Analysis report examination with CUBE

Christian Feld
Jülich Supercomputing Centre
CUBE

Parallel program analysis report exploration tools
- Libraries for XML report reading & writing
- Algebra utilities for report processing
- GUI for interactive analysis exploration
  - requires Qt4.6 or later

Originally developed as part of Scalasca toolset

Now available as a separate component
- Can be installed independently of Score-P, e.g., on laptop or desktop
- Latest release: CUBE 4.3.4 (April 2016)
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 colour: for easy identification of hotspots
  - Inclusive value when closed & exclusive value when expanded
  - Customizable via display modes
Analysis presentation

What kind of performance metric?

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

How is it distributed across the processes/threads?
Analysis report exploration (opening view)
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
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;
}
```
Selecting a call path

<table>
<thead>
<tr>
<th>Selection updates metric values shown in columns to right</th>
<th>Selection updates metric values shown in columns to right</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Display Topology Help</td>
<td>File Display Topology Help</td>
</tr>
<tr>
<td>Absolute</td>
<td>Absolute</td>
</tr>
<tr>
<td>Metric tree</td>
<td>Call tree</td>
</tr>
<tr>
<td>1.63e9 Visits</td>
<td>0.01 MAIN_</td>
</tr>
<tr>
<td>767.48 Time</td>
<td>0.82 mpi_setup_</td>
</tr>
<tr>
<td>0.00 Minimum Inclusive Time</td>
<td>0.00 MPI.Cast</td>
</tr>
<tr>
<td>48.58 Maximum Inclusive Time</td>
<td>0.00 env_setup_</td>
</tr>
<tr>
<td>5.27e8 bytes_sent</td>
<td>0.00 zone_setup_</td>
</tr>
<tr>
<td>5.27e8 bytes_received</td>
<td>0.00 zone_starts_</td>
</tr>
<tr>
<td></td>
<td>0.00 set_constants_</td>
</tr>
<tr>
<td></td>
<td>5.02 initialize_</td>
</tr>
<tr>
<td></td>
<td>1.11 exact_rhs_</td>
</tr>
<tr>
<td></td>
<td>0.00 timer_clear_</td>
</tr>
<tr>
<td></td>
<td>3.67 exch_sbc_</td>
</tr>
<tr>
<td></td>
<td>0.04 adi_</td>
</tr>
<tr>
<td></td>
<td>39.91 compute_rhs_</td>
</tr>
<tr>
<td></td>
<td>233.49 x_solve_</td>
</tr>
<tr>
<td></td>
<td>239.34 y_solve_</td>
</tr>
<tr>
<td></td>
<td>0.07 z_solve_</td>
</tr>
<tr>
<td></td>
<td>0.04 !omp parallel @z_solve.f43</td>
</tr>
<tr>
<td></td>
<td>100.67 !omp do @z_solve.f52</td>
</tr>
<tr>
<td></td>
<td>2.89 lhsinit</td>
</tr>
<tr>
<td></td>
<td>37.70 bincrhs</td>
</tr>
<tr>
<td></td>
<td>27.24 matvec_sub</td>
</tr>
<tr>
<td></td>
<td>36.11 matmul_sub</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Selection updates metric values shown in columns to right</th>
<th>Selection updates metric values shown in columns to right</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>767.48 (100.00%)</td>
</tr>
</tbody>
</table>
Source-code view via context menu

Right-click opens context menu
Source-code view

```
/home/geimer/Projects/Tests/NPB3.3-MZ-MPI/BT-MZ/solve_subs.f

subroutine binvcrhs(lhs,c,r)
C-------------------------------------------------------------
C-------------------------------------------------------------
C-------------------------------------------------------------
c
C----------------------------------------------------------------

implicit none

double precision pivot, coeff, lhs
dimension lhs(5,5)
double precision c(5,5), r(5)
C----------------------------------------------------------------
c
C----------------------------------------------------------------
pivot = 1.00d0/lhs(1,1)
lhs(1,2) = lhs(1,2)*pivot
lhs(1,3) = lhs(1,3)*pivot
lhs(1,4) = lhs(1,4)*pivot
lhs(1,5) = lhs(1,5)*pivot
c(1,1) = c(1,1)*pivot
c(1,2) = c(1,2)*pivot
c(1,3) = c(1,3)*pivot
c(1,4) = c(1,4)*pivot

Read only  Save          Save as          Font...          Close
```
Flat profile view

Select flat view tab, expand all nodes, and sort by value.
Box plot view

Box plot shows distribution across the system; with min/max/avg/median/quartiles
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
Multiple selection

Select multiple nodes with Ctrl-click
Derived metrics in Cube

- Value of the derived metric is not stored, but calculated on-the-fly
- One defines an CubePL expression, e.g.:
  \[ \text{metric::time}(i)/\text{metric::visits}(e) \]
- Types of derived metrics:
  - Prederived: evaluation of the CubePL expression is done before the aggregation
  - Postderived: evaluation of the CubePL expression is performed after the aggregation
- Examples:
  - “Average execution time” Postderived metric with an expression:
    \[ \text{metric::time}(i)/\text{metric::visits}(e) \]
  - “Number of FLOP per second” Postderived metric with an expression:
    \[ \text{metric::FLOP}() / \text{metric::time}() \]
Derived metrics in Cube GUI

Collection of derived metrics

Parameters of the derived metric

CubePL expression
Example derived metric FLOPS based on PAPI_FP_OPS and time
Context-sensitive help available for all GUI items
CUBE algebra utilities

- Extracting solver call-tree from analysis report
  ```bash
  % cube_cut -r '<<ITERATION>>' scorep_bt-mz_B_8x8_sum/profile.cubex
  Writing cut.cubex... done.
  ```

- Extracting partition of system-tree processes from analysis report
  ```bash
  % cube_part -R 0-2,4,6- scorep_bt-mz_B_8x8_sum/profile.cubex
  Writing part.cubex... done.
  ```

- Calculating difference of two reports
  ```bash
  % cube_diff scorep_bt-mz_B_8x8_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
Loop Unrolling

- Show time dependent behavior by unrolling iterations

- Preparations:
  - Mark loops by using Score-P user instrumentation in your source code

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

- Result in the CUBE profile:
  - Iterations shown as separate call trees
  - Useful for checking results for specific iterations
    - Or
  - Select your user instrumented region and mark it as loop
  - Choose hide iterations
  - View the Barplot statistics or the (thread x iterations) Heatmap
Loop Unrolling - Barplot

Aggregation selection

Iterations
Loop Unrolling – Heatmap
Further information

CUBE

- Parallel program analysis report exploration tools
  - Libraries for XML report reading & writing
  - Algebra utilities for report processing
  - GUI for interactive analysis exploration
- Available under New BSD open-source license
- Documentation & sources:
  - http://www.scalasca.org
- User guide also part of installation:
  - `cube-config --cube-dir` /share/doc/CubeGuide.pdf
- Contact:
  - mailto: scalasca@fz-juelich.de