Google

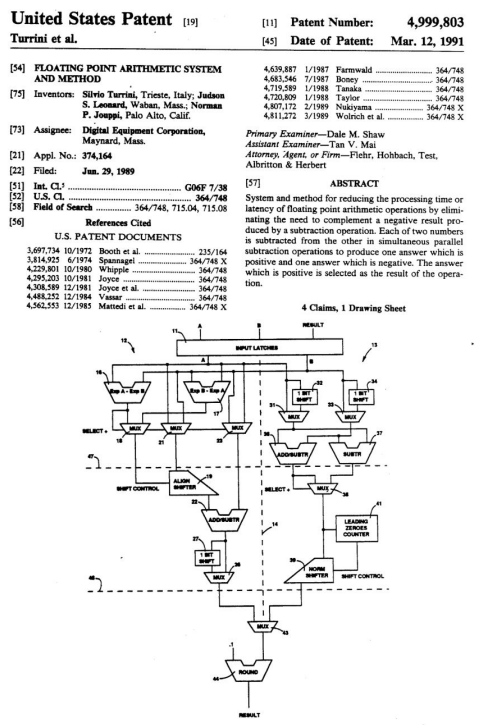
ML:
The Big, the Small, and the Not Right at All

Norman P. Jouppi, with contributions from the TPU team

June 11, 2024

I’ve Always Appreciated Computer Arithmetic

- My first US patent (and many others) were on computer arithmetic



United States Patent [19]
Turrini et al.

[11] **Patent Number: 4,999,803**
[45] **Date of Patent: Mar. 12, 1991**

[54] **FLOATING POINT ARITHMETIC SYSTEM AND METHOD**

[75] Inventors: **Silvio Turrini**, Trieste, Italy; **Judson S. Leonard**, Waban, Mass.; **Norman P. Jouppi**, Palo Alto, Calif.

[73] Assignee: **Digital Equipment Corporation**, Maynard, Mass.

[21] Appl. No.: **374,164**

[22] Filed: **Jun. 29, 1989**

[51] **Int. Cl.**[5] **G06F 7/38**
[52] **U.S. Cl.** **364/748**
[58] **Field of Search** 364/748, 715.04, 715.08

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,697,734	10/1972	Booth et al.	235/164
3,814,925	6/1974	Spannagel	364/748 X
4,229,801	10/1980	Whipple	364/748
4,295,203	10/1981	Joyce	364/748
4,308,589	12/1981	Joyce et al.	364/748
4,488,252	12/1984	Vassar	364/748
4,562,553	12/1985	Mattedi et al.	364/748 X
4,639,887	1/1987	Farmwald	364/748
4,683,546	7/1987	Boney	364/748
4,719,589	1/1988	Tanaka	364/748
4,720,809	1/1988	Taylor	364/748
4,807,172	2/1989	Nukiyama	364/748 X
4,811,272	3/1989	Wolrich et al.	364/748 X

Primary Examiner—Dale M. Shaw
Assistant Examiner—Tan V. Mai
Attorney, Agent, or Firm—Fiehr, Hohbach, Test, Albritton & Herbert

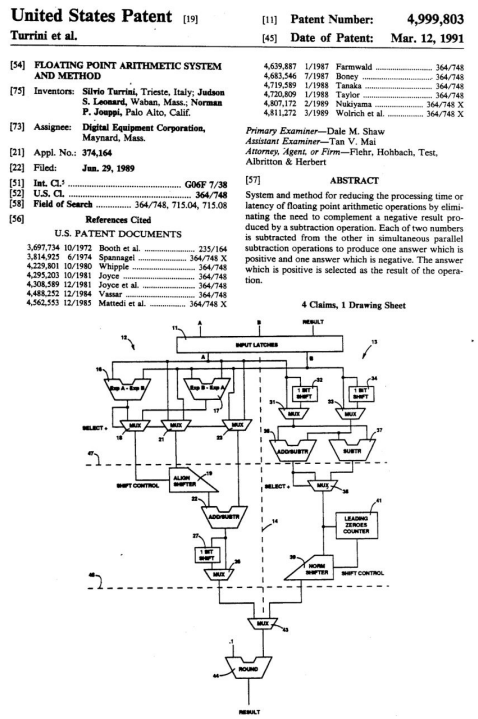
[57] **ABSTRACT**

System and method for reducing the processing time or latency of floating point arithmetic operations by eliminating the need to complement a negative result produced by a subtraction operation. Each of two numbers is subtracted from the other in simultaneous parallel subtraction operations to produce one answer which is positive and one answer which is negative. The answer which is positive is selected as the result of the operation.

4 Claims, 1 Drawing Sheet

I’ve Always Appreciated Computer Arithmetic

- My first US patent (and many others) were on computer arithmetic
- Thanks to the organizers for the invitation to speak



United States Patent [19]
Turrini et al.

[11] **Patent Number: 4,999,803**
[45] **Date of Patent: Mar. 12, 1991**

[54] **FLOATING POINT ARITHMETIC SYSTEM AND METHOD**

[75] Inventors: **Silvio Turrini**, Trieste, Italy; **Judson S. Leonard**, Waban, Mass.; **Norman P. Jouppi**, Palo Alto, Calif.

[73] Assignee: **Digital Equipment Corporation**, Maynard, Mass.

[21] Appl. No.: **374,164**

[22] Filed: **Jun. 29, 1989**

[51] **Int. Cl.**[5] **G06F 7/38**
[52] **U.S. Cl.** **364/748**
[58] **Field of Search** 364/748, 715.04, 715.08

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,697,734	10/1972	Booth et al.	235/164
3,814,925	6/1974	Spannagel	364/748 X
4,229,801	10/1980	Whipple	364/748
4,295,203	10/1981	Joyce	364/748
4,308,589	12/1981	Joyce et al.	364/748
4,488,252	12/1984	Vassar	364/748
4,562,553	12/1985	Mattedi et al.	364/748 X
4,639,887	1/1987	Farmwald	364/748
4,683,546	7/1987	Boney	364/748
4,719,589	1/1988	Tanaka	364/748
4,720,809	1/1988	Taylor	364/748
4,807,172	2/1989	Nukiyama	364/748 X
4,811,272	3/1989	Wolrich et al.	364/748 X

Primary Examiner—Dale M. Shaw
Assistant Examiner—Tan V. Mai
Attorney, Agent, or Firm—Fiehr, Hohbach, Test, Albritton & Herbert

[57] **ABSTRACT**

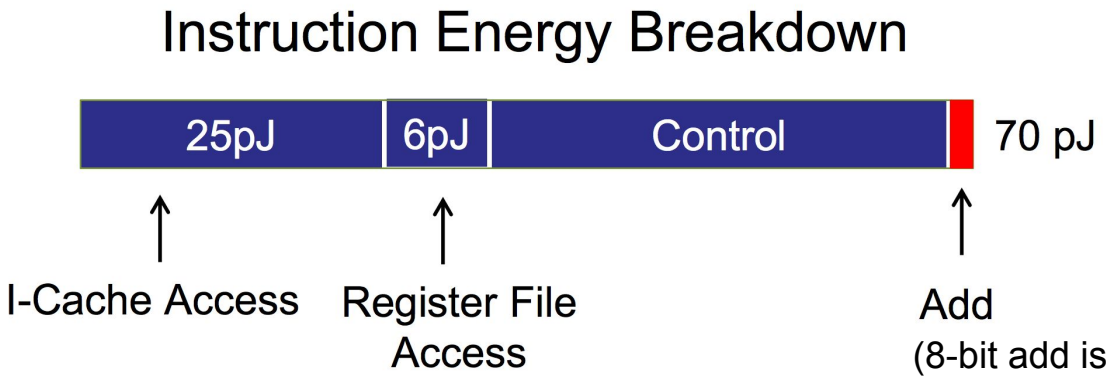
System and method for reducing the processing time or latency of floating point arithmetic operations by eliminating the need to complement a negative result produced by a subtraction operation. Each of two numbers is subtracted from the other in simultaneous parallel subtraction operations to produce one answer which is positive and one answer which is negative. The answer which is positive is selected as the result of the operation.

