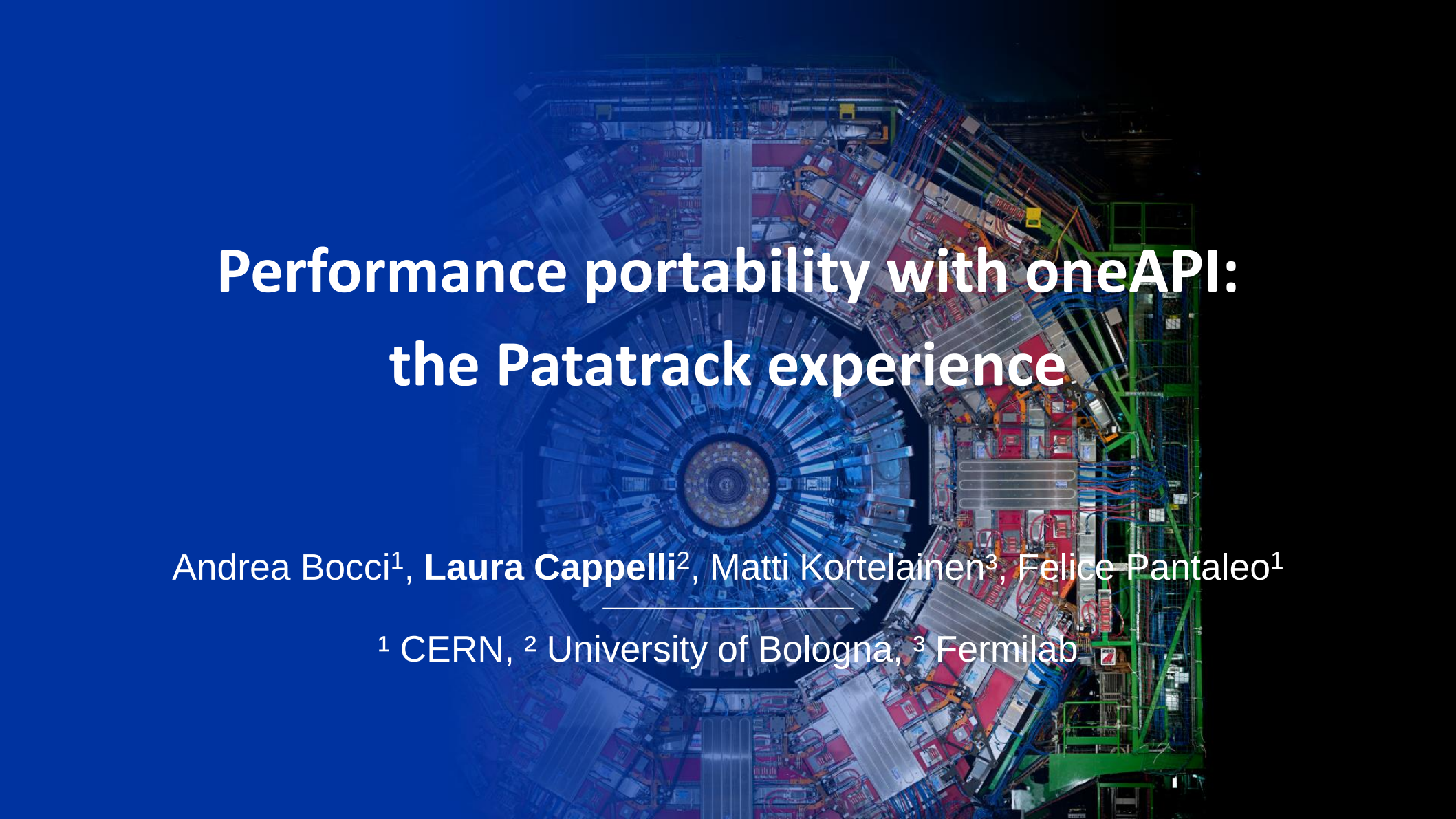
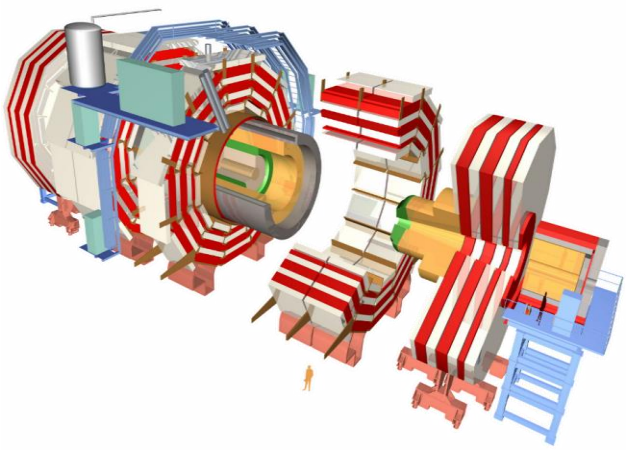


