GammaLib - Feature #3312

Implement a source distance cache in GCTAResponseCube

08/06/2020 01:00 PM - Knödlseder Jürgen

Status:	Closed	**Start date:**	08/06/2020
Priority:	High	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2.0.0		

Description

An important and time consuming part of the cube response computation is the determination of the distance between the model centre and the observed event direction. The relevant code is shown below:

```
double GCTAResponseCube::irf_ptsrc(const GEvent&       event,
                                   const GSource&      source,
                                   const GObservation& obs) const
{
   ...
   double delta = bin->dir().dir().dist(srcDir);
   ...
}
double GCTAResponseCube::irf_radial(const GEvent&       event,
                                    const GSource&      source,
                                    const GObservation& obs) const
{
   ...
   double rho_obs = model->dir().dist(obsDir);
   ...
}
double GCTAResponseCube::irf_elliptical(const GEvent&       event,
                                        const GSource&      source,
                                        const GObservation& obs) const
{
   ...
   double rho_obs      = model->dir().dist(obsDir);
   double posangle_obs = model->dir().posang(obsDir); // Celestial
   ...
}
```

Since for a cube the observed event direction is known a priori (the spatial cube pixels), the distances can be cached for each model. The cache would be a sky map for each model, with the sky map definition defined by the event cube. The sky map can be preset with negative values (for example -1), signaling that the distance was not yet computed.

Related issues:

Related to GammaLib - Action # 3404: Implement a sparse vector cache in GResp...	**Closed**	**10/17/2020**

History

#1 - 08/06/2020 02:48 PM - Knödlseder Jürgen

The caching can be implement even at higher level, in GResponse::irf_spatial(), where a

virtual void GResponse::use_source(const GEvent& event, const GSource& source, const GObservation& obs);

can be implemented that at the base class level aways returns true.

Any instrument specific response implementation can reimplement the method, adding whatever checking or caching is needed, with the aim that the spatial IRF is only computed in case that it is actually relevant.

The GResponse::irf_spatial() method should then look like

```
double GResponse::irf_spatial(const GEvent&       event,
                              const GSource&      source,
                              const GObservation& obs) const
{
    // Initialise IRF value
    double irf = 0.0;

    // Continue only if source should be used
    if (use_source(event, source, obs)) {

        (response computation here)

    } // endif: source was to be used

    // Return IRF
    return irf;
}
```

#2 - 08/07/2020 06:13 PM - Knödlseder Jürgen

- Assigned To set to Knödlseder Jürgen

- Priority changed from Normal to Urgent

- Target version set to 1.7.1

Note also that the current caching implementation in GResponse::irf_spatial() does a lot of unnecessary caching for IRF values that are actually zero because the source is too far away from the event. **There must be a use_source check before the method does caching to avoid unnecessary caching of values.**

A caching mechanism should already be implemented in release 1.7.1 that does not add public interfaces, but that provides the required functionality for binned analysis.

#3 - 08/07/2020 11:19 PM - Knödlseder Jürgen

- Priority changed from Urgent to High

- Target version changed from 1.7.1 to 2.0.0

It is not possible to implement such a mechanims without touching the interface, hence the issue should be implemented for release 1.8.0. In the meantime, the cache should be switched off for stacked analysis (#3315).

#5 - 08/08/2020 03:17 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- % Done changed from 0 to 10

Note that there is already a branch at https://cta-gitlab.irap.omp.eu/gammalib/gammalib/tree/3312-source-distance-cache that starts implementing the issue. In this branch there is already the required interface change that is necessary. The branch needs to be rebased on devel before continuing the development.

#6 - 10/21/2020 09:26 AM - Knödlseder Jürgen

- Related to Action #3404: Implement a sparse vector cache in GResponse added

#7 - 10/21/2020 09:26 AM - Knödlseder Jürgen

- Status changed from In Progress to Closed

- % Done changed from 10 to 100

This issue is now covered by the vector response caching, see #3404.