OpenACC
More Science, Less Programming

OpenACC Course October 2018
Lecture 1 Q&As

Question	Response
Which operating system works with OpenACC: OS, Windows or Linux?	Linux or Windows. PGI doesn't include GPU support on iOS but you can target multicore CPUs.
OpenACC vs. CUDA, which should I use for my computational fluid code?	OpenACC is going to easier to start with since it's a simpler model and requires less rewriting of code. Though, OpenACC is inter-operable with CUDA so later, if you want to use CUDA for some performance critical loops that you want to hand-tune, you can intermix the two.
What is the difference between OpenACC and CUDA?	OpenACC is a directives-based model that is designed to simply onboarding to accelerators and parallel programming in general. CUDA is a low-level programming language that allows for full customization of the code, but requires more development time.
Is there a way to use OpenACC together with the AVX 2 extensions? Or does OpenACC use AVX 2 under the hood?	That will depend on the target accelerator and the compiler you're using. When targeting a NVIDIA GPU, then no, it would not use AVX-2. For the PGI compiler when targeting a multicore CPU, then the compiler will auto-vectorize the code and use AVX-2 depending on the target CPU. It wouldn't be recommended to use AVX-2 intrinsic directly since it would limit the portability of your code.
Does OpenACC support KNL (may it rest in piece)?	PGI does support using OpenACC on a KNL by targeting the multicore CPU (-ta=multicore). The caveat being that PGI didn't tune for performance on KNL, instead focusing on Skylake.
What about ARM CPU support for OpenACC?	It is possible for OpenACC to run on ARM. However, I don't believe that there is a current compiler implementation that supports it.
I see "AMD GPU" on the list. Is there a current compiler that will compile OpenACC to an AMD GPU? GNU?	GNU does support AMD, though I don't know the current state.
Is Intel Many Integrated Core architecture supported?	Yes, OpenACC runs on multicore. Jeff will cover it in a bit.
Does OpenACC C++ use CUDA internally?	Under the hood, compilers will generate CUDA kernels using the same PTX instructions that the nvcc compiler from the CUDA toolkit would generate, and compilers will generally also take advantage of the CUDA runtime API for things such as memory transfers. You can use the NVIDIA profiler tool, nvprof, at the command line to see the CUDA runtime API calls and kernel launches that are generated by OpenACC.
My code wraps C++ core code in Lua language, can I do similar thing to OpenACC code?	Currently supported languages are C, C , and Fortran. Though if Lua can import shared objects built by one of these languages, then you can include OpenACC in the shared object. We've had many users do similar things with Python and R.
Which compilers support OpenACC directives? Is it only PGI?	OpenACC is supported by PGI and GCC in addition to a few research compilers. Complete information is available on www.openacc.org/tools
Can OpenACC use the Volta and Turing tensor cores?	This is not available currently.

