lrz

Leibniz Supercomputing Centre

of the Bavarian Academy of Sciences and Humanities

Simulating quantum algorithms on HPC systems: a performance perspective

IXPUG 2020 Annual Meeting | 12.10.2020 | *Luigi Iapichino (LRZ),* Fabio Baruffa (Intel)

About the presenter

Dr. Luigi Iapichino

Application Specialist, Quantum Computing and Astrophysics

High Performance Systems Division, LRZ

Luigi.Iapichino@lrz.de

Lead of Quantum Computing at LRZ

Co-founder of the Bavarian Quantum Computing eXchange (BQCX)

This work has been made possible by the LRZ/Intel joint project “Highly scalable Quantum Computing simulations on leadership-class HPC systems,” funded by Intel.

The study presented here has been developed together with Fabio Baruffa (Intel).

Structure of this talk

- Why **quantum computing simulations?**
- The Intel **Quantum** Simulator
- A few notions on QC and performance considerations
- Profiling and first performance results
- Summary and outlook

Why a **QC simulator?**

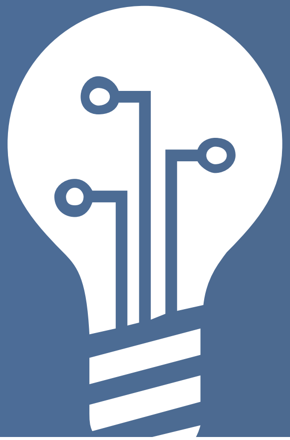
How to work on QC without available hardware?

Simulation is an invaluable tool for modelling classical computer systems

Simulating a quantum algorithm to better understand gates behaviour:

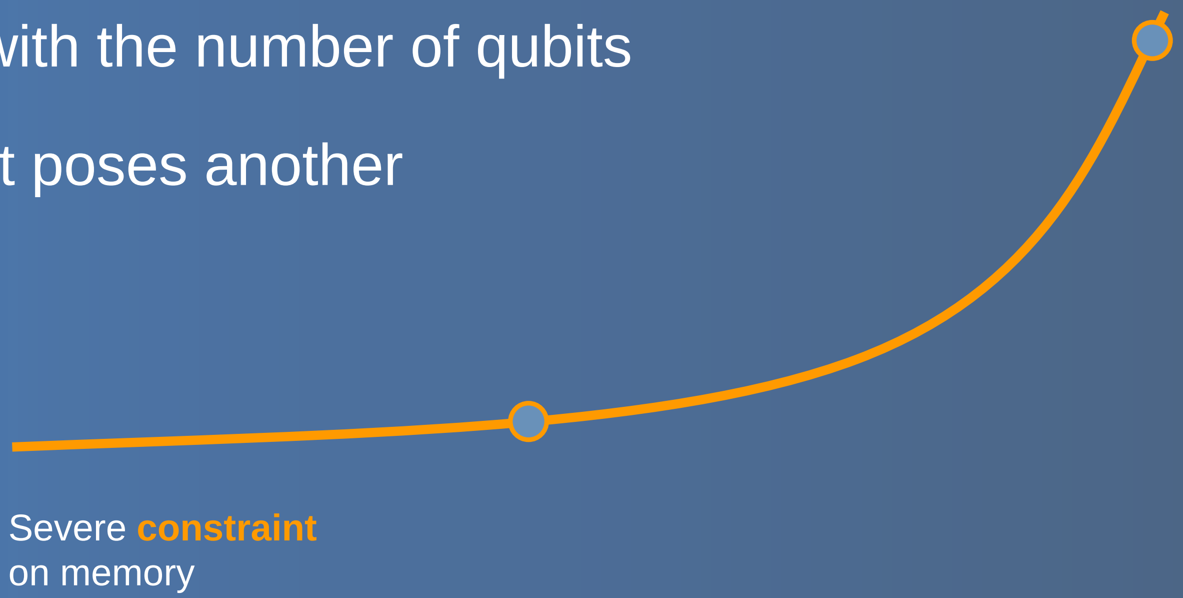
- evaluating **complexity;**
- **study circuits** that cannot be modelled analytically;
- investigate performance also in the **presence of noise.**

The access to a simulator can ease the learning process and provide an **entry point to QC.**



