FONDAZIONE links
PASSION FOR INNOVATION

Demistifying HPC-QC

Programming model, resource allocation and scheduling for HPC integration

PAOLO VIVIANI – CPE RESEARCH DOMAIN – LINKS FOUNDATION

HPQCI – PISA, JUNE 3RD 2024

Who is this guy?

I feel you: you can blame the chairs for inviting me

- Msc in theoretical physics, University of Torino
- PhD in computer science, University of Torino
  - Data parallel training of Deep Neural Networks (before it was cool)
- A couple of years in CAE industry doing numerical/ML stuff before getting bored
- Now I'm back to research at **Advanced Computing, Photonics and Electromagnetics research domain** of **LINKS Foundation**, in Torino

LINKS Foundation

Advanced Computing, Photonics and Electromagnetics research domain

- Private, not-for-profit research centre in Torino, Italy
- **advanced Computing, Photonics and Electromagnetics (CPE) research domain**
  - **~40 researchers** (including **12** PhD students) + **5** thesis students (2023)
- Quantum activities focusing on
  - Algorithms and applications
  - Neutral atoms
  - **HPC-QC integration**

Quantum computing team

Olivier Terzo
PhD, Head of CPE domain

Giacomo Vitali
Particle physicist
Technology specialist / team leader

Alberto Scionti
PhD, Computer engineer

Chiara Vercellino
Mathematical engineer

Paolo Viviani
PhD, Theoretical Physicist

Current collaborations: Politecnico di Torino, CINECA, PASQAL, QuEra Computing Inc.

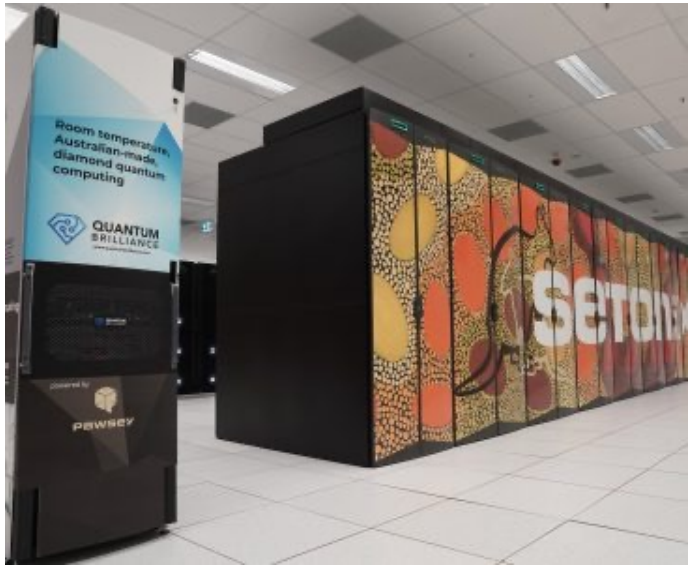
Upcoming IQM

HPC-QC integration

What it looks like?

Since we are here discussing… probably we don't know yet

- I've witnessed countless discussions about compilers, software stacks, open source vs. vendor software etc.
- The truth is that we don't know what it looks like **today**
  - We have examples of QC put inside an HPC data hall, but real integration… not so much
- Maybe a historical perspective could help
  - Where do we come from?
  - Where are we going?



Where do we come come from?

Our HPC background

HPC without QC

The (recent) past

- HPC jobs still dominated by **MPI+X** workloads
  - At least if we forget about LLMs, but they are not so different
- **Accelerators are commonplace** (unless you run Fugaku and your HPCG numbers mop the floor with western HPC)
  - As it is the **offloading** programming model
- Still **lots of Fortran** around
- What can we learn from this about how HPC-QC will look like?

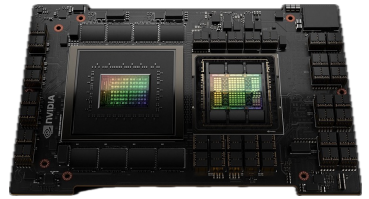
Where are we going?

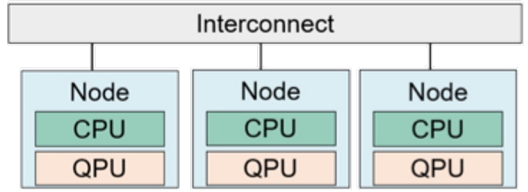
Our HPC-QC future

Back to the future

An (optimistic) glipse from 2040

- Oups… we slipped in **2040**
- I will still be 20 years away from retirement (if any)
- We have flying DeLoreans, and now we have also
  - **FTQC**
  - **at room temperature**
  - **with standard lithography**





- We have **QPUs embdeed in each computational node** as PCIe Gen XX devices
- Have we seen it before? Any idea of what kind of programming model/paradigm is everyone using?
- **Offloading** anyone?
  - Maybe (still…) some flavour of **MPI+X**? Lots of Fortran still around?

Where are we now?

Our NISQ present

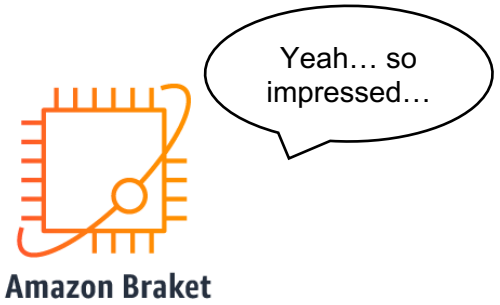
Sadly back to 2024

- Todays' quantum computers are basically big **standalone** experimental racks
- **Heterogenous technologies** with different features (discussed widely yesterday)
- Also **different computational models**
  - Digital
  - Analog (e.g., current neutral atoms, annealers)
- All this affect the software stack, the programming model and the HPC-QC integration approach

Sadly back to 2024

What are we doing right now?

