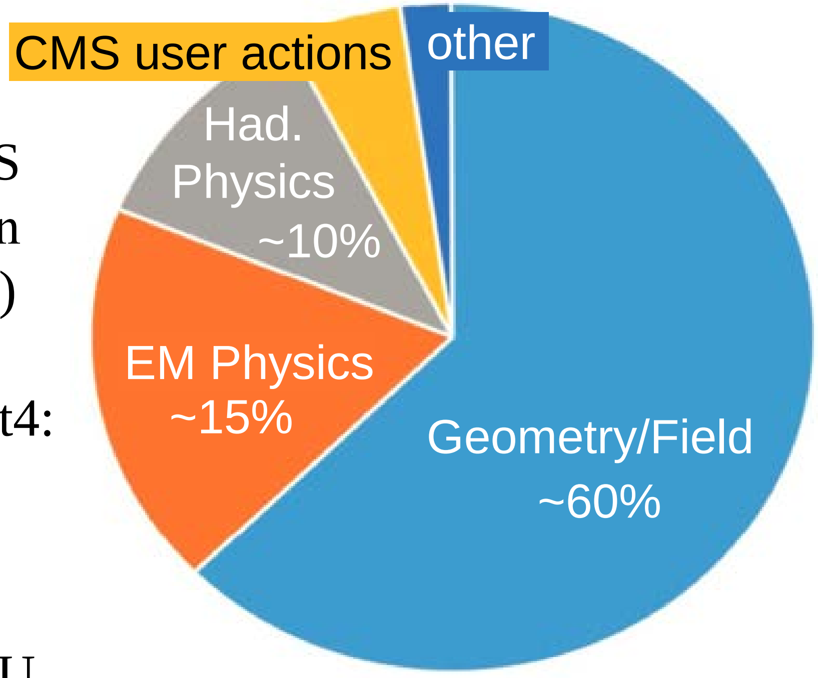
Current and Future Performance of the CMS Simulation

Kevin Pedro (FNAL)
on behalf of the CMS Collaboration
July 9, 2018

Overview

- CMS full simulation uses **Geant4**
- Sim is 40% of total CPU time used by CMS
  → most expensive "step" in MC production
  (vs. generation, digitization, reconstruction)
- Largest contributors to CPU usage in Geant4: geometry, magnetic field, EM physics
- CMS has implemented numerous technical options and approximations to improve CPU usage in the simulation
- Continue to explore new options and improvements



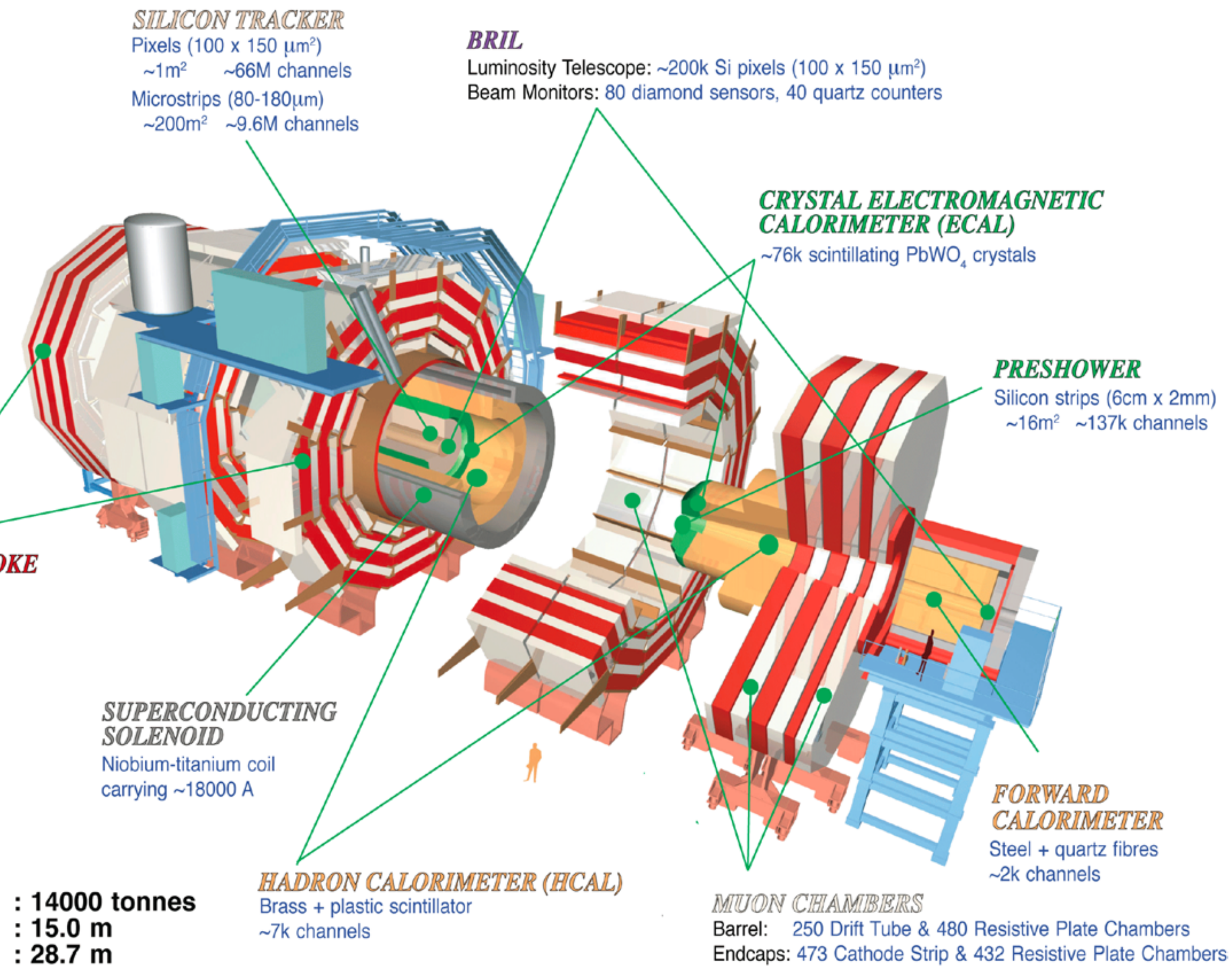
Geant4 10.0p02
(similar in other versions)

➢ Including **GeantV**, vectorized transport engine (currently in development)

The CMS Detector

"Phase 0"

