

Supercomputing at the exascale and beyond: future trends and challenges



Erik Draeger, Lawrence Livermore National Laboratory (LLNL)
Director, High Performance Computing Innovation Center (HPCIC)
Deputy Director of Application Development for the DOE Exascale Computing Project (ECP)

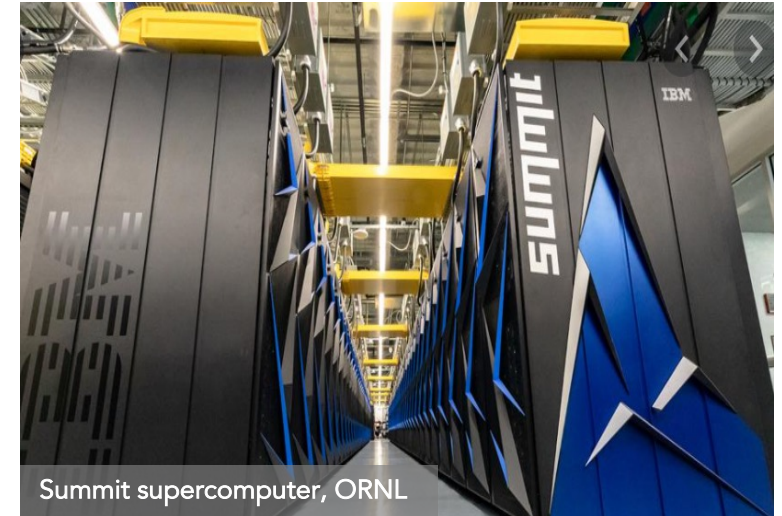
IPAM Workshop “New Mathematics for the Exascale: Applications to Materials Science”

UCLA, March 2023

DOE labs host the nation's largest supercomputers

- DOE has been a world leader in high performance computing for decades.
- The fastest supercomputer in the world has been at a DOE national lab for **14 of the past 25 years**.
- Today, **three of the six fastest supercomputers** in the world are at DOE labs (including the fastest!)

Position	Name	Country	Petaflops	Power (MW)
1	Frontier	USA	1102.0	21.1
2	Fugaku	Japan	537.2	29.9
3	LUMI	Finland	309.1	6.0
4	Leonardo	Italy	174.7	5.6
5	Summit	USA	200.8	10.1
6	Sierra	USA	125.7	7.4



Summit supercomputer, ORNL

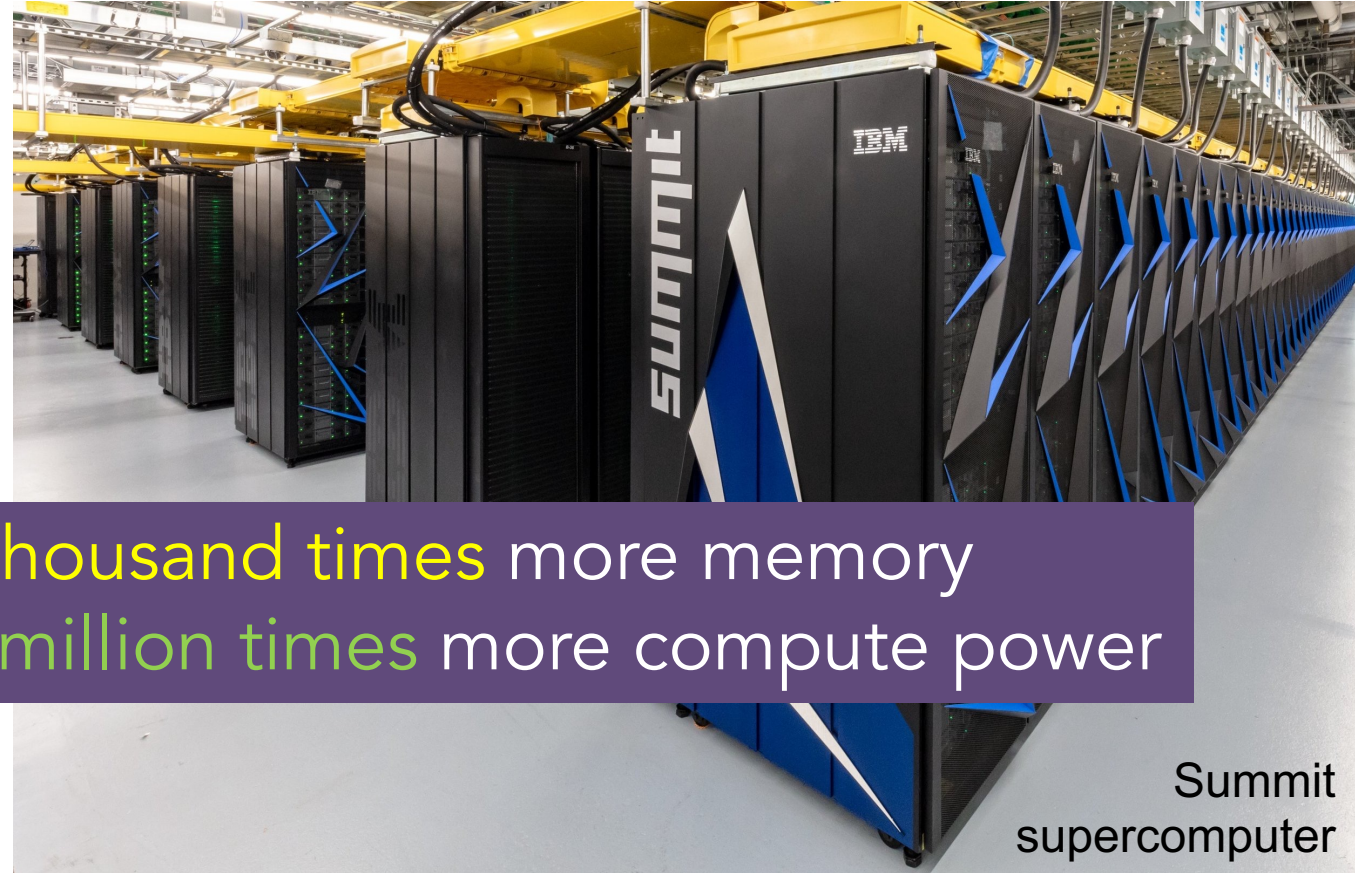


Sierra supercomputer, LLNL

Parallel computing enables new science



16" Macbook Pro

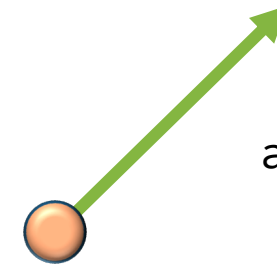


Supercomputers let us do simulations that are **bigger**, **faster**, and **more accurate**

More memory = bigger simulations

- Memory in the context of supercomputing primarily enables us to study systems we couldn't otherwise study
- One "thing" is represented by a set of numbers, i.e. location, velocity, shape, etc.
- Total computer memory / thing size = how many things you can simulate.

velocity = (0.07, 24 bytes, 1, 0.03778)



atom 4 bytes

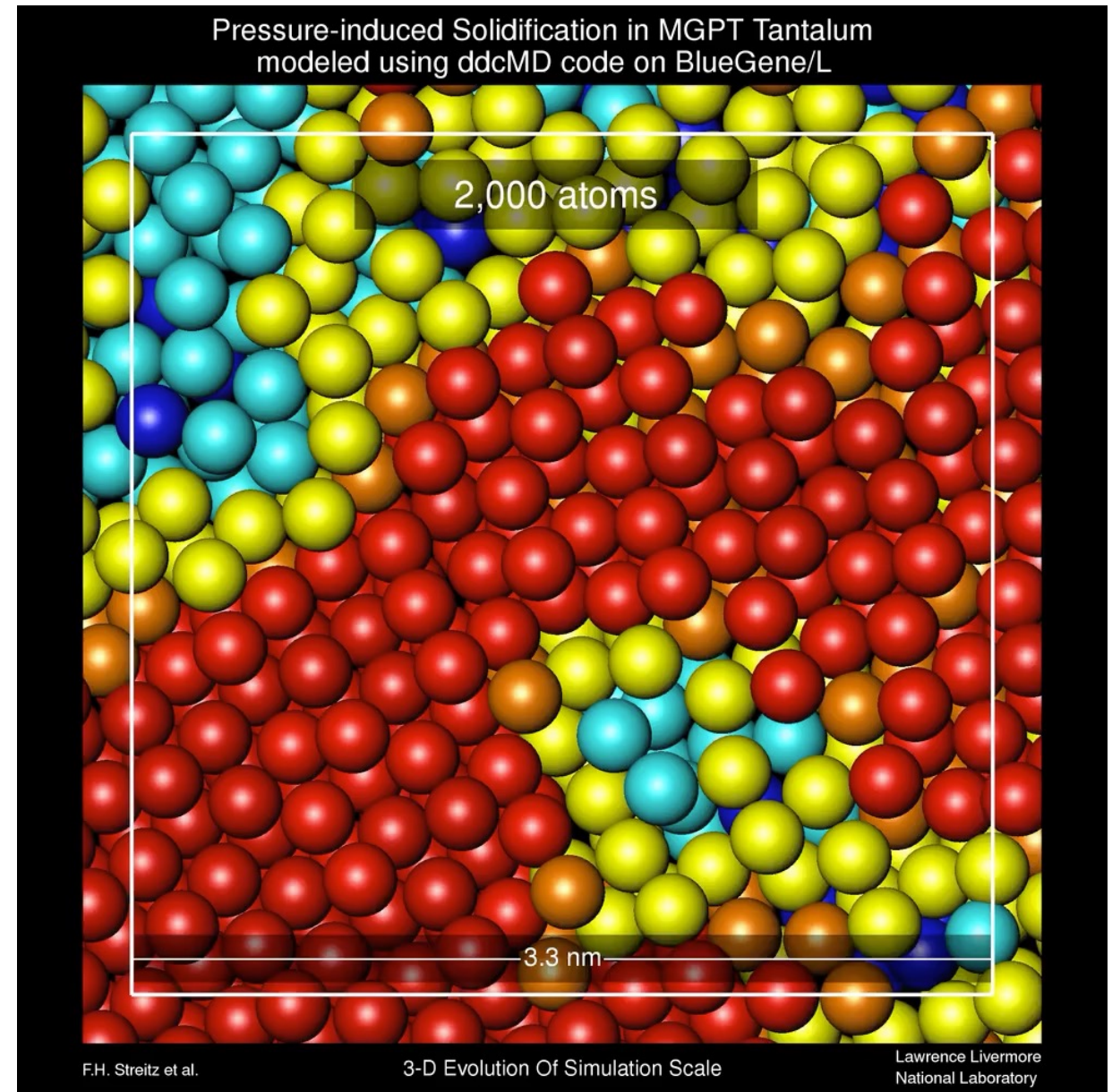
position = (2.8, 24 bytes, 9.00324)

128 GB = 2.64 billion carbon atoms
= 3,600,000 red blood cells
= 2.4 km² seismic model



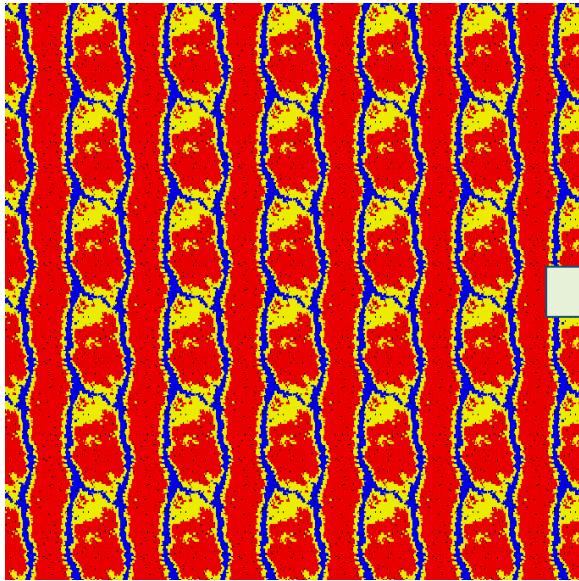
How big is big enough?

- In 2005, researchers were studying how molten tantalum freezes under pressure.
- For small simulations, the structures that formed were roughly the size of the simulation box.
- It took a supercomputer to run a large enough simulation to properly capture the physics of the freezing process.



How big is big enough?

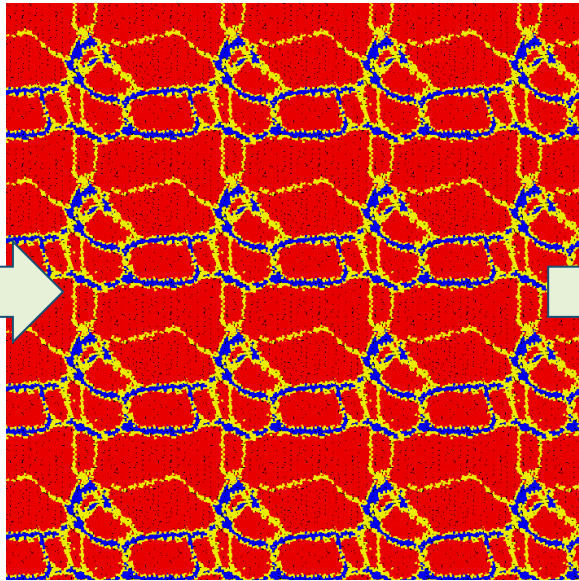
64,000 atoms



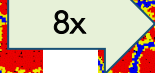
Too small!



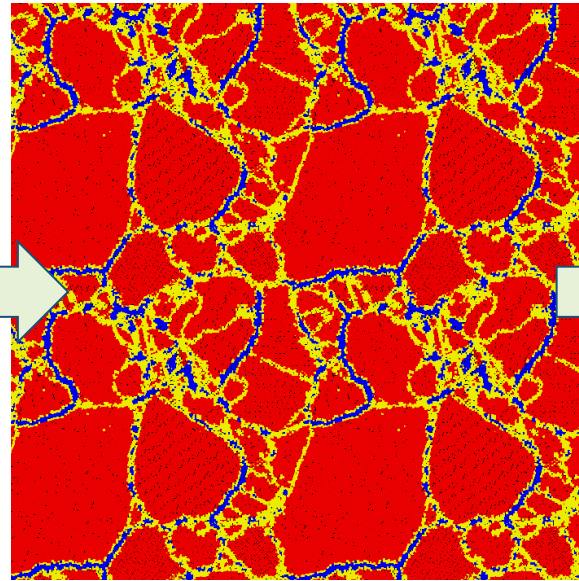
256,000 atoms



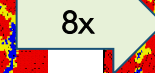
Too small!



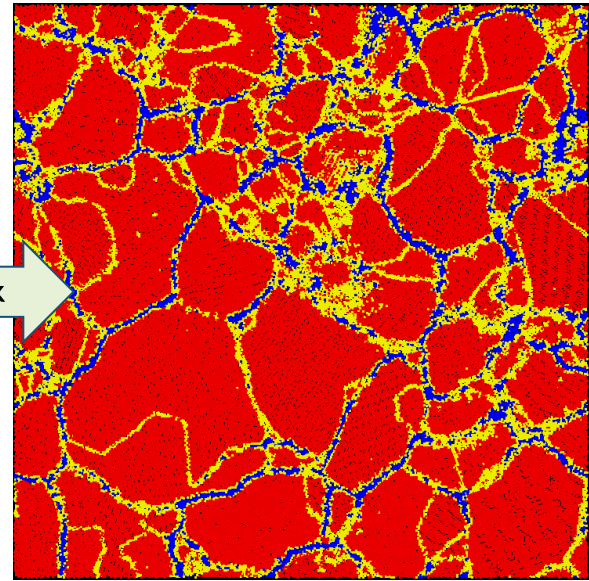
2,048,000 atoms



Big enough?



16,384,000 atoms



Yes, it was.

The only way to be sure a simulation is big enough is to run a bigger one.

More CPUs = faster simulations

- Supercomputing enables us to study systems we couldn't otherwise study.
- Ignoring memory, the amount of math that needs to be done is proportional to the system size (sometimes the system size to the second or third power). The time to solve a given problem is dependent on the number of floating point operations per second (FLOPs) a given system can perform.



One hour of work
at 200 PetaFLOPs

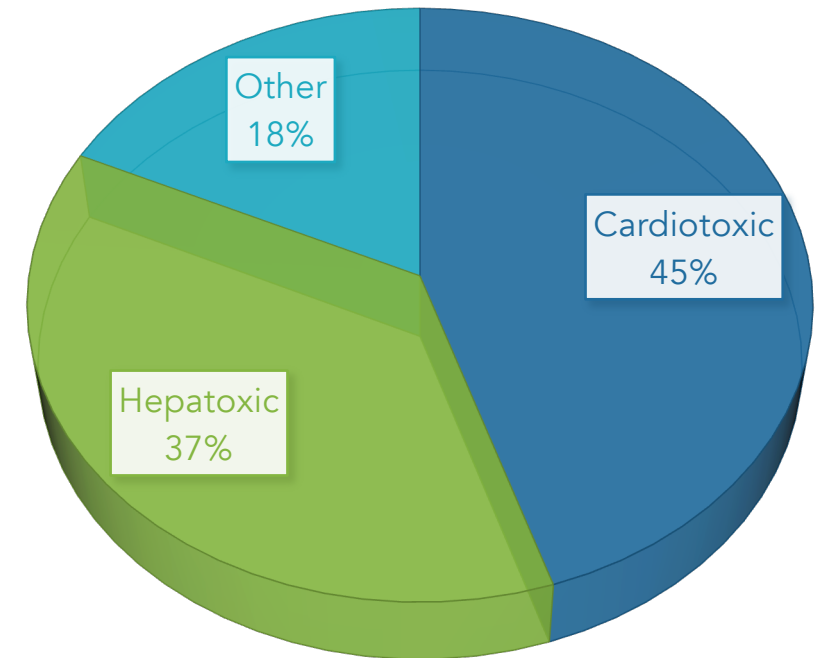


736 years of work
at 31 GigaFLOPs

How fast is fast enough?

- Researchers at IBM and LLNL wanted to see if simulation could predict drug side effects in patients, specifically **increased risk of cardiac arrhythmias**.
- To accurately simulate a full human heart **for even an hour would have taken months** on a typical compute cluster.
- The Cardioid project was started to create a faster, more accurate tool for studying arrhythmias.

