OpenACC
More Science, Less Programming

OpenACC Course October 2018
Lecture 3 Q&As

Question	Response
When I read OpenACC 2.6 document, I have noticed the following statements regarding copyin(var) clause: " If var is not present on the current device, a runtime error is issued". 1. Could you please explain how I can check the variable is on the current device or not. 2. How could this error happen, in what situation the variable not in the current device? 3. How can I have used the data clause correctly and the variable does not appear in the GPU, Could you please give me an example of this situation?	For the case of a "copyin" clause, it's an unlikely situation. The phrase you're quoting pertains to exiting a data region where var is no longer present. This can only really occur if you are mixing a structured data region with a copyin, and then adding an unstructured "exit data delete(var)". Hence, when the structured data region exits, var is no longer present. There might be other situations that this could occur, but it would be rare.
Is there an OpenACC equivalent to OpenMP's "threadprivate" declaration? the use case is a small array used as a temporary storage for each thread independently. Would one simply add the array to the "private()" clause?	"threadprivate" has the notion of a persistent thread which doesn't exist under OpenACC since threads are created/destroyed upon entering/exiting a compute region. For small private arrays, you would use the "private" or "firstprivate" clauses. The difference is "firstprivate" the private arrays are initialized to the value of the host array. "private" is uninitialized.
How does OpenACC tackle deep copy?	With OpenACC 2.6 you transfer the root type, then copy over the other components and connect them up. in the future standards you will be able to tell how much of the data structure you want to copy to the Device. Otherwise, you need to break up your data structures and copy over the pieces to the Device. Then you have to use the pieces individually.
So in CUDA terms, a "worker" is the "y"-block dimension and the vector is the "x" block dimension?	The standard is agnostic as to how to implement this, but as Jeff says, this is a valid way to reason about it and may in fact be how it is implemented, depending on the compiler.
Do you have a link to a simple example showing the use of the	I have an example of using the API call "acc_attach" which could be modified to use the directive version. See Chapter 5

Question	Response
"attach" clause for manual deep copy? The documentation is a little vague and I cannot find an actual example with a simple Fortran derived type	from https://github.com/rmfarber/ParallelProgrammingWithOpenACC
Is CUDA grid block thread a one to one mapping to gang worker vector? (I may have order mixed up)	For PGI's implementation, an OpenACC gang maps to a CUDA block, a vector to the threadid+x dimension of a block, and worker to threadid+y. Other implementation may be different, but it's the most likely mapping.
Comparing this with CUDA, a gang would be a block, workers would be threads and vector would be a warp. Am i correct?	Actually, a gang is a threadblock (block), a worker is effectively a warp, and an OpenACC vector is a CUDA thread.
Can we do a vector loop reduction? That is each worker does a separate reduction?	Yes, it's valid to have a reduction clause on a vector loop.
How could we download the three labs for future use to have more time to learn?	We plan to publish a container with all the labs towards the end of the month. We will announce on Slack availability.
Is the "warp" the same length for float and double?	I believe the warp size is always 32.
Is a GPU sensitive to a crips or bloods gang :)	Hopefully not. Unless they're branching out into using HPC...
In the context of the painter model, I am currently on this event and my GTX 745 is on one display with 12 programs can you relate this to the Painter model.	The display and 12 programs would be more like painting 12 houses. The painter model Jeff is refer is to is distributing the painting work across many workers for each individual house (i.e. each program).
Is there a way to have nvprof/pgprof not show data transfers smaller than a specified size? My program does a LOT of 1-8-byte transfers due to derived type pointers and scalars and it is hard to figure out if there is a performance-limiting data transfer within all the multitude of brown boxes	That is not available as an option currently, but I will file an RFE with NVIDIA, indicating this as the use case in mind, thank you!

Question	Response
In unified memory, can there be two copies of constant data or is there one and only one copy? That is, can I have a constant table of data on both the host and device at the same time w/o having to explicitly copy it? Will UVM repeatedly move these data back and forth or allow two copies?	Chris - it's Jeff Layton. I'm not sure - I'll check with the PGI folks and get back to you over private email.
If the warp size is 32, how can it be that I sometimes get better performance with a warp size of 16? Could it have to do with the resulting block sizes?	In CUDA the warp size is always 32, and so indeed if you launched work in chunks of 16 this is suboptimal, but in OpenACC it is not necessarily the case that e.g. vector(32) maps directly to a CUDA concept like a warp of size 32. As Jeff said earlier, if you think about this in terms of x and y dimensions of a thread block, then the size of a given chunk of work can at least be size 32 assuming the worker length is > 1.
What recommendations are there for profiling MPI+OpenACC and MPI+OpenMP+OpenACC code?	While you can't have a single profile across all ranks, you can profile each individual rank. Something like "mpirun -np 32 pgprof -o myprofile.+p.prof a.out". The "-o" says to save the profile to a file, where "+p" will be replaced with the process id of the rank. You will then have 32 profiles. These can then be inspected individually in the GUI or command line, but can also be merged in the GUI. However, in general, the GPU performance information you're using the profiler for is single rank. For MPI performance, you'll need to use something like TotalView.
Do you have any documentation about Deep Learning implementation using OpenACC directives?	The classic DL frameworks do not use OpenACC. They use CUDA and NVIDIA libraries such as cuDNN, cuBLAS, etc or DL frameworks. It's possible to used OpenACC for DL for matrix multiplication, etc. but we don't have any specific documentation on how to do that. I would recommend to write your DL implementation for a serial implementation and then port it using OpenACC.
Why do you have the copyin/copyout clauses on the inner loops along with the outer copy clause?	That is done in this case to ensure the avoidance of race conditions since the tiles can operate independently of each other.

