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

ZIH
Center for Information Services &
High Performance Computing

Overview

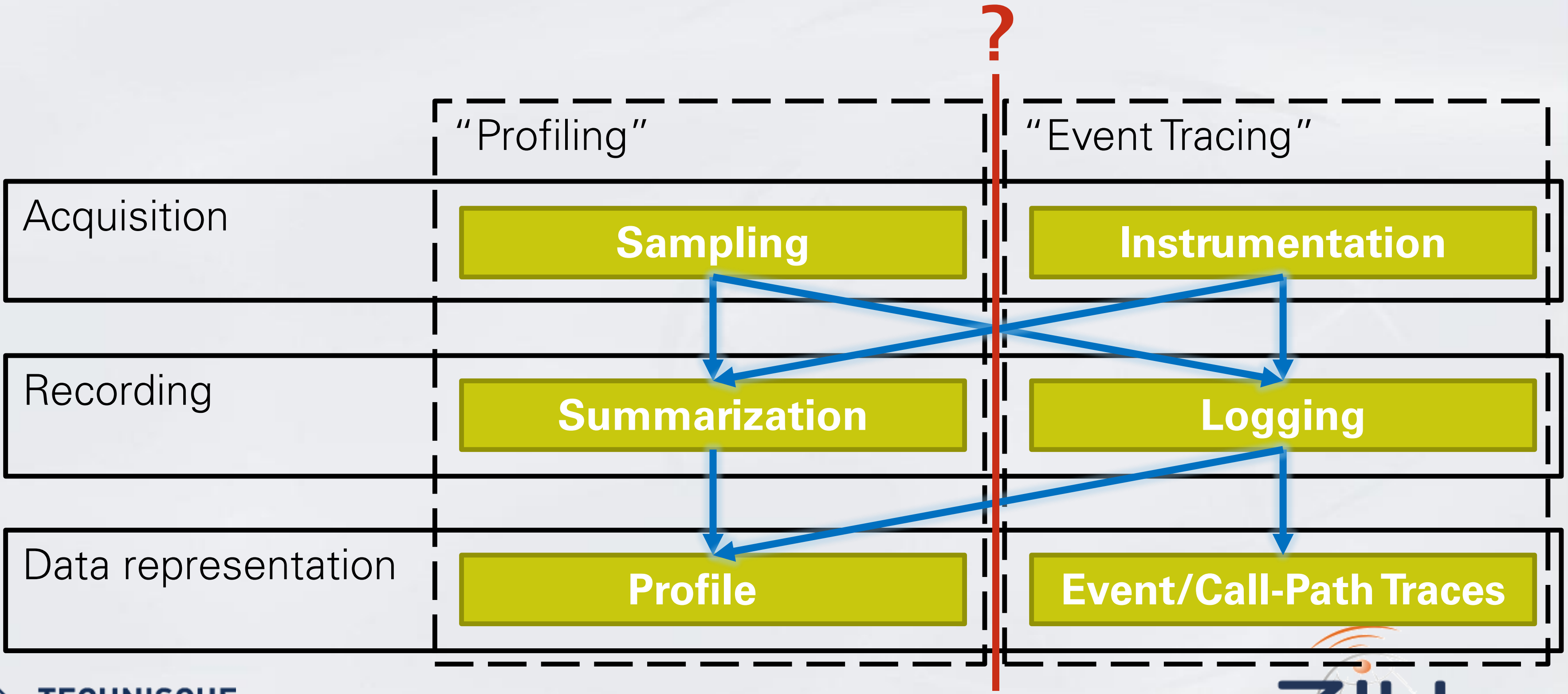
- Introduction and existing approaches
- Recording and data formats
- Analysis of samples and events combined
  - Timeline visualization
  - Statistics
- Conclusions