4 Claims, 1 Drawing Sheet

The Big

Key Insight #1

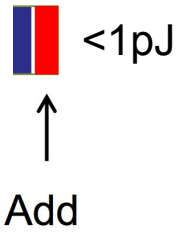
- Energy for control logic, SRAM, and register accesses needed by matrix multiply dominates in conventional processors
- Example from Mark Horowitz's ISSCC 2014 Keynote, slide 33: "Computing's Energy Problem: (and what we can do about it)":



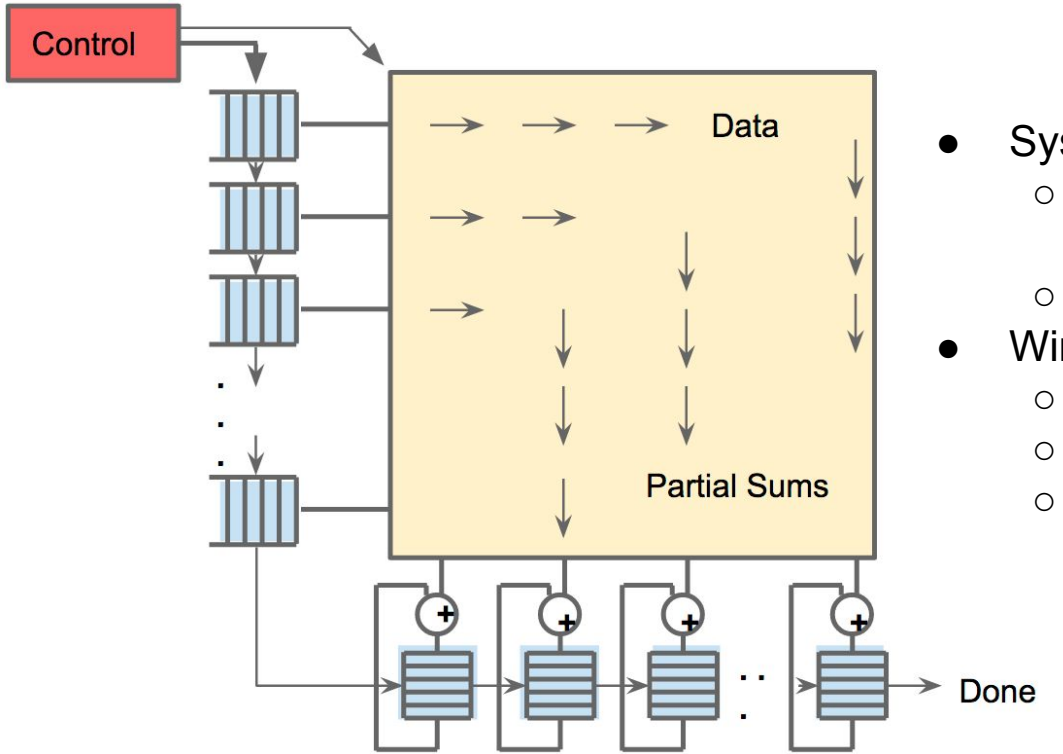
Key Insight #1

- Solution: matrix operations on a 256x256 systolic array
  - Eliminate complex control logic (use pipelined enable bit)
  - Reuse fetched memory and register data >100X
  - Reduce energy overhead per compute by >10X

Instruction Energy Breakdown



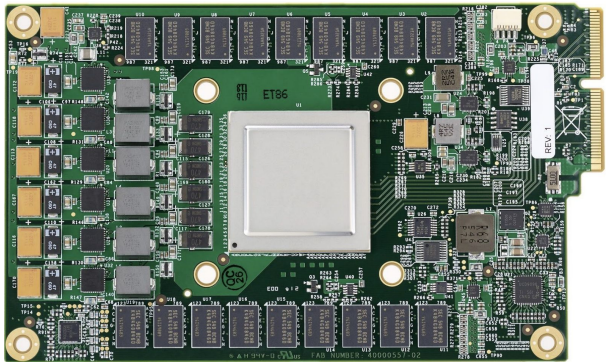
Systolic Execution: Data is Pipelined

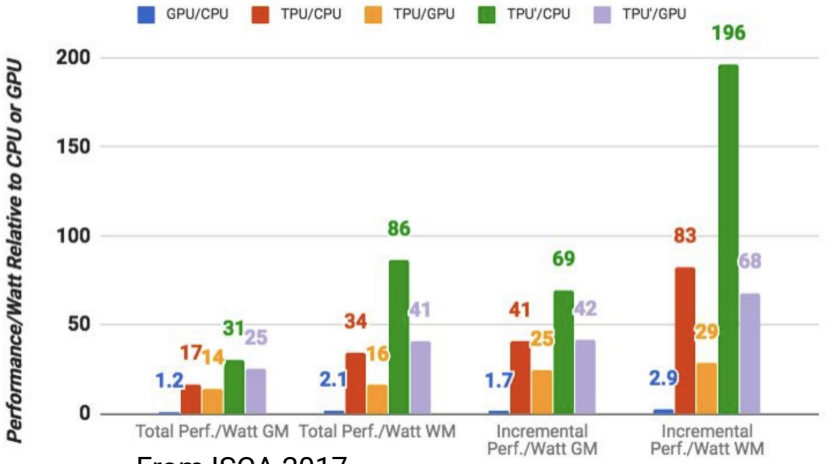


- Systolic arrays first proposed in 1970's
  - But largely forgotten in mainstream computer architecture
  - Only multipliers, adders, and flops
- Wiring by abutment within array
  - Saves wire power
  - Avoids memory accesses
  - No complex control logic

Late 2013

- TPUv1 project started
  - TPU = Tensor Processing Unit, an example of a DSA
  - DSA = Domain-specific architecture
  - Tensor = multidimensional array
- Provided >10X better perf/TCO than contemporary alternatives
  - perf/TCO = end-to-end performance / total cost of ownership (including power over lifetime)
  - Simple to deploy PCIe card
  - But it only accelerated inference





From ISCA 2017

Late 2014

- TPUv1 was being fabbed
- We realized training capability was the limiting factor to producing models
- People thought a training chip would be too complicated to build

Late 2014

- TPUv1 was being fabbed
- We realized training capability was the limiting factor to producing models
- People thought a training chip would be too complicated to build
- So we decided to build a training supercomputer 😀

How Much Should We Specialize For Existing Models?

