ACCELERATED COMPUTING: THE PATH FORWARD

Axel Koehler, Principal Solution Architect
FACTORS DRIVING CHANGES

End of Dennard Scaling places a cap on single threaded performance
Increasing application performance will require fine grain parallel code with significant computational intensity

Cloud based usage models, in-situ execution and visualization emerging as new workflows critical to the science process and productivity
Tight coupling of interactive simulation, visualization, data analysis/AI
Service Oriented Architectures (SOA)

AI and Data Science emerging as important new components of scientific discovery
Dramatic improvements in accuracy, completeness and response time yield increased insight from huge volumes of data
ACCELERATED COMPUTING
Performance & Energy Efficiency

HIGH PERFORMANCE COMPUTE

AI / DEEP LEARNING

DATA ANALYTICS

ACCELERATED VDI
# NVIDIA TESLA PLATFORM

World’s Leading Data Center Platform for Accelerating HPC and AI

<table>
<thead>
<tr>
<th>CUSTOMER USECASES</th>
<th>CONSUMER INTERNET</th>
<th>ENTERPRISE APPLICATIONS</th>
<th>SUPERCOMPUTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech</td>
<td>Translate</td>
<td>Recommender</td>
<td>Healthcare</td>
</tr>
<tr>
<td>Translate</td>
<td>Recommend</td>
<td>Healthcare</td>
<td>Manufacturing</td>
</tr>
<tr>
<td>Recommend</td>
<td>Engineering</td>
<td></td>
<td>Engineering</td>
</tr>
<tr>
<td>Healthcare</td>
<td>Manufacturing</td>
<td></td>
<td>Engineering</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>Engineering</td>
<td></td>
<td>Molecular Simulations</td>
</tr>
<tr>
<td>Engineering</td>
<td></td>
<td></td>
<td>Weather Forecasting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Seismic Mapping</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INDUSTRY FRAMEWORKS &amp; APPLICATIONS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Caffe</td>
<td>Chainer</td>
</tr>
<tr>
<td>PaddlePaddle</td>
<td>mxnet</td>
</tr>
<tr>
<td>PyTorch</td>
<td>Amber</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>ANSYS</td>
</tr>
<tr>
<td></td>
<td>CHROMA</td>
</tr>
<tr>
<td></td>
<td>GROMACS</td>
</tr>
<tr>
<td></td>
<td>SIMULA</td>
</tr>
<tr>
<td></td>
<td>+550 Applications</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NVIDIA SDK &amp; LIBRARIES</th>
<th>cuBLAS</th>
<th>cuDNN</th>
<th>cuFFT</th>
<th>cuRAND</th>
<th>cuSPARSE</th>
<th>DeepStream</th>
<th>NCCL</th>
<th>TensorRT</th>
<th>PGI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OpenACC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TESLA GPUs &amp; SYSTEMS</th>
<th>TESLA GPU</th>
<th>NVIDIA DGX FAMILY</th>
<th>NVIDIA HGX</th>
<th>SYSTEM OEM</th>
<th>CLOUD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CONTINUED DEMAND FOR COMPUTE POWER

Ever-increasing compute power
Demand in HPC

Neural Network complexity is Exploding

- **2015**
  - 7 ExaFLOPS
  - 60 Million Parameters
  - Microsoft ResNet
  - Superhuman Image Recognition

- **2016**
  - 20 ExaFLOPS
  - 300 Million Parameters
  - Baidu Deep Speech 2
  - Superhuman Voice Recognition

- **2017**
  - 100 ExaFLOPS
  - 8700 Million Parameters
  - Google Neural Machine Translation
  - Near Human Language Translation

Comprehensive Earth System Model
Coupled simulation of entire cells
Simulation of combustion for new high-efficiency, low-emission engines.
Predictive calculations for supernovae
DEEP LEARNING DEMANDS

TRAINING

Billions of TFLOPS per training run
GPU speeds up heavily

INFERENCING

Scalable Performance
Throughput + Efficiency

Billions of FLOPS per inference
GPU for instant response (realtime)

Data / Users
GPUS FOR HPC AND DEEP LEARNING

Huge demand on compute power (FLOPS)

NVIDIA Tesla V100

5120 energy efficient cores + TensorCores
7.8 TF Double Precision (fp64), 15.6 TF Single Precision (fp32),
125 Tensor TFLOP/s mixed-precision

