
Subject: Re: Asynchronous IDL_IDLBridge causing memory leak
Posted by [Seth Johnson](#) on Fri, 27 Aug 2010 18:39:08 GMT
[View Forum Message](#) <> [Reply to Message](#)

Sorry, I realized there was a mistake in the second example, it should be:

```
oBridge=OBJARR(5)
FOR chain=0,4 DO BEGIN oBridge[chain]=Obj_New('IDL_IDLBridge')

FOR i=0,999 DO BEGIN
  FOR chain=0,4 do BEGIN
    a=bindgen(1E4,1E3)
    oBridge[chain]->SetVar,'a',a
    oBridge[chain]->Execute,'a=a+a',/NOWAIT
  ENDFOR

  FOR chain=0,4 DO WHILE oBridge[chain]->Status() NE 0 DO wait,0.0001
ENDFOR
OBJ_DESTROY,oBridge
```

I do not destroy the objects until the very end as there are parameters and routines that need to be loaded into each IDL_IDLBridge for various computations in addition to parameters that change with every iteration. Destroying and recreating would be a rather large boon to processing time while the initial problem caused by asynchronous operation still remains.

Subject: Re: Asynchronous IDL_IDLBridge causing memory leak
Posted by [Juggernaut](#) on Mon, 30 Aug 2010 12:35:20 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Aug 27, 2:39 pm, Seth Johnson <seth.spjoh...@gmail.com> wrote:
> Sorry, I realized there was a mistake in the second example, it should
> be:
>
> oBridge=OBJARR(5)
> FOR chain=0,4 DO BEGIN oBridge[chain]=Obj_New('IDL_IDLBridge')
>
> FOR i=0,999 DO BEGIN
> FOR chain=0,4 do BEGIN
> a=bindgen(1E4,1E3)
> oBridge[chain]->SetVar,'a',a
> oBridge[chain]->Execute,'a=a+a',/NOWAIT
> ENDFOR
>
> FOR chain=0,4 DO WHILE oBridge[chain]->Status() NE 0 DO wait,0.0001

> ENDFOR
> OBJ_DESTROY,oBridge
>
> I do not destroy the objects until the very end as there are
> parameters and routines that need to be loaded into each IDL_IDLBridge
> for various computations in addition to parameters that change with
> every iteration. Destroying and recreating would be a rather large
> boon to processing time while the initial problem caused by
> asynchronous operation still remains.

I've noticed that leak in 6.3 but not in 7.0+. Which version are you running?

Subject: Re: Asynchronous IDL_IDLBridge causing memory leak
Posted by [Seth Johnson](#) on Mon, 30 Aug 2010 14:02:01 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Aug 30, 8:35 am, Bennett <juggernaut...@gmail.com> wrote:

> On Aug 27, 2:39 pm, Seth Johnson <seth.spjoh...@gmail.com> wrote:
>
>
>
>> Sorry, I realized there was a mistake in the second example, it should
>> be:
>
>> oBridge=OBJARR(5)
>> FOR chain=0,4 DO BEGIN oBridge[chain]=Obj_New('IDL_IDLBridge')
>
>> FOR i=0,999 DO BEGIN
>> FOR chain=0,4 do BEGIN
>> a=bindgen(1E4,1E3)
>> oBridge[chain]->SetVar,'a',a
>> oBridge[chain]->Execute,'a=a+a',/NOWAIT
>> ENDFOR
>
>> FOR chain=0,4 DO WHILE oBridge[chain]->Status() NE 0 DO wait,0.0001
>> ENDFOR
>> OBJ_DESTROY,oBridge
>
>> I do not destroy the objects until the very end as there are
>> parameters and routines that need to be loaded into each IDL_IDLBridge
>> for various computations in addition to parameters that change with
>> every iteration. Destroying and recreating would be a rather large
>> boon to processing time while the initial problem caused by
>> asynchronous operation still remains.
>
> I've noticed that leak in 6.3 but not in 7.0+. Which version are you

> running?

