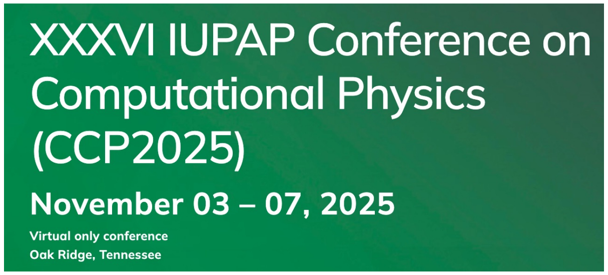


XXXVI IUPAP Conference on Computational Physics (CCP2025)

November 03 – 07, 2025

Virtual only conference
Oak Ridge, Tennessee

High-Performance Computing and Responsibly Reckless Algorithms

Jack Dongarra
University of Tennessee
University of Manchester

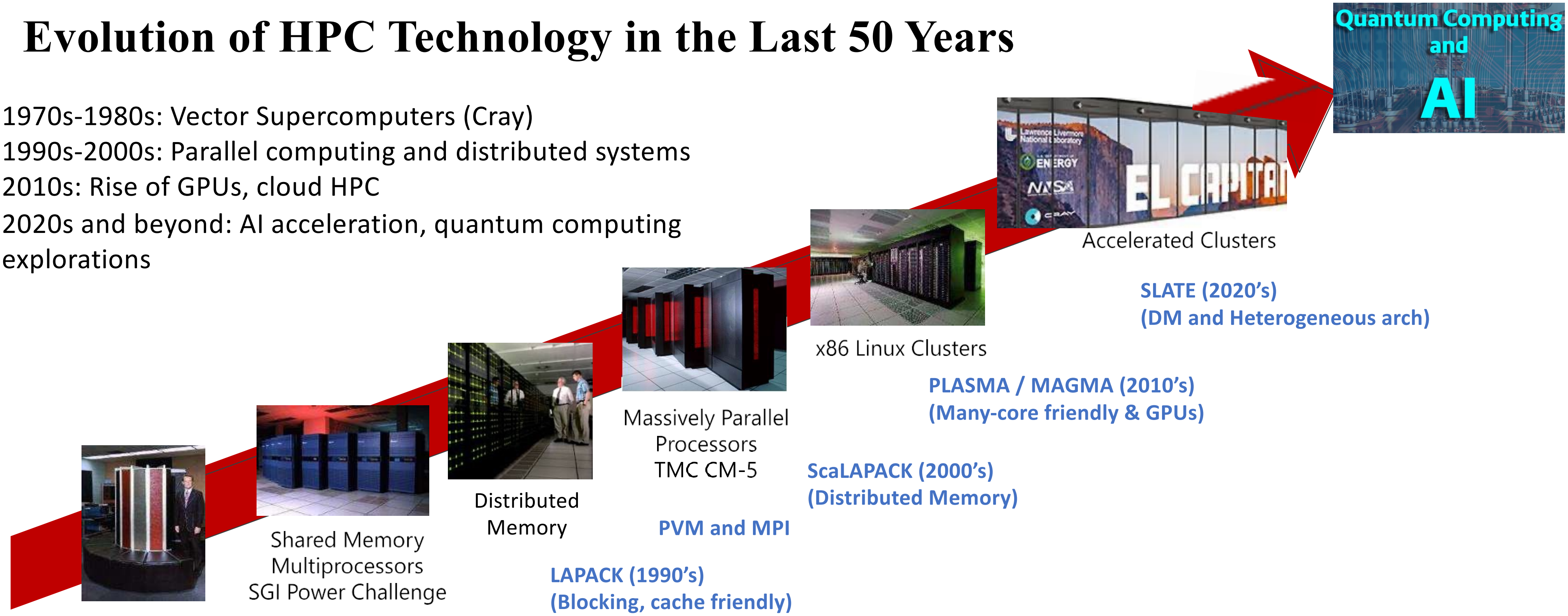
Evolution of HPC Technology in the Last 50 Years

1970s-1980s: Vector Supercomputers (Cray)

1990s-2000s: Parallel computing and distributed systems

2010s: Rise of GPUs, cloud HPC

2020s and beyond: AI acceleration, quantum computing explorations



Vector Supercomputers
Cray-1

Shared Memory
Multiprocessors
SGI Power Challenge

Distributed
Memory

Massively Parallel
Processors
TMC CM-5

x86 Linux Clusters

Accelerated Clusters

EISPACK (1970’s) NATS Project
(Translation of Algol to F66)

Level 1 Basic Linear Algebra
Subprograms (BLAS)

LINPACK (1980’s)
(Vector operations)

Level 2 & 3 BLAS - ATLAS

LAPACK (1990’s)
(Blocking, cache friendly)

PVM and MPI

ScaLAPACK (2000’s)
(Distributed Memory)

PLASMA / MAGMA (2010’s)
(Many-core friendly & GPUs)

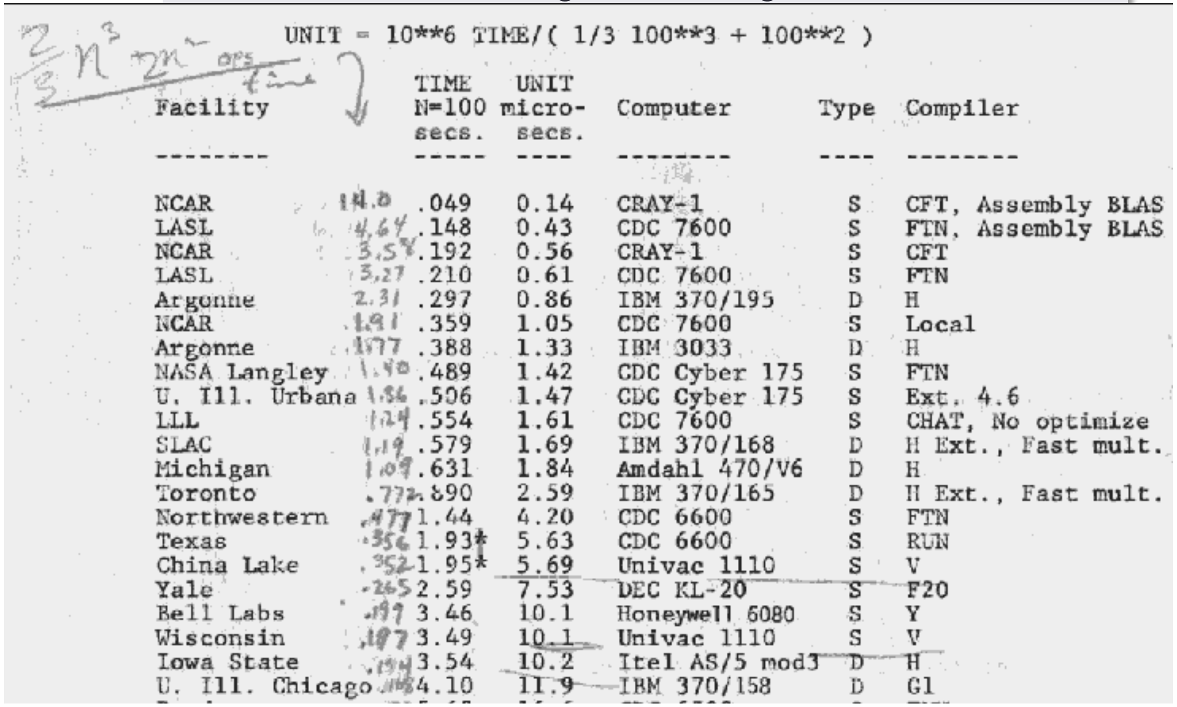
SLATE (2020’s)
(DM and Heterogeneous arch)

Evolving Software and Algorithms
Following Hardware Developments

An Accidental Benchmarker

LINPACK was an NSF Project w/ ANL, UNM, UM, & UCSD
We worked independently and came to Argonne in the summers

Top 23 List from 1977
Performance of solving *Ax=b* using LINPACK software



$\frac{2}{3}n^3 + 2n^2$ ops / time

UNIT = 10**6 TIME/(1/3 100**3 + 100**2)

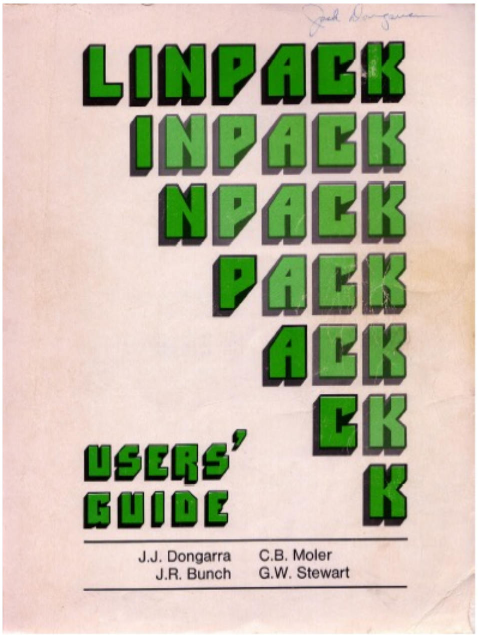
Facility		TIME N=100 secs.	UNIT micro-secs.	Computer	Type	Compiler
NCAR	14.0	.049	0.14	CRAY-1	S	CFT, Assembly BLAS
LASL	4.64	.148	0.43	CDC 7600	S	FTN, Assembly BLAS
NCAR	3.58	.192	0.56	CRAY-1	S	CFT
LASL	3.27	.210	0.61	CDC 7600	S	FTN
Argonne	2.31	.297	0.86	IBM 370/195	D	H
NCAR	1.91	.359	1.05	CDC 7600	S	Local
Argonne	1.77	.388	1.33	IBM 3033	D	H
NASA Langley	1.40	.489	1.42	CDC Cyber 175	S	FTN
U. Ill. Urbana	1.36	.506	1.47	CDC Cyber 175	S	Ext. 4.6
LLL	1.24	.554	1.61	CDC 7600	S	CHAT, No optimize
SLAC	1.19	.579	1.69	IBM 370/168	D	H Ext., Fast mult.
Michigan	1.09	.631	1.84	Amdahl 470/V6	D	H
Toronto	.772	.890	2.59	IBM 370/165	D	H Ext., Fast mult.
Northwestern	.477	1.44	4.20	CDC 6600	S	FTN
Texas	.356	1.93*	5.63	CDC 6600	S	RUN
China Lake	.352	1.95*	5.69	Univac 1110	S	V
Yale	.265	2.59	7.53	DEC KL-20	S	F20
Bell Labs	.199	3.46	10.1	Honeywell 6080	S	Y
Wisconsin	.197	3.49	10.1	Univac 1110	S	V
Iowa State	.194	3.54	10.2	Itel AS/5 mod3	D	H
U. Ill. Chicago	.168	4.10	11.9	IBM 370/158	D	G1

Appendix B of the Linpack Users' Guide

Designed to help users estimate the run time for solving systems of equation using the Linpack software.

First benchmark report from 1977;

Cray 1 to DEC PDP-10



LINPACK Benchmark ⟶ Top500

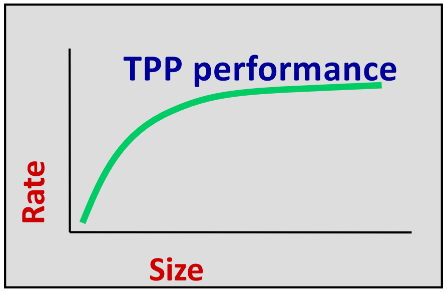
- Since 1977 I maintained a LINPACK Benchmark list.
- Hans Meuer and Erich Strohmaier had a list of fastest computers ranked by peak performance.
- Since 1993 listing of the 500 most powerful computers using 64-bit floating point arithmetic.
- Yardstick: Performance for

  Ax=b, dense problem

Maintained and updated twice a year:

SC'xy in the States in November

Meeting in Germany in June



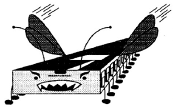
TOP 500 The List.

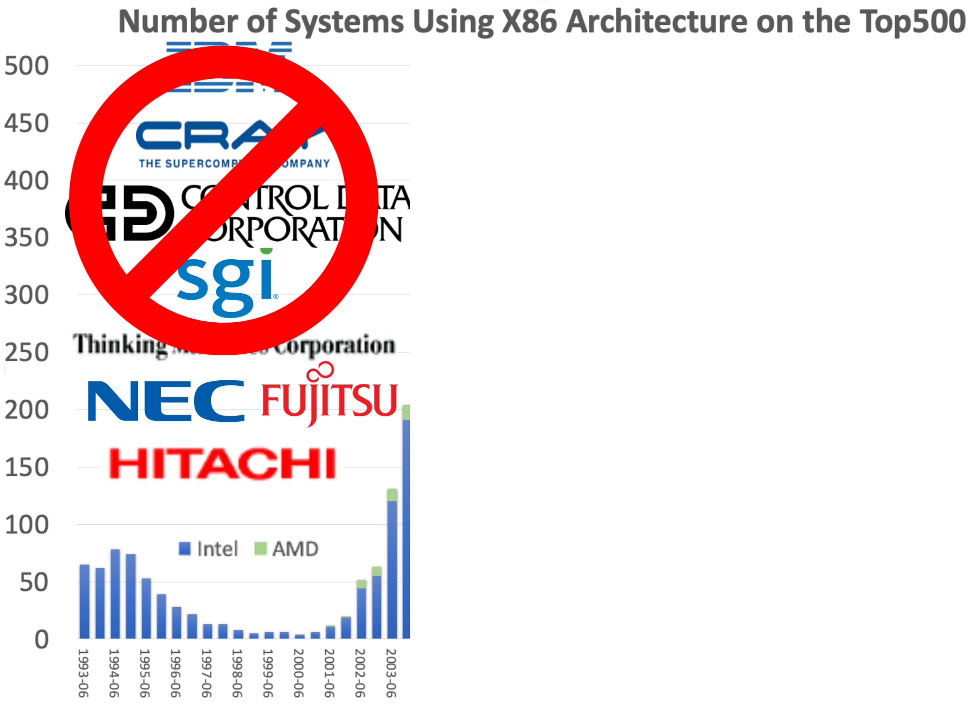
- TOP500 list began in 1993
  - 65 systems used Intel’s i860 architecture
  - Remainder had specialized architectures, mainly vector based

Most of the HPC systems were specially built for computational science applications

