Subject: Re: FOR loops removal
Posted by loebasboy on Thu, 21 Aug 2008 07:59:33 GMT
View Forum Message <> Reply to Message

On Aug 20, 4:07 pm, Jeremy Bailin <astroco...@gmail.com> wrote:
>> So vectorization comes down to instead of repeating an action per
>> element in matrix, putting all elements on the right spot in a matrix
>> and doing the action on the matrix, right?
>
> Yup, that sums it up pretty well!
>
> -Jeremy.

So I tested the new finetuned program on the standard image and
instead of a calculated 15 hour time profit it has become almost 20,5
hour time profit. The program takes now 2.15 hours instead of 22.5
hours. That is a major improvement (< 10x), so thanks for all the info
allready. So I started out with even more improvements, I haven't
found any vectorisation possibilities yet though. I tried to fasten
the following code:

```
n = 20
size = 2*n+1
array = randomn(seed, size)
array[0] = 0
array[5] = 0
array[10] = 0
array[20] = 0
array[size-2] = 0
array[size-1] = 0

      FOR x = 1, size-2 DO BEGIN
       IF (array[x] EQ 0) THEN BEGIN
        IF ((array[x-1] LE 2) AND (array[x+1] LE 2)) THEN
BEGIN
         array[x] = 2
        ENDIF ELSE BEGIN
         IF ((array[x-1] GE 2) AND (array[x+1] GE 2)) THEN
BEGIN
          array[x] = -2
         ENDIF
        ENDELSE
       ENDIF
      ENDFOR
```

So I figured that if i use the WHERE function to find where the array
equals 0, and then use a FOR loop that only goes trough the indices
that the WHERE function has found. So If you consider the WHERE

function to be much faster than the FOR loop, you could expect that
the second FOR loop would be faster or equally fast than the first FOR
loop. The code for the second FOR loop goes like this (some other
extra IF functions are needed for special cases like a zero as a first
element, last element or no zero at all):

```
    zeroindex = where (array EQ 0,m)
    IF (zeroindex[0] NE -1) THEN BEGIN
     IF (zeroindex[0] EQ 0) THEN k = 1 ELSE k = 0
     IF (zeroindex[m-1] EQ size-1) THEN l = 2 ELSE l = 1
     FOR i= k, m-l DO BEGIN
      IF ((array[zeroindex[i]-1] LE 2) AND (array[zeroindex[i]+1]
LE 2)) THEN BEGIN
       array[zeroindex[i]] = 2
      ENDIF ELSE BEGIN
       IF ((array[zeroindex[i]-1] GE 2) AND (array[zeroindex[i]
+1] GE 2)) THEN BEGIN
        array[zeroindex[i]] = -2
       ENDIF
      ENDELSE
     ENDFOR
```

you could hear me coming from afar ofcourse ;) . The second FOR loop
doesn't go faster, at all, with the variables set as above and the two
loops repeated for 50000 times. The first loop takes 0.304 s and the
second one 0.337 s. Only if the n-value is made larger than 25 the
second loop starts to go faster. I checked out profiler to check if
the WHERE function makes up for this slowing down this bit of
programming and ofcourse it does, the difference in time is 0.033s
while the WHERE function takes up 0.066s. So the second loop goes
faster but the use of the WHERE function slows the whole program down.
This is some nice checking out ofcourse but it doesn't help me getting
any further. Is there a faster alternative of the WHERE function? Or
did I reach the limit in finetuning here? :)