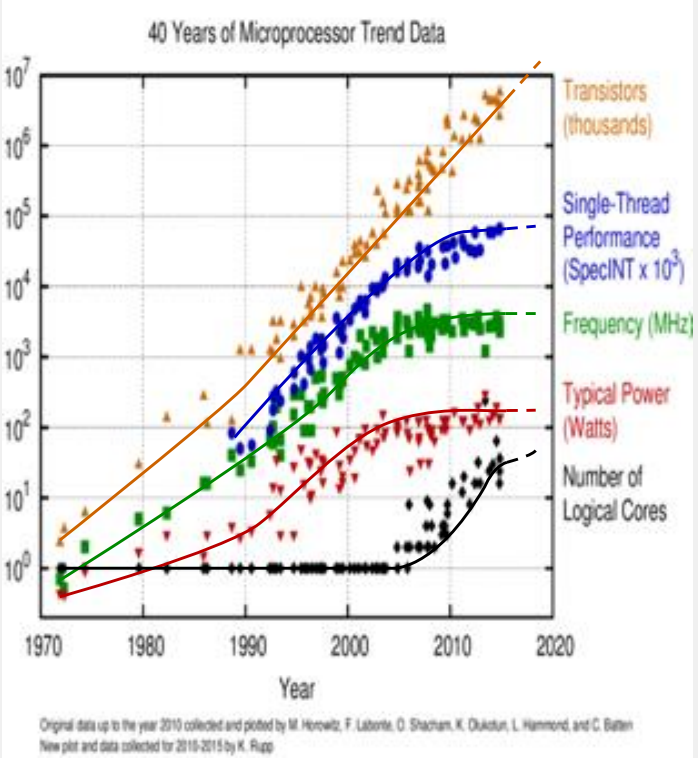
ACCELERATED COMPUTING

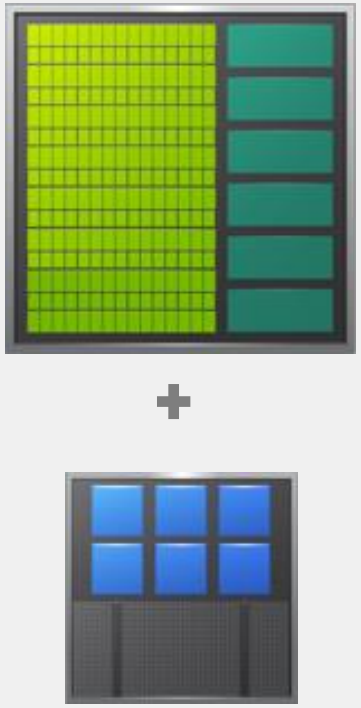
Fred Allman

January 2019

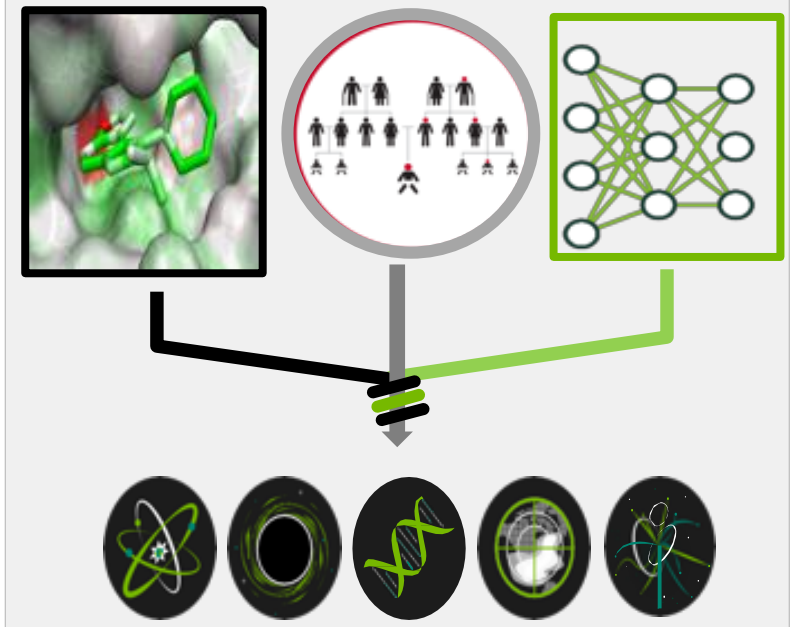
FORCES SHAPING SUPERCOMPUTING



END OF MOORES LAW

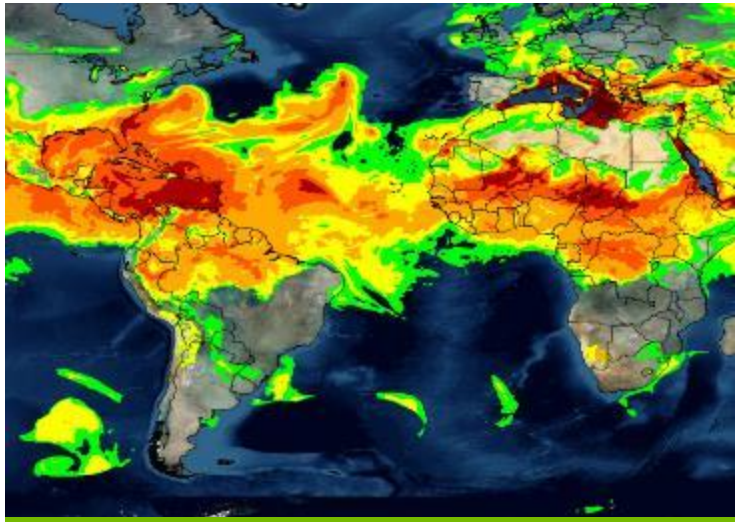


ACCELERATED COMPUTING



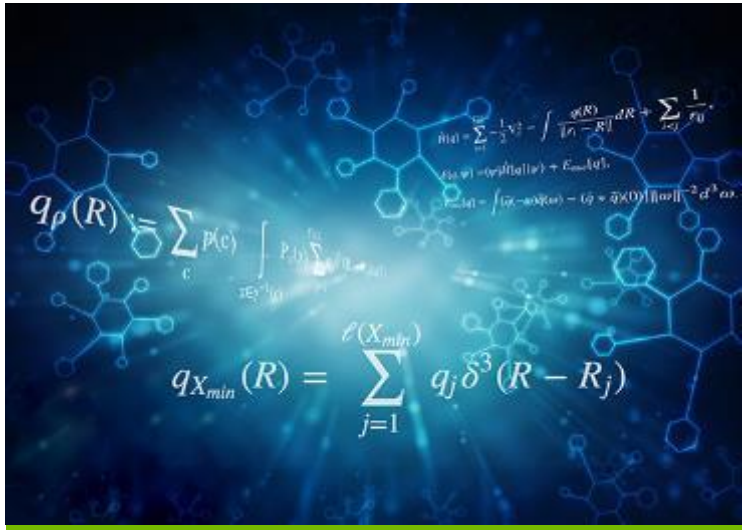
AI – A NEW TOOL FOR SCIENCE

THE NEW HPC MARKET



SIMULATION

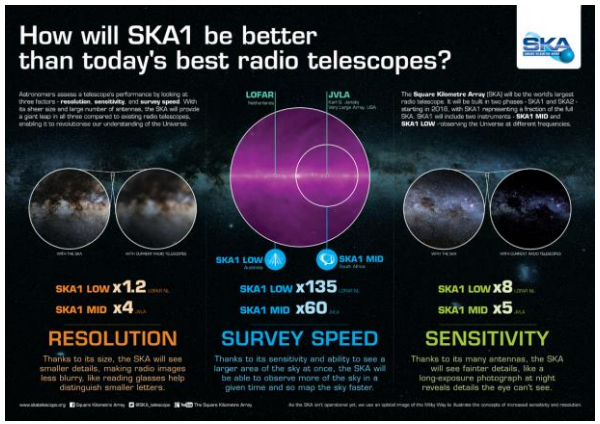
DEEP LEARNING



MACHINE LEARNING

POWERING THE EXASCALE ERA

Massive data growth

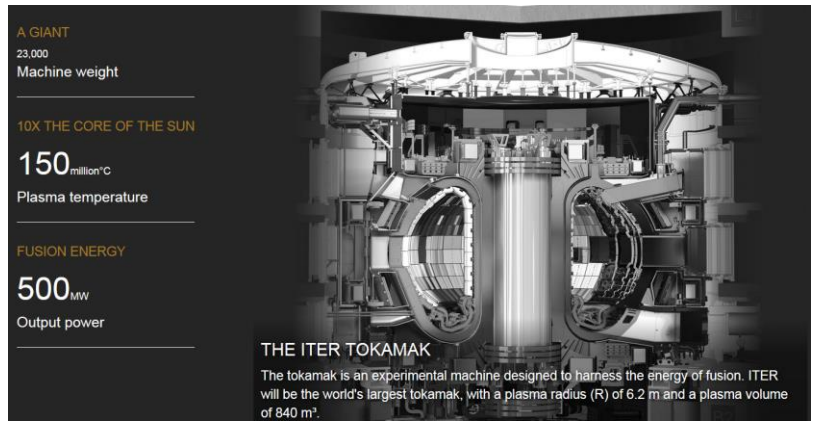


1EB/day

The SKA1 Square Kilometre Array radio telescope will generate more than an Exabyte of data every day.

10X

The CERN large Hadron collider's High Luminosity upgrade will result in a 10X increase in data volume.



30X

The 500 MW ITER fusion experiment will provide a 30X increase in output power over the largest previous experiment.

