HPC/QC benchmark: a Happy Marriage or an Contradictory Alliance?

Patrick Carribault *(Research Director & Fellow)*

patrick.carribault@cea.fr

Philippe Deniel *(Fellow)*

philippe.deniel@cea.fr

Outline

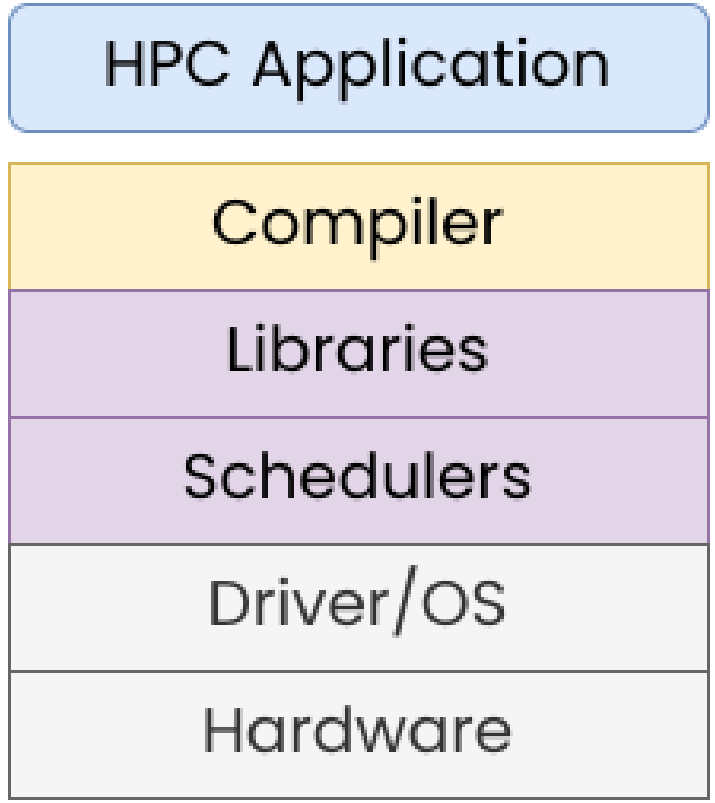
1. Feedback from Classical Benchmark

2. Towards HPC/QC Benchmarks

1. Feedback from Classical Benchmark

Classical Benchmark Purpose

- Benchmarks are designed for a **specific purpose**
  - Usually: some evaluation ➔ **categories**
  - Examples of categories
    - Correctness
    - Verification & Validation
    - *Performance evaluation*
- *Performance*: what is the main goal/output ?
  - **Evaluation of specific components**
    - Application ?
    - Software stack ?
    - Hardware/Architecture ?
- Example HPC benchmark for performance → **Linpack**



Benchmark Example: HPL

- Example of Benchmark: `High-Performance Linpack (HPL)`
  - Linear solver based on linear algebra
  - Evaluate performance of floating-point compute units
    - Relies on performance of DGEMM (Double General Matrix Multiply)
  - Rules:
    - Solve problem in FP64 (64-bit floating-point numbers) or higher
    - Input size and software stack can be chosen
    - Implementation of LU factorization and solver steps can be adapted

- Output ranking ➔ **Top500**
  - Rank machines according to the computational power on regular codes
  - Homepage: http://www.top500.org
  - Updated twice a year (June & November)
  - List machine information (including performance `Rmax` & `Rpeak`)

- Notes
  - Performance in Pflops/s (10^{15} floating-point operations per second)
  - Difference between peak performance (`Rpeak`) and Linpack result (`Rmax`)
  - Power measured in kW

Top500 June 2025 (#1 to #5)

Rank	Country	System	Cores	Rmax (Pflops)	Rpeak (Pflops)	Power (kW)
1	United States	El Capitan	11,039,616	1,742.00	2,746.38	29,581
2	United States	Frontier	9,066,176	1,353.00	2,055.72	24,607
3	United States	Aurora	9,264,128	1,012.00	1,980.01	38,698
4	Germany	JUPITER Booster	4,801,344	793.40	930.00	13,088
5	United States	Eagle	2,073,600	561.20	846.84	

