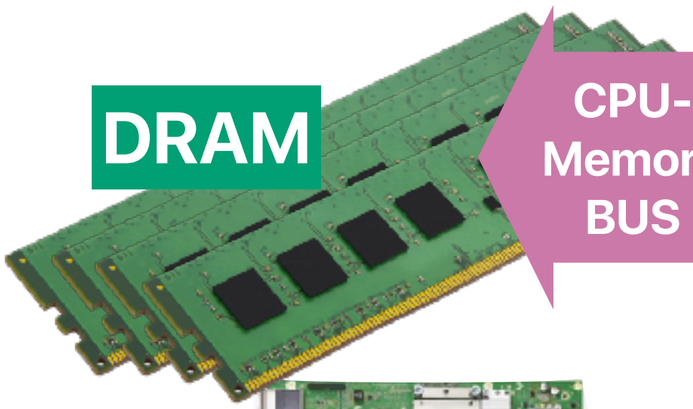
General purpose computing on hardware accelerators (3)

Hung-Wei Tseng

Recap: The landscape of modern computers



CPU-Memory BUS

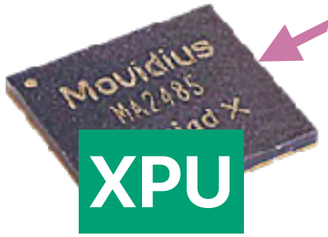
DDR4

Memory Controller

High-performance, but very specialized

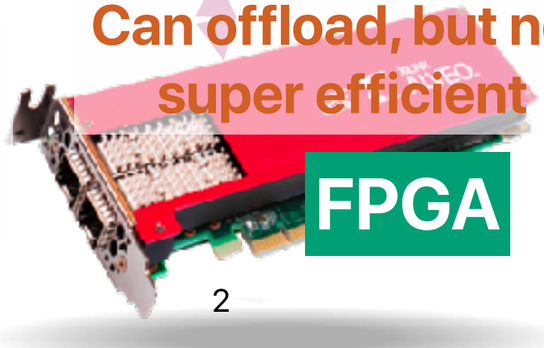
PCI EXPRESS

PCIe Root Complex



General-purpose for "regular" workloads, but can be power consuming

Can offload, but not super efficient



Recap: The concept of approximate computing on NPUs

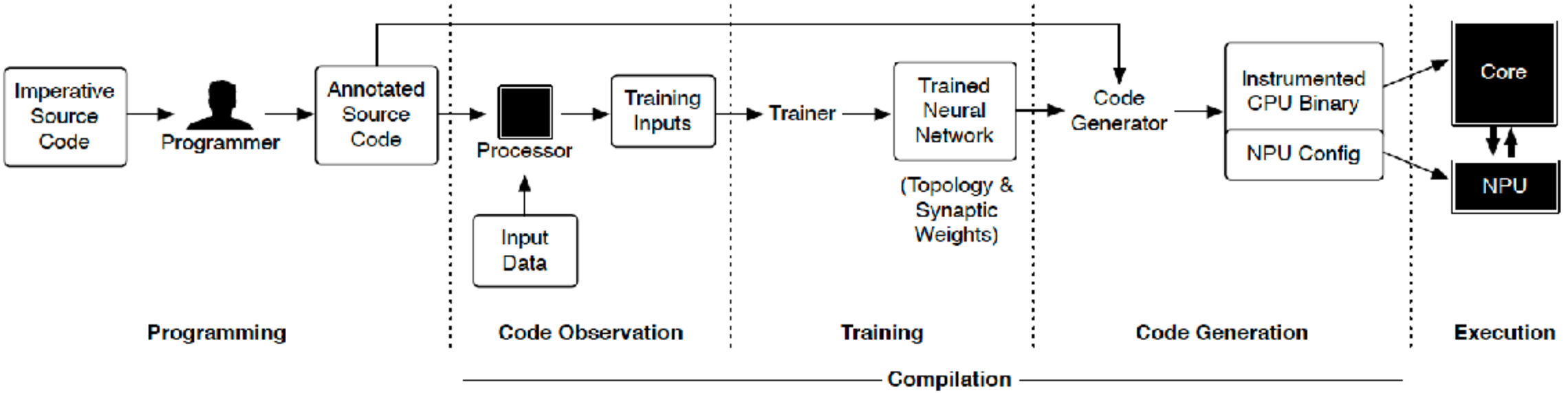
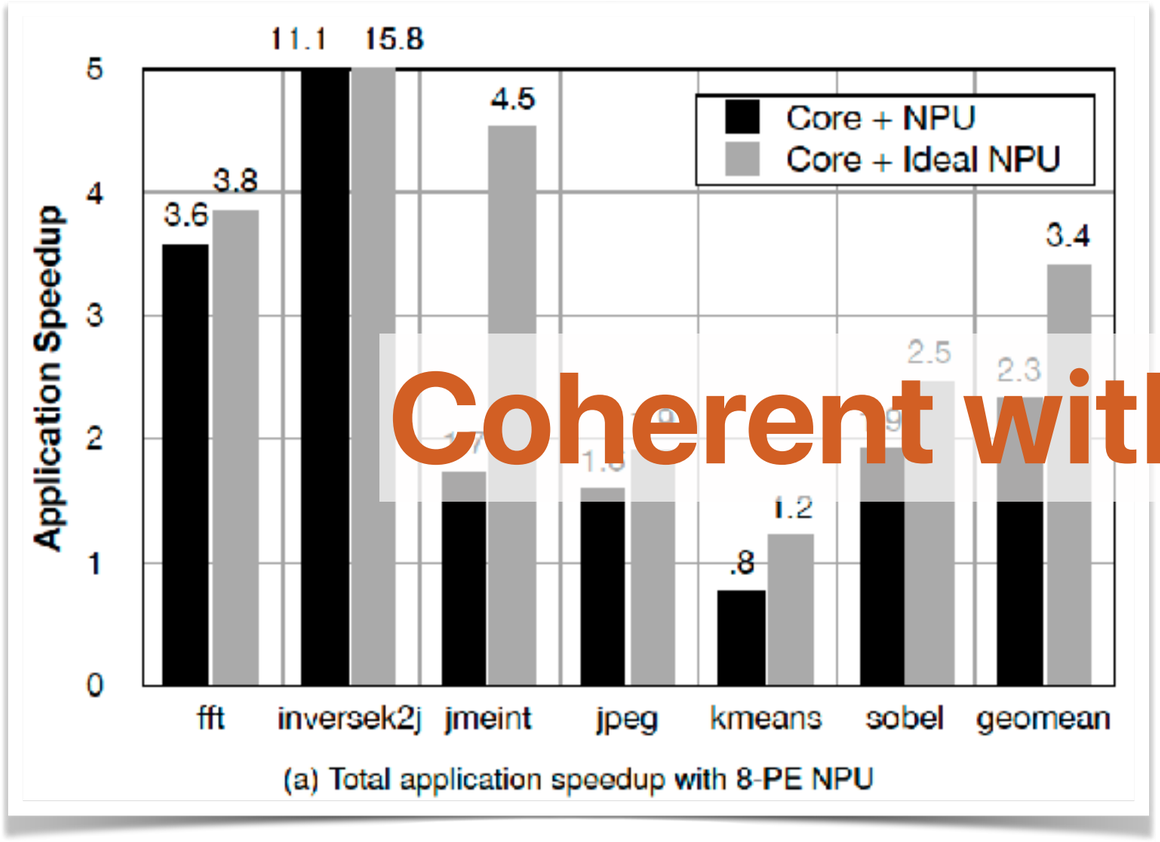


Figure 1: The Parrot transformation at a glance: from annotated code to accelerated execution on an NPU-augmented core.

The idea works more efficiently if accelerator exists



(a) Total application speedup with 8-PE NPU

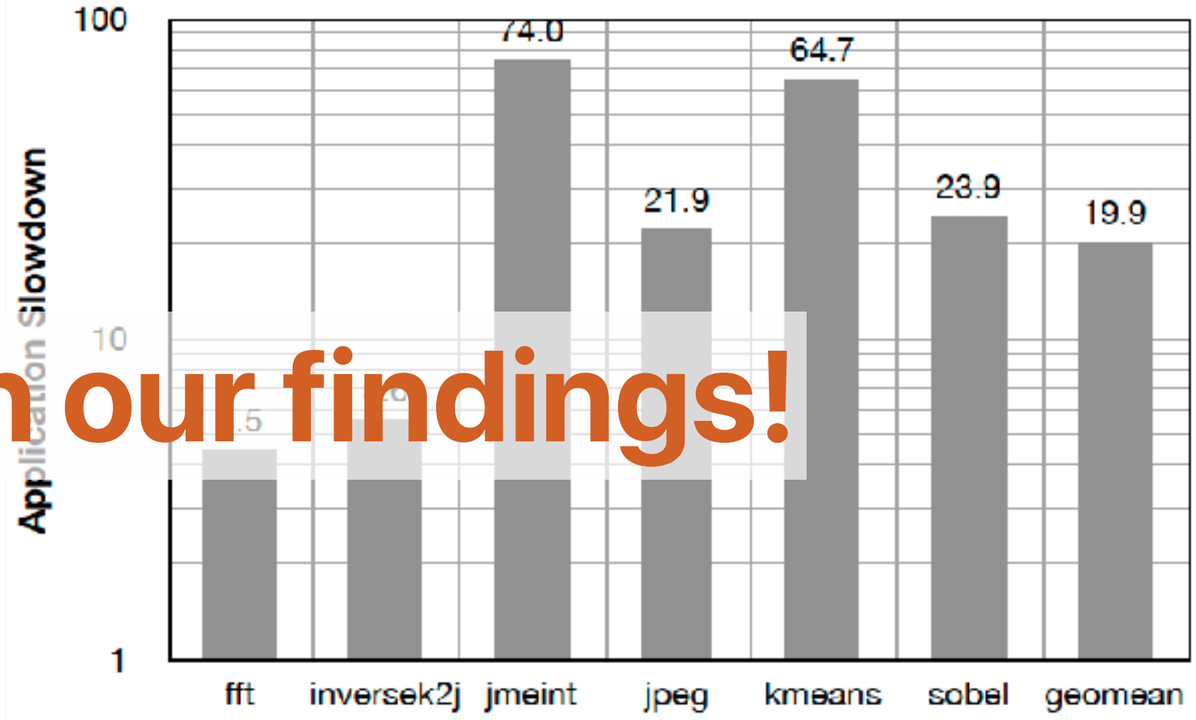
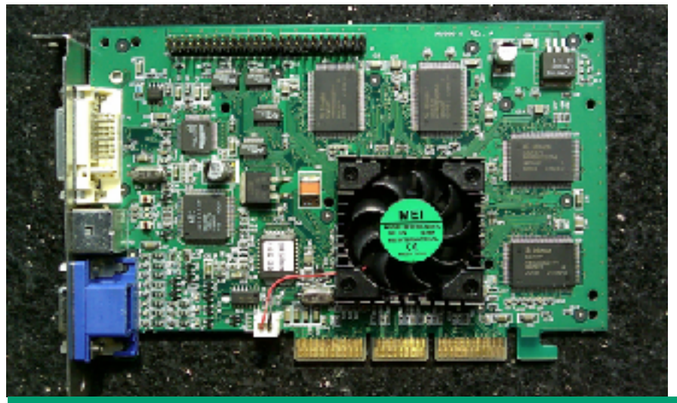


Figure 9: Slowdown with software neural network execution.

Coherent with our findings!

Can we create another revolution with TPUs?



Graphics "Accelerator", 1999

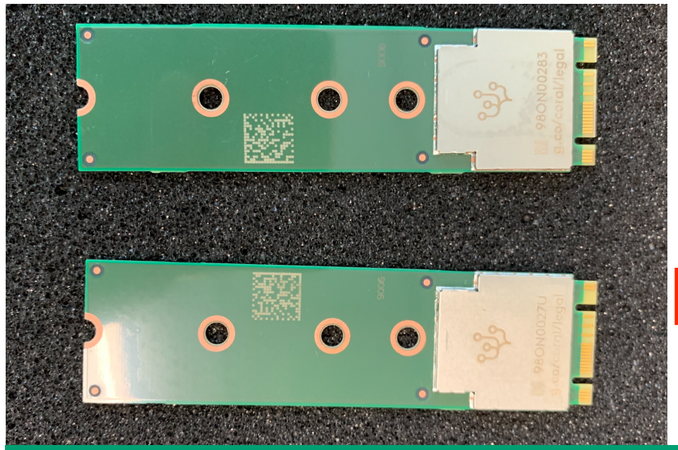
```
__global__ void
vectorAdd(const float *A, const float *B, float *C, int numElements)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;

    if (i < numElements)
    {
        C[i] = A[i] + B[i];
    }
}
```

- CUDA
- OpenCL

G80 (e.g., GTX 8800) with CUDA, 2007

General Purpose Computing on GPUs



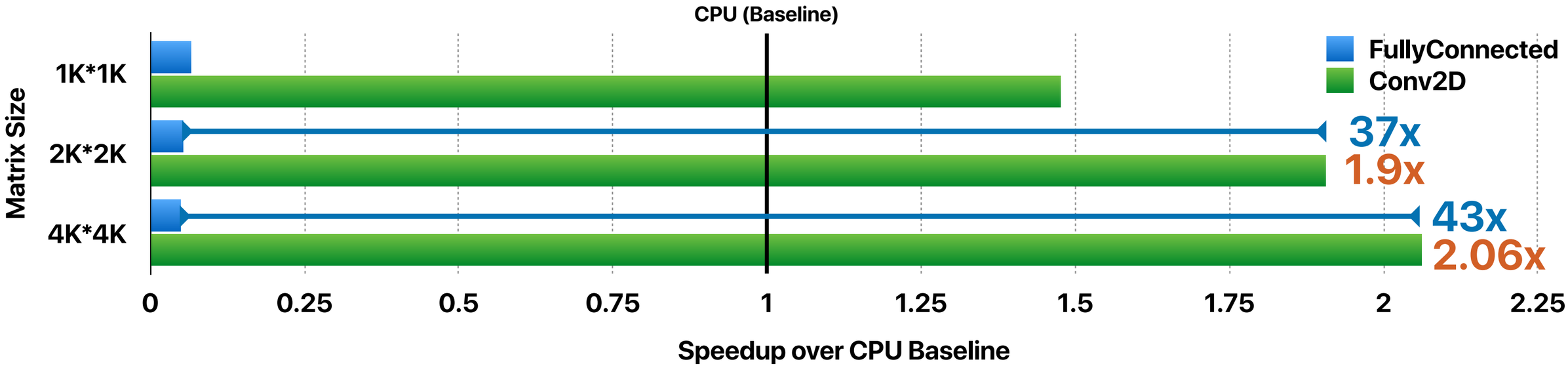
AI "Accelerator", 2013—

- Programing Language
- Compiler
- Runtime

?

