TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Strategies for Real-Time Event Reduction

PROPER @ Euro-Par 2012, Rhodes Island

Michael Wagner and Wolfgang E. Nagel

Michael Wagner (michael.wagner@zih.tu-dresden.de)

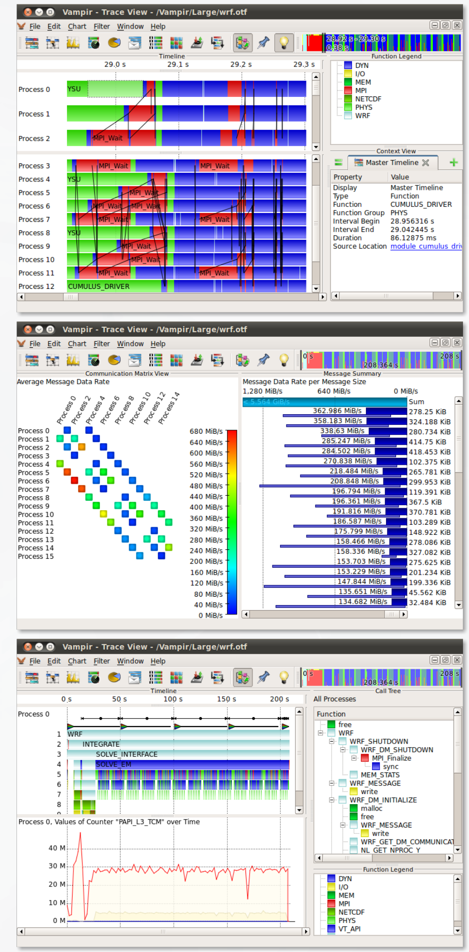
ZIH
Center for Information Services & High Performance Computing

Outline

- Introduction
- Strategies for event reduction
- Realization concepts
- Future Work
- Conclusion

Introduction

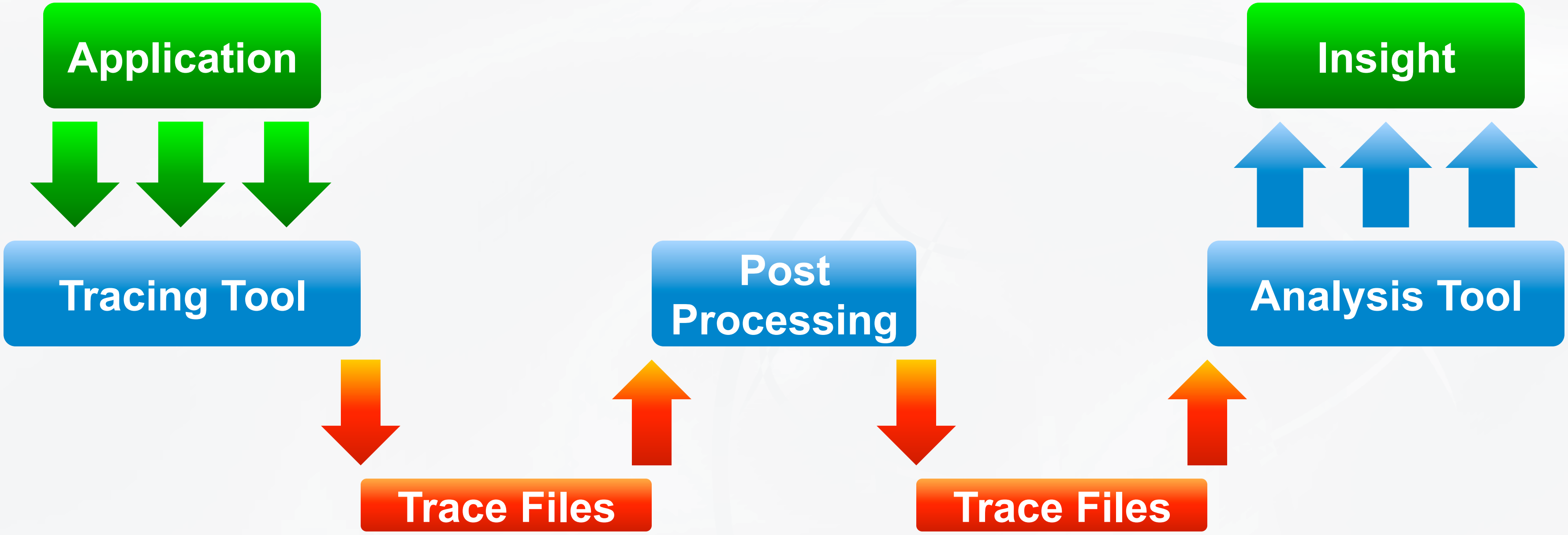
- Event tracing logs runtime events with precise time stamp
- Detailed information but huge generated data volumes
  - → More detail, longer application runs, higher scale
- Frequent non-synchronous buffer flushes to file system
  - → Biases recorded program behavior (communication patterns, load balancing)
- Storage of data on parallel file system
  - → High scale = a lot of files



