# A Trace-driven Approach for Fast and Accurate Simulation of Manycore Architectures

**Anastasiia Butko**, Luciano Ost, Abdoulaye Gamatié, Vianney Lapôtre, Rafael Garibotti and Gilles Sassatelli
LIRMM (CNRS and University of Montpellier), Montpellier, France

Chris Adeniyi-Jones, ARM Ltd.,
Cambridge, United Kingdom





# Introduction


Design a new type of computer architecture capable of setting future global High Performance Computing (HPC) standards that will deliver Exascale performance while using 15 to 30 times less energy.

An efficient manycore system exploration framework is required:

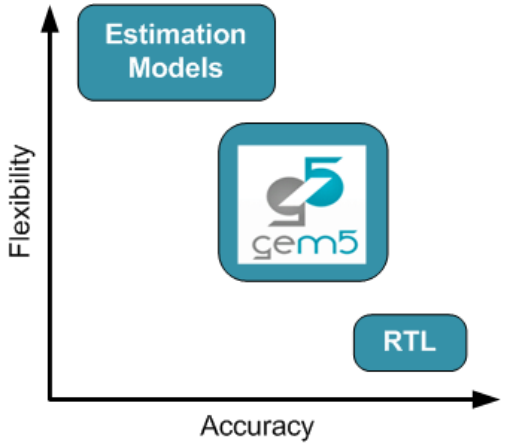
- Architectural parameters exploration
- Architectural scaling (increasing core count)

# Simulation Frameworks

| Simulator | Accuracy | Supported processor architectures | License | Development/ Support activity |
|---|---|---|---|---|
| Simics | Functionally-accurate | Alpha, ARM, MIPS, PowerPC, SPARC and x86 | Private | Yes |
| PTLsim | Cycle-accurate | x86 | Open | Yes |
| SimpleScalar | Cycle-accurate | Alpha, ARM, PowerPC and x86 | Open | No |
| OVPsim | Instruction-accurate | Open Cores Open RISC, ARM, Synopsys ARC, MIPS, PowerPC and others | Open and Private | Yes |
| **Gem5** | **Quasi-cycle-accurate** | **Alpha, ARM, x86, SPARC, PowerPC and MIPS** | **Open** | **Yes** |

# Gem5 Simulation Framework

- **Flexibility**
  - Multiple ISAs and CPU models
  - Memory systems
  - Debugging capabilities




- **Accuracy**
  - Reported between 2% and 18% *

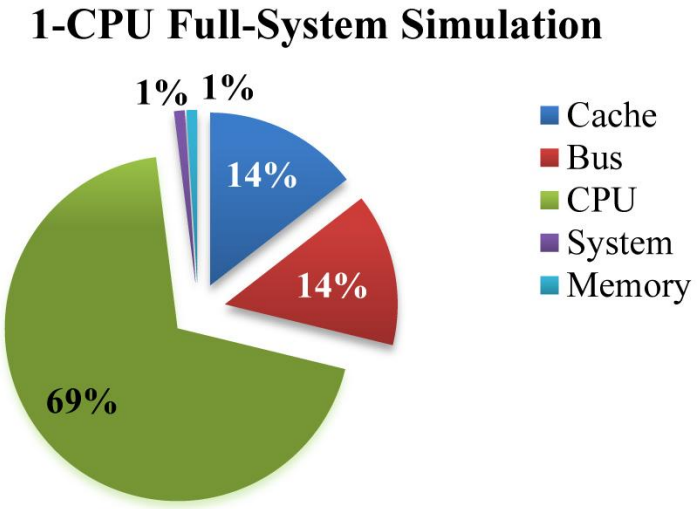
- **Speed**
  - **Between 1 KIPS and 1MIPS**

* Butko A. et al., *Accuracy evaluation of Gem5 simulator system*

# Efficient system simulation techniques

**Motivation**:

- For regular applications:
  - CPU Computation phase is deterministic
  - Memory Communication phase is traffic dependent

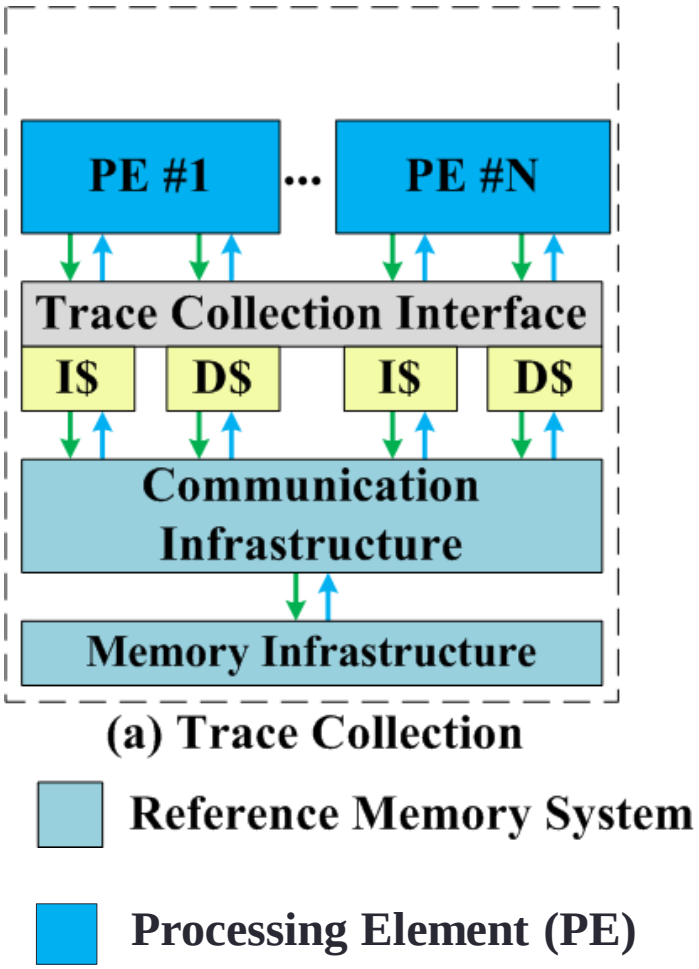



- **Trace-Driven simulation principle:**
  - Collect execution traces in a reference simulation
  - Replay those traces on the trace-driven simulation in which only interconnect and memory system are simulated

