Subject: Re: Interpolate whole array instead of looping through elements
Posted by wlandsman on Thu, 13 Apr 2017 04:10:51 GMT
View Forum Message <> Reply to Message

I can't help directly as I get dizzy working in more than 3 dimensions ;-)
But it looks like you can parallelize your code.   Just today I used the IDL_IDLBridge for the first time to farm out my code to 16 CPUs, and I was able to speed things up by a factor of 10.

You might be able to use the wrapper to the IDL_IDLBridge developed by Mike Galloy
 http://michaelgalloy.com/2015/05/26/a-simple-multicore-libra ry-for-idl.html
though it turns out it wasn't appropriate for my problem.   I also found the following Web page useful:
http://daedalus.as.arizona.edu/wiki/ParallelIDL

--Wayne

On Wednesday, April 12, 2017 at 7:50:17 PM UTC-4, liam....@gmx.co.uk wrote:
> thanks for the replies everyone,
>
> Well it is a little bit more complicated than I originally said. I made the original question simpler to avoid confusing things!
>
> Basically, temp is of size [280,280,60,720]. What I am actually doing is getting the average temperatures in a radially symmetric crater. Take the following image as an example: https://s22.postimg.org/6r50fbrld/bilinear.jpg
>
> Imagine the black grid is the lat-lon temperature field at a certain level and time (i.e. temp[*,*,0,0]). Once I have this 2D field, I need to calculate how the average temperature varies from the centre of the crater to the edge. So I define a line (shown in red, with the black dots the locations I want values at), and for each black dot I use the bilinear function to get a value. I then rotate the red line a bit more and do the calculation again, and repeat. On each line there are about 100 points.
>
> Once a full circle of rotations is complete, the average temps from the centre to the edge of the crater are found. But only for one time and one level. At the moment I'm rotating the line by 5 degrees. So each time and each level of data has 36 rotations with each rotation having 100 points on the line to use the bilinear function on. So, it's something like:
>
```
> for iangle = 0, 35 do begin
>   for ipoint = 0, 99 do begin
>
>     ; Find ival and jval of the point we want to interpolate to
>     ival = ....
>     jval = ....
>
>     for itime = 0, 719 do begin
>       for ilev = 0, 59 do begin
>         out_vals[ipoint,ilev,itime] = out_vals[ipoint,ilev,itime] +
bilinear(temp[*,*,ilev,itime],ival,jval)/36
>       endfor
>     endfor
>
>   endfor
> endfor
>
```
> And it goes really rather slowly. Looping through just iangle, ipoint and itime takes 131 seconds (using the TIC,TOC functions). This then needs multiplied by 60 to loop through each atmospheric level, so it takes more than two hours in total. And this is just for one lot of data. At the moment I have 50 or so of these to calculate, so that's almost 5 days of IDL calculation!
>
> I was thinking there was maybe something that could be done where the iangle and ipoint loops still occur (as they have to, in order to find the i and j indices for the bilinear interpolation), but then interpolation could occur for all itime and ilev values at once in some speedy IDL vectorized way (since they are using the same indices). But maybe not! Maybe I need to find something other than IDL that might be quicker. Or just accept it is going to take a while to calculate!
>
> Apologies if none of this makes sense!
>
> Liam