Subject: Re: REPLICATE question
Posted by steinhh on Thu, 17 Oct 1996 07:00:00 GMT
View Forum Message <> Reply to Message

In article <540h21$eu5@danberg.llnl.gov>, dan@danberg.llnl.gov (Dan Bergmann) writes:
|>
|> In article <32636705.41C6@jmc-luni.u-bordeaux2.fr>, Mario Noyon
<mnoyon@jmc-luni.u-bordeaux2.fr> writes:
|> |> I would like to divide an arr1(n,n,m) by an arr2(n,n) and be able to
|> |> obtain an array(n,n,m). I suppose there is a way to avoid the for
|> |> statements.
|> |> I had the idea to transform my arr2(n,n) in an arr2(n,n,m) where the
|> |> m-elements are all the same but REPLICATE does not work with the arrays.
|> |>
|> |> Does some-one have an idea that could help me?
|> |>
|> |> Thanks.
|>
|> I would try two ways and see which is faster
|>
|>   for i=0,m-1 do array(*,*,i) = arr1(*,*,i) / arr2(*,*)
|>
|>   array = arr1 / rebin(reform(arr2,n,n,1),n,n,m)

Well, several methods were suggested, and if I got them all
right, here are some timing results for you (if anybody finds
errors or have complaints about the implementations, please say
so, and I'll rerun the tests...on the alphaserver at least). Timing
programs are included at the end of this message.

Method 0 is based on the first one of the above, just adding a MAKE_ARRAY
call to get the original ARRAY.

Method 1 is the second one of the above, adding a few /OVERWRITE and
/SAMPLE keywords (see method 5 for results without these).

Method 2 is the one using a one-dimensional index array as suggested
by Paul Ricchiazzi, with some alterations to make it work.

Method 3 is the same as #2, except with a MAKE_ARRAY call instead
of copying.

Method 4 is the "trick of the trailing zero subscript", suggested by
James Tappin. I wouldn't have guessed right away that this produced
the right result!!

Method 5 (See method 1)

The timing results for IDL v 4.0 and 3.6.1c are presented below
for an AlphaServer (4CPUs, large cache, one other process competing
for the cache space) and an AlphaStation.

Lo and behold, version 3.6.1c is (as I've pointed out in earlier
tests) *faster* than version 4.0, by about 10% in most cases
(though exceptions occur, e.g., method 0 for the AlphaStation).

. and the winning method is method 1.

Timing results:

IDL 4.0 running on alphaserver
Timing starts with n=100, m=100
    8.1483519 seconds for method   0
    2.7696639 seconds for method   1
    21.355264 seconds for method   2
    20.596912 seconds for method   3
    4.1581120 seconds for method   4
    3.0858879 seconds for method   5

IDL 3.6.1c running on alphaserver
Timing starts with n=100, m=100
    7.1977280 seconds for method   0
    2.7202880 seconds for method   1
    18.001952 seconds for method   2
    17.213344 seconds for method   3
    3.0156159 seconds for method   4
    2.8003199 seconds for method   5

IDL 4.0 running on alphastation
Timing starts with n=100, m=100
    16.392998 seconds for method   0
    5.8692130 seconds for method   1
    44.179048 seconds for method   2
    39.757835 seconds for method   3
    8.7850729 seconds for method   4
    6.3661391 seconds for method   5

