Subject: Re: Summing an array.
Posted by Wox on Mon, 28 Jul 2008 17:35:20 GMT
View Forum Message <> Reply to Message

On Mon, 28 Jul 2008 08:36:02 -0700 (PDT), "crazywhiteboy311@gmail.com"
<crazywhiteboy311@gmail.com> wrote:

> Hello all,
>
> I've got a little problem I've been stuck on for a few days now, and I
> was hoping to get at least a little guidance to a viable solution. My
> issue involves the summation of an array in two ways. The array is the
> used as the output of model plasma measurement device to express the
> number of particle strikes at a specific position at a certain energy-
> per-charge value:
>
> X-Coordinate   Y-Coordinate   Count   Energy-per-Charge
>
> X and Y are both integers that range from 0 to 63, Count is a floating
> point value, and Energy-per-Charge is a floating point value
> logarithmically spaced from 0.1 to ~13 in 64 bins.
>
> The goal is to produce two arrays from this (that need to be used for
> two pre-made plotting programs). The first array is a 64x64 element
> array that contains the sum of all counts at a certain coordinate.
> Ie :
>
> Counts at (1,1)   Counts at (1,2)   Counts at (1,3) ....
> Counts at (2,1)
> ...
>
> The second is to produce an array that is simply a list of all the
> counts at a single Energy-per-Chage value. ie:
> Energy-per-Charge   Counts
>
> Now I have been able to do this in the past with loop operations, but
> considering IDL's 'penalty' for using loops, it runs quite a bit
> slower then needed. Specifically, it has to deal with arrays that have
> at max 1.5 million pairs of XY coordinate, count, and Energy-per-
> Charge value, and it needs to do this operation between 20 and 50
> times per execution (number of time steps used). Where I've been
> getting stuck is switching this to an array based method which I hope
> to be significantly faster (hopeing for an order of magnitude, but
> cutting it down by a factor of just 3 would be great). Any advice that
> can be offered would be appreciated. Thanks for your time, and for
> reading this lengthy post.
>
> Aron

The code below should get you going.

```
IDL> .r test
IDL> test




function chunktotal,x,n,_REF_EXTRA=extra
; Cumulative sum in chunks: e.g. x=[1,1,2,3],n=[3,0,1] =>
return,[4,0,3]

nx=n_elements(x)
nn=n_elements(n)

 h=histogram(total(n,/CUMULATIVE,/pres)-1,/BINSIZE,MIN=0,REVE RSE_INDICES=ri)

sum=replicate(x[0]*0,nx,nn)
nh=n_elements(h)
sum[indgen(nh),ri[0:nh-1]-ri[0]]=1
sum*=rebin(x,nx,nn,/sample)

return,total(sum,1,_EXTRA=extra)
end;function chunktotal

pro test

X=[1,2,3,1,3]
Y=[0,2,3,0,3]
counts=[10,20,30,40.,50]
EpC=[10.,10.,10.,10.,11]

; First array
arr1=fltarr(64,64)
i=X+64*Y

h=histogram(i,binsize=1,omin=omin)
counts2=chunktotal(counts[sort(i)],h,/pres)
arr1[omin+indgen(n_elements(h))]=counts2

; Second array
binsize=0.1
h=histogram(EpC,binsize=binsize,omin=omin,rev=R)
ind=where(h ne 0,ct)

EpC2=omin+binsize*findgen(ct)
for i=0,ct-1 do print,EpC2[i],counts[R[R[ind[i]] : R[ind[i]+1]-1]]
```

end