Version 11.3

12th International Geant4 Tutorial in Korea 2025

Date 3-7 Feb 2025

Place Pohang Accelerator Res. Bld-2-201

Heterogeneous Computing

Soon Yung Jun (Fermilab)

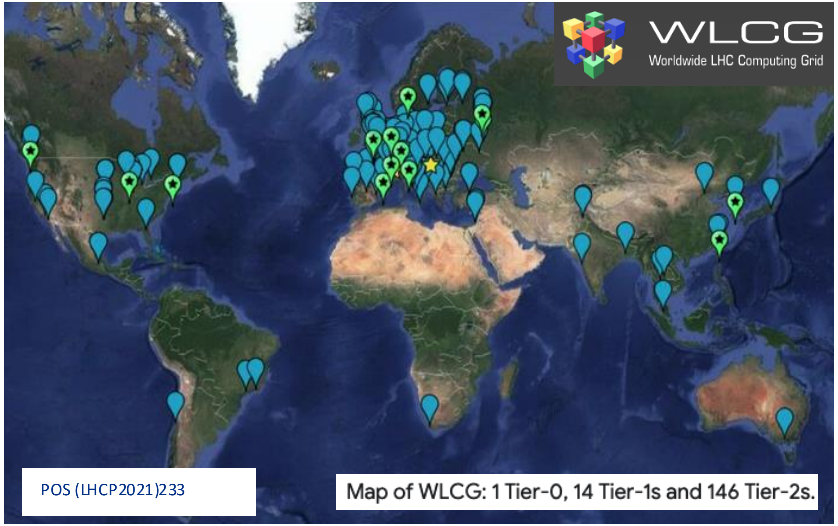
12th International Geant4 Tutorial in Korea,

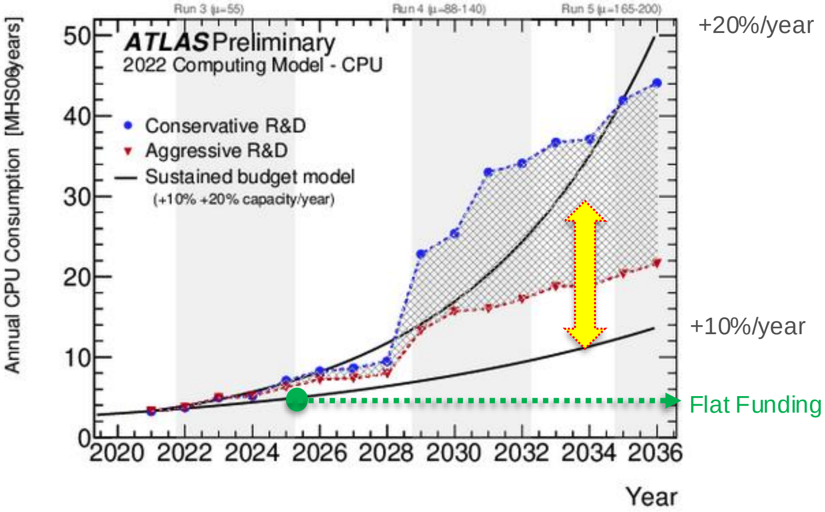
Feb 3-7, 2025@ Pohang Accelerator Laboratory, Pohang

Contents

Computing and Software for Big Science

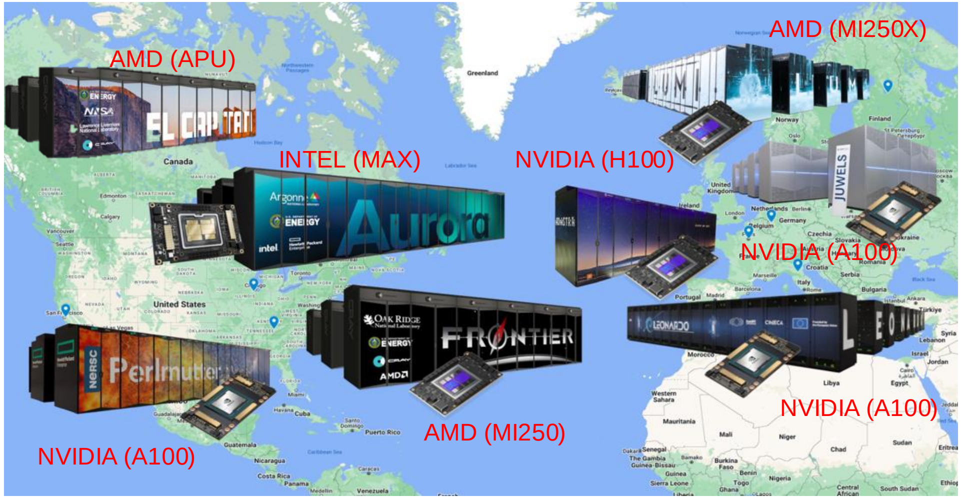
- Consider: Conventional HEP computing and challenges for future experiments
  - Distributed and High Throughput Computing (HTC): WLCG as an example – Dennard scaling, ESnet
  - Future programs (e.g., HL-HLC) requires ~10x computational throughout
    - Detector simulation (Geant4) is significant part of computing usage
  - “Heterogeneous” architectures are increasingly common in high performance computing (HPC)





