GDP - Bug #107

gdp_gcl_close() hangs the second time

06/12/2017 04:45 PM - Anonymous

Status:	Closed	**Start date:**	06/12/2017
Priority:	Normal	**Due date:**	
Assignee:	Eric Allman	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

The attached Ptolemy model hangs during wrapup in a somewhat non-deterministic fashion.

Summary: It looks like a lock is sometimes being held the second time gdp_gcl_close() is being called.

Details:

To install Ptolemy II, see https://www.icyphy.org/accessors/wiki/Main/PtolemyII

After building using ant, on a laptop with a camera, download the attached file and invoke:

```
$PTII/bin/ptinvoke AprilTagImageAnnotate.xml
```

Run the model a few times. Eventually, the model will hang with the text "Wrapping up" in the status bar.

I was not able to get a stack trace using jstack, but I was able to get one with jvisualvm that showed that org.terraswarm.gdp.GDP_GCL.close() was being called.

Below is the GDP_GCL close() method from gdp/lang/java/org/terraswarm/gdp/GDP_GCL.java

```
    /** Close the GCL.

     */
    public void close() {
        // If close() is called twice, then the C code aborts the process.

        // See https://gdp.cs.berkeley.edu/redmine/issues/83

        if (gclh != null) {
            // Remove ourselves from the global list.

            _allGclhs.remove(gclh);
            // Free the associated gdp_gcl_t.

            Gdp07Library.INSTANCE.gdp_gcl_close(gclh);
            gclh = null;
        }
    }
```

Setting the debugging to 40 was useful.