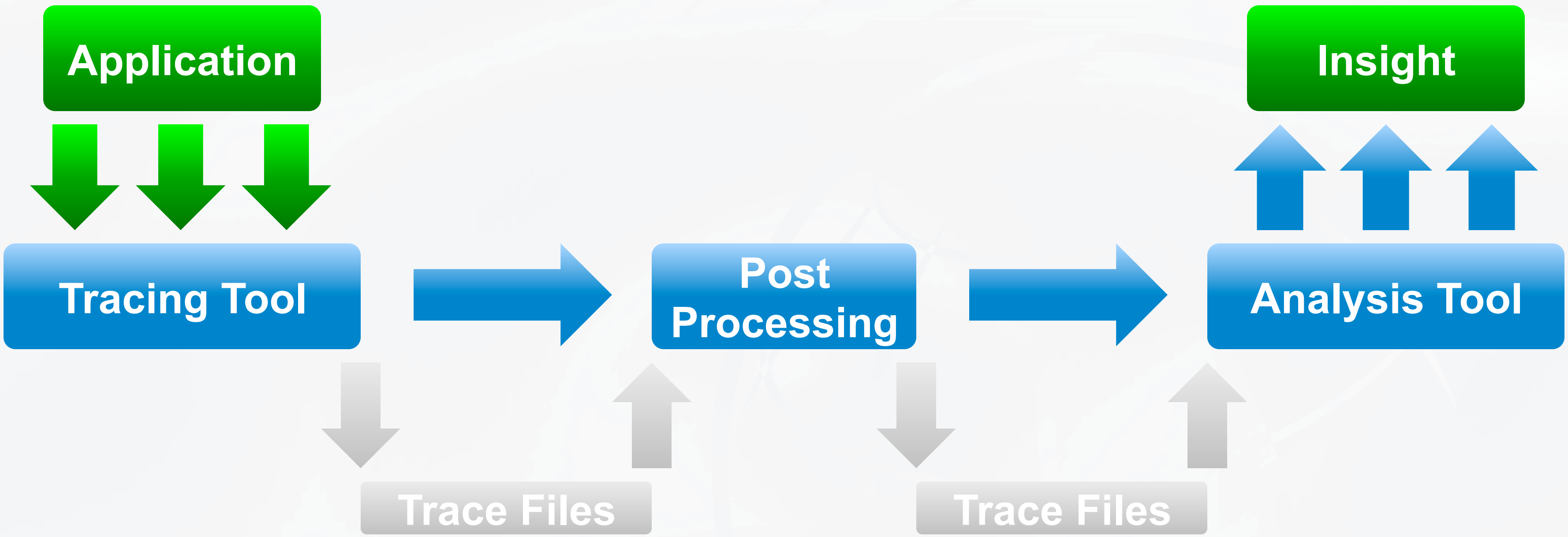
vampir.eu

Challenge: How to deal with millions of files generated by tracing applications on future systems

Motivation: The Typical Workflow

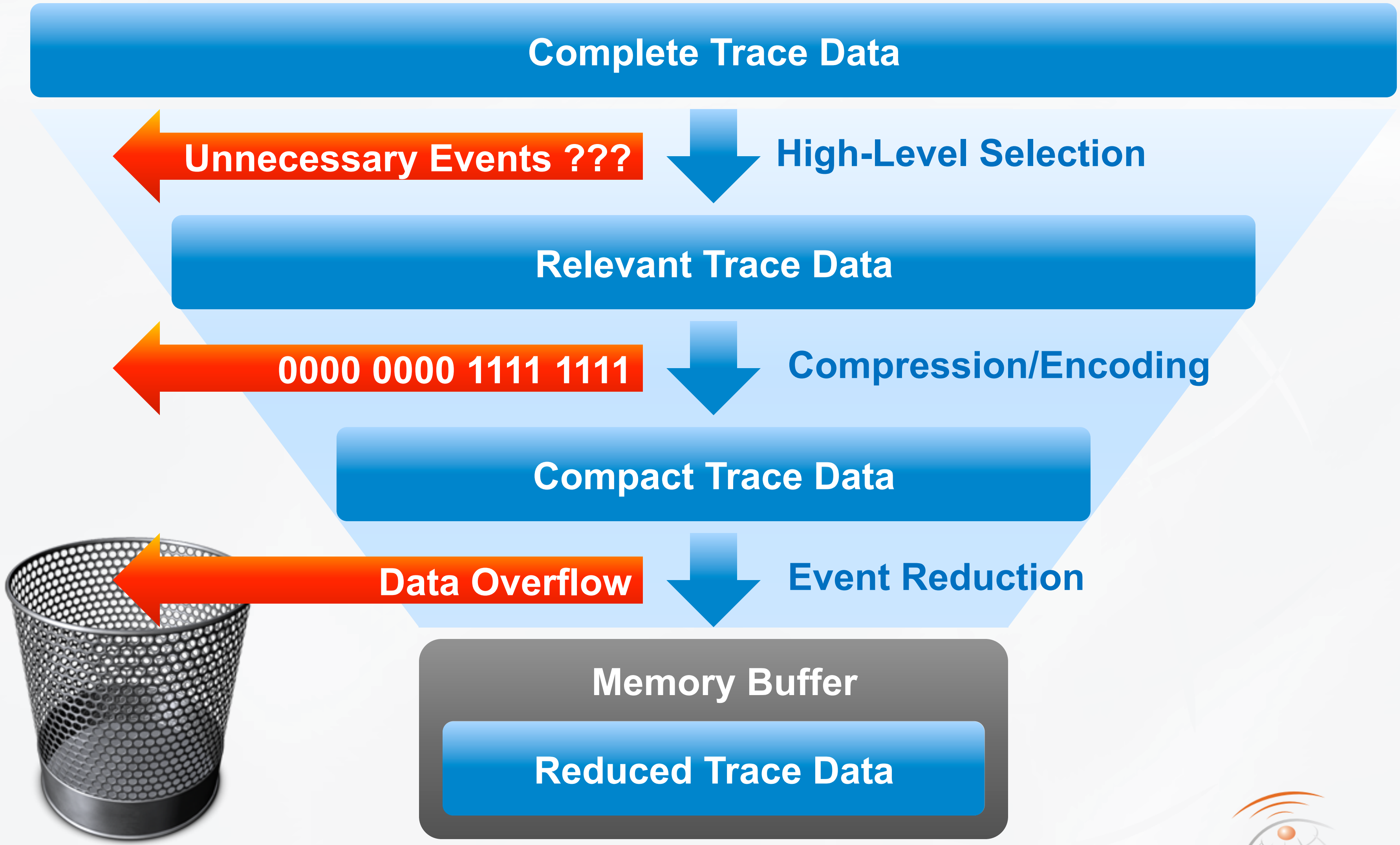


Alternative: In-Memory Event Tracing Workflow



So what‘s the catch?

How to Fit All the Data into a Single Memory Buffer?