Major paradigm shift

Attach of the Killer Micros

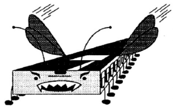




Scientific High Performance Computing based on Commodity Processors

Major paradigm shift

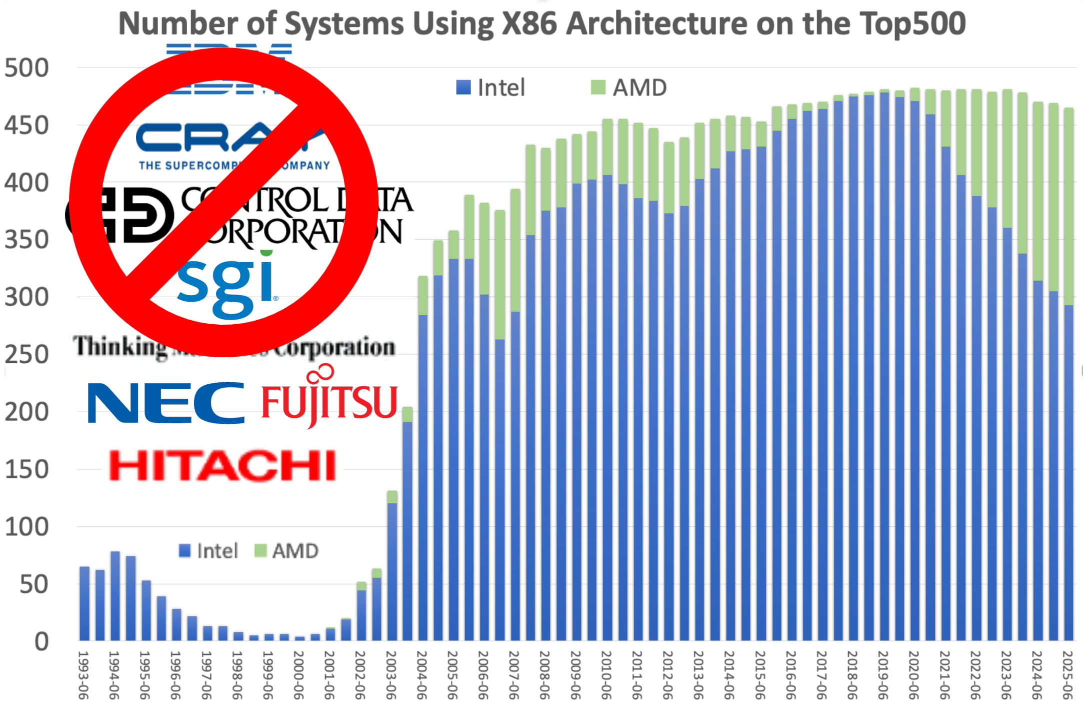
Attach of the Killer Micros



- TOP500 list began in 1993
  - 65 systems used Intel's i860 architecture
  - Remainder had specialized architectures, mainly vector based

- Today's TOP500 list
  - 59% of systems used Intel processors
  - Another 34% used AMD processors

- **93% of the systems use x86-64 architecture**
  - Many use GPU accelerators



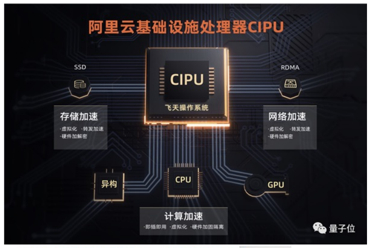
Today, Our HPC Systems are Based on Commodity Parts

- Commodity Processors
  - 93% of the Top500 system use X86 (Intel & AMD) instruction set
- Commodity Accelerators
  - 92% of accelerated systems use NVIDIA
- Commodity Interconnect
  - 85% of the Top500 systems use Ethernet or Infiniband
- Commodity OS
  - 100% of the Top500 systems run on Linux

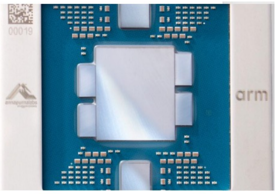
- Unlike the HPC Community, the Hyperscalers (Cloud Providers)
  - They are building their processors, accelerators, and interconnects

Cloud Providers are Designing and Using Their Own Processors

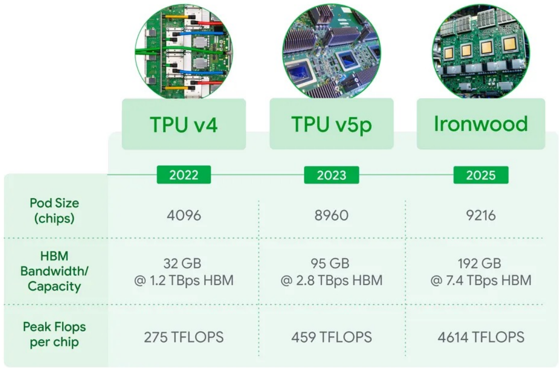
- Alibaba
  - CIPU, 128 core ARM based
  - Alibaba’s Elastic Compute Service



- AWS Graviton4
  - 96 ARM cores, 7 chiplet design
  - ~100 billion transistors, DDR5 memory



- Google TPU7
  - 2X TPU3 performance
  - 4096 units per “pod”
  - Reconfigurable optical interconnect



	TPU v4	TPU v5p	Ironwood
	2022	2023	2025
Pod Size (chips)	4096	8960	9216
HBM Bandwidth/ Capacity	32 GB @ 1.2 TBps HBM	95 GB @ 2.8 TBps HBM	192 GB @ 7.4 TBps HBM
Peak Flops per chip	275 TFLOPS	459 TFLOPS	4614 TFLOPS

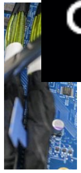
- Microsoft Azure
  - Project Catapult/Brainwave FPGA accelerator
  - Cobalt 100 (128 Neoverse N2 ARMv9 cores)
  - Maia100 (Athena) AI accelerator
  - $10B+ OpenAI investment/$80B in data centers

OpenAI

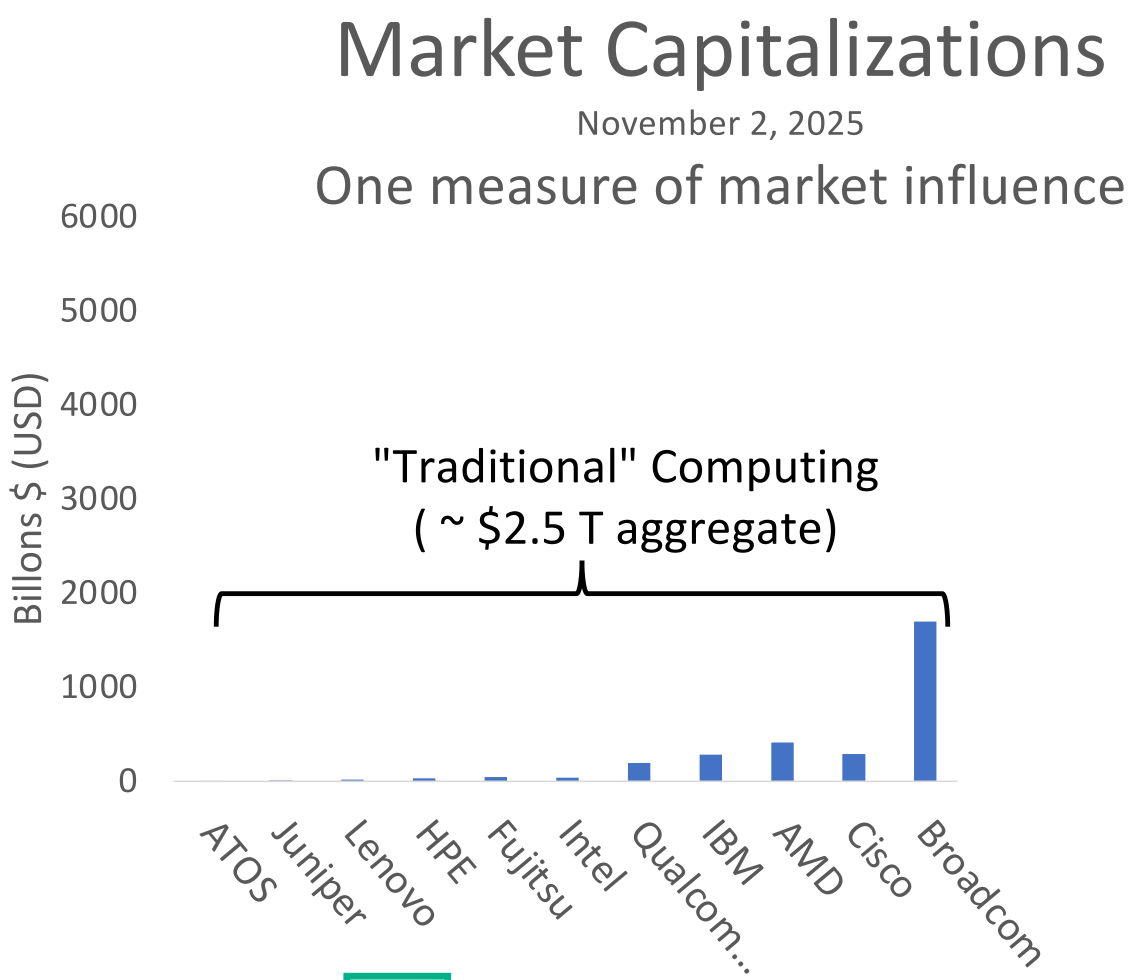
SambaNova™ SYSTEMS

GRAPHCORE

cerebras



groq™



Market Capitalizations

November 2, 2025

One measure of market influence

"Traditional" Computing
(~ $2.5 T aggregate)

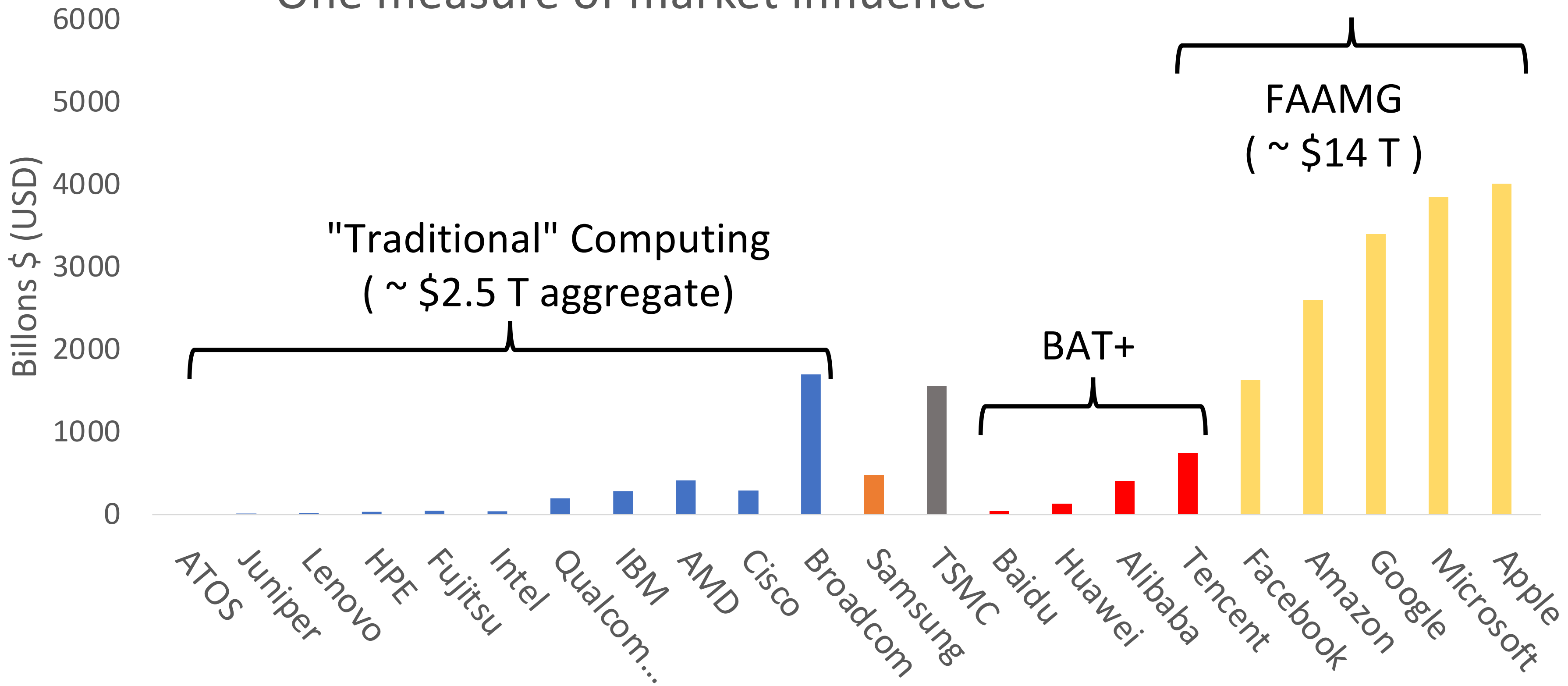
Hewlett Packard Enterprise intel IBM AMD

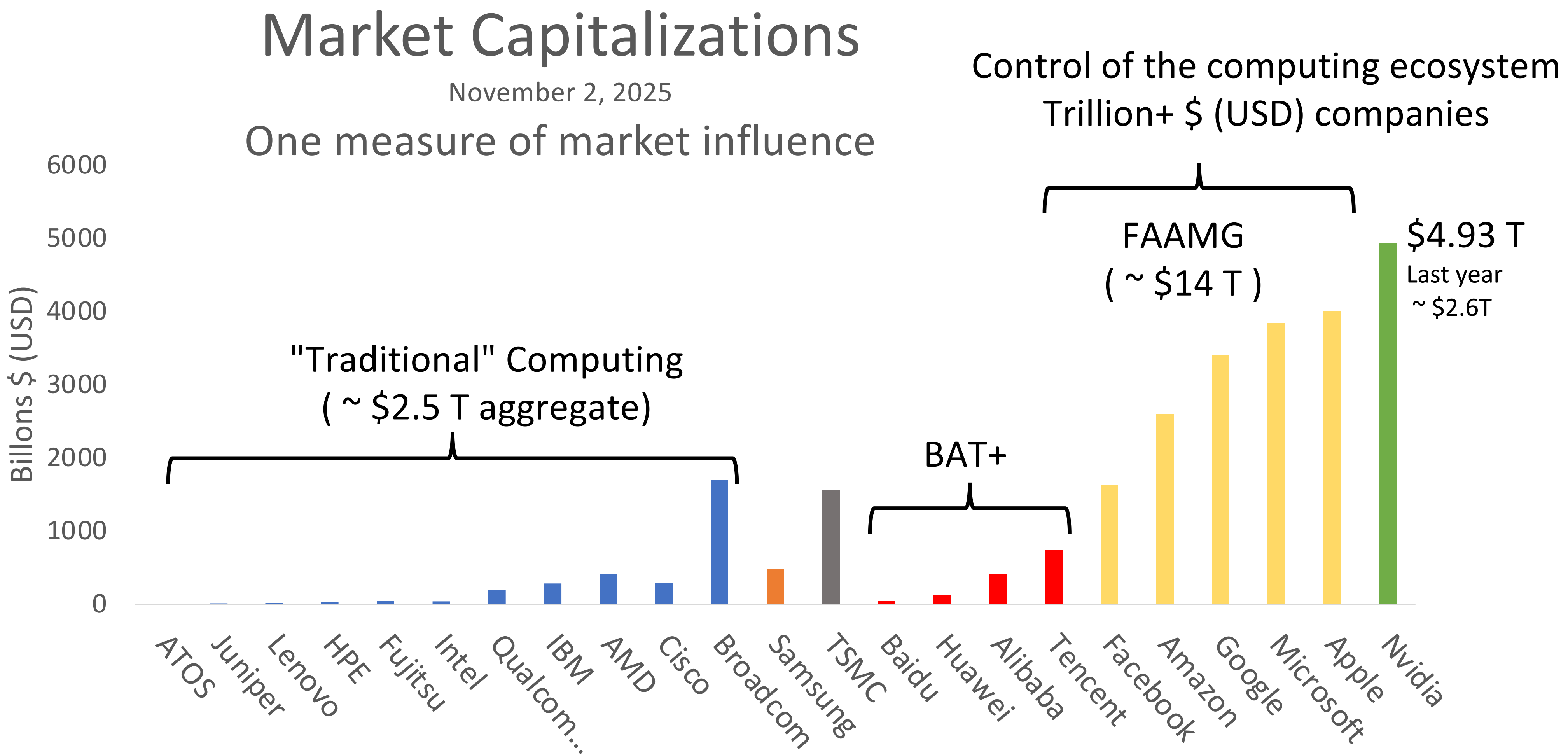
Market Capitalizations

November 2, 2025

One measure of market influence

Control of the computing ecosystem
Trillion+ $ (USD) companies





Market Capitalizations

November 2, 2025

One measure of market influence

Control of the computing ecosystem
Trillion+ $ (USD) companies

"Traditional" Computing
(~ $2.5 T aggregate)