Heterogeneous Computing

- Computing with more diverse processing elements other than CPU
  - Heavy CPU centric (HTC) → combination of CPU and Accelerators (GPUs, TPUs, NPUs, FPGA etc.)
  - Optimize for very specific domain applications for the triplet (power, performance, cost) and maximize workload/throughput by verticalization (hardware designs ←→ software stacks)
- Ever-changing hardware landscape
  - GPU is the current main HPC architecture
  - Equipped with different type of GPUs
  - Trend will be accelerated by AI/ML
  - Requires scalable, portable, flexible and energy efficient software workflows

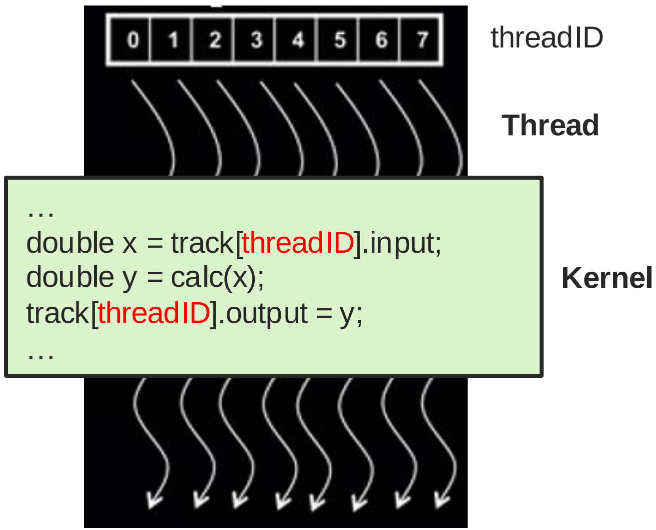


- Detector simulation needs to takes advantage of increasing heterogeneity in computing, especially Exa-scale computers with GPUs.

Detector Simulation on GPUs

- GPU (massively many cores) architecture
  - Single Instruction, Multiple Threads (SIMT)
  - Single Instruction, Multiple Data (SIMD)
  - Optimal algorithms for GPUs
    - Maximal instruction throughput and data locality
    - Minimal branch and memory access

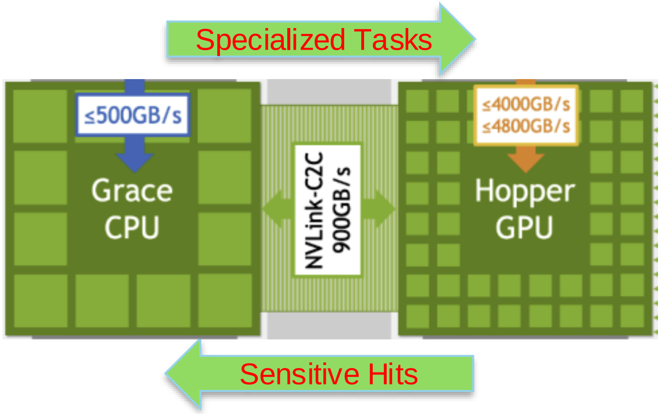
	Maximize	Minimize
Instruction	Throughput	Divergence
Data/Memory	Locality	Latency



- It is challenging to port the full fidelity detector simulation on GPU
  - Particle tracking is stochastic and history dependent → many branches and divergence
  - Detector simulation with a complex geometry is usually a memory-intensive application→ poor locality and high latency (i.e., data re-usability is relatively low)

Geant4 Strategies

- Do as much as detailed simulation on CPU
  - Modern CPU architecture is also evolving toward power efficient many-core technologies
  - Geant4 supports options for parallel workflows as events or tracks can be simulated independently
    - G4Multithreading for event level parallelism (Since 10.0)
    - G4Tasking for task- or track-level parallelism (Since 11.0)
- Offload specialized tasks on GPU (massively-many-core)
  - Each GPU task should have strictly limited scope
  - Particle type and geometry are better to be self-contained
  - Minimize communication between host and device
  - Minimize output (hits, unprocessed tracks) from GPU



