QGIS Application - Bug report #20747
Crash with PG rasters

2018-12-07 09:28 AM - Marco Hugentobler

Status:	Closed		
Priority:	High		
Assignee:	Even Rouault		
Category:	Rasters		
Affected QGIS version:	3.3(master)	**Regression?:**	Yes
Operating System:	Linux, Windows	**Easy fix?:**	No
Pull Request or Patch supplied:	No	**Resolution:**	fixed/implemented
Crashes QGIS or corrupts data:	Yes	**Copied to github as #:**	28567

Description

Since commit 90f381b43d606739c75ac546db9b7fd387764ca8 , zooming /panning with PG raster layers leads to crashes, both on 2.18 and on 3.4 (tested on Ubuntu Linux and Windows). It seems there is a conflict if the last render request is not yet finished and the new one is already coming (although each one has its own GDAL dataset).

Wouldn't it be better to still call cancel() instead of cancelWithoutBlocking() ?

Here is a stacktrace:

(gdb) bt
#0 0x00007ffff403fc78 in __memcpy_ssse3 () at ../sysdeps/x86_64/multiarch/memcpy-ssse3.S:2608
#1 0x00007fffe57724ea in () at /usr/lib/x86_64-linux-gnu/libpq.so.5
#2 0x00007fffe577cf39 in () at /usr/lib/x86_64-linux-gnu/libpq.so.5
#3 0x00007fffe5773868 in PQgetResult () at /usr/lib/x86_64-linux-gnu/libpq.so.5
#4 0x00007fffe5773bbd in () at /usr/lib/x86_64-linux-gnu/libpq.so.5
#5 0x00007ffff093b4de in () at /usr/lib/libgdal.so.20
#6 0x00007ffff0a4f789 in GDALRasterBand::RasterIO(GDALRWFlag, int, int, int, int, void*, int, int, GDALDataType, long long, long long, GDALRasterIOExtraArg*) () at /usr/lib/libgdal.so.20
#7 0x00007fff6670eae4 in QgsGdalProviderBase::gdalRasterIO(void*, GDALRWFlag, int, int, int, int, void*, int, int, GDALDataType, int, int, QgsRasterBlockFeedback*) (hBand=0x55555d8649f0, eRWFlag=GF_Read, nXOff=65, nYOff=90, nXSize=369, nYSize=273, pData=0x7fffb44b1440, nBufXSize=369, nBufYSize=273, eBufType=GDT_Int32, nPixelSpace=0, nLineSpace=0, feedback=0x55555d783420) at /home/marco/src/QGIS_2_18/src/providers/gdal/qgsgdalproviderbase.cpp:317
#8 0x00007fff66713f09 in QgsGdalProvider::readBlock(int, QgsRectangle const&, int, int, void*, QgsRasterBlockFeedback*) (this=0x555556f270b0, theBandNo=1, theExtent=..., thePixelWidth=1278, thePixelHeight=944, theBlock=0x7fffb40171b0, feedback=0x55555d783420) at /home/marco/src/QGIS_2_18/src/providers/gdal/qgsgdalprovider.cpp:638
#9 0x00007fff66712808 in QgsGdalProvider::block2(int, QgsRectangle const&, int, int, QgsRasterBlockFeedback*) (this=0x555556f270b0, theBandNo=1, theExtent=..., theWidth=1278, theHeight=944, feedback=0x55555d783420)
at /home/marco/src/QGIS_2_18/src/providers/gdal/qgsgdalprovider.cpp:437
#10 0x00007ffff61e965b in QgsSingleBandGrayRenderer::block2(int, QgsRectangle const&, int, int, QgsRasterBlockFeedback*) (this=0x55555d79c020, bandNo=1, extent=..., width=1278, height=944, feedback=0x55555d783420)
at /home/marco/src/QGIS_2_18/src/core/raster/qgssinglebandgrayrenderer.cpp:97
#11 0x00007ffff61cf1b4 in QgsBrightnessContrastFilter::block2(int, QgsRectangle const&, int, int, QgsRasterBlockFeedback*) (this=0x555556ce19f0, bandNo=1, extent=..., width=1278, height=944, feedback=0x55555d783420)
at /home/marco/src/QGIS_2_18/src/core/raster/qgsbrightnesscontrastfilter.cpp:130
#12 0x00007ffff61d3e31 in QgsHueSaturationFilter::block2(int, QgsRectangle const&, int, int, QgsRasterBlockFeedback*) (this=0x55555d79cd20, bandNo=1, extent=..., width=1278, height=944, feedback=0x55555d783420)
at /home/marco/src/QGIS_2_18/src/core/raster/qgshuesaturationfilter.cpp:139
#13 0x00007ffff61e78cd in QgsRasterResampleFilter::block2(int, QgsRectangle const&, int, int, QgsRasterBlockFeedback*) (this=0x555556cbc400, bandNo=1, extent=..., width=1278, height=944, feedback=0x55555d783420)
at /home/marco/src/QGIS_2_18/src/core/raster/qgsrasterresamplefilter.cpp:177