BAT+

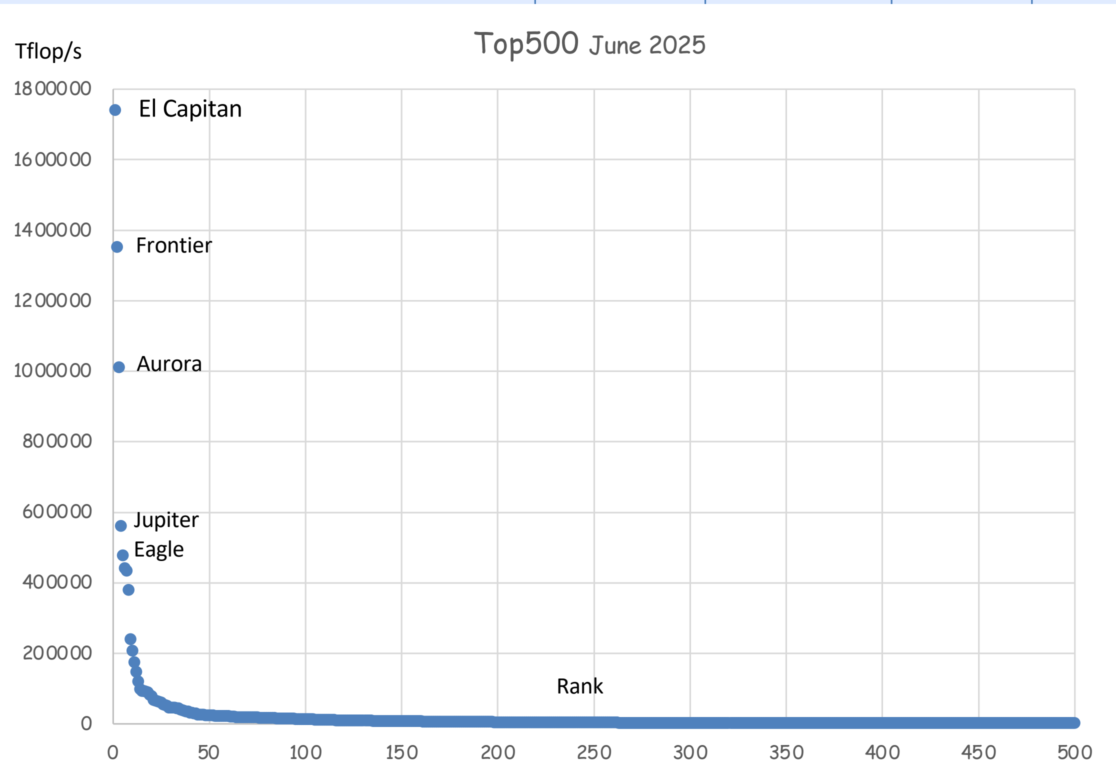
FAAMG
(~ $14 T)

$4.93 T
Last year
~ $2.6T

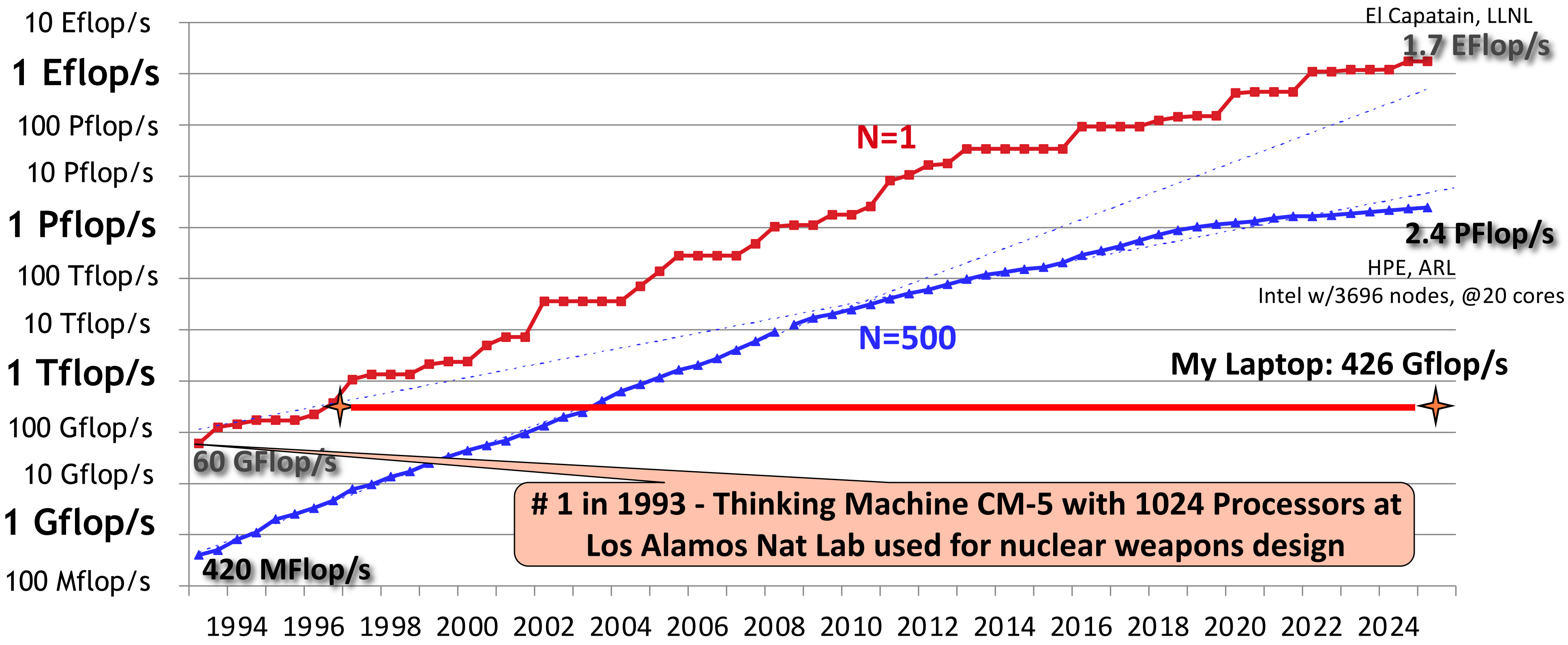
amazon Google Microsoft NVIDIA

June 2025: The TOP 10 Systems (54% of the Total Performance of Top500)

Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak	Power [MW]	GFlops/ Watt
1	DOE / NNSA LLNL	El Capitan GHz					29.5	58.9
2	DOE / OS Oak Ridge Nat Lab	Frontie 2 GH					24.6	55.0
3	DOE / OS Argonne Nat Lab	Auror Intel L					38.7	26.1
4	EuroHPC/FZL	JUPI 72 C					13.1	60.5
5	Microsoft, Azure Cloud						-	
6	Eni S.p.A.	HPE Cr 2GH					8.46	56.5
7	RIKEN Center for Computational Science	F					29.9	14.8
8	Swiss National Supercomputing Center CSCS	Alps, HPE					7.12	61.0
9	EuroHPC /CSC	LUMI, HPE Cray EX235a, AMD 3rd EPYC 64C, 2 GHz, AMD Instinct MI250X, Slingshot 11	Finland	2,752,704	380.	71	7.10	52.3
10	EuroHPC/CINECA	Leonardo, BullSequana XH2000, Xeon Platinum 8358 32C, 2.6GHz, NVIDIA A100 (108C), Quad-rail NVIDIA HDR100	Italy	1,824,768	241.	78	7.49	32.2



Performance Development of HPC over the Last 33 Years from the Top500



El Capatain, LLNL
1.7 EFlop/s

2.4 PFlop/s
HPE, ARL
Intel w/3696 nodes, @20 cores

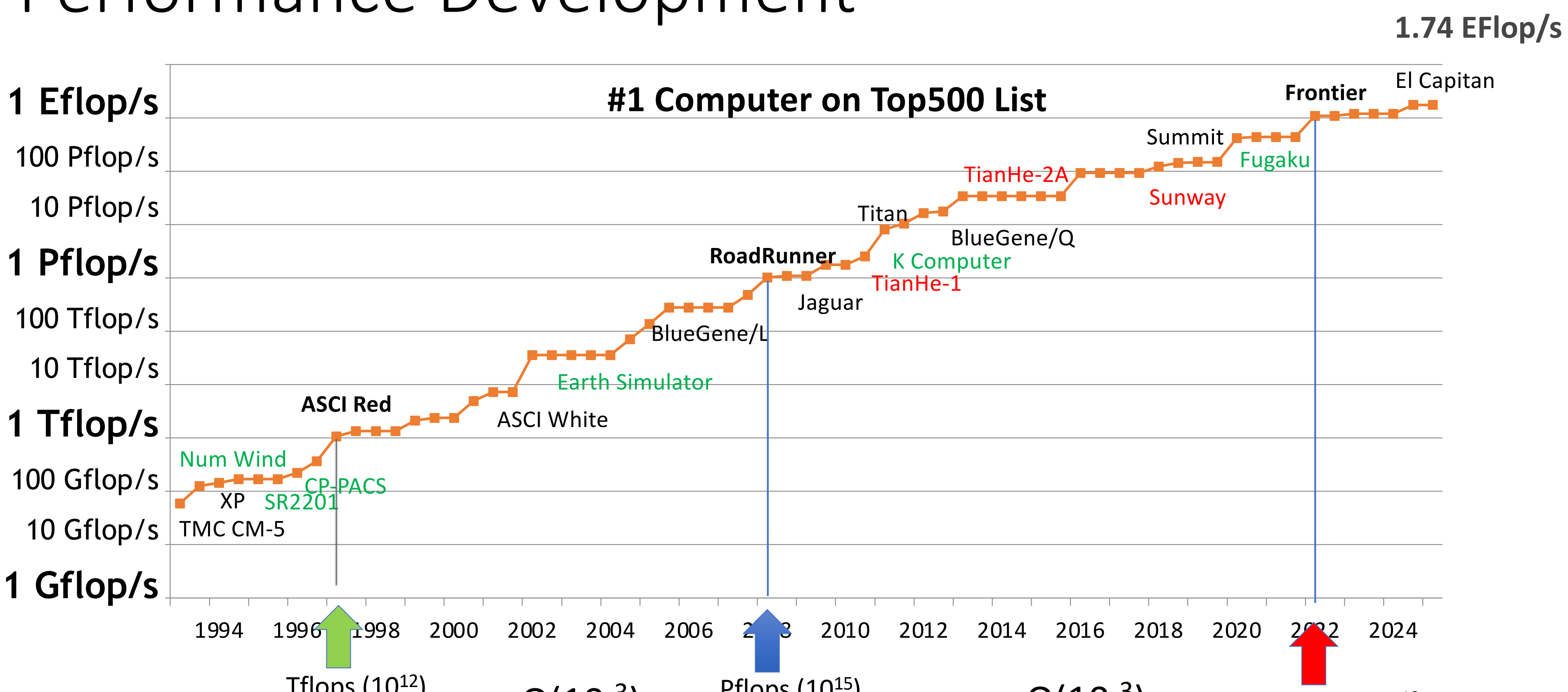
My Laptop: 426 Gflop/s

60 GFlop/s

420 MFlop/s

# 1 in 1993 - Thinking Machine CM-5 with 1024 Processors at Los Alamos Nat Lab used for nuclear weapons design

Performance Development



Tflops (10^{12})
Achieved
ASCI Red
Sandia NL

O(10^{3})
11 Years

Pflops (10^{15})
Achieved
RoadRunner
Los Alamos NL

O(10^{3})
14 Years
It’s taking longer to get O(10^{3})