- G4Tasking or Sub-event parallelism are solutions that allow various tasks running on GPUs in parallel (or concurrently) while conducting the rest of event simulation on CPUs

Geant4 Tasking

- Geant4 supports a task-based framework (G4Tasking) from v11.0 (source/tasking)
  - Based on PTL (parallel tasking library, tasking system featuring thread-pool, task-group, and task-queue using C++ thread) or TBB backend
  - Support G4RunManagerType = {Serial, MT, Tasking,TBB, SubEvt} using G4RunManagerFactory or environment variables (G4RUN_MANAGER_TYPE)

```
auto* rm = G4RunManagerFactory::CreateRunManager(G4RunManagerType::Tasking);
```

- G4Tasking opens opportunities for heterogeneous computing or hybrid workflows
  - Sub-event level parallelism (from events to tracks)
    - Each G4PrimaryParticle or G4Track can be a task
    - A group of selected particles can be executed in a thread-pool
    - A group of special tasks can be a task-group
  - Concurrent simulation workflow with co-processors and support offloading specialized tasks to GPUs

Toward Sub-event Parallelism in Geant4

- Split an event into sub-events (sub-group of primary tracks) and task them separately
- Phase-I: Geant4 Kernel extension (version 11.3)
  - Add *G4SubEvent* and *G4SubEventTrackStack*
  - Uses *G4SubEvtRunManager* and *G4WorkerSubEvtRunManager* for sub-event parallelism which takes a vector of tracks for a specific subevent type
  - Only the master run manager owns the primary event generator while *G4WorkerSubEvtRunManager* takes a vector of tracks for a specific subevent type
  - Extend G4RunManagerType = {Serial, MT, Tasking,TBB, SubEvt} using *G4RunManagerFactory*

```
auto* rm = G4RunManagerFactory::CreateRunManager(G4RunManagerType::SubEvt);
```

or environment variables

```
export G4RUN_MANAGER_TYPE =“SubEvt”
```

```
export G4FORCE_RUN_MANAGER_TYPE =“SubEvt”
```

Toward Sub-event Parallelism in Geant4

- Phase-II: Extension for optimal use of sub-event parallelism
  - Support specialized physics lists and/or detector construction (geometry) dedicated to sub-tasks
  - Support trajectory and visualization for tracks inside sub-tasks
- Example: examples/extended/runAndEvent/RE03
  - Set *G4RunManagerType* to SubEvt (using env G4RUN_MANAGER_TYPE="SubEvt")

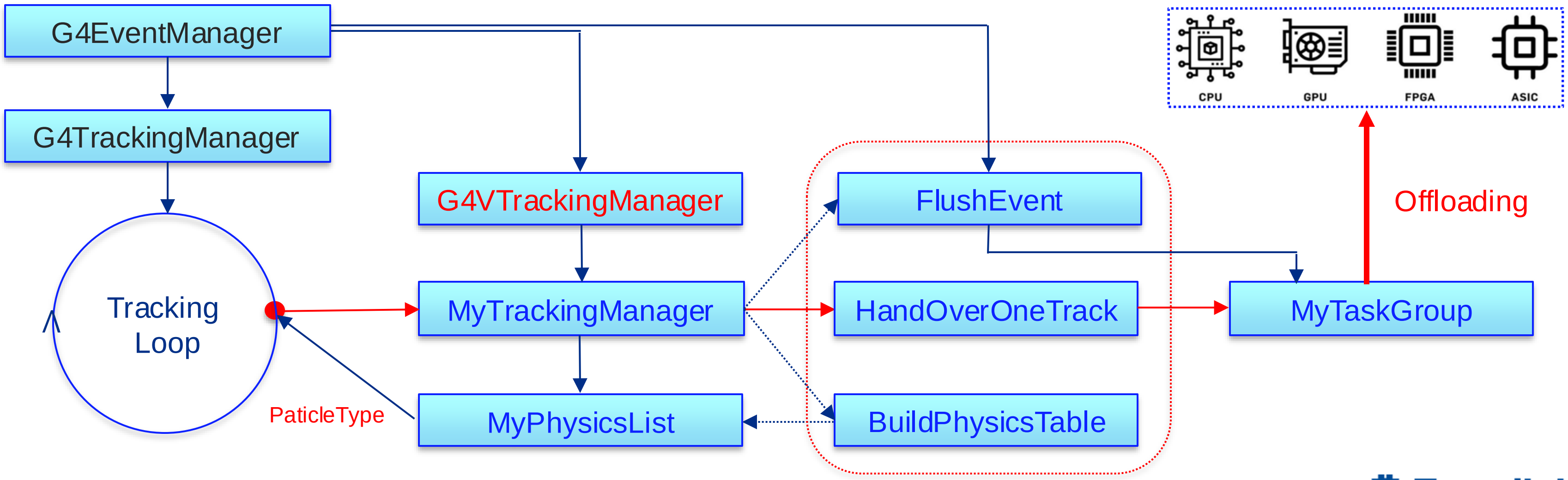
```
void ActionInitialization::BuildForMaster() const
{
  auto* runManager = G4RunManager::GetRunManager();

  // Check if in sub-event parallel mode
  if(runManager->GetRunManagerType()==G4RunManager::subEventMasterRM)
  {
    // Defining size of sub-event
    runManager->RegisterSubEventType(0,100);
    // Sending gamma to the G4WorkerSubEvtRunManager work thread
    runManager->SetDefaultClassification(G4Gamma::Definition(),fSubEvent_0);

    // Primary generator action is defined to the master thread
    SetUserAction(new RE03PrimaryGeneratorAction);
  }
}
```

