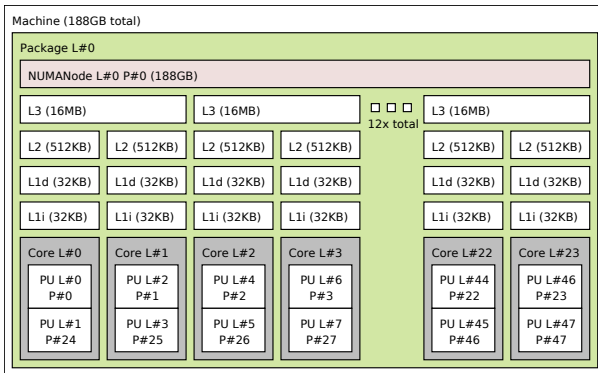






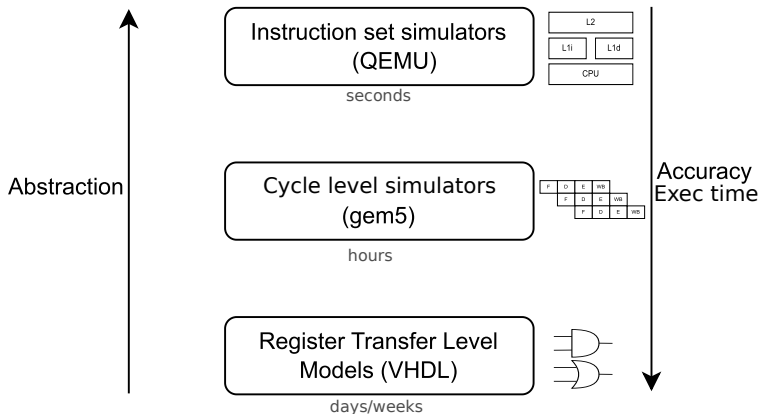
# Current multi-core systems



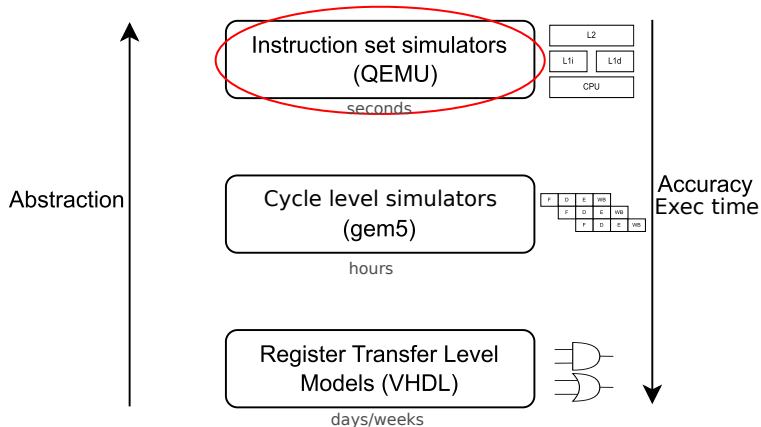
Dell PowerEdge R6515: 48 CPUs



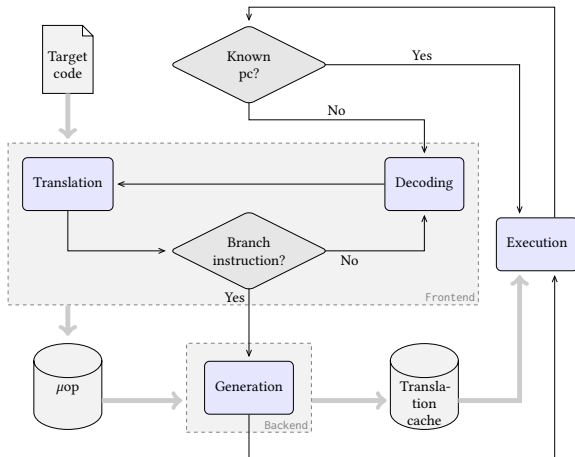
# Simulation: different levels of abstraction



# Simulation: different levels of abstraction





# Instruction set simulation technology: Dynamic Binary Translation



- Translation Block (TB): block that ends with a branch
- Support for parallel execution  
→ **multi-core** systems

# Instruction set simulation technology: Dynamic Binary Translation

1 IN: **Target instructions**   
2 0x000000000000325d6: **add** a5,a5,a3  
3 OP: **Intermediate representation**   
4 ...  
5 – 000000000000325d6  
6 **add\_i64** a5,a5,a3  
7 ...  
8 OUT: **Host instructions** **x86**  
9 – guest addr 0x000000000000325d6  
10 0x7f5204000f93: **movq** 0x68(%rbp), %r12  
11 0x7f5204000f97: **addq** %r12, %rbx  
12 0x7f5204000f9a: **movq** %rbx, 0x78(%rbp)  
13 ...  
14 ...