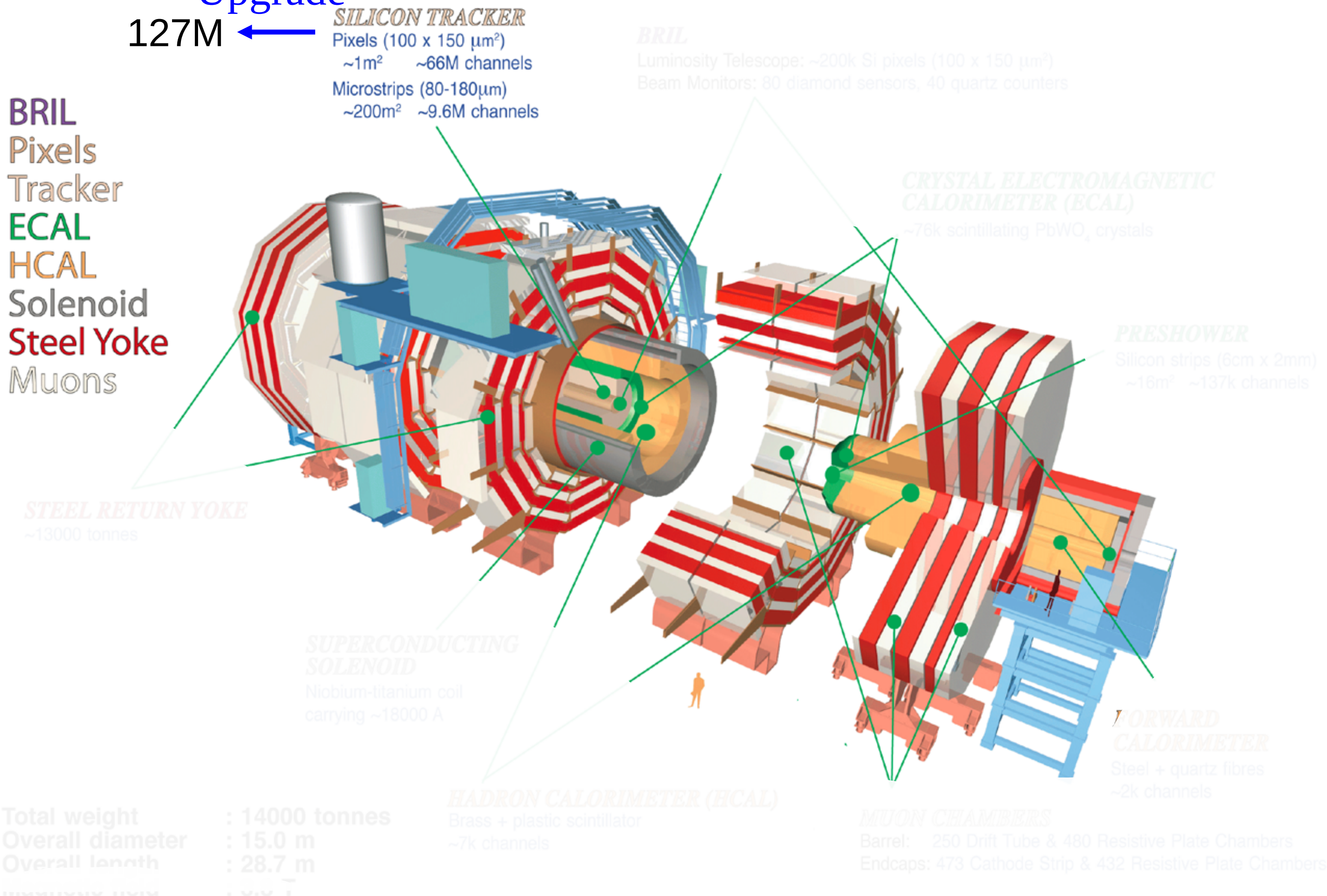
BRIL
Pixels
Tracker
ECAL
HCAL
Solenoid
Steel Yoke
Muons



Total weight	**: 14000 tonnes**
Overall diameter	**: 15.0 m**
Overall length	**: 28.7 m**
Magnetic field	**: 3.8 T**

