Potential speedup for MKL operations on AMD Ryzen/Threadripper/Epyc

Fri, 12/06/2019 - 15:41 #1

gdodge77

Potential speedup for MKL operations on AMD Ryzen/Threadripper/Epyc

Hi all, thought I'd share this in case anyone is using newer AMD hardware. Apparently Intel's MKL library runs a hardware check, and by default will prevent the AVX2 unit on AMD processors from functioning. For a detailed example of the speedup in some workflows, see here.

On linux, the workaround is quite simple, set the environment variable below. Note that intel may disable this debug command in the future, but for now it works really well.

```
export MKL_DEBUG_CPU_TYPE=5
```

Cheers,

-Greg

timgrant

Thank you Greg - this is very

Thank you Greg - this is very useful!

vamseedharr

Thank you Greg

Thank you Greg. Great info. There seems to be a consensus about this across several forums. This workaround actually improves the performance of Ryzens so much they are better than their Intel counterparts. Although, the caveat at the end of that link puts me in a bind while I try to assemble a workstation.

"**Conclusion (BIG Caveat!)**

I have to reiterate, MKL_DEBUG_CPU_TYPE is an undocumented environment variable. That means that Intel can remove it at any time without warning. And, they have every right to do that! It is obviously intended for internal debugging, not for running with better performance on AMD hardware. **It is also possible that the resulting code path has some precision loss or other problems on AMD hardware. I have not tested for that!**

The best solution for running numerical intensive code on AMD CPU's is to try working with AMD's BLIS library if you can. Version 2.0 of BLIS gave very good performance in my recent testing on the new 3rd gen Threadripper. For the numpy testing above it would be great to be able to use the BLIS v2.0 library with Anaconda Python the same way that I used OpenBLAS. Someone just needs to setup the conda package with the proper hooks to set it as default BLAS. I don't have the time or expertise to do this myself, so, if you can do it then please do! and let me know about it!"

Tim, any thoughts on this? or should I just play it safe and stick to an Intel processor?

Thanks,

Vamsee

Wed, 03/11/2020 - 21:16 (Reply to #3)

timgrant

Hi Vamsee,

Hi Vamsee,

I have not done any testing on AMD myself, so don't feel I can really comment in detail. However, I presume this fix is linked to the version of the intel compiler / MKL use for compilation, and so if you do take a plunge and go for AMD I can always compile you a version with an intel compiler version that works?

Thanks

Tim

Thu, 03/12/2020 - 01:14 (Reply to #4)

vamseedharr

Jumped on the AMD bandwagon

Hi Tim,

Thank you for the reply. I assembled an AMD system and have been testing cisTEM on it. Seems to be working pretty decently actually with the stock version but I wouldn't mind testing something specific to the AMDs. You can either send me a compiled version or the instructions to compile them. Thanks again.

Vamsee

Thu, 03/12/2020 - 17:19 (Reply to #5)

timgrant

Hi Vamsee,

Hi Vamsee,

You are using the environment variable discussed above yes? In that case, there is no need for you to have different binaries. What I meant, was that if we use newer version of the intel compiler for future releases and that breaks the use of this environment variable, I would be able to specially compile using an older version.

Thanks,

Tim

vamseedharr

Ah ok! Yes, I am applying the

Ah ok! Yes, I am applying the environment variable changes on my machine right now and it seems to be doing the job. I'll get in touch if and when I run into issues with future releases. Thank you

Vamsee