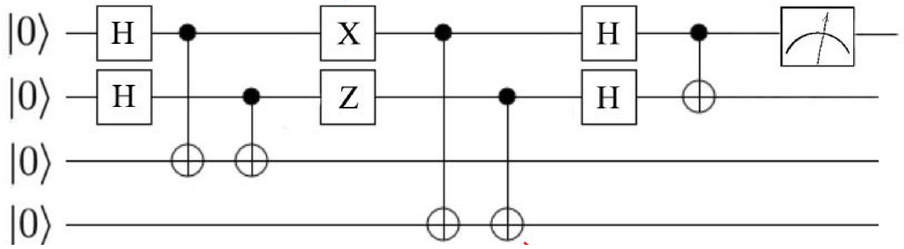
NVIDIA cuQuantum SDK: Accelerating Quantum Circuit simulation II – cuStateVec

Shinya Morino, Pr. Math Libraries Engineer, Quantum Computing
CUDA Math Libraries Team, NVIDIA

Quantum Computing: Devices, Cryogenic Electronics and Packaging, 10/25/2023

Two Most Popular Quantum Circuit Simulation Approaches



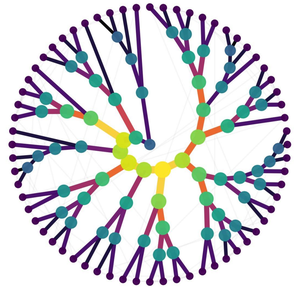
State vector simulation

“Gate-based simulation of a quantum computer”

- Maintain full 2^n qubit vector state in memory
- Update all states every timestep, probabilistically sample n of the states for measurement

Memory capacity & time grow exponentially w/ # of qubits - practical limit around 50 qubits on a supercomputer

Can model either ideal or noisy qubits



Tensor networks

“Only simulate the states you need”

- Uses tensor network contractions to dramatically reduce memory for simulating circuits
- Can simulate 100s or 1000s of qubits for many practical quantum circuits

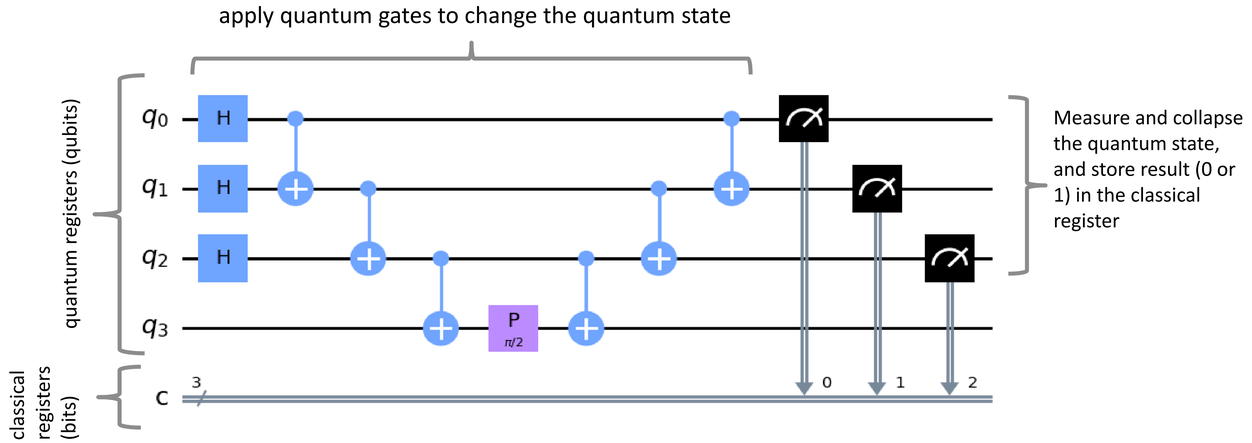
GPUs are a great fit for either approach

Tensor Network image from Quimb: https://quimb.readthedocs.io/en/latest/index.html

cuStateVec

A library to accelerate state vector-based quantum circuit simulation

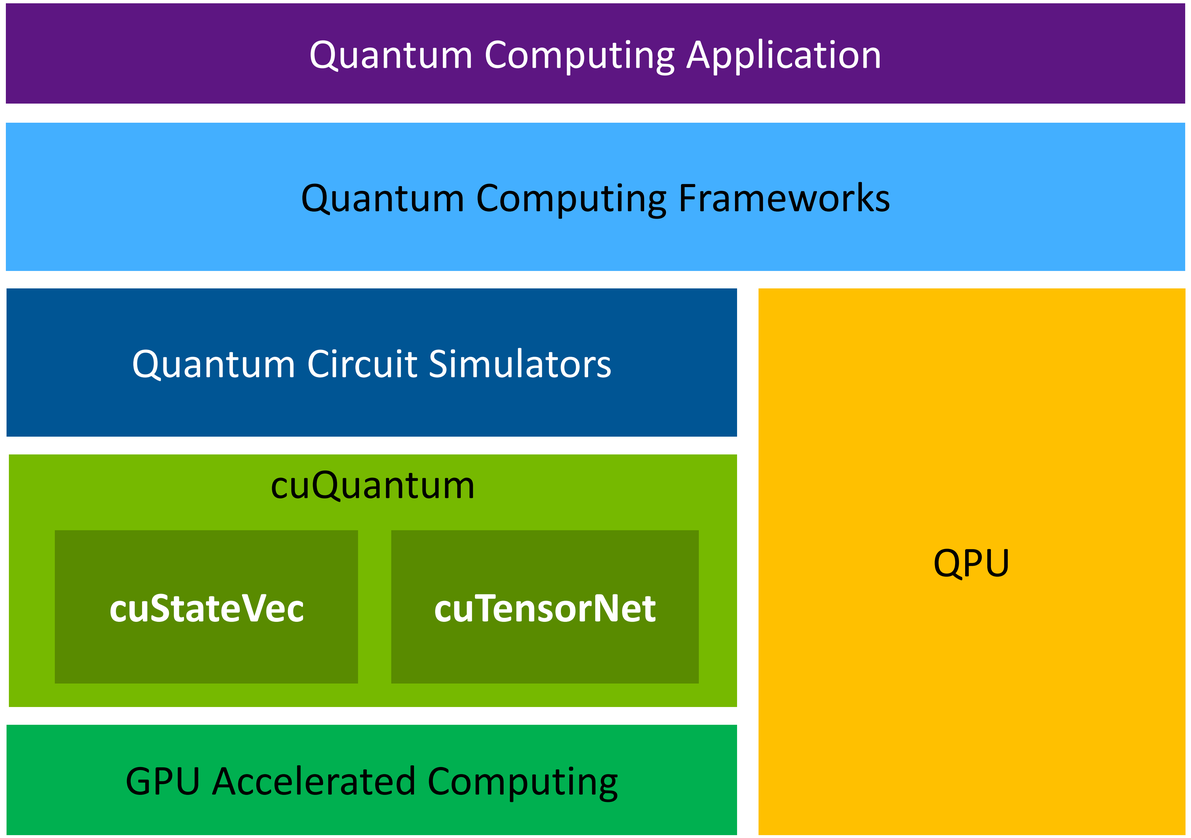
- APIs are specifically designed for state vector simulators, operating ‘in-place’ to save memory usage
- Covers common use cases including:
  1) Apply gate matrix (facilitates gate fusion)
  2) Apply general permutation matrix
  3) Apply multi-qubit Pauli rotation
  4) Expectation using matrix as observable
  5) Expectation on multi-qubit Pauli basis
  6) Measurement on a Z-product basis
  7) Batched single qubit measurement
  8) Sampling
  9) State vector accessor
  10) Bit swap of the state vector index




```
custatevecStatus_t
custatevecApplyMatrix(custatevecHandle_t           handle,
                      void*                        sv,
                      cudaDataType_t               svDataType,
                      const uint32_t               nIndexBits,
                      const void*                  matrix,
                      cudaDataType_t               matrixDataType,
                      custatevecMatrixLayout_t     layout,
                      const int32_t                adjoint,
                      const int32_t*               targets,
                      const uint32_t               nTargets,
                      const int32_t*               controls,
                      const uint32_t               nControls,
                      custatevecComputeType_t      computeType,
                      void*                        extraWorkspace,
                      size_t                       extraWorkspaceSizeInBytes);
```

cuQuantum

- **Platform for quantum computing research**
  - Accelerate Quantum Circuit Simulators on GPUs
  - Simulate ideal or noisy qubits
  - Enable algorithms research with scale and performance not possible on quantum hardware, or on simulators today
- **cuQuantum General Access available now**
  - Integrated into leading quantum computing frameworks Cirq, Qiskit, and Pennylane
  - C and Python APIs
  - Available today at developer.nvidia.com/cuquantum



Cirq Qiskit Pennylane

XACC QUANTUM FRAMEWORK QIBO

Stefano Carrazza - Quantum Simulation with Just-in-time Compilation [S41366]

Agenda

Overview of state vector simulation

Acceleration on single device simulations

Acceleration on distributed simulations

State Vector

$$|\psi\rangle = \sum_{p=0,\, i_p \subseteq \{0,1\}}^{N-1} |i_{N-1}\rangle \otimes |i_{N-2}\rangle \otimes |i_{N-3}\rangle \otimes \cdots \otimes |i_0\rangle \; \alpha_{\underbrace{i_{N-1} i_{N-2} i_{N-3} \ldots i_0}}$$

