Scaling GPU Compute Performance

White Paper

blades without boundaries™
www.cirrascale.com

Cirrascale

Scaling GPU Compute Performance
M987 – 06 01 15

Introduction

The use of GPUs for general purpose computing is now well out of its infancy, and the challenge faced by researchers, software developers, and engineers for applications such as deep learning, molecular dynamics and high performance computing has shifted from *"How can I make use of a GPU?"* to *"How can I get more performance out of GPUs?"*

For their part, GPU chip manufacturers – such as NVIDIA® Corporation – have been providing more computational power within each GPU card. Recently, this includes packing multiple GPUs within each card, such as the NVIDIA® Tesla® K80 Dual-GPU Accelerator cards which contain two GPUs per card. Additionally, with new iterations, they continue to increase overall memory size and bandwidth accessible to those GPUs.

While significant, the improvements to performance of individual GPUs are dwarfed by the gains that can be achieved by simply adding more GPU cards to handle the work… but it's not quite that simple.

Data Flows

Using a single GPU card in a system, the pattern for data access is pretty well established:

1. The host CPU pulls some data from persistent storage, and optionally does some pre-processing on it.
2. The GPU then pulls the data from the host memory into its local memory, and performs work on that data.
3. Finally, the CPU pulls the results back from the GPU into host memory.

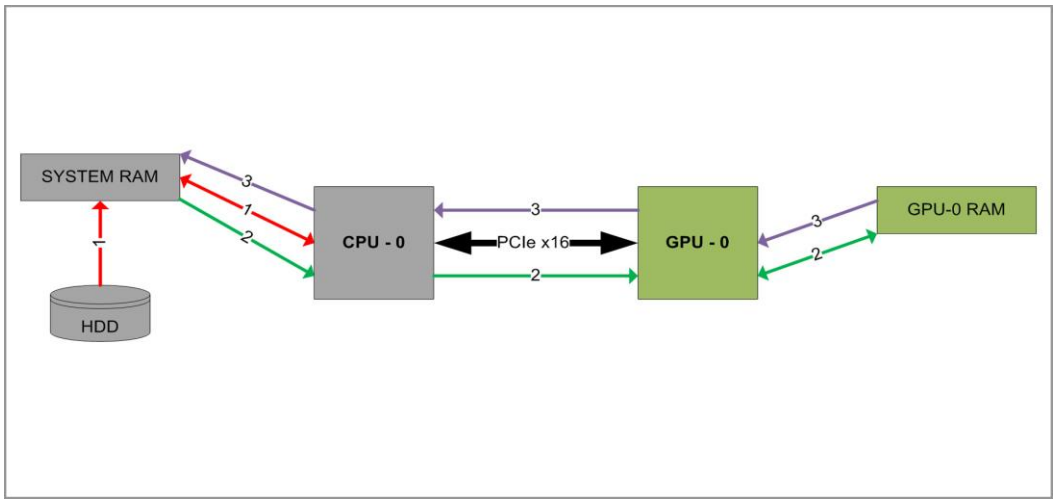

Figure 1: Showing the data flow for a system with a single GPU card.

This process repeats as necessary to increment time-steps, work on the next section of a matrix, or whatever the specifics are of the work being performed. There are obviously some variations on this -- such as overlapping data transfers to ensure the GPU isn't starved for data -- but the overall principal remains the same: data is shuffled back and forth between the host and the GPUs memory so that the GPU has data to operate on.

When a second GPU is added to a system, the single-GPU data access pattern can be re-used only if the work being done is embarrassingly parallel. If that's not the case and the work to be done by the GPU depends upon prior output, or there is more than one type of work that the GPU needs to perform on the data, then the data access pattern changes. It is easy to envision a case where the data access pattern looks like:

1. The host CPU pulls in some data from persistent storage, and optionally does some pre-processing on it.
2. GPU-0 then pulls the data from the host memory into its local memory, and performs work on it.
3. The CPU pulls the interim results from GPU-0 into host memory.
4. GPU-1 then pulls the post-GPU-0 processed data into its local memory to do the second type of work on it.
5. As the last step, the CPU pulls the final results back from GPU-1.

