Multicore OS Benchmark

Francisco J. Cazorla

Mikel Fernandez

Roberto Gioiosa

Eduardo Quiñones

Marco Zulianello (TO)

Luca Fossati

Francisco J. Cazorla
(francisco.cazorla@bsc.es)
PhD. Researcher and Director of the CAOS group at BSC
(www.bsc.es/caos)
Software Systems Division & Data Systems Division
Final Presentation Days
25th April 2012, ESTEC

Agenda

- BSC and Background of the CAOS team
- Project Objectives and Motivation
- Execution infrastructure
- Experiments and results
- Conclusions and Future work

Agenda

- **BSC and Background of the CAOS team**
- Project Objectives and Motivation
- Execution infrastructure
- Experiments and results
- Conclusions and Future work

BSC

- Barcelona Supercomputing Center
- Spanish national research center (www.bsc.es)
  - +300 people at the end of 2011 (>80% are researchers)
- Areas of research:
  - Life Sciences
  - Earth Sciences
  - Computer Applications
  - Computer Sciences. It comprises several research groups
    - Compiler group
    - Programming Models group
    - …

The CAOS group

- Computer Architecture/Operating System (CAOS) (www.bsc.es/caos)
  - 16 people
  - Research lines and collaborations:
    - HPC systems: IBM
    - Networking systems: Sun Microsystems
    - Real-Time systems
      - FP7 MERASA
      - FP7 PROARTIS
      - FP7 parMERASA
      - VeTeSS ARTEMIS Project
      - Projects with ESA

Agenda

- BSC and Background of the CAOS team
- **Project Objectives and Motivation**
- Execution infrastructure
- Experiments and results
- Conclusions and Future work

Introduction

- Critical Real-time Embedded Systems (CRTESs) or hard real-time systems are in everyday life



- Some of the main requirements of hard real-time systems
  - Functional correctness (like any other computing system)
  - **Timing correctness**

esa

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Requirements

- Increasingly higher functional value to keep competitive edge
- CRTEs require **increasing computational power**
  - More and more functions required
  - Functions are becoming more complex
  - Examples:
    - Automotive: (5x-10x) driver assistance in steer-by-wire, brake-by-wire, etc
    - Aerospace: (>4x) Unmanned Aerial Vehicles
    - Space: computational-intensive value-added on-board functions
- Within bounded development and production costs

Achieving High Performance

- Such required performance could be achieved by designing complex single-core processors
  - Longer pipelines
  - Out of order execution
  - Higher clock frequency
- These solutions are not feasible in CRTEs
  - Hard to derive WCET
    - Too complex due to their non deterministic run-time behaviors
    - Timing anomalies
  - High-energy requirements of such complex processors don't satisfy CRTE low-power constraints

Achieving High Performance

- Such required performance could be achieved by designing complex single-core processors
  - Longer pipelines
  - Out of order execution

Multi-core processors are considered the solution for some of these problems!

  - Too complex due to their non deterministic run-time behaviors
  - Timing anomalies
  - High-energy requirements of such complex processors don't satisfy CRTE low-power constraints

Multi-cores for Hard Real-Time Systems

- Pros:
  - Better performance per watt than single-core processors
  - Maintain simple core design
  - Enable co-hosting mixed-criticality applications
    - Hardware utilization is maximized, while cost, size, weight and power requirements are reduced.

Multi-cores for Hard Real-Time Systems

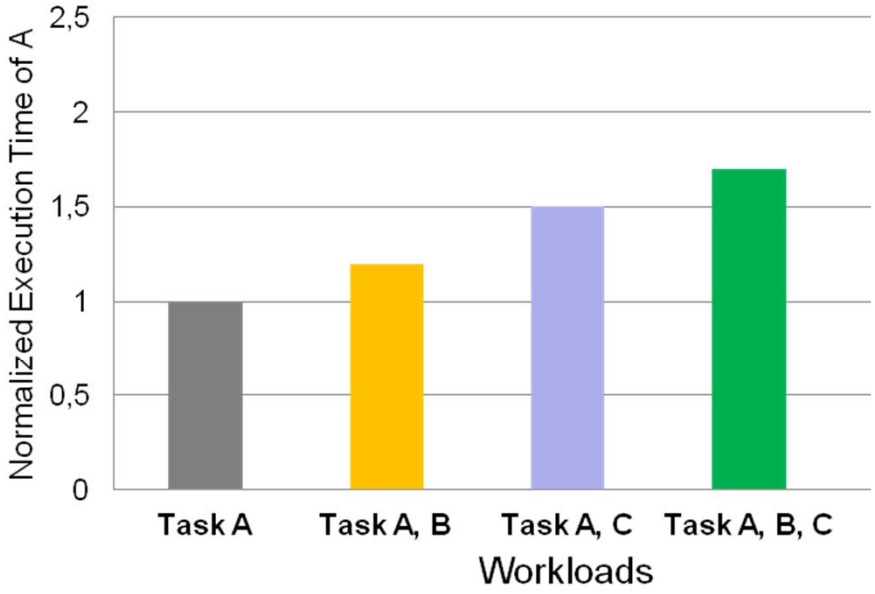
- Cons:
  - Require functional isolation
    - Prevent that one application corrupts the state of other applications;
      - Low-criticality applications must not affect high-criticality ones
    - Software isolation has been achieved within the space domain through the use of hypervisors [1]
  - Harder to time analyze w.r.t. single-core chips
    - It is hard to provide a safe and tight WCET estimation in multi-cores
    - Because of inter-task interferences!

[1] ESA contract 4200023100, System Impact of Distributed Multi-core Systems

Inter-task interferences

- Appear when several tasks that share a hardware resource want to access to it at the same time, so an arbitration stage is required

- The Execution time, and hence the WCET, of a task in a multi-core depends on the co-running tasks!

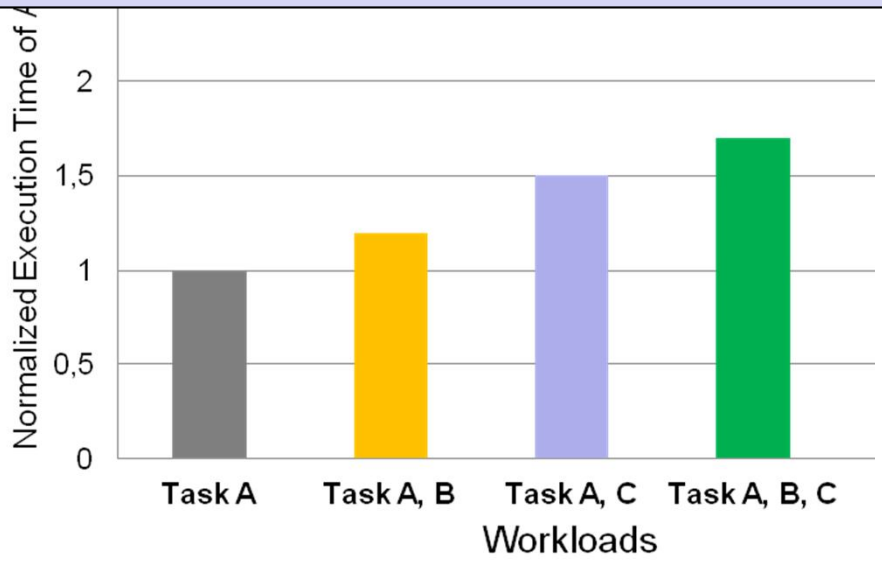


Inter-task interferences

- Appear when several tasks that share a hardware resource want to access to it at the same time, so an arbitration stage is required

