Subject: Re: IDL Matrix Multiply and Dual-Core CPUs
Posted by s.haenger on Thu, 17 Jul 2008 08:42:21 GMT
View Forum Message <> Reply to Message

On 20 Mai, 20:19, Karl <Karl.W.Schu...@gmail.com> wrote:
> On May 20, 1:21 am, s.haen...@gmail.com wrote:
>
>
>
>> On 9 Mai, 20:34, Pierre <pierre.villene...@gmail.com> wrote:
>
>>> Hi Samuel,
>
>>> I saw a very similar problem with my quad-corePC running XP (32 bit)
>>> with 4gigs of ram. I re-ran my test script on our two-core, 4-gig
>>> linux box and got similar results with just slightly different array
>>> sizes. Here is the script I ran:
>
>>> cpu, /reset
>
>>> help, !cpu, /str
>
>>> Nk = 258
>>> K = fltarr(Nk, Nk)
>
>>> ;
>>> ; Case 1.
>>> ;
>>> Npix = 129047
>>> d = fltarr(Npix, Nk)
>>> t0 = systime(1)
>
>>> d #= K
>
>>> t1 = systime(1) - t0
>
>>> print, 'Case #1: ', Npix, t1
>
>>> ;
>>> ; Case 2.
>>> ;
>>> Npix = Npix + 1
>>> d = fltarr(Npix, Nk)
>>> t0 = systime(1)
>
>>> d #= K
>
>>> t2 = systime(1) - t0

>
>>> print, 'Case #2: ', Npix, t2
>
>>> On each of our computers case #2 used all available cores while case
>>> #1 only used onecore. The only difference between them is the
>>> dimension of one of the arrays (Npix) is simply incremented by one.
>>> The total memory used by theIDLprocess during this test is never
>>> more and two-hundred megs or so. There is no way this problem is due
>>> to a lack of physical memory. The sizes of these arrays are also
>>> significantly larger then the default minimum number of elements
>>> (default = 10000) required to enable multi-threading.
>
>>> Any ideas?
>>> Pierre
>
>> It's not a Windows Problem. We have the same Problem also with
>> Ubuntu...
>
> There's a lot of speculation to follow, so be warned.
>
> Making sure that using multiple threads is really faster isn't very
> straightforward. There's lot of overhead involved when splitting the
> problem into threads. There is a lot of data movement, creating
> tasks, waiting for them all to complete, etc. There are also other
> factors such as memory page sizes, cache lines, etc. So, using
> multiple threads isn't always a win, as hinted by the minimum data
> size.
>
> I would suppose that there is a set of "heuristics" that are used to
> decide whether to multi-thread or not, based on the data size, shape,
> layout and the algorithm being implemented. I wasn't very closely
> involved, but when this was being developed, there were some very
> interesting surprises about what sorts of problems multi-threading
> would yield a net gain and what sort of problems ended up being a net
> loss.
>
> There's probably a lot of effort being made to avoid ending up with a
> slower result when using multiple threads. It might be too
> conservative, or the decision might be wrong due to a bug. But it
> might even be correct and that changing the data size that least
> little bit in this example ends up changing the decision as to whether
> to use multiple threads or not. I find it odd, given the data in the
> example, but it is possible.
>
> The only way you'll know is to ask ITTVIS why MT was rejected for one
> array and not for the other.
>
> Karl

Thanks a lot for the response.
Actually we did ask the ITTVIS Support guys and they responded pretty much the same things as you (except for the bug thing) :-)
I think we have to accept, that IDL just uses MT when it wants to :-)

Samuel
