Subject: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Chip Helms on Tue, 09 Jul 2013 17:06:05 GMT
View Forum Message <> Reply to Message

Hi all,

So I've been playing around with using IDL_IDLBridge to spawn child processes in the hope of reducing the time it takes to get through a pile of data. For testing this method, I've been calculating the average distance from each point on a grid to every other point on that grid. I have a routine that calculates the distance to every point in the grid from a single point. The traditional, single process approach would loop through each grid point and call this function. The child process approach I have written has 4 children, each one executes the function for a different grid point before I advance the loop to the next set of four grid points. The idea is that I have all four children running simultaneously and once all four are complete, I will take the output and concatenate it onto an array. (Once I figure out how to do this I'd like to apply this to performing operations whose results are arrays of data, so having the child process write the results to a file isn't preferable as I'll be processing 32 years worth of 6 hourly data).

So here I come across a couple of questions:

First off, am I crazy for trying to use IDL_IDLBridge outside of anything to do with widgets? All of the examples in the IDL documentation make use of the bridge in conjunction with widgets and, as I've never used widgets before, the example code can be somewhat hard for me to follow. (To this end http://slugidl.pbworks.com/w/page/29199259/Child%20Processes has been very useful).

Second question: What would be the best way to have my loop wait until each child has finished crunching the data? Should I just use a while loop that checks the status of the children every second or so? I feel like there should be a better way than this.

Thanks in advance,
Cheeers,
Chip

P.S. It was really hard not to title this "A question about good parenting with multiple children"

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Phillip Bitzer on Tue, 09 Jul 2013 21:33:59 GMT
View Forum Message <> Reply to Message

Yes, the bridge object can be used for things other than widgets.

Have you seen: http://www.iluvatar.org/~dwijn/parallelidl ? I'll admit that I do something a little different, but I think this might lead you down the right path. Basically, you're looking to set the callback property to handle writing the output of the distance calculation (your properly behaving children) to your array (your doting parent). It's a respectful way your children can talk to their

parent :-)

As an aside, there are better ways to get the distance between points in an array. See http://www.idlcoyote.com/code_tips/fastdist.php and the related newsgroup discussion. But, it looks like you're simply using this as an example of how to use bridges.

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Chip Helms on Tue, 09 Jul 2013 21:47:44 GMT
View Forum Message <> Reply to Message

Thanks, I hadn't seen that link. I'll have to take a closer look tomorrow. I got my actual project working using a while loop to wait for the child processes to finish. I have to say, I'm impressed by the lack of overhead. Using the four child processes resulted in the runtime for each iteration dropping from ~57s to ~15 seconds (my guess is it's because I only create the bridge objects once and then reuse them for each iteration...so the overhead is primarily passing data back and forth).

Thanks again for the link. By the sound of it, I imagine it's a bit more elegant than what I've labeled as a babysitter loop.

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Russell Ryan on Wed, 10 Jul 2013 02:57:13 GMT
View Forum Message <> Reply to Message

On Tuesday, July 9, 2013 5:47:44 PM UTC-4, Chip Helms wrote:
> Thanks, I hadn't seen that link. I'll have to take a closer look tomorrow. I got my actual project working using a while loop to wait for the child processes to finish. I have to say, I'm impressed by the lack of overhead. Using the four child processes resulted in the runtime for each iteration dropping from ~57s to ~15 seconds (my guess is it's because I only create the bridge objects once and then reuse them for each iteration...so the overhead is primarily passing data back and forth).
>
>
>
> Thanks again for the link. By the sound of it, I imagine it's a bit more elegant than what I've labeled as a babysitter loop.

Well, I'd start with mastering the IDL_IDLBridge before integrating it into a widget program.

But before you do that, let me strongly caution you. IDL_IDLBridge has a *MAJOR* bug in it --- there is a serious memory leak. IDL does not free the internal memory usage when an asychronous command. I'm certain of this behavior in Mac OSX, and have heard it's true for Linux (I don't actually have access to a Linux machine, but a friend said it was true). I don't use

Windows, but think Exelis said it was not a problem on Windows. This AFFECTS parallelidl as well, because all that code does is all the book keeping of the bridges for you.

As with most things in life, there are work arounds. Unfortunately, this workaround requires you kill IDL and restart, so it's not really a work around per se. The memory leak does happen, but may not really kill you if you're only triggering each bridge a few dozen times. If you're triggering each bridge more than a few 100 times (and it doesn't seem to scale with the data operated on by the bridge), then you can start eating away at an appreciable amount of your RAM. If let run for a long time (many thousand triggers), it will completely kill the computer. The computer will start to swap to disk and the execution will grind to a halt.

Obviously, I've filed a bug report with Exelis. They confirmed everything I've said, and are "working" on it. That was many months (if not close to a year) ago, and it has not been addressed in any of the releases of IDL 8.x.

