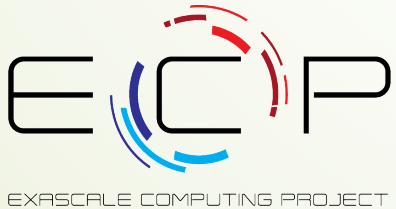
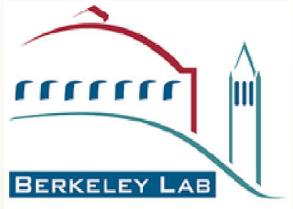
Geant Exascale/GPU Pilot Project

Presented by Liz Sexton-Kennedy
Philippe Canal for the Geant Exascale Pilot team







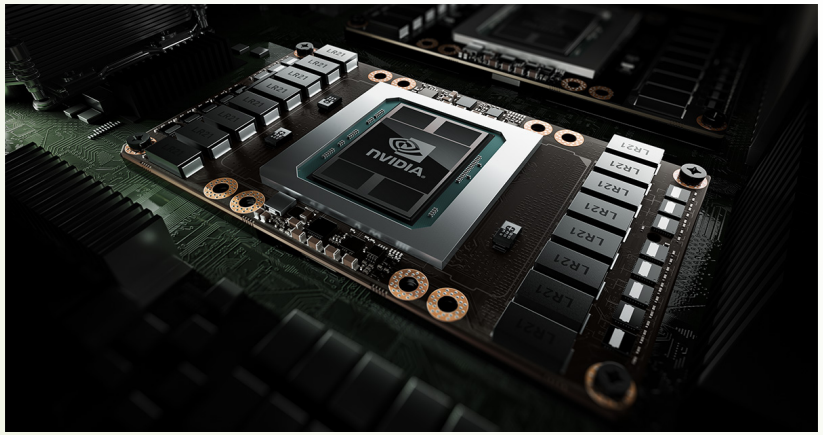
Fermilab

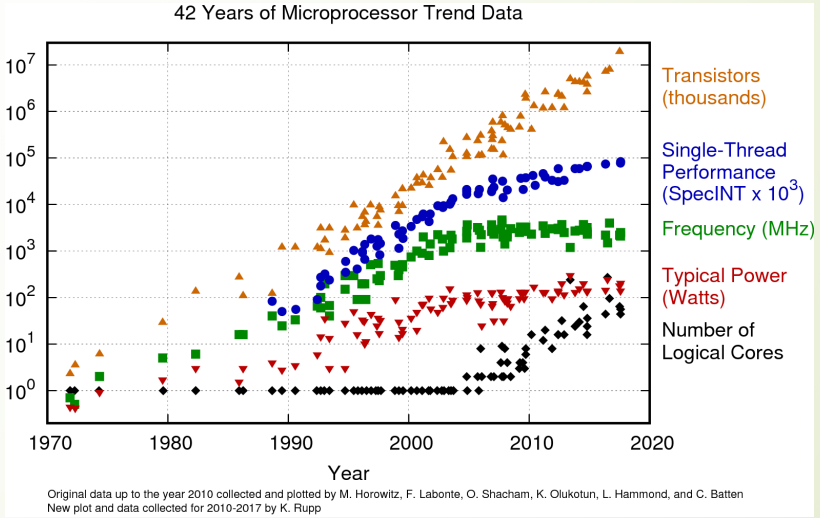
OAK RIDGE National Laboratory

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics

Outline

- Exascale: GPU for the foreseeable future
- DOE's Exascale Computing Project: Investing in hardware and software for science
- Geant On GPU: prior efforts
- Exascale Geant Pilot: a collaboration between ECP and HEP
- Goals and strategy





Exascale: GPU for the Foreseeable Future



- **Perlmutter** (NERSC-9, LBNL) 2020: AMD and Nvidia GPUs
  - GPU nodes will have a GPU to CPU ratio of 4:1
  - 256GB memory per node or greater
  - >4000 node CPU partition, approximately same capability as full Cori system today
- **Frontier** (ORNL) 2021 and **El Capitan** (LLNL) 2023:
  - one AMD EPYC™ processor to four AMD Instinct graphics cards
- **A21** (ANL) 2021:
  - Intel Configurable Spatial Accelerator
    - Dataflow graph engine
    - Maps compiler execution graph (IR) to hardware fabric
    - Elements of modern switches, FPGAs, KNL
  - Intel GPU, Intel ONE API





