Scalable Silicon Compute

NIPS 2017 Workshop on Deep Learning at Supercomputer Scale

GRAPHCORE

simon@graphcore.ai

Our comprehension and mechanization of intelligence is nascent

- best models and algorithms change every year
- increasingly heterogeneous composition of models
- huge appetite for more compute throughput

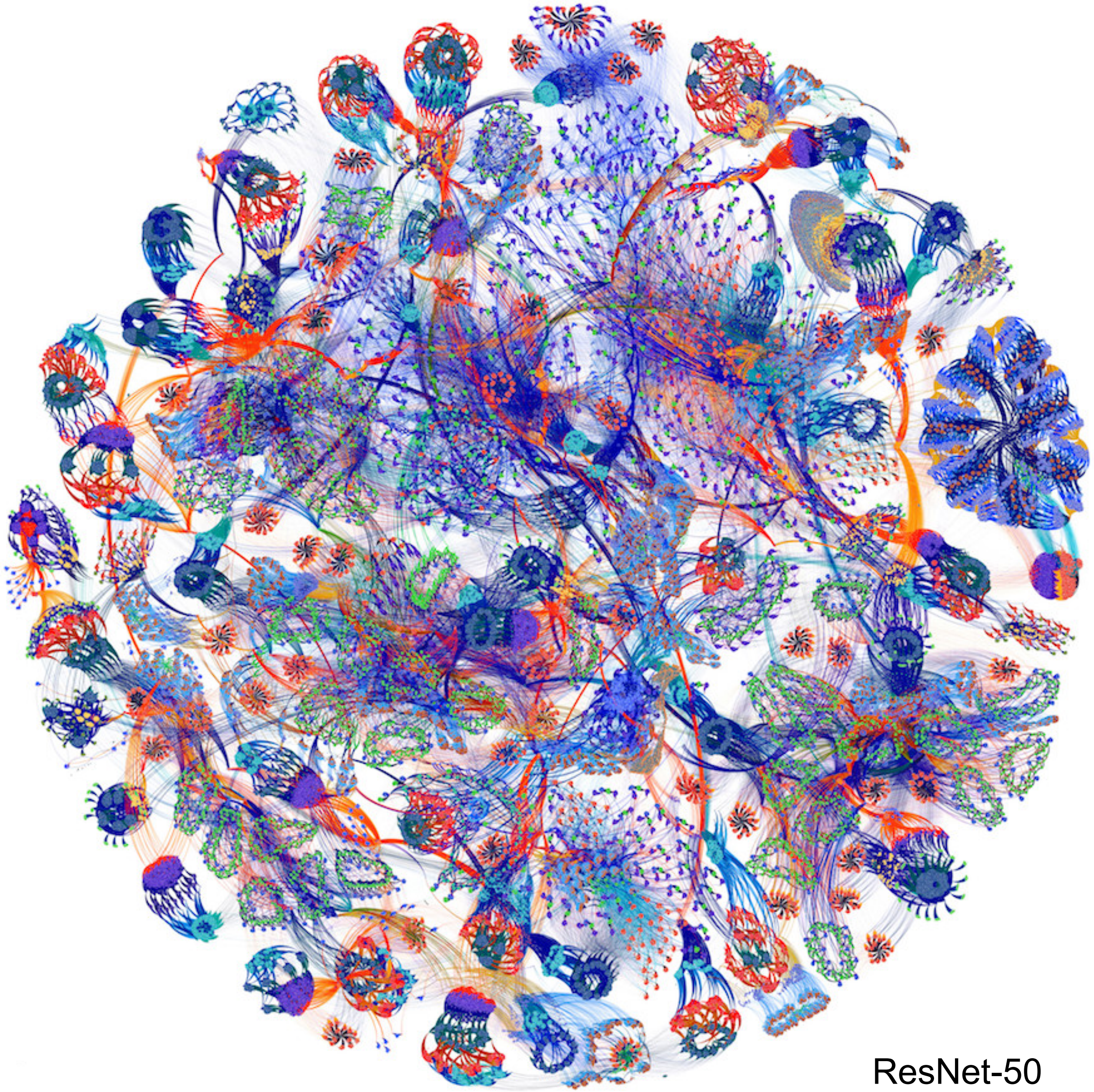
At least until we understand intelligence better, we need machines which...

- exploit massive parallelism
- are agnostic to model structure
- have a simple programming abstraction
- are efficient for both learning and inference

Graphs can represent arbitrary model structure and expose compute parallelism.

We need to expose a lot of parallelism...
  O(1000) work items/processor
x O(1000) processors/chip
x O(1000) chips/system

In a parallel computer, communication is part of the computation. It's attractive to compile "communication kernels", as well as compute kernels.