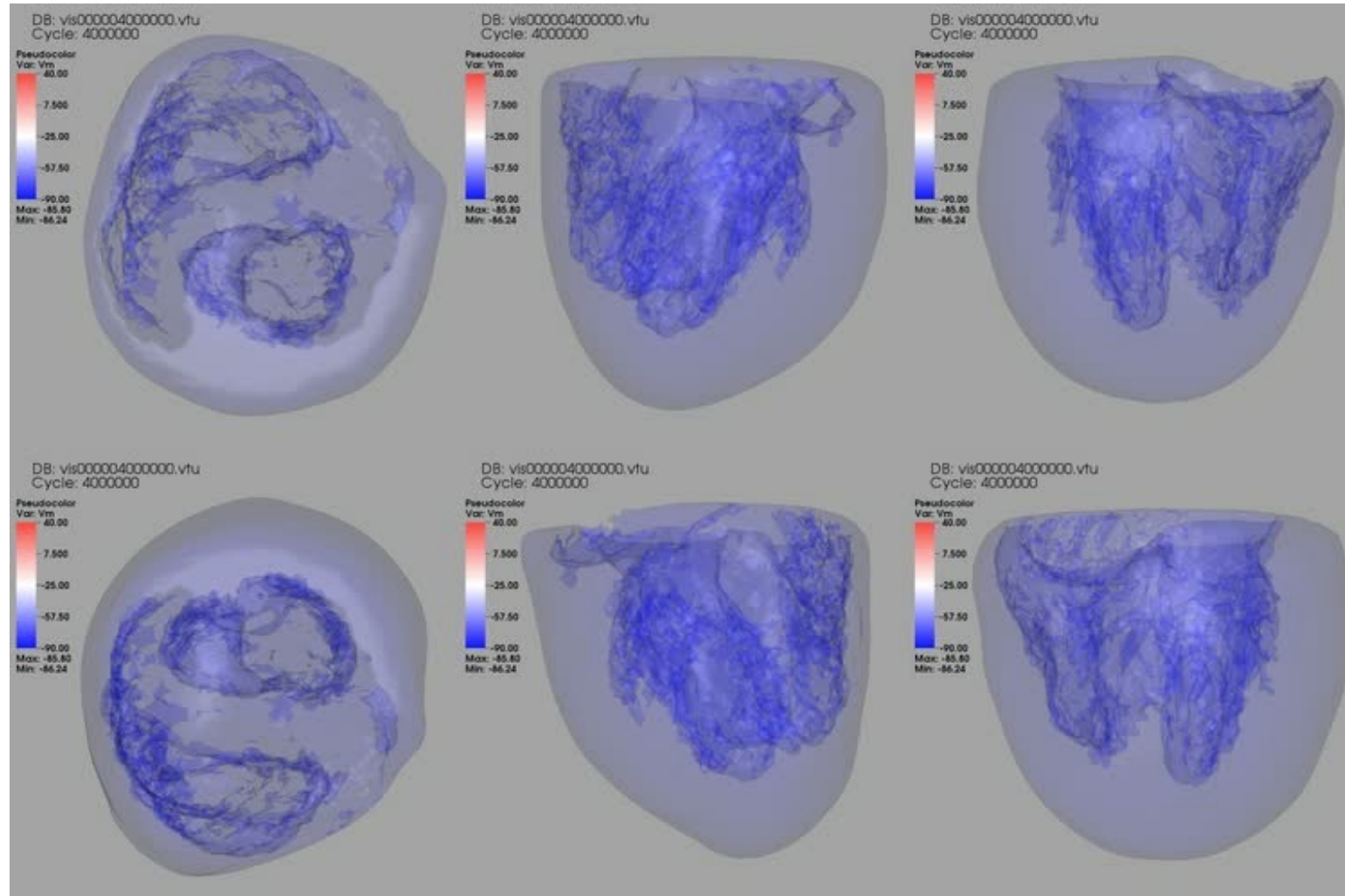
Drug recalls 1994-2006



GE Healthcare – Innovating Preclinical Development (March 2012)

How fast is fast enough?

- Cardioid was ultimately able to run a full human heart simulation in almost real time: simulating an **hour of heartbeats** on the Sequoia supercomputer took **an hour and seven minutes**.
- This opens the door to using simulation to screen new drugs before doing expensive and costly patient trials.

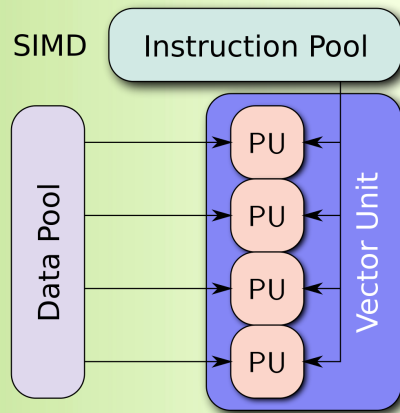


We are at a transition point in HPC

Vector Era

MFLOP/s - GFLOP/s

- Parallelism through vector processors.
- Codes often written at very low level to make optimal use of hardware.



1980s

Distributed Memory Era

GFLOP/s - TFLOP/s – PFLOP/s

- Parallelism through MPI.
- Using an optimal parallel algorithm was critical to avoid duplication of work or unnecessary communication.
- Once distributed, code could be treated serially.

- For the most part, an MPI code ran anywhere. For best performance, key kernels could be tuned.
- As CPU frequencies stopped increasing, parallelism became more extreme and specialized hardware more common.

10s to 100s of cores

1000s of cores

10⁴ to 10⁶ cores

1990s

2000s

2010s

We are at a transition point in HPC

Heterogeneous Era

PFLOP/s - EFLOP/s

- CPUs + accelerators with separate memory spaces to start, unclear what else will join the fray.
- Massive fine-grained parallelism required.
- Programming model has to match the architecture.
- Architectural landscape is changing rapidly, with an unclear future.

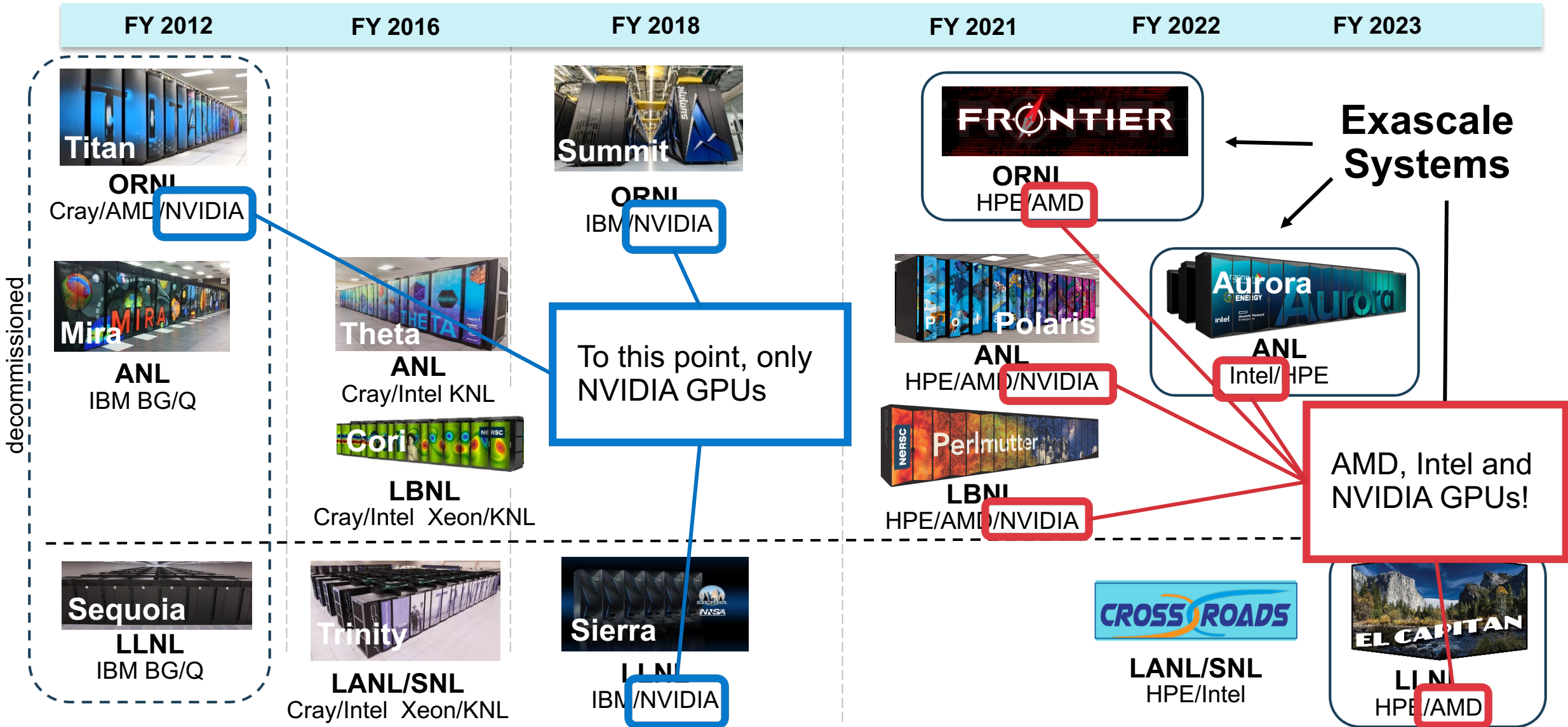
Heterogeneity is the new reality

- Computational horsepower has significantly outpaced memory capacity and speed.
- Separate memory spaces add complexity, and can cause performance issues (e.g. NUMA) or errors if not handled correctly.
- Performance or portability?
- Refactoring an existing code is a lot of work! You really don't want to have to do it again in ten years.

2010s

2020s

DOE HPC Roadmap to Exascale Systems



Frontier

Frontier is the world's fastest supercomputer and the world's first supercomputer to break the performance barrier known as exascale, debuting in May 2022 at 1.1 exaflops.

Compute Node

1 AMD EPYC CPU
4 AMD MI250X GPUs

System Size

>9,000 nodes

Memory

4.6 PB DDR4
4.6 PB HBM2e
36 PB on-node storage

On-node Interconnect

AMD Infinity fabric
Node-level coherence

System Interconnect

Four-port Slingshot network
100 GB/s



1.1
EXAFLOPS

FRONTIER CAN DO MORE THAN **1 QUINTILLION** CALCULATIONS PER SECOND.

1
SECOND

IF EACH PERSON ON EARTH COMPLETED **ONE CALCULATION PER SECOND**, IT WOULD TAKE MORE THAN **4 YEARS** TO DO WHAT AN EXASCALE COMPUTER CAN DO IN **1 SECOND**.

6,000
GALLONS

OF WATER IS MOVED THROUGH THE SYSTEM **PER MINUTE** BY FOUR **350-HORSEPOWER PUMPS**. THESE POWERFUL PUMPS COULD FILL AN **OLYMPIC-SIZED SWIMMING POOL** IN ABOUT **30 MINUTES**.

700
PETABYTES

FRONTIER'S ORION STORAGE SYSTEM HOLDS **33 TIMES** THE AMOUNT OF DATA HOUSED IN THE **LIBRARY OF CONGRESS**.

8,000
POUNDS

EACH CABINET WEIGHS THE EQUIVALENT OF **2 FULL-SIZE PICKUP TRUCKS**.

40
MEGAWATTS

FRONTIER'S MECHANICAL PLANT CAN COOL THE EQUIVALENT POWER DEMAND OF ABOUT **30,000 U.S. HOMES**.

Aurora

Argonne's upcoming exascale supercomputer will leverage several technological innovations to support machine learning and data science workloads alongside traditional modeling and simulation runs.

PEAK PERFORMANCE

≥2 Exaflop DP

Intel® X^e ARCHITECTURE-BASED GPU

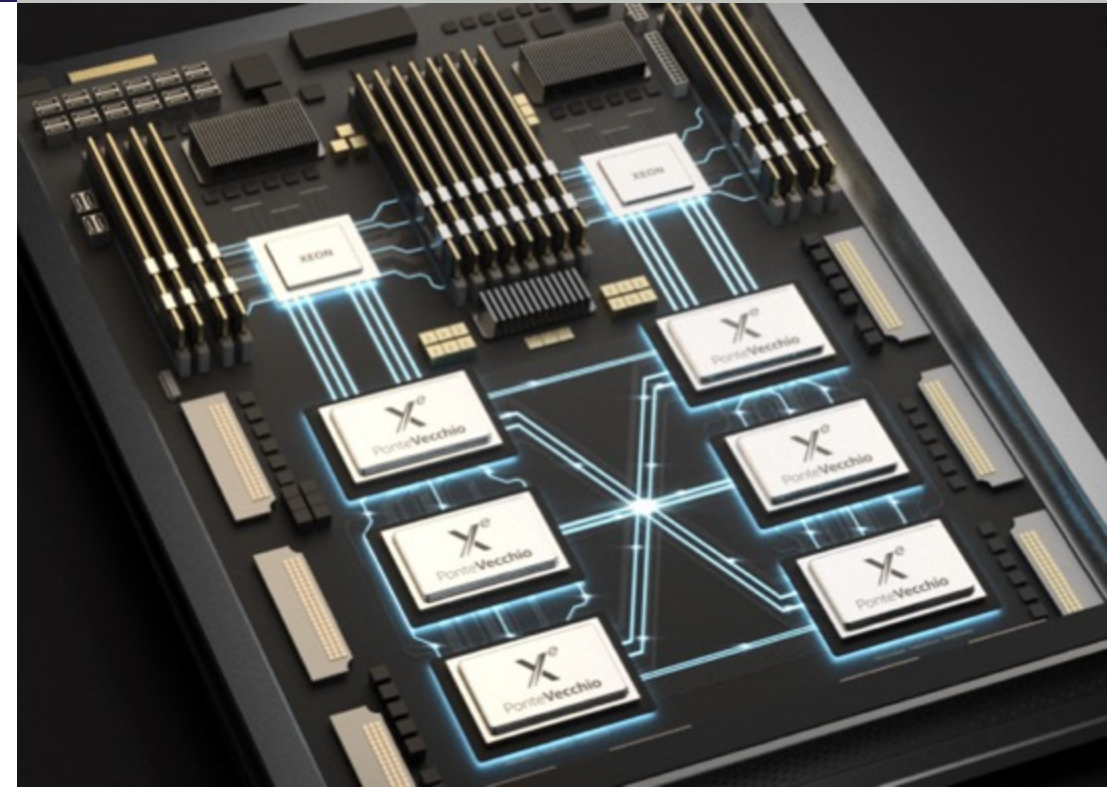
Data Center GPU Max Series

INTEL® XEON® SCALABLE PROCESSOR

Intel Xeon CPU Max Series

PLATFORM

HPE Cray EX



Compute Node

2 Intel® Xeon® CPU Max Series processors; 6 Intel® Data Center GPU Max Series GPUs; Unified Memory Architecture; 8 fabric endpoints; RAMBO

GPU Architecture

Intel® Data Center GPU Max Series; Tile-based chiptlets, HBM stack, Foveros 3D integration, 7nm

CPU-GPU Interconnect

CPU-GPU: PCIe
GPU-GPU: X^e Link

System Interconnect

HPE Slingshot; Dragonfly topology with adaptive routing

Network Switch

25.6 Tb/s per switch, from 64–200 Gbs ports (25 GB/s per direction)

High-Performance Storage

≥230 PB, ≥25 TB/s (DAOS)

Programming Models

Intel oneAPI, MPI, OpenMP, C/C++, Fortran, SYCL/DPC++

Node Performance

>130 TF

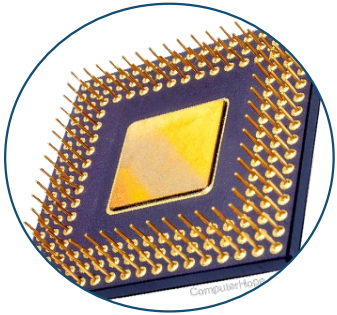
System Size

>10,000 nodes

Computing hardware trends point to even more specialization

CPUs

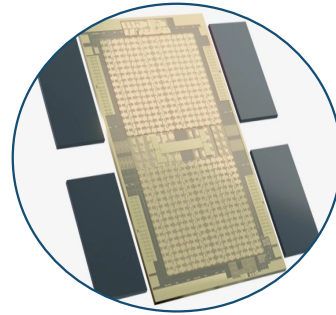
1950s -



- Still remains integral to computing architectures
- Speed improvements are waning

GPUs

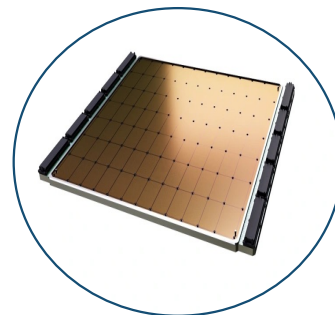
2010s -



- Initially specialized for graphics
- Highly parallel and energy efficient
- Suitable for many (not all) ASC workloads

AI Accelerators

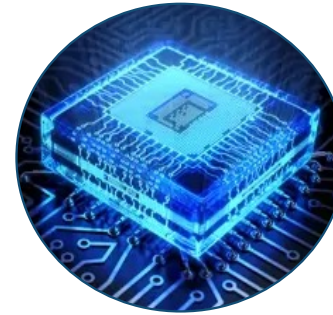
2020s -



- Specialized for machine learning
- Huge industry investments
- Work with vendors to adapt these technologies for DOE applications

Specialization

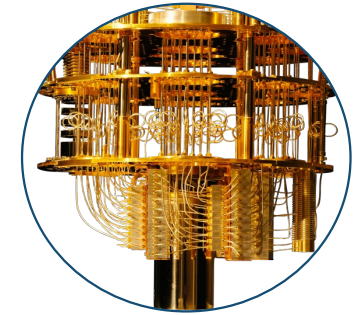
Late 2020s? -



- Increased specialization as chip technology advances
- System-level heterogeneity
- New business models for chip design