Exascale Computing Project

EXASCALE COMPUTING PROJECT

- Preparing for the next generation of supercomputers.
- 10-year project led by six DOE and NNSA laboratories and executed in collaboration with academia and industry
- Accelerating delivery of a capable exascale computing ***ecosystem*** for breakthroughs in scientific discovery, energy assurance, economic competitiveness, and national security.
- Goal to deliver breakthrough modeling and simulation solutions that analyze more data in less time, providing insights and ***answers to the most critical US challenges in scientific discovery***, energy assurance, economic competitiveness, and national security

DOE, Exascale, GPU, and HEP

- DOE push for use of Exascale Computing (and thus GPU) not limited to ECP.
  - ECP is through the DOE Office of Science's ***Advanced Scientific Computing Research***
- DOE Office of Science's ***High Energy Physics*** program is also investing towards use of GPU
- Explicit constraints for experiments in upcoming years

  Use (mostly/only) Exascale machines

  and

  Use of Exascale machines allowed ***only*** if making efficient use of accelerators

- Started first (AFAIK) official collaboration between HEP and ECP
  - Investigating general purpose detector simulation on GPU
  - Fermilab, Lawrence Berkeley Lab, Oak Ridge National Lab, University of Pittsburg
  - Interest from researchers associated with US-ATLAS and US-CMS

Geant On GPU: Existing Efforts

- **MPEXS-DNA: simulator for track structures and radiation chemistry at subcellular scale.**
  - EM Physics and diffusion and chemical reactions for molecular species for micro dosimetry simulation
  - Speed up ***2900x*** (NVidia Volta)
- **GATE simulation on GPU**
  - Simulations of Preclinical and Clinical Scans in Emission Tomography, Transmission Tomography and Radiation Therapy
  - GPU only for voxelized phantom (very time consuming) - ***60x***
  - GPU only for PET and CT applications (validated in Bert et al2013) – ***70x***
- **G4CU:** Geant4 Based Simulation of Radiation Dosimetry in CUDA
  - Focused on dose calculation for radiation therapy
  - EM Physics in H_2O in voxel geometry, ***75~97 speedup***
- **Opticks:** GPU Optical photon simulation for Particle Physics with NVDIA's OptiX
  - ***200x speedup*** with mobile GPU (GeForce GT 750M)

- Applications where unlike HEP there are usually a small subset of code/work/configuration that dominates execution time and that have inherently less conditional branches
- Those packages are often closed source.

Geant Exascale/GPU Pilot Project

- Goal to **study and characterize architecture and performance**
- Will encapsulate, at first, a minimum set of physics and software framework processes necessary to describe a HEP simulation problem.
- Will then be used to exercise/research communication, computation and data access patterns.

- Main objective is to identify re-engineering opportunities that will **increase event throughput** by improving single node performance and being able to make efficient use of of the next generation of accelerators available in ***exascale*** facilities.

Goals

- Investigate how to best use GPUs for full HEP simulation
  - Estimate higher limit of speed-ups.
- Explore memory access, computation ordering, and CPU/GPU communication patterns
  - Require flexibility on data structure and code structure
- ***Avoid over-simplification***
  - Look at the whole Geant simulation chain not just a few components
- Proxy for ***generic*** Monte-Carlo transport simulation framework that executes on either the CPU or GPU or combination of both
  - E.g. use GPU where speed-up has been demonstrated, avoid GPU where speed-up is highly unlikely (e.g. physics with significant branching; particles with short-lifetimes and/or a relatively small average number of interactions)
  - Cost of data transfers will be a key metric

Strategies

- Avoid constraints
  - Backward compatibility is not a requirement
  - Tracking change ‘upstream’
- Reuse or leverage existing packages
  - Newer components like VecGeom, latest magnetic field and physics implementation.
- Focus on NVidia compiler at first
  - Will need to understand AMD and Intel programming direction
  - Likely might switch to a library like Kokkos

GPUs and Polymorphism

- Detector simulation relies (needs) polymorphismto support a wide range of particles and physics models and to allow customization of the simulation (e.g. Custom Physics lists).
- Copying objects to and from GPU invalidates virtual table (if any)
- Full Static polymorphism
  - typically leads to template propagation throughout the code
- Solution: Partial Static Polymorphism
  - Hybrid generic and type-specific container
  - Virtuality "Eraser"
    - Mechanism to access content of concrete class 'Track' without going through virtual table.
- Physics list configuration at compile time through template and traits

Separation Of State and Access.

