SPARCITY

An Optimization and Co-design Framework for Sparse Computation

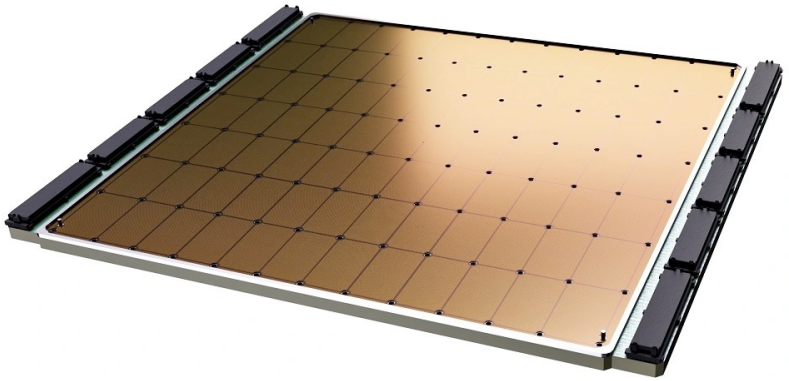
SIAM ACDA Aussois● September 2022

ML Accelerator Hardware: A Model for Parallel Sparse Computations ?

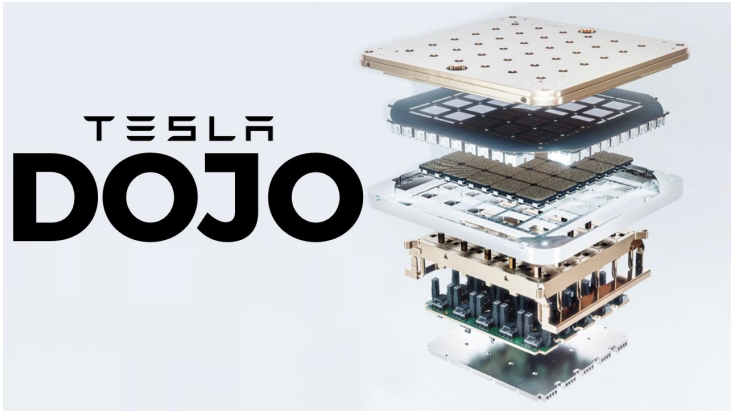
This project has received funding from the European High-Performance Computing Joint Undertaking under grant agreement No. 956213.

Current Trend in Computer Architecture: ML-specific Hardware

Graphcore GC200
Intelligence Processing Unit



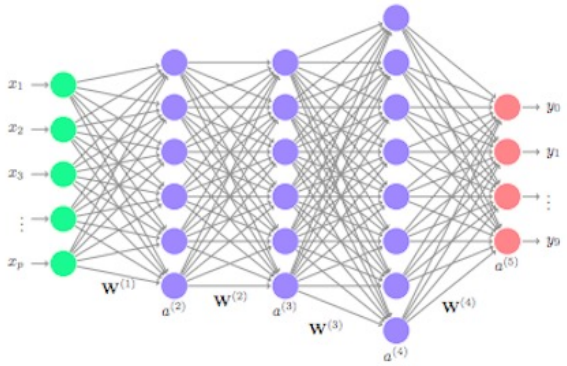
Cerebras WSE-2
Wafer Scale Engine

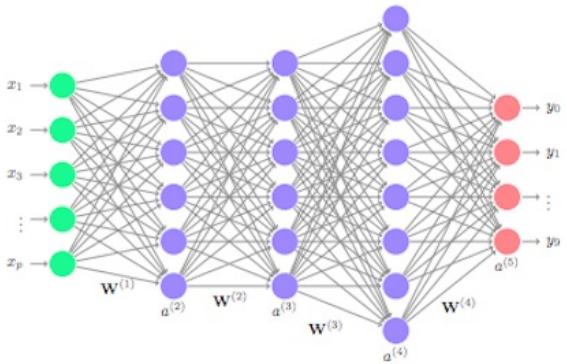


etc…

The golden age of computer architecture is here

Why Should we look at ML-specific Hardware ? (outside of the core ML/AI computations)

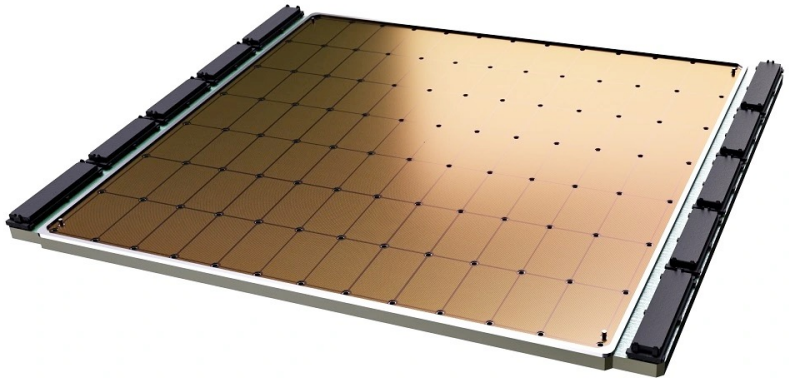




?

What are the Potential Benefits ?

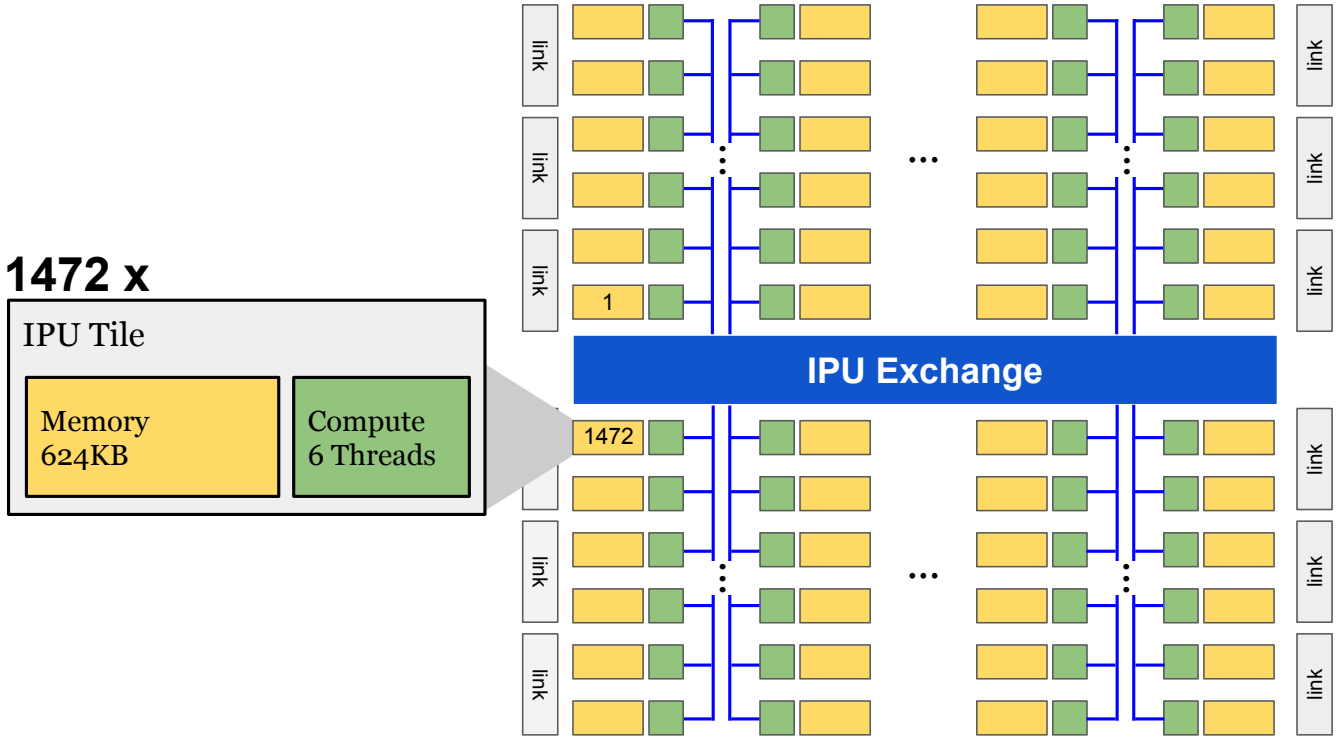
Graphcore GC200
Intelligence Processing Unit



Cerebras WSE-2
Wafer Scale Engine

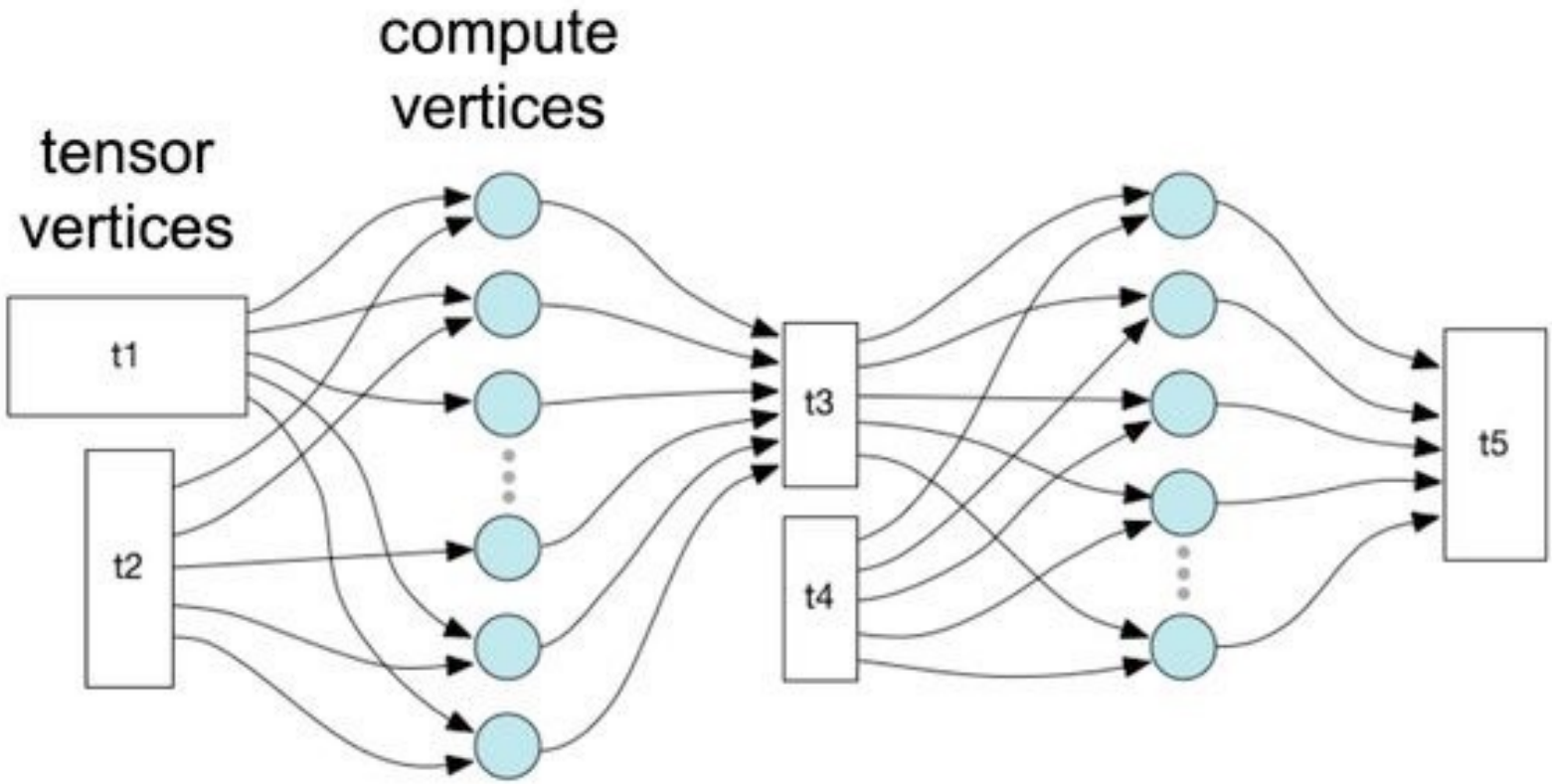
- **Graphcore IPU and Cerebras WSE are tile-based.**
- **A Tile is a core + SRAM functional unit**
- **Large SRAM is used as memory**
- **No memory hierarchy**
- **Very high bandwidth, low latency data access**
- **Larger problem ? More devices!**