The CMS Detector

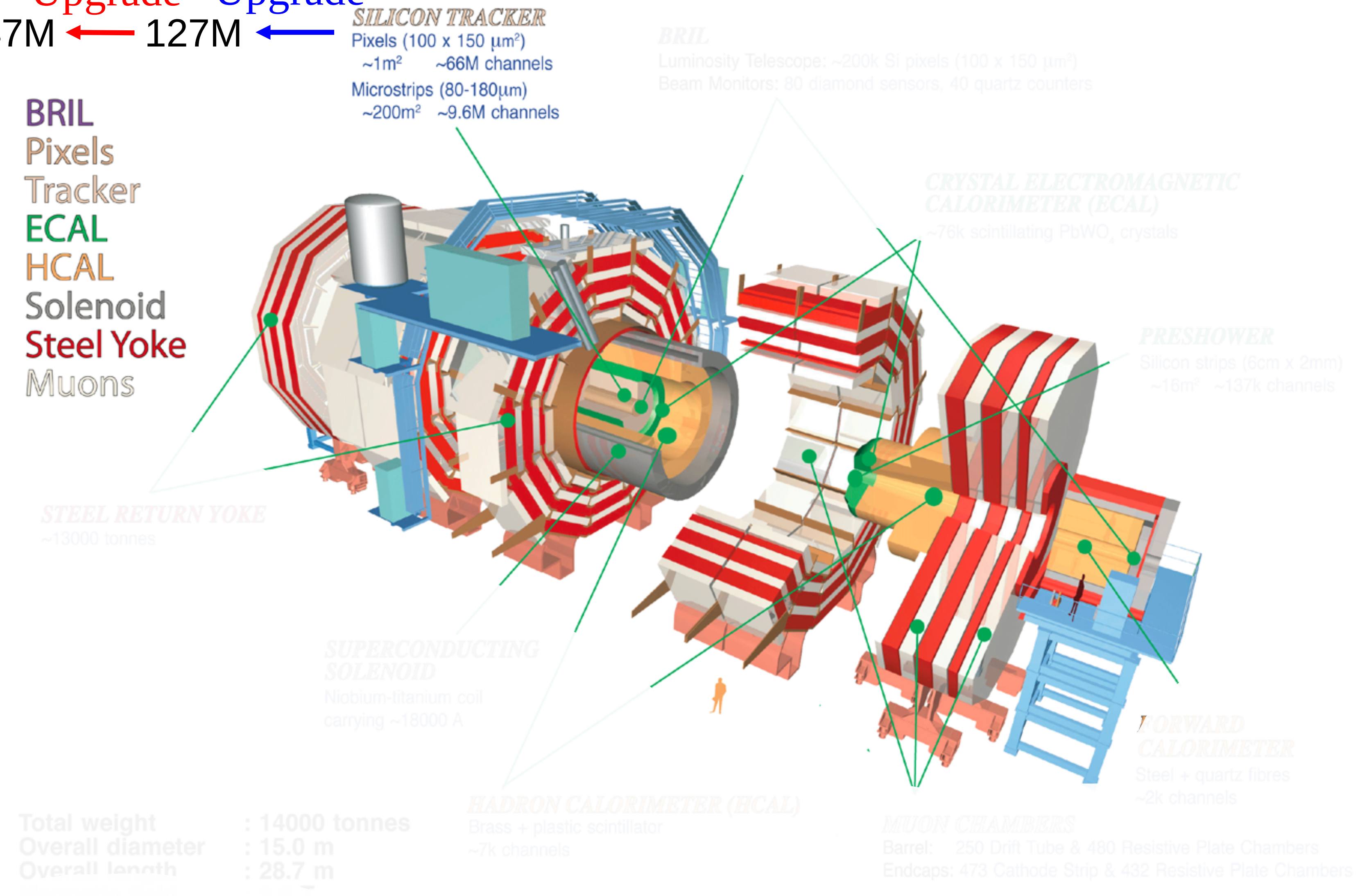
Phase 1 Upgrade



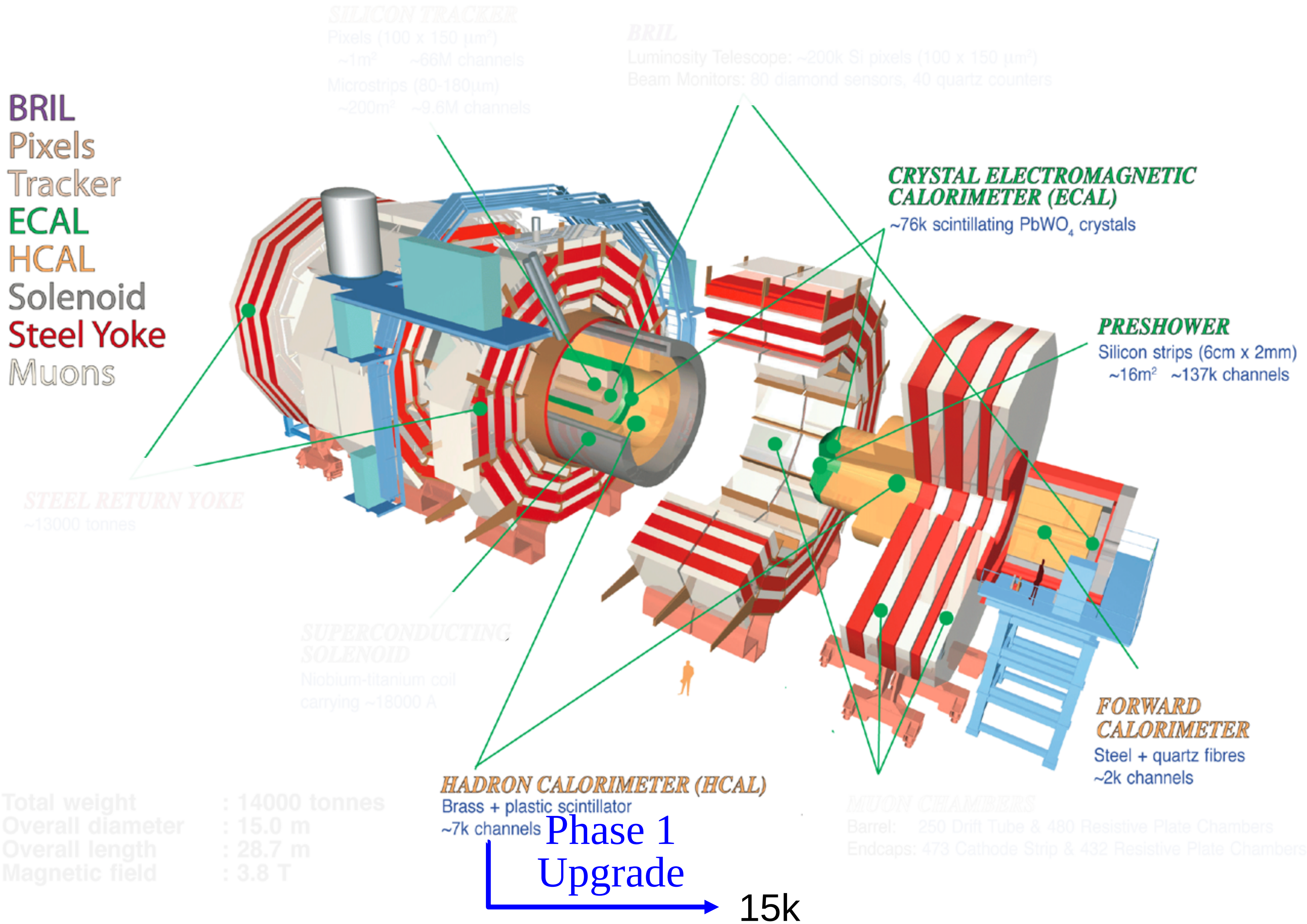
The CMS Detector

Phase 2 Upgrade | Phase 1 Upgrade

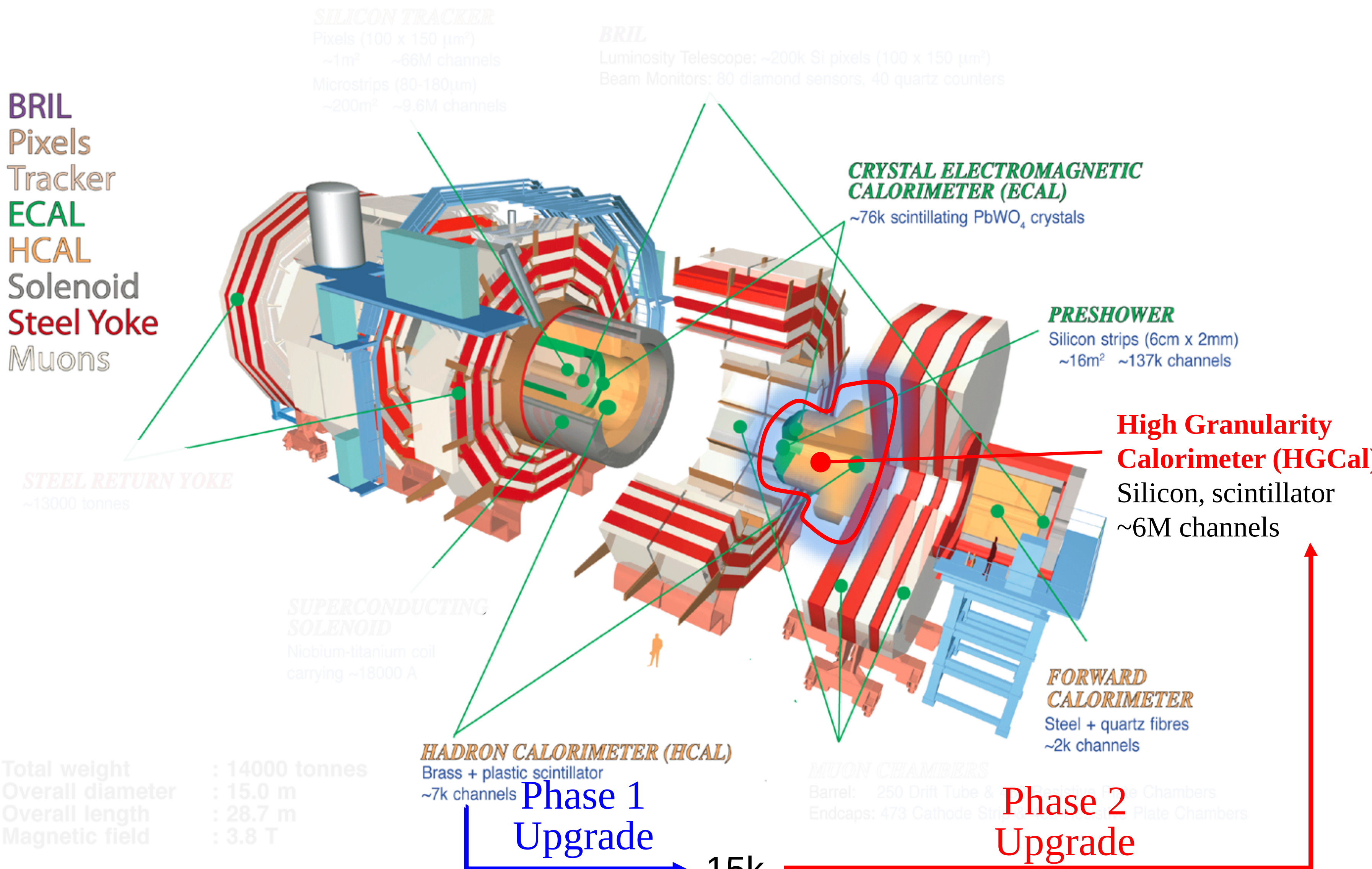
1947M ← 127M ←



The CMS Detector

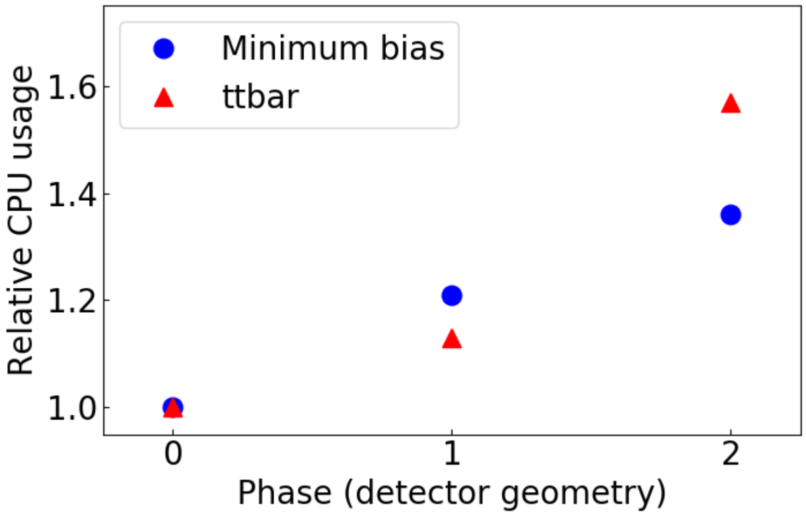


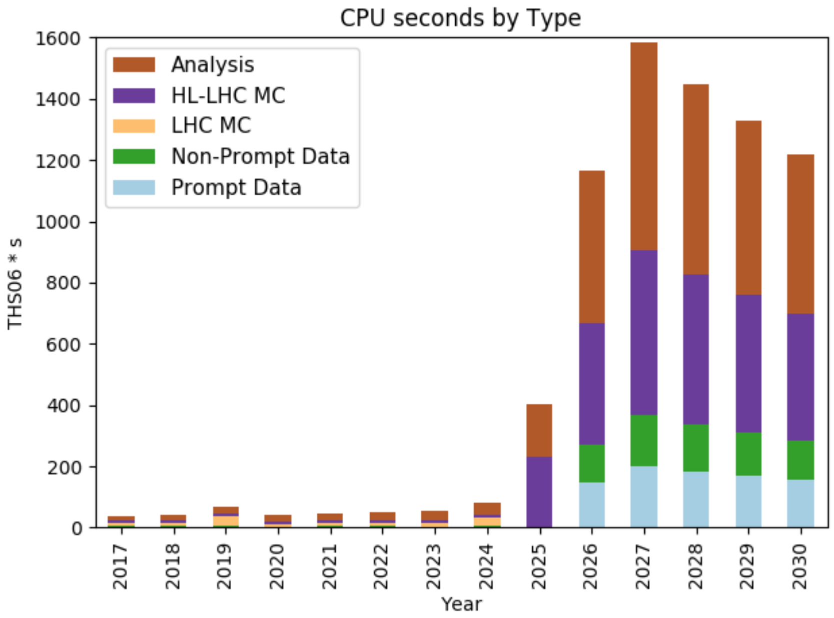
The CMS Detector



Challenges of Phase 2

- CMS Phase 0 and Phase 1 simulation geometries have 2.1 million elements
- Phase 2 geometry has 21.9 million elements: 10×(Phase 1)
- This translates to an increase in CPU time for simulation





- Simulate more events to keep up with HL-LHC data volumes: 10×(Phase1)
- May also need to improve accuracy of physics lists to simulate HGCal
- Reconstruction will take longer due to high pileup and granular detectors

➢ Need more events, more accuracy, in more complicated geometry… w/ relatively smaller fraction of total CPU usage

Existing Improvements

- **Static library**: avoid calls to procedure linkage table (PLT) for dynamic loading of libraries
- **Production cuts**: 0.01mm (pixel), 0.1mm (strip tracker), 1 mm (ECAL/HCAL), 0.002 mm (muon systems), 1 cm (support structure)
- **Tracking cut**: 2 MeV (within beampipe) → avoid looping electrons
- **Time cut**: 500 ns
- **Shower library**: use pre-generated showers in forward region (HF, ZDC, Castor)
- **Russian roulette**: discard N-1 neutrons < 10 MeV or gammas < 5 MeV (in calorimeters), retain N^{th} particle and assign it a weight of N
- **FTFP_BERT_EMM**: modified physics list, simplified multiple scattering model for most regions (default used for HCAL, HGCal)

