THALES

ECSEL Joint Undertaking

CEAN 12

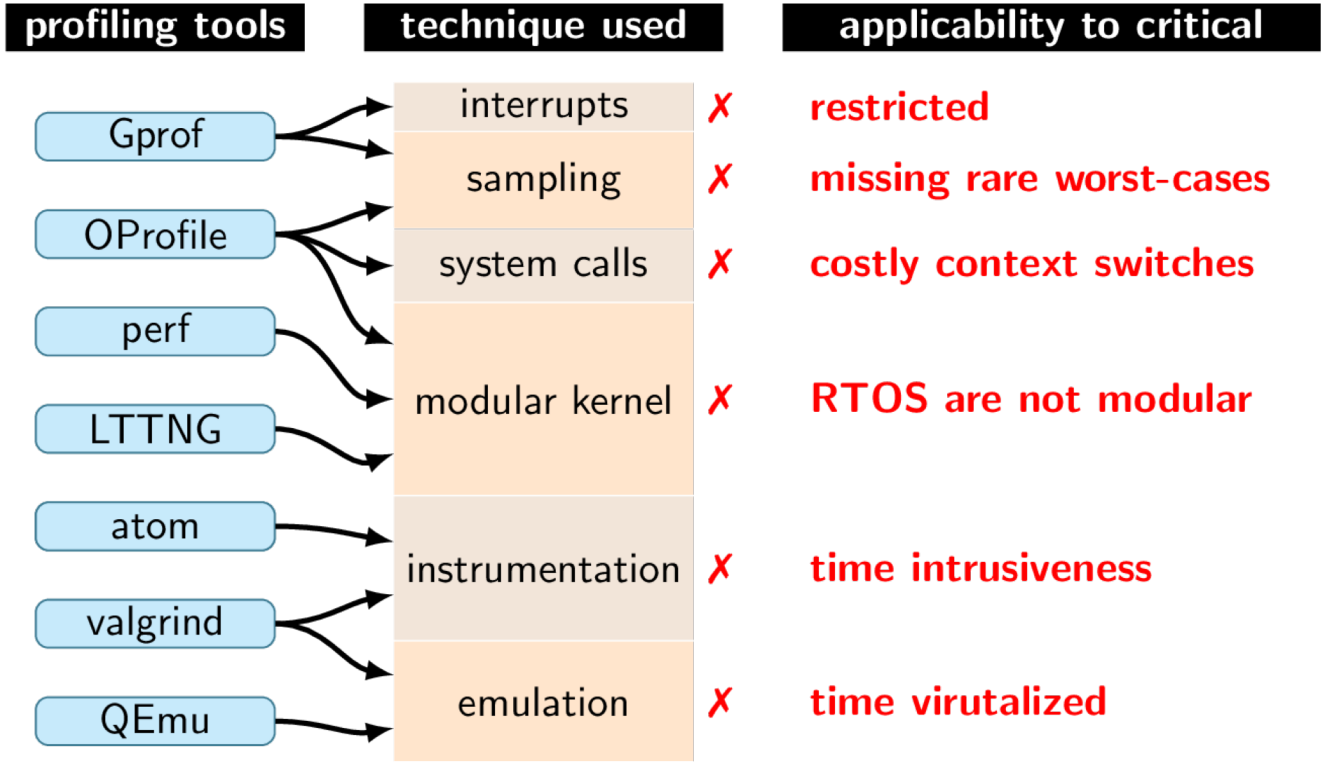
The challenge of profiling multi-core safety-critical embedded systems

Sylvain Girbal, Jimmy Le Rhun
Thales Research & Technology

WATERS 2019: Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems

At 31st Conference on Real-Time Systems (ECRTS'19)
9-12 July 2019, Stuttgart, Germany

Applicability of HPC profiling tools to safety-critical systems



Existing Linux tools are not available for time critical-class **RTOS** environments

THALES

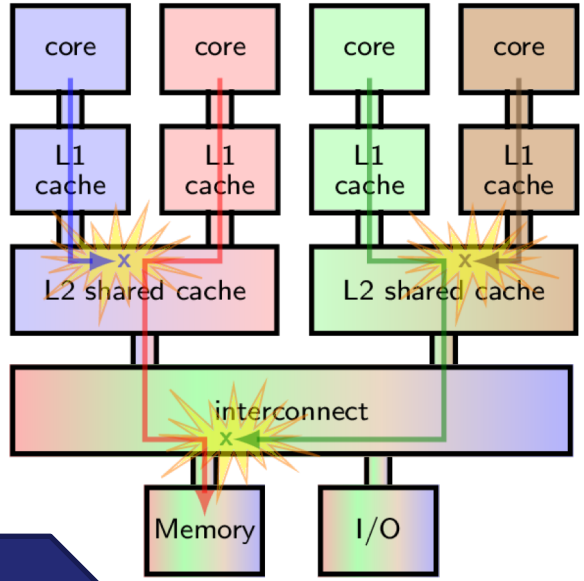
The Challenge of Timing Interferences

Using multi-core COTS in Safety Critical Systems

- Successfully faces the exponential increase of performance requirements
- But focuses on **best-effort** performance, not on **worst-case** performance

The problem: inter-core timing interferences

- Multi-core ➔ shared hardware resources
- Concurrent accesses to these resources are involving some **arbitration mechanisms** at hardware level
- Hardware contention is introducing unpredictable **timing interference** appearing as extra time delays
- Breaking the **time isolation** principles



How to **profile** safety-critical real-time systems?
How to measure **timing interference**?

