# HPC simulation of quantum Computer

Nobuyasu Ito and Naoki Yoshioka
RIKEN Center for Computational Science

braket: A open-source quantum computer simulator for parallel computer
C++
https://github.com/naoki-yoshioka/braket

**RIKEN R-CCS and supercomputers**

K 10PFLOPS, 2011-2019


→

Fugaku 442PFLOPS, 2020-

→

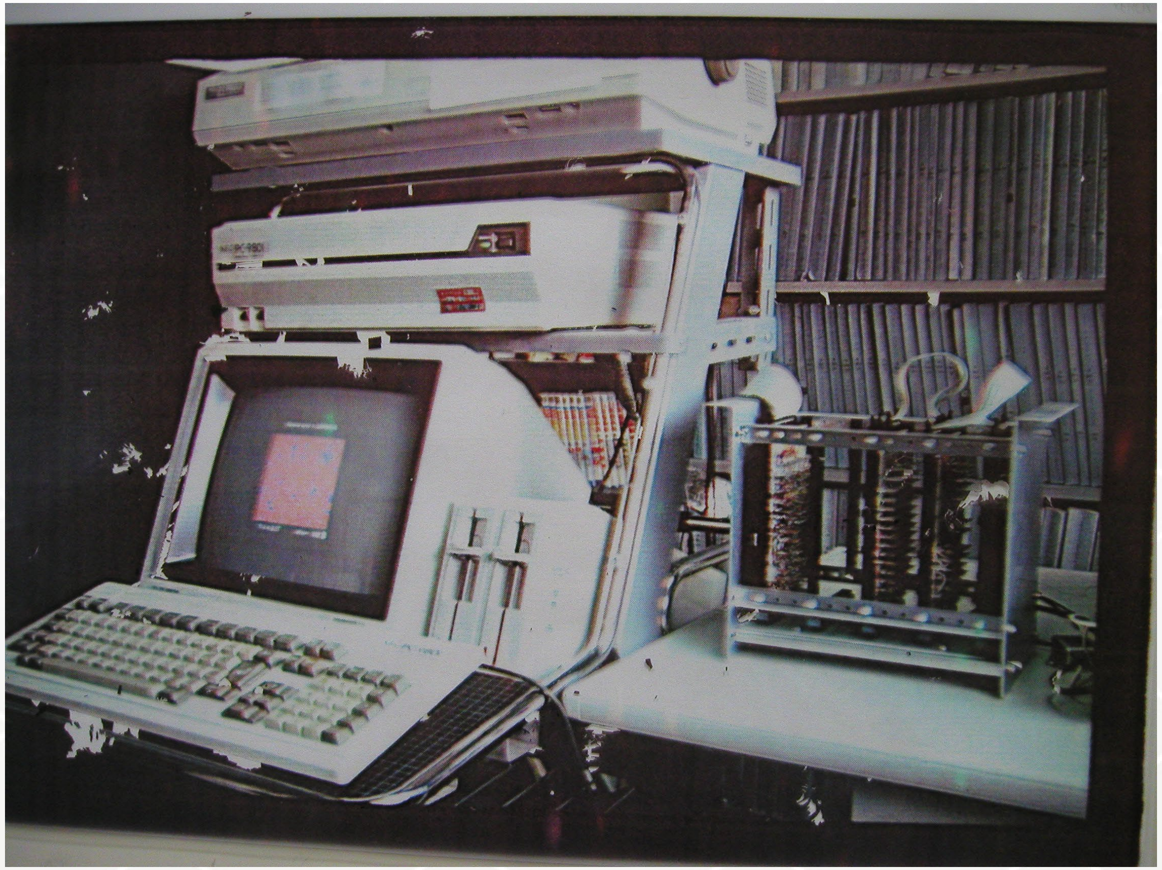
**Fugaku NEXT ～2030?**

with quantum-hybrid architecture?

# ISING MACHINE m-TIS

## - first Ising annealer in Japan, second after A. T. Ogielski in 1985 at Bell Lab.

N. Ito, M. Taiji, M. Suzuki, R. Ishibashi, K. Kobayashi, K. Mitsubo and S. Katsura,
in "Computer Simulation Studies in Condensed Matter Physics III",
edited by D. P. Landau, K. K. Mon and H.-B. Schuettler (Springer--Verlag, 1991) (1991) p.201-203.
M. Taiji, N. Ito and M. Suzuki, Rev. Sci. Intrum. 59 (1988) p.2483
N. Ito, M. Taiji and M. Suzuki, in "Computational Approaches in Condensed Matter Physics",
edited by S. Miyashita, M. Imada and H. Takayama (Springer-Verlag, 1992) (1992) p.297-298.



全学講演会の記録
No.4

R. P. ファインマン

# 未 来 の 計 算 機

1985年 8 月 9 日講演

学 習 院 大 学

# The Computing Machines in the Future

Richard P. Feynman

California Institute of Technology
Pasadena, California, U.S.A.

It's a great pleasure and an honor to be here as a speaker in memorial for a scientist that I have respected and admired as much as Prof. Nishina. To come to Japan and talk about computers is like giving a sermon to Buddha. But I have been thinking about computers and this is the only subject I could think of when invited to talk.

The first thing I would like to say is what I am not going to talk about. I want to talk about the future computing machines. But the most important possible developments in the future, are things that I will not speak about. For example, there is a great deal of work to try to develop smarter machines, machines which have a better relationship with the humans so that input and output can be made with less effort than the complex programing that's necessary today. This goes under the name often of artificial intelligence, but I don't like that name. Perhaps the unintelligent machines can do even better than the intelligent ones. Another problem is the standardization of programing languages. There are too many languages today, and it would be a good idea to choose just one. (I hesitate to mention that in Japan, for what will happen will be that there will simply be more standard languages; you already have four ways of writing now and

---

Nishina Memorial Lecture, August 9, 1985,
Nishina Foundation and Gakushuin.

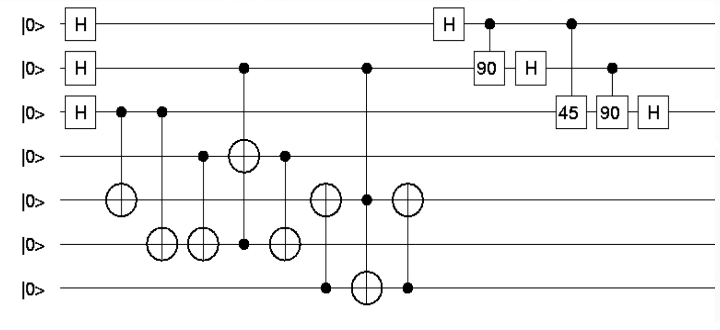
# preliminary challenges

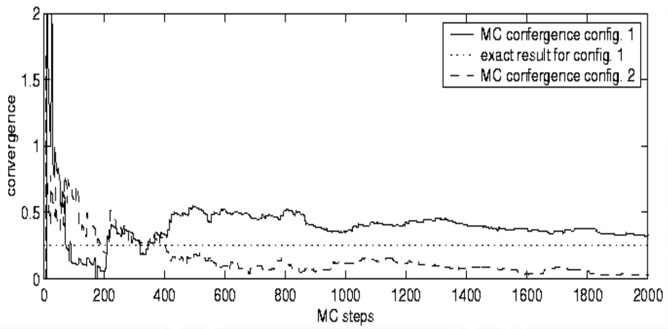
## ◎Application of quantum Monte Carlo simulation

### Hubbard-Stratnovich transformation and multicanonical sampling

H.G. Matuttis et al, Intern. J. Modern Phys. C vol. 13 No. 7 (2002) p.917—929.
K. Fischer et al, Intern. J. Modern Phys. C vol. 13 No. 7 (2002) p.931—945.

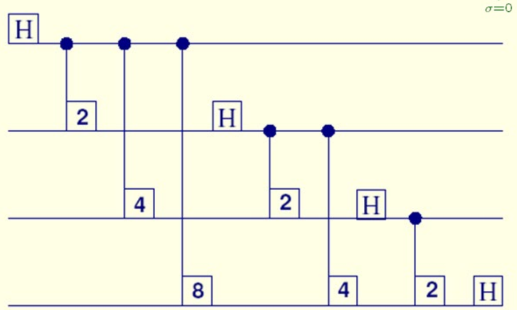
Quantum Circuit from L.M.K. Vandersypen et. al., Nature 414, p.883 (2001): Shor-like circuit with 4 qubits



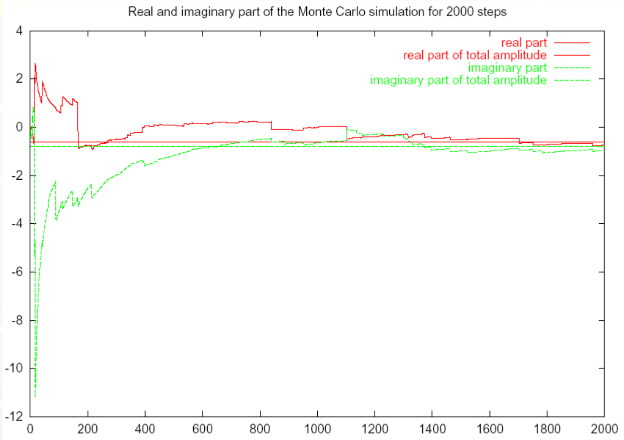

Fourier transformation

$$R_q = \exp\left[\frac{2\pi i}{2^q} n_1 n_2\right] = \underbrace{1}_{\sigma=0} + \underbrace{\left[\exp\left(\frac{2\pi i}{2^q}\right) - 1\right] n_1 n_2}_{\sigma=1}$$

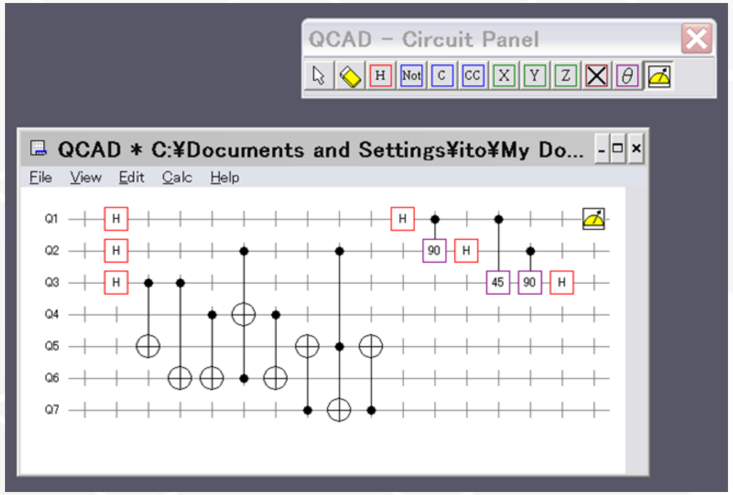



For $q = 2, 4, 8$ controlled phase gates, $\exp\left[\frac{2\pi i}{2^q} n_1 n_2\right]$ acting on bits $1, 2$ decomposed with $2^{15} = 32768$ configurations

8bits FT




## ◎state-vector simulator with GUI: QCAD



Fugaku

Available online at www.sciencedirect.com
ScienceDirect
Computer Physics Communications 176 (2007) 121–136
Computer Physics Communications
www.elsevier.com/locate/cpc

# Massively parallel quantum computer simulator

K. De Raedt [a], K. Michielsen [b], H. De Raedt [b,*], B. Trieu [c], G. Arnold [c], M. Richter [c], Th. Lippert [c], H. Watanabe [d], N. Ito [e]

[a] *Department of Computer Science, University of Groningen, Blauwborgje 3, NL-9747 AC Groningen, The Netherlands*
[b] *Department of Applied Physics, Materials Science Centre, University of Groningen, Nijenborgh 4, NL-9747 AG Groningen, The Netherlands*
[c] *Zentralinstitut für Angewandte Mathematik, Forschungszentrum Jülich, D-52425 Jülich, Germany*
[d] *Department of Complex Systems Science, Graduate School of Information Science, Nagoya University, Furouchou, Chikusaku, Nagoya 464-8601, Japan*
[e] *Department of Applied Physics, School of Engineering, The University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113-8656, Japan*