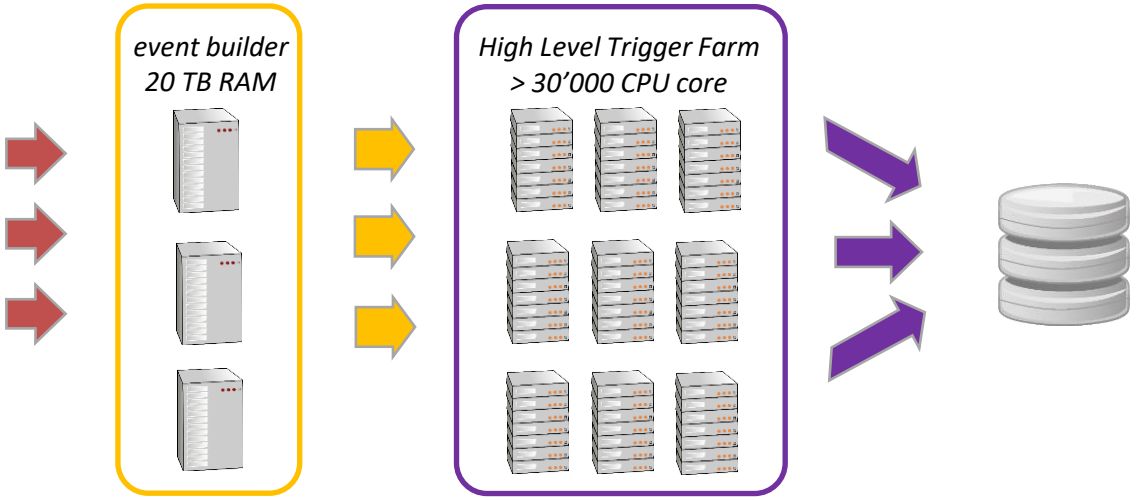
Performance portability with oneAPI: the Patatrack experience

Andrea Bocci[1], **Laura Cappelli**[2], Matti Kortelainen[3], Felice Pantaleo[1]