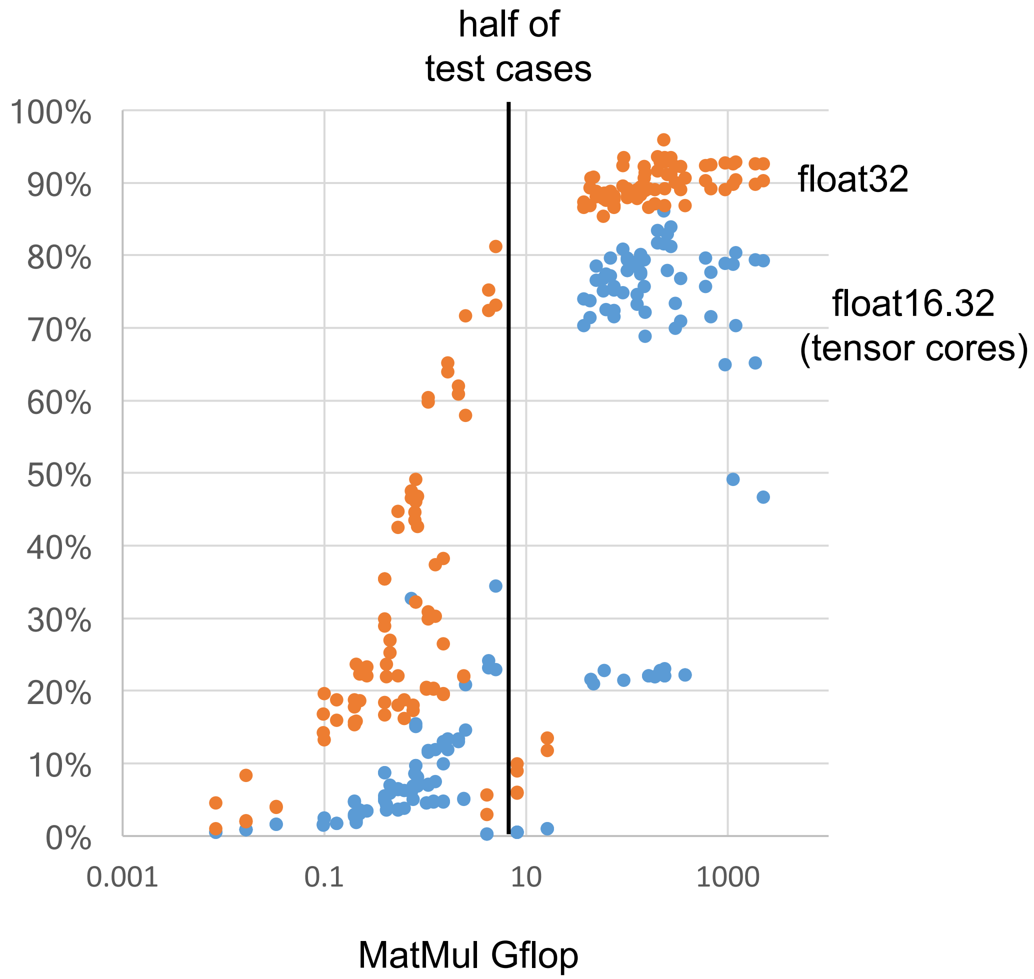


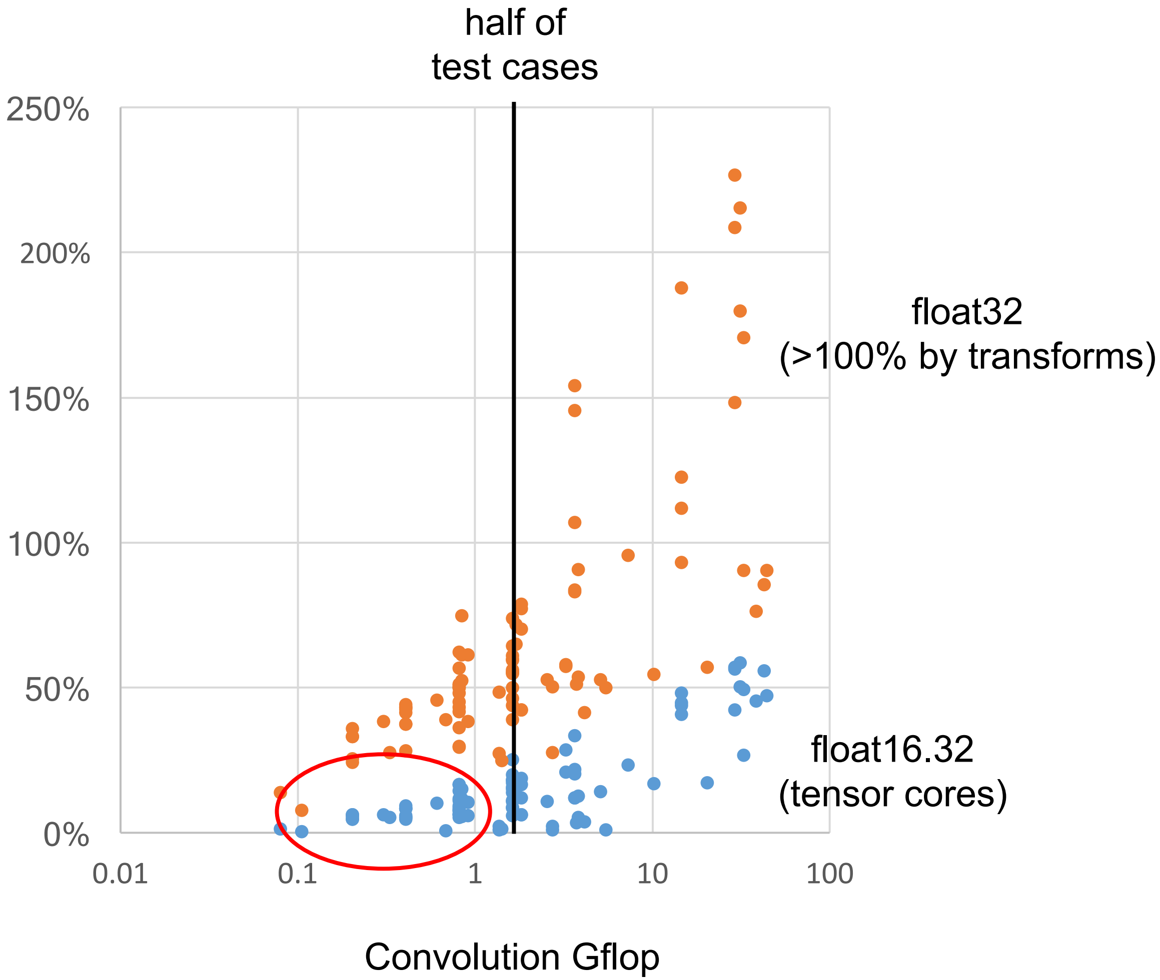
ResNet-50
training batch=4

Types of Parallelism

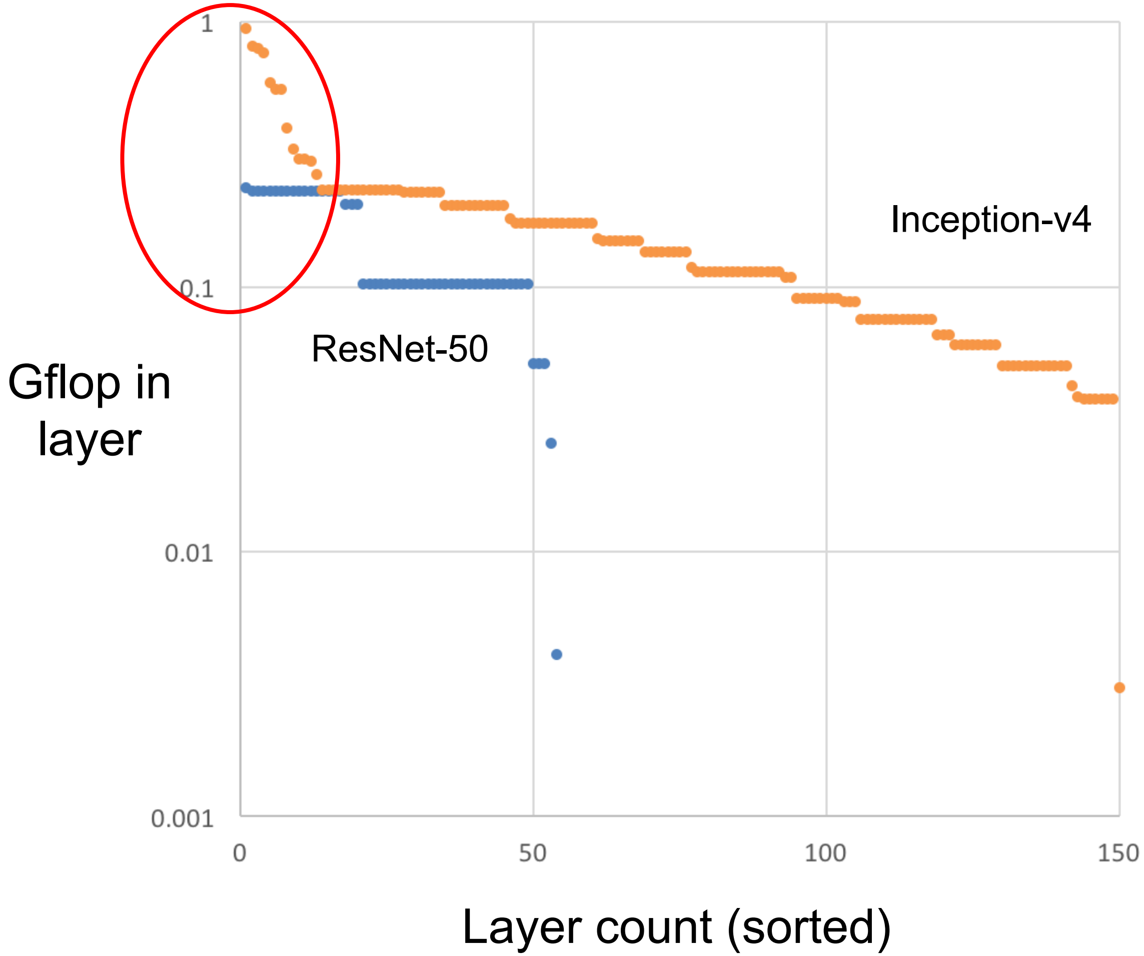
- kernel partitioning
- pipelining
- parallel paths
- model clones sharing parameters, different data
- model clones with different parameters/cost functions/hyper-params, shared data
- multiple models of different structure
- …

