http://www.geant4.org

Geant4 on HPC systems

Andrea Dotti for the Geant4 collaboration (adotti@slac.stanford.edu) ;
Fundamental Interactions /Advanced Modeling Solutions

Outlook

Introduction

Parallelization results:

1. Multi-threading latest results
2. Towards HPC: MPI inclusion
3. Results on Mira@ANL

Simplifying migration: use of containers

Conclusions

Introduction

The path forward

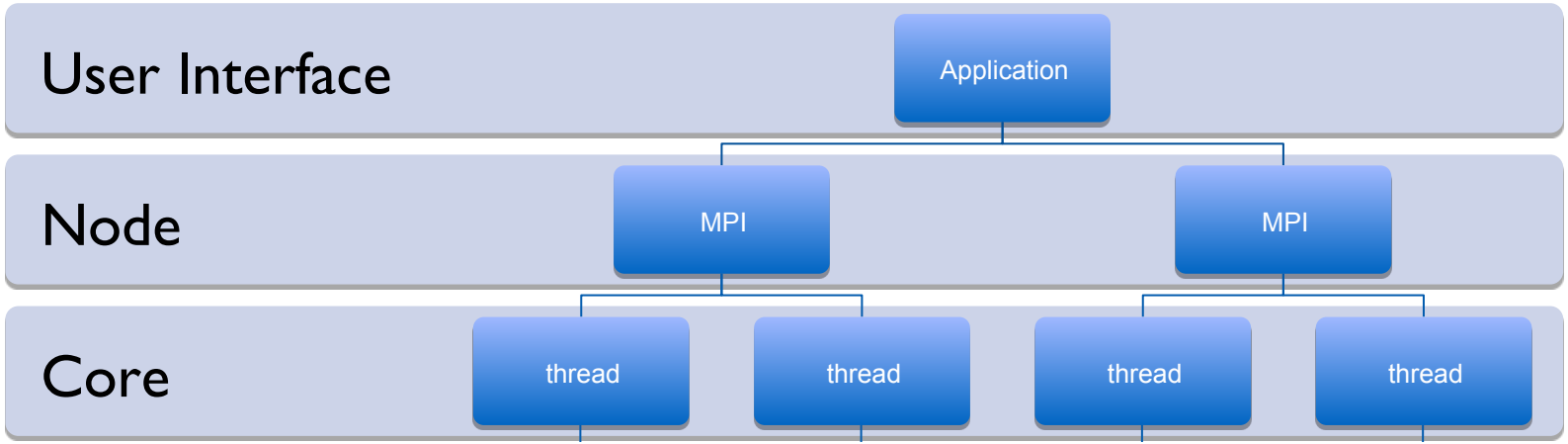
To use HPC systems:

1. Introduce multithreading parallelism (**DONE**)
2. Introduce multi-process parallelism (**DONE**)
3. Study strategy to distribute code efficiently (**DONE**)
4. Leverage accelerators (**BEING STUDIED**)

Leveraging HPC needs success in **all** areas

Will anyway benefit also traditional systems (GRIDs)

Geant4 Strategy for parallelism

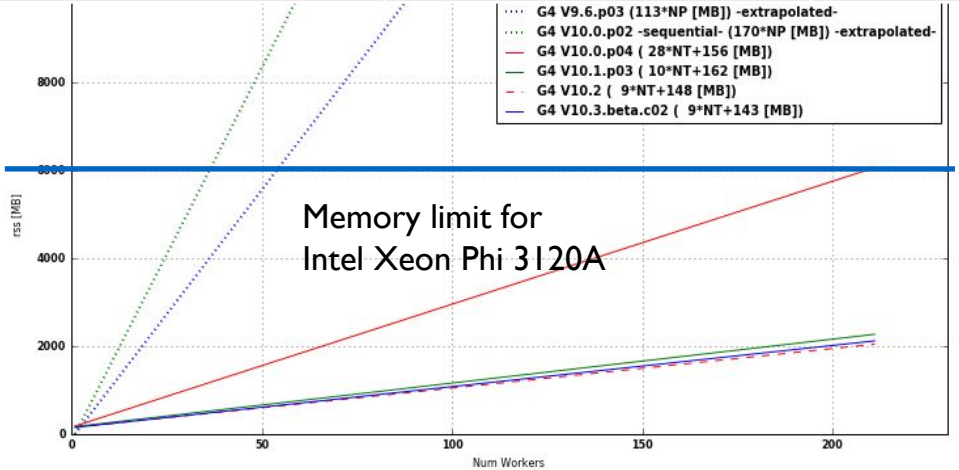


We provide defaults for all level of parallelism, users can overwrite with experiment framework specific technologies
E.g. LHC experiments: GRID instead of MPI, TBB instead of pthread

