



Horizon 2020
European Union funding
for Research & Innovation

MONT-BLANC

# Trace-driven simulation of multithreaded applications in gem5

Gilles SASSATELLI
sassatelli@lirmm.fr

Alejandro NOCUA, Florent BRUGUIER, Anastasiia BUTKO

Cambridge, UK – September 11, 2017

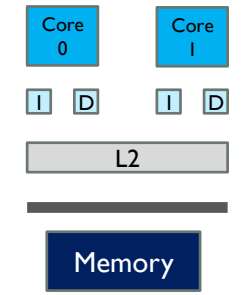
ARM®

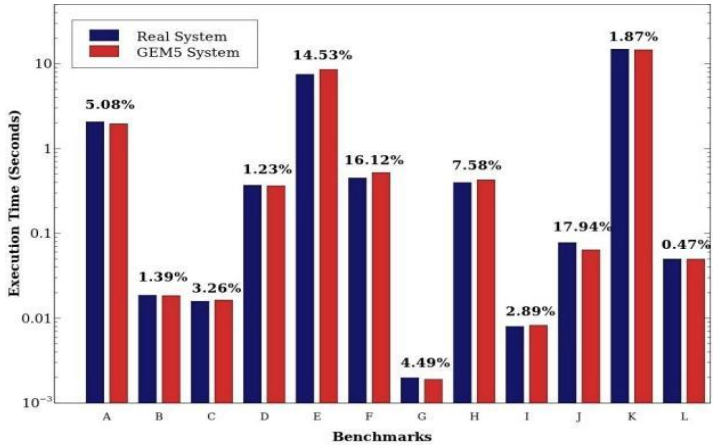
**ARM gem5 Research Summit 2017 Workshop**

# Background motivations

- **Mont-Blanc 1, 2 & 3 projects (FP7, H2020)**
  - Getting ARM technology ready for HPC: HW, SW & Apps
  - Advances in energy efficiency towards Exascale

- **Initial effort: using gem5 for performance prediction (2011)**
  - STE Nova A9500 SoC (dual-core Cortex A9)
  - Fed publicly available parameters into a gem5 FS model
  - 1.5% - 18% error, due to rough DRAM model and interconnect



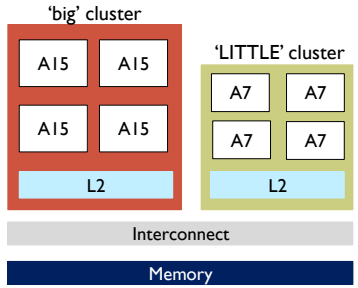




BUTKO A et Al, « Accuracy Evaluation of GEM5 Simulator System », IEEE ReCoSoC 2012: Reconfigurable Communication Centric SoCs, ISBN 978-1-4673-2570-7, 9-11 juillet 2012, York, pp. 1-7.

# Background motivations cont'd

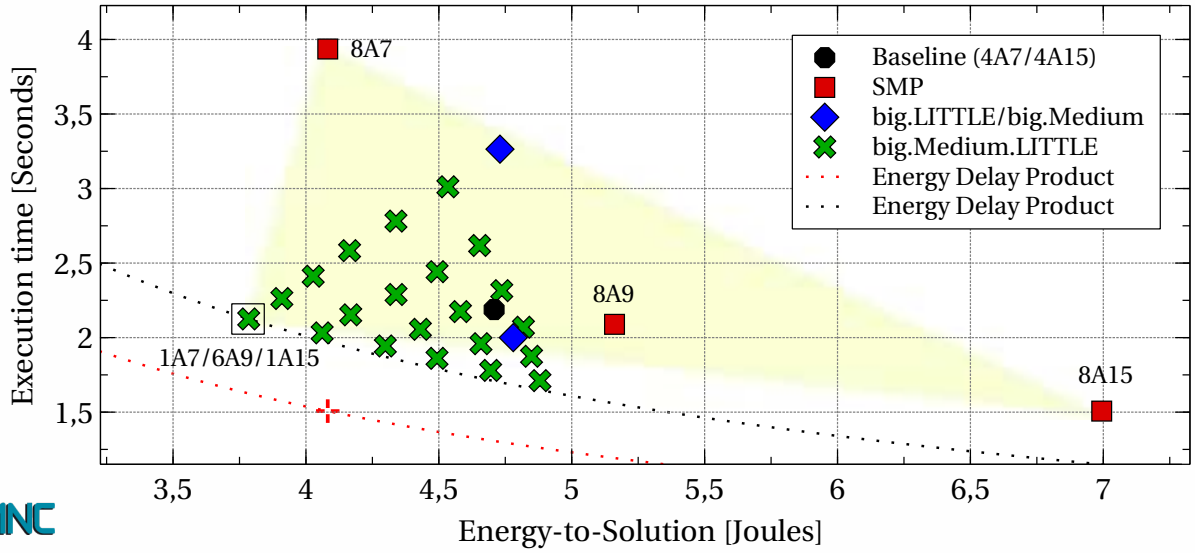
## Calibration against real hardware

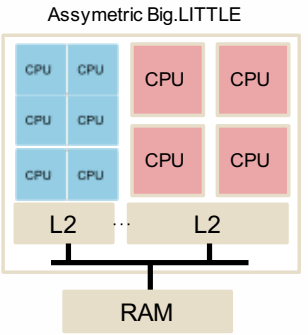
- Using **gem5** for performance prediction
- And **McPAT** for power estimation

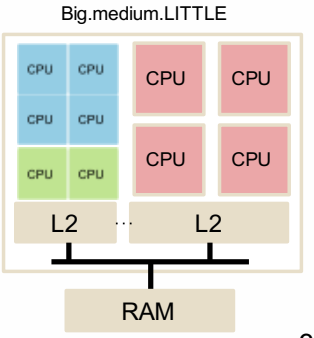



## Then onto exploring fancy architecture configurations

- Heterogeneous single-ISA multicores *à la* big.LITTLE
- Assymetric, 3 levels of heterogeneity etc.

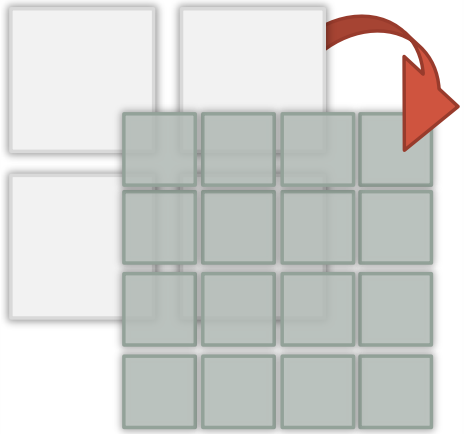


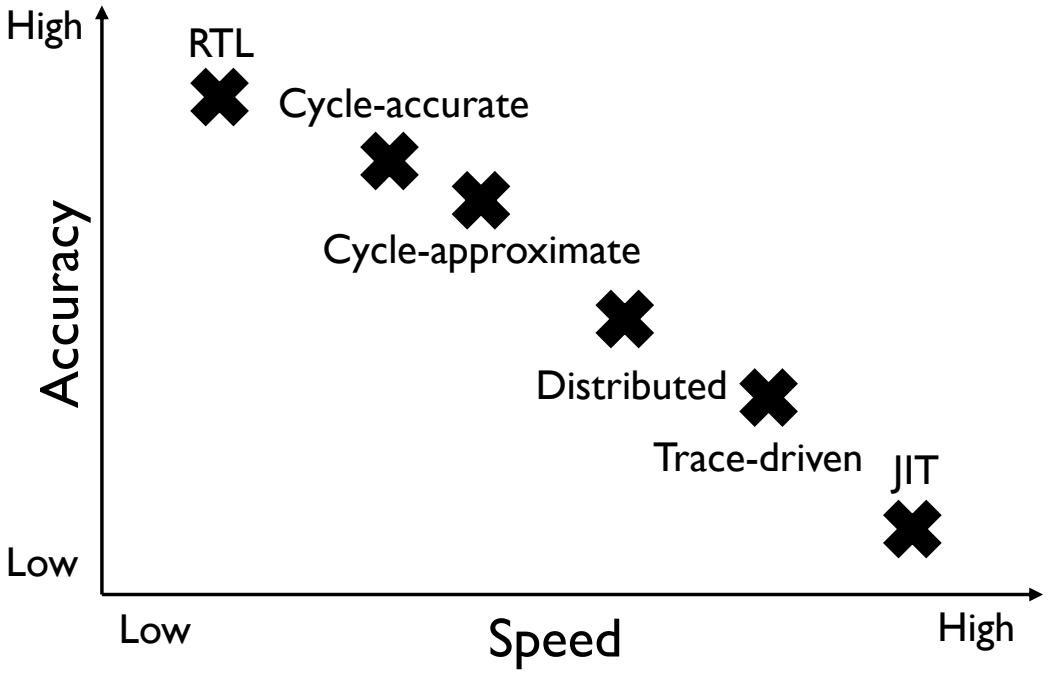



# Background motivations cont'd

- **Not ready for manycores, too slow!**
  - 1K-1M (simulated) IPS
  - Scales bad with system size
  - Already much better than RTL though

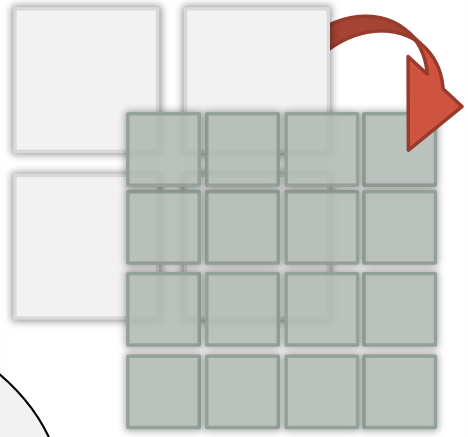


- **Trading speed for accuracy ? Any sweet spot?**

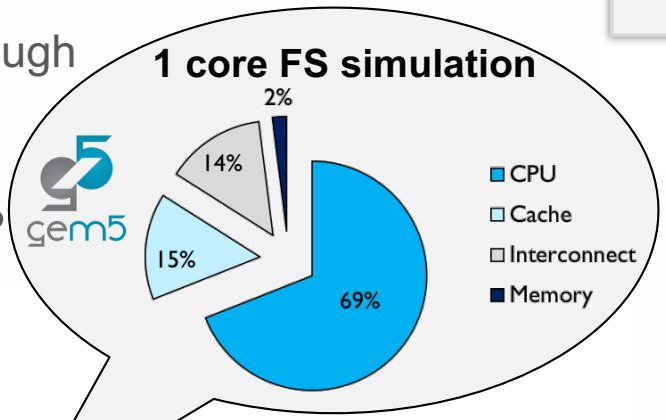


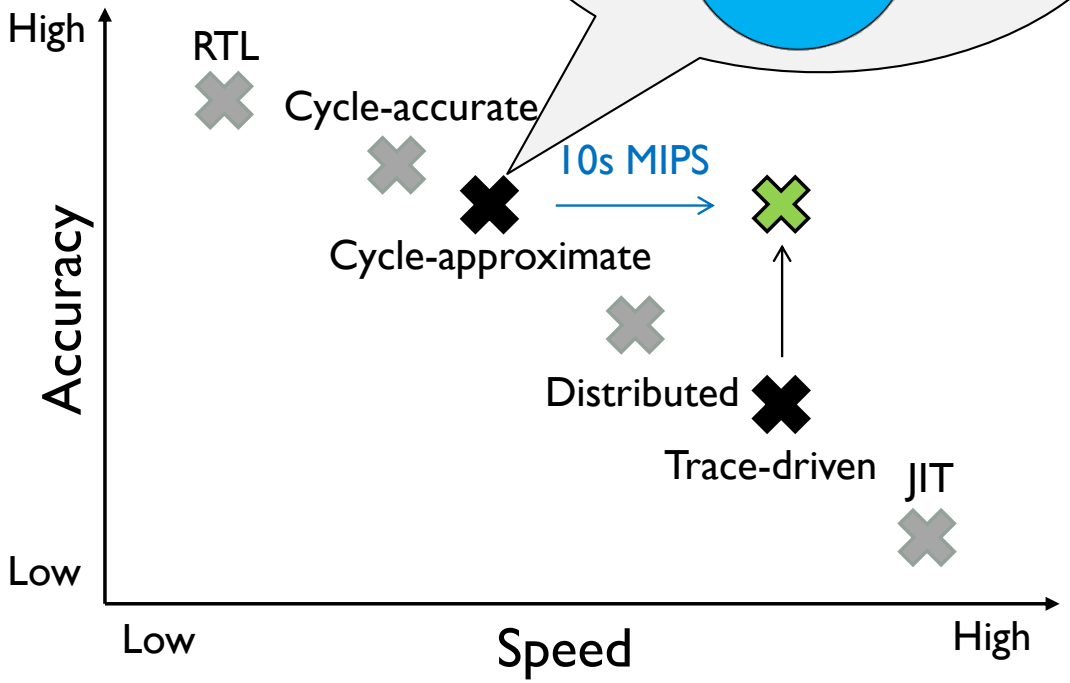
# Background motivations cont'd

- **Not ready for manycores, too slow!**
  - 1K-1M (simulated) IPS
  - Scales bad with system size
  - Already much better than RTL though