The GC200 IPU: Distributed Memory on a Chip



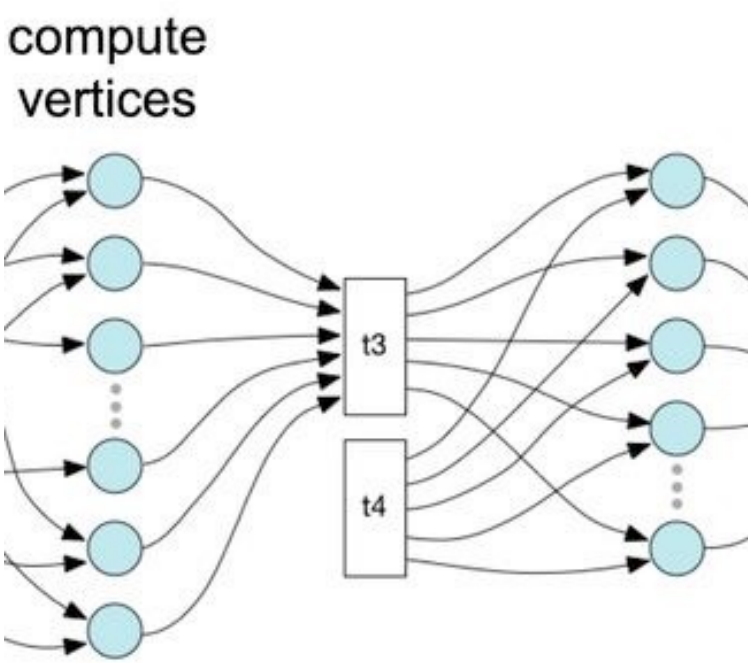
- 1472 cores per chip / 6 Threads per core
- Temporal multithreading (a.k.a. barrel processor)
- Scheduling order fixed, 6 cycles SRAM access
- Threads do not experience memory/cache latency
- True MIMD

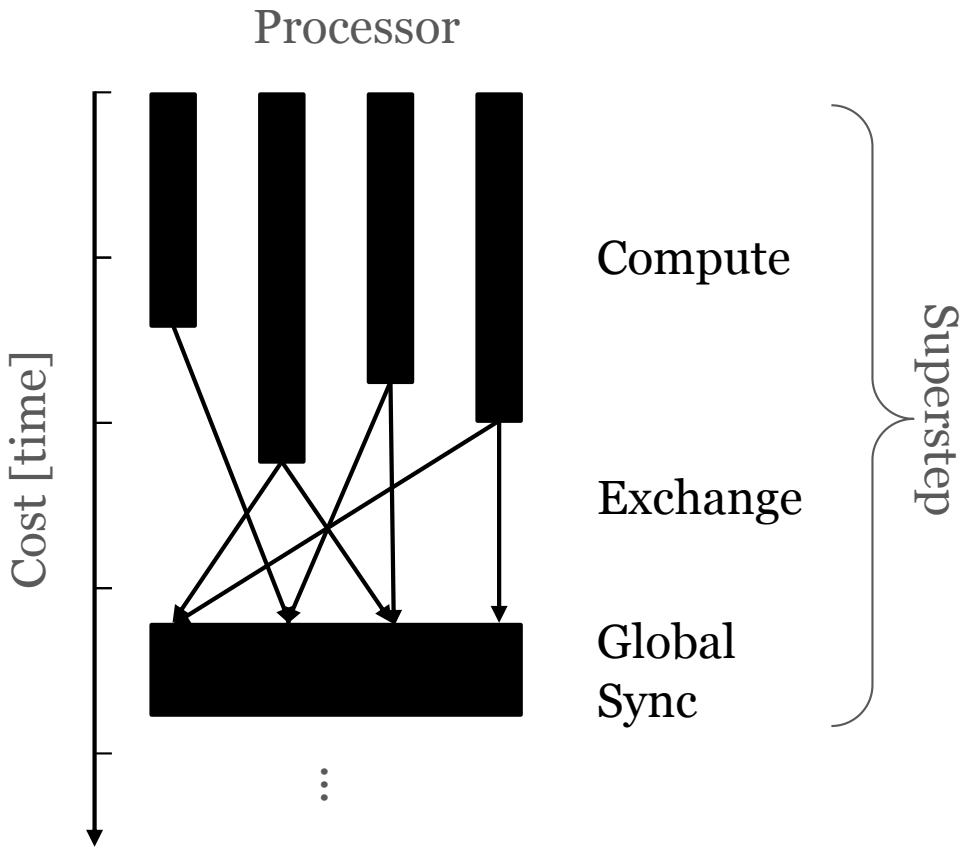
Dataflow-Based Programming Abstraction



- Data is arranged in immutable tensors
- Code is organized in codelets (compute vertices)
- Bipartite graph of data dependency
- Independent compute vertices are scheduled concurrently

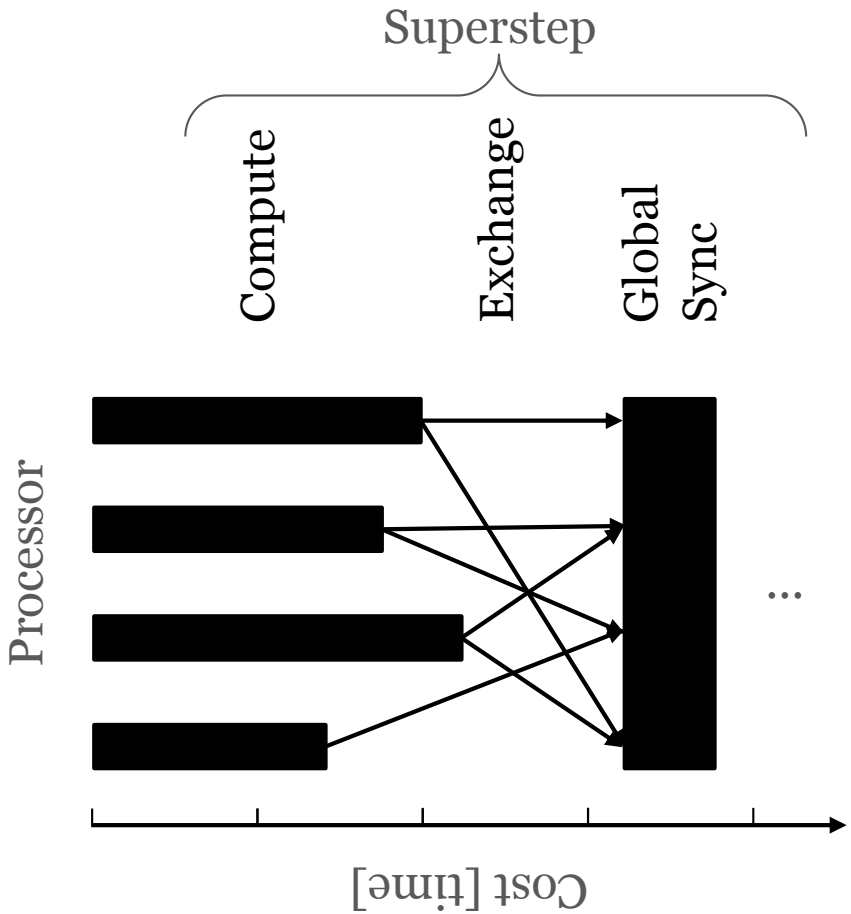
Bulk-Synchronous Parallel (BSP) Communication on the IPU

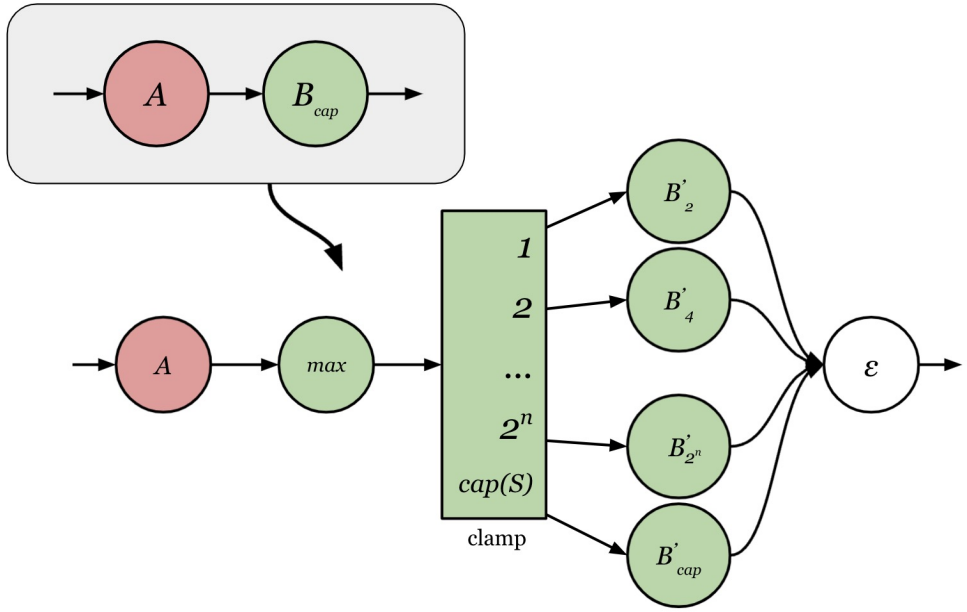




- Data exchange between concurrent phases
- Communication planned at compile time
- No communication/computation overlap
- No need for buffers
- High risk of load imbalance