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Definitions

- Certain terms are used almost synonymously even though they aren't



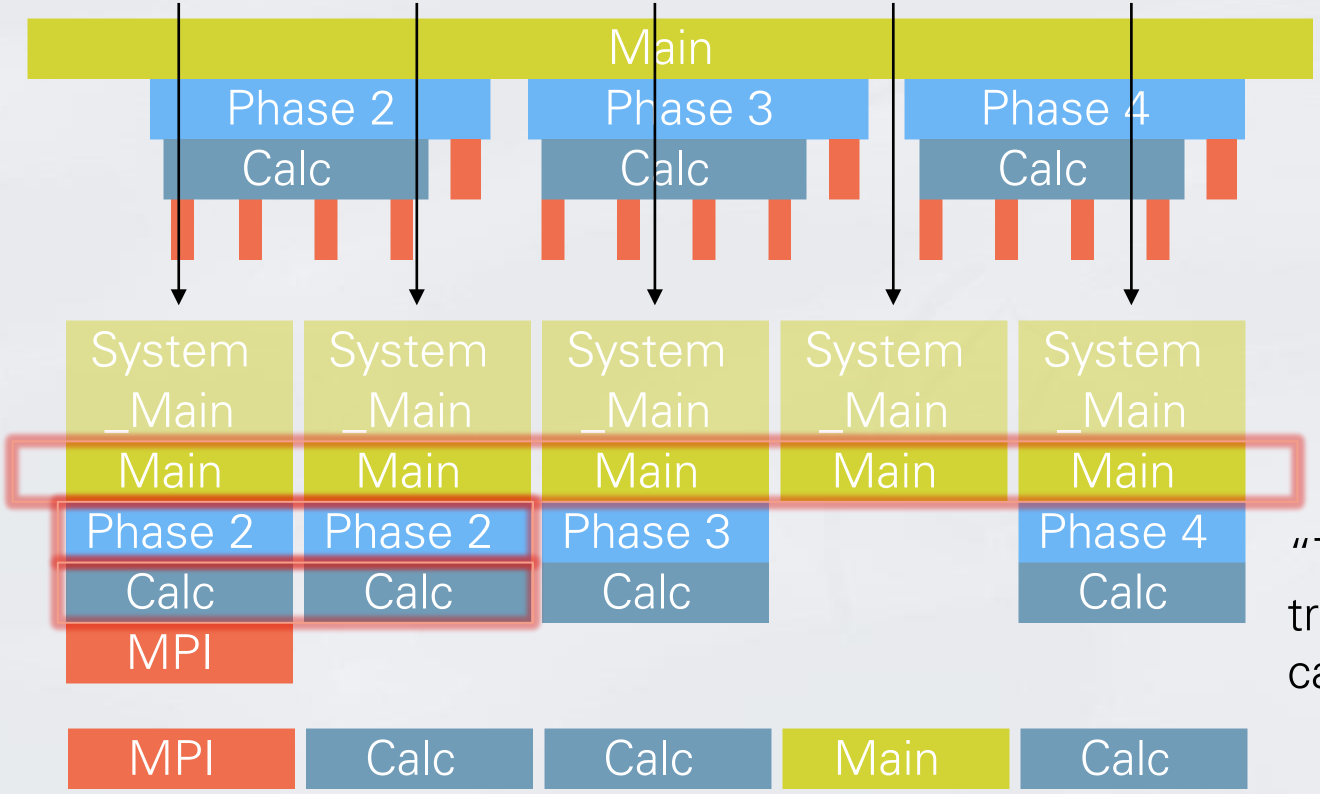
Existing Combinations:

- 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



Fine-grained call timeline from instrumentation

Call-stack representation

“Trampolines” allow tracking uninterrupted calls, reduce overhead

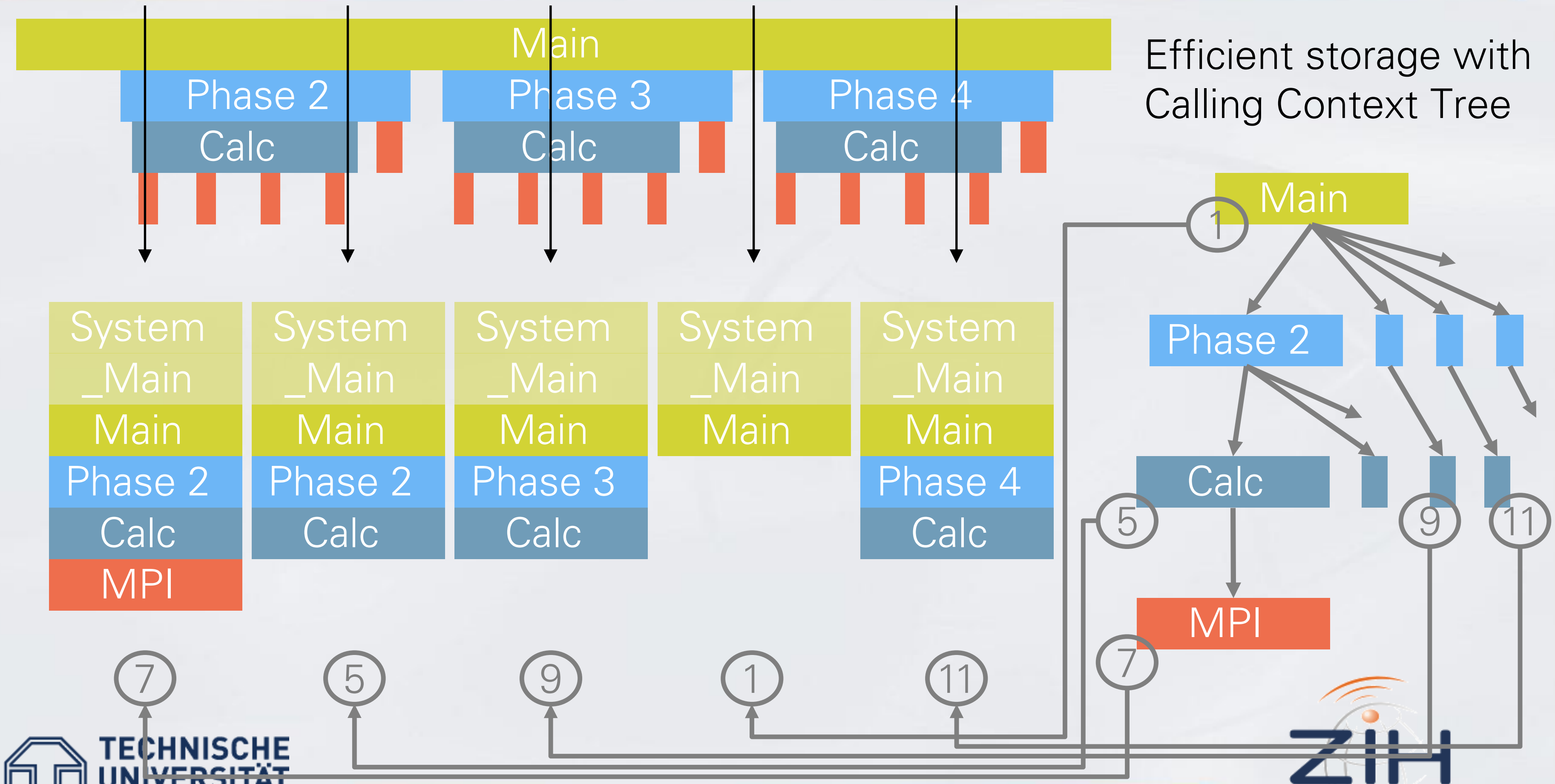
Flat representation

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Samples with Calling Context Tree

Efficient storage with Calling Context Tree



Representation in OTF2: CCT and Sample Points

- Define calling context nodes recursively:

```
DefCallingContext {
    CallingContextRef         self,
    RegionRef                 region, // Routine or function
    SourceCodeLocationRef     sourceCodeLocation,
    CallingContextRef         parent }
```