QC simulations on **HPC systems**

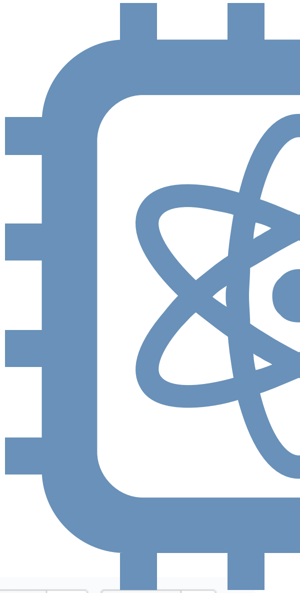
- Simulating quantum circuits on classical computers is **very demanding**
- The size of state **grows exponentially** with the number of qubits
- The number of gates in a quantum circuit poses another constraint on the **simulation time.**



Solution: **running on HPC resources.**

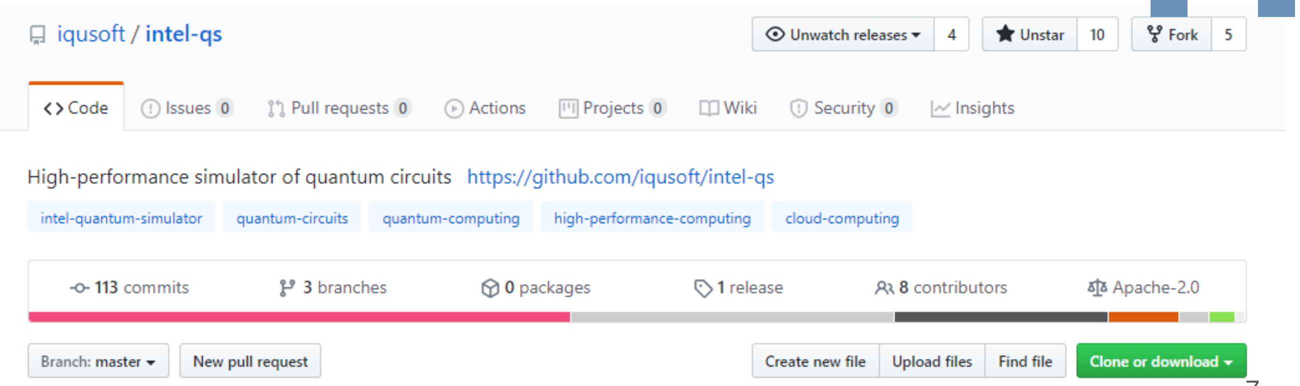
The Intel® Quantum Simulator **(IQS)**

- **IQS** (formerly known as *qHiPSTER*) is an open-source **simulator of quantum circuits** on HPC systems
- Currently developed by **J. Hogaboam, G.G.Guerreschi and F. Baruffa (Intel)**
- It simulates **general single-qubit gates** and **two-qubit controlled gates**
- Written in **C++**
- Parallelized with **MPI** and **OpenMP**



Resources:

- Paper: *https://arxiv.org/abs/2001.10554*
- Github: *https://github.com/iqusoft/intel-qs*



Representation of **single-qubit gates** in IQS

We will limit us for brevity to **single-qubit gates** (controlled two-qubit gates are similar)

Vector representation: a quantum state of a system with *n* qubits is represented as a complex vector of 2^n components.

$$|\Phi\rangle = \alpha_{00...0}\,|00...0\rangle + \alpha_{00...1}\,|00...1\rangle + ...\alpha_{11...1}\,|11...1\rangle$$

One can consider using more computing nodes, each storing part of the state.

MPI Communication between nodes is required to simulate certain quantum gates.