Sub-event ype = 0
Max Capacity = 100

enum fSubEvent_0 is defined
In G4ClassificationOfNewTrack

Specialized or Custom Tracking Manager for Offloading

- *G4VTrackingManager*: Interface class for implementing a custom tracking manager that is specialized for stepping one or a small number of particle types. Key virtual methods are
  - BuildPhysicsTable(const G4ParticleDefinition&){}
  - HandOverOneTrack(G4Track* aTrack) = 0;
  - FlushEvent(){}



Custom Tracking Manager

- Example: examples/extended/runAndEvent/RE07
- Add particle types

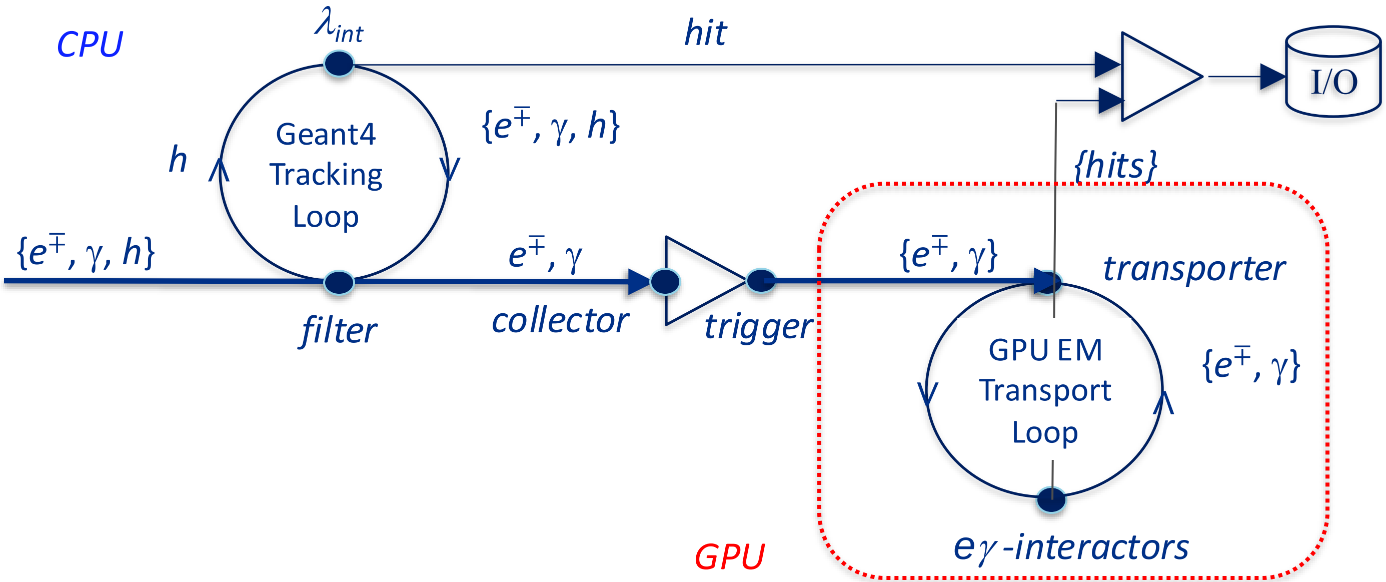
```
void PhysicsListGPU::ConstructProcess()
{
  G4EmStandardPhysics::ConstructProcess();
  G4Gamma::Definition()->SetTrackingManager(new TrackingManagerGPU);
}
```