Kernel partitioning exhausted?





Volta V100 fmac utilization on DeepBench


1
0.1
0.01
0.001
0
50
100
150
Gflop in layer
Layer count (sorted)
ResNet-50
Inception-v4

Challenges for the Age of Parallel Processors

- Power
- Memory
- Abstraction

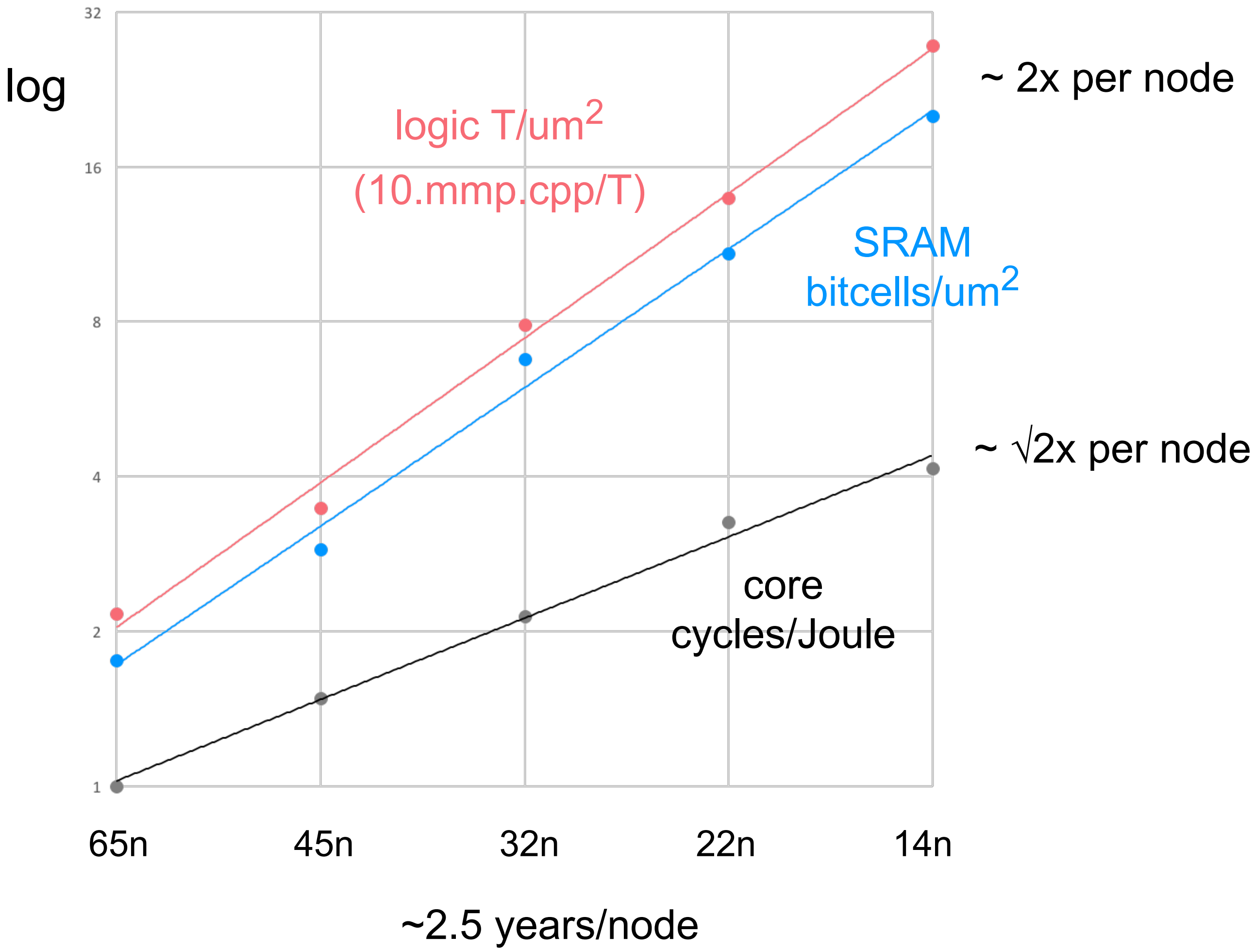
As parallelism goes up, local resources per processor go down.

