Subject: Re: large matrix operations
Posted by David Fanning on Mon, 13 Oct 2008 15:04:08 GMT
View Forum Message <> Reply to Message

arp244@gmail.com writes:

> It seems that very large matrix, e.g.1.0e6*1.0e5*100 operations, such
> as multiplication and matrix multiplication, in IDL is very slow.
> Anyone know how to speed up large matrix operations ?

More RAM? 64-bit OS? Prayer?

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: large matrix operations
Posted by Vince Hradil on Mon, 13 Oct 2008 15:08:54 GMT
View Forum Message <> Reply to Message

On Oct 13, 9:44 am, arp...@gmail.com wrote:
> It seems that very large matrix, e.g.1.0e6*1.0e5*100 operations, such
> as multiplication and matrix multiplication, in IDL is very slow.
> Anyone know how to speed up large matrix operations ?

Multiplication should be pretty fast. Matrix multiplication can be
slow if you have to transpose the matrices. This can be sped up by
using MatrixMultiply() instead of # or ##. Perhaps Temporary() can be
used to avoid allocation of extra memory. I think David has the best
ideas, though.

Subject: Re: large matrix operations
Posted by Vince Hradil on Mon, 13 Oct 2008 16:00:38 GMT
View Forum Message <> Reply to Message

On Oct 13, 10:08 am, Vince Hradil <vincehra...@gmail.com> wrote:
> On Oct 13, 9:44 am, arp...@gmail.com wrote:
>
>> It seems that very large matrix, e.g.1.0e6*1.0e5*100 operations, such

>> as multiplication and matrix multiplication, in IDL is very slow.
>> Anyone know how to speed up large matrix operations ?
>
> Multiplication should be pretty fast. Matrix multiplication can be
> slow if you have to transpose the matrices. This can be sped up by
> using MatrixMultiply() instead of # or ##. Perhaps Temporary() can be
> used to avoid allocation of extra memory. I think David has the best
> ideas, though.

oops, that should be MATRIX_MULTIPY()

Subject: Re: large matrix operations
Posted by Vince Hradil on Mon, 13 Oct 2008 17:07:47 GMT
View Forum Message <> Reply to Message

On Oct 13, 11:00 am, Vince Hradil <vincehra...@gmail.com> wrote:
> On Oct 13, 10:08 am, Vince Hradil <vincehra...@gmail.com> wrote:
>
>> On Oct 13, 9:44 am, arp...@gmail.com wrote:
>
>>> It seems that very large matrix, e.g.1.0e6*1.0e5*100 operations, such
>>> as multiplication and matrix multiplication, in IDL is very slow.
>>> Anyone know how to speed up large matrix operations ?
>
>> Multiplication should be pretty fast. Matrix multiplication can be
>> slow if you have to transpose the matrices. This can be sped up by
>> using MatrixMultiply() instead of # or ##. Perhaps Temporary() can be
>> used to avoid allocation of extra memory. I think David has the best
>> ideas, though.
>
> oops, that should be MATRIX_MULTIPY()

So much for that theory...

Hardware:
IDL> print, !version
{ x86 Win32 Windows Microsoft Windows 7.0 Oct 25 2007 32 64}

Results:
% TEST: Allocating A array
% TEST: took 0.14000010 sec
% TEST: Allocating B array
% TEST: took 0.21899986 sec
% TEST: A#B
% TEST: took 40.878000 sec
% TEST: matrix_multipy(A,B)
% TEST: took 42.284000 sec

```
% TEST: transpose(A)#B
% TEST: took 42.645000 sec
% TEST: matrix_multipy(A,B,/atranspose)
% TEST: took 43.449000 sec
% TEST: transpose(temporary(A))#B
% TEST: took 43.387000 sec
% TEST: matrix_multipy(temporary(A),B,/atranspose)
% TEST: took 50.029000 sec
```

