



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386



Machine Learning Accelerators in Bioinformatics

Kazem Shekofteh

Post-Doctoral Research Fellow, ZITI Fellow

Computing Systems Group, Institute of Computer Engineering, Heidelberg University

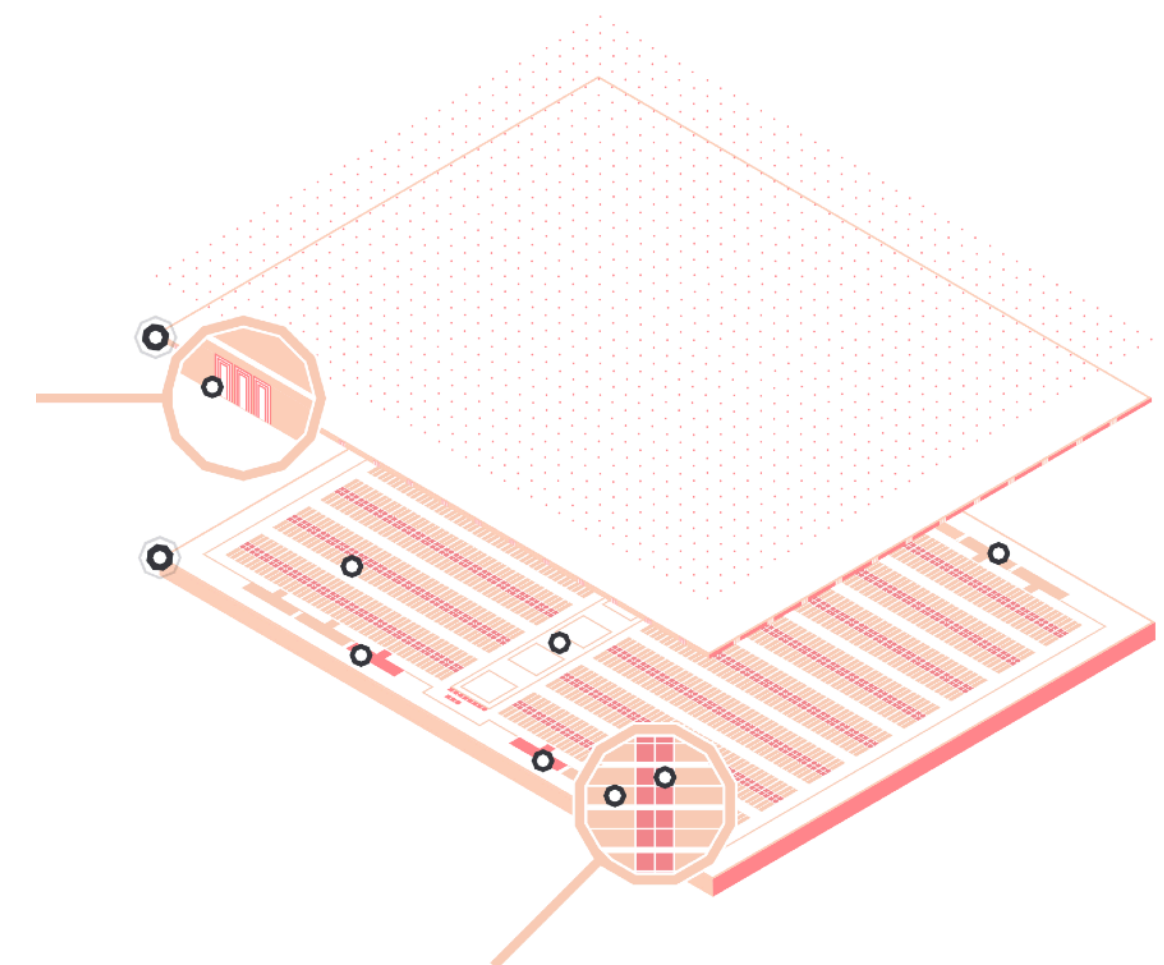
Sino-German Workshop on Multi-Physics Device Simulation and hardware-aware Computing

Oct 10-15, Xi'An, China

Intelligence Processing Units

Introduction and History

- Early 2010: Need for specialized hardware in AI accelerates, with companies like Graphcore leading the way.
- 2017: Graphcore introduced the first IPU for AI/ML workloads (Colossus™ MK1 - GC2 IPU).
- 2020: Colossus™ MK2 - GC200 IPU
- 2022: Graphcore and TSMC presented the Bow IPU, a 3D package of a GC200 die bonded face to face to a power-delivery die that allows for higher clock rate at lower core voltage



Intelligence Processing Units

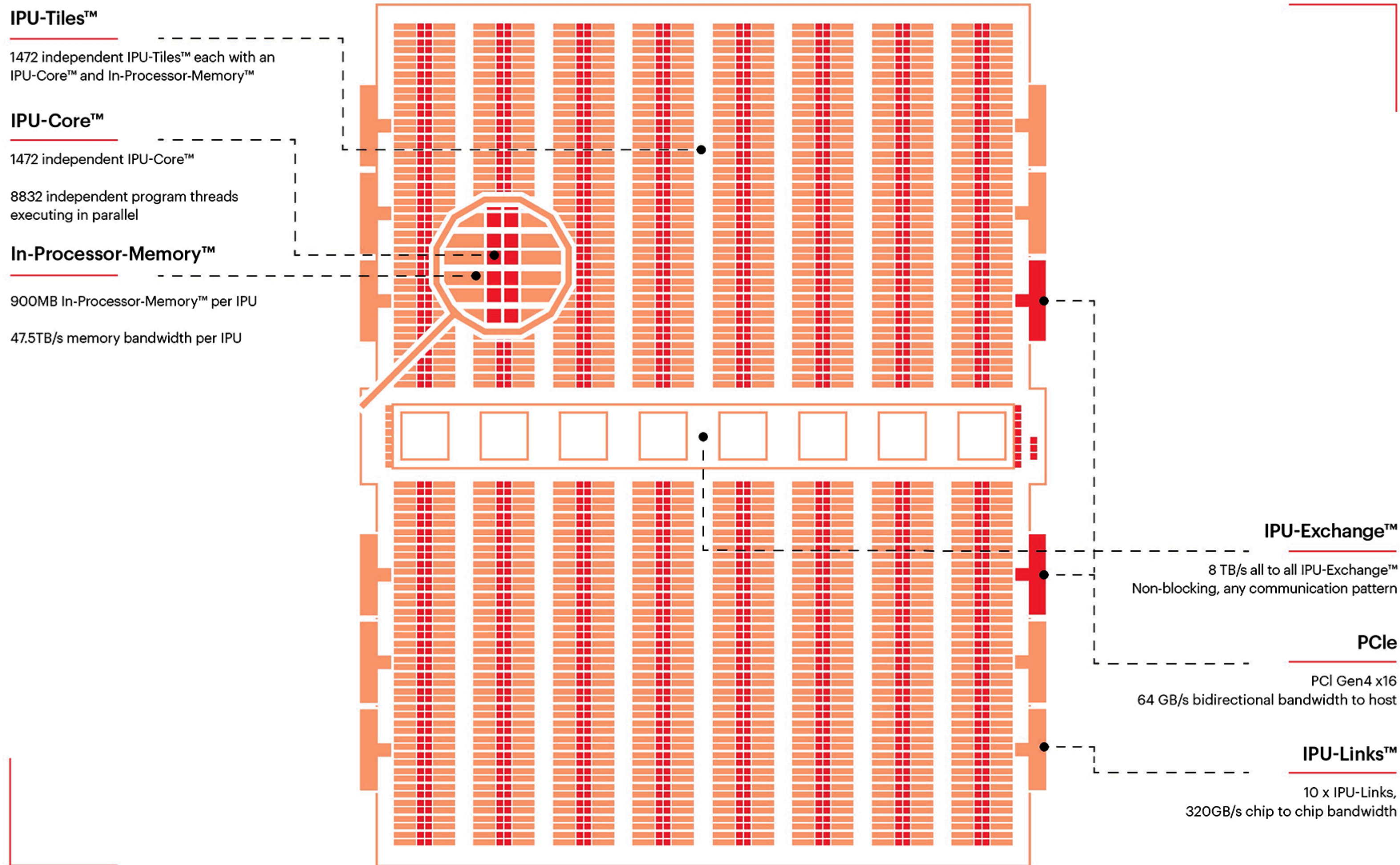
Architecture

- Key components:
 - Massively Parallel Processor Cores: Thousands of small, independent cores designed to handle parallel tasks simultaneously.
 - Memory Architecture: IPU's have on-chip memory designed to provide ultra-fast access to data for efficient AI training/inference.
 - Fine-Grained Processing: Suited for sparse data sets and small computational graphs, unlike GPUs which are optimized for dense, large-scale computation.
- Key Features:
 - Low-latency, high-bandwidth (SRAM) memory access.
 - Optimized for machine learning models, including graph-based models.
 - Were developed for AI applications but showed potential for other HPC applications



Intelligence Processing Units

Architecture



- 1472 Tiles per GC200
- 600KB Memory per Tile (On-Chip)
- 900MB Memory per IPU
- Only Tile-local Memory access
- Up to 6 Threads per Tile
- 250 TFLOPs of peak FP16
- 62.5 TFLOPs of peak FP32
- 150W TDP
- 1.325 GHz
- 11 TB/s all-to-all IPU-Exchange
- 320 GB/s chip-to-chip BW