Strange, I have tested this on IDL versions 7.0 and 7.1, both of which produce the leak. Could the cause perhaps lie in the setup or one of the required packages? I have noticed while testing on different machines that 7.0 and 7.1 use different versions of the shared library libstdc++.so.

Subject: Re: Asynchronous IDL_IDLBridge causing memory leak
Posted by [Seth Johnson](#) on Tue, 31 Aug 2010 15:31:59 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Aug 30, 10:02 am, Seth Johnson <seth.spjoh...@gmail.com> wrote:
> On Aug 30, 8:35 am, Bennett <juggernaut...@gmail.com> wrote:
>
>
>
>> On Aug 27, 2:39 pm, Seth Johnson <seth.spjoh...@gmail.com> wrote:
>
>>> Sorry, I realized there was a mistake in the second example, it should
>>> be:
>
>>> oBridge=OBJARR(5)
>>> FOR chain=0,4 DO BEGIN oBridge[chain]=Obj_New('IDL_IDLBridge')
>
>>> FOR i=0,999 DO BEGIN
>>> FOR chain=0,4 DO BEGIN
>>> a=bindgen(1E4,1E3)
>>> oBridge[chain]->SetVar,'a',a
>>> oBridge[chain]->Execute,'a=a+a',/NOWAIT
>>> ENDFOR
>
>>> FOR chain=0,4 DO WHILE oBridge[chain]->Status() NE 0 DO wait,0.0001
>>> ENDFOR
>>> OBJ_DESTROY,oBridge
>
>>> I do not destroy the objects until the very end as there are
>>> parameters and routines that need to be loaded into each IDL_IDLBridge
>>> for various computations in addition to parameters that change with
>>> every iteration. Destroying and recreating would be a rather large
>>> boon to processing time while the initial problem caused by
>>> asynchronous operation still remains.
>
>> I've noticed that leak in 6.3 but not in 7.0+. Which version are you
>> running?
>
> Strange, I have tested this on IDL versions 7.0 and 7.1, both of which

> produce the leak. Could the cause perhaps lie in the setup or one of
> the required packages? I have noticed while testing on different
> machines that 7.0 and 7.1 use different versions of the shared library
> libstdc++.so.

It is not the most elegant of solutions, but I have found a temporary work around for the memory leak. Rather than calling the asynchronous processes from the main routine, I create a single child process that then creates its own children and performs the asynchronous calls similar to:

```
oBridge=Obj_New('IDL_IDLBridge')
oBridge->SetVar,'a',a
oBridge->Execute,"oBridge=Obj_New('IDL_IDLBridge')"
oBridge->Execute,"oBridge->SetVar,'a',a"
FOR i=0,999 DO BEGIN
    tmp=memory()
    oBridge->Execute,"oBridge->Execute,'a=a+a',/NOWAIT"
    print,memory(/high)
    WHILE oBridge->GetVar('oBridge->Status()') NE 0 DO wait,0.0001
ENDFOR
```

The child process (and its children) do not appear to leak memory as the parent call does. I find it rather peculiar that this method works, even after loading the IDL startup file into the child processes.

Subject: Re: Asynchronous IDL_IDLBridge causing memory leak
Posted by [Russell Ryan](#) on Fri, 18 Jan 2013 17:56:19 GMT
[View Forum Message](#) <> [Reply to Message](#)

Forgive me for waking the dead and releasing the zombie posts. But I've noticed a similar behavior on IDL 8.1. From a little testing, I've found that if I put calls to systime() and memory() on either side of the Bridge->Execute,/nowait call I can see (1) the time to start an asynchronous call and (2) it's memory usage increase with time. I'll try implementing this ugly-looking work around and see what ITT has to say about it?

