Parallelisation Session (2B)

J. Apostolakis

C++ 11 Threads – Marc Paterno

- <std:: > thread has potential for portability
- New capabilities – move from C to C++
  - Full checking of arguments
  - C++ type mutex locks: safe for exceptions
  - Sentry object to guard resource
- Status: gcc 4.7.1 with flag –std=c++11
  - Has std::thread
  - Does **not** have Thread Local Storage ‘thread_local’

On-demand simulation: Similar requests

- CMS: to co-work with other modules
- ATLAS ISF: dispatch G4Event to workers

Impact – separate stages

- Initialisation at start of N workers – or rendevous of work unit and workspace ?

Need to clarify adaptation from thread to ‘worker’ model – what impact ?

G4 - GPU efforts: external + G4-related

- GPU efforts external to G4 Collaboration
  - hGATE project, gamma (2011) and e-
  - G4MCD (Germany), both gamma and e-
- Use case medical physics phantoms
  - Simple geometry
- G4 related efforts
  - CERN / FNAL: focus on HEP: energy & intensity frontier
  - SLAC/G4 Japan (with Stanford, Nvidia): medical: dose estimate

Finnish/CERN/GSoC:

Dhruva T.B. (GSoC 2012), Otto Seiskari (2010)

- Separate code with 4 solid types
- Existing Tests – Otto S.
  - Simple (2 volumes) to simplified CMS (1600)
- Automated Checks
- New Navigation algorithm
  - Revamped algorithm: New way of Computing Steps
  - Taking GPU execution model into account
  - Ongoing effort to make robust & optimise

Philippe Canal, Soon Yung Jun

Performance Numbers

Ratio of Processing Time: CPU/GPU	Kernel (Computation) Only	Kernel + Data Transfer
Stepper	67	29
Geometric limit of step length	74	32
Particle Transportation (no physics)	33	27
EM Physics	88	9
Particle Transportation with EM Physics	34	30

Magnetic Field

Stanford ICME, SLAC, G4-Japan Collaboration
supported by NVIDIA

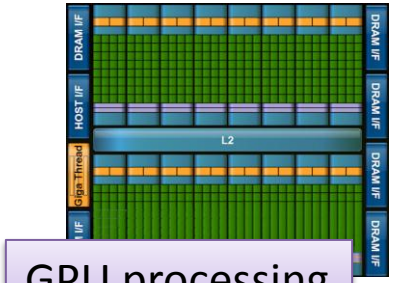
Nicholas Henderson ,
Koichi Murakami

Project Goal

Dose calculation for radiation therapy

- GPU-powered
  - parallel processing with *CUDA*
  - boost-up calculation speed
- voxel geometry
  - including DICOM interface
  - material : water with variable densities
- limited Geant4 EM physic processes
  - electron/positron/gamma
  - medical energy rage (< 10-100 MeV)
- scoring dose in each voxel

