

Did you behave in real-time?

A probabilistic evaluation with trace data

Benno Bielmeier¹, Wolfgang Mauerer^{1,2}, Ralf Ramsauer¹

LPC 2024 / Real-Time MC
September 19, 2024

¹Technical University of Applied Sciences Regensburg ²Siemens AG, Technology

Timing Analysis (TA)

► Static TA

```

/ arch / riscv / sb / memmove5
All symb Search Identifier

174
175 /*
176  * Fix Misalignment Copy Loop - Reverse
177  * load_val1 = load_ptr[0];
178  * do {
179  *     [load_val0 = load_ptr[-1];
180  *     store_ptr -= 2;
181  *     store_ptr[1] = (load_val0 >> {a6}) | (load_val1 << {a7});
182  *
183  *     if (store_ptr == {a2})
184  *         break;
185  *
186  *     load_val1 = load_ptr[-2];
187  *     load_ptr -= 2;
188  *     store_ptr[0] = (load_val1 >> {a6}) | (load_val0 << {a7});
189  *
190  * } while (store_ptr != store_ptr_end);
191  * store_ptr = store_ptr_end;
192  */
193
194 REG_L t1, { 0 * SZREG}(a4)
195 13
196 REG_L t0, (-1 * SZREG)(a4)
197 addi t4, t4, (-2 * SZREG)
198 sll t1, t1, a7
199 srl t2, t0, a6
200 or t2, t1, t2
201 REG_S t2, { 1 * SZREG}(t4)
202
203 beq t4, a2, 2f
204
205 REG_L t1, (-2 * SZREG)(a4)
206 addi a4, a4, (-2 * SZREG)
207 sll t0, t0, a7
208 srl t2, t1, a6
209 or t2, t0, t2
210 REG_S t2, { 0 * SZREG}(t4)
211
212 bne t4, t5, 1b
213 25
214 HW t4, t5 /* Fix the dest pointer in case the loop was broken */
215
216 add a4, t4, a5 /* Restore the src pointer */
217 ] .Lbyte_copy_reverse /* Copy any remaining bytes */
218
219 /*
220  * Simple copy loops for SZREG co-aligned memory locations.
221  * These also make calls to do byte copies for any unaligned
222  * data at their terminations.
223  */
224 .Lcoaligned_copy:
225 bltu a1, a0, .Lcoaligned_copy_reverse
226
227 .Lcoaligned_copy_forward:
228 jal t0, .Lbyte_copy_until_aligned_forward
229
230 33
231 REG_L t1, { 0 * SZREG}(a1)
232 addi a1, a1, SZREG
233
linux % v6.11 Download file powered by Elsie 2.2

```

Timing Analysis (TA)

- ▶ Static TA
- ▶ Measurement-Based TA



Image source: osadl.org

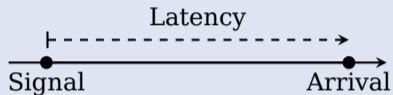
Timing Analysis (TA)

- ▶ Static TA
- ▶ Measurement-Based TA
- ▶ Measurement-Based Probabilistic TA

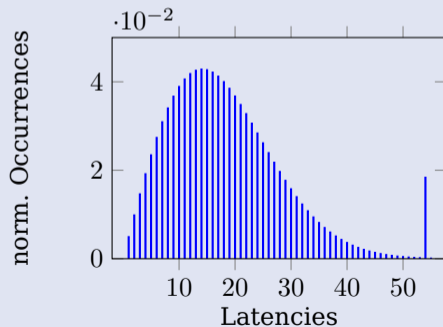
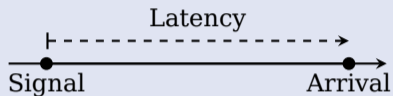
Quality vs. Quantity

Confidence vs. Measurement Period

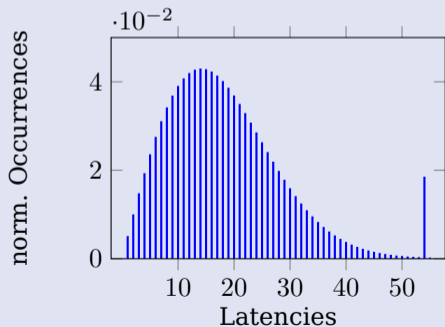
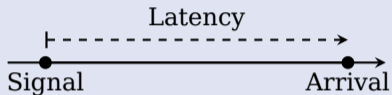
Measurement Approach