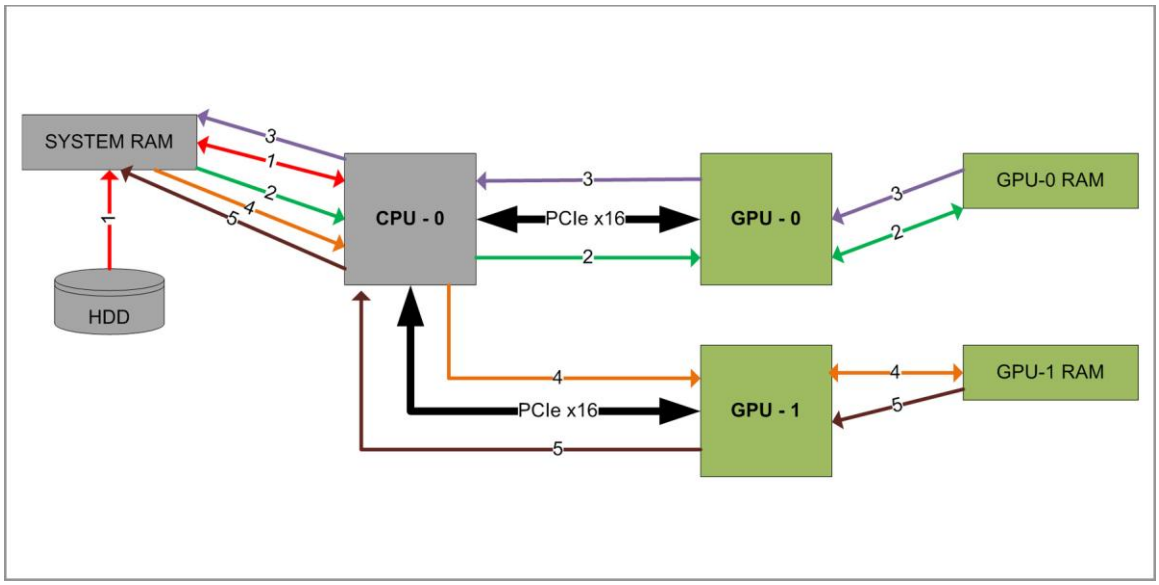

Figure 2: Showing the data flow for a system with dual GPU cards.

Looking at where work is performed shows that Steps #3 and #4 of the above are just busy-work actions, mainly moving data but not manipulating it. This can be solved by allowing GPU-1 to directly access the memory of GPU-0, which would ultimately lead to the following data access pattern instead:

1. The host CPU pulls in some data from persistent storage, and optionally does some pre-processing on it.
2. GPU-0 then pulls the data from the host memory into its local memory, and performs some work.
3. GPU-1 then pulls the post-GPU-0 processed data into its local memory to do the second type of work on it.
4. As the last step, the CPU pulls the final results back from GPU-1.

