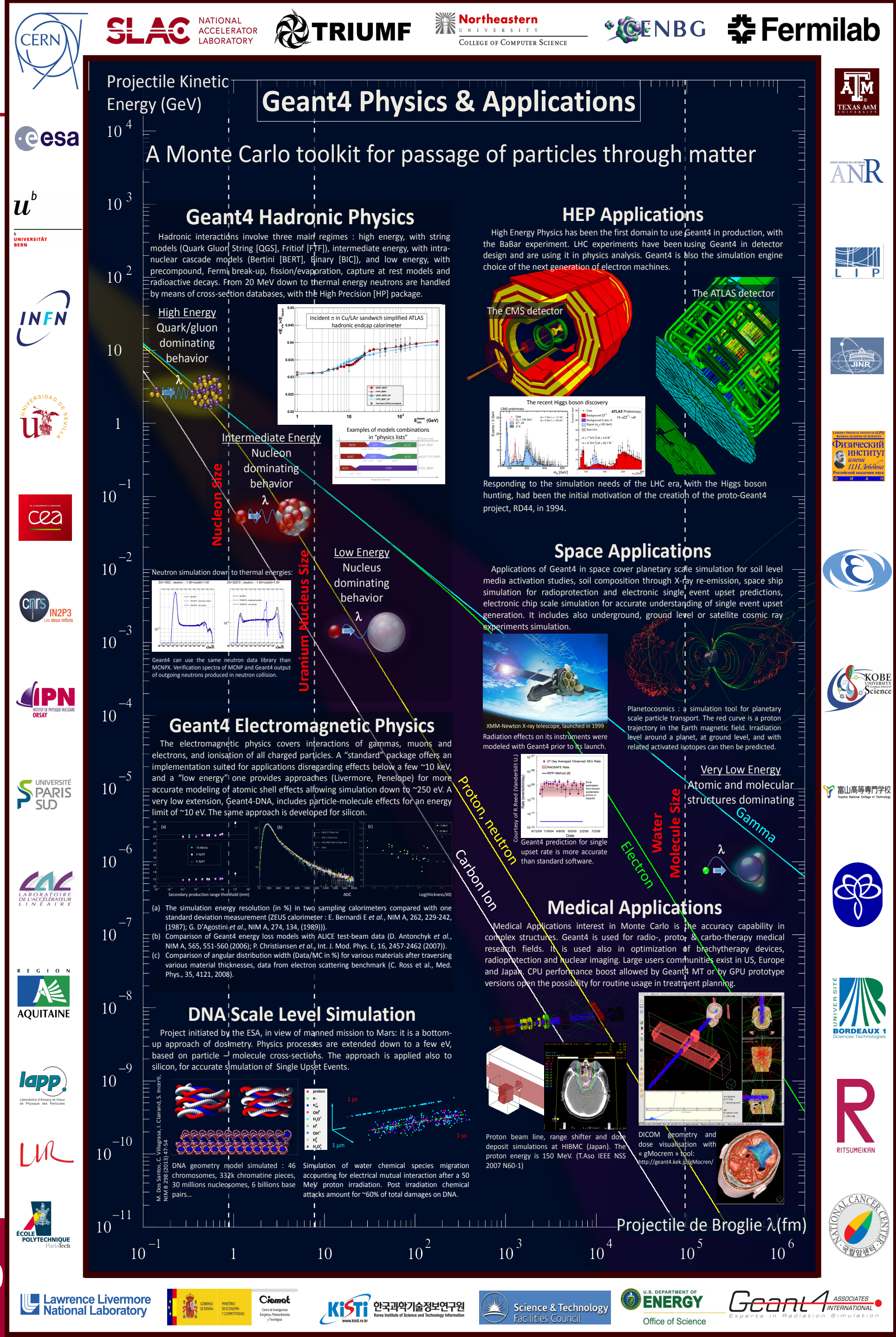
Geant4 in 2030
~ a personal view ~

Makoto Asai (SLAC)
11/09/2021
@ Software & Computing Roundtable

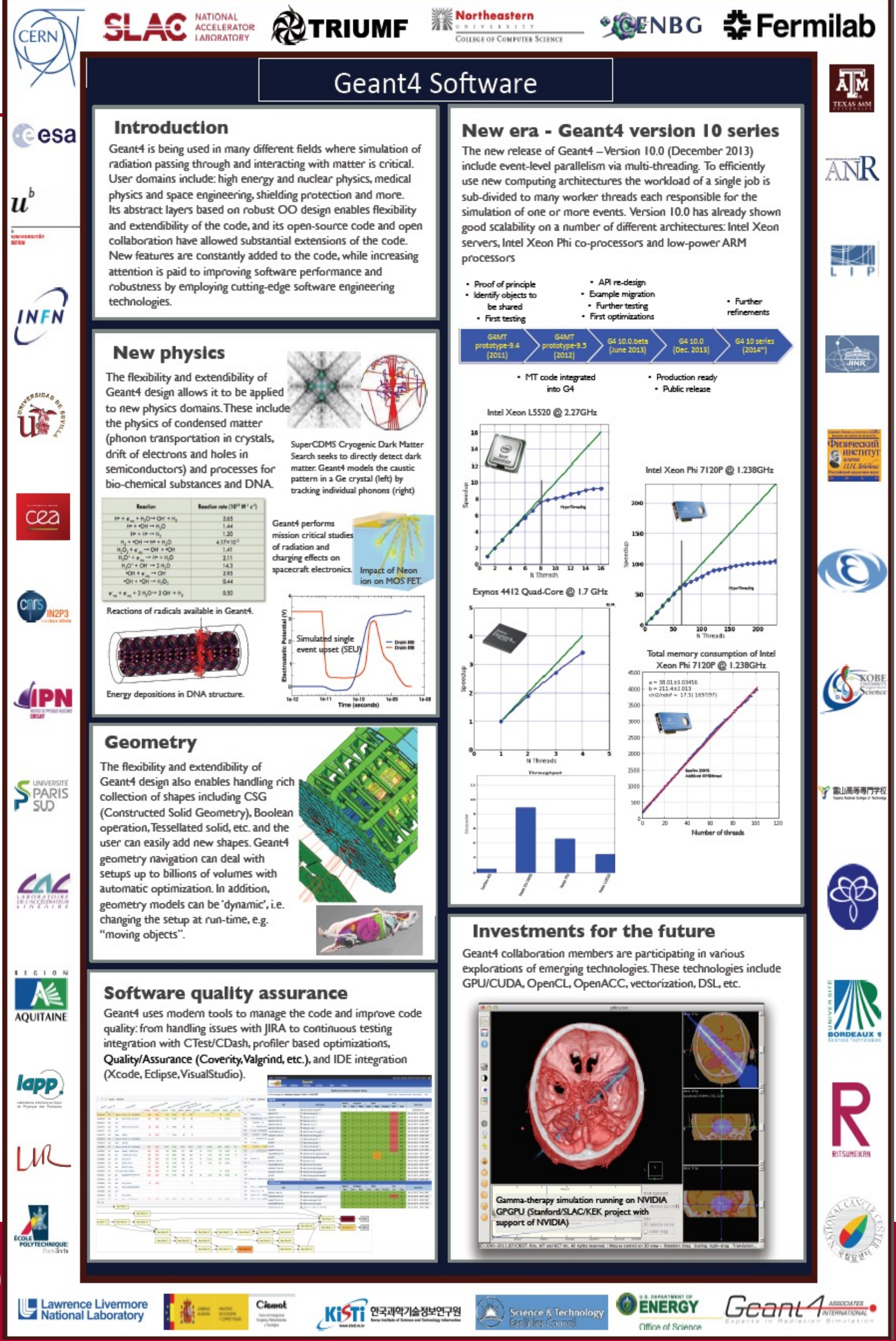
Preamble

- Geant4 will likely remain as the main detector simulation engine for HENP in 2030's.
  - Let me try to present my personal view on how Geant4 would/should evolve in coming 10 years to meet the requirements of 2030's.
- This is a personal view.
  - In this presentation, I'm not representing any institution or collaboration.
  - And I won't take any responsibility on ensuring these evolutions to happen unless I'm paid for.