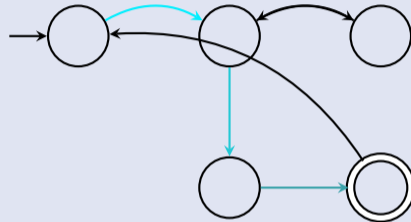
Measurement Approach



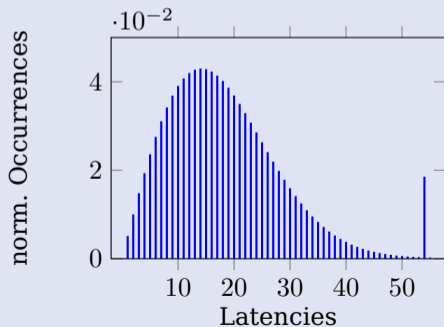
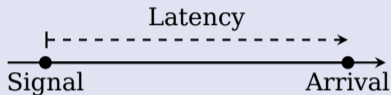
Measurement Approach



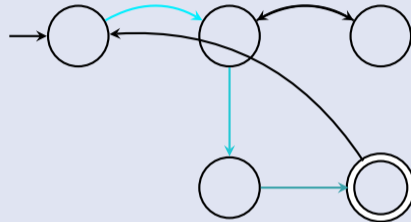
Hybrid Approach



Measurement Approach



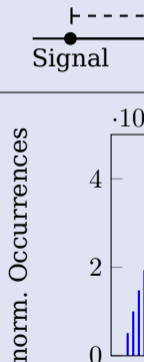
Hybrid Approach



$$SMM = (S, P, F_{ij}(t), \pi)$$

$$\mathbb{E}[F(i, j_f)] = \mathbb{E}[T_i] + \sum_{k \neq j_f} P_{ik} \mathbb{E}[F(k, j_f)]$$

Measurement



Modeling the Behavior of Threads in the PREEMPT_RT Linux Kernel Using Automata

Daniel Bristot de Oliveira
Red Hat, Inc.
Scuola Superiore Sant'Anna
Universidade Federal de Santa
Catarina
bristol@redhat.com

Tommaso Cucinotta
Scuola Superiore Sant'Anna
tommaso.cucinotta@santannapisa.it

Rômulo Silva de Oliveira
Universidade Federal de Santa
Catarina
romulo.deoliveira@ufsc.br

ABSTRACT

This article proposes an automata-based model for describing and verifying the behavior of thread management code in the Linux PREEMPT_RT kernel, on a single-core system. The automata model defines the events that influence the timing behavior of the execution of threads, and the relations among them. This article also presents the extension of the Linux trace features that enable the trace of such events in a real system. Finally, one example is presented of how the presented model and tracing tool helped catching an inefficiency bug in the scheduler code and ultimately led to improving the kernel.

CCS CONCEPTS

•Computer systems organization → Real-time operating systems; Embedded systems

KEY WORDS

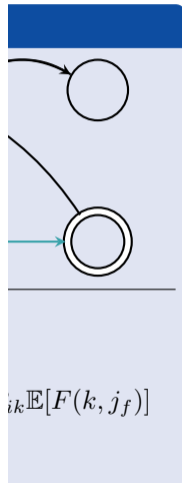
Real-time systems, Linux kernel, behavioral modeling, code verification, automata, tracing.

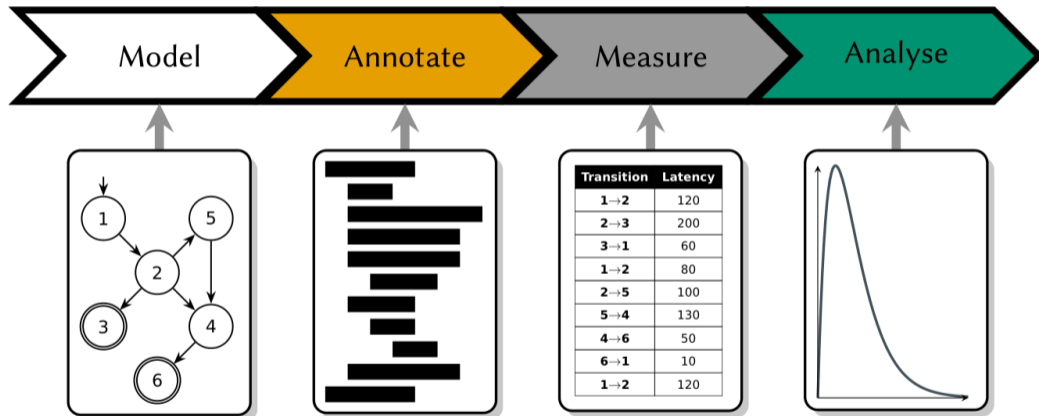
1 INTRODUCTION

Real-time Linux has been a research topic for more than a decade now, with many scheduling algorithms being implemented in Linux [3, 7, 9, 9]. Despite the constant iteration of Linux development

Then, interrupts must be disabled as well, to avoid race conditions with interrupt handlers. Hence, delays in the scheduling and interrupt handler are created during activation of a thread [12].