In short, I highly discourage you from using IDL_IDLBridge until this is fixed --- even if you're writing code that may not be affected (such as Windows) or expect the run to be short (so maybe the memory leaks won't pile up). Eventually, that code (if it's any good) will proliferate to workstations were it will cause problem. In my line of work, I'm aware of two IDL codes that have this problem and I see people happily using them. Normally I wouldn't care, but we share workstations and when they kill the machine, it hurts all of us.

But, if you must. Then yes. You need to poll every bridge and ask for its status. If the status is running, then go to the next bridge. If the bridge is done, then re-trigger. You'll find that a wait of short time between successive polls will actually be faster. Also, you'll need to be very careful because the order in which the bridges are started, may not be the order in which they finish. Consequently, whatever data the bridge processed may need to be sorted out, this can be accomplished by the USERDATA keyword to keep track of which data a bridge processed.

Good luck, you're going to need it.
Russell

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Chip Helms on Wed, 10 Jul 2013 16:31:20 GMT
View Forum Message <> Reply to Message

Thanks for the heads up on that memory leak. I knew there was the potential for memory leaks if you don't destroy the objects at the end, but was unaware of any additional leaks. I'm running this on a departmental server, so I always like to be cautious of how I'm using the resources (hence only using 4 bridges at a time).

When you say 'triggering each bridge', do you mean just running 'Execute' or every time I make any call to the bridge?

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by natha on Wed, 10 Jul 2013 16:55:12 GMT
View Forum Message <> Reply to Message

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

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Chip Helms on Wed, 10 Jul 2013 17:14:45 GMT
View Forum Message <> Reply to Message

So if I'm reusing the same variables (overwritten with new data) for each new execution, would this result in just a single set of the variables leaking into memory (for each bridge), or will IDL allocate the new data to a new memory address? It probably wouldn't be a bad idea to delete the variables each iteration anyway just to be on the safe side.

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by dpmc07 on Thu, 11 Jul 2013 19:30:11 GMT
View Forum Message <> Reply to Message

Hey Guys,

I think it may be that the IDL timers aren't destroyed correctly.. you could try

pro clear_idl_timers

```
  annoying = widget_info(/managed)
  for i = 0, n_elements(annoying)-1 do $
    if widget_info(annoying[i],/event_pro) eq 'IDL_IDLBRIDGETIMER_EVENT' then $
      widget_control,annoying[i],/destroy

end
```

Cheers,
Dan Perera

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by dpmc07 on Thu, 11 Jul 2013 19:47:23 GMT
View Forum Message <> Reply to Message

Hi All,

I think the IDL bridge timers aren't destroyed correctly. You could try running this when you all the children have finished doing their stuff.

```
pro clear_idl_timers

  annoying = widget_info(/managed)
  for i = 0, n_elements(annoying)-1 do $
    if widget_info(annoying[i],/event_pro) eq 'IDL_IDLBRIDGETIMER_EVENT' then $
      widget_control,annoying[i],/destroy

end
```

Cheers,
Dan Perera

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by natha on Thu, 11 Jul 2013 20:18:00 GMT
View Forum Message <> Reply to Message

Hahahaha Are you one of the excelis guys? How is it possible that you have found something like that?...
I don't know if this is a final solution, because you may have several bridges running at the same time and you are destroying all the timers. I don't know how this thing would interfere with other bridges.

I think this is something to be tested...

nata

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by dpmc07 on Thu, 11 Jul 2013 20:43:19 GMT
View Forum Message <> Reply to Message

I'm think I just got lucky testing out a bunch of things. Saw that the X manager was going crazy using top after thousands of passes to the children. destroying the bridge objects didn't seem to work, so looked at the stuff the xmanager was dealing with and there were thousands of the idlbridge timers. Deleted them and my programs been doing well since. I also threw an empty command in there, though don't know if it does anything, and also periodically destroy the bridge objects too, plus explicitly delete and passed variables in the children. No memory issues since :)

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by Chip Helms on Thu, 11 Jul 2013 20:50:45 GMT
View Forum Message <> Reply to Message

So I suppose ideally, we should add this to the callback routine, right? Is there any way to identify which timer is associated with a given 'execute, /nowait' call?

Subject: Re: Speeding up data crunching using IDL_IDLBridge with asychronous execution
Posted by natha on Thu, 11 Jul 2013 23:20:10 GMT
View Forum Message <> Reply to Message

On Thursday, July 11, 2013 4:50:45 PM UTC-4, Chip Helms wrote:
> So I suppose ideally, we should add this to the callback routine, right? Is there any way to identify which timer is associated with a given 'execute, /nowait' call?

It seems that the timer ID is concatenated to the other ones. So you just have to get the last ID. It works....

```
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

  timer_id=widget_info(/managed)
  ww=where(widget_info(timer_id,/event_pro) eq 'IDL_IDLBRIDGETIMER_EVENT',nn_w)
  timer_id=timer_id[ww[nn_w-1]]
  widget_control, timer_id, /destroy
```

```
  print, memory(/high)
endfor
```