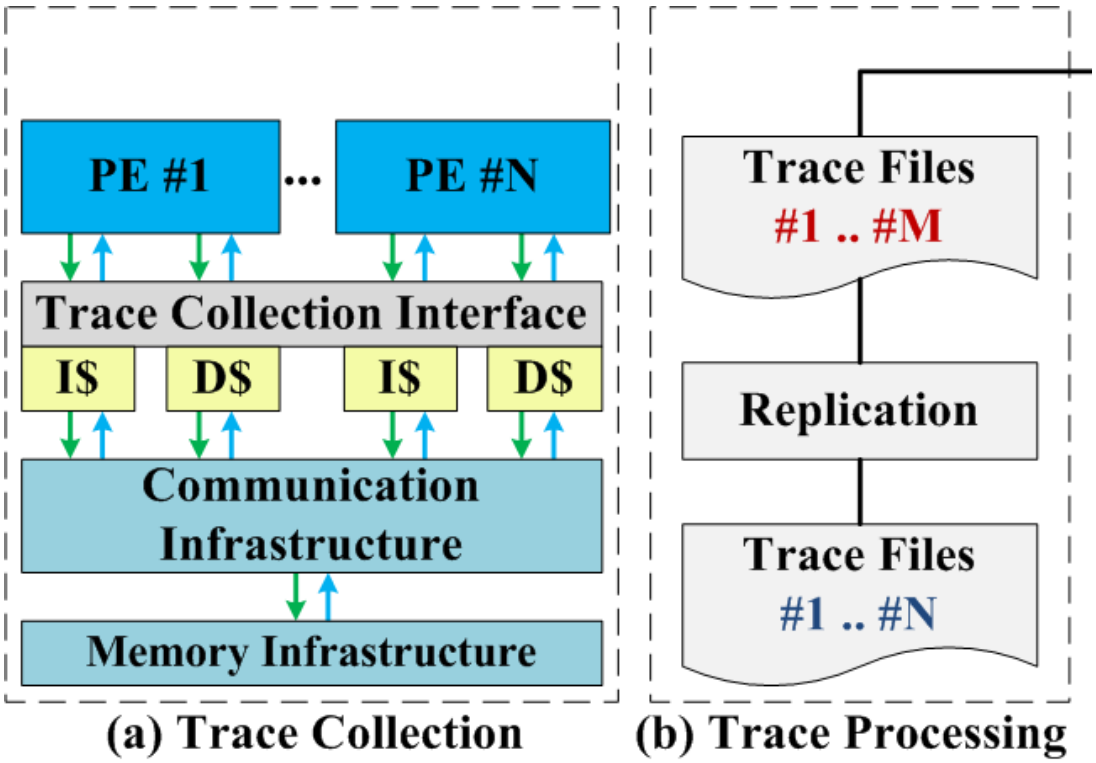
# Outline

- Introduction
- Trace-driven approach
  - General concept
  - Integration in Gem5
- Evaluation
- Conclusion and Future Work

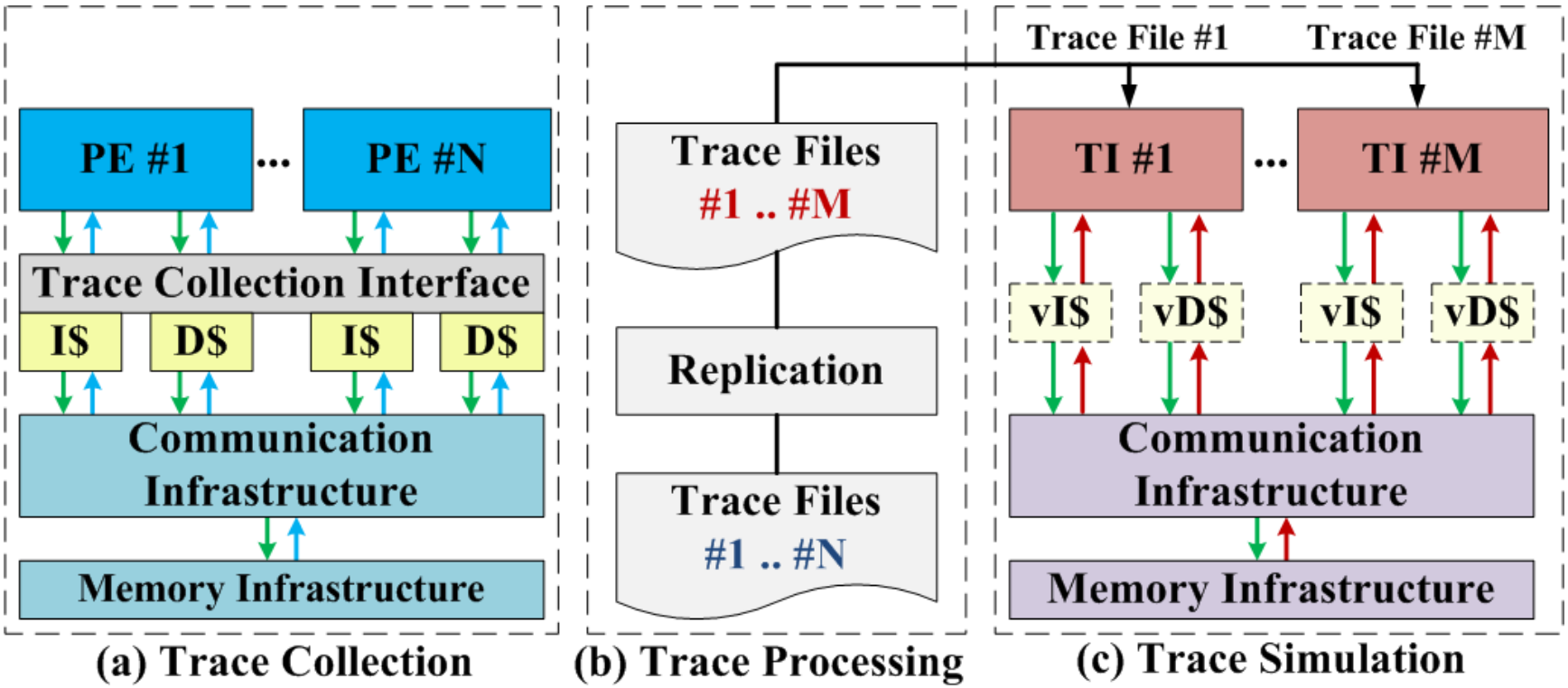
# Trace-Driven Approach: General Concept