Received 21 February 2006; accepted 24 August 2006
Available online 13 October 2006

*up to 36 qbits*

Contents lists available at ScienceDirect
Computer Physics Communications
journal homepage: www.elsevier.com/locate/cpc

# Massively parallel quantum computer simulator, eleven years later

Hans De Raedt [a], Fengping Jin [b], Dennis Willsch [b,c], Madita Willsch [b,c], Naoki Yoshioka [d], Nobuyasu Ito [d,e], Shengjun Yuan [f,*], Kristel Michielsen [b,c,**]

[a] *Zernike Institute for Advanced Materials, University of Groningen, Nijenborgh 4, NL-9747 AG Groningen, The Netherlands*
[b] *Institute for Advanced Simulation, Jülich Supercomputing Center, Forschungszentrum Jülich, D-52425 Jülich, Germany*
[c] *RWTH Aachen University, D-52056 Aachen, Germany*
[d] *RIKEN Center for Computational Science, 7-1-26 Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo 650-0047, Japan*
[e] *Department of Applied Physics, School of Engineering, The University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113-8656, Japan*
[f] *School of Physics and Technology, Wuhan University, Wuhan 430072, China*

*up to 48 qbits*

RIKEN Code: https://github.com/naoki-yoshioka/braket

# An example

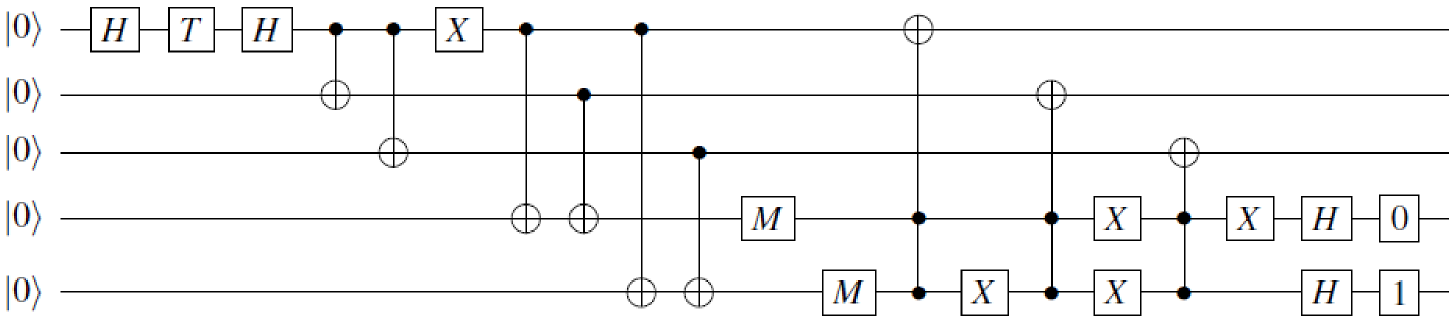



Figure B.6: Quantum circuit performing error correction on the top three qubits. The corresponding JUQCS input file is listed in Example: input. Qubits are numbered from zero (top) to four (bottom). Reading from left to right, the first three gates prepare the initial state, the next two (CNOT) gates perform the encoding, the $X$ gate on qubit 0 introduces a spin flip error, the next 11 gates detect and correct the error and the last 3 (2) gates on qubit 3 (4) illustrate how to reset a qubit to 0 (1).

## Program: Quantum assembler – input to simulators

```
QUBITS          5
! DEPOLARIZING CHANNEL p_X =0.01, p_Y =0.01

H 0 ! initial state
T 0
H 0 ! initial state

CNOT 0 1 ! encode
CNOT 0 2
BEGIN MEASUREMENT

!x 1 ! error
x 0 ! error

CNOT 0 3 ! correct
CNOT 1 3
CNOT 0 4
CNOT 2 4
M 3
M 4
TOFFOLI 3 4 0
X 4
TOFFOLI 3 4 1
X 3
X 4
TOFFOLI 3 4 2
X 3

H 3
CLEAR 3  ! must be preceeded by Hadamard
H 4
SET 4 ! must be preceeded by Hadamard

BEGIN MEASUREMENT
GENERATE EVENTS 8192 −1
```

# Some basic aspects on quantum computer

- $|0\rangle$ or $|1\rangle$: orthogonal basis vectors
- $n_L n_{L-1} \cdots n_2 n_1$: a binary representation for $n$
- State of an $L$-qubit quantum computer

$$|\Phi\rangle = \sum_{n=0}^{2^L-1} a_n |n\rangle ,$$

where $|n\rangle = |n_L\rangle |n_{L-1}\rangle \cdots |n_2\rangle |n_1\rangle$,

$$\sum_{n=0}^{2^L-1} |a_n|^2 = 1, \ \langle\Phi|\Phi\rangle = 1$$

- $|a_n|^2$: probability of obatining $n$ by repeated measurements

# Requirements for QC simulation

Fugaku

RIKEN R-CCS

Memory limits number of qbits.

| N | $2^N$ | double precision $\sim10^{16}$ op. (over-spec) | single precision $\sim10^{8}$ op. | half precision $\sim10^{4}$ op. (practical use) | byte precision $\sim10^{2}$ op. (simple test) | |
|---|---|---|---|---|---|---|
| 10 | 1K | 16KB | 8KB | 4KB | 2KB | |
| 20 | 1M | 16MB | 8MB | 4MB | 2MB | |
| 30 | 1G | 16GB | 8GB | 4GB | 2GB | |
| 36 | 16G | 1TB | 512GB | 256GB | 128GB | ← 2007 on Blue Gene/L and Regatta p69 |
| 40 | 1T | 16TB | 8TB | 4TB | 2TB | |
| 45 | 32T | 512TB | 256TB | 128TB | 64TB | ← 2019 on K and Taihu Light |
| 46 | 64T | 1PB | 512TB | 256TB | 128TB | |
| 47 | 128T | 2PB | 1PB | 512TB | 256TB | ← on Fugaku with two-to-N allocation |
| 48 | 256T | 4PB | 2PB | 1PB | 512TB | ← on Fugaku with packed allocation |
| 49 | 512T | 8PB | 4PB | 2PB | 1PB | |
| 50 | 1P | 16PB | 8PB | 4PB | 2PB | |
| 51 | 2P | 32PB | 16PB | 8PB | 4PB | |
| 52 | 4P | 64PB | 32PB | 16PB | 8PB | |
| … | | | | | | |

→ Memory transfer rate limits number of qbits.

| | per node | | full system | |
|---|---|---|---|---|
| | in-node | inter-node | in-node | inter-node |
| K: | 64GB/s | 40GB/s | 5.4PB/s | 3.4PB/s |
| Fugaku: | 1TB/s | 40.8GB/s | 155.3PB/s | 6.2PB/s |

富岳 Fugaku

RIKEN R-CCS

CPU A64FX 7nm FinFET
Architecture Armv8.2-A SVE(512bit SIMD)
**48 cores for compute and 4 for OS activities**
Normal 2.0GHz (DP:3.072TF, SP: 6.144TF, HP:12.288TF)
Boost 2.2GHz(DP:3.3792TF, SP: 6.7584TF, HP:13.5168TF)
Cache L1: 64KB 4way >230GB/s(load) >115GB/s(store)
L2: 8MB 16way >115GB/s(load) >57GB/s(store)
**Memory HBM2 8GB X 4=32GiB 1024GB/s**
I/O PICe Gen3 x 16 lane

Performance: Stream triad 830GB/s, Dgemm 2.5TF

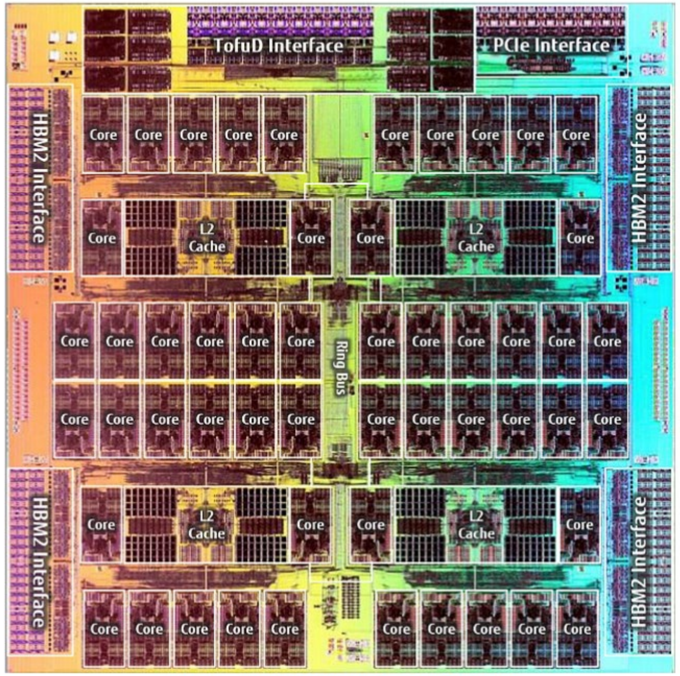
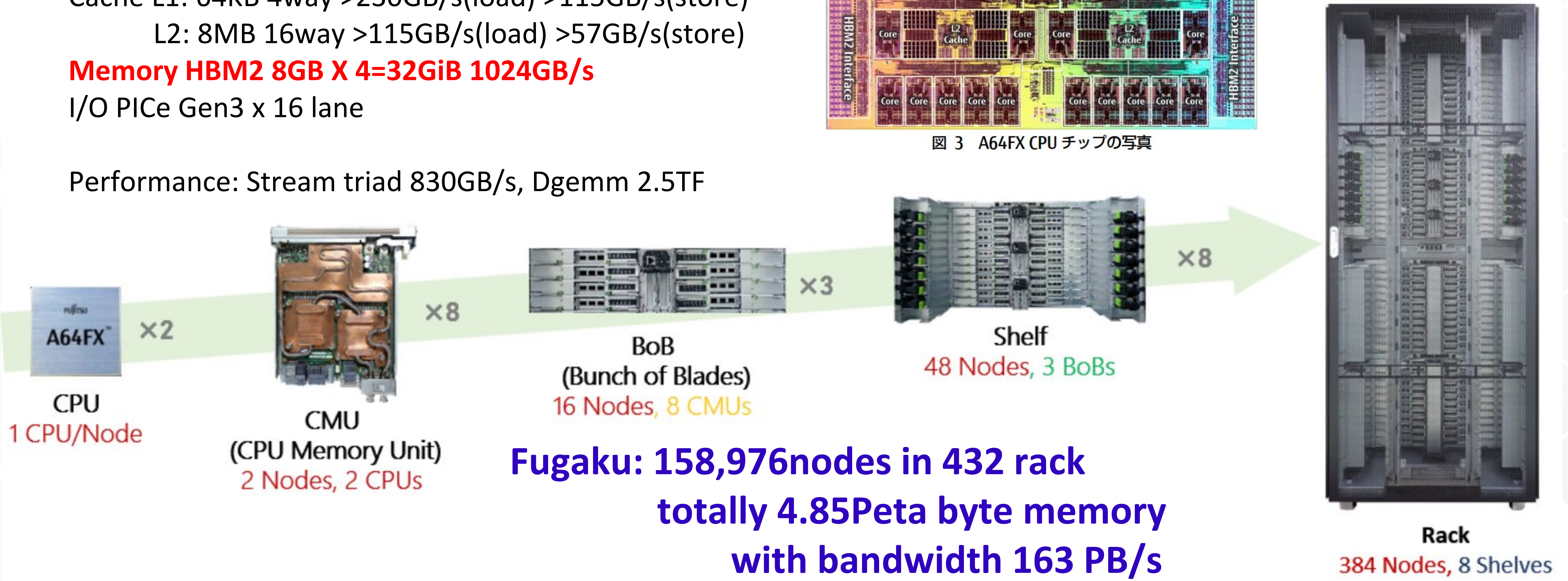


