Subject: Re: Concatenating arrays - speed issues?
Posted by Craig Markwardt on Tue, 07 Jun 2011 16:42:54 GMT
View Forum Message <> Reply to Message

On Jun 7, 11:48 am, Rob <rj...@le.ac.uk> wrote:
> Hi
>
> This is a pretty basic question but I'm trying to speed up some code
> and at the minute it's quite ugly and does this:
>
> 1) Loop over input data
> 2) Perform some calculations based on input data which may or may not
> produce a result we want to use
> 3a) If a result is produced and it's the first time within the loop,
> create an array to hold it
> 3b) If a result is produced but it's not the first time within the
> loop, concatenate the result to the array that's already been created.
>
> I'm doing the concatenation with something like:
>
> array = [[array],value]
>
> Now this works fine but as we get more input data it seems the
> concatenation is becoming quite slow (presumably as the arrays are
> getting larger and larger).
>
> The alternative I guess would be to define an array at the start with
> some arbitrarily large size, subscript the values to it and then check
> at the end to trim empty elements but that doesn't seem much nicer and
> in this case estimating an arbitrary size isn't that straight-forward.
>
> Is there an "IDL way" way to do this?

What you are doing is the "IDL way" in the sense that it's a natural
use of the concatenation feature of the language.

But as you noticed, the performance degrades for lots of append
operations.

The next best way is to grow the array in chunks, and then fill in the
chunks with available data. This forces you to keep track of the
number of used elements in the array, separate from the array size.
Once you fill the available chunk, only then do you add another
chunk.

This doesn't really get rid of the problem you noticed, but it does
reduce the problem significantly. So, if each chunk has 1000
elements, then the performance degradation is 1000x less. Then you

can start to get fancy by growing the array with variable sized chunks.

Craig