Multi-threading

Memory reduction

Version	Initial memory	Memory/thread
9.6 (no MT)	113 MB	(113 MB)
10.0.p02 (no MT)	170 MB	(170 MB)
10.0.p02	151 MB	28 MB
10.4	148 MB	11 MB



Geant4 MT design principle: share between threads read-only data (geometry, physics tables): **lock-free event loop**

Goal: **substantially reduce memory usage w.r.t. pure multi-process application (e.g. MPI) to allow all cores to be used**

Recent feedback from CMS: full CMSSW sw stack of ttbar events:
~200MB/thread
Includes all user-code
Needs KNL for moderate/large number of threads

HepExpMT benchamrk: Simplified CMS geometry (via GDML), uniform B-Field, 50 GeV π- w/ FTFP_BERT

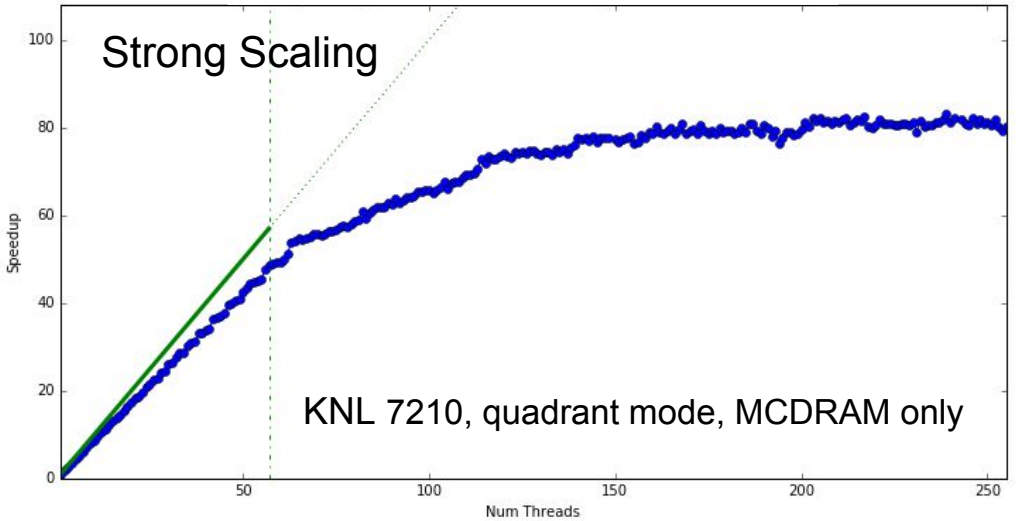
Linearity speedup

Number of events/second is the most important metric for users

Very good linearity (>93%) with the number of physical cores available

Benefits from hyper-threading: ~30%

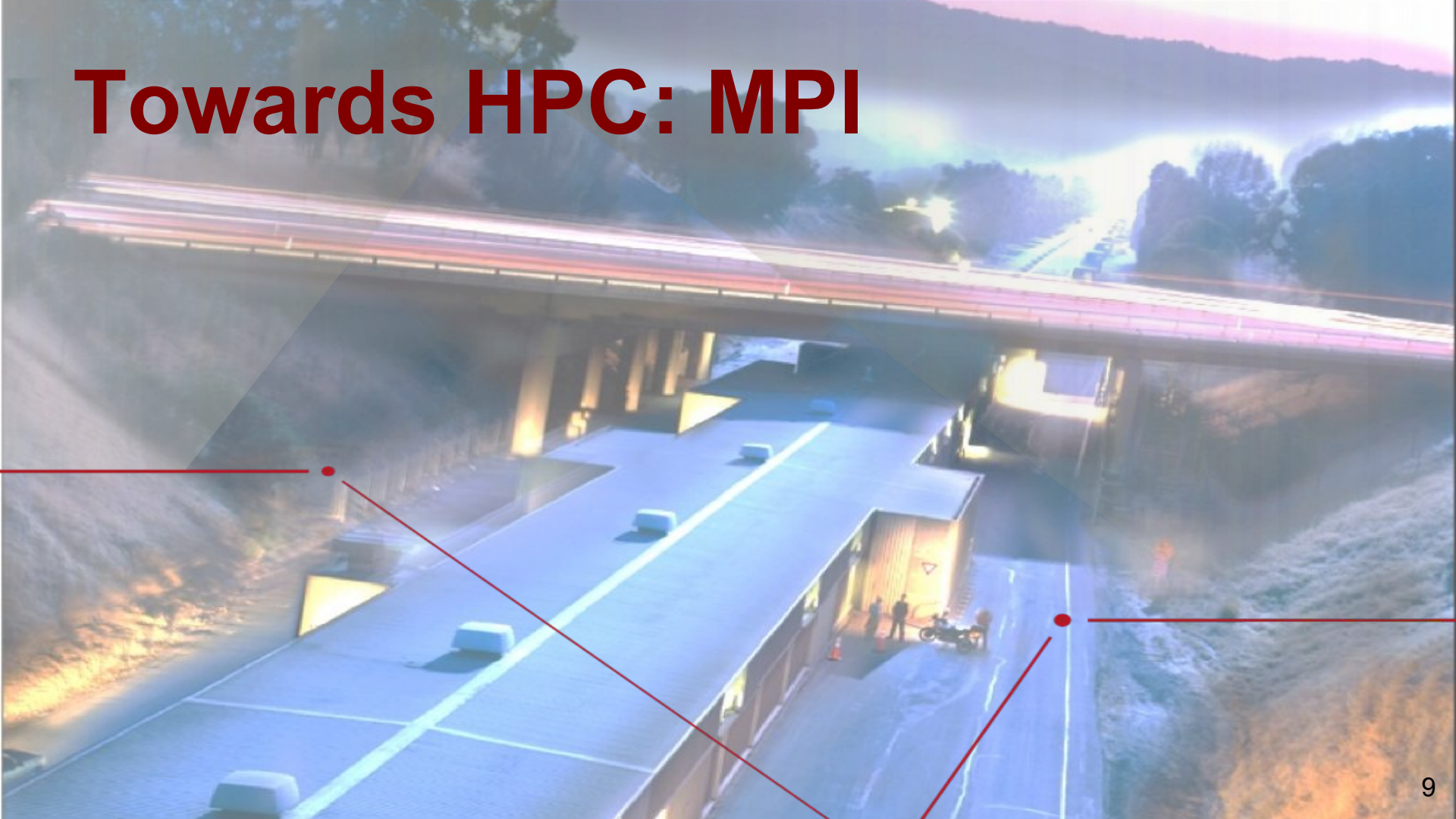
Verified for different types of applications:
Medical physics applications
HEP experiments



HepExpMT benchmark: Simplified CMS geometry (via GDML), uniform B-Field, 50 GeV π- w/ FTFP_BERT

