Subject: Re: IDL_IDLBridge and the virtual machine
Posted by Russell[1] on Thu, 14 Mar 2013 17:26:43 GMT
View Forum Message <> Reply to Message

Hi Helder,

Not exactly. Suppose in your main program you want to initialize 5 bridges. You will need to set (and maybe get) some data to (from) each of them at startup (at completion). You *could* use the SetVar (GetVar) methods to the IDL_IDLBridge, but as you said it can't be used with VM but also those methods are slow as they duplicate the memory usage. Meaning, if you have a data cube that is 1Gb in size that you pass to five bridges, then you have to duplicate that data five times, now ballooning your total RAM usage to 5 Gb. Perhaps that's not an issue for you, but the VM usage seems to be.

The alternative to GetVar and SetVar is to use the shared memory maps. Here any data you want to pass between children and parent processes you will store in a shared memory map. Then, to use the data from within the bridge, you simply initialize the shared memory map. Here's a simple example:

```
pro test
cd,cur=cwd & cwd=cwd[0]+'/'     ;get the current working directory

data2share=[1.,2.,3]

;create shared memory of the data
dim=size(data2share,/dim)
type=size(data2share,/type)
shmmap,'shareddata',dim=dim,type=type

;we'll need this command to set the data to the bridges, but
;only want to do it one time...
cmd='shmmap,"shareddata",dim='+string(dim,f='(I1)')+',type='+string(type,f='(I2)')



;now populate that shared memory
d=shmvar('shareddata')
d[0]=data2share

n=2                         ;number of bridges


bridges=objarr(n)            ;save the objects
for i=0,n-1 do begin

   ;create the bridge
    bridges[i]=obj_new('IDL_IDLBridge',out=cwd+string(i,f='(I1)' )+'.log')
   ;set the data to the bridge
   bridges[i]->Execute,cmd

   ;restore the data within a bridge
   bridges[i]->Execute,"thisdata = shmvar('shareddata')"

   ;print the data within the bridge
   bridges[i]->Execute,"print,thisdata"
endfor


;destroy the objects
for i=0,n-1 do obj_destroy,bridges[i]


end
```

Now, look in the files 0.log, 1.log, etc. Then you could get the data out via a similar prescription.

This has (one big advantage) for me, is that you do not duplicate any data in the children processes. *BUT* it's also considerably faster than SetVar and GetVar (as described by Fanning). I'm not sure if it'll work with the VM stuff or not, because you still have to use the Execute method (which smells an awful lot like the execute procedure).

-Russell

On Thursday, March 14, 2013 11:21:24 AM UTC-4, Helder wrote:
> On Thursday, March 14, 2013 3:30:04 PM UTC+1, rr...@stsci.edu wrote:
>
>> On Wednesday, March 13, 2013 7:58:07 AM UTC-4, Helder wrote:
>
>>
>
>>> Hi,
>
>>
>
>>>
>
>>
>
>>> I would like to use the IDL_IDLBridge, but I just read in the documentation that it is not possible to use execute, getvar and setvar methods in VM mode.
>
>>

>
>>>
>
>>
>
>>> This is quite annoying, but is there another functional way to parallelize computations in IDL?
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> I basically have a list of n images (n~100, maybe more) and I perform computations on images k with k-1 for the whole list. I wanted to start computations separately:
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> Res1 = ComputationFunction(RefImg=0, Img=1)
>
>>
>
>>>
>
>>
>
>>> Res2 = ComputationFunction(RefImg=1, Img=2)
>

>>
>
>>>
>
>>
>
>>> Res3 = ComputationFunction(RefImg=2, Img=3)
>
>>
>
>>>
>
>>
>
>>> ...
>
>>
>
>>>
>
>>
>
>>> and put all the results in an array.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> Any suggestions?
>
>>
>
>>>
>
>>
>
>>>
>

>>
>
>>>
>
>>
>
>>> I could try writing a batch process and producing a .sav that puts the result in a file and so on... but this seems a bit "dirty".
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> Thanks,
>
>>
>
>>>
>
>>
>
>>> Helder
>
>>
>
>>
>
>>
>
>> Hi Helder,
>
>>
>
>>
>
>>
>
>> Two things.

>
>>
>
>>
>
>>
>
>> (1) Look at the shmmap and shmvar routines for passing data. You essentially create a shared memory space and load the data there. Then all the nodes use a single string variable to initialize that memory space within their scope. Fanning has a good tutorial on this:
>
>>
>
>> http://www.idlcoyote.com/code_tips/bridge.html
>
>>
>
>>
>
>>
>
>> (2) I'm not sure how you're planning on using the IDL_IDLBridge. But if you use the /nowait feature (which, frankly is the main reason for using the Bridge in the first place), then there is a major memory leak which affects (it affects Linux and Mac as far as I can tell). If you only call the bridges a few times (say <100), it'll probably be fine --- the leaked memory will be smaller than your RAM so you'll never notice. But, if you call it many times >1000 --- then the run time *PER NODE* will increase very rapidly. I've emailed Exelis about this and I pinged them about it yesterday (I have some simulation code that would really benefit from the Bridges). I'll post more if I hear anything.
>
>>
>
>>
>
>>
>
>> Good luck,
>
>>
>
>> Russell
>
>
>
> Hi Russell,
>
> thanks for the tip.
>

> Let me see if I got your point or not...
>
> What you're trying to tell me is to use shared memory on which different processes share their status and results and the main code launches and waits for these processes to finish running?
>
> This way I would avoid using IDL_IDLBridge, and simply Spawn .sav files that read the info from shared memory.
>
> I can see that this could work, but every time I spawn a new .sav file, the VM will up with the splash screen and each process has to be "clicked on".
>
>
>
> Thanks for the info on the memory allocation sizes (from the Coyote) and the memory leak problems. However, the latter hopefully does not influence me too much because I have to deal with something like hundreds of processes.
>
>
>
> Oh yeah... the idea is that I'm looking for image drift in a stack of n images (up to a few hundreds at a time). Each may take a few seconds to compute and have 4+ processors would speed up things (I hope).
>
>
>
> Cheers,
>
> Helder