VOLTA: PROGRAMMABILITY AND PERFORMANCE
## GPU PERFORMANCE COMPARISON

<table>
<thead>
<tr>
<th></th>
<th>P100</th>
<th>V100</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL Training</td>
<td>10 TFLOPS</td>
<td>120 TFLOPS</td>
<td>12x</td>
</tr>
<tr>
<td>DL Inferencing</td>
<td>21 TFLOPS</td>
<td>120 TFLOPS</td>
<td>6x</td>
</tr>
<tr>
<td>FP64/FP32</td>
<td>5/10 TFLOPS</td>
<td>7.5/15 TFLOPS</td>
<td>1.5x</td>
</tr>
<tr>
<td>HBM2 Bandwidth</td>
<td>720 GB/s</td>
<td>900 GB/s</td>
<td>1.2x</td>
</tr>
<tr>
<td>STREAM Triad Perf</td>
<td>557 GB/s</td>
<td>855 GB/s</td>
<td>1.5x</td>
</tr>
<tr>
<td>NVLink Bandwidth</td>
<td>160 GB/s</td>
<td>300 GB/s</td>
<td>1.9x</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>4 MB</td>
<td>6 MB</td>
<td>1.5x</td>
</tr>
<tr>
<td>L1 Caches</td>
<td>1.3 MB</td>
<td>10 MB</td>
<td>7.7x</td>
</tr>
</tbody>
</table>
# NVLINK - PROGRAMMABILITY

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU Mastering</strong></td>
<td>Direct Load/Store access for all agents</td>
</tr>
<tr>
<td></td>
<td>Flat shared address space</td>
</tr>
<tr>
<td><strong>GPU &amp; CPU atomics</strong></td>
<td>Atomic completion at destination</td>
</tr>
<tr>
<td></td>
<td>Read-Modify-Write primitive for unsupported atomics</td>
</tr>
<tr>
<td><strong>Address Translation Services</strong></td>
<td>GPU uses CPU page tables directly</td>
</tr>
</tbody>
</table>
# NVLINK - PERFORMANCE AND POWER

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bandwidth</strong></td>
<td>25Gbps signaling</td>
</tr>
<tr>
<td></td>
<td>6 NVLinks for GV100</td>
</tr>
<tr>
<td></td>
<td>1.9 x Bandwidth improvement over GP100</td>
</tr>
<tr>
<td><strong>Coherence</strong></td>
<td>Latency sensitive CPU caches GMEM</td>
</tr>
<tr>
<td></td>
<td>Fast access in local cache hierarchy</td>
</tr>
<tr>
<td></td>
<td>Probe filter in GPU</td>
</tr>
<tr>
<td><strong>Power Savings</strong></td>
<td>Reduce number of active lanes for lightly loaded link</td>
</tr>
</tbody>
</table>
NVLINK NODES

DL - HYBRID CUBE MESH - DGX-1 w/ Volta

HPC - P9 CORAL NODE - SUMMIT
VOLTA GV100 SM
Redesigned for Productivity and Accessible Performance

- Twice the schedulers
- Simplified Issue Logic
- Large, fast L1 cache
- Improved SIMT model
- Tensor acceleration
- +50% energy efficiency vs GP100 SM
SM MICROARCHITECTURE

Shared L1 I$

4 Independently Scheduled Sub-Cores

Shared MIO (TEX/L1$/SMEM)

L1 I$
4 Warp Instr/clk

Sub-Core
1 Warp Inst/clk
64 B/clk

Sub-Core
1 Warp Int/clk
64 B/clk

Sub-Core
1 Warp Intr/clk
64 B/clk

Sub-Core
1 Warp Intr/clk
64 B/clk

TEX
1 quad/clk

L1 D$ & SMEM
128KB
128 B/clk

L2 $
Warp Scheduler
- 1 Warp instr/clk
- L0 I$, branch unit

Math Dispatch Unit
- Keeps 2+ Datapaths Busy

MIO Instruction Queue
- Hold for Later Scheduling

Two 4x4x4 Tensor Cores
L1 AND SHARED MEMORY

Streaming L1$

- Unlimited cache misses in flight
- Low cache hit latency
- 4x bandwidth vs GP100
- 4x capacity vs GP100

Shared Memory

- Unified Storage with L1$
- Configurable up to 96KB
NARROWING THE SHARED MEMORY GAP
with the GV100 L1 cache

Cache: vs shared
- Easier to use
- 90%+ as good

Shared: vs cache
- Faster atomics
- More banks
- More predictable

Directed testing: shared in global

Average
Shared
Memory
Benefit

0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Pascal
Volta

70%
93%
INDEPENDENT THREAD SCHEDULING

Convergence Optimizer
PROGRAMMING MODEL: MEMORY ADDRESSES

if(...) ... causes Programs written using annotations causes Programs written as normal

$ (CC) $ causes Convergence must be proven causes Convergence is perf optimization

Programs written using annotations

Convergence must be proven

Convergence is prescribed

(e.g. Vector)

GPU Innovation

Convergence is an optimization

(e.g. NVIDIA GPUs)
Convergence is prescribed if(...)

causes

Programs written without synchronization

causes

Convergence must be proven

Convergence is an optimization

Convergence is prescribed (e.g. Scalar + Predicated Vector) (e.g. pre-Volta GPUs)

Volta Innovation (e.g. Volta’s Independent Thread Scheduling)

causes

Programs written as normal

Convergence is perf optimization
NVIDIA SIMT GPUS: SCALAR THREADS ON SIMD UNITS

C, C++, ..., FORTRAN

Array of scalar threads

Scalar Compiler
Scalar ISA
Thread Virtualization Tech
SIMD Units

50+ year history of optimizations
NVIDIA GPUs have always used
30+ year history of TLP-to-ILP conversion
Where the efficiency comes from
COOPERATION ➔ SYNCHRONIZATION

Example: match.any.sync

Warp-synchronizing operation

e.g. our NEW lane data compare

Result distributed across warp
struct mutex {
    void lock() {
        while (!)
            bool state = false;
        if (locked.compare_exchange_weak(state, true, std::memory_order_acquire))
            return;
        while (locked.load(std::memory_order_relaxed))
            do_exponential_backoff(); // for brevity
    }

    void unlock() {
        locked.store(false, std::memory_order_release);
    }

    atomic<bool> locked = ATOMIC_VAR_INIT(false);
};
TENSOR CORE
TENSOR CORE
Mixed Precision Matrix Math
4x4 matrices

\[ D = AB + C \]

FP16 or FP32
FP16
FP16
FP16 or FP32
TENSOR SYNCHRONIZATION

Full Warp 16x16 Matrix Math

Warp-synchronizing operation

Composed Matrix Multiply and Accumulate for 16x16 matrices

Result distributed across warp
VOLTA TENSOR OPERATION

FP16 storage/input

Full precision product

Sum with FP32 accumulator

Convert to FP32 result

Also supports FP16 accumulator mode for inferencing
A GIANT LEAP FOR DEEP LEARNING

Relative Performance

9.3x faster

P100 V100 - Tensor Cores
(CUDA 8) (CUDA 9)

cuBLAS Mixed Precision
(FP16 input, FP32 compute)
Matrix Multiply (M=N=K=2048)

V100 measured on pre-production hardware.
The Fastest and Most Productive GPU for Deep Learning and HPC

More V100 Features: 2x L2 atomics, int8, new memory model, copy engine page migration, MPS acceleration, and more ...

QUESTIONS?