Access to KNL processor provided by Colfax International

Towards HPC: MPI



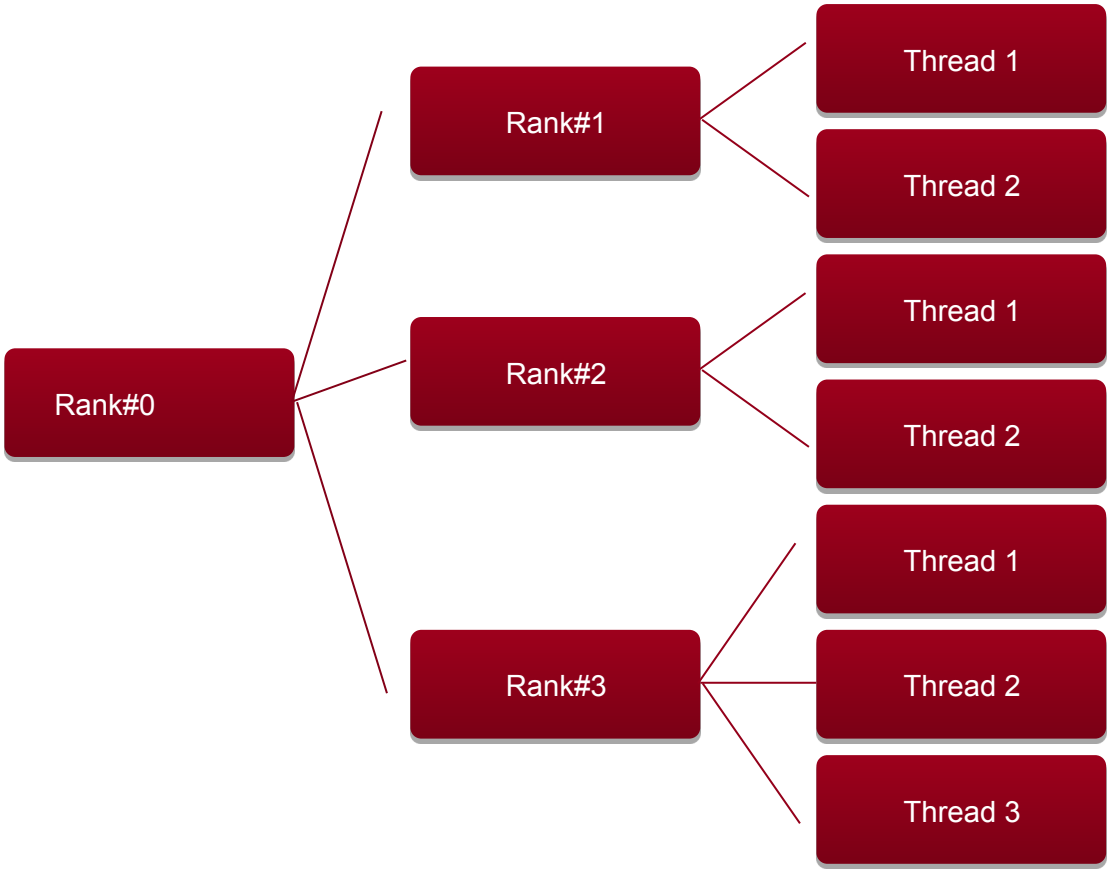
MPI and Geant4

SLAC

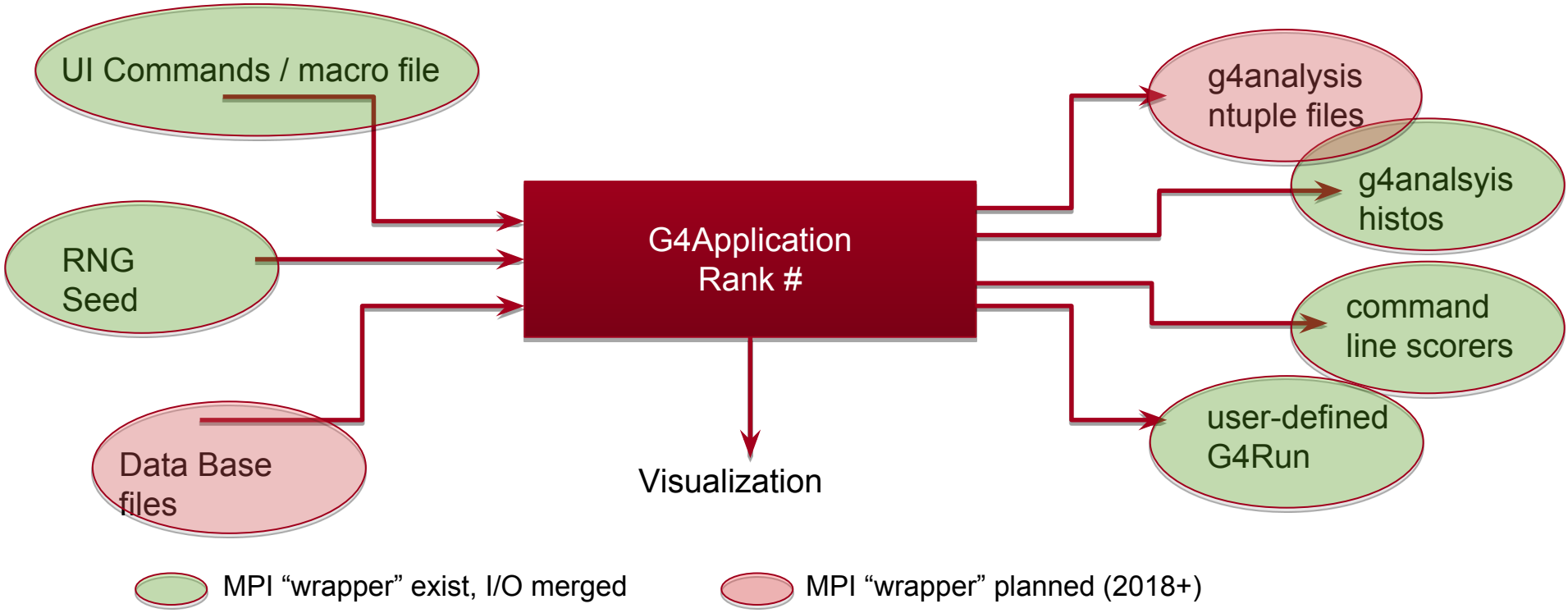
Rank#0 broadcasts UI commands and RNG seeds
Workers send back results for merging: histograms, ntuples, hits