THALES

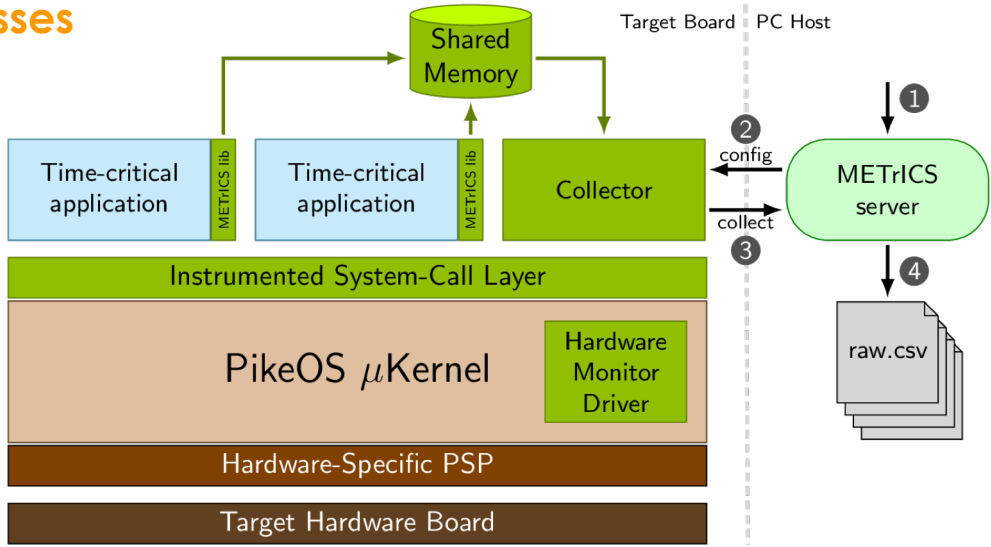
METrICS: a Measurement Environment for Time Critical Systems

Features

- Running on top of the PikeOS RTOS
- Providing accurate measurement of **timing** and **hardware resource accesses**
- Rely on hardware time base & performance monitor counters
- **Minimizing timing intrusiveness**
- Ability to observe timing interference
- Also monitor the RTOS

Software Architecture

- Driver
- Library
- Collector
- SHM
- Instrumented Syscalls



Réf. : 62 442 509 – 000 – 3 - 31/01/2017
Thales Research & Technology France
Template trtp version 8,0,2 / template : 87211168-GRP-EN-003

THALES

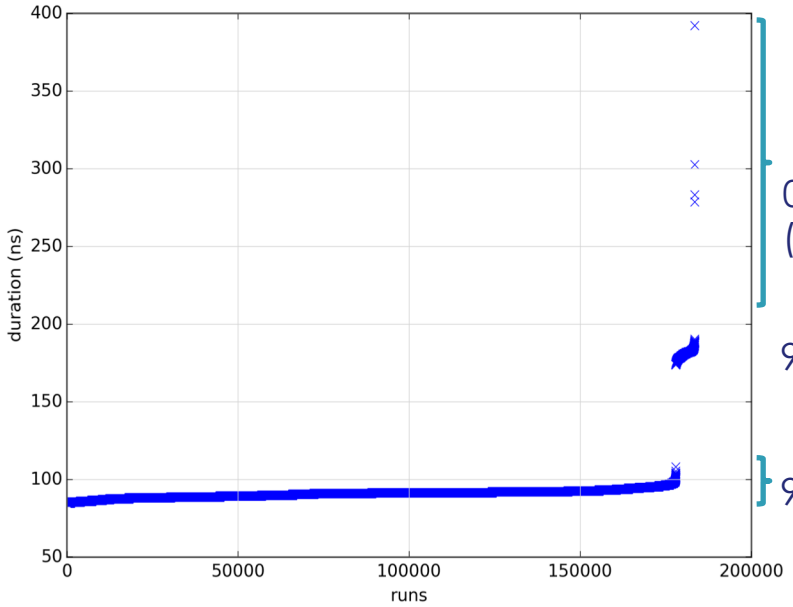
METrICS: Evaluation of precision & intrusiveness

CPU clock cycle precision

Intrusiveness of software probe:

- Inline Assembly, Macros
- No system call in the probe

Service	Layer	Resolution	Overhead
GET_TIME()	APEX	10 ms	10 ms
p4_get_time()	Kernel	1 ns	240 ns
Timebase	Register	48 cycles @1.8GHz	1.67 ns
Alt. timebase	Register	1 cycle @1.8GHz	1.67 ns



0.002% up to 392 ns (4 samples)

99.998% below 190 ns

97% below 110 ns

- Full METrICS probe:
  - Time
  - CPU ID, thread ID
  - 6 x hardware counters
  - Store in shared memory

THALES

Timing Interference versus the industry process

Interference Slowdown

- Significant WRT expected multi-core speedup
- Not covered by the usual +30% margin

processor	#core	speeddown
ARM Cortex A72	2-core	x1.98
ARM Cortex A53	2-core	x2.16
ARM Cortex A53	4-core	x3.39
T2080 PowerPC	4-core	x4.65 / x11.7
P4080 PowerPC	8-core	x12.1

Avionics domain practice

- Industry process composed of several actors: a **platform supplier**, some **function suppliers** and the **integrator**
- Mapping is decided at integration time ➔ no real opportunity to **codesign**
- Function suppliers have to guarantee execution time with unknown co-running applications
- The impact of **timing interference** depends on their application **resource usage** as well as co-runners

How to **exercise** shared hardware resource to actually **quantify timing interference** sensitivity?