General Purpose Computing on TPUs

On the real machine



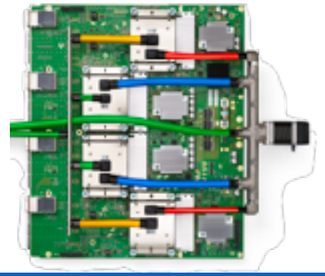
Recap: Lessons learned from developing “GPTPU”

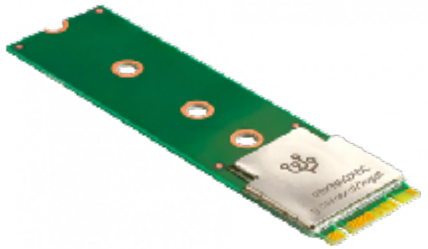
- Reverse engineering is interesting but not worth the time
- Developing software frameworks on hardware with steady support
- The accessibility to low-level and backward compatible APIs
- Estimating the performance gain of design using the right/appropriate metrics
- Algorithm implementations should fit the most efficient operations
- The precision support is still below the demand of applications beyond AI/ML

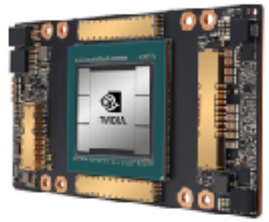
Outline

- General purpose computing on Edge TPUs (cont.)
- TCUDB
- General purpose computing on Ray Tracing Hardware
- RTNN

TPU v.s. Edge TPU v.s. Tensor Cores







	TPU v4	Edge TPU	RTX 3090 Tensor Cores
Year	2021	2018	2020
Precision	BF16	INT8	FP16/BF16/TF32
Accessibility	Google Cloud Only	Public Available	Public Available
Programming Framework	Tensorflow/PyTorch	Tensorflow/TFLite/C++	CUDA/Tensorflow/Pytorch
TOPS	275	4	285
Power (Watt)	170	2	350
TOPS/W	1.62	2	0.814285714285714
Device Memory	32 GB	8MB	24 GB

Lessons learned from developing “GPTPU”

- Reverse engineering is interesting but not worth the time
- Work with the company to get important insights/hints/feedbacks
- Developing software frameworks on hardware with steady support
- The accessibility to low-level and backward compatible APIs
- Estimating the performance gain of design using the right/appropriate metrics
- Algorithm implementations should fit the most efficient operations
- The precision support is still below the demand of applications beyond AI/ML

3:00

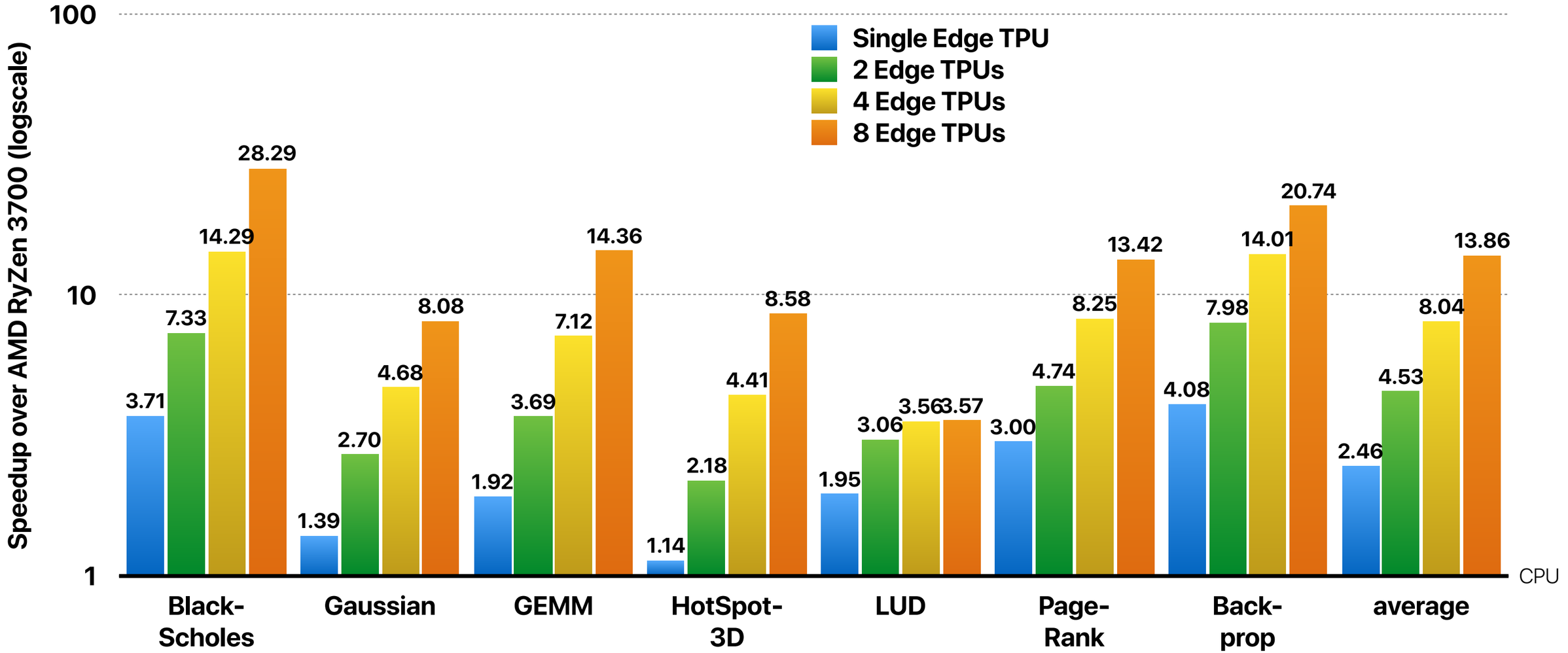
Can you compare Approx. on NPU & GPTPU? Under what situations would you prefer each?

Approx. on NPU v.s. GPTPU

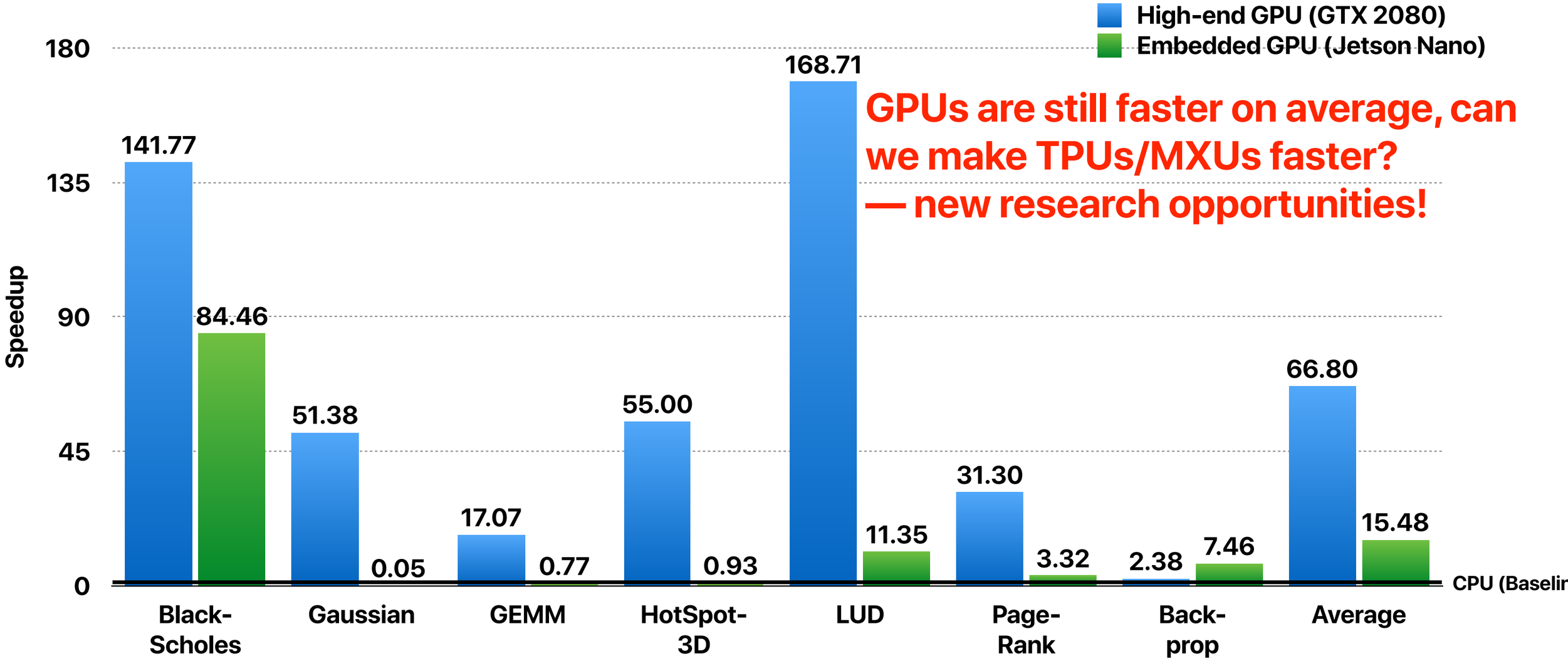
Recap: Approx. on NPU v.s. GPTPU

	Approx. on NPU	GPTPU
Performance	Better if the algorithm is more complex, but worse otherwise	Depending on the complexity of the algorithm
Result quality	Approximate, not accurate	Exact — if the hardware supports the desired precision
Programming efforts	Easier — the programmer does not need to take care of programming on AI/ML accelerators	Difficult — the programmer has to map the computation into operators that the hardware supports
Applicability	Any algorithm on an application tolerating inexactness	Only algorithms that map well to the hardware architecture

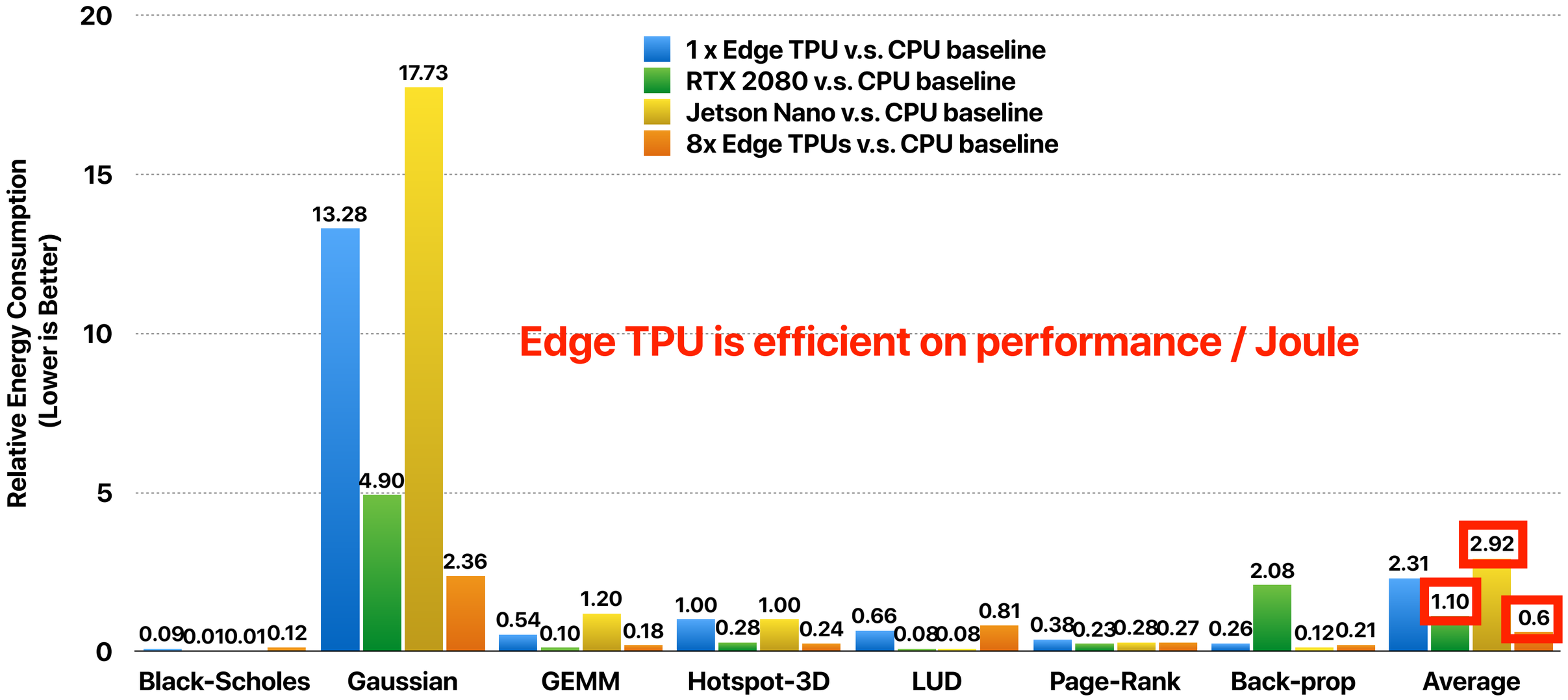
Performance for general-purpose Workloads