Source: LLNL

- HPL enables ranking machines according to their FP performance
  - Notion of Exaflop/s
- Some information are not mandatory (e.g., Power)
- Evaluation of specific part (floating-point compute capabilities) but on the *whole machine* (including multiple CPUs, devices, compute nodes, network levels, …)
- Same results can be used to derive other ranking
  - ➔ Use the same benchmark (HPL) to enable another ranking based on power efficiency: **Green500**

Green500 June 2025 (#1 to #5)

R	Top500	System	Cores	Rmax	Power (kW)	Gflops/W
1	259	JEDI	19,584	4.50	67	72.733
2	148	ROMEO-2025	47,328	9.86	160	70.912
3	484	Adastra 2	16,128	2.53	35	69.098
4	183	Isambard-AI	34,272	7.42	117	68.835
5	255	Otus	19,440	4.66		68.177

- Ranking made out of Top500
  - Machines listed in this ranking **have to be ranked** in Top500
- Compute the *efficiency of computation* regarding power consumption
  - Main ranking column is Gflops/W
- First machine might not the most "powerful"
  - Rank 259 in Top500

- But HPL evaluates only FP performance

➔ Need different benchmarks to evaluate other capabilities
  - Depends on *evaluation target*
  - Towards *benchmark type*

➔ Can lead to other supercomputer ranking (e.g., Graph500)
  - Graph traversal (output is number of edges traversed per second)
  - Evaluate capability to process indirect memory accesses

Source: GENCI

Types of Benchmarks

Evaluation Target + Category ➔ Different benchmark « sizes » or « types »

Micro-benchmarks

- Very small source code that will evaluated a very specific part of software/hardware
- Execution should be fast
- Examples
  - Memory bandwidth
    - Local: Stream
    - Remote: Nvidia CUDA Bandwidth
  - Parallel performance
    - MPI OSU
    - OpenMP EPCC suite
  - Compiler capabilities
    - Polly bench
  - Architecture performance

Mini apps / Proxy Apps

- Small applications
- Mid-range number of lines of code
- Small to medium input set
- Examples
  - HPL: Linpack for Top500
  - CORAL: Sets of scientific small applications