Post-Dennard Silicon Scaling

- Transistors / chip ~30% / year
- Compute / Watt ~15% / year

Big Xeon exemplars:

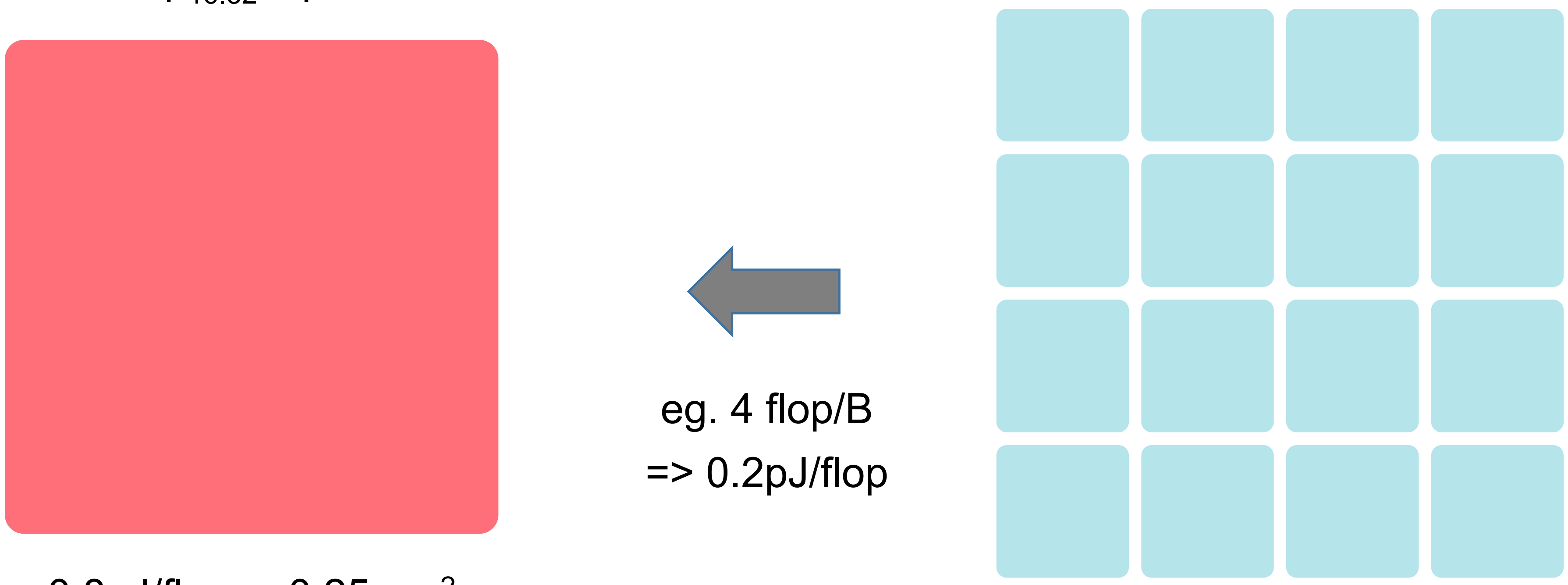
65n	Woodcrest X5160 2c Conroe X3085 2c
45n	Lynnfield X3470 4c Nehalem X7550 8c
32n	Westmere E7-8870 10c SandyBridge E5-2690 8c
22n	IvyBridge E7-2890v2 15c Haswell E7-8890v3 18c
14n	Broadwell E7-8894v4 24c Skylake 8180P 28c



Silicon efficiency is the full use of available power

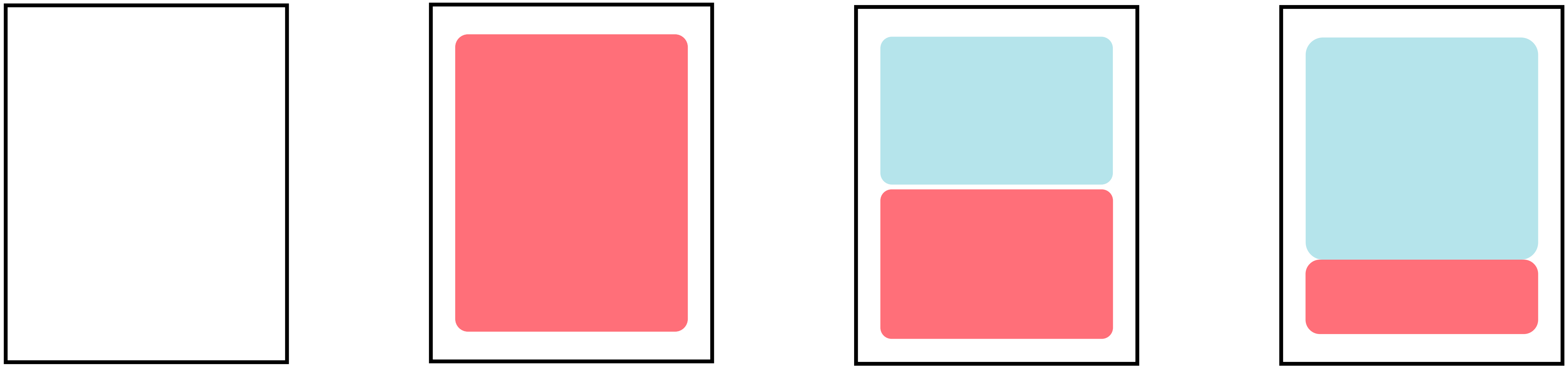
- Keep data local
- Serialise communication and compute
- Re-compute what you can't remember

Proximity of memory is defined by energy, more than latency.



(rough metrics for 16nm)

All logic chips are power limited



Largest manufacturable die ~825mm^2

500Tflop$_{16.32}$
500W
NOK

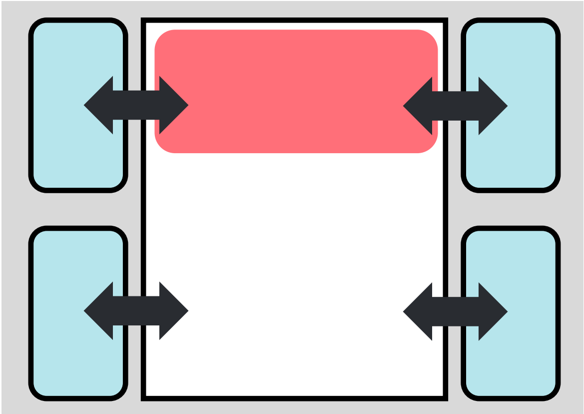
250Tflop$_{16.32}$
225MB
250W

125Tflop$_{16.32}$
337MB
125W

Processor + memory systems @ 240W (for 300W card)

