Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Russell Ryan on Wed, 10 Jul 2013 17:26:09 GMT
View Forum Message <> Reply to Message

On Wednesday, July 10, 2013 12:55:12 PM UTC-4, nata wrote:
> Hi Russell,
>
>
>
> The memory leak you are talking about is it related to the variables defined at the end of the execution?
>
>
>
> If you execute something, after getting the result, the variables are not destroyed and they keep using memory. To prevent that I always execute the following commands:
>
>
>
> all_var='tmp_var'
>
> IDLBridge->Execute, all_var+'=ROUTINE_INFO("$MAIN$",/VARIABLES)'
>
> all_var=self->IDL_IDLBridge::GetVar(all_var)
>
>
>
> command='DELVAR, '+STRJOIN(all_var,', ')
>
> IDLBridge->Execute, command
>
>
>
> Are you talking about something different?
>
> I guess that using "help, /memory" I would be able to track the usage of memory and see if the bridge definitely has a bug.
>
>
>
> nata

@Chris. Yes, by "triggering", I mean a call to the EXECUTE method.

@Nata. No, that doesn't fix the problem. Here is a simple script to try. You can see where I added your suggestion to the test code that Exelis asked me to try. You can see that the memory usage continues to increase with time. Again, it's a small increase --- but the point is it should

read 0. And if you run this long enough, that small bit *WILL* cause problems, trust me. I lost a very long time "debugging" my "bridged" code before I just emailed Exelis. They confirmed it was a problem, and I "unbridged" my code. I thought to run the bridged version for a long time (but not long enough for the leak to be a problem), then save the results, end IDL, restart IDL, start bridged code with old outputs. Heck, you could put this in a csh script or something and so it's seamless to me. But, I hated the idea of this, because gosh darn it, it should just work! This "feature" was added in IDL 6, they're on 8, and it's expensive software that shouldn't need a shell-script to "patch" something. I have since rewritten that bit of code in Fortran, which is easily multithreaded and (if necessary) call it from IDL using call_external.

```
pro b43494
  mbegin=(memory())[0]
  for i=0,100 do begin
     m0 = (memory())[0]
     o = obj_new('idl_idlbridge')
     o->Execute, 'x = 1'        ;, /nowait

     ;Nata's additions----
     all_var='tmp_var'
     o->Execute, all_var+'=ROUTINE_INFO("$MAIN$",/VARIABLES)'
     all_var=o->GetVar(all_var)

     command='DELVAR, '+STRJOIN(all_var,', ')
     o->Execute, command
     ;End of Nata's additions-----

     obj_destroy, o
     m1 = (memory())[0]
     if (i gt 0) then begin
        print
        print,'iteration: ',i
        print,'Memory lost in this step thru loop: ', m1 - m0
        print,'Total memory lost since beginning:  ',m1-mbegin
        print
     endif
  endfor

end
```

-Russell

PS: This is without using the nowait flag to the EXECUTE method, which frankly is stupid. I mean, why would you do this effort of IDL_IDLBridge if not to *USE* the nowait feature. So, if you uncomment that bit, you will find the leaks are *WORSE*.

PPS: If I've done something wrong, please let me know. But if it's all okay, then I strongly discourage anyone from doing anything with IDL_IDLBridge

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by natha on Wed, 10 Jul 2013 17:40:16 GMT
View Forum Message <> Reply to Message

Wow, this is a MAJOR bug! now I see what you mean....

I've spent a lot time developing a library to allow parallel computing under IDL and now I realize that there is a bug that causes memory problems. f**k !
Thank you for pointing it out. What is the bug report error and how can we know if in a future version this will be solved?

Thank you Russell to share this,
nata

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Russell Ryan on Wed, 10 Jul 2013 17:53:28 GMT
View Forum Message <> Reply to Message

On Wednesday, July 10, 2013 1:40:16 PM UTC-4, nata wrote:
> Wow, this is a MAJOR bug! now I see what you mean....
>
>
>
> I've spent a lot time developing a library to allow parallel computing under IDL and now I realize that there is a bug that causes memory problems. f**k !
>
> Thank you for pointing it out. What is the bug report error and how can we know if in a future version this will be solved?
>
>
>
> Thank you Russell to share this,
>
> nata