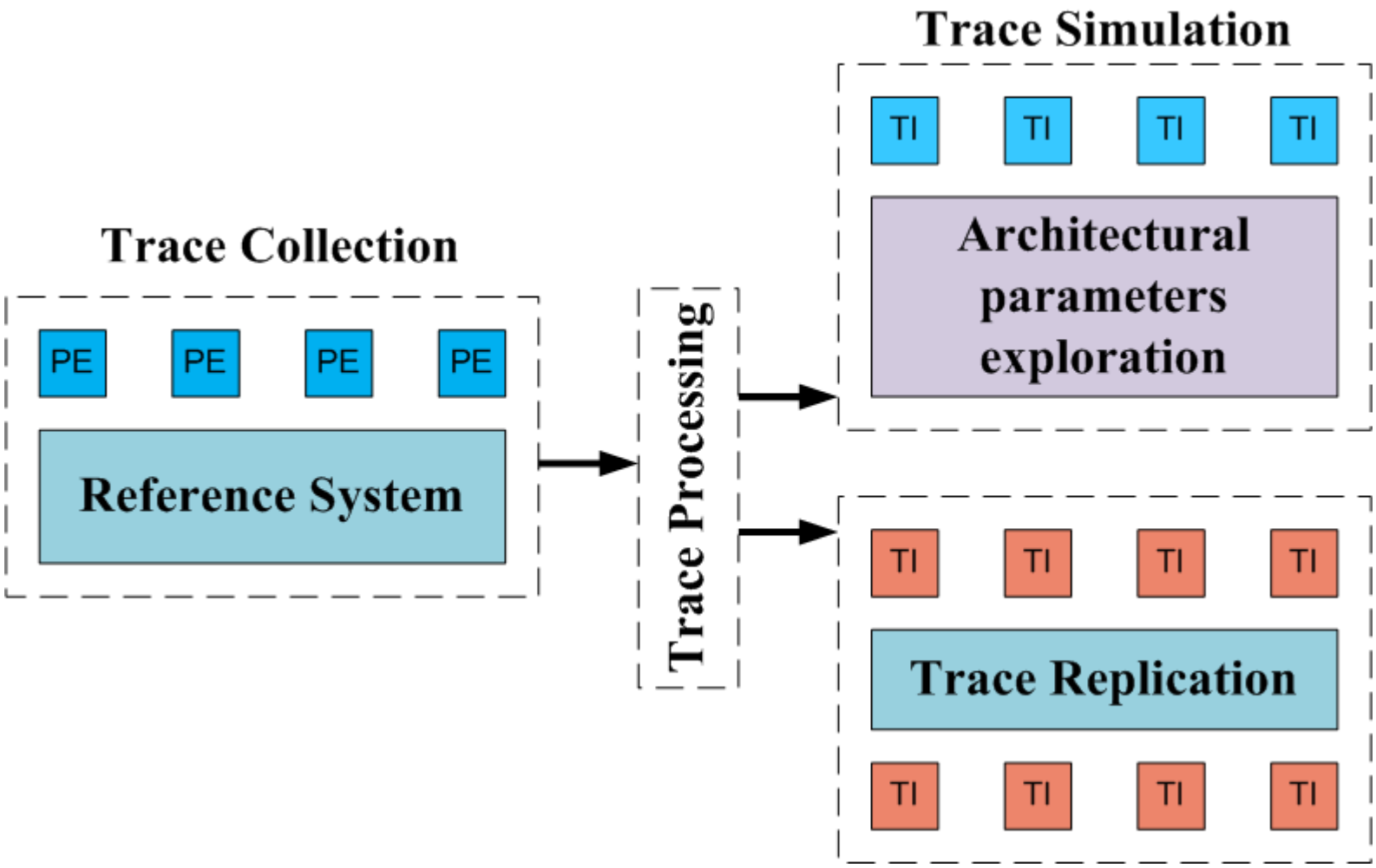
# Trace-Driven Approach: General Concept