Huge demand on communication and memory bandwidth

CoWoS with HBM2

900 GB/s Memory Bandwidth
Unifying Compute & Memory in Single Package

NVLink

6 links per GPU a 50 GB/s bi-directional for maximum scalability between GPU’s

NCCL

High-performance multi-GPU and multi-node collective communication primitives optimized for NVIDIA GPUs

GPU Direct / GPU Direct RDMA

Direct communication between GPUs by eliminating the CPU from the critical path
New CUDA TensorOp instructions & data formats

4x4x4 matrix processing array

\[ D[\text{FP32}] = A[\text{FP16}] \times B[\text{FP16}] + C[\text{FP32}] \]

Using Tensor cores via

- Volta optimized frameworks and libraries (cuDNN, CuBLAS, TensorRT, ..)
- CUDA C++ Warp Level Matrix Operations
cuBLAS GEMMS FOR DEEP LEARNING
V100 Tensor Cores + CUDA 9: over 9x Faster Matrix-Matrix Multiply

Note: pre-production Tesla V100 and pre-release CUDA 9. CUDA 8 GA release.
LINEAR ALGEBRA + TENSOR CORES

Double Precision LU Decomposition

- Compute initial solution in FP16
- Iteratively refine to FP64

Achieved FP64 Tflops: 26
Device FP64 Tflops: 7.8

Data courtesy of: Azzam Haidar, Stan. Tomov & Jack Dongarra, Innovative Computing Laboratory, University of Tennessee
“Investigating Half Precision Arithmetic to Accelerate Dense Linear System Solvers”, A. Haidar, P. Wu, S. Tomov, J. Dongarra, SC’17
GTC 2018 Poster P8237: Harnessing GPU’s Tensor Cores Fast FP16 Arithmetic to Speedup Mixed-Precision Iterative Refinement Solves
HBM2 MEMORY ARCHITECTURE

- Unifying Compute & Memory in Single Package
- More bandwidth and more energy efficient
- ECC can be active without a bandwidth or capacity penalty

STREAM: Triad-Delivered GB/s

1.5x Delivered Bandwidth

P100
76% DRAM Utilization

V100
95% DRAM Utilization
UNIFIED MEMORY
Large datasets, simple programming, High Performance

CUDA 8 and beyond

GPU

CPU

Unified Memory

Allocate Beyond GPU Memory Size

Enable Large Data Models
Oversubscribe GPU memory
Allocate up to system memory size

Tune
Unified Memory Performance
Usage hints via cudaMemAdvise API
Explicit prefetching API

Simpler Data Access
CPU/GPU Data coherence
Unified memory atomic operations
VOLTA NVLINK

- 6 NVLINKS @ 50 GB/s bidirectional
- Reduce number of lanes for lightly loaded link (Power savings)
- Coherence features for NVLINK enabled CPUs

Hybrid cube mesh (eg. DGX1V)

POWER9 based node
COMMUNICATION BETWEEN GPUS

Large scale models:
- Some models are too big for a single GPU and need to be spread across multiple devices and multiple nodes
- The size of the model will further increase in the future

Data parallel training
- Each worker trains the same layers on a different data batch

Model parallel training
- All workers train on same batch; workers communicate as frequently as network allows
- NVLINK allows the separation of data loading and exchanges for activation

http://mxnet.io/how_to/multi_devices.html
**NVLINK AND MULTI-GPU SCALING**

For Data Parallel Training

- Data loading over PCIe
- Gradient averaging over PCIe and QPI
- Data loading and gradient averaging share communication resources: Congestion

---

**PCIe based system**

**NVLINK based system**

- Data loading over PCIe (red)
- Gradient averaging over NVLink (blue)
- No sharing of communication resources: No congestion
NVLINK AND CNTK MULTI-GPU SCALING
VOLTA MULTI-PROCESS SERVICE

Volta MPS Enhancements:

• MPS clients submit work directly to the work queues within the GPU
  • Reduced launch latency
  • Improved launch throughput

• Improved isolation amongst MPS clients
  • Address isolation with independent address spaces
  • Improved quality of service (QoS)

• 3x more clients than Pascal

Hardware Accelerated Work Submission
Hardware Isolation
Volta GV100
VOLTA MPS FOR INFERENCE
Efficient inference deployment without batching system

V100 measured on pre-production hardware.
TESLA GPUS
# V100 WITH 16 OR 32GB HBM2

Maintain Form Factor Compatibility

<table>
<thead>
<tr>
<th>Form Factor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>7.8T F DP, 15.7 TF SP, 125TF TensorCore</td>
</tr>
<tr>
<td>Memory Size</td>
<td>16 or 32GB HBM2</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>900GB/s</td>
</tr>
<tr>
<td>GPU Peer to Peer</td>
<td>NVLink</td>
</tr>
<tr>
<td>Power</td>
<td>300W</td>
</tr>
<tr>
<td>Available From All Major OEMs</td>
<td></td>
</tr>
</tbody>
</table>