Each node (rank) executes MT job

All processes are “clones” of the master



Geant4 applications from MPI point of view



Results on MIRA@ANL

More memory-efficient, more HPC friendly

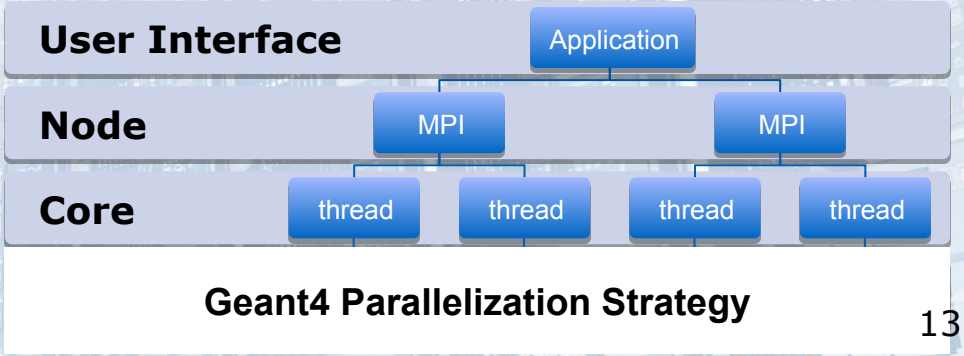
ATLAS: "[...] to help address this *[CPU resources]* shortage, we propose **to move 75% of US ATLAS CPU intensive simulation production tasks** to ASCR HPC facilities."

General characteristics of HPC: millions of tightly interconnected *cores* with a relatively small ratio memory/cores.

(Detector) Simulations are ideal use-cases of HPC: little input/output w.r.t. number crunching, embarrassingly parallel (e.g. fill "holes" in HPC with smaller productions)

Requirements and Achievements:

1. substantially reduce thread-memory usage: **obtained factor 10 reduction**
2. scale linearly to very large number of threads: **3 millions concurrent threads using full MIRA@ANL w/ MPI+MT**
3. opportunistic computing (fill "holes") with checkpointing