During a regular run, the output was:

```
>>> gdp_gcl_close(4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY)
_gdp_gcl_close: GCL@0x7fda4defa440: 4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        iomode = 1, refcnt = 1, reqs = 0x7fda53855370, nrecs = 4209099
        flags = 0xa<INCACHE,INUSE>
```

```
        freefunc = 0x0, gclmd = 0x7fda4defacf0, digest = 0x0
        utime = 2017-06-12 23:26:26, x = 0x0
_gdp_req_new: allocated new pdu @ 0x7fda4dbb8440

>>> _gdp_invoke(req=0x7fda49850250 rid=0): CMD_CLOSE (69), gcl@0x7fda4defa440
        datum @ 0x7fda4dbb4500: recno -1, len 0, no timestamp
_gdp_invoke: sending 69, retries=2
_gdp_pdu_out, fd = 129, basemd = 0x0:
    PDU@0x7fda4dbb8440:
        v=3, ttl=15, rsvd1=0, cmd=69=CMD_CLOSE
        dst=4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        src=-OJzaemscJ_3GuB-3EWQJkq0y9MRBrLMILbLmNvC0cE
        rid=0, olen=0, chan=0x0, seqno=0
        flags=0
        datum=0x7fda4dbb4500, recno=(none), dbuf=0x7fda4dfd6990, dlen=0
                ts=(none)
        sigmdalg=0x0, siglen=0, sig=0x0
        total header=80
_gdp_pdu_out: sending PDU:
00000000 03 0f 00 45 e1 13 e6 38 41 1b a2 e8 38 ea b1 31
00000010 6a 55 fc 17 d1 dd 18 a0 7d 68 7e 64 03 38 53 7d
00000020 73 56 e0 36 f8 e2 73 69 e9 ac 70 9f f7 1a e0 7e
00000030 dc 45 90 26 4a b4 cb d3 11 06 b2 cc 20 b6 cb 98
00000040 db c2 d1 c1 00 00 00 00 00 00 00 00 00 00 00 00
_gdp_invoke: waiting on 0x7fda49850250

        >>>>>  _gdp_pdu_in  >>>>>
_gdp_pdu_in: read PDU header:
00000000 03 0f 00 80 f8 e2 73 69 e9 ac 70 9f f7 1a e0 7e
00000010 dc 45 90 26 4a b4 cb d3 11 06 b2 cc 20 b6 cb 98
00000020 db c2 d1 c1 e1 13 e6 38 41 1b a2 e8 38 ea b1 31
00000030 6a 55 fc 17 d1 dd 18 a0 7d 68 7e 64 03 38 53 7d
00000040 73 56 e0 36 00 00 00 00 00 00 02 02 00 00 00 00
00000050 00 00 00 00 00 40 39 cb
_gdp_pdu_in: reading 0 data bytes (0 available)
_gdp_pdu_in(ACK_SUCCESS) => OK
PDU@0x7fda4dfd6890:
        v=3, ttl=15, rsvd1=0, cmd=128=ACK_SUCCESS
        dst=-OJzaemscJ_3GuB-3EWQJkq0y9MRBrLMILbLmNvC0cE
        src=4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        rid=0, olen=8, chan=0x7fda53f03b40, seqno=0
        flags=0x2<HAS_RECNO>
        datum=0x7fda53855490, recno=4209099, dbuf=0x7fda53855ab0, dlen=0
                ts=(none)
        sigmdalg=0x0, siglen=0, sig=0x0
        total header=96

*** Processing ack/nak 128=ACK_SUCCESS from socket 129
gdp_pdu_proc_resp(0x7fda4dfd6890 ACK_SUCCESS) gcl 0x7fda4defa440
gdp_pdu_proc_resp: searching gcl 0x7fda4defa440 for rid 0
_gdp_req_dispatch >>> ACK_SUCCESS (128) [gcl->refcnt 1], req@0x7fda49850250:
    nextrec=0, numrecs=0, chan=0x7fda53f03b40
    postproc=0x0, sub_cb=0x0, udata=0x0
    state=WAITING, stat=OK
    act_ts=2017-06-12 23:26:30.134368000Z
    flags=0x100<ON_CHAN_LIST>
    GCL@0x7fda4defa440: 4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        iomode = 1, refcnt = 1, reqs = 0x7fda53855370, nrecs = 4209099
        flags = 0xa<INCACHE,INUSE>
    cPDU@0x7fda4dbb8440:
        v=3, ttl=15, rsvd1=0, cmd=69=CMD_CLOSE
        dst=4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        src=-OJzaemscJ_3GuB-3EWQJkq0y9MRBrLMILbLmNvC0cE
        rid=0, olen=0, chan=0x0, seqno=0
        flags=0
        datum=0x7fda4dbb4500, recno=(none), dbuf=0x7fda4dfd6990, dlen=0
                ts=(none)
```

```
        sigmdalg=0x0, siglen=0, sig=0x0
        total header=80
    rPDU@0x7fda4dfd6890:
        v=3, ttl=15, rsvd1=0, cmd=128=ACK_SUCCESS
        dst=-OJzaemscJ_3GuB-3EWQJkq0y9MRBrLMILbLmNvC0cE
        src=4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        rid=0, olen=8, chan=0x7fda53f03b40, seqno=0
        flags=0x2<HAS_RECNO>
        datum=0x7fda53855490, recno=4209099, dbuf=0x7fda53855ab0, dlen=0
                ts=(none)
        sigmdalg=0x0, siglen=0, sig=0x0
    total header=96

ack_success: received ACK_SUCCESS for CMD_CLOSE
_gdp_req_dispatch <<< ACK_SUCCESS [gcl->refcnt 1]
    OK [200 = 0xc8]
gdp_pdu_proc_resp: signaling req@0x7fda49850250:
    nextrec=0, numrecs=0, chan=0x7fda53f03b40
    postproc=0x0, sub_cb=0x0, udata=0x0
    state=WAITING, stat=OK [200 = 0xc8]
    act_ts=2017-06-12 23:26:30.191747000Z
    flags=0x103<ASYNCIO,DONE,ON_CHAN_LIST>
    GCL@0x7fda4defa440: 4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        iomode = 1, refcnt = 1, reqs = 0x7fda53855370, nrecs = 4209099
        flags = 0xa<INCACHE,INUSE>
    cPDU@0x7fda4dbb8440:
        v=3, ttl=15, rsvd1=0, cmd=69=CMD_CLOSE
        dst=4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        src=-OJzaemscJ_3GuB-3EWQJkq0y9MRBrLMILbLmNvC0cE
        rid=0, olen=0, chan=0x0, seqno=0
        flags=0
        datum=0x0
        total header=80
    rPDU@0x7fda4dfd6890:
        v=3, ttl=15, rsvd1=0, cmd=128=ACK_SUCCESS
        dst=-OJzaemscJ_3GuB-3EWQJkq0y9MRBrLMILbLmNvC0cE
        src=4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        rid=0, olen=8, chan=0x7fda53f03b40, seqno=0
        flags=0x2<HAS_RECNO>
        datum=0x7fda4dbb4500, recno=4209099, dbuf=0x7fda4dfd6990, dlen=0
                ts=(none)
        sigmdalg=0x0, siglen=0, sig=0x0
        total header=96
gdp_pdu_proc_resp <<< done
<<< _gdp_invoke(0x7fda49850250 rid=0) CMD_CLOSE: OK [200 = 0xc8]
  req@0x7fda49850250:
    nextrec=0, numrecs=0, chan=0x7fda53f03b40
    postproc=0x0, sub_cb=0x0, udata=0x0
    state=ACTIVE, stat=OK [200 = 0xc8]
    act_ts=2017-06-12 23:26:30.191747000Z
    flags=0x103<ASYNCIO,DONE,ON_CHAN_LIST>
    GCL@0x7fda4defa440: 4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        iomode = 1, refcnt = 1, reqs = 0x7fda53855370, nrecs = 4209099
        flags = 0xe<INCACHE,ISLOCKED,INUSE>
    cPDU@0x7fda4dbb8440:
        v=3, ttl=15, rsvd1=0, cmd=69=CMD_CLOSE
        dst=4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        src=-OJzaemscJ_3GuB-3EWQJkq0y9MRBrLMILbLmNvC0cE
        rid=0, olen=0, chan=0x0, seqno=0
        flags=0
        datum=0x0
        total header=80
    rPDU@0x7fda4dfd6890:
        v=3, ttl=15, rsvd1=0, cmd=128=ACK_SUCCESS
        dst=-OJzaemscJ_3GuB-3EWQJkq0y9MRBrLMILbLmNvC0cE
        src=4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        rid=0, olen=8, chan=0x7fda53f03b40, seqno=0
```

```
        flags=0x2<HAS_RECNO>
        datum=0x7fda4dbb4500, recno=4209099, dbuf=0x7fda4dfd6990, dlen=0
                ts=(none)
        sigmdalg=0x0, siglen=0, sig=0x0
        total header=96

_gdp_gcl_freehandle(0x7fda4defa440)
_gdp_gcl_cache_drop: 4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY => 0x7fda4defa440
gdp_gclmd_free(0x7fda4defacf0)
<<< gdp_gcl_close(4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY): OK [200 = 0xc8]
11796 ms. Memory: 974336K Free: 793711K (81%)
```

