ctools - Feature #2147

Reduce computation time of ctmodel and parallelize using OpenMP.

07/04/2017 10:53 AM - Cardenzana Josh

Status:	Closed	**Start date:**	07/04/2017
Priority:	Normal	**Due date:**	
Assigned To:	Cardenzana Josh	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.3.1		

Description

ctmodel takes a long time to compute the model cube. The method can be profiled to reduce the ctmodel's overhead, followed by parallelizing the method with openmp, as some of the other ctools are.

Related issues:

Related to GammaLib - Feature # 2148: GSkyDir does not cache ra,dec or l,b va...	**Closed**	**07/04/2017**

History

#1 - 07/04/2017 03:30 PM - Cardenzana Josh

- Related to Feature #2148: GSkyDir does not cache ra,dec or l,b values when they are computed added

#2 - 07/05/2017 04:30 PM - Cardenzana Josh

- Status changed from In Progress to Pull request

Continuing the profiling of ctmodel from issue #2148, here's the comparison for further tests:

```
...
2017-07-04T12:06:59: === GCTAEventCube ===
2017-07-04T12:06:59:  Number of events ..........: 21746400
2017-07-04T12:06:59:  Number of elements ........: 67200000
2017-07-04T12:06:59:  Number of pixels ..........: 3360000
2017-07-04T12:06:59:  Number of energy bins .....: 20
2017-07-04T12:06:59:  Time interval .............: 59215.5 - 59256.8548611111 days
2017-07-04T12:06:59: === GEbounds ===
2017-07-04T12:06:59:  Number of intervals .......: 20
2017-07-04T12:06:59:  Energy range ..............: 100 GeV - 100 TeV
2017-07-04T12:06:59: === GSkyMap ===
2017-07-04T12:06:59:  Number of pixels ..........: 3360000
2017-07-04T12:06:59:  X axis dimension ..........: 2800
2017-07-04T12:06:59:  Y axis dimension ..........: 1200
2017-07-04T12:06:59:  Number of maps ............: 20
2017-07-04T12:06:59:  Shape of maps .............: (20)
...
2017-07-04T12:07:07: Application "ctmodel" terminated after 6332 wall clock seconds, consuming 6284.84 seconds of CPU time.
```

In the original run the most time intensive part of the computation is evaluating whether a bin is contained within the region of interest of an observation. That is evaluating:

```
roi.contains(*bin)
```

This is evaluated for each spatial bin in each energy range. The for-loop can be restructured so that the bin is checked for containment for the first energy bin, and then the energy bins are looped on without having to check for containment again. This results in more than 7x speed up in the computation for the above analysis:

```
2017-07-04T13:27:05: Application "ctmodel" terminated after 827 wall clock seconds, consuming 800.555 seconds of CPU time.
```

Each iteration on the loop also includes a printout of the current cube information. This takes about 25-30% of the remaining computation time, but the information is only printed if "chatter == EXPLICIT". I wrapped those lines in a check on the chatter value, since there's no need to do those computations unless chatter is less than EXPLICIT. This is probably only a major contributor at this point because of the other modifications and the

large size of the model cube.

2017-07-05T07:49:24: Application "ctmodel" terminated after 550 wall clock seconds, consuming 547.987 seconds of CPU time.

Next, parallelizing the method required caching the bin value (so that threads don't clobber the map bins when filling) and to create an independent model object for each core (so that the internal GModels caching doesn't conflict between threads). For the bin issue, I added a new method to GCTAEventCube to point each binHere's the updated

2017-07-05T13:36:53: Application "ctmodel" terminated after 166 wall clock seconds, consuming 565.627 seconds of CPU time.

I also had to change the if-statements in the ctmodel::run() for-loop, since openmp wrapped by python doesnt appear to play well with 'continue' statements in the for-loop directly under a '#pragma omp parallel for' statement.

So, with these changes and the change from issue #2148 (which I merged into the 2147 branch for gammalib) the computation time is down now from about 4.5 hours to under 9 minutes before parallelizing, and just under 3 minutes with parallelizing through openmp with 4 threads. The remaining intensive computations are in the 'roi.contains(bin)' computation and evaluating the models.

Pull branches (gitlab):
josh cardenzana / gammalib: 2147-parallelize_ctmodel
josh cardenzana / ctools: 2147-parallelize_ctmodel

#3 - 07/06/2017 04:58 PM - Knödlseder Jürgen

I looked over the new code and I think I found a way to avoid the change of the GammaLib interface. I would like to avoid this since the method you added is pretty low level, and seems basically only needed for the parallelization of the code.

I pushed back the changes in your josh cardenzana / ctools: 2147-parallelize_ctmodel branch. I though that I would push to ctools but it pushed into your directory. Sorry for that. I was not aware that I can do that (maybe it's a thing that can be configured on GitLab).

Anyway: could you please check the new code in the josh cardenzana / ctools: 2147-parallelize_ctmodel branch and tell me the computation time it needs? I had to add one more OMP critical zone.

#4 - 07/07/2017 01:19 AM - Knödlseder Jürgen

- Target version changed from 1.4.0 to 1.3.1

- % Done changed from 0 to 90

I changed the code again a bit to circumvent the GCTAEventCube::set_bin() method. I now explicitly set the attributes of a bin and do no longer extract a bin from a counts cube. This prevents handling of pointers or reallocation of memory that was needed with the previous logic.

I did some computational speed performance checks and it looks as if I would get the same speed as with your previous code. But it would be good if you could verify that. The code is in the bugfix-1.3.1 branch.

#5 - 07/07/2017 02:38 AM - Cardenzana Josh

The computation time appears to be similar (maybe even a little faster), however the predicted counts values being stored in the map are just shy of 500x larger. I tracked this down to the value of 'ontime' being given to the bin. What is actually being stored is the combined on time of the entire observation list, rather than the on time of the run being evaluated. I made one small change to set the bin's on time from the observation being filled into the cube, which also allows removing the 'm_ontime' variable. The bin values are now the same as they were in the original devel branch. I've pushed these changes to the 2147-parallelize_ctmodel branch.

#6 - 07/07/2017 11:24 PM - Knödlseder Jürgen

- Status changed from Pull request to Closed

- % Done changed from 90 to 100

Thanks for checking and finding the bug.

This illustrates that we need to add a check of reference values to the ctools tests. Currently the values in the files that are produced by the ctools are not tested, which is definitely a lack. I added issue #2151 to request the addition of such tests.

I merged your changes into the bugfix-1.3.1 and devel branches.