

Parallel Programming Methodologies

Maciej Cytowski

International HPC Summer School

8th July 2024, Kobe, Japan

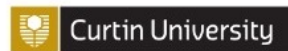
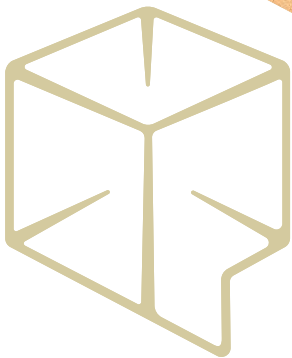


Pawsey Supercomputing Research Centre

An Australian National Tier-1 Facility



- Launched as Pawsey in 2014, UJV foundations since 2000
- Recent AU\$70m capital refresh by Australian Government
- 60+ Staff employed via CSIRO, Australia's national science agency
- Home to:
 - 227+ research projects
 - 4,000+ researchers
 - 10,000+ training attendees
 - 1,000,000,000+ hours of research computing





Agenda

- **The Need for Speed Scale**
- Current Trends in Supercomputing Technology
- Parallel Programming Models
- Challenges
- How to choose your track?



Square Kilometre Array



Inyarrimanha Ilgari Bundara, the CSIRO Murchison Radio-astronomy Observatory

SKA1 LOW - the SKA's low-frequency instrument

The Square Kilometre Array (SKA) will be the world's largest radio telescope, revolutionising our understanding of the Universe. The SKA will be built in two phases - SKA1 and SKA2 - starting in 2018, with SKA1 representing a fraction of the full SKA. SKA1 will include two instruments - SKA1 MID and SKA1 LOW - observing the Universe at different frequencies.

Location: Australia

Frequency range:
50 MHz to 350 MHz

~130,000
antennas spread between
500 stations

Total collecting area:
0.4km²

Maximum distance between stations:
65km

Total raw data output:
157 terabytes per second
4.9 zettabytes per year

Enough to fill up
35,000 DVDs every second

5x the estimated global internet traffic in 2015
(source: Cisco)

Compared to LOFAR Netherlands, the current best similar instrument in the world

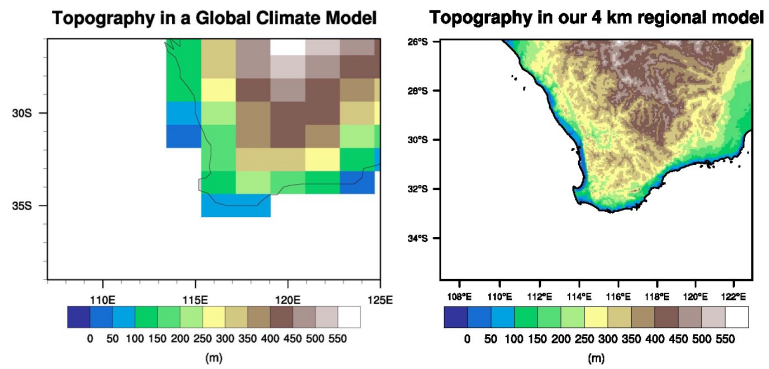
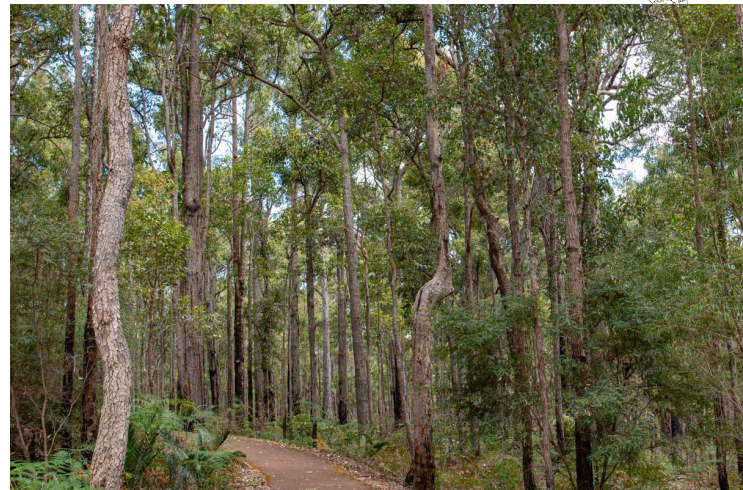
25% better resolution
8x more sensitive
135x the survey speed

www.skatelescope.org | Square Kilometre Array | @SKA_tlescope | The Square Kilometre Array

Weather and Climate: Save The Giants



- Weather and climate models are now delivering global predictions at 10-kilometre resolution and regional forecasts at the kilometre-scale
- Australia's Southwest has been in the grip of human-triggered climate change since around 1970
- Along with other Mediterranean-type climate regions, this area is rapidly drying due to global warming, with rainfall declines of 15-20%
- The Climate Science Initiative will produce the most detailed and comprehensive Western Australian climate change projections to date, extending 75 years into the future
- Nearly 150M core hours and 7PB of data

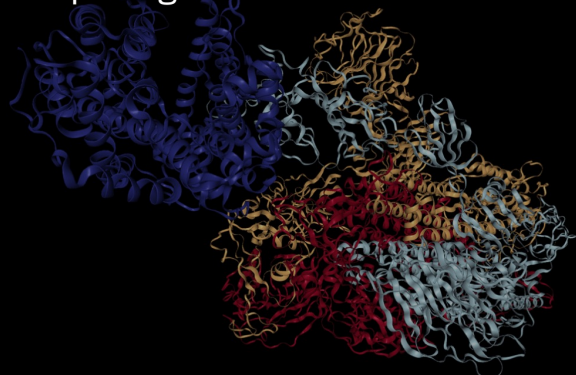


Sources:
<https://theconversation.com/ecocheck-australias-southwest-jarrah-forests-have-lost-their-iconic-giants-49150>
<https://exploreparks.dbca.wa.gov.au/site/king-jarrah-wellington-national-park>



COVID-19 Research

The COVID-19 High Performance Computing Consortium



Bringing together the Federal government, industry, and academic leaders to provide access to the world's most powerful high-performance computing resources in support of COVID-19 research.

89

Projects

600

Petaflops

- Molecular Dynamics
- Quantum Chemistry
- **Bioinformatics Workflows**
- **AI/ML tools**



Supercomputer Research Leads to Human Trial of Potential COVID-19 Therapeutic Raloxifene

By Oliver Peckham

October 29, 2020

Six months of patient trials with the ostensible COVID-19 therapeutic remdesivir have cast serious doubt on its ability to even reduce the severity of COVID-19, let alone reliably reduce patient mortality. This has left medical professionals once again without a proven COVID-19 therapeutic as the pandemic shows signs of reaching unprecedented spikes in Europe and North America. As the world confronts the possibility of a dark winter, a supercomputer-powered pharmaceutical research coalition based in Italy is producing a glimmer of light: a repurposed drug called raloxifene, identified by supercomputing research, that will now be entering clinical trials to test its efficacy as a therapeutic for COVID-19.

