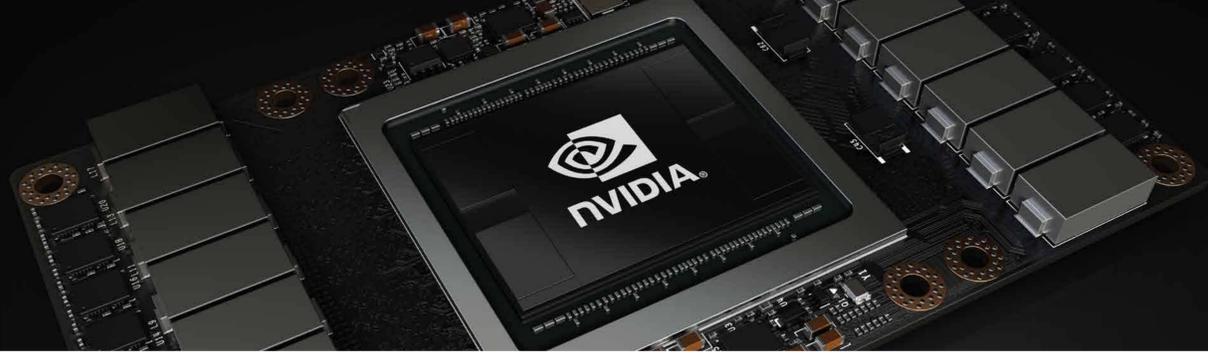# LATTICE QCD ON
NVIDIA® TESLA® V100

APPROACHING QUDA 1.0

QUDA: A LIBRARY FOR QCD ON GPUs

QUDA is an open source community-developed and NVIDIA-supported library for performing LQCD calculations on GPUs, leveraging NVIDIA's CUDA platform. QUDA provides highly optimized mixed-precision linear solvers, eigen-vector solvers, gauge-link fattening and fermion force algorithms.

Supported fermion types are: Wilson, Wilson-clover, twisted mass, twisted mass clover, naïve staggered, improved staggered (ASQTAD or HISQ), domain-wall (4-d or 5-d) and möbius.

Use of multiple GPUs in parallel is supported throughout the library, with inter-GPU communication achieved using MPI or QMP. Several commonly-used LQCD applications integrate support for QUDA as a compile-time option, including Chroma, MILC, CPS, BQCD, TIFR and tmLQCD.

WHY GPUs?

- LQCD simulations are typically memory-bandwidth bound, and so run very efficiently on GPUs.
- LQCD simulations have high degrees of data parallelism that can be expressed effectively using the single instruction multiple data (SIMD) paradigm. This makes LQCD ideal for GPU deployment.
- Most LQCD calculations require only local communication on the 4-d lattice. This makes them suitable for deployment on multiple GPUs through partitioning the lattice into disjoint equal sub-lattices and distributing these between GPUs.

Approaching QUDA 1.0

QUDA is under active development as well as being a production-ready library. As it approaches version 1.0 the development focus concentrates on

- Ease of Development an Stability
- Improved Multi-GPU Scaling
- Readiness for Exascale

EASE OF DEVELOPMENT AND STABILITY

QUDA leverages modern C++11 to simplify code development and achieve high performance.
A flexible git workflow, relying on GitHub pull requests and build testing via Jenkins CI is employed for rapid and flexible development. Cmake is used for build configuration, allowing for running unit based on googletest through ctest.

> Friday 15:00 (Software Development Session): 'Developing QCD Algorithms For NVIDIA GPUs Using the QUDA Framework'
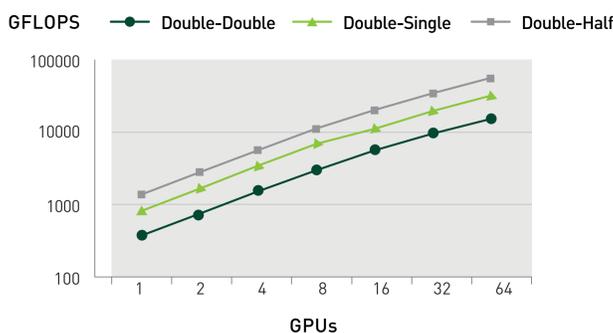
IMPROVED SCALING

QUDA exploits various techniques to improve communication between GPUs on multiple levels:

- CUDA IPC for direct P2P transfers within fat nodes over NVLink and PCIe
- GPU Direct RDMA for internode communication
- Topology aware scheduling of communication

SOLVER SCALING ON SATURN V (DGX-1NODES)

Volume per GPU = 24^4x16, Mixed precision Shamir CG, 8 P100 GPUs per node
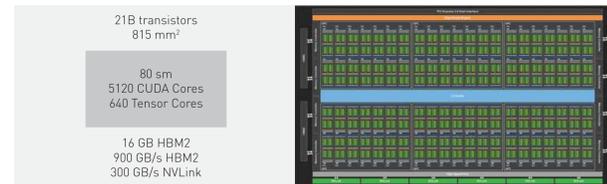

US TO BUILD TWO FLAGSHIP SUPERCOMPUTERS

OAK RIDGE National Laboratory

Lawrence Livermore National Laboratory

SUMMIT	SIERRA
150-300 PFLOPS Peak Performance	> 100 PFLOPS Peak Performance

- IBM POWER9 + NVIDIA Volta V100
- NVLink high-speed interconnect
- > 40 TFLOPS (DP) per node
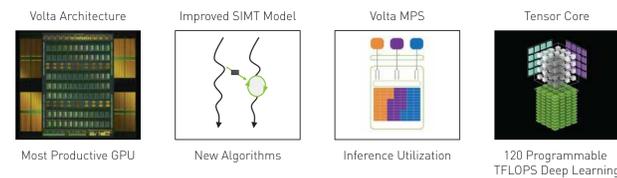- > 4600 nodes (Summit)
- Initial deployment in 2017

