

Re: threaded application not using all processors

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.performance/2005-05/msg00118.htm>

- *From:* "Ian Griffiths [C# MVP]" <ian-interact-sw@xxxxxxxxxxxxxx>
 - *Date:* Tue, 31 May 2005 18:33:21 +0100
-

cbenmichael@xxxxxxxx wrote

> I have a main thread that gets requests using remoting, and then
> spawns worker threads (currently limited to 4), which run a
> long running non-linear regression analysis.

How are you spawning the threads? Can you show us a code snippet?

> All the worker threads seem to be running in a single processor.

How are you determining this? Are you looking at the CPU usage in task manager or are you doing something more sophisticated?

Are you certain that all 4 threads are actually running simultaneously? Have you checked to make sure they are all making progress, and that there isn't some synchronization issue making them take it in turns? I'd use something like Process Explorer (a free tool from www.sysinternals.com) to look at the per-thread CPU usage.

> I have set the environment variable "COMPLUS_BuildFlavor", and have
> verified that the process is using the mscorsvr.dll.

That should really only affect your GC behaviour. The workstation version of the CLR is perfectly capable of using multiple CPUs, so long as it's your code that's doing the actual work.

> I read somewhere that if the threads are running in a .net form,
> then they will always run in a single processor

Err, no. It sounds like that message got badly garbled somewhere along the way.

In a Windows Forms application, it is vitally important that any access to the UI is done on the same *thread*. But there are no limitations as to

Re: threaded application not using all processors

which processor that thread runs on. And you're welcome to have as many worker threads as you like so long as they don't try to update the UI. So it's entirely possible for a Windows Forms app to make use of multiple CPUs.

> – Are there other parts of the framework which would cause all
> the threads to run in one processor?

There's a Windows feature: thread affinity. It's possible to tell Windows to

> – Why is this happening and how can I fix :)?

No idea.

Here's some simple code that fires up two threads, and which does pointless busy work that consumes 100% of the CPU on my dual-proc system (i.e. it is using both CPUs completely) for a minute or two:

```
using System;
using System.Threading;

class app
{
    static void DoWork()
    {
        int i = 0;
        double total = 0;
        while (i < 1000000000)
        { ++i; total += 1.0 / i; }
        Console.WriteLine(total);
    }

    static void Main()
    {
        ThreadStart threadProc = new ThreadStart(DoWork);
        Thread t1 = new Thread(threadProc);
        Thread t2 = new Thread(threadProc);

        t1.Start();
        t2.Start();
        Console.WriteLine("Both threads running.");
        t1.Join();
        t2.Join();
        Console.WriteLine("Done");
    }
}
```

So I suggest you start by comparing that with your code to see how it differs.

Re: threaded application not using all processors

I compiled this with no special flags. I didn't do anything special to the environment. I just ran the code on a Windows Server 2003 box with version 1.1 of the .NET Framework installed.

—
Ian Griffiths – <http://www.interact-sw.co.uk/iangblog/>

• **References:**

- ◆ ***threaded application not using all processors***

◇ From: cbenmichael

- Prev by Date: ***threaded application not using all processors***
- Previous by thread: ***threaded application not using all processors***
- Index(es):
 - ◆ ***Date***
 - ◆ ***Thread***