GDP - Bug #16

Threaded GDP Clients Fail

07/18/2016 09:26 AM - Nitesh Mor

Status:	Closed	**Start date:**	07/18/2016
Priority:	Normal	**Due date:**	08/15/2016
Assignee:	Eric Allman	**% Done:**	0%
Category:	libgdp	**Estimated time:**	0.00 hour
Target version:			

Description

On behalf of *Rico*:

I've been doing experiments for our "central" service that I've mentioned before. Currently, I have multiple threads. Some of these threads have an active GCL file handle for writing and some have GCL file handle for subscribing to GCLs.

The initial setup has three GCL file handles for writing and two GCL file handles subscribed to different GCLs in different threads. This works fine except that new records in the GCL somehow get "stucked" until another new record is appended. To clarify that: at start, new record A is appended to GCL 1. This will be received by the get_next_event() instance method of the handle of GCL1. When I append new record B, get_next_event() will not give me anything until I append new record C. By then, get_next_event() will give me record B first, and then record C. It seems that two events got stucked somehow. This behavior will repeat again and again.

The next setup is that I have four GCL file handles for writing and three GCL file handles subscribed to different GCLs in different threads. This will sometime fail when subscribing to third GCL giving a "network pdu write failure" error. In times that it doesn't give an error (which is quite rare), the same scenario as described above happens, but instead of two events being stucked, there are now three events being stucked. I think the number of stucked events is dependent on the number of GCL handles that are subscribed.

History

#1 - 08/09/2016 03:51 PM - Nitesh Mor

- Due date set to 08/15/2016

#2 - 08/30/2016 10:19 AM - Nitesh Mor

- File stuck-sub.py added

I think I can reproduce this bug now. At a first look, this does not appear to be specific to the Python API, but needs some further testing. Attached is an example program that starts up a configurable number of threads for appends and subscriptions. Here's the output that I get; an example of problematic lines is at time 7.013285, 8.016445 and 8.018598:

```
$ ./stuck-sub.py 2 2 5 10
 0.000015 ## MainThread] Starting AThread1
 0.000874 ## MainThread] Starting AThread2
 0.001687 ## MainThread] Starting SThread1
 0.003374 ## AThread1] <gdp.GDP_GCL.GDP_GCL object at 0x7f93480fa110>
 0.003965 ## AThread2] <gdp.GDP_GCL.GDP_GCL object at 0x7f93480fa210>
 0.004018 ## MainThread] Starting SThread2
 0.005122 ## SThread1] <gdp.GDP_GCL.GDP_GCL object at 0x7f93480fa110>
 0.005528 ## SThread2] <gdp.GDP_GCL.GDP_GCL object at 0x7f93480fa210>
 6.009890 ## AThread1] Appending "log1-AThread1-0" to log1
 6.012592 ## SThread1] Got data: "log1-AThread1-0" on log1
 7.013285 ## AThread1] Appending "log1-AThread1-1" to log1
 8.016445 ## AThread1] Appending "log1-AThread1-2" to log1
 8.018598 ## SThread1] Got data: "log1-AThread1-1" on log1
 8.018977 ## SThread1] Got data: "log1-AThread1-2" on log1
 9.019804 ## AThread1] Appending "log1-AThread1-3" to log1
10.022866 ## AThread1] Appending "log1-AThread1-4" to log1
10.025178 ## SThread1] Got data: "log1-AThread1-3" on log1
10.025426 ## SThread1] Got data: "log1-AThread1-4" on log1
11.015211 ## AThread2] Appending "log2-AThread2-0" to log2
11.017364 ## SThread2] Got data: "log2-AThread2-0" on log2
11.026064 ## AThread1] Appending "log1-AThread1-5" to log1
11.029483 ## SThread1] Got data: "log1-AThread1-5" on log1
12.018308 ## AThread2] Appending "log2-AThread2-1" to log2
```

```
12.020499 ## SThread2] Got data: "log2-AThread2-1" on log2
12.067137 ## AThread1] Appending "log1-AThread1-6" to log1
12.069445 ## SThread1] Got data: "log1-AThread1-6" on log1
13.021757 ## AThread2] Appending "log2-AThread2-2" to log2
13.023927 ## SThread2] Got data: "log2-AThread2-2" on log2
13.070698 ## AThread1] Appending "log1-AThread1-7" to log1
13.072925 ## SThread1] Got data: "log1-AThread1-7" on log1
14.025260 ## AThread2] Appending "log2-AThread2-3" to log2
14.027593 ## SThread2] Got data: "log2-AThread2-3" on log2
14.073742 ## AThread1] Appending "log1-AThread1-8" to log1
14.075762 ## SThread1] Got data: "log1-AThread1-8" on log1
15.028820 ## AThread2] Appending "log2-AThread2-4" to log2
15.031395 ## SThread2] Got data: "log2-AThread2-4" on log2
15.076953 ## AThread1] Appending "log1-AThread1-9" to log1
15.079309 ## SThread1] Got data: "log1-AThread1-9" on log1
16.032583 ## AThread2] Appending "log2-AThread2-5" to log2
16.034825 ## SThread2] Got data: "log2-AThread2-5" on log2
17.036029 ## AThread2] Appending "log2-AThread2-6" to log2
18.039092 ## AThread2] Appending "log2-AThread2-7" to log2
18.041472 ## SThread2] Got data: "log2-AThread2-6" on log2
18.041732 ## SThread2] Got data: "log2-AThread2-7" on log2
19.042383 ## AThread2] Appending "log2-AThread2-8" to log2
20.045391 ## AThread2] Appending "log2-AThread2-9" to log2
20.047909 ## SThread2] Got data: "log2-AThread2-8" on log2
```

