CESGA
GALICIA SUPERCOMPUTING CENTER

Partitioned QPU Execution for Accelerated Gradient Computation in Quantum Recurrent Neural Networks Training

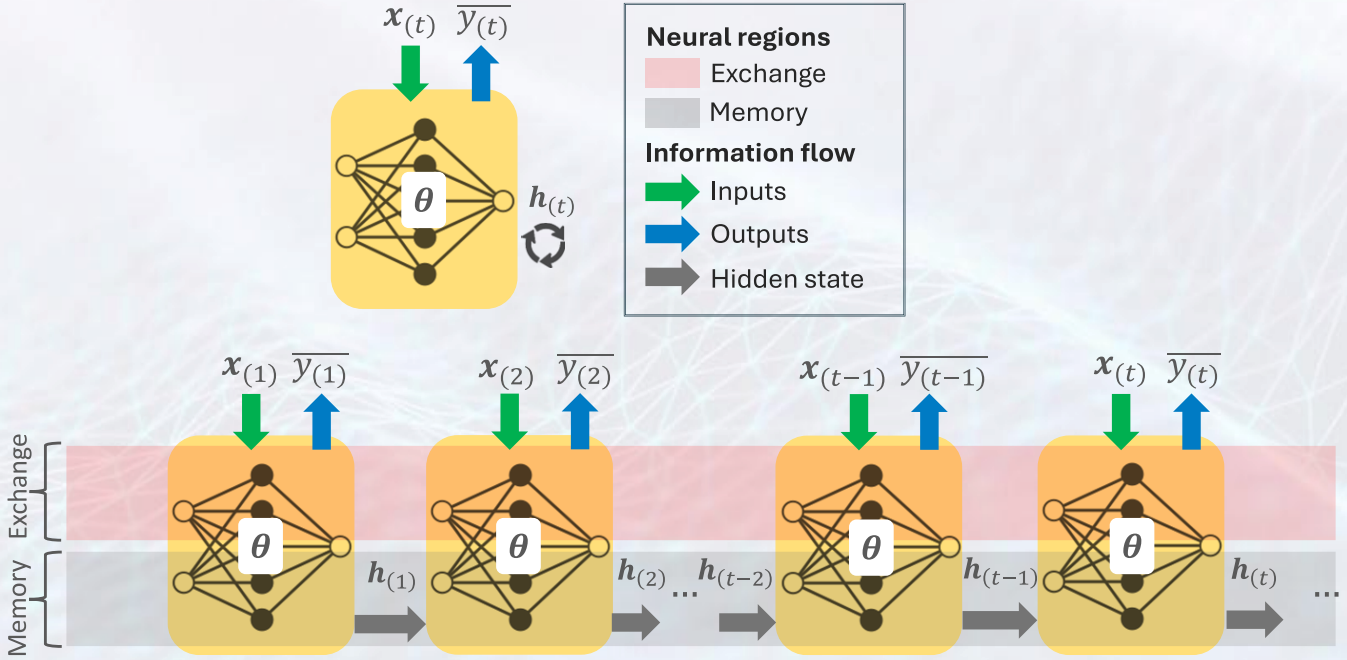
José Daniel Viqueira Cao[1,2], Andrés Gómez Tato[1], David Mera Pérez[2]

[1]*Departamento de Aplicaciones y Proyectos (Fundación CESGA)*
[2]*COGRADE, Departamento de Electrónica e Computación (USC)*

Introduction to variational Quantum Recurrent Neural Networks

From classical RNNs...

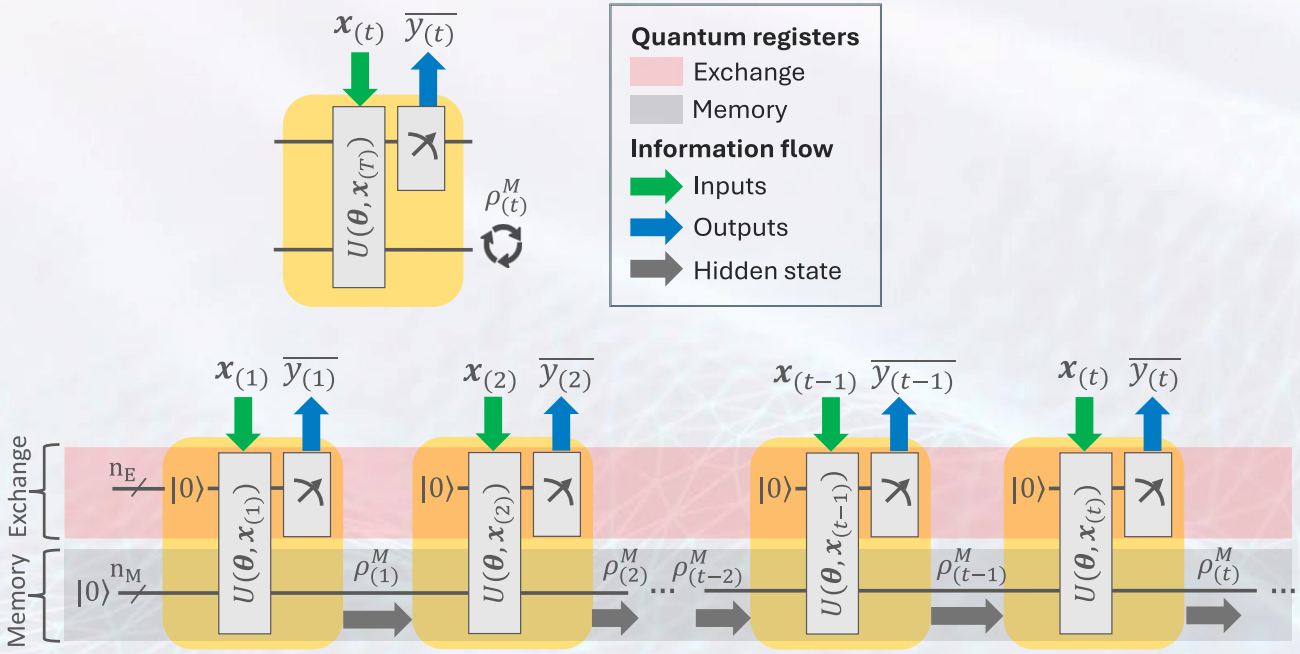
- As classical RNN, QRNN can be used for any problem of forecasting or classification involving **time series**.
- The particularity in an RNN are the recurrent connections, acting as a form of memory on past inputs.



