Heterogeneous Computing

Soon Yung Jun (Fermilab)

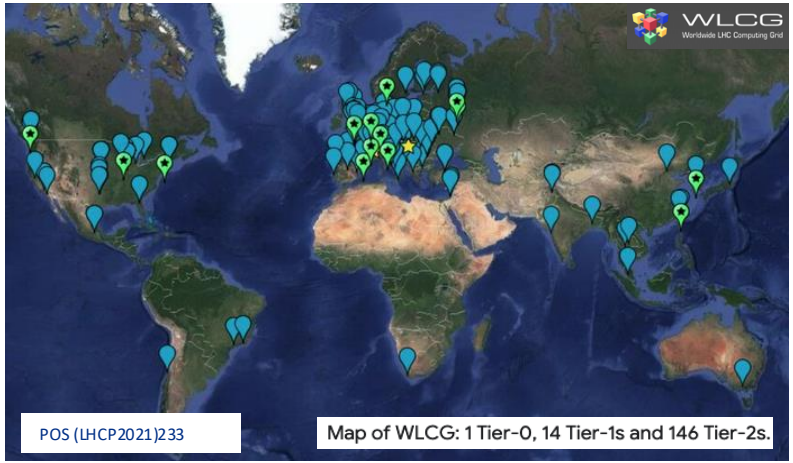
11th International Geant4 Tutorial in Korea,
Aug 19-23, 2024@YonSei Univ., Wonju

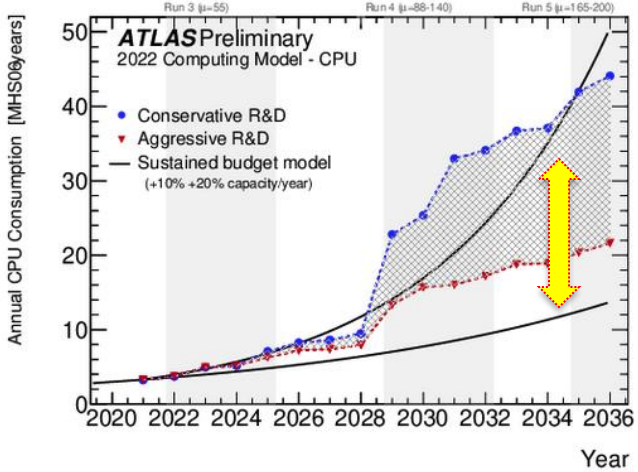
Contents

- Introduction
  - What is heterogeneous computing?
  - Hardware landscape: exascale computing facilities
- Geant4 Tasking
- Examples of Geant4 applications using GPUs
  - Optical photon transport
  - EM particle transport
- Summary

What is Heterogeneous Computing?

