Subject: Re: How to speed up code which checks lots of values of an array
Posted by ben.bighair on Fri, 14 Jan 2011 00:59:50 GMT
View Forum Message <> Reply to Message

On 1/13/11 11:06 AM, Robin Wilson wrote:
> Hi all,
>
> I've been writing some code which checks that all of the values in one
> half of an array are less than a threshold, and all of the values on the
> right half of an array are greater than a threshold. I've written the
> code below, which seems to be a fairly clean way of writing it, and
> works for any length of array (which is important for my use of it).
>
> However, it's not very fast. I wondered if it might be able to be sped
> up using some sort of magic like the Histogram command... Any ideas?
>
> ---CODE---
> ; Assume we're given a 1D array called line with our values in it
>
> len = N_ELEMENTS(line)
>
> section_len = (len - 1) / 2
>
> ; Get the left-hand half of the array
> LHS = line[0:section_len - 1]
> RHS = line[section_len + 1: len - 1]
>
> res = WHERE(LHS GT 180, LHS_count)
> res = WHERE(RHS LT 180, RHS_count)
>
> IF LHS_count EQ 0 AND RHS_count EQ 0 THEN BEGIN
> ; Do stuff
> ENDIF
>
> ---END CODE---
>

Hi,

I vaguely recall a discussion much like this from years ago - one of the
wizards (Craig? JD?) suggested that using TOTAL would reduce the number
of comparison scans required. For example, you have the comparison scan
of the line inside the WHERE (or MAX) (LHS GE threshold) and then the
scan by WHERE (or MAX) itself. Using TOTAL still requires two
traversals of the line: one by (LHS GE threshold) and one by TOTAL, but
TOTAL doesn't have the overhead of doing a comparison as it scans.

It seems to pan out pretty well in the test code below.

Cheers,
Ben

```
IDL> .comp line_test
% Compiled module: MAX_TEST.
% Compiled module: TOTAL_TEST.
% Compiled module: WHERE_TEST.
% Compiled module: LINE_TEST.
IDL> line_test
MAX_TEST
  result =    1   1
  elapsed =       1.0753469

TOTAL_TEST
  result =                  2489                  2472
  elapsed =      0.82091904

WHERE_TEST
  result =        2489        2472
  elapsed =       1.3972421
```

```
;----- START HERE

FUNCTION max_test, line, threshold

   len = N_ELEMENTS(line)

   section_len = (len - 1) / 2

   LHS = line[0:section_len - 1]
   RHS = line[section_len + 1: len - 1]

   x = [MAX(LHS GE threshold), MAX(RHS LT threshold)]

   return, x

END;


FUNCTION total_test, line, threshold


   len = N_ELEMENTS(line)

   section_len = (len - 1) / 2
   LHS = line[0:section_len - 1]
   RHS = line[section_len + 1: len - 1]

   x = [TOTAL(LHS GE threshold, /INT), TOTAL(RHS LT threshold, /INT) ]

   Return, x

END


FUNCTION where_test, line, threshold

   len = N_ELEMENTS(line)

   section_len = (len - 1) / 2

   ; Get the left-hand half of the array
   LHS = line[0:section_len - 1]
   RHS = line[section_len + 1: len - 1]

   res = WHERE(LHS GE threshold, LHS_count)
   res = WHERE(RHS LT threshold, RHS_count)

   x = [LHS_count, RHS_count]

   return, x

END


PRO line_test

   N = 10000
   thresh = 180
   x = RANDOMU(s, N) * (thresh * 2)

   t0 = Systime(/SEC)
   for i = 0L, N do  mx = max_test(x, thresh)
   dt_mx = Systime(/SEC) - t0

   t0 = Systime(/SEC)
   for i = 0L, N do  tot = total_test(x, thresh)
   dt_tot = Systime(/SEC) - t0

   t0 = Systime(/SEC)
   for i = 0L, N do wh = where_test(x, thresh)
   dt_wh = Systime(/SEC) - t0
```

```
    PRINT, "MAX_TEST"
    PRINT, " result = ", mx
    PRINT, " elapsed = ", dt_mx
    PRINT, " "
    PRINT, "TOTAL_TEST"
    PRINT, " result = ", tot
    PRINT, " elapsed = ", dt_tot
    PRINT, " "
    PRINT, "WHERE_TEST"
    PRINT, " result = ", wh
    PRINT, " elapsed = ", dt_wh


END;

;---- END HERE
```