A Design Framework for the Simulation of Distributed Quantum Computing

Davide Ferrari, Michele Amoretti

Quantum Software Laboratory, Università di Parma

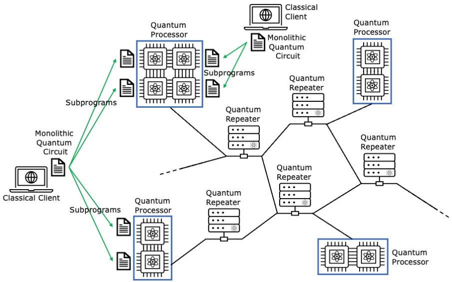
Contact: davide.ferrari1@unipr.it

High Performance and Quantum Computing Integration (HPQCI'24)
June 3, 2024 - Pisa, Italy

Summary

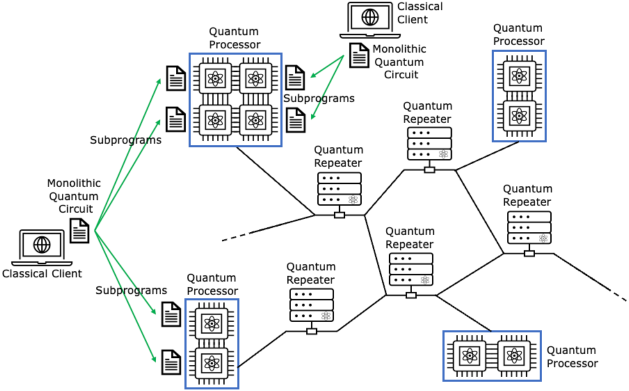
Distributed Quantum Computing (**DQC**)

- Increasing demand for large-scale quantum computers
- different Quantum Processing Units (QPUs) could communicate and cooperate
  - On multi-chip quantum processors
  - At short distance, probably integrated into HPC facilities
  - **Quantum Internet**



Distributed Quantum Computing

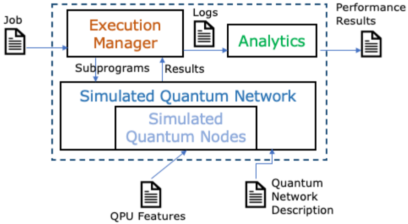
Simulation is crucial for establishing the correctness of compiled distributed quantum programs, and evaluating the quality of their execution against different network configurations, hardware platforms and scheduling algorithms.



Design Framework

The proposed design framework includes four components, namely:

- **Execution Manager**
- Simulated Quantum Network
- Simulated Quantum Nodes
- **Analytics**



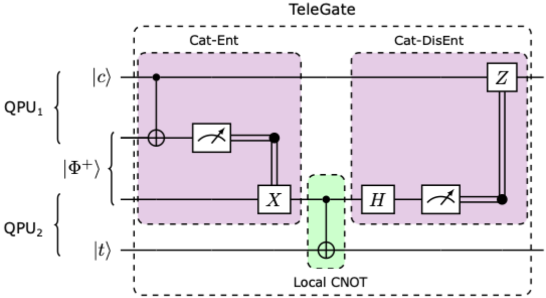
With respect to classical jobs, DQC jobs have more static properties

- number of qubits (width)
- layers of computation (depth)
- ratio between local and remote gates (computation-to-communication)

The Execution Manager could make clever decisions to deal with the *parallel job scheduling* problem (generally *NP-hard*)

The Execution Manager is characterized by:

- a **Queue** of jobs to be handled
- a **scheduling** algorithm
- a **Job** is quantum circuit that has been split into subprograms
- **Interactions between Subprograms** are defined in terms of *remote gates* requiring **shared EPR pairs** and **classical communication**



Analytics

Specific to DQC is the **performance evaluation** of the Execution Manager.