- Implement *G4VTrackingManager* virtual methods

```
void TrackingManagerGPU::HandOverOneTrack(G4Track* track)
{
    // Collect tracks for offloading
    gpu_transport->Push(*track);

    // G4VTrackingManager takes ownership, so kill Geant4 track
    track->SetTrackStatus(fStopAndKill);
}

void TrackingManagerGPU::FlushEvent()
{
    // Process remaining tracks from the stack
    gpu_transport->Flush():
}
```

Offloading using User Actions

- A hybrid workflow with selected tasks executed on GPUs
  - Optical photon transport in Cerenkov or Scintillation detectors → G4UserSteppingAction
  - EM shower in calorimeter → G4UserTrackingAction



Code Example: Tracking Action

- *PreUserTrackingAction*
  - Select candidate tracks (e.g., $e^{\pm}$, γ) for offloading
  - Store tracks in a stack and trigger tasks within a same task group when the number of stored tracks reaches a pre-defined work unit (configurable)
  - Kill the Geant4 track from the Geant4 tracking loop

```
void TrackingAction::PreUserTrackingAction(const G4Track* track)
{
    // Select applicability for a device task
    if (fDeviceManager->IsApplicable(*track))
    {
        // Add this track for a device task
        fDeviceManager->DoIt(fEventId, *track);

        // Kill it from the Geant4 tracking loop
        (const_cast<G4Track*>(track))->SetTrackStatus(fStopAndKill);
    }
}
```

Code Example: Device Manager

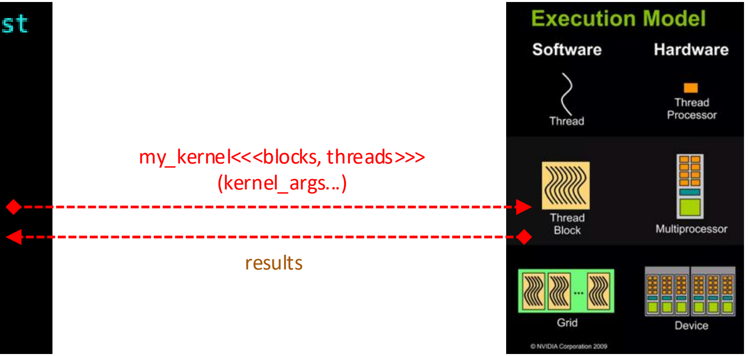
- DoIt action: Add tracks and trigger a device task if criteria are met

```
void DeviceManager::DoIt(id_type eventId, const G4Track& track)
{
    // Convert and store this track for a device stack
    AddTrack(eventId, track);

    if (fStack.size() == Configuration::Instance()->GetChunkTracks())
    {
        // Submit work to Geant4/PTL task-groups
        TaskGroup<void> device_task(Synchronize, fManager->GetThreadPool());
        device_task.exec(DeviceTask, fStack);
        device_task.join();

        // Clear the stack of device tracks
        fStack.clear();
    }
}

void DeviceManager::DeviceTask(const TrackStack& tracks)
{
    fAction.get()->PropagateTracks(tracks);
}
```

Code Example: Device Action

- Connectivity to an external software suitable for GPU

```
void DeviceAction::PropagateTracks(const TrackStack& tracks) const
{
   // Reset and activate devices
   ActivateDevice();

   // Run kernels with input tracks
   auto result = my_gpu_project::em_transporter(tracks);

   // Merge hits
   MergeHit(result.hits);
}
```



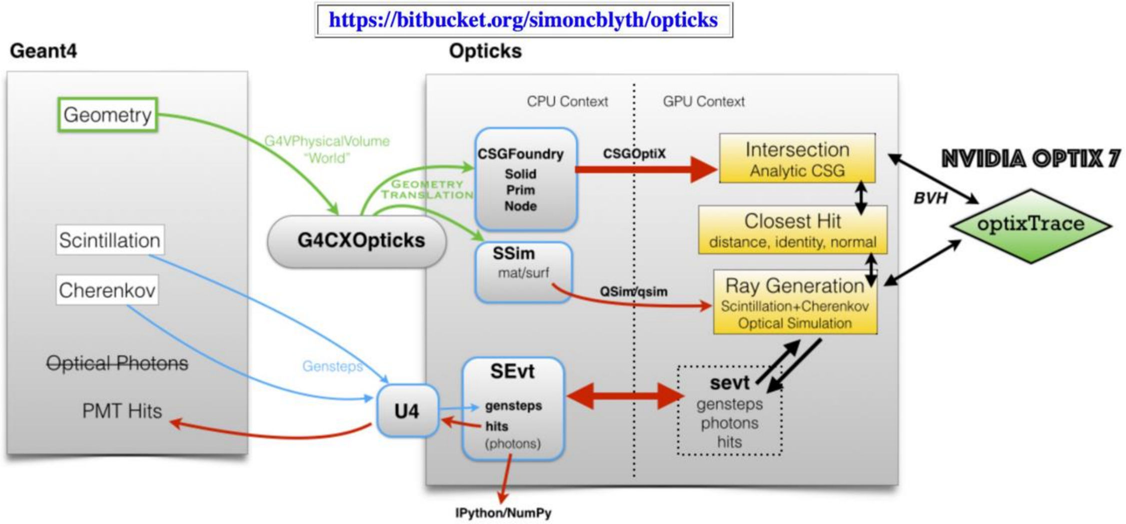
  - em_transporter(tracks) launches GPU kernels with a set of device data {geometry, physics data, input stack of tracks} (H2D)
  - Transfer sensitive hits from devices (D2H) and merge them with Geant4 hits created on the host
  - Process left-over tracks at the end of event or run action (depending on how taskings are organized and scheduled)
  - Execute CPU task (currently event-level-multi-threading), device task, I/O task groups as concurrent as possible (latency hiding)