Intelligence Processing Units

CPU vs. GPU vs. IPU

Parallelism

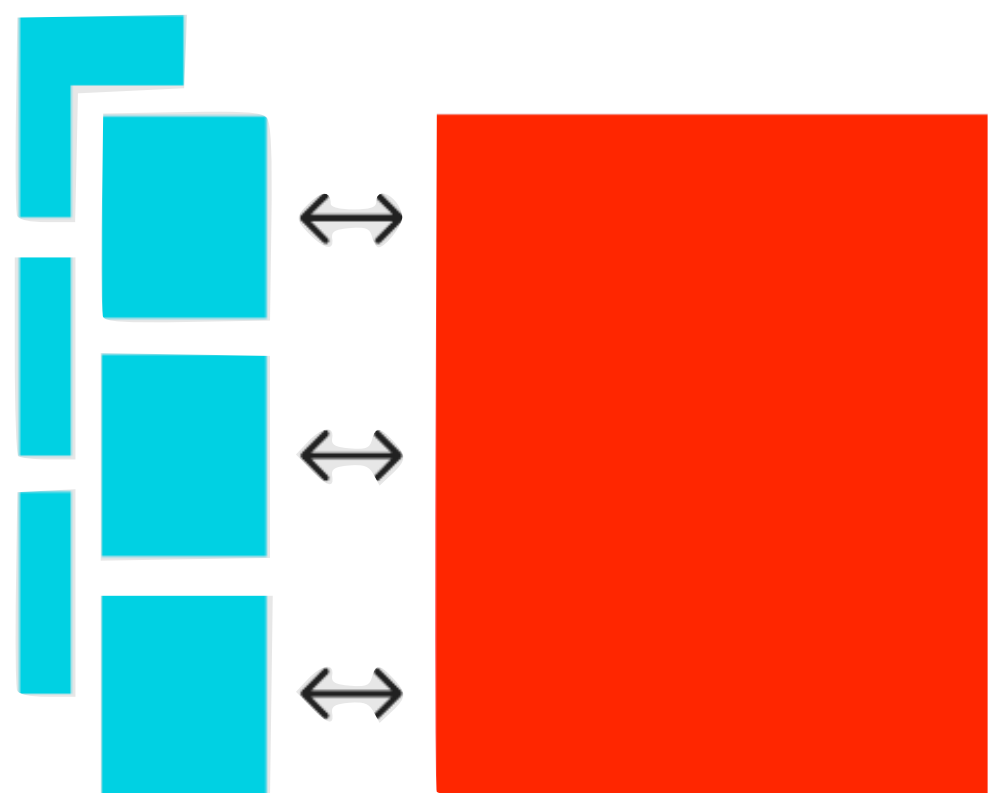
Processors

Memory



CPU

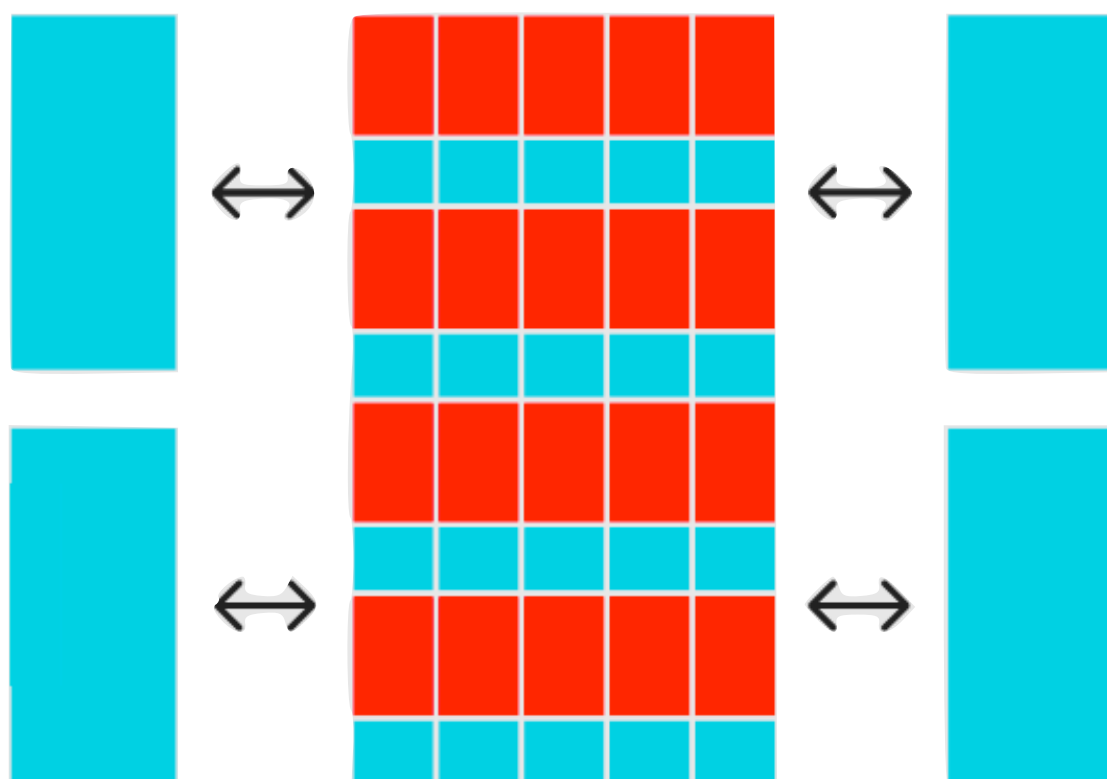
Designed for scalar processes



Off-chip memory

GPU

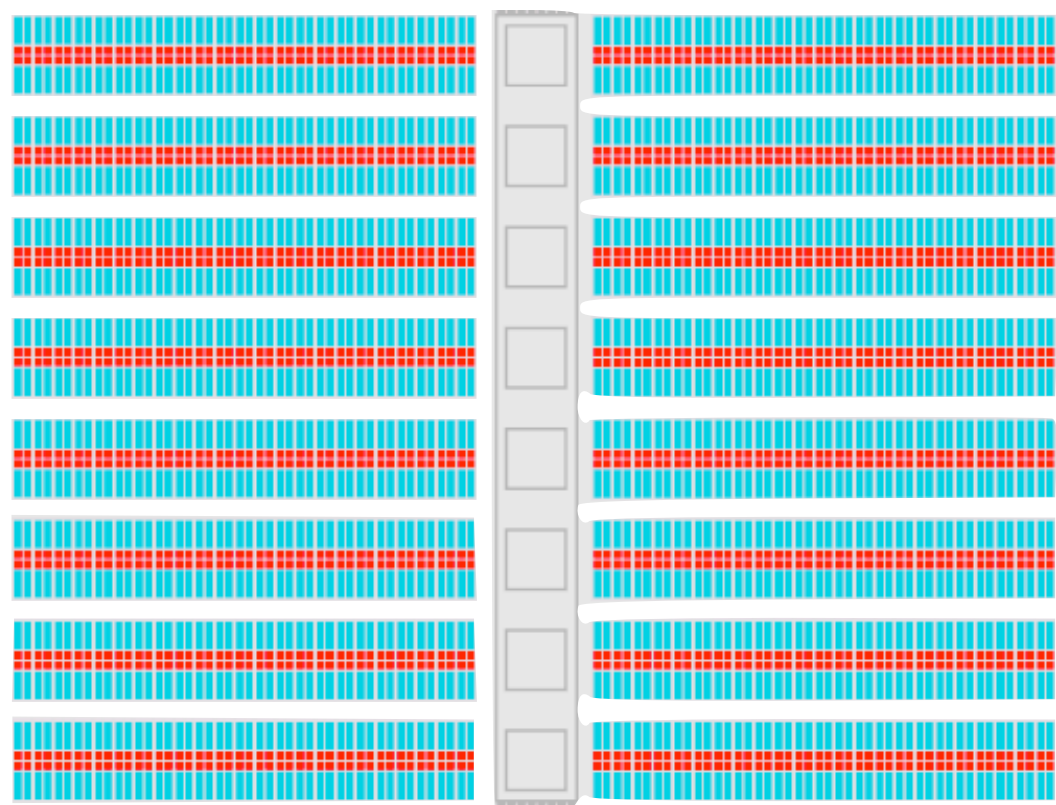
SIMD/SIMT architecture.
Designed for large blocks of
dense contiguous data



Model and data spread across
off-chip and small on-chip cache,
and shared memory

IPU

Massively parallel MIMD.
Designed for fine-grained, high-
performance computing



Model and data tightly coupled,
and large locally distributed
SRAM

Memory Access

Intelligence Processing Units

IPU Chip vs. GPU chip



Chip	GC200	A30
Number of cores	1472	3584
Total SRAM	918 MB	10.75 MB
DRAM bandwidth	20 GB/s	933 GB/s
Clock frequency	1.33 GHz	1.44 GHz
FP32 peak compute	62.5 TFlops/s	10.3 TFlops/s
Power Consumption	150 W	165 W
Inter-Chip-Bandwidth	350 GB/s	200 GB/s

IPU Machine: M2000

4 x Colossus™ GC200 IPU
1 petaFLOPS AI compute
Up to 260GB Exchange Memory™
- 256GB Streaming Memory™
- 3.6GB In-Processor-Memory™
2.8Tbps IPU-Fabric™

59.4Bn transistors, TSMC 7nm @ 823mm²
250 teraFLOPS AI compute
1472 independent processor cores
8832 separate parallel threads

Arm Cortex-A quad-core SoC
Super low latency IPU-Fabric™ interconnect

Board Management Controller

M.2 Slot

PCIe FH3/4L G4x8 Slot
(RNIC/SmartNIC)

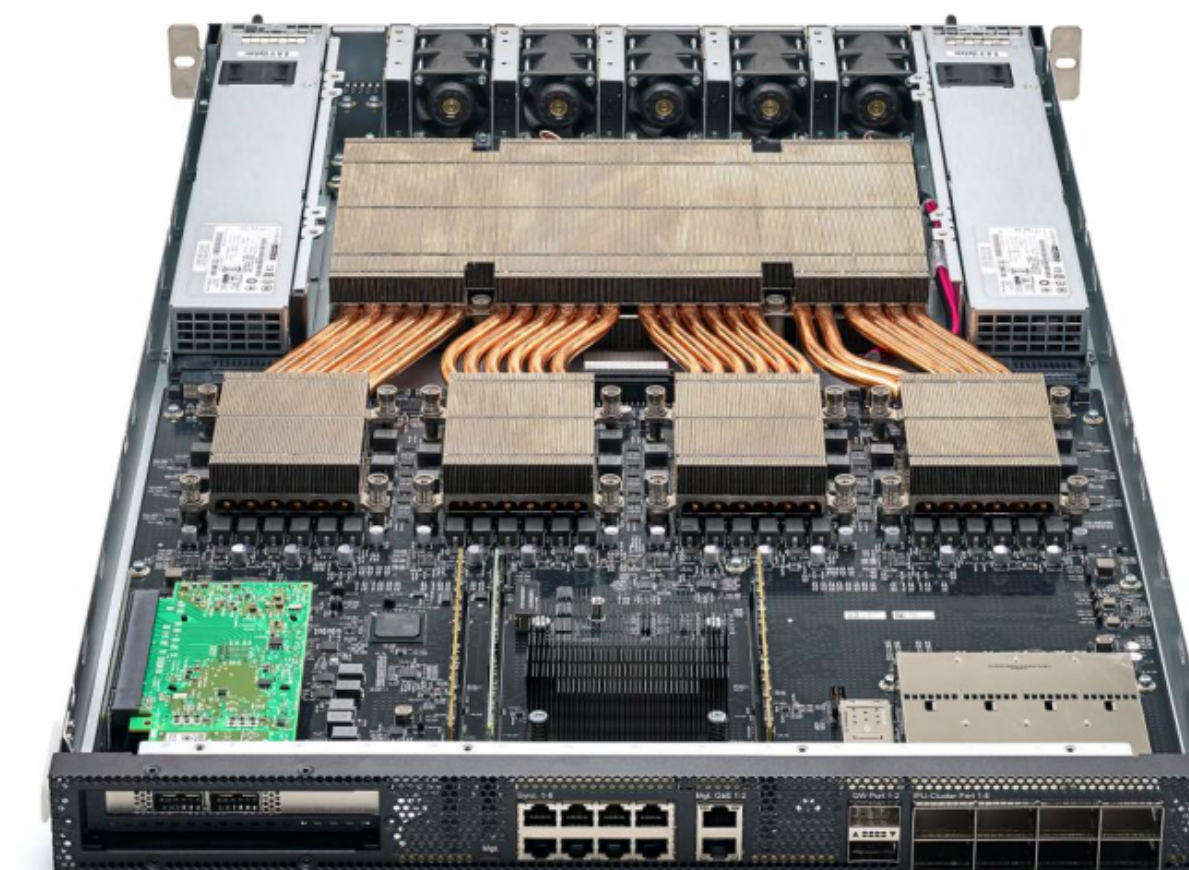
DDR4 DIMM DRAM x 2

Advanced air cooling system

Power Supply Unit (x2)

Ultra compact 1U server chassis



eMMC 32G Flash device



- Four GC200 IPU
- Totally 1 PFlops of AI compute
- 3.6GB On-Chip Memory
- Up to 256GB Streaming Memory
- Scales up to 64K-IPUs
- 512Gbps inter IPU BW

Intelligence Processing Units

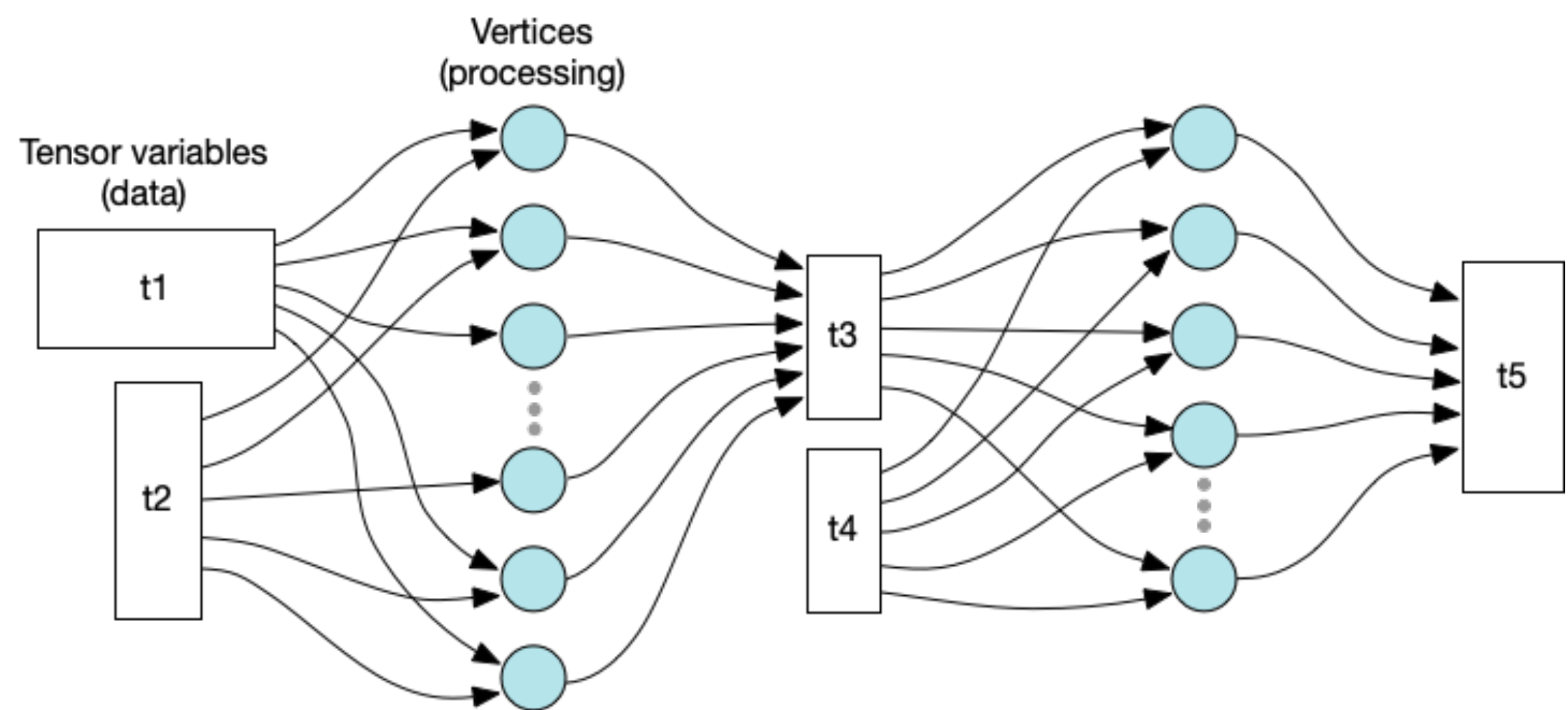
NVIDIA DGX-A100 vs. COLOSSUS MK2

	 NVIDIA DGX-A100 (8x A100)	 GRAPHCORE 8x M2000
FP32 compute	156TFLOP	2PFLOP
AI compute	2.5PFLOP ^[1]	8PFLOP ^[2]
AI Memory	320GB ^[3]	3.6TB ^[4]
System Price	\$199,000 _{MSRP}	\$259,600 _{MSRP}

NOTES:
[1] Actual figure for TF32/FP16. NVIDIA 8xA100 5PFlop reference is for 50% sparsity which includes Pfllops for operations that aren't run
[2] Graphcore AI Float with IEEE FP16.16 multiply.accumulate and IEEE FP16.SR 16bit float with stochastic rounding, with equivalent accuracy performance as FP32
[3] 40GB HBM memory on A100 modules *8 modules per DGX-A100 system
[4] IPU-Exchange Memory which includes attached DRAM and IPU In-Processor-Memory with 100x bandwidth vs. HBM memory sub-system

Intelligence Processing Units

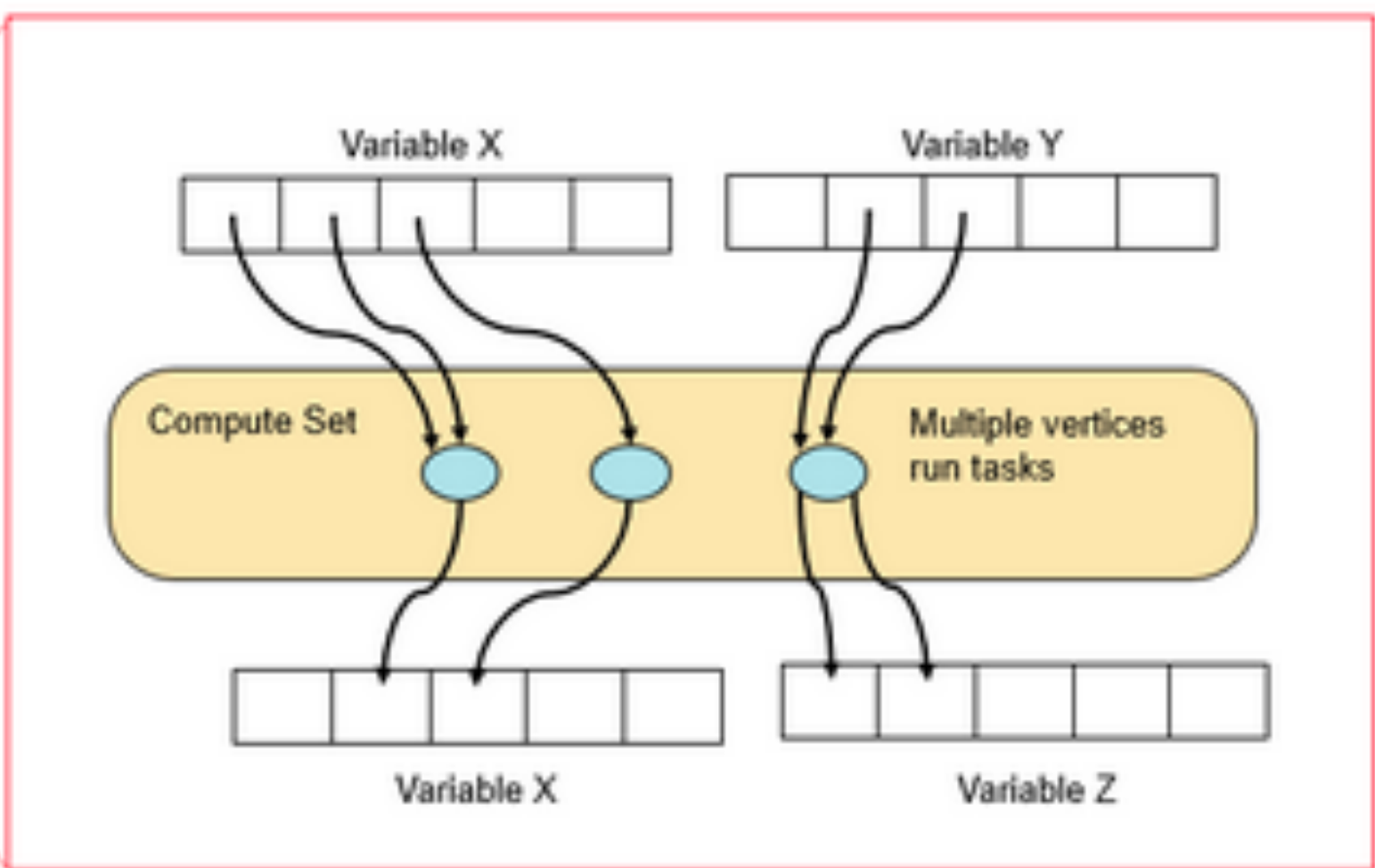
Programming Model



...executes a program..

```
while(v1 < v2):  
  copy(t1, t2);  
  execute(cs1);  
  if (v3 < v4):  
    copy(t2, t3);  
  else:  
    copy(t3, t2);  
  execute(cs2);
```

