Subject: Re: Array operation question
Posted by K. Bowman on Fri, 07 Feb 2003 14:29:28 GMT
View Forum Message <> Reply to Message

In article <b207kmenf1@news.ox.ac.uk>,
Edd Edmondson <eddedmondson@hotmail.com> wrote:

> It's yet another question about how to get an efficient operation on an
> array:
>
> I have one array
> q=[num1,num2,num3,num4]
> and an array
> r=[[num1a,num1b,num1c...],[num2a,num2b..],[num3a...],[num4a..]]
> and I want to find w=r-q such that
> w=[[num1a-num1,num1b-num1,num1c-num1...],[num2a-num2..],[num3a- num3..]..]
>
> Is there an efficient way of doing it without expanding q so that it is
> the same dimension as r? That'd be very expensive in terms of memory for
> me, unfortunately. I could loop over the 4 elements of q and r and do that
> seperately but I'd quite like to eliminate that last loop.
>
> I've tried various things but all fall victim to the 'feature' mentioned
> earlier that IDL will make the result have the dimensions of the smaller
> array.

The first rule of thumb of optimization is "optimize the innermost
loop", so

FOR j = 0, nj-1 DO w[0,j] = r[*,j]-q[j]

will be pretty efficient if the first dimension or r is large. (Note
that the zero on the lhs is important for efficiency.)

Ken Bowman

Subject: Re: Array operation question
Posted by Pavel A. Romashkin on Fri, 07 Feb 2003 17:31:26 GMT
View Forum Message <> Reply to Message

The way you present it, you still need spare RAM twice the size of R
because W is the same in size to calculate
w = r - q
Therefore, I see no problem with expanding Q. BUT - it may be slower
than the loop.

q = findgen(200000)

```
r = findgen(200000, 500)
s = size(r)
w = r - q[lindgen(s[1]*s[2]) mod s[1]]
;or
w = r - rebin(q, 200000, 500)
```

Double the size of R fits in RAM but exhausts it, and no additional sizeable allocations are possible. But this is what W and R would take. Rebin does it, while subscripting fails for the lack of RAM. Apparently, linearizing the 2D array is more taxing than Rebin.
I am not sure that with this size arrays Rebin is any faster than a loop offered by Ken. For me, allocating this much RAM is often slower that looping through smaller chunks or it.

Now, here is something really curious that I just found out while looking for a better way to do this.
If you want to take up, say, 80% or RAM with R, I tried to set R up as a pointer array like

```
r = ptrarr(1500, /allocate)
q = findgen(200000)
for i = 0, 1499 do *R[i] = findgen(200000)
for i = 0, 1499 do *R[i] = temporary(*R[i]) - q
```

It also turned out, to my great surprise, that while I can only allocate r = findgen(200000, 1000) and run out of RAM (1.12 Gb used) after that, I can go as high as
R = ptrarr(1500) - 50% higher - and put a 200,000 FLOAT in each and still don't run out of ram (1.13 Gb used). The entire process with pointers is twice as fast as with REBIN.
Cheers,
Pavel

Edd Edmondson wrote:
>
> It's yet another question about how to get an efficient operation on an
> array:
>
> I have one array
> q=[num1,num2,num3,num4]
> and an array
> r=[[num1a,num1b,num1c...],[num2a,num2b..],[num3a...],[num4a..]]
> and I want to find w=r-q such that
> w=[[num1a-num1,num1b-num1,num1c-num1...],[num2a-num2..],[num3a- num3..]..]
>
> Is there an efficient way of doing it without expanding q so that it is
> the same dimension as r? That'd be very expensive in terms of memory for
> me, unfortunately. I could loop over the 4 elements of q and r and do that

> seperately but I'd quite like to eliminate that last loop.
>
> I've tried various things but all fall victim to the 'feature' mentioned
> earlier that IDL will make the result have the dimensions of the smaller
> array.
>
> --
> Edd