THALES

Exercising shared hardware resources with stressing benchmarks

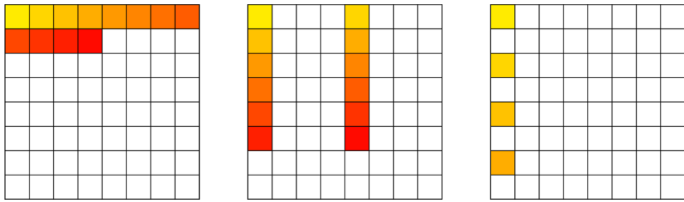
Purpose

- Stressing benchmarks are aiming at stressing a particular **hardware resource** by performing a tweakable amount of accesses to it
- To get an **upper bound estimate** of the **maximum slowdown due to** unknown co-running applications

Implementation

- Assembly code performing load/store operations
- With different possible **time** and **spatial access patterns** varying:
  - Address space ➔ memory hierarchy or I/O
  - Access stride ➔ cache locality
  - Access rate ➔ stress level
  - Total size ➔ cache locality / eviction policy

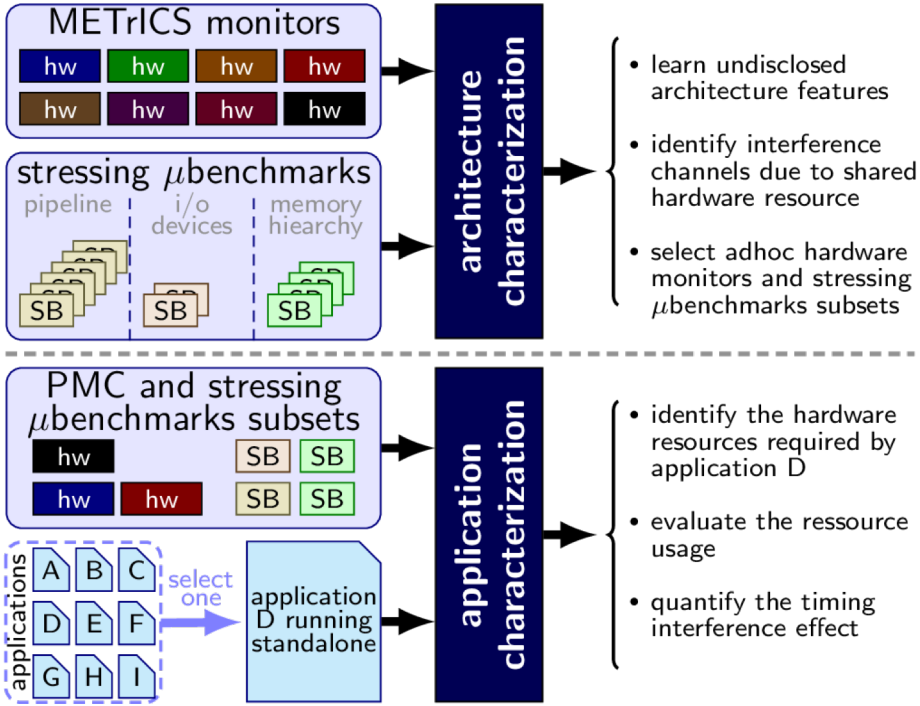
```
mov x19, address
add x18, x19, size
loop:
ld w17, [x19,0]
add x19, x19, stride
nop
...        } rate
nop
cmp x19, x18
blt loop
```

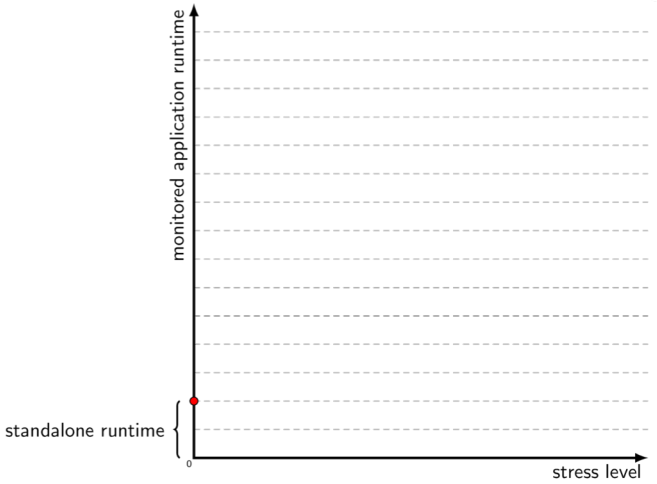


THALES

Exercising shared hardware resources with stressing benchmarks

Using stressing benchmarks



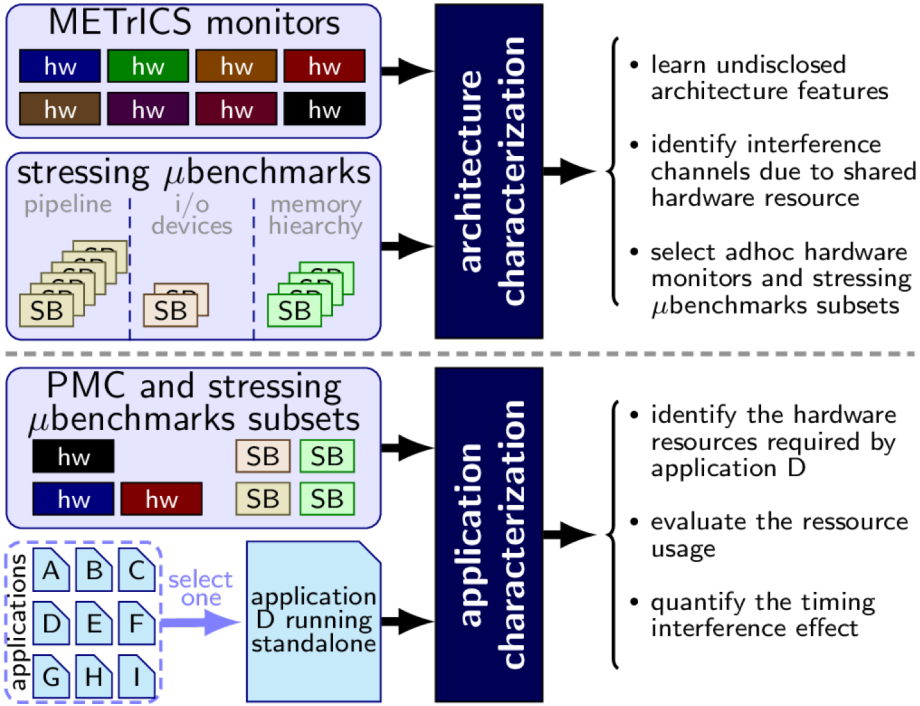


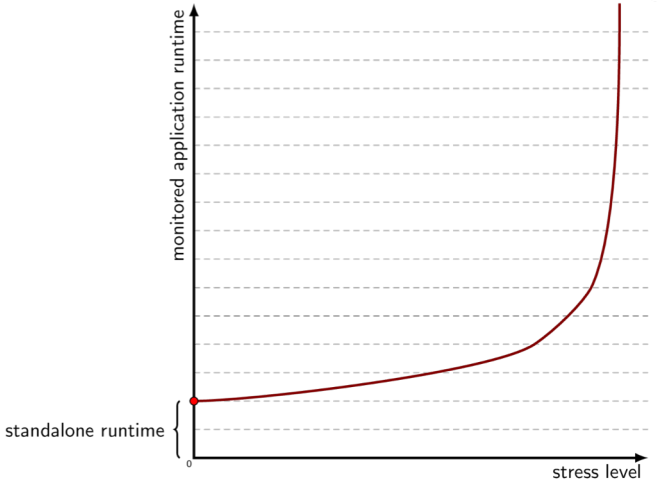
METrICS + SB ➔ **statistical traces** of timing / resource usage data with different levels of pressure on the resources