- The Execution time, and hence the WCET, of a task in a

WCET depends on the workload!!!



esa

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

State of the art (hardware proposals)

- Several proposals developed to ease the computation of WCET estimates for CMPs (MERASA, ACROSS, GENESYS, PRET, TTA, PROATIS, PREDATOR ...)
  - Either by means of isolating interactions between tasks or
  - upper-bounding the maximum interaction between tasks (MERASA)
  - NPI activity between BSC and ESA.
    - Title: Architectural solutions for the timing predictability of next-generation multi-core processors
    - Objective: Creating hardware support for taking inter-task interferences into account when computing WCET estimations for the NGMP (simulator)
    - People: Javier Jalle, Francisco J. Cazorla, Eduardo Quiñones, Luca Fossati, Marco Zulianello

State of the art (hardware proposals)

- Current multicores do not implement those hw features
  - It will take several years to be implemented
  - Industry cannot benefit nowadays from those proposals
  - **COTS multicore processors have to be used instead**

Timine Analysis: multicore processors

- Static Timing Analysis (STA) has several problems when used in industrial-size applications [1]
  - Hardware analysability, Computational tractability , Information gathering
- Measurement-based Timing Analysis (MBTA) approaches, or hybrid approaches, have emerged
  - MBTA for single-threaded architectures
    - WCET estimation = longest observed execution time (LOET) x safety margin

[1] *"On the Industrial Fitness of WCET Analysis". Mezzeti, Vardanega. WCET Worskshop 2011.*

Timine Analysis: multicore processors

- In multicores the effect of inter-task interferences affect the computation of the safety margin
  - WCET estimation = longest observed execution time (LOET) x safety margin x **margin for inter-task interferences**

- **Similarly, contention-aware scheduling algorithms has to be designed**

Objective of this project

- Define and develop a *benchmark suite…*
  - able to mimic the CPU behavior of reference ESA applications,
  - suitable to exercise the new NGMP multicore processor
  - capable of generating different inter-task interference scenarios that may arise in the NGMP processor
- The ultimate goal of the benchmark suite is to provide a methodology to measure the real-time capabilities of multi-core architectures and, in particular, of the NGMP.

Agenda

- BSC and Background of the CAOS team
- Project Objectives and Motivation
- **Execution infrastructure**
- Experiments and results
- Conclusions and Future work

The board: ML510

- NGMP with 4 cores
- Private per-core resources
  - Core, 16KB Data and instruction caches
- Shared resources
  - The bus to the L2, 256KB L2, and the memory bandwidth (memory controller)
  - I/O resources are also shared but are not considered in this project
- DDR2 interface runs at 140 MHz
- NGMP frequency: 70 MHz