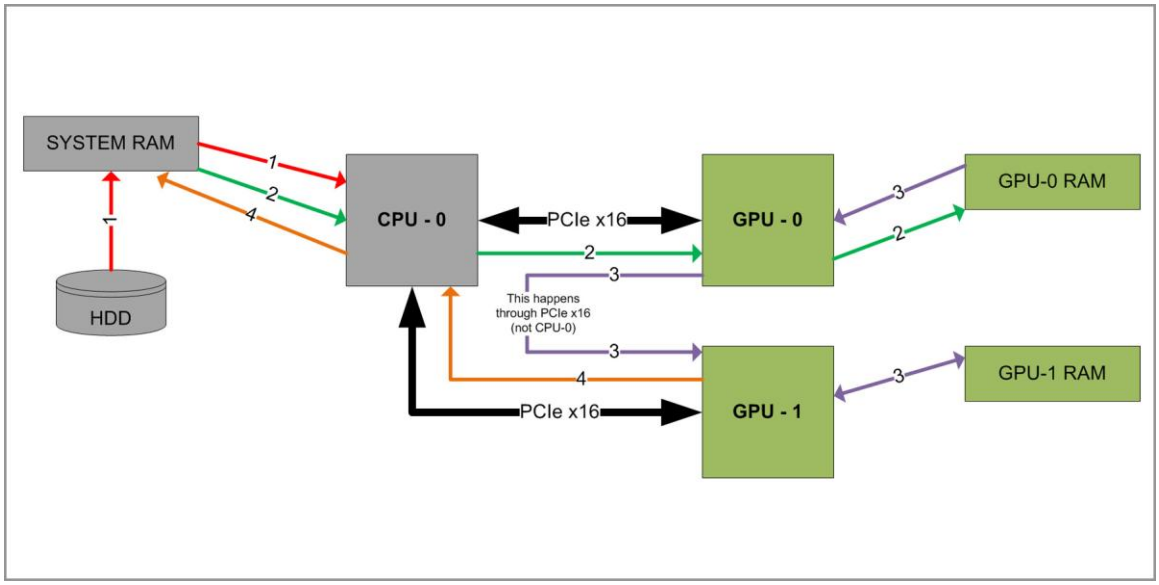

Figure 3: Showing the data flow for a system with dual GPU cards with peer-to-peer communication.

This ability for moving data by way of a DMA transfer between the GPUs has a number of benefits, including reduced load on the host CPU (since it's no longer pushing as many bits around), reduced load on the host memory bus (since there's much less data going to or from host memory), and a decrease in the latency of the data copy. While the decrease in CPU cycles and host memory usage is often (but not always) negligible, the change in latency is not.

As an example, utilizing a pair of NVIDIA Tesla K40 Accelerators (GPUs) on a PCIe root complex of an Intel® Xeon® processor E5-2600 v3 series with DDR4 memory, the time for NVIDIA CUDA to copy one byte of data between GPUs via DMA is only 6.67µs compared to 20.68µs when going through the CPU. For chatty inter-GPU communication protocols, this can result in a significant performance change! As more and more GPUs are added to a system, and data-flows become more varied, the overhead of trips to and from the CPU can quickly remove much of the potential benefit of the additional GPU compute capability.

Diving Deeper

Given that increasing the number of GPUs in a system is an effective way to grow aggregate performance, and moving data directly between GPUs is more efficient than multiple trips for data to and from the host, *what are the options for designing a system?*

The naive way of increasing the GPU count in a system is to simply attach more GPUs to the PCIe root complex. The first issue that arises from this solution is the limited number of PCIe lanes made available by PCIe root complexes available today.

On current x86 architectures, the CPU contains the PCIe root complex, but the Intel® Xeon® processor E5-2600 v3 product family only offers 40 lanes of PCIe Gen3. Additionally, current ARM64 and POWER8 CPU architectures offer even fewer PCIe lanes. Since most GPUs can make full use of PCIe x16 bandwidth, this means at most two GPUs can be attached to a single Intel® Xeon® processor without compromising performance (*See Figure 4*). While a doubling of GPU capacity in a system is great, it's only doubling. Let's set our target a little higher than that.

Of course, the Intel® Xeon® processor E5-2600 v3 product family supports more than one CPU in a system, so adding a second CPU is a relatively easy way to double the number of PCIe lanes, and hence the number of GPUs in the system. However, it's not quite that simple.

