Exiv2 - Patch #967

Back metadata by map instead of vector

05 Jul 2014 19:14 - Michael Pratt

Status:	New	**Start date:**	05 Jul 2014
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	api	**Estimated time:**	0.00 hour
Target version:	1.0		

Description

I have been using Exiv2 for a while, through the gexiv2 Python bindings. The images that I am most commonly processing contain about 300 Xmp tags, which I need to read most of. As the number of images I was processing began to scale up (1000+ images at a time), I began to notice significant performance issues. In profiling, I found that my program was spending upwards of 50ms per image just to read all of the XMP tags. Drilling down further, I found that Exiv2 stores all metadata in a std::vector of Exif, Xmp, or Iptc datums. The lookup time for a single tag is O(n) from the vector, since the entire vector needs to be scanned to find the tag. Since I was looking up all n tags, the total lookup runtime ended up being O(n^2), thus the poor performance for a large number of tags. Ideally, the metadata would be stored in a hash map, where it would have O(1) lookup times, or O(n) time for all tags.

Exiv2 does provide an iterator for the metadata, so this issue can be worked around by simply iterating through all tags once and either extracting the desired information, or by putting the metadata is secondary hash map data structure, which is used for all future lookups. Unfortunately, gexiv2 only provides the ability to get a list of all keys and then look up each value one at a time, not a way to directly iterate over all values. I submitted a patch to gexiv2[1] that adds the ability to get all key, value pairs in a single pass. This allowed me to build my own secondary hash map of metadata, which my application uses for all lookups.

Unfortunately, all of this is just a big workaround for the poor performance of lookups in Exiv2. It would be much preferred to simply improve the performance of Exiv2, both to avoid ugly workarounds, and to benefit all users of Exiv2.

Thus, I took it upon myself to convert the Exif, Iptc, and Xmp data storage from std::vector to std::multimap. The attached patches contain the conversion, and provide a massive performance increase for lookups with a large number of tags.

I spent quite a while trying to avoid changing the Exiv2 API, which is fairly tightly coupled to the use of std::vector. For example, most of the tag lookup methods return an iterator to values, which is simply the std::vector::iterator. On the other hand, std::multimap::iterator is a iterator to key, value pairs, and thus incompatible. The added "map_value_iterator" wraps the map iterator, providing the value on dereference, thus behaving like the std::vector::iterator.

The biggest publicly visible changes are to the sortByKey()/sortByTag() methods. std::multimap is kept sorted internally, so it does not make sense to provide sorting methods. Xmp tags are sorted by key, Exif by IfdId then tag number, and Iptc by record then tag number. XmpData::sortByKey(), ExifData::sortByTag(), and Iptc::sortByTag() are thus completely unnecessary, as the map is already sorted. Unfortunately, there is no practical way to resort the map, as would be necessary for ExifData::sortByKey() and IptcData::sortByKey(). These patches mark all of the sort*() methods as deprecated, and make them no-ops. A search on Debian Code Search[2] turns up only 3 uses of these methods. I sorted Exif and Iptc by their tag numbers, as it seems to be a more logical true sorting. If it were switched to sort by string key, then the sortByTag() functionality would be lost, which only has 1 use.[3]

The performance improvements moving from O(n) to O(1) lookups are quite impressive. The attached bench.cpp was used to benchmark total lookup times.

Looking at some of the typical images, we can see significant improvement, especially for XMP:

Before:

```
./bench /tmp/20140515-Yosemite-2233.jpg
Exif:    Tags: 52    Lookup time: 0.128897ms
Xmp:    Tags: 143    Lookup time: 7.52281ms
Iptc:    Tags: 12    Lookup time: 0.008165ms
```

After:

```
./bench /tmp/20140515-Yosemite-2233.jpg
Exif:    Tags: 52    Lookup time: 0.051482ms
Xmp:    Tags: 143    Lookup time: 0.414214ms
```

```
Iptc:    Tags: 12    Lookup time: 0.010078ms
```

This runtime reduction is really powerful in more extreme cases:

Before:

```
./bench /tmp/20140515-Yosemite-2233_quadxmp.jpg
Exif:    Tags: 52    Lookup time: 0.111087ms
Xmp:    Tags: 415    Lookup time: 35.7657ms
Iptc:    Tags: 12    Lookup time: 0.00966ms
```

After:

```
./bench /tmp/20140515-Yosemite-2233_quadxmp.jpg
Exif:    Tags: 52    Lookup time: 0.036755ms
Xmp:    Tags: 415    Lookup time: 0.844112ms
Iptc:    Tags: 12    Lookup time: 0.008161ms
```

Notice that the before lookup time has quadrupled, but the after lookup time only doubled. For a sufficiently small number of tags, the change is negligible.

I think this change would significantly improve Exiv2 across the board, and benefit all users.

