Subject: IDL_IDLBridge GetVar vs. Shared Memory
Posted by Dick Jackson on Tue, 05 Dec 2006 21:27:26 GMT
View Forum Message <> Reply to Message

Hi all,

Using IDL_IDLBridge has been very helpful in some of our work lately, helping us to get more processing done while one IDL process is waiting on a DLL, and other fun. I thought I'd share some findings that show how results can come back 4-10 times faster using shared memory (SHMMAP, etc.) rather than GetVar(). Any comments on this are very welcome, and I'd be interested to see how this plays out on other hardware.

Notes:

I split out the SHM Cleanup timing since some applications may not require cleanup between successive calls.

I thought it a bit odd to have to do this when assigning the value on the bridge process:
  oBridge -> Execute, "shmA[0,0,0,0] = a"
... but if you use [0] then only the first row of values is filled in. Sure, it's faster, but... :-) (and if you use [0,0] only the first 2D plane is filled in, etc.) Here's my !Version:

{ x86 Win32 Windows Microsoft Windows 6.3 Mar 23 2006 32 64}

I'll send another posting with the .pro file as an attachment, in case that's more convenient for some to use.

Cheers,
-Dick

--
Dick Jackson Software Consulting http://www.d-jackson.com
Victoria, BC, Canada +1-250-220-6117 dick@d-jackson.com

=====

```
PRO SHMvsGetVar

;;  Using an IDL_IDLBridge process, test timings of getting results back
;;  using shared memory (SHM) vs. oBridge -> GetVar()
;;
;;  SHM appears to be four to ten times faster in my testing.
;;  Dick Jackson - dick@d-jackson.com
```

```
oBridge = Obj_New('IDL_IDLBridge')
shmName = 'SHMvsGetVar'

;;    Describe sizes of arrays to use for testing (as strings)

sizes = ['10', '1E7', '[10,1E6]', '[1E6,10]', '[1E4,1E3]', '[1E2,1E2,1E3]']
;;  '1E8','[10,1E7]','[1E7,10]','[1E4,1E4]','[1E3,1E2,1E3]','[1E 2,1E2,1E2,1E2]']

nTests = N_Elements(sizes)

FOR testI=0, nTests-1 DO BEGIN

  sizeStr = sizes[testI]
  ok = Execute('sizeNum = Long('+sizeStr+')')
  IF ~ok THEN Message, 'Execute() failed: check sizeStr "'+sizeStr+""

  Print, 'Testing with byte array of size: '+sizeStr
  oBridge -> Execute, 'a=BIndGen('+sizeStr+')'
  localA = BIndGen(sizeNum)

  ;;    Test SHM method

  t0 = SysTime(/Seconds)
  oBridge -> Execute, "SHMMap,'"+shmName+"',/Byte,Dimension="+sizeStr
  oBridge -> Execute, "shmA = SHMVar('"+shmName+"')"
  oBridge -> Execute, "shmA[0,0,0,0] = a" ; Must have N_Dims(a) or more (bug?)
  SHMMap, shmName,/Byte,Dimension=sizeNum
  shmA = SHMVar(shmName)
  shmTime = SysTime(/Seconds)-t0
  Print, Format="('    SHM:        ',F0.3,' s')",shmTime

  ;;    Check result
  IF ~(Array_Equal(Size(shmA), Size(localA)) && $
      Array_Equal(shmA, localA)) THEN Print, '   *** Result check failed!'

  ;;    Remove references to mapped variables and unmap memory segments
  t0 = SysTime(/Seconds)
  shmA = 0B
  SHMUnmap, shmName
  oBridge -> Execute, "shmA = 0B"
  oBridge -> Execute, "SHMUnmap,'"+shmName+"'"
  Print, Format="('    SHM+Cleanup:',F0.3,' s')",shmTime+(SysTime(/Seconds)-t0)

  ;;    Test GetVar method

  t0 = SysTime(/Seconds)
  getVarA = oBridge -> GetVar('a')
  Print, Format="('    GetVar:     ',F0.3,' s')",SysTime(/Seconds)-t0
```

```
  ;;    Check result
  IF ~(Array_Equal(Size(getVarA), Size(localA)) && $
      Array_Equal(getVarA, localA)) THEN Print, '   *** Result check failed!'

  Wait, 0.001                    ; To allow Print statements to flush

ENDFOR

Obj_Destroy, oBridge

END
```

Subject: Re: IDL_IDLBridge GetVar vs. Shared Memory
Posted by JD Smith on Tue, 05 Dec 2006 23:16:43 GMT
View Forum Message <> Reply to Message

On Tue, 05 Dec 2006 21:27:26 +0000, Dick Jackson wrote:

> Hi all,
>
> Using IDL_IDLBridge has been very helpful in some of our work lately, helping us
> to get more processing done while one IDL process is waiting on a DLL, and other
> fun. I thought I'd share some findings that show how results can come back 4-10
> times faster using shared memory (SHMMAP, etc.) rather than GetVar(). Any
> comments on this are very welcome, and I'd be interested to see how this plays
> out on other hardware.