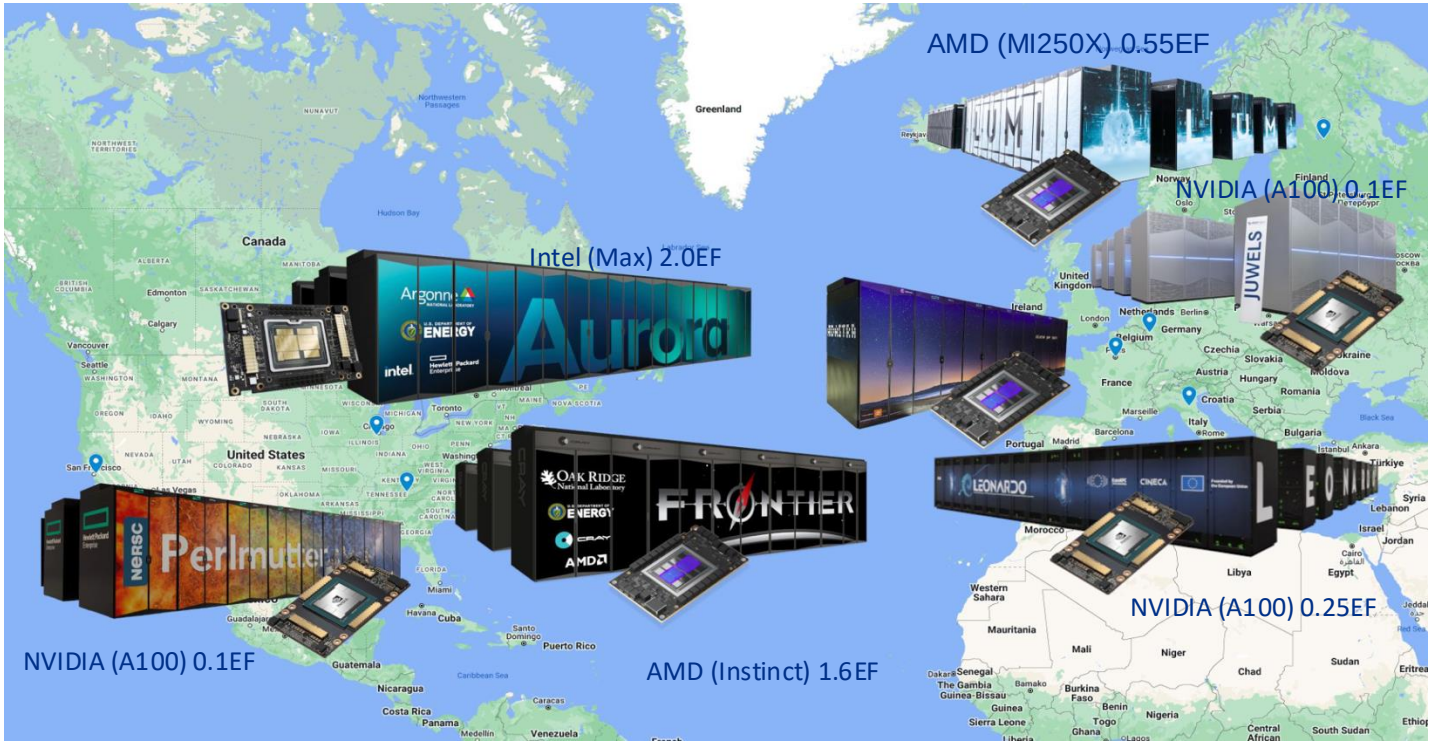
- Computing with more diverse processing elements other than CPU
  - Heavy CPU centric (HTC) → combination of CPU + Accelerators (GPUs, TPUs, NPUs, FPGA etc.)
  - Optimize for very specific domain applications for the triplet (power, performance, cost) and maximize workload by efficient software stacks
  - Verticalization (hardware designs ←→ software stacks)
- Conventional HEP computing and challenges for HL-LHC: WLCG as an example





Ever-Changing Hardware Landscape

- GPU is the current main HPC architecture and can no longer be ignored
- Take advantages of increasing heterogeneity in compute resources



Geant4 Strategies

- Geant4 provides various options for parallel workflows as tracks or events can be simulated independently
  - MPI and multithreading (examples/extended/parallel)
- The recent introduction of G4Tasking opens opportunities for the sub-event level parallelism
- Challenges of tracking particles through a complicated geometry utilizing heterogenous systems
  - History-dependent → high Amdahl's
  - Memory intensive → high latency
  - Many branches → poor locality and instruction level parallelism
- Need to support specialized tasks to accelerators (GPUs)
  - Maximize event throughput per Watt (Power efficient computing)
  - Support flexible workflow managements (load-balancing)
  - Sub-event level parallelism is a good intermediate solution

