

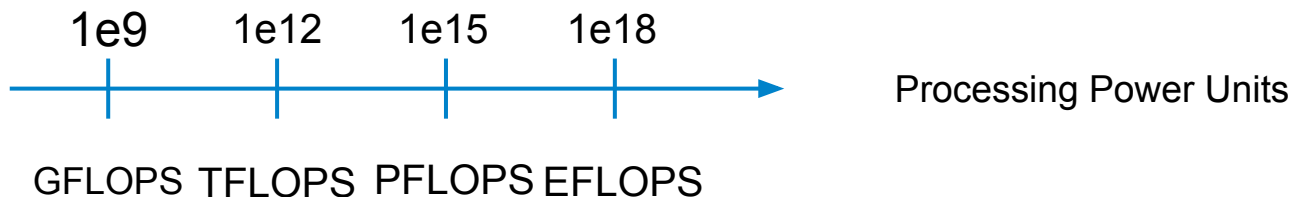
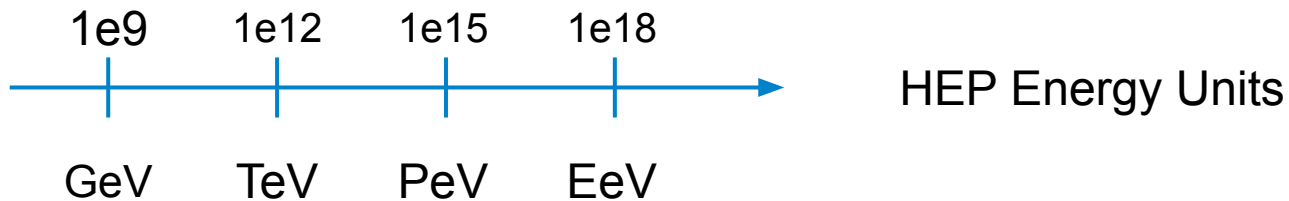


Supercomputers As a Resource

J. Taylor Childers - Argonne Leadership Computing Facility



Let's Unify Scales and Units



FLOPS = Floating Point Operations Per Second

eV = Electron Volt: kinetic energy gained by an electron accelerated through an electric potential difference of one volt in a vacuum

Hardware



U.S. DEPARTMENT OF
ENERGY

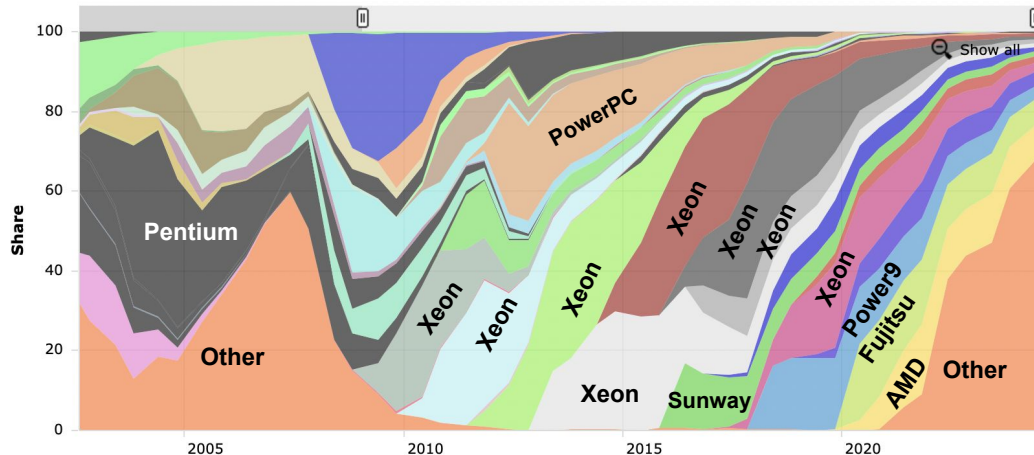
Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

Argonne  | **ALCF**
NATIONAL LABORATORY

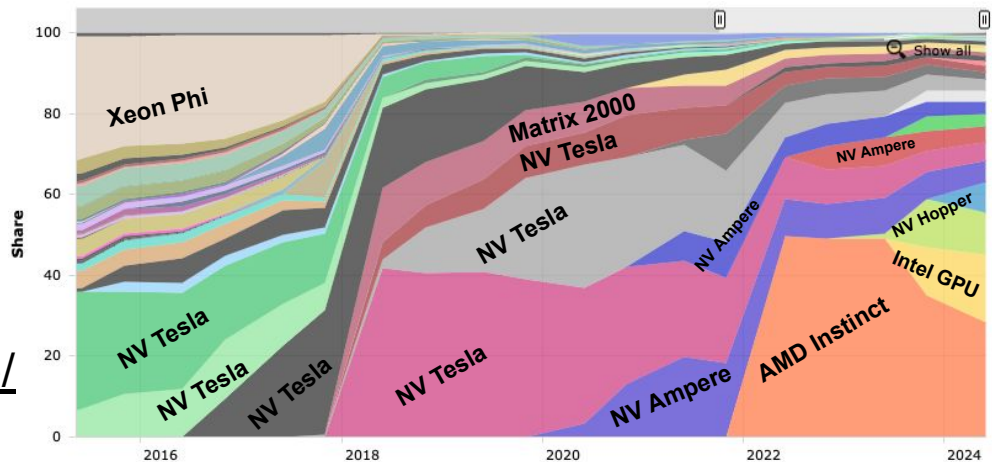
A Quick History

- Things reached a stable point in the 2010s
- Majority of CPUs in HPC were x86 and Intel
- This was and continues to be true for the LHC Grid as well.
- HPC architectures follow industry
- AI is driving industry innovation, so HPCs have become GPU machines

CPU Processor Generation - Performance Share



Accelerator/Co-Processor - Performance Share



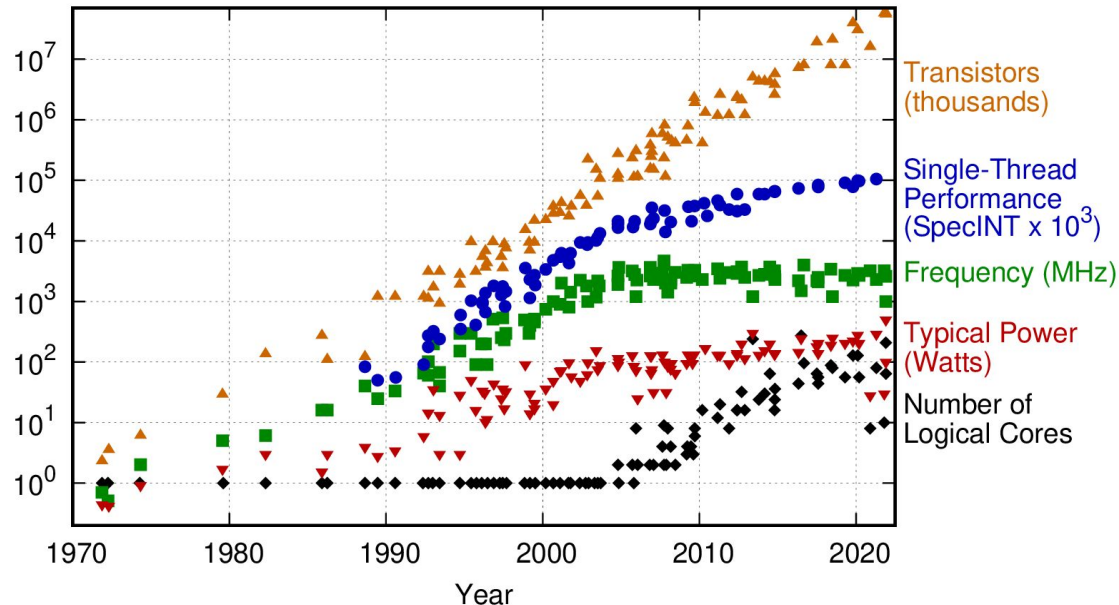
<https://top500.org/statistics/overtime/>

Need for Speed

Computational Power in scientific computing has been driven by the ability to fit more and more transistors in the same chip size.

As time went on, more powerful chips resulted in faster simulation and more science with little effort. All you had to do was buy the latest CPU.