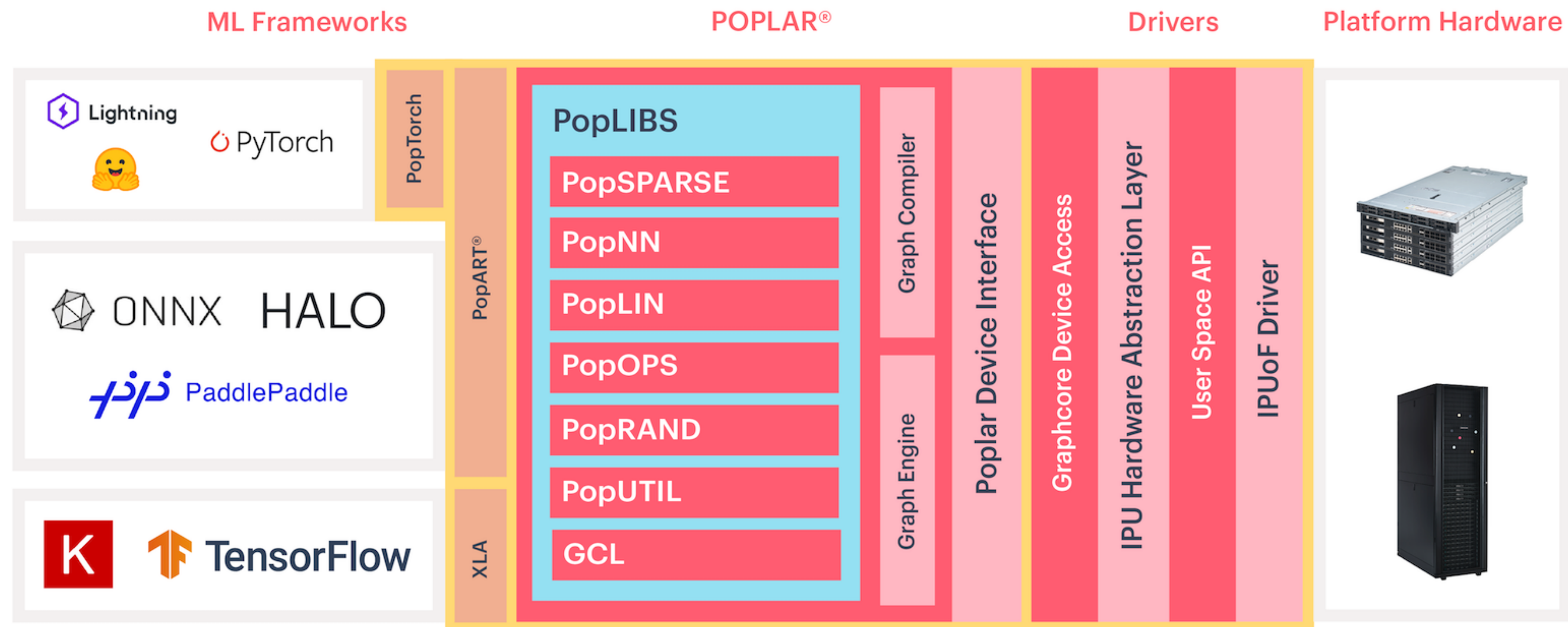
...which manipulates multi-element variables
via highly parallel compute sets of vertices



Intelligence Processing Units

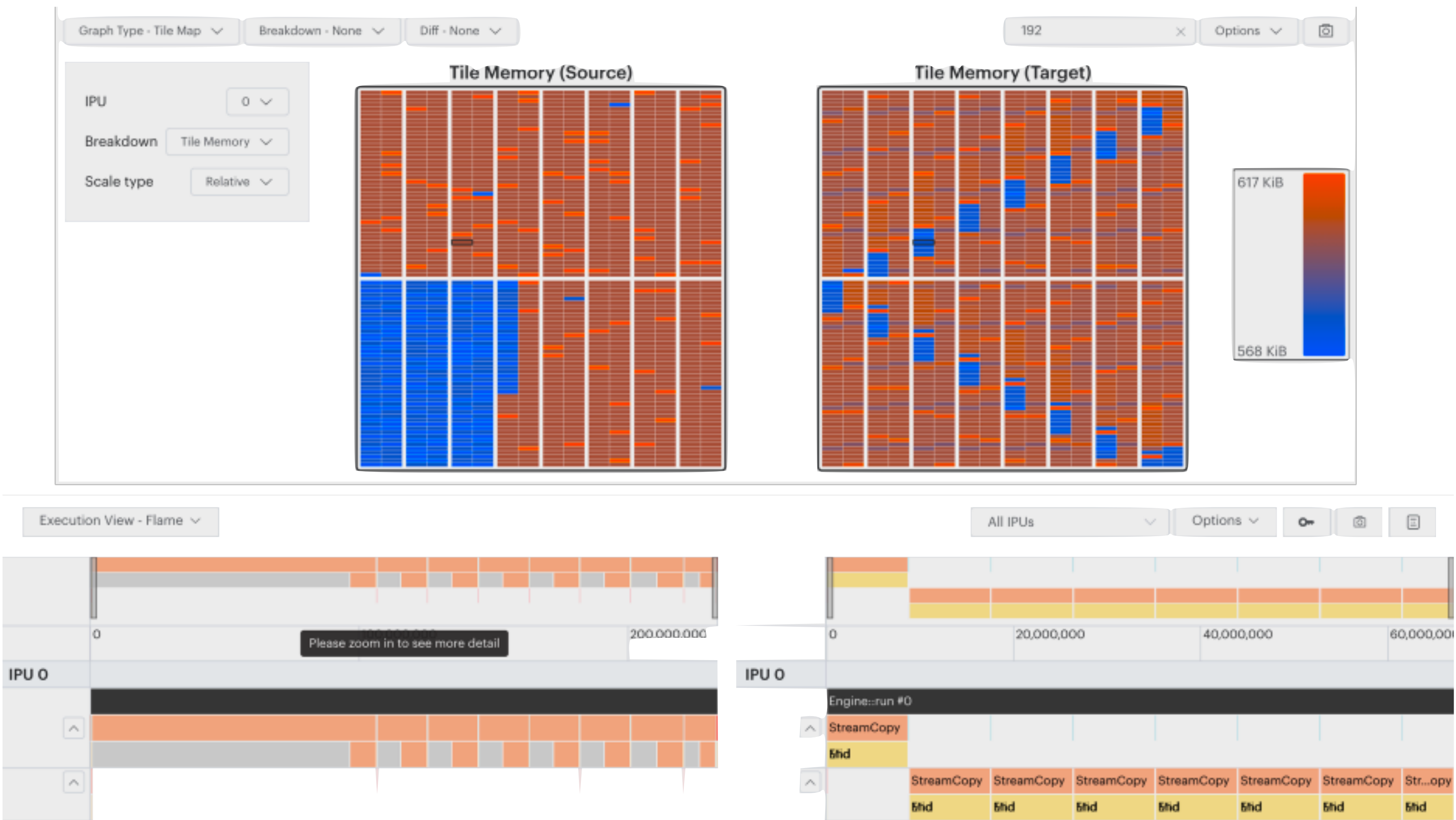
Programming Model: Poplar SDK

- A comprehensive software development toolkit designed specifically for programming and optimizing applications on Graphcore IPU.