Restrictions

Event Tracing Measurement Tool

↓

Event Tracing Storage Library

High level information about events

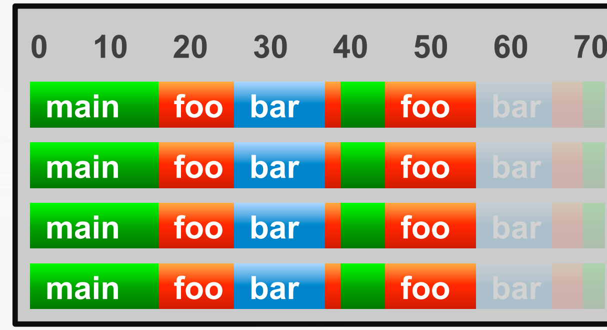
- Call tree
- Other processes
- Context of an event

Low level information about events

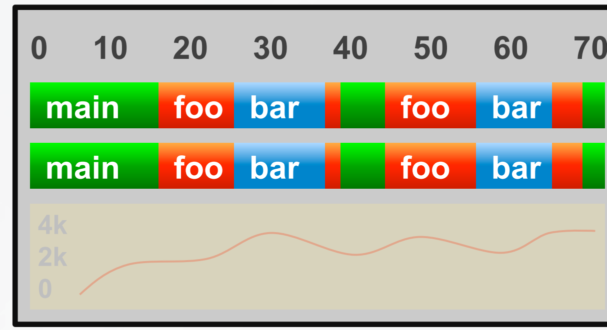
- Time
- Event class

Event Reduction Strategies

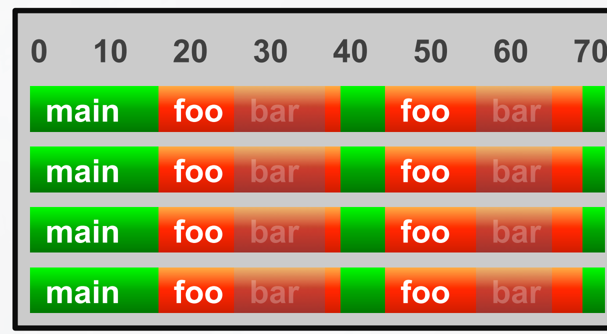
- **Reduction by Order of Occurrence**
  - i.e. stop recording once the memory buffer is exhausted



- **Reduction by Event Class**
  - Order events by event class
  - Discard all events of a specific event class



- **Reduction by Call Stack Level**
  - Order events by call stack level
  - Discard all events of the highest call stack level



Comparison Criteria

Requirements

- Minimal overhead introduced
- Ability to work with limited low level information

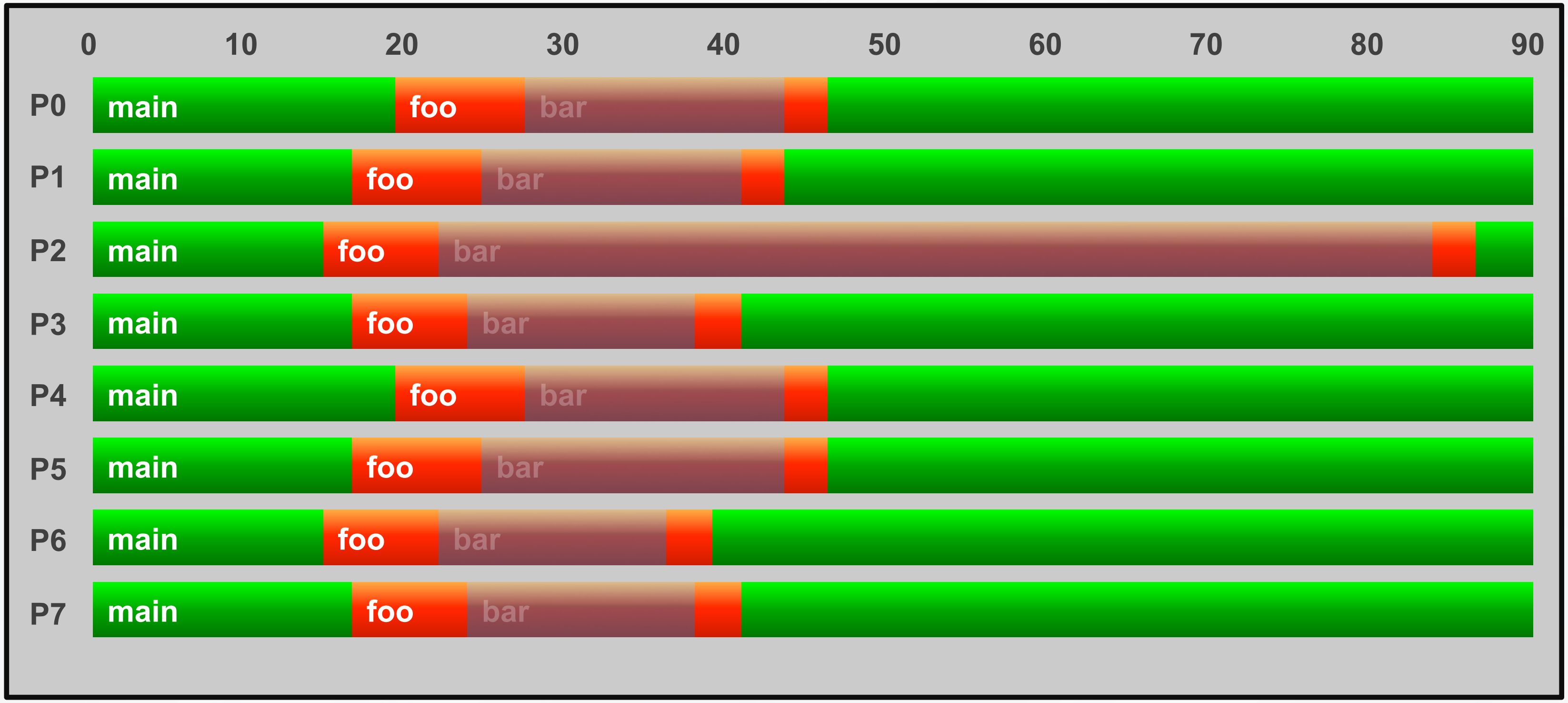
Criteria:

- Quality of remaining information
  - Is it still possible to understand the application behavior?
  - Is it still possible to detect occurring performance problems?
- Size of single reduction steps

Comparison of Reduction Strategies

	Occurrence	Event Class	Call Stack Level
Quality of remaining information	100% for events before buffer exhaustion 0% for events after buffer exhaustion	100% for remaining event classes 0% for discarded event classes	Reduced but not completely lost