➢ When all optimizations applied together, CMS achieves ~3–5× speedup!

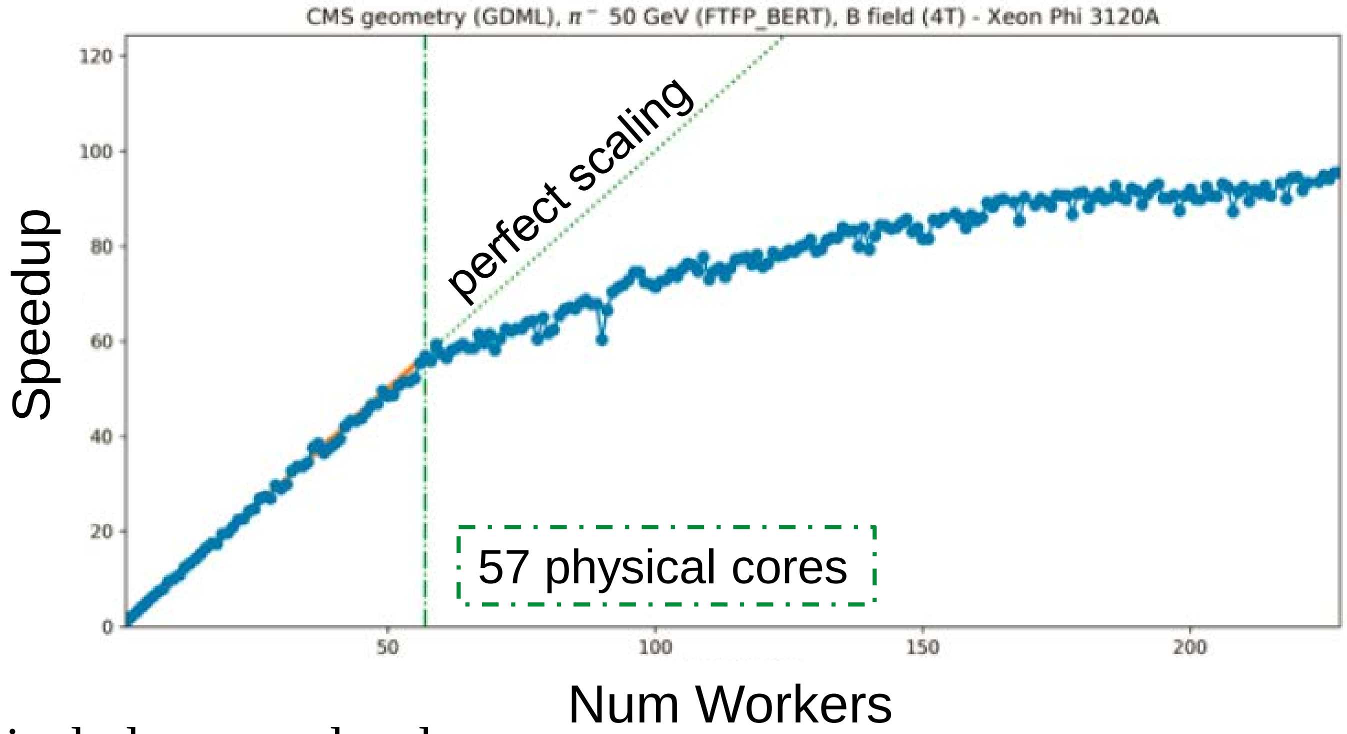
Results of Existing Improvements

	Relative CPU usage	
Configuration	**MinBias**	**ttbar**
No optimizations	1.00	1.00
Static library	0.95	0.93
Production cuts	0.93	0.97
Tracking cut	0.69	0.88
Time cut	0.95	0.97
Shower library	0.60	0.74
Russian roulette	0.75	0.71
FTFP_BERT_EMM	0.87	0.83
All optimizations	0.21	0.29

- From HEP Software Foundation Community White Paper
  - CMS Phase 0 detector, Geant4 10.2
- HF shower library, Russian Roulette have largest impacts
- Cumulative effects: with all improvements, simulation is 4.7× (3.4×) faster for MinBias (ttbar)
- CMS simulation takes 4.3 sec†/event (24.6 sec†/event) for MinBias (ttbar)

†1 sec = 11 HS06 for test machine

Multithreading



- Geant4 includes event-level multithreading
- Nearly perfect scaling with physical cores, further 30% gain from hyperthreading
- Memory reduced by factor of 10 (vs. multiprocessing approach)

- CMSSW framework supports multithreading
- Similar gains in throughput observed, memory usage remains under 2GB

➢ More efficient use of grid resources (included in CMS production releases)

New Improvements: Geometry

VecGeom: new library for detector geometry

- Supports vectorization and new architectures
- Code rewritten to be more modern and efficient (vs. Geant4, ROOT, USolids)
- Can be used in scalar mode with Geant4
- CMS observes 7–13% speedup with similar memory usage

  → Just from code improvements, no vectorization!

