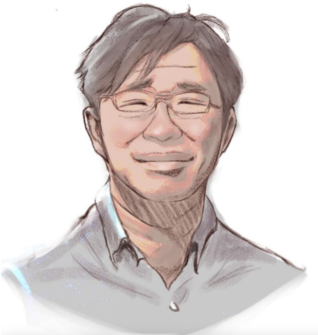
Full Optimization for Quantum Circuit Simulation

haofan2023 (灝粉2023)

Members: Yu-Cheng Lin, Alan Kuan, Ching-Wen Chen, and Chuan-Chi Wang

Advisor: Shih-Hao Hung (洪士灝教授)

Performance, Application and Security Laboratory (PAS Lab)



Mentors: Tian Zheng, Frank Lin, and Pika Wang (NVIDIA) 👍

Two Key Features for Our SoTA Quantum Circuit simulator

Arbitrary Quantum Circuit Optimization

Cache Optimization

Super Block Finding Algorithm

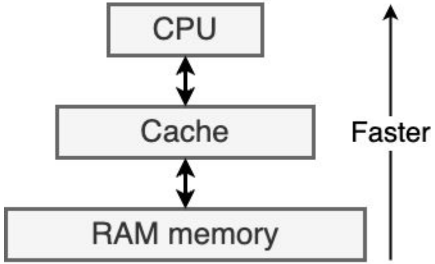
Specific Quantum Circuit Optimization for QAOA

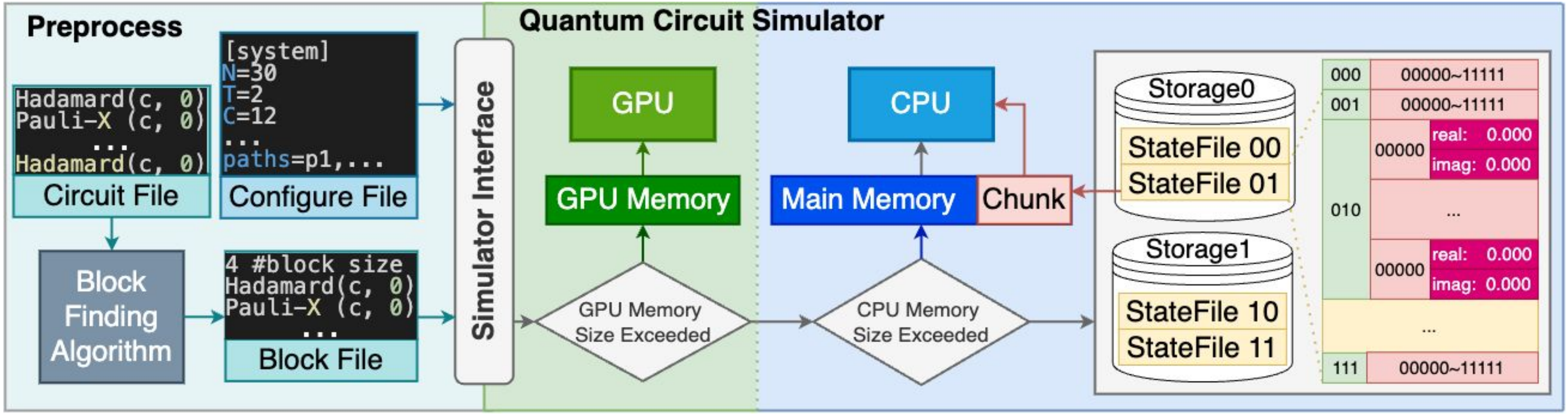
Launch Control

Rotation Compression

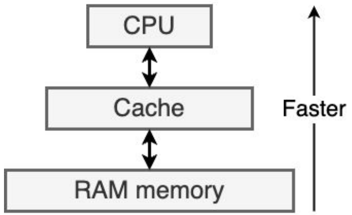
Workflow and Cache Optimization

- Cache optimization and qubit extension for the state vector-based quantum circuit simulator on CPU and GPU.