50 Years of Microprocessor Trend Data



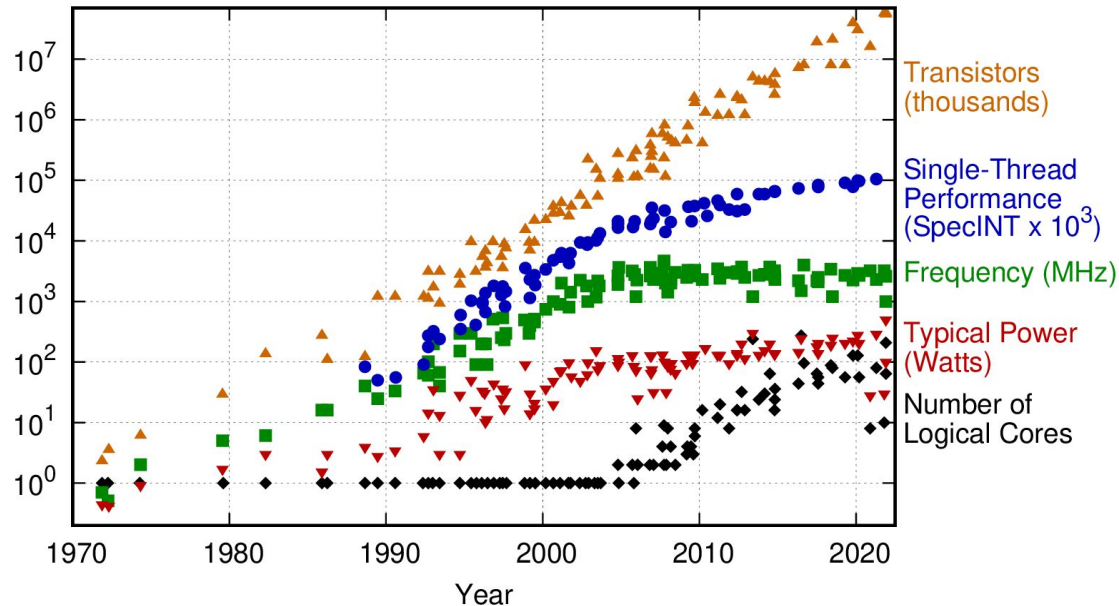
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

<https://github.com/karlrupp/microprocessor-trend-data>

Need for Speed

However, as transistor sizes reach the physical limit where quantum effects make them unreliable, we can no longer get free speed ups every generation.

50 Years of Microprocessor Trend Data

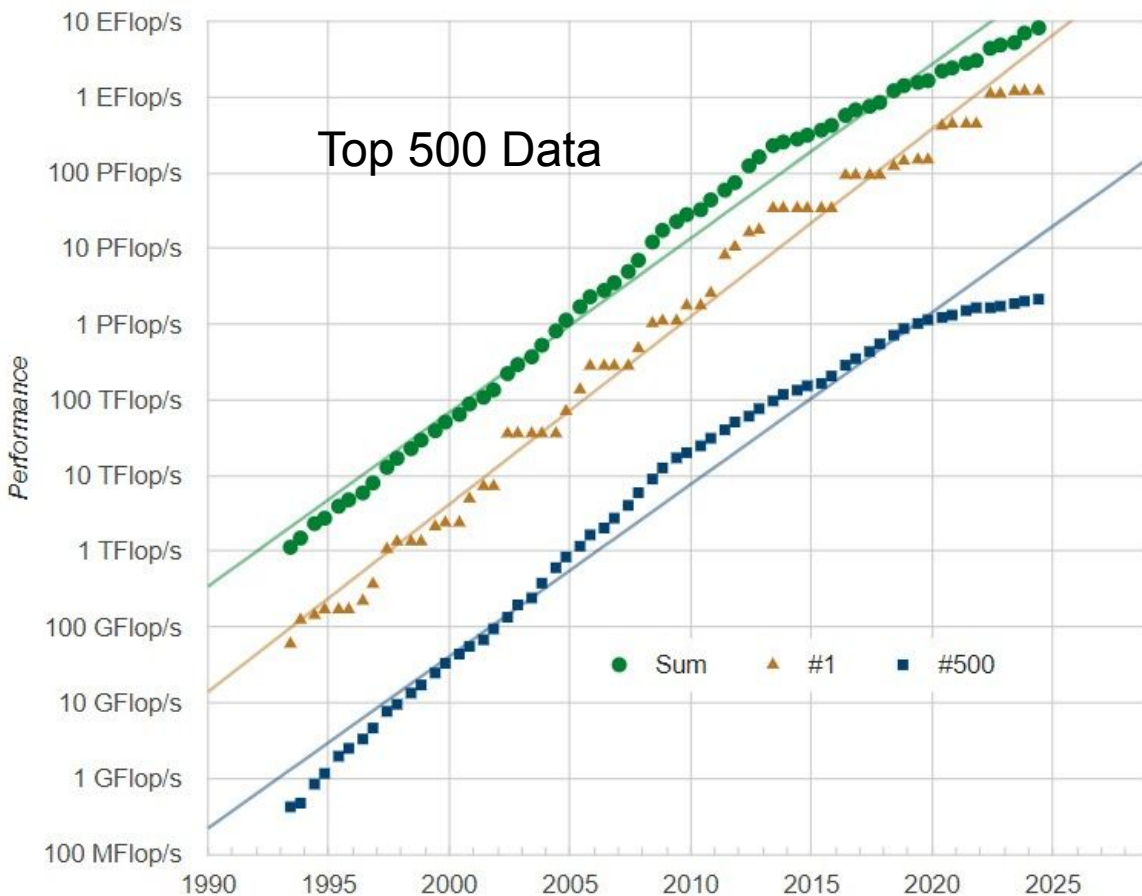


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

<https://github.com/karlrupp/microprocessor-trend-data>

Need for Speed

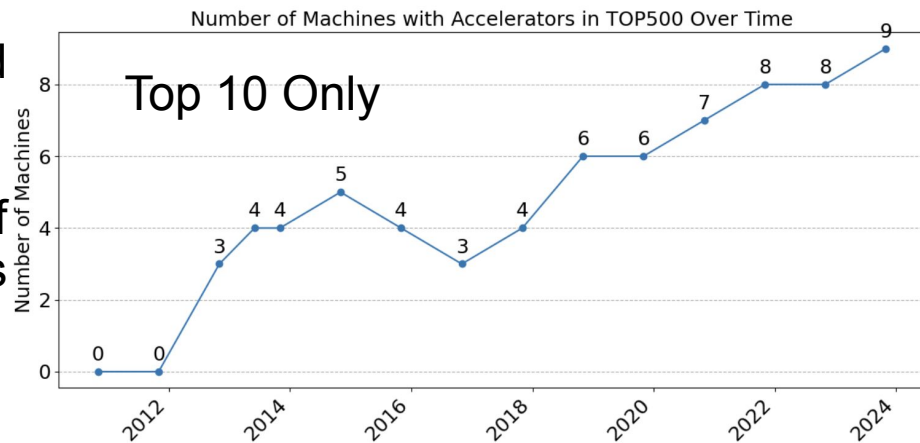
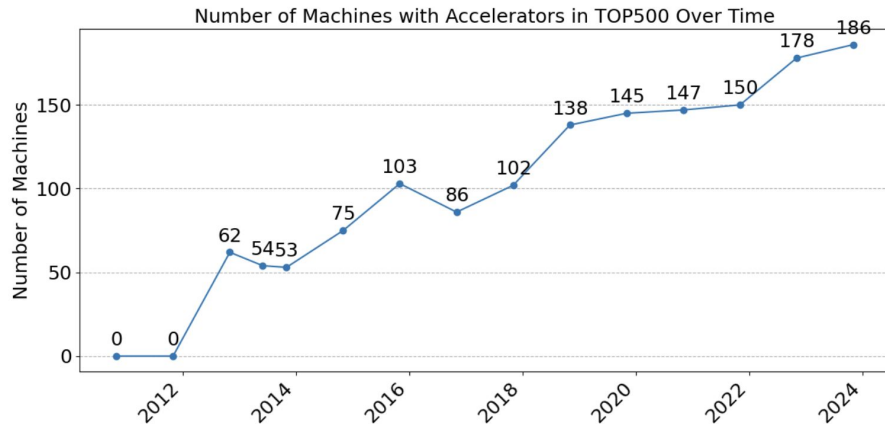
We can see similar trends in Top 500 data with linear increases giving way to asymptotes.