DCMTK

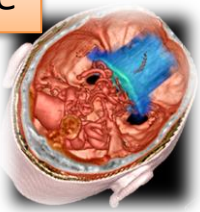


GPU processing

Dose

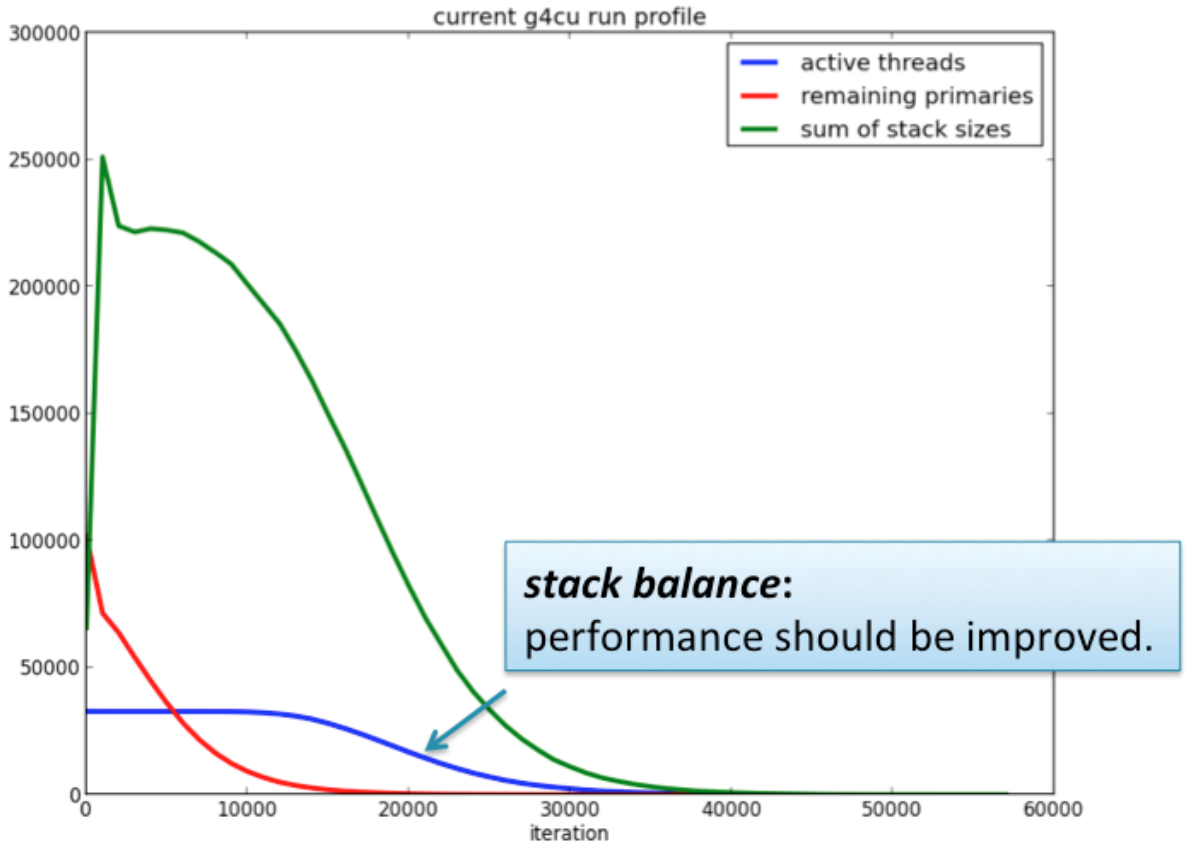
DICOMRT-Dose

gMocren



Run profile

- Run profile with current fake physics processes and no stack balancing



GPU: *Next development topics*

CERN / FNAL: Coordinate and collaborate - moving to common code base

- (CERN) Improve Voxel/Combined Navigation
  - Seek compatible physics models from potential external partners
- (FNAL): Focus on Performance, extension of physics
  - ATLAS field lookup (costlier than CMS), other motion integrators
  - Use streams to overlap work and communication
  - Collaborate with CPU to share work
  - Investigate geometry algorithms alternatives
  - Extend geometry (shapes) and EM physics processes.

KEK/SLAC:

- Port physics processes Eloss, photon interactions,
- Optimize stack/memory management