Contents

- Who am I?
- How Geant4 is evolving
- Geant4 as a detector simulation engine
- Hybrid computing – what GPU should do
- How Geant4 should evolve in next 10 years



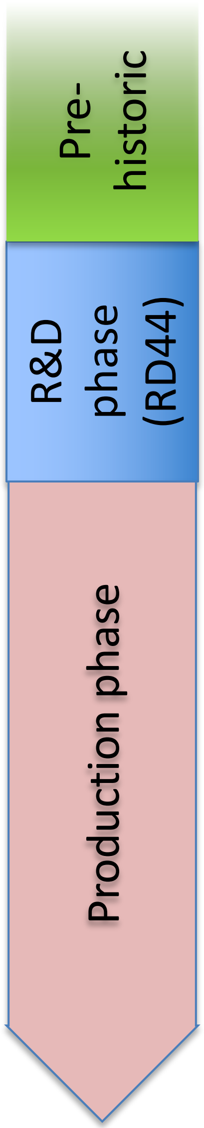
Who am I?

- Under-grad
  - Ion implantation simulation
- Graduate student
  - CERN NA23 : worked with original GEANT developers
- Solenoidal Detector Collaboration @ SSC
  - Detector simulation group
  - Early discussions on GEANT evolution
- Geant4 R&D
  - Not because of OO/C++ knowledge but long history with GEANT
- ATLAS @ LHC
  - CHAOS (CompreHensive Atlas Object-oriented Simulator) project
- Moved to SLAC
  - Geant4 Deputy Spokesperson (2001-2010)
  - Geant4 Spokesperson (2010-2020)

Learned how to read Fortran code written in French

Sudden death of SSC
(and postponement of LHC)
gave us time for Geant4 R&D

Geant4 History



- Early discussions, for example at CHEP 1994 @ San Francisco
  - "Geant steps into the future" R. Brun *et al*.
  - "Object oriented analysis and design of a GEANT-based detector simulator" K. Amako *et al*.
- Dec '94 – R&D project start
- Apr '97 – First alpha release
- Jul '98 – First beta release
- Dec '98 – First Geant4 public release - version 1.0

- **Several major architectural or design revisions**
  - **E.g. STL migration, cuts per region, parallel worlds, multithreading**

- Dec 4th, '20 – Geant4 version 10.7 release
  - Jun 11th, '21 – Geant4 10.7-patch02 release ← Current version
- We currently provide one public release every year.
  - Dec 10th, '21 – Geant4 11.0 release ← Major release : introducing Task-based parallelism

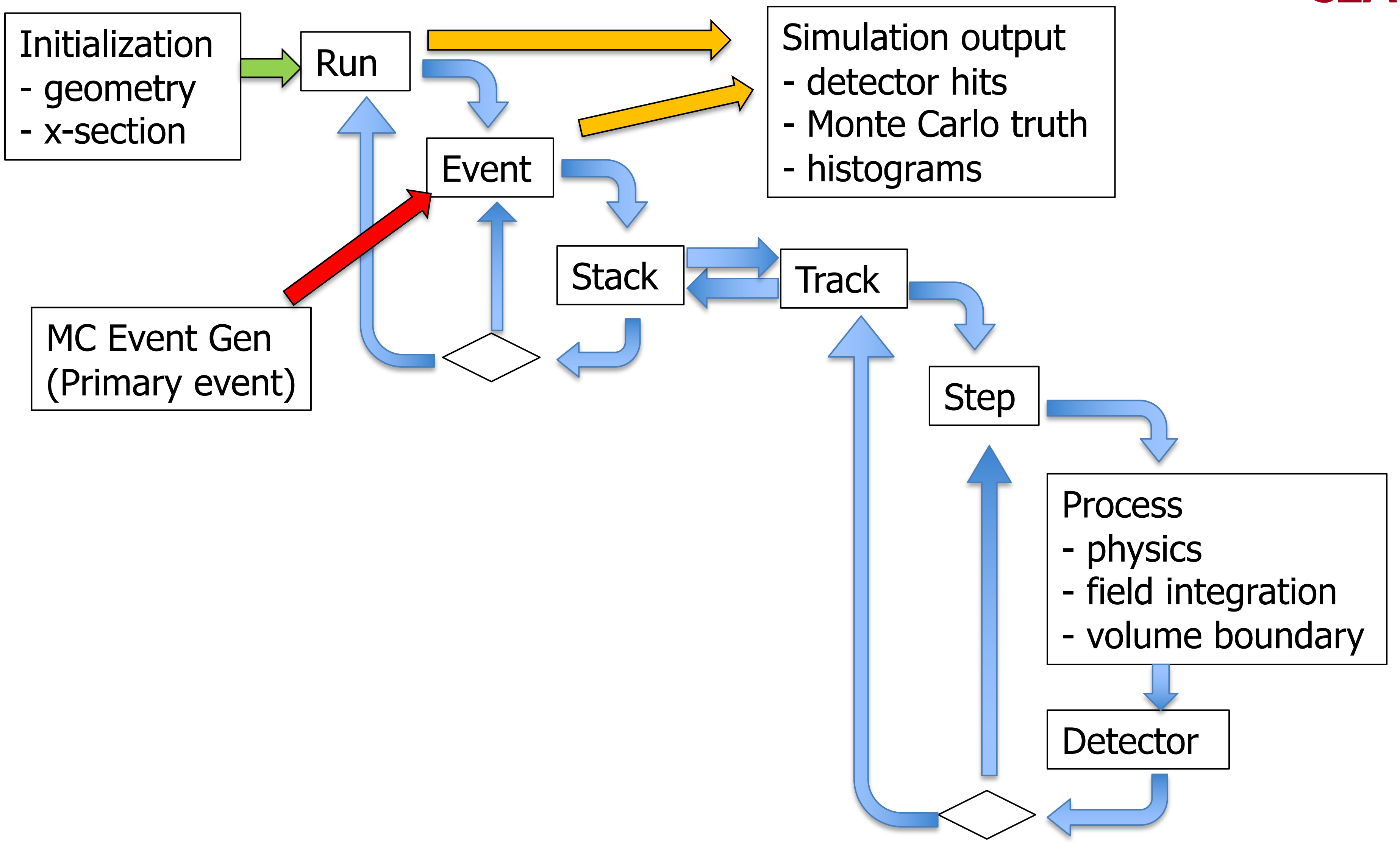
Geant4 is widely adapted because…

- Of course, wide coverage of accurate physics models, powerful geometry package, nice GUI and Visualization options, etc., are provided.
- It is a toolkit, so it was easily adapted or wrapped into user's framework.

- But also, stability of the major API's is an important aspect.
  - New architectures were well encapsulated and mostly invisible to the user.
  - For example, when Geant4 introduced multithreading, migration of user's code was extremely simple and straightforward.
    - Except making user's code itself thread-safe…
- Stability of major API's also incubates new developments / contributions.
  - New physics models, new geometry modules, new GUI and Vis drivers, etc.

- Most cases, new functionality based on new architecture comes with additional API's rather than changed API's.
- Geant4 is not a turn-key application, and should not be a turn-key application in 2030's. Thus, user must build an application on top of Geant4.
  - Do not create too many layers of thick wrappers. Better having a single thin wrapper for the sake of quick catching up of new interfaces.