-Russell

On Tuesday, August 31, 2010 11:31:59 AM UTC-4, Seth Johnson wrote:
> On Aug 30, 10:02 am, Seth Johnson <seth.spjoh...@gmail.com> wrote:
>> On Aug 30, 8:35 am, Bennett <juggernaut...@gmail.com> wrote:
>>
>>
>>

```

>>> On Aug 27, 2:39 pm, Seth Johnson <seth.spjoh...@gmail.com> wrote:
>>
>>>> Sorry, I realized there was a mistake in the second example, it should
>>>> be:
>>
>>>> oBridge=OBJARR(5)
>>>> FOR chain=0,4 DO BEGIN oBridge[chain]=Obj_New('IDL_IDLBridge')
>>
>>>> FOR i=0,999 DO BEGIN
>>>>   FOR chain=0,4 do BEGIN
>>>>     a=bindgen(1E4,1E3)
>>>>     oBridge[chain]->SetVar,'a',a
>>>>     oBridge[chain]->Execute,'a=a+a',/NOWAIT
>>>>   ENDFOR
>>
>>>>   FOR chain=0,4 DO WHILE oBridge[chain]->Status() NE 0 DO wait,0.0001
>>>> ENDFOR
>>>> OBJ_DESTROY,oBridge
>>
>>>> I do not destroy the objects until the very end as there are
>>>> parameters and routines that need to be loaded into each IDL_IDLBridge
>>>> for various computations in addition to parameters that change with
>>>> every iteration. Destroying and recreating would be a rather large
>>>> boon to processing time while the initial problem caused by
>>>> asynchronous operation still remains.
>>
>>> I've noticed that leak in 6.3 but not in 7.0+. Which version are you
>>> running?
>>
>> Strange, I have tested this on IDL versions 7.0 and 7.1, both of which
>> produce the leak. Could the cause perhaps lie in the setup or one of
>> the required packages? I have noticed while testing on different
>> machines that 7.0 and 7.1 use different versions of the shared library
>> libstdc++.so.
>
> It is not the most elegant of solutions, but I have found a temporary
> work around for the memory leak. Rather than calling the asynchronous
> processes from the main routine, I create a single child process that
> then creates its own children and performs the asynchronous calls
> similar to:
>
> oBridge=Obj_New('IDL_IDLBridge')
> oBridge->SetVar,'a',a
> oBridge->Execute,"oBridge=Obj_New('IDL_IDLBridge')"
> oBridge->Execute,"oBridge->SetVar,'a',a"
> FOR i=0,999 DO BEGIN
>   tmp=memory()
>   oBridge->Execute,"oBridge->Execute,'a=a+a',/NOWAIT"

```

```
> print,memory(/high)
> WHILE oBridge->GetVar('oBridge->Status()') NE 0 DO wait,0.0001
> ENDFOR
>
> The child process (and its children) do not appear to leak memory as
> the parent call does. I find it rather peculiar that this method
> works, even after loading the IDL startup file into the child
> processes.
```

Subject: Re: Asynchronous IDL_IDLBridge causing memory leak
Posted by [Russell Ryan](#) on Sat, 26 Jan 2013 18:27:16 GMT

[View Forum Message](#) <> [Reply to Message](#)

I've been emailing folks at Exelis about this. They've now issued a formal bug report to the engineers. I'll repost if I learn of any answers...

