Performance and Numerical Stability

Fermilab's Geant4 Performance Subgroup:
Mark Fischler Jim Kowalkowski Marc Paterno

14 September 2007

Profiling tools

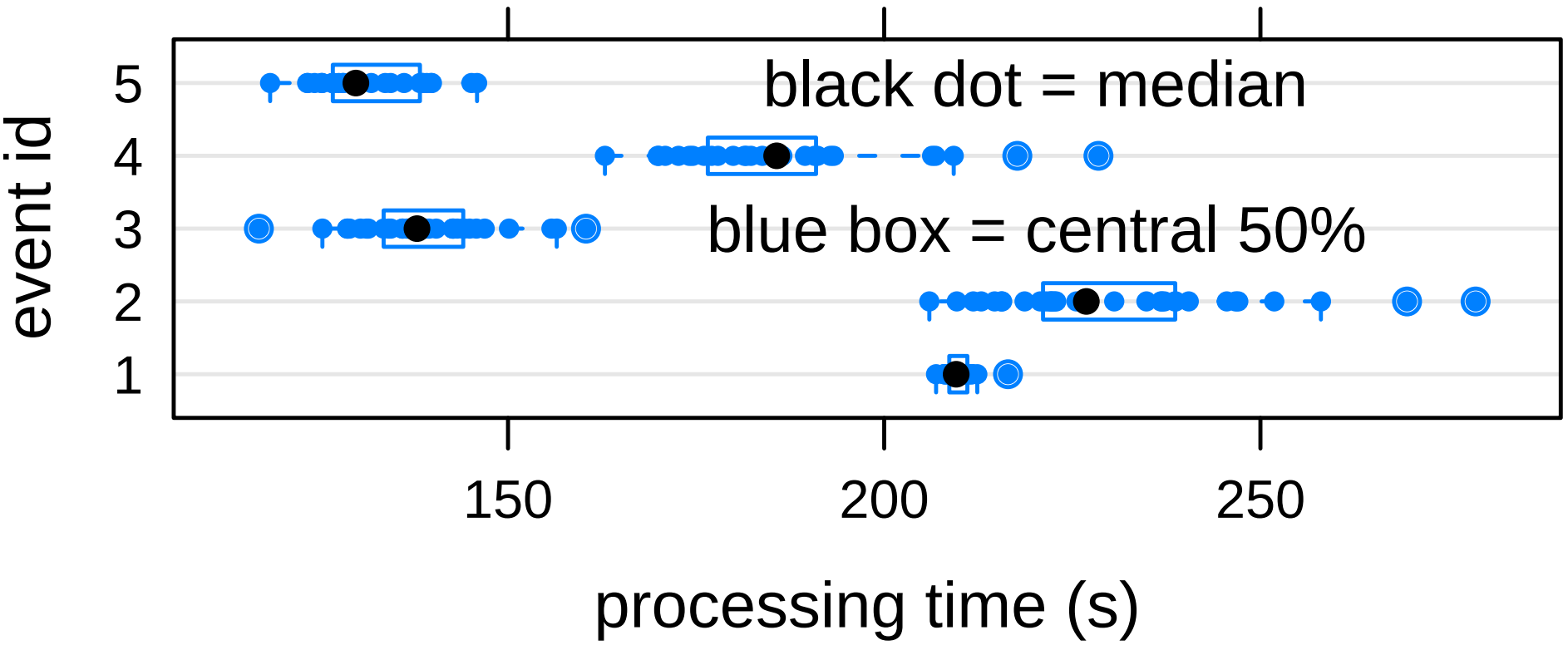
- We used our own `SimpleProfiler` to collect data
  - We have implemented a *sampling profiler*, because we believe that is the best way to obtain unbiased measurements of the program of interest
  - 100 times per second, we capture:
    - the address of the current function, and
    - the address of each function in the call stack
  - It thus collects full call path information, not provided by other tools (*e.g.* `gprof`, when obtaining sampling information).
  - In post-processing, we determine the function names and library location for each function we observed
  - We load all the information into an SQLite3 database
  - We also have scripts to create Graphviz plots summarizing the function call information

Profiling results

- We have completed two profiling studies, using the full CMS simulation (and *QGSP_EMV* physics lists) as a vehicle for measurements. We've reported on these previously, so here we'll only summarize.
- Each resulted in suggested changes to Geant4 source code; in each case similar (but not identical) modifications were implemented.
- (Using **G4 8.2p1**) In *G4ParticleDefinition* and *G4HadronicProcess*, we suggested refactoring that improved code speed $\sim$ 4%. Code changes went into **G4 8.3**.
- (Using **G4 8.3**, pre-release patch) In *G4QNucleus*, we suggested the removal of needless memory allocations; such removal improved code speed $\sim$ 1.5%. Code changes went into **G4 9.0**