Cause and Impact

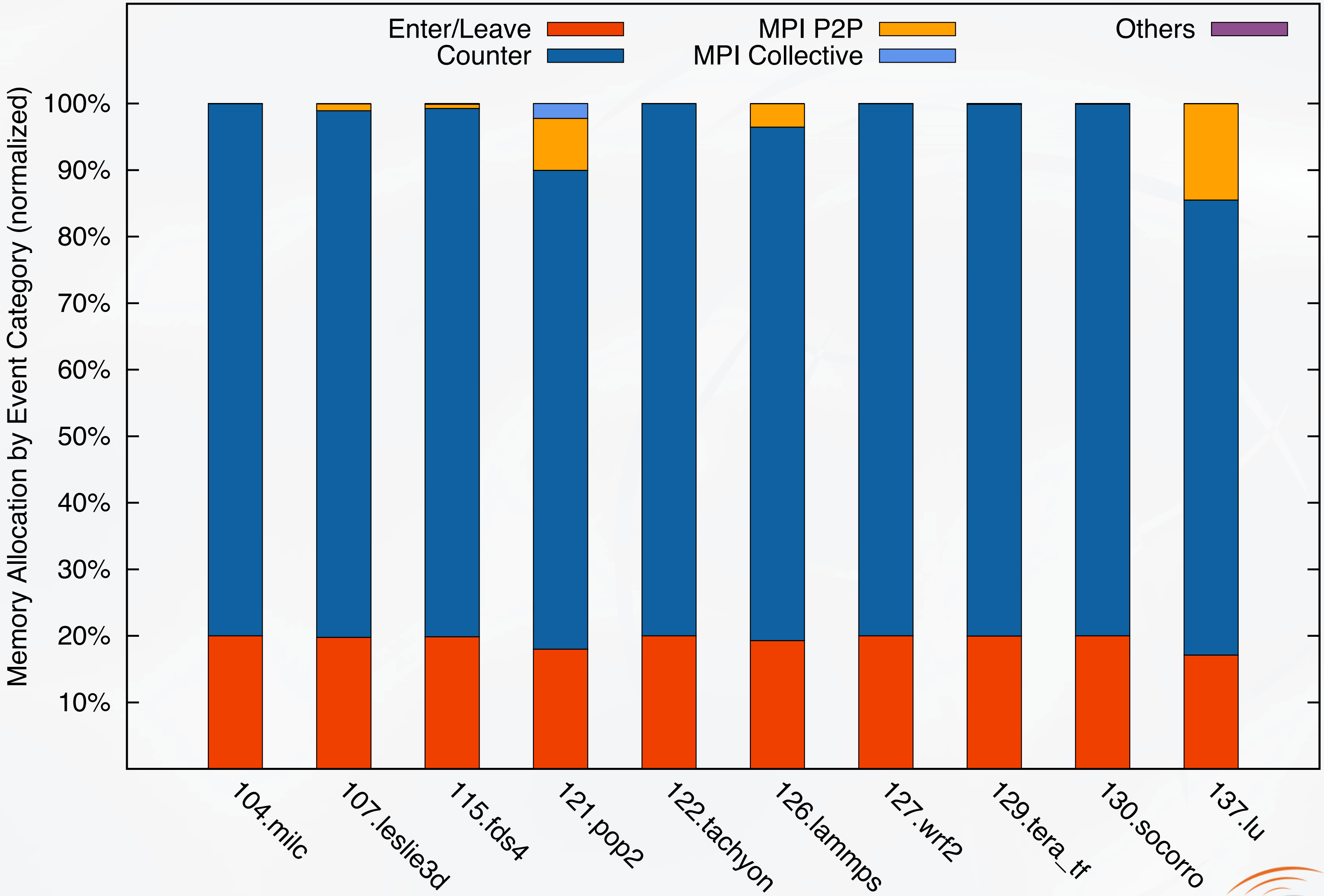


Even if the **cause** of an performance problem (function bar on P2) is lost, the **impact** of the performance problem is still identifiable.

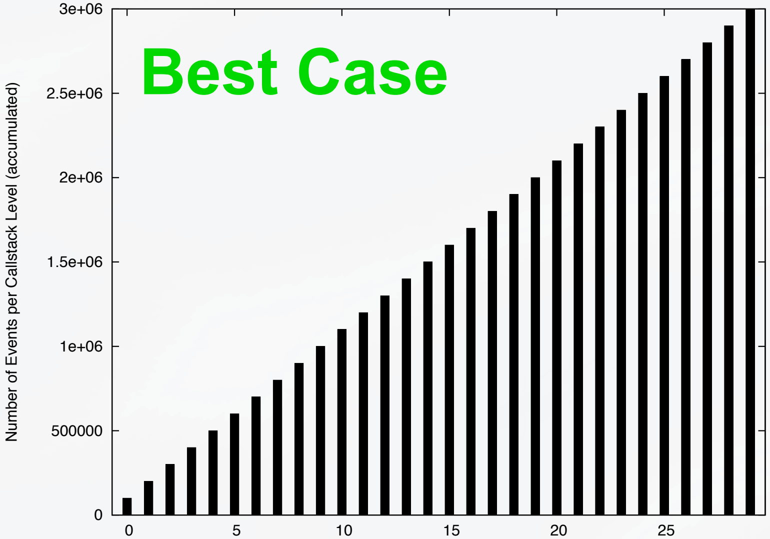
Comparison of Reduction Strategies

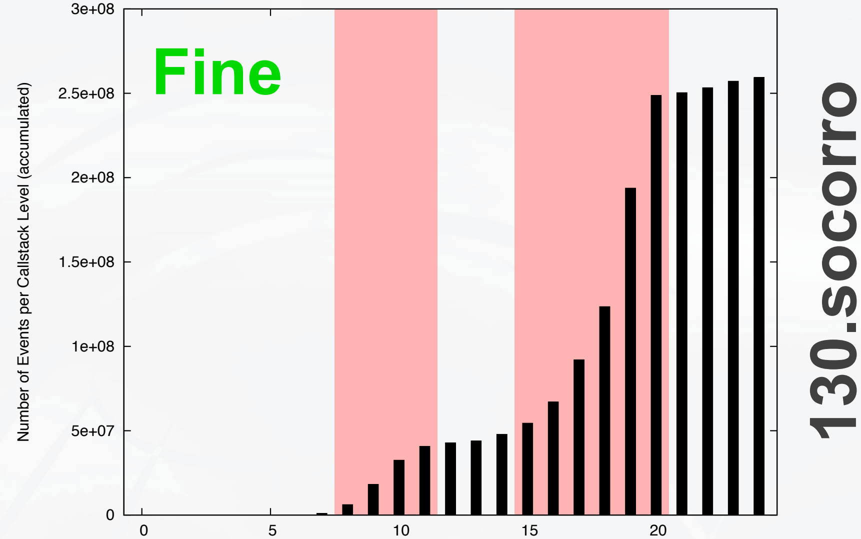
	Occurrence	Event Class	Call Stack Level
Quality of remaining information	100% for events before buffer exhaustion 0% for events after buffer exhaustion	100% for remaining event classes 0% for discarded event classes	Reduced but not completely lost
Size of single reduction steps	Very small (events)	Large (complete event classes)	Depends on number of events per call stack level