Quantum

2030s? -

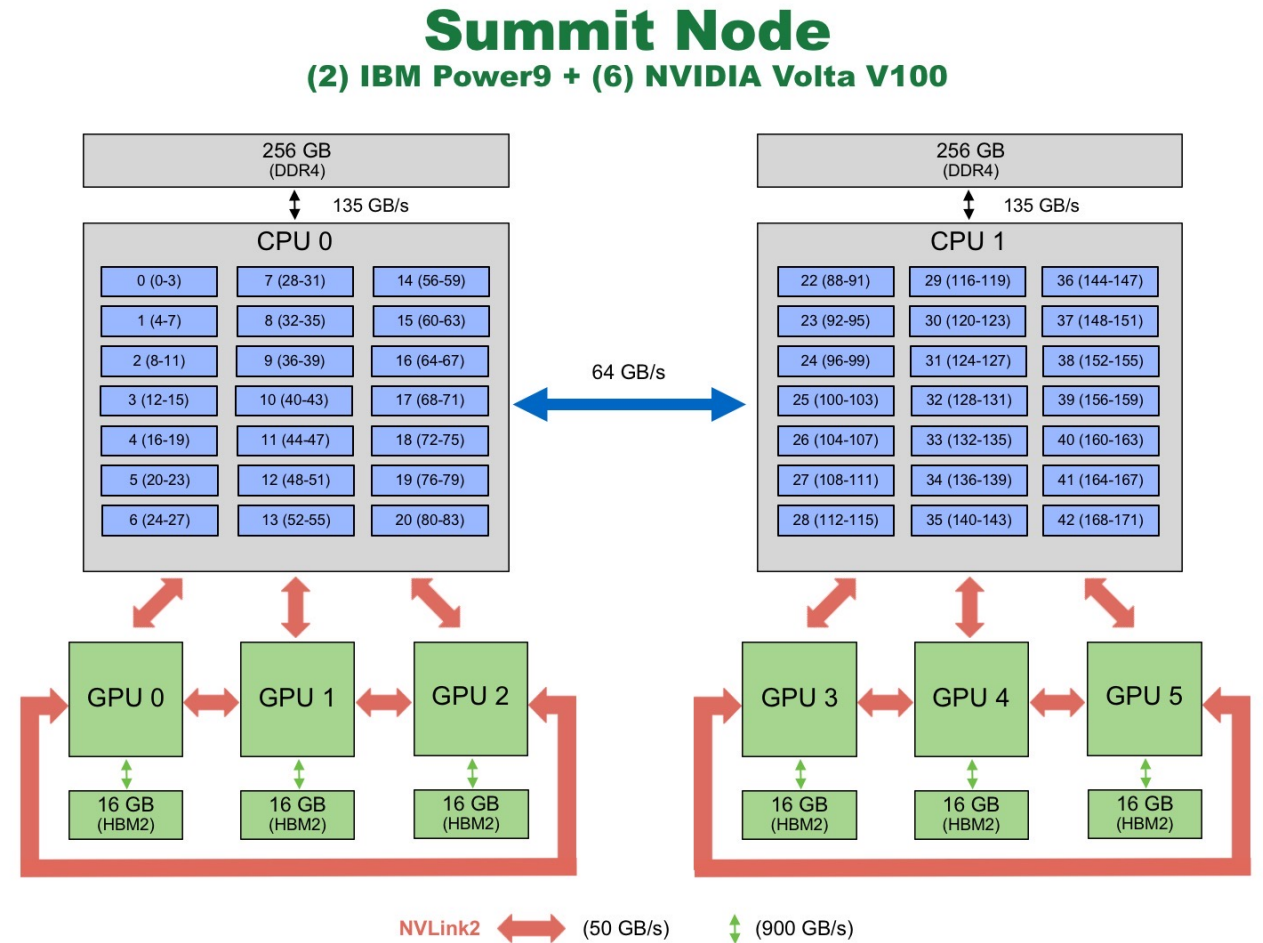


- Potential for quantum to act as another accelerator type
- Amenable to a very limited number of applications/ algorithms

The rate of change in computing architectures is increasing as more creative approaches are required to overcome fundamental limitations in chip performance

Why heterogeneous computing is hard

- Data movement is now expensive relative to compute. Parallel algorithms need to be written to minimize transfers.
- Having multiple memory spaces requires careful bookkeeping. Even hardware-unified memory (e.g. Summit/Sierra) is not without pitfalls.
- Multiple accelerator vendors means compute kernels must either have separate versions for each architecture or an abstraction layer.



Why using accelerators is hard: SIMD/SIMT

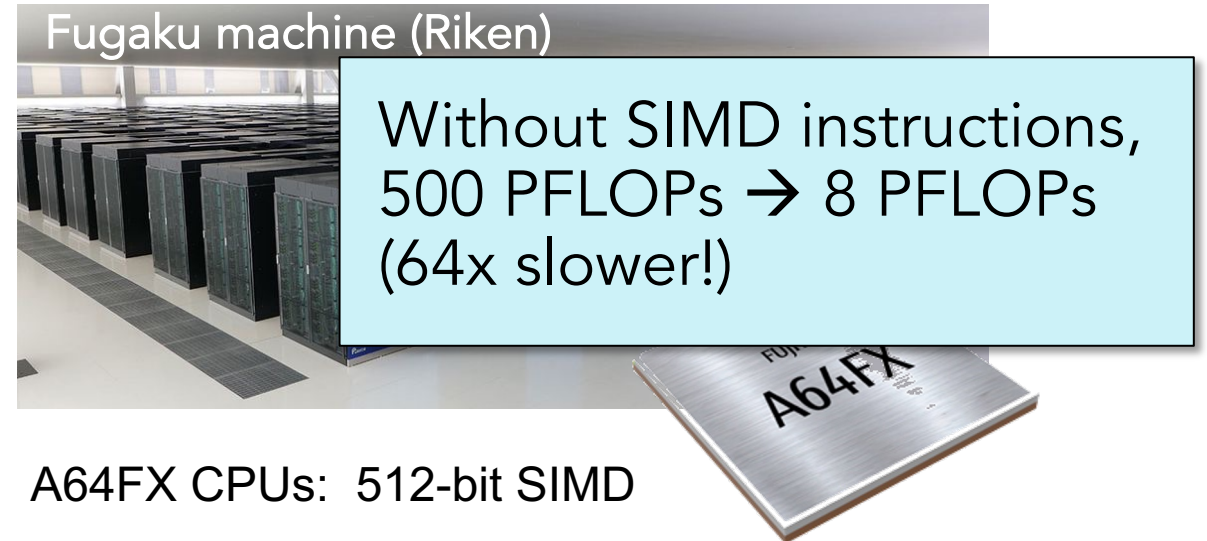
- GPUs use SIMT (Single Instruction, Multiple Threads). GPU architecture is designed around the assumption of highly concurrent workloads.
- Threads that follow different code paths are executed separately (sequentially).
- All CPUs now utilize vector instructions (SIMD) to achieve their advertised peak performance.
- SIMD = Single Instruction, Multiple Data. Requires chunks of data all traversing the same code path at the same time.
- If compiler can't find a full SIMD instruction, it reverts to sequential.

Scalar Operation

$$\begin{array}{l} A_1 \times B_1 = C_1 \\ A_2 \times B_2 = C_2 \\ A_3 \times B_3 = C_3 \\ A_4 \times B_4 = C_4 \end{array}$$

SIMD Operation

$$\begin{array}{l} A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \times \begin{array}{l} B_1 \\ B_2 \\ B_3 \\ B_4 \end{array} = \begin{array}{l} C_1 \\ C_2 \\ C_3 \\ C_4 \end{array}$$



Who should care about this?

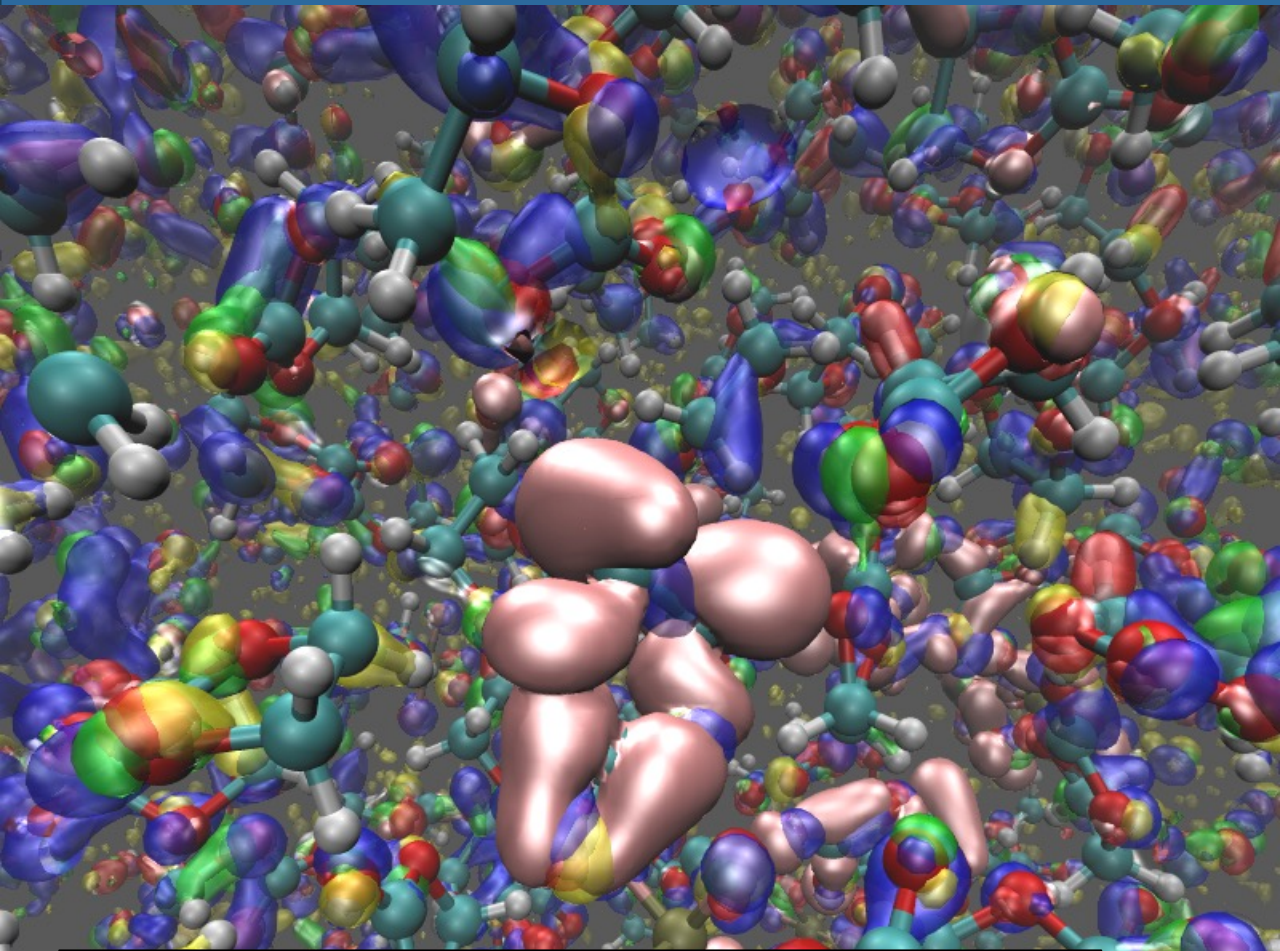
- **People who want to write new codes**
 - Designing new codes requires a clear understanding of the trade-offs of decisions at all levels (language, programming model, data structure, communication, I/O, etc.)
- **People who maintain and modernize existing codes**
 - Adapting an existing code to new architectures is a more constrained problem. Fewer decisions to make but greater consequence of error.
- **People who run code**
 - You need to understand the equation you are solving, the algorithm you are using and how it maps onto the system to use compute resources effectively.

Best practices from a decade ago no longer apply. Can we future-proof our codes any better this time around?

GPUs have forced us to reevaluate everything

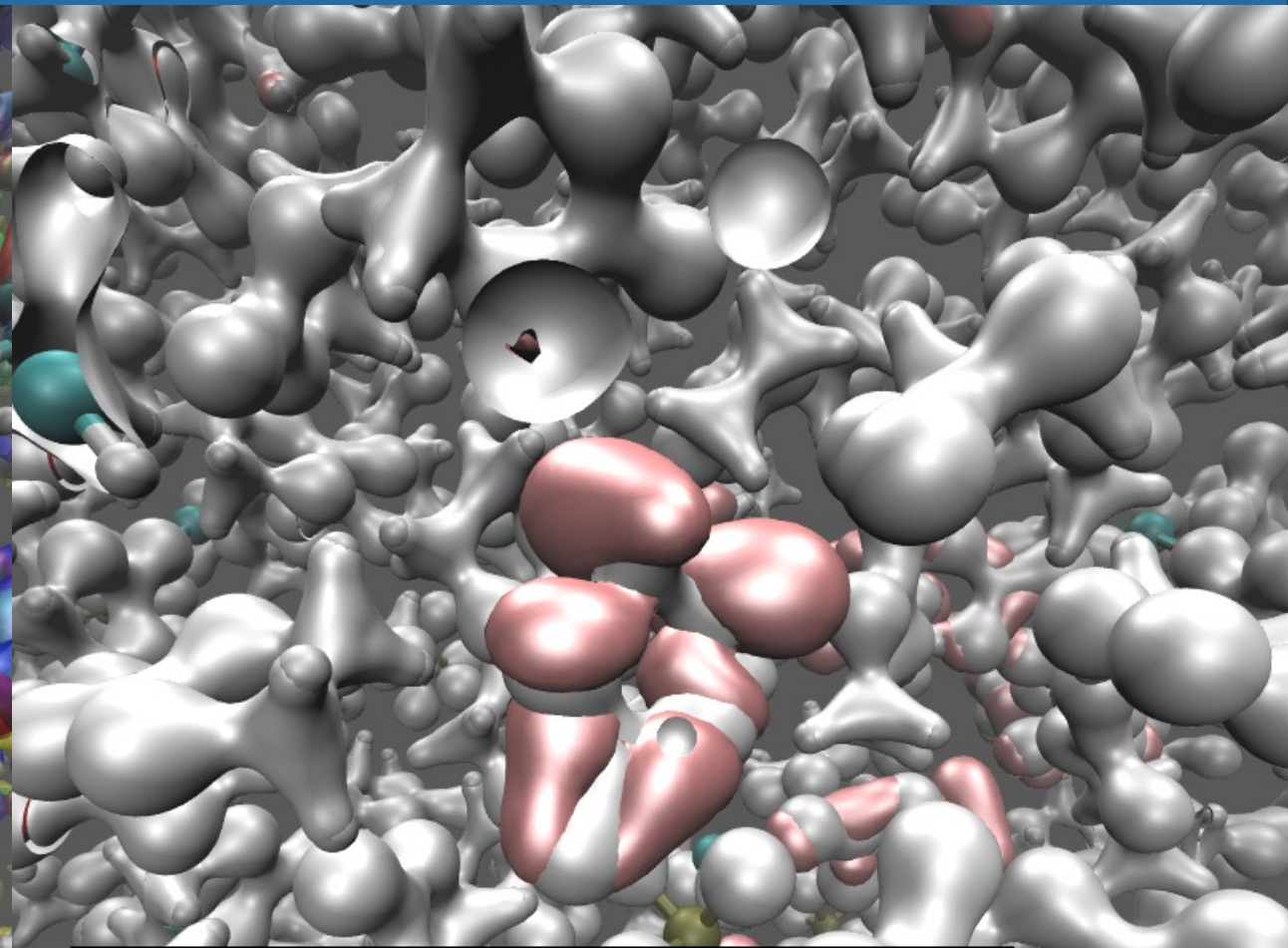
- The rapid change from distributed memory, CPU-only systems to heterogeneous, CPU-GPU has shaken up computational science.
- Some algorithms are fundamentally incompatible with SIMD/SIMT architectures. Others need to be carefully tuned for each type of GPU.
- Long-standing codes were designed around assumptions that no longer hold, it's unclear how to adapt them for an uncertain future.

Example: Density Functional Theory



Many-body Schrödinger Equation:

- Exact
- $O(N!)$ complexity



Kohn-Sham Equation:

- Surprisingly accurate
- $O(N^3)$ complexity

Example: Density Functional Theory

$$-\Delta\phi_i + V(\rho, \mathbf{r})\phi_i = \varepsilon_i\phi_i \quad i = 1 \dots N_{\text{el}}$$

Kohn-Sham Equation

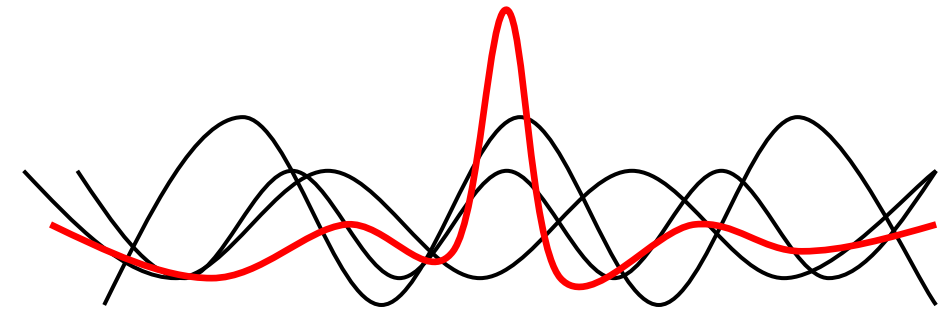
where

$$V(\rho, \mathbf{r}) = V_{\text{ion}}(\mathbf{r}) + \int \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' + V_{\text{xc}}(\rho(\mathbf{r}), \nabla\rho(\mathbf{r}))$$

$$\rho(\mathbf{r}) = \sum_{i=1}^{N_{\text{el}}} f_i |\phi_i(\mathbf{r})|^2$$

$$\int \phi_i^*(\mathbf{r}) \phi_j(\mathbf{r}) d\mathbf{r} = \delta_{ij}$$

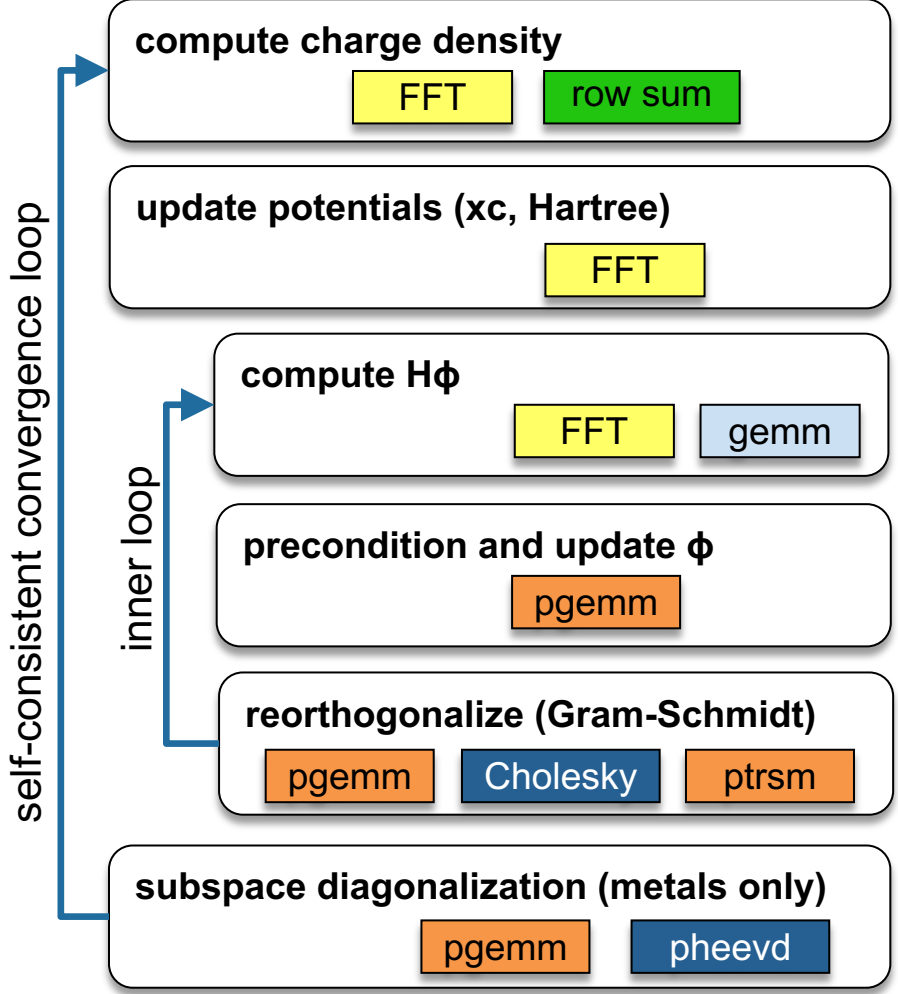
- Constrained nonlinear eigenvalue problem
- Plane wave basis allows for efficient evaluation of terms in reciprocal space, but allows no opportunities to leverage sparsity / short-range interactions.



