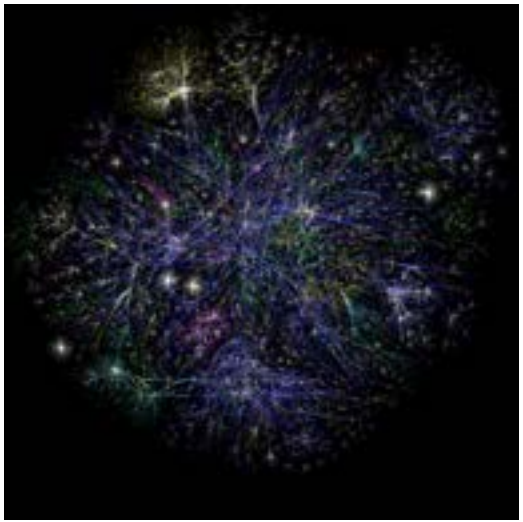
Engines for Intelligence

Simon Knowles, CTO XMOS

BCS London, 14-May-15



Intelligence?

Intelligent computing aims to solve problems…

- For which no efficient exact algorithm can exist (NP hard)
- For which we can't conceive an exact algorithm, or it's too expensive to run.
- Where there's not enough input information for exactness.

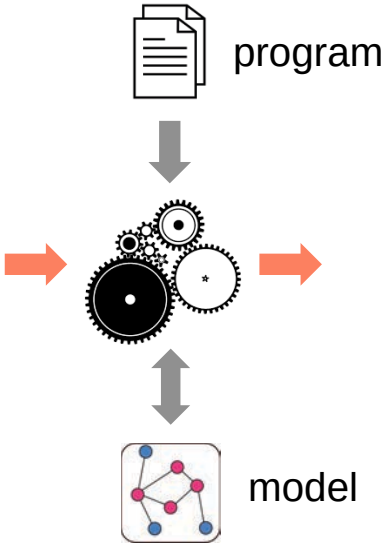
Intelligence requires "experience" = a model learned from data.

All results all probabilistic. Decision policies decide what to do with the uncertainty.

Intelligent machines may correctly arrive at incorrect answers, like humans.

The end of the primacy of the program

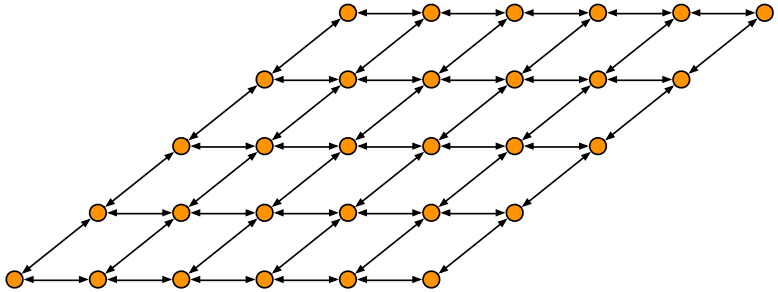
- Machines will act according to learned probability models, mediated by programs.
- Very many model parameters, which are dimensions … sparsity must be high for utility.
- Arbitrary sparse structures are well represented by graphs.
- High topological dimension … little memory access locality.
- Model structure, including hierarchy, helps reduce dimensionality.



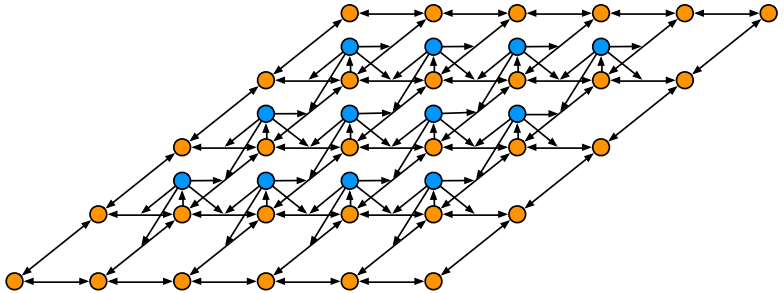
Markov Random Field

Strong local 2D prior, eg. expressing the likelihood that neighbour vertices will have similar state…

- Image denoising, vertices are pixels
- Depth from disparity, vertices are pixel pairs



Blue vertices can break the prior by switching off edge correlations.

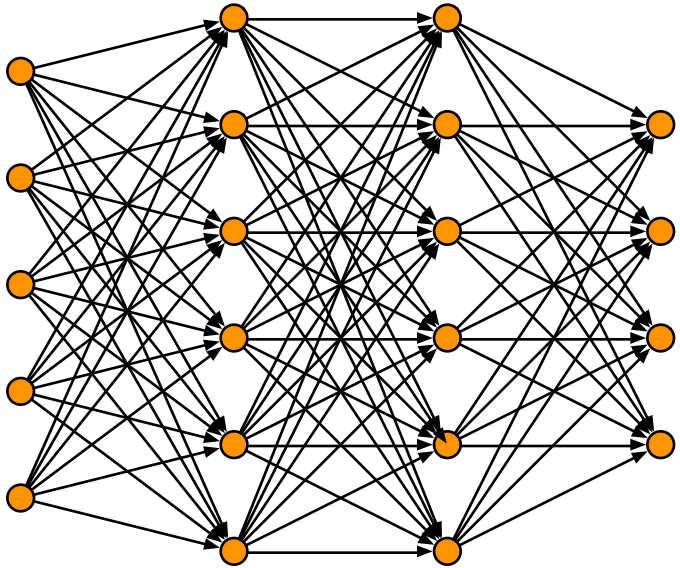


Neural Net

Millions ~ billions of learned parameters are effective.

Deep hierarchy is important.

Days ~ months of training on GPUs.