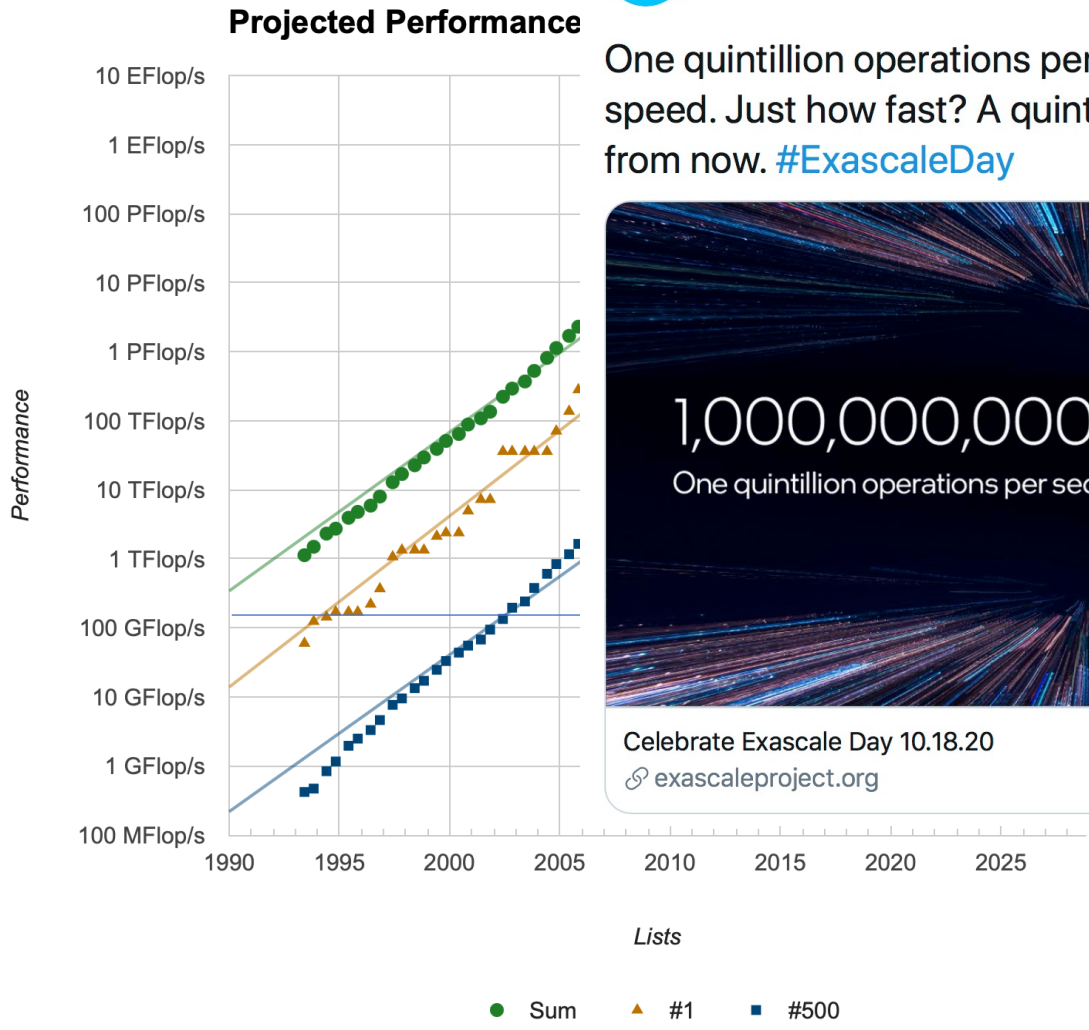


Agenda

- The Need for Speed Scale
- **Current Trends in Supercomputing Technology**
- Parallel Programming Models
- Challenges
- How to choose your track?

Top500 List (top500.org)

The **TOP500** project ranks and details the 500 most powerful supercomputer in the world based on High Performance Linpack benchmark



One quintillion operations per second is mind blowing 🤯 speed. Just how fast? A quintillion seconds is 31B years from now. [#ExascaleDay](#)

intel.

1,000,000,000,000,000,000

One quintillion operations per second. That's how fast **exascale** is.

Celebrate Exascale Day 10.18.20

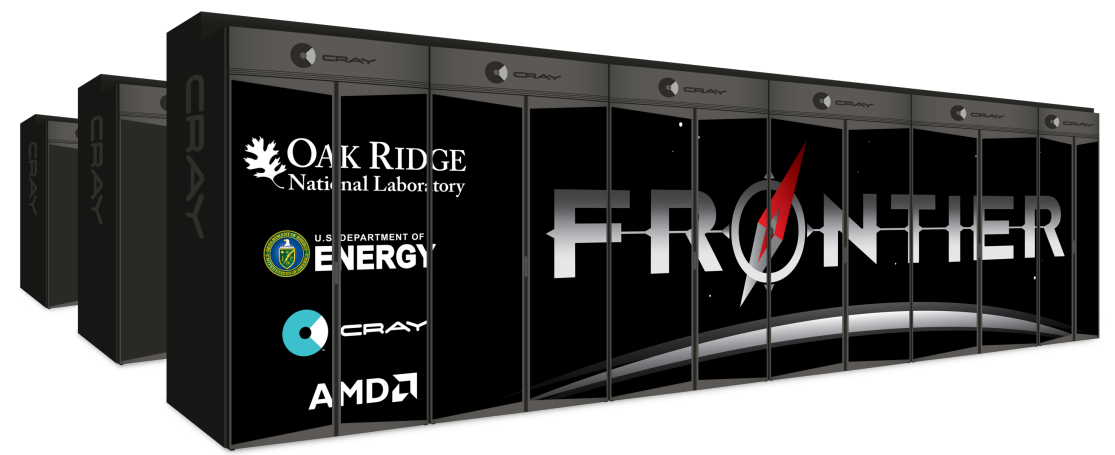
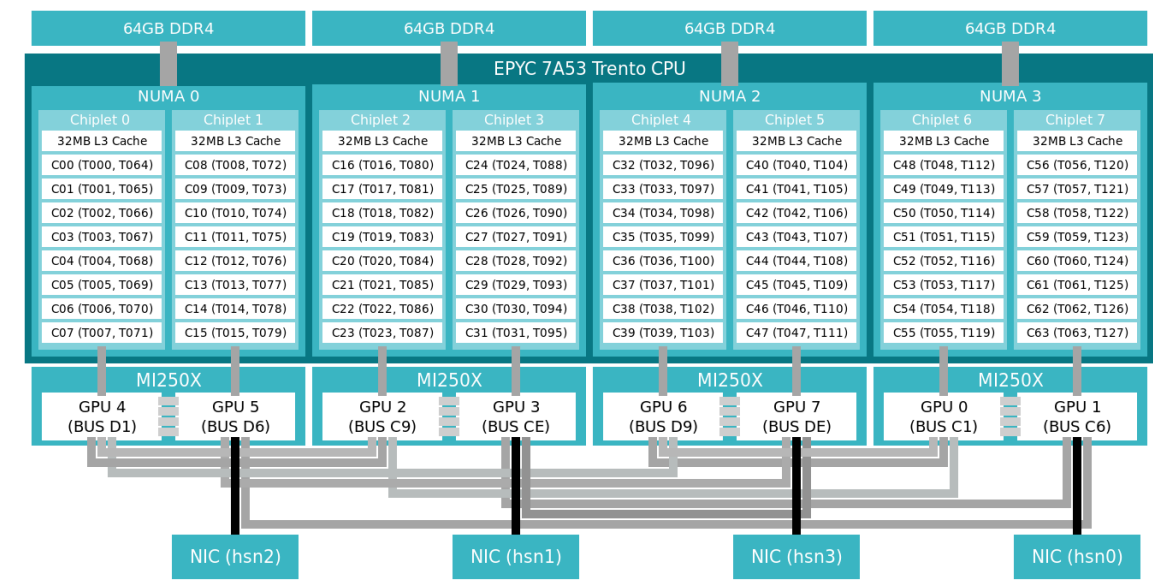
[exascaleproject.org](#)

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	Aurora - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel oratory	9,264,128	1,012.00	1,980.01	38,698
3	Platinum 8480C 48C 2GHz, NVIDIA Microsoft Azure	2,073,600	561.20	846.84	
4	Supercomputer Fugaku, A64FX 48C Fujitsu Fugaku Science	7,630,848	442.01	537.21	29,899
5	Optimized 3rd Generation EPYC X, Slingshot-11, HPE	2,752,704	379.70	531.51	7,107
6	IA Grace 72C 3.1GHz, NVIDIA I, HPE g Centre (CSCS)	1,305,600	270.00	353.75	5,194
7	00, Xeon Platinum 8358 32C GB, Quad-rail NVIDIA HDR100	1,824,768	241.20	306.31	7,494
8	Juana XH3000, Xeon Platinum 00 64GB, Infiniband NDR, EVIDEN	663,040	175.30	249.44	4,159
9	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	148.60	200.79	10,096
10	Eos NVIDIA DGX SuperPOD - NVIDIA DGX H100, Xeon Platinum 8480C 56C 3.8GHz, NVIDIA H100, Infiniband NDR400, Nvidia NVIDIA Corporation United States	485,888	121.40	188.65	



Frontier, Top1 HPL

- The first Exascale system Frontier installed in ORNL, US
- Based on HPE Cray EX architecture
- Each node with 64 AMD Trento compute cores and 4 AMD MI250X GPUs
- Slingshot interconnect
- 9,472 nodes
- 74 racks
- 1,206 Pflop/s HPL (70% peak)





HPCG

- The TOP500 list has incorporated the High-Performance Conjugate Gradient (HPCG) benchmark results starting from 2017
- Provides an alternative metric for assessing supercomputer performance
- Complements the HPL measurement to give a fuller understanding of the machine

Rank	Rank	System	Cores	Rmax (PFlop/s)	HPCG (TFlop/s)
1	4	Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	16004.50
2	1	Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	14054.00
3	2	Aurora - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	5612.60
4	5	LUMI - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	4586.95
5	6	Alps - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	1,305,600	270.00	3671.32
6	7	Leonardo - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA Italy	1,824,768	241.20	3113.94
7	9	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	148.60	2925.75
8	14	Perlmutter - HPE Cray EX 235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-11, HPE DOE/SC/LBNL/NERSC United States	888,832	79.23	1905.00
9	12	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	94.64	1795.67
10	15	Selene - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63.46	1622.51



Fugaku, Top1 HPCG

- The so-called “Post K” computer installed in RIKEN, Japan
- Based on Fujitsu’s ARM A64FX architecture
- Each node with 48 compute cores, 32GB High Bandwidth Memory, on-die Tofu-D interconnect and AI support (FP16, INT8, etc.)
- 158,976 nodes
- 432+ racks
- 442 Pflop/s HPL Linpack (82% peak)



Source: “REPORT ON THE FUJITSU FUGAKU SYSTEM”, J.Dongarra, Tech Report No. ICL-UT-20-06
<https://www.top500.org/news/report-fujitsu-fugaku-system-jack-dongarra/>

Green500

- The **Green500** is a biannual ranking of supercomputers, from the Top500 list of supercomputers, in terms of energy efficiency.
- The list measures performance per watt using the Top500 measure of HPL benchmarks at double-precision floating-point format.



