Why Use GPU Accelerators

An Introduction to the Pros and Cons of GPU Computing

Daniel Howard *- dhoward@ucar.edu*

CISL HPCD, Consulting Services Group

NCAR
UCAR

February 17th, 2022

Workshop Etiquette

- Please mute yourself and turn off video during the session.
- Questions may be submitted in the chat and will be answered when appropriate. You may also raise your hand, unmute, and ask questions during Q&A at the end of the presentation.
- By joining today, you are agreeing to UCAR's Code of Conduct
- Recordings & other material will be archived & shared publicly.
- Feel free to follow up with the GPU workshop team via Slack or submit support requests to support.ucar.edu
  - **Office Hours**: Asynchronous support via Slack or schedule a time

Workshop Series and Logistics

- Scheduled biweekly through August 2022 (short break in May)

- Sequence of sessions detailed on main webpage
  - Full workshop course description document/syllabus
  - Useful resources for independent self-directed learning included

- Registrants may use workshop's Project ID & Casper core hours
  - Please only **submit non-production, test/debug scale jobs**
  - For non-workshop jobs, request an allocation. Easy access startup allocations may be available for new faculty and graduate students.
  - New NCAR HPC users should review our HPC Tutorials page

- Interactive sessions will share code via GitHub and JupyterHub notebooks. More details will be shared prior to these sessions.

GPU Community Engagement

Below are recommended community resources

- Join NCAR GPU Users Slack and #gpu_workshop_participants

- Consider joining other Slack communities or online spaces
  - OpenACC and GPU Hackathon Slack workspace (NVIDIA managed)
  - If you're excited about Julia, they have a Slack and #GPU channel
  - NCAR GPU Tiger Team for latest updates and future directions at NCAR
  - Watch Stackoverflow tags for OpenACC, OpenMP, CUDA, or others

- Prepare an application for an upcoming GPU Hackathon

Find your GPU community! Key to modern science is collaboration!

Overview

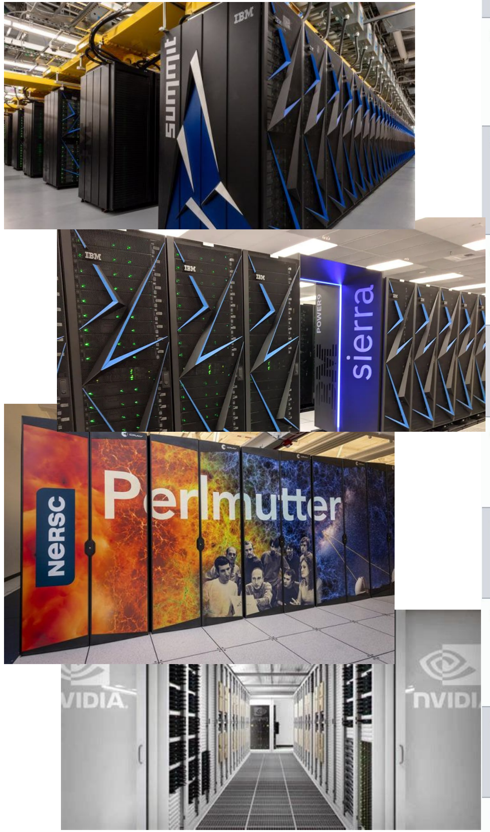
- Achieving High Performance Computing with GPUs
- History of GPU Computing
- Trends in GPU and CPU Performance
  - Power consumption? Acceleration? Availability?
- Vector and Thread Processing - CPU vs GPU
- GPU Software Paradigms and Community Support
- How to Approach GPU Projects or GPU Refactoring

GPUs Enable Exascale!

- 7 of top 10 supercomputers leverage GPUs from Top500
- 9 of top 10 power Green500

- GPUs enable ~3.5x more FLOPs/Watt efficiency
  - Lower Costs
  - Eases Access

- GPUs are designed inherently parallel
- CPU cores designed for complex serial tasks

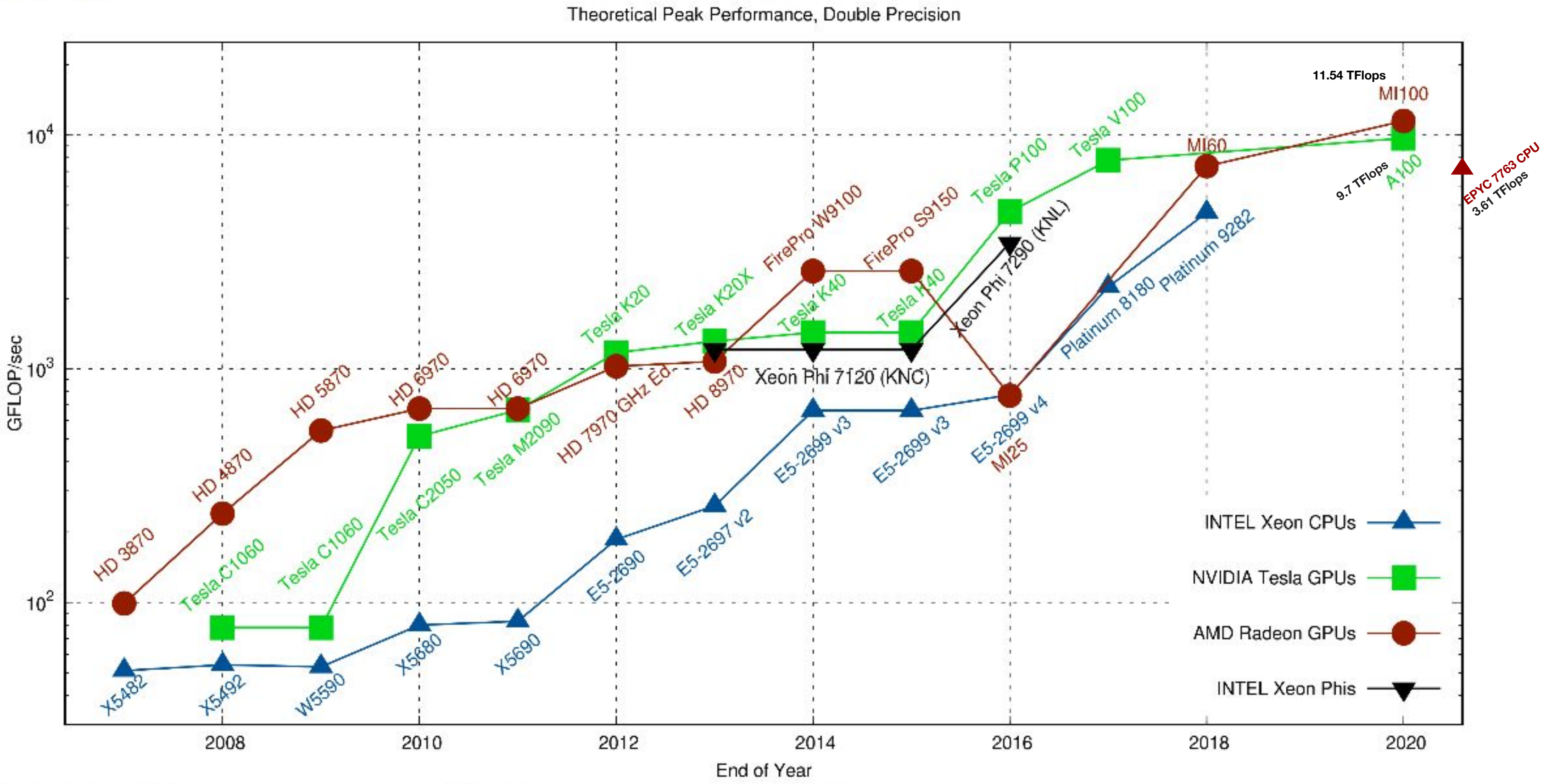


Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Perlmutter** - HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10, HPE DOE/SC/LBNL/NERSC United States	761,856	70,870.0	93,750.0	2,589
6	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	79,215.0	2,646
7	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
8	**JUWELS Booster Module** - Bull Sequana XH2000 , AMD EPYC 7402 24C 2.8GHz, NVIDIA A100, Mellanox HDR InfiniBand/ParTec ParaStation ClusterSuite, Atos Forschungszentrum Juelich (FZJ) Germany	449,280	44,120.0	70,980.0	1,764
9	**HPC5** - PowerEdge C4140, Xeon Gold 6252 24C 2.1GHz, NVIDIA Tesla V100, Mellanox HDR Infiniband, DELL EMC Eni S.p.A. Italy	669,760	35,450.0	51,720.8	2,252
10	**Voyager-EUS2** - ND96amsr_A100_v4, AMD EPYC 7V12 48C 2.45GHz, NVIDIA A100 80GB, Mellanox HDR Infiniband, Microsoft Azure	253,440	30,050.0	39,531.2	