➢ Included in latest CMS production releases

  - First mainstream use of vectorized library by experiment

	Relative CPU usage	
Geometry library	**MinBias**	**ttbar**
Native	1.00	1.00
VecGeom	0.87	0.93

New Improvements: Magnetic Field

- Faster stepper (G4DormandPrince745) for tracking in magnetic field
  - Also a more robust algorithm
- Smart tracking: energy-dependent propagation through EM fields
- CMS observes 8–10% speedup with these optimizations (preliminary)
  - Enabled by migration to latest Geant4 version 10.4

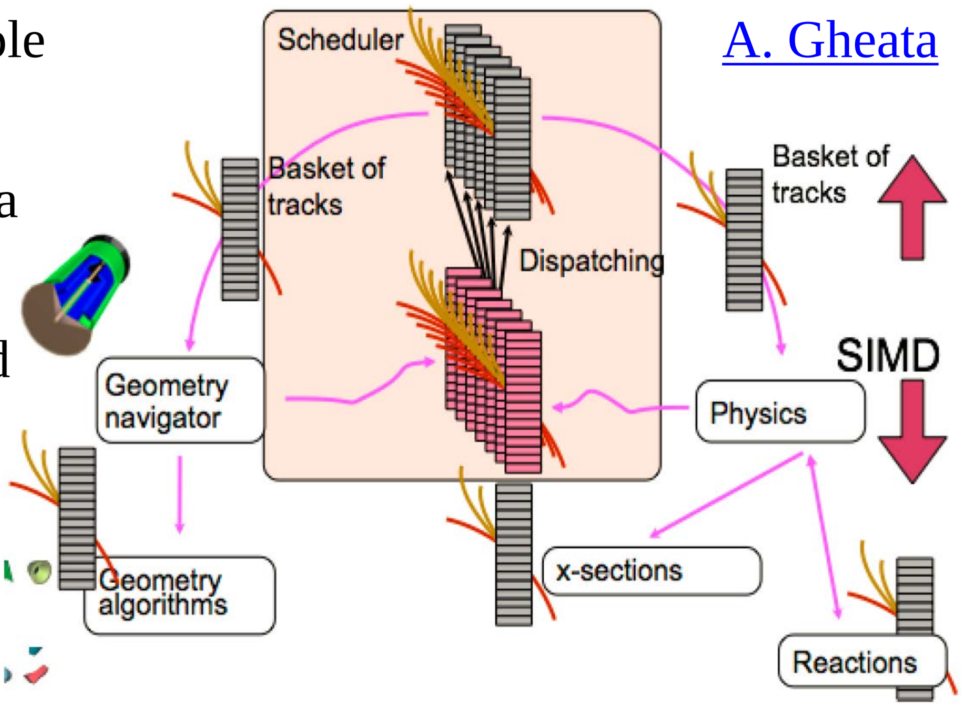
	Relative CPU usage	
Stepper	**MinBias**	**ttbar**
G4ClassicalRK4	1.00	1.00
G4DormandPrince745	0.93	0.98
G4DormandPrince745 + smart tracking	0.92	0.90

(tested w/ gcc 7.0 and 16 threads)

Potential Improvements: GeantV

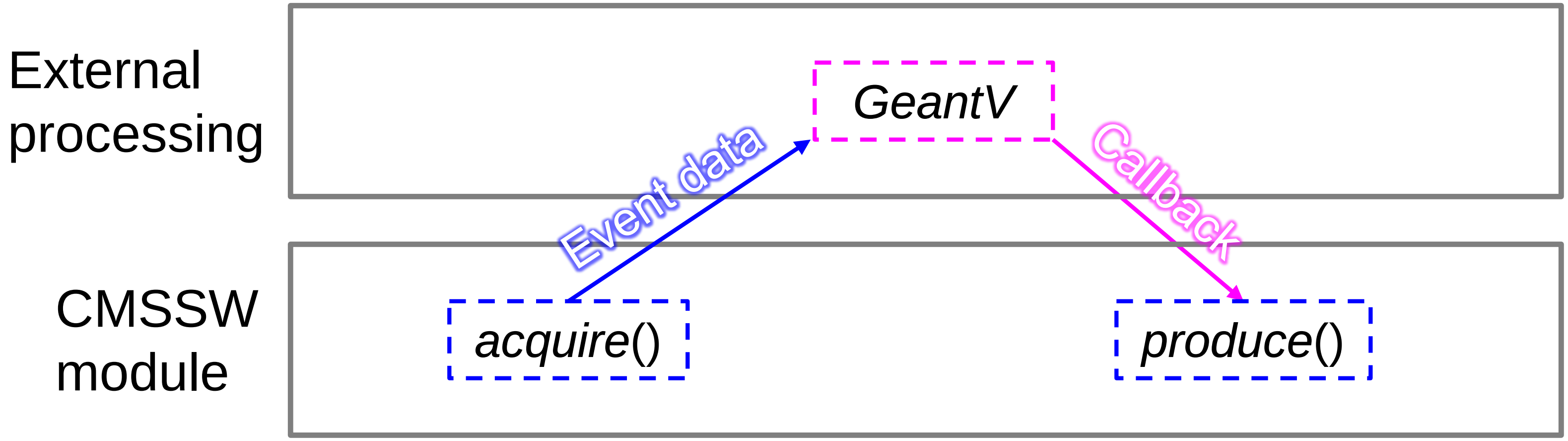
- CMS has already achieved significant speedups in Geant4 and enabled event-level multithreading for more efficient use of resources
- However, even this will not suffice for the demands of Phase 2
- Enter GeantV: Vectorized Transport Engine
  - Track-level parallelism: process multiple events simultaneously
  - Exploit single instruction, multiple data (SIMD) vectorization
  - Group similar tracks into *basket* (based on particle type, geometry/material)
  - Send entire basket to algorithm: process particles in parallel

A. Gheata



Early Testing of GeantV in CMSSW

- Started with integration into toy-mt-framework → included in alpha release
  - o Used for CMS multithreading R&D (Intel Thread Building Blocks)
- ➢ Now have a working example compatible w/ CMSSW development release
- Run GeantV in "external loop" mode using CMSSW ExternalWork feature:
  - o Asynchronous task-based processing



- ➢ Co-development approach: test consistency of threading models, interfaces
  - o Provide feedback to prevent divergence between CMS and GeantV

Elements of GeantV Integration

- ✓ Generate events in CMSSW framework, convert HepMC to GeantV format
- ✓ Build CMSSW geometry natively and pass to GeantV engine
- Using constant magnetic field, limited EM-only physics list
- Sensitive detectors and scoring not yet adapted to new interfaces
- Production cuts also not yet included
- ➢ First integration of GeantV into experimental software framework
  - Run with elements specified above
  - Integration with downstream steps (e.g. digitization): longer timescale, requires more development for thread-safe scoring
- CMS will test GeantV beta release, targeting demonstration of speedup
  - Community decision to support GeantV engine as part of Geant4 on timescale of HL-LHC

Conclusions

- CMS has substantially reduced CPU usage of Geant4 full simulation
  - ~3–5× speedup using various technical improvements and physics-preserving approximations
  - Continue to find ~10% improvements, e.g. from VecGeom and magnetic field stepper/tracking optimizations
- HL-LHC and Phase 2 upgrades bring significant challenges:
  - ➢ Need more events, more accuracy, in more complicated geometry… w/ relatively smaller fraction of total CPU usage
- GeantV is one promising approach to speed up full simulation even further
  - Track-level parallelism (rather than event-level), vectorized components
  - Alpha release is available, beta release planned for 2019
  - ➢ Successful early integration in CMS software framework!
  - Aim for 2–5× speedup with final version

Acknowledgements

- Results and R&D presented here are the products of years of work by many scientists, developers, etc. – a (multi-) team effort!
- Thanks to:
  - Geant4 Collaboration
  - GeantV R&D Team
  - CMS Simulation Group
  - CMS Core Software Group
  - HEP Software Foundation
  - Support from Intel, Fermilab, and CERN OpenLab

References

- M. Hildreth et al., "CMS Full Simulation for Run-2", *J. Phys. Conf. Ser.* 664 (2015) 072022, doi:10.1088/1742-6596/664/7/072022.
- HEP Software Foundation, "A Roadmap for HEP Software and Computing R&D for the 2020s", HSF-CWP-2017-01, arxiv:1712.06982, December 2017.
- HEP Software Foundation, "Detector Simulation White Paper", HSF-CWP-2017-07, arxiv:1803.04165, October 2017.
- D. Elvira et al., "CMS Simulation in the HL-LHC Era", HSF-CWP-011, January 2017.
- D. Elvira, "VecGeom in CMS, Mu2e, Muon g-2". Joint WLCG & HSF Workshop, Napoli, March 2018.
- J. Apostolakis et al., "Towards a high performance geometry library for particle-detector simulations", *J. Phys. Conf. Ser.* 608 (2015) 012023, doi:10.1088/1742-6596/608/1/012023.
- K. Pedro, "Tests of GeantV in CMS Software Framework". Joint WLCG & HSF Workshop, Napoli, March 2018.