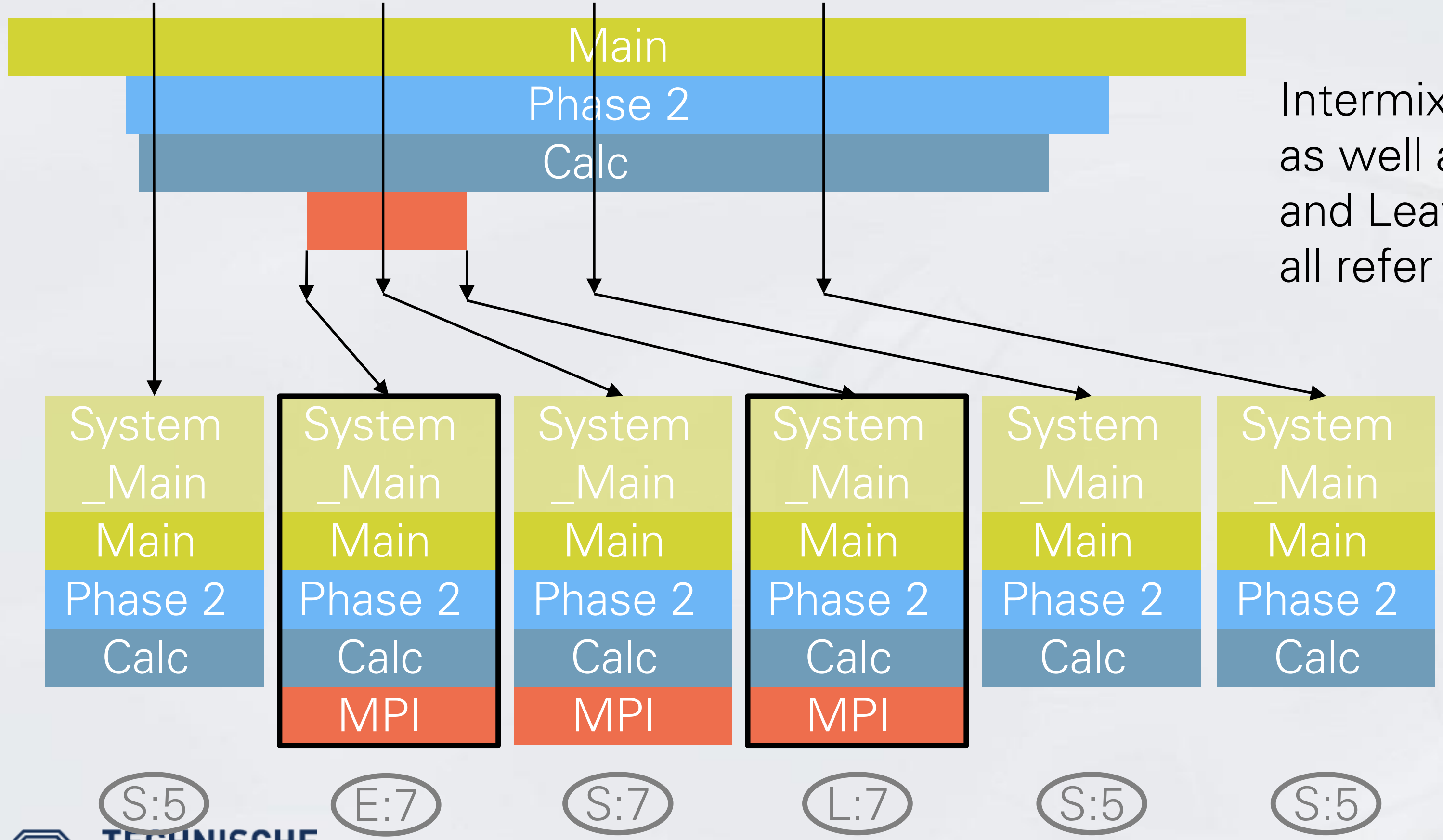
- Use them at a sample point to specify entire call stack by single ID:

```
CallingContextSample {
    <process>, <time>,
    CallingContextRef         callingContext,
    uint                      unwindDistance,
    InterruptGeneratorRef     interruptGenerator }
```

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

Now add Events from Instrumentation



Intermix Samples (S) as well as Enter (E) and Leave (L) events, all refer to the CCT

