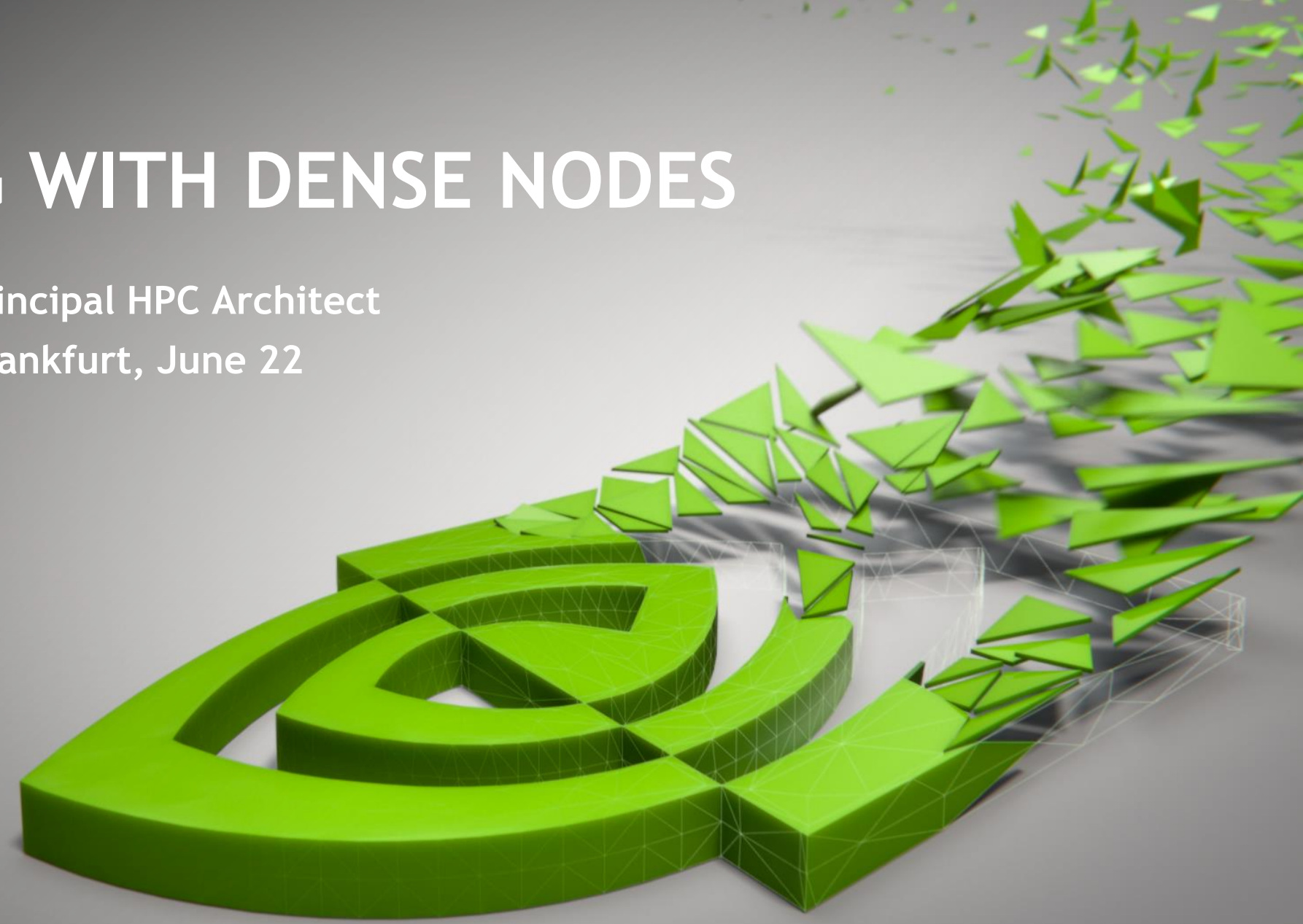
SCALING WITH DENSE NODES

CJ Newburn, Principal HPC Architect

ExaComm17, Frankfurt, June 22

NVIDIA



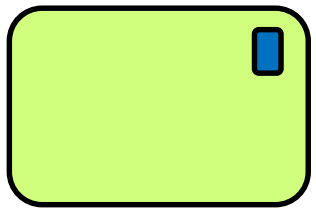
OUTLINE

- Scaling
- System architecture
- CPU initiated
  - GPU Direct
  - NCCL
- CPU-initiated MPI vs. GPU-initiated NVSHMEM
- GPU initiated
  - NVSHMEM
  - GPU verbs for IB

CHOOSING A SCALED PLATFORM:
DENSE NODES ARE A BUILDING BLOCK FOR EFFICIENCY

TOWARD A SCALED SYSTEM

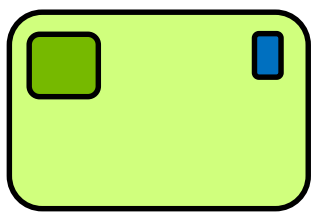
A basic CPU



TOWARD A SCALED SYSTEM

Lots of threads

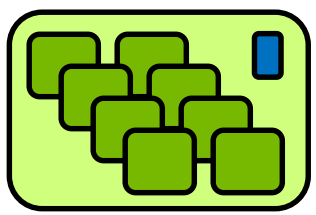
- Threads



TOWARD A SCALED SYSTEM

LOTS of threads

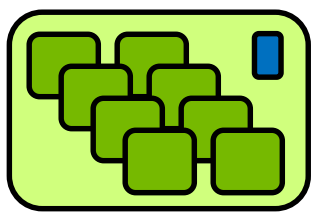
- Denser stronger nodes
  - Threads
  - Efficiency
  - Density