Code:

```
.........................
;;;;;;;;;;;;;;;;;;;;;;;;
pro test

  nel = 2000L

  t0 = systime(1)
  message, 'Allocating A array', /info
  a = randomu(seed,[nel,nel])
  message, 'took '+strtrim(systime(1)-t0,2)+' sec', /info

  t0 = systime(1)
  message, 'Allocating B array', /info
  b = randomu(seed,[nel,nel])
  message, 'took '+strtrim(systime(1)-t0,2)+' sec', /info

  t0 = systime(1)
  message, 'A#B', /info
  c = a#b
  message, 'took '+strtrim(systime(1)-t0,2)+' sec', /info

  t0 = systime(1)
  message, 'matrix_multipy(A,B)', /info
  c = matrix_multiply(a,b)
  message, 'took '+strtrim(systime(1)-t0,2)+' sec', /info

  t0 = systime(1)
  message, 'transpose(A)#B', /info
  c = transpose(a)#b
  message, 'took '+strtrim(systime(1)-t0,2)+' sec', /info

  t0 = systime(1)
  message, 'matrix_multipy(A,B,/atranspose)', /info
  c = matrix_multiply(a,b,/atranspose)
  message, 'took '+strtrim(systime(1)-t0,2)+' sec', /info

  ahold = a

  t0 = systime(1)
```

```
  message, 'transpose(temporary(A))#B', /info
  c = transpose(temporary(a))#b
  message, 'took '+strtrim(systime(1)-t0,2)+' sec', /info

  a = ahold

  t0 = systime(1)
  message, 'matrix_multipy(temporary(A),B,/atranspose)', /info
  c = matrix_multiply(temporary(a),b,/atranspose)
  message, 'took '+strtrim(systime(1)-t0,2)+' sec', /info

  return
end
```

Subject: Re: large matrix operations
Posted by David Fanning on Mon, 13 Oct 2008 17:39:09 GMT
View Forum Message <> Reply to Message

Vince Hradil writes:

> So much for that theory...
>
> Hardware:
> IDL> print, !version
> { x86 Win32 Windows Microsoft Windows 7.0 Oct 25 2007 32 64}
>
> Results:
> % TEST: Allocating A array
> % TEST: took 0.14000010 sec
> % TEST: Allocating B array
> % TEST: took 0.21899986 sec
> % TEST: A#B
> % TEST: took 40.878000 sec
> % TEST: matrix_multipy(A,B)
> % TEST: took 42.284000 sec
> % TEST: transpose(A)#B
> % TEST: took 42.645000 sec
> % TEST: matrix_multipy(A,B,/atranspose)
> % TEST: took 43.449000 sec
> % TEST: transpose(temporary(A))#B
> % TEST: took 43.387000 sec
> % TEST: matrix_multipy(temporary(A),B,/atranspose)
> % TEST: took 50.029000 sec

I'm glad to hear this, because I always thought that theory leaned a little toward the bogus side. In my experience, slow matrix operations are always due to memory paging problems.

According to a VERY interesting program on the History Channel
last night, shamanic rituals are sometimes effective in these
situations where we are up against demonic forces.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: large matrix operations
Posted by Michael Galloy on Mon, 13 Oct 2008 20:51:51 GMT
View Forum Message <> Reply to Message

On Oct 13, 11:07 am, Vince Hradil <vincehra...@gmail.com> wrote:
> So much for that theory...
>
> Hardware:
> IDL> print, !version
> { x86 Win32 Windows Microsoft Windows 7.0 Oct 25 2007 32 64}
>
> Results:
> % TEST: Allocating A array
> % TEST: took 0.14000010 sec
> % TEST: Allocating B array
> % TEST: took 0.21899986 sec
> % TEST: A#B
> % TEST: took 40.878000 sec
> % TEST: matrix_multipy(A,B)
> % TEST: took 42.284000 sec
> % TEST: transpose(A)#B
> % TEST: took 42.645000 sec
> % TEST: matrix_multipy(A,B,/atranspose)
> % TEST: took 43.449000 sec
> % TEST: transpose(temporary(A))#B
> % TEST: took 43.387000 sec
> % TEST: matrix_multipy(temporary(A),B,/atranspose)
> % TEST: took 50.029000 sec

I think there are problems in the testing mechanism:

* you're only timing one execution of the operation
* you're timing MESSAGE which has an I/O component

  * allocating B was 50% longer than allocating A? They are the same
size, only 12 MB.

I ran the same tests 10 times and averaged the results, also removing
the MESSAGE statements from the timed portions.

Here are my results:

```
IDL> print, !version
{ i386 darwin unix Mac OS X 7.0 Oct 25 2007      32      64}

% TEST: Allocating A: 0.0767788
% TEST: Allocating B: 0.0775957
% TEST: A # B: 7.68955
% TEST: MATRIX_MULTIPLY(A, B): 7.62423
% TEST: TRANSPOSE(A) # B: 7.64414
% TEST: MATRIX_MULTIPLY(A, B, /ATRANSPOSE): 6.66077
% TEST: TRANSPOSE(TEMPORARY(A)) # B: 7.51523
% TEST: MATRIX_MULTIPLY(TEMPORARY(A), B, /ATRANSPOSE): 6.50667
```

