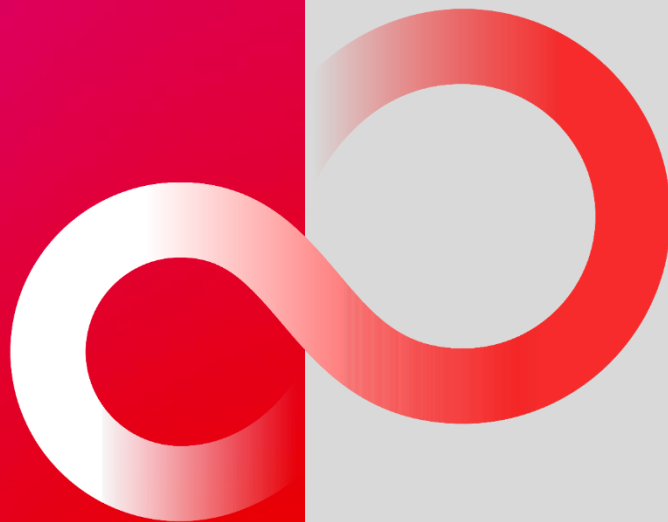
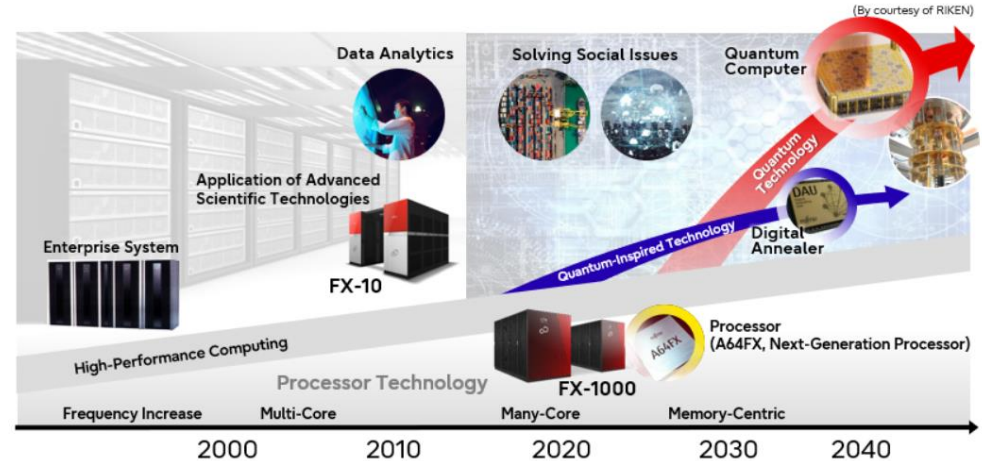


Emulación Cuántica – MPI Qulacs



- Formador:
 - Álvaro Caride
 - Fujitsu-CESGA
 - Quantum computing platform engineer
- Instalación QMIO
 - IQC
- Curso de formación emulación cuántica en Qmio
 - Scheduling
 - Foco



● Arquitectura

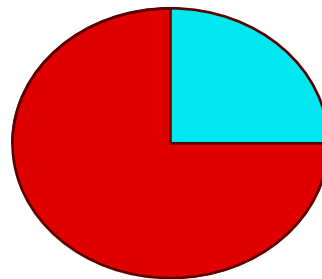
- Cluster Híbrido
- Overview instalación
- FX700
- MPI-Qulacs

● Características del Sistema

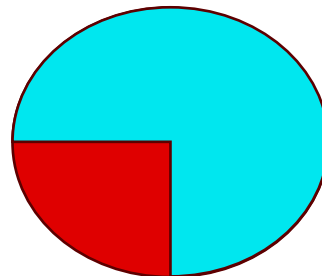
- Gestión de recursos
- Software Stack
- Ventajas y limitaciones

● ¿Cómo utilizar el Sistema?

- qulacs
- Implementación en paralela
- Hands-On (Dojo)
 - Base
 - Paralelización
 - Ejemplo
- Migración al QC

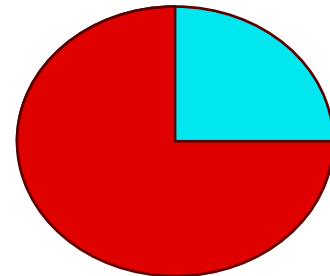


Día 1

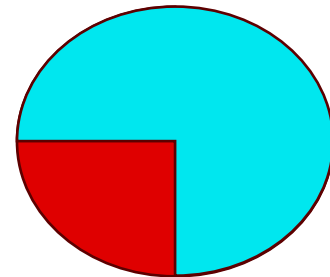


Día 2

- Intro
- Primer Lanzamiento
- Arquitectura
- Software Stack
- Revisión del primer lanzamiento
- Resultados



- Intro
- Overview Qulacs base
- Qulacs y modelos de ruido
- Qulacs y circuitos paramétricos
- Implementación Paralela
- Migración a Qmio
- Resultados



- ¿Cuánto estas familiarizada/familiarizado con emulación cuántica?
- ¿Qué lenguaje de programación prefieres?
- ¿En qué rango de qubits sueles trabajar?
 - ¿Teneis posibilidad de lanzar al cluster?

Primer lanzamiento

```
from qulacs import QuantumState, QuantumCircuit
from mpi4py import MPI

comm = MPI.COMM_WORLD

qubits = 5
state = QuantumState(qubits)
state.set_zero_state()

circuit = QuantumCircuit(qubits)
circuit.add_H_gate(0)
for i in range(1, qubits):
    circuit.add_CNOT_gate(0, i)

circuit.update_quantum_state(state)
print(state.get_vector())
```



```
#!/bin/bash
#SBATCH -p a64
#SBATCH -N 1 > > # Numero de nodos
#SBATCH --tasks-per-node=1> > # Número de ranks mpi por
#SBATCH -c 48
#SBATCH -t 5:0:0
#SBATCH --mem-per-cpu=600M

source /etc/profile.d/lmod.sh

export OMP_NUM_THREADS=48
export QULACS_NUM_THREADS=48

module load qulacs-hpcx

# Casos grandes se benefician de
#OMP_PROC_BIND=TRUE
#numactl -N 0-3 -m 0-3

mpirun -npernode ${SLURM_NTASKS_PER_NODE} python ghz.py
```

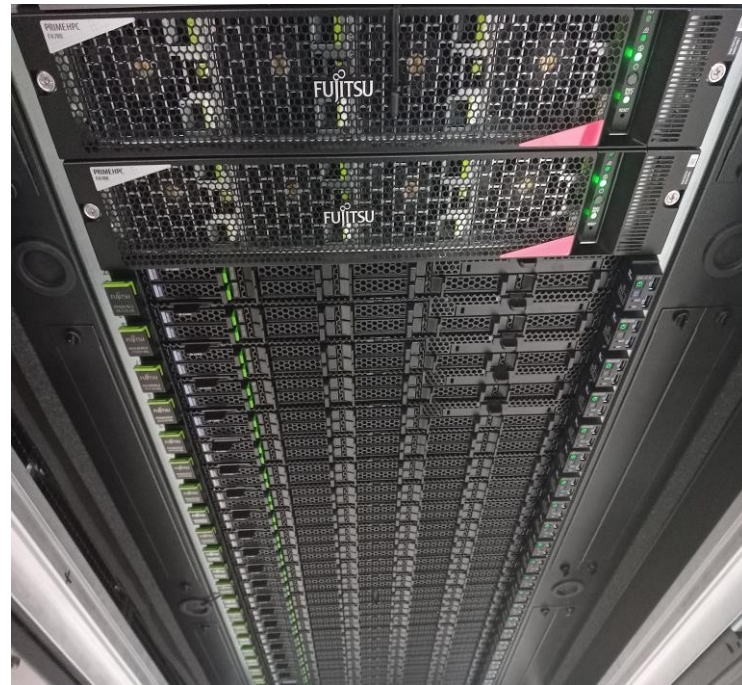
```
25 libfabric/1.13.0 loaded
26 openmpi4/4.1.4 loaded
27 py3-mpi4py/3.1.3 loaded
28 hpcx-mt-mpi loaded
29 qulacs-hpcx/1.0 loaded
30 [0.70710678+0.j 0.          +0.j 0.          +0.j 0.          +0.j
31 0.          +0.j 0.          +0.j 0.          +0.j 0.          +0.j
32 0.          +0.j 0.          +0.j 0.          +0.j 0.          +0.j
33 0.          +0.j 0.          +0.j 0.          +0.j 0.          +0.j
34 0.          +0.j 0.          +0.j 0.          +0.j 0.          +0.j
35 0.          +0.j 0.          +0.j 0.          +0.j 0.          +0.j
36 0.          +0.j 0.          +0.j 0.          +0.j 0.          +0.j
37 0.          +0.j 0.          +0.j 0.          +0.j 0.70710678+0.j]
```

Arquitectura

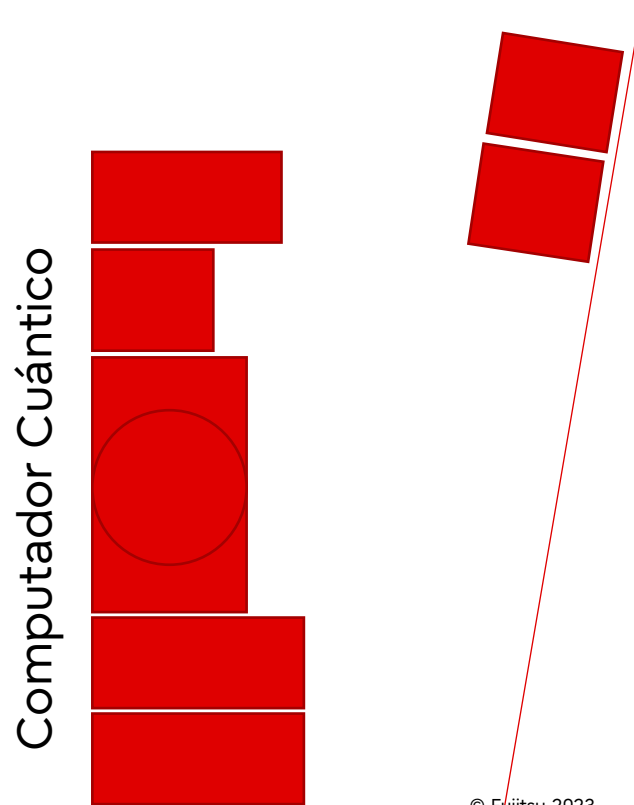
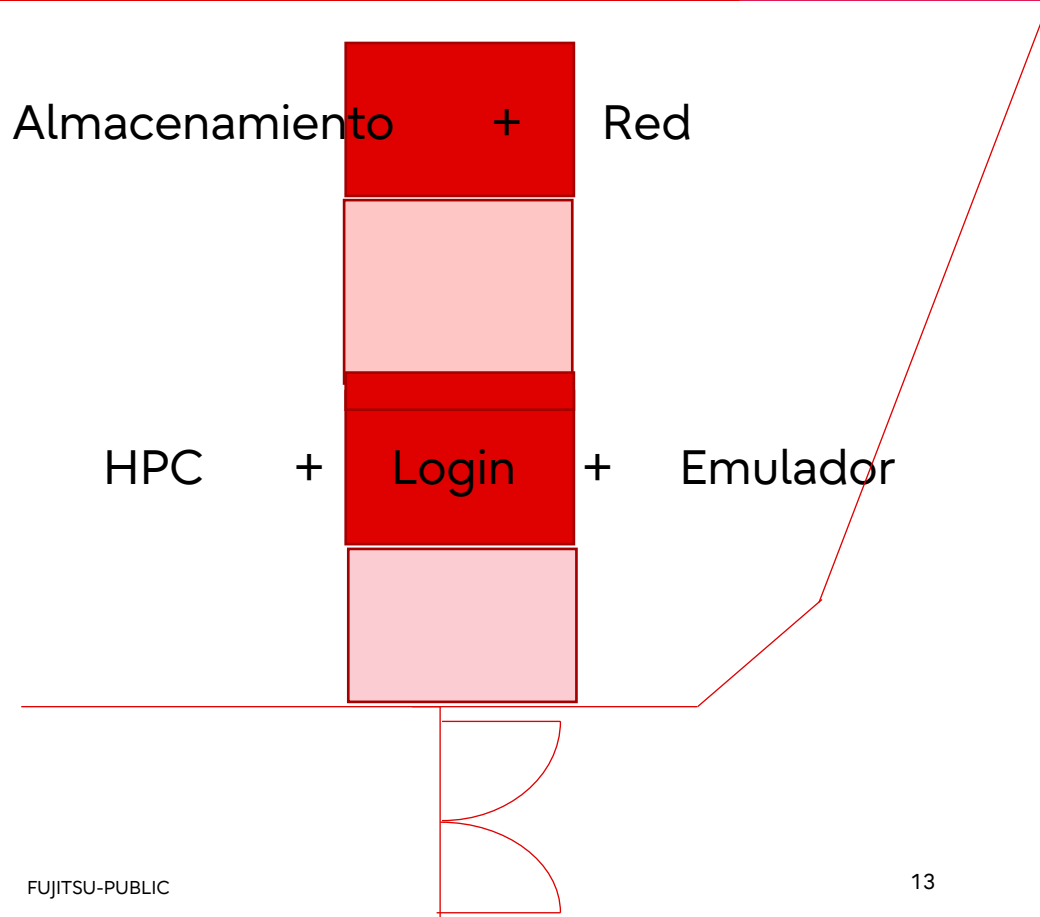
¿Qué es emulador Cuántico?