IDL 3.6.1c running on alphastation
Timing starts with n=100, m=100
    18.452170 seconds for method   0
    5.9062841 seconds for method   1
    34.888986 seconds for method   2
    33.091774 seconds for method   3
    4.8394741 seconds for method   4
    5.3670160 seconds for method   5

```
; Testing program

PRO meth_0,arr1,arr2,array,iter

  ;; dan@danberg.llnl.gov (Dan Bergmann)

  sa = size(arr1)

  FOR i = 1,iter DO BEGIN
    array = 0

    array = make_array(size=sa,/nozero)

    FOR j=0,sa(3)-1 DO array(*,*,j) = arr1(*,*,j)/arr2(*,*)
  END
END


PRO meth_1,arr1,arr2,array,iter

  ;; dan@danberg.llnl.gov (Dan Bergmann)

  ;; Additional keywords by SVHH.

  sa = size(arr1)

  FOR i = 1,iter DO BEGIN
    array = 0

    array = arr1/rebin(reform(arr2,sa(1),sa(2),1,/overwrite),$
                       sa(1),sa(2),sa(3),/sample)

    arr2 = reform(arr2,sa(1),sa(2),/overwrite)
  END

END

PRO meth_2,arr1,arr2,array,iter
  ;; paul@skua.s2k.ucsb.edu (Paul Ricchiazzi)
  ;; Corrected by an@danberg.llnl.gov (Dan Bergmann)
  ;;
  ;; Modified  array    = arr1(ii)....
  ;; to        array(*) = arr1(ii)...
  sa = size(arr1)

  FOR i = 1,iter DO BEGIN
    array = 0
```

```
    ;; Well, we have to make the array somehow...
    array = arr1

    ii = lindgen(sa(1)*sa(2)*sa(3)) MOD (sa(1)*sa(2))
    array(*) = arr1(ii)/arr2(ii)
  END
END

PRO meth_3,arr1,arr2,array,iter
  ;; paul@skua.s2k.ucsb.edu (Paul Ricchiazzi)
  ;; Corrected by an@danberg.llnl.gov (Dan Bergmann)
  ;;
  ;; Modified  array    = arr1(ii)....
  ;; to        array(*) = arr1(ii)...
  ;;
  ;; Added make_array call to produce array in the first place...
  ;;

  sa = size(arr1)

  FOR i = 1,iter DO BEGIN
    array = 0
    ;; Well, we have to make the array somehow...
    array = make_array(size=sa,/nozero)

    ii = lindgen(sa(1)*sa(2)*sa(3)) MOD (sa(1)*sa(2))
    array(*) = arr1(ii)/arr2(ii)
  END
END

PRO meth_4,arr1,arr2,array,iter
  ;; sjt@xuna.sr.bham.ac.uk (James Tappin)

  sa = size(arr1)

  FOR i = 1,iter DO BEGIN
    array = 0
    array = arr1/arr2(*,*,intarr(sa(3)))
  END
END

PRO meth_5,arr1,arr2,array,iter

  ;; dan@danberg.llnl.gov (Dan Bergmann)

  ;; Additional keywords by SVHH.

  sa = size(arr1)
```

```
  FOR i = 1,iter DO BEGIN
    array = 0

    array = arr1/rebin(reform(arr2,sa(1),sa(2),1),$
                       sa(1),sa(2),sa(3))

    ;; Not necessary when no /overwrite is done in previous reform
    ;; arr2 = reform(arr2,sa(1),sa(2),/overwrite)
  END

END

PRO timetest,n,m,iter

  IF n_elements(n) EQ 0 THEN n = 100
  IF n_elements(m) EQ 0 THEN m = 100
  IF n_elements(iter) EQ 0 THEN iter = 10

  arr1 = findgen(n,n,m)
  FOR i = 1,m-1 DO arr1(*,*,i) = arr1(*,*,0)

  arr2 = findgen(n,n) + 1.0 ;; Avoid division by zero

  ;; Just warming up...

  IF getenv("HOST") EQ "achernar" THEN type = "server" $
  ELSE IF getenv("HOST") EQ "amon" THEN type = " DECstation 5000" $
  ELSE type = "station"

  print,"IDL "+!version.release+" running on "+$
    !version.arch+type

  meth_0,arr1,arr2,array0,2
  meth_1,arr1,arr2,array1,2
  meth_2,arr1,arr2,array2,2
  meth_3,arr1,arr2,array3,2
  meth_4,arr1,arr2,array4,2
  meth_5,arr1,arr2,array5,2

  print,"Timing starts with n="+strtrim(string(n),1)+",   m="+$
    strtrim(string(m),1)

  FOR meth = 0,5 DO BEGIN
    t = systime(1)
    call_procedure,"meth_"+strtrim(string(meth),1),$
      arr1,arr2,array,iter
    print,systime(1)-t," seconds for method ",meth
```

```
  END

END
```