http://www.geant4.org

Status of Multi-threading

Makoto Asai, Andrea Dotti (adotti@slac.stanford.edu) ; SD/EPP/Computing

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Overview

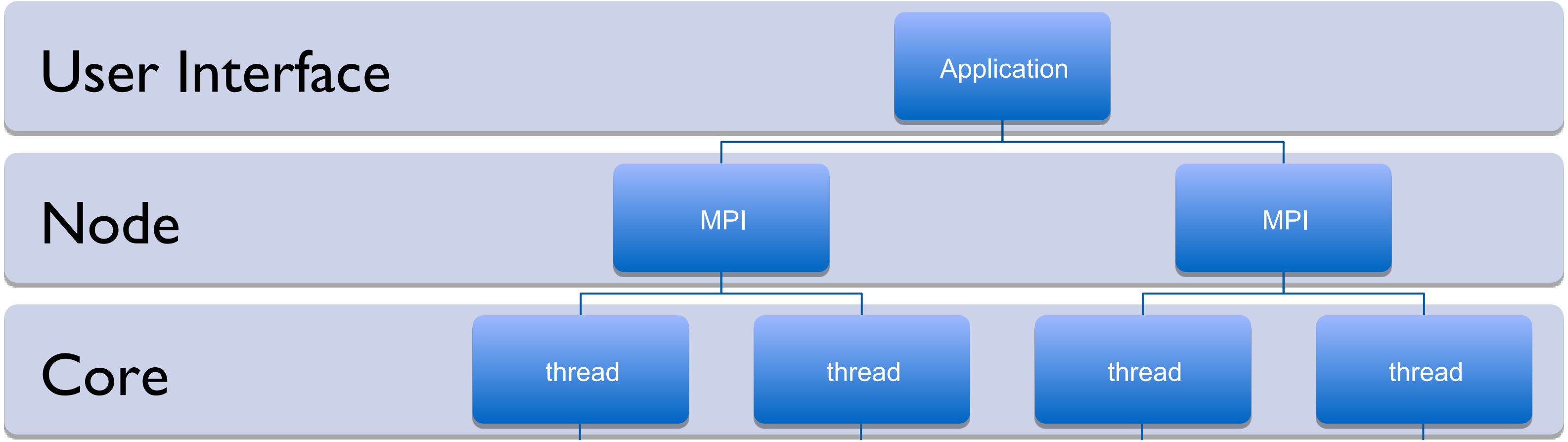
Status of Parallelization efforts: MT, MPI

Results with recent versions of Geant4

Recent fix for abnormal memory usage

Future activities

Reminder Geant4 Strategy for parallelism



We provide defaults for all level of parallelism, users can overwrite with experiment framework specific technologies

Status

Multi-threading

Developments in MT related areas:

- New design of physics lists for cleanup of memory in MT (10.4):
  - Introduce base class for physics lists constructors
  - When Workers end job, call destruction of objects created by physics list
- Added deletion of field-manager store and geometry manager thread-local singletons in G4RunManagerKernel destructor
- Minor cleanup of *workspace* classes (containers to encapsulate thread-local data), work ongoing
- New extended example `parallel/ThreadLocalScorers` shows use of scorers that accumulate over threads and thus save memory for MT application

MPI

No major changes in MPI interfaces

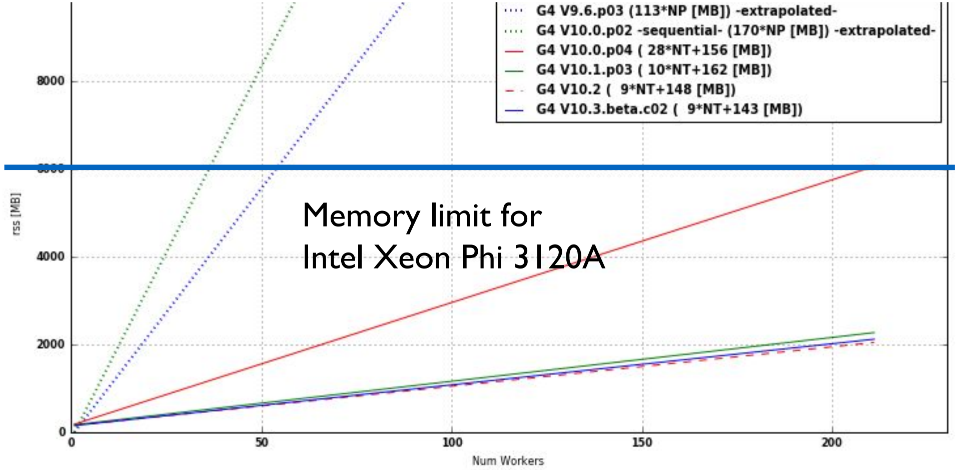
Large scaling tests on Mira revealed issues on RNG seed distributions that appear only on O(10^5) threads. Not an issue except for SuperComputers. May require some important changes in MPI library