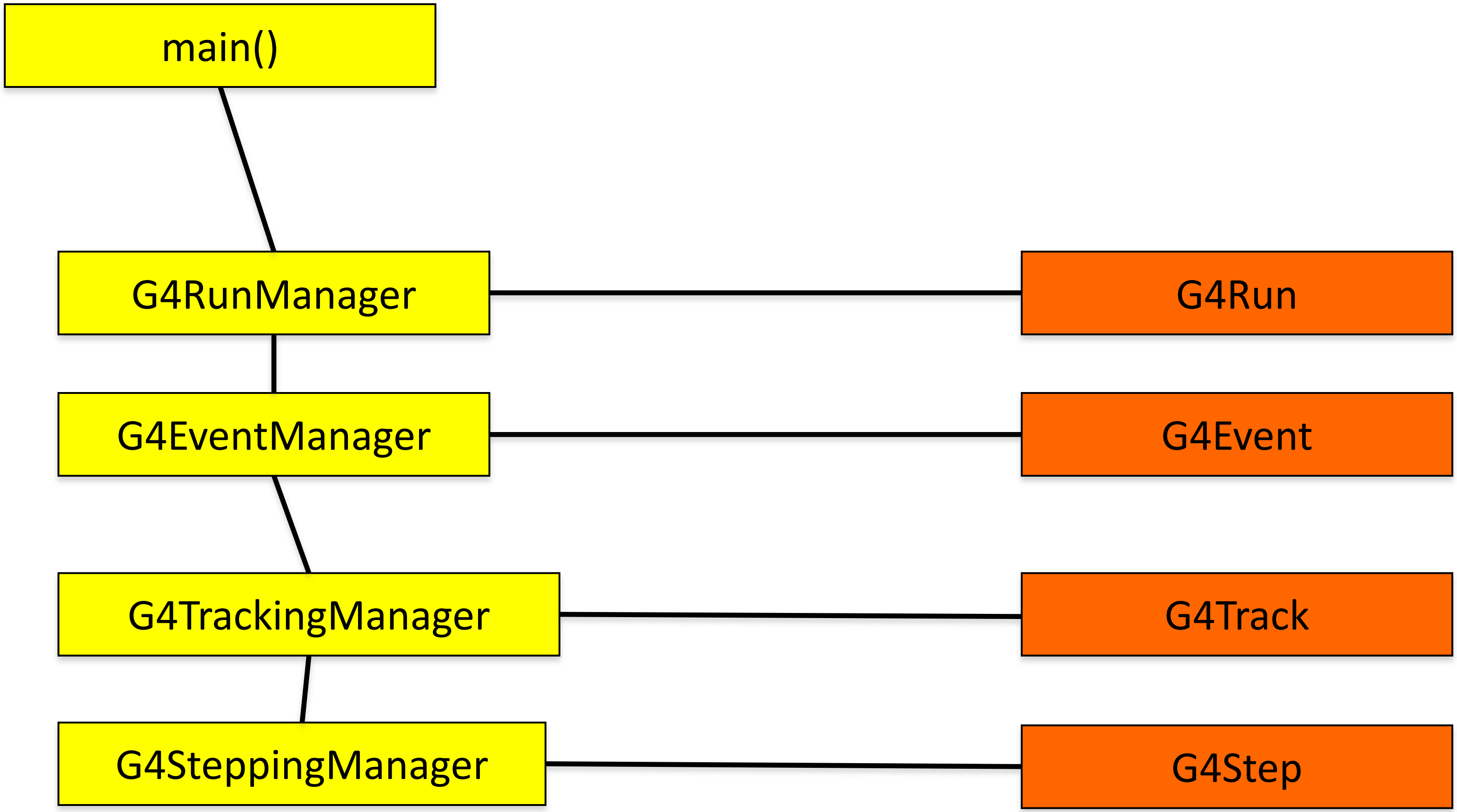
Geant4 as a detector simulation engine



Geant4 evolutions in parallelization

1. Sequential mode : original since Geant4 v1.0
   - Single core (thread) does everything

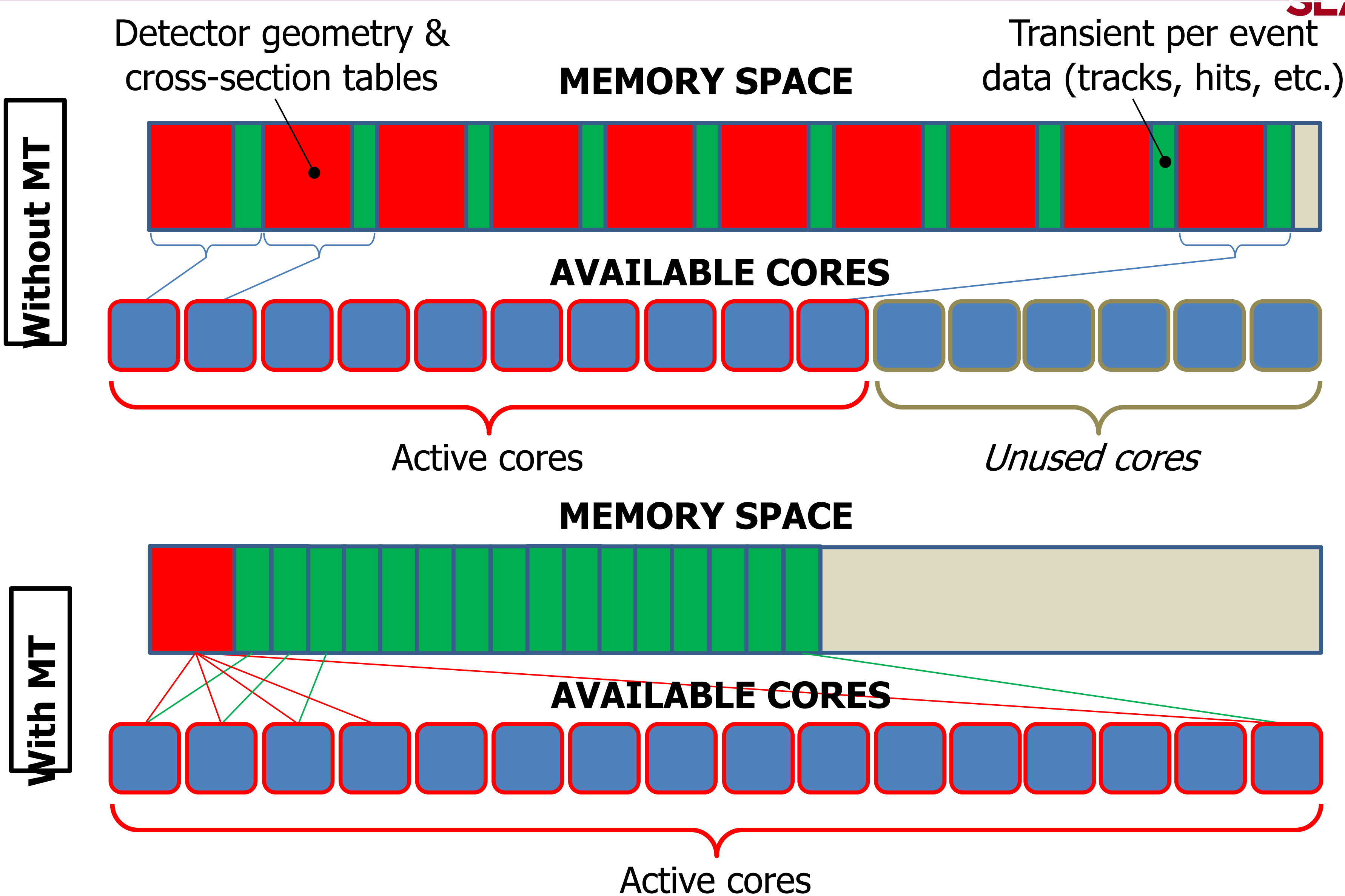
Sequential mode



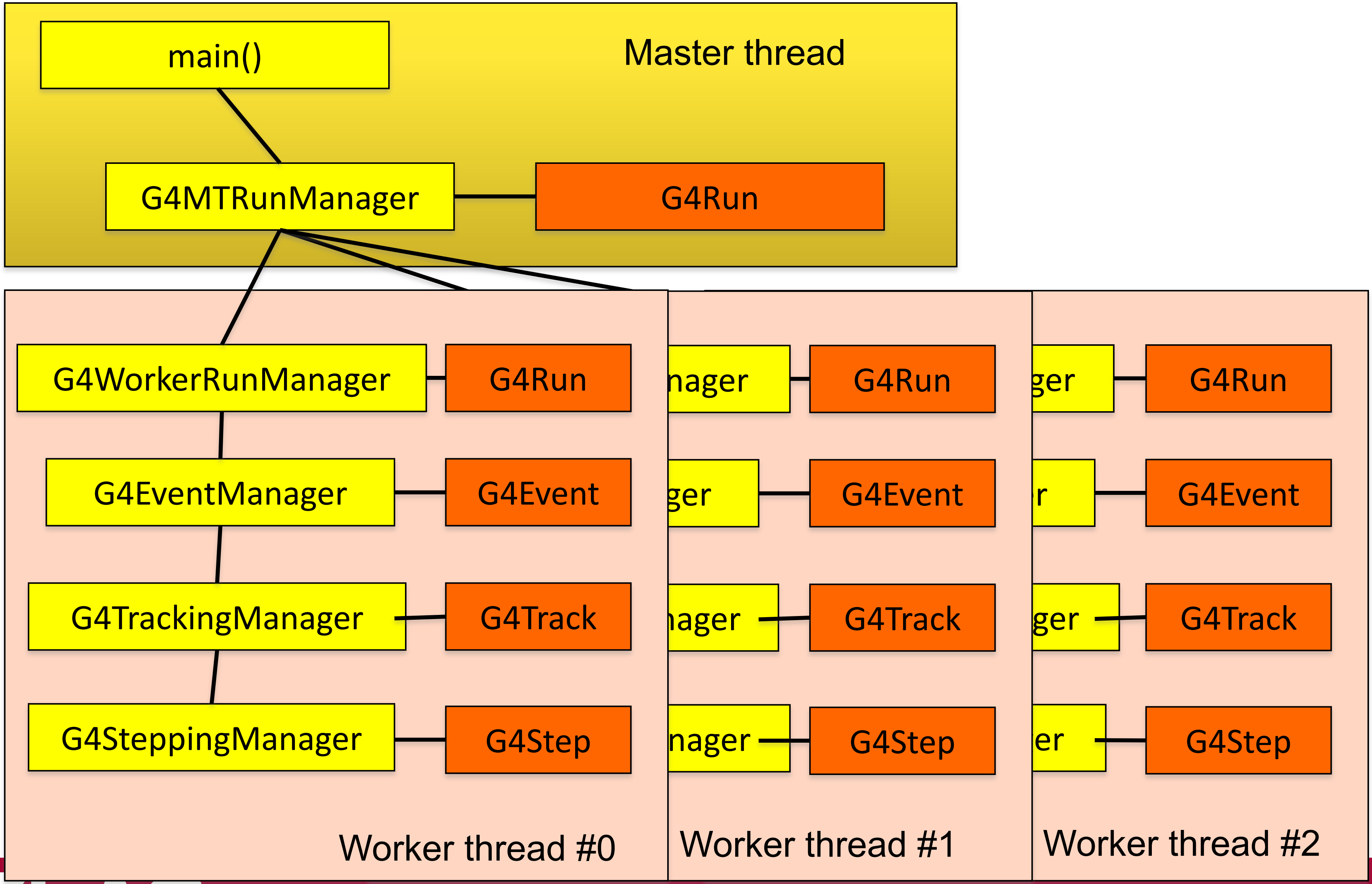
Geant4 evolutions in parallelization

1. Sequential mode : original since Geant4 v1.0
   - Single core (thread) does everything
2. Multithreaded event-level parallelism mode : since Geant4 v10.0 (Dec.2013)
   - Taking the advantage of independence of events, many cores (threads) process events in parallel
   - Geometry / x-section tables are shared over threads

