MEEP

MareNostrum Experimental Exascale Platform

Spike+Sparta: Developing a scalable RISC-V simulation infrastructure for HPC architectures

Borja Pérez

29/06/2020

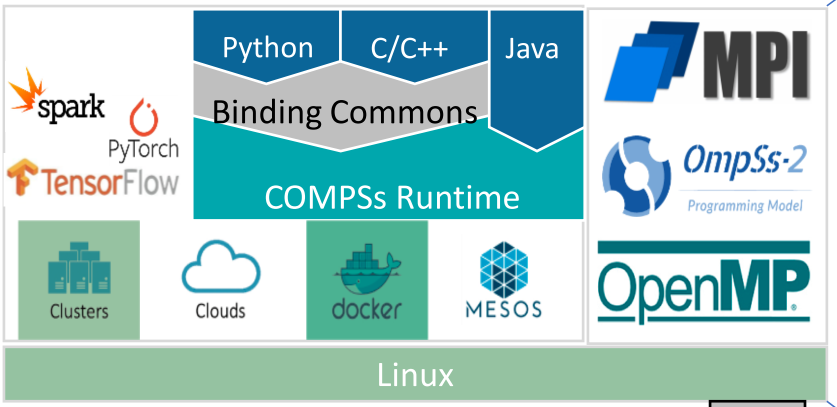
The MEEP project has received funding from the European High-Performance Computing Joint Undertaking Joint Undertaking (JU) under grant agreement No 946002. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Croatia, Turkey.