Represent each orbital as a Fourier series of plane waves

$$\phi_j(\mathbf{r}) = \sum_{|\mathbf{k}+\mathbf{q}|^2 < E_{\text{cut}}} c_{\mathbf{k}+\mathbf{q},j} e^{i\mathbf{q}\cdot\mathbf{r}}$$

Example: communication profile of Qbox DFT code



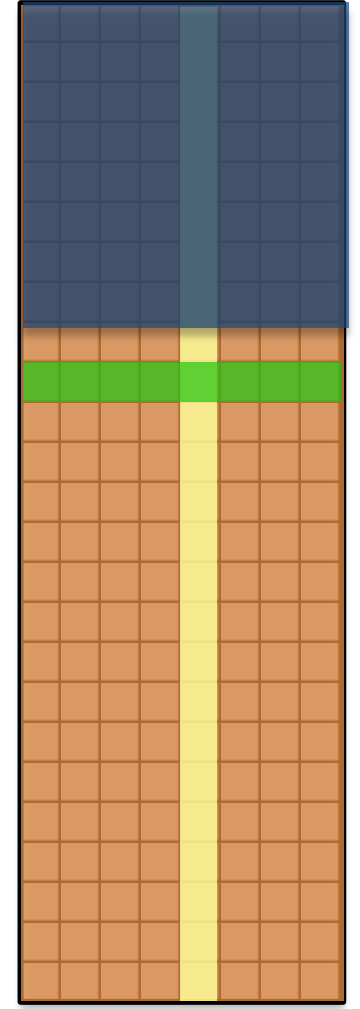
$$\rho(\mathbf{r}) = \sum_{i=1}^{N_{el}} f_i |\phi_i(\mathbf{r})|^2$$

$$\int \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' + V_{xc}(\rho)$$

$$-\Delta\phi_i + V(\rho, \mathbf{r})\phi_i$$

$$\phi_i \rightarrow \phi_i - \alpha K H \phi_i$$

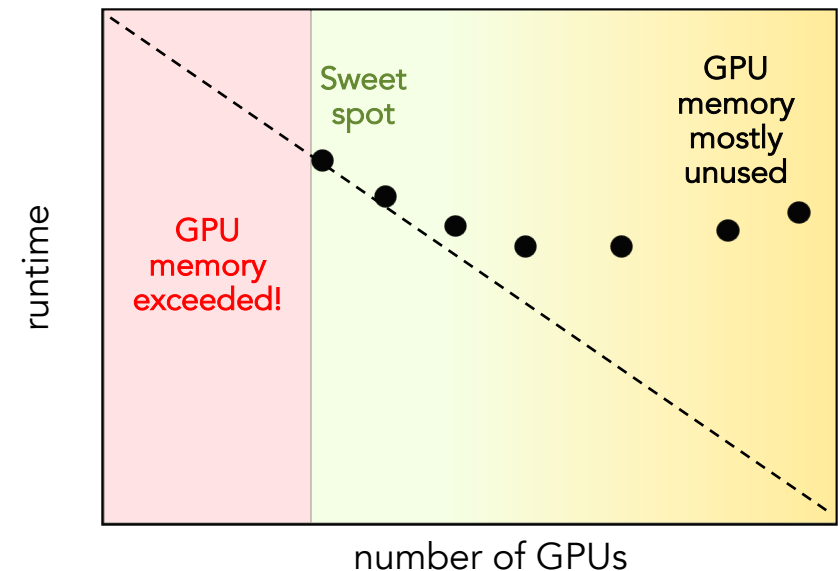
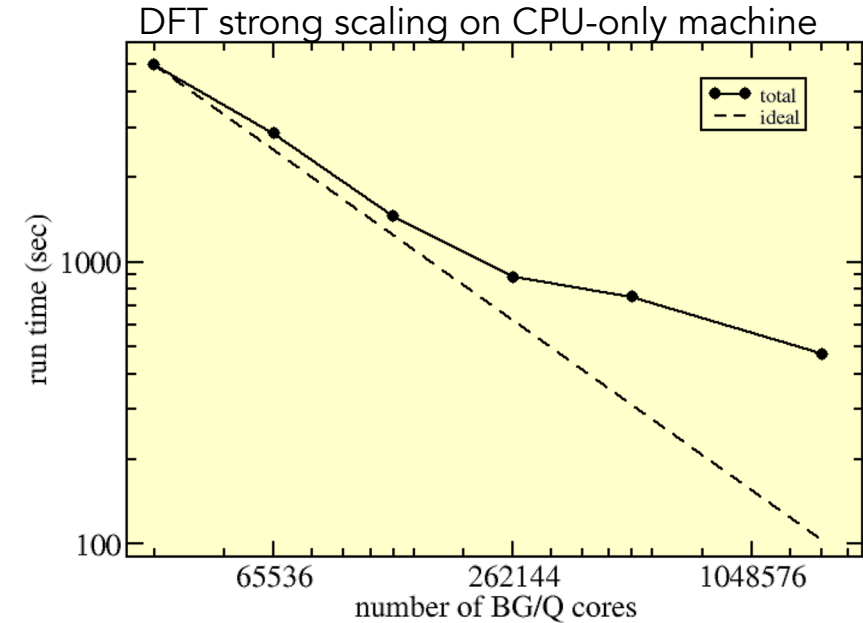
$$\int \phi_i^*(\mathbf{r}) \phi_j(\mathbf{r}) d\mathbf{r} = \delta_{ij}$$



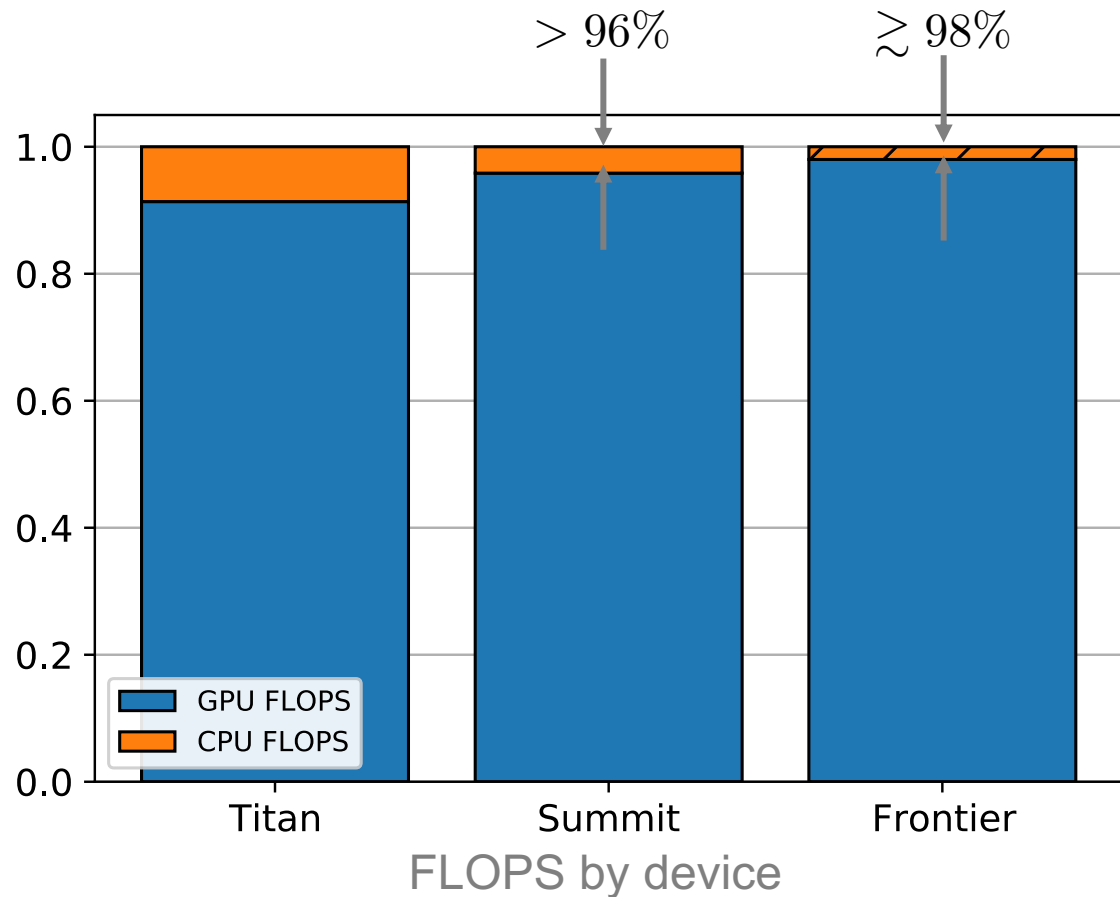
MPI process grid

Need to redesign codes around data movement, both within a node and between nodes

- Homogeneous architectures require a straightforward balance of computation vs. communication. Basically, you just strong scale until there is too little work per task.
- Codes built on libraries that don't map to heterogeneous now have to be rewritten. For example, ScaLAPACK was written for homogeneous distributed-memory architectures, we need a new heterogeneous parallel linear algebra library, e.g. SLATE (<https://icl.utk.edu/slate/>).
- Heterogeneous architectures also have a Goldilocks problem: local data sizes need to fill GPU memory to maximize concurrency but not exceed it to minimize data movement into and out of the GPU.



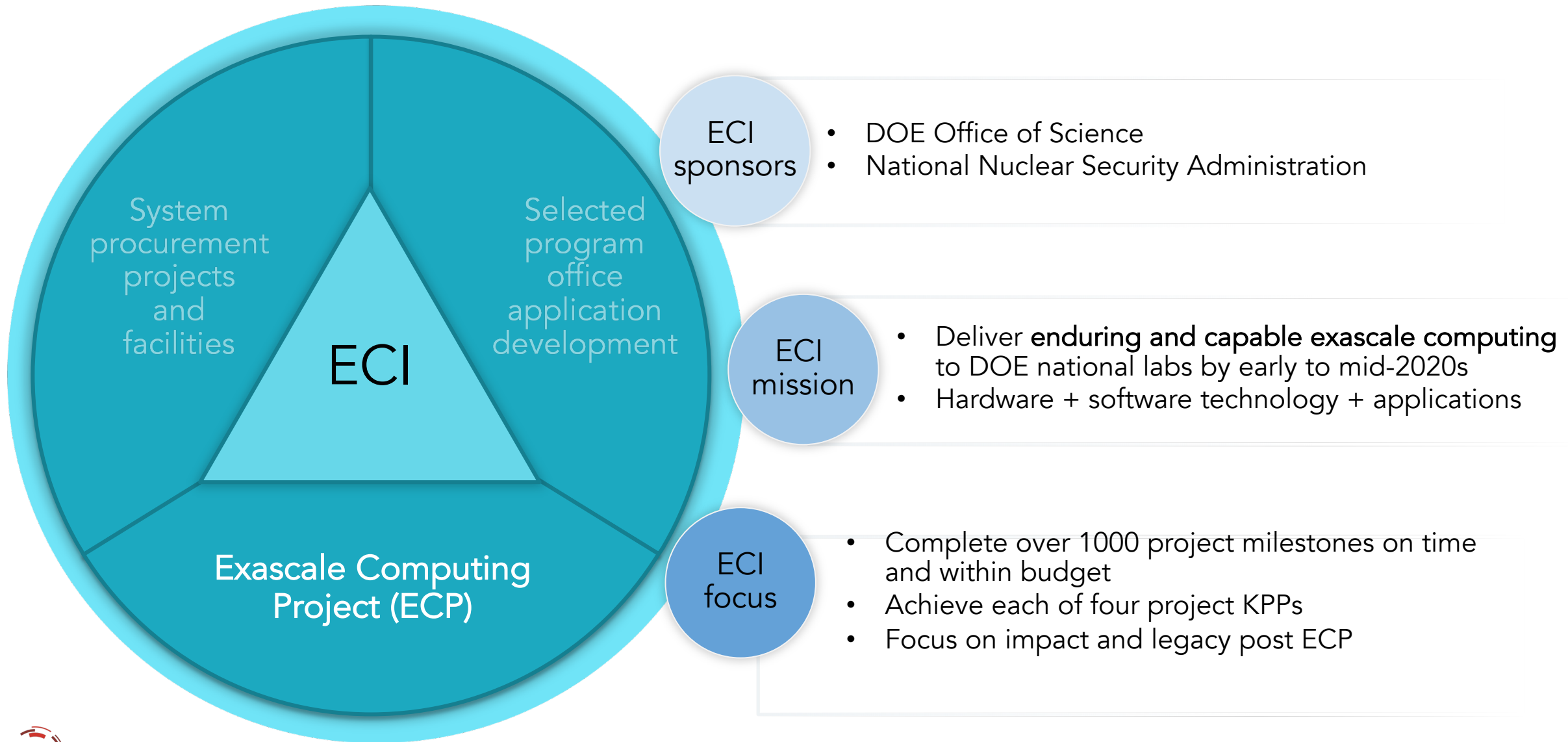
Performance on current and next-gen HPC architectures requires effective use of accelerators



Getting performance on-node is the real challenge

- We used to think of scaling as running $O(100k)$ – $O(1M)$ MPI ranks
- Starting in 2016 (Summit) the FLOPS per node has risen dramatically (48 TF)
- This focuses effort on “scaling in” instead of “scaling out”
- Bottom line: we need to do more work per node on fewer MPI ranks.
- Using the GPUs well is critical!

DOE Exascale Computing Initiative (ECI)



The Exascale Computing Project

7
Years
\$1.8B

- A seven-year, \$1.8B R&D effort that launched in 2016

6
Core DOE
Labs

- Argonne
- Lawrence Berkeley
- Lawrence Livermore
- Oak Ridge
- Sandia
- Los Alamos

- Staff from most of the 17 DOE national laboratories take part in the project
- 6 HPC vendors participated in Path Forward supporting R&D

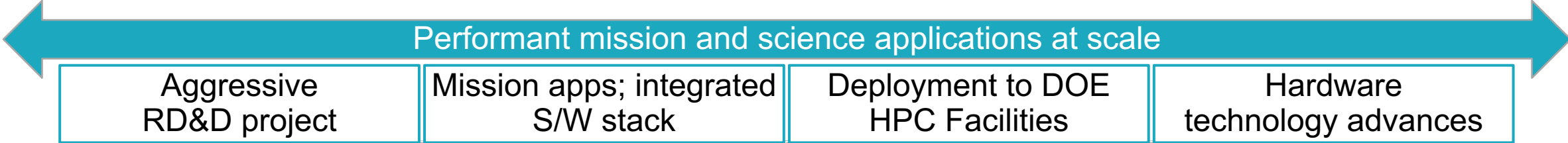
3
Technical
Focus
Areas

- Hardware and Integration
- Software Technology
- **Application Development**

81
R&D Teams
1000
Researchers

- 81 research teams, roughly 10 researchers per team

ECP's holistic approach uses co-design and integration to achieve exascale computing



Application Development (AD)

Develop and enhance the predictive capability of applications critical to DOE

24 applications

National security, energy, Earth systems, economic security, materials, data

6 co-design centers

ML, graph analytics, mesh refinement, PDE discretization, particles, online data analytics



Andrew Siegel, AD Director
Erik Draeger, AD Deputy Director

Software Technology (ST)

Deliver expanded and vertically integrated software stack to achieve full potential of exascale computing

70 unique software products spanning programming models and runtimes, math libraries, data and visualization, development tools



Mike Heroux, ST Director
Lois Curfman McInnes, ST Deputy Director

Hardware and Integration (HI)

Integrated delivery of ECP products on targeted systems at leading DOE HPC facilities

6 US HPC vendors

focused on exascale node and system design; application integration and software deployment to Facilities



Katie Antypas, HI Director
Susan Coghlan, HI Deputy Director

Science and beyond: Applications and discovery in ECP

National security

Energy security

Economic security

Scientific discovery

Earth systems

Health care

24 applications and 6 co-design projects

- Including **62 separate codes**
- Representing over **10 million lines of code**
- Many supporting large user communities
- Covering broad range of mission critical S&E domains
- Mostly all MPI or MPI+OpenMP on CPUs at beginning of ECP
- Each project defines a domain-specific challenge problem for final benchmark
- Applications are evaluated in one of two categories
- Performance – achieve a **50x** performance increase
- Capability – utilize new architectures for expanded S&E

Cosmological probe of the standard model of particle physics

Validate fundamental laws of nature

Plasma wakefield accelerator design

Light source-enabled analysis of protein structure and design

Find, predict, and synthesize materials and properties

Control magnetically confined plasmas

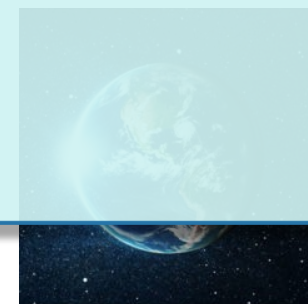
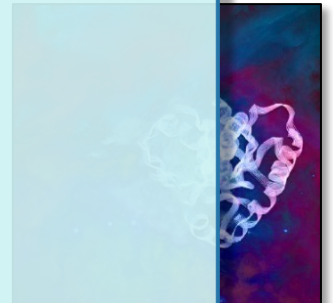
Demystify origin of chemical elements

Accurate regional impact assessments in **Earth system models**

Stress-resistant crop analysis and catalytic conversion of **biomass-derived alcohols**

Genomics for analysis of biochemical cycles, climate change, environmental remediation

Accelerate and translate **cancer research** (partnership with NIH)



Biofuel catalyst design

Four key ingredients of an ECP Application Development Project



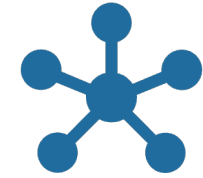
Science goal



Algorithmic
innovation



Porting



Integration

Four key ingredients of an ECP Application Development Project



Science goal



Algorithmic
innovation



Porting



Integration

Algorithmic innovation goes beyond simply porting code

"The downside of ... benchmarks is that innovation is chiefly limited to the architecture and compiler. Better data structures, algorithms, programming languages, ...cannot be used, since that would give a misleading result. The system could win because of, say, the algorithm, and not because of the hardware or the compiler. While these guidelines are understandable when the foundations of computing are relatively stable, as they were in the 1990s and the first half of this decade, they are undesirable during a programming revolution. For this revolution to succeed, we need to encourage innovation at all levels."