The understanding of such operations, and how they affect the timing behavior of a task, are fundamental for the development of real-time algorithms for Linux. However, the amount of effort required for a researcher to understand all these constraints is not negligible. Rather, it might take years for a newcomer to understand the internals of the Linux kernel. The complexity of Linux is indeed a barrier, not only for researchers but for developers as well. Inside the kernel, scheduling operations interact with low-level details of the underlying processor and memory architectures, where complex locking protocols and “hacks” are used. This is done to ensure that such a general-purpose operating system (GPOS) behaves as efficiently as possible in the average case, while at the same time it is done to ensure that, with proper tuning, the kernel can serve an increasing number of real-time use cases as well, turning effectively into a real-time operating system (RTOS). The progressive chasing and elimination over the years of any use of the old global kernel lock, the extensive use of fine-grain locking, the widespread adoption of memory barrier primitives, or even the post-ponement of most interrupt handling code to kernel threads as done in the PREEMPT_RT kernel, are all examples of a big commitment into reducing the duration of non-preemptible kernel sections to the bare minimum, while allowing for a greater control over the priority and scheduling among the various internal activities, with various benefits for the system administrator





Measurement-based Timing Analysis

Assumptions and Requirements

- ▶ System/Domain expertise required
- ▶ Payload-specific
- ▶ Precise timing information of the events

Limitations

- ▶ Measurements influence system behaviour
- ▶ Extreme outliers may remain unseen

Measurement-based Probabilistic Timing Analysis

Assumptions and Requirements

- ▶ System/Domain expertise required
- ▶ Payload-specific
- ▶ Precise timing information of the events

Limitations

- ▶ Measurements influence system behaviour
- ▶ Extreme outliers may remain unseen

Measurement-based Probabilistic Timing Analysis

Assumptions and Requirements

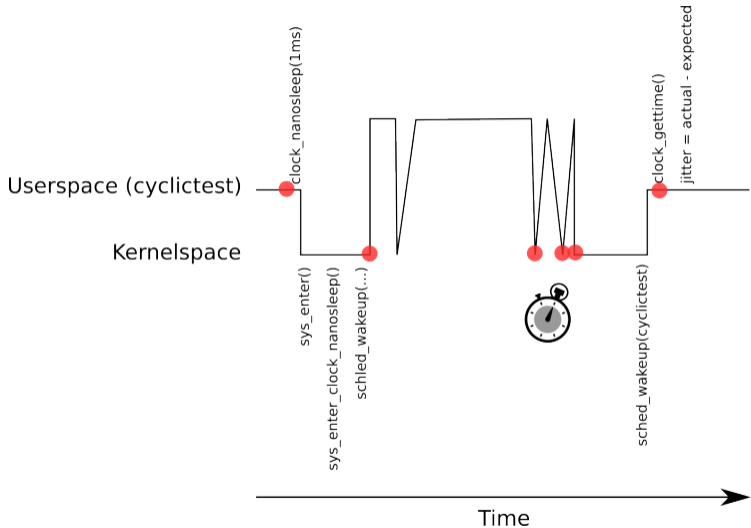
- ▶ System/Domain expertise required
- ▶ Payload-specific
- ▶ Precise timing information of the events

Limitations

- ▶ Measurements influence system behaviour
- ▶ Extreme outliers may remain unseen

Opportunities

- ▶ Introspection
- ▶ Reduced measurement duration
- ▶ Stochastic "extrapolation" of tail latencies



System Setup

System

- ▶ Kernel v6.11-rc5-rt5
- ▶ StarFive VisionFive 2
 - ▶ StarFive JH7110 64bit SoC with RV64GC (up to 1.5GHz)
 - ▶ LPDDR4 8 GB
 - ▶ 4 cores, 1-3 isolated

Steps

- ▶ Board Enabling
- ▶ RT Tuning
- ▶ Implement POC of "Temporal Tracepoints"
- ▶ Patch Kernel and cyclictest
- ▶ Run measurement
- ▶ Fit model
- ▶ Derive Predictions

System Setup

System

- ▶ Kernel v6.11-rc5-rt5
- ▶ StarFive VisionFive 2
 - ▶ StarFive JH7110 64bit SoC with RV64GC (up to 1.5GHz)
 - ▶ LPDDR4 8 GB
 - ▶ 4 cores, 1-3 isolated

Steps

- ▶ Board Enabling
- ▶ RT Tuning
- ▶ Implement POC of "Temporal Tracepoints"
- ▶ Patch Kernel and cyclictest
- ▶ Run measurement
- ▶ Fit model
- ▶ Derive Predictions