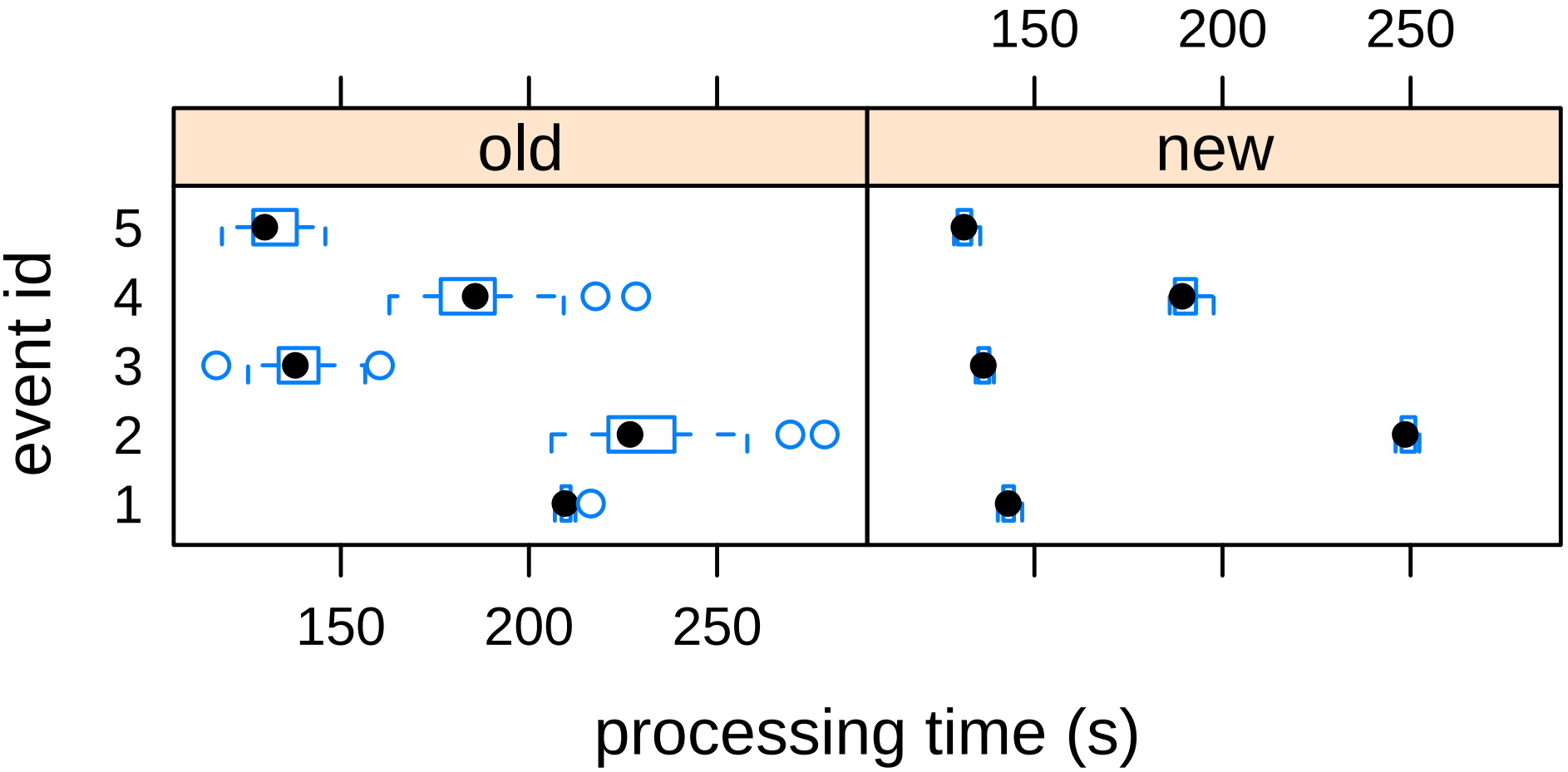
Reproducibility of timing studies

- To compare the timing of different program configurations (*e.g.* physics lists or code changes) we need to know the magnitude of the uncertainties involved in timing a single program execution.
- Measurements showed execution times were *not* reproducible: repeated runs of the same event differed.