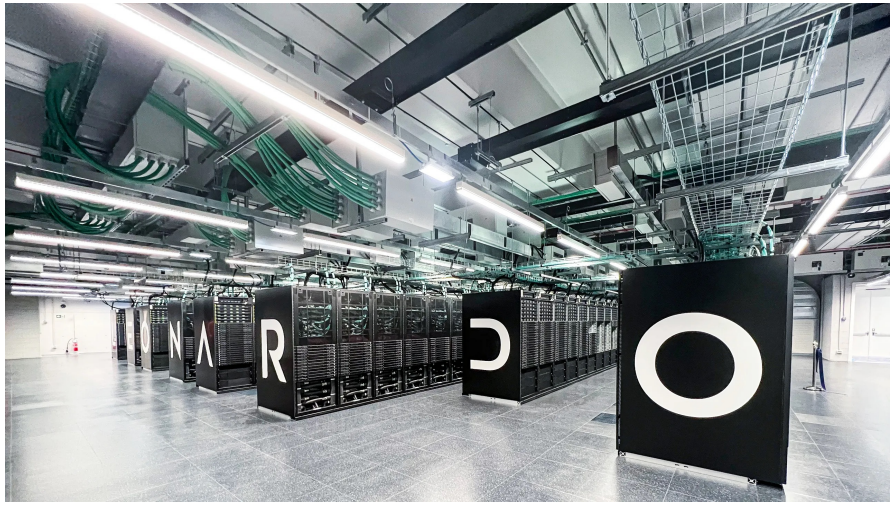
- Ok, I just installed a quantum machine next to a supercomputer, what do I do?
  - I install qiskit/pulser/whatever as a module
  - I define a **SLURM quantum partition** with one node configured to connect to the machine
  - I define a **quantum GRES**
  - Wow, I can run python scripts from an HPC node!



- An HPC node running **single threaded python code** randomly calling
  - a quantum backend locally
  - a local (HPC) emulator
- is the lowest hanging fruit of HPC-QC integration!

How to do real HPC with QC?

What if we want to join real HPC (i.e., large MPI jobs) and QC?



160k cores + 14k GPUs

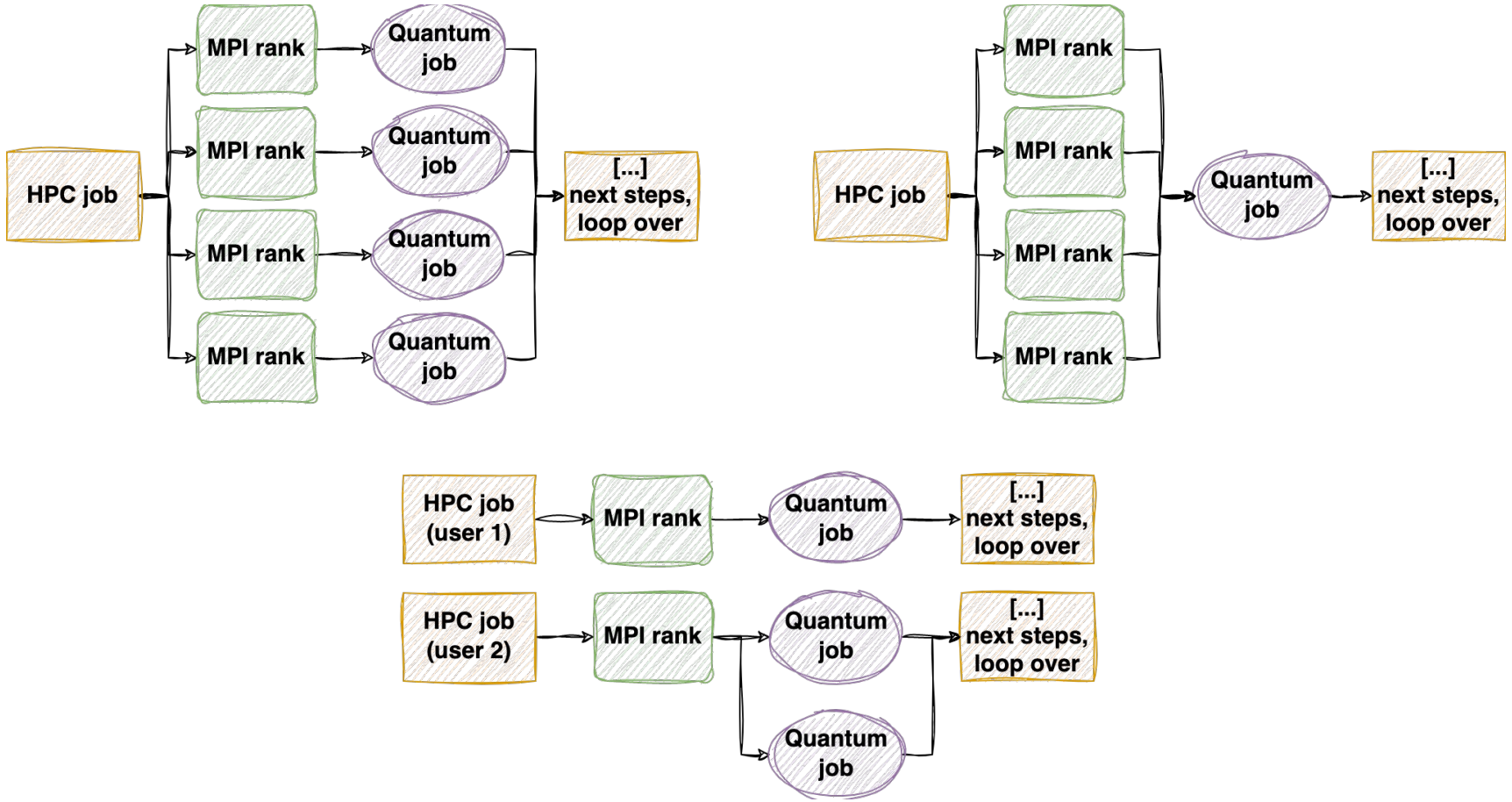
400 gbps IB ?

1 QPU

HPC-QC challenges

The future is clear, the present less so

- We need to understand how to make **multiple users** running **large MPI jobs** live happily together with **current NISQ devices** and beyond, as far as **QPUs are a scarce resource**
- Can we create a taxonomy of these integration scenarios?

