OpenACC Course
Class #1 Q&A

Contents

OpenACC/CUDA/OpenMP

Q: Is OpenACC an NVIDIA standard or is it accepted as an industry standard?
A: OpenACC is an industry standard. NVIDIA is a member same as Cray, Pathscale, Oak Ridge National Llab, University of Houston and others. It is developed by a group of companies all working together.

Q: What's the basic difference between CUDA and OpenACC?
A: CUDA is a very low level explicit approach when you need control on optimization. OpenACC is a high level approach - just add directive to the code, simpler to use.

Q: Will OpenACC replace CUDA totally in near future?
A: No. There are complementary approaches. OpenACC is a high level and CUDA gives more control with acceleration. They are fully interoperable. You can start with OpenACC and complete with CUDA for more complex parts of the code.

Q: What is the difference between OpenACC and OpenMP?
A: OpenMP is a more established specification that was initially designed for shared memory parallelism, but has recently added support for offloading. OpenMP explicitly states that no dependency analysis or automatic parallelization is required by the compiler in order to implement the specification, placing the full burden of ensuring correctness upon the programmer. OpenACC is a more recent specification that is focused on performance portability and scalable parallelism. As a result, it does not include many of the explicit synchronization available in OpenMP and requires much more analysis by the compiler to ensure correctness. These are two, complementary approaches to parallel programming. Many of the same people are working on both specifications and working hard to converge upon a common set of features, although the question of programmer-driven vs. compiler-driven still remains. OpenMP has much more mature compilers in terms of shared memory processors, such as multicore processors, while OpenACC has more mature compilers for GPUs and GPU-like architectures.

Q: Can we mix OpenACC with other paradigms, MPI or OpenMP?

A: Absolutely. OpenACC can be mixed with OpenMP, although both cannot be used on the same loop simultaneously, MPI, CUDA, and other paradigms. Later in the course we will provide examples of some of these.

OpenACC Implementation

Q: How does the PGI C++ compiler compare with others like GCC and Cray?
A: PGI and Cray both have mature implementations of OpenACC and which will perform better for a given code will vary. The GCC implementation of OpenACC is still very new and incomplete, but developing rapidly.

Q: Can I install/use OpenACC on Windows?
A: Currently the OpenACC Toolkit that includes PGI compiler supports Linux only. However the PGI Compiler itself is available for Mac and Windows. You could always find more information on www.pgroup.com.

Q: Which compiler is needed to use OpenACC with e.g. C codes?
A: OpenACC is supported by the PGI, Cray, PathScale, and to a limited extend GCC compilers, plus several research compilers.

Q: Is this Intel only or arm as well?
A: At the moment the PGI OpenACC compiler only supports x86 hosts, but more architectures will be added in the future. We cannot discuss any specific dates or targets at this time.

Q: can I use open AAC in Visual Studio Community 2015
A: Currently Visual Studio does not support OpenACC, but the PGI compiler is available on Windows.

Q: How many compilers support OpenACC?
A: PGI, Cray, Pathscale and etc.

Q: Can the OpenACC toolkit be downloaded and used on a Mac?
A: OpenACC Toolkit is available on Linux right now, but the PGI compiler which is a part of the toolkit can be downloaded to be used on Mac

Q: Any issues if using the Jetson board for this class?
A: OpenACC Toolkit is not available for ARM as of now.

Q: Does the PGI compiler only support linux?
A: PGI compiler supports across the platforms: Linux, Windows and OSX.

Licensing

Q: Is the academic license is for one year only for the PGI compiler?
A: It is renewable every year. As long as you stay in academia, the license remains free

Q: I have a GeForce GTX 480, is this GPU enabled to work with OpenACC?
A: Yes, you are welcome to use it.

Q: To prove the academic position, what do we need, institutional email?
A: The Free OpenACC University Developer license is open to students and faculty members of degree-granting universities worldwide. Proof of eligibility is required. Accepted forms of proof include:

- Registration using a qualified university email address
- Scanned copy of your university ID or similar documentation (e.g tuition enrollment, etc.). Send via email to sales@pgroup.com from the address you used to register.
- A link to your entry on the staff web page for your university. Please enter the link in the 'University URL' field on the following page

Course Materials

Q: Hi! Are there any textbooks about this course?
A: There are no textbooks, but we'll be providing several code samples over the course and will be giving you a link to the OpenACC Programming Guide, which is freely available online.

Q: Does some basic knowledge about parallel programing needed?
A: Some knowledge of parallel programming will be helpful, but not required. This course is intended for all audiences.

Q: Hi. We can use these slides to teach things about OpenACC in my classes?
A: Yes, you are welcome to use these slides. We are also working with educators on a separate OpenACC program. Please contact us for more information.

Q: There is some manual that i can see about Optimizations done by OpenACC compiler?
A: Please see the OpenACC Programming Guide available in the OpenACC Toolkit and at OpenACC.org.

Q: Do we need to install Nvidia sdk for this course ?
A: We recommend installing the toolkit, but you can also access it through our hands-on labs remotely. You can download and install our latest OpenACC Toolkit (OTK) to try the samples in this course. The OTK requires CUDA 7.5 driver installed.