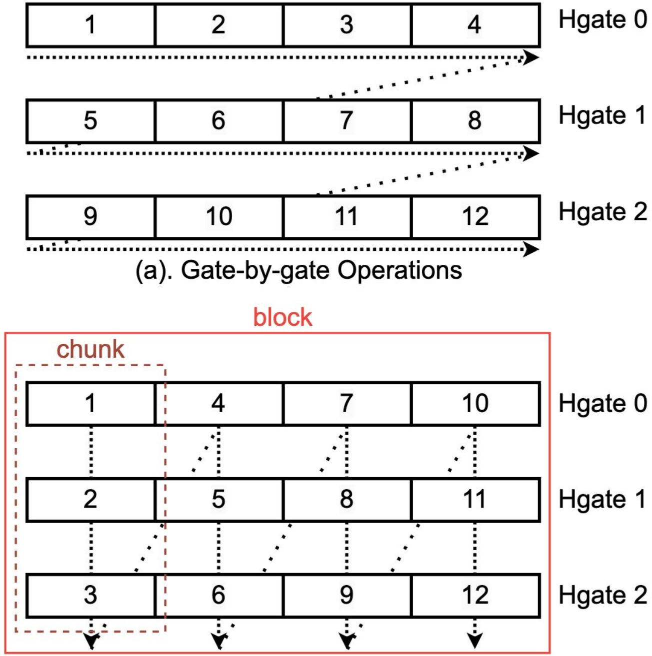




Super Block Access Pattern

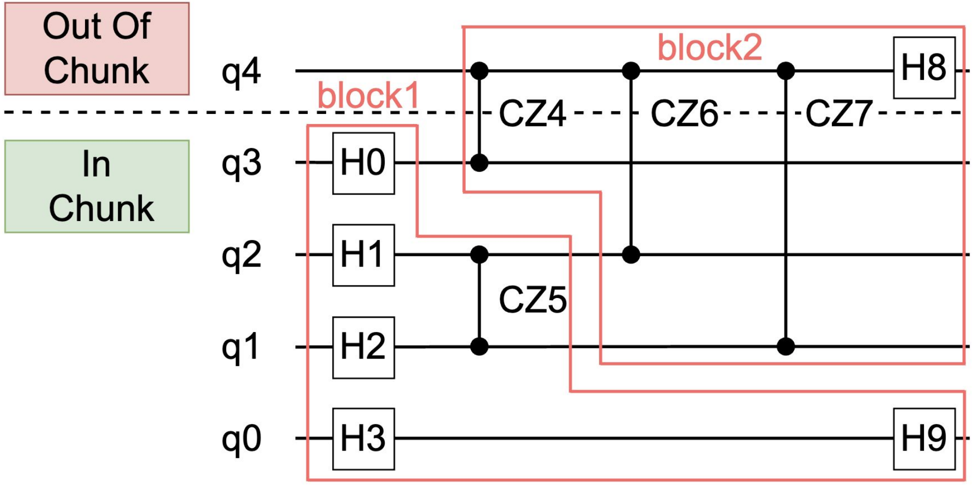


- Improve data locality and reduce the execution time



(a). Gate-by-gate Operations

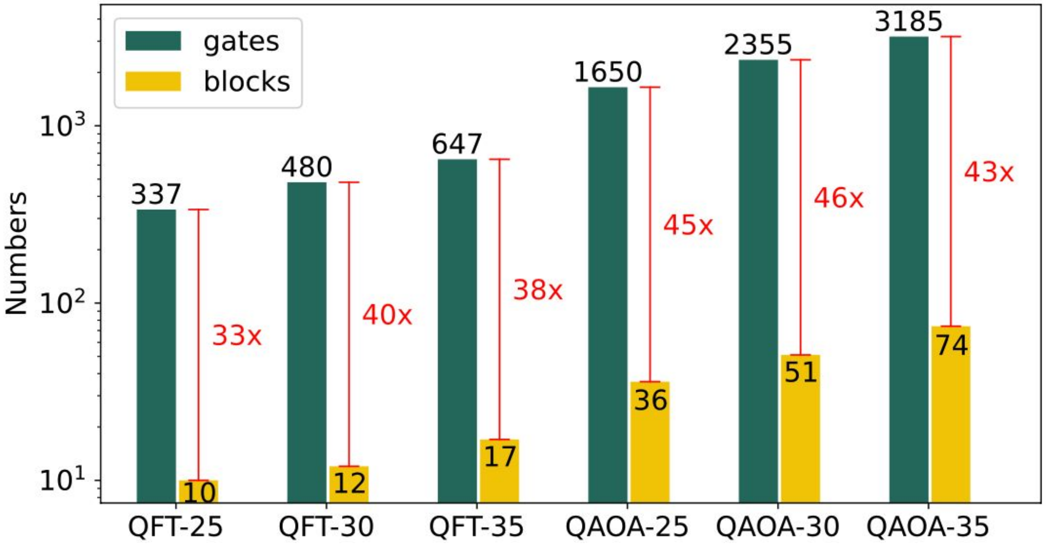
(b). Chunk-by-chunk Operations



Super Block Finding Algorithm Efficiency

- **QFT**: a fundamental quantum algorithm that efficiently transforms the representation to the frequency domain.
  - 33x ~ 38x

- **QAOA**: a quantum computing algorithm designed to tackle combinatorial optimization problems by leveraging variational principles and parameterized quantum circuits.
  - 43x ~ 46x



Profiler Output: Cache Miss Rate (CMR)

	QFT		5-level QAOA	
Simulator	Time	CMR	Time	CMR
GPU	0.13	24.7% (-)	2.02	24.7% (-)
GPU_{sub}	0.09	11.5% (2.2x)	1.20	5.2% (4.8x)
MEM	0.98	16.9% (-)	15.32	16.7% (-)
MEM_{sub}	0.44	8.6% (2.0x)	3.42	3.0% (5.6x)
SSD	4.75	10.5% (-)	22.20	10.5% (-)
SSD_{sub}	0.89	8.5% (1.2x)	5.19	5.5% (1.9x)

subscript *'sub'* is used for applying the cache optimization for super block.

