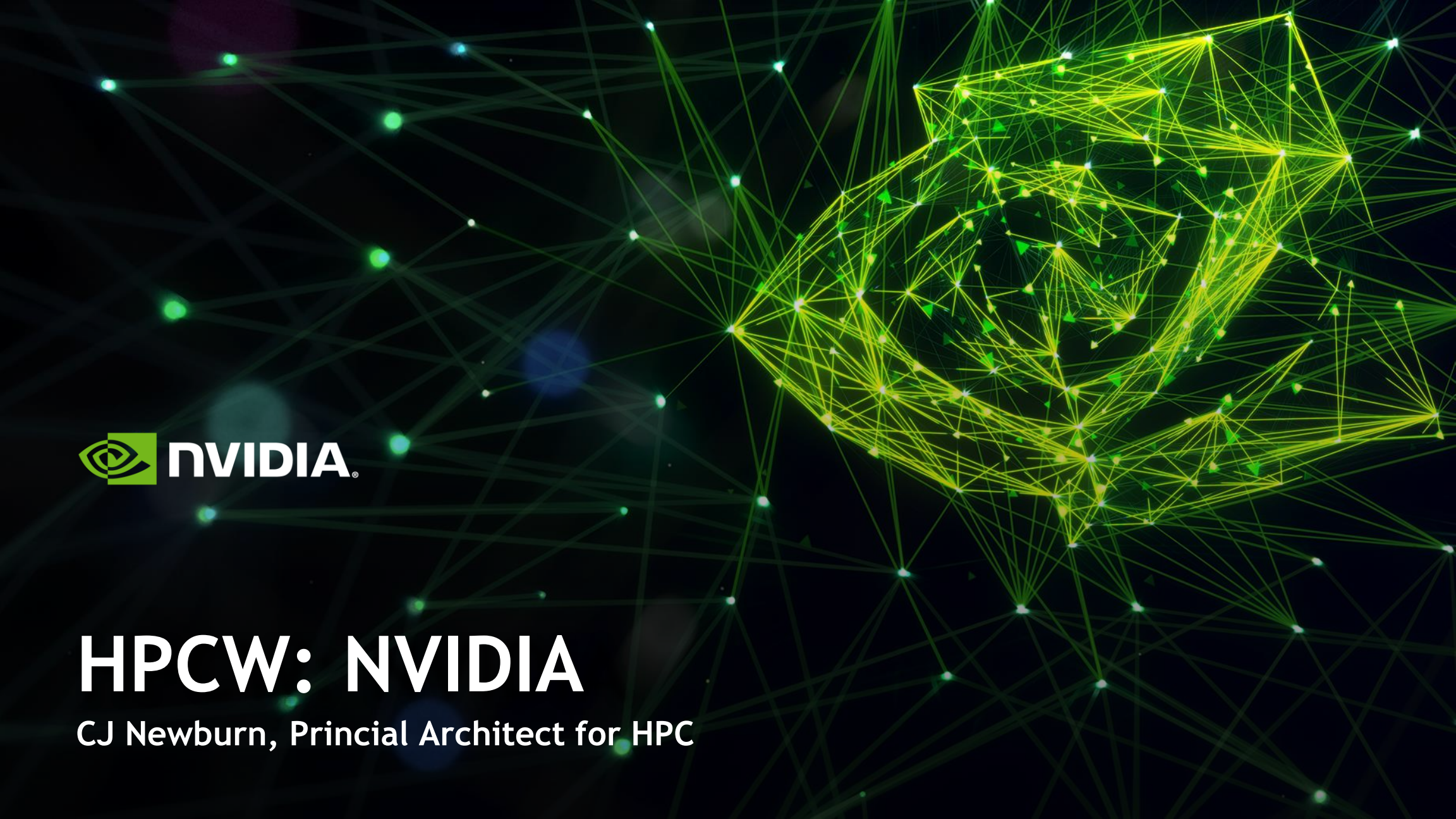


# HPCW: NVIDIA

CJ Newburn, Princial Architect for HPC

# GOALS

- Broad container technology support
- Multiple optimized GPU architectures within a single image
- Optimized multi-node cluster support
- Smallest image size possible

# CONTAINER TECHNOLOGY SUPPORT

- HPCCM outputs Docker files, Singularity recipe files, shell scripts
- NGC images are in Docker format, convertible to Singularity
  - Docker format enables multi-stage builds more easily
  - Singularity supports pulling from NGC natively
    - $ singularity pull docker://nvcr.io/hpc/lammps:24Oct2018
- A custom entrypoint handles setup in a way that's usable by Docker and Singularity
  - Must assume image FS is read-only vs. writing to /usr/lib vs. bind-mounted dir (mofed)
  - User is whoever starts the container, no sudo, no apt get install
- Documentation provides application specific best practices for both runtimes
  - Examples of running canonical problems/benchmarks provided

