CPU Performance: ATLAS & CMS

Input from ATLAS & CMS

CMS Full Simulation Computing performance

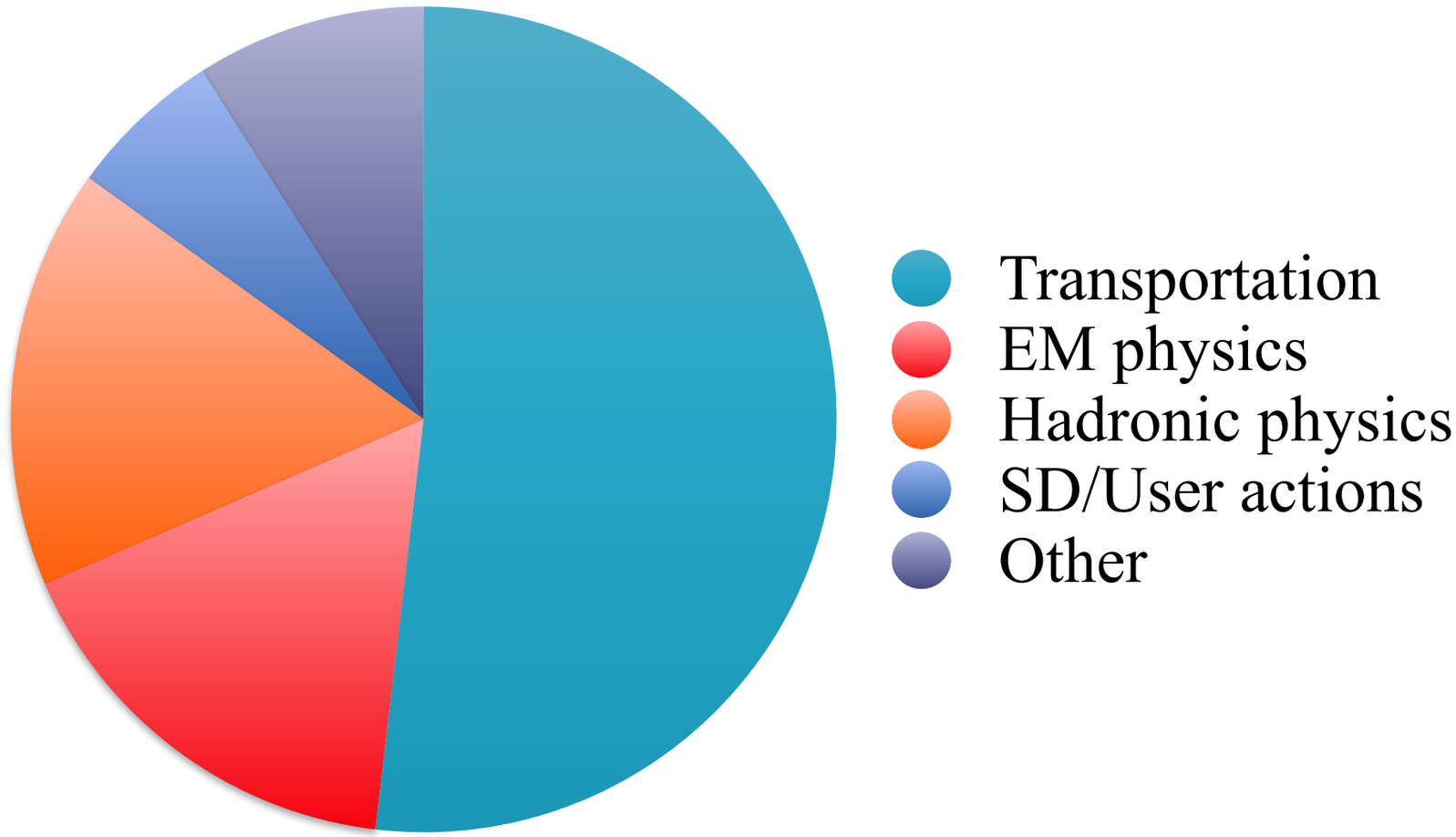
Vladimir Ivantchenko for the CMS Simulation team

Geant4 status in CMS

- Production version of Geant4 for 2015
  - Geant4 version10.0p02 built in sequential mode
  - Production platform slc6_amd64_gcc491
  - Default physics list: QGSP_FTFP_BERT_EML
  - ~5 billion events already produced in 2015
  - This number will increase for the end of the year