However, when the process hung, the output was:

```
_gdp_pdu_in(ACK_DATA_CONTENT) => OK
PDU@0x7fda4dbb8440:
        v=3, ttl=15, rsvd1=0, cmd=133=ACK_DATA_CONTENT
        dst=-OJzaemscJ_3GuB-3EWQJkq0y9MRBrLMILbLmNvC0cE
        src=4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        rid=4, olen=24, chan=0x7fda53f03b40, seqno=0
        flags=0xa<HAS_RECNO,HAS_TS>
        datum=0x7fda4dbb4500, recno=4209107, dbuf=0x7fda4dfd6990, dlen=355
                ts=2017-06-12 23:27:01.707393000Z
        sigmdalg=0x3, siglen=78, sig=0x7fda50c27690
        total header=128

*** Processing ack/nak 133=ACK_DATA_CONTENT from socket 129

>>> gdp_gcl_close(4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY)
_gdp_gcl_close: GCL@0x7fda53fb24d0: 4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        iomode = 1, refcnt = 2, reqs = 0x7fda53855370, nrecs = 4209106
        flags = 0xe<INCACHE,ISLOCKED,INUSE>
        freefunc = 0x0, gclmd = 0x7fda53fb3ab0, digest = 0x0
        utime = 2017-06-12 23:26:57, x = 0x0

subscr_poker_thread: poking

subscr_poker_thread: poking
```