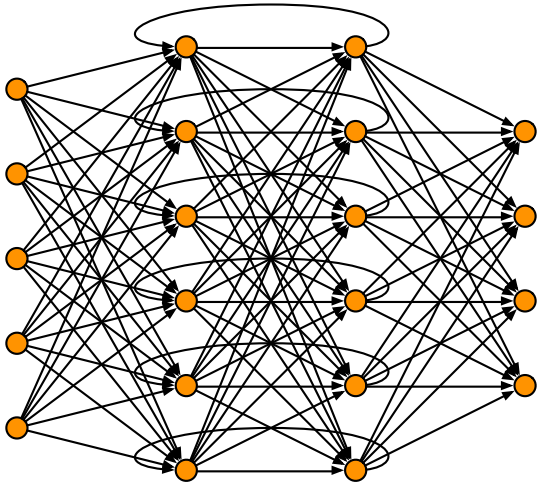


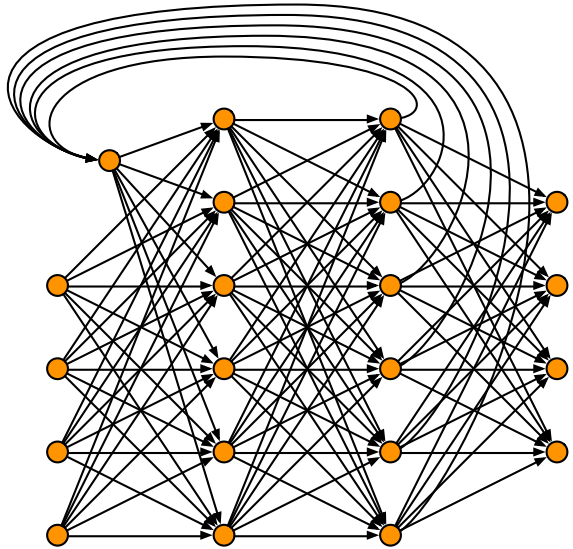
Recurrent NN

The basis of temporal memory, eg. LSTM.

Many possible structures.

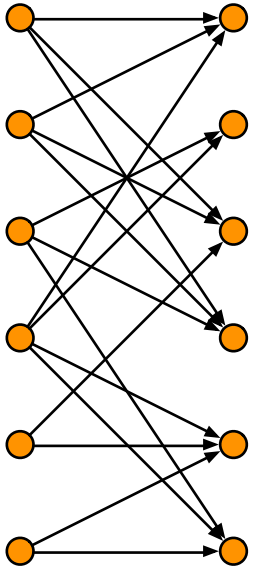
Mammalian cortex has more feed-back axons then feed-forward.