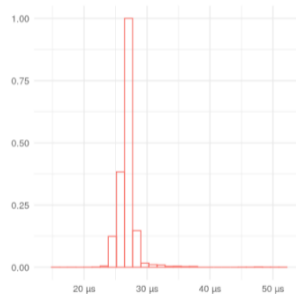
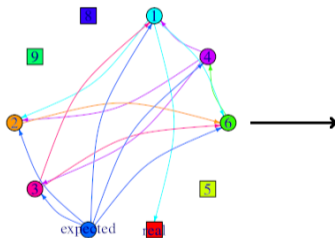
```
+ riscv_clock_event_stop();  
  ttp_emit(2);
```

Transition	Latency
1→2	120
2→3	200
3→1	60
1→2	80
2→5	100
5→4	130
4→6	50
6→1	10
1→2	120

cyclictest

Transition	Latency
1→2	120
2→3	200
3→1	60
1→2	80
2→5	100
5→4	130
4→6	50
6→1	10
1→2	120

Kernel

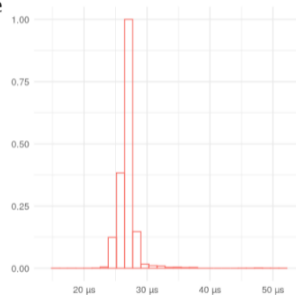
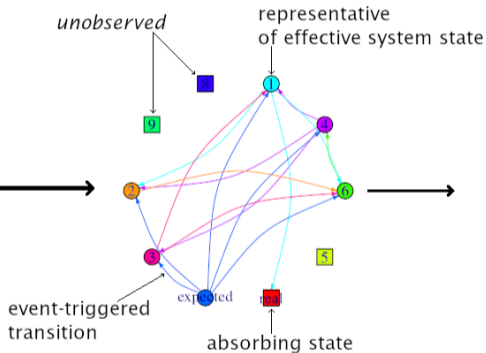


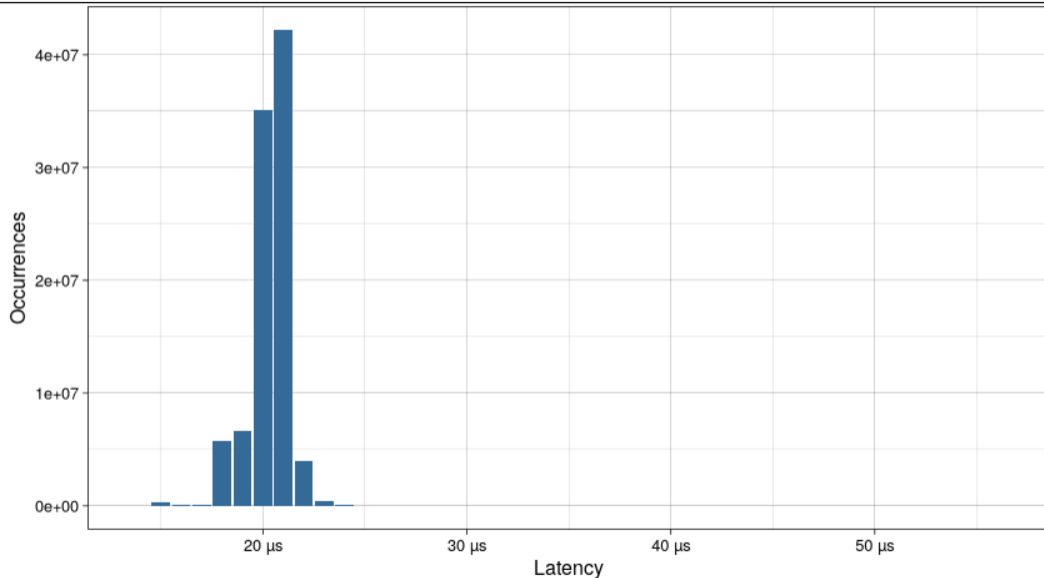
Transition	Latency
1→2	120
2→3	200
3→1	60
1→2	80
2→5	100
5→4	130
4→6	50
6→1	10
1→2	120