Here is my code:

```
pro test
  nel = 2000L
  ntests = 10L
  times = fltarr(8, ntests)

  for i = 0L, ntests - 1L do begin
    print, 'Running test ' + strtrim(i, 2) + '...'
    t0 = systime(1)
    a = randomu(seed,[nel,nel])
    times[0, i] = systime(1)-t0

    t0 = systime(1)
    b = randomu(seed,[nel,nel])
    times[1, i] = systime(1)-t0

    t0 = systime(1)
    c = a#b
    times[2, i] = systime(1)-t0

    t0 = systime(1)
    c = matrix_multiply(a,b)
    times[3, i] = systime(1)-t0

    t0 = systime(1)
    c = transpose(a)#b
    times[4, i] = systime(1)-t0
```

```
    t0 = systime(1)
    c = matrix_multiply(a,b,/atranspose)
    times[5, i] = systime(1)-t0

    ahold = a

    t0 = systime(1)
    c = transpose(temporary(a))#b
    times[6, i] = systime(1)-t0

    a = ahold

    t0 = systime(1)
    c = matrix_multiply(temporary(a),b,/atranspose)
    times[7, i] = systime(1)-t0
  endfor

  message, 'Allocating A: ' + strtrim(mean(times[0, *]), 2), /info
  message, 'Allocating B: ' + strtrim(mean(times[1, *]), 2), /info
  message, 'A # B: ' + strtrim(mean(times[2, *]), 2), /info
  message, 'MATRIX_MULTIPLY(A, B): ' + strtrim(mean(times[3, *]), 2), /
info
  message, 'TRANSPOSE(A) # B: ' + strtrim(mean(times[4, *]), 2), /info
  message, 'MATRIX_MULTIPLY(A, B, /ATRANSPOSE): ' +
strtrim(mean(times[5, *]), 2), /info
  message, 'TRANSPOSE(TEMPORARY(A)) # B: ' + strtrim(mean(times[6,
*]), 2), /info
  message, 'MATRIX_MULTIPLY(TEMPORARY(A), B, /ATRANSPOSE): ' +
strtrim(mean(times[7, *]), 2), /info
end
```

Mike
--
www.michaelgalloy.com
Tech-X Corporation
Software Developer II

Subject: Re: large matrix operations
Posted by Vince Hradil on Mon, 13 Oct 2008 21:17:10 GMT
View Forum Message <> Reply to Message

On Oct 13, 3:51 pm, "mgal...@gmail.com" <mgal...@gmail.com> wrote:
> On Oct 13, 11:07 am, Vince Hradil <vincehra...@gmail.com> wrote:
>
>
>

>> So much for that theory...
>
>> Hardware:
>> IDL> print, !version
>> { x86 Win32 Windows Microsoft Windows 7.0 Oct 25 2007 32 64}
>
>> Results:
>> % TEST: Allocating A array
>> % TEST: took 0.14000010 sec
>> % TEST: Allocating B array
>> % TEST: took 0.21899986 sec
>> % TEST: A#B
>> % TEST: took 40.878000 sec
>> % TEST: matrix_multipy(A,B)
>> % TEST: took 42.284000 sec
>> % TEST: transpose(A)#B
>> % TEST: took 42.645000 sec
>> % TEST: matrix_multipy(A,B,/atranspose)
>> % TEST: took 43.449000 sec
>> % TEST: transpose(temporary(A))#B
>> % TEST: took 43.387000 sec
>> % TEST: matrix_multipy(temporary(A),B,/atranspose)
>> % TEST: took 50.029000 sec
>
> I think there are problems in the testing mechanism:
>
> * you're only timing one execution of the operation
> * you're timing MESSAGE which has an I/O component
> * allocating B was 50% longer than allocating A? They are the same
> size, only 12 MB.
>
> I ran the same tests 10 times and averaged the results, also removing
> the MESSAGE statements from the timed portions.
>
> Here are my results:
>
> IDL> print, !version
> { i386 darwin unix Mac OS X 7.0 Oct 25 2007 32 64}
>
> % TEST: Allocating A: 0.0767788
> % TEST: Allocating B: 0.0775957
> % TEST: A # B: 7.68955
> % TEST: MATRIX_MULTIPLY(A, B): 7.62423
> % TEST: TRANSPOSE(A) # B: 7.64414
> % TEST: MATRIX_MULTIPLY(A, B, /ATRANSPOSE): 6.66077
> % TEST: TRANSPOSE(TEMPORARY(A)) # B: 7.51523
> % TEST: MATRIX_MULTIPLY(TEMPORARY(A), B, /ATRANSPOSE): 6.50667
>

```
>  Here is my code:
>
>  pro test
>    nel = 2000L
>    ntests = 10L
>    times = fltarr(8, ntests)
>
>    for i = 0L, ntests - 1L do begin
>      print, 'Running test ' + strtrim(i, 2) + '...'
>      t0 = systime(1)
>      a = randomu(seed,[nel,nel])
>      times[0, i] = systime(1)-t0
>
>      t0 = systime(1)
>      b = randomu(seed,[nel,nel])
>      times[1, i] = systime(1)-t0
>
>      t0 = systime(1)
>      c = a#b
>      times[2, i] = systime(1)-t0
>
>      t0 = systime(1)
>      c = matrix_multiply(a,b)
>      times[3, i] = systime(1)-t0
>
>      t0 = systime(1)
>      c = transpose(a)#b
>      times[4, i] = systime(1)-t0
>
>      t0 = systime(1)
>      c = matrix_multiply(a,b,/atranspose)
>      times[5, i] = systime(1)-t0
>
>      ahold = a
>
>      t0 = systime(1)
>      c = transpose(temporary(a))#b
>      times[6, i] = systime(1)-t0
>
>      a = ahold
>
>      t0 = systime(1)
>      c = matrix_multiply(temporary(a),b,/atranspose)
>      times[7, i] = systime(1)-t0
>    endfor
>
>    message, 'Allocating A: ' + strtrim(mean(times[0, *]), 2), /info
>    message, 'Allocating B: ' + strtrim(mean(times[1, *]), 2), /info
```