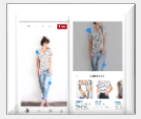
NVIDIA UNIVERSAL ACCELERATION PLATFORM

Single Platform Drives Utilization and Productivity

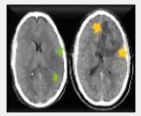
CUSTOMER USECASES


Speech

Translate


Recommender

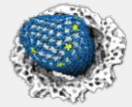
CONSUMER INTERNET

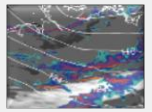

Healthcare

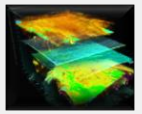

Manufacturing


Finance

INDUSTRIAL APPLICATIONS


Molecular Simulations


Weather Forecasting


Seismic Mapping

SCIENTIFIC APPLICATIONS

APPS & FRAMEWORKS

python™ RAPIDS

TensorFlow PYTORCH mxnet Chainer ONNX

Amber ANSYS NAMD SIMULIA +550 Applications

NVIDIA SDK & LIBRARIES

MACHINE LEARNING/ ANALYTICS

cuDF | cuML | cuGRAPH

DEEP LEARNING

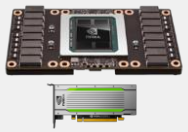
cuDNN | cuBLAS | CUTLASS | NCCL | TensorRT

HPC

CuBLAS | CuFFT | OpenACC

CUDA

TESLA GPUs & SYSTEMS


TESLA GPU


VIRTUAL GPU


NVIDIA DGX FAMILY


NVIDIA HGX

DELL Hewlett Packard Enterprise IBM
SYSTEM OEM

aws Google Cloud Platform Microsoft Azure
CLOUD

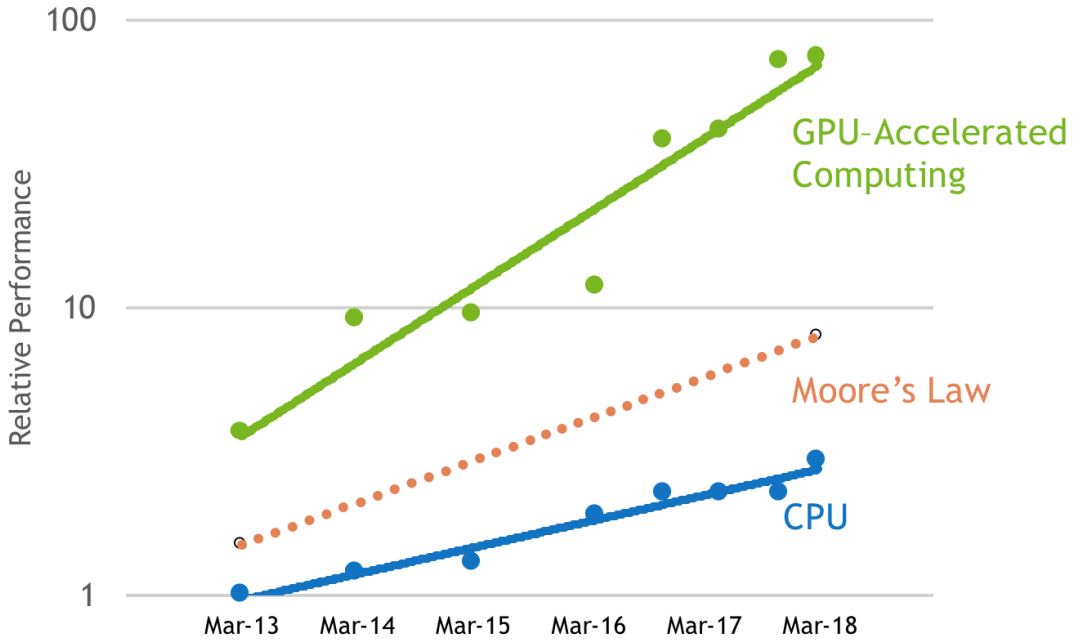
BEYOND MOORE'S LAW

Progress Of Stack In 5 Years

2013

- cuBLAS: 5.0
- cuFFT: 5.0
- cuRAND: 5.0
- cuSPARSE: 5.0
- NPP: 5.0
- Thrust: 1.5.3
- CUDA: 5.0
- Resource Mgr: r304
- Base OS: CentOS 6.2

Accelerated Server With Fermi



Measured performance of Amber, CHROMA, GTC, LAMMPS, MILC, NAMD, Quantum Espresso, SPECFEM3D

2018

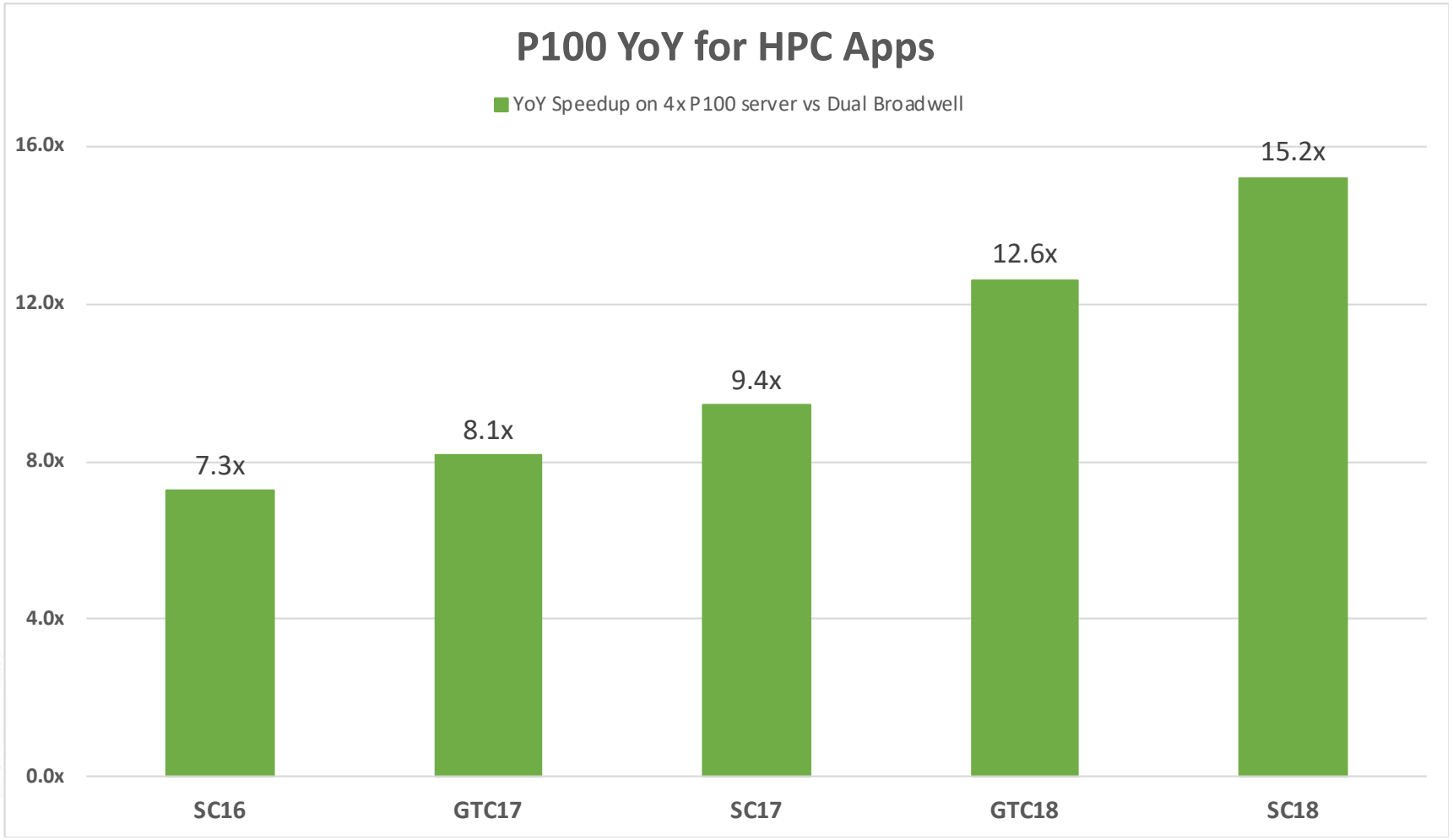
- cuBLAS: 10.0
- cuFFT: 10.0
- cuRAND: 10.0
- cuSOLVER: 10.0
- cuSPARSE: 10.0
- NPP: 10.0
- Thrust: 1.9.0
- CUDA: 10.0
- Resource Mgr: r384
- Base OS: Ubuntu 16.04

Accelerated Server with Volta

2X HIGHER VALUE IN 2 YEARS

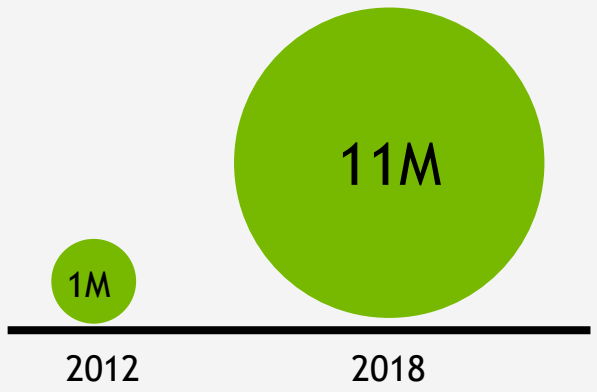
Same Hardware More Performance With Software Stack Optimizations