# Trace-Driven Approach: General Concept

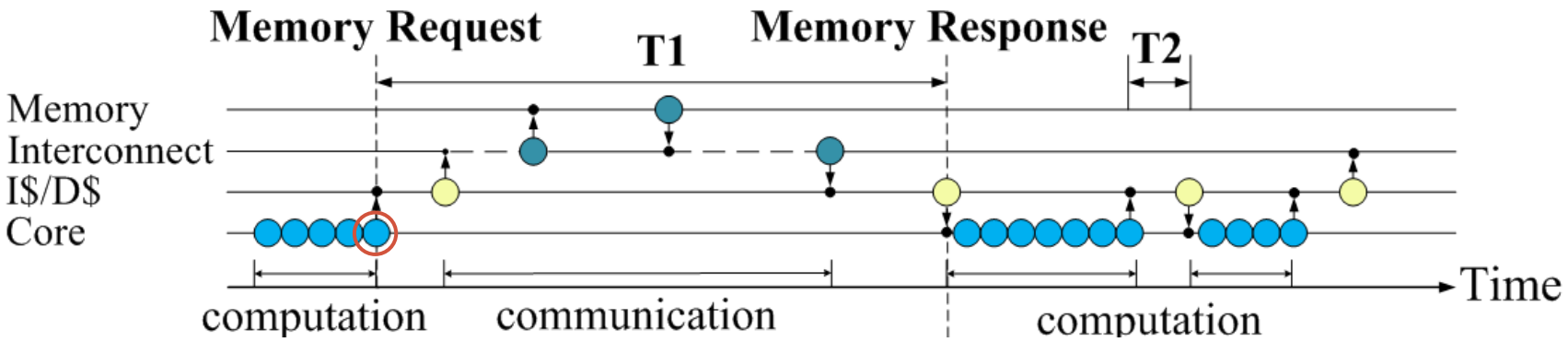


# Case Studies



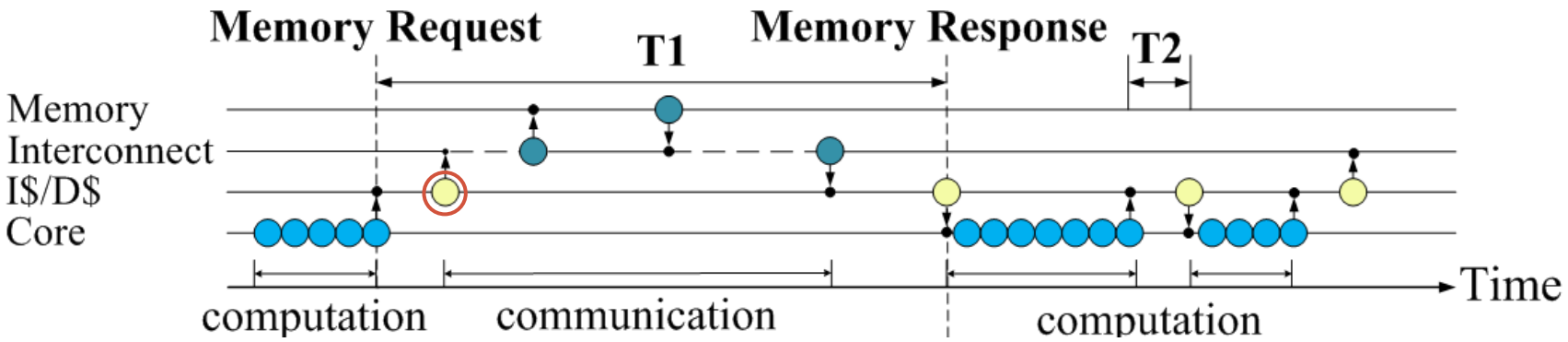
# General Concept: Simulation Gain

Event-Driven



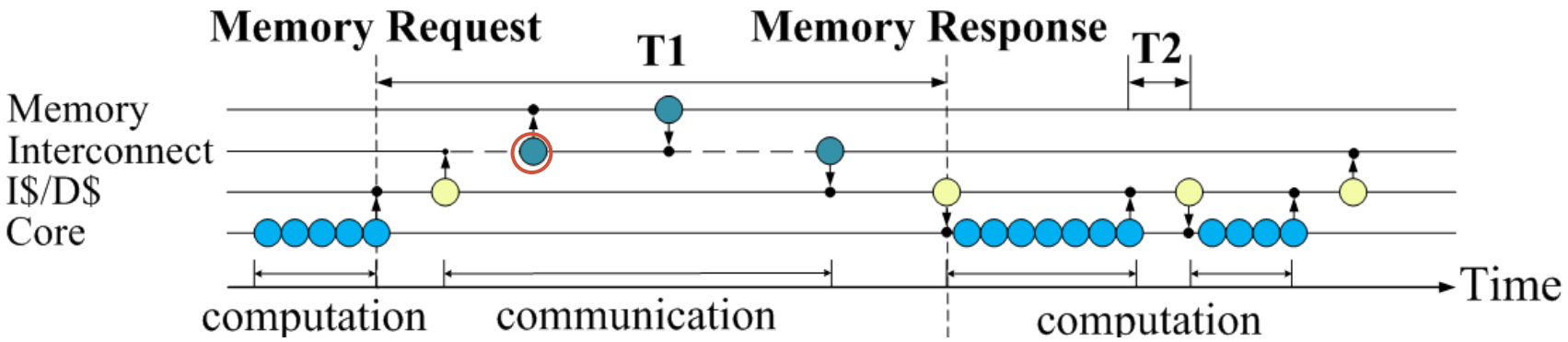
# General Concept

Event-Driven



# General Concept

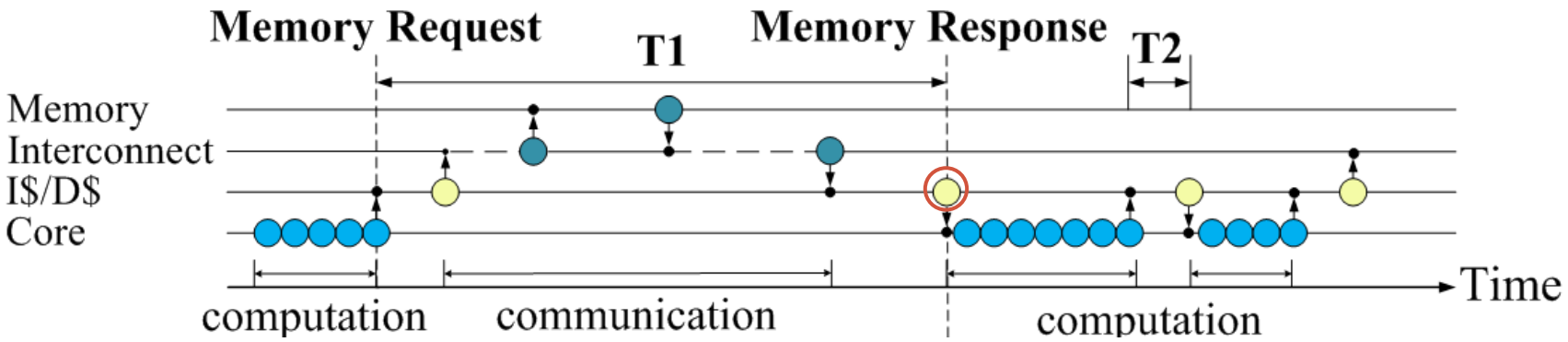
Event-Driven




**Traffic dependent phase**

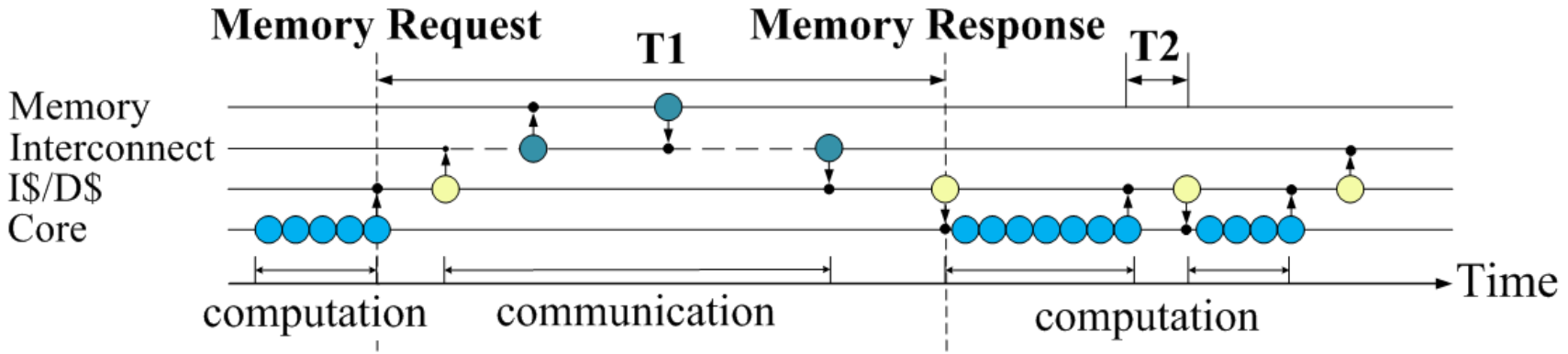
# General Concept

Event-Driven

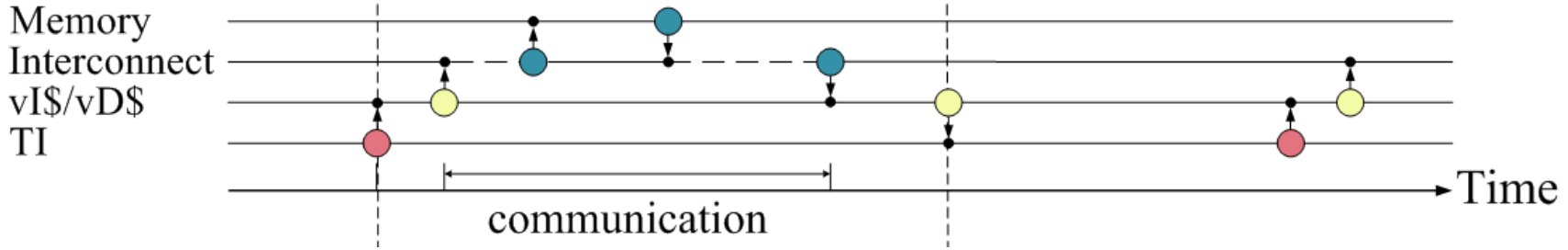


# General Concept

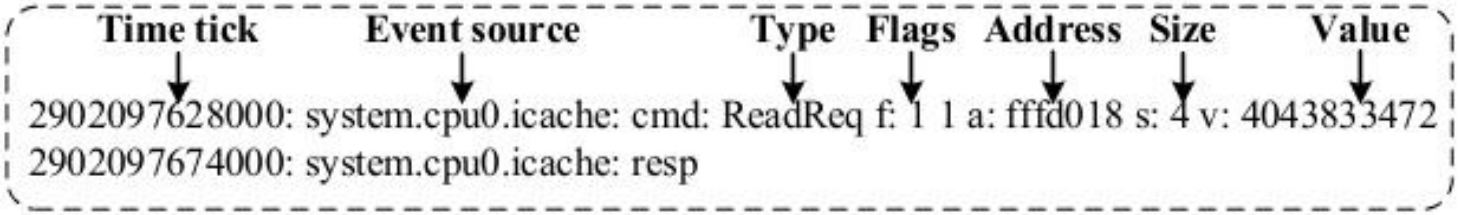
Event-Driven

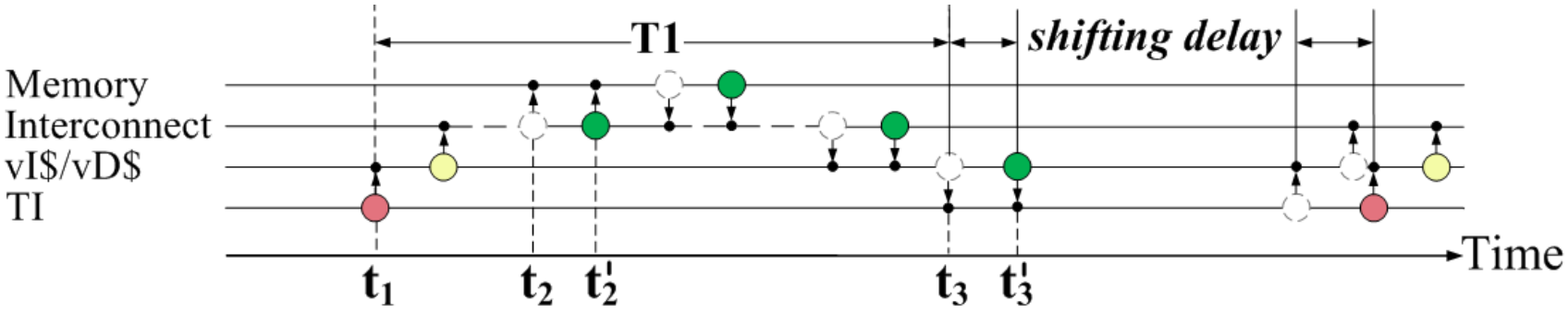



Trace-Driven



# Integration in gem5



# Synchronization

## Trace Collection

Macros are inserted into the application source code

```
#define PTHREAD_CREATE(...)   pthread_create(&threads[i], 0,func,(void *) &params);
                              printf("Thread id %i create tick: %lu\n", ID, rpns());

#define PTHREAD_BARRIER(...)  pthread_barrier_wait(barrier);
                              printf("Thread id %i barrier tick: %lu\n", ID, rpns());

#define PTHREAD_JOIN(...)     pthread_join(threads[i], 0);
                              printf("Thread id %i join tick: %lu\n", ID, rpns());