Compare 8 Edge TPUs with GPUs



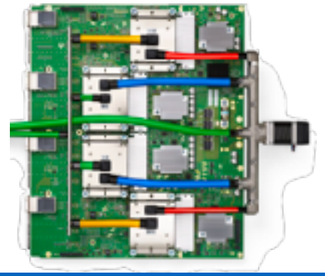
Compare to GPUs

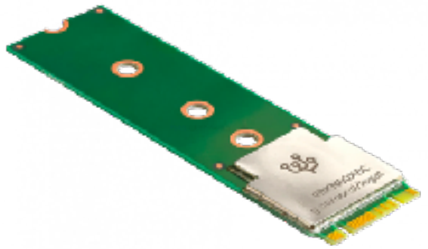


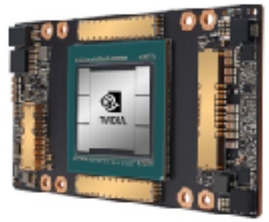
Lessons learned from developing “GPTPU”

- Reverse engineering is interesting but not worth the time
- Developing software frameworks on hardware with steady support
- The accessibility to low-level and backward compatible APIs
- Estimating the performance gain of design using the right/appropriate metrics
- Algorithm implementations should fit the most efficient operations
- The precision support is still below the demand of applications beyond AI/ML

TPU v.s. Edge TPU v.s. Tensor Cores







	TPU v4	Edge TPU	RTX 3090 Tensor Cores
Year	2021	2018	2020
Precision	BF16	INT8	FP16/BF16/TF32
Accessibility	Google Cloud Only	Public Available	Public Available
Programming Framework	Tensorflow/PyTorch	Tensorflow/TFLite/C++	CUDA/Tensorflow/Pytorch
TOPS	275	4	285
Power (Watt)	170	2	350
TOPS/W	1.62	2	0.814285714285714
Device Memory	32 GB	8MB	24 GB

Lessons learned from developing "GPTPU"

- Reverse engineering is interesting but not worth the time
- Work with the company to get important insights/hints/feedbacks
- Developing software frameworks on hardware with steady support
- The accessibility to low-level and backward compatible APIs
- Estimating the performance gain of design using the right/appropriate metrics
- Algorithm implementations should fit the most efficient operations
- The precision support is still below the demand of applications beyond AI/ML

TCUDB: Accelerating Database with Tensor Processors

Yu-Ching Hu, Yuliang Li* and Hung-Wei Tseng
University of California, Riverside and Megagon Labs*
In The ACM SIGMOD/PODS 2022 International Conference on Management of Data.
https://github.com/escalab/TCUDB

Who is my most loyal customer?

$O(m \times n)$

```
SELECT A.customer, B.brand, SUM(A.Quantity * B.Price) AS value FROM A INNER JOIN B WHERE ON
A.ProductID = B.ProductID GROUP BY A.customer, B.brand;
```

Compare and sum

A — m records

Customer	ProductID	Brand	Quant.
Abe	i9-12900	Intel	1
Abe	i7-12700	Intel	1
Abe	RTX3080	NVIDIA	1
Abe	660p	Intel	2
Bob	RyZen 5800G	AMD	1
Bob	980Pro	Samsung	1
Cindy	RTX3080	NVIDIA	1
Cindy	i7-12700	Intel	2
Cindy	660p	Intel	2
Diana	RyZen 5800G	AMD	1
Diana	RX5000	AMD	1
Diana	980Pro	Samsung	1

B — n records

Brand	ProductID	Price
Intel	i9-12900	600
Intel	i7-12700	400
Intel	660p	100
AMD	RX5000	500
AMD	RyZen 5800G	200
NVIDIA	RTX3080	1200
Samsung	980Pro	100

Joined A*B

Customer	ProductID	Brand	Quant.	Price	Value
Abe	i9-12900	Intel	1	600	600
Abe	i7-12700	Intel	1	400	400
Abe	RTX3080	NVIDIA	1	1200	1200
Abe	660p	Intel	2	100	200
Bob	RyZen 5800G	AMD	1	400	400
Bob	980Pro	Samsung	1	100	100
Cindy	RTX3080	NVIDIA	1	1200	1200
Cindy	i7-12700	Intel	2	400	800
Cindy	660p	Intel	2	100	200
Diana	RyZen 5800G	AMD	1	400	400
Diana	RX5000	AMD	1	500	500
Diana	980Pro	Samsung	1	100	100

Customer	Brand	Value
Abe	Intel	1200
Abe	NVIDIA	1200
Bob	AMD	400
Bob	Samsung	100
Cindy	Intel	1000
Cindy	NVIDIA	1200
Diana	AMD	900
Diana	Samsung	100

Memory copy

Compare & Element-wise Memory copy

Vector multiplications

$O(2 \times m \times n)$ $O(m \times n)$

Who is my most loyal customer?

```
SELECT A.customer, B.brand, SUM(A.Quantity * B.Price) AS value FROM A INNER JOIN B WHERE ON
A.ProductID = B.ProductID GROUP BY A.customer, B.brand;
```

A

Customer	ProductID	Brand	Quant.
Abe	i9-12900	Intel	1
Abe	i7-12700	Intel	1
Abe	RTX3080	NVIDIA	1
Abe	660p	Intel	2
Bob	RyZen 5800G	AMD	1
Bob	980Pro	Samsung	1
Cindy	RTX3080	NVIDIA	1
Cindy	i7-12700	Intel	2
Cindy	660p	Intel	2
Diana	RyZen 5800G	AMD	1
Diana	RX5000	AMD	1
Diana	980Pro	Samsung	1

B

$O(n)$

Brand	ProductID	Price
Intel	i9-12900	600
Intel	i7-12700	400
Intel	660p	100
AMD	RX5000	500
AMD	RyZen 5800G	200
NVIDIA	RTX3080	1200
Samsung	980Pro	100

Customer	Intel	AMD	NVIDIA	Samsung
Abe	1200	0	1200	0
Bob	0	400	0	100
Cindy	1000	0	1200	0
Diana	0	900	0	100

Matrix multiplications

$O(m \times n \times k)$

— This could be done by 1 Tensor Core OP!

Left

Customer/ ProductID	i9-12900	i7-12700	660p	RX500 0	RyZen 5800G	RTX3080	980Pro
Abe	1	1	2	0	0	1	0
Bob	0	0	0	0	1	0	1
Cindy	0	2	2	0	0	1	0
Diana	0	0	0	1	1	0	1

×

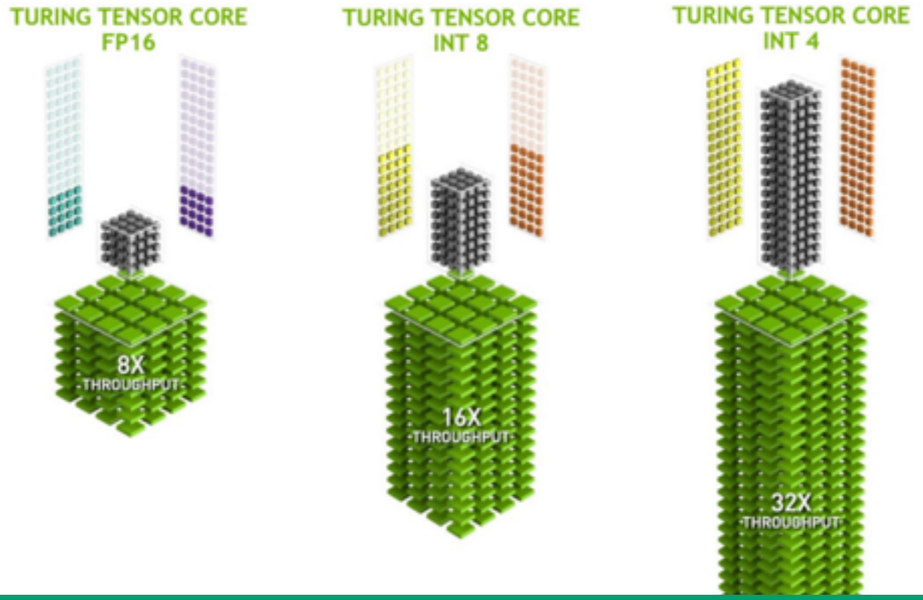
ProductID/ Brand	Intel	AMD	NVIDIA	Samsung
i9-12900	600	0	0	0
i7-12700	400	0	0	0
660p	100	0	0	0
RX5000	0	500	0	0
RyZen 5800G	0	400	0	0
RTX3080	0	0	1200	0
980Pro	0	0	0	100

Memory copy

$O(m)$

Recap: Efficiency of Tensor Cores

The NVIDIA Tesla T4 GPU includes 2,560 CUDA Cores and 320 Tensor Cores, delivering up to 130 TOPs (Tera Operations per second) of INT8 and up to 260 TOPS of INT4 inferencing performance (see Appendix A, *Turing TU104 GPU* for more Tesla T4 specifications). Compared to CPU-based inferencing, the Tesla T4, powered by the new Turing Tensor Cores, delivers up to 40X higher inference performance[6] (see Figure 9).



FLOPS of 8K by 8K matrix multiplications =

$$8192 \times 8192 \times 8192 \times 2$$

$$\frac{8192 \times 8192 \times 8192 \times 2}{2560} = 429496730 \ CUDA \ core \ cycles$$

$$\frac{8192 \times 8192 \times 8192 \times 2}{320 \times 128} = 26843546 \ Tensor \ core \ cycles$$

Tensor cores make matrix processing 16x faster

Each tensor core operation performs 4x4x4 MMA in one cycle
~ 128 FP operations in conventional scalar models

NVIDIA, "NVIDIA Turing GPU architecture: Graphics reinvented," 2018. [Online]. Available: https://www. nvidia.com/content/dam/en-zz/Solutions/designvisualization/technologies/turing-architecture/NVIDIATuring- Architecture-Whitepaper.pdf

TCU friendly queries

Two-way natural join	`SELECT A.Val, B.Val FROM A, B` `WHERE A.ID = B.ID;`

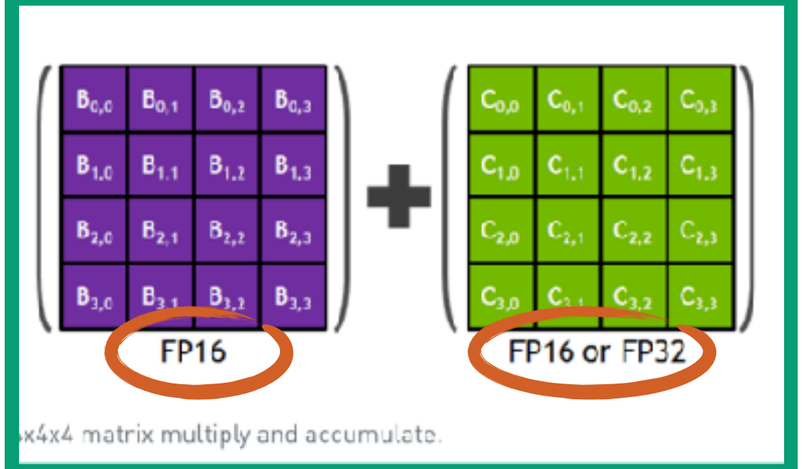
Multi-way joins	`SELECT A.Val, B.Val, C.Val FROM A,B,C`
	`WHERE`
	`A.ID_1 = B.ID_1 AND B.ID_2 = C.ID_2;`

Group-by aggregates over joins	`SELECT SUM(A.Val), B.Val FROM A, B`
	`WHERE A.ID = B.ID`
	`GROUP BY B.Val;`

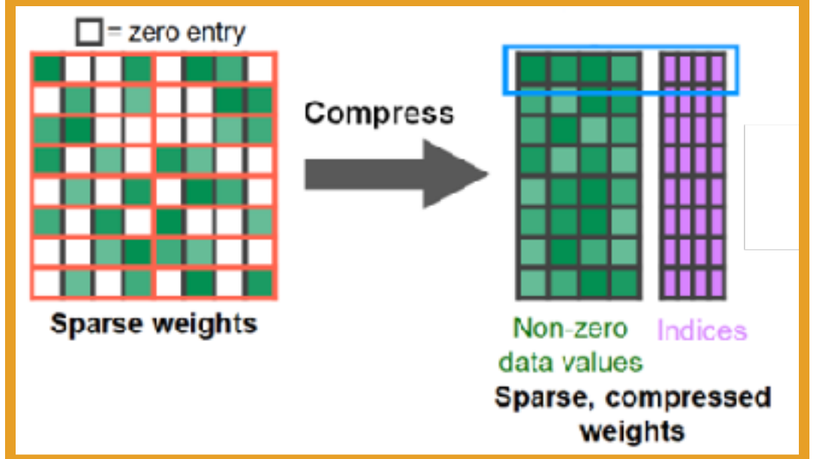
Aggregate join without group-by	`SELECT SUM(A.Val * B.Val)`
	`FROM A, B`
	`WHERE A.ID = B.ID;`

