Subject: IDL_IDLBridge: No gain?
Posted by dg86 on Sat, 10 Aug 2013 20:05:56 GMT

View Forum Message <> Reply to Message

Dear Folks,

I'd like to run multiple instances of a computationally expensive routine in separate threads to improve performance. I tried to achieve this by running each instance in a separate IDL_IDLBridge with the NOWAIT flag set. Disappointingly, two instances require roughly twice as long to run as one. There seems to be no benefit! What am I doing wrong?

I'm running IDL 8.2.3 64-bit on a MacBook Pro with 8 cores.

Many thanks,

David

Here's a sketch of my attempt using just two IDL_IDLBridge objects:

```
a = fltarr(640,480) + 1.                    ; in practice, this would be data
cmd = 'p = lmfeature(a, 0.447, 0.135)' ; the expensive routine

bridge1 = IDL_IDLBridge()
bridge2 = IDL_IDLBridge()
bridge1.setvar, 'A', a
bridge2.setvar, 'A', a ; use same data for demo

tic
bridge1.execute, cmd, /nowait
bridge2.execute, cmd, /nowait

while 1 do $
   if (bridge1.status() eq 1) or (bridge2.status() eq 1) then $
      wait, 0.1 $
   else $
      break
toc

p1 = bridge1.getvar('P')
p2 = bridge2.getvar('P')
```

Subject: Re: IDL_IDLBridge: No gain?
Posted by rlkling on Mon, 12 Aug 2013 11:08:23 GMT

View Forum Message <> Reply to Message

Hi David,

What's killing you is the wait in the while loop. What you need to do is specify a callback for your routines to call after they are done.

In the callbacks you can set some flag to indicate when they are called and then in your main program just check the status of the flags every second or so.

-Ronn Kling

Subject: Re: IDL_IDLBridge: No gain?
Posted by dg86 on Mon, 12 Aug 2013 13:59:43 GMT

View Forum Message <> Reply to Message

HI Ronn,

Thanks very much for fielding my question. I added a callback routine that prints
the bridge's ID the elapsed time in the bridge's execution. Both the ID and the
starting time are passed to each bridge as USERDATA upon creation. I removed the
wait statements altogether. Instead, I stare at the command line, which reappears
immediately, until the bridges report back.

The bridges report back all at once, and in apparently random order, suggesting that they
are indeed running simultaneously and not sequentially. They report nearly the same
running time. Unfortunately, that time seems to scale linearly with
the number of running bridges.

Two bridges each report needing 11 seconds to run my expensive routine (lmfeature).
Four bridges each report 18.5 seconds.
Seven bridges report 39 seconds.
Eight bridges report 46 seconds.

Why does the execution time for one bridge depend on the number of bridges running simultaneously?

All the best,

David

On Monday, August 12, 2013 7:08:23 AM UTC-4, rlk...@gmail.com wrote:
> Hi David,
>
>
>
> What's killing you is the wait in the while loop. What you need to do is specify a callback for your routines to call after they are done.
>

>
>
> In the callbacks you can set some flag to indicate when they are called and then in your main program just check the status of the flags every second or so.
>
>
>
> -Ronn Kling

Subject: Re: IDL_IDLBridge: No gain?
Posted by Lajos Foldy on Mon, 12 Aug 2013 15:05:08 GMT
View Forum Message <> Reply to Message

On Monday, August 12, 2013 3:59:43 PM UTC+2, David Grier wrote:

> Why does the execution time for one bridge depend on the number of bridges running simultaneously?

Some IDL routines are multithreaded, they are using all CPU cores. If your bridges use mainly these MT routines, their are competing for the CPU cores. Running N bridges means that each bridge gets only 1/Nth of the total CPU resources, so each bridge will be slower.

regards,
Lajos

Subject: Re: IDL_IDLBridge: No gain?
Posted by dg86 on Mon, 12 Aug 2013 15:16:04 GMT
View Forum Message <> Reply to Message

On Monday, August 12, 2013 11:05:08 AM UTC-4, fawltyl...@gmail.com wrote:
> On Monday, August 12, 2013 3:59:43 PM UTC+2, David Grier wrote:
>
>
>
>> Why does the execution time for one bridge depend on the number of bridges running simultaneously?
>
>
>
> Some IDL routines are multithreaded, they are using all CPU cores. If your bridges use mainly these MT routines, their are competing for the CPU cores. Running N bridges means that each bridge gets only 1/Nth of the total CPU resources, so each bridge will be slower.
>
>
>

> regards,
>
> Lajos

Ahh, that makes sense. That also would explain why the scaling does not get much worse when the
number of bridges exceeds the number of cores.

Sounds like it's time for me to get to work with PROFILER.

All the best,

David