OpenACC
More Science, Less Programming

OpenACC Course October 2017. Lecture 1 Q&As.

Question	Response
I am currently working on accelerating code compiled in gcc, in your experience should I grab the gcc-7 or try to compile the code with the pgi-c ?	The GCC compilers are lagging behind the PGI compilers in terms of OpenACC feature implementations so I'd recommend that you use the PGI compilers. PGI compilers provide community version which is free and you can use it to compile OpenACC codes.
New to OpenACC. Other than the PGI compiler, what other compilers support OpenACC?	You can find all the supported compilers here, https://www.openacc.org/tools
Is it supporting Nvidia GPU on TK1? I think, it must be	Currently there isn't an OpenACC implementation that supports ARM processors including TK1. PGI is considering adding this support sometime in the future, but for now, you'll need to use CUDA.
OpenACC directives work with intel xeon phi platform?	Xeon Phi is treated as a MultiCore x86 CPU. With PGI you can use the "-ta=multicore" flag to target multicore CPUs when using OpenACC.
Does it have any application in field of Software Engineering?	In my opinion OpenACC enables good software engineering practices by enabling you to write a single source code, which is more maintainable than having to maintain different code bases for CPUs, GPUs, whatever.
Does the CPU comparisons include simd vectorizations?	I think that the CPU comparisons include standard vectorisation that the PGI compiler applies, but no specific hand-coded vectorisation optimisations or intrinsic work.
Does OpenMP also enable parallelization on GPU?	OpenMP does have support for GPUs, but the compilers are just now becoming available. There's work going on to support it in both CLANG and GCC, but neither ship with it enabled by default. OpenACC and OpenMP are very similar programming models, although OpenACC is a bit higher level, which we believe makes it more portable and more scalable in the long run.
Is the NVIDIA OpenACC toolkit needed, or can I just use the CUDA library?	You can download PGI Community Edition to get started with OpenACC
Will examples also be in FORTRAN?	Yes, both C and Fortran will be available
Are there VMs to practice OpenACC when we don't have access to HPC?	As part of this course you will have the opportunity to try out OpenACC in our qwiklabs system, which runs on Amazon AWS instances. Unfortunately, GPU passthrough isn't available with most VMs, but you can use it by installing the compiler on any of the GPU-enabled cloud providers. If you just want to learn the programming model, you can build OpenACC to target multi-core CPUs, so any VM with Linux and the PGI compiler should work.
And work with old Xeon phi coprocessor?	No, we (PGI) did an unreleased Alpha compiler several which we demoed several years ago at the Supercomputing conference. However, we did not productize it since Intel was moving Phi away from the co-processor so it didn't make sense to continue working on a product was going to be discontinued.
Does OpenACC take advantage of avx-512 on Xeon CPU or phi?	PGI is working on AVX-512 support but this support has not yet been released.

Question	Response
Can we do a hybrid approach, OpenACC%2B MPI%2BOpenMP on CPU%2BGPU nodes?	Yes, I presented some examples of this at the GPU Technology Conference earlier this year. You can use OpenMP to control CPU threading, OpenACC within those threads to accelerate on one or more GPUs, and even mix in some MPI, CUDA, or GPU libraries. In the "Parallel Programming with OpenACC" book you'll find several examples that should get you going.
Are you saying we can mix OpenACC & CUDA in a single program?	Correct
For a typical explicit FVM based CFD code, how much more speed up can one expect by re-writing it in CUDA, compared to using OpenACC on the old code?	The performance of the code many vary. On average OpenACC give about 80% of CUDA performance, and now with unified memory feature we see matching performance of OpenACC and CUDA
Does OpenACC mix well with MPI?	Absolutely! Not only that, but you can even use it with GPU-aware MPI. Check out NVIDIA's parallel forall blog, I believe there are several blog posts on this. https://devblogs.nvidia.com/parallelforall
Can you create libraries with PGI that can be linked with VS?	Yes. There are general issues with inter-language calling and you will need to compile with "-ta=tesla:nordc" which disable so features such as device routines and accessing global data, but we do have several large customers that integrate PGI built OpenACC libraries with their larger applications on Windows.
Any plan to support C++ in Windows OS?	PGI does not currently have plans to support C on Windows.
Can OpenACC libraries be used with Fortran and C++?	There are OpenACC Fortran and C compilers. OpenACC directives are implemented by the compiler.
What are the flags with which the program should be compiled assuming OpenACC libraries are installed	If you're using the PGI compiler, you can use -acc to turn on building for NVIDIA GPUs and the host CPU. You can also use the -ta flag to tell it more about what machine you want to be able to run on. For GCC, I believe the flag is -fopenacc.
How is it different than OpenMP?	There are some major differences in the current versions of OpenACC and OpenMP. You will hear about them during the presentation. I am listing below what I find as the major differences. 1. OpenACC supports both prescriptive and descriptive pragmas whereas OpenMP currently only supports prescriptive pragmas. 2. The OpenACC have more features for programming GPUs then OpenMP.
Can OpenACC be easily installed on a local account on a cluster without SUDO priviledges? If so, how do you do so?	OpenACC is not a standalone library installation like MPI. It ships with the compilers. So as long as you have one the compilers listed here, https://www.openacc.org/tools, you can start using OpenACC in your code. Installing compilers on a cluster without sudo privileges is not straight forward. I'd recommend to talk with the system administrator with sudo privileges to get one of the compilers installed.
Will OpenACC support python later?	That's actually something we're exploring, but at the moment only C, C , and Fortran are supported. You should check out Numba though, because it has capabilities very similar to OpenACC for supporting GPUs.