Challenges

Limited precision



Data-dependent code optimization

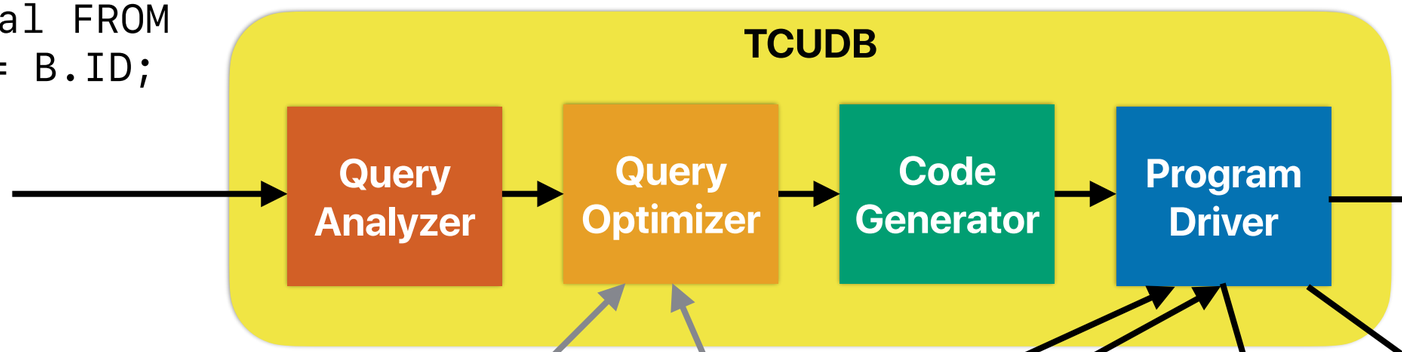


Limited device memory

Peak TF32 Tensor TFLOPS[1]	NA
Peak INT8 Tensor TOPS[1]	261
Peak INT4 Tensor TOPS[1]	522
Frame Buffer Memory Size and Type	24576MB GDDR6
Memory Interface	384-bit
Memory Clock (Data Rate)	14 Gbps
Memory Bandwidth (GB/sec)	672 GB/sec
ROPs	96
Pixel Fill-rate (Gigapixels/sec)	169.9

The TCUDB system architecture

```
SELECT A.Val, B.Val FROM
A, B WHERE A.ID = B.ID;
```




```
wmma_optimized_gemm<<<deviceProp.multiProcess
orCount, THREADS_PER_BLOCK, SHMEM_SZ>>>(At,
Bt, Ct, Ct, M, N, K, 1.0, 0.0));
```

A

ID	Value
1	307
2	299
3	265
4	262
5	285
6	306
7	287
8	266

B

ID	Value
1	2
2	706
3	5
4	857
5	99
6	264
7	936
8	916

Table Storage

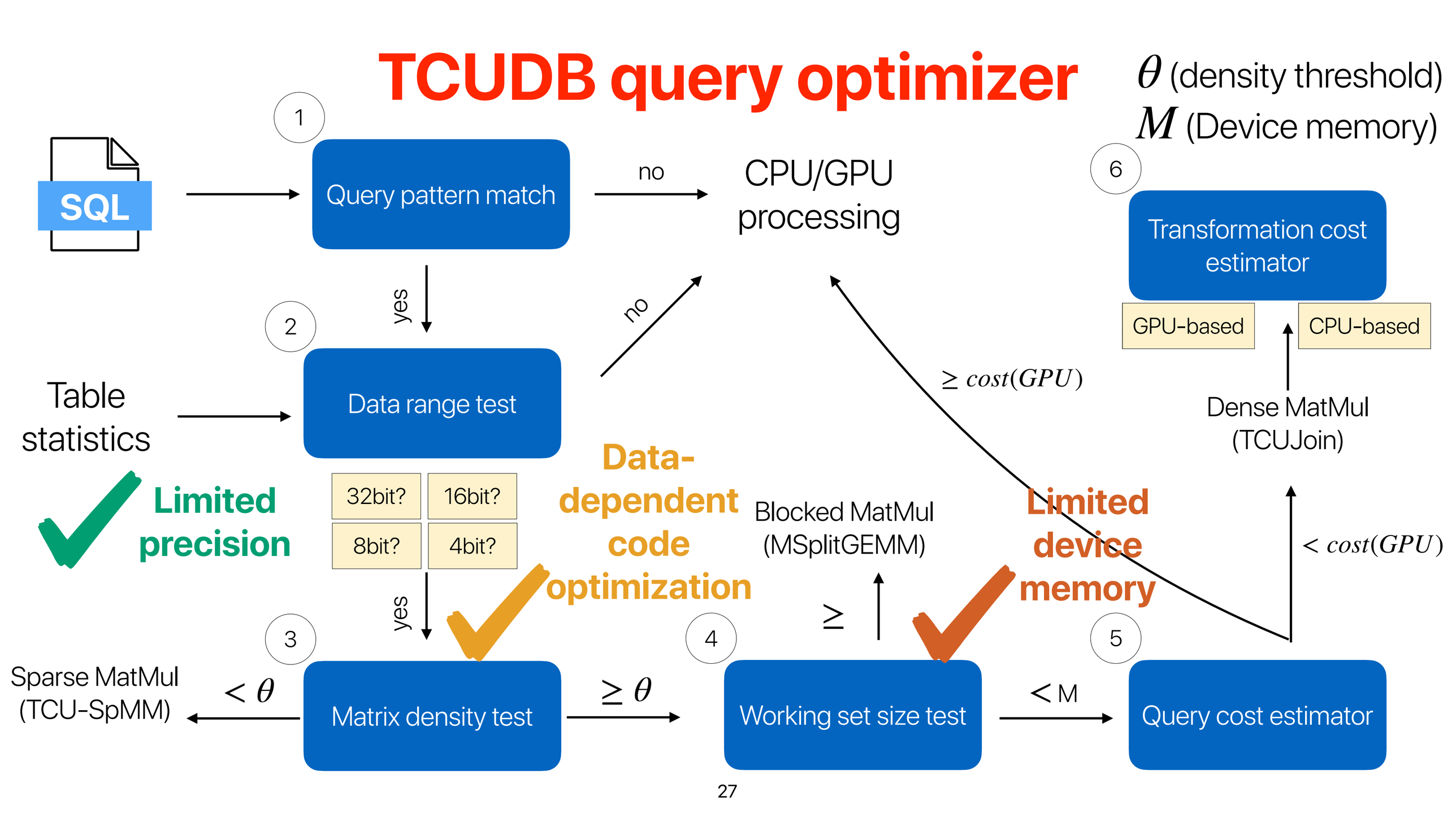
At

	1	2	3	4	5	6	7	8
1	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	1

Bt

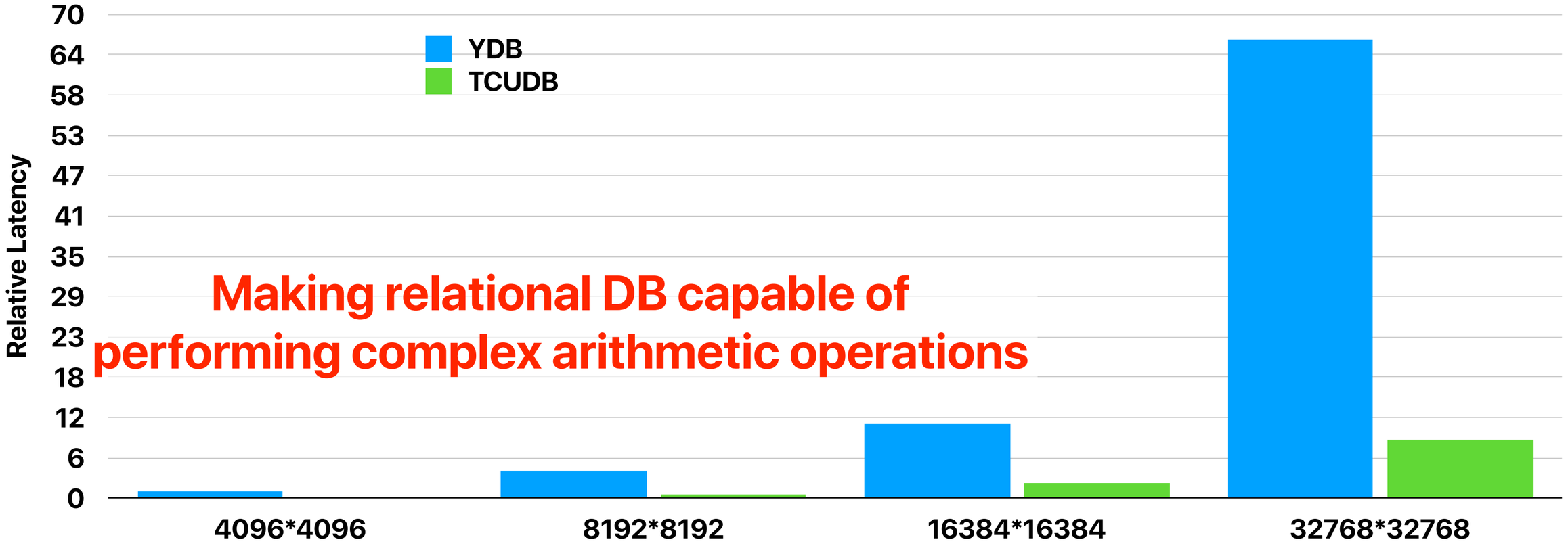
	1	2	3	4	5	6	7	8
1	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	1

TCUDB query optimizer



Matrix Multiplication Query

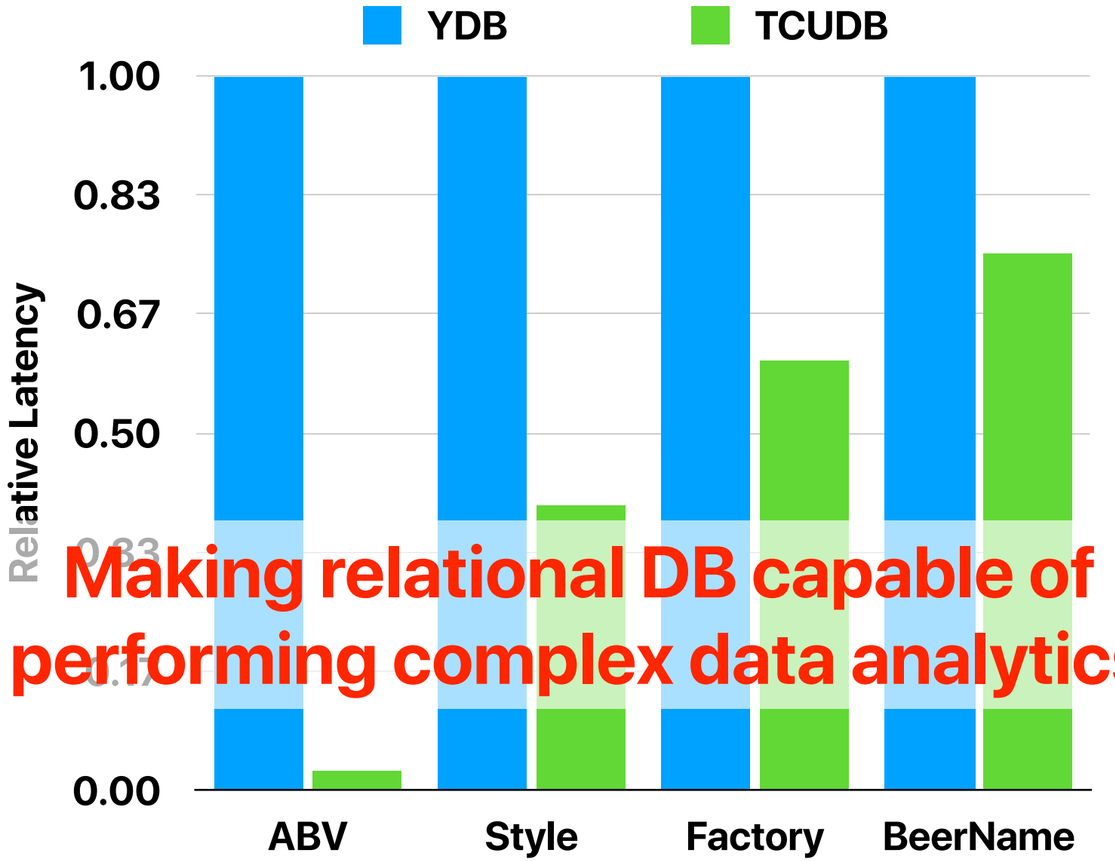
- `SELECT A.col_num, B.row_num, SUM(A.val * B.val) as res FROM A, B WHERE A.row_num = B.col_num GROUP BY A.col_num, B.row_num;`