The NGMP

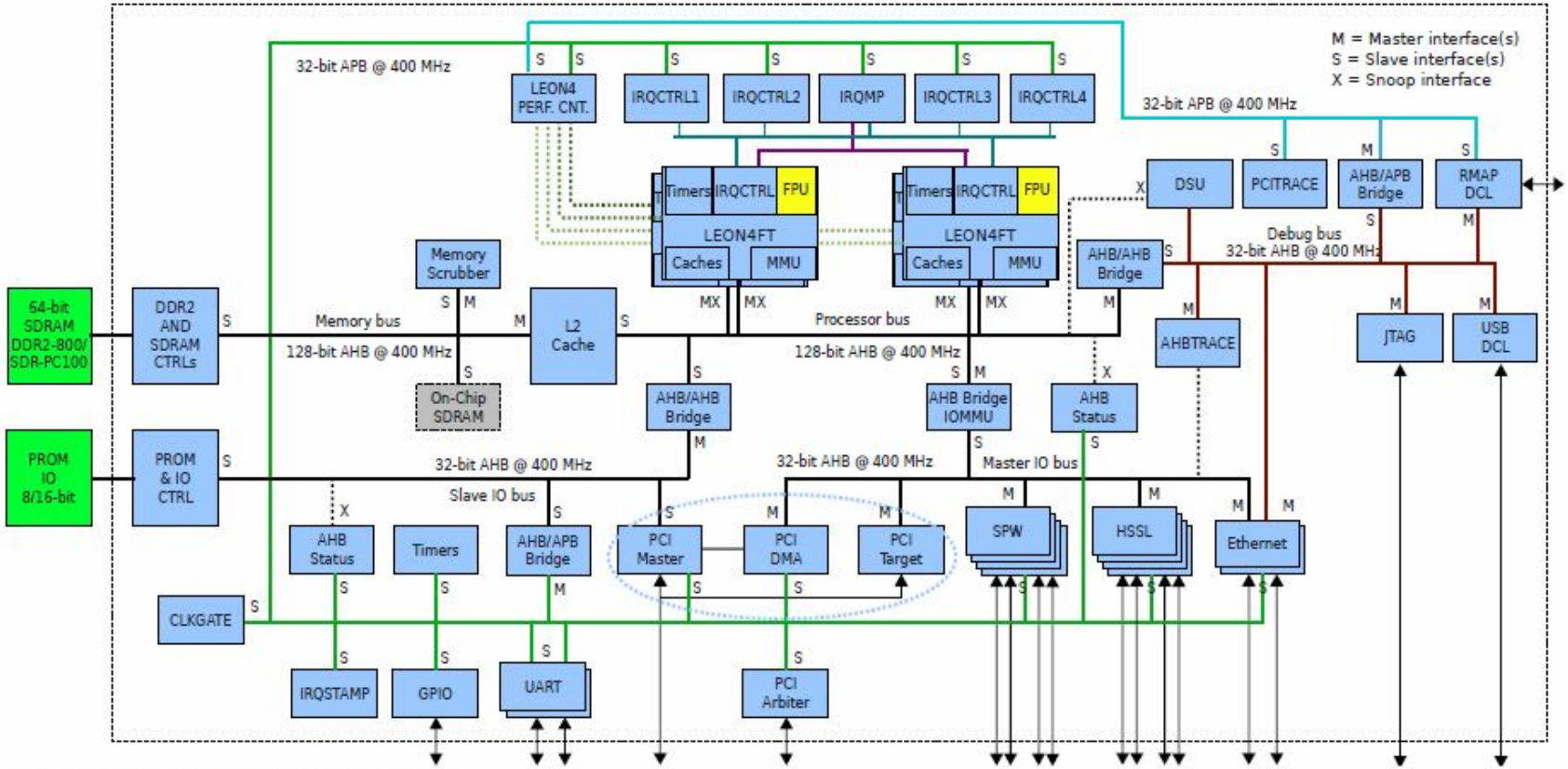


Figure 1: NGMP Block Diagram

http://microelectronics.esa.int/ngmp/LEON4-NGMP-DRAFT-1-6-changebars.pdf

The NGMP

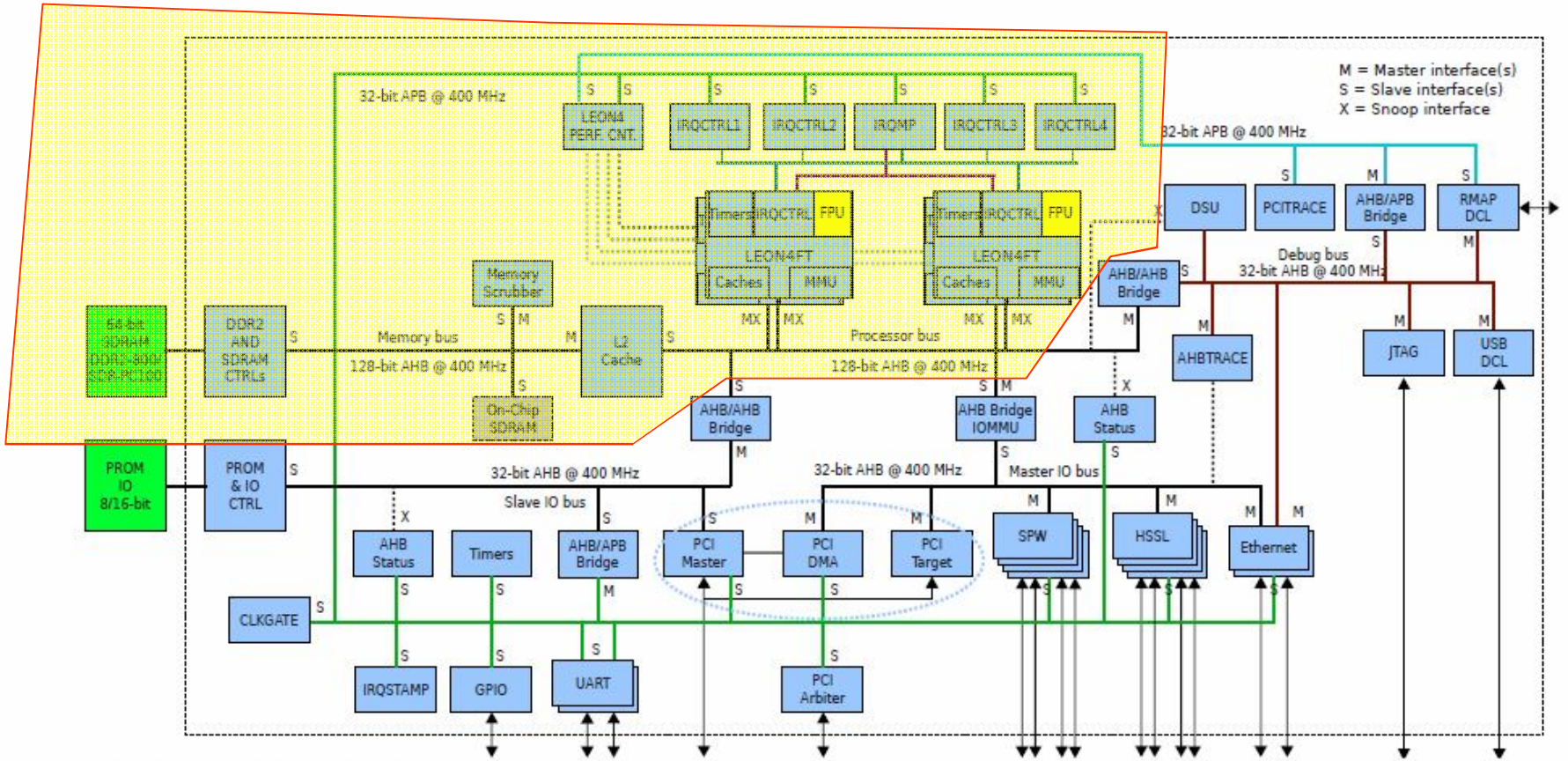


Figure 1: NGMP Block Diagram

http://microelectronics.esa.int/ngmp/LEON4-NGMP-DRAFT-1-6-changebars.pdf

Target applications and metrics

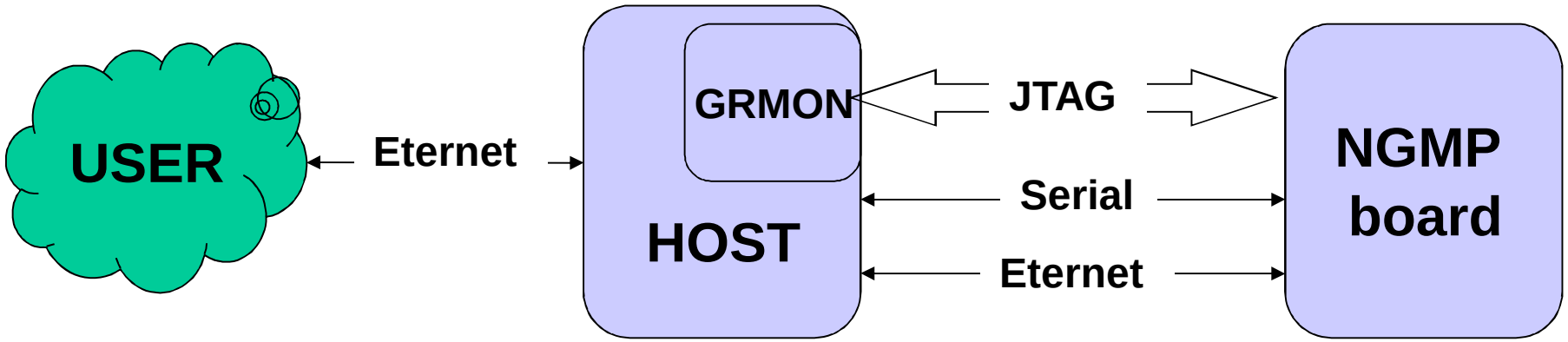
- The target workloads comprise both
  - Hard real-time applications or *control applications*
  - non-hard real-time applications or *payload applications*
- Metrics
  - Hard Real-Time Applications:
    - The sensitivity (jitter) of the HRT applications to the execution environment which include the other HRT and NRT applications.
    - Understanding and quantifying the impact of interferences on shared resources
  - Non-Hard Real-Time Applications:
    - The performance of the NRT applications
    - How much performance can be obtained by NHRT tasks without affecting (much) HRT apps?

Developing representative benchmarks

- Hard-real time applications
  - Micro benchmarks
    - Put high load on a single resource (L1, L2, cpu)
    - Used to measure the highest interference an application can suffer
    - No data sharing
  - Standard benchmarks: EEMBC, CoreMark
  - Mimicking applications
    - Applications that mimic main characteristic of some selected reference apps.
      - Instruction mix, memory access frequency, ..
- Non-hard real-time applications
  - Standard benchmarks: ParSec

