Subject: Re: Threads in IDL 7.0
Posted by Allan Whiteford on Tue, 28 Oct 2008 15:45:22 GMT
View Forum Message <> Reply to Message

Vince Hradil wrote:
> On Oct 27, 12:26 pm, Heinz Stege <public.215....@arcor.de> wrote:
>
>> On Mon, 27 Oct 2008 16:44:29 +0100, Bernhard Reinhardt wrote:
>>
>>> Well, I wasnï¿½t really precise. Iï¿½m not doing this on a 1d-array but on a
>>> 4d-array, where 2 dimensions are time and 2 dimensions are space. I try
>>> to filter special events in time and count those on a 2d-map. Hereï¿½s the
>>> code:
>>

```
>>>      for i = 0, N_ELEMENTS(STRUC.data[*,0,0,0])-1 do begin
>>>        for j = 0, N_ELEMENTS(STRUC.data[0,*,0,0])-1 do begin
>>>          indices = Where(STRUC.data[i,j,*,*] GE 150., count)
>>>          freq [i,j]=count
>>>        endfor
>>>      endfor
```

>>
>>> Although the array data is quite big "where" only getï¿½s a small portion
>>> to see of it. So thread-pool isnï¿½t invoked. => CPU-Usage still 50%
>>
>>> I also asked some more IDL-experienced colleagues about generating
>>> threads manually but they also didnï¿½t know about anything like that :(
>>
>>> BUT using your method still brought me a gain of 3.6 times faster
>>> execution :)
>>
>>> regards
>>
>>> Bernhard
>>
>> Please try the following command:
>>

```
>>   freq=total(total(STRUC.data ge 150.,4,/integer),3,/integer)
```

>>
>> The IDL manual says, that TOTAL makes use of IDLï¿½s thread pool. And
>> there is no for-loop needed anymore...
>>
>> HTH, Heinz
>
>
> At the risk of repeating myself (I tried to post this earlier, but it
> hasn't shown up?):
>
> Which is faster - (1)TRANSPOSE and TOTAL over leading dims or (2)

```
>  TOTAL over trailing dims
>
>  Here's my quick (un-scientific) test:
>
>  IDL> arr = bytscl(randomu(sd, [100,101,102,103]))
>  IDL> t0 = systime(1) & help, total(total(arr,4,/integer),3,/integer) &
>  print, systime(1)-t0
>  <Expression>    LONG64    = Array[100, 101]
>       0.42199993
>  IDL> t0 = systime(1) & help, total(total(transpose(arr,[2,3,0,1]),1,/
>  integer),1,/integer) & print, systime(1)-t0
>  <Expression>    LONG64    = Array[100, 101]
>        1.7979999
>  IDL> print, !version
>  { x86 Win32 Windows Microsoft Windows 7.0 Oct 25 2007      32      64}
```

It makes a certain amount of sense that doing the total over trailing dims would be faster. When you take the transpose IDL is having to go over the array in a non-optimal fashion anyway so you'd be as well extracting the totals at the same time.

The solution given by Heinz goes over the data in a non-optimal way (a) getting the totals (b).

The solution given by me goes over the data in a non-optimal way (1), copying all the data (2) then goes over the copied data in an optimal way (3) getting the totals (4).

In the above (a) and (1) will take about the same time as will (4) and (b). So I'm doing (2) and (3) which Heinz wasn't. I'd expect (3) to be fairly quick but I guess (2) will be the one which eats up the unnecessary time.

In my defense I only looked at the loops from Lajos' solution and seen than they could go - I didn't step back and look at the original question to see a better overall solution.

I never knew (or forgot about) the /integer keyword to total().

Thanks,

Allan