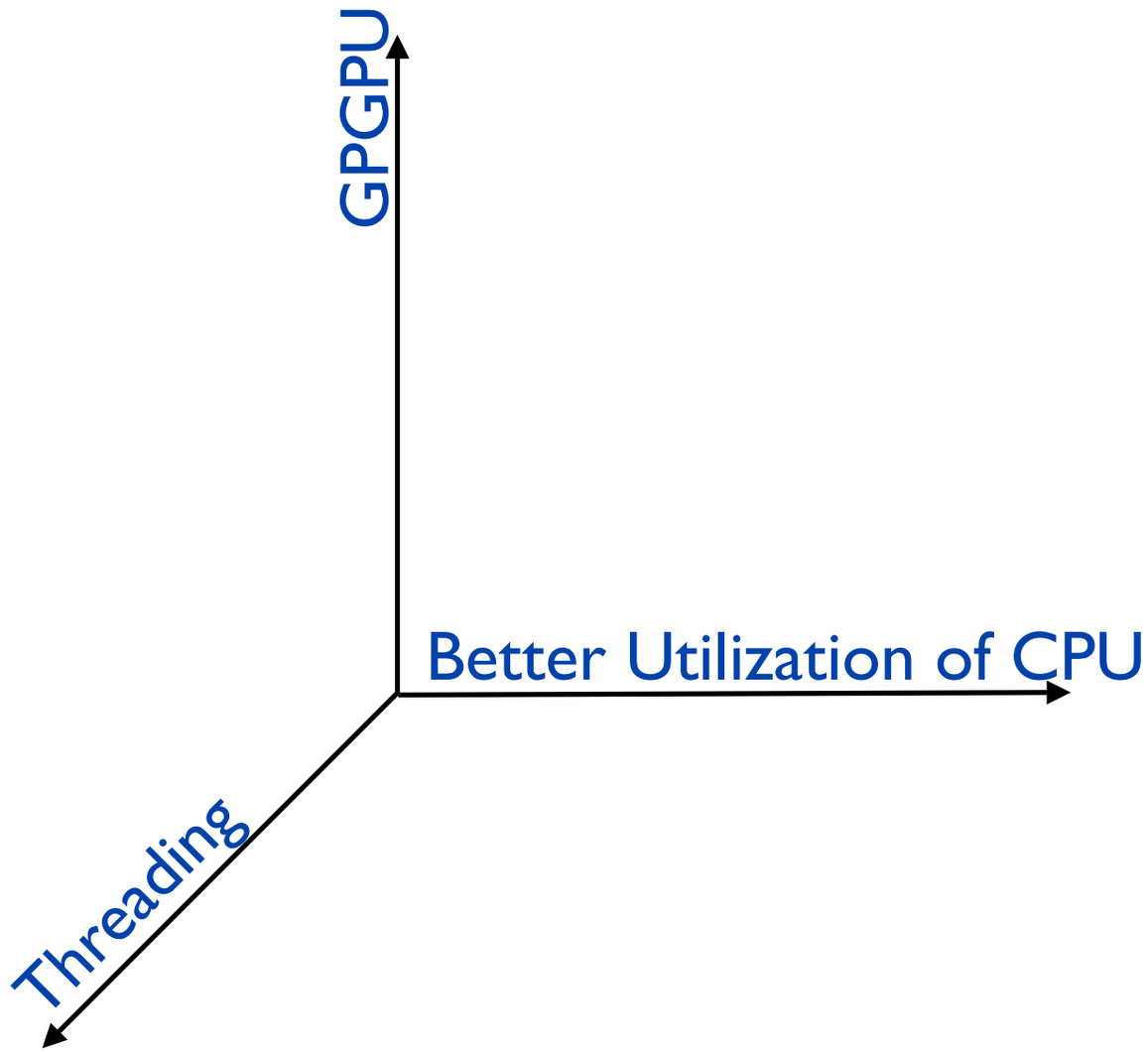
Perspectives on Parallelism in HEP Frameworks