Bulk-Synchronous Parallel (BSP) Communication on the IPU





- Poplar framework offers PGAS-style data exchange (programmer sees shared memory, system handles data exchange)
- Communication can be optimized by controlling data placement
- Communication codes exist at compile time, but we can chose between calling different codes at runtime

IPU Programing

```
bool compute() {
  auto size = endPos - startPos;
  for (int j = 0; j < size; j++) {
    float a = A[j * RNZ + 0] * V[I[j * RNZ + 0]];
    float b = A[j * RNZ + 1] * V[I[j * RNZ + 1]];
    float c = A[j * RNZ + 2] * V[I[j * RNZ + 2]];
    float d = A[j * RNZ + 3] * V[I[j * RNZ + 3]];

    a = a + A[j * RNZ + 4] * V[I[j * RNZ + 4]];
    b = b + A[j * RNZ + 5] * V[I[j * RNZ + 5]];
    c = c + A[j * RNZ + 6] * V[I[j * RNZ + 6]];
    d = d + A[j * RNZ + 7] * V[I[j * RNZ + 7]];

    a = a + A[j * RNZ + 8] * V[I[j * RNZ + 8]];
    b = b + A[j * RNZ + 9] * V[I[j * RNZ + 9]];
    c = c + A[j * RNZ + 10] * V[I[j * RNZ + 10]];
    d = d + A[j * RNZ + 11] * V[I[j * RNZ + 11]];

    a = a + A[j * RNZ + 12] * V[I[j * RNZ + 12]];
    b = b + A[j * RNZ + 13] * V[I[j * RNZ + 13]];
    c = c + A[j * RNZ + 14] * V[I[j * RNZ + 14]];
    d = d + A[j * RNZ + 15] * V[I[j * RNZ + 15]];
    newV[j] = D[j] * V[j] + a + b + c + d;
  }
  return true;
}
```

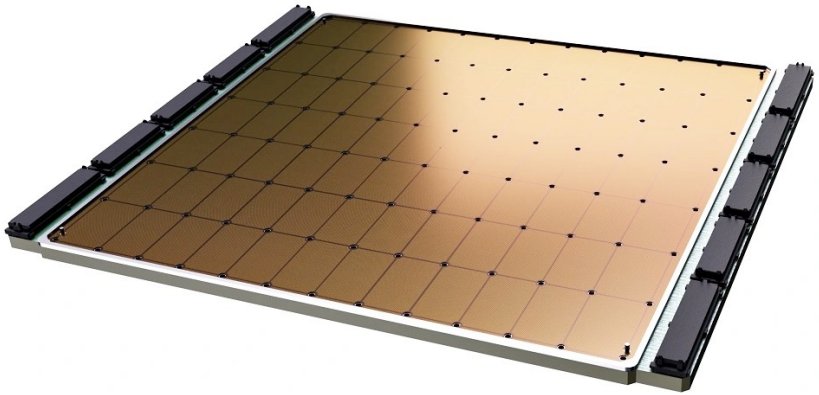
```
.LBB2_2:   # =>This Inner Loop Header:
Depth=1
   ld32       $a1, $m5, $m15, $m7
   {
      ld32    $a2, $m6, $m15, $m4
      f32mul $a1, $a2, $a1
   }
   {
      ld32    $a3, $m5, $m15, $m4
      f32add $a0, $a0, $a1
   }
   . . . # More of this stuff
   {
      add     $m7, $m4, 16
      f32mul $a1, $a2, $a3
   }
   f32add      $a0, $a0, $a1
   st32        $a0, $m0, $m15, $m4
   sort4x16lo $m4, $m7, $m15
   cmpslt      $m7, $m4, $m1
   brnz        $m7, .LBB2_2
```

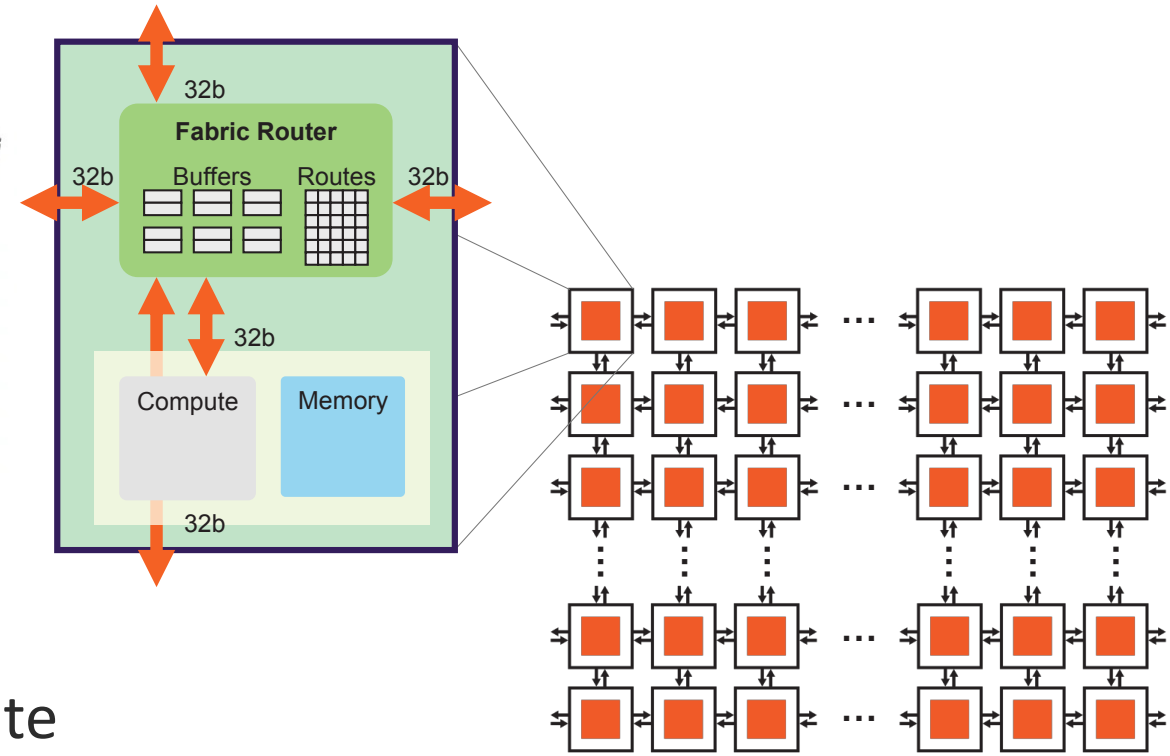
Computation optimization is straightforward:

minimize number of instructions

No performance modeling required here

Cerebras WSE-2 Wafer Scale Engine

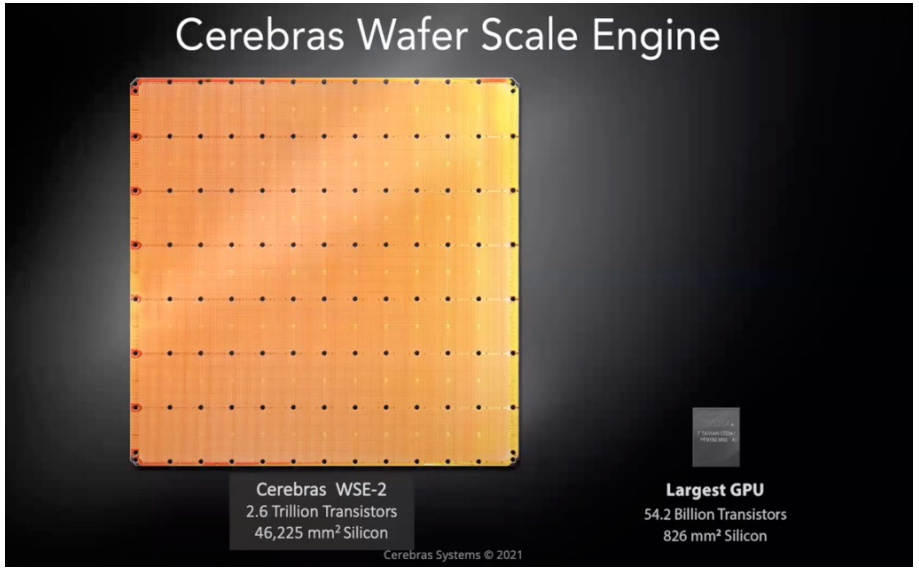




- 50% SRAM, 50% compute
- 2D grid-based communication
- Very fast neighbor to neighbor communication
- Fully user controlled
- Dynamic communication, fully overlapped
- High programing complexity, high potential reward

Comparison of radically different Devices is Challenging

Some confusion on what should be compared…



	Cerebras WSE	Largest GPU	Cerebras Advantage
Chip size	46,225 mm²	815 mm²	56.7 X
Cores	400,000	5,120	78 X
On chip memory	18 Gigabytes	6 Megabytes	3,000 X
Memory bandwidth	9 Petabytes/S	900 Gigabytes/S	10,000 X
Fabric bandwidth	100 Petabits/S	300 Gigabits/S	33,000 X

Important to clarify what **memory** means

Better distinguish SRAM and DRAM

Comparison of radically different Devices is Challenging

	#	Metric	Google TPU v3	Nvidia V100	Nvidia A100	Cerebras WSE	GraphCore IPU1	GraphCore IPU2
Raw Metrics	1	Technology node	>12nm (16 nm est.)	TSMC 12 nm	TSMC 7 nm	TSMC 16 nm	TSMC 16 nm	TSMC 7 nm
	2	Die Area (mm2)	<648 (600 est.)	815	826	46225	900 (est.)	823
	3	Transistor Count (B)	11 (est.)	21	54.2	1200	23.6	59.4
	4	Architecture	Systolic Array	SIMD + TC	SIMD + TC	MIMD	MIMD	MIMD
	5	Theoretical TFLOPS (16-bit mixed precision)	123	125	312	2500	125	250
	6	Freq (GHZ)	0.92	1.5	1.4	Unknown	1.6	Unknown
	7	DRAM Capacity (GB)	32	32	80	N/A	N/A	112
	8	DRAM BW (GB/sec)	900	900	2039	N/A	N/A	64 (est.)
	9	Total SRAM Capacity	32MB	36 MB (RF+L1+L2)	87 MB (RF+L1+L2)	18 GB	300 MB	900 MB
	10	SRAM BW (TB/sec)	Unknown	224 @RF + 14 @L1 + 3 @L2	608 @RF+ 19 @L1 + 7 @L2	9000	45	47.5
	11	Max TDP (Watts)	450	450	400	20K	150	150 (est.)
Efficiency M	12	GEMM Achievable TFLOPS	98%	88%	93%	Unknown	47%	61%
	13	Energy Efficiency (Achievable GEMM TFLOPS/Max Watts)	0.26	0.24	0.72	Unknown	0.39	1.0
	14	Theoretical Energy Efficiency (Theoretical TFLOPS/Max Watts)	0.27	0.27	0.78	0.125	0.83	1.6
	15	Memory Capacity (GB)	16	32	80	18	0.3	112
	16	Memory Efficiency (FLOP/DRAMByte)	133	122	158	N/A	N/A	Unknown
	17	Memory Efficiency	Unknown	32	35	Unknown	1.28	3.2
	18	Area Efficiency (Achievable TFLOPS/mm2)	0.2	0.13	0.35	Unknown	0.06	0.17
	19	Area Efficiency (Achievable TFLOPS/BTran)	11	5.2	5.3	Unknown	2.5	2.6