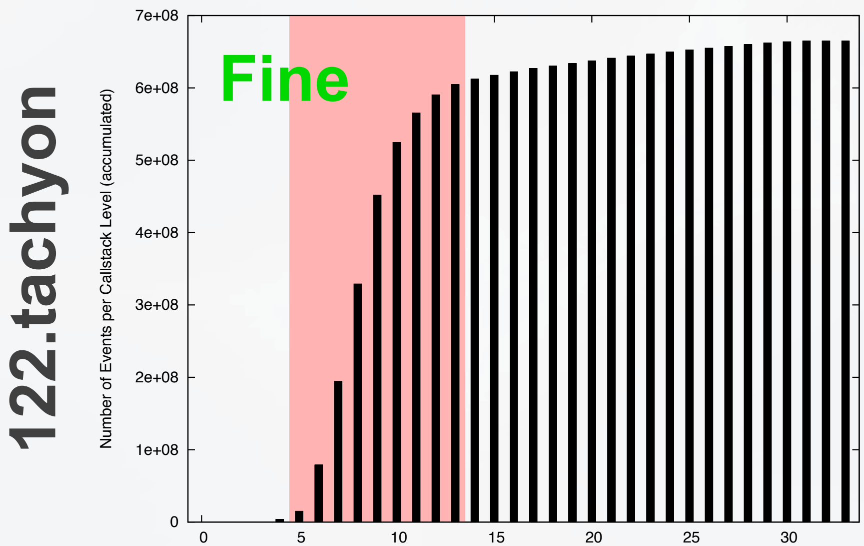
Event Distribution by Event Class (w/o Time Stamps)

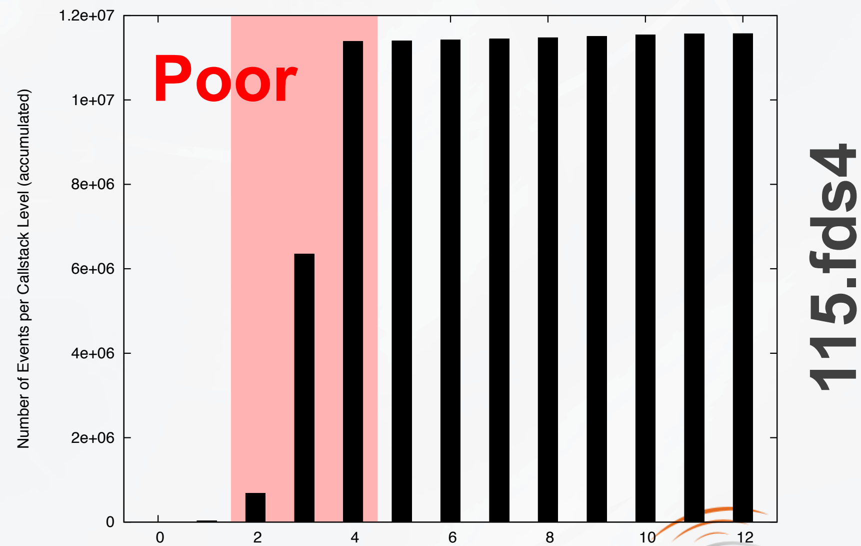


Event Distribution by Call Stack Level (Accumulated)





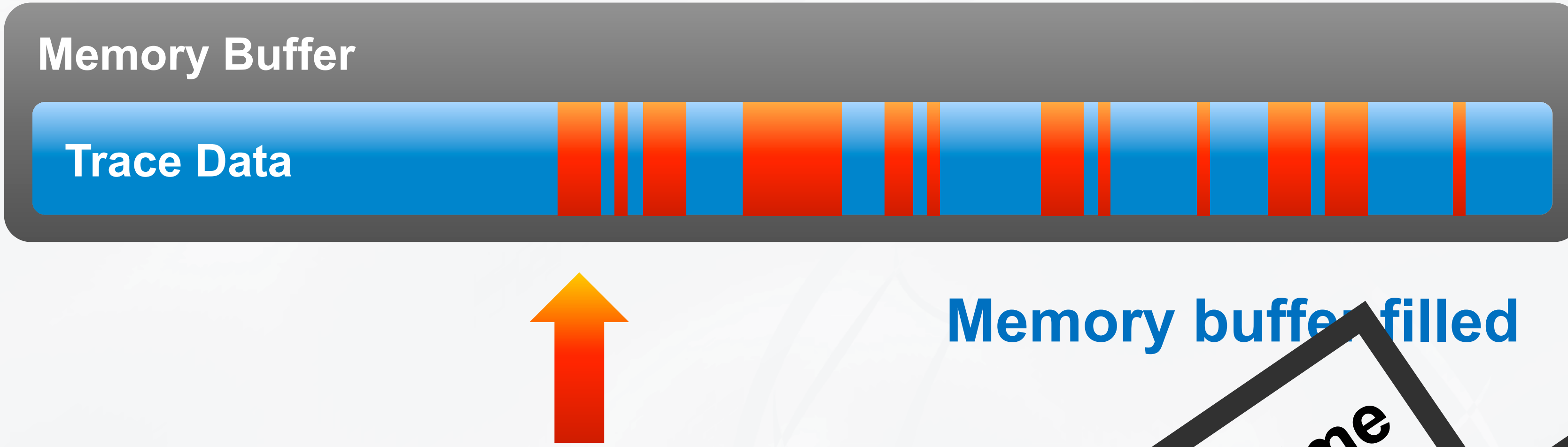




Comparison of Reduction Strategies

	Occurrence	Event Class	Call Stack Level
Quality of remaining information	100% for events before buffer exhaustion 0% for events after buffer exhaustion	100% for remaining event classes 0% for discarded event classes	Reduced but not completely lost
Size of single reduction steps	Very small (events)	Large (complete event classes)	Depends on number of events per call stack level
Quality of remaining information depends on …	Right time phase to record	Appropriate order of event classes by importance	Application structure with regard to call stack level distribution

