
# Growing Up at Argonne National Laboratory

Jack Dongarra

University of Tennessee

University of Manchester

# I wanted to be a science high school teacher


- Enrolled as an undergraduate at a college for teachers for the Chicago public school system
- My last semester in college my physics professor encouraged me to apply to a program to spend a semester at Argonne working with a scientist.



Brian Smith

Worked on a software project called EISPACK.

Many visitors from various universities.

Cleve Moler, U of New Mexico

# 1970s HPC Systems

CDC 7600 36.4 MHz (27.5 ns clock cycle)

- Primary memory 65 Kwords (60-bit words)
- Seymour Cray design
- Peak 36 Mflop/s
- Broke down at least once/day (often four or five times)


IBM 370/195 18.5 MHz (54 ns clock cycle)

- High degree of parallelism
- Up to 7 operations at a time
- Up to 4 MB of memory

Both systems had a high degree of instruction-level pipelining and parallelism.

# 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

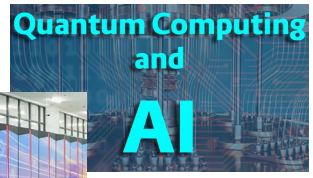
Distributed
Memory

Massively Parallel
Processors
TMC CM-5

x86 Linux Clusters

Accelerated Clusters

ARM w/SVE-SME




**Level 1 Basic Linear Algebra Subprograms (BLAS)**

**LAPACK (1990's)**
**(Blocking, cache friendly)**

**LINPACK (1980's)**
**(Vector operations)**

**PVM and MPI**

**Level 2 & 3 BLAS - ATLAS**

**ScaLAPACK (2000's)**
**(Distributed Memory)**

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

**SLATE (2020's)**
**(DM and Heterogeneous arch)**

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

Evolving Software and Algorithms
Following Hardware Developments

# LINPACK Benchmark ⟶ Top500

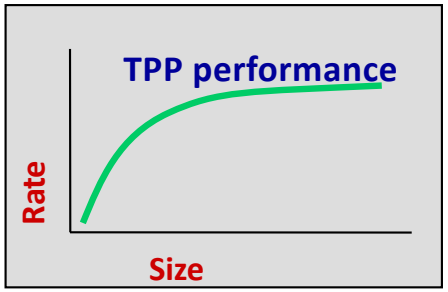
- Since 1977 I maintained a LINPACK Benchmark list.
- Hans Meuer and Erich Strohmaier had a list of fastest computers ranked by peak performance.
- Since 1993 listing of the 500 most powerful computers using 64-bit floating point arithmetic.
- Yardstick: Performance for

  *Ax=b, dense problem*

Maintained and updated twice a year:

SC‘xy in the States in November

Meeting in Germany in June



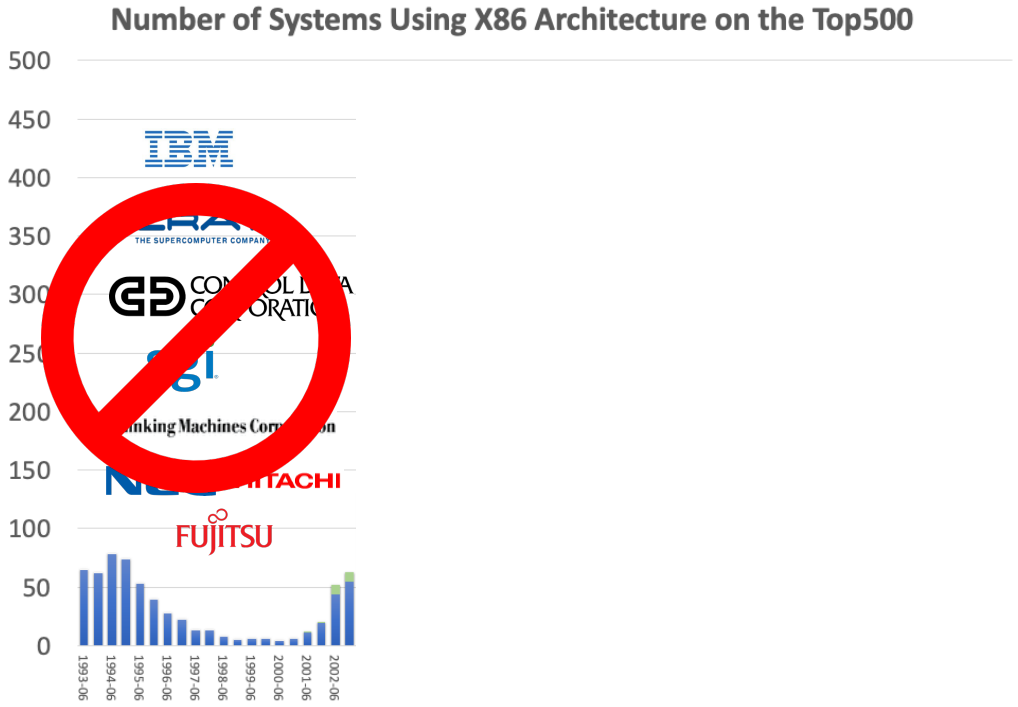

TOP 500 The List.

# 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
  - 53% of systems used Intel processors
  - Another 38% used AMD processors
- **92% of the systems use x86-64 architecture**
  - Many use GPU accelerators



# Today, Our HPC Systems are Based on Commodity Parts

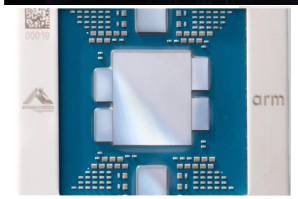
- **Commodity Processors**
  - 92% of the Top500 system use X86 (Intel & AMD) instruction set
- **Commodity Accelerators**
  - 88% of accelerated systems use NVIDIA
- **Commodity Interconnect**
  - 91% of the Top500 systems use Ethernet or Infiniband
- **Commodity OS**
  - 100% of the Top500 systems run on Linux

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

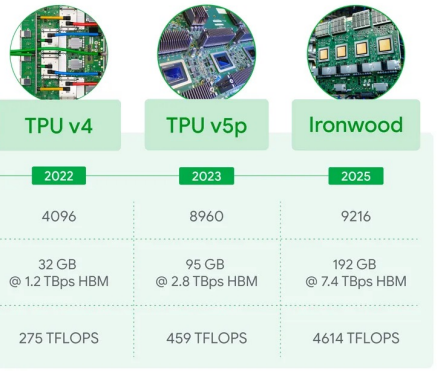
# Cloud Providers are Designing and Using Their Own Processors

- 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




- Microsoft Azure
  - Project Catapult/Brainwave FPGA accelerator
  - Cobalt 100 (128 Neoverse N2 ARMv9 cores)
  - Maia 200 AI accelerator
  - Q1 2026 $32.9B investment in data centers

In generative AI and cloud computing, we see genuine co-design successes.

Modern accelerators and AI ASICs are shaped in close dialogue with a small set of dominant workloads: transformer models, recommendation engines, cloud infrastructure accelerators, and related kernels.

Every hyperscaler AI vendor has designed and fabricated custom solutions to improve performance, increase reliability, and decrease costs.

