Demonstrators for Geant4MT

John Apostolakis

Overview

- Geant4-MT
  - Introduction
- Potential use in CMS
  - Adaptations required
- Seeking to use instructions better
  - Track parallelism benefit ?
- Summary

G4MT overview in 1 slide

- G4MT today parallelizes at the event level
  - Static scheduling of events, per-event RNG seed
- Goal: 100% reproducibility – or as good as G4
- Threads share RO data: geometry and EM XS
  - A small set of classes is involved
  - Different instances of most classes on each thread
- Cost of extra worker (thread) is small fraction of the total memory footprint

Some numbers

- Linear Speedup:
  - Excellent scaling: N workers vs. 1 Worker
- CMS2008 detector model: “Full CMS”
  - Baseline ~250MB
  - To add one worker (thread) costs an extra 20MB
  - Compare to worker for G4MP extra ~50MB
- Issue: overhead 1 worker vs. sequential
  - Overhead of 35% (P. Canal), since reduced to 18%

Contributors

- Design & implementation is work of
  - **Xin Dong** (Northeastern Univ.),
  - supervised by Prof Gene Cooperman,
  - in collaboration with J.A.
- Scaling studies and bottleneck investigations
  - by Andrzej Nowak (Openlab)
  - and recently Ph. Canal (FNAL).
- Documentation and refinement for release
  - by Daniel Brandt (SLAC).

Potential near-term use

- Discussions with CMS on model for multi-threaded 'all-in-one' event processing
  - Spans Ev. Gen. / Sim. / Trigger / Reco. / Analysis
  - Looking to be finished & validated by LHC 2014 restart.

  Brought to G4 Technical Forum, 27 Mar 2012
- Examine whether G4-MT can be adapted to fit into this use.

Co-working in one concurrency model

- All parts must work within one concurrency model
  - Share work as requested
  - Would need to adapt to ‘dispatcher’ parallelism, with control external to G4-MT
- Expect this to be achievable without large changes in G4-MT
  - Foresee startup of a fixed number of threads
  - Use existing code to handle, with small adaptations / refinement

Adapting to use-on-demand

- Expect to accommodate it by refining the Parallel Run Manager
  - Separate the ‘core’ elements, needed for G4MT, from the ones which control the event loop
  - Similar to creation of ‘RunManagerKernel’ class in sequential Geant4
- Part of next iteration of G4MT
  - After the update release based on 9.5-p01 (15 April)

Adapting to experiment framework

- In addition to the Event loop control, a number of other aspects are revised in G4MT:
  - Handling Replicas and Parameterised volumes
  - Creation of sensitive detectors for each worker
  - Merging of output into a single hit collection
- Guide to adapt an application is in “Geant4MT – User’s Guide”, in the source download
  - Today it covers only simple standalone programs
  - Procedure improved in 2011 by D. Brandt (SLAC)
  - Expect further adaptation, simplification.

Parallelism for better cache use?

- Instruction fetch is a significant bottleneck
  - Studies by David Leventhal, Daniel Kruse
- Can sequential or parallel G4 improve?
  - In sequential code need to work continuously on the same particle type and/or part of the geometry
  - Threads could specialize on a particle type (or 2) at the cost of extra queuing (communication) if the simulation was parallel at track-level

Estimating benefit

- Investigate multiple queues in sequential Geant4
  - An implementation was developed by Makoto Asai, using three stacks per particle type
- First tests did not show benefit
  - No speedup in an Atlas simulation use case
  - Need to understand if this is an implementation issue, a fundamental limitation or a different bottleneck.
- Is ‘hunch’ correct that better code locality can be achieved this way?

Summary: short term goals

- Adapt to external dispatcher parallelism
  - Identify what changes would be required
  - Proceed in collaboration with G4 experts
- Investigate potential for improved use of caches by ‘bunching’ particles by type
  - Use sequential Geant4 as test-bed
  - Check effect of reordering tracks – is cache use changed ? Does CPU time profile change?