[1] CERN, [2] University of Bologna, [3] Fermilab

CMS Trigger & DAQ

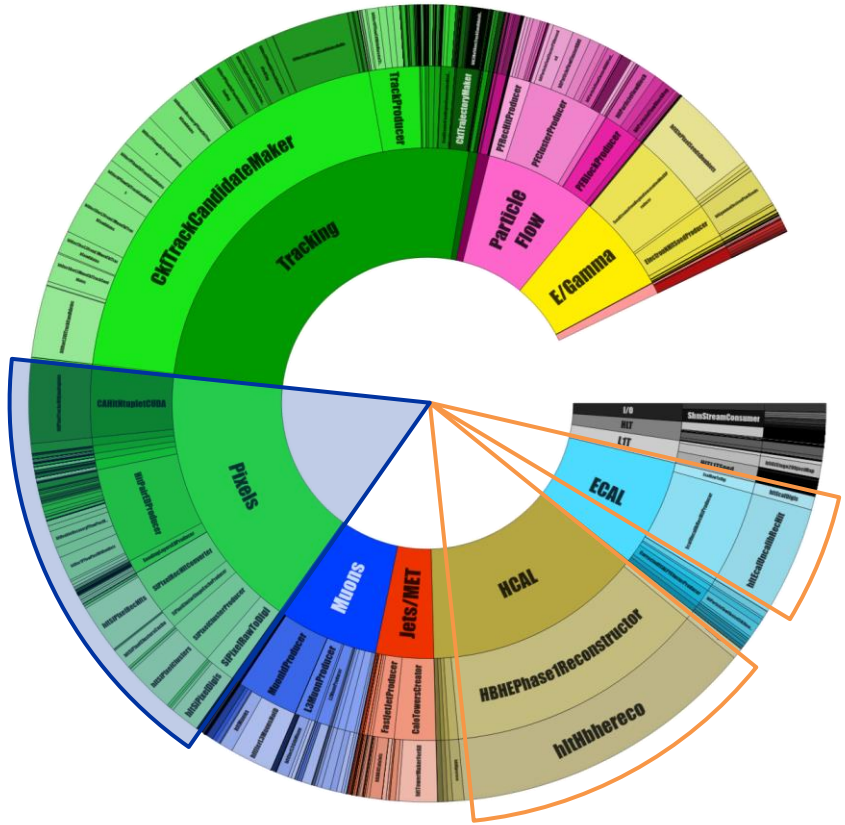




CMS Data acquisition	Level 1 Trigger	High Level Trigger	Storage Manager
Detect particles produced by a high energy proton-proton collision	Hardware based (FPGA)	Software based (runs on Event Builder and High Level Trigger farm)	Distributed filesystem
Many complex layers	Synchronous with LHC	On-demand reconstruction & event selection	Transfer data to Tier 0
40 MHz	***100 kHz***	***1 kHz***	

CMS software

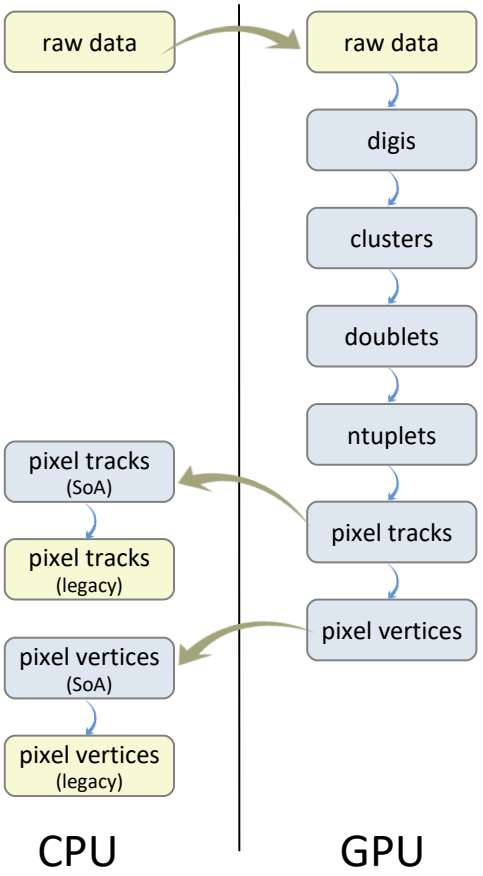
- CMS software (or ***CMSSW***):
  - Modular C++ reconstruction software with over 4000 modules
  - Reconstruct the physics proprieties of the stable particles passing through the detector
  - exploit **Intel TBB** to run multiple modules and reconstruct multiple events concurrently
- Part of CMSSW is under development to take advantage of accelerators
  - The highlighted slices of the pie chart have been rewritten in CUDA for running on GPU
  - A test code has been created with the purpose of testing other technologies (oneAPI, Kokkos, Alpaka) performance portability
    - CMSSW lite software framework + Pixel modules (blue highlighted slice)
    - Standalone Patatrack pixel tracking



Patatrack oneAPI project

- Develop support for Intel oneAPI in the “CMSSW lite” framework
  - Extend the framework to support efficient offloading of modules using oneAPI
- Rewrite algorithms from native C++ and native CUDA to **DPC++** and oneAPI kernels
- Run oneAPI code on:
  - **CPUs**: **host mode** and parallel implementation
  - **GPUs**: Intel hardware with **OpenCL and Level Zero interfaces**
- Compare the performance of the oneAPI “port” to that of the native implementation