Experimental Environment in Our Local Machine

Name	Description
CPU	AMD Ryzen Threadripper PRO 3995WX 64-Core
GPU	NVIDIA GeForce RTX 4090 16,3884-Core
RAM	Kingston 256 GB (8*32GB) DDR4 3600MHz
Storage	Kingston 16 TB (8*2TB on PCIe 4.0 bus)
OS	Ubuntu 20.04 LTS (kernel version 5.15.0-58-generic)
CUDA	CUDA Toolkit version 12.1.105
Compiler	GCC version 9.4.0
API	OpenMP version 4.5

Gate Benchmark

unit: s

		GPU				CPU				
	Qubit	*QuEST*	*cuQuantum*	*GPU*	*GPU_{sub}*	*QuEST*	*MEM*	*MEM_{sub}*	*SSD*	*SSD_{sub}*
H gate	23	0.007	0.007	0.007	0.005 (1.4x)	0.013	0.012	0.012 (1.1x)	0.043	0.020 (0.4x)
	24	0.015	0.014	0.014	0.010 (1.4x)	0.042	0.041	0.025 (1.7x)	0.103	0.058 (0.6x)
	25	0.030	0.029	0.029	0.022 (1.4x)	0.232	0.227	0.084 (2.8x)	0.287	0.167 (1.4x)
	26	0.063	0.060	0.061	0.045 (1.4x)	0.489	0.487	0.175 (2.8x)	0.584	0.341 (1.4x)
	27	0.131	0.126	0.126	0.094 (1.4x)	1.015	1.011	0.417 (2.4x)	1.169	0.701 (1.4x)
	28	0.272	0.261	0.261	0.197 (1.4x)	2.092	2.085	0.842 (2.5x)	2.344	1.445 (1.5x)
	29	0.564	0.541	0.541	0.413 (1.4x)	4.367	4.303	1.684 (2.6x)	4.770	2.988 (1.5x)
	30	1.157	1.119	1.119	0.863 (1.3x)	8.943	8.972	3.527 (2.5x)	9.934	6.275 (1.4x)
	31	-	-	-	-	18.404	18.602	7.180 (2.6x)	21.460	13.906 (1.3x)
	32	-	-	-	-	38.046	37.765	14.241 (2.7x)	81.186	30.385 (1.3x)
	33	-	-	-	-	78.533	77.768	29.874 (2.6x)	156.876	71.691 (1.0x)
	34	-	-	-	-	-	-	-	315.845	148.602 (∞)
	35	-	-	-	-	-	-	-	644.295	308.866 (∞)
	36	-	-	-	-	-	-	-	1,612.000	629.316 (∞)
	37	-	-	-	-	-	-	-	2,787.740	1,415.690 (∞)
	38	-	-	-	-	-	-	-	5,680.410	2,778.510 (∞)
	39	-	-	-	-	-	-	-	11,616.501	5,861.420 (∞)
	SU	-	-	-	1.39x	-	-	2.39x	-	∞

up to
39-qubit

SU: Average Speedup

Quantum Circuit Simulation

GPU Speedup: **1.6x**

CPU Speedup: **5.5x**

CPU with SSD Speedup: **3.3x**

unit: s (double)

		GPU				CPU				
	Qubit	*QuEST*	*cuQuantum*	*GPU*	*GPU_{opt}*	*QuEST*	*MEM*	*MEM_{opt}*	*SSD*	*SSD_{opt}*
QFT	24	0.064	0.101	0.057	0.044 (1.45x)	0.196	0.305	0.163 (1.20x)	1.822	0.305 (0.64x)
	27	0.673	0.995	0.600	0.448 (1.50x)	6.583	5.895	1.745 (3.77x)	20.837	3.101 (2.12x)
	30	6.545	9.676	5.758	4.314 (1.52x)	59.952	55.158	16.358 (3.66x)	187.148	26.469 (2.26x)
	33	-	-	-	-	544.946	514.077	188.558 (2.89x)	2,848.325	431.315 (1.26x)
	36	-	-	-	-	-	-	-	26,568.467	3,813.630 (∞)
	39	-	-	-	-	-	-	-	243,090.020	36,980.132 (∞)
5-level QAOA	24	0.889	0.890	0.938	0.553 (1.61x)	2.114	3.468	1.579 (1.34x)	8.827	2.259 (0.94x)
	27	8.952	8.962	9.369	5.594 (1.60x)	72.859	72.667	13.132 (5.55x)	96.762	22.426 (3.25x)
	30	87.987	88.052	91.666	55.207 (1.59x)	707.322	718.330	153.065 (4.62x)	878.957	211.956 (3.34x)
	33	-	-	-	-	6,806.290	6,897.180	1,472.740 (4.62x)	12,788.421	2,273.152 (2.99x)
	36	-	-	-	-	-	-	-	133,461.023	22,307.930 (∞)
	39	-	-	-	-	-	-	-	1,226,450.254	212,438.108 (∞)

Quantum Circuit Simulation

GPU Speedup: **1.6x**

CPU Speedup: **5.5x**

CPU with SSD Speedup: **3.3x**

unit: s (double)

