ctools - Feature #117

Define ctools profiling procedure

03/08/2012 05:50 PM - Knödlseder Jürgen

Status:	New	**Start date:**	03/08/2012
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

A major parameter of the ctools is the execution speed, as this drives the time it takes to perform scientific analysis. While code optimizations should in principle increase execution speed, additional features may decrease the speed. In any case, the execution speed of each tool should be monitored so that one gets alarmed if things degrade substantially between releases.

For this purpose, a procedure should be defined and implemented for each tool that allows the monitoring of the execution speeds.

History

#1 - 10/13/2012 07:20 PM - Deil Christoph

Jürgen Knödlseder wrote:

> For this purpose, a procedure should be defined and implemented for each tool that allows the monitoring of the execution speeds.

In addition memory usage and maybe even disk I/O should be measured, although I don't know a tool for disk I/O profiling.

I just ran ctbin on a 3.7 GB input event list created with ctobssim and saw it take up over 10 GB of RAM while running, whereas I would have expected only 2 GB !?

```
In [11]: import gammalib, sys

In [12]: sys.getsizeof(gammalib.GCTAEventAtom()) # object size in bytes
Out[12]: 64

In [10]: 33462103. * 64 / (1024)**3 # n_events * event_size / bytes_per_Gb
Out[10]: 1.994496762752533
```

Also I noticed there is a quite long phase where 100% CPU is used before starting to read data from disk, I assume to call the constructor GCTAEventAtom constructor 33M times. This could probably be avoided, plus ctbin only needs to read RA, DEC, Energy, not the 20 other numbers from the FITS file. I'm not saying we should start optimizing anything now, but I agree that profiling is very important. And I think it's fun, so I'd be very interested to help set this up.

For Python there is a nice small tool called vbench by Wes McKinney for CPU profiling from python and monitoring the evolution over time (i.e. with every commit). Here's an example for pandas , which was also written by Wes McKinney.
Very recently Vlad Niculae extended vbench to also do memory profiling and set it up on the scikit-learn Jenkins
I think vtools can't profile the ctools binaries, but probably well over 90% of the interesting benchmarks for gammalib and ctools could be run via the python interface?

vtools is a very young tool that doesn't have many users or good documentation, but I think it looks simple and promising enough.
Jürgen, do you think I should give vtools a shot or do you have something else in mind?