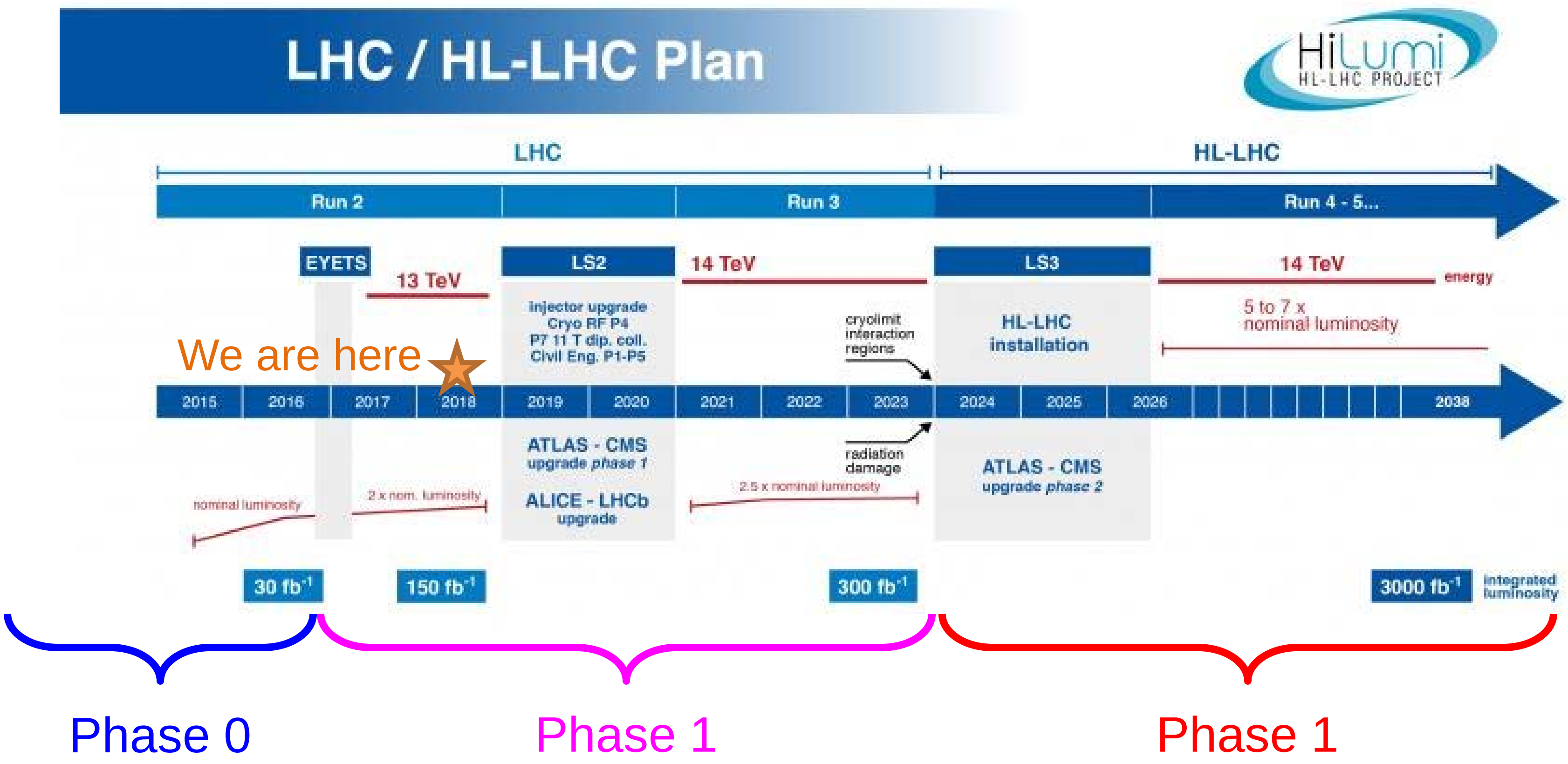
Repositories

- **CMSSW** (GitHub/cms-sw)
  - CMS Offline Software, ~6 million LOC
- **VecCore** (GitHub/root-project)
  - SIMD abstraction library
  - Supports backends: Vc, UME::SIMD, CUDA
- **VecMath** (GitHub/root-project)
  - Vectorized math utilities
  - Built on top of VecCore
- **VecGeom** (CERN/GitLab)
  - Vectorized geometry and navigation, multi-particle interface
- **GeantV** (CERN/GitLab)
  - Alpha release now available!
  - cmsToyGV example
- **toy-mt-framework** (GitHub/Dr15Jones)
  - Original toy framework for CMS multithreading development
- **install-geant** (GitHub/kpedro88), **SimGVCore** (GitHub/kpedro88)
  - Test repositories to install and integrate GeantV in CMSSW

Backup

CMS & LHC Upgrade Schedule

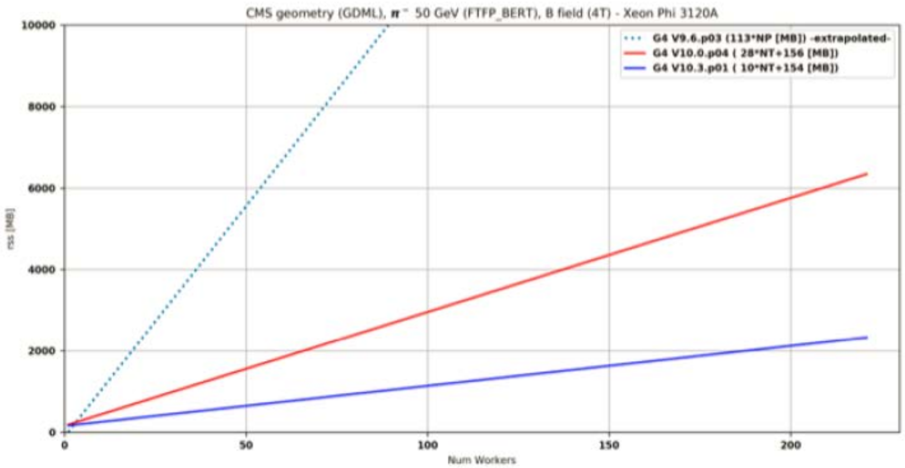
- Phase 1 upgrades began during Run 2 and will be in operation through the end of Run 3 (installation finishes during Long Shutdown 2)
- Phase 2 upgrades will be in operation during Runs 4, 5 (installation during Long Shutdown 3)