- We focus on time series **forecasting**.

... to variational QRNNs

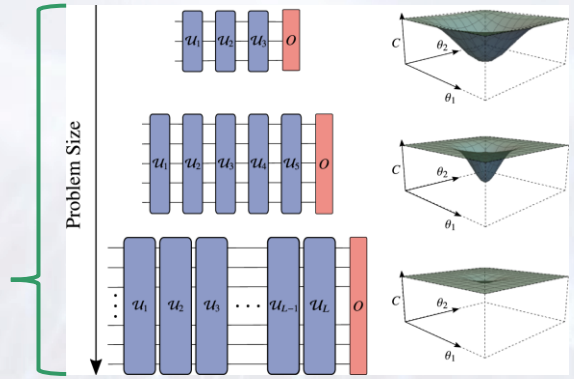
- Equivalently, the recurrent connections are present.



- Potential advantages:
  - Exponentially large Hilbert spaces.
  - Desired properties of echo-state networks.

Training VQAs is a challenge

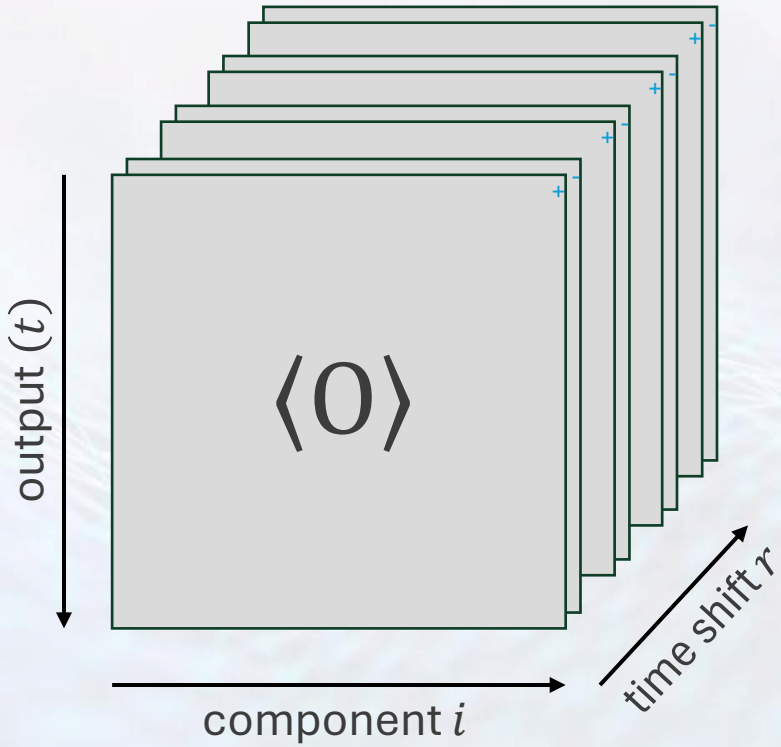
- QRNN belong to the variational quantum algorithms (VQA) family. A parameterised quantum circuit (PQC) is trained with a classical optimizer.
  - In a quantum machine learning (QML) setting, the PQC acts as a neural network.
- Theoretical threat: ***barren plateaus***.
- Practical limitations (regarding gradient descent):
  - The powerful *backpropagation* algorithm in classical ML is no longer available (wavefunction collapse).
  - *Finite difference* methods suffer from severe numerical instability due to the stochastic nature of quantum measurements.
  - Derivatives can be computed analytically through *Parameter-Shift Rule*, requiring $2 \times N_\theta$ circuit evaluations.

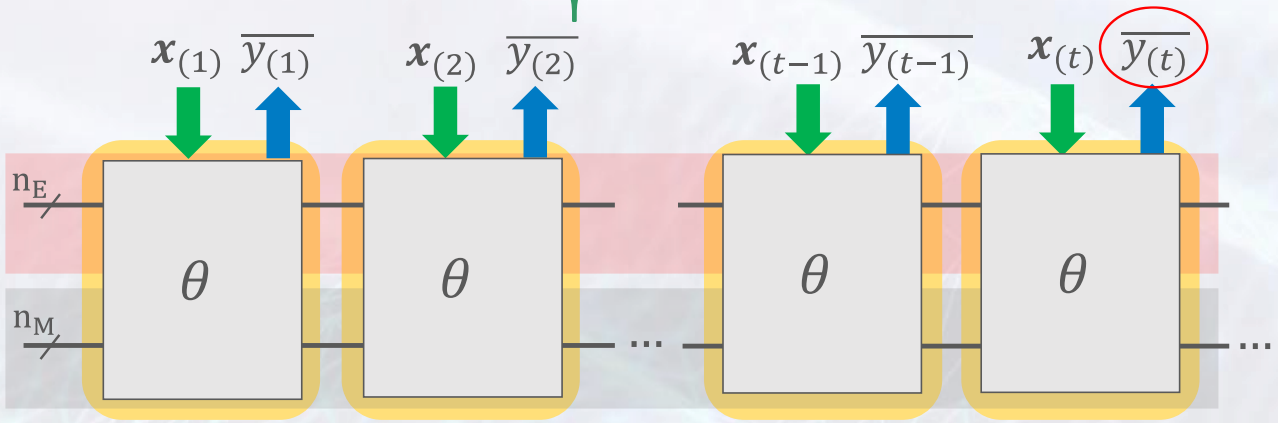


From Wang et al. *(2021) Noise-induced barren plateaus in variational quantum algorithms*

QRNNs require more circuit evaluations

- For a sample of length n_T, the total cost of a gradient evaluation is $2 \times N_\theta \times n_T$ circuit evaluations.
  - One gradient component requires $2 \times t$ circuit evaluations with **independent parameter shifts** inside each one.





$$\overline{y_{(t)}} = f\left(\langle O_{(t)} \rangle\right)$$

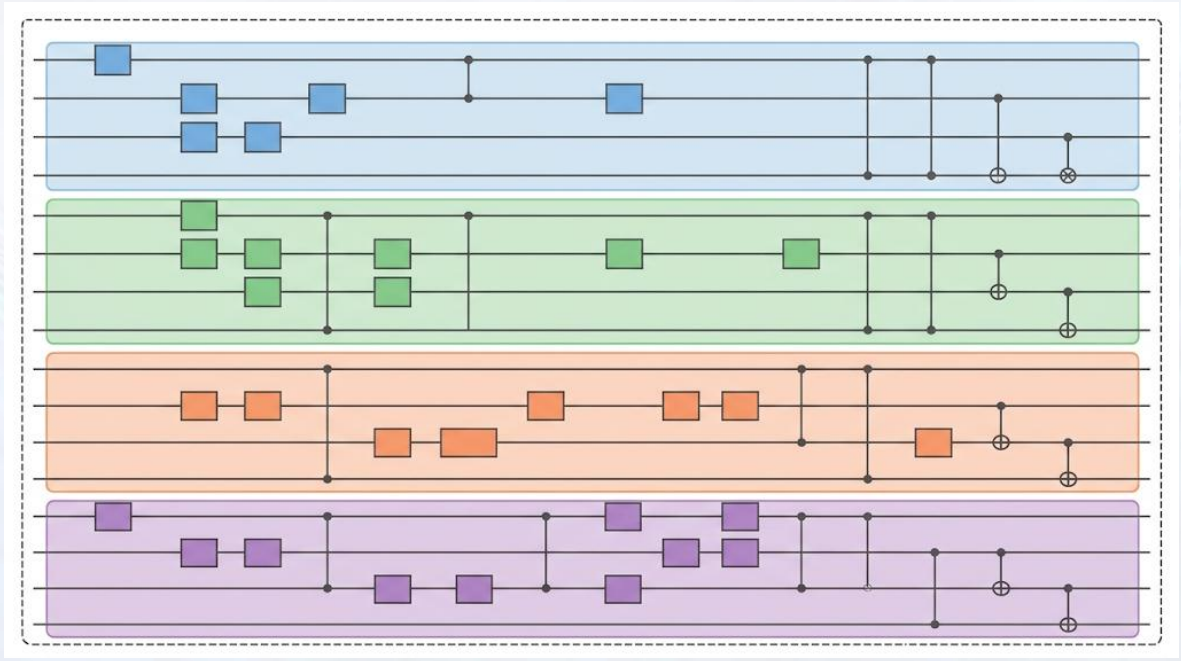
$$\partial_i \langle O_{(t)} \rangle \equiv \frac{\partial \langle O_{(t)} \rangle}{\partial \theta_i} = \sum_{r=0}^{t} \frac{1}{2} \left(\langle O^+_{(t)} \rangle - \langle O^-_{(t)} \rangle \right) \Big|_{ri}$$

Shifts are $\pm \pi/2$.

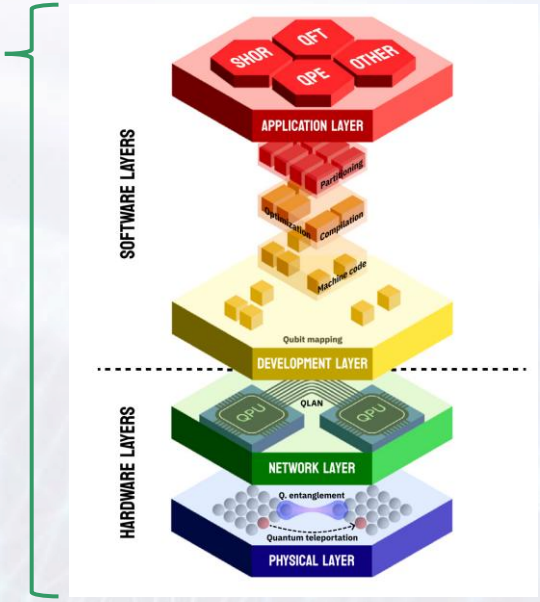
QPU partitioning for parallel quantum circuit evaluation

Parallelization as the solution

- Distributed Quantum Computing
  - *Embarrasingly parallel* architecture.
- QPU partitioning.
  - Immediate solution to the problem of scarce quantum computers.

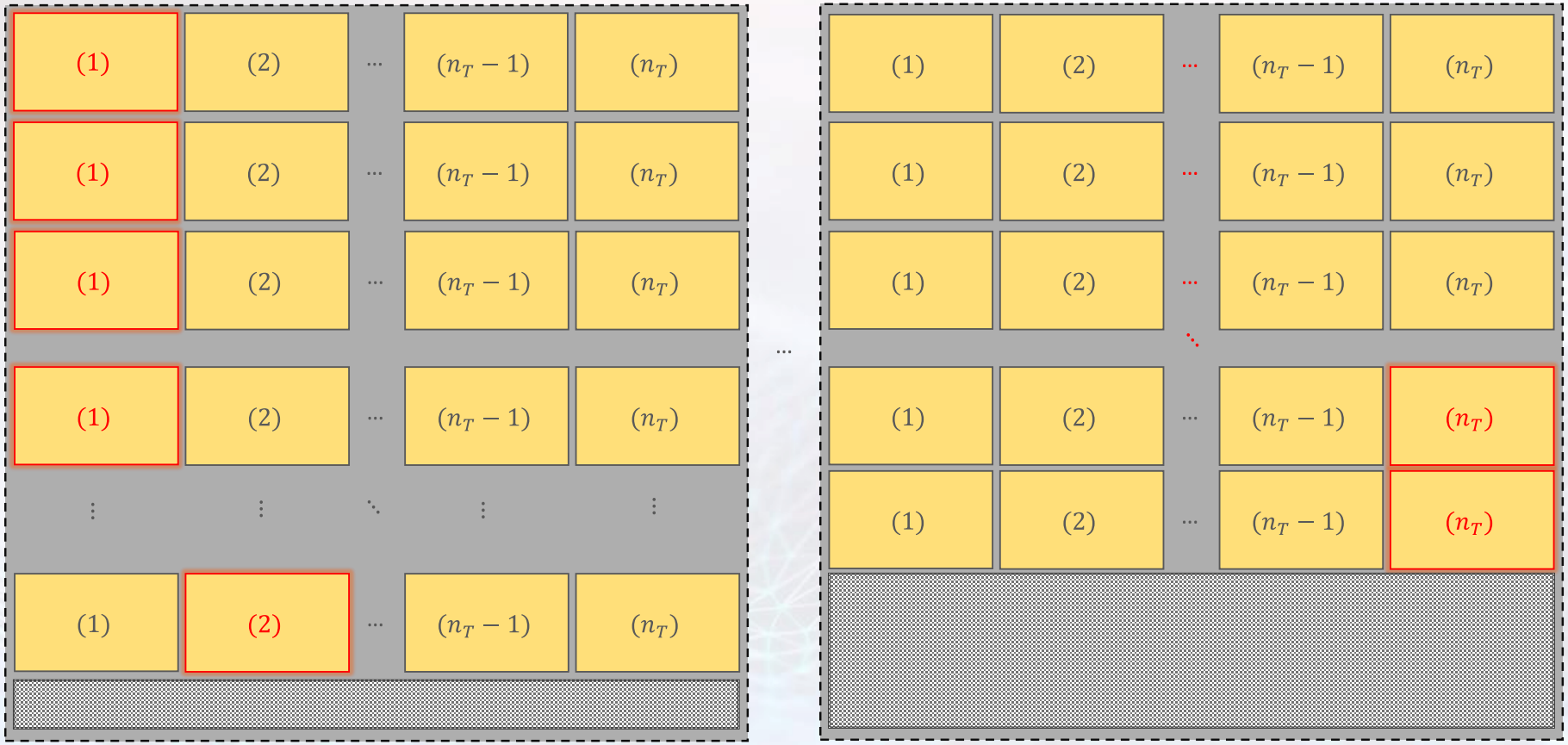


AI generated image



From Barral et al. *(2025) Review of Distributed Quantum Computing: From single QPU to High Performance Quantum Computing*

