Subject: speed of accessing different dimensions
Posted by Jeremy Bailin on Wed, 20 Aug 2008 12:11:36 GMT
View Forum Message <> Reply to Message

Following on David's note that IDL is faster when accessing columns rather than rows, I decided to test exactly how much of a difference it makes. The result is quite interesting... for a 2D array:

```
IDL> q = fltarr(10000,10000)
IDL> a = systime(/sec) & p=total(q,1) & print, systime(/sec)-a
     0.38179994
IDL> a = systime(/sec) & p=total(q,2) & print, systime(/sec)-a
     0.80439496
```

So basically a factor of 2. But for a 3D array:

```
IDL> q = fltarr(300,300,300)
IDL> a = systime(/sec) & p=total(q,1) & print, systime(/sec)-a
     0.10734391
IDL> a = systime(/sec) & p=total(q,2) & print, systime(/sec)-a
     0.12382507
IDL> a = systime(/sec) & p=total(q,3) & print, systime(/sec)-a
     0.15879583
```

Early dimensions are still better, but there's not nearly the same difference. I'm guessing that this is because the memory gap between contiguous junks is bigger even when running over the first index? Going to 4D:

```
IDL> q = fltarr(80,80,80,80)
IDL> a = systime(/sec) & p=total(q,1) & print, systime(/sec)-a
     0.18169713
IDL> a = systime(/sec) & p=total(q,2) & print, systime(/sec)-a
     0.20045400
IDL> a = systime(/sec) & p=total(q,3) & print, systime(/sec)-a
     0.22229695
IDL> a = systime(/sec) & p=total(q,4) & print, systime(/sec)-a
     0.30576181
```

Hmmm... slight improvement for each of the first 3 dimensions, but a big penalty on the 4th! Not sure why, but it's very repeatable. In 5D we reach saturation:

```
IDL> q = fltarr(35,35,35,35,35)
IDL> a = systime(/sec) & p=total(q,1) & print, systime(/sec)-a
     0.26922297
IDL> a = systime(/sec) & p=total(q,2) & print, systime(/sec)-a
     0.32297802
```

```
IDL> a = systime(/sec) & p=total(q,3) & print, systime(/sec)-a
      0.35127592
IDL> a = systime(/sec) & p=total(q,4) & print, systime(/sec)-a
      0.35729289
IDL> a = systime(/sec) & p=total(q,5) & print, systime(/sec)-a
      0.35672593
```

I guess that means that this is the point where each successive chunk is always too far to be in cache and so there's a constant penalty to go get it from the bus each time?

Anyway, thought you guys might be interested in that. I guess the moral of the story is that in a 2D array you should always operate on the first index if possible, but in higher dimensions it doesn't matter so much.

-Jeremy.
