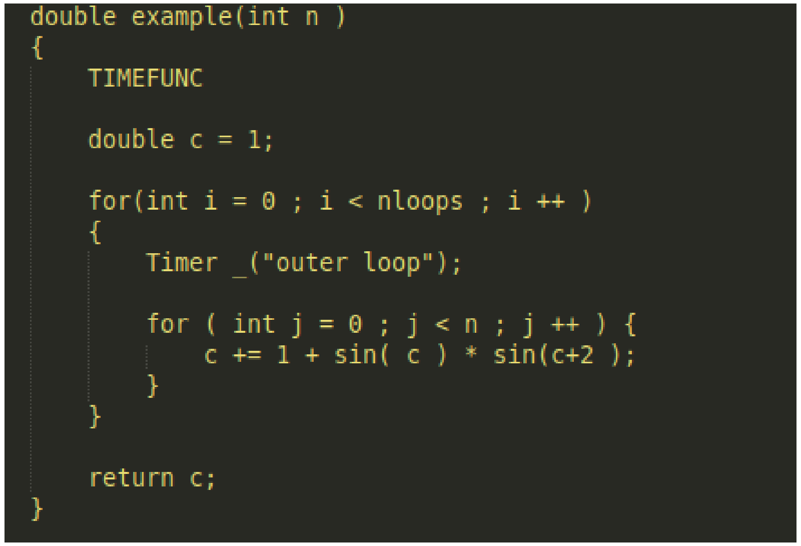
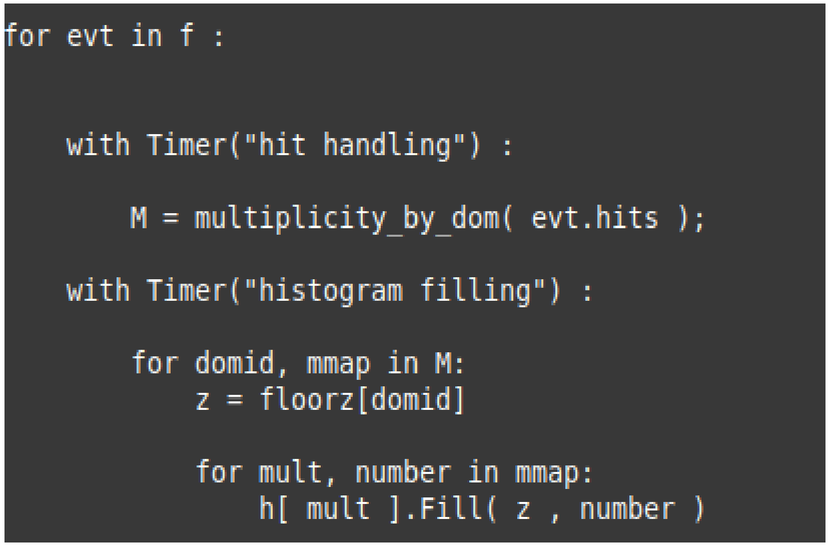
Aanet Timer utility

C++

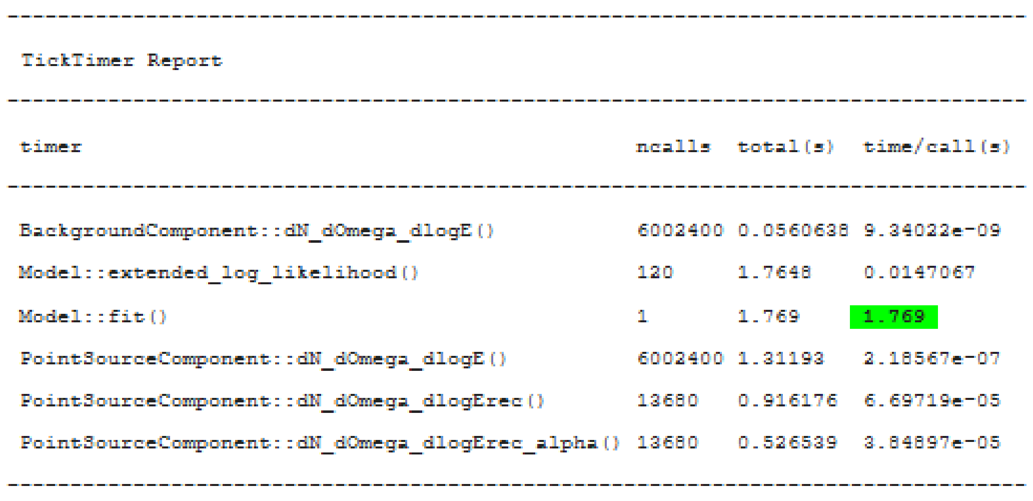


```cpp
double example(int n )
{
    TIMEFUNC

    double c = 1;

    for(int i = 0 ; i < nloops ; i ++ )
    {
        Timer _("outer loop");

        for ( int j = 0 ; j < n ; j ++ ) {
            c += 1 + sin( c ) * sin(c+2 );
        }
    }

    return c;
}
```