#14 0x00007ffff61c6553 in QgsRasterProjector::block2(int, QgsRectangle const&, int, int, QgsRasterBlockFeedback*) (this=0x55555d78d530, bandNo=1, extent=..., width=1278, height=944, feedback=0x55555d783420)
    at /home/marco/src/QGIS_2_18/src/core/raster/qgsrasterprojector.cpp:852
#15 0x00007ffff61ad9b0 in QgsRasterIterator::readNextRasterPart(int, int&, int&, QgsRasterBlock**, int&, int&) (this=0x7fffc2dca200, bandNumber=1, nCols=@0x7fffc2dca128: 1278, nRows=@0x7fffc2dca12c: 944, block=0x7fffc2dca150, topLeftCol=@0x7fffc2dca130: 0, topLeftRow=@0x7fffc2dca134: 0) at /home/marco/src/QGIS_2_18/src/core/raster/qgsrasteriterator.cpp:98
#16 0x00007ffff61d9b25 in QgsRasterDrawer::draw(QPainter*, QgsRasterViewPort*, QgsMapToPixel const*, QgsRenderContext const*, QgsRasterBlockFeedback*) (this=0x7fffc2dca1e0, p=0x55555d0e7fb0, viewPort=0x555555fd03a0, theQgsMapToPixel=0x555556ce12c8, ctx=0x0, feedback=0x55555d783420) at /home/marco/src/QGIS_2_18/src/core/raster/qgsrasterdrawer.cpp:57
#17 0x00007ffff61bca19 in QgsRasterLayerRenderer::render() (this=0x5555566c1b40) at /home/marco/src/QGIS_2_18/src/core/raster/qgsrasterlayerrenderer.cpp:230
#18 0x00007ffff5f14247 in QgsMapRendererCustomPainterJob::doRender() (this=0x55555d711d50) at /home/marco/src/QGIS_2_18/src/core/qgsmaprenderercustompainterjob.cpp:273
#19 0x00007ffff5f13eb4 in QgsMapRendererCustomPainterJob::staticRender(QgsMapRendererCustomPainterJob*) (self=0x55555d711d50) at /home/marco/src/QGIS_2_18/src/core/qgsmaprenderercustompainterjob.cpp:230
#20 0x00007ffff5f16775 in QtConcurrent::StoredFunctorCall1<void, void (**)(QgsMapRendererCustomPainterJob**), QgsMapRendererCustomPainterJob*>::runFunctor() (this=0x55555d756370)
    at /usr/include/qt4/QtCore/qtconcurrentstoredfunctioncall.h:277
#21 0x00007ffff5f150f9 in QtConcurrent::RunFunctionTask<void>::run() (this=0x55555d756370) at /usr/include/qt4/QtCore/qtconcurrentrunbase.h:134
#22 0x00007ffff55c5e0a in () at /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#23 0x00007ffff55d2e3c in () at /usr/lib/x86_64-linux-gnu/libQtCore.so.4
#24 0x00007fffed4be6db in start_thread (arg=0x7fffc2dcb700) at pthread_create.c:463
#25 0x00007ffff3ff288f in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.S:95

Associated revisions

Revision 0072dbbf - 2019-01-18 07:40 AM - Marco Hugentobler

Fix crash when zooming in PostGIS raster layers (ticket 20747)

Revision 9da842d4 - 2019-01-21 02:34 AM - Marco Hugentobler

Fix crash when zooming in PostGIS raster layers (ticket 20747)

History

#1 - 2018-12-07 09:52 AM - Nyall Dawson

- Assignee changed from Nyall Dawson to Even Rouault

Looks like a thread safety issue in the gdal driver. Even are you aware of any known issues like this in the driver?