TESLA VOLTA V100

21B transistors
815 mm²

80 sm
5120 CUDA Cores
640 Tensor Cores

16 GB HBM2
900 GB/s HBM2
300 GB/s NVLink

GENERATIONAL IMPROVEMENTS

	P100	V100	RATIO
FP64/FP32	5/10 TFLOPS	7.5/15 TFLOPS	1.5X
HBM2 Bandwidth	720 GB/s	900 GB/s	1.2X
NVLink Bandwidth	160 GB/s	300 GB/s	1.9X
L2 Cache	4 MB	6 MB	1.5X
NVLink Bandwidth	1.3 MB	10 MB	7.7X

ARCHITECTURAL IMPROVEMENTS

Volta Architecture — Most Productive GPU
Improved SIMT Model — New Algorithms
Volta MPS — Inference Utilization
Tensor Core — 120 Programmable TFLOPS Deep Learning

IMPROVED MEMORY SUBSYSTEM

Improved NVLink & HBM2

25 Gbps

Efficient Bandwidth

STREAM: Triad- Delivered GB/s

1.5x Delivered Bandwidth

76% DRAM Utilization (P100)
95% DRAM Utilization (V100)

ENHANCED L1 CACHE

Pascal SM
Load/Store Units
Shared Memory 64 KB
L1$ 24 KB
L2$ 4 MB

Volta SM
Load/Store Units
L1$ and Shared Memory 128 KB
L2$ 6 MB

Low Latency
Streaming
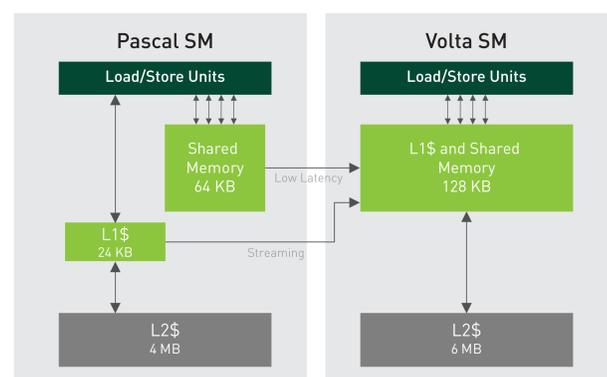
WILSON CLOVER DSLASH

Volume = 32^4

GFLOPS — Reconstruct 8, Reconstruct 12, Reconstruct 18
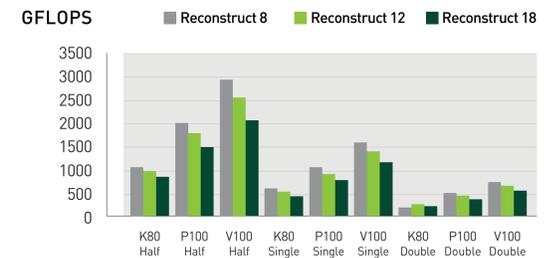
MULTI-GPU WILSON CLOVER DSLASH

Global Volume = 32^4, NVLink

GFLOPS — Reconstruct 8, Reconstruct 12, Reconstruct 18
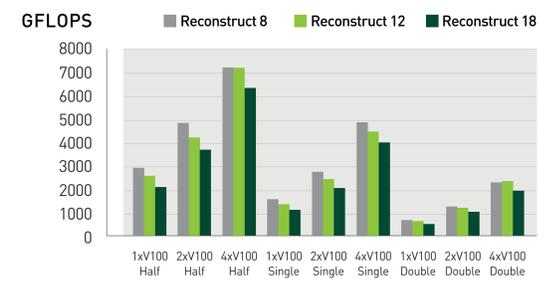
MÖBIUS DSLASH

Volume = 32^4x16

GFLOPS — Reconstruct 8, Reconstruct 12, Reconstruct 18
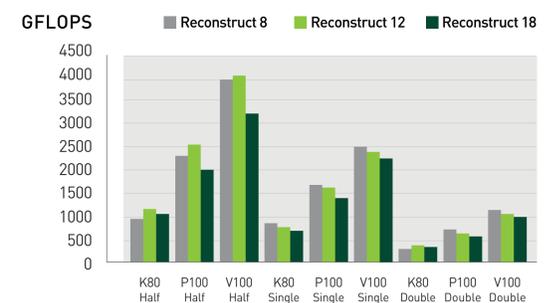
MULTI-GPU MÖBIUS DSLASH

Global Volume = 32^4x16, NVLink

GFLOPS — Reconstruct 8, Reconstruct 12, Reconstruct 18
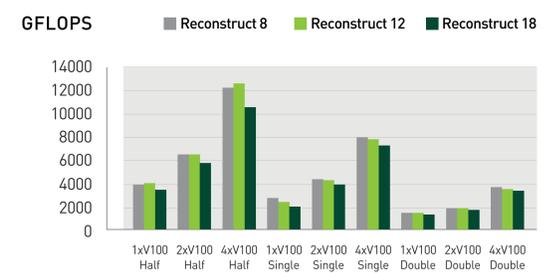
HISQ Dslash for multiple rhs

Volume = 24^4, Gauge reconstruct

GFLOPS — V100 (single), P100 (single), V100 (double), P100 (double)

REFERENCES

https://lattice.github.io/quda
https://developer.nvidia.com/cuda-zone
https://devblogs.nvidia.com/parallelforall/inside-volta/