#3 - 08/31/2016 01:04 PM - Nitesh Mor

- File stuck-sub.c added

- Category changed from Python API to libgdp

- Assignee changed from Nitesh Mor to Eric Allman

Reproduced with C library. See attached sample program, here's the output (ignore the timestamp precision, for the moment):

```
$ ./stuck-sub 2 2 5 10
 0.000000 ## MainThread] Starting AThread1
 0.000000 ## MainThread] Starting AThread2
 0.000000 ## MainThread] Starting SThread1
 0.000000 ## MainThread] Starting SThread2
 6.000000 ## AThread1] Appending "log1-AThread1-0" to log1
 6.000000 ## SThread1] Got data: "log1-AThread1-0" on log1
 7.000000 ## AThread1] Appending "log1-AThread1-1" to log1
 8.000000 ## AThread1] Appending "log1-AThread1-2" to log1
 8.000000 ## SThread1] Got data: "log1-AThread1-1" on log1
 8.000000 ## SThread1] Got data: "log1-AThread1-2" on log1
 9.000000 ## AThread1] Appending "log1-AThread1-3" to log1
10.000000 ## AThread1] Appending "log1-AThread1-4" to log1
10.000000 ## SThread1] Got data: "log1-AThread1-3" on log1
10.000000 ## SThread1] Got data: "log1-AThread1-4" on log1
```

```
11.000000 ## AThread2] Appending "log2-AThread2-0" to log2
11.000000 ## SThread2] Got data: "log2-AThread2-0" on log2
11.000000 ## AThread1] Appending "log1-AThread1-5" to log1
11.000000 ## SThread1] Got data: "log1-AThread1-5" on log1
12.000000 ## AThread2] Appending "log2-AThread2-1" to log2
12.000000 ## SThread2] Got data: "log2-AThread2-1" on log2
12.000000 ## AThread1] Appending "log1-AThread1-6" to log1
12.000000 ## SThread1] Got data: "log1-AThread1-6" on log1
13.000000 ## AThread2] Appending "log2-AThread2-2" to log2
13.000000 ## SThread2] Got data: "log2-AThread2-2" on log2
13.000000 ## AThread1] Appending "log1-AThread1-7" to log1
13.000000 ## SThread1] Got data: "log1-AThread1-7" on log1
14.000000 ## AThread2] Appending "log2-AThread2-3" to log2
14.000000 ## SThread2] Got data: "log2-AThread2-3" on log2
14.000000 ## AThread1] Appending "log1-AThread1-8" to log1
14.000000 ## SThread1] Got data: "log1-AThread1-8" on log1
15.000000 ## AThread2] Appending "log2-AThread2-4" to log2
15.000000 ## SThread2] Got data: "log2-AThread2-4" on log2
15.000000 ## AThread1] Appending "log1-AThread1-9" to log1
15.000000 ## SThread1] Got data: "log1-AThread1-9" on log1
16.000000 ## AThread2] Appending "log2-AThread2-5" to log2
16.000000 ## SThread2] Got data: "log2-AThread2-5" on log2
17.000000 ## AThread2] Appending "log2-AThread2-6" to log2
18.000000 ## AThread2] Appending "log2-AThread2-7" to log2
18.000000 ## SThread2] Got data: "log2-AThread2-6" on log2
18.000000 ## SThread2] Got data: "log2-AThread2-7" on log2
19.000000 ## AThread2] Appending "log2-AThread2-8" to log2
20.000000 ## AThread2] Appending "log2-AThread2-9" to log2
20.000000 ## SThread2] Got data: "log2-AThread2-8" on log2
20.000000 ## SThread2] Got data: "log2-AThread2-9" on log2
```

#4 - 09/02/2016 04:44 PM - Eric Allman

- Subject changed from Python API of GDP in Threads to Threaded GDP Clients Fail

- Status changed from New to In Progress

I've found some problems, and it behaves differently now, but it still isn't correct. Some of the problems may be in gdplogd as well as the library. Disturbingly, it might require a change to the gdp_event API to somehow pass through the request id. Since rids have not been externally visible, this would be a major change. More study is needed.

I changed the name of the issue; it isn't specific to Python.

#5 - 09/04/2016 03:23 PM - Eric Allman

- Status changed from In Progress to Resolved

I believe this is fixed by commit:1aa4fc2c. The problem was that the lack of a lock around the open calls permitted multiple name ⇒ GCL handle mappings. The code to associate responses with requests assumed these mappings were unique. The fix isn't ideal since the mutex currently includes the calls to the log server; it really only needs to be while the data structures are being set up. However, that would require changing when the name ⇒ handle mapping was instantiated, and I thought there might be some issues with that.

#6 - 09/04/2016 04:10 PM - Eric Allman

- File stuck-sub.c added

I've added an updated version of stuck-sub.c that gives more information, such as the return status of various GDP APIs (instead of assuming that they all succeed).

#7 - 09/05/2016 04:23 PM - Nitesh Mor

- Status changed from Resolved to Closed

Verified that it works from Python API as well. Marking as closed.

Files

stuck-sub.py	2.79 KB	08/30/2016	Nitesh Mor
stuck-sub.c	4.2 KB	08/31/2016	Nitesh Mor
stuck-sub.c	5.02 KB	09/04/2016	Eric Allman