

Re: Garbage collectable pinned arrays!

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-02/msg01506.html>

- *From:* "Willy Denoyette [MVP]" <willy.denoyette@xxxxxxxxxxx>
 - *Date:* Tue, 12 Feb 2008 13:52:55 +0100
-

"Atmapuri" <janez.makovsek@xxxxxxxx> wrote in message
<news:5E3B6EF0-6F5A-4DA0-A3B0-4B3C4693995E@xxxxxxxxxxxxxxxxxxxx>

Hi!

As I understand it, pinning is an attribute of the **reference**, not the object itself. The object is pinned when there's a pinning reference to it anywhere on the call stack.

Thus, it doesn't make sense for a pinned object to be garbage collectible. The object can't be considered garbage anyway while you're still holding a reference to it, and once you let go of the last reference, it's no longer pinned.

I don't see why you'd even want it to be collectible, actually. The point of pinning is to let unmanaged code access the object without worrying that the GC will move it. But collecting the object and letting its memory be used for something else is just as dangerous as moving it!

You are mixing two points:

- reference to unmanaged memory, where the word pinned also means that it won't be collected.
- location of the array in GC (Heap or else).

When the array is pinned it is copied to heap. All I would like to see is an option to allocate the array on the heap initially.

Array's are reference types, so always stored on the heap!

The array is automatically allocated on the heap once it exceeds a certain size and thus becomes "pinned", because the heap is never compacted and all addresses are absolute.

Re: Garbage collectable pinned arrays!

Arrays that exceed a certain size (85Kb currently), are moved to the Large Object Heap, but they aren't pinned by this. The LOH is not compacted, but that doesn't mean that the objects cannot get collected.

Therefore, the pinned keyword during memory allocation would only instruct the GC where to put the array. From there on it can work as usual.

The reason to put the array on the heap is to avoid the need to pin it, if you have to pass it to unmanaged code multiple times. Thus in turn improving performance in such cases.

Arrays are always on the heap, when passed to unmanaged land, all heap allocated objects must be pinned in order to avoid relocation by the *compacting* GC.

..

So, I would like to specify that the memory is allocated on the heap, and before passing it to the external code,

What memory are you talking about? Reference types are always on the heap, arrays are reference types so.....

I can still call `GC.KeepAlive()`, to prevent the array from being collected if needed. That is still less code than calling `GCHandle.Alloc` and multiple times faster than current implementation which copies the array.

Not sure what you mean here, mind to elaborate?

The GC can still track which arrays have been declared as pinned and make sure that they are not collected while unmanaged code is executing. Same as now.

Not sure what you mean here, mind to elaborate?

Willy.

Re: Garbage collectable pinned arrays!