Whitepaper

# NVIDIA® NVLink™ High-Speed Interconnect: Application Performance

**November 2014**

# Contents

## Accelerated Computing as the Path Forward for HPC

Accelerated systems have become the new standard for high performance computing (HPC) as GPUs continue to raise the bar for both performance and energy efficiency. In 2012, Oak Ridge National Laboratory announced what was to become the world's fastest supercomputer, *Titan*, equipped with one NVIDIA® GPU per CPU – over 18 thousand GPU accelerators. *Titan* established records not only in absolute system performance but also in energy efficiency, with 90% of its peak performance being delivered by the GPU accelerators.

Since *Titan*, a trend has emerged toward heterogeneous node configurations with larger ratios of GPU accelerators per CPU socket, with two or more GPUs per CPU becoming common as developers increasingly expose and leverage the available parallelism in their applications.

Recently, the U.S. Department of Energy announced its plans to build two of the world's fastest supercomputers – the *Summit* system at Oak Ridge National Laboratory and the *Sierra* system at Lawrence Livermore National Laboratory – which are each expected to have well over 100 petaFLOPS of peak performance. Although each system is unique, they share the same fundamental multi-GPU node architecture.

While providing a vehicle for scaling single node performance, multi-GPU applications can find themselves constrained by interconnect performance between the GPUs. Developers must overlap data transfers with computation or carefully orchestrate GPU accesses over PCIe interconnect to maximize performance. However, as GPUs get faster and GPU-to-CPU ratios climb, a higher performance node integration interconnect is warranted. Enter NVLink.

## NVLink: High-Speed GPU Interconnect

NVLink is an energy-efficient, high-bandwidth path between the GPU and the CPU at data rates of at least 80 gigabytes per second, or at least 5 times that of the current PCIe Gen3 x16, delivering faster application performance. NVLink is the node integration interconnect for both the *Summit* and *Sierra* pre-exascale supercomputers commissioned by the U.S. Department of Energy, enabling NVIDIA GPUs and CPUs such as IBM POWER to access each other's memory quickly and seamlessly. NVLink will first be available with the next-generation NVIDIA Pascal™ GPU in 2016.

In addition to speeding CPU-to-GPU communications for systems with an NVLink CPU connection, NVLink can have significant performance benefit for GPU-to-GPU (peer-to-peer) communications as well. **This paper focuses on these peer-to-peer benefits from NVLink**. We will show how systems with next-generation NVLink-interconnected GPUs are projected to deliver considerable application speedup compared to systems with GPUs interconnected via PCIe.

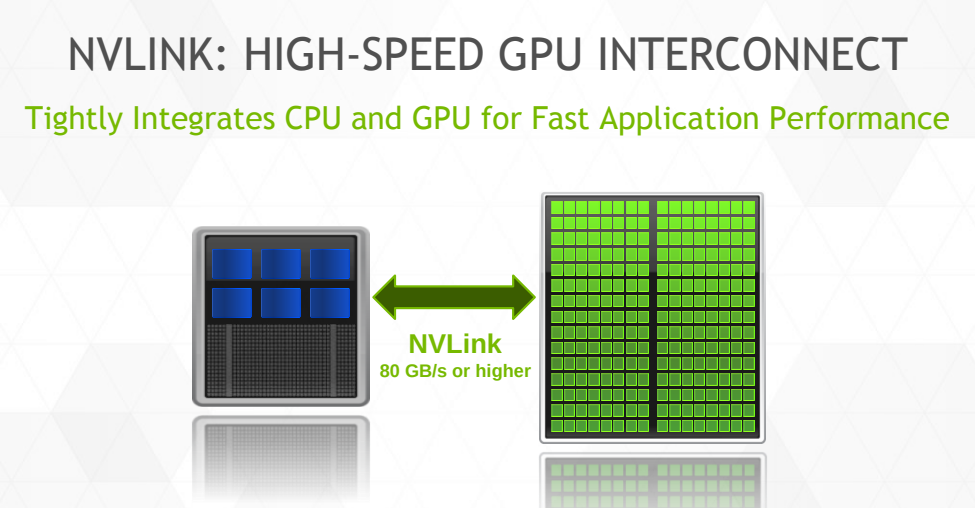



Figure 1: NVLink delivers 80 GB/s or higher bandwidth to enable faster communication in a node.

## Server Configuration with NVLink

In the following sections of this paper, we analyze the performance benefit of NVLink for several algorithms and applications by comparing model systems based on PCIe-interconnected next-gen GPUs to otherwise-identical systems with NVLink-interconnected GPUs. GPUs are connected to the CPU using existing PCIe connections, but the NVLink configurations augment this with interconnections among the GPUs for peer-to-peer communication. The following analyses assume future-generation GPUs with performance higher than that of today's GPUs, so as to better correspond with the GPUs that will be contemporary with NVLink.

Table 1: List of assumptions in this paper for NVLink application performance analysis.

| Assumptions | NVLink | PCIe Gen3 |
|---|---|---|
| **Connection Type** | 4 connections | 16 lanes |
| **Peak Bandwidth** | 80 GB/s | 16 GB/s |
| **Effective Bandwidth** | 64 GB/s | 12 GB/s |
| **GPU** | Future-generation GPU | |

For purposes of this study, we will assume that each GPU has four NVLink connection points, each providing a point-to-point connection to another GPU at a peak bandwidth of 20 GB/s. We will further assume an effective bandwidth similar to what is seen on PCIe, which is around 80% of peak, yielding an assumed effective bandwidth of 16 GB/s per NVLink connection in both directions simultaneously. Multiple NVLink connections can be bonded together, multiplying the available interconnection bandwidth between a given pair of GPUs.