- We don't know what models will be in 2-8 years
  - But we know they will be based on tensor math
- Fun fact: when we were initially brainstorming for TPUv1, I proposed a hardwired convolution unit that would be more efficient for image recognition
  - Luckily that idea only lasted a day, because convolutions changed in size (e.g., 7x7 vs.11x11, depthwise vs. width, etc.)
  - And while image recognition was an early customer, after a couple of years it was only a small part of the workload
- So it was better to give up the last 10-20% of optimization in order not to overspecialize the accelerator and make it obsolete after a year

Basic Plan for TPUv2

- Don’t invent anything more than necessary
  - Required to meet aggressive schedule
- Codesign from compiler down to chip physical design
- Start from a typical vector CPU architecture and add matrix operations
  - Similar to how the Cray-1 extended previous scalar machines with vector operations
  - Advantage: start with an architecture model with a compiler and add stuff
  - Leverage long-known compiler techniques for matrices in HPC (e.g., blocking, loop unrolling)
  - Use 8-operation VLIW architecture baseline since compiler schedules multiple ops/cycle
  - 8 instructions per cycle is a super-beefy scalar core!

Cray-1 Architecture

Circa 1975

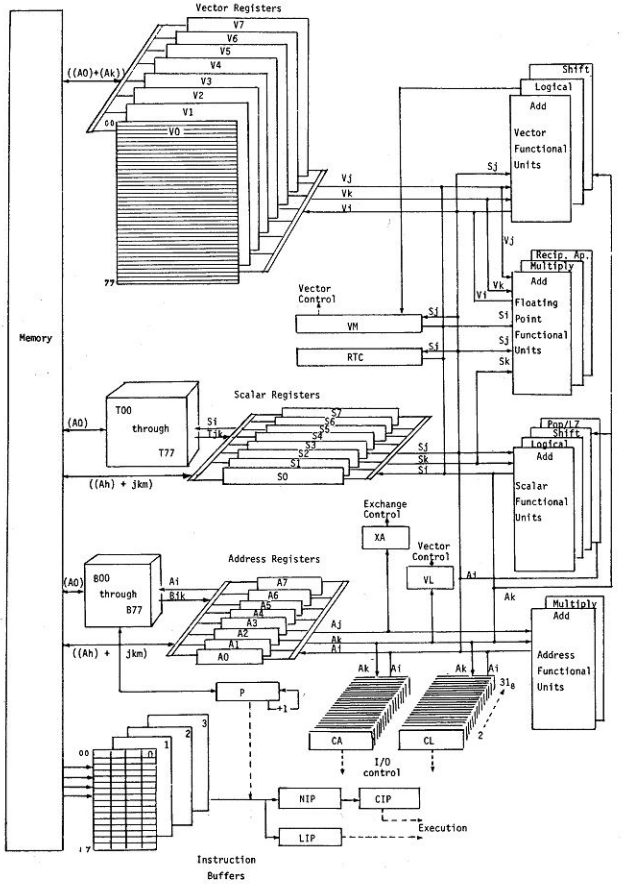


Figure 3-1. Computation section

This part looks like previous CDC6600 and CDC7600 machines

Cray-1 Architecture

Circa 1975

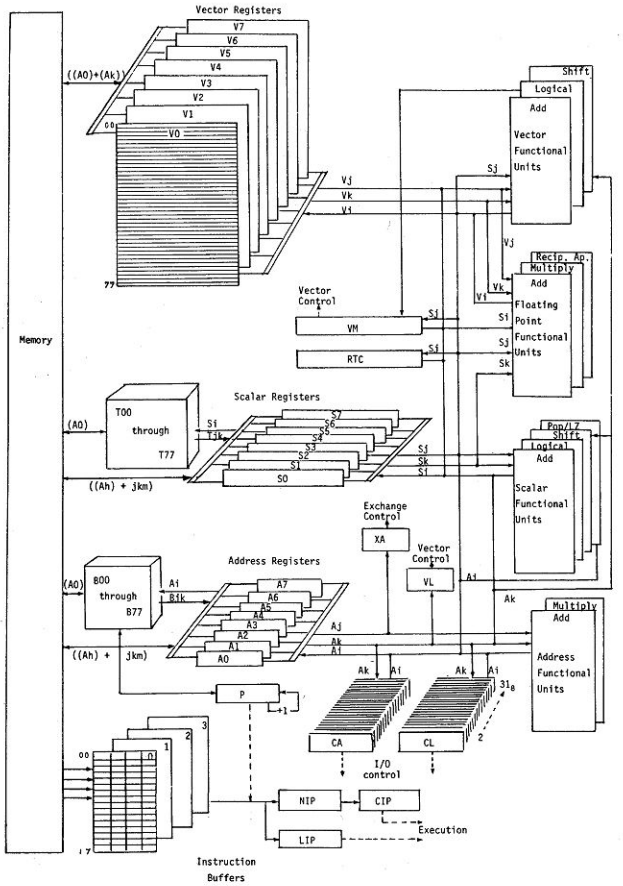


Figure 3-1. Computation section

Cray-1 added vector hardware in a consistent manner

This part looks like previous CDC6600 and CDC7600 machines

Basic Plan for TPUv2 (Part 2)

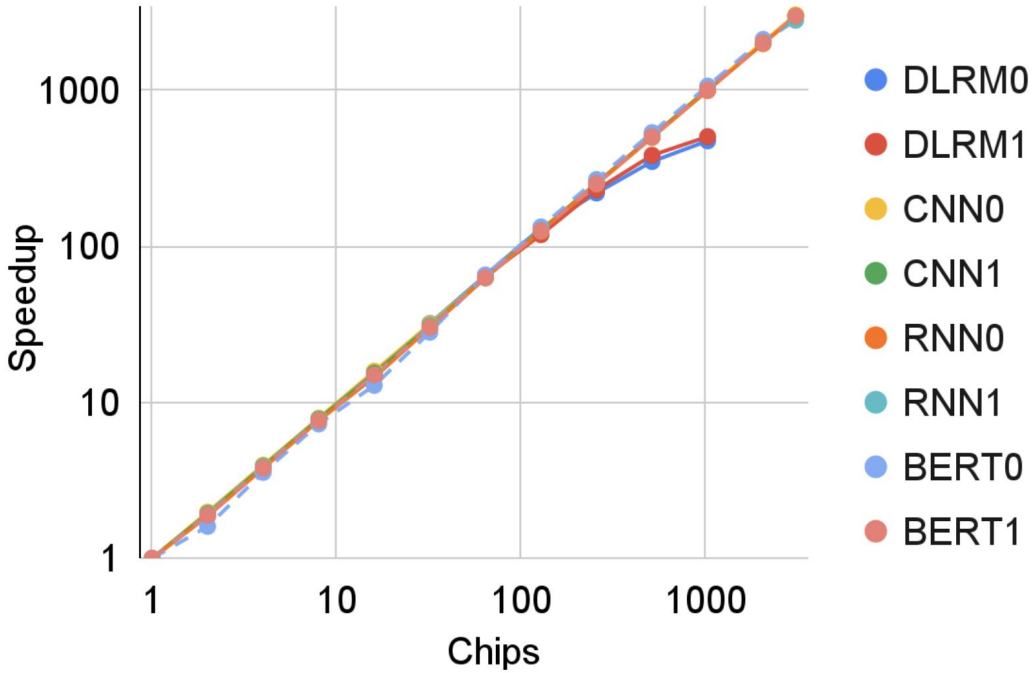
Key Insight #2

- Connect TPUs with **shared memory** distributed with high-bandwidth torus (ICI)
  - Similar to the Cray T3E torus, but simpler
  - Leverages the array structure of tensor math mapped to compute
- ICI is 50X faster and 10X cheaper than Ethernet
- **ICI is the second key TPU feature (after systolic arrays)**

Scalability on Real Workloads

Key Insight #2

- Due to **shared memory** using **extreme interconnects** at unprecedented scales (Many ExaFLOPS):
  - No overheads from Ethernet protocol stacks, etc.
  - 99% scaling efficiency on 75% of workloads to 3K TPUv4