Reducing the memory Footprint

Few key classes are changed to reduce memory footprint (TMR)

Share **Read-Only** (RO) data

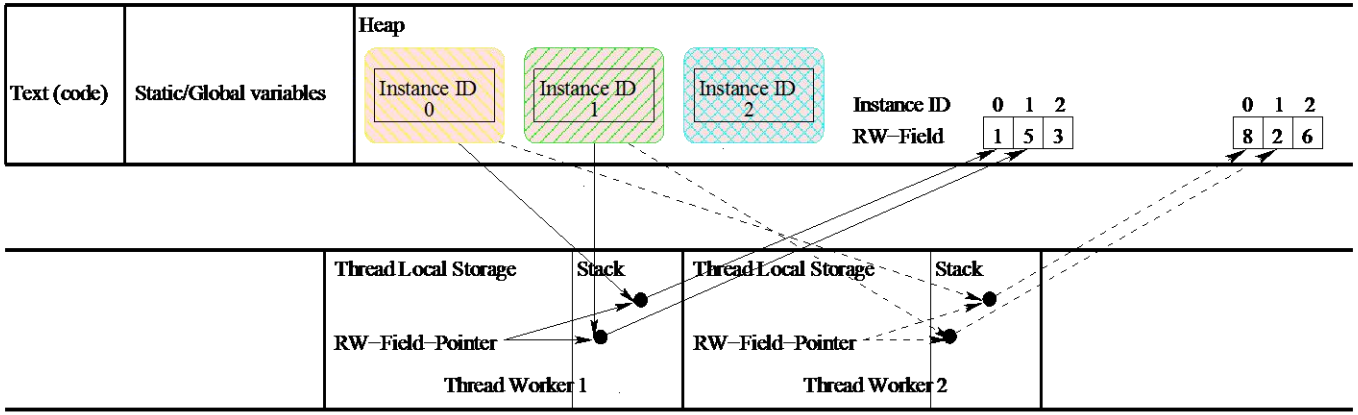
Separate **Read-Write** (RW) data

TMR Implementation Example

If those object instances are relatively read-only, just share them. Otherwise, reorganize the data structure as follows.

```
class volume                          __thread RW_t *rw_array;
{                                     class volume {
   RW_t rw;                              int instanceID;
   RD_t rd;                              RD_t rd;}
}                                     #define rw (rw_array[instanceID])
__thread vector< volume*> store;      vector< volume*> store;
```

Corresponding Data Model



G4mt: some characteristics

- Chose to implement parallelisation at the event level
  - Simpler
  - Less overhead if $N_{events} >> N_{threads}$: More efficient
- Can be used today for sub-event parallelism
  - Treating a subset of primary particles as an 'event'
- Very good speedup for 40+ workers

Existing programs & tests

- Examples
  - ExampleN02: simple tracker
  - “FullCMS”: uses CMS 2008 geometry from GDML file (CMS hits are NOT created.)
- Uses thread local storage (gcc __thread)
  - Works only on Linux (today).
  - Could be extended simply within this model (e.g. Windows dllspec thread).
- Tests and safeguards: ensuring correctness
  - Access of Thread-local storage from other threads can be made to raise alert signal
  - Used for testing today
  - Potential to extend to production use.

Porting an application to G4 9.4MT

- Requires checking/revising User classes:
  - UserXXXAction: Stepping Run, Track
  - Sensitive Detector and Hit classes,
  - Parameterisation & Shower Library.
- Port to using MT RunManager from ‘driver’
  - Review handling of I/O
- Actions required
  - Ensuring they are thread safe (or adapting them).
  - Inspecting Sensitive Detectors
  - Making test runs with ‘guard’ configuration and/or helgrind to check for problems.

Status

- Alpha code
  - In Geant4 SVN on a branch, but not integrated yet
- Each G4MT release created from sequential
  - Check singletons, global variables
  - Merge changes, check, fix, ..
  - Tests: 1-worker vs. sequential – check exact reproducibility of RNG at end of job
- SLAC team will create MT releases during 2012.

Near-term issues

- More tests required for production quality
  - Skeleton of test suite identified for test coverage
- Capture/Improve process
  - for creation of MT release
- Eliminate overhead of calls to get ‘thread Id’
  - Map thread_id => worker instance kept by G4 ?
- Consolidation: simplify and expose MT code
  - Embed __thread
  - Move MT to mainstream G4 in 2013

Resources for G4MT

- Status and porting to new versions, talks at G4 Workshop 2011 (Xin Dong, Gene Cooperman) – also at http://bit.ly/wPja8h
- Geant4MT Performance Studies, Ph. Canal
- Geant4MT: Maintenance for Both Geant4 and Geant4MT, G4 Workshop 2010, XD, GC, JA.
- Auxiliary
  - Hints for Writing Thread-Safe Code, Gene C.
  - Adapting the user code to use Geant4-MT , Xing D. (outdated, as process has been simplified.)