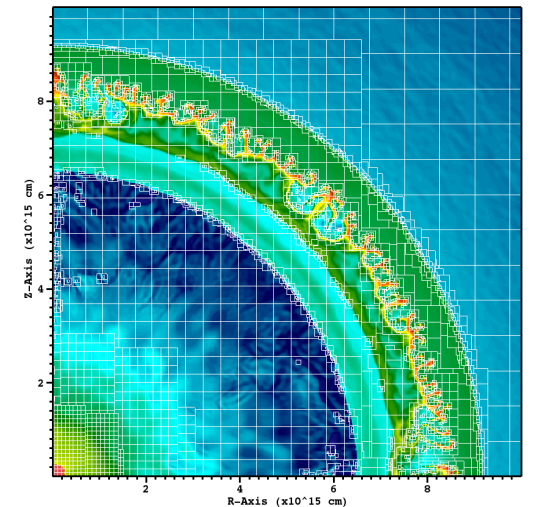
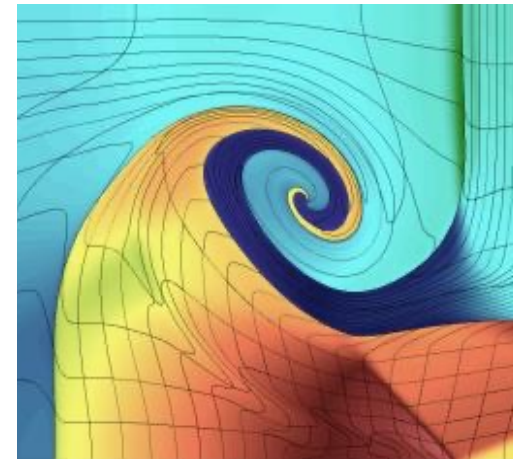
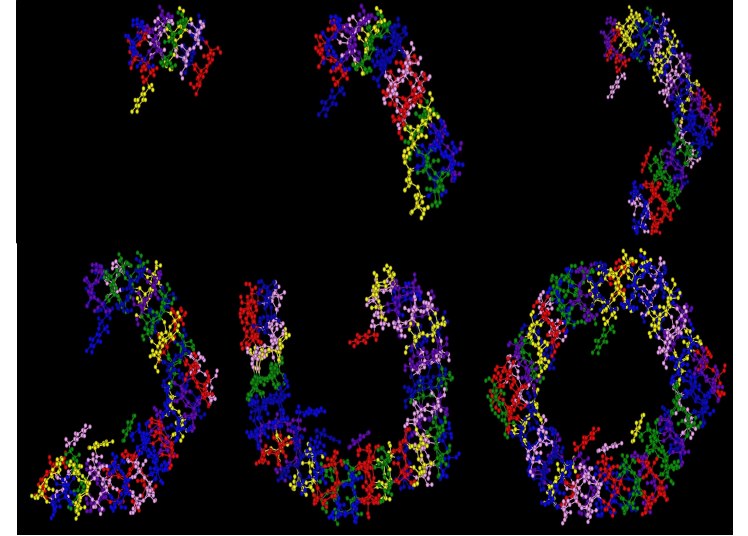
-Hennessy and Patterson, Computer Architecture, A Quantitative Approach

GPUs do best for codes given ...

- ✓ massive fine-grained parallelism
- ✓ concentrated performance bottlenecks
- ✓ weak scaling problems
- ✓ high arithmetic intensity and/or low data movement
- ✓ minimal branching
- ✓ high FLOP to byte (of storage) ratio
- ✓ use of specialized instructions

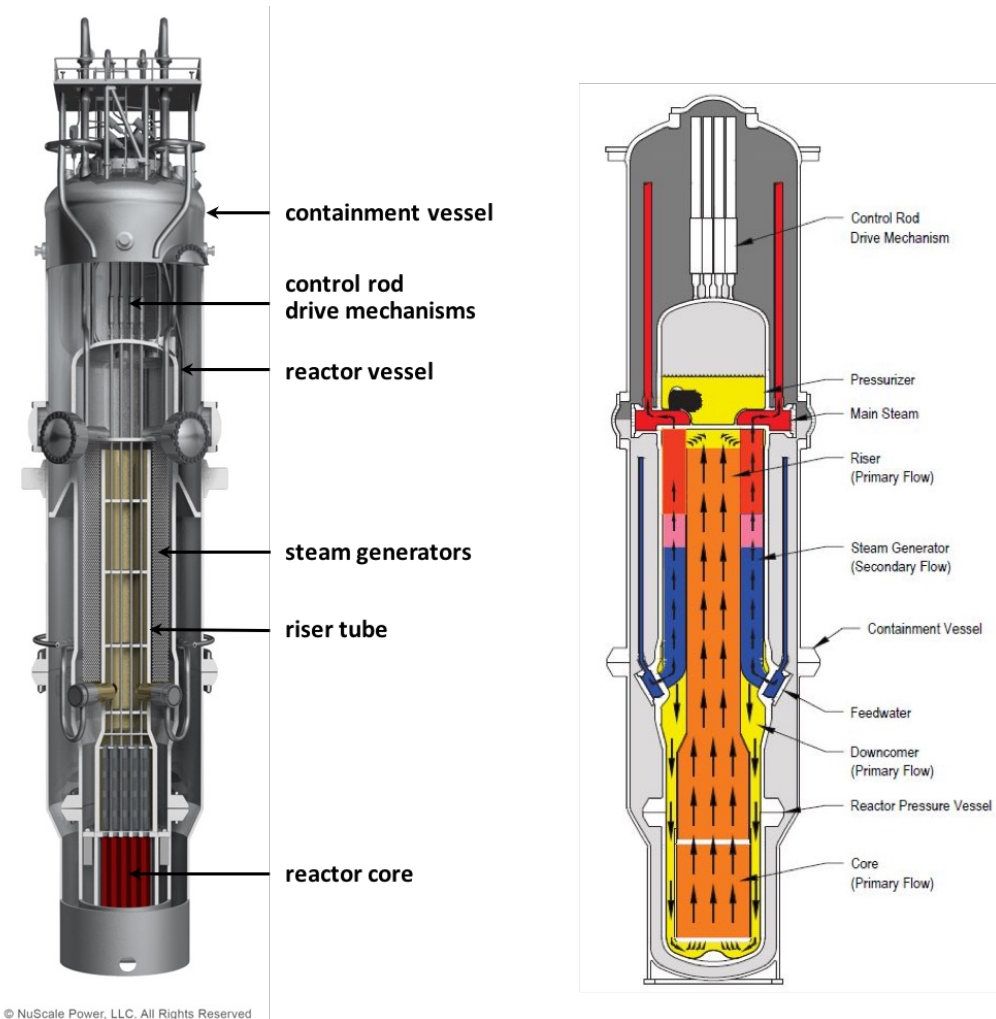
Algorithmic innovation: domain-driven adaptations critical for making efficient use of exascale systems

- Inherent strong scaling challenges on GPU-based systems →
 - Ensembles vs. time averaging
 - Fluid dynamics, seismology, molecular dynamics, time-stepping
- Increased dimensions of (fine-grained) parallelism to feed GPUs
 - Ray tracing, Markov Chain Monte Carlo, fragmentation methods
- Localized physics models to maximize "free flops"
 - MMF, electron subcycling, enhanced subgrid models, high-order discretizations
- Alternatives to sparse linear systems
 - Higher order methods, Monte Carlo
- Reduced branching
 - Event-based models



Example: modeling and simulation of small modular reactors

- ExaSMR is a coupled multiphysics ECP application to perform “virtual experiment” simulations of small modular nuclear reactor designs.
- Small modular nuclear reactors present significant simulation challenges
 - Small size invalidates existing low-order models
 - Natural circulation flow requires high-fidelity fluid flow simulation
- Two primary methods:
 - Monte Carlo neutronics
 - CFD with turbulence models



Reproduced with permission

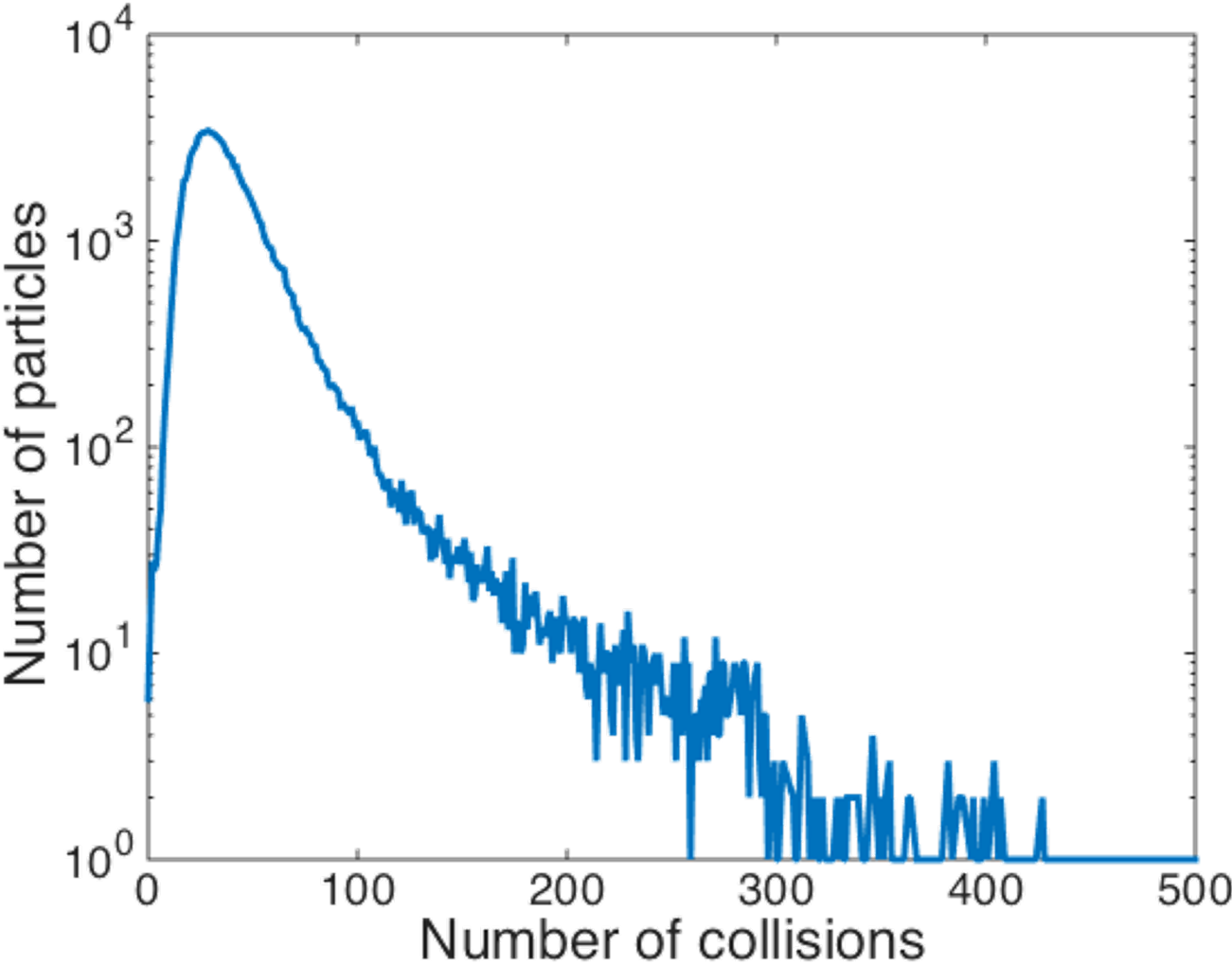
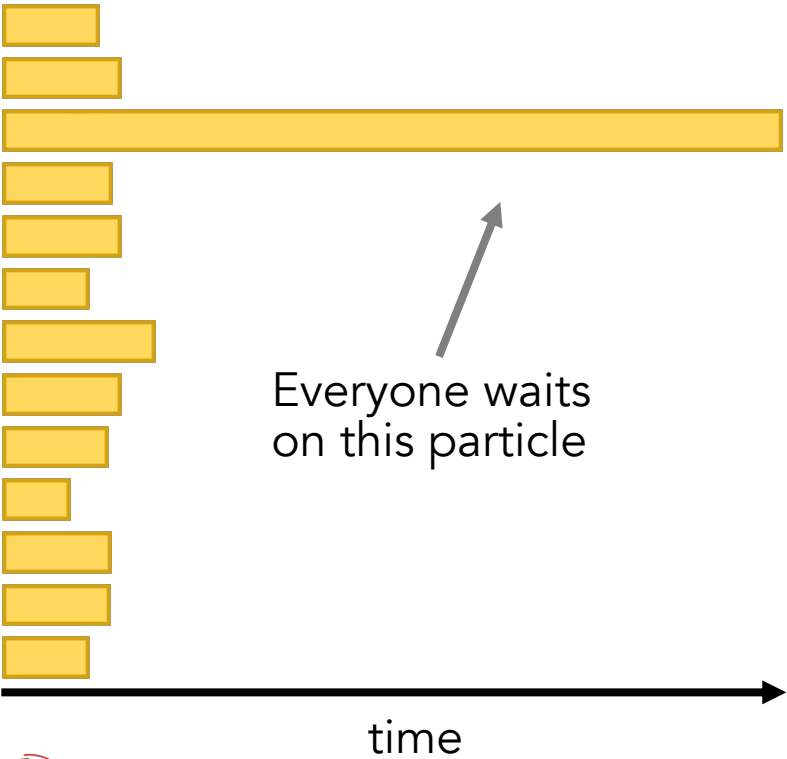
Monte Carlo has been used in particle simulations since the Manhattan Project

- Markov Chain Monte Carlo used by Stanislaw Ulam at Los Alamos for neutron transport calculations in 1946.
 - Very efficient algorithm to evolve particle ensembles across phase space.
 - Continued to be popular as we moved to distributed memory systems. Parallelizes well across particles, even better across samples (“embarrassingly parallel”).
- Unfortunately, Monte Carlo codes written for distributed memory systems do not do well on GPUs!

```
for (i=0, nLocal)
{
  [stuff]
  if (collisionProb > random())
  {
    [stuff]
  }
  else
  {
    [stuff]
    if (absorptionProb > random())
    {
      [stuff]
    }
    else
    {
      [stuff]
    }
  }
  ...
}
}
```

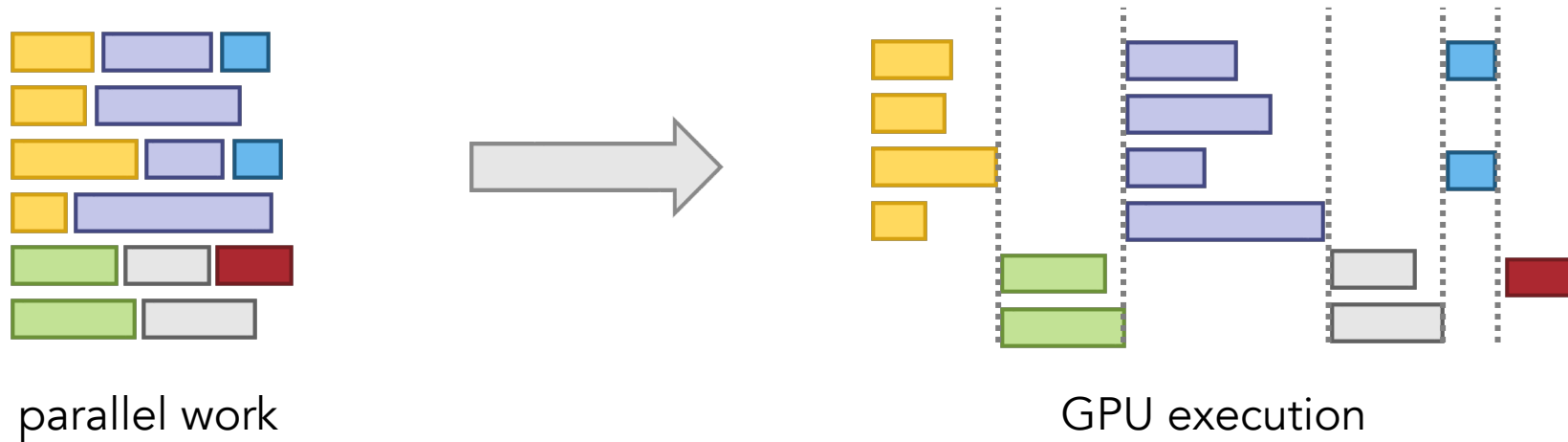
Neutron transport: random particle statistics poorly suited to GPUs

- Stochastic history-based algorithm follows particles from birth to death.
- Most particles are short-lived, a few are not.