I used lldb to attach to the process:

```
lldb -p 56007
```

Below is how I listed the threads and got a backtrace of the thread that is calling gdp_gcl_close(). It looks like it is waiting on a lock.

```
(lldb) thread list
thread list
error: jna2799580332201107342.tmp(0x0000000154913000) debug map object file '/Users/cxh/ptII/vendo
rs/gdp/gdp/gdp/gdp_version.o' has changed (actual time is 0x5932\
0054, debug map time is 0x592f7371) since this executable was linked, file will be ignored
Process 56007 stopped
* thread #1: tid = 0x450b4b, 0x00007fffe492934a libsystem_kernel.dylib`mach_msg_trap + 10, name =
'AppKit Thread', queue = 'com.apple.main-thread', stop reason = \
signal SIGSTOP
  thread #2: tid = 0x450b4c, 0x00007fffe4930f46 libsystem_kernel.dylib`__semwait_signal + 10
  thread #3: tid = 0x450b51, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #4: tid = 0x450b52, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #5: tid = 0x450b53, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
```

```
  thread #6: tid = 0x450b54, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #7: tid = 0x450b55, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #8: tid = 0x450b56, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #9: tid = 0x450b57, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #10: tid = 0x450b58, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #11: tid = 0x450b59, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #12: tid = 0x450b5a, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #13: tid = 0x450b5b, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: Reference Handler'
  thread #14: tid = 0x450b5c, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: Finalizer'
  thread #15: tid = 0x450b65, 0x00007fffe4929386 libsystem_kernel.dylib`semaphore_wait_trap + 10,
name = 'Java: Signal Dispatcher'
  thread #16: tid = 0x450b66, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: C2 CompilerThread0'
  thread #17: tid = 0x450b67, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: C2 CompilerThread1'
  thread #18: tid = 0x450b68, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: C2 CompilerThread2'
  thread #19: tid = 0x450b69, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: C1 CompilerThread3'
  thread #20: tid = 0x450b6a, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: Service Thread'
  thread #21: tid = 0x450b6b, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #22: tid = 0x450ba2, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: AWT-Shutdown'
  thread #23: tid = 0x450bb4, 0x00007fffe492934a libsystem_kernel.dylib`mach_msg_trap + 10, name =
 'com.apple.NSEventThread'
  thread #24: tid = 0x450bb6, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: Java2D Queue Flusher'
  thread #25: tid = 0x450bb9, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: Java2D Disposer'
  thread #26: tid = 0x450bba, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: AWT-EventQueue-0'
  thread #27: tid = 0x450c8f, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: atomic-processor-1'
  thread #28: tid = 0x450c94, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: webcam-discovery-service'
  thread #29: tid = 0x450c9b, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'CVDisplayLink'
  thread #30: tid = 0x450ca5, 0x00007fffe4930bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name
 = 'Java: TimerQueue'
  thread #31: tid = 0x450d0b, 0x00007fffe4930c22 libsystem_kernel.dylib`__psynch_mutexwait + 10
  thread #32: tid = 0x450d0d, 0x00007fffe4930f46 libsystem_kernel.dylib`__semwait_signal + 10
  thread #33: tid = 0x450e07, 0x00007fffe4930c22 libsystem_kernel.dylib`__psynch_mutexwait + 10, n
ame = 'Java: AprilTagImageAnnotate'
  thread #34: tid = 0x45216b, 0x00007fffe493144e libsystem_kernel.dylib`__workq_kernreturn + 10
(lldb) thread select 33
thread select 33
(lldb) ^[[1G^[[J* thread #33, name = 'Java: AprilTagImageAnnotate'
    frame #0: 0x00007fffe4930c22 libsystem_kernel.dylib`__psynch_mutexwait + 10
libsystem_kernel.dylib`__psynch_mutexwait:
->  0x7fffe4930c22 <+10>: jae    0x7fffe4930c2c            ; <+20>
    0x7fffe4930c24 <+12>: movq   %rax, %rdi
    0x7fffe4930c27 <+15>: jmp    0x7fffe4929caf            ; cerror_nocancel
    0x7fffe4930c2c <+20>: retq
^[[1G^[[J(lldb)  ^[[8G

(lldb) ^[[1G^[[J* thread #33, name = 'Java: AprilTagImageAnnotate'
    frame #0: 0x00007fffe4930c22 libsystem_kernel.dylib`__psynch_mutexwait + 10
libsystem_kernel.dylib`__psynch_mutexwait:
->  0x7fffe4930c22 <+10>: jae    0x7fffe4930c2c            ; <+20>
    0x7fffe4930c24 <+12>: movq   %rax, %rdi
    0x7fffe4930c27 <+15>: jmp    0x7fffe4929caf            ; cerror_nocancel
    0x7fffe4930c2c <+20>: retq
.
(lldb) thread backtrace
```

```
thread backtrace
* thread #33, name = 'Java: AprilTagImageAnnotate'
  * frame #0: 0x00007fffe4930c22 libsystem_kernel.dylib`__psynch_mutexwait + 10
    frame #1: 0x00007fffe4a1be6e libsystem_pthread.dylib`_pthread_mutex_lock_wait + 100
    frame #2: 0x000000015492abe5 jna2799580332201107342.tmp`_ep_thr_mutex_lock(mtx=0x00007fda53fb2
4d0, file="gdp_gcl_ops.c", line=<unavailable>, name=<unavailable\
>) at ep_thr.c:403 [opt]
    frame #3: 0x00000001549197de jna2799580332201107342.tmp`_gdp_gcl_lock_trace(gcl=0x00007fda53fb
24d0, file=<unavailable>, line=<unavailable>, id=<unavailable>) \
at gdp_gcl_cache.c:496 [opt]
    frame #4: 0x000000015491b41c jna2799580332201107342.tmp`_gdp_gcl_close(gcl=0x00007fda53fb24d0,
 chan=0x00007fda53f03b40, reqflags=0) at gdp_gcl_ops.c:503 [opt]
    frame #5: 0x00000001549150a2 jna2799580332201107342.tmp`gdp_gcl_close(gcl=0x00007fda53fb24d0)
at gdp_api.c:447 [opt]
    frame #6: 0x0000000122eaf514 jna6492267650297403476.tmp`ffi_call_unix64 + 76
    frame #7: 0x0000000122eaf42d jna6492267650297403476.tmp`ffi_call + 781
    frame #8: 0x0000000122ea6565 jna6492267650297403476.tmp`___lldb_unnamed_symbol19$$jna6492267650
297403476.tmp + 2181
    frame #9: 0x000000010f14c9d4
    frame #10: 0x000000010f13d2bd
    frame #11: 0x000000010f13d040
    frame #12: 0x000000010f13d040
    frame #13: 0x000000010f13d040
    frame #14: 0x000000010f13d114
    frame #15: 0x000000010f13d114
    frame #16: 0x000000010f13d2bd
    frame #17: 0x0000000110b3b94c
    frame #18: 0x000000010fe80210
    frame #19: 0x000000010faac394
    frame #20: 0x000000011026c274
    frame #21: 0x000000010fe80210
(lldb)
```

Related issues:

Related to GDP - Support #109: GDP with accessors (an umbrella item)	**New**

History

#1 - 06/16/2017 02:14 PM - Eric Allman

- Status changed from New to In Progress

- Assignee set to Eric Allman

I'm not sure I can follow what's going on with the debugging output you've given me. If this is in a single run then of course it doesn't work: you can't close a GCL twice (i.e., free memory twice) without opening it again between the two close calls. This might happen if Java were somehow calling gdp_gcl_close but was still holding on to the pointer. If this is between two runs then I'm not understanding the lifecycle of the process.