Monte Carlo simulation on GPUs

- It is challenging to port the full fidelity detector simulation on GPU
  - Each GPU process should have strictly limited scope
  - Physics coverage, particle type, geometry/material are better to be self-contained
  - e.g., optical photon transport in Cerenkov detector, EM shower in calorimeter (w/o back splash or punch through)
- A task on GPU should behave like a blackhole (Makoto Asai)
  - The darker (less output) a task is, the better performance it has
  - Individual step/track should not be taken out from a task
  - Reshuffling tracks over tasks is not a good thing to do
  - Minimize output information
- Sub-event parallelism is the only solution that allows various tasks running on GPU in parallel while conducting the full event simulation

G4Tasking

Geant4 Tasking

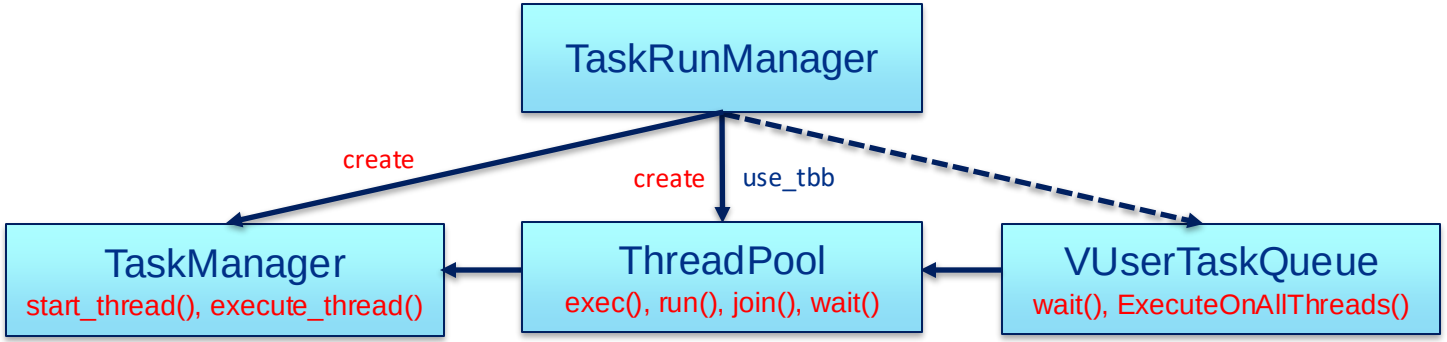
- Geant4 supports a task-based framework (G4Tasking) from v11.0 (source/tasking)