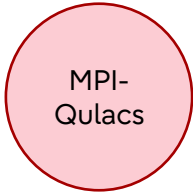
Piezas del cluster híbrido

- Nodos de Administración
- Nodos de login
- Almacenamiento
 - Cabina netapp – NFS
 - Lustre
- Nodos de Cómputo
 - HPC – x86_64
 - **Emulador Cuántico – A64FX**
 - Computador Cuántico - Superconductor

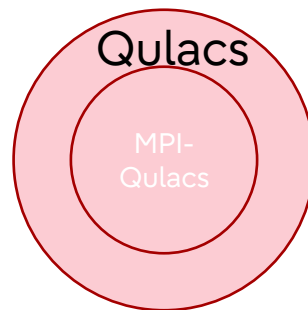


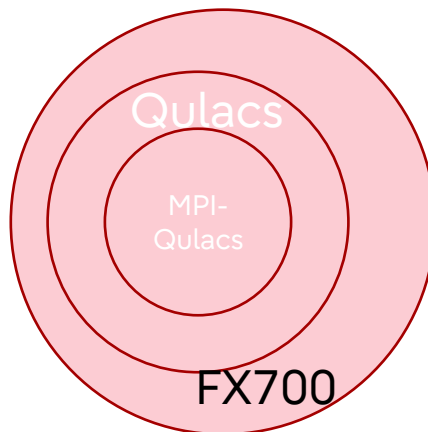
¿Dónde?

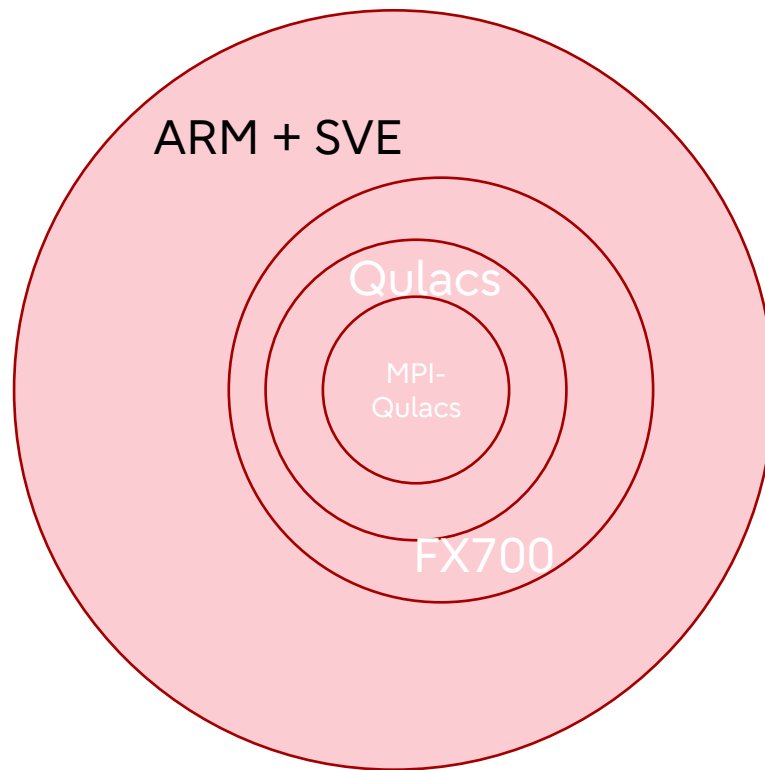


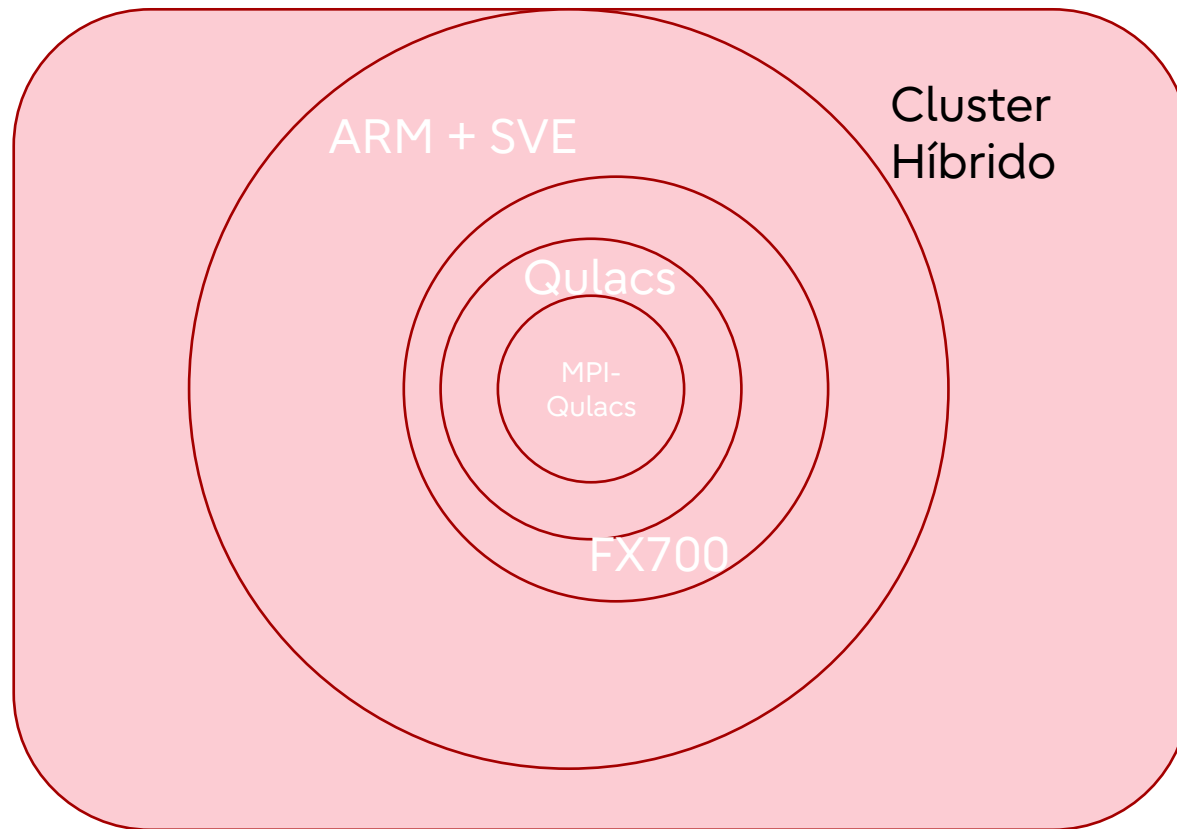
A light pink circle with a dark pink border, containing the text "MPI-Qulacs" in a black, sans-serif font.

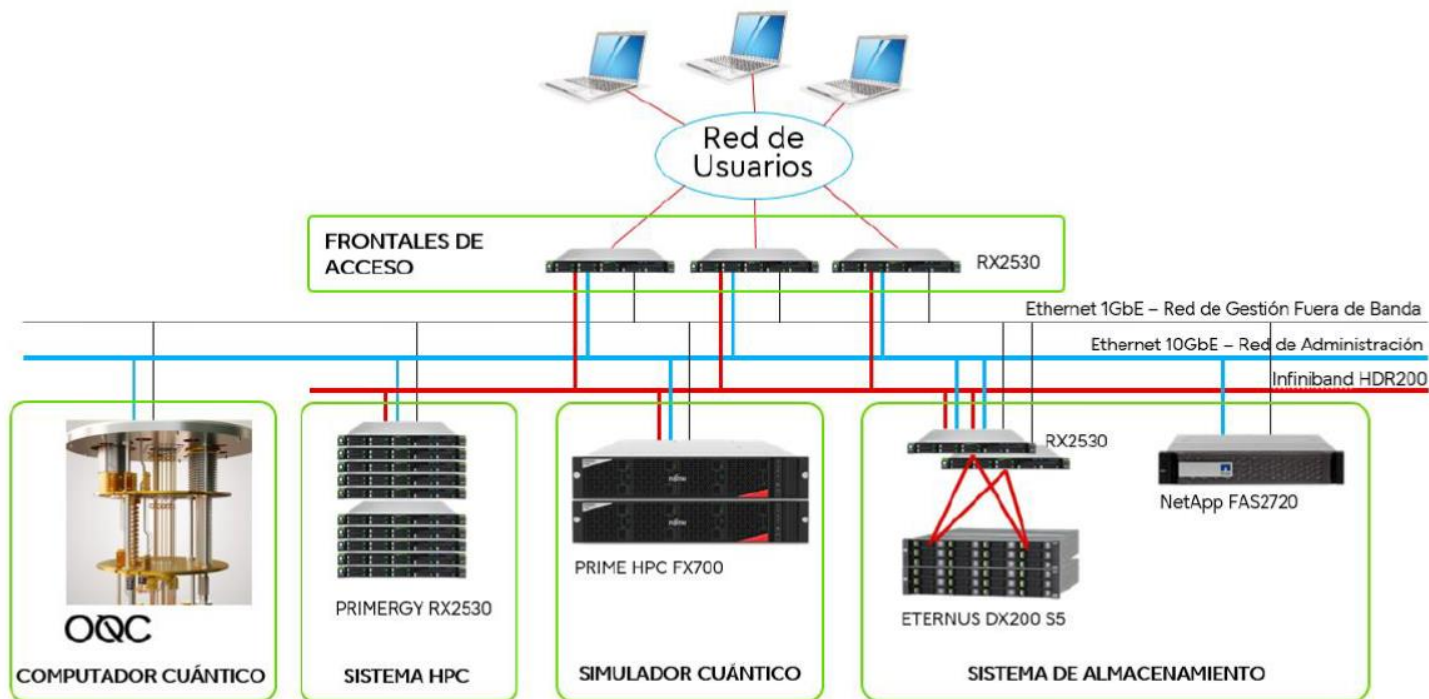
MPI-
Qulacs



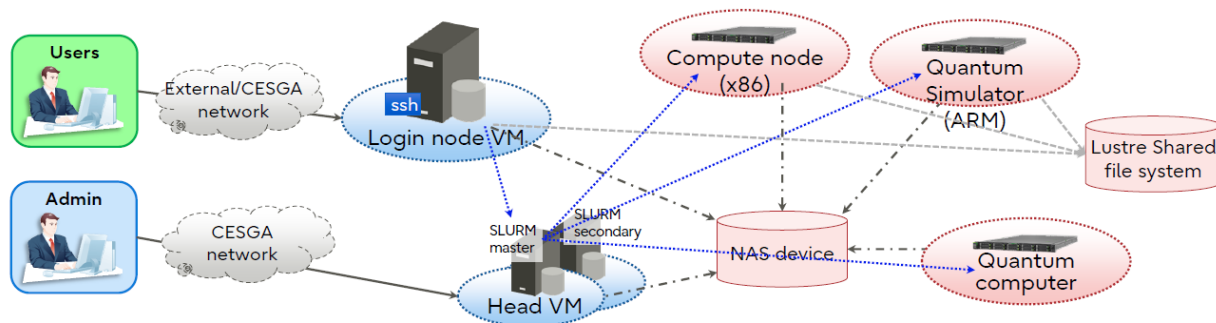








- Nodos de Administración
 - Controlador de colas
- Nodos de login
 - Acceso
 - Qmio.cesga.es



batchlim

- Almacenamiento

- Cabina netapp – NFS
- Lustre

- \$HOME
- \$STORE
- \$LUSTRE

- \$Q_SWAP

myquota

- Nodos de Cómputo

- HPC – x86_64
- Emulador Cuántico – A64FX
- Computador Cuántico - Superconductor



compute

- Nodos de Cómputo

- HPC – x86_64 -> -p ilk
- **Emulador Cuántico – A64FX -> -p a64**
- Computador Cuántico – Superconductor -> -p qpu

```
acaride@login03 ~  
$ sinfo  
PARTITION      AVAIL  TIMELIMIT  NODES  STATE NODELIST  
qpu             up      infinite    1    idle c7-23  
ilk_interactive up      infinite    2    idle c7-[1-2]  
ilk*            up      infinite   20    idle c7-[3-22]  
a64             up      infinite   16    idle c7-[101-116]  
acaride@login03 ~
```

sinfo
scontrol
...