Source: https://en.wikipedia.org/wiki/PlayStation_3



Source: IBM, ICM, University of Warsaw, Nautilus supercomputer, 2008

Rank	TOP500		Cores	Rmax (PFlop/s)	Power (kW)	Energy Efficiency (GFlops/watts)
	Rank	System				
1	189	JEDI - BullSequana XH3000, Grace Hopper Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, ParTec/EVIDEN EuroHPC/FZJ Germany	19,584	4.50	67	72.733
2	128	Isambard-AI phase 1 - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE University of Bristol United Kingdom	34,272	7.42	117	68.835
3	55	Helios GPU - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Cyfronet Poland	89,760	19.14	317	66.948
4	328	Henri - ThinkSystem SR670 V2, Intel Xeon Platinum 8362 32C 2.8GHz, NVIDIA H100 80GB PCIe, Infiniband HDR, Lenovo Flatiron Institute United States	8,288	2.88	44	65.396
5	71	preAlps - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	81,600	15.47	240	64.381
6	299	HoreKa-Teal - ThinkSystem SD665-N V3, AMD EPYC 9354 32C 3.25GHz, Nvidia H100 94Gb SXM5, Infiniband NDR200, Lenovo Karlsruher Institut für Technologie (KIT) Germany	13,616	3.12	50	62.964
7	54	Frontier TDS - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	120,832	19.20	309	62.684
8	11	Venado - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE DOE/NNSA/LANL United States	481,440	98.51	1,662	59.287
9	20	Adastra - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Grand Equipement National de Calcul Intensif - Centre Informatique National de l'Enseignement Suprieur (GENCI-CINES) France	319,072	46.10	921	58.021
10	28	Setonix – GPU - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Pawsey Supercomputing Centre, Kensington, Western Australia Australia	181,248	27.16	477	56.983



JEDI, Top1 Green500

- The first module of the exascale supercomputer JUPITER, installed at Juelich Supercomputing Centre
- Build by ParTec – Eviden consortium, BullSequana XH3000 platform
- Cluster module of Jupiter will utilize SiPearl Rhea1 processor (ARM)
- 24 compute nodes
- 1 rack
- 4.5 Pflop/s HPL Linpack (87% peak)
- 72.733 Gflops/Watt



Source: <https://www.fz-juelich.de/en/news/archive/press-release/2024/european-exascale-supercomputer-jupiter-sets-new-energy-efficiency-standards-with-1-ranking-on-green500>



What else to measure?

- **HPL MxP** (<https://hpl-mxp.org>) – using mixed precision version of HPL benchmark to highlight and best utilize the emerging convergence of HPC and AI workloads
- **IO500** (<https://io500.org>) – using a selection of IO workloads to highlight the importance of high-performance IO subsystems
- **MLPerf** (<https://mlcommons.org/benchmarks/training-hpc/>) - benchmark suite measuring how fast HPC systems can train models to a target quality metric
-

IHPCSS24 Talk: Parallel I/O, *John Cazes, TACC, Tuesday 2pm*



Trends (Top500 list)

- Two Exaflop systems (Frontier and Aurora)
- All systems on Top500 list are Linux based
- 9 systems in Top10 use GPU accelerators
- A total of **193** systems on the list are using accelerator/co-processor technology
- The entry level to the list moved up to the **2.13** Pflop/s mark on the Linpack benchmark
- The last system on the newest list was listed at position **458** in the previous TOP500



Technology Trends – Computing

Interesting and diverse market

- **Intel** with future-generation CPUs and GPUs and their oneAPI programming environment
- **AMD** with future-generation CPUs, GPUs, APUs their open-source compilers, tools and libraries available in ROCm
- **Fujitsu** with **ARM** A64FX architecture used in Fugaku system and future **SiPearl Rhea1** European-based chips
- **NVIDIA** with their next-generation GPUs and Superchips

Technology Trends – Data

It is all about Data

Data closer to compute

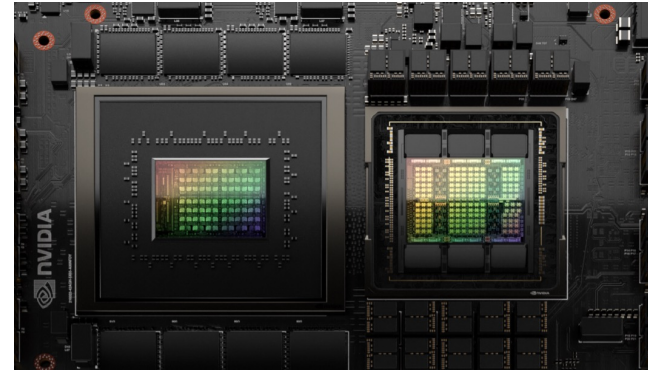
- High Bandwidth Memory close to CPUs / GPUs
- NVMe devices

Better and Unified Access to Data

- PCIe 4.0, NVLINK and InfinityFabric
- Superchips and APUs

More Bandwidth and Less Latency

- FLOP/Byte – measure of compute intensity per memory block
- Used in Roofline Model to optimize single node algorithm design



Source: <https://developer.nvidia.com/blog/nvidia-grace-hopper-superchip-architecture-in-depth/>

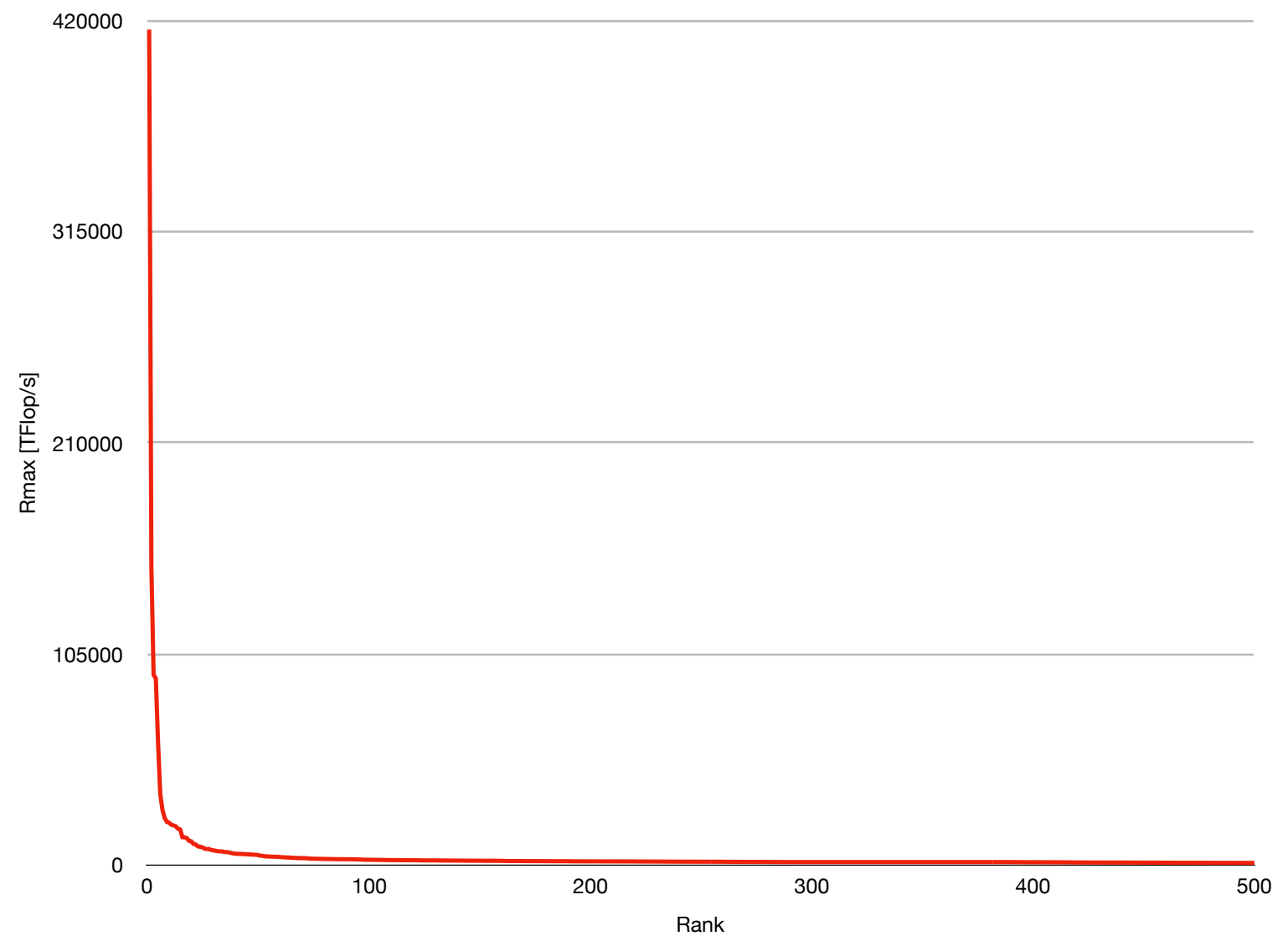
	G80	GP100	GV100
GFLOPS (SP)	384	10,600	15,000
GPU↔GPU memory	84 GB/s	720 GB/s	900 GB/s
FLOP/Byte	4.5	14.7	16.67
GPU↔GPU (NVLINK)	n/a	20 GB/s	20 GB/s
FLOP/Byte		530	750
CPU↔GPU (PCI Express)	3.1 GB/s	3.1 GB/s	3.1 GB/s
FLOP/Byte	124	3,419	4,839