		GPU				CPU				
	Qubit	*QuEST*	*cuQuantum*	*GPU*	*GPU$_{opt}$*	*QuEST*	*MEM*	*MEM$_{opt}$*	*SSD*	*SSD$_{opt}$*
QFT	24	0.064	0.101	0.057	0.044 (1.45x)	0.196	0.305	0.163 (1.20x)	1.822	0.305 (0.64x)
	27	0.673	0.995	0.600	0.448 (1.50x)	6.583	5.895	1.745 (3.77x)	20.837	3.101 (2.12x)
	30	6.545	9.676	5.758	4.314 (1.52x)	59.952	55.158	16.358 (3.66x)	187.148	26.469 (2.26x)
	33	-	-	-	-	544.946	514.077	188.558 (2.89x)	2,848.325	431.315 (1.26x)
	36	-	-	-	-	-	-	-	26,568.467	3,813.630 (∞)
	39	-	-	-	-	-	-	-	243,090.020	36,980.132 (∞)
5-level QAOA	24	0.889	0.890	0.938	0.553 (1.61x)	2.114	3.468	1.579 (1.34x)	8.827	2.259 (0.94x)
	27	8.952	8.962	9.369	5.594 (1.60x)	72.859	72.667	13.132 (5.55x)	96.762	22.426 (3.25x)
	30	87.987	88.052	91.666	55.207 (1.59x)	707.322	718.330	153.065 (4.62x)	878.957	211.956 (3.34x)
	33	-	-	-	-	6,806.290	6,897.180	1,472.740 (4.62x)	12,788.421	2,273.152 (2.99x)
	36	-	-	-	-	-	-	-	133,461.023	22,307.930 (∞)
	39	-	-	-	-	-	-	-	1,226,450.254	212,438.108 (∞)

Can we continually optimize it?

Specific Quantum Circuit Optimization for QAOA

Arbitrary Quantum Circuit Optimization

Cache Optimization

Super Block Finding Algorithm

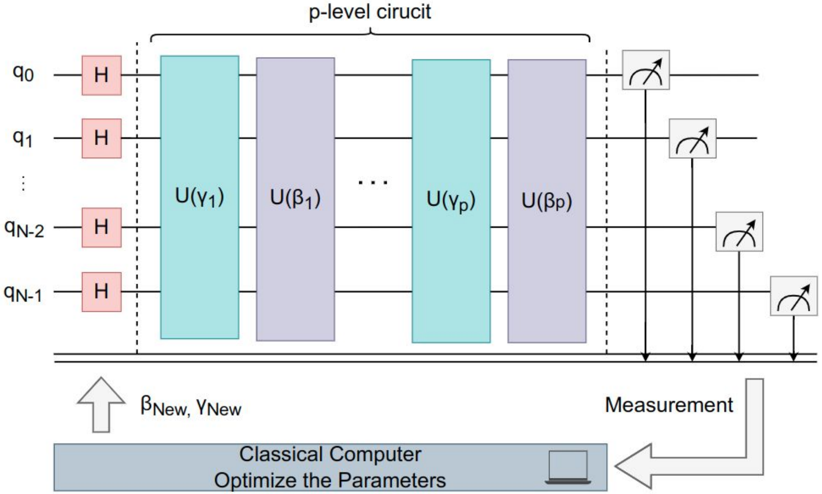
Specific Quantum Circuit Optimization for QAOA

Launch Control

Rotation Compression

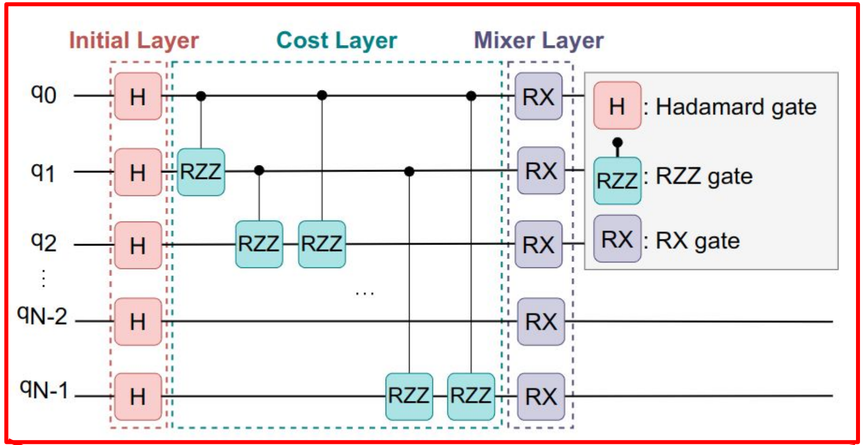
Quantum Approximate Optimization Algorithm (QAOA)

- QAOA is one of the quantum algorithm for optimization problems. It is a hybrid approach combining classical and quantum computing.
- Parameterized Quantum Gates:
  - Quantum gates with tunable parameters.
  - Varying parameters to explore solution space efficiently.
  - In general, we can achieve $r \longrightarrow 1$ when $p \longrightarrow \infty$
    - where r is the approximation ratio (and 1 is the optimal solution).



Quantum Approximate Optimization Algorithm (QAOA) Workflow

- H gate initialization
- *p*-level optimization:
  - minimize the cost function
  - γ: cost layer
  - β: mixer layer
- Using measured state of the quantum circuit to update the next round value.
  - *C*: The cost function for the given problem.