TOWARD A SCALED SYSTEM

LOTS of threads

- Denser stronger nodes
  - Threads
  - Efficiency
  - Density

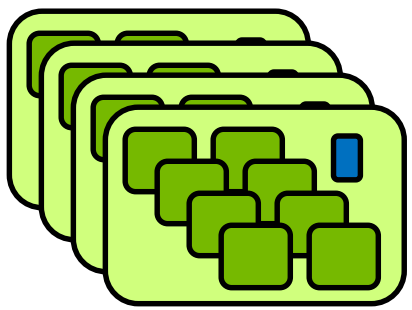


- Strong connections
  - Push out IB “sharp edge” with a memory-bus link (NVLink)
  - Direct load/store/atomics

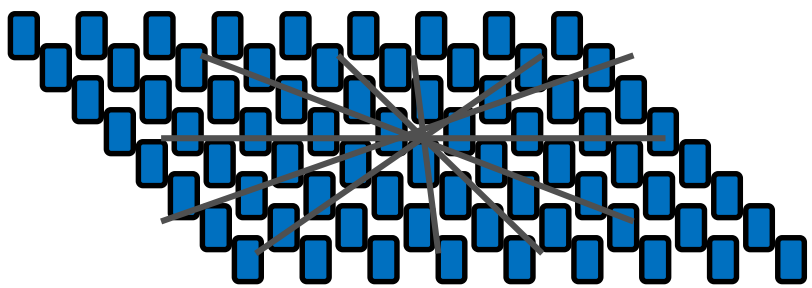
CHOOSING A SCALED PLATFORM

Better strong scaling, easier to manage

- Denser stronger nodes
  - Threads
  - Efficiency
  - Density



vs.



- Strong connections
  - Push out IB “sharp edge” with a memory-bus link (NVLink)
  - Direct load/store/atomics

SYSTEM ARCHITECTURE
DENSE NODES AND SUB-CLUSTERS

NVLink: THE DENSE NODE ENABLER

NVLink softens the edges: makes more memory look fast and close

- Addressing among GPUs
  - Loads and stores “just work” and enjoy coalescing benefits, **same memory model as GPU**
  - **Atomics** “just work” across devices, synchronization is based on atomics
- Communications among GPUs
  - Remote memory accesses enjoy the same **latency tolerance** as local ones
  - **Direct loads and stores** avoid staging, extra buffers, copying; easier to program
  - Full access by default, but can **manage locality** (affinity, replication) as desired

CLUSTERING SUBSETS OF GPUS

Problem size and communication patterns drive configuration

- **Deep learning** wants all-reduce, then all-gather/scatter
  - All-all algorithms were avoided because they don't fit current platforms well
  - Model/hybrid trending toward point-point halo exchange vs. all reduce
  - So a cross-node connection per GPU seems sufficient
- **HPC** sometimes wants all-all → many more cross-node connections

- 1 process or rank may want to map to {1,2,4,8,16} GPUs
- Close communication may happen in 1 or many ranks/processes
- May want strongly-connected sub-clusters (e.g. 16-GPU HGX)

OUTLINE

- Scaling
- System architecture
- **CPU initiated**
  - **GPU Direct**
  - **NCCL**
- **CPU-initiated MPI vs. GPU-initiated NVSHMEM**
- GPU initiated
  - NVSHMEM
  - GPU verbs for IB