Latest version of g4tools (analysis backbone) supports merging of objects (including ntuples) via MPI: to be followed by introduction of methods in manager

Results

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
10.3.beta	148 MB	9 MB



Memory limit for
Intel Xeon Phi 3120A

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)**

Recent campaign to reduce more than a factor 2 memory use in MT mode

Recent feedback from CMS: full CMSSW sw stack of ttbar events: ~200MB/thread

Includes all user-code

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

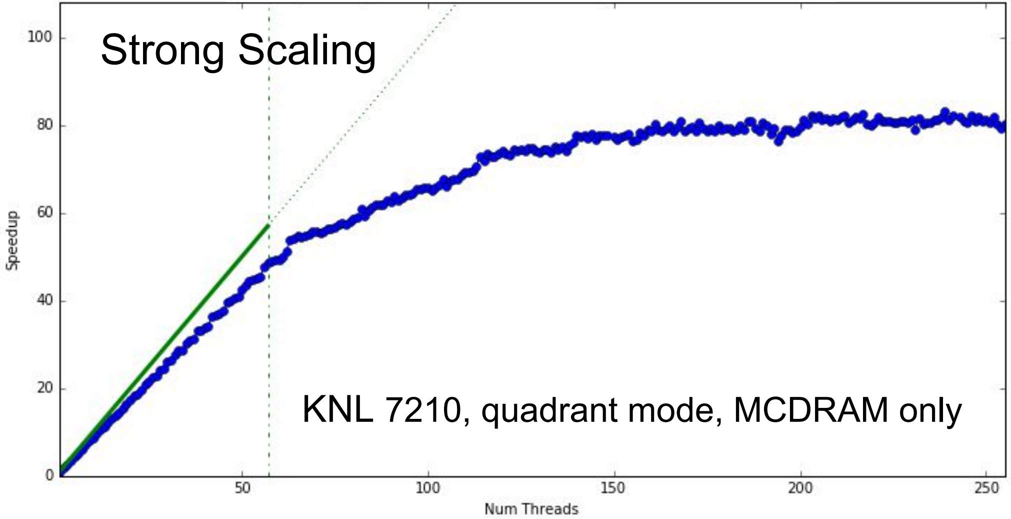
Linearity speedup

- 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

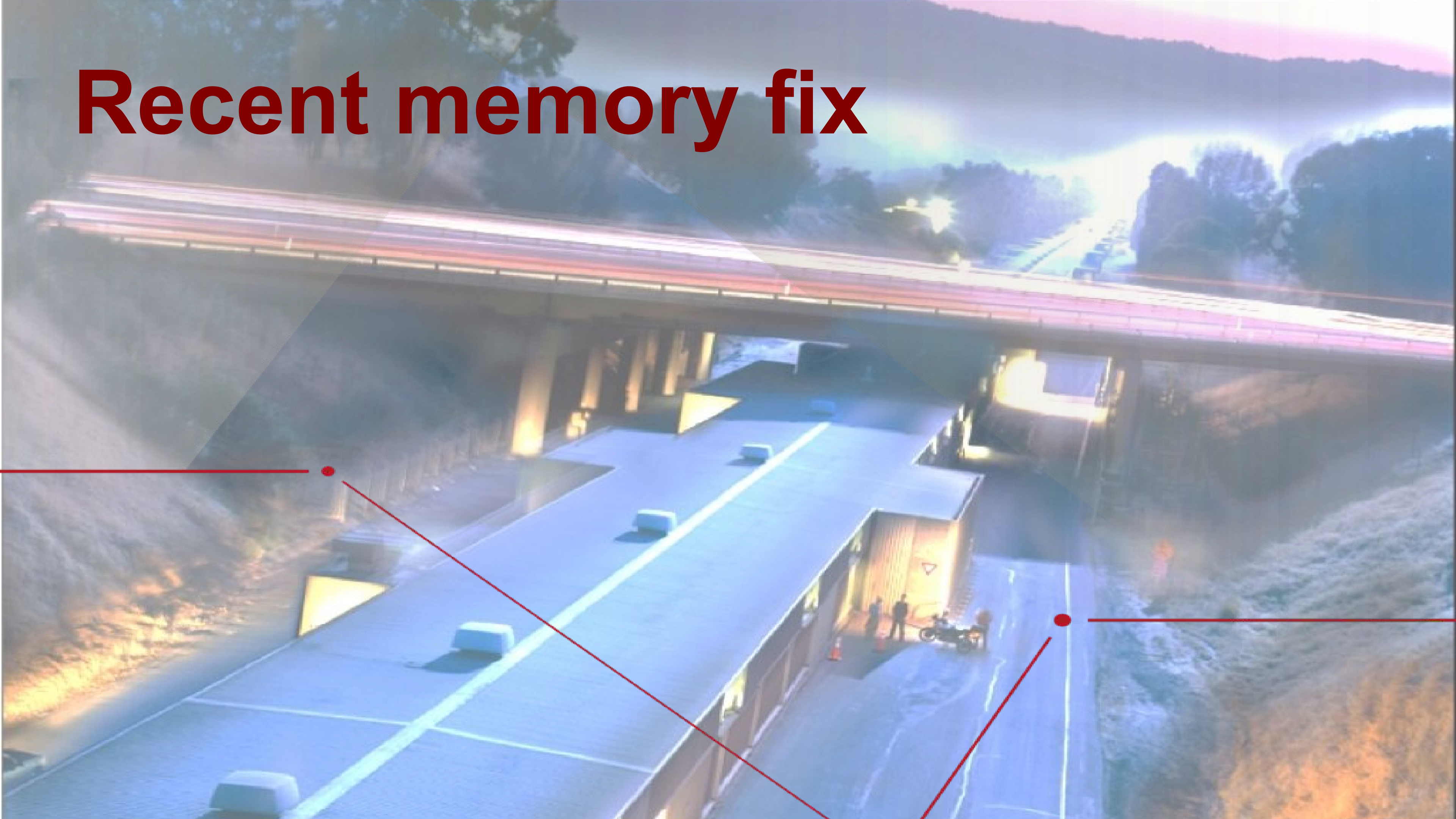
KNL vs KNC

We provide support for running G4 on KNC, https://goo.gl/qEFo6u , will update for KNL

Due to x86 binary compatibility, work-flow is tremendously simplified

System	Time to completion (5k events)
Xeon E5-2620 @ 2.1 GHz (12x2 cores)	570 s
KNC (31s1P) @ 1.0 GHz (228 threads)	1000 s
KNL (7210, quadrant mode, MCDRAM only) @ 1.3 GHz (255 threads)	378 s (x3 improvement w.r.t. KNC)
KNL (shared library)	480 s (25% slower)

Recent memory fix



TL;DR Version

With a very large number of threads we saw large memory use in 10.3.p01 with Bertini: due to new transition region in physics lists that use Bertini at higher energies

After the Bertini fix for coalescence introduced in 10.3.ref04 the **memory footprint of Geant4 is well under control and better than 10.2.p03**