AMBER
Chroma
GROMACS
GTC
LAMMPS
MILC
NAMD
QE
RTM
SPECFEM3D
VASP

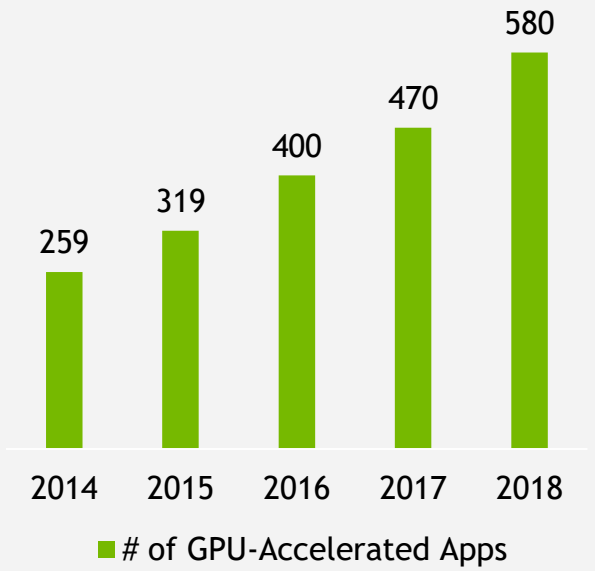


MOST ADOPTED PLATFORM FOR ACCELERATING HPC

11M CUDA Downloads



580 Applications Accelerated



127 Systems on Top 500

World's #1 Summit: 144 PF

World's #2 Sierra: 95 PF

Europe's #1 Piz Daint: 21 PF

Japan's #1 ABCI: 20 PF

Industrial #1 ENI: 12 PF

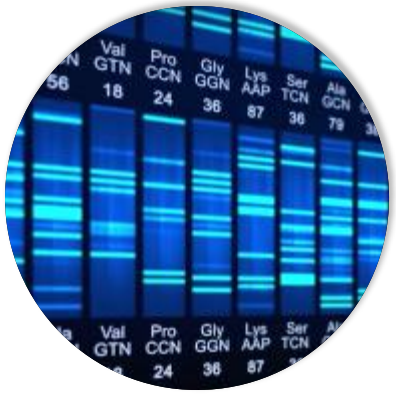
11X CUDA DOWNLOADS

ALL TOP 15 APPLICATIONS ACCELERATED

NEW HIGHS IN TOP 500 LIST

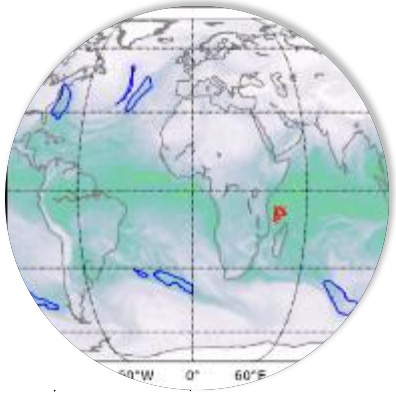
NVIDIA POWERS 5 OF 6 GORDON BELL NOMINATIONS

GPU Acceleration Critical To HPC At Scale Today



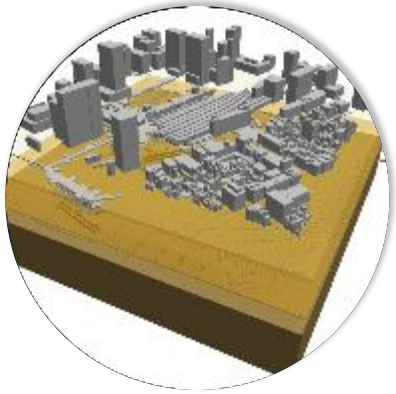
BERKELEY LAB · OAK RIDGE National Laboratory

Genomics
2.36 ExaOps



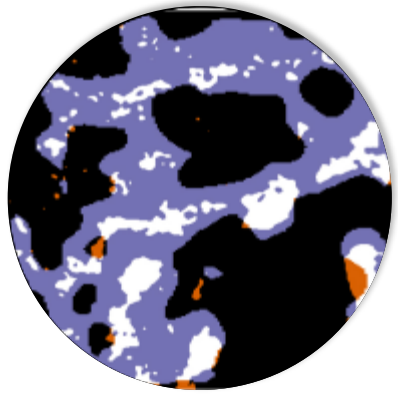
BERKELEY LAB · OAK RIDGE National Laboratory · NVIDIA

Weather
1.15 ExaOps



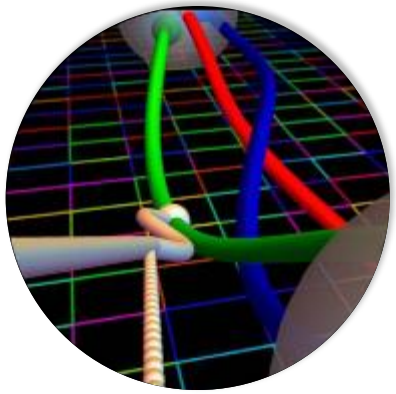
東京大学 · OAK RIDGE National Laboratory

Seismic
1st Soil & Structure Simulation



OAK RIDGE National Laboratory

Material Science
300X Higher Performance



BERKELEY LAB · Lawrence Livermore National Laboratory · NVIDIA

Quantum Chromodynamics
<1% of Uncertainty Margin

AI - A NEW INSTRUMENT FOR SCIENCE

HPC

> +40 years of algorithms based on first-principles theory.

> **P**roven statistical models for accurate results

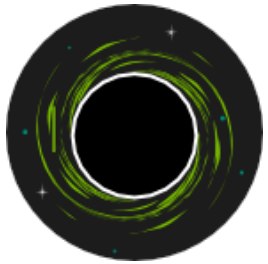
AI

> Improve predictive accuracy and faster response time.

> previously unmanageable data sets.

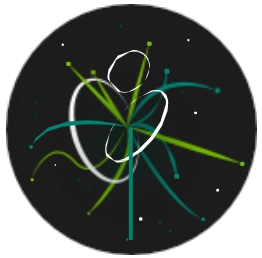
Dramatically Improves Accuracy and Time-to-Solution

Commercially viable fusion energy



Understanding cosmological dark energy and matter

Clinically viable precision medicine

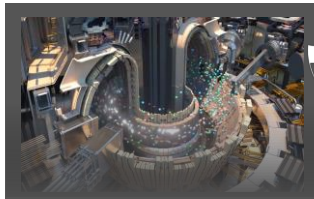


Improvement and validation of the Standard Model of Physics

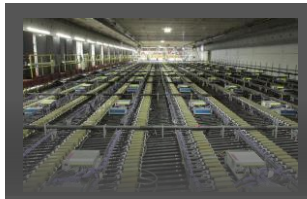
Climate/weather forecasts with ultra-high fidelity

AI FOR SCIENCE

Transformative Tool To Accelerate The Pace of Scientific Innovation

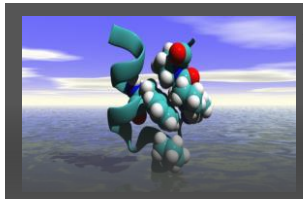


90% accuracy
Fusion Sustainment
Clean Energy

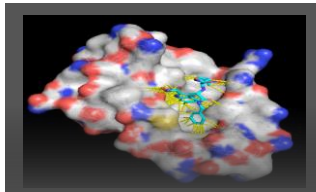


33% better detection rate
Track Neutrinos
Particle Physics

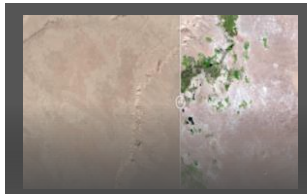
5,000X Faster
Process LIGO Signal
Understanding Universe



300,000X Faster
Predict Molecular Energetics
Drug Discovery

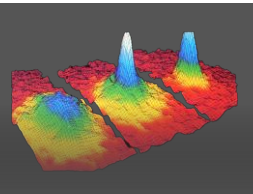


70% accuracy
Score Protein Ligand
Drug Discovery



11% higher accuracy
Monitor Earth's Vital
Climate

Weeks to 10 milliseconds
Analyze Gravitational Lensing
Astrophysics



14X Faster
Generate Bose-Einstein
Condensate (Physics)

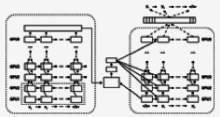
IMPROVES ACCURACY
Enabling realization of full scientific potential

ACCELERATES TIME TO SOLUTION
Unlocking the use of science in exciting new ways

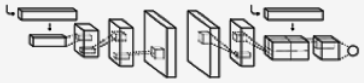
MOST ADOPTED PLATFORM FOR ACCELERATING AI

Convolutional Networks

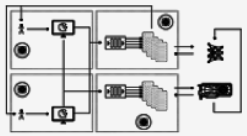
Recurrent Networks



Generative Adversarial Networks



Reinforcement Learning



BROADEST ARRAY OF NETWORKS

Chainer

mxnet

ONNX

PaddlePaddle

PYTORCH

TensorFlow

EVERY DEEP LEARNING FRAMEWORK ACCELERATED

Alibaba Cloud aliyun.com

aws

Google Cloud

IBM Cloud

Microsoft Azure

Tencent Cloud

Cloud Services

DELL

Hewlett Packard Enterprise

IBM

inspur

Lenovo

SUPERMICRO

Systems

Desktops

AVAILABLE EVERYWHERE

https://mlperf.org/

Supporting companies

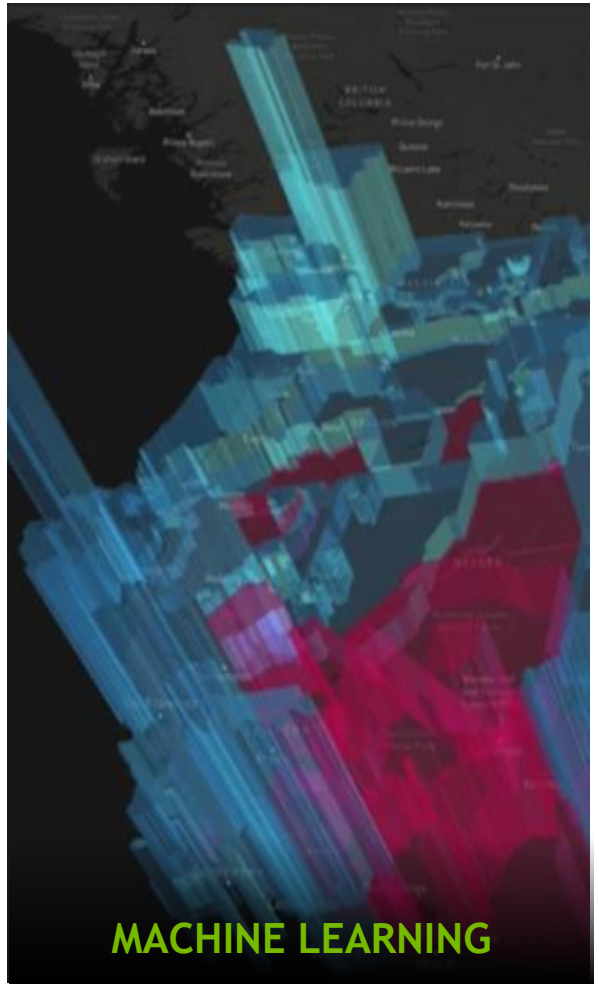
Alibaba, AMD, Arm, Baidu, Cadence, Cerebras, Cisco, Cray, Dividiti, Enflame Tech, Esperanto, Google, Groq, Huawei, Intel, MediaTek, Mentor Graphics, Mythic, NetApp, NVIDIA, One Convergence, Rpa2ai, Sambanova, Samsung S.LSI, Sigopt, Synopsys, Tensyr, Wave Computing