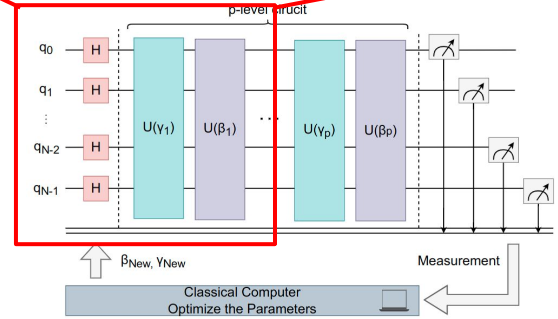


$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U(\hat{H}_M, \beta_p) U(\hat{H}_C, \gamma_p) \cdots U(\hat{H}_M, \beta_1) U(\hat{H}_C, \gamma_1) \,|+\rangle^{\otimes N}$$

$$Fp(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta} \mid C \mid \boldsymbol{\gamma}, \boldsymbol{\beta} \rangle$$

$$M_p = \min_{\boldsymbol{\gamma}, \boldsymbol{\beta}} F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$$

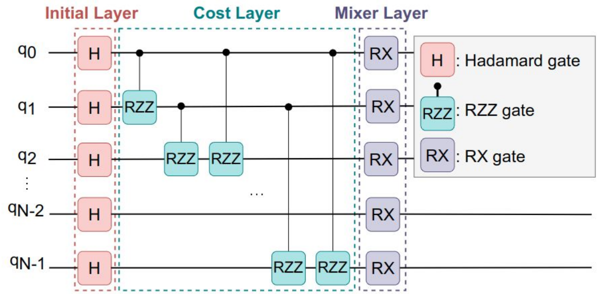
$$\boxed{r = \frac{\min_z C(z)}{M_p}}$$

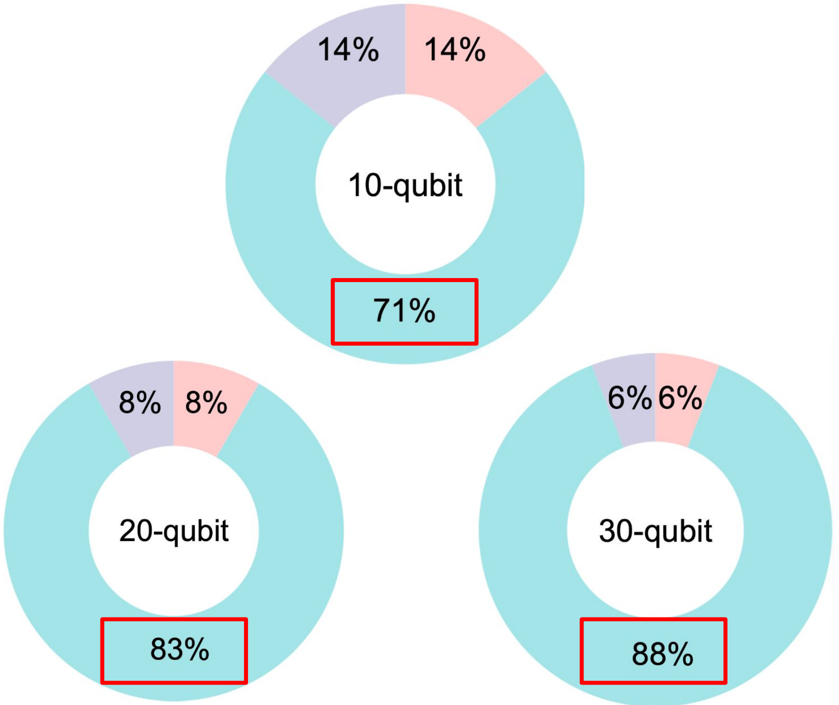


The Proportion of Numbers with Each Gate

Algorithm 1 A typical methodology for the QAOA simulation.

$stateVector \leftarrow initZeroState()$
for $0 < i < N$ **do** ▷ Initial Layer
  $stateVector \leftarrow HGate(stateVector, i)$
end for
for $0 < p < P$ **do** ▷ P-level QAOA
  for $0 < i < N$ **do** ▷ Cost Layer
    for $i < j < N$ **do**
      if $graph_{i,j}$ **then**
        $\theta = w_{i,j} * \gamma_p$
        $stateVector \leftarrow RZZGate(stateVector, i, j, \theta)$
      end if
    end for
  end for
  for $0 < i < N$ **do** ▷ Mixer Layer
    $stateVector \leftarrow RXGate(stateVector, i, \beta_p)$
  end for
end for





Rotation Compression:
Reduce the Rotation Operations form O(N^2) to O(1)

Algorithm 1 A typical methodology for the QAOA simulation.

```
stateVector ← initZeroState()
for 0 < i < N do                                   ▷ Initial Layer
    stateVector ← HGate(stateVector, i)
end for
for 0 < p < P do                                   ▷ P-level QAOA
    for 0 < i < N do                               ▷ Cost Layer
        for i < j < N do
            if graph_{i,j} then
                θ = w_{i,j} * γ_p
                stateVector ← RZZGate(stateVector, i, j, θ)
            end if
        end for
    end for
    for 0 < i < N do                               ▷ Mixer Layer
        stateVector ← RXGate(stateVector, i, β_p)
    end for
end for
```

Algorithm 2 The optimized simulation method of the cost layer.

```
function rotationCompression(stateVector, graph, w, γ)
    for 0 < b < 2^N do                             ▷ Scan 2^N state
        totRotation ← 0
        for 0 < i < N do
            for i < j < N do
                if graph_{i,j} then
                    if (b_i ⊕ b_j) == 1 then
                        totRotation ← totRotation − w_{i,j}
                    else
                        totRotation ← totRotation + w_{i,j}
                    end if
                end if
            end for
        end for
        stateVector[b] ← stateVector[b] * e^{−(1/2) iγ totRotation}
    end for
end function
```

GPU Results (Unit: ms, *float*)

- Our optimization outperforms QuEST in all cases.
  - SU_w: 6.7x; SU_u: 8.3x

Qubit	*Baseline*	Opt_w	SU_w	Opt_u	SU_u
21	17	8	2.1x	7	2.4x
22	34	17	2.0x	14	2.4x
23	121	38	3.1x	31	3.9x
24	488	96	5.1x	81	6.0x
25	1,050	196	5.4x	168	6.2x
26	2,247	398	5.7x	331	6.8x
27	4,809	811	5.9x	676	7.1x
28	10,268	1,676	6.1x	1,383	7.4x
29	21,877	3,407	6.4x	2,790	7.8x
30	46,548	6,957	6.7x	5,640	8.3x

CPU Results (Unit: ms, *float*)

- Again, our optimization consistently outperforms QuEST in all cases.
  - SU_w: 7.5x; SU_{wAVX}: 9.9x
  - SU_u: 13.7x; SU_{uAVX}: 17.1x

Qubit	*Baseline*	Opt_w	SU_w	Opt_{wAVX}	SU_{wAVX}	Opt_u	SU_u	Opt_{uAVX}	SU_{uAVX}
21	325	124	2.6x	63	**5.2x** (+2.6)	30	10.9x	21	**15.5x** (+4.6)
22	710	264	2.7x	135	**5.2x** (+2.5)	61	11.6x	42	**17.1x** (+5.5)
23	1,940	575	3.4x	308	**6.3x** (+2.9)	145	13.4x	118	**16.5x** (+3.1)
24	8,418	1,762	4.8x	1,164	**7.2x** (+2.4)	798	10.5x	790	**10.7x** (+0.2)
25	24,333	4,159	5.8x	3,020	**8.1x** (+2.3)	2,100	11.6x	1,967	**12.4x** (+0.8)
26	60,794	9,257	6.6x	6,856	**8.9x** (+2.3)	4,962	12.3x	4,693	**13.0x** (+0.7)
27	138,475	20,137	6.9x	15,126	**9.2x** (+2.3)	10,957	12.6x	10,410	**13.3x** (+0.7)
28	302,501	42,128	7.2x	31,898	**9.5x** (+2.3)	22,885	13.2x	21,653	**14.0x** (+0.8)
29	647,955	88,188	7.3x	66,667	**9.7x** (+2.4)	47,221	13.7x	44,724	**14.5x** (+0.8)
30	1,385,228	185,209	7.5x	139,663	**9.9x** (+2.4)	97,482	14.2x	92,291	**15.0x** (+0.8)

Speedup Trends in QAOA

Table 1: The elapsed time of 5-level QAOA (unit: second, double).

Qubit	CPU_{Single}	$CPU_{Mutiple}$	CPU_{Cache}	GPU_{Cache}	GPU_{All}
23	29.80	1.28 (23x)	1.28 (63x)	0.24 (120x)	0.06 (**341x**)
24	68.00	3.46 (20x)	3.46 (43x)	0.55 (123x)	0.12 (**382x**)
25	152.52	15.32 (10x)	15.31 (45x)	1.19 (127x)	0.23 (**404x**)
26	330.69	33.83 (10x)	33.83 (56x)	2.60 (126x)	0.56 (**417x**)
27	712.26	72.66 (10x)	72.66 (54x)	5.59 (127x)	1.08 (**427x**)
28	1556.87	156.52 (10x)	156.52 (54x)	11.96 (130x)	2.17 (**445x**)
29	3325.55	335.09 (10x)	335.09 (49x)	25.73 (129x)	4.45 (**451x**)
30	7226.46	718.33 (10x)	718.33 (47x)	55.20 (130x)	9.22 (**468x**)

We can get the **450x speedup** compared to CPU with a single thread.