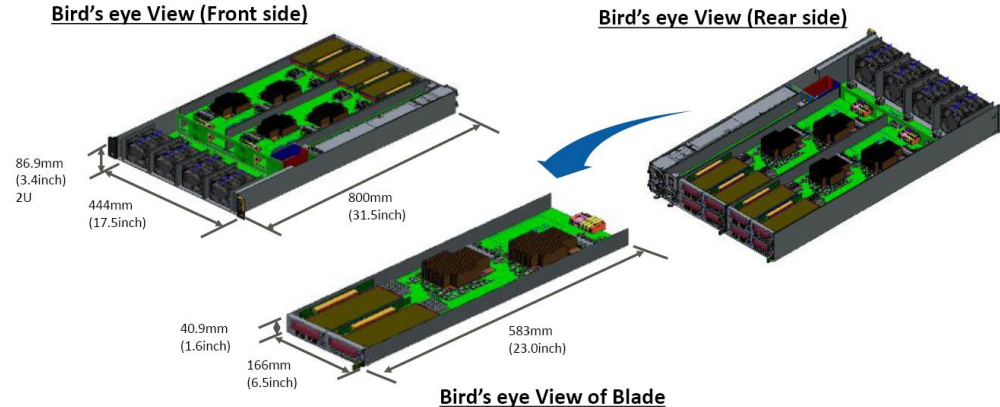
```
acaride@login03 ~  
$ scontrol show Node=c7-101  
NodeName=c7-101 Arch=aarch64 CoresPerSocket=12  
CPUAlloc=0 CPUEfctv=48 CPUTot=48 CPULoad=0.08  
AvailableFeatures=a64  
ActiveFeatures=a64  
Gres=(null)  
NodeAddr=c7-101 NodeHostName=c7-101 Version=23.11.4  
OS=Linux 4.18.0-425.3.1.el8.aarch64 #1 SMP Thu Nov 10 00:36:38 UTC 2022  
RealMemory=30000 AllocMem=0 FreeMem=22343 Sockets=4 Boards=1  
State=IDLE ThreadsPerCore=1 TmpDisk=0 Weight=30 Owner=N/A MCS_label=N/A  
Partitions=a64  
BootTime=2024-04-18T09:56:35 SlurmdStartTime=2024-05-08T15:35:04  
LastBusyTime=2024-05-12T10:53:05 ResumeAfterTime=None  
CfgTRES=cpu=48,mem=30000M,billing=48  
AllocTRES=  
CapWatts=n/a  
CurrentWatts=0 AveWatts=0  
ExtSensorsJoules=n/a ExtSensorsWatts=0 ExtSensorsTemp=n/a
```

● Hardware Específico:

- 2 Chasis (4us) correspondientes a 16 nodos en total.
- Cada nodo cuenta con un procesador A64FX con 48 cores, NoC y 32 GB de memoria. Están interconectados por IB.
- Procesadores ARM 64 bits con instrucciones vectoriales extendidas (SVE).

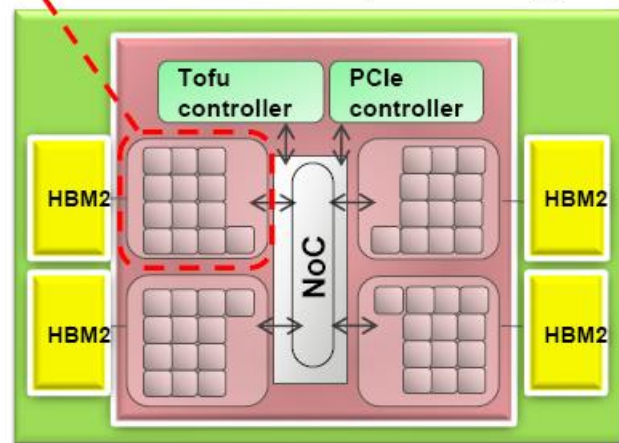
● Software:

- Rocky Linux 8.7
- OpenHPC
 - gnu12
 - Openmpi4
 - HPC-X
 - UCX
 - Python3.8.13
 - Papi, extrae, likwid...



Specifications	A64FX
ISA (Base, extension)	Armv8.2-A, SVE
Process technology	7 nm
Peak DP performance	2.7 or 3.0 TFLOPS
SIMD width	512-bit
# of cores	48 (+ 4)
Memory capacity	32 GiB (HBM2 x4)
Memory peak bandwidth	1024 GB/s
PCIe	Gen3 16 lanes
High speed interconnect	TofuD integrated

CMG 12x compute cores
1x assistant core (FX1000 only)



- GCC 11.2
- CMake 3.24.0
- Open MPI 4.1

```
C\__COMPILER=mpicc CXX\__COMPILER=mpic++ USE\_MPI=Yes pip install .
```

- Construido sobre el emulador opensource Qulacs.
- Implementación paralela
- Optimizaciones
- En desarrollo

MPI-Qulacs. Niveles de paralelismo

item	keywords	detail
1. Parallismo a nivel cluster	MPI	<ul style="list-style-type: none">- Separa y mantiene el vector de estado para soportar procesamiento distribuido.- Los datos deben ser intercambiados utilizando la interfáz de paso de mensajes. MPI
2. Paralelismo a nivel de procesador	OpenMP	<ul style="list-style-type: none">- Divide las tareas equitativamente entre cada core y las ejecuta- Para datos grandes, es más rápido fijar el área de memoria más cercana a los grupos de cores de inicio a fin.
3. Paralelismo a nivel de core	pipeline SIMD	<ul style="list-style-type: none">- CPU's recientes usan microcódigo y tuberías de operaciones paralelas.- Para maximizar la performance, los datos deben entrar continuamente en los pipelines.- Escribir en ensamblador, considerando los accesos a memoria, puede ayudar a llenar efectivamente esos pipelines.
4. Reduce inter-node communication	FusedSWAP	<ul style="list-style-type: none">- La Puerta fused SWAP recoloca datos del vector de estado en bloques para minimizer el coste de comunicación realizando la recolocación antes del cálculo.

Paralelismo a nivel de procesador

- El A64FX consiste en cuatro grupos de cores principales (CMG)
- Cada grupo tiene 12 cores, cache de nivel 2 y un controlador de memoria conectado a la memoria de alto ancho de banda (HBM)
- Para cálculos con alta demanda de memoria, es más rápido bloquear la memoria más cercana al CMG de inicio a fin

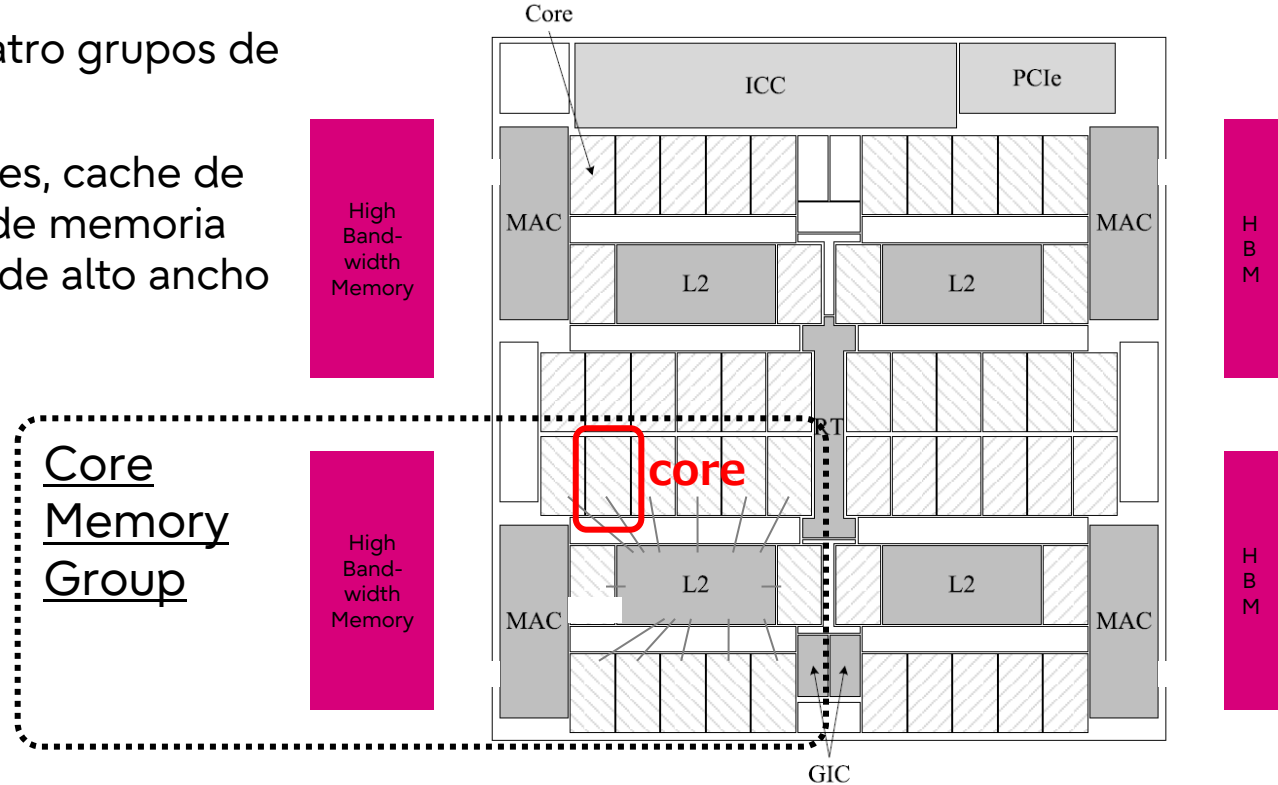
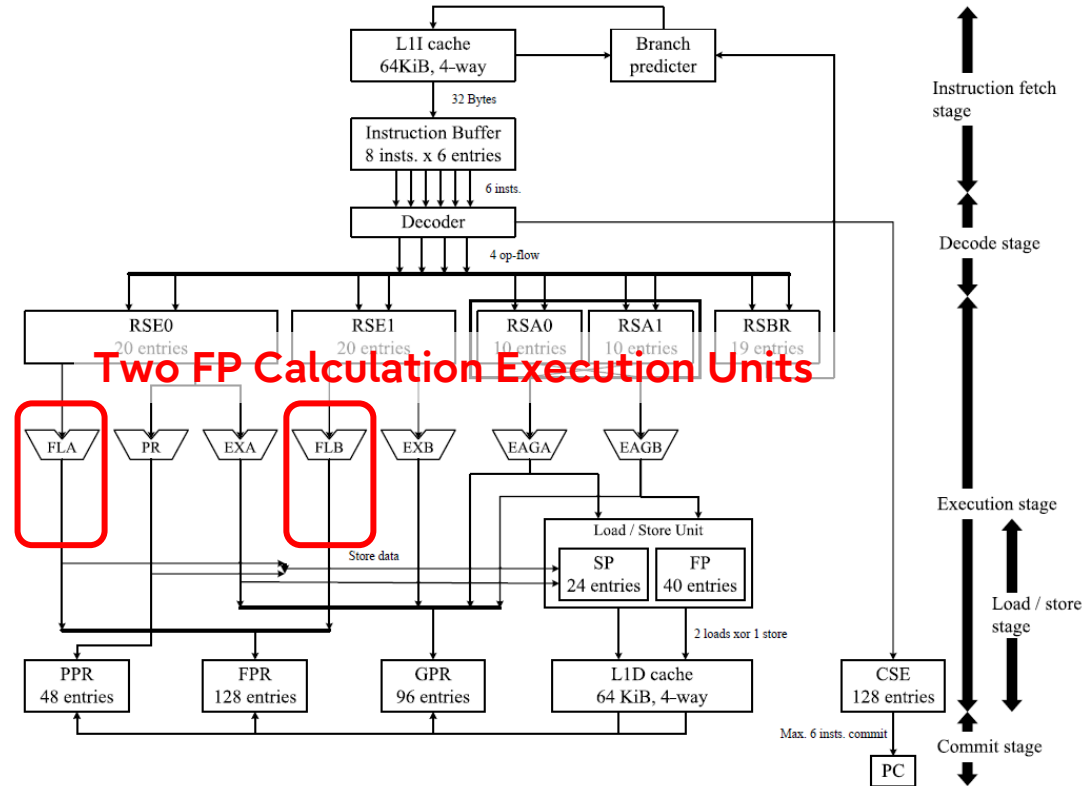