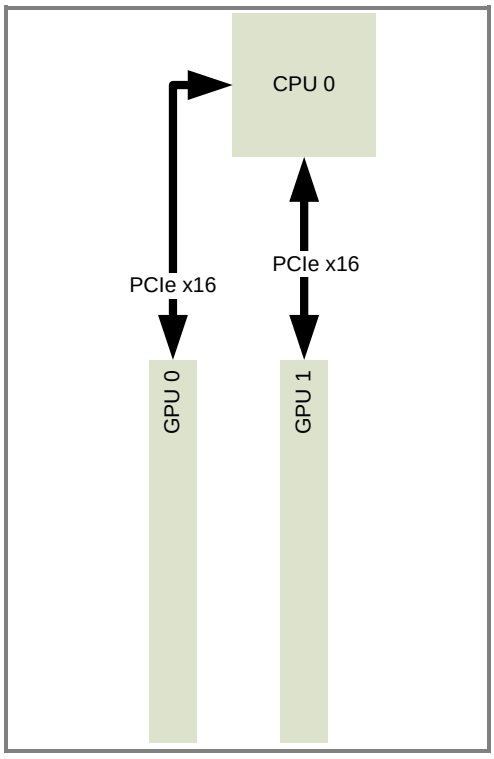

Figure 4: Diagram of Intel Xeon processor with two GPUs connected via PCIe Gen3.

Because each Intel® Xeon® processor is also the PCIe root complex, GPUs attached to one CPU cannot have direct memory access (DMA) to those GPUs on the other CPU. Instead, data must traverse a pair of PCIe root complexes and the QPI link connecting both CPUs, making this configuration highly undesirable from a performance standpoint (*See Figure 5 on next page*).

Even if this configuration wasn't detrimental to the overall performance, there is a marked cost to scaling in this fashion since each pair of GPUs necessitates the purchase of an additional CPU and associated host memory.

For example, to build a multi-GPU system

consisting of eight GPUs this way would require purchasing four times the host RAM and CPU (and CPUs that support 4-way SMP are rarely cheap), even though they are only being used for their PCIe lanes.

However, we can still do better than that.

