# OpenACC Course
Office Hour #1 Q&A

**Q1:** What's the basic difference between CUDA and OpenACC? OpenACC and OpenMP?

**A:** CUDA is a very low level language, it is very similar to C and C++. It is a low level and you can express exactly what your computation is doing and how it is doing it. You express the parallelism explicitly, there is no compiler deducing what parallelism to expose. OpenACC on the other hand is higher level than CUDA, it is a pragma based parallelization scheme, where you are going to give hints to the compiler on how to parallelize your code, but not necessarily give the exact steps on how to do it. The compiler then does the analysis and decides on the best way to parallelize the code. Both OpenACC and CUDA have the same goal, which is parallelize your code to run on CUDA-enabled hardware. OpenACC also works with non-CUDA hardware as well. In general we expect well-optimized CUDA code run faster than well-optimized OpenACC code, but in many cases they should be similar.

OpenMP is an alternative to OpenACC, which focusing on parallelizing on CPU with recent extension to accelerators. Those scheme are very similar with some minor differences. There is a lot of work going in right now with both OpenACC and OpenMP committees to standardize both and agree on the set of features.

As of now OpenMP is better with CPU code, where OpenACC has a few more years of maturity targeting GPU codes and therefore it is an accelerator focused software.

**Q2:** What OS is supported by OpenACC Toolkit? PGI Compiler?
**A:** OpenACC Toolkit supports Linux. PGI Compiler is supported on Windows, Linux, and Mac, but only on x86 platforms. Both do no run on ARM Linux.

**Q3:** Is PGI the only OpenACC compiler available?
**A:** No. Other compilers include Cray and in the development branch GNU has an OpenACC compiler also.

**Q4:** Can libraries be used with OpenACC?
**A:** In general what you should always do is to look at the libraries you can use. For example if you use BLAS routines use cuBLAS, FFT - check cuFFT. Those libraries do exist and there was a lot of work put into them, so it is a good start. So please use libraries with OpenACC, they interoperate just fine.

**Q5:** Does OpenACC support multiple GPUs?
**A:** Yes. OpenACC does support multiple GPUs and there are API calls that can be used to identify which GPU to use. Will be covering it in the later lectures.

**Q6:** Does OpenACC use shared memory in a proper way? Distributed memory?
**A:** Yes, it uses shared memory when it can do so. For example when you do a reduction, it will often do a reduction in shared memory. For multiple threads, OpenACC most likely won't use shared memory, since it is focused on a single thread parallelism.

**Q7:** What is the difference: Matlab with parallel tool box and OpenACC?
**A:** The difference is that Matlab with parallel tool box is just another way to achieve parallelism, but it is built into Matlab and it works side-by-side with OpenACC.

**Q8:** How many dimensions does OpenACC support in calculating vectors?
**A:** You cannot put vectors in multiple loops, only on a single loop. If you have multiple loops that you want to paralyze over, use a collapse clause, which will collapse two loops in one and will allow to put vectors over multiple loops.

**Q9:** Can two applications run in parallel on GPU that is optimized with OpenACC?
**A:** 2 applications can share a single GPU, however it is not recommended. Yes, but it's not recommended because of possible bugs (older hardware) and slower simulations (newer hardware). Instead use the Multiple Process Service (MPS).

**Q10:** How many kernels can I generate? Is there a limitation?
**A:** There is no practical limit on the number of kernels you can generate.

**Q11:** Is there any criteria to choose the best combination of gangs, (workers) and vectors?
**A:** Yes. In general, your vector size should be divisible by 32, the product of workers and vectors should be at least 128 in size, and gangs should be as large as possible just to expose as much parallelism as possible.

**Q12:** Are there Fortran free compilers able to handle with OpenACC?
**A:** The PGI compiler has a free academic developer license. The GNU compiler is also free.

**Q13:** OpenACC supports STL::vector of structures?
**A:** With STL vector, you can get a raw pointer and use that in OpenACC.

**Q14:** Does OpenACC use shared or distributed memory?
**A:** Distributed memory will be an MPI process and shared will be more OpenMP, where everybody is using the same memory system. OpenACC uses the shared memory system - all the threads can access all the memory. You can also make some memory private to a certain thread, please look into documentation for more details.

**Q15:** Does OpenACC support the Xeon Phi yet? In particular, the PGI compiler.
**A:** Not currently, but it is on the roadmap.

**Q16:** If we have a code that use CUDA and OpenACC, which compiler should we use? Can we use PGI?
**A:** For CUDA, you would use NVIDIA, but for OpenACC, you can use whichever compiler that best suits your application or environment (i.e PGI, GNU, Cray, etc). You can interoperate CUDA and OpenACC in the same application without a problem.

**Q17:** When managing data movement, does the data region need to go outside the kernels regions? If so, then why?
**A:** Yes, the data movement sits outside of a kernel regions. A kernel's region just saying that this is an area code where we want kernel to run, the data should be already present. The data needs to go outside the kernels region because the data needs to be present or else it will be implicitly copied down or up in every single kernel call that happens. You cannot put data inside the kernel's regions - it needs to be outside.

