What is the use of preallocating memory for an array in perl?

Asked 8 years, 8 months ago Modified 8 years, 8 months ago

Viewed 722 times

12

Perl allows preallocated arrays. We can preallocate the array before use, then we can add more elements. For example, allocating 50 array members then adding a 51st member, because arrays are expandable. So does preallocating an array improve performance?

arrays perl

Share

Improve this question

Follow

edited Apr 22, 2016 at 0:12

oldtechaa

1,524 ● 1 ● 15 ● 28

asked Apr 21, 2016 at 16:49

SSN

886 ● 2 ● 9 ● 20

1 Answer

Sorted by: Highest score (default)

Its because of how memory is allocated in computers. Computer memory is like space on a whiteboard: it has a

position in relation to other memory; and it cannot be moved, it must be copied.

If you create a small array it might look like this:

```
@array = (1, 4, 8, 12, 19);

allocate memory for @array
_____________________|                |_____| a b c|_

copy in the data
_____________________|  1  4  8 12 19|_____| a b c|_
```

`_` is unallocated memory. `|` indicates the bounds of what is allocated to your array. `| a b c|` is some other array.

Then if you push onto that array a few times, Perl will have to reallocate memory. In this case it can grow the memory it already has into the unallocated space.

```
push @array, 23, 42;

grow the existing memory
_____________________|  1  4  8 12 19      | a b c|__

add the new data
_____________________|  1  4  8 12 19 23 42| a b c|__
```

Now what happens if you push more numbers onto `@array`? It can't grow your memory anymore, there's another array in the way. So, just like on a whiteboard, it has to copy the entire array to a clear chunk of memory.

```
push @array, 85, 99;

Allocate a new chunk of memory
|                              |  1  4  8 12 19 23 42| a

Copy the existing data
|  1  4  8 12 19 23 42         |  1  4  8 12 19 23 42| a

Deallocate the old memory
|  1  4  8 12 19 23 42         |__1__4__8_12_19_23_42| a

Add the new data
|  1  4  8 12 19 23 42 85 99|__1__4__8_12_19_23_42| a
```

To save time, Perl will not bother to erase the old data. It will just deallocate it and something else can scribble over it when they need to.

This makes push more expensive, especially with very large arrays which need to copy more data. As your array gets larger it's more and more likely that Perl will have to allocate a fresh hunk of memory and copy everything.

There's another problem: memory fragmentation. If you're allocating and reallocating over and over again, the hunks of memory can get chopped up so it's difficult to find big blocks of free memory. This is less of a problem on modern operating systems, but still a concern. It can make it seem like you have less memory than you really have, and it can cause the Operating System to use the disk as memory (virtual memory) more than it should. Disks are slower than memory.

I simplified a lot of things. I made it look like Perl has to reallocate every time you `push`. This isn't true. Perl allocates more memory to arrays than it needs for just this reason. So you can safely add a few extra entries to an array without Perl having to reallocate. The same goes for strings and hashes.

The other thing is this is probably a somewhat outdated view of how memory allocation works on modern operating systems... although Perl will sometimes do its own memory allocation if it doesn't trust the OS. Check `use Config; print $Config{usemymalloc}`. `n` indicates Perl is using the operating system's memory allocation, `y` indicates it's using Perl's.

The rule of thumb is: don't preallocate, it's probably a waste of your time and the computer's memory. However, if *all* of the conditions below are true, see if preallocating helps.

- You profiled and found a problem.
- You're building a data structure incrementally by adding to it.
- You know for certain its minimum eventual size.
- That size is "large".

What is "large" is up for debate and depends on your version of Perl, your operating system, your hardware, and your performance tolerance.

Share Improve this answer Follow

edited Apr 21, 2016 at 17:45

answered Apr 21, 2016 at 17:31

Schwern

164k ● 27 ● 218 ● 362