Binary representation of the index of state vector

Brute-force exact simulation, basic simulation method

State vector size: 2^{nQubits}

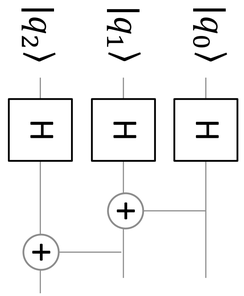
- Quantum state is defined as a vector with elements, $\alpha_{i_{N-1} i_{N-2} i_{N-3} \ldots i_0}$
- Has 2N-dimension Hilbert space. Th entity is an N-dimensional tensor

Simulation model

State vector

Qubits

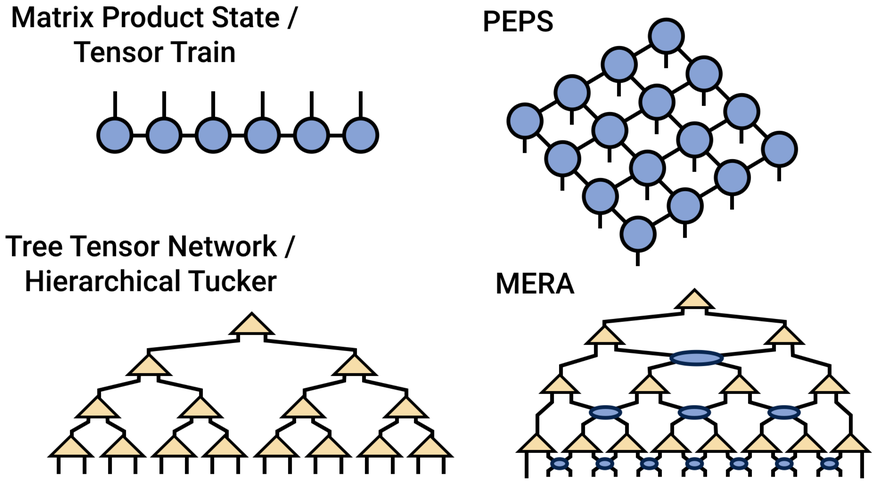
Quantum circuit



Bits in state vector index

b_2	b_1	b_0	State vector	
0	0	0	$\alpha[000]$	$\|0\rangle \otimes \|0\rangle \otimes \|0\rangle$
0	0	1	$\alpha[001]$	$\|0\rangle \otimes \|0\rangle \otimes \|1\rangle$
0	1	0	$\alpha[010]$	$\|0\rangle \otimes \|1\rangle \otimes \|0\rangle$
0	1	1	$\alpha[011]$	$\|0\rangle \otimes \|1\rangle \otimes \|1\rangle$
1	0	0	$\alpha[100]$	$\|1\rangle \otimes \|0\rangle \otimes \|0\rangle$
1	0	1	$\alpha[101]$	$\|1\rangle \otimes \|0\rangle \otimes \|1\rangle$
1	1	0	$\alpha[110]$	$\|1\rangle \otimes \|1\rangle \otimes \|0\rangle$
1	1	1	$\alpha[111]$	$\|1\rangle \otimes \|1\rangle \otimes \|1\rangle$

Tensor network



From tensornetwork.org

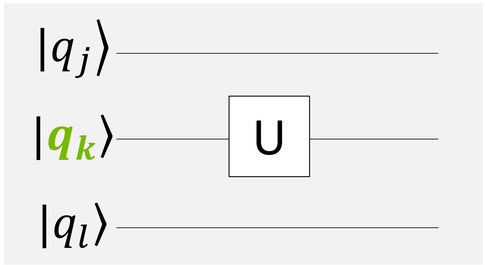
Gate matrix application

- Quantum logic gate

  2^n x 2^n unitary matrix
  n: number of target qubits

$$U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$$

- Gate application



$$\begin{pmatrix} \alpha'_{i_{N-1}i_{N-2}\ldots\mathbf{0}_k\ldots i_0} \\ \alpha'_{i_{N-1}i_{N-2}\ldots\mathbf{1}_k\ldots i_0} \end{pmatrix} = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} \begin{pmatrix} \alpha_{i_{N-1}i_{N-2}\ldots\mathbf{0}_k\ldots i_0} \\ \alpha_{i_{N-1}i_{N-2}\ldots\mathbf{1}_k\ldots i_0} \end{pmatrix}$$

$$\begin{pmatrix} \alpha'_{1\ldots\mathbf{0}\ldots11} \\ \alpha'_{1\ldots\mathbf{0}\ldots10} \\ \alpha'_{1\ldots\mathbf{0}\ldots01} \\ \alpha'_{1\ldots\mathbf{0}\ldots00} \\ \alpha'_{0\ldots\mathbf{0}\ldots11} \\ \alpha'_{0\ldots\mathbf{0}\ldots10} \\ \alpha'_{0\ldots\mathbf{0}\ldots01} \\ \alpha'_{0\ldots\mathbf{0}\ldots00} \\ \alpha'_{0\ldots\mathbf{1}\ldots00} \end{pmatrix} = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} \begin{pmatrix} \alpha_{1\ldots\mathbf{0}\ldots11} \\ \alpha_{1\ldots\mathbf{0}\ldots10} \\ \alpha_{1\ldots\mathbf{0}\ldots01} \\ \alpha_{1\ldots\mathbf{0}\ldots00} \\ \alpha_{0\ldots\mathbf{0}\ldots11} \\ \alpha_{0\ldots\mathbf{0}\ldots10} \\ \alpha_{0\ldots\mathbf{0}\ldots01} \\ \alpha_{0\ldots\mathbf{0}\ldots00} \\ \alpha_{0\ldots\mathbf{1}\ldots00} \end{pmatrix}$$

2^{N-1}

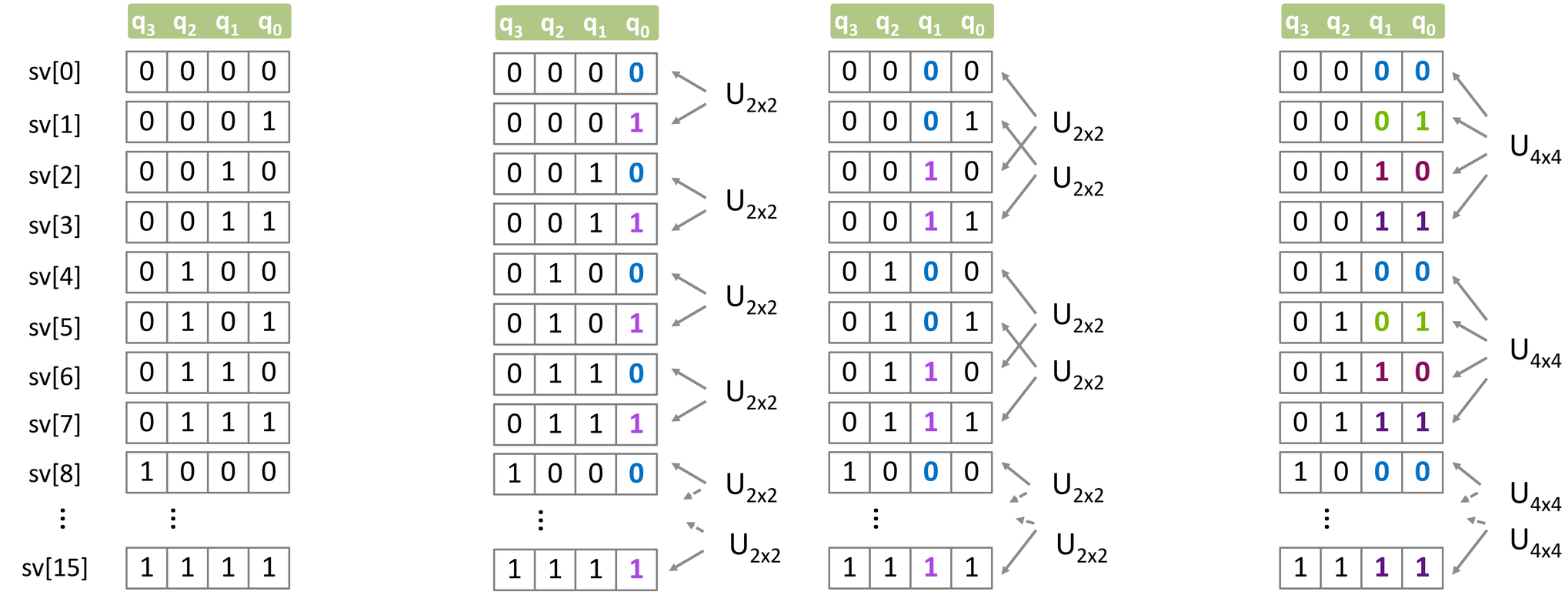
Superposition

Quantum computer: Quantum parallelism

GPU: Massively-parallel computation
(Billions of parallel computation)

