# GammaLib - Change request #2770

## GWcs::dir2pix critical statement possibly too restrictive

12/12/2018 12:28 PM - Cardenzana Josh

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 12/12/2018 |
| **Priority:** | Low | **Due date:** | |
| **Assigned To:** | Cardenzana Josh | **% Done:** | 100% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | 1.6.0 | | |

**Description**

There's an 'omp critical(GWcs_dir2pix_1)' statement in GWcs::dir2pix that effectively blocks the computations in this method from running in parallel. The reason is that the critical statement covers not only the initial if-statement, but also the associated else-statement. As a result, the following 'omp critical(GWcs_dir2pix_2)' actually doesn't do anything as the statement still falls under the influence of the preceding 'omp critical(GWcs_dir2pix_1)'.

Is there any reason why the computations carried out in the else-statement shouldn't be run in parallel? If not, I propose changing the critical to only wrap the initial 'if' and the subsequent updating of the cached variables. A similar change would also apply to GWcs::pix2dir.

### History

#### #1 - 12/12/2018 02:23 PM - Knödlseder Jürgen

I think the intention was to put only the if statement in the critical zone, not the entire else block. So if there is a better way to protect the computations then let's do that.

#### #2 - 12/12/2018 03:35 PM - Cardenzana Josh

*- Assigned To set to Cardenzana Josh*

I'll make the changes and run some tests to see how significant they are.

#### #3 - 12/14/2018 09:57 AM - Cardenzana Josh

I've taken care of the issue with the locking. To test it I've run ctmodel using the IEM model from DC1 and it improves the computation speed by a little.

However, this is still quite inefficient. Because ctmodel makes a copy of the models for each thread, there's no danger of encountering a race condition, but this critical will still 'critical' the execution of this section of code for every thread. As a result, about 50% of my test is waiting at this critical statement. It will also block the execution of this code for different models if there are multiple models that make calls to GWcs::dir2pix.

The way around this seems to be OpenMP locks. I've tested it by generating a unique omp_lock_t m_thread_lock for each GWcs object and wrapping the sections that would be #pragma omp critical(<name>) with omp_set_lock( m_thread_lock ) and omp_unset_lock( m_thread_lock ) instead. Basically, this just says "I want this to only be critical if I'm accessing this same section of code for the same GWcs object on multiple threads, but if I'm accessing different GWcs objects then it wont block its execution". This does reduce the computation time by about 2x (running on 6 cores).

The downside is that this *requires* one of the following options:

1. OpenMP must be compiled in, otherwise 'omp_lock_t' and the relevant functions wont be recognized
2. Each call to set/unset the lock is wrapped in an #ifdef _OPENMP statement.
3. Create a new class (something like GThreadLock) that handles all of the appropriate definitions and calls depending on whether or not OpenMP is defined. It could also be setup to have named locks, just like named critical regions.

The first option would obviously cause problems on Macs since the default Mac clang doesn't have OpenMP support. The second option would make the code unreadable. The third option gets around both of these, but it will also adds some additional (non-pragma) code and another variable to GWcs. Thoughts?

#### #4 - 12/18/2018 03:10 PM - Cardenzana Josh

*- Status changed from New to Pull request*

*- % Done changed from 0 to 100*

Doing a quick survey of the critical statements in GammaLib, this is really the only situation that I can see where an OpenMP lock would be useful. Following the advice from Jürgen, I localized the fix to GWcs by adding the following methods:

- init_lock: Initializes a lock when the GWcs object is created by ID
- set_lock: Sets a given lock by ID
- unset_lock: Frees a given lock based on ID

The lock ID is an integer because apparently searching for the ID as a string is discernibly more computationally intensive than searching for an integer map key. It is also important to note that the methods call OpenMP functions, which are wrapped in #ifdef _OPENMP statements.

I've also incorporated a change to the == operator for GSkyDir which also has some impact on the computation time. In the previous implementation, if the left GSkyDir object in the comparison had Galactic coordinates it would still compute the l,b coordinates of the right hand GSkyDir object even if both had ra,dec coordinates as well. This meant that if you're looking at a map with galactic coordinates, the computation would almost always have to compute the l,b coordinates of a test position in the map for ctmodel.

The overall impact on the computation time is a >2x reduction in computation time running with 6 cores on my Mac. Running on a less efficient cluster with 24 cores the computation time was improved by >10x. This is finished as far as I can tell, unless there are comments or suggested changes to the implementation.

Pull information:
Repo: Josh Cardenzana / gammalib
branch: 2770-loosen_dir2pix_critical

#### #5 - 01/09/2019 03:31 PM - Knödlseder Jürgen

*- Status changed from Pull request to Closed*

*- Target version set to 1.6.0*

Looks good. I merged the code into devel.