- Split the struct holding the Track class from the accessors and modifiers functions.
- Allow for smaller set of interfaces
  - Geometry info, Physics info, Etc.
- Will allow change/investigation in physical layout of Track transparently to the rest of code.
  - Possibly different layout (eg. shorter/smaller) on GPU than CPU
  - Struct of Array vs Array of Struct
  - Etc.
- Allow to factor out more ‘track’ related functionality
  - while still keep it close to the (kind of) code that uses it and without having unwieldly large classes.
- “Should” disappear thanks to inlining and optimization

Conclusion

- New project to explore GPU use for HEP Geant Simulations
- Targeting the upcoming Exascale class machine
- Collaboration between ECP and HEP involving FNAL, ORNL and LBL
  - Joint expertise in HPC, GPU, Geant4/V
- Currently setting up the infrastructure classes and functionality
- Timeframe for incorporating ideas/solutions into production quality simulation toolkit is HL-HLC

Backup slides

Partial Static Polymorphism

- Hybrid generic and type-specific container

```
template <typename... ParticleTypes> struct VariadicTrackManager {
    …
    // without template param, we push to generic at end
    void PushTrack(Track* track) { m_tracks[num_types]->PushTrack(track); }

    // with template params, we push to particle-specific queue
    template <typename ParticleType,
              enable_if_t<(is_one_of_v<ParticleType, TupleType>)> = 0>
    void PushTrack(Track* track) {
        m_tracks[index_of<ParticleType, TupleType>::value]->PushTrack(track);
    }

    // without template params, we pop from end
    Track* PopTrack()

    template <typename ParticleType> TrackCaster<ParticleType>* PopTrack();
};
```

Virtuality "Eraser"

- Mechanism to access content of concrete class 'Track' without going through virtual table:

```
template <typename Type>
struct TrackCaster : public Track
{
    Type*  GetParticleDefinition() const { return static_cast<Type*>(m_pdef); }
    std::string GetParticleName() const
    {
       return static_cast<Type*>(m_pdef)->Type::GetName();
    }
    // no need to reimplement GetTrackId()
};
```

Dispatching

- Configuration of actions at compile time via “traits”.

```
template <typename... ParticleTypes> struct ParticleStepper
{
    // ...
    void AlongStep( … )
    {
        // component::start is the operator struct this is the "TypeList"
        using ProcessTypes = std::tuple< AlongStep<ParticleTypes>… >;

        // "ProcessTypes" (our operator typelist) expands to
        //
        //    std::tuple < AlongStep < Photon >,
        //                 AlongStep < Electron > >;
        apply<void>::access< ProcessTypes >(  …. );
    }
    // ...
};
```

Description of the functionality necessary for the Geant Exascale Proxy

The proxy should be a standalone application with a mean to explore various detector layouts, for example the CMS detector and a Liquid Argon TPC. It should provide interfaces for configuration and options for transport, event input, and output and either indirectly or directly include performance measurement tools. The primary goal of the proxy is to explore the best ways to leverage the type of coprocessor (GPGPU) expected in the next generation of HPC machines (even if this require more code and dependency than a typical proxy application)

Configuration interfaces

physics lists
detector descriptions (e.g. GDML)
input event types

Some lessons learned from GeantV

- Main factors in the speedup: modern code developed from ground up
  - Better cache use
  - Tighter code (e.g., less indirections and branching)
- Vectorization's impact (much) smaller than hoped for
  - Effectively small fraction of the GeantV code has been vectorized or is run in vector mode. (Amdahl's law limits overall speedup.)
- Basketization is challenging
  - Either extra memory copy (using collection of tracks)
  - Or lower memory access coherency (using collection of pointers)
- "Localized" benchmark performance challenging to extend to full use case
- Migration of backward incompatible solutions [not necessarily GeantV] to a real experiment is manageable within the timescale of HL-LHC
  - Example: Successful integration tests with CMSSW, including physics validation and computing performance evaluation – GeantV events reconstituted, hits ready for digitization!
  - Heterogeneous computing solutions possible – CMSSW external work feature