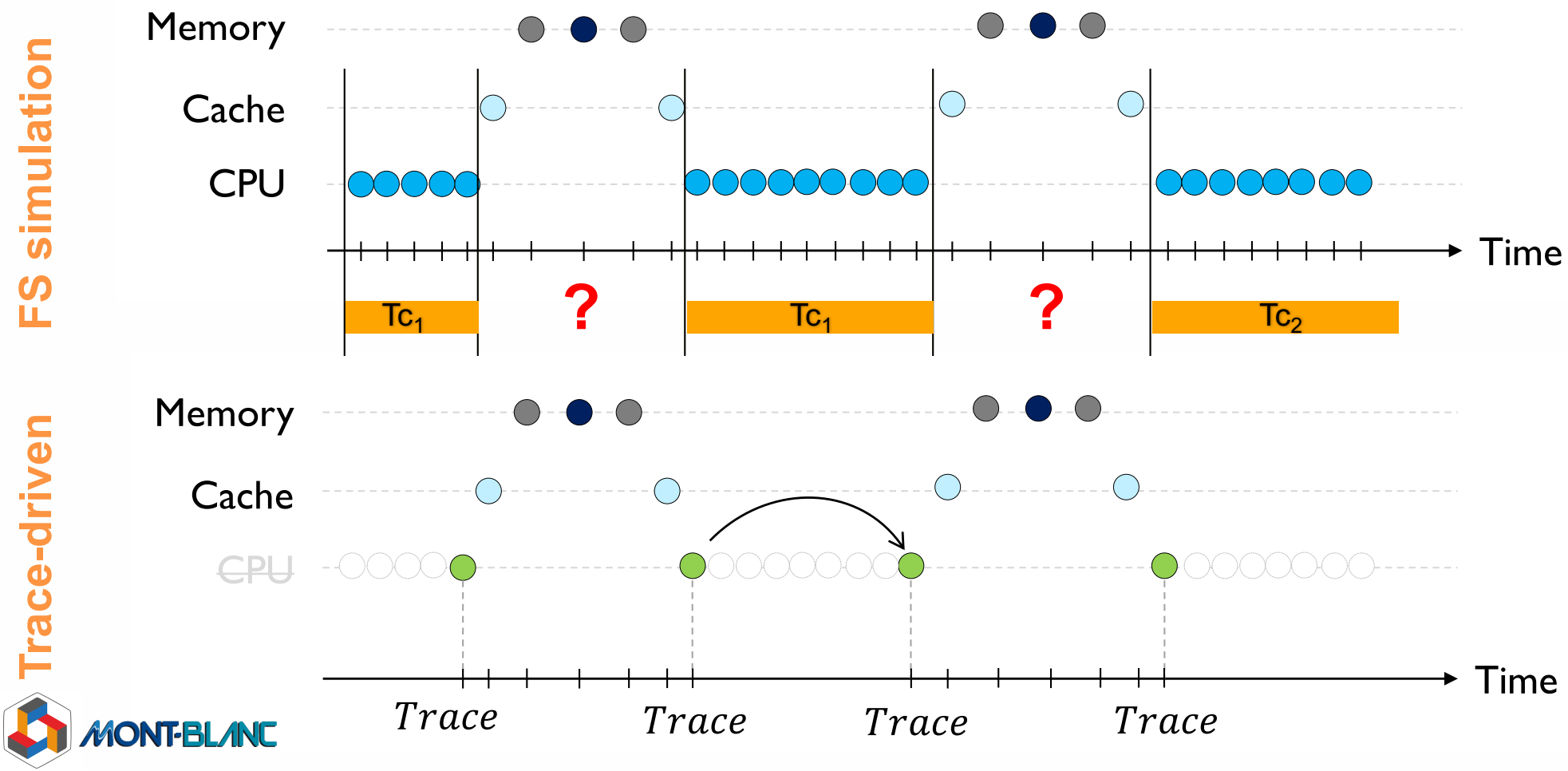
- **Trading speed for accuracy ?**



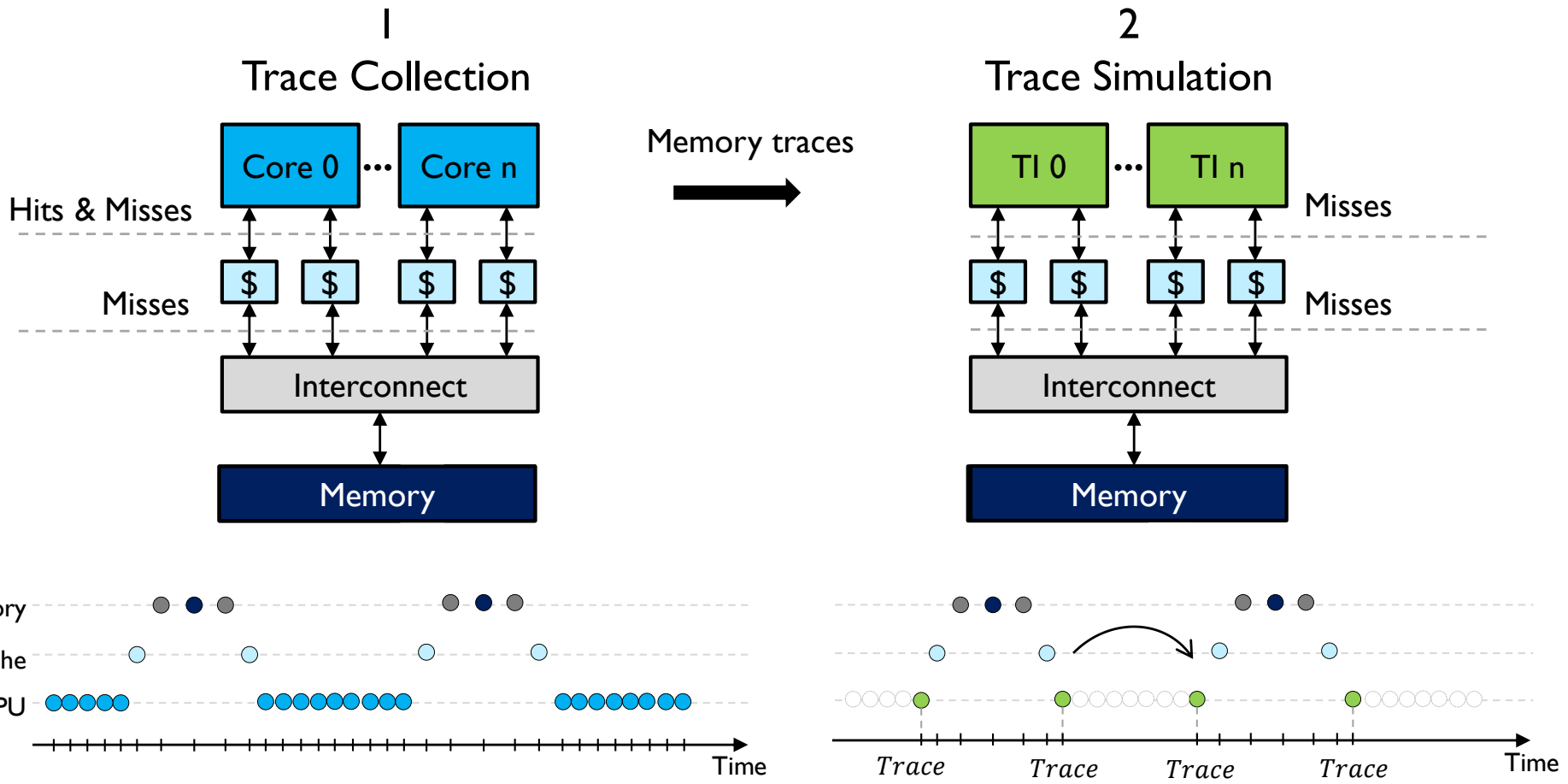
# Trace-driven simulation principle

- **~ 70% simulation effort goes into CPU**
  - Abstracting away CPU cores sounds like a good idea
  - Between 2 consecutive L1 cache misses (in-order) CPU cores perform « consistently »



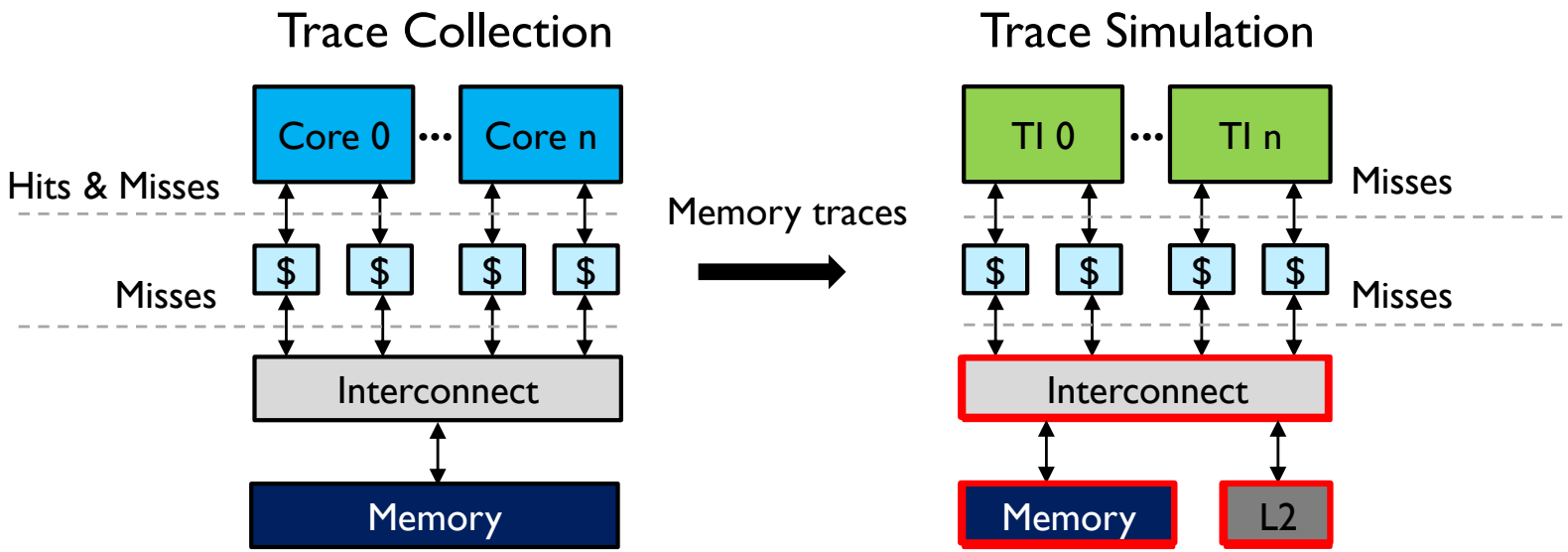
## SimMATE: 2-stage process

- Trace collection: tracing only L1 miss related transactions
- Trace replay: Using trace injectors that initiate transactions as previously recorded



