencies
    - **Categorize and rank** inefficiencies found
    - Supports MPI 2.2 (P2P, collectives, RMA, IO) and OpenMP 3.0 (exception: nesting)

http://www.scalasca.org/
Optimized measurement configuration

Score-P

Instr. target application

Instrumenter compiler / linker

Instrumented executable

Source modules

PAPI

Local event traces

Scalasca Trace Analyzer

Wait-state report

Summary report

Report manipulation

Which problem?

Where in the program?

Which process?

CUBE
## Scalasca Command

<table>
<thead>
<tr>
<th></th>
<th>Scalasca 1</th>
<th>Scalasca 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare application objects</td>
<td>1) scalasca –instrument <code>&lt;compile-or-link-command&gt;</code></td>
<td>1) scalasca –instrument <code>&lt;compile-or-link-command&gt;</code>*</td>
</tr>
<tr>
<td>and executable for measurement</td>
<td>2) skin <code>&lt;compile-or-link-command&gt;</code></td>
<td>2) skin <code>&lt;compile-or-link-command&gt;</code>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3) scorep <code>&lt;compile-or-link-command&gt;</code>**</td>
</tr>
<tr>
<td>Run application under control</td>
<td>1) scalasca –analyze <code>&lt;application-launch-command&gt;</code></td>
<td></td>
</tr>
<tr>
<td>of measurement system</td>
<td>2) scan <code>&lt;application-launch-command&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3) set environment variables and run as usual</td>
<td></td>
</tr>
<tr>
<td>Interactively explore</td>
<td>1) scalasca –examine `&lt;experiment-archive</td>
<td>report&gt;`</td>
</tr>
<tr>
<td>measurement analysis report</td>
<td>2) square `&lt;experiment-archive</td>
<td>report&gt;`</td>
</tr>
</tbody>
</table>

* command is deprecated and only provided for backwards compatibility with Scalasca 1.x.
** recommended option
## Scalasca 1 vs Scalasca 2

<table>
<thead>
<tr>
<th>Feature</th>
<th>Scalasca 1</th>
<th>Scalasca 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instrumentation</strong></td>
<td>EPIK</td>
<td>Score-P</td>
</tr>
<tr>
<td><strong>Command line switches</strong></td>
<td></td>
<td>different</td>
</tr>
<tr>
<td><strong>Manual instrumentation API</strong></td>
<td></td>
<td>different</td>
</tr>
<tr>
<td><strong>Environmental variables</strong></td>
<td></td>
<td>different</td>
</tr>
<tr>
<td><strong>Memory buffers</strong></td>
<td>separate for each thread</td>
<td>memory pool on each process</td>
</tr>
<tr>
<td><strong>Trace format</strong></td>
<td>EPILOG</td>
<td>OTF2</td>
</tr>
<tr>
<td><strong>Structure of the filterfile</strong></td>
<td></td>
<td>different</td>
</tr>
<tr>
<td><strong>Scalable I/O</strong></td>
<td>supports SIONlib</td>
<td>partially supports SIONlib</td>
</tr>
<tr>
<td><strong>Report format</strong></td>
<td>CUBE3</td>
<td>CUBE4</td>
</tr>
<tr>
<td><strong>Experiment directory</strong></td>
<td>epik_</td>
<td>scorep_</td>
</tr>
<tr>
<td><strong>License</strong></td>
<td></td>
<td>3-clause BSD</td>
</tr>
</tbody>
</table>
For more information

- Zhukov, I. ; Feld, C. ; Geimer, M. ; Knobloch, M. ; Mohr, B. ; Saviankou, P.

**Scalasca v2: Back to the Future**
Niethammer, Christoph (Editor), ISBN: 978-3-319-16011-5
[doi:10.1007/978-3-319-16012-2_1]
Integration

• Need integrated tool (environment) for all levels of parallelization
  ▪ Inter-node (MPI, PGAS, SHMEM)
  ▪ Intra-node (OpenMP, multi-threading, multi-tasking)
  ▪ Accelerators (CUDA, OpenCL)

• Integration with performance modeling and prediction

• No tool fits all requirements
  ▪ Interoperability of tools
  ▪ Integration via open interfaces
Score-P Functionality

• Provide typical functionality for HPC performance tools
• Instrumentation (various methods)
  - Multi-process paradigms (MPI, SHMEM)
  - Thread-parallel paradigms (OpenMP, POSIX threads)
  - Accelerator-based paradigms (CUDA, OpenCL)
  - And their combination
• Flexible measurement without re-compilation:
  - Basic and advanced profile generation
  - Event trace recording
  - Online access to profiling data
• Highly scalable I/O functionality
• Support all fundamental concepts of partner’s tools
Non-functional Requirements

- **Portability**: support all major HPC platforms
  - IBM Blue Gene, Cray X*, Fujitsu K/FX10
  - x86, x86_64, PPC, Sparc, ARM clusters (Linux, AIX, Solaris)
- **Scalability**
  - Petascale, supporting platforms with more than 100K cores
- **Low measurement overhead**
  - Typically less than 5%
- **Robustness and QA**
  - Nightly Builds, Continuous Integration Testing Framework
- Easy and uniform installation through **EasyBuild**
- Open Source: New BSD License
Note!
- Only 1 tool chain (compiler/mpi combination)
- Only 1 version
Score-P Partners

- Forschungszentrum Jülich, Germany
- German Research School for Simulation Sciences, Aachen, Germany
- Gesellschaft für numerische Simulation mbH Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA
The Score-P Tool Ecosystem