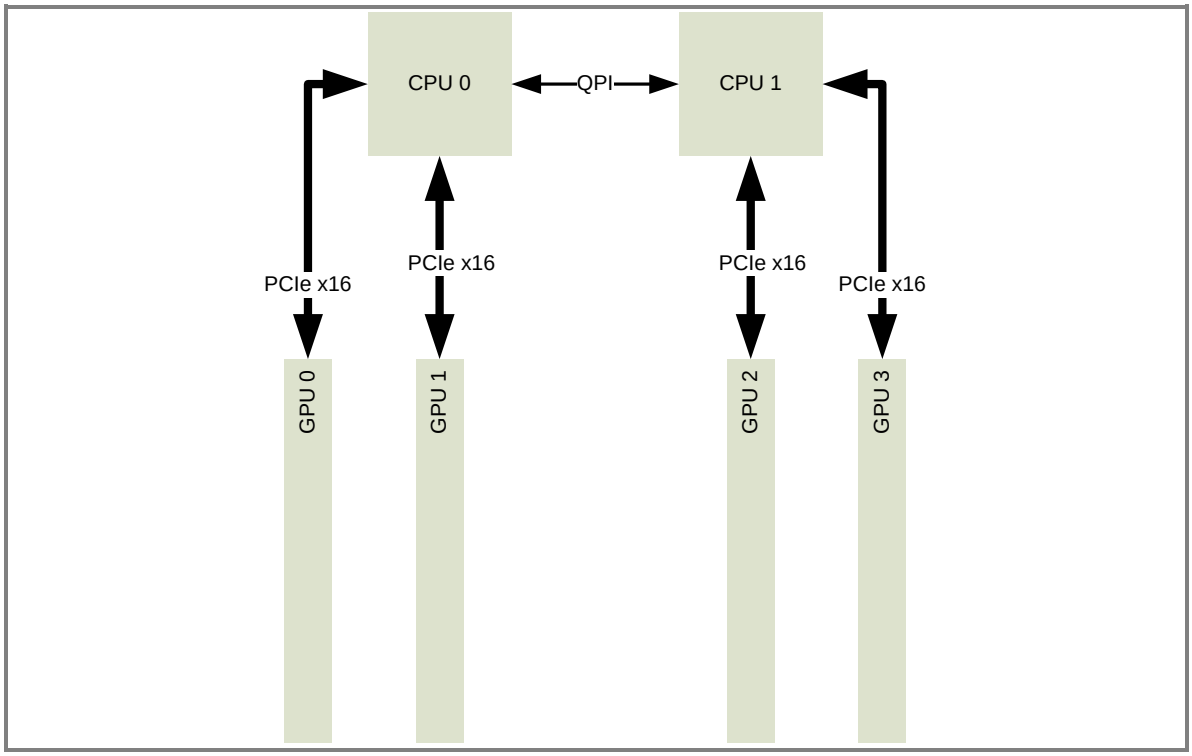

Figure 5: Diagram of Intel Xeon processors with multiple GPUs connected via PCIe Gen3.

NVIDIA, as one of the leaders in GPU computing, has also recognized that having multiple GPUs that can DMA amongst each other is an excellent way to scale system performance. To further multi-GPU compute efforts, the NVIDIA Tesla K80 Dual-GPU Accelerators incorporate two GPUs and a PCIe switch onto a single card (this was also done with earlier NVIDIA Tesla K10 cards, but for other reasons. Mainstream NVIDIA Tesla cards since the Tesla K10 were single GPU up until the Tesla K80s). Using a pair of dual-GPU accelerator cards in a system yields four GPUs, all able to DMA to each other.