```

## Trace file

```
Thread id 0 create tick: 1627449500
Thread id 1 create tick: 1627690378

Thread id 1 barrier tick: 1628594414
Thread id 0 barrier tick: 1628632384

Thread id 0 join tick: 1727194499
Thread id 1 join tick: 1727322559
```

# Synchronization

## Trace File 0

**Thread id 0 create tick: 2859888754000**

2859888764000: cpu0.dcache: WriteReq f: 0 0 a: f89e240 s: 4 v:0

2859888795000: cpu0.dcache: resp

2859888831000: cpu0.dcache: WriteReq f: 0 0 a: f89e280 s: 4 v:0

2859888862000: cpu0.dcache: resp

**Thread id 0 barrier tick: 2859888890000**

2859888924000: cpu0.dcache: WriteReq f: 0 0 a: f89e2c0 s: 4 v:0

2859888955000: cpu0.dcache: resp

2859889004000: cpu0.dcache: ReadReq f: 0 0 a: f89e240 s: 4 v:0

2859889035000: cpu0.dcache: resp

**Thread id 0 join tick: 2859889120000**

## Trace File 1

**Thread id 1 create tick: 2859888754000**

2859888774000: cpu1.dcache: WriteReq f: 0 0 a: f89e240 s: 4 v:0

2859888805000: cpu1.dcache: resp

2859888841000: cpu1.dcache: WriteReq f: 0 0 a: f89e280 s: 4 v:0

2859888872000: cpu1.dcache: resp

**Thread id 1 barrier tick: 2859888900000**

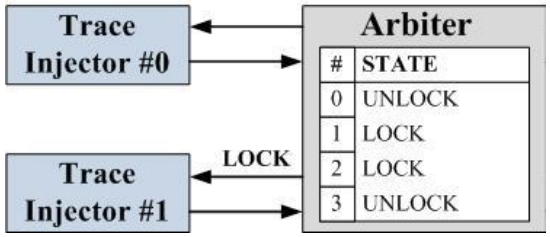
2859888934000: cpu1.dcache: WriteReq f: 0 0 a: f89e2c0 s: 4 v:0

2859888965000: cpu1.dcache: resp

2859889014000: cpu1.dcache: ReadReq f: 0 0 a: f89e240 s: 4 v:0

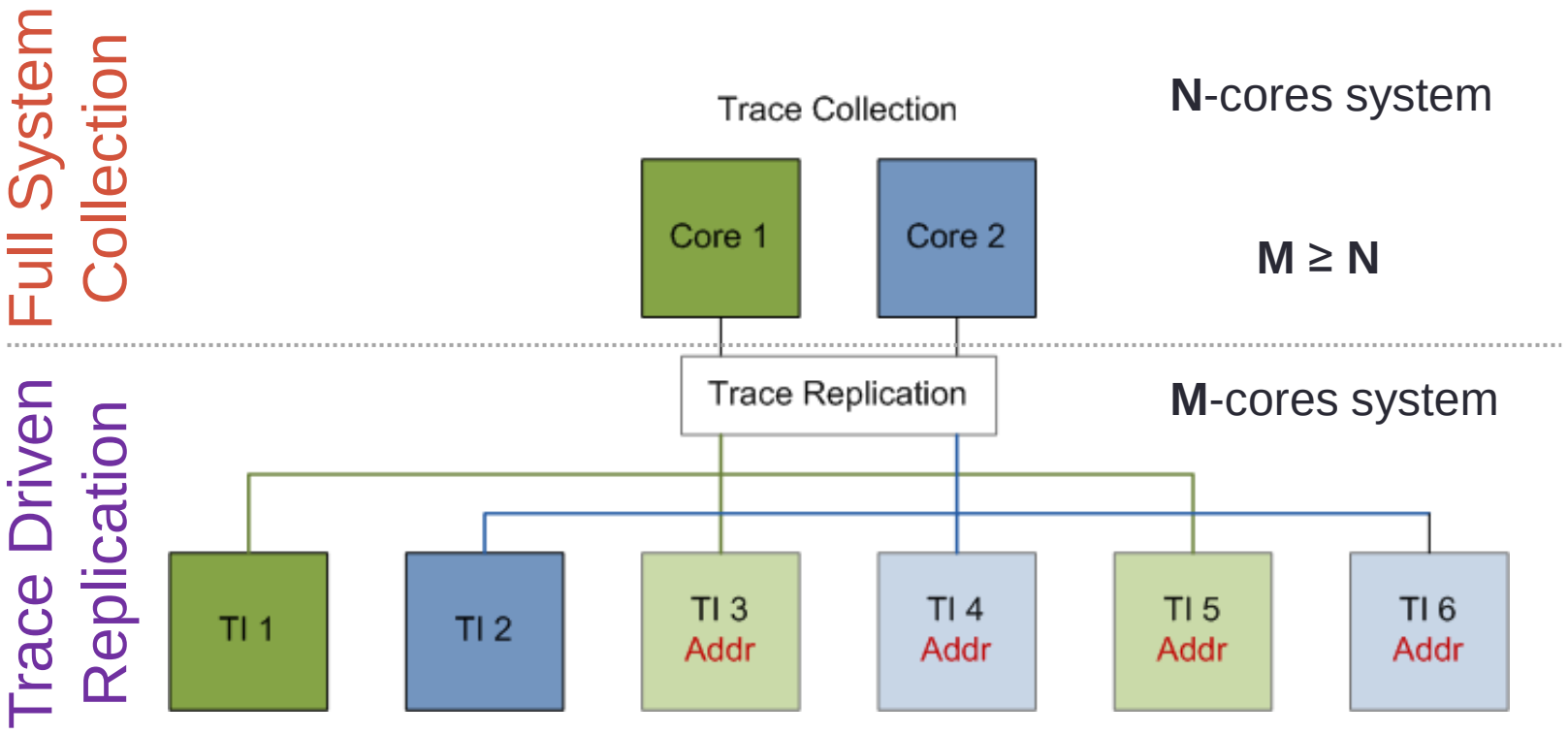
2859889045000: cpu1.dcache: resp

**Thread id 1 join tick: 2859889100000**



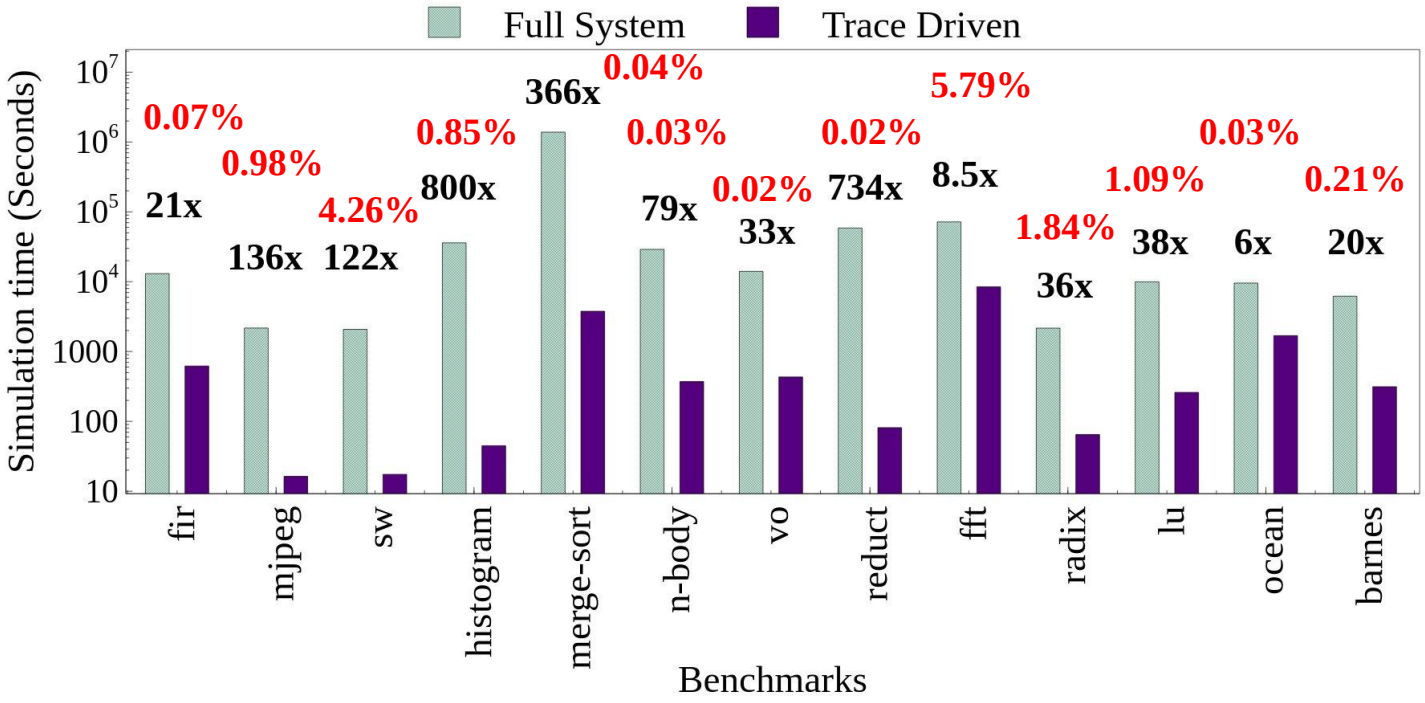
# Integration in gem5

- Trace Replication
  - timing behavior, synchronization, address distribution



# Speedup and Accuracy

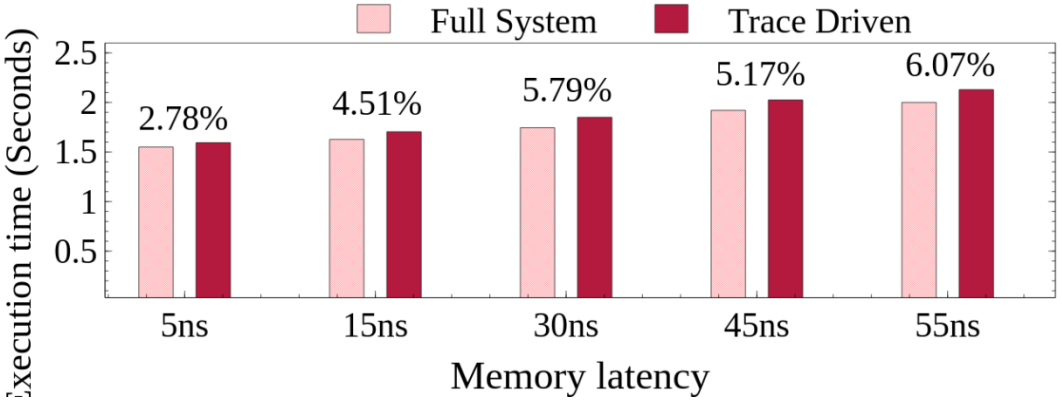
- Scientific application benchmark suite



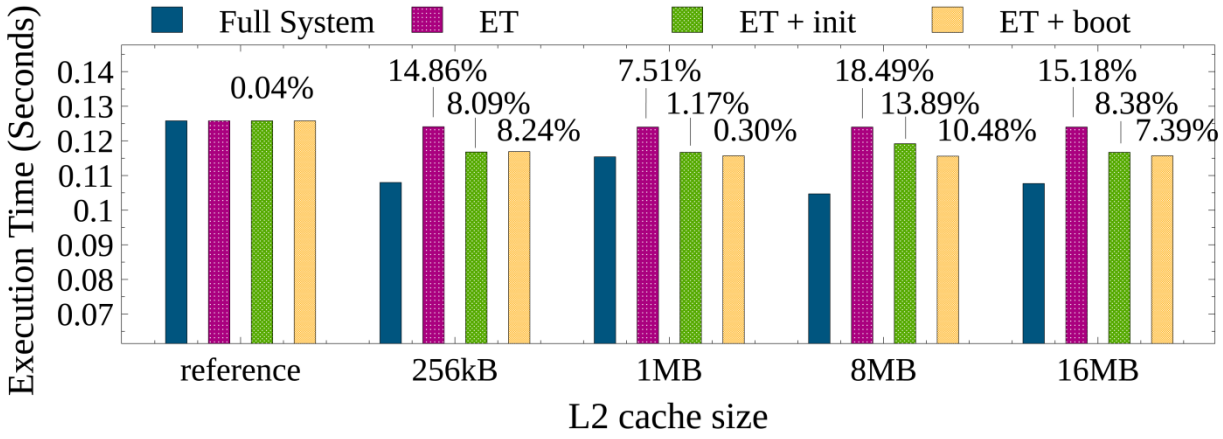

- Error percentage on execution time varies from 0.02% to 6%
- Speedup ranges from 6x to 800x