Parallel on partitioned QPU



✅ Reduced elapsed times

✅ Reduced idle qubits

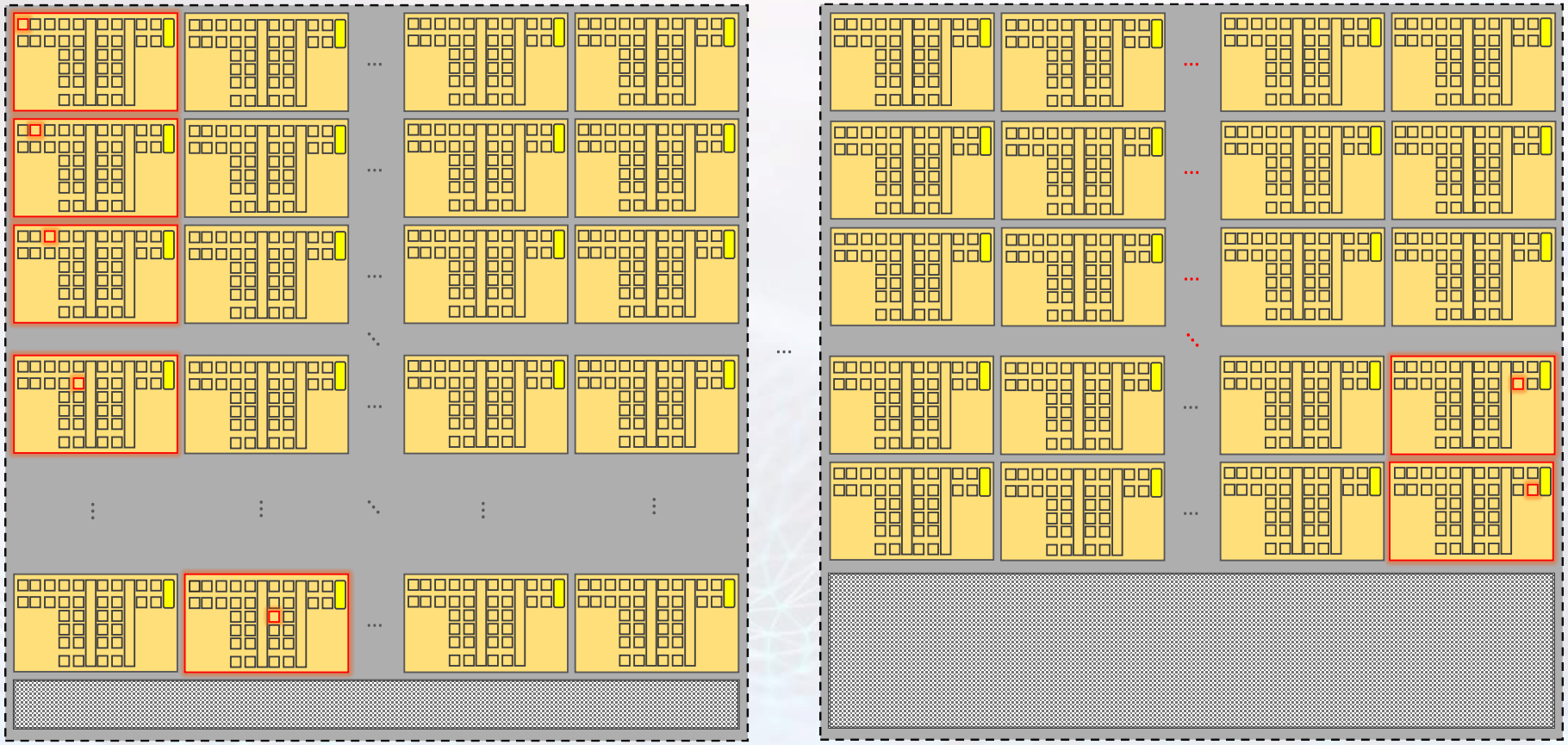
✅ Reduced energy and money costs

✅ Easy scheduling due to constant structure

❗ Restricted qubit layout choice

❗ Possible crosstalk between qubits

Parallel on partitioned QPU



✅ Reduced elapsed times

✅ Reduced idle qubits

✅ Reduced energy and money costs

✅ Easy scheduling due to constant structure

❗ Restricted qubit layout choice

❗ Possible crosstalk between qubits

Objective

- **Purpose**: accelerated training of variational QRNN with parallel gradient computation on quantum hardware.
- **Scope**: get gradient evaluations with parallelization in a partitioned QPU.