1 IPU ≈ 1 GPU (Silicon)
2 IPU ≈ 1 GPU (Power)

GPU has more FLOPS via SIMD

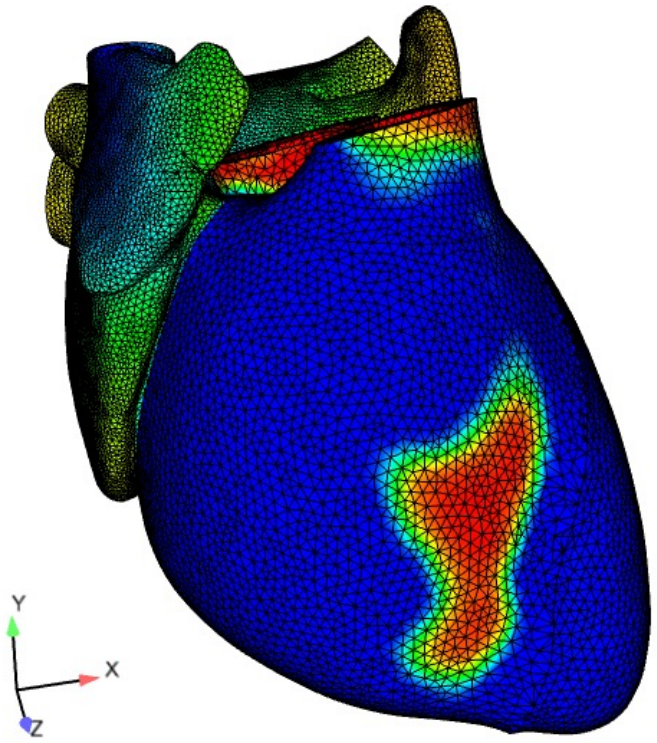
1 WSE ≈ 64 IPU

64 IPU = 94K tiles
1 WSE = 850K tiles

WSE has less memory, more FLOPS

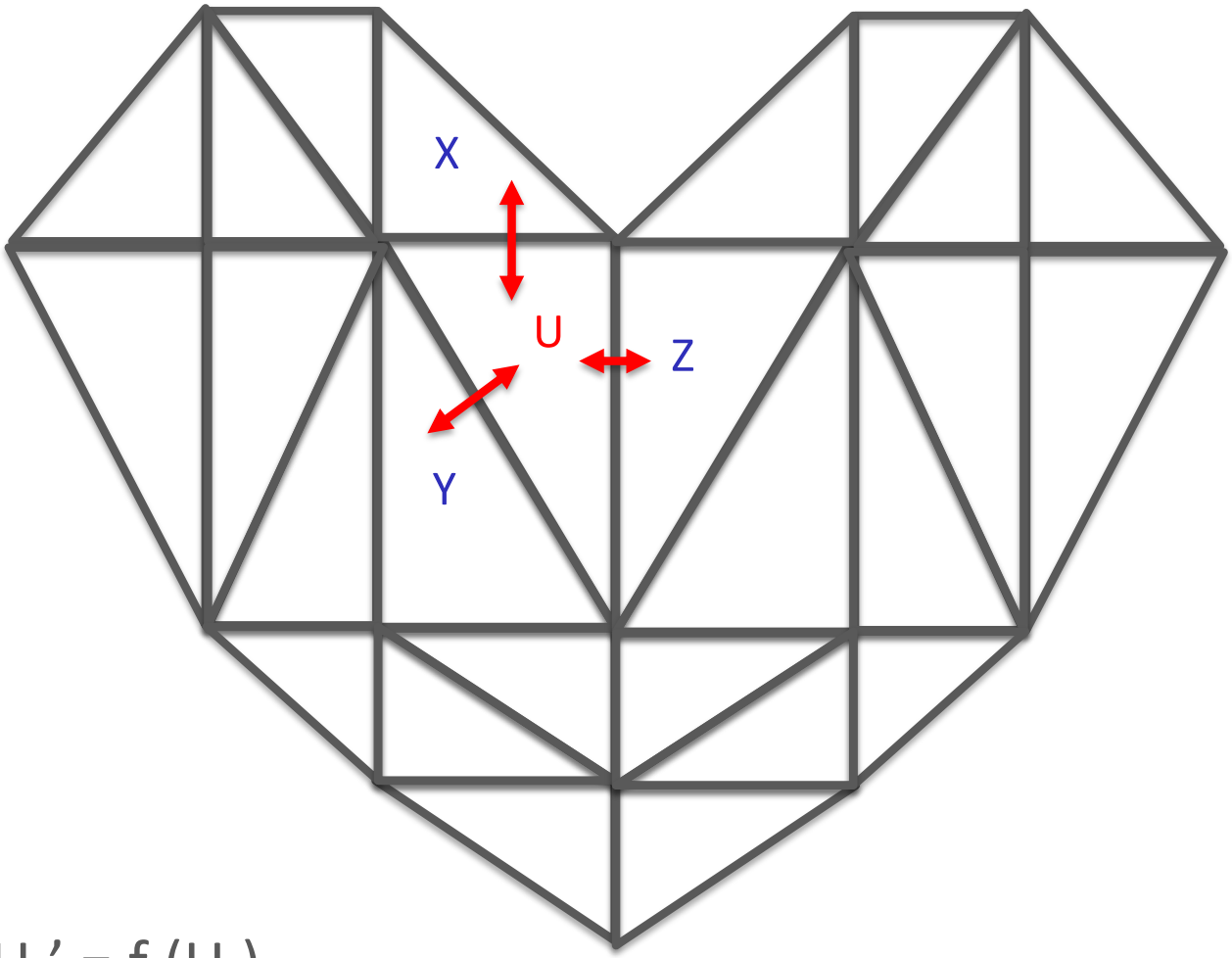
Tim Rogers and Mahmoud Khairy, An Academic's Attempt to Clear the Fog of the Machine Learning Accelerator War, ACM SIGARCH blog

LYNX: Cardiac Electrophysiology Monodomain Simulation **on the IPU**



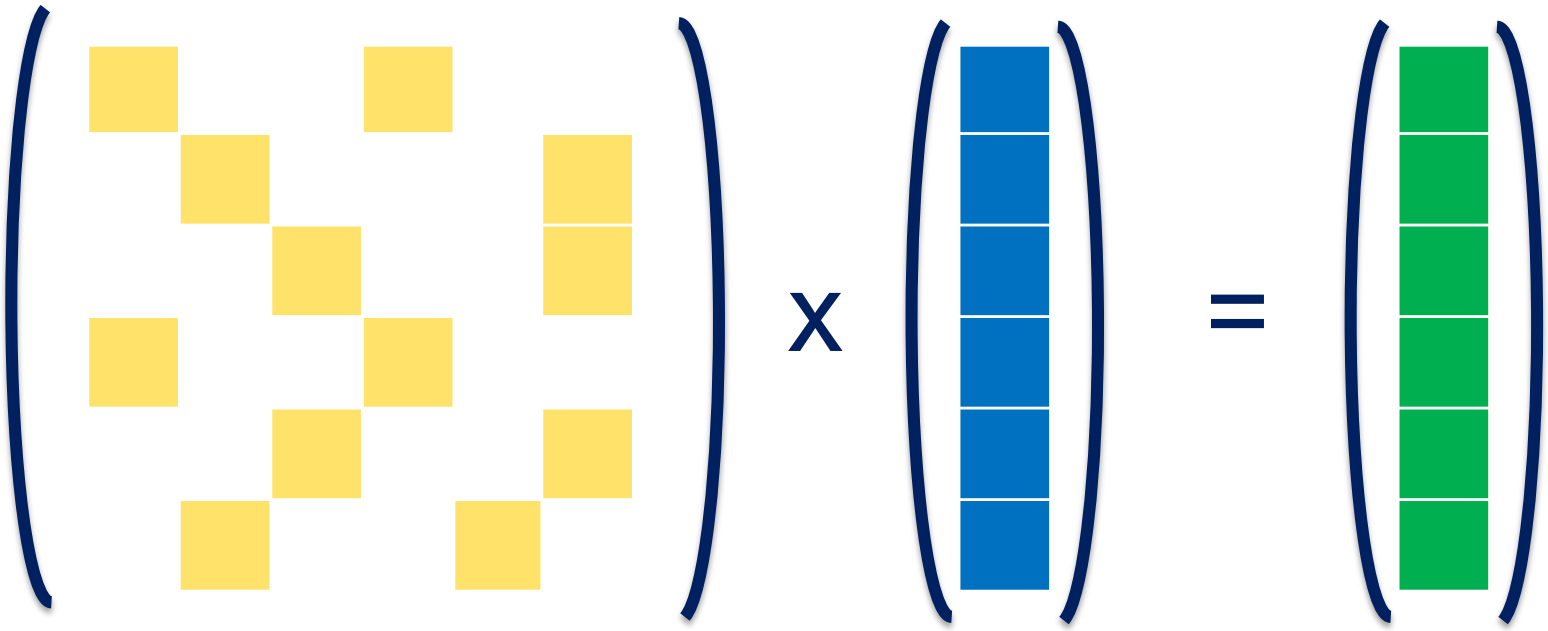
- We want to study electrical activity in the heart
- LYNX code: finite volume monodomain simulation
- Dissolve heart into mesh cells
- Simulate diffusion of voltage over time
- Discretize time and space

