Subject: Re: Today's IDL Lesson
Posted by Allan Whiteford on Fri, 15 Aug 2008 09:05:50 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Kenneth P. Bowman writes:
>
>
>> Shouldn't that be "increases linearly with the image size and
>> number of invocations of WHERE"?
>
>
> I don't think so. A 1500x1500 array took about 3.2 sec.
> a 2500x2500 array took about 35 sec. Just eyeballing it,
> it doesn't look linear to me. :-)
>
> Cheers,
>
> David

This scared me!

So, I wrote the following:

```
pro profile_where

    max=26
    time=fltarr(max)
    sizes=(1+2*(findgen(max) mod 3))*10l^float(indgen(max)/3)

    for i=0l,max-1 do begin
 array=findgen(sizes[i])
       time[i]=call_external('libidl.so','clock')
 idx=where(array lt 3)
       time[i]=call_external('libidl.so','clock') -time[i]
    end

    plot,sizes,time/1e6,xtitle='Number of elements',ytitle='Time in s'
    oplot,sizes,time/1e6,psym=2

    ; or...

    plot,alog10(sizes),alog10(time/1e6),xrange=[5,10]
    oplot,alog10(sizes),alog10(time/1e6),xrange=[5,10],psym=2

end
```

and I get something which looks basically linear. The maximum array size were chosen so that I wasn't hitting swap space and the where() criterion so that each iteration returned the same number of results (hence allocated the same amount of memory).

I have no doubt that in real world applications we get non-linear scaling due to hitting swap or various other reasons but I think the above shows that the basic usage of where() has linear scaling.

I hope this is as much a relief to others as it is to me.

Thanks,

Allan