Nov 2021 Top500

Status Quo Across Architectures

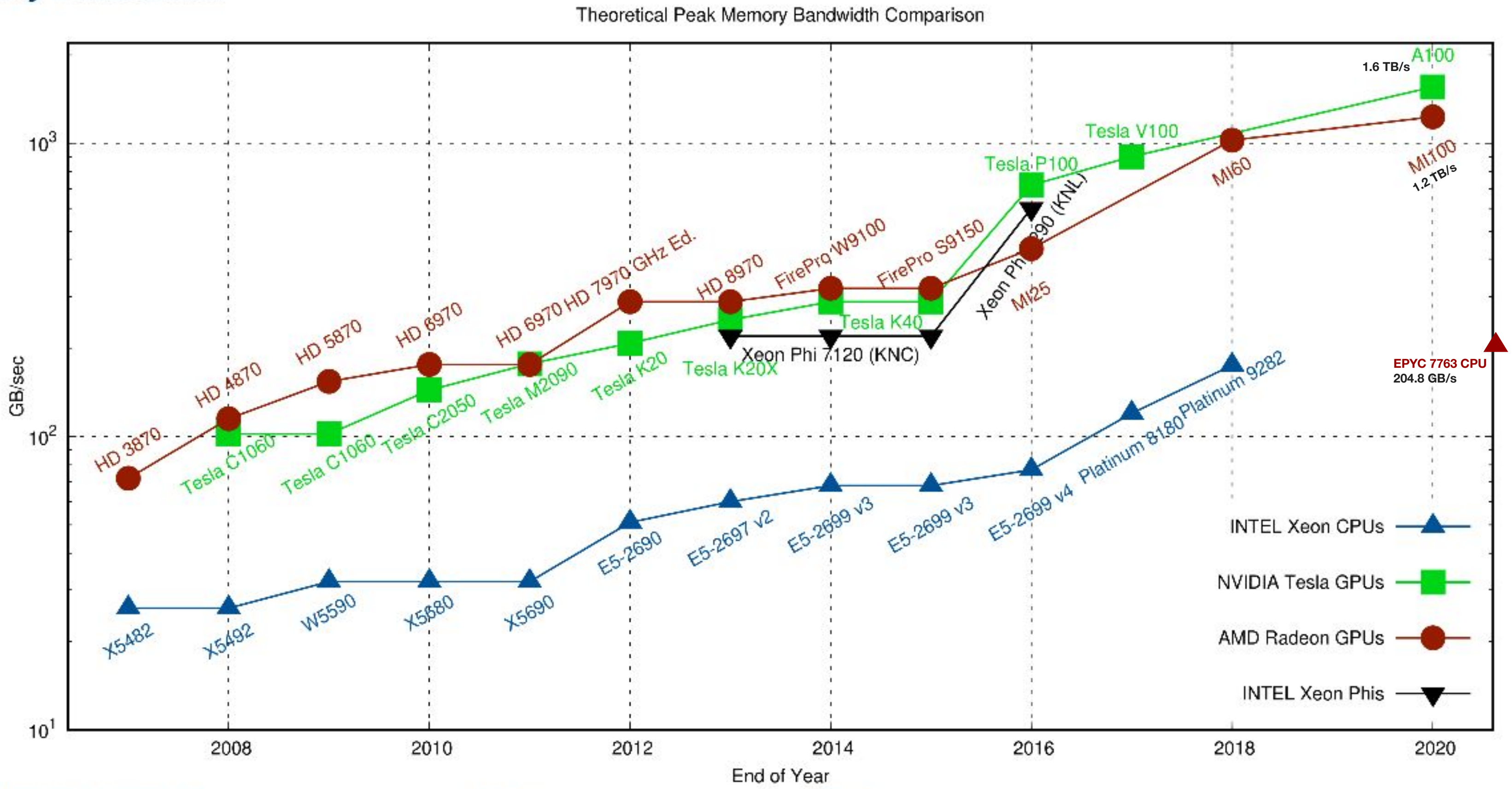
Performance



Karl Rupp, TU Wien, GitHub

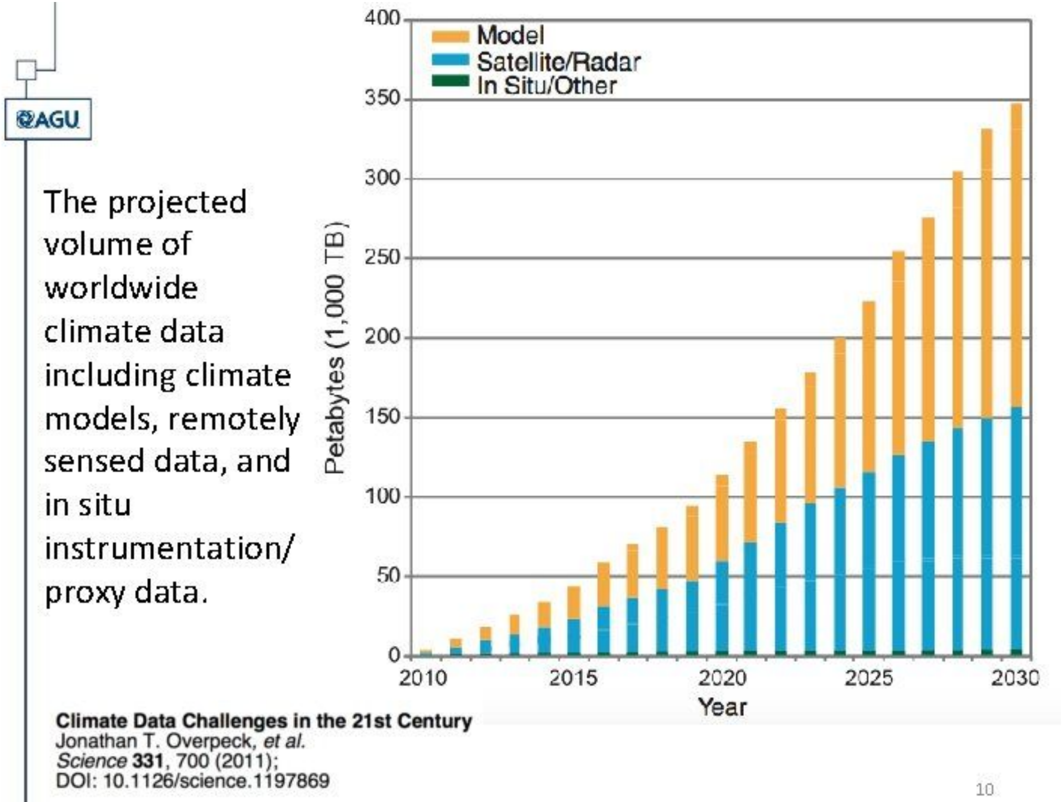
Status Quo Across Architectures

Memory Bandwidth



Karl Rupp, TU Wien, GitHub

Data is the Future of Scientific Discovery



https://www.science.org/doi/10.1126/science.1197869

- Larger amounts of data to process in Earth Sciences
- GPUs can provide the needed bandwidth to process this data
- GPUs excel in ML & may "avoid" physics compute cost
- Note: General Purpose GPU (GPGPU) Programming allow use beyond GPU specialties
  - Matrix operations (AI)
  - Graphics (per pixel ops)

Significant GPU Adoption and Exploration in Earth Sciences

Global:

Model	Organizations	Funding Source
E3SM, SCREAM	US DOE: ORNL, SNL	E3SM, ECP
HOMEXX, SCREAM		
MPAS-A	NCAR, UWyo, KISTI, IBM	WACA II
FV3/UFS	NOAA	SENA
NUMA/NEPTUNE	US Naval Res Lab, NPS	ONR
IFS	ECMWF	ESCAPE, US DOE
GungHo/LFRic	MetOffice, STFC	PSyclone
ICON	DWD, MPI-M, CSCS, MCH	PASC ENIAC
CLIMA / NUMA	CLiMA (NASA JPL, MIT, NPS)	Private, US NSF
FV3	Vulcan, UW/Bretherton	Private

Regional:

Model	Organizations	Funding Source
COSMO	MCH, CSCS, DWD	PASC GridTools
AceCAST-WRF	TempoQuest	Venture backed

NVIDIA Collaborations with Atmospheric Models (S. Posey, MultiCore10)