The execution infrastructure

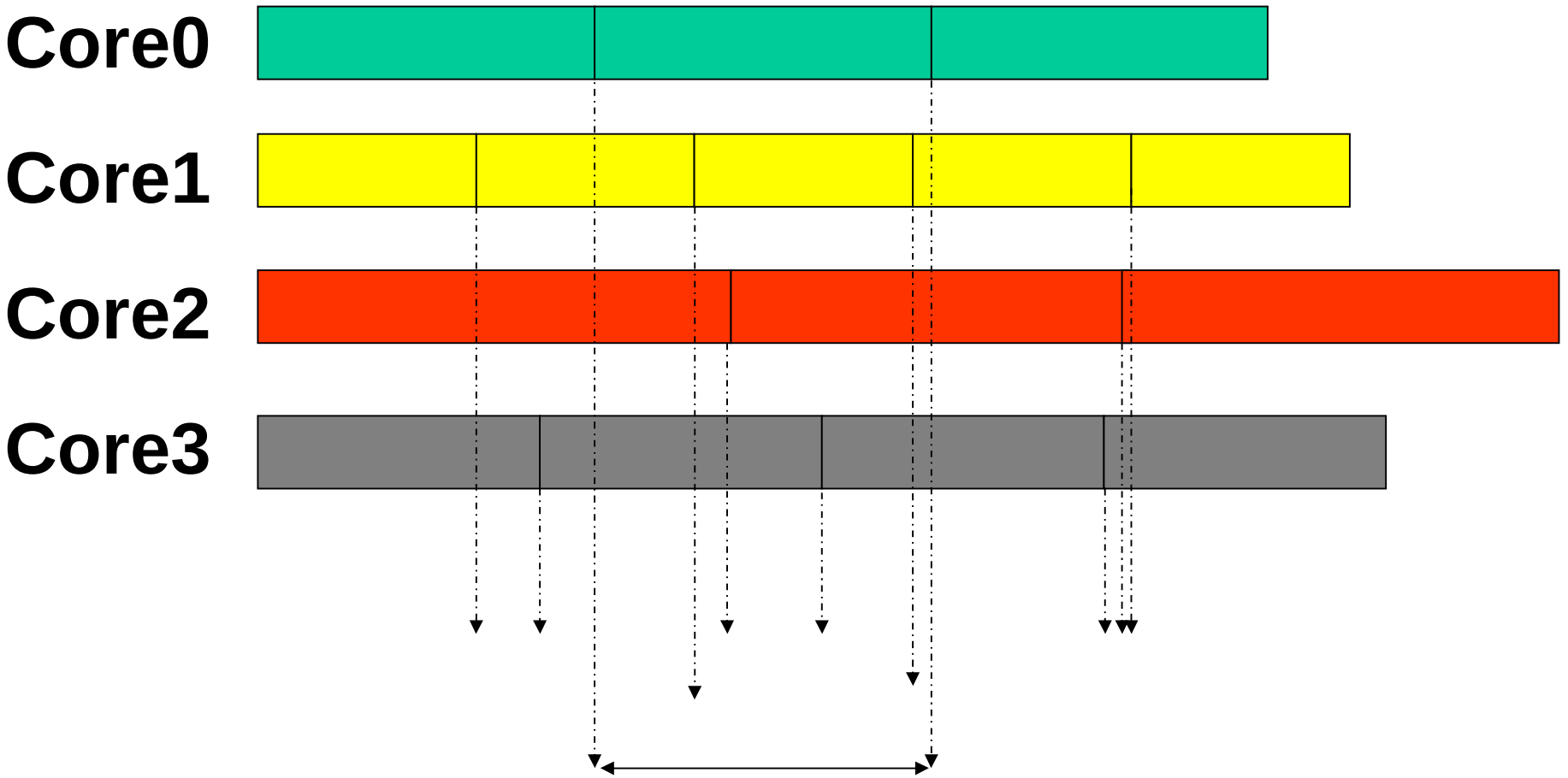
- We developed a set of scripts that allows (remotely)
  - Connecting to the host machine
  - Running experiments on NGMP (Linux and RTEMS)
  - Collecting results



System setup

- Host machine
  - Linux desktop
    - Compiling, linking toolchains
    - GRMon
  - Connected to NGMP board
    - JTAG (debug), preferred
    - Serial (standard)
    - Ethernet (standard, debug)
- NGMP board
  - Software
    - Linux
    - RTEMS
  - Connected to Host
    - debug to GRMon
    - Standard interface (serial, Ethernet)

Experiments (conceptually)



Execution time of Task 0 when running with a constant load on C1, C2 and C3

By comparing T0 Exec. Time w.r.t its run in isolation → inter-task interferences

Agenda

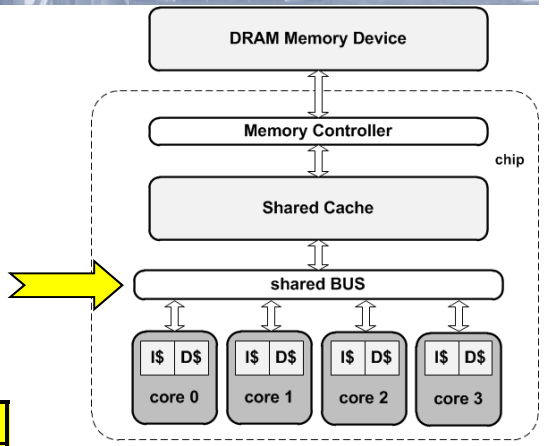
- BSC and Background of the CAOS team
- Project Objectives and Motivation
- Execution infrastructure
- **Experiments and results**
  - **Results on Linux**
  - **Results on RTEMS**
- Conclusions and Future work

(1) Experiments on Linux. Microbenchmarks

- *In all cases, as reference execution time we take the execution time of each benchmark when it runs in isolation*
- Run different sets of microbenchmarks and compute the execution time variation of each of them
  - quadruples: (L2 L1 ADD MULT), …
  - Pairs: (L1, L1), …

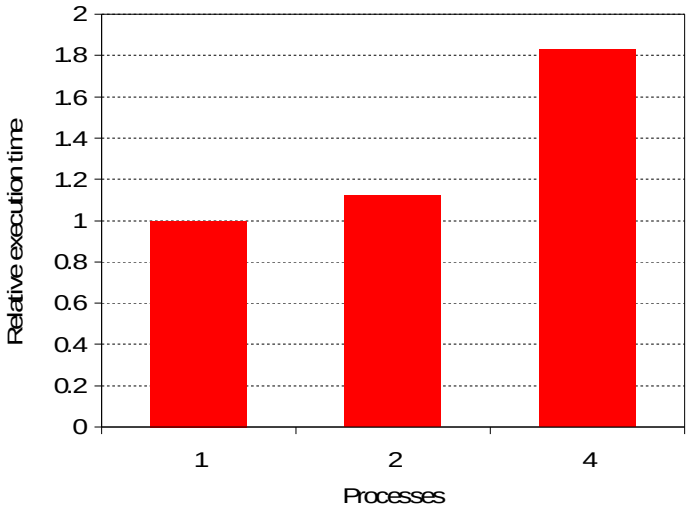
Results: Amba bus

- AMBA Bus that connects core to L2
  - 4 copies of $L2_{40KB}$ (less than 1/ 4 of the L2, bigger than DL1)
  - Each copy always misses in DL1 and hits in L2
  - N copies → interaction in Amba bus



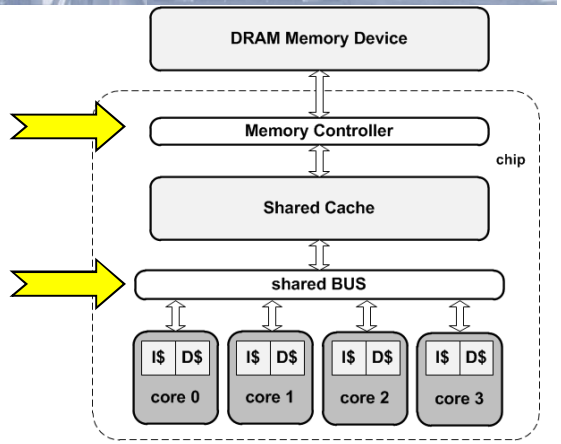
	percentage of L2 misses per Id			percentage of D misses per Id		
	1	2	4	1	2	4
L2-40	0.06%	0.12%	1.42%	95.05%	95.14%	98.39%

- Conclusions.
  - The worst delay due to sharing the AMBA Bus
    - 12% for 2 tasks
    - 83% for 4 tasks