Gate Application

For 4-qubit state vector



Qubits map to the index bits of state vector

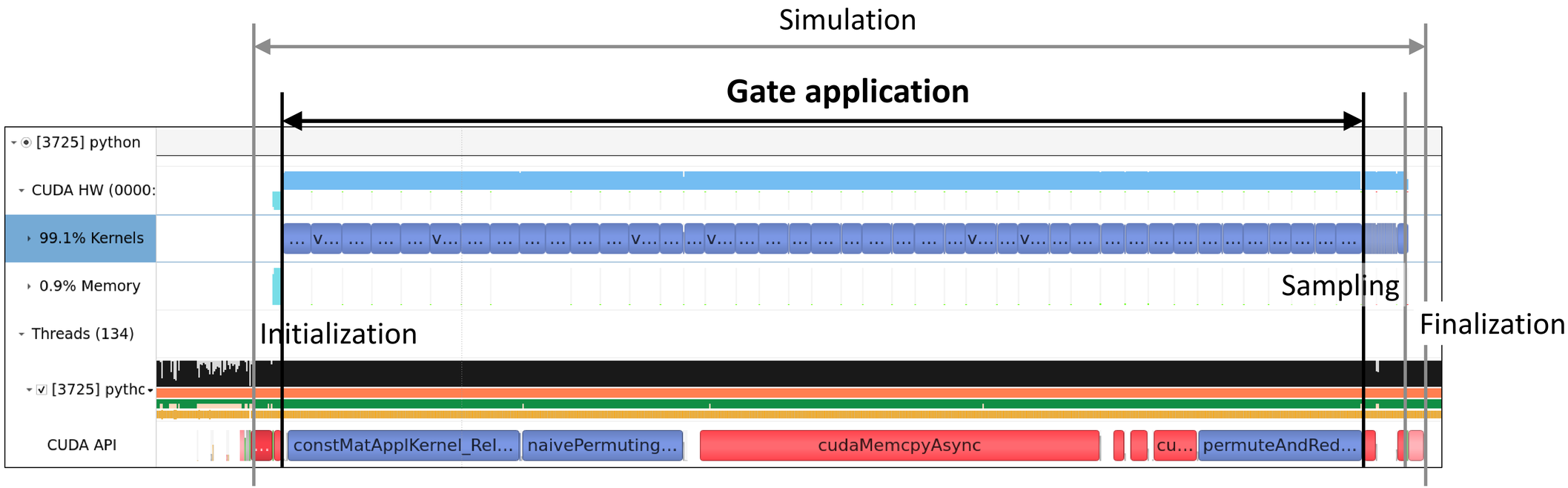
Agenda

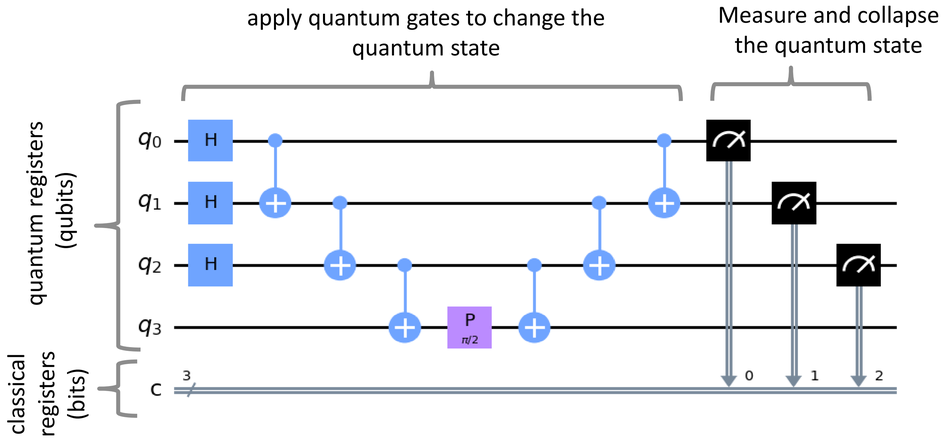
Overview of state vector simulation

Acceleration on single device simulations

Acceleration on distributed simulations

Simulation Timeline

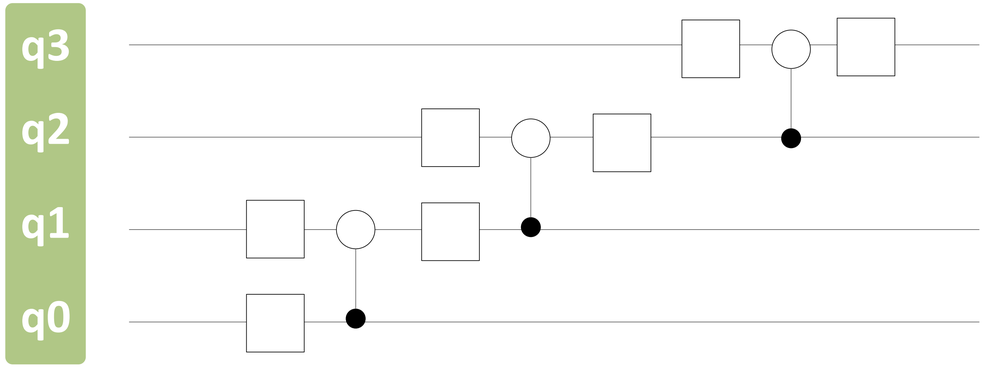




Gate Fusion

Fusing gates to multi-qubit gates

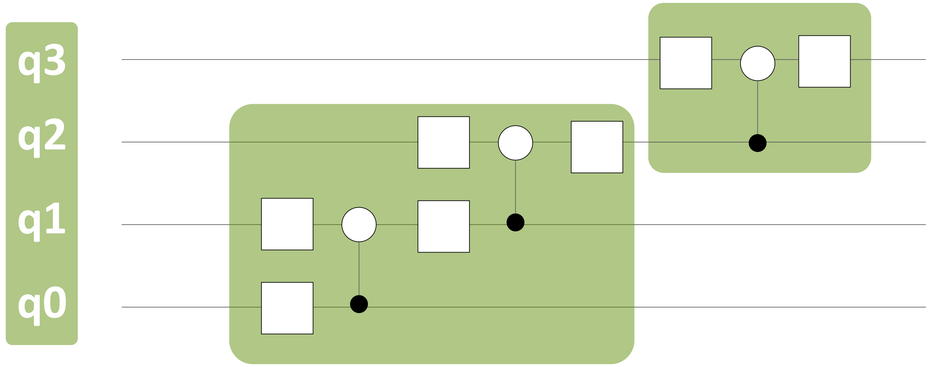
- Reduce the number of gates, and reduce the number of gate applications



Single qubit gates x7

Controlled gates x3

Simulation cost: 7 + 0.5 x 3 = **8.5**



3-qubit gate x1

2-qubit gate x1

Simulation cost: 1 + 1 = **2**

Can simulation be accelerated by a factor of 4.25 times ?

Gate Fusion

Memory bound / Compute bound

NVIDIA H100 SXM

Peak FLOPS: 67 TFLOPS, Mem BW: 3.35 TB

State vector : 30 qubits, 8 GiB (complex 64)

Time to update state vector : 5.1 ms

8 GiB x 2 [Read and Write] / 3.35 [TB / s] = **5.1 ms**

Computation time

Single qubit gate application

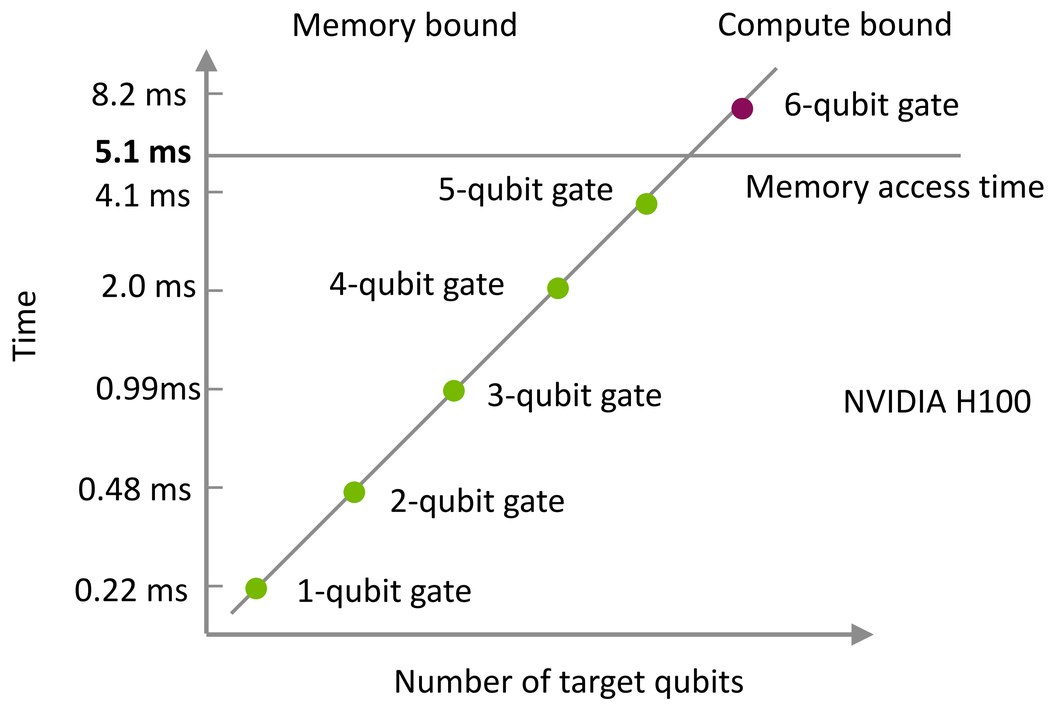
14 [FLOP / sv element] * 2^{30} / 67 [TFLOPS] = **0.22 ms**