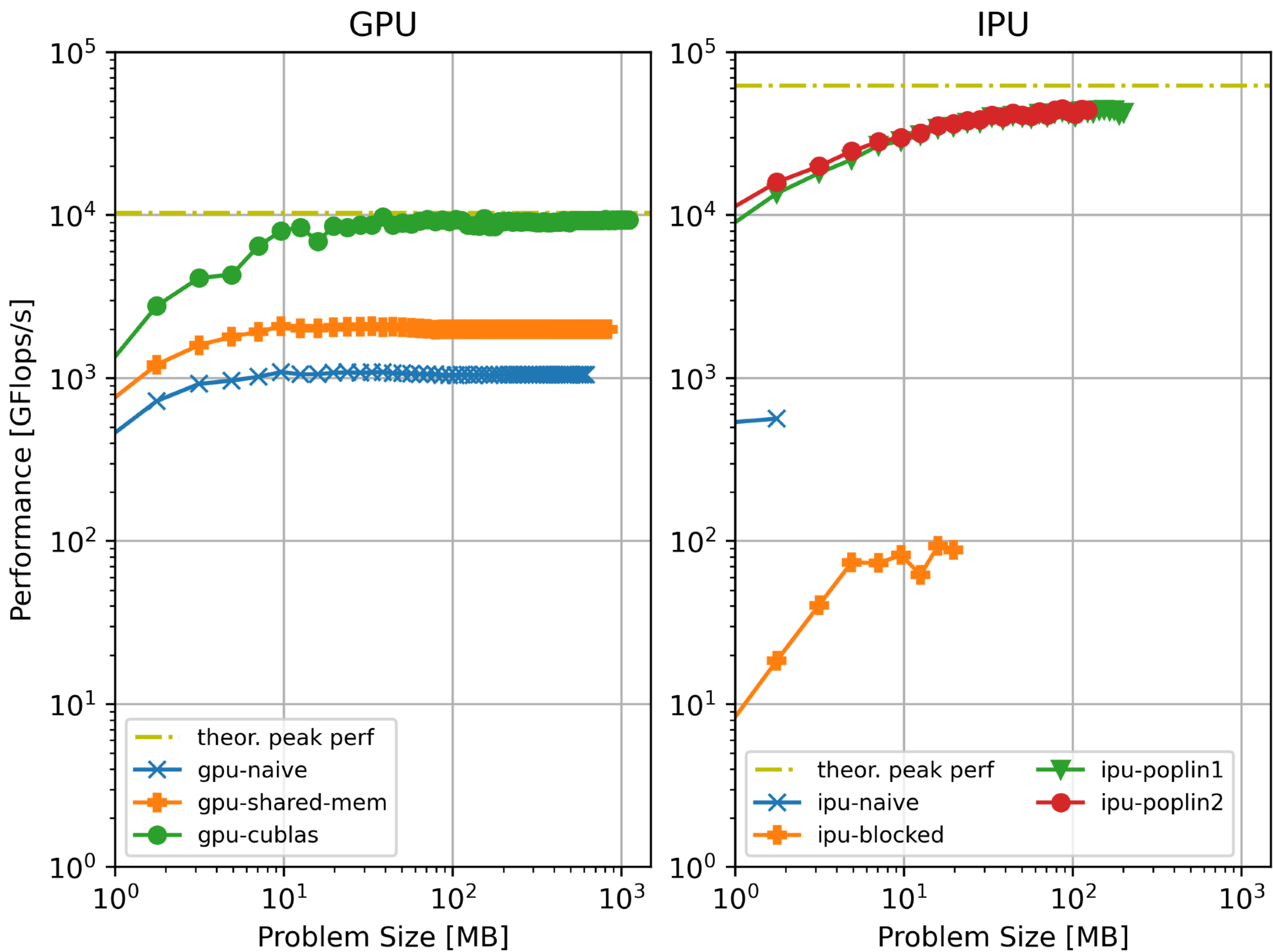
Intelligence Processing Units

Tools: PopVision Graph Analyzer



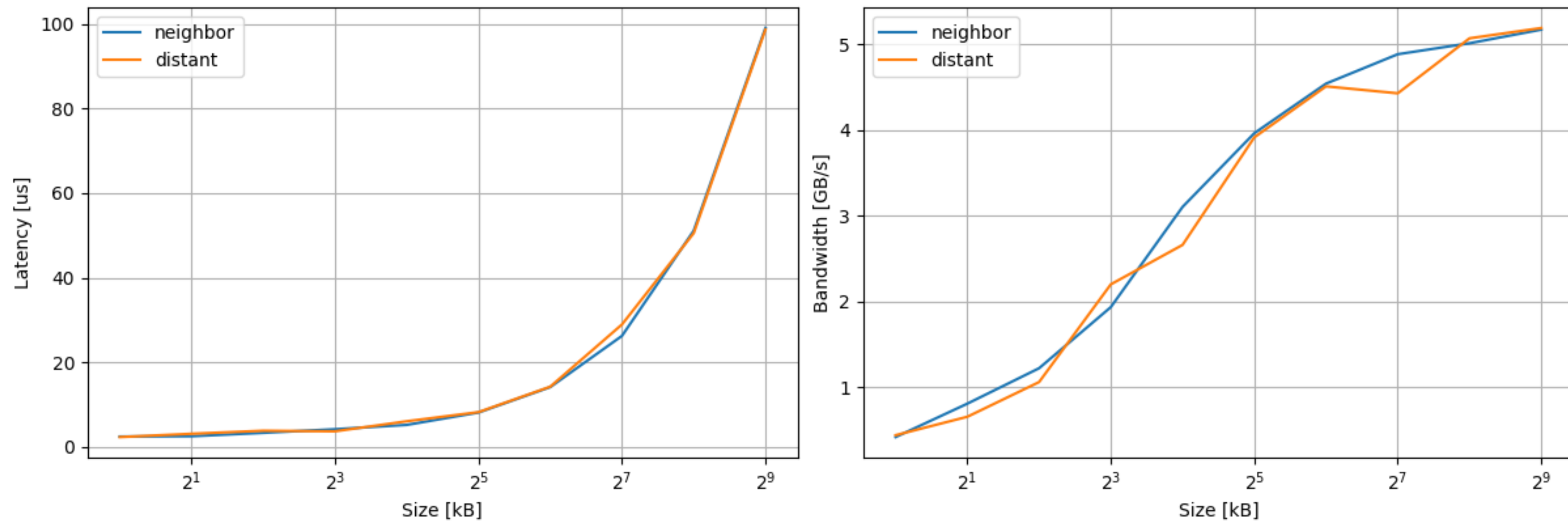
IPU Performance Analysis

Motivation Example: MatMul A30 vs GC200



IPU Performance Analysis

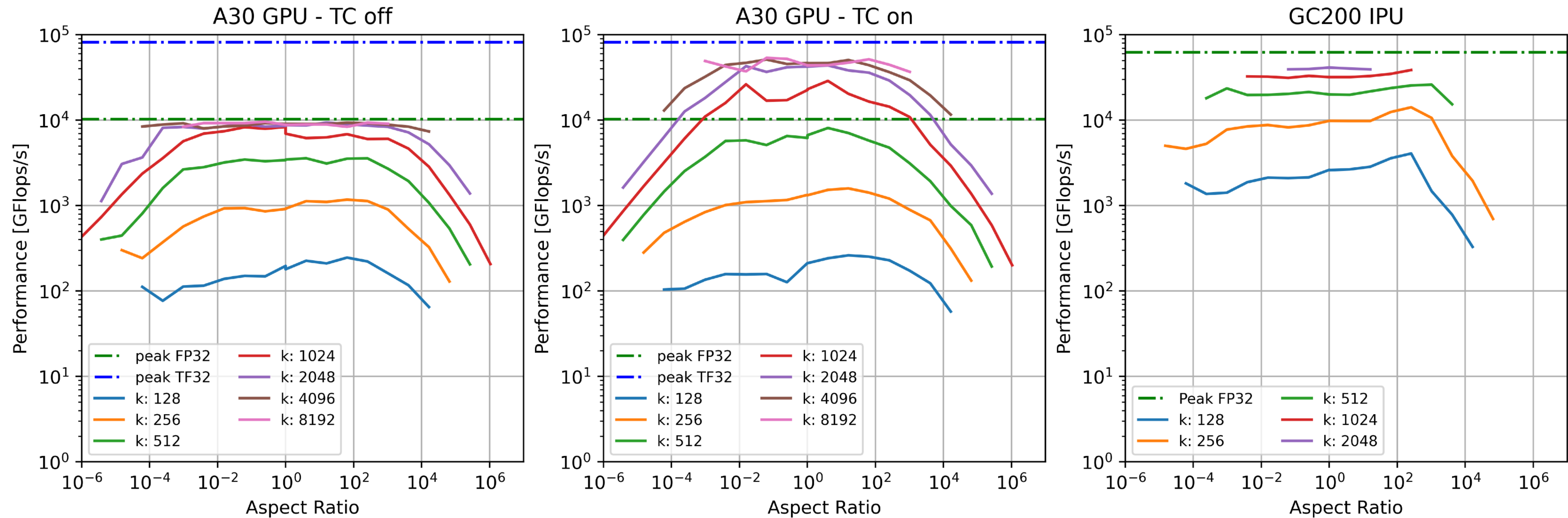
Locality in communication (IPU-Exchange)



Observation 1: Latency and bandwidth of data accesses in between different IPU-Tiles are tightly coupled with data size, but are independent of their location.

IPU Performance Analysis

Squared and Skewed MatMul

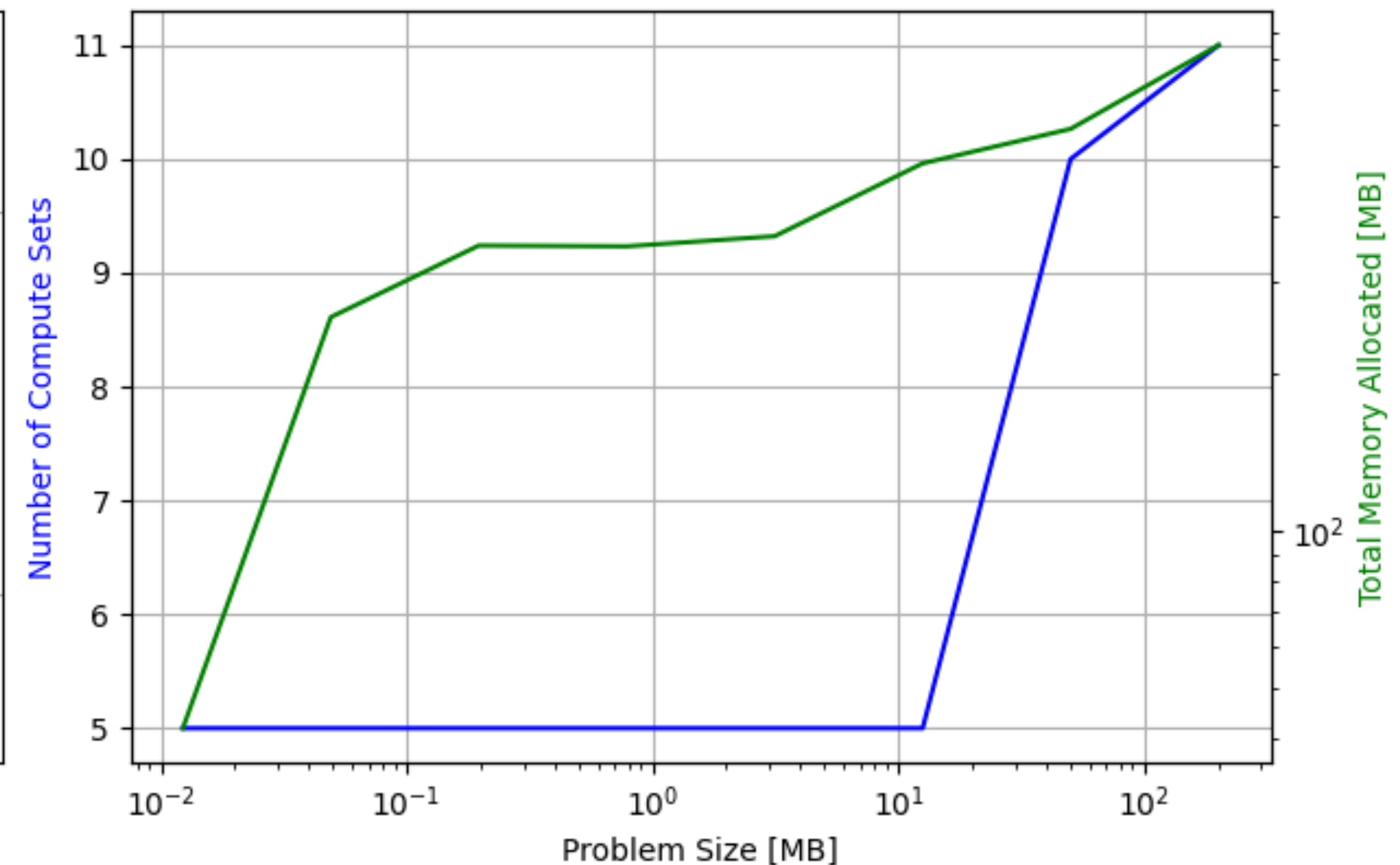
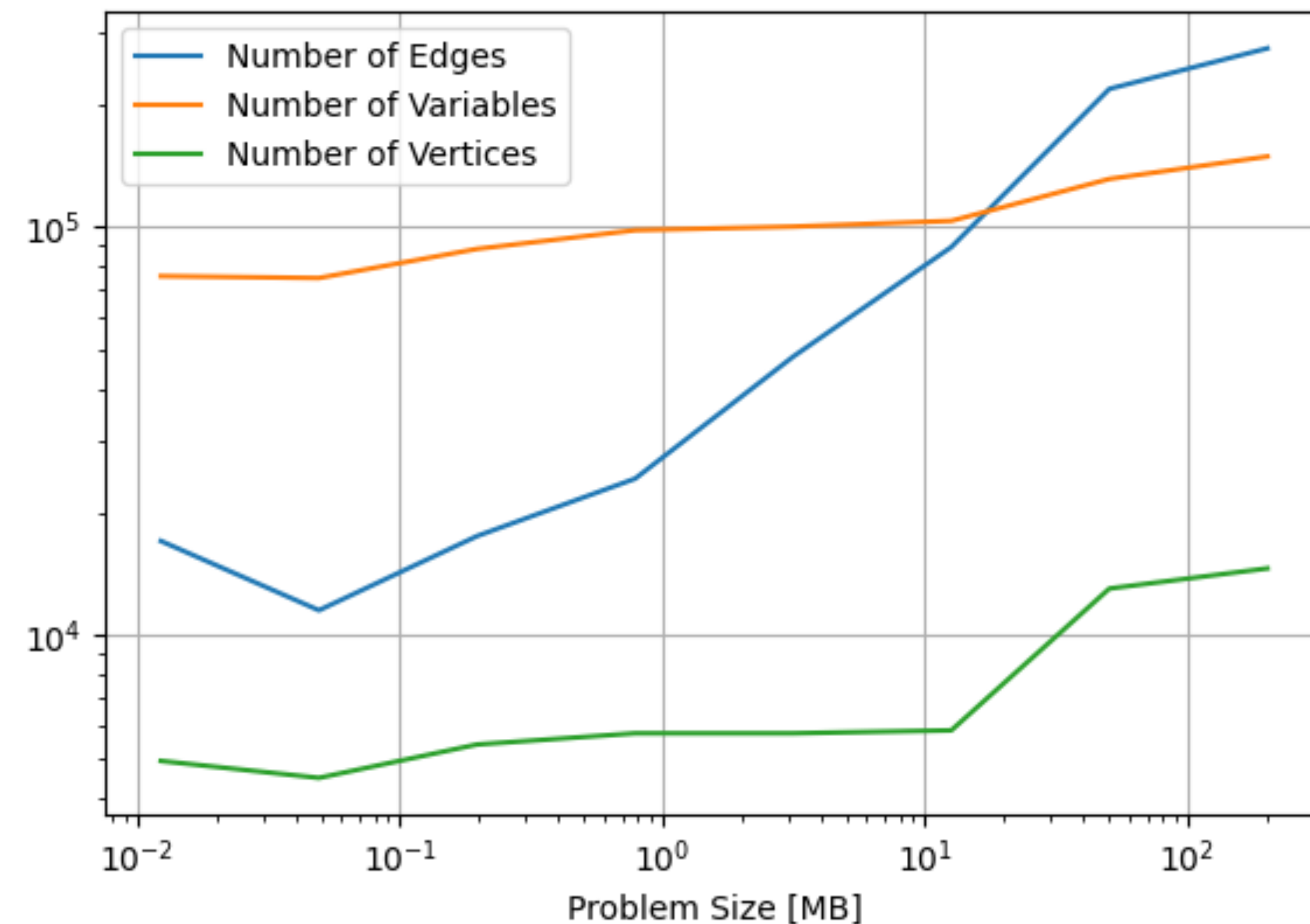


Observation 2: Evaluation results show promising performance for the IPU in different scenarios, especially for linear algebra operations based on skewed matrices or sparse matrices.

$$A(m \times n) \times B(n \times k) = C(m \times k) : skewness = \frac{m}{n}$$

IPU Performance Analysis

Memory usage for the IPU



Observation 3: *The overall memory usage for the IPU does not only depend on the problem size, but there are additional effects with substantially increase in overall memory usage.*

Intelligence Processing Units

Our Past and Ongoing Research on IPUs

- S.-Kazem Shekofteh, Christian Alles, and Holger Fröning. 2023. Reducing Memory Requirements for the IPU using Butterfly Factorizations. 14th IEEE International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems @ SCS'23, Denver, CO
- On Performance Analysis of Graphcore IPUs: Analyzing Squared and Skewed Matrix Multiplication, K. Shekofteh, C. Alles, N. Kochendörfer, H. Fröning - arXiv preprint arXiv:2310.00256, 2023
- Distributed Butterfly Machine Learning with IPUs
- X-Drop Sequence Alignment Implementation on IPUs
- MIMD Kernels for Sequence Alignment on IPUs

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

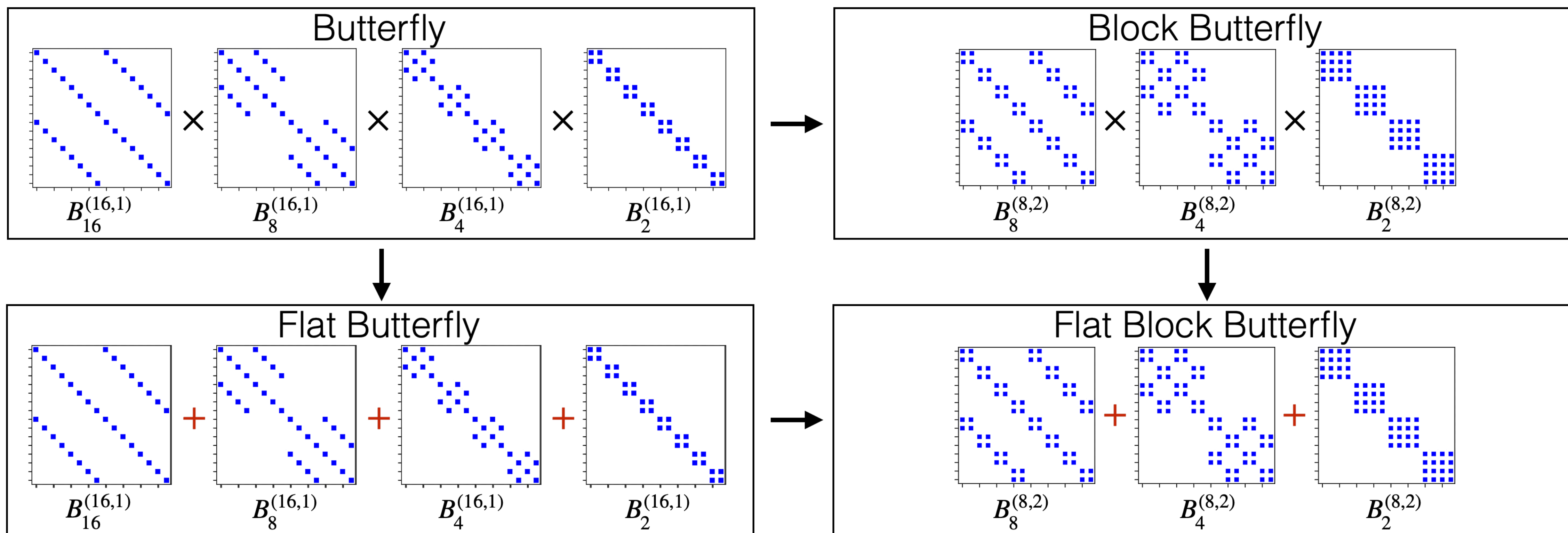
S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

- Butterfly factorization
 - a technique used in applied mathematics to efficiently factorize a matrix into a product of **sparse matrices**.
 - accelerates matrix computations (matrix-vector multiplications)
 - reducing computational complexity and memory requirements
 - commonly used in fast algorithms like FFT
- Dao et al. [1]: Butterfly matrices: replacing specific transformations by universal building blocks called **butterfly factors**