- Translation Block (TB): block that ends with a branch
- Support for parallel execution  
→ **multi-core** systems

Translation process of a single **add** instruction



# Instrumentation

## What?

⇒ Evaluation and analysis of programs

## How?

⇒ Run time analysis

⇒ Production of traces

*Details of what happens during execution: ex memory accesses*

```
-- guest addr 0x0000000000026546
0x7f5204079c9e: 49 ff cc          decq    %r12
0x7f5204079ca1: 4d 63 e4          movslq  %r12d, %r12
0x7f5204079ca4: 4c 89 65 68       movq    %r12, 0x68(%rbp)
-- guest addr 0x000000000002654a
0x7f5204079ca8: 48 83 c3 04       addq    $4, %rbx
0x7f5204079cac: 44 89 23          movl    %r12d, (%rbx)
```

## Why?

⇒ Adding new features, improvement of the simulation (accuracy)

*For example a cache...*



# Problem Statement Outline

*Dynamic Binary Translation **speed** and **accuracy** trade-offs: investigating parallel scalability and cache simulation*

## DBT Simulation Speed

- Scalability of DBT parallel execution on multi-core host
- Relying on host configuration to improve simulation time

## DBT Simulation Accuracy

- Memory hierarchy model related to DBT
- General solutions to enhance accuracy without degrading performance

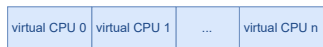






# QEMU: Multi-Threaded Tiny Code Generator (MTTCG)

## QEMU Processes



## Host CPUs



...



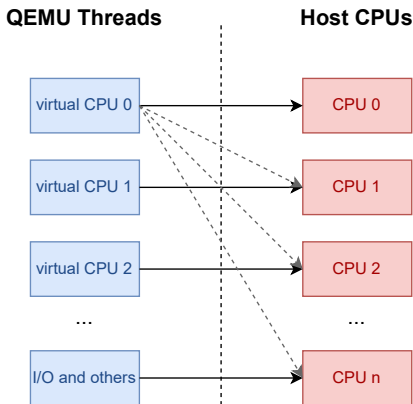
...



TCG principle (round-robin)

- Tiny Code Generator (TCG): cross compilation tool
- **Before 2015:** single-threaded simulation
- Execution of the virtual CPUs on one host CPU

# QEMU: Multi-Threaded Tiny Code Generator (MTTCG)



MTTCG principle

- Tiny Code Generator (TCG): cross compilation tool
- **Since 2015:** multi-threaded simulation <sup>a b</sup>
- Each virtual CPU executes on a separate thread

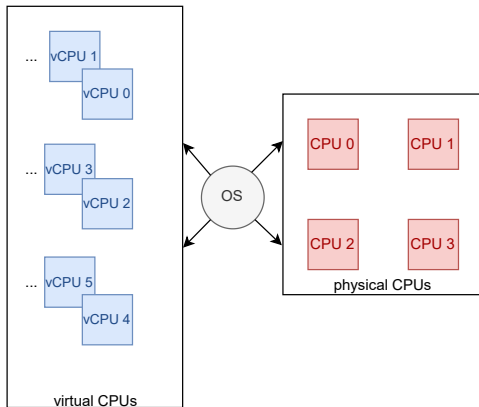
<sup>a</sup> Alvise Rigo, Alexander Spyridakis, and Daniel Raho. Atomic instruction translation towards a multi-threaded qemu. 2016

<sup>b</sup> Emilio G. Cota, Paolo Bonzini, Alex Bennée, and Luca P. Carloni. Cross-isa machine emulation for multicores. 2017





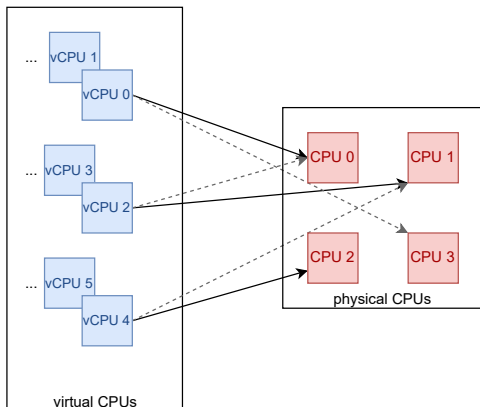
# To pin or not to pin: What is pinning? <sup>1</sup>



