# Tracing and Sampling for Real-Time partially simulated Avionics Systems

Progress Report Meeting
May 14, 2014

Raphaël Beamonte  Michel Dagenais

Distributed Open Reliable Systems Analysis Lab
Computer and Software Engineering Department

# Introduction

- Tracing:
  - Study runtime behavior
  - Can be used to measure latency = fundamental for RT debugging
  - Can now be used on real-time applications
- Analysis:
  - Show unusual latencies
  - Identify the origin of these latencies
  - Need a minimum knowledge about tracing events
- Automatic analysis:
  - That's what we'll see!

# Table of Contents


1. Research Context
   - Real-Time Operating Systems
   - The Linux Trace Toolkit next-generation, LTTng
2. Automatic analysis using traces
   - Why?
   - Literature
3. Conclusion

# Real-Time Operating Systems

Why using the Linux kernel ?

- Able to do Soft Real Time, can reach Hard Real Time :
  - BIOS configuration: would you use hyperthreading ?
  - Kernel configuration: PREEMPT_RT patch, which is more and more integrated to the standard kernel
  - Software configuration: interrupts redirection, cpu shielding...
- The power of the community

# The Linux Trace Toolkit next-generation, LTTng

Why LTTng is pertinent for RT applications ?

- Both userspace and kernel tracers (same clock source)
- Statically compiled tracepoints
- External process to consume events
- Arbitrary event types (Common Trace Format)
- Per-CPU ring buffers
- Important tracing variables protected by RCU

Moreover, LTTng can trace real-time applications without adding more than 7 microseconds latency on Linux.

# Automatic analysis using traces: why?

Why?

- Manually analyzing a trace is not easy for everyone
- The bigger the system is, the more information you'll have, the less you'll see where to look
- Automatic analysis will allow to identify unusual latencies
- Automatic analysis will not pinpoint the exact location of the issue, but gives indications about the probable origin(s)
- Automatic analysis will allow a beginner user to look at these specific places of the trace where the problem could be

# Automatic analysis using traces: why?

How would it work?

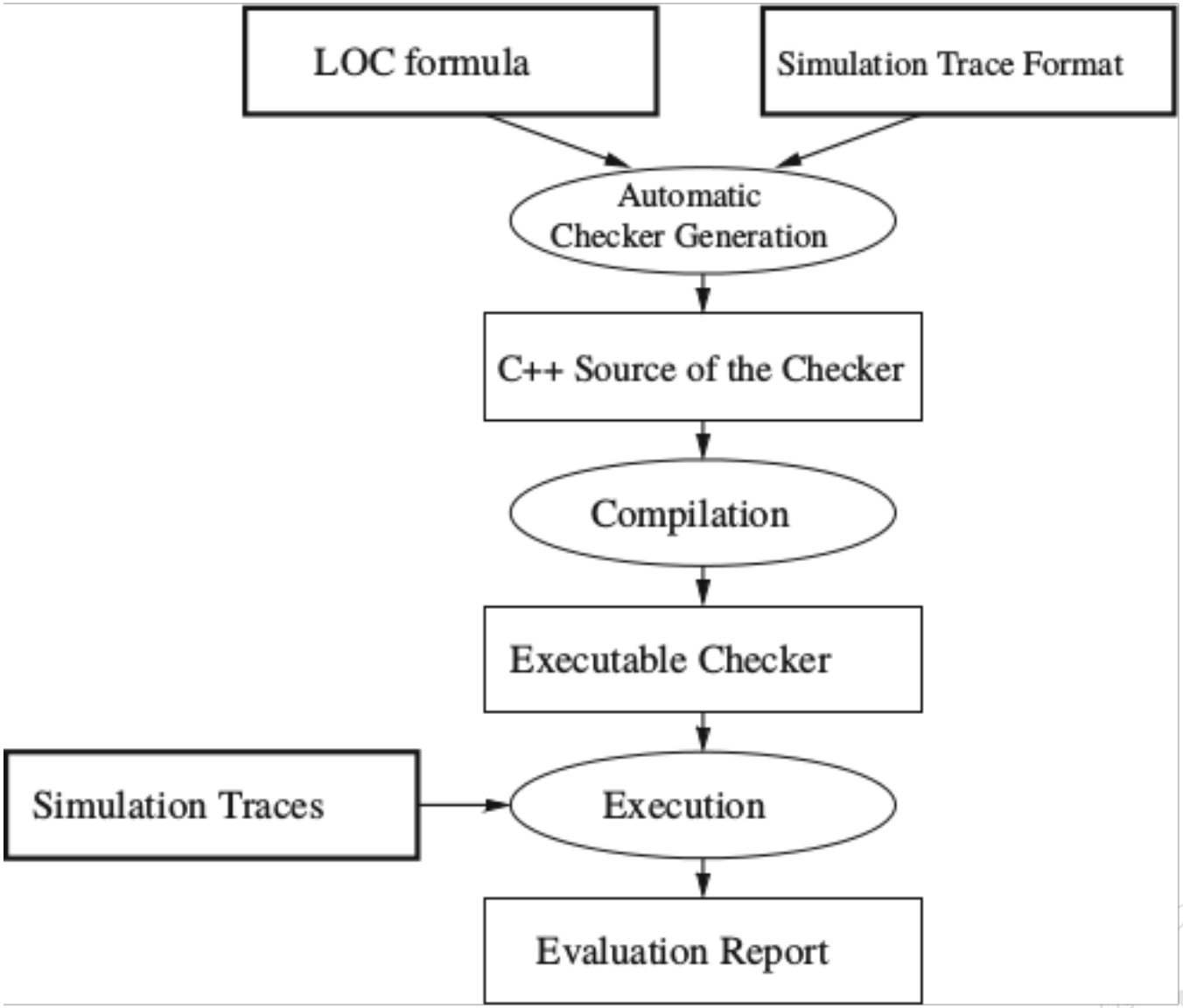
- We know how our application should work: we could give a state machine with real-time constraints
- We could then match this state machine with a userspace trace of our real-time application
- If there is discrepancies between the state machine constraints and the trace: we could use the kernel trace to know what happened differently

