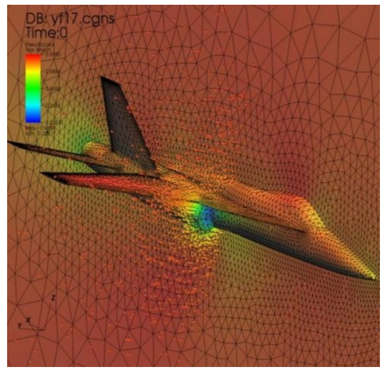
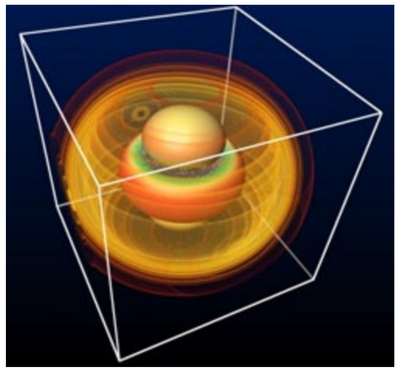
ADVANCED PARALLEL COMPUTING 2017 LECTURE 11 - CONSTRAINTS, TRENDS & DEEP LEARNING

Holger Fröning
holger.froening@ziti.uni-heidelberg.de
Institute of Computer Engineering
Ruprecht-Karls University of Heidelberg

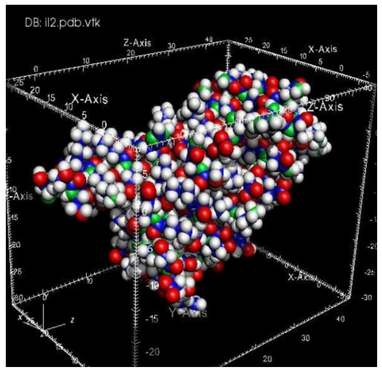
APPLICATIONS & ARCHITECTURES



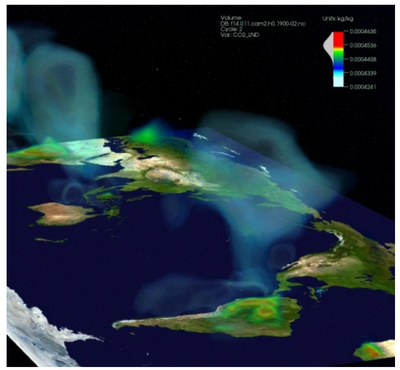
CFD



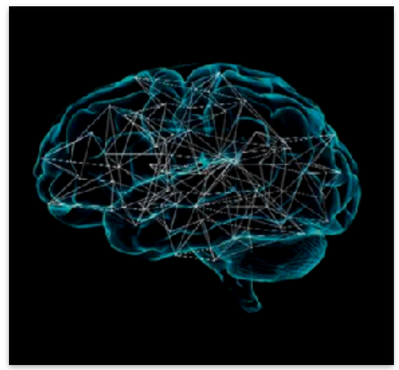
Cosmology



Molecules



Weather&Climate



AI

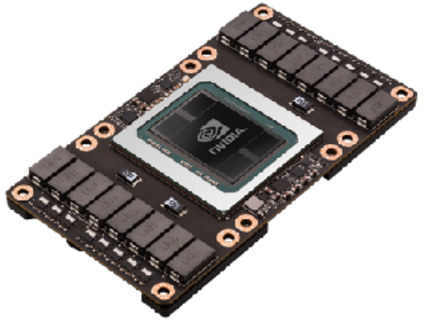
Tera-FLOP/s

September 1997

15 years

10 Tera-FLOP/s

November 2016

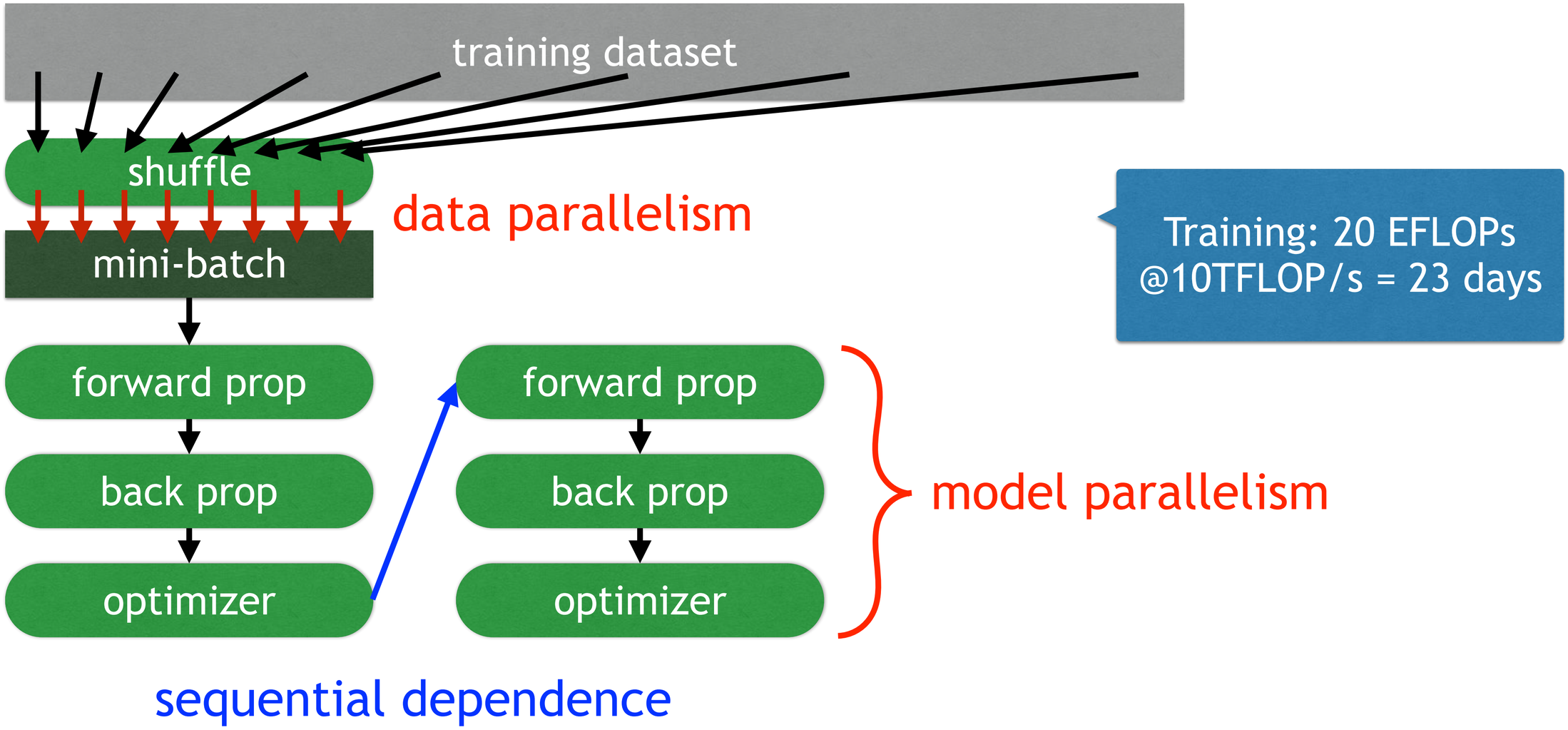


Exa-FLOP/s

2018-2012

?