OpenAI

SambaNova SYSTEMS

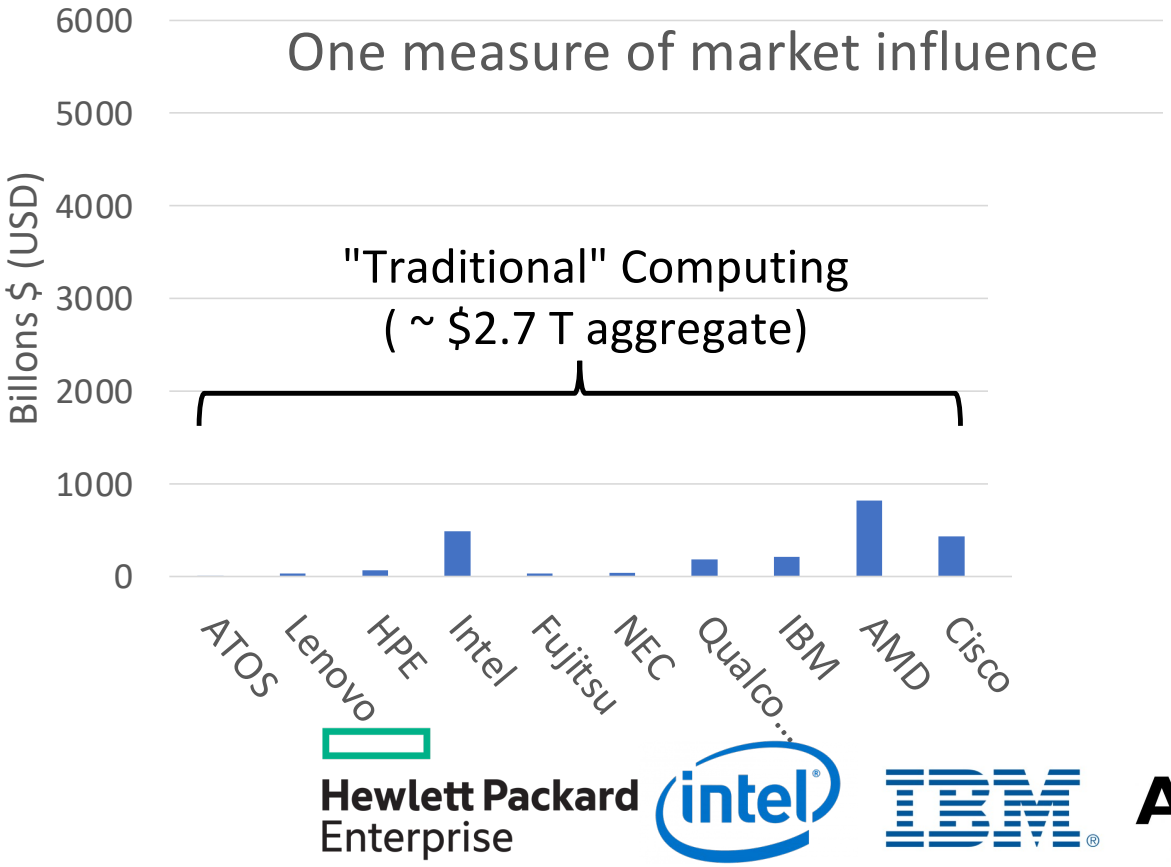
cerebras

GRAPHCORE

groq

# Market Capitalizations

July 17, 2026

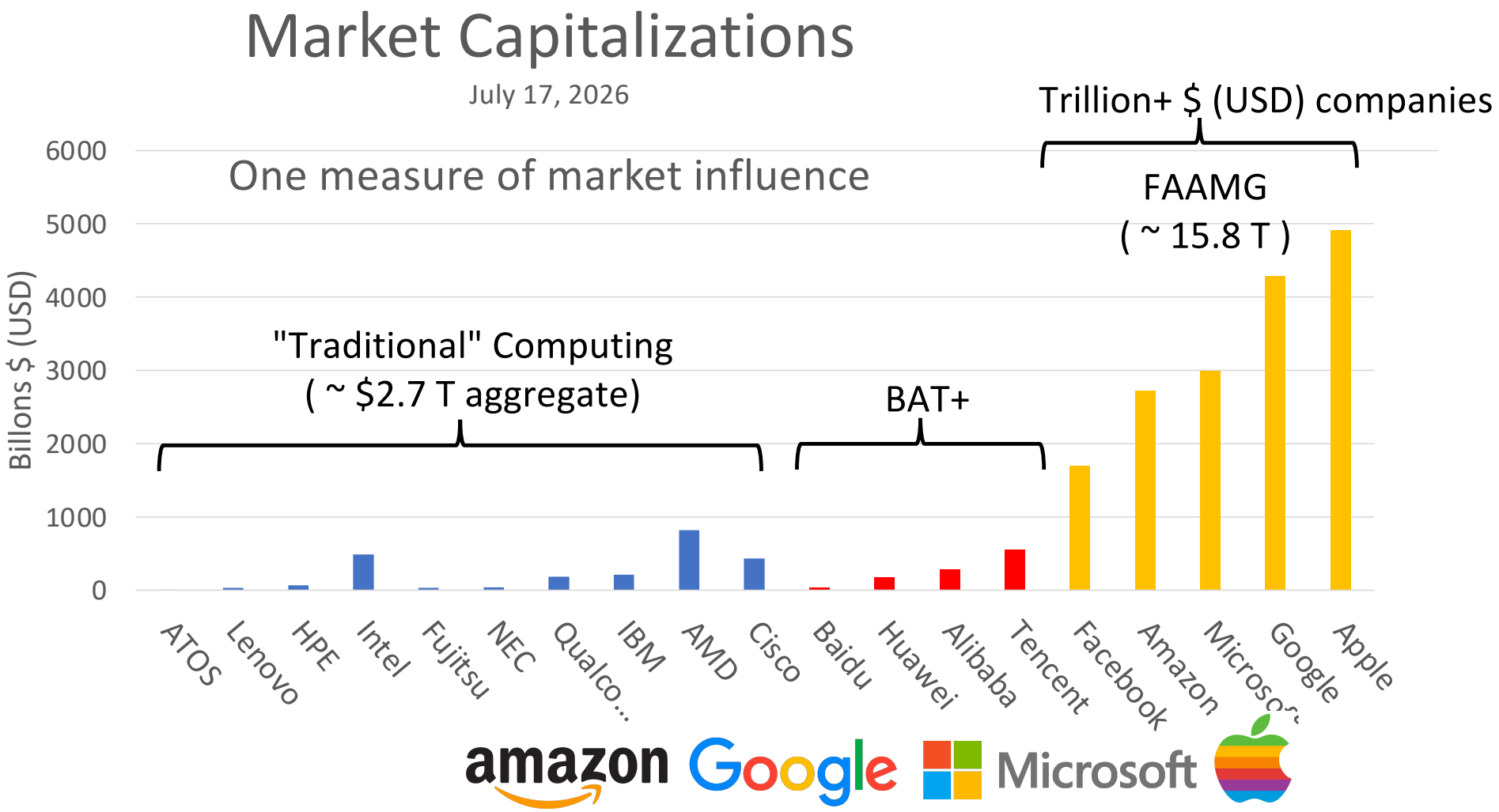


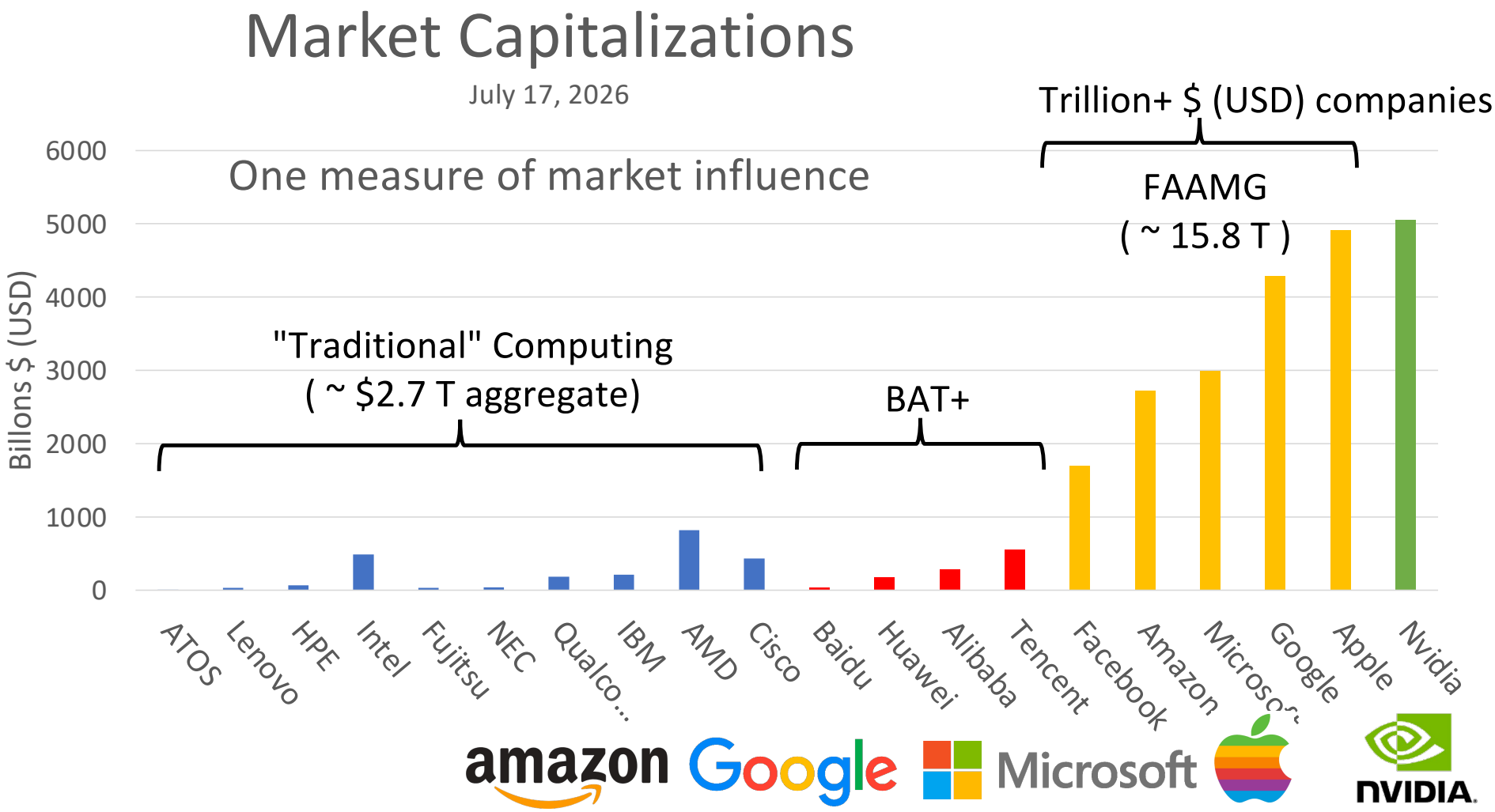

Hewlett Packard Enterprise

intel

IBM

AMD


Market Capitalizations
July 17, 2026
One measure of market influence
Trillion+ $ (USD) companies
FAAMG
( ~ 15.8 T )
"Traditional" Computing
( ~ $2.7 T aggregate)
BAT+
Billons $ (USD)
6000
5000
4000
3000
2000
1000
0
ATOS
Lenovo
HPE
Intel
Fujitsu
NEC
Qualco...
IBM
AMD
Cisco
Baidu
Huawei
Alibaba
Tencent
Facebook
Amazon
Google
Apple
amazon
Google
Microsoft


Market Capitalizations
July 17, 2026
One measure of market influence
Trillion+ $ (USD) companies
FAAMG
( ~ 15.8 T )
"Traditional" Computing
( ~ $2.7 T aggregate)
BAT+
Billons $ (USD)
6000
5000
4000
3000
2000
1000
0
ATOS
Lenovo
HPE
Intel
Fujitsu
NEC
Qualco...
IBM
AMD
Cisco
Baidu
Huawei
Alibaba
Tencent
Facebook
Amazon
Microsoft
Google
Apple
Nvidia
amazon
Google
Microsoft
NVIDIA