Real-Time Reduction



Memory buffer filled

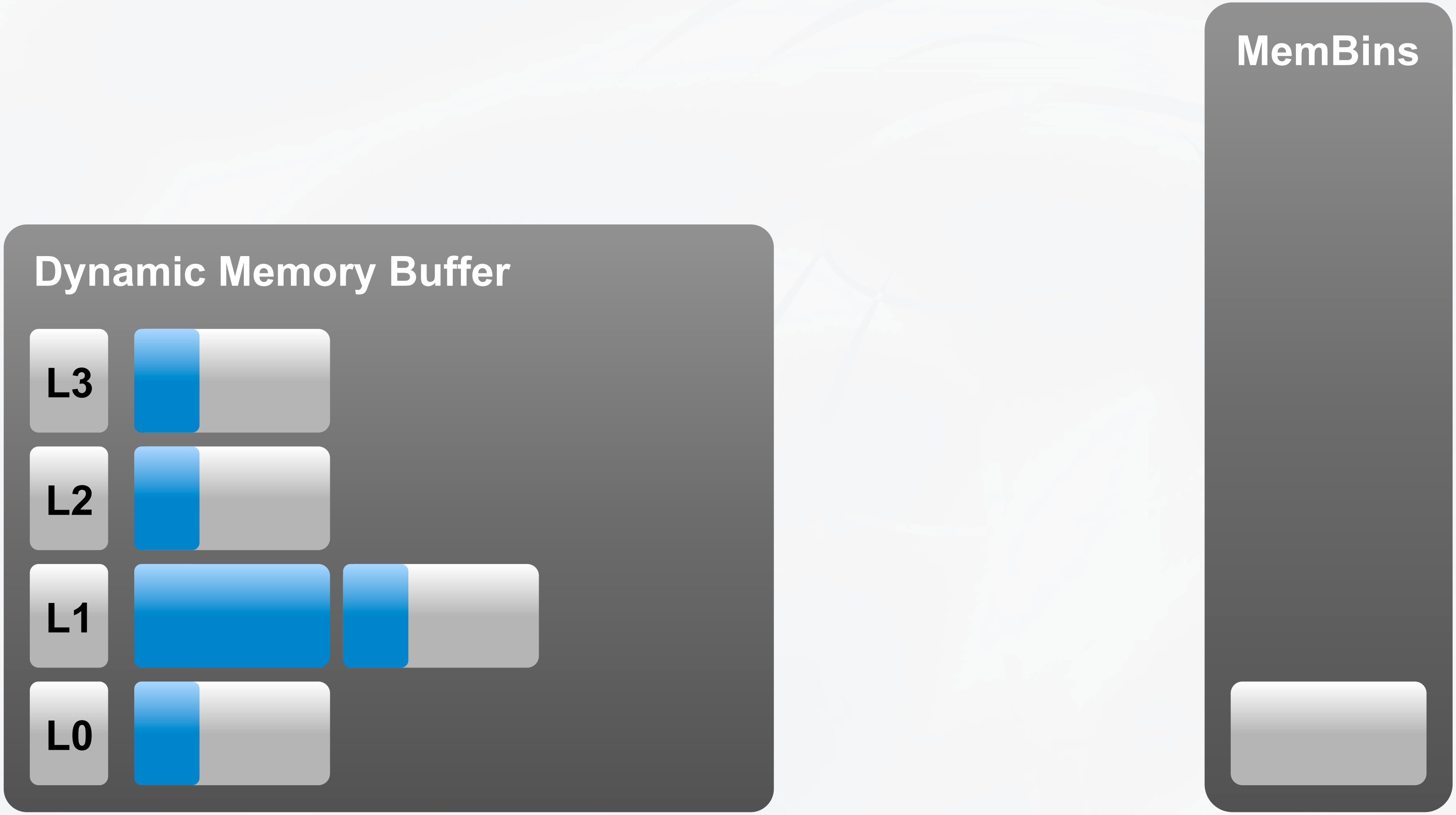
Find memory associated with eliminated events