Christopher Jones

On behalf of the CMS Offline Organization

Future Performance

Three axes for better performance on future hardware



Future Performance

CPU

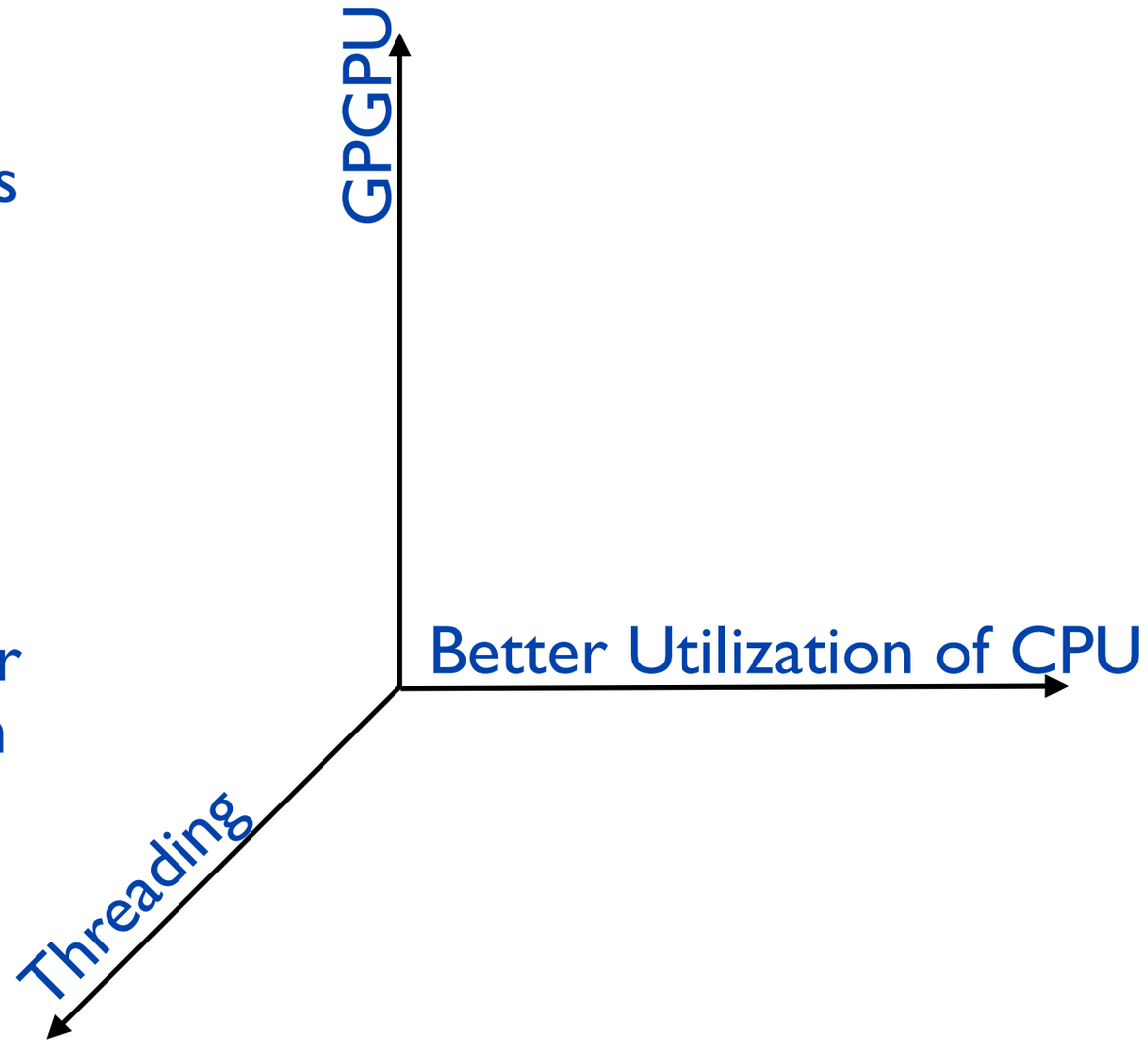
HEP code does a poor job of using the Cores
Lots of L1, L2 and L3 cache misses
Instruction level parallelism
Vectorization instructions
Can be utilized via C++
Requires carefully designed data structures
Good for very fine grained parallelism

GPGPU

Handled via custom languages
Needs special handling of memory
Requires the presence of the co-processor
Good for very medium grained parallelism

Threading

Requires carefully designed algorithms
Good for coarse grained parallelism



Present Application

CMS uses one application for all event processing

- Particle generation
- Simulation
- Online High Level Trigger
- Reconstruction
- Analysis

Each event processing algorithm is encapsulated into a 'module'

- Geant4 is wrapped by one particular module

CMS' application controls the processing

- It decides which event to process next
- It decides the order to call each module and passes it the proper event
  - Application calls specific Geant4 functions when it is Geant4's turn to do work