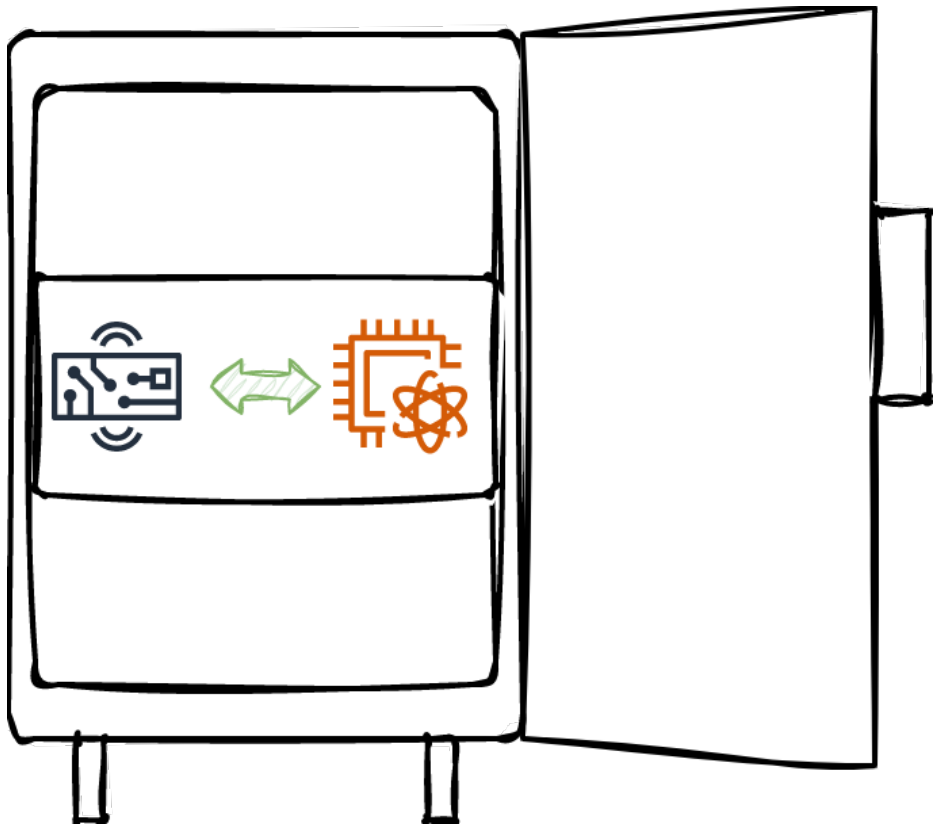


Quantum-classical workloads taxonomy

Let’s try to look at the literature, for instance Humble et al. provide a nice overview of **3 categories**

Type 1 – In-sequence processing

VERY fast feedback loop interaction



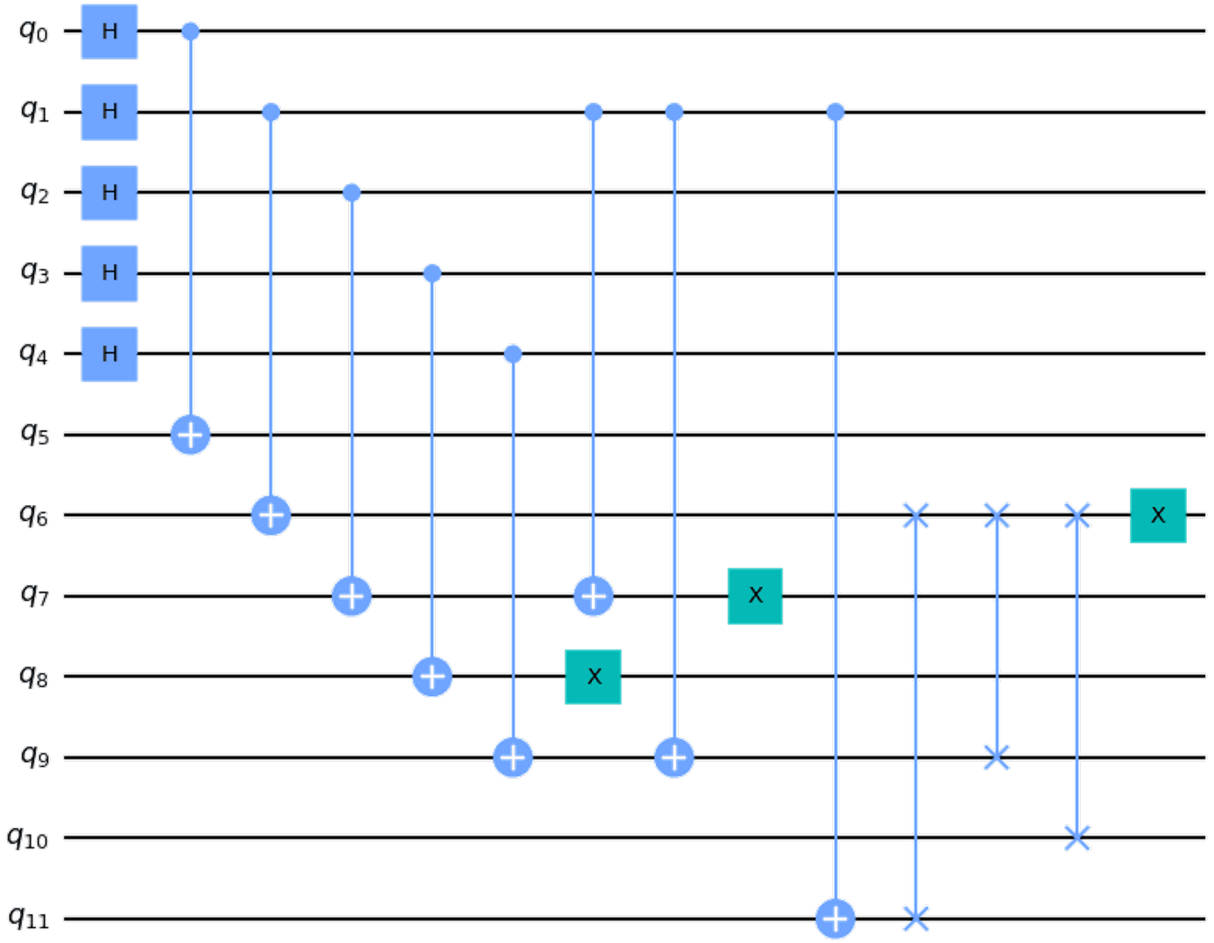
- This is about having classical computing interacting with quantum circuits **within their coherence time**
- Typically achieved by putting a QPU and an FPGA in a freezer together, **someone refers to this as tightly-coupled mode**
- Used mostly for error correction (i.e., syndrome calculation)