“Don’t Move The Data!”, The CUDA Handbook, cudahandbook.com

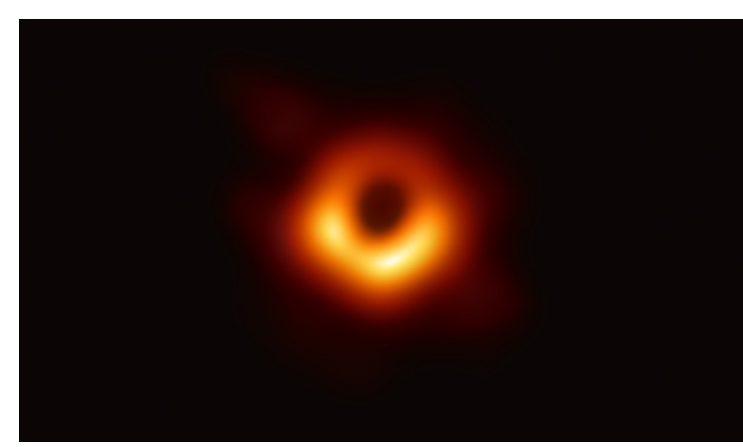


Three Worlds of Supercomputing

Thomas Sterling, HPC Achievements and Impact 2019, ISC'19 Keynote



Source: <https://www.hpcwire.com/2019/06/20/isc-keynote-thomas-sterlings-take-on-whither-hpc/>



Nature **568**, 284-285 (2019)
doi: <https://doi.org/10.1038/d41586-019-01155-0>



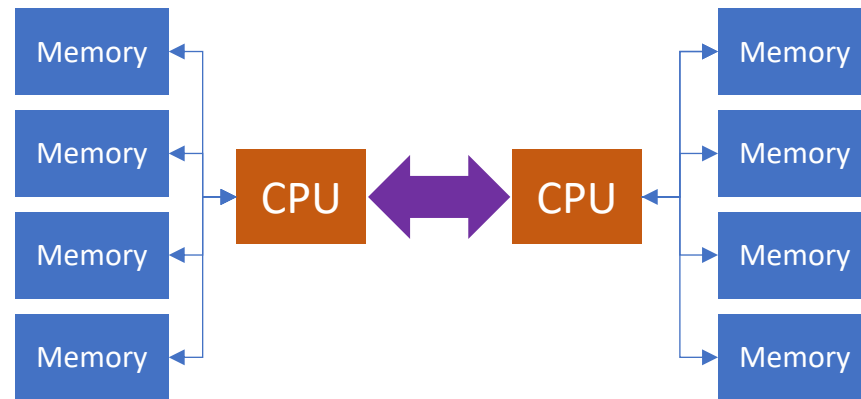
Agenda

- The Need for Speed Scale
- Current Trends in Supercomputing Technology
- **Parallel Programming Models**
- Challenges
- How to choose your track?

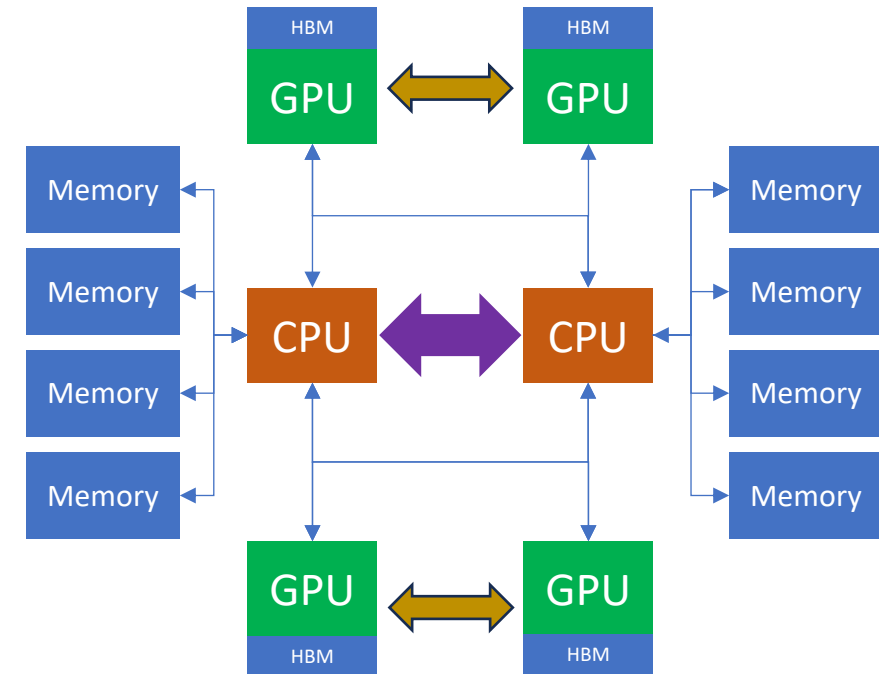


Types of Memory Architectures – Single Node

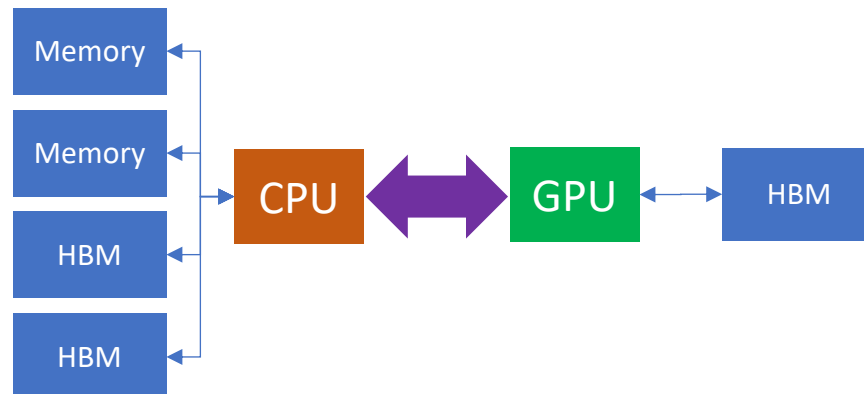
Non-Uniform Memory Access (NUMA) – CPU only