TRAINING OF DEEP NEURAL NETWORKS



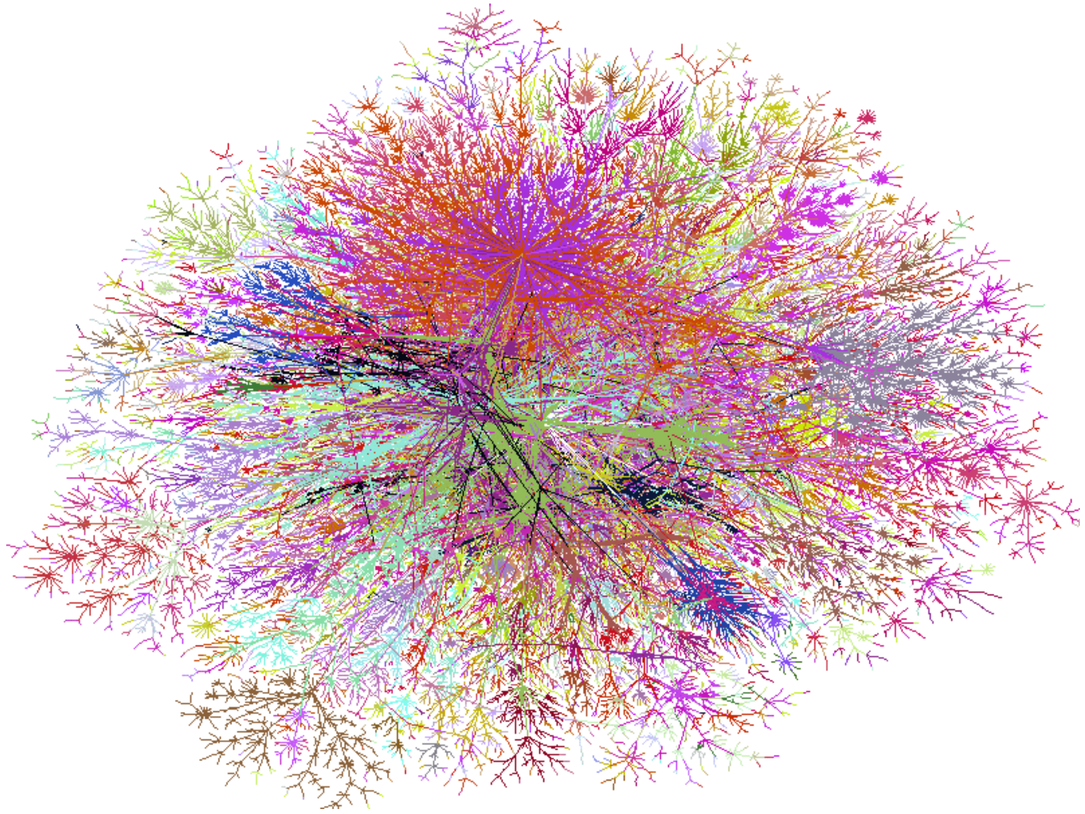
GRAPH COMPUTATIONS

Graph computation = algorithm + input data

- Program = algorithm + data structure, Niklaus Wirth
- Shortest-path
- Strongly-connected components
- Page rank

Problems

- Data-driven computation
- Unstructured problem
- Poor locality
- Low computational intensity (F/B)



Internet network, www.math.cmu.edu

CONSTRAINTS & TRENDS

CMOS POWER MODEL

$$P = afCV^2 + \cancel{VI_{leakage}}$$

$$C = E_r * E_0 * \frac{A}{d}; A = W * L \quad \Rightarrow C \rightarrow \frac{C}{\alpha}$$

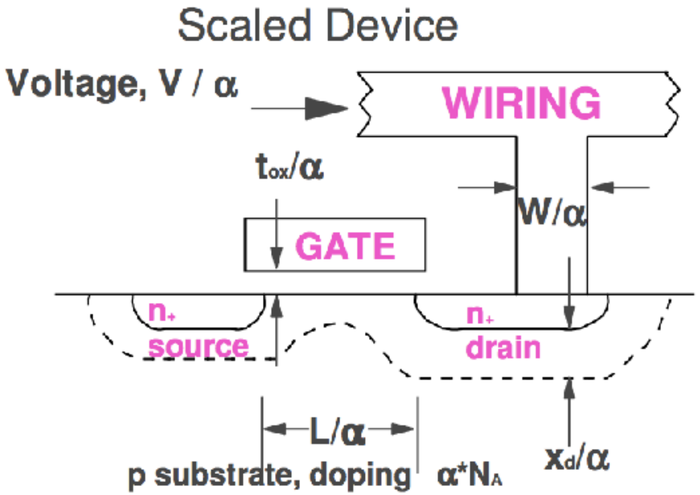
$$P_{new} = (a * \alpha^2)(f)(\frac{C}{\alpha})(\frac{V^2}{\alpha^2}) = \frac{P_{old}}{\alpha}$$

Or, for $P_{new} = P_{old}$: $f \rightarrow f * \alpha$

Problem: V does not scale any more

End of Dennard scaling

$$P_{new} = P_{old} * \alpha$$

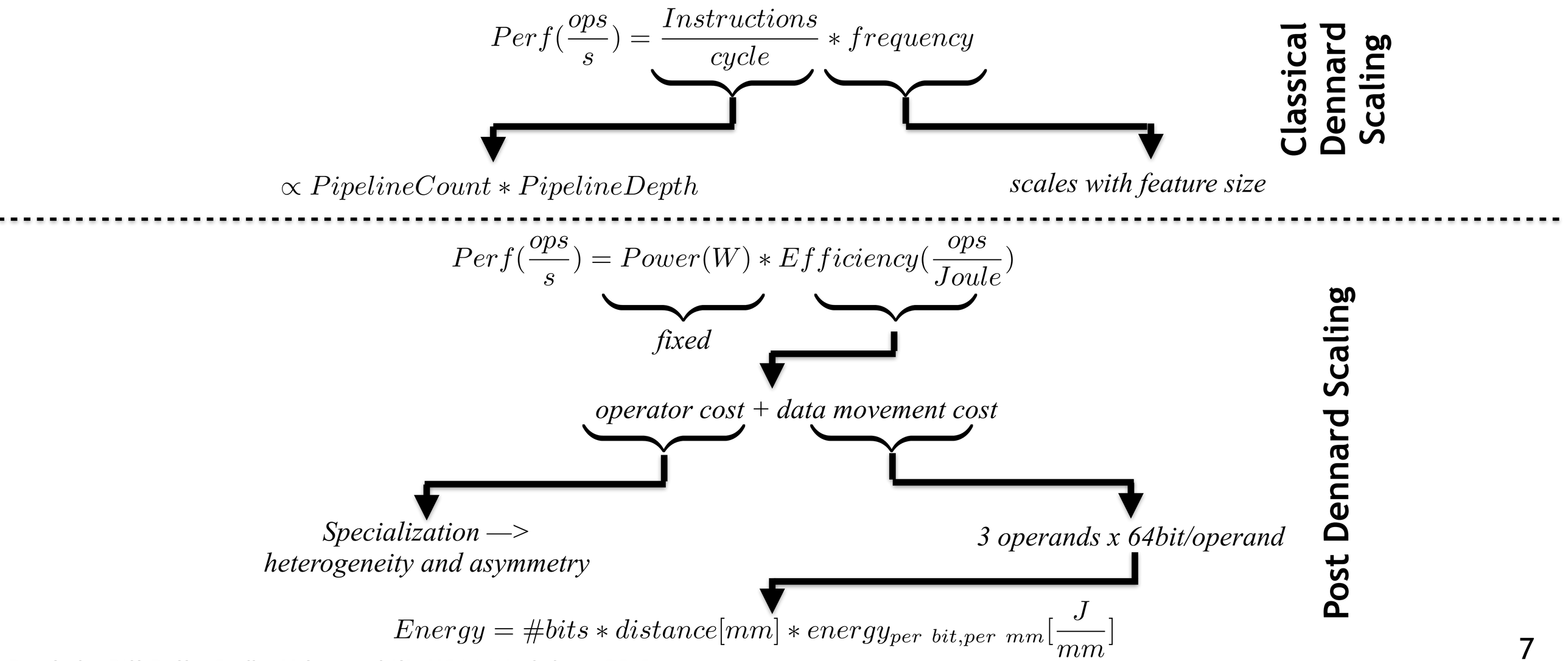


Classic Dennard
Oxide: t_{ox} /α
Wire width: W/α
Gate width: L/α
Diffusion: x_d/α
Substrate: α * NA
~~Voltage: V/α~~
Current: I/α

R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of Ion-implanted MOSFET's with Very Small Physical Dimensions," in Proceedings of the IEEE Journal of Solid-State Circuits, vol. 9, no.5, Oct. 1974.