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

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

TECHNISCHE UNIVERSITÄT DRESDEN

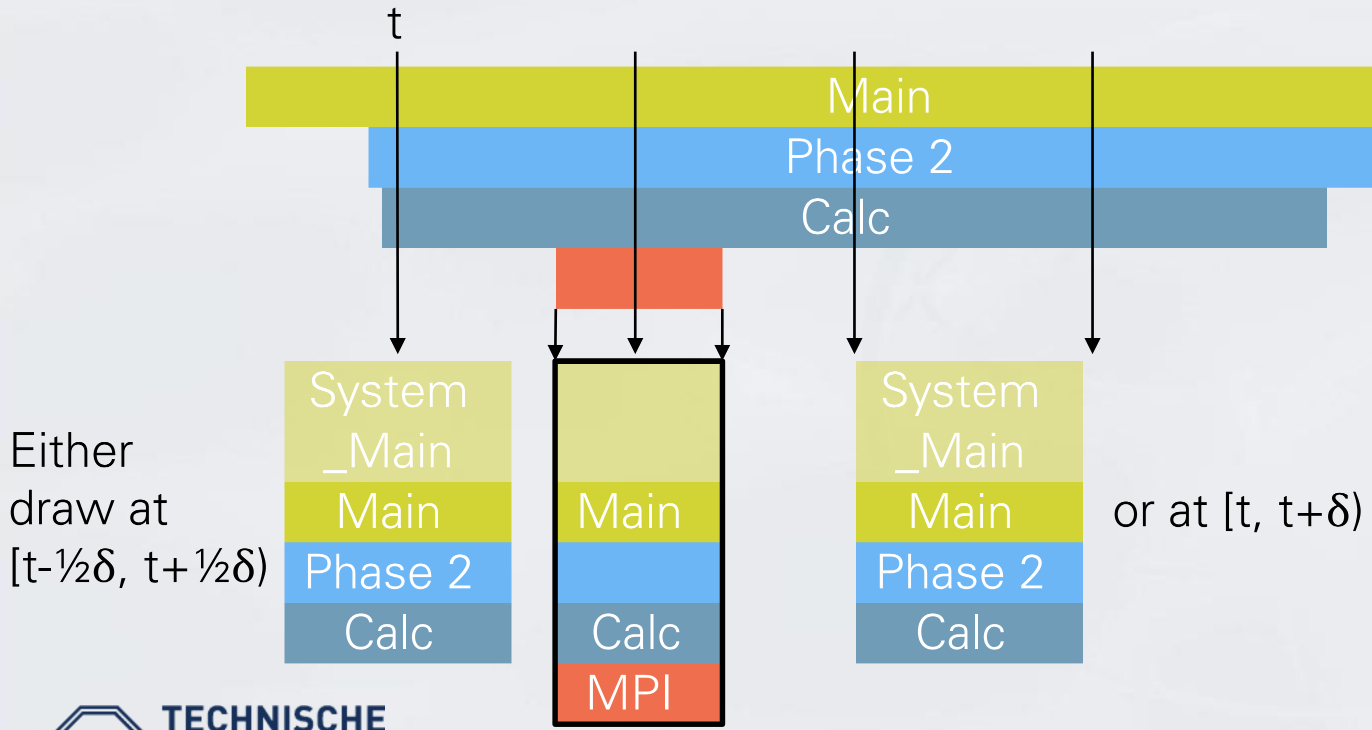
ZIH Center for Information Services & High Performance Computing