SXM2 32GB P/N = 900-2G503-0010-000, PCIe 32GB P/N = 900-2G500-0010-000
TESLA T4

2,560 CUDA cores + 320 Tensor Cores
8.1 TFLOPS FP32 | 65 FP16 TFLOPS
130 INT8 TOPS | 260 INT4 TOPS

16GB GDDR6 Memory | 320GB/s

75 W Low Profile PCI-e
TURING TENSOR CORE

- FP16: 8x CUDA core, 40-70 TFLOPs
- Int8: 16x CUDA core, 80-140 TOPs
- Int4: 32x CUDA core, 160-280 TOPs
- Binary: 128x CUDA core, 640-1100 TOPs

SW at Launch
- TensorRT, Libraries
- CUTLASS Open Source Tensor Library, CUDA

Downloaded from Arrow.com.
## TESLA PRODUCTS DECODER

<table>
<thead>
<tr>
<th></th>
<th>P100 (SXM2)</th>
<th>P100 (PCIE)</th>
<th>P40</th>
<th>P4</th>
<th>T4</th>
<th>V100 (PCIE)</th>
<th>V100 (SXM2)</th>
<th>V100 (FHHL)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GPU CHIP</strong></td>
<td>GP100</td>
<td>GP100</td>
<td>GP102</td>
<td>GP104</td>
<td>TU104</td>
<td>GV100</td>
<td>GV100</td>
<td>GV100</td>
</tr>
<tr>
<td><strong>PEAK FP64 (TFLOps)</strong></td>
<td>5.3</td>
<td>4.7</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>7</td>
<td>7.8</td>
<td>6.5</td>
</tr>
<tr>
<td><strong>PEAK FP32 (TFLOps)</strong></td>
<td>10.6</td>
<td>9.3</td>
<td>12</td>
<td>5.5</td>
<td>8.1</td>
<td>14</td>
<td>15.7</td>
<td>13</td>
</tr>
<tr>
<td><strong>PEAK FP16 (TFLOps)</strong></td>
<td>21.2</td>
<td>18.7</td>
<td>NA</td>
<td>NA</td>
<td>65</td>
<td>112</td>
<td>125</td>
<td>105</td>
</tr>
<tr>
<td><strong>PEAK TOPs</strong></td>
<td>NA</td>
<td>NA</td>
<td>47</td>
<td>22</td>
<td>260</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Memory Size</strong></td>
<td>16 GB HBM2</td>
<td>16/12 GB HBM2</td>
<td>24 GB GDDR5</td>
<td>8 GB GDDR5</td>
<td>16 GB HBM2</td>
<td>32 GB HBM2</td>
<td>32 GB HBM2</td>
<td>16GB HBM2</td>
</tr>
<tr>
<td><strong>Memory BW</strong></td>
<td>732 GB/s</td>
<td>732/549 GB/s</td>
<td>346 GB/s</td>
<td>192 GB/s</td>
<td>320GB/s</td>
<td>900 GB/s</td>
<td>900 GB/s</td>
<td>900 GB/s</td>
</tr>
<tr>
<td><strong>Interconnect</strong></td>
<td>NVLINK + PCIe Gen3</td>
<td>PCIe Gen3</td>
<td>PCIe Gen3</td>
<td>PCIe Gen3</td>
<td>PCIe Gen3</td>
<td>NVLINK + PCIe Gen3</td>
<td>PCIe Gen3</td>
<td>PCIe Gen3</td>
</tr>
<tr>
<td><strong>ECC</strong></td>
<td>Internal + HBM2</td>
<td>Internal + HBM2</td>
<td>GDDR5</td>
<td>GDDR5</td>
<td>GDDR6</td>
<td>Internal + HBM2</td>
<td>Internal + HBM2</td>
<td>Internal + HBM2</td>
</tr>
<tr>
<td><strong>Form Factor</strong></td>
<td>SXM2</td>
<td>PCIE Dual Slot</td>
<td>PCIE Dual Slot</td>
<td>PCIE LP</td>
<td>PCIE LP</td>
<td>PCIE Dual Slot</td>
<td>SXM2</td>
<td>PCIE Single Slot Full Height Half Length</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>300 W</td>
<td>250 W</td>
<td>250 W</td>
<td>50-75 W</td>
<td>75 W</td>
<td>250W</td>
<td>300W</td>
<td>150W</td>
</tr>
</tbody>
</table>
DGX-STATION / DGX-1
DGX-2 / HGX-2
NVIDIA DGX-STATION
AI supercomputer for the desk

