Analysis report examination with Cube

Brian Wylie
Jülich Supercomputing Centre
Cube

- Parallel program analysis report exploration tools
  - Libraries for XML+binary report reading & writing
  - Algebra utilities for report processing
  - GUI for interactive analysis exploration
    - Requires Qt4 ≥4.6 or Qt 5

- Originally developed as part of the Scalasca toolset

- Now available as a separate components
  - Can be installed independently of Score-P, e.g., on laptop or desktop
  - Latest release: Cube v4.4.x (March 2019)
Analysis presentation and exploration

- Representation of values (severity matrix) on three hierarchical axes
  - Performance property (metric)
  - Call path (program location)
  - System location (process/thread)

- Three coupled tree browsers

- Cube displays severities
  - As value: for precise comparison
  - As color: for easy identification of hotspots
  - Inclusive value when closed & exclusive value when expanded
  - Customizable via display modes
Inclusive vs. exclusive values

- **Inclusive**
  - Information of all sub-elements aggregated into single value
- **Exclusive**
  - Information cannot be subdivided further

```c
int foo()
{
    int a;
    a = 1 + 1;
    bar();
    a = a + 1;
    return a;
}
```
Case study: TeaLeaf
Case study: TeaLeaf

- HPC mini-app developed by the UK Mini-App Consortium
  - Solves the linear 2D heat conduction equation on a spatially decomposed regular grid using a 5 point stencil with implicit solvers
  - Part of the Mantevo 3.0 suite

- Measurements of TeaLeaf reference v1.0 taken on Jureca cluster @ JSC
  - Using Intel 19.0.3 compilers, Intel MPI 2019.3, and Score-P 5.0
  - Run configuration
    - 8 MPI ranks with 12 OpenMP threads each
    - Distributed across 4 compute nodes (2 ranks per node)
    - Test problem "5": 4000 × 4000 cells, CG solver

```bash
% cp -r /p/scratch/share/VI-HPS/examples/TeaLeaf ./
% cd TeaLeaf
% cube scorep_tea_leaf_baseline_8x12_sum/profile.cubex
```

[GUI showing summary analysis report]
Score-P analysis report exploration (opening view)

What kind of performance metric?

Where is it in the source code? In what context?

How is it distributed across the processes/threads?
Metric selection

Selecting the “Time” metric shows total execution time.
Expanding the system tree

Distribution of selected metric for call path by process/thread
Expanding the call tree

Distribution of selected metric across the call tree

Collapsed: inclusive value
Expanded: exclusive value
Selecting a call path

Selection updates metric values shown in columns to the right
Multiple selection

Select multiple nodes with Ctrl-click
Box plot view

Box plot shows statistical distribution across the system; with min/max/avg/median/quartiles
Violin plot view

Violin plot shows graphical distribution across the system.
Topology view

Shows topological distribution across the system (here: processes × threads)
Topology view (cont.)

Selection & right-click shows details
Alternative display modes

Data can be shown in various percentage modes.
Important display modes

- **Absolute**
  - Absolute value shown in seconds/bytes/counts

- **Selection percent**
  - Value shown as percentage w.r.t. the selected node “on the left” (metric/call path)

- **Peer percent (system tree only)**
  - Value shown as percentage relative to the maximum peer value
Source-code view via context menu
Source-code view

Note:
This feature depends on file and line number information provided by the instrumentation, i.e., it may not always be available.
Context-sensitive help available for all GUI items
Scalasca report post-processing

- Scalasca’s report post-processing derives additional metrics and generates a structured metric hierarchy

- Automatically run (if needed) when using the `square` convenience command:

```bash
% square scorep_tea_leaf_baseline_8x12_sum
INFO: Post-processing runtime summarization report (profile.cubex)...
INFO: Displaying ./scorep_tea_leaf_baseline_8x12_sum/summary.cubex...

[GUI showing post-processed summary analysis report]
```
Post-processed summary analysis report

Split base metrics into more specific metrics
TeaLeaf summary report analysis (I)

91% of the execution time is computation...

...almost entirely spent in 3 OpenMP `do` loops...

...with a slight imbalance across ranks & threads
TeaLeaf summary report analysis (II)

7% of the execution time are lost due to idle threads…

…in non-OpenMP parallelized code regions
MPI communication time is negligible (0.34%); communication is only on the master threads (MPI_THREAD_FUNNELED)
Cube: Further information

- Parallel program analysis report exploration tools
  - Libraries for Cube report reading & writing
  - Algebra utilities for report processing
  - GUI for interactive analysis exploration
- Available under 3-clause BSD open-source license
- Documentation & sources:
  - http://www.scalasca.org
- User guide also part of installation:
- Contact:
  - mailto: scalasca@fz-juelich.de
Reference material
Derived metrics

- Derived metrics are defined using CubePL expressions, e.g.:
  \[
  \text{metric::time}(i)/\text{metric::visits}(e)
  \]

- Values of derived metrics are not stored, but calculated on-the-fly

- Types of derived metrics:
  - Prederived: evaluation of the CubePL expression is performed before aggregation
  - Postderived: evaluation of the CubePL expression is performed after aggregation

- Examples:
  - “Average execution time”: Postderived metric with expression
    \[
    \text{metric::time}(i)/\text{metric::visits}(e)
    \]
  - “Number of FLOP per second”: Postderived metric with expression
    \[
    \text{metric::FLOP}()/{\text{metric::time}()}
    \]
Derived metrics in Cube GUI

Collection of derived metrics

Parameters of the derived metric

CubePL expression
Example: FLOPS based on PAPI_FP_OPS and time
CUBE algebra utilities

- Extracting solver sub-tree from analysis report

```bash
% cube_cut -r '<<ITERATION>>' scorep_bt-mz_C_32x4_sum/profile.cubex
Writing cut.cubex... done.
```

- Calculating difference of two reports

```bash
% cube_diff scorep_bt-mz_C_32x4_sum/profile.cubex cut.cubex
Writing diff.cubex... done.
```

- Additional utilities for merging, calculating mean, etc.
- Default output of cube_utility is a new report utility.cubex
- Further utilities for report scoring & statistics
- Run utility with `-h` (or no arguments) for brief usage info
Iteration profiling

- Show time dependent behavior by “unrolling” iterations

- Preparations:
  - Mark loop body by using Score-P instrumentation API in your source code

```c
SCOREP_USER_REGION_DEFINE(scorep_bt_loop)
SCOREP_USER_REGION_BEGIN(scorep_bt_loop, "<<bt_iter>>", SCOREP_USER_REGION_TYPE_DYNAMIC)
SCOREP_USER_REGION_END(scorep_bt_loop)
```

- Result in the Cube profile:
  - Iterations shown as separate call trees
    - Useful for checking results for specific iterations
    - Select your user-instrumented region and mark it as loop
    - Choose “Hide iterations”
  - View the Barplot statistics or the (thread x iterations) Heatmap
Iteration profiling: Barplot

- Iterations
- Aggregation selection
Iteration profiling: Heatmap