Methodology

Methodology overview

- We follow a methodology that allows us to get a **metric** measuring the distance between an ideal (simulated) baseline and gradients computed through hardware.
- Steps
  1) Get a set of parameters $\boldsymbol{\theta}$ from an ideal training. Get corresponding baseline gradient.
  2) Evaluate ideal gradients with shot noise.
  3) Evaluate the corresponding gradients on hardware.
  4) Get the distance of gradients evaluated on hardware with respect to the baseline.

1) Ideal training for getting baseline gradient

- We train a model through density matrix (DM) emulation (ideal exact outputs).
  - Dataset: NARMA-3.
- We choose the epoch with higher average norm of loss gradients.
- We choose the data sample with higher norm of gradient. The corresponding loss gradient is our **baseline**.

2,3,4) Evaluation and comparison against ideal machines

2) Simulate the gradient evaluation 100 times with a classical simulator (Qiskit's Aer simulator) including sampling (shot) noise.
   - This gives us the empirical distribution of distances with respect to the baseline for a hypothetical ideal quantum computer.
3) Compute on hardware (**54-qubits IQM Emerald**).
   - Compute the gradient **sequentially** 8 times in hardware.
   - Compute the gradient **in parallel** 8 times in hardware —with and without dynamical decoupling (DD).
4) The distance metric between the baseline gradient vector $\boldsymbol{b}$ and an experimental gradient vector $\boldsymbol{g}$ is the slope s,

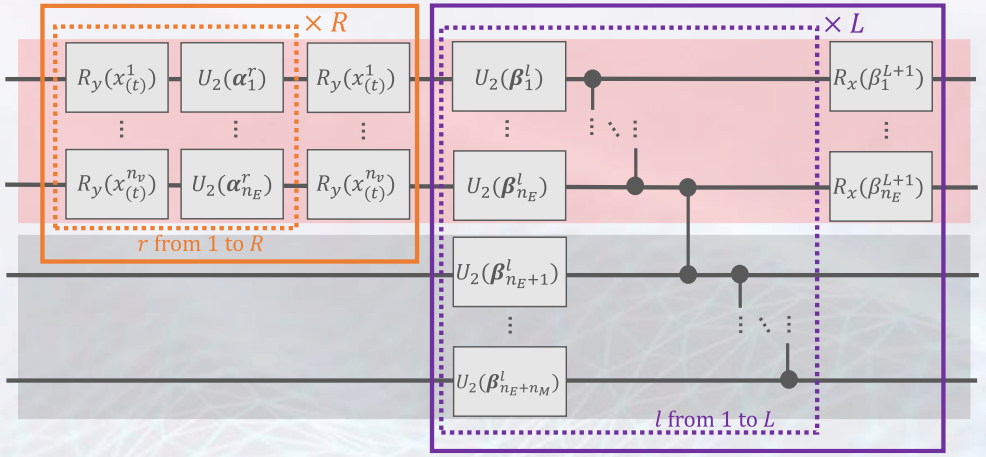
$$s = \frac{\|\boldsymbol{g}\|}{\|\boldsymbol{b}\|}\hat{b}\cdot\hat{g} = \frac{\boldsymbol{b}\cdot\boldsymbol{g}}{\|b\|^2},$$

that accounts for both the **angle** between the directions and the **contraction** of norms.

Details	
# shots	1024
loss	Mean squared error (MSE)

Model details

$U(\boldsymbol{\theta}, \boldsymbol{x}_{(t)})$ =



Quantum Registers:

- Exchange (n_E qubits)
- Memory (n_M qubits)

Sub-operators:

- Data encoding
- Entanglement

3-parameter gate:

$U_2(\boldsymbol{\theta}_{j,i}) = R_x(\theta_{j,i,0}) - R_z(\theta_{j,i,1})$

$\boldsymbol{\alpha}_i^r, \boldsymbol{\beta}_i^l \in \{\boldsymbol{\theta}_i^l\}$

Hyperparameters
$\boldsymbol{n_T = 10}$
$N_{wout} = 5$
$n_E = 1$
$n_M = 2$
$L = 3$
$R = 3$
$\boldsymbol{N_{qubits} = 3}$
$\boldsymbol{N_\theta = 25}$
$\boldsymbol{n_{CNOT} = 60}$ **(depth)**