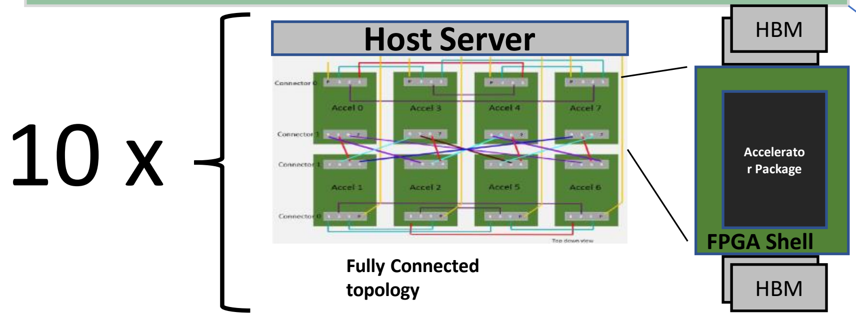
MEEP Goals

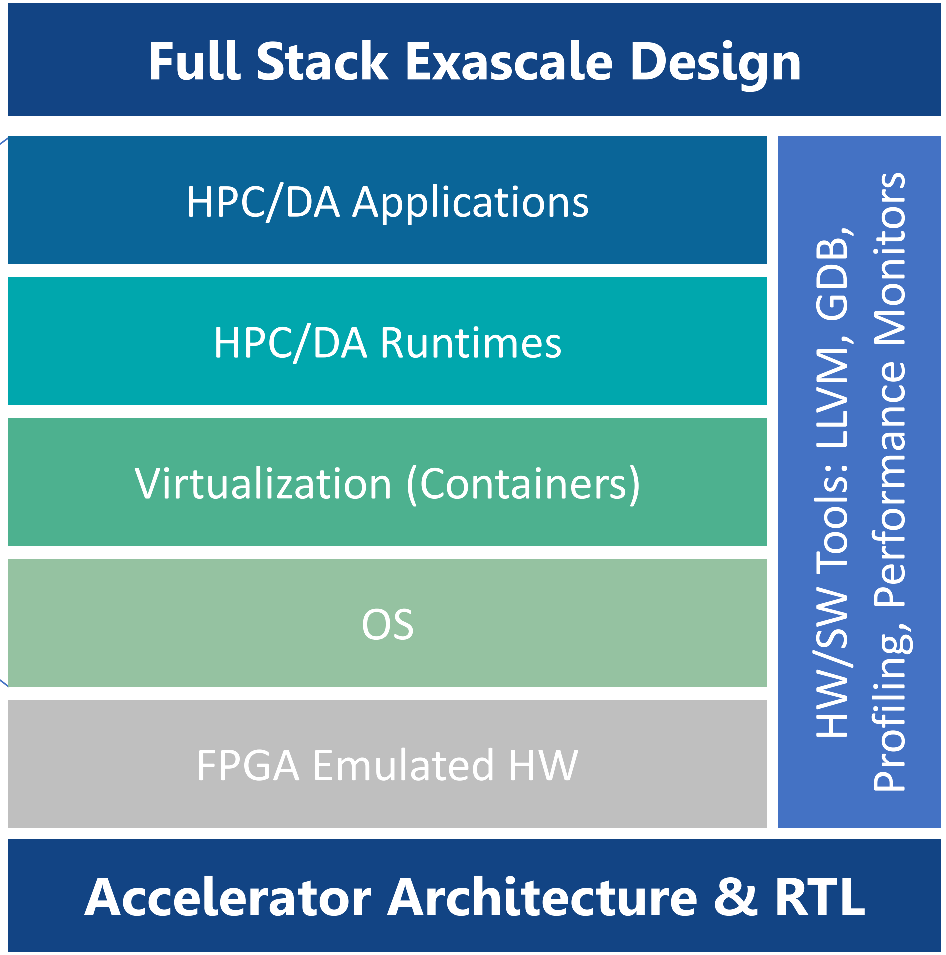
- What type of HPC Exascale accelerator would we build in 2025?
  - MEEP is a flexible FPGA-based emulation platform that will explore hardware/software co-designs for Exascale Supercomputers
    - Software Development Vehicle
    - An evaluation platform of pre-silicon IP and ideas, at speed and scale
      - Efficiently Map RTL to FPGAs (not gate2gate)
- Flexible and Reusable
  - Enable other SW and HW projects @ BSC and in the EU
    - Physical platform
    - Emulation IP
    - Open SW and HW IP

MEEP in a Nutshell: A Co-Design Laboratory

1. Software Ecosystem
2. Hardware (RTL)
3. FPGAs

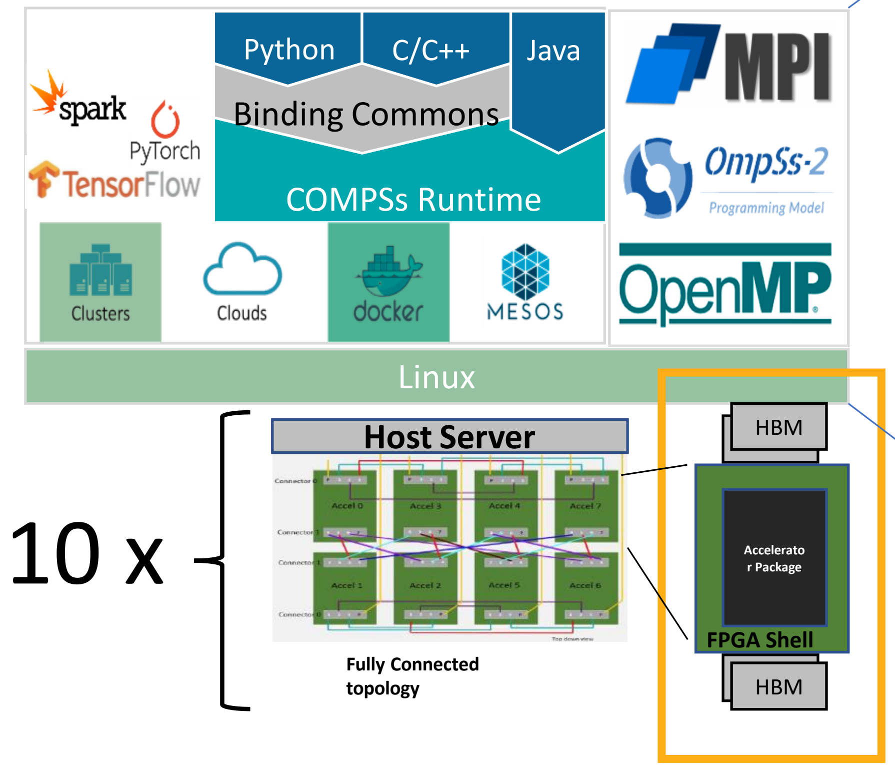


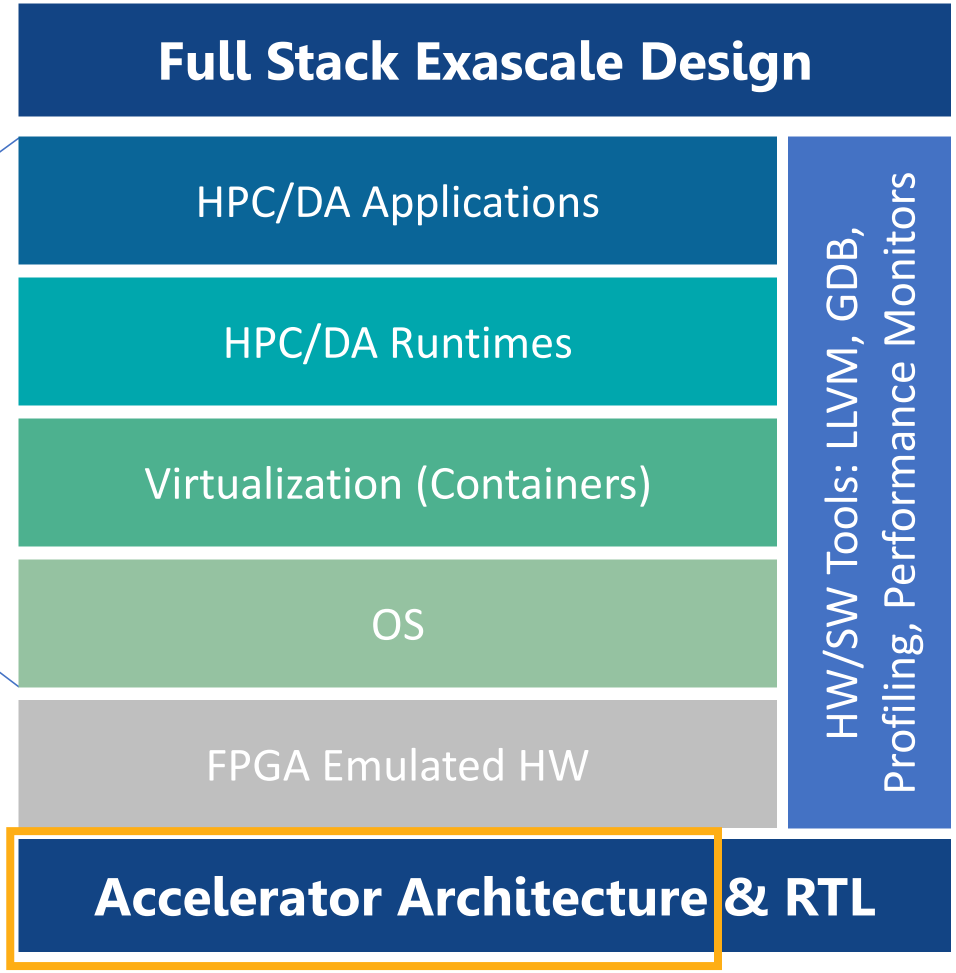




MEEP in a Nutshell: A Co-Design Laboratory

1. Software Ecosystem
2. Hardware (RTL)
3. FPGAs



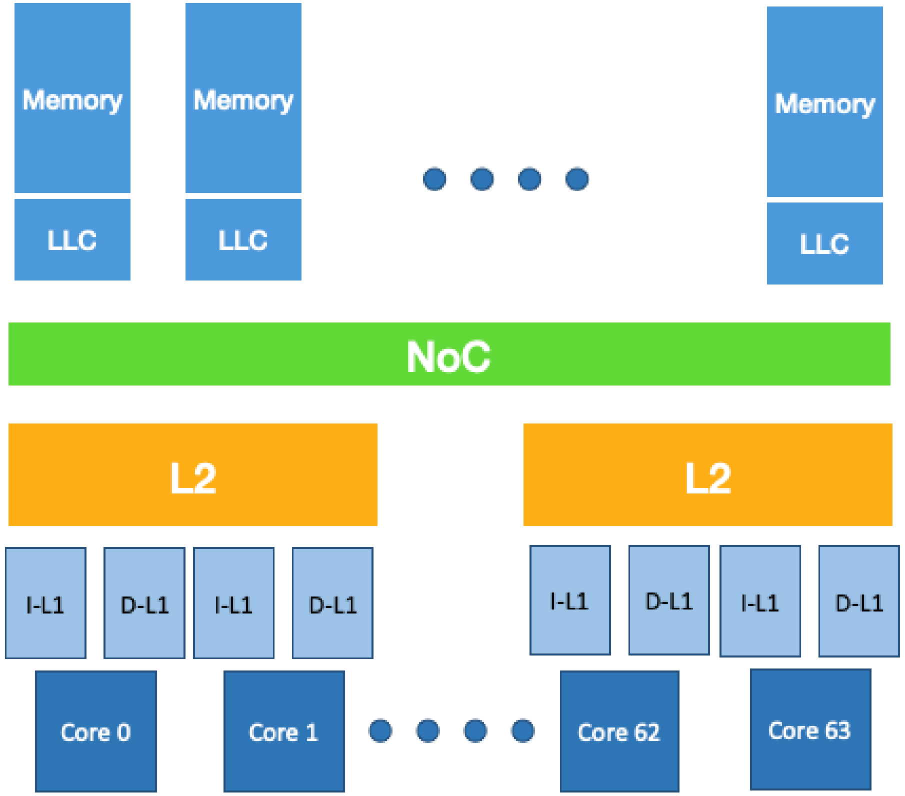


Goals

- A simulation infrastructure for design space exploration of HPC architectures.
    - RISC-V support.
        - Both scalar and vector instructions.
    - High simulation throughput. Get the level of detail/performance that we need.
    - Flexible and easy to extend.
    - Leverage existing tools.
    - **Scalable**

Modelling needs

- We target HPC-oriented architectures:
  - Significant number of cores.
  - Deep memory hierarchy.
- Very different configurations and techniques might be considered.
  - How is the L2 distributed?
  - What does the NoC look like?
  - Prefetching? Streaming?
  - …
- Special focus on the memory side.



An Approximate Simulatable 64-Core Slice of the Chip

Leveraging existing tools

- Leveraging existing open-source tools adds value to the community.
  - QEMU
  - Gem5
  - Renode
  - Imperas
  - EPI simulation tools (Tasksim)
  - Spike
  - SST
  - Sparta
  - …

Leveraging existing tools

- QEMU
  - Fast execution time.
  - Support for the RISC-V vector extension is unclear.
- Gem5
  - Too slow for our current needs.
  - Not straightaway to extend.
- Renode
  - Focused on embedded systems.
  - No support for vector instructions.
- Imperas
  - Vector instructions supported.
  - Partially non-open source.

Leveraging existing tools

- Tasksim (BSC)
  - Workflow
    - Execute in real risc-v hardware (hifive platform)
      - Only trace vector instructions via traps.
    - Feed the output trace to the simulator.
    - Obtain a paraver output trace.
  - Basic memory hierarchy is modelled that could be extended to our needs.
  - Cons:
    - Scalar instructions cannot currently be simulated.
    - Simulation throughput.

Leveraging existing tools

- Spike
  - Golden standard for RISC-V functional simulation.
  - Can simulate multicore systems.
    - Only in baremetal mode.
  - Can simulate vector instructions.
    - There is no compiler support, so need to be programmed in assembly code.
  - Fast: ~100 MIPS
  - Cons:
    - Only models a very simple memory hierarchy.
    - Purely functional.

Leveraging existing tools

- SST (Structural Simulation Toolkit)
  - Not a simulator, but a framework.
  - 20+ modules to choose from for different levels of detail and kinds of simulation:
    - Pattern-driven.
    - Trace-driven.
    - Execution-drive.
    - Different cache configurations.
    - Different memory technologies (DDR, NVM, HBM…).
    - NoC.
  - Cons: Very limited RISC-V support.
    - Only single core.
    - Very slow.

Leveraging existing tools

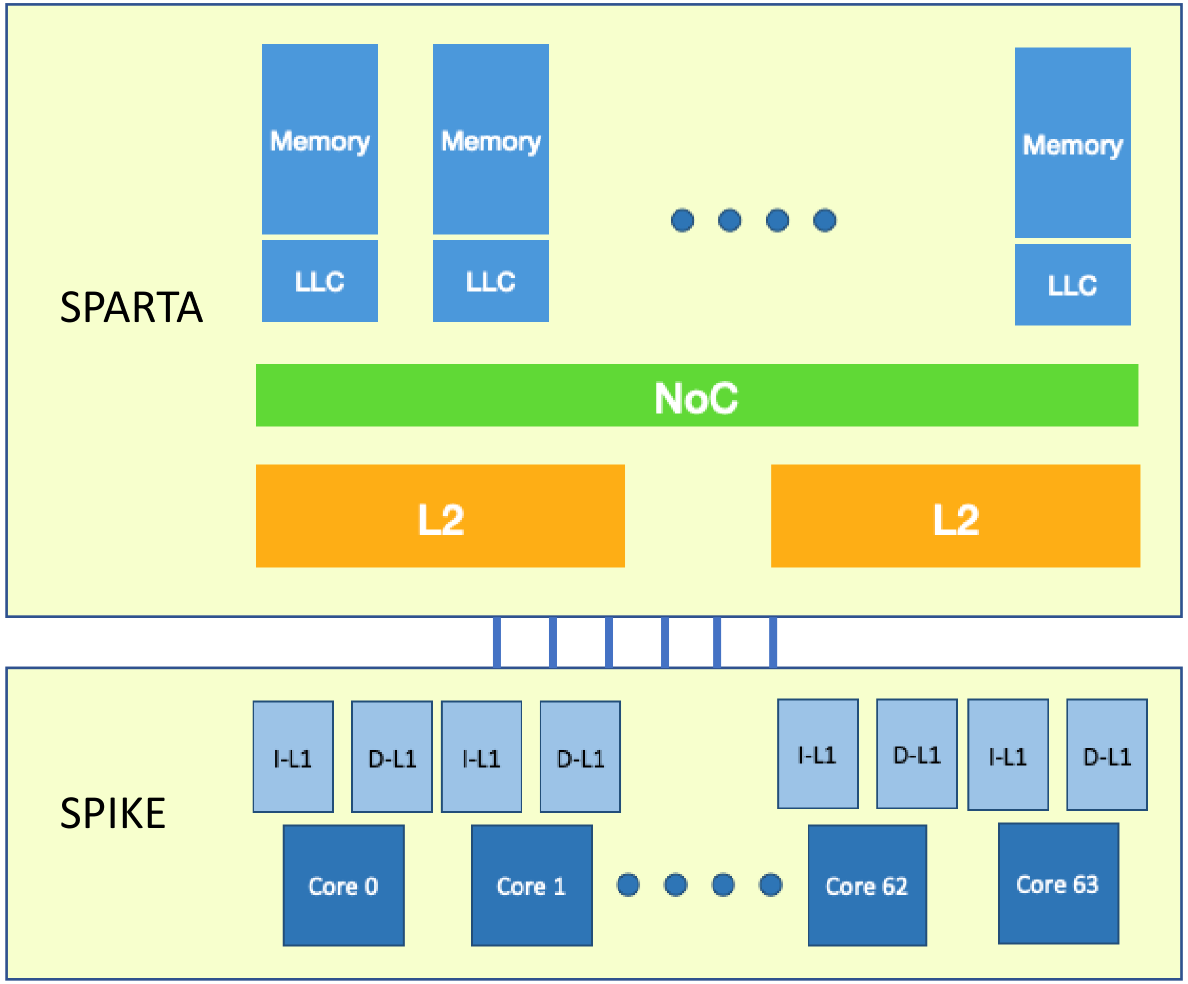
- Sparta
  - Currently developed by Si-Five.
    - History of the tool goes back 20 years.
  - Industry grade code.
  - Also a framework to build event driven simulators.
    - Does not offer modules, but building blocks.
  - Offers lots of useful capabilities.
    - Checkpointing.
    - Visualization.
  - Cons: No RISC-V support.