Results: memory bandwidth

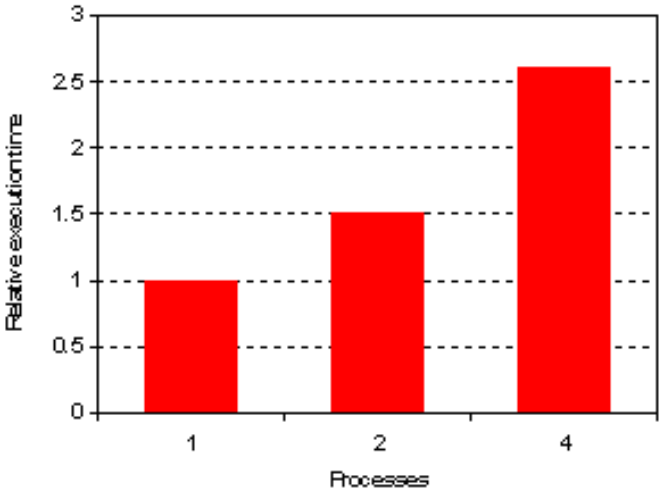
- Memory bandwidth
  - 4 copies of $L2_{miss}$ (memory)
  - All accesses in each copy always miss in L2
  - N copies → interaction in the memory controller & the memory BW **(and also in the AMBA bus)**



	percentage of L2 misses per Id			percentage of D misses per Id		
	1	2	4	1	2	4
L2-miss	100.00%	99.59%	98.51%	99.20%	98.89%	98.08%

- Conclusions*
  - Worst delay due to sharing memory bandwidth
    - 50% for 2 tasks
    - 2.5x for 4 tasks

*(In our FPGA implementation of the NGMP the ratio core_frequency/memory frequency is lower than in reality)

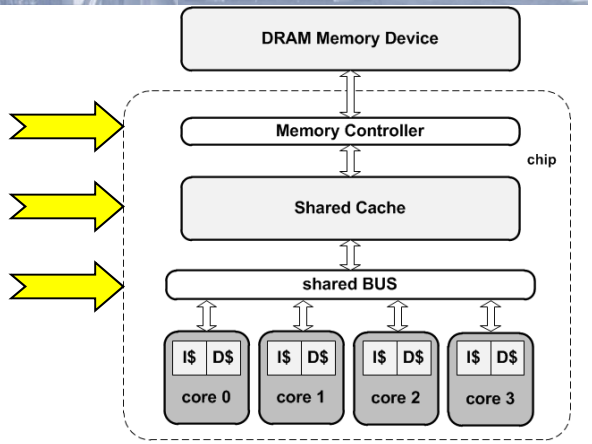


esa

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

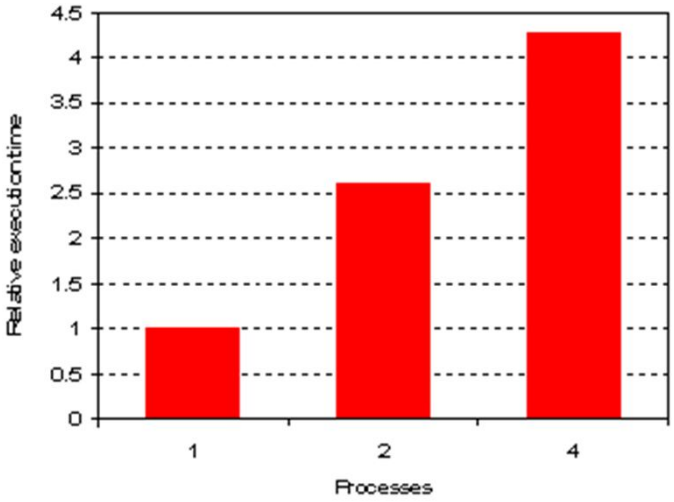
Results: L2 + memory

- Memory bandwidth + L2 cache
  - 4 copies of $L2_{200}$
  - Each copy will hit in L2 many times
  - N copies → interaction in L2 and memory and memory controller (also in the AMBA bus)



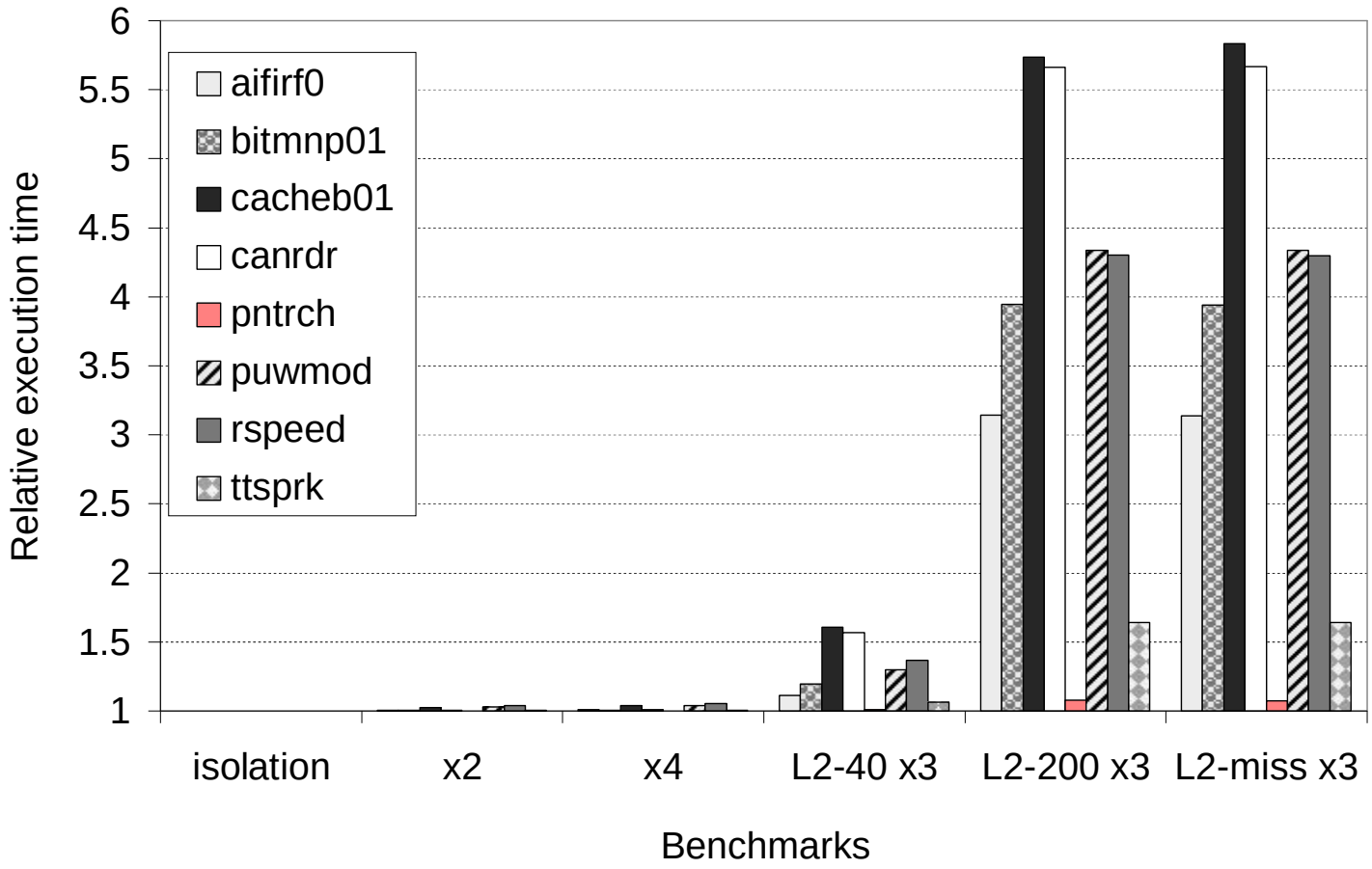
	L2 miss per load			DC miss per load		
	1	2	4	1	2	4
L2-200	31.45%	88.42%	98.55%	99.53%	98.98%	98.09%