OpenACC
More Science, Less Programming

OpenACC Course October 2017. Lecture 1 Q&As.

Question	Response
Will ARM support ACC?	At the moment I'm not aware of any ARM compilers for OpenACC. There was previously support in the PathScale compiler for OpenACC on Cavium ARM CPUs, but unfortunately the company behind that compiler went out of business and I don't think anyone has picked up that effort.
Is there multi-gpu support with OpenACC?	Yes, check out my talk on the subject: http://on-demand-gtc.gputechconf.com/gtc-quicklink/9TEoXz
Is it still possible to fine level optimizations with OpenACC?	OpenACC has a "loop" directive where you can tell the compiler how you want it decomposed and really tune the code for the best performance. Usually the compiler's guess will get pretty close to full performance, but there are knobs to turn for improved performance. Come to the November 2 lecture, I'll be talking about this then.
The Intel Phi work with the older generation or only the new ones	Only the KNL with the caveat that PGI does not officially support KNL and have not yet released AVX-512 support.
What does OpenACC use under the hood?	It depends on the target architecture you specify during compilation. For e.g. NVIDIA GPUs the code will be translated into CUDA kernels.
Will OpenACC use both CPUs?	The PGI compiler supports building OpenACC codes for multicore CPUs. In the hands-on lab that will be available at the end of this lecture you will see such an example.
Is GCC support for OpenACC at a mature level?	It's improving, but the performance still usually lags behind the PGI and Cray compilers. There's exceptions where GCC wins, but most of the time you have to be much more explicit with GCC in telling it how you want your loops parallelized. There's ongoing work to improve GCC's automatic parallelization capabilities and performance.
Is the GCC 7 implementation on the same level as the PGI one?	In general, you really have to hold GCC's hands a lot. PGI will often make smarter decisions on how to optimize the OpenACC code automatically, but GCC often requires you to tell the compiler exactly how to optimize your code. There's work ongoing to improve GCC's performance and on occasion I've see GCC even beat PGI, but most of the time PGI will beat GCC by a wide margin, at least when building for NVIDIA GPUs.
OpenACC use both CPU and GPU?	Yes, but it cannot currently automatically divide your loops across both at the same time. That's a pretty significant challenge for the compiler. I know of some research institutions looking at whether that would be possible. You can certainly use OpenACC on your loops and build for the CPU or GPU and choose which one based on what's available on your machine.
How the performance is improved in medical imaging with PowerGrid?	you can read the complete story on openacc.org: https://www.openacc.org/success-stories/powergrid
Does the Intel compiler have any support for OpenACC?	Not currently, the list of compilers with support is at: https://www.openacc.org/tools
Can the PGI compiler take advantage of the dedicated TPU on Nvidia Volta based GPU's?	Not at this time. The tensor cores are so specialized that they're only really useful in very specific situations. I'm not aware of any work to detect these specific situations and automatically use them. Your best bet for using the Tensor Cores is via cuBLAS, cuDNN, and several common deep learning frameworks, such as caffe2, tensorflow, pytorch, etc.