DRAM on interposer

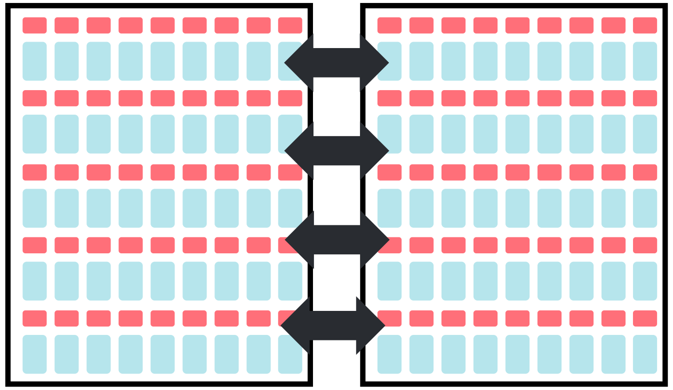
180W GPU + 60W HBM2



16GB @ 900GB/s

Distributed SRAM on chip

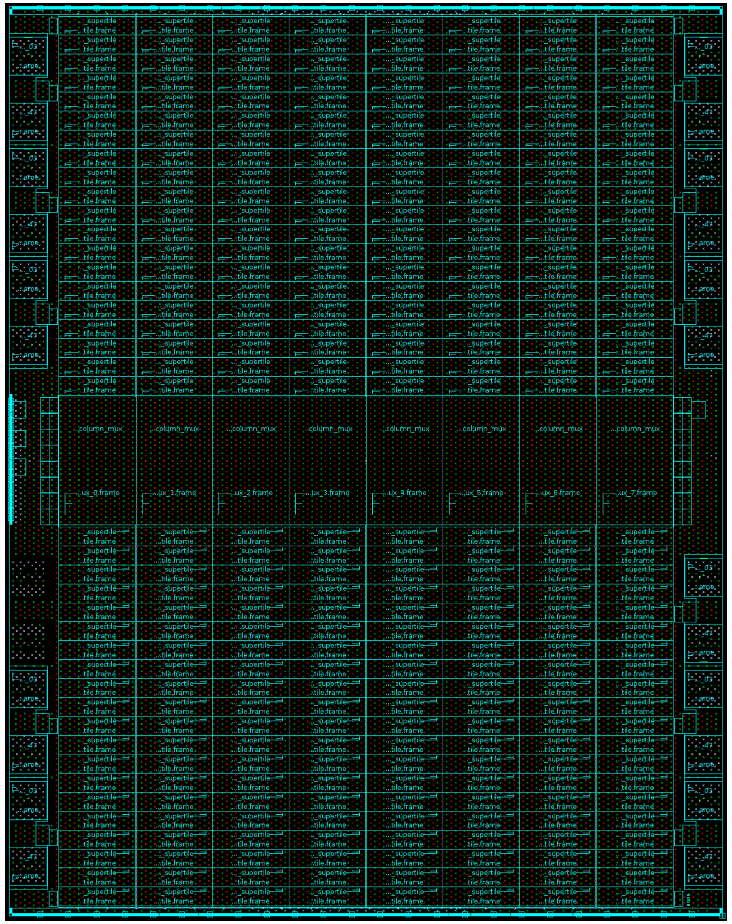
120W IPU x2



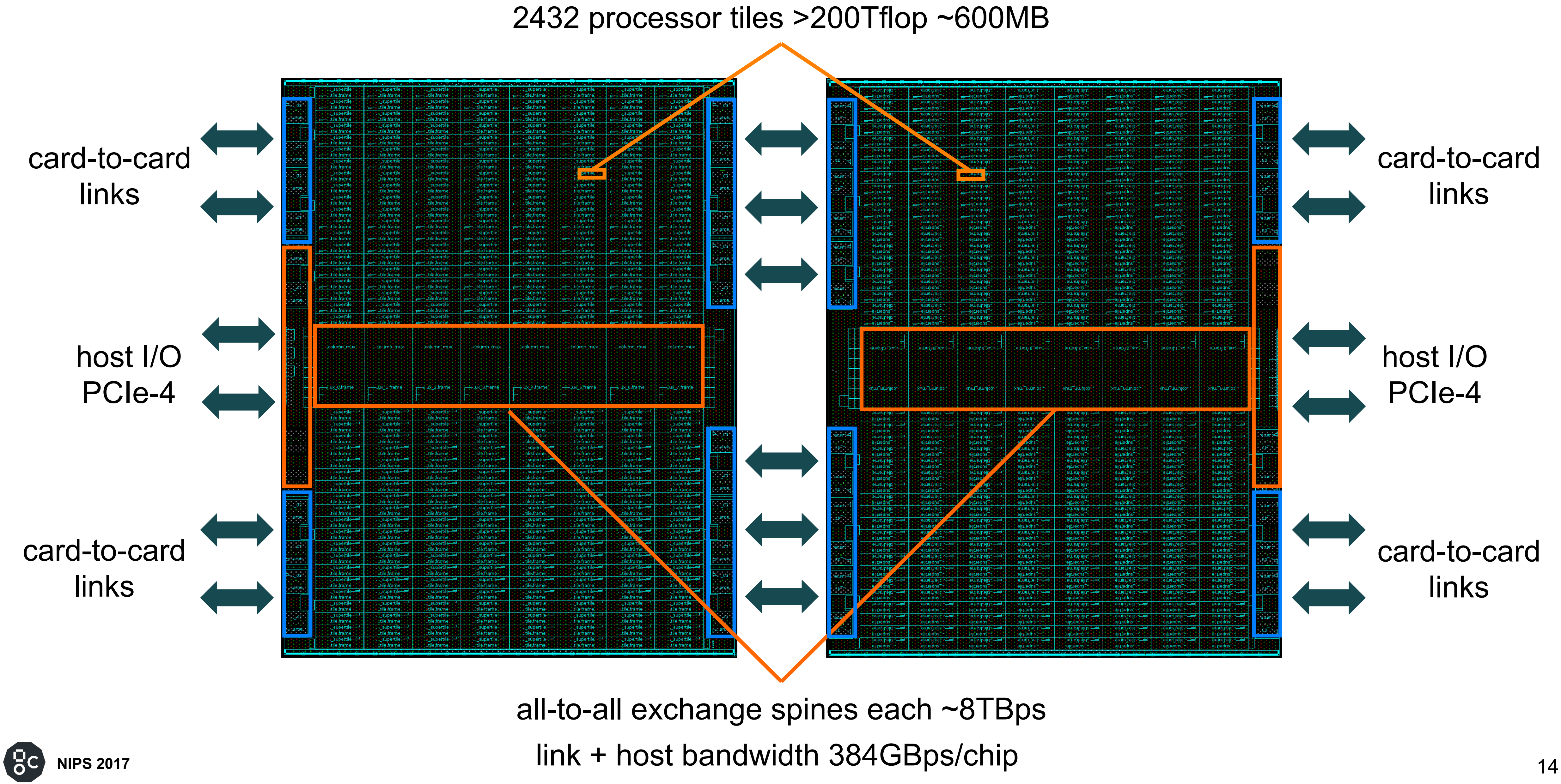
600MB @ 90TB/s

“Colossus” IPU
(in honour of Tommy Flowers)

- All-new pure distributed multi-processor for MI.
- Mostly memory, for “model on chip”.
- A cluster of IPUs acts like a bigger IPU.
- Bulk Synchronous Parallel (BSP) execution.
- Stepwise-compiled deterministic communication.
- Programmed using standard frameworks (TensorFlow, ...) over Poplar™ native graph abstraction.