GPUDIRECT ASYNC

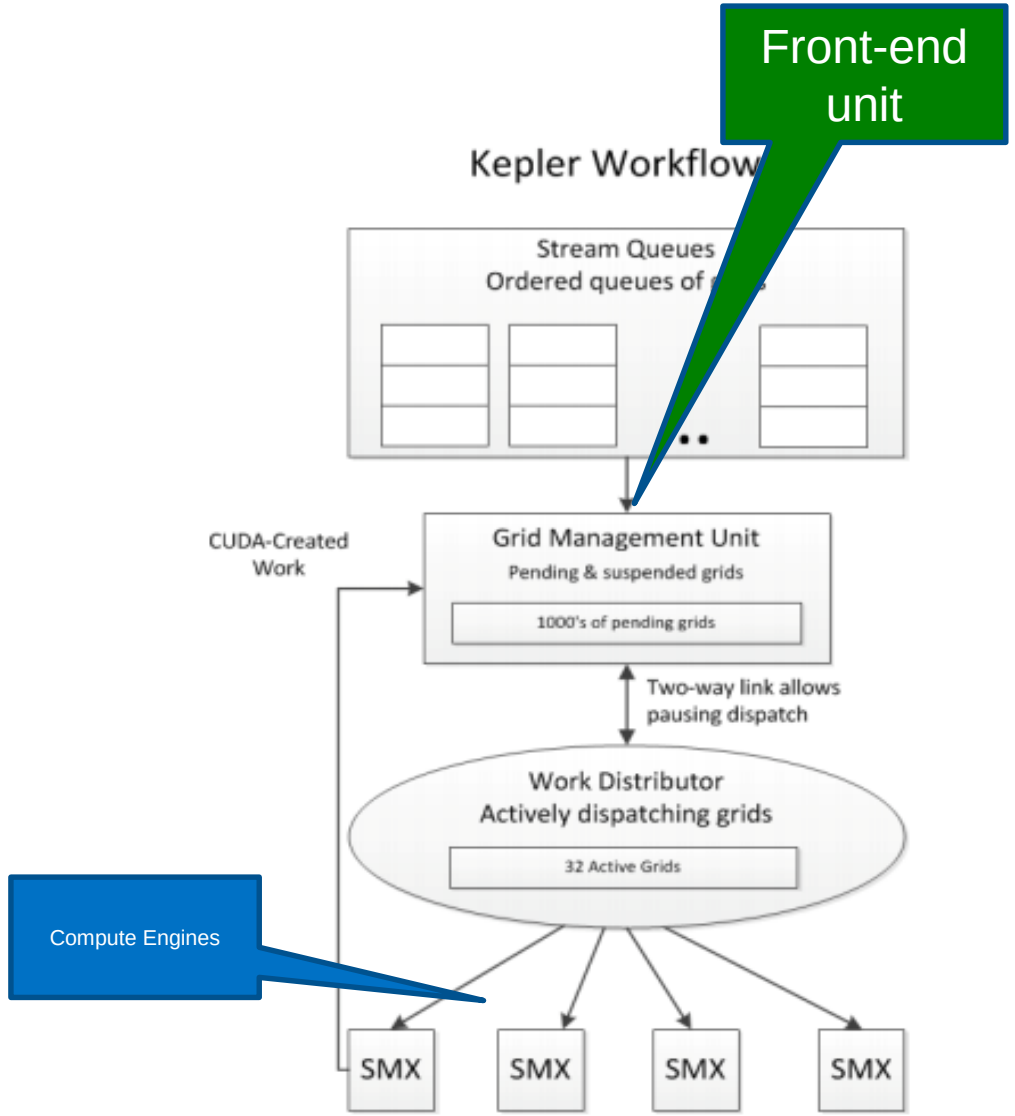
leverage GPU front-end unit

Communications prepared by CPU

- hardly parallelizable, branch intensive
- GPU orchestrates flow

Run by GPU front-end unit

- Same one scheduling GPU work
- Now also scheduling network communications



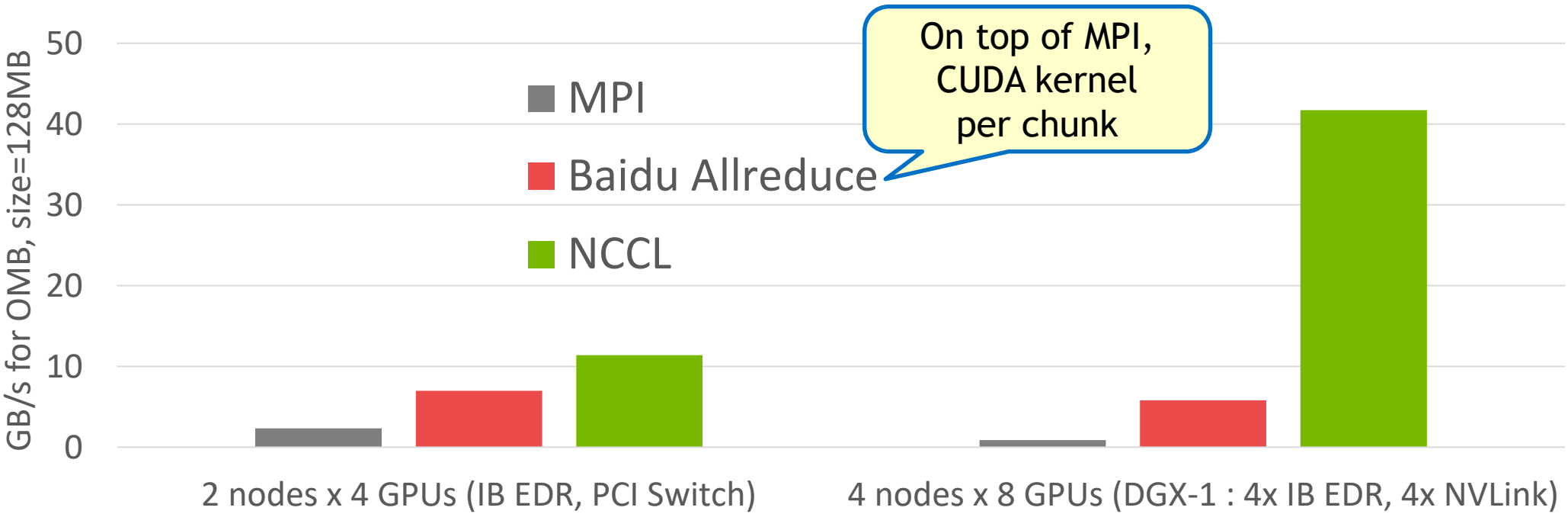
NCCL: NVIDIA COLLECTIVE COMMUNICATIONS LIB