Geant4 runs in multithreaded mode

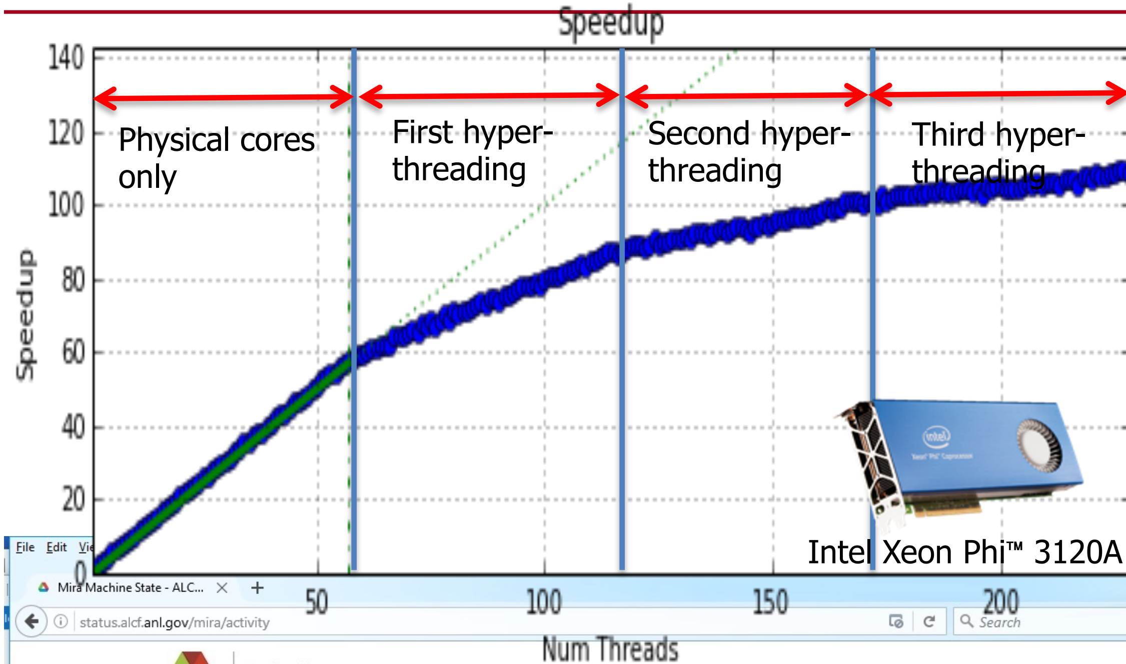


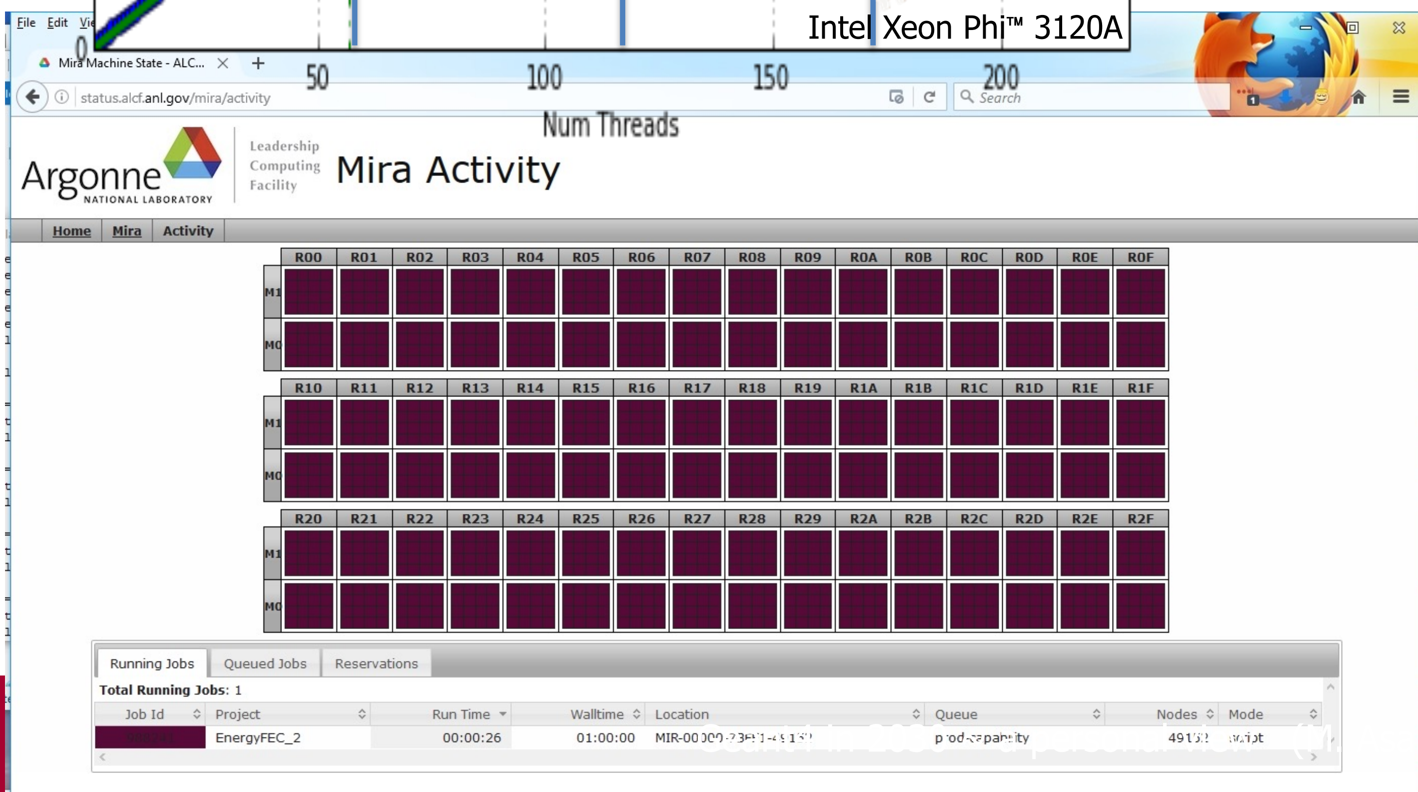
Multithreaded mode



Multithreaded mode in Geant4 v.10 series

SLAC



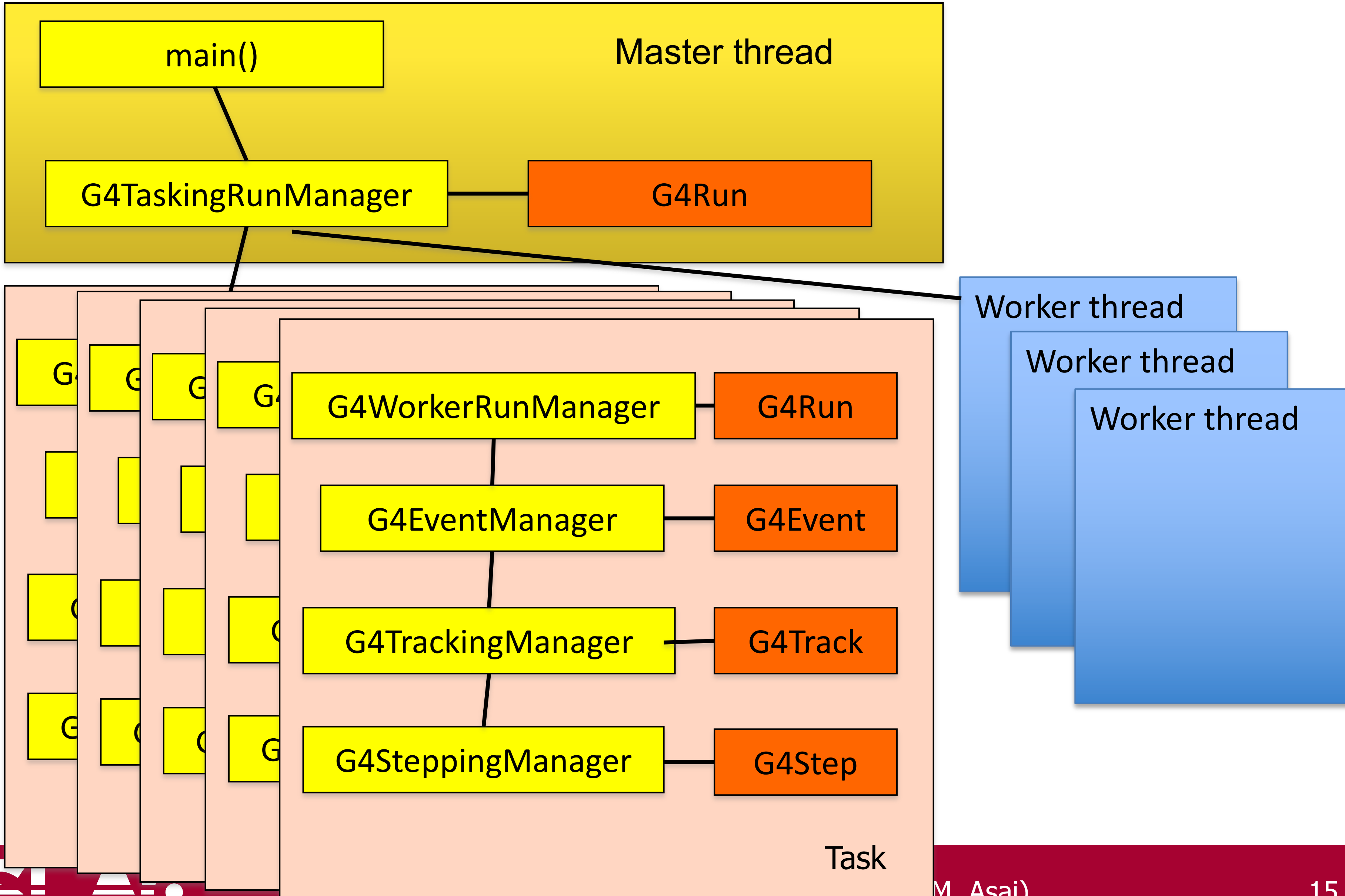


- Geant4 has successfully run with a combination of MT and MPI on Mira Bluegene/Q Supercomputer (@ANL) with all of its 3 million threads
  - Full-CMS geometry & field
  - Demonstrated that spare time of Mira was enough to cover Run-2 LHC simulation needs
  - I/O is the limiting factor to scale large concurrent threads:
    - » Granular input data files, output data /histograms, etc.

Geant4 evolutions in parallelization

1. Sequential mode : original since Geant4 v1.0
   - Single core (thread) does everything
2. Multithreaded event-level parallelism mode : since Geant4 v10.0 (Dec.2013)
   - Taking the advantage of independence of events, many cores (threads) process events in parallel
   - Geometry / x-section tables are shared over threads
3. Task-based event-level parallelism mode : since Geant4 v11.0 (Dec.2021)
   - Decoupling task (event loop) from thread
   - More flexible load-balancing