Non-Uniform Memory Access (NUMA) – with accelerator

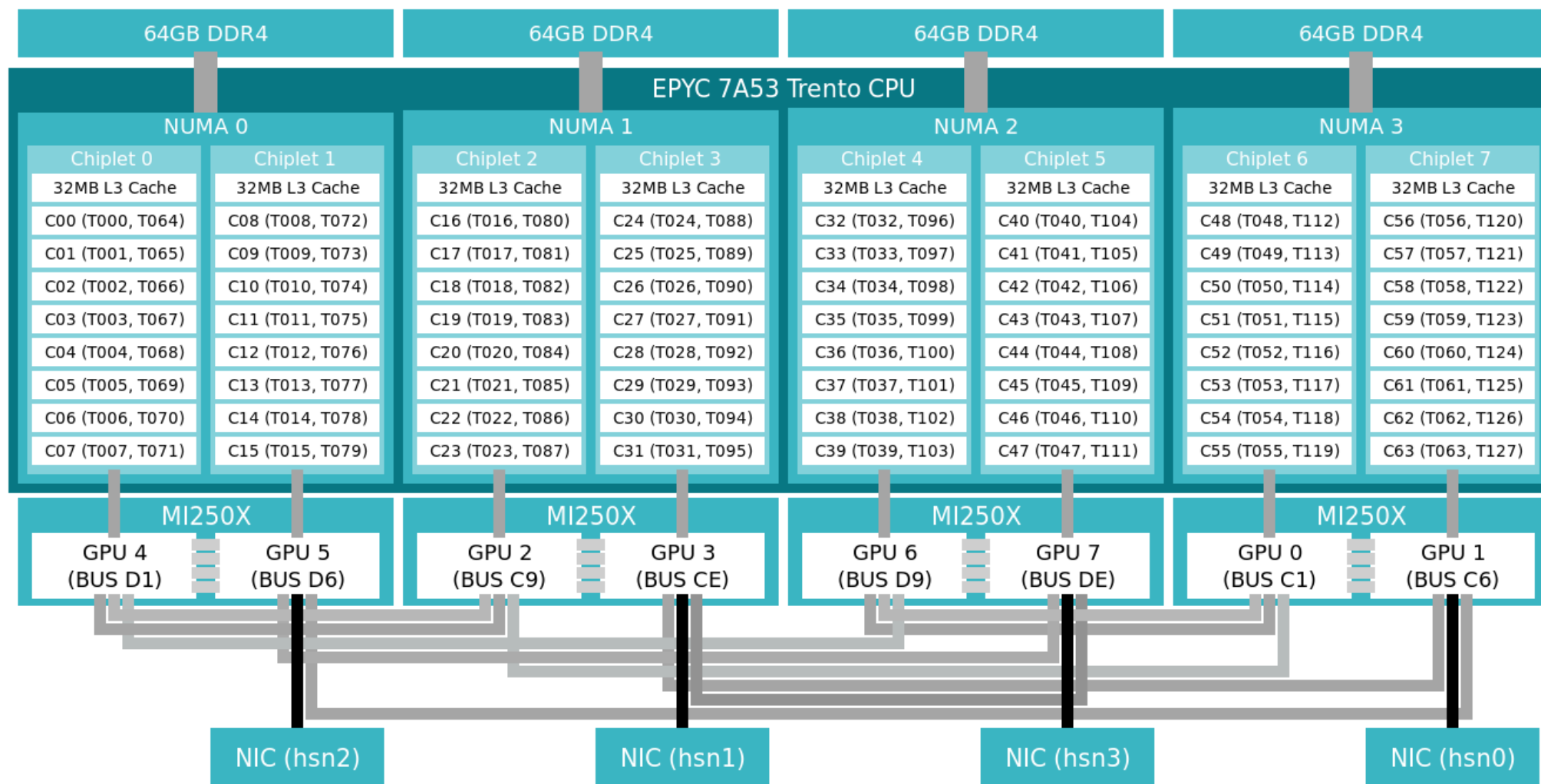


Superchip/APU

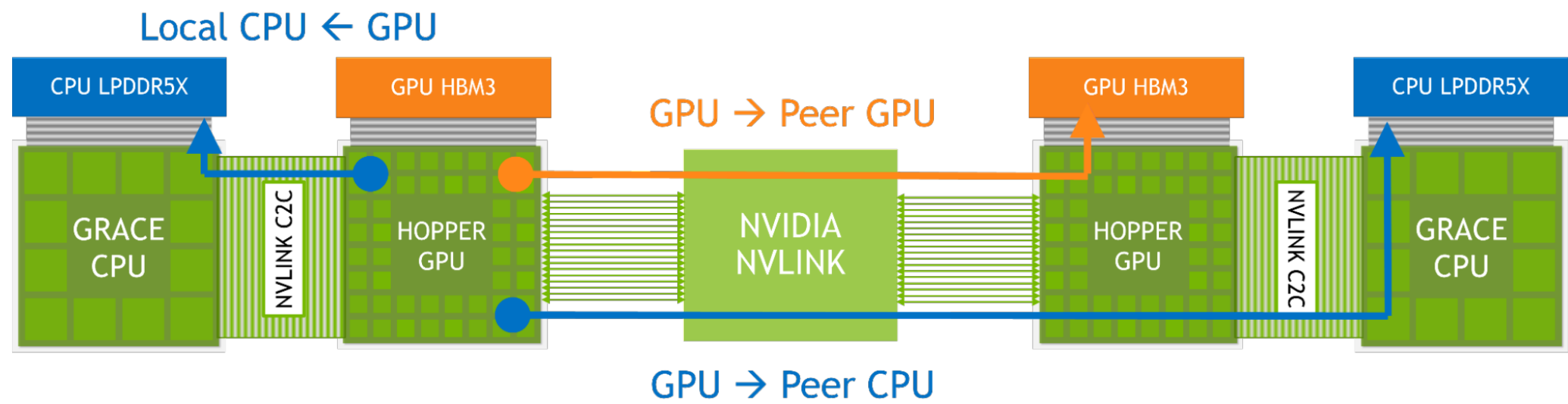




Types of Memory Architectures – Single Node (Ex: Frontier)



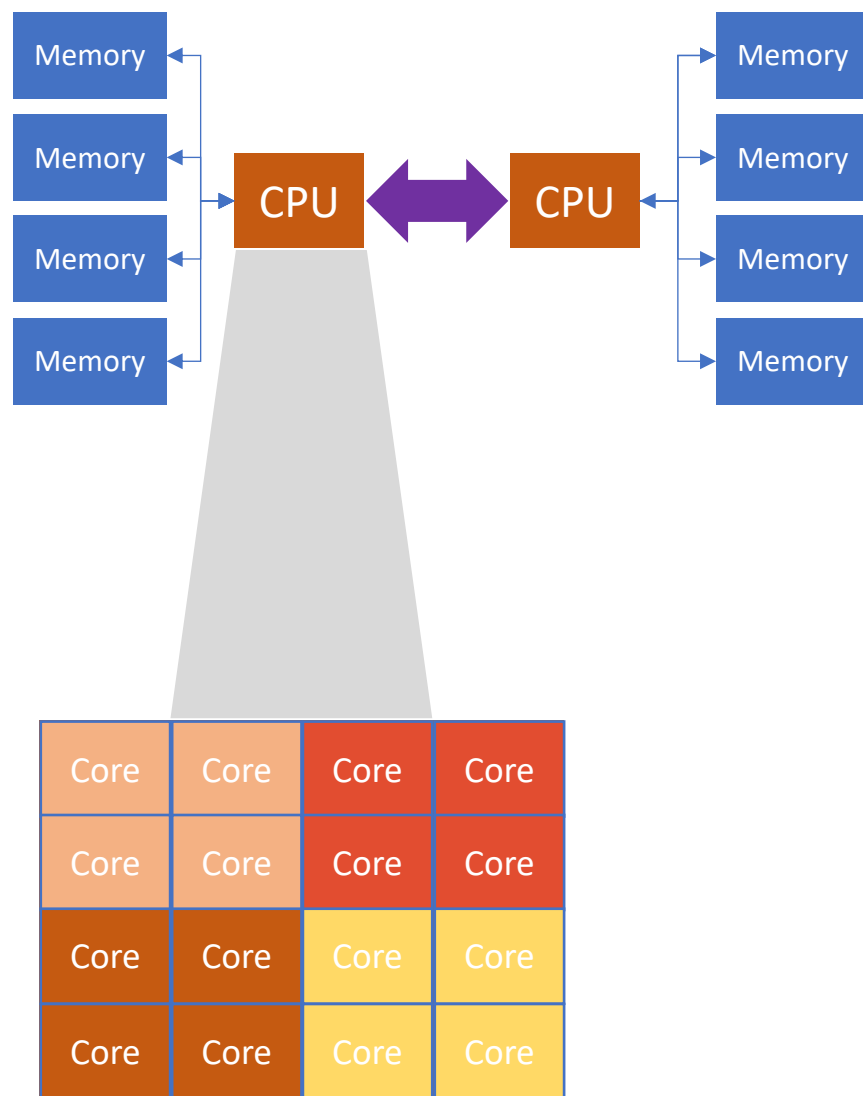
Types of Memory Architectures – Single Node (Ex: NVIDIA Superchip)