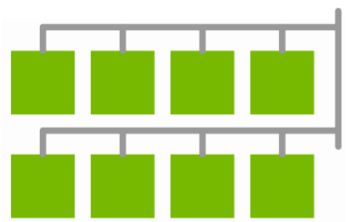
Best-available communications for deep learning

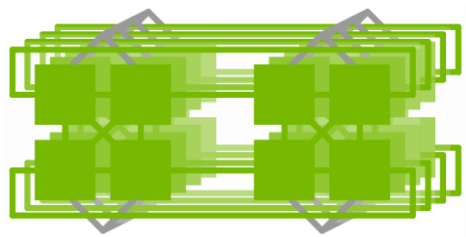
- Optimized collective communication library between CUDA devices.
- Easy to integrate into any DL framework, as well as traditional HPC apps using MPI.
- Runs on the GPU using asynchronous CUDA kernels, for faster access to GPU memory, parallel reductions, NVLink usage.
- Similar to MPI collectives, but has a CUDA stream parameter, operates on CUDA ptrs.
  - → Enables use of multiple threads per MPI rank, multiple GPUs per thread.
  - Minimizes GPU threads to permit other computation to progress simultaneously.
- NCCL 2.0 adds support for inter-node communication using Sockets or IB verbs.

PERFORMANCE

Inter-node performance: all-reduce bandwidth, coarse-grained







MVAPICH2 GDR 2.2

GPU-CENTRIC COMMUNICATION

GPU gains autonomy

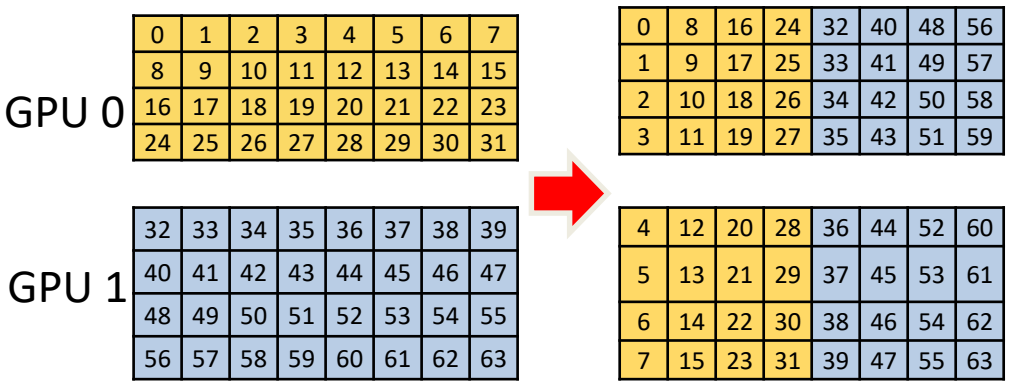
- GPU-initiated communication
  - Removes overheads caused by reliance on CPU
    - Kernel launches
    - Synchronization
  - Moves communication into parallel regions
    - Better utilization of GPUs
    - Better overlap
    - Smoothened traffic
- Prototyped with NVSHMEM
  - Among PCIe- and NVLink-connected GPUs
  - Peer memory mappings
  - Direct load/store accesses

CPU-INITIATED MPI VS. GPU-INITIATED NVSHMEM

Different approaches for different needs

- MPI
  - Buffered, bursty
  - Write to local, copy to MPI, copy to other rank, copy to user code
  - Communication is separate from computation
  - Fine to do from CPU at the end of a phase of computation
- SHMEM
  - Fine-grained, smooth
  - Direct load/store, no extra allocs or copies
  - Communication is naturally integrated with computation
  - Better to do from GPU while you're doing the work