# SimMATE for faster DSE

- **Trace collection = freezing**
  - CPU parameters alongside application SW
  - Private caches sizes, speed etc.
- **Trace replay allows exploring the rest**
  - L2 size, policy etc.
  - Interconnect type & speed
  - Main memory speed

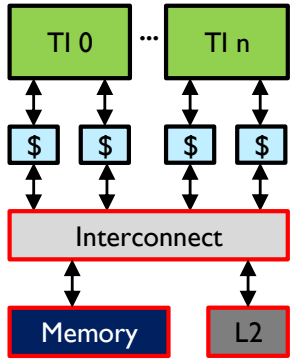


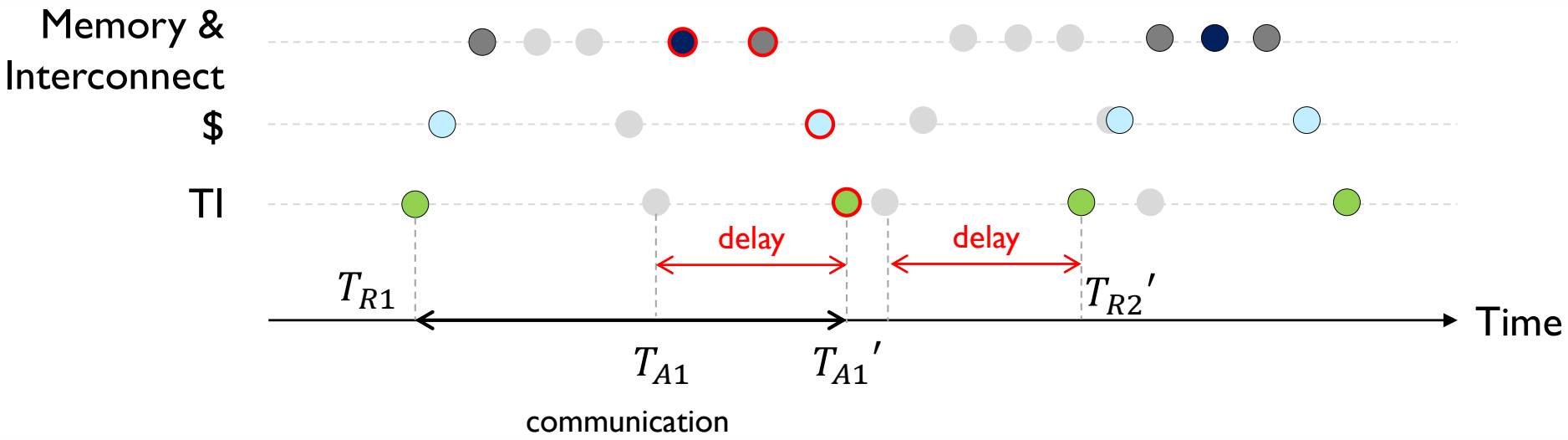

1 time

Multiple times

# SimMATE for faster DSE cont'd

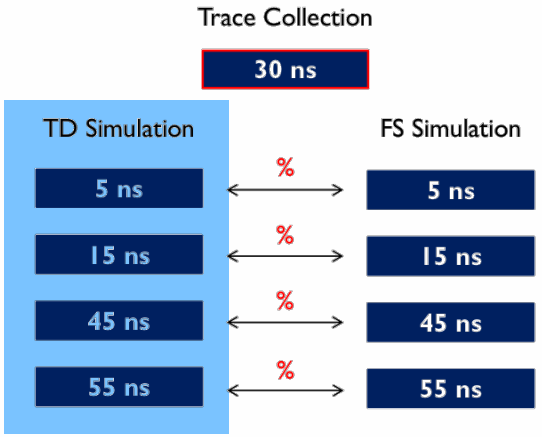
- **Trace replay performs event (re-) scheduling**
  - Simple « time shifting » approach
  - Maintaining constant compute phases

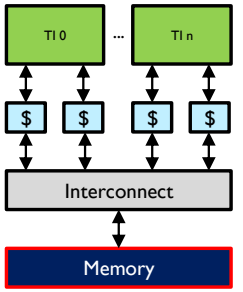


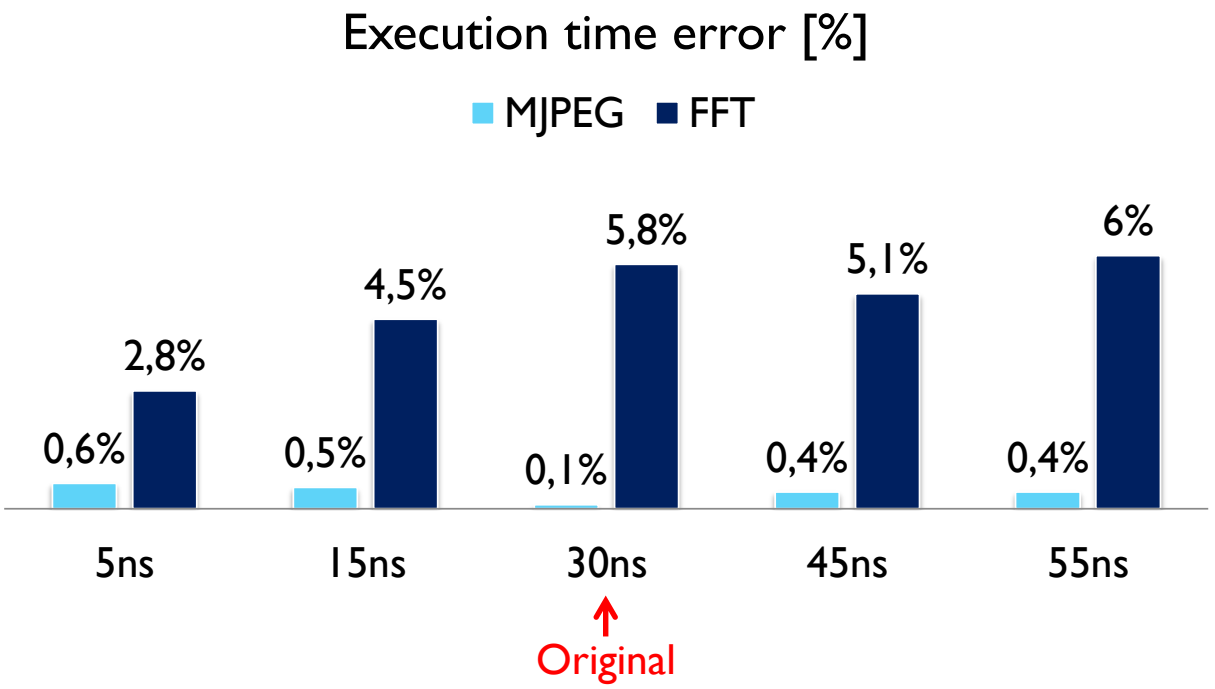



# SimMATE Benchmarking

- **Tuning DRAM latency**
  - Collection performed with 30ns
  - TD simulation from 5ns to 55ns
  - FS used as reference

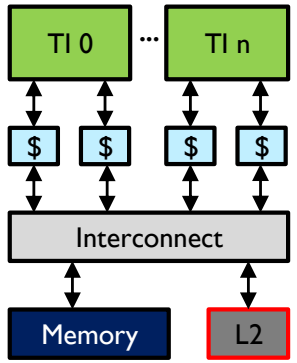


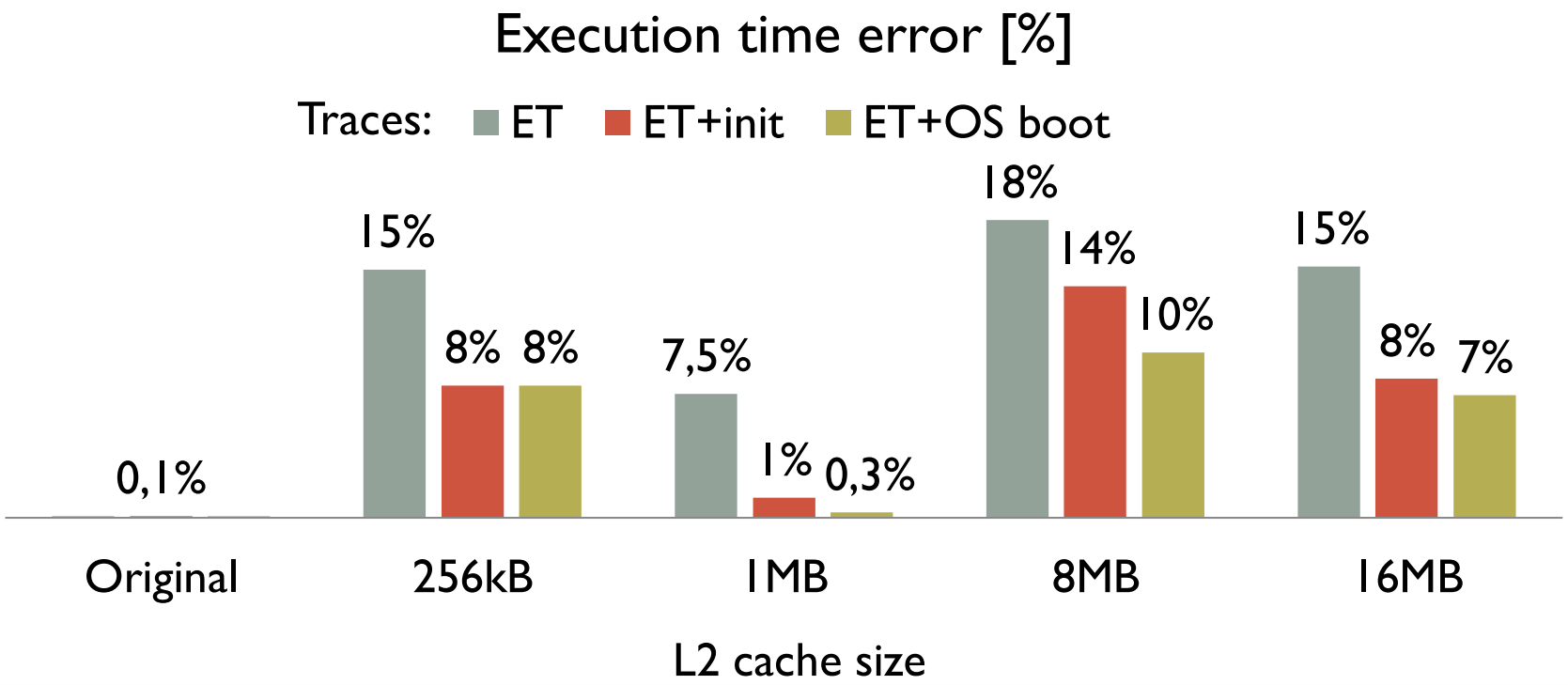



# SimMATE Benchmarking

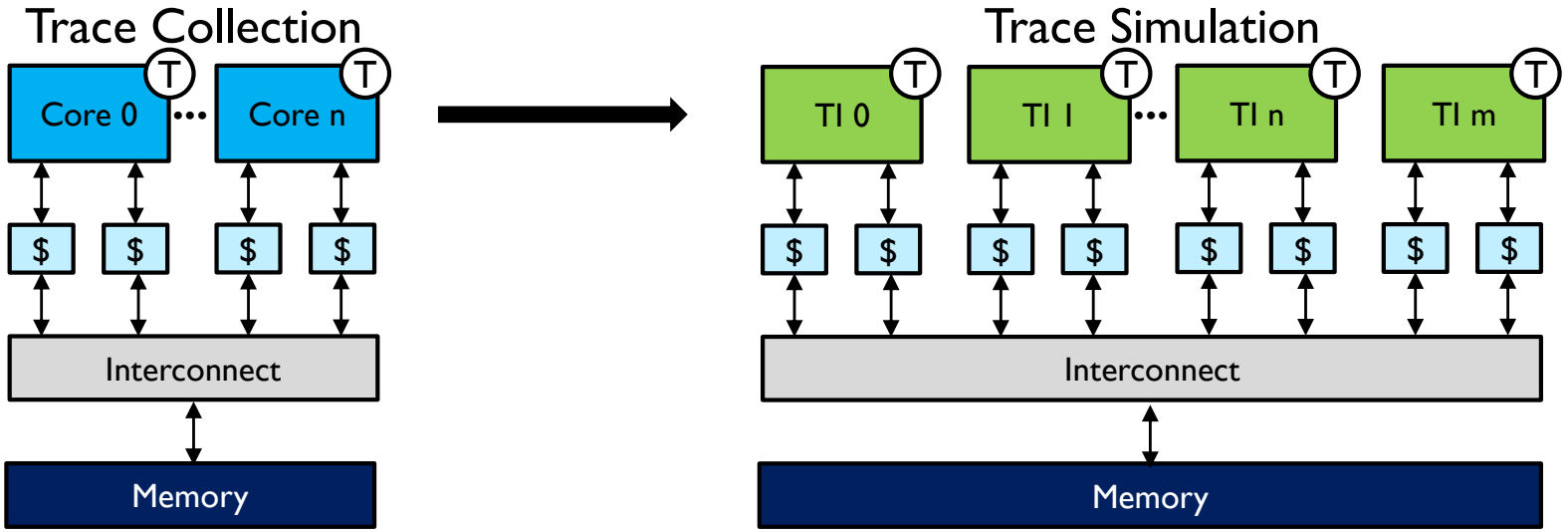
- **Tuning L2 size**
  - Collection performed without L2
  - TD simulation from 0 to 16MB L2
  - Errors originate from Cold-start bias / cache warmup