<https://www.nextplatform.com/2024/05/13/top500-supers-this-is-peak-nvidia-for-accelerated-supercomputers/>

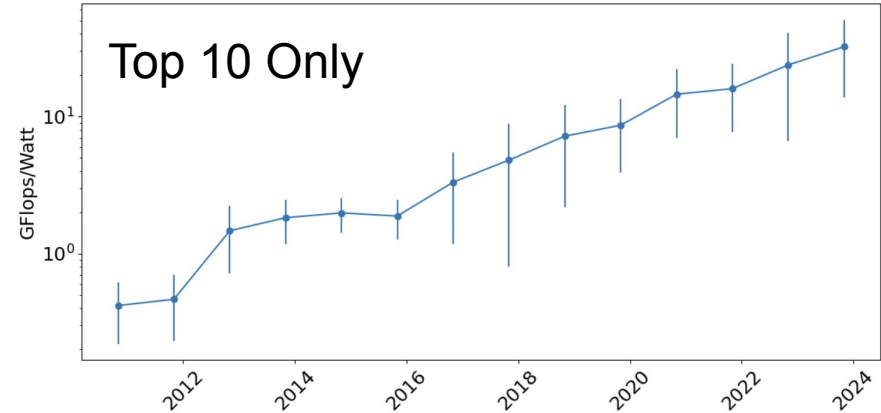
Accelerating Efficiency

- So how do we continue to improve?
- Generic CPUs are capable of doing everything not very well (a bit of an exaggeration)
- Custom accelerators are capable of doing a few things very well (not an exaggeration)
- Things like those calculations used in AI, which is driving industry
- And HPC follows industry
- Here we see the number of systems containing accelerators has continued to increase.



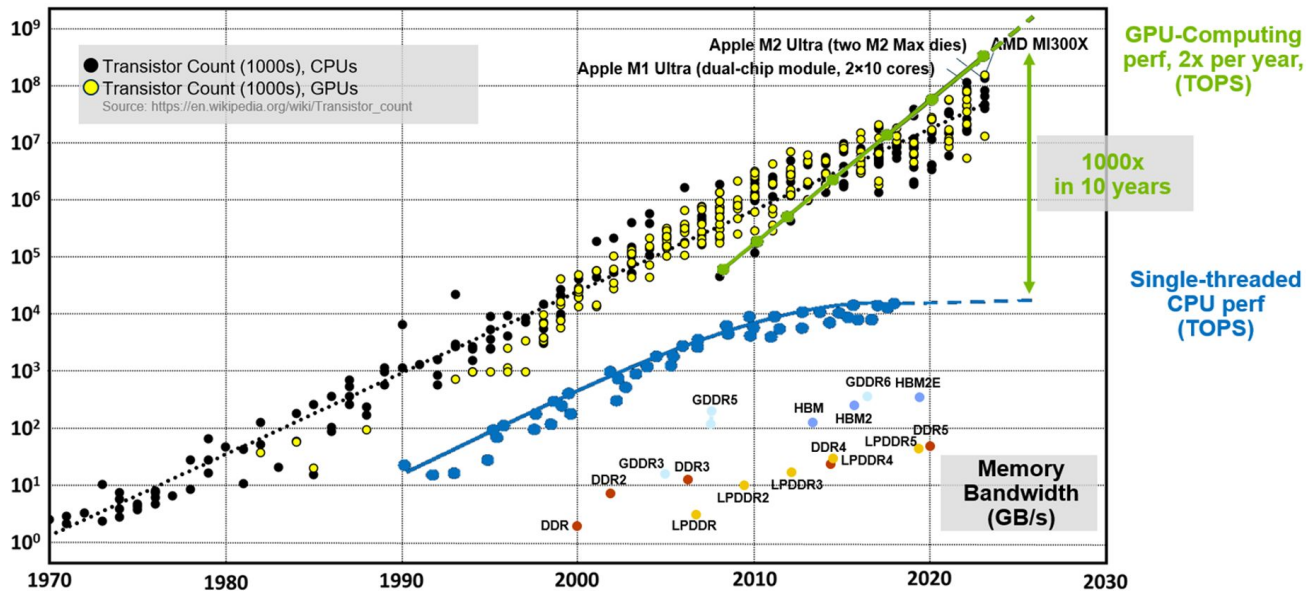
Accelerating Efficiency

- So how do we continue to improve?
- Generic CPUs are capable of doing everything not very well (a bit of an exaggeration)
- Custom accelerators are capable of doing a few things very well (not an exaggeration)
- Things like those calculations used in AI, which is driving industry
- And HPC follows industry
- Here we see the number of systems containing accelerators has continued to increase.



Power Efficiency continues to increase with more GPUs

Accelerators continue to Accel



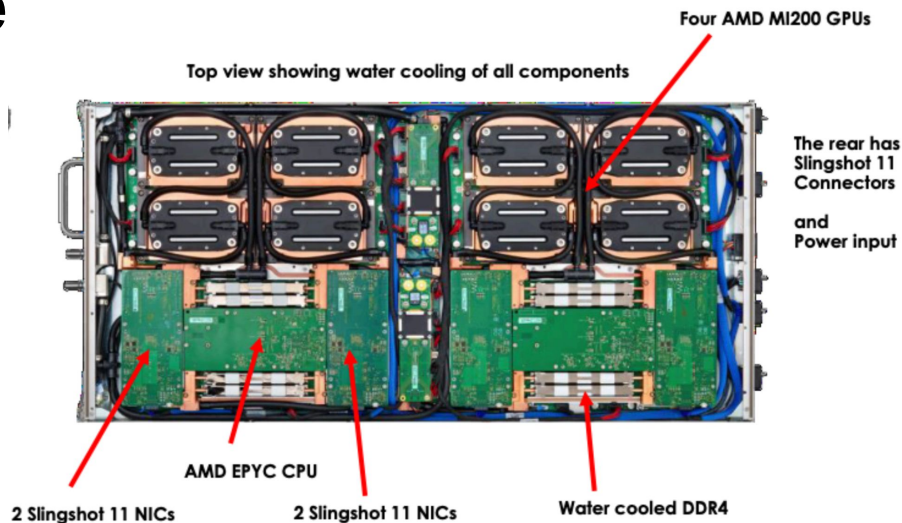
<https://semiengineering.com/ai-accelerator-architectures-poised-for-big-changes/>

Despite CPU performance stalling, GPU performance has managed to continue to improve.

Modern HPC Architecture

Frontier:

- 1 “blade” has 2 compute nodes.
- Each compute node has:
 - 1 AMD EPYC CPU
 - 4 AMD MI200 GPU
 - 4 network interface controllers (NIC)
- The high speed network is what distinguishes an HPC from a cluster.
 - Infiniband (Nvidia)
 - Slingshot (HPE)



Modern HPC Architectures

Aurora:

- 1 “blade” has 1 compute node
- 2 Intel Xeon CPU Max Series
- 6 Intel Data Center GPU Max Series
- 8 NICs per node (1/GPU, 1/CPU)

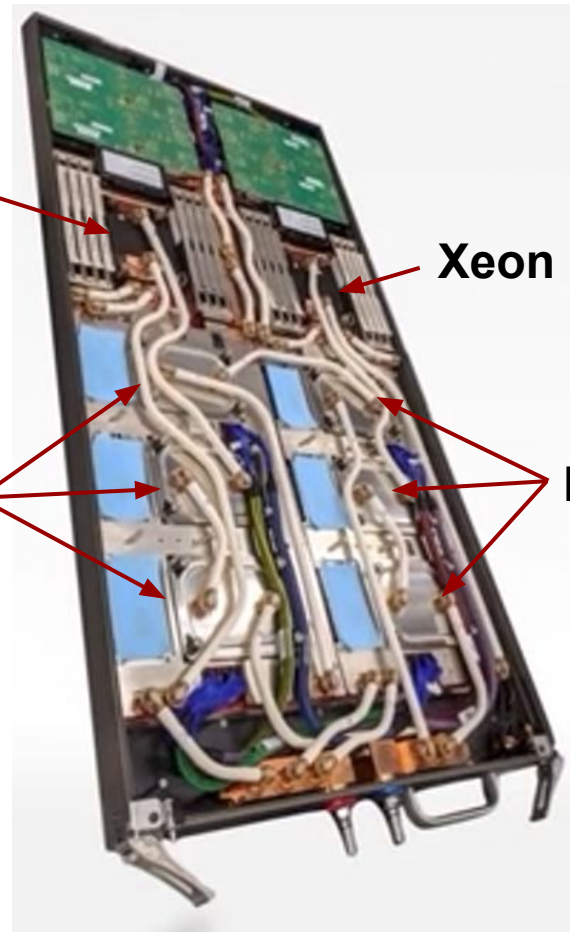
The tubing is for the water cooling.