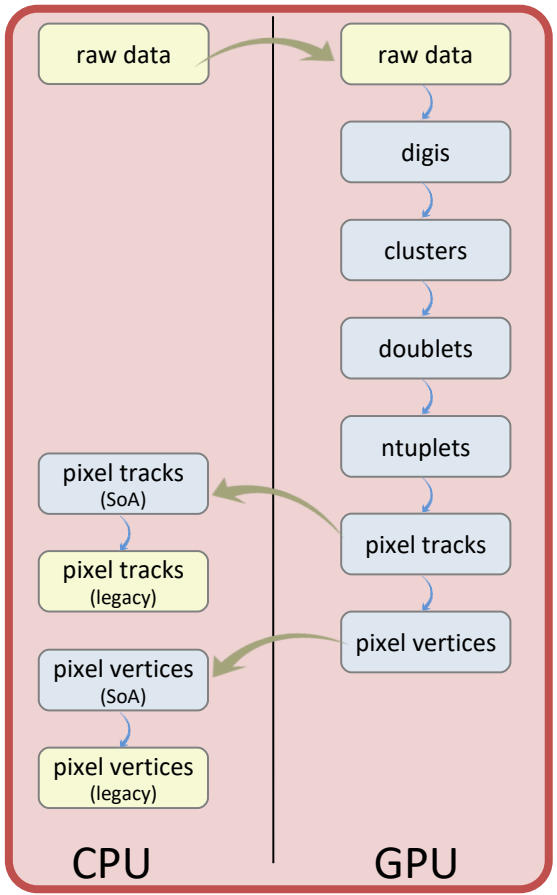
Patatrack reconstruction



- the full workflow
  - copy the raw data to the GPU or to other accelerators
  - run multiple kernels to perform the various steps for the pixel-based tracks and vertices reconstructions
  - copy only the final results back to the host

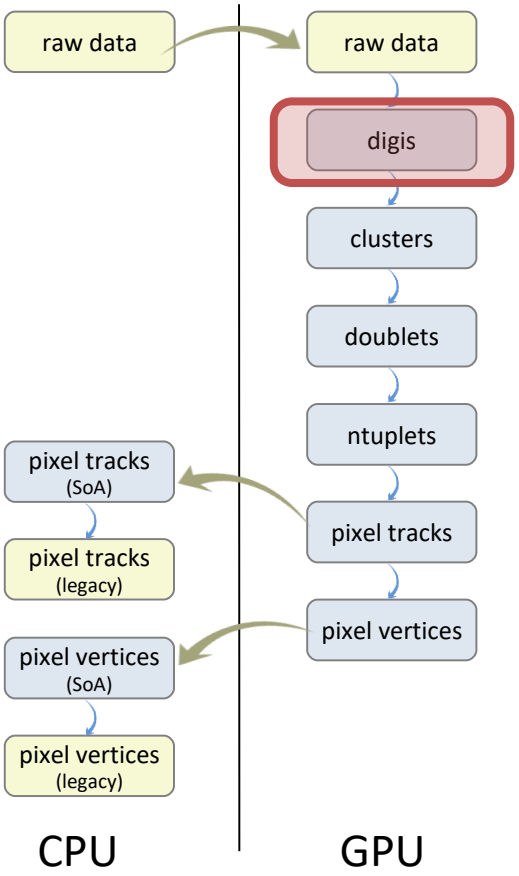
- Work done until now:
  1. Framework conversion
  2. Digis module conversion
  3. Clusters module conversion (… in progress)

Framework conversion



- From native CUDA to oneAPI with the **compatibility tool**
- **successfully converted code**:
  - CUDA stream → in-order queues
  - CUDA memory operations (like malloc, memcpy, …) → oneAPI restricted USM
  - CUDA synchronization operations → oneAPI synchronization operations
- **oneAPI improves the code**:
  - SYCL queues include device information, CUDA streams don't include it
- **Problems with oneAPI**:
  - Events:
    - cannot create not-ready events associated devices
    - wrap events into `std::optional<sycl::event>`
  - Memory allocations:
    - CUDA version uses a caching allocator based on NVIDIA CUB to improve performance
    - not implemented yet for the oneAPI version

Digis conversion



- From native CUDA to oneAPI with the **compatibility tool**
  - The code was almost all converted correctly
- Annoying oneAPI aspects:
  - SYCL streams vs `printf`: require an additional parameter to every function
  - Very verbose kernel syntax calls
- Tests (see next slide for performance measures):
  1. Native CPU algorithm
  2. oneAPI algorithm on CPU device with OpenCL interface
  3. oneAPI algorithm on GPU device with OpenCL interface
  4. oneAPI algorithm on GPU device with Level Zero interface
  5. oneAPI algorithm on SYCL host device
  6. oneAPI algorithm on FPGA emulation device