Question	Response
Does the tile clause imply cache clause? Since each tile will be put into shared memory, is that not the same as the cache clause? Do they need to be combined?	No, tiling is about how to distribute the work across multiple thread dimensions. Cache is about putting a subset of data in shared memory. Yes, they can be combined.
Will the next version of PGI support using tile and cache clauses for derived type arrays?	Tile has to do about distribution of work of the loops so wouldn't apply. I'm not expecting any additional support in the cache clause the upcoming PGI release. Please send us a request for enhancement and we'll take a look at what can be done.
With tiling what if you have a prime number of loop iterations?	That is probably suboptimal in performance compared to an even number, or better yet a power of 2, but you will get correct results.
What's the parallel idea of tiles in cuda?	Tiling is a general parallelism concept, not an OpenACC specific concept. You could tile a loop in CUDA in many ways. The most canonical example is dividing an array into N chunks and then launching each of the N chunks in a separate CUDA stream.
Is there a relationship between the warp size and the tile sizes in dimensions?	The warp size is always 32. The product of the integers in a tile() clause cannot be greater than 1024. It's recommended to use a tile combination that is divisible by 32. So things like (32,7) is OK, but tile(7,7) is not a good idea (it will work, but the performance won't be good).
Is the size of tile dynamic, or should it be constant?	The tile size must either be a compile-time constant, or left as an * in which case the compiler will figure out the optimal value.
If one has a nested loop that does internal points of a grid (such as do i=2:n-1) etc is the collapse clause efficient or does the break in stride-1 of the array break any advantage of the collapse?	I'm thinking that in most cases, it probably doesn't matter. The striding would be the same if the loops were collapsed or if you used a loop directive on each loop. Of course the optimal loop schedule is dependent on the specific of the loops and the data access pattern, so there may be cases where it could be a problem.
With modern GPUs that have a nice cache, does the cache clause (i.e. manual shared memory) yield any performance gain? Are there any example data of the cache clause improving performance on Pascal/Volta GPUs?	In general NVIDIA finds that there is some benefit from using shared memory compared to relying on L1 cache, but on modern GPUs like Volta, the performance gap between hand-tuned code using shared memory and non-tuned OpenACC code is of order 10% for the SPEC accel benchmarks, so that is probably an upper bound on the difference.

Question	Response
Is there a way to have the CPU and the GPU working together with OpenACC? For example, have the CPU working on a for loop and the GPU working on the other for loop?	Yes. By putting an "async" clause on OpenACC data and compute regions will make them non-blocking. The CPU will continue with execution until it reaches an OpenACC "wait" directive. During this time the CPU can be executing another loop.
Does OpenACC (or the PGI implementation) allocate to a good (i.e., performant) alignment automatically?	Yes, data allocation is aligned.
Any way to change the default vector length at compile-time with PGI?	No, this is not possible, and in any case you may not want to do such a thing because it would prevent the compiler from analyzing loops to figure out what is the best choice.
What if the number of gangs*#workers*#vector is smaller than the loop iterations? will the compiler use sets of #gangs to fill out the iterations?	Yes, by default it adds a loop so that each gang/worker/vector may compute more than one iteration.
What is the usage of OpenACC API functions, and what are their benefits?	Are you looking for specific functions of OpenACC?
Is it possible to target multiple GPUs or a CPU system with multiple sockets using OpenACC?	There are options. We recommend to wrap your OpenACC code with OpenMP or MPI, or both. So you write your code with OpenACC to run on one GPU. Then this code can be used as a thread with OpenMP or a rank with MPI. There is a way to create "queues" with OpenACC so you can assign certain work to a queue and they work between queues is independent.
Why OpenACC was created?	OpenACC was designed to help start with parallel programming faster. It is a directives-based programming model that allows to minimize programming efforts and code modifications.
The tiles operate entirely independent of each other, right? That would mean I couldn't use them for stencils, because they more one element by one element (such as to create image blur)	Loops with the loop directive are independent. The tile clause is used to schedule these independent loops across multi-dimensional thread blocks. Though you should be able to use tile with a stencil.

Question	Response
Given the hardware configuration of the host is unknown until run-time, would it be possible to use the OpenACC API library calls to dynamically determine what optimizations should be done?	The optimization is performed at compile time, so no would not be able to change at runtime. However with the PGI compiler you can create a unified binary which targets multiple architectures. On the host side, this would be done by giving a list of architectures to the "-tp" flag (target processor). For example "-tp=penryn,haswell,skylake". For a target device, you'd use the "-ta" flag (target accelerator). for example "-ta=multicore,tesla:cc35,cc50,cc60,cc70".
Regarding two books on OpenACC, which is best to start with? Many thanks.	Is it better to start with "OpenACC for Programmers: Concepts and Strategies"
What is the link to NGC?	https://www.nvidia.com/en-us/gpu-cloud/