Xeon

Xeon

PVC

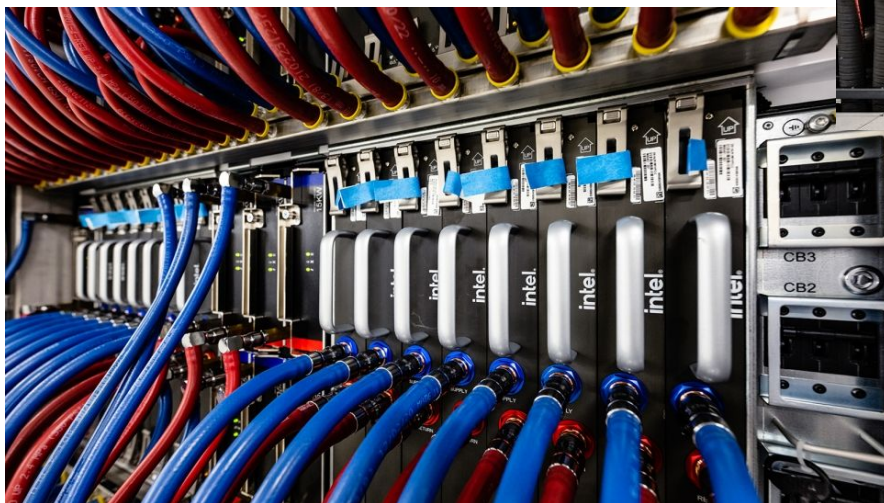
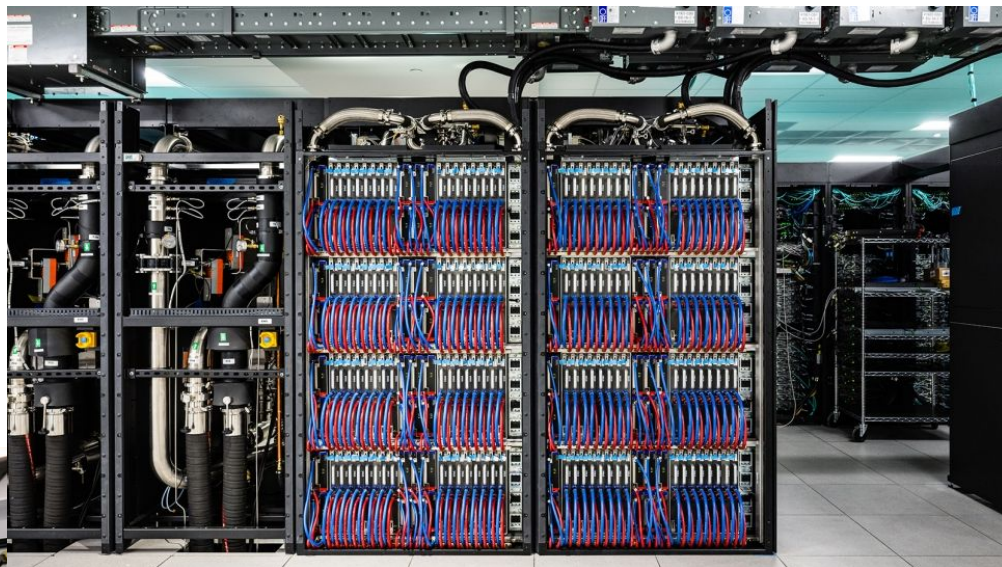
PVC



Modern HPC Architectures

Aurora

16 nodes in a row



4x16 nodes per rack

166 racks

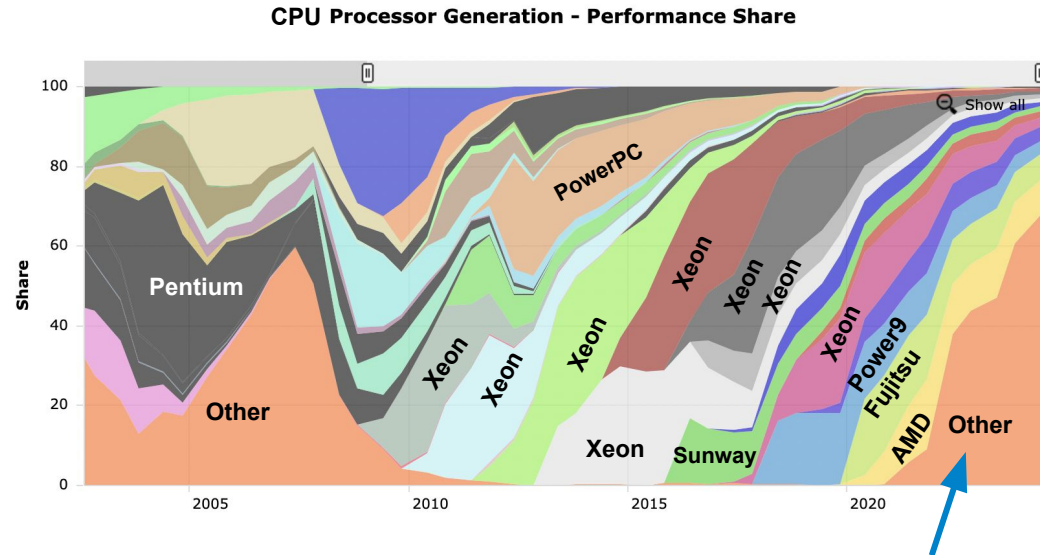
10,624 nodes

63,744 GPUs

21,248 CPUs

Looking to the Future: Diversity, Diversity


- Already have 3 major GPU makers, Nvidia, AMD, Intel, with three different methods for programming.
- Not everyone is happy paying \$40,000/GPU to Nvidia
- European HPC agency is developing their own architectures based on RISC-V
- China already has their own chips
- Microsoft recently announced development of their own chips
- Meta wants to dev RISC-V
- Apple Silicon exists
- There's a lot of "other" in the pipeline










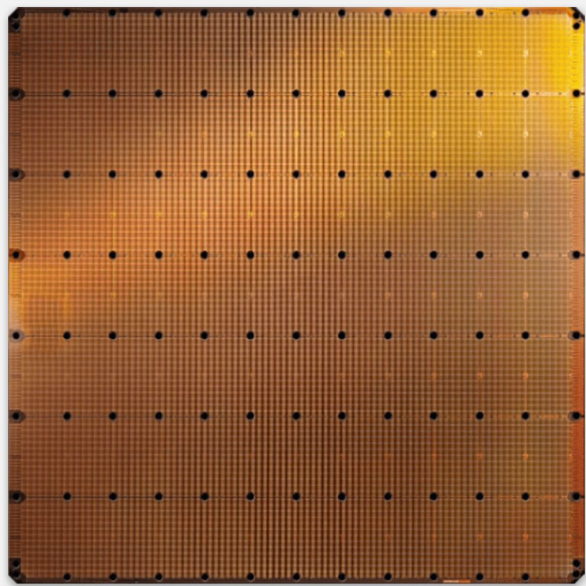
AI Accelerators

AI Accelerators are very custom chips especially made to run AI training and inference.

Cerebras is in the process of deploying multiple data centers for customers.

G42 Condor Galaxy 1 AI Supercomputer 

-  **64** CS-2 nodes
-  **54 million** AI cores
-  **4 exaFLOPS** AI compute at FP16
-  **82 TB** parameter memory
-  **388 Tbps** internal bandwidth
-  **72,704** AMD EPYC™ cores
-  **10 days** to first training run



Cerebras WSE-2
46,225mm² Silicon
2.6 Trillion transistors



Largest GPU
826mm² Silicon
54.2 Billion transistors



Now vendors are making new chips that merge GPU+CPU capabilities into one package.

Some include “Neural Cores” or other kinds of tensor math accelerators.

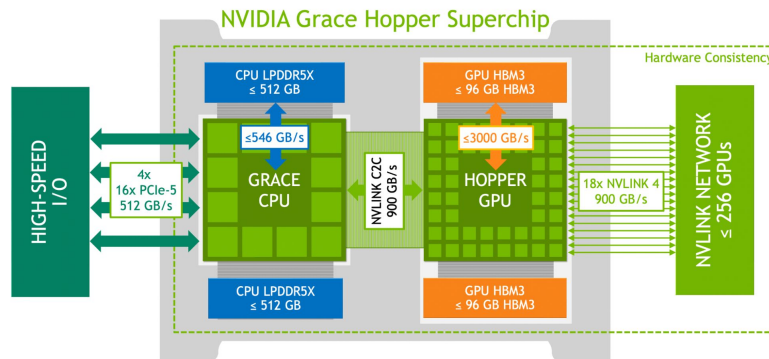
Again, industry is driven by investment, investment is chasing AI, AI needs accelerators.