Not Real-Time

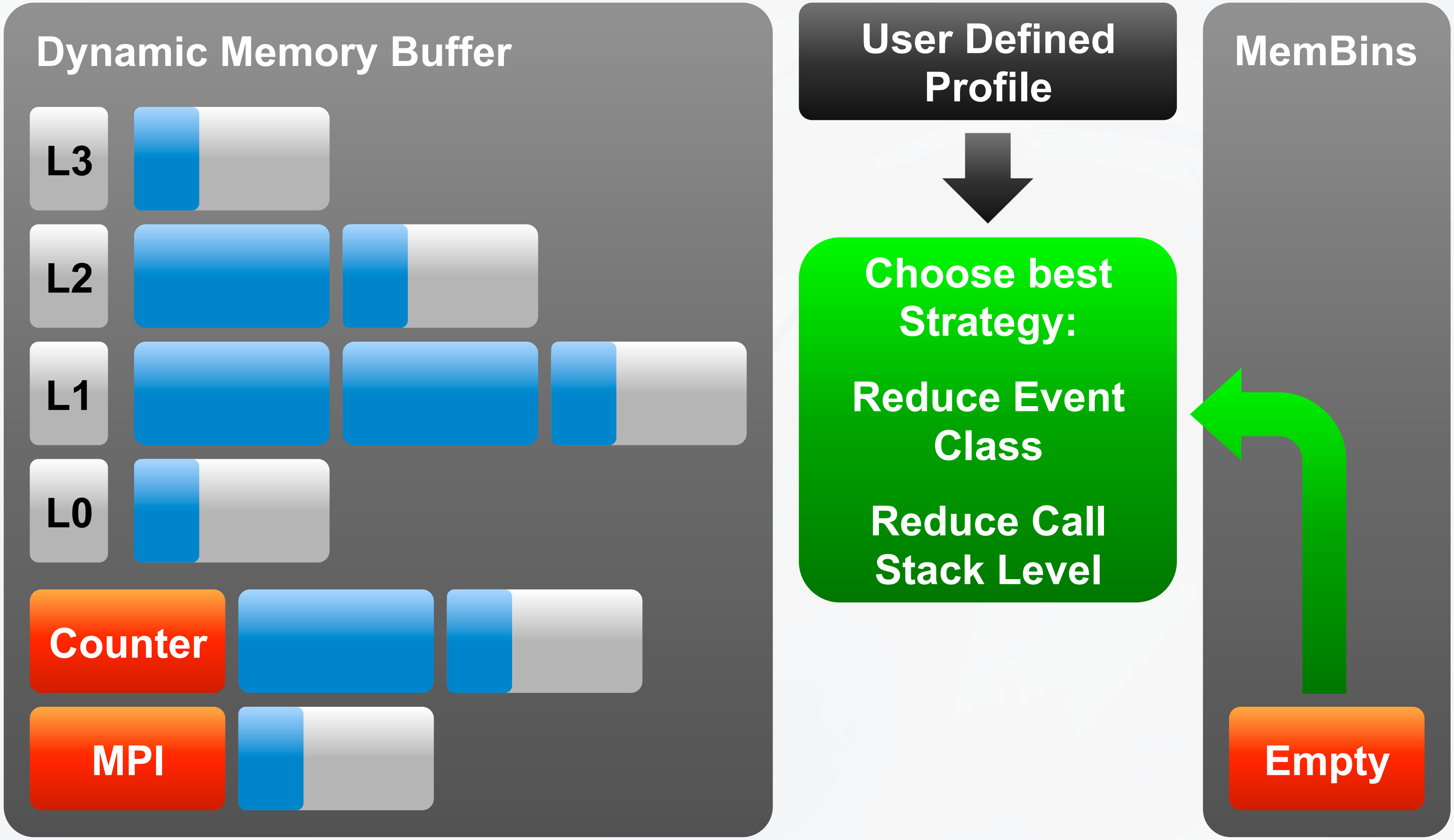
Not Useful

- Costly to find all given events
- Removal of events leaves a highly fragmented memory buffer