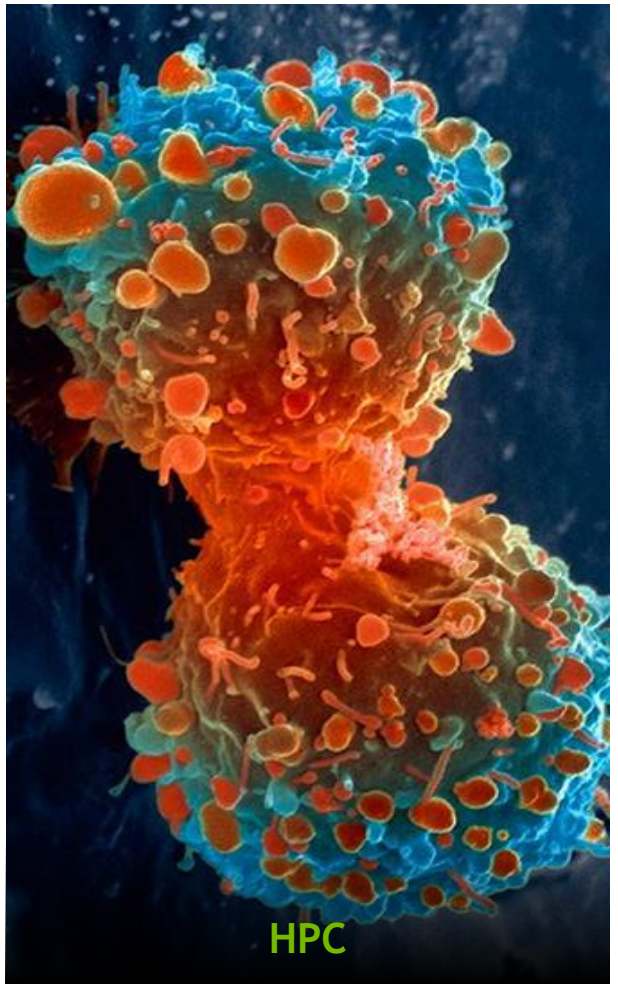
Contributions by researchers from

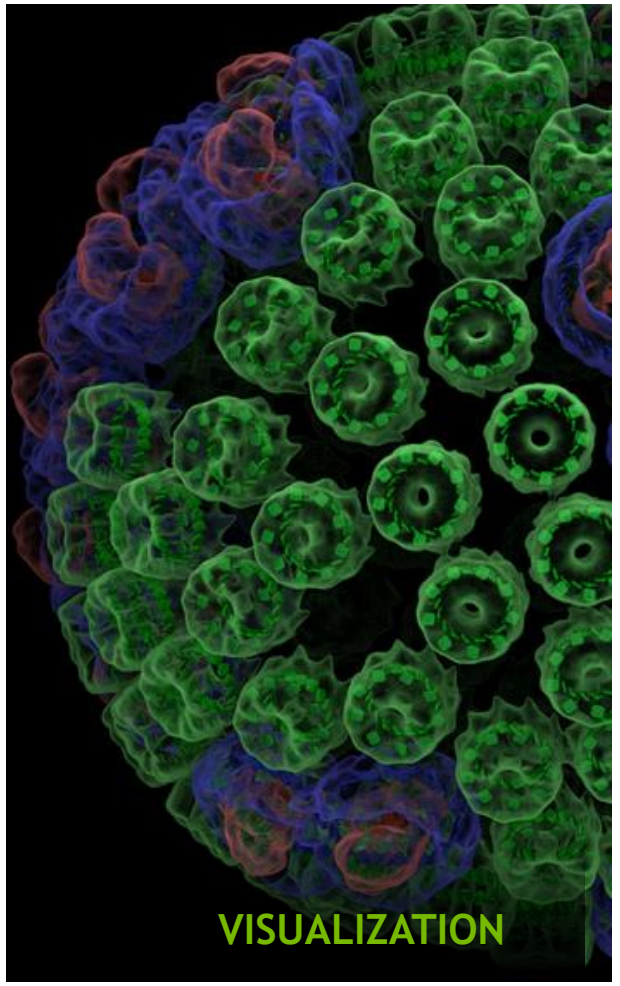
Harvard University, Stanford University, University of Arkansas, Littlerock, University of California, Berkeley, University of Minnesota, University of Texas, Austin, University of Toronto

First Industry Benchmark for Measuring AI Performance

NVIDIA GPU CLOUD CONTAINER REGISTRY







NGC: THE DESTINATION FOR GPU-ACCELERATED SOFTWARE

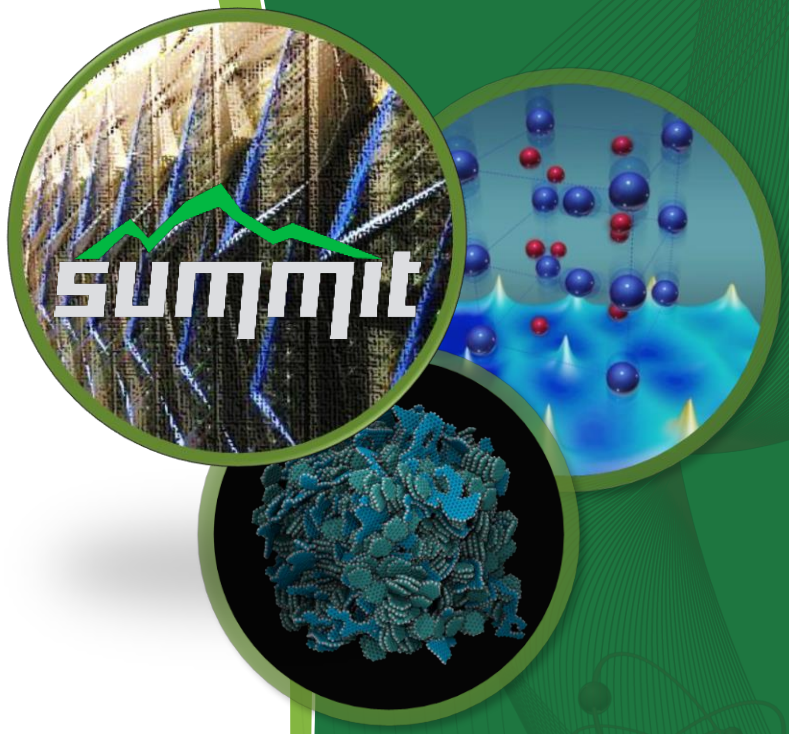
Deep Learning	HPC	HPC Visualization	Analytics
caffe	bigDFT	IndeX*	Kinetica
Caffe2	CANDLE	ParaView*	MapD
chainer	CHROMA	ParaView-holodeck	
CNTK	GAMESS	Paraview-IndeX*	
CUDA	GROMACS	Paraview-OptiX	**KUBERNETES**
deep learning studio	LAMMPS*	VMD	Kubernetes on NVIDIA GPUs
Digits	Lattice-Microbes		
H20.ai	MATLAB		**MACHINE LEARNING**
inferenceserver	MILC*		RAPIDS
MXNet	NAMD*		
PaddlePaddle	PGI-compilers		
PyTorch	PIConGPU		
TensorFlow	QMCPACK*		
TensorRT	RELION		
TensorRT Server			
Theano			
Torch			

*Multi-node HPC containers @ SC18
New since GTC18 in San Jose

Summit Update

Fred Allman

January 2019



ORNL is managed by UT-Battelle
for the US Department of Energy

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

OAK RIDGE National Laboratory

ORNL Summit System Overview

System Performance

- Peak of 200 Petaflops (FP_{64}) for modeling & simulation
- Peak of 3.3 ExaOps (FP_{16}) for data analytics and artificial intelligence

The system includes

- 4,608 nodes
- Dual-rail Mellanox EDR InfiniBand network
- 250 PB IBM file system transferring data at 2.5 TB/s

Each node has

- 2 IBM POWER9 processors
- 6 NVIDIA Tesla V100 GPUs
- 608 GB of fast memory
  (96 GB HBM2 + 512 GB DDR4)
- 1.6 TB of non-volatile memory

What makes Summit so powerful?

Summit's architecture is a sweet spot that has broad capability across:

- Traditional HPC modeling and simulation
- High performance data analytics
- Artificial Intelligence

- Powerful CPUs for scalar operations and data analytics
- Accelerators that address high-performance arithmetic in FP_{64}, FP_{32}, and FP_{16} precision
- Exceptional performance on artificial intelligence and machine learning
- High speed interconnect with switch based collective operations
- High-performance file system

Summit Contains 27,648 NVIDIA Tesla V100s

Each Tesla v100 GPU has:

- 300 GB/s total BW (**NVLink** v2.0)
- 5,120 CUDA cores (64 on each of 80 SMs)
- 640 **Tensor cores** (8 on each of 80 SMs)
- 20MB Registers | 16MB Cache | 16GB HBM2 @ 900 GB/s
- 7.5 DP TFLOPS | 15 SP TFLOPS | 120 FP_{16} TFLOPS

- Tensor cores do mixed precision multiply-add of 4x4 matrices

$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

FP16 or FP32 | FP16 | FP16 | FP16 or FP32

$$D = AB + C$$

Type	Size	Range	$u = 2^{-t}$
half	16 bits	$10^{\pm 5}$	$2^{-11} \approx 4.9 \times 10^{-4}$
single	32 bits	$10^{\pm 38}$	$2^{-24} \approx 6.0 \times 10^{-8}$
double	64 bits	$10^{\pm 308}$	$2^{-53} \approx 1.1 \times 10^{-16}$
quadruple	128 bits	$10^{\pm 4932}$	$2^{-113} \approx 9.6 \times 10^{-35}$

- The Modeling & Simulation community can benefit from better utilizing mixed / reduced precision
- Eg: Possible to achieve 4x FP64 peak for 64bit LU on V100 with iterative mixed precision (Dongarra et al.)

