Subject: Re: Dealing with Large data arrays, reducing memory and ASSOC
Posted by Kenneth Bowman on Thu, 14 Jun 2007 13:08:44 GMT
View Forum Message <> Reply to Message

In article <1181824433.145388.26020@d30g2000prg.googlegroups.com>,
 Ambrosia_Everlovely <ambrosia_everlovely@hotmail.com> wrote:

> Hi,
> I have a fairly large datacube, DC(x,y,t)=DC(512,512,2048) and I want
> to perform an FFT in the t direction. Now I can do,
> FFTDC=fft(DC,-1,dim=3) which takes an excessive amount of memory (19 G
> + 50 G virtual) and slows the whole system down.
> Since this must be a fairly common practice amongst astronomers, can
> anyone provide - or link to - a small IDL algorithm which will allow
> me to use ASSOC or reduce the memory in some way? I have also tried
> TEMPORARY, but this doesn't seem to help at all.
>
> Thankyou!!!!

I would just do it in slices

dct = COMPLEXARR(512,512,2048)
FOR j = 0, 511 do dct[*,j,*] = FFT(REFORM(dc[*,j,*]), -1, DIM = 2)

This does access memory in nearly the worst possible way. If you are going
to be doing this a lot, you might want to consider rearranging the
data so that t is the first dimension

dct = COMPLEXARR(2048,512,512)
FOR k = 0, 255 D0 xt[0,0,k] = FFT(REFORM(x[*,*,k]), -1, DIM = 1)

Ken Bowman

Subject: Re: Dealing with Large data arrays, reducing memory and ASSOC
Posted by bill.dman on Thu, 14 Jun 2007 13:41:26 GMT
View Forum Message <> Reply to Message

On Jun 14, 8:33 am, Ambrosia_Everlovely
<ambrosia_everlov...@hotmail.com> wrote:
> Hi,
> I have a fairly large datacube, DC(x,y,t)=DC(512,512,2048) and I want
> to perform an FFT in the t direction. Now I can do,
> FFTDC=fft(DC,-1,dim=3) which takes an excessive amount of memory (19 G
> + 50 G virtual) and slows the whole system down.
> Since this must be a fairly common practice amongst astronomers, can
> anyone provide - or link to - a small IDL algorithm which will allow
> me to use ASSOC or reduce the memory in some way? I have also tried

> TEMPORARY, but this doesn't seem to help at all.
>
> Thankyou!!!!

Assuming you are using single precision, you can limit memory needed
to about 6GB with

fftdc = complexarr(512,512,2048)
for i=0,511 do for j=0,511 do fftdc[i,j,0] = fft(dc[i,j,*],-1)

this should help if your machine has more than 6GB for you to use.

Subject: Re: Dealing with Large data arrays, reducing memory and ASSOC
Posted by Haje Korth on Thu, 14 Jun 2007 16:18:25 GMT
View Forum Message <> Reply to Message

Just curious: Have you tried my fftw3 dlm available at the ittvis codebank?
does it work any better? Haje

"Ambrosia_Everlovely" <ambrosia_everlovely@hotmail.com> wrote in message
news:1181824433.145388.26020@d30g2000prg.googlegroups.com...
> Hi,
> I have a fairly large datacube, DC(x,y,t)=DC(512,512,2048) and I want
> to perform an FFT in the t direction. Now I can do,
> FFTDC=fft(DC,-1,dim=3) which takes an excessive amount of memory (19 G
> + 50 G virtual) and slows the whole system down.
> Since this must be a fairly common practice amongst astronomers, can
> anyone provide - or link to - a small IDL algorithm which will allow
> me to use ASSOC or reduce the memory in some way? I have also tried
> TEMPORARY, but this doesn't seem to help at all.
>
> Thankyou!!!!
>

Subject: Re: Dealing with Large data arrays, reducing memory and ASSOC
Posted by JD Smith on Thu, 14 Jun 2007 16:32:00 GMT
View Forum Message <> Reply to Message

On Thu, 14 Jun 2007 08:08:44 -0500, Kenneth Bowman wrote:

> In article <1181824433.145388.26020@d30g2000prg.googlegroups.com>,
> Ambrosia_Everlovely <ambrosia_everlovely@hotmail.com> wrote:
>
>> [quoted text muted]

>
> I would just do it in slices
>
> dct = COMPLEXARR(512,512,2048)
> FOR j = 0, 511 do dct[*,j,*] = FFT(REFORM(dc[*,j,*]), -1, DIM = 2)
>
> This does access memory in nearly the worst possible way. If you are
> going to be doing this a lot, you might want to consider rearranging the
> data so that t is the first dimension
>
> dct = COMPLEXARR(2048,512,512)
> FOR k = 0, 255 D0 xt[0,0,k] = FFT(REFORM(x[*,*,k]), -1, DIM = 1)

I'd be interested to hear whether this "in order" type of array
re-arrangement results in a real speedup. I had always assumed this
is true, but in recent testing on a very different problem, found
little or no gain, to my surprise.