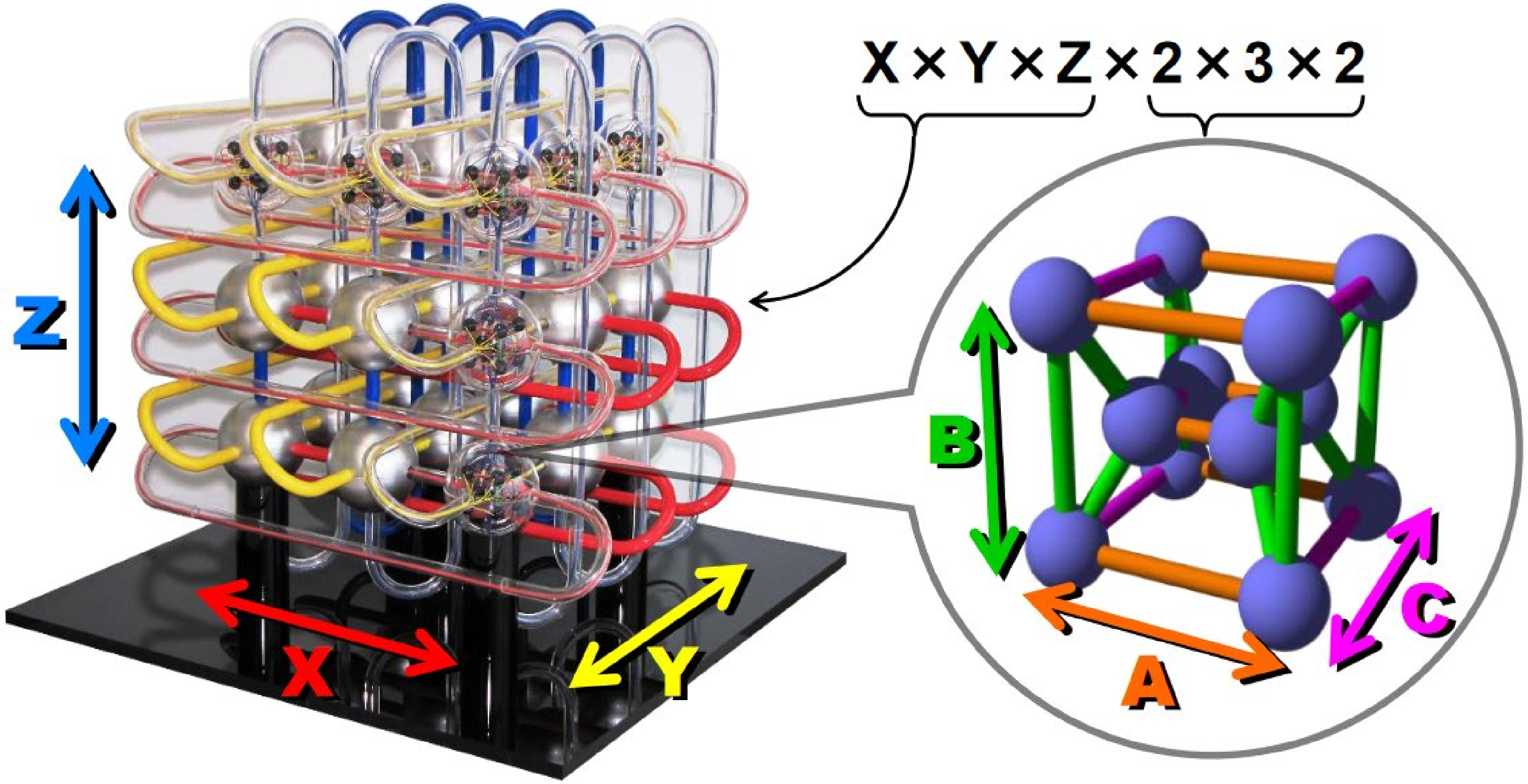
図 3 A64FX CPU チップの写真




**Fugaku: 158,976nodes in 432 rack**
**totally 4.85Peta byte memory**
**with bandwidth 163 PB/s**

# TofuD interconnect

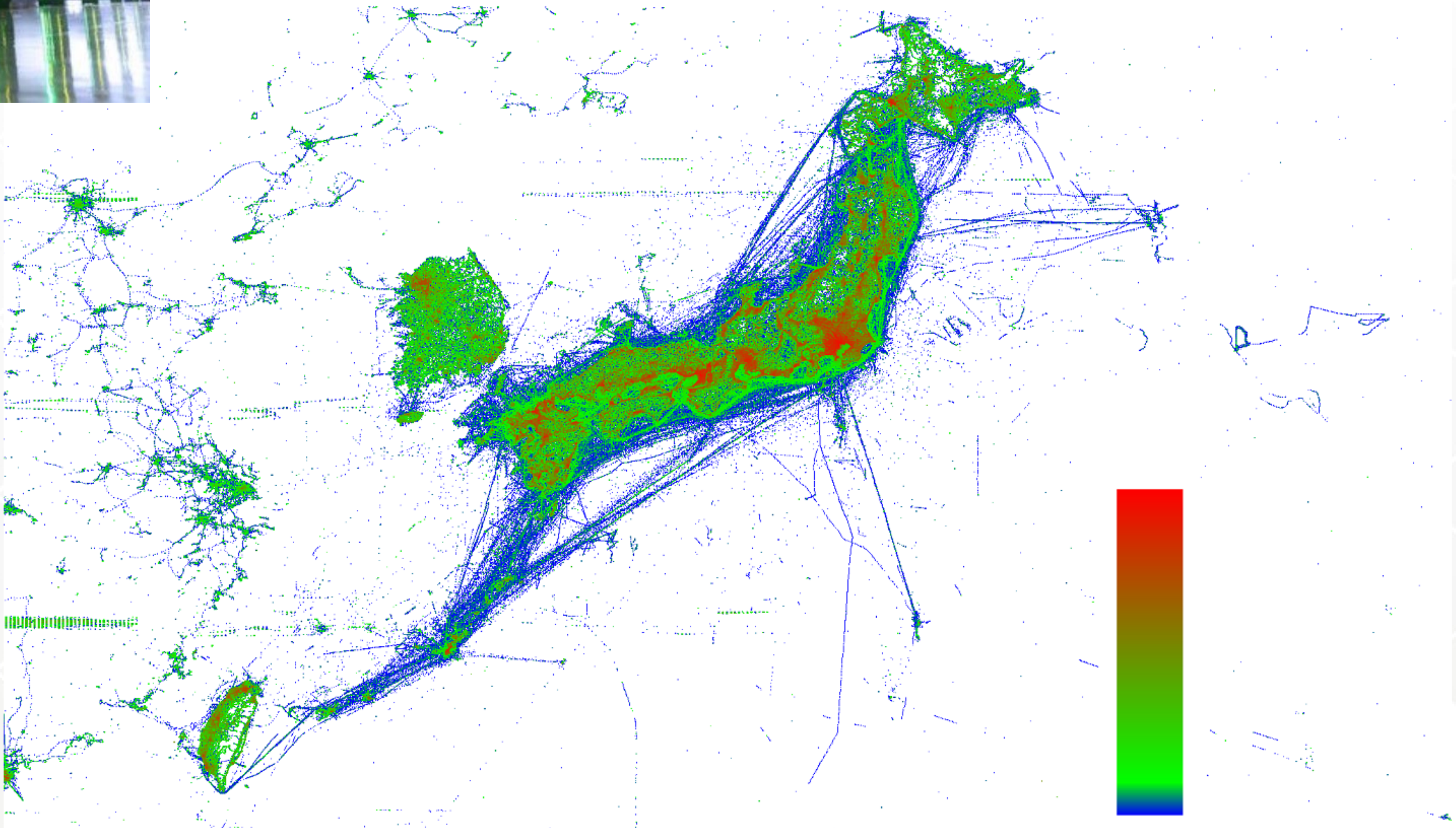
- Six coordinate axes: X, Y, Z, A, B, C
  - X, Y, Z: the size varies according to the system configuration
  - A, B, C: the size is fixed to 2 × 3 × 2
- Tofu stands for “torus fusion”: (X, Y, Z) × (A, B, C)




**Max performance**

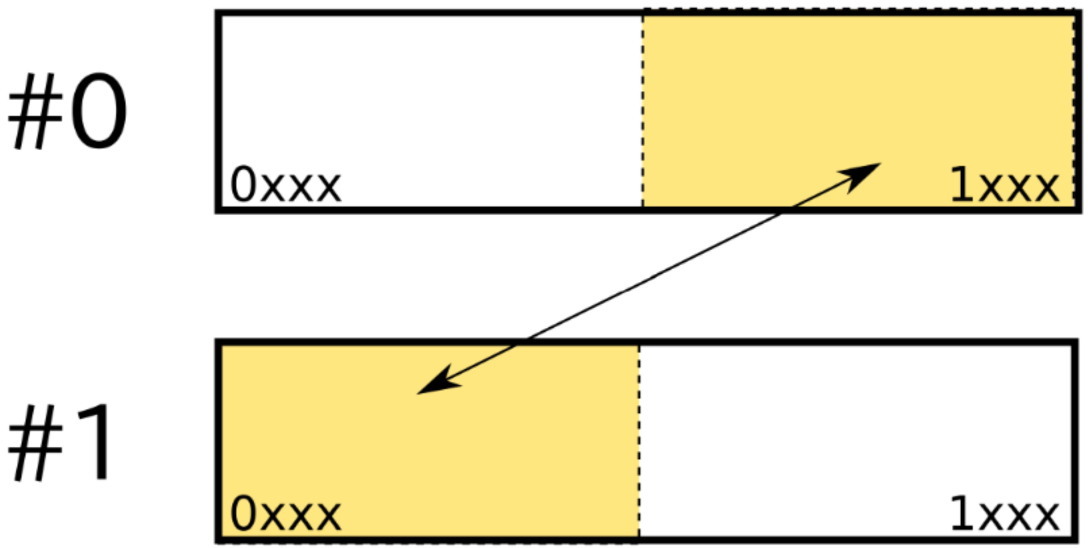
**each node**
**40.8 GB/s**
**2 lanes x 10 ports**

**full system**
**6.19 PB hop/s**



- 2 MPI processes ($M = 1$ global qubit)
  - process #0 and #1
- Single-qubit gate on the global qubit $(*| * * \cdots)$
  - swap the global qubit and the leftmost local qubit

$$\sigma \mapsto \begin{pmatrix} N & N-1 & N-2 & \cdots & 1 \\ N-1 & N & N-2 & \cdots & 1 \end{pmatrix} \cdot \sigma$$