How is Summit Architecture different from Titan?

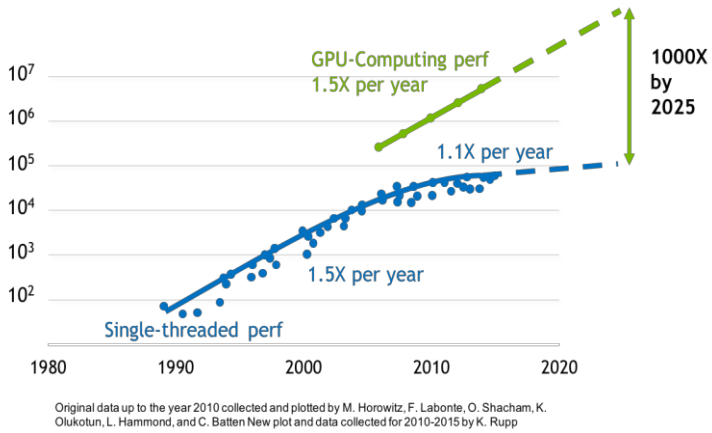
- Many fewer nodes
- Much more powerful nodes
- Much more memory per node and higher memory bandwidth
- Faster interconnect
- Much higher bandwidth between CPUs and GPUs
- Much larger and faster file system
- 7x more performance for only slightly more power
  (HPL 122 PF run was 8.8 MW)

Feature	Titan	Summit
Peak $FLOPS_{64}$	27 PF	200 PF
Max possible Power	9 MW	13 MW
Number of Nodes	18,688	4,608
Node performance	1.4 TF	42 TF
Memory per Node	32 GB DDR3 + 6 GB GDDR5	512 GB DDR4 + 96 GB HBM2
NV memory per Node	0	1.6 TB
Total System Memory	0.7 PB	2.8 PB + 7.4 PB NVM
System Interconnect	Gemini (6.4 GB/s)	Dual Rail EDR-IB (25 GB/s)
Interconnect Topology	3D Torus	Non-blocking Fat Tree
Bi-Section Bandwidth	15.6 TB/s	115.2 TB/s
Processors on node	1 AMD Opteron™ 1 NVIDIA Kepler™	2 IBM POWER9™ 6 NVIDIA Volta™
File System	32 PB, 1 TB/s, Lustre®	250 PB, 2.5 TB/s, GPFS™

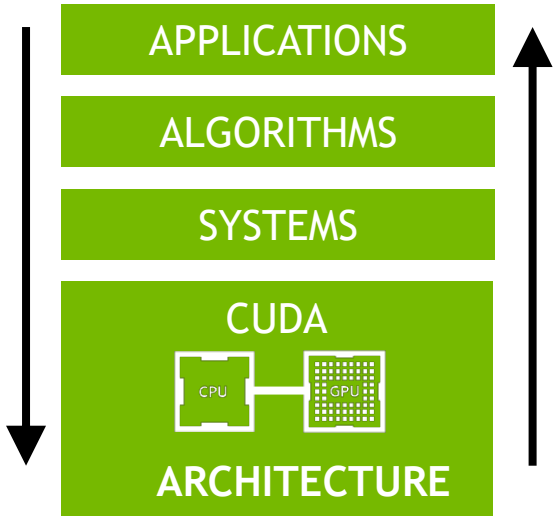
Summit Excels Across Simulation, Analytics, AI

- Data analytics – CoMet bioinformatics application for comparative genomics. Used to find sets of genes that are related to a trait or disease in a population. Exploits cuBLAS and Volta tensor cores to solve this problem 5 orders of magnitude faster than previous state-of-art code.
  - **Has achieved 2.36 ExaOps** mixed precision (FP_{16}-FP_{32}) on Summit
- Deep Learning – global climate simulations use a half-precision version of the DeepLabv3+ neural network to learn to detect extreme weather patterns in the output
  - **Has achieved a sustained throughput of 1.0 ExaOps (FP_{16})** on Summit
- Nonlinear dynamic low-order unstructured finite-element solver accelerated using mixed precision (FP_{16} thru FP_{64}) and AI generated preconditioner. Answer in FP_{64}
  - **Has achieved 25.3 fold speedup** on Japan earthquake – city structures simulation
- **Half-dozen Early Science codes are reporting >25x speedup on Summit vs. Titan**

CONCLUSION



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten New plot and data collected for 2010-2015 by K. Rupp



EXASCALE REQUIRED

HPC REQUIRES A FULL STACK SOLUTION

EXASCALE DELIVERED


NVIDIA

SUMMIT SCIENCE

Fred Allman

January 2019

Five Gordon Bell Finalists Credit Summit Supercomputer

The finalists—representing Oak Ridge, Lawrence Berkeley, and Lawrence Livermore National Laboratories and the University of Tokyo—leveraged Summit's unprecedented computational capabilities to tackle a broad range of science challenges and produced innovations in machine learning, data science, and traditional modeling and simulation to maximize application performance. The Gordon Bell Prize winner will be announced Thursday in the 12:45 awards session. Finalists include:

- An ORNL team led by computational systems biologist Dan Jacobson and OLCF computational scientist Wayne Joubert that developed a genomics algorithm capable of using mixed-precision arithmetic to attain exascale speeds.
- A team from the University of Tokyo led by associate professor Tsuyoshi Ichimura that applied AI and mixed-precision arithmetic to accelerate the simulation of earthquake physics in urban environments.
- A Lawrence Berkeley National Laboratory-led collaboration that trained a deep neural network to identify extreme weather patterns from high-resolution climate simulations.
- An ORNL team led by data scientist Robert Patton that scaled a deep learning technique on Summit to produce intelligent software that can automatically identify materials' atomic-level information from electron microscopy data.
- A LBNL and Lawrence Livermore National Laboratory team led by physicists André Walker-Loud and Pavlos Vranas that developed improved algorithms to help scientists predict the lifetime of neutrons and answer fundamental questions about the universe.

Genomics Code Exceeds Exaops on Summit Supercomputer

ORNL researchers leverage GPU tensor cores to deliver unprecedented performance, winning Gordon Bell prize

The Science

Using Summit, ORNL researchers broke the exascale barrier, achieving a peak throughput of 2.36 exaops—faster than any previously reported science application—while analyzing genomic data. The feat, orchestrated by OLCF computational scientist and lead code developer Wayne Joubert, was accomplished using a mixture of numerical precisions supported by Summit's GPU tensor cores. The algorithm deployed in this demonstration, called Combinatorial Metrics (CoMet), specializes in analyzing genomes and comparing gene variations within a given population.

The Impact

Exascale-level performance allows researchers to analyze datasets composed of millions of genomes—a size that was previously impossible—and study variations among all possible combinations of two or three alleles at a time. Scientists can use this information to uncover hidden networks of genes in plants and animals that contribute to observable traits, such as biomarkers for drought resistance in plants or disease in humans.

ORNL computational biologist Dan Jacobson led the team that broke the exascale barrier on the Summit supercomputer using mixed numerical precisions. Jacobson is pursuing projects related to human and plant systems biology.

PI(s)/Facility Lead(s): Daniel Jacobson
ASCR Program/Facility: Summit Early Science/OLCF
ASCR PM: Christine Chalk
Publication(s) for this work: TBD

https://www.olcf.ornl.gov/2018/06/08/genomics-code-exceeds-exaops-on-summit-supercomputer/

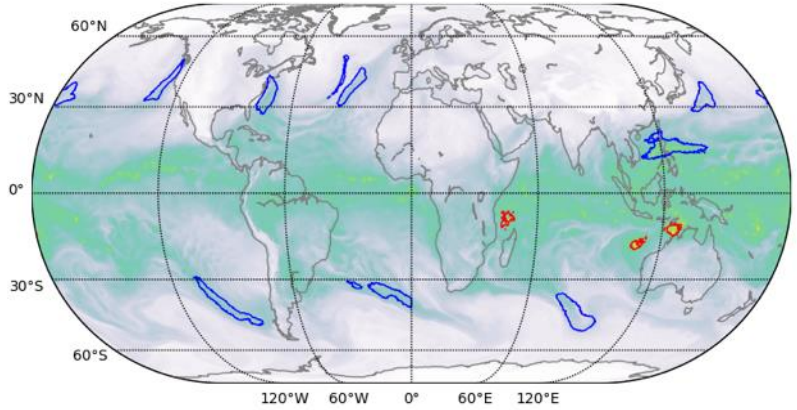
ORNL and Berkeley Lab Share 2018 Gordon Bell Prize

Identifying Extreme Weather Patterns at Exascale Speeds

LBNL-led team breaks deep-learning exascale barrier, solves segmentation problems in climate science

The Science

A team from Lawrence Berkeley National Laboratory led by data scientist Prahbat used the OLCF's Summit supercomputer to train a deep neural network to identify extreme weather patterns from high-resolution climate simulations. By tapping into Summit's multiprecision capabilities, the researchers achieved a peak performance of 1.13 exaops and a sustained performance of 0.999 exaops—the fastest deep-learning algorithm reported to date. Traditional image segmentation tasks work on three-channel red/blue/green images. But scientific datasets often comprise many channels; in climate, for example, these can include temperature, wind speeds, pressure values, and humidity. Running the optimized neural network on Summit allows the team to use all 16 channels and dramatically improve accuracy.



High-quality segmentation results produced by deep learning on climate datasets.

The Impact

Extracting pixel-level classifications of extreme weather patterns could aid in the prediction of how extreme weather events are changing as the climate warms. Though the team applied its work to climate science, many of its innovations, such as pattern-recognition algorithms, high-speed parallel data staging, an optimized data ingestion pipeline, and multichannel segmentation, lay the groundwork for future exascale deep-learning applications.

PI(s)/Facility Lead(s): Prahbat
ASCR Program/Facility: OLCF, NERSC
ASCR PM: Christine Chalk
Publication(s) for this work: Thorsten Kurth, et al. "Exascale Deep Learning for Climate Analysis," *Proceedings of SC18* (2018). ***Submitted.***

https://www.olcf.ornl.gov/2018/10/10/berkeley-lab-oak-ridge-nvidia-team-breaks-exaop-barrier-with-deep-learning-application/

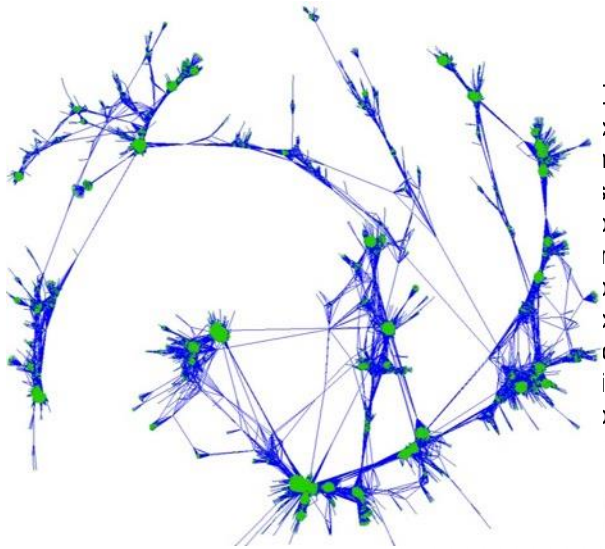
ORNL and Berkeley Lab Share 2018 Gordon Bell Prize

Computing Genes to Support Living Cleanly

ORNL researchers recognized as Gordon Bell Finalists for breakthrough genomic data science

The Science

An ORNL team led by computational systems biologist Dan Jacobson and computational scientist Wayne Joubert developed a genomics algorithm capable of using mixed-precision arithmetic to attain a speedup of more than 20,000-fold over the previous state of the art. On Summit, the team's Combinatorial Metrics application achieved a peak throughput of 2.36 exaops—or 2.36 billion billion calculations per second, the fastest science application ever reported. Jacobson's work compares genetic variations within a population to uncover hidden networks of genes that contribute to complex traits, including diseases. One condition Jacobson's team is studying is opioid addiction, which was linked to the overdose deaths of more than 49,000 people in the United States in 2017.



One component of a correlation network mapping variations in single nucleotides that occur at the same location in the genome across a population. These correlations can be used to identify genetic markers linked to complex observable traits.

The Impact

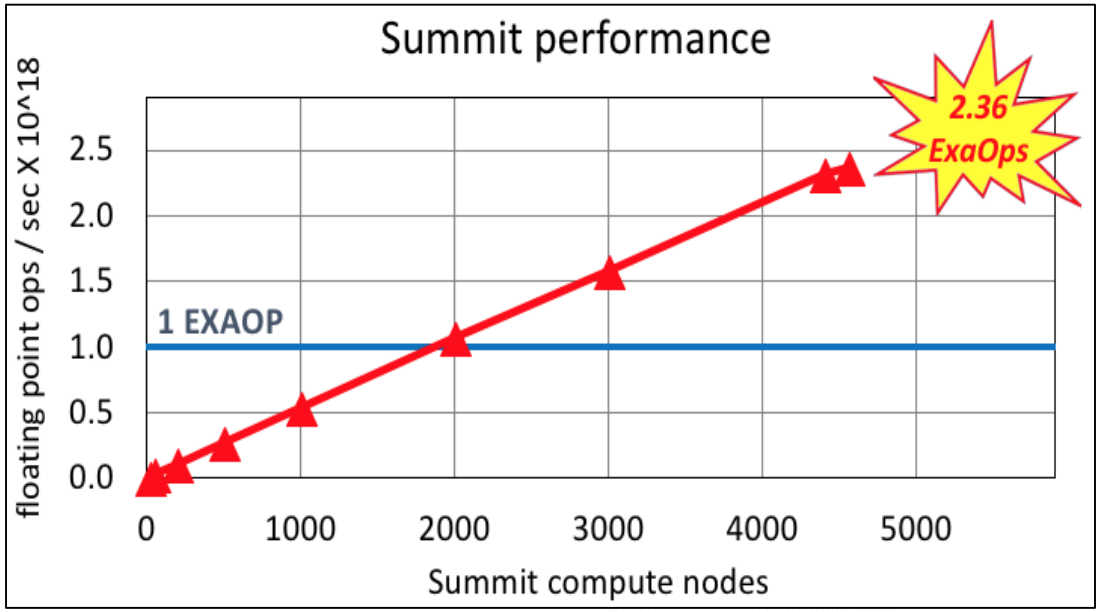
Exascale-level performance allows researchers to analyze datasets composed of millions of genomes—a problem size that was previously intractable. Combining clinical and genomic data with machine learning and Summit's advanced architecture will allow researchers to gain new insight into things like the genetic factors that contribute to conditions such as cardiovascular disease, prostate cancer, Alzheimer's disease, and opioid addiction. This kind of knowledge can inform medical treatment and improve patient outcomes.

PI(s)/Facility Lead(s): Dan Jacobson
ASCR Program/Facility: OLCF
ASCR PM: Christine Chalk
Publication(s) for this work: Wayne Joubert, et al. "Attacking the Opioid Epidemic: Determining the Epistatic and Pleiotropic Genetic Architectures for Chronic Pain and Opioid Addiction," *Proceedings of SC18* (2018). ***Submitted.***

https://www.olcf.ornl.gov/2018/10/15/computing-genes-to-support-living-clean/

Summit Science: Deep Learning for Human Systems Biology

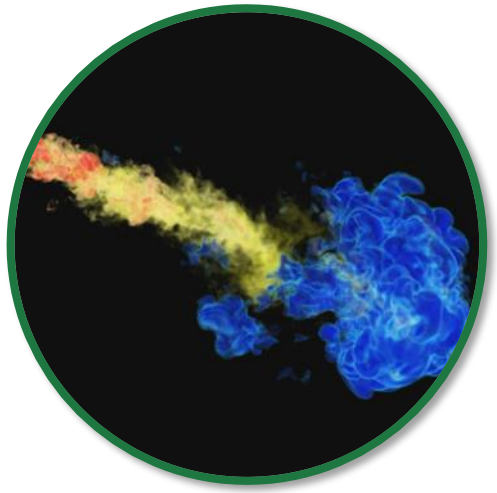
- Each vector containing 2-bit values is converted into two columns representing the number of 0s and 1s in each element, forming new matrix V'
- Applying dense matrix-matrix product to calculate $V'^{T} V'$ generates all vector-vector correlation tables
- Use cuBlasGemmEx
- Input values are FP16
- Results are computed and stored as FP32



Performance

- Achieved **2.36 ExaOps** (mixed precision ExaFlops) at 4,560 nodes (99% of Summit) using the Tensor Cores – first reported application to reach ExaOp
- Equivalent to **86.4 TF** per GPU for the whole computation (including communications and transfers) at 4,560 nodes
- Excellent scaling made possible by Summit fat tree network with adaptive routing
- **> 4X faster** than original bitwise (non-flop) implementation on GPUs (= 4X more science possible)

Summit Science: Combustion



Joe Oefelein
Georgia Tech

Ramanan Sankaran
Oak Ridge National Laboratory

RAPTOR

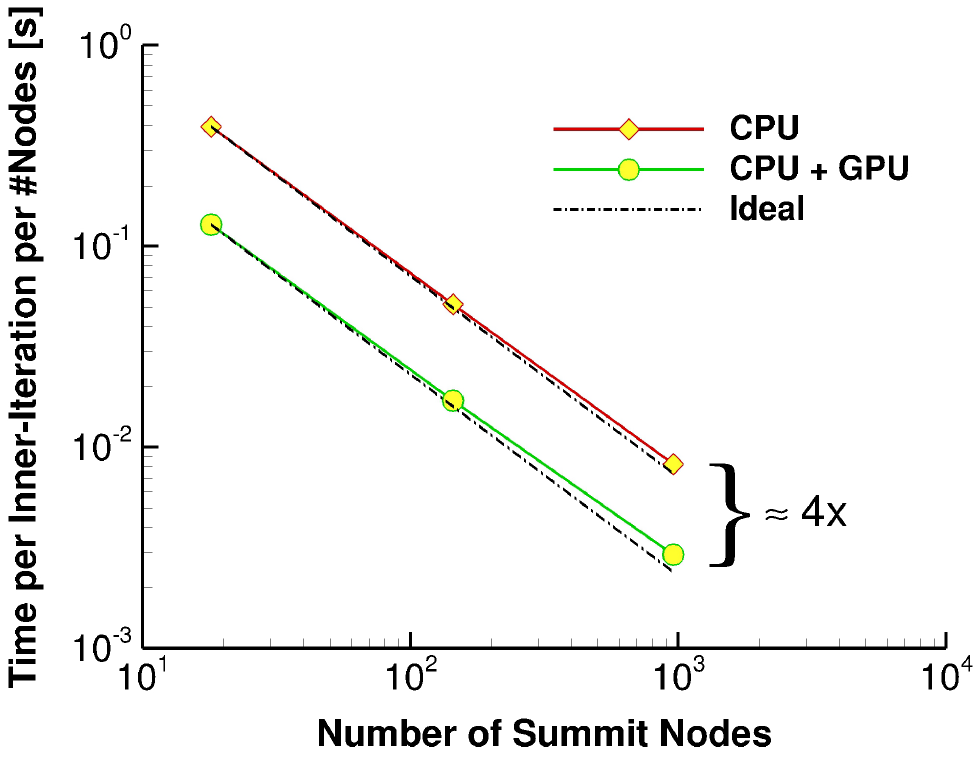
- Our focus is turbulent reacting flows in propulsion and power systems-or how combustion takes place in turbulent environments such as gas turbines or car engines.
- Our code, RAPTOR, is a fluid dynamics solver that incorporates the multiscale and multiphysics processes at play in these systems.
- Because of the sheer size of the machine, Summit will allow us to work on problems that we haven't been able to do before.
- Summit will speed our time-to-solution, enabling us to carry out hundreds of runs with RAPTOR in a short time. This will enhance the information we get from our solutions by facilitating detailed parametric studies across a wide range of operating conditions.
- Results from RAPTOR on Summit will enable predictive simulations of advanced concepts and will accelerate the design cycle of internal combustion engines and gas turbines.

Summit Science: Combustion

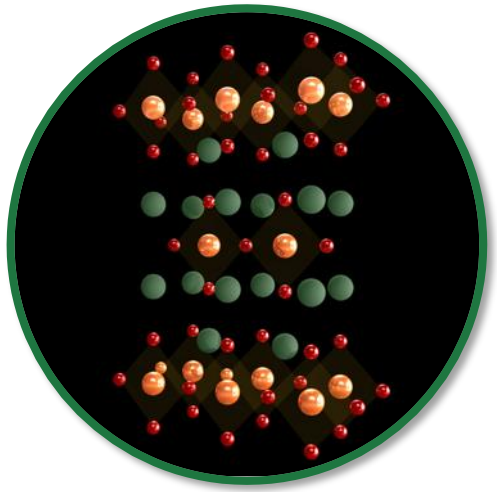
- Physics models that are computationally intensive are externalized as libraries
  - New accelerated version of the libraries implemented in templated C++ using Kokkos programming model
  - Performance portable through use of multiple backends (Cuda, OpenMP etc.)
  - Tests developed to verify correctness against the original implementation
  - Interfaces developed to invoke C++ library from original code and exchange data
- Main flow solver and other physics models, that are not rewritten, are accelerated through a hybrid MPI+OpenMP programming model
- Performance portability is emphasized in both Kokkos and the MPI+OpenMP developments
- RAPTOR uses the GPU accelerators on Summit for a significant fraction of the computation

Early Results on Summit

Weak scaling attributes of the hybrid-MPI+OpenMP version of RAPTOR on Summit



Summit Science: Materials



Paul Kent
Oak Ridge National Laboratory

Andreas Tillack
Oak Ridge National Laboratory

Ying Wai Li
Oak Ridge National Laboratory

QMCPACK

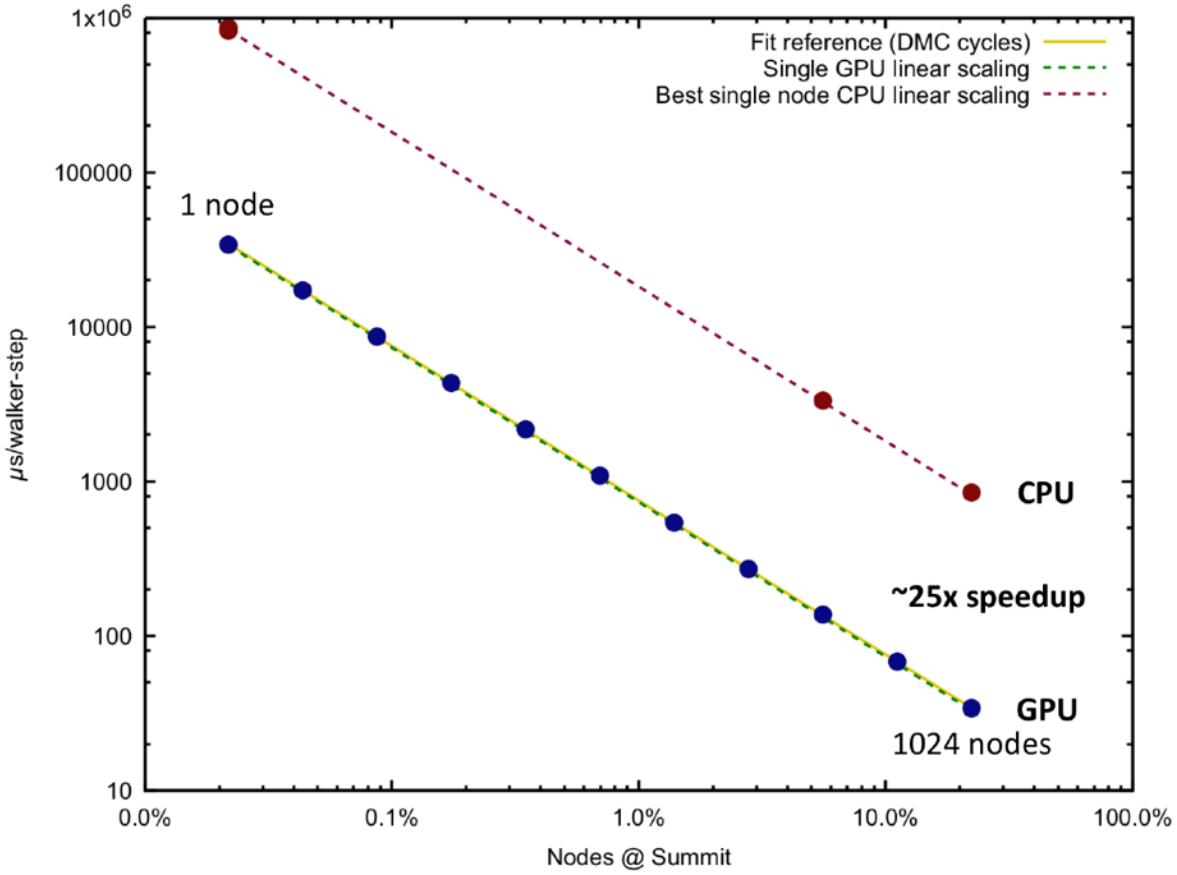
- QMCPACK is an open source, ab initio quantum Monte Carlo code for the study and prediction of materials properties by stochastically solving quantum many-body problems using variational MC and more accurate but computationally demanding diffusion MC.
- It allows for highly accurate, predictive ab initio calculations of larger and more complex systems that are limited or inaccessible by conventional methods such as density functional theory (DFT).
- Summit will let us study many new materials, but one famous class of materials that we're interested in simulating is high-temperature superconductors, which conduct energy without any loss of efficiency. It would be a breakthrough if we could explain those.
- The large on-node memory is very important for increasing the range and complexity of materials and physical phenomena that we can study.

Summit Science: Materials

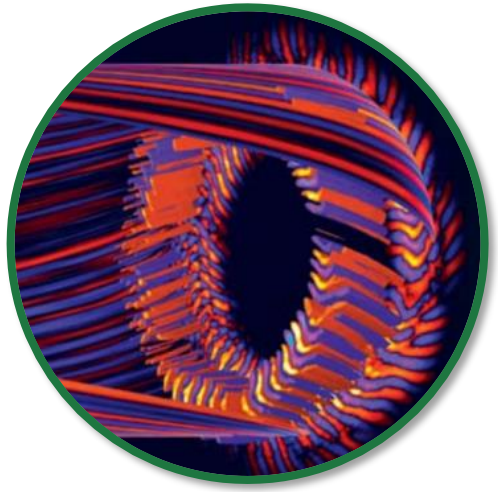
- Implementation of complex wavefunctions on GPUs to handle general twist boundary conditions. This added functionality enables most standard QMC calculations to use GPUs for acceleration.
- Development and implementation of a new Monte Carlo update scheme, the rank-k delayed updates algorithm, to increase compute intensity. The multiple, sequential BLAS-2 matrix operations are fused into a single BLAS-3 operation.
- Implementation that enables a more general combination of the number of MPI ranks and GPUs on a node.
- Investigation and implementation of distributed read-only data ("spline table") over multiple GPUs, as well as mutual access of GPU memory among MPI ranks and GPUs, within a node. This lifted the on-chip memory limitations of a GPU, enabling more memory intensive calculations.
- Investigation of using task-based programming techniques to improve parallelism on GPUs.

Early Results on Summit

Scaling plot of NiO 256 atom cell runs on up to 1024 nodes of Summit.



Summit Science: Plasma Simulations Supporting ITER



Zhihong Lin
University of California, Irvine

Wayne Joubert
Oak Ridge National Laboratory

GTC

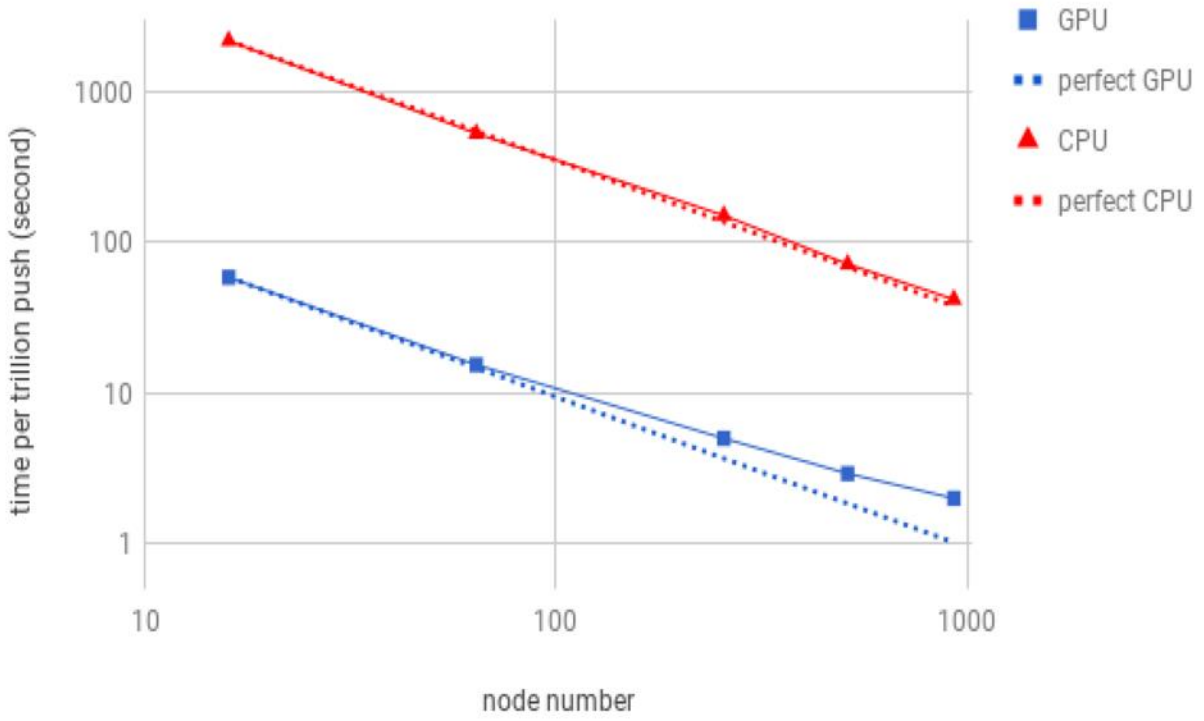
- Plasma simulations supporting the ITER project are a key DOE/FES focus and are required to understand the complex kinetic dynamics governing magnetic confinement properties of fusion-grade plasmas.
- The Gyrokinetic Toroidal Code (GTC) is a massively parallel particle-in-cell code for first-principles, integrated simulations of burning plasma experiments such as the International Thermonuclear Experimental Reactor (ITER), the crucial next step in the quest for the fusion energy. GTC solves the five-dimensional (5D) gyrokinetic equation in full, global torus geometry to address kinetic turbulence issues in magnetically-confined fusion tokamaks.
- On Summit, the GTC has worked to develop the kinetic capability for first-principles-based direct numerical simulations of key instabilities that limit the burning plasma performance and threaten device integrity in magnetically-confined fusion systems.
- The GTC particle-in-cell (PIC) algorithm is the most computationally dominant component of the GTC code. As a sequel to previous work, a large part of the project's performance optimization work is focused on efficient multithreading of this computation for Summit. The particle PUSH and SHIFT operations are the two most dominant operations of the PIC computation, which are targets for acceleration on Summit.

Summit Science: Plasma Simulations Supporting ITER

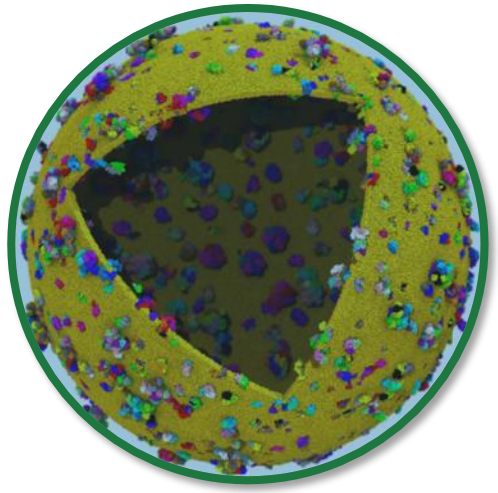
- GTC is a gyrokinetic toroidal fusion code for modeling fusion reactors
- Much of the code already used GPUs effectively on Titan, e.g., particle push, using OpenACC
- Additional development work was done to optimize for Summit
- Code uses the NVIDIA AmgX solver to improve performance over the previous PETSc field solver

Early Results on Summit

Wall-clock time for one trillion particle pushes in the GTC weak scaling test on Summit



Summit Science: Computational Biophysics



Jim Phillips
University of Illinois Urbana-Champaign

NAMD

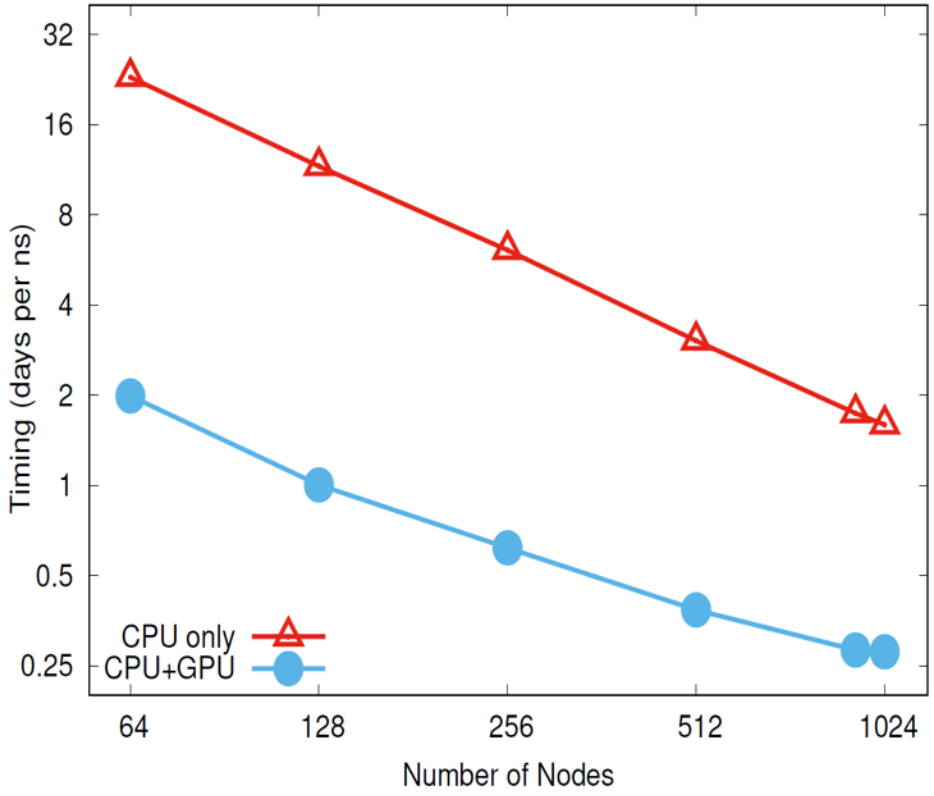
- DOE/BER funded programs are using molecular dynamics simulations to acquire a molecular level understanding of the relationships between biomass structure and recalcitrance.
- NAMD is a high performance molecular dynamics code that is capable of massively parallel simulations, and it will be used to study the molecular level neural mechanisms of cellular and neural signaling. The proposed research requires the computational power of Summit and significant modifications of NAMD will be required in order to exploit Summit's computational capabilities. NAMD has a large worldwide user base and these modifications will also greatly benefit the larger computational biophysical community.
- The BRAIN (Brain Research through Advancing Innovative Neurotechnologies) initiative is one of the Administration's "Grand Challenges". This research will elucidate molecular details of neural dynamics, synapse dynamics, and neural to synapse dynamics.

Summit Science: Computational Biophysics

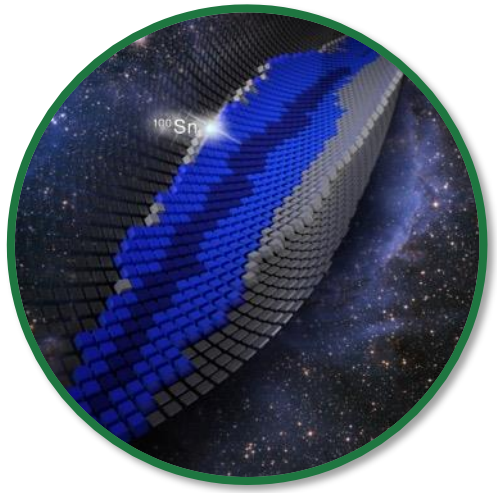
- Major challenge is time to solution on small problems (strong scaling)
- Multiple levels of parallelism:
  - Within GPU work is parallelized to thousands of threads and to overlapping GPU kernels
  - Within node, work is split to among GPUs and CPUs
  - System is divided spatially into groups of atoms that are distributed to the nodes
- Atoms are split on to nodes and only atom coordinates and forces that are needed are communicated from other nodes
- C++ using Charm++ parallel library for thread and node parallelism. CUDA C is used for implementing the GPU kernels
- Current GPU implementation does not prevent other hardware implementations
- Direct GPU-GPU communication (both NVLINK and over IB)

Early Results on Summit

NAMD strong scaling performance for a one billion atom HIV capsid proto-cell simulation on Summit.



Summit Science: Building Blocks of Matter



Gaute Hagen
Oak Ridge National Laboratory

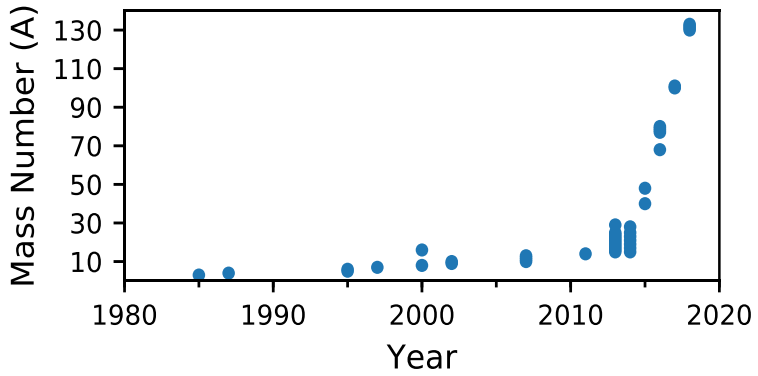
Gustav Jansen
Oak Ridge National Laboratory

NUCCOR

- