

---

# Sparse cogitations on (some) MCS challenges



---

**Tullio Vardanega**

tullio.vardanega@math.unipd.it

HiPEAC 2016

EMC<sup>2</sup> workshop

20 January 2016



UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA

---

# Disclaimers /1

- This talk addresses the MC<sup>2</sup> (*mixed-criticality on multicore*) systems problem space
- This slant is consistent with the grander challenge of the hosting EMC<sup>2</sup> project (supposedly)
- And it matches the dominant direction of the MCS research by the real-time systems community



---

# Disclaimers /2

- This presentation uses material from
  - Jim Anderson's (UNC) keynote talk at WATERS 2015
  - The PROXIMA project tutorial at ESWEEK 2015
  - A technical report by Vincent Nelis (CISTER)
  - A couple of earlier presentations of mine
- But this is *not* stale and old material because it addresses a



---

# Understanding the MCS question

- The advent of multicore processors creates a wave of opportunities and challenges in many application domains
- **Opportunity**
  - Transition from **federated systems** (with unwelcome harness, unused spare, workmanship hazard) to **integrated systems** *with some degree of isolation*
- **Challenge**
  - The integration solutions adopted for single processors **do not scale** well



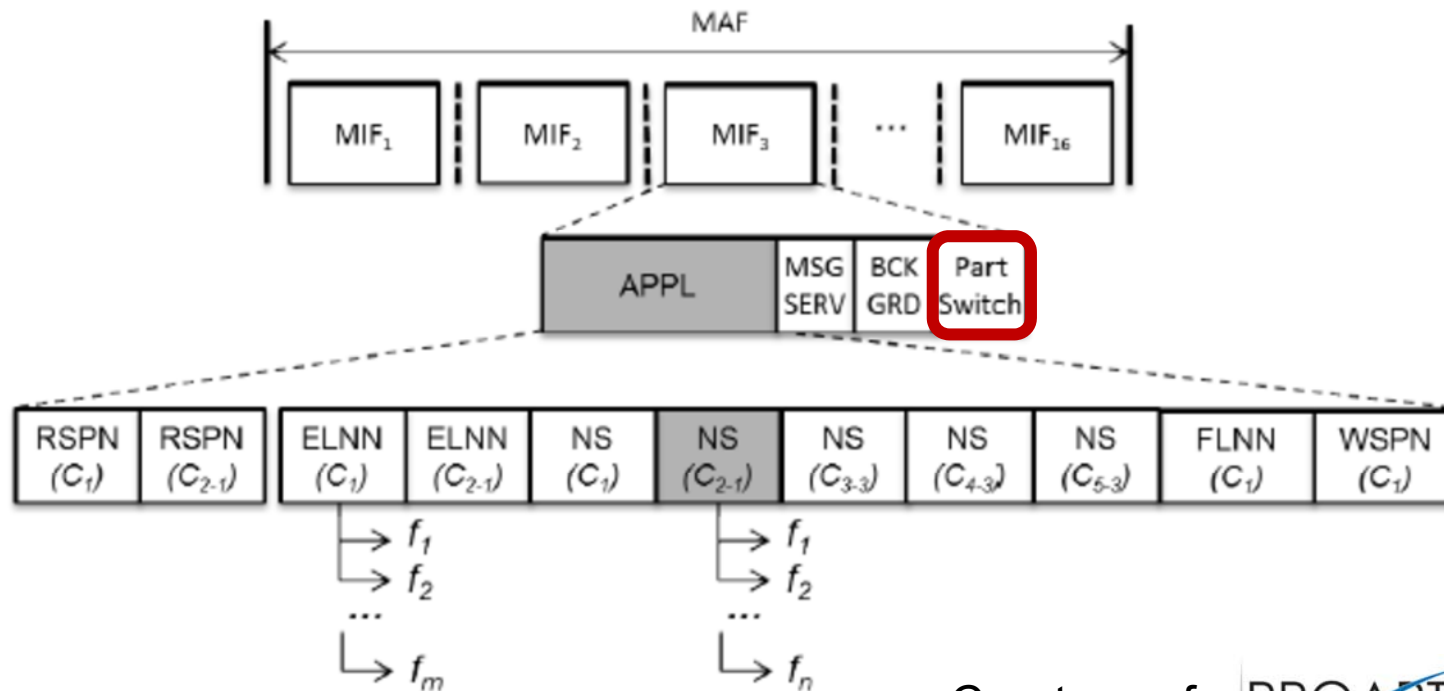
# Understanding the ex-ante /1

- The integration solutions for single processors fall under the umbrella term of **TSP** (*time and space partitioning*)
  - Memory space is segregated by design and supervised at *partition* switches
    - Caches are flushed on partition switch so that there is no inter-partition interference
  - Time is allocated in slices to partitions and partitions do what they please with their slices
    - Slice overruns are prevented by margin provisioning (insufficient science but sufficient confidence or extreme scientific pessimism)



# Understanding the ex-ante /2

- The TSP model is typified by the IMA (*integrated modular avionics*) and its ARINC 653 interpretation



Courtesy of PROARTIS

---

# Understanding the ex-ante /3

- The TSP model silently builds on the *single-runner* assumption
  - The intrinsic reality of single-CPU computing
    - Execution is strictly sequential
    - Concurrency is obtained by transparent interleaving
- The level of underutilization caused by *overprovisioned* static allocation is naturally upper bounded by the limited CPU capacity available
- The waste is more than offset by securing the grail of *incremental development and qualification (IDQ)*



---

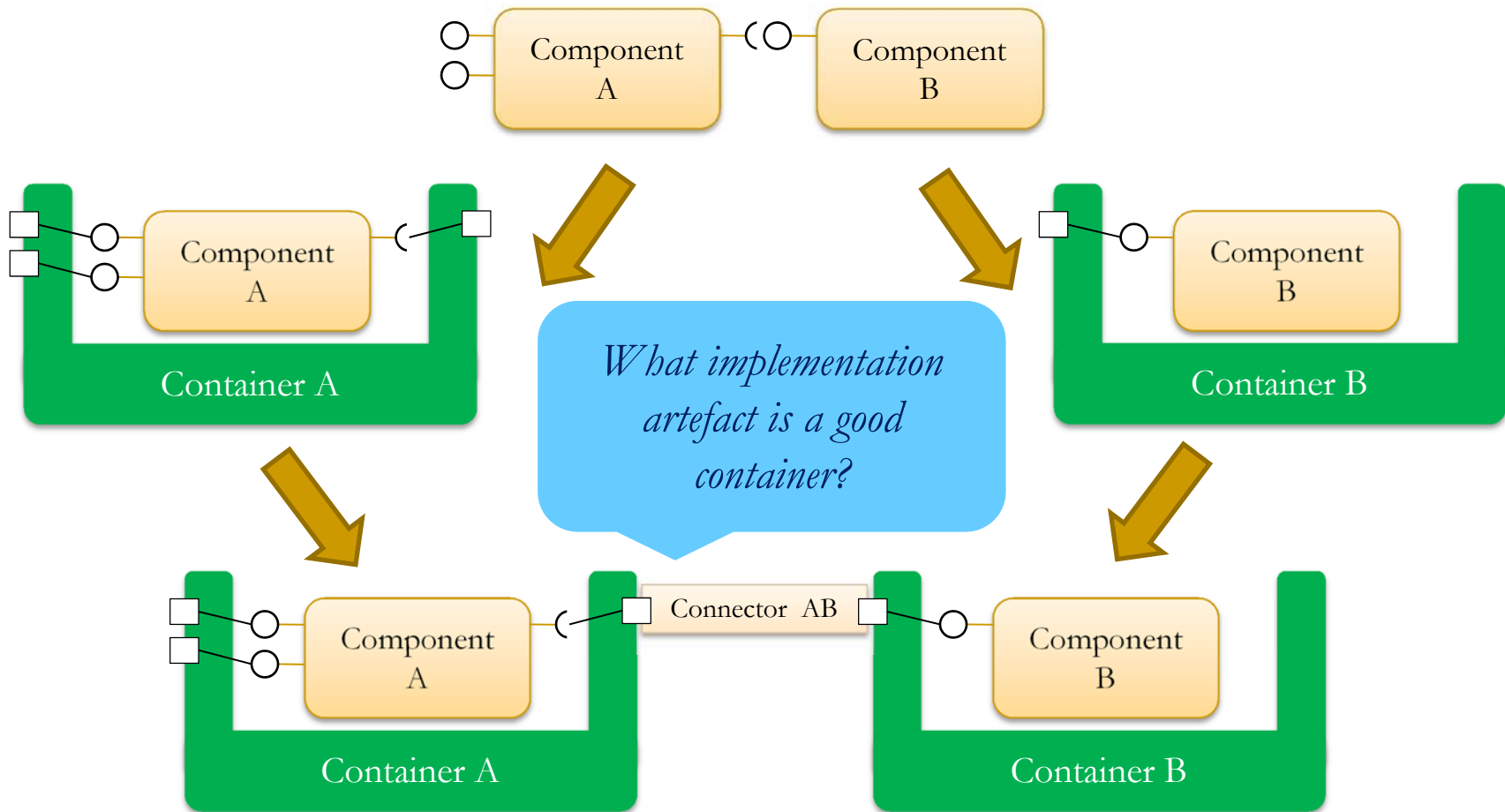
# Understanding IDQ /1

- In general, one pursues IDQ by separating **application contents** (considered in isolation) from their individual *system container*
  - The former is a distinct part of the application (aka *component*), independently developed or supplied
  - The latter is the cocoon that the system architecture wraps around individual components to assure conformance to the required model of computation and the sought guarantees of **sufficient independence**





# Understanding IDQ /2



Courtesy of  **CONCERTO**  
Guaranteed Component Assembly with Round Trip Analysis  
for Energy Efficient High-integrity Multi-core Systems



# Understanding IDQ /3

- Industry wishes IDQ to be preserved on transition to multicore processors
- Perhaps, even at the cost of hitting the notorious “*one out of m*” problem



When using an m-core platform in a safety-critical domain, **analysis pessimism** can be so great that the capacity of the “additional”  $m - 1$  cores is entirely negated

Jim Anderson @ WATERS 2015

- The MCS RT literature makes a number of strong assumptions here that we should look into carefully

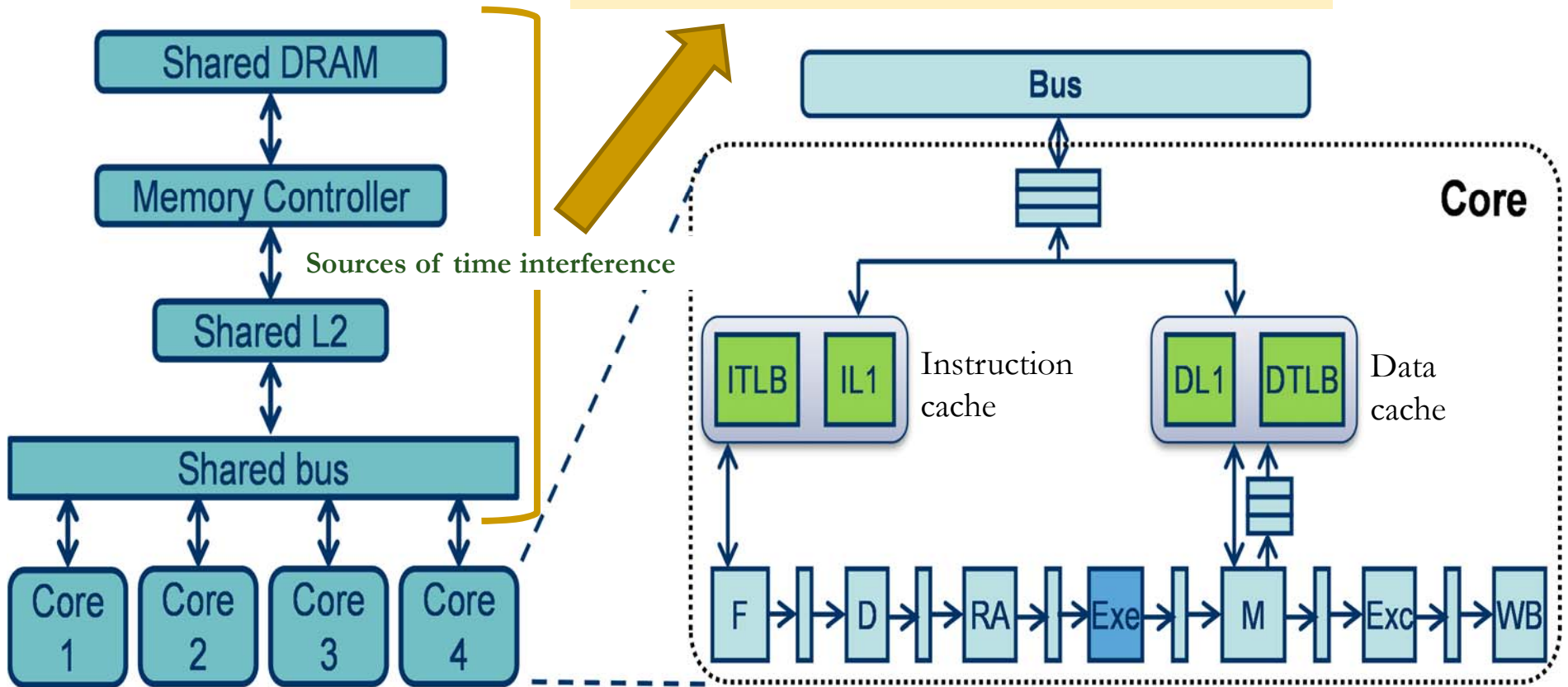
# Understanding the ex-ante /4

- Why is multicore timing analysis prone to gigantic pessimism?
- Because the single-runner assumption just **breaks** on transition to multicore processors
- Conventional HW architectures have numerous sources of interference among parallel co-runners
  - **Difficult to avoid**, short of imposing a single runner per time slice for the *whole* system
    - Which – hopefully – is unspeakable
  - **Difficult to bound**, short of massive overprovisioning that defeats the purpose
    - Which is bad



# Understanding the ex-ante /5

Lots of shared HW, before even looking at SW!



Courtesy of **PROXIMA**



# Composability & compositionality / 1

$R_i$  is a *compositional* expression

Its RHS equation benefits from *composable* terms

$$R_i^{n+1} = B_i + CS1 + C_i + \sum_{j \in hp(i)} \left[ \frac{R_i^n + J_j^A}{T_j} \right] (CS1 + C_j + TS + CS2) + I_{clock}^{R_i^n} + I_{extInt}^{R_i^n}$$

Blocking time  
(resource access  
protocol or *kernel*)

“In” context  
switch

“Activation”  
jitter

“Out” context  
switch

Interference  
from the clock

Interference  
from interrupts


Time to issue  
a *suspension* call

$$R_i^0 = B_i + CS1 + C_i$$

$$R_i = R_i^n + J^W \text{ “Wake-up” jitter}$$



# Composability & compositionality /2

- The single-runner model of computation of traditional processor HW allows systems to enjoy some extent of *time composability* (TC)
  - Intrinsic at the HW level
  - Aided by design and implementation choices at SW level
- Multicore processor architectures shatter those premises and TC can longer be attained 
- The question then becomes whether TC has more shades of grey than just all-or-nothing

# Ramifications /1

Intentionally avoiding use  
of the term “criticality”

- With IDQ, distinct application parts may be developed at different levels of *integrity*
  - The essential obligations attached to them are sanctioned by the customer in contractual arrangements
    - Calling a given API, limiting the footprint, living within a bounded execution time budget, meeting all application requirements, ...
  - Responded by the supplier with the provision of factual evidence and assurance of given guarantees
- As those guarantees may be insufficient, safeguarding measures must be adopted against violations
  - **By architectural choices and run-time means**

# Ramifications /2



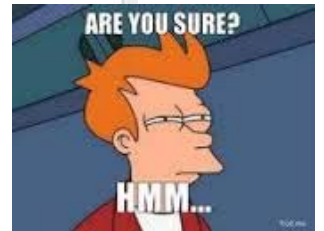
- The ‘C’ in MCS makes rather lax use of the term “criticality” to mean something else really
- The central concern of the *mixed-criticality analysis model* is the WCET of SW programs
- But “criticality” does **not** correspond (directly) to it!
- In the same vein, some authors relate “criticality level” to SIL (*safety integrity level*)
- But this is equally doubtful as the SIL concept is related to importance and confidence
- This abuse has given rise to quite some confusion and a number of ill-founded speculations



# What is Mixed-Criticality Analysis?

(Vestal [RTSS '07])

- Each task is assigned a **criticality level**.
- Each task has **provisioned** execution time (PET) specified at each criticality level.
  - » PETs at higher levels are (typically) larger.
- **Example:** Assuming criticality levels A (highest), B, C, etc., task  $\tau_i$  might have PETs  $C_i^A = 20$ ,  $C_i^B = 12$ ,  $C_i^C = 5$ , ...
- **Rationale:** Will use more pessimistic analysis at high levels, more optimistic at low levels.



# Insert: understanding SIL



P. Graydon and I. Bate  
Safety assurance driven problem formulation for  
mixed-criticality scheduling  
Workshop on Mixed-Criticality Systems @ RTSS 2013  
[http://www-users.cs.york.ac.uk/~robdavis/wmc2013/  
paper14.pdf](http://www-users.cs.york.ac.uk/~robdavis/wmc2013/paper14.pdf)

The link from this to WCET is far removed

# Ramifications /3



- Time is the principal area of concern for misbehaviour in real-time systems, so that's what we talk about here
- Two contrasting high-level goals around time
  - High schedulable utilization (aka, **maximum guaranteed performance**)
  - Sufficient guarantees that the important services are always delivered in time (**no hard deadline miss**)
- Two alternative solution architectures
  - *Asymmetric guarantees*: one-level scheduling with some run-time monitoring
  - *Symmetric guarantees*: multi-level scheduling or hierarchical execution with run-time enforcement



# Memento

- Assuming we know how to compute the WCET of SW programs running on a multicore
  - (Which we don't really ...)
- We can first consider the system architecture challenge
- And then return to the WCET analysis problem



# Asymmetric guarantees

Containers are essential here  
but not much discussed!

- **Overarching goal:** *low-importance low-guarantee parts cannot cause higher-importance parts to miss their deadline*
  - If a task executes longer than budgeted at the current “criticality”  $L$ ,  $L$  is raised to  $L+1$  (higher)
  - The only tasks that can continue executing at  $L+1$  are those that have residual budget at that level
    - Every other task is **immediately terminated** as their claiming CPU time would imperil the feasibility of tasks at level  $Q \geq L + 1$
  - Associated scheduling analysis algorithms ensure that schedulability is always guaranteed at the higher levels, in a cascading fashion
- No industrial-quality results as yet ...
- Unsolved real-world issues and doubts on the sanity of this model
  - Termination, no return to lower “critical levels”, no functional dependence, ...



# Survivability



Clarifying

- The system capacity to continue to deliver essential services in the event of internal or external failure
  - Executing longer than budgeted is an error state arising from a development fault but not necessarily a system failure
  - Missing a deadline might be a system failure if there is no residual utility in later completion
- **Survivability might require system reconfiguration** into an acceptable degraded form
  - The assurance goal of reconfiguration for survivability is **graceful degradation**
  - The assurance goal of tolerating overruns is **partitioning integrity**
- Reconfiguration has requirements that are poorly captured in the current MCS theory



# Symmetric guarantees

- **Overarching goal:** *every part stays within their assigned bounds regardless of any other considerations*
- More traditional ambit
- Various solutions
  - ❑ Resource-reservation kernels
  - ❑ Hierarchical budget servers
  - ❑ Partitioning (dislocation)
  - ❑ Hypervising with or without virtualization
- Yet, **no option can achieve *true* time isolation**, short of using custom HW
  - ❑ All budgeting must overprovision to compensate for intrinsic interference

All these architectures are evidently *based on containers!*

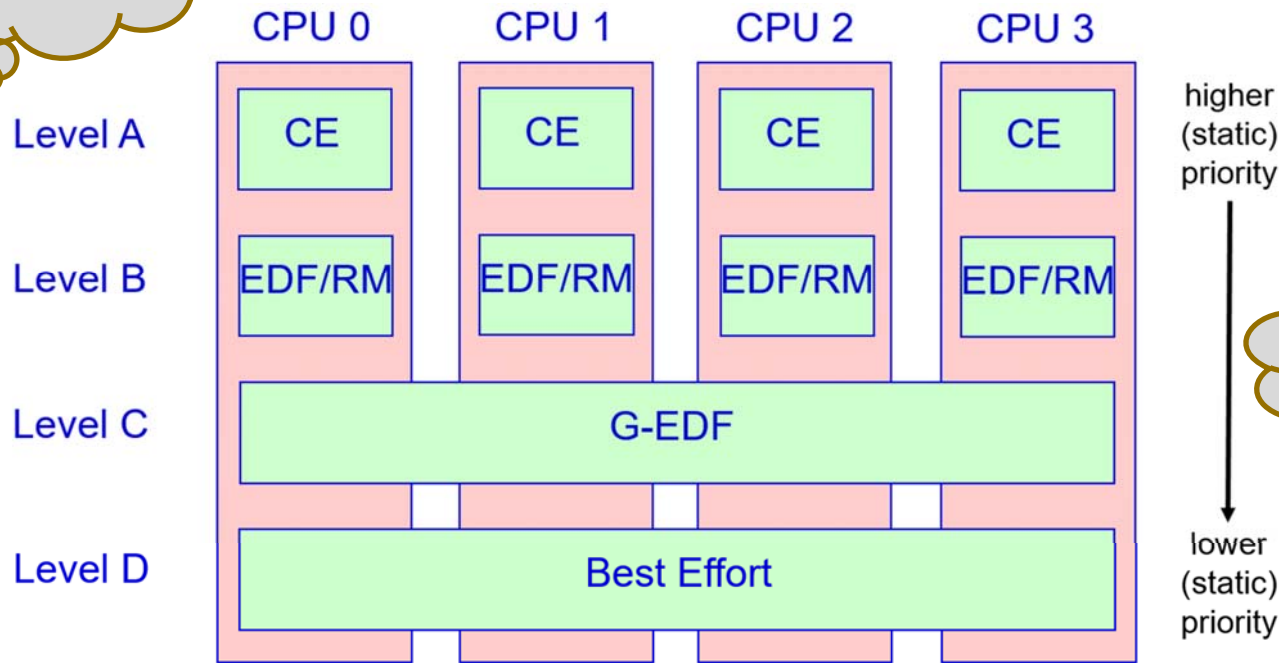


# A heavy-weight experimental architecture

## MC<sup>2</sup> Architecture

Implemented as a LITMUS<sup>RT</sup> Plugin

Symmetric  
in concept



Asymmetric in  
provisioning

What kind of containers here?

WATERS, July 2015

Jim Anderson 11



---

# Performance and guarantee trade-offs

- The “UNC” MC<sup>2</sup> architecture rests on important postulates backed by research results
  - Partitioned scheduling is better when *higher assurance guarantees* are sought
    - Trades performance for time predictability
    - Cyclic executive *vs* priority scheduling trade-off not obvious
  - Global scheduling is preferable when *higher guaranteed utilization* is sought
    - Higher run-time overhead for higher performance
    - Solutions are needed to maximally bound OS interference

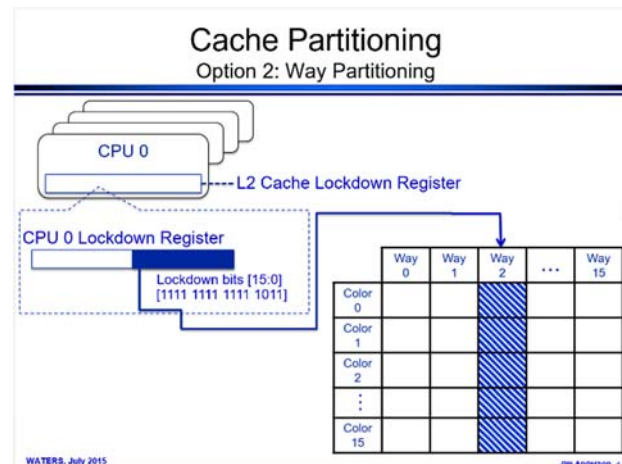


# Bounding contention: HW / 1

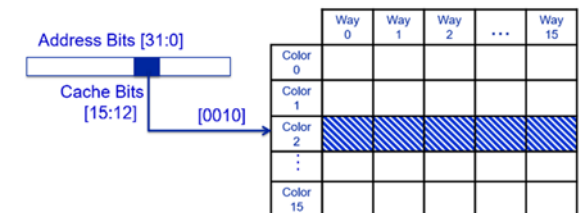


- L1 cache → already partitioned
- L2 cache → from shared (why?) to partitioned
  - Set partitioning (page colouring)
  - Way partitioning
  - Combined

	Way 0	Way 1	Way 2	...	Way 15
Color 0					
Color 1					
Color 2					
...					
Color 15					

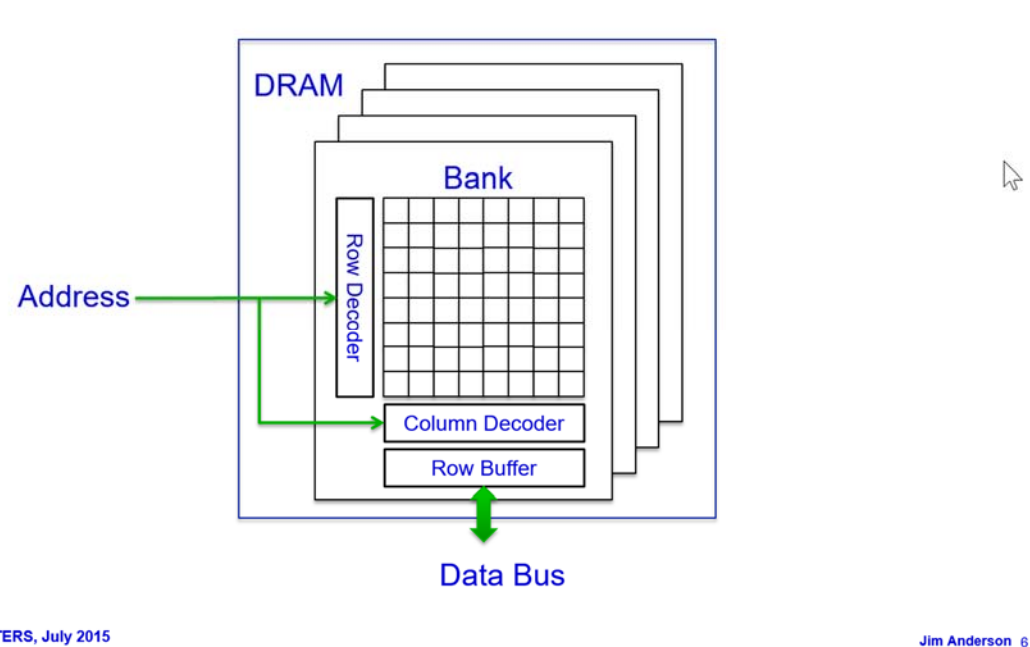


Cache Partitioning (of the Shared L2)  
Option 1: Set Partitioning, i.e., Page Coloring



# Bounding contention: HW /2

- DRAM → from shared to partitioned
  - The extent of benefit depends on the proportion of L2 that can be privately assigned



WATERS, July 2015

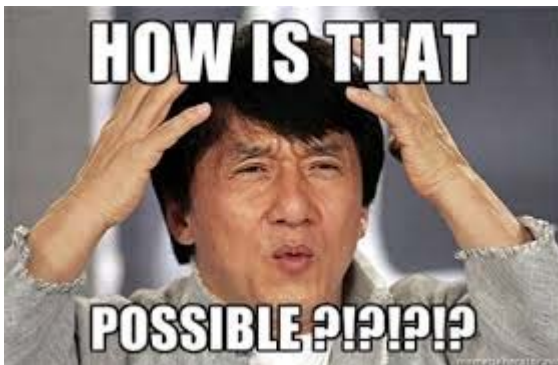
Jim Anderson 6



# Bounding HW contention / 3



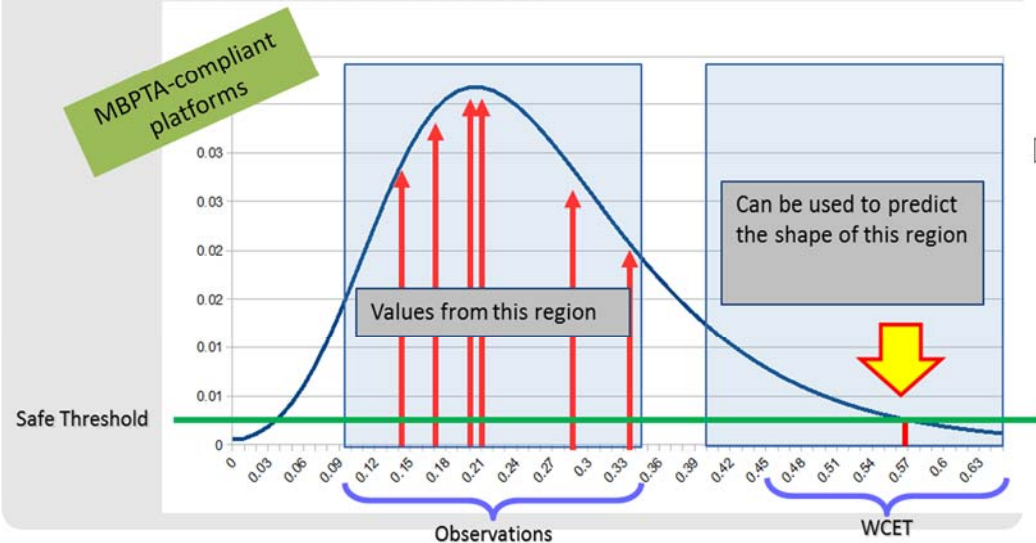
- Or taking a totally different turn and *going probabilistic*



## Measurement-Based PTA (MBPTA)

Based on knowledge of a set of measurements around the centre of the distribution

Derived from a branch of Extreme Value Theory (EVT) that allows predicting the shape of the tail of a distribution of execution times



