

Re: perf & Try Catch

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-12/5791.html>

From: W.G. Ryan eMVP (*WilliamRyan_at_NoSpam.gmail.com*)

Date: 12/29/04

Date: Tue, 28 Dec 2004 22:57:17 -0500

David:

Don't mean to nitpick but there isn't a hit with simply wrapping the code in a try catch

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/dotnetperftips.asp>

"Finding and designing away exception-heavy code can result in a decent perf win. Bear in mind that this has nothing to do with try/catch blocks: you only incur the cost when the actual exception is thrown. You can use as many try/catch blocks as you want. Using exceptions gratuitously is where you lose performance. For example, you should stay away from things like using exceptions for control flow."

Throwing the exception is definitely of much more concern and where you should be concerned -- but you can wrap away -- no hit there whatsoever.

--

W.G. Ryan MVP (Windows Embedded)

TiBA Solutions

www.tibasolutions.com | www.devbuzz.com | www.knowdotnet.com

"David Levine" <noSpamdlevineNNTP2@wi.rr.com> wrote in message
news:en78jkr7EHA.2592@TK2MSFTNGP09.phx.gbl...

> It depends entirely on what the code does - ultimately the code must be
> correct regardless of the perf, but there may be different approaches you
> can use to maximize perf and still get correct results.
>
> There is a small performance hit in setting up a try-catch block, so if
you
> are in a loop you can get a perf gain by wrapping the for loop in a single
> try-catch instead of using a try-catch inside the loop. However, if the
loop
> must continue to enumerate the items regardless of a single item's thrown
> exception then you must put the try-catch inside the loop. There is also a
> small perf hit in the catch block when the exception is caught because the
> runtime must examine each catch handler to determine if it is suitable to
> handle a given exception. If not handled the exception continues to
> propagate up the call stack, but there is still a non-zero hit in making
the
> decision not to handle the exception; small, but not nothing.
>
> Most of the perf hit comes from actually throwing the exception, not just
> from using a try-catch construct, or just from catching it. In the example

microsoft.public.dotnet.languages.csharp: Re: perf & Try Catch

> code you have, if an exception is never thrown then the difference a
single
> versus a nested try-catch would usually be unnoticable. Even in the nested
> case, if the inner catch handles the exception then there should be no
perf
> difference between the single vs. the nested handler because it is only
> thrown and caught once.
>
> However, if the inner catch rethrows the exception, then you now have two
> exceptions thrown, so the perf hit will be greater, approx twice as much
> perf hit. The reason is that the majority of the perf hit comes from the
> mechanics of walking the stack twice, the first time to locate a catch
> handler, and the second time from running downstream finally blocks before
> execution control is handed off the catch handler that deals with the
> exception. The stack walk on the 1st pass actually transitions through the
> kernel, makes system calls to determine if a debugger is attached, etc.
This
> is entirely non-local, so cache misses occur, pages may get faulted into
> memory, etc. This can happen each time an exception is thrown, so if you
do
> a lot of try-catch-throw your performance will degrade, one would expect
> in a linear fashion after the system has warmed up.
>
> As a double-caveat, I regard the extra perf hit from rethrowing an
exception
> to be the cost of doing business, and I will rethrow as necessary in order
> to add sufficient context to make correcting the problem as easy for the
end
> user as possible. This is because once you accept that you are in a
> non-performant path then the extra cost is negligible compared to the cost
> associated with an end user calling support saying "you know that error
> message that says 'null reference'? Yeah, well, what the heck does that
> mean?". In other words, my feeling is that clarity is more important than
a
> squeezing a few grams of perf fat out of the code. Of course, the
> particulars of the code and how it is used dictate the ultimate strategy
and
> structure of what you must do.
>
> Basically, the strategy I use is to never throw an exception unless there
is
> a problem that cannot be handled, and then use try-catch-wrap-throw as
> often as necessary to provide context as much as is necessary; once
started
> down the exception path I regard the extra perf hit as a minor issue.
>
>
> "Pohihihi" <pohihihi@hotmail.com> wrote in message
> news:%23T1MFJR7EHA.208@TK2MSFTNGP12.phx.gbl...
> >I was wondering what is the ill effect of using try catch in the code,
both
> >nested and simple big one.
> >
> > e.g.
> >
> > try
> > {
> > \\ whole app code goes here
> > } catch (Exception ee) {}
> >
> > -- AND --
> > try

```
> > {  
> >     try  
> >     {  
> >         try{...} catch(...){...}  
> >     }catch(Exception ee) {}  
> > } catch (Exception ee) {}  
> >  
>  
>
```