- Is this HPC-QC?
- It's certainly high performance (we need feedback loops in the order of nanosecond), but…
- **it's not HPC** in the sense we are discussing here
- It could become very interesting if we allow, e.g., individual MPI ranks to interact with QPUs mid circuit, but that's a 2040 problem

Type 2 – Single-circuit applications

- This refers to most quantum computing applications, including
  - Variational algorithms
  - Multiple shot sampling
  - Analogue algorithms

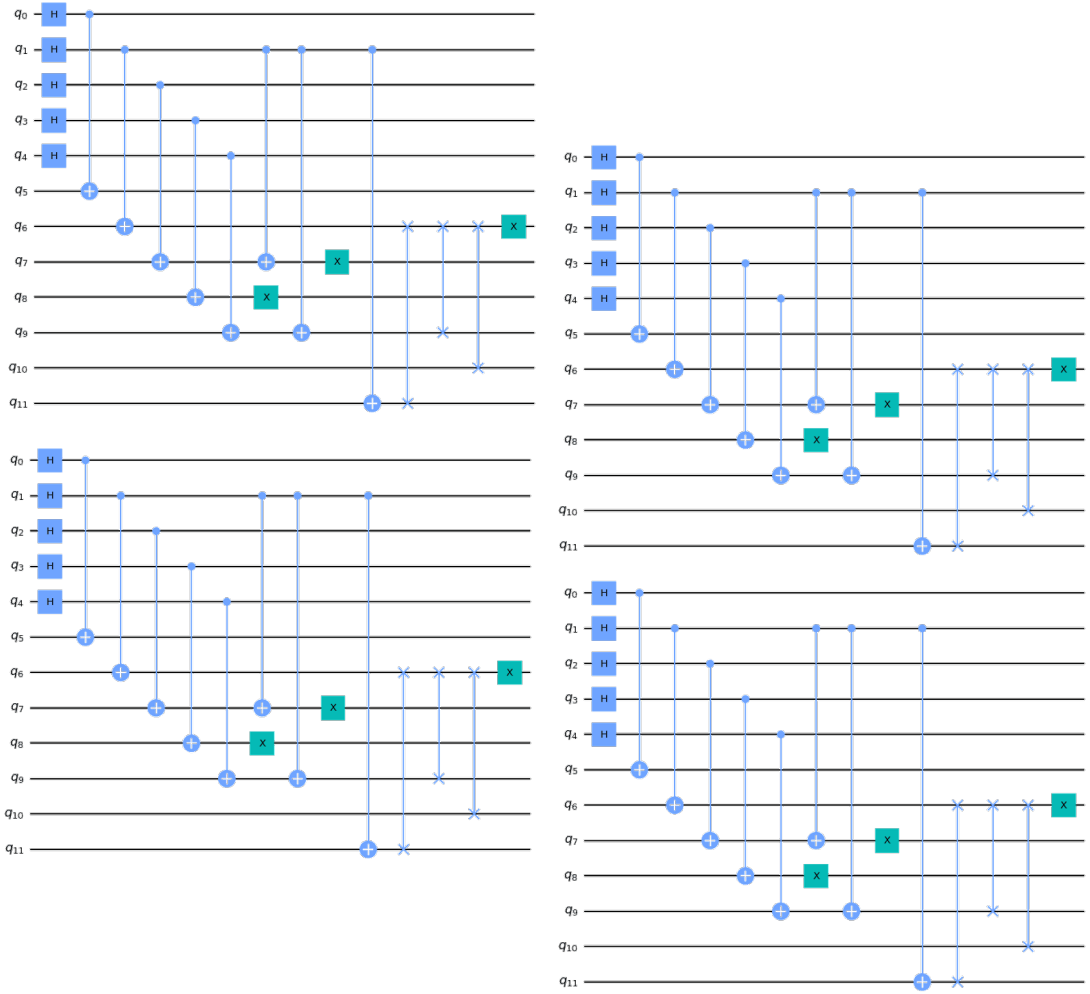
- Is this HPC-QC?
- It can be, but…
- The category is too broad
- **It doesn't say anything about the classical workload and how we should allocate quantum resources**



Type 3 – Ensemble-circuit applications

- This refers to cases where we are averaging the results of different circuits, and possibly also of emulators

