Subject: Re: array multiplying (for a change)
Posted by Craig Markwardt on Tue, 17 Feb 2004 16:21:04 GMT
View Forum Message <> Reply to Message

"Christopher Lee" <cl@127.0.0.1> writes:
> What I want is result = a * b'
> where b' = rebin(reform(b, [1,20,1]), 10,20,30)
> , which (clearly :) I know how to do in principle.
...
> Are there any functions, buit-in or otherwise, that I can use? I found
> CMAPPLY, which I can beat into a form which works. (I use a similar
> function now but it's very _VERY_ bad code).
>
> A quick test using loops versus rebin/reform of the shows loops to be
> slower (for a matrix 72,36,31,200) which I'm not really surprised by. Is
> this a case where a DLM would be faster?

My philosophy is that DLMs are almost always bad, unless you are developing an embedded system. They tie you to a particular version of IDL and a particular OS and architecture. They are rather difficult to debug, and making changes is rather laborious. DLMs = bleccchhh.

With that out of my system, I think that a slab-oriented multiply would probably do okay. By "slab oriented" I mean to expand B in a few but not all dimensions, so essentially this will be a hybrid between REBIN/REFORM and FOR-loop.

Example:
```
bp = rebin(reform(b, [1,20,1]), 10,20,1)
result = a
for i = 0, 29 do result(*,*,i) = a(*,*,i) * bp
```

Since that last dimension of 20 is not a part of BP, it doesn't take up nearly as much memory, but the number of loop iterations is also rather small. This notation is compact enough that I typically do not make a wrapper procedure.

You can save even more execution time by using the IDL "trick" of specifying only the start index for the destination array:

```
for i = 0, 29 do result(0,0,i) = a(*,*,i) * bp
```

Good luck!
Craig

--
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@REMOVEcow.physics.wisc.edu

Subject: Re: array multiplying (for a change)
Posted by JD Smith on Tue, 17 Feb 2004 20:18:47 GMT
View Forum Message <> Reply to Message

On Tue, 17 Feb 2004 10:21:04 -0600, Craig Markwardt wrote:

>
> "Christopher Lee" <cl@127.0.0.1> writes:
>> What I want is result = a * b'
>> where b' = rebin(reform(b, [1,20,1]), 10,20,30)
>> , which (clearly :) I know how to do in principle.
> ...
>> Are there any functions, buit-in or otherwise, that I can use? I found
>> CMAPPLY, which I can beat into a form which works. (I use a similar
>> function now but it's very _VERY_ bad code).
>>
>> A quick test using loops versus rebin/reform of the shows loops to be
>> slower (for a matrix 72,36,31,200) which I'm not really surprised by. Is
>> this a case where a DLM would be faster?

I can think of almost no case where a DLM wouldn't be faster; the real
questions is, is a DLM faster by a large enough margin to make it
worth it?

> My philosophy is that DLMs are almost always bad, unless you are
> developing an embedded system. They tie you to a particular version
> of IDL and a particular OS and architecture. They are rather difficult
> to debug, and making changes is rather laborious. DLMs = bleccchhh.

That may be true to some extent, but I have a method for calling
compiled C code automatically within IDL which is, as far as I can
tell, as portable as possible. The MAKE_DLM routine allows you to
invoke a standard compiler to produce a shared executable library. A
few other tricks then check that the compilation succeeded, and
execute the compiled code (I usually just use CALL_EXTERNAL). Is this
guaranteed to work? No, of course not. The compiler could be
mis-configured or missing. But it does provide a decent degree of
portability, and completely relieves the end-user from having to know
which end of a compiler is up. The AUTO_GLUE functionality makes it
easy to call existing functions (e.g. N.R.) without too much trouble.
In my case, I include an equivalent but slow version of the algorithm
coded in IDL, which I use as a fall-back if the compilation fails.

JD

Subject: Re: array multiplying (for a change)
Posted by Chris Lee on Wed, 18 Feb 2004 09:28:15 GMT
View Forum Message <> Reply to Message

In article <pan.2004.02.17.20.18.46.724041@as.arizona.edu>, "JD Smith" <jdsmith@as.arizona.edu> wrote:
> JD
> I can think of almost no case where a DLM wouldn't be faster; the real
> questions is, is a DLM faster by a large enough margin to make it worth
> it?

Indeed, I guess there's only one way I'll ever find out :(

>> CM
>> With that out of my system, I think that a slab-oriented multiply
>> would probably do okay. By "slab oriented" I mean to expand B in a
>> few but not all dimensions, so essentially this will be a hybrid
>> between REBIN/REFORM and FOR-loop.

I did try this, though not with the IDL trick of specifying only the start index (I thought this worked with the last dimension only?). A quick test shows that this would double the speed.

Something for the weekend, perhaps :)

Chris.

>> CM
>> My philosophy is that DLMs are almost always bad, unless you are
>> developing an embedded system. They tie you to a particular version of
>> IDL and a particular OS and architecture. They are rather difficult to
>> debug, and making changes is rather laborious. DLMs = bleccchhh.
> That may be true to some extent, but I have a method for calling
> compiled C code automatically within IDL which is, as far as I can tell,
> as portable as possible. The MAKE_DLM routine allows you to invoke a
> standard compiler to produce a shared executable library. A few other
> tricks then check that the compilation succeeded, and execute the
> compiled code (I usually just use CALL_EXTERNAL). Is this guaranteed to
> work? No, of course not. The compiler could be mis-configured or
> missing. But it does provide a decent degree of portability, and
> completely relieves the end-user from having to know which end of a
> compiler is up. The AUTO_GLUE functionality makes it easy to call
> existing functions (e.g. N.R.) without too much trouble. In my case, I
> include an equivalent but slow version of the algorithm coded in IDL,
> which I use as a fall-back if the compilation fails. JD