3

Amsterdam, The Netherlands

04/10/2015

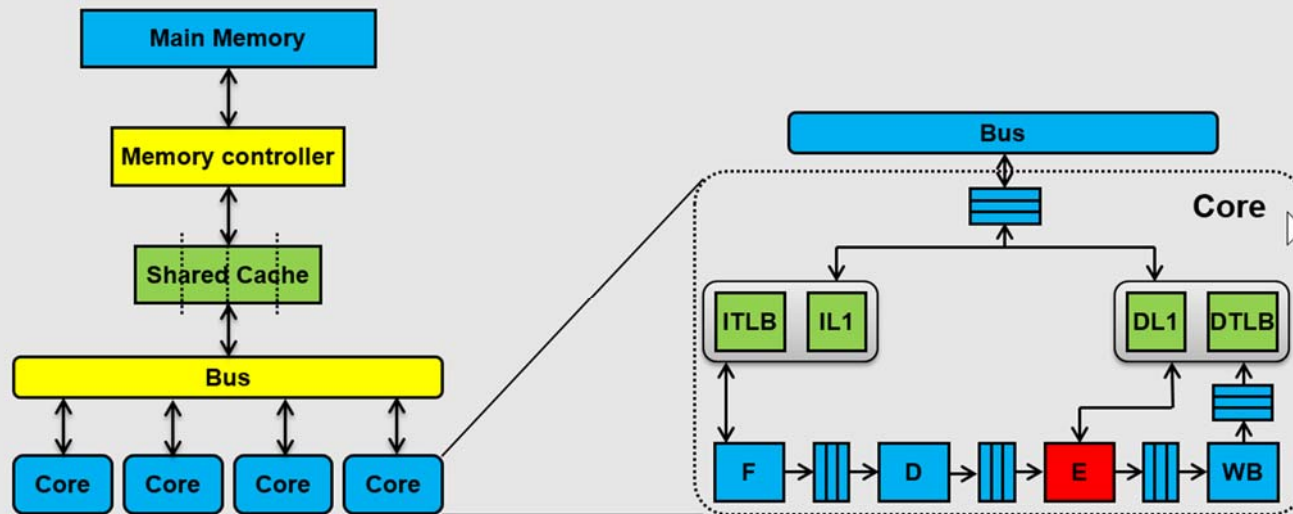
PROXIMA



# Bounding HW contention / 4

## Processor architecture

- In-order quad-core based on commercial product

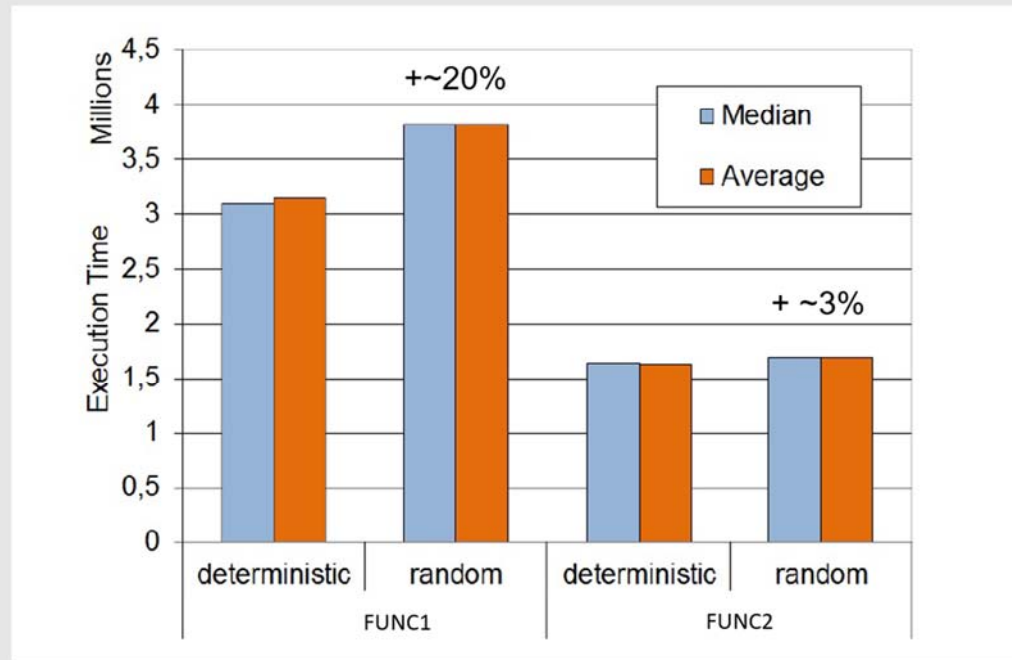


- Random placement, random replacement
- Random arbitration (e.g., random permutations)
- Upper-bounded/deterministic latencies



# Bounding HW contention /5

## Avionics use case: performance loss?



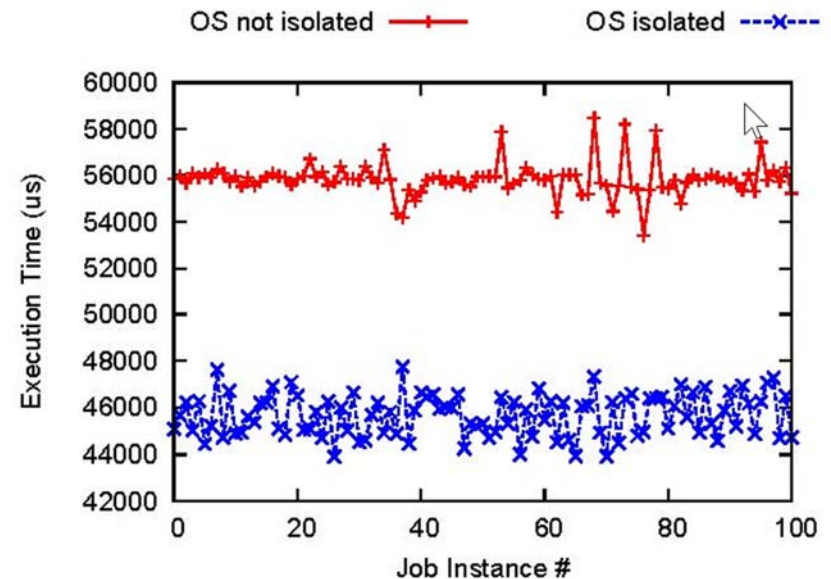
Average and Median of execution times on the randomised and deterministic architectures




# Bounding RTOS interference

- The RTOS must be made either time composable
  - Which requires a much needed revamp for *zero-interference constant-time response time*
- Or be rendered isolated
  - Which requires it to take part in the partition allocation

This is still  
Jim Anderson  
@ WATERS  
2015



# Bounding SW contention /1

- What if you share SW resources across supposedly isolated programs?
- Bad things happen ... 
- But I have *never* seen a “system” in which the application programs do *not* share logical resources
  - ❑ Do they in IMA?
  - ❑ Of course they do: “communalizing” services was a distinct purpose of it





# Bounding SW contention /2



- The premises on which single-runner sharing solutions based now fall apart
  - ❑ Suspending is no longer conducive to earlier release of shared resource ← parallelism gets in the way
  - ❑ Boosting the priority of the lock holder does not too ← per-CPU priorities may not have global meaning
  - ❑ Having local *and* global resources causes suspending to become dangerous ← local priority inversions may occur
  - ❑ Spinning protects against that hazard but wastes CPU cycles



# Bounding SW contention /3

## ■ **(Bad News) Theorem**

B. Brandenburg, 2013

B. Brandenburg, 2013

□ Under non-global scheduling (for cluster size  $c < m$ ) it is *impossible* for a resource access control protocol to simultaneously:

- Prevent unbounded priority-inheritance (PI) blocking
- Be independence-preserving
  - Tasks do not suffer PI-blocking from resources they do not use
- Avoid inter-cluster job migration

■ *Seeking independence preservation and bounded PI-blocking requires inter-cluster job migration (!)*



---

# Conclusions

- How badly do we need a good system architecture!
- Analysis work should descend from it
- Not quite the converse ...