The first scenario, shown in Figure 2, compares a pair of configurations with two GPUs each, referred to throughout this paper as *2-GPU-PCIe* and *2-GPU-NVLink*. 2-GPU-NVLink provides a fast NVLink interconnect between the two GPUs, bonding together all four of the NVLink interconnection points for a total peak bandwidth of 80 GB/s (64 GB/s effective) per direction between them. By contrast, 2-GPU-PCIe, reflecting a common configuration seen in production today, requires that peer-to-peer communication share the same PCIe links as are used for communication with the CPU.

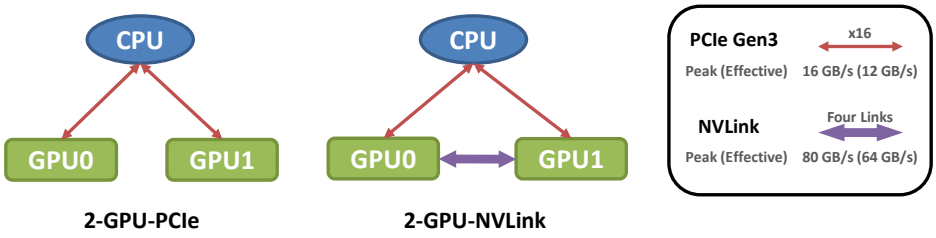



**Figure 2: Comparing 2-GPU topologies with NVLink and PCIe. GPU0 and GPU1 are connected with 80 GB/s peak bandwidth when using NVLink.**

The second scenario, shown in Figure 3, compares a pair of configurations with four GPUs each, referred to as *4-GPU-PCIe* and *4-GPU-NVLink*. 4-GPU-PCIe uses a PCIe tree topology, again mirroring a configuration used commonly today. 4-GPU-NVLink enhances this by providing point-to-point connections with either one or two NVLink connections per pair of GPUs, yielding 16 GB/s or 32 GB/s effective bandwidth per direction per pair, respectively.

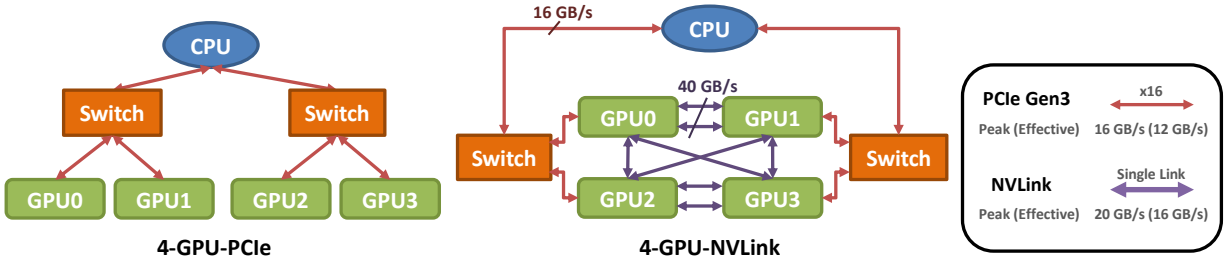



**Figure 3: Comparing 4-GPU topologies with NVLink and PCIe. In 4-GPU-NVLink, GPU0 and GPU1 have 40 GB/s peak bandwidth between them, as do GPU2 and GPU3. The other peer-to-peer connections have 20 GB/s peak bandwidth.**

## APPLICATION 1: MULTI-GPU EXCHANGE AND SORT

Sorting is an important algorithmic pattern used throughout computing. In this section, we will look at the performance of a multi-GPU exchange and sort algorithms.

The multi-GPU exchange algorithm exchanges values on GPUs according to a hashing function. This involves locally sorting based on the hash and then sending data via an all-to-all communication algorithm to the final location. The multi-GPU sorting algorithm first performs the exchange as a coarse-grained "sort" (binning) and then sorts all the binned keys locally to complete the overall sort.

The exchange algorithm begins with data evenly distributed across multiple GPUs as shown below.

**Original Data**


In this diagram, the color indicates which device the original data began on.

The first step is to hash the data to determine which GPU's bin each element should be placed in. For sorting, this might be as simple as extracting the highest order bits, but in theory we could apply any arbitrary hashing function to determine the keys (i.e., the destination bins). For this example, we will use the hash function `ceil(x/5)`, which produces the following keys given the data above.

**Calculated Keys**


Next we locally sort the data based on the hashed keys, yielding the following. Note that since we know these keys are in the range of `[0, NUM_GPUS)`, we can restrict the sorting to only a couple of bits.

**Locally-sorted Data**


**Locally-sorted Keys**


Next, we transfer all data with a key of 0 to device 0 and all data with a key of 1 to device 1. This becomes an all-to-all exchange among GPUs, producing the following.

**Exchanged Data**


Multi-GPU sorting is a simple extension of the above. After exchanging the data, we simply sort the data again locally to produce the final sorted list.

**Fully-sorted Data**


Here we can see the final list has been completely sorted. For many applications, the initial data exchange is all that is needed. Thus we will model both the exchange and the final sort separately.

## Performance Considerations

The exchange algorithm can be easily pipelined by operating on small portions of the domain. The first step of the pipeline is to perform the local sort on the generated keys; the second is to perform the all-to-all exchange. This allows the initial sorting and the exchange to be completely overlapped. For the full multi-GPU sorting algorithm, the final local sort cannot begin until all data has arrived, so this portion cannot be pipelined with the other two.

If we assume the original data is randomly distributed among GPUs following a uniform random distribution, then each GPU must communicate `W*N/P` bytes of data to every other GPU, where W is the size of each element in bytes, N is the number of elements per GPU, and P is the number of GPUs.

In a PCIe tree topology, the communication is limited by the bisection bandwidth. Each GPU must communicate half of its data (N/2) across the top link in the system, and there are P/2 devices all trying to communicate across the same PCIe lanes in the same direction.

On an NVLink system, however, the communication can occur in parallel, because there are dedicated links between all pairs of GPUs.

## Performance Projections

Figure 4 shows the speedup of multi-GPU exchange for the 2-GPU scenario, showing performance of 2-GPU-NVLink relative to 2-GPU-PCIe for this algorithm, and for the 4-GPU scenario, showing performance of 4-GPU-NVLink relative to 4-GPU-PCIe.

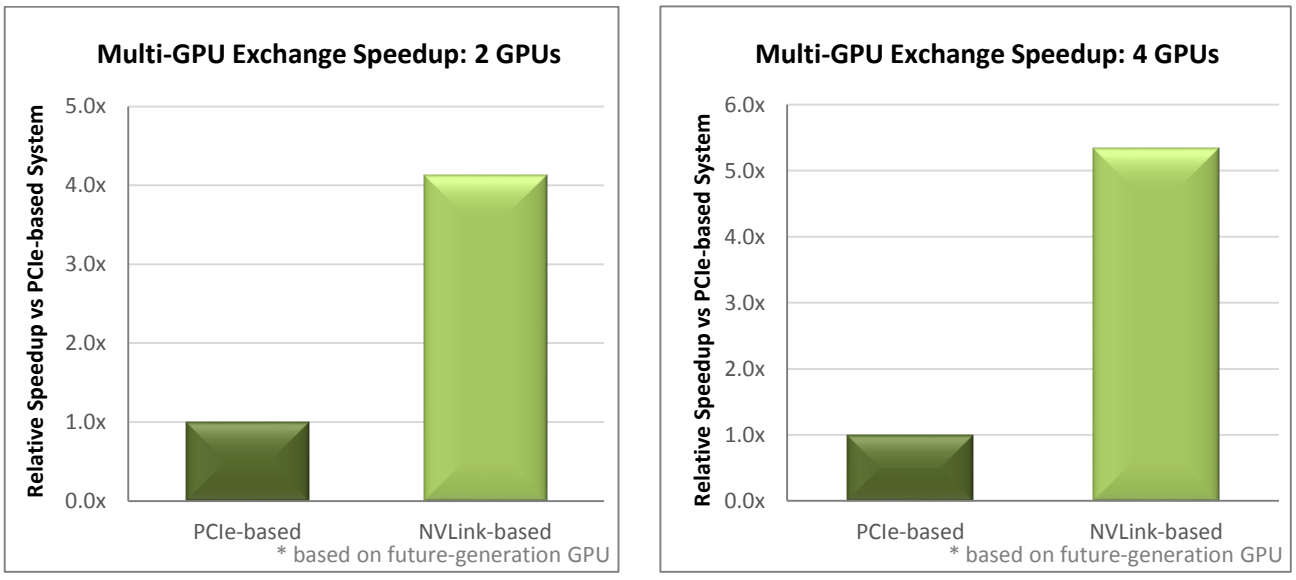



**Figure 4: Multi-GPU exchange performance in 2-GPU and 4-GPU configurations, comparing NVLink-based systems to PCIe-based systems.**

As mentioned above, the performance of this exchange algorithm depends on the available interconnect bandwidth. The NVLink configurations, having both higher bandwidth as well as point-to-point rather than tree-style topology, show much better scaling as the number of GPUs and the relative speed of those GPUs increases.

Figure 5 illustrates the speedup due to NVLink for multi-GPU sort in both the 2-GPU and the 4-GPU scenario.

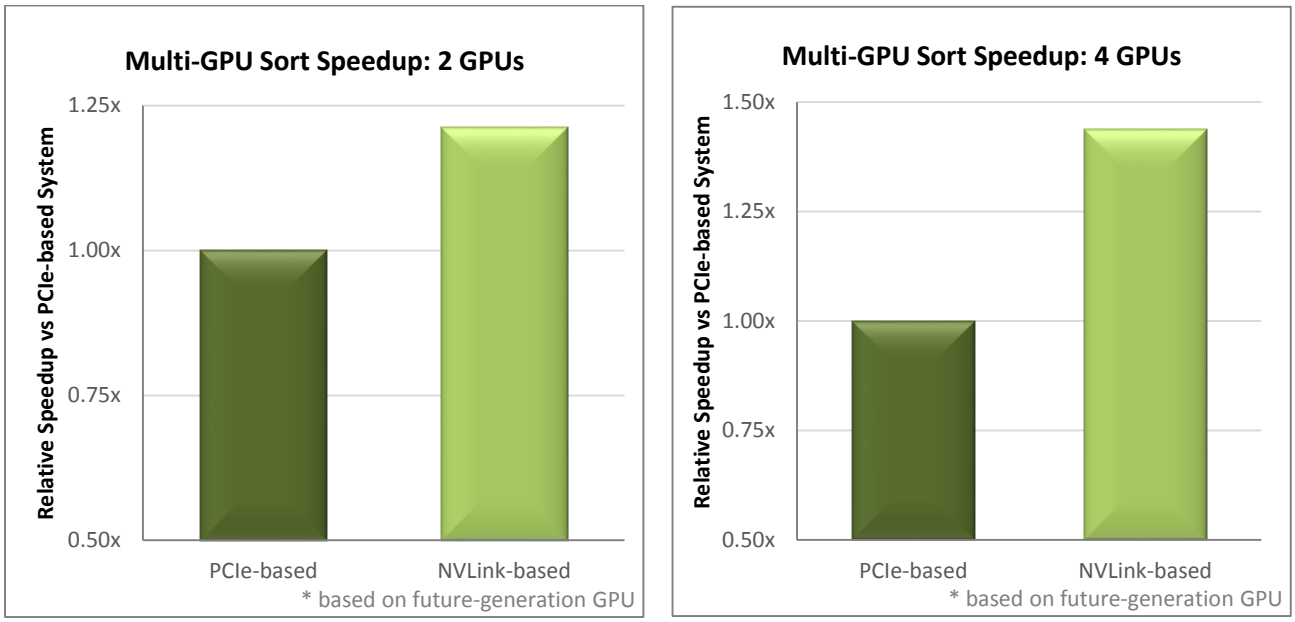



**Figure 5: Multi-GPU sorting performance in 2-GPU and 4-GPU configurations, comparing NVLink-based systems to PCIe-based systems.**

These comparisons show clearly that multi-GPU exchange and sorting both benefit significantly from NVLink. Multi-GPU exchange – and sorting by extension – requires significant communication among GPUs, exceeding what a PCIe tree topology can deliver.