Operating System decides  
where to assign the vCPUs

<sup>1</sup> collaboration with Saverio Miroddi

# To pin or not to pin: What is pinning? <sup>1</sup>



Force each virtual CPU to run on a chosen physical CPU

**Goal:** Scalability study

<sup>1</sup> collaboration with Saverio Miroddi

# Implementation in QEMU

## Linux interfaces

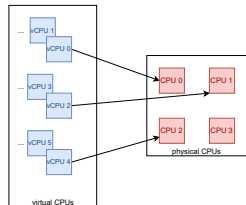
`cpu_set_t` and `pthread_setaffinity_np`

## Added command line options

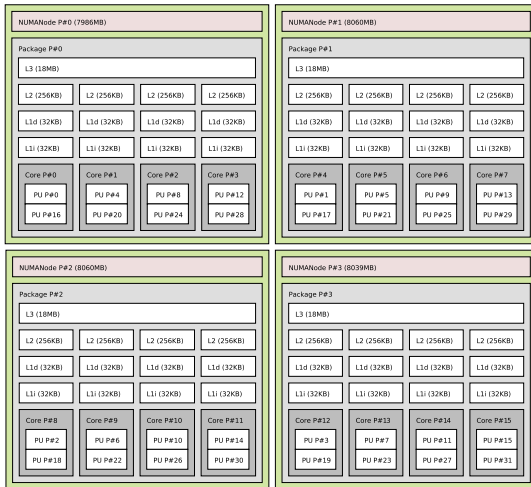
```
qemu-system-riscv64 \  
-smp $total_vcpus,cores=$vcores,sockets=$vsockets,threads=$vthreads \  
-vcpu vcpunum=$vcpu_number,affinity=$host_physical_cpu_number \  
-vcpu vcpunum=$vcpu_number,affinity=$host_physical_cpu_number \  
...
```

## Example

```
qemu-system-riscv64 -smp 6 \  
-vcpu vcpunum=0,affinity=0 -vcpu vcpunum=1,affinity=0 \  
-vcpu vcpunum=2,affinity=1 -vcpu vcpunum=3,affinity=1 \  
-vcpu vcpunum=4,affinity=2 -vcpu vcpunum=5,affinity=2
```



# How to choose the affinity?



- Following the NUMA (Non Uniform Memory Access) architecture of the host
- **Simultaneous MultiThreading (SMT):** number of host hardware threads (harts) per core, 1 or 2

Dell PowerEdge R910 (1stopo)

## Methodology: Parameters

- Simultaneous MultiThreading enabled or not: 16 or 32 CPUs,
- Number of virtual CPUs  $n_c$ : {1, 2, 4, 8, 16, 24, 32, 48, 64, 96, 128},
- Number of threads  $n_t$  of the PARSEC benchmarks: {1, 2, 4, 8, 16, 24, 32, 48, 64, 96, 128},
- PARSEC threads affinity,
- Pinning QEMU virtual CPUs to physical CPUs,
- Isolcpus to strictly separate the physical CPUs allocation between QEMU virtual CPU threads and the kernel threads.



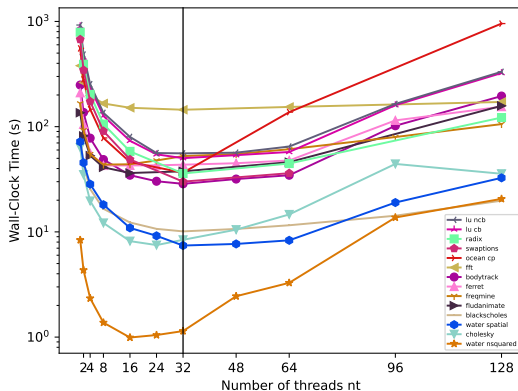
# Environment

QEMU	Multi-core Benchmarks	Targets	Host
full-system + Busybear Linux	PARSEC <i>LARGE</i>	RISC-V & ARM	x86 Dell PowerEdge R910 <b>32 CPUs</b>



# Scalability without pinning

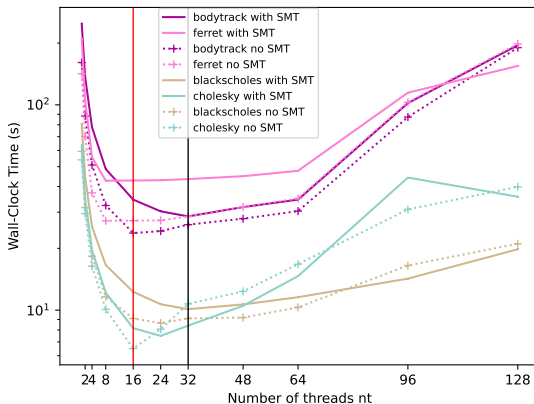
$n_c$  = number of virtual CPUs,  $n_t$  = number of threads for the PARSEC application



