Advanced OpenACC Course
Class #1 Q&A

Q1: Is OpenACC compatible with CUDA Code?
A: Yes, you can use both in your code.

Q2: How much GB of model size?
A: Test dataset is 72 MB, but real field data can be several hundred GB or few TB.

Q3: Seeing the full code may be useful. Can you point us to the URL to download the Open Seismic code?
A: ftp://ftp.cwp.mines.edu/pub/cwpcodes/cwp_su_all_43R8.tgz

Q4: acc loop only for inner loop?
A: Yes, used "acc parallel for" on outer loop, and "acc loop" to vectorize inner loop.

Q5: What is the performance cost for using unified memory?
A: For the Kirchhoff test case (2301 x 751 model), we measured 46s elapsed time using managed memory, and 12s using data directives.

Q6: GPUs only tesla or is that generic to any Nvidia GPU?
A: Assume asking about restrictions for Managed Memory support – can be any NVIDIA GPU, but must be 64-bit Linux targets (no Windows) – see more details here.

Q7: Unified memory is logically unified, but physically the CPU and GPU memories are separate. Is that correct?
A: Yes

Q9: How it compares to OpenMP acceleration?
A: In this example, we did not need OpenMP (which may require reorganizing code) – we measured very good speedup with "multicore" option using OpenACC parallel and data directives.

Q10: Are the advantages with NVLink limited to PowerPC systems?
A: No. Xeon-based systems can benefit from NVLINK too. For workloads that require higher memory bandwidth and low latency between host and GPU, then OpenPOWER systems with NVLINK connections between CPUs and GPUs will have an advantage.

Q11: The initial code was not already parallelized, correct? Also, if the code is already parallelized for MPI, will it be as simple as adding the pragma loops?
A: Correct, but at the end we used OpenACC multicore compiler switch to highlight performance portability. For production-level MPI example, please reference this excellent presentation deliver during GTC 2016: "High Performance and Productivity with Unified Memory and OpenACC: A Lattice Boltzmann Case Study".

Q12: Can you show the compiler verbose output with managed data?
A: Yes, here is content shown on slide 37 of presentation today:

```
main:
 453, Generating update host(mig[:noff][:nxo][:nzo])
 455, Generating update host(mig1[:noff][:1][:1])
 459, Generating update host(mig1[:noff][:nxo][:nzo])

resit:
 539, Generating copyin(ttab[:ns],tb[:][:nz])

sum2:
 571, Generating copyin(t2[:nx][:nz],t1[:nx][:nz])
      Generating copyout(t[:nx][:nz])

mig2d:
 721, Generating copy(ampt1[nxtf:nxte-nxtf+1][:])
      Generating copyin(cssum[nxtf:nxte-nxtf+1][:],tvsum[nxtf:nxte-nxtf+1][
      Generating copy(tmt[nxtf:nxte-nxtf+1][:],ampti[nxtf:nxte-nxtf+1][:])
      Generating copyin(pb[:][:])
      Generating copy(ampt[nxtf:nxte-nxtf+1][:])
      Generating copyin(cs0b[:][:],angb[:][:])
      Generating copy(zpt[nxtf:nxte-nxtf+1])
 782, Generating copy(mig1[nxf:nxe-nxf+1][:])
      Generating copyin(ampt1[:][:], tb[:][:], tsum[:][:], ampt[:][:], ...          Generating copy(mig[nxf:nxe-nxf+1][:])
      Generating copyin(zpt[:])
```

Q13: Can we use FORTRAN 95 with this?
A: Yes

Q14: Which is the "restrict" keyword in pgfortran?
A: Don't need restrict with FORTRAN – is primarily needed to guide compiler that safely optimize code without pointer aliasing concerns.

Q15: How does the GPU performance compare to using all of the CPU as opposed to just one core?
A: Near the end of the presentation we compare results using all CPU cores, but not by having to add OpenMP directives, nor pthreads, but rather with "multicore" option using OpenACC parallel and data directives.

Q16: On the timing slide, is the CPU speed on 1 core or 1 multi-core CPU?
A: Near the end of the presentation we compare results using all CPU cores, but not by having to add OpenMP directives, nor pthreads, but rather with "multicore" option using OpenACC parallel and data directives.

Q17: How does the GPU performance compare to using all of the CPU as opposed to just one core?
A: Near the end of the presentation we compare results using all CPU cores, but not by having to add OpenMP directives, nor pthreads, but rather with "multicore" option using OpenACC parallel and data directives.