# Litterature

Xi Chen et al. "Automatic Trace Analysis for Logic of Constraints". In: *Proceedings of the 40th annual Design Automation Conference*. DAC '03. ACM. Anaheim, CA, USA: ACM, June 2003, pp. 460–465. ISBN: 1-58113-688-9. DOI: 10.1145/775832.775952. URL: http://alumni.cs.ucr.edu/~xichen/pub/dac03_loc.pdf

- Automatic trace analysis used for formal design verification (many articles on the subject!)
- Generate a C++ checker
- Checker can be used on simulation traces and generate evaluation reports

# Litterature



# Literature

Judit Giménez et al. "Guided Performance Analysis Combining Profile and Trace Tools". In: *Euro-Par 2010 Parallel Processing Workshops*. Ed. by MarioR. Guarracino et al. Vol. 6586. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 513–521. ISBN: 978-3-642-21877-4. DOI: 10.1007/978-3-642-21878-1_63. URL: http://dx.doi.org/10.1007/978-3-642-21878-1_63

- Use of profile-based tools to aggregate statistics
- Correlation between statistics and timeline
- Using KOJAK to do automatic pattern search in event traces to identify the sources of wait states in parallel applications

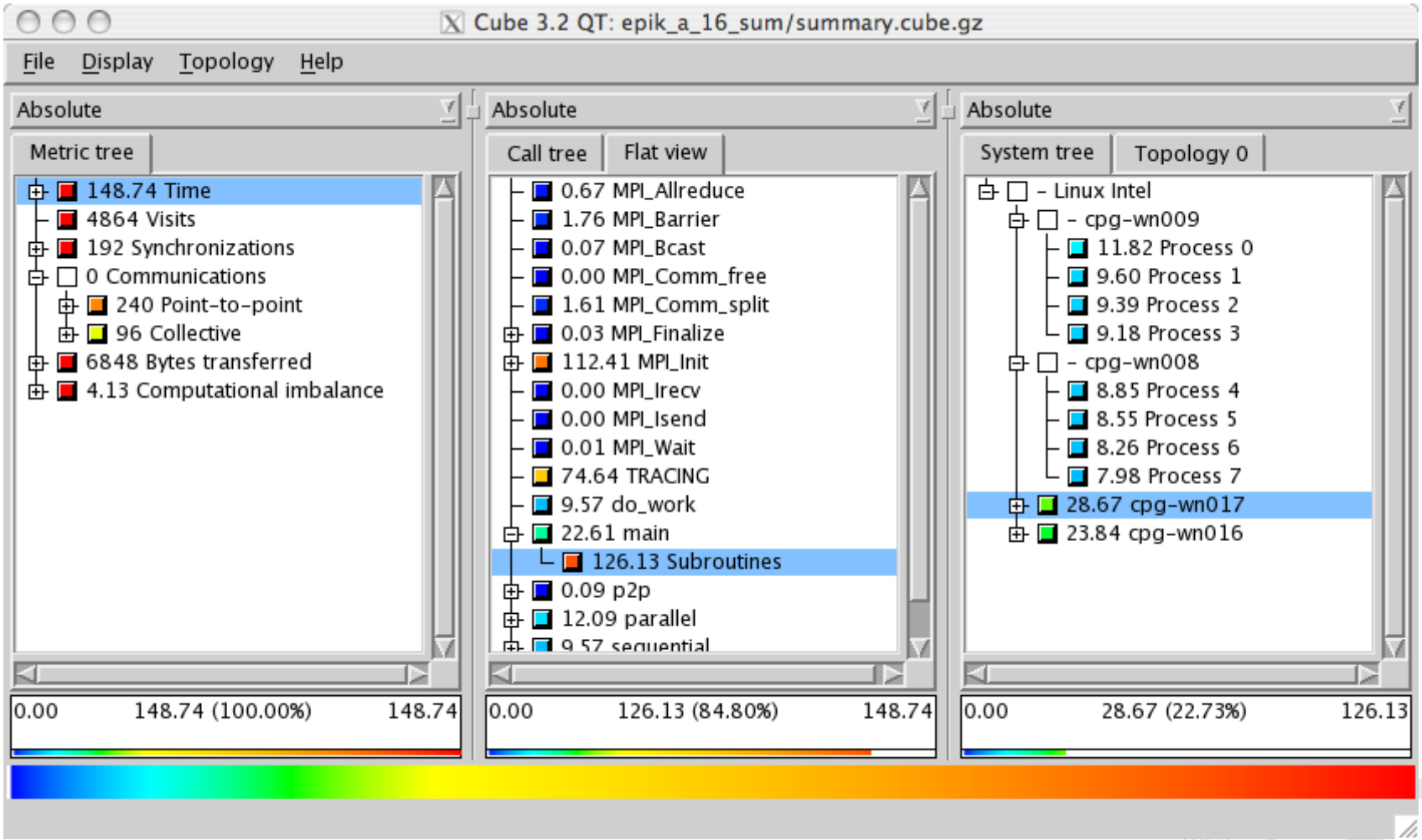
# Literature

Bernd Mohr. "Automatic Trace Analysis with Scalasca". In: Jülich Supercomputing Center. 2012. URL: http://www.vi-hps.org/upload/projects/hopsa/hopsa-nov12-scalasca.pdf

- Presentation "follow-up" of the previous article
- Scalasca project successor of KOJAK
- Aims to use execution traces to simplify the identification of bottlenecks
- Scalasca is faster than KOJAK

# Literature

CUBE3 window used to analyze Scalasca results (source: wiki.grid.auth.gr)