4x Tesla V100 connected via NVLINK
(60 TFLOPS FP32, 0.5 PFLOPS Tensor performance)
Xeon CPU, 256 GB Memory
Storage:
   3X 1.92 TB SSD RAID 0 (Data)
   1X 1.92 TB SSD (OS)
Dual 10GbE
1500W, Water-cooled → Quiet
Optimized Deep Learning Software across the entire stack
   Containerized frameworks
   Always up-to-date via the cloud
NVIDIA DGX-1
AI supercomputer-appliance-in-a-box

8x Tesla V100 connected via NVLINK
(125 TFLOPS FP32, 1 PFLOPS Tensor Core performance)
Dual Xeon CPU, 512 GB Memory
7 TB SSD Deep Learning Cache
Dual 10GbE, Quad IB 100Gb
3RU - 3200W
Optimized Deep Learning Software across the entire stack
  Containerized frameworks
  Always up-to-date via the cloud
NVIDIA DGX-2

1. NVIDIA Tesla V100 32GB
2. Two GPU Boards
   - 8 V100 32GB GPUs per board
   - 6 NVSwitches per board
   - 512GB Total HBM2 Memory interconnected by Plane Card
3. Twelve NVSwitches
   - 2.4 TB/sec bi-section bandwidth
4. Eight EDR Infiniband/100 GigE
   - 1600 Gb/sec Total Bi-directional Bandwidth
5. PCIe Switch Complex
6. Two Intel Xeon Platinum CPUs
7. 1.5 TB System Memory
8. 30 TB NVME SSDs
   - Internal Storage
9. Dual 10/25 Gb/sec Ethernet
NVSWITCH

• 18 NVLINK ports
  • @50 GB/s per port bi-directional
  • 900 GB/s total bi-directional
• Fully connected crossbar
• X4 PCIe Gen2 Management port
• GPIO
• I2C
• 2 billion transistors
FULL NON-BLOCKING BANDWIDTH
FULL 6-WAY POINT-TO-POINT

NVSwitch Fabric
NVSWITCH

NVLINK PROVIDES
• All-to-all high-bandwidth peer mapping between GPUs
• Full inter-GPU memory interconnect (incl. Atomics)

UNIFIED MEMORY PROVIDES
• Single memory view shared by all GPUs
• Automatic migration of data between GPUs
• User control of data locality
2X HIGHER PERFORMANCE WITH NVSWITCH

- **Physics (MILC benchmark)**: 2x faster
- **Weather (ECMWF benchmark)**: 2.4x faster
- **Recommender (Sparse Embedding)**: 2x faster
- **Language Model (Transformer with MoE)**: 2.7x faster

2x DGX-1 (Volta) vs. DGX-2 with NVSwitch

2 DGX-1V servers have dual socket Xeon E5 2698v4 Processor, 8 x V100 GPUs. Servers connected via 4X 100Gb IB ports. DGX-2 server has dual-socket Xeon Platinum 8168 Processor, 16 V100 GPUs.
GPU PROGRAMMING
CUDA DEVELOPMENT ECOSYSTEM

- **GPU Users**
- **Domain Specialists**
- **Problem Specialists**
- **New Algorithm Developers and Optimization Experts**

Applications

Frameworks

Libraries

Directives and Standard Languages

Extended Standard Languages

---

Ease of use

Specialized Performance

CUDA: Programming Model, GPU Architecture, System Architecture
WHAT IS OPENACC
Programming model for an easy onramp to GPUs

Directives-based programming model for parallel computing

Add Simple Compiler Directive

```c
main()
{
    <serial code>
    #pragma acc kernels
    {
        <parallel code>
    }
}
```

Simple

Designed for performance portability on CPUs and GPUs

Powerful & Portable

Read more at www.openacc.org/about

OpenACC is an open specification developed by OpenACC.org consortium
OPENACC

Incremental
- Make small, incremental changes to the code
- If any errors occur, easily able to revert back to an earlier, working version of the code
- Start with a working sequential code, and add improvements

Single Source
- A single OpenACC code can be compiled for, and ran on, many different parallel hardware
- An OpenACC code retains its ability to run sequentially at all times
- No need for multiple versions of your code