## The Fastest Supercomputers are at an Exaflop.

## What's an Exaflop?

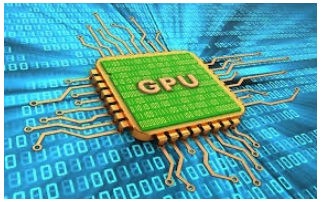
- 1 flop = Addition or Multiplication of 64-bit floating point numbers
- Exaflop is a billion-billion ($10^{18}$) floating point operations per second
- If each person on Earth completed 1 calculation per second, it would take more than 4 years to do what an Exascale computer can do in 1 second

# The Environment for High Performance Computing in Scientific Computation

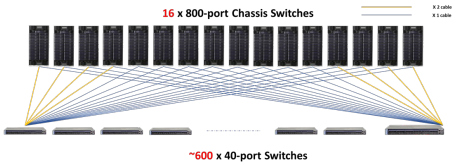
- Highly parallel
  - Distributed memory
  - MPI + Open-MP programming model


**LLNL El Capitan, 2.7 Eflop/s,**
**11 x $10^6$ Cores, 11,136 nodes, 35 MW**
**(node = 3-AMD CPU + 3-AMD GPUs)**
**> 99% of performance from GPUs**

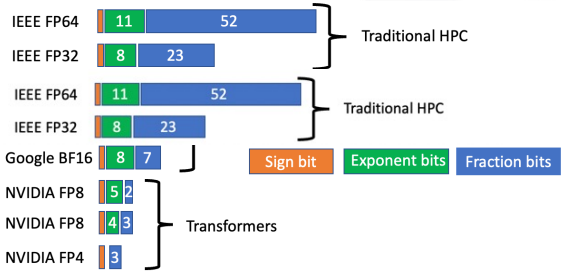
- Heterogeneous
  - Commodity processors + GPU accelerators


- Communication between parts very expensive compared to floating point ops




- Comparison of operation counts may not reflect time to solution

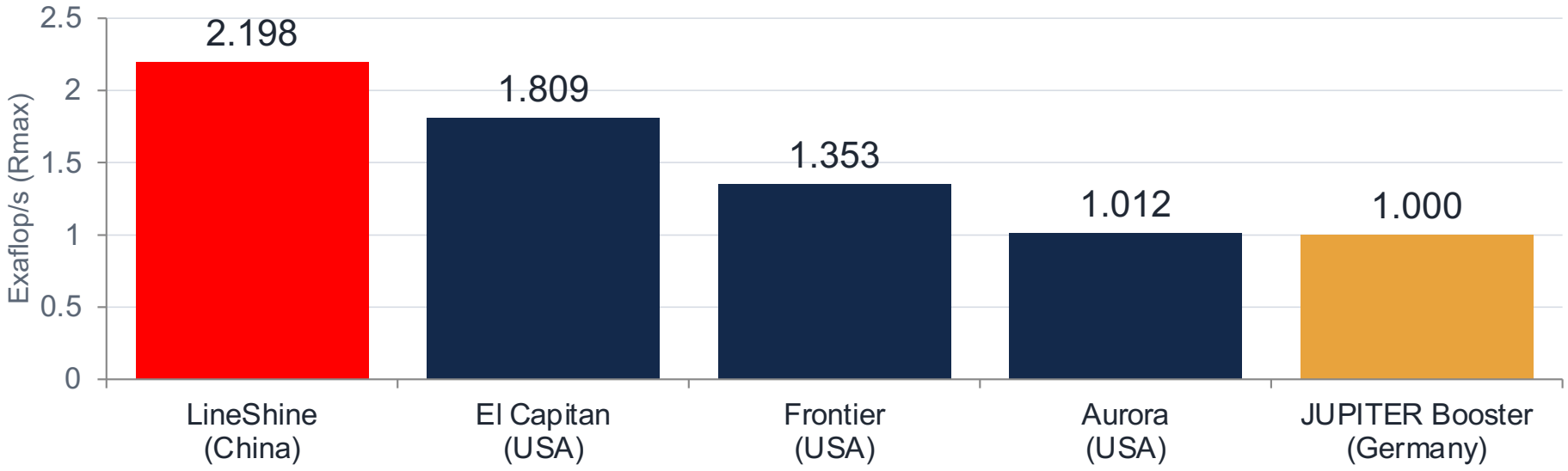



- Floating point hardware at 64 & 32 bits

- Floating point hardware at 64, 32, 16, 8, & 4 bits

THE EXASCALE CLUB

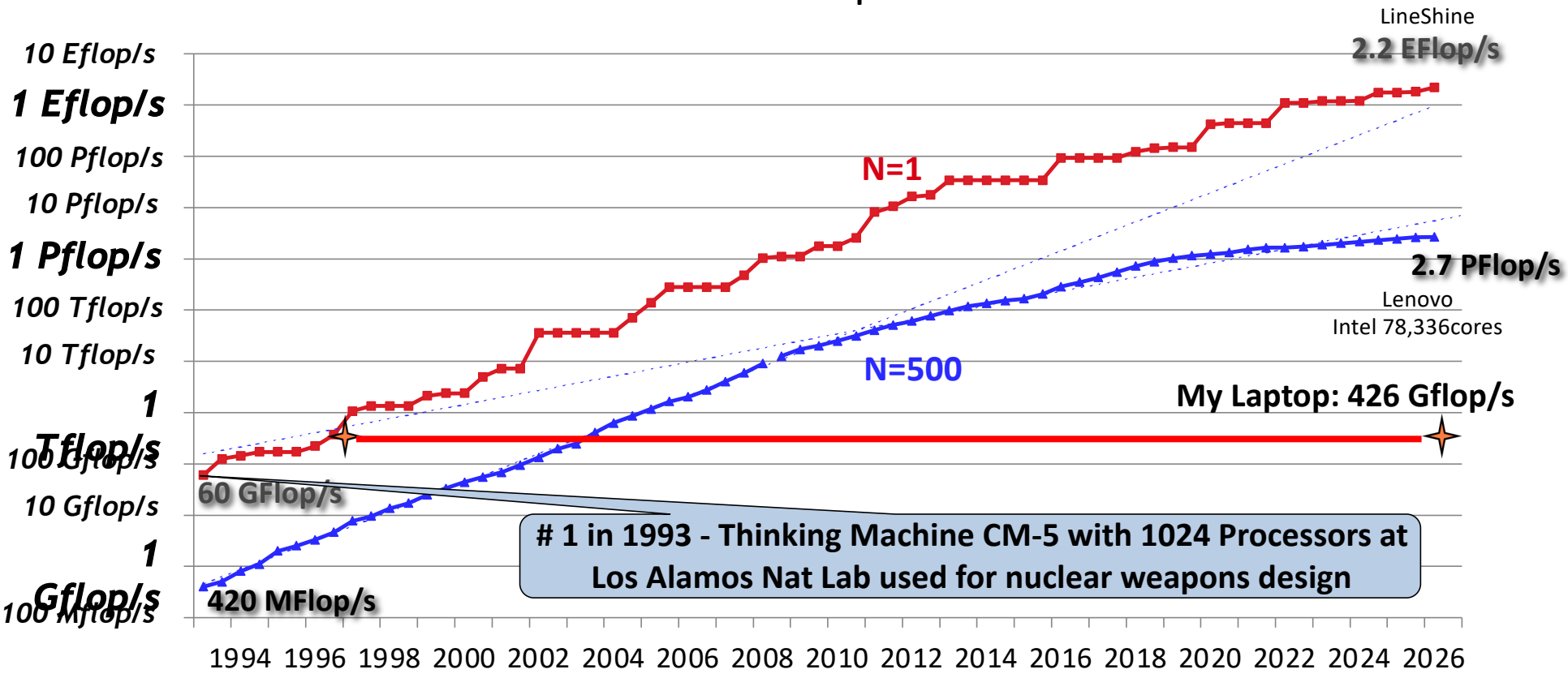
# Five Systems Now Break the Exaflop Barrier




JUPITER Booster was the first non-US exascale system in November.

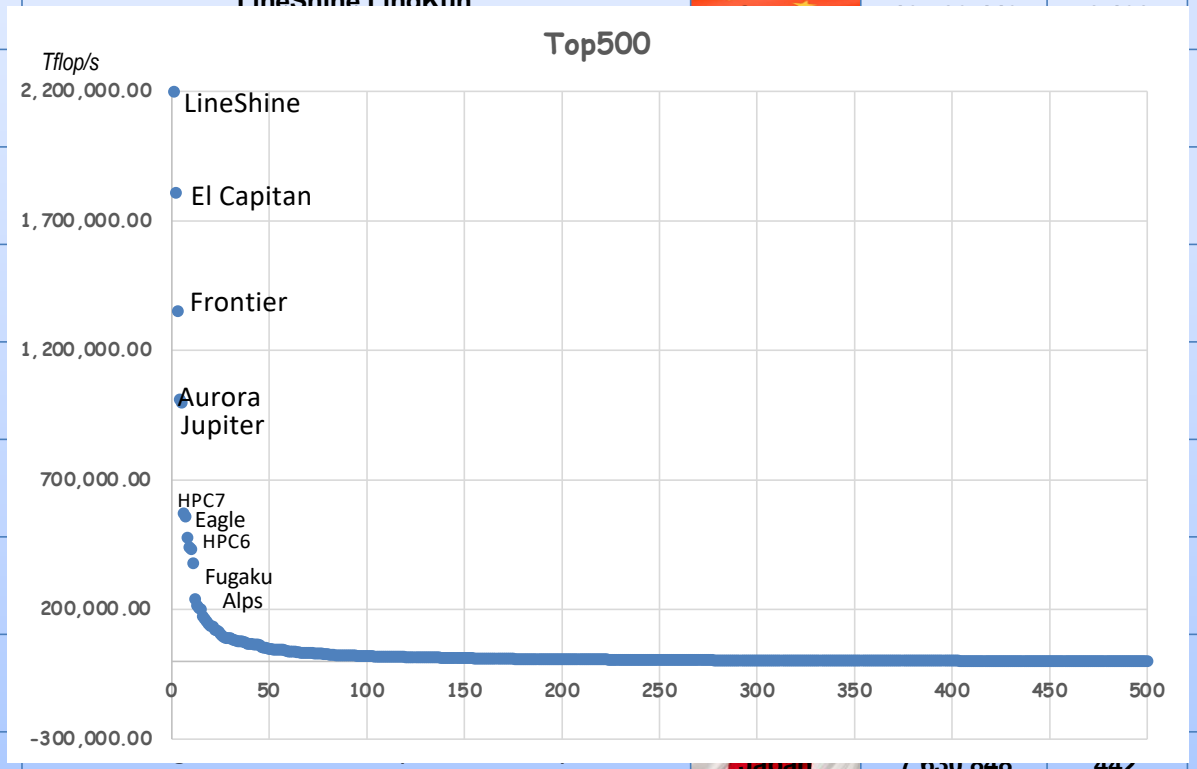
LineShine is the second — and now the leader.