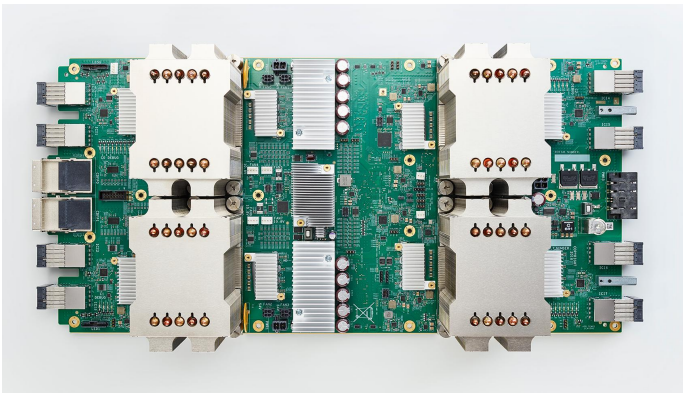
Basic Plan for Jellyfish (Part 3)

Key Insight #3

- Training was currently being done on CPUs and GPUs using FP32
  - Google's SW stored FP32 values in 16 bits to save storage space
  - Conversion from FP32 to 16 bits was performed by simple truncation (ouch!)
  - Preserved dynamic range while reducing precision
  - This datatype was called BF16 (Brain Float 16) in the SW
- Existing 16-bit FP formats (IEEE FP16) didn't have enough dynamic range
- We realized we could supply BF16 inputs to multipliers, keep all product bits (i.e., FP24), and perform FP32 accumulation and get identical results as current SW
  - This saved multiplier HW
  - And we rounded to nearest even on conversion, giving better results
- But most importantly, it maintained SW compatibility with CPUs and GPUs
  - Models could train on CPUs, GPUs, and TPUs and all get the same results
- **Hence BF16 is the 3rd key TPU feature**

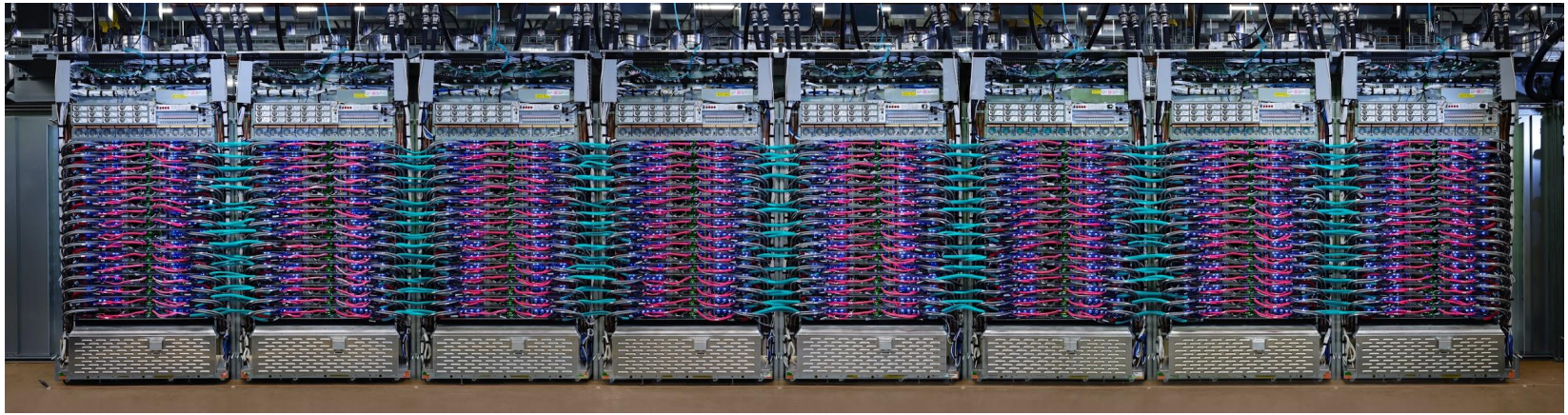
TPUv2

- 256 chips connected in a 2D torus
  - Narrow TPU trays, 4 per rack shelf
  - ICI only ran SerDes at 40Gb/s
- Air cooled due to lower power consumption and time-to-market
- Servers were in separate racks



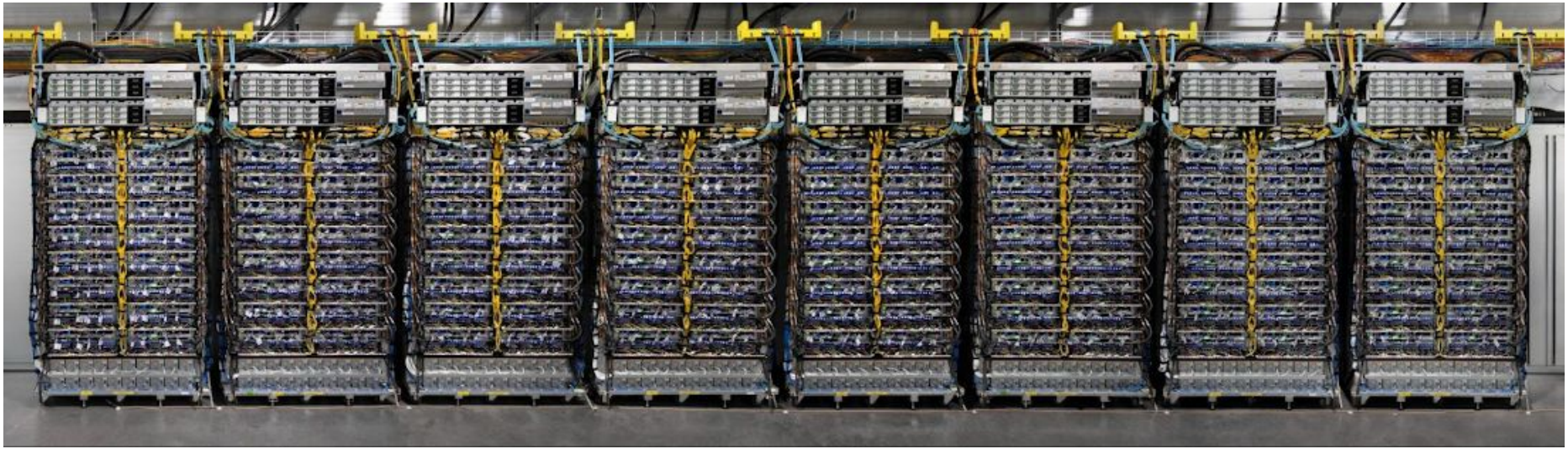
TPUv3

- Most of the team was working on TPUv2 bringup
- Hence only limited logic changes to TPUv2 were possible in TPUv3, but:
  - Optimized chip physical design to make room for 2nd MXU per core
  - Larger scale (4X chips) in 2X racks with 2X rack power supplies per rack
  - First TPU with water cooling
  - Optical cables for wrap-around torus links
  - 2X capacity per HBM