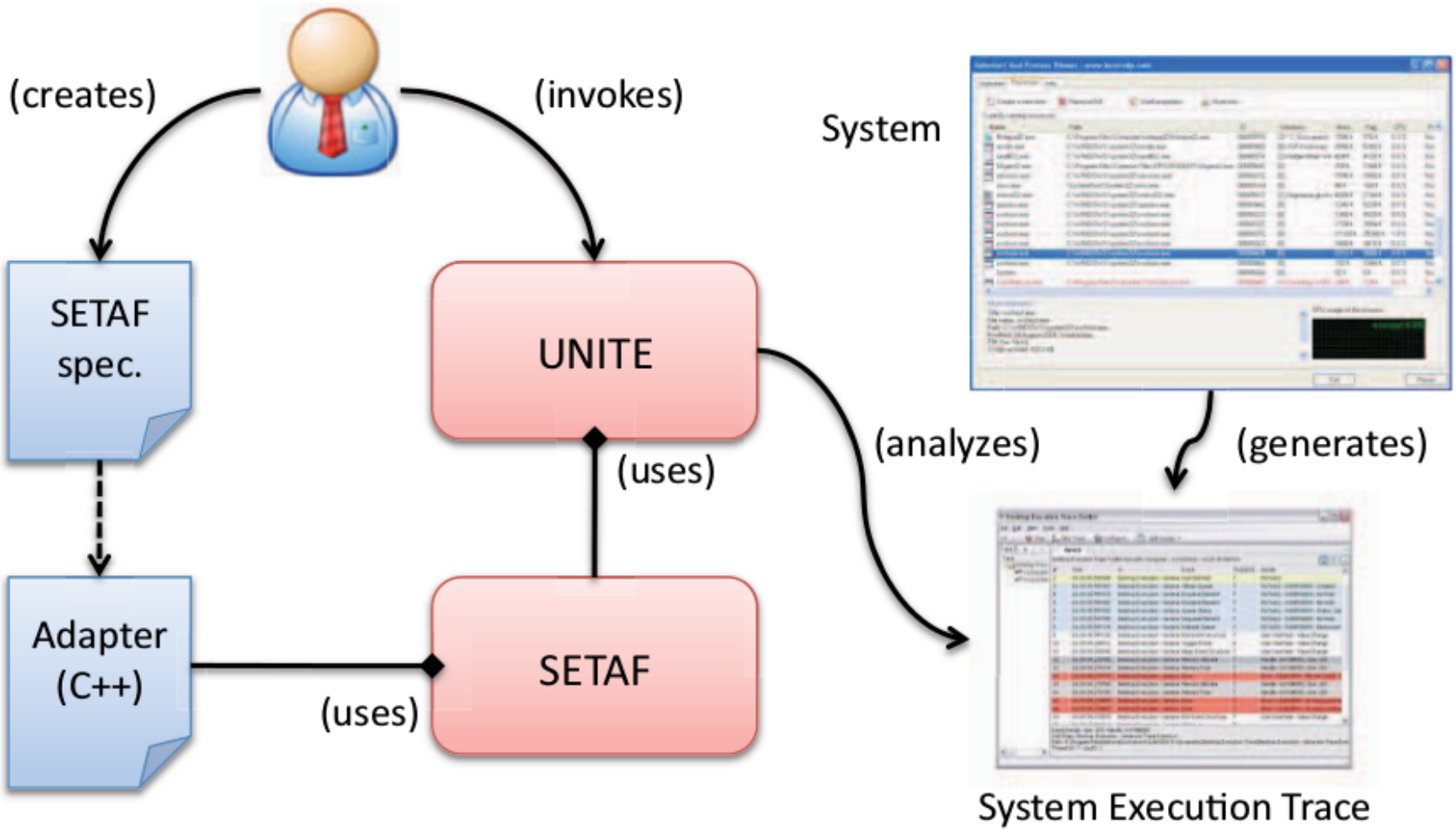
# Literature

T Manjula Peiris and James H Hill. "Adapting System Execution Traces for Validation of Distributed System QoS Properties". In: *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2012 IEEE 15th International Symposium on*. IEEE. 2012, pp. 162–171

- Validating distributed systems QoS using system execution traces
- Presentation of the System Execution Trace Adaptation Framework (SETAF)
- Using timed events to know if QoS is respected
- Auto-generated C++ adapter to wrap the original trace before using the Understanding Non-functional Intensions via Testing and Experimentation (UNITE) tool

# Literature

Conceptual overview of SETAF's workflow



# Conclusion

- Automatic trace analysis can simplify performance analysis for complicated systems
- LTTng traces give many informations about the system state, much more than common traces used in the literature
- Statistics seems to be a relevant way to identify inconsistencies
- Pattern identification and pattern verification are also used
- Current work: correlating LTTng traces and a state machine to generate statistics and make assumptions about probable errors leading to inconsistencies

# Tracing and Sampling for Real-Time Systems Video

Pedagogical video explaining the context, problem and solution to trace real-time systems using LTTng.

http://youtu.be/cBaTDz6mi98

# Thank you. Any question ?

LTTng `www.lttng.org`
**mailing list:** `lttng-dev@lttng.org`

npt: `git.dorsal.polymtl.ca/?p=npt.git`

Slides: `www.dorsal.polymtl.ca/~rbeamonte/dorsal-pm-may2014.pdf`