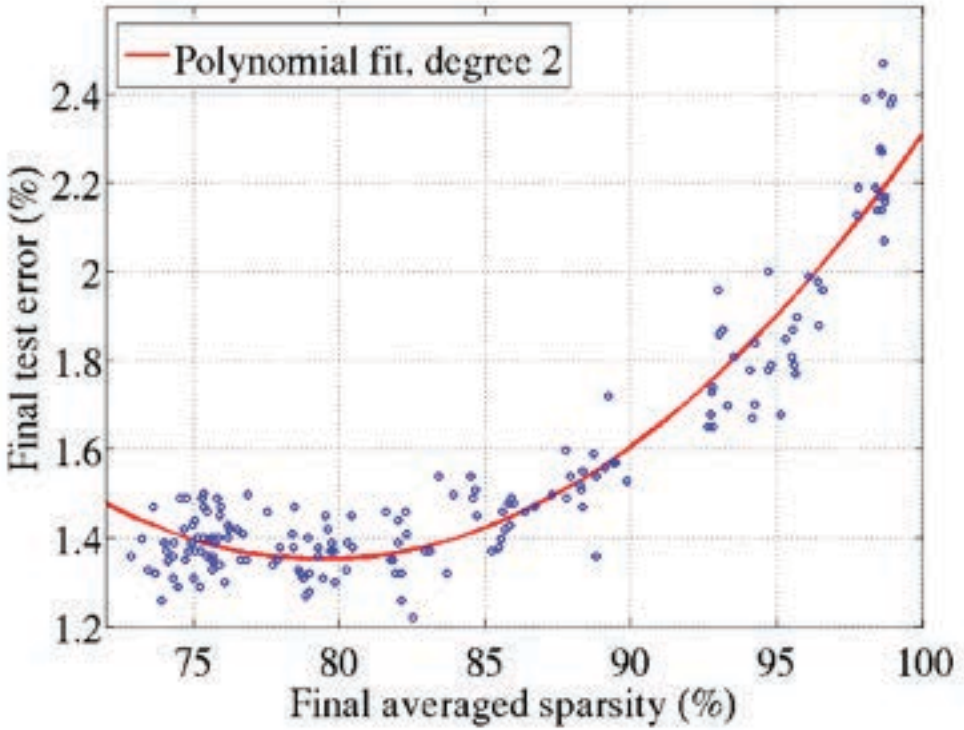




Thinning out trained nets

Deployed NNs may be quite sparse





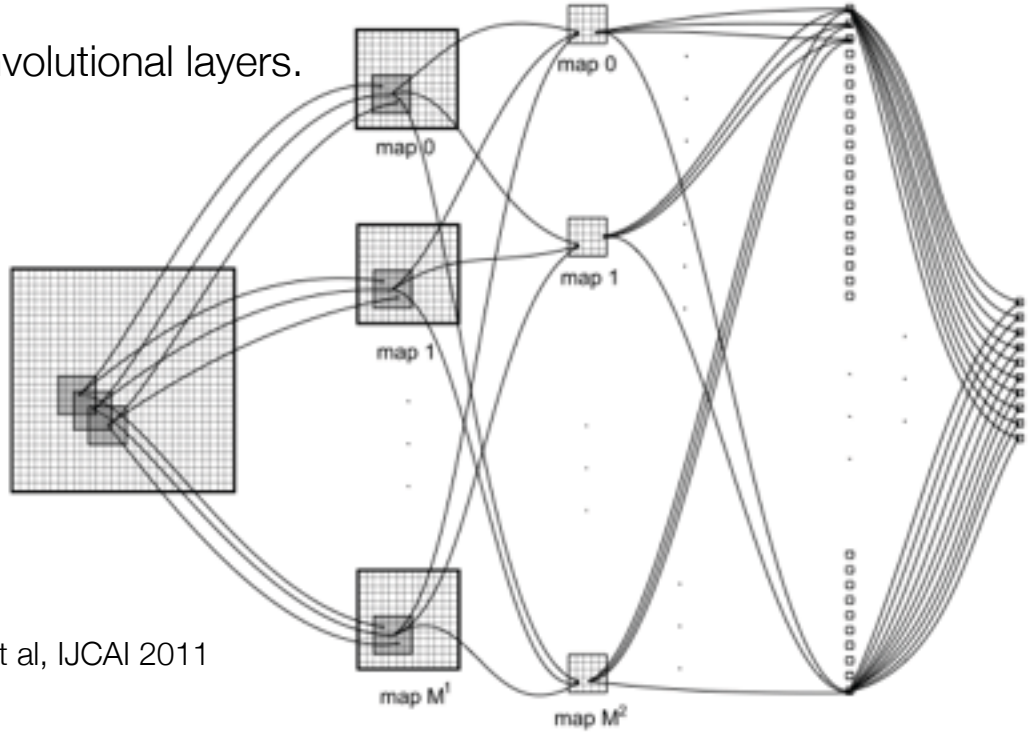
Glorot et al, U.Montreal, AISTATS 2011

Convolutional NN

Strong locality prior allows shared convolutional weights.

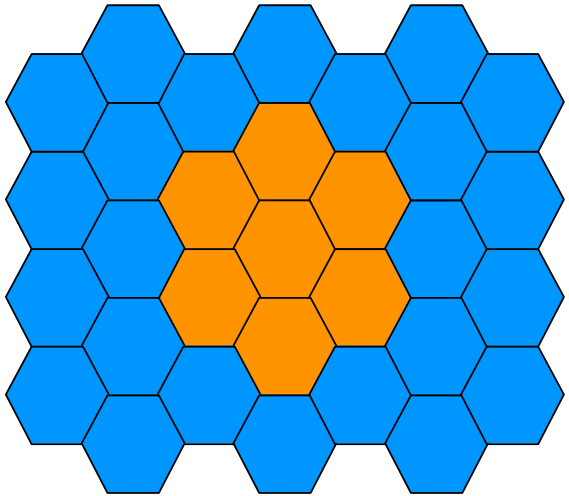
Deep hierarchy is important.

Often pooling (averaging) between convolutional layers.



Ciresan et al, IJCAI 2011

CNN



Local receptive fields need not be square

Depth is effective

GoogLeNet 2014

Social and commerce graphs, and brains

- Complex and irregular, but certainly structured, resulting from biased growth processes.
- Much lower dimension than random graphs, so partitioning is not a catastrophe.
- Wide distributions of vertex degree.

Especially…

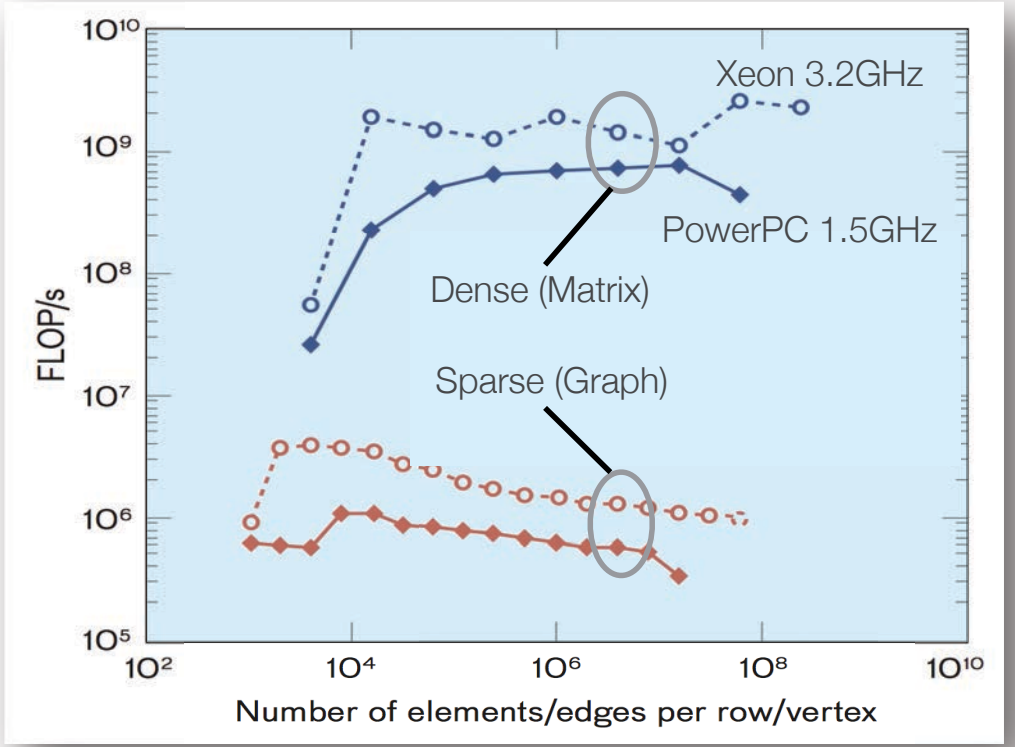
- Small world — linked clusters, skewing degree distribution towards lower degree. Topological distances scale as log V.
- Scale-free — very-small-world, with dominant hub vertices. Inverse power law degree distribution yields almost constant topological distance, scaling as log log V.