LYNX: Cardiac Electrophysiology Monodomain Simulation



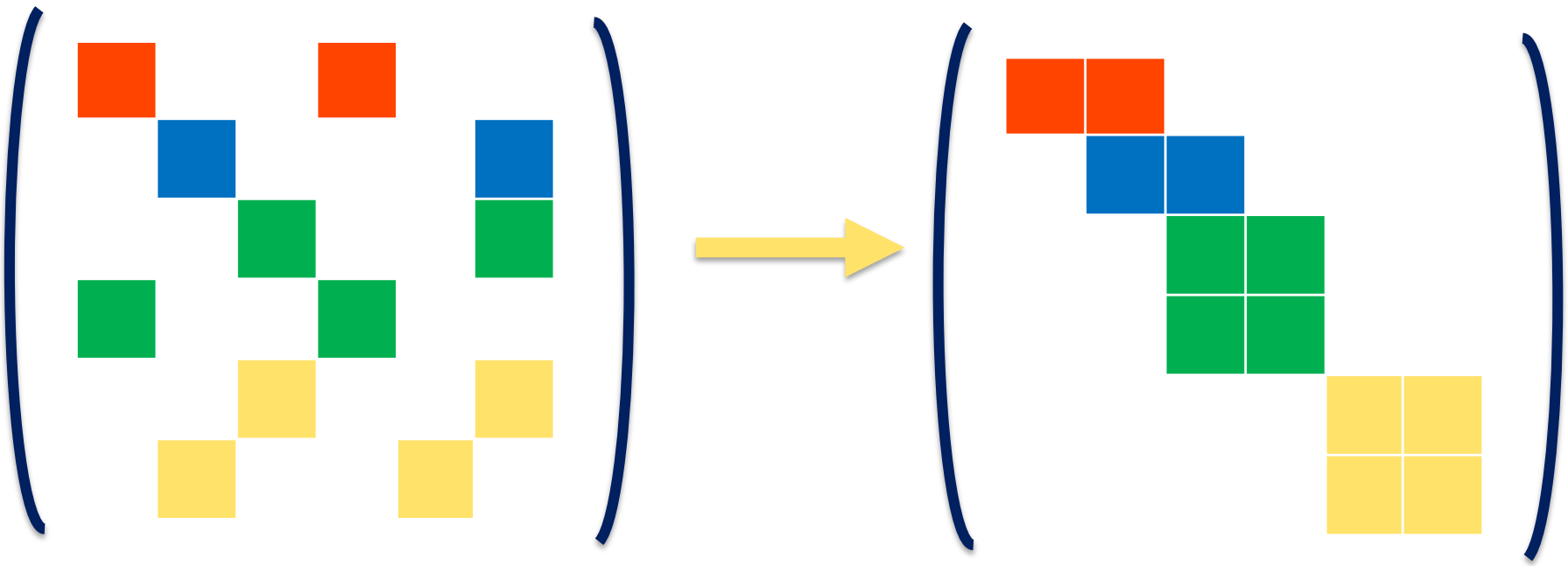
- ODE: $U_t' = f(U_t)$
- PDE: $U_{t+1} = g(U_t', X_t', Y_t', Z_t')$

PDE step: Repeated Sparse Matrix Dense Vector



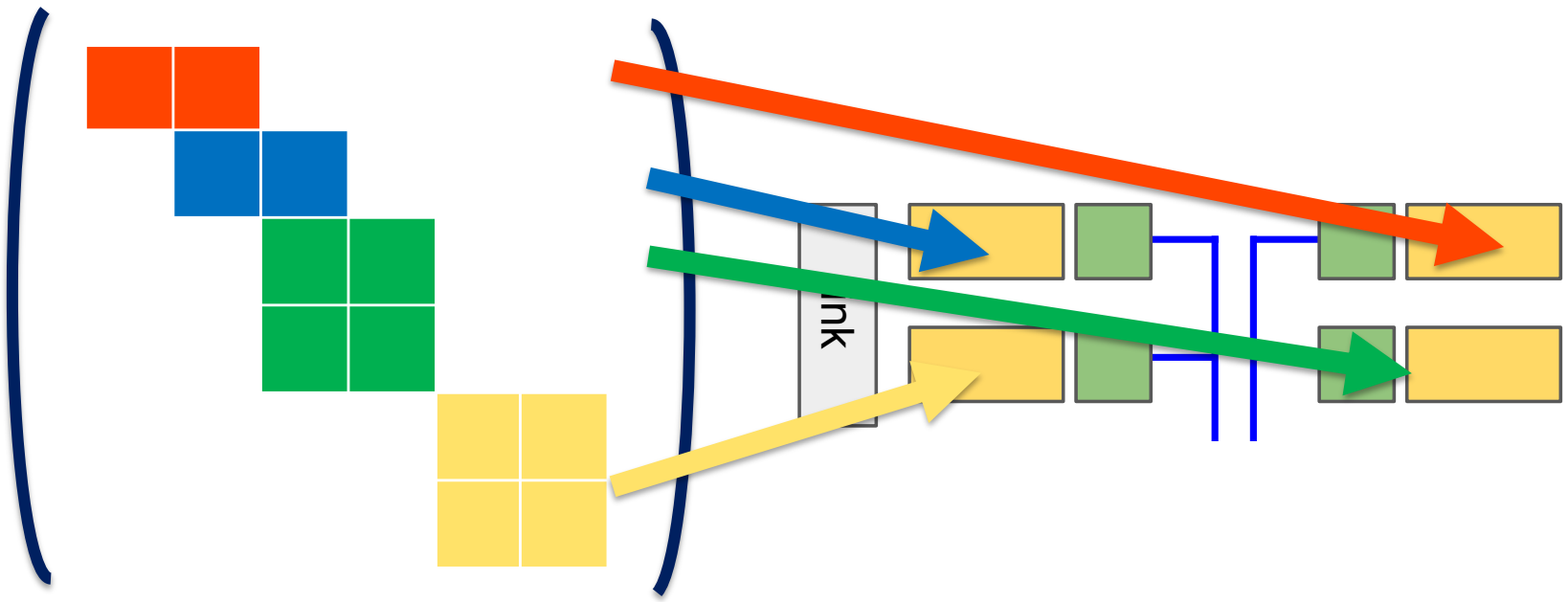
- Data access pattern irregular but static
- Sparsity pattern defines a graph
- Typically memory bandwidth bound on CPUs and GPUs

LYNX: 1D Partitioning



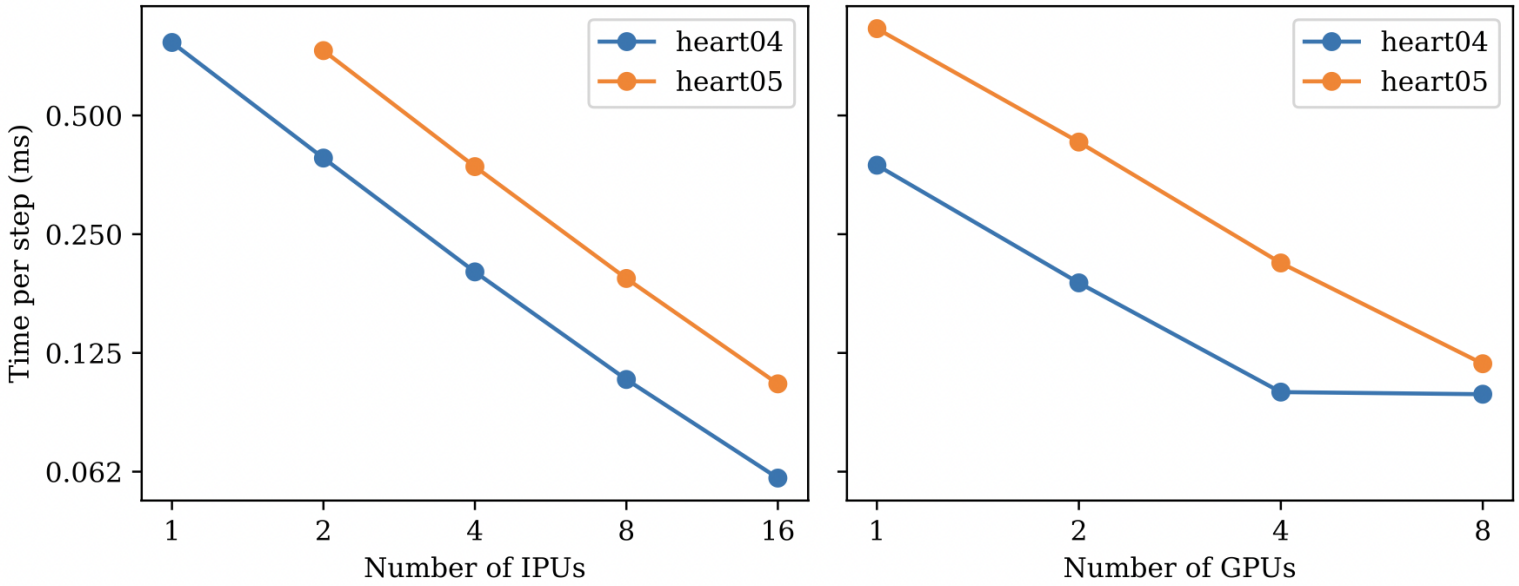
- Use Metis, PaToH, KaHIP, etc…
- Partition and reorder to minimize communication
- Amortize cost over a large number of repetitions

LYNX: 1D Partitioning

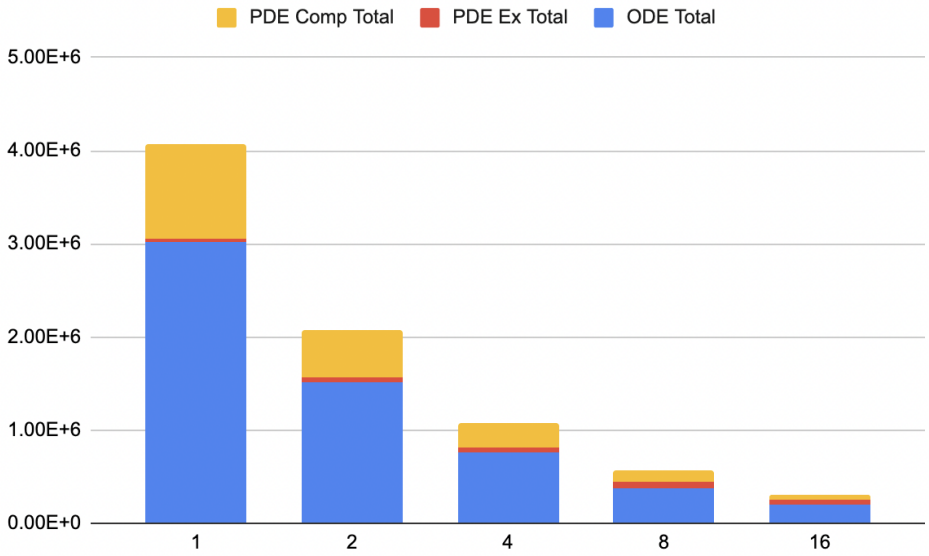


- 1D assignment of matrix rows to IPU tiles

Performance Comparison IPU vs GPU

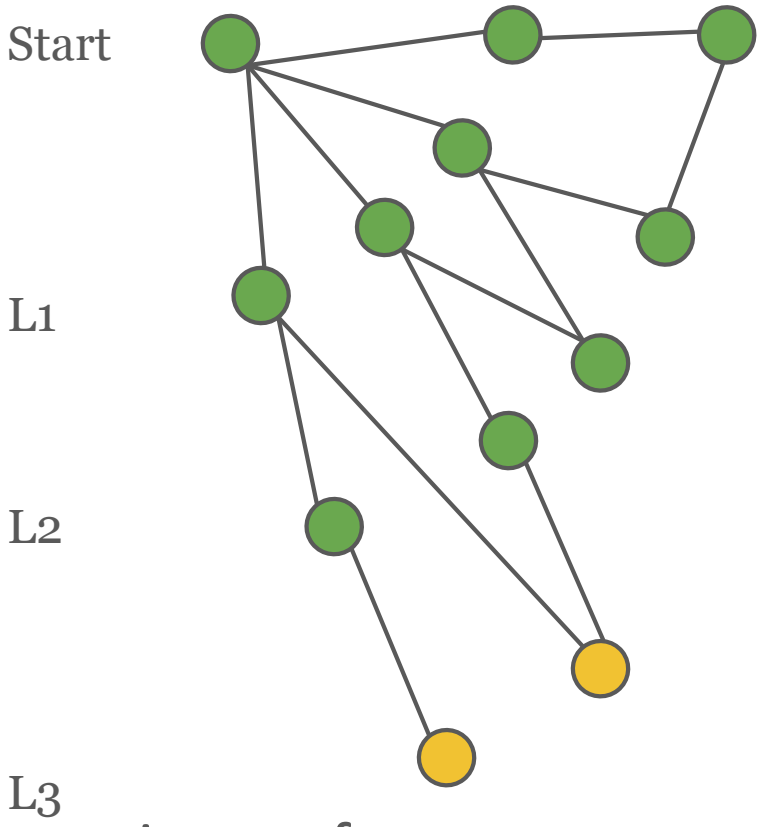


- Devices are comparable at 2 IPU : 1 GPU
- PDE (SpMV) is much faster on IPU.
- ODE faster on the GPU (more FLOPS)
- FP32 is a severe limitation for scientific computing