Improved timing reproducibility

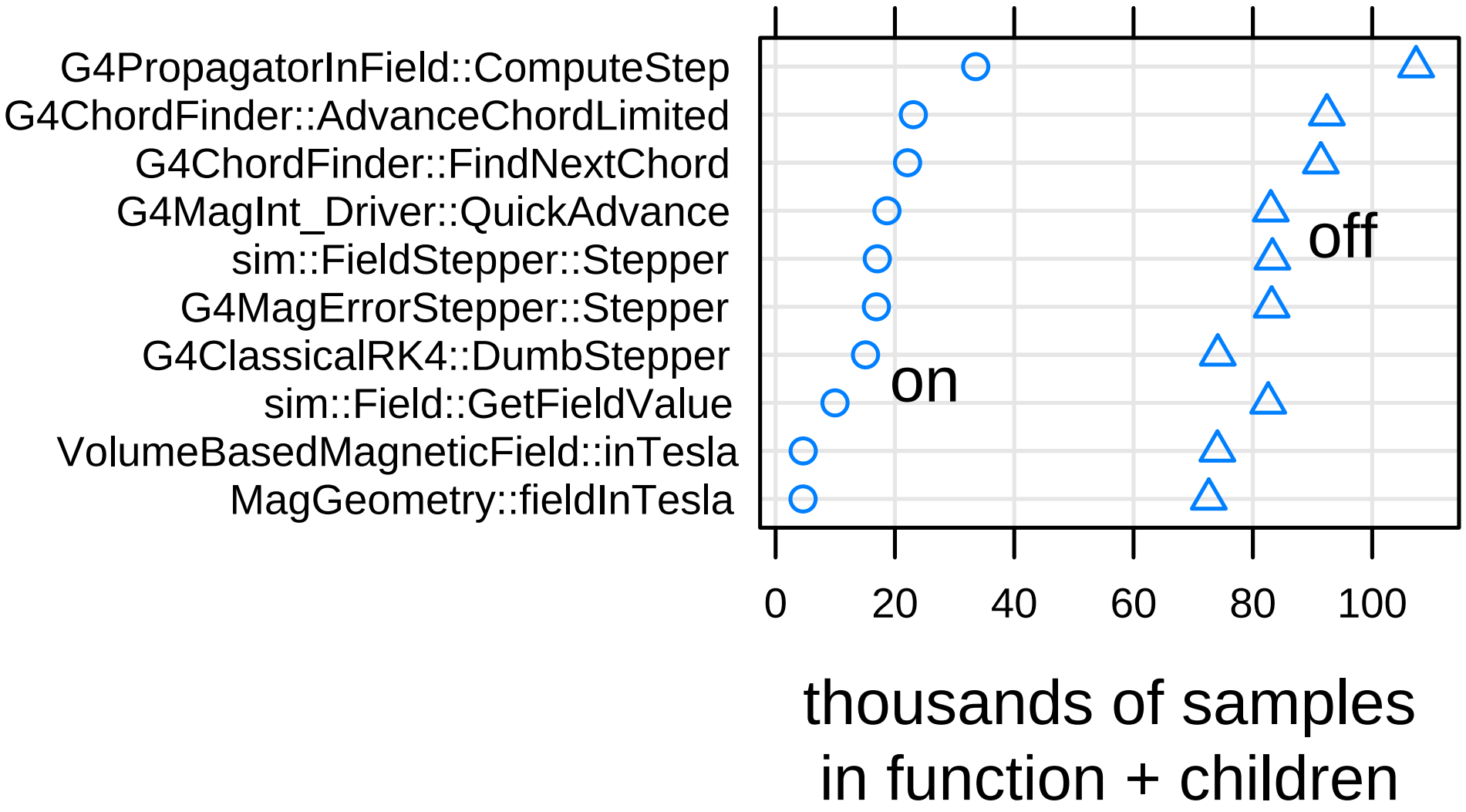
- $<$ 1% timing comparisions needed improved reproducibility.
- We found that the version of **HepMC** used by CMS (2.00.02) was the major source of timing irreproducibility
- Moving to **HepMC** (2.01.06) reduced the spread to $\sim$ 1%.



Magnetic field caching

- We have studied the effect of magnetic field caching.
- Measured the time taken for a sequence of 10 function calls in particle tracking code.
  - Reproducible timing results gives us the ability to compare between runs.
- The following plot shows number of profiling samples obtained for each of the 10 functions—proportional to cumulative function time.
  - We show *cumulative* function time, *i.e.* time taken by a function and all functions it calls,
  - compare two runs of the program, one with with field caching *on*, the other with it *off*.

Timing: field cache “on” *vs.* “off”



Field caching observations

- For Geant4 team: it would be valuable to have a stepper that is optimized with the expectation of having a fast field calculation (which can be obtained through caching)—or, if one already exists, to advertise it as such to the users.
- For Geant4 users: careful field caching can dramatically speed up your program with negligible physics impact.

Effort on numerical stability

- The floating-point unit can generate *invalid operand faults* to signal illegal data supplied for an operation, *e.g.* $\sqrt{-1}$.
- The default behavior under these circumstances is to generate a NaN.
- Using a modified version of our profiling tool, we are able to identify each such case, and determine the call path that lead to it.
- We have uncovered a problem in *G4Torus* that resulted in the calculation of square roots of negative numbers.
  - Problem recently communicated to Tatiana Nikitina, maintainer of this code.
  - She's working on a patch for us to test.
- After removing the use of *G4Torus* from the CMS simulation, we find neither invalid operand faults, nor divide by zero faults, nor overflows, in processing 50 events.

Conclusion

- In further profiling efforts we will move to **G4 9.0**.
- We will continue to concentrate our profiling efforts on functions taking 1-5% of total program time.
- Previous difficulties with NaN generation *may* be solved.
  - Further study is required.