Trouble with Sparsity

1000x collapse in performance…

- DRAM lines
- Cache lines
- Vector datapaths

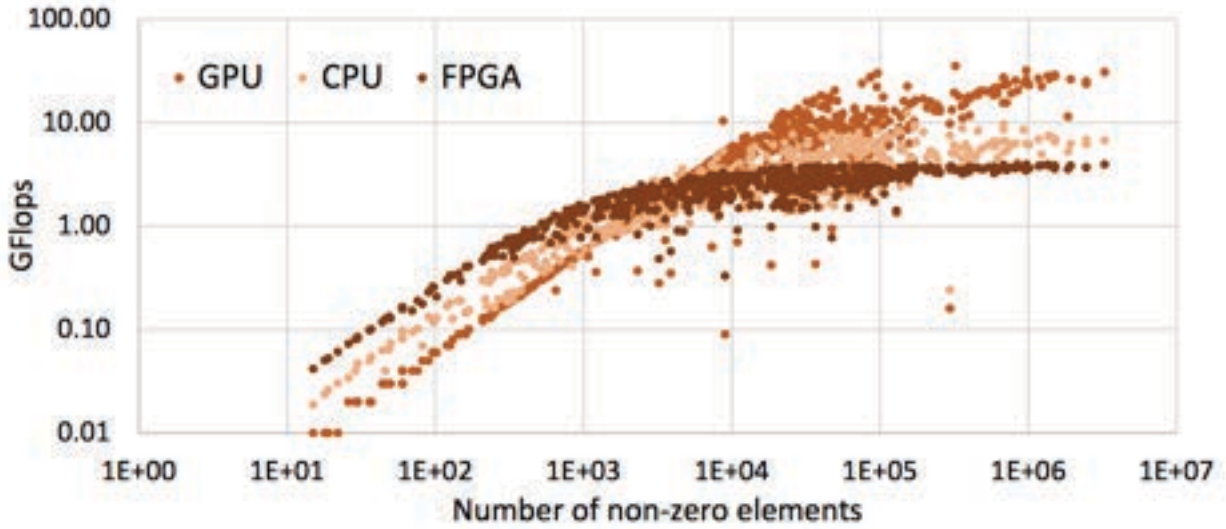
Source: Song et al, Lincoln Lab Journal 2013



Sparse matrix-vector multiply

~2% utilization of multi-Teraflop machines

Too much emphasis on flops, not enough on comms



Fowers et al, Microsoft/U.Florida, FCCM 2014

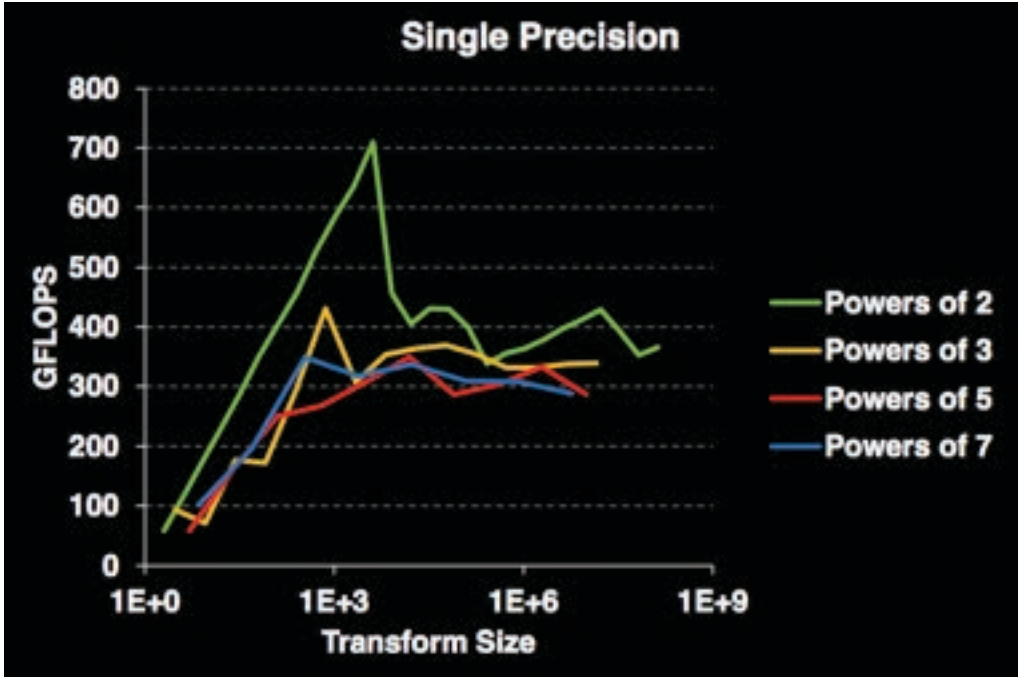
FFTs on vector machines

cuFFT, 1D, batched.

GK110 28nm 4.3 theoretical Tflops.

~10% utilization of FPUs.

Even very regular graphs are challenging for wide SIMD.



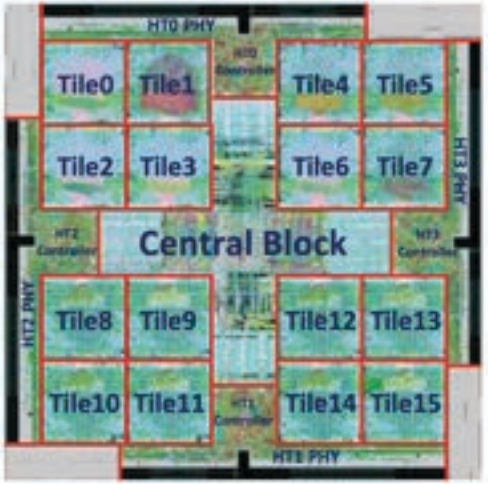
Nvidia CUDA Performance Report v6.5, 2014

The risks of fashion

DNNs, CNNs, RNNs are all the rage this decade, and commercially useful.
But the canon of machine learning is much broader and the field is young.
We know there are big holes — feedback, linear transformation invariance.