Question	Response
Can OpenACC benefit CPU-only setting? If there is no GPU, OpenACC will only generate serial code?	Yes, actually in the first lab, which will become available after this lecture, you will be taking a small code and parallelizing it on an 8-core CPU using only OpenACC.
so if I want to make simple loop parallel with pragma, I can use #pragma acc kernels, but I can also use openmp #pragma parallel. I am a bit confused about pros and cons of those two methods and how to choose openmp vs OpenACC. (my initial understanding was that OpenACC was for GPUs and openmp for CPUS)	OpenMP was originally designed for shared memory machines and later extended to support other types of parallelism, including support for offloading to GPUs. With OpenMP, you the programmer are telling the compiler how to parallelize your loops and certain legacy limitations can sometimes limit the scalability. OpenACC wasn't explicitly designed for GPUs, but it was designed from the beginning to work on GPU-like machines. It forces you to write your code in a parallel way. When I talk to users who have tried both with the same codes, I often hear that the developers feel like OpenACC is simpler to use, because the compiler is doing the analysis and planning the parallelism for you, where OpenMP requires you to do more work.
Can you have race-conditions or deadlocks with OpenACC?	OpenACC is designed to force the developers to remove any potential race conditions or conditions that could cause deadlock.
Does C++ have a standard restrict keyword yet?	No, though PGI does support restrict in C as an extension.
Which flag should I use with gcc?	I haven't used GCC 7.0 so this may have changed, but with GCC 6.3 I used "-foffload=-lm -lm -fopenacc -fopenacc-dim=1024:1" You may not need to set the "dim" option since I believe 7.0 has gotten a lot better with loop scheduling.
Which generation(s) of Tesla accelerators does `-ta=tesla` target? I've had all manners of fun supporting CUDA code across multiple microarchitecture generations.	By default with PGI 17.4 Community edition, "-ta=tesla" supports Fermi, Kepler, and Maxwell since by default we use CUDA 7.5. Using "-ta=tesla:cuda8.0" will also add Pascal. With PGI 17.7, we also added support for Volta. You can also select which devices to compile for using "-ta=tesla:cc35,c60,..etc" or for a single target device using "-ta=tesla:cc35". See "pgcc -help -ta" for a full list of options and target devices.
Is there support for heterogeneous hardware i.e. parallelisation over cpu + gpu?	Yes. PGI's implementation of OpenACC does support both multicore CPU as well as NVIDIA GPUs.
What is the approach for multi-GPU computers? Any difference or the procedure is the same.	I actually did a talk on this a few months ago. Take a look at http://on-demand-gtc.gputechconf.com/gtc-quicklink/9TEoXz.
How does hyper-threading affect OpenACC optimization?	Very interesting question. I'll answer based on my experience on Linux, but I'm not certain if you try on other architectures. On a Linux machine hyper-threading will show up as 2 CPUs, so an 8-core hyperthreaded CPU will be presented by the OS as 16 CPUs and by default the OpenACC runtime will try to run across all of them. Depending on the specifics of your application, it's not uncommon to get better performance running only 1 thread per physical CPU or disabling hyperthreading altogether.
Does OpenACC work with SSE vectorization?	When targeting x86 multicore CPUs with OpenACC, the PGI will use SSE vectorization.

Question	Response
How does -ta=multicore compare with OpenMP	We have tested against a range of applications that have both OpenACC and OpenMP and the performance is usually quite close, with OpenACC winning sometimes and OpenMP other. Usually they're essentially the same performance. OpenMP supports thread affinity and OpenACC does not, so that's one place where OpenMP can sometimes take the lead.
What happens when there are underlying hardware changes post-compilation of OpenACC enabled codes? Specifically, if a different type of GPU (AMD instead of NVIDIA, or vice-versa) is added to the system or removed? Will the code just grab whatever is available?	Depending upon how you built the code, it's possible that you need to recompile for the new target. By default, PGI does create multiple targets within a single binary for multiple NVIDIA GPUs (Kepler, Fermi, Maxwell), but also supports Pascal and Volta. If you compiled targeting the new device, no need to recompile. If not, then you just need to recompile. We (PGI) do not currently support AMD devices.
Will the course talk about "kernels" vs. "parallel" directives, one being a "suggestion" and one being an instruction for how to compile something?	I'm going to be the instructor on the 3rd week. If it's not covered by that point, I'll make sure it's covered. You've got the gist of it though: with kernels the compiler is in control and must figure out what it can and can't parallelize and with parallel the programmer takes more control and gives stronger guarantees to the compiler.
OpenACC seems wonderful! Any plans to add OpenACC support to GPUs from other vendors besides NVIDIA?	PGI has no plans on supporting non-NVIDIA GPU, though other compilers certainly can. We (PGI) focus on HPC where NVIDIA is the dominate company for acceleration and we haven't found a business case to support other GPUs.
Does opencc allow distributed parallel programming or just shared parallel programs?	OpenACC is strictly single-node. For multi-node programs you'll usually see OpenACC MPI.
Does OpenACC have a mechanism to send data from one GPU to another?	Yes. There are data and update directives. These will be discussed in the next lecture.
Can I mix NVIDIA and AMD GPU using OpenACC? Since you explained that you don't need specific hardware instructions like CUDA programming, I guess OpenACC can handle the hardware behind scenes.	Yes, the target architecture will be handled behind the scenes. For the PGI compiler you can specify the target architecture using the -ta compiler option. Like this, you can e.g target an NVIDIA or AMD GPU, or also a multicore CPU. In the following link you can also find some information about nvidia and radeon targets: https://www.pgroup.com/lit/presentations/ieee_webinar_dec2013_slides.pdf
Is there a way to limit the resources of CPUs targeted (e.g., use only half of available memory/cores etc)	Yes. With the PGI compiler you can use the environment variable ACC_NUM_CORES to say how many cores to use. There's an example of this in the hands-on lab associated with today's lecture.
Can we use the CPU and GPU at the *same time*	The compiler won't automatically parallelize over both multiple cores of a CPU and a GPU. The difficulty being the managing the data between the discrete memories. Typically, users with use MPI OpenACC to parallelize across multiple CPU cores and GPUs.
Can the host code on an OpenACC application be written in C++ and the OpenACC in ANSI C?	OpenACC can be used in ANSI C as well as C code. Both is possible.