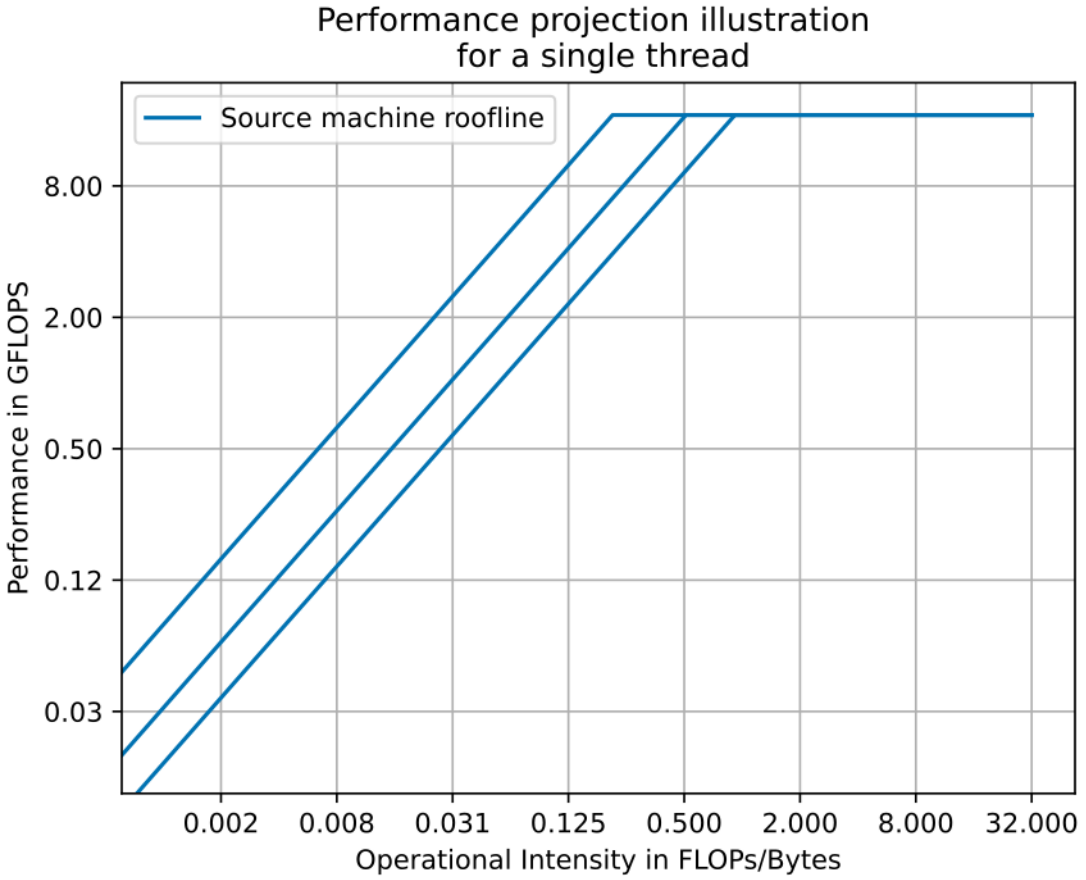
Applications

- Large amount of code
- Larger input set
- Parallel application
- Various architecture/devices

Benchmark Combination

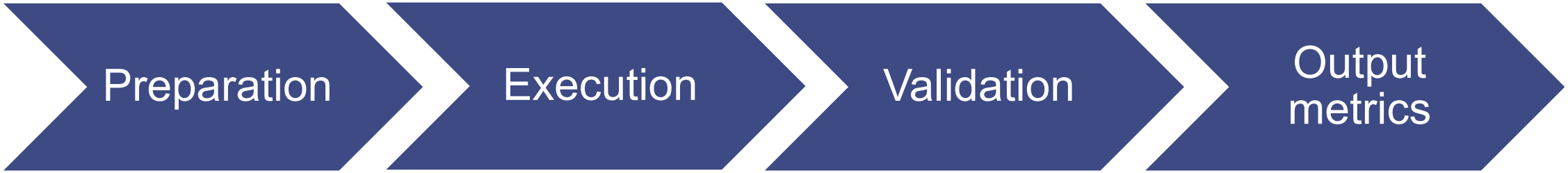
- Multiple benchmarks can be used to enable further profiling/evaluation

- Example with roofline model
  - Goal: graphically place the code performance → see the bottlenecks and room for improvement
  - Roofs are related to benchmarks to evaluate reachable targets
    - Compute roof ➔ HPL
    - Bandwidth roof ➔ Stream
  - Better estimation than with peak performance (compute & bandwidth)

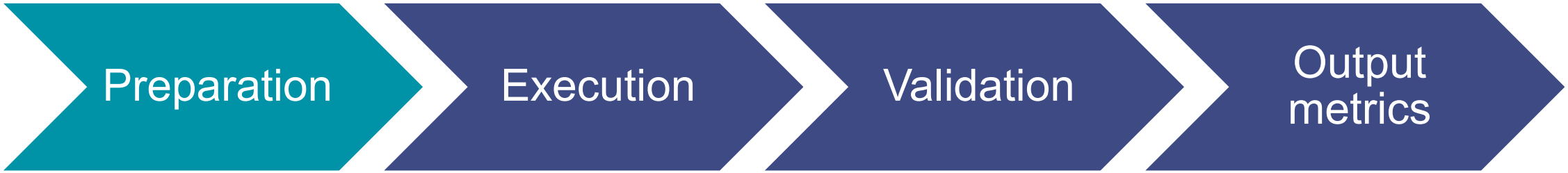


Methodology

- ***Benchmarks = code with environment & rules***
  - Goal: to ensure fairness
  - What can be modified?
    - But what should be kept? (semantics, piece of code, …)
  - What should remain untouched
  - Design of different steps (*preparation*, *execution*, *validation* and *output*)