For 30 years neuron nonlinearities had to be expensive sigmoids
…now we know cheap rectifiers work better (ReLU)

Only the brave will build a fixed DNN/CNN machine now…



DaDianNao, Chen et al, MICRO 2014

Intelligence is a new workload

CPUs were designed for office apps, then web search.

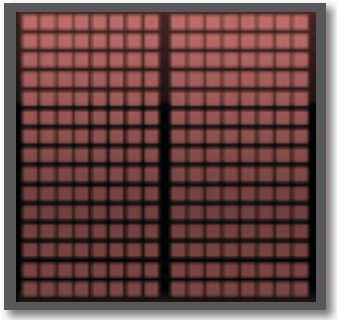
GPUs were designed for graphics, then linear algebra.

IPUs will be designed for machine learning.

CPU

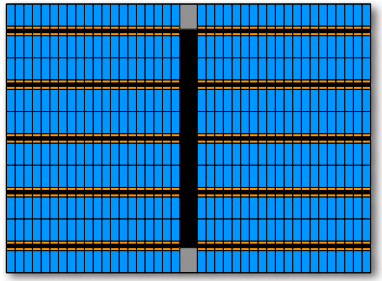
Scalar

GPU



Vector

IPU

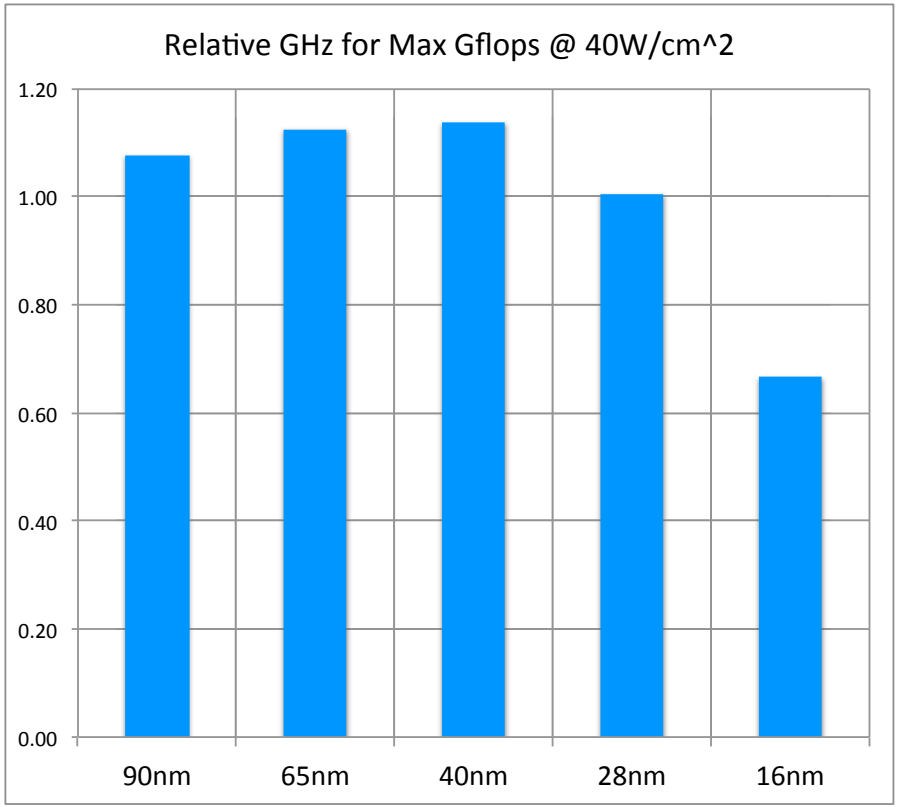


Graph

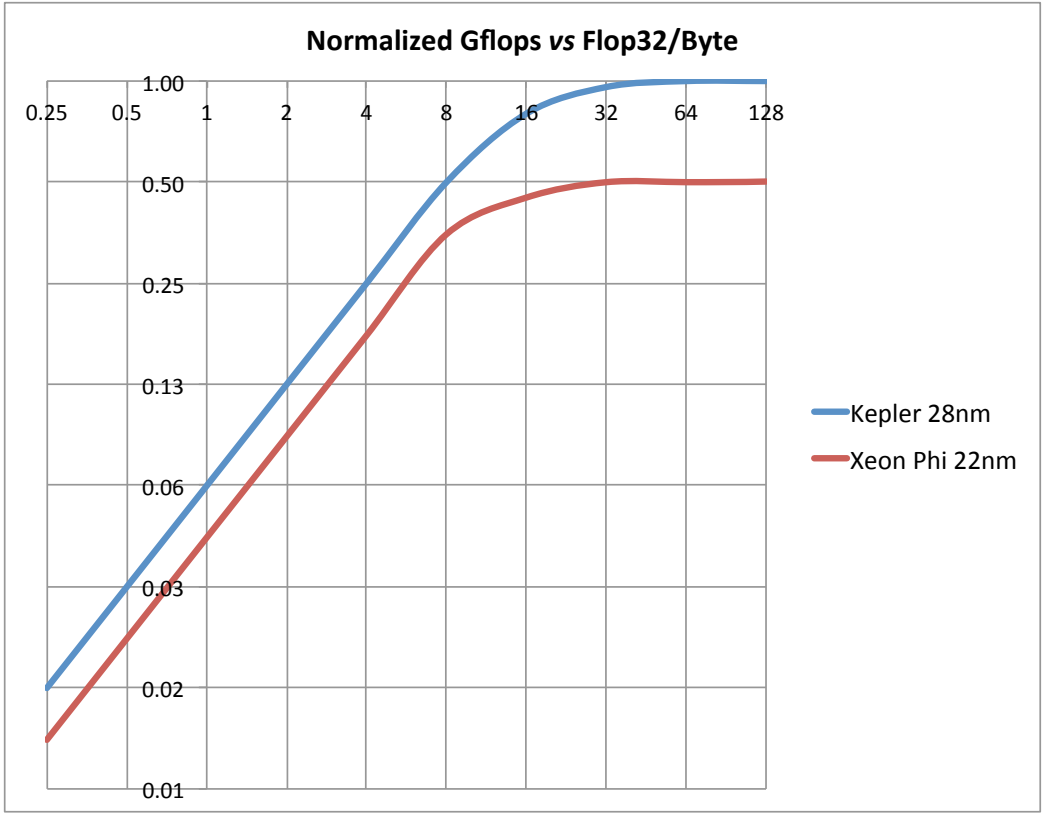
Clock speed inversion

Fixed logic-dominated microarchitecture re-pipelined to maximise Gflops in each technology within 40W/cm^2 limit.