Leveraging existing tools

- No current simulation infrastructure is capable of the needed scalability/flexibility.
- Spike is the golden standard for RISC-V.
- Sparta offers the tools to build the performance model we need.
- Let’s put them together.
- But…
  - The samples provided with Sparta are trace-driven and focus on the core.
  - Spike is a purely functional simulator. It does not know about timing or events.

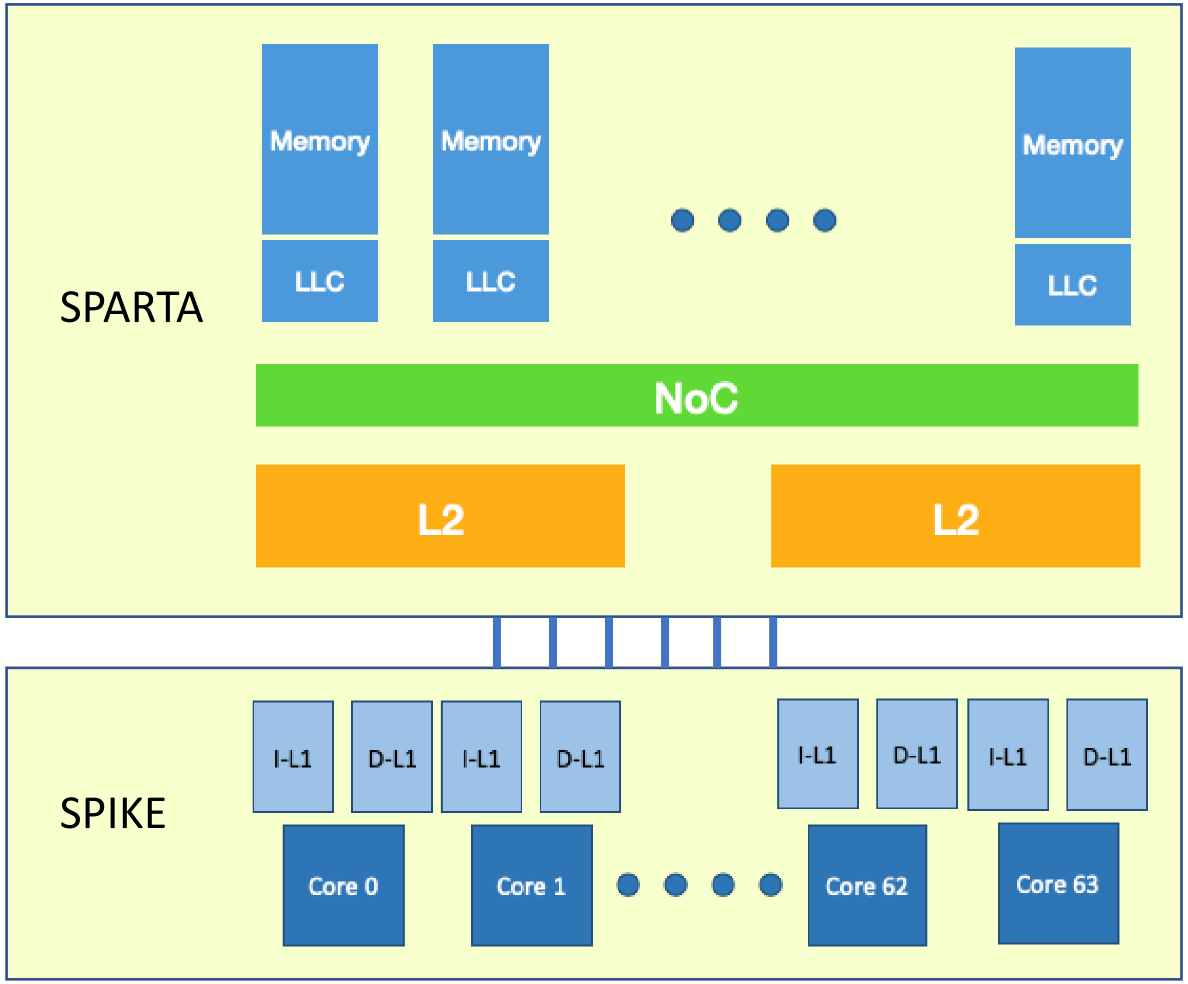
Integration approach

- Spike:
  - Handles functional execution (baremetal).
  - Models the L1s to reduce interaction with Sparta.
- Sparta:
  - Simulates instructions using Spike.
  - Handles L1 misses.
  - Keeps track of time.



Overall functioning

- Sparta controls the simulation.
- Every cycle, simulate an instruction on each active core.
- If no RAW dependency is detected:
  - Check for L1 misses and generate events.
  - Schedule the next cycle for the core.
- If there is a dependency:
  - Set core as inactive until it is satisfied.



The Spike side

- The *SpikeWrapper* class extends the Sparta Unit and is the interface with Spike.
- It offers three basic methods to:
  - Setup the spike simulation.
    - L1 cache configuration modified from the original.
  - Simulate a single instruction on a core.
    - Returns whether a RAW was detected and potentially generates L2 requests.
  - Set the destination register of an L2 request as available for RAW tracking.

The Spike side

- Some surgical modifications were necessary to achieve the desired functionality.

<table>
<tr><th>Functionality</th><th>Affected Class</th><th>New code</th></tr>
<tr><td>Setup a simulation and have it wait.</td><td>sim_t</td><td rowspan="2">~150 lines</td></tr>