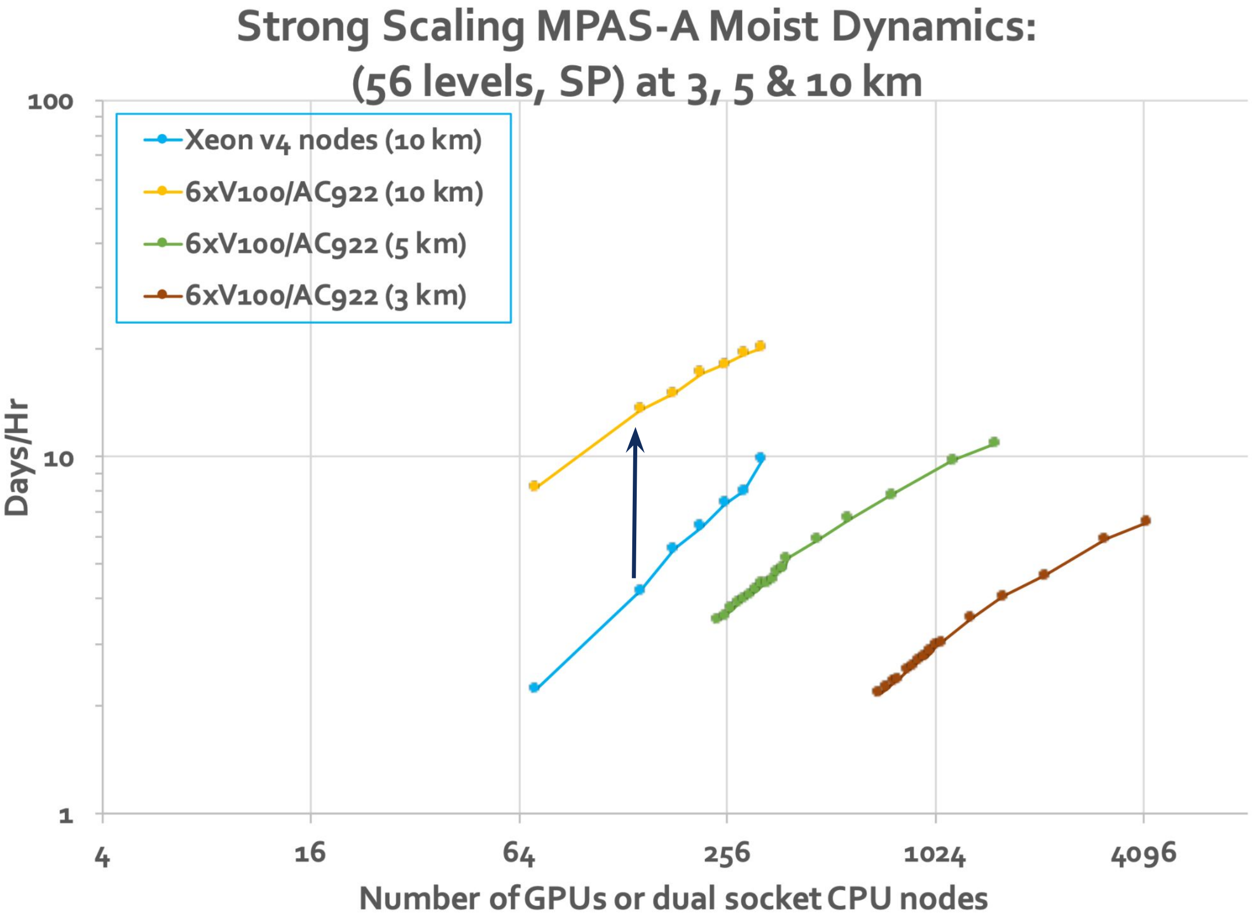
MPAS-A

An atmospheric model that solves the compressible non-hydrostatic equations in both global and regional configurations with variable resolution configurations

Blue (Xeon CPUs)

Gold (V100 GPUs)

Significant increase in simulated Days/Hr for 10km resolution case



Kumar, et al (Multicore10) and Randall, et al EarthWorks Proposal

History of GPU Computing

GPU Course of History

- 1978 - First GPGPU from Ikonas Graphics Systems

  A Graphics Processing and Raster Display for cockpit instrumentation (SIGGRAPH 78)

- 1986 - Tim Van Hook, solid modeling and ray tracing microcode (SIGGRAPH 86)
- 1994 - “GPU” term coined by Sony under PlayStation video game console
- 2001 - First matrix multiplication and PDE solvers run on GPUs

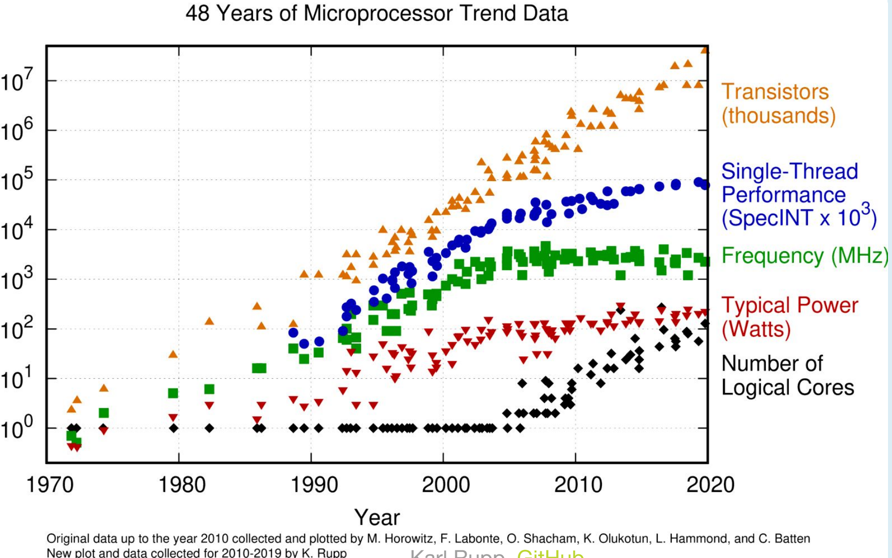
  NVIDIA GeForce 3 with programmable shaders and floating-point

  Mark Harris, “Real Time Cloud Rendering” and “Real Time Cloud Simulation and Rendering” 2003 Dissertation

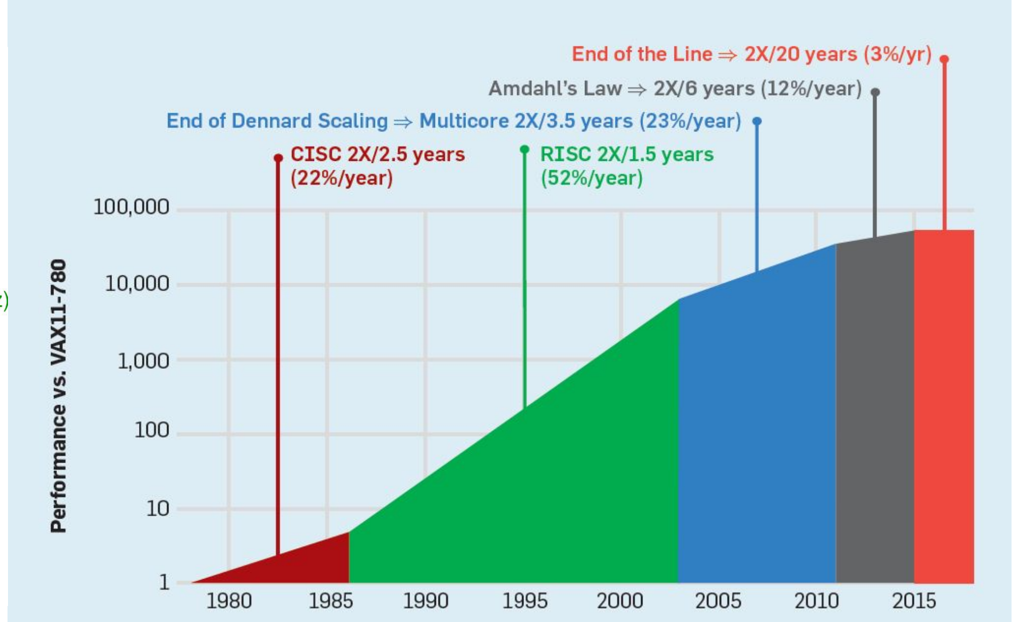
- 2002 - “GPGPU” term coined by Mark Harris
- 2007 - Release of CUDA
- 2009 - Release of OpenCL, Foundations laid for unified memory concept
- 2011 - OpenACC v1.0 “forks” from OpenMP
- 2020 - 25% of Top500 with NVIDIA GPUs

graphics-history.org/ikonas/ and M. Harris “A Brief History of GPGPU”

Faltering Moore's Law



Karl Rupp, GitHub



"A New Golden Age for Computer Architecture", Hennessy & Patterson ACM Communications

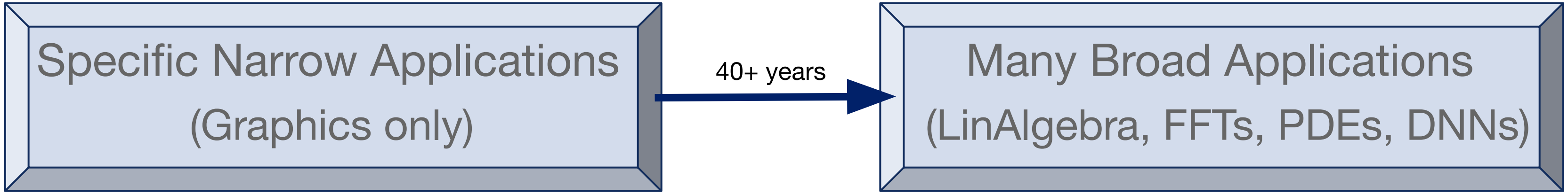
Transistor counts still increasing but had to switch to parallel processors in 2004

Diminishing returns on CPU performance forced greater adoption of GPUs

Why Is GPU History Important?

Understand CONTEXT & FUTURE DIRECTION of GPU development

Trust the LONG & ESTABLISHED communities around scientific use of GPUs

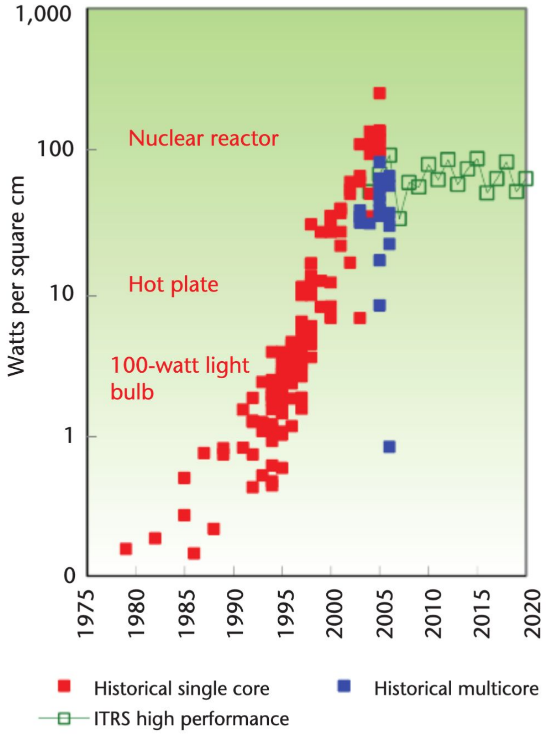


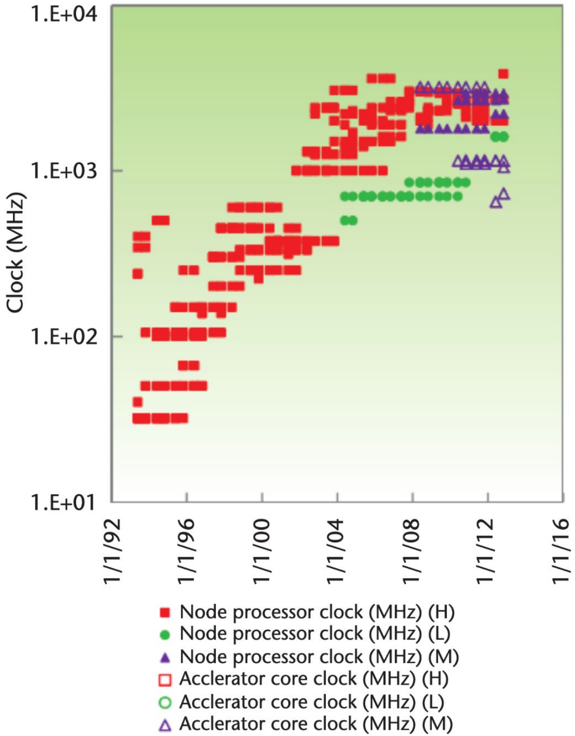
Many years of software abstraction to enable general developers to use GPUs.