Pradip Bose, IBM Research, 2014

POST-DENNARD PERFORMANCE SCALING

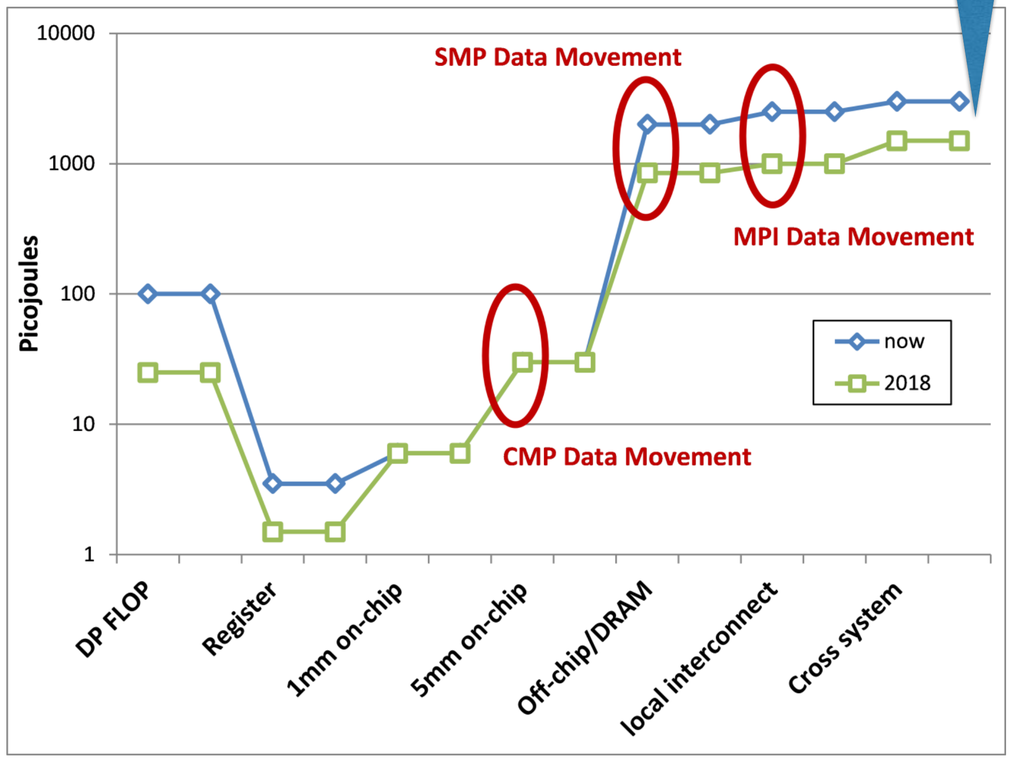


Partly by Bill Dally, Sudha Yalamanchili, UCAA Workshop, 2012

POST-DENNARD II: ANOTHER FUNDAMENTAL TRANSIT

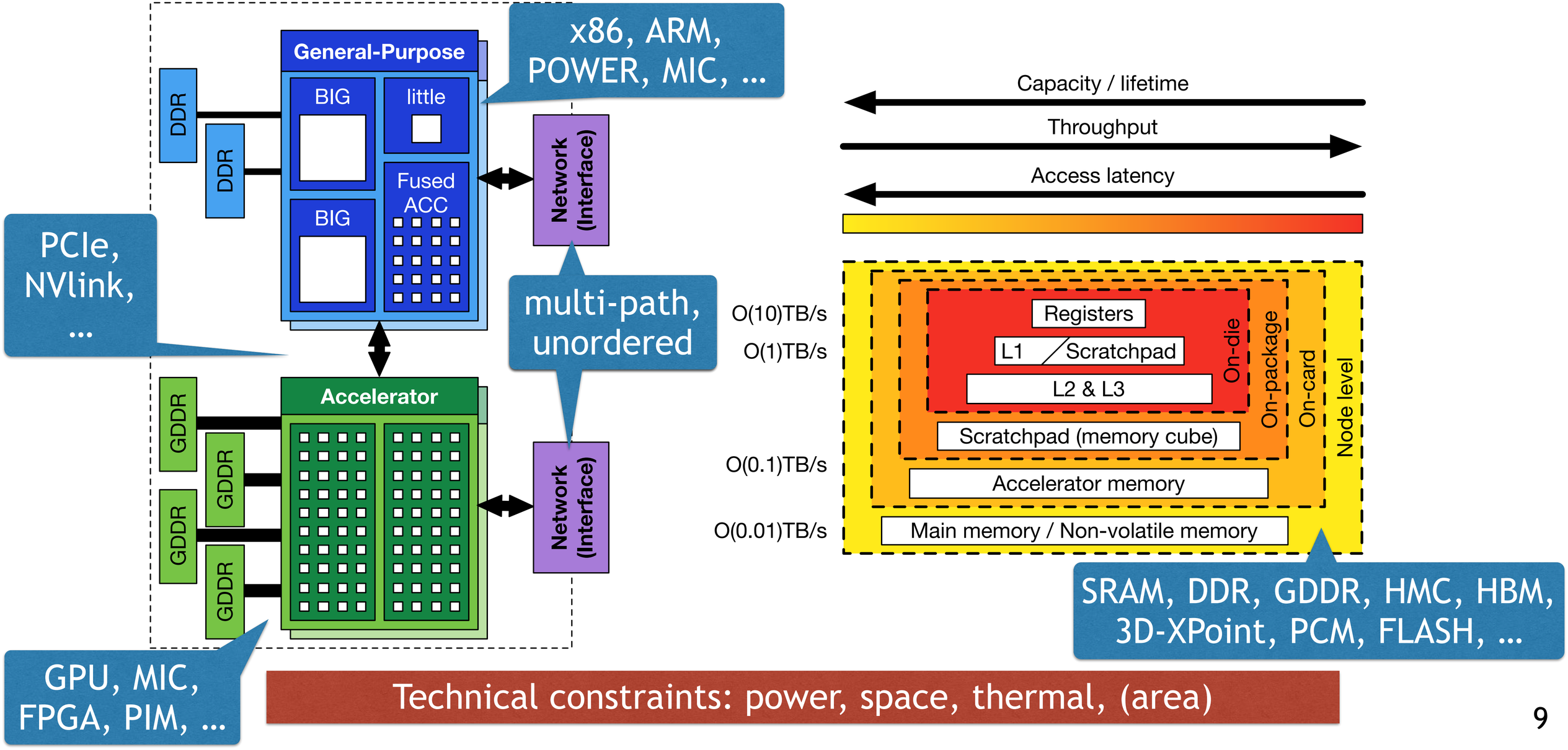
Validated by us: CML serializer @ 10Gbps: ~30pJ/bit = 1920pJ/DP

- Communication-centric systems
  - Data movements matter, computations are for “free”
  - Energy optimization = Locality optimization
  - Implications will percolate up through the complete compute stack
- Need to understand massively-parallel communication



US DOE, Scientific Grand Challenges: Architectures and Technology for Extreme Scale Computing, San Diego, CA, 2009.

FUTURE ARCHITECTURE TEMPLATE



Technical constraints: power, space, thermal, (area)

REMINDER: BULK-SYNCHRONOUS PARALLEL

In 1990, Valiant already described GPU computing pretty well

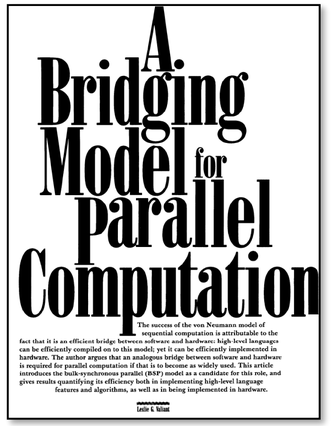
Superstep

- Compute, communicate, synchronize

Parallel slackness: # of virtual processors v, physical processors p

- v = 1: not viable
- v = p: unpromising wrt optimality
- v >> p: leverage slack to schedule and pipeline computation and communication efficiently

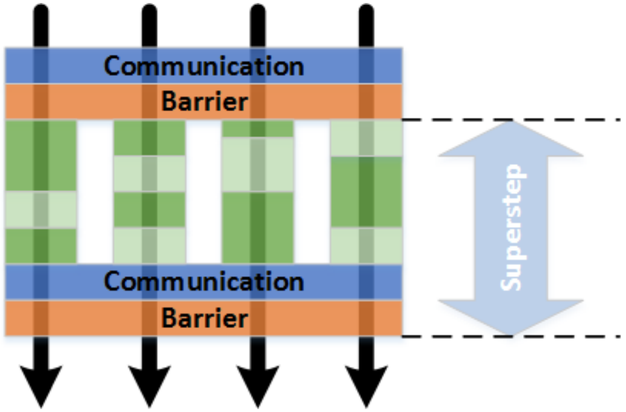
Extremely scalable, bad for unstructured parallelism

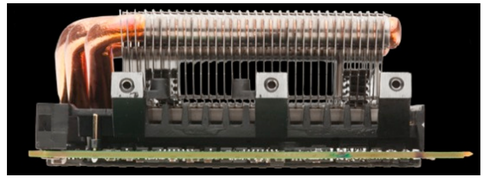

A Bridging Model for parallel Computation

The success of the von Neumann model of sequential computation is attributable to the fact that it is an efficient bridge between software and hardware: high-level languages can be efficiently compiled on to this model; yet it can be efficiently implemented in hardware. The author argues that an analogous bridge between software and hardware is required for parallel computation if that is to become as widely used. This article introduces the bulk-synchronous parallel (BSP) model as a candidate for this role, and gives results quantifying its efficiency both in implementing high-level language features and algorithms, as well as in being implemented in hardware.

Leslie G. Valiant

Leslie G. Valiant, A bridging model for parallel computation, Communications of the ACM, Volume 33 Issue 8, Aug. 1990





REMINDER: VECTOR ISAS

Compact: single instruction defines N operations

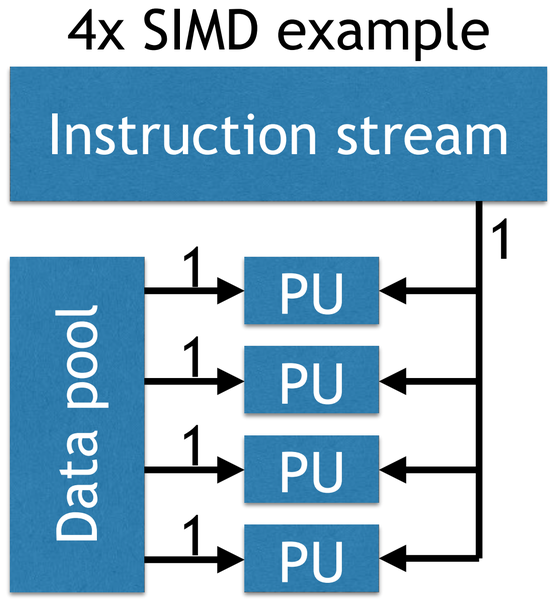
- Amortizes the cost of instruction fetch/decode/issue
- Also reduces the frequency of branches

Parallel: N operations are (data) parallel

- No dependencies
- No need for complex hardware to detect parallelism (similar to VLIW)
- Can execute in parallel assuming N parallel datapaths

Expressive: memory operations describe patterns

- Continuous or regular memory access pattern
- Can prefetch or accelerate using wide/multi-banked memory
- Can amortize high latency for 1st element over large sequential pattern



OUR VIEW OF A GPU

Software view: a programmable many-core scalar architecture

- SIMT: single instruction, multiple threads
- Huge amount of scalar threads to exploit parallel slackness, operates in lock-step

IT'S A PERFECT INCARNATION OF THE BSP MODEL

Hardware view: a programmable multi-core vector architecture

- SIMD: single instruction, multiple data
- Illusion of scalar threads: hardware packs them into compound units

IT'S A VECTOR ARCHITECTURE THAT HIDES ITS VECTOR UNITS

TRANSITIONING TO MULTI-GPU IS FUNDAMENTAL

Transition from SMP to NUMA

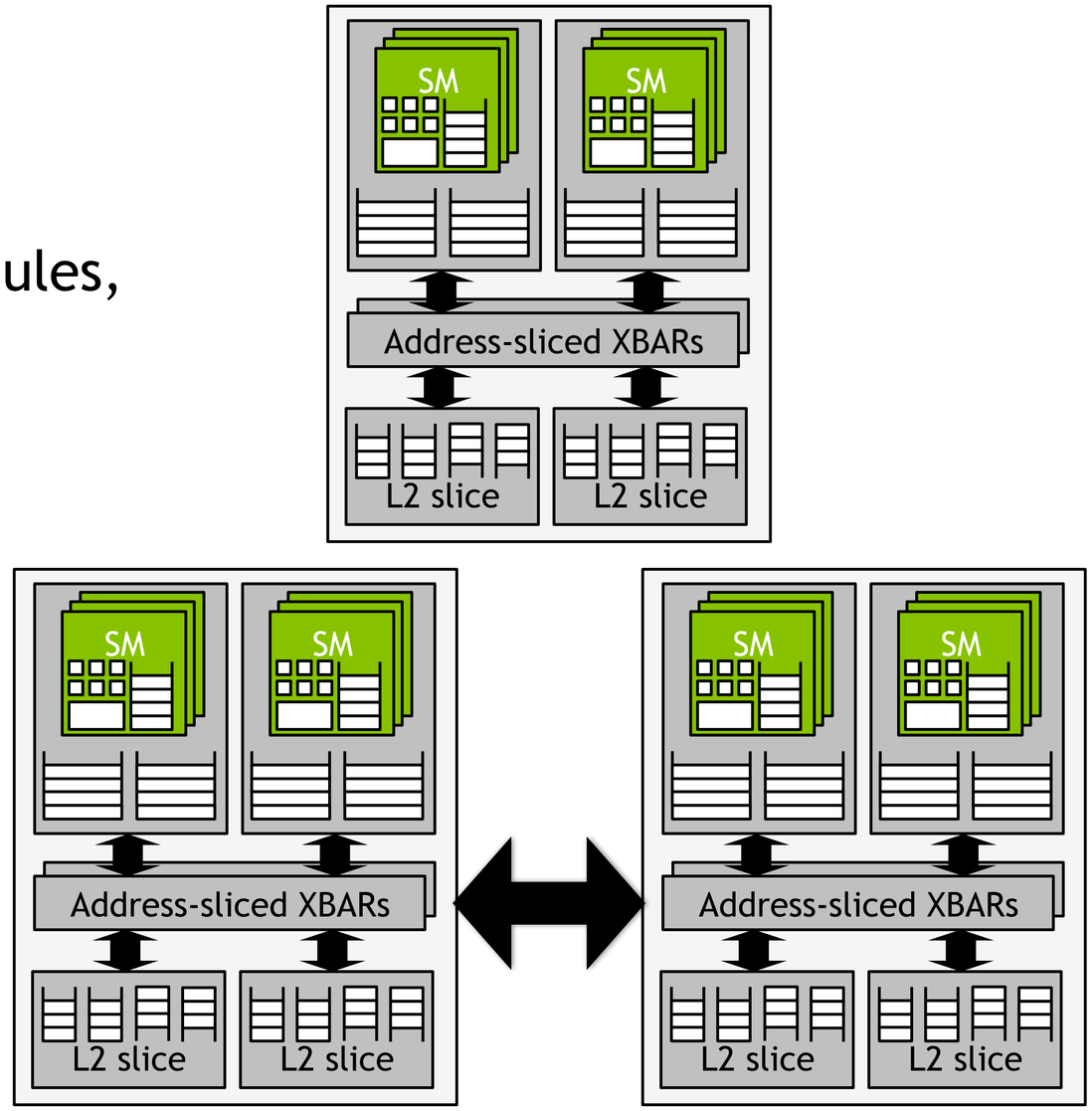
- Reasons: multi-GPU systems, multi-chip modules, heterogeneous memory, tiled layout

Beauty of BSP is lost

- Kernel launch orchestration
- Data movement operations
- Naming a physical resource is disgusting