Options and Potential Strategies

1. "Physics Preserving" options
   a) Code modernization using lessons learned from GeantV and other R&D lines of work
   b) Adaptation to HPC and efficient use of accelerators
2. Fast Simulation options
   a) Improvements to parametrized simulation or extend fast simulation use cases – Example: CMS' current parametrized simulation (<<1 sec/event and reproduces most physics observables within 10%)
   b) R&D on Machine Learning (ML/AI) to achieve acceptable speed/fidelity balance

External conditions from experiments and computing technology

- Experiments need high precision for e.g. MET, boosted objects, jet sub-structure, particle separation in high-granularity detectors – need full G4 simulation as in run 2
- ML may prove to replace full sim (physics preserving) or be just another fast sim option with limited applications – need large "full simulation" for training anyway (HPC?)
- HPC facilities increasing fraction of HEP computing resources – guidance from funding agencies: "adapt, run on accelerators, save us money"

Transport Options

For a reasonably representative example of simulation, we find that processing photon, e-, e+, take have about 80% of the CPU time specifically dedicated to particle processing and represents 70% of the number of particles. If we also include neutrons, pi+, pi-, and protons this goes up to 99% of the cpu time and 98% of the number of particles. In addition the code for the simulating the former set exist in a easier-to-extract form than for the later set.

Given these ratio, in the first phase the application should focus on:

charged particles in uniform electromagnetic field realistic electromagnetic field description used in typical HEP experiments standard set of electromagnetic physics processes and models required for most of HEP detector simulation, including:

- Charged particles
  - Ionization: Bremsstrahlung, Multiple scattering models (electron and positron)
- Photons
  - Photoelectric Effect, Compton Scattering, Pair-production

Event Input

Particle gun with customizations:

- number of particles
- particle types
- random distributions in energy, eta, phi in given ranges
- Ability to use HEPevents would be a plus

Output

The proxy should be capable of producing output data representative of a typical HEP experiment, for example digitized hits (scoring) in sensitive detectors.

Performance Measurements

The proxy should include (directly or indirectly) the ability to measure:

- scalability
- event throughput
- memory usage under various configuration settings, including multi-threading

Additional Notes

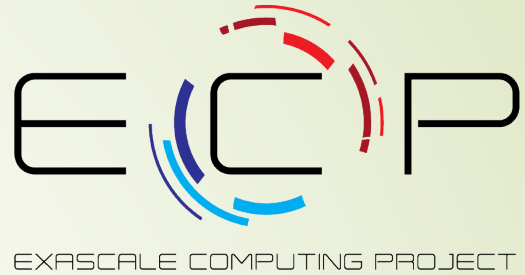
An estimate of the date size for input events for a real physics process (for example Higgs -> ZZ , Z to all decays, O(1000) primary tracks) is in the order of 10's of kilobytes (for example 32K/event at the LHC energy). A typical output rate is in the order of 1.5 MB per event (for simulated data, compared to about 1MB for the equivalent real data.)

ECP Strategic Goals and Outcomes

EXASCALE COMPUTING PROJECT

- ***Goal: Applications***
  - Foundational element of ECP and vehicle for delivery of results from the exascale systems enabled by ECP. Each addresses an exascale challenge problem—a problem of strategic importance and national interest that is intractable without at least **50** times the computational power of today's systems
- ***Goal: Software Technologies***
  - Underlying technologies that applications are built and relied on: essential for application performance, portability, integrity, and resilience. Spans low-level system software to high-level application development environments, including infrastructure for large-scale data science and an expanded and vertically integrated software stack with advanced mathematical libs and frameworks, extreme-scale programming environments, tools, and visualization libs
- ***Goal: Hardware and Integration***
  - Partnerships between U.S. vendors and DOE's application and software developers to develop a new generation of commodity computing components. Assure at least two diverse and viable exascale computing technology pathways for the Nation to meet identified mission needs and proactively engage and integrate with DOE HPC facilities in the process.
- ***Outcome: Accelerated delivery of a capable exascale computing ecosystem*** to provide breakthrough solutions addressing our most critical challenges in scientific discovery, energy assurance, economic competitiveness, and national security
  - *Capable*: Wide range of applications can effectively use the systems developed through ECP, ensuring that both science and security needs will be addressed (affordable, usable, useful)
  - *Exascale*: Enhance application performance by **50x** relative to today's systems (e.g., ORNL's Titan)
  - *Ecosystem*: Not just more powerful systems, but all methods and tools needed for effective use of ECP-enabled exascale systems to be acquired by DOE labs

