Geant4 FNAL Team:
Performance and Robustness Group

Mark Fischler, Jim Kowalkowski, Marc Paterno

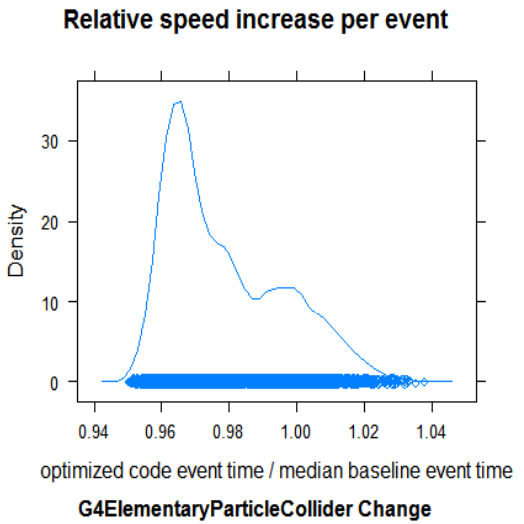
July 2, 2008

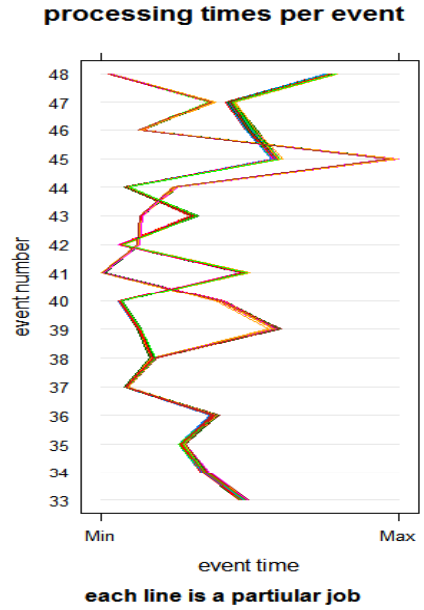
Performance and Robustness only – FNAL activities NOT included in this presentation:

- Hadronic improvement and Validation
- Physics Validation
- See Sunanda Banerjee, Julia Yarba, Daniel Elvira

G4 Performance Highlights – A Quick Summary

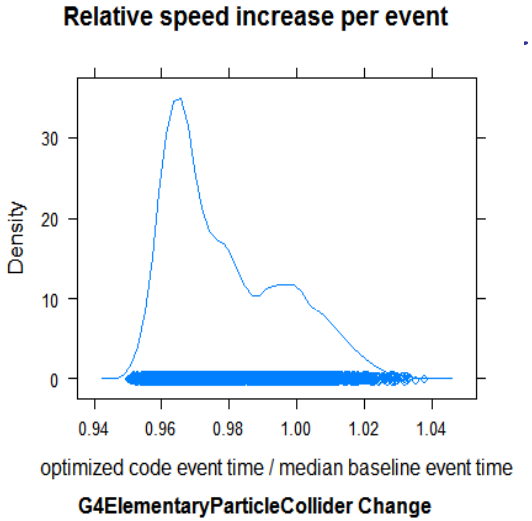
- Database system for recording performance run data (a work in progress). Useful for finding target for improvements and accurately measuring speed increases. Includes data from
  - Profiling results
  - Event timings
  - Program and environment configuration
- G4QHadron classes
  - reduction of “memory churn” (CMS desirable)
  - enhancement increased performance by > 5%
- QGSP-BERT: G4ElementaryParticleCollider restructuring
  - Performance increase ~ 4%
  - Reduced lines of code by about 40 printed pages
- Still addressing irreproducibility issues we found:
  - Bifurcation in event processing times across many of the same jobs at event 38
  - Different amount of random numbers drawn from generator depending on architecture