**QEMU has a good scalability**

Full execution time in QEMU RISC-V  $n_c = n_t$  **without pinning**

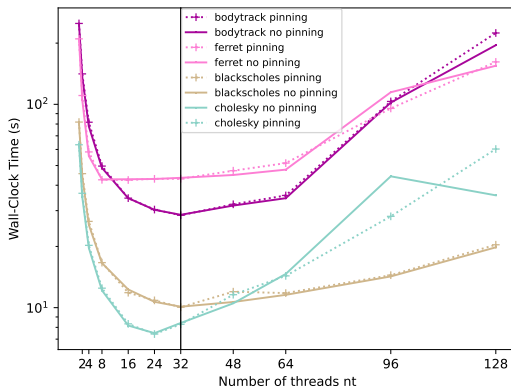
# Scalability with/without SMT (without pinning)



Comparison of full execution time in QEMU RISC-V without pinning with  $n_c = n_t$  for the host machine with and without SMT

# Comparison to pin or not to pin

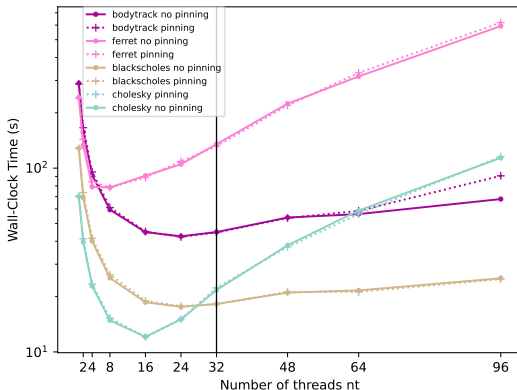
$n_c$  = number of virtual CPUs,  $n_t$  = number of threads for the PARSEC application



Comparison of full execution time in QEMU RISC-V  $n_c = n_t$  **without pinning** and **with pinning**

# Is Pinning Helpful?

- Linux perf tool
  - *CPU migrations*: less when pinning
  - *L1 data cache misses*: no significant differences
- ARM: same observations



Comparison of full execution time in QEMU ARM **without pinning** and **with pinning**

## Is Pinning Helpful?

- Linux perf tool
  - *CPU migrations*: less when pinning
  - *L1 data cache misses*: no significant differences
- ARM: same observations

### Conclusion:

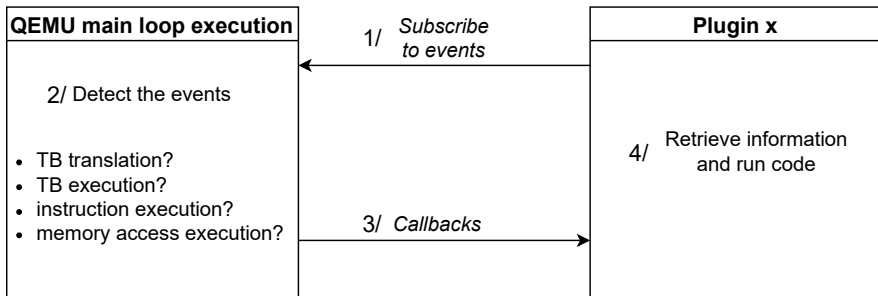
- Pinning QEMU virtual CPUs not helpful
- Cannot do better than Linux scheduler





# QEMU TCG Plugins

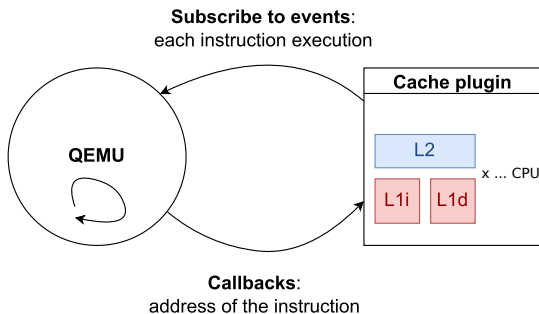
⇒ Code instrumentation **easily** and **efficiently**