# Exploration of Architectural Parameters

- Memory latency variation

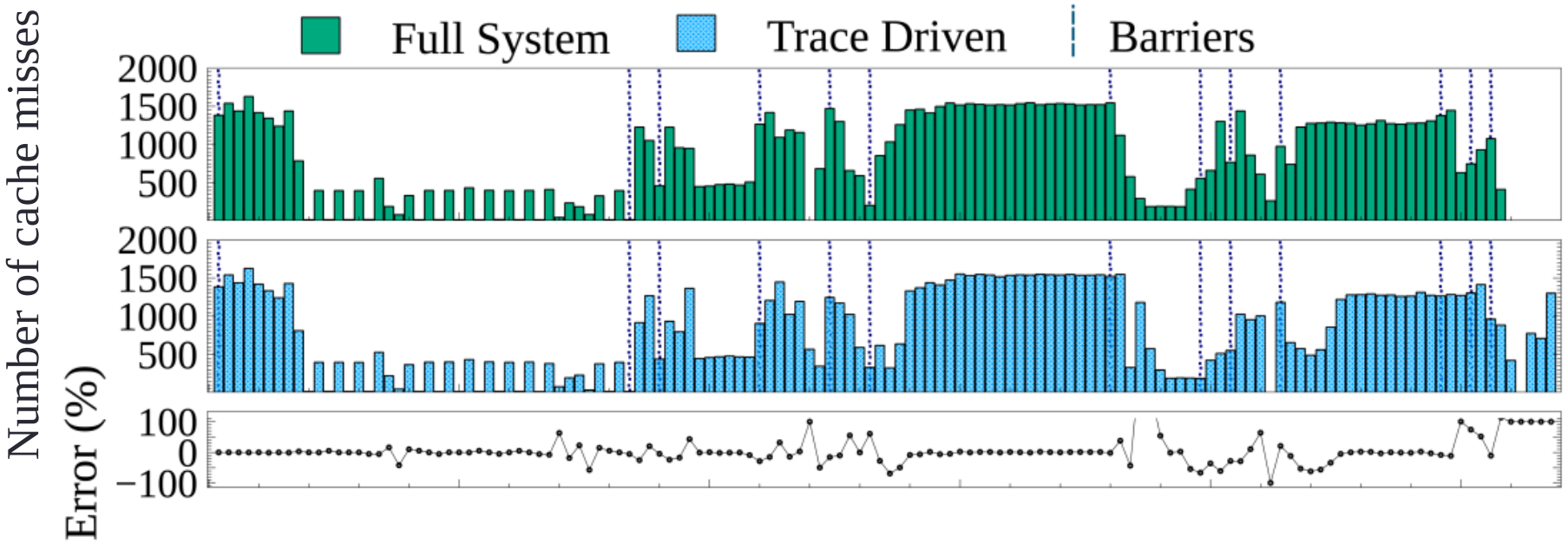



- L2 cache



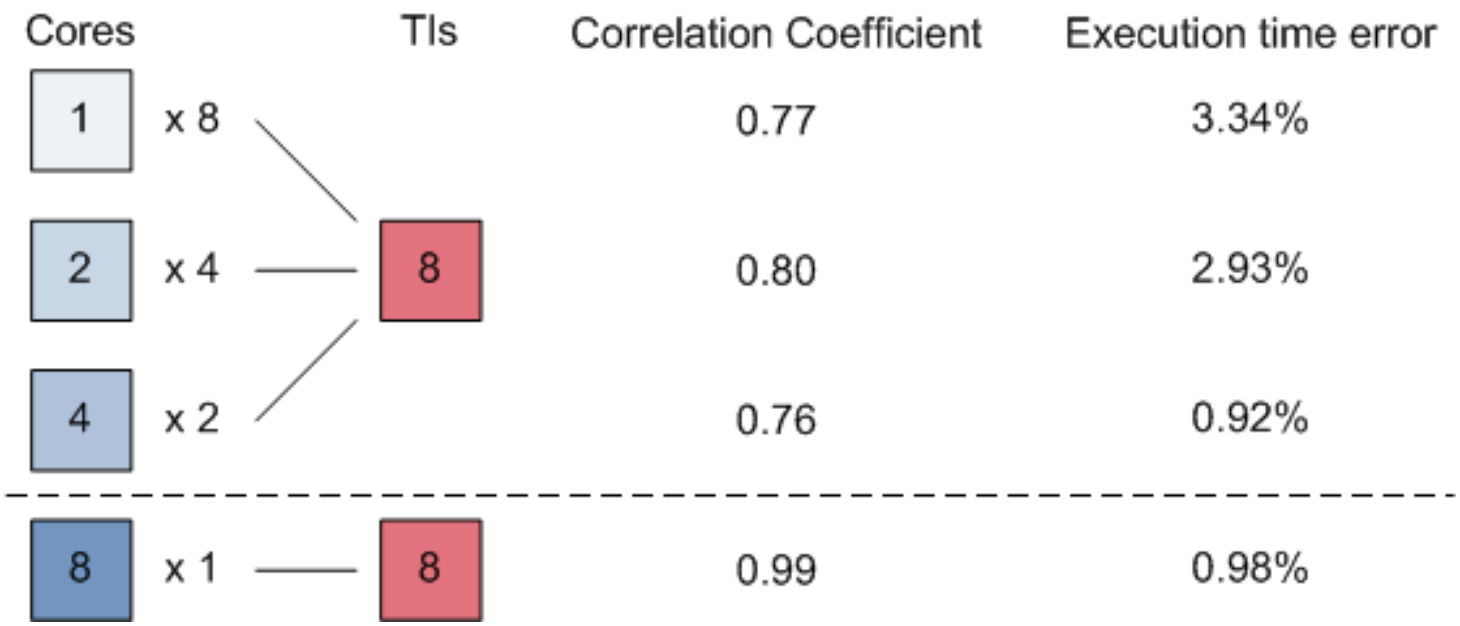
# Consistency of Simulations in Presence of Dependencies

- Similarity between cache miss behaviors
- Negligible fluctuations are compensated