python



```python
for evt in f :


   with Timer("hit handling") :

       M = multiplicity_by_dom( evt.hits );

   with Timer("histogram filling") :

       for domid, mmap in M:
           z = floorz[domid]

           for mult, number in mmap:
               h[ mult ].Fill( z , number )
```

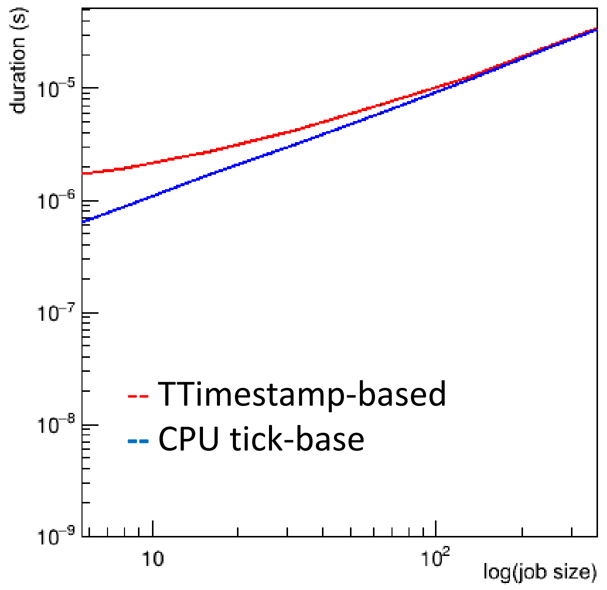


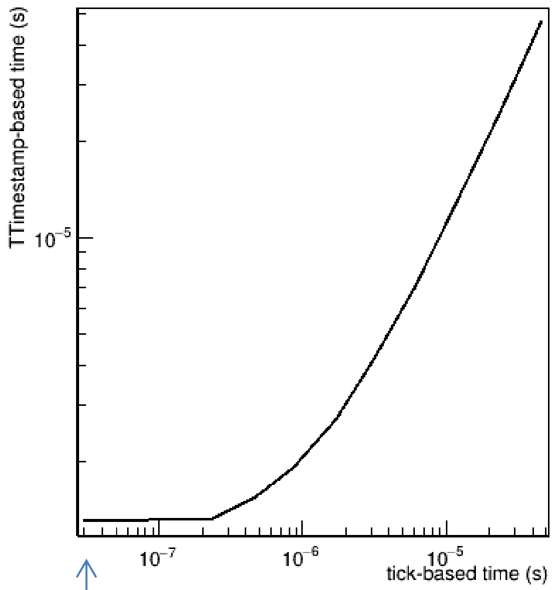
```
---------------------------------------------------------------------------------

 TickTimer Report

---------------------------------------------------------------------------------

 timer                                                   ncalls   total(s)   time/call(s)

---------------------------------------------------------------------------------

 BackgroundComponent::dN_dOmega_dlogE()                  6002400  0.0560638  9.34022e-09

 Model::extended_log_likelihood()                        120      1.7648     0.0147067

 Model::fit()                                            1        1.769      1.769

 PointSourceComponent::dN_dOmega_dlogE()                 6002400  1.31193    2.18567e-07

 PointSourceComponent::dN_dOmega_dlogErec()              13680    0.916176   6.69719e-05

 PointSourceComponent::dN_dOmega_dlogErec_alpha()        13680    0.526539   3.84897e-05

---------------------------------------------------------------------------------
```

Example from point-source Likelihood fit (functions timed with TIMEFUNC macro)

Timing very small jobs…

- Root's TTimeStamp and (std::chrono) themselves take a lot of time
  - Accessing the (fancy) system clock
  - Not suitable < 1 us
- Alternative: access directly a counter in the CPU : __rdtsc()





30 ns

conclusion

- Aanet has nice timer utility for timing and counting pieces of code
- Now suitable for really short intervals also
- (Works in python too!)