# Performance Development of HPC over the Last 34 Years from the Top500



# June 2026: The TOP 10 Systems (53% of the Total Performance of Top500)

| Rank | Site | Computer | Country | Cores | Rmax [Pflops] | % of Peak | Power [MW] | GFlops/ Watt |
|---|---|---|---|---|---|---|---|---|
| 1 | NSCS Shenzhen | LineShine LingKun | | | | 80 | 42.2 | 52.1 |
| 2 | DOE / NNSA LLNL | | | | | 66 | 29.7 | 60.9 |
| 3 | DOE / OS Oak Ridge Nat Lab | | | | | 65 | 24.6 | 55.0 |
| 4 | DOE / OS Argonne Nat Lab | | | | | 51 | 38.7 | 26.1 |
| 5 | EuroHPC/FZL | | | | | 82 | 13.1 | 60.5 |
| 6 | Eni S.p.A. | | | | | 66 | 8.7 | 65.7 |
| 7 | Microsoft, Azure Cloud | | | | | 66 | - | |
| 8 | Eni S.p.A. | | | | | 78 | 8.46 | 56.5 |
| 9 | RIKEN Center for Computational Science | Tofu D Interconnect | Japan | 7,630,848 | 442. | 82 | 29.9 | 14.8 |
| 10 | Swiss National Supercomputing Center CSCS | Alps, HPE Cray EX254n, Nvidia Grace 72C 3.1. Nvidia GH200 Superchip, Slingshot 11 | Swiss | 2,121,600 | 434. | 76 | 7.12 | 61.0 |

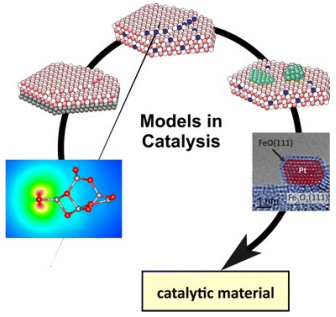


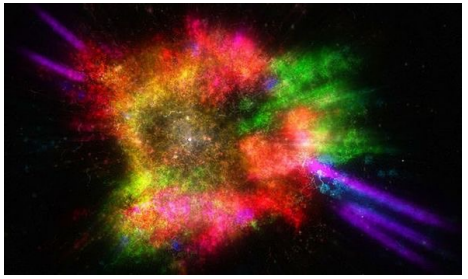
# Performance and Benchmarking Evaluation Tools

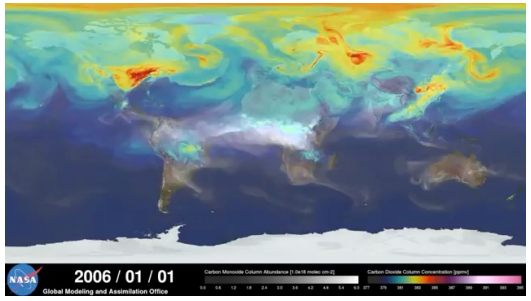
- **Linpack Benchmark - Longstanding benchmark started in 1979**
  - **Lots of positive features; easy to understand and run; shows trends**
- **However, much has changed since 1979**
  - **Arithmetic was expensive then and today it is over-provisioned and inexpensive**
- **Linpack performance of computer systems is no longer strongly correlated to real application performance**
  - **Linpack benchmark based on dense matrix multiplication**
- **Designing a system for good Linpack performance can lead to design choices that are wrong for today's applications**

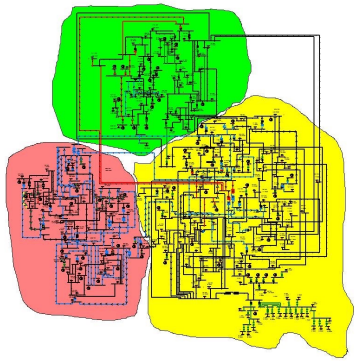
# Today's Top HPC Systems Used to do Simulations

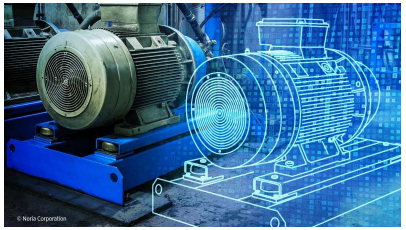
- *Climate*
- *Combustion*
- *Nuclear Reactors*
- *Catalysis*
- *Electric Grid*
- *Fusion*
- *Stockpile*
- *Supernovae*
- *Materials*
- *Digital Twins*
- *Accelerators*
- …

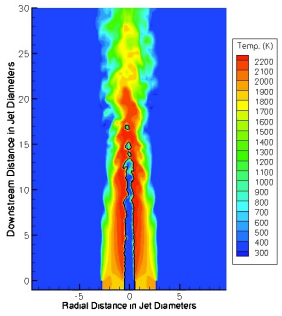


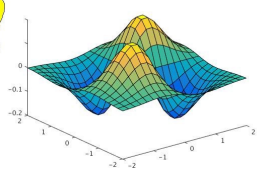



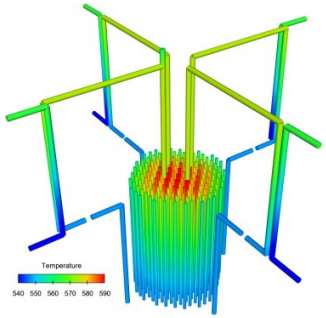


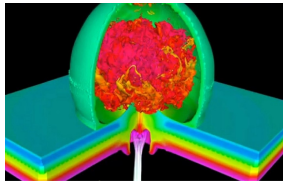



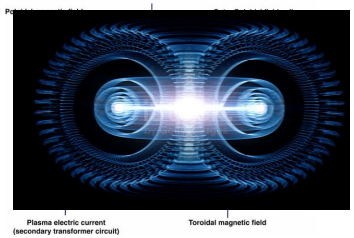






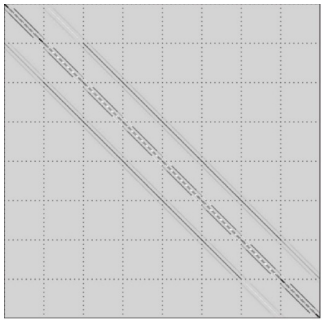



- Usually 3-D PDE's
  - Sparse matrix computations, not dense

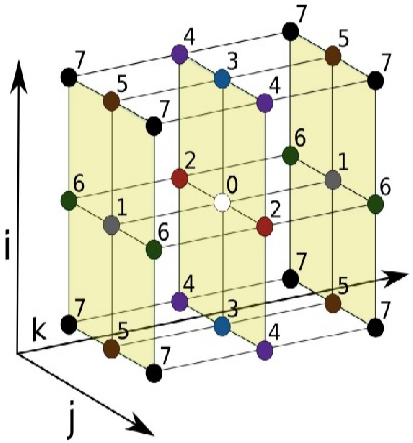
hpcg-benchmark.org With Piotr Luszczek and Mike Heroux

# 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 2026

Ax=b
Dense A

Ax=b
Sparse A

| Rank | Site | Computer | Cores | HPL Rmax (Pflop/s) | TOP500 Rank | HPCG (Pflop/s) | Fraction of Peak |
|---|---|---|---|---|---|---|---|
| 1 | NSCS (Shenzhen) **China** | **LineShine**, LingKun, LX2 304C 1.55GHz, LingQi | 13,789,440 | 2198 | 1 | 22.0 | 0.8% |
| 2 | 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 | 2 | 17.4 | 0.6% |
| 3 | RIKEN Center for Computational Science **Japan** | **Fugaku**, Fujitsu A64FX 48C 2.2GHz, Tofu D, Fujitsu | 7,630,848 | 442 | 9 | 16.0 | **3.0%** |
| 4 | DOE/SC/ORNL **USA** | **Frontier,** HPE Cray Ex235a, AMD 3rd EPYC 64C, 2 GHz, AMD Instinct MI250X, Slingshot-11 | 9,066,176 | 1353 | 3 | 14.1 | 0.7% |
| 5 | Eni S.p.A **Italy** | **HPC7,** HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11 | 3,461,472 | 571 | 6 | 5.95 | 0.7% |
| 6 | DOE/SC/ANL **USA** | **Aurora,** HPE Cray EX, Intel Max 9470 52C, 2.4 GHz, Intel GPU MAX, Slingshot-11 | 9,264,128 | 1012 | 4 | 5.6 | 0.3% |
| 7 | EuroHPC/CSC **Finland** | **LUMI**, HPE Cray EX235a, AMD Zen-3 (Milan) 64C 2GHz, AMD MI250X, Slingshot-11 | 2,752,704 | 380 | 11 | 4.6 | 0.9% |
| 8 | Softbank **Japan** | **CHIE-4**, NVIDIA DGX B200, Xeon Platinum 8570 56C 2.1 GHz, B200 SXM 180GB, Infiniband NDR400 | 662,256 | 135 | 21 | 3.76 | **2.5%** |
| 9 | CSCS **Switzerland** | **Alps**, HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11 | 2,121,600 | 435 | 10 | 3.67 | 0.6% |
| 10 | 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 | 12 | 3.1 | 1.0% |