Low Learning Curve
- OpenACC is meant to be easy to use, and easy to learn
- Supports C, C++, and Fortran coding
- Takes a very high-level approach to parallelism, and allows the compiler to do a lot of extra work in parallelizing the code
PGI — THE NVIDIA HPC SDK

Fortran, C & C++ Compilers
   Optimizing, SIMD Vectorizing, OpenMP

Accelerated Computing Features
   OpenACC Directives, CUDA Fortran

Multi-Platform Solution
   X86-64 and OpenPOWER Multicore CPUs
   NVIDIA Tesla GPUs
   Supported on Linux, macOS, Windows

MPI/OpenMP/OpenACC Tools
   Debugger
   Performance Profiler
   Interoperable with DDT, TotalView
**PGI COMPILERS FOR EVERYONE**
The PGI 18.4 Community Edition

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenACC, CUDA Fortran, OpenMP, C/C++/Fortran Compilers and Tools</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X86, OpenPOWER, NVIDIA GPU</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2 times a year</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User Forums</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Annual</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

[pgicompiers.com/community](http://pgicompiers.com/community)
CUDA RELEASES

Accelerating the Pace

Four CUDA releases per year

Faster release cadence for new features and improved stability for existing users

Upcoming limited decoupling of display driver and CUDA release for ease of deployment

Monthly cuDNN & other library updates

Rapid innovation in library performance and functionality

Library Meta Packages independent of toolkit for easy deployment
INTRODUCING CUDA 10.0

TURING AND NEW SYSTEMS
New GPU Architecture, Tensor Cores, NVSwitch Fabric

CUDA PLATFORM
CUDA Graphs, Vulkan & DX12 Interop, Warp Matrix

LIBRARIES
GPU-accelerated hybrid JPEG decoding, Symmetric Eigenvalue Solvers, FFT Scaling

DEVELOPER TOOLS
New Nsight Products - Nsight Systems and Nsight Compute
**cuFFT 10.0**

*Multi-GPU Scaling across DGX-2 and HGX-2*

- Strong scaling across 16-GPU systems - DGX-2 and HGX-2
- Multi-GPU R2C and C2R support
- Large FFT models across 16-GPUs - effective 512GB vs 32GB capacity

---

**Up to 17TF performance on 16-GPUs**

3D 1K FFT

---

![Graph showing performance comparison](https://developer.nvidia.com/cufft)

*cuFFT (10.0 and 9.2) using 3D C2C FFT 1024 size on DGX-2 with CUDA 10 (10.0.130)*
NSIGHT PRODUCT FAMILY

Nsight Systems
System-wide application algorithm tuning

Nsight Compute
CUDA Kernel Profiling and Debugging

Nsight Graphics
Graphics Shader Profiling and Debugging

IDE Plugins
Nsight Eclipse Edition/Visual Studio (Editor, Debugger)
NSIGHT SYSTEMS
System-wide Performance Analysis

Observe Application Behavior: CPU threads, GPU traces, Memory Bandwidth and more

Locate Optimization Opportunities: CUDA & OpenGL APIs, Unified Memory transfers, User Annotations using NVTX

Ready for Big Data: Fast GUI capable of visualizing in excess of 10 million events on laptops, Container support, Minimum user privileges

https://developer.nvidia.com/nsight-systems
NVIDIA NSIGHT COMPUTE
Next Generation Kernel Profiler

- Interactive CUDA API debugging and kernel profiling
- Fast Data Collection
- Improved Workflow and Fully Customizable (Baselining, Programmable UI/Rules)
- Command Line, Standalone, IDE Integration
- Platform Support
  - OS: Linux (x86, ARM), Windows
  - GPUs: Pascal, Volta, Turing
NVIDIA DEEP LEARNING SOFTWARE STACK

TRAINING

Data Management

Training

Trained Neural Network

Model Assessment

INFERENCEx

Data center

TensorRT

Embedded

JETPACK SDK

Automotive

DriveWorks SDK

NVIDIA DEEP LEARNING SDK and CUDA

cuDNN

NCCL

cuBLAS

TensorRT

cuSPARSE

DeepStream SDK

developer.nvidia.com/deep-learning-software
NVIDIA cuDNN 7
Deep Learning Primitives

High performance building blocks for deep learning frameworks

Drop-in acceleration for widely used deep learning frameworks such as Caffe2, Microsoft Cognitive Toolkit, PyTorch, Tensorflow, Theano and others

Accelerates industry vetted deep learning algorithms, such as convolutions, LSTM RNNs, fully connected, and pooling layers

Fast deep learning training performance tuned for NVIDIA GPUs

developer.nvidia.com/cudnn

Deep Learning Training Performance

“NVIDIA has improved the speed of cuDNN with each release while extending the interface to more operations and devices at the same time.”