cyclictest

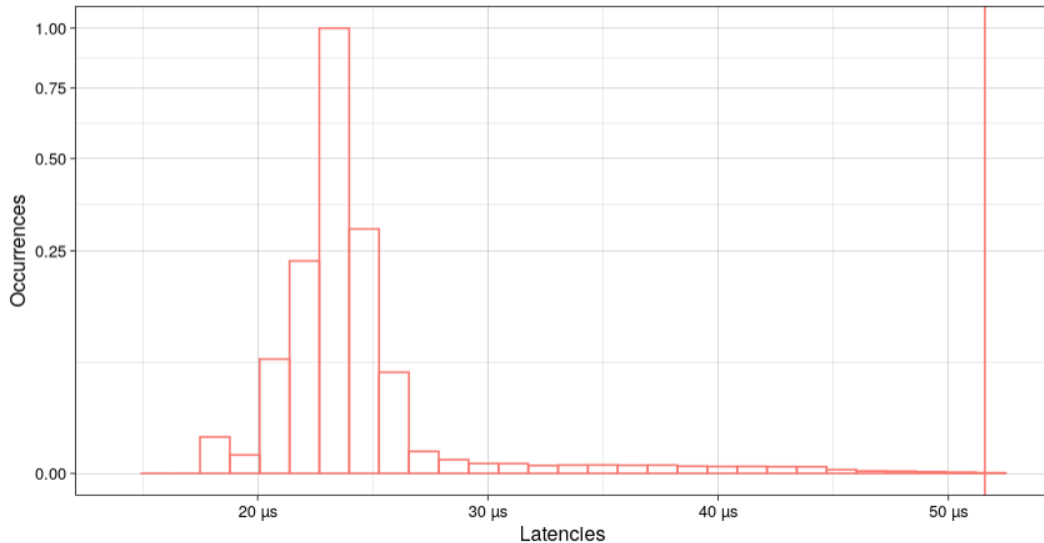
Transition	Latency
1→2	120
2→3	200
3→1	60
1→2	80
2→5	100
5→4	130
4→6	50
6→1	10
1→2	120