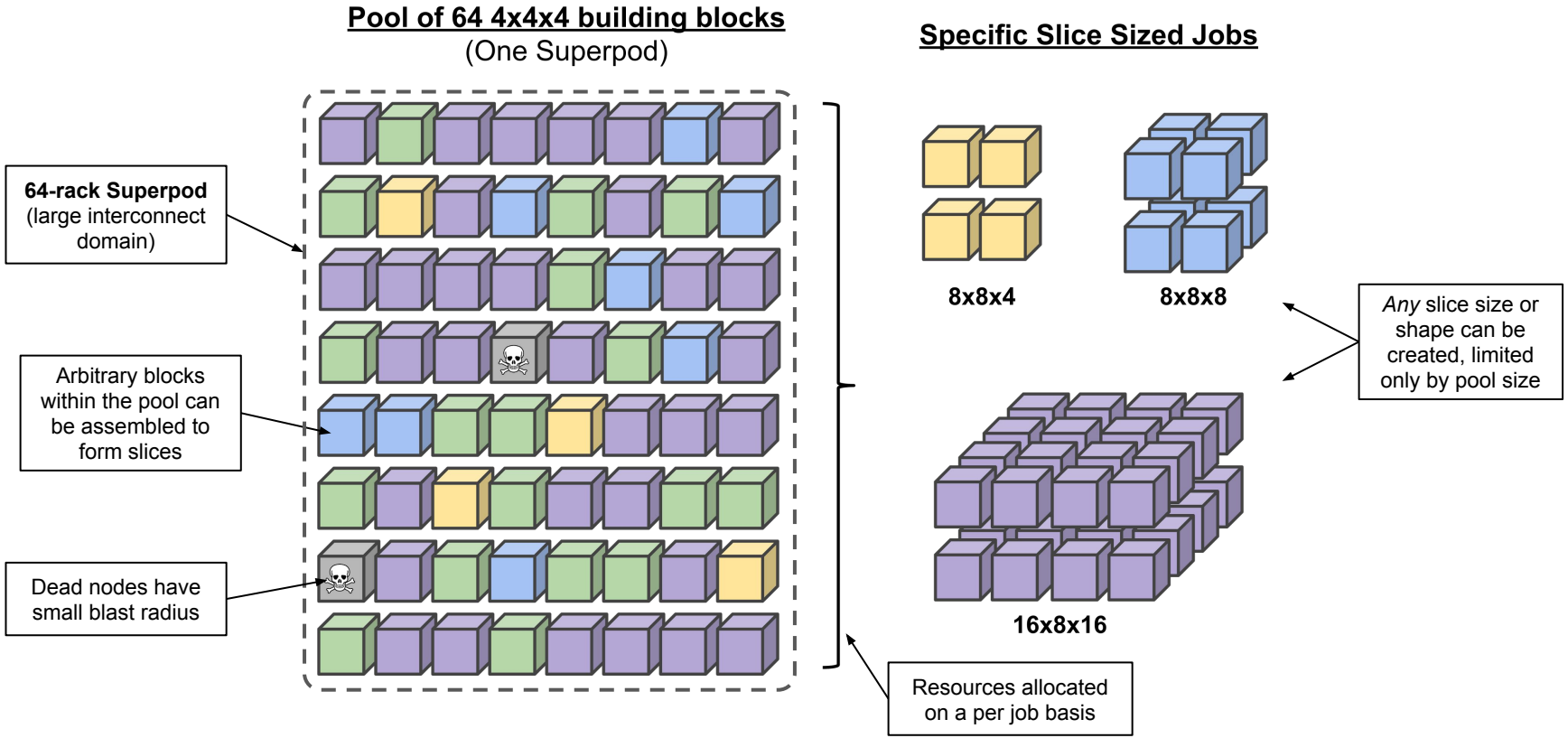
TPU v4p

- Superpods of 64 racks of water-cooled compute (8 shown in photo)
  - Provides over 1 ExaFLOP
- Superpods are connected via datacenter networking into bigger clusters



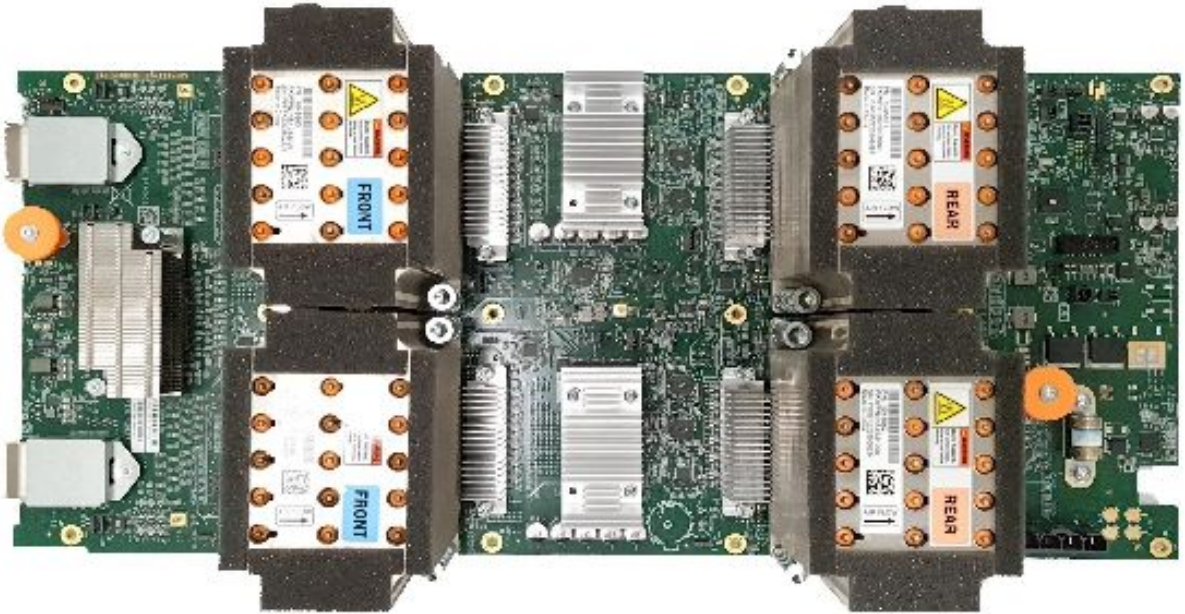
What Is a TPU Superpod?

A large pool of building blocks that can be connected on a per-job basis to form larger slices.



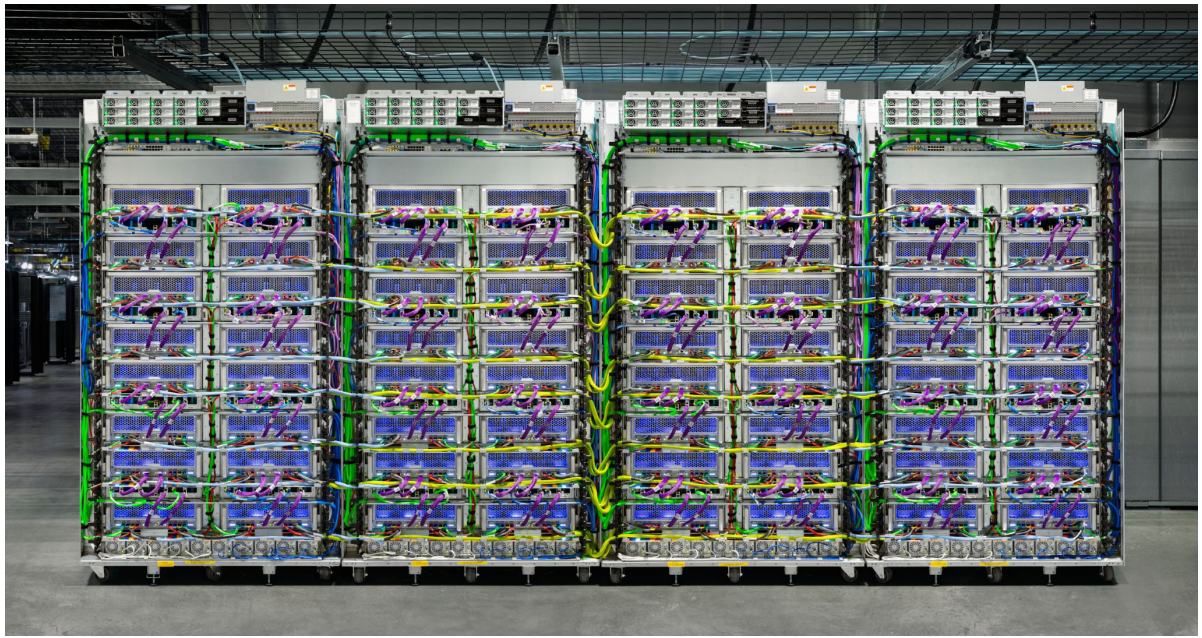
TPU v4e

- Air cooled for worldwide deployments
- 4 racks of compute, connected via datacenter networking into bigger clusters