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

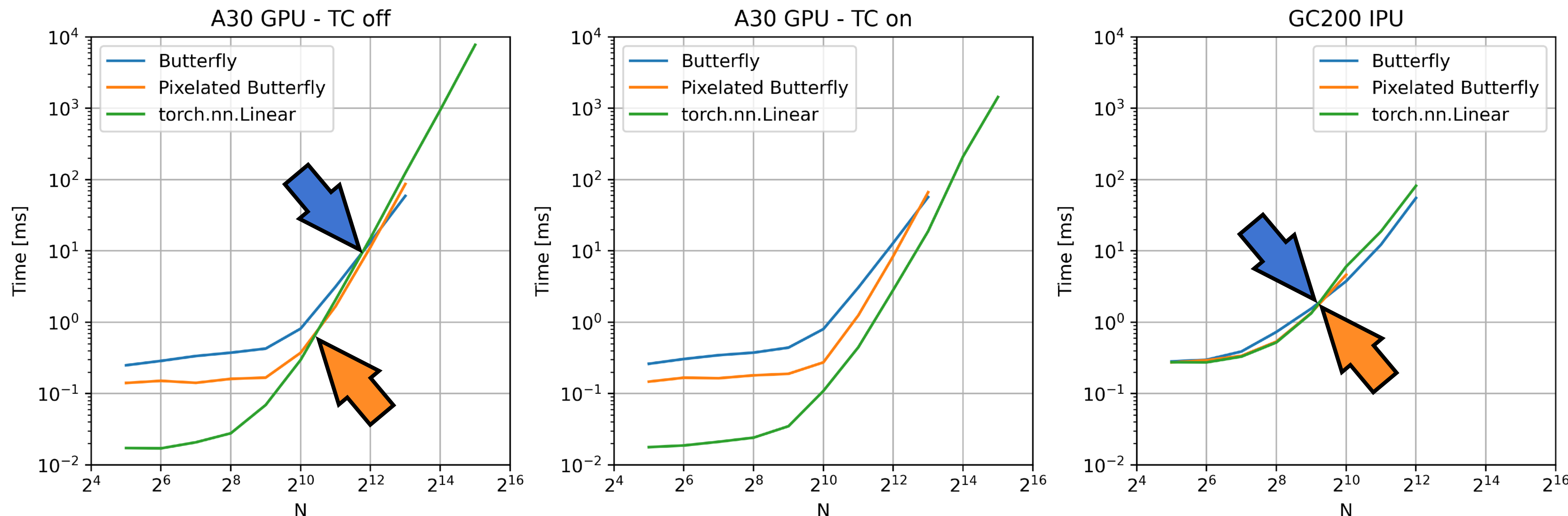


Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

- Evaluation of BF and PF, compared to `torch.nn.Linear` on both A30 and GC200





Intelligence Processing Units



Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Single-Hidden-Layer (SHL) benchmark on CIFAR-10 dataset with different structures matrix methods compared to baseline matrix approach on GPU and IPU

Method	N _{Params}	Accuracy [%]			Execution Time [s]		
		GPU		IPU	GPU		IPU
		w/ TC	w/o TC		w/TC	w/o TC	
<i>Baseline</i>	1059850	43.94	43.4	44.7	50.43	49.46	24.69
Butterfly	16390	42.27	40.75	41.13	61.93	61.46	37.73
Fastfood	14346	38.64	37.94	37.68	53.55	51.15	60.70
Circulant	12298	28.74	29.21	28.40	54.26	53.92	21.82
Low-rank	13322	18.64	18.49	18.59	49.71	53.21	21.75
Pixelfly	404490	42.61	43.31	43.79	52.79	56.01	71.62

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Single-Hidden-Layer (SHL) benchmark on CIFAR-10 dataset with different structures matrix methods compared to baseline matrix approach on GPU and IPU

Method	N _{Params}	Accuracy [%]			Execution Time [s]		
		GPU		IPU	GPU		IPU
		w/ TC	w/o TC		w/TC	w/o TC	
Baseline	1059850	43.94	43.4	44.7	50.43	49.46	24.69
Butterfly	16390	42.27	40.75	41.13	61.93	61.46	37.73
Fastfood	14346	38.64	37.94	37.68	53.55	51.15	60.70
Circulant	12298	28.74	29.21	28.40	54.26	53.92	21.82
Low-rank	13322	18.64	18.49	18.59	49.71	53.21	21.75
Pixelfly	404490	42.61	43.31	43.79	52.79	56.01	71.62

98.5%
compress

61.9%
compress

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations



S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Single-Hidden-Layer (SHL) benchmark on CIFAR-10 dataset with different structures matrix methods compared to baseline matrix approach on GPU and IPU

Method	N _{Params}	Accuracy [%]			Execution Time [s]		
		GPU		IPU	GPU		IPU
		w/ TC	w/o TC		w/TC	w/o TC	
Baseline	1059850	43.94	43.4	44.7	50.43	49.46	24.69
Butterfly	16390	42.27	40.75	41.13	61.93	61.46	37.73
Fastfood	14346	38.64	37.94	37.68	53.55	51.15	60.70
Circulant	12298	28.74	29.21	28.40	54.26	53.92	21.82
Low-rank	13322	18.64	18.49	18.59	49.71	53.21	21.75
Pixelfly	404490	42.61	43.31	43.79	52.79	56.01	71.62

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Single-Hidden-Layer (SHL) benchmark on CIFAR-10 dataset with different structures matrix methods compared to baseline matrix approach on GPU and IPU

Method	N _{Params}	Accuracy [%]			Execution Time [s]		
		GPU		IPU	GPU		IPU
		w/ TC	w/o TC		w/TC	w/o TC	
Baseline	1059850	43.94	43.4	44.7	50.43	49.46	24.69
Butterfly	16390	42.27	40.75	41.13	61.93	61.46	37.73
Fastfood	14346	38.64	37.94	37.68	53.55	51.15	60.70
Circulant	12298	28.74	29.21	28.40	54.26	53.92	21.82
Low-rank	13322	18.64	18.49	18.59	49.71	53.21	21.75
Pixelfly	404490	42.61	43.31	43.79	52.79	56.01	71.62

<4%
loss

1-7%
loss

3-8%
loss

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Single-Hidden-Layer (SHL) benchmark on CIFAR-10 dataset with different structures matrix methods compared to baseline matrix approach on GPU and IPU

Method	N _{Params}	Accuracy [%]			Execution Time [s]		
		GPU		IPU	GPU		IPU
		w/ TC	w/o TC		w/TC	w/o TC	
Baseline	1059850	43.94	43.4	44.7	50.43	49.46	24.69
Butterfly	16390	42.27	40.75	41.13	61.93	61.46	37.73
Fastfood	14346	38.64	37.94	37.68	53.55	51.15	60.70
Circulant	12298	28.74	29.21	28.40	54.26	53.92	21.82
Low-rank	13322	18.64	18.49	18.59	49.71	53.21	21.75
Pixelfly	404490	42.61	43.31	43.79	52.79	56.01	71.62

1.62x

1.17x

1.10x

0.53x

0.72x

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations



S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Method	N _{Params}	Accuracy [%]			Execution Time [s]		
		GPU		IPU	GPU		IPU
		w/ TC	w/o TC		w/TC	w/o TC	
Baseline	1059850	43.94	43.4	44.7	50.43	49.46	24.69
Butterfly	16390	42.27	40.75	41.13	61.93	61.46	37.73
Fastfood	14346	38.64	37.94	37.68	53.55	51.15	60.70
Circulant	12298	28.74	29.21	28.40	54.26	53.92	21.82
Low-rank	13322	18.64	18.49	18.59	49.71	53.21	21.75
Pixelfly	404490	42.61	43.31	43.79	52.79	56.01	71.62

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations



S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Single-Hidden-Layer (SHL) benchmark on CIFAR-10 dataset with different structures matrix methods compared to baseline matrix approach on GPU and IPU

Method	N _{Params}	Accuracy [%]			Execution Time [s]		
		GPU		IPU	GPU		IPU
		w/ TC	w/o TC		w/TC	w/o TC	
Baseline	1059850	43.94	43.4	44.7	50.43	49.46	24.69
Butterfly	16390	42.27	40.75	41.13	61.93	61.46	37.73
Fastfood	14346	38.64	37.94	37.68	53.55	51.15	60.70
Circulant	12298	28.74	29.21	28.40	54.26	53.92	21.82
Low-rank	13322	18.64	18.49	18.59	49.71	53.21	21.75
Pixelfly	404490	42.61	43.31	43.79	52.79	56.01	71.62

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Single-Hidden-Layer (SHL) benchmark on CIFAR-10 dataset with different structures matrix methods compared to baseline matrix approach on GPU and IPU

Method	N _{Params}	Accuracy [%]			Execution Time [s]		
		GPU		IPU	GPU		IPU
		w/ TC	w/o TC		w/TC	w/o TC	
Baseline	1059850	43.94	43.4	44.7	50.43	49.46	24.69
Butterfly	16390	42.27	40.75	41.13	61.93	61.46	37.73
Fastfood	14346	38.64	37.94	37.68	53.55	51.15	60.70
Circulant	12298	28.74	29.21	28.40	54.26	53.92	21.82
Low-rank	13322	18.64	18.49	18.59	49.71	53.21	21.75
Pixelfly	404490	42.61	43.31	43.79	52.79	56.01	71.62

⇒ butterfly variants: high compression ratio + acceptable accuracy

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Comparison of mean and standard deviation of metrics when varying parameters on the IPU

Butterfly size	Block size	Low-Rank size	Metric	mean	std
var.	2^3	2^1	Time[s]	372	107
	2^4		Accuracy[%]	43.8	2.2
	2^5		N _{Params}	1064970	326625
2^1	var.	2^2	Time[s]	465	192
		2^6	Accuracy[%]	38.9	1.4
		2^7	N _{Params}	81930	184638
2^2	2^4	var.	Time[s]	465	18
2^7	2^3		Accuracy[%]	37.8	2.7
2^4	2^4		N _{Params}	344074	181317

Recommendation:
Set the low-rank size to the maximum

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Comparison of mean and standard deviation of metrics when varying parameters on the IPU

Butterfly size	Block size	Low-Rank size	Metric	mean	std
var.	2^3	2^1	Time[s]	372	107
	2^4		Accuracy[%]	43.8	2.2
	2^5		NParams	1064970	326625
2^1	var.	2^2	Time[s]	465	192
		2^6	Accuracy[%]	38.9	1.4
		2^7	NParams	81930	184638
2^2	2^4		Time[s]	465	18
2^7	2^3	var.	Accuracy[%]	37.8	2.7
2^4	2^4		NParams	344074	181317

Recommendation:
Set the low-rank size to the maximum

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Comparison of mean and standard deviation of metrics when varying parameters on the IPU

Butterfly size	Block size	Low-Rank size	Metric	mean	std	
var.	2^3	2^1	Time[s]	372	107	
	2^4		Accuracy[%]	43.8	2.2	
	2^5		N _{Params}	1064970	326625	
2^1	2^2		Time[s]	465	192	
	var.	2^6	Accuracy[%]	38.9	1.4	
		2^7	N _{Params}	81930	184638	
2^2	2^4	2^4		Time[s]	465	18
2^7	2^3	var.	Accuracy[%]	37.8	2.7	
2^4	2^4		N _{Params}	344074	181317	

Recommendation:
Set the low-rank size to the maximum

Intelligence Processing Units

Reducing Memory Requirements for the IPU Using Butterfly Factorizations

S-Kazem Shekofteh, Christian Alles, and Holger Fröning. Reducing Memory Requirements for the IPU using Butterfly Factorizations. PMBS 2023 @ SC '23

Comparison of mean and standard deviation of metrics when varying parameters on the IPU

Butterfly size	Block size	Low-Rank size	Metric	mean	std
	2^3		Time[s]	372	107
	var.	2^1	Accuracy[%]	43.8	2.2
	2^5		NParams	1064970	326625
		2^2	Time[s]	465	192
2^1	var.	2^6	Accuracy[%]	38.9	1.4
		2^7	NParams	81930	184638
2^2	2^4		Time[s]	465	18
2^7	2^3	var.	Accuracy[%]	37.8	2.7
2^4	2^4		NParams	344074	181317

Recommendation:
Set the low-rank size to the maximum

Intelligence Processing Units

Distributed Butterfly Machine Learning with IPUs

Master Project work by Daniel Bogacz, May -August 2023

- An extension to our previous work to implement it on multiple IPUs.
 - Dataset: Fashion-MNIST
 - Model: Vision Transformer
 - 12 Encoder Layers
 - each with 12 Attention Heads
 - Hidden Layer sizes: 256
 - Intermediate sizes: 1024
 - trainable parameters:
 - 6,375,178 (torch.nn.Linear)
 - 180,198 (Butterfly)
 - Distributed across 4 IPUs:
 - IPU (1): Embedding Layer + 3 Encoder Layers
 - IPU (2) + (3): 3 Encoder Layers each
 - IPU (4): 3 Encoder Layers + Classification Layer
- **97.17%** compression rate