Question	Response
Does it support both shared memory and distributed memory parallelism?	OpenACC supports shared memory style parallelism but does not handle distributed memory parallelism -- the programmer is still required to use a model such as MPI to handle this.
What is the difference between OpenACC and OpenMP for the incremental example, parallelizing the for loop?	If you include OpenMP 4.5 target regions, OpenACC and OpenMP can both generate code to parallelize a for loop for both CPU cores and GPU cores. The difference is mainly in performance details on various architectures and compilers, and in the fact that OpenMP is generally more prescriptive, relying on the programmer to give details about how the loop iterations should be mapped to the hardware, whereas OpenACC can be more descriptive and can generate parallel code without as much specification by the programmer.
How do I know what device I am using, gpu or cpu when I write those scripts? thx	You can specify your target architecture when compiling your code. If you follow along the labs after the lecture, you will see the difference in compiling for serial CPU, multicore CPU, and NVIDIA GPU. For a quick reference though, when compiling OpenACC code there is a flag called "-ta" which stands for target accelerator. The flag for multicore is -ta=multicore, and the flag for NVIDIA GPU -ta=tesla. You can also include GPU compute capability for specify a GPU architecture. To compile for compute capability 7.0 (which is Volta architecture) would be -ta=tesla:cc70
Is it correct to assume that OpenACC is a mix of OpenMP (pragma directives) and OpenCL (heterogeneous platforms)?	OpenMP as of the 4.5 standard can also be used for heterogeneous systems with accelerators, using target offload regions. OpenACC is thus fairly similar to OpenMP in this regard. The main difference is in how parallel regions are specified by the programmer by the two models.
What is the programming model of OpenACC for multi-node, multi-GPU program?	For multi-node programs you would generally combine OpenACC for doing within-GPU parallelism with MPI for multi-GPU and multi-node parallelism.
What about using high-level languages, like Python, Go, or Julia (becoming popular with e.g. data science).	Certain high-level languages do have the ability to generate code for GPUs, for example on Python you can use Numba to generate GPU code: https://developer.nvidia.com/how-to-cuda-python
So can OpenACC be used for CPUs instead of OpenMP?	Yes, OpenACC works on CPU. The same code will run on CPUs and GPUs.
Can OpenACC parallelize across multiple compute nodes? Or, do you still have to manage this by hand with MPI	The programmer still is required to deal with parallelism across compute nodes with an approach such as MPI.
Can you combine OpenACC and CUDA for specific applications?	Yes, OpenACC and CUDA can be used together in one application.
Which are the languages that support OpenACC directives?	OpenACC supports C, C++ and Fortran
Does OpenACC run on any multi-core ARM-based SoC?	There's no limitations on OpenACC itself, but currently there's no implementation that support ARM. Though, you might want to check with GNU to see there plans.
Can an OpenACC code run on multiple GPUs, where GPU0 would be a GEFORCE GTX TITANX and GPU1 would be a TESLA K40C? I hope I can	Yes! I have done this before with one of my university codes, I ran on a GTX 1080, and a Tesla K80 at the same time. There are a few different ways to accelerate code for multiple GPU, there are some materials online (generally you would use OpenMP or MPI

Question	Response
use both GPUs at the same time through one execution of a code	with OpenACC). Then when you compile, you will add flags for all desired GPUs. For example, when I compiled my own code I included the -ta=tesla:cc35,cc60 flag. This compiled for 2 different GPU architectures, and then the GPU kernels launced depended on which GPU they were being launched on.
What compilers, if any, support CUDA on MacOS?	The CUDA toolkit is supported on MacOS and typically one would use clang as a compiler.
Any problems using OpenACC with MPI?	Nope! MPI+ OpenACC is a popular setup running on large systems (such as ORNL Summit). You will essentially have your standard MPI code, but then offload your large computational loops to 1 or more GPUs.
Does OpenACC support Windows?	The PGI compiler does support OpenACC on Windows.
Is it possible to exploit shared memory on the GPU while using OpenACC?	Yes. PGI will implicitly use shared memory for private variables on a gang loop. Also, OpenACC has a "cache" directive which will put data in shared memory. The caveat being that "cache" can be a bit tricky to use.
Wouldn't deep learning algorithms that do regression analysis be best coded on OPENACC/CUDA Fortran? You would get the highest performance, due to the speed of Fortran codes. What would be the problems of using OpenACC Fortran against just using Python...	Typically the math cores of deep learning frameworks are written in a lower-level model like CUDA C to maximize performance.
Can I use it with Python as well?	OpenACC is currently only supported in C, C++ , and Fortran. However, many users will implement their code using OpenACC in one of these languages, build the code into a shared object which can then be imported can called from a Python program.
Can CUDA and OpenACC can combined?	Yes, OpenACC and CUDA are fully interoperable.
Can one combine OpenMP and OpenACC?	Yes, OpenMP and OpenACC will work together.
Can OpenACC be used on, and has it been tested on small embedded devices such as the Raspberry pi?	The is no limitation if OpenACC could be implemented on a Rasberry Pi, however, there currently is no implementation that I'm aware of that support Pi. In general, OpenACC is used in high performance and scientific computing.
Is there an NVIDIA math library alternative to Intel MKL?	NVIDIA does provide several libraries for handling linear algebra (cuBLAS, cuSOLVER, etc.), fast Fourier transforms (cuFFT), and other math primitives. More details here: https://docs.nvidia.com/cuda/#cuda-api-references
Are OpenACC directives recognized by GNU compilers?	GNU is starting to support OpenACC. You have to install it separately, but it is doable. However, OpenACC support on GNU is new, and somewhat limited. For example, I know that at the moment it does not support compiling for multicore CPU
Could it be possible to use it in a v-san architecture with a multi-GPU and combine it like a middleware like hadoop?	Not sure what a "v-san" architecture is, but the OpenACC standard is meant to target a generic accelerator device. Not sure if there is a compiler implementation for your scenario.
Does OpenACC has support to use multi-GPU?	Yes, OpenACC works on multiple GPUs through MPI
Will CLANG support ACC anytime soon?	Sorry, I'm not sure what CLANG's current plans are regarding adding OpenACC support. However, I would encourage you to

