Subject: Re: find max in 3D array -- slow
Posted by Maxwell Peck on Mon, 12 Apr 2010 08:07:47 GMT
View Forum Message <> Reply to Message

```
On Apr 12, 7:41 am, Maxwell Peck <maxjp...@gmail.com> wrote:
>  On Apr 11, 4:23 am, FÖLDY Lajos <fo...@rmki.kfki.hu> wrote:
>
>
>
>>  On Sat, 10 Apr 2010, Timothy W. Hilton wrote:
>>>  Hello IDL users,
>
>>>  I have a 1200x1200x2900 array of floats.  The dimensions correspond to
>>>  latitude x longitude x time.  I need to find the maxium at each
>>>  location -- that is, I need the 1200x1200 array containing the max
>>>  along the 3rd dimsion.  IDL takes almost 3 minutes to do this on my
>>>  system.  This seemed slow.  I compared it with Matlab, which took ten
>>>  seconds.  Is there a better way to search for the maxima using IDL?
>
>>>  The demo code I used to compare IDL and Matlab is below (with output).
>
>>>  I'm wondering if I ought to switch to Matlab.  I just spent a couple
>>>  of days writing IDL code to read my data, so I'd rather not.
>
>>>  Many thanks,
>>>  Tim
>
>>>  --
>
>>>  Timothy W. Hilton
>>>  PhD Candidate, Department of Meteorology
>>>  The Pennsylvania State University
>>>  503 Walker Building, University Park, PA   16802
>>>  hil...@meteo.psu.edu
>
>>>  ========
>>>  scratch.pro:
>
>>>  foo = randomu(0, 1200, 1200, 2920)
>>>  PRINT, systime()
>>>  foo_max = max(foo, DIMENSION = 3)
>>>  PRINT, systime()
>>>  END
>
>>>  IDL> .run scratch
>>>  % Compiled module: $MAIN$.
>>>  Sat Apr 10 10:44:44 2010
>>>  Sat Apr 10 10:47:36 2010
>>>  IDL>
>
>>>  ========
>>>  scratch.m:
>
>>>  foo = rand(1200,1200,2920);
>>>  fprintf('%s\n', datestr(now()));
>>>  foo_max = max(foo, [], 3);
>>>  fprintf('%s\n', datestr(now()));
>
>>>> > scratch
>>>  10-Apr-2010 10:42:45
>>>  10-Apr-2010 10:42:55
>
>>  I think that randomu(0, 1200,1200,2920) should be rand(2920, 1200, 1200)
>>  in Matlab (an array of 2920 rows x 1200 columns x 1200 something). The
>>  memory layout makes a big difference.
>
>>  regards,
>>  lajos
>
>  That's probably a good point, maybe storing the dataset in the
>  equivalent of a Byte Interleaved by Pixel storage order would speed
>  things up considerably.
```

It certainly does seem to be memory layout related. Here are some numbers.

```
foo = randomu(seed, 100, 100, 2900)
```

```
foo_max = max(foo, DIMENSION = 3)
```
This takes 0.36 seconds

Reforming and transposing the array as follows:
```
h = transpose(reform(foo,100*100,2900))
```

Then finding the max along the row dimension
```
k=max(h,dimension=1)
```
gives 0.11 seconds. This is NOT including the initial transpose/reform (or one after). This adds considerable time. There might be a smarter way to do this bit...

Not using the transpose and finding the max along the columns gives similar times to using the dimension=3 as done initially.

Clearly having the values stored contiguous in memory as it is across the rows gives much faster results. I'm not sure if there are paging issues happening as well though, you're using a pretty big array!

I'm not sure what the best way in actual application is to do this in IDL, perhaps there is opportunity when the file is being read in to store it in this way as it's probably I/O limited anyway at this point. Someone smarter on here might have a better solution..

Max

```
pro testsort
l=100L
c=2900L

foo = randomu(seed, l, l, c)
t=systime(1)
foo_max = max(foo, DIMENSION = 3)
PRINT,systime(1) -t

h = transpose(reform(foo,l*l,c))
t=systime(1)
k=max(h,dimension=1)
PRINT,systime(1) -t
j=reform(k,l,l)

print,'make sure nothing stupid has happened', total(j-foo_max)

END
```