Put It All Together to NCHC's A100 in QAOA

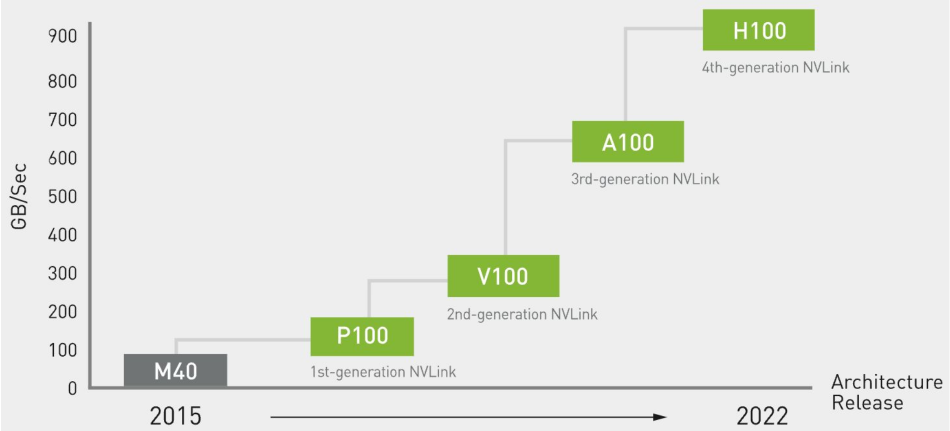
Table 2: [NCHC's GPU A100] The elapsed time of 5-level QAOA (unit: second, double).

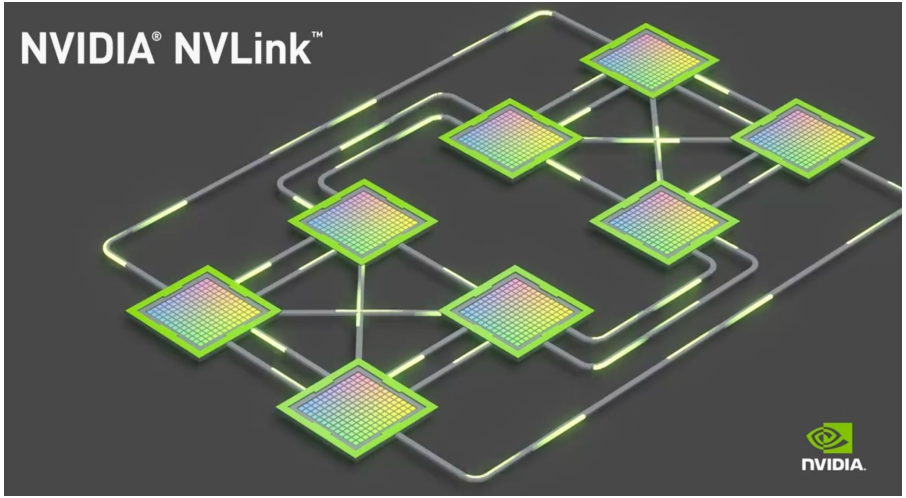
Qubit	cuQuantum	GPU_{Ori}	GPU_{Cache}	GPU_{All}
23	0.28	0.32	0.25 (1.1x)	0.06 (**4.7x**)
24	0.60	0.69	0.53 (1.1x)	0.12 (**5.0x**)
25	1.30	1.49	1.17 (1.1x)	0.23 (**5.6x**)
26	2.79	3.19	2.50 (1.1x)	0.56 (**5.0x**)
27	6.00	6.84	5.47 (1.1x)	1.08 (**5.6x**)
28	12.87	14.62	11.72 (1.1x)	2.17 (**6.0x**)
29	27.58	31.19	24.97 (1.1x)	4.45 (**6.2x**)
30	58.93	66.44	52.65 (1.1x)	9.22 (**6.4x**)

- The optimization can also be observed on A100
  - The better results can be achieved when implementations for RZZ gates are further optimized.