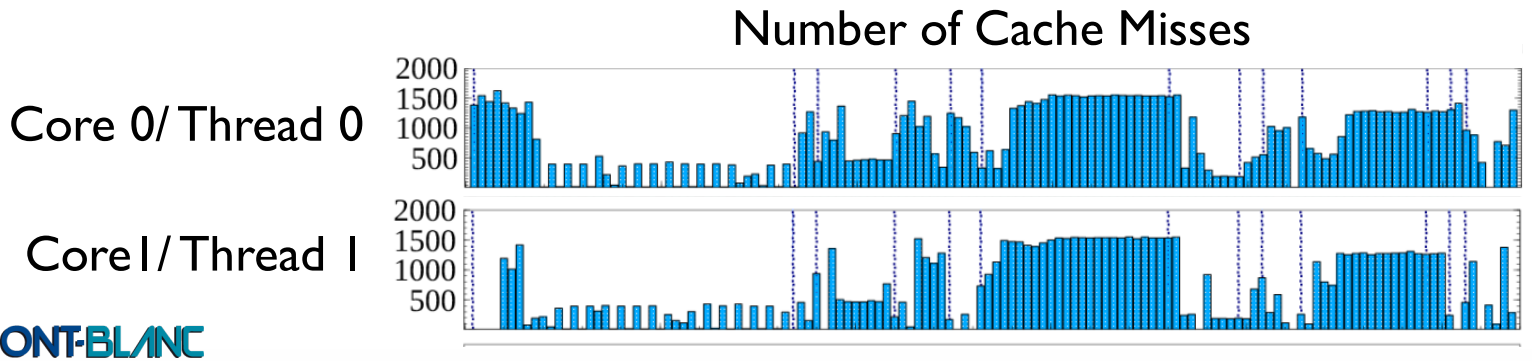


# Multithreaded applications

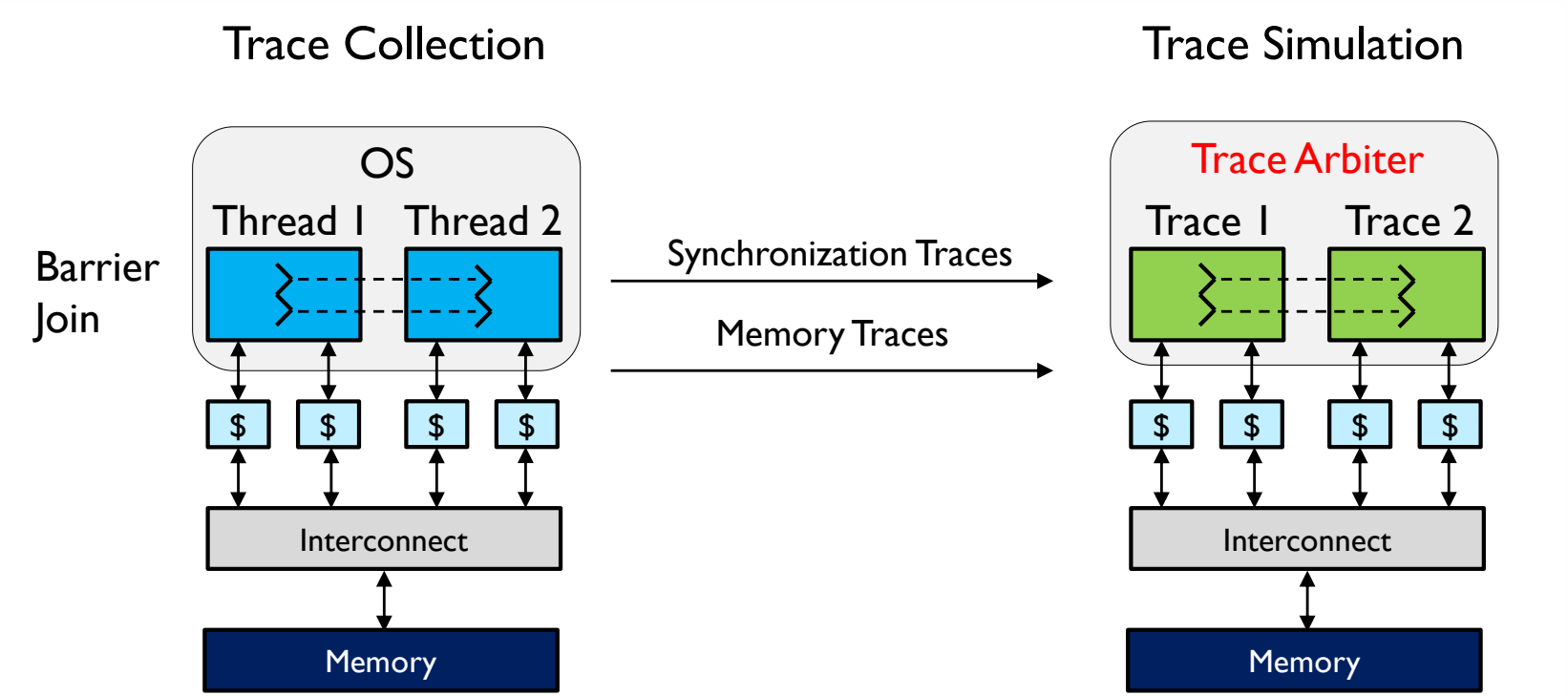
- **Having these traces collected makes it easy to:**
  - Perform « Trace replication » i.e. emulate more CPU cores for scalability study
  - This corresponds to *weak scaling* experiments, i.e. per-core workload remains same



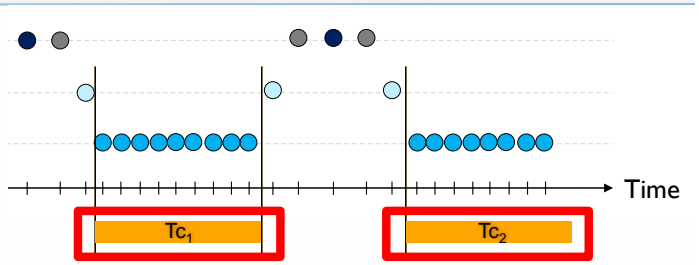



# Multithreaded applications

- **Yet synchronizations must be accounted for!**
  - Using whatever API: POSIX threads, OpenMP 3.0 …
  - Approach: embed synchronizations into traces
  - Have an arbiter that takes care of locking (when barrier reached) and unlocking TIs

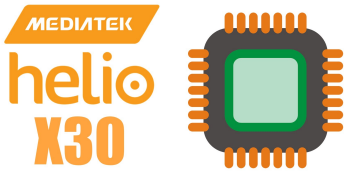


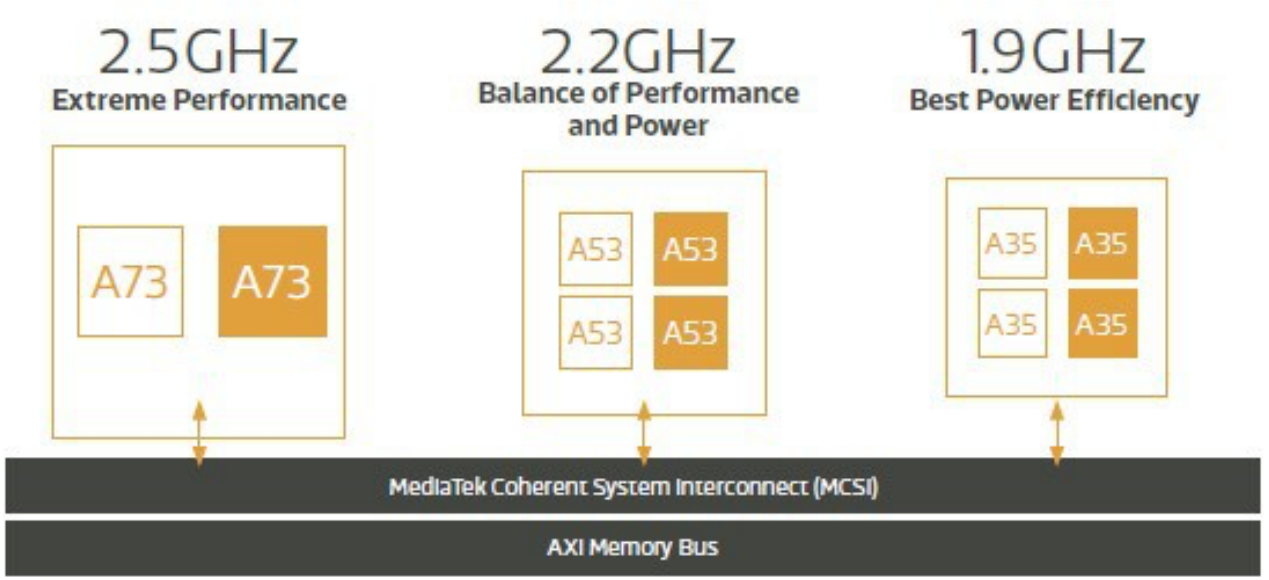
# Limitation: in-order only!