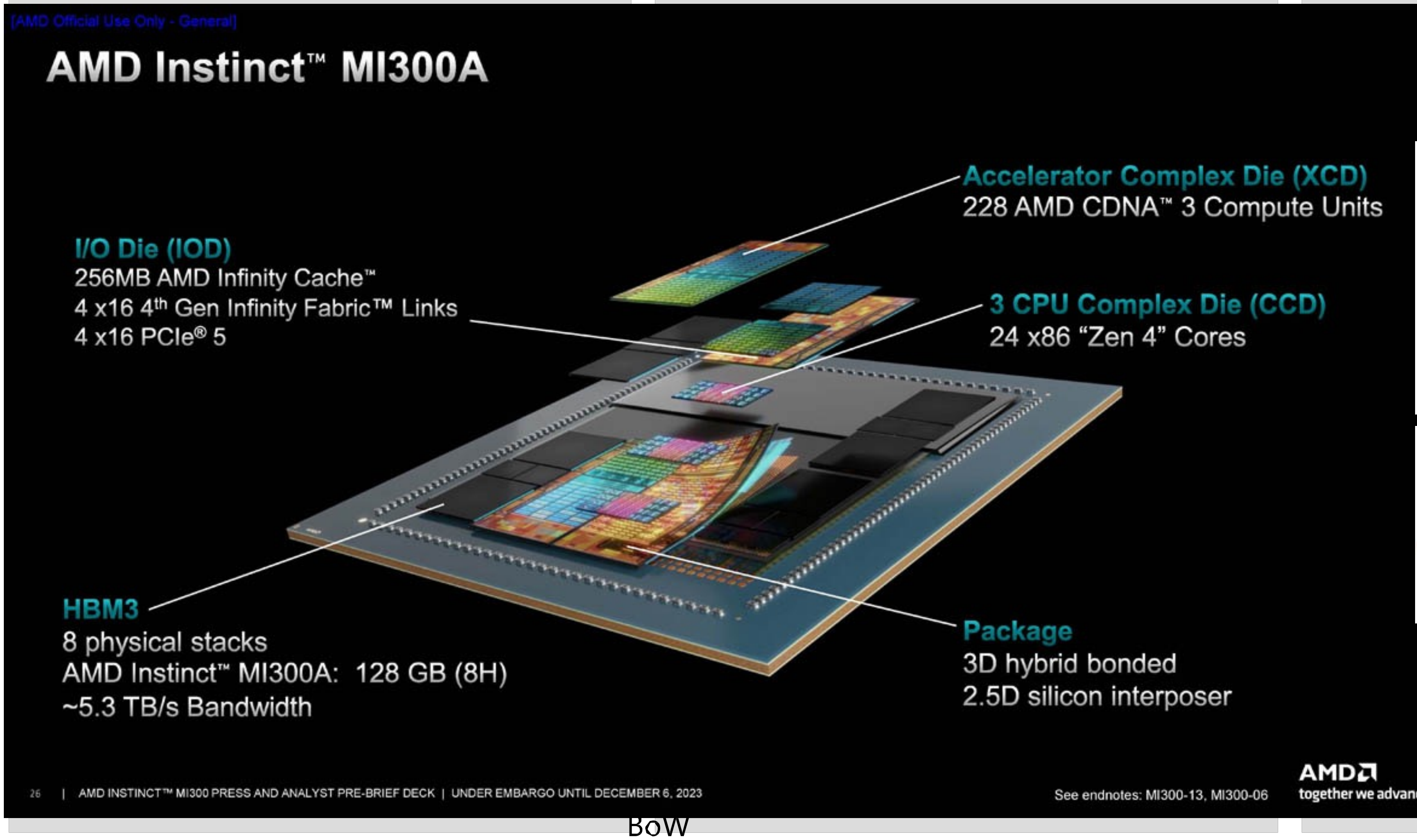
Eflops (10^{18})
Achieved
Frontier ORNL

El Capitan Current #1 System Overview

System Performance

Each node has

The system includes



[AMD Official Use Only - General]

AMD Instinct™ MI300A

I/O Die (IOD)
256MB AMD Infinity Cache™
4 x16 4th Gen Infinity Fabric™ Links
4 x16 PCIe® 5

Accelerator Complex Die (XCD)
228 AMD CDNA™ 3 Compute Units

3 CPU Complex Die (CCD)
24 x86 “Zen 4” Cores

HBM3
8 physical stacks
AMD Instinct™ MI300A: 128 GB (8H)
~5.3 TB/s Bandwidth

Package
3D hybrid bonded
2.5D silicon interposer

26 | AMD INSTINCT™ MI300 PRESS AND ANALYST PRE-BRIEF DECK | UNDER EMBARGO UNTIL DECEMBER 6, 2023

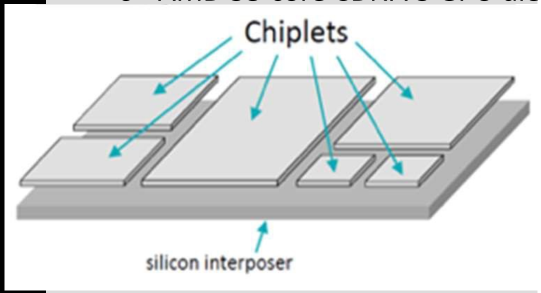
See endnotes: MI300-13, MI300-06

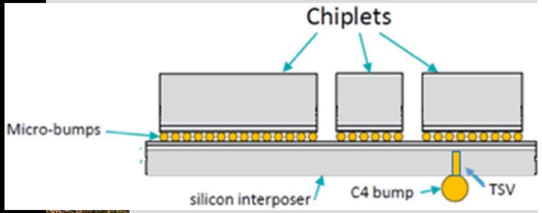
AMD
together we advance_

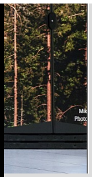
11,136 nodes

3 - 8-core “Zen 4” CPU dies

6 - AMD 38-core CDNA 3 GPU die





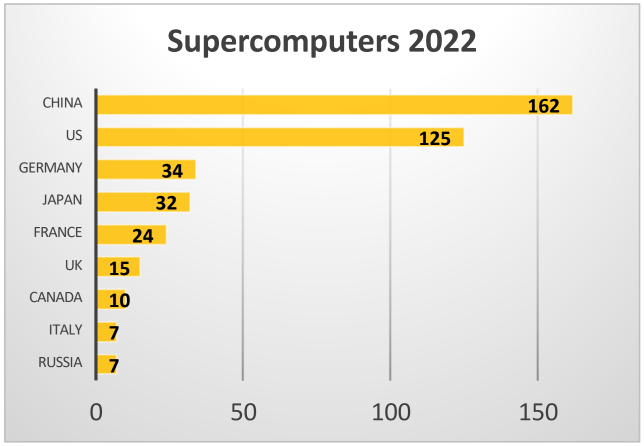


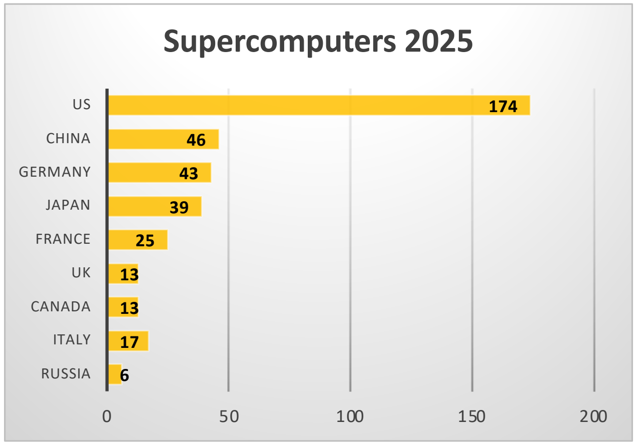
BoW

Rumored to be 3-4 Exascale Systems in China

- In the US, El Capitan, Frontier, and Aurora systems remain the only exascale systems on the Top500

- China stopped its submissions to the Top500





Elon Musk's xAI Colossus System Used for Training Grok, Musk’s LLM for Their Chatbot for X/Twitter

- Built on Nvidia’s H100
    - 67 Tflop/s each 64 bit fl pt
        - 990 Tflop/s 16 bit fl pt
        - 1980 Tflop/s 8 bit fl pt
    - 64 GPUs + 16 CPUs / rack
        - 2 CPUs for 8 GPUs
    - 8 racks / group (512 GPUs)
    - 1,500 racks in total
    - Integrated by Super Micro
        - Ethernet 400 Gb/s
- 200,000 Nvidia’s H100
    - 13.4 Eflop/s 64 bit fl pt
        - 200 Eflop/s 16 bit fl pt
        - .4 Zflop/s 8 bit fl pt 10^{21} Ops/s
    - $3-4B Cost
    - 300 MW Power
        - 2800 m^2

Under Development: Colossus-2 first gigawatt-scale AI training cluster. Host 550,000 Nvidia GB200 GPUs; $30–45 billion

1 MW = 750 homes; 300 MW = 225,000 homes (In TN - 1 MW-Year ~ $1M)

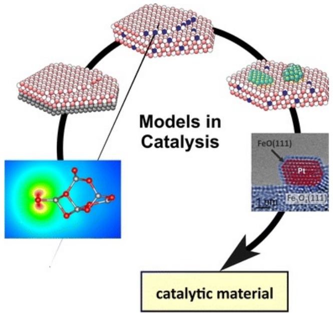
Performance and Benchmarking Evaluation Tools

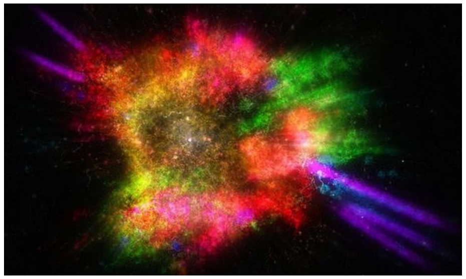
- **Linpack Benchmark - Longstanding benchmark started in 1979**
  - **Lots of positive features; easy to understand and run; shows trends**
- **However, much has changed since 1979**
  - **Arithmetic was expensive then and today it is over-provisioned and inexpensive**
- **Linpack performance of computer systems is no longer strongly correlated to real application performance**
  - **Linpack benchmark based on dense matrix multiplication**
  - **Not “typical” of scientific HPC applications, distorts the field**
- **Designing a system for good Linpack performance can lead to design choices that are wrong for today’s applications**

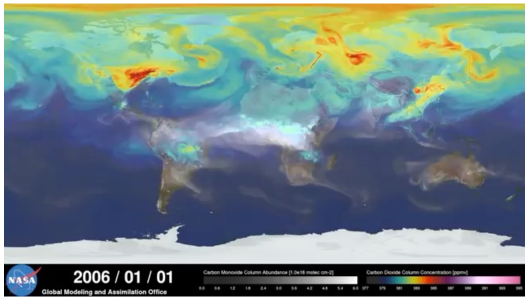
Today’s Top HPC Systems Used to do Simulations

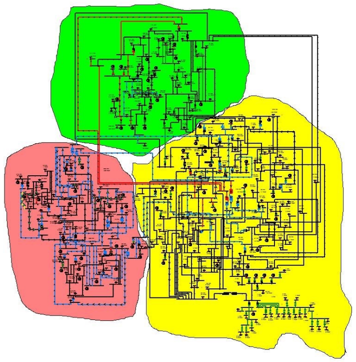
- *Climate*
- *Combustion*
- *Nuclear Reactors*
- *Catalysis*
- *Electric Grid*
- *Fusion*
- *Stockpile*
- *Supernovae*
- *Materials*
- *Digital Twins*
- *Accelerators*
- …

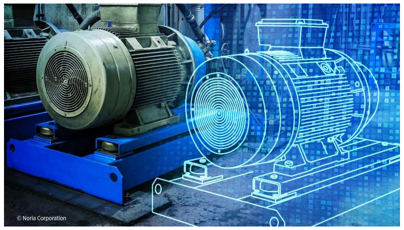
- Usually 3-D PDE’s
    - Sparse matrix computations, not dense

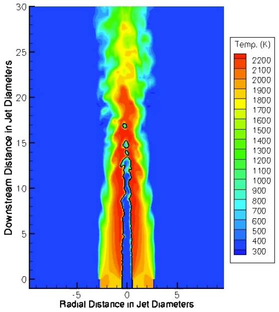


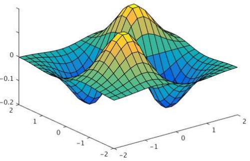


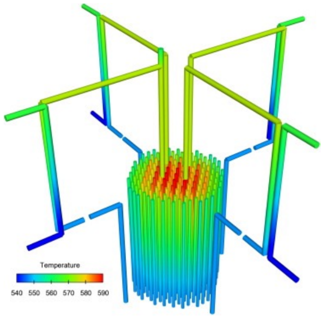


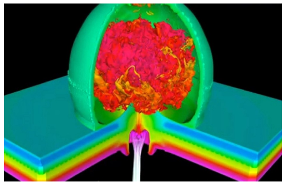


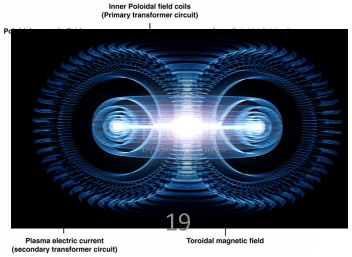








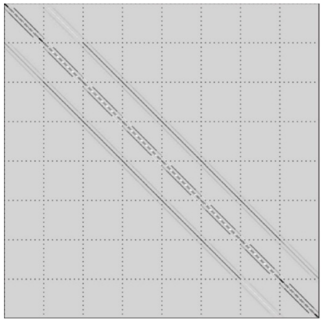



Inner Poloidal field coils
(Primary transformer circuit)
Plasma electric current
(secondary transformer circuit)
Toroidal magnetic field

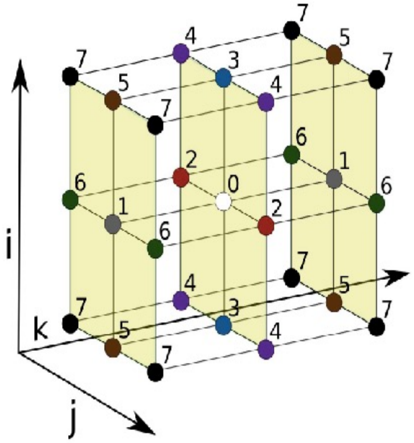
hpcg-benchmark.org With Piotr Luszczek and Mike Heroux

HPCG Results; The Other Benchmark

- High Performance Conjugate Gradients (HPCG).
- Solves *Ax=b, A* large, sparse, *b* known, *x* computed.
- An optimized implementation of PCG contains essential computational and communication patterns that are prevalent in a variety of methods for discretization and numerical solution of PDEs



- Patterns:
  - Dense and sparse computations.
  - Dense and sparse collectives.
  - Multi-scale execution of kernels via MG (truncated) V cycle.
  - Data-driven parallelism (unstructured sparse triangular solves).
- Strong verification (via spectral properties of PCG).