— Evan Shelhamer, Lead Caffe Developer, UC Berkeley
NVIDIA Collective Communications Library (NCCL) 2

Multi-GPU and multi-node collective communication primitives

High-performance multi-GPU and multi-node collective communication primitives optimized for NVIDIA GPUs

Fast routines for multi-GPU multi-node acceleration that maximizes inter-GPU bandwidth utilization

Easy to integrate and MPI compatible. Uses automatic topology detection to scale HPC and deep learning applications over PCIe and NVink

Accelerates leading deep learning frameworks such as Caffe2, Microsoft Cognitive Toolkit, MXNet, PyTorch and more

developer.nvidia.com/nccl
WHAT’S NEW IN NCCL 2

Performance
• Delivers over 90% multi-node scaling efficiency using up to eight GPU-accelerated servers

New Features
• Multi-node, multi-GPU communication collectives
• Automatic topology detection to determine optimal communication path
• Optimized to achieve high bandwidth over PCIe and NVink high-speed interconnect

Available now as a free download to members of NVIDIA Developer Program

developer.nvidia.com/nccl
Horovod is a distributed training framework for TensorFlow. The goal of Horovod is to make distributed Deep Learning fast and easy to use.

Leverage TensorFlow + MPI + NCCL2 for a simplified and performant API to enable synchronous multigpu + multinode Tensorflow.

Instead of Parameter Server architecture leverage MPI.

Support features such as RDMA, GPUDirectRDMA (GDR), via leveraging MPI and NCCL2.
AI INFERENCING IS EXPLODING

PERSONALIZATION
2 Trillion Messages Per Day On LinkedIn

SPEECH
500M Daily active users of iFlyTek

TRANSLATION
140 Billion Words Per Day Translated by Google

VIDEO
60 Billion Video frames/day uploaded on Youtube
NVIDIA TensorRT
Deep Learning Inference Optimizer and Runtime

High performance neural network inference optimizer and runtime engine for production deployment

Maximize inference throughput for latency-critical services in hyperscale datacenters, embedded, and automotive production environments

Optimize TensorFlow and ONNX-framework models to generate high-performance runtime engines

Deploy faster, more responsive and memory efficient deep learning applications with INT8 and FP16 optimized precision support
NVIDIA TENSORRT 4

Maximize RNN and MLP Throughput

- RNN and MLP Layers
- ONNX Import
- NVIDIA DRIVE Support

Optimize and Deploy ONNX Models

- Easily import and accelerate inference for ONNX frameworks (PyTorch, Caffe 2, CNTK, MxNet and Chainer)

Support for NVIDIA DRIVE Xavier

- Deploy optimized deep learning inference models NVIDIA DRIVE Xavier

Free download to members of NVIDIA Developer Program
developer.nvidia.com/tensorrt
**TENSORRT INTEGRATED WITH TENSORFLOW**

Speed up TensorFlow inference with TensorRT optimizations

Speed up TensorFlow model inference with TensorRT with new TensorFlow APIs

Simple API to use TensorRT within TensorFlow easily

Sub-graph optimization with fallback offers flexibility of TensorFlow and optimizations of TensorRT

Optimizations for FP32, FP16 and INT8 with use of Tensor Cores automatically

```python
# Apply TensorRT optimizations
trt_graph = trt.create_inference_graph(frozen_graph_def, output_node_name, max_batch_size=batch_size, max_workspace_size_bytes=workspace_size, precision_mode=precision)

# INT8 specific graph conversion
trt_graph = trt.calib_graph_to_infer_graph(calibGraph)
```

Available from TensorFlow 1.7
https://github.com/tensorflow/tensorflow
TENSORRT 5 & TENSORRT INFERENCE SERVER

World’s Most Advanced Inference Accelerator

New optimizations & flexible INT8 APIs

TensorRT inference server

Up to 40x faster inference for apps such as translation using mixed precision on Turing Tensor Cores

Achieve highest throughput at low latency with newly optimized operations, INT8 workflows, and support for Win and CentOS

Maximize GPU utilization by executing multiple models from different frameworks on a node via API

developer.nvidia.com/tensorrt
KUBERNETES on NVIDIA GPUs

- Scale-up Thousands of GPUs Instantly
- Self-healing Cluster Orchestration
- GPU Optimized Out-of-the-Box
- Powered by NVIDIA Container Runtime
- Included with Enterprise Support on DGX
THE BIG PROBLEM IN DATA SCIENCE