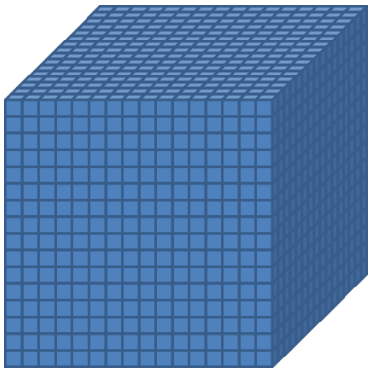
## APPLICATION 2: FAST FOURIER TRANSFORM (FFT)

Fast Fourier Transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT), which converts a sampled function into the frequency domain. FFT is used pervasively throughput computing, with applications in numerous areas such as signal and image processing, partial differential equations, polynomial multiplication, and large integer multiplication.

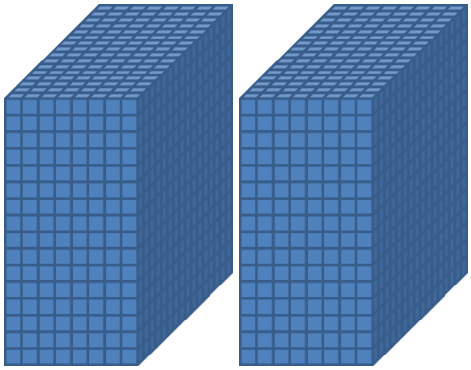
Direct computation of the DFT requires $N^2$ time for an input of size $N$, as it is essentially a matrix-vector multiplication. Through smart data manipulation, however, an FFT algorithm achieves a running time of order $N*\log(N)$. For sufficiently large inputs, this can result in a speedup of several orders of

magnitude. FFT is so fast, actually, that the data movement it requires ultimately cannot keep up with today's computing power, meaning that FFT is bandwidth-bound.

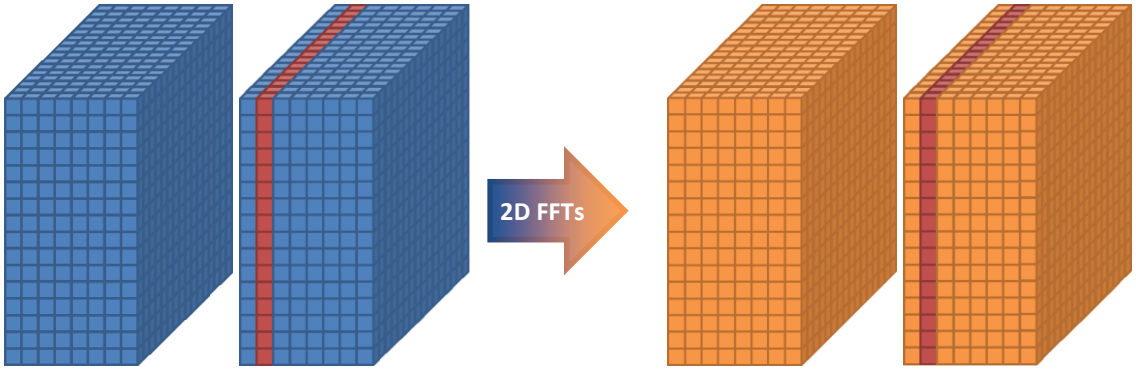
By way of illustration, consider computing the Fourier transform of a three-dimensional cube of data with $k^3$ points (in the figures below, $k=16$).



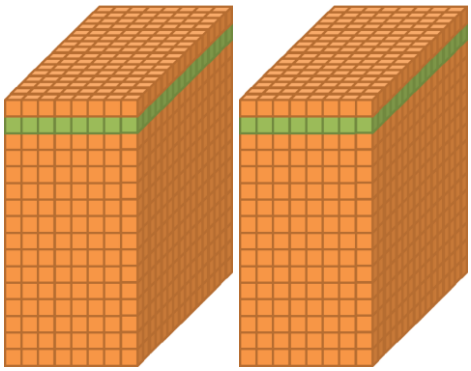
As fast as FFT is algorithmically, large FFTs are still a common application bottleneck. A logical way to accelerate them is to divide the work among multiple processors. To parallelize using two GPUs, for example, we might start by splitting the volume into halves, one per GPU.



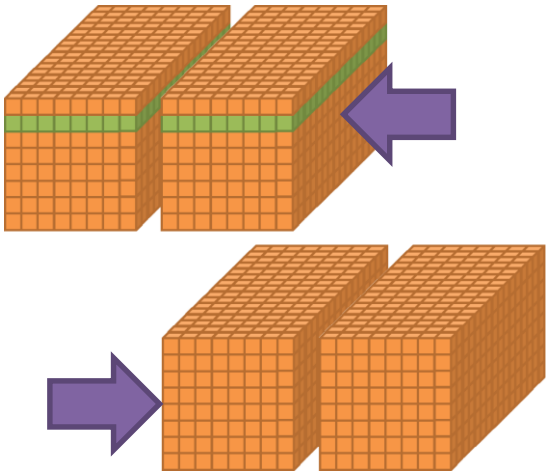
The next step is to compute the 2D FFT of each two-dimensional slice of the data. These can be computed entirely using data local to a single GPU, as illustrated here by the slice shown in red.



To complete the 3D FFT, we must further transform along the other axis (here, the horizontal one). However, any slice in the third dimension requires data that is spread across the two GPUs, as with the green horizontal slice shown here, requiring a data exchange step.



This is a multi-dimensional variant of the exchange used in the sorting example presented above (alternatively, an inter-GPU transposition). Ultimately, half of each GPU's data (one quarter of the original volume) must migrate to the other GPU before the final pass can be completed.