27-point stencil operator

HPCG Top 10, June 2025

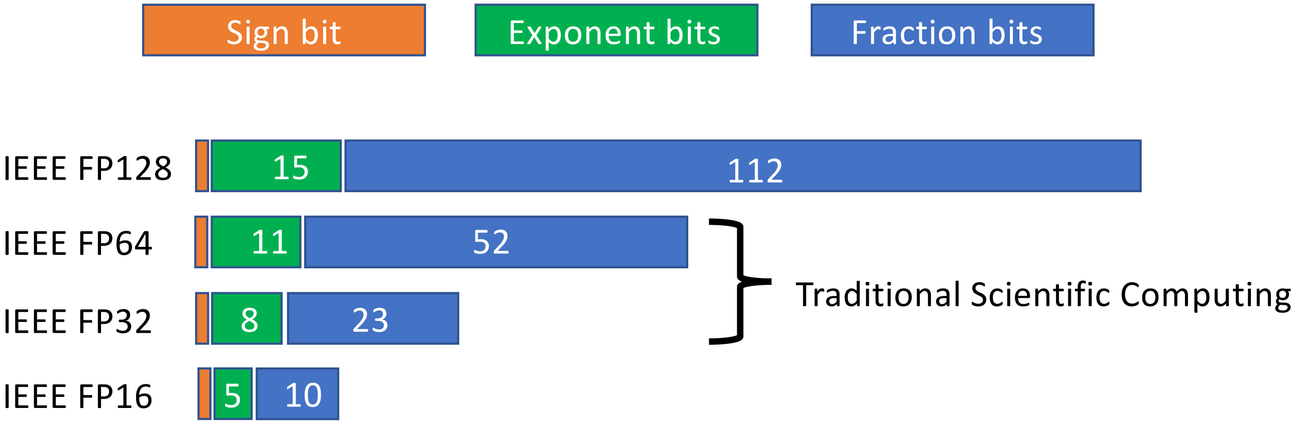
Ax=b Dense A

Ax=b Sparse A

Rank	Site	Computer	Cores	HPL Rmax (Pflop/s)	TOP500 Rank	HPCG (Pflop/s)	Fraction of Peak HPCG
1	DOE/SC/LLNL **USA**	**El Capitan**, HPE Cray 255a, AMD 4th Gen EPYC 24C 1.8 GHz, AMD Instinct MI300A, Slingshot-11	11,039,616	1742	1	17.4	0.6%
2	RIKEN Center for Computational Science **Japan**	**Fugaku**, Fujitsu A64FX 48C 2.2GHz, Tofu D, Fujitsu	7,630,848	442	7	16.0	**3.0%**
3	DOE/SC/ORNL **USA**	**Frontier,** HPE Cray Ex235a, AMD 3rd EPYC 64C, 2 GHz, AMD Instinct MI250X, Slingshot-11	9,066,176	1353	2	14.1	0.7%
4	DOE/SC/ANL **USA**	**Aurora,** HPE Cray EX, Intel Max 9470 52C, 2.4 GHz, Intel GPU MAX, Slingshot-11	9,264,128	1012	3	5.6	0.3%
5	EuroHPC/CSC **Finland**	**LUMI**, HPE Cray EX235a, AMD Zen-3 (Milan) 64C 2GHz, AMD MI250X, Slingshot-11	2,752,704	380	9	4.6	0.9%
6	CSCS **Switzerland**	**Alps**, HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11	2,121,600	435	8	3.7	0.6%
7	EuroHPC/CINECA **Italy**	**Leonardo**, BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 40 GB, Quad-rail NVIDIA HDR100 Infiniband	1,824,768	241	10	3.1	1.0%
8	AIST [illegible]	**ABCI 3.0**, HPE Cray XD670, Xeon Platinum 8558 48C 2.1GHz, NVIDIA H200 SXM5 141 GB, Infiniband	479,232	145	15	2.4	1.3%
9	DOE/SC/LBNL **USA**	**Perlmutter**, HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10	888,832	79	25	1.9	[illegible]%
10	DOE/NNSA/LLNL **USA**	**Sierra**, S922LC, IBM POWER9 20C 3.1 GHz, Mellanox EDR, NVIDIA Volta V100, IBM	1,572,480	95	20	1.8	**1.4%**

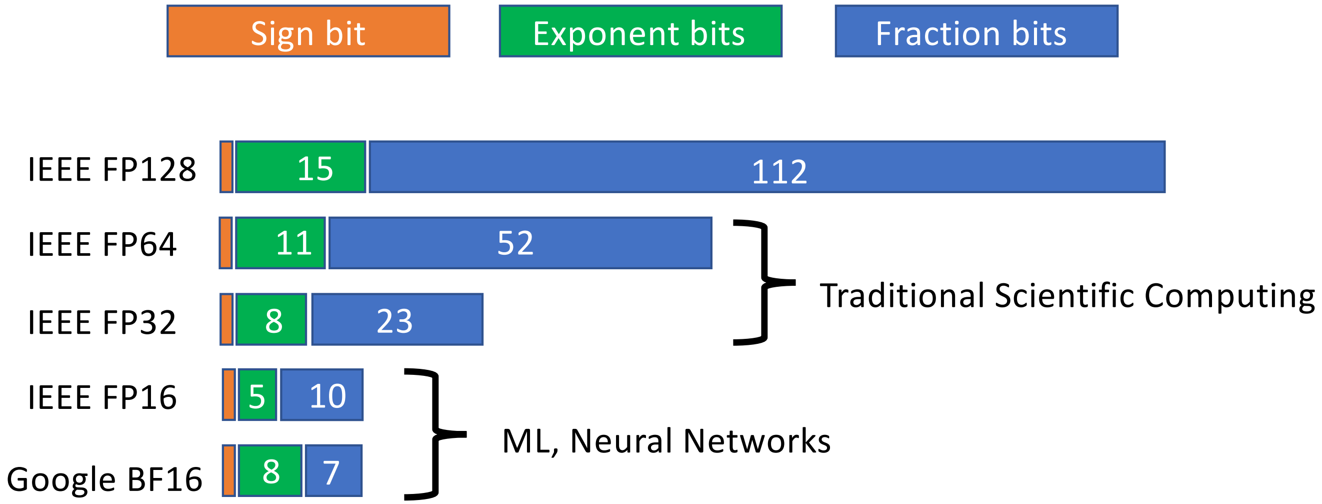
Think of a race car that has the potential of 200 MPH but only goes 2 MPH!

Floating Point Representation



Can we leverage the short precision in our "traditional" scientific numerical computations?

Floating Point Representation



Sign bit

Exponent bits

Fraction bits

IEEE FP128 15 112

IEEE FP64 11 52

IEEE FP32 8 23

Traditional Scientific Computing

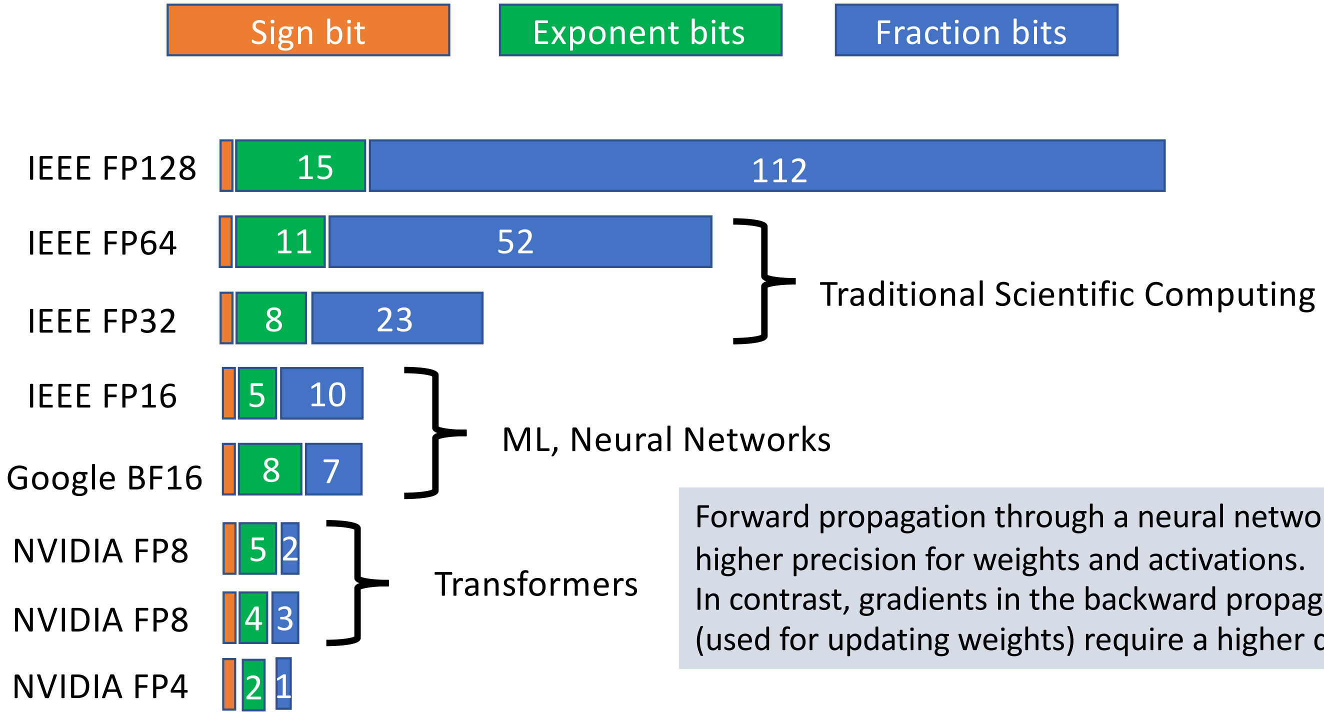
IEEE FP16 5 10

Google BF16 8 7

ML, Neural Networks

Can we leverage the short precision in our “traditional” scientific numerical computations?

Floating Point Representation



Forward propagation through a neural network requires higher precision for weights and activations.
In contrast, gradients in the backward propagation (used for updating weights) require a higher dynamic range.

Can we leverage the short precision in our “traditional” scientific numerical computations?

“Responsibly Reckless” Algorithms

- Try a fast algorithm (that may be unstable) and might fail (but rarely)
  - Avoiding Data Movement
  - Avoiding Synchronization
  - Use Mixed Precision
- Check for instability
- If needed, recompute with a stable algorithm

WHY MIXED PRECISION? (Less is Faster)

- **There are many reasons to consider using mixing precisions within an application:**
  - Less Communication
    - Reduce memory traffic (from memory to processor)
    - Reduce network traffic (from node to node)
  - **Reduce memory footprint (less data to store*)**
  - **Arithmetic faster (usually factor of 2 or more)**
    - **Lower precision is usually faster than high precision operations**
    - **Architecture may have an accelerator**
  - **Suitable numerical properties for the algorithm & problems.**

The hope is to improve the algorithm performance without compromising the quality of science

Leveraging Mixed Precision for Linear Algebra

Idea: use low precision to compute the expensive flops (LU $O(n^3)$) and then iteratively refine ($O(n^2)$) the solution in order to achieve the FP64 arithmetic

Iterative refinement for dense systems, $Ax = b$, can work this way.

```
L U = lu(A)                                                    lower precision   $O(n^3)$
x = U\(L\b)                                                    lower precision   $O(n^2)$
r = b – Ax (with original A)                                   FP64 precision    $O(n^2)$

WHILE || r || not small enough
    1.  find a correction “z” to adjust x that satisfy Az=r
        solving Az=r could be done by either:
          ➢ GMRes preconditioned by the LU to solve Az=r Iterative Refinement GMRes   lower precision   $O(n^2)$
    2.  x = x + z                                              FP64 precision    $O(n^1)$
    3.  r = b – Ax (with original A)                           FP64 precision    $O(n^2)$
END
```

Higham and Carson showed can solve the inner problem with iterative method and not infect the solution with the conditioning of the original matrix.

J. Langou, et al., Exploiting the Performance of 32 bit fl-pt Arithmetic in Obtaining 64 bit Accuracy, in: Proc. of SC06

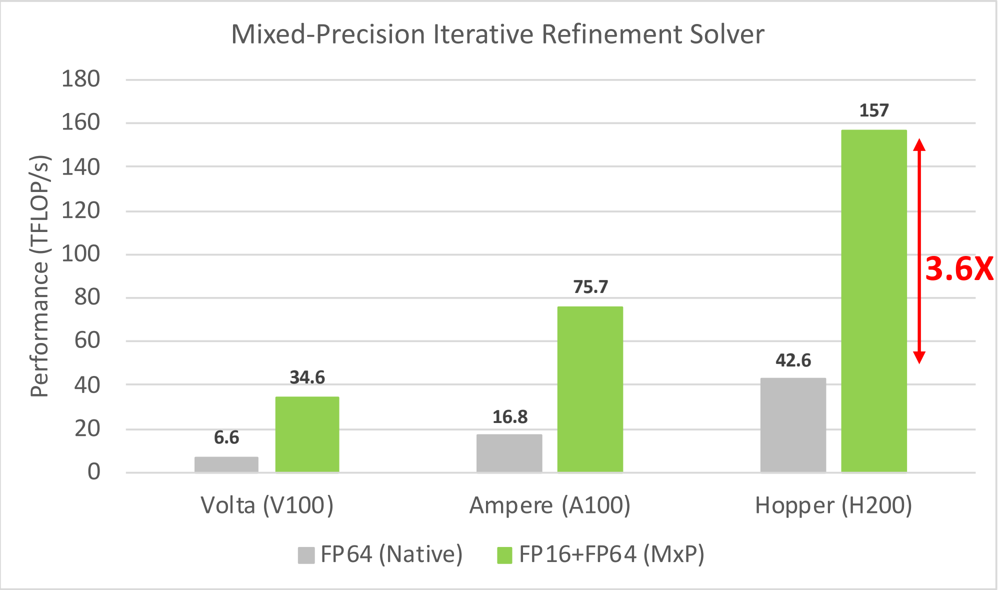
E. Carson & N. Higham, “Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions *SIAM J. Sci. Comput.*, 40(2), A817–A847.

Originally motivated by the Sony PlayStation
SP peak 205 Gflop/s, DP peak 15 Gflop/s

➢ Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt

Mixed-Precision Iterative Refinement Solver

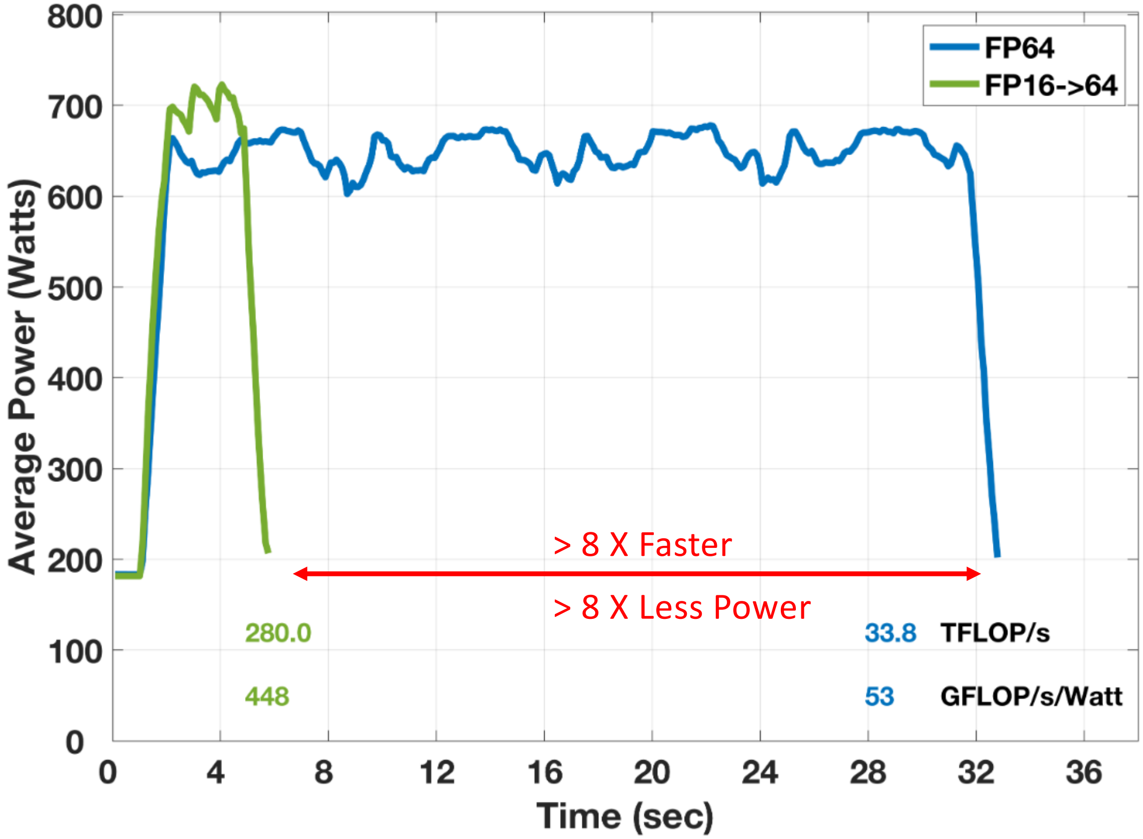
Performance and Efficiency Improvements Across Three Generations



■ FP64 (Native) 🟩 FP16 & FP32 & FP64 (MxP)

32k matrix size solution

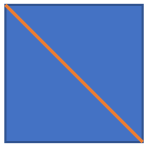
NVIDIA Blackwell B200 GPU



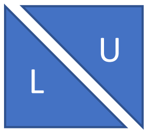
> 8 X Faster

> 8 X Less Power

HPL-MxP Benchmark Utilizing 16-bit Arithmetic



1. Generate random linear system Ax=b
2. Represent the matrix A in low precision (16-bit floating point)
3. Factor A in lower precision into LU by Gaussian elimination



4. Compute approximate solution with LU factors in low precision
5. Perform a few iterations of refinement, e.g., GMRES to get accuracy up to 64-bit floating point
   a. Use LU factors for preconditioning

Iterative refinement for dense systems, *Ax = b*, can work this way.