Source: <https://developer.nvidia.com/blog/nvidia-grace-hopper-superchip-architecture-in-depth/>



Non-Uniform Memory Access (NUMA) – CPU only Programming



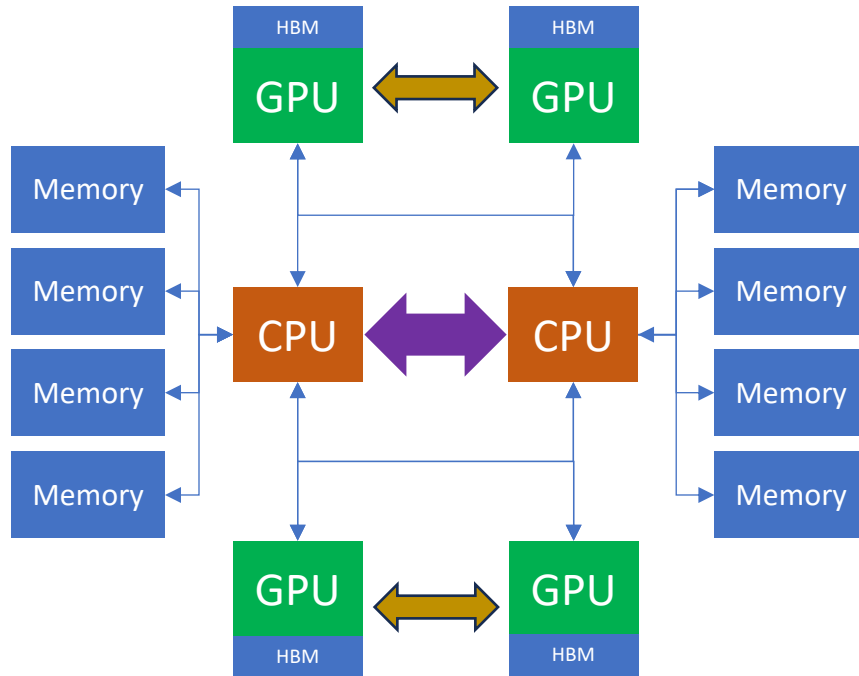
OpenMP

- Directive-based programming model
- Current CPUs are composed of many cores
- Parallel regions of the code need to be annotated with directives
- Parallel data types need to be annotated (e.g. shared, private)

```
void daxpy(int sz) {  
    double a;  
    double* x=(double*)malloc(sz);  
    double* y=(double*)malloc(sz);  
    //initialize  
    ...  
  
    #pragma omp parallel for default(none) shared(x,y) firstprivate(a)  
    for (int i = 0; i < sz; i++) {  
        y[i] = a * x[i] + y[i];  
    }  
}
```



Non-Uniform Memory Access (NUMA) – with accelerator Programming



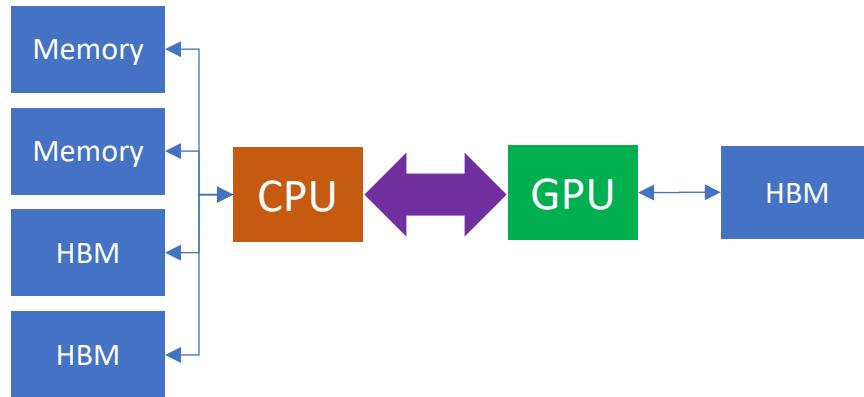
OpenMP

- Supports GPU offloading
- Compute kernels need to be annotated with directives
- Data transfers to/from GPU memory need to be annotated

```
void daxpy(int sz) {  
    double a;  
    double* x=(double*)malloc(sz);  
    double* y=(double*)malloc(sz);  
    //initialize  
    ...  
  
    #pragma omp target data map[to:x(0:sz)] map[tofrom:y(0:sz)]  
    {  
        #pragma omp teams distribute parallel for simd  
        for (int i = 0; i < sz; i++) {  
            y[i] = a * x[i] + y[i];  
        }  
    }  
}
```




Superchip/APU Programming



OpenMP

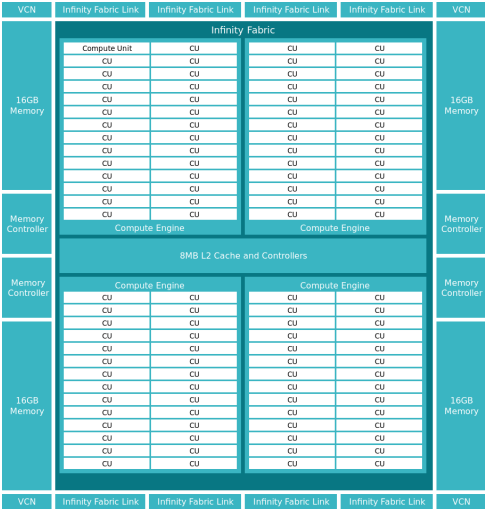
- Supports GPU unified memory
- Compute kernels need to be annotated with directives

```
void daxpy(int sz) {  
    #pragma omp requires unified_shared_memory  
    double a;  
    double* x=(double*)malloc(sz);  
    double* y=(double*)malloc(sz);  
    //initialize  
    ...  
  
    #pragma omp target teams distribute parallel for simd  
    for (int i = 0; i < sz; i++) {  
        y[i] = a * x[i] + y[i];  
    }  
}
```

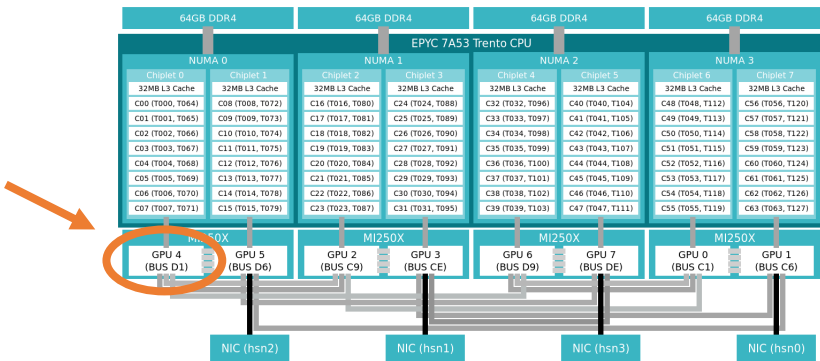
Types of Memory Architectures – Multi-node

Distributed memory link model

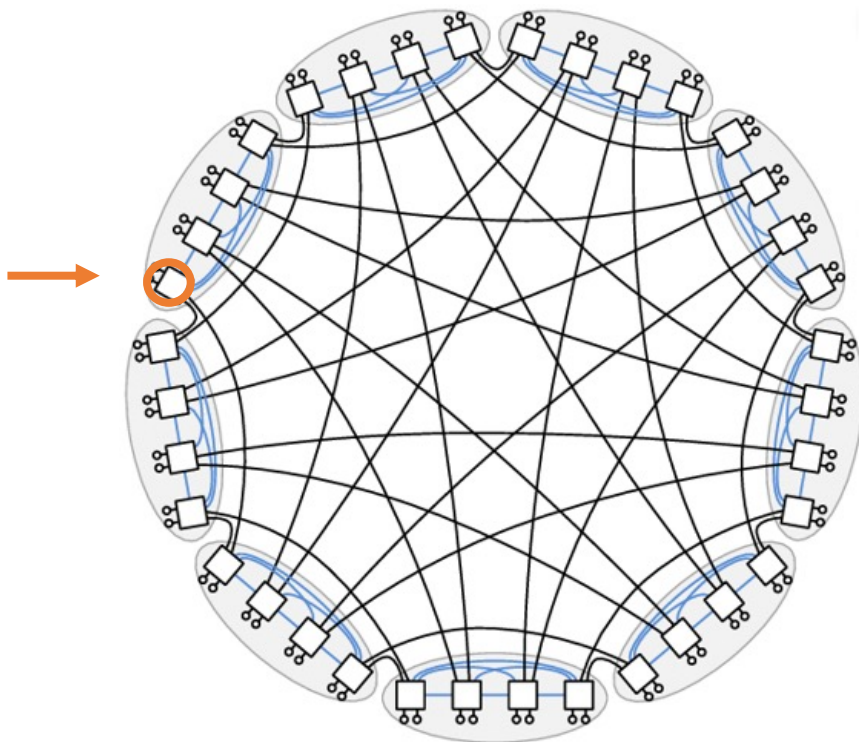
GPU



Node
Intraconnect



Nodes
Interconnect



A supercomputer...