Kernel

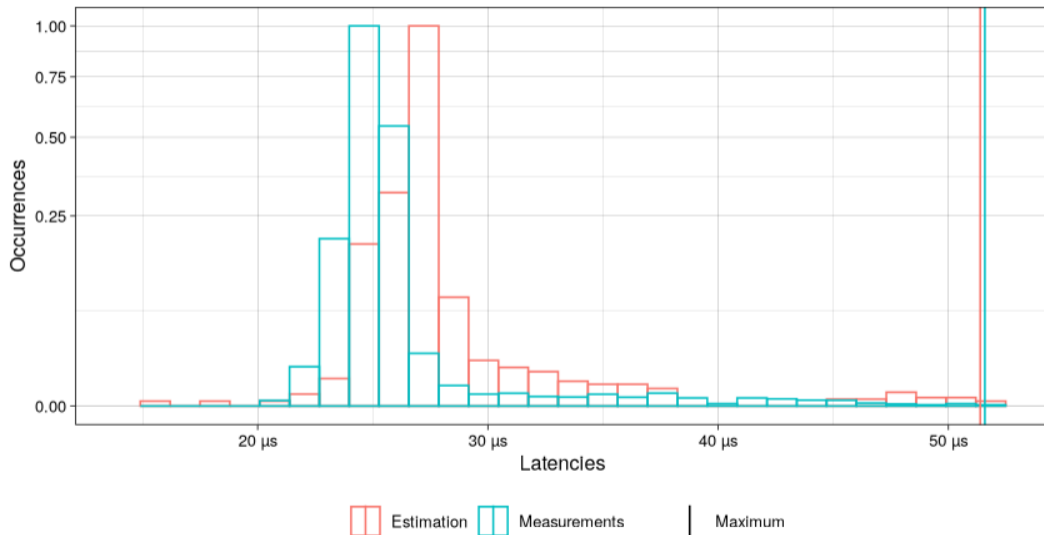




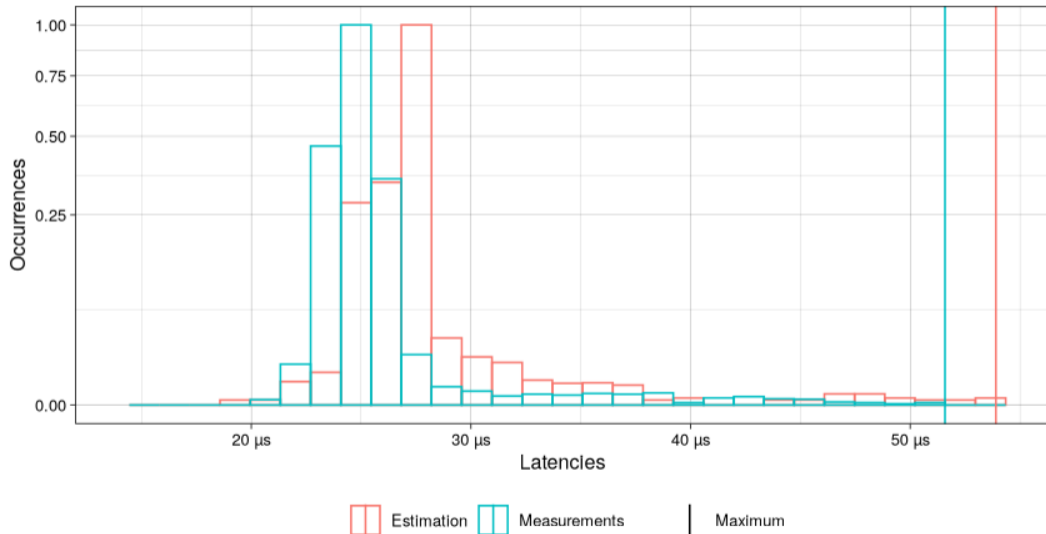
Measured 600 seconds



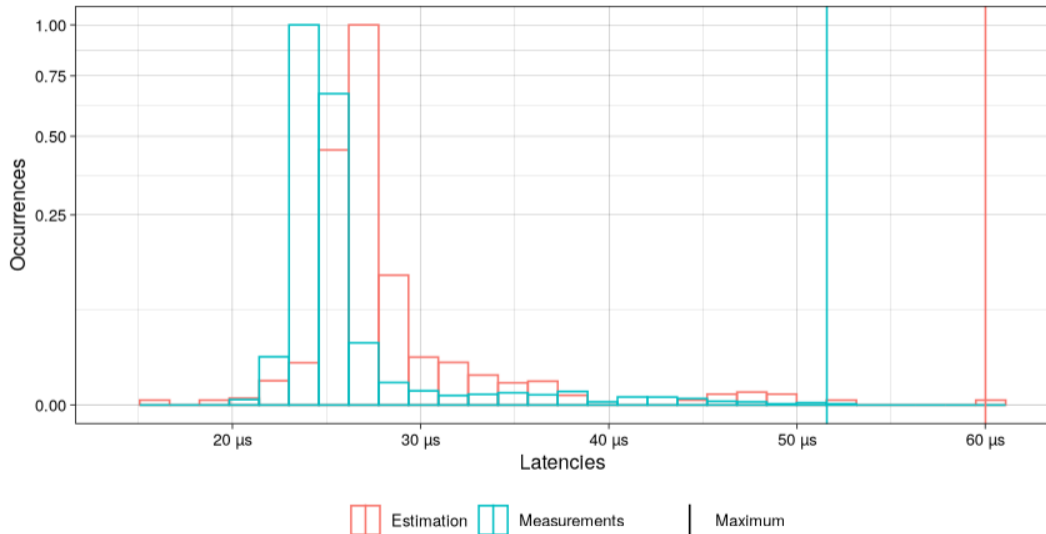
Measured 60 seconds



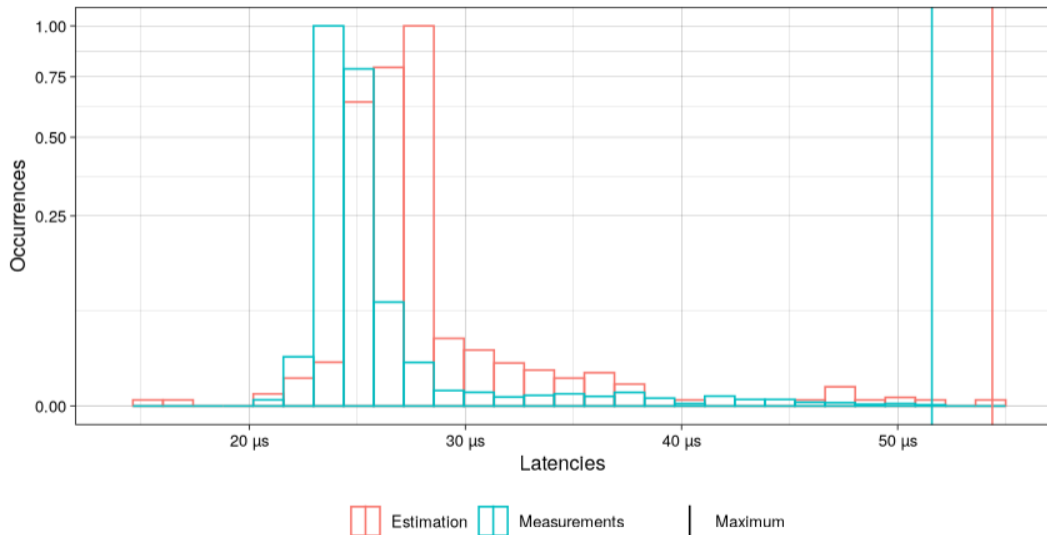
Measured 120 seconds



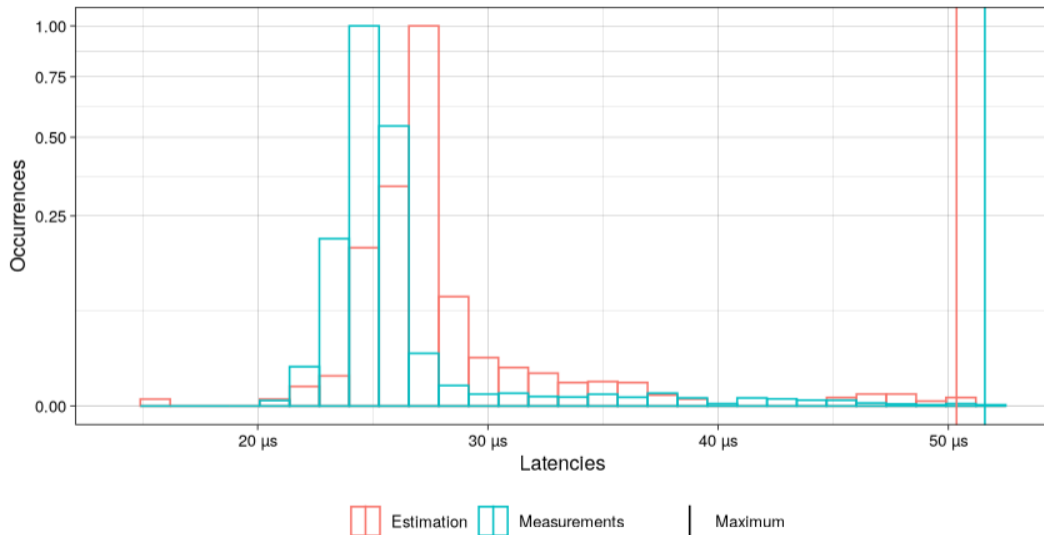
Measured 180 seconds



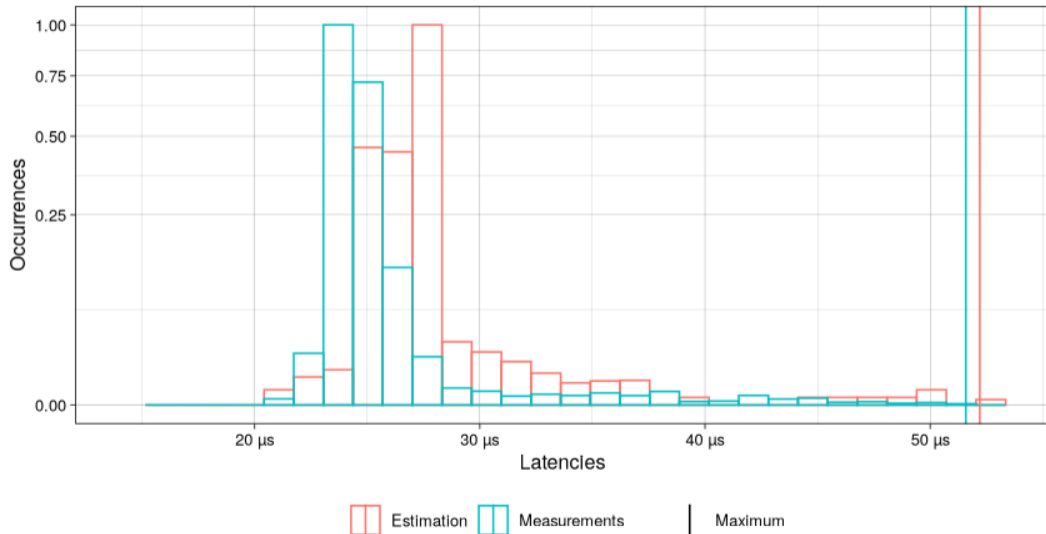
