

Re: numerically intensive app doesn't benefit from multi threading

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2006-12/msg00131.ht>

- *From:* JC <JC@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 4 Dec 2006 15:53:01 -0800
-

Here is the info from Dell:

PROCESSOR, 80547, PENTIUM 4 PRESCOTT DT, Pentium 4 Prescott DT, 3.0GHZ, 1 MEGB, 800FSB, SOCKET T, E0, MALE

Can you tell if it's hyperthreading?

Thx.

"JC" wrote:

How can you tell?

"m" wrote:

Are your 2 CPUs a single hyperthreaded CPU?

"JC" <JC@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:0E9C29A6-F054-4E39-98E3-2FB506F27827@xxxxxxxxxxxxxxxxxxxxx

This may be related to my another post in this group here
"program runs
almost twice as slow on 2 CPU machine when 1 CPU is
used".

When I run single thread version of the program on 2 CPU
concurrently, it's
almost twice as slow as when I run one instance only. When
I run multi
thread version of the program with 2 thread on 2 CPU, it's
same as running
single thread on 1 CPU.

"JC" wrote:

Re: numerically intensive app doesn't benefit from multi threading

I have a numerically intensive app where there is a large loop. I want to use multithreading to parallelize the loop. Say we have two threads, the 1st one will do 1st half the loop, the 2nd thread will do the other half. The multithreaded loop looks like:

```
for (i = indexBeginForThread; i <
indexEndForThread; i++) {
do_numerical_stuff;
update_common_data;
}
```

The 1st part 'do_numerical_stuff', taking most of the time, and requires no mutex to synchronize access to data. The 2nd part 'update_common_data' takes little time, and requires mutex to synchronize access to data. You would expect such a program will have a speed up of 40–50% when running on 2 CPU. But there is no speed up at all.

I am not sure what is causing no speed up at all.

I inserted 'clock()' function to time the time spent on 'do_numerical_stuff' and 'update_common_data' in both threads. (I am not sure whether this works as I don't know whether clock() is multithread safe). I found that in both threads, the total time spent on 'do_numerical_stuff' is the same as when the program is run in single thread mode. The total time spent on 'update_common_data' is tiny. So there shouldn't be a problem of waiting on the mutex when doing 'update_common_data'.

Re: numerically intensive app doesn't benefit from multi threading

Theoretically "do_numerical_stuff" in both thread should use half the time as in single thread mode.

The whole loop takes 2 seconds in a test run, and it should benefit from multi-threading.

Any insights?

Thanks a lot.