<tr><td>Simulate instruction by instruction, on an individual core.</td><td>sim_t
processor_t</td></tr>
<tr><td>Keep track of RAW dependencies.</td><td>regfile_t
Decode macros</td><td>~50 lines</td></tr>
<tr><td>Log L1 misses</td><td>mmu_t *
cache_sim_t *</td><td>~60 lines</td></tr>
</table>

** An alternate implementation that does not modify these classes would be possible.*

The Sparta side

- Only base Sparta classes were used:

Functionality	Sparta Class
Wrap Spike	Unit
Encapsulate functionality	Unit
Model the L2 cache	Sample L1 cache
Handle events	Event
Communicate elements	DataInPort/DataOutPort
Gather statistics	Counter/StatisticDef

Results and next steps

- We can currently model L1s and a banked L2 with fixed latencies and get basic statistics:
  - Execution time.
  - L1 and L2 misses.
  - Miss rates.
  - Number of stalls due to dependencies.
- Tested using a regular multi-thread matrix multiplication:
  - Compiled with the standard riscv-gnu-toolchain.
  - Simulated up to 64 cores.
  - Simulation throughput ranging between 5 and 7 MIPS.
    - Running @ laptop, 5th gen. Intel i7, 16GB RAM.

Conclusions and next steps

- A scalable simulation infrastructure for the design space exploration is necessary.
  - Model different configurations.
  - Offer enough throughput to enable significant simulation.
- The Sparta+Spike integration is a viable solution that uses open source components.
  - No canonical simulation infrastructure is available.
- Can we get more simulation throughput?
- We are on track to simulate more complex systems. Let's start modelling them!

MEEP

MareNostrum Experimental
Exascale Platform

www.meep-project.eu

Thank you

borja.perez1@bsc.es