

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

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-07/msg00322.html>

---

- *From:* "Cor Ligthert [MVP]" <[notmyfirstname@xxxxxxxxxx](mailto:notmyfirstname@xxxxxxxxxx)>
  - *Date:* Wed, 4 Jul 2007 07:29:39 +0200
- 

Peter,

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

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

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).

Just my opinion,

Cor

"Peter Duniho" <[NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx](mailto:NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx)> schreef in bericht [news:op.tuwzk20s8jd0ej@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:op.tuwzk20s8jd0ej@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

On Tue, 03 Jul 2007 16:28:39 -0700, Lit <[sql\\_agentman@xxxxxxxxxxxx](mailto:sql_agentman@xxxxxxxxxxxx)> wrote:

Interesting, how does asynchronously make it faster?  
please enlighten.

It depends. In some cases, it won't. But if you have multiple elements to your initialization that are mutually independent (that is, the results of some initialization are not required in order to perform other initialization), then you could see improved performance by processing them asynchronously for a variety of reasons. The most common being that on a multi-CPU computer, each asynchronous thread can do work in parallel with other asynchronous threads.

For example, suppose that when initializing you have two different

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

calculations you need to do, neither of which depend on each other. Suppose they both take 500 milliseconds. Doing them synchronously, you would take 1000 milliseconds, but doing them asynchronously you could theoretically do them both in only 500 milliseconds, assuming you have at least two CPUs to handle the processing.

Another common reason would be if the initialization required a variety of i/o to different subsystems or remote entities. An example of this might be if you have to query two different web servers. Even if you only have one CPU, you could still see an improvement in speed, for similar reasons as the CPU-bound case. The difference being that in this case you are waiting for a remote server instead of a CPU. But the basic logic is the same: if you have to wait 500 milliseconds for each server, doing the requests synchronously would delay you 1000 milliseconds, but doing them asynchronously in parallel would delay you only 500 milliseconds.

Now, all that said, there's little about your problem description that suggests to me that you would gain much benefit from changing your initialization at all, never mind making some or all of it happen asynchronously. In particular, the only thing that sounds remotely slow is the "check for a certain file", so you have no other slow item to do in parallel with that. Even that operation is unlikely to be slow enough for you to be concerned with its performance anyway. Say, for example, you had to make two such checks. Even so, unless you were trying to instantiate a number of this class all at once (like thousands or more), I doubt that you'd achieve any practical performance improvement by parallelizing the operations.

Basically, it appears to me that you are worrying about something that simply does not warrant concern.

Pete