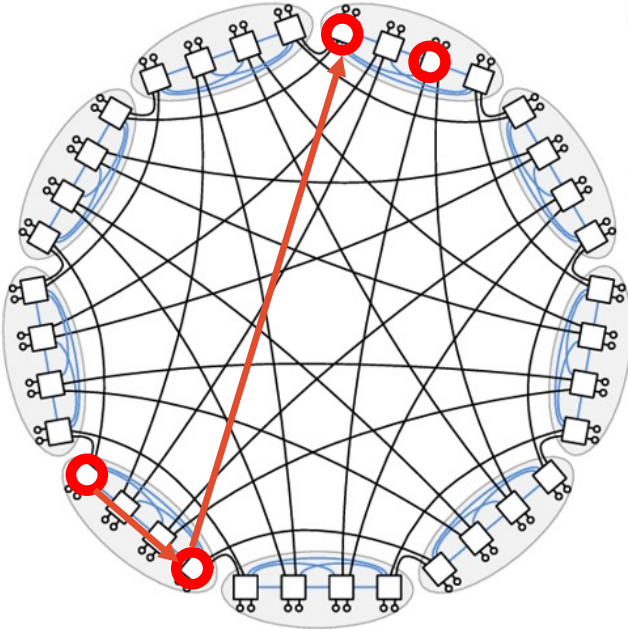
within a supercomputer...

within a supercomputer!



Multi-node Programming

Distributed memory model



Note: MPI is not the only distributed memory programming methodology. There are other models like Partitioned Global Address Space (PGAS) languages like Coarray Fortran, UPC, Chapel, ...



- Message Passing Interface (MPI) is a standardized message-passing standard designed for parallel computing architectures
- Programmers can define multiple processes, synchronize them and send/recv messages containing data
- MPI supports parallel I/O
- MPI now also supports GPU to GPU communication

```
#include <stdio.h>
#include <string.h>
#include <mpi.h>

int main(int argc, char **argv)
{
    char buf[256];
    int my_rank, num_procs;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &num_procs);

    if (my_rank == 0) {
        int other_rank;
        printf("We have %i processes.\n", num_procs);

        for (other_rank = 1; other_rank < num_procs; other_rank++)
        {
            sprintf(buf, "Hello %i!", other_rank);
            MPI_Send(buf, 256, MPI_CHAR, other_rank,
                     0, MPI_COMM_WORLD);
        }
    }
    ....
}
```



Example: Weather Forecasting

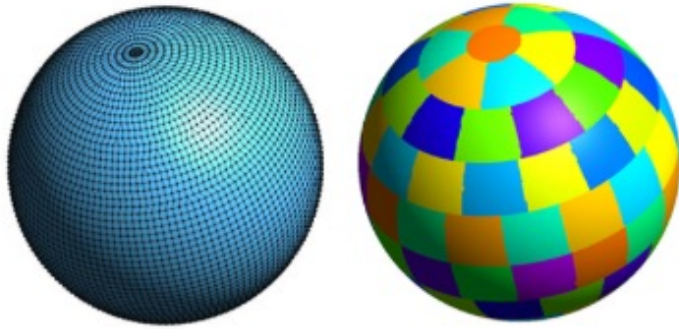


Figure 1. The IFS alternative dynamical core option:
Left, an example of an unstructured mesh for a low-resolution model. Right, the domain decomposition used in IFS; each patch represents the grid area owned by an MPI task.

Source: <https://www.olcf.ornl.gov/2015/08/18/the-future-of-forecasting/>

Step 0: Discretize mathematical model: global weather forecasting discretized over 3D grid

Step 1: Domain decomposition between nodes: each patch represents the grid area owned by an MPI task

Step 2: Communication: neighboring MPI tasks (patches) need to communicate

Step 3: Intranode parallelization: within each node OpenMP is used to parallelize across available CPU cores and GPUs

Hybrid programming:

MPI for internode parallelization

OpenMP for intranode parallelization



Agenda

- The Need for Speed Scale
- Current Trends in Supercomputing Technology
- Parallel Programming Models
- **Challenges**
- How to choose your track?

Challenge: Programming Language

Increasing popularity of modern programming languages:

- **C++** - dynamically developing, enabling software engineering, represented in ECP projects (e.g. Slate library, Kokkos)
- **Python** – high performance available through libraries like Numpy, Scipy or machine learning packages and tools like Tensorflow
- **Rust**, ...

“New” programming paradigms and languages: Legion, PaRSEC, UPC++

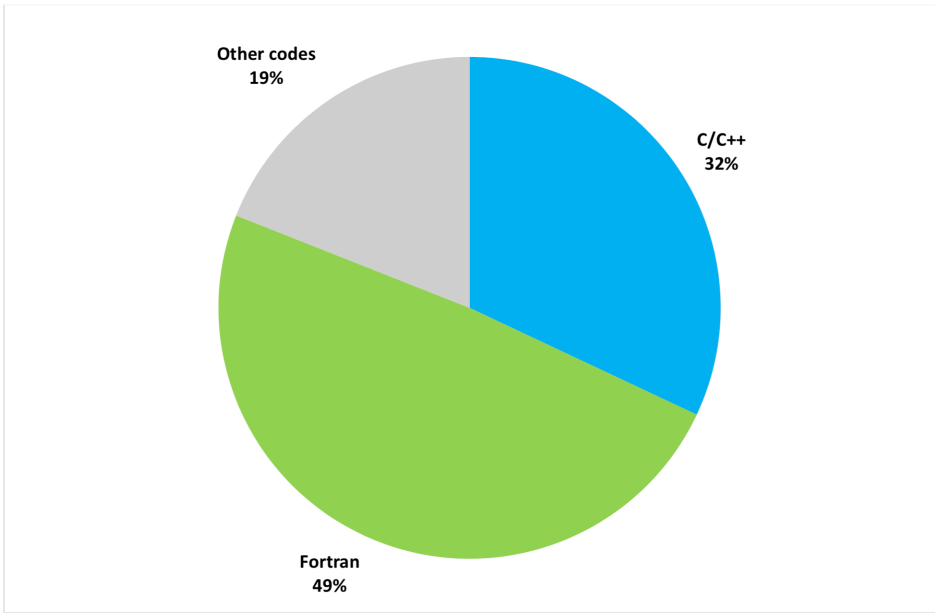


Figure 2 Programming languages as a breakdown of CPU hours on Magnus (1st half 2019)

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

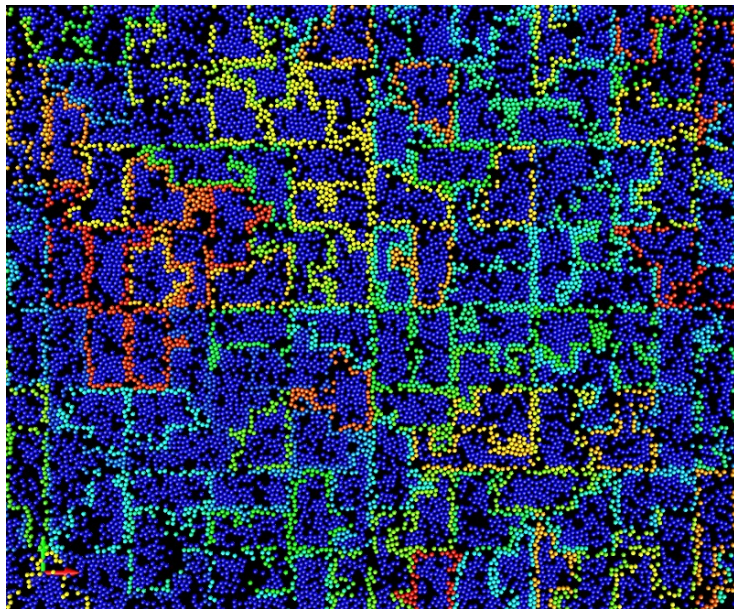
Rui Pereira et al. 2017. “Energy efficiency across programming languages: how do energy, time, and memory relate?” In Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2017). DOI:<https://doi.org/10.1145/3136014.3136031>

