

Re: whats faster, initialize component, or form load?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-07/msg00397.html>

- *From:* "Cor Ligthert [MVP]" <notmyfirstname@xxxxxxxxxx>
 - *Date:* Wed, 4 Jul 2007 19:30:43 +0200
-

Alex,

Can you give me a real world situation where you initialize thousands of forms?

The video card and whatever offline devices even internet are in my 20% where parallel processing can be or is useful. I have never denied that. Although I have not the idea that many visitors here are creating programs for video cards.

Cor

"AlexS" <salexru2000NO@xxxxxxxxxxxxxxxxxxxxxxxx> schreef in bericht news:e7OmzNkvHHA.4800@xxxxxxxxxxxxxxxxxxxxxxxx

I believe you need to check how modern CPUs and graphics cards are built – they use parallel execution in several places. Or just look and compare 2 core and single core processors.

Point is, for every algorithm there is parallel version. Just like for every algorithm there is estimate of complexity like $o(n)$ or $o(n \log n)$ or whatever. Question if parallel version is faster than sequential, is usually answered in favor of parallel. However, this kind of research requires pretty significant effort and is worth the effort only for significant volumes of sufficiently important data to process. And also keep in mind that parallel algorithm usually doesn't resemble sequential version at all.

Suppose you need to initialize 1 form – parallel might not do any good there. However if you need to initialize thousands or millions of such forms, your only hope is parallel execution. Same applies for any kind of data.

All kinds of modern (and grid) computing are using parallel execution and algorithms. This is the only way to increase performance "ignoring" physical limitations of underlying hardware.

Re: whats faster, initialize component, or form load?

"Cor Ligthert [MVP]" <notmyfirstname@xxxxxxxx> wrote in message
news:Ow%23G5qivHHA.536@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

Peter,

Let say it than all what we in a much simpler way imo did you write in your first message you can win proces time with parallaliation. I say parallaliation cost forever procestime, because of that proces you can win throughput time. However be aware that this is only in certain circumstances.

The OP is writting about winning time not all whatever benefits that there can be more from multiprocessing (to use the old and in my idea still correct word for it).

I have still never heard of a successful results with parallaliation from internal (dependable) algoritmes although there are AFAIK spent billions for it in Japan on mainframe computers.

Cor

"Peter Duniho" <NpOeStPeAdM@xxxxxxxxxxxxxxxx> schreef in bericht
news:op.tuxjkrqn8jd0ej@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

On Tue, 03 Jul 2007 22:29:39 -0700, Cor Ligthert [MVP]
<notmyfirstname@xxxxxxxx> wrote:

Peter,

In 80% of the situations will parallaliation imo only slow down the process, just because the parallaliation needs to be processed. I assume that the second part of a processor will completely eat that.

And 98.6% of all statistics are completely made up.

Honestly, I don't find that "80%" comment to be particularly useful. I think that if you tried to apply parallelization to ALL code, that in fact even more than 80% of it would wind up slower. But not because of the overhead of handling multiple threads. Rather, simply because the work being done doesn't lend itself to parallelization. I

Re: whats faster, initialize component, or form load?

did point
out in my post that not all cases can be successfully
parallelized.

For those that can be parallelized, the question becomes
how much
processing is actually involved. There is overhead in context
switching, but if you can use the thread pool then you don't
have the
overhead of creating threads, and you don't have to have a
lot of
processing for the context switching to be worth it.

You do have to have a lot more processing that you
typically would
when simply initializing a class instance, but it appears to me
that we
have diverged from the original scenario at this point. You
and I are
both writing more generally, and I disagree that for those
scenarios in
which the algorithm lends itself to parallelization, that you
would
still fail in 80% of the situations to improve performance by
taking
advantage of that.

Let me make sure I'm clear: if we're talking about a
constructor that
initializes 15 value type fields, and by "parallelization" we're
talking
about making a secondary thread that initializes 8 of those
fields,
initializing the remaining 7 in the main thread (or worse, in
another
secondary thread), then I agree this isn't going to help
performance.
It takes so little time to initialize data that there's no way
starting
up a new thread (even if it's already created) can be faster
than just
initializing all 15 in one thread.

But the discussion is not about that scenario only, and the
general
statements need to be true for all code that is parallelizable.
Not
just the OP's question.

Re: whats faster, initialize component, or form load?

Multithreading can be helpful in by instance your given sample where the program needs to wait on an offline process..

It can be helpful for CPU-bound algorithms as well, assuming they can be parallelized. Not all CPU-bound algorithms meet that requirement, but I specifically excluded those from my discussion.

I find that telling about the hyperthreading processor a fable. There are in my computer at least 40 other tasks which are awake or running so that other part of the processor will in my idea never given to a multithreading thread (Or it should be with an OS where at the moment C# is not able to work).

You have two completely unrelated statements there. Let's look at the latter first, the question of how many processes you have running. Yes, on a Windows PC there are a number of processes always running. But it is false to think that just because they are there, running multiple threads in a single process won't improve performance.

Of those 40 processes, most are completely idle, or nearly so. They are waiting on some kind of i/o and have exactly no CPU load until that happens. Furthermore, even when they do want to use the CPU, they are unlikely to use their entire timeslice. On the other hand, a CPU-bound algorithm will use its entire timeslice. It would not be uncommon at all to find a CPU-bound, multi-threaded application that is able to consume close to 100% of the CPU time on all CPUs, since it would

Re: whats faster, initialize component, or form load?

normally be the only process on the computer that actually spends any significant amount of time using the CPU.

So, your "at least 40 other tasks" is a red herring, and has very little to do with the success or failure of multi-threading.

So, how about the hyperthreading question? Well, a number of points are relevant here:

— Yes, hyperthreading is not as good as having two CPUs. However, it is better than having just one, and I have both written and used code that shows that to be true.

— Some caveats do apply for using HT CPUs: HT cores share a cache, so you have to write your code carefully to ensure that multiple threads are not trashing each others cache. You can very easily write multi-threaded code that runs much worse on a HT CPU than on a single CPU. Also, because not all of the CPU is duplicated, only certain kinds of processing will see any benefit, and even of the kinds of processing that can benefit, some algorithms will do better than others. It's not like true independently multi-CPU systems, where one can achieve very close to optimal scaling proportional to the CPU count if the algorithm is implemented well.

— Most importantly, however, I never said anything about HT processors, nor do I see any reason that this discussion should be restricted by HT processors. Especially today, when dual-core is now the norm and many PCs are sold with four or eight cores (and in some cases, those cores are hyperthreaded!), thinking that one should not bother with multithreading just because on a single-core HT system there's little benefit is, well...fairly shortsighted.

Re: whats faster, initialize component, or form load?

Just my opinion,

You're welcome to it, of course. However, I'll suggest that there are some naive assumptions behind that opinion, at least as described in your post. When more than one CPU is available, there very often is real benefit for multi-threaded code when the algorithm is suited to parallelization. I think that claiming that 80% of those cases would wind up worse off when actually parallelized is a broadly incorrect statement.

Pete