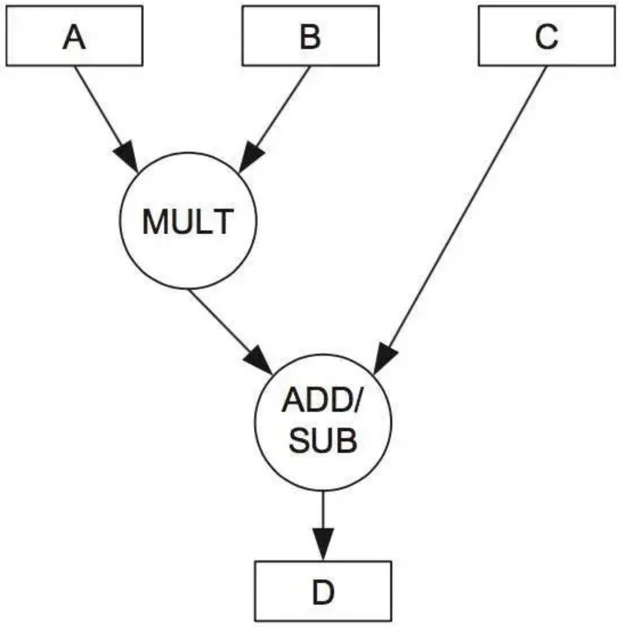
L U = lu(A)	Lower precision	$O(n^3)$
x = U\(L\b)	Lower precision	$O(n^2)$
GMRes preconditioned by the LU to solve Ax=b	FP64 precision	$O(n^2)$

6. Validate the answer is correct: scaled residual small $\frac{||Ax-b||}{||A||||x||+||b||} \times \frac{1}{n\epsilon} \leq O(10)$
7. Compute performance rate as $\frac{2}{3} \times \frac{n^3}{\text{time}}$

HPL-MxP Top 10 for June 2025

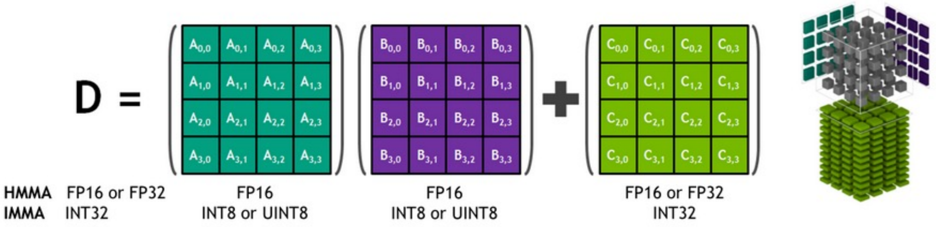
Rank	Site	Computer	Cores	HPL Rmax (Eflop/s)	TOP500 Rank	HPL-MxP (Eflop/s)	Speedup
1	DOE/SC/LLNL **USA**	**El Capitan**, HPE Cray 255a, AMD 4th Gen EPYC 24C 1.8 GHz, AMD Instinct MI300A, Slingshot-11	11,039,616	1.742	1	16.7	9.6
2	DOE/SC/ANL **USA**	**Aurora**, HPE Cray EX, Intel Max 9470 52C, 2.4 GHz, Intel GPU MAX, Slingshot-11	8,159,232	1.012	3	11.6	11.5
3	DOE/SC/ORNL **USA**	**Frontier**, HPE Cray EX235a, AMD Zen-3 (Milan) 64C 2GHz, AMD MI250X, Slingshot-11	8,560,640	1.353	2	11.4	8.4
4	AIST **Japan**	**ABCI 3.0**, HPE Cray XD670, Xeon Platinum 8558 48C 2.1GHz, NVIDIA H200 SXM5 141 GB, Infiniband NDR200, HPE	479,232	0.145	15	2.36	16.3
5	EuroHPC/CSC **Finland**	**LUMI**, HPE Cray EX235a, AMD Zen-3 (Milan) 64C 2GHz, AMD MI250X, Slingshot-11	2,752,704	0.380	9	2.35	6.2
6	RIKEN Center for Computational Science, **Japan**	**Fugaku**, Fujitsu A64FX 48C 2.2GHz, Tofu D	7,630,848	0.442	7	2.0	4.5
7	EuroHPC/CINECA **Italy**	**Leonardo**, BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 40 GB, Quad-rail NVIDIA HDR100 Infiniband	1,824,768	0.241	10	1.8	7.6
8	CII, Institute of Science **Japan**	**TSUBAME 4**, HPE Cray XD665, AMD EPYC 9654 96C 2.4GHz, NVIDIA H100 SXM5 94 GB, Mellanox NDR200	172,800	0.035	46	0.64	16.2
9	NVIDIA **USA**	**Selene**, DGX SuperPOD, AMD EPYC 7742 64C 2.25 GHz, Mellanox HDR, NVIDIA A100	555,520	0.063	30	0.63	9.9
10	DOE/SC/LBNL/NERSC **USA**	**Perlmutter**, HPE Cray EX235n, AMD EPYC 7763 64C 2.45 GHz, Slingshot-10, NVIDIA A100	761,856	0.079	25	0.59	7.5

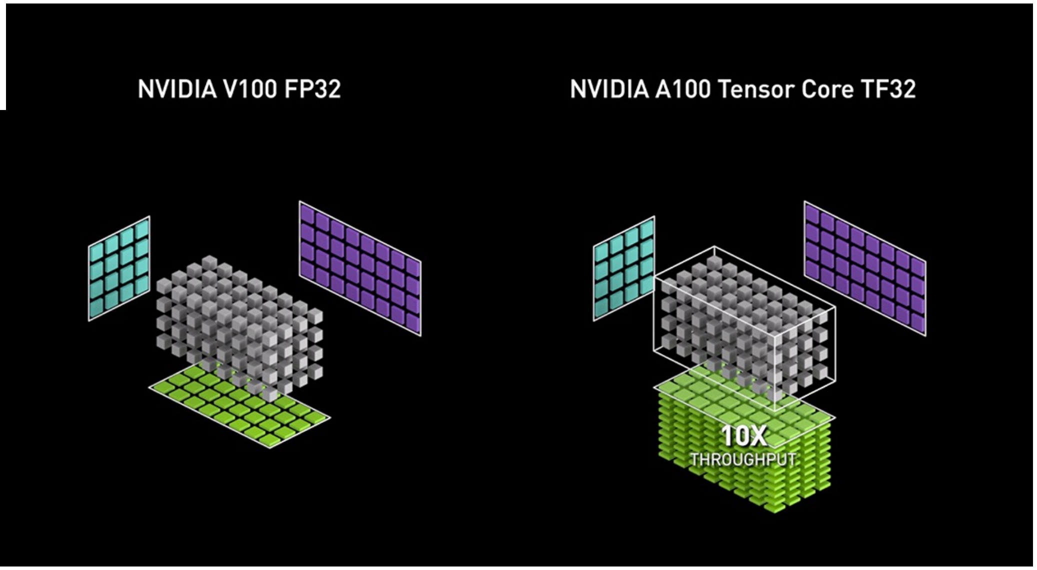
Conventional Computing FMA Operation



GPUs and Tensor Core Operations (Think Matrix Multiply)

- Tensor cores are specialized hardware units within GPUs, designed to accelerate matrix operations





Recent Nvidia GPUs

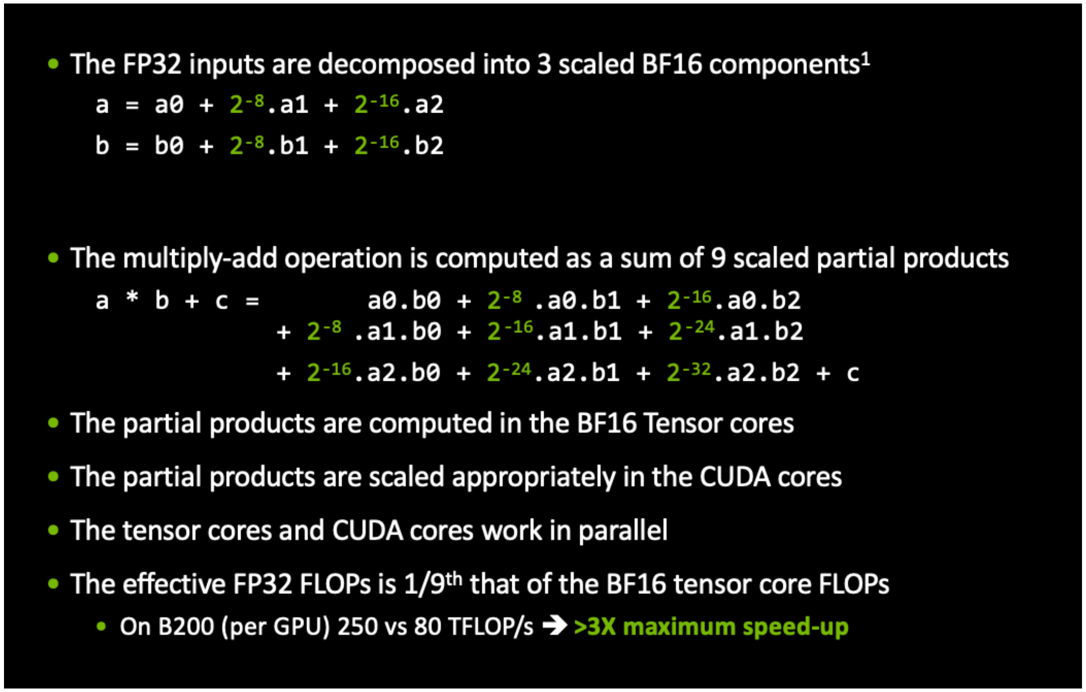
	Figure of Merit Peak Performance	
Operations	2022 Hopper (H200)	2024 Blackwell (B200)
FP64 FMA	33.5 Tflop/s	40 Tflop/s
FP64 Tensor Core	67 Tflop/s	40 Tflop/s
FP32 FMA	67 Tflop/s	80 Tflop/s
FP16 Tensor Core	989 Tflop/s	2250 Tflop/s
BF16 Tensor Core	989 Tflop/s	2250 Tflop/s
INT8 Tensor Core	1979 TOP/s	4500 TOP/s
Memory BW	4.8 TB/s	8 TB/s

112X

Opportunity Breeds Innovation

$$\begin{aligned} d &= a \cdot b + c \\ &= (a_0 + 2^{-8}a_1 + 2^{-16}a_2) \cdot (b_0 + 2^{-8}b_1 + 2^{-16}b_2) + c \\ &= \quad a_0b_0 + 2^{-8}a_0b_1 \quad + 2^{-16}a_0b_2 \\ &\quad 2^{-8}a_1b_0 + 2^{-16}a_1b_1 \quad + 2^{-24}a_1b_2 \\ &\quad 2^{-16}a_2b_0 + 2^{-24}a_2b_1 \quad + 2^{-32}a_2b_2 + c \end{aligned}$$

Divide the numbers into “slices” of 2^{-8}



- The FP32 inputs are decomposed into 3 scaled BF16 components[1]