A single-qubit gate acting on qubit *k* can be represented as **unitary transformation:**

$$U = I \otimes I \otimes ... \otimes Q \otimes ... \otimes I \otimes I$$

where **Q** is a unitary matrix:

$$Q = \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix}$$

Simple case: **system of two qubits**

In case of a **2-qubit system,**

$$|\Phi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

Also as **vector of amplitudes:**

$$|\psi\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}$$

How to write a **gate operation for the amplitudes?**

In general:

$$\alpha'_{*\ldots*0_k*\ldots*} = q_{11} \cdot \alpha_{*\ldots*0_k*\ldots*} + q_{12} \cdot \alpha_{*\ldots*1_k*\ldots*}$$
$$\alpha'_{*\ldots*1_k*\ldots*} = q_{21} \cdot \alpha_{*\ldots*0_k*\ldots*} + q_{22} \cdot \alpha_{*\ldots*1_k*\ldots*}$$

In case of the 2-qubit system with the gate applied to qubit **0** (**zero-based and from right to left**!),

$$\alpha'_{00} = q_{11}\alpha_{00} + q_{12}\alpha_{01}$$

$$\alpha'_{01} = q_{21}\alpha_{00} + q_{22}\alpha_{01}$$

$$\alpha'_{10} = q_{11}\alpha_{10} + q_{12}\alpha_{11}$$

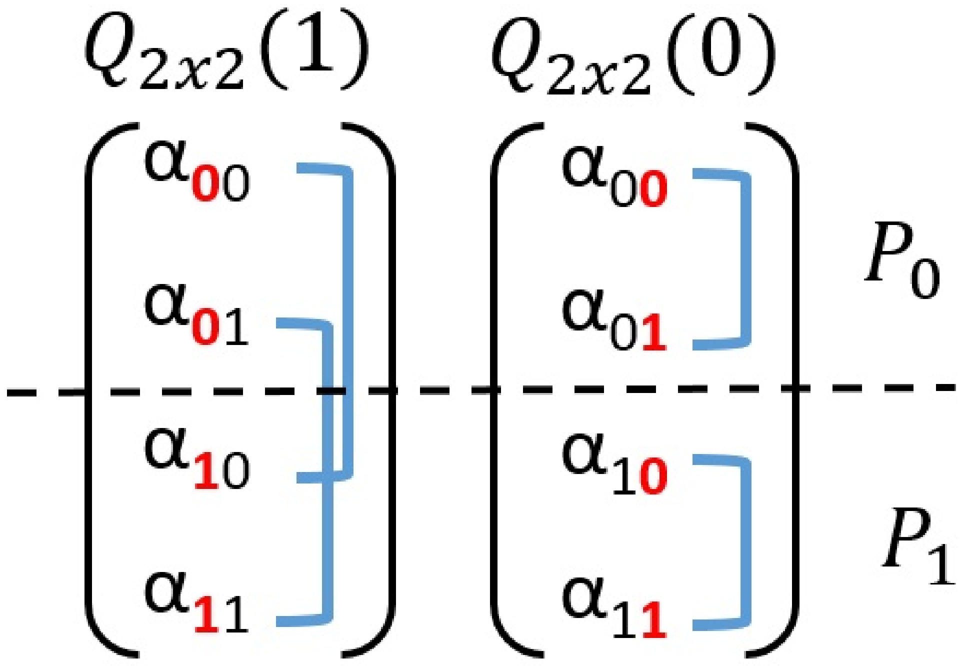
$$\alpha'_{10} = q_{21}\alpha_{10} + q_{22}\alpha_{11}$$

Amplitudes and **strides** in memory

In memory, the amplitudes of the quantum state are stored **sequentially** according to the binary representation.

Single-qubit gate operations on the qubit *k* access the elements with stride 2^k

This is the **most crucial feature** to understand the performance of IQS!



Stride **0**: sequential access

Small stride: **strided access** within the memory of a given node

Larger stride: memory on **different nodes** → **MPI** communication needed

Performance **considerations**

IQS is a **data-intensive, memory-bound** application

The **memory footprint** of the state grows as 2^{n+4} bytes with the problem size n (number of qubits)

The performance of a **gate operation** on qubit k depends on n **and** k

For simple operations, the **DRAM bandwidth** (or, between nodes, the network bandwidth) is all we need to determine the maximum performance