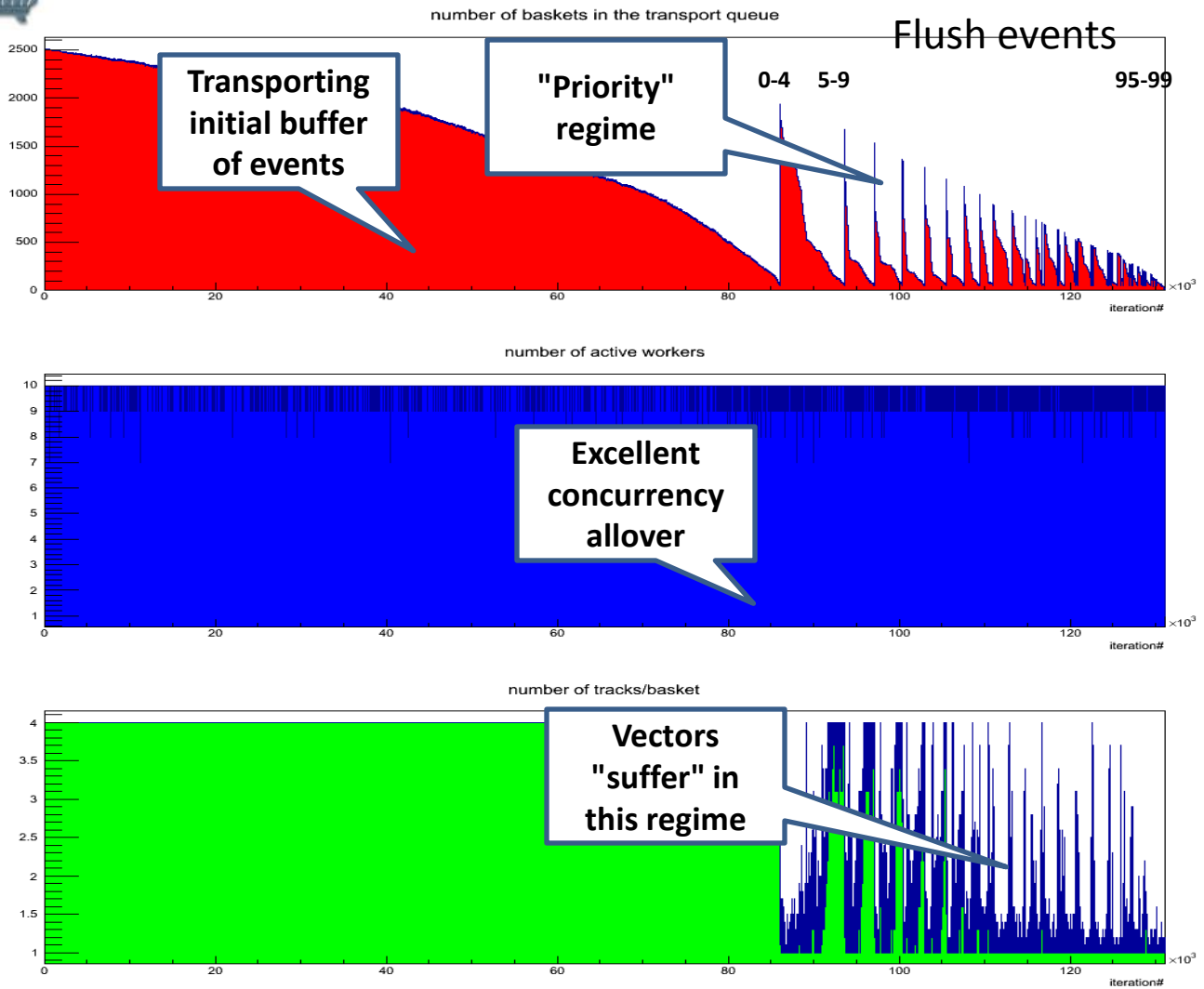
Each to keep other informed on work items – try to avoid duplication where possible;

- At a later stage, could potentially share source

Andrei Gheata

Prototype for Vectors

Rene Brun,
Federico Carminati



Vector Prototype: Next steps

- Include hits, digitization and I/O in the prototype
  - Factories allow contiguous pre-allocation and re-usage of user structures
- Introduce realistic EM physics
  - estimate memory requirements
- Look further to auto-vectorization options
  - New compiler features, Intel **Cilk+** array notation (icc, gcc 4.7 branch, *proposed* for C++ standardisation)
  - Check impact of vector-friendly data restructuring
    - Vector of objects -> object with vectors
- Push vectors lower level
  - Geometry and physics models are main candidates

Outlook

- Hardware future
  - Wider Vector units, more cores in CPU
  - GPUs: architectures evolving,
  - Intel MIC arch.: ~60 cores, 4 threads, wider vector
- Software platform & future – the **toughest** issue
  - CUDA but more flexible, easier to debug (than OpenCL)
  - Auto-vectorising compilers, CILK+ array syntax
- Not choice between Multi-threading & SIMD
  - **Both** are required for performance