No more microcode or low level languages required!

Trends in CPU and GPU Performance

Heat Dissipation and Performance





Why the switch to parallel processors in 2004?

- Higher clock frequencies led to higher heat output
- No longer able to reduce supply voltages

"Exascale Computing Trends" Kogge & Shalf, 2004 and LLNL slides

Energy Efficiency and Data Locality

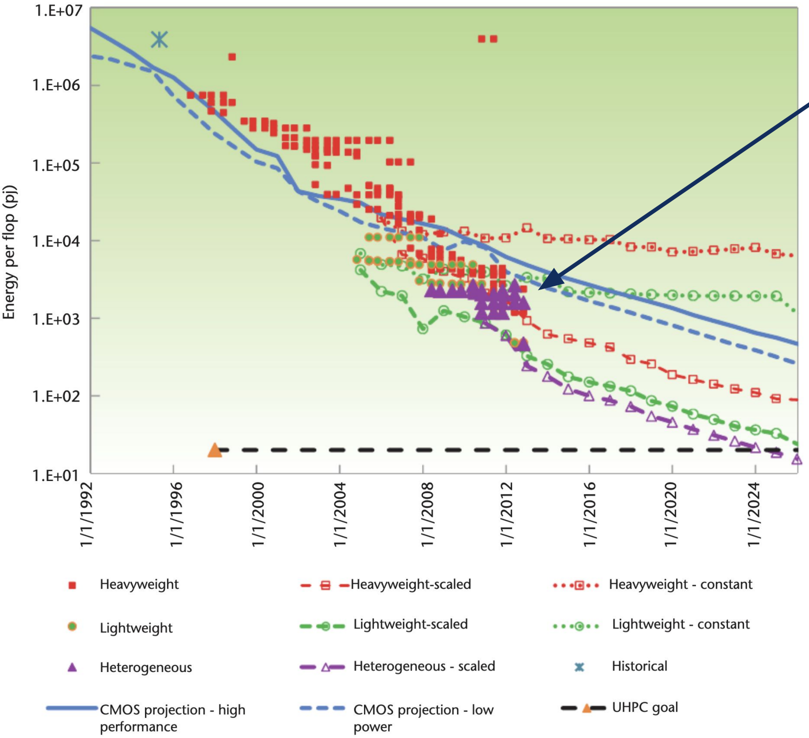


Figure 4. Performance. Projections in energy per flop show that only the hybrids have a chance of reaching exaflop performance within the 20-megawatt (MW) power budget by 2024, but with the caveat that they must improve efficiency to offer commensurate improvements in application performance. (Here, pJ = picojoules and UHPC = ubiquitous high-performance computing.)

<u>Heterogeneous (CPU+Accelerator)</u> fastest way predicted to achieve lowest energy use & high efficiency.

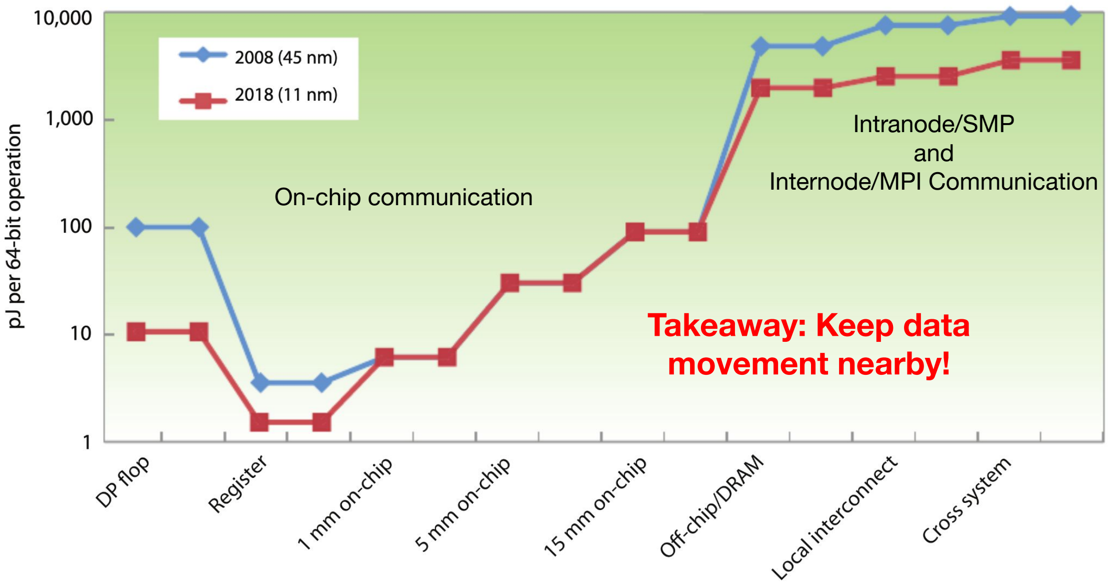


Figure 5. Data movement is overtaking computation as the most dominant cost of a system both in terms of dollars and in terms of energy consumption.

"Exascale Computing Trends" Kogge & Shalf, 2004 and LLNL slides

Data Locality Promotes Green Computing

Maintaining data locality, as inherent by design of GPU architectures, enables reduction in carbon emissions

- MPAS 10 km, 72 GPUs or nodes
  - CPU Cheyenne: 9.75 kg CO_2
  - GPU Summit: 2.26 kg CO_2
- MPAS 3 km, on 804 GPUs or nodes
  - CPU Cheyenne (est): 330 kg CO_2
  - GPU Summit: 87 kg CO_2

R. Loft, AMS 2020 - "Reducing the Footprint of MPAS Weather Modeling with GPUs"

Performance Efficiency of GPUs

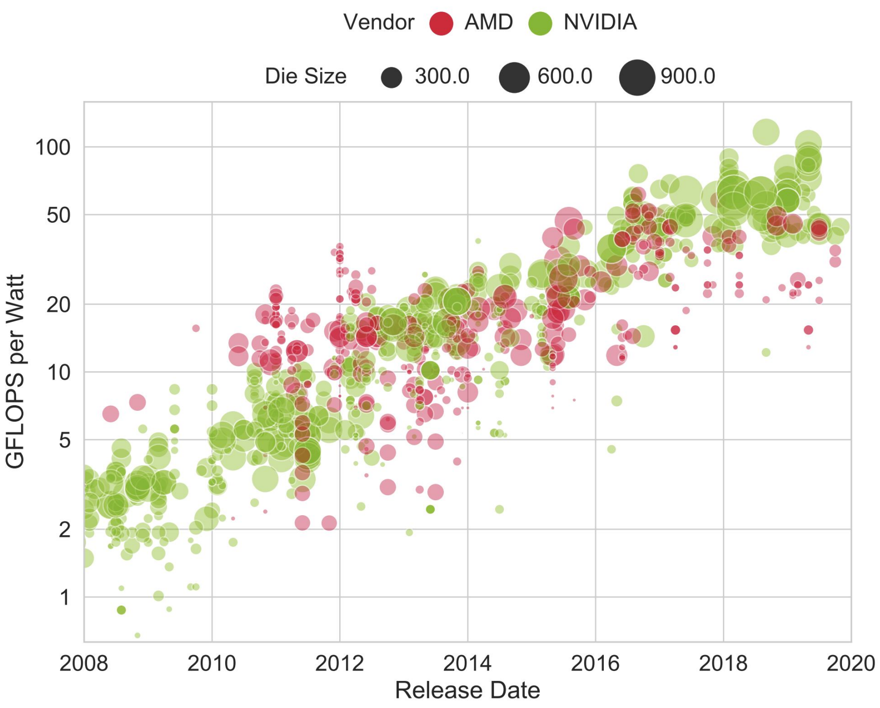


Fig. 5. GPU FLOPS per watt.