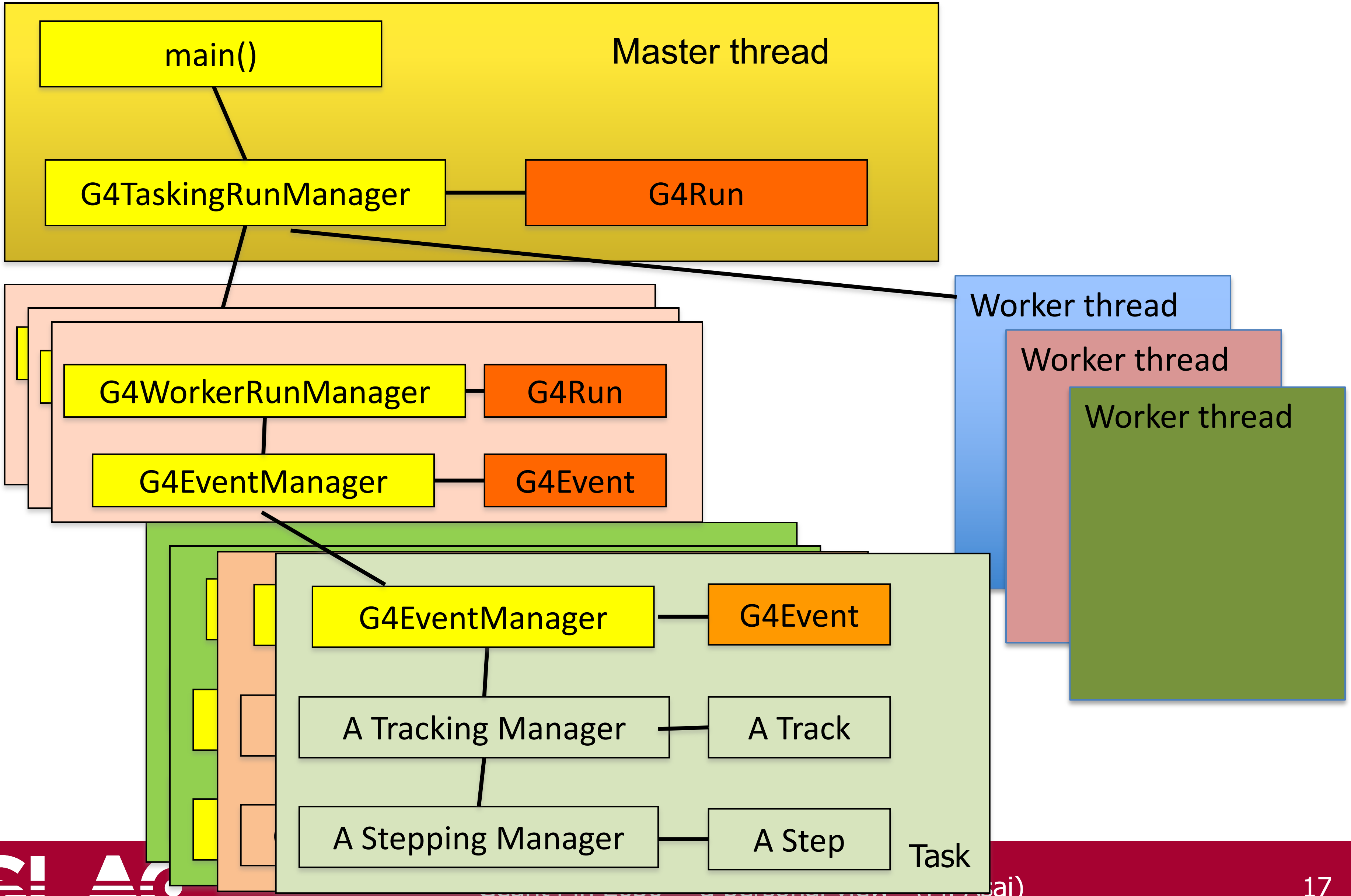
Task-based parallel mode



Geant4 evolutions in parallelization

1. Sequential mode : original since Geant4 v1.0
   - Single core (thread) does everything
2. Multithreaded event-level parallelism mode : since Geant4 v10.0 (Dec.2013)
   - Taking the advantage of independence of events, many cores (threads) process events in parallel (event-level parallelism)
   - Geometry / x-section tables are shared over threads
3. Task-based event-level parallelism mode : since Geant4 v11.0 (Dec.2021)
   - Decoupling task (event loop) from thread
   - More flexible load-balancing
4. Task-based sub-event parallel mode : planned (Dec.2022~)
   - Split into sub-events and task
   - Sub-event :
     - Sub-group of primary tracks, or
     - Group of tracks getting into a particular detector component
   - Suitable for heterogeneous hybrid hardware

Sub-event parallel mode



Hybrid computing – what GPU should do

- Two different usages of GPU
    1. Full Monte Carlo detector simulation
    2. Fast simulation

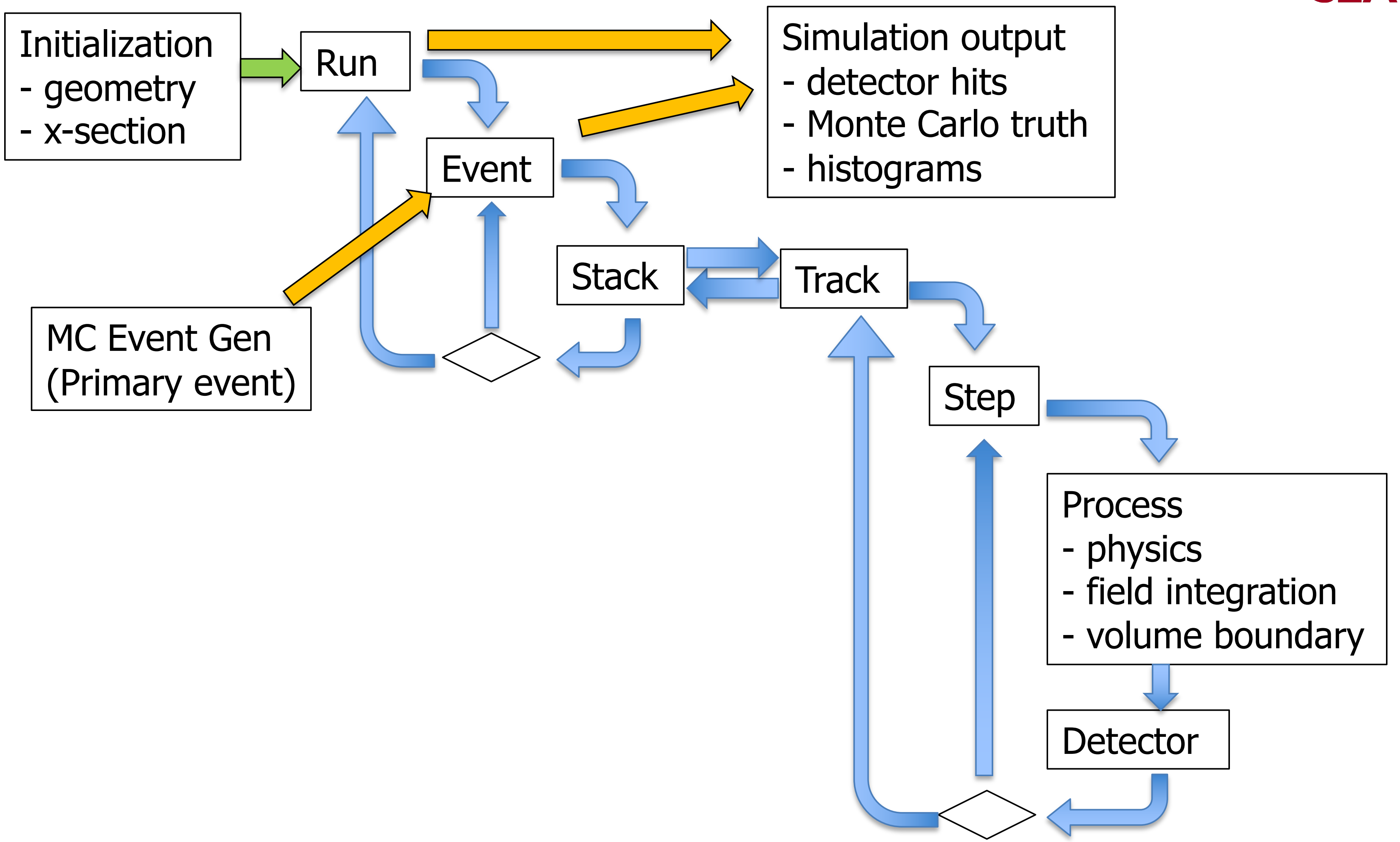
Monte Carlo simulation on GPU – lessons learned

- It is hopeless to port the entire Geant4 to a single process on GPU.
  - Each GPU process should have limited scope.
    - Physics coverage, particle type, geometry/material
    - E.g. optical photon transport, EM shower in calorimeter (w/o back splash or punch through)
- A task on GPU should behave like a blackhole.
  - The darker a task is, the better performance it has.
    - "Darker" means "less output".
  - Step/track should not be taken out.
    - Reshuffling tracks over tasks is the worst thing to do.
  - Minimize taking step/track information.
    - E.g. transferring output is a serious bottleneck.