Slow Training Times for Data Scientists
RAPIDS — OPEN GPU DATA SCIENCE

Software Stack

Data Preparation → Model Training → Visualization

DASK

PYTHON

RAPIDS

CUDF

CUML

CUGRAPH

DEEP LEARNING FRAMEWORKS

CUDNN

CUDA

APACHE ARROW

RAPIDS is a set of open source libraries for GPU accelerating data preparation and machine learning

www.rapids.ai
cuDF — ANALYTICS
GPU DataFrame Library Built on Apache Arrow

Libraries

daskgdf: Distributed Computing pygdf using Dask; Support for multi-GPU, multi-node
pygdf: Python bindings for libgdf (Pandas like API for DataFrame manipulation)
libgdf: CUDA C++ Apache Arrow GPU DataFrame and operators (Join, GroupBy, Sort, etc.)

Memory Allocation Requirement
Budget 2-3X dataset size for cuDF working memory

Multi-GPU Multi-node Roadmap

<table>
<thead>
<tr>
<th>Availability</th>
<th>Multi-GPU</th>
<th>Multi-Node</th>
<th>Peer-to-peer Data Sharing*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Now</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Q4 2018</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Note: No peer-to-peer data sharing means computation performed via map/reduce style programming in Dask

Downloaded from Arrow.com.
cuIO — FILE I/O

Direct File Loading to cuDF

cuIO file readers are GPU accelerated and load data directly into cuDF

Dask parallelizes data ingestion across cores

<table>
<thead>
<tr>
<th>Availability</th>
<th>Supported File Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Now</td>
<td>CSV</td>
</tr>
<tr>
<td>Q4 2018</td>
<td>Parquet, ORC</td>
</tr>
</tbody>
</table>

Note: Dask/Pandas can be used to read all formats CSV, Parquet, ORC, JSON, AVRO in cuDF; However, it is slower because it uses the CPU and needs to be read to system memory, then copied over to GPU memory
cuML — MACHINE LEARNING
GPU Accelerated Scikit-learn + XGBoost Libraries

Dask
**Distributed Training**: Used for distributed cuML model training

Python API
**Language Bindings**: Python bindings to C++/CUDA based cuML | Uses cuDF DataFrames as input

cuML
**C++/CUDA ML Algorithms**: C++/CUDA machine learning algorithms

ml-prims
**CUDA ML Primitives**: Low level machine learning primitives used in cuML | Linear algebra, statistics, matrix operations, distance functions, random number generation
# cuML — ROADMAP

Scikit-learn + XGBoost

<table>
<thead>
<tr>
<th>cuML Algorithms</th>
<th>Available Now</th>
<th>Q4-2018</th>
<th>Q1-2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>XGBoost GBDT</td>
<td>MGMN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Truncated Singular Value Decomposition (tSVD)</td>
<td>SG</td>
<td>MG</td>
<td></td>
</tr>
<tr>
<td>Principal Component Analysis (PCA)</td>
<td>SG</td>
<td>MG</td>
<td></td>
</tr>
<tr>
<td>Density-based Spatial Clustering of Applications with Noise (DBSCAN)</td>
<td>SG</td>
<td>MG</td>
<td></td>
</tr>
<tr>
<td>XGBoost Random Forest</td>
<td></td>
<td>MGMN</td>
<td></td>
</tr>
<tr>
<td>K-Means Clustering</td>
<td>MG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kalman Filter</td>
<td>SG</td>
<td>MG</td>
<td></td>
</tr>
<tr>
<td>FAISS K-NN</td>
<td>MG</td>
<td>MGMN</td>
<td></td>
</tr>
<tr>
<td>GLM (including Logistic)</td>
<td></td>
<td></td>
<td>MGMN</td>
</tr>
<tr>
<td>Time Series</td>
<td>MG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support Vector Machines</td>
<td>MGMN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collaborative Filtering</td>
<td>MG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UMAP</td>
<td>MG</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Last updated 10.16.18
cuGRAPH — GRAPH ANALYTICS

GPU Accelerated Unified Graph Analytics

- Unifies the GPU accelerated graph analytics libraries nvGraph, Gunrock and Hornet
- Available in 2019
BENCHMARKS

**cuIO/cuDF — Load and Data Preparation**

- 20 CPU Nodes: 2.741
- 30 CPU Nodes: 1.675
- 50 CPU Nodes: 715
- 100 CPU Nodes: 379
- DGX-2: 42
- 5x DGX-1: 19