## Performance Projections

Fortunately, each slice of the data at a given stage of the FFT can be operated on independently, so computation and data exchange can be overlapped by pipelining. We found that a very good performance/effort balance is to split each GPU's data into eight slices, transferring data immediately after computation completes for a given slice. This achieves performance within a few percentage points of the theoretical peak for larger sizes.

On a 2-GPU-PCIe system, the performance obtained computing the single precision complex-to-complex 3D FFT of a $256^3$ volume is quite similar to the performance that could be obtained by a single GPU computing the same transform locally (where no communication step is needed); in other words, the cost of communicating over PCIe leads to poor scaling for a workload of this size. Larger problems, say

$512^3$, are able to scale performance on multiple GPUs over PCIe, but ideally we would like smaller problems to scale as well.

With NVLink, our 3D FFT performance model shows that we can achieve near-perfect scaling on a $256^3$ problem, as Figure 6 shows. Such scaling requires more than the 12 GB/s per direction that PCIe can deliver, but it can be achieved with the 32 GB/s or higher per direction aggregate NVLink bandwidth seen in the 2-GPU-NVLink and 4-GPU-NVLink configurations. Moreover, the algorithm sketched above can easily extend to more than two GPUs; near-perfect scaling would also be achieved on 4 GPUs.

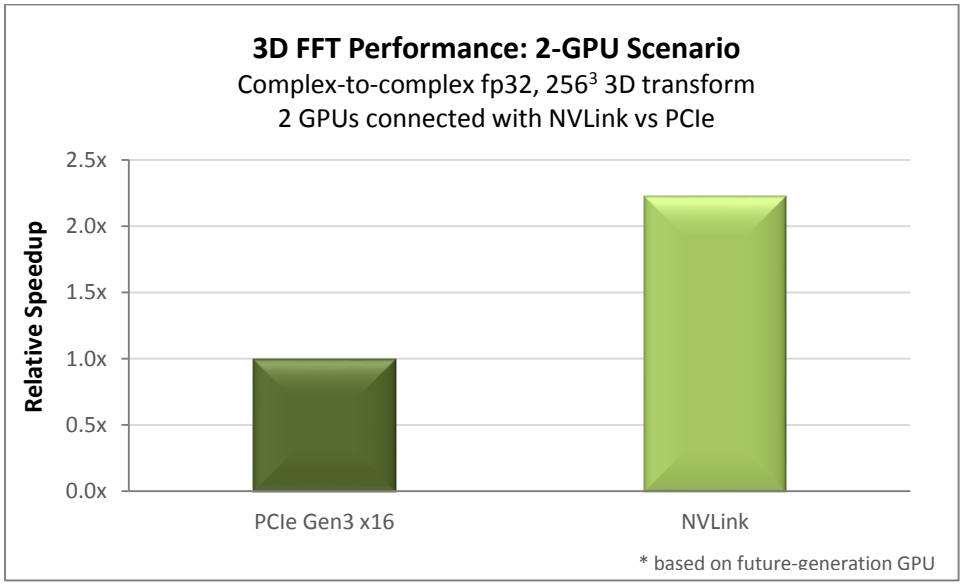



**Figure 6: 3D FFT performance in 2-GPU configurations. NVLink-connected GPUs deliver over 2x speedup.**

NVLink brings one more important benefit to multi-GPU execution of FFTs: it allows smaller input sizes to be executed with lower latency when multiple GPUs are available. With PCIe, only larger inputs such as $512^3$ benefit from multi-GPU scaling. With NVLink, the break-even point is much lower, enabling multi-GPU performance scaling for sizes that are far too small to show a benefit when GPUs are connected to each other via PCIe.

## APPLICATION 3: AMBER- MOLECULAR DYNAMICS (PMEMD)

AMBER is an important molecular dynamics application, which simulates the movements of atoms and molecules interacting in a system using CUDA. Numerous methods exist for such simulations. In this document, we focus on the particle-mesh Ewald's (PME) method, which is popular due to its speed without excessive loss of fidelity: rather than simply truncating the interactions at some cutoff distance, PME allows periodic boundaries to be employed in a simulation with the electrostatic interactions computed out to infinity. When combined with a short-range cutoff that partitions the electrostatic calculation into direct-space and reciprocal-space components, it reduces the cost of the calculation to

order $N$ for the direct-space part (through use of nonbond interaction lists) and to order $N*\log(N)$ for the reciprocal-space sum. This compares with $N^2$ or higher complexity for direct-space-only methods.

AMBER's PMEMD implementation has been heavily optimized and designed for GPU peer-to-peer communication and scales on current hardware to 4 GPUs for a single instance (or to many thousands of GPUs when run within a replica exchange paradigm). Conceptually, the PME approach combines two major algorithms:

- An N-body simulation for non-bonded forces in real space; and
- An FFT in order to calculate long-range interactions in Fourier space.

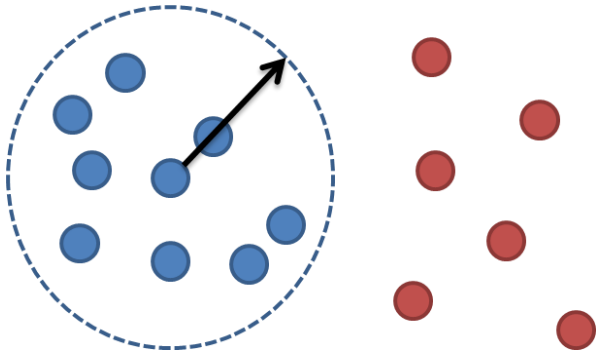


**Figure 7: Illustration of the two cases in PME. In blue are the atoms within a certain radius of a given target atom (typically 8 to 10 angstroms) whose interactions are computed using a pairwise additive N-body simulation; in red are the atoms outside this radius whose interaction are computed using an FFT-based algorithm.**

These two parts of the calculation, illustrated in Figure 7, can be carried out asynchronously on each time step, with the corresponding fractional forces on each atom being summed prior to the integration step. One approach to multi-GPU PME is to compute the FFT for long-range interactions on a single GPU, while dividing the non-bonded forces across the remaining GPUs (possibly also sharing the GPU used for FFT with some of the direct space work), given that scaling the FFT across multiple GPUs is communication-heavy as described in the previous section and that the FFT grid sizes are typically fairly small (48 to 128 in each dimension). However, even while keeping the FFT on a single GPU, the PME approach still requires significant communication to scatter the results of both the direct-space (pairwise-additive) and the reciprocal-space (FFT) steps to the other GPUs, which is particularly problematic for the tree topologies used with PCIe.