However, the debug output is pretty mysterious. You are showing:

```
_gdp_pdu_in(ACK_DATA_CONTENT) => OK
PDU@0x7fda4dbb8440:
        v=3, ttl=15, rsvd1=0, cmd=133=ACK_DATA_CONTENT
        dst=-OJzaemscJ_3GuB-3EWQJkq0y9MRBrLMILbLmNvC0cE
        src=4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY
        rid=4, olen=24, chan=0x7fda53f03b40, seqno=0
        flags=0xa<HAS_RECNO,HAS_TS>
        datum=0x7fda4dbb4500, recno=4209107, dbuf=0x7fda4dfd6990, dlen=355
                ts=2017-06-12 23:27:01.707393000Z
        sigmdalg=0x3, siglen=78, sig=0x7fda50c27690
        total header=128

*** Processing ack/nak 133=ACK_DATA_CONTENT from socket 129

>>> gdp_gcl_close(4RPmOEEboug46rExalX8F9HdGKB9aH5kAzhTfXNW4DY)
```

I don't know of any way that an incoming ACK_DATA_CONTENT could immediately call gdp_gcl_close unless memory is badly trashed — not

impossible, but it's very unlikely that a function pointer gets trashed so that it points to a different function which it invokes with the correct parameters and without crashing. In particular, gdp_gcl_close is the external API, and I have verified that there are no calls to that function from within libgdp (i.e., only applications).

If ACK_DATA_CONTENT is *not* calling gdp_gcl_close, that implies that another thread is. So one scenario is that another thread (A) has the GCL locked, the ACK_DATA_CONTENT comes in on a second thread (B), and then another thread (maybe A, but more likely C) decides to close the GCL. That seems pretty far-fetched, but it's the best idea I have so far. It would be interesting to see the other thread backtraces, especially thread 31.

#2 - 06/16/2017 03:01 PM - Anonymous

Thanks for taking a look. Under Mac OS, you should be able to look at thread 31 by using lldb -p pid.

Have you been able to replicate this?

I can't replicate this with the $PTII/lib/libgdp.0.7.dylib from May 31.

I updated the GDP and I can't reproduce this bug right now, the model runs fine? I guess we could close it, but I feel like we definitely saw a problem.

I remember we went back and forth a bit about gdp_gcl_close(). My point is that it should be OK to call it multiple times. I believe your point, which is well taken and I understand the reasoning, is that because of the way the code is structured, calling gdp_gcl_close() twice is, in reality, freeing the gcl.

I don't know if we ever reached a resolution about this issue.

gdp/lang/java/org/terraswarm/gdp/GDP_GCL.java looked like:

```
    /** Close the GCL.

     */
    public void close() {
        // If close() is called twice, then the C code aborts the process.

        // See https://gdp.cs.berkeley.edu/redmine/issues/83

        if (gclh != null) {
            // Remove ourselves from the global list.

            _allGclhs.remove(gclh);
            // Free the associated gdp_gcl_t.

            Gdp07Library.INSTANCE.gdp_gcl_close(gclh);
            gclh = null;
        }
    }
```

I added

```
        System.out.println("GDP_GCL.java close(): current thread: " + Thread.currentThread()c.getName());
```

and I can see that the Finalizer and the main Ptolemy thread are calling this code.

```
GDP_GCL.java close(): current thread: Finalizer
GDP_GCL.java close(): current thread: Finalizer
GDP_GCL.java close(): current thread: Finalizer
GDP_GCL.java close(): current thread: Finalizer
GDP_GCL.java close(): current thread: AprilTagImageAnnotate
```

So there could be a threading issue. I modified close() to be:

```
    /** Close the GCL.

     */
    public void close() {
        // If close() is called twice, then the C code aborts the process.

        // See https://gdp.cs.berkeley.edu/redmine/issues/83

        //  Added synchronization, see https://gdp.cs.berkeley.edu/redmine/issues/107

    synchronized (gclh) {
            if (gclh != null) {
                // Remove ourselves from the global list.

                _allGclhs.remove(gclh);

                // Free the associated gdp_gcl_t.

                Gdp07Library.INSTANCE.gdp_gcl_close(gclh);
                gclh = null;
            }
        }
    }
```

I just checked in the above change to the gdp repo and $PTII/lib/gdp-0.7.2.jar.

If you can't replicate this bug, then we should ask Edward to replicate it and if he can't replicate it, then we should close it.

#3 - 06/16/2017 03:22 PM - Eric Allman

FIrst off, no I haven't been able to figure out how to get anything to run reliably. I got a camera image to pop up once, but I'm not sure I know how I did that, and other times didn't get that far. I'm also not sure which processes to attach to, and so forth, nor do I understand the difference between vergil, capecode, and some of the others. I think we're just going to have to sit down and walk through this sometime.

However, I emphatically disagree with one of your points: it should ***not*** be possible to close an open entity more than once, any more than it is legal to fclose a file more than once. That's a basic "use after free" error. The exception is that in the GDP, opening the same log more than once gives you references to a single data structure, so:

```
g1 = gdp_gcl_open(log1, ...);
```

```
g2 = gdp_gcl_open(log1, ...);
gdp_gcl_close(g1);
gdp_gcl_close(g2);
```

should work. But:

```
g1 = gdp_gcl_open(log1, ...);
gdp_gcl_close(g1);
gdp_gcl_close(g1);
```

does not work, should not work, and never will work.