@Nata,

Trust me, I feel your pain. I had written two large codes for analyzing galaxies, one of the codes was an advanced MCMC sampler which utilized the bridges to distribute the processing. This code was very general and (in part) modeled after MPFIT, in terms of usage, flexibility, and organizational philosophy. The unbridged version is fine, but it's just very slow (what computers do not have multiple processors, we should be using them). This problem is perfectly parallelizable, since it has running several simple MCMC chains w/o communicating between the chains, and at some prescribed time having them exchange information. The plan was, run each chain on a separate bridge for some time. Then, bring them together for the information

exchange. Then repeat for some amount of time. (It was the Evolutionary MCMC and parallel tempering sampler). Since this was one of primary usages of the bridges, the scripting solution of exiting and restarting IDL is particularly clunky for distributing this MCMC sampler.

Again, the thing that kills me the most, is when people use the bridges on shared computers --- despite numerous warnings from me. I say, spread the word. Exelis is aware of the issue, but my gut tells me that the more people who complain the faster it gets solved. Outside of my work and your work, I know of 4 codes used in astronomy which suffer from this problem but the authors and users are not aware (or don't care).

R

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Russell Ryan on Wed, 10 Jul 2013 18:01:22 GMT
View Forum Message <> Reply to Message

BTW, I meant to say. I didn't find this problem. I (re-)discovered it for myself; it had been puzzled over on this blog back in 2010. When I came aware of the problem, I posted to that (then dead) blog

https://groups.google.com/forum/#!searchin/comp.lang.idl-pvw
ave/IDL_IDLBridge$20memory/comp.lang.idl-pvwave/5ehCtUB-CCY/ 2Ytp2ulnFAYJ

From this you can see two things:

(1) it seems to be independent of the IDL version. Again, I tested both Mac OSX and Linux, but Exelis tells me that Windows is unaffected (to some degree or another). I'm still running 8.1, but I've poured over "new features" of later releases and never have I seen a mention of the IDL_IDLBridge. Also, I have not tried any of the other IDL_XBridges.

(2) I posted in Jan, which is about the time I pinged Exelis. So, it's been ~6 months (perhaps more if I misremembered something) and still no answer.

On Wednesday, July 10, 2013 1:53:28 PM UTC-4, rr...@stsci.edu wrote:
> On Wednesday, July 10, 2013 1:40:16 PM UTC-4, nata wrote:
>
>> Wow, this is a MAJOR bug! now I see what you mean....
>
>>
>
>>
>
>>

>
>> I've spent a lot time developing a library to allow parallel computing under IDL and now I realize that there is a bug that causes memory problems. f**k !
>
>>
>
>> Thank you for pointing it out. What is the bug report error and how can we know if in a future version this will be solved?
>
>>
>
>>
>
>>
>
>> Thank you Russell to share this,
>
>>
>
>> nata
>
>
>
> @Nata,
>
>
>
> Trust me, I feel your pain. I had written two large codes for analyzing galaxies, one of the codes was an advanced MCMC sampler which utilized the bridges to distribute the processing. This code was very general and (in part) modeled after MPFIT, in terms of usage, flexibility, and organizational philosophy. The unbridged version is fine, but it's just very slow (what computers do not have multiple processors, we should be using them). This problem is perfectly parallelizable, since it has running several simple MCMC chains w/o communicating between the chains, and at some prescribed time having them exchange information. The plan was, run each chain on a separate bridge for some time. Then, bring them together for the information exchange. Then repeat for some amount of time. (It was the Evolutionary MCMC and parallel tempering sampler). Since this was one of primary usages of the bridges, the scripting solution of exiting and restarting IDL is particularly clunky for distributing this MCMC sampler.
>
>
>
> Again, the thing that kills me the most, is when people use the bridges on shared computers --- despite numerous warnings from me. I say, spread the word. Exelis is aware of the issue, but my gut tells me that the more people who complain the faster it gets solved. Outside of my work and your work, I know of 4 codes used in astronomy which suffer from this problem but the authors and users are not aware (or don't care).
>
>

>
> R

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Chip Helms on Wed, 10 Jul 2013 18:05:25 GMT
View Forum Message <> Reply to Message

I'm not actually seeing any memory leak. The memory lost during a single iteration remains zero at least through iteration 29. (The total memory lost stays constant...my interpretation is the value is the memory taken by the bridge object)

The IDL docs on OBJ_DESTROY suggest using the cleanup method is better in the instance of recursive object definitions (and I've no clue if IDL_IDLBridge generates objects within itself) so I thought to try "o->Cleanup" (despite the docs saying that "o->cleanup" is the same as "OBJ_DESTROY, o").