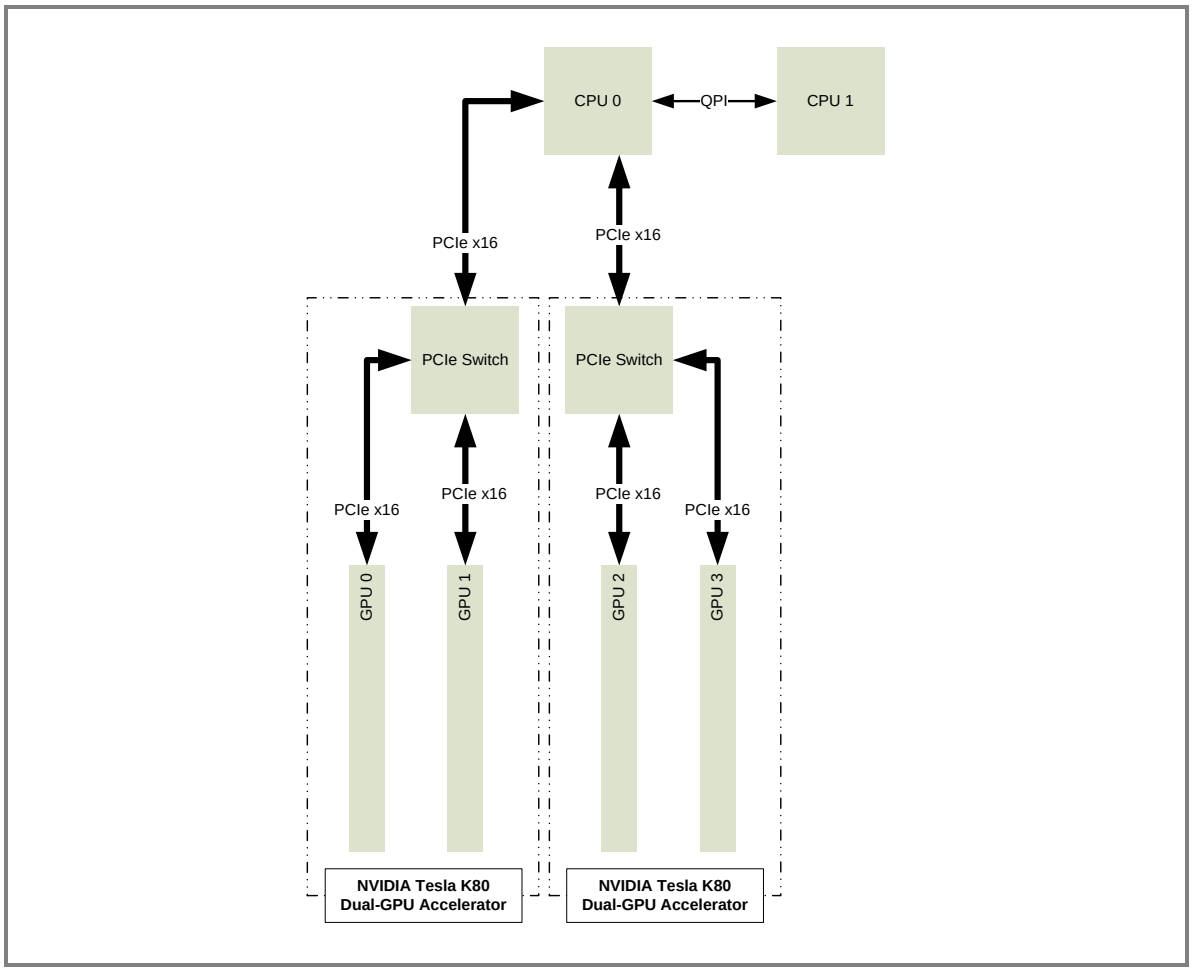

Figure 6: Diagram of Intel Xeon processors with multiple NVIDIA Tesla K80s connected via PCIe Gen3.

Of note is that some data-paths between GPUs in the configuration above traverse not only the PCIe Root Complex, but also a pair of PCIe switches. Fortunately, PCIe switches from manufacturers such as Avago (formerly PLX Technologies) have outstanding performance, and induce a negligible amount of latency.

Going back to the one byte memory copy test mentioned previously, the latency of a transfer in the above configuration (*see Figure 6*) is less than 0.2µs (around 3%) more than without the pair of switches. For communication between GPUs that are on the same PCIe switch, the story is even better as the latency between ports on the PCIe switch is typically lower than the PCIe root complex. Using the same one byte memory copy, going between GPUs on the same PCIe switch is typically 0.1µs faster than through the PCIe root complex. Quadrupling of the GPU compute capabilities while preserving the ability to transfer data directly between any combination of GPUs is usually worth an almost unnoticeable increase in worst-case latency, and certainly worth it for a potential improvement in best-case latency.

Another way to achieve four GPUs in a single system is to leverage PCIe switches in a different way. Instead of using two GPUs per PCIe switch, why not use four? The

Cirrascale SR3514 PCIe switch riser makes use of a more advanced PCIe switch than is present in the NVIDIA Tesla K80 Dual-GPU accelerators to provide four PCIe x16 slots on a single switch.