# How to allocate coefficients to node? → qubit swap algorithm

Case of N=6 qbits on M=4( using $2^{4-2}$=4 nodes) : $|\Psi> = \sum_{n_0,n_1,n_2,n_3,n_4,n_5=0,1} a(n_0,n_1,n_2,n_3,n_4,n_5)|n_0 n_1 n_2 n_3 n_4 n_5>$

node00

| | |
|---|---|
| 0000 | a(0,0,0,0,0,0) |
| 0001 | a(0,0,0,0,0,1) |
| 0010 | a(0,0,0,0,1,0) |
| 0011 | a(0,0,0,0,1,1) |
| 0100 | a(0,0,0,1,0,0) |
| 0101 | a(0,0,0,1,0,1) |
| 0110 | a(0,0,0,1,1,0) |
| 0111 | a(0,0,0,1,1,1) |
| 1000 | a(0,0,1,0,0,0) |
| 1001 | a(0,0,1,0,0,1) |
| 1010 | a(0,0,1,0,1,0) |
| 1011 | a(0,0,1,0,1,1) |
| 1100 | a(0,0,1,1,0,0) |
| 1101 | a(0,0,1,1,0,1) |
| 1110 | a(0,0,1,1,1,0) |
| 1111 | a(0,0,1,1,1,1) |

node01

| | |
|---|---|
| 0000 | a(0,1,0,0,0,0) |
| 0001 | a(0,1,0,0,0,1) |
| 0010 | a(0,1,0,0,1,0) |
| 0011 | a(0,1,0,0,1,1) |
| 0100 | a(0,1,0,1,0,0) |
| 0101 | a(0,1,0,1,0,1) |
| 0110 | a(0,1,0,1,1,0) |
| 0111 | a(0,1,0,1,1,1) |
| 1000 | a(0,1,1,0,0,0) |
| 1001 | a(0,1,1,0,0,1) |
| 1010 | a(0,1,1,0,1,0) |
| 1011 | a(0,1,1,0,1,1) |
| 1100 | a(0,1,1,1,0,0) |
| 1101 | a(0,1,1,1,0,1) |
| 1110 | a(0,1,1,1,1,0) |
| 1111 | a(0,1,1,1,1,1) |

node10

| | |
|---|---|
| 0000 | a(1,0,0,0,0,0) |
| 0001 | a(1,0,0,0,0,1) |
| 0010 | a(1,0,0,0,1,0) |
| 0011 | a(1,0,0,0,1,1) |
| 0100 | a(1,0,0,1,0,0) |
| 0101 | a(1,0,0,1,0,1) |
| 0110 | a(1,0,0,1,1,0) |
| 0111 | a(1,0,0,1,1,1) |
| 1000 | a(1,0,1,0,0,0) |
| 1001 | a(1,0,1,0,0,1) |
| 1010 | a(1,0,1,0,1,0) |
| 1011 | a(1,0,1,0,1,1) |
| 1100 | a(1,0,1,1,0,0) |
| 1101 | a(1,0,1,1,0,1) |
| 1110 | a(1,0,1,1,1,0) |
| 1111 | a(1,0,1,1,1,1) |

node11

| | |
|---|---|
| 0000 | a(1,1,0,0,0,0) |
| 0001 | a(1,1,0,0,0,1) |
| 0010 | a(1,1,0,0,1,0) |
| 0011 | a(1,1,0,0,1,1) |
| 0100 | a(1,1,0,1,0,0) |
| 0101 | a(1,1,0,1,0,1) |
| 0110 | a(1,1,0,1,1,0) |
| 0111 | a(1,1,0,1,1,1) |
| 1000 | a(1,1,1,0,0,0) |
| 1001 | a(1,1,1,0,0,1) |
| 1010 | a(1,1,1,0,1,0) |
| 1011 | a(1,1,1,0,1,1) |
| 1100 | a(1,1,1,1,0,0) |
| 1101 | a(1,1,1,1,0,1) |
| 1110 | a(1,1,1,1,1,0) |
| 1111 | a(1,1,1,1,1,1) |

Global qubits: allocated to node-number bits

Local qubits: allocated to memory-address bits

Fugaku

# Apply H 5

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

RIKEN R-CCS

## node00

| | |
|---|---|
| 0000 | a(0,0,0,0,0,0) |
| 0001 | a(0,0,0,0,0,1) |
| 0010 | a(0,0,0,0,1,0) |
| 0011 | a(0,0,0,0,1,1) |
| 0100 | a(0,0,0,1,0,0) |
| 0101 | a(0,0,0,1,0,1) |
| 0110 | a(0,0,0,1,1,0) |
| 0111 | a(0,0,0,1,1,1) |
| 1000 | a(0,0,1,0,0,0) |
| 1001 | a(0,0,1,0,0,1) |
| 1010 | a(0,0,1,0,1,0) |
| 1011 | a(0,0,1,0,1,1) |
| 1100 | a(0,0,1,1,0,0) |
| 1101 | a(0,0,1,1,0,1) |
| 1110 | a(0,0,1,1,1,0) |
| 1111 | a(0,0,1,1,1,1) |

## node01

| | |
|---|---|
| 0000 | a(0,1,0,0,0,0) |
| 0001 | a(0,1,0,0,0,1) |
| 0010 | a(0,1,0,0,1,0) |
| 0011 | a(0,1,0,0,1,1) |
| 0100 | a(0,1,0,1,0,0) |
| 0101 | a(0,1,0,1,0,1) |
| 0110 | a(0,1,0,1,1,0) |
| 0111 | a(0,1,0,1,1,1) |
| 1000 | a(0,1,1,0,0,0) |
| 1001 | a(0,1,1,0,0,1) |
| 1010 | a(0,1,1,0,1,0) |
| 1011 | a(0,1,1,0,1,1) |
| 1100 | a(0,1,1,1,0,0) |
| 1101 | a(0,1,1,1,0,1) |
| 1110 | a(0,1,1,1,1,0) |
| 1111 | a(0,1,1,1,1,1) |

## node10

| | |
|---|---|
| 0000 | a(1,0,0,0,0,0) |
| 0001 | a(1,0,0,0,0,1) |
| 0010 | a(1,0,0,0,1,0) |
| 0011 | a(1,0,0,0,1,1) |
| 0100 | a(1,0,0,1,0,0) |
| 0101 | a(1,0,0,1,0,1) |
| 0110 | a(1,0,0,1,1,0) |
| 0111 | a(1,0,0,1,1,1) |
| 1000 | a(1,0,1,0,0,0) |
| 1001 | a(1,0,1,0,0,1) |
| 1010 | a(1,0,1,0,1,0) |
| 1011 | a(1,0,1,0,1,1) |
| 1100 | a(1,0,1,1,0,0) |
| 1101 | a(1,0,1,1,0,1) |
| 1110 | a(1,0,1,1,1,0) |
| 1111 | a(1,0,1,1,1,1) |

## node11

| | |
|---|---|
| 0000 | a(1,1,0,0,0,0) |
| 0001 | a(1,1,0,0,0,1) |
| 0010 | a(1,1,0,0,1,0) |
| 0011 | a(1,1,0,0,1,1) |
| 0100 | a(1,1,0,1,0,0) |
| 0101 | a(1,1,0,1,0,1) |
| 0110 | a(1,1,0,1,1,0) |
| 0111 | a(1,1,0,1,1,1) |
| 1000 | a(1,1,1,0,0,0) |
| 1001 | a(1,1,1,0,0,1) |
| 1010 | a(1,1,1,0,1,0) |
| 1011 | a(1,1,1,0,1,1) |
| 1100 | a(1,1,1,1,0,0) |
| 1101 | a(1,1,1,1,0,1) |
| 1110 | a(1,1,1,1,1,0) |
| 1111 | a(1,1,1,1,1,1) |

# Apply H 1: exchange blocks

## Swap data in memory blocks …

### node00

| | |
|---|---|
| 0000 | a(0,0,0,0,0,0) |
| 0001 | a(0,0,0,0,0,1) |
| 0010 | a(0,0,0,0,1,0) |
| 0011 | a(0,0,0,0,1,1) |
| 0100 | a(0,0,0,1,0,0) |
| 0101 | a(0,0,0,1,0,1) |
| 0110 | a(0,0,0,1,1,0) |
| 0111 | a(0,0,0,1,1,1) |
| 1000 | a(0,0,1,0,0,0) |
| 1001 | a(0,0,1,0,0,1) |
| 1010 | a(0,0,1,0,1,0) |
| 1011 | a(0,0,1,0,1,1) |
| 1100 | a(0,0,1,1,0,0) |
| 1101 | a(0,0,1,1,0,1) |
| 1110 | a(0,0,1,1,1,0) |
| 1111 | a(0,0,1,1,1,1) |

### node01

| | |
|---|---|
| 0000 | a(0,1,0,0,0,0) |
| 0001 | a(0,1,0,0,0,1) |
| 0010 | a(0,1,0,0,1,0) |
| 0011 | a(0,1,0,0,1,1) |
| 0100 | a(0,1,0,1,0,0) |
| 0101 | a(0,1,0,1,0,1) |
| 0110 | a(0,1,0,1,1,0) |
| 0111 | a(0,1,0,1,1,1) |
| 1000 | a(0,1,1,0,0,0) |
| 1001 | a(0,1,1,0,0,1) |
| 1010 | a(0,1,1,0,1,0) |
| 1011 | a(0,1,1,0,1,1) |
| 1100 | a(0,1,1,1,0,0) |
| 1101 | a(0,1,1,1,0,1) |
| 1110 | a(0,1,1,1,1,0) |
| 1111 | a(0,1,1,1,1,1) |

### node10

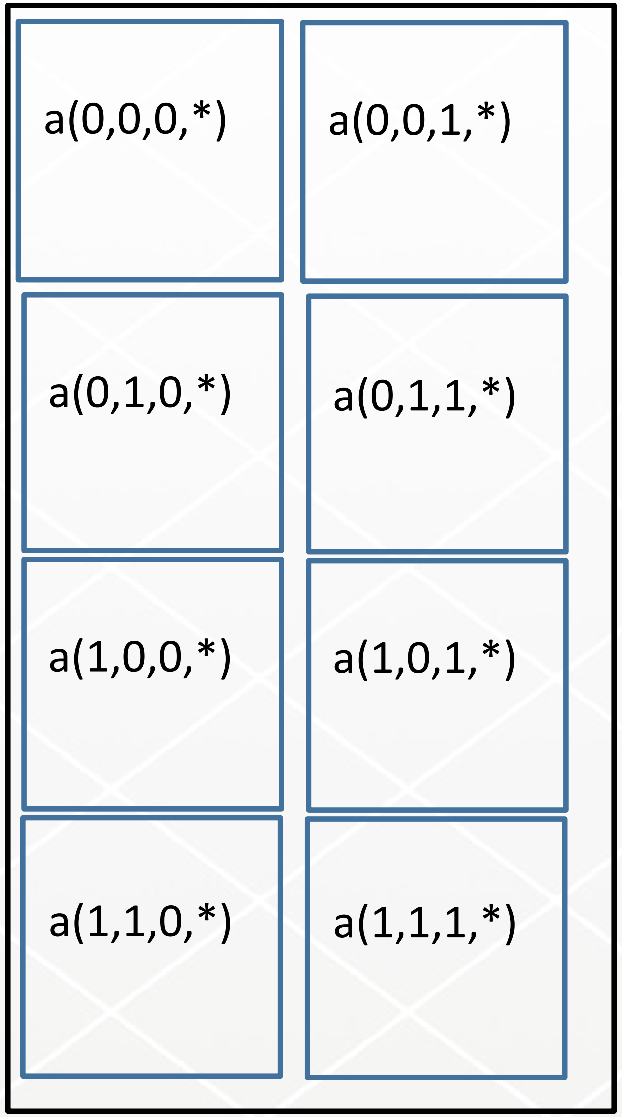
| | |
|---|---|
| 0000 | a(1,0,0,0,0,0) |
| 0001 | a(1,0,0,0,0,1) |
| 0010 | a(1,0,0,0,1,0) |
| 0011 | a(1,0,0,0,1,1) |
| 0100 | a(1,0,0,1,0,0) |
| 0101 | a(1,0,0,1,0,1) |
| 0110 | a(1,0,0,1,1,0) |
| 0111 | a(1,0,0,1,1,1) |
| 1000 | a(1,0,1,0,0,0) |
| 1001 | a(1,0,1,0,0,1) |
| 1010 | a(1,0,1,0,1,0) |
| 1011 | a(1,0,1,0,1,1) |
| 1100 | a(1,0,1,1,0,0) |
| 1101 | a(1,0,1,1,0,1) |
| 1110 | a(1,0,1,1,1,0) |
| 1111 | a(1,0,1,1,1,1) |

### node11

| | |
|---|---|
| 0000 | a(1,1,0,0,0,0) |
| 0001 | a(1,1,0,0,0,1) |
| 0010 | a(1,1,0,0,1,0) |
| 0011 | a(1,1,0,0,1,1) |
| 0100 | a(1,1,0,1,0,0) |
| 0101 | a(1,1,0,1,0,1) |
| 0110 | a(1,1,0,1,1,0) |
| 0111 | a(1,1,0,1,1,1) |
| 1000 | a(1,1,1,0,0,0) |
| 1001 | a(1,1,1,0,0,1) |
| 1010 | a(1,1,1,0,1,0) |
| 1011 | a(1,1,1,0,1,1) |
| 1100 | a(1,1,1,1,0,0) |
| 1101 | a(1,1,1,1,0,1) |
| 1110 | a(1,1,1,1,1,0) |
| 1111 | a(1,1,1,1,1,1) |

# Bit-swap algorithm for inter-node

node $n_0 n_1 n_2$...