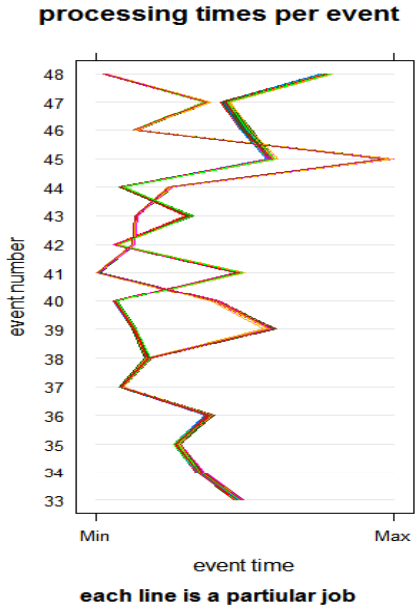




One Infrastructure Slide: Database System

- Database system for recording performance run data
- Allows SQL queries and packaged graphic displays of information (see pics at right)
- Not yet ready for use by others, but getting there.
- Useful for finding targets for improvements and accurately measuring speed increases. Includes data from
  - Profiling results
  - Event timings
  - Program and environment configuration



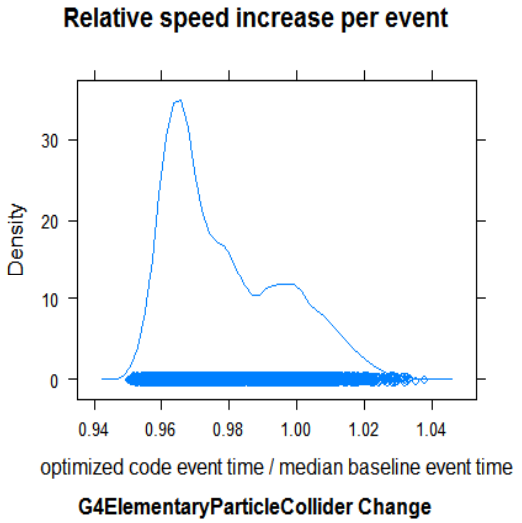


Performance Tweaks

- Bertini Cascade
  - Found classes with many data members, none of which ever changed
    - One static instance therefor would suffice
  - Observed that ctors for these were eating >3% of the total time!
  - Restructured to eliminated the need for those ctor calls

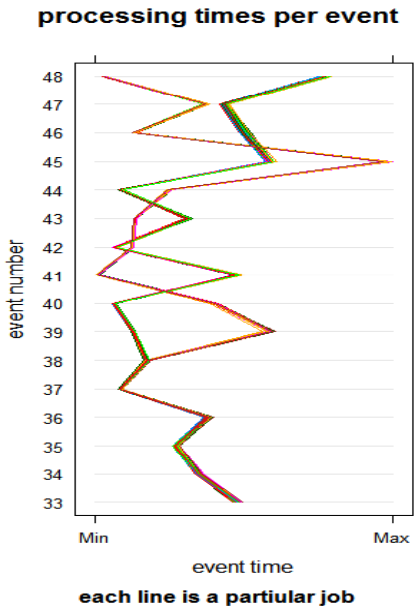
- Result:
  - 40 pages of code collapes to 2
  - 3.5 – 4 % speedup

- Status: D. Wright has validated that the physics results are unchanged



Robustness: Reproducibility Mysteries

- Had observed bifurcation of event-time pattern
  - Was reproducible across multiple grid submissions
  - Bifurcation happened on same event whether CPU was Intel or AMD
  - This went away!
    - Ascribing it to some change in grid environment
    - Can't invstigate further



Robustness: Reproducibility Mysteries

- New mystery:
  - Encountered while trying to investigate timing bifurcation
- Sequence of randoms consumed varies according to AMD or Intel CPU!
  - Reproducible
  - Happens a number of events into job
  - Consistent with one stream requiring an extra number and then the meanings of the delivered randoms change
- When we learn why this is happening, it might expose some important issue (or perhaps something trivial)

Code Quality: CHIPS

- Observation: The CHIPS code is organized in idiosyncratic way
  - Probably hard to maintain if original author is not on board
  - Very hard to attack for performance improvements
  - Could be excellent target for “refactoring” and design improvement
    - One important prerequisite is good unit test and validation capability, so that we can know that the changes don’t “break” anything

Code Quality: CHIPS

- This is the sort of thing Marc Paterno and Jim. Kowalkoski have done very well for several experiments
  - Code ends up more maintainable
  - Sometimes big speed increases too
- Estimate: 3 solid weeks including analysis and restructuring
  - Can't possibly start until mid August at earliest
  - If tests are inadequate, somebody ought to provide better validation before starting on the re-design
- This may or may not be worth doing