**Q18:** Does OpenACC toolkit contains the latest NVIDIA GPU toolkit?
**A:** Yes. PGI supports the latest CUDA version.

**Q19:** Does OpenACC support 4 dimensional array, such as a[i][j][k][l]?
**A:** Yes, it supports any size of arrays you have. In Fortran you don't even need to shape the arrays, because Fortan carries this knowledge together with type. In C you may need to shape it.

**Q20:** How does the overhead of parallelism compare with OpenMP?
**A:** Typically from the point the kernel gets launch to time when is actually launches it takes around 3-8 microseconds. When you decide where you want to put your kernel, put it on larger loops. If a loop is 100 elements wide, it isn't worth putting it on a GPU. But if your loop is quite big it is definitely worthwhile to put on a GPU.

**Q21:** Can you please explain gangs and vectors?
**A:** The finest level of the parallelism you have will be a vector. A vector is any size closest to the device and is similar to a simD type of parallelism. A group of vectors is called a worker and group workers is called a gang. This gives us 3 levels of parallelism.

**Q22:** Why is the inner loop normally what you want to vectorize?
**A:** For example you are adding 2 vectors together, you will often want to parallelize that with a vector, because you want it to be the faster changing dimension in your block and it will give you a better access pattern in memory access.

**Q23:** If I have 12 cores working on a task simultaneously, and each task at some point needs to say do an FFT on the GPU, do I need to somehow synchronize that traffic? i.e. how do I assure that only one CPU thread accesses the GPU at a time?
**A:** You can have multiple CPU threads accessing the same GPU without any problems. If it's 12 independent FFTs, it does not need to be synchronized. Each of those will launch their own work in an FFT. If those 12 threads produced one field you want to do an FFT on, then you will need synchronization on a host, and launch FFT from a single core.

**Q24:** What is worth to spend time? Start OpenACC from scratch or get a bit of intel from CUDA programing and after that coming back to the OpenACC?
**A:** It depends what you are doing and comfortable with. In my opinion, I would start with OpenACC first and put as much work on the GPU. Then, go back to profile the code and find any "hotspots" (i.e. a kernel taking too much run time) and if those cannot be resolved with OpenACC, run them on CUDA, so you'll have the best of the both worlds - rapid development from OpenACC and high performance from CUDA.

**Q25:** But apparently learning OpenACC is a lot easier, isn't it?
**A:** Yes. Mainly because it does data looping for you and compiler does most of the work for you.

**Q26:** Is it possible to organize concurrent memory copy to the host and kernel run?
**A:** Yes, using the A-sync clause that says that this code runs asynchronous with the host.

**Q27:** Does Intel compiler support CUDA?
**A:** No. You'd have to use NVCC. If referring to CUDA hardware, the answer is still no. Intel does not currently support it.

**Q28:** Is it possible to use vector clause in nested loop construct with collapse and reduction clauses?
**A:** Yes, you can use a vector clause in a nested (inner) loop with a collapse clause in the inner loop, but not in the outer loops because the vector clause would need to be in the outer loop as well. If the reduction clause is in the outer loop, it is possible, but unsure if it's in the inner loop. Need a code example for a complete answer.

**Q29:** How important is manual gang/worker/vector splitting to performance?
**A:** It isn't required. It is considered a tuning part of your workflow. Let compiler do the work. In your individual case you might find a better combination, but you have a potential to get both good and bad performance, so make sure you check your work.

**Q30:** Does the zero copy in CUDA supported by OpenACC?
**A:** Not for sure, but it might not likely support zero copy.

**Q31:** You mentioned that certain libraries are optimized for use with OpenACC. Is FFTW available?
**A:** Correction, libraries aren't necessarily optimized for use with OpenACC, but for NVIDIA hardware and can be used with OpenACC. And FFTW is available it is called cuFFT.

**Q32:** I have AMD R9 gpu, without cuda support how effective is OpenACC?
**A:** OpenACC is a cross platform language and AMD hardware works just fine on the PGI compiler.

**Q33:** Does OpenACC support stream, like CUDA does?
**A:** Yes. This is an A-sync close that essentially maps to what we call streams.

**Q34:** If OpenACC is only available for Linux, is not compatible with DirectX and only for OpenGL?
**A:** OpenACC is independent of DirectX and OpenGL. They're unrelated to each other since OpenACC is a compute language and DirectX and OpenGL are graphic languages.

**Q35:** Does OpenACC automatically make use of multiple GPUs if they are installed?
**A:** No, it does not automatically use multiple GPUs, but instead there are API calls and constructs in the language to specify how to use multiple GPUs.

**Q36:** What accelerators are supported currently by OpenACC that are not NVidia products?
**A:** PGI compile down to AMD, x86, NVIDIA GPU, Xeon Phi is down the road. Potentially FPGAs can be supported as well.