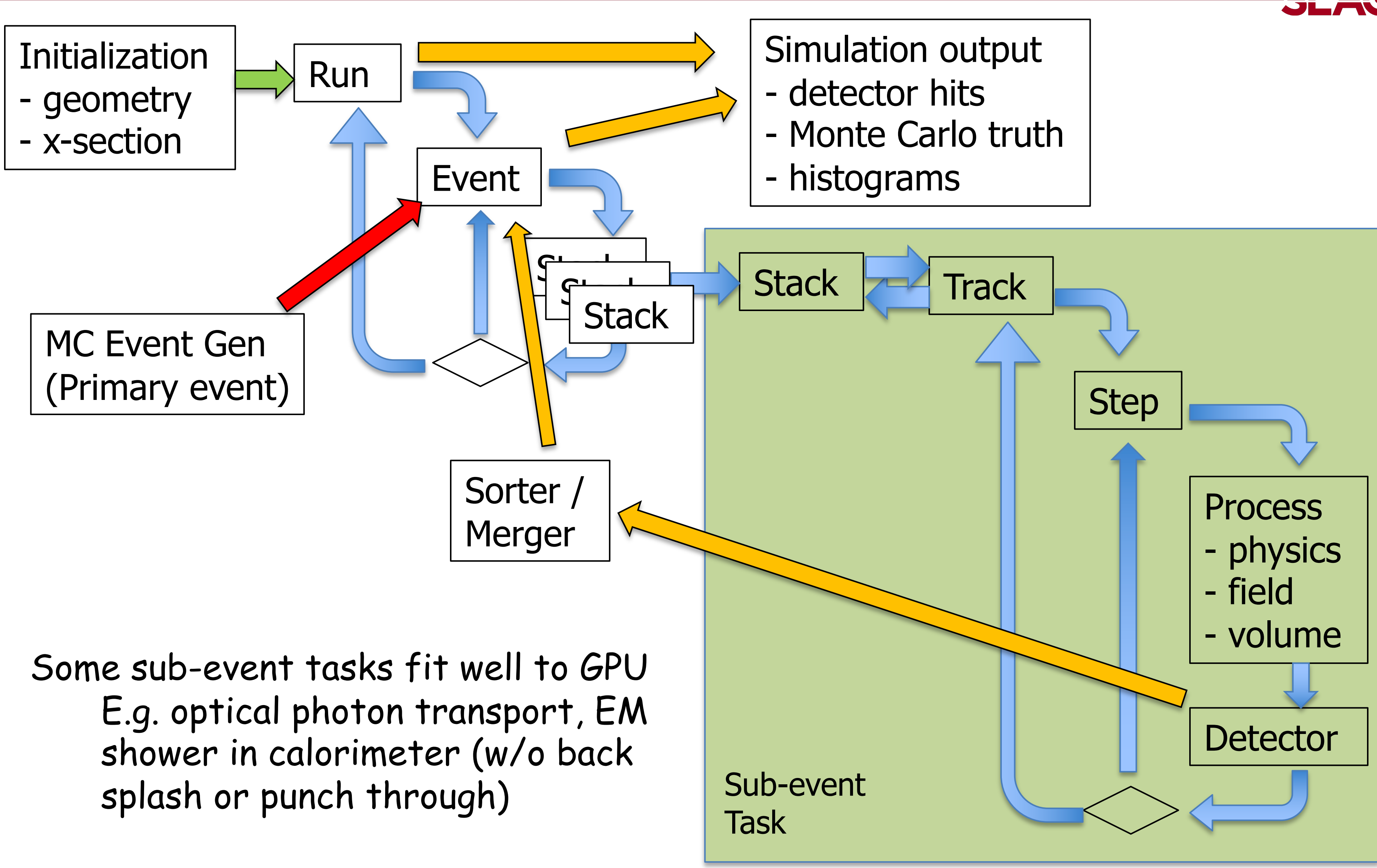
Side note:

Geant4 version 11.0 will include the first version of G4HepEm module, a complete set of EM physics specialized and optimized for HEP calorimeter, potentially ported to GPU.

Geant4 as a detector simulation engine

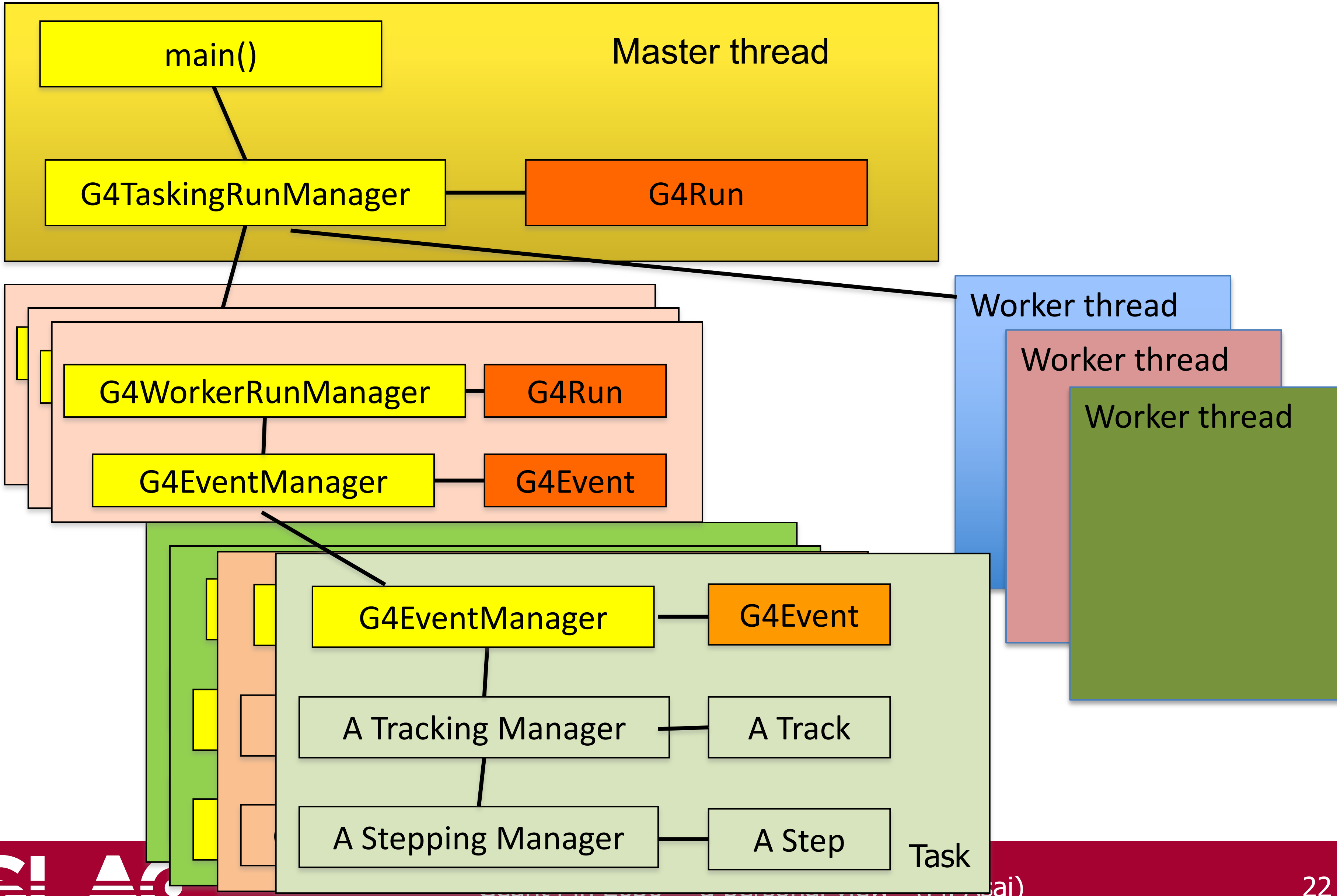


Geant4 as a detector simulation engine



Some sub-event tasks fit well to GPU
E.g. optical photon transport, EM shower in calorimeter (w/o back splash or punch through)

Sub-event parallel mode



Fast simulation with GPU

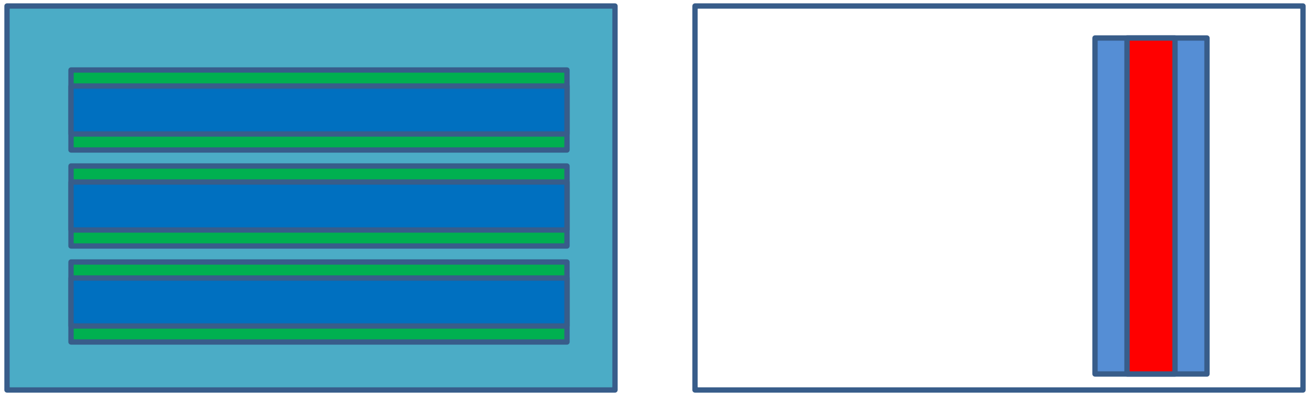
- Classical fast simulation such as Gflash or modern fast simulation powered by AI
  - Parameter optimization is an ideal usage of GPU (ML, in particular) *!!*
  - Parameters must be optimized / tuned for every detector
    - Start training as soon as detector hardware is fixed

- Need careful consideration for outgoing tracks.
  - Back splash or punch through track will be the bottleneck.