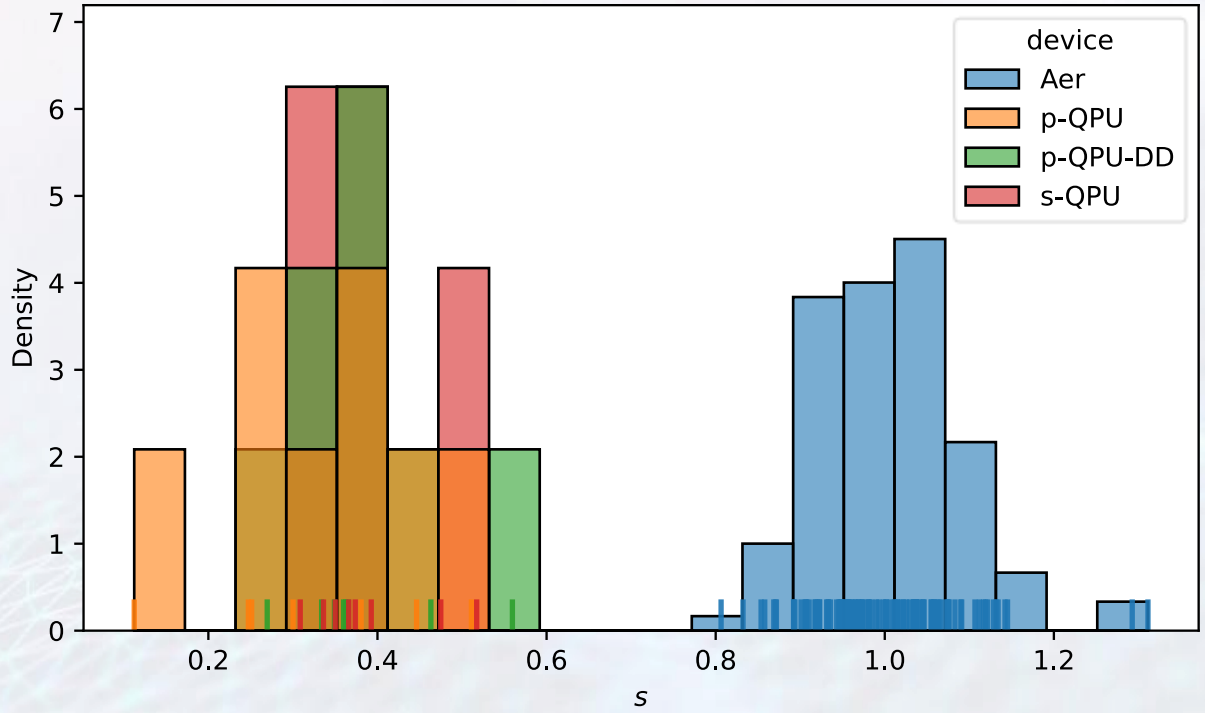
18 circuits in parallel

N_{wout} is the number of outputs involved in loss computation

Results

Loss gradient comparison

Comparison between **Aer** simulation, partitioned QPU (**p-QPU**), partitioned QPU with Dynamical Decoupling (**p-QPU-DD**) and sequential QPU (**s-QPU**) gradient evaluations of the loss function.



$$s = \frac{\boldsymbol{b} \cdot \boldsymbol{g}}{\|b\|^2}$$

Null hypothesis H_0: QPU samples come from the same probability distribution as those of Aer.

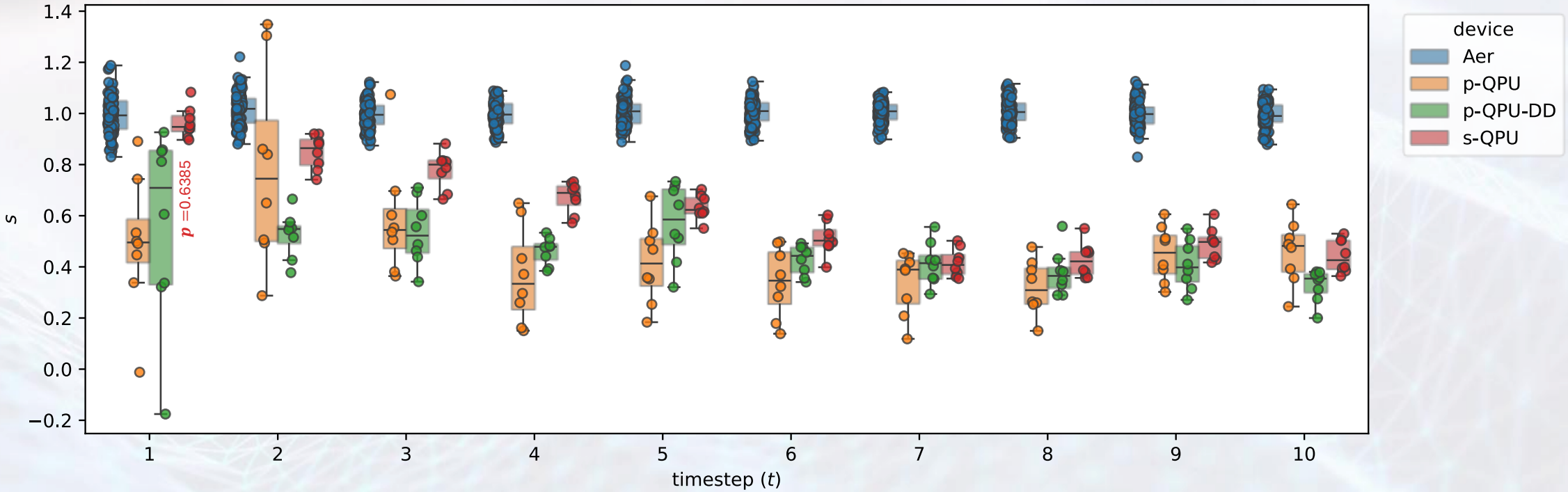
$\boldsymbol{p - value = 0}$ The number of times that a random sub-sample of the reference distribution (Aer's) gets a log-likelihood lower than the observed one is 0.