- Current development versions of Geant4 in CMSSW
  - Geant410.0p03 + patch of Geant4e for threading is established
    - Multi-threaded Geant4 is fully integrated with CMS multi-threaded framework
    - Our goal is to use it in production 2016
    - Platform slc6_amd64_gcc493
  - Geant4 10.1p02 is also available in development branches
  - Geant4 10.2 planned be our production version in 2017

Where our simulation CPU goes (CHEP'15 talk)



- Technical performance improvements for Run 2 simulation:
  1. Upgrade to Geant4 10.0 (~5%)
  2. Implementation of Russian Roulette technique (~30%)
  3. CMSSW code optimization (~15%)
  4. Library repackaging (~10%)

10% performance gain from hidden visibility without playing with linker scripts (CHEP'15 talk)

Repackage all shared libraries in CMSSW that depend on Geant4 into a single static library to "hide" Geant4 from the rest of CMSSW

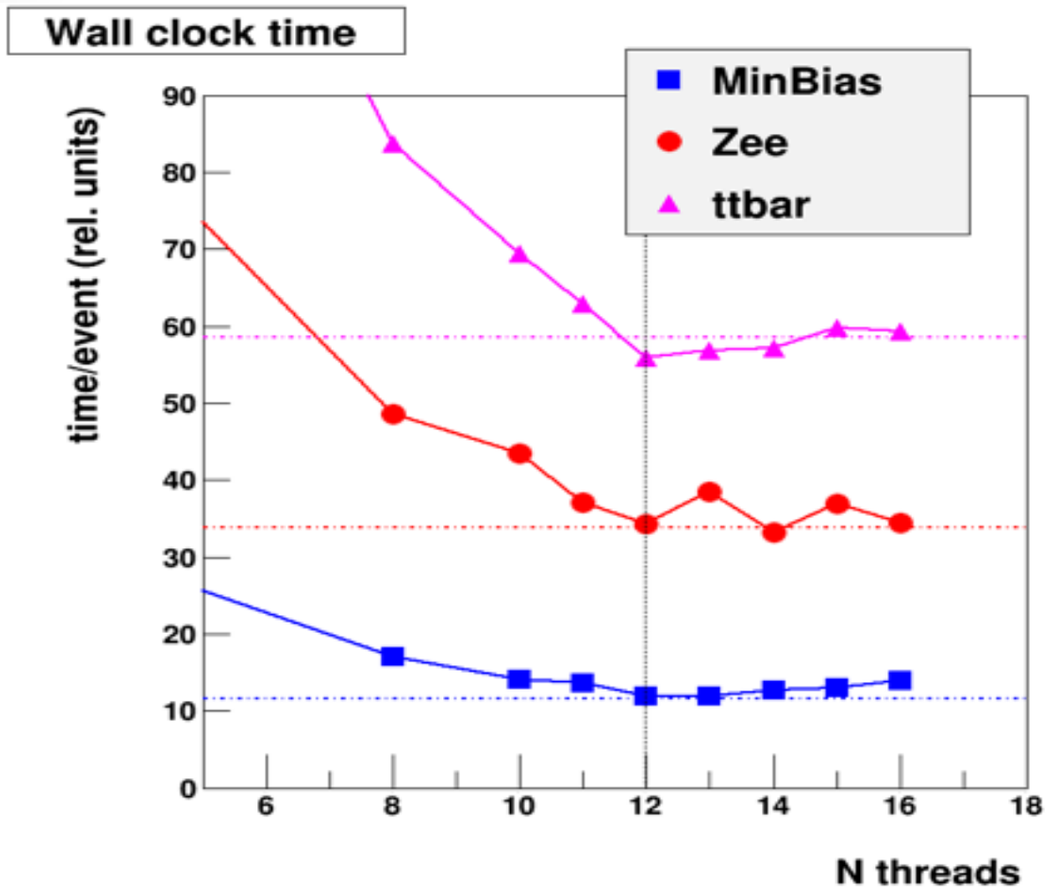
- Use single archive library for Geant4 itself
- This allowed us to more aggressively optimize at link time: adding "**-flto -Wl,--exclude-libs,ALL**" works best

Constraints this imposes:

- Must control dependencies to use Geant4 only within this single library:
  - This is "easy" for simulation (<2% of our libraries)
  - However, extending this idea to something effecting the full reconstruction is difficult
- Impact on simulation code developers minimized by keeping .so cached in release. Static library rebuild is the only extra step if developer builds a package in this static library.