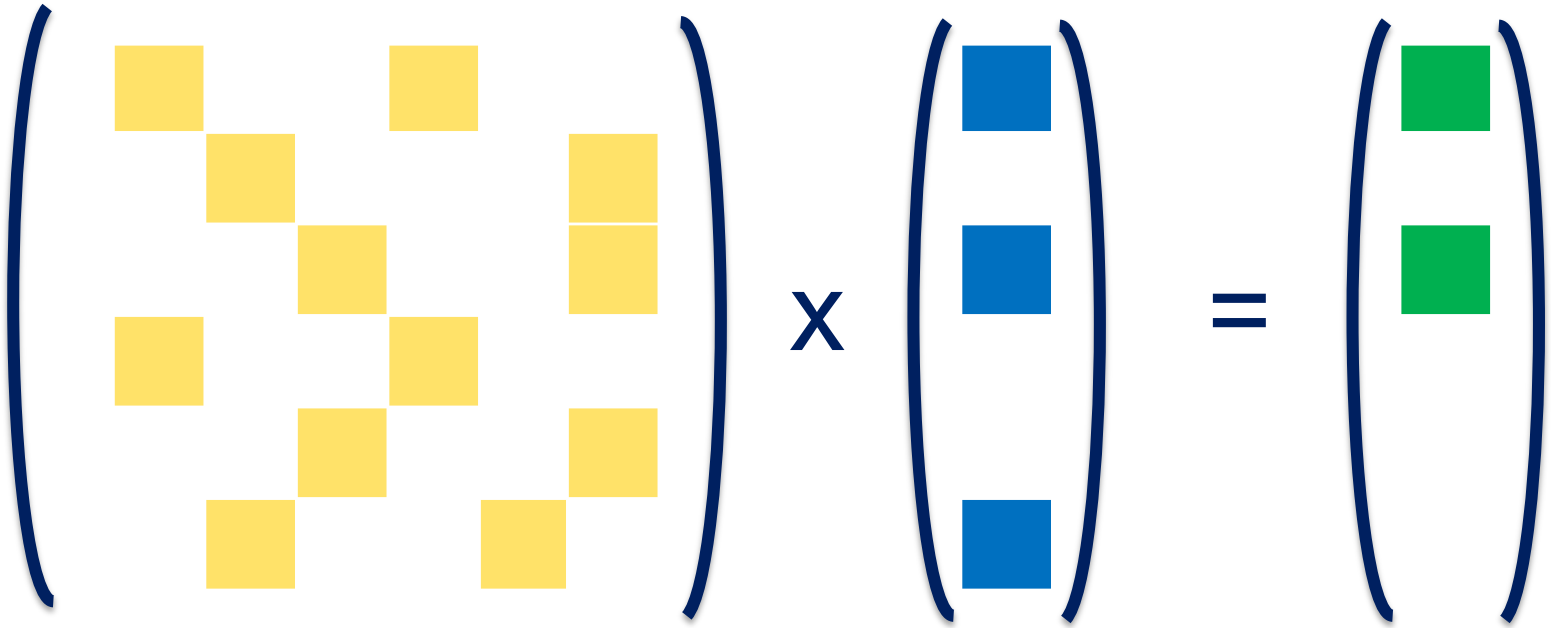
Second Example: BFS

Let Q be the frontier
Let Q' be the next frontier
Let G(V,E) be the Graph
Let B be visited nodes



- Basic measure of graph processing performance
- Single O(n+m) execution, no time for expensive partitioning
- Need 2D block partitioning for power-law graphs

Linear Algebra View: Sparse Matrix Sparse Vector



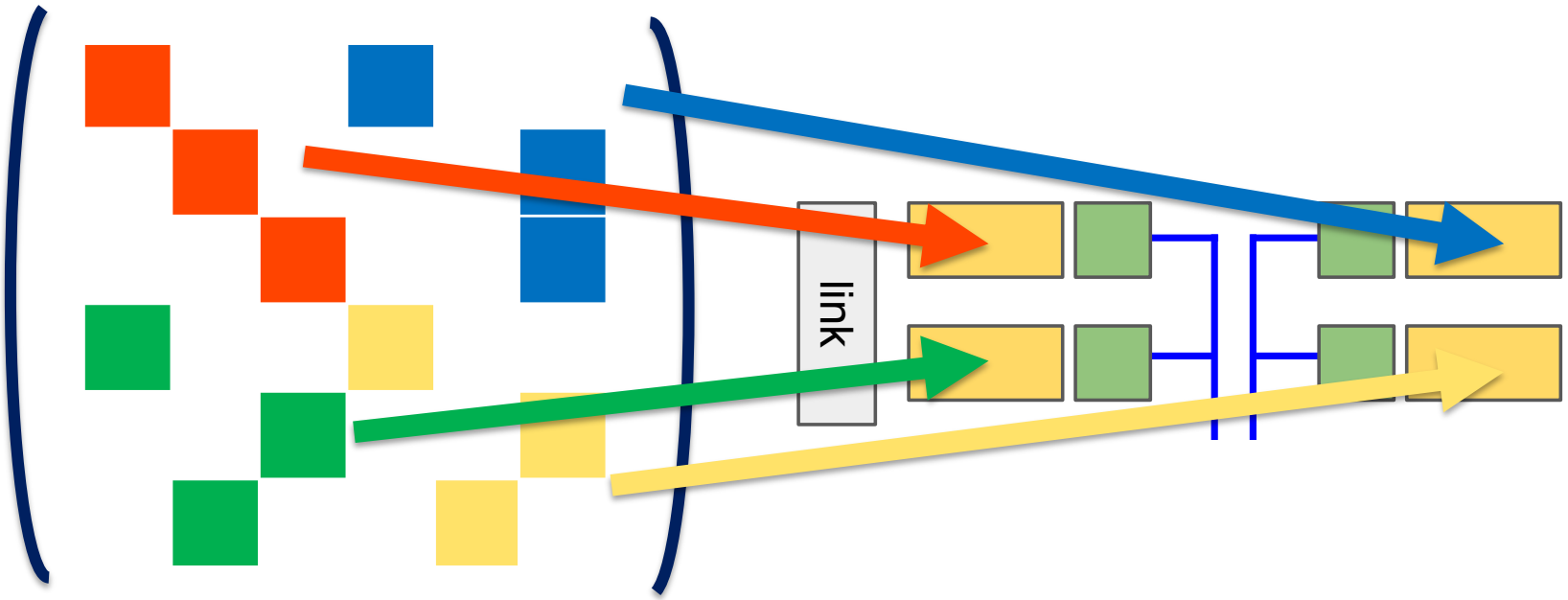
- Common in “true” graph algorithms
- Data dependent computation paths
- Data access pattern irregular and dynamic
- Often memory latency bound

➢ Great potential due to SRAM use

➢ BFS maps directly onto repeated SpMspV

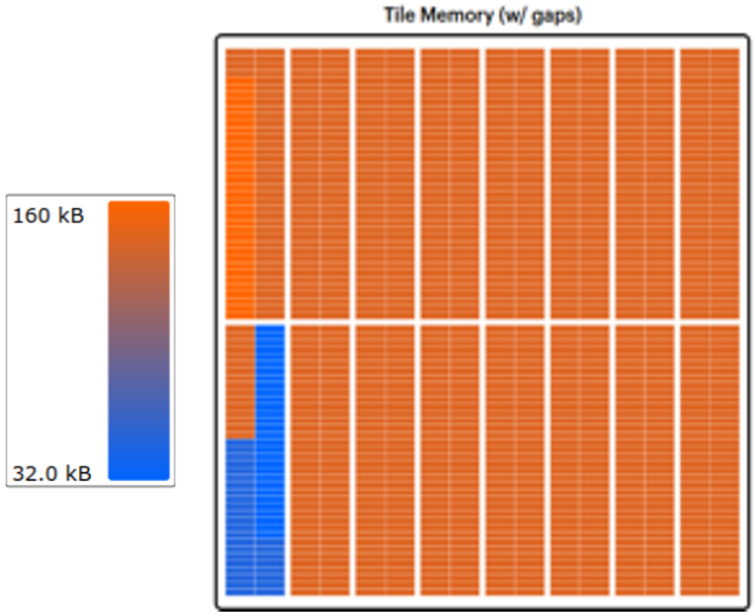
Buluç and Gilbert: The Combinatorial BLAS: Design, implementation, and applications, 2011