Intelligence Processing Units

Distributed Butterfly Machine Learning with IPUs

Master Project work by Daniel Bogacz, May -August 2023

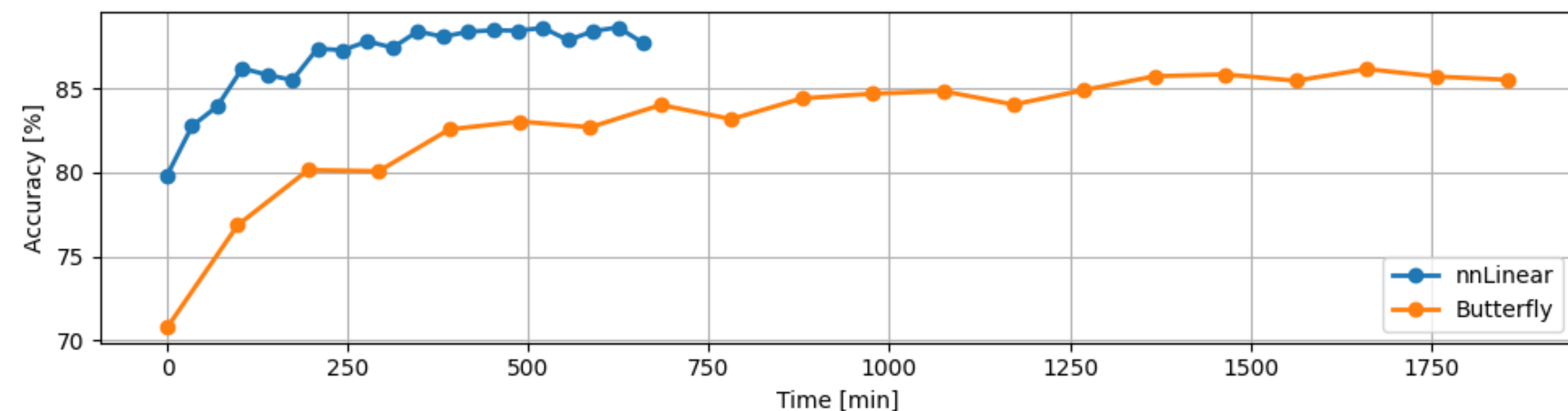
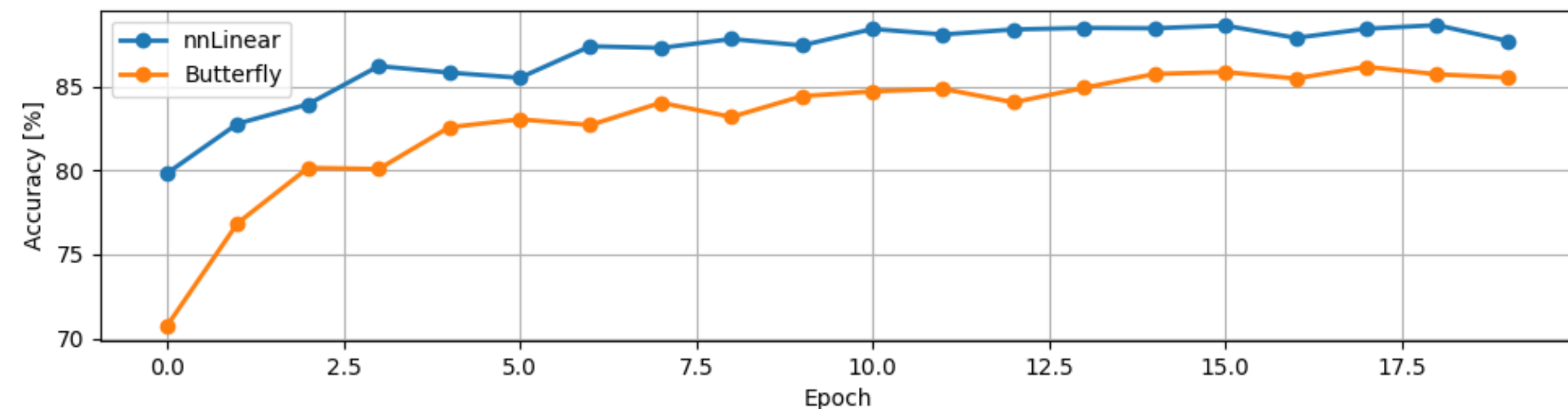
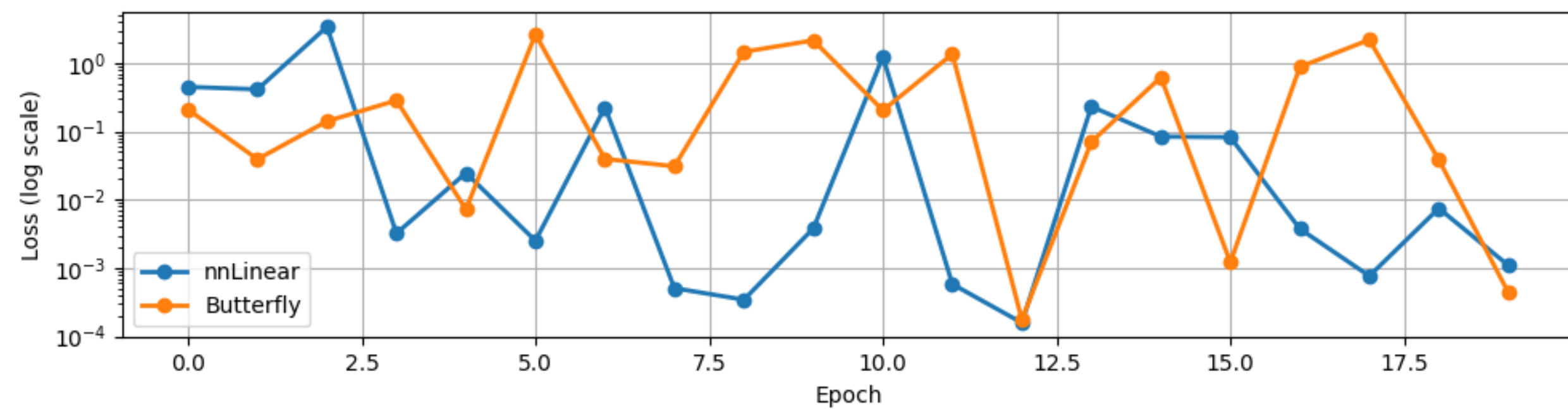
torch.nn.Linear vs Butterfly Layers on Multi-IPU

Butterfly:

- max accuracy: 86.10%
- time: 1953.27min

nn.Linear:

- max accuracy: 88.65%
- time: 694.03min



Loss and accuracy over training epochs and time with
torch.nn.Linear and butterfly layers on IPUs

Intelligence Processing Units

Distributed Butterfly Machine Learning with IPU

Master Project work by Daniel Bogacz, May -August 2023

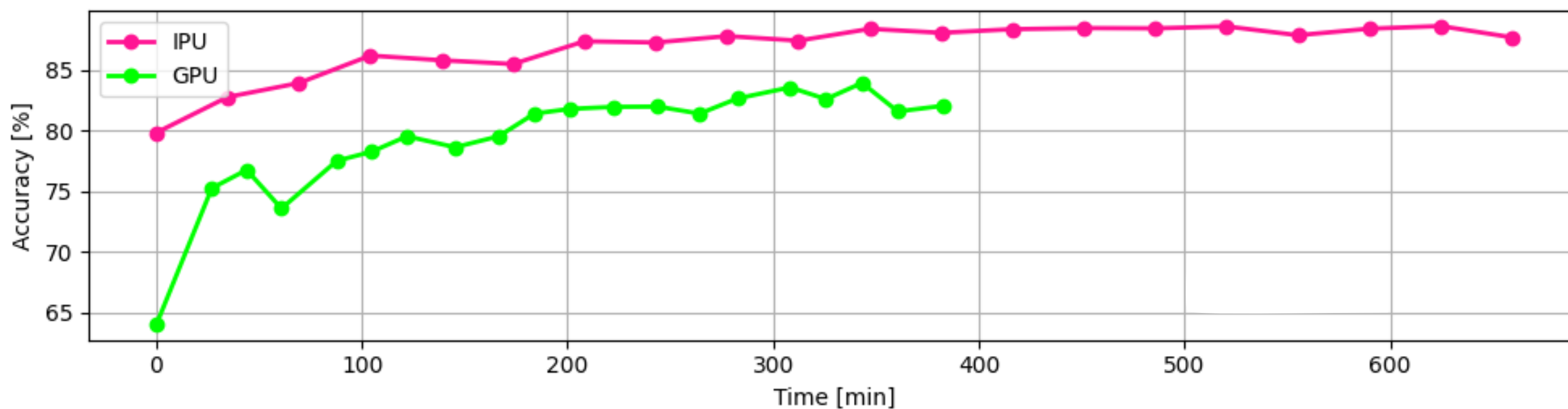
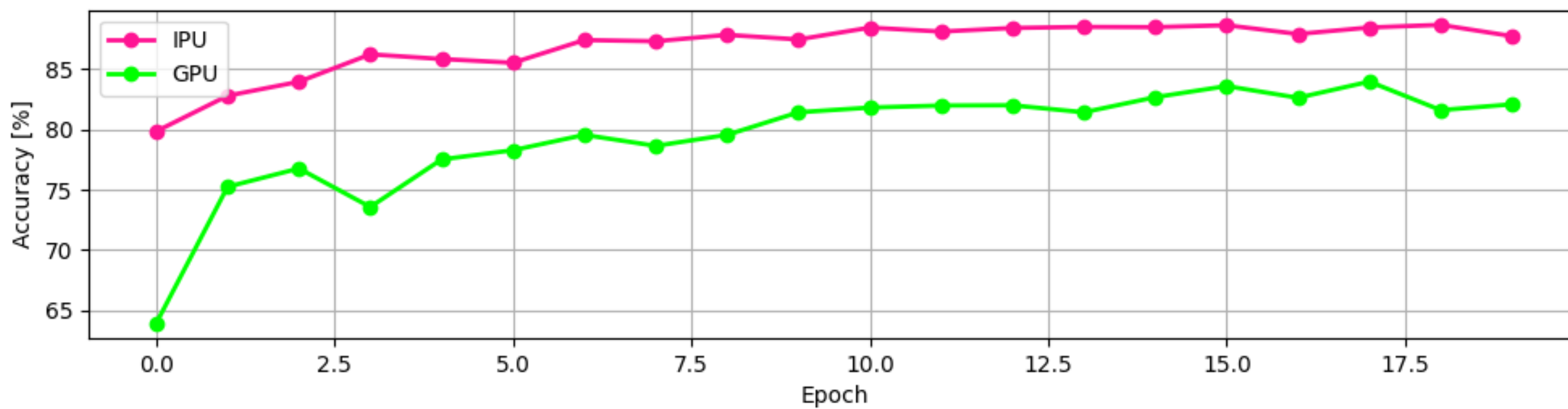
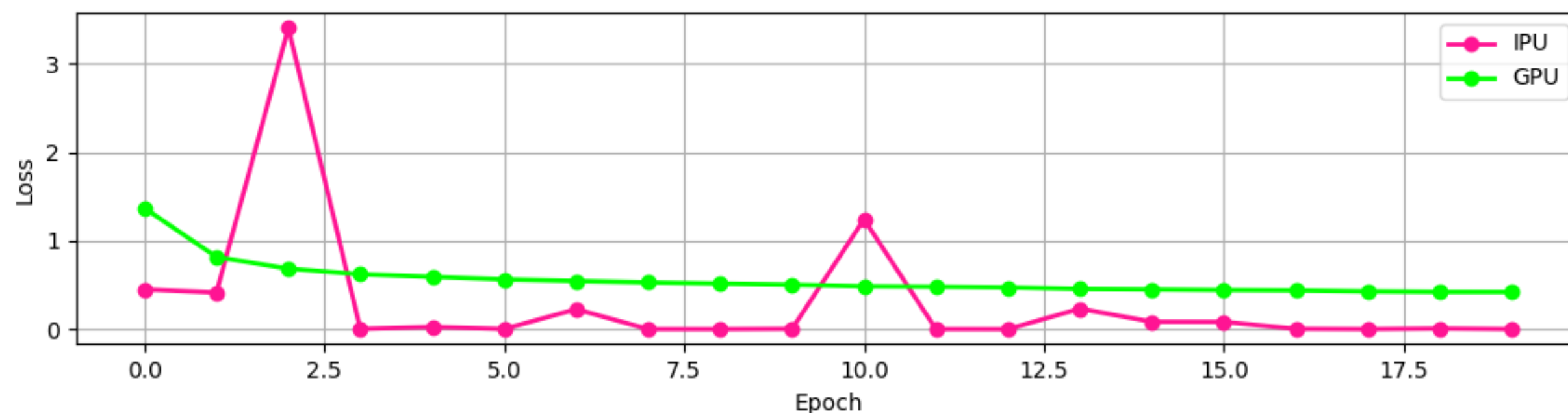
Multi-IPU vs Multi-GPU (4 K80 GPUs)

GPUs:

- max accuracy: 84.51%
- time: 401.44min

IPUs:

- max accuracy: 88.92%
- time: 694.03min



Loss and accuracy over training epochs and time with butterfly layers on GPUs and IPU

Intelligence Processing Units

Distributed Butterfly Machine Learning with IPUs

Master Project work by Daniel Bogacz, May-August 2024

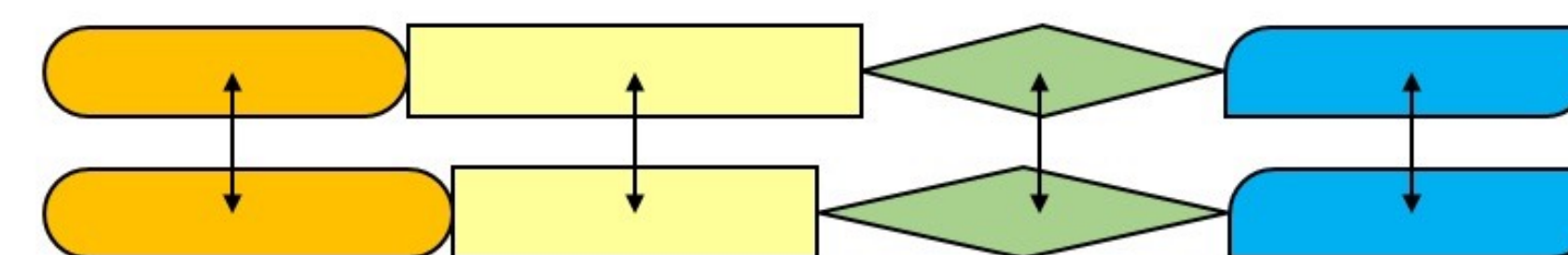
- `torch.nn.Linear` vs Butterfly Layers on Multi-IPU
 - Our previous results [Shekofteh et al. 2023] could be confirmed
 - High compression rate with Butterfly (97.17%)
 - Longer training time due to slow Butterfly implementation (Torch instead of Poplar)
 - Similar accuracy achieved (86.10% Butterfly and 88.65% `nn.Linear`)
- Multi-IPU vs Multi-GPU
 - More stable loss trajectory for GPUs than IPUs (different optimizer implementation)
 - 1.73x speedup in training time with GPUs (401.44min GPUs vs 694.03min IPUs)

IPU in Bioinformatics

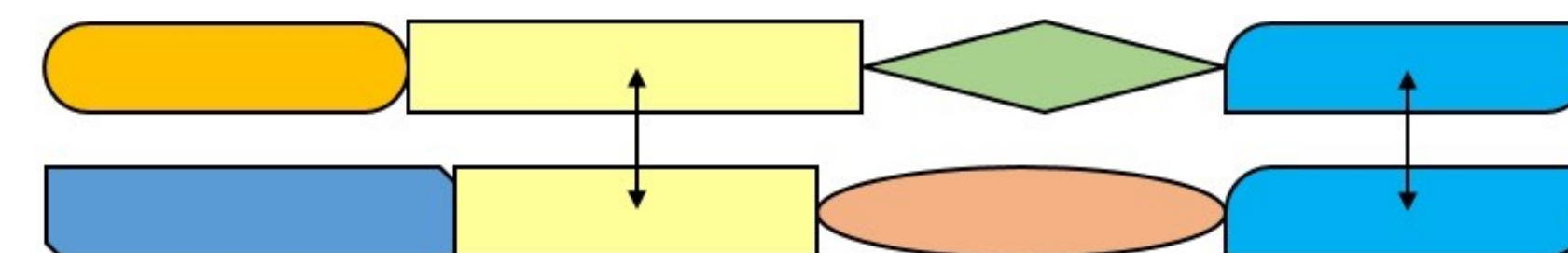
Introduction to Bioinformatics

A C T C G C A A T A T G C T A G G C C A G C
A C T _ _ _ _ T T A T G C T A T G C _ _ G C

- Sequence Alignment problem
- Well-known solutions are dynamic-programming-based:
 - Needleman-Wunsch (NW): global alignment
 - Smith-Waterman (SW): local alignment
 - Semi-global methods: X-Drop
- Finding the optimal solution for these algorithms
 - quadratic time as a function of sequence length.
- Main properties of the problem and common solutions:
 - Irregularity in parallelization of dynamic-programming methods
 - Sparsity in some cases