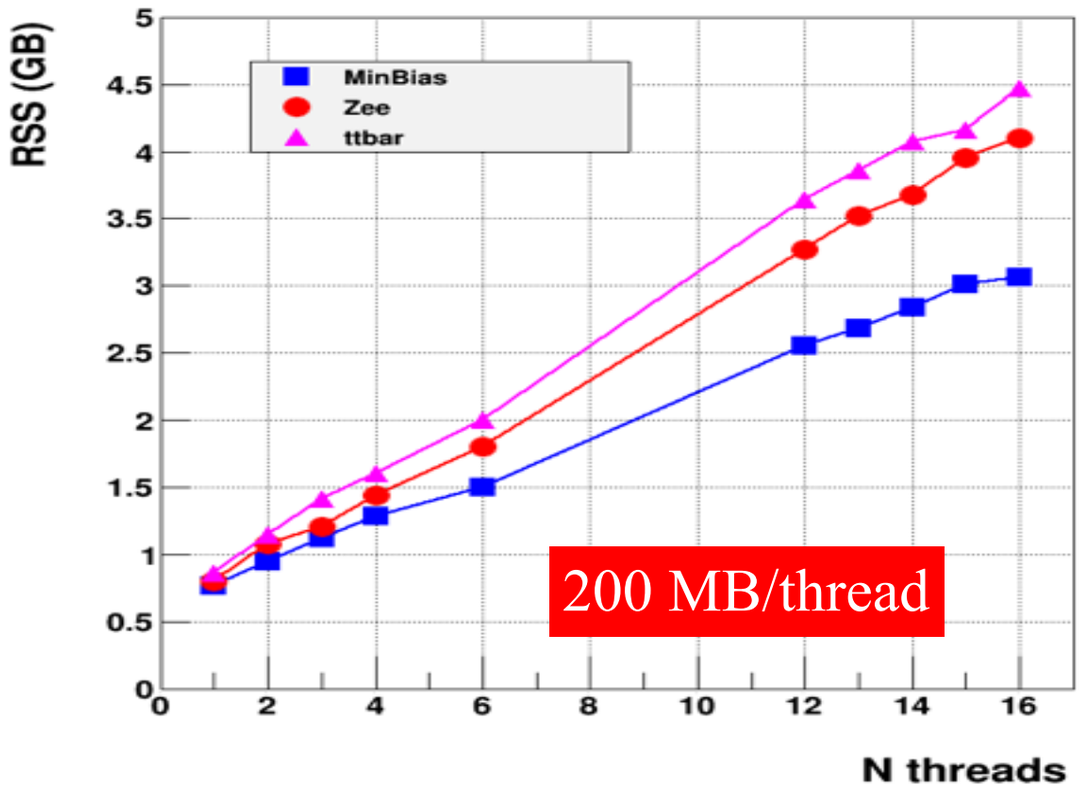
Current performance of CMS MT GEN-SIM (CHEP'15 talk)