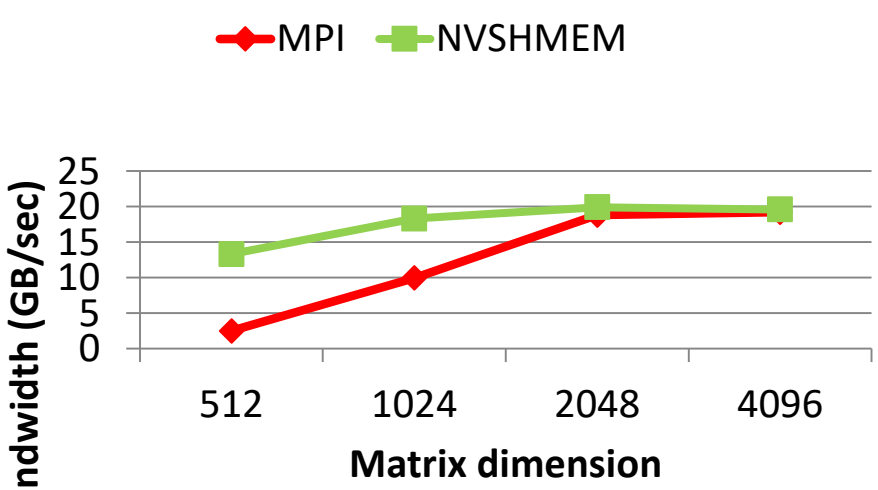
MULTI-GPU TRANSPOSE



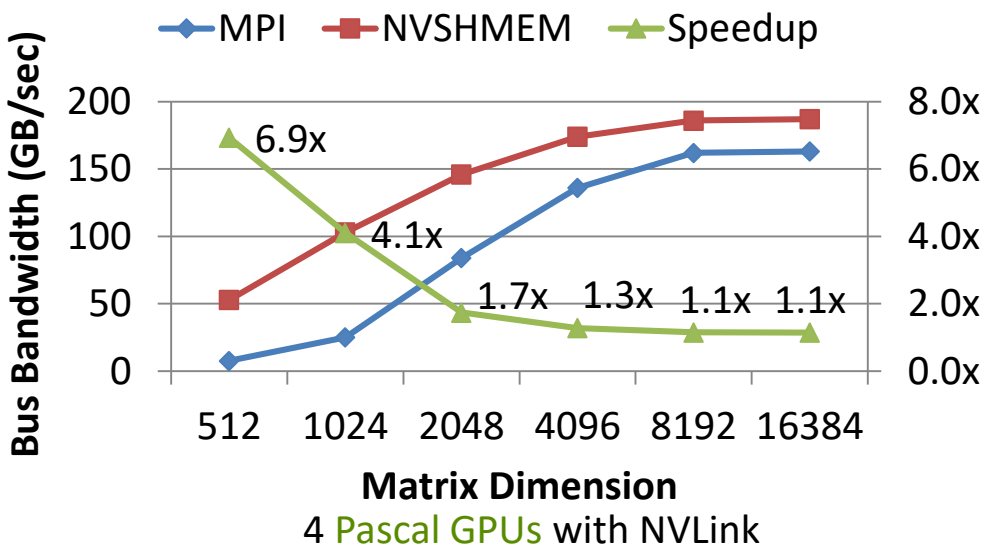
Similarly, speedup on HPGMG, Caffe can approach 2+x

MPI version requires 3X higher memory bandwidth than NVSHMEM version

NVSHMEM significantly reduces code complexity and delivers better performance



4 K40 GPUs with PCIe Gen3 (same root)
Both MPI and nvSHMEM bandwidth Limited



4 Pascal GPUs with NVLink

MPI more limited by memory bandwidth

Poll

WOULD AN MPI/SHMEM LAB BE HELPFUL?

Demonstrating performance while building developer confidence

- With OpenACC, same code run on CPU or GPU; create a CPU-enabled version of nvSHMEM
- For CPU data
  - MPI buffered outside parallel
  - SHMEM buffered outside parallel: *similar*
  - MPI buffered inside parallel, each thread works on its own buffer: *parallel accesses*
  - SHMEM buffered inside parallel, each thread works on its own buffer: *< overhead, > parallelism*
  - SHMEM ld/st inside parallel: *remove buffer overheads, smooth accesses*
- For GPU data
  - MPI buffered outside parallel
  - nvSHMEM buffered inside parallel, each thread works on its own buffer: *< overhead, > parallel*
  - nvSHMEM ld/st inside parallel: *remove buffer overheads, smooth accesses*
- We are looking at creating such a lab. Interested?

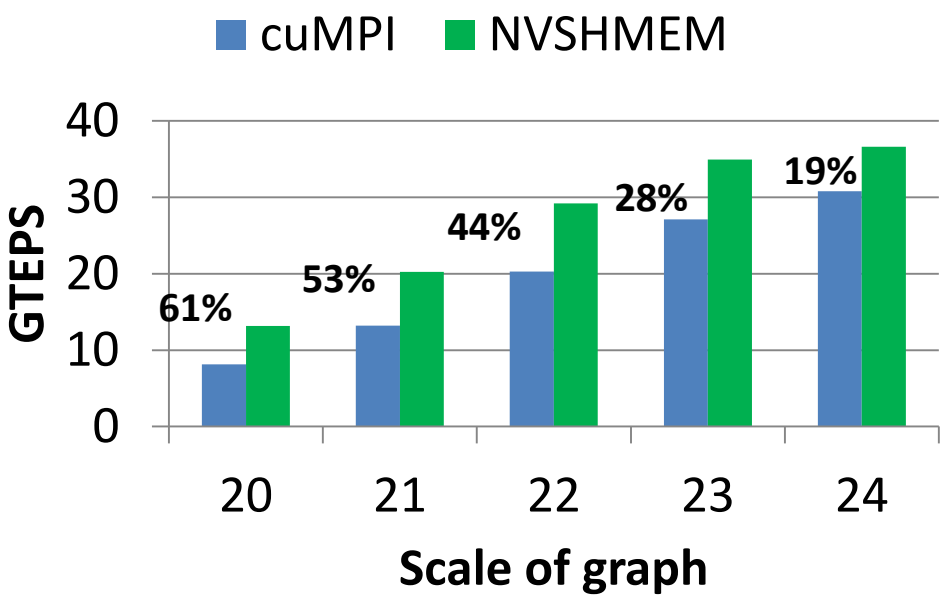
OUTLINE

- Scaling
- System architecture
- CPU initiated
  - GPU Direct
  - NCCL
- CPU-initiated MPI vs. GPU-initiated NVSHMEM
- **GPU initiated**
  - **NVSHMEM**
  - **GPU verbs for IB**