Steps



- ***Preparation of benchmark***
  - Modifications can be related to various components
  - Source code (evaluation of specific language or more abstract like algorithm?)
  - Software stack components (libraries, compiler)
    - Compiler flags (optimization, behavior w/ computation, …)
    - Specific implementation of standards (e.g., MPI or OpenMP)
    - …

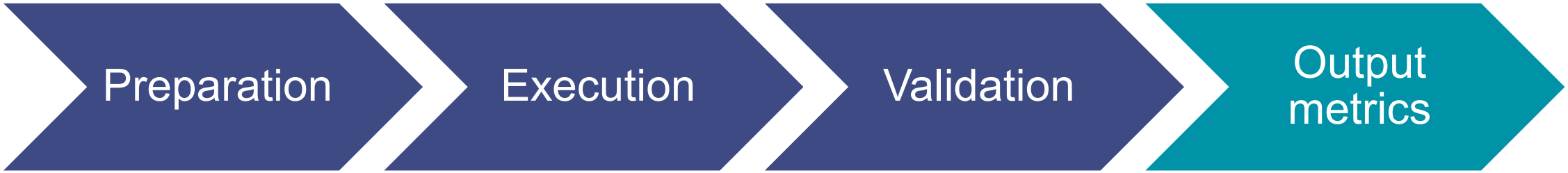
Steps

- ***Execution***
  - What is the execution flow?
  - Machine dedicated to this specific benchmark?
    - Avoid noise due to other applications
    - Need to take care of all shared resources
  - Running from login node?
    - Specific to architecture of supercomputer
    - Evaluation of whole job/resource management
  - Reading input set from disk?
    - Include I/O?

Steps

- ***Validation***
  - Need an « easy » way to validate results
  - Did it run on the actual target architecture?
  - Is the result correct (or at least is it what we expected)?
    - Notion of acceptance

Steps



- ***Measurement***
  - May require multiple runs for stable output
  - Be careful about the environment (e.g., caches in CPU)
  - What about with interfering with measurement?
    - Example w/ scheduling → gathering data may change the execution order

Caveat

Non-exhaustive list of caveats!

- *Rules (e.g., physical vs. logical)*
  - Linpack should execute on 64b precision
  - Requires FP64 hardware units (emulation of FP64 is not allowed yet)
  - ➔ might not be clear in existing rules
- *Timing*
  - Measuring time can be tricky (right granularity)
  - What is the accuracy of the target timer?
  - Example in CPU w/ `gettimeofday` vs `rdtsc`
- *Placement*
  - Be sure about the placement of the code during execution
  - Did it run on the target architecture? (e.g., GPU, no fallback to CPU)
  - What about the pinning of thread/process?
- *Validation*
  - Be sure that the run was « correct »
  - Need an easy way to compare the output (with simple command like `diff` is good if possible)
  - Did you run the whole benchmark?
    - E.g., dead code elimination by compiler!
- *Measurement*
  - May require multiple runs for stable output
  - Be careful about the environment (e.g., caches in CPU)
  - What about with interfering with measurement?
    - Example w/ scheduling → gathering data may change the execution order

2. Towards HPC/QC Benchmarks

What is benchmarking? (More Details)

- According to Cambridge Online Dictionnary (https://dictionary.cambridge.org/)

” the act of measuring the quality of something by comparing it with something else of an accepted standard”

- Benchmarks are associated with Key Performance Indicators (KPI)
  - Performances (~= Time to Solution) is not the one and only KPI
  - Other KPIs may include
    - Energy metrics (Energy to Solution)
    - Durability
    - Resources Consumption (memory, network bandwidth)
    - Latencies
    - Reproducibility of results

