Profiling gtobssim

Profiling one of my programs, I found some surprising things in gtobssim.

I am using ScienceTools-LATEST-1-4066, build x86_64-64bit-gcc47-Optimized.

Execution time

I am simulating a single point source, with this xml file:

```
<source_library title="Library">
  <source flux="0.667283950775" name="bn131231198">
    <spectrum escale="MeV">
    <particle name="gamma">
      <power_law emax="200000.0" emin="15.0" gamma="2.0" />
    </particle>
    <celestial_dir dec="-2.42" ra="10.11" />
    </spectrum>
  </source>
</source_library>
```

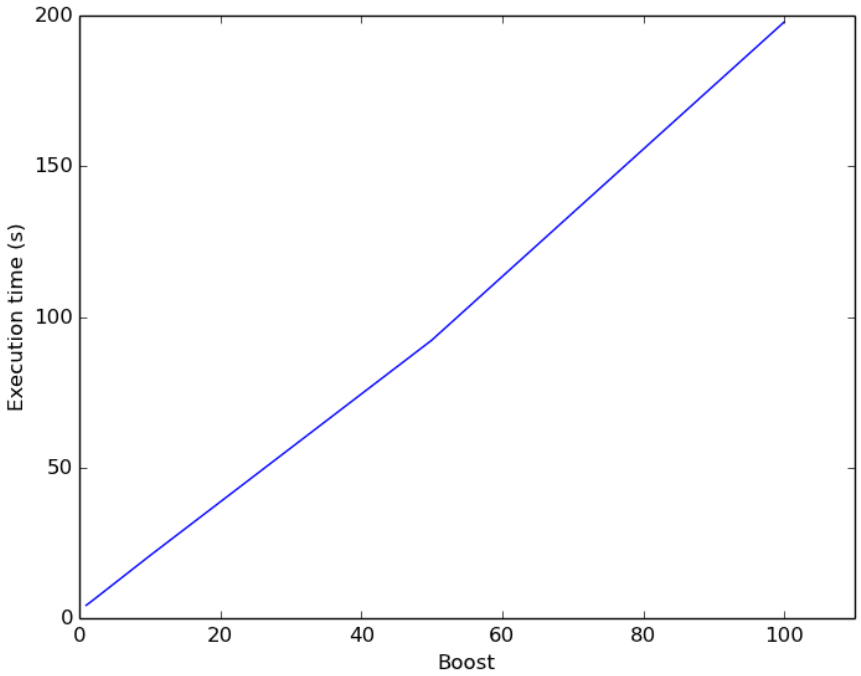
This is the command line I am using:

```
gtobssim infile=sim_model.xml srclist=source_list.txt scfile=/home/giacomov/FermiData/bn131231198
/gll_ft2_tr_bn131231198_v00.fit evroot=boost simtime=100000 ltfrac=1 tstart=410157919.083 nevents=no maxtime=3.
155e8 use_ac=no emin=30 emax=100000 edisp=no irfs=P8_SOURCE_V4 maxrows=100000 seed=653786
```

These are the execution times with the flux as reported above, and then "boost" times that value:

Boost	Execution time (s)
1	4.438
2	6.176
10	20.802
50	92.315
100	197.873

This shows that the algorithm is essentially linearly dependent on the number of events (the inflection between 50 and 100 might be due to memory management, or just noise):