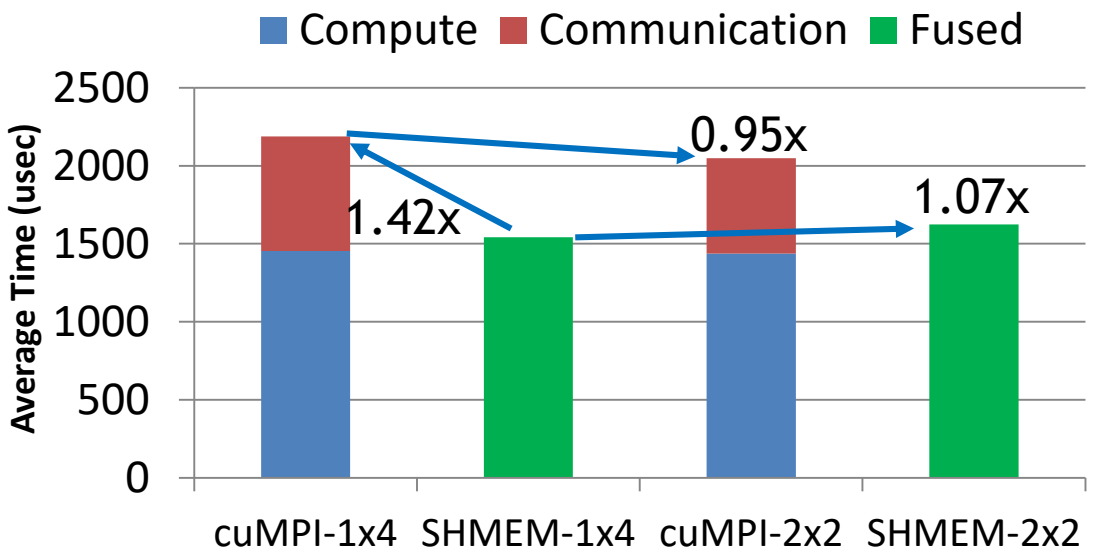
BREADTH FIRST SEARCH

MPI has no compute/communication overlap; NVSHMEM fuses them

8 P100 GPUs in DGX-1



Scale of 20, 4 P100 GPUs



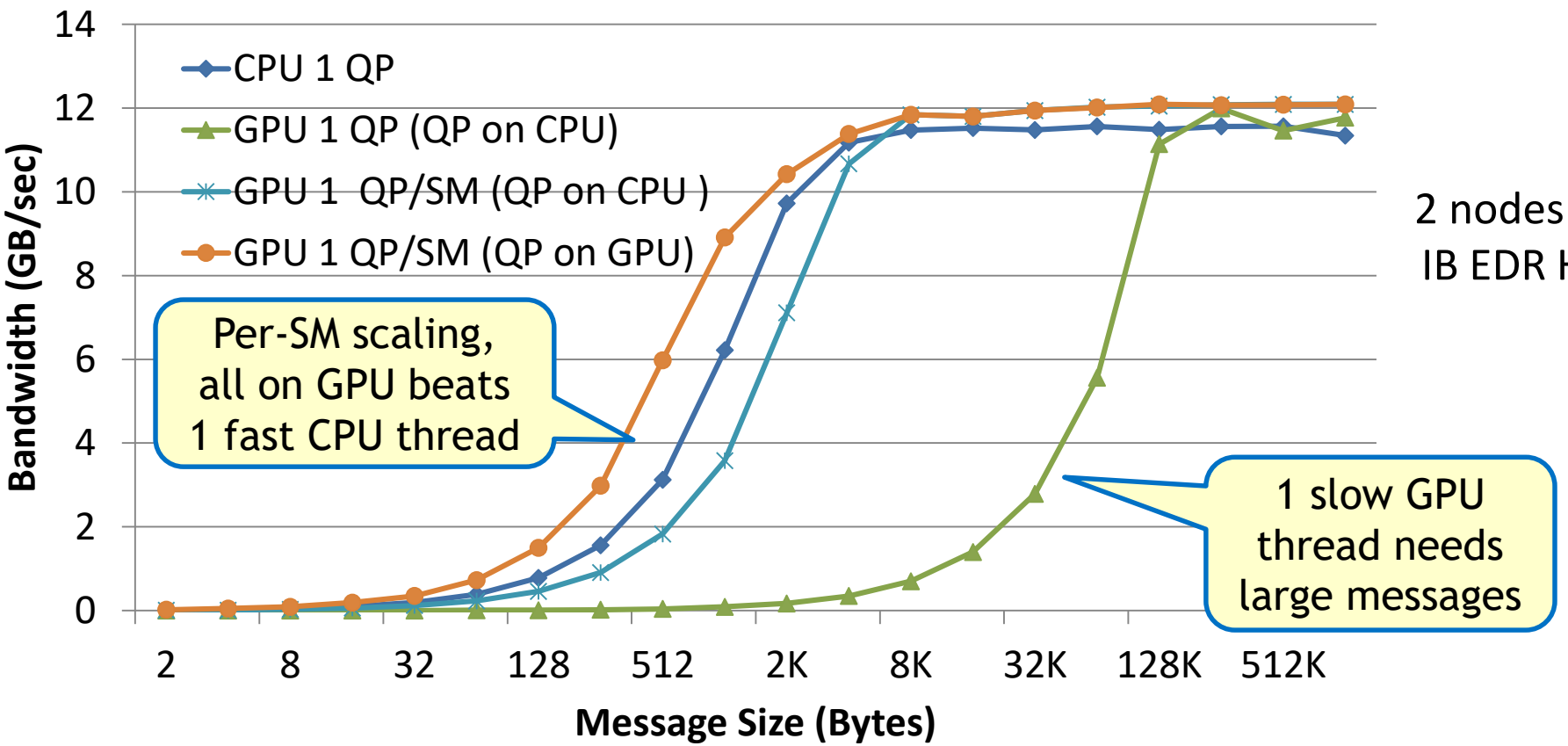
- Greater impact of NVSHMEM's atomics vs. MPI's packed transfers with congestion at large sizes
- 2D MPI decomposition already more efficient, gain from fusing is relatively smaller