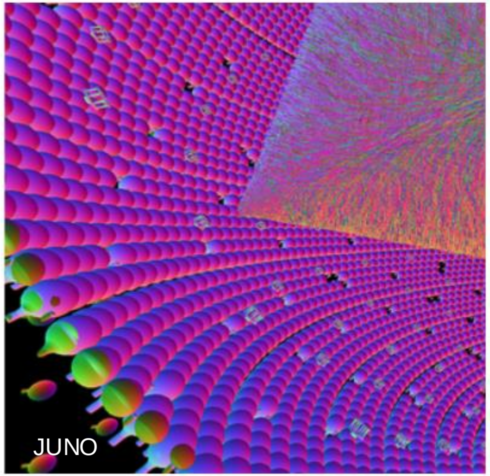
Examples

Example 1: Opticks/OptiX + CaTS

- Optical photons are copiously produced in optical materials (e.g., 45K/MeV energy loss in LAr). Optical photon simulation for a GeV-level charged particle requires significant computational resources (CPU and memory) → usually approximation by using lookup tables.

- Opticks[1] is a project that accelerates optical photon simulation by integrating NVIDIA GPU ray tracing (OptiX™). CaTS is an advance Geant4 example of interfacing with Opticks/OptiX™



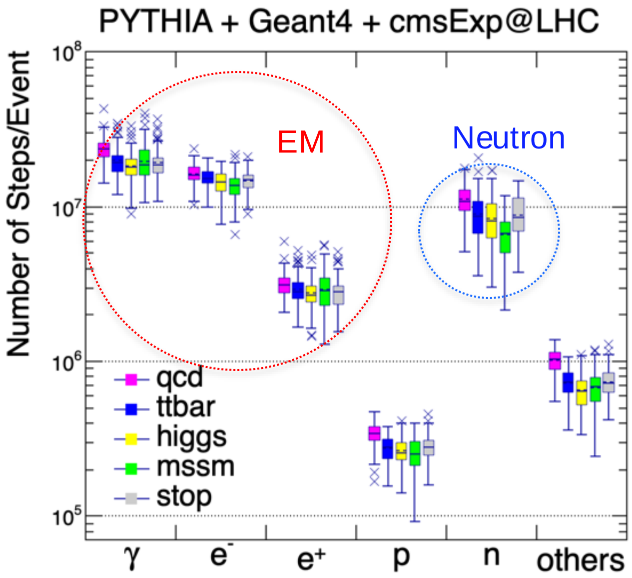


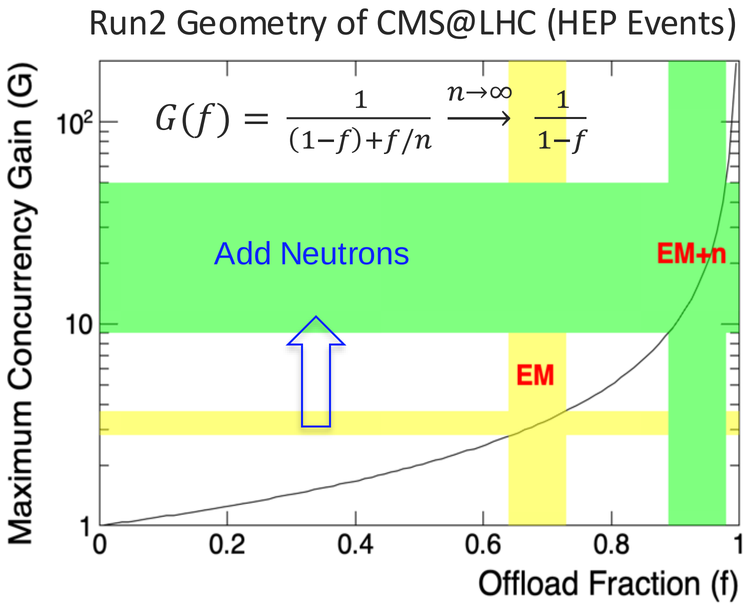
- Potential integration for HEP: DUNE (LAr-TPC), LZ (Liquid Xeon), Calvison, ePIC, etc.