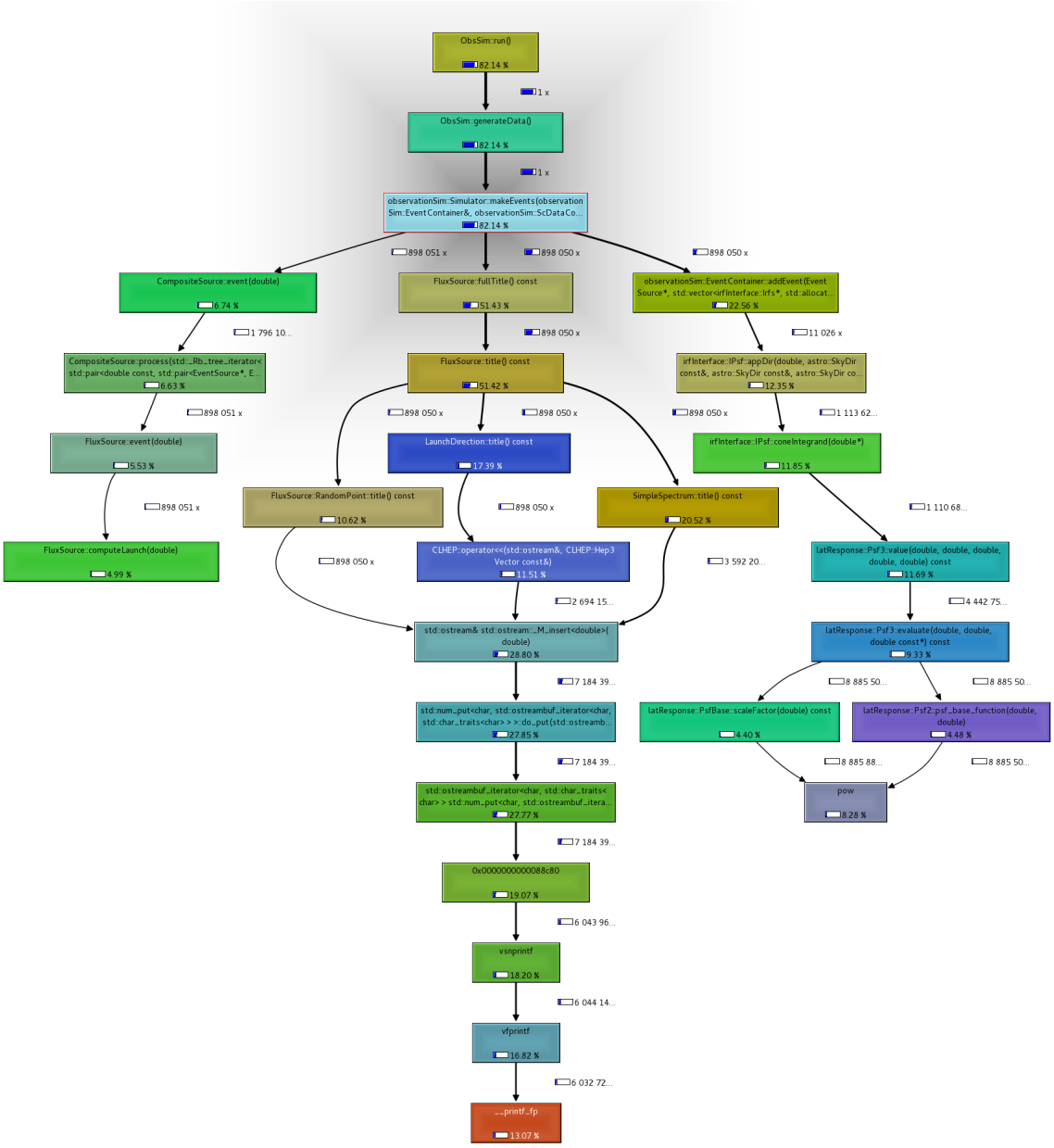


Profiling

I used this command to profile the execution:

```
valgrind --tool=callgrind gtobssim infile=sim_model.xml srclist=source_list.txt scfile=/home/giacomov/FermiData
/bn131231198/gll_ft2_tr_bn131231198_v00.fit evroot=boost simtime=100000 ltfrac=1 tstart=410157919.083
nevents=no maxtime=3.155e8 use_ac=no emin=30 emax=100000 edisp=no irfs=P8_SOURCE_V4 maxrows=100000 seed=653786
```

and then the tool kcachegrind to produce this plot:



I am puzzled by the fact that 50 % of the time is spent in the FluxSource::fullTitle() function, which afaik should only return the full name of the source. The relevant piece of code which calls fullTitle is in observationSim/src/Simulator.cxx:

```
void Simulator::makeEvents(EventContainer &events,
                           ScDataContainer &scData,
                           std::vector<irfInterface::Irfs *> &respPtrs,
                           Spacecraft *spacecraft,
                           bool useSimTime) {
   m_useSimTime = useSimTime;
   m_elapsedTime = 0.;
// Loop over event generation steps until done.
   while (!done()) {

         ...

         std::string name = m_newEvent->fullTitle();
         if (name.find("TimeTick") != std::string::npos) {
            if (!m_usePointingHistory) {
               scData.addScData(m_newEvent, spacecraft);
            }
         } else {
            if (events.addEvent(m_newEvent, respPtrs, spacecraft)) {
               m_numEvents++;
            }
         }

    ...

   } // while (!done())
}
```

This is essentially only checking if the name contains "TimeTick", because in that case it has to generate a new piece of simulated FT2 file. In my case I am using a real FT2 file, so the check is false. Now, as a quick-and-dirty fix, just by substituting that line with:

```
  while (!done()) {

         ...

         //std::string name = m_newEvent->fullTitle();

         std::string name = "just a source";

         if (name.find("TimeTick") != std::string::npos) {
            if (!m_usePointingHistory) {
               scData.addScData(m_newEvent, spacecraft);
            }
         } else {
            if (events.addEvent(m_newEvent, respPtrs, spacecraft)) {
               m_numEvents++;
            }
         }

    ...

   } // while (!done())
```

I reduce the execution time to ~1/3 of what it used to be:

Boost	Execution time (s)	Old execution time (s)
10	7.636	20.802
50	28.375	92.315
100	53.868	197.873

An easy fix would be to cache the name of the source in the FluxSource class, so that it is only computed once.