…cramming FPUs is wasting silicon.

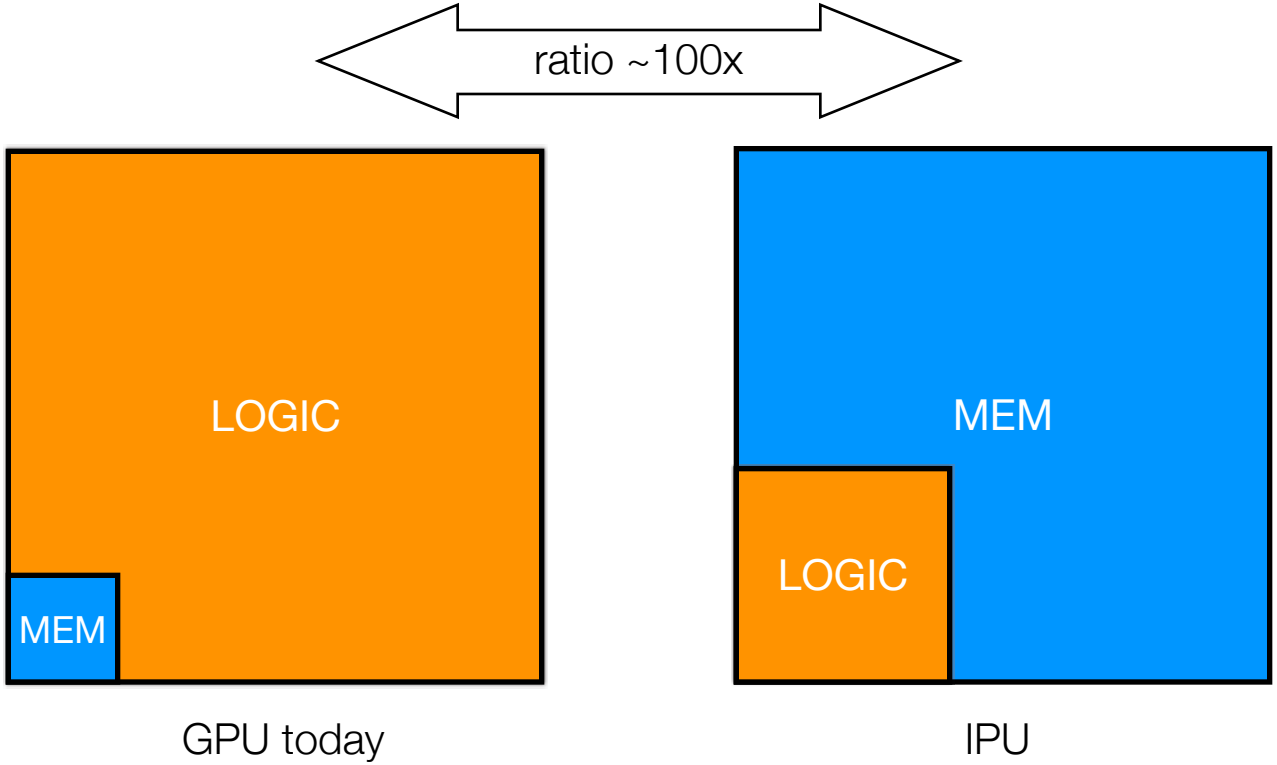


Off-chip memory is increasingly useless during compute



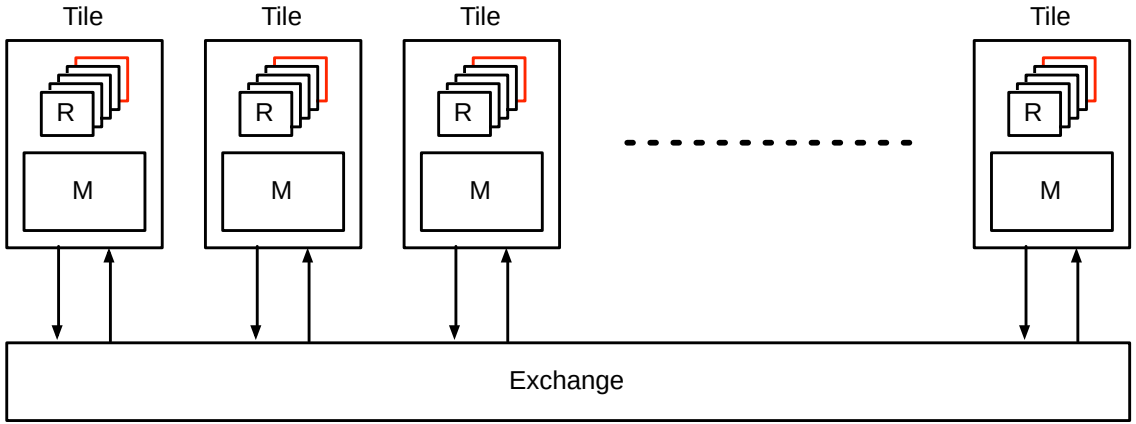
Choi et al, GATech, IPDPS14

Memory should dominate the die



Scalable pure distributed machine

- No central processor or shared state
- Threads hide latency within tiles, not across machine
- Pure logical structure extends across chip, board, rack



Partitioning a random graph

- Generate a random graph…
    - Split vertices into P partitions.
    - Assign each end of each edge to randomly-chosen vertices.
- Probability that both ends are in the same partition is 1/P.
    - Fraction of edges which cross partitions is 1-1/P …99% for P=100

Interesting graphs are far from random…

Nevertheless this is a job for the compiler

So we should treat graph structure as part of the program…?

How quickly will communications really scale?

Mouse ➛ Elephant cortex

Volume ~ Neurons$^{1.33}$

Topological dimension ~4

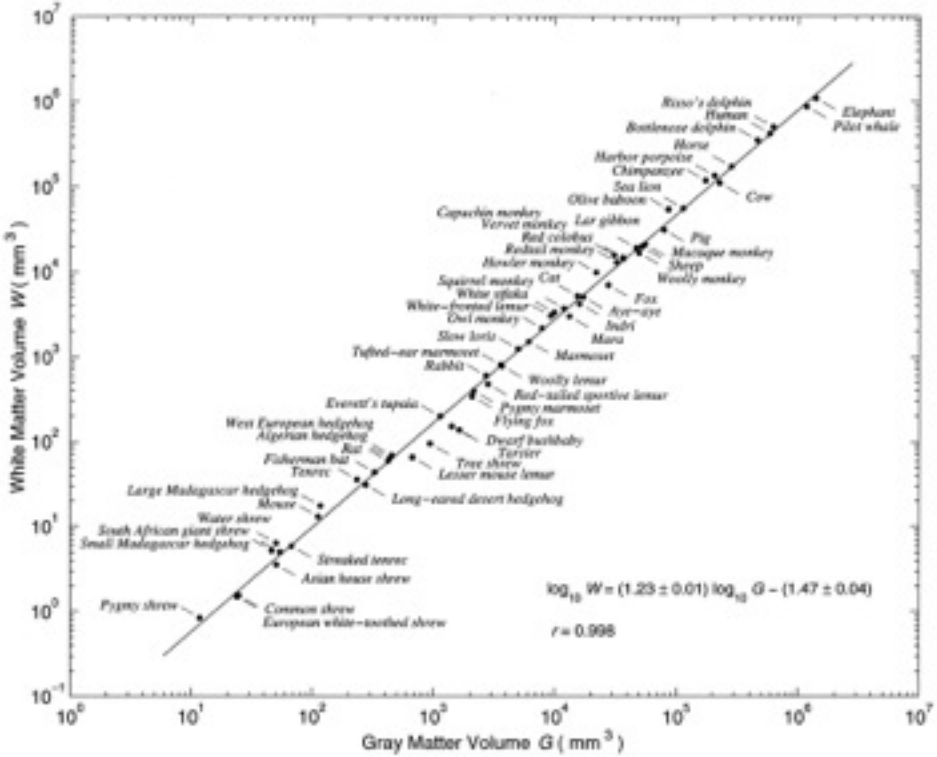
Other complex networks exhibit similar:

- VLSI circuits
- Software call-graphs
- Commerce networks

(see eg. Dan Greenfield's PhD, Cambridge)

So expecting infinite topological dimension,

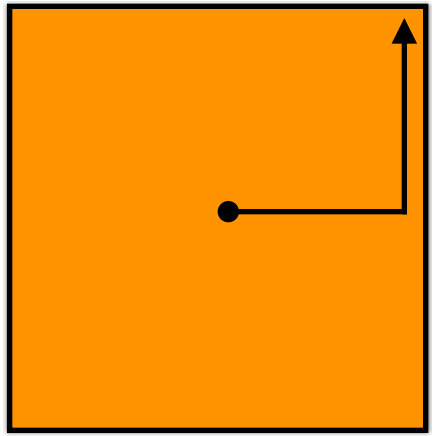
Edges ~ Vertices^2, is certainly pessimistic.



