Subject: Re: Vectorization question
Posted by Liam E. Gumley on Mon, 11 Sep 2000 22:06:11 GMT
View Forum Message <> Reply to Message

"Liam E. Gumley" wrote:

```
>
> Given the following arrays
>
> a = intarr(10)
> x = [2, 2, 2, 3, 3, 4]
> b = [1, 3, 4, 2, 1, 8]
>
> How would I vectorize the following operation
>
> for i = 0, n_elements(x) - 1 do a[x[i]] = a[x[i]] + b[i]
>
> To achieve this result
>
> print, a, format='(10i4)'
>    0   0   8   3   8   0   0   0   0   0
>
> In the real-world case where this occurs, I need to repeat this kind of
> operation several hundred times, where the size of 'a' is around
> 1,000,000 and the size of 'x' is around 100,000 ('a' and 'b' are float
> type in the real-world case).
```

Here's one solution:

```
a = intarr(10)
x = [2, 2, 2, 3, 3, 4]
b = [1, 3, 4, 2, 1, 8]
tmp = intarr(n_elements(a), n_elements(x))
tmp[x, indgen(n_elements(x))] = b
print, a + total(tmp, 2), format='(10i4)'
   0   0   8   3   8   0   0   0   0   0
```

It's a bit memory hungry, but it's fast. Any other offers?

Cheers,
Liam.

Subject: Re: Vectorization question
Posted by promashkin on Tue, 12 Sep 2000 19:34:51 GMT
View Forum Message <> Reply to Message

Hi Liam,
I came up with a different approach. For short arrays in a loop, it is 3

times slower than your method, Liam. However, with A being 1,000,000 and
both X and B 100,000 elements long, your method could not allocate
memory on my machine, while mine ran in 0.07 s. Check out the code
below. The loops were put in to get runtime extimates.
I run 5.3 on PowerMac G4, 192 MB total RAM, 64 MB allocated to IDL.
Cheers,
Pavel

```
pro pavel, a, b, x
out = a
start = systime(1)
;for i =0, 10000 do begin
ind = x[uniq(x, sort(x))]
loc = value_locate(x, ind)
sum_b = total(b, /cumulative)
res = [0, sum_b[loc], 0]
a_values = (res-shift(res, 1))[1:n_elements(res)-2]
out[ind] = a_values
;endfor
print, systime(1) - start
;print, out, format='(10i4)'
end

pro liam, a, b, x
out = a
start = systime(1)
;for i =0, 10000 do begin
tmp = intarr(n_elements(a), n_elements(x))
tmp[x, indgen(n_elements(x))] = b
out = a + total(tmp, 2)
;endfor
print, systime(1) - start
;print,  out, format='(10i4)'
end
```

Subject: Re: Vectorization question
Posted by Craig Markwardt on Sun, 17 Sep 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Pavel Romashkin <promashkin@cmdl.noaa.gov> writes:

> Hi Liam,
> I came up with a different approach. For short arrays in a loop, it is 3
> times slower than your method, Liam. However, with A being 1,000,000 and
> both X and B 100,000 elements long, your method could not allocate
> memory on my machine, while mine ran in 0.07 s. Check out the code
> below. The loops were put in to get runtime extimates.

> I run 5.3 on PowerMac G4, 192 MB total RAM, 64 MB allocated to IDL.
> Cheers,
> Pavel

Here is my belated entry. Getting the repeats right is the trick. My technique does use HISTOGRAM, but is thinking outside the box a little bit by iterating over the number of hits in the histogram rather than the reverse index array.

I use this technique some in my own work and it's fairly fast. I use it to place values known at semi-regularly spaced times onto a regular grid. Unfortunately I can't use these new-fangled VALUE_LOCATE() or TOTAL(...,/CUMULATIVE) functions since I am staying compatible with earlier versions of IDL. We don't even have IDL 5.3 yet, so I can't even test against any others. Pavel, can you compare? :-)

```
n = n_elements(a)
hh = histogram(x, min=0, max=n-1, reverse=rr)
wh = where(hh GT 0) & mx = max(hh(wh), min=mn)
for i = mn, mx do begin
  wh = wh(where(hh(wh) GE i, ct))   ;; Get X cells with GE i entries
  a(wh) = a(wh) + b(rr(rr(wh)+i-1)) ;; Add into the total
endfor
```

Below are my array setups, and the functions I used. The pavel function is slightly modified for practicality reasons.

Craig

```
a = lonarr(1000000)
b = long(randomu(seed,100000)*1000000)
x = long(randomu(seed,100000)*1000000)

pro craig, a, b, x, iter=iter
start = systime(1)
if n_elements(iter) EQ 0 then iter = 1
start = systime(1)
for j = 0, iter-1 do begin
n = n_elements(a)
hh = histogram(x, min=0, max=n-1, reverse=rr)
wh = where(hh GT 0) & mx = max(hh(wh), min=mn)
for i = mn, mx do begin
  wh = wh(where(hh(wh) GE i, ct))
  a(wh) = a(wh) + b(rr(rr(wh)+i-1))
endfor
endfor
print, systime(1) - start
end
```

```
pro pavel, a, b, x, iter=iter
start = systime(1)
if n_elements(iter) EQ 0 then iter = 1
out = a
start = systime(1)
for i = 0, iter-1 do begin
ind = x[uniq(x, sort(x))]
loc = value_locate(x, ind)
sum_b = total(b, /cumulative)
res = [0, sum_b[loc], 0]
a_values = (res-shift(res, 1))[1:n_elements(res)-2]
out[ind] = a_values
endfor
print, systime(1) - start
end

pro liam, a, b, x, iter=iter
start = systime(1)
if n_elements(iter) EQ 0 then iter = 1
out = a
start = systime(1)
for i = 0, iter-1 do begin
tmp = intarr(n_elements(a), n_elements(x))
tmp[x, indgen(n_elements(x))] = b
out = a + total(tmp, 2)
endfor
print, systime(1) - start
end
```

```
>
>  pro pavel, a, b, x
>  out = a
>  start = systime(1)
>  ;for i =0, 10000 do begin
>  ind = x[uniq(x, sort(x))]
>  loc = value_locate(x, ind)
>  sum_b = total(b, /cumulative)
>  res = [0, sum_b[loc], 0]
>  a_values = (res-shift(res, 1))[1:n_elements(res)-2]
>  out[ind] = a_values
>  ;endfor
>  print, systime(1) - start
>  ;print, out, format='(10i4)'
>  end
```

```
>
>  pro liam, a, b, x
>  out = a
>  start = systime(1)
>  ;for i =0, 10000 do begin
>  tmp = intarr(n_elements(a), n_elements(x))
>  tmp[x, indgen(n_elements(x))] = b
>  out = a + total(tmp, 2)
>  ;endfor
>  print, systime(1) - start
>  ;print,  out, format='(10i4)'
>  end
```

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------