# CONTAINER TECHNOLOGY ENABLING

- Desires:
  - Common plugin across various container runtimes
- Challenges:
  - Plugin may have limited or only predefined capabilities
  - Want to copy in files from host system (libcuda.so, nvidia-smi.so), bind mount, query inside container for compatibility (e.g. CUDA 9.x/driver 384), set LD_LIBRARY_PATH to use files in container that are compatible with a specific set of kernel mode drivers
- Creative ideas
  - Make plugin parameterizable, e.g. suppress some actions if scheduler pleasant
  - Container technology has different levels of trust for plugins

# MULTI-TARGET SUPPORT

- Single image optimized for Pascal/Volta/Turing when possible
- nvcc can create multi-arch GPU binary targeting all desired architectures
- If build system doesn't support multi-arch compilation, use multiple bins
- entrypoint validates and selects correct binary based on host GPU

# MULTINODE: OPENMPI

- Plugin/component-based architecture makes it very flexible
  - Many NGC images use OpenMPI which supports Slurm, PMI2, PMIx, UCX
  - Whereas MPICH seems to require static compile-time config
  - Most decisions made at runtime, ideal for portable containers
- Provides robust GPU-aware MPI support

- Use of .la metadata files inhibits our flexibility via rpath mechanism

# MULTINODE: UCX

- Alternate choices
  - IB component is default starting with OpenMPI/4.0
  - Or could use legacy OpenIB byte transfer layer
  - Can compare perf between these without recompilation
- UCX features
  - IB, GDRcopy, CUDA IPC, xpmem, knem, cma optimized transports
  - Picks optimized transport at runtime based upon host capabilities
    - Compile-time decisions based upon detection of MOFED on host
    - Only enables GDRcopy if GDRcopy kernel modules available
    - Requires shipping multiple versions in the container

# MULTINODE: INFINIBAND

- Support for Mellanox InfiniBand through MOFED/RDMA-Core
  - User/Kernel driver components not cross version compatible until 4.4+
- Support GPU extensions(nv_peer_mem, gdrcopy)
- Passing in host driver libs can be problematic due to varying transitive dependencies
  - rhel libnl.so <≠> ubuntu libnl-3.so

# MULTINODE: INFINIBAND

- Package multiple MOFED/RDMA-Core releases within container
- Selection handled by entrypoint application

- Relocate libibverbs, libdapl, librdmacm, libmlx4, libmlx5
- Read host kernel driver version from /sys/module/mlx5_core/version
- Set LD_LIBRARY_PATH to best matching libibverbs, libdapl, librdmacm libraries
- Set IBV_DRIVERS to point to best matching libmlx4, libmlx5 driver libraries

# MULTINODE: PMI

- Glue between resource mgrs, process managers, and processes
- Three common APIs( PMI, PMI2, PMIx )
- Implementations not ABI compatible, even within same API
- PMIx/3.x has robust backwards compatibility and solves many container issues
- PMIx/3.x supported by OpenMPI and Slurm
- PMI2 support useful for legacy Slurm integration

# MULTINODE: LAUNCH WITH HOST MPIRUN



- $ mpirun cmd
- $ mpirun singularity run --nv nvidia.simg cmd

- Pros
  - Familiar interface: prefix cmd with Singularity
  - Maintains integration with host resource manager
- Cons
  - Requires compatible host OpenMPI/PMI installation
  - OpenMPI/4.x with PMIx provide good cross version compatibility
  - External mpi may default to using components not in container build

# MULTINODE: LAUNCH WITH HOST SRUN

- $ srun cmd
- $ srun --mpi=pmix singularity run --nv nvidia.simg cmd
- $ srun --mpi=pmi2 singularity run --nv nvidia.simg cmd

- Pros
  - Familiar interface: prefix cmd with Singularity and set PMI
  - Maintains integration with host resource manager
- Cons
  - Requires compatible PMI installation; Slurm PMI2 available on most systems

# MULTINODE: LAUNCH W/ CONTAINER MPIRUN

- $ mpirun cmd

- HOSTFILE=".hostfile.${SLURM_JOB_ID}"
  for host in $(scontrol show hostnames); do
    echo "${host}" >> ${HOSTFILE}
  done
- OMPI_MCA_plm=rsh
  OMPI_MCA_plm_rsh_args='-o PubkeyAcceptedKeyTypes=+ssh-dss -o ...'
  OMPI_MCA_orte_launch_agent="singularity run --nv nvidia.simg orted"
- singularity run nvidia.simg mpirun --hostfile $HOSTFILE cmd

# MULTINODE: LAUNCH W/ CONTAINER MPIRUN

- Pros
  - Works on most systems without external compatibility issues
  - Workload is better contained, better reproducibility
- Cons
  - No integration with host resource manager
  - Exact SSH arguments depend on host specifics

# IMAGE SIZE

- Image size important to users and administrators alike
- Heavy use of Docker multi-stage builds to ensure smallest image possible
- Use tools such as dive to audit image size
- LAMMPS container ~100MB, single "baremetal" binary ~70MB