Caliper:
Putting Performance Data in Context

9th Scalable Tools Workshop

August 3, 2015
Performance Analysis requires Data Correlation

- **Environment**
  - Hardware topology
  - Software configuration (OS, runtime versions)

- **Application domain**
  - Domain topology
  - Physics data

- **Source Code Context**
  - Call path
  - Phase

- **Measurements**
  - Time
  - FLOPS
  - Memory access

---

No interface to collect generic context data!
Example: Visualization in MemAxes
Caliper Contributions

1. Interface for applications to provide arbitrary context information

2. Interface for plug-in tools that provide measurement data

3. Enable composition of context annotations and measurement providers
Composite Data Collection

- Generic attribute:value data model
  - Allows storage of any type or kind of data: not limited to pre-defined context categories

- In-memory data store
  - Data sources update context / measurement data independently
  - Combines data across the software stack

Application

Component 1
- phase, iteration no.

Component 2
- coordinate, element

Libraries

Solver
- solver iteration

Mesh
- ref. level

Runtime

MPI
- rank, function, wait time

OpenMP
- thread id, construct, wait time

Measurements

- Timer
- PAPI
- ...

Common context space

app.c1.phase, app.c1.iteration, app.c2.coordinate, app.c2.element, solver.iteration, mesh.refinementlvl, mpi.rank, mpi.function, mpi.wait, openmp.thread, openmp.construct, openmp.wait, time.duration, time.timestamp, papi.l2miss, papi.flops, ...
Caliper Framework

- Annotated source code
- Measurement module
- In-memory data store
- Query API
- External on-line Tool
- Measurement trigger
- Context scope manager
- File system / Database
Caliper Workflow (1)

- Instrumented modules independently update context information
• Trigger creates record with context snapshot and measurement data
Caliper Workflow (3)

- Alternatively, external tool pulls context and measurement snapshot on-line
Data Model

- Fully flexible *attribute:value* format
  
  \texttt{app.phase="solve",mpi.rank=42,time.duration=1234,mesh.level=3}

- Attributes contain
  
  - Unique name
  - Data type (integer, floating point, string, binary blob)
    - Future extension: JSON description for complex types
  - Scope (process, thread, or task)

- Entries can be hierarchical (e.g., for call paths)

- Automatic scoping
  
  - Caliper keeps separate entries per thread or tasks

- Efficient tree-based data representation
Efficient Context Representation

- Build up *context tree*
  - Stores values from multiple attributes
  - Transparent to the user

- Represent context snapshot through single node

- Non-repetitive data (measurements) stored explicitly

```
Phase = main/loop/work; state=parallel; iteration=1
```
Data Format

- **Context streams** include performance/context and metadata records of a single Caliper instance.

- **Node records** describe context tree and metadata (attributes)
  
  __rec=node,id=24,attr=8,data=iter,parent=23

- **Context records** combine context and immediate data entries of a context snapshot
  
  __rec=ctx,ref=25,attr=19=24,data=13=1
Data Processing / Analysis Stack

- Caliper-instrumented Process
  - In-memory data store
  - Context stream
  - Local DB
  - On-line or off-line parallel merge / aggregation
  - Global DB
  - Analysis

- Caliper-instrumented process
  - In-memory data store
  - Context stream
Annotation API

- **cali::Annotation**
  - Encapsulates attribute

- **begin()**
  - Append new value

- **set()**
  - Set (overwrite) value

- **end()**
  - Remove last value

```cpp
#include <Annotation.h>

int main(int argc, char* argv[])
{
    cali::Annotation phase_ann("phase");

    phase_ann.begin("init");
    // Perform initialization
    initialize();
    phase_ann.end(); // ends "init"

    phase_ann.begin("loop");
    #pragma omp parallel for
    for (int i; i < MAX; ++i) {
        cali::Annotation("iteration").set(i);
        do_work(i);
    }
    phase_ann.end(); // ends "loop"
}
```
Service API

- Instrumentation + context query for measurement services and third-party tools
  - push_context()
    - Trigger snapshot and write to stream
  - pull_context()
    - Trigger and pull snapshot
  - create_attribute()
    - Creates attribute
  - begin() / end() / set()
    - Set values
    - (Query API) (not defined yet)

- Callback functions for various events
  - E.g. snapshot triggered, attribute created, value changed, …
## Measurement Services

<table>
<thead>
<tr>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer</td>
<td>Timestamps and/or time duration</td>
</tr>
<tr>
<td>Callpath</td>
<td>Performs stack unwinding to retrieve call path</td>
</tr>
<tr>
<td>MPI</td>
<td>Wraps MPI functions and provides MPI rank</td>
</tr>
<tr>
<td>OMPT</td>
<td>OpenMP tools interface, provides thread ID, state, and OpenMP construct wrappers</td>
</tr>
<tr>
<td>PAPI</td>
<td>PAPI hardware counters</td>
</tr>
</tbody>
</table>
Usage (1): Configure, run

- Link caliper library

- Configure
  - Add measurement services, set output flags

  ```
  $ export CALI_SERVICES_ENABLE=recorder:timestamp
  ```

- Run

  ```
  $ ./test/cali-basic
  == CALIPER: Registered recorder service
  == CALIPER: Registered timestamp service
  == CALIPER: Initialized
  == CALIPER: Wrote 38 records.
  == CALIPER: Finished
  $ ls *.cali
  150724-073336_479_ytM1by52l3yV.cali
  ```
Usage (2): Examine

- **cali-query** expands records and provides basic aggregation, filter, and merge functionality

- Export to your favorite data analytics / visualization tool

```
$ cali-query -e *.cali
```

<table>
<thead>
<tr>
<th>phase</th>
<th>time.duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>221</td>
</tr>
<tr>
<td>main/init</td>
<td>140</td>
</tr>
<tr>
<td>main</td>
<td>15</td>
</tr>
<tr>
<td>main/loop</td>
<td>12</td>
</tr>
<tr>
<td>main/loop</td>
<td>18</td>
</tr>
<tr>
<td>main/loop</td>
<td>15</td>
</tr>
<tr>
<td>main/loop</td>
<td>7</td>
</tr>
<tr>
<td>main/loop</td>
<td>6</td>
</tr>
<tr>
<td>main/loop</td>
<td>10</td>
</tr>
<tr>
<td>main/loop</td>
<td>13</td>
</tr>
<tr>
<td>main</td>
<td>7</td>
</tr>
</tbody>
</table>
```
Ongoing Work

- On-line aggregation
  - Turns Caliper into a “real” profiler

- Complex datatypes
  - Describes layout of binary blobs

- Scalable cross-process on-line merge / aggregation
  - Use MRNet
Further Information

- Available on github LLNL LC Stash:
  
  https://lc.llnl.gov/stash/projects/PIPER/repos/caliper/browse

- BSD License

- Release version available soon

- Contact:
  
  David Boehme
  boehme3@llnl.gov