GammaLib - Bug #3308

Speed-up source saving

08/04/2020 05:13 PM - Knödlseder Jürgen

Status:	Closed	**Start date:**	08/04/2020
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.7.1		

Description

The source saving after the source detection step takes a very long time, not clear what's going on.

History

#1 - 08/04/2020 06:06 PM - Knödlseder Jürgen

- Status changed from New to In Progress

- % Done changed from 0 to 20

The actual step that takes very long is the saving of the XML files to disk using the GModels::save() method.

For the fill GPS run the saving of the high TS model took 24 minutes, while for the background it took 2 seconds. The file size is 2 MB, still, not clear why the saving takes so long. Loading takes a few seconds only. Note, however, that the file contains 3032 sources. Maybe that's the reason.

#2 - 08/05/2020 03:16 PM - Knödlseder Jürgen

Analysing gsrvy with valgrind reveals that a lot of time is used for building-up the XML object. Note that the results below are an intermediate dump, hence the numbers do not exactly add up as expected.

For 1 050 calls to GModelSky::write(GXmlElement) there are 552 153 calls to GXmlNode::element(std::string,int) (about 526 calls per GModelSky::write(GXmlElement) call). From GXmlNode::element(std::string,int) there are:

- 558 423 calls to GXmlNode::elements(std::string) using 42.2% of the time
- 19 246 140 calls to __dynamic_cast using 32.4% of the time (corresponding to 19329 calls per model!)

For the 1 050 calls to GModelSky::write(GXmlElement) there are 1 049 calls to GModelSpectralPlaw::write(GXmlElement) (the difference of 1 can be explained that the execution did not reach the write statement when the dump was done). From there, there are:

- 3 147 calls to gammalib::xml_need_par(std::string, GXmlElement,std::string) (3 times 1049, i.e. the number of spectral parameters), which calls 3 128 times GXmlElement::GXmlElement(std::string)
- 3 147 calls to GModelPar::write(GXmlElement)

For the 1 050 calls to GModelSky::write(GXmlElement) there are also 1 049 calls to GModelSpatialRadialDisk::write(GXmlElement) which calls 1 048 times the base class method GModelSpatialRadial::write(GXmlElement). Note that the base class writes RA and DEC, while the derived class writes Radius. Both classes together make:

- 3 145 calls to gammalib::xml_need_par(std::string, GXmlElement,std::string) (about 3 times per GModelSky::write() call, one base class call is missing), which calls 3 164 times GXmlElement::GXmlElement(std::string)
- 3 145 calls to GModelPar::write(GXmlElement)

Inspecting the code it becomes clear that the main bottle neck should be the gammalib::xml_need_par(std::string, GXmlElement,std::string) method. This method loops over all parameters in an XML file, calling in the loop the GXmlNode::element(std::string,int) method. Since in total there are 6 294 calls to gammalib::xml_need_par(std::string, GXmlElement,std::string), there may be 18 882 calls to GXmlNode::element(std::string,int), still well below the observed number of 558 423 calls.

The remaining calls are consumed in GModelSky::write(GXmlElement&) (exactly 550 055):

```
// Search corresponding source
int n = xml.elements("source");
for (int k = 0; k < n; ++k) {
    GXmlElement* element = static_cast<GXmlElement*>(xml.element("source", k));
    if (element->attribute("name") == name()) {
```

```
            src = element;
            break;
        }
    }
```

For each source this loop checks all past sources, which is about N*N/2 for N sources, i.e. about 550 201 times. **The worst thing is that, in principle, this loop should never find a source!** So it's a sanity check, since for a general API, one can never be sure that the source does not yet exist in the GXmlElement.

Inspecting GXmlElement* GXmlNode::element() gives a hint for some optimisation:

```
GXmlElement* GXmlNode::element(const std::string& name, const int& index)
{
    // Determine number of child elements
    int n = elements(name);

    // Signal if no children exist
    if (n < 1) {
        throw GException::xml_name_not_found(G_ELEMENT3, name);
    }

    // If index is outside boundary then throw an error
    if (index < 0 || index >= n) {
        throw GException::out_of_range(G_ELEMENT3, index, 0, n-1);
    }

    // Get the requested child element
    GXmlElement* element  = NULL;
    int          elements = 0;
    for (int i = 0; i < m_nodes.size(); ++i) {
        GXmlElement* src = dynamic_cast<GXmlElement*>(m_nodes[i]);
        if (src != NULL) {
            if (src->name() == name) {
                if (elements == index) {
                    element = src;
                    break;
                }
                elements++;
            }
        }
    }

    // Return child element
    return element;
}
```

The code comparses the source names twice, once in the call to elements(name) and once in the loop. The exceptions can be placed after the loop, and n be determined in the loops, so that the double looping is avoided. I implement this change.

In the gammalib::xml_need_par() method, calls to GXmlNode::elements() are within the for-loop, which may eventually lead to an overhead since the GXmlNode::elements() method recomputes each time the number of elements. I therefore extracted the number of elements before entering the loop (same in gammalib::xml_has_par()

In several places in GXmlNode there are constructs like

```
for (int i = 0; i < m_nodes.size(); ++i) {
    GXmlElement* src = dynamic_cast<GXmlElement*>(m_nodes[i]);
    if (src != NULL) {
        (do something)
    }
}
```

which I all replaced by

```
for (int i = 0; i < m_nodes.size(); ++i) {
    if (m_nodes[i]->type() == NT_ELEMENT) {
        GXmlElement* src = static_cast<GXmlElement*>(m_nodes[i]);
        (do something)
    }
```

}

avoiding the overhead related to dynamic type casting.

#3 - 08/05/2020 03:17 PM - Knödlseder Jürgen

- Project changed from gsrvy to GammaLib

- Target version changed from 0.2.0 to 1.7.1

- % Done changed from 20 to 50

Moved to GammaLib since all changes are on the level of GammaLib.

#4 - 08/05/2020 04:31 PM - Knödlseder Jürgen

- Status changed from In Progress to Feedback

- % Done changed from 50 to 90

I did some last change, which is rewriting the GModels::save() method so that the looping over existing sources is avoided. **This did the job!** Now the saving of the models is down to 1 second (from 25 minutes before).

#5 - 08/07/2020 11:20 AM - Knödlseder Jürgen

- Status changed from Feedback to Closed

- % Done changed from 90 to 100