#2 - 2018-12-07 09:53 AM - Nyall Dawson

- Assignee deleted (Even Rouault)

#3 - 2018-12-17 06:09 PM - Marco Hugentobler

- Assignee set to Even Rouault

@Even: do you think this could be a multithreading issue in the GDAL PostGIS-Raster driver? I've tested it with also with todays GDAL master branch, here is the stackgrace:

Thread 15 "Thread (pooled)" received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7fff2ffff700 (LWP 10469)]
0x00007fffdcb1d458 in ?? () from /usr/lib/x86_64-linux-gnu/libssl.so.1.1
(gdb) bt
#0 0x00007fffdcb1d458 in () at /usr/lib/x86_64-linux-gnu/libssl.so.1.1
#1 0x00007fffdcb18215 in () at /usr/lib/x86_64-linux-gnu/libssl.so.1.1
#2 0x00007fffdcb15a73 in () at /usr/lib/x86_64-linux-gnu/libssl.so.1.1
#3 0x00007fffdcb1b9aa in () at /usr/lib/x86_64-linux-gnu/libssl.so.1.1
#4 0x00007fffdcb257d5 in SSL_read () at /usr/lib/x86_64-linux-gnu/libssl.so.1.1
#5 0x00007fffe46bd1c2 in () at /usr/lib/x86_64-linux-gnu/libpq.so.5
#6 0x00007fffe46aeb5a in () at /usr/lib/x86_64-linux-gnu/libpq.so.5
#7 0x00007fffe46ac8bd in PQgetResult () at /usr/lib/x86_64-linux-gnu/libpq.so.5
#8 0x00007fffe46acbbd in () at /usr/lib/x86_64-linux-gnu/libpq.so.5
#9 0x00007ffff265f289 in PostGISRasterRasterBand::IRasterIO(GDALRWFlag, int, int, int, int, void*, int, int, GDALDataType, long long, long long, GDALRasterIOExtraArg*) (this=
0x7fff3c02d200, eRWFlag=<optimized out>, nXOff=24, nYOff=315, nXSize=1509, nYSize=750, pData=<optimized out>, nBufXSize=<optimized out>, nBufYSize=<optimized out>, eBufType=GDT_Float32, nPixelSpace=4, nLineSpace=5084, psExtraArg=0x7fff2fffd910) at postgisrasterrasterband.cpp:530
#10 0x00007ffff27b0485 in GDALRasterBand::RasterIO(GDALRWFlag, int, int, int, int, void*, int, int, GDALDataType, long long, long long, GDALRasterIOExtraArg*) (this=0x7fff3c02d200, eRWFlag=eRWFlag@entry=GF_Read, nXOff=24, nYOff=315, nXSize=1509, nYSize=750, pData=0x7fff4c419010, nBufXSize=1271, nBufYSize=631, eBufType=GDT_Float32, nPixelSpace=4, nLineSpace=5084, psExtraArg=0x7fff2fffd910) at gdalrasterband.cpp:361
#11 0x00007ffff27ce64c in GDALRasterBand::OverviewRasterIO(GDALRWFlag, int, int, int, int, void*, int, int, GDALDataType, long long, long long, GDALRasterIOExtraArg*) (this=this@entry=0x7fff3c038460, eRWFlag=eRWFlag@entry=GF_Read, nXOff=<optimized out>, nXOff@entry=95, nYOff=<optimized out>,
nYOff@entry=1258, nXSize=<optimized out>, nXSize@entry=6037, nYSize=<optimized out>,
nYSize@entry=2999, pData=0x7fff4c419010, nBufXSize=1271, nBufYSize=631, eBufType=GDT_Float32, nPixelSpace=4, nLineSpace=5084, psExtraArg=0x7fff2fffddb0)
at rasterio.cpp:3549
#12 0x00007ffff265f3eb in PostGISRasterRasterBand::IRasterIO(GDALRWFlag, int, int, int, int, void*, int, int, GDALDataType, long long, long long, GDALRasterIOExtraArg*) (this=0x7fff3c038460, eRWFlag=GF_Read, nXOff=95, nYOff=1258, nXSize=6037, nYSize=2999, pData=0x7fff4c419010, nBufXSize=1271, nBufYSize=631, eBufType=GDT_Float32, nPixelSpace=4, nLineSpace=5084, psExtraArg=0x7fff2fffddb0) at postgisrasterrasterband.cpp:204
#13 0x00007ffff27b0485 in GDALRasterBand::RasterIO(GDALRWFlag, int, int, int, int, void*, int, int, GDALDataType, long long, long long, GDALRasterIOExtraArg*) (this=0x7fff3c038460, eRWFlag=GF_Read, nXOff=95, nYOff=1258, nXSize=6037, nYSize=2999, pData=0x7fff4c419010, nBufXSize=1271, nBufYSize=631, eBufType=GDT_Float32, nPixelSpace=4, nLineSpace=5084, psExtraArg=0x7fff2fffddb0) at gdalrasterband.cpp:361
#14 0x00007fffb62665e8 in QgsGdalProviderBase::gdalRasterIO(void*, GDALRWFlag, int, int, int, int, void*, int, int, GDALDataType, int, int, QgsRasterBlockFeedback*) (hBand=0x7fff3c038460, eRWFlag=GF_Read, nXOff=95, nYOff=1258, nXSize=6037, nYSize=2999, pData=0x7fff4c419010, nBufXSize=1271, nBufYSize=631, eBufType=GDT_Float32, nPixelSpace=0, nLineSpace=0, feedback=0x55555885d940) at /home/marco/src/QGIS/src/providers/gdal/qgsgdalproviderbase.cpp:292
#15 0x00007fffb626ec66 in QgsGdalProvider::readBlock(int, QgsRectangle const&, int, int, void*, QgsRasterBlockFeedback*) (this=
0x5555582691b0, bandNo=1, extent=..., pixelWidth=1271, pixelHeight=631, block=0x7fff4fa3d010, feedback=0x55555885d940)
at /home/marco/src/QGIS/src/providers/gdal/qgsgdalprovider.cpp:857
#16 0x00007fffb626d1a1 in QgsGdalProvider::block(int, QgsRectangle const&, int, int, QgsRasterBlockFeedback*) (this=0x5555582691b0, bandNo=1, extent=..., width=1271, height=631, feedback=0x55555885d940) at /home/marco/src/QGIS/src/providers/gdal/qgsgdalprovider.cpp:648
#17 0x00007ffff6425d15 in QgsSingleBandGrayRenderer::block(int, QgsRectangle const&, int, int, QgsRasterBlockFeedback*) (this=
0x555558850660, bandNo=1, extent=..., width=1271, height=631, feedback=0x55555885d940)
at /home/marco/src/QGIS/src/core/raster/qgssinglebandgrayrenderer.cpp:91