- M, makespan of a schedule
- p_i, lenght of the i-th job
- q_i, number of QPUs required by the i-th job
- n_{QPU}, number of QPUs available
- r, number of layers required to implement a remote gate
- N_{Ri}, number of remote gates in the i-th job

Analytics

Two specific metrics are proposed, going beyond the traditional concept of makespan:

- **QPU Utilization**

$$U_{QPU} = \frac{\sum_i p_i q_i}{M n_{QPU}} \in [0, 1]$$

- **Quantum Network utilization**

$$U_{QN} = \frac{\sum_i N_{Ri}}{\frac{(n_{QPU}-1)M}{r}} = \frac{r \sum_i N_{Ri}}{(n_{QPU}-1)M} \in [0, 1]$$

Computation-to-Communication Ratio (**CCR**) per job

$$\frac{\#\text{ of local gates}}{\#\text{ of remote gates}} = \frac{N_L}{N_R}$$

Job Scheduling

Given 6 QPUs and 4 jobs, $J_i(p_i, q_i, N_{Ri}, N_{Li})$:

$$J = J_1(24, 4, 9, 19), J_2(16, 3, 4, 13), J_3(32, 2, 4, 15), J_4(8, 3, 2, 6)$$

Algorithm FIFO-Scheduling

Input: job queue J, idle QPU set Q

Output:

function SCHEDULE
    $i \leftarrow 0$
    while $Q \neq \emptyset$ **do**
        $next \leftarrow J[i]$
        if $\exists q \subseteq Q : q = next.q$ **then**
            schedule *next*
            $Q \leftarrow Q \setminus q$
            $J \leftarrow J \setminus next$
        end if
    end while
end function

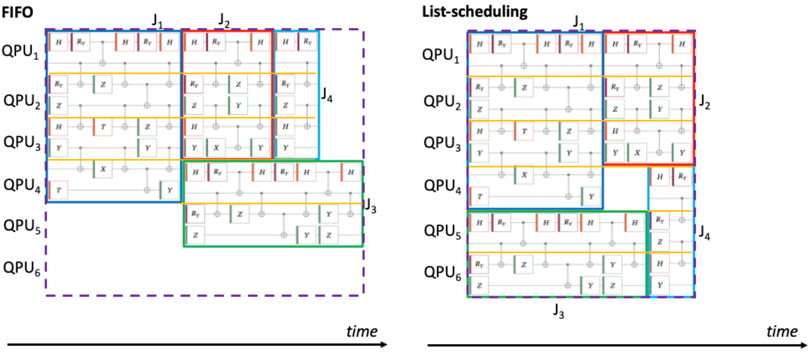
Algorithm List-Scheduling

Input: job queue J, idle QPU set Q

Output:

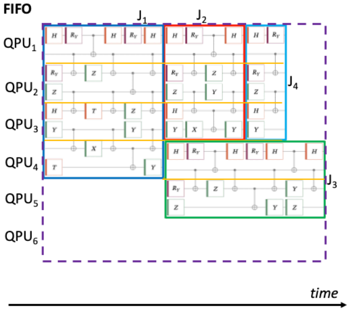
function SCHEDULE
    $i \leftarrow 0$
    while $Q \neq \emptyset$ **do**
        $next \leftarrow J[i]$
        if $\exists q \subseteq Q : q = next.q$ **then**
            schedule *next*
            $Q \leftarrow Q \setminus q$
            $J \leftarrow J \setminus next$
        else
            $i \leftarrow i + 1$
        end if
    end while
end function

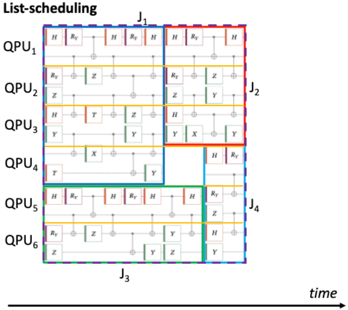
Job Scheduling



Job Name	i	p	q	N_R	N_L	CCR
J_1	1	24	4	9	19	2.11
J_2	2	16	3	4	13	3.25
J_3	3	32	2	4	15	3.75
J_4	4	8	3	2	6	3

Job Scheduling





With FIFO

- $M_{FIFO} = 56$
- $U_{QPU}^{FIFO} = 0.69$
- $U_{QN}^{FIFO} = 0.475$

With List:

- $M_{LS} = 40$
- $U_{QPU}^{LS} = 0.96$
- $U_{QN}^{LS} = 0.665$

Conclusions and Future Work

In this work we proposed:

- a design framework for DQC simulation
- two performance metrics for job scheduling

Future work:

- study and develop modules to bridge simulation tools
- study quantum specific job scheduling algorithms
- define fair share queue policies

Thank you!

Quantum Software Laboratory

http://www.qis.unipr.it/quantumsoftware.html