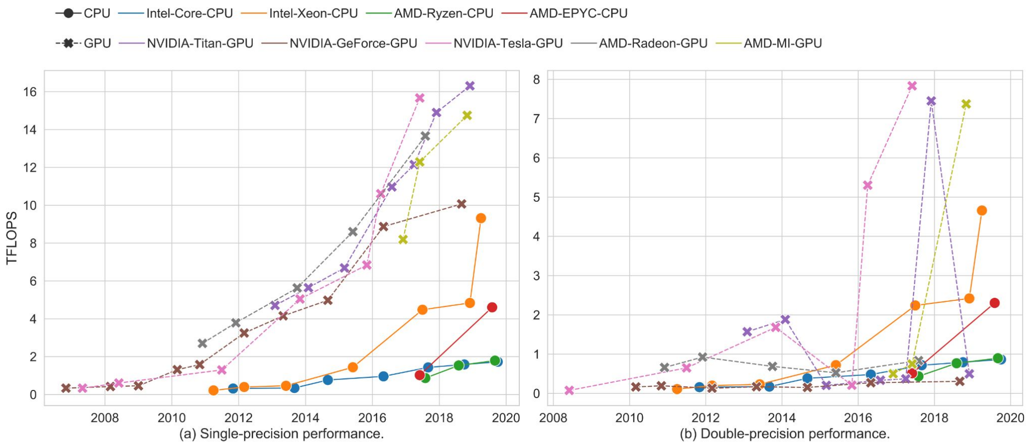


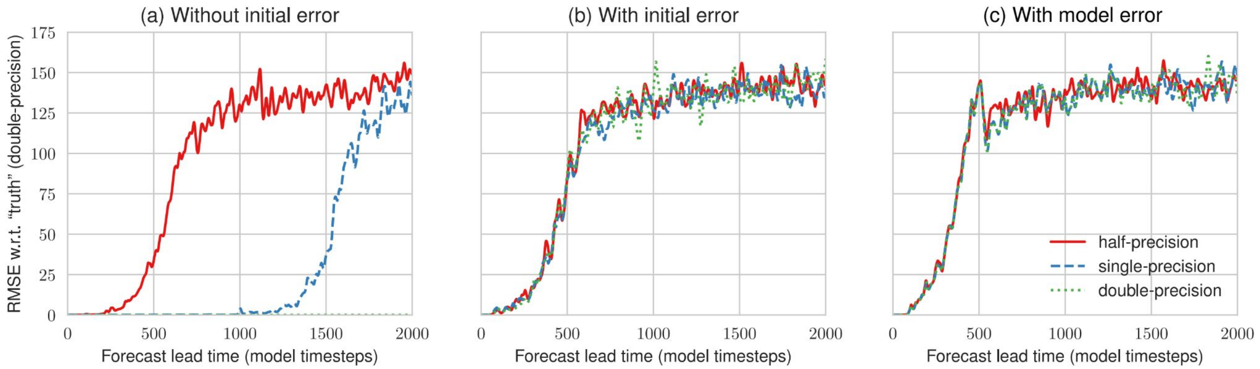
Fig. 6. Comparing single-precision and double-precision performance of CPUs and GPUs.

- GPUs are continuing to see a consistent increase in performance per watt
- Recently, both theoretical single and double precision TFLOP GPU performance consistently exceed CPU
- Significant opportunity cost on GPUs using double precision

"Summarizing CPU and GPU Design Trends" Sun, et al 2019

Side Note: Consider Mixed Precision Arithmetic

- Single Precision floating point can often still maintain significant accuracy in specific ranges
- Models often memory bound, reduced precision increases bandwidth
- Any precision error handled by mean of ensemble or nonlinear chaos
  - Hatfield's ECMWF presentation "Mixed Precision Arithmetic in Earth System Modeling"
  - JAMES article "Weather and Climate Model in 16-bit Arithmetic"



Lorenz '63 example

Present Shift in the HPC Computational Model

Old Constraints

- **Peak clock frequency** as primary limiter for performance improvement
- **Cost**: FLOPs are biggest cost for system: optimize for compute
- **Concurrency**: Modest growth of parallelism by adding nodes
- **Memory Scaling**: maintain byte per flop capacity and bandwidth
- **Locality**: MPI+X model (uniform costs within node & between nodes)
- **Uniformity**: Assume uniform system performance
- **Reliability**: It's the hardware's problem

New Constraints

- **Power** is primary design constraint for current HPC system design
- **Cost**: Data movement dominates: optimize to minimize data movement
- **Concurrency**: Exponential growth of parallelism within chips
- **Memory Scaling**: Compute growing 2x faster than capacity or bandwidth
- **Locality**: must reason about data locality and possibly topology
- **Heterogeneity**: Architectural and performance non-uniformity increase
- **Reliability**: Cannot count on hardware protection alone

"Exascale Computing Trends" Kogge & Shalf, 2004 and LLNL slides

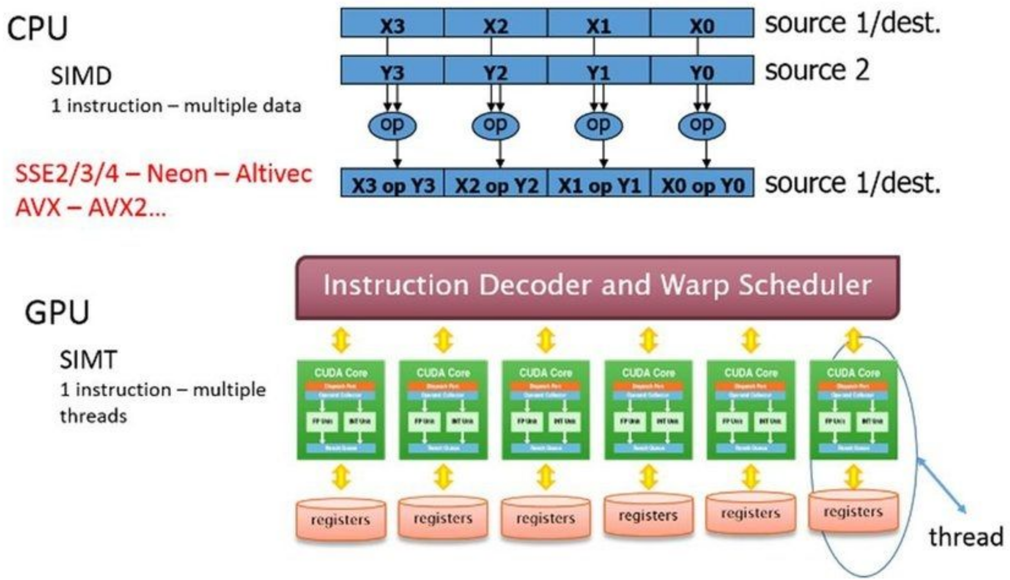
Vector and Thread Processing
CPU vs GPU

Recall - Flynn's Taxonomy

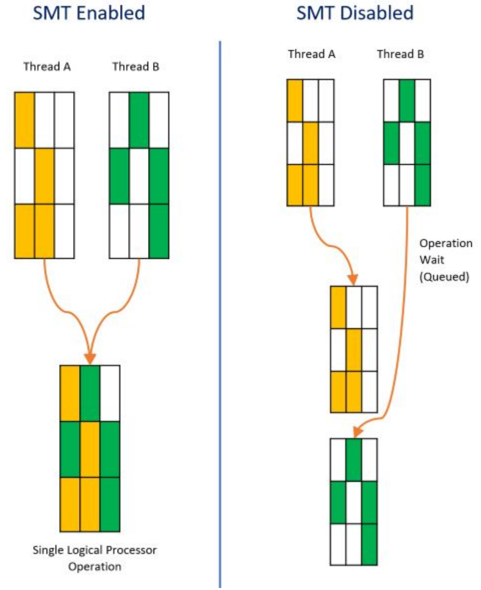
	Single Data	Multiple Data
Single Instruction	• *SISD* • typical CPU thread	• *SIMD* • vector processors • GPU thread blocks
Multiple Instruction	• *MISD* • possibly set of filters • fault tolerance and redundancies	• *MIMD* • cluster of nodes • multi-core CPU

GPUs process essentially SIMD operations at massive parallel scale.
Hardware differences renames this to SIMT, T=Threads.

SIMD vs SMT vs SIMT



https://www.hardwaretimes.com/simd-vs-simt-vs-smt-whats-the-difference-between-parallel-processing-models/



https://thomassmart.github.io/2018/10/24/Hyper-V-CPU-Scheduler.html

SIMD uses vector units while SIMT leverages threads. A hybrid of the two, SMT (Simultaneous Multi-Threading) refers to multiple threads per CPU core

CPU vs GPU

CPU - AMD EPYC Bus
64 cores (seats)
2 threads per core
4 double precision values in 256-bit vector registers

64 x 2 x 4 = **512 SIMD ops**

Busiest Bus Terminal (NYC): ~225 thousand/day

GPU - A100 Shinkansen
108 SMs (cars)
64 warps/SM (seats)
32 SIMT threads per warp
1 register set per thread

108 x 64 x 32 = **221,184 SIMT ops!**

Busiest Train Station (Shinjuku, Tokyo): ~3.5 million/day

CPU vs GPU

CPU - AMD EPYC Bus

Adaptable to different road types
(complex ALUs)
Many different roads to use
(branch prediction)

Can change lanes, traffic manageable?
(Large memory and caches)
Small parallel capacity, complex tasks

GPU - A100 Shinkansen

Must follow laid track
(simpler ALUs)
Fewer options for destinations
(no branch prediction)

Passenger load/de-load very slow
(High cost to move memory to device)
Large parallel capacity, high throughput

Side Note: Branchless Programming

If **IF-ELSEIF** branches in your GPU kernels concern you...

```
def Smaller(a, b):
        if a < b:
            return a
               else
          return b
```

→

```
def Smaller_Branchless(a, b):
        return a * (a < b) + b * (b <= a)
```

Creel, YouTube - "Why 'If' Is Sloowww… and What You Can Do About It?"

Some awareness of threaded processing will help towards optimizing performance of GPU kernels, ie compute units, that are scheduled on the GPU.