#4 - 06/16/2017 04:37 PM - Anonymous

I understand the design of gdp_gcl_close(). I think it is safe to say that we disagree and move on. Let's see if the synchronization I added helps.

About capecode vs vergil etc. $PTII/bin/capecode and $PTII/bin/vergil are symbolic links to $PTII/bin/ptinvoke. $PTII/bin/ptinvoke is created by $PTII/configure from $PTII/bin/ptinvoke.in. What ptinvoke does is build up the java classpath depending on what optional packages were found by configure. It does a bunch of other things.

In the end, ptinvoke invokes java with the classpath. https://ptolemy.eecs.berkeley.edu/ptolemyII/ptIIfaq.htm#invoking%20Ptolemy%20II? describes how to invoke Ptolemy. The short answer is that something like the following will invoke the model with all the jar files in $PTII/lib

```
java -classpath $PTII:${PTII}/lib/\*   ptolemy.vergil.VergilApplication ~/Downloads/AprilTagImageAnnotate.xml
```

No matter how you start up capecode and vergil, there should be just one java process.

Under Mac OS X, one can use the lldb debugger to attach to the process. Get the pid using ps and then run "lldb -p pid". gdb does not work very well on the Mac, one has to mess with their rootless crap $^{H+P}$ system. Under Linux, gdp -p pid would probably work.

I also use jvisualvm to attach to processes. jvisualvm is part of the JVM and provides a display of what threads are active etc.

To get a stack trace, one can also use "kill -3 pid" on the java process pid

As I can't reproduce the bug anymore I think the thing to do is to see if Edward can reproduce it and go on from there.

#5 - 06/18/2017 03:24 PM - Anonymous

On 6/18/17 1:40 PM, Edward A. Lee wrote:

> I just got a model with GDP subcribe to hang on wrapup.
> I have it sitting there. What diagnostics should I run?
>
> Edward

It was pretty hard for me to get a stack trace, but you could try a few things:

https://gdp.cs.berkeley.edu/redmine/issues/107 says:

> Under Mac OS X, one can use the lldb debugger to attach to the process. Get the pid using ps and then run "lldb -p pid". gdb does not work very well on the Mac, one has to mess with their rootless crap HHH^ system. Under Linux, gdp -p pid would probably work.
>
> I also use jvisualvm to attach to processes. jvisualvm is part of the JVM and provides a display of what threads are active etc.
>
> To get a stack trace, one can also use "kill -3 pid" on the java process pid

See https://gdp.cs.berkeley.edu/redmine/issues/107 for how I used lldb.

I was not able to reproduce this the other day.

Maybe you could post to https://gdp.cs.berkeley.edu/redmine/issues/107

#6 - 06/19/2017 12:11 PM - Nitesh Mor

- Related to Support #109: GDP with accessors (an umbrella item) added

#7 - 06/21/2017 05:12 PM - Eric Allman

- Status changed from In Progress to Resolved

I'm marking this resolved because (a) I think your change addresses the race condition which was the root of the problem, and (b) I have made some mods so that if this happens again there is a larger chance (but not a 100% guarantee) that it will be caught and return an error (see commit:ddb857b7). If your testing verifies that it is working, please close it.

#8 - 06/22/2017 08:50 AM - Anonymous

- File hang2.txt added

It seems like the problem continues.

1. Under Darwin or Ubuntu, update Ptolemy II and start it:

```
cd $PTII
svn update
$PTII/bin/capecode $PTII/ptolemy/actor/lib/jjs/modules/gdp/demo/GDPToWebSocket/*.xml >& /tmp/hang2.txt
```

See the attached hang2.txt file for the output.