THALES

Exercising shared hardware resources with stressing benchmarks

Using stressing benchmarks



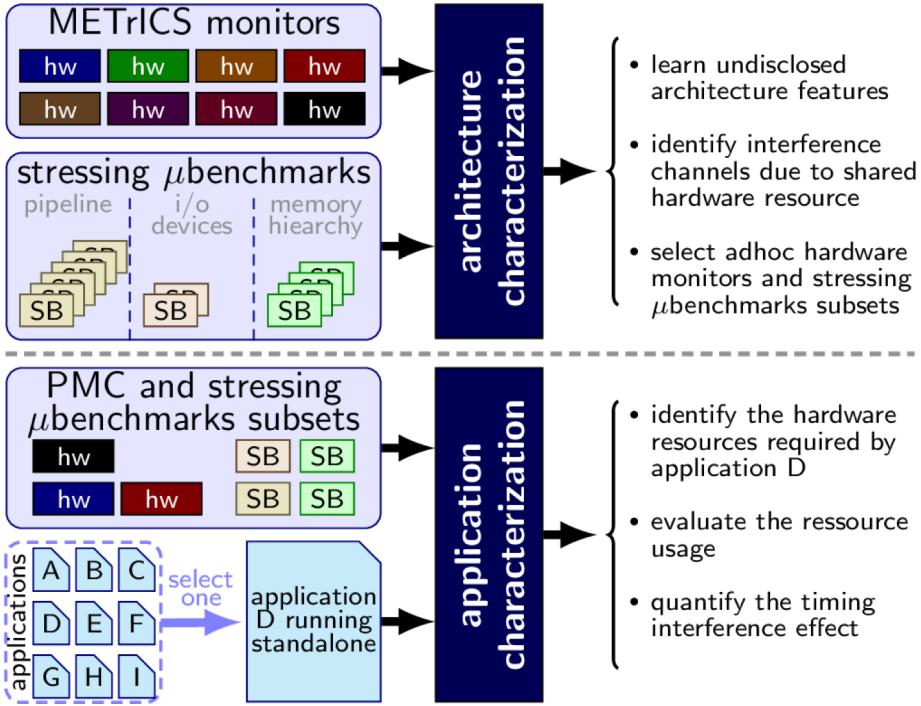


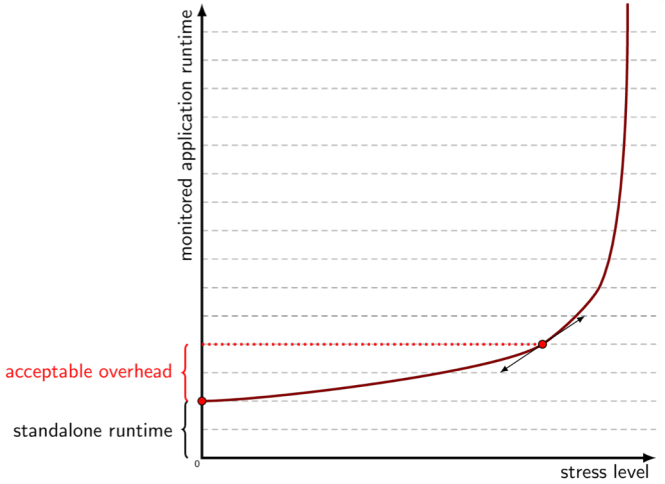
METrICS + SB ➔ **statistical traces** of timing / resource usage data with different levels of pressure on the resources

THALES

Exercising shared hardware resources with stressing benchmarks

Using stressing benchmarks



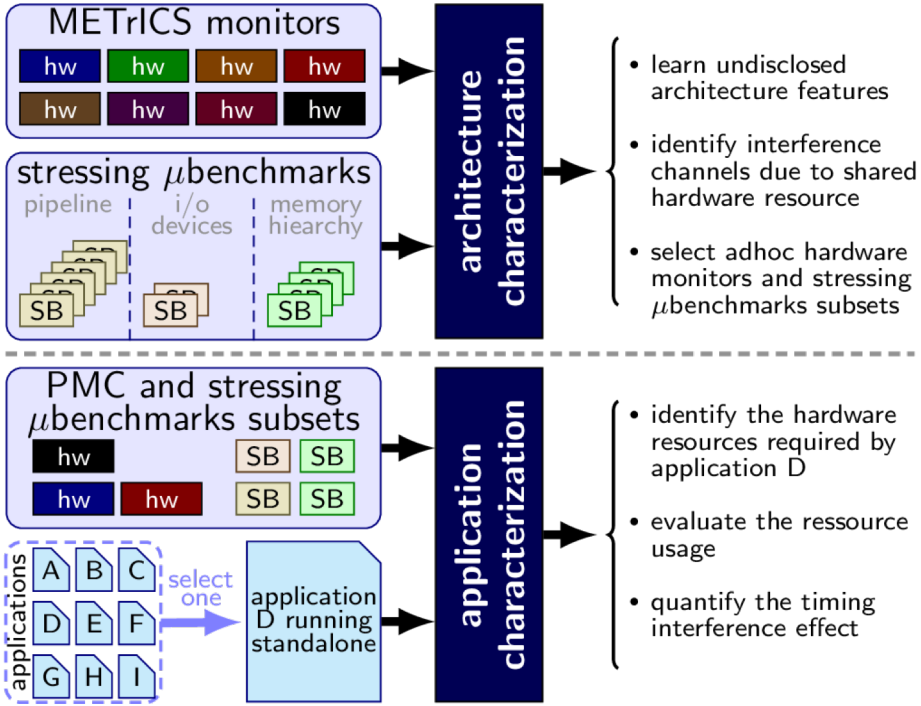


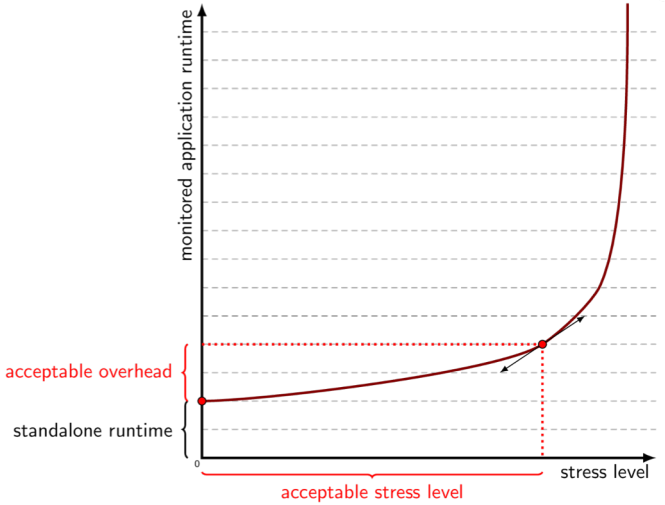
METrICS + SB ➔ **statistical traces** of timing / resource usage data with different levels of pressure on the resources