GPU-VERBS FOR IB

- Developing a prototype IB-verbs for GPU
  - Extension to GPUDirect RDMA (host-bypass in data path) and Async (host-enqueued GPU-triggered communication)
  - Truly GPU-initiated communication: GPU prepared, GPU initiated
  - Funded in part by ORNL, sub-contract #4000145249
- GPU-centric design of verbs
  - Slow cores and massive concurrency
  - Inverted memory hierarchy (register and shared memory usage for performance)
  - Implicit WARP-level coalescing of accesses
  - Relaxed Memory Model
- Can this allow NVSHMEM model to extend beyond NVLink-connected clusters of GPUs?

GPU-GPU IB BANDWIDTH TEST

Preliminary Results

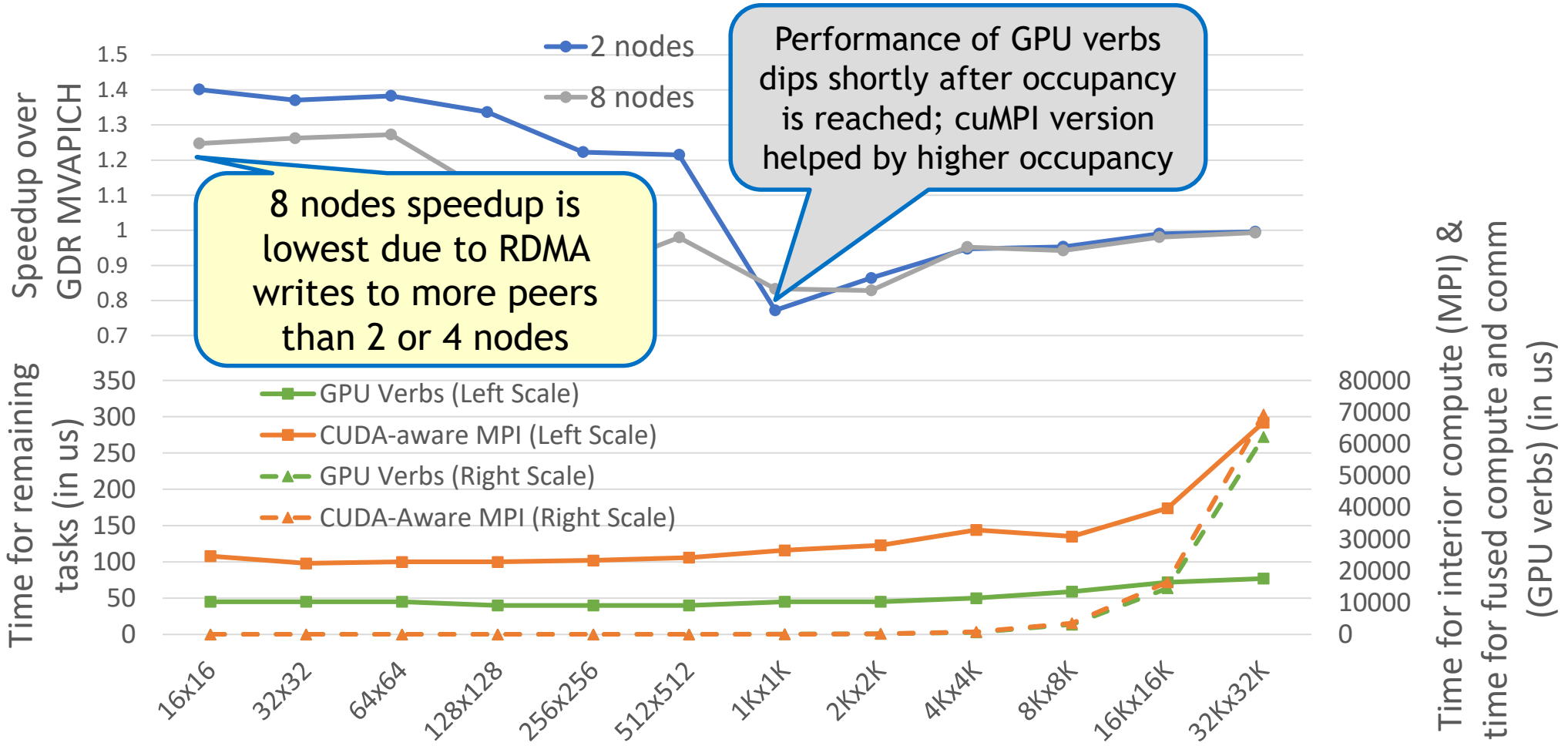


2 nodes - 1 Pascal GPU + 1 IB EDR HCA on each node

2D STENCIL MINI APP IMPLEMENTED IN GPU VERBS

Preliminary results for GPU Verbs vs. cuMPI

8 nodes - 1 Pascal GPU +
1 IB FDR HCA on each node



WIP to implement near constant time RDMA write Independent of number of peers using WQE templating