JD

Subject: Re: Dealing with Large data arrays, reducing memory and ASSOC
Posted by Kenneth Bowman on Thu, 14 Jun 2007 18:50:33 GMT
View Forum Message <> Reply to Message

In article <pan.2007.06.14.16.32.00.193280@as.arizona.edu>,
JD Smith <jdsmith@as.arizona.edu> wrote:

> On Thu, 14 Jun 2007 08:08:44 -0500, Kenneth Bowman wrote:
>
>> In article <1181824433.145388.26020@d30g2000prg.googlegroups.com>,
>> Ambrosia_Everlovely <ambrosia_everlovely@hotmail.com> wrote:
>>
>>> [quoted text muted]
>>
>> I would just do it in slices
>>
>> dct = COMPLEXARR(512,512,2048)
>> FOR j = 0, 511 do dct[*,j,*] = FFT(REFORM(dc[*,j,*]), -1, DIM = 2)
>>
>> This does access memory in nearly the worst possible way. If you are
>> going to be doing this a lot, you might want to consider rearranging the
>> data so that t is the first dimension
>>
>> dct = COMPLEXARR(2048,512,512)
>> FOR k = 0, 255 D0 xt[0,0,k] = FFT(REFORM(x[*,*,k]), -1, DIM = 1)
>
> I'd be interested to hear whether this "in order" type of array

> re-arrangement results in a real speedup. I had always assumed this
> is true, but in recent testing on a very different problem, found
> little or no gain, to my surprise.
>
> JD

Here is a quick test that only measures the FFT time:

```
nx =  512
ny =  512
nz = 2048

x  = FINDGEN(nx, ny, nz)
xt = COMPLEXARR(nx, ny, nz)

t = SYSTIME(/SECONDS)

FOR j = 0, ny-1 DO xt[*,j,*] = FFT(REFORM(x[*,j,*]), -1, DIM = 2)

PRINT, 'Time for FFT of 3rd dimension : ', t - SYSTIME(/SECONDS)

x  = REFORM(x,  nz, nx, ny)
xt = REFORM(xt, nz, nx, ny)

t = SYSTIME(/SECONDS)

FOR k = 0, ny-1 DO xt[0,0,k] = FFT(REFORM(x[*,*,k]), -1, DIM = 1)

PRINT, 'Time for FFT of 1st dimension : ', t - SYSTIME(/SECONDS)
```

I ran it on our new dual quad-core Xeon with 16 GB of memory and got this

```
IDL> @fft_3_test
Time for FFT of 3rd dimension :       -127.68938
Time for FFT of 1st dimension :       -27.563090
```

On my Mac G5 for smaller arrays (256 x 256 x 512) I get this

```
IDL> @fft_3_test
Time for FFT of 3rd dimension :       -4.6950710
Time for FFT of 1st dimension :       -2.5009661
```

I think is a fact of life with cache systems that out-of-order

memory access will cause some penalty.

Ken

Subject: Re: Dealing with Large data arrays, reducing memory and ASSOC
Posted by Kenneth Bowman on Thu, 14 Jun 2007 18:52:07 GMT
View Forum Message <> Reply to Message

In article <1181828486.257277.182530@q19g2000prn.googlegroups.com>,
bill.dman@gmail.com wrote:

> On Jun 14, 8:33 am, Ambrosia_Everlovely
> <ambrosia_everlov...@hotmail.com> wrote:
>> Hi,
>> I have a fairly large datacube, DC(x,y,t)=DC(512,512,2048) and I want
>> to perform an FFT in the t direction. Now I can do,
>> FFTDC=fft(DC,-1,dim=3) which takes an excessive amount of memory (19 G
>> + 50 G virtual) and slows the whole system down.
>> Since this must be a fairly common practice amongst astronomers, can
>> anyone provide - or link to - a small IDL algorithm which will allow
>> me to use ASSOC or reduce the memory in some way? I have also tried
>> TEMPORARY, but this doesn't seem to help at all.
>>
>> Thankyou!!!!
>
> Assuming you are using single precision, you can limit memory needed
> to about 6GB with
>
> fftdc = complexarr(512,512,2048)
> for i=0,511 do for j=0,511 do fftdc[i,j,0] = fft(dc[i,j,*],-1)
>
> this should help if your machine has more than 6GB for you to use.