- **And most ARM AP are OoO (Out-of-Order)**
  - Meaning multiple outstanding memory transactions
  - The assumption of constant time btw. 2 misses does not hold

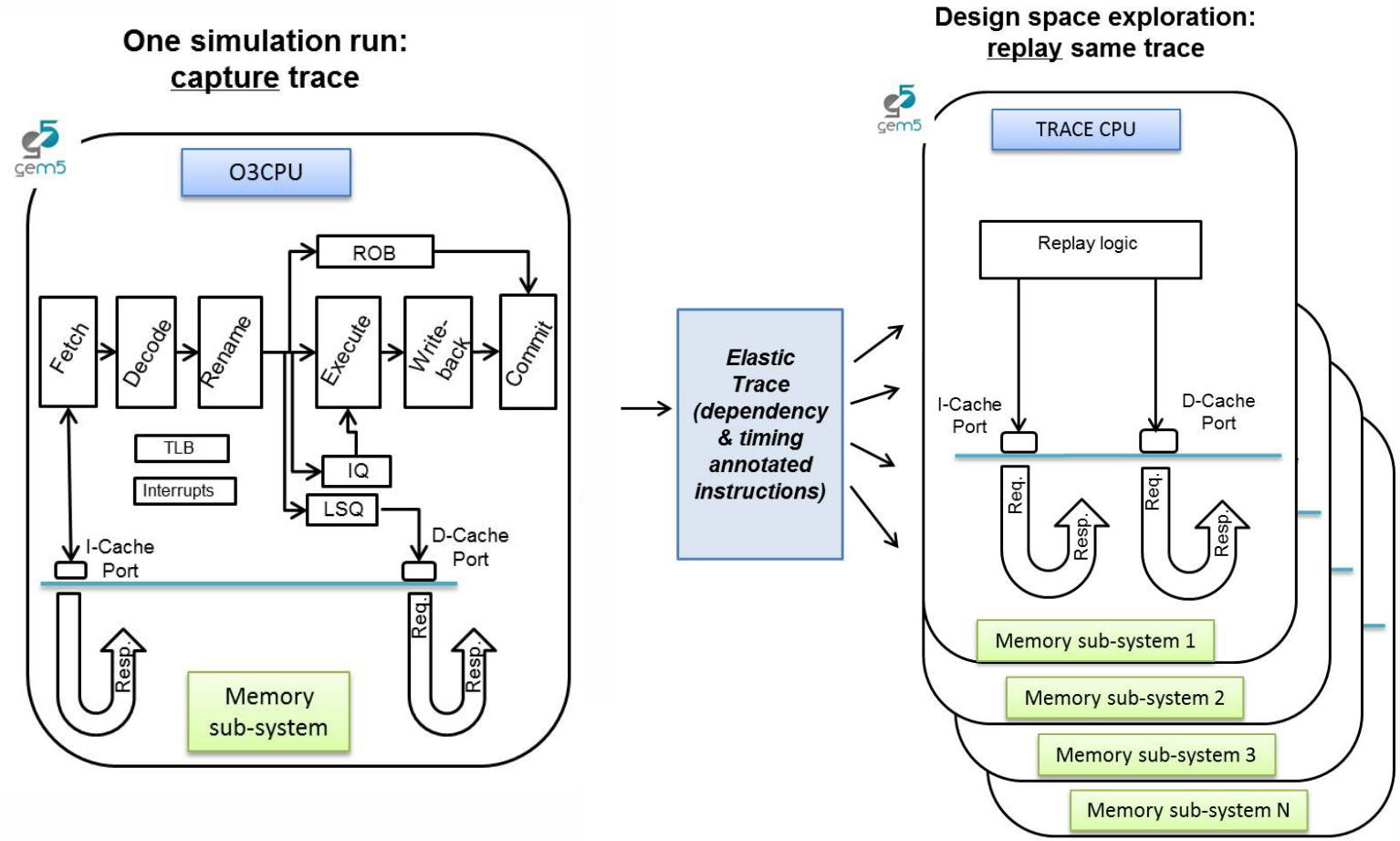
- **big-LITTLE & other heterogeneous friends everywhere**
  - And there microarchitecture details cannot be overlooked



# Elastic Traces: Trace-driven simulation for OoO

- **Modeling micro-architecture timing & dependencies**
  - Tracing with O3 model + probes, without L2 cache
  - Replay done in a smart « elastic »fashion

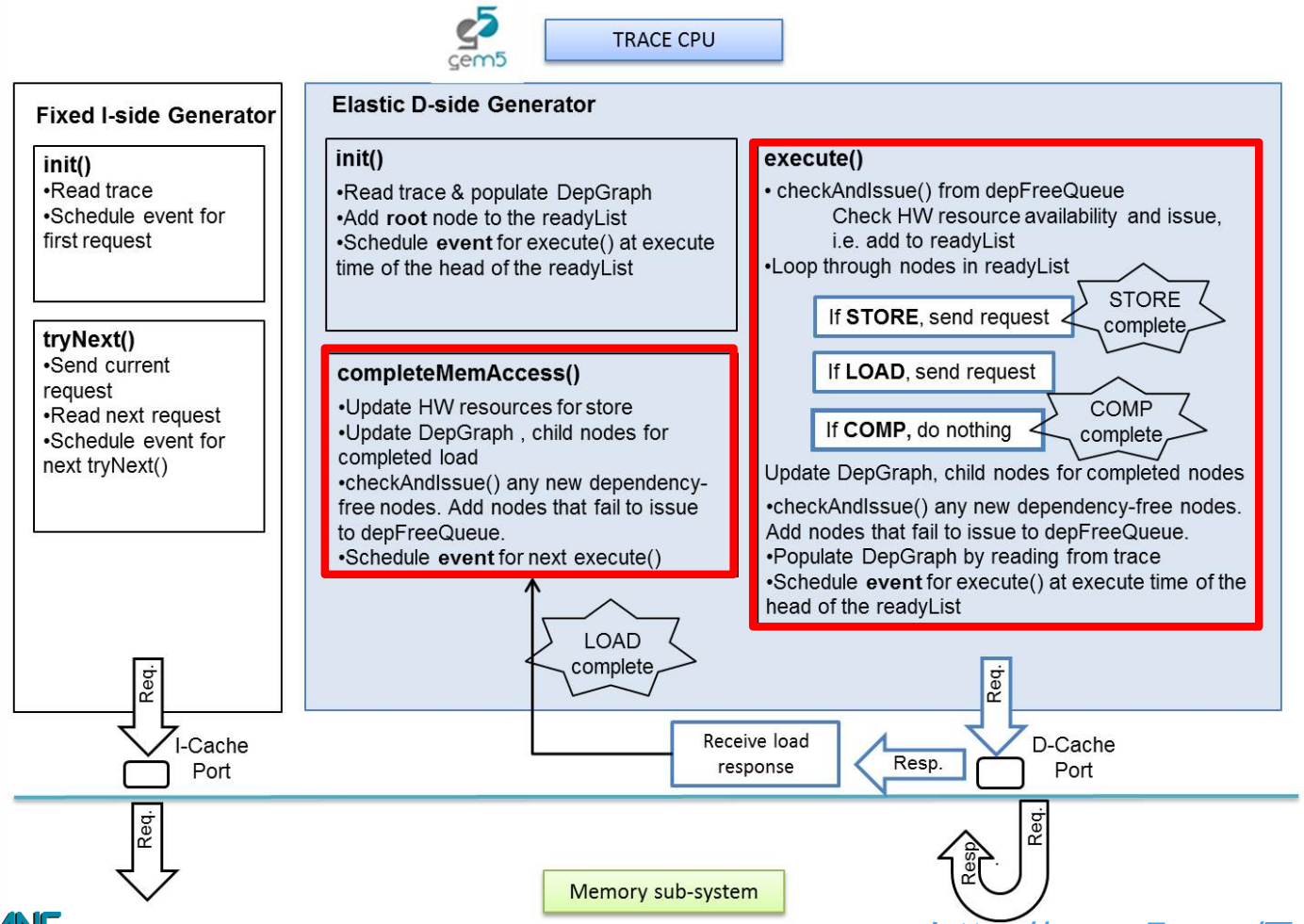



http://gem5.org/TraceCPU

# Elastic Traces: Trace-driven simulation for OoO

## Smart TraceCPU

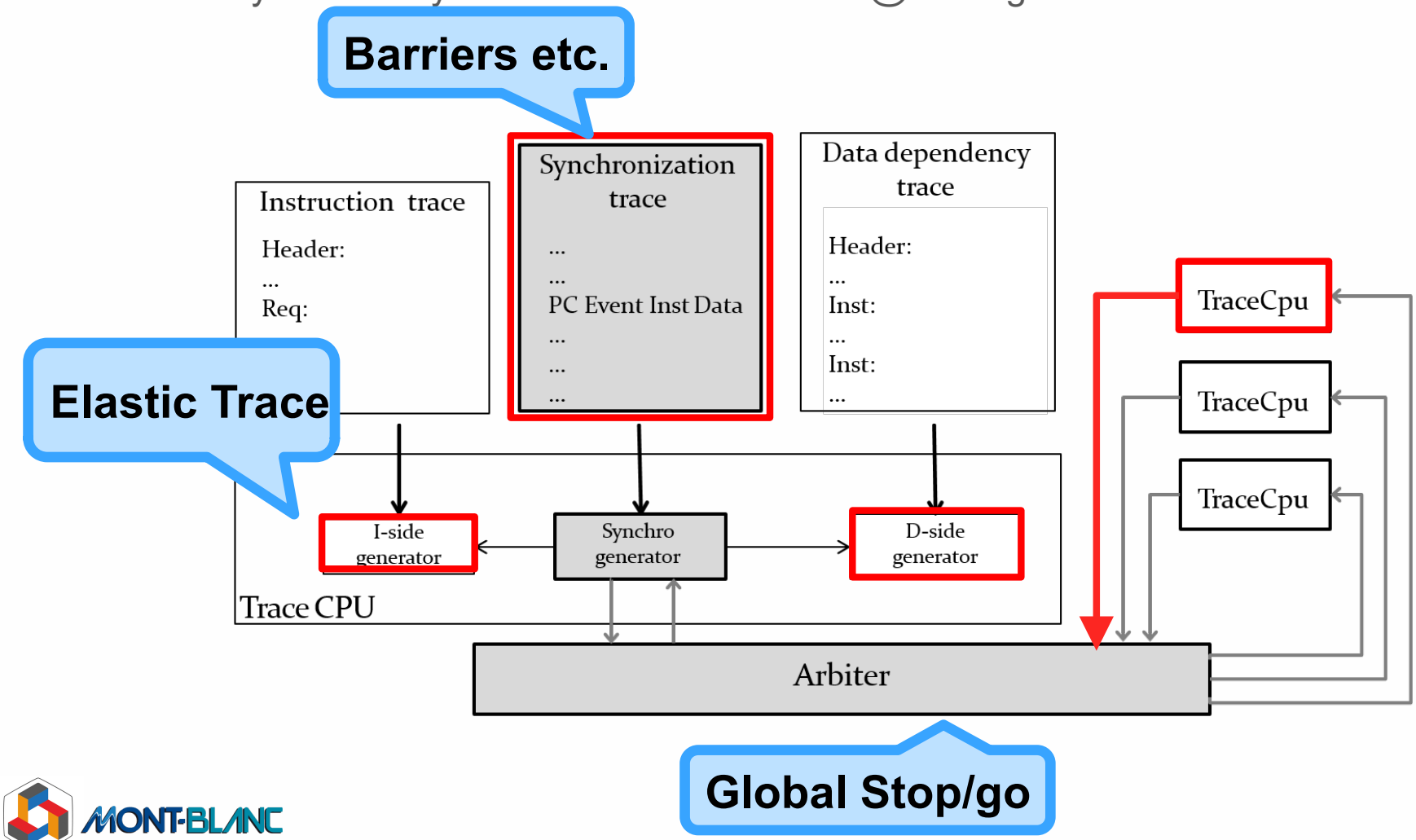
- Updating a dependency graph pushing ready instructions into a queue for issue




http://gem5.org/TraceCPU

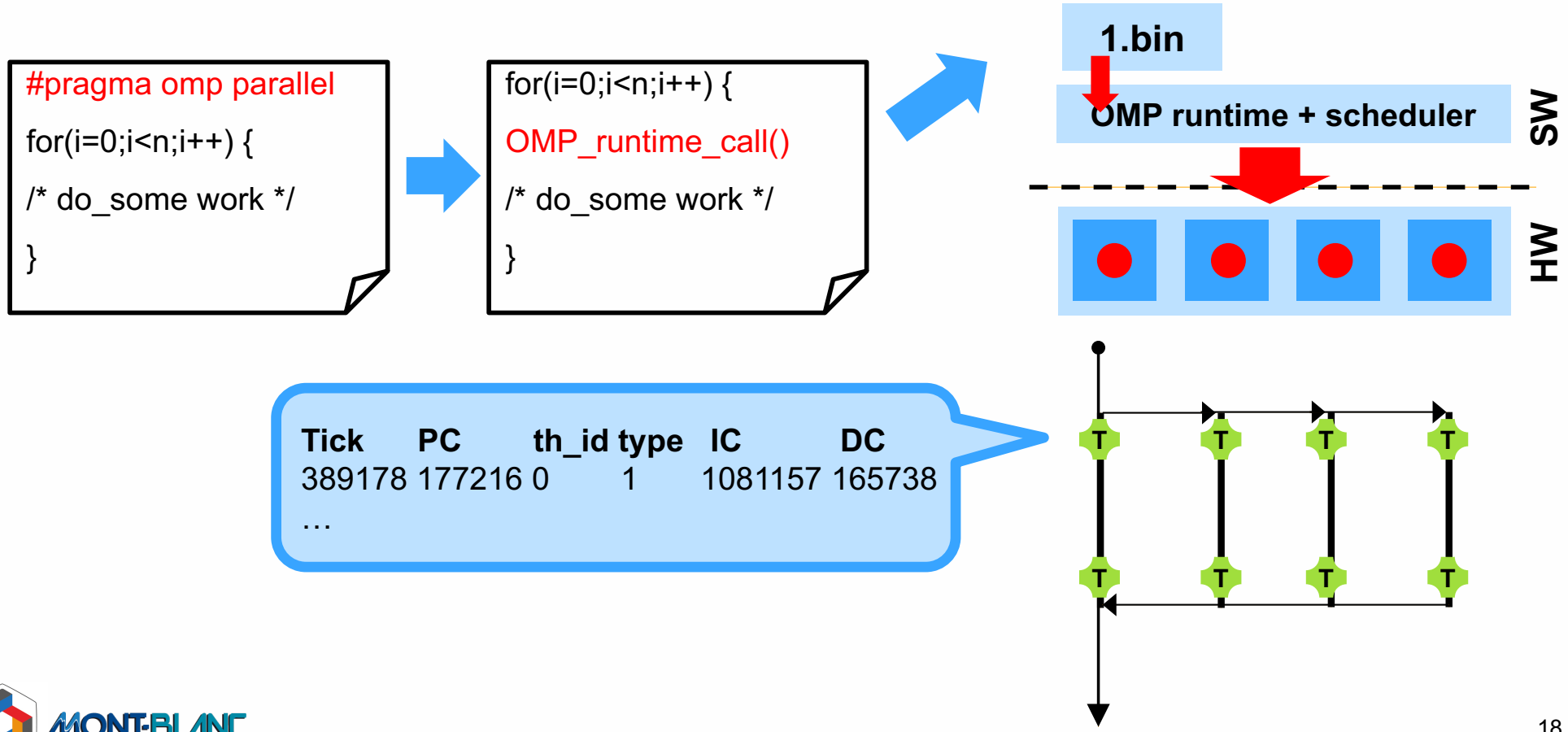
# ElasticSimMATE (ESM)

- **SimMATE + Elastic Traces = ElasticSimMATE**
  - Enabling both OoO + multithreaded applications
  - Key: embed synchronization information @ tracing time.



# ElasticSimMATE (ESM)

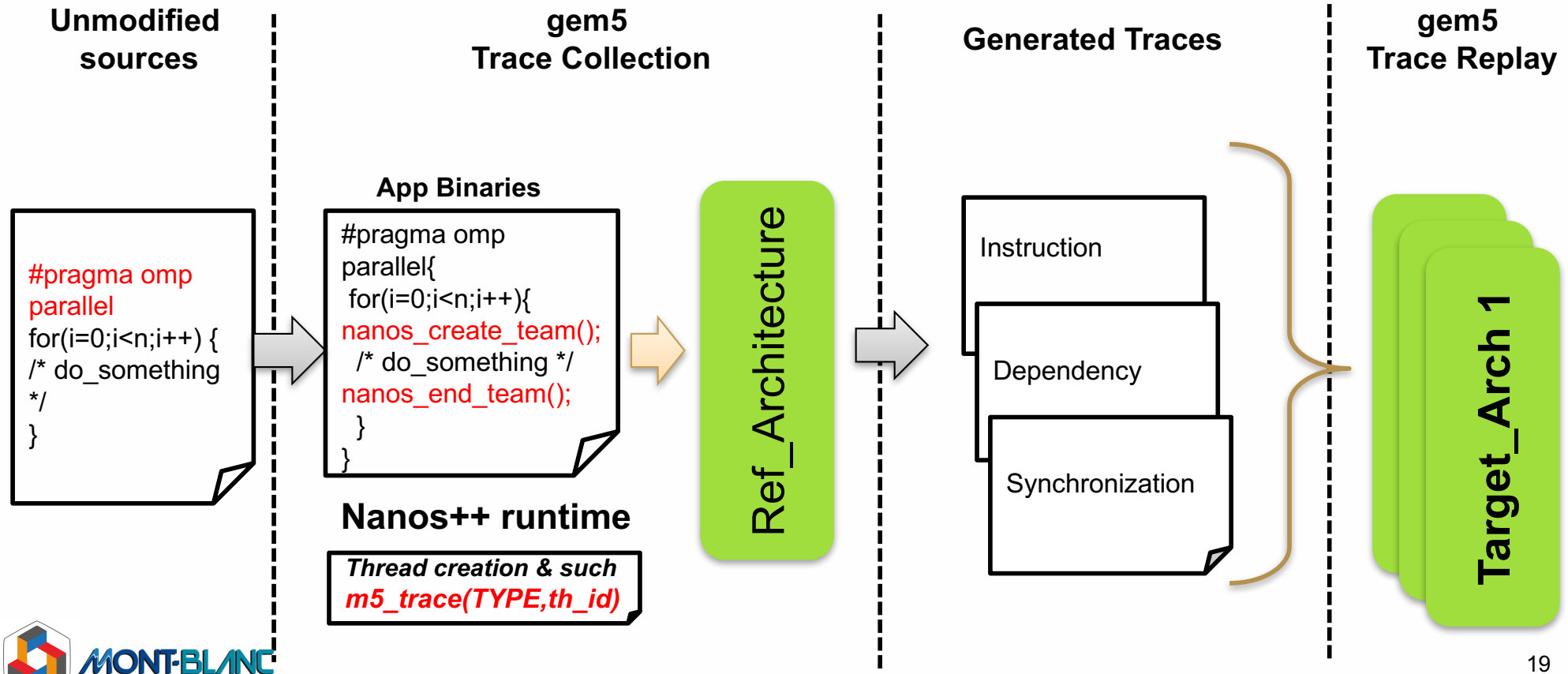
- **Proper tracing of synchronizations**
  - API-dependant: **OpenMP 3.x**
  - Tracing whenever entering or leaving parallel region, barrier etc.



# ESW wrapup

## ESM flow wrapup

- Using BSC **Mercurium compiler** / **Nanos++** runtime
- Tweaked runtime such that custom m5 pseudo instructions produce **trace records**

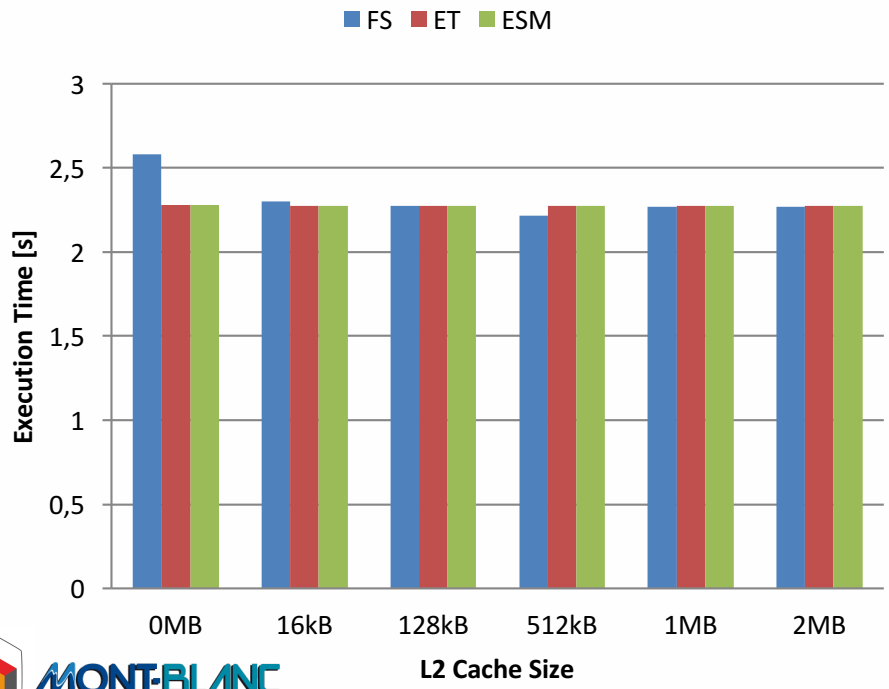


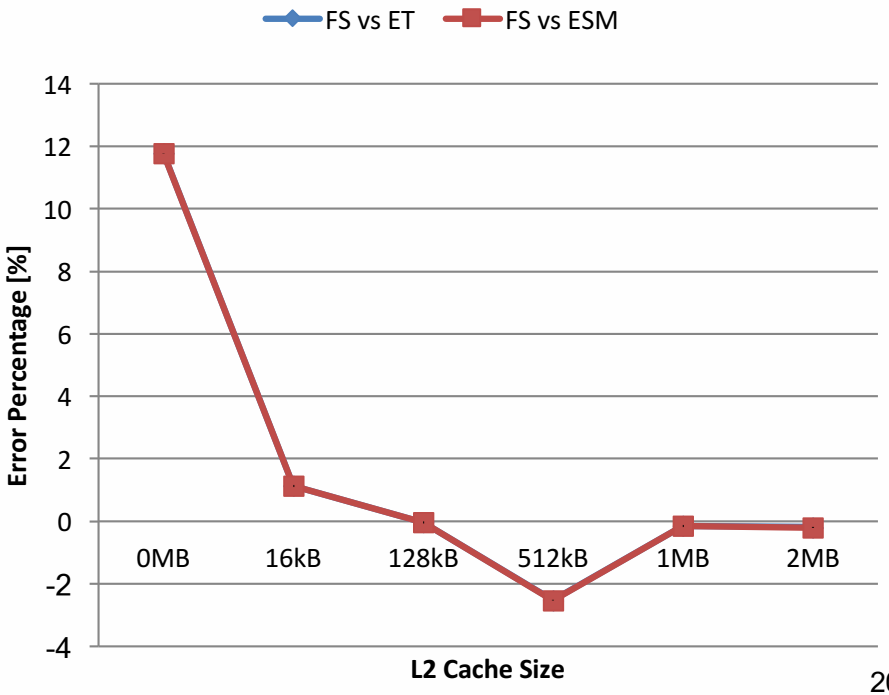
# Benchmarking

- **Two main use cases:**
  - **Fast parameter exploration**
  - Scalability study: « trace replication »
- **Speedup & accuracy?**
  - Experiments on Rodinia application kernels

1 core

**HOTSPOT**



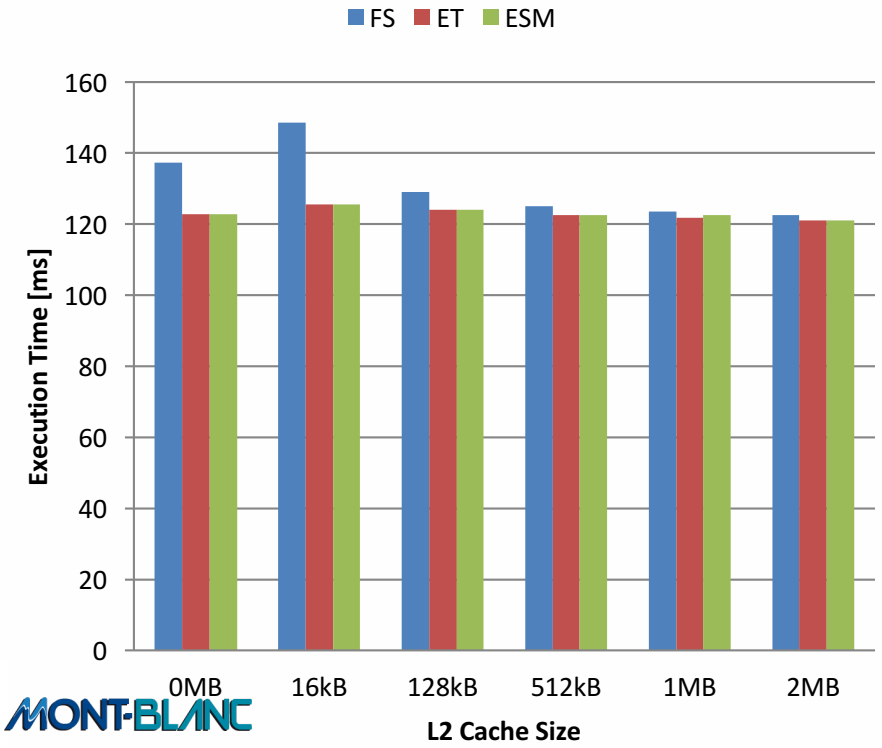



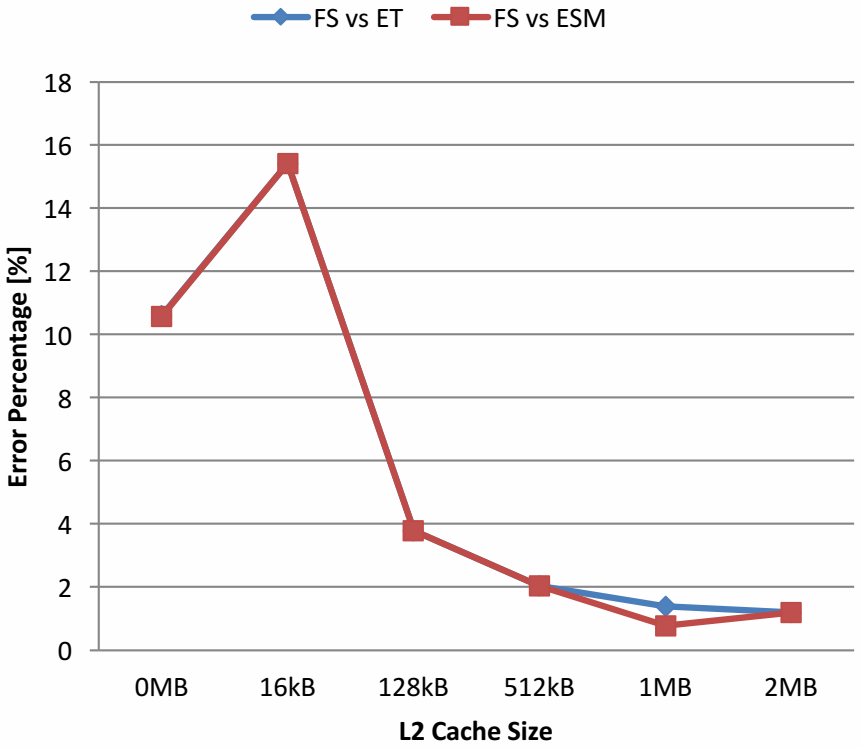
# Benchmarking

- **Two main use cases:**
  - **Fast parameter exploration**
  - Scalability study: « trace replication »
- **Speedup & accuracy?**
  - Experiments on Rodinia application kernels

1 core

**KMEANS**



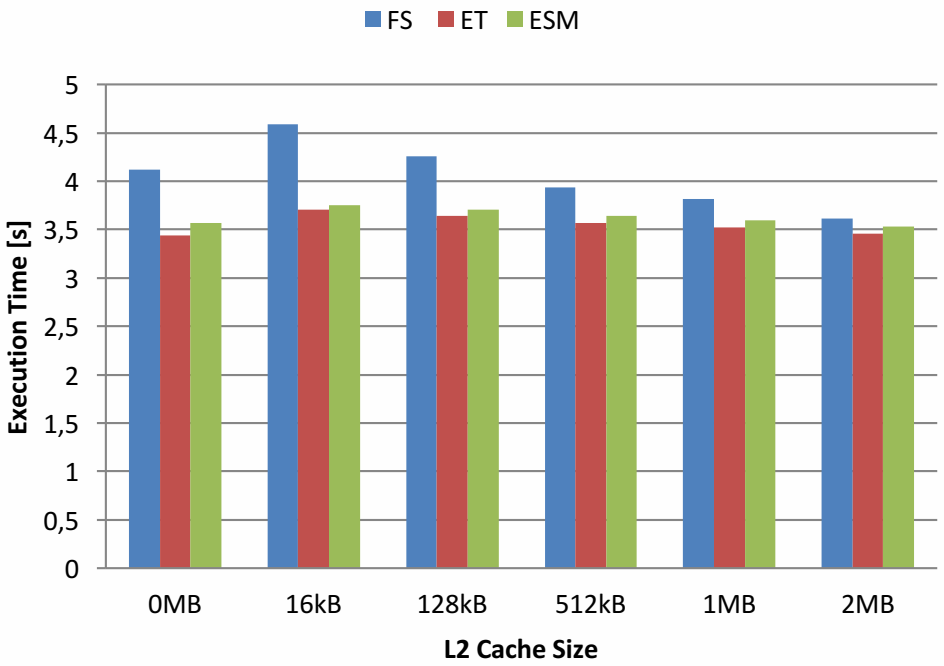



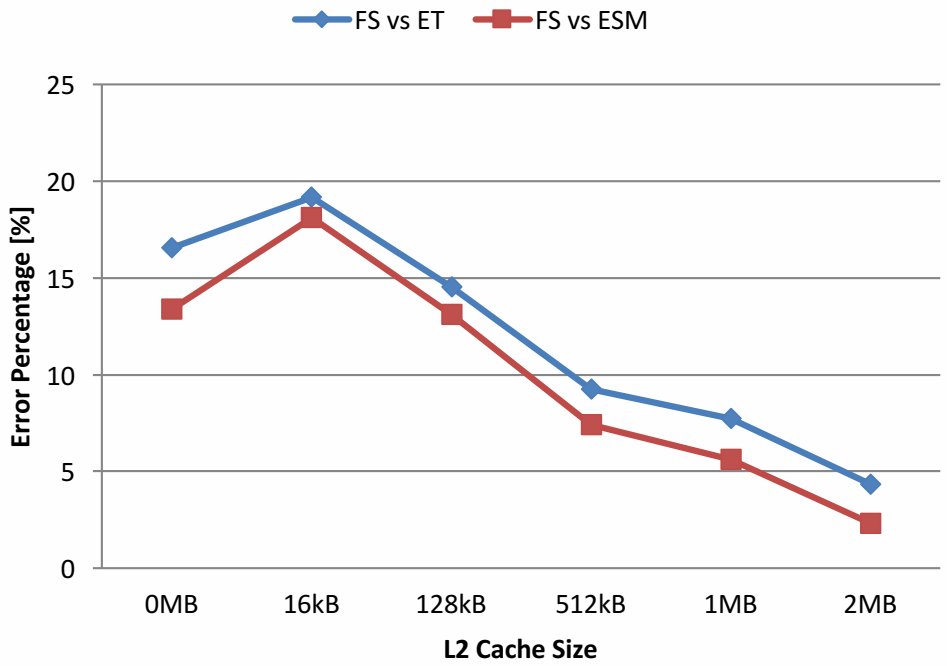
# Benchmarking

- **Two main use cases:**
  - **Fast parameter exploration**
  - Scalability study: « trace replication »
- **Speedup & accuracy?**
  - Experiments on Rodinia application kernels

1 core

**CANNEAL**

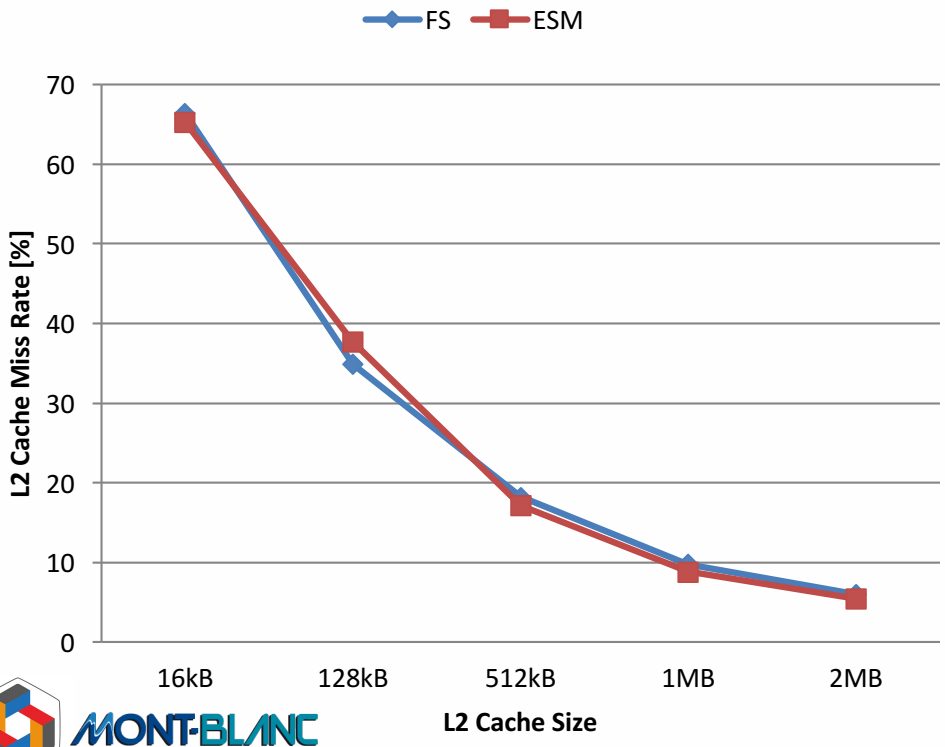


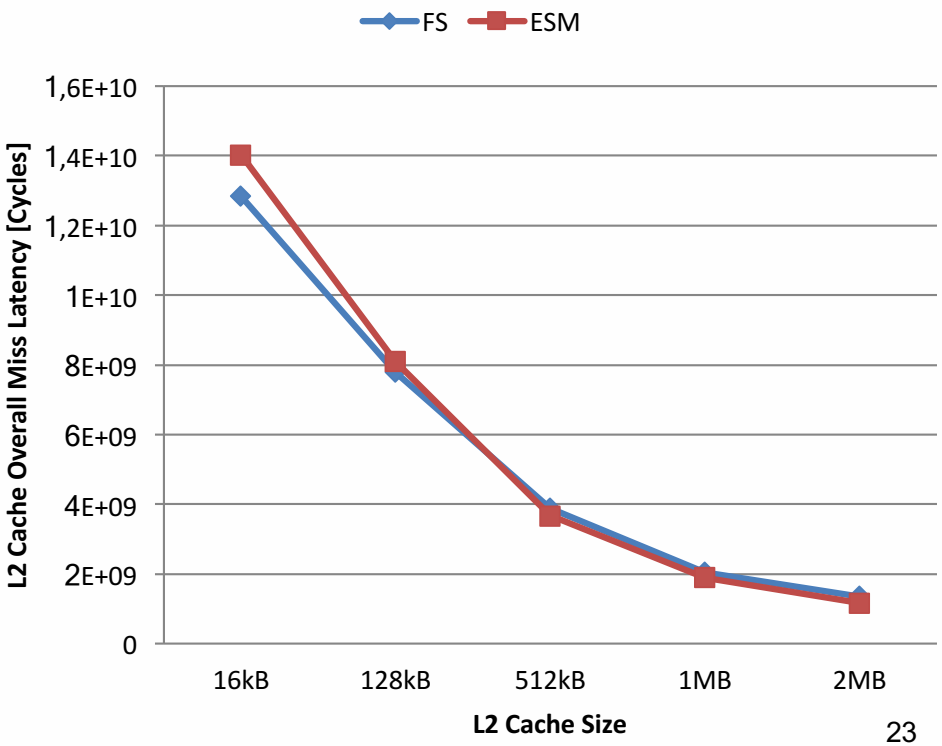



# Benchmarking

- **Two main use cases:**
  - **Fast parameter exploration**
  - Scalability study: « trace replication »
- **Speedup & accuracy?**
  - Experiments on Rodinia application kernels