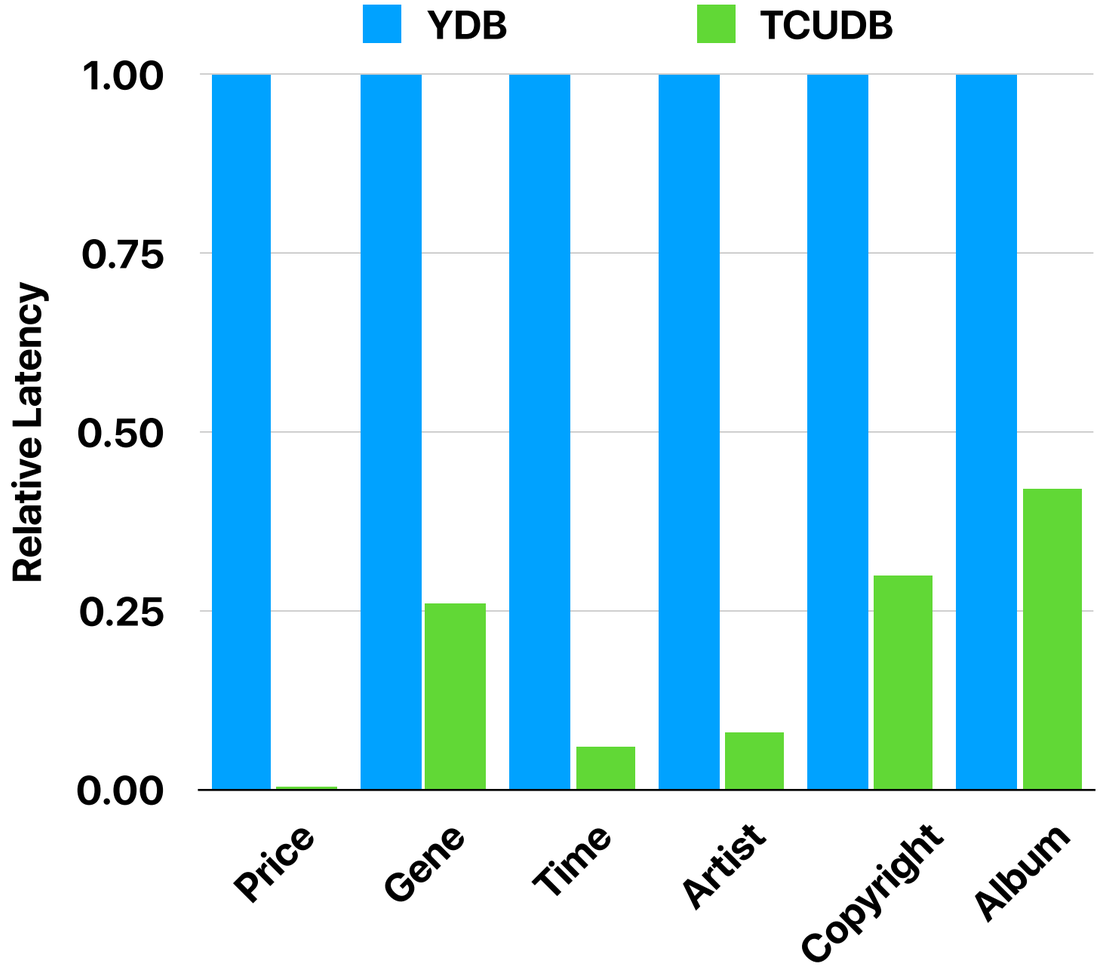
Making relational DB capable of performing complex arithmetic operations

Entity matching

- SELECT TABLE_A.ID, TABLE_A.BEER_NAME, TABLE_B.ID, TABLE_B.BEER_NAME FROM TABLE_A, TABLE_B WHERE TABLE_A.ABV = TABLE_B.ABV;



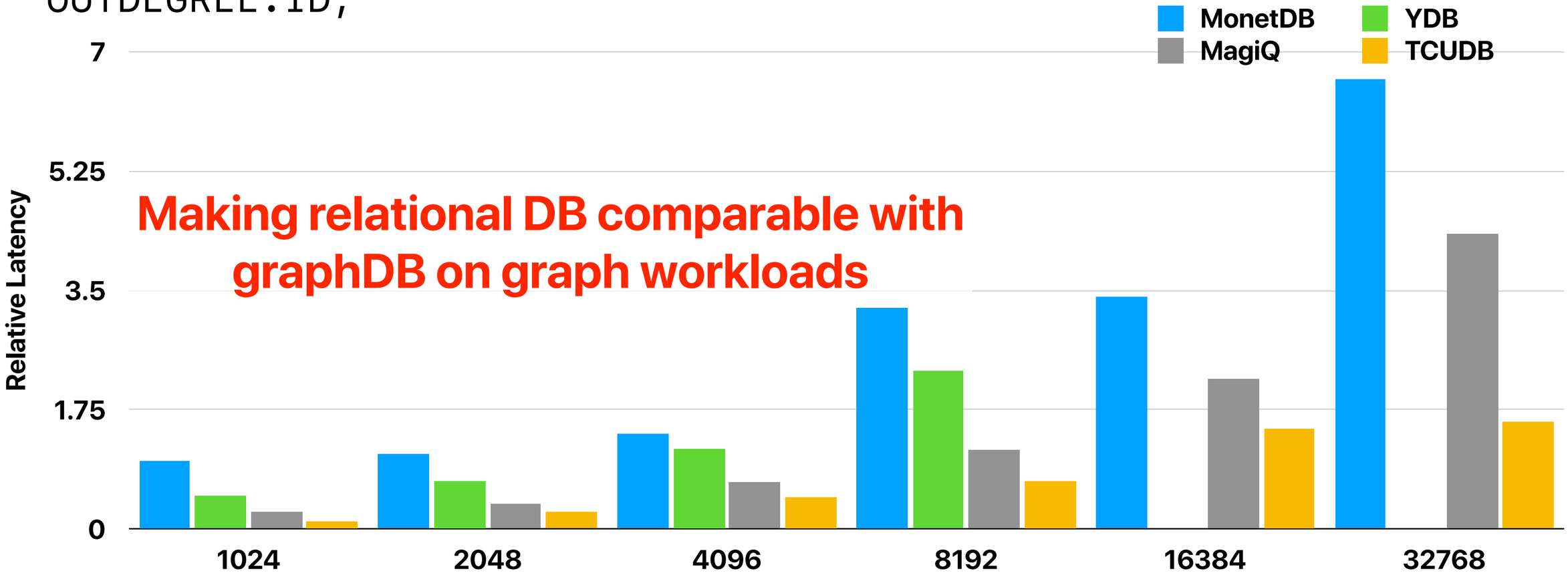
- SELECT TABLE_A.ID, TABLE_A.SONG, TABLE_B.ID, TABLE_B.SONG FROM TABLE_A, TABLE_B WHERE TABLE_A.ARTIST = TABLE_B.ARTIST;



Making relational DB capable of performing complex data analytics

PageRank

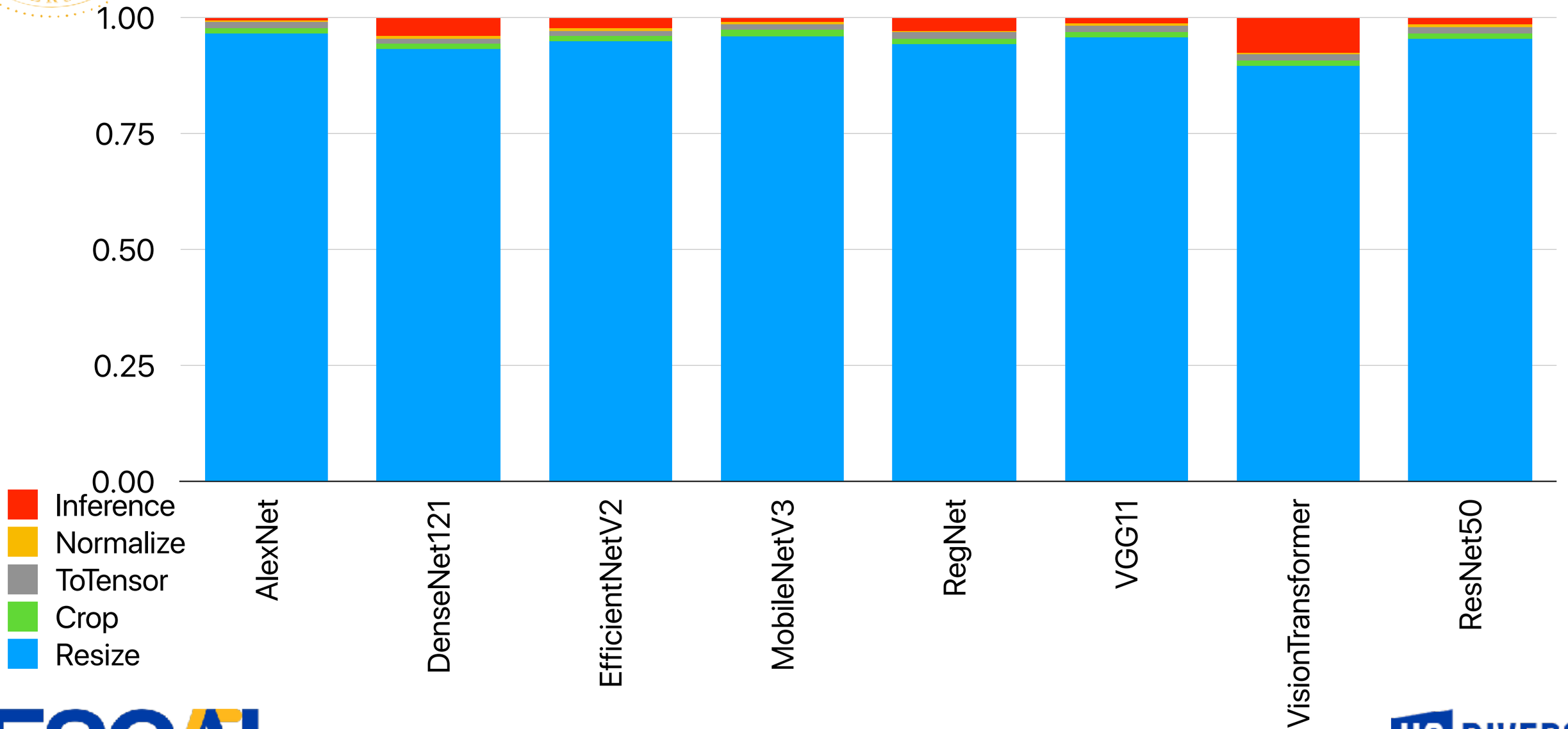
- SELECT SUM(@alpha * PAGERANK.rank / OUTDEGREE.DEGREE) + (1-@alpha)/@num_node FROM PAGERANK, OUTDEGREE WHERE PAGERANK.ID = OUTDEGREE.ID;



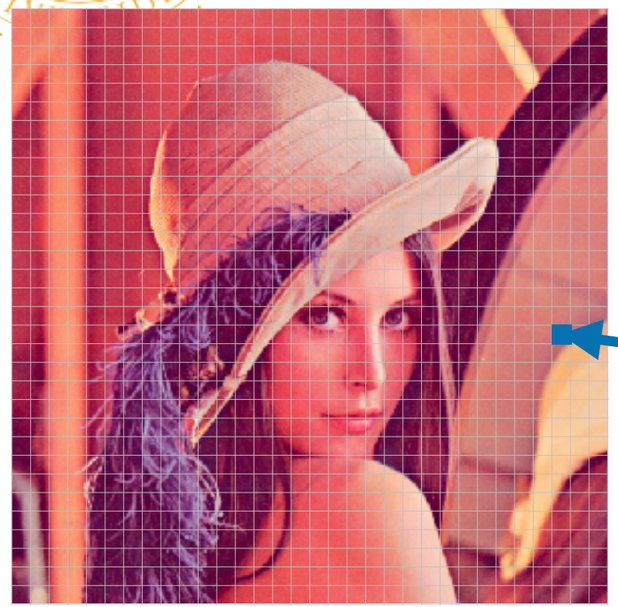
TensorCV: Accelerating ML-adjacent Computation Using Tensor Processors

Dongho Ha+*, Won Woo Ro* and Hung-Wei Tseng+
Yonsei University* and University of California, Riverside+
In International Symposium on Low Power Electronics and Design (ISLPED 2023)

Image pre-processing is the new bottleneck



Traditional image pre-processing



Pixel position: (i, j)

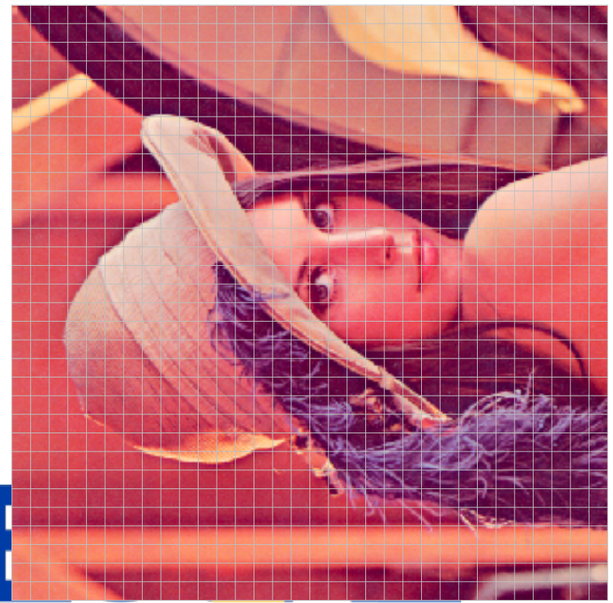
Pixel value: (Rij, Gij, Bij)

$$\begin{bmatrix} i_{rot} \\ j_{rot} \end{bmatrix} = \begin{bmatrix} i \\ j \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Matrix-vector multiplications

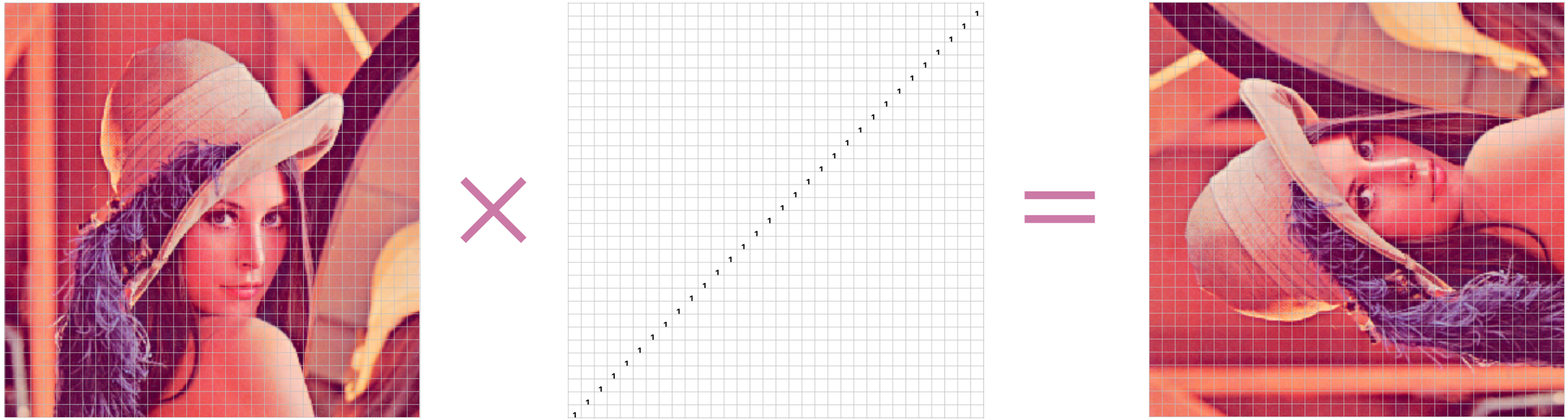
$$Rotated\_image_{i_{rot}, j_{rot}} = (R_{ij}, G_{ij}, B_{ij})$$