Application Challenges on Exascale Systems



- **Get ready by running well on today's machines**
  - If you can run well on today's machines you will be ready enough that you can update your code in time
  - Leverage standards or frameworks:
    - OpenMP, Kokkos and RAJA will likely work on whatever comes out and can limit your exposure to porting yourself
  - Avoid the newest language features unless needed
    - In particular, modern Fortran support with OpenMP tends to lag behind
- **Exascale is happening through increased parallelism not clock speeds**
  - While a continuation of the last decade this makes two things harder
    - Improving time to solution
    - Non-parallel code sections/algorithms
  - This will be made worse because network latencies, gpu kernel launch times and other Amdahl bottlenecks are hard to drive down at the rate of throughput increases.
- **Total memory capacity is unlikely to scale**

Ian Karlin , LLNL
at 2019 ECP Annual Meeting

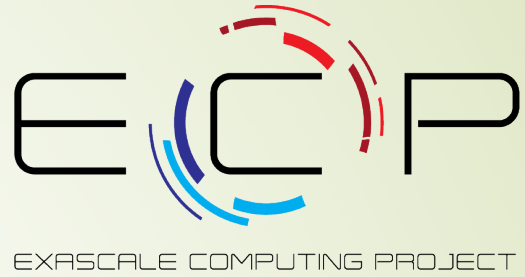
Most significant challenges in preparing applications for exascale machines

EXASCALE COMPUTING PROJECT

- Granularity of parallelism
  - Expose a lot, use less.
- Transition overhead and data movement
  - Network and memory speed will decrease relative to compute speed.
- Data-structure restructuring needed – Not just loops

Hal Finkel, ANL
at 2019 annual ECP meeting

Programming Preparation Thoughts



David Donofrio, BNL
at 2019 ECP Annual Meeting

- Do what you can on existing machines
  - Port to GPUs now (E.g., Summit, Sierra)
  - Port to KNL now (vectorization) (E.g., Cori)
- Begin to partition Data structures and multi-level problem decomposition
- Accelerators (surprise!) are important
  - Highest fraction of peak performance is here
  - SW tools continue to improve
- Increasing on-node parallelism
- Interconnect performance not tracking increase in node performance
- More memory per node (good news!)
  - Less memory per thread (not so good news)
- Trending to tighter integration of CPUs and accelerators
- Increasing coherence – simplifies programming... but often at performance cost
- Memory / Allocation increasingly important

ECP focus areas

Application Development (AD)

Develops and enhances predictive capability of applications critical to DOE across science, energy, and national security mission space

- Targeted development of requirements-based models, algorithms, and methods
- Integration of appropriate software and hardware via co-design methodologies
- Systematic improvement of exascale system readiness and utilization
- Demonstration and assessment of effective software integration

Software Technology (ST)

Spans low-level operational software to high-level applications software development environments, including support for data needs of science and national security activities at exascale

- Line of sight to applications
- Software Development Kits (SDKs) to enhance the drive for collaboration
- Delivery of specific software products

Hardware and Integration (HI)

Ensures integrated delivery of specific outcomes and products on targeted systems at leading DOE HPC facilities

- PathForward
- Hardware evaluation
- Application integration at facilities
- Software deployment at facilities
- Facility resource utilization
- Training and productivity

Exascale Computing Project 2.0

Project Management 2.1

- Project Planning and Management 2.1.1
- Project Controls and Risk Management 2.1.2
- Business Management 2.1.3
- Procurement Management 2.1.4
- Information Technology and Quality Management 2.1.5
- Communications and Outreach 2.1.6

Application Development 2.2

- Chemistry and Materials Applications 2.2.1
- Energy Applications 2.2.2
- Earth and Space Science Applications 2.2.3
- Data Analytics and Optimization Applications 2.2.4
- National Security Applications 2.2.5
- Co-Design 2.2.6

Software Technology 2.3

- Programming Models and Runtimes 2.3.1
- Development Tools 2.3.2
- Math Libraries 2.3.3
- Data and Visualization 2.3.4
- Software Ecosystem and Delivery 2.3.5