[1] Opticks, S. Blyth, https://bitbucket.org/simoncblyth/opticks/).

Example 2: EM Particle Transport on GPUs

- Hybrid workflow simulation EM particles on GPUs while the rest of hadronic simulation on CPU

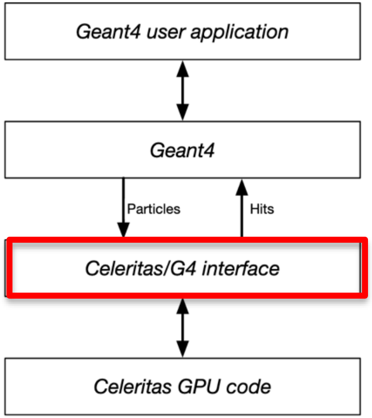


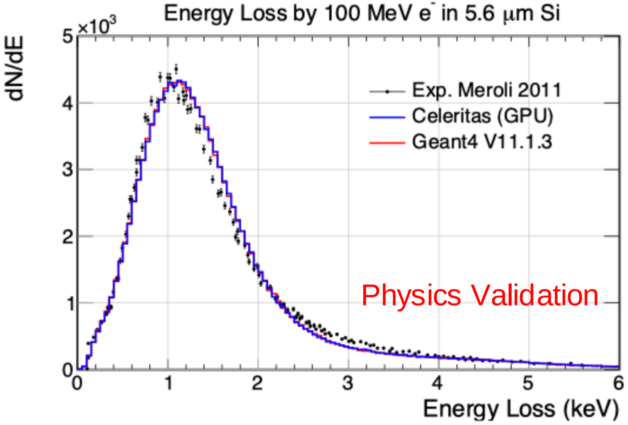


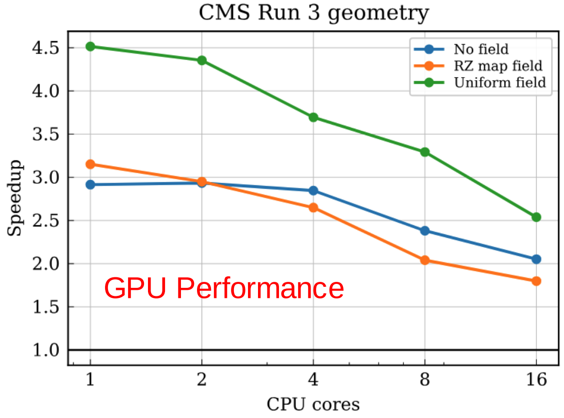
- The maximum gain (G) by the offloading fraction (f) when TimeGPU (f) ⩽ TimeCPU (1- f) with full concurrency (assuming no overhead).
  - EM particles: f = 67% → G = ~3 (EM + neutrons: f = 95% → G = 20)

Example 2: EM Particle Transport on GPUs

- Geant4 provides interfaces for GPU offloading
  - User (Tracking or Stepping) Actions
  - Fast Simulation interfaces
  - *G4VTrackingManager*: a custom tracking manager that is specialized for stepping selected particle types.
- R&D activities: Celeritas and Orange (US HEP-CCE) and AdePT and VecGeom2.0 (CERN)
  - Actively being integrated into experimental frameworks (CMS, ATLAS, etc.) and shows promising results