Overview

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

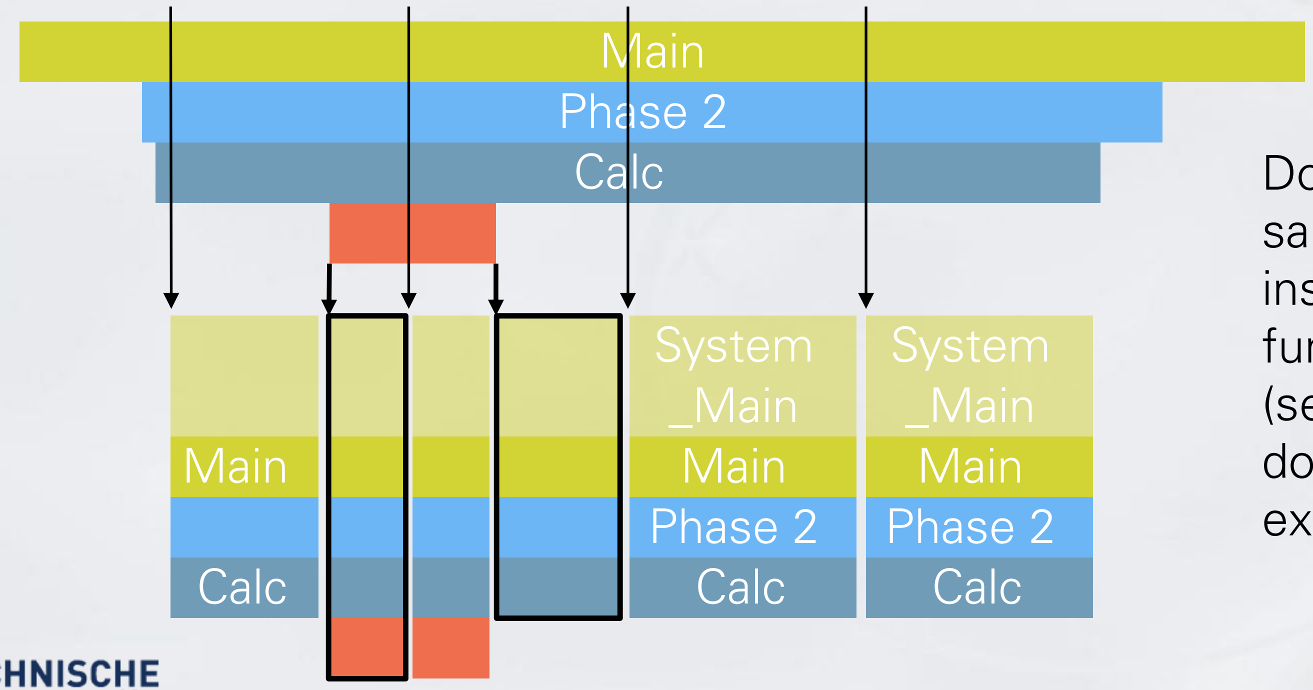
Combined Visualization in Timeline

- Events are status changes, usually drawn from “now” until “following event”
- Samples are points in time, but usually drawn 1δ wide (with sample distance δ)



Combined Visualization in Timeline: Shift by ½δ

- Unified strategy for events and samples:
  - draw from "now" until "following event or sample"



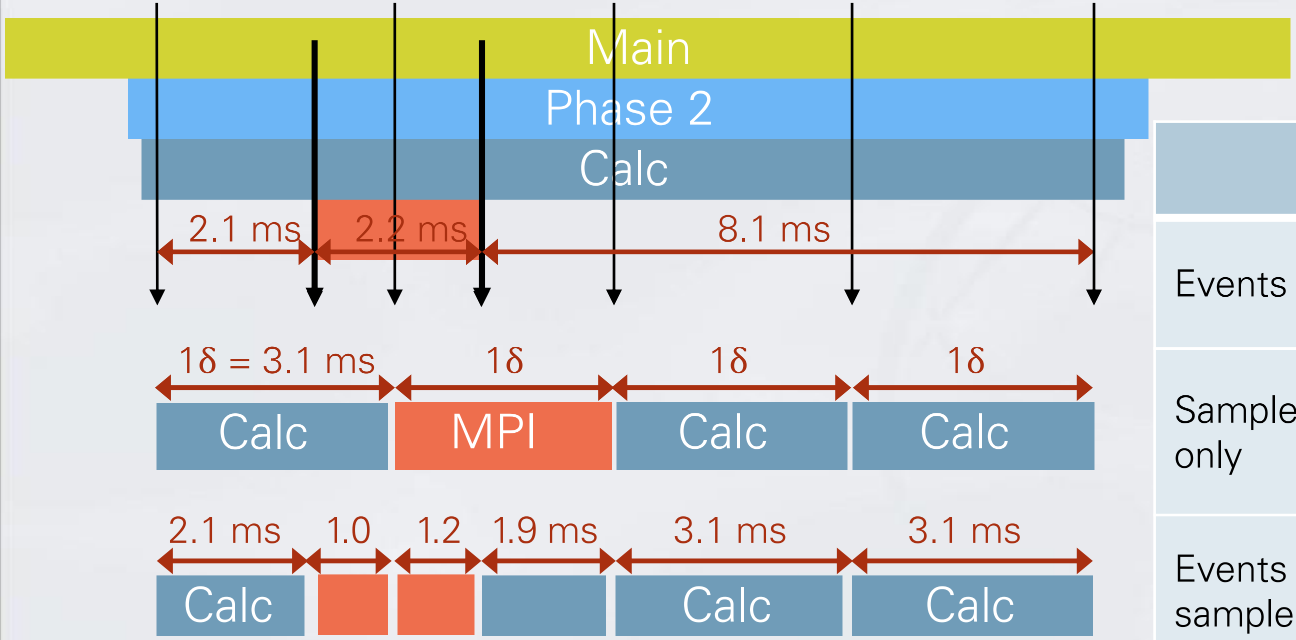
Do not suppress samples in instrumented function calls (see below), but do optimize the extra stack walk

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?