THALES

Exercising shared hardware resources with stressing benchmarks

Using stressing benchmarks



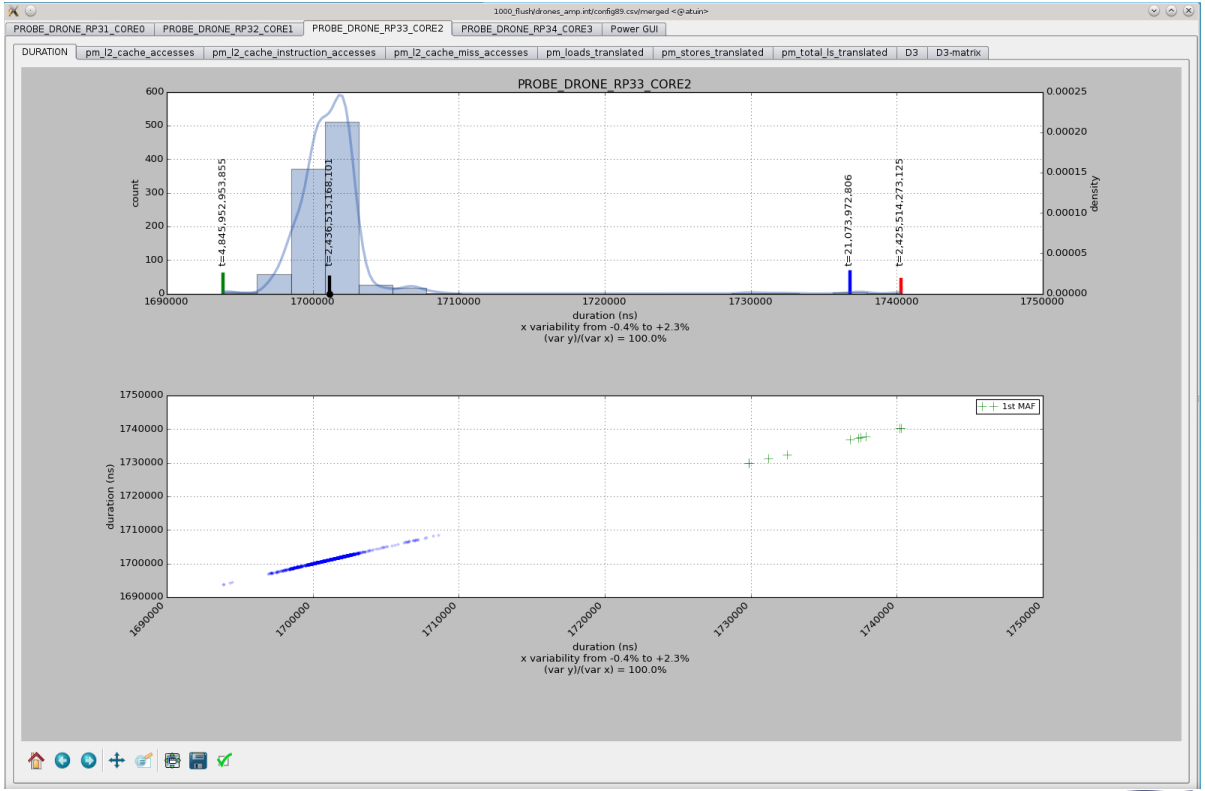


METrICS + SB ➔ **statistical traces** of timing / resource usage data with different levels of pressure on the resources