# *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**


# “Responsibly Reckless” Algorithms

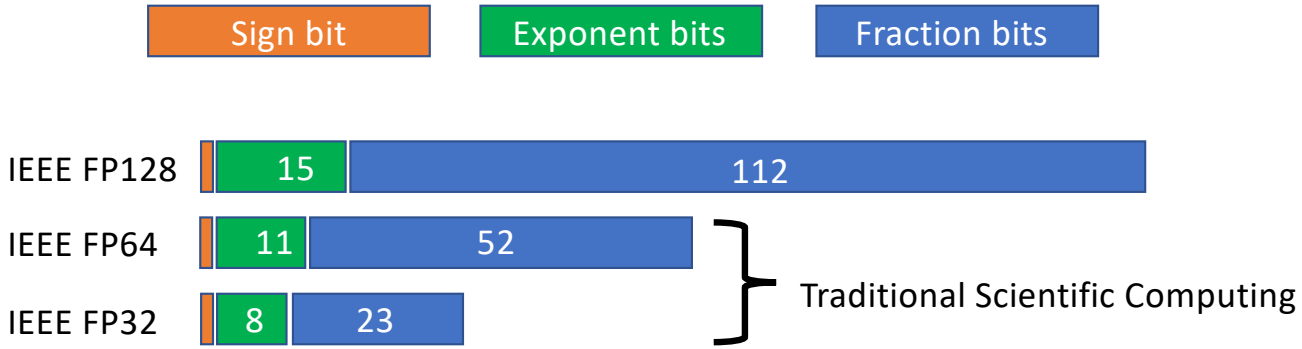
Try a fast algorithm (unstable algorithm) that might fail (but rarely)

- Avoiding Data Movement
- Avoiding Synchronization
- Use Mixed Precision

Check for instability

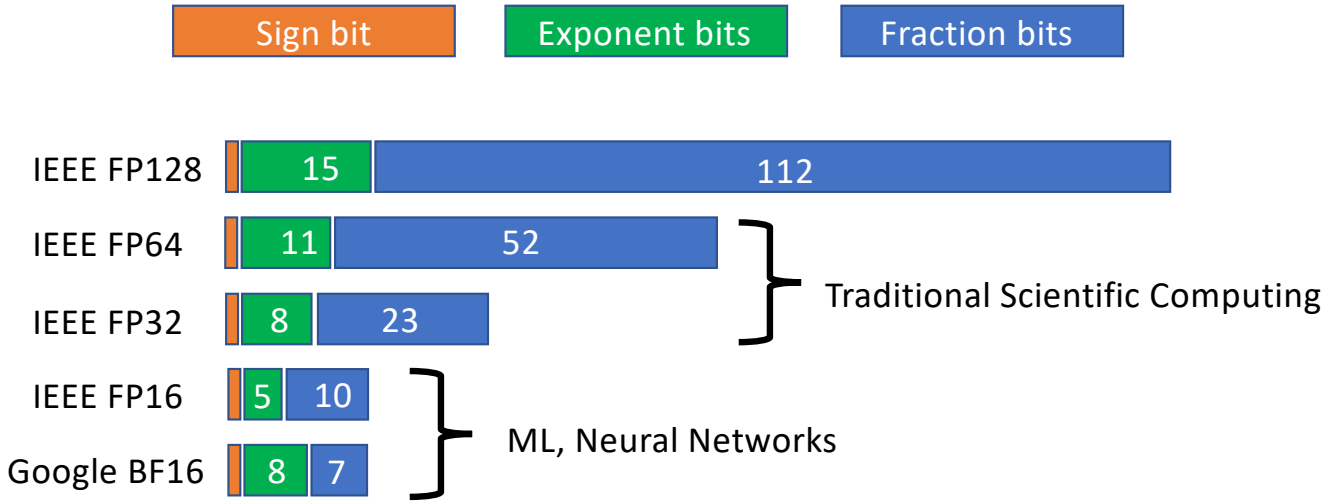
If needed, recompute with a stable algorithm

# Floating Point Representation



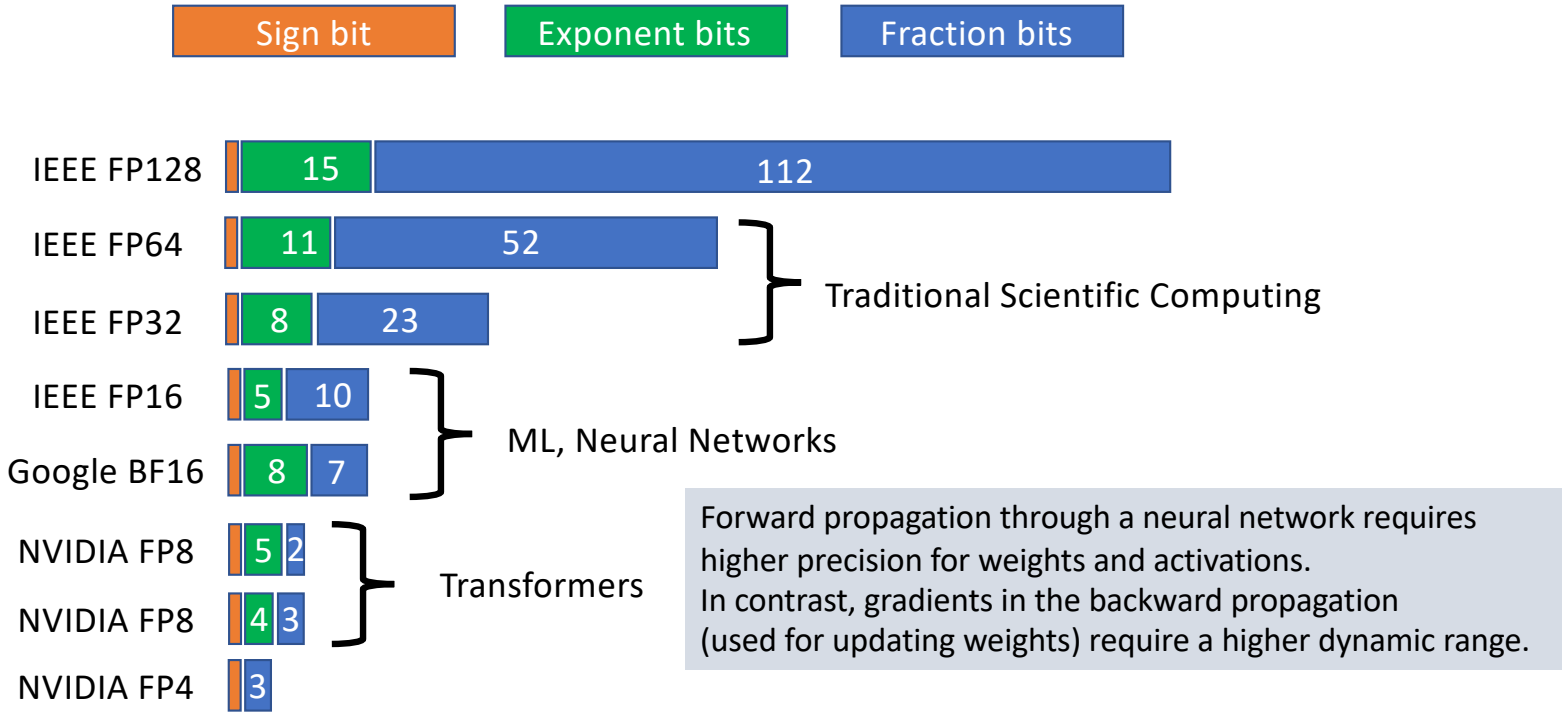

Can we leverage the short precision in our "traditional" scientific numerical computations?

# Floating Point Representation




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?