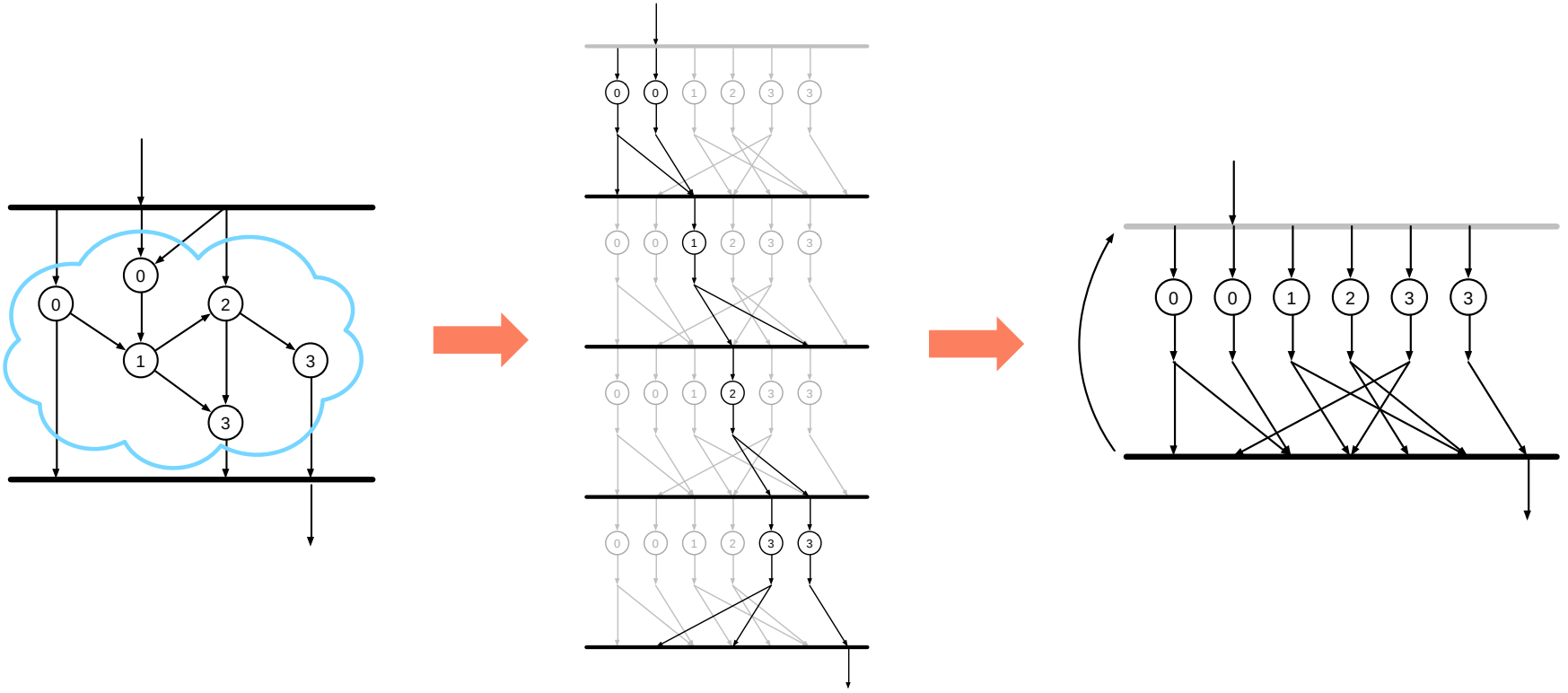
Zhang & Sejnowski, PNAS 2000

How much data can we move on chip?

- 16nm structured wire ~10pJ/B/cm @ 50% bit transition probability.
- Huge 6cm^2 die, power limited at 40W/cm^2 — allocate half the power to raw transmission.
- Allows 1200 simultaneous transmitters of 4GBps across average 2.5cm Manhattan radius.
- Graph structure and good partitioning should somewhat reduce the effective radius.

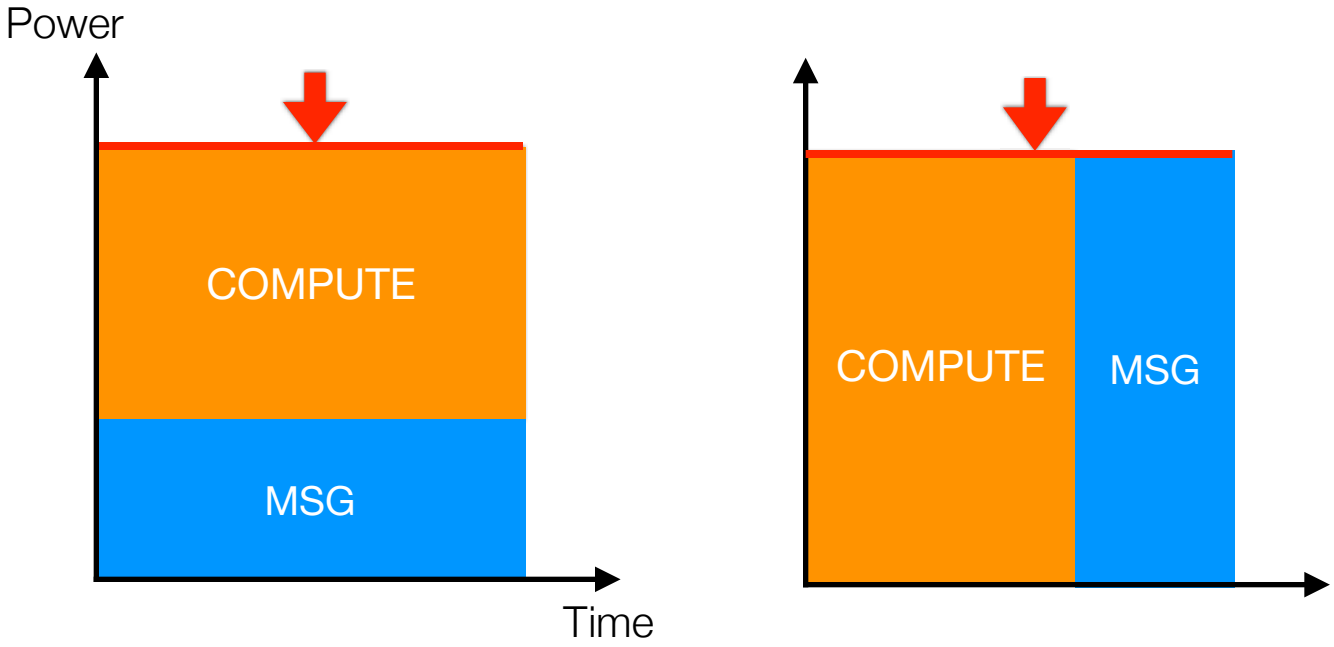


Hazard-free concurrency under BSP



BSP is efficient

Under a constant power limit, load-balanced BSP is energy efficient



Graph-parallel programming abstraction

- A sequential outer control program
    - defines a graph
    - partitions the vertices into acyclic execution sets
    - defines an execution procedure on the sets.
- Vertices within each set may execute in any order which respects the causality defined by graph edges.
- Recurrent edges may connect vertices across control program iterations.
- The control program may read and react to vertex state.
- The vertex programs may read and react to control program state.

Codelets

- Codelets are multi-input, multi-output functions which declare the state and behaviour of vertices.
- Any mix of codelets may execute simultaneously and asynchronously on any number of vertices of mixed degree; input dimensions adapt to vertex degree.
- Codelets execute atomically — read inputs, update internal state and outputs, terminate.
- Codelets inter-communicate only by posting output values over graph edges. There is no shared writeable state.
- Codelets can read control program values.

Higher language levels

Two schools…

- Native graph frameworks
    - Google Pregel, 2010
    - GraphLab, Giraph, and 20 others within 2 years
- Structure-specific application frameworks
    - Torch
    - Theano
    - Caffe

Finally… what about Moore?

- Scaling continues
- Cost saving continues
- 2x density every 2.5 years

…and cost scaling continues too