Subject: Re: Matrix expansion performance
Posted by Chris Lee on Mon, 28 Mar 2005 13:43:39 GMT
View Forum Message <> Reply to Message

In article <d28tre$j32$1@pegasus.fccn.pt>, "Ricardo Bugalho"
<rbugalho@ibili.uc.pt> wrote:

> Hi,
> I have a matrix A (m,n) is and I want to create a matrix B(m,n,p) such
> that each B(*,*,i) slice equals A. p is very large and n is usually
> smaller than m so I have:
> B=bytArr(m,n,p)
> C=byteArr(p) + 1
> FOR i = 0, n-1 DO B[*,i,*] = REFORM(A[*,i]) # p Quite fast, but not
> enough for my needs. Any one has better sugestions? Still stuck in IDL
> 5.4, by the way.
> Thanks,
>   Ricardo
>

Assuming your code was wrong, and that #p should be #C. A bit of reform
magic will do what you want.

http://www.dfanning.com/tips/rebin_magic.html

e.g.

b=rebin(reform(a, [m, n,1]), [m,n,p])

Chris.

Subject: Re: Matrix expansion performance
Posted by Kenneth P. Bowman on Mon, 28 Mar 2005 13:50:00 GMT
View Forum Message <> Reply to Message

In article <d28tre$j32$1@pegasus.fccn.pt>,
 "Ricardo Bugalho" <rbugalho@ibili.uc.pt> wrote:

> Hi,
> I have a matrix A (m,n) is and I want to create a matrix B(m,n,p) such that
> each B(*,*,i) slice equals A. p is very large and n is usually smaller than
> m so I have:
>
> B=bytArr(m,n,p)
> C=byteArr(p) + 1
> FOR i = 0, n-1 DO B[*,i,*] = REFORM(A[*,i]) # p

This should be quite fast, if I understand your problem correctly:

```
B = BYTARR(m,n,p)
FOR k = 0, p-1 DO B[0,0,k] = A
```

This will avoid subscript arrays and should access memory efficiently on
most machines.

Ken Bowman

Subject: Re: Matrix expansion performance
Posted by Ricardo Bugalho on Tue, 29 Mar 2005 09:55:29 GMT

View Forum Message <> Reply to Message

"Kenneth P. Bowman" <kpb@null.com> wrote in message
news:kpb-E62DD6.07500028032005@news.tamu.edu...
> In article <d28tre$j32$1@pegasus.fccn.pt>,
> "Ricardo Bugalho" <rbugalho@ibili.uc.pt> wrote:
>
>> I have a matrix A (m,n) is and I want to create a matrix B(m,n,p) such
>> that
>> each B(*,*,i) slice equals A. p is very large and n is usually smaller
>> than
>> m so I have:

>
> This should be quite fast, if I understand your problem correctly:

I think I didn't make clear the ranges of m,n and p.
In the problem I have at hand, m is always 8, n is usually 5 (min 1, max 16)
and p is in the range of 10,000 to 100,000.
Looping over p is a BadThing(tm) due to IDL's high interpretation overhead.

>
> B = BYTARR(m,n,p)
> FOR k = 0, p-1 DO B[0,0,k] = A
>
> This will avoid subscript arrays and should access memory efficiently on
> most machines.

Subject: Re: Matrix expansion performance
Posted by Timm Weitkamp on Wed, 30 Mar 2005 08:22:46 GMT

View Forum Message <> Reply to Message

On 29.03.05 at 10:55 +0100, Ricardo Bugalho wrote:

> I think I didn't make clear the ranges of m,n and p.
> In the problem I have at hand, m is always 8, n is usually 5 (min 1, max 16)
> and p is in the range of 10,000 to 100,000.
> Looping over p is a BadThing(tm) due to IDL's high interpretation overhead.

The method that Chris Lee suggested does not use loops. But I think there
is no need for any call to REFORM. And the dimension arguments to REBIN
must be scalars in IDL 5.4. A simple

```
  b = rebin(a, m, n, p, /sample)
```

should therefore work (and, hopefully, be fast enough for your purposes).

Timm

--
Timm Weitkamp <http://people.web.psi.ch/weitkamp>

Subject: Re: Matrix expansion performance
Posted by Ricardo Bugalho on Wed, 30 Mar 2005 12:18:32 GMT
View Forum Message <> Reply to Message

I used Chris' method.
However, I've been having some problems posting and my thanks to him got
lost.

"Timm Weitkamp" <dont.try@this.address> wrote in message
news:Pine.LNX.4.44.0503301010060.7505-100000@localhost.local domain...
>
> The method that Chris Lee suggested does not use loops. But I think there
> is no need for any call to REFORM. And the dimension arguments to REBIN
> must be scalars in IDL 5.4. A simple

Subject: Re: Matrix expansion performance
Posted by Chris Lee on Wed, 30 Mar 2005 15:21:02 GMT
View Forum Message <> Reply to Message

In article
<Pine.LNX.4.44.0503301010060.7505-100000@localhost.localdomain>, "Timm
Weitkamp" <dont.try@this.address> wrote:

> On 29.03.05 at 10:55 +0100, Ricardo Bugalho wrote:

>> I think I didn't make clear the ranges of m,n and p. In the problem I
>> have at hand, m is always 8, n is usually 5 (min 1, max 16) and p is in
>> the range of 10,000 to 100,000. Looping over p is a BadThing(tm) due to
>> IDL's high interpretation overhead.
> The method that Chris Lee suggested does not use loops. But I think
> there is no need for any call to REFORM. And the dimension arguments to
> REBIN must be scalars in IDL 5.4. A simple
>
>   b = rebin(a, m, n, p, /sample)
> should therefore work (and, hopefully, be fast enough for your
> purposes). Timm
>

My first reaction was "when did that happen?", I tried it without the
reform, and it works...except

```
IDL> help, rebin(fltarr(4,5),[7,4,5,6])
% REBIN: Result dimensions must be integer factor of original dimensions
```

doesn't work (6.1.1 Linux), but the reform version does

```
IDL> help, rebin(reform(fltarr(4,5),[1,4,5,1]),[7,4,5,6])
<Expression>    FLOAT     = Array[7, 4, 5, 6]
```

So my world-view isn't completely shattered :)

Chris.