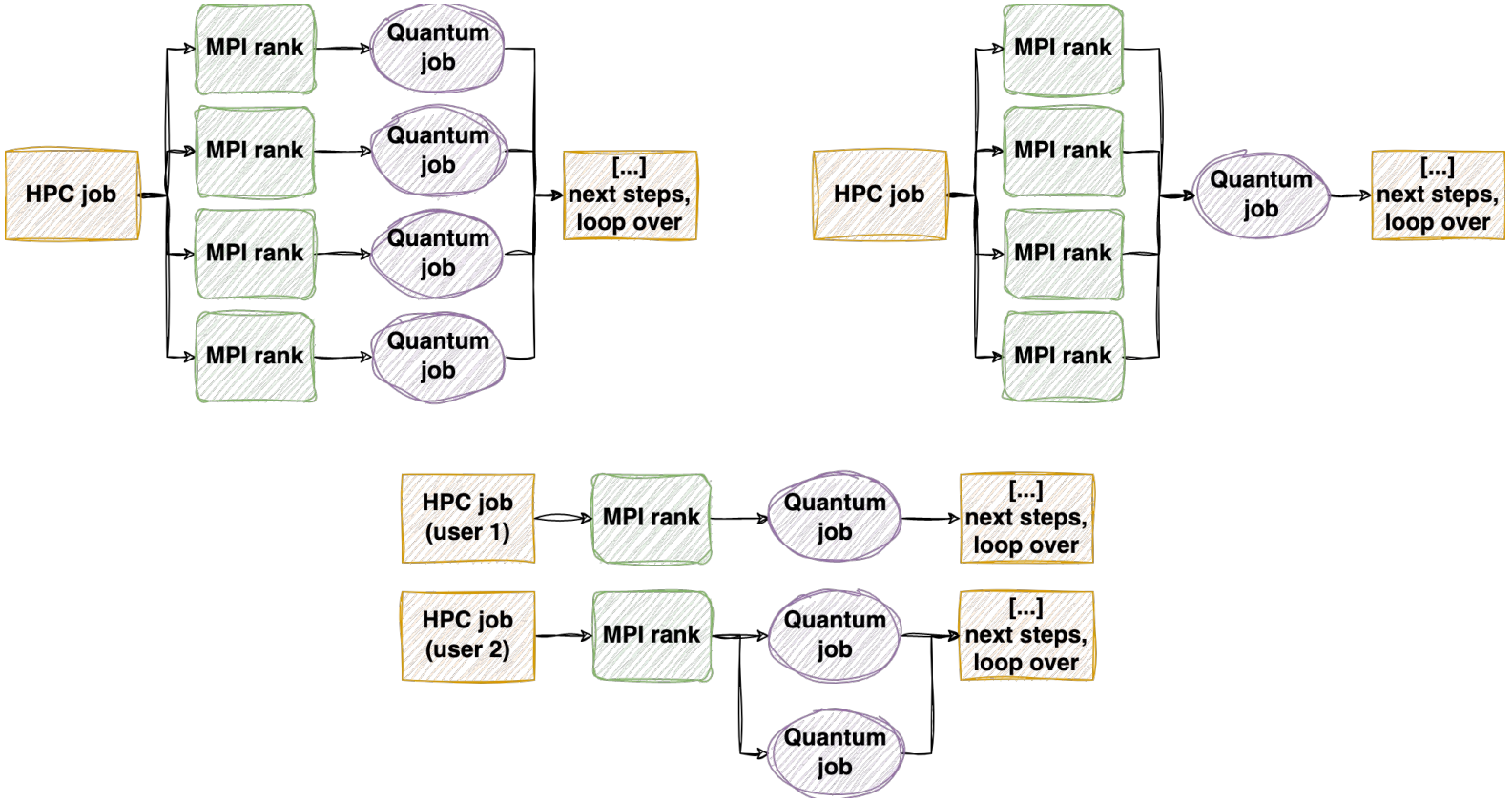
- Is this HPC-QC?
- Lends itself to parallel execution
- Again: **it doesn't say anything about quantum resource allocation and classical HPC part**



Ok, then?

Workload characterization

- Single or ensemble-circuit applications characterize the quantum workload, but don't say anything about the nature of the classical workload. **The workload taxonomy question is still open.**
- We should create a taxonomy that **characterizes both quantum and HPC workloads together** (e.g., variational algorithms)



Beyond workload characterization

It’s all about scheduling!

HPC-QC integration in a QPU scarcity era

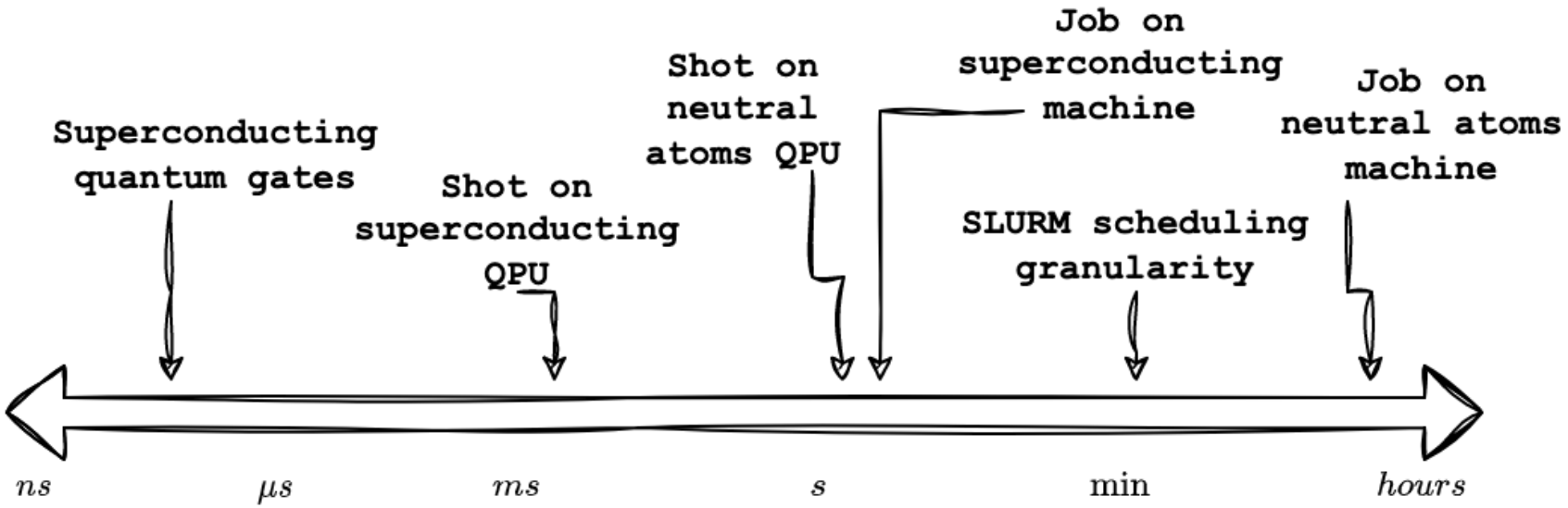
Fairly Noisy, Not-so-Intermediate Scale anymore, Scarce and very expensive Quantum