Hardware and Integration 2.4

- PathForward 2.4.1
- Hardware Evaluation 2.4.2
- Application Integration at Facilities 2.4.3
- Software Deployment at Facilities 2.4.4
- Facility Resource Utilization 2.4.5
- Training and Productivity 2.4.6

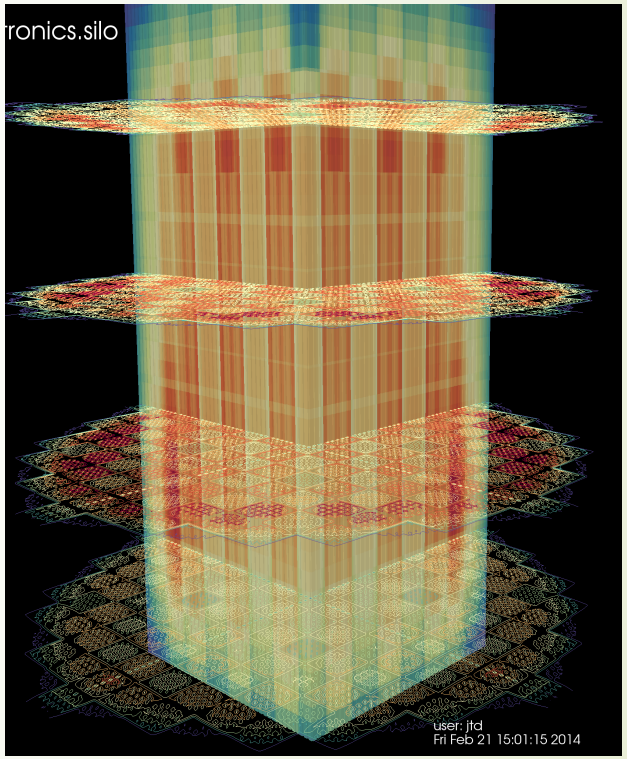
Energy Applications

Modeling and simulation of existing and future technologies for the efficient and responsible production
of energy that meets the growing needs of the US

WarpX	Modeling of plasma-based particle reactors, leading to the design of a 1 TeV electron-positron high-energy collider
MFIX-Exa	Increase the efficiency and reduce the cost and time to market for complex multiphase flow reactor designs for carbon capture and storage technologies
ExaSMR	Design optimization for small modular reactors, a key component in affordable nuclear energy for the future
WDMApp	Whole-device, high-fidelity modeling approach for magnetically confined fusion plasmas applicable to a high-performance advanced tokamak regime
Combustion-PELE	Detailed modeling and simulation of combustion physics to reduce petroleum use (increase efficiency) and reduce greenhouse gas emissions
ExaWind	Modeling and simulation of whole wind plant performance

ExaSMR – motivation for HPC in nuclear reactor modeling

- Value of HPC computing in nuclear was demonstrated by CASL
  - 2012: Shift (MC) neutronics calculations were used to validate startup conditions of the new WEC AP1000® reactor
  - 2015 – 2017: VERA – Shift is being used to model WB2 startup
  - These calculations use most of Titan
    - 18 688 compute nodes
    - 10^{12} MC particle histories
  - CASL is primarily focused on industry-level computing
    - Not addressing coupled, multicycle high-resolution modeling
    - High-order neutronics used for benchmarking
    - Transport for Ex-core dosimetry and pressure vessel fluence