Measured 240 seconds



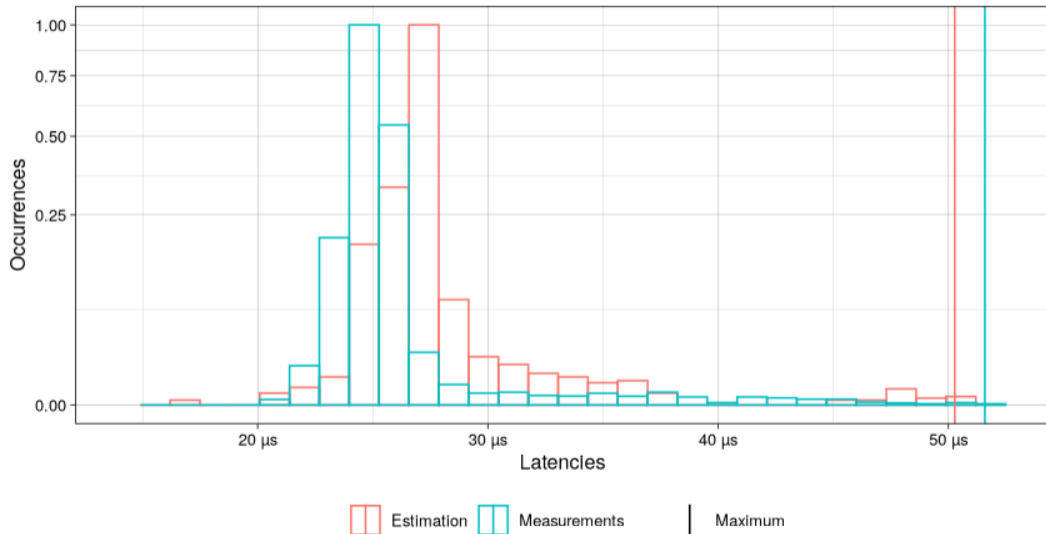
Measured 300 seconds



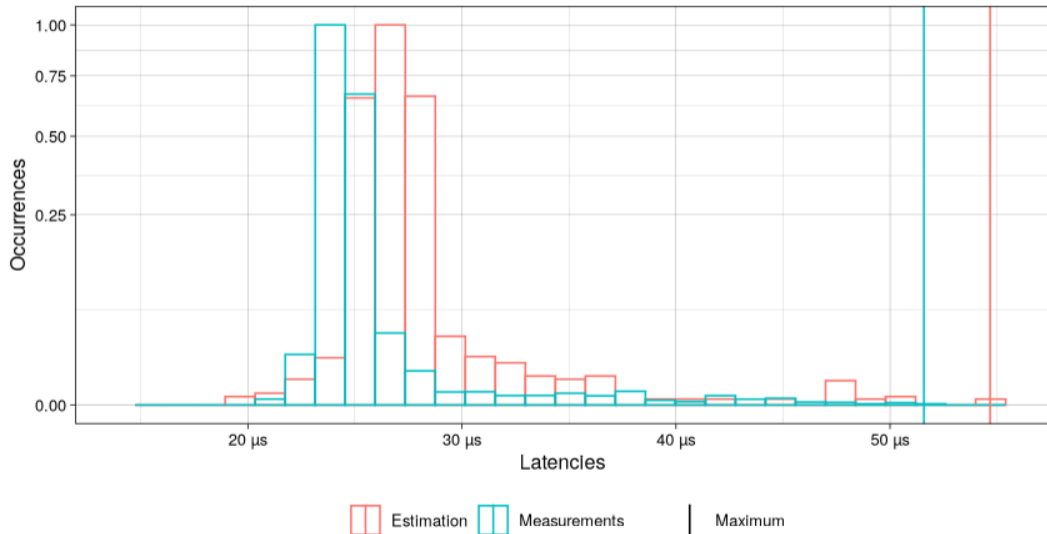
Measured 360 seconds



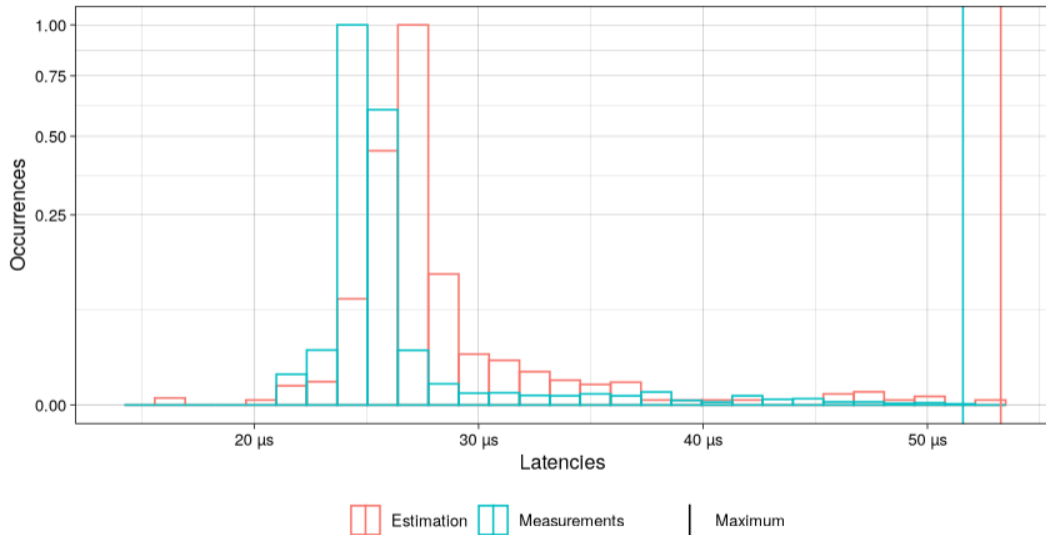
Measured 420 seconds



Measured 540 seconds



Measured 600 seconds



Discussion with the RT folks

Discussion with the RT folks

- ▶ Current focus: prediction of WCET. Other applications?

Discussion with the RT folks

- ▶ Current focus: prediction of WCET. Other applications?
- ▶ Do you incorporate using statistical method?

Discussion with the RT folks

- ▶ Current focus: prediction of WCET. Other applications?
- ▶ Do you incorporate using statistical method?
- ▶ What have been pain points using probabilistic tools?

Thank You!