Figure 1-1 Main Functional Blocks on A64FX Processor Chip

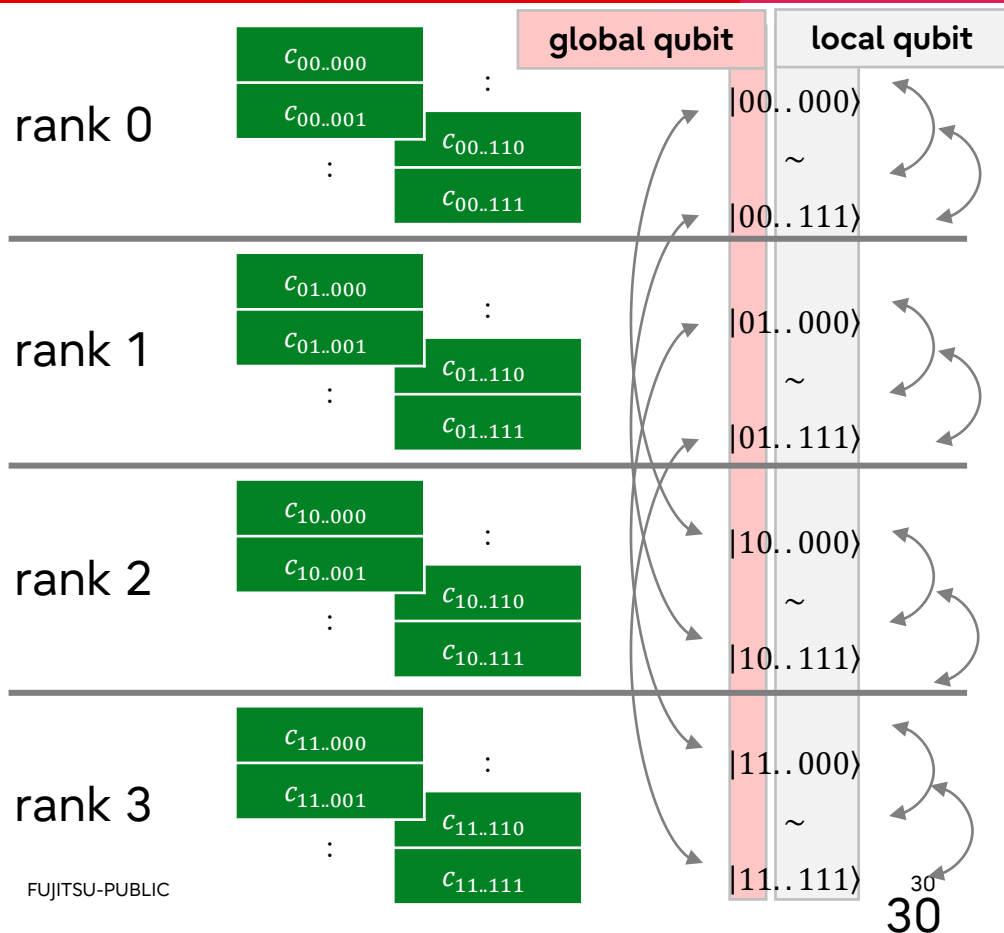
Paralelismo a nivel de core

- Cada core tiene dos unidades de pipeline como unidades de cálculo de punto flotante.
- El A64FX tiene dos pipelines que pueden realizar 8 operaciones de punto flotante (512 bit de largo) a la vez cada ciclo.
- Escribiendo en ensamblador con el conocimiento del acceso a memoria de la aplicación, es posible llenar esos pipelines de manera eficiente.



Processing stage diagram inside the core

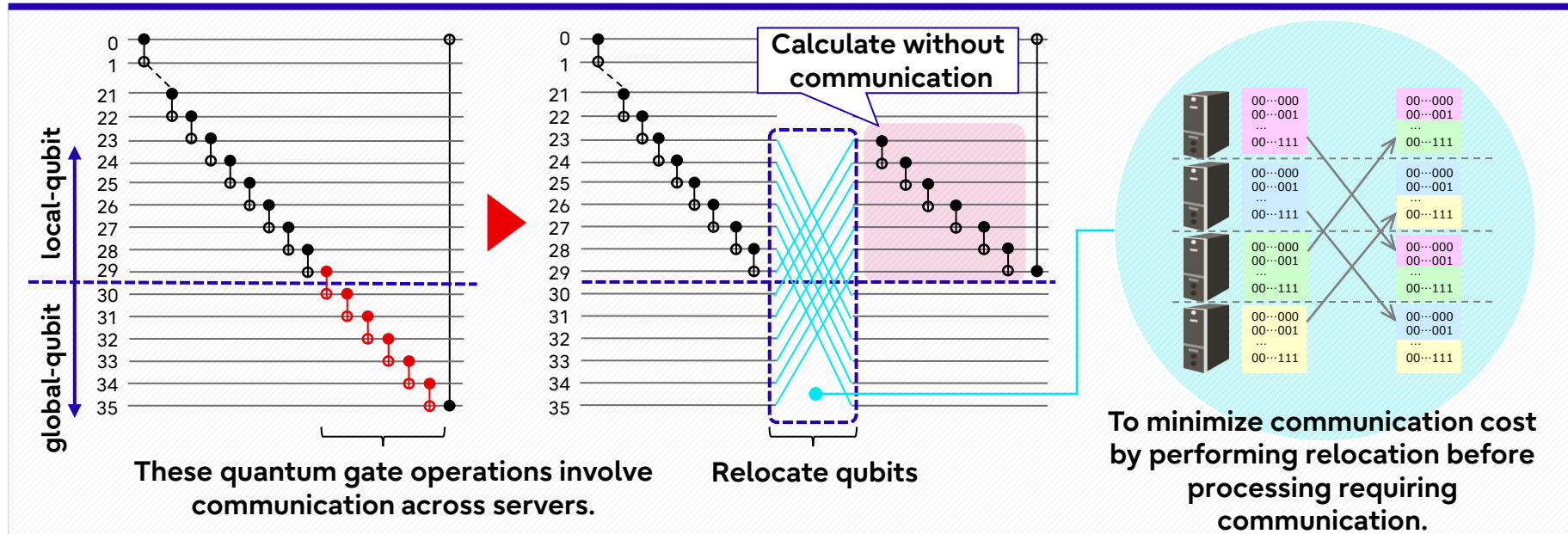
Vector de estado distribuido



- Divide y guarda el vector de estado para realizar el cálculo en paralelo
- El qubit the más abajo de cada rank es el **qubit local**. Los pares existen en el mismo rank.
- El de arriba coincide con el MPI-rank y es llamado **qubit global**.

Reducir la comunicación entre nodos

- La fused SWAP aplica una reordenación de los datos del vector de estad.
- Ejemplo: Operaciones CNOT con puertas de dos qubits
 - Las operaciones de puertas cuánticas a través de servidores causan comunicación cada vez
 - Recoloca los datos para reducir las comunicaciones durante las operaciones



Software Stack

- Árbol de módulos (Lmod)
- Diferentes para las diferentes arquitecturas

```

acaride@login03 /mnt/netapp2/Home_FT2/home/cesga/acaride/D0J0_20240513/py/quilacs
$ module av

----- /opt/cesga/qmio/hpc/software/Core/hpcx/2.17.1/modulefiles -----
hpcx-debug-ompi  hpcx-debug  hpcx-nt-ompi  hpcx-nt  hpcx-ompi  hpcx-prof-ompi  hpcx-prof  hpcx-stack  hpcx

----- Core -----
apptainer/1.2.3          (Tool)  libseccomp/2.5.5          (Tool)
bison/3.1                (Tool)  libxml2/2.9.7            (Tool)
boost.python/1.85.0-python-3.9.9  (Math)  libxslt/1.1.32          (Tool)
boost/1.85.0             (Tool)  m4/1.4.18                (Tool)
cmake/3.18.2             (Tool)  meson-python/0.16.0-python-3.9.9  (Tool)
cmake/3.25.0             (Tool)  meson/0.53.1             (Tool)
cmake/3.27.6             (D)     meson/1.4.0-python-3.9.9  (Tool,D)
cython/3.0.9-python-3.9.9  (Comp)  miniconda3/22.11.1-1     (Comp)
flex/2.6.4               (Tool)  ncurses/6.1              (Tool)
gcc/12.3.0               (Comp)  ninja/1.9.0              (Tool)
gcccore/12.3.0           (Comp)  numpy/1.26.4-python-3.9.9-mkl  (Math)
glib/2.58.2              (VisF)  numpy/1.26.4-python-3.9.9-openblas  (Math)
glpk/4.65                (Math)  numpy/1.26.4-python-3.9.9  (Math,D)
gmp/6.1.2                (Math)  openblas/0.3.24          (Tool)
go/1.20.4                (Comp)  openssl/1.1.0i           (Tool)
gocryptfs/2.4.0-linux-static_amd64  (Tool)  openssl/1.1.1b          (Tool)
gperf/3.1                (Tool)  openssl/1.1.1q           (Tool,D)
help2man/1.47.6          (Tool)  pixman/0.38.4            (VisF)
icu/75.1-python-3.9.9    (Math)  python/3.9.9             (Comp)
imkl/2023.2.0            (Math)  qmio-run/0.1.1-python-3.9.9  (QComp)
jupyter-server/1.13.5-python-3.9.9  (Tool)  rust/1.75.0              (Tool)
libffi/3.2.1             (Tool)  singularity/4.0.0        (Tool)
libffi/3.4.2             (Tool)  sqlite/3.45.3            (Tool)
libffi/3.4.4             (Tool,D)  squashfs/4.3             (Tool)
libreadline/8.0          (Tool)  squashfuse/0.5.0         (Tool)

----- /opt/cesga/modules/orgs-qmio -----
qmio/hpc (S,L)
    
```

```

$ module av
----- GCC/12.3.0 -----
boost.python/1.85.0-python-3.9.9 (D)      pytket/1.23.0-python-3.9.9      (QComp)      qulacs/0.6.3-python-3.9.9 (QComp,D)
boost/1.85.0 (Math,D)      qiskit-qulacs/0.1.0-python-3.9.9-mpi (QComp)      qutip/5.0.0-python-3.9.9
cvxpy/1.4.3-python-3.9.9      qiskit-qulacs/0.1.0-python-3.9.9 (QComp,D)      scipy/1.11.0-python-3.9.9 (Math)
networkx/2.8.8-python-3.9.9 (Tool)      qiskit/1.0.2-python-3.9.9-mpi (QComp)      scipy/1.13.0-python-3.9.9 (Math,D)
openblas/0.3.24 (D)      qiskit/1.0.2-python-3.9.9 (QComp,D)
pythran/0.15.0-python-3.9.9      qulacs/0.6.3-python-3.9.9-mpi (QComp)

----- Core -----
apptainer/1.2.3 (Tool)      matplotlib/3.5.3-python-3.9.9 (VisF)
binutils/2.40 (L,Tool)      meson-python/0.16.0-python-3.9.9 (D)
bison/3.1 (Tool)      meson-python/0.16.0-python-3.9.9
bison/3.1 (Tool,D)      meson/0.53.1 (Tool)
boost.python/1.85.0-python-3.9.9      meson/0.53.1 (Tool)
boost/1.85.0 (Math)      meson/0.63.3-python-3.9.9 (Tool)
catch2/2.13.9      meson/1.4.0-python-3.9.9 (Tool,D)
cmake/3.18.2 (Tool)      meson/1.4.0-python-3.9.9 (Tool)
cmake/3.18.2 (Tool)      miniconda3/22.11.1-1 (Comp)
cmake/3.25.0 (Tool)      mpc/1.3.1
cmake/3.25.0 (Tool)      mpfr/4.2.1
cmake/3.27.6      nasm/2.16.03
cmake/3.27.6 (D)      ncurses/6.1 (Tool)
conan/1.64.0-python-3.9.9      ncurses/6.1 (Tool,D)
conan/2.2.3-python-3.9.9 (D)      ninja/1.9.0 (Tool)
cython/0.29.24-python-3.9.9 (Comp)      ninja/1.9.0 (Tool,D)
cython/3.0.9-python-3.9.9 (Comp,D)      nlohmann_json/3.11.3
cython/3.0.9-python-3.9.9 (Comp)      nodejs/18.12.1-python-3.9.9-with-icu (Comp)
eigen/3.4.0 (Math)      numpy/1.26.4-python-3.9.9-mkl (Math)
flex/2.6.4 (Tool,D)      numpy/1.26.4-python-3.9.9-openblas (Math)
flex/2.6.4 (Tool)      numpy/1.26.4-python-3.9.9 (Math)
flint/3.1.2 (Math)      numpy/1.26.4-python-3.9.9 (Math,D)
gcc/12.3.0 (L,Comp)      openblas/0.3.24
gcccore/12.3.0 (L,Comp)      openssl/1.1.1.0i (Tool)
glib/2.58.2 (VisF)      openssl/1.1.1b (Tool)
glpk/4.65 (Math)      openssl/1.1.1q (Tool,D)
gmp/6.1.2 (Math)      pcre2/10.35
gmp/6.3.0 (D)      pixman/0.38.4 (VisF)
go/1.20.4 (Comp)      pybind11/2.8.1-python-3.9.9 (Tool)
gocryptfs/2.4.0-linux-static_amd64 (Tool)      pybind11/2.12.0-python-3.9.9 (Tool,D)
gperf/3.1 (Tool)      pylatexenc/2.10-python-3.9.9 (Tool)
graphviz/10.0.1      pyqt-builder/1.16.1-python-3.9.9
help2man/1.47.6 (Tool,D)      pytest/6.2.5-python-3.9.9
help2man/1.47.6 (Tool)      python/3.9.9-base (Comp)
icu/75.1-python-3.9.9      python/3.9.9 (Comp)
imkl/2023.2.0 (Math)      python/3.9.9 (Comp)
jupyter-bundle/20240425-python-3.9.9      python/3.11.9 (D)
jupyter-server/1.13.5-python-3.9.9      qmio-run/0.1.1-python-3.9.9 (QComp)

```

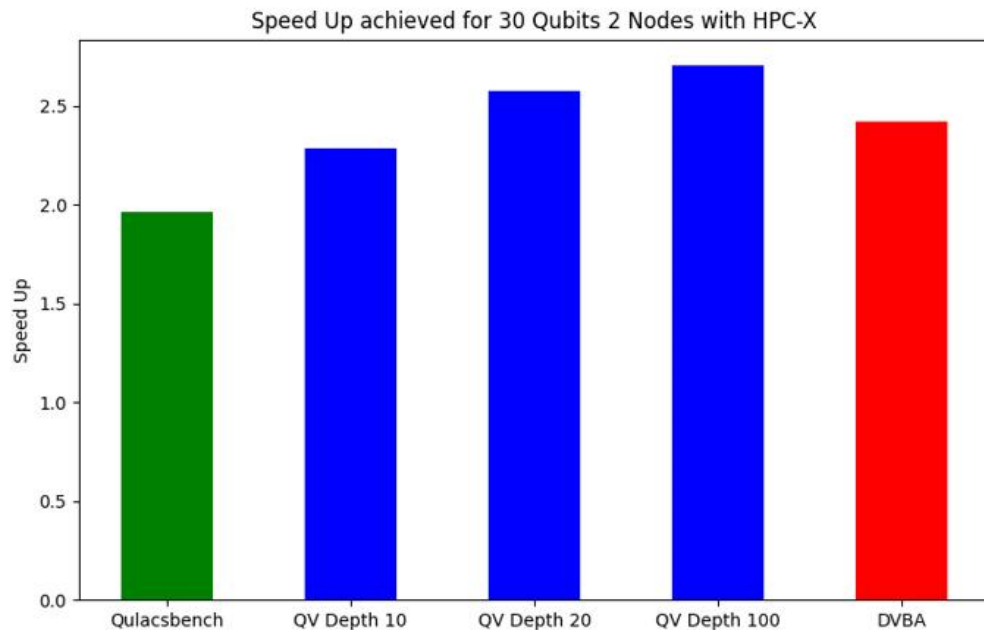
```
acaride@c7-101 ~/Exploracion-qulacs/ejemplos  
$ module av
```

```
----- /opt/ohpc/pub/moduledeps/gnu12-openmpi4 -----  
boost/1.80.0      extrae/3.8.3      py3-mpi4py/3.1.3  
  
----- /opt/ohpc/pub/moduledeps/gnu12 -----  
openblas/0.3.21   openmpi4/4.1.4 (L)   py3-numpy/1.19.5  
  
----- /opt/ohpc/pub/modulefiles -----  
EasyBuild/4.6.2   gnu12/12.2.0 (L)    libfabric/1.13.0 (L)  papi/6.0.0      prun/2.2      (L)    valgrind/3.19.0  
autotools        (L)    hpcx-mt-mpi        ohpc            (L)    pmix/4.2.1     qulacs-hpcx/1.0  
cmake/3.24.2      hwloc/2.7.0 (L)    os                 pmix/4.2.9 (D)    ucx/1.11.2     (L)
```

- Module spider
- Module av
- Module list
- Module purge (--force)
 - Source /etc/profile.d/lmod.sh

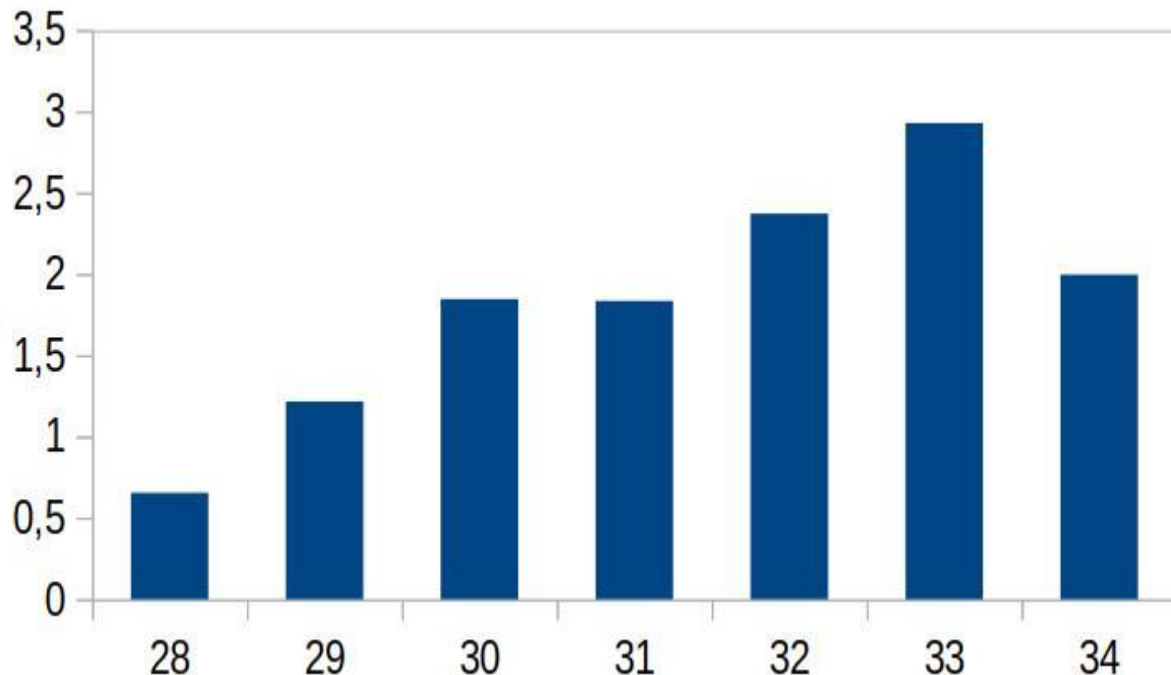
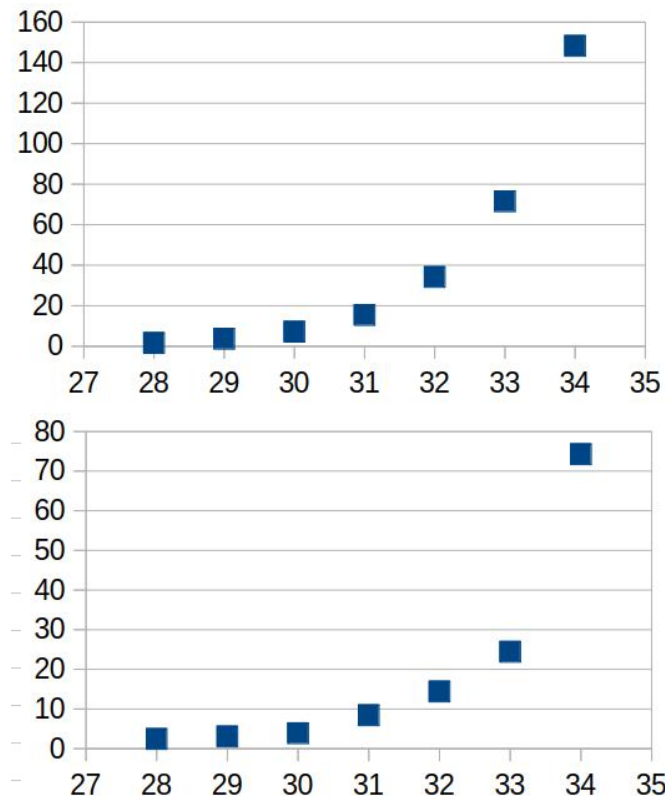
- En vivo en el cluster
- Diff DOJO/py/mpi-qulacs/base.sh DOJO/py/mpi-qulacs/base_x86.sh

● env-comparison.org



```
$ ucx_info -d | grep -i transport
# Transport: self
# Transport: tcp
# Transport: tcp
# Transport: tcp
# Transport: tcp
# Transport: tcp
# Transport: sysv
# Transport: posix
# Transport: dc_mlx5
# Transport: rc_verbs
# Transport: rc_mlx5
# Transport: ud_verbs
# Transport: ud_mlx5
# Transport: dc_mlx5
# Transport: rc_verbs
# Transport: rc_mlx5
# Transport: ud_verbs
# Transport: ud_mlx5
# Transport: cma
# Transport: knem
# Transport: xpmem
```

Weak scaling full opt – Sped ups



Ventajas y Limitaciones

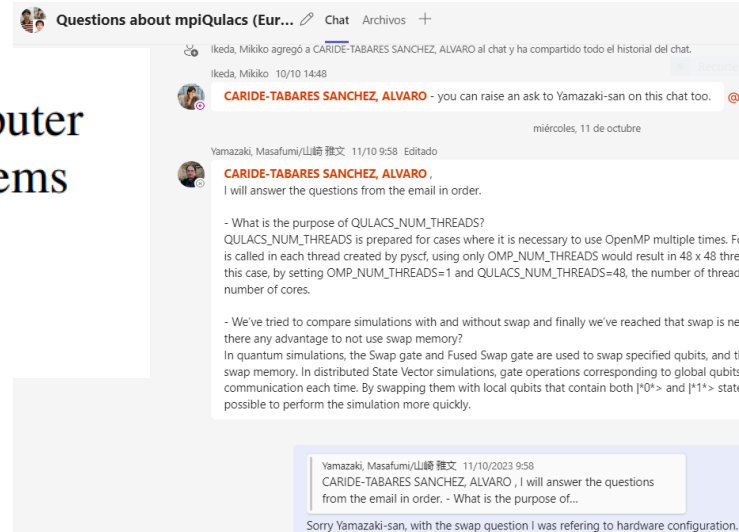
Disponibilidad

- Software instalado y funcionando en el cluster. Implementado para sacar partido de este hardware.
- Software en desarrollo activo. Contacto con el equipo de desarrollo.

mpiQulacs: A Distributed Quantum Computer Simulator for A64FX-based Cluster Systems

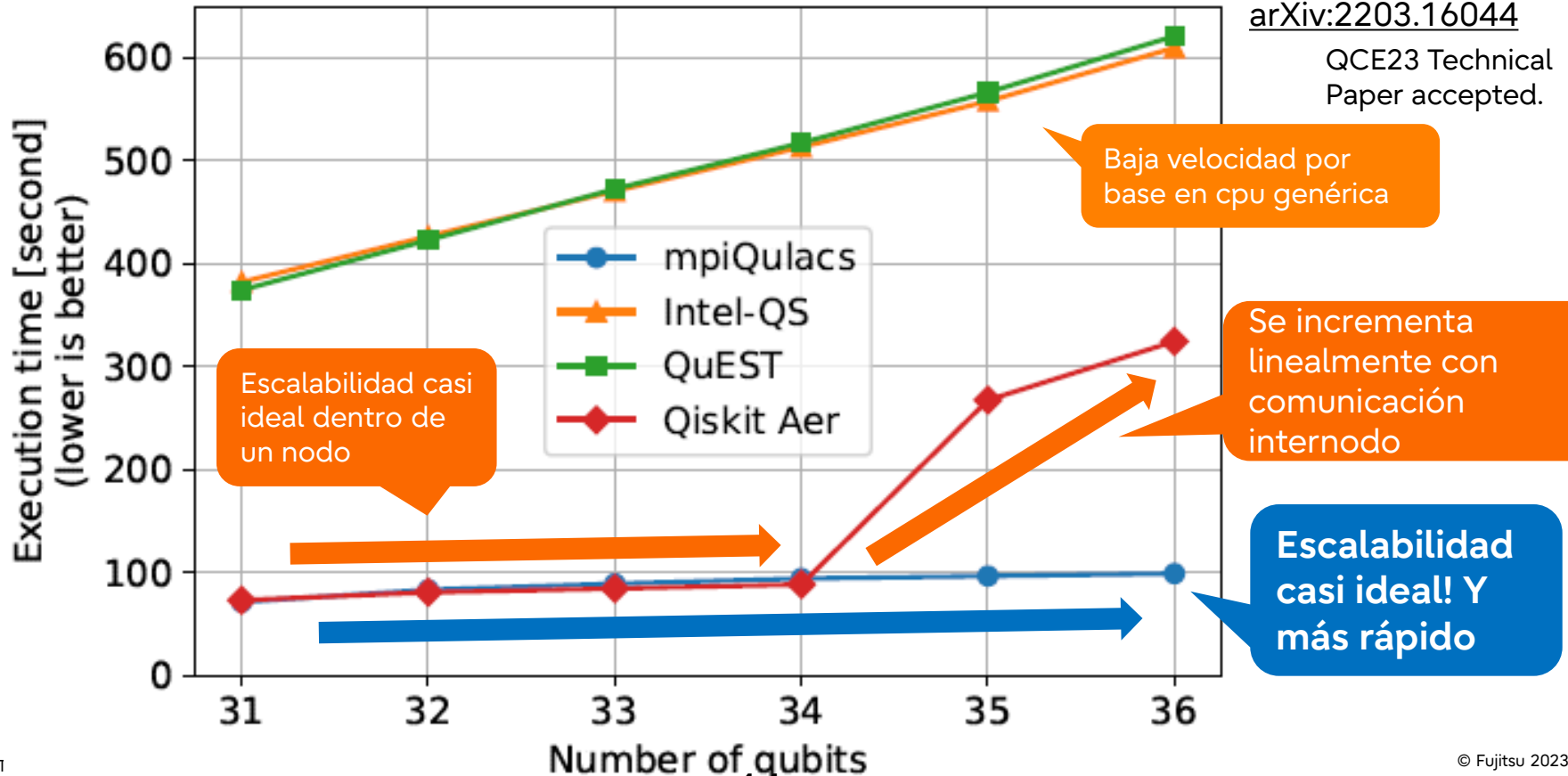
Satoshi Imamura, Masafumi Yamazaki, Takumi Honda, Akihiko Kasagi,
Akihiro Tabuchi, Hiroshi Nakao, Naoto Fukumoto, and Kohta Nakashima
*ICT Systems Laboratory
Fujitsu LTD.*