Thanks Dick, looks very interesting.

I tried it, and shared memory definitely seems to be much faster for me,
up to 10x. I also added back your 1E8 case, and got this unhelpful error:

```
IDL> shmvsgetvar
% Loaded DLM: IDL_IDLBRIDGE.
Testing with byte array of size: 1E8
% IDL_IDLBRIDGE Error: Undefined variable: 'a'
% Execution halted at: SHMVSGETVAR       39
  /home/jdsmith/idl/test/test_shm_iib.pro
%                    $MAIN$
```

Any reason why you can't create and retrieve a 100MB array? Then I
recalled advice from earlier this week:

```
 echo 1073741824 > /proc/sys/kernel/shmmax
```

After increasing this limit, it ran fine. The default is 32MB in

Linux/Intel.

What's quite interesting, however, is that I noticed the SHMMap version didn't run into this limit. Without any modification to shmmax, it would still happily send 1GB across shared memory. It wasn't abundantly clear why IDL_IDLBridge's GetVar hits this limit, while SHMMap does not.

I thought perhaps the Bridge uses SYSV shared memory segments (whereas the latter defaults to the more modern POSIX flavor), I modified the code to read:

```
SHMMap, shmName,/Byte,Dimension=sizeNum,/SYSV
```

and sure enough, hit the 32MB limit quickly. Also, somewhat strangely, with SYSV shared memory, only 0b was returned from the shared memory segment. Then I realized you need an "OS handle" for SYSV, but not for POSIX, since POSIX shared memory just uses the supplied name with a slash. After passing OS_HANDLE back to the IDL Bridge, it worked fine. The speed was almost identical (actually SYSV shared memory seems about 5% faster).

So, it appears that:

1) POSIX shmem "does the right" thing, without any arbitrary limits on the amount of shared memory.

2) SYSV shmem runs into a maximum memory limit defined by your kernel.

3) IDL_IDLBridge uses SYSV shared memory under UNIX to implement GetVar, thus inheriting this limit.

It's not at all clear why they use SYSV shmem for the Bridge, since they go on about how important POSIX shared memory is and how all decent Unixes support it, etc.

The bottom line is, not only is it much faster to use SHMMap (without /SYSV), but it's probably more portable and obviously less likely to run into arbitrary memory boundaries. So if you plan on passing any data larger than a few MB through the IDL_IDLBridge, do look into it.

Thanks,

JD

Subject: Re: IDL_IDLBridge GetVar vs. Shared Memory

Posted by Richard French on Wed, 06 Dec 2006 00:40:05 GMT
View Forum Message <> Reply to Message

Timing results with Intel Mac 3 GHz, 8 GB RAM:

IDL> print,!version
{ i386 darwin unix Mac OS X 6.3 Jun 27 2006 32 64}

IDL> shmvsgetvar
Testing with byte array of size: 10
SHM: 0.000 s
SHM+Cleanup:0.000 s
GetVar: 0.000 s
Testing with byte array of size: 1E7
SHM: 0.016 s
SHM+Cleanup:0.019 s
GetVar: 0.125 s
Testing with byte array of size: [10,1E6]
SHM: 0.030 s
SHM+Cleanup:0.032 s
GetVar: 0.126 s
Testing with byte array of size: [1E6,10]
SHM: 0.017 s
SHM+Cleanup:0.020 s
GetVar: 0.128 s
Testing with byte array of size: [1E4,1E3]
SHM: 0.015 s
SHM+Cleanup:0.018 s
GetVar: 0.128 s
Testing with byte array of size: [1E2,1E2,1E3]
SHM: 0.016 s
SHM+Cleanup:0.019 s
GetVar: 0.130 s
Testing with byte array of size: 1E8
SHM: 0.159 s
SHM+Cleanup:0.177 s
GetVar: 1.225 s
Testing with byte array of size: [10,1E7]
SHM: 0.278 s
SHM+Cleanup:0.296 s
GetVar: 1.259 s
Testing with byte array of size: [1E7,10]
SHM: 0.156 s
SHM+Cleanup:0.195 s
GetVar: 1.247 s
Testing with byte array of size: [1E4,1E4]
SHM: 0.138 s
SHM+Cleanup:0.157 s

```
   GetVar:     1.257 s
Testing with byte array of size: [1E3,1E2,1E3]
   SHM:        0.141 s
   SHM+Cleanup:0.162 s
   GetVar:     1.240 s
Testing with byte array of size: [1E2,1E2,1E2,1E2]
   SHM:        0.152 s
   SHM+Cleanup:0.172 s
   GetVar:     1.241 s
IDL>
```