- Conclusions
  - The worst delay due to sharing memory bandwidth and L2
    - 2.5x for 2 tasks and
    - 4.3x for 4 tasks



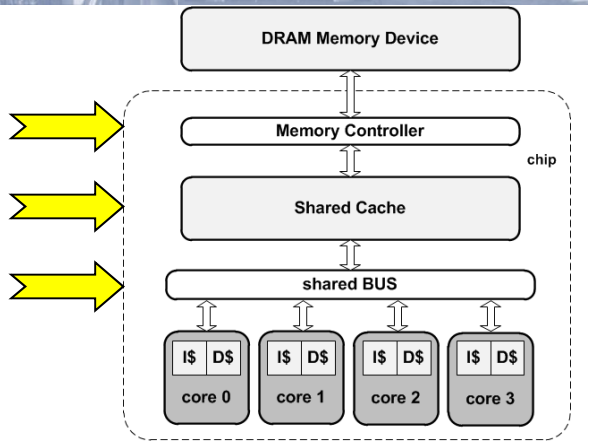
(1) Experiments on Linux. EEMBC

- We run several copies of EEMBC against one or several copies of micro-benchmarks

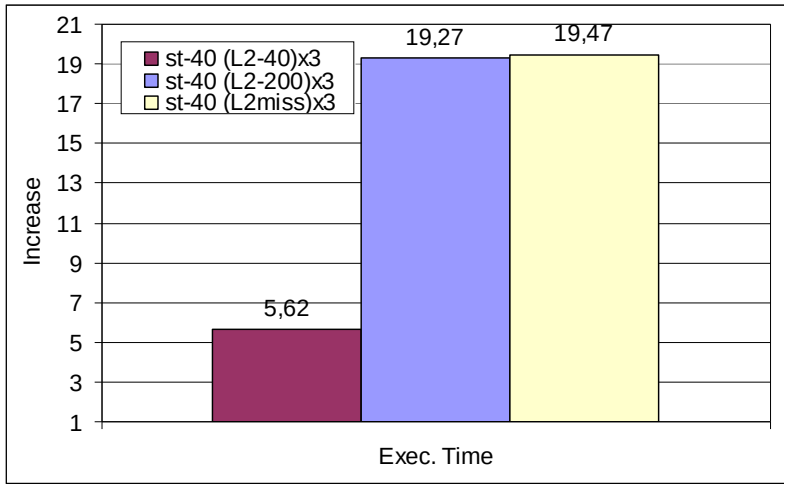


Results: store operations

- Store operations
  - $L2st_{40KB}$ (less than 1/ 4 of L2 cache)
  - Each copy always misses in DL1 and hits in L2

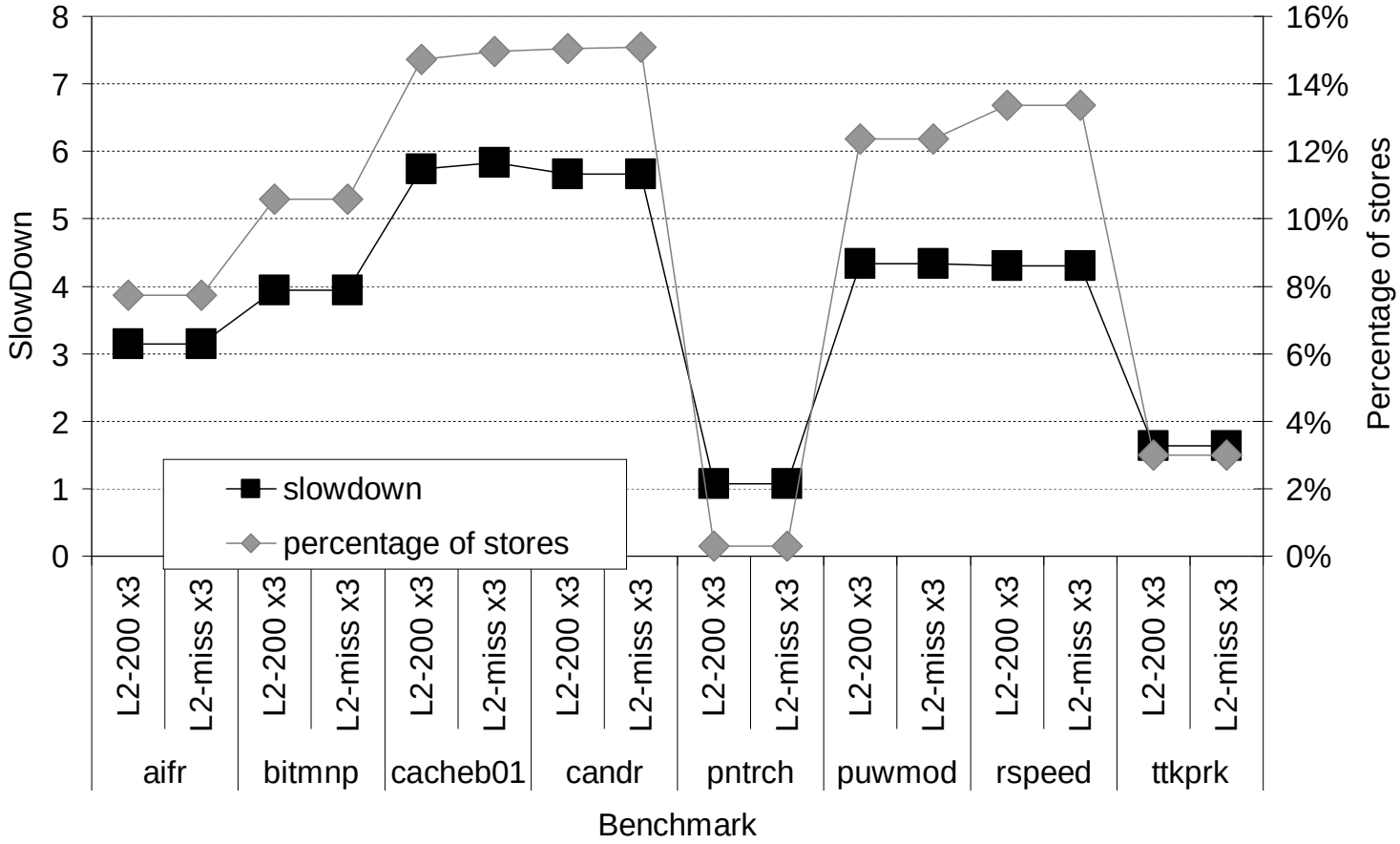


- Conclusions
  - $L2_{40}$ and $L2_{200}$ and $L2_{miss}$ cause a big slowdown, up to almost 20*x* on L2st
    - It has to access to the L2 and hence use the AMBA AHB Processor bus on every store operation
      - This make it quite sensitive to other benchmarks using the bus.