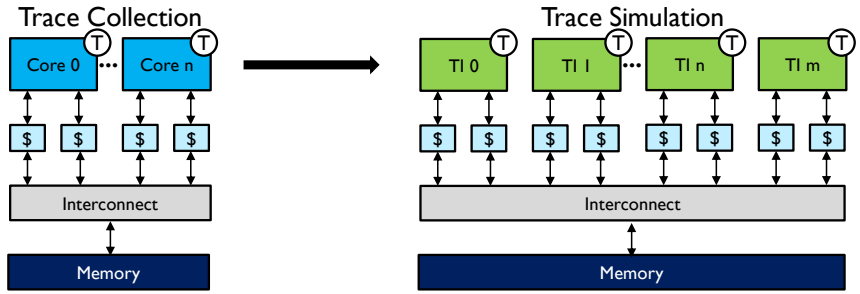
**L2** **KMEANS**



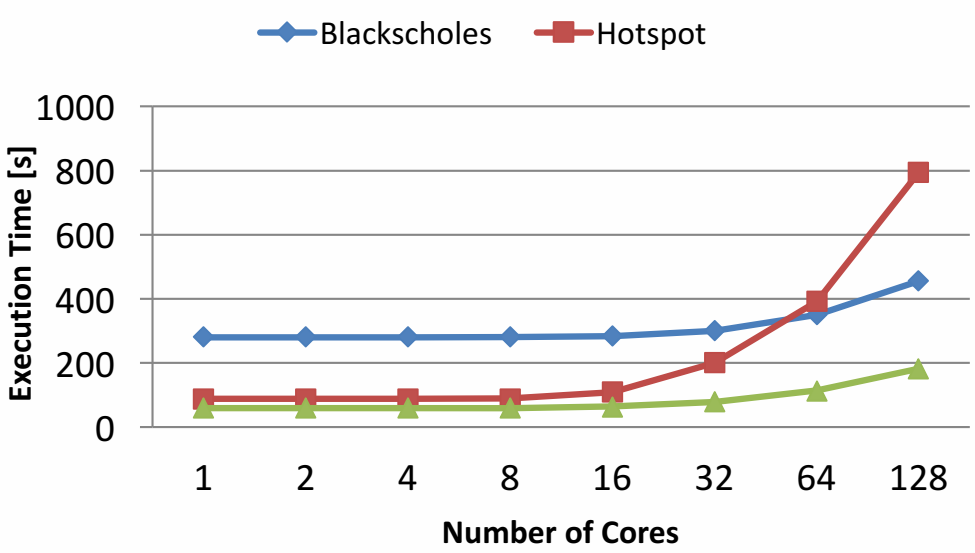
# Benchmarking

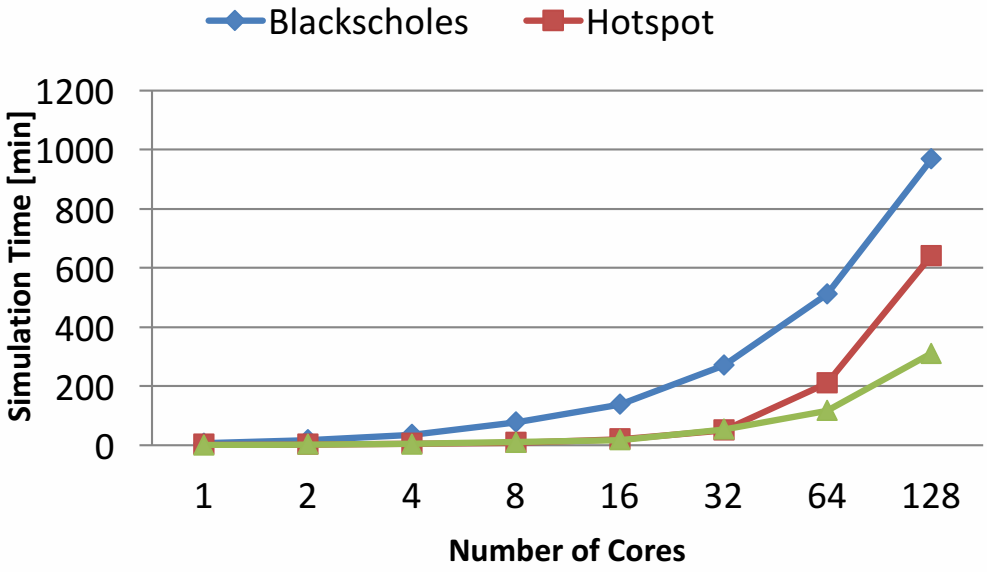
- **Two main use cases:**
  - Fast parameter exploration
  - **Scalability study: « trace replication »**




- **Speedup & accuracy?**
  - Experiments on Rodinia application kernels

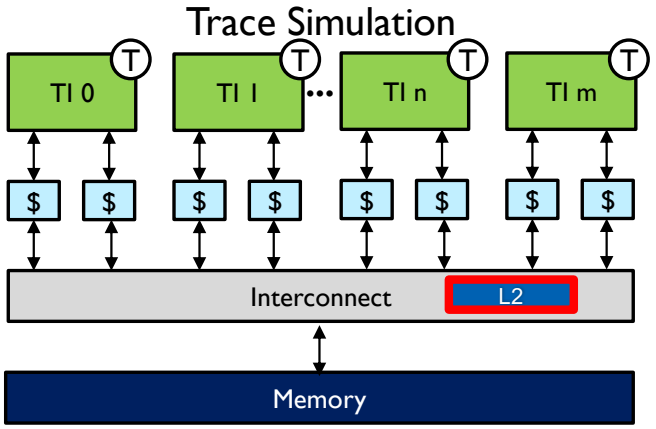
**128 cores** **KMEANS**



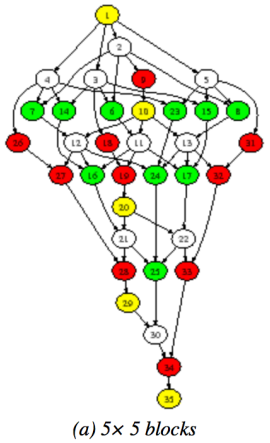
# Perspectives

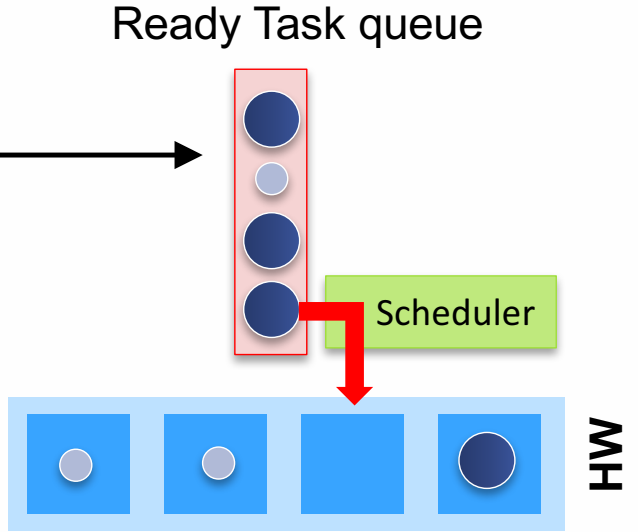
- **Scalability analysis has limits**
  - Requires additional features s.a. **address offsetting**
  - **Weak scaling** only (replicated per-core workloads)




- **Programming models moving from loops to *tasks***
  - OpenMP 4.0, OmpSs
  - Still pragma-based
  - More parallelims available at run-time … more opportunities for smart job scheduling

```
#pragma omp task depend(in: x) depend( out: y) depend(inout: z)
```

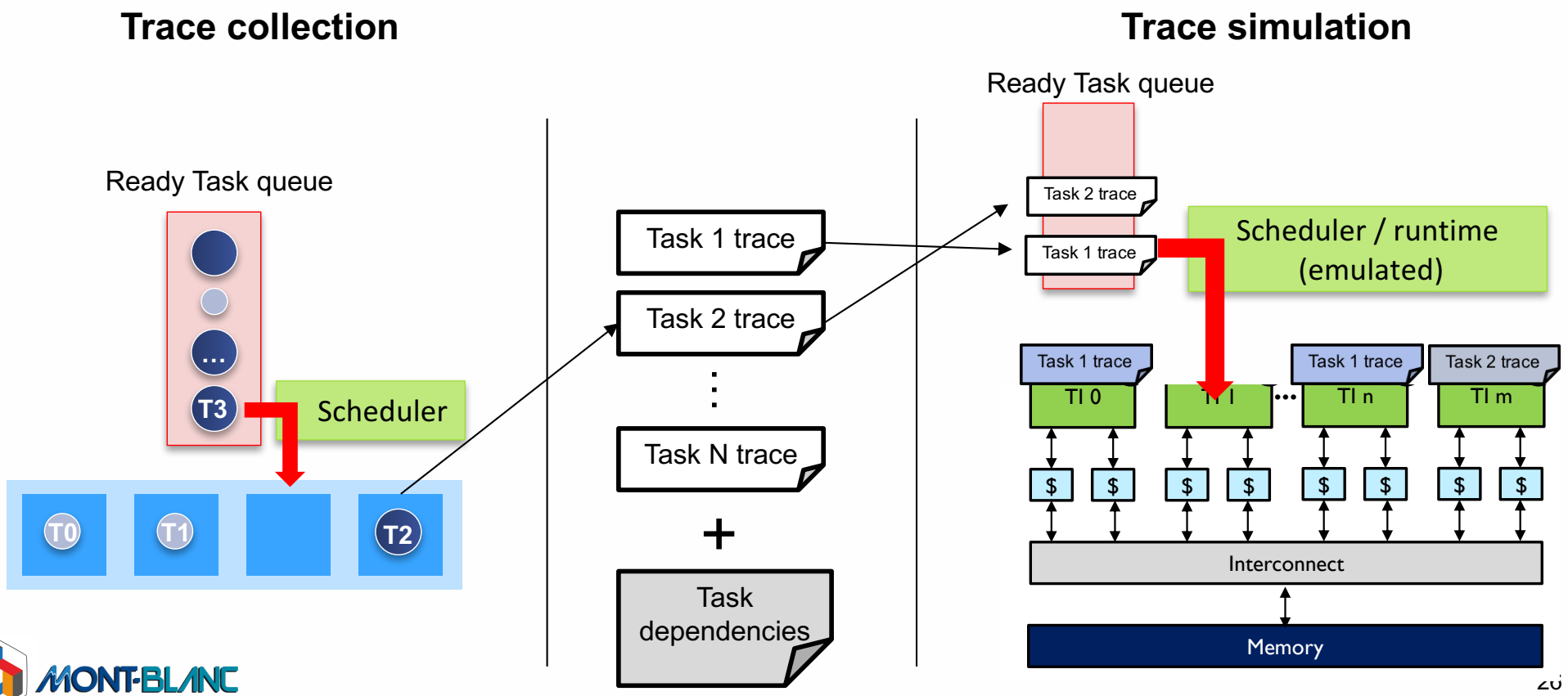

(a) 5× 5 blocks



# Perspectives

## Unbinding traces from cores

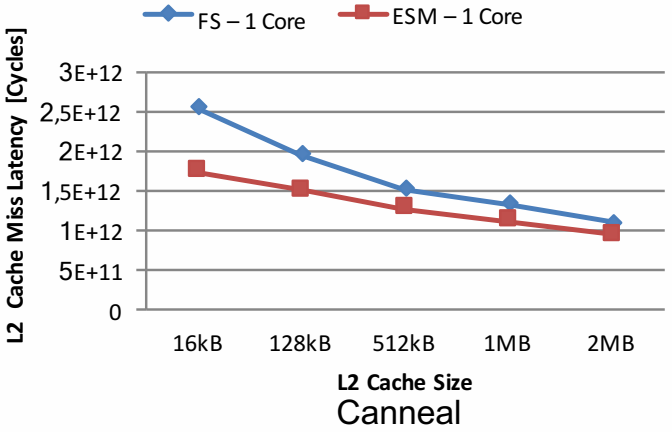
- One **trace** per **Task,** not per core!
- Assign traces to cores by emulating runtime behaviour in trace replay
- This is real strong scaling



http://montblanc-project.eu

## Current ESM prototype

- ~5x - 10x speedup for low core count, probably more for tens / thousands
- Nice solution for fast DSE
- Remaining accuracy issues for some applications
  - Common to ESM & Elastic Traces
  - Under investigation with ARM



Canneal

## Use cases

- Exploration of memory subsystem
- Some microarchitecture parameters (Elastic Traces)
- …

## Future directions

- Ruby compatibility
- Could be combined with other initiatives (dist-gem5)
- Can be extended to other PM / APIs (Tasking, MPI…)