Q: Are we going to go through the installation?
A: You could find OpenACC_Toolkit_Quick_Start_Guide.pdf in your OTK installation package. Note that you need to install CUDA 7.5 driver separately.

Languages and Libraries

Q: Where parallel algorithms libraries like Thrust or Bolt fit in?
A: Thrust is another approach to accelerated computing. Thrust is a C++ template library for parallel programming. It is a complementary approach. You can use THrust and OpenACC together.

Q: can I use OpenACC inside PyCuda kernels?
A: Not directly. OpenACC is only supported in C, C++, and Fortran. One could use OpenACC within a C, C++, or Fortran function that is called from PyCUDA, but the directives cannot be added directly into Python codes.

Multi-GPU support

Q: How many devices of a node can OpenACC utilize?
A: OpenACC's multi-device support is similar to CUDA. OpenACC kernels and functions can be directed to a particular device by using the acc_set_device_num API routine and supports as many devices as are available on a node.

Q: OpenACC recognize if I have more than one GPU in my computer?
A: OpenACC will allow you to explicitly send data and work to multiple GPUs, but it will not automatically distribute work across multiple GPUs.

Q: Can OpenACC be used with MPI?
A: Yes. This will be demonstrated in lecture 4.

How OpenACC Works

Q: Can the compiler reject a directive if it thinks the loop is not parallelizable?
A: Yes. PGI compiler will analyze the loops and decide whether they're parallelizable or not. There is also an option to tell compiler explicitly to parallelize the loop. We will cover it later in the course.

Q: Why is the iter++ inside the kernel?
A: This code could be placed inside or outside of the kernels region. Since the compiler can easily see that this is not parallel work, it will execute it on the host CPU.

Q: How much does the CPU contribute to the OpenACC speedup?
A: In all of the graphs (for the class #1) shown the green "OpenACC" speed-up is running the calculation completely on the GPU. Only the checking for convergence is running on the CPU.

Q: The offloaded code, does it run asynchronously to the portion that stays on the CPU?
A: The CPU will synchronize at the closing curley-brace of the kernel directives, but OpenACC also provides the `async` clause for allowing the CPU and GPU to work asynchronously from each other.

Q: How was the CPU code parallelized - OMP, MPI, Pthreads? For the graph in blue
A: OpenMP threads. The CPU has 16 physical cores, but the speed-up trails off above 8 cores as the application begins saturating the memory bandwidth.

Q: If use Intel/AMD iGPU for acceleration, Will the program/compiler need to copy the data between CPU and iGPU?
A: It is generally best to assume discrete CPU and GPU memories and express the data movement, although the compiler may choose to ignore these directives on platforms where they are not necessary.

Q: Is there a way to generate CUDA code from OpenACC source? It would be nice to check if the solution given by the compiler is correct/better than previous code I had on CUDA
A: With the PGI compiler, the `-ta=tesla:nollvm,keep` compiler flag will generate and keep intermediate CUDA code, although it will not always be simple to understand.

Q: Which optimizations are Open ACC trying to implement? Does it use shared memory in a proper way?
A: In many cases the compiler will try to use shared memory when it sees a way to do so. OpenACC also provides the `cache` directive for explicitly telling the compiler to cache a section of the array into shared memory.

Q: What does "tile(32, 4)" mean in for loop fine tuning?
A: Tiling is a means of exploiting locality in the code, where loop iterations frequently share data with nearby loop iterations. The compiler will add additional loops such that the innermost loops will operate within a 32x4 tile of the J and I loops before moving to the next 32x4 tile. The same total loop operations occur, but the order of the iterations is changed to exploit the locality within these 32x4 tiles.

Q: Can you define asynchronous GNU tasks?
A: Yes. Asynchronous execution will be discussed in a later lecture.

Q: Can one use OpenACC with distributed memory?
A: OpenACC can only be used for on-node parallelism. Another paradigm, usually MPI, must be used for distributed memory machines.

Q: Is this OpenACC library included on CUDA 7.5 Toolkit?
A: Yes. The OTK includes an extensive collection of GPU accelerated libraries. You can find them in linux86-64/2015/cuda/7.5/lib64.

Q: How to know the GPU info on my machine? In linux, we can see cpuinfo by typing "more /proc/cpuinfo" on terminal. Is there similar command for gpuinfo?
A: If you have NVIDIA GPU available on your machine, you could use command “nvidia-smi” to check your GPU information. In the OTK, you could use command “pgaccelinfo”, which will give your more detailed information about the GPU and its driver.

Q: Can the compiler reject a directive if it thinks the loop is not parallelizable?
A: Yes. PGI compiler will analyze the loops and decide whether they’re parallelizable or not.

Q: Is there a way to get an annotated compiler report (like the Cray compilers generate)?
A: Using OTK, you could compile your OpenACC source codes with -Minfo=accel,ccff. This will generate the compiler feedback and report for you.

Q: if I place a loop directive around a loop that has dependencies between iterations without realizing it, will the compiler tell me?
A: Yes. Using OTK, PGI compiler will give you feedback information (-Minfo=accel,ccff) telling you that there’s dependency between the iterations.

Q: Is OpenACC 2.0 functioning now? Specifically, can one parallelize across subroutine boundaries?
A: Yes. PGI compiler now supports most of the OpenACC 2.0 functions. You could find more information in the OTK release note.