Subject: Re: a=a(*,*,[4,1,2,3,0]) efficiency
Posted by davis on Fri, 17 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

On 15 Jul 1998 01:30:30 +0200, David Kastrup
<dak@mailhost.neuroinformatik.ruhr-uni-bochum.de>
wrote:
> temporary in the first place. How about
>
> a = (temporary(a))[*,*,[4,1,2,3,0]]

I have no knowledge of the internals of IDL, but I do not think that the use of `temporary' will help. I am guessing that `temporary' simply does the following:

1. Push value of `a' onto the stack. This results in the reference count to array attached to `a' being increased by 1.

2. Free `a' and undefine the variable. This has the effect of decrementing the reference count of array attached to `a' by 1.

The net result is that the ownership of the array attached to `a' will have changed from `a' to the stack. Now consider:

a = a[*,*,[4,1,2,3,0]]

This will probably do the following:

1. Push value of `a' onto stack. Reference count of array increased by 1.

2. Retrieve array from stack.

3. Create a new array that is a copy of the array on the stack but with elements interchanged. Push result onto stack with a reference count of 1.

4. Free array popped from stack. This reduces the reference count of array attached to `a' by 1.

5. Assign the value of array on stack to `a'. First free the array attached to `a', reducing the reference count by 1.

6. Then remove the new array from the stack and assign it to `a'. The reference count of this array is still 1.

In both cases, at some instant, the original array and its ``interchanged'' copy will both exist. All `temporary' does is move

step 5 to between steps 1 and 2.

I imagine that `temporary' is really only useful in more complex expressions, e.g., consider

    a = (a + b) + c

which consists of 3 arrays `a', `b', and `c'. During the evaluation of the RHS of this statement, 2 extra arrays will be created: (a+b) and the result (a+b)+c. Thus at some point, 5 arrays will exist. Just prior to the assignment to `a', the temporary arrat (a+b) will be freed. Now consider:

    a = (temporary(a) + b) + c

After the evaluation of (temporary(a)+b), only 3 arrays will exist: (a+b), b, and c. Then when (a+b) is added to `c', another array will be created raising the total number needed to 4.

Again, this is pure speculation and I may be totally wrong. But I cannot thing of another way to implement this.

--John