**Q37:** What is more efficient? CUDA or OpenACC?
**A:** It depends. In general a well programmed CUDA code should be at least as efficient as an OpenACC code. And the reason is both do the exact same thing, but in CUDA you have a little bit more control and can do a little bit more things than in OpenACC. However, the effort to do similar things with CUDA is more than with OpenACC. In some cases for complex kernels CUDA can start outperforming OpenACC.

**Q38:** Does compiler optimization levels (i.e -O3) affect OpenACC performance?
**A:** It will because that flag goes down into a compilation of the device code and that optimization level will be used when compiling it. For example I can have kernel with O0 and O3 and O3 will be quicker because it will use more optimization techniques.

**Q39:** If I have multiple threads executing calculations on the GPU simultaneously, might I exhaust the memory on the GPU by copying in too much data at once? So for e.g. if the GPU has 1 Gig of memory, and each CPU core will at some point send .5 Gig of data to the GPU to be processed, do I need to manage that memory flow myself?
**A:** Yes, if your CPU application has more memory than the GPU has, you have to be careful not to allocate too much memory on a GPU at one time. In this case asked - it will cause an out of memory error.

**Q40:** What if I want to parallelize operations on a dataset whose memory is much larger than the GPU can support? Can OpenACC handle this?
**A:** You would have to stream data in an out of the device and process it there.

**Q41:** Is OpenACC aware of the specifics of your GPU? Or is that what the Gang/Worker/Vector control is for?
**A:** Yes, OpenACC is aware. It is up to the runtime to define how to handle it: through compiler or manually.

**Q42:** How hard is achieve loop fusion of four or more nested for nested loops using OpenACC?
**A:** if fusing multiple levels of the loop together by using of collapse fuse. Other case a loop that exists on another loop etc, then create a separate loop and put all the code together there and you can do it with multiple nests as you see fit.

**Q43:** What about OpenCL compatibility?
**A:** OpenCL supports Xeon, Xeon Phi and FPGA. We expect a similar level of adoption for OpenACC as well.

**Q44:** Are there free compilers for Fortran that can I use?
**A:** Yes. PGI and GNU compilers are free for academia, but if you want to deploy using PGI, you'll have to purchase a license with PGI.

**Q45:** I noticed one of the examples had a #pragma omp parallel for block *inside* a #pagma acc kernels block. Does OpenMP run on the GPU inside the kernels?
**A:** No, it does not. When you use OpenMP it is just a hint, the compiler is free to choose the parallelization scheme it wants. It possible to run OpenACC and OpenMP at the same time just not at the same loop body.

**Q46:** Is it possible to provide a link for more information on compiling CUDA Fortran using the GNU compiler?
**A:** CUDA Fortran is only available with the PGI Fortran compiler. As far as we know, there is no effort underway to support CUDA extension in gfortran. University students and faculty are eligible for a free University Developer license to PGI compilers including CUDA Fortran <http://www.nvidia.com/openacc>.

**Q47:** Can I use intel compiler for OpenACC?
**A:** Currently Intel compiler doesn't support OpenACC.

**Q48:** Assuming the compiler can parallelize a kernels region loop, is there a difference between using kernels versus an explicit parallel loop directive? Is there a reason to use one over the other?

**A:** The kernels construct may be thought of as a hint to the compiler of where it should look for parallelism while the parallel directive is an assertion to the compiler of where there is parallelism. Parallel directive relies on the programmer to have correctly identified parallelism in the code and remove anything in the code that may be unsafe to parallelize.

**Q49:** How does OpenACC work with C-cross compatible languages like Rust?
**A:** We don't have experience with it, but cannot see why it won't work.

**Q50:** I have a 540m. Does it matter?
**A:** GT 540M's compute capability (cc) is cc21, which is supported in OTK.

**Q51:** Does OpenACC support 980M NVIDIA GPUs on laptops?
**A:** 980M's is cc52. Yes.

**Q52:** Are we able to compile a single binary that runs on different gpus from nvidia and amd?
**A:** Yes. The PGI 2015 compilers can produce PGI Unified Binary object or executable files containing code streams fully optimized and supported for both AMD and Intel x64 CPUs and NVIDIA GPUs. Please check -tp option using PGI compiler.

**Q53:** Do I need to install some extra dll to run program using OpenACC ?
**A:** OTK currently doesn't support Windows. But check PGI workstation for Windows, which supports OpenACC.

**Q54:** I don't have a GPU card yet, but is there a place where I can test my code?
**A:** We have an academic seeding program. Through this program you can submit a request to get a GPU for your research.

**Q55:** Will GT 240 work with OpenACC?
**A:** PGI compilers in the OpenACC Toolkit support most NVIDIA CUDA-enabled GPU with Compute Capability 2.0 (cc20) or newer. See developer.nvidia.com/cuda-gpus for a comprehensive list of NVIDIA CUDA-enabled GPUs.