R

On Friday, January 18, 2013 12:56:19 PM UTC-5, rr...@stsci.edu wrote:

```
> Forgive me for waking the dead and releasing the zombie posts. But I've noticed a similar
behavior on IDL 8.1. From a little testing, I've found that if I put calls to systime() and memory() on
either side of the Bridge->Execute,/nowait call I can see (1) the time to start an asynchronous call
and (2) it's memory usage increase with time. I'll try implementing this ugly-looking work around
and see what ITT has to say about it?
```

```
>
>
>
> -Russell
```

```
>
>
>
>
>
>
>
>
> On Tuesday, August 31, 2010 11:31:59 AM UTC-4, Seth Johnson wrote:
```

```
>
>> On Aug 30, 10:02 am, Seth Johnson <seth.spjoh...@gmail.com> wrote:
```

```
>
>>> On Aug 30, 8:35 am, Bennett <juggernaut...@gmail.com> wrote:
```

```
>
>>>
```

```
>
>>>
```

```
>
>>>
>
```

```

>>>> On Aug 27, 2:39 pm, Seth Johnson <seth.spjoh...@gmail.com> wrote:
>
>>>>
>
>>>> > Sorry, I realized there was a mistake in the second example, it should
>
>>>> > be:
>
>>>>
>
>>>> > oBridge=OBJARR(5)
>
>>>> > FOR chain=0,4 DO BEGIN oBridge[chain]=Obj_New('IDL_IDLBridge')
>
>>>>
>
>>>> > FOR i=0,999 DO BEGIN
>
>>>> >   FOR chain=0,4 do BEGIN
>
>>>> >     a=bindgen(1E4,1E3)
>
>>>> >     oBridge[chain]->SetVar,'a',a
>
>>>> >     oBridge[chain]->Execute,'a=a+a',/NOWAIT
>
>>>> >   ENDFOR
>
>>>>
>
>>>> >   FOR chain=0,4 DO WHILE oBridge[chain]->Status() NE 0 DO wait,0.0001
>
>>>> > ENDFOR
>
>>>> > OBJ_DESTROY,oBridge
>
>>>>
>
>>>> > I do not destroy the objects until the very end as there are
>
>>>> > parameters and routines that need to be loaded into each IDL_IDLBridge
>
>>>> > for various computations in addition to parameters that change with
>
>>>> > every iteration. Destroying and recreating would be a rather large
>
>>>> > boon to processing time while the initial problem caused by
>

```

```

>>>> > asynchronous operation still remains.
>
>>>
>
>>>> I've noticed that leak in 6.3 but not in 7.0+. Which version are you
>
>>>> running?
>
>>>
>
>>> Strange, I have tested this on IDL versions 7.0 and 7.1, both of which
>
>>> produce the leak. Could the cause perhaps lie in the setup or one of
>
>>> the required packages? I have noticed while testing on different
>
>>> machines that 7.0 and 7.1 use different versions of the shared library
>
>>> libstdc++.so.
>
>>
>
>> It is not the most elegant of solutions, but I have found a temporary
>
>> work around for the memory leak. Rather than calling the asynchronous
>
>> processes from the main routine, I create a single child process that
>
>> then creates its own children and performs the asynchronous calls
>
>> similar to:
>
>>
>
>> oBridge=Obj_New('IDL_IDLBridge')
>
>> oBridge->SetVar,'a',a
>
>> oBridge->Execute,"oBridge=Obj_New('IDL_IDLBridge')"
>
>> oBridge->Execute,"oBridge->SetVar,'a',a"
>
>> FOR i=0,999 DO BEGIN
>
>>     tmp=memory()
>
>>     oBridge->Execute,"oBridge->Execute,'a=a+a',/NOWAIT"
>

```



```
>> print,memory(/high)
>
>> WHILE oBridge->GetVar('oBridge->Status()') NE 0 DO wait,0.0001
>
>> ENDFOR
>
>>
>
>> The child process (and its children) do not appear to leak memory as
>
>> the parent call does. I find it rather peculiar that this method
>
>> works, even after loading the IDL startup file into the child
>
>> processes.
```

Subject: Re: Asynchronous IDL_IDLBridge causing memory leak
Posted by [chris_torrence@NOSPAM](#) on Wed, 31 Jul 2013 19:21:02 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi all,

Just FYI, the memory leak with the IDL_IDLBridge (bug 43494) has been fixed for IDL 8.3. IDL 8.3 should be out sometime later in the Fall.

Cheers,
Chris
ExelisVIS

Subject: Re: Asynchronous IDL_IDLBridge causing memory leak
Posted by [Haje Korth](#) on Thu, 01 Aug 2013 12:45:21 GMT
[View Forum Message](#) <> [Reply to Message](#)

Chris,
this is really good news for one of my projects!

Thanks,
Haje

On Wednesday, July 31, 2013 3:21:02 PM UTC-4, Chris Torrence wrote:

```
> Hi all,
>
>
>
> Just FYI, the memory leak with the IDL_IDLBridge (bug 43494) has been fixed for IDL 8.3. IDL
```

8.3 should be out sometime later in the Fall.

>

>

>

> Cheers,

>

> Chris

>

> ExelisVIS