1 bit swap: swap of 1/2

2 bit swap: swap of 3/4

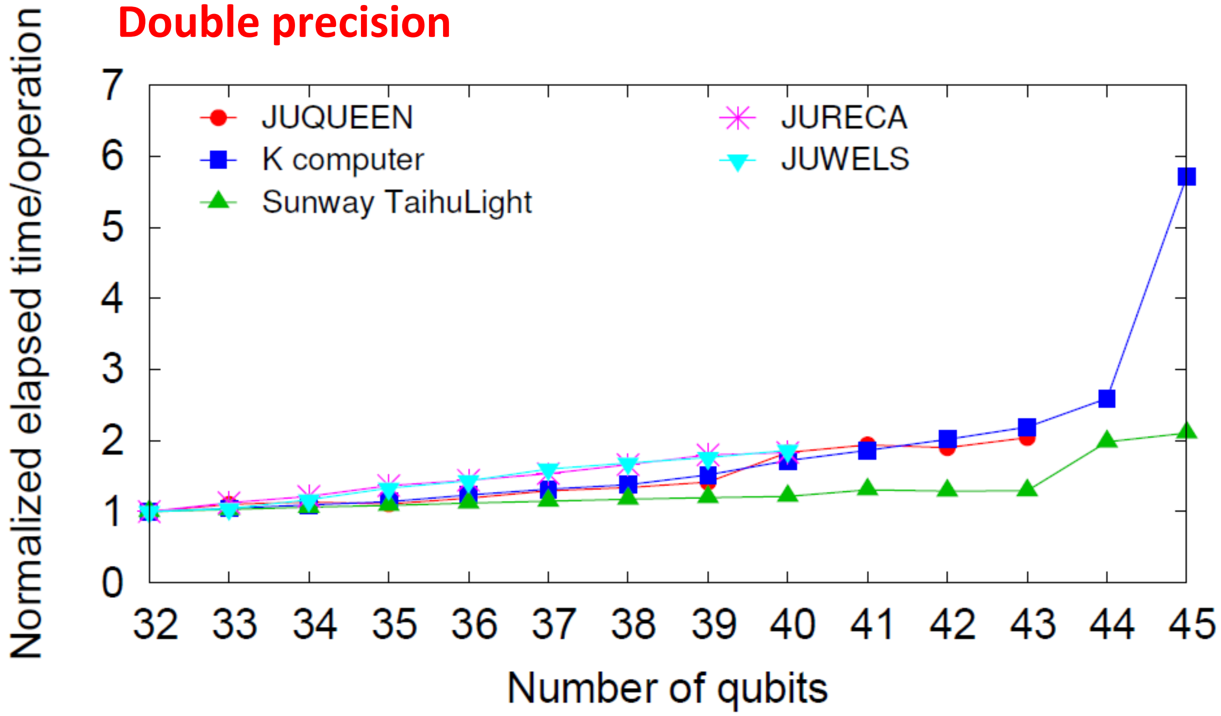
3 bit swap: swap of 7/8

...

Table 1: Overview of the computer systems used for benchmarking. The IBM Blue Gene/Q JUQUEEN [3] (decomissioned), JURECA [4] and JUWELS are located at the Jülich Supercomputing Centre in Germany, the K computer of the RIKEN Center for Computational Science in Kobe, Japan, and the Sunway TaihuLight [5] at the National Supercomputer Center in Wuxi, China. The row "# qubits" gives the maximum number of qubits $N$ that can be simulated with JUQCS–A (JUQCS–E). At the time of running the benchmarks on JUWELS, the maximum number of qubits $N$ was limited to 43 (40).

| | JUQUEEN | K computer | Sunway TaihuLight | JURECA-CLUSTER | JUWELS |
|---|---|---|---|---|---|
| CPU | IBM PowerPC A2 | eight-core SPARC64 VIIIfx | SW26010 manycore 64-bit RISC | Intel Xeon E5-2680 v3 | Dual Intel Xeon Platinum 8168 |
| clock frequency | 1.6 GHz | 2.0 Ghz | 1.45 GHz | 2.5 GHz | 2.7 GHz |
| memory/node | 16 GB | 16 GB | 32 GB | 128 GB | 96 GB |
| # threads/core used | 1 – 2 | 8 | 1 | 1 – 2 | 1 – 2 |
| # cores used | 1 – 262144 | 2 – 65536 | 1 – 131072 | 1 – 6144 | 1 – 98304 |
| # nodes used | 1 – 16384 | 2 – 65536 | 1 – 32768 | 1 – 256 | 1 – 2048 |
| # MPI processes used | 1 – 524288 | 2 – 65536 | 1 – 131072 | 1 – 1024 | 1 – 2048 |
| # qubits | 46 (43) | 48 (45) | 48 (45) | 43 (40) | 46 (43) |
| | No.1 in 2011 10PFLOPS | No.5 in 2013 5PFLOPS | No.1 in 2016 93PFLOPS | | |

**Double precision**




Normalied elapsed time by elapsed time of 32 qbits case

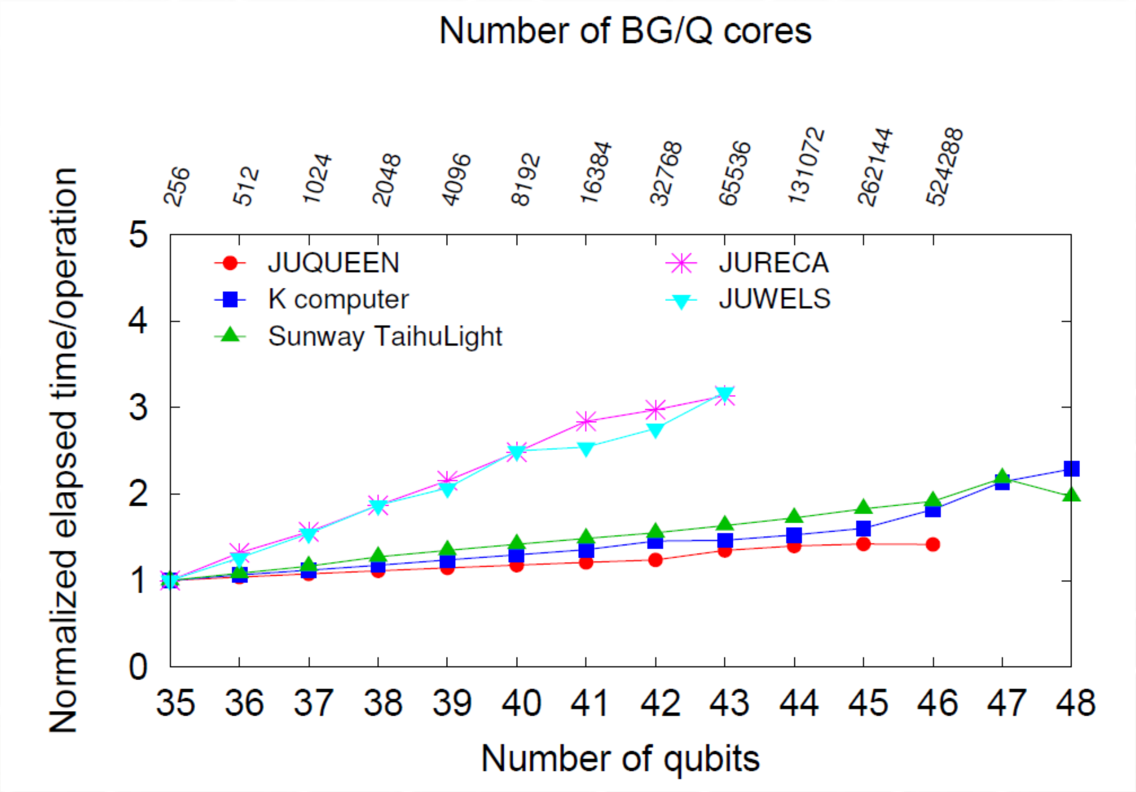
Normalization factors:

- JUQUEEN 1.2sec
- K 1.0sec
- Sunway TaihuLight 7.7sec
- JURECA 1.9sec
- JUWELS 1.3sec

H. De Raedt, F. Jin, D. Wilsch, M. Nocon, N. Yoshioka, N. Ito, S. Yuan and K. Michielsen, Comp. Phys. Comm. Vol.237 (2019) p.47-61

# Performance: one Hadamard operation for all qbits

# more qbits with reduced precision using one-byte coding



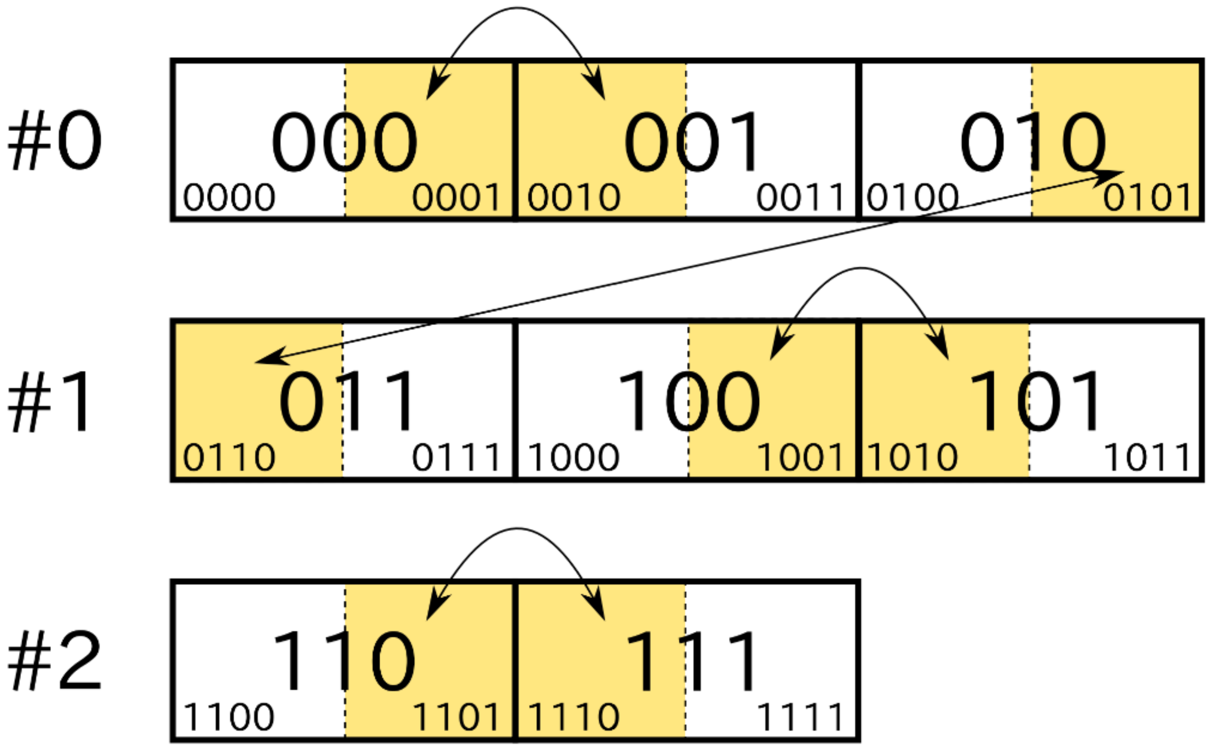

Normalied elapsed time by elapsed time of 35 qbits case

Normalization factors:

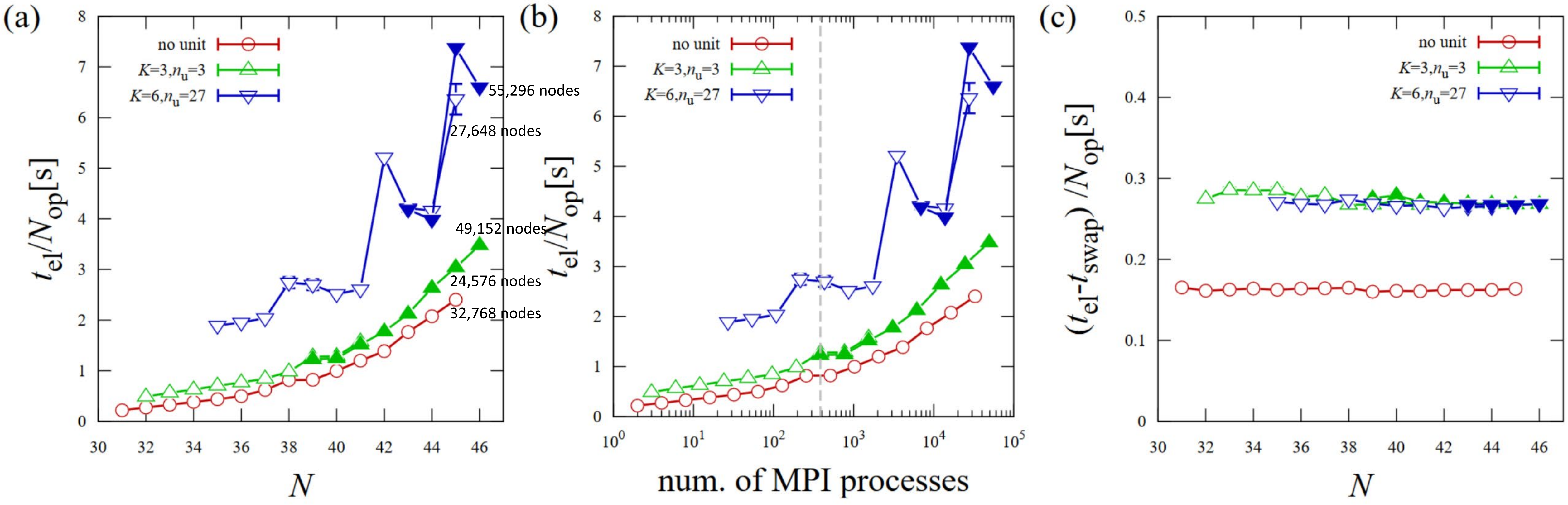
- JUQUEEN 2.7sec
- K 3.8sec
- Sunway TaihuLight 19.9sec
- JURECA 2.4sec
- JUWELS 2.2sec

H. De Raedt, F. Jin, D. Wilsch, M. Nocon, N. Yoshioka, N. Ito, S. Yuan and K. Michielsen, Comp. Phys. Comm. Vol.237 (2019) p.47-61

- 1 unit ($M = 0$ global qubit), $K = 3$ unit qubits, $n_{\mathrm{u}} = 3$ MPI processes
  - process #0: $u = 000, 001, 010$, #1: $u = 011, 100, 101$, #2: $u = 110, 111$
  - apply single-qubit gate on rightmost unit qubit ($* * *| \cdots$)
    - swap the unit qubit and the leftmost local qubit



# Performance of Hadamard benchmark(once) on the "Fugaku"




cf on K, 35 qubits: 1.2 sec
40 qubits: 2.0 sec

- no unit : basic parallelization using $2^M$ processes
- others : non $2^M$ parallelization

Fugaku

## Limitations of the qubit-swap implementations

### Number of coefficients on each node is $2^n$.

| | memory per node | available memory |
|---|---|---|
| K | 16GB | 8 GB |
| Fugaku | 32GB | 16 GB |

### Number of used nodes is $2^m$.

| | number of nodes | available nodes |
|---|---|---|
| K | 88,128 | $65{,}536 = 2^{16}$ |
| Fugaku | 158,976 | $131{,}072 = 2^{17}$ |

→

## Remove these limitations by allocating more qubits

Pack more coefficients to each node:

Local set: use smaller number of local qubits 0.5GB

allocate many local sets to each node 53 local sets using 26.5GB

→ 4PB using 158,276 nodes of 158,976 nodes of Fugaku

48 qubits(double), 49(single), 50(half) and 51(byte)

Or "unit" qubits between local and global qubits

Coefficients of (unit+local) qubits are in some nodes or processes.

## Limitations of memory band-width

**In node: 1TB/s(Fugaku) ~100GB(x86 Workstation)**

**Internode: 40.8GB/s(Fugaku)**

one load and one store per operatio

16GB(30 qubits in double)/node → 16GBx2/1TB = 31 msec

one bit-swap per operation

8GB transfer → 8TBx2/40.8GB = 392 msec

**sub-second per operation → ～$10^5$ operations in 10 hours**

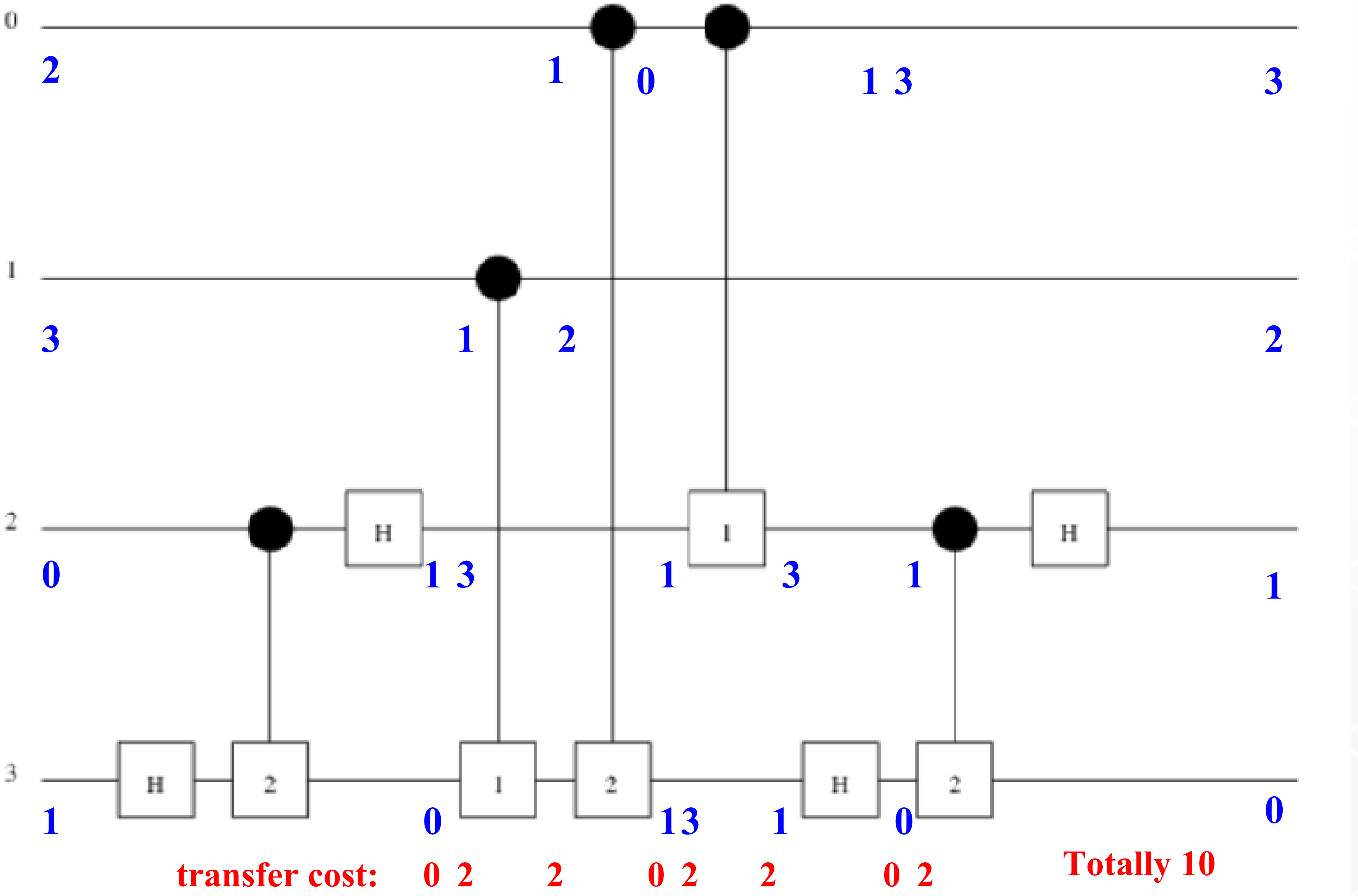
→

## Remove these limitations by optimal allocation to increase circuit depth

0.5GB(26 qubits in single)/node → 0.5GBx2/1TB = 1 msec

internode 0.25GBx2/40.8GB = 12 msec

**10 ms per operation → ～$10^6$ operations in 10 hours**




**example: 2+2, 4qbit, 9 lines, local 2bit**

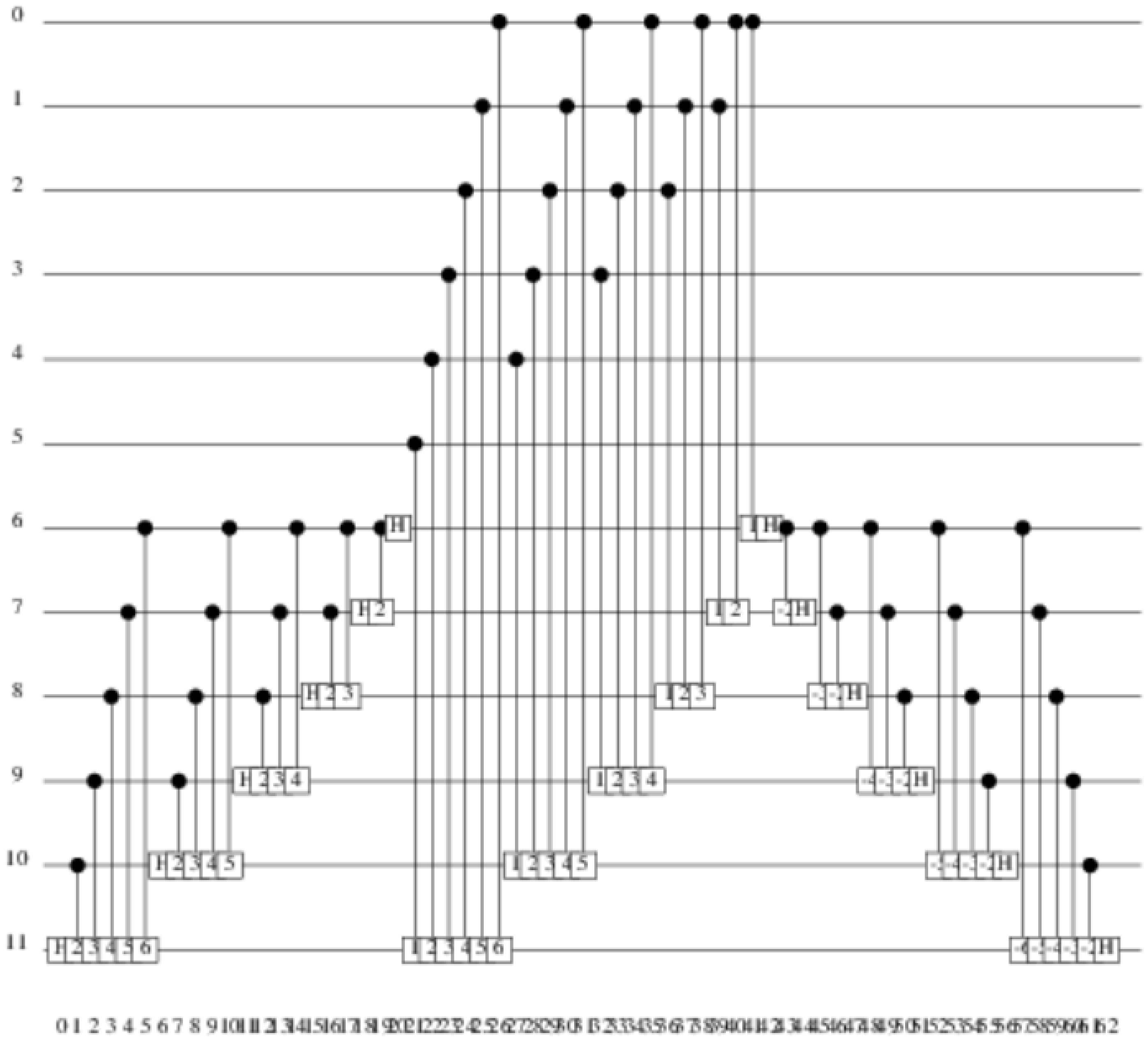
**Blue figures show bit-position of address**
**0 and 1: local, 2 and 3: global**