Simulation Test Details

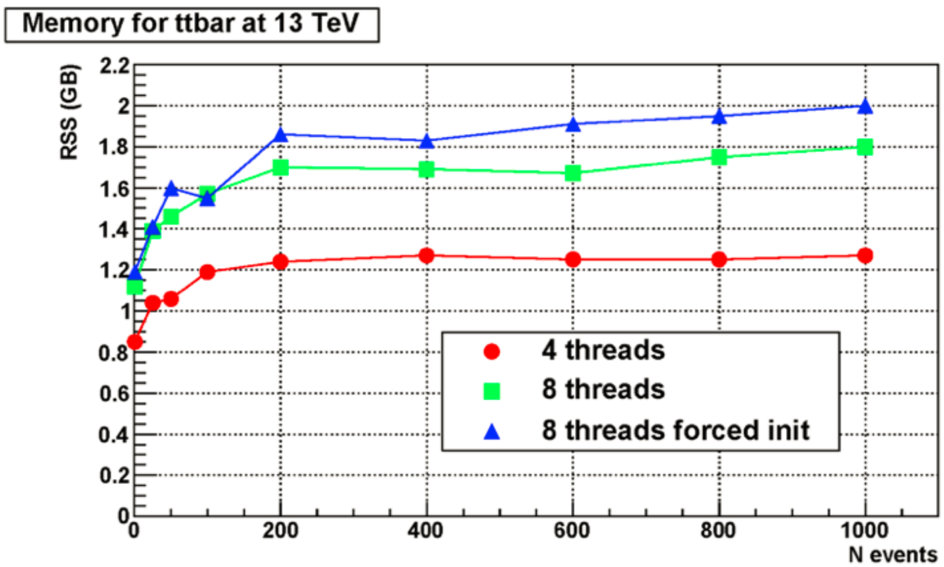
- Machine: olhswep16.cern.ch (CERN OpenLab)
- Single-threaded runs
- Compiler: gcc 6.3.0
- Geant4: version 10.2
- FTFP_BERT physics list
- Pythia event generation: $\sqrt{s}$ = 13 TeV, 300 events, $|\eta| < 5.5$ (minimum bias, ttbar)
- Particle gun event generation: 50 GeV electrons, muons, pions flat distribution in $\eta = [-0.8, 0.8]$, $\eta = [2.0, 2.7]$, $\varphi = [0, 2\pi]$
- Geometry: 2016 detector version (default)

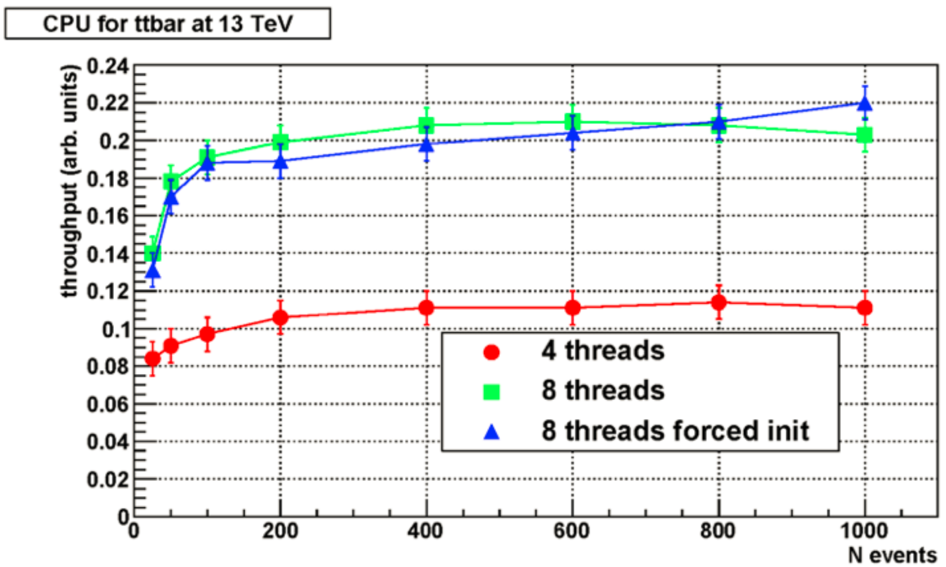
Multithreading



← Standalone test

CMSSW framework ↓

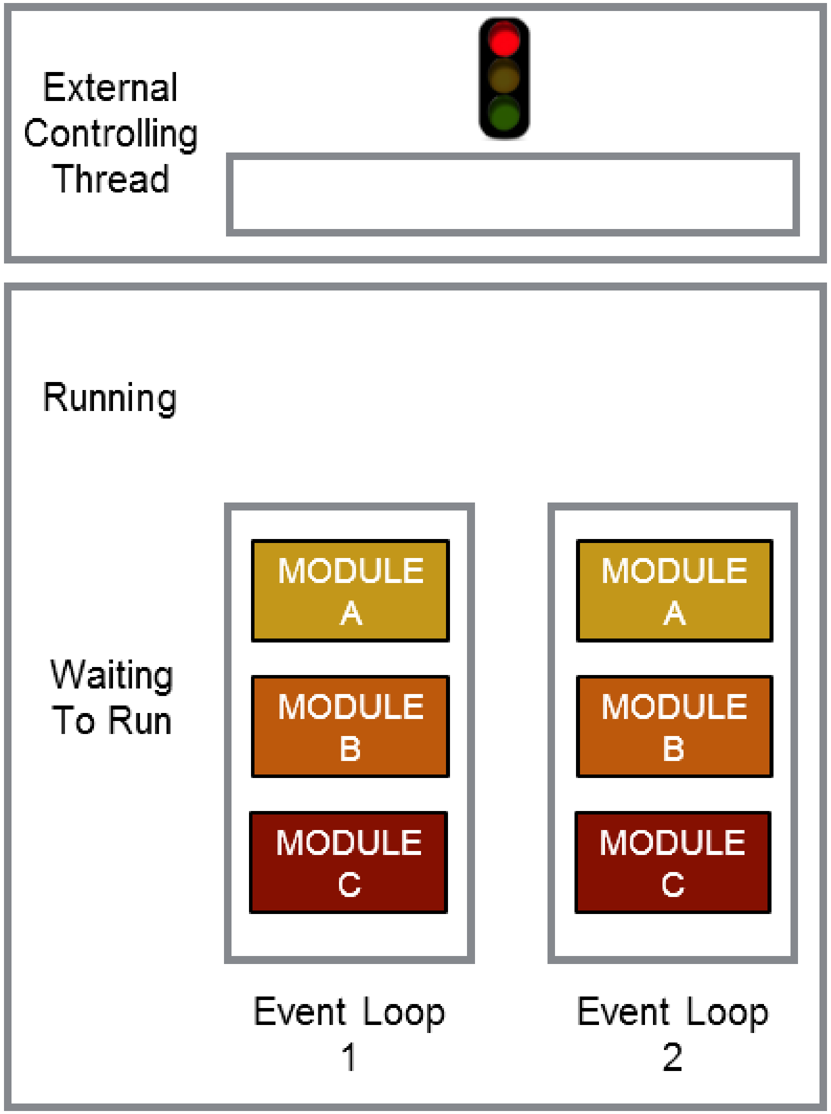




External Work in CMSSW (1)

Setup:

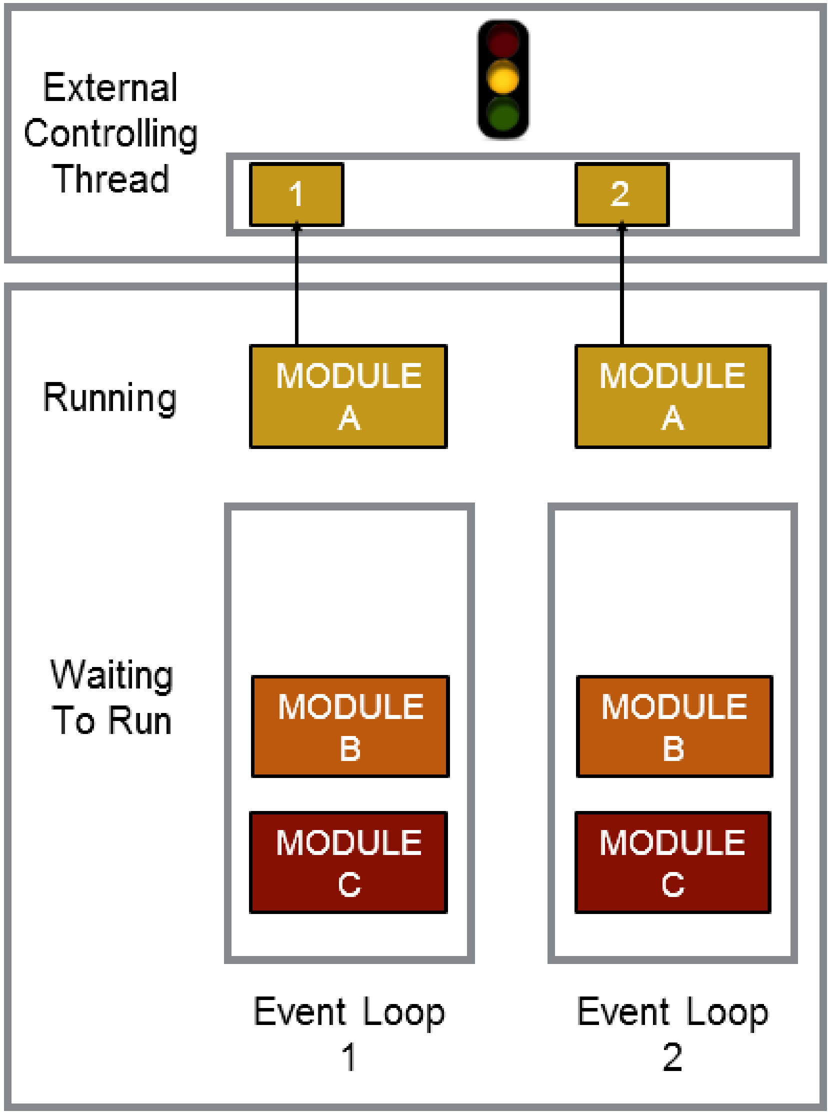
- TBB controls running modules
- Concurrent processing of multiple events
- Separate helper thread to control external
- Can wait until enough work is buffered before running external process