One more thing…

NVLink for A100



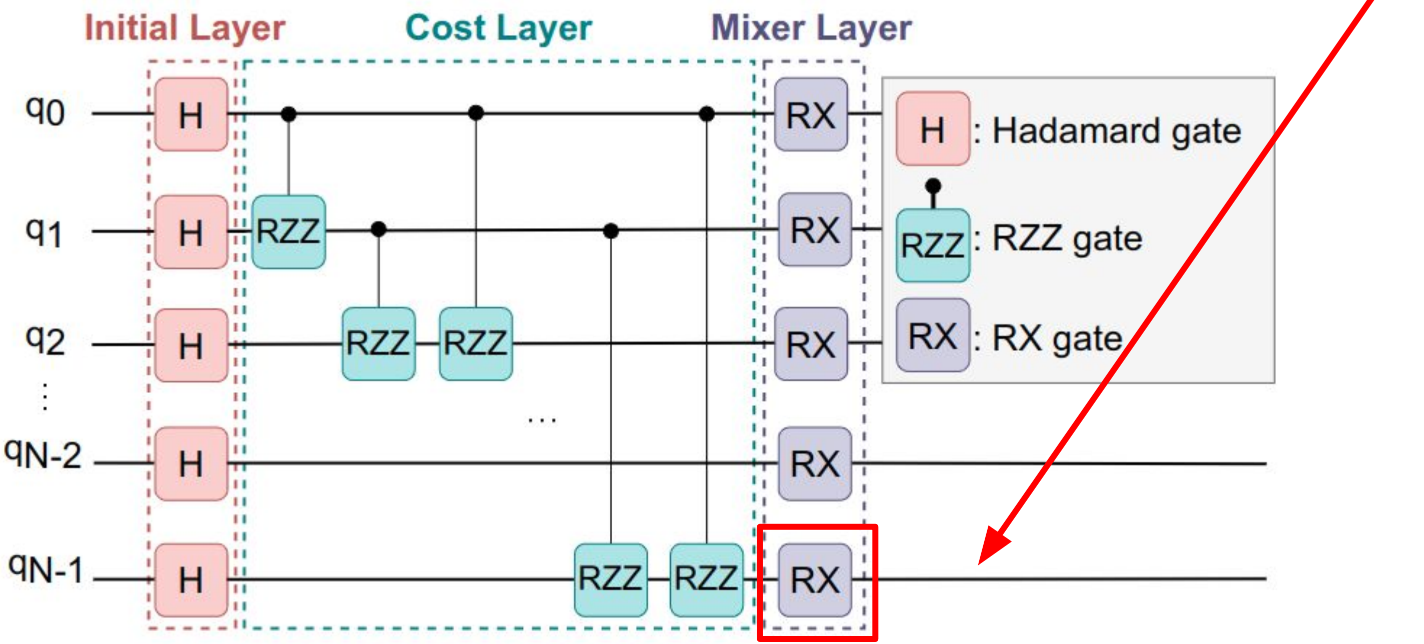


Open-Source Bechmark[1]	NVLink	PCI-E
Bandwidth	247 GB/s	16 GB/s

[1]: https://medium.com/gpgpu/multi-gpu-programming-6768eeb42e2c

NVLink in QAOA

- With Launch Control and Rotation Compression, we can focus on transferring data for one single RX gate.



RX-Gates Benchmark (unit: μs, double)

Qubit	PCIe (μs)	NVLink (μs)	Speedup
23	4,155	365	11.4x
24	7,983	535	15.0x
25	15,817	877	18.0x
26	31,484	1,559	20.2x
27	63,314	2,868	22.1x
28	124,576	5,700	21.9x
29	249,862	10,836	23.1x
30	502,660	22,692	22.2x

NVLink in QAOA w/ 2 A100s

- Utilizing NVLink, two GPUs yield an additional 2x increase in performance.

Table 2: [NCHC's GPU A100] The elapsed time of 5-level QAOA (unit: second, double).

Qubit	cuQuantum	GPU_{Cache}	GPU_{All}	GPU_{PCIe}	GPU_{NVLink}
23	0.28	0.25 (1.1x)	0.06 (4.7x)	0.04 (6.2x)	0.02 (**11.6x**)
24	0.60	0.53 (1.1x)	0.12 (5.0x)	0.09 (6.8x)	0.05 (**12.4x**)
25	1.30	1.17 (1.1x)	0.23 (5.6x)	0.18 (7.3x)	0.10 (**13.1x**)
26	2.79	2.50 (1.1x)	0.56 (5.0x)	0.35 (7.9x)	0.20 (**13.7x**)
27	6.00	5.47 (1.1x)	1.08 (5.6x)	0.73 (8.2x)	0.42 (**14.3x**)
28	12.87	11.72 (1.1x)	2.17 (6.0x)	1.72 (7.4x)	0.97 (**13.3x**)
29	27.58	24.97 (1.1x)	4.45 (6.2x)	3.33 (8.2x)	2.13 (**12.9x**)
30	58.93	52.65 (1.1x)	9.22 (6.4x)	6.60 (8.9x)	4.06 (**14.5x**)

Energy Efficiency

INPUTS	
# CPU Cores	64
# GPUs (A100)	2
Application Speedup	92.0x

Node Replacement	184.0x

GPU NODE POWER SAVINGS			
	AMD Dual Rome 7742	**8x A100 80GB SXM4**	**Power Savings**
Compute Power (W)	202,400	6,500	195,900
Networking Power (W)	8,544	93	8,451
Total Power (W)	**210,944**	**6,593**	204,351

Node Power efficiency	**32.0x**

ANNUAL ENERGY SAVINGS PER GPU NODE			
	AMD Dual Rome 7742	**8x A100 80GB SXM4**	**Power Savings**
Compute Power (kWh/year)	1,773,024	56,940	1,716,084
Networking Power (kWh/year)	74,849	814	74,035
Total Power (kWh/year)	**1,847,873**	**57,754**	**1,790,119**

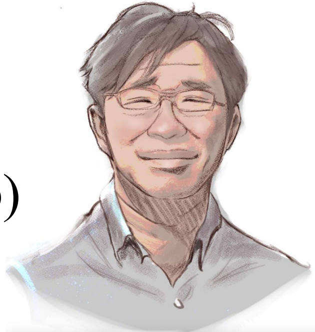
$/kWh	$	0.34
Annual Cost Savings	$	608,640.46
3-year Cost Savings	$	1,825,921.38

Metric Tons of CO2	1,269
Gasoline Cars Driven for 1 year	274
Seedlings Trees grown for 10 years	20,980

(source: Link)

Contact us to resolve your performance issues.

Professor: Shih-Hao Hung
Email: hungsh@csie.ntu.edu.tw
Performance, Application and Security Laboratory (PAS Lab)



Thank you for listening :)

Any Questions / Feedback?

Yu-Cheng Lin
Contact me: r11922015@csie.ntu.edu.tw