	Time for Calc	Time for MPI	Sum time
Events only	(10.2)	2.2	(12.4)
Samples only	9.3 = ¾	3.1 = ¼	12.4
Events AND samples	10.2	2.2	12.4
Events OR samples	9.3	2.2	11.5

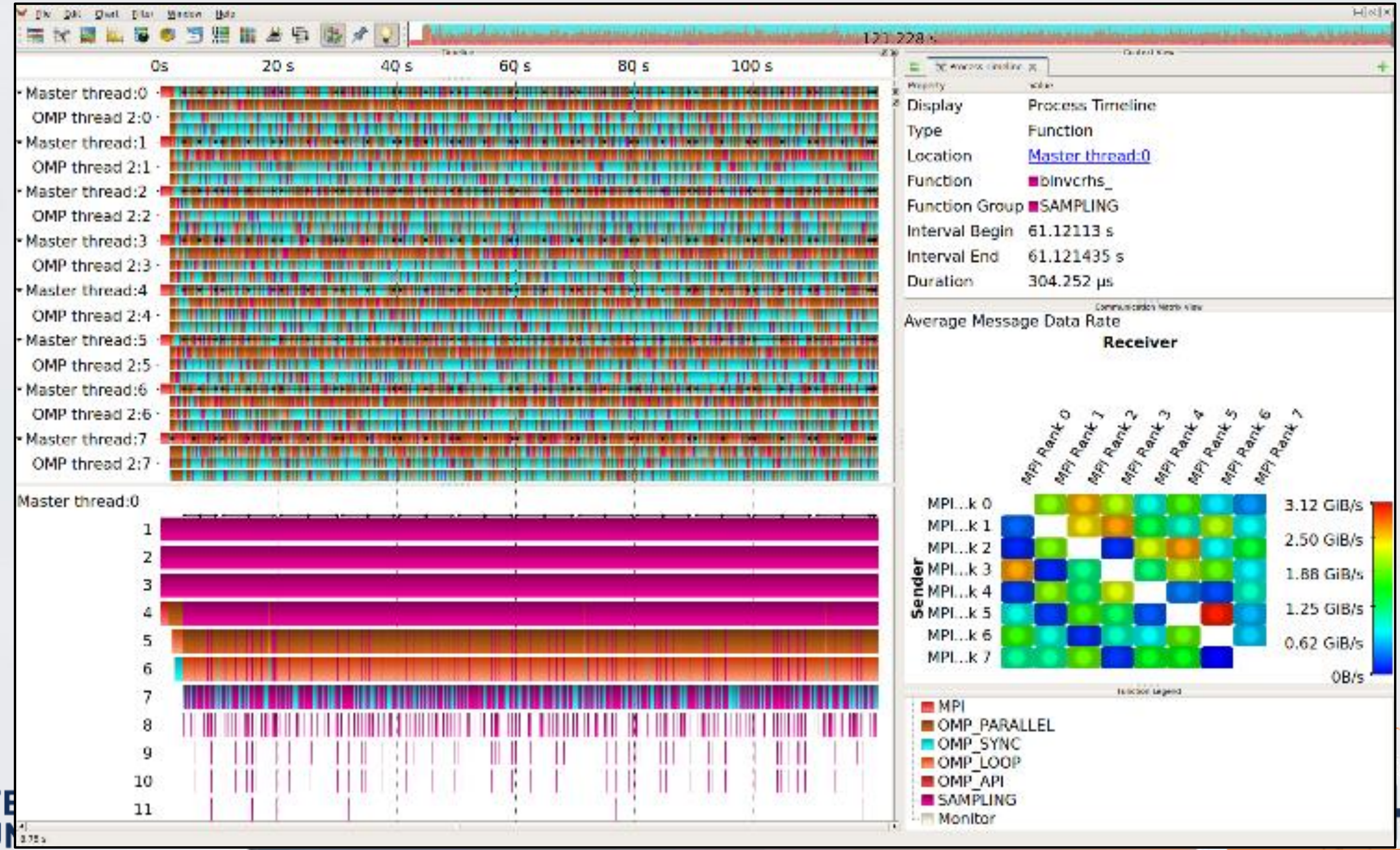
How to Compute Run-Time Statistics?

- 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 1δ 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?