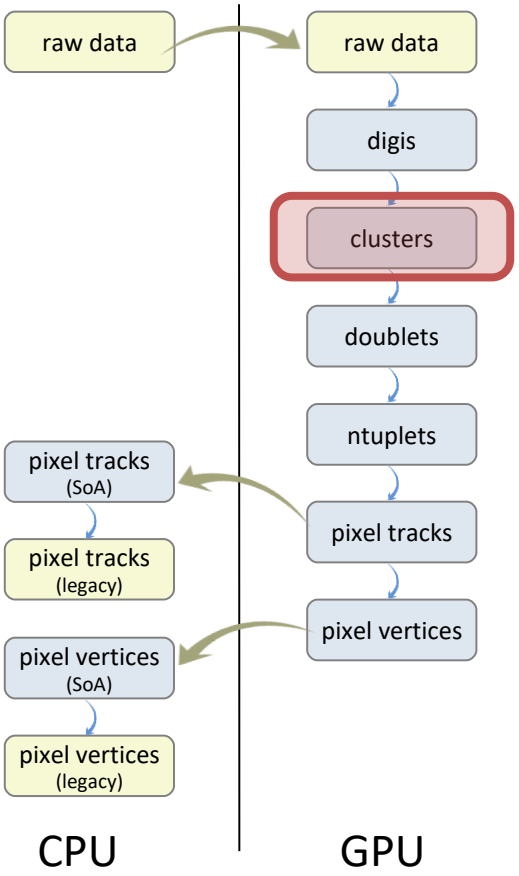
Digis conversion - performance

- performance comparison with different devices and backends, measured on Intel® DevCloud:
  - CPU: Intel(R) Xeon(R) E-2176G CPU @ 3.70GHz, OpenCL driver version 2020.11.10.0.05
  - GPU: Gen9 HD Graphics integrated GPU, OpenCL and Level 0 NEO driver version 20.41.18123

device	**native CPU code**	**CPU**	**GPU**	**GPU**	**SYCL HOST**	**FPGA emulator**
interface	*Single thread*	*OpenCL*	*OpenCL*	*Level Zero*	-	*OpenCL*
Work groups	1	9	24	24	1	9
Work items per group	1	4096	256	256	1	4096
processing time per event	**0.45 ± 0.01 ms**	**17.07 ± 0.03 ms**	**2.67 ± 0.01 ms**	**4.75 ± 0.04 ms**	**3.6 ± 0.2 ms**	**13.0 ± 0.7 ms**

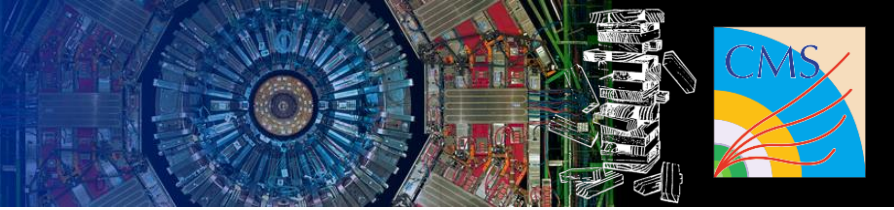
- average processing time per event (lower is better)
  - each value is the average of 10 measurements
  - the work group parameters were not optimised for each device, only adapted to the device capabilities

Clusters conversion



- From native CUDA to oneAPI with the **compatibility tool**
  - The code was almost all converted correctly
- Future developments and encountered problems
  - A **library version** of *prefixScan*, a part of the code, was recently released
    - Compare the performance of these two versions
  - The **subgroup size** must be set on compile time
    - CPU and GPU have different max subgroup sizes
    - It isn't possible to write a parameterized program
    - Try template classes with explicit subgroup sizes
  - The client code blocks and leaves the GPU driver in a frozen state
    - reproduced locally and on **Intel® DevCloud**
    - reported to Intel, working to isolate the cause and find a fix or a workaround

Conclusions



- **About** the aim of Patatrack Portability Project on SYCL and oneAPI:
  - Use the Patatrack code base to investigate different technologies for **performance portability**
    - oneAPI implementation vs native implementation, e.g. on Intel CPUs, or NVIDIA GPUs
    - across different backends and accelerators: CPUs, integrated and discrete GPUs, FPGAs
  - The project is in progress. Next steps will be:
    - main focus: features and debugging
    - more backends: NVIDIA GPUs, Intel discrete GPUs
    - in parallel: evaluate and optimize the performance
- **Intel oneAPI** is a promising technology for portability across CPUs and accelerators
  - the compatibility tool is very useful for porting code from CUDA
  - the tools and ecosystem have significantly improved over the past months
    - DPC++ compiler, plugin mechanism, OpenCL and Level Zero runtimes
    - SYCL 2020 and extensions, oneAPI libraries
- **Our suggestion** for the development of the oneAPI ecosystem: focus on documentation and examples!
  - for example, a SYCL/oneAPI section in https://cppreference.com/ would be invaluable

Questions?

Laura Cappelli

laura.cappelli6@studio.unibo.it

@LauraCappelli8