> message, 'A # B: ' + strtrim(mean(times[2, *]), 2), /info
> message, 'MATRIX_MULTIPLY(A, B): ' + strtrim(mean(times[3, *]), 2), /
> info
> message, 'TRANSPOSE(A) # B: ' + strtrim(mean(times[4, *]), 2), /info
> message, 'MATRIX_MULTIPLY(A, B, /ATRANSPOSE): ' +
> strtrim(mean(times[5, *]), 2), /info
> message, 'TRANSPOSE(TEMPORARY(A)) # B: ' + strtrim(mean(times[6,
> *]), 2), /info
> message, 'MATRIX_MULTIPLY(TEMPORARY(A), B, /ATRANSPOSE): ' +
> strtrim(mean(times[7, *]), 2), /info
> end
>
> Mike
> --www.michaelgalloy.com
> Tech-X Corporation
> Software Developer II

Mike,
Thanks for cleaning that up. I think the conclusions are the same -
matrix_multiply does not speed things up. I wonder what happens if we
really push the envelope wrt RAM and matrix size. Unfortunately, I
don't have a lot of time to play with this.

Vince

Subject: Re: large matrix operations
Posted by Michael Galloy on Mon, 13 Oct 2008 22:14:29 GMT
View Forum Message <> Reply to Message

On Oct 13, 3:17 pm, Vince Hradil <vincehra...@gmail.com> wrote:
> Thanks for cleaning that up. I think the conclusions are the same -
> matrix_multiply does not speed things up. I wonder what happens if we
> really push the envelope wrt RAM and matrix size. Unfortunately, I
> don't have a lot of time to play with this.

MATRIX_MULTIPLY is about 15% faster in my tests, but I guess you were
wanting a *very* significant speedup i.e. like the speedup you get
when vectorizing code?

Mike
--
www.michaelgalloy.com
Tech-X Corporation
Software Developer II

Subject: Re: large matrix operations
Posted by Vince Hradil on Tue, 14 Oct 2008 13:10:24 GMT
View Forum Message <> Reply to Message

On Oct 13, 5:14 pm, "mgal...@gmail.com" <mgal...@gmail.com> wrote:
> On Oct 13, 3:17 pm, Vince Hradil <vincehra...@gmail.com> wrote:
>
>> Thanks for cleaning that up. I think the conclusions are the same -
>> matrix_multiply does not speed things up. I wonder what happens if we
>> really push the envelope wrt RAM and matrix size. Unfortunately, I
>> don't have a lot of time to play with this.
>
> MATRIX_MULTIPLY is about 15% faster in my tests, but I guess you were
> wanting a *very* significant speedup i.e. like the speedup you get
> when vectorizing code?
>
> Mike
> --www.michaelgalloy.com
> Tech-X Corporation
> Software Developer II

Yeah- I guess if you're doing a lot of these and each one can be 15%
faster then you might gain something. I was really expecting
(hoping?) MATRIX_MULTIPLY to be 50% or better faster. Besides, if my
programs run too fast how can I justify my salary? 8^)

I'm reminded of this: http://xkcd.com/303/