# Mixed Precision++

## Use a mathematical technique

- Get an approximation in lower precision (fast) then use something like Newton's method to enhance accuracy.
- Newton's Method
  - $x_+ = x - f(x)/f'(x)$
  - For $Ax = b$; $f(x) = b - Ax$ and $f'(x) = -A$
  - $x_+ = x + A \setminus ( b - Ax )$; $r = b - Ax$
  - $(x_+ - x) = A^{-1} * r$
  - $\Delta = (L*U)^{-1} * r$

# Leveraging Half Precision

**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:
   - ~~z = U\(L\r) Classical Iterative Refinement~~ ~~lower precision~~ ~~$O(n^2)$~~
   - 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.*

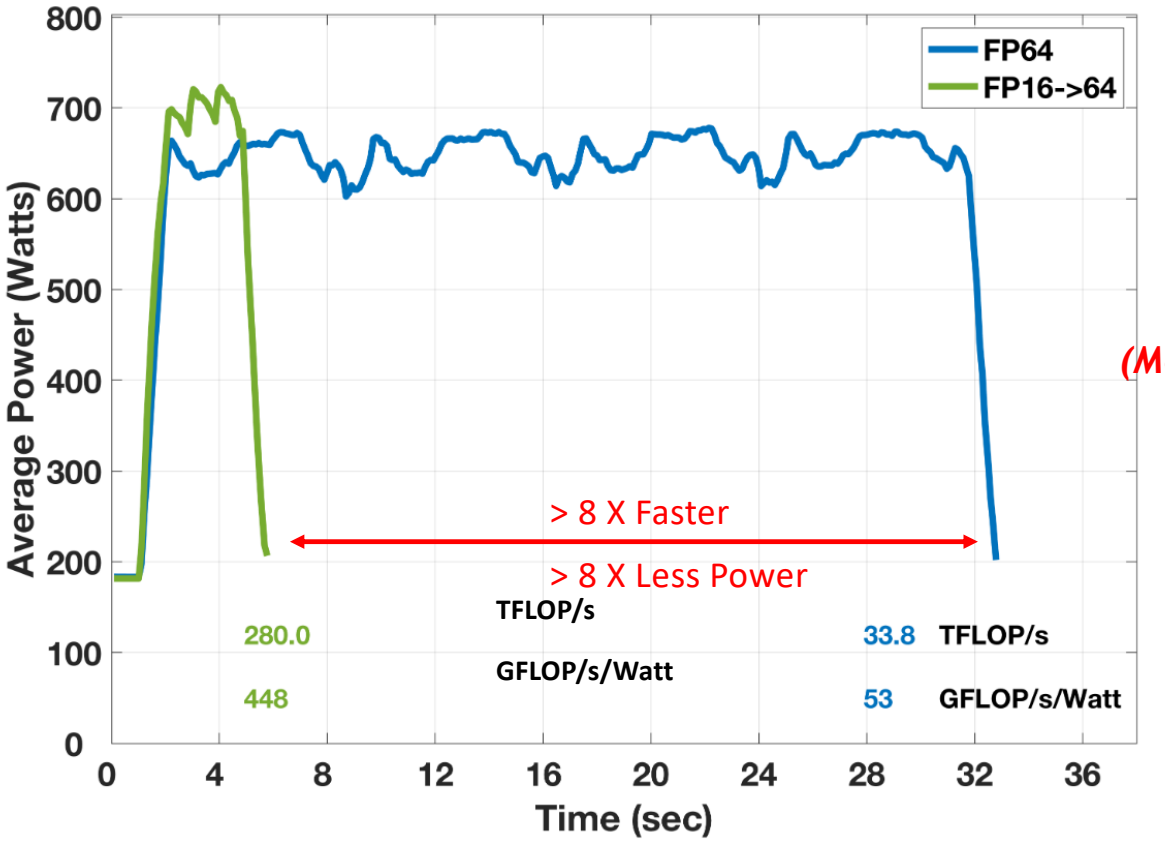
J. Langou, et al., Exploiting the Performance of 32 bit fl-pt Arithmetic in Obtaining 64 bit Accuracy, in: Proc. of SC06

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

NVIDIA Blackwell B200 GPU
Accelerated Iterative Refinement




*Flop/s = $2n^3/(3\ time)$*

*(Meaning 8X is 8 times faster)*

# 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 2026

**Mixed precision is no longer a niche technique.**
**It is now a credible path to production-scale performance.**

| 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 | 2 | 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 | 4 | 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 | 3 | 11.4 | 8.4 |
| 4 | NSCS (Shenzhen) **China** | **LineShine**, LingKun, LX2 304C 1.55GHz, LingQi | 13,789,440 | 2.198 | 1 | 7.9227 | 3.6 |
| 5 | EuroHPC/FZJ **Germany** | **JUPITER Booster**, BullSequana XH3000, NVIDIA Grace 72C 3GHz, GH200, NVIDIA Infiniband NDR200 | 4,801,344 | 1.0 | 5 | 7.9221 | 7.9 |
| 6 | Eni S.p.A **Italy** | **HPC7,** HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11 | 3,461,472 | 0.571 | 6 | 4.3 | 7.5 |
| 7 | Softbank **Japan** | **CHIE-4,** NVIDIA DGX B200, Xeon Platinum 8570 56C 2.1 GHz, B200 SXM 180GB, Infiniband NDR400 | 662,256 | 0.135 | 21 | 3.3 | 24.4 |
| 8 | 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 | 19 | $2.36_{8}$ $1.2_{16}$ | $16.3_{8}$ $8.3_{16}$ |
| 9 | EuroHPC/CSC **Finland** | **LUMI**, HPE Cray EX235a, AMD Zen-3 (Milan) 64C 2GHz, AMD MI250X, Slingshot-11 | 2,752,704 | 0.380 | 11 | 2.35 | 6.2 |
| 10 | RIKEN Center for Computational Science, **Japan** | **Fugaku**, Fujitsu A64FX 48C 2.2GHz, Tofu D | 7,630,848 | 0.442 | 9 | 2.0 | 4.5 |

# Recent Nvidia GPUs

| | Figure of Merit<br>Peak Performance |
|---|---|
| Operations | 2022<br>Hopper (H200) |
| FP64 FMA | 33.5 Tflop/s |
| FP64 Tensor Core | 67 Tflop/s |
| | |
| FP16 Tensor Core | 989 Tflop/s |
| BF16 Tensor Core | 989 Tflop/s |
| | |
| INT8 Tensor Core | 1979 TOP/s |
| | |
| Memory BW | 4.8 TB/s |

# Recent Nvidia GPUs

| | Figure of Merit<br>Peak Performance | |
|---|---|---|
| Operations | 2022<br>Hopper (H200) | 2024<br>Blackwell (B200) |
| FP64 FMA | 33.5 Tflop/s | 40 Tflop/s |
| FP64 Tensor Core | 67 Tflop/s | 40 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 |

# Recent Nvidia GPUs

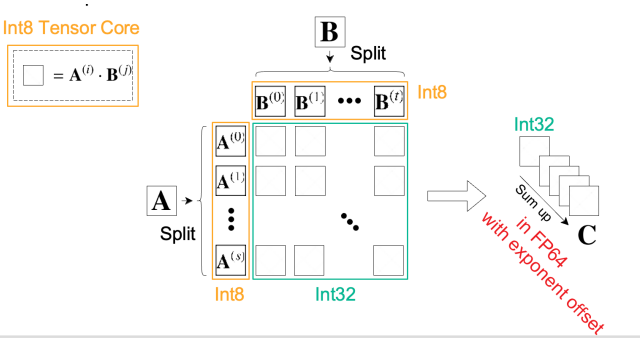
https://www.nvidia.com/en-us/data-center/vera-rubin-nvl72/

| | Figure of Merit<br>Peak Performance | | |
|---|---|---|---|
| Operations | 2022<br>Hopper (H200) | 2024<br>Blackwell (B200) | 2026<br>Rubin |
| FP64 FMA | 33.5 Tflop/s | 40 Tflop/s | 33 Tflop/s |
| FP64 Tensor Core | 67 Tflop/s | 40 Tflop/s | 33 Tflop/s |
| | | | |
| FP16 Tensor Core | 989 Tflop/s | 2250 Tflop/s | 4000 Tflop/s |
| BF16 Tensor Core | 989 Tflop/s | 2250 Tflop/s | 4000 Tflop/s |
| | | | |
| INT8 Tensor Core | 1979 TOP/s | 4500 TOP/s | 250 TOP/s |
| | | | |
| Memory BW | 4.8 TB/s | 8 TB/s | 22 TB/s |

# Opportunity Breeds Innovation, Emulating Fl.Pt. with Integer arithmetic, “Ozaki Scheme”

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

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




Use Int8 Tensor Cores for each matrix multiplication
Int8-input Int32-accumulation

**Analysis of Floating-Point Matrix Multiplication Computed via Integer Arithmetic**

Ahmad Abdelfattah, Jack Dongarra, Massimiliano Fasi, Mantas Mikaitis, Françoise Tisseur

