Subject: Re: Threads in IDL 7.0
Posted by David Fanning on Mon, 27 Oct 2008 13:23:10 GMT
View Forum Message <> Reply to Message

Bernhard Reinhardt writes:

> I have have read the Chapter "Multithreading in IDL" in the IDL-Help. As
> far as I can see some IDL functions use threads.
>
> Basically I want to count elements of a big array that exceed a given
> number.
>
> count=0
> for i = 0, 200000 do begin
> if array[i] ge 10. then count++
> endfor

How about something like this:

```
   indices = Where(array GE 10, count)
```

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Threads in IDL 7.0
Posted by Bernhard Reinhardt on Mon, 27 Oct 2008 15:44:29 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Bernhard Reinhardt writes:
>
>> I have have read the Chapter "Multithreading in IDL" in the IDL-Help. As
>> far as I can see some IDL functions use threads.
>>
>> Basically I want to count elements of a big array that exceed a given
>> number.
>>
>> count=0
>> for i = 0, 200000 do begin
>> if array[i] ge 10. then count++
>> endfor

>
> How about something like this:
>
>   indices = Where(array GE 10, count)

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

Subject: Re: Threads in IDL 7.0
Posted by Foldy Lajos on Mon, 27 Oct 2008 16:11:39 GMT

View Forum Message <> Reply to Message

On Mon, 27 Oct 2008, Bernhard Reinhardt wrote:

> David Fanning wrote:
>> Bernhard Reinhardt writes:
>>
>>> I have have read the Chapter "Multithreading in IDL" in the IDL-Help. As
>>> far as I can see some IDL functions use threads.
>>>
>>> Basically I want to count elements of a big array that exceed a given
>>> number.
>>>
>>> count=0
>>> for i = 0, 200000 do begin

```
>>>    if array[i] ge 10. then count++
>>>   endfor
>>
>>   How about something like this:
>>
>>      indices = Where(array GE 10, count)
>
```

> Well, I wasn´t really precise. I´m 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. Here´s 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)
```

You are accessing elements in the wrong order, resulting in cache misses
if struc.data is greater than the CPU data cache.

```
>           freq [i,j]=count
>         endfor
>       endfor
>
```

Try this:

```
d=transpose(struc.data, [2,3,0,1])
dims=size(d, /dim)
for i = 0, dims[2]-1 do begin
   for j = 0, dims[3]-1 do begin
     indices = Where(d[*,*,i,j] GE 150., count)
     freq [i,j]=count
   endfor
endfor
```

regards,
lajos

Subject: Re: Threads in IDL 7.0
Posted by Allan Whiteford on Mon, 27 Oct 2008 16:46:29 GMT
View Forum Message <> Reply to Message

Bernhard Reinhardt wrote:
> Hi,
>
> I have have read the Chapter "Multithreading in IDL" in the IDL-Help. As
> far as I can see some IDL functions use threads.

```
>
> Basically I want to count elements of a big array that exceed a given
> number.
>
> count=0
> for i = 0, 200000 do begin
>     if array[i] ge 10. then count++
> endfor
>
> On my dual-core computer top shows a CPU-load of about 50%. So I guess
> IDL handles this problem single-threaded.
>
> Is there a simple way to split the task. Something like
>
> ;Thread 1
> count1=0
> for i = 0, 100000 do begin
>     if array[i] ge 10. then count1++
> endfor
>
> ;Thread 2
> count2=0
> for i = 100001, 200000 do begin
>     if array[i] ge 10. then count2++
> endfor
>
> count = count1 + count2
>
> Regards
>
> Bernhard
```

Bernhard,

You can't get access to the threads directly as far as I know, you might want to read over:

http://www.ittvis.com/portals/0/whitepapers/IDL_MultiThread. pdf

basically it seems that ITTVIS thought it would be too complicated for scientists to understand and use.

It makes a certain amount of sense when you think through the complications of making things thread-safe although I think with a bit more work they could have delivered something far more useful which allows people who know what they are doing (or at least think they know what they are doing) to mess with threads.

Thanks,

Allan

Subject: Re: Threads in IDL 7.0
Posted by Foldy Lajos on Mon, 27 Oct 2008 16:59:29 GMT
View Forum Message <> Reply to Message

On Mon, 27 Oct 2008, Allan Whiteford wrote:

> You can't get access to the threads directly as far as I know, you might want
> to read over:
>
> http://www.ittvis.com/portals/0/whitepapers/IDL_MultiThread. pdf
>
> basically it seems that ITTVIS thought it would be too complicated for
> scientists to understand and use.

There is a simple solution for scientists (for C and Fortran :-):

http://www.openmp.org

For me, OpenMP for IDL would be much more appreciated than Eclipse and Co.

regards,
lajos

Subject: Re: Threads in IDL 7.0
Posted by Allan Whiteford on Mon, 27 Oct 2008 17:23:23 GMT
View Forum Message <> Reply to Message