**cuML — XGBoost**

- 20 CPU Nodes: 2.290
- 30 CPU Nodes: 1.956
- 50 CPU Nodes: 1.999
- 100 CPU Nodes: 1.948
- DGX-2: 169
- 5x DGX-1: 157

**End-to-End**

- 20 CPU Nodes: 8.000
- 30 CPU Nodes: 6.000
- 50 CPU Nodes: 4.000
- 100 CPU Nodes: 2.000

Time in seconds — Shorter is better

- **cuIO / cuDF (Load and Data Preparation)**
- **Data Conversion**
- **XGBoost**

---

**Benchmark**

200GB CSV dataset; Data preparation includes joins, variable transformations.

**CPU Cluster Configuration**

CPU nodes (61 GiB of memory, 8 vCPUs, 64-bit platform), Apache Spark

**DGX Cluster Configuration**

5x DGX-1 on InfiniBand network
DOWNLOAD AND DEPLOY

Source available on Github | Container available on NGC and Dockerhub | PIP available at a later date

Source code, libraries, packages

On-premises

Cloud
CHALLENGES

Current DIY deep learning environments are complex and time consuming to build, test and maintain.

Development of frameworks by the community is moving very quickly.

Requires high level of expertise to manage driver, library, framework dependencies.
SIMPLIFY PORTABILITY WITH NVIDIA CONTAINERS

Benefits of Containers:

- Simplify deployment of GPU-accelerated applications
- Isolate individual frameworks or applications
- Share, collaborate, and test applications across different environments
NVIDIA GPU CLOUD REGISTRY
Common Software stack across NVIDIA GPUs

**Deep Learning**
All major frameworks with multi-GPU optimizations Uses NCCL for NVLINK data exchange Multi-threaded I/O to feed the GPUs

- Caffe, Caffe2, CNTK, mxnet, PyTorch, Tensorflow, Theano, Torch

**HPC**
NAMD, Gromacs, LAMMPS, GAMESS, Relion, Chroma, MILC

**HPC Visualization**
Paraview with Optix, Index and Holodeck with OpenGL visualization base on NVIDIA Docker 2.0, IndeX, VMD

**Single NGC Account**
For use on GPUs everywhere - [https://ngc.nvidia.com](https://ngc.nvidia.com)

NVIDIA GPU Cloud containerizes GPU-optimized frameworks, applications, runtimes, libraries, and operating system, available at no charge
HPC CONTAINER MAKER - HPCCM
“h-p-see-um”

- HPC Container Maker (HPCCM) generates container specification files (Dockerfiles or Singularity recipe) based on recipes
- A recipe specifies the series of steps to be performed when building a container

https://github.com/NVIDIA/hpc-container-maker
HPC CONTAINER MAKER - HPCCM

- Container implementation abstraction
  - The same recipe file generates specification files for Docker or Singularity
- Availability of full programming languages
  - A recipe is Python code. This means that you can use the full power of Python in a recipe for conditional branching, input validation, searching the web for the latest version of a component, etc.
- Higher level abstraction
  - Provides building blocks to simplify recipes and encapsulate best practices
- Container Maker generates human readable Dockerfiles and Singularity recipe files
CONVERGENCE OF HPC AND AI
# COMBINING THE STRENGTHS OF HPC AND AI

## HPC
- Proven algorithms based on first principles theory
- Proven statistical models for accurate results in multiple science domains
- Develop training data sets using first principal models
- Incorporate AI models in semi-empirical style applications to improve throughput
- Validate new findings from AI

## AI
- New methods to improve predictive accuracy, insight into new phenomena and response time
- Implement inference models with real time interactivity
- Train inference models to improve accuracy and comprehend more of the physical parameter space
- Analyze data sets that are simply intractable with classic statistical models
- Control and manage complex scientific experiments
NVIDIA DEEP LEARNING INSTITUTE (DLI)
Hands-on training for developers, data scientists, and researchers

Online self-paced labs across beginner and intermediate levels available at www.nvidia.com/dlilabs

Onsite workshops covering e.g. Deep Learning Fundamentals can be requested through our page www.nvidia.com/requestDLI
SUMMARY

• Same GPU technology enabling powerful science is also enabling the revolution in deep learning

• Convergence of HPC and Deep learning
  • Applications can use DL to train neural networks with already simulated data and DL network can predict about the output

• Rich Development Ecosystem

• NVIDIA GPU Cloud provides container images for DL frameworks and HPC applications