5-qubit gate application

254 [FLOP / sv element] * 2^{30} / 67 [TFLOPS] = **4.1 ms**

6-qubit gate application

510 [FLOP / sv element] * 2^{30} / 67 [TFLOPS] = **8.2 ms**

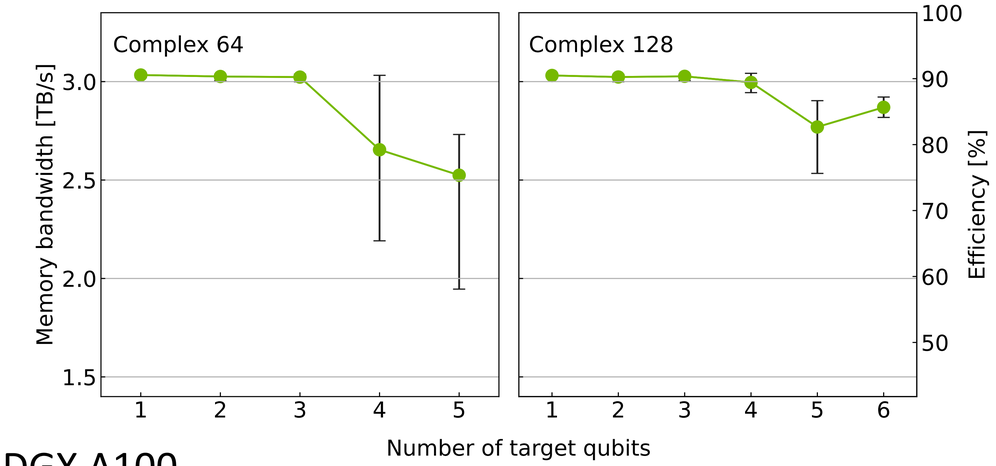


Up to 5 qubits,

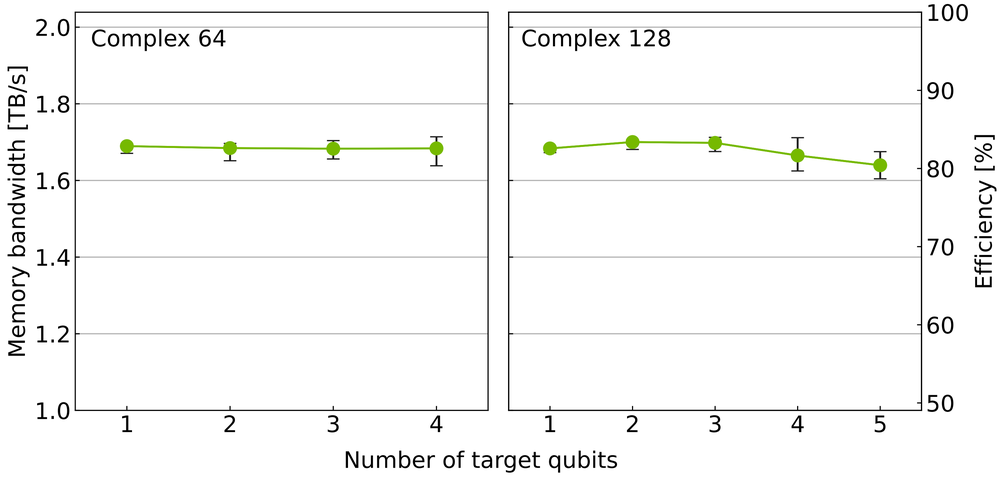
Gate application time is constant

Gate Application Performance of cuStateVec

DGX H100



DGX A100



NVIDIA H100

- Up to 5-qubit gate applications are memory-bound
- 1 ~ 3 qubit gate application
  Memory bandwidth reached the hardware limit
- H100 is still new
  We are working to improve performance for 4- and 5-qubit gate application

NVIDIA A100

- Up to 4-qubit gate applications are memory-bound
- Constantly exceeded 80% of memory bandwidth efficiency

Simulation Performance

33 qubits, c64

Circuit	Gate fusion size	Memory BW
NVIDIA H100 SXM	5 qubits	3.35 TB/s
NVIDIA A100 SXM	4 qubits	2.04 TB/s

Circuit	# Gates	NVIDIA H100 SXM		NVIDIA A100 SXM	
		# Fused gates	Simulation time [s]	# Fused gates	Simulation time [s]
QFT*	577	18	**1.21 ** **	18	2.30
QAOA*	1650	90	**4.85**	131	10.7
Quantum Volume depth=30	480	114	**6.92**	154	12.6

*Gate fusion is applied also for diagonal gate matrices

**65 times faster then CPU (qsim on 2 sockets of EPYC 7742)

Agenda

Overview of state vector simulation

Acceleration on single device simulations

Acceleration on distributed simulations

Distributed State Vector Simulation

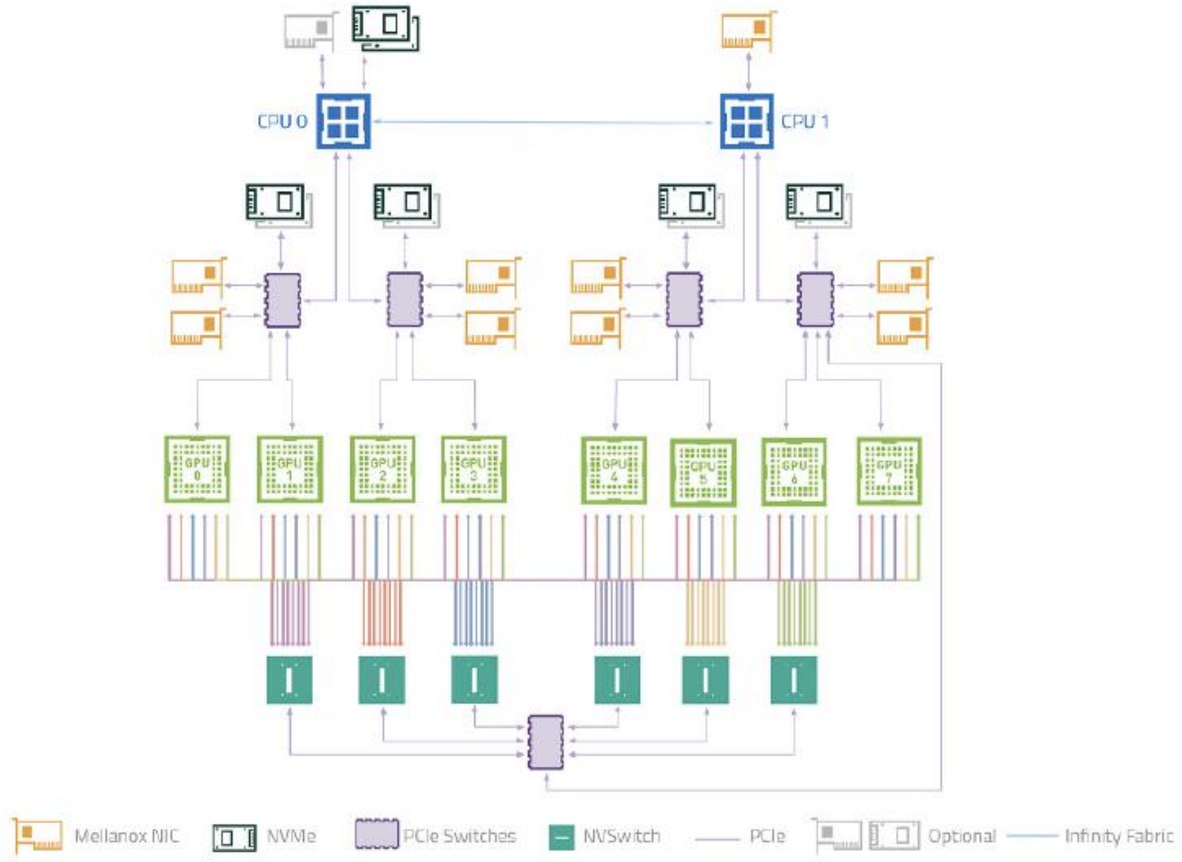
Motivation

Utilize memory in multiple GPUs and servers

- Allocate big state vector

Interconnect (DGX A100)

- NVLink / NVSwitch
  - Connect GPUs
  - 600 GB/s (Bidirectional)
- Infiniband network
  - GPU-to-GPU direct data transfer
  - 50 GB/s (Unidirectional)



* This slide has been updated after the presentation to be consistent with the results in this section. The original version is moved to the end of this material.