External Work in CMSSW (2)

Acquire:

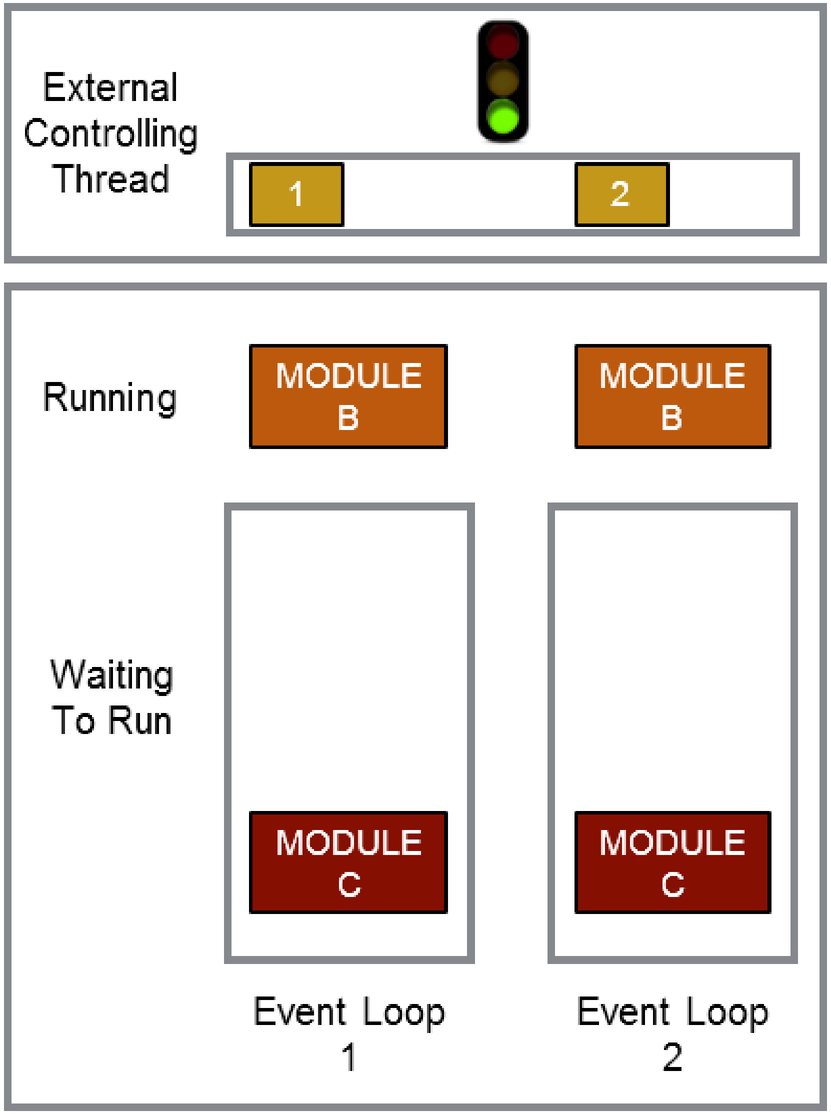
- Module *acquire*() method called
- Pulls data from event
- Copies data to buffer
- Buffer includes callback to start next phase of module running



External Work in CMSSW (3)

Work starts:

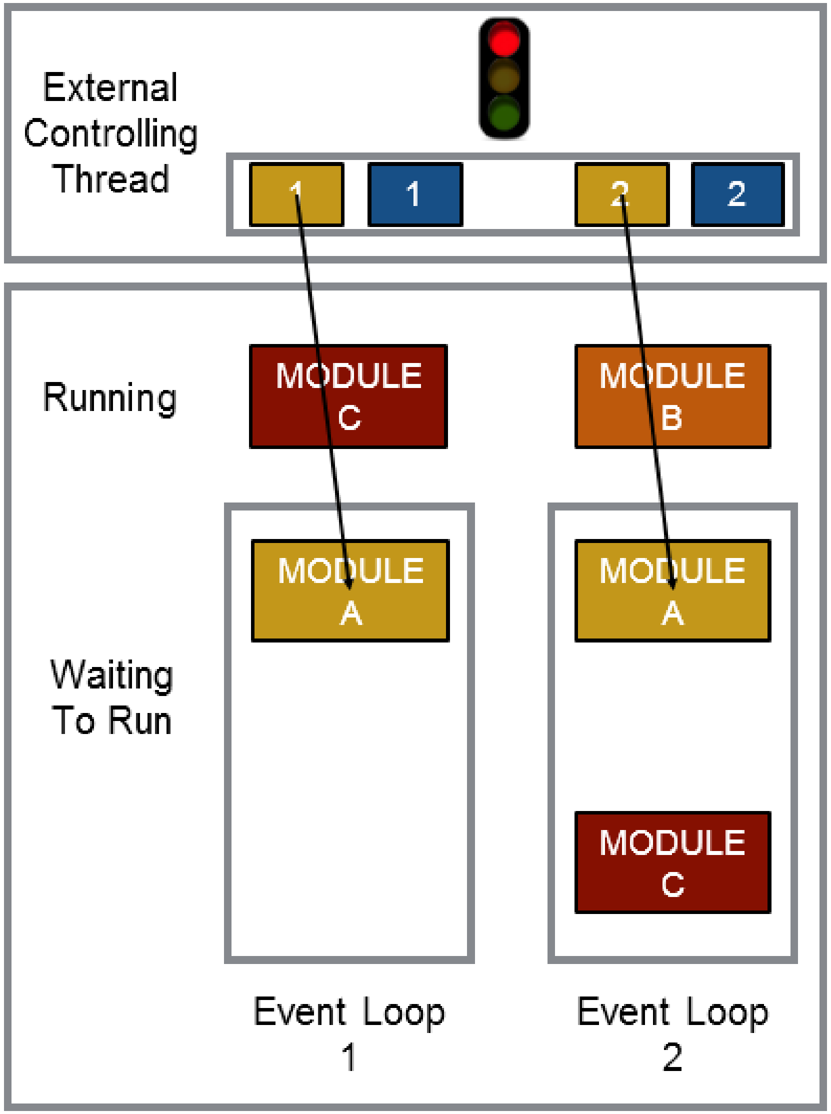
- External process runs
- Data pulled from buffer
- Next waiting modules can run (concurrently)



External Work in CMSSW (4)

Work finishes:

- Results copied to buffer
- Callback puts module back into queue



External Work in CMSSW (5)

Produce:

- Module *produce*() method is called
- Pulls results from buffer
- Data used to create objects to put into event