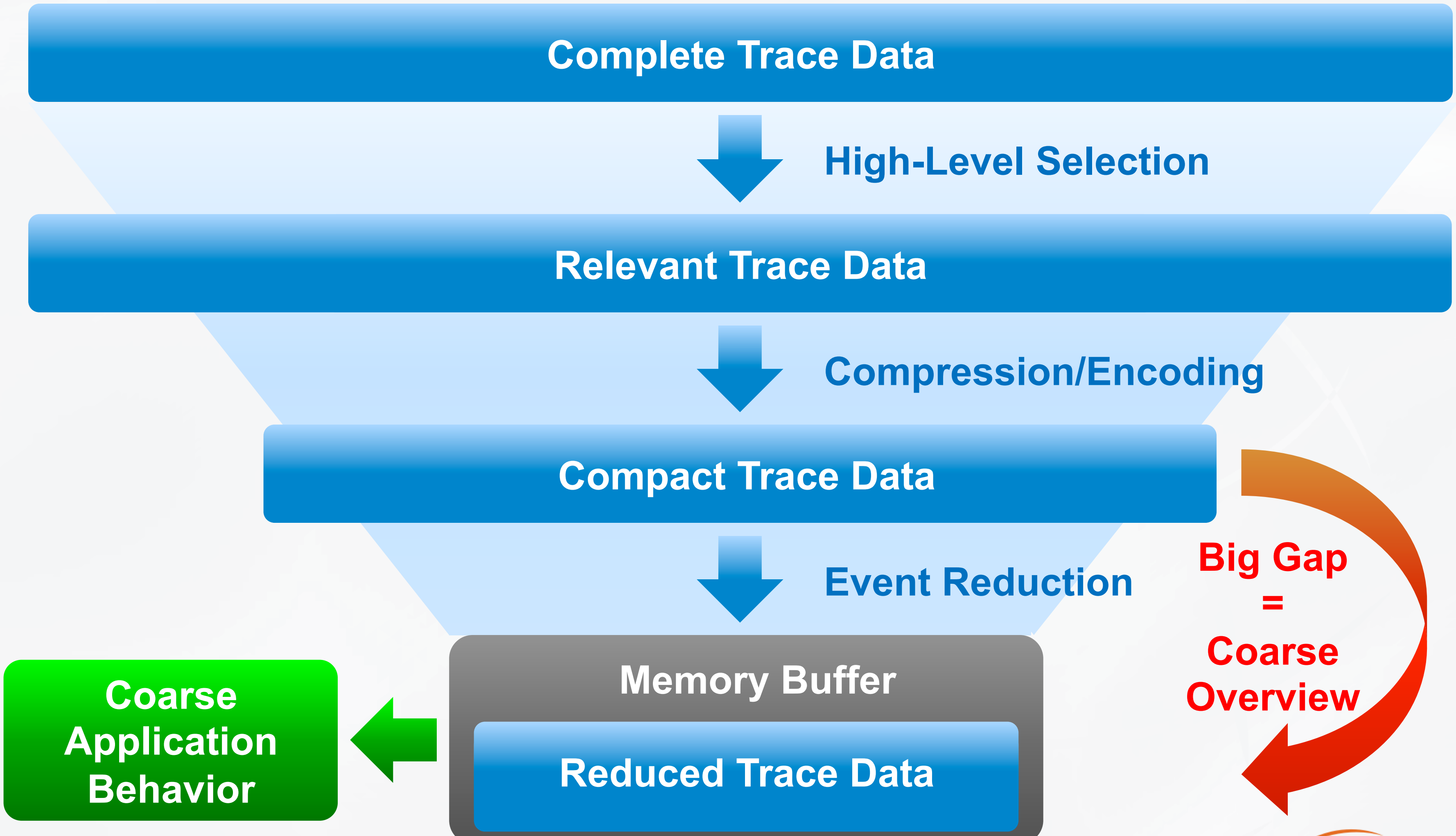
Real-Time Call Stack Reduction



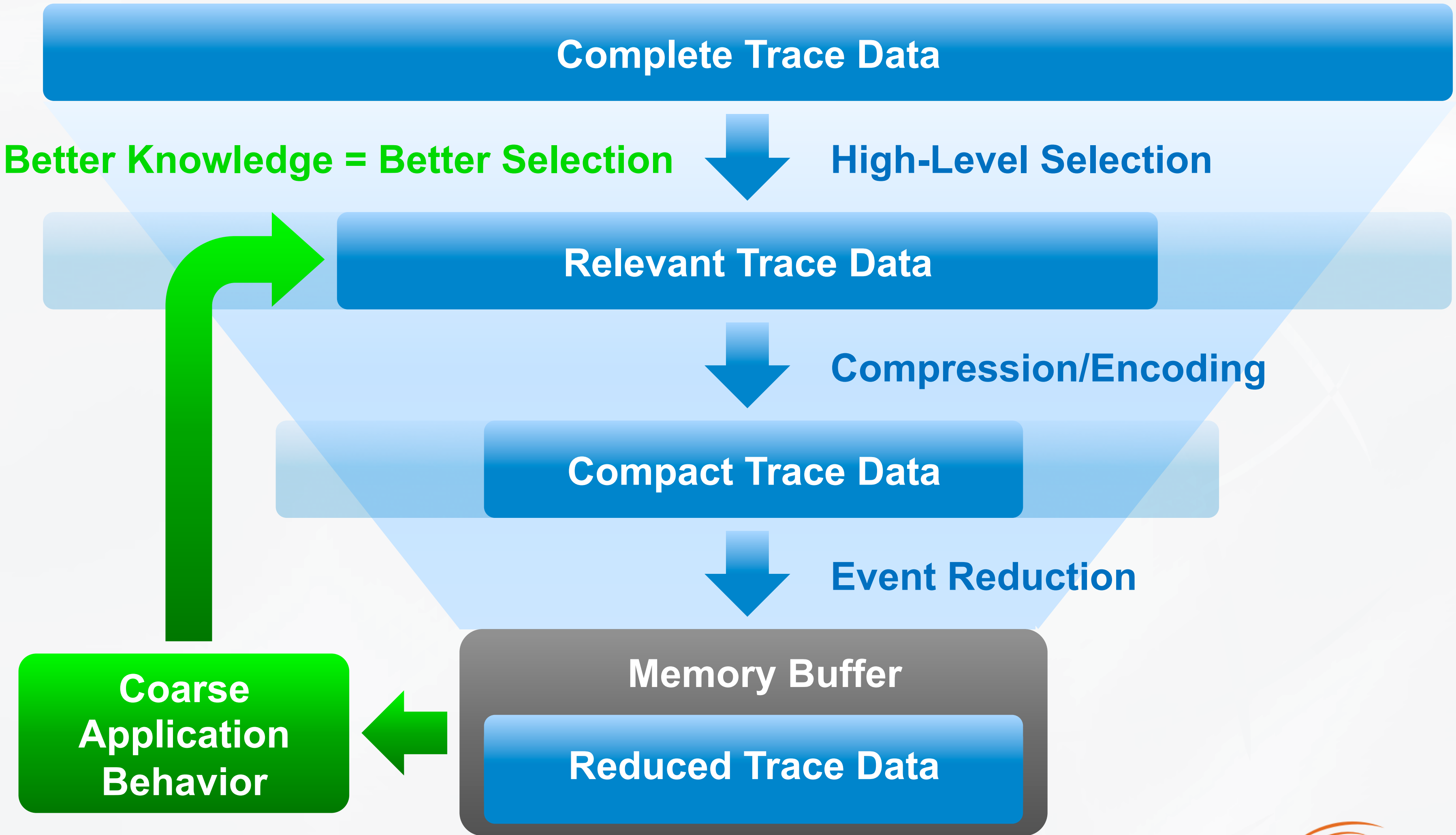
Extended Real-Time Reduction



Reduction Optimization Cycle

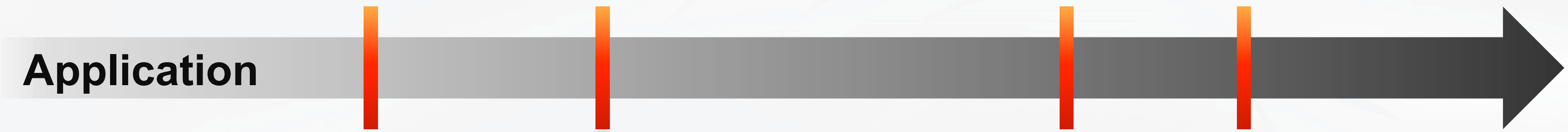


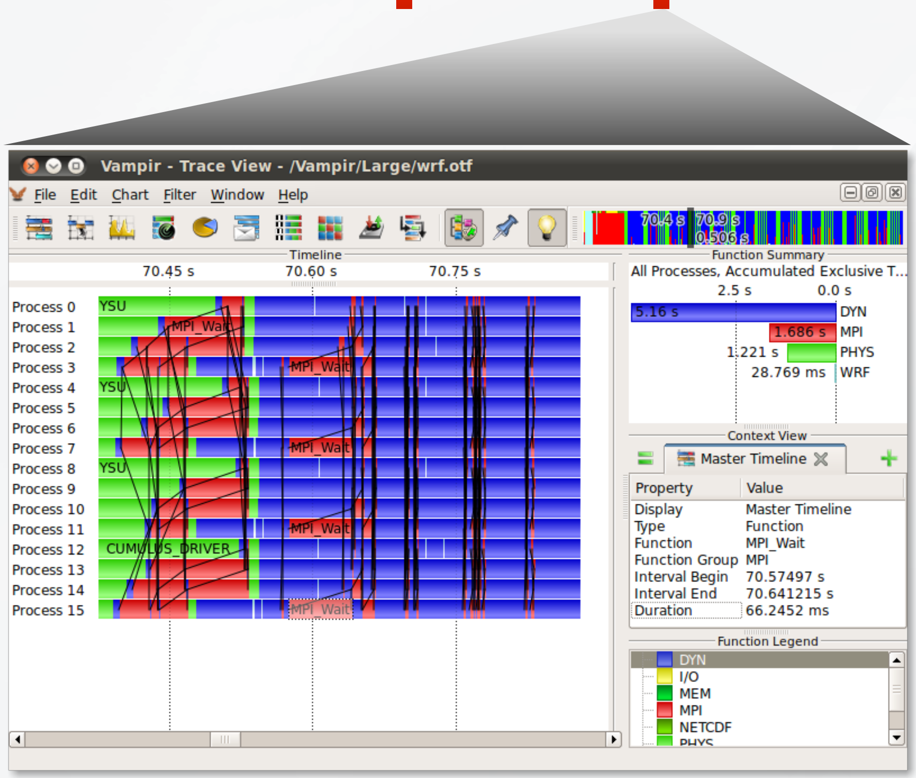
Reduction Optimization Cycle



Future Work: Interactive Online Analysis

Break Points
Global Synchronization or User Defined





Interaction

- Resume
- Stop
- Throw Away
- Add Metric
- Modify Detail
- Remove Iter.

Conclusion

- Strategies for real-time event reduction

  → Guarantee that data of an event tracing measurement fits into a single memory buffer

  → Basic step towards a complete in-memory tracing workflow

  → Enables event trace recording on high scales without limitation of today's parallel file systems

- Defined criteria to compare different strategies and evaluated their benefits
- Enhancements on traditional memory buffering to realize these startegies

Thank You for Your Attention

Questions?