Simplified representation of the QEMU TCG plugins mechanism



# QEMU TCG Plugins: existing cache<sup>2</sup> plugin



Simplified representation of the QEMU **cache** TCG plugin mechanism

<sup>2</sup>Mahmoud Mandour. Cache modelling tcg plugin, 2021



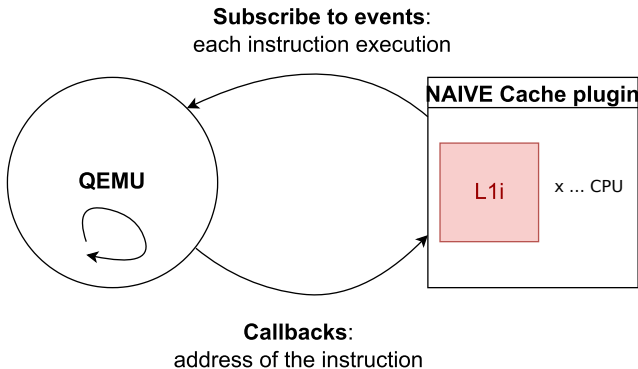
## Initial Intuition

- DBT TB per TB execution principle
- In a TB, all instructions are consecutive in memory  
⇒ **Know which instructions will hit and which might miss**

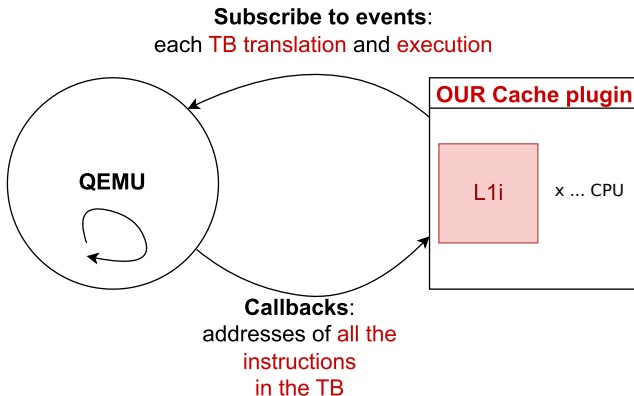
0x800fa7bc:	1141	addi	sp,sp,-16	← possible miss
0x800fa7be:	e022	sd	s0,0(sp)	← hit!
0x800fa7c0:	e406	sd	ra,8(sp)	← possible miss
0x800fa7c2:	0800	addi	s0,sp,16	← hit!
0x800fa7c4:	00dbc797	auipc	a5,14401536	← hit!
0x800fa7c8:	2347a783	lw	a5,564(a5)	← hit!
0x800fa7cc:	eb95	bnez	a5,52	← hit!

Example of static hit/miss decision within a TB

# Implementation in QEMU: Naive cache plugin



# Implementation in QEMU: Our cache plugin



# Implementation in QEMU

## QEMU TCG Plugins: callbacks

# NAIVE solution

CALL\_PLUGIN\_ins(...)

0x7f1ffbe5693e : sb s0,1470(a5)

CALL\_PLUGIN\_ins(...)

0x7f1ffbe56942 : ld s0,0(sp)

CALL\_PLUGIN\_ins(...)

0x7f1ffbe56944 : addi sp,sp,16

CALL\_PLUGIN\_ins(...)

0x7f1ffbe56946 : ret

# OUR solution

CALL\_PLUGIN\_tb(...)

0x7f1ffbe5693e : sb s0,1470(a5)

0x7f1ffbe56942 : ld s0,0(sp)

0x7f1ffbe56944 : addi sp,sp,16

0x7f1ffbe56946 : ret



## Error in Counting Instructions: 1st Problem

### Stopped TB execution: page-fault problem 2/4

# Insns in TB

**CALL\_PLUGIN\_tb(...)** # 9 insns counted

0x7f1ffbe5692c : auipc a5,237568

0x7f1ffbe56930 : ld a5,-612(a5)

0x7f1ffbe56934 : sb s0,0(a5)

0x7f1ffbe56938 : ld ra,8(sp)

0x7f1ffbe5693a : auipc a5,270336

0x7f1ffbe5693e : sb s0,1470(a5)

0x7f1ffbe56942 : ld s0,0(sp)

0x7f1ffbe56944 : addi sp,sp,16

0x7f1ffbe56946 : ret

# New TB after return from handler

**CALL\_PLUGIN\_tb(...)** # 7 insns counted

0x7f1ffbe56934 : sb s0,0(a5)

0x7f1ffbe56938 : ld ra,8(sp)

0x7f1ffbe5693a : auipc a5,270336

0x7f1ffbe5693e : sb s0,1470(a5)

0x7f1ffbe56942 : ld s0,0(sp)

0x7f1ffbe56944 : addi sp,sp,16

0x7f1ffbe56946 : ret

Assumption that all instructions  
in TB are executed

⇒ **Not in practice**



## Error in Counting Instructions: 1st Problem

### Stopped TB execution: page-fault problem 3/4

# Insns in TB

**CALL\_PLUGIN\_tb(...)** # 9 insns counted

0x7f1ffbe5692c : auipc a5,237568

0x7f1ffbe56930 : ld a5,-612(a5)