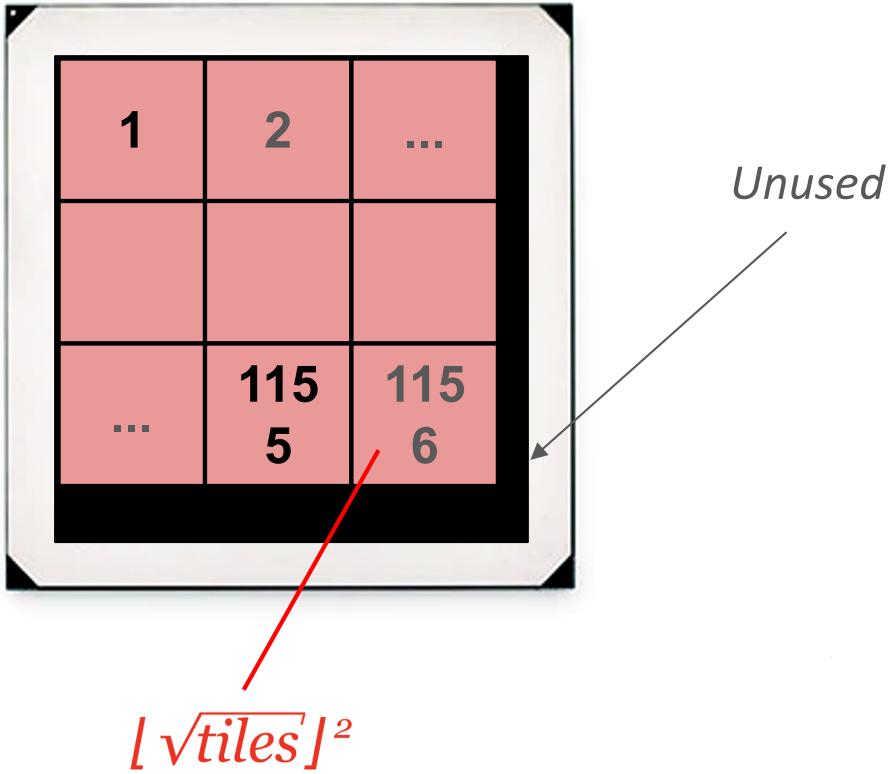
Mapping to the IPU



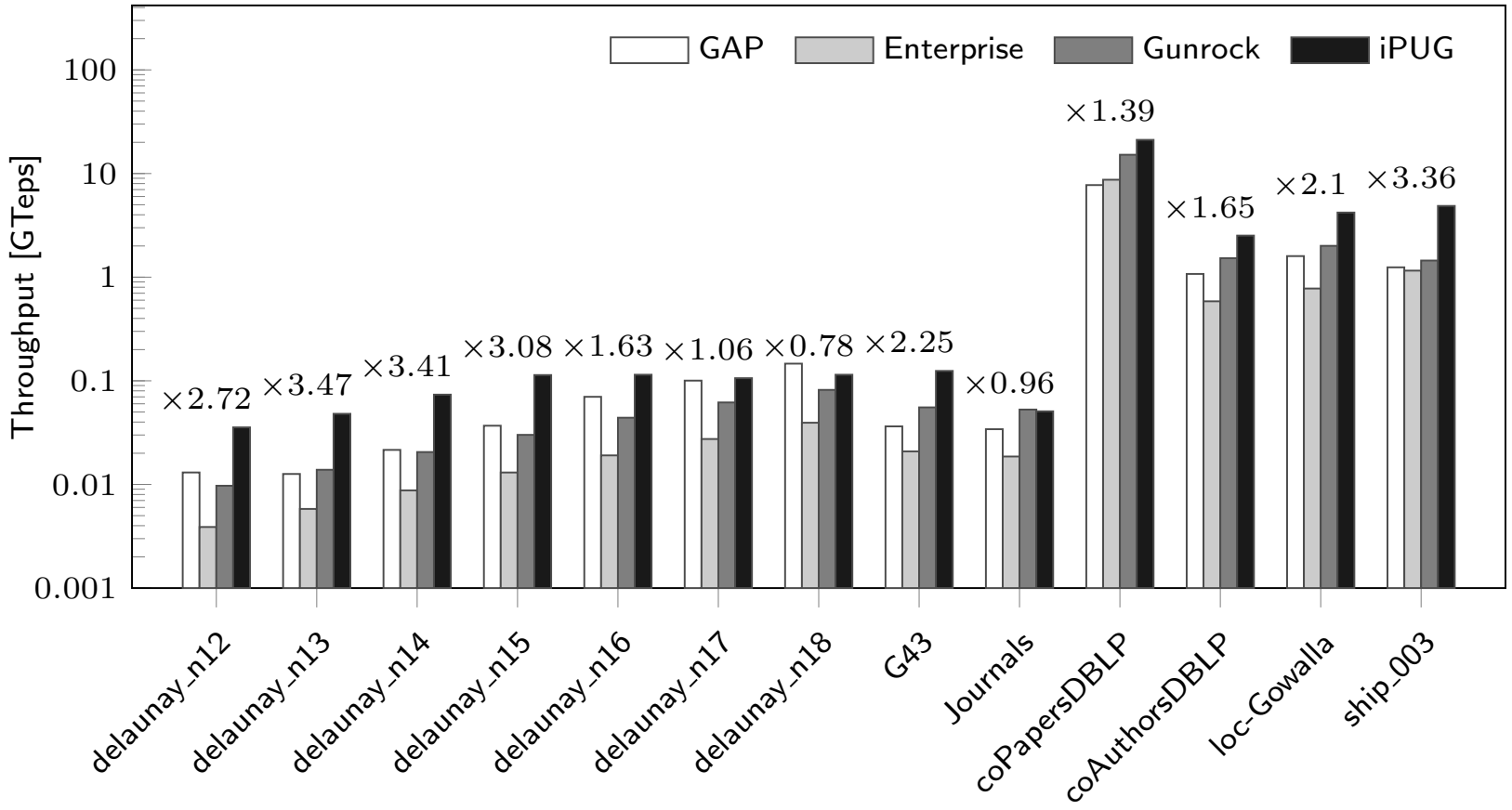
- 2D block partitioning based on nonzero position
- No partitioning cost
- Memory efficient, no need to store partition boundaries
- Partitioning can be improved via suitable reordering