  - Based on PTL (parallel tasking library, tasking system featuring thread-pool, task-group, and task-queue using C++ thread) or TBB backend
  - Support G4RunManagerType = {Serial, MT, Tasking,TBB} using G4RunManagerFactory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

```
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

Key Components of Geant4 Tasking

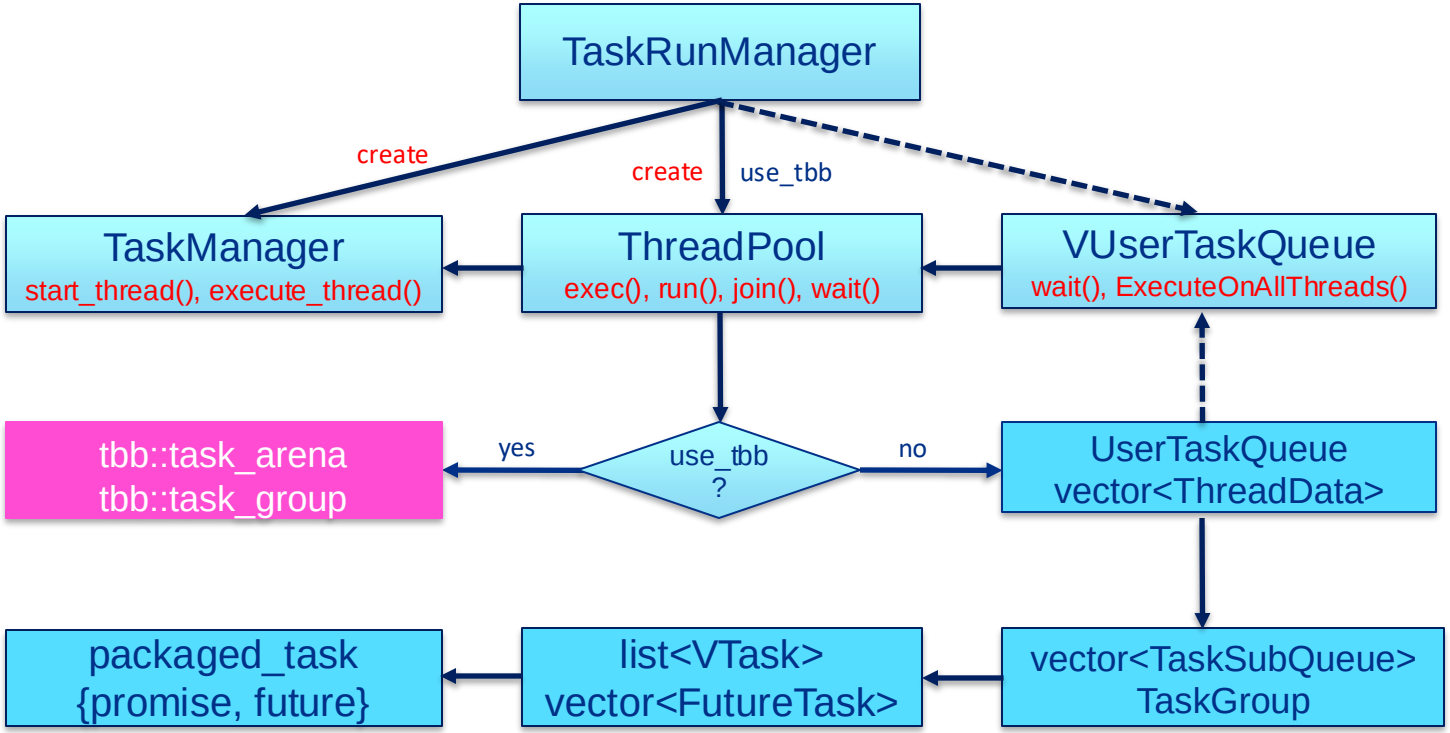
- TaskRunManager: a class for run control in tasking for multi-threaded runs which initializes ThreadPool and TaskManager



- *ThreadPool*: a class for an efficient thread-pool that accepts work in the form of tasks
- *TaskManager*: a class for handling the wrapping of functions into task objects and submitting them to a thread pool
- *VUserTaskQueue*: an abstract base class for creating a task queue used by ThreadPool

Tasking Types: Native (PTL) vs. TBB

- Supports both native TaskGroup/UserTaskQueue and Intel® TBB

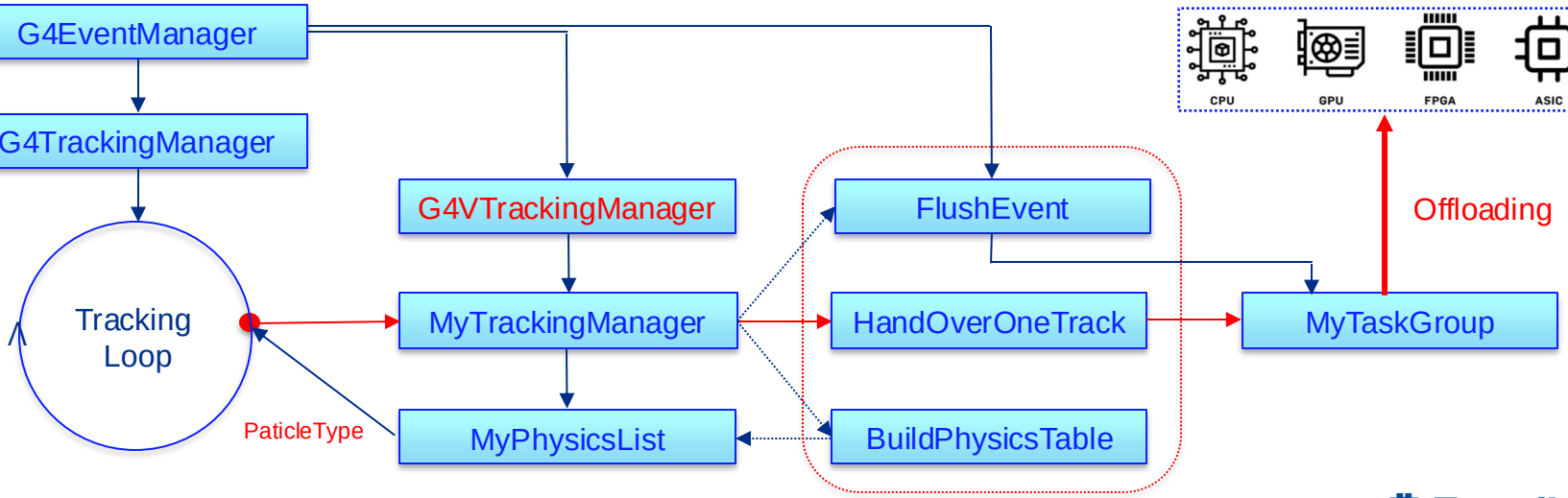


Toward Sub-event Parallelism in Geant4

