Subject: Re: Array Concatenation Optimization
Posted by KM on Tue, 08 Feb 2005 23:18:51 GMT
View Forum Message <> Reply to Message

On Tue, 8 Feb 2005, Peter Mason wrote:
> Interesting. What platform are you using? I ran a little test
> program (included below) on an "un-hyperthreaded" P4 Windows2000
> laptop. I got the following times for N=2048 (a square, 2048*2048
> image):
> Method 1 (traditional): 0.30 (IDL6.1), 0.30 (IDL6.0), 0.74 (IDL5.5)
> Method 2 (transpose): 0.26 (IDL6.1), 0.27 (IDL6.0), 0.43 (IDL5.5)
> Method 3 (insertion): 0.57 (IDL6.1), 0.57 (IDL6.0), 0.76 (IDL5.5)
> Method 4 (steam power): 6.21 (IDL6.1), 6.40 (IDL6.0), 6.64 (IDL5.5)

I have two OS X boxes
1) 1 GHz G4 IDL 5.6 (slower than any of yours)
2) 2.5 GHz dual G5 IDL 6.0.3 (faster than all of yours)

My image size varies (depends on what the user drags the window to),
but it is up to 4800x2400. It gets larger, but at that point the
user can deal with a lag while I generate the image. Sometimes it is
as small as 1800x1800.

Computer 1 (about 1/2 the speed if your IDL5.5):
Method 1: 0.78
Method 2: 0.96
Method 3: 1.28
Method 4: 18.217
Method 5: 0.34 (fast!)

Computer 2 (a bit faster than your IDL6.1):
Method 1: 0.25
Method 2: 0.19
Method 3: 0.35
Method 4: 7.40
Method 5: 0.07 (fast!)

FYI, I added method 5 which is my one-liner:
```
t0 = systime(1)
rgb = [[[r[img]]],[[g[img]]],[[b[img]]]]
print, systime(1)-t0
```

> Curiously, the transpose method performed the best on my platform.
You taught me something about transpose. I just did a transpose()
and it takes about 3 times as long as your transpose(...[2,0,1]).
That subscript vector takes it from the slowest method to what is so

far usually the fastest... Thank you muchly!

-k.
http://edgcm.org/
http://spacebit.dyndns.org/