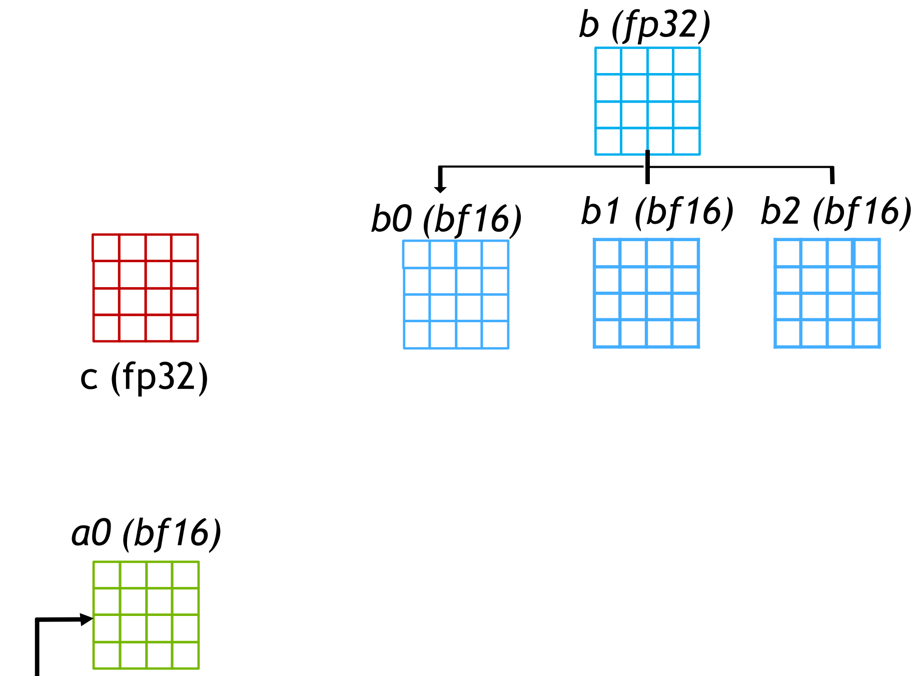
$$a = a0 + 2^{-8}.a1 + 2^{-16}.a2$$

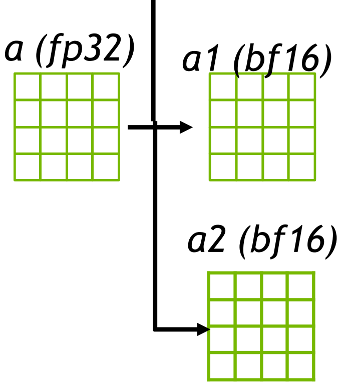
$$b = b0 + 2^{-8}.b1 + 2^{-16}.b2$$

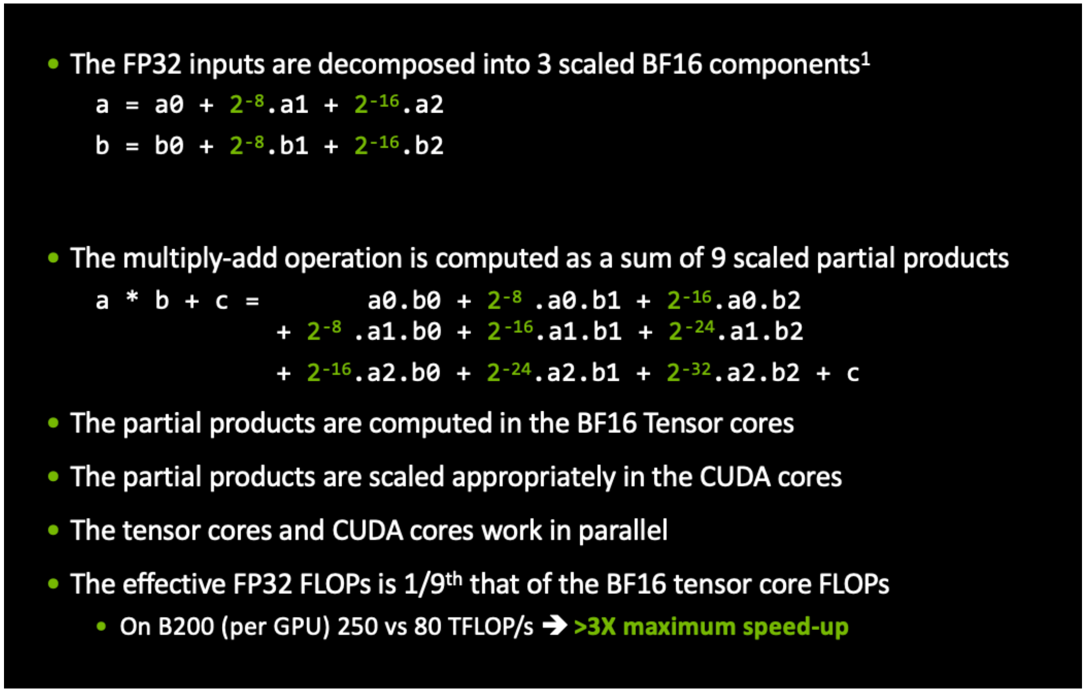
- The multiply-add operation is computed as a sum of 9 scaled partial products

$$\begin{aligned} a * b + c = {} & a0.b0 + 2^{-8}.a0.b1 + 2^{-16}.a0.b2 \\ & + 2^{-8}.a1.b0 + 2^{-16}.a1.b1 + 2^{-24}.a1.b2 \\ & + 2^{-16}.a2.b0 + 2^{-24}.a2.b1 + 2^{-32}.a2.b2 + c \end{aligned}$$

- The partial products are computed in the BF16 Tensor cores
- The partial products are scaled appropriately in the CUDA cores
- The tensor cores and CUDA cores work in parallel
- The effective FP32 FLOPs is 1/9th that of the BF16 tensor core FLOPs
  - On B200 (per GPU) 250 vs 80 TFLOP/s ➔ **>3X maximum speed-up**







- The FP32 inputs are decomposed into 3 scaled BF16 components[1]

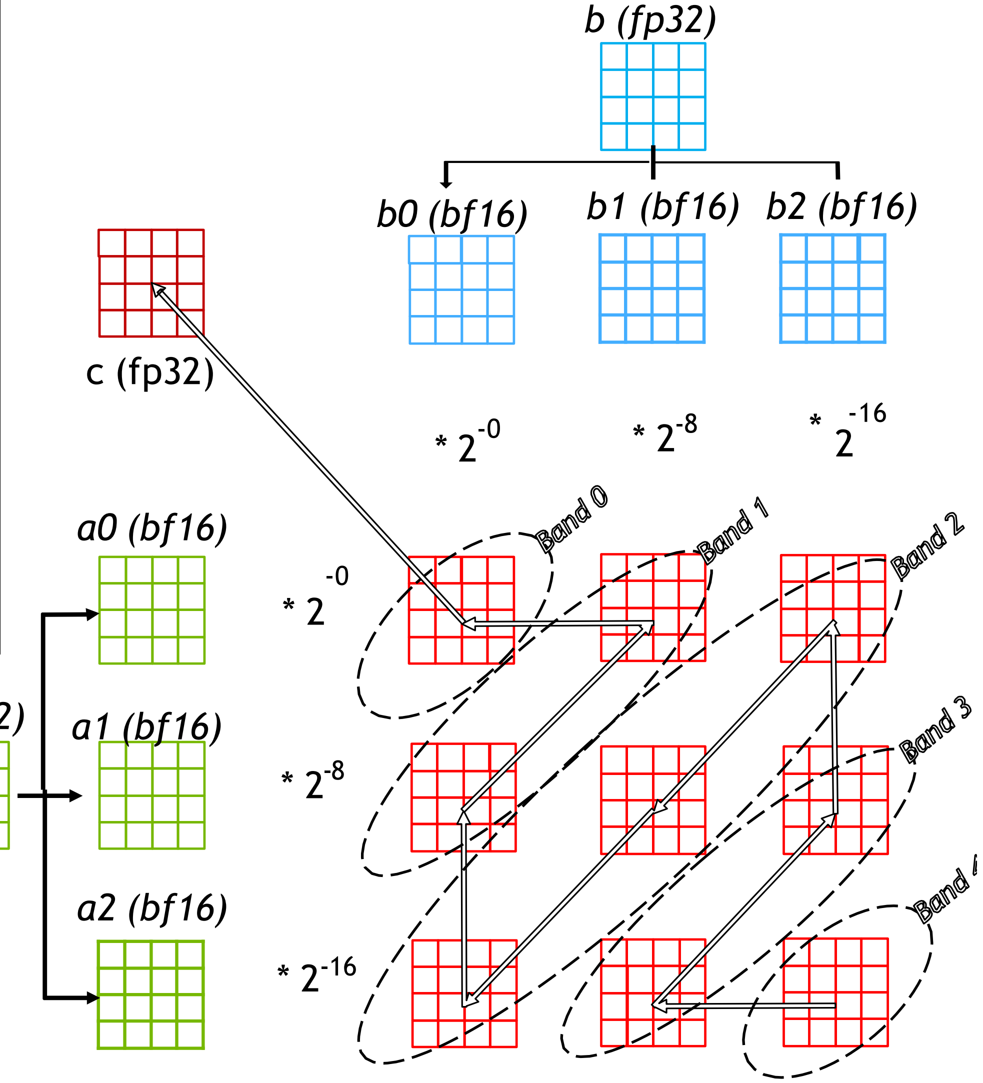
  $a = a0 + 2^{-8}.a1 + 2^{-16}.a2$

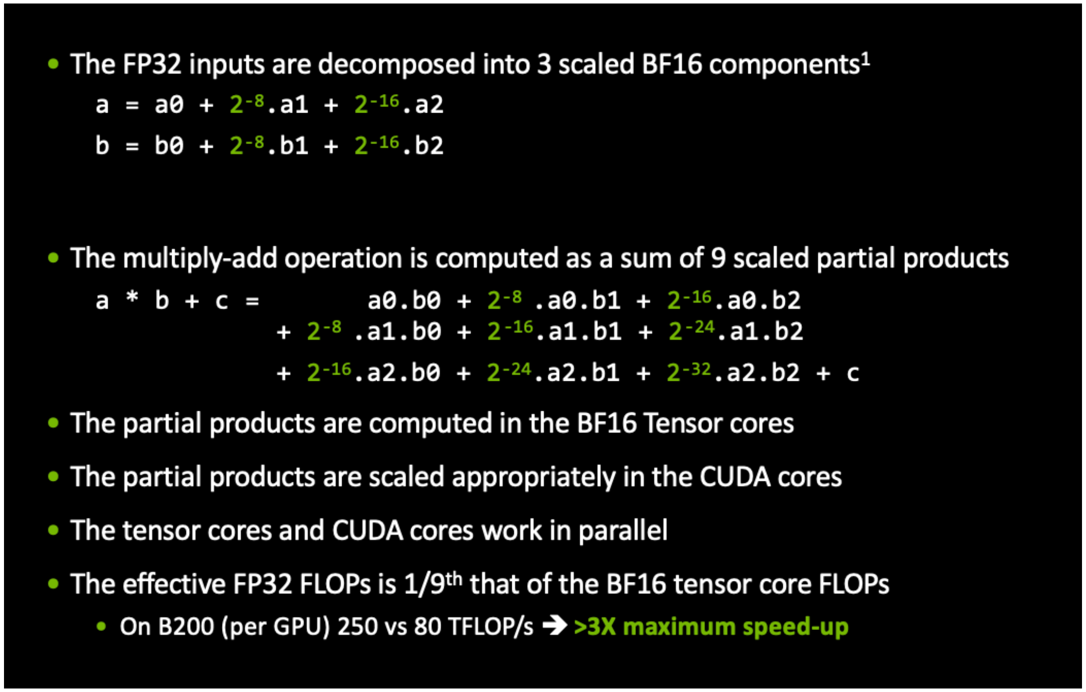
  $b = b0 + 2^{-8}.b1 + 2^{-16}.b2$

- The multiply-add operation is computed as a sum of 9 scaled partial products

  $$\begin{aligned} a * b + c = {} & a0.b0 + 2^{-8}.a0.b1 + 2^{-16}.a0.b2 \\ & + 2^{-8}.a1.b0 + 2^{-16}.a1.b1 + 2^{-24}.a1.b2 \\ & + 2^{-16}.a2.b0 + 2^{-24}.a2.b1 + 2^{-32}.a2.b2 + c \end{aligned}$$

- The partial products are computed in the BF16 Tensor cores
- The partial products are scaled appropriately in the CUDA cores
- The tensor cores and CUDA cores work in parallel
- The effective FP32 FLOPs is 1/9th that of the BF16 tensor core FLOPs
  - On B200 (per GPU) 250 vs 80 TFLOP/s ➔ >3X maximum speed-up





- The FP32 inputs are decomposed into 3 scaled BF16 components[1]