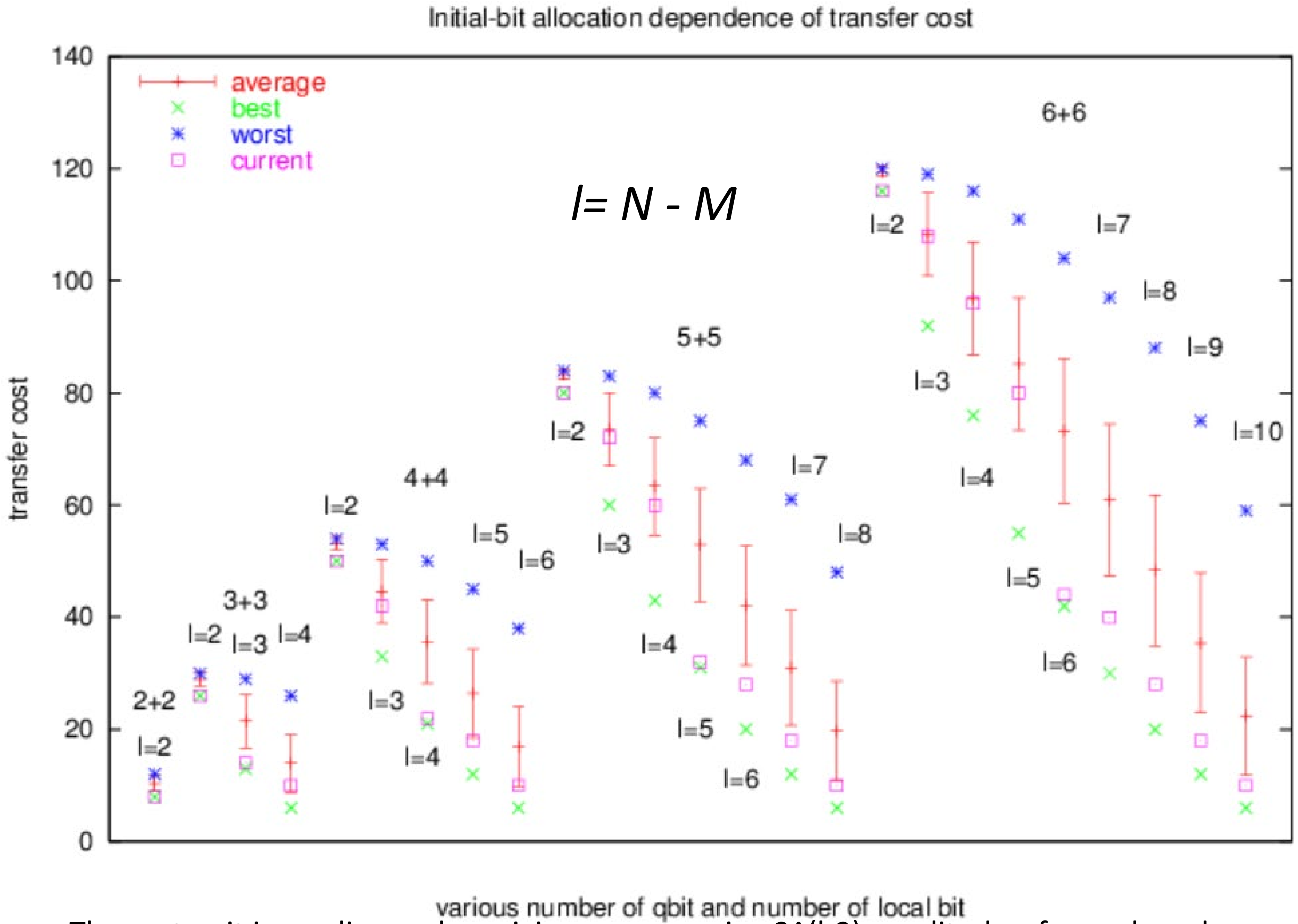
0 1 2 3 4 5 6 7 8

# 6+6, 12qbit, 63 lines, local 6bit

average 73.22
standard deviation 12.94

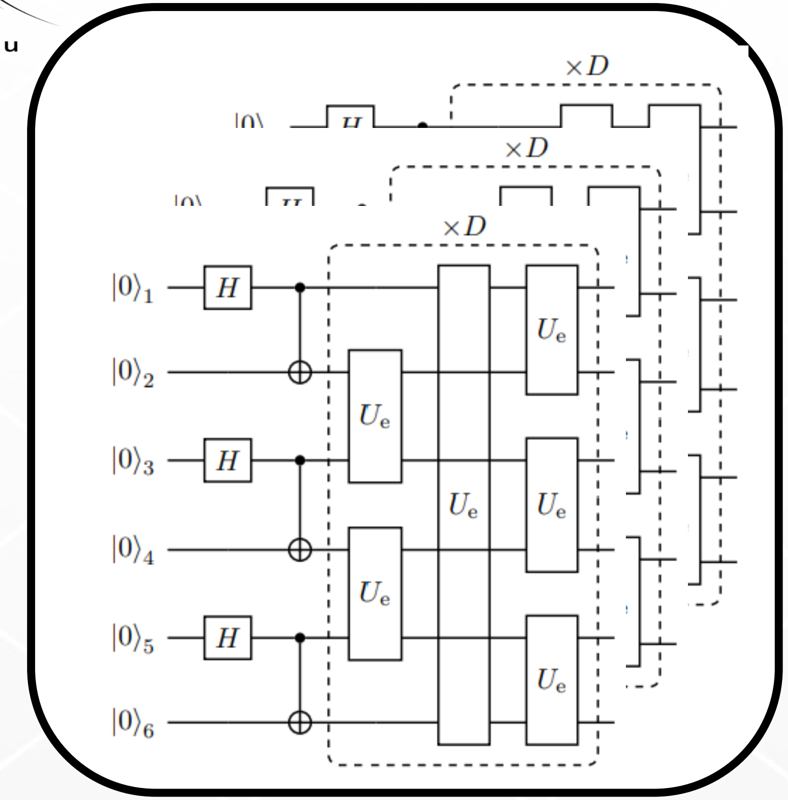
| best | worst |
| --- | --- |
| 42 | 104 |
| 0 | 4 |
| 6 | 5 |
| 7 | 0 |
| 8 | 1 |
| 9 | 2 |
| 10 | 3 |
| 11 | 6 |
| 1 | 7 |
| 2 | 8 |
| 3 | 9 |
| 4 | 10 |
| 5 | 11 |
| Totally 34560 | Totally 933120 |
| 0.0072% | 0.19% |

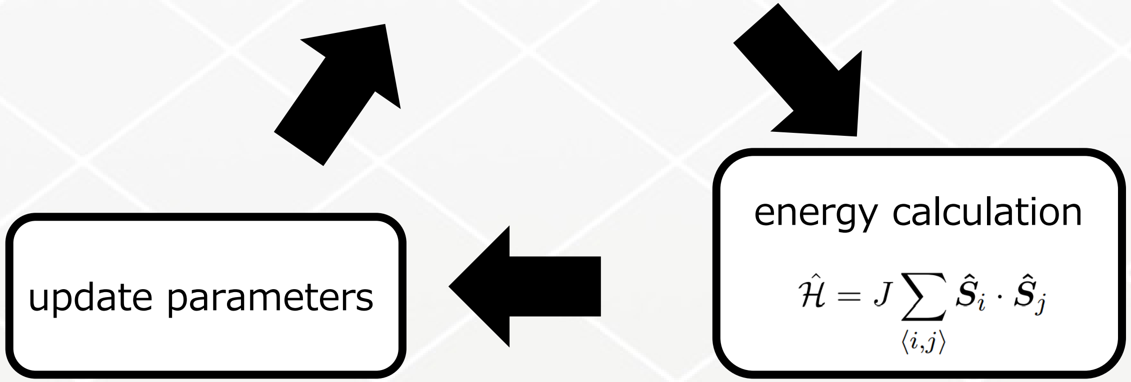


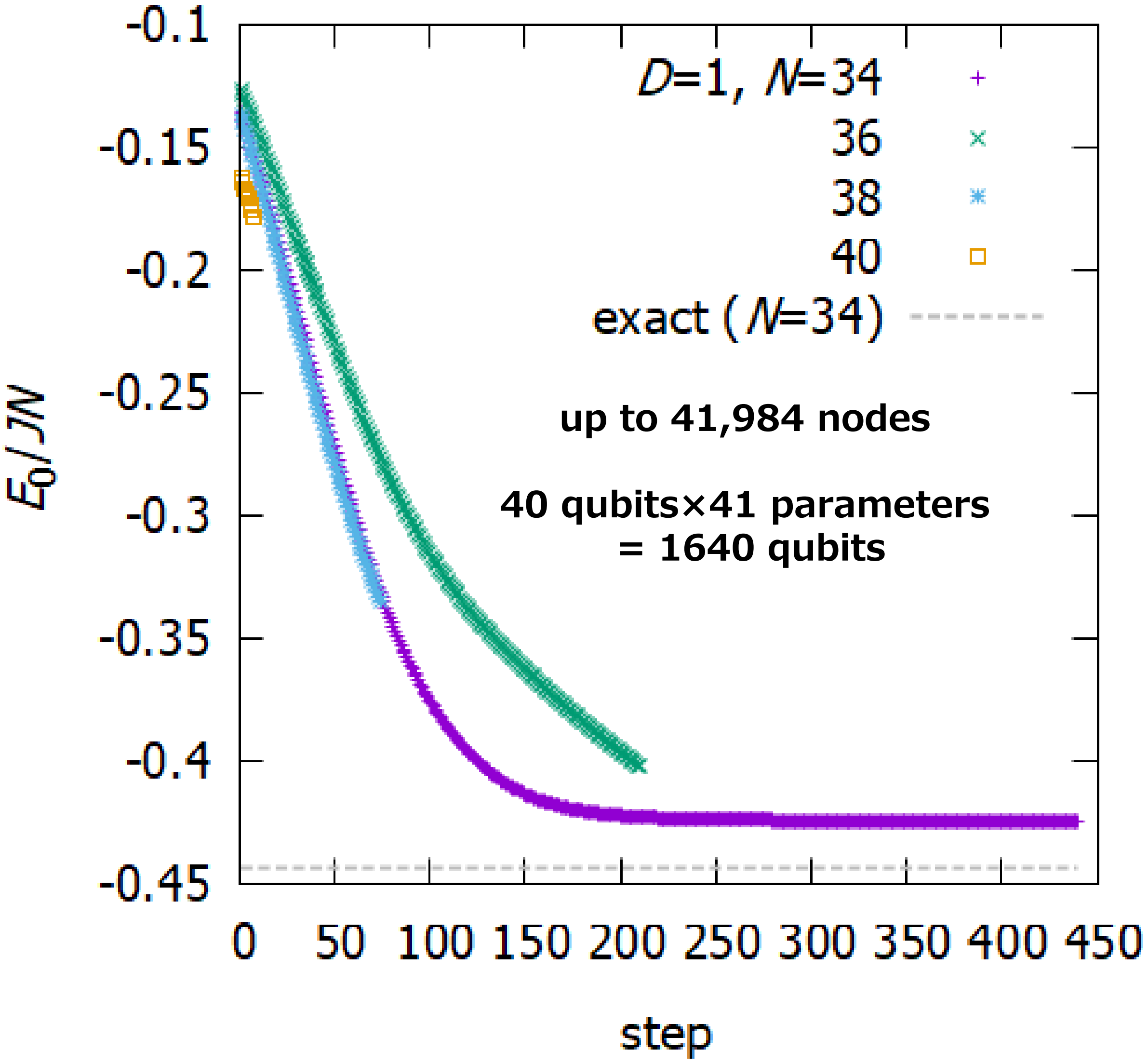



The cost unit is sending and receiving one page, i.e. 2^(l-2) amplitudes, for each nodes.