## Performance Considerations

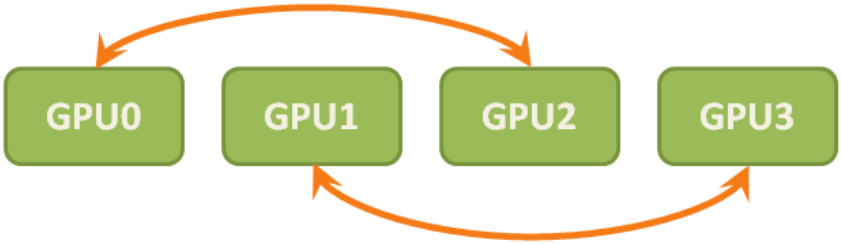
As the overall performance of this application is difficult to model analytically, we profiled a typical time-step for a PME simulation in the microcanonical ensemble (also known as NVE) benchmarks, splitting the measured time into the following categories:

- N-body (FLOPS-limited)
- FFT (memory bandwidth-limited)
- Device-to-device communication (interconnect bandwidth-limited)
- All other kernels (which we can assume are primarily memory bandwidth-limited)
- Synchronization overhead

These measurements of relative percent time spent in each of these operations allows us to model the performance when any combination of computation, memory bandwidth, and/or interconnect bandwidth is improved. For example, comparing relative performance on 2-GPU-PCIe and 2-GPU-NVLink is straightforward for this application; we simply apply the ratio of the bandwidths to the fraction of application time spent on device-to-device transfers.

4-GPU-PCIe to 4-GPU-NVLink comparison at first appears somewhat more complex, since the communication step scatters data among all pairs of GPUs, and effective all-pairs bandwidths in this situation are asymmetric. However, consider the exchange in two stages, with the first communicating between GPUs 0 and 1 and between GPUs 2 and 3.


Now GPUs 0 and 1 both hold the sum of their data, and likewise for GPUs 2 and 3. We now sum across GPU 0 and 2 and across GPUs 1 and 3.




All four GPUs now hold the same sum. In 4-GPU-PCIe, the first of these steps runs at full PCIe bandwidth, while the second step has to share the single link at the top of the PCIe tree between the two switches. In 4-GPU-NVLink, the first step takes advantage of two NVLink connections between each pair of GPUs, while the second step pairs GPUs connected by only a single NVLink. In both cases, the effective bandwidth for the second step is halved due to the nature of the topology, so we can apply a uniform speedup factor across both stages to determine the expected speedup due to NVLink.

## Performance Projections

As explained above, relative performance is projected by simply applying the ratio of the bandwidths to the fraction of application time spent on device-to-device transfers. Figure 8 illustrates the results of this comparison for the 4-GPU scenario across a range of different test cases.

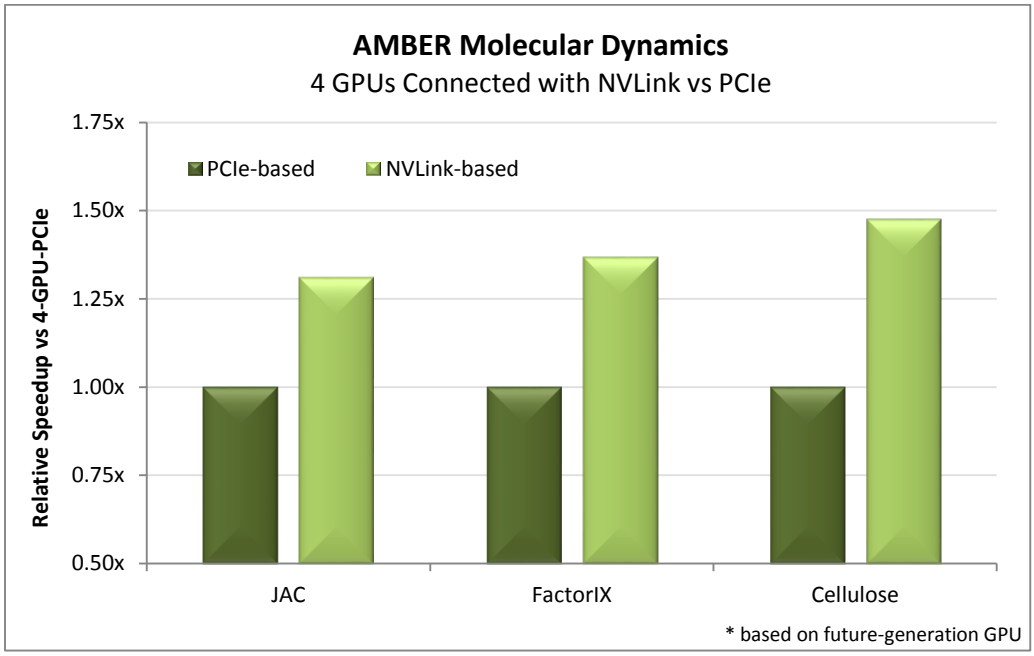



**Figure 8: Up to 50% performance boost for various AMBER models comparing 4-GPU configurations.**

For all of these benchmarks, NVLink provides a clear advantage over PCIe, yielding a dramatic 30-50% improvement for four GPUs, just by replacing the interconnection among them.