Question	Response
	reach out to the CLANG community and ask if they can add support for OpenACC.
On a computer with multiple cores and a GPU, how will the pragmas inform the compiler whether we want a multi-threaded code or GPU code.?	When you compile you will have to switch a flag. The "-ta" flag lets you specify which parallel architecture to compile for. -ta=multicore will build for multicore CPU. -ta=tesla will build for NVIDIA GPU. You cannot run on multicore and GPU at the same time with OpenACC. You would need to include something else, like OpenMP for example.
Is it possible to use OpenACC in Nvidia Jetson TX2 Development Kit?	OpenACC PGI compiler doesn't support ARM architechure yet.
What is the best way to combine both OpenACC and OpenMPI?	OpenACC generally handles within-node parallelism, similar to how OpenMP typically handles this for on-node parallelism among CPU cores. MPI is used for communicating between nodes.
Will there be examples in this course on how to use OpenACC with MPI?	We are covering only intro to OpenACC in this course. But you can refer to the advanced part we did before here: https://developer.nvidia.com/openacc-advanced-course
Does OpenACC only support NVIDIA hardware?	OpenACC supports a variety of platforms including NVIDIA GPUs, x86 CPU, POWER CPU, AMD GPU, PEZY, Sunway, FPGA. Support will vary based on the compiler implementation.
For CFD applications, it is usually said that GPU affects the performance because of the constant movement of the data from and out of GPU, can you please give us your thoughts about this issue?	It depends on the specific application. If you can engineer your code to keep data on the GPU as long as possible and not return it to the CPU regularly, then you will amortize the cost of the memory transfer to and from the GPU. If your application requires constantly transferring the data back to the CPU, then the application performance may be suboptimal, especially if the amount of work done on the GPU is relatively small. However if the work done on the GPU takes a long time, it may still dominate the transfer cost.
Can OpenACC allow to program multicore CPU in MPI that offload some calculations separately to a single GPU?	Yes, with PGI, you can create a unified binary targeting both multicore CPU and GPUs (i.e. -ta=multicore,tesla), then at run time you can call an API routine to select which target device the rank uses.
Is there a way to get the correspondent CUDA kernel of a parallel region code generated by the compiler (for instance the PGI compiler)?	Bu default, the PGI compiler will target LLVM code for the device code generation. However, you can have the compiler instead perform a translation to CUDA and save the output via the flags "-ta=tesla:nollvm,keep". The CUDA code will outputed to a ".gpu" file. However, the CUDA is very low level so may be difficult to read.
Just saw you reply "OpenACC works on multi GPUs through MPI", is there GPU programing today allows "smp" style on multi GPUs in One node (not via MPI)?	Yes, PGI will target multiple CPUs on the same node when targeting multicore (-ta=multicore). No code changes are needed if then wanted to instead target a single GPU (i.e. just recompile using -ta=tesla)
How do I know that an OpenACC directive is slower than my serial code? Is there an option of timing the execution such as CUDA Event Timer?	There are a few ways to judge runtime. The two that I recommend is to either: 1. profile the code. The profile will include runtimes and a bunch of other details about how your code is running. There is a profiler included in the PGI compiler if you are using it. 2. Again, if using PGI compiler, there is an environment variable