[2203.16044.pdf \(arxiv.org\)](#)



Rapidez

Quantum Software Benchmark



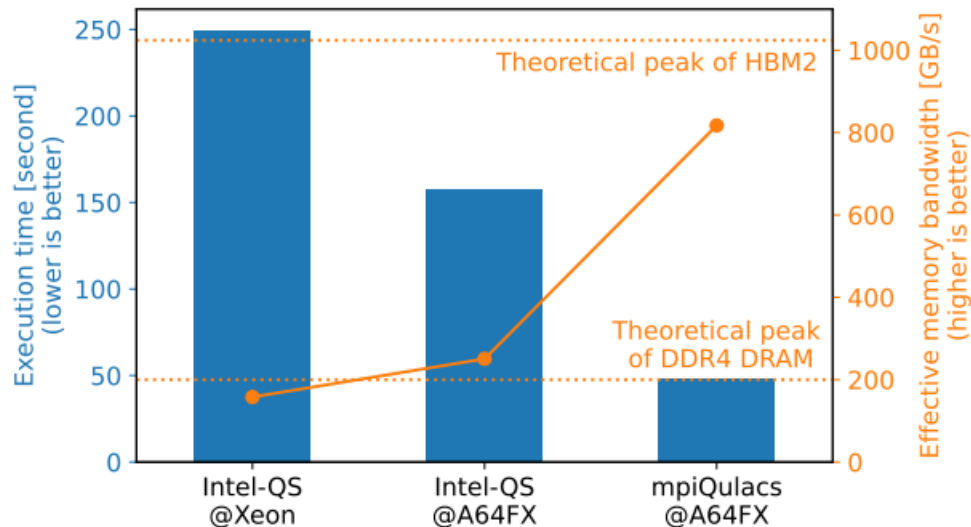
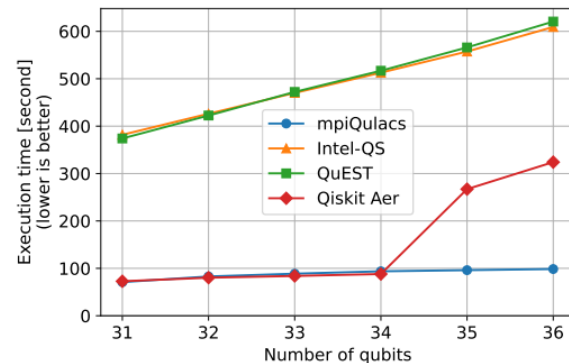
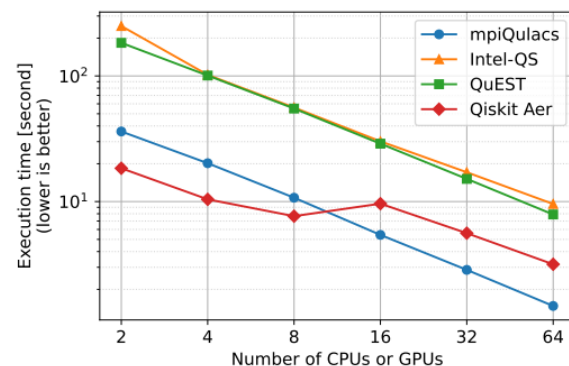


Fig. 12. The execution time of the 30-qubit Quantum Software Benchmark circuit and effective memory bandwidth.

[2203.16044.pdf \(arxiv.org\)](https://arxiv.org/abs/2203.16044)



(a) Weak scaling (30 qubits per CPU, 31 qubits per GPU)



(b) Strong scaling (fixed 30 qubits)

Fig. 10. The execution time of Quantum Software Benchmark circuit with mpiQulacs running on Todoroki and Intel-QS, QuEST, and Qiskit Aer running on ABCI.

Cómodidad

- Pensado para interactuar con MPI-Qulacs desde Python.
- La API para MPI no requiere de conocer todos los parámetros para comenzar a distribuir la ejecución.

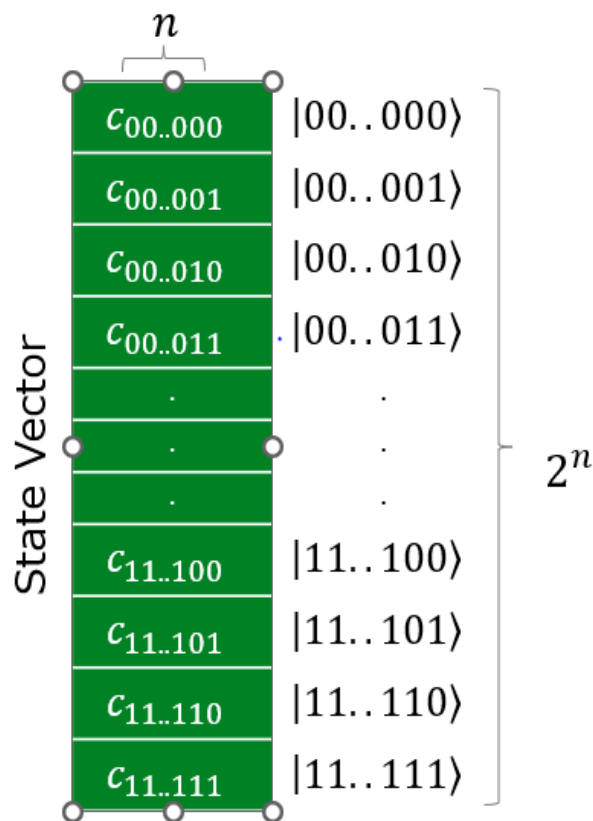
API for MPI [🔗](#)

- Instantiation of QuantumState
 - QuantumState state(qubits, use_multi_cpu)
 - use_multi_cpu = false
 - Generate state vector in a node (same as the original)
 - use_multi_cpu = true
 - Generate a state vector in multiple nodes if possible.
 - qubits are divided into local_qc + global_qc internally.
 - local_qc: qubits in one node
 - global_qc: qubits in multiple nodes (=log2(#rank))
 - state.get_device_name()
 - return the list of devices having the state vector.

ret value	explanation
"cpu"	state vector generated in a cpu
"multi-cpu"	state vector generated in multi cpu
"gpu"	state vector generated in a gpu

◦ state.to_string() Each rank outputs only state information that has itself

Contexto del cluter Híbrido



- Un estado de n qubits es base de un espacio de Hilbert 2^n dimensional.
- Si un número complejo es expresado como 16 bytes, la cantidad de memoria requerida es de 2^{n+4} [Byte]

$$|\psi\rangle = c_{000}|000\rangle + \dots + c_{111}|111\rangle$$

$n=30$

Gi Byte

Disponible en tu PC

$n=40$

Ti Byte

Foco de MPI-Qulacs

:

$n=170$

Byte

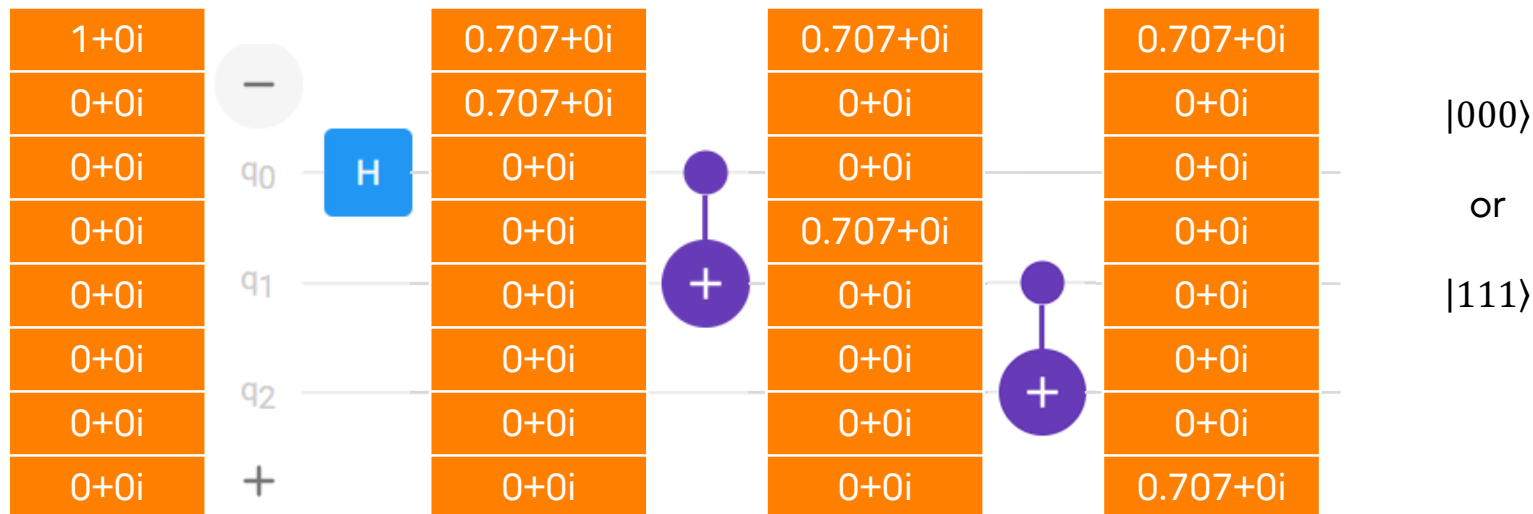
Excede cualquier expectativa

Flujo de procesamiento de la simulación de un circuito cuántico

Estado inicial
general

Aplicación de puertas/
Desarrollo temporal

Muestreo



Vuelta al primer lanzamiento

- En vivo en el cluster

- En vivo en el cluster

- En vivo en el cluster

- En vivo en el cluster

- Compilación en c++
 - MAKEFILE
- Ejecución de example qulacs y mpiqtest -1 25 5 6

Recolecta día 1

- ¿Qué commando utilizo para consultar las características de las colas?
- ¿Son los mismos módulos en todos los nodos?
- ¿Qué peculiaridades tienen los procesadores a64FX?

- Intro
- Primer Lanzamiento
- Arquitectura
- Software Stack
- Revisión del primer lanzamiento
- Resultados

- Intro
- Overview Qulacs base
- Qulacs y modelos de ruido
- Qulacs y circuitos paramétricos
- Implementación Paralela
- Migración a Qmio
- Resultados

Gracias por vuestra atención!

Álvaro Caride

alvaro.caride-tabataessanchez@fujitsu.com

Emulación Cuántica – MPI Qulacs

Día 2



FUJITSU



Fujitsu announces winners of the Fujitsu quantum simulator challenge

Global competition to accelerate research on advanced quantum technologies using Fujitsu's quantum simulator

Announcing the Fujitsu \$100,000 Quantum Simulator Challenge 2024

May 10, 2024

Japanese >

● Arquitectura

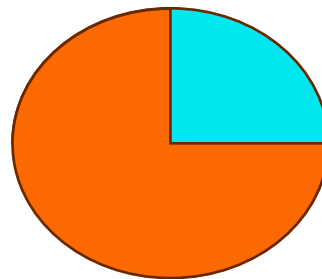
- Cluster Híbrido
- Overview instalación
- FX700
- MPI-Qulacs

● Características del Sistema

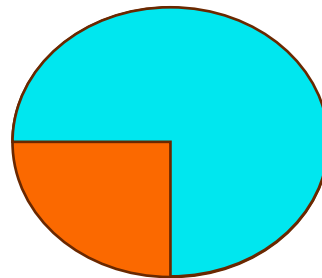
- Gestión de recursos
- Software Stack
- Ventajas y limitaciones

● ¿Cómo utilizar el Sistema?