Also, in this example a new object is created and destroyed each iteration. If I rewrote this to mirror my own implementation, it would create the bridge once and then each iteration it would use setvar, execute, then getvar (or in this case just use execute each iteration). This also has the benefit of avoiding the overhead of creating and destroying the objects, but I'm unsure as to what impact it has on any memory leak.

Regardless, here's my version structure if you're interested:
{ x86_64 linux unix linux 8.2.3 May 2 2013 64 64}

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Russell Ryan on Wed, 10 Jul 2013 18:12:47 GMT
View Forum Message <> Reply to Message

Hi Chip,

That's new. Two quick questions, in my test script, did you use the /nowait feature in the call to EXECUTE method? If not, try that. Again, that's the point of using the IDL_IDLBridge.

Second question, how exactly did you destroy the object if not with obj_destroy? I guess I didn't understand how you used the cleanup method. I never tried any of that, so I'm anxious to see what that does. I doubt this is the answer, because you can move the object creation and destruction *OUTSIDE* of my loop, and the memory problems do not go away.

Like I said, I don't have Linux. I sent that script to a friend who does use Linux, and he told me it did have problems, but I don't know which linux, which IDL, or which options he did/didn't use.

R

On Wednesday, July 10, 2013 2:05:25 PM UTC-4, Chip Helms wrote:
> I'm not actually seeing any memory leak. The memory lost during a single iteration remains zero at least through iteration 29. (The total memory lost stays constant...my interpretation is the value is the memory taken by the bridge object)
>
>
>
> The IDL docs on OBJ_DESTROY suggest using the cleanup method is better in the instance of recursive object definitions (and I've no clue if IDL_IDLBridge generates objects within itself) so I thought to try "o->Cleanup" (despite the docs saying that "o->cleanup" is the same as "OBJ_DESTROY, o").
>
>
>
> Also, in this example a new object is created and destroyed each iteration. If I rewrote this to mirror my own implementation, it would create the bridge once and then each iteration it would use setvar, execute, then getvar (or in this case just use execute each iteration). This also has the benefit of avoiding the overhead of creating and destroying the objects, but I'm unsure as to what impact it has on any memory leak.
>
>
>
> Regardless, here's my version structure if you're interested:
>
> { x86_64 linux unix linux 8.2.3 May 2 2013 64 64}

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by David Fanning on Wed, 10 Jul 2013 18:27:21 GMT
View Forum Message <> Reply to Message

Chip Helms writes:

> The IDL docs on OBJ_DESTROY suggest using the cleanup method is better in the instance of recursive object definitions (and I've no clue if IDL_IDLBridge generates objects within itself) so I thought to try "o->Cleanup" (despite the docs saying that "o->cleanup" is the same as "OBJ_DESTROY, o").

This seems unlikely. The CLEANUP routine is a lifecycle method and can't
be called directly. It can only be called from within another CLEANUP
method. Or, at least, that is how it is *suppose* to work. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Phillip Bitzer on Wed, 10 Jul 2013 18:46:24 GMT
View Forum Message <> Reply to Message

Just to throw another data point out there:

IDL> print, !VERSION
{ x86_64 darwin unix Mac OS X 8.2.2 Jan 23 2013 64 64}

I am losing memory at every iteration. Including Nata's code seems to leak even more memory. And setting NOWAIT, well, Russell wasn't kidding when he said it was worse.

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by David Fanning on Wed, 10 Jul 2013 18:48:50 GMT
View Forum Message <> Reply to Message

Phillip Bitzer writes:

> I am losing memory at every iteration. Including Nata's code seems to leak even more memory. And setting NOWAIT, well, Russell wasn't kidding when he said it was worse.