Branching code is highly undesirable on SIMT architectures (GPUs)

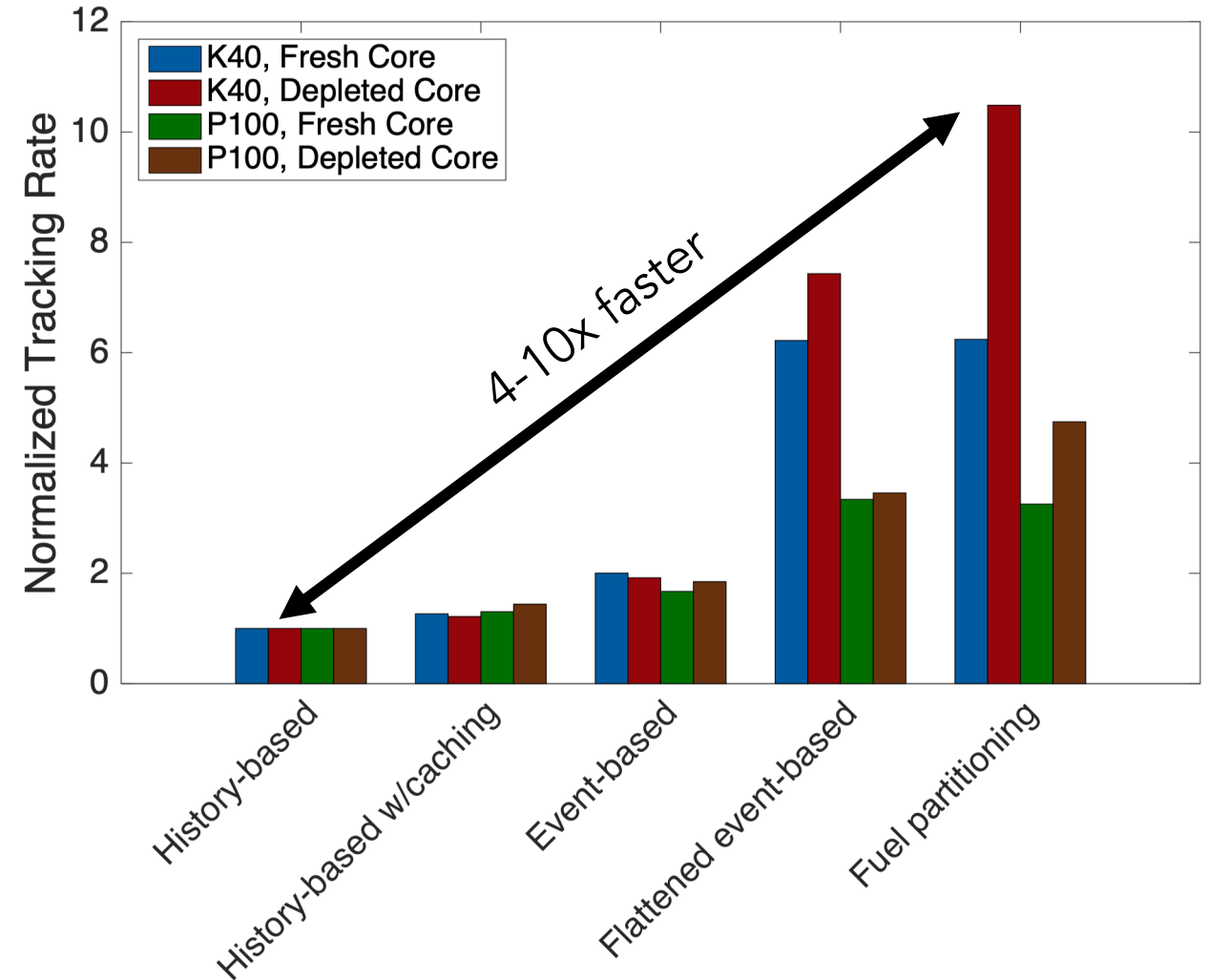
Even when each particle has roughly the same amount of work, **thread divergence** is a big problem when random sampling sends them down different code paths



Need to rethink code execution based on the target hardware. For example, parallelizing over **events** (i.e. common code paths) rather than particles.

New event-based algorithm gave ExaSMR significant speedup

- Parallelizing over events is a much better match for a SIMT architecture than parallelizing over particles.
- Further improvements gained by identifying parts of the system that have significantly different behavior and separating them out.
- Smaller, focused kernels allow for better occupancy, i.e. more efficient use of the hardware



Four key ingredients of an ECP Application Development Project



Science goal



Algorithmic
innovation



Porting



Integration

Porting must be done with hardware in mind

Map algorithm to GPUs

- Rewrite, profile, and optimize
 - Generally preserve the exact answer
- Data Layout for memory coalescing
- Loop ordering
- Kernel flattening
- Increased locality
- Recomputing vs. storing
- Reduced branching
- Eliminating copies

Map calculation to GPUs

- Reduced communication
- Reduced synchronization
- Increased parallelism
- Reduced precision
- Optimized linear algebra

Identify opportunities for improvement

- Mathematical representation
- “On the fly” recomputing vs. lookup tables
- Prioritization of new physical models
- Alternate discretizations (high AI)
- Localized subgrid models
- Sparse → dense systems
- Defining weak scaling target
- Initial condition from ROM

Hardware has significant impact on all aspects of simulation strategy

Choosing the right programming model is all about balancing trade-offs

GPU-specific kernels

- Isolate the computationally-intensive parts of the code into CUDA/HIP/SYCL kernels.
- Refactoring the code to work well with the GPU is the majority of effort.

Loop pragma models

- Offload loops to GPU with OpenMP or OpenACC.
- Most common portability strategy for Fortran codes.

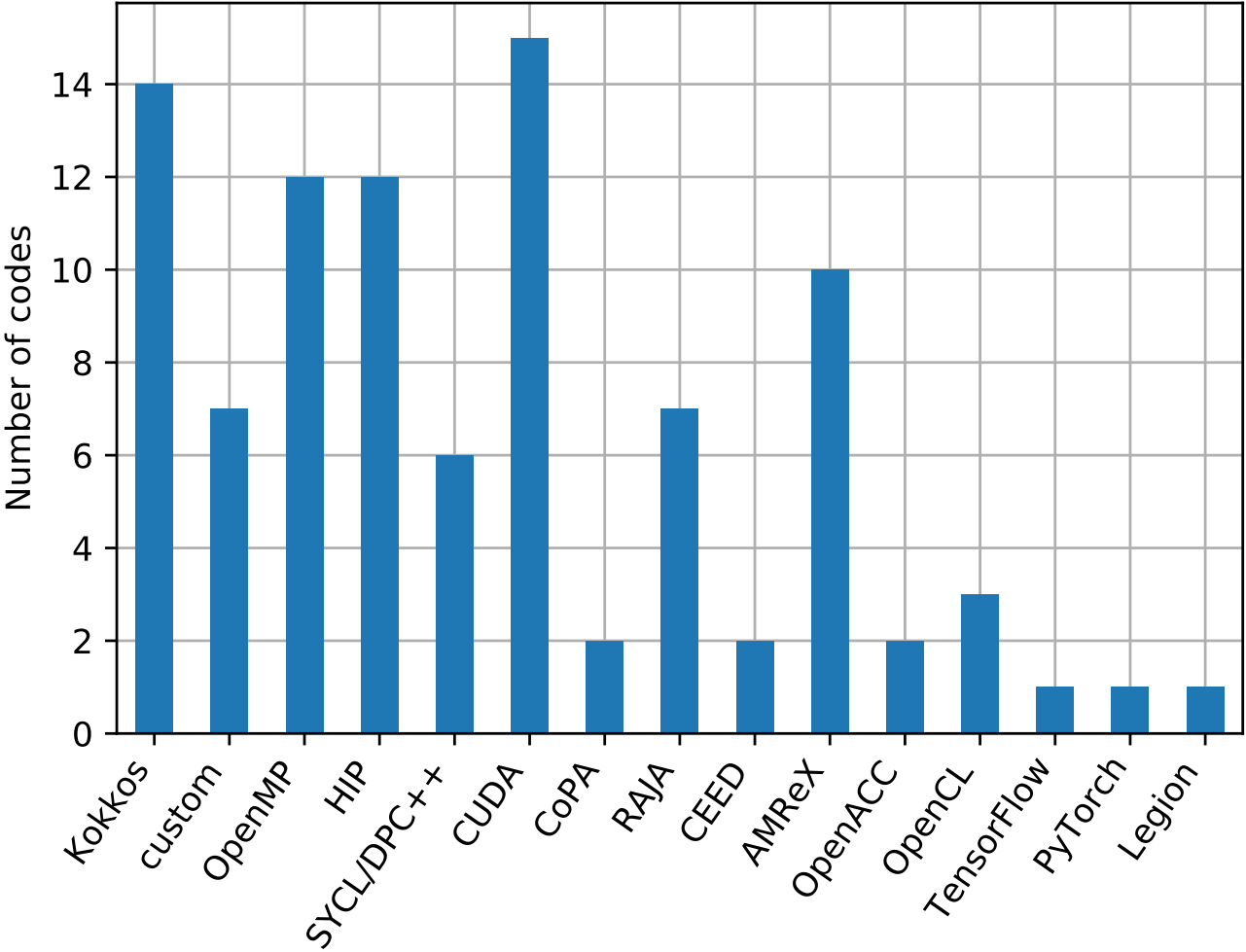
C++ abstractions

- Fully abstract loop execution and data management using advanced C++ features.
- Kokkos and RAJA developed by NNSA in response to increasing hardware diversity.

Co-design frameworks

- Design application with a specific motif to use common software components
- Depend on co-design code (e.g. CEED, AMReX) to implement key functions on GPU.

Programming models used in ECP applications

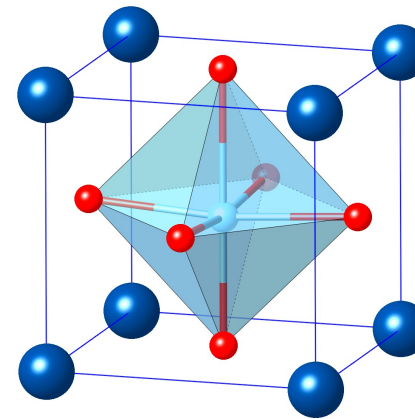
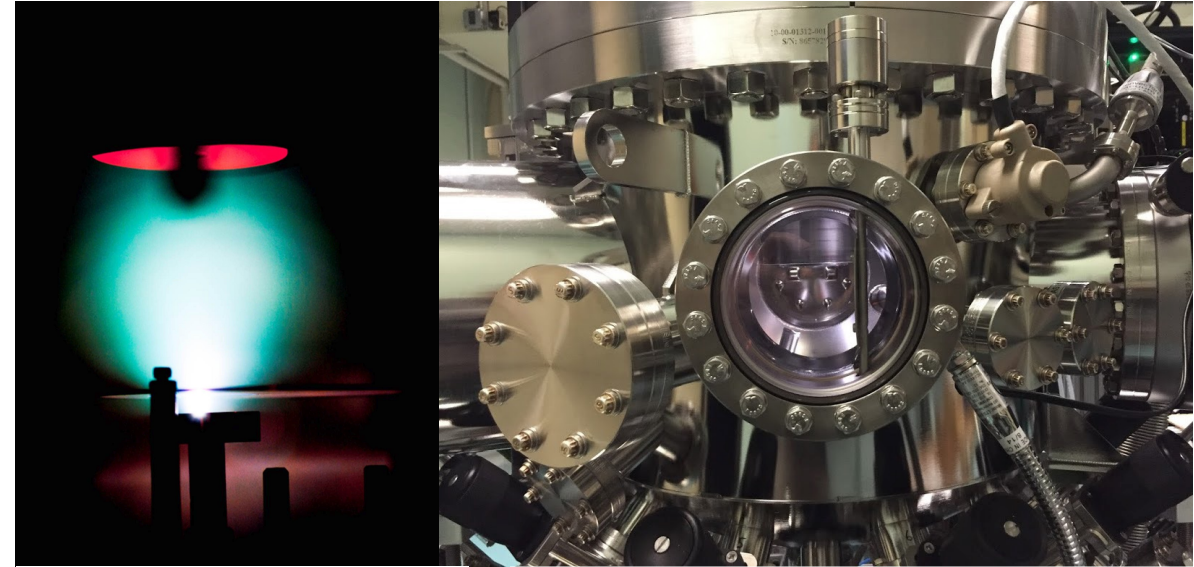


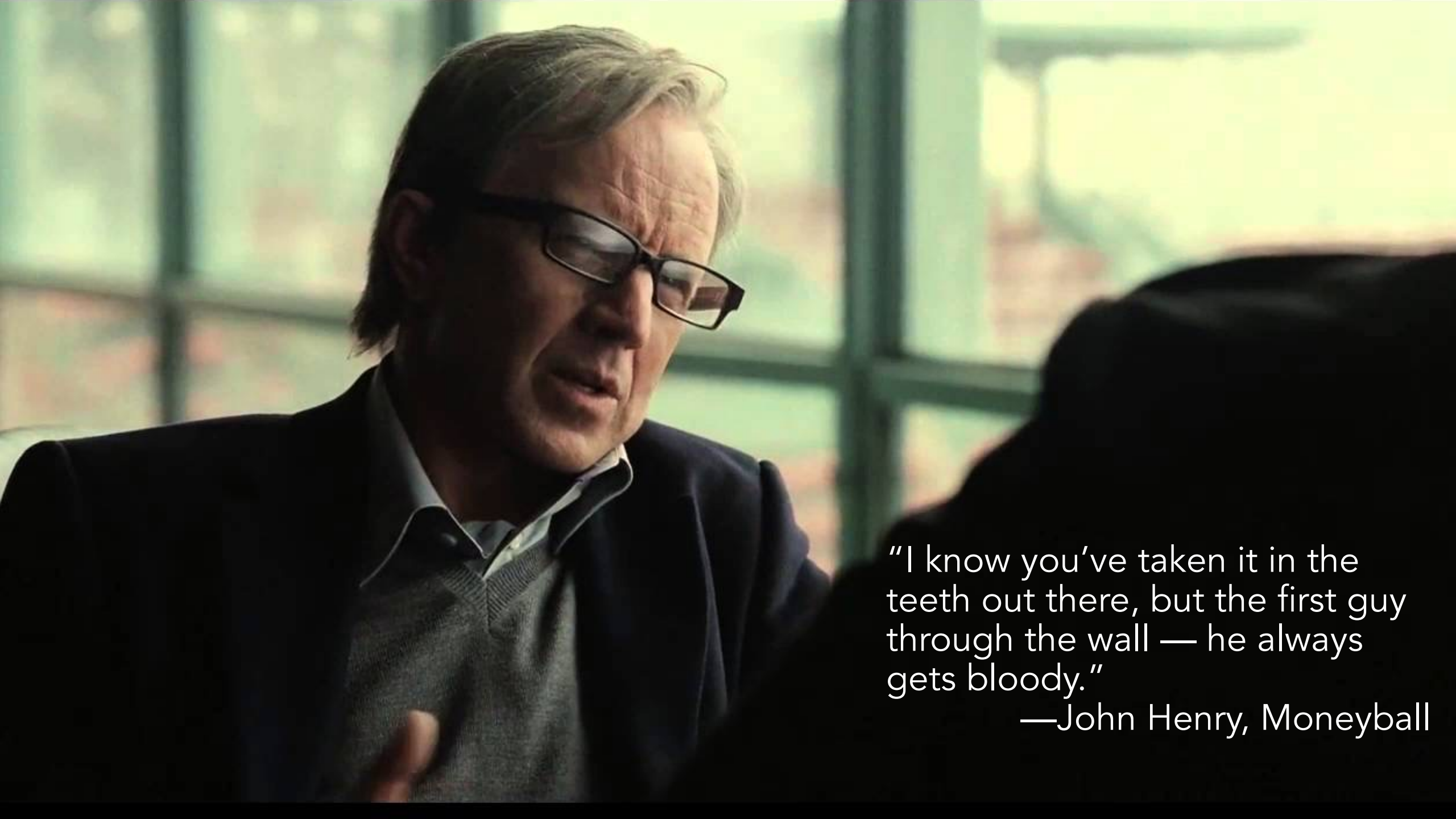
Platform portability provided by co-design projects (CoPA, CEED, AMReX)	33%
Native (CUDA/HIP/SYCL) or custom implementations	33%
ST programming models (Kokkos, RAJA, Legion)	18%
Directive-based programming models: (OpenMP, OpenACC)	16%

- Use of co-design/ST technologies provides significant benefit. Fine-scale architectural details provided by co-design technologies
- Large percent of custom implementations reflects difficulty of universal platform-portable programming models that span diverse apps

Example: Quantum Monte Carlo for Materials

- To predict, understand, and design next generation materials requires reliable, non-empirical, atomistic quantum mechanics-based methods.
- ECP application QMCPACK implements multiple Quantum Monte Carlo (QMC) algorithms to achieve this. Primary focus for ECP is on the real-space diffusion Monte Carlo (DMC) and orbital space auxiliary field QMC (AFQMC) algorithms to enable cross-validation.
- OpenMP was selected as the GPU programming model to maximize future portability.



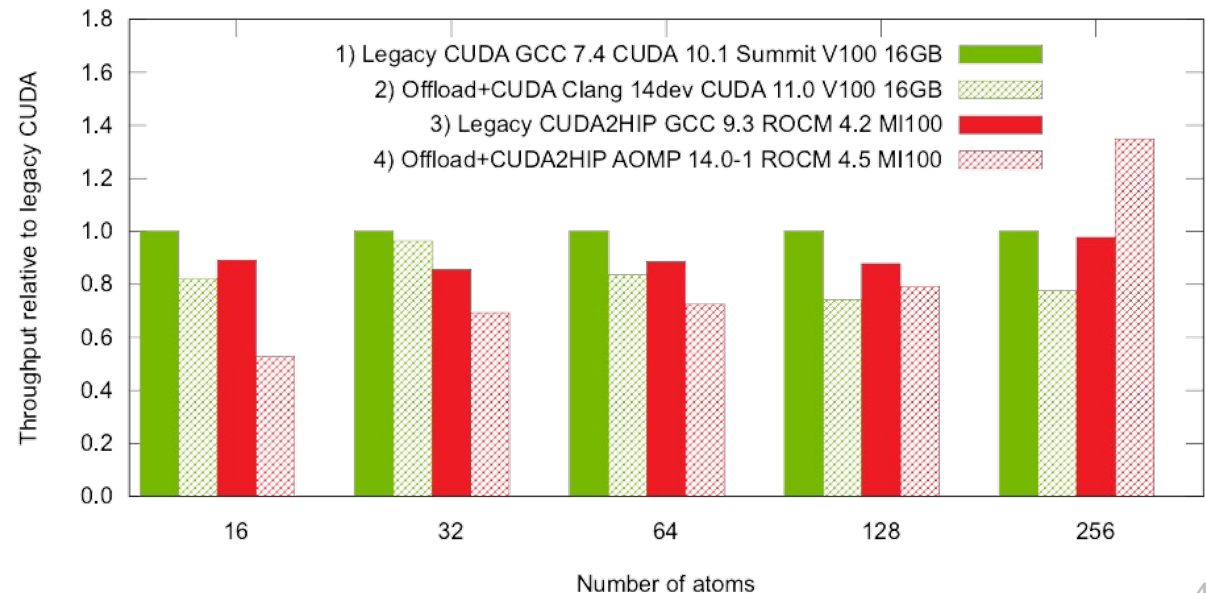
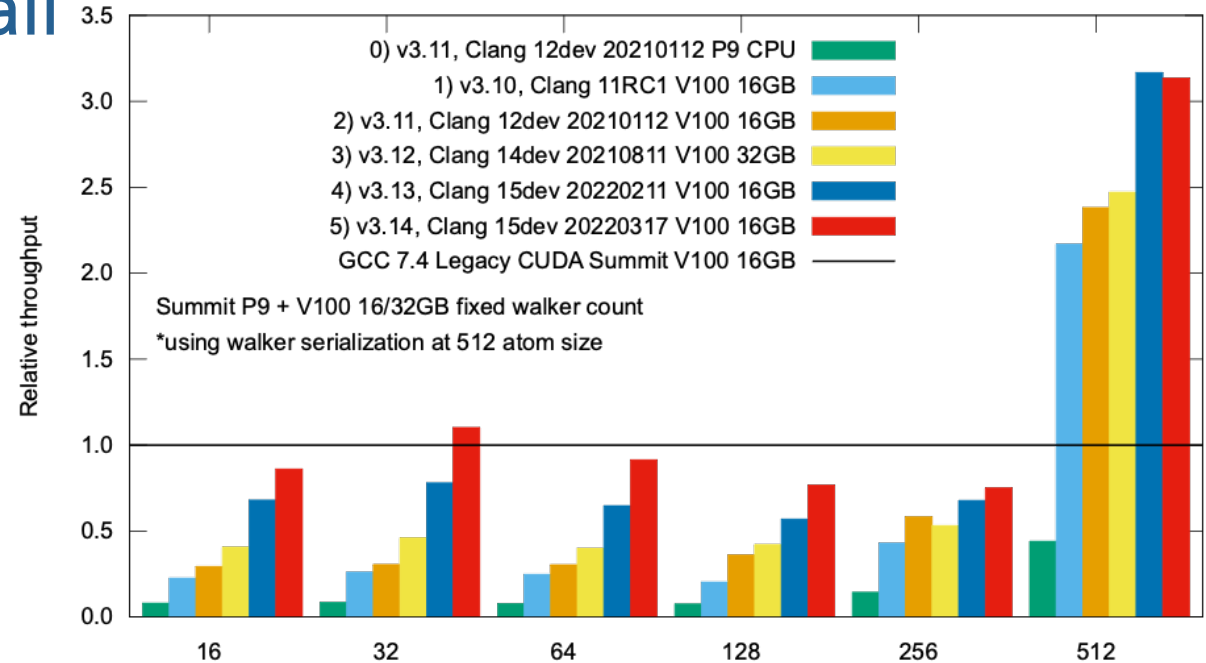


“I know you’ve taken it in the teeth out there, but the first guy through the wall — he always gets bloody.”