WIP to debug kernel performance of GPU verbs and cuMPI around respective occupancy

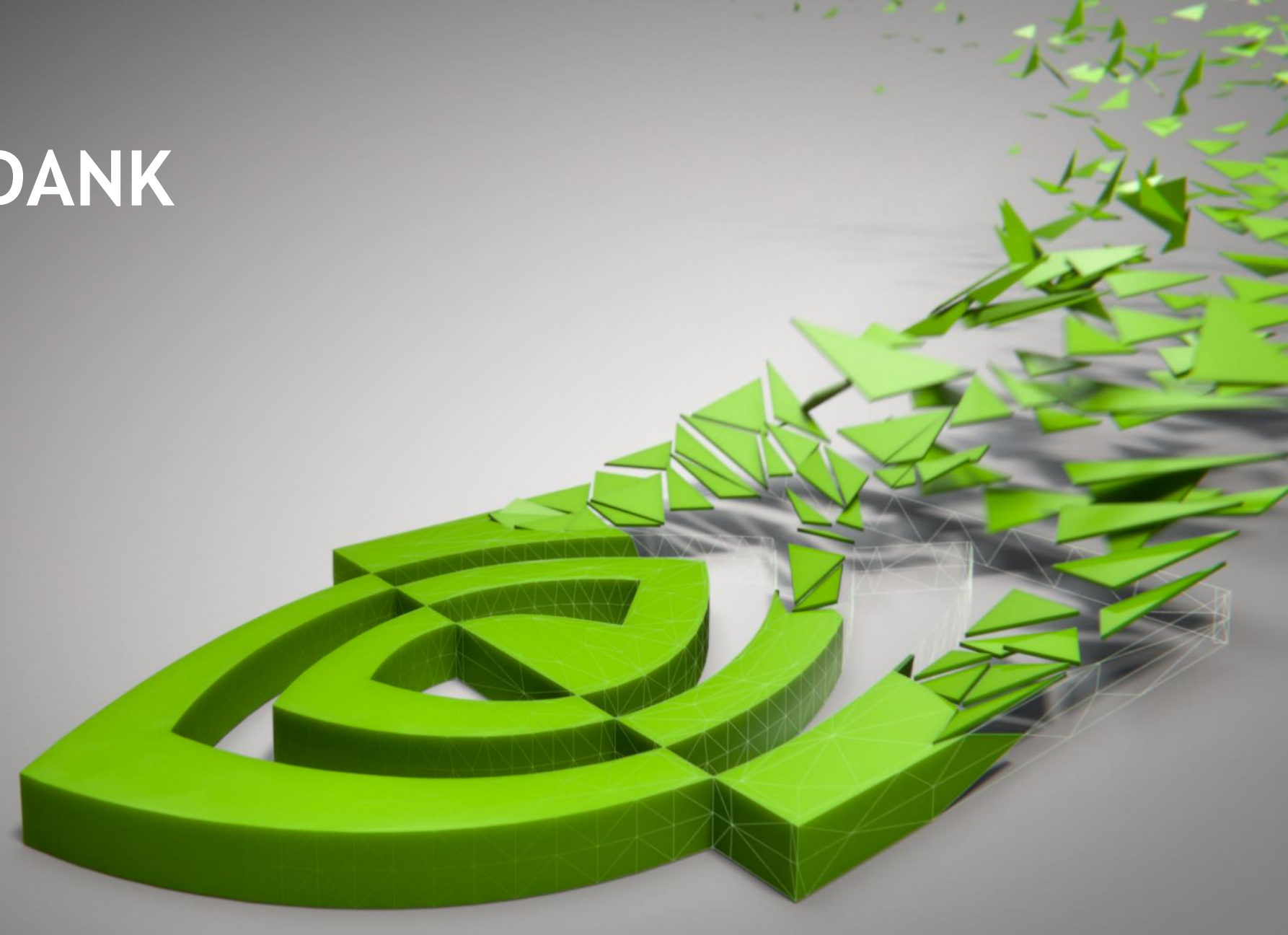
TAKE-AWAYS

Pushing communication down to parallel computing resources

- Dense nodes and highly-connected sub-clusters create more opportunity for efficiency
- SHMEM shows benefit from fewer copies, overlapped communication and computation
- CPU-initiated communication can be done in parallel, even triggered by GPU
- GPU-initiated communication avoids overheads, benefits from more parallelism
- NCCL and GPU verbs can span nodes

VIELEN DANK

NVIDIA.



COMPARISON WITH ALTERNATIVES

NVSHMEM and GPU verbs are easier to use and/or more performant

- CUDA IPC is lower level than what most users want
  - This is what NVSHMEM uses for GPU peers. It could be used directly, and users could set up peer-peer access, shared memory allocation, and cross-rank communication themselves
- MPI Windows pays a performance price for its generality
  - Superset of NVSHMEM functionality
    - Create or allocate shared window, get address to enable direct load/store
  - Sources of MPI performance overheads, unless restricted
    - Different sizes on different processes → same size enables base + offset addressing
    - Window object gets translated to base address translation → restrict for use on GPU only
    - Data type can be arbitrary vs. baked into function name → restrict types, focus on HW support