Time/event decreases until the # of threads is equal to the # of cores



Extrapolation from single core

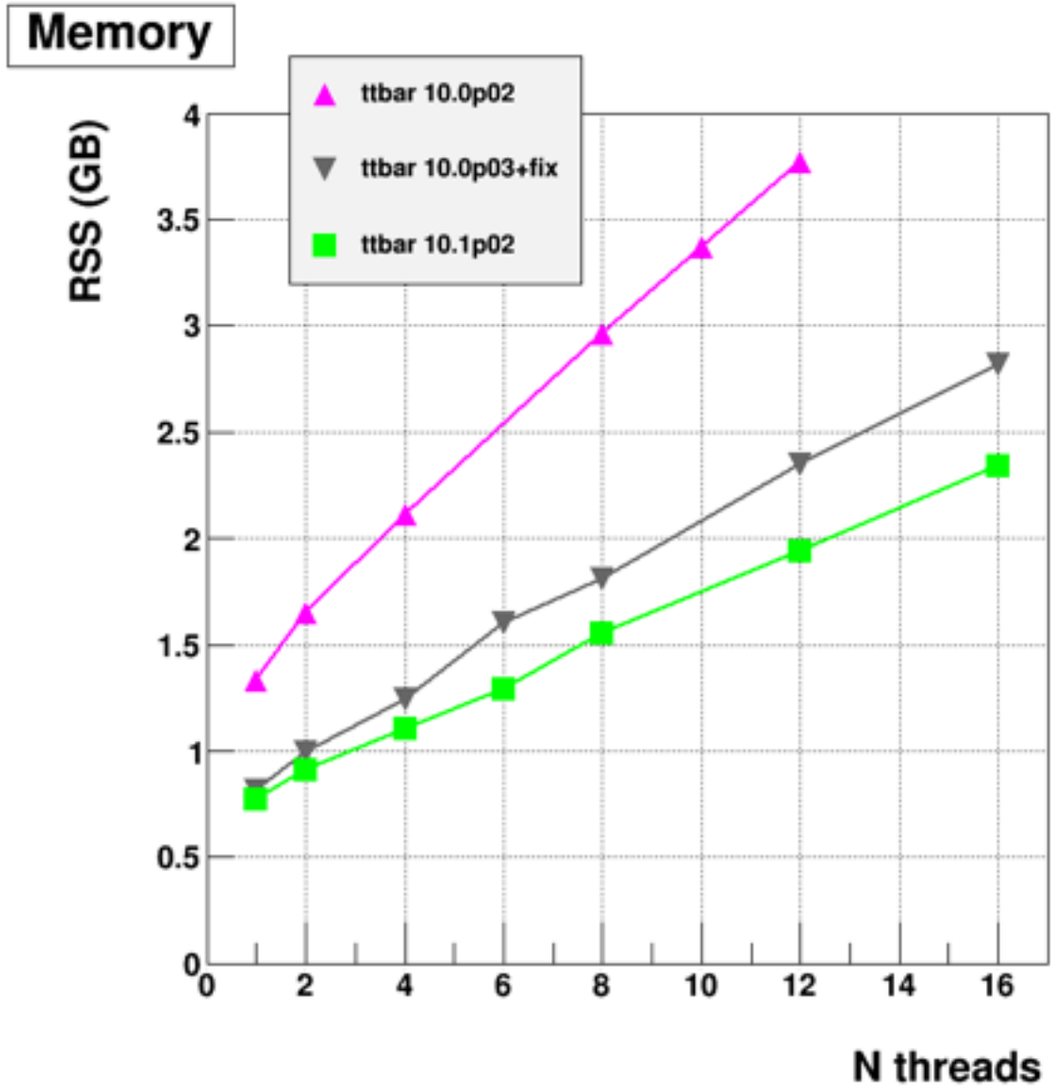
Example memory savings: a single 12 threaded MT job requires **~4 GB** RSS instead of **11 GB** for 12 single threaded jobs



Excellent scaling performance seen in our tests so far.
Geant4 version 10.0p02

Performance results for Geant4 10.1p02

- **For CMS CPU performance is the same in 10.1 and 10.0**
- Memory grows was observed in 10.0p02 about 1.3 M/event
- There is two main contributions to the memory grows:
  - Memory leak in gamma-nuclear and FTF models
    - Objects created with G4Allocator were not deleted
  - Ineffective data structure for nuclear gamma evaporation models
    - Significant memory was used per thread for nuclear level data
- Both problems were fixed in 10.1p02 and these fixes were backported to CMSSW on top of 10.0p03
- **After backport of fixes required RSS memory for 10.0p03+fix become very similar to 10.1p02**



For 10.0p02 results are shown after 1000 events

ATLAS Full Simulation Computing performance

Zach Marshall, Elmar Ritsch, Philip Clark, Steve Farrell

Slides - John Apostolakis

ATLAS: Overall

- Integrated Sim. Framework (ISF) use almost all production
- Geant 9.6 full simulation used in 80% of production
  - ratio will drop soon (more fast sim)
- Expect move to 10.1 for next campaign (end 2015)
- Appreciate CPU improv. of e- & γ-nuclear x-section. Would welcome also improv. of hadronic x-sections

Stability and production

- **Crash rate** is a potential **time sink** in production
  - MC15 (G4 9.6) rate was **1.5 10^{-5} per event**
  - Rate for G4 10.1 appears similar - around 1.5 * 10^{-5}
    - This becomes a **1.5% failure** for jobs of **1,000 events** - which is unacceptable. (Push for 1,000 event jobs for efficiency)
- The Multi Level Locator (field propagation) has proven to be a weakness
  - for crashes and possibly cause of very small steps (a time waste)

