Q1: Can this be done with tile (M,N) directive, without changing the code? Slide 21
A: That's an interesting idea and we could use the tile clause on the y and x loops, but these loops will still be executing in the same GPU kernel, so it won't be possible to progress to the steps that follow to enable overlapping of copies and execution. By explicitly blocking the loops up by hand, we're able to break up the work into multiple kernel launches and multiple data transfers.

Q2: Please explain the timeline. Slide 25
A: The tan colored boxes are data copies over PCIe. The blue/green boxes are kernel execution. Reading from left to right is elapsed time.

Q3: Can one achieve overlapping of two kernels? Slide 27. Can I fill the machine with multiple kernels at the same time?
A: Yes, if multiple kernels are launched on the same device and neither can saturate the device, it's possible that they will overlap. This is a decision made by the hardware, so it's not something that you can force to happen. This commonly happens as one kernel finishes, freeing some resources, and the next kernel begins to fill those resources.

Q4: Does OpenACC know that if all the data has been transferred to the host by update clauses that it need not actually do the copyout as specified at the end of the data region? Does it keep track of whether the data has been modified on the device after an update clause on that data?
A: No, the copyout clause specifies that the data must be copied out at the end of the region, so the copyout must be changed to a create to prevent data movement outside of our update directives.

Q5: Is the n parameter for Async(n) the specific Block of the timeline?.
A: The parameter to async corresponds to CUDA streams. The additional async queues/cuda streams show up in visual profiler as additional horizontal rows in the timeline, allowing you to see when activities in different queues overlap.

Q6: Is this homework also available on a github site as was the case for lessons 2 and 3?
A: Yes, please see the labs/ directory in https://github.com/NVIDIA-OpenACC-Course/nvidia-openacc-course-sources. They are also available on qwiklabs for free through the end of the course. The qwiklabs version have full instructions to complete the lab, but the github version currently only has the sources.

Q7: where is this block%2 set? Slide 31
A: The block variable is used by the loop we added over blocks and the %2 causes the queue number to alternate between 0 and 1. In the later example we added 1 to this number to avoid the "0" stream, which may have special behavior on some devices.

Q8: Or is the "n" parameter the Block size? Slide 33
A: The parameter to async(n) is the block number, not the block size.

Q9: Where can we get the Mandelbrot code itself. I'd like to use our own GPU cluster. I don't see how on wiklab (maybe I'm missing something?). Thanks.
A: Please see https://github.com/NVIDIA-OpenACC-Course/nvidia-openacc-course-sources/tree/master/labs/lab4.pipelining

Q10: Is it possible to develop the programming for a multiple 8x K80 GPU system, and write the code for one GPU and have the programming transferred to the remaining GPUs so they all perform the same action simultaneously to accelerate computational data processing at full capacity?
A: It's not currently possible to do this automatically. The issue is that the data needs to be resident on the *right* device for the work being sent to that GPU. The gangs of work are independent, so in theory they could be spread to any devices, but the location of the data is a big issue. This is a really challenging problem that may become easier on future architectures. If you happen to be doing a lot of linear algebra, you should look at cublas-XT, which has some ability to scale across GPUs using a technique similar to what we did in the mandelbrot example.

Q11: Then explain to why and how companies like Cray, Amax, Dell & SGI are selling 4-way & 8-way GPU HPC systems and making claims that their systems are capable of allowing all 4-way or 8-way GPU nodes to function simultaneously under full capacity of every single GPU? I am not crazy! I confirmed this three times with their engineers of each company.
A: It's absolutely possible to execute across all of the GPUs available on this machine, but the programmer must explicitly manage dividing the work across the GPUs in the system. It's not possible to apply a single "acc kernels" or "acc parallel loop" directive and expect the compiler to divide the work automatically among all of the GPUs. The techniques shown in the mandelbrot example will scale across 8 GPUs and are similar to techniques used to scale CUDA programs as well. Using MPI, as shown in the later example, also allows the work to scale across any number of GPUs on a node, typically by assigning a different MPI process/rank to each GPU. One exception to this rule about automatic division of the work is the cublas-xt library, which handles pipelining the work across all available devices automatically for you.

Q12: Does the new PGI x86 feature for OpenACC work with code like this? (OpenACC+MPI)?
A: Yes but the async clause won't give overlapping of execution. This is no issue for the mandelbrot example as with the multicore target there will be no data transfers. But the technique to get overlap with MPI in the second part of the lecture won't work when compiling for multicore. You can find more information about the multicore support for OpenACC in Michael Wolfe's blog post "OpenACC for Multicore CPUs"
http://www.pgroup.com/lit/articles/insider/v6n3a1.htm

Q13: When Jeff refers to 2 Gpu in the last example, Is he refer multi-gpu card?.
A: It can be a mult-gpu card, such as a K80, or 2 single-gpu cards, such as 2 K40s. These examples have been tested in both ways.

Q14: Can OpenACC work with libraries the other way, with libraries allocating and initializing data on GPU?
A: Yes, there is an example of this https://github.com/NVIDIA-OpenACC-Course/openacc-interoperability. Specifically, the "deviceptr" clause informs the OpenACC compiler that the memory came from outside of OpenACC, such as coming from CUDA.

Q16: When using MPI with the device pointers, does the code automatically use nvdirect (nvlink?) if available?

A: If 2 GPUs communicate directly to each other, we call this technology GPUDirect P2P. the MPI library will utilize P2P path whenever it is available.