1. Run both models. A text display with data from the edu.berkeley.eecs.testlog log should appear.
2. Stop both models.
3. Start both models.
4. One data point appears in the text display. This is unexpected, there should be a new line every second.
5. Stop both models. The TestGDPToWebSocket model will stop, the GDPToWebSocket model hangs.
6. To get a stack trace under Darwin, use lldb:

```
bash-3.2$ lldb -p 11107
(lldb) process attach --pid 11107
Process 11107 stopped
* thread #1, name = 'AppKit Thread', queue = 'com.apple.main-thread', stop reason = signal SIGSTOP
    frame #0: 0x00007fff95b7234a libsystem_kernel.dylib`mach_msg_trap + 10
libsystem_kernel.dylib`mach_msg_trap:
->  0x7fff95b7234a <+10>: retq
    0x7fff95b7234b <+11>: nop

libsystem_kernel.dylib`mach_msg_overwrite_trap:
    0x7fff95b7234c <+0>:  movq   %rcx, %r10
    0x7fff95b7234f <+3>:  movl   $0x1000020, %eax          ; imm = 0x1000020

Executable module set to "/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java".
Architecture set to: x86_64h-apple-macosx.
(lldb) thread list
thread list
Process 11107 stopped
* thread #1: tid = 0x220bb, 0x00007fff95b7234a libsystem_kernel.dylib`mach_msg_trap + 10, name = 'AppKit Threa
d', queue = 'com.apple.main-thread', sto\
p reason = signal SIGSTOP
  thread #2: tid = 0x220bc, 0x00007fff95b79f46 libsystem_kernel.dylib`__semwait_signal + 10
  thread #3: tid = 0x220c0, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #4: tid = 0x220c4, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #5: tid = 0x220c5, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #6: tid = 0x220c6, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #7: tid = 0x220c7, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #8: tid = 0x220c8, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #9: tid = 0x220c9, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #10: tid = 0x220ca, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #11: tid = 0x220cb, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #12: tid = 0x220cc, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #13: tid = 0x220cd, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: Ref
erence Handler'
  thread #14: tid = 0x220ce, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: Fin
alizer'
  thread #15: tid = 0x220d7, 0x00007fff95b72386 libsystem_kernel.dylib`semaphore_wait_trap + 10, name = 'Java:
 Signal Dispatcher'
  thread #16: tid = 0x220d8, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: C2
CompilerThread0'
  thread #17: tid = 0x220d9, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: C2
CompilerThread1'
  thread #18: tid = 0x220da, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: C2
CompilerThread2'
```

```
  thread #19: tid = 0x220db, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: C1
CompilerThread3'
  thread #20: tid = 0x220dc, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: Ser
vice Thread'
  thread #21: tid = 0x220dd, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10
  thread #22: tid = 0x220fd, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: AWT
-Shutdown'
  thread #23: tid = 0x22111, 0x00007fff95b7234a libsystem_kernel.dylib`mach_msg_trap + 10, name = 'com.apple.N
SEventThread'
  thread #24: tid = 0x22113, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: Jav
a2D Queue Flusher'
  thread #25: tid = 0x22117, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: Jav
a2D Disposer'
  thread #26: tid = 0x22118, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: AWT
-EventQueue-0'
  thread #27: tid = 0x221df, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: ver
tx-blocked-thread-checker'
  thread #28: tid = 0x221e2, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'CVDisplay
Link'
  thread #29: tid = 0x221e4, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: Tim
erQueue'
  thread #30: tid = 0x223d8, 0x00007fff95b7ad96 libsystem_kernel.dylib`kevent + 10, name = 'Java: vert.x-event
loop-thread-1'
  thread #31: tid = 0x223d9, 0x00007fff95b7ad96 libsystem_kernel.dylib`kevent + 10, name = 'Java: vert.x-event
loop-thread-0'
  thread #32: tid = 0x223dc, 0x00007fff95b7ad96 libsystem_kernel.dylib`kevent + 10, name = 'Java: vert.x-accep
tor-thread-0'
  thread #33: tid = 0x223ee, 0x00007fff95b7ad96 libsystem_kernel.dylib`kevent + 10, name = 'Java: vert.x-event
loop-thread-3'
  thread #34: tid = 0x223ef, 0x00007fff95b7ad96 libsystem_kernel.dylib`kevent + 10, name = 'Java: vert.x-event
loop-thread-2'
  thread #35: tid = 0x223f1, 0x00007fff95b79bf2 libsystem_kernel.dylib`__psynch_cvwait + 10, name = 'Java: thr
eadDeathWatcher-2-1'
  thread #36: tid = 0x22414, 0x00007fff95b79c22 libsystem_kernel.dylib`__psynch_mutexwait + 10
  thread #37: tid = 0x22415, 0x00007fff95b79f46 libsystem_kernel.dylib`__semwait_signal + 10
  thread #38: tid = 0x2255f, 0x00007fff95b79c22 libsystem_kernel.dylib`__psynch_mutexwait + 10, name = 'Java:
GDPToWebSocket'
  thread #39: tid = 0x227dd, 0x00007fff95b7a44e libsystem_kernel.dylib`__workq_kernreturn + 10
(lldb) thread select 38
thread select 38
(lldb) ^[[1G^[[J* thread #38, name = 'Java: GDPToWebSocket'
    frame #0: 0x00007fff95b79c22 libsystem_kernel.dylib`__psynch_mutexwait + 10
libsystem_kernel.dylib`__psynch_mutexwait:
->  0x7fff95b79c22 <+10>: jae    0x7fff95b79c2c            ; <+20>
    0x7fff95b79c24 <+12>: movq   %rax, %rdi
    0x7fff95b79c27 <+15>: jmp    0x7fff95b72caf            ; cerror_nocancel
    0x7fff95b79c2c <+20>: retq
^[[1G^[[J(lldb)  ^[[8G

(lldb) ^[[1G^[[J* thread #38, name = 'Java: GDPToWebSocket'
    frame #0: 0x00007fff95b79c22 libsystem_kernel.dylib`__psynch_mutexwait + 10
libsystem_kernel.dylib`__psynch_mutexwait:
->  0x7fff95b79c22 <+10>: jae    0x7fff95b79c2c            ; <+20>
    0x7fff95b79c24 <+12>: movq   %rax, %rdi
    0x7fff95b79c27 <+15>: jmp    0x7fff95b72caf            ; cerror_nocancel
    0x7fff95b79c2c <+20>: retq
^[[1G^[[J(lldb)  ^[[8Gbt
bt
* thread #38, name = 'Java: GDPToWebSocket'
  * frame #0: 0x00007fff95b79c22 libsystem_kernel.dylib`__psynch_mutexwait + 10
    frame #1: 0x00007fff95c64e6e libsystem_pthread.dylib`_pthread_mutex_lock_wait + 100
    frame #2: 0x0000000156763fc5 jna8738236484031965205.tmp`_ep_thr_mutex_lock(mtx=0x00007fad915448e0, file="g
dp_gcl_ops.c", line=<unavailable>, name=<unavailable>) at ep_thr.c:403 [opt]
    frame #3: 0x0000000156750e5e jna8738236484031965205.tmp`_gdp_gcl_lock_trace(gcl=0x00007fad915448e0, file=<
unavailable>, line=<unavailable>, id=<unavailable>) at gdp_gcl_cache.c:496 [opt]
    frame #4: 0x0000000156752b6c jna8738236484031965205.tmp`_gdp_gcl_close(gcl=0x00007fad915448e0, chan=0x0000
7fad913c3950, reqflags=0) at gdp_gcl_ops.c:520 [opt]
    frame #5: 0x000000015674c1b6 jna8738236484031965205.tmp`gdp_gcl_close(gcl=0x00007fad915448e0) at gdp_api.c
:488 [opt]
    frame #6: 0x0000000107211514 jna612451396793971134.tmp`ffi_call_unix64 + 76
    frame #7: 0x000000010721142d jna612451396793971134.tmp`ffi_call + 781
    frame #8: 0x0000000107208565 jna612451396793971134.tmp`___lldb_unnamed_symbol9$$jna612451396793971134.tmp
+ 2181
    frame #9: 0x00000001080989d4
    frame #10: 0x0000000109944d64
```

```
    frame #11: 0x0000000108089114
    frame #12: 0x00000001080892bd
    frame #13: 0x00000001080892bd
    frame #14: 0x00000001080892bd
    frame #15: 0x0000000109372e5c
    frame #16: 0x00000001098fe8d0
    frame #17: 0x00000001099a031c
    frame #18: 0x00000001089ebfd4
    frame #19: 0x000000010996bdcc
(lldb)
```