```
>
> On Mon, 27 Oct 2008, Bernhard Reinhardt wrote:
>
>> David Fanning wrote:
>>
>>> Bernhard Reinhardt writes:
>>>
>>>> I have have read the Chapter "Multithreading in IDL" in the
>>> IDL-Help. As > far as I can see some IDL functions use threads.
>>>> > Basically I want to count elements of a big array that exceed a
>>> given > number.
>>>> > count=0
>>>> for i = 0, 200000 do begin
```

```
>>>>      if array[i] ge 10. then count++
>>>>   endfor
>>>
>>>   How about something like this:
>>>
>>>      indices = Where(array GE 10, count)
>>
>>
```

```
>>  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.
```

```
>>
>>       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)
>
>
>  You are accessing elements in the wrong order, resulting in cache misses
>  if struc.data is greater than the CPU data cache.
>
>>           freq [i,j]=count
>>         endfor
>>       endfor
>>
>
>  Try this:
>
>  d=transpose(struc.data, [2,3,0,1])
>  dims=size(d, /dim)
>  for i = 0, dims[2]-1 do begin
>    for j = 0, dims[3]-1 do begin
>      indices = Where(d[*,*,i,j] GE 150., count)
>      freq [i,j]=count
>    endfor
>  endfor
>
>
>  regards,
>  lajos
```

Or this:

```
freq = fix(total(total(transpose(struc.data,[2,3,0,1]) ge 150.,1),1))
```

which should go faster because of the lack of loops.

Thanks,

Allan

Subject: Re: Threads in IDL 7.0
Posted by Heinz Stege on Mon, 27 Oct 2008 17:26:40 GMT
View Forum Message <> Reply to Message

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

Subject: Re: Threads in IDL 7.0
Posted by Allan Whiteford on Mon, 27 Oct 2008 17:27:55 GMT

Fï¿½LDY Lajos wrote:
>
>
> On Mon, 27 Oct 2008, Allan Whiteford wrote:
>
>> You can't get access to the threads directly as far as I know, you
>> might want to read over:
>>
>> http://www.ittvis.com/portals/0/whitepapers/IDL_MultiThread. pdf
>>
>> basically it seems that ITTVIS thought it would be too complicated for
>> scientists to understand and use.
>
>
> There is a simple solution for scientists (for C and Fortran :-):
>
> http://www.openmp.org
>
> For me, OpenMP for IDL would be much more appreciated than Eclipse and Co.
>
> regards,
> lajos
>

Me too but I guess we're not the customers they are after.

If they managed to go down the threading path more seriously then maybe
we'd get a re-entrant parser along the way - what a wonderful day that
would be!

http://www.txcorp.com/products/FastDL/ is worth a look for anyone
interested.

Thanks,

Allan

Subject: Re: Threads in IDL 7.0
Posted by David Fanning on Mon, 27 Oct 2008 18:07:07 GMT

Allan Whiteford writes:

> Me too but I guess we're not the customers they are after.

Clearly not, but while ITTVIS pursues its agenda of offering
more (at least from my point of view) useless bells and
whistles, they have done a better job of opening the
IDL architecture up to useful innovations by third-party
developers. It's not clear to me there is a market for
these products (since everything so far appears to be
given away for free), but perhaps it will be possible for
IDL's old-timey customers to provide some of the tools
that ITTVIS would, given its strategic vision, prefer not to.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Threads in IDL 7.0
Posted by Bernhard Reinhardt on Tue, 28 Oct 2008 08:28:42 GMT
View Forum Message <> Reply to Message

Heinz Stege 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...

Now thatï¿½s fast! Who cares about threads? ;) Thanks to everyone.

Regards

Bernhard

Subject: Re: Threads in IDL 7.0
Posted by Vince Hradil on Tue, 28 Oct 2008 13:01:28 GMT

View Forum Message <> Reply to Message

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 wasn´t really precise. I´m 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. Here´s 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 get´s a small portion
>> to see of it. So thread-pool isn´t invoked. => CPU-Usage still 50%
>
>> I also asked some more IDL-experienced colleagues about generating
>> threads manually but they also didn´t 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 IDL's thread pool. And
> there is no for-loop needed anymore...
>
> HTH, Heinz

So, which is faster?

```
freq = fix(total(total(transpose(struc.data,[2,3,0,1]) ge 150.,1),1))
```

or

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

TRANSPOSE or 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]
      0.45300007
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]
       1.5929999
IDL> print, !version
{ x86 Win32 Windows Microsoft Windows 7.0 Oct 25 2007      32      64}
```

Subject: Re: Threads in IDL 7.0
Posted by Vince Hradil on Tue, 28 Oct 2008 14:31:21 GMT

View Forum Message <> Reply to Message

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 wasn´t really precise. I´m 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. Here´s 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 get´s a small portion
>> to see of it. So thread-pool isn´t invoked. => CPU-Usage still 50%
>
>> I also asked some more IDL-experienced colleagues about generating
>> threads manually but they also didn´t 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 IDL's 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}
```

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