Colossus pair on a PCIe card



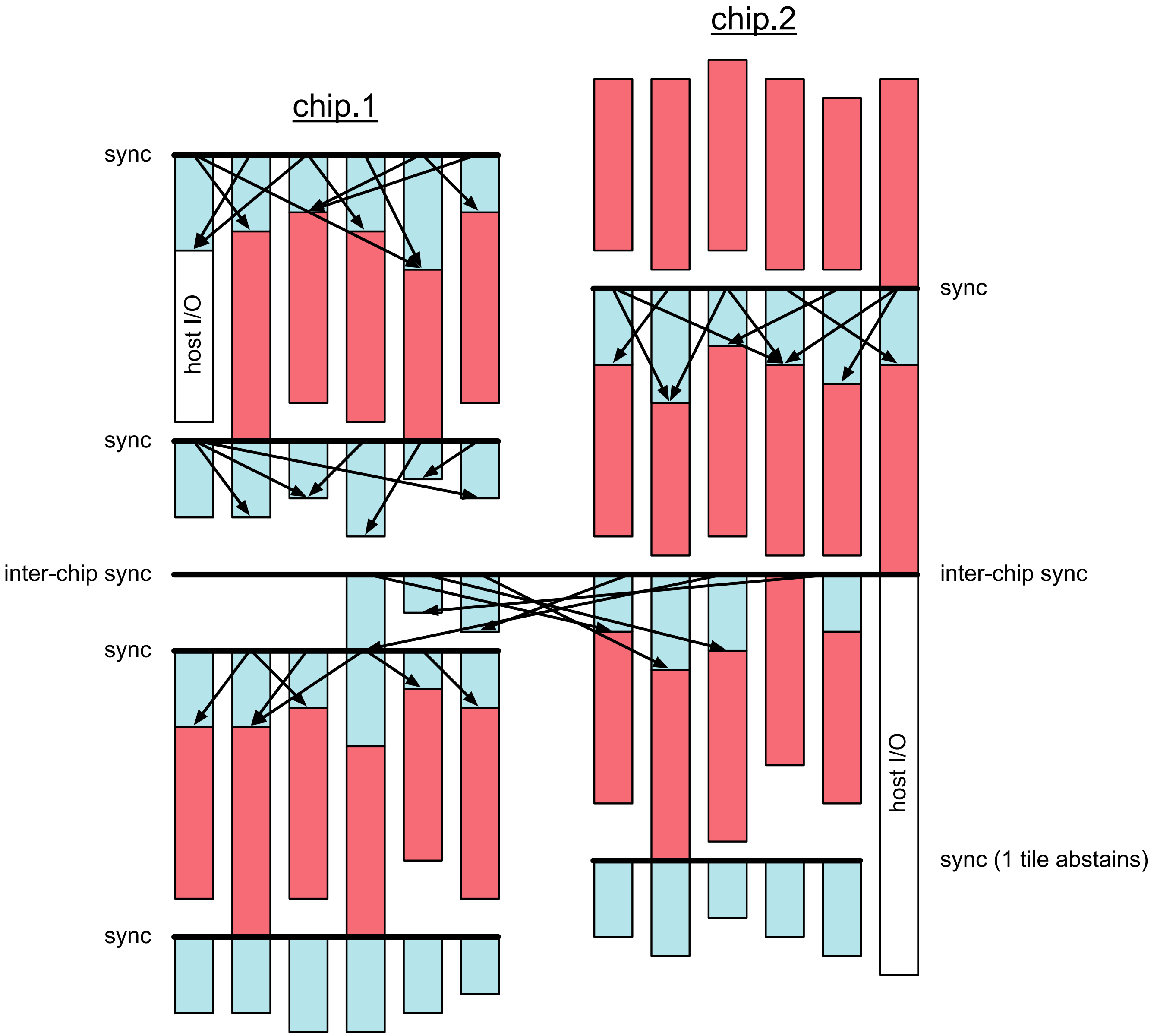
Bulk Synchronous Parallel

Massive parallelism with no concurrency hazards

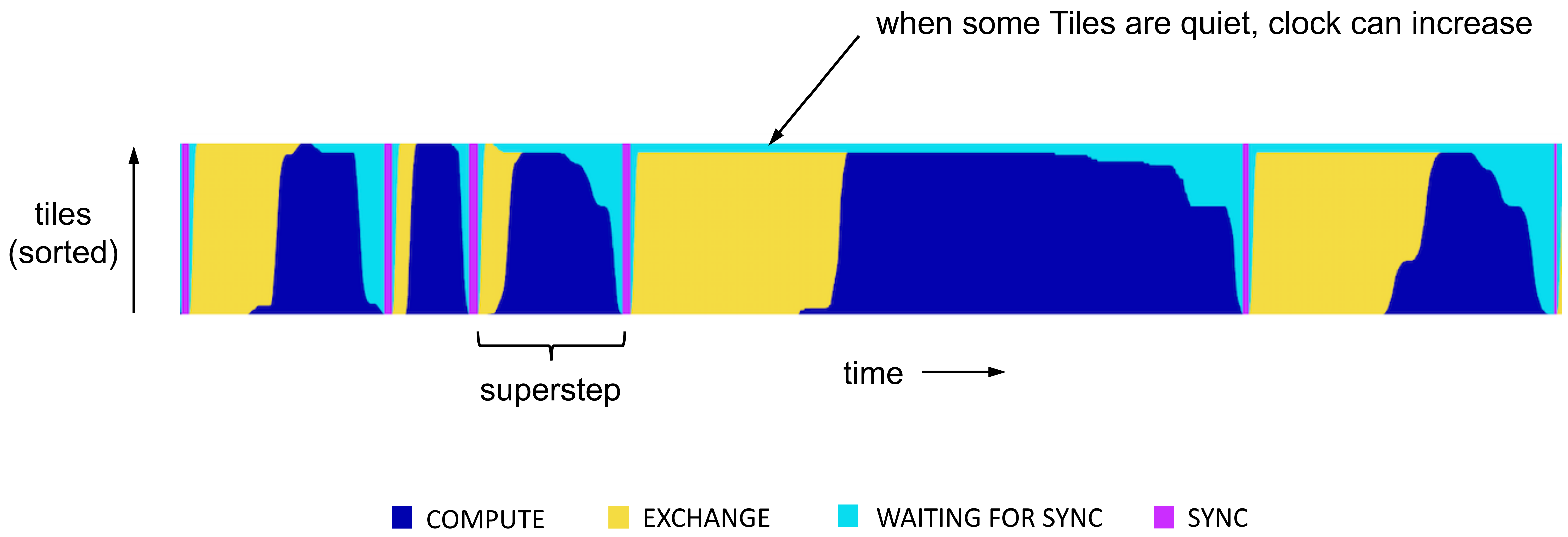
compute phase

exchange phase

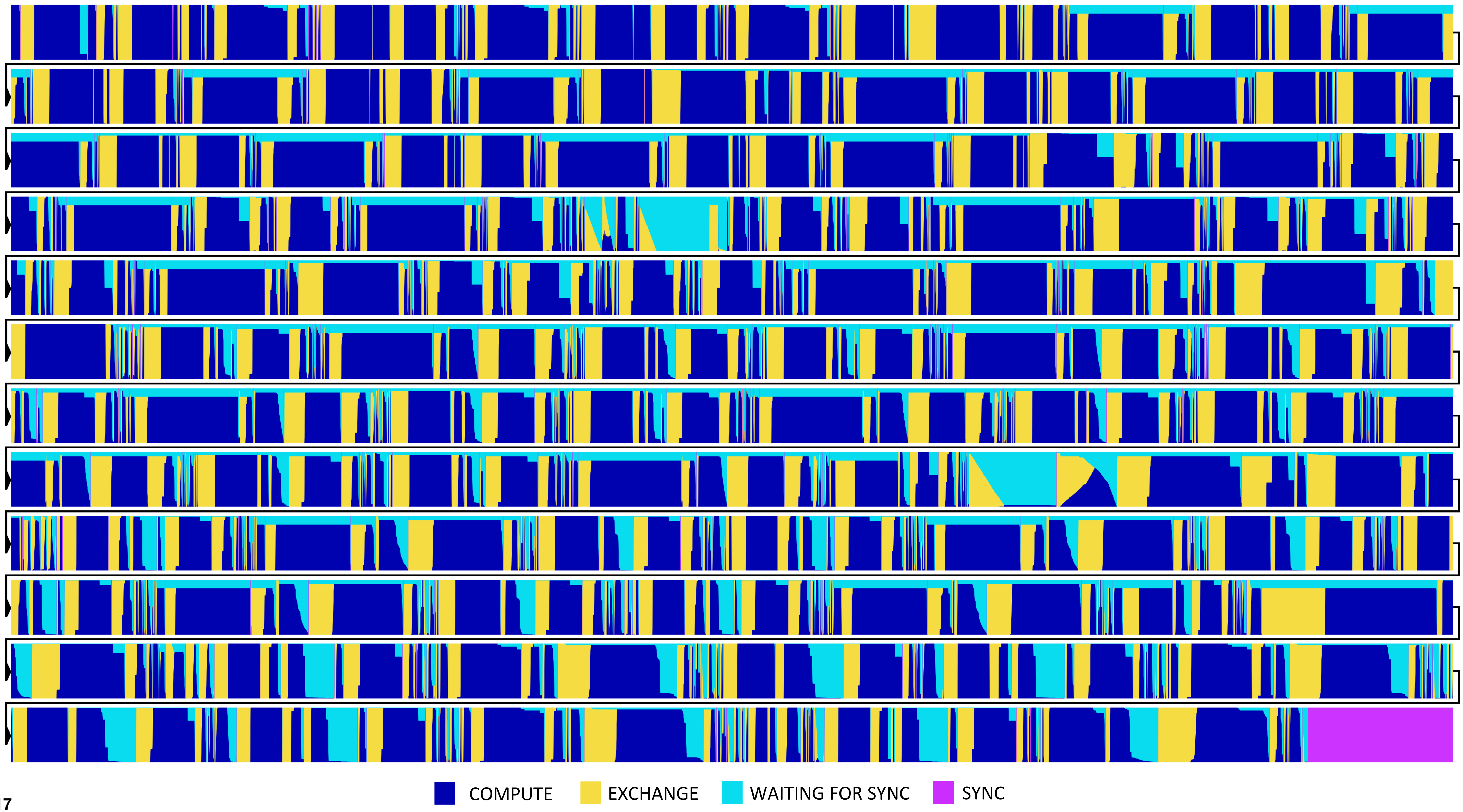
Communication patterns are compiled, but dynamically selected.



BSP Execution Trace



BSP Trace: ResNet-50 training, batch=4



Managing Memory