Scope of our work

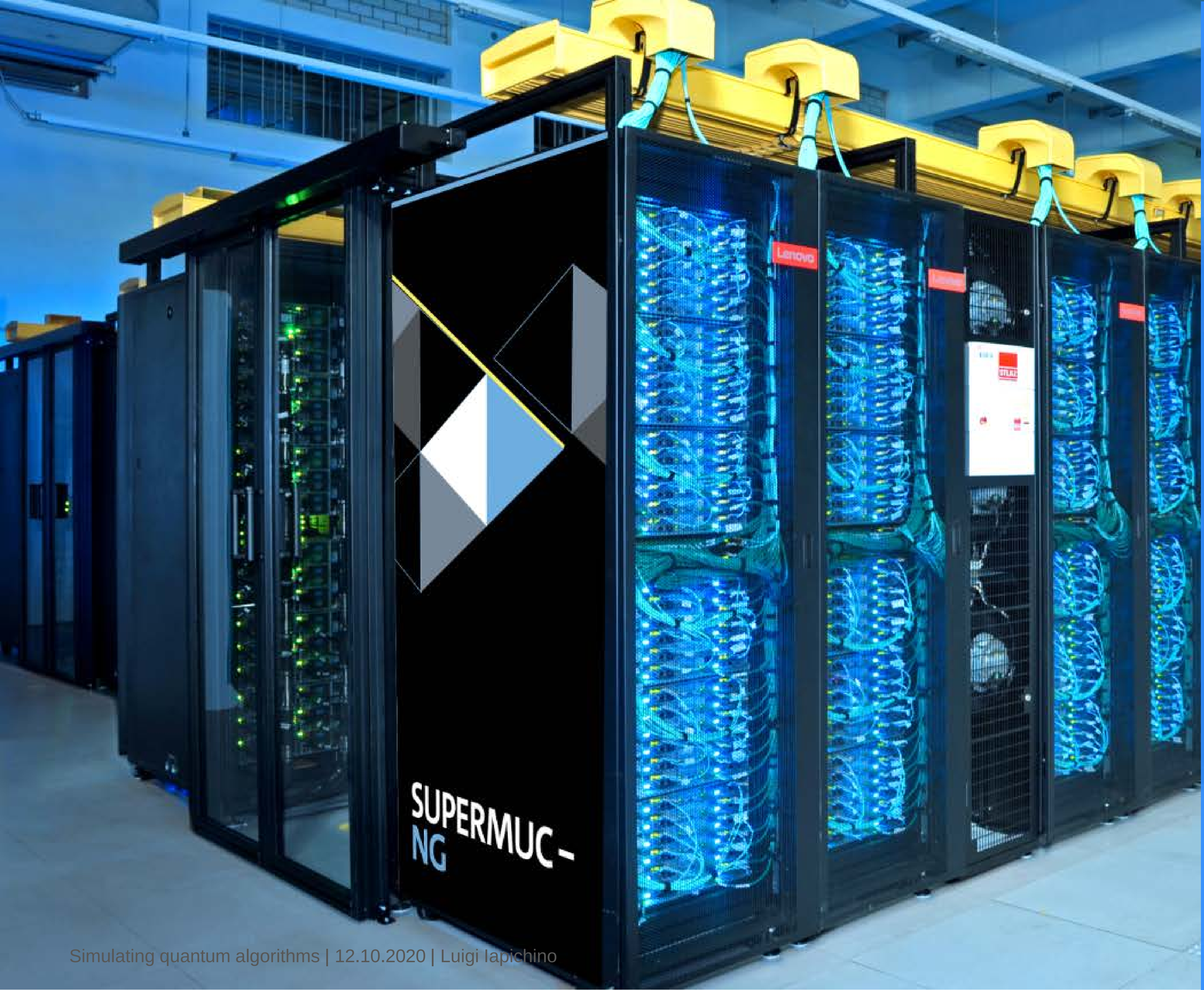
IQS has a strong focus on **efficiency on modern architectures.**

Following **optimizations** are implemented (see method paper for details):

- **Vectorization** via SIMD pragmas;
- **Threading optimization;**
- **Cache blocking** through gate fusion;
- Usage of the **Intel® MKL library.**

We want to probe mainly the **MPI layer** of the code, on a **complex test case.**

Our **target system…**



SuperMUC-NG

Top500 (June 2020): #13

Lenovo Intel (2019)