- qulacs
- Implementación en paralela
- Hands-On (Dojo)
 - Base
 - Paralelización
 - Ejemplo
- Migración al QC

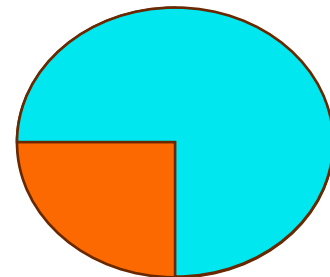


Día 1



Día 2

- Intro
- Overview Qulacs base
- Qulacs y modelos de ruido
- Qulacs y circuitos paramétricos
- Implementación Paralela
- Migración a Qmio
- Resultados



Recolecta día 1

● Slurm

- Particiones por arquitectura.
 - -p a64
 - Máximo número de cores por nodo 48
 - Máximo número de nodos 16
- Acceso a almacenamiento del CESGA
 - El directorio de lanzamiento debe ser accesible por slurm en todos los nodos del trabajo.
- Lanzar trabajo a las colas → sbatch
- Trabajos interactivos → compute/salloc ...

- Module av

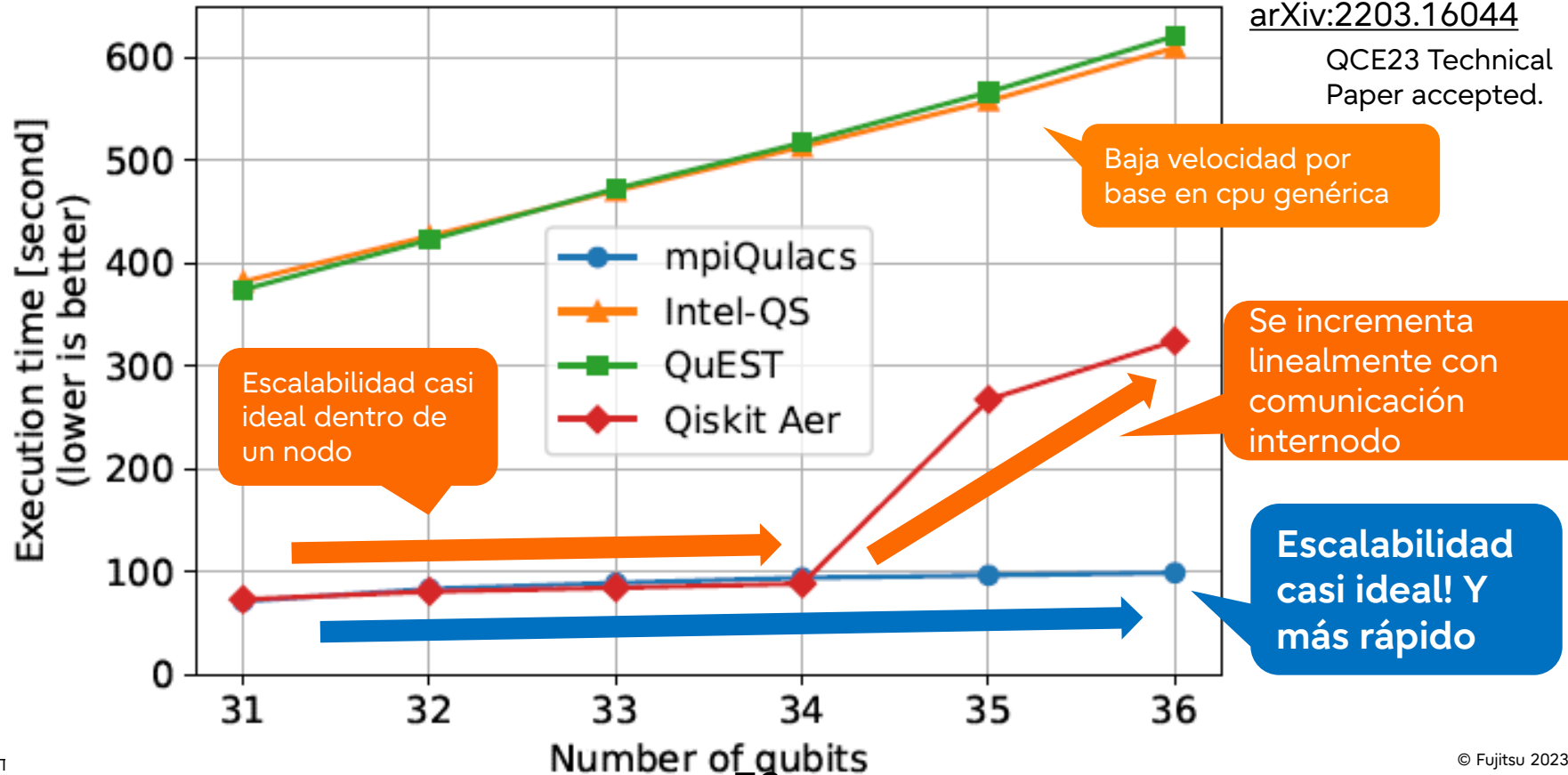
```
acaride@c7-101 ~/Exploracion-qulacs/ejemplos
$ module av

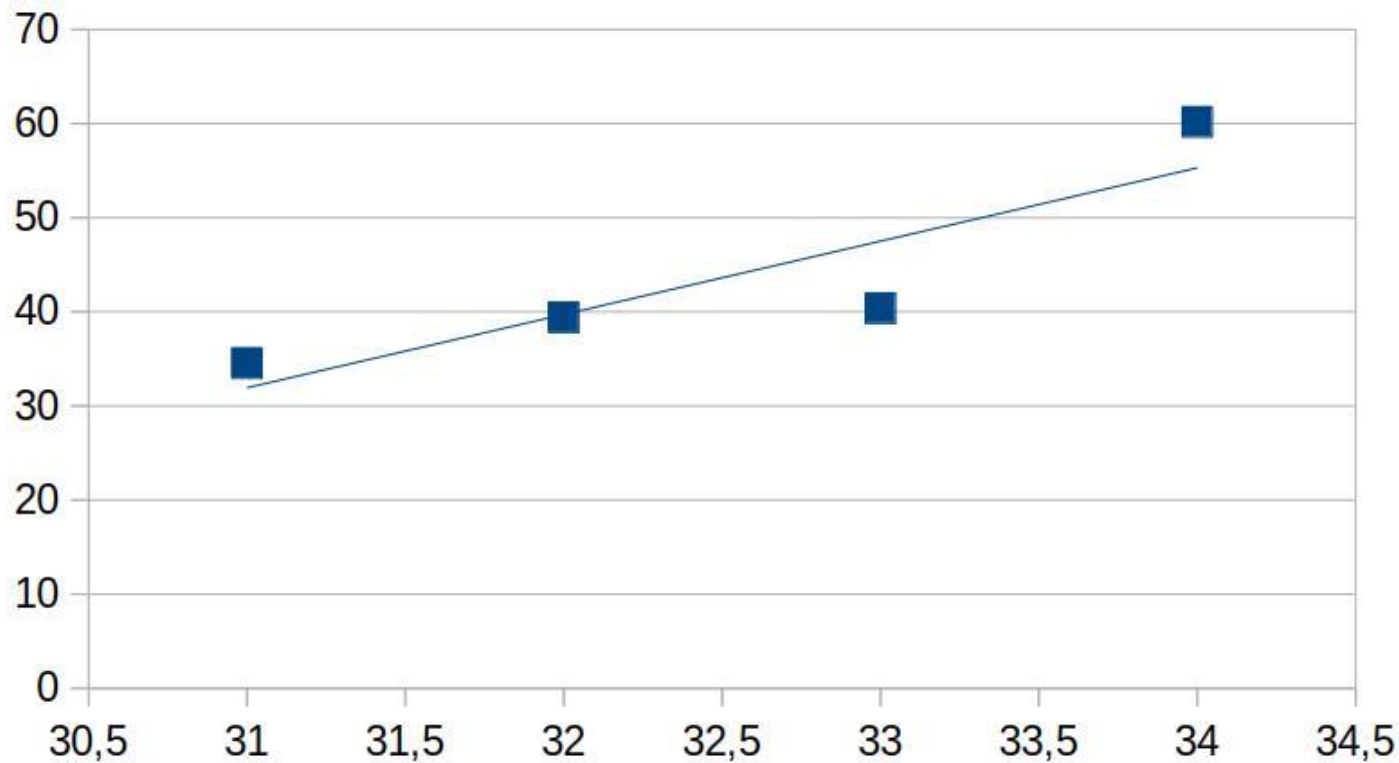
----- /opt/ohpc/pub/moduledeps/gnu12-openmpi4 -----
boost/1.80.0      extrae/3.8.3      py3-mpi4py/3.1.3
----- /opt/ohpc/pub/moduledeps/gnu12 -----
openblas/0.3.21   openmpi4/4.1.4 (L)      py3-numpy/1.19.5
----- /opt/ohpc/pub/modulefiles -----
EasyBuild/4.6.2   gnu12/12.2.0 (L)      libfabric/1.13.0 (L)  papi/6.0.0      prun/2.2 (L)      valgrind/3.19.0
autotools (L)     hpcx-mt-mpi          ohpc (L)      pmix/4.2.1      qulacs-hpcx/1.0
cmake/3.24.2      hwloc/2.7.0 (L)      os           pmix/4.2.9 (D)   ucx/1.11.2 (L)
```

- Module load qulacs-hpcx

- Interfaz en Python y en c++. Software pensado para interactuar con Python. La parte de c++ es más para el testeo de módulos internos
- Las librerías csim, cppsim y vqcsim no son separables.

Quantum Software Benchmark





- Compilación en c++
 - MAKEFILE
- Ejecución de example qulacs y mpiqtest -1 25 5 6

Overview Qulacs

- Crear un vector de estado de tu sistema
- Inicializar el vector de estado
- Recolectar información sobre ese estado
- Puertas
- Circuitos cuánticos
- Actualización del vector de estado

- Crear vector de estado
- Inicializarlo
- Recoger información
- Copiarlo y pegarlo
- Eliminarlo
- Aplicarle puertas
- Merge
- ...

*Revisar los scripts
dojo-1.py, dojo-
2.py y dojo-3.py en
la carpeta DEMO*

```
from mpi4py import MPI

from qulacs import QuantumState

from qulacs.state import inner_product

def main():
    # MPI initialization
    comm = MPI.COMM_WORLD
    size = comm.Get_size()
    rank = comm.Get_rank()
    #####
    # State operations
    n = 5
    state = QuantumState(n)

    #Zero State
    state.set_zero_state()
    print(state.get_vector())

    #initalize to |00101>
    state.set_computational_basis(0b00101)
    print(state.get_vector())

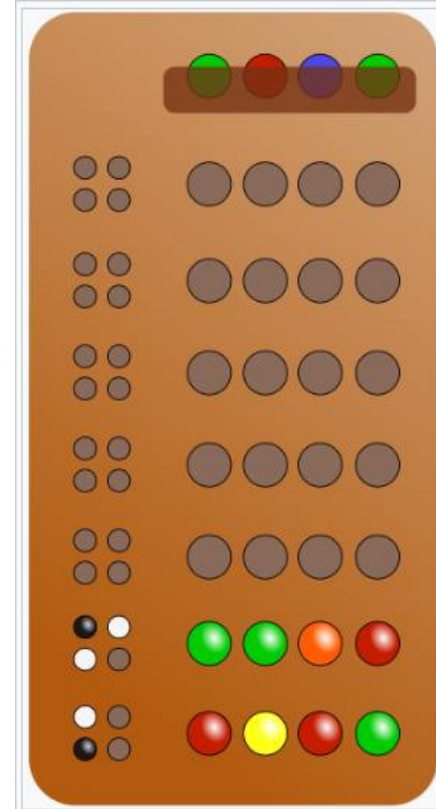
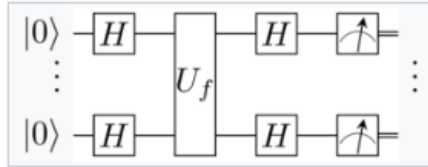
    # Random State
    state.set_Haar_random_state()
    print(state.get_vector())

    # Copy and load Data
    second_state = state.copy()
    print(second_state.get_vector())

    third_state = QuantumState(n)
    third_state.load(state)
```

- `QuantumState state(qubits, use_multi_cpu)`
- `State.get_device_name()` -> `cpu/multi-cpu`
- `State.set_Haar_random_state()`
- `Circuit.update_quantum_state(state)`
- `State.sample(number_sampling, [seed])`
 - `QULACS_NUM_THREADS`

- Algoritmo de Bernstein-Vazirani
 - Distribuido
 - Optimizaciones



- QuantumState state(qubits, use_multi_cpu=TRUE)

```
"""
Iniciación MPI
"""
mpicomm = MPI.COMM_WORLD
mpirank = mpicomm.Get_rank()
mpisize = mpicomm.Get_size()
globalqubits = int(np.log2(mpisize))

def add_oracle_to_circuit(circuit, s):
    """
    Añade las puertas del oráculo U_f al circuito basado en el número secreto s.
    """
    n = len(s)
    for i, bit in enumerate(s):
        if bit == "1":
            circuit.add_gate(CNOT(i, n))

def bernstein_vazirani(s):
    """
    Implementa el algoritmo de Bernstein-Vazirani para encontrar s.
    """
    n = len(s)
    global nqubits
    nqubits = n + 1
    #state = QuantumState(n + 1)
    """
    Distribución del quantum state
    """
    global state
    state = QuantumState(n + 1, use_multi_cpu = True)
    state.set_zero_state()
    circuit = QuantumCircuit(n + 1)
    # Inicialización: Aplica X y H al último qubit.
    circuit.add_gate(X(n))
```

● Implementación paralela de algoritmo de Bernstein-Vazirani

```
from qulacs import QuantumState, QuantumCircuit
from qulacs.gate import X, H, CNOT
```

```
import numpy as np
from mpi4py import MPI
```

```
mpicomm = MPI.COMM_WORLD
mpirank = mpicomm.Get_rank()
mpisize = mpicomm.Get_size()
globalqubits = int(np.log2(mpisize))
```

```
def add_oracle_to_circuit(circuit, s):
    """
    Añade las puertas del oráculo U_f al circuito basado en el número secreto s.
    """
    n = len(s)
```

```
+-- 3 lines: for i, bit in enumerate(s):-----
```

```
def bernstein_vazirani(s):
    """
    Implementa el algoritmo de Bernstein-Vazirani para encontrar s.
    """
    n = len(s)
    global nqubits
    nqubits = n +
    #state = QuantumState(n + 1)
```

```
global state
state = QuantumState(n + , use_multi_cpu = True)
state.set_zero_state()
circuit = QuantumCircuit(n + 1)
```

```
from qulacs import QuantumState, QuantumCircuit
from qulacs.gate import X, H, CNOT
```

```
def add_oracle_to_circuit(circuit, s):
    """
    Añade las puertas del oráculo U_f al circuito basado en el número secreto s.
    """
    n = len(s)
```

```
+-- 3 lines: for i, bit in enumerate(s):-----
```

```
def bernstein_vazirani(s):
    """
    Implementa el algoritmo de Bernstein-Vazirani para encontrar s.
    """
    n = len(s)
    state = QuantumState(n + 1)
```

```
circuit = QuantumCircuit(n + 1)
```

- git clone <https://gitlab.com/acaride/qulacs-dojo.git>

Ejercicio:
Conseguir el estado GHZ más grande que puedas en un máximo de 4 nodos.

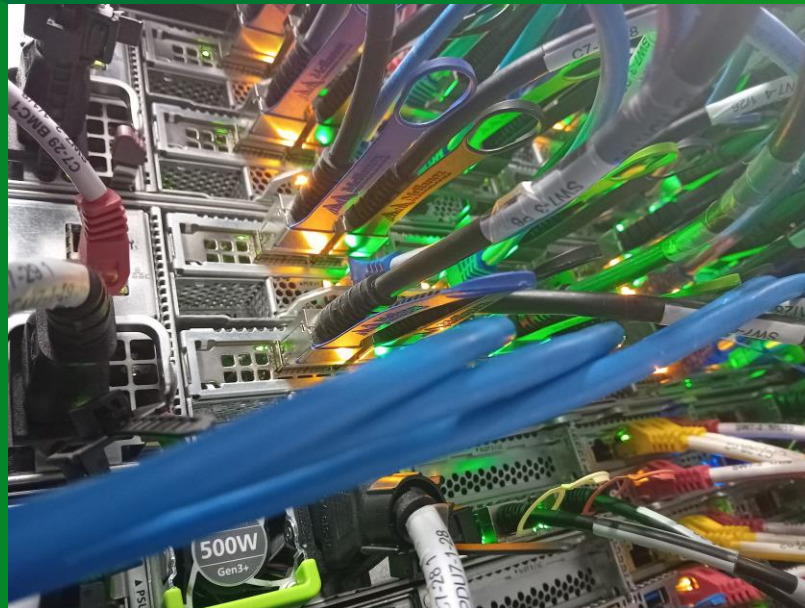
$$Score = \frac{\# \text{ Qubits}}{\text{Tiempo Ejecución}}$$

Optimizaciones para mayor rendimiento

OMP_NUM_THREADS=1
QULACS_NUM_THREADS=48

OMP_PROC_BIND=TRUE

Numactl -N 0-3



Ruido y Paramétricos

- SDAG. Capacidad de lanzar contra slurm workloads en forma de grafos directos acíclicos.
- En cada uno de los nodos del grafo puedes tener definida una parte del trabajo que se ejecuta en un hardware específico.
- SDAG se ocupa de establecer dependencias –afterok automáticamente en relación al grafo lanzado.
- El objetivo de esto es minimiza el tiempo IDLE de los recursos más limitantes del sistema.

Het-Workload - /apps/DEMO/het-Workload

```
acaride@login01 ~/qtest/submits/het-workflow
$ sdag 0-dag.txt
Job A ID: 3022
Parents:
Children: E,B

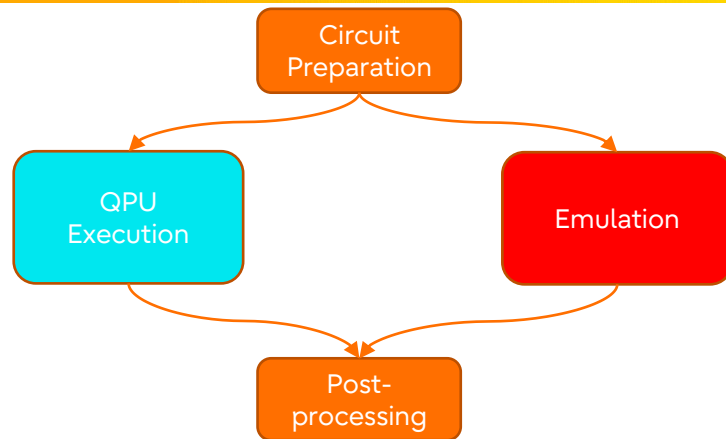
Job B ID: 3024
Parents: A
Children: C

Job C ID: 3025
Parents: B
Children: D

Job D ID: 3026
Parents: E,C
Children:

Job E ID: 3023
Parents: A
Children: D
```

- Hpc-nodes
- Emulation-nodes
- qpu



```
acaride@login01 ~/qtest/submits/het-workflow
$ squeue -u acaride
```

JOBID	USER	ACCON	NAME	PARTITION	QOS	FEATURES	ST	REASON	SUBMIT_TIME	START_TIME	END_TIME	TIME_LEFT	TIME_LIMIT	N
3024	acaride	proyec	2-compilation	hpc	normal	(null)	PD	Dependency	2023-10-23T12:16:36	N/A	N/A	5:00	5:00	
1	1	4294901677	0											
3026	acaride	proyec	4-post-proces	hpc	normal	(null)	PD	Dependency	2023-10-23T12:16:36	N/A	N/A	5:00	5:00	
1	1	4294901675	0											
3022	acaride	proyec	1-preparation	hpc	normal	(null)	R	None	2023-10-23T12:16:36	2023-10-23T12:16:37	2023-10-23T12:36:37	19:48	20:00	
1	64	4294901679	16											
3025	acaride	proyec	3-gpu-executi	qpu	normal	(null)	PD	Dependency	2023-10-23T12:16:36	N/A	N/A	5:00	5:00	
1	1	4294901676	0											
3023	acaride	proyec	5-emulation.s	qs	normal	(null)	PD	Dependency	2023-10-23T12:16:36	N/A	N/A	1-00:00:00	1-00:00:00	
1	48	4294901678	0											

- Tres lenguajes principales:
 - Qiskit (hpc)
 - Mpi-Qulacs (qs)
 - QAT-QASM (qpu)
- Existen métodos de autotraducción desde y hasta QASM en qiskit y qulacs.
- Planteamiento: utilizar QASM como lenguaje universal.
- Después de la autotraducción, revisar siempre si el algoritmo de salida se corresponde con el de entrada.
- Todos los sistemas reciben trabajos desde slurm.
- Piedra Roseta. Ojo!

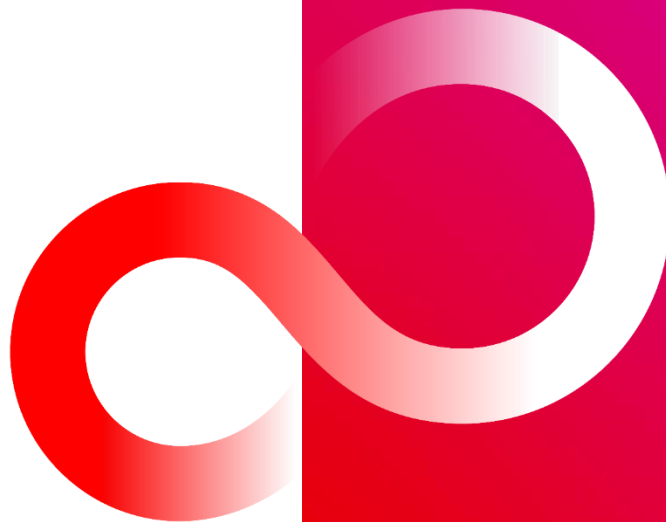
Referencias

- Qulacs Docs – [Qulacs Documentation — Qulacs documentation](#)
- Quantum Native Dojo – [Welcome to Quantum Native Dojo! — Quantum Native Dojo documentation \(qulacs.org\)](#)
- Git repo MPI – [qulacs/README_MPI.md at main · qulacs/qulacs · GitHub](#)
- Qulacs Gui – [Qulacs Simulator \(qulacs-gui.github.io\)](#)
- Artículo citado – [\[2011.13524\] Qulacs: a fast and versatile quantum circuit simulator for research purpose \(arxiv.org\)](#)
- https://qce.quantum.ieee.org/2023/wp-content/uploads/sites/7/2023/08/QALG_QAPP_QSYS_QNET_QTEM_conference_agenda_0830-v67.pdf p.18 EC251

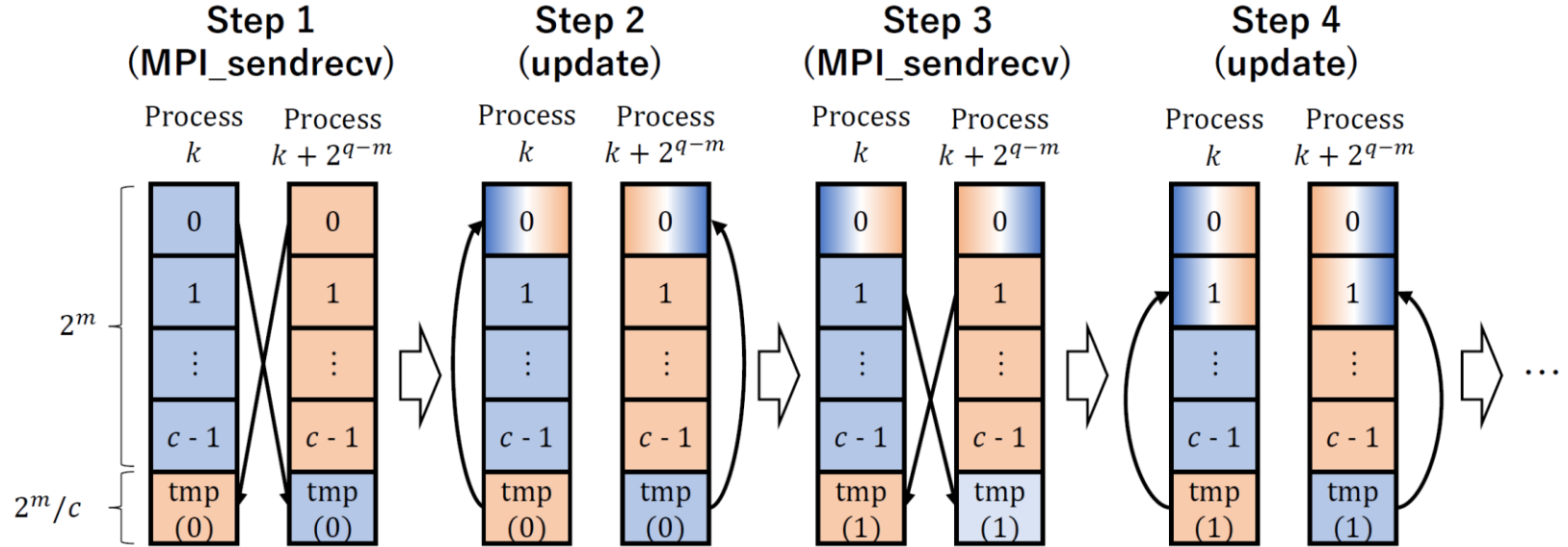
¿Preguntas?



Thank you



Gate processing that requires data transfer between nodes using MPI



- Exchange and update probability amplitudes between two paired processes
- Message Passing Interface(MPI) is a standardized message passing standard for parallel computing architectures.