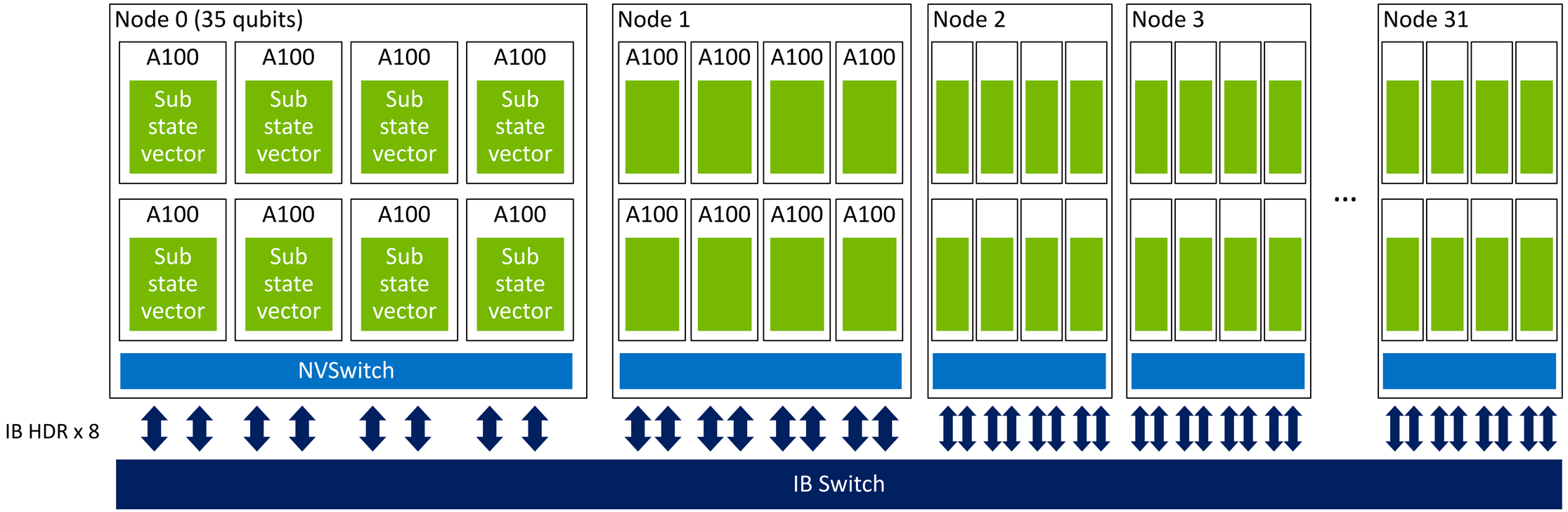
40 Qubit State Vector Distributed to 32 Nodes

Single GPU

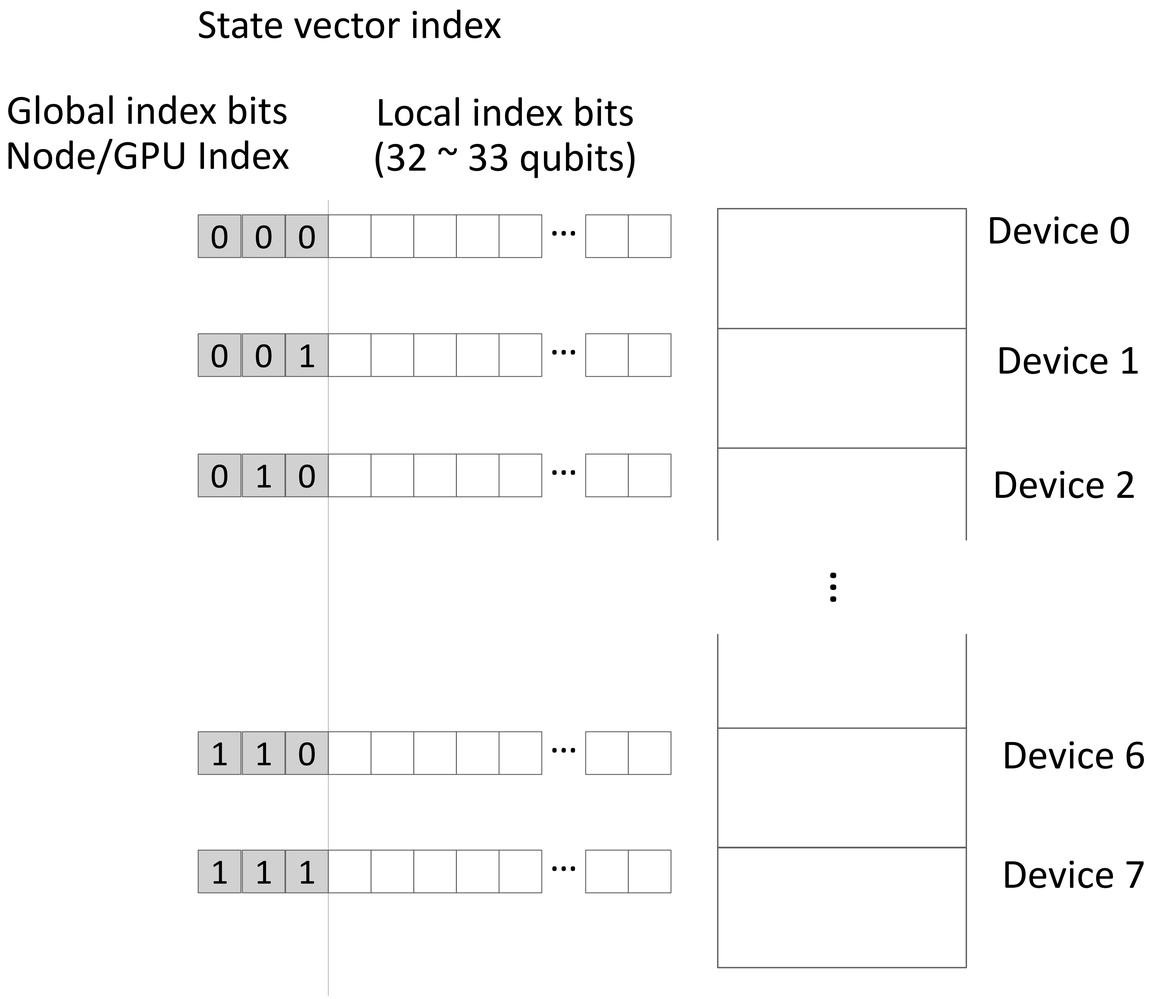
- 32 qubits (c128)
- 64 GiB = 16 bytes x 2^{32}

- Equally slice the state vector
  - Allocate a slice on each GPU
- +1 qubits by doubling # GPUs

- 40 qubits with 32 nodes
  - 32 qubits, 64 GiB in device
  - 3 qubits, 8 GPUs / node
  - 5 qubits, 32 nodes



Distributed State Vector



Upper limit of single GPU simulation

- 64 GiB / A100
- 32 qubits(c128), 33 qubits (c64)