Consistency models lack NUMA support

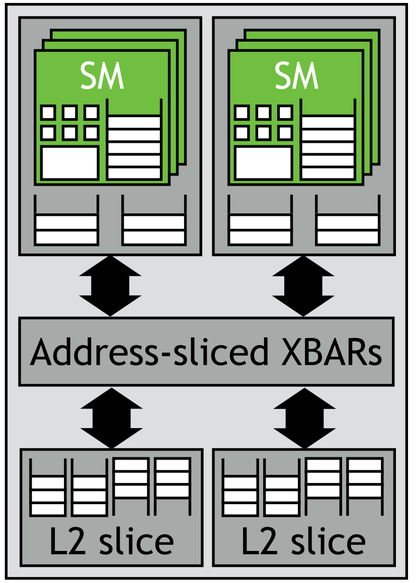
- Concentric memory scopes are broken
- Memory fences, too



REVIEW OF OTHER MANY-CORE PROCESSORS

In principle: no difference between GPUs & FPGAs

	GPU	FPGA
High concurrency at reduced frequency	y	y
Flat memory hierarchy	y	y
Scratch-pad memory	y	y
Programming using data-parallel kernels	y	partly
Latency hiding techniques (BSP-like)	**y**	**n**
Reprogrammable	**n**	**y**

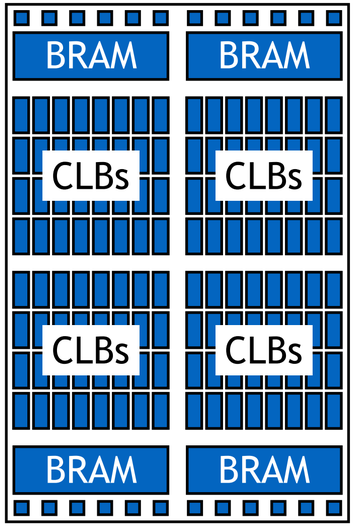


GPU

Architecturally similar, execution model fundamentally different

- GPUs: absolute performance
- FPGAs: absolute power consumption
- FLOPS/Joule depending on various aspects

FPGA = the only true general-purpose processor?

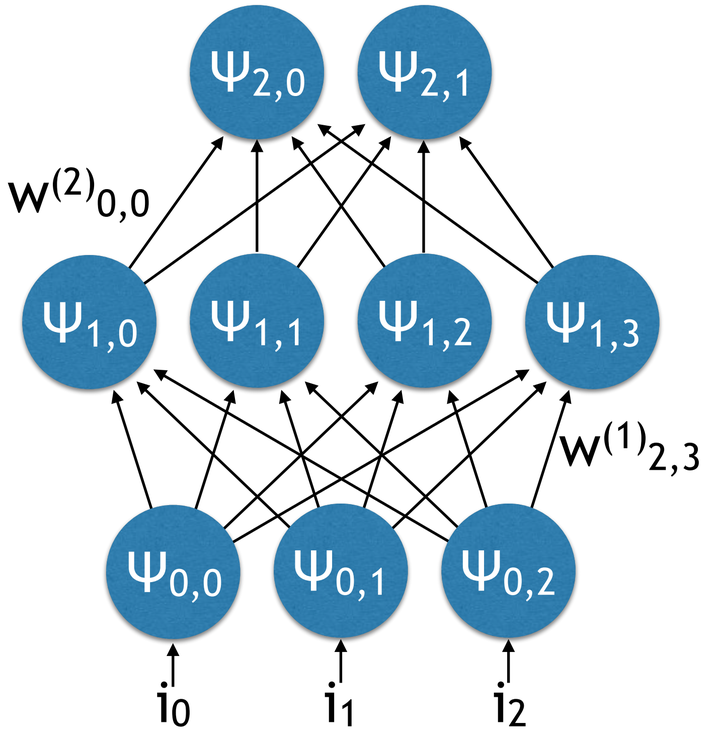


FPGA

DEEP LEARNING

MULTI-LAYER PERCEPTRON

MNIST handwritten database



$$\Psi = f(W \cdot x + b)$$

$$\psi_i = f(\sum_j (w_{j,i} \cdot x_j) + b_i)$$

f: non-linear function (sigmoid, reLU, …)
Ψ: neuron vector
W: weight matrix
x: input vector
b: bias vector

Foundation: “Fully-connected layer” => matrix-vector operations

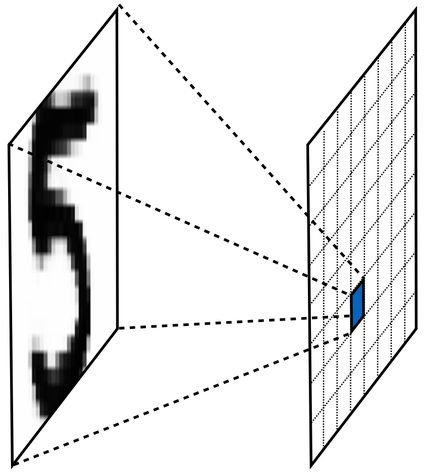
For parallel training: matrix-matrix operations (x becomes matrix)

CONVOLUTIONAL NETWORKS

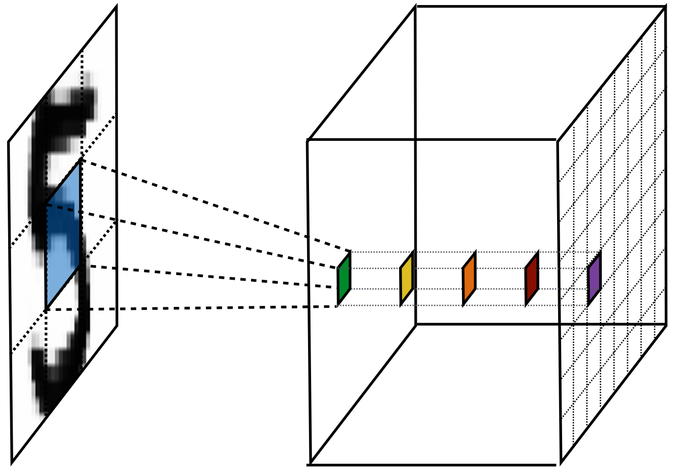
Convolutional layer

- Receptive field: spatially local correlation (patches)
- Shared weights: as each filter is applied to all patches of the input
- 3D layers: “depth” of one layer is the number of filters (kernels) learned

=> Even higher computational intensity

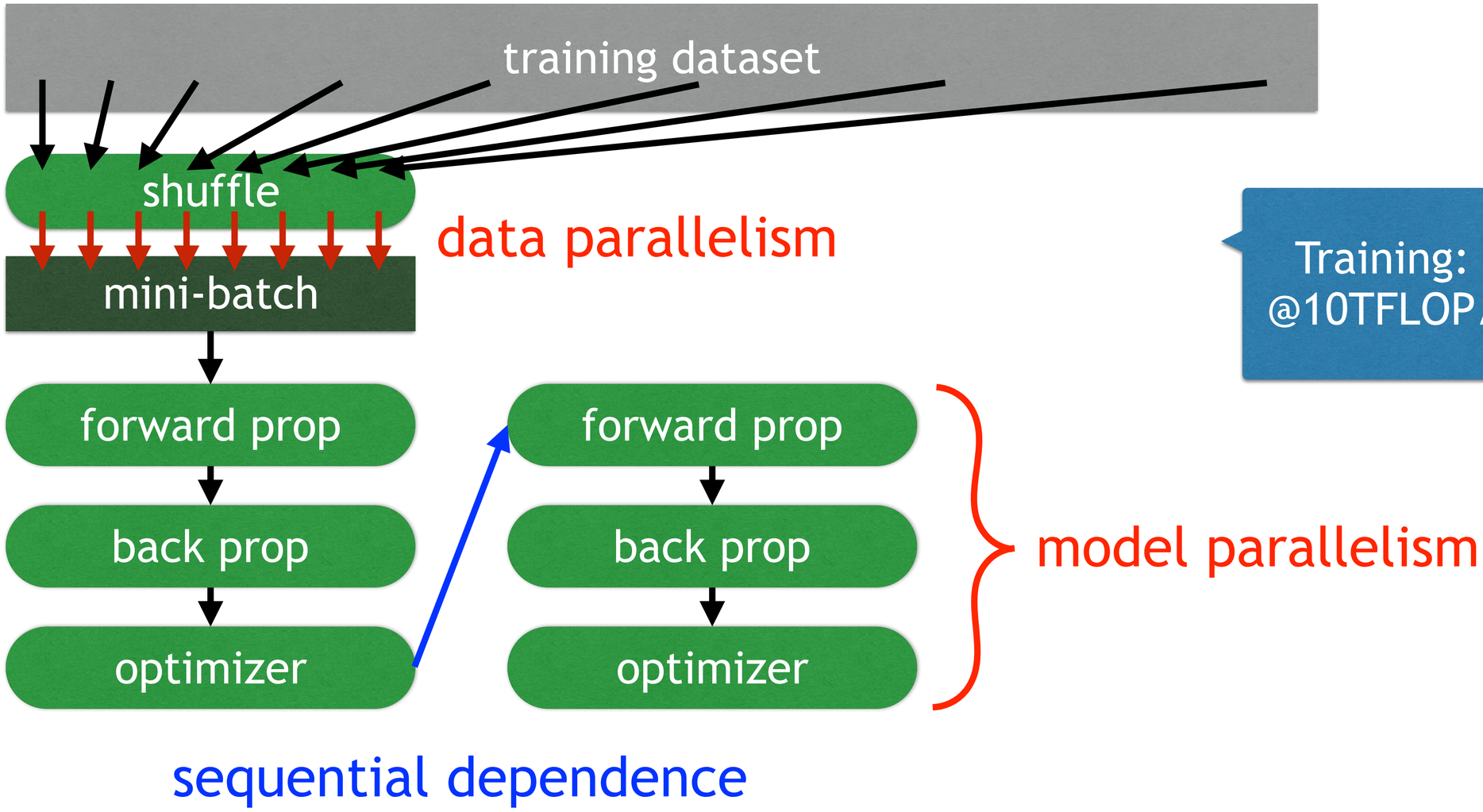


Fully-connected layer



Convolutional layer

TRAINING



Training: 20 EFLOPs
@10TFLOP/s = 23 days

BACKWARD PATH

Error function E

- i^{th} training example
- $t^{(i)}$: true answer
- $y^{(i)}$: result from NN

$$E = \frac{1}{2} * \sum_i (t^{(i)} - y^{(1)})^2$$

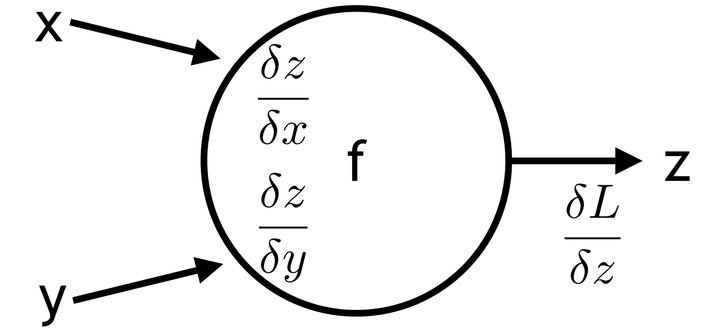
Gradient descent

- Multiply steepness with learning rate
- Stochastic gradient descent: use randomization and noise to find global minimum

Backpropagation

- Propagate error backwards through the network: partial derivatives & chain rule
- Usually an operation based on a Jacobian (J) matrix multiplication

$$\frac{\delta L}{\delta x} = \frac{\delta L}{\delta z} * \frac{\delta z}{\delta x}$$

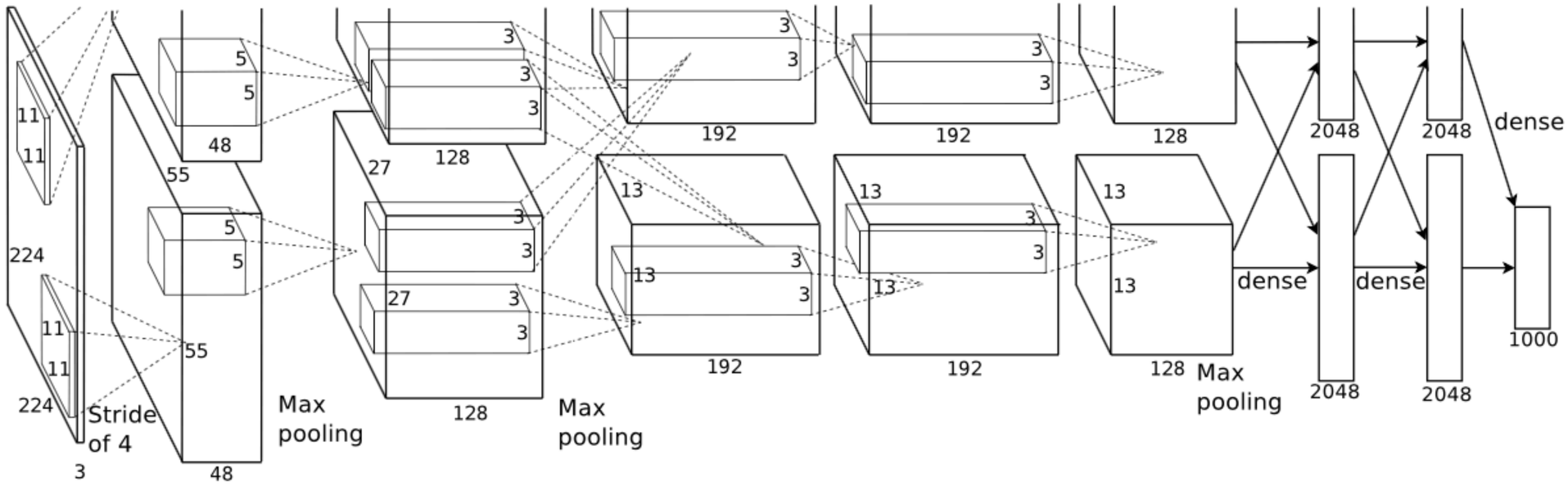


$$\frac{\delta L}{\delta y} = \frac{\delta L}{\delta z} * \frac{\delta z}{\delta y}$$

$$J_{i,j} = \frac{\delta f_i}{\delta x_j}$$

COMMON NEURAL NETWORKS TODAY

Net	Parameters	Neurons	Layers	FLOPs
AlexNet	60M	0.7M	8	0.72G
VGG-16	138M	14M	16	15.3G
GoogLeNet	6.8M	4.5M	22	1.5G
ResNet-152	26M	20M	152	11.3G



Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12)