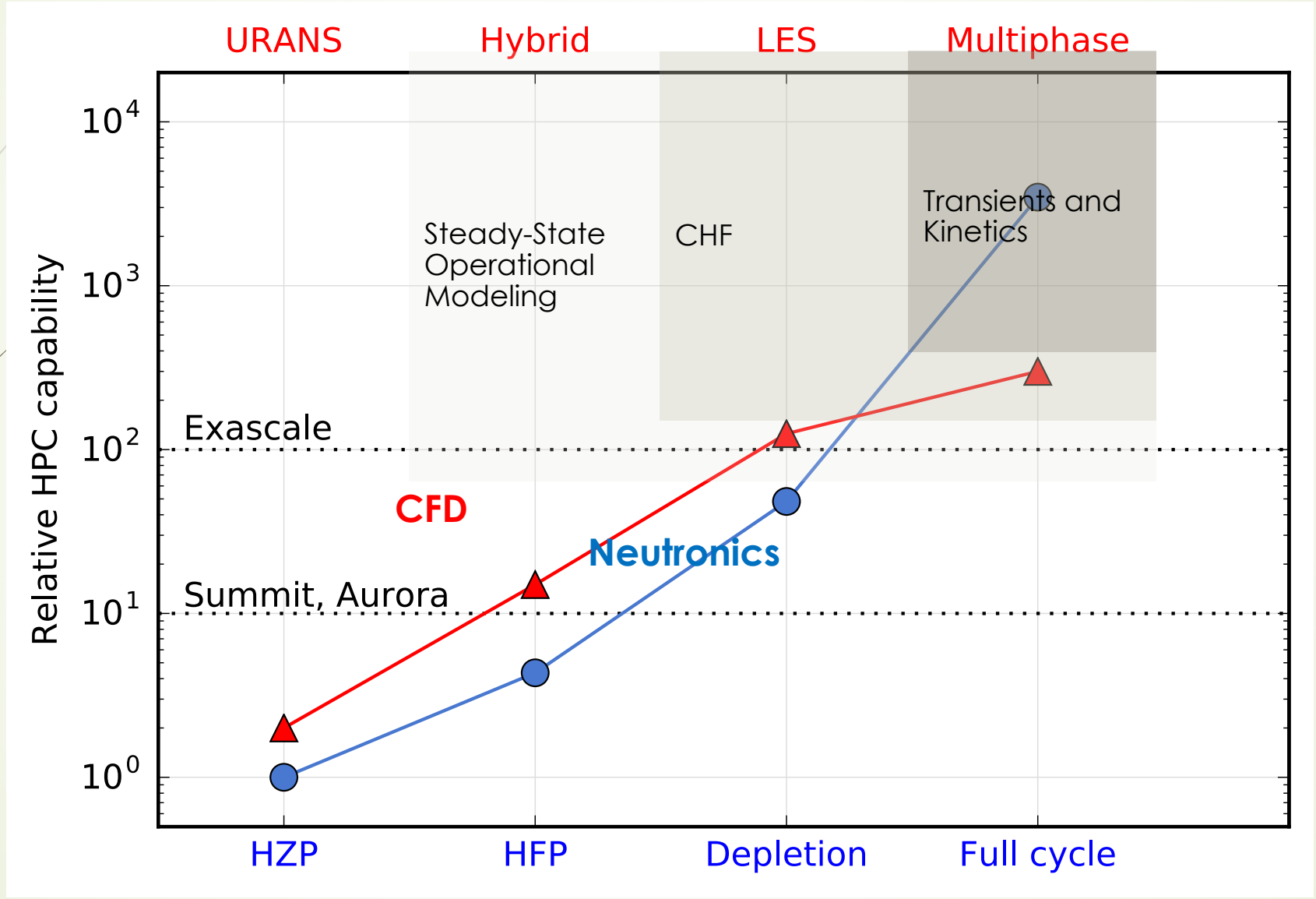


Small Modular Reactor (SMR) Challenge Problems require high-resolution models

- Coupled MC neutronics/CFD simulations steady state operation
  - Model movable absorbers
  - Optimize fuel assembly designs with strong power gradients
- Coupled MC neutronics/CFD simulations of the full core in steady-state, low-flow critical heat flux (CHF) conditions
  - Demonstrate a substantial steady-state CHF margin
- Transient coupled MC neutronics/CFD simulations of the low-flow natural circulation startup
  - Significant advancement over empirically correlated 5 × 5 rod bundle experiments that are used for design

Couple MC neutronics and LES, multiphase CFD are required to address these challenges

Required compute capability for Challenge Problems



Geant on GPU

- MPEXS-DNA: https://www.ncbi.nlm.nih.gov/pubmed/30593679 , https://aapm.onlinelibrary.wiley.com/doi/full/10.1002/mp.13048
- Gate http://www.opengatecollaboration.org/sites/default/files/Talk6.pdf , https://www.creatis.insa-lyon.fr/~dsarrut/articles/Bert2012.pdf , http://www.opengatecollaboration.org/#
- G4CU: https://kds.kek.jp/indico/event/15926/session/30/contribution/110/material/slides/0.pdf
- Opticks: https://indico.cern.ch/event/647154/contributions/2733123/attachments/1529047/2392245/opticks_gpu_optical_photon_simulation_sep2017_wollongong.pdf