Memory operation



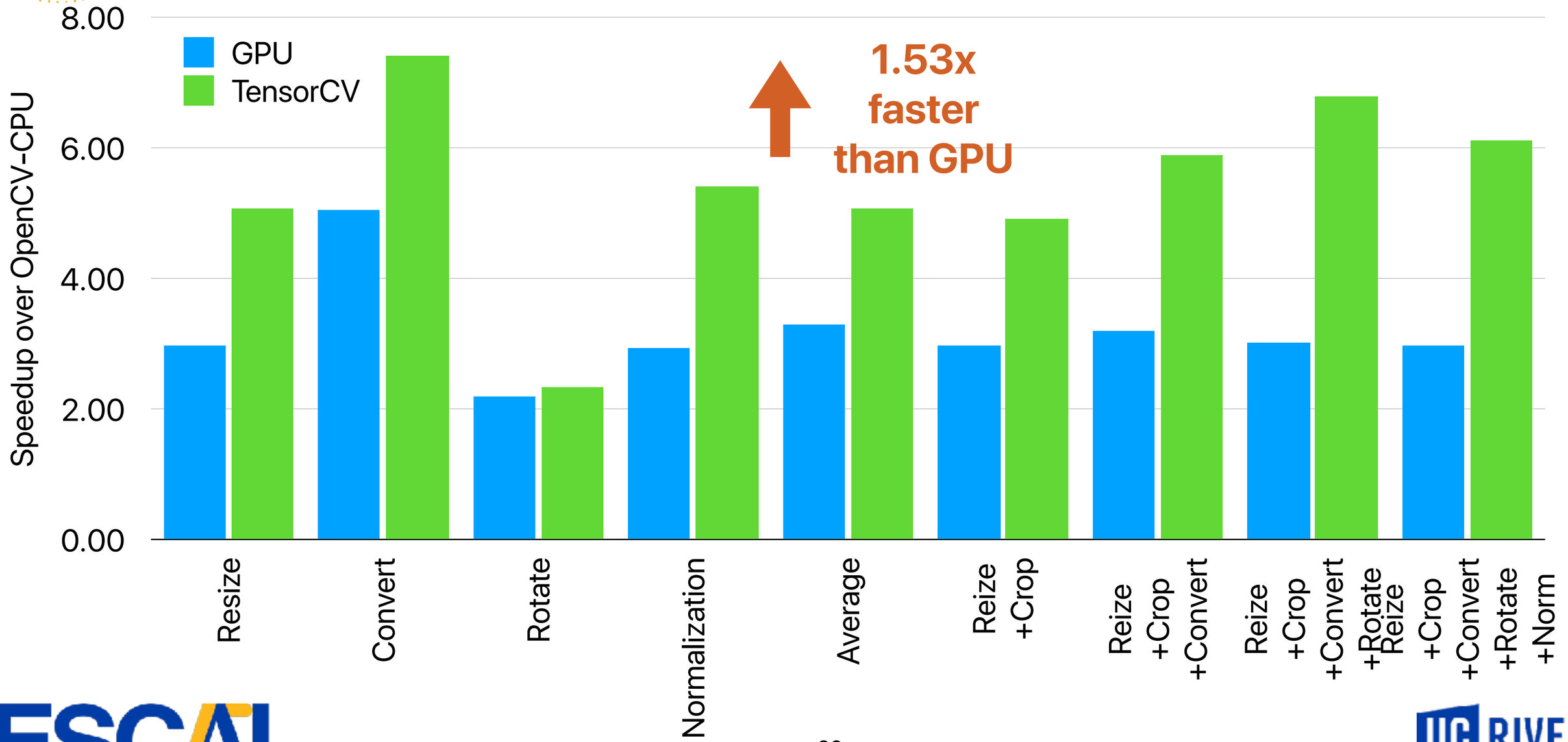
$$\begin{bmatrix} i_{rot} \\ j_{rot} \end{bmatrix} = \begin{bmatrix} i \\ j \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

"Tensorized" image processing algorithm



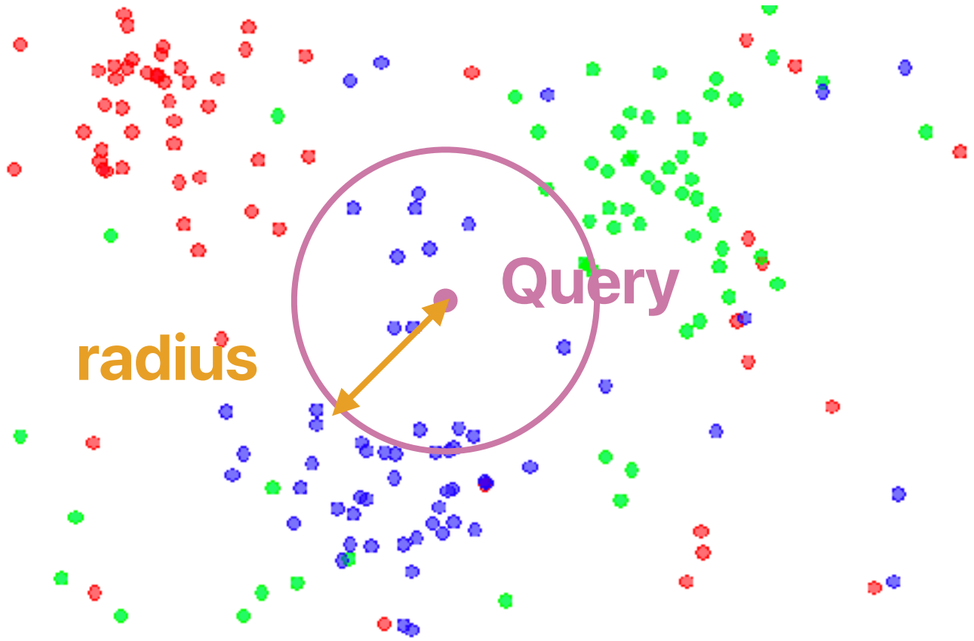
Only 1 mac operation in Tensor Processors!

Speedup over CPU and GPU



Nearest neighbor search

- Fixed-radius search (Range search) — returns all the neighbors within a fixed radius.
- K nearest neighbor search (kNN) — returns the nearest K neighbors of a query



3:00

Can you map the NN problem as a ray tracing problem?

NN Search as Ray Tracing?

RTNN: accelerating neighbor search using hardware ray tracing

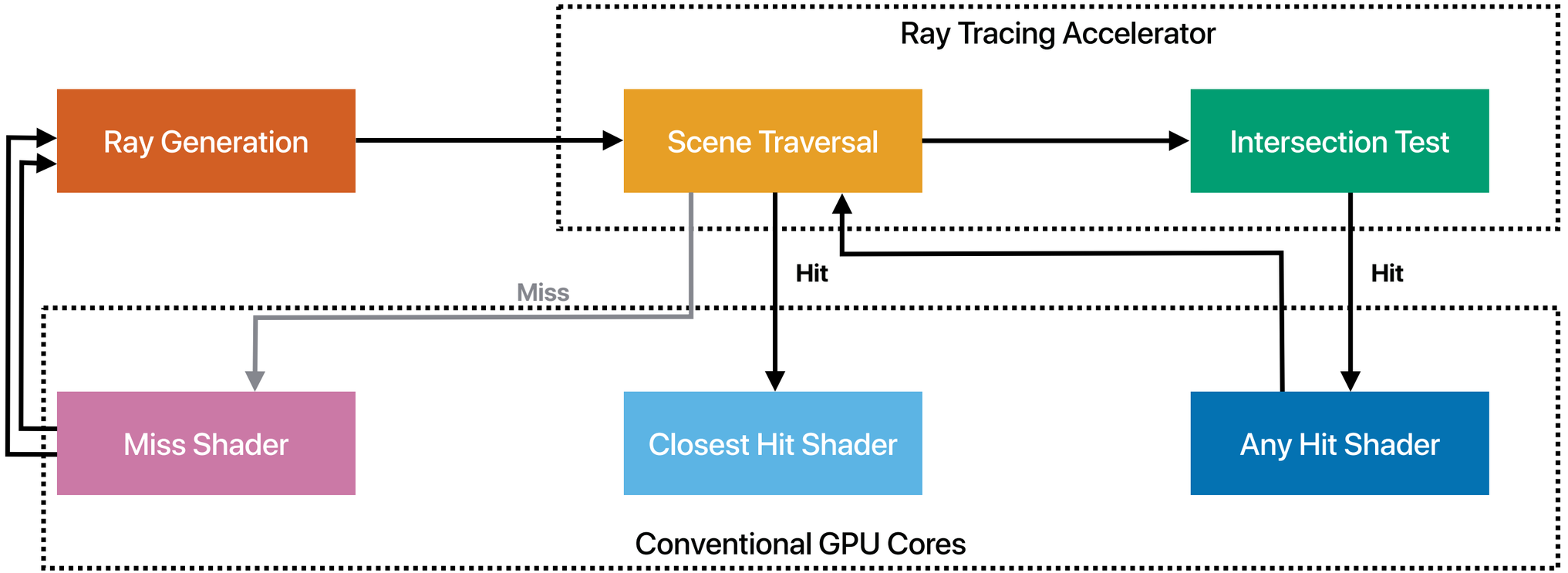
Yuhao Zhu
University of Rochester
In Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '22)
https://github.com/horizon-research/rtnn

Recap: Simplest ray tracing algorithm

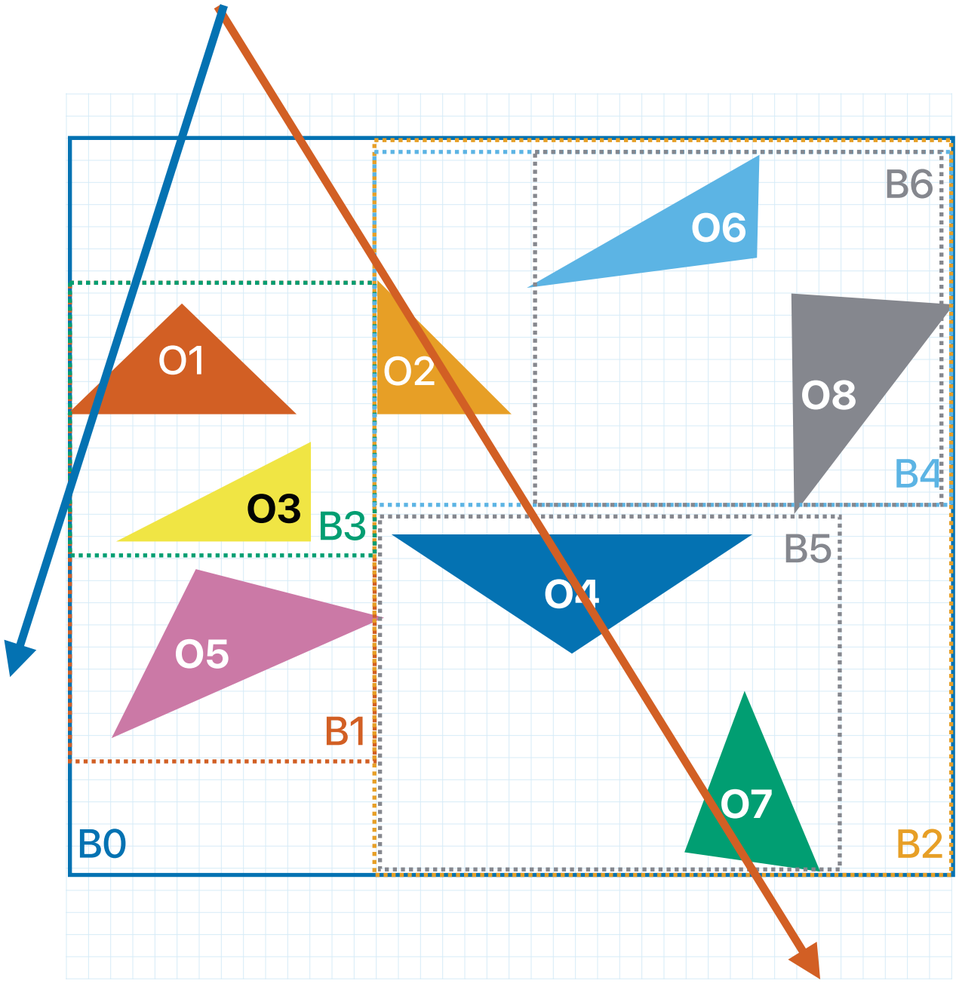
```
for (int j = 0; j < imageHeight; ++j) {
    for (int i = 0; i < imageWidth; ++i) {
        // compute primary ray direction
        Ray primRay;
        computePrimRay(i, j, &primRay);
        // shoot prim ray in the scene and search for the intersection
        Point pHit;
        Normal nHit;
        float minDist = INFINITY;
        Object object = NULL;
        for (int k = 0; k < objects.size(); ++k) {
            if (Intersect(objects[k], primRay, &pHit, &nHit)) {
                float distance = Distance(eyePosition, pHit);
                if (distance < minDistance) {
                    object = objects[k];
                    minDistance = distance;  //update min distance
                }
            }
        }
        if (object != NULL) {
            // compute illumination
            Ray shadowRay;
            shadowRay.direction = lightPosition - pHit;
            bool isShadow = false;
            for (int k = 0; k < objects.size(); ++k) {
                if (Intersect(objects[k], shadowRay)) {
                    isInShadow = true;
                    break;
                }
            }
        }
        if (!isInShadow)
            pixels[i][j] = object->color * light.brightness;
        else
            pixels[i][j] = 0;
    }
}
```