Mapping to the IPU: Need Quadratic Tile Array



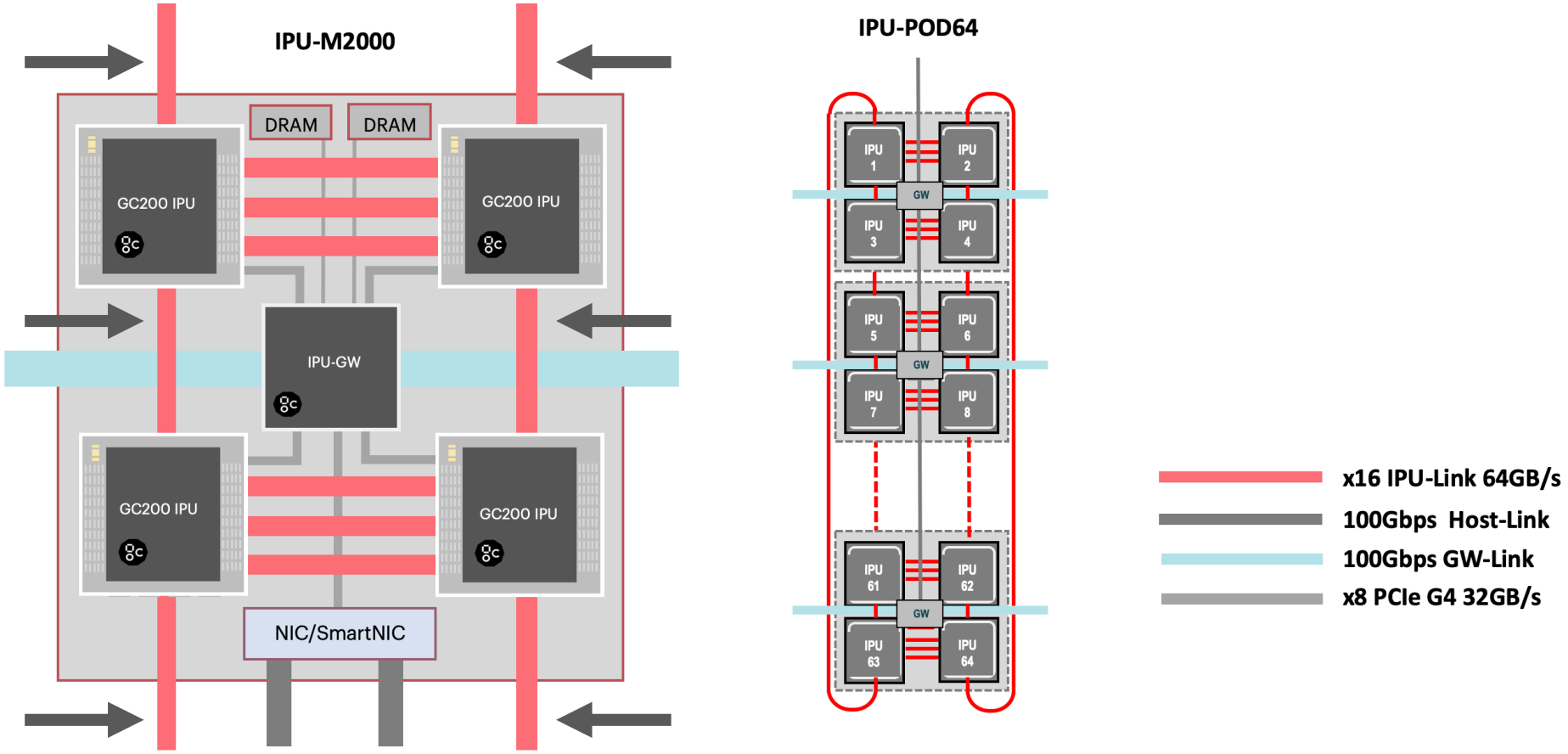


Single Device BFS – IPU vs GPU vs CPU



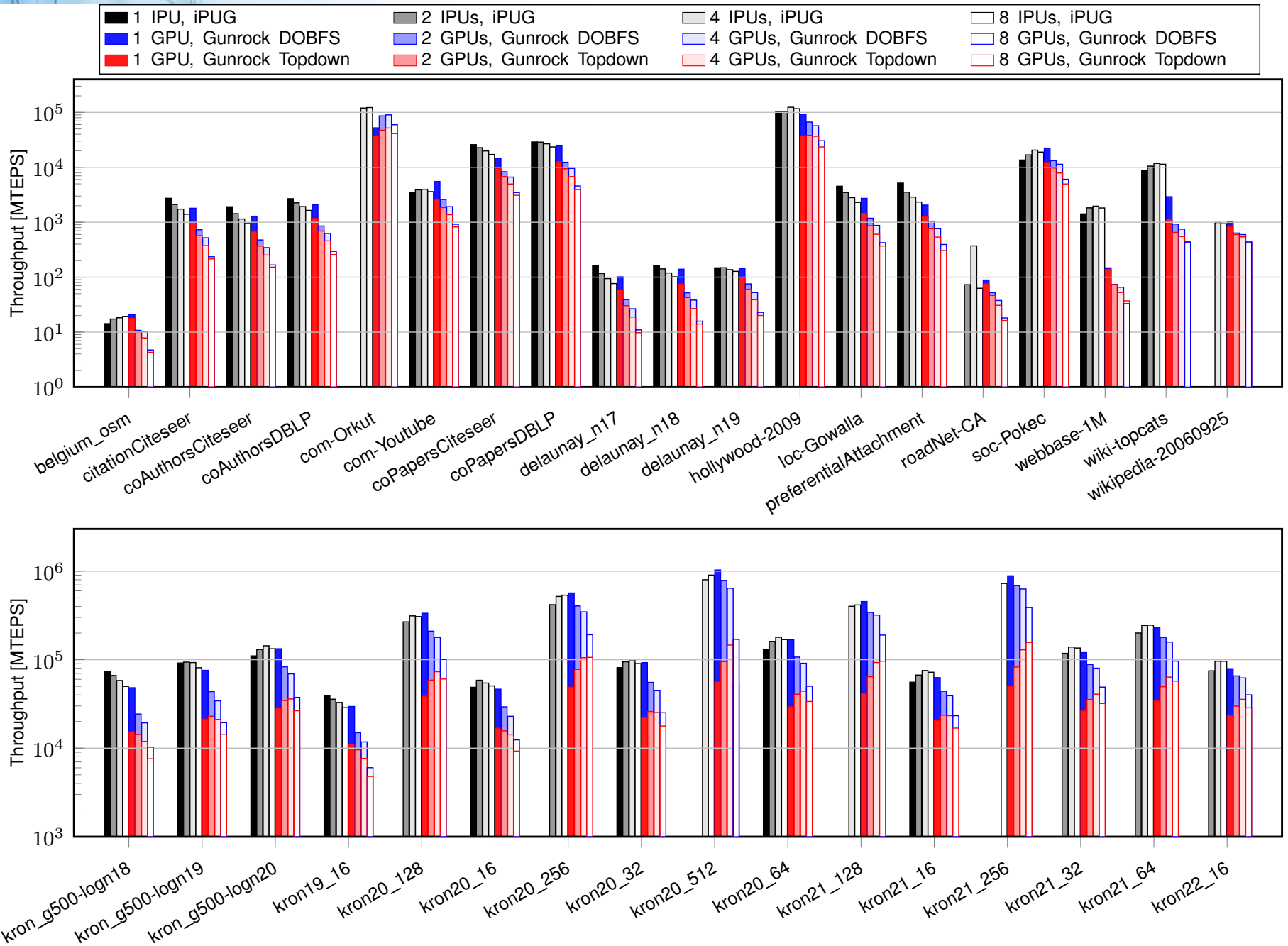
- GC2 IPU (1st generation) beats V100 GPU by about 2x
- Cannot beat CPU on high-diameter graphs
- Good performance/watt, but very limited scope

Scaling Out: Multi-IPU BFS

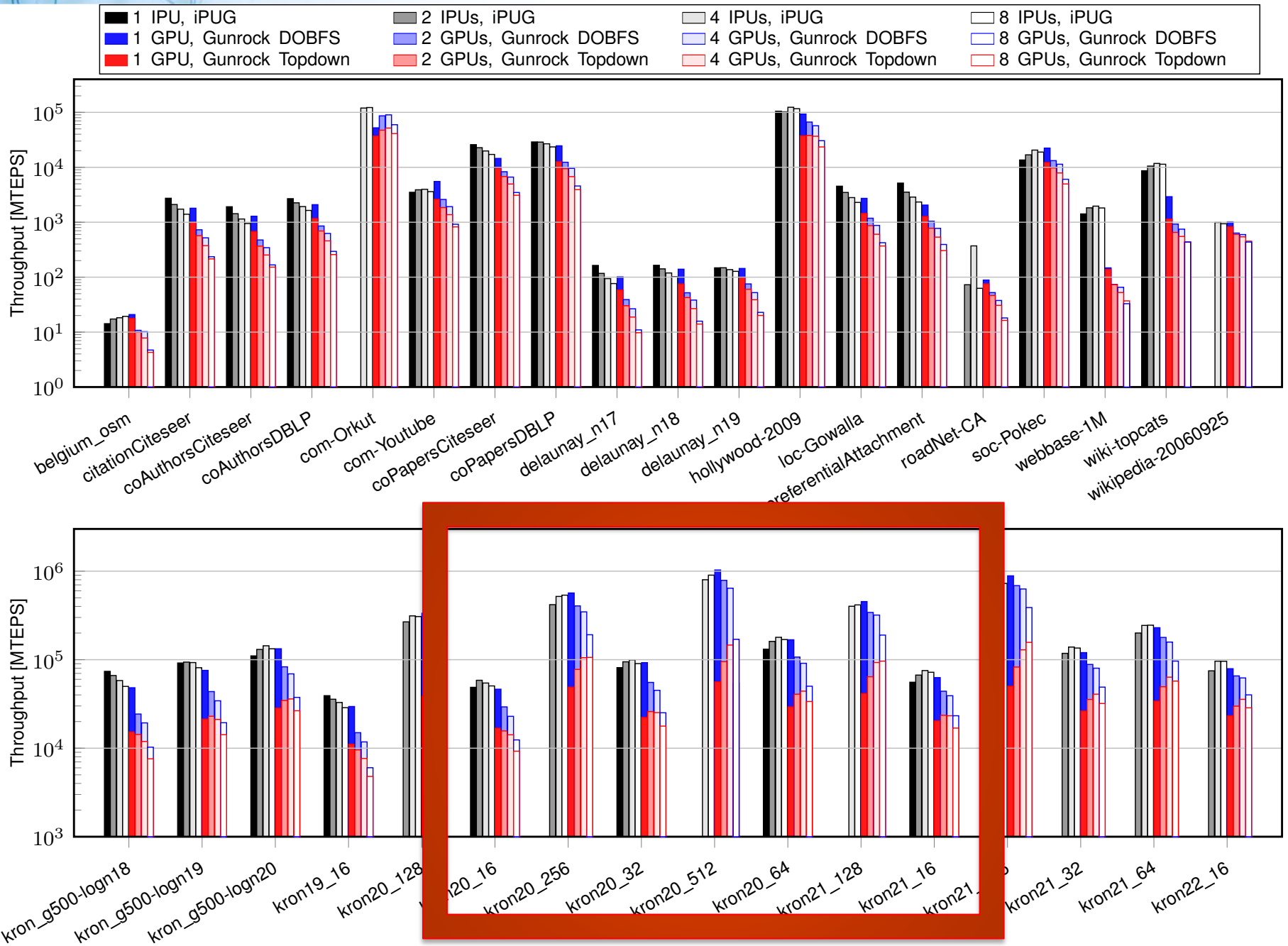


- Computation can scale to multiple IPUs
- BFS is extremely communication-heavy
- Large-scale problems are mostly network dependent

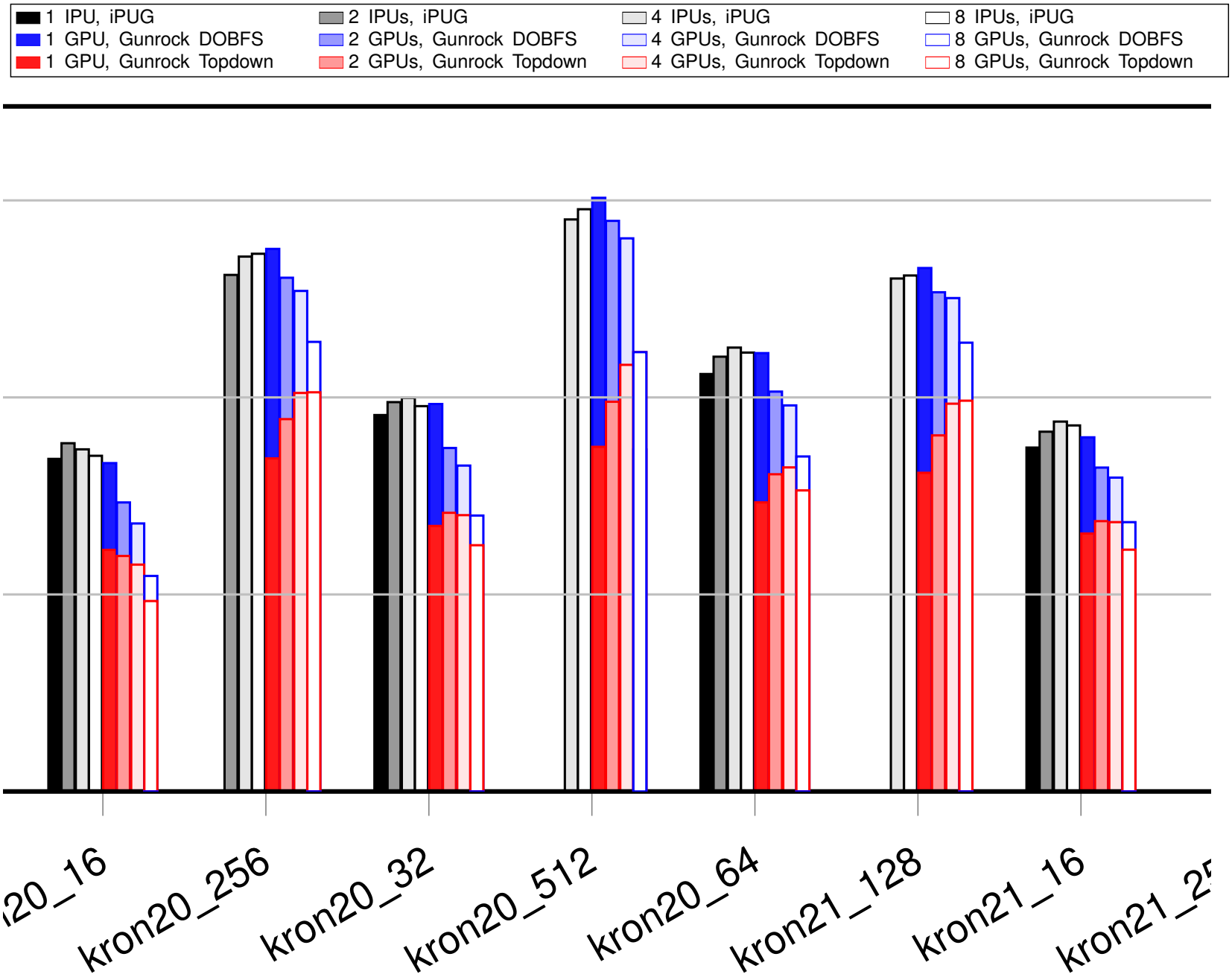
Multiple Device BFS – IPU vs GPU



Multiple Device BFS – IPU vs GPU



Multiple Device BFS – IPU vs GPU



Performance Lessons Learned

- Fast SRAM memory works (mostly) as advertised
- Load balanced memory bound operations (SpMV) very fast
- Very susceptible to bad load balance (hurts BFS results)
- AI oriented -> high FLOPS count restricted to matrix units
- Results update to A100 vs BOW IPU (40% higher clock rate)

Acknowledgement and Future Work

- Show streaming results
  (they are pretty good)
- BFS on Cerebras results
  (works in the simulator)
- Interesting graph algorithm results
  (Matching, centrality, repeated BFS,
  still working on that)
- More advanced partitioning for multi-IPU
  (hierarchical, balance ghost cell count,
  graph constrained partioning, etc.)

Luk Burchard