Hotspots & remedies

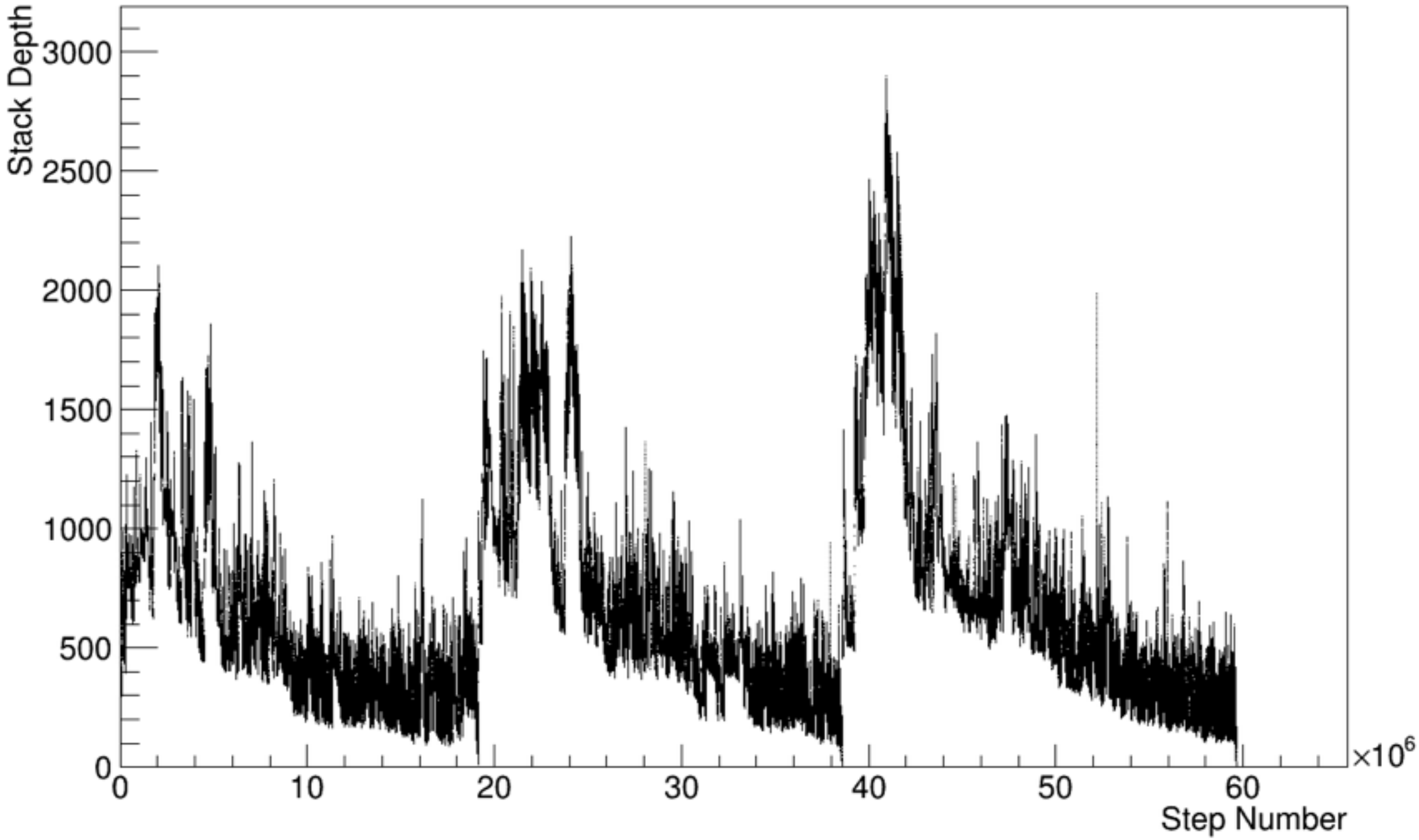
- Neutrons take a lot of CPU time. Plan to profile after getting patches/fixes for previous issues

- Might seek to use available biasing features

- If improvements are provided, happy to benchmark & profile to identify any hotspots.

Memory - use and churn

- Memory consumption is significant (using 600 materials!), but is not enormous concern
- Memory churn was issue in production systems
  - Back-port of fix in Navigation Level reduced it
  - Geant4 no longer fully dominates churn!



Interesting: stack depth in 3 t-tbar events

- average 500-750, maximum 2,000-3,000.

Is this expected ?

Evolution

- Seen potential of static builds vs DLLs. Believe this is very difficult to achieve for ATLAS.
- Looking into other possibilities:
  - Wish further study of profile guided optim
  - Does it point to suboptimal code?

HPC and MT

- Reasonably advanced prototype of MT app for Cori (= Xeon-Phi based next gen HPC at NERSC.)

- G4 developers responsive & helpful for questions and functionality needed

- Expect prototype to continue to mature, and more perf. questions to arise

- Had difficulty due to different design choices between G4 threading model and Gaudi Hive (task-based). Andrea Dotti was very helpful in getting this going