- Fast simulation must not be applied to a detailed geometry.
  - Lessons learned on Gflash shower parametrization
  - CMS successfully employed Gflash on its crystal calorimeter, while ATLAS couldn't use Gflash on its accordion calorimeter.
  - Geant4 allows more than one geometry descriptions
    - "Layered mass geometry"

Layered mass geometries in parallel worlds

- Parallel geometry may be stacked on top of mass geometry, allowing a user to define more than one worlds with materials (and region/cuts).
  - Track will see the material of top-layer, if it is null, then one layer beneath.
  - Alternative way of implementing a complicated geometry by Boolean operation



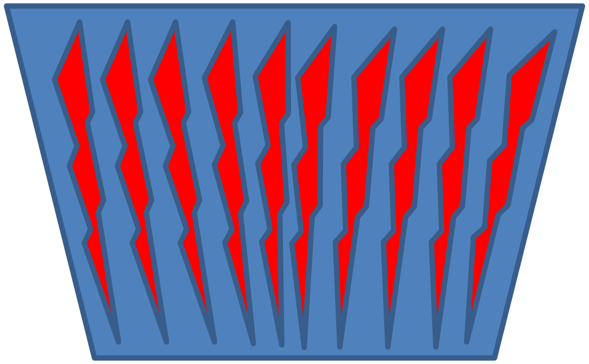
Mass world

Parallel world

Layered mass geometries in parallel worlds - continued

- A parallel world may be associated to some selected types of particles.
  - May define geometries of different levels of detail for different particle types
    - e+, e- and gamma do not see volume boundaries defined in the parallel world, i.e. their steps won't be limited
  - Shower parameterization such as Gflash may have its own simplified geometry

Geometry seen by e+, e-, γ



Geometry seen by other particles

Simulation challenges toward 2030

- Simulation throughput needs to increase by O(10-100)
  - Contribution of simulation to maintain the systematic uncertainty low
- New detector hardware comes with higher demands
  - e.g. highly granular calorimeter

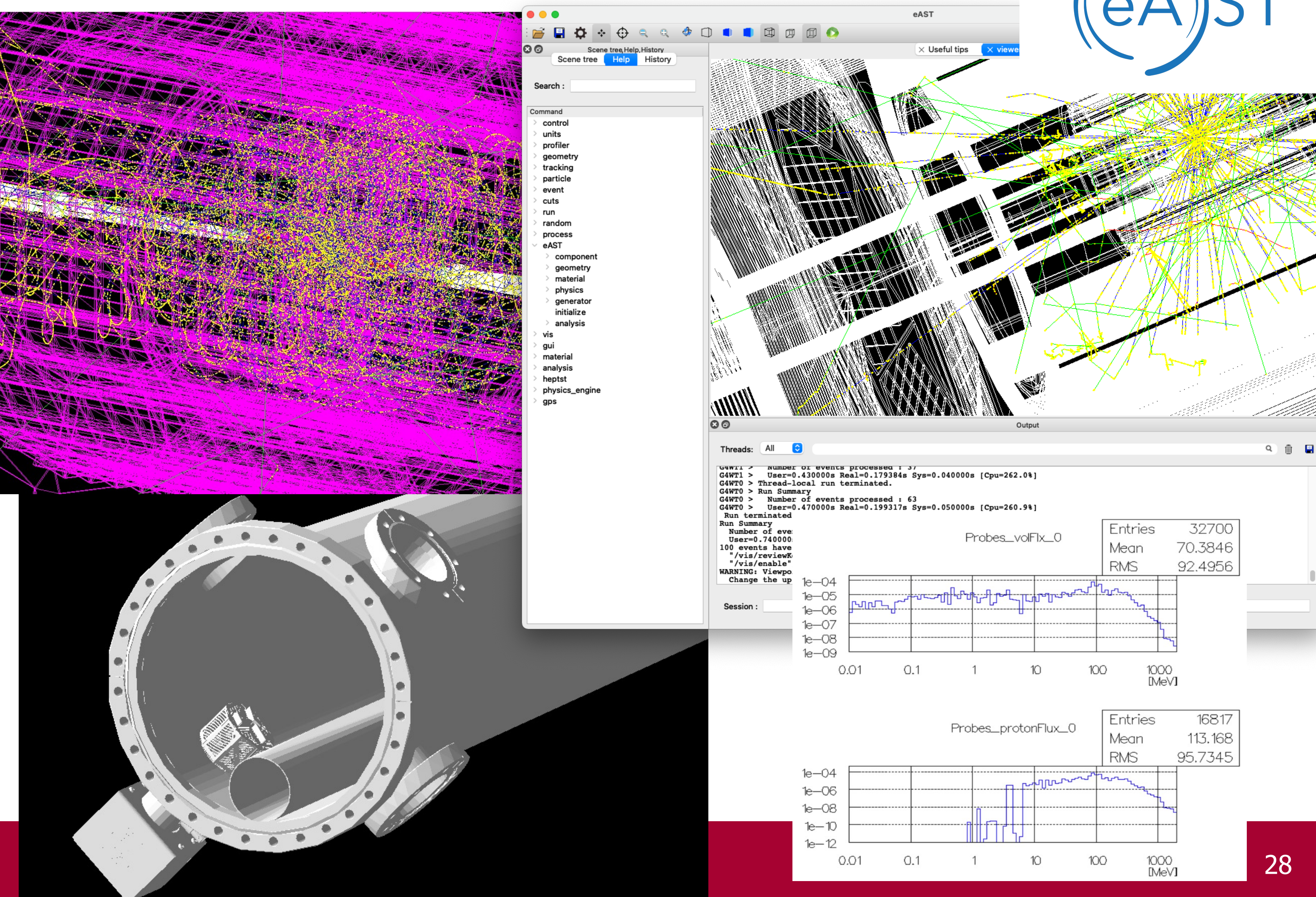
- Strategy
  - Detailed simulation as much as possible
    - On CPU
      - When sustainable or when no alternative
      - With creativity (e.g. stack management, layered mass geometry, event biasing, …)
    - On GPU
      - Combining variety of single-purpose, optimized tasks
      - Output must be minimized
  - Fast simulation as much as needed
    - On CPU
      - Parameter optimized by GPU
    - On GPU
      - AI-based fast simulation

eAST (eA Simulation Tool)

- eAST is a **turn-key application** as well as a **toolkit** for Electron-Ion Collider simulation studies built on top of Geant4.
- Requirements:
  - ability to **reuse existing simulation works**
  - ease of **switching detector options** with comparable levels of detail
  - ease of switching between **full Monte Carlo and fast simulation** at component level
  - ability to **integrate beam test setups** for physics validation and optimization of fast simulation parameters

  and
  - thinnest wrapper for direct use of native Geant4 functionalities as much as possible
    - for timely catchup of Geant4 evolutions
    - for **leveraging new and rapidly evolving** computing architectures
- Strong tie-up
  - EIC user community
  - eAST development team
  - Geant4 core developers

eAST (eA Simulation Tool) - https://eic.github.io/east/



To sum up

- This is a personal view.
- Geant4 will likely remain as the main detector simulation engine for HENP experiments in 2030's.
  - And also for training AI/ML
- There are plenty of opportunities in GPU-based detector simulation.
  - Both in full Monte Carlo and fast simulations
  - Each GPU process should have limited scope.
  - Each GPU process should make minimal output.
- Geant4 is making steady steps to meet requirements.
  - As a simulation engine that runs on heterogeneous computing architecture.