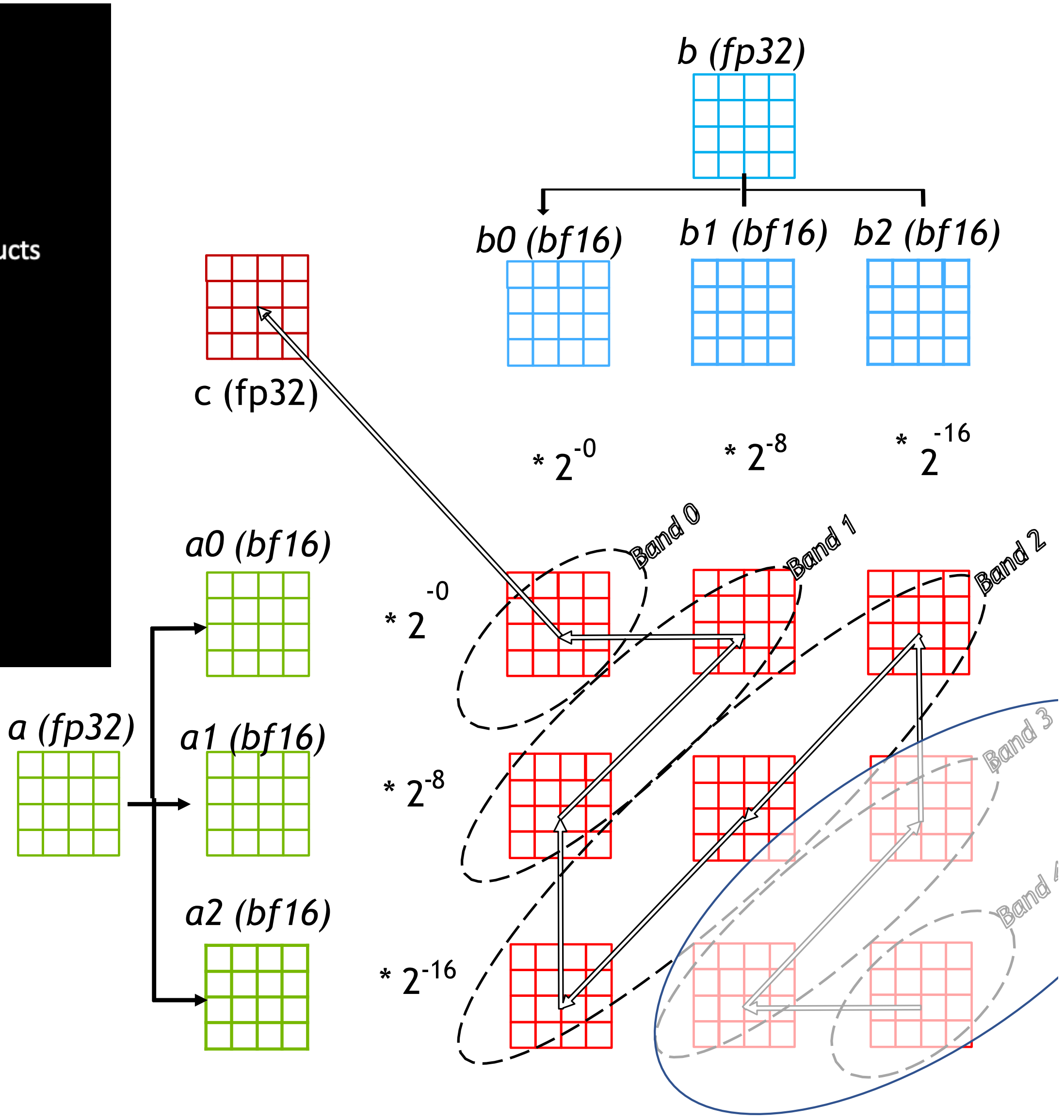
  $a = a0 + 2^{-8}.a1 + 2^{-16}.a2$

  $b = b0 + 2^{-8}.b1 + 2^{-16}.b2$

- The multiply-add operation is computed as a sum of 9 scaled partial products

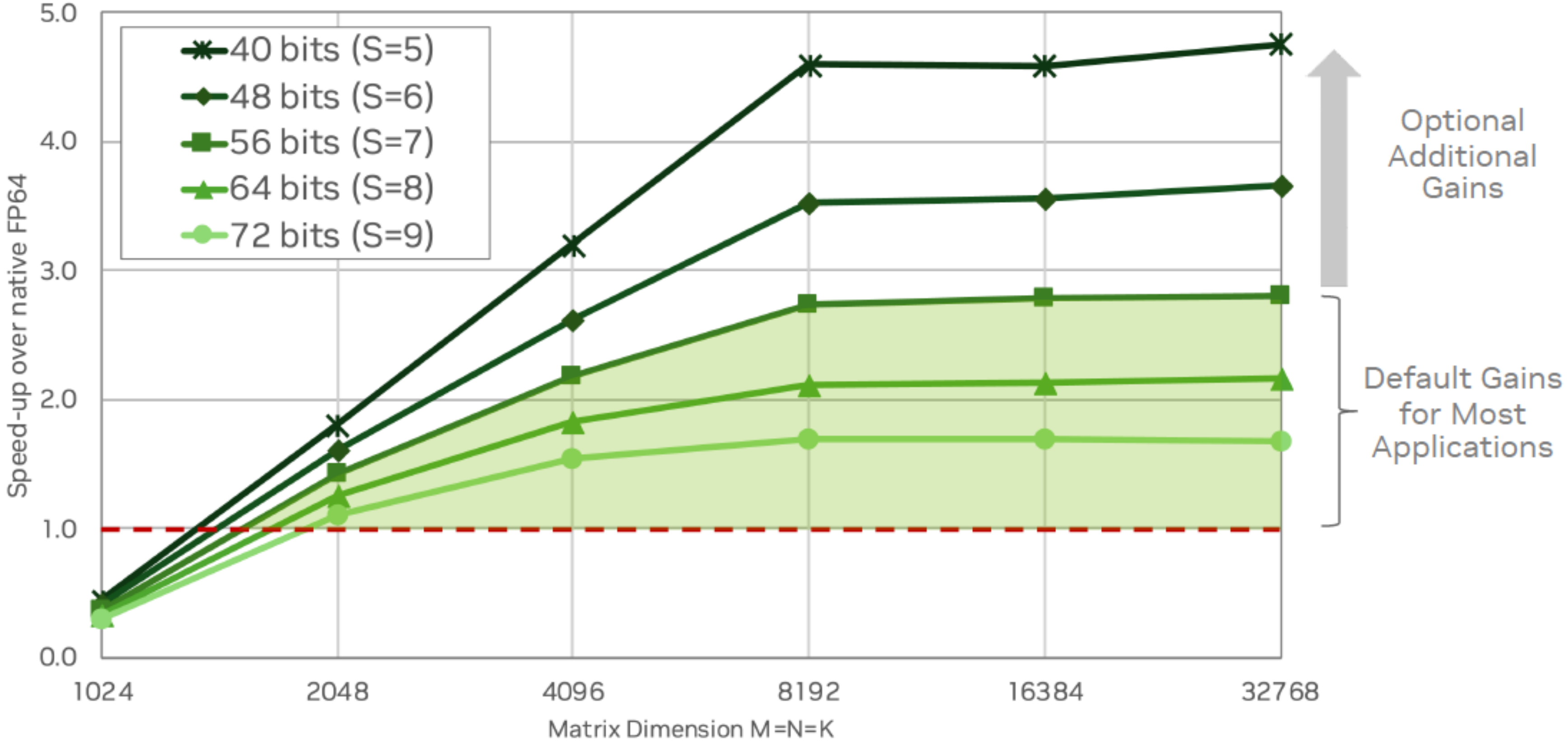
  $$\begin{aligned} a * b + c = {} & a0.b0 + 2^{-8}.a0.b1 + 2^{-16}.a0.b2 \\ & + 2^{-8}.a1.b0 + 2^{-16}.a1.b1 + 2^{-24}.a1.b2 \\ & + 2^{-16}.a2.b0 + 2^{-24}.a2.b1 + 2^{-32}.a2.b2 + c \end{aligned}$$

- The partial products are computed in the BF16 Tensor cores
- The partial products are scaled appropriately in the CUDA cores
- The tensor cores and CUDA cores work in parallel
- The effective FP32 FLOPs is 1/9th that of the BF16 tensor core FLOPs
  - On B200 (per GPU) 250 vs 80 TFLOP/s ➔ **>3X maximum speed-up**



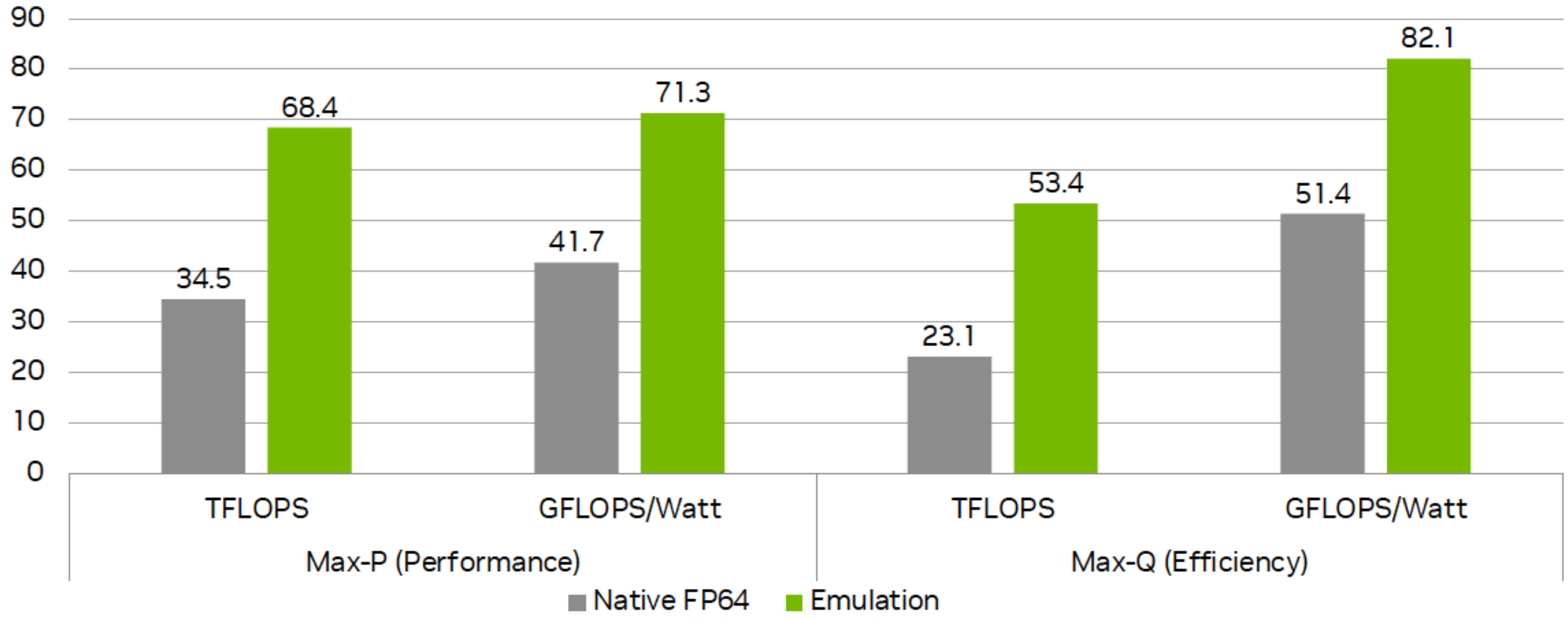
Potential additional efficiency gains

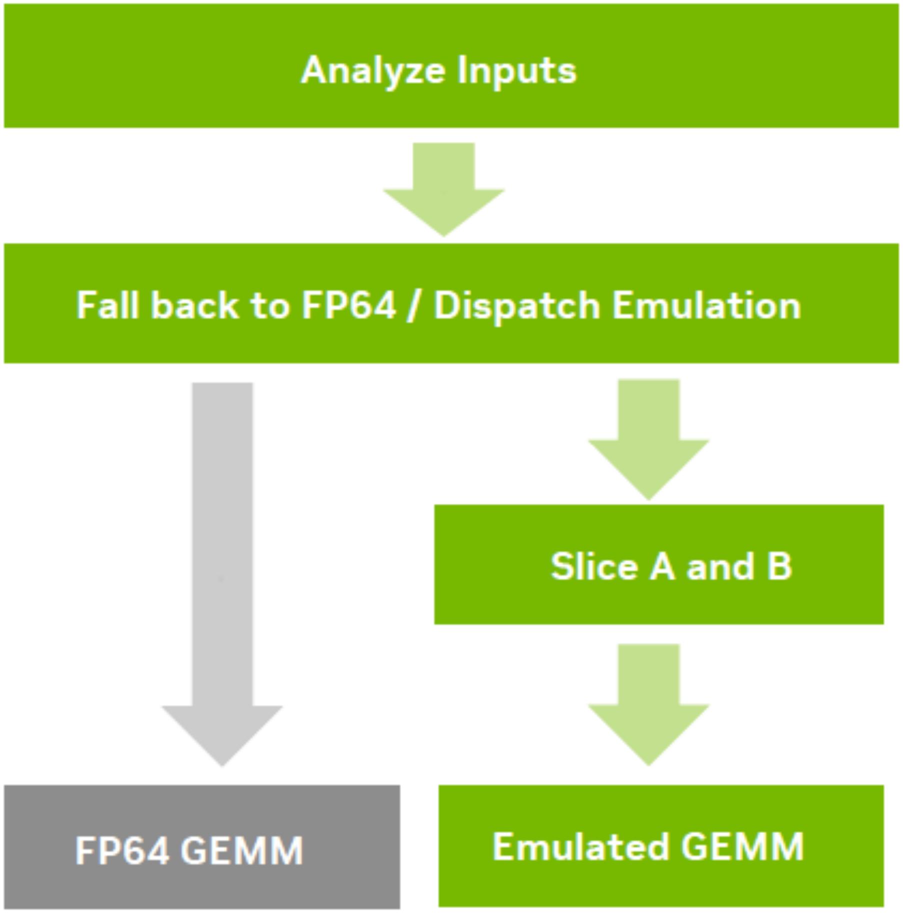
Performance of Emulated GEMM on B200 GPUs for various number of bits used



Performance and Perf/Watt: Emulated vs. Native HPL

At Max-P (Performance) Blackwell HPL runs 2.0x faster and 1.7x more efficiently using emulation (56 bits)
At Max-Q (Efficiency) Blackwell HPL runs 2.3x faster and 1.6x more efficiently using emulation (56 bits)




Analyze Inputs
Fall back to FP64 / Dispatch Emulation
Slice A and B
FP64 GEMM
Emulated GEMM

References: Ozaki’s Method and others

D. Mukunoki, K. Ozaki, T. Ogita, T. Imamura: DGEMM Using Tensor Cores, and Its Accurate and Reproducible Versions, Lecture Notes in Computer Science, 12151, 2020, 230-248.

H. Ootomo, K. Ozaki, R. Yokota: DGEMM on Integer Matrix Multiplication Unit, The International Journal of High Performance Computing Applications, 38 (2024), 297--313. https://github.com/enp1s0/ozIMMU

Y. Uchino, K. Ozaki, T. Imamura: Performance Enhancement of the Ozaki Scheme on Integer Matrix Multiplication Unit, arXiv:2409.13313. https://github.com/RIKEN-RCCS/accelerator_for_ozIMMU42

Lin, Zejia, Sun, AoyuanZha, ng, Xianwei and Lu, Yutong, MixPert: Optimizing Mixed-Precision Floating-Point Emulation on GPU Integer Tensor Cores, Proceedings of the 25th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded System, 2024

Katsuhisa Ozaki, Yuki Uchino, and Toshiyuki Imamura, Ozaki Scheme II: A GEMM-oriented emulation of floating-point matrix multiplication using an integer modular technique, Archive, 2025.

Ahmad Abdelfattahi, Jack Dongarra, Massimiliano Fasi, Mantas Mikaitisi, and Françoise Tisseur, Analysis of Floating-Point Matrix Multiplication Computed via Integer Arithmetic, submitted to SIAM SISC, https://arxiv.org/abs/2506.11277,

- The Ozaki scheme offers an effective method for computing the product of two floating-point matrices on hardware using integer matrix-multiplication units.

- Input matrices are divided into integer slices, multiplied using integer arithmetic, and then converted and accumulated in floating-point.

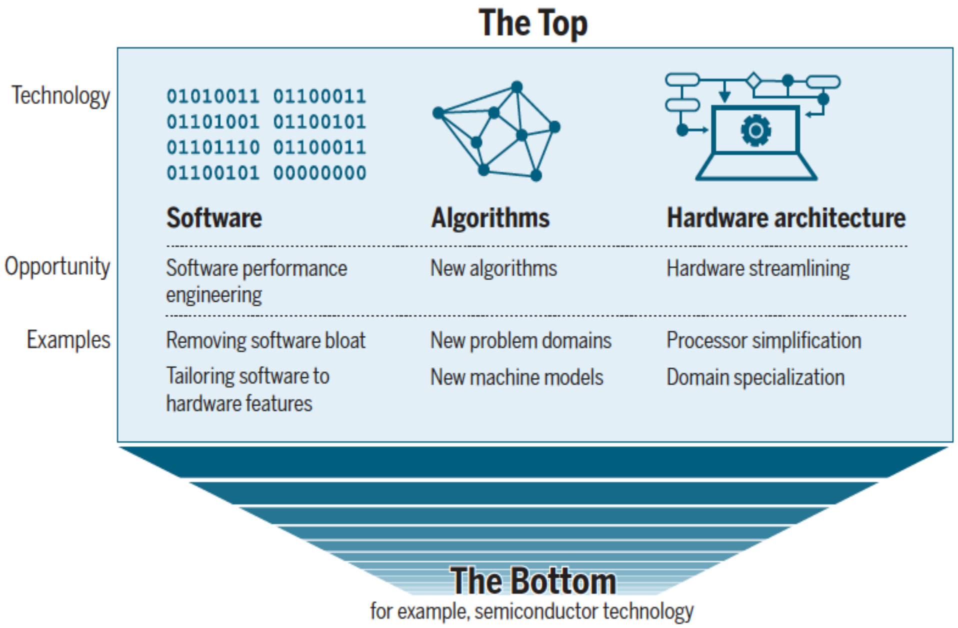
- However, our error analysis shows that the algorithm can become highly inaccurate if the matrices have entries with vastly different magnitudes, even with many slices.

The Take Away

- HPC Hardware is Constantly Changing
  - Scalar
  - Vector
  - Distributed
  - Accelerated
  - Mixed precision
- Algorithm / Software advances follows hardware.
  - And there is “plenty of room at the top”

“There’s plenty of room at the Top: What will drive computer performance after Moore’s law?”

Leiserson *et al.*, *Science* **368**, 1079 (2020)   5 June 2020

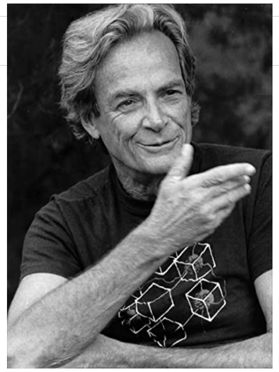


The Top

	Software	Algorithms	Hardware architecture
Technology	01010011 01100011 01101001 01100101 01101110 01100011 01100101 00000000		
Opportunity	Software performance engineering	New algorithms	Hardware streamlining
Examples	Removing software bloat Tailoring software to hardware features	New problem domains New machine models	Processor simplification Domain specialization

The Bottom

for example, semiconductor technology



Feynman’s 1959
Lecture @ CalTech