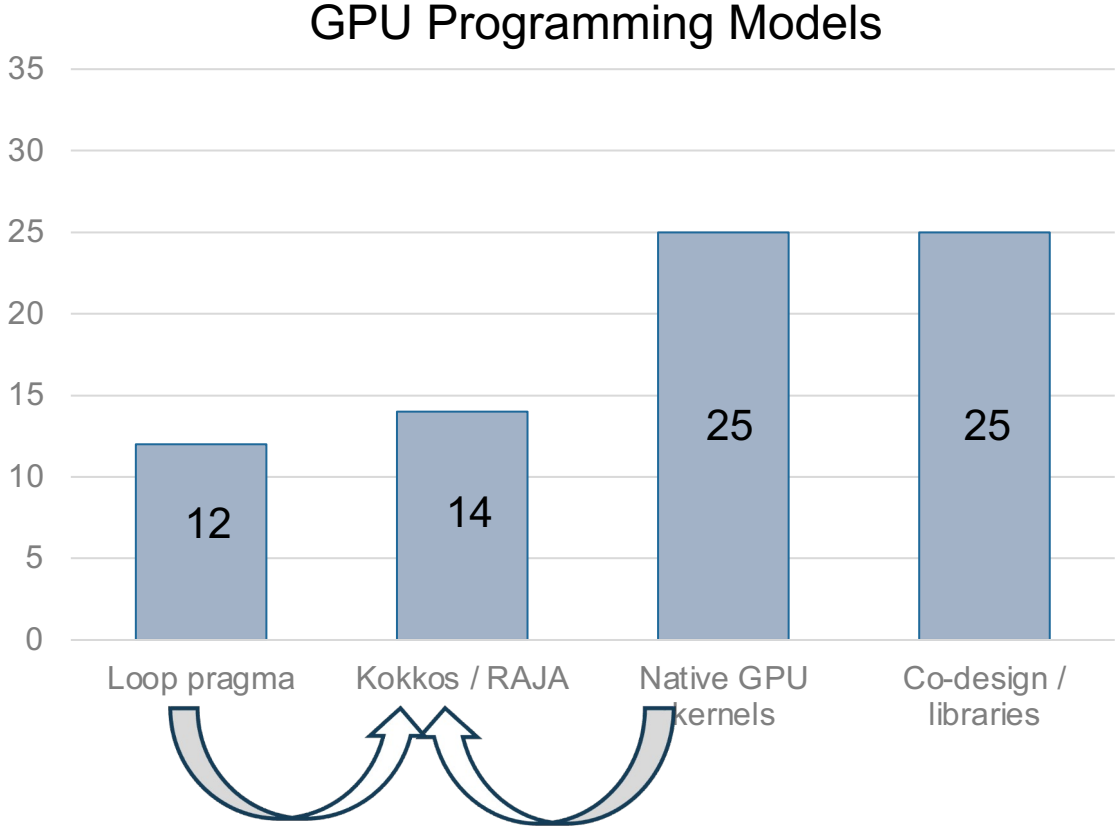
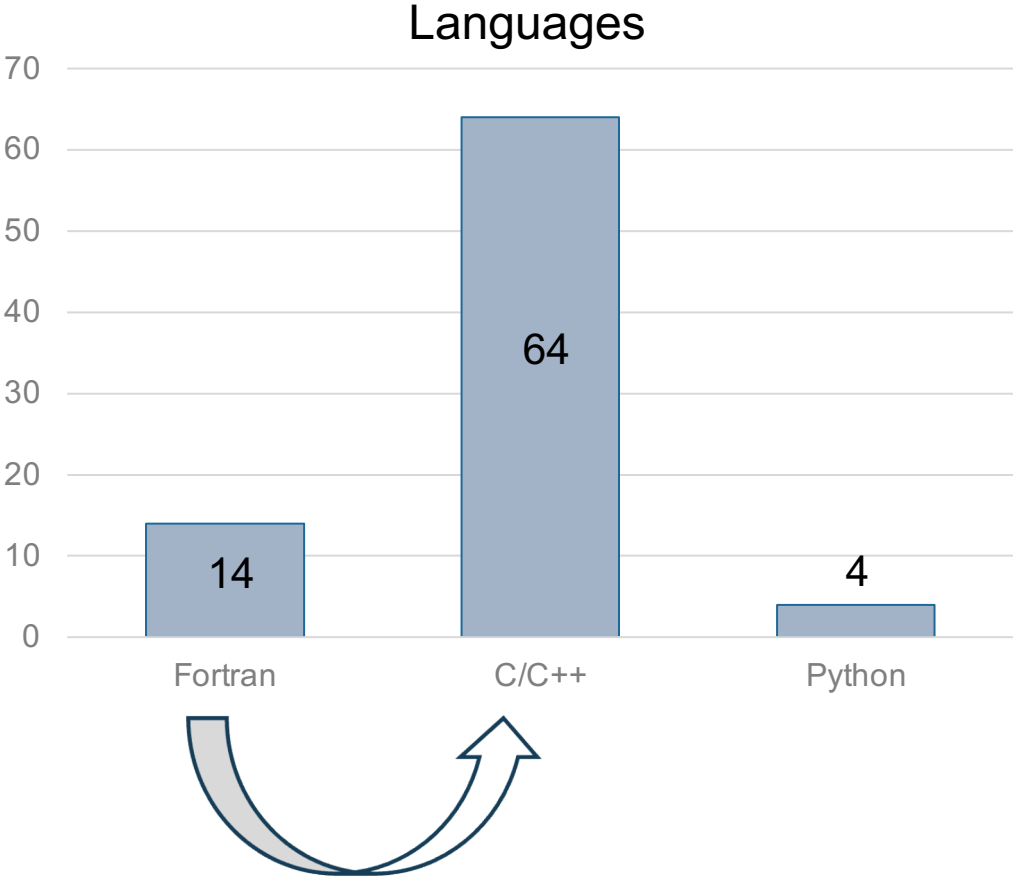
—John Henry, Moneyball

QMCPACK was first through the wall

- QMCPACK had a working CUDA implementation of the code that proved invaluable in understanding where OpenMP performance was falling short.
- OpenMP offload runtimes are not yet consistently performant across vendors. Initial OpenMP results were significantly slower than CUDA.
- With careful performance analysis and by working closely with the vendors, the QMCPACK team was able to steadily improve performance of their OpenMP version until it is now on par with CUDA.



Distribution of ECP programming models has changed over time



Programming language/model choices have evolved over course of ECP

Four key ingredients of an ECP Application Development Project



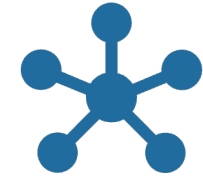
Science goal



Algorithmic innovation

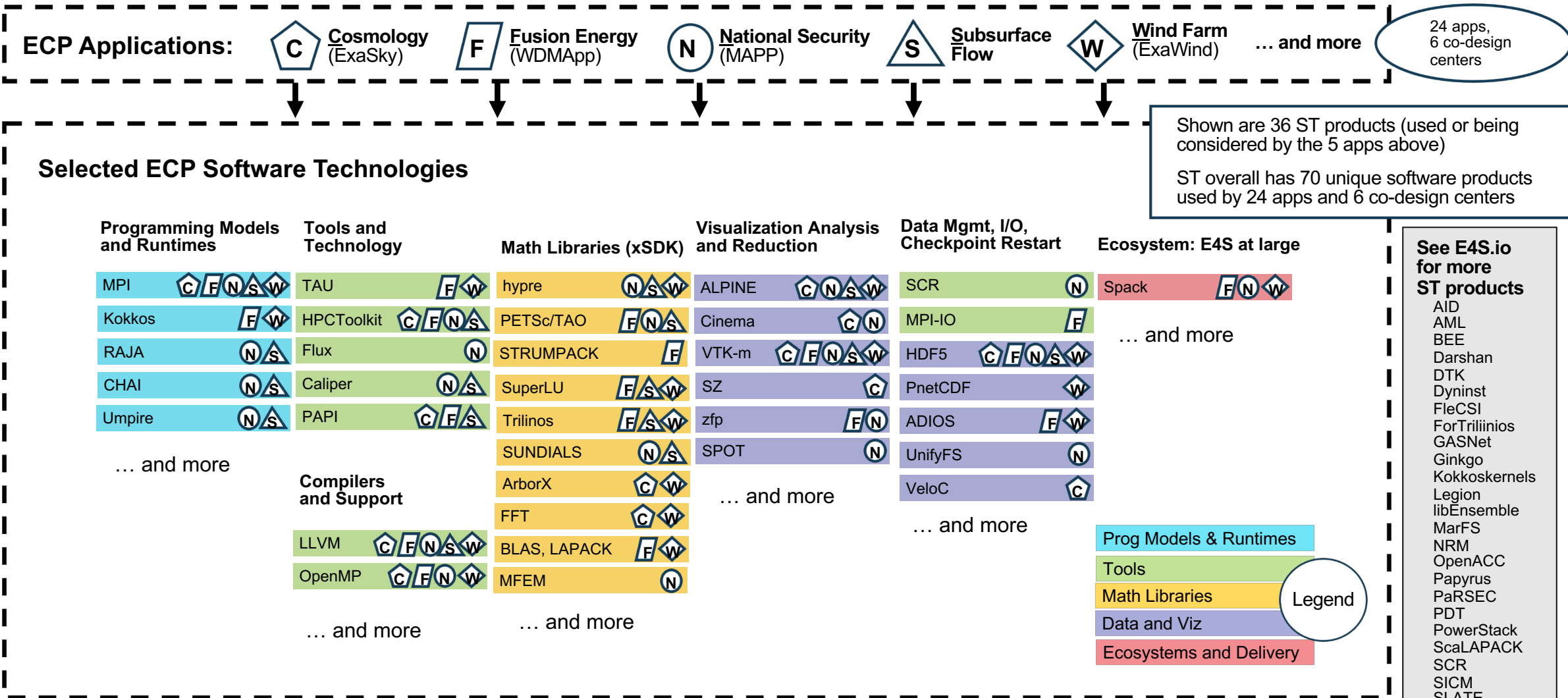


Porting



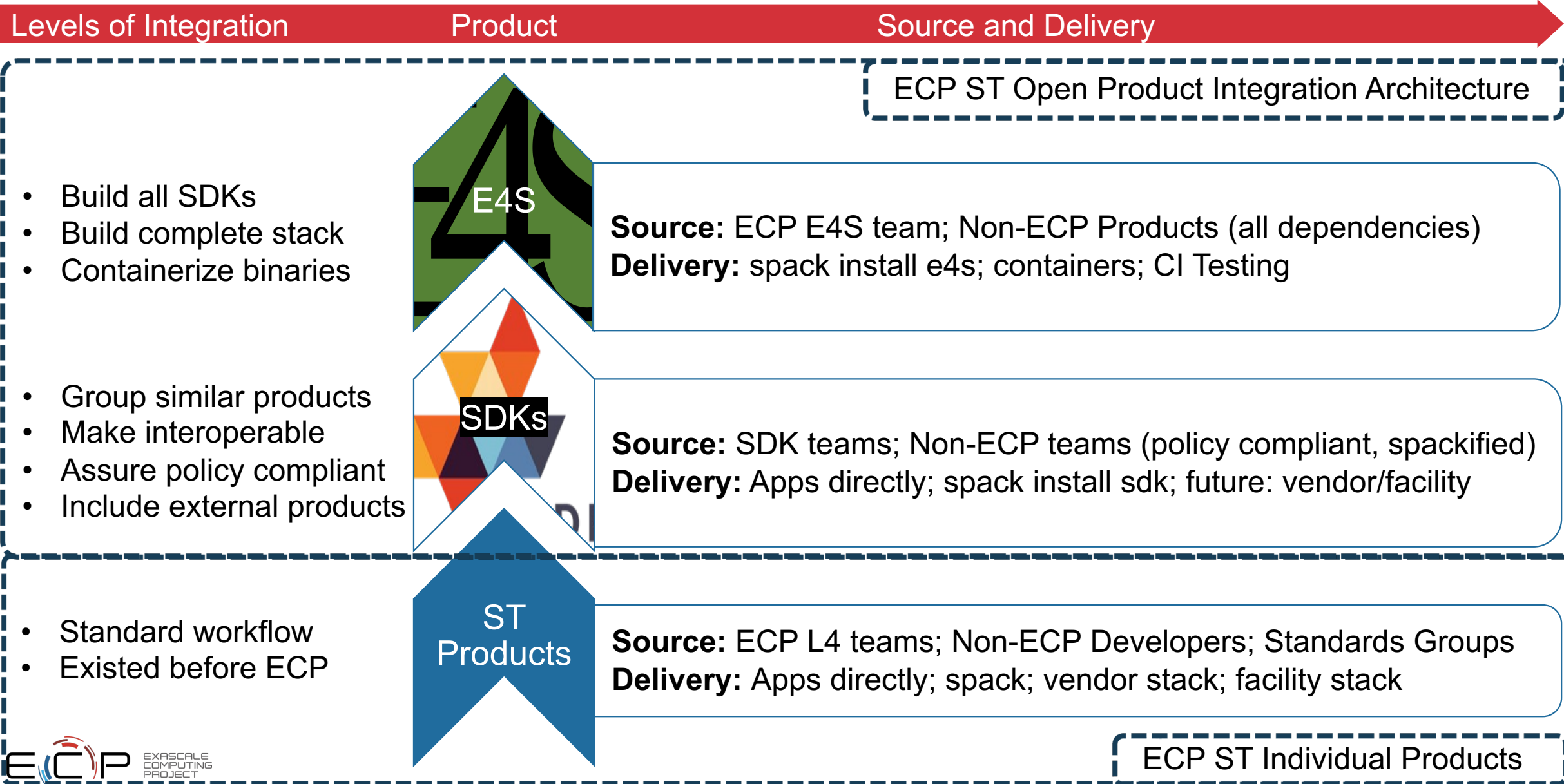
Integration

Integration: ECP applications rely heavily on high quality software tools and libraries




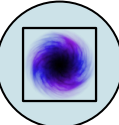







- See E4S.io for more ST products**
- AID
 - AML
 - BEE
 - Darshan
 - DTK
 - Dyninst
 - FileCSI
 - ForTriliinos
 - GASNet
 - Ginkgo
 - Kokkoskernels
 - Legion
 - libEnsemble
 - MarFS
 - NRM
 - OpenACC
 - Papyrus
 - PaRSEC
 - PDT
 - PowerStack
 - ScaLAPACK
 - SCR
 - SICM
 - SLATE
 - SWIG
 - Tasmanian
 - Umap
 - UPC++

ECP is delivering an open, hierarchical software ecosystem



Extreme-scale Scientific Software Stack (E4S)

- E4S: HPC software ecosystem – a curated software portfolio
- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures
- Available from **source, containers, cloud, binary caches**
- Leverages and enhances SDK interoperability thrust
- Not a commercial product – an open resource for all
- Growing functionality: Nov 2021: E4S 21.11 – 91 full release products

	Community Policies Commitment to software quality		DocPortal Single portal to all E4S product info		Portfolio testing Especially leadership platforms
	Curated collection The end of dependency hell		Quarterly releases Release 1.2 – November		Build caches 10X build time improvement
	Turnkey stack A new user experience		https://e4s.io		E4S Strategy Group US agencies, industry, international



<https://spack.io>

Spack lead: Todd Gamblin (LLNL)



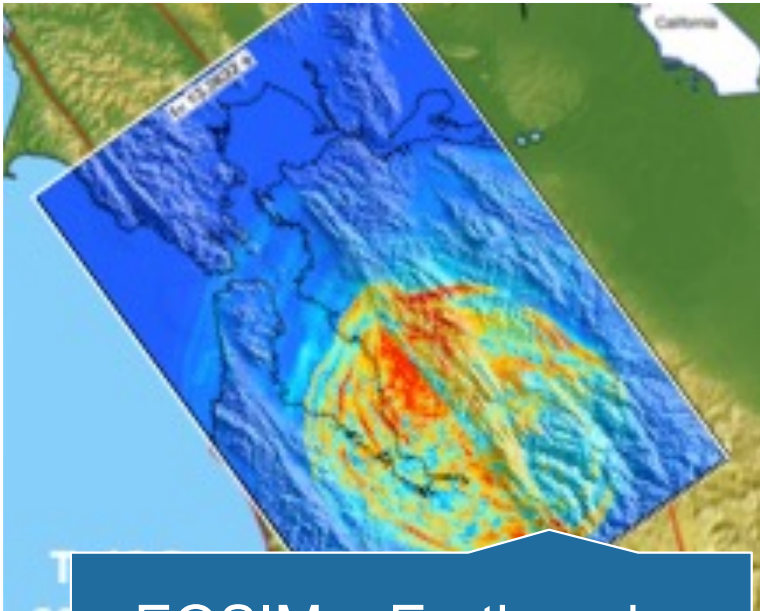
<https://e4s.io>

E4S lead: Sameer Shende (U Oregon)