IHPCSS24 Talk: HPC Python, Ramses van Zon, Univ. Toronto, Wednesday 9am

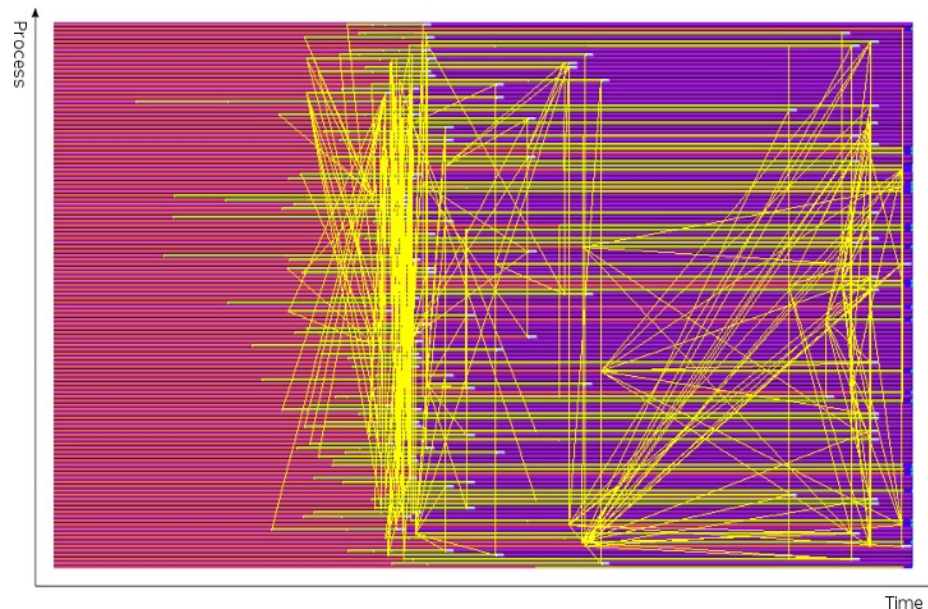
IHPCSS24 Talk: Software engineering, Erik Lindahl, Univ. Stockholm, Tuesday 4:30pm



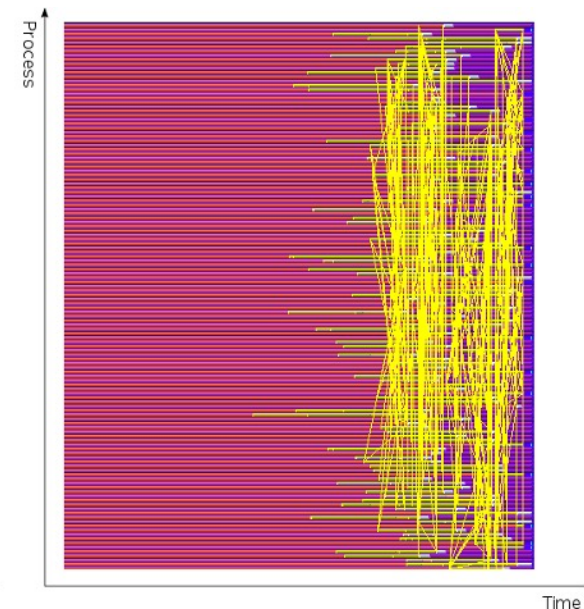
Challenge: Scalability



2D-box of interacting particles



Trace-based profiling of MPI processes



M. Cytowski and Z. Szymanska, "Large-Scale Parallel Simulations of 3D Cell Colony Dynamics: The Cellular Environment," in *Computing in Science & Engineering*, vol. 17, no. 5, pp. 44-48, Sept.-Oct. 2015, doi: 10.1109/MCSE.2015.66.

IHPCSS24 Talk: Performance Analysis & Optimization, Ilya Zhukov, JSC, Wednesday 9am



Challenge: Portability and Performance Portability

- **OpenMP**

- “Enabling HPC since 1997”
- target directive available since OpenMP 4.0 (2013)
- support for accelerators currently available in Clang, GCC, IBM XL, AOMP

- **OpenACC**

- joint effort to developed OpenMP-like standard for accelerators (2012)
- kernels directive
- support for accelerators currently available in PGI, GCC, Cray

Programming abstraction layers to hide complexity of accelerators

Kokkos ecosystem <https://kokkos.org>

Raja software library of C++ abstractions
<https://raja.readthedocs.io>



HPC Guru @HPC_Guru · Oct 22

All #HPC users will give up performance for portability?

May be true for 5% performance hit, not sure if it will hold for 50%.

#SYCL via @science_dot @thoefler @simonmcs



@science_dot · Oct 22

Replying to @science_dot @thoefler and 3 others

All HPC users will give up performance for portability. That's why nobody writes in assembly anymore. It's merely a question of how much. Some folks will give up 5%, others 50%.



9



3



14



<https://crpl.cis.udel.edu/ompvvsolve/>



Challenge: Reproducibility and complex workflows

- How to make sure that the results of my extremely complex workflow consisting of tens of packages and stages can be reproduced on:
 - Different supercomputing systems,
 - Same supercomputer but after programming and software stack upgrade,
 - My laptop,
 - My paper reviewer's laptop.
- One of the possible solutions: use containerized environments and workflow systems that integrate with containers

IHPCSS24 Talk: Containers, *Sarah Beecroft, Pawsey, Tuesday 2pm*

IHPCSS24 Talk: Workflow Tools, *Scott Callaghan, Univ. Southern California, Friday 9am*

Which programming language is best?

Which parallel programming model is the most appropriate?

Why the speaker looks confused as well?

Do I really need to learn Fortran?

Which architecture should I target?

Should I profile and optimize my code?

Should I use Python?

What are these marsupials he mentioned at the beginning?

Should I use GPUs?

Will my application be portable between different GPUs?

Is my code energy efficient?





This is why we love HPC!!!

- There is so much you can do
- Diverse tools and techniques
- A lot of things to discover and develop

Start with learning well established standards and methodologies and go from there.

Good luck!



Agenda

- The Need for Speed Scale
- Current Trends in Supercomputing Technology
- Parallel Programming Models
- Challenges
- **How to choose your track?**



Track 1: An introduction to shared-memory parallelism and accelerator programming

- Single-node programming, and provides techniques to parallelize codes over multiple CPUs, as well as offload processing to GPUs.
- It will also teach techniques to tackle challenges commonly faced in shared-memory programming and GPU offloading, such as:
 - load-balancing,
 - data race protection,
 - GPU offloading latency hiding.
- Track 1 is the opportunity to get a first experience with parallel programming. Therefore, Track 1 is most valuable for students who are discovering parallel programming.
- This track will be taught using OpenMP, a technology that has been the major solution to shared-memory programming in HPC for the better part of the last three decades. In addition to being highly optimized, this directive-based solution allows one to incrementally parallelize codes, which makes it widely popular.
- Prerequisites: this track has no prerequisite other than a familiarity with compiled programming languages as the teaching material is available in C and FORTRAN.



Track 2: Advanced distributed-memory programming

- Multi-node programming with the MPI library, leveraging the processing power, memory bandwidth and filesystem IO available across distributed-memory architectures.
- Track 2 is the opportunity for students with existing MPI experience to learn to write efficient scalable programs for large HPC systems by understanding more about:
 - the internals of the MPI library;
 - advanced use of collective operations;
 - MPI derived datatypes.
- Prerequisites: due to the more advanced nature of this track, participants are required to have existing knowledge of basic MPI programming.
- Participants must already be able to compile and run an MPI program, construct simple point-to-point communications, use basic collective operations such as broadcast and reduce, use non-blocking operations and construct basic derived datatypes.
- Similarly to Track 1, the teaching content in this track will be provided in C, C++ or Fortran.



Which track should I choose?

Disclaimer: Every individual case is unique. This slide is a collection of recommendations rather than definite answer to the above question. There might be other important reasons related to your individual case. If in real doubt – talk to your mentor !

- I have never done parallel programming – **Track 1**
- I have done some parallel programming but very basic and never used MPI or OpenMP – **Track 1**
- I need to implement GPU acceleration in my code – **Track 1**
- My code uses MPI and it scales relatively well, I am familiar with MPI and am looking to further improve its parallel performance:
 - Improving MPI performance – **Track 2**
 - Improving single node performance (hybrid approach), including GPU – **Track 1/Track 2**
- My code uses MPI and OpenMP including GPU offloading – **Track 2**
- I don't use C/C++ or Fortran in my work and although I have some basic knowledge to follow lectures, I am not even sure if OpenMP and/or MPI are relevant to my case – **talk to your mentor !**
- My work is mainly to develop workflows (e.g. bioinformatics) rather than codes for each workflow component – **Track 1**
- I was sure about my choice, but after Maciej's talk I'm not – **sorry! talk to your me!**



THANK YOU
QUESTIONS?