0x7f1ffbe56934 : sb s0,0(a5)

0x7f1ffbe56938 : ld ra,8(sp)

0x7f1ffbe5693a : auipc a5,270336

0x7f1ffbe5693e : sb s0,1470(a5)

0x7f1ffbe56942 : ld s0,0(sp)

0x7f1ffbe56944 : addi sp,sp,16

0x7f1ffbe56946 : ret

# Executed insns until new page fault

0x7f1ffbe56934 : sb s0,0(a5)

0x7f1ffbe56938 : ld ra,8(sp)

0x7f1ffbe5693a : auipc a5,270336

0x7f1ffbe5693e : sb s0,1470(a5)

Assumption that all instructions  
in TB are executed  
⇒ **Not in practice**

# Error in Counting Instructions: 1st Problem

## Stopped TB execution: page-fault problem 4/4

# Insns in TB

**CALL\_PLUGIN\_tb(...)** # 9 insns counted

0x7f1ffbe5692c : auipc a5,237568

0x7f1ffbe56930 : ld a5,-612(a5)

0x7f1ffbe56934 : sb s0,0(a5)

0x7f1ffbe56938 : ld ra,8(sp)

0x7f1ffbe5693a : auipc a5,270336

0x7f1ffbe5693e : sb s0,1470(a5)

0x7f1ffbe56942 : ld s0,0(sp)

0x7f1ffbe56944 : addi sp,sp,16

0x7f1ffbe56946 : ret

# New TB after return from handler

**CALL\_PLUGIN\_tb(...)** # 4 insns counted

0x7f1ffbe5693e : sb s0,1470(a5)

0x7f1ffbe56942 : ld s0,0(sp)

0x7f1ffbe56944 : addi sp,sp,16

0x7f1ffbe56946 : ret

Assumption that all instructions  
in TB are executed

⇒ **Not in practice**

*page-fault, wfi, pause*

Total instruction counted:

9+7+4=20, 11 wrongly counted

## Error in Counting Instructions: 2nd Problem

### Dependency on Simulator Runtime

#### Unexpected behavior

- Time dependency of the flow of executed target instructions  
**The faster the simulator, the lower the number of executed instructions for a given program**  
⇒ Only for programs running on top of Linux

#### Causes

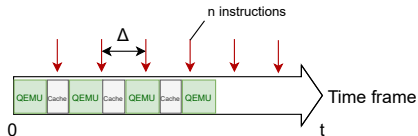
- Repeated occurrences of timer interrupts  
**The more interrupts, the more instructions counted in the cache statistics**

## Error in Counting Instructions: 2nd Problem

### Dependency on Simulator Runtime

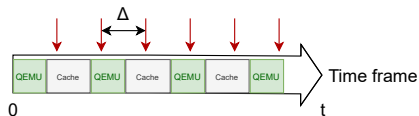
#### Unexpected behavior

- Time dependency of the flow of executed target instructions  
**The faster the simulator, the lower the number of executed instructions for a given program**  
⇒ Only for programs running on top of Linux



Total ins case 1 ( $N + 4n$ )

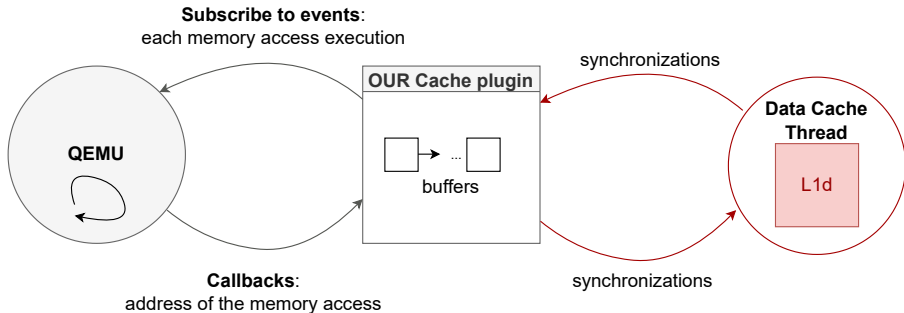
<



Total ins case 2 ( $N + 6n$ )