Up to 128GB of unified memory
92 billion transistors

16-core CPU
12 performance cores
4 efficiency cores
Up to 80% faster than M1 Max
Up to 50% faster than M2 Max

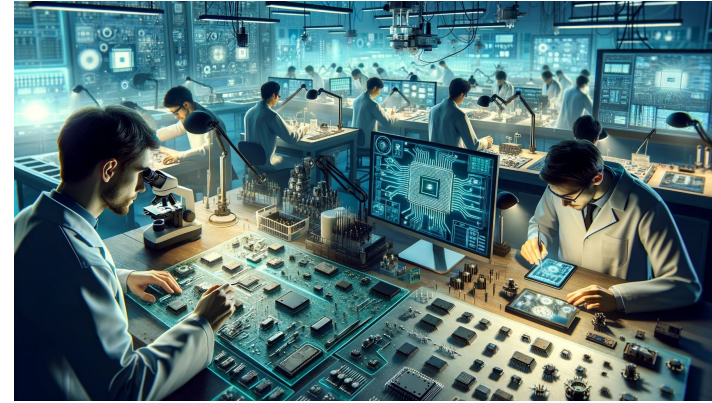
40-core GPU
Next-generation architecture
Dynamic Caching
Mesh shading
Ray tracing
Up to 50% faster than M1 Max
Up to 20% faster than M2 Max

The diagram shows a grid-like layout of the M3 Max chip with different colored regions representing the CPU and GPU components.



So what can we expect to see?*

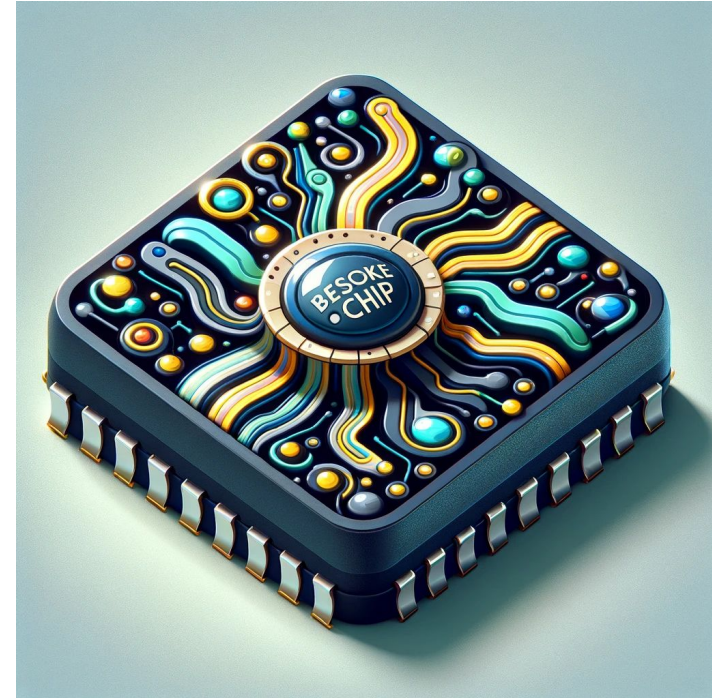
- I think we are in a period of industrial experimentation.
- HPCs will likely follow some, but not all.
- HPCs will have a diverse architectural makeup.
- Eventually (5-10yrs), some settling will occur but our computing landscape will look very different.



*By reading these predictions you agree not to hold the speaker responsible for any erroneous outcomes.

So what can we expect to see?*

- I think we are in a period of industrial experimentation.
- HPCs will likely follow some, but not all.
- HPCs will have a diverse architectural makeup.
- Eventually (5-10yrs), some settling will occur but our computing landscape will look very different.
- Instead of Chip Maker X producing a single D2000 model accelerator, there will be a model D2000 for AI, one for finance, one for bitcoin, one for the office users, etc.
- The silicon real estate will be optimized toward a subset of computational tasks.



*By reading these predictions you agree not to hold the speaker responsible for any erroneous outcomes.

Software



U.S. DEPARTMENT OF
ENERGY

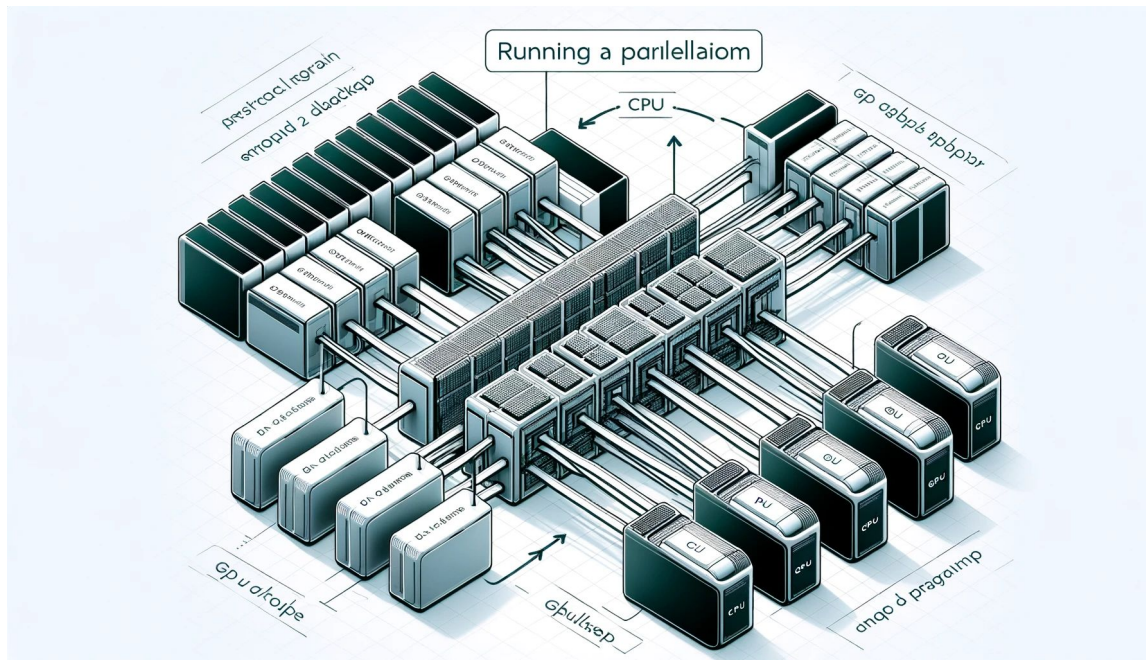
Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

Argonne  | **ALCF**
NATIONAL LABORATORY

Parallel Programming & Scaling

Writing software using parallelism is important to effectively utilize an HPC. Without it, one does not benefit from the architecture.

This is what GPT-4 thinks it is.



Parallel Programming: Concurrency

Process Parallelism

```
import multiprocessing

# Function to increment the counter
def increment_counter(counter):
    for i in range(1000):
        counter.value += 1

# Shared counter (using Value)
counter = multiprocessing.Value('i', 0)

# Create processes
processes = []
for i in range(10):
    process = multiprocessing.Process(target=increment_counter,
                                     args=(counter,))
    processes.append(process)

# Start processes
for process in processes:
    process.start()

# Wait for all processes to complete
for process in processes:
    process.join()

print(f"Final counter value (processes): {counter.value}")
```