Question	Response
	called PGI_ACC_TIME. If you set this variable, you code will output some extra info about runtimes after it finishes executing.
What is the difference between gefore and rtx?	GeForce is the brand name for NVIDIA GPUs such as the GTX 1080, which are of the Pascal GPU generation. RTX is the brand name for the higher-end, more recent GPUs such as the RTX 2080. The main new feature of the RTX GPUs is tensor cores for performing matrix multiplication steps (that are common in deep learning) and RT cores for doing real-time ray tracing.
Can you use directive to switch between using CPU-based bLAS library and CUBLAS (which have slightly different calling conventions), depending on the -ta (target architecture): multicore vs GPU?	This is not generally available, but it would be relatively straightforward for you as the developer to write a unified wrapper interface that can take either of these paths and then toggle between them with, say, a compile-time flag of your own.
Can you please provide a sample FORTRAN code for MULTI-GPU code with OpenACC and OpenMPI? Most sample codes one can find online are just parts of codes and not complete...	Have you checked the recording of the course on advanced OpenACC topics here: https://developer.nvidia.com/openacc-advanced-course. If you can't find your answer there, please join our slack channel at www.openacc.org/community#slack and we will help you there.
Debugging: How does one debug paralellized code if that become necessary?	If targeting multicore CPUs with PGI, you can use the PGI debugger, pgdbg. One the GPU, you can use cuda-gdb. However, since the code has been optimized, debugging GPU can be a bit difficult.
Is it possible to modify my current CPU-based Fortran simulation code into gpu-based parallel computing using OpenACC?	Yes, it should be possible, assuming your algorithm is parallelizable or can be modified to be parallel. OpenACC works very well with Fortran.
Any recommended profiling tool for OpenACC and CUDA?	The NVIDIA command-line profiler, nvprof, can be used to profile both OpenACC code and CUDA code, as well as the GUI version of this, the NVIDIA Visual Profiler (nvvp). PGI provides a similar profiling tool, pgprof.
Will NVIDIA be using AI soon?	NVIDIA is deeply involved in artificial intelligence, and NVIDIA GPUs have played a substantial role in the development of deep learning using neural networks for the last 5-10 years.
Does OpenACC provide preprocessor variable or ordinary variable that I can check to know whether the code is being compiled on CPU or on GPU (i.e. what target architecture is), to e.g. chose between FFTW and CUFFT?	The PGI compiler defines the _OPENACC macro when you are compiling for OpenACC.
Gang is a thread?	Gang is a general term that can mean a few different things. In short, it depends on your architecture. On a multicore CPU, generally gang=thread. On a GPU, generally gang=thread block. The way I like to think of it is that gang represents my outer-most level of parallelism for any architecture I am running on.
Is it worth it to parallelize with OpenACC a code that is already parallelized with OpenMP (in a multicore CPU - no GPU involved)?	If all you're ever going to run is on a multicore CPU, then sticking with your OpenMP code is fine. Though, if you ever want to move to using GPUs, then you'll want to use OpenACC or possibly OpenMP 4.5 offload directives.
Looking for a suggestion for a master's project work, where OpenACC could be used.	You are welcome to join our slack channel and ask your questions to 400 people out there. You should be able to find collaborators :) www.openacc.org/community#slack

Question	Response
How does OpenACC help in deep learning applications?	OpenACC is mostly targeted to HPC and we recommend existing deep learning libraries and frameworks for deep learning work.
In the OpenACC computing can we access threads using thread ids just like in CUDA?	OpenACC does not expose this like CUDA does.
Will use OpenACC for FPGA boards?	There is an implementation by ORNL of OpenACC for FPGAs - OpenARC
Doesn't using multiple parallel regions add kernel initialization latency?	Yes, additional kernel launches (corresponding to multiple parallel regions) will lead to additional kernel launch latency. However note that this may be true even for a single parallel region, if the compiler generates multiple kernel launches corresponding to multiple loops.
Are there scientific code uses for the new RT cores on Turing? If so, will OpenACC eventually somehow target RT or tensor cores?	Certain applications such as particle-based codes could theoretically benefit from RT cores. However it is not clear how or when OpenACC would take advantage of this.
When I try install the driver for GTX Titan X, my OpenACC code fails to understand that PC setup has a 2 GPUs, the other being the Tesla K40. Is this a driver problem? I was compiling with -ta:tesla=cc35	You can use acc_set_device_num() to select which GPU you would like to run on.
Nvidia TX2 and Xavier are ARM %2B GPU systems. When you said OpenACC runs on them, does it mean we can use OpenACC for both the ARM and the GPU sections?	As a standard, there's nothing inhibiting an implementation of OpenACC. Currently, I do not know of a released compiler implementation that supports ARM. Though, if/when there is an implementation, then yes, it could be used for both ARM and GPU sections.
Can you say what gonna happen if we do not use reductions?	Reductions are necessary if the threads all need to coordinate to create an answer, such as the minimum value over an array. If a reduction is not performed, the answer will be incorrect and meaningless.
Parallel loop, as in OpenMP, works only with for loop?	OpenACC parallel loops are intended to expose parallelism in "spatial" loops such as for loops. It would not make sense for "temporal" loops such as while loops.
Is there any Minfo alike for gcc?	There is not a direct analogue that I am aware of in gcc.
Does the build of the compiler depend on hardware? for example for windows can get portable build?	I'm you're asking if you can build your application on one Windows system and then move it to another Windows system using different hardware. Yes, this is done all the time. Though, you do need to compile your code to target multiple CPUs (or a generic CPU) and multiple GPUs. For example, the flags "-tp=penryn,haswell,skylake" will target multiple CPUs covering most current x86 based systems. By default, "-ta=tesla" will target multiple NVIDIA GPUs, the exact GPU will vary by compiler version and CUDA version. You can give an explicit list of target GPUs by specifying them, for example "-ta=tesla:cc35,cc50,cc60,cc70". Other issues are runtime libraries and CUDA driver version.
Is a function called allowed in an OpenACC parallel loop code segment?	Yes! If you are using a multicore CPU, you shouldn't have a problem. If you are using a GPU, I would give a search for "OpenACC Routine Directive" to learn how to run subfunctions on GPUs in OpenACC.