311,040 cores
Intel Xeon Skylake

26.9 PetaFlops
Peak

19.5 PetaFlops
Linpack*

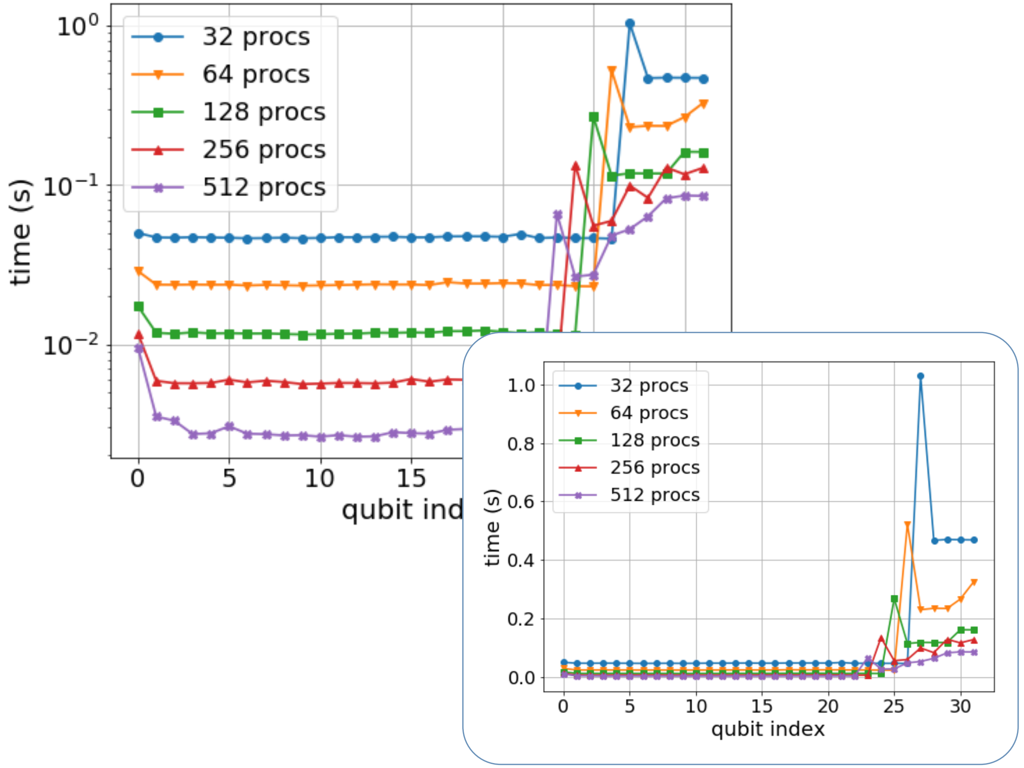
719 TeraByte
Main Memory

70 PetaByte
Disk

Scaling results on SuperMUC-NG

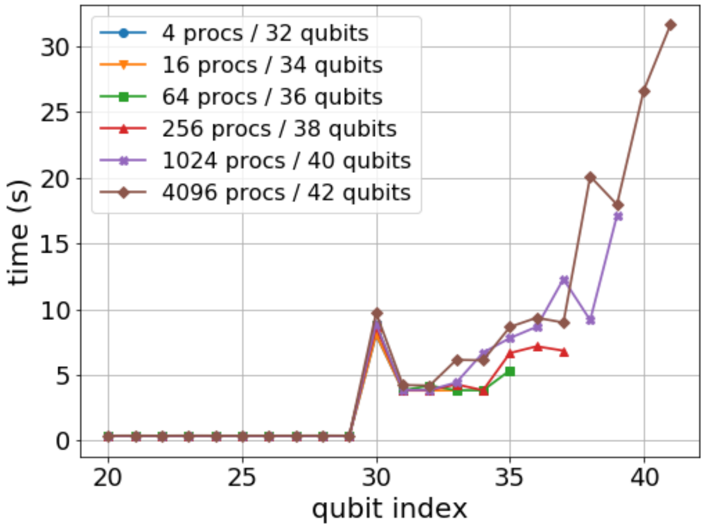
Strong scaling

Double the processes maintaining the same task.
Simulation of 1-qubit random gate



Weak scaling

Double the processes = add one qubit
Maintaining the same local size of the state vector.
Simulation of 1-qubit random gate.



Each node of SuperMUC-NG is equipped with 2 socket Intel® Xeon® Scalable Processor 8174 CPU (24 cores per socket and 96 GB DRAM per node).

Single-node profiling: setup and test case

Quantum Fourier transform: quantum analogue of the inverse discrete Fourier transform.