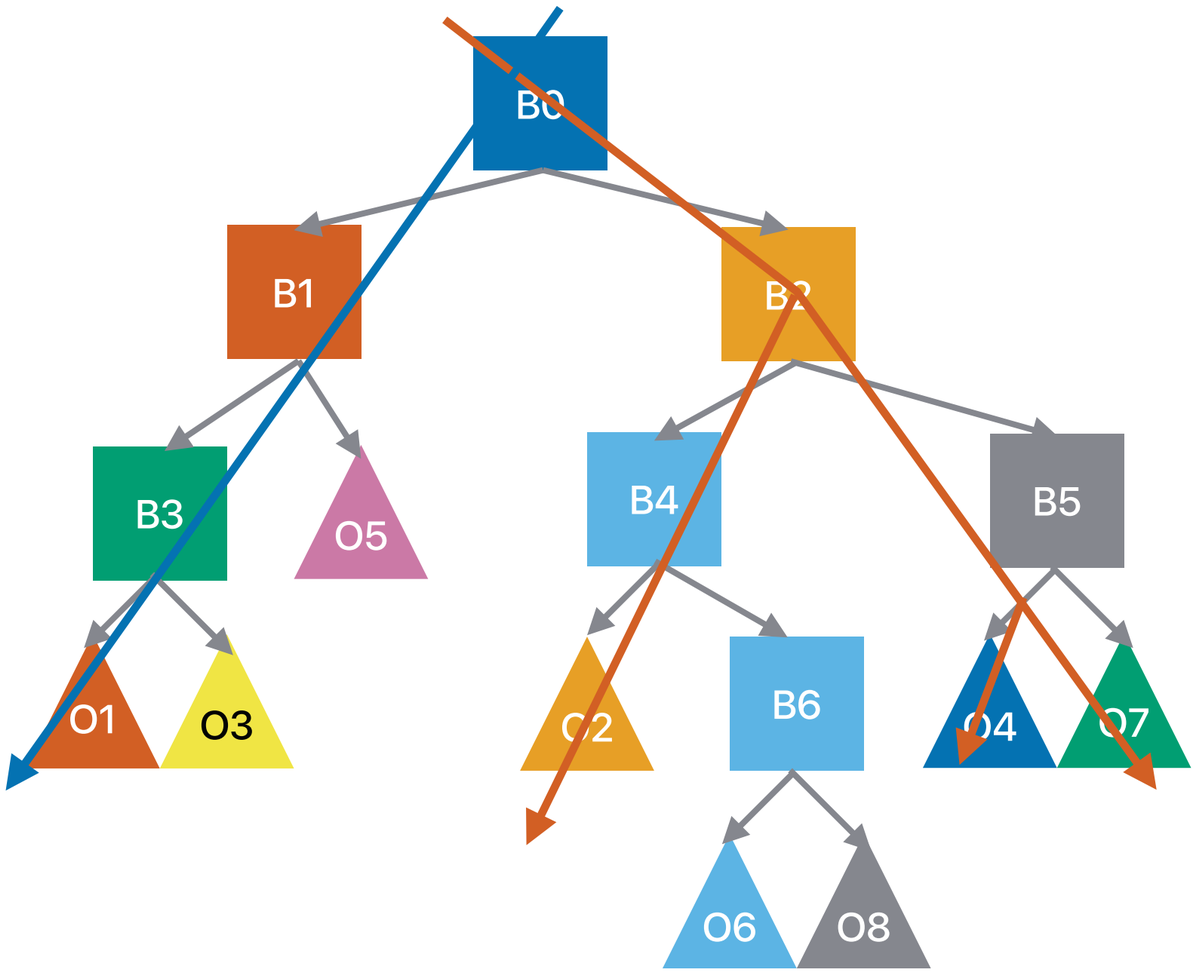
https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/raytracing-algorithm-in-a-nutshell.html

Recap: Ray tracing pipeline

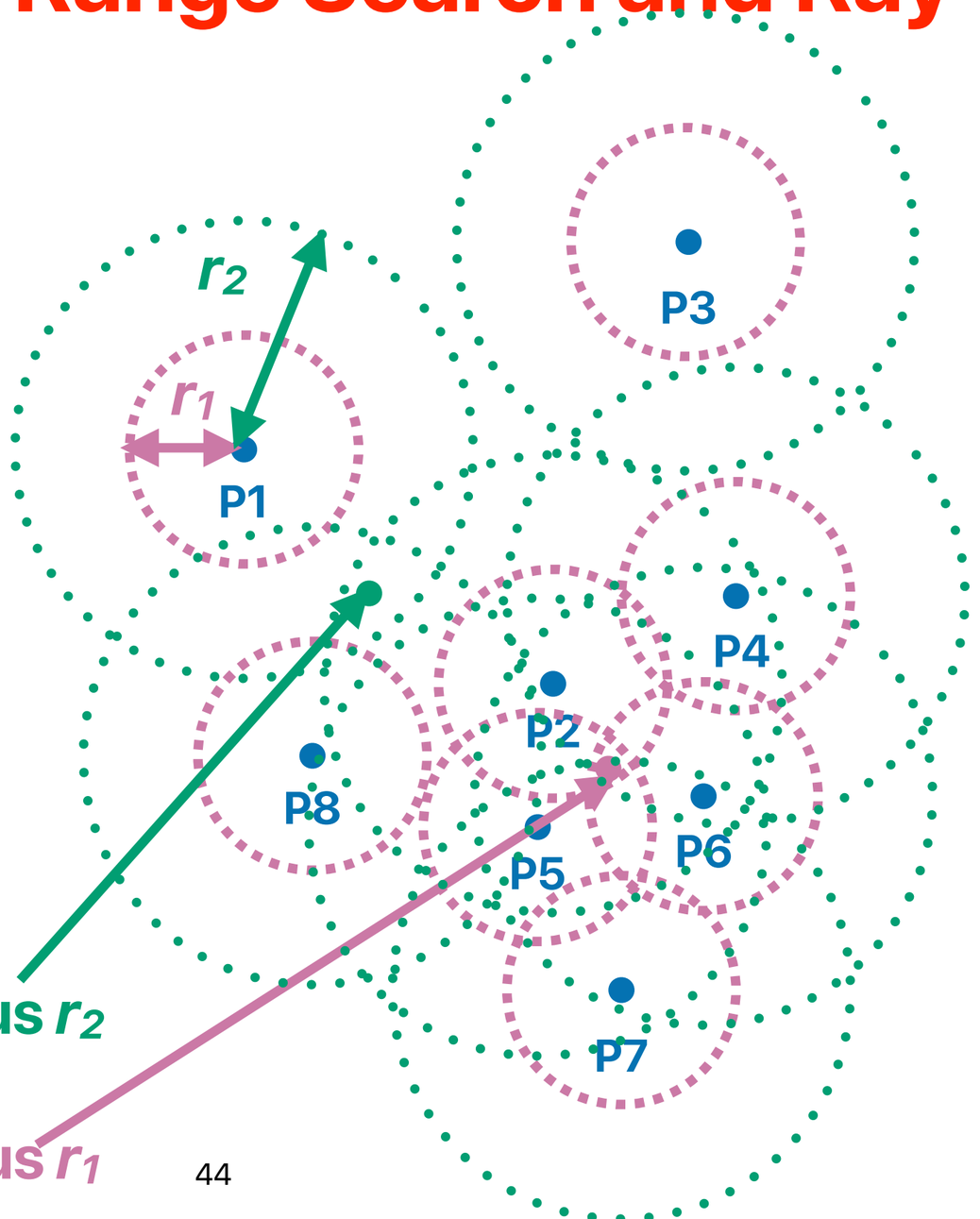


Accelerated search structure: bounding volume hierarchy tree





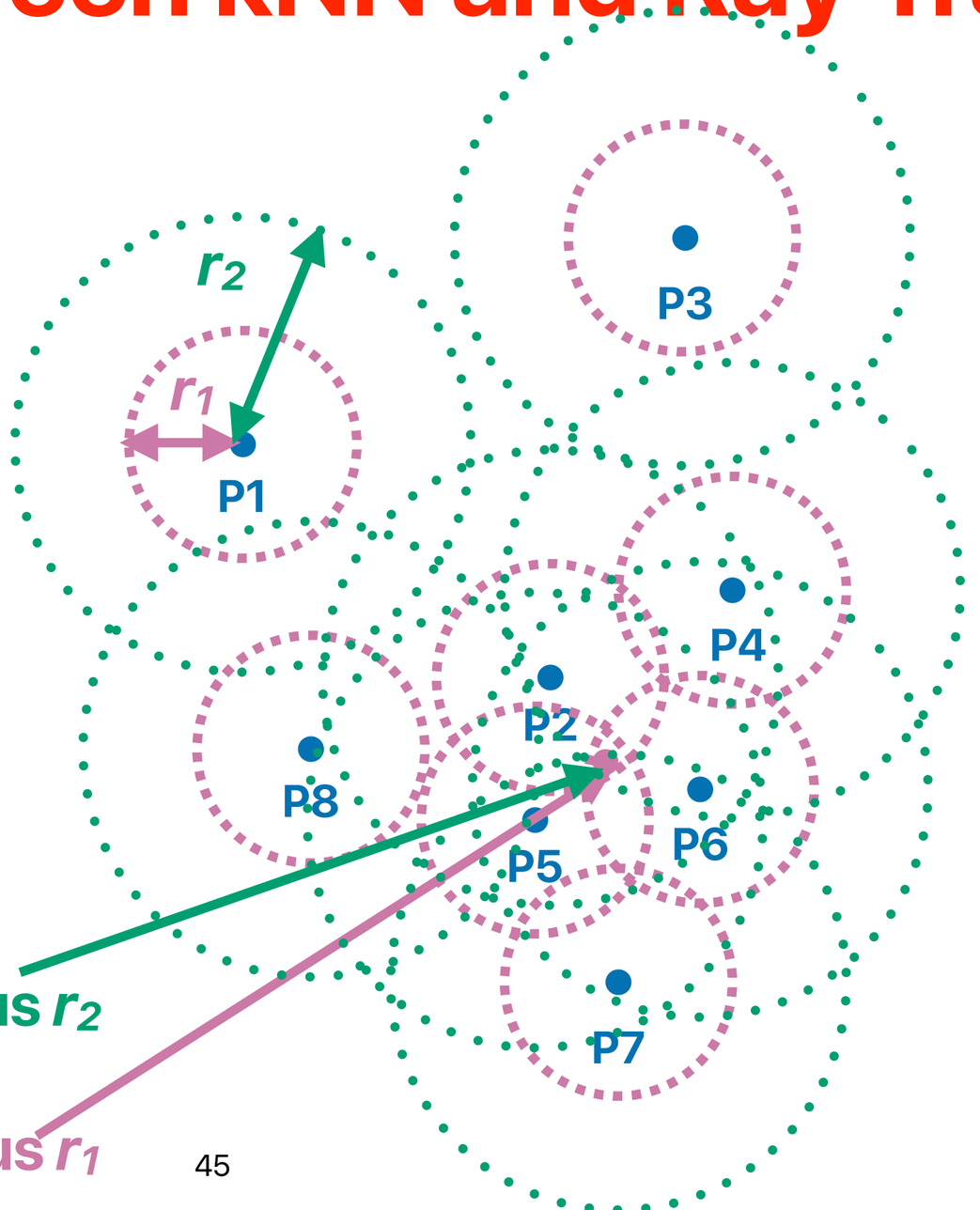
Similarities between Range Search and Ray Tracing