# APPLICATION 4: ANSYS FLUENT- COMPUTATIONAL FLUID DYNAMICS

ANSYS Fluent is a popular Computational Fluid Dynamics (CFD) package, representative of a wide class of CFD applications. For our CFD study, we chose Fluent's truck_14m benchmark, which represents a typical external aerodynamics problem. This case has around 14 million cells of mixed type and uses an implicit coupled solver with the *k-e* turbulence model.

The main computational part for this benchmark is a coupled linear solver that uses the *Flexible Generalized Minimal Residual method* (FGMRES) preconditioned by an algebraic multigrid (AMG). In Fluent 14.5 and later versions, the linear solver is GPU-accelerated using the NVIDIA AmgX library. We have captured profiles of typical cases in AmgX as used in truck_14m, yielding a model of time spent in various operations that allows us to investigate the benefits of NVLink for CFD applications.

## Performance Projections

The execution time of the AMG timestep can be broken down into two main phases: the setup phase and the solve phase. During the setup phase, the multigrid hierarchy is created, while during the solve phase, FGMRES iterations are performed to update the solution vector until a desired residual convergence is achieved.

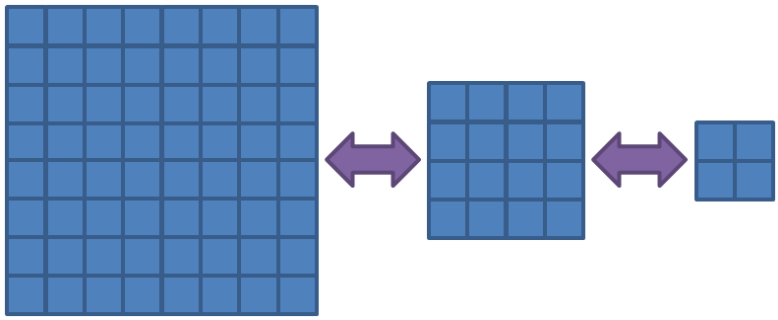


Figure 9: Illustration of the multiple resolutions of the multigrid hierarchy constructed during the setup phase.

AMG's performance depends mainly on the efficiency of BLAS Level 1 operations, sparse matrix-vector multiplication (SpMV), and point-to-point communication. The most significant factor for performance is memory bandwidth, hence AMG's performance is more sensitive to hardware capabilities (high memory bandwidth is required) than to software optimization. With increased memory bandwidth in future GPUs, we can expect significant improvement in the per-GPU performance of these kernels, especially in the solve phase.

AmgX transfers data between the CPU and GPU at several points during its execution using both synchronous and asynchronous transfer mechanisms. For purposes of this study, the transfer that is significant is the (synchronous) copy of MPI buffers before and after each MPI exchange; the message sizes vary from a few bytes to a few dozen megabytes. We must complete these synchronous transfers as rapidly as possible, because the GPU is basically idle during the copies, and Amdahl's Law applies. With NVLink, these transfers will accelerate in direct proportion to the relative interconnect bandwidths, yielding a clear benefit for Fluent, as shown in Figure 10.

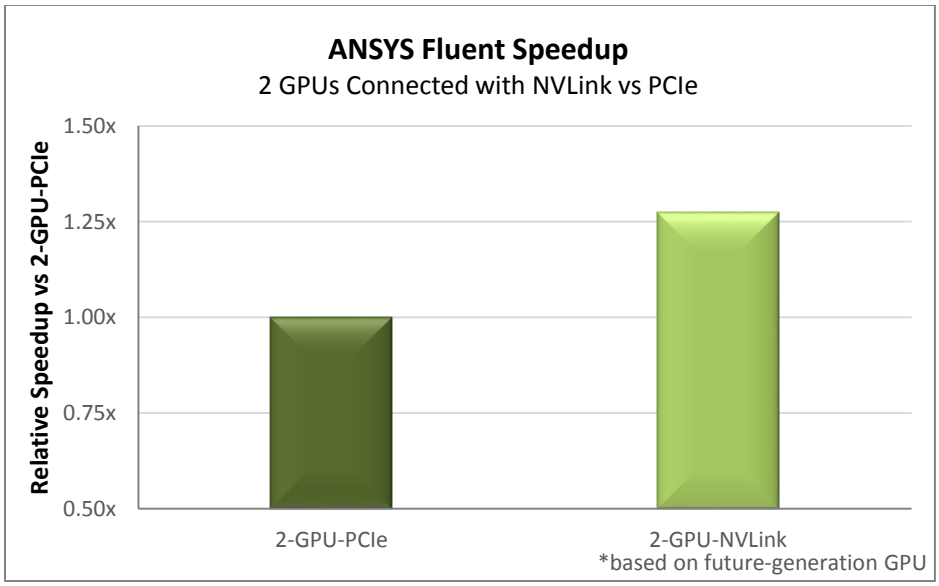



**Figure 10: ANSYS Fluent gains over 25% speedup with NVLink comparing 2-GPU configurations.**

# APPLICATION 5: LATTICE QUANTUM CHROMODYNAMICS (LQCD)

QUDA "*Q*CD on C*UDA*" is a library that provides GPU acceleration for legacy lattice quantum chromodynamics (LQCD) applications, including Chroma, MILC, and others. For LQCD simulations, the dominant computational cost is in the iterative solution of a sparse linear system, where the sparse matrix is a radius-one finite-difference stencil (the Wilson `dslash` operator) acting on a four-dimensional hypercubic lattice (grid).

## Multi-GPU Implementation

In deploying these computations on multiple GPUs, a simple domain-decomposition strategy is utilized, where each GPU is responsible for updating a distinct contiguous sub-volume of the overall problem. To update the sites at the sub-domain boundaries, a boundary exchange must occur between logically neighboring GPUs. Figure 11 illustrates this sort of domain decomposition in two dimensions.