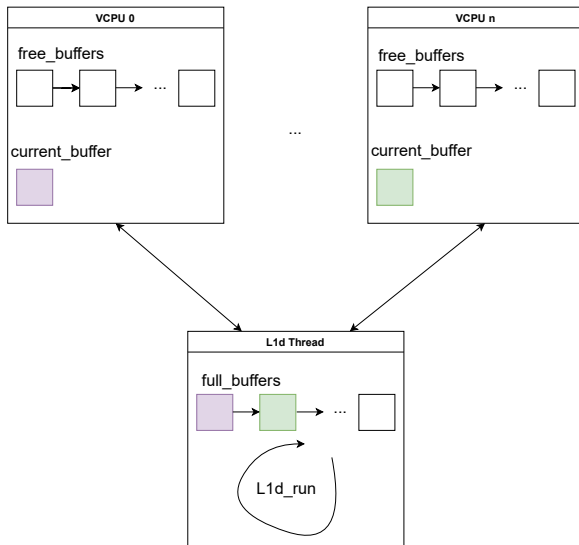


# QEMU TCG Plugins: A threaded execution



Simplified representation of the L1d implementation in **our cache plugin**

# Buffers synchronization



Simplified representation  
of the data thread  
interactions with the  
virtual CPUs

⇒ Adjustable size  
and number of buffers

Out-of-sync from  
QEMU execution





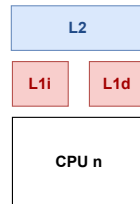
# L2 modeling

## Mitigations on the L1d threaded simulation

- Scalability of our model  
→ bottleneck with many CPUs
- Out-of-sync simulation  
→ validity of the data in L2

## L2 implementation

- Keep our L1i model
- Naive L1d simulation at each memory access execution





# Environment

QEMU	Mono-core Benchmarks	Multi-core Benchmarks	Target	Host
user-mode	PolyBench/C <i>MEDIUM</i>	PARSEC <i>LARGE</i>	RISC-V	x86 PowerEdge R6515 <b>128 CPUs</b>

## Why *user-mode* instead of *full-system*?

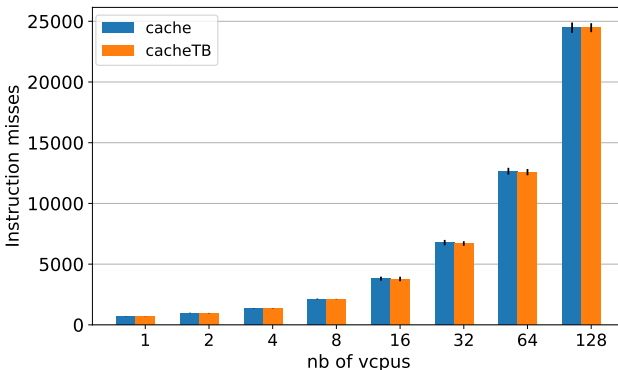
Time dependency, repeated occurrences of timer interrupts (2nd problem)  
⇒ QEMU not run with Linux

## What about the 1st problem?

Stopped TB execution  
⇒ Error negligible on instructions (less than 0,001%)

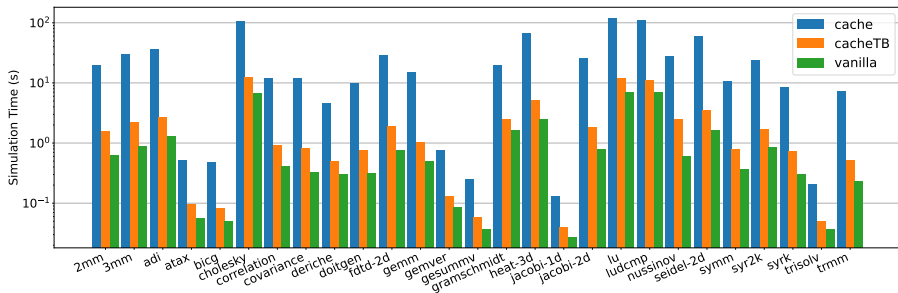


# L1i Statistics validation: comparison with existing cache plugin



Total of **instruction misses** for lu\_cb (log scale on x-axis)

# L1i Simulation time: Mono-core



Simulation time of the PolyBench/C programs (log scale on y-axis)

## L1i Simulation time comparison

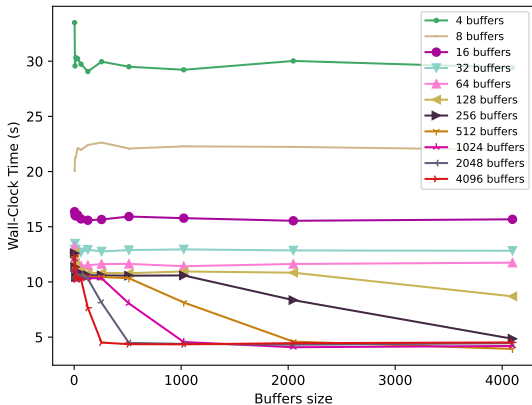
	PolyBench/C	PARSEC
Speedup of <b>cacheTB</b> over <b>cache</b>	10.87	7.18
Slowdown of <b>cache</b> over <b>vanilla</b>	23.67	59.85
Slowdown of <b>cacheTB</b> over <b>vanilla</b>	2.07	10.16

Mean simulation time ratios.