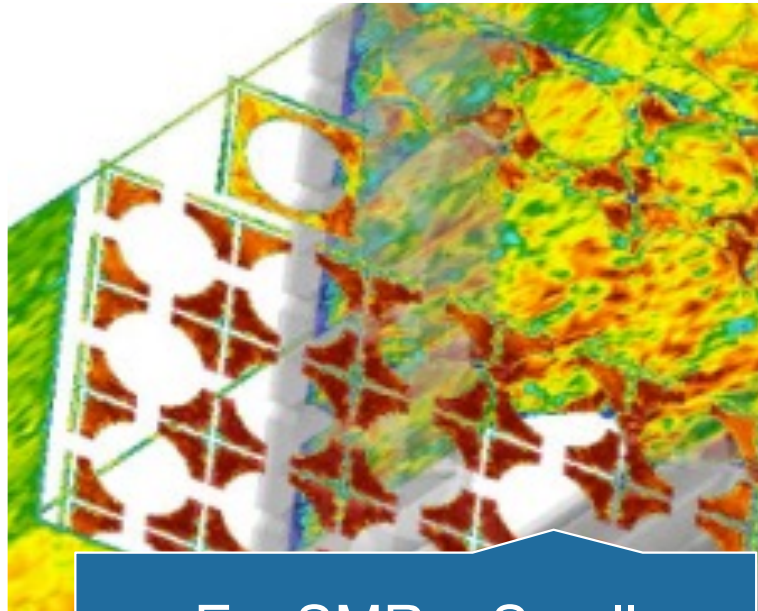


Also includes other products, e.g.,
AI: PyTorch, TensorFlow, Horovod
Co-Design: AMReX, Cabana, MFEM

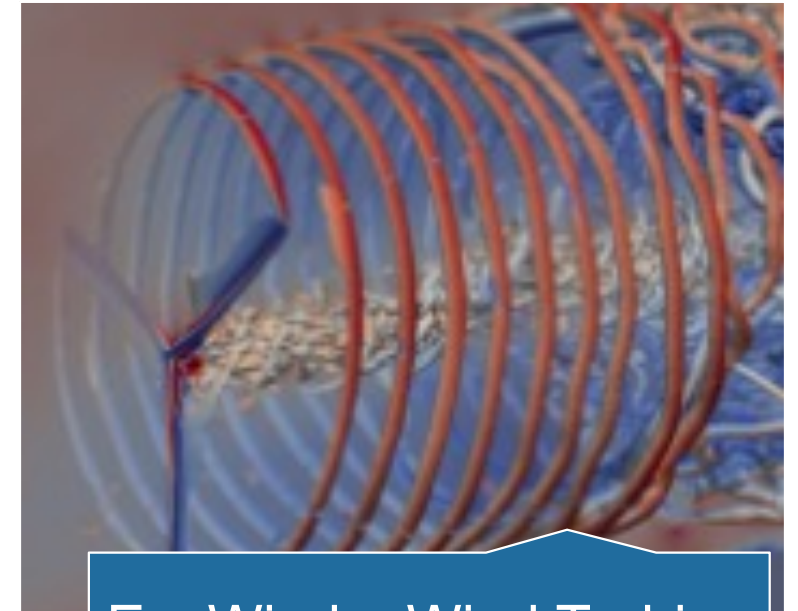
Three application examples to show the kind of progress possible with Exascale computing resources



EQSIM – Earthquake Hazard and Risk



ExaSMR – Small Modular Reactors



ExaWind – Wind Turbine Siting and Operations

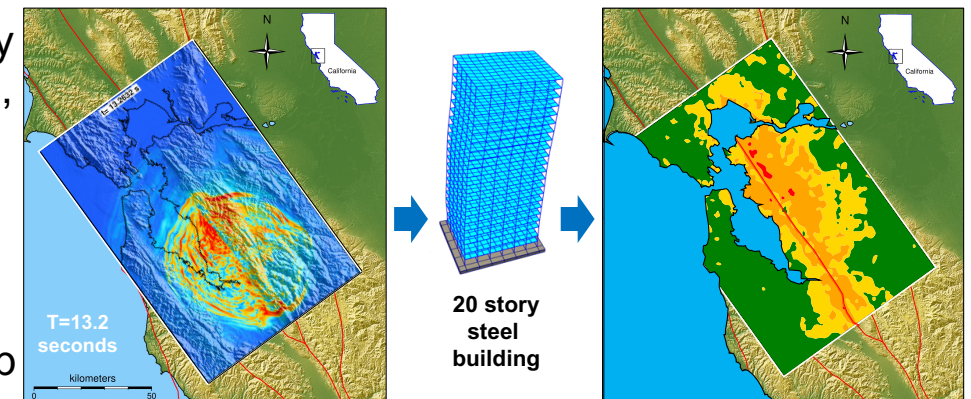
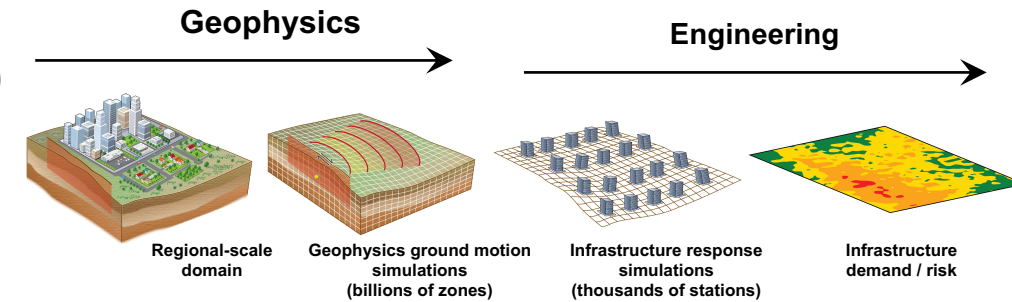
The EQSIM project is developing a frameworks for regional scale earthquake hazard and risk

PI: Dave McCallen, LBNL

- **Objective:** Create a simulation tools that answers questions such as *What is the regional distribution of ground motions and associated infrastructure response?* and *How do complex (realistic) incident seismic waves interact with infrastructure?*

- **ECP accomplishments**

- Algorithmic improvements using curvilinear mesh refinement improved speeds by a factor of 2.85
- Developing a GPU-enabled full waveform inversion algorithm with many algorithmic improvements for separated phase and amplitude matching, gradient smoothing and Hessian-based preconditioning
- Use RAJA for performance portability and ZFP data compression to save sufficient data to maintain adequate precision in stored data
- Infrastructure simulations now include strong coupling with OpenSees soil/building modeling and using in soil-structure interaction models; help gain insight into areas of maximum risk



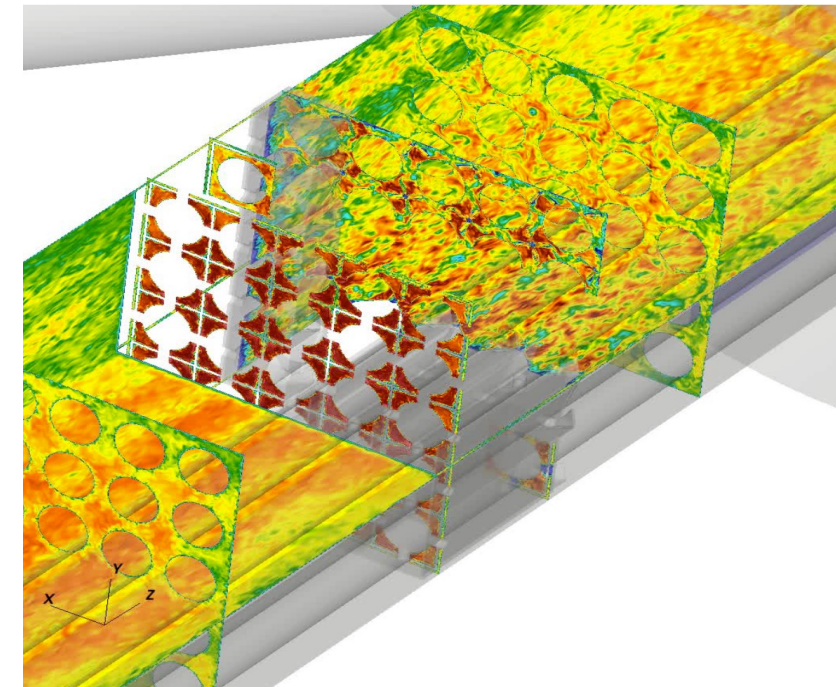
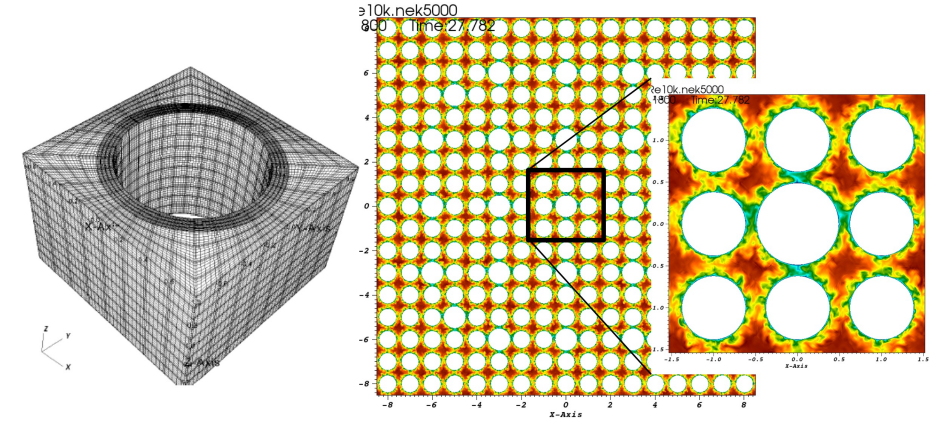
Achieved a 1000X improvement in computational performance compared to all previous San Francisco Bay Area simulations; Simulation of regional-scale ground motions at frequencies of engineering interest (5-10 Hz) now within reach



The ExaSMR project is developing first of a kind simulations of small modular reactors (SMR)

PI: Steve Hamilton, ORNL

- **Objective:** Help DOE meet its goal of an operational SMR by 2025 through advanced modeling and simulation; coupling Monte Carlo neutron simulation with computational fluid dynamics
- **ECP accomplishments**
 - Developed GPU-enabled Monte-Carlo transport codes (Shift and OpenMC), targeting Frontier and Aurora respectively
 - Refactored neutron transport algorithms from particle-based to event-based; dramatically improving performance on GPUs
 - Demonstrated first CFD simulation of a full SMR core
 - Optimized CFD simulations; improving performance by a factor of 5 through improved preconditioners, use of half-precision numerical methods, and GPU-aware gather-scatter kernels
 - Working toward full core coupled physics and isotopic depletion using domain decomposed Monte Carlo solvers



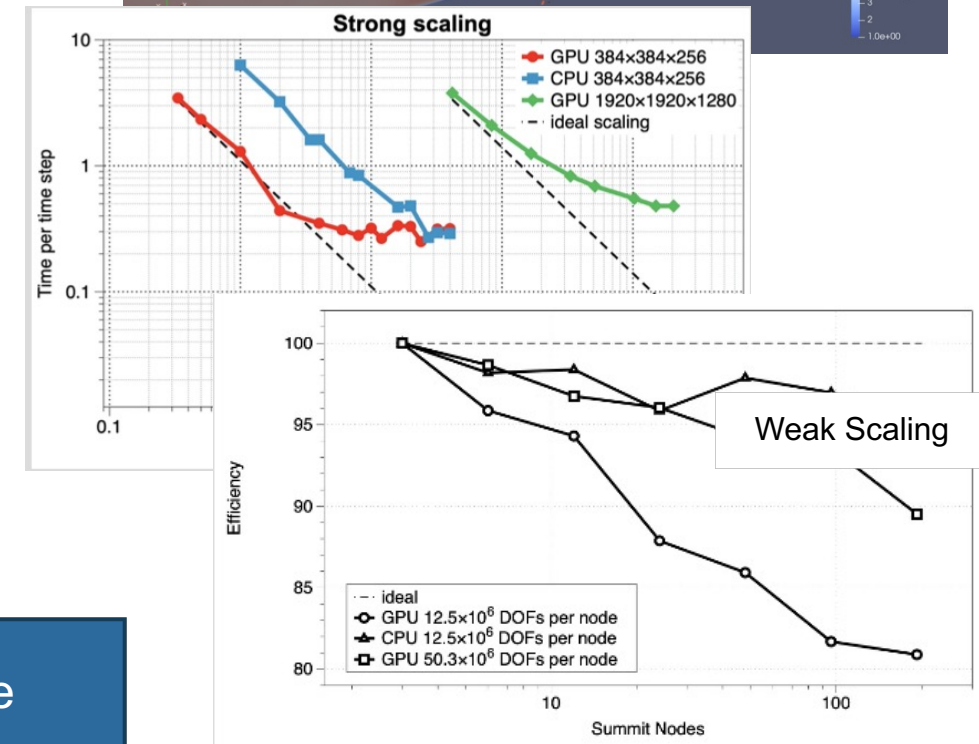
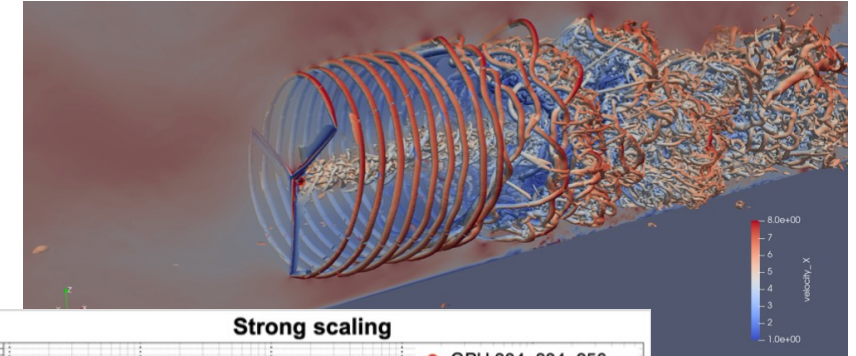
Achieved >70X overall performance improvements in the science workrate for the simulation; will allow full coupled steady-state simulations, modeling quasi-static full cycle depletion, and coupling transient natural circulation reactor start-up

The ExaWind project is developing wind plant simulation capabilities for siting and operations

PI: Mike Sprague, NREL

- **Objective:** Create a predictive physics-based simulation capability that will provide a validated "ground truth" foundation for siting and operational controls of wind plants, and the reliable integration of wind energy into the grid
- **ECP accomplishments**
 - New hybrid Nalu-Wind/AMR-Wind solver strategy: Leveraging the best of structured & unstructured grids
 - Uses a large number of software technologies for performance portability, linear solvers, block structured AMR, package management
 - New hybrid solver enables validation-quality blade-resolved turbine simulations
 - Optimized Nalu-Wind/*hypre* simulations performed on over 4000 Summit GPUs
 - AMR-Wind strong/weak-scaling atmospheric-boundary-layer (ABL) simulations on Summit reach billions of grid points

New capabilities allow detailed fluid structure interactions of the blade with turbulence models; then scaling up to many turbines to capture impact of terrain, atmospheric boundary layer, and inter-blade effects

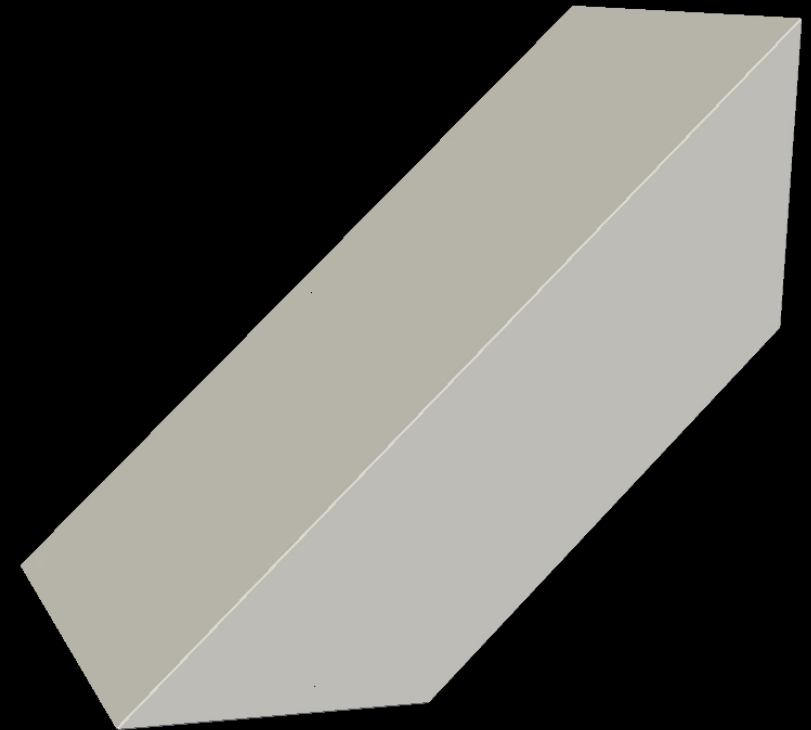


Bonus example: advanced particle accelerator design

- Scientists have proposed new approaches to building smaller, more efficient particle accelerators using plasmas.
- Simulations are critical to aid in the development of accelerator designs.
- Potential applications include improved radiation treatment for cancer.
- WarpX was built on top of AMReX adaptive mesh refinement library.
- 2022 ACM Gordon Bell Prize winner!

WarpX project, PI: Dr. Jean-Luc Vay (LBNL)

Movie: D. Pugmire (ORNL)
From WarpX simulation on
4096 Summit nodes



Final thoughts

- This is an exciting and terrifying time to be doing computational science.
- Those who take the time to understand the hardware they are running on and/or coding for will have a major advantage over those who try to use past practices blindly.
- For computational capabilities, don't reinvent the wheel! Build on the successes of others whenever possible.
- For applied math, re-examine and question everything! Many best practices are based on assumptions from the past that no longer apply. There are many opportunities for innovation.

LLNL Auspices and Disclaimer

Prepared by LLNL under Contract DE-AC52-07NA27344. This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.