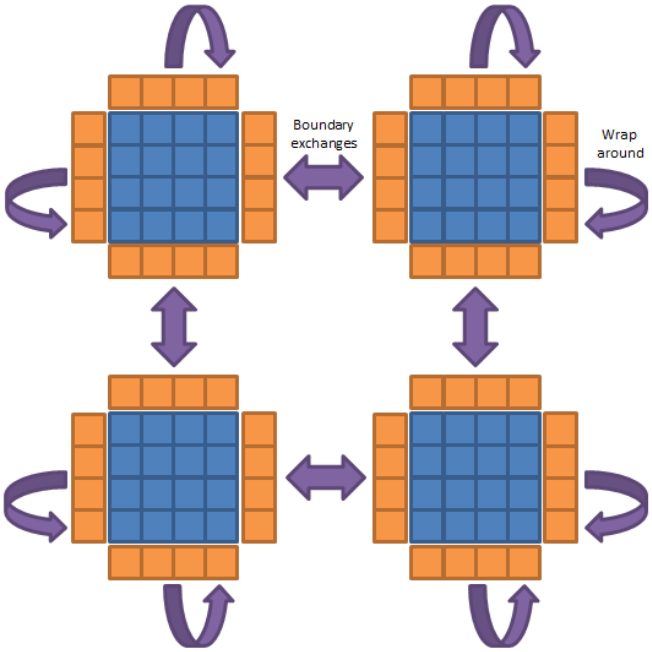



Figure 11: Two-dimensional partitioning scheme employed by QUDA on 4 GPUs

## Performance Projection

The `dslash` kernel is split into two parts: one for update of the interior (non-boundary) regions entirely local to a given GPU and one for update of the boundary regions. In order for the performance to scale with the number of GPUs, the application of `dslash` to the interior regions should be overlapped with the exchange of boundary data among adjacent GPUs. QUDA uses peer-to-peer memory transfers among pairs of adjacent GPUs to allow this concurrency between computation and inter-GPU transfers.

Given the uniform nature of this problem, we can model the expected performance of such an approach using relatively few parameters:

- Memory bandwidth: the performance of the local `dslash` kernels are largely memory bandwidth-limited.
- Interconnect bandwidth: this determines how fast the boundary exchange can occur between neighboring GPUs.
- The local sub-volume size per GPU: the surface-to-volume ratio dictates the communication-to-computation ratio.

Comparing the relative performance of the Wilson `dslash` computation on 4-GPU-NVLink and 4-GPU-PCIe configurations, Figure 12 shows that the NVLink configuration outperforms PCIe significantly. This is driven by the increase in computational performance of the future-generation GPU as enabled by NVLink.

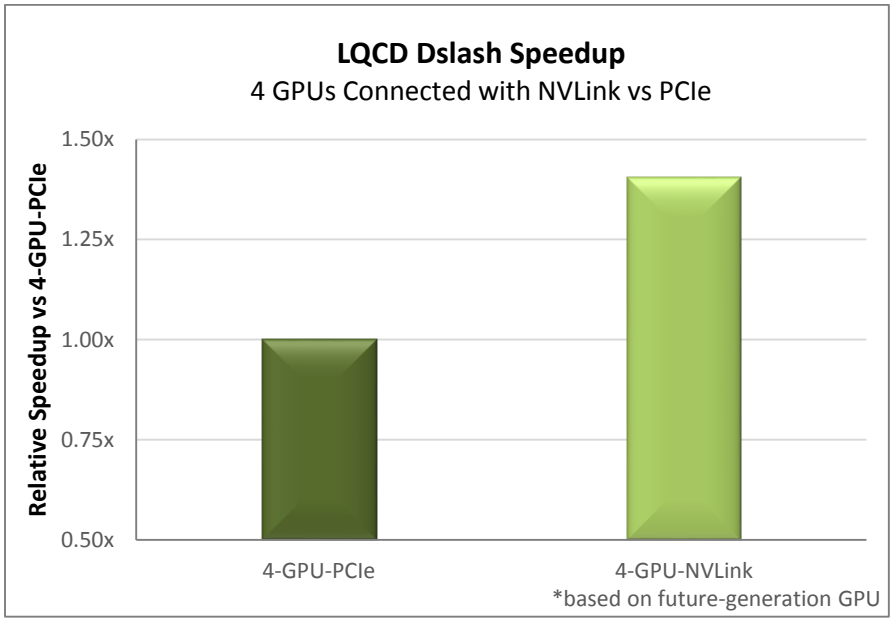



Figure 12: LQCD dslash benchmark in 4-GPU configurations.

## SUMMARY

The U.S. Department of Energy's announcement of the *Summit* and *Sierra* supercomputers puts it in the position to lead the world into the exascale era. **Many technological innovations play a role in the architecture of these pre-exascale systems, but NVLink is one of the most important ingredients to pull the system nodes together.** The NVLink high-speed interconnect not only provides a fast path for CPUs to communicate with GPUs, but it allows multiple GPUs to share data with unrivaled performance.

In this study, we analyzed several algorithms and applications familiar to the industry. NVLink is projected to deliver significant performance boost – up to 2x in many applications – simply by replacing the PCIe interconnect for communication among peer GPUs. This clearly illustrates the growing challenge NVLink addresses: as the GPU computation rate grows, GPU interconnect speeds must scale up accordingly in order to see the full benefit of the faster GPU. Similarly, as both CPUs and GPUs improve in next-generation systems, the speed of the CPU-to-GPU interconnection must continue to grow beyond that of PCIe Gen3.

NVLink delivers the bandwidth to keep up with the ever-increasing performance of GPUs for several generations of GPU architectures to come.

**Notice**

ALL INFORMATION PROVIDED IN THIS WHITE PAPER, INCLUDING COMMENTARY, OPINION, NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

**Trademarks**

NVIDIA, the NVIDIA logo, CUDA, Pascal, and NVLink are trademarks or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

**Copyright**