ctools - Action #2309

Speed up IRF and RING background methods in ctskymap

02/10/2018 12:52 AM - Knödlseder Jürgen

Status:	Closed	**Start date:**	02/10/2018
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.6.0		

Description

The IRF and specifically the RING background methods make ctskymap pretty slow.

To speed-up the IRF background method that GCTABackground::rate_ebin() method should be used for background integration (see #2308).

To speed-up the RING background method the ctskymap::map_significance() method needs to be accelerated. This method calls the ctskymap::compute_ring_values() method that checks for every pixel whether it is contained in the source or background region. Loop over a smaller zone of relevant values will speed up the computation.

History

#1 - 02/10/2018 12:53 AM - Knödlseder Jürgen

- Target version set to 1.6.0

#2 - 02/10/2018 05:10 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- % Done changed from 0 to 50

For the IRF background the GCTABackground::rate_ebin() method is now used, which considerably speeds up things.

For the RING background a GSkyDir::cos_dist() method was implemented in GammaLib that returns the cosine of the distance, avoiding an acos() operation. The logic was also changed to minimize the angular distance computations. For a test run that took before 141.43 CPU seconds an execution time of 27.46 CPU seconds is now reached, a speed up of more than a factor of 5.

#3 - 02/10/2018 09:55 PM - Knödlseder Jürgen

I analysed the application of ctskymap to the GPS data since Akira has reported that the following task lasted for about 8 hours on a 3.1 GHz MacBook Pro:

```
$ ctskymap
Input event list or observation definition XML file [$CTADATA/obs/obs_gps_baseline.xml]
First coordinate of image center in degrees (RA or galactic l) (0-360) [0]
Second coordinate of image center in degrees (DEC or galactic b) (-90-90) [0]
Projection method (AIT|AZP|CAR|GLS|MER|MOL|SFL|SIN|STG|TAN) [CAR]
Coordinate system (CEL - celestial, GAL - galactic) (CEL|GAL) [GAL]
Image scale (in degrees/pixel) [0.1] 0.02
Size of the X axis in pixels [3600] 18000
Size of the Y axis in pixels [200] 500
Lower energy limit (TeV) [0.1]
Upper energy limit (TeV) [300]
Background subtraction method (NONE|IRF|RING) [NONE] IRF
Output skymap file [gps_skymap.fits]
```

For the current software version including the IRF background improvements the job lasted for 3.25 hours on a 2.8 GHz MacBook Pro. Note that without background subtraction the job takes 4 min, hence the time is used for the background estimation.

#4 - 02/10/2018 10:37 PM - Knödlseder Jürgen

- % Done changed from 50 to 60

After using also the GSkyDir::cos_dist() method in the IRF background computation a sky map of the GPS with IRF background is computed in 2004 seconds (33.4 min).

#5 - 02/11/2018 12:14 AM - Knödlseder Jürgen

- % Done changed from 60 to 70

I also added OpenMP support to the code. The following loops are parallelized:

- loop over observations when filling the sky map and computing the background map from the IRF template
- loop over the pixels when evaluating the RING background

The code needs to be tested on a machine with OpenMP support (the Mac OS X compiler still does not have one).

#6 - 02/11/2018 08:16 AM - Knödlseder Jürgen

On kepler the code with OpenMP support and no background subtraction takes 2402.29 seconds (40 min). Recall that on Mac OS X it was 4 min. Note that the event binning is so quick that barely two threads were in parallel.

The IRF background subtraction on kepler took 1948 wall clock seconds (32 min) and about 46 threads were used in parallel (68891.9 seconds of CPU time, hence an effective of 35 threads). Recall that on Mac OS X it took 33 min, hence the wall clock time between both systems are similar, despite the fact that on Mac OS X the analysis runs serial and on kepler in parallel.

The RING background is too slow for meaningful results on the GPS. The only way to improve now the RING background is to create a bounding box around the pixel of interest and to check only those pixels that are relevant. The bounding box can be determined using the binsz parameter under the assumption that the map distortions are not too large. Alternatively the local map scale could be determined from the pixel coordinate difference at the centre.

#7 - 02/12/2018 09:10 AM - Knödlseder Jürgen

- % Done changed from 70 to 90

The background RING computation is now restricted to a bounding box around the outer circle of the background ring. This considerably speeds up the computations for maps that are larger than the typical background ring size.

On Mac OS X, the generation of a RING background map for the entire GSP with 18000 x 500 pixels takes 5275.88 seconds (1h 28min). On kepler, using OpenMP, the computation takes 5116 seconds.

Below a summary table of sky map benchmarks for the GPS with 18000 x 500 pixels:

System	Raw	IRF	RING
Mac OS X	4 min	33 min	1h28min
Kepler (wall clock)	16 min	32 min	1h25min
Kepler (CPU time)	24 min	19h08min	37h04min

#8 - 02/13/2018 04:06 PM - Knödlseder Jürgen

- % Done changed from 90 to 100

I added a FFT-based method to compute the RING background that basically takes no time. The FFT-based method uses a cartesian approximation for the sky map pixel grid, which is not exactly the same as computing the correct angular distance between pixels to determine whether they are inside or outside a circle or a ring. The user can switch between the faster FFT-based method and the direct computation using the usefft parameter.

Using the FFT-based method the tool now takes about the same time for the RING and the IRF background methods.

#9 - 02/14/2018 10:04 AM - Knödlseder Jürgen

- Status changed from In Progress to Closed

Merged into devel.