http://arxiv.org/abs/2506.11277

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

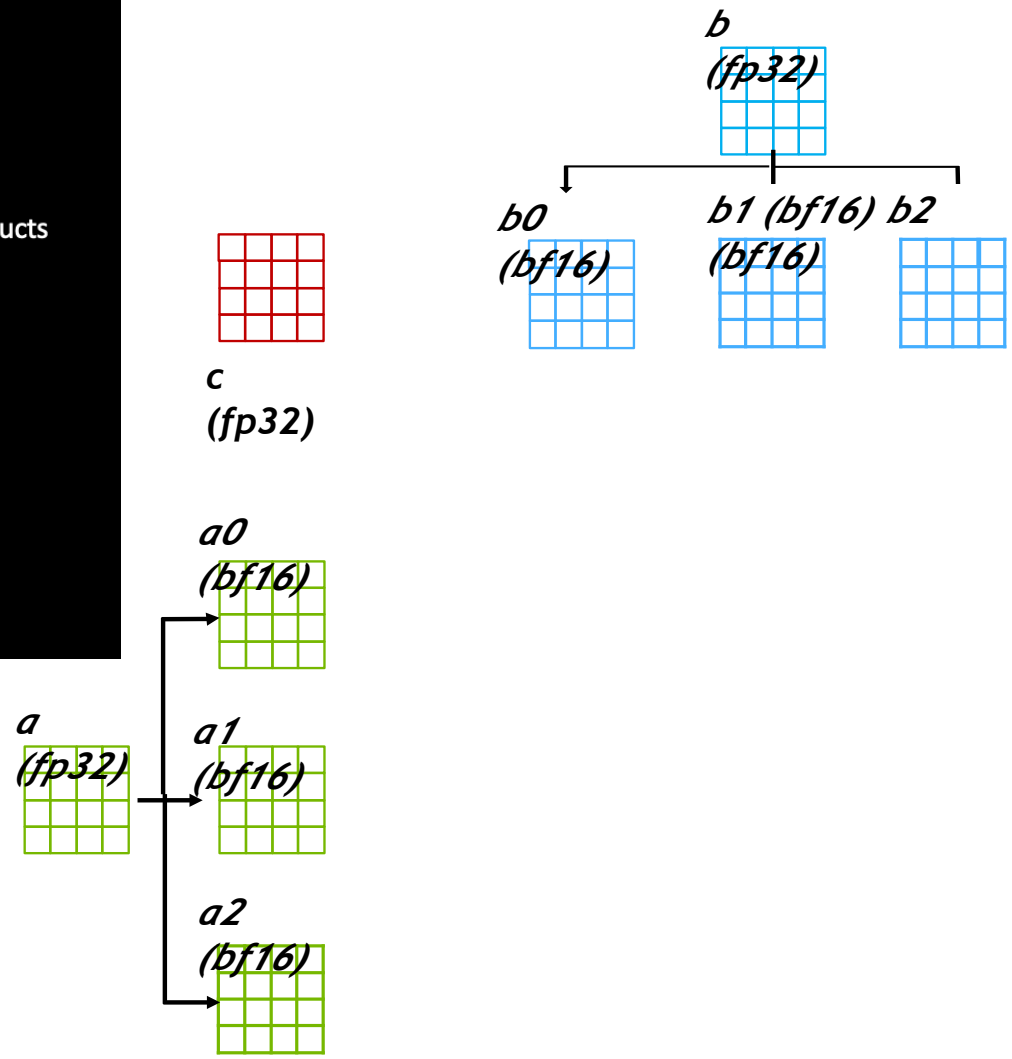
  $a = a0 + 2^{-8}.a1 + 2^{-16}.a2$

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

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

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

- 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]

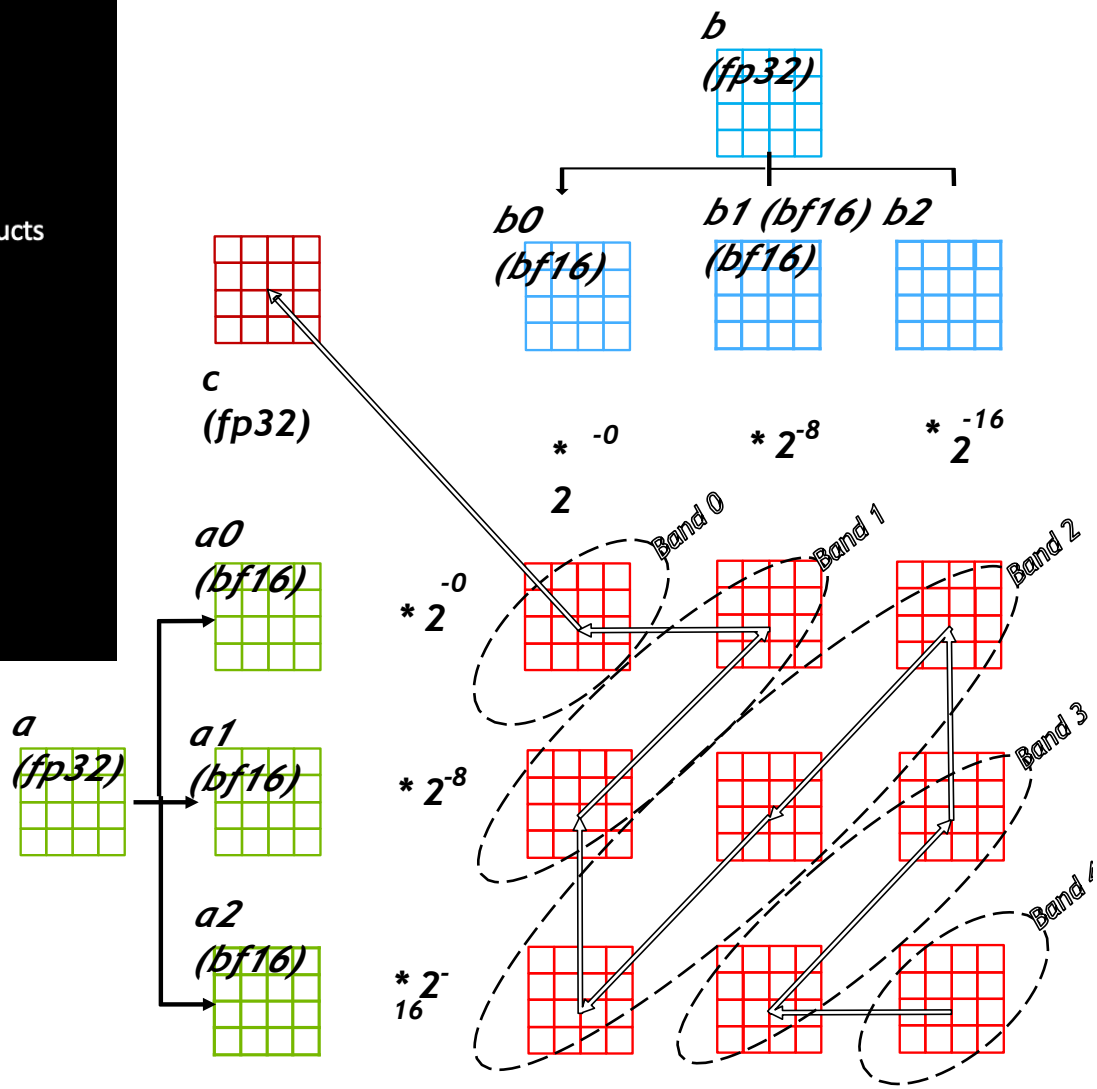
  $a = a0 + 2^{-8}.a1 + 2^{-16}.a2$

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

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

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

- 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]