[1] Related gexiv2 bug: https://bugzilla.gnome.org/show_bug.cgi?id=728072

[2]http://codesearch.debian.net/search?q=sortByKey%5C%28

[3]http://codesearch.debian.net/search?q=sortByTag%5C%28

History

#1 - 05 Jul 2014 23:06 - Robin Mills

- Category set to api

- Status changed from New to Assigned

- Assignee set to Andreas Huggel

- Target version set to 0.25

Thanks very much, Michael. This is a really interesting issue report. And thank you for the thorough analysis, patches and test files.

I have a little hesitation about changing this immediately. My hesitation is because of the API implications that you discuss. This may cause new .dll/.so/.dylibs to be binary incompatible. We are having an Exiv2 Team Meeting today at 09:00GMT and API Management is one of the topics for discussion. http://clanmills.com/files/Exiv2-0.25.pdf

I'm going to assign this to Andreas (the project leader) to request his comments.

Robin

#2 - 30 Aug 2014 10:28 - Michael Pratt

Are there any updates or new thoughts on this change? I spent quite a while ensuring that the Exiv2 API does not change with this patch, though the ABI does change, as you mention. So, users would need to recompile their application for this version of the library. I'm not very familiar with library releases and versioning, so I'm not sure how much of an issue that is.

#3 - 30 Aug 2014 10:46 - Robin Mills

Michael

Thanks for giving us a little push about this. I assure you that this will be discussed before the v0.25 release scheduled for November. Andreas has responsibility in Team Exiv2 for the API, so we'll have to wait for him to update this issue. The world of open source is different from work because there is no escalation process. So we have to wait for Andreas to juggle his work, family, biking and other things that contend for his valuable time.

Robin

#4 - 30 Aug 2014 10:48 - Michael Pratt

No problem, I understand. I just wanted to make sure this wasn't forgotten completely. :)

#5 - 19 Sep 2014 03:18 - Andreas Huggel

- Target version deleted (0.25)

Thanks Michael, that's a valuable set of patches! Not only because of the performance improvements, but also because it is nicely aligned with the old work done for the "unified metadata container" feature - keep all metadata in one container (unstable branch in svn). Eventually, that's still something I very much would like to finish.

Here are some comments of the issues encountered with that, including a bit of an explanation about the sorting:
http://dev.exiv2.org/projects/exiv2/repository/entry/branches/unstable/src/metadatum.cpp#L201

- Exiv2 doesn't sort all of the tags, e.g., it makes it a point to retain whatever order the original image's Makernote tags are in
- Using an index and copying the tags into a list that is sorted by this index may solve this. I think some of this is already implemented in the unstable branch

Your patches apply nicely and compile fine on my (somewhat outdated) revision of the trunk. However, many of the regression tests fail with the patched version (cd test; make; or run the shell scripts there individually). The next step will be to go through these one-by-one and fix them, before we can check this into the trunk. It may be worth creating a branch for that in SVN. I'm definitely interested in this, but will only have more time to look into it in about a month.

I don't see this going into 0.25, so I'm removing the target version. Once it's ready, it could be something we can release shortly after 0.25.

#6 - 21 Sep 2014 03:32 - Andreas Huggel

Just to clarify this ordering thing:

1. The concern is about the order of the metadata tags **when they are serialized**. That should conform with the respective standard, which is probably not a big problem for IPTC and XMP, but is more hairy for Exif. That's mainly because of the Makernote tags, for which there is no specification. Thus Exiv2 just maintains whatever order the Makernote tags are in when it writes them back to the image. That behaviour should remain.
2. The order of the tags in Exiv2's metadata containers is less important, that should just make some kind of sense.
3. I agree the sortByXyz functions can go. Looking at how they are used out in the wild seems to support that.

#7 - 23 Aug 2015 12:55 - Robin Mills

- Target version set to 1.0

#8 - 26 Apr 2017 22:25 - Robin Mills

- Status changed from Assigned to New

- Assignee deleted (Andreas Huggel)

Files

20140515-Yosemite-2233.jpg	3.35 MB	05 Jul 2014	Michael Pratt
20140515-Yosemite-2233_quadxmp.jpg	3.35 MB	05 Jul 2014	Michael Pratt
0001-types-Add-map_value_iterator.patch	2.55 KB	05 Jul 2014	Michael Pratt
0002-xmp-convert-metadata-store-from-std-vector-to-std-mu.patch	6.66 KB	05 Jul 2014	Michael Pratt
0003-exif-convert-metadata-store-from-std-vector-to-std-m.patch	7.31 KB	05 Jul 2014	Michael Pratt
0004-iptc-convert-metadata-store-from-std-vector-to-std-m.patch	8.62 KB	05 Jul 2014	Michael Pratt
bench.cpp	4.49 KB	05 Jul 2014	Michael Pratt