Q18: Does managed memory approach support multiple GPUs?
A: Yes, using MPI to associate a rank with an individual GPU – to learn more details about strong scaling across clustered multi-GPU systems, please reference this excellent presentation deliver during GTC 2016: "High Performance and Productivity with Unified Memory and OpenACC: A Lattice Boltzmann Case Study". Support for multiple devices using this approach were added to PGI 15.4 and later releases.

Q19: Any pragma acc of "routine" usage in this code?
A: No.

Q20: When using the data region - what is then the advantage of the managed memory if we specify what data to move in/out through the data region? Are there some other data being moved automatically by the managed memory that I missed from the slides?
A: We use the verbose compiler output in combination with the “managed” compiler option to help us understand how the underlying system is migrating data. This information is used to help us insert explicit data directives. Once the data directives are inserted, the “managed” compile option is no longer used.

Q21: Good idea to just use collapse() in general? I can see situations where it probably won't improve anything if each loop size is large enough, are there any caveats? I ask because I've never used it and consistently see 100% occupancy in the profiler
A: In general, collapse() will increase number of threads, which may increase occupancy, but other resource limitations may cause performance inhibitors.

Q22: What is the difference between OpenMPI and OpenACC?
A: OpenMPI is an particular implementation of MPI (another example is MVAPICH2) which is a message-passing API that allows distributed computing with barriers/reduces/send/receive communications support. OpenACC is a directives-based approach which can work in concert with MPI-based products (like OpenMPI and MVAPICH2). You can use MPI to associate an individual rank (sort of like an individual process) with each GPU in a multi-GPU system, and across clustered GPU nodes, and from that rank use OpenACC directives to: parallelize multiply nested loops, manage data migration, specify asynchronous behavior, adjust schedules, etc.

Q23: How does the compiler retrieve where the "present" variables are on the CPU? Is that done at runtime?
A: The data directives are interpreted at compile time (and the compiler generated very verbose output helping developers understand explicitly how data is migrated). Careful analysis of compiler and profile information allow the developer to determine where to use the “present” clause.

Q24: Can I profile arbitrary FORTRAN compiler in a program calling CUDA with OpenACC?
A: You can mix CUDA FORTRAN and OpenACC directives, and use PGPROF to profile your code.

Q25: What would be the impact of using Multi-CPU and GPU acceleration together?
A: Here is a nice example that shows strong scaling across multi-CPU + multi-GPU cluster nodes by combining MPI + OpenACC + Managed memory to accelerate a Lattice Boltzmann code.

Q26: We can combine OpenACC with cuBLAS to do parallel loops and BLAS calls. But in multicore, how would we do parallel loops and BLAS calls?
A: Use conditional (#ifdef) macros to use proper library for GPU or CPU.

Q27: When deploying an OpenACC compiled binary, are there options for statically linking with the library? Is this library available for most platforms and with a redistributable license?
A: Yes for both.

Q28: Is this possible to use one OpenMP to run on GPU and other threads do their own job?
A: Yes.

Q29: The example shows code with simple nested loops, how the performance would change if you have "if" statement inside the nested do?
A: If statements are supported, but may require careful attention to messages printed from verbose compiler output. Here is an example using if-statements with OpenACC.

Q30: Have you tested this case with CUDA? And what about the comparison between OpenACC and CUDA?
A: No. With CUDA the developer has more control over shared memory and registers which can yield significant performance uplift.

Q31: I have used pgprof as a part of the PGI compilers. Is pgprof now available independently or does one need a PGI compilers license?
A: You will need a PGI compiler license to get PGPROF.

Q32: Can you use pgprof with MPI code?
A: Currently it does not support profiling MPI enabled applications.

Q33: what version of pgprof is this?
A: This is the 2016 version of PGPROF which started shipping with PGI 16.1.

Q34: Are the same functionalities available with NVPROF?
A: Yes

Q35: What is the advantage of using pgprof over nvprof?
A: There is a difference in default behavior between the two. PGPROF enables CPU profiling by default whereas NVPROF does not. Also, if you have the PGI compiler/tools installed then you would not need to also install the CUDA SW just to use the profiler.

Q36: When is OpenACC support expected with main stream compilers such as Visual Studio?
A: PGI has a product called PGI Visual Fortran which supports OpenACC. For more information see the following URL: http://www.pgroup.com/products/pvf.htm

Q37: How about OpenMP on Multicore?
A: PGPROF supports profiling OpenMP enabled applications on Multicore CPU's.

Q38: Is it possible to target GPU with some OpenACC pragmas, and multicore with others in the same program?
A: Currently this is not supported.

Q39: Is it possible to have both multicore and GPU kernels within the same executable?
A: Currently this is not supported.

Q40: Does the visual profiler work in a client-server (remote) setting?
A: Yes