Question	Response
Will OpenACC be integrated into OpenMP?	This is a very politically charged subject. At this point in time, however, the two standards have moved their own directions long enough that it's very unlikely they'll ever become a single group. I work on both standards bodies and try to keep them from doing things that are fundamentally incompatible with the other, but I don't expect that there will ever be a single group again.
Could OpenMP and OpenACC be used together to achieve greater performance?	Absolutely, but not on the same loop. I show an example of how to use OpenMP and OpenACC to manage multiple GPUs in http://on-demand-gtc.gputechconf.com/gtc-quicklink/9TEoXz.
Does OpenACC support AMD GPUs?	Keep in mind that OpenACC is a standard with individual compiler implementations targeting particular device. OpenACC can work on AMD GPUs, but the PGI compiler does not currently support AMD. We dropped support after the 16.10 compiler since we couldn't make a business case for continuing this support.
Does OpenACC work well with FFTW/cuFFT?	Yes. You can use OpenACC's `host_data` directive to send data to any GPU-aware library. There's an example in one of the OpenACC books showing how to interface an OpenACC kernel with cuFFT to do an image filter.
Can OpenACC partition a loop between both the CPU and the GPU? I do not mean either or, but run on both at the same time.	No. There's some research into this capability, but it's only research at this point. Not only is there a load balancing problem, but you need to worry about in which memory the data lives. With NVIDIA's unified memory on recent architectures, this may become easier, but it's still a ways off.
Can I use OpenMP and OpenACC together? For example, inside an OpenMP loop, there is anther loop, can I use OpenACC to parallelize the inner loop?	Yes, OpenMP and OpenACC can be used together. Using a combination of OpenMP and OpenACC, e.g. multiple GPUs can be used.
Does OpenACC have a mechanism to use heterogeneous clusters automatically?	With OpenACC you can compile your code for different target architectures like GPUs or also multi-core CPU. When compiling your code, you can specify different target architectures using the -ta option (PGI compiler). Like this you can have one source and can compile your code for heterogeneous architectures.
is it possible to run simultaneously in the CPU and the GPU using OpenACC?	Yes, but you cannot currently automatically split the same loop across both. You can manually break up operations and put some on the GPU and leave some on the CPU though. There's research into making this automatic, but it's just research at this point in time.
OpenMP supports GPU offloading since version 4.0. How does it compare with OpenACC ?	You can write GPU code now using OpenMP and I've had some success doing so. The GPU compilers for OpenMP are less mature than OpenACC, so the performance often lags. What's more important though is that when you write code with OpenMP you'll be writing very explicitly what you want the compiler to do, but with OpenACC you rely a lot on the smarts of the compiler. This means that with OpenMP you will need different directives for each distinct type of machine but with OpenACC we rely on the compiler to take the same code and optimize it for any machine you may wish to run on.
Does OpenACC programs runs on heterogeneous computing platforms(mixed GPU and CPU	Yes. Support for NVIDIA GPUs is very mature. PGI has also announced that they're working on supporting Xeon Phi.