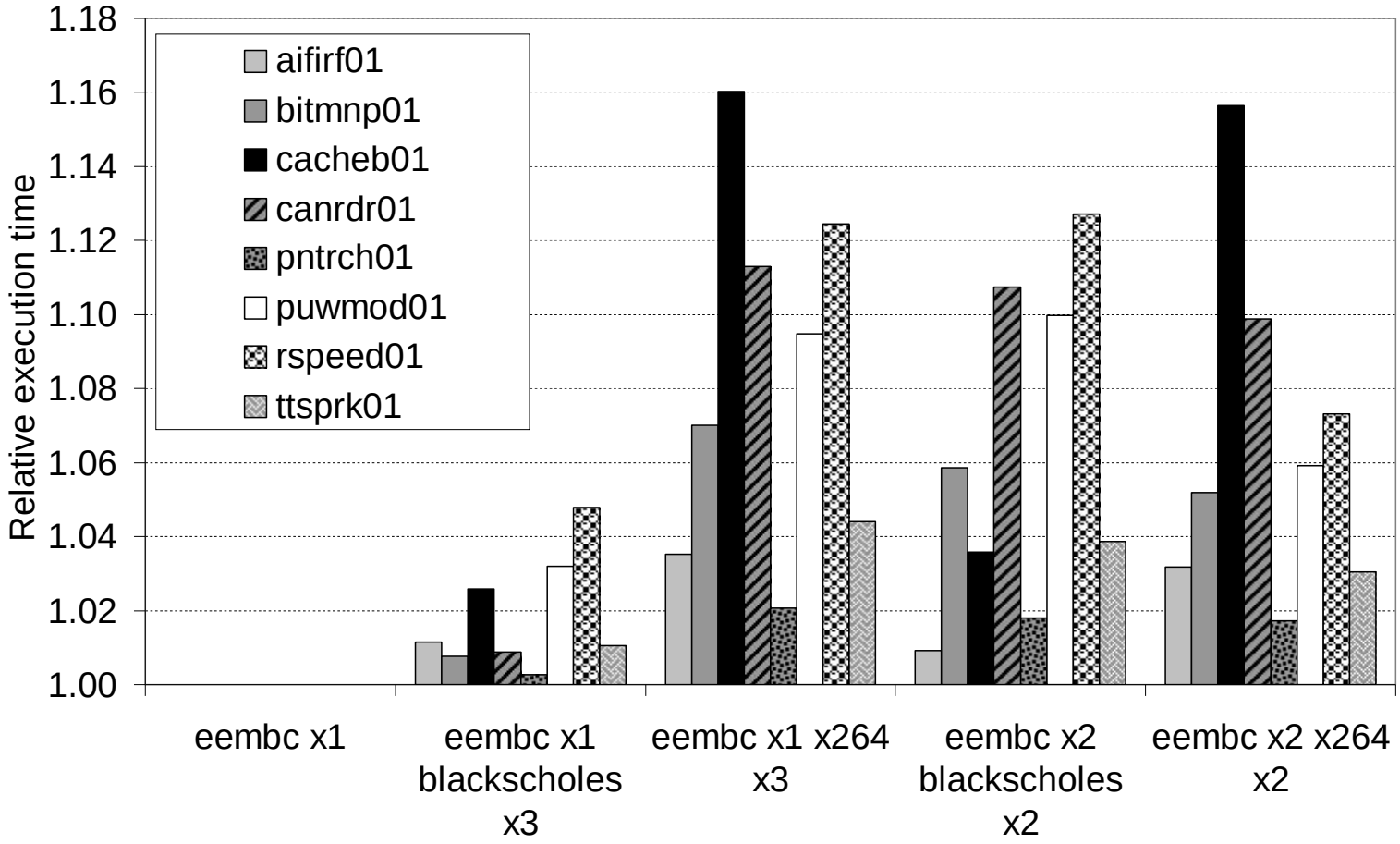
(1) Experiments on Linux. EEMBC

- Why this behavior?
  - High correlation between the density of store instructions and the slowdown.