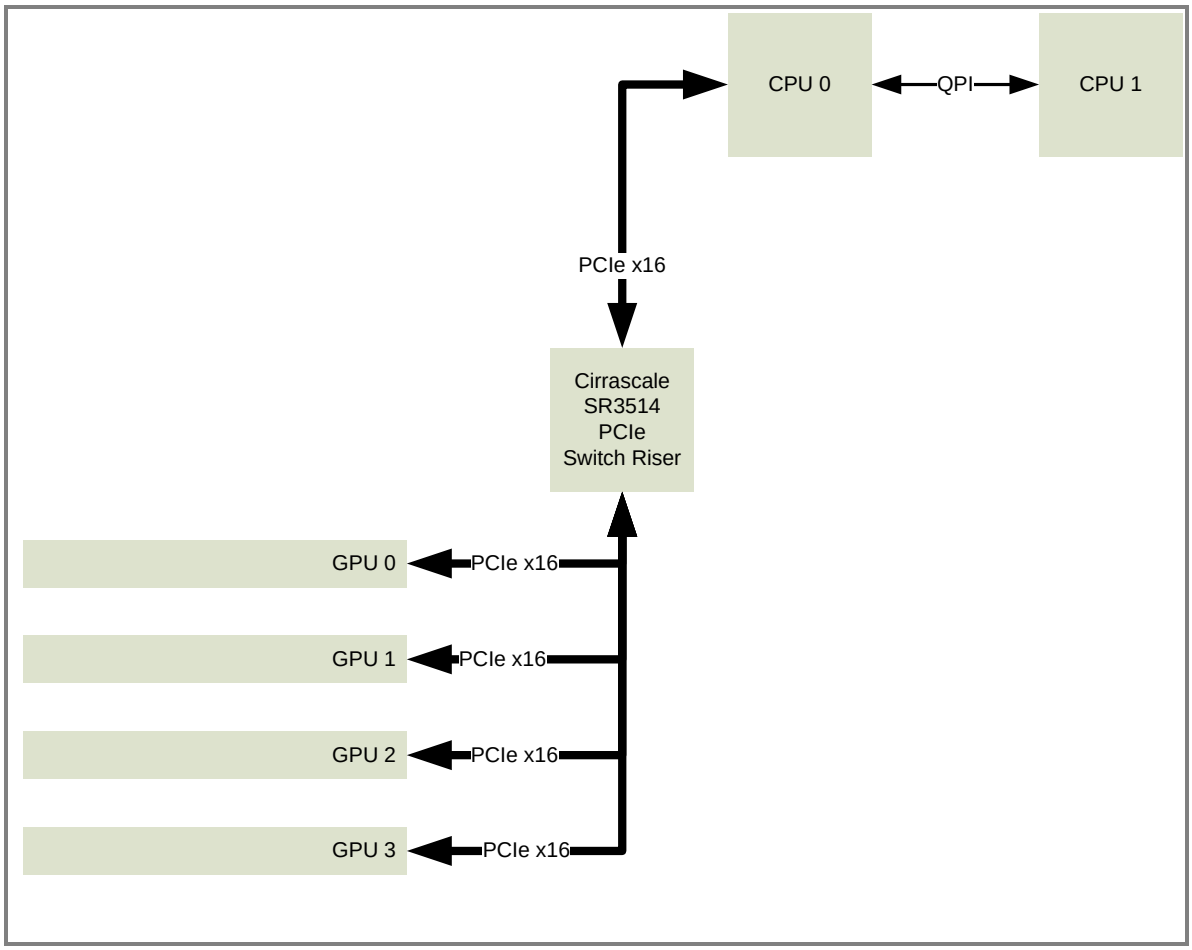

Figure 7: Diagram of Intel Xeon processors with four K40s connected via Cirrascale SR3514.

With this configuration, all GPUs can DMA each other, and data follows an identical path regardless of which GPUs are communicating. Granted, the latency added by the PCIe root complex and the extra PCIe switch in the previous configuration is pretty negligible.

Less obvious, but more important, this configuration reduces the requirements on the host system. Whereas the previous configuration required 32 lanes of PCIe x16 from the PCIe root complex (16 lanes for each of the two Tesla K80 cards), this configuration requires only 16, making it suitable for a much wider variety of hosts.

Recall earlier where POWER8 and 64-bit ARM systems have even fewer PCIe lanes than the Intel® Xeon® processor E5-2600 v3 product family? The configuration implemented by the Cirrascale SR3514 PCIe switch riser enables these multi-GPU configurations even on those hosts. Alternatively, reducing the PCIe lanes used by GPUs on the PCIe root complex allows those lanes to be used for other devices, such as InfiniBand or NVMe cards, also directly accessible from the GPUs without intervention by the host.

Better Together

The fact that there are multiple ways to achieve a system with four GPU cards doesn't mean they need to be used in isolation. Because the data transfer latency induced by the presence of a PCIe switch is small relative to the overall transaction latency, system designers can take advantage of nested PCIe switches.

Combining the NVIDIA Tesla K80 Dual-GPU accelerator cards (which include a PCIe switch) with the Cirrascale SR3514 (which also includes a PCIe switch) enables an eight GPU card (16 internal GPUs) configuration while avoiding significant impacts to latency, such as that which would be seen traversing a QPI link (recall the 6.67µs jumping up to 20.68µs from the discussion earlier). This configuration also only requires a singular PCIe link to the PCIe root complex, making an eight GPU configuration possible even on hosts constrained by the number of available PCIe lanes.