I don't think this will work as written. The trick of zero-subscripting
on the LHS of an assignment works for the leading dimensions only.

```
IDL> x = findgen(4,4)
IDL> print, x
     0.00000     1.00000     2.00000     3.00000
     4.00000     5.00000     6.00000     7.00000
     8.00000     9.00000     10.0000     11.0000
     12.0000     13.0000     14.0000     15.0000
IDL> x[0,2] = replicate(99.0, 4)
IDL> print, x
     0.00000     1.00000     2.00000     3.00000
     4.00000     5.00000     6.00000     7.00000
     99.0000     99.0000     99.0000     99.0000
```

12.0000 13.0000 14.0000 15.0000

If you try this with a trailing dimension you get this

IDL> x = findgen(4,4)
IDL> x[2,0] = replicate(99.0, 4)
% Out of range subscript encountered: X.
% Execution halted at: $MAIN$

To make your expression work, you would have to write

fftdc[i,j,*] = fft(dc[i,j,*],-1)

which results in some performance penalty.

Ken Bowman

Subject: Re: Dealing with Large data arrays, reducing memory and ASSOC
Posted by Ambrosia_Everlovely on Fri, 15 Jun 2007 07:26:09 GMT
View Forum Message <> Reply to Message

On Jun 14, 8:52 pm, Kenneth Bowman <k-bow...@tamu.edu> wrote:
> In article <1181828486.257277.182...@q19g2000prn.googlegroups.com>,
>
>
>
> bill.d...@gmail.com wrote:
>> On Jun 14, 8:33 am, Ambrosia_Everlovely
>> <ambrosia_everlov...@hotmail.com> wrote:
>>> Hi,
>>> I have a fairly large datacube, DC(x,y,t)=DC(512,512,2048) and I want
>>> to perform an FFT in the t direction. Now I can do,
>>> FFTDC=fft(DC,-1,dim=3) which takes an excessive amount of memory (19 G
>>> + 50 G virtual) and slows the whole system down.
>>> Since this must be a fairly common practice amongst astronomers, can
>>> anyone provide - or link to - a small IDL algorithm which will allow
>>> me to use ASSOC or reduce the memory in some way? I have also tried
>>> TEMPORARY, but this doesn't seem to help at all.
>
>>> Thankyou!!!!
>
>> Assuming you are using single precision, you can limit memory needed
>> to about 6GB with
>
>> fftdc = complexarr(512,512,2048)
>> for i=0,511 do for j=0,511 do fftdc[i,j,0] = fft(dc[i,j,*],-1)

>
>> this should help if your machine has more than 6GB for you to use.
>
> I don't think this will work as written. The trick of zero-subscripting
> on the LHS of an assignment works for the leading dimensions only.
>
> IDL> x = findgen(4,4)
> IDL> print, x
> 0.00000 1.00000 2.00000 3.00000
> 4.00000 5.00000 6.00000 7.00000
> 8.00000 9.00000 10.0000 11.0000
> 12.0000 13.0000 14.0000 15.0000
> IDL> x[0,2] = replicate(99.0, 4)
> IDL> print, x
> 0.00000 1.00000 2.00000 3.00000
> 4.00000 5.00000 6.00000 7.00000
> 99.0000 99.0000 99.0000 99.0000
> 12.0000 13.0000 14.0000 15.0000
>
> If you try this with a trailing dimension you get this
>
> IDL> x = findgen(4,4)
> IDL> x[2,0] = replicate(99.0, 4)
> % Out of range subscript encountered: X.
> % Execution halted at: $MAIN$
>
> To make your expression work, you would have to write
>
> fftdc[i,j,*] = fft(dc[i,j,*],-1)
>
> which results in some performance penalty.
>
> Ken Bowman

Yes, this is what I have done combined with the TEMPORARY function, it has reduced the memory use a fraction. I was not so concerned with the time it was taking, but the memory it was using (according to sys admin, swapping memory in and out), and thus slowing everyone else down. I have also taken this out of a loop and am calculating it individually - but this is clearly not ideal. It appears that the memory "builds up" with each loop. This may be the real root of my problems - is there a way to "clear" all the variables after each loop? DELVAR only works at the MAINS level.....

You guys have been great, thanks.