THALES

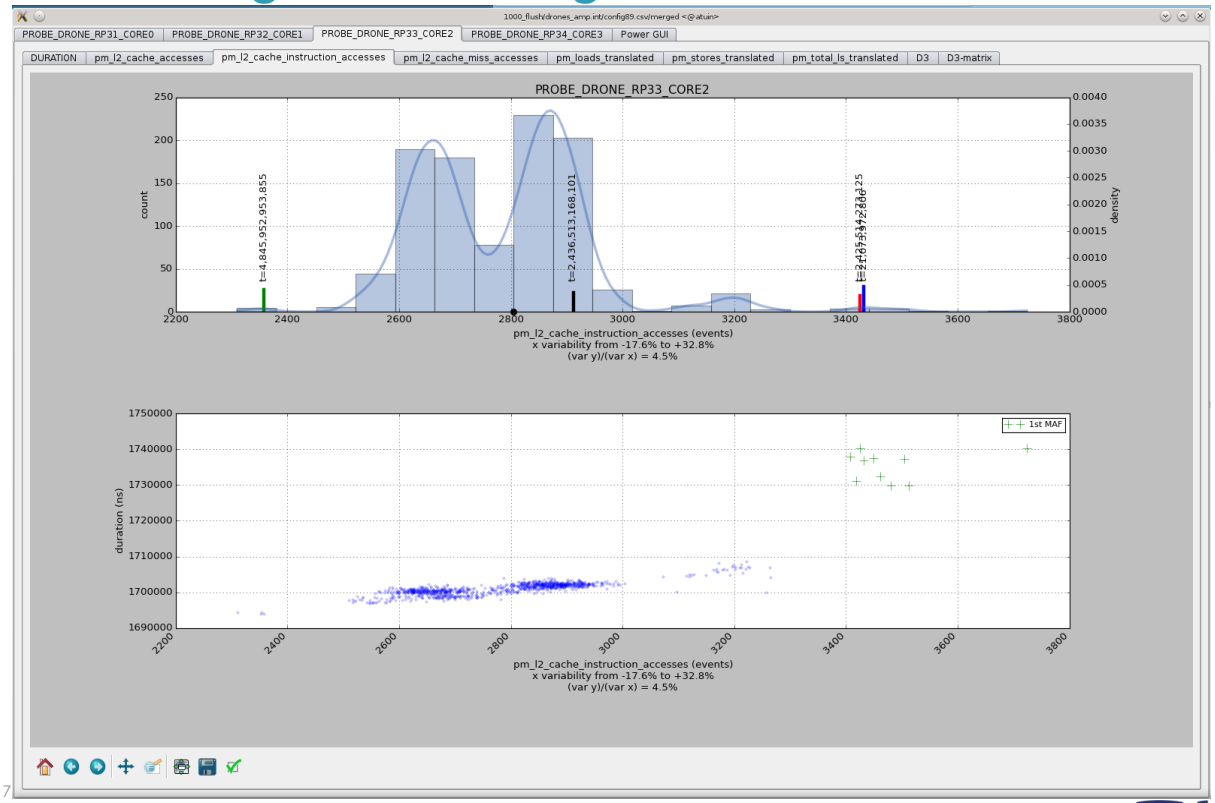
xTRACT visualizer (expert Timing and Resource Access Counting Trace visualizer)

Timing Histograms



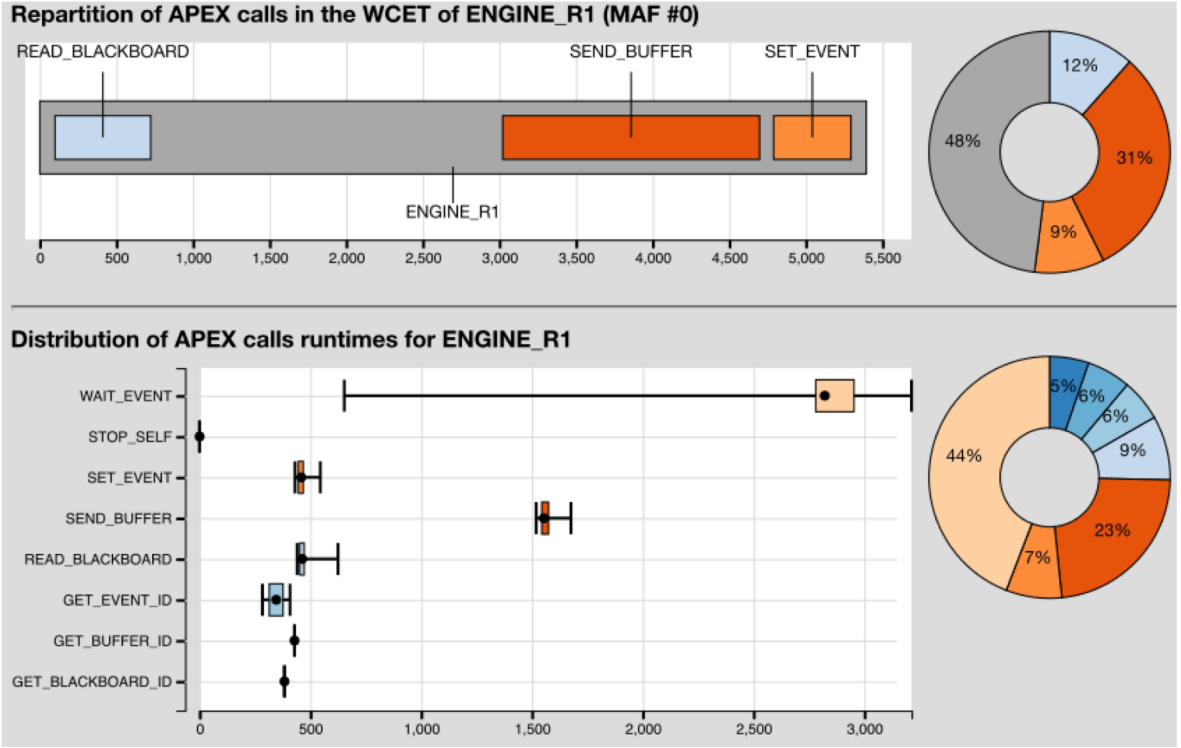
xTRACT visualizer (expert Timing and Resource Access Counting Trace visualizer)

Resource Access Histograms & Timing vs. Access correlation



xTRACT visualizer (expert Timing and Resource Access Counting Trace visualizer)