#9 - 06/25/2017 09:38 AM - Eric Allman

- Status changed from Resolved to Feedback

Christopher, this bug was originally about AprilTagImageAnnotate.xml, but your latest update refers to GDPToWebSocket, which is the model noted in Bug #112. Is this actually a new problem? I'm not seeing the connection.

Regarding the GDPToWebSocket problem in your latest posting, I have to apologize: Nitesh and I had been working in our own sandbox including the new log server code. I have now updated the log server you are using, and the problem seems much better, albeit not 100% fixed. At this point it looks like this is mostly (but perhaps not completely) working. If I start up the server model I can then arbitrarily start and stop the client model with no problems. If I start and stop both models it also seems to work. However, in some (but seemingly not all) cases, if I stop and restart the server model with the client still running then it doesn't seem to resynchronize (although there no obvious hangs or crashes). However, restarting the client side then puts it back to working state.

I'm not sure what the status of the original problem is. Attempts to run it now reports

```
Error: Could not find or load main class org.terraswarm.accessor.demo.AprilTagImageAnnotate.AprilTagImageAnnot
ate.xml
```

(but that file exists and is readable).

Given the confusion, I'm setting the status back to Feedback for now.

#10 - 06/25/2017 08:11 PM - Anonymous

- Status changed from Feedback to Closed

Thanks, this seems to be fixed so I'm closing it.

I think the error message:

```
Error: Could not find or load main class org.terraswarm.accessor.demo.AprilTagImageAnnotate.AprilTagImageAnnot
ate.xml
```

Would happen if ptinvoke was run like so:

```
bash-3.2$ $PTII/bin/ptinvoke $PTII/org/terraswarm/accessor/demo/AprilTagImageAnnotate/AprilTagImageAnnotate.xm
l
Error: Could not find or load main class .Users.cxh.src.ptII11.0.devel.org.terraswarm.accessor.demo.AprilTagIm
ageAnnotate.AprilTagImageAnnotate.xml
bash-3.2$
```

I apologize, my notes about state that $PTII/bin/ptinvoke should be used, when in fact $PTII/bin/capecode should be used.

This issue was happening with all Accessors that used the GDP. I started using a different model because the new model was more likely to work. I'm sorry about any confusion, I know that changing the setup in the middle of an issue is not a big help.

Anyway, this seems to be working so we are done now.

Files

AprilTagImageAnnotate.xml	60.9 KB	06/12/2017	Anonymous
hang2.txt	43.2 KB	06/22/2017	Anonymous