Thread Parallelism

```
import threading

# Shared counter
counter = 0

# Lock for synchronizing access to the counter
lock = threading.Lock()

# Function to increment the counter
def increment_counter():
    global counter
    for i in range(1000):
        with lock:
            counter += 1

# Create threads
threads = []
for i in range(10):
    thread = threading.Thread(target=increment_counter)
    threads.append(thread)

# Start threads
for thread in threads:
    thread.start()

# Wait for all threads to complete
for thread in threads:
    thread.join()

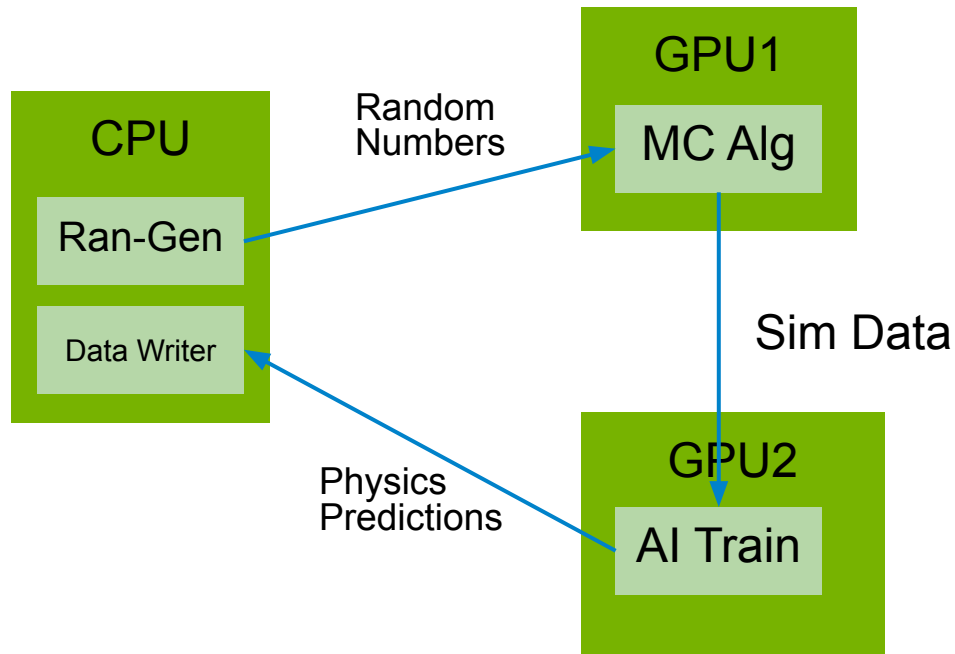
print(f"Final counter value (threads): {counter}")
```

Parallel Programming: Task Parallelism

Task Parallelism distributes tasks across available hardware.

Example of node-level, and many nodes split up doing different tasks.

As an HPC user, it would be important to balance resources to ensure effective utilization.

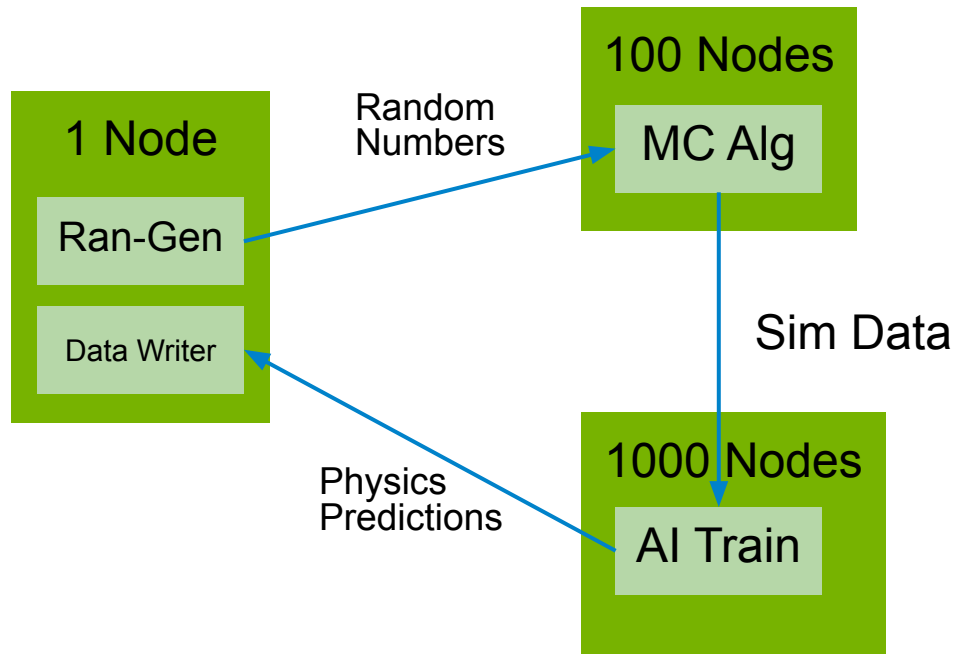


Parallel Programming: Task Parallelism

Task Parallelism distributes tasks across available hardware.

Example of node-level, and many nodes split up doing different tasks.

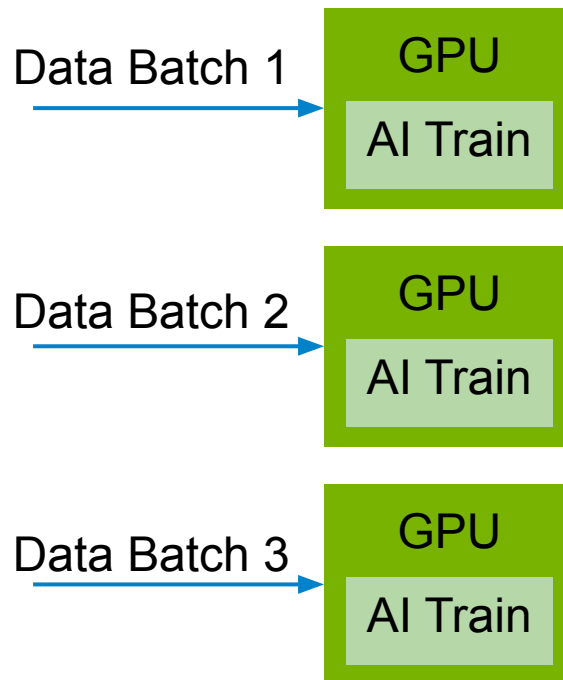
As an HPC user, it would be important to balance resources to ensure effective utilization.



Parallel Programming: Data Parallelism

Data Parallelism executes the same task on different data distributed across devices.

Data Parallel training of an AI model is depicted in this diagram.



Parallel Programming

Tools we use to span many nodes via the network:

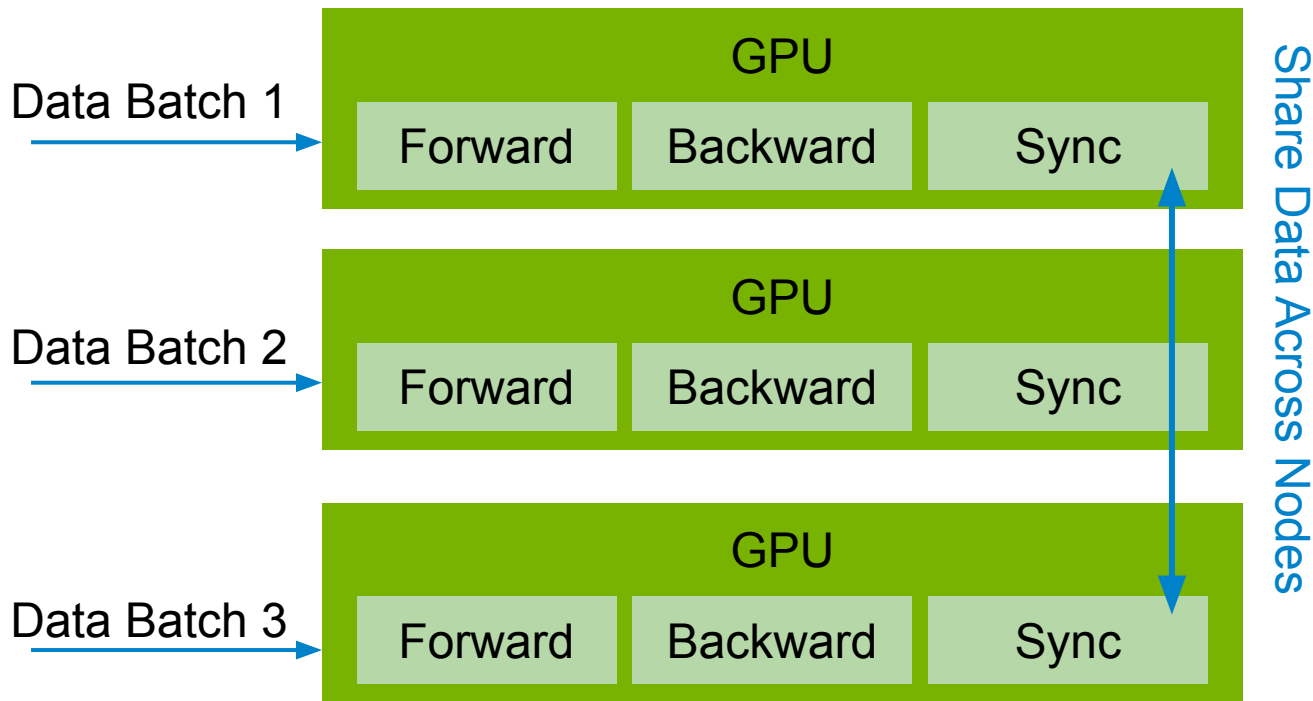
- For parallelism within a node:
 - Python (threading, multiprocessing)
 - C++: OpenMP, SYCL, Kokkos, CUDA/HIP
- For parallelism across nodes:
 - MPI (Message Passing Interface)
 - mpi4py
 - PyTorch DDP, Horovod

This is what you want to use on an HPC to make effective use of the system.