Distribute state vector

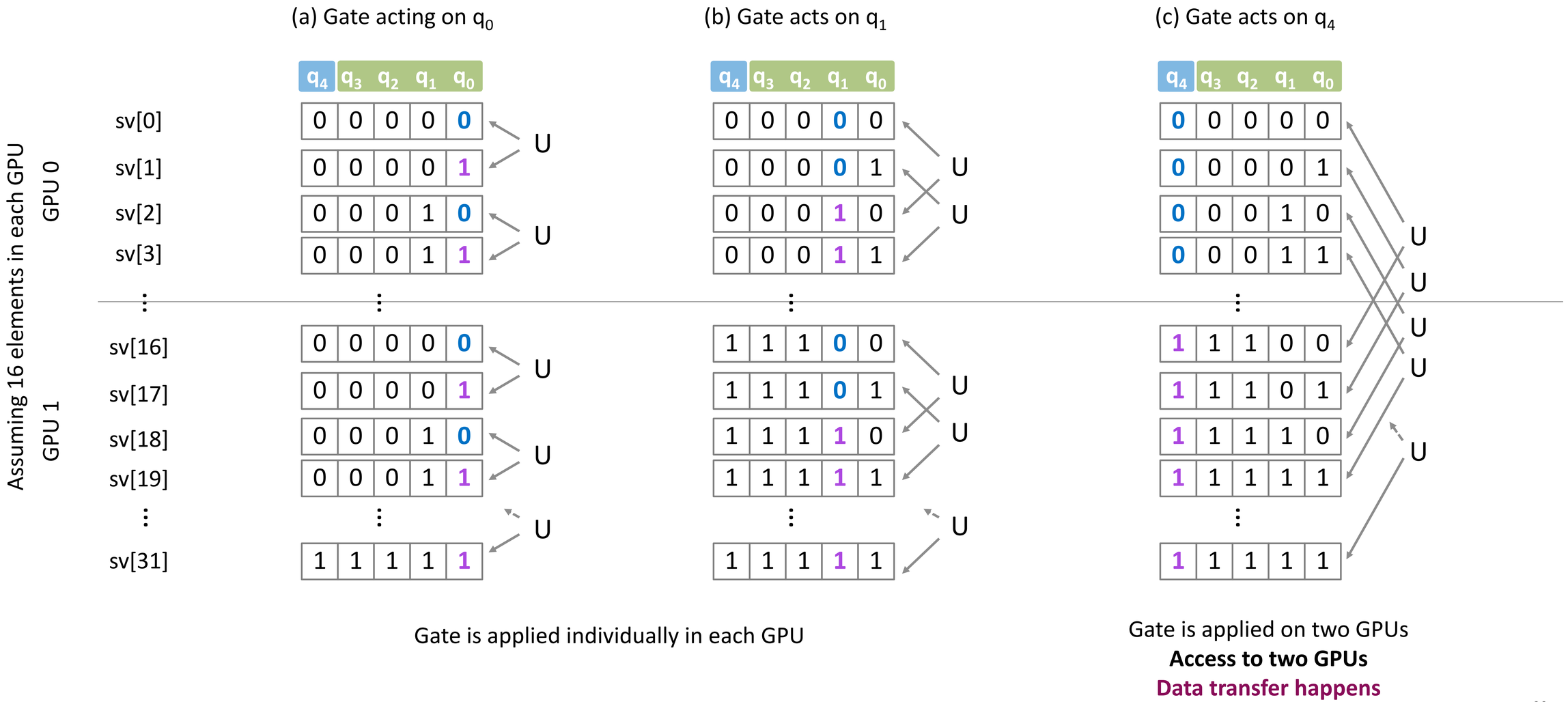
- Equally slice the state vector

  Global bits = Device idx bits

  Local bits = index bits in device

Gate Application for Distributed State Vector

Example of single qubit gate application



Gate is applied individually in each GPU

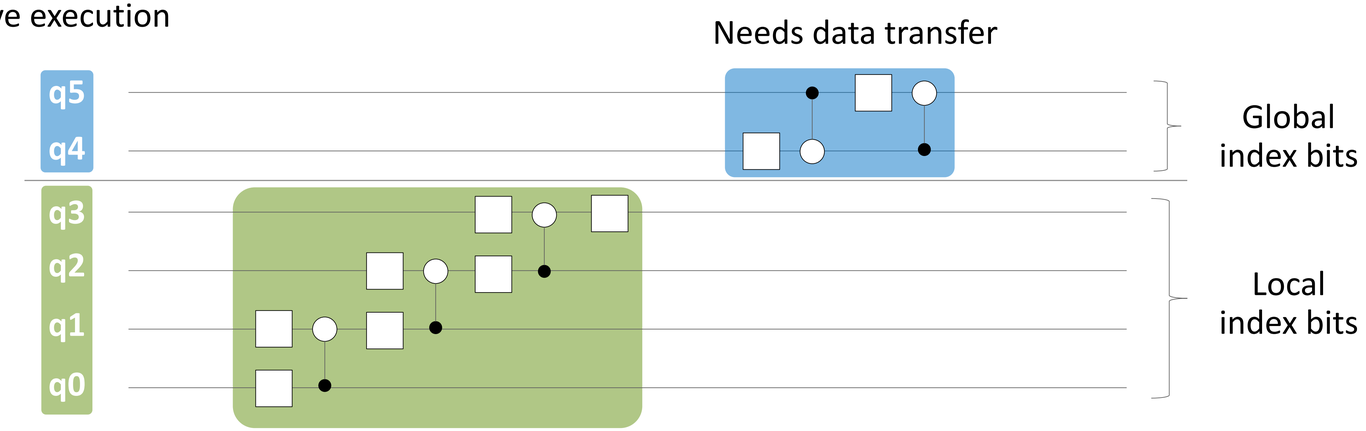
Gate is applied on two GPUs

Access to two GPUs

Data transfer happens

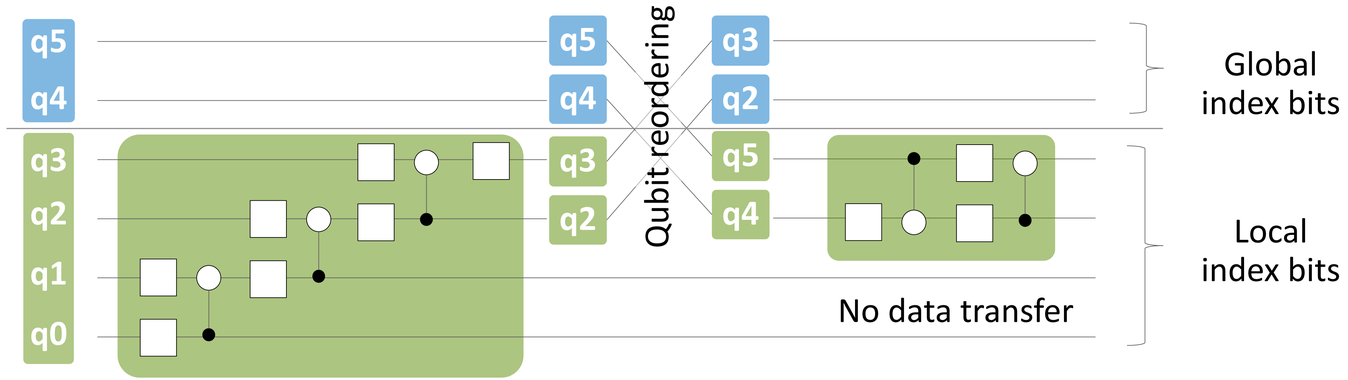
Qubit Reordering

Naive execution



- Gates in **Blue** box needs data transfers
  - Slow gate application

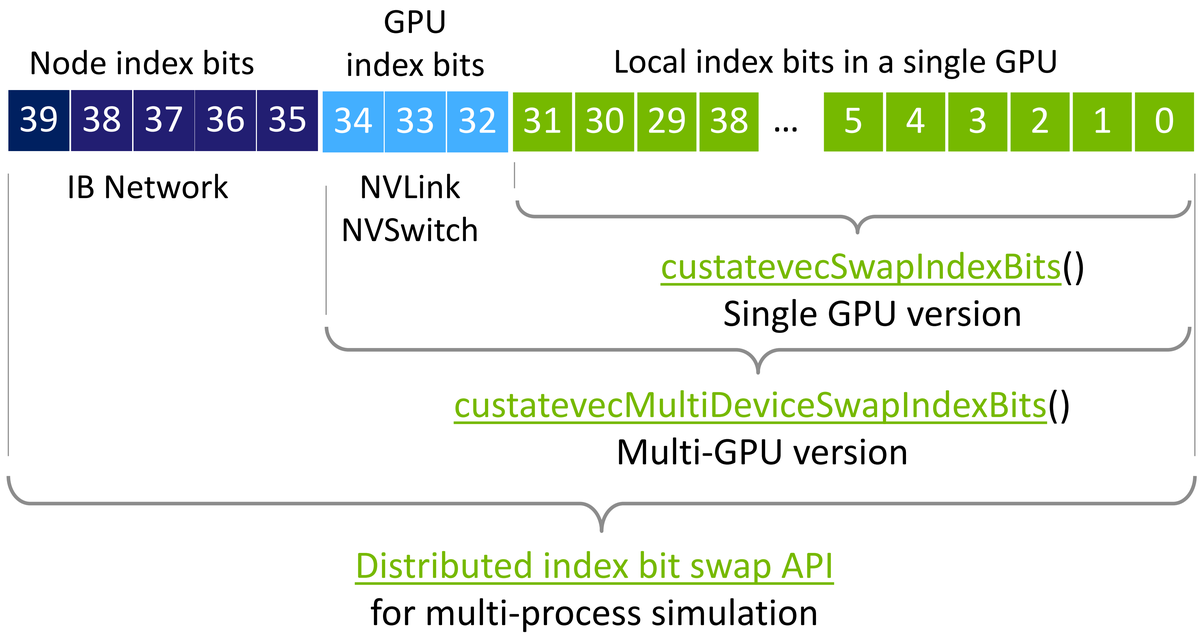
Use qubit reordering



- Move **Blue** box on local index bits
  - Fast gate aplication

Index Bit Swap API

“SwapIndexBits” API in libcustatevec

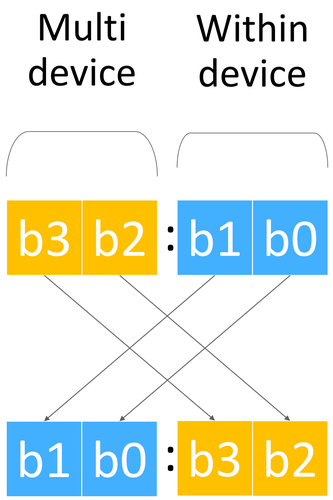


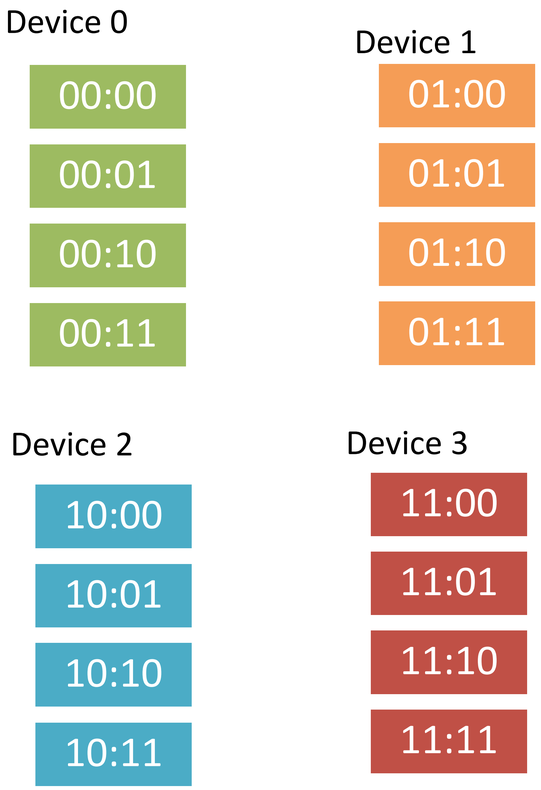
Qubit reordering:

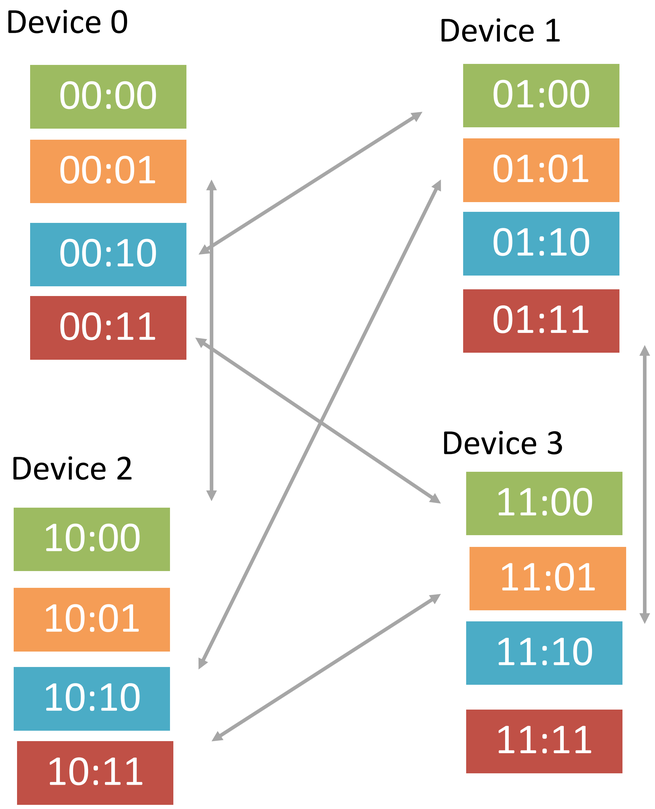
- NVLink/NVSwitch
  - 300 GB/[sec•GPU] (unidirectional)
- IB network
  - 12.5 GB/[sec•GPU] (unidirectional)

Index bit Swaps Between Multiple Devices

In-place all-to-all data transfer





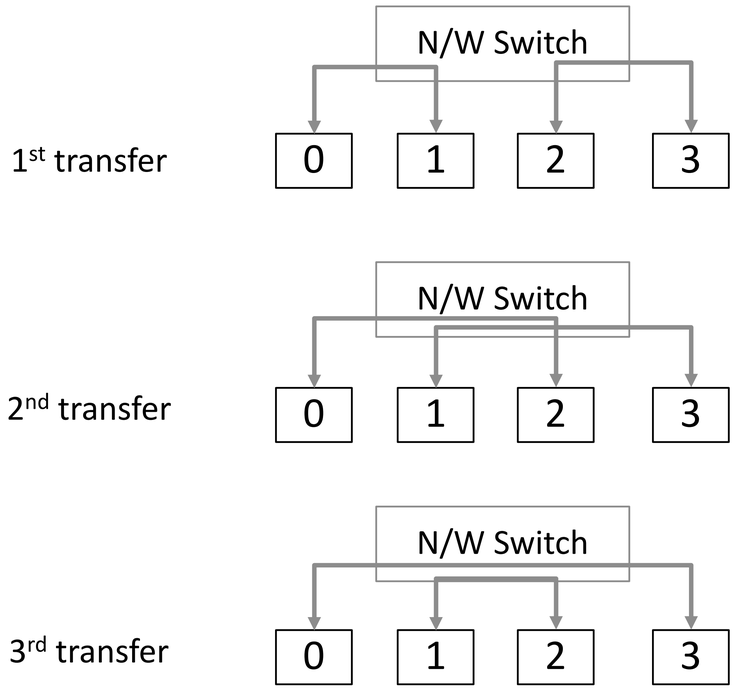


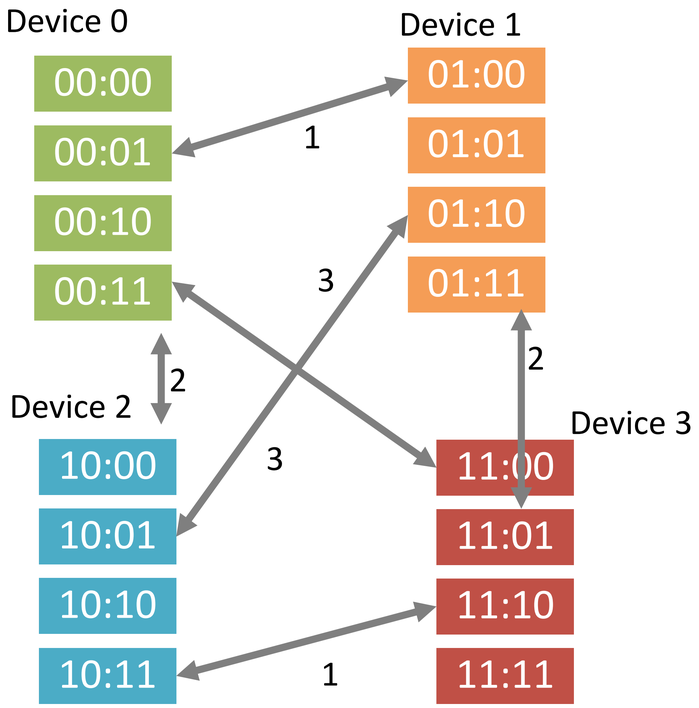
*actual bit swap patterns depends on circuits.
This example assumes that {b0, b1} and {b2, b3} are swapped.

Schedule Swaps of State Vector Segments

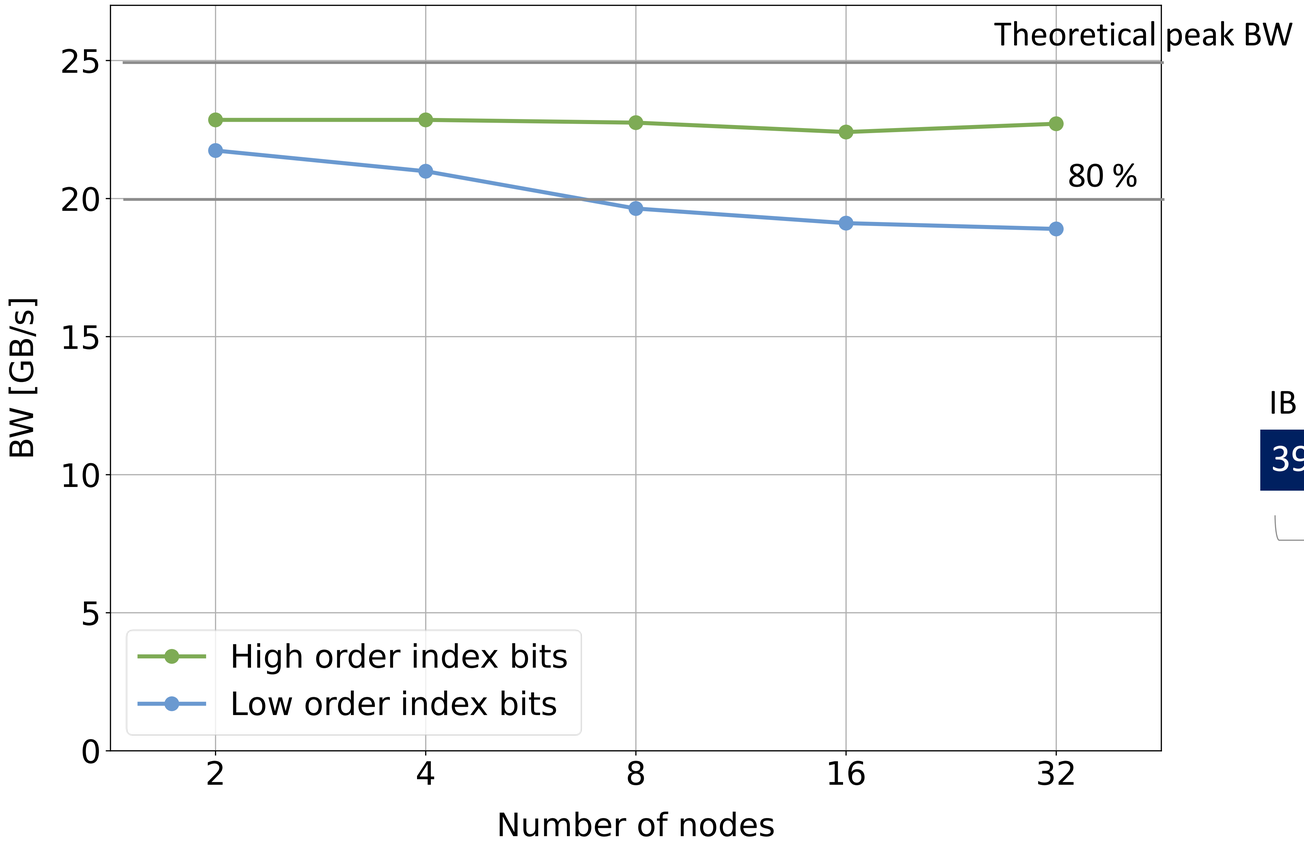
Manually handling in-place all-to-all transfer

- All-to-all transfer → “Pairwise swap of state vector segments”
- Utilize full bisection bandwidth in networks with switches.