We will go into more detail on these concepts in later sessions

GPU Software Paradigms and Community Support

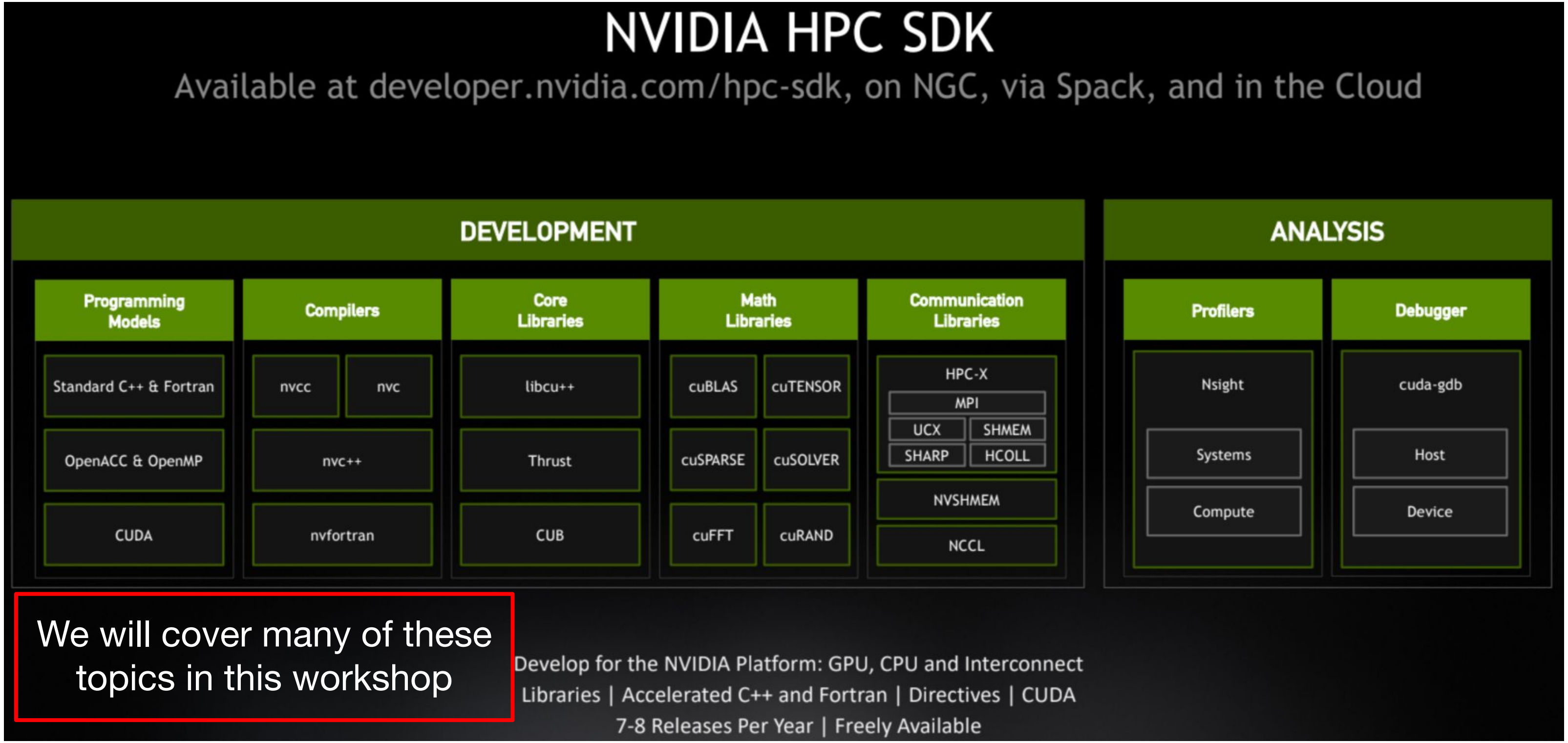
Building Trust in GPU Community Tools

When developing software, there are many lower levels of computer architecture that we don't directly interact with and many times require little awareness of.

Essentially, we have to trust that decisions made by microprocessor architects and compiler engineers are making good decisions for us when their tools build our intended software

Do you trust your colleagues to make optimal decisions?

NVIDIA's Software Ecosystem



We will cover many of these topics in this workshop

https://developer.nvidia.com/hpc-sdk

Recall - Descriptive vs Prescriptive Programming

Prescriptive	Descriptive
OpenMP (flexible) MPI CUDA	OpenACC (flexible) do concurrent std::par
Programmer explicitly parallelizes the code, compiler obeys	Compiler parallelizes the code with guidance from the programmer
Requires little/no analysis by the compiler	Compiler must make decisions from available information
Substantially different architectures require different directives	Compiler uses information from the programmer and heuristics about the architecture to make decisions
Fairly consistent behavior between implementations	Quality of implementation greatly affects results.

Progress in GPU computing has significantly reduced barrier to entry.

Compilers and other tools can do the heavy lifting for you.

Comparison of GPU Programming - Compatibility

If you need the best performance for a GPU kernel, CUDA can be used alongside pragmas

From / To	CUDA	OpenCL	SYLC	OpenACC OpenMP	HIP
C/C++	Add Code	Add Code	Add Code	Pragmas	Add Code
Fortran	Add Code	Rewrite	Rewrite	Pragmas	Rewrite
CUDA		Keep Structure	Keep Structure	Rewrite	Convert
OpenCL	Keep Structure		Keep Structure	Rewrite	Keep Structure
SYLC	Keep Structure	Keep Structure		Rewrite	Keep Structure
OpenACC OpenMP	Rewrite	Rewrite	Rewrite		Rewrite
HIP	Convert	Keep Structure	Keep Structure	Rewrite	

CUDA / HIP (AMD) - Often requires new codes and rewrites

OpenACC/OpenMP - Easy to implement, achieves good enough performance

PRACE, "Best Parctice Guide - Modern Accelerators"

Best Practices with GPU Software Development

Libraries

Drop-In, heavily tested, optimized performance

1

Directives

Easy acceleration, good enough performance

2

CUDA, Programming Languages

Max flexibility, control, & performance; Most effort

3

Only a suggested order of implementation as part of a GPU project, depends highly on model needs.

Best Practices with GPU Software Development

Libraries

Directives

CUDA, Programming Languages

Drop-In, heavily tested, optimized performance

Easy acceleration, good enough performance

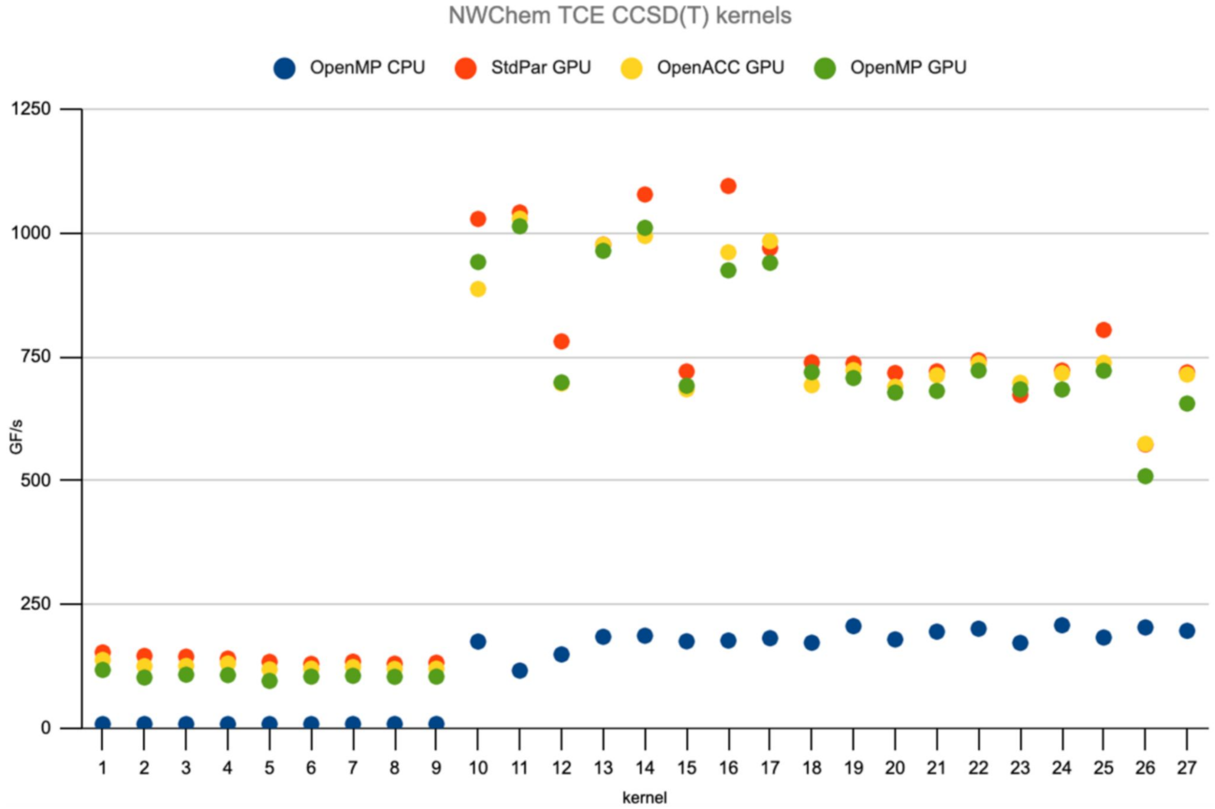
Max flexibility, control, & performance; Most effort