# Trace Replication

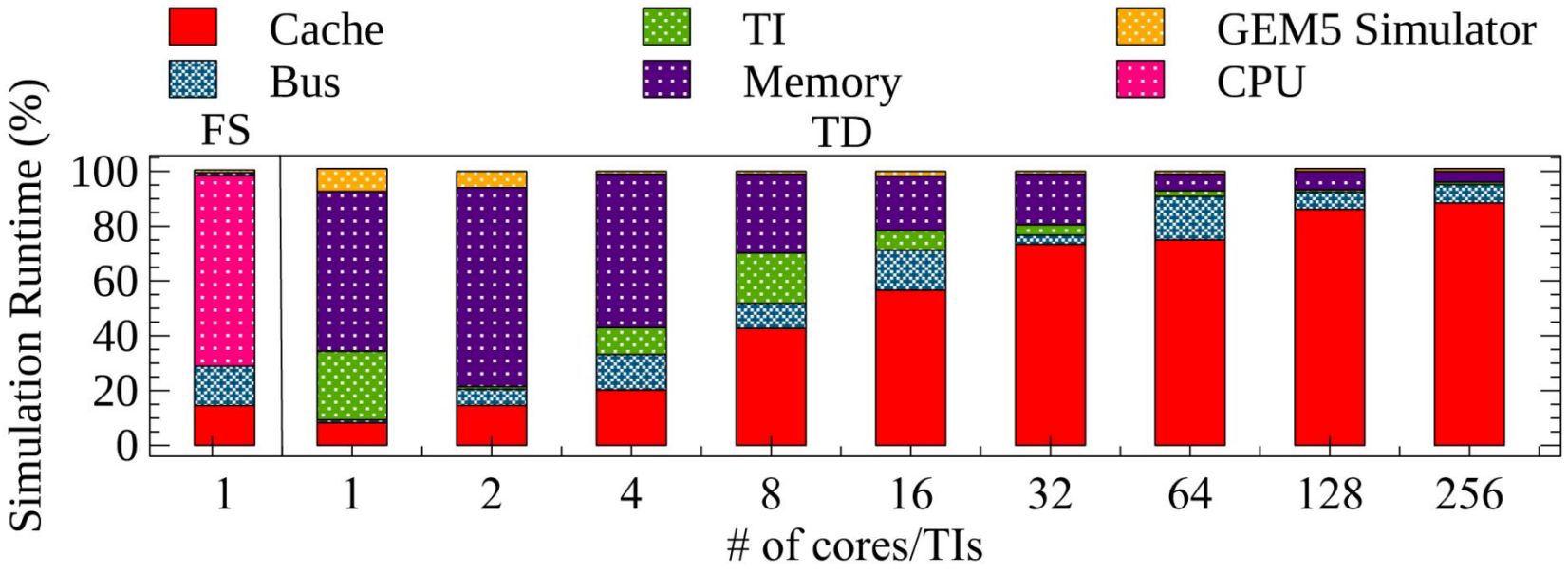
- Correlation of traces from 1-,2-,4-,8-cores RP collection



# Simulation Cost

- Simulation profiling on the work station

  Intel Core i5 CPU @ 2.60 GHz x 4



# Conclusion and Future Work

- **Results (for regular not data-dependent workloads)**
  - Simulation speedup of up to 800x
  - Error percentage varies from 0.02% to 6%
  - Implementation is freely available online

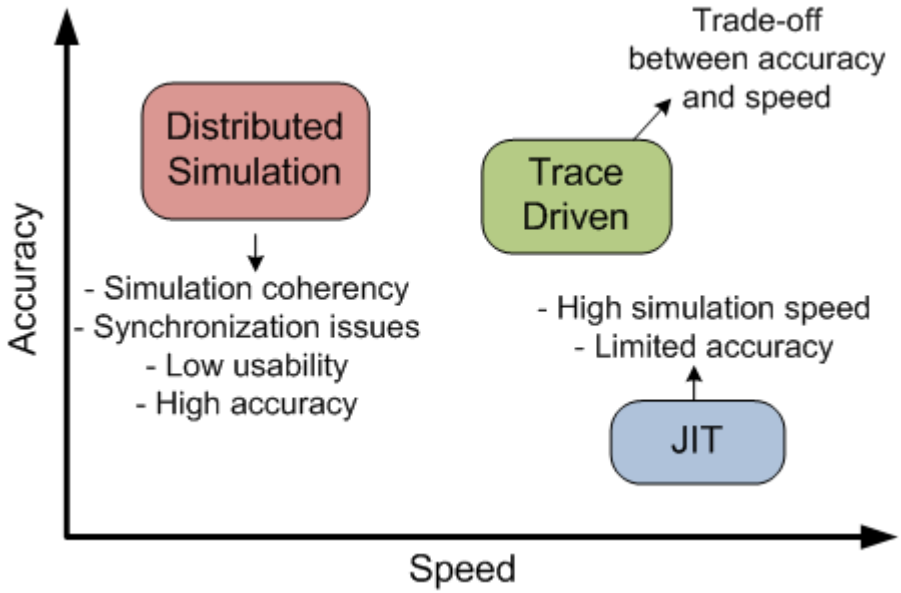
    http://www.lirmm.fr/ADAC

- **Future Work**
  - optimization of trace file
  - out-of-order processor support
  - different memory mapping algorithms

# Thank you for your attention!

# Questions?

# Efficient system simulation techniques




- Trace-driven simulations:
  - TaskSim : exploring multithreaded application behaviors *(BSC, Spain*)
  - ManySim: memory models evaluation for 32-cores architecture *(Systems Technology Lab, Intel)*
  - Netrace: network exploration only *(The University of Texas, NVIDIA)*

# Integration in gem5

1. Trace Collection/Processing Phases
   - Memory access traces

Collection

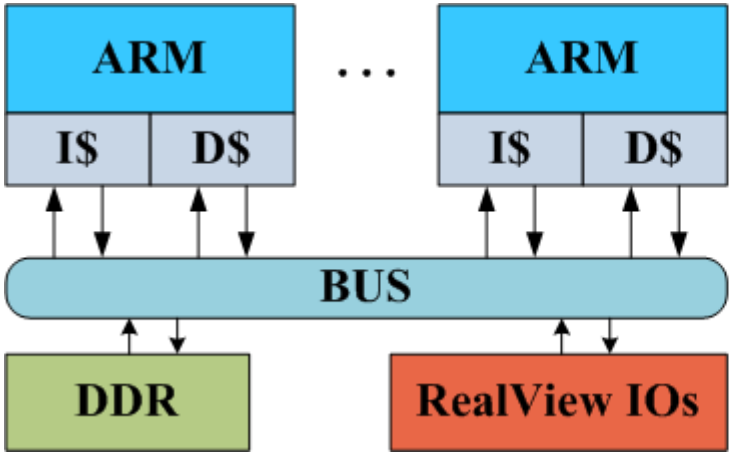
```
DPRINTF(CacheTraceMiss, "cmd: %s f: %d %d a: %x s: %d v: %d \n" ,
                        pkt->cmdString(),          // request type
                        pkt->req->isInstFetch(),   // is instruction fetch
                        pkt->isRead(),             // is read
                        pkt->getAddr(),            // destination address
                        pkt->getSize(),            // packet size
                        data);                     // packet value
DPRINTF(CacheTraceMiss, "resp\n");
```

Traces

```
Time tick   Event source                                  Type  Flags  Address  Size  Value
2902097628000: system.cpu0.icache: cmd: ReadReq f: 1 1 a: fffd018 s: 4 v: 4043833472
2902097674000: system.cpu0.icache: resp
```

# Evaluation

- Reference platform (RP)
  - Gem5 Full System experimental setup




| Parameter | Value |
|---|---|
| # of cores | from 1 to 8 |
| CPU frequency | 500 MHz |
| L1 cache size | 4 kB |
| Channel width | 64 bits |
| DDR latency | 30 ns |
| OS | Linux Kernel 2.6.38 |
| Programming | POSIX Threads |