#4 - 2018-12-17 07:05 PM - Even Rouault

The driver has a PG connection cache, but it takes into account the thread. However a potential issue is if you create 2 postgisraster datasets in the same thread, and use them afterwards in different threads: as they have been created by the same thread, they will share the same PG connection. When you get the crash, did you check the stacktrace of other threads to see if there was also another one in a PG request (and if so, do they use the same PG connection) ?

#5 - 2018-12-18 12:35 PM - Marco Hugentobler

There are two threads doing PG requests (probably the old one which is not finished yet and the new one). They however are using different PGconn objects in PostGISRasterRasterBand::IRasterIO

#6 - 2018-12-18 03:51 PM - Marco Hugentobler

The datasets are created in different threads. There is a dataset cache in qgsgdalprovider, so the datasets are normally not used in the thread they were created. Ist this a problem?
If the dataset cache in qgsgdalprovider is disabled, the crash does not happen.

#7 - 2018-12-18 04:00 PM - Even Rouault

> *so the datasets are normally not used in the thread they were created. Ist this a problem?*

If they are really created by different threads, that shouldn't be a problem, but indeed QgsGdalProvider has a GDALDataset recycling mechanism that could (maybe ?) cause datasets originally created by the same thread to be used later by different threads. But above you mentionned, the PGconn were different, so it doesn't seem to be that issue. I don't have a theory.

#8 - 2018-12-27 10:39 AM - Marco Hugentobler

- Affected QGIS version changed from 3.4.0 to 3.3(master)

#9 - 2019-01-17 05:41 PM - Marco Hugentobler

I think the theory with two datasets created by the same thread was right. I didn't see it in the debugger because one thread already left the section when the crash happened.

https://github.com/qgis/QGIS/pull/8890 provides a fix. If the driver is postgis raster, the dataset is not cloned.

#10 - 2019-01-22 04:21 PM - Jürgen Fischer

- Resolution set to fixed/implemented
- Status changed from Open to Closed

fixed in commit:0072dbbfa00036796409a59045d71545a2984dc0 and commit:9da842d48881fe54af734f033be91c2772fe1177