# VQE simulation Heisenberg chain



Fugaku

RIKEN R-CCS

# Summary

- Algorithm and program for massively parallel supercomputers are showns.

- On Fugaku, state-vector simulation can reach
  48 qubits in double precision and 51 qubits in byte precision.

# Perspectives

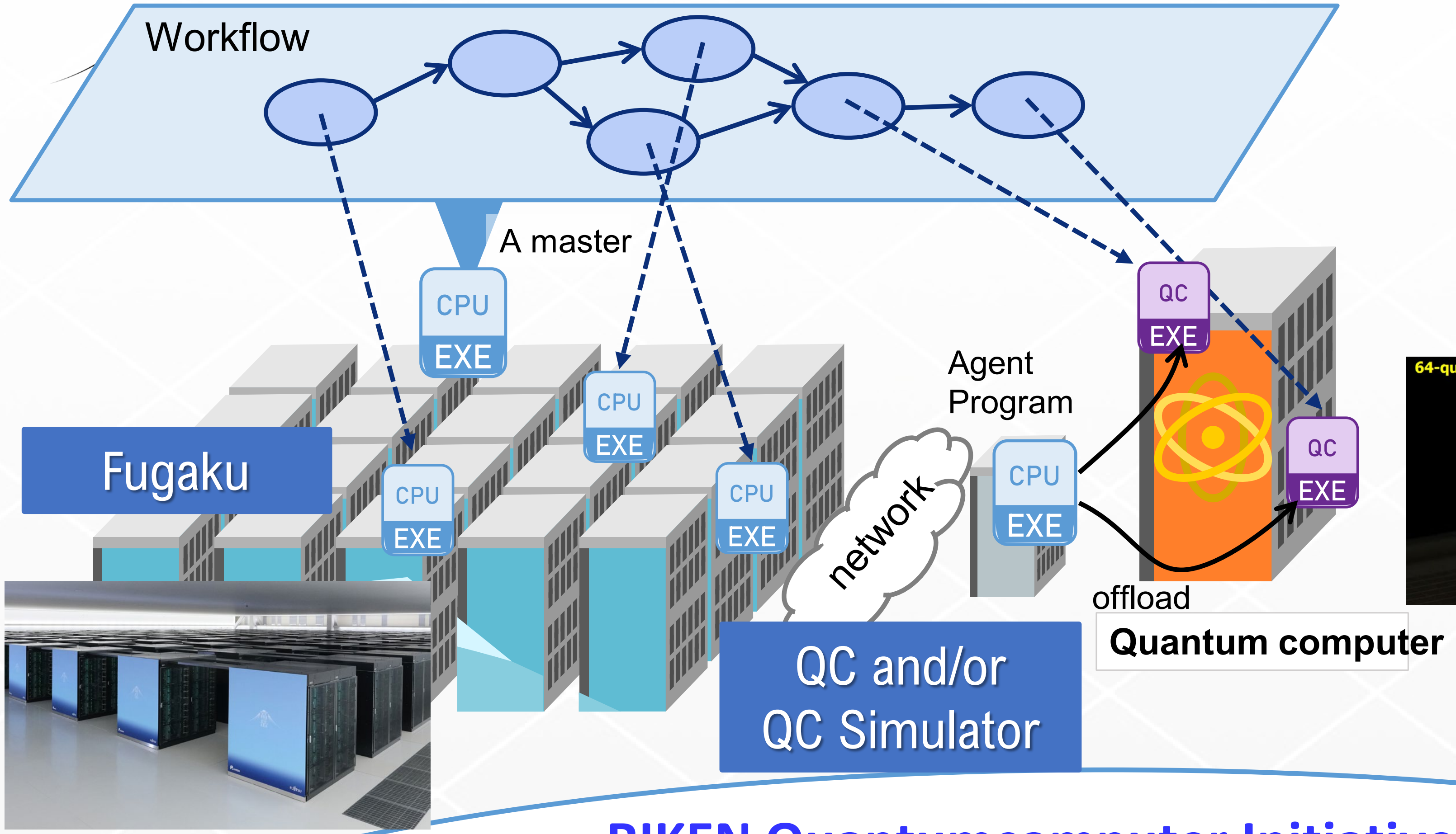
- Further tuning, mainly for data transfer, will improve performance.

- Applications to development of quantum computer and to test quantum algorithms have been started jointly with RIKEN quantum computer center(RQC).

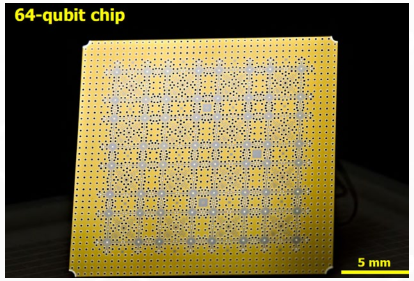
- Optimization of execution speed.

**braket: A open-source quantum computer simulator for parallel computer**
**C++, two-to-N implementation, no unit qubit**
**https://github.com/naoki-yoshioka/braket**

RIKEN R-CCS









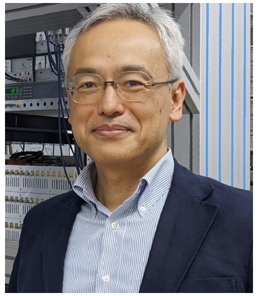
## RIKEN Quantumcomputer Initiative

SG NSCC SINGAPORE National Supercomputing Centre

Agency for Science, Technology and Research SINGAPORE

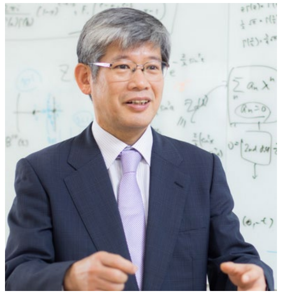
President
M. Gonokami

RQC
Y. Nakamura



R-CCS
S. Matsuoka

iTHEMS
T. Hatsuda



AIP
M. Sugiyama

FUJITSU

IBM

NVIDIA

# Software environment for classical-quantum hybrid computation

Fugaku/Fugaku NEXT

API and work-flow scheduler

cloud/network

Near-QC system （GPU & CPU)

Interface with low latency and high bandwidth

QPU

QPU simulator on Fugaku

Quantum applications

API and near-QC programming model

Quantum programming language with unified intermediate representation

Software layders necessary to test and evaluation quantum applications

Algorithms

- Hybrid
- Quantum
- NISQ

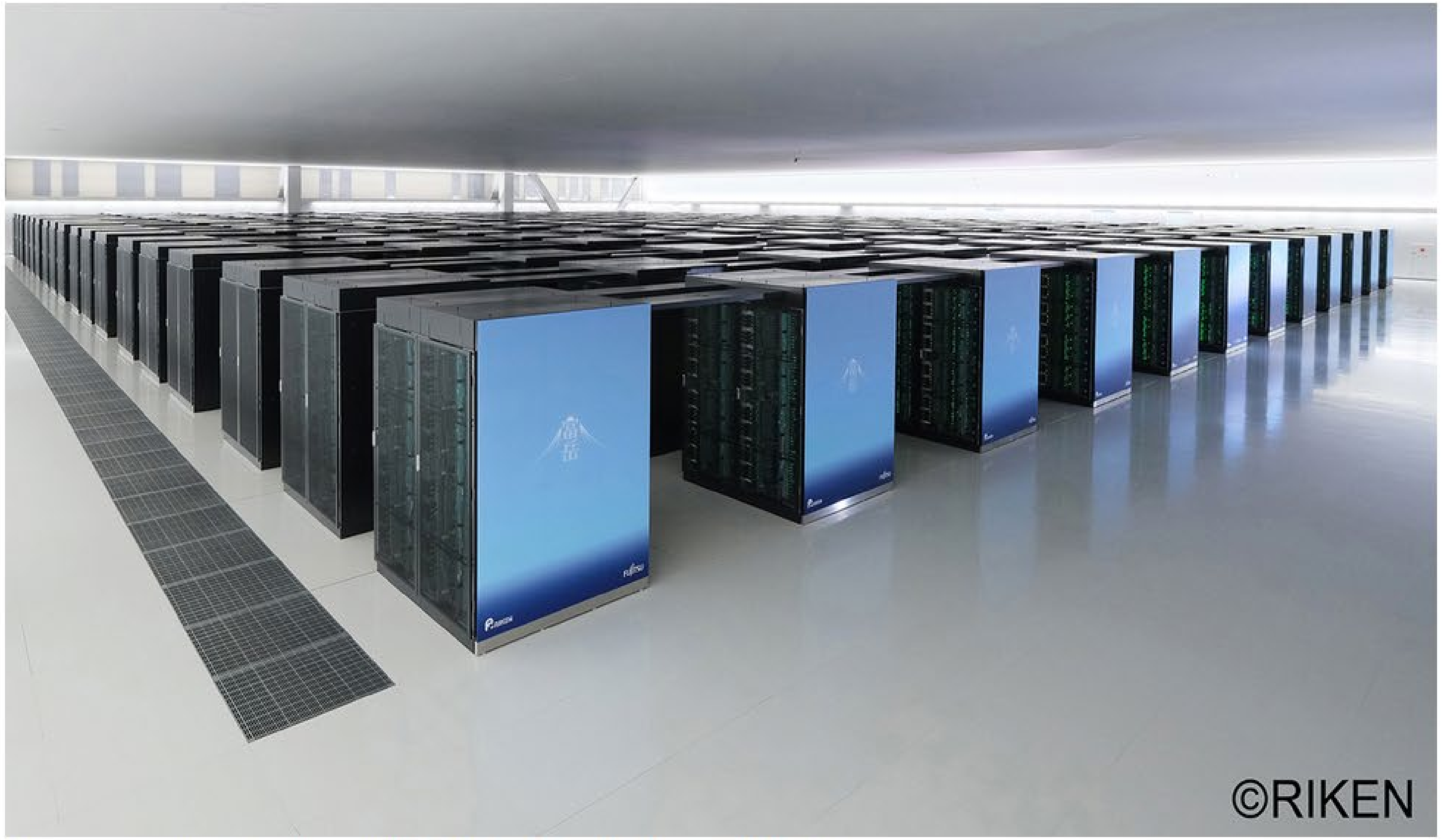
Various tools and environments

- Quantum BLAS： basic linear algebra subprogram
- Qiskit：framework by IBM
- Cirq：development tools by Google
- TensorFlow Quantum：framework for quantum machine learning
- Q#:language by Microsoft
- Qulacs, QunaSys: simulator and language by Osaka University
- Covalent: development workflow for quantum HPC environment by Agnostiq
- PyQubo：library for quantu annealing
- cuQuantum and QODA: simulator and programing language for GPU by NVIDIA
- TKET: development kit by Quantinuum

https://ccp2023.jp

CCP2023

- Home
- Key Dates
- Venue
- Registration
- Abstracts
- Program
- ECSA 2023
- Organizers
- History
- Sponsors
- Travel Information
- Contact




Supercomputer Fugaku at RIKEN R-CCS in Kobe (© RIKEN)

# CCP2023 - 34th IUPAP Conference on Computational Physics

- Date: August 4 (Fri) - 8 (Tue), 2023
- Venue: Kobe International Conference Center, Kobe Port Island, Kobe, Japan