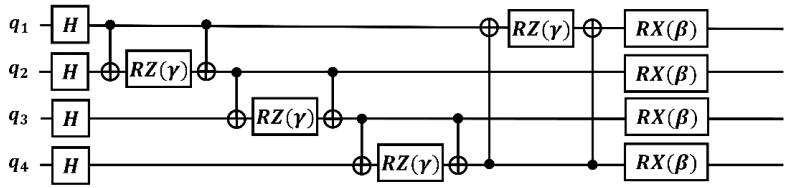
Benchmarking a device

- Today, Quantum Computers are kind of network attached devices
  - They embed a standard machine (like a PC) as a frontend
  - A client/server network protocol operates to submit computations shaped as ”computations requests”
  - Requests are enqueued and processed on the QPU
  - This is pretty similar to a network attached printer

- When benchmarking a HPC/QC platform, what KPIs are relevant ?
  - HPC/QC hybrid jobs include HPC steps and QC steps
    - The benchmarks require internal metrics to evaluate two different kinds of resources consumption at HPC and QC levels
  - HPC/QC hybrid jobs require submissions of jobs to QPU
    - What is the cost of this communication?

Knowing the environment is critical

- Most of the benchmark will be high level application
  - Variationnal algorithms : QAOA, VQE, ...
  - Classical NP problems : QUBO, MIS, MaxCut, ...



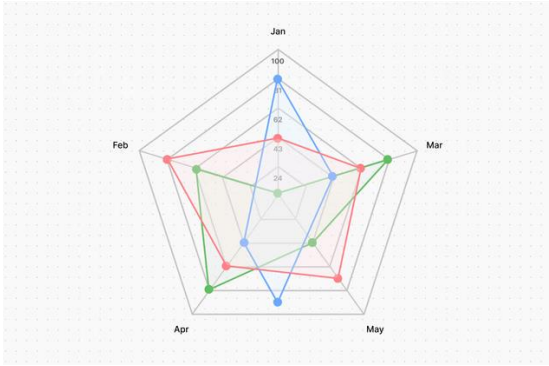
- Before performing the benchmark, other resources are to be benchmarked
  - Running the benchmark with an ”empty” QC step, with no use of the QPU
    - This is critical to evaluate the ”HPC noise” within the HPC/QC computation
    - For example, evaluating the cost of running the optimizer in QAOA or VQE
  - All of the HPC resources are to be monitored
    - memory bandwidth, network bandwidth, CPU, ...
    - HPC has already many probes to do this

- **Once** you know all HPC KPIs in the HPC/QC platform, **then** you can run the HPC/QC benchmark
  - Each biaises is a source of noise and misled becnhmark
  - Many biaises exist, each has to be identified and considered

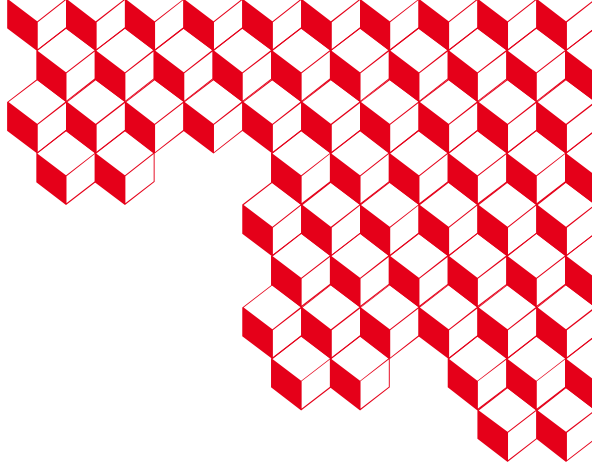
One to rule them all?

- For many, HPC supercomputers are ranked in Top500 via LINPACK
  - LINPACK has structured the design and features of HPC components
  - Getting the highest mark on LINPACK became crucial sometimes on the detriment other KPIs
  - LINPACK is clearly unavoidable in HPC
  - LINPACK is a ”one to run them all” HPC benchmark

- It’s logical to think about such a benchmark in HPC/QC
  - Too many different technologies with few common aspects
  - Performances for a given application may differ with different use cases
  - A ”one to rule them all” performance is unthinkable for the moment



- HPC/QC platform have to combine different aspects (and biaises)
  - All bechmark should be a combination of several smaller benchmarks
  - The result could be exposed as a radar diagram (or Kiviat diagram)



Merci