TPU v5e

- Air cooled for worldwide deployments
- 4 racks of compute, connected via datacenter networking into bigger clusters



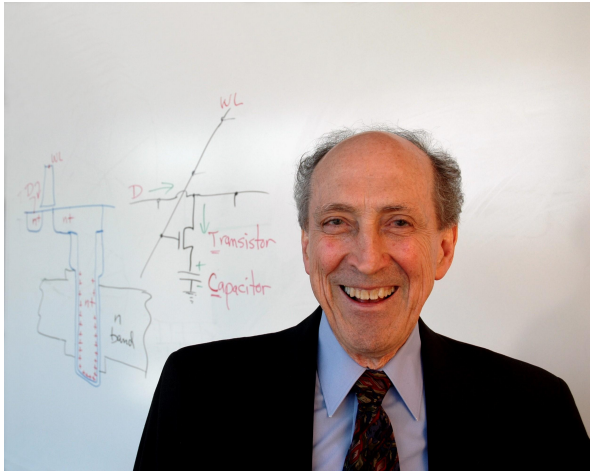
V5p Superpods (8960 chips each)

Google I/O 2024 Keynote

The Small

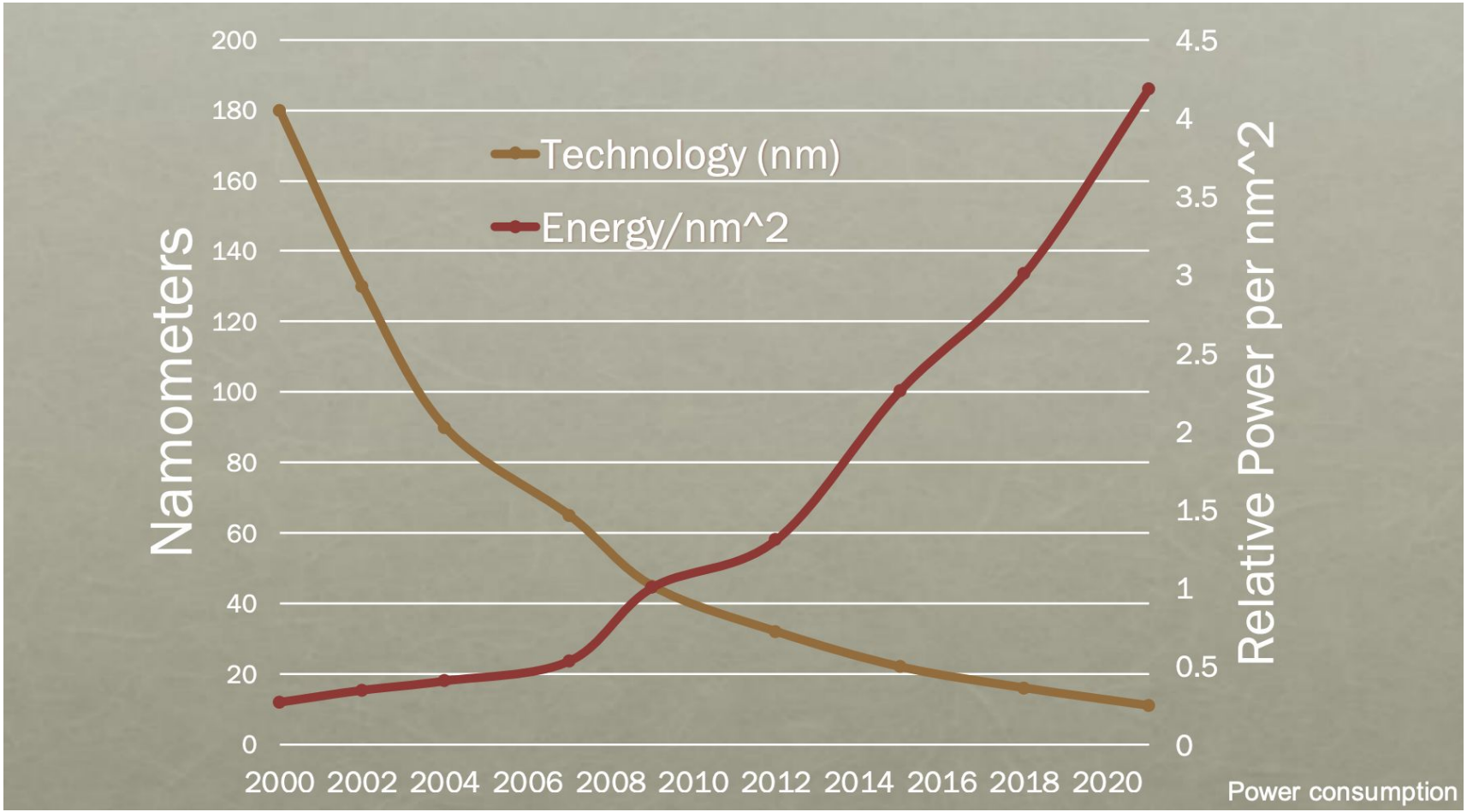
The End of Dennard Scaling

- Dennard Scaling:
  - Shrinking process lithography gives you more transistors
  - Scale voltage down
  - Power per area at same frequency constant
- This scaling ended around around 2004
- Hence chip **power density** will increase every year from now to the end of lithographic scaling
  - This is not the same world as 15 years ago
  - We need to “Think Different”



Robert Dennard, member NAE
P.S. He also invented the 1T1C DRAM

- Reference: John Hennessy’s Turing Award talk: “The end of Dennard scaling is an equally important problem as the end of Moore’s Law, but doesn't receive as much attention”



From John Hennessy's Plenary talk at DARPA 2018 ERI Summit

SRAM Accesses Consume a Lot of Power

- 2014 data in Accelerator's Energy Problem:

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

Memory	
SRAM	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ

SRAM Accesses Consume a Lot of Power

- 2014 data in [Accelerator's Energy Problem]:

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

Memory	
SRAM	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ

- Since 2014 computation has gotten ~3X cheaper but SRAM is still about the same

SRAM Accesses Consume a Lot of Power

- 2014 data in [Accelerator's Energy Problem]:

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

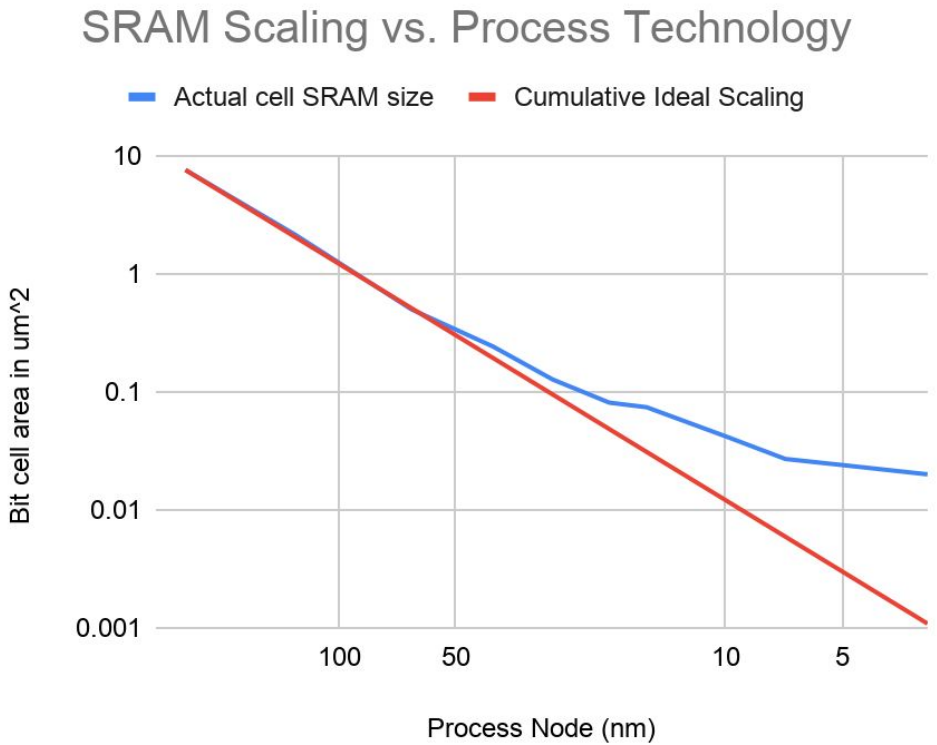
FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

Memory	
SRAM	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ

- Since 2014 computation has gotten ~3X cheaper but SRAM is still about the same

∴ Data reuse is REALLY important for efficiency

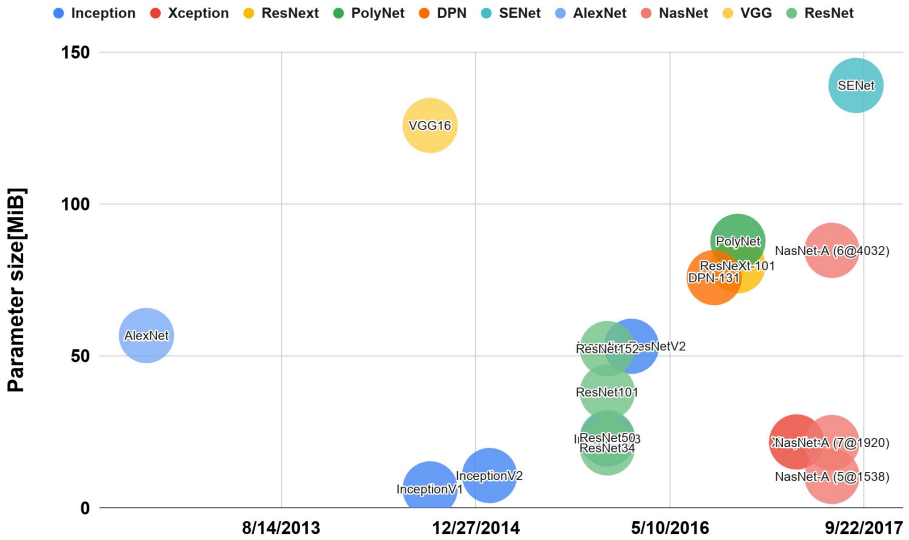
EoML: SRAM Density Was the First to Stop Scaling



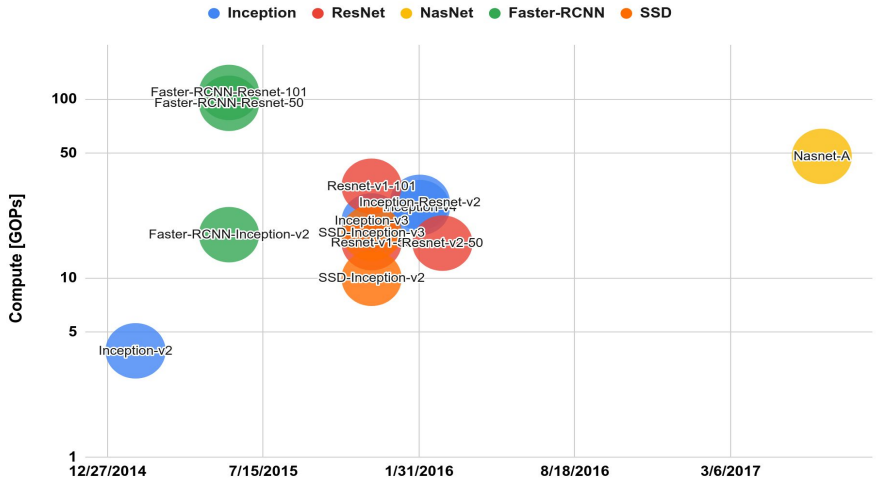
On-chip SRAM not scaling; >10X gap

Data from WikiChip

Early Trends: Image Model Size Over Time



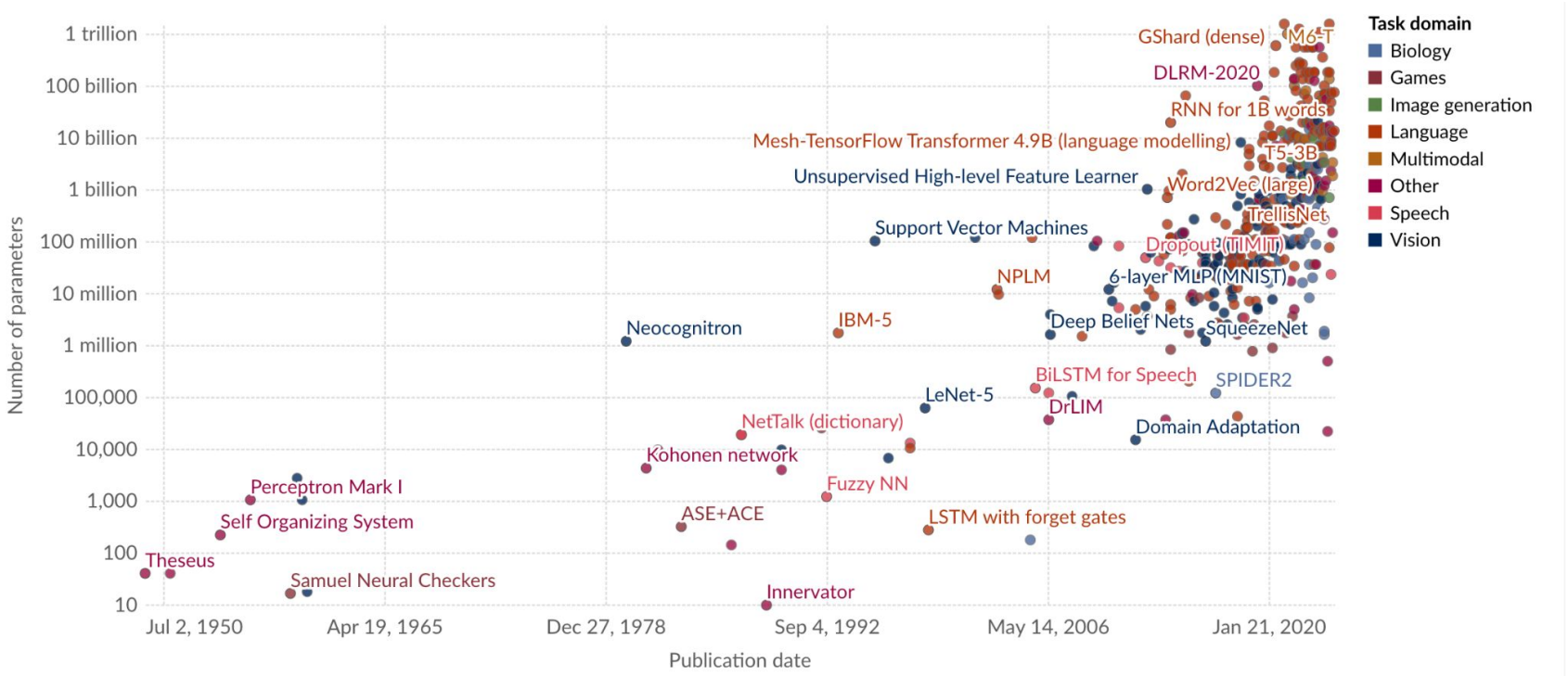
Parameter size of Imagenet models vs. time.