System Call Instrumentation

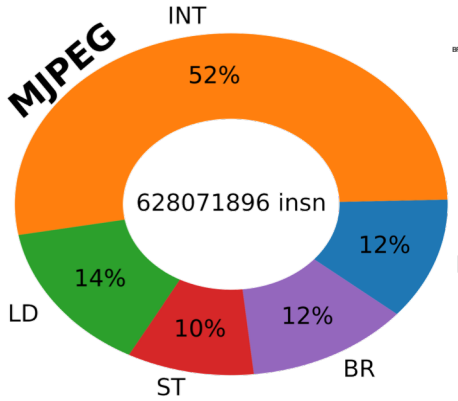


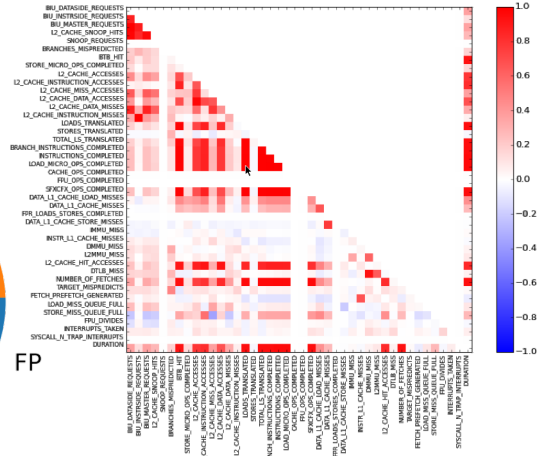
Réf. : 62 442 509 – 000 – 3 - 31/01/2017
Thales Research & Technology France
Template trtp version 8,0,2 / template : 87211168-GRP-EN-003

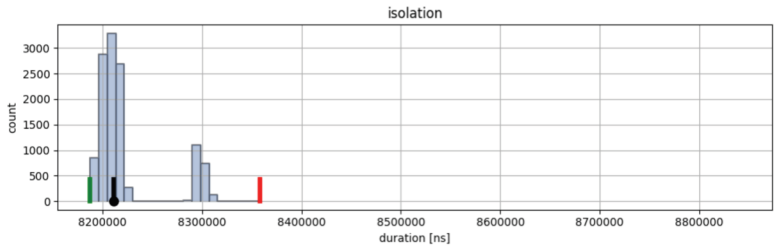
THALES

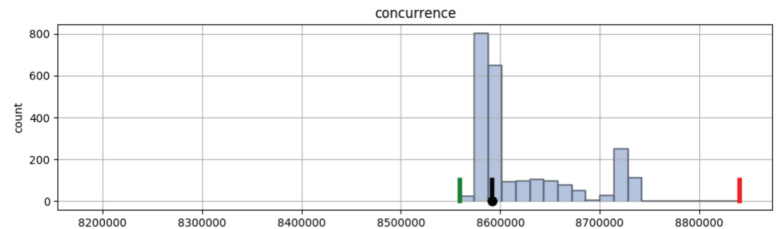
xTRACT visualizer (expert Timing and Resource Access Counting Trace visualizer)

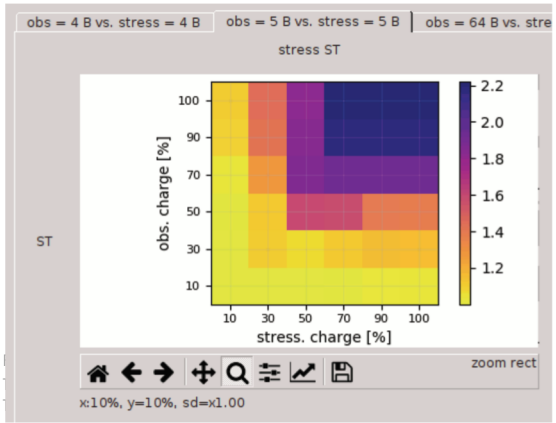
Other profiling views

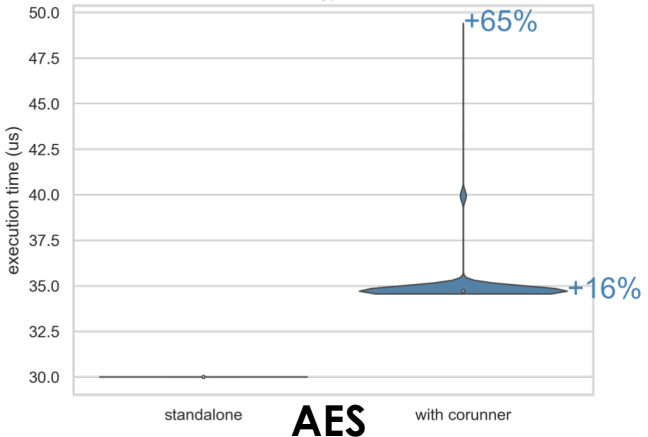












THALES

Conclusion and future works

METrICS for profiling time-critical systems

- High precision and low overhead profiling solution
- Taking into account hardware performance monitors
- With automation infrastructure and data analysis tools
- **Perspective:** support more hardware & RTOS

Stressing Benchmarks

- Critical for hardware & application characterization VS usage domain
- Identifying **timing interference channels** became parts of standard requirements
- Also used to check for **software locks** at RTOS service level
- **Perspective:** support more **hardware family** & more **hardware peripherals**

xTRACT visualizer

- For multi-core, statistical information are now critical to efficiently bound WCET
- **Perspective:** provide time behavioural information to the **function provider**