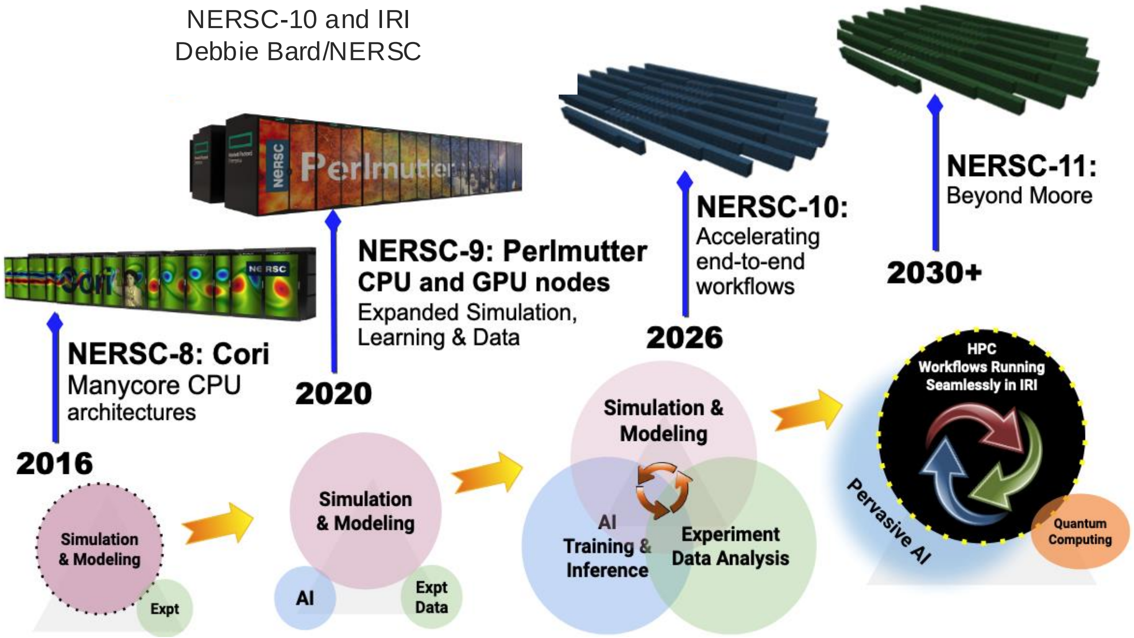






Future Ecosystem: Integrated Research Infrastructure (IRI)

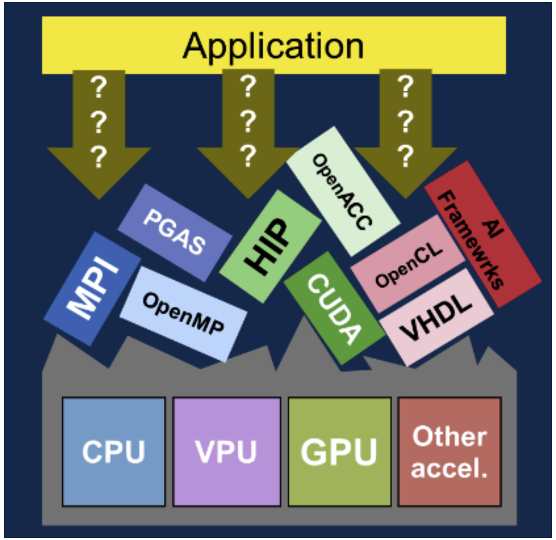
- e.g., NERSC-10 strategic planning: IRI of HPC systems accelerating end-to-end workflows
  - Simulation and modeling
  - AI Training & Inference
  - Experiment data analysis
- Beyond Moore
  - HPC workflow running seamlessly in IRI
- Innovation in software is a key



- Geant4 is complying with the evolving HPC/heterogenous/IRI ecosystems
  - Efficient algorithms, data structure, interfaces for complex workflows → Sustainable and scalable toolkit

Future Ecosystem

- Hardware:
  - Energy-efficient chips
  - AI chips for machine learning and deep learning solutions
  - (quantum Q-bits, neuromorphic chips)
- Software
  - More efficient algorithms, data structures and I/O
  - GPU parallelism and scalability
  - Portability (CUDA, HIP, oneAPI, Alkapa, Kokkos, SYCL, …)



- Integrate infrastructure of HPC/GRID of heterogenous systems
  - WLCG moved from a grid of homogeneous systems to a GRID of interconnected heterogeneous systems
  - Integrated Research Infrastructure (e.g., NERSC-10/11)
  - Federation Platform (e.g., EuroHPC), etc.

Summary

- Advance software tools to use range of architectures to take advantages of increasing heterogeneity in compute resources
- Geant4 supports a task-based framework (G4Tasking) suitable for heterogeneous workflows
  - Event level task parallelism
  - Sub-event level parallelism
  - (Track-level parallelism)
- Examples of on-going HEP detection simulation projects using GPUs within the Geant4 R&D task force
  - Optical photon simulation (Opticks/Nivida OptiX) → CaTS
  - Offloading EM particle transport to (G4HepEM/AdePT and Celeritas)
  - Surface-based geometry models (bounded/VecGeom2.0 or unbounded/ORANGE)
- Geant4 and its applications keep evolving toward HPC/heterogenous systems