Answer: P1, P2, P8 Q_2 with radius r_2

Answer: P2, P5, P6 Q_1 with radius r_1

Similarities between kNN and Ray Tracing



Assume we want to find "5" NN

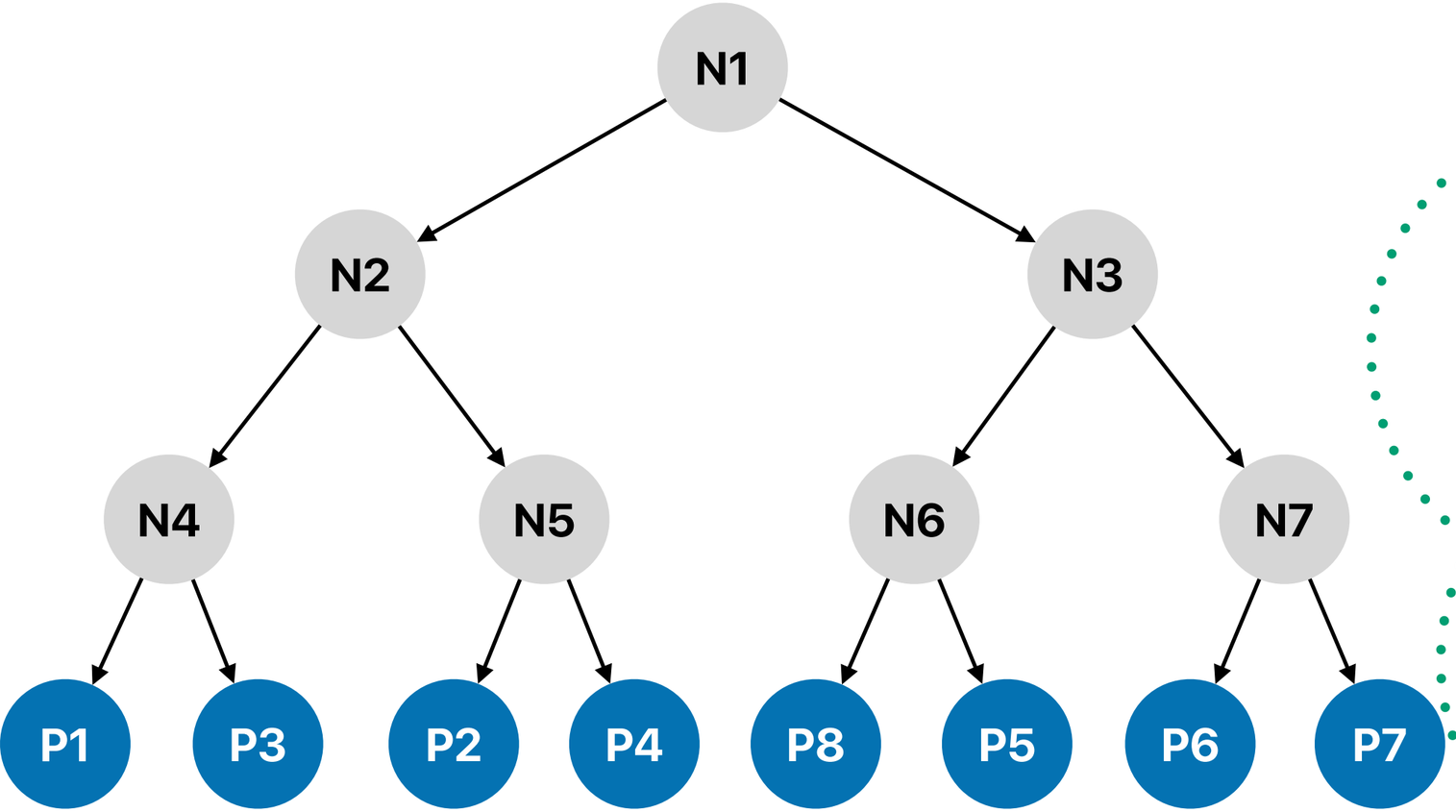
Answer: P2, P4, P5, P6, P7

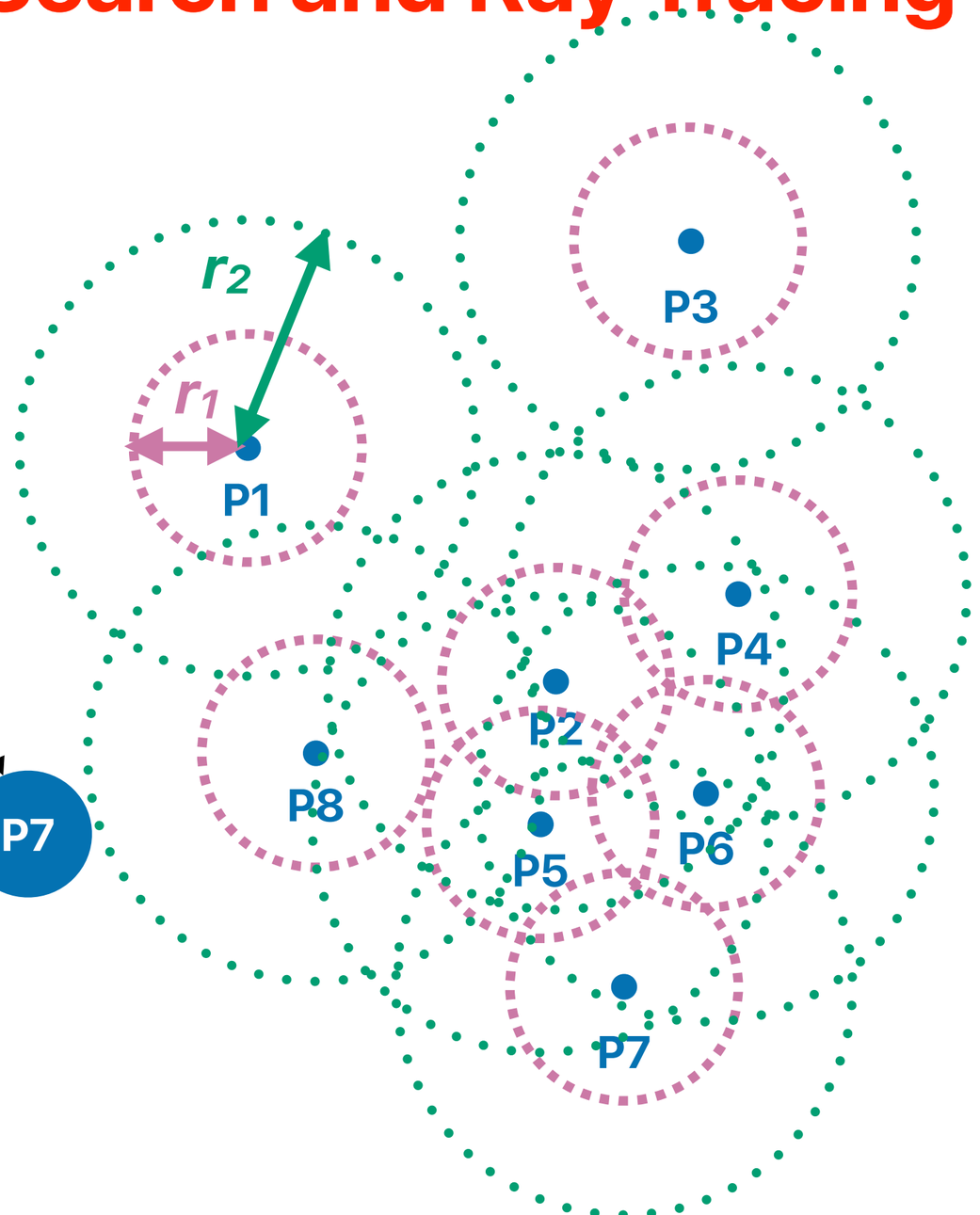
Q_1 **with radius** r_2

Answer: P2, P5, P6

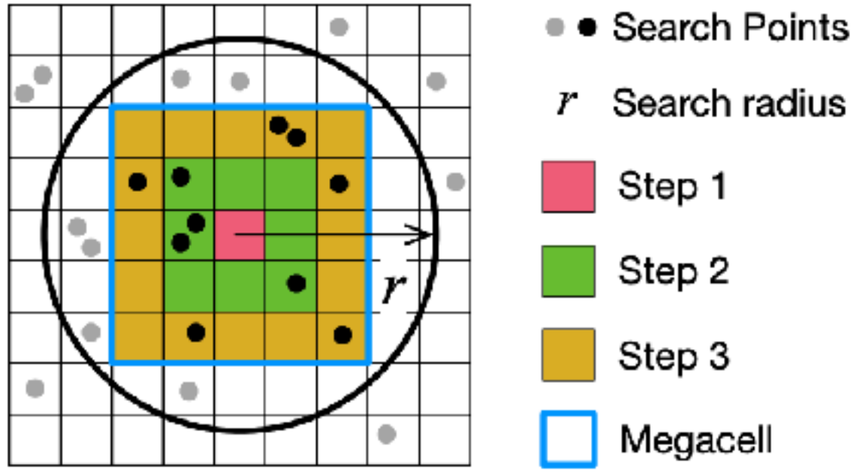
Q_1 **with radius** r_1

Similarities between Range Search and Ray Tracing

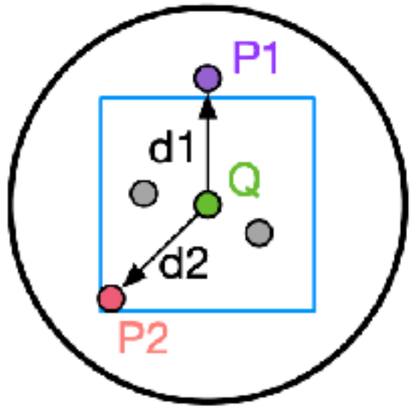




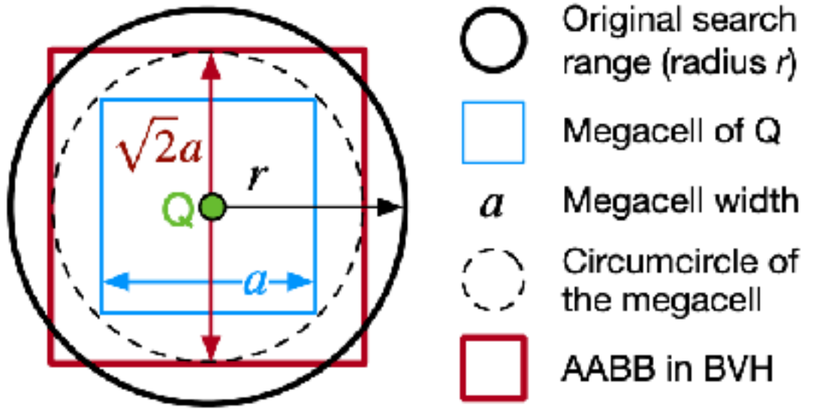
Ideas behind RTNN



(a) Calculating the megacell of a query by incrementally growing from the cell that contains the query. The growth stops when either the sphere boundary is reached or at least K neighbors are found.



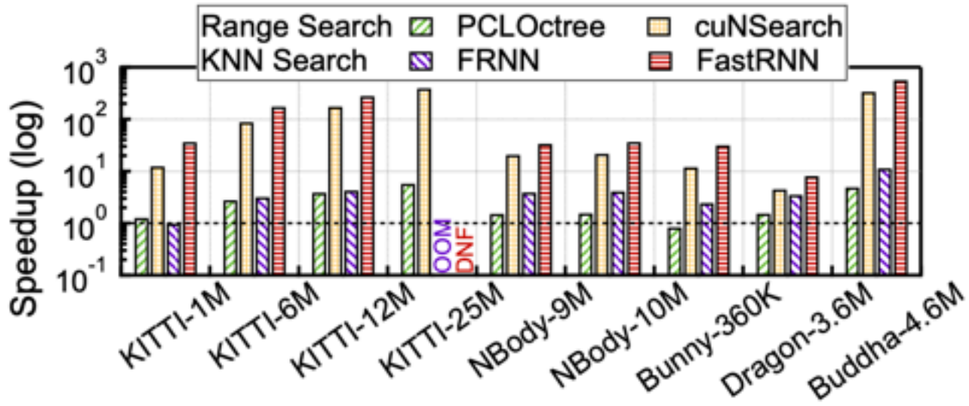
(b) $P1$ is outside of Q's megacell that contains 3 points, but is among the 3 nearest neighbors of Q (as $d1 < d2$).



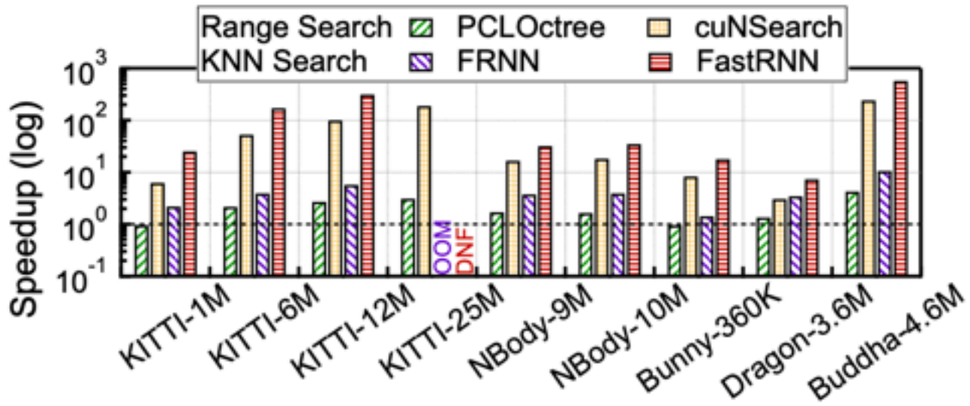
(c) For KNN search, the AABB must circumscribe the sphere that circumscribes the megacell. The AABB width is $\sqrt{2}a$ for 2D search (illustrated here) and $\sqrt{3}a$ for 3D search (as the AABB is 3D and the circumcircle becomes the circumsphere), where a is the megacell width.

Fig. 10. Determining the megacell and AABB size used for neighbor search.

Performance



(a) Speedups on RTX 2080.



(b) Speedups on RTX 2080Ti.

Fig. 11. Speedup of RTNN over the baselines (log-scale). OOM denotes that the baseline ran out of memory; DNF denotes that the baseline did not finish within the time that would have given RTNN a 1,000 speedup.

Summary of General-purpose computing on “something”

- Map your problem as a problem that the target hardware can solve
  - Machine learning models (NPU)
  - Machine learning mathematical operations (GP Edge TPU)
  - Matrix multiplications (TCUDB)
  - Ray Tracing (RTNN)
- Due to the domain specific nature of accelerators, we have to program in a more domain specific way, “currently”.
- Not necessarily the most performant, but typically more energy-efficient

Recap: Summary of General-purpose computing on "something"

- Map your problem as a problem that the target hardware can solve
  - Machine learning models (NPU)
  - Machine learning mathematical operations (GP Edge TPU)
  - Matrix multiplications (TCUDB)
  - Ray Tracing (RTNN)
- Due to the domain specific nature of accelerators, we have to program in a more domain specific way, "currently".
- Not necessarily the most performant, but typically more energy-efficient

Roogle Project Presentations

- Make an appointment on 4/29 and 5/1 through the Google Calendar
- 15 minute presentation with 3 minute Q & A
- Why & what & how!!! — considering you're giving a presentation at Apple's keynote
  - 7-minute why — why should everyone care about this problem? Why is this still a problem?
  - 5-minute what — what are you proposing in this project to address the problem?
  - 3-minute how — expected platforms/engineering efforts, milestones and workload distribution among members
  - Please reference this article to make a good presentation https://cseweb.ucsd.edu//~swanson/GivingTalks.html

Electrical
Computer Science
Engineering

277

つづく