Global Alignment

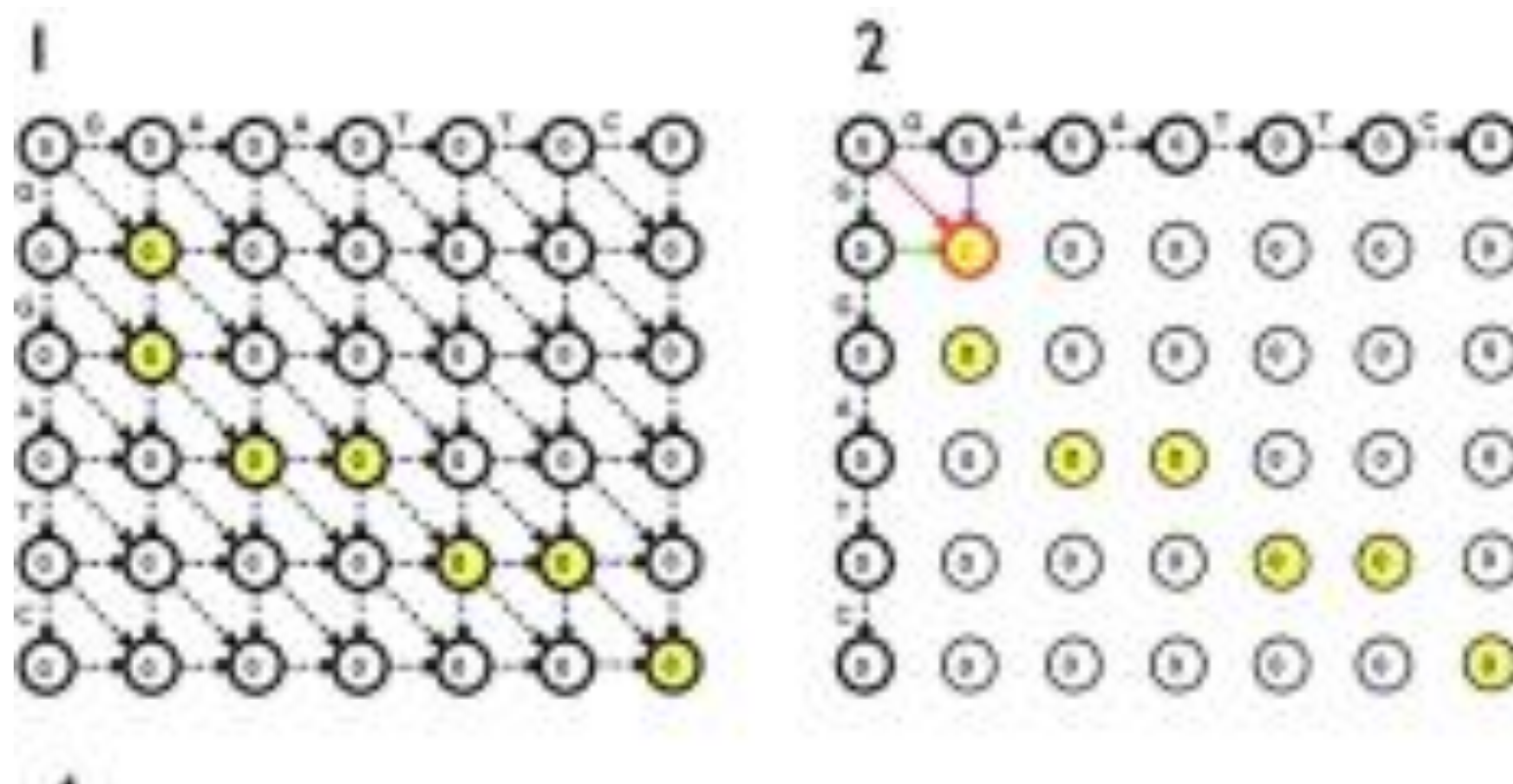


Local Alignment

IPU in Bioinformatics

Introduction to Bioinformatics

- Core computations in NW and SW



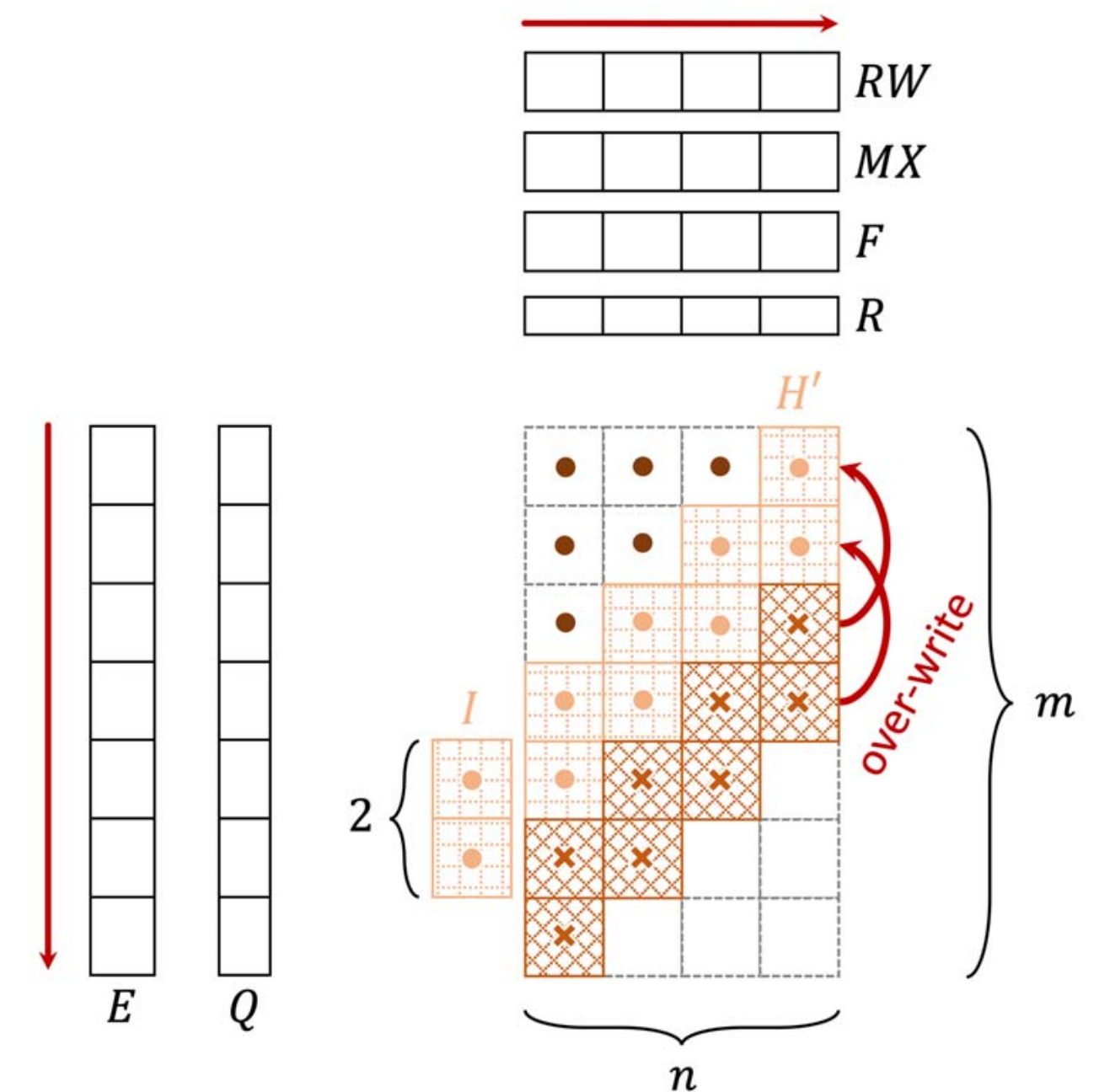
$$H[i, j] = 0, \forall i = 0 \text{ and } j = 0..n \quad (2)$$

$$H[i, j] = 0, \forall i = 0..m \text{ and } j = 0, \quad (3)$$

$$E[i] = \max \begin{cases} H[i, j-1] + GAP_{OPEN} \\ E[i] + GAP_{EXT} \\ 0 \end{cases} \quad \forall i = 1..m, \quad (4)$$

$$F[j] = \max \begin{cases} H[i-1, i] + GAP_{OPEN} \\ E[j] + GAP_{EXT} \\ 0 \end{cases} \quad \forall j = 1..n. \quad (5)$$

$$H[i, j] = \max \begin{cases} H[i-1, j-1] + S[Q[i-1], R[j-1]] \\ E[i] \\ F[j] \\ 0 \end{cases} \quad \forall i = 1..m \text{ and } j = 1..n, \quad (6)$$



IPU in Bioinformatics

Motivations: NW and SW

- X-Drop Method [1]
 - reduces the quadratic cost by dynamically searching only for a high-quality alignment and stopping the computation early when a good alignment is impossible
 - Good for long-read sequencing
 - Seems to be more scalable than NW and SW

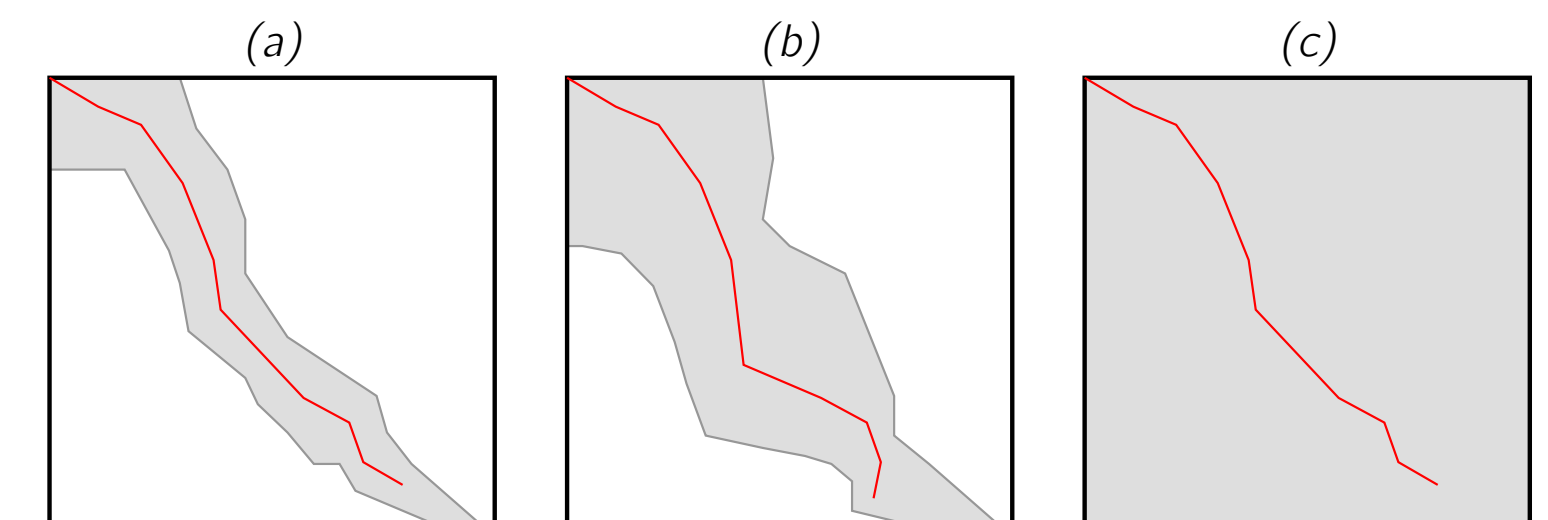
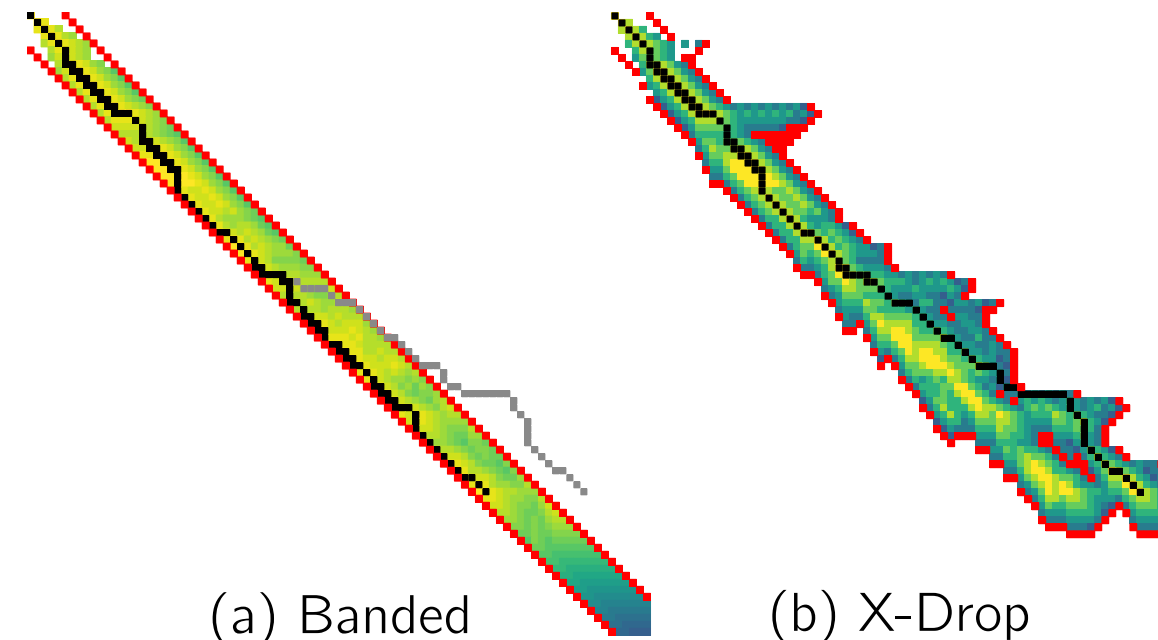
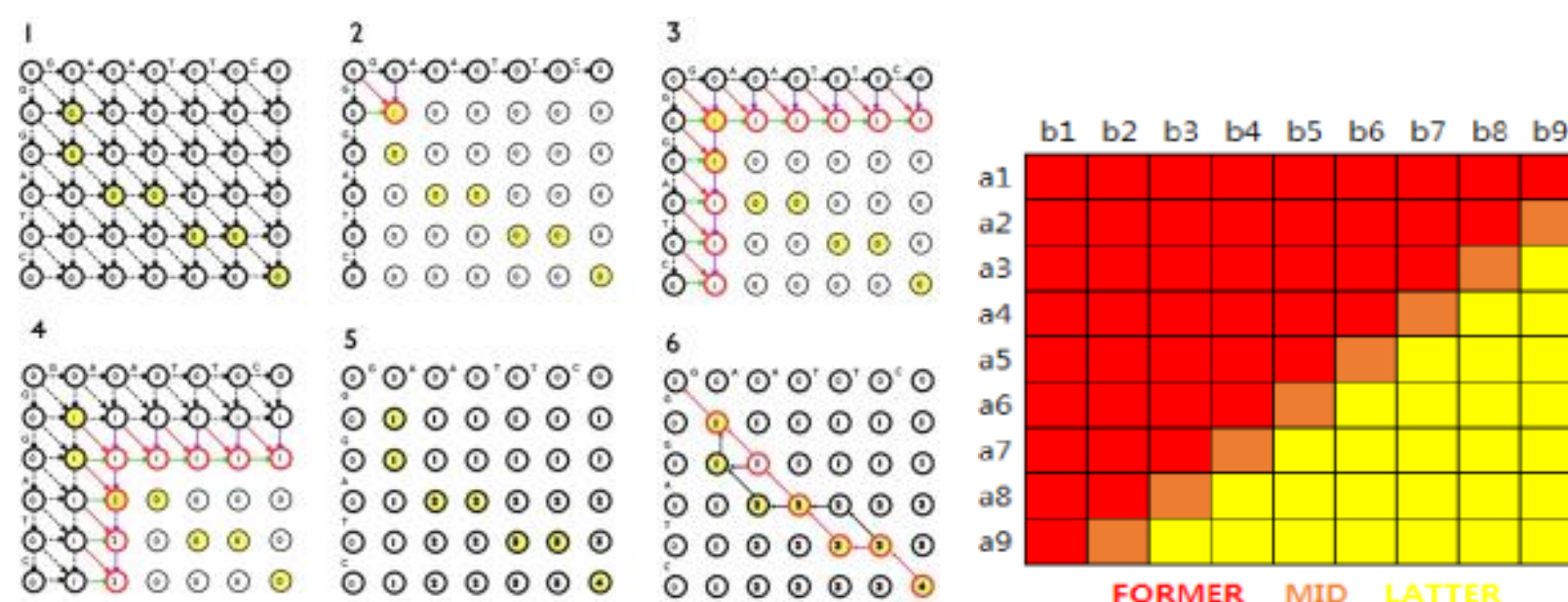