- If we want to couple large MPI jobs with quantum algorithms we can identify two issues:
  1. Software stack (e.g., Python/Julia vs . C++/Fortan) and compilation (where to compile? **Can we consider quantum compilation as an HPC job *per se***)?
     > Significant prior work on this
  2. **Quantum resource scheduling** policies and their relationship with the programming model
     > No significant literature

- Let's focus on no. 2!
- If we want to integrate anything into an HPC system, we need to deal with

A granularity problem

Time scales for reference



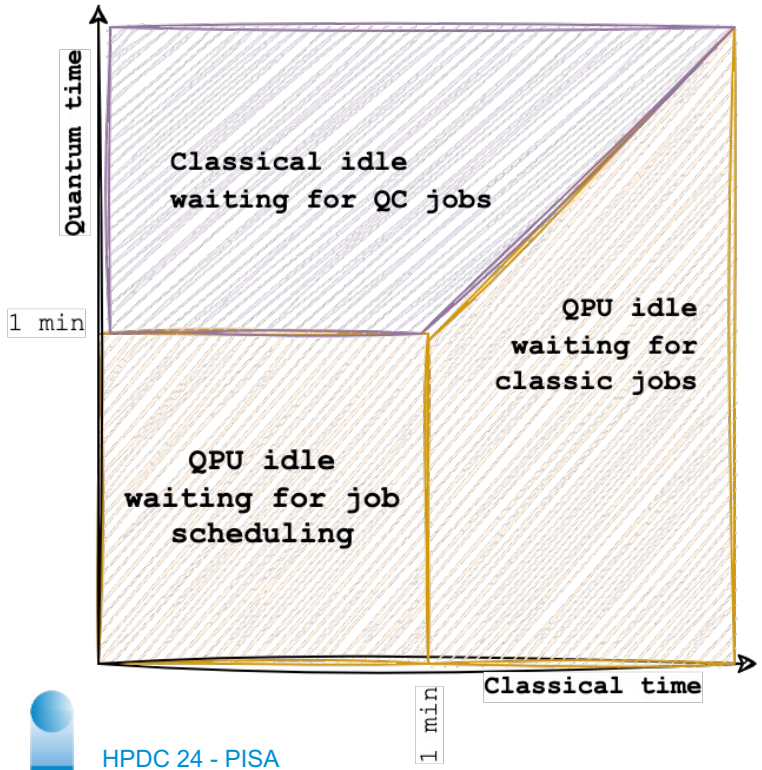
- 10-100 μs superconducting quantum gates
- 1-10 ms shot on superconducting HW
- 0.1-1 s shot on neutral atoms machine
- **1-10 s** **job on superconducting HW** (~1000 shots)
- ~1 min typical SLURM scheduling granularity
- **30-60 min** **job on neutral atoms machine** (custom register preparation + 1000 shots)

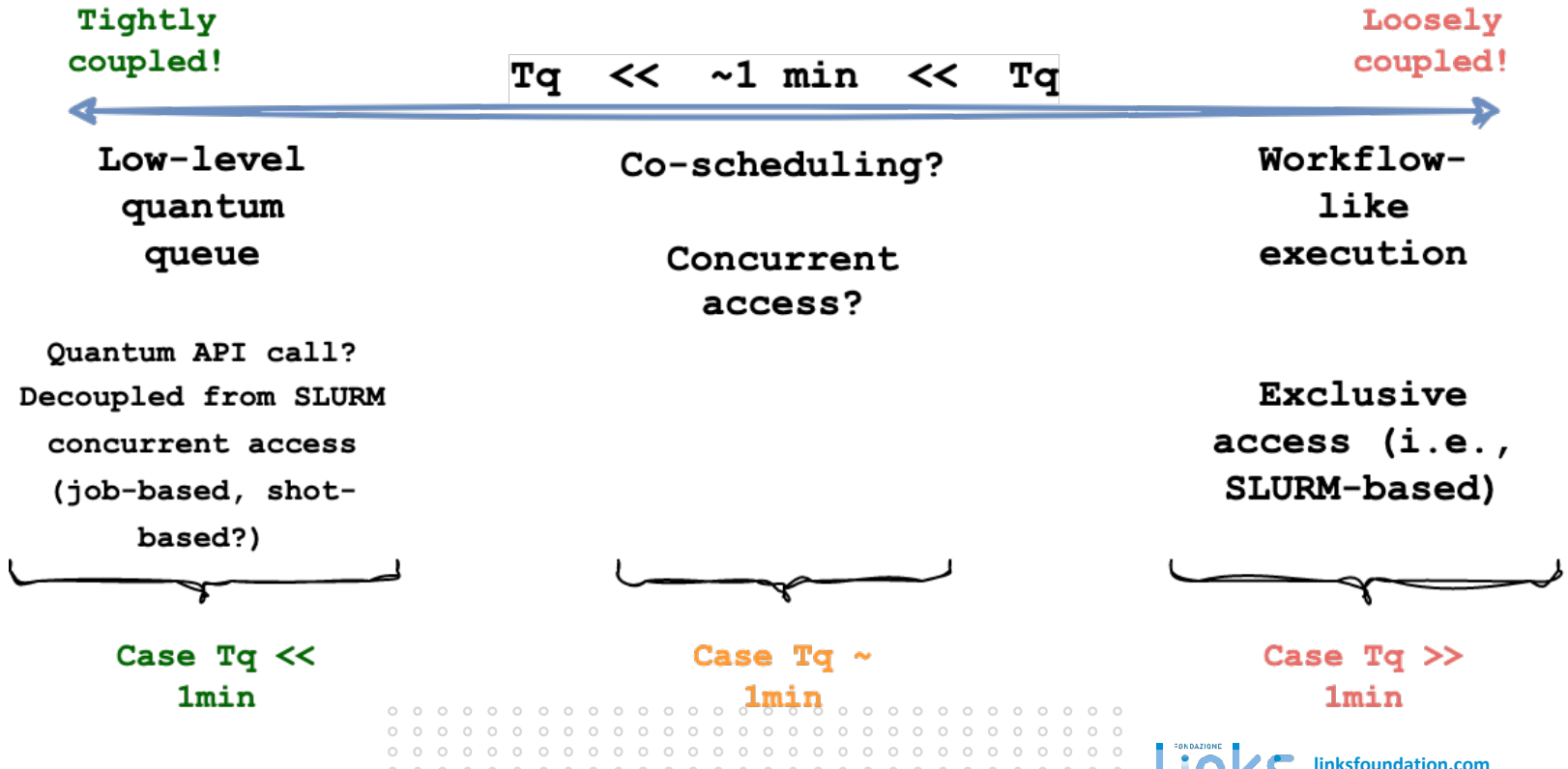
Scheduling scenarios

Tq is the typical duration of a quantum job

Two extreme scenarios that **disallow (exclusive) co-scheduling**

- **Tq << 1 min** → cannot block QPU for whole classical job duration, low level access queue needed
- **Tq >> 1 min** → cannot block large jobs waiting for the QPU to answer, loosely coupled approach needed

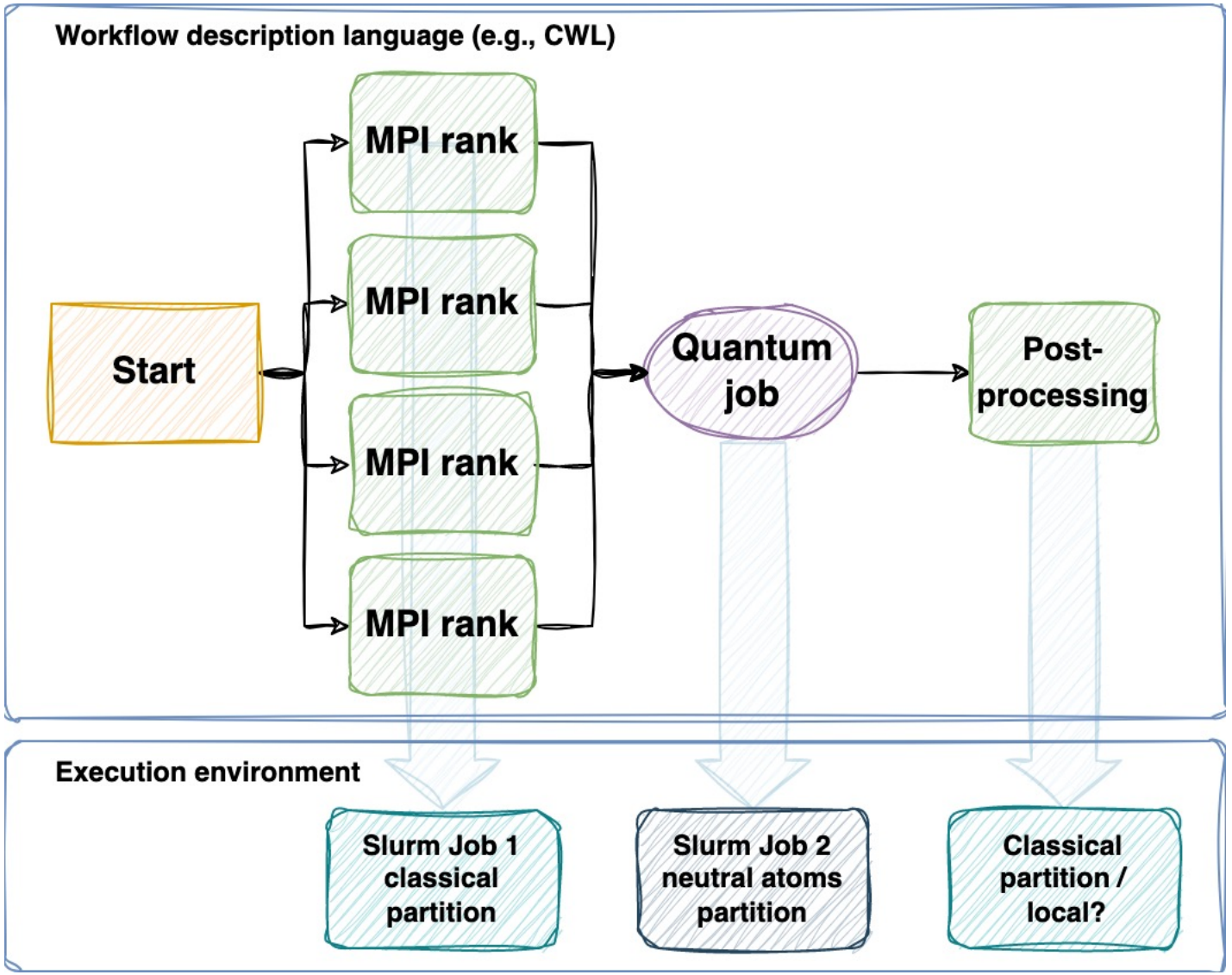




Neutral atoms case

Tq >> 1 min

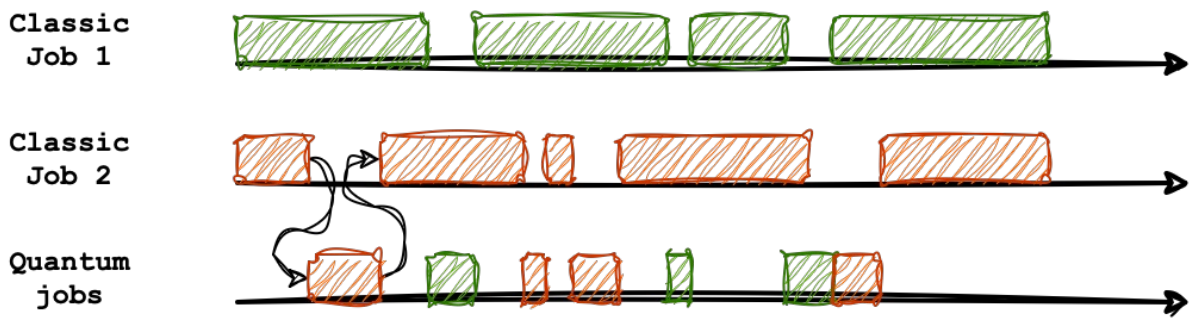
- Co-scheduling at SLURM level is harmful: classical nodes will wait >30min for the QPU to complete its job
- We advocate for a **loosely coupled, workflow-like approach**, where each step is scheduled independently thanks to
  - SLURM dependencies (maybe too painful?)
  - Workflow management systems (e.g., StreamFlow, Airflow, Nextflow, pyCompSS)

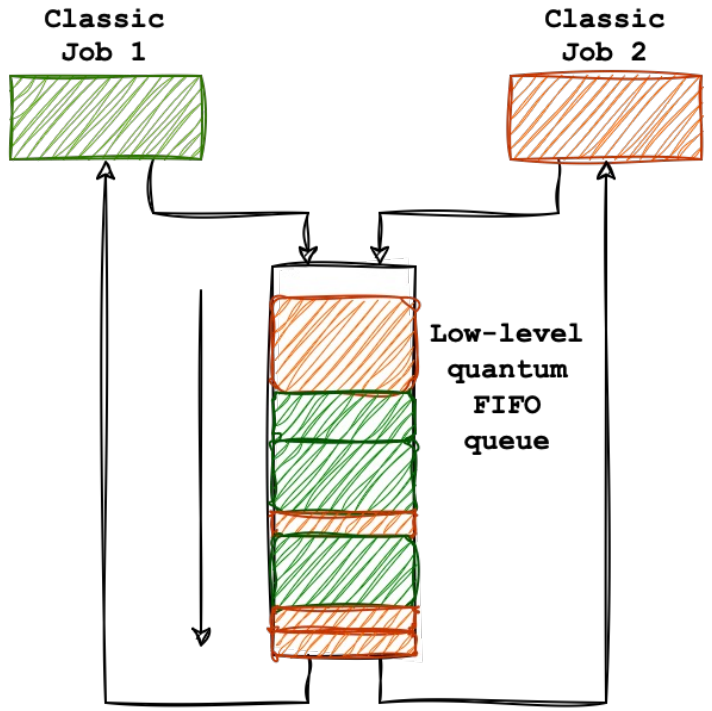


Superconducting-like case

Quantum jobs are short enough to allow

- Blocking/non-blocking calling from classical job
- **Concurrent access** to the quantum machine
- A different, low-level queue w.r.t. to the SLURM queue (i.e., specific API called from application logic)





Superconducting-like case

Seems trivial, but there are potential pitfalls

- Unlimited access **to the low-level queue can still cause inefficient allocation**
  - i.e., 100 jobs on a superconducting machine take in the order of 1000 seconds to run
  - If slurm has no visibility on the low-level quantum queue (i.e., if no quantum gres is defined), it could allocate classical resources using the QPU even when the QPU is busy for “long” time
  - This could lead again to stalling classical nodes
- A quantum gres should be defined anyway to provide access to the low-level queue
- The amount of gres should be either calibrated in order to avoid starting too many classical jobs
- or even better, a SLURM plugin should keep track of the queue status and provide access to the gres accordingly (maybe defining a
- gres_time variable with the equivalent of expected max_wall_time for the QPU)
- In 2040 this will be managed by node-local queues like CUDA streams or, for multi-tenancy, CUDA MPS

Conclusion

Some remarks

From captain obvious

- HPC-QC integration is not just compilers and software stack, but also
  - Programming model
  - Resource allocation and scheduling
  - User access model
- Loosely coupled approach could be an effective approach for current neutral atoms QPUs
  - Doesn’t waste classical resources
  - Can leverage existing software tools
  - Can also implement variational algorithms (slowly but surely!)
- Superconducting qubits allow for tighter round trips between the CPUs and the QPU
  - Some care in integrating the necessary low-level queue with the batch scheduler
- All the ongoing discussions about the software stack and the vendor vs. tools/libraries developers responsibilities are still relevant in this picture!

Thanks for your attention!

Questions? and some references….

paolo.viviani@linksfoundation.com

- M. Ruefenacht *et al.*, “Bringing quantum acceleration to supercomputers.”
- T. S. Humble, A. McCaskey, D. I. Lyakh, M. Gowrishankar, A. Frisch, and T. Monz, “Quantum Computers for High-Performance Computing,” *IEEE Micro*, vol. 41, no. 5, pp. 15–23, Sep. 2021, doi: 10.1109/MM.2021.3099140.
- A. Gazda and O. Koska, “A pragma based C++ framework for hybrid quantum/classical computation.” arXiv, Jan. 08, 2024. Accessed: Jan. 10, 2024. [Online]. Available: http://arxiv.org/abs/2309.02605
- J. Wurtz *et al.*, “Aquila: QuEra’s 256-qubit neutral-atom quantum computer.” arXiv, Jun. 20, 2023. Accessed: Jul. 12, 2023. [Online]. Available: http://arxiv.org/abs/2306.11727
- M. Schulz, M. Ruefenacht, D. Kranzlmuller, and L. B. Schulz, “Accelerating HPC With Quantum Computing: It Is a Software Challenge Too,” *Comput. Sci. Eng.*, vol. 24, no. 4, pp. 60–64, Jul. 2022, doi: 10.1109/MCSE.2022.3221845.
- G. S. Ravi, K. N. Smith, P. Murali, and F. T. Chong, “Adaptive job and resource management for the growing quantum cloud.” arXiv, Mar. 24, 2022. Accessed: May 04, 2024. [Online]. Available: http://arxiv.org/abs/2203.13260
- Bartsch, Valeria *et al.*, “< QC | HPC >: Quantum for HPC,” Zenodo, Oct. 2021. doi: 10.5281/ZENODO.5555960.
- A. J. McCaskey, D. I. Lyakh, E. F. Dumitrescu, S. S. Powers, and T. S. Humble, “XACC: a system-level software infrastructure for heterogeneous quantum–classical computing,” *Quantum Sci. Technol.*, vol. 5, no. 2, p. 024002, Feb. 2020, doi: 10.1088/2058-9565/ab6bf6.
- K. A. Britt and T. S. Humble, “High-Performance Computing with Quantum Processing Units,” *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, pp. 1–13, Jul. 2017, doi: 10.1145/3007651.