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing

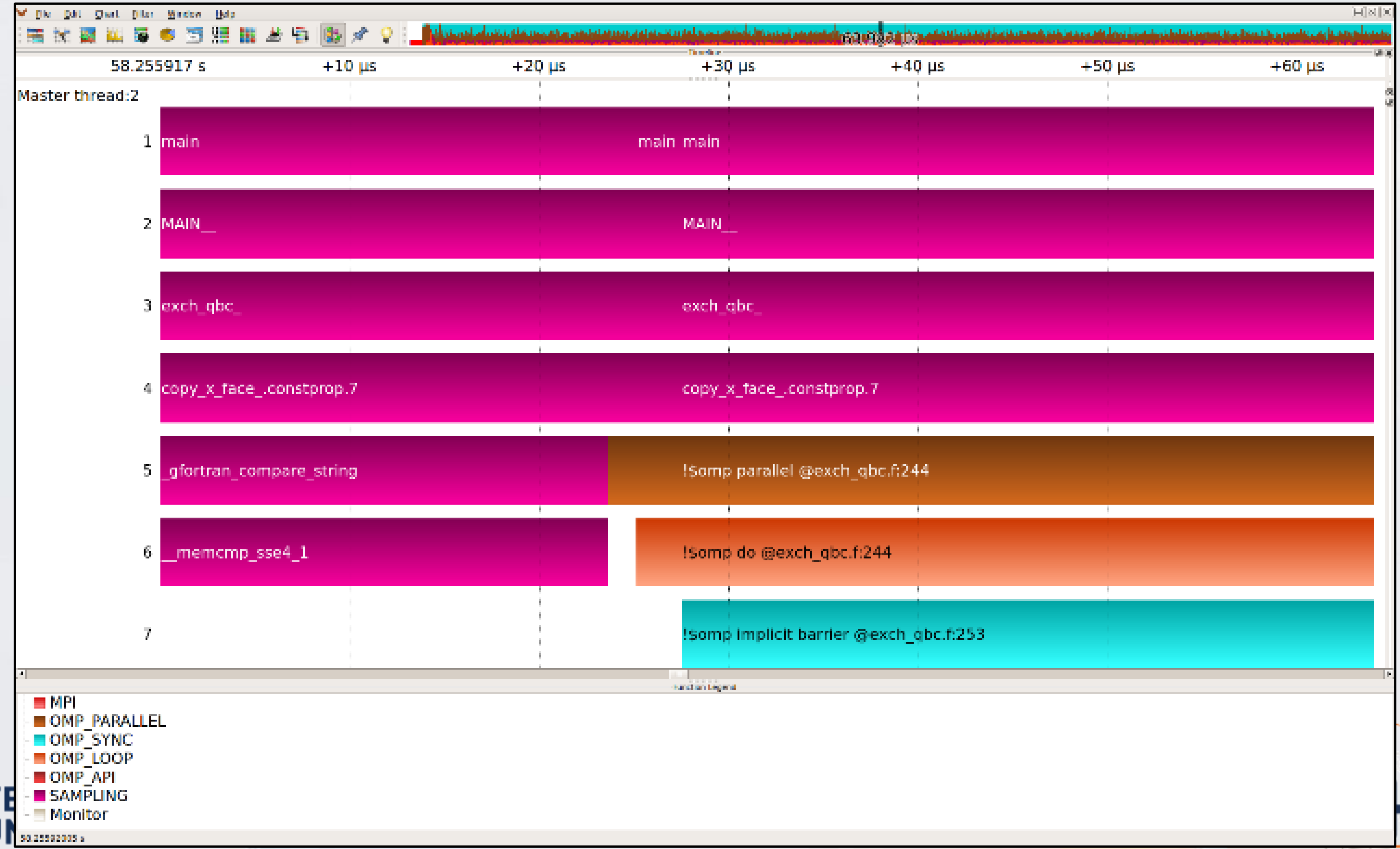
Impressions



Display	Process Timeline
Type	Function
Location	Master thread:0
Function	binvcrhs_
Function Group	SAMPLING
Interval Begin	61.12113 s
Interval End	61.121435 s
Duration	304.252 µs

- 🟥 MPI
- 🟫 OMP_PARALLEL
- 🟦 OMP_SYNC
- 🟧 OMP_LOOP
- 🟥 OMP_API
- 🟪 SAMPLING
- ⬜ Monitor

Impressions



DRESDEN

Center for Information Services & High Performance Computing

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

Advertisements

- 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

TECHNISCHE UNIVERSITÄT DRESDEN

ZIH Center for Information Services & High Performance Computing