HANDS-ON: TRAINING PERFORMANCE

Workload: AlexNet Training

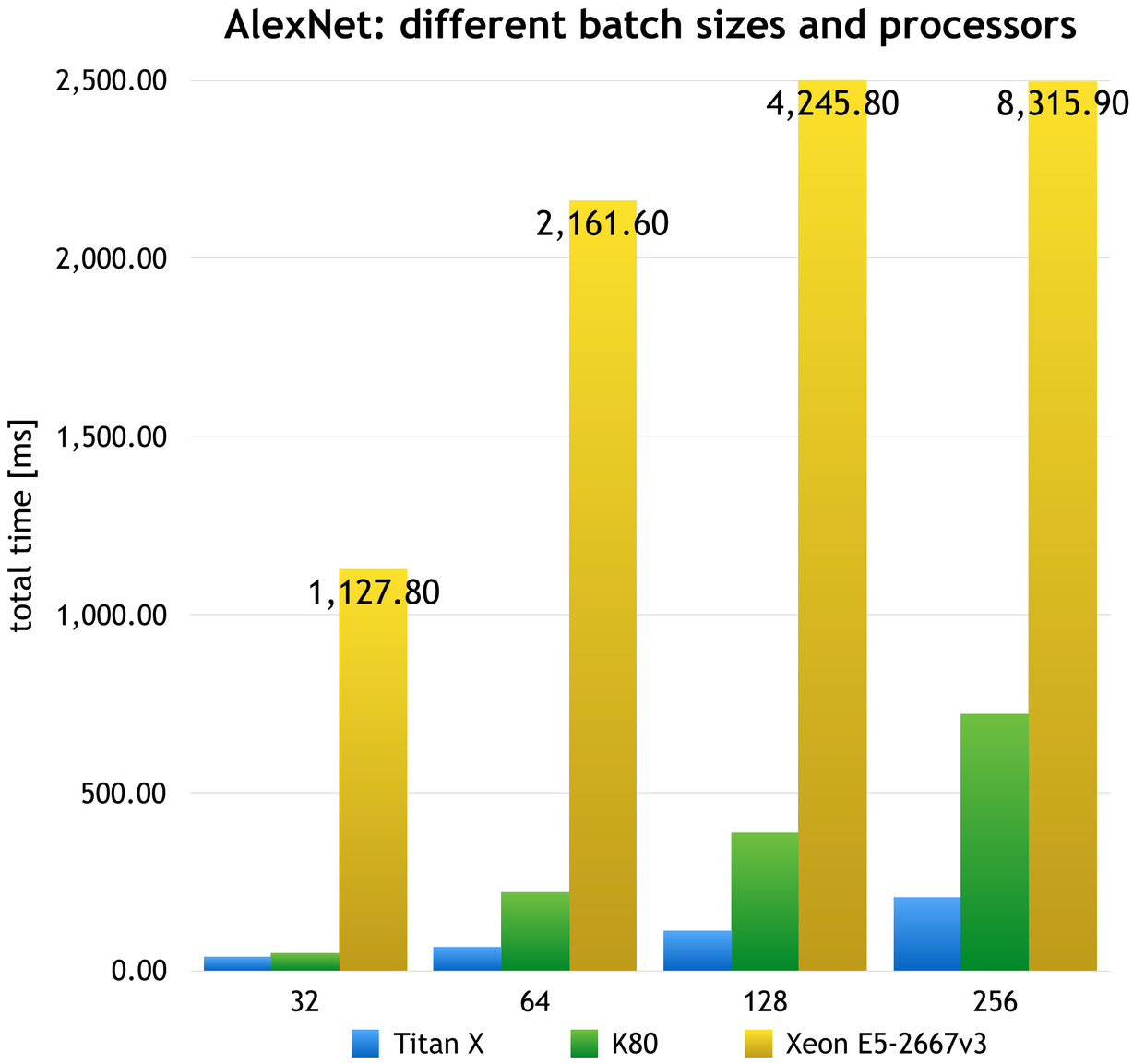
- Forward+Backward Pfad

Processors

- Titan-X (Pascal-class)
- K80 (Kepler-class)
- Xeon-E5 (Haswell-class)

Different batch sizes

- (+) Increase of computational intensity
- (+) Reduce variance of stochastic gradient updates
- (-) Increase memory requirement
- (-) Convergence problem if too large



PARALLEL TRAINING: COMMUNICATION

Here: synchronous data parallelism

- AlexNet training
- NVIDIA Kepler K80 GPUs

Asynchronous data parallelism

- Overlap communicate with next compute
- (+) random noise helps convergence
- (-) stale gradient problem

	2 GPU	4 GPU	8 GPU
Effective batch size	64	32	16
Communication/batch [ms]	61	183	427
Computation/batch [ms]	760	488	355
Computation/communication	12.48	2.66	0.83
Speedup	1.6	1.97	1.69

Pitfall: plain DNN training is not scalable

Benjamin Klenk, Communication Architectures for Scalable GPU-centric Computing Systems, PhD thesis, Heidelberg University, 2017 (pending).

PARALLEL TRAINING: COMMUNICATION

	Data parallelism	Model parallelism		
Net	4-16 GPUs	4	16	64
AlexNet	240MB	179MB (22MB)	45MB (6MB)	11MB (1MB)
VGG-16	552MB	3,584MB (224MB)	896MB (56MB)	224MB (14MB)
GoogLeNet	27MB	1,152MB (52MB)	288MB (13MB)	72MB (3MB)
ResNet-152	104MB	5,120MB (34MB)	1,280MB (8MB)	320MB (2MB)

Communication volume per GPU

- Number in brackets: average volume per layer and GPU
- Batch size of 256

Model parallelism can reduce data volume

Benjamin Klenk, Communication Architectures for Scalable GPU-centric Computing Systems, PhD thesis, Heidelberg University, 2017 (pending).

REMINDER: SYSTOLIC ARRAYS

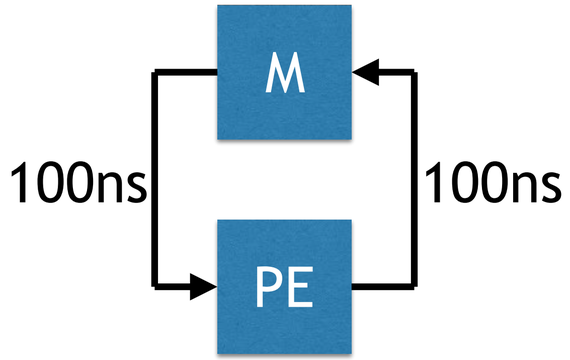
Basic principle: replace single PE with PE array

- High throughput without increasing memory bandwidth requirement

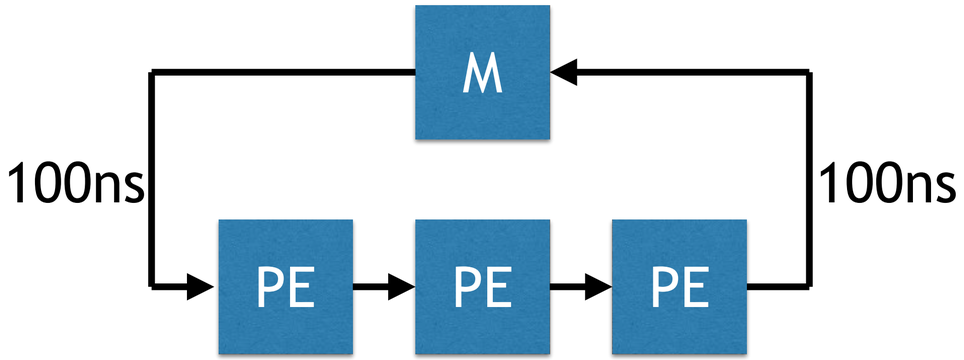
Differences to pipelined architecture

- Array structure can be non-linear (e.g., hexagonal)
- Connections between PEs can be bi-directional
- PEs may have local instruction and data memory
- PEs often more complex than a pipeline stage