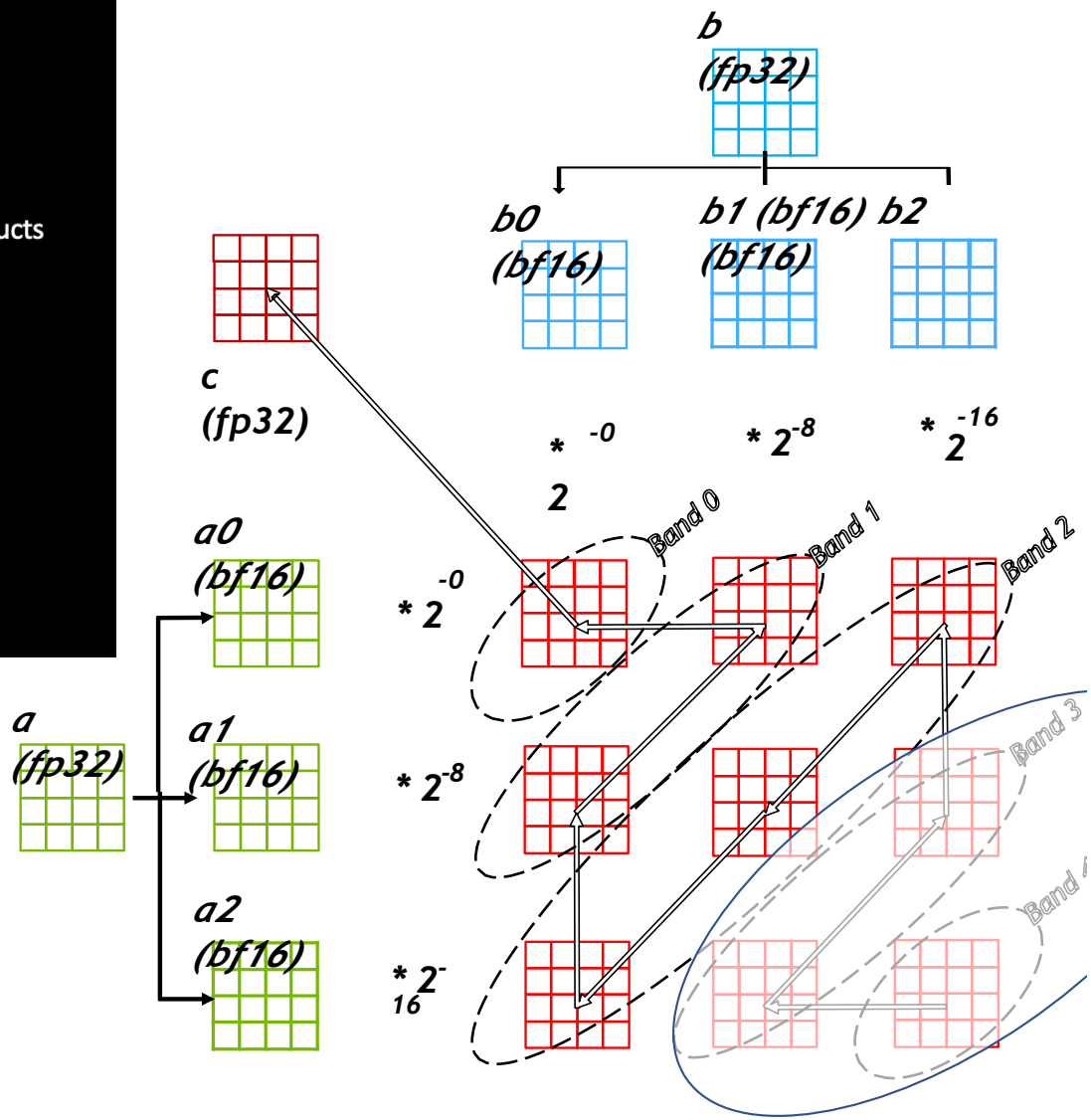
  $a = a0 + 2^{-8}.a1 + 2^{-16}.a2$

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

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

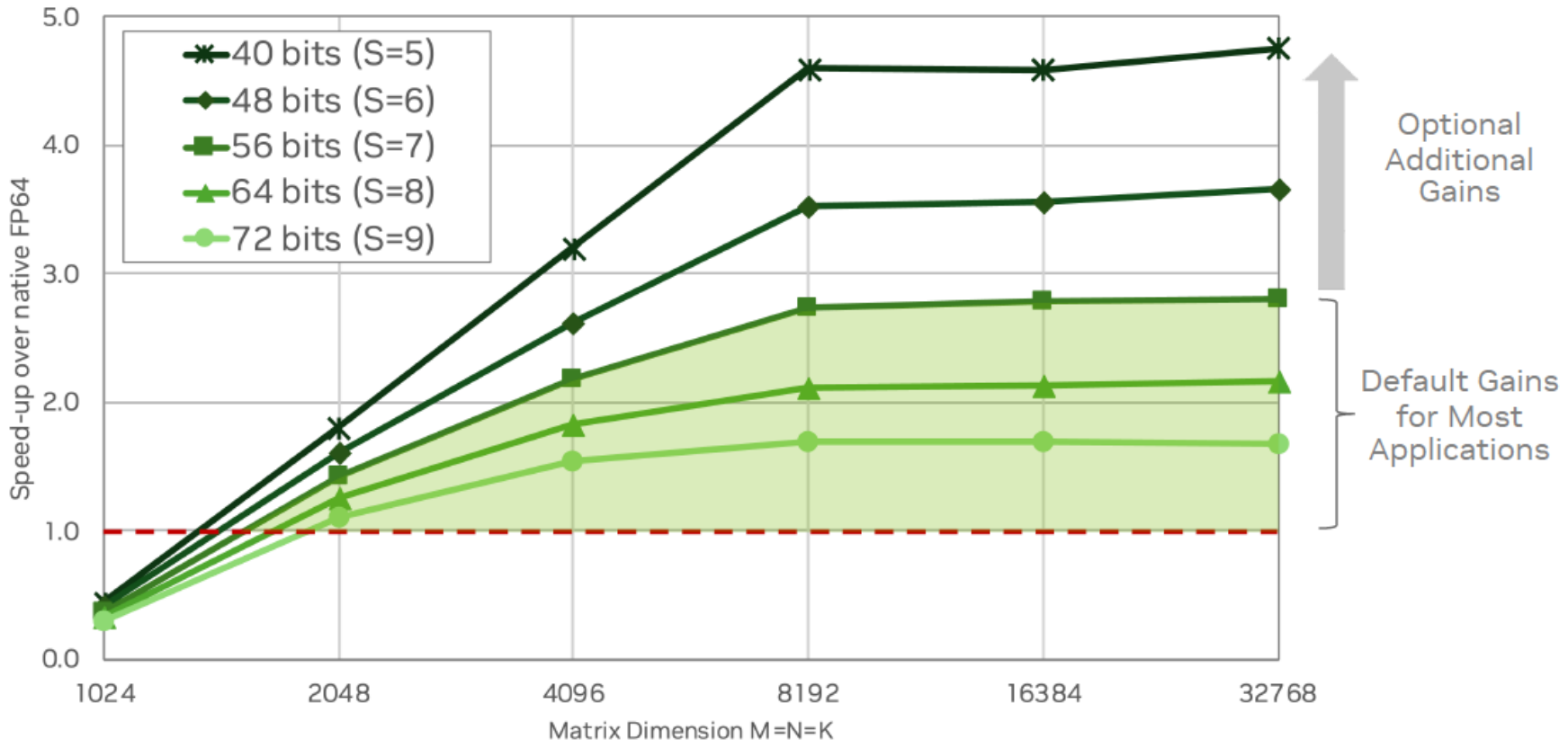
  $a * b + c = a0.b0 + 2^{-8}.a0.b1 + 2^{-16}.a0.b2 + 2^{-8}.a1.b0 + 2^{-16}.a1.b1 + 2^{-24}.a1.b2 + 2^{-16}.a2.b0 + 2^{-24}.a2.b1 + 2^{-32}.a2.b2 + c$

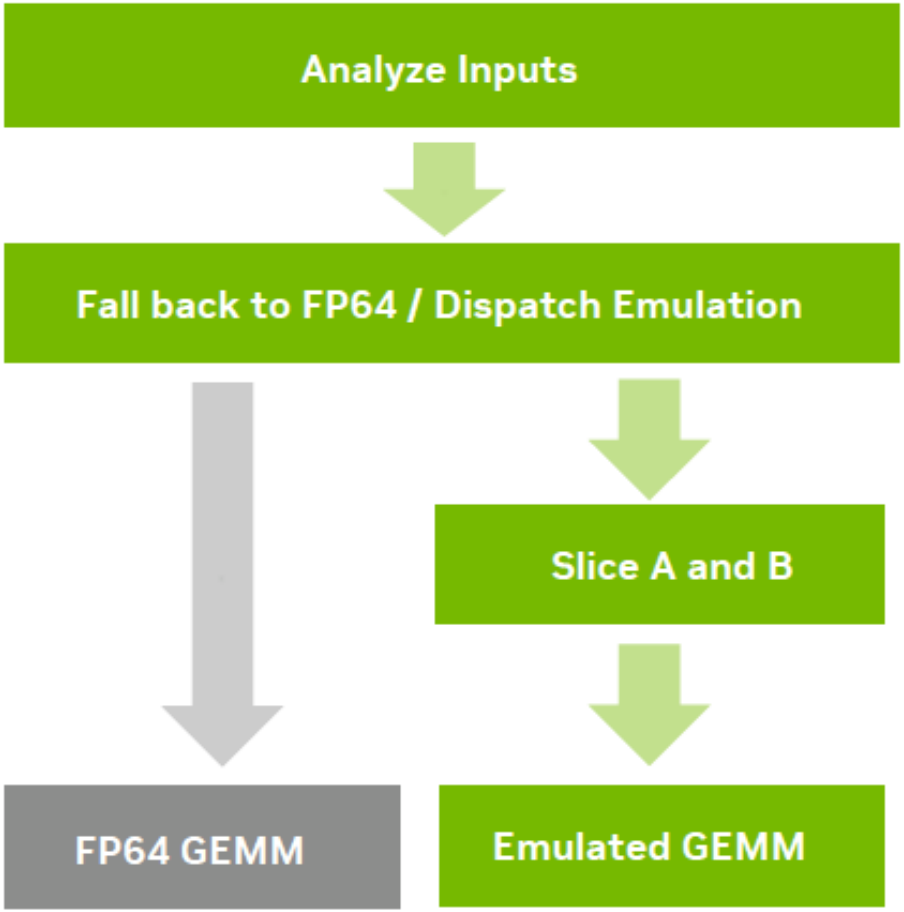
- The partial products are computed in the BF16 Tensor cores
- The partial products are scaled appropriately in the CUDA cores
- The tensor cores and CUDA cores work in parallel
- The effective FP32 FLOPs is 1/9th that of the BF16 tensor core FLOPs
  - On B200 (per GPU) 250 vs 80 TFLOP/s ➔ **>3X maximum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



# Potential additional efficiency gains

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



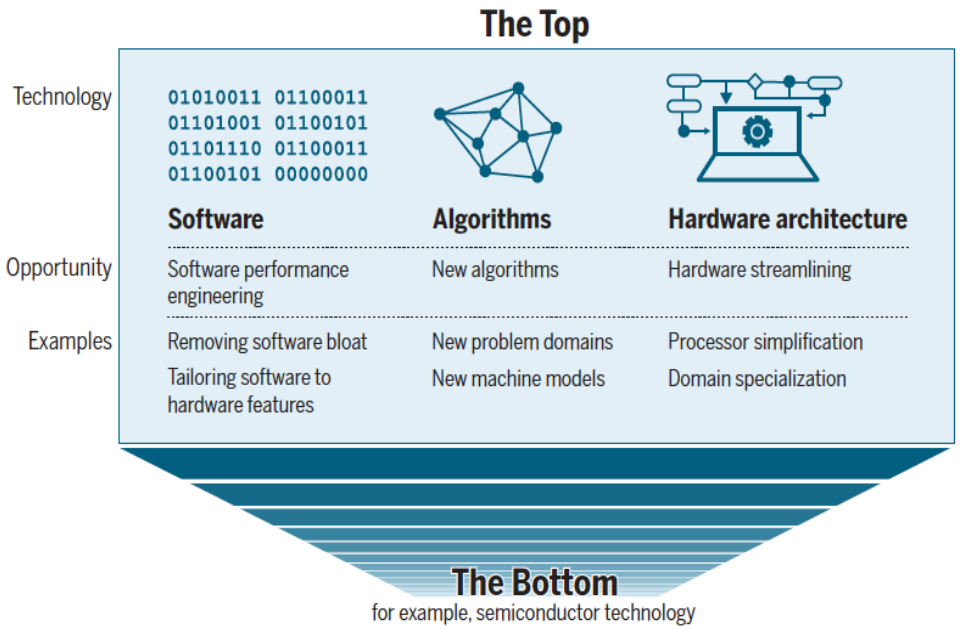

Analyze Inputs
Fall back to FP64 / Dispatch Emulation
Slice A and B
FP64 GEMM
Emulated GEMM

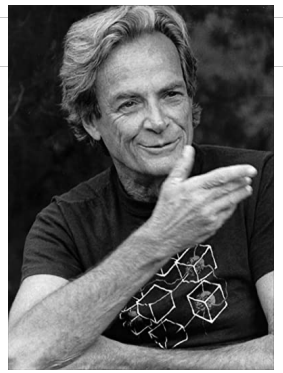
# The Take Away

- HPC Hardware is Constantly Changing
  - Scalar
  - Vector
  - Distributed
  - Accelerated
  - Mixed precision
- Three computer revolutions
  - High performance computing
  - Deep learning
  - Edge & AI
- 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



Feynman’s 1959 Lecture @ CalTech