Multithreading

Plan for new multithreaded application

Will process multiple events simultaneously

Will run multiple modules processing the same event simultaneously

This will all be controlled explicitly by the application

All parts need to work within one concurrency model

Present application is memory resource limited

- in future may not be able to afford 2GB / CPU core

Each additional thread requires its own stack

- default size on SL5 is 10MB/stack

One concurrency model will allow use of only one thread pool

- minimizes memory
- avoids oversubscribing available cores

Interested in Geant-MT if it can fit with this working model

Where concurrency is controlled by the experiment's application

- E.g. Application calls specific Geant methods at proper time from a thread controlled by application

Technologies

Reviewing several high level threading technologies

libdispatch

Open source port of Apple Inc's system

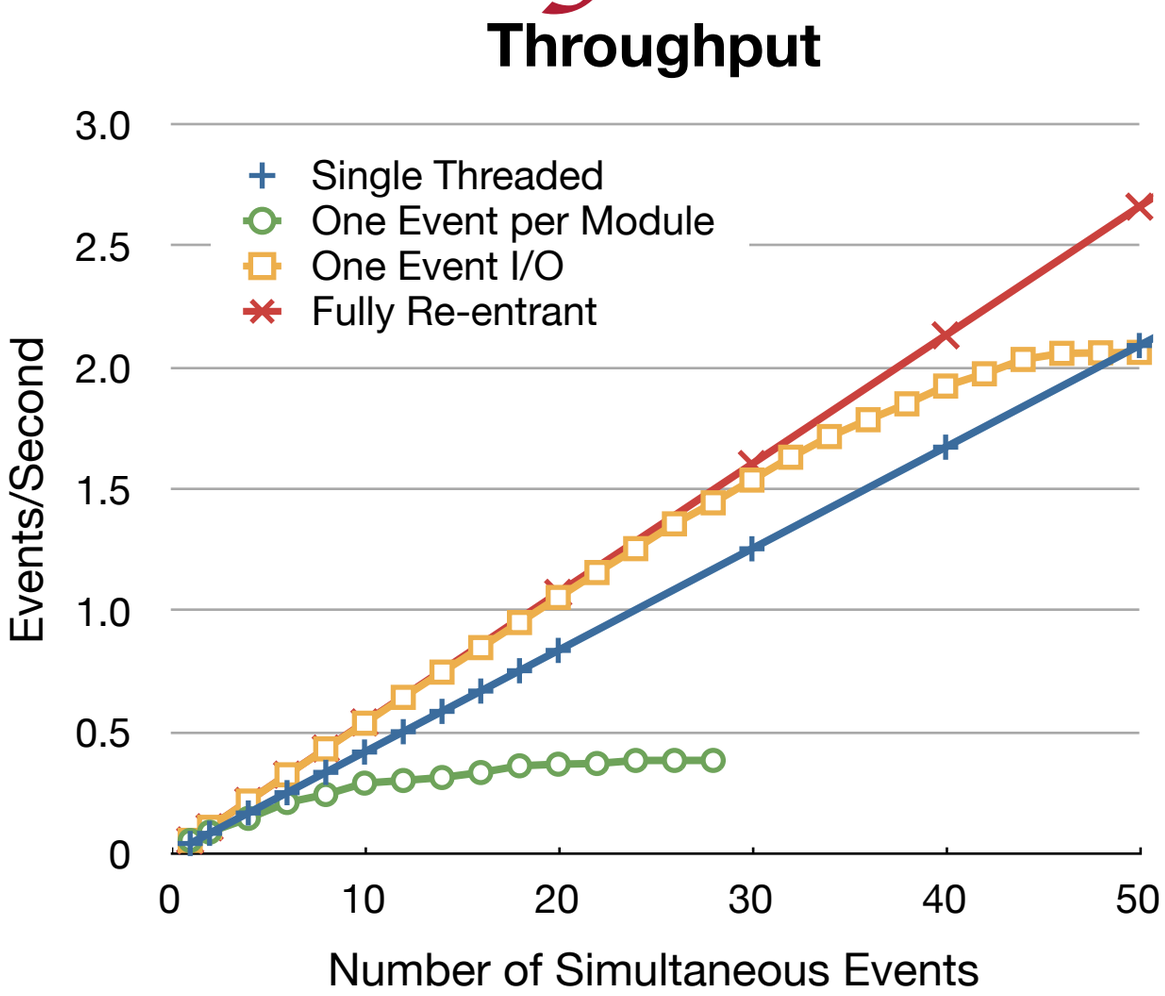
OpenMP

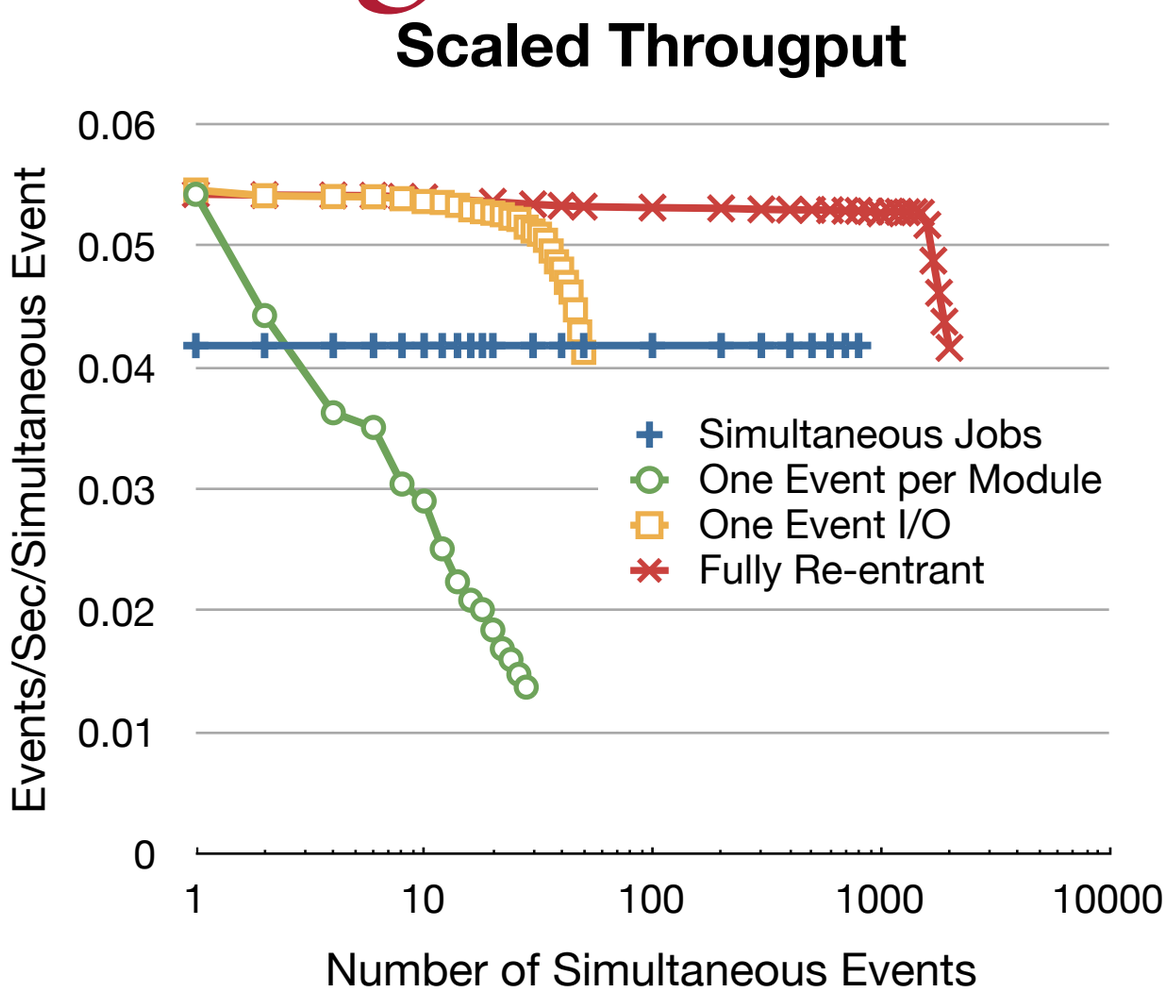
Built into gcc and intel compilers

Intel's Thread Building Blocks

Cross-platform C++ library

libdispatch Scaling





All Modules are calling usleep

One event per module slower after 2 simultaneous events (se)

One event I/O turns over at 25se and stops growing at 44se

Fully re-entrant stays 30% faster till runs out of system threads

Single threaded runs out of memory at 800se

Time Scale

Work is beginning now

Needs to be finished and validated before LHC 2014 restart