### Conclusion:

- Our L1i is 7 to 10 times faster than the existing cache plugin

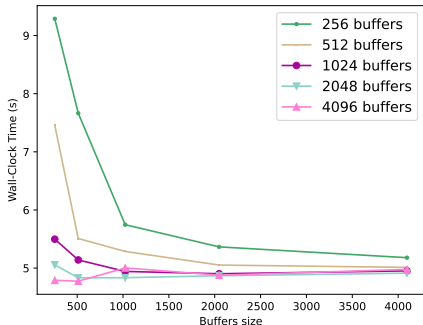
# L1d Optimal buffer size and buffer count: Mono-core



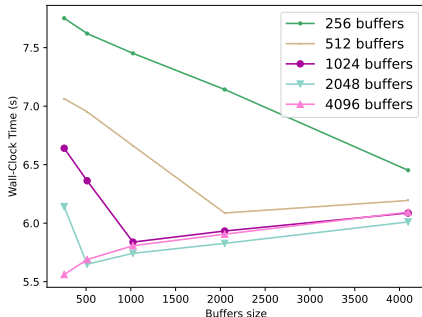
Simulation time of  $\text{trmm}$  with variations of number of buffers and buffer size



# L1d Optimal buffer size and buffer count: Multi-core water\_nsquared



Simulation time of the PARSEC  
water\_nsquared with **4** vCPUs



Simulation time of the PARSEC  
water\_nsquared with **64** vCPUs

# L1d Simulation time comparison

## PolyBench/C

Config number x size: 1024x1024

**Results:** no improvements, same execution time as the existing plugin

## PARSEC water\_nsquared

Config number x size: 4096x256

**Results:** improvements with 1,2 and 8 vCPUs only

## Conclusion:

- Finding the optimal combination of values needs to be done by investigating each benchmark

## L2 Simulation time comparison

### L2 implementation

- Our L1i model
- Naive L1d simulation at each memory access execution

	PolyBench/C	PARSEC
Speedup <b>cache</b> to <b>cacheTB</b>	2.19	2.43
Slowdown <b>cache</b> to <b>vanilla</b>	38.20	99.79
Slowdown <b>cacheTB</b> to <b>vanilla</b>	17.03	43.14

Mean simulation time ratios.

### Conclusion:

- Our memory hierarchy with L2 is 2 times faster than the existing cache plugin





# Summary

*Dynamic Binary Translation **speed** and **accuracy** trade-offs: investigating parallel scalability and cache simulation*

## DBT Simulation Speed

- QEMU parallel implementation scales well on a multi-core host
- Bypassing host Linux scheduler with pinning does not have any effect

## DBT Simulation Accuracy

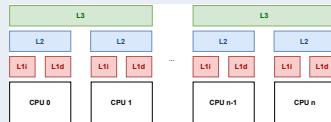
- Significant results with per TB execution of DBT mechanism for instruction cache model
- Limited results with separated threaded data cache simulation



## Future works

### Cache coherency

- Copies of a data among the cache levels
- Must ensure correct state of all the caches
- Scalability?



### Dependency on QEMU runtime

- Full-system mode reflects more closely the reality
- Deeper understanding of QEMU time handling mechanisms

### Cache simulation and security

- Attacks related to caches
- Pseudo timed cache simulation



Marie Badaroux and Frédéric Pétrot. "Arbitrary and Variable Precision Floating-Point Arithmetic Support in Dynamic Binary Translation," 26th Asia and South Pacific Design Automation Conference (ASP-DAC), Tokyo, Japan, pp. 325-330. <https://doi.org/10.1145/3394885.3431416>

Marie Badaroux, Saverio Miroddi and Frédéric Pétrot. "To Pin or Not to Pin: Asserting the Scalability of QEMU Parallel Implementation", 24th Euromicro Symposium on Digital Systems Design (DSD), pp. 238-245. <https://doi.org/10.1109/DSD53832.2021.00045>

Marie Badaroux, Julie Dumas, and Frédéric Pétrot. 2023. Fast Instruction Cache Simulation is Trickier than You Think. In Proceedings of the DroneSE and RAPIDO: System Engineering for constrained embedded systems (RAPIDO '23). Association for Computing Machinery, pp. 48–53. <https://doi.org/10.1145/3579170.3579261>

*Thank you!*  
*Q&A*