Preparing for Next Generation SC

SLAC

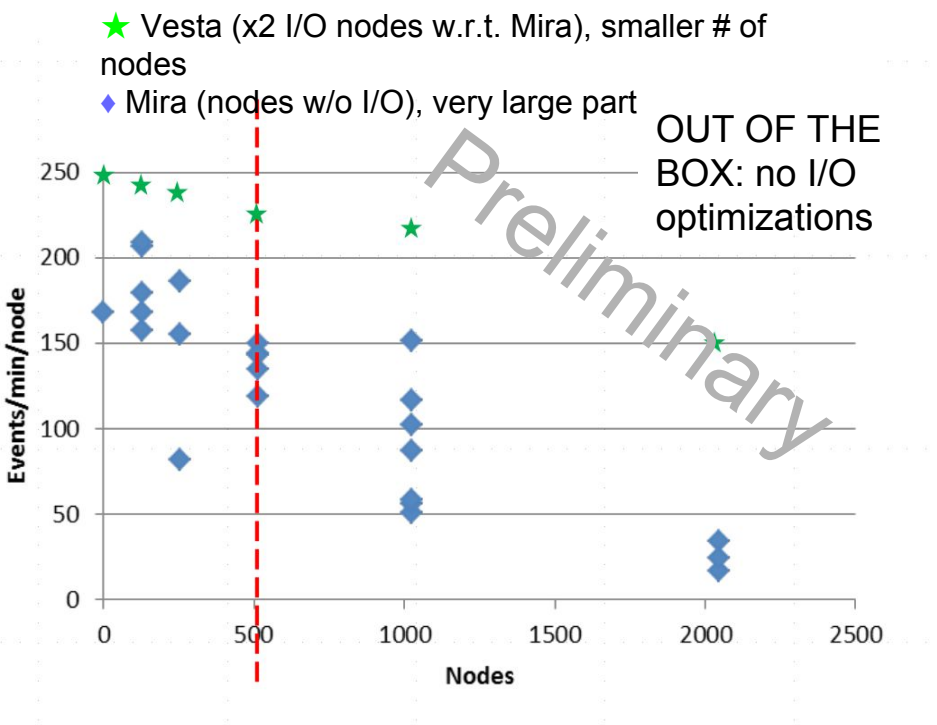
Testing Geant4 at Mira@ANL (BlueGene/Q) with up to 3 million threads

Scaling up to 64k threads, above that hit scaling limit
- Due to limitations in I/O, need to aggregate access to disk, cout/cerr

Special test suppressing all output: mostly recovered saturation up to 3 million threads

Results (eff>80%) confirmed on smaller systems (Tachyon2@KISTI)

We are on the good path to scale to O(10^6) threads provided we manage to address I/O



Courtesy of T. LeCompte ALCF (at ANL)

1 node = 16 BlueGene/Q cores @ 1.6 GHz

Lessons Learned & challenges for the future:

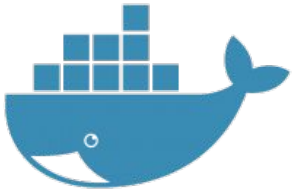
Beside the obvious lack of explicit vectorization (very difficult on O(2M) LOC with physics models, not yet managed after several years of R&D)

- HepExMT@Mira limited by I/O (cout: 100B/event)
- Legacy RNG Engine from CLHEP are not designed for very large parallel systems. Getting close to their limits (exhaust seed state). New MixMax solves issue (to become default soon)
- Reduction of results at the end of the job needs attention: on very large systems, binary tree merging of histograms
- Geant4 “DataBases” (i.e. granular small data-files ~1.5GB organized by Atom {Z,A}) needs to be completely reviewed: at least reduce their granularity (ideally a single large file), real DB MPI-aware distribution of data?

Simplifying porting

docker vs singularity vs shifter

Docker: per-profit company, that open sourced the technology (well, the technology is in linux since long...)



Singularity: compatible with docker, tries to address some docker security concerns. Developed by scientists for scientists (at Berkeley)

Shifter: compatible with docker, developed for HPC systems (at NERSC/Berkeley). Not really for public use (yet?)

As long as you create images in docker format you can always use them in singularity or shifter

Done, images available on Docker Hub

Conclusions

Geant4 and HPC

The introduction of MT has been the first successful step towards the use of HPC

In the last ~2 years we have focused on understanding parallelization across nodes. Success integration of MPI

Last year we have experimented with containers as a way to distribute G4 workloads. Very positive results

Short-term plans (all being addressed): Geant4 “DataBase” distribution, cerr/cout merging at very large scales

Use of hybrid machines remains a concern: how to efficiently use accelerators? Only very specialized workloads have shown promising results

HepExpMT: Geant4 “miniApp”

SLAC/LBL/NERSC collaboration (presented at IEEE/NSS 2015):
S. Farrell, A. Dotti

To be used as a “public candle” for Geant4 performance measurement

http://ieeexplore.ieee.org/document/7581868/

https://twiki.cern.ch/twiki/bin/view/Geant4/Geant4HepExpMTBenchmark

20