- Split an event into sub-events (sub-group of primary tracks) and task them separately
- Phase-I: Geant4 Kernel extension (already available in 11.3.beta) (M. Asai)
  - Use *G4SubEvtRunManager* and *G4WorkerSubEvtRunManager* for sub-event parallelism
  - Only the master run manage owns the primary event generator while *G4WorkerSubEvtRunManager* takes a vector of tracks for a specific subevent type
  - Add *G4SubEvent* and *G4SubEventTrackStack*
  - Support offloading pipelines to integrate external packages, e.g.,
    - Opticks/OptiX™ (Optical photon simulation), Adept(CERN) and Celeritas (US), (EM particle transport)
  - Support parallel tasking for a single large problem (e.g., processing a big event)
- Phase-II: Extension for optimal use of sub-event parallelism
  - Support specialized physics lists (data) and/or detector construction (geometry) dedicated to sub-tasks
  - Support trajectory and visualization for tracks inside sub-tasks

Specialized or Custom Tracking Manager for Offloading

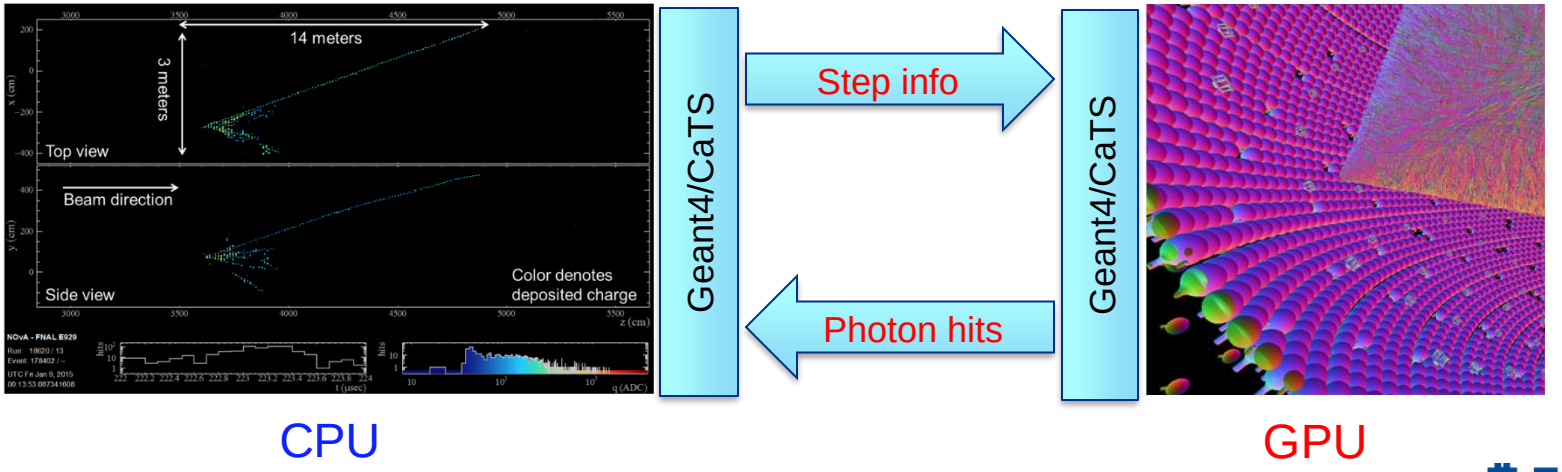
- *G4VTrackingManager*: Interface class for implementing a custom tracking manager that is specialized for stepping one or a small number of particle types. Key virtual methods are
  - BuildPhysicsTable(const G4ParticleDefinition&){}
  - HandOverOneTrack(G4Track* aTrack) = 0;
  - FlushEvent(){}



Examples

Optical Photon Simulation on GPU

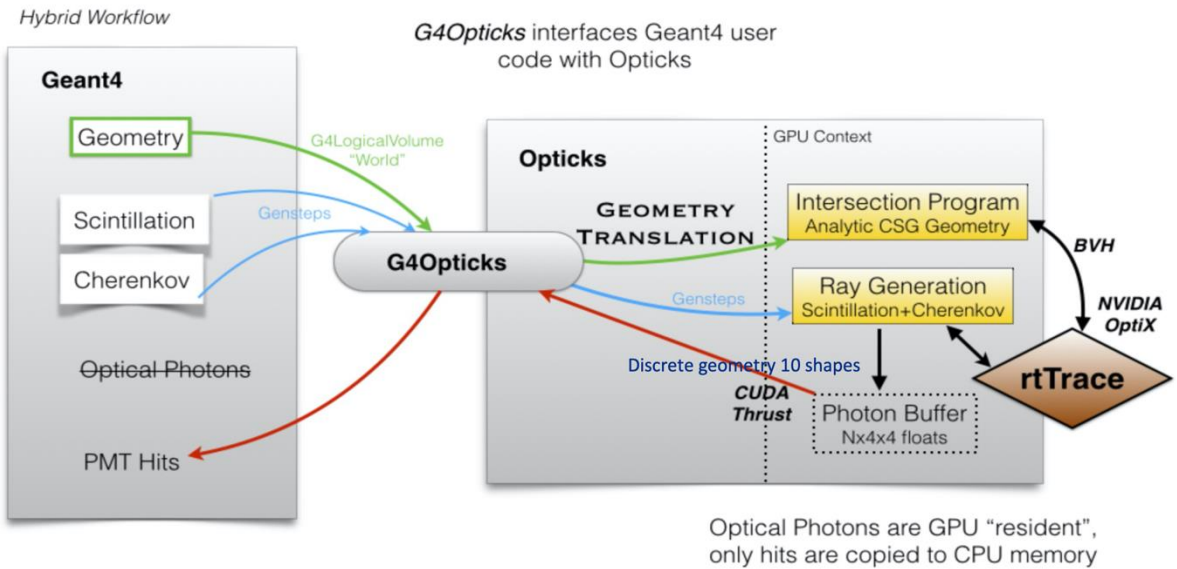
- Optical photons are copiously produced in optical materials (ex. 45,000 for 1 MeV energy loss in LAr) and most neutrino and dark matter search experiments need an accurate simulation of optical photons.
- A full optical photon simulation for a charged particle of GeV-level energy requires significant computational resources (CPU and memory) → usually approximation by using lookup tables.
- A solution: GPU accelerated optical photon simulation using ray tracing (e.g., NVIDIA OptiX)



Example 1: Opticks/OptiX + CaTS

