Subject: Re: Faster approach for total(data,dimension) possible?
Posted by Jeremy Bailin on Wed, 24 Jun 2009 21:04:37 GMT
View Forum Message <> Reply to Message

On Jun 24, 4:53 pm, Jeremy Bailin <astroco...@gmail.com> wrote:
> On Jun 24, 12:34 pm, wlandsman <wlands...@gmail.com> wrote:
>
>
>
>> On Jun 24, 11:38 am, chris <rog...@googlemail.com> wrote:
>
>>> Min and Max approach is two times slower in my case, so this doesn't
>>> seem to be a solution. Any other ideas?
>
>> Be sure to calculate min and max at the same time, e.g.
>> mask1 = max(data,dimen=3,min=mask)
>> mask = (mask or mask1) NE 0
>
>> But it seems that the best performance is hardware dependent.
>> Below are the repeatable times in seconds I get for the different
>> methods for a 1536 x 231 x 126 array on different systems.
>
>> { x86_64 linux unix linux 7.0
>> TOTAL 0.26
>> TOTAL(/INTEGER) 0.28
>> TOTAL(byte) 0.17
>> MINMAX 0.25
>
>> (x_86_64 darwin unix Mac OS X 7.06)
>> TOTAL 0.24
>> TOTAL(/INTEGER) 0.16
>> TOTAL(byte) 0.22
>> MINMAX 0.24
>
>> Since you are getting the best times for the first (TOTAL()) method, I
>> suspect your hardware is optimized for floating point calculations.
>> If you were to code it in C (i.e. not worry about loops) the quickest
>> method should be some variant of ARRAY_EQUAL
>> where you stop the comparisons once you find a non-zero element in a
>> band. But until ARRAY_EQUAL gets a dimension keyword like MIN and
>> MAX I don't think any other IDL method is going to be much faster.
>> --Wayne
>
> How about using product? It should be well-optimized for the cases of
> multiplying-by-one and multiplying-by-zero:
>
> mask = ~product(data gt 0, 3, /preserve_type)
>

> -Jeremy.

Oops, that should of course read:

mask = ~product(data eq 0, 3, /preserve_type)

-Jeremy.