Compute Requirements of Imagenet models vs. time.

~10X model size growth in 4 years = 78% per year

Current Era: Model Growth Accelerates to 10X/Year!

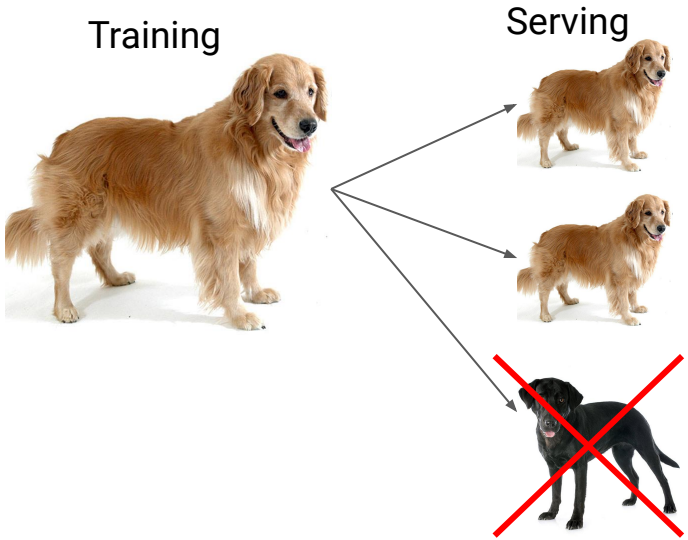


Source: Epoch (2024) – with minor processing by Our World in Data

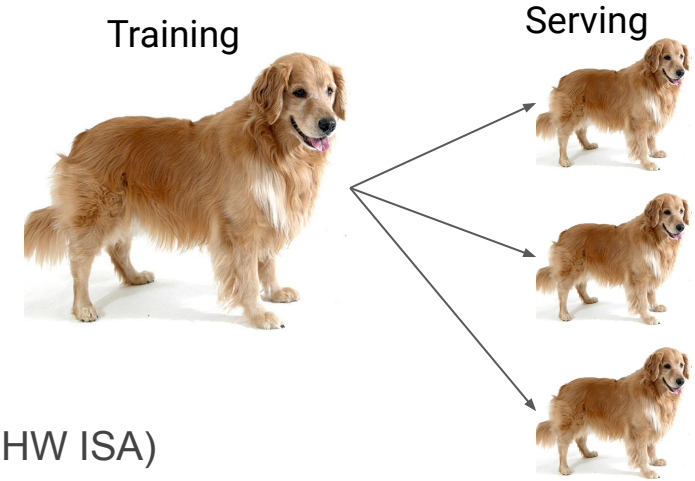
1+ year design, 1+ year deployment, 6+ year service

- Scrimping on memory can be one of the easiest ways to reduce cost
- But memory requirements have grown incessantly since the first computers
  - EDSAC (1949) only had 1KB of memory
  - Remember: “640K ought to be enough for anybody”?
- But on-chip SRAM isn’t scaling!
- Need a memory hierarchy
  - On-chip SRAM
  - On-package HBM
  - Off-package cheap swappable DIMMs
- Adequate memory provisioning raises costs in the near term
  - But increases the useful lifetime dramatically -> net positive

Backwards ML Compatibility



Backwards ML Compatibility



- Backwards ML compatibility (at the XLA level, not the HW ISA)
  - Use same software stack for inference and training
  - Performance correlation - if it trains well, inference should work similarly
  - Same exception behavior
  - Avoids accuracy problems - some customers (e.g., Ads) have very stringent requirements
    - Subtle quantization problems delayed rolling out a Seastar model by months
- Luiz Barroso (while in Geo): “We want to train models overnight and deploy them the next day without the involvement of anyone with ML experience.”
  - This is possible with backwards ML compatibility

Reduced Precision Formats

- Important to reduce the cost of serving large LLMs:
  - Memory footprint
  - Compute
  - Energy / power consumption
- Subject of first ARITH 2024 keynote
- Must preserve accuracy compared to larger formats
  - One mistake in a million can make the news
- Automation is crucial to enable adoption
  - Hyperscale Hardware Optimized Neural Architecture Search

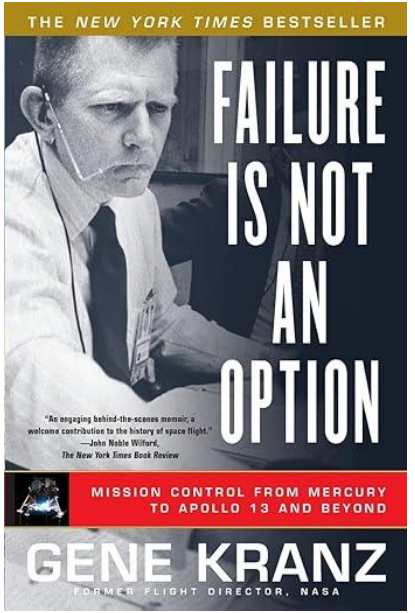
The Not Right at All

RAS: Reliability, Availability, and Serviceability

- Received a lot of attention in the past from financial applications
  - Tandem Nonstop
  - IBM mainframes
  - Etc.
- Now a significant issue with ML training
  - Similar to other large-scale supercomputer calculations
- Real-time reliability not required
  - Recovery from checkpoints fine
  - But errors must be detected

Silent Data Corruption (SDC)

- Compared to silent data corruption, failure is a good option
  - SDCs can cause training to diverge instead of converge
- But they can be expensive to detect
  - Tandem Nonstop fully duplicates computation and storage
- Algorithm Based Fault Tolerance
  - Opportunities for further research
  - A range of techniques would be useful
- Additional references
  - "Algorithm-Based Fault Tolerance for Matrix Operations," by KUANG-HUA HUANG, AND JACOB A. ABRAHAM in IEEE Trans. Computer, June 1984.
  - A Comparison of Several Fault-Tolerance Methods for the Detection and Correction of Floating-Point Errors in Matrix-Matrix Multiplication, by Valentin Le Fèvre, Thomas Herault, Julien Langou & Yves Robert in Euro-Par 2020: Parallel Processing Workshops.



Conclusions

- WIP…

Q & A