When using any of these approaches, **ALWAYS MEASURE PROGRESS**

Profile → **Determine Limiter** → **Apply Optimizations** → **Iterate**

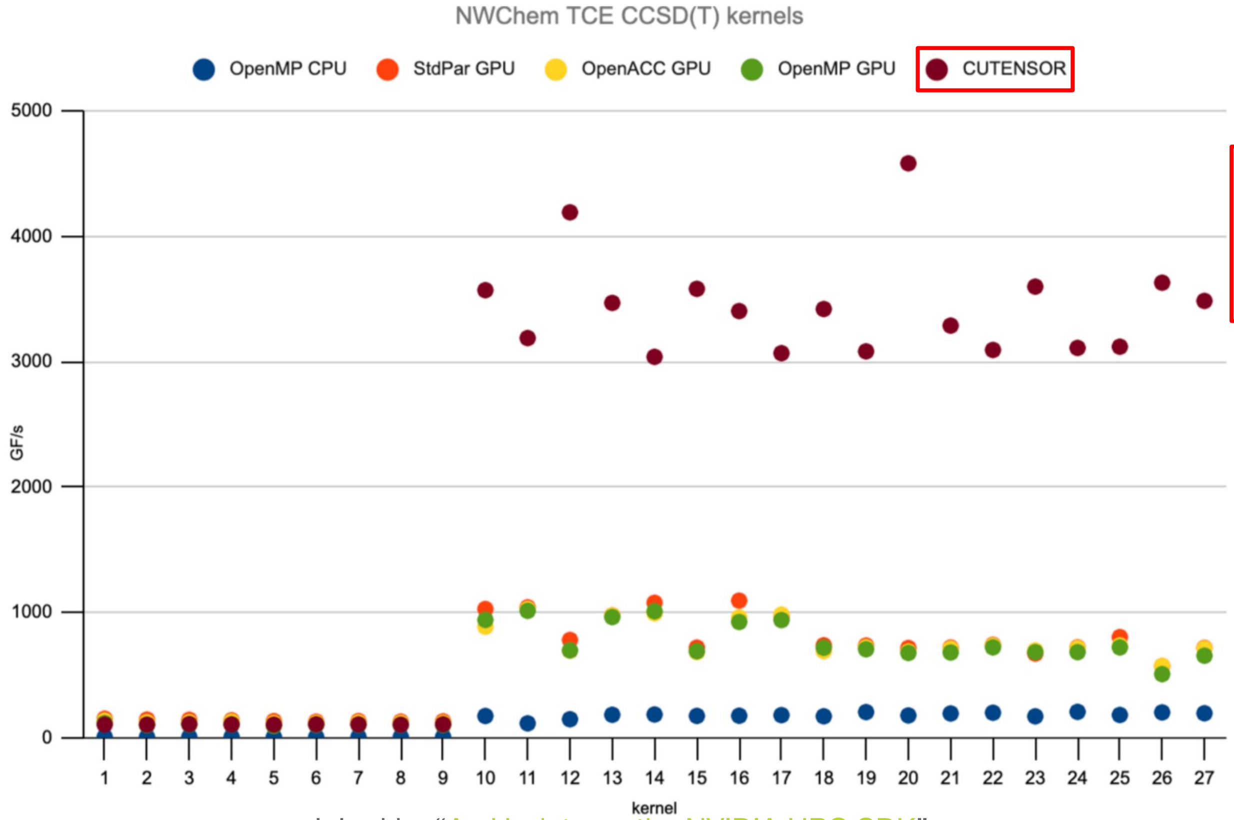
using NSight Systems and Nsight Compute

Comparing Performance of Programming Paradigms



J. Larkin, "An Update on the NVIDIA HPC SDK"

Comparing Performance of Programming Paradigms



Accelerated libraries offer best performance

J. Larkin, "[An Update on the NVIDIA HPC SDK]"

Future Directions of GPU Programming

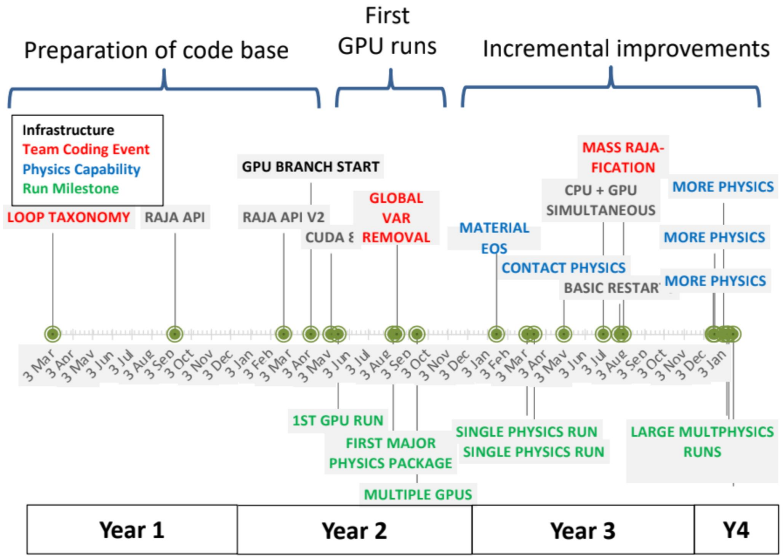
Tools are constantly being expanded upon towards...

- Increasing performance of descriptive programming approaches
  - OpenACC/OpenMP robustly supported, may merge?
  - ISO Standard parallelism (`do concurrent` & `std::par`)
- Promoting portability in software design
- Developing robust libraries that are widely available
  - Very easy to drop in ML/AI apps for GPUs, Legate Numpy/cuNumeric
  - GPU equivalents to MKL and many important math algorithms

How to Approach GPU Projects or GPU Refactoring

GPU Modernization Project Refactorization Template

1. Refactor and remove GPU anti-patterns
    a. Global variables, memory movements
    b. Incompatible data constructs, ie STL
2. Create a mini-app to explore design space
3. Use portable abstractions and frameworks
4. Focus on a specific use case
5. Search for additional parallelism
6. Manually manage memory
7. Iteratively apply the steps above

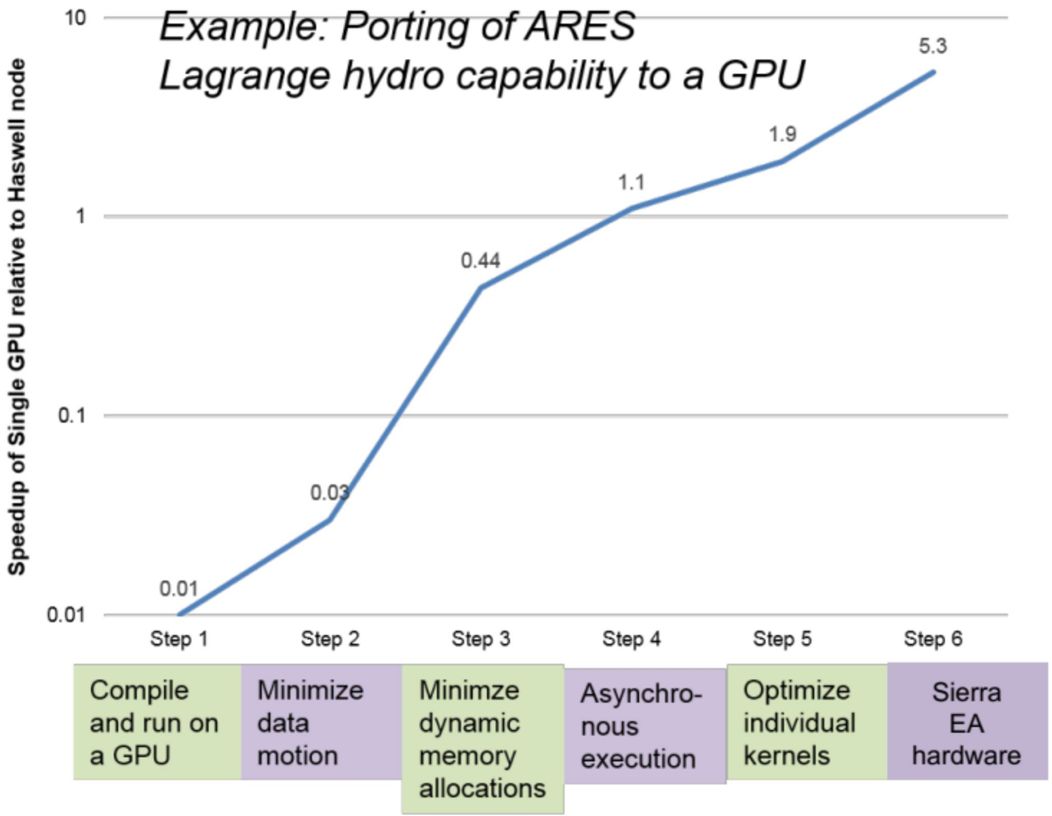


D. Richards, LLNL exascaleproject.org/event/sierra_and_elcapitan_coe/

Large projects & new software that intends to have a long life cycle require effective planning.

This LLNL project example took ~4 years, but this was early in the team's learning of heterogeneous GPU architectures. Newer tools & portable abstraction frameworks, like OpenACC, will speed this up!

Performance Impacts Along Project Development



Initial work will likely see reduced performance of your model.

However, once data movement and other optimizations are performed, the benefits can be significant.

D. Richards, LLNL exascaleproject.org/event/sierra_and_elcapitan_coe/

Effective planning and coordination amongst your team and with external communities of practice will provide you the most benefit.

We will cover aspects of this process as part of a later topic, Co-Design

Thank you!

Any Questions?

