ctools - Feature #1970

Use OpenMP to parallelize computation in ctbin

03/18/2017 01:00 PM - Knödlseder Jürgen

Status:	Closed		**Start date:**	07/28/2017
Priority:	Normal		**Due date:**	
Assigned To:	Cardenzana Josh		**% Done:**	100%
Category:			**Estimated time:**	0.00 hour
Target version:	1.4.0			

Description

History

#1 - 07/04/2017 10:14 AM - Cardenzana Josh

- Status changed from New to Pull request

- Assigned To set to Cardenzana Josh

- % Done changed from 0 to 90

Since this is immediately relevant to the analysis pipeline I'm working on, I've gone ahead and implemented this. Here are the results from testing on the EGS data in the 1dc (489 observations, each 30-minutes long) using 4 cores for the parallel computation:

devel branch:

```
...
2017-07-03T10:22:32: +===================+
2017-07-03T10:22:32: | Binned observation |
2017-07-03T10:22:32: +===================+
2017-07-03T10:22:32: === GObservations ===
2017-07-03T10:22:32:  Number of observations ....: 1
2017-07-03T10:22:32:  Number of models ..........: 0
2017-07-03T10:22:32:  Number of observed events .: 21748359
2017-07-03T10:22:32:  Number of predicted events : 0
...
2017-07-03T10:22:48: Application "ctbin" terminated after 1312 wall clock seconds, consuming 1298.28 seconds of CPU time.
```

1970-parallelize_ctbin branch:

```
...
2017-07-03T12:54:32: +===================+
2017-07-03T12:54:32: | Binned observation |
2017-07-03T12:54:32: +===================+
2017-07-03T12:54:33: === GObservations ===
2017-07-03T12:54:33:  Number of observations ....: 1
2017-07-03T12:54:33:  Number of models ..........: 0
2017-07-03T12:54:33:  Number of observed events .: 21748359
2017-07-03T12:54:33:  Number of predicted events : 0
...
2017-07-03T12:54:48: Application "ctbin" terminated after 386 wall clock seconds, consuming 1371.89 seconds of CPU time.
```

Since the point of the above was to speed up the computation of ctbin, I went ahead and profiled the method and found the most time was spent in the 'set_weights()' method doing coordinate conversions. I worked around this by caching the sky direction of each pixel position before looping over the observations.

The computation of the angular distance from the bin positions and the observation central position was also taking a substantial amount of computation time. This is due to the fact that the coordinates in the map are in galactic coordinates and the computation has to first convert these coordinates into RA,Dec before computing the distance. I don't see any reason why these coordinates could not be cached after they are computed the first time, but at the moment they are not. Filling the pixel GSkyDir objects with their RA,Dec coordinates instead of their galactic coordinates means this conversion does not need to be done every time the distance is computed and the code runs much faster. The updated computation time is now:

...
2017-07-03T19:06:10: +===================+
2017-07-03T19:06:10: | Binned observation |
2017-07-03T19:06:10: +===================+
2017-07-03T19:06:11: === GObservations ===
2017-07-03T19:06:11: Number of observations ....: 1
2017-07-03T19:06:11: Number of models ....: 0
2017-07-03T19:06:11: Number of observed events .: 21748359
2017-07-03T19:06:11: Number of predicted events : 0
...
2017-07-03T19:06:25: Application "ctbin" terminated after 99 wall clock seconds, consuming 251.791 seconds of CPU time.

Caching the sky directions increases the amount of memory used. For the map I was running on (2000 x 1800 = 3.6 million pixels) this increased the memory from about 1.3 GB to 1.6 GB (<1 kB per bin increase). With all the above changes, computation time is reduced by more than a factor of 10 using 4 cores.

Branch for pull:
joshcardenzana/ctools : 1970-parallelize_ctbin

#2 - 07/28/2017 06:58 PM - Knödlseder Jürgen

- Status changed from Pull request to Closed

- Target version set to 1.4.0

- Start date set to 07/28/2017

- % Done changed from 90 to 100

Sorry for being late with the merging. The change is now in the devel branch.