Data Parallel AI Training

PyTorch DDP
Horovod

Both
implement
built in easy
data parallel
training;
essentially no
experience
needed.



Data Parallelism with MPI

- Here is a simple example of using MPI in python.
- In MPI speak:
 - If you launch 50 processes
 - Each process gets a unique ID number 0-49 called “rank”
- MPI provides functions to share information across these ranks
- All-reduce, Broadcasts, Scatters, and many others.

```
from mpi4py import MPI
import numpy as np

def main():
    comm = MPI.COMM_WORLD
    rank = comm.Get_rank()
    size = comm.Get_size()

    # Define array size
    array_size = 1000
    array = None

    if rank == 0:
        # Initialize the array with random numbers on the
        root process
        array = np.random.random(array_size)
        print(f"Original array (first 10 elements):
        {array[:10]}")

        # Determine the portion of the array each process will
        handle
        local_size = array_size // size

        # Create a buffer for the local portion of the array
        local_array = np.zeros(local_size)

        # Scatter the array to all processes
        comm.Scatter(array, local_array, root=0)

        # Each process computes the sum of its local array
        local_sum = np.sum(local_array)

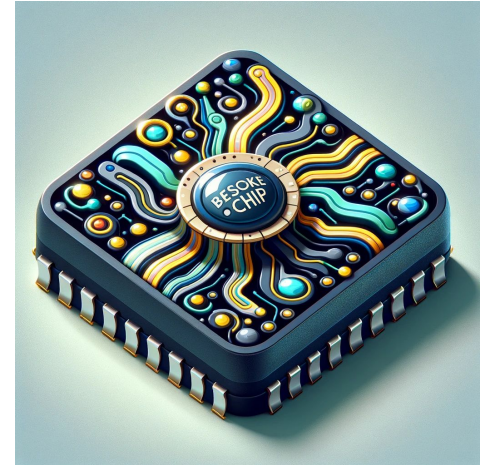
        # Gather all local sums to the root process
        total_sum = comm.reduce(local_sum, op=MPI.SUM, root=0)

        if rank == 0:
            print(f"Total sum: {total_sum}")

if name == "__main__":
    main()
```

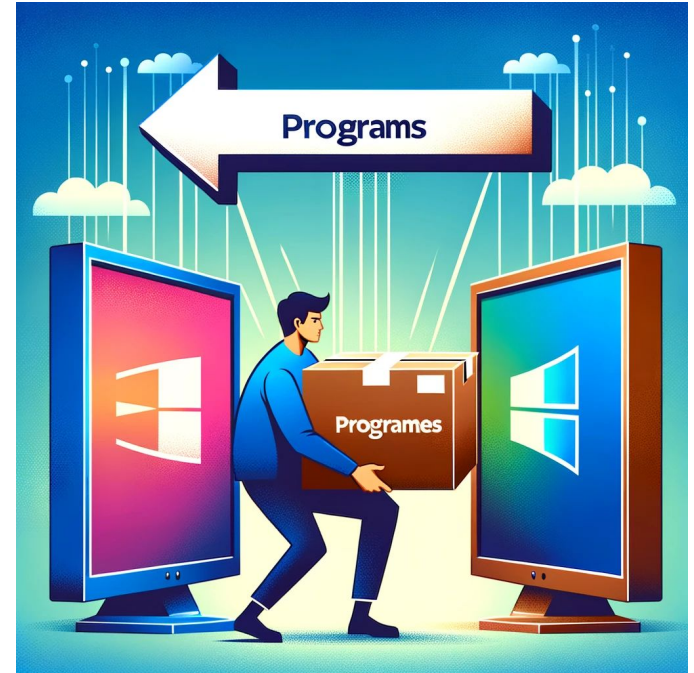
Software in a Bespoke Future

- How do we deal with a “Besoke Chip” future?
- Each chip traditionally comes with a custom programming library:
 - CUDA, HIP, ROCm, SYCL, OpenCL
- Could we write our algorithms in every language?
 - Increased software support effort
 - Software tends to become hard to read
- Could we chose one technology and use it forever?
 - Chip prices change as demand changes
 - Choosing incorrectly leads to bad things



Portability in Programming

- People are (have been) planning for this outcome by designing portable frameworks
 - C++: Kokkos, SYCL, OpenCL, std::par
 - Python: Tensorflow, PyTorch, JAX
- These programming libraries allow you to write your algorithm once, then they manage the creation or use of architecture specific code.
- C++ version do this at compile time typically.
- Python contains both compiled versions in the library and are called when needed.



Software Abstraction Support

- All portable solutions rely on vendors (SYCL, OpenCL, std::par), someone else (Kokkos), or both (Tensorflow, PyTorch, JAX) to create and support their implementations.
- Tensorflow & PyTorch are simply C++ libraries with a Python interface.
- The Tensorflow & PyTorch devs first supported Nvidia devices, though now Nvidia is directly contributing to the CUDA portions of the code.
- AMD and Intel joined in to add support for their devices.
- If software moves away from these tools, vendors likely will too.
- Kokkos was developed by the DOE Exascale Program and is developed by scientists to support portability. It is not a vendor driven activity.



Science Examples



U.S. DEPARTMENT OF
ENERGY

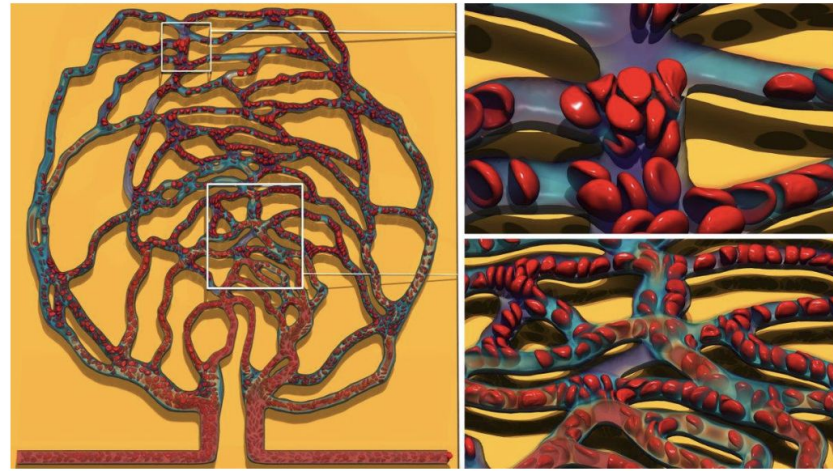
Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

Argonne  | **ALCF**
NATIONAL LABORATORY

Modeling Blood Flow

Complex Fluid Dynamics

- Modeling deformable blood cells traversing capillaries with complex geometries.
- One of the first times using 3D model instead of 1D model.
- Improving blood transport models will impact the ability to analyze cardiovascular disease risks in patients.



A snapshot of complex flow with cell suspensions in the retina network showing the transport of cells in different regions of the network. The blue/green coloring indicates the flow velocity within the network and red blood cells are colored red. (Image courtesy of Kacper Ostalowski (NIU), Jifu Tan (NIU) and Joseph A. Insley (ALCF).)

Contact PI: Jifu Tan (NIU, Dept. Mech. Eng.)

ASCR Allocation PI: Jifu Tan (NIU, Dept. Mech. Eng.)