Does someone want to write this up? With code examples?
I'd be happy to publish it on my web page to warn other users. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/

Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Chip Helms on Wed, 10 Jul 2013 18:50:51 GMT
View Forum Message <> Reply to Message

The plot thickens. So I reran the memory leak test after uncommenting the /nowait option and sure enough, the memory leak appeared. Also, I just finished reading in the documentation that you shouldn't be able to call a cleanup method. But oddly enough the following appears to have the same behavior as using obj_destroy:

```
o = obj_new('IDL_IDLBridge')
o->Cleanup
```

Also, I have confirmed that there is no apparent difference between creating and destroying the bridges inside or outside of the loop regardless of whether I used "obj_destroy, o" or "o->Cleanup" (I'm guessing obj_destroy simply calls o->Cleanup)

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by natha on Wed, 10 Jul 2013 19:08:27 GMT
View Forum Message <> Reply to Message

David, here is an example;

```
 obridge=obj_new('idl_idlbridge')

 for i=0,999 do begin
   obridge->execute,'a=1',/nowait
   while obridge->status() ne 0 do wait,0.0001
   print, memory(/high)
 endfor

 obj_destroy, obridge
```

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by David Fanning on Wed, 10 Jul 2013 19:08:50 GMT
View Forum Message <> Reply to Message

Chip Helms writes:

> o = obj_new('IDL_IDLBridge')

> o->Cleanup

This is too friggin' weird!

```
IDL> o = obj_new('IDL_IDLBridge')
IDL> print, obj_valid(o)
  1
IDL> o->Cleanup
IDL> print, obj_valid(o)
  0
```

Not suppose to be possible to do that. :-)

But, I can even do it with my own objects:

```
IDL> obj = cgMap()
IDL> print, obj_valid(obj)
  1
IDL> obj -> Cleanup
IDL> print, obj_valid(obj)
  0
```

What next!?

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Russell Ryan on Wed, 10 Jul 2013 19:11:34 GMT

View Forum Message <> Reply to Message

On Wednesday, July 10, 2013 2:48:50 PM UTC-4, David Fanning wrote:
> Phillip Bitzer writes:
>
>
>
>> I am losing memory at every iteration. Including Nata's code seems to leak even more memory. And setting NOWAIT, well, Russell wasn't kidding when he said it was worse.
>
>

>
> Does someone want to write this up? With code examples?
>
> I'd be happy to publish it on my web page to warn other users. :-)
>
>
>
> Cheers,
>
>
>
> David
>
>
>
>
>
>
>
>
>
> --
>
> David Fanning, Ph.D.
>
> Fanning Software Consulting, Inc.
>
> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
>
> Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Hi David,

I'll try to send you something tonight. I'd love for more people to know of this problem, ideally it would lead to a faster fix, but I'd be happy to know people just abandon it until it's resolved.

R

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by David Fanning on Wed, 10 Jul 2013 19:13:01 GMT
View Forum Message <> Reply to Message

nata writes:

> David, here is an example;
>
> obridge=obj_new('idl_idlbridge')
>

```
>    for i=0,999 do begin
>      obridge->execute,'a=1',/nowait
>      while obridge->status() ne 0 do wait,0.0001
>      print, memory(/high)
>    endfor
>
>    obj_destroy, obridge
```

Humm. Well, I can confirm then that there doesn't appear to be anything amiss in IDL 8.2.3 on Windows. No leakage here.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Chip Helms on Wed, 10 Jul 2013 19:24:49 GMT
View Forum Message <> Reply to Message

Alright, so I tried looking at a couple things with the following code:

```
pro memissue
  mbegin=(memory())[0]
   ; create the object outside the loop (no difference memory-wise...just faster to run)
  o = obj_new('IDL_IDLBridge',output='out_memissue.asc')
   ; write bridge memory
  o->execute, 'print, "ini", (memory())[0]'
  for i=0,100 do begin
     m0 = (memory())[0]
     o->Execute, 'x = 1'        , /nowait

      ;wait for execution to complete (I doubt this is needed for 'x=1'...but just incase)
     while (o->status() ne 0) do wait, 0.1
     ;Nata's additions----
     all_var='tmp_var'
     o->Execute, all_var+'=ROUTINE_INFO("$MAIN$",/VARIABLES)'
     all_var=o->GetVar(all_var)
```

```
    command='DELVAR, '+STRJOIN(all_var,', ')
    o->Execute, command
    ;End of Nata's additions-----

    m1 = (memory())[0]
    if (i gt 0) then begin
      print
      print,'iteration: ',i
      print,'Memory lost in this step thru loop: ', m1 - m0
      print,'Total memory lost since beginning:  ',m1-mbegin
      print
    endif
     ; write iteration memory
    o->execute, 'print, '+strtrim(i,2)+', (memory())[0]'
  endfor
   ; write final memory
  o->execute, 'print, "fin", (memory())[0]'
   ;wait for any remaining execution to complete (really shouldn't be needed)
  while (o->status() ne 0) do wait, 0.5
   ; destroy object
  obj_destroy, o
;  o->Cleanup
  mfinal=(memory())[0]
  print, 'Total run memory difference:', mfinal-mbegin
end
```