Additional Resources

AI Learning Materials/Courses for Earth Sciences - AI2ES

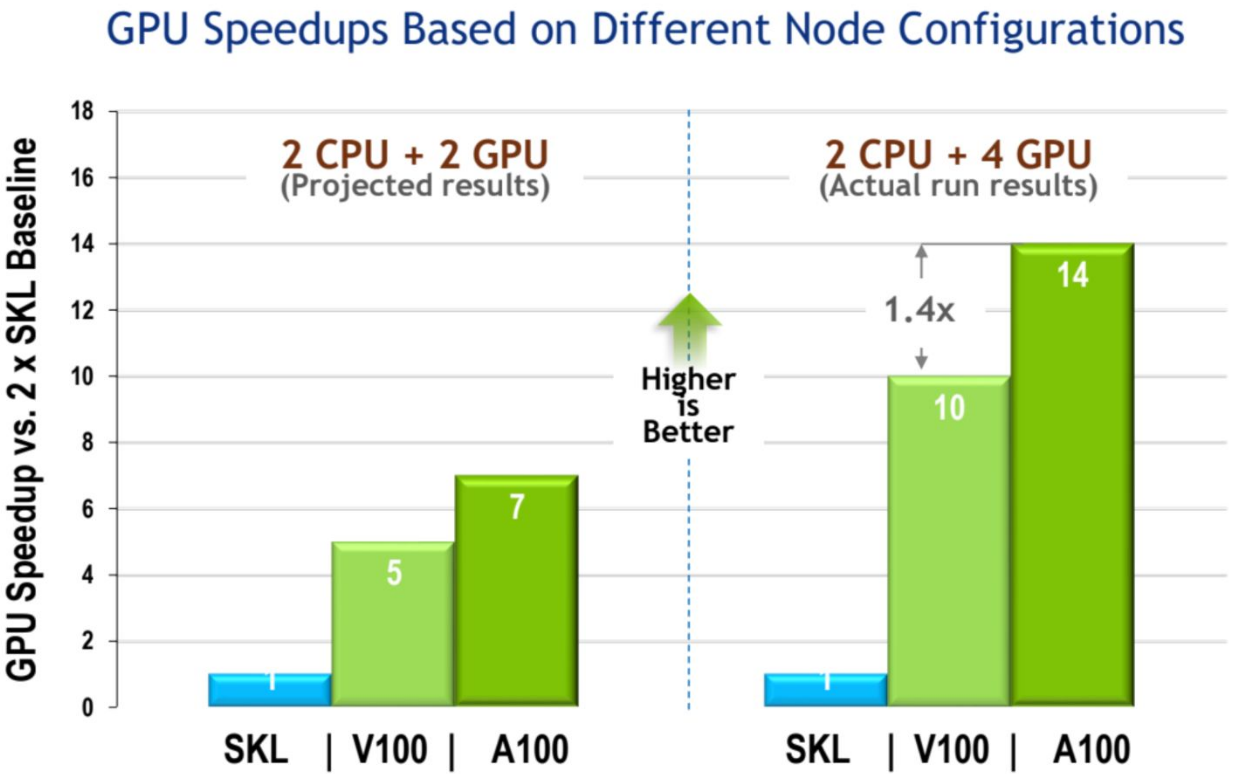
PRACE: "Best Practice Guide for Modern Accelerators"

Nature: "The Digital Revolution of Earth-System Science"

ACM: "A vision for GPU-accelerated parallel computation on geo-spatial datasets"

Extra Slides

Example Model Speedups



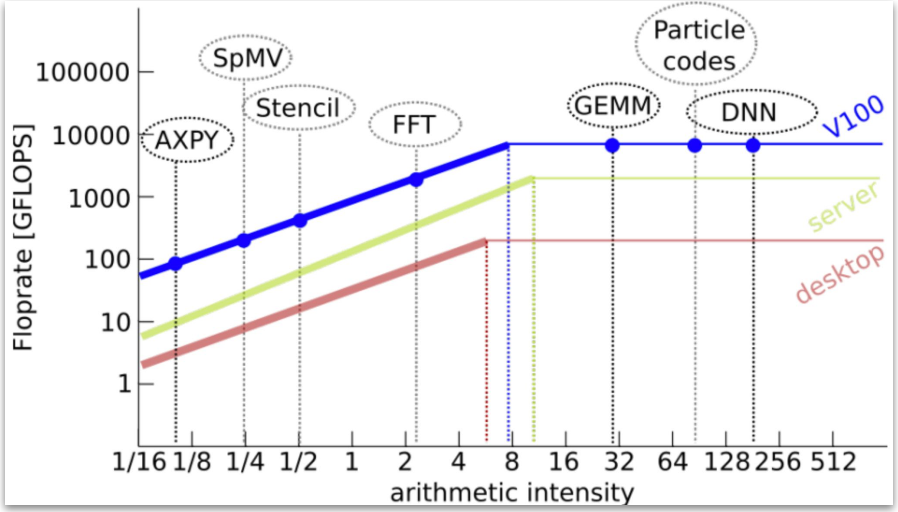
WRF: A100 vs. SKL = 7x
Socket-level Speedup

Source: NVIDIA – Adie, Jul 2020

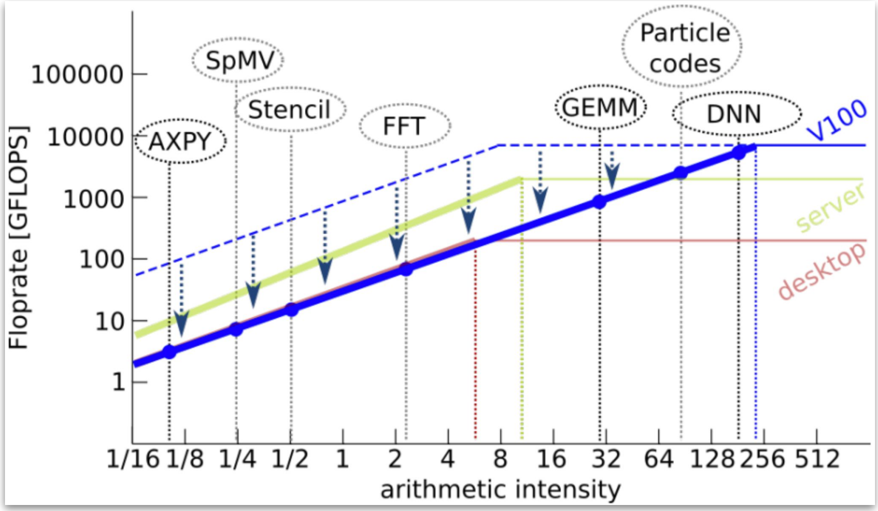
- Based on NCAR WRF 3.8.1
- Single domain, 113M points
- Single node CPU-only
  - 1 MPI task each core
- Single node CPU+GPU
  - 1 MPI task per GPU

J. Adie & S. Posey via WRFg from Tempoquest and NCAR

Roofline V100 Performance of “Dwarf” Algorithms



Normal roofline model for V100

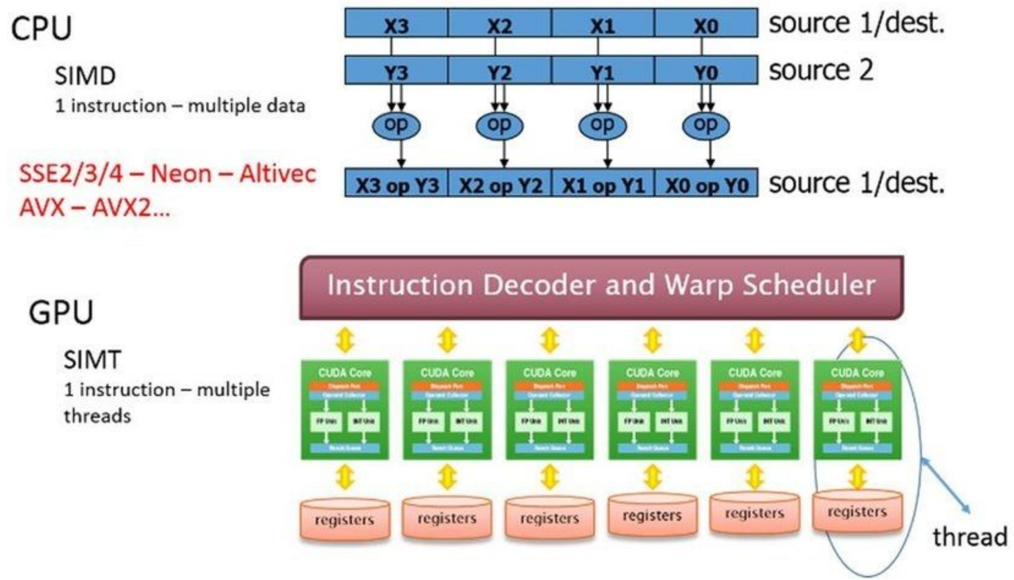


Same, constrained to PCIe Bandwidth

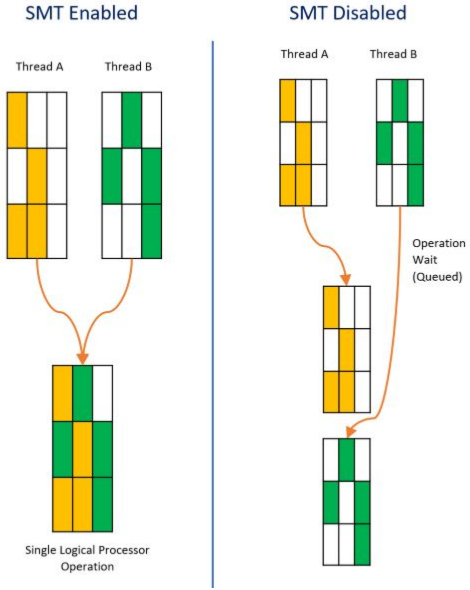
Performance of GPUs capable of exceeding CPUs, however if data movement is not managed appropriately (uses PCIe too frequently), CPUs are better.

PRACE, “Best Practice Guide - Modern Accelerators”

SIMD vs SMT vs SIMT



https://www.hardwaretimes.com/simd-vs-simt-vs-smt-whats-the-difference-between-parallel-processing-models/



https://thomassmart.github.io/2018/10/24/Hyper-V-CPU-Scheduler.html

Flexibility: SMT > SIMT > SIMD

Performance: SIMD > SIMT > SMT

Threaded processes allow

Multiple register sets, addresses, flowpaths