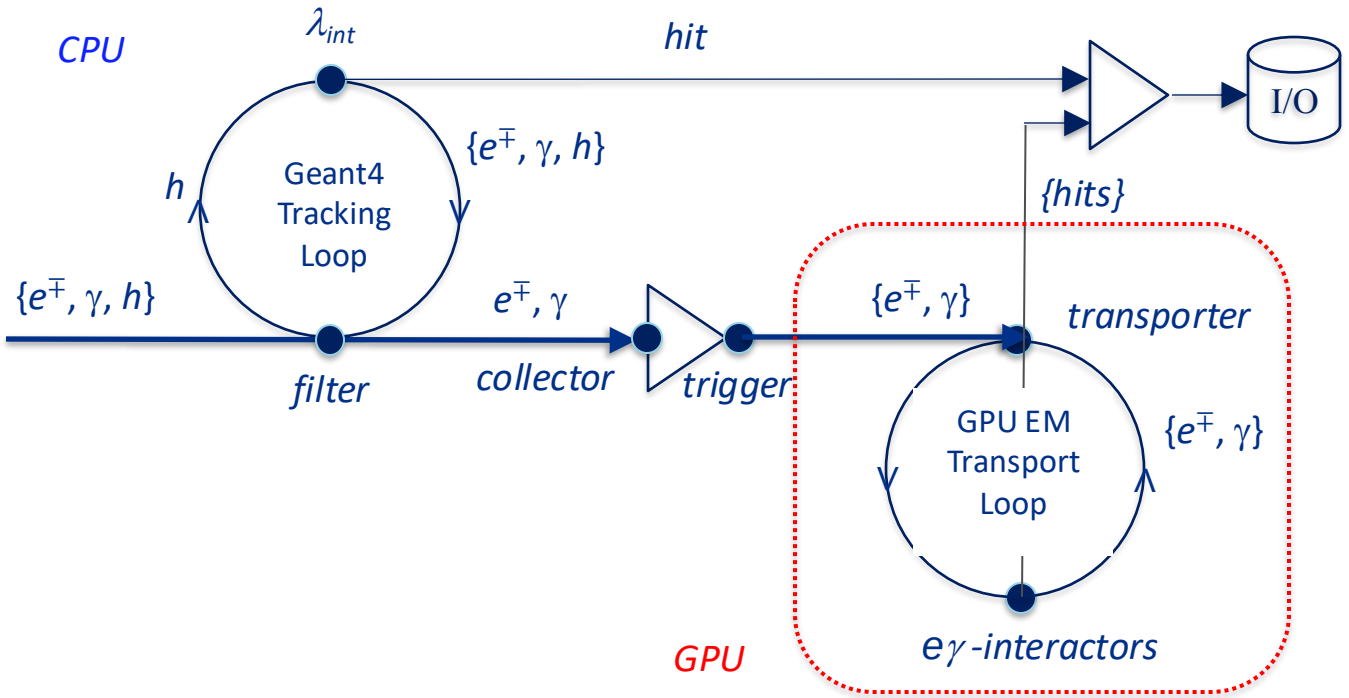
- Opticks is an open-source project that accelerates optical photon simulation by integrating NVIDIA GPU ray tracing, accessed via NVIDIA OptiX (developed by Simon Blyth for DayaBay and JUNO experiments, https://bitbucket.org/simoncblyth/opticks/).
- CaTS is an advance Geant4 example with Opticks/OptiX (examples/advanced/CaTS)

Figure from Simon's presentation



Example 2: EM Particle Transport on GPU

- A hybrid workflow with selected tasks executed on GPUs
  - Example: a Geant4 application which offloads EM particle transport on GPU using user actions.



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}
  - Transfer sensitive hits from devices 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)

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 systems accelerating end-to-end workflows
  - Simulation and modeling
  - AI Training & Inference
  - Experiment data analysis

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