Sampling XOR Instrumentation? Or both?

Scalable Tools Workshop, Lake Tahoe, 2015-08-04

Andreas Knüpfer, Bert Wesarg, Thomas Ilsche, Ronny Tschüter, Joseph Schuchart, Hartmut Mix, Holger Brunst from ZIH, TU Dresden, Germany
Overview

- Introduction and existing approaches
- Recording and data formats
- Analysis of samples and events combined
  - Timeline visualization
  - Statistics
- Conclusions
Certain terms are used almost synonymously even though they aren’t.

**Definitions**

- **Acquisition**
  - Sampling
- **Recording**
  - Summarization
  - Logging
- **Data representation**
  - Profile
  - Event/Call-Path Traces

- **Profiling**
- **Event Tracing**

See image for layout.
Existing Combinations:

- Sample one thing, instrument another:
  - Sampling of user routines or call-path tracing, instrumentation of MPI
    [Tallent et.al. 2011, Ilsche et.al. 2014]
  - Sampling of hardware counters, instrumentation of user routines and MPI

- Sampling of energy consumption next to instrumentation-based performance monitoring [Hackenberg et. al. 2014]

- Instrumentation maintains shadow stack, sampling reads it as shortcut of a stack walk [Iwainsky et.al. 2014]

- Very coarse-grained sampling, then “folding” over many repeated instances, instrumentation is only guiding the folding mechanism, instrumented events are not recorded [Servat, Ph.D. thesis 2015]
Overview

- Introduction and existing approaches
- **Recording and data formats**
- Analysis of samples and events combined
  - Timeline visualization
  - Statistics
- Conclusions
Example with Instrumentation and Sampling

- **Phase 2**: Calc
- **Phase 3**: Calc
- **Phase 4**: Calc

**Main**

- **System exist**: Main
- **Phase 2**: Calc
- **MPI**:

- **System exist**: Main
- **Phase 2**: Calc

- **System exist**: Main
- **Phase 3**: Calc

- **System exist**: Main
- **Phase 4**: Calc

**Call-stack representation**

"Trampolines" allow tracking uninterrupted calls, reduce overhead

**Flat representation**

Fine-grained call timeline from instrumentation
Samples with Calling Context Tree

Efficient storage with Calling Context Tree

System _Main
Main
Phase 2
Calc
MPI

System _Main
Main
Phase 2
Calc

System _Main
Main
Phase 3
Calc

System _Main
Main
Phase 4
Calc

Center for Information Services & High Performance Computing
Define calling context nodes recursively:

\[\text{DefCallingContext} \{
\text{CallingContextRef} \quad \text{self},
\text{RegionRef} \quad \text{region,} \quad // \text{Routine or function}
\text{SourceCodeLocationRef} \quad \text{sourceCodeLocation},
\text{CallingContextRef} \quad \text{parent} \}\]

Use them at a sample point to specify entire call stack by single ID:

\[\text{CallingContextSample} \{\n\text{<process>, <time>,}
\text{CallingContextRef} \quad \text{callingContext},
\text{uint} \quad \text{unwindDistance},
\text{InterruptGeneratorRef} \quad \text{interruptGenerator} \}\]
Now add Events from Instrumentation

Intermix Samples (S) as well as Enter (E) and Leave (L) events, all refer to the CCT.
Representation in OTF2: Special Enter/Leave Events

- Introduce new form of enter and leave events:

  CallingContextEnter {
    <process>, <time>,
    CallingContextRef callingContext,
    uint32_t unwindDistance
  }

  CallingContextLeave {
    <process>, <time>,
    CallingContextRef callingContext
  };

- Refer to CCT, easily converted to old mode for legacy purposes if needed
- Little to no storage overhead, but more information (e.g., hidden stack entries)
- ... no reason to keep the old enter/leave event records referring to routines
Introduction and existing approaches
Recording and data formats
Analysis of samples and events combined
  - Timeline visualization
  - Statistics
Conclusions
Events are status changes, usually drawn from “now” until “following event”

Samples are points in time, but usually drawn \(1\delta\) wide (with sample distance \(\delta\))

Either draw at \([t-\frac{1}{2}\delta, t+\frac{1}{2}\delta]\) or at \([t, t+\delta]\)
Combined Visualization in Timeline: Shift by $\frac{1}{2}\delta$

Unified strategy for events and samples:

- draw from “now” until “following event or sample”

Do not suppress samples in instrumented function calls (see below), but do optimize the extra stack walk.
Overview

- Introduction and existing approaches
- Recording and data formats
- Analysis of samples and events combined
  - Timeline visualization
  - Statistics
- Conclusions
How to Compute Run-Time Statistics?

- From samples alone or from samples and events combined?

<table>
<thead>
<tr>
<th></th>
<th>Time for Calc</th>
<th>Time for MPI</th>
<th>Sum time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events only</td>
<td>(10.2)</td>
<td>2.2</td>
<td>(12.4)</td>
</tr>
<tr>
<td>Samples only</td>
<td>9.3</td>
<td>3.1</td>
<td>12.4</td>
</tr>
<tr>
<td>Events AND samples</td>
<td>10.2</td>
<td>2.2</td>
<td>12.4</td>
</tr>
<tr>
<td>Events OR samples</td>
<td>9.3</td>
<td>2.2</td>
<td>11.5</td>
</tr>
</tbody>
</table>
How to Compute Run-Time Statistics?

- Cannot compute from events alone with selective instrumentation
- Do not compute some from events and some from samples (cherry picking)
- Compute from samples only: produces statistically correct results
  - Don’t expect sampling to be more precise than $1\delta$ in the first place
- Compute from samples and events combined: produces different correct result!
  - It is not more accurate than the one from sampling (max. error is the same)
  - Different granularity for instrumented calls may become evident

What is easier to comprehend by users? What is easier to explain? Which is the expected model that brings the lesser surprise?
Impressions
Conclusions & Outlook

- Sampling and Instrumentation should be combined
  - Allow a completely flexible mix from samples and events
  - Event tracing should adopt favorable event representation via CCT
  - Make sure to present it in a clear way

Release plans:
- Sampling records already part of OTF2
- Include sampling in next Score-P release
- Visualization in Vampir release version at SC’15
9th Parallel Tools Workshop in Dresden, 2-3 September
https://tools.zih.tu-dresden.de/2015/

Extreme Scale Programming Tools Workshop (ESPT) at SC’15
http://www.vi-hps.org --> News
Deadline extended until 14 August