Now there is **no memory footprint dependence on the BERT/FTF transition**

Full details here: https://goo.gl/TmHaZC

Memory test strategy for Geant4

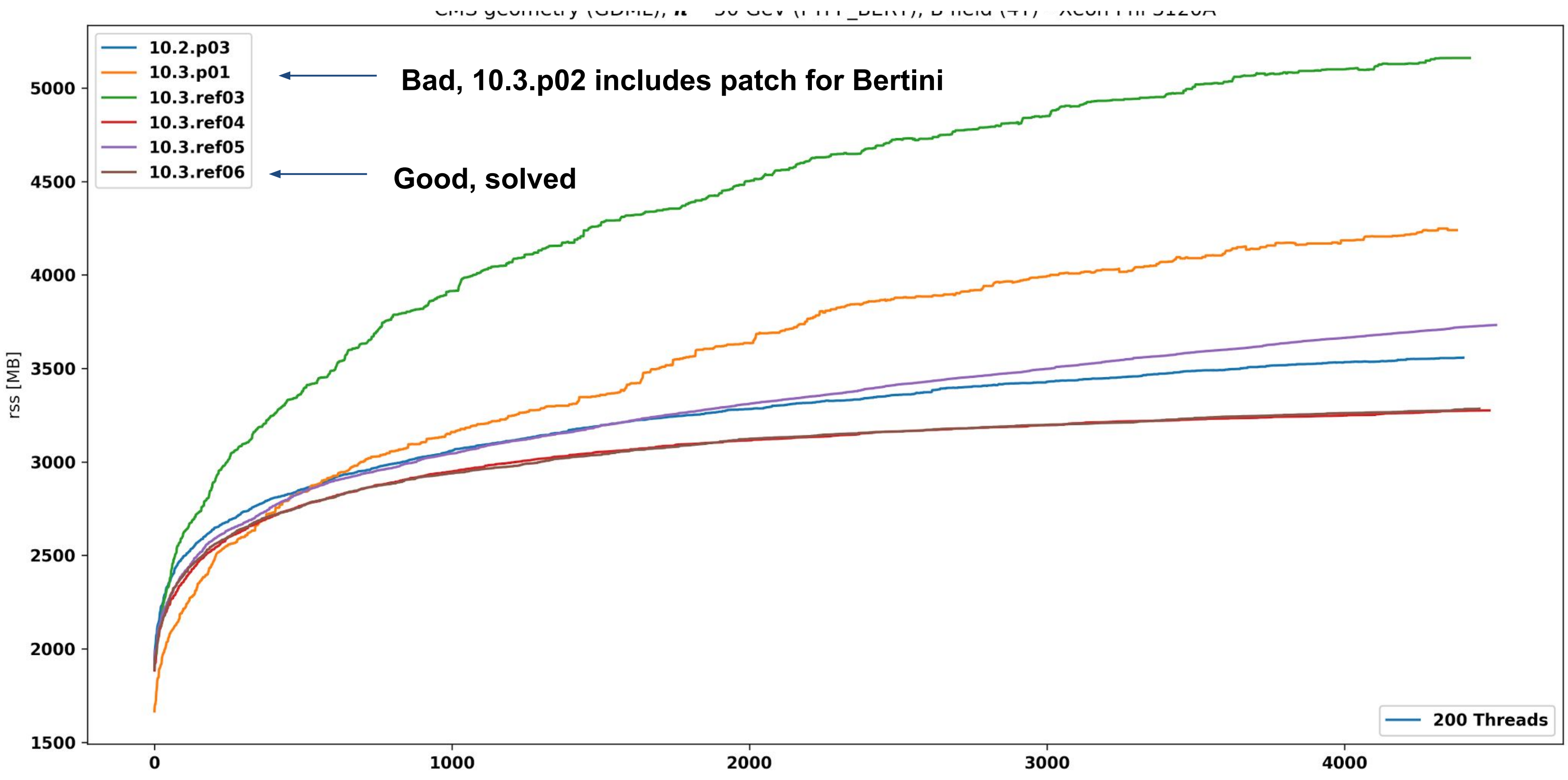
We measure memory in different ways:

1. Soon’s traditional plots “memory after first and after last events in sequential mode”: sensitive to memory leaks (from difference last-first). Not sensitive to transient spikes in memory use (e.g. collections being freed at the end of event)

2. Andrea’s “average memory use per thread”: verify that we share memory in MT mode as expected

3. **NEW** Andrea’s “very long MT test with 200 threads”. Memory measured ~5000 times “at random” during a >24 hours job. Spot transient spikes in memory use

Memory usage with many threads



Bad, 10.3.p02 includes patch for Bertini

Good, solved

Future Activities

Next steps

The following two activities were scheduled for 10.4 and will be postponed to 2017+:

1. Migration to C++11/14 threading model
2. Better integration with task based frameworks (TBB) via use of workspaces concepts