

Geant4 Performance Studies

using the CMS simulation

Mark Fischler Jim Kowalkowski Marc Paterno

17 April 2007

Our plans

- Use (primarily) the **cmsRun** program, configured by “standard” CMS configurations, to test the robustness and speed of Geant4
- Identify places for improvement in the Geant4 code
- Design and implement improvements
- Feed those improvements back to the Geant4 team

We are making modifications to our local copy of the Geant4 code, and may also consider suggesting changes to the CMS code.

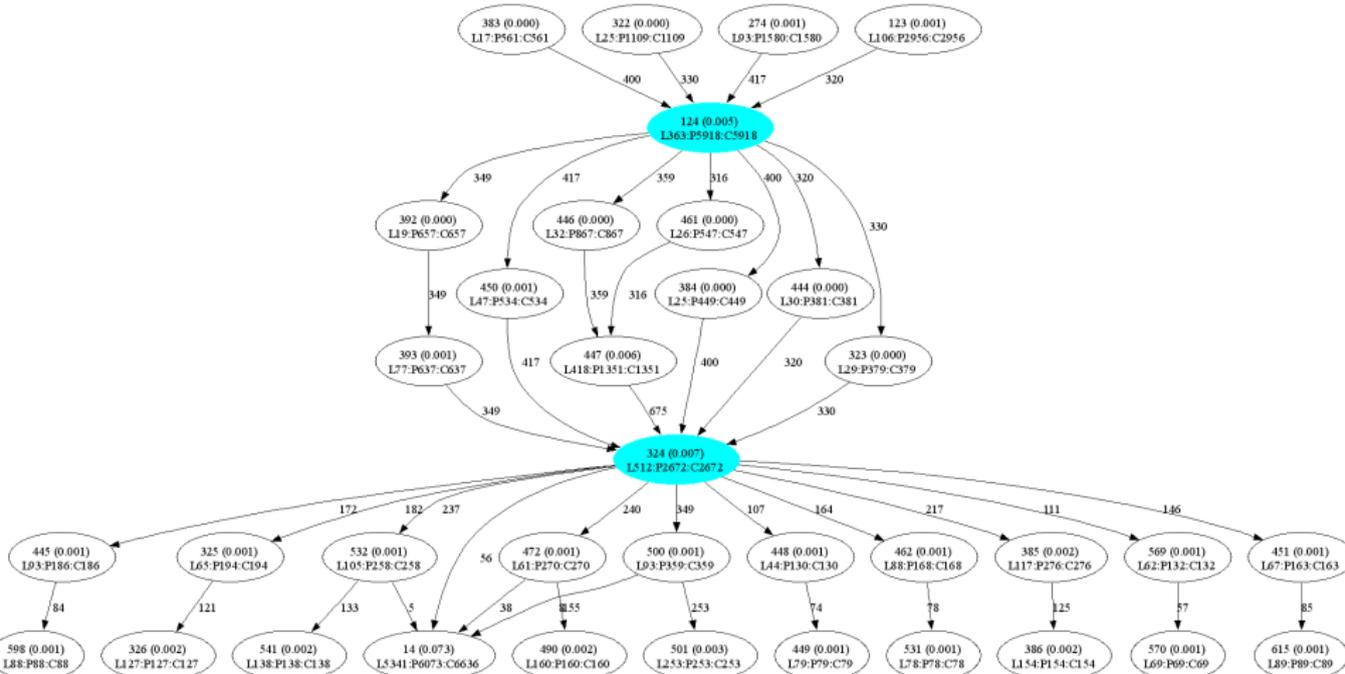
We are still becoming familiar with the code.

First step

- We profiled one program: the CMS simulation configured for standard CMS physics validation.
- We used the `SimpleProfiler` to collect data
 - `SimpleProfiler` was written by one of us (JBK)
 - It is a very low-impact sampling profiler
 - It provides full `call path` information, not provided by other tools (e.g. `gprof`, when obtaining sampling information).
 - There are few tools to create graphical displays of the recorded information
 - These tools are in a “primitive” state, and are still being developed
- Today we will show the results of our first round of measuring, re-coding, and measuring again.

Partial call graph

ID	function name	time
124	G4CrossSectionDataStore::GetCrossSection	8.1%
324	G4HadronCrossSections::GetParticleCode	3.7%



Timing results, after change

Knowing where to look, inspection of the source code showed us a place where an improvement in OO design should lead to speed improvement. The results were good:

function name	before	after
G4CrossSectionDataStore::GetCrossSection	8.1%	4.2%
G4HadronCrossSections::GetParticleCode	3.7%	0.1%

- The function we rewrote has nearly disappeared from the profiling; all the functions *it* calls have *entirely* disappeared.
- The result is about a 4% speed-up in the program execution time.

Future plans (I)

- Do more of the same
 - Look for places to gain a few percent by small code changes
 - Solution is often to put behavior in the right class
 - Such changes clearly (from code inspection) do not change the program output
 - How much (speed) gain can we expect from such changes? Perhaps 15%
- Investigate algorithmic improvements (*e.g.* improve particle stepping algorithms)
 - Such changes are larger in scope
 - ... and harder to verify
 - ... but have larger potential gains

Future plans (II)

- Look into long initialization time
 - This might mean changes in Geant4 code
 - ... or changes in the CMS code that uses it
- Look into issues of numerical stability (*e.g.* NaN production)
 - Existing instability limits our ability to study performance
 - May be in Geant4 code, may be in CMS code
- We will also continually improve our measurement tools, with the goal of making them generally available.
 - The preliminary versions are already in the CMS code base.
 - Older versions are in use at D0 and CDF.
 - We propose to make the tools we are developing available to others.