IPU has smaller memory than GPU, but greater effective compute:

- Use small batches per IPU.
- Trade compute for memory.
- Use memory-efficient algorithms/models

An *efficient* gradient learner will converge in a minimum number of parameter updates ...so smaller batches should learn faster, as well as generalizing better.

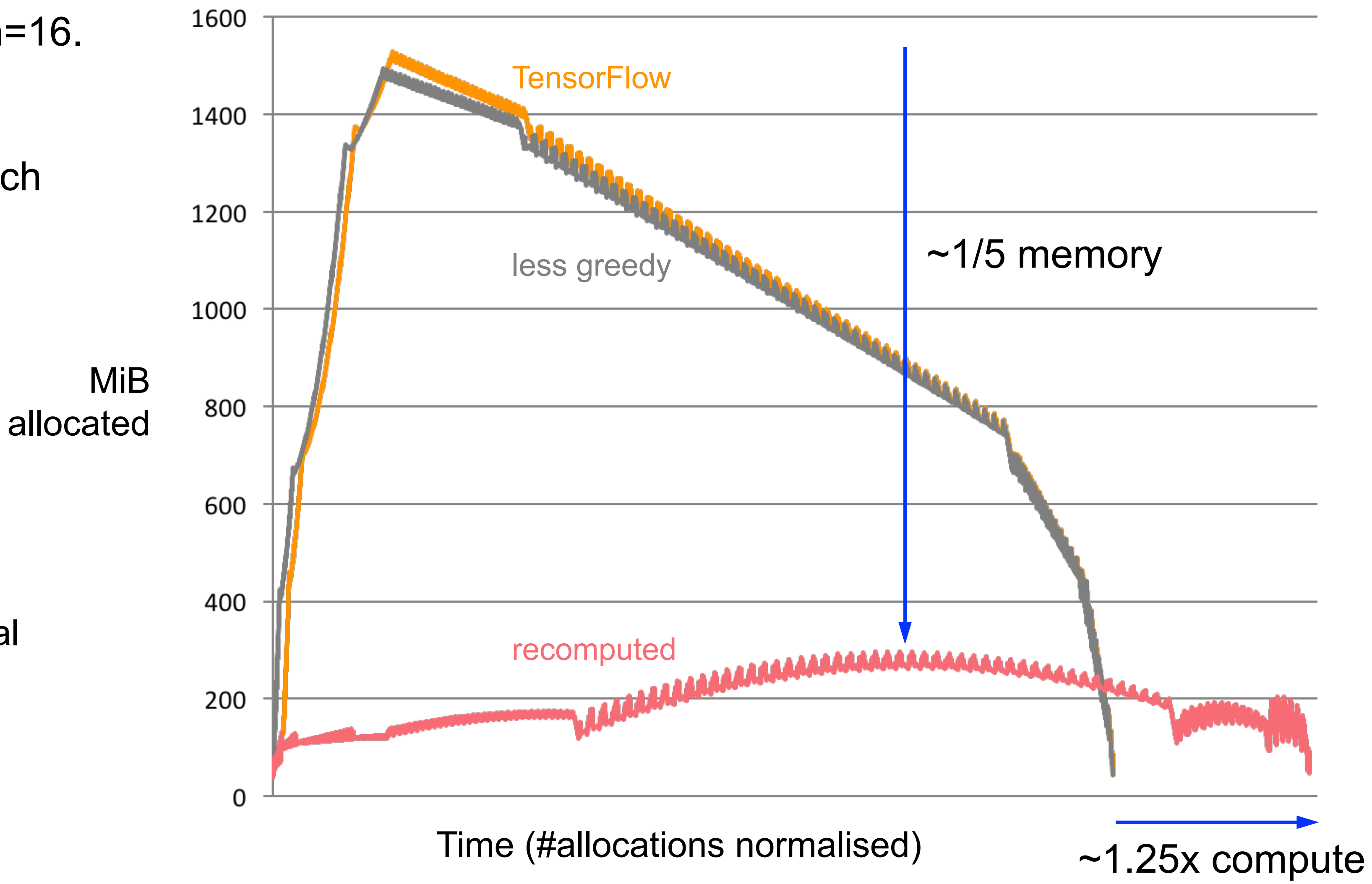
Efficient small batch machines allow learning to parallelise over more machines.