Question	Response
accelerators - Nvidia Tesla K20 and Intel Xeon Phi)?	
If I set openacc directive for a loop which contains a part which involves data dependency and other which is embarrassingly parallel, will it selectively parallelize the part which has no data dependency, or should i put the directives at better location?	If you use the OpenACC kernels directive around this code, the compiler is required to analyze the code and decide whether it can be parallelized. If the compiler sees a data dependency (or even the threat of one) it will leave that section of the code unparallelized and only parallelize the part that it knows is safe to do so. If you *know* that something is unsafe to parallelize, you can use 'acc loop seq` to tell the compiler "please run this loop sequentially".
Does PGI support OpenCL devices? If yes, is OpenCL 2.x supported as targets?	We (PGI) have used OpenCL in the past, but is not needed for our current targets. We've been moving to using LLVM code device generation instead of targeting either OpenCL or CUDA.
Does OpenACC support any FPGA models leveraged within accelerator scenarios?	I'm not aware of any OpenACC compilers for FPGAs. It's possible that some of the research compilers, such as OpenARC or OpenUH have support though.
How can I specify the stream number that the parallel section runs into? (kernel <<< grid, block, 0, stream >>> (…, dev2, …))	You'd use the "async(id)" clause with a queue id where each queue mapping to a CUDA stream. This should be covered in the third lecture. Also, if you need to map a queue to a particular CUDA stream, then you can use the API "acc_get_cuda_stream (int async)" to get the stream the async queue is using or "acc_set_cuda_stream(int async, void * stream) to set a queue to a particular stream
Are slides available for download?	Yes, go here: https://www.openacc.org/sites/default/files/inline-files/OpenACC_Day1.pptx
Can OpenACC utilise memcopies concurrently with compute?	Yes. OpenACC uses the 'async' clause to make work happen on both the CPU and GPU simultaneously, including data movements. Search on http://on-demand-gtc.gputechconf.com for OpenACC talks, I've given several that show this feature, but I'm not sure I can point you to a specific one right this moment.
Is there any BLAS library using OpenACC?	I'm not aware of any BLAS libraries using OpenACC internally, although it could certainly be added to something like OpenBLAS. On GPU machines you can use cuBLAS with OpenACC to get the best possible performance.
About the example, do we abandon Gauss-Seidel and prefer Jacobi with parallel programming?	I'm not familiar with Gauss-Seidel, but if it's can be parallelized then there shouldn't be any reason why you could use it with OpenACC.
Can I watch this webinar and read the resources later again? where is the link?	The slides are posted at https://www.openacc.org/sites/default/files/inline-files/OpenACC_Day1.pptx. Check back on the course webpage later today, and probably the OpenACC youtube channel, for the videos.
Can I use a GTX 1060 to try this or is there an specific NVIDIA GPU for this.	Yes, you're good to go!
OpenMP 4.5 can target GPUs. Should we expect that OpenMP would eventually absorb OpenACC?	That's a very politically charged question. They both have offloading support, but for a variety of reasons I think momentum will continue to push both forward as two separate specs.
Intel compilers, then I understand are not supported.	Correct, Intel has not announced any support for OpenACC directives. It will, however, ignore directives it doesn't understand, so you can still build a code that has OpenACC pragmas in it.

Question	Response
What embedded SOCs work with OpenACC?	Sunway's compiler targets their SOC.
Does OpenACC supports dynamic parallel programming from GPU	Yes and no. OpenACC supports nesting parallelism within parallelism. Unfortunately, I'm not sure if any of the compilers actually do anything useful with that. =(
what's the lab URL again?	You can find the lab here: https://nvlabs.qwiklab.com/
Does OpenACC have an open interoperability?	Please see https://github.com/jefflarkin/openacc-interoperability
What is the max cores we can go to in the lab?	The lab systems have 8 physical cores. You can have the runtime over-saturate the cores by setting ACC_NUM_CORES to larger values, but may not see any additional speed-up to contention on the cores.
What site does the discount code for the OpenACC book apply?	only informit site that mentioned on the slide
Any compatibility with Apple's Xcode	Not that I'm aware.
Could you comment on the implicit reduction in the lab? I suppose the 'max' is done via a reduction, but there is another line in the loop. Is there a parallel loop running alongside the reduction?	When you get a little farther along to the point that it's running on the GPU when you'll find out is that the compiler usually reorganizes the code a bit to handle the parallel reduction, even generating a specific GPU kernel just for that operation.