Commercial: iWarp (linear array), produced by Intel



1op/200ns = 50MOPS



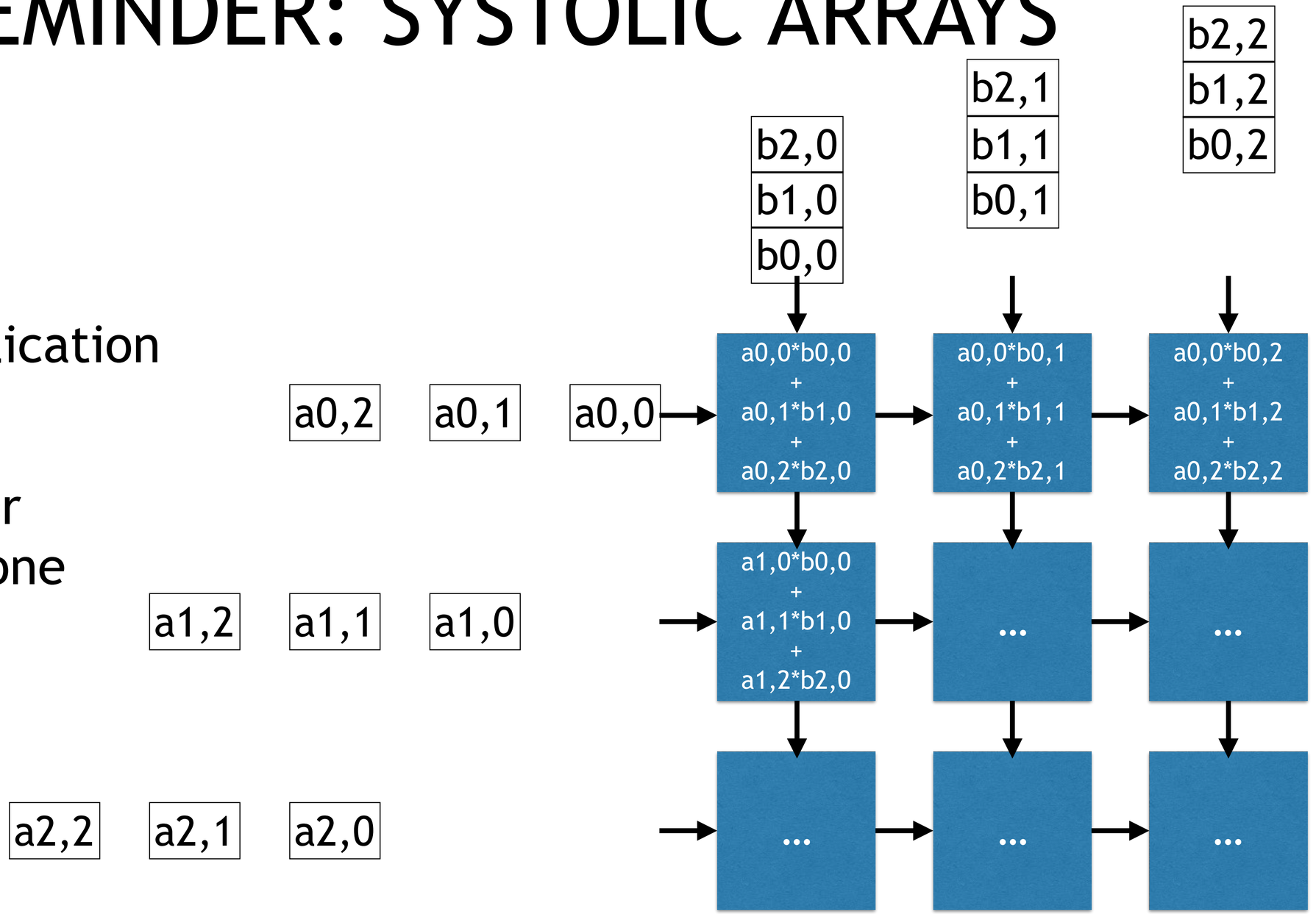
3op/200ns = 150MOPS

REMINDER: SYSTOLIC ARRAYS

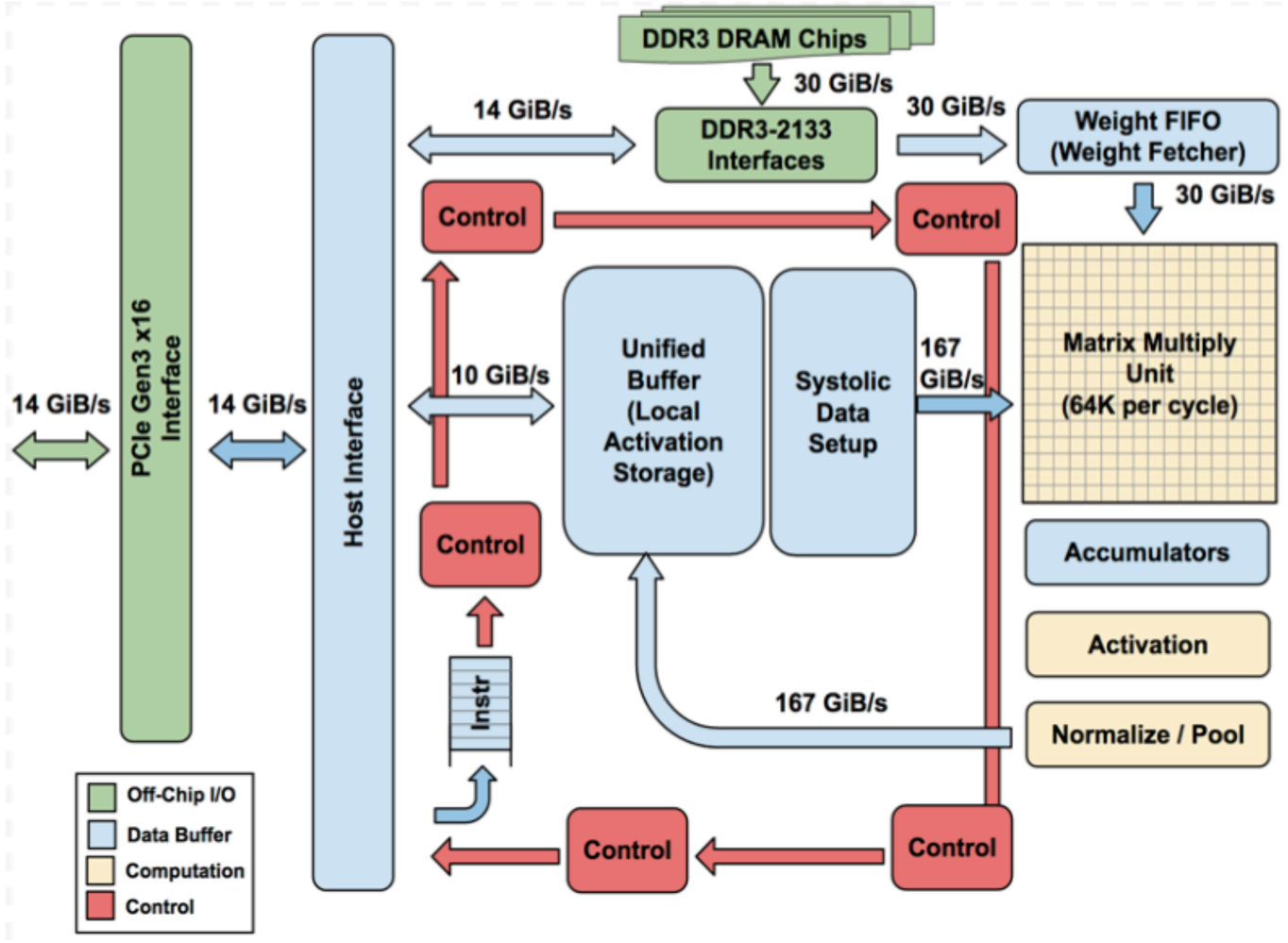
Matrix multiplication

C = A * B

Each processor accumulates one element of C



DNN ACCELERATORS FOR INFERENCE (GOOGLE TPU)



Norman P. Jouppi et al., In-Datacenter Performance Analysis of a Tensor Processing Unit, 44th Annual International Symposium on Computer Architecture, ISCA'17

SUMMARY

THIS COURSE

Communication architectures

Synchronization: locks & barriers

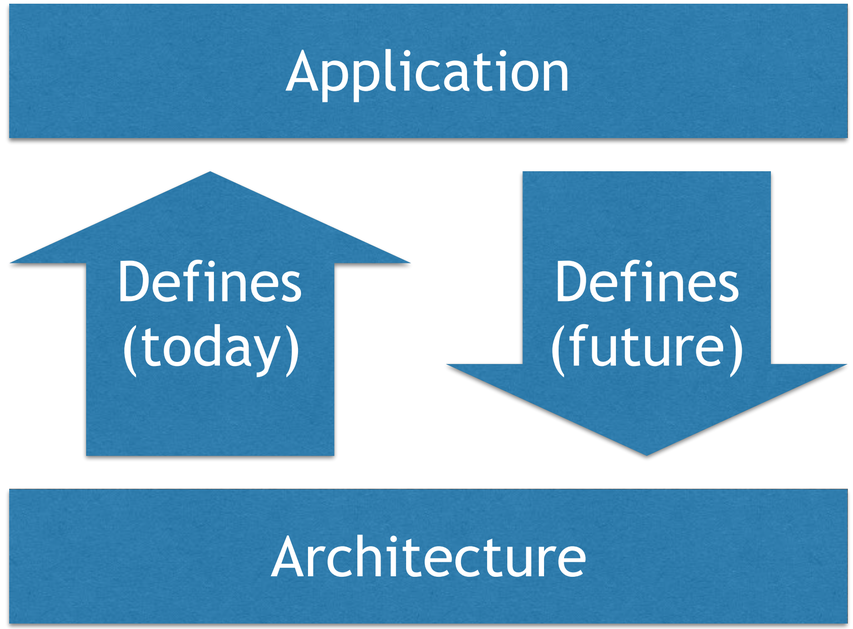
Snooping coherence

Scalable coherence - directories, token

Transactional memory

Sequential and relaxed consistency models

Trends & constraints



LEARN TO LOVE THE PICOJOULE