Periscope

TAU ParaProf

Online interface

Score-P

TAU PerfExplorer

CUBE4 report

CUBE4 report

Scalasca

wait-state analysis

Vampir

Remote Guidance

OTF2 traces

Instrumented target application

PAPI
Past Funded Integration Projects

• **SILC (01/2009 to 12/2011)**
  - Unified measurement system (Score-P) for Vampir, Scalasca, Periscope

• **PRIMA (08/2009 to 10/2013)**
  - Integration of TAU and Scalasca

• **LMAC (08/2011 to 07/2013)**
  - Evolution of Score-P
  - Analysis of performance dynamics

• **H4H (10/2010 to 09/2013)**
  - Hybrid programming for heterogeneous platforms

• **HOPSA (02/2011 to 01/2013)**
  - Integration of system and application monitoring
Current Funded Integration Projects

- **Score-E (10/2013 to 09/2016)**
  - Analysis and Optimization of Energy Consumption
- **PRIMA-X (11/2014 to 10/2017)**
  - Extreme scale monitoring and analysis
- **RAPID (04/2014 to 03/2015)**
  - Enhanced support for node-level programming models
    - POSIX, ACE, Qt threads, MTAPI
  - Microsoft Windows support
- **Mont-Blanc-2 (10/2013 to 09/2016)**
  - OpenCL support
  - OmpSs support
CUBE V4 PLUGIN INTERFACE
GUI Plugin: CallGraph
Cube Viz Plugins: Phase Heatmap

- **Phase profiling**

- Collects data for each instance of phases marked in program instead of aggregating it

- Shows data over “time” (phase instances) for each rank/thread
Cube Viz Plugins: Phase Barplot

- **Phase profiling**
- Collects data for each instance of phases marked in program instead of aggregating it
- Shows min/max/avg metric value over “time” (phase instances)
Integration of Measurement and Modelling

- Example: DFG SPPEXA Catwalk Project

```
main() {
  foo()
  bar()
  compute()
}
```

Performance measurements (profiles)

- All functions

<table>
<thead>
<tr>
<th>Rank</th>
<th>Function</th>
<th>Model [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bar()</td>
<td>$4.0 \times p + 0.1 \times \log(p)$</td>
</tr>
<tr>
<td>2</td>
<td>compute()</td>
<td>$0.5 \times \log(p)$</td>
</tr>
<tr>
<td>3</td>
<td>foo()</td>
<td>65.7</td>
</tr>
</tbody>
</table>

Instrumentation

Automated modeling

- DFG SPPEXA Catwalk Project
Catwalk: Result Visualization

Selected:
main::parseinput::ewdfilesize_8

Value:
1.02e-05*x^(2/1)*log(x)^(2) + 6.67e+01

Graph showing the relationship between x and y values with a logarithmic scale.
CUBE Derived Metrics

- Cube v4 now also supports definition of derived metrics
  - Based on CubePL DSL
  - PreDerived and PostDerived metrics
- List of selected features:
  - Support for various arithmetic calls
  - Support of arrays and variables
  - Automatic data type conversion
  - Lambda-function definitions
  - Predefined variables
  - Redefinition of aggregation operation

Saviankou, P. ; Knobloch, M. ; Visser, A. ; Mohr, B.
Cube v4: From Performance Report Explorer to Performance Analysis Tool
International Conference On Computational Science (ICCS 2015)
Procedia computer science 51, 1343 - 1352 (2015) [doi:10.1016/j.procs.2015.05.320]
SUCCESS STORIES
Performance Tool Scaling: Scalasca

- **Latest test case**
  - Granular Dynamics Simulation
  - Based on Physics Engine (PE) Framework (Erlangen)
  - PRACE @ ISC Award winner
  - MPI only

- **Scalasca 1.x Experiments on JUQUEEN**
  - Full machine experiment: 28,672 nodes x 32 MPI ranks
    - 917,504 processes [Limit: Memory / System metadata]
    - Largest no. of threads: 20,480 nodes x 64 MPI ranks
      - 1,310,720 processes [Limit: Memory / System metadata]

- **Scalasca 2.x / Score-P 1.4.1 NAS BT-MZ on JUQUEEN**
  - Profiles: 16,384 x 64 = 1,048,576 threads [Limit: BT-MZ]
  - Traces: 10,240 x 64 = 655,360 thread [Limit: OTF2]
Scalasca: 1,310,720 process test case
Showcase: TerrSysMP

- Scale-consistent highly modular integrated multi-physics sub-surface/surface hydrology-vegetation atmosphere modelling system

- Fully-coupled MPMD simulation consisting of
  - COSMO (Weather prediction)
  - CLM (Community Land Model)
  - ParFlow (Parallel Watershed Flow)
  - OASIS coupler
Success Story: TerrSysMP

- Identified several sub-components bottlenecks:
  - Inefficient communication patterns
  - Unnecessary/inefficient code blocks
  - Inefficient data structures

- Performance of sub-components improved by factor of 2!

- Scaling improved from 512 to 32768 cores!
The Team

Markus Geimer  Michael Knobloch  Bernd Mohr  Christian Rössel  Marc Schlütter

Pavel Saviankou  Alexandre Strube  Brian Wylie  Anke Visser  Ilja Zhukov

Sponsors
Questions?

- Check out http://www.scalasca.org
- Or contact us at scalasca@fz-juelich.de