Subject: Re: Dealing with Large data arrays, reducing memory and ASSOC
Posted by bill.dman on Fri, 15 Jun 2007 18:40:19 GMT
View Forum Message <> Reply to Message

On Jun 14, 2:52 pm, Kenneth Bowman <k-bow...@tamu.edu> wrote:
> In article <1181828486.257277.182...@q19g2000prn.googlegroups.com>,
>
>
>
> bill.d...@gmail.com wrote:
>> On Jun 14, 8:33 am, Ambrosia_Everlovely
>> <ambrosia_everlov...@hotmail.com> wrote:
>>> Hi,
>>> I have a fairly large datacube, DC(x,y,t)=DC(512,512,2048) and I want
>>> to perform an FFT in the t direction. Now I can do,
>>> FFTDC=fft(DC,-1,dim=3) which takes an excessive amount of memory (19 G
>>> + 50 G virtual) and slows the whole system down.
>>> Since this must be a fairly common practice amongst astronomers, can
>>> anyone provide - or link to - a small IDL algorithm which will allow
>>> me to use ASSOC or reduce the memory in some way? I have also tried
>>> TEMPORARY, but this doesn't seem to help at all.
>
>>> Thankyou!!!!
>
>> Assuming you are using single precision, you can limit memory needed
>> to about 6GB with
>
>> fftdc = complexarr(512,512,2048)
>> for i=0,511 do for j=0,511 do fftdc[i,j,0] = fft(dc[i,j,*],-1)
>
>> this should help if your machine has more than 6GB for you to use.
>
> I don't think this will work as written. The trick of zero-subscripting
> on the LHS of an assignment works for the leading dimensions only.
>
> IDL> x = findgen(4,4)
> IDL> print, x
> 0.00000 1.00000 2.00000 3.00000
> 4.00000 5.00000 6.00000 7.00000
> 8.00000 9.00000 10.0000 11.0000
> 12.0000 13.0000 14.0000 15.0000
> IDL> x[0,2] = replicate(99.0, 4)
> IDL> print, x
> 0.00000 1.00000 2.00000 3.00000
> 4.00000 5.00000 6.00000 7.00000
> 99.0000 99.0000 99.0000 99.0000
> 12.0000 13.0000 14.0000 15.0000
>
> If you try this with a trailing dimension you get this

>
> IDL> x = findgen(4,4)
> IDL> x[2,0] = replicate(99.0, 4)
> % Out of range subscript encountered: X.
> % Execution halted at: $MAIN$
>
> To make your expression work, you would have to write
>
> fftdc[i,j,*] = fft(dc[i,j,*],-1)
>
> which results in some performance penalty.
>
> Ken Bowman

Two issues:

First, it's not exactly true that the base indexing trick works only
for leading dimensions on the LHS. Its a question of shape matching.
So your example works ok with x[2,0] = replicate(99.0, 1, 4).

Second, I agree with you that memory access order can be very
important
for performance. If it is inconvenient to reorganize the data, the
base
indexing trick is still worth while, but I should have more careful
with
the loop nesting order, because (for one smaller test case I just ran)

```
  for i=0,511 do for j=0,511 do fftdc[J,I,0] = fft(dc[J,I,*],-1)
```

ran twice as fast as

```
  for i=0,511 do for j=0,511 do fftdc[I,J,0] = fft(dc[I,J,*],-1)
```

Subject: Re: Dealing with Large data arrays, reducing memory and ASSOC
Posted by Kenneth Bowman on Fri, 15 Jun 2007 19:54:13 GMT
View Forum Message <> Reply to Message

In article <1181932819.315280.237270@q66g2000hsg.googlegroups.com>,
bill.dman@gmail.com wrote:

>> If you try this with a trailing dimension you get this
>>
>> IDL> x = findgen(4,4)
>> IDL> x[2,0] = replicate(99.0, 4)
>> % Out of range subscript encountered: X.
>> % Execution halted at: $MAIN$
>>
>> To make your expression work, you would have to write
>>

>> fftdc[i,j,*] = fft(dc[i,j,*],-1)
>>
>> which results in some performance penalty.
>>
>> Ken Bowman
> Two issues:
>
> First, it's not exactly true that the base indexing trick works only
> for leading dimensions on the LHS. Its a question of shape matching.
> So your example works ok with x[2,0] = replicate(99.0, 1, 4).

Ah, this is good to know. This trick is in a Tech Tip (I can't find it
in the manual), and the explanation is ... perhaps overly succinct.

Ken Bowman