The null hypothesis is **rejected**.

Timestep gradients comparison

Slopes when comparing gradients of outputs per timestep (one subplot per timestep).

- This is the step before the gradient evaluation of the loss function (computed with the chain rule).

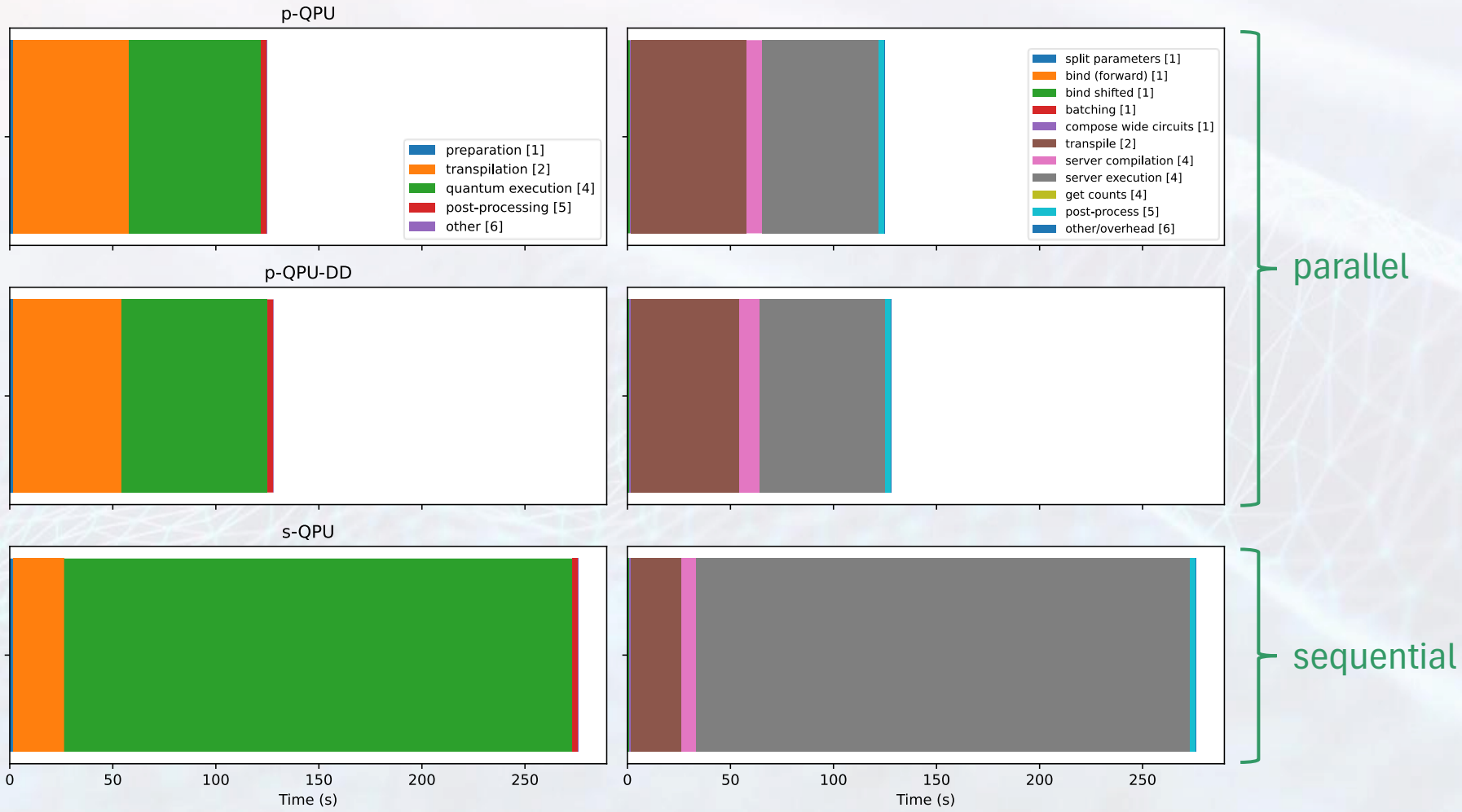


Null hypothesis H_0: QPU samples come from the same probability distribution as those of Aer.

Only timestep 1 of sequential QPU cannot be rejected.

Timings comparison

Time monitoring for sequential steps, from circuit preparation to gradient evaluation of the loss function.



Conclusions

Conclusions and future work

- Implemented successful QPU partitioning as a way of accelerating gradient evaluation in the training of variational QRNN.
- Results show that hardware gradient evaluations are not close to the baseline,
  - either in direction, norm or both;
  - but they could be enough for training purposes, because the optimiser could learn systematic errors;
  - moreover, parallel evaluation returns similar results to sequential, specially at advanced timesteps (likely due to decoherence).
- Timings are substantially reduced,
  - but not the theoretical factor, which in the case shown should be ~18.
- Future work:
  - explore crosstalk effects;
  - decrease transpilation time.

Thank you!

jdviqueira@cesga.es

Cofinanciado por la Unión Europea