Does using delegates slow down my .NET programs?

Asked 16 years, 1 month ago Modified 13 years, 9 months ago

Viewed 7k times

16

Does using delegates slow down my programs?

I've been avoiding them because I really have no clue if they make my programs any slower. I know if I cause a (catch) exception, that uses quite a bit of CPU power but I don't know about Delegates and Events and what .NET does to them.

.net delegates

Share

Improve this question

Follow

asked Nov 20, 2008 at 9:29

danmine

11.5k • 18 • 57 • 75

1 why haven't you accepted an answer even for this question? – nawfal Apr 19, 2012 at 4:43

4 Answers

Sorted by: Highest score (default)

18

Delegates are very, very fast. Not quite as fast as direct method calls, but not far off. The chances of them becoming a bottleneck are miniscule.

(Likewise exceptions, when used properly, rarely actually cause a performance issue.)

Would using delegates make your code simpler, more readable, and more robust? If so, use them. Measure your performance carefully, and keep an eye on it. Move away from readability for the sake of performance only when the data is clear.

I'm sure there are some graphs around showing the speed of delegates vs interfaces vs non-virtual method calls etc - I don't know where they are, but you could always run tests yourself if you're really worried.

Share Improve this answer

Follow

edited May 23, 2017 at 10:26

Community Bot

1 • 1

answered Nov 20, 2008 at 9:32

Jon Skeet

1.5m • 889 • 9.3k • 9.3k

3 Jon - I just fired up reflector and looked at the default implementation of MulticastDelegate.CombineImpl, and was somewhat horrified as to how lengthy the method is, and to see that it's recreating arrays insternally. I'd have thought delegates would be combined using linked lists. Have you run any tests to see how performant they really are? – Mark Oct 24, 2010 at 15:43

2 @Mark: Not for the combining and removing, but that's rarely what happens very often in my experience. Usually the same delegates are called many, many times (e.g. for LINQ) so it's the invocation speed that matters rather more. – Jon Skeet Oct 24, 2010 at 15:54

9

Just a small addition to Jon's post: when used in the normal C# way (i.e. via lambdas / anonymous methods / event handlers / etc), then they are definitely very quick - but note that another important use of delegates can be to execute dynamic code (either methods built at runtime, or existing methods via reflection and Delegate.CreateDelegate). When used in this second way, delegates offer a very significant speed *improvement* (compared to reflection Invoke etc).

So don't be shy about using delegates. And in particular, if in doubt - measure it for realistic code: it is meaningless whether it takes 1 or 100 micro-weasels*, if this still only accounts for 0.01% of your overall execution time.

*=just some arbitrarily small amount of time...

Share Improve this answer answered Nov 20, 2008 at 9:41

Follow

Marc Gravell
1.1m ● 272 ● 2.6k ● 3k

3 From now on I think I'll give any performance results in pico-fortnights. No-one can claim that they're missing information, but it emphsizes the *relative* nature of most performance comparisons. Time for an extension method on TimeSpan... – Jon Skeet Nov 20, 2008 at 9:50

1 Is a pico-fortnight longer or shorter than a micro-weasel? Inquiring minds wants to know :) – Lasse V. Karlsen Nov 20, 2008 at 12:17

3 If it helps,, a pico-fortnight is roughly 1.2 microseconds – Marc Gravell Nov 20, 2008 at 15:07

2 @Agnel - which still matters precisely zero if it "only accounts for 0.01% of your overall execution time" - you have bigger things to fix! – Marc Gravell Mar 16, 2009 at 9:14

1 btw, considering pico is 10^-12, one pico-fortnight is 0.0012096 milliseconds. Probably an acceptable type of time length, but I very much like the unit micro-weasel. – flq Jul 8, 2010 at 19:10

6

I work on Windows CE so this kind of thing is sometimes a little more pertinent. For example a brash application of reflection can really hurt so we tend to avoid reflection where sensible (obviously small applications of reflection are *fine*). Obviously I do no such madness on desktop.

I've heard people murmour about delegates and CE performance but as far as i'm concerned its utter guff. I heard that it "slows down the method call by 30%" but if thats 30% of a crappy algorithm then whose fault is that?

Another slow down in CE is virtual methods, as there is no lookup table it manually works them out first time and caches the result. This means if you piss away all your memory those caches will be purged and it will result in a perf hit next time round. But that considered, should you throw away useful OOP skills for the sake of performance?

I find that a lot of these "OMG don't use that it's too slow" are just excuses. Mainly excuses because people don't know whats *really* wrong with their app and its easy to blame some internal working of the CLR instead of their own code. If your performance sucks then I'd reckon that 99.9% of the time you can change something in your part of the app or design that doesn't throw away tools and results in much better improvements.

Share Improve this answer Follow

edited Nov 20, 2008 at 12:52

answered Nov 20, 2008 at 11:59

Quibblesome

25.4k ● 10 ● 62 ● 104

6

Performance can be a touchy subject when it comes to programming. For example, some people are absolutley adamant that boxing is the root of all evil. Other people think string concats are a big performance hit.

In reality everything is relative and it all boils down to what context you are talking about. If you are programming on a mobile device, then you will want to optimise more than if you were working on a desktop application.

It usually comes down to a trade off between performance and code elegance. Lets say you made the most wonderfully elegant, maintainable and understandable code base in the world. As soon as we throw in some performance optimizations we start to cloud the code with some possibly counter intuitive, very specialised stuff. If we went to town on optimizing it we could possibly make a performance saving of say 5 or 10 percent, but in the process completley destroy the elegance of the code.

The question is "Is it worth it?".

If performance is absolutley critical for your project, then run a profiler on your code. If you find that 90% of your processor time is being eaten by a particularly inefficient method, then that method is a good candidate for optimization. It's usually not worth chasing low performance benefits unless you are working on a performance critical application.

edited Mar 23, 2011 at 17:01

answered Nov 20, 2008 at 11:32

Doctor Jones
21.7k ● 13 ● 80 ● 102