Non-trivial test case and part of many important quantum algorithms like e.g. **Shor's factorization** (quantum cryptography).

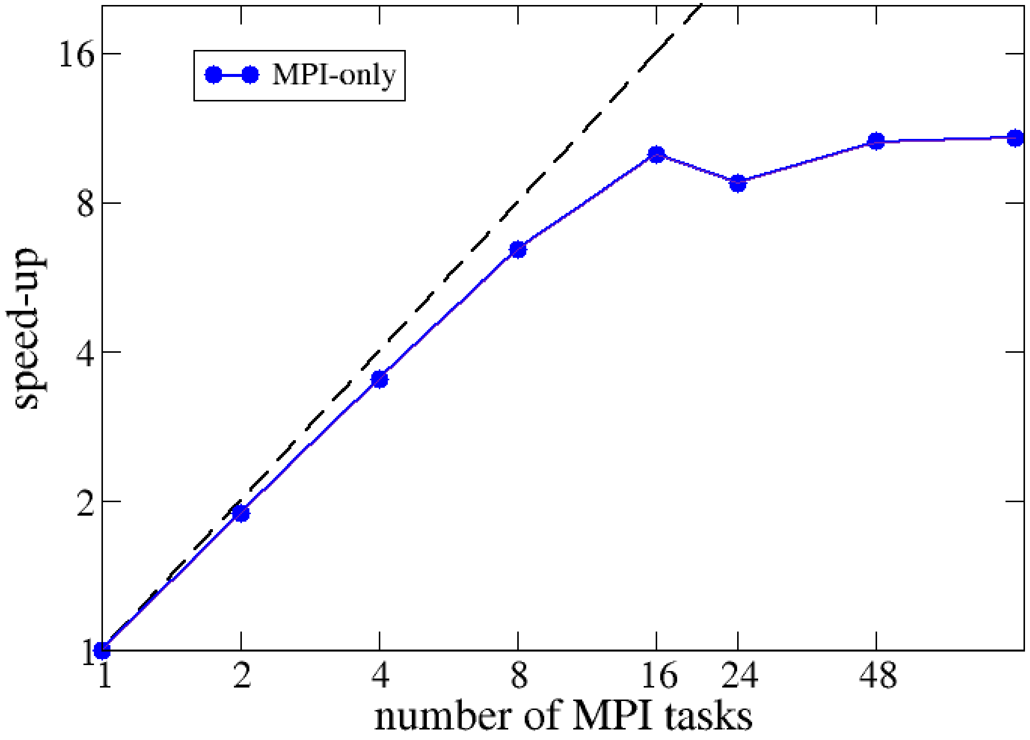
We run a system of 29 qubits (memory for the quantum state: 8GB). Both SP and DP QFT are executed.

Initial profiling on a single node: dual socket Intel® Xeon® Scalable Processor 8174 CPU (24 cores per socket, 96 GB DRAM per node).

Compilers and tools: Intel® Parallel Studio XE 2019.