From what I've seen, nothing is gained by deleting the variables within the bridge (the bridge output file has 768987 for (memory())[0] at each iteration regardless of variable being deleted or not, which doesn't seem quite right to me). As far as the difference in memory between start and end of the parent: 210245 regardless of if I include "o->Execute, command" or not. If I comment out "/nowait" I find that every iteration has a 0 for memory lost and the total run difference is 2421. Without "/nowait" the bridge memory returns 769163 every iteration. So to my untrained eye, it seems like execute with "/nowait" produces something (maybe some sort of placeholder?) that isn't taken care of before moving on, but I have to admit that's just speculation on my part. On the upside, it seems to be restricted to only execute calls that are asynchronous (at least on my machine).

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Chip Helms on Wed, 10 Jul 2013 19:54:50 GMT

View Forum Message <> Reply to Message

Poking around a little harder I've come up with something else. Including "/nowait" results in IDL compiling a number of procedures it wouldn't otherwise compile:

Procedures:

ADDMANAGEDWIDGET
IDL_IDLBRIDGE::EXECUTETIMER
IDL_IDLBRIDGETIMER_EVENT
VALIDATEMANAGEDWIDGETS
XMANAGER
XMANAGERPRINTERROR
XMANAGER_EVLOOP_FAKE_MODAL
XMANAGER_EVLOOP_REAL_MODAL
XMANAGER_EVLOOP_STANDARD
XUNREGISTER

Functions:
LOOKUPMANAGEDWIDGET

None of these are compiled when running the same code without "/nowait"
Here's the code I ran:

```
pro memissue2, nowait=nowait
  m0 = (memory())[0]
  o = obj_new('IDL_IDLBridge')
  o->execute, 'print, "hello"', nowait=nowait
  while (o->status() ne 0) do wait, 0.01
  obj_destroy, o
  help, /full
  m1 = (memory())[0]
  print, m1-m0
end
```

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Mark Piper on Thu, 11 Jul 2013 17:32:19 GMT
View Forum Message <> Reply to Message

I agree that this is a debilitating issue with the IDL_IDLBridge. However, Russell is currently the only user listed on the report (IDL-68399, for reference). Please contact Tech Support (support@exelisvis.com) and ask to be added to the report. With more evidence, I can push for this to be fixed sooner.

mp

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Heinz Stege on Mon, 22 Jul 2013 12:02:38 GMT
View Forum Message <> Reply to Message

On Wed, 10 Jul 2013 13:08:50 -0600, David Fanning wrote:

> Chip Helms writes:
>
>> o = obj_new('IDL_IDLBridge')
>> o->Cleanup
>
> This is too friggin' weird!
>
> IDL> o = obj_new('IDL_IDLBridge')
> IDL> print, obj_valid(o)
> 1
> IDL> o->Cleanup
> IDL> print, obj_valid(o)
> 0
>
> Not suppose to be possible to do that. :-)
>
> But, I can even do it with my own objects:
>
> IDL> obj = cgMap()
> IDL> print, obj_valid(obj)
> 1
> IDL> obj -> Cleanup
> IDL> print, obj_valid(obj)
> 0
>
> What next!?
>
> Cheers,
>
> David

I know, I'm a little bit late. I just stumbled over the related part of the documentation: "To destroy objects, instead of calling OBJ_DESTROY, obj, you can now do: myContainer.Cleanup"

I found this under the headline "Object Syntax" in the section "What's New in IDL 8.0" within the IDL 8.0 Help.

The new syntax was introduced in Version 8.0. May be a good idea not to use it, if you want to keep your code compatible with earlier versions.

Heinz

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by chris_torrence@NOSPAM on Wed, 31 Jul 2013 19:20:52 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: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Mark Piper on Thu, 01 Aug 2013 14:30:39 GMT
View Forum Message <> Reply to Message

On Wednesday, July 31, 2013 1:20:52 PM UTC-6, Chris Torrence wrote:
>
> 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.
>

And IDL-68399!

mp