Re-compute what you can't remember

DenseNet-201 training, batch=16.

Naive strategy: memorize activations only at input of each residue block, recompute all others in backward pass.

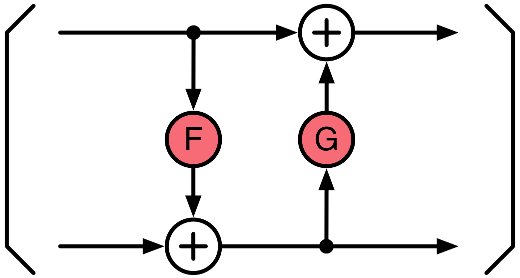
Executing on CPU, recording total memory allocated for weights + activations. Float16 weights and activations, single weight copy.



Reversible Nets

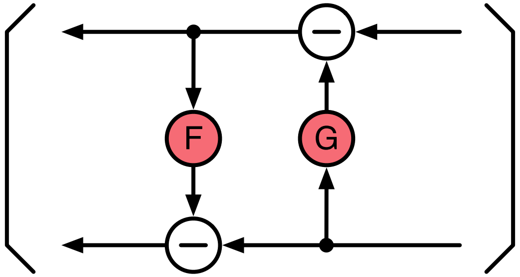
arxiv.org/abs/1707.04585

2 sets of activations, layer N



2 sets of activations, layer N+1

2 sets of activations, layer N



2 sets of activations, layer N+1

No need to save any activations, except for layers which lose information (pool, stride).

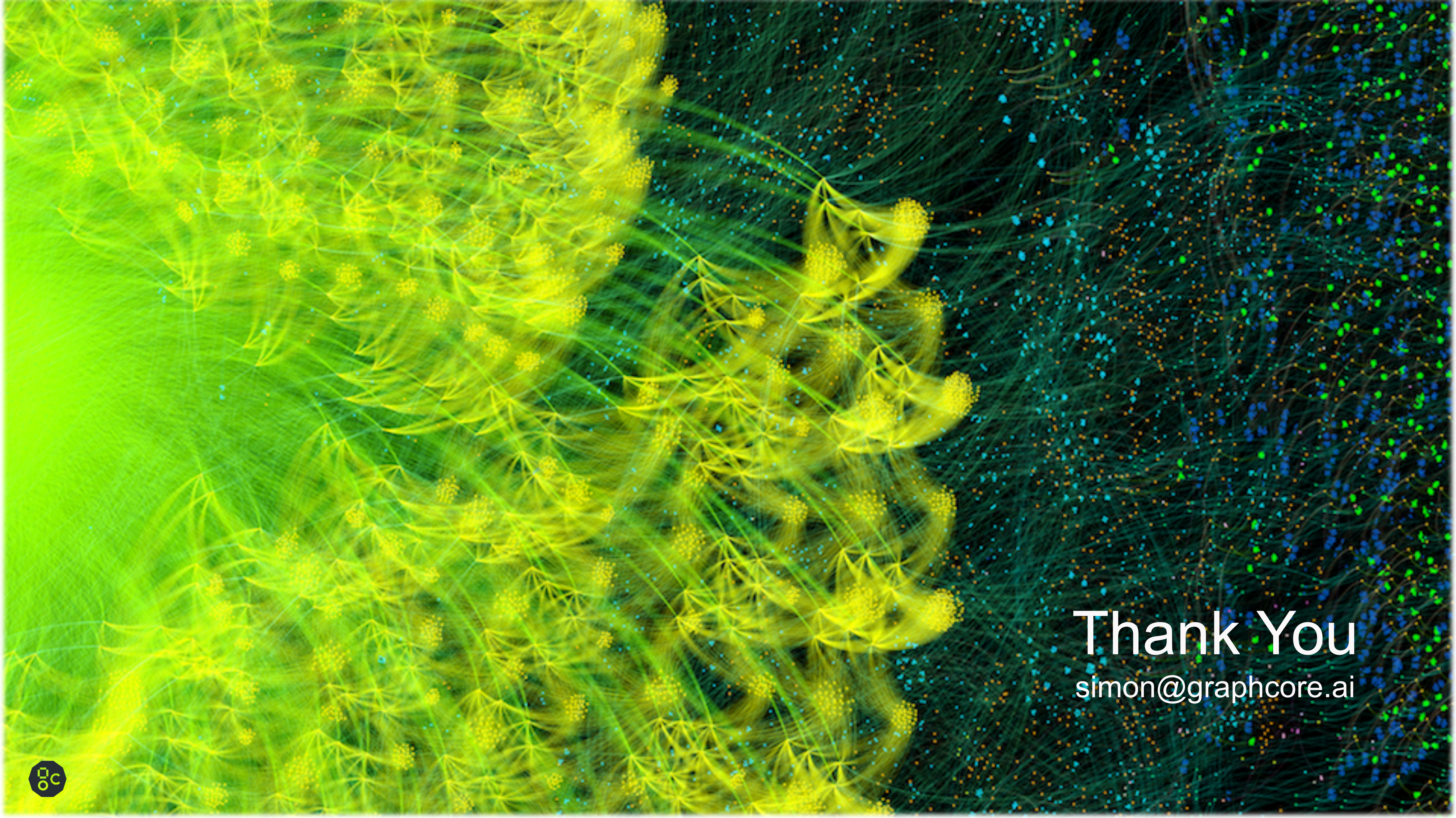
Some preliminary IPU benchmarks...

- ResNet-50 training at 2,250 images/s on 1 card with batch=8
  16,000 image/s over 8 cards with batch=64

- DeepBench LSTM inference (per layer, 1536 hidden units, 50 steps)
  60,000 iteration/s on 1 card at 7ms latency

- 600 full WaveNet voice generators on 1 card at 16k sample/s
  (MOS 3.35, 20 layers, 64 residual channels, 128 skip channels)

More at www.graphcore.ai



Thank You

simon@graphcore.ai