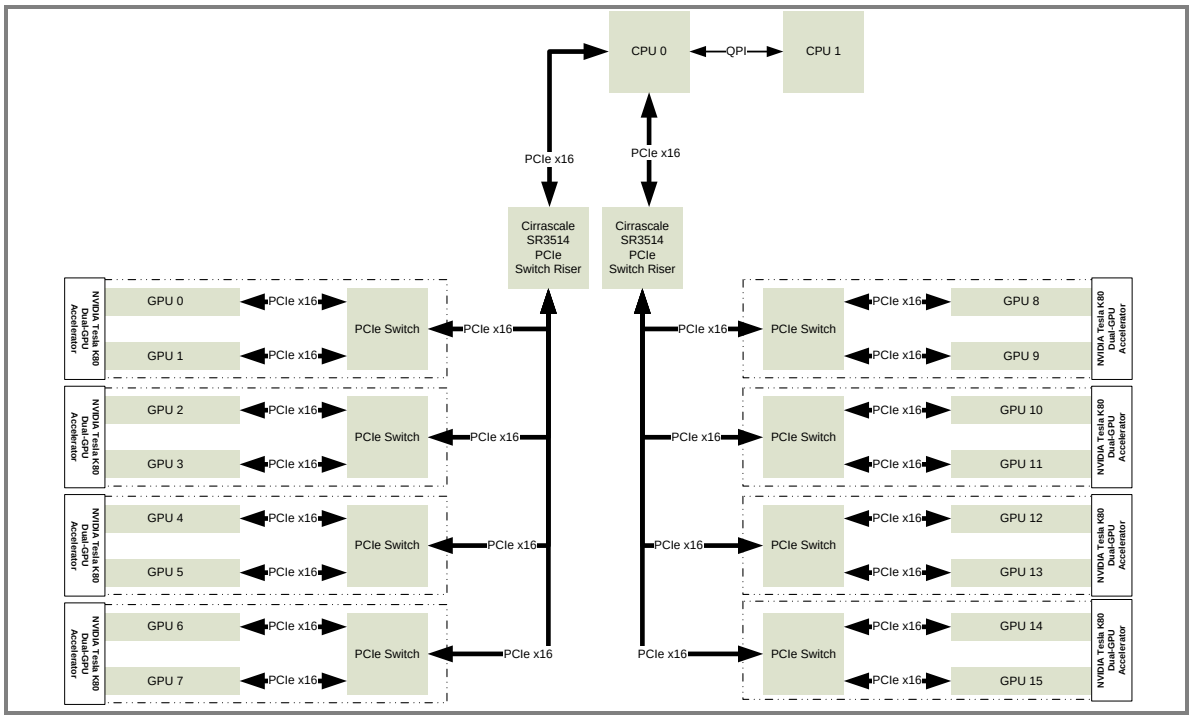

Figure 8: Diagram of Cirrascale GB5400/GB5600 Series system topology.

Because the Intel® Xeon® processor E5-2600 v3 product family offers enough PCIe lanes for two PCIe Gen 3 x16 links, the 8 GPU configuration can be doubled to 16 GPUs using a combination of NVIDIA Tesla K80 Dual-GPU accelerators and a Cirrascale SR3514 PCIe switch riser.

In fact, products such as the Cirrascale GB5400/GB5600 Series are configured this way, and allow up to 16 GPUs to pass data directly between each other while maintaining transfer latencies consistent with 4 and 8 GPU configurations. Configurations such as this, providing up to 16 times the GPU compute performance as a single GPU, enable

not only a large amount of raw compute power, but the flexibility to support complex and varied data flows typical of modern GPU compute applications.

###

Cirrascale Corporation
12140 Community Road
Poway, CA 92064

Phone: (858) 874-3800
Web: www.cirrascale.com