

Re: arrays = pointers?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-03/msg00480.html>

- *From:* "Peter Duniho" <NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sat, 03 Mar 2007 17:26:11 +0800
-

On Sat, 03 Mar 2007 00:15:41 +0800, Zytan <zytanlithium@xxxxxxxxxx> wrote:

Pete, consider that it is optimized for the most gain as a whole.

One hopes (and I assume). However, it's not a given. If it turns out that the memory scheme used by .NET is less efficient than other possible schemes, well...let's just say that wouldn't be the first time some major blunder was made when creating some computer architecture.

Most of the time, people get it right. But it's not a given that they always do. I've been around long enough to know better than to just assume that they have.

Meaning, if most often, things are only referenced once or twice, then that's more important to make them fast than anything else, since that's where the speed will be gained or lost.

Huh? I think you have that backwards. If something is referenced only once or twice, then it doesn't matter how long it takes to resolve that reference. It's when something is referenced a huge number of times that the cost to resolve the reference is significant.

10x to update a single reference? Yes, i know that single reference must be stored somehow that allows other references to be in there, as well, in some kind of data structure, or hash, or something, but, still. And, as you said, even if it was 10x as bad as the "handle" scheme, we're probably still ahead.

Well, except that I suspect that the difference in the relocation algorithm is greater than 10x.

Also, isn't it unfair to compare it to the "handle" scheme? Shouldn't you be comparing it to whatever other alternative the C# could have used? Or, are you claiming the "handle" scheme IS something C# could

Re: arrays = pointers?

have used (I may have missed that)?

..NET (C# is just the compiler) certainly could have used the exact same handle scheme. There's nothing fundamentally wrong with it. The major disadvantage it has, of course, is that it does complicate resolving references to objects. And it would require the compiler to do more work. It introduces a variety of extra complexities in the compiled code (eg, having to double-dereference all the time, having to remember to lock the object when modifying the data, and the scheme wouldn't work nearly so well in a true multitasking environment as it did in the cooperative multitasking environment used in the old Mac OS), as a cost of simplifying the garbage collection.

Pete

.