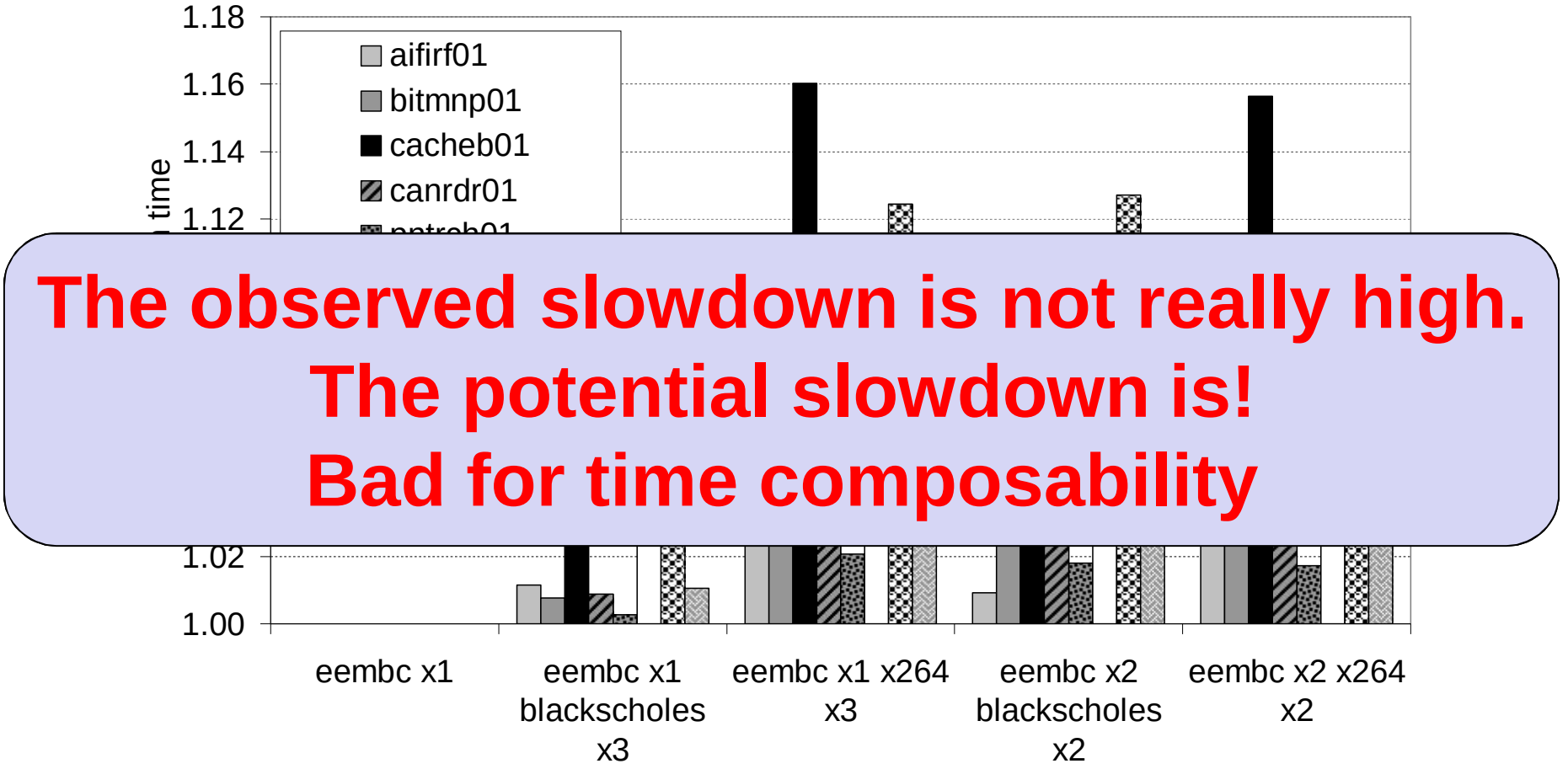
(1) Experiments on Linux. EEMBC

- EEMBC vs. PARSEC



(1) Experiments on Linux. EEMBC

- EEMBC vs. PARSEC

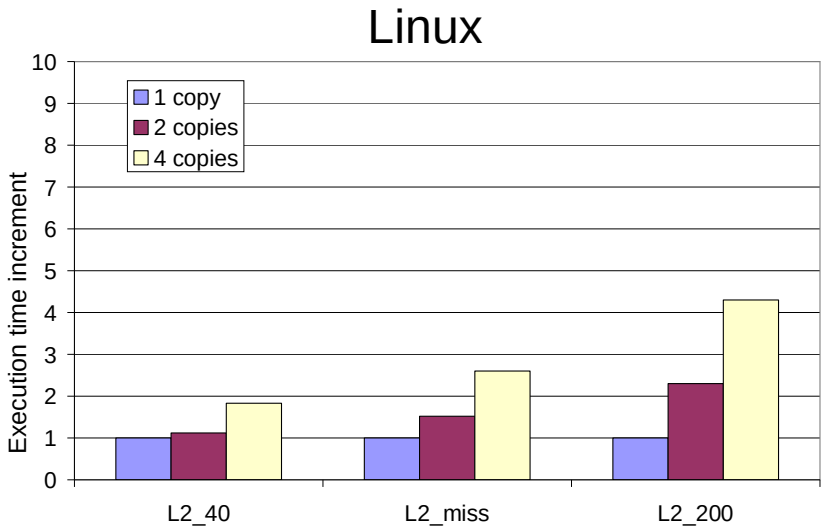


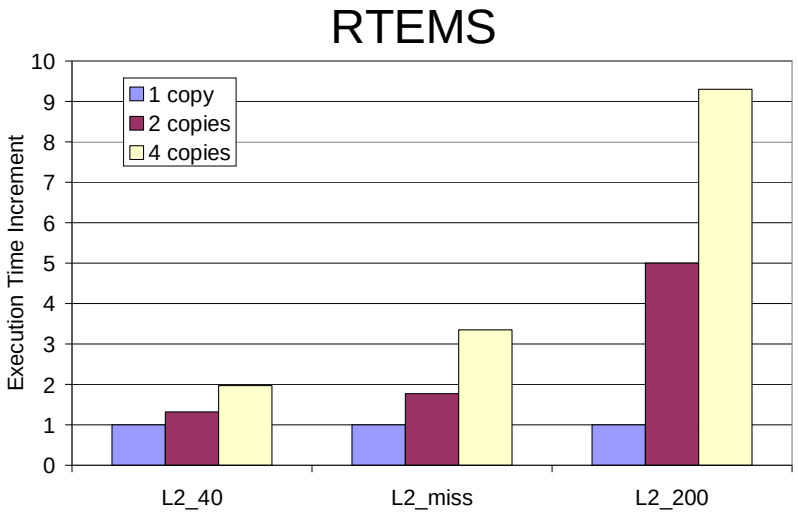
The observed slowdown is not really high.
The potential slowdown is!
Bad for time composability

esa

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación

(1) Experiments on RTEMS





- $L2_{40}$ → similar
- $L2_{miss}$ → similar
- $L2_{200}$ → bigger degradation on RTEMS than on Linux
  - On RTEMS, the baseline $L2_{200}$ run in isolation causes very few L2 misses, thanks to the small memory footprint of the operating system.
  - Single copy of $L2_{200}$: 0.31% miss rate on RTEMS 31% on Linux

Agenda

- BSC and Background of the CAOS team
- Project Objectives and Motivation
- Execution infrastructure
- Experiments and results
- **Conclusions and Future work**

Conclusions

- The lack of quantitative studies on inter-task interferences on real COTS multi-core processors, limit their use by industry
- We have developed a *benchmark suite* that is…
  - suitable to exercise the new multicore processors
  - capable to generate different inter-task interference scenarios
- The benchmark suite…
  - Provides accurate figures on the impact of interferences arisen in the main shared resources in the NGMP under both Linux and RTEMS.
  - Represents a first step towards providing effective interference-aware measurement-based WCET estimation and scheduling for the NGMP.

Conclusions: Quantitative

- We have presented initial results about the effect of inter-task interferences on time predictability for the NGMP.
- Worst observed behaviors are the following:
  - AMBA bus effect. Up to 83% (95%) for 4 cores
  - Memory bandwidth effect*. Up to 2.6x (3.4x) for 4 cores
  - L2 cache + memory BW effect*. Up to 4.3x (9x) for 4 cores
- Clear correlation between stores density and slowdown suffered by EEBMC

Conclusions: Qualitative

- Integrated Architectures
  - Guarantee that there is no interaction between the different functions sharing the resources to contain verification costs
  - Functional level: functional isolation
    - A bug/misbehavior in a function does not affect the others
  - Timing level: timing isolation
    - Timing behavior of a task is not affected by the others
    - Timing composability: timing behavior of an individual component does not change by the composition, i.e. composing the system
      - Alleviates system integration cost

Conclusions: Qualitative

- Time Composability in the NGMP:
  - The main software features affecting time composability are
    - (1) the percentage of store instructions and
    - (2) whether its footprint fits in the first level data cache

Corollary 1

- For application developers
  - Reduce the number of stores
  - Obviously, this is intrinsic to the functionality of the application and hence it can be difficult to change it
- Otherwise, in order to ensure time composability: play with scheduling
  - Store-intensive applications have to be scheduled in isolation or
  - any other application that may run on the other cores fits their data cache so they do not introduce traffic in the AHB bus

Corollary 2

- A write-back policy for the L1 data cache will significantly reduce the overhead of inter-task interferences.

- This will introduce several challenges

  - Implementation of consistency protocol, as MESI/MOESI or directory-based protocols will be needed.

    - Some data for which only one copy would exist in the system, located indeed in the L1 cache

  - NGMP features error detection only in the L1 cache.

    - To maintain adequate protection from errors (frequent in space, the target environment for the NGMP) error correction schemes would have to be implemented in the L1 cache,

    - Increasing the latency of read/write operations in such cache and reducing the maximum frequency of the processor

Future Work

- Hardware support for isolation:
  - Exploring hardware support for timing isolation: In the documentation of the NGMP, some features are explained to provide isolation in the L2 cache between threads.
- Explore the I/O path
  - We have explore the *path* to memory from the processor: data cache, main AMBA AHB processor and memory buses, L2, memory.
  - In addition to this main path, there are other paths that use other AMBA APB buses in the NGMP, to handle I/O traffic.

Future Work

- Scheduling:
  - In the literature there are many works dealing with scheduling algorithms for multicore processors.
  - Common assumption: threads receive an even part of the resources
    - Threads receive 1/N of the resources
    - WCET is independent of the workload
  - From the results of this study it is clear that this assumption cannot be done in the NGMP.
    - In fact, the characteristics of each thread determines the percentage of the shared resources, mainly AMBA bus and L2, they receive → its WCET

Multicore OS Benchmark

Francisco J. Cazorla

Mikel Fernandez

Roberto Gioiosa

Eduardo Quiñones

Marco Zulianello (TO)

Luca Fossati

Francisco J. Cazorla
(francisco.cazorla@bsc.es)
PhD. Researcher and Director of the CAOS group at BSC
(www.bsc.es/caos)
Software Systems Division & Data Systems Division
Final Presentation Days
25th April 2012, ESTEC