ASCR Program/Facility: DD/ALCF

ASCR PM: Saswata Hier-Majumder

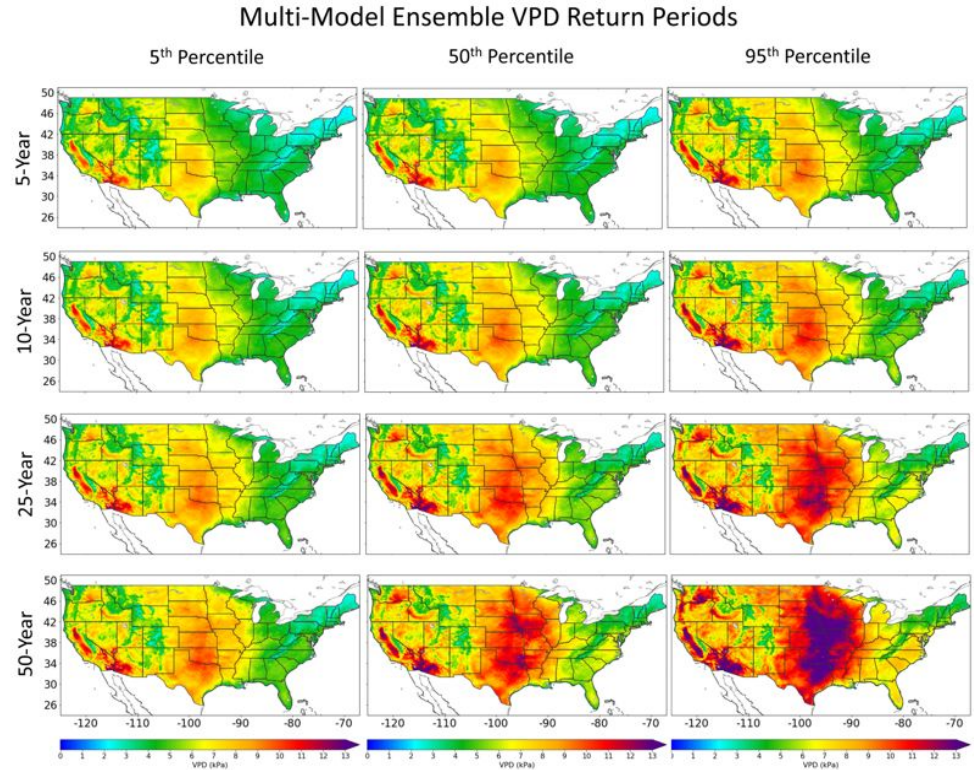
Date submitted to ASCR: May 15, 2023

Publication(s) for this work: K. Ostalowski and J. Tan., Phys. Fluids.

34, 041912 (2022)

Drought Prediction

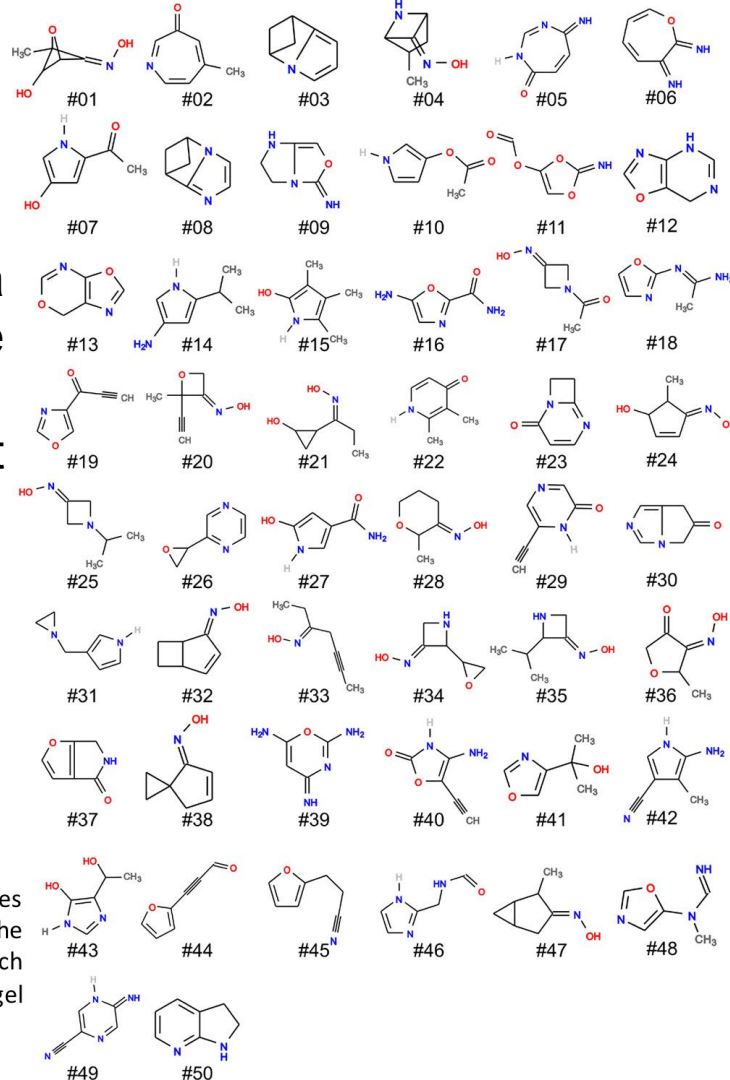
- Based on analysis of a large community dataset and projections from three Global Climate Models
- Identifies short-term droughts and predicts future extreme droughts
- Including “flash drought” events with a quick onset period that could be as short as few weeks



The 5, 10, 25, and 50-year return periods in the United States for the 5th, median (50th), and 95th percentile of the sampled model ensemble for VPD return periods. (Image courtesy of Brandi L. Gamelin of Argonne National Laboratory)

AI for Molecular Energetics

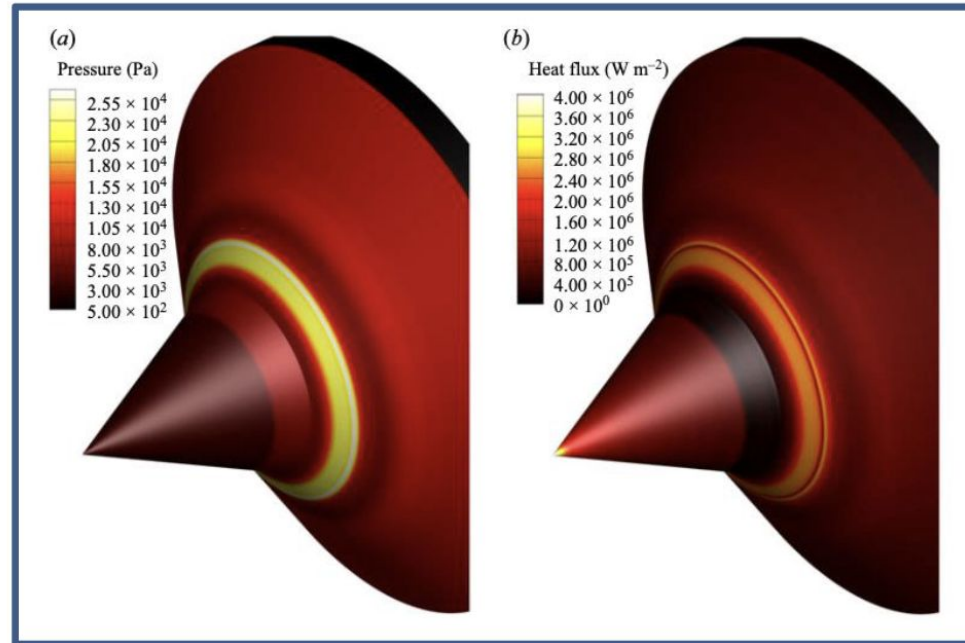
- Using AI, Quantum Chemists were able to predict the energetics of larger molecules at a fraction of the computing cost, while maintaining chemical accuracy.
- This new method could have a significant impact on fields such as drug discovery, materials science, and energy research.



Figures from left to right show the structures of 50 test molecules made up of nine heavy atoms and the increase of accuracy as the size of the molecular fragment (A_{mon}) is increased to reach chemical accuracy. (Image courtesy of A. Benali (ANL), J.T. Krogel (ORNL), B. Huang (U. Vienna), and o.A. von Lilienfeld (U of T)).

Hypersonic Flight

- The fluid dynamics of air when objects travel at $>5x$ the speed of sound is not well modeled.
- This team aims to use fundamental quantum mechanical interactions to model the behavior of this system.



Three-dimensional view of the surface pressure and heat flux on the double cone geometry. (Image courtesy of P. Valentini)

Conclusion



U.S. DEPARTMENT OF
ENERGY

Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

Argonne  | **ALCF**
NATIONAL LABORATORY

Wrapping Up

- Computing architectures are growing in diversity.
 - Things will likely settle into a smaller subset of “winning” architectures.
 - Software is trying to keep up to facilitate ease of use, but remain performant across devices.
 - Knowledge of underlying hardware is still a benefit to the programmer, but with Python supplanting C++ as the commonly learned language amongst scientists, this knowledge becomes more obscure.
 - In many ways we are dependent on vendors to implement portable solutions for their hardware, but this isn’t naturally in their interest.
-
- Our goal in scientific computing is always to enable the most science output, while reducing complexity where possible.