Figure 2: The red path is the optimal alignment, the gray area is calculated values, and the white area is non-calculated values. Due to the X-Drop condition, the white nonzeros contain a score of $-\infty$. Panel (a) shows an iteration with $X = 10$, (b) with $X = 20$, and (c) with $X = \infty$.

IPU in Bioinformatics

X-Drop Sequence Alignment Implementation on IPU

Master Project work by Jonathan Hirsch, August 2024 - ...

- Memory is the most limitation on the IPU
- The idea first introduced by Burchard et al, in [1] for
 - Limited SRAM-based computation
 - Multi-IPU machines
 - Clusters of IPU machines
 - Long input sequences

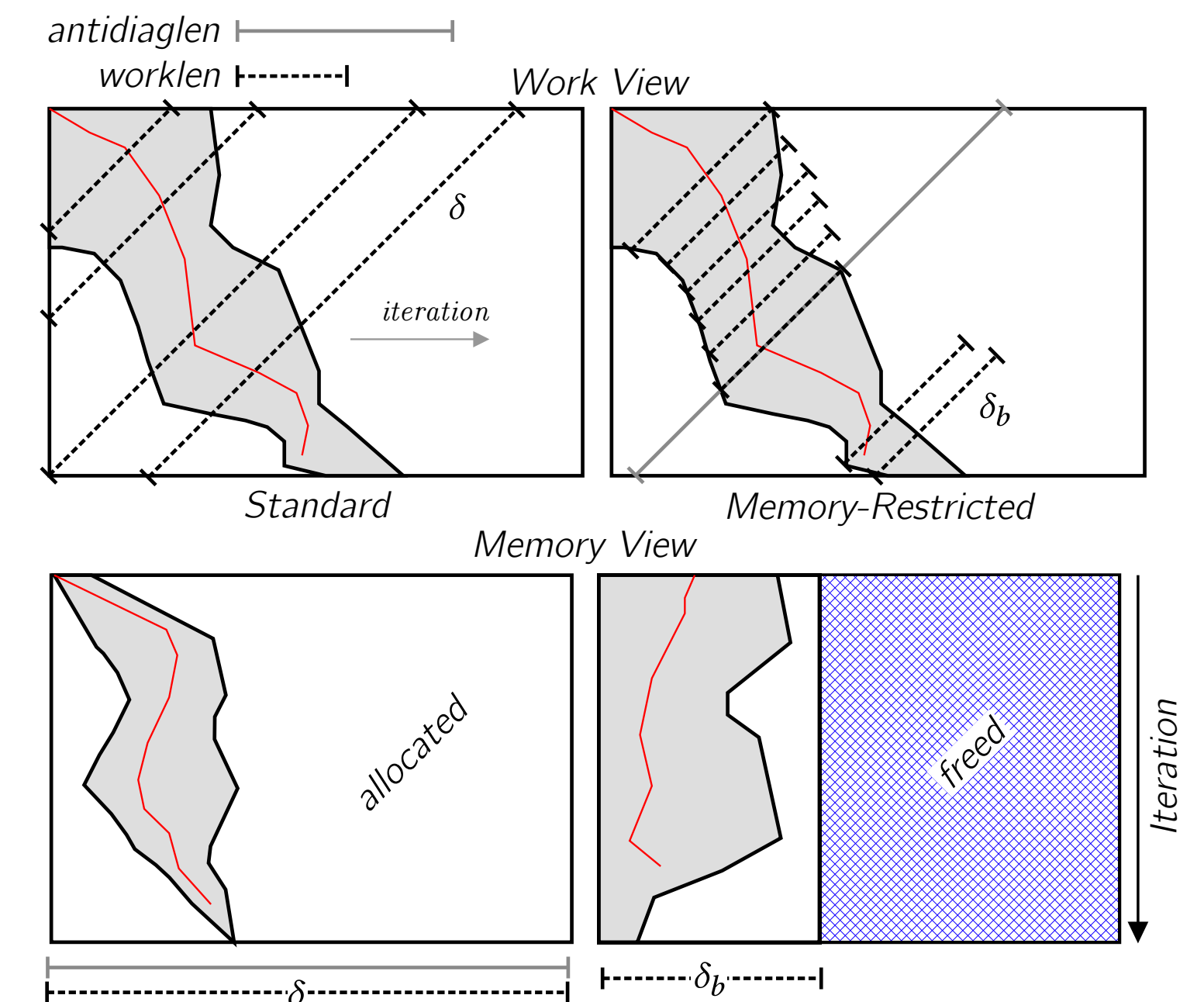


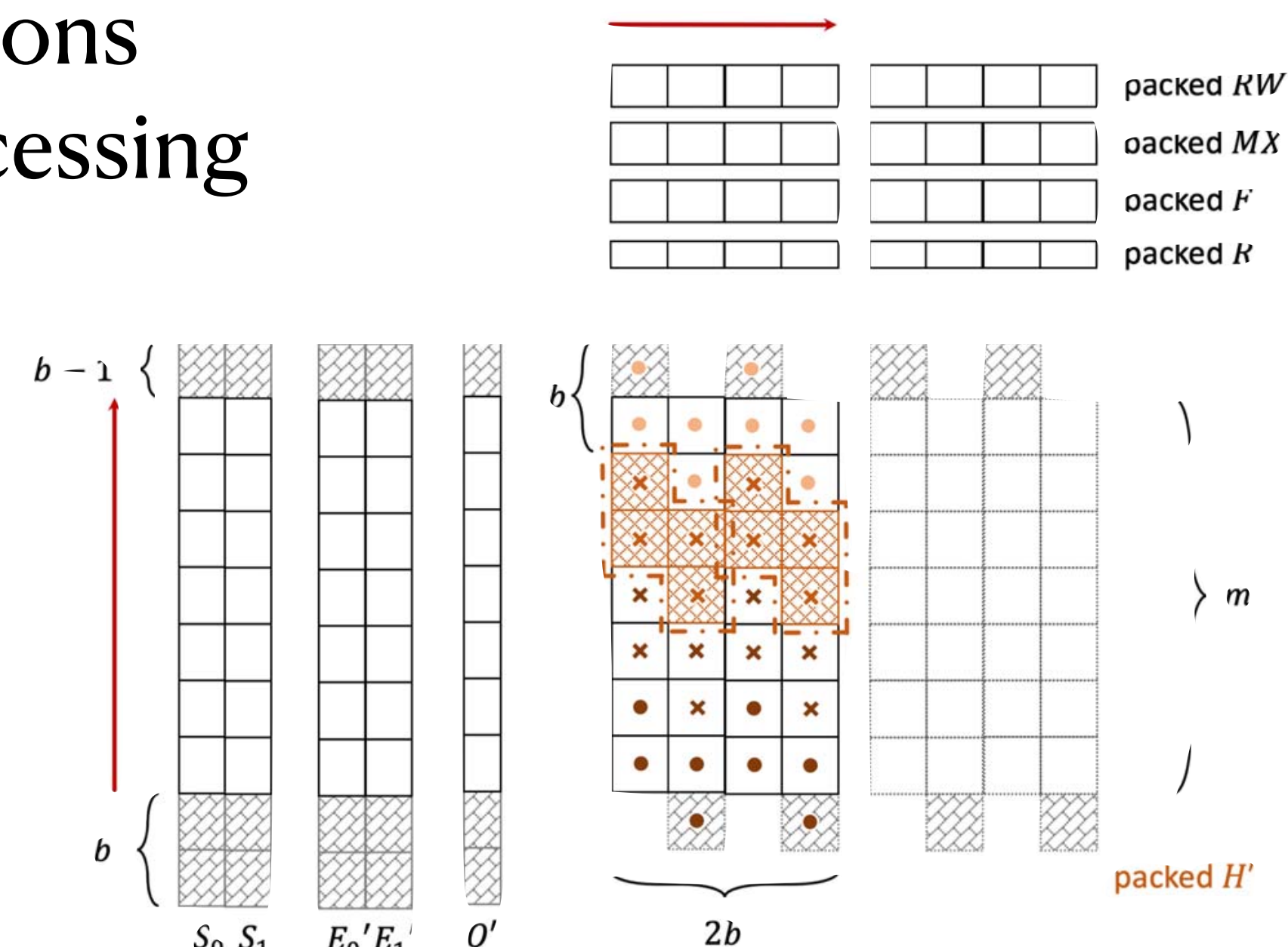
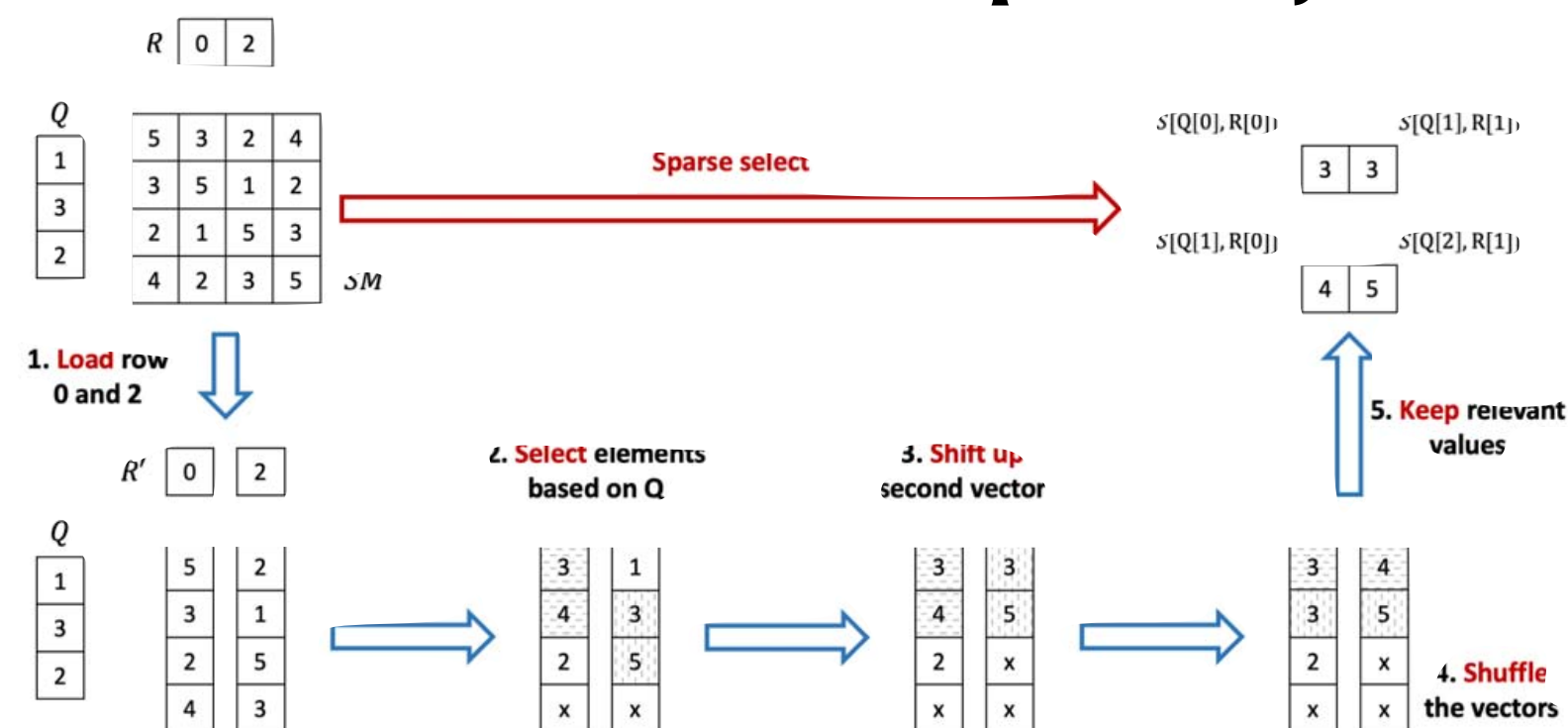
Figure 3: The antidiagonal length is $\delta = \min(|\mathcal{H}|, |\mathcal{V}|)$. The memory-restricted version allocates work memory of $\max_k |U_k - L_k| \leq \delta_b \leq \delta$. The left side illustrates the standard algorithm (3δ memory). The right side illustrates our algorithm ($2\delta_b$ memory).

IPU in Bioinformatics

MIMD Kernels for Sequence Alignment on IPU

Master Project work and thesis by Jeremias Kunz, August 2024 - ...

- The project is defined based on a recent work by Popovici [1]
- Target hardware: CPUs with SSE and AVX instructions to implement SIMD solutions for SIMD vector processing
- The aim is to extend the work on the IPU
- Make uses of the sparsity



[1] D. T. Popovici et al., "Designing Efficient SIMD Kernels for High Performance Sequence Alignment," 2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), St. Petersburg, FL, USA, 2023, pp. 167-176, doi: 10.1109/IPDPSW59300.2023.00038

Machine Learning Accelerators in Bioinformatics

Summary and Future Path



image was generated by ChatGPT image generator (DALL·E)