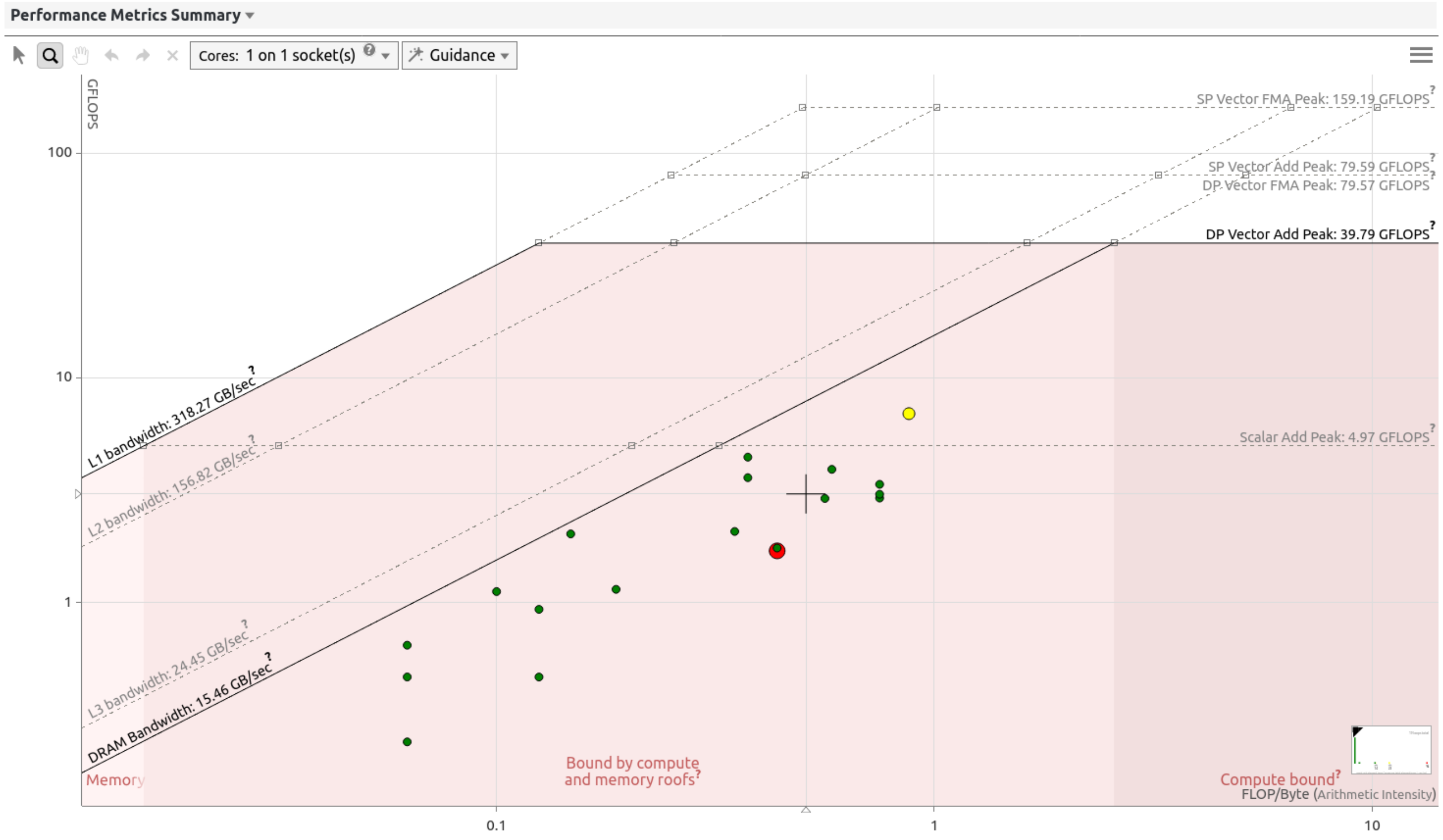
We focus on MPI scalability and performance, because the OpenMP layer is followed by a different project.

MPI scaling and Application Performance Snapshot (APS)



- MPI scaling not satisfactory.
- Hybrid parallelisation relieves somewhat the bottleneck on MPI.
- Severe bottleneck on memory/cache stalls.
- Vectorisation very good: ~ 90% of vector instruction. However, nearly none of them are 512 bit-wide.
- Forcing by compiler option the use of ZMM registers does not help.
- Despite of good fraction of vector instructions, the performance gain due to vectorisation is 1.25x.

Results from Intel Advisor



- 90% of runtime is spent in MKL instructions.
- 65% is spent in vector instructions.
- However, the top five vectorized loops use only SSE/SSE2 and AVX instructions
- The **roofline plot** shows that the code is DRAM bound, as already anticipated by APS.

Summary and outlook

- QC has a growing momentum, however the access to suitable hardware is an issue in the community.
- IQS can be used for exploring, testing and implementing quantum algorithms, making use of HPC resources to simulate quantum system of large size.
- This application has been developed keeping HPC performance in mind; simple gates perform very well, while in more complex algorithms there is room for improvements.
- For the future: prepare IQS for upcoming systems and programming models.
- QC on (pre-)Exascale machines has a potential comparable with the one of current noisy hardware.

Dr. Luigi Iapichino
Leibniz Supercomputing Centre
Luigi.Iapichino@lrz.de
bqcx@lrz.de

Simulating quantum algorithms on HPC systems: a performance perspective