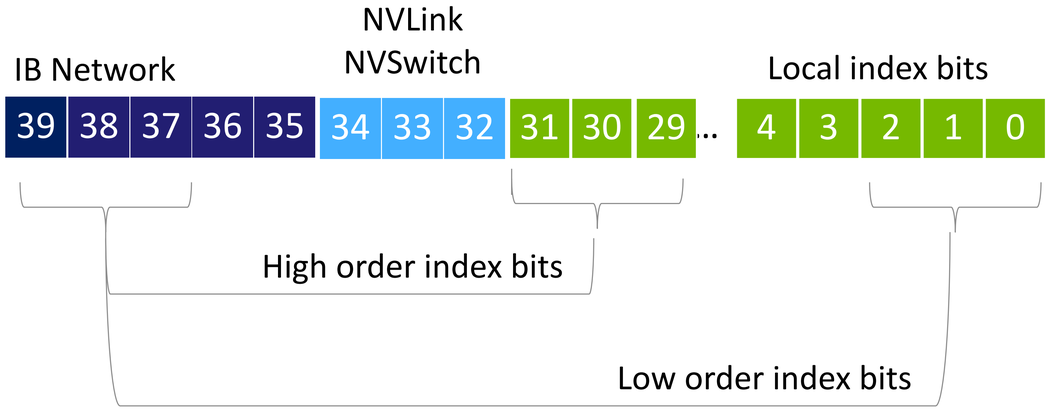
The Performance of Multi-node Index Bit Swap



40 qubit c128 state vector

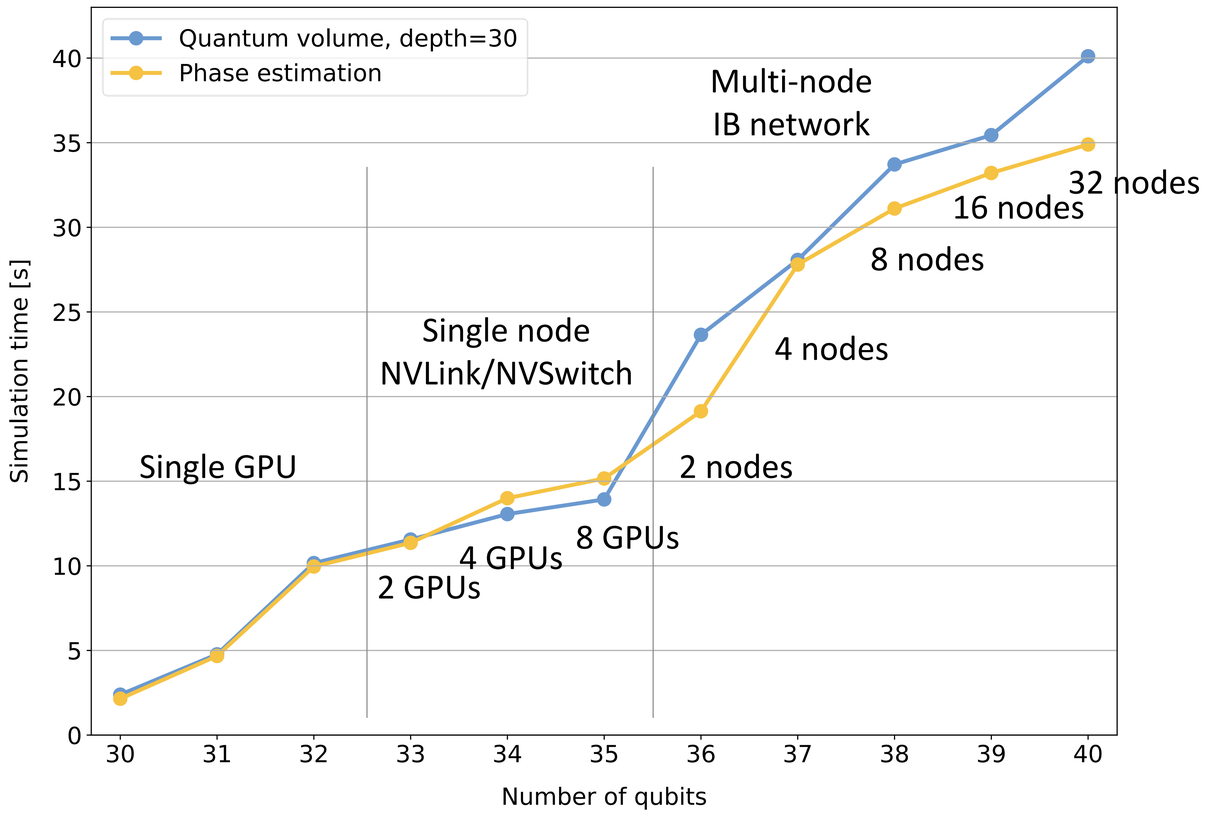
Two patterns of index bit swaps

- High order index bits: ca. 90 %
- Low order index bits: 76 ~ 87 %



Multi-node State Vector Simulator

cusvaer, cuQuantum 22.11



DGX A100 Cluster (NVIDIA SuperPOD)

Utilizing ca. 80 % of bandwidths in all components

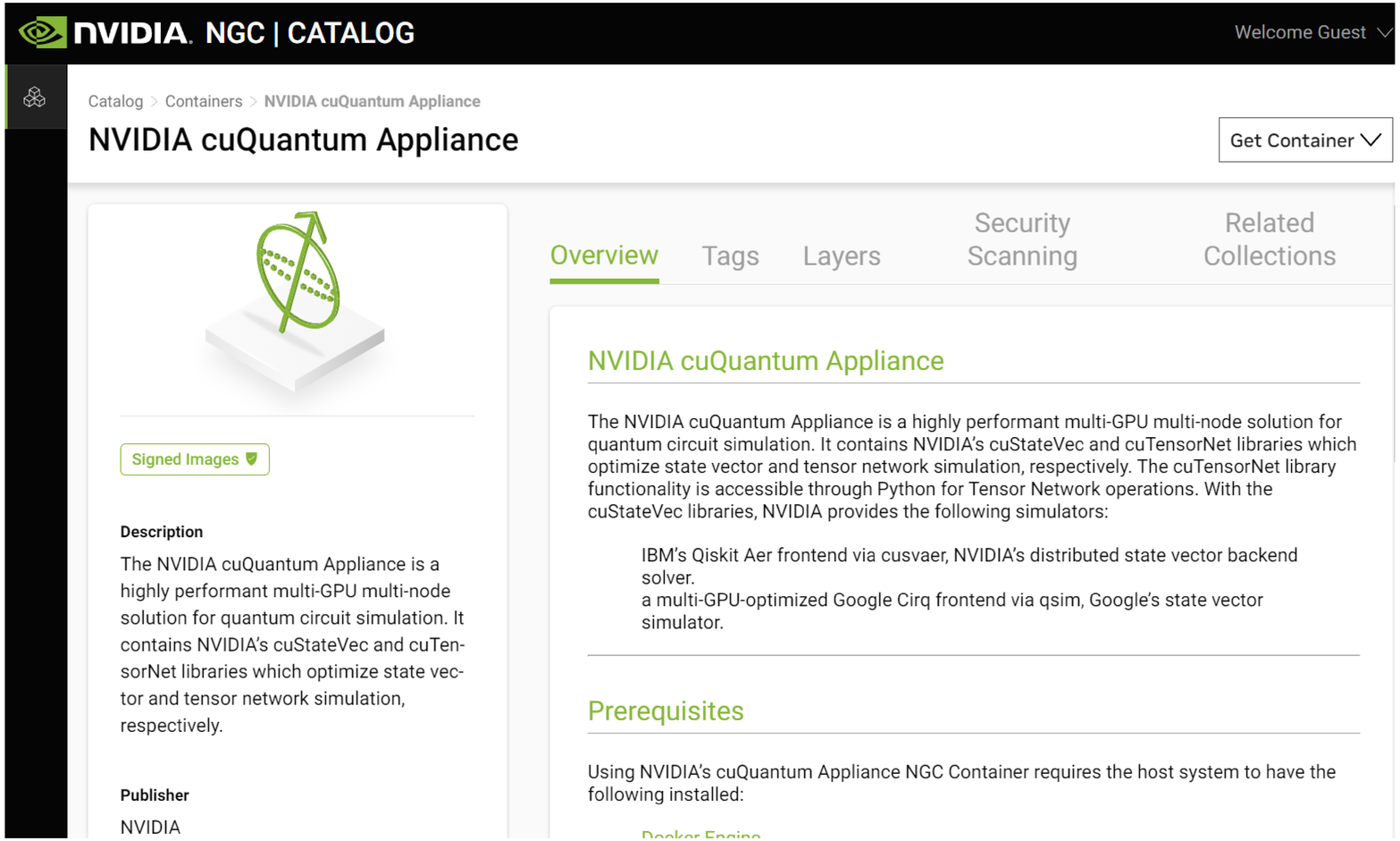
- Device memory
- NVLink/NVSwtch
- IB network

3.5x faster than the previous state-of-the-art implementation

Performance headroom

- Tensor Core and other H100 features
- Algorithmic optimizations

NVIDIA cuQuantum Appliance



- Multi-GPU version of cirq/qsim Simulator
- Multi-node version of Qiskit simulator

https://catalog.ngc.nvidia.com/orgs/nvidia/containers/cuquantum-appliance

Distributed State Vector Simulation

Motivation

Utilize memory in multiple GPUs and servers

- Allocate big state vector

Interconnect (DGX H100)

- NVLink / NVSwitch
  - Connect GPUs
  - 900 GB/s (Bidirectional)
- Infiniband network
  - GPU-to-GPU direct data transfer
  - 50 GB/s (Unidirectional)