Question	Response
Is it possible to use the GPU constant memory with OpenACC?	OpenACC does not expose the ability to use constant memory directly (though the compiler may choose to take advantage of it under the hood).
Can I use OpenACC in NVIDIA GTX instead a TESLA?	Yes, OpenACC can generate code for GeForce GPUs in addition to Tesla GPUs.
Is pgc++ supported on Windows?	PGI currently supports C and Fortran on Windows. No C++ support.
How about if we use a consumer level GPUs? for example Nvidia GTX 10xx, how is the floating point performance in double precision?	Double precision performance on GeForce GPUs is typically substantially lower than on Tesla GPUs. For example, there is typically 1 double precision core for every 2 single precision cores on a high-end Tesla GPU such as V100, while on GeForce cards the ratio is typically 1:16 or 1:32, something in that ballpark.
How long is the lab available?	The labs will be available after the course for a few months. We suggest though to do the lab now, so we can help you along the way.
What about amd strix?	You may want to check with GNU. I believe that they are adding OpenACC support for AMD devices but I don't know specifically if this include Strix.
Do you think is it wise to build a C++ application from scratch on OpenACC or is it just a good fit for parallelize existing applications?	OpenACC is pretty good both ways. The major benefit you get from using OpenACC on any code is that you will have a single source-code that can be run on a variety of architectures. Also, this is ultimately my opinion, but if I were planning to write a CUDA code, and I wasn't concerned with completely maximizing my performance I would definitely consider OpenACC as a good option.
How about multiple cores using a single GPU? do they have to take turns or can the GPU used simultaneously?	Yes, you can use OpenMP host threads to generate parallel regions independently. You should launch them in separate streams if you want them to execute simultaneously.
How about the license of OpenACC? Does it mean we can use as commercial version?	PGI Community Edition is available for free to everyone. More details here: https://www.pgroup.com/products/community.htm
How to work with Visual Studio?	PGI is supported for Windows with Visual Studio. More information on Windows support here: https://www.pgroup.com/support/win-ce.htm
Do we need to be concerned about synchronization barriers while using OpenACC?	No. By default, barriers are placed after each compute region. Later classes will introduce the "async" clause where you can have the GPU compute region be non-blocking so the CPU will continue while the GPU is executing the compute region.
Does NVLINK help in copying data between GPUs?	NVLink substantially improves the bandwidth when copying data between GPUs, relative to PCIe.
My question related with productization then how much do I have pay in order to sell my product shipped with OpenACC?	PGI does not charge for run time, so there's no additional cost. Though, you may consider purchasing the PGI Professional Edition, as opposed to the no-cost Community Edition, so you have support from PGI. Again though, there is no fee for you to redistribute your program.
Can OpenACC check if the loop is parallel or not and reject parallelization?	The compiler will check for you whether a loop can be validly parallelized and will refuse to do so if it cannot be. You can use the -Minfo=all messages to get information on which loops were parallelized.

Question	Response
Are there OpenACC directory to copy data between GPUs via NVLINK, or that is dumb thing and does not make sense?	The NVLink connection between GPUs will be automatically used when doing data transfers between GPUs, if available; the programmer does not need to request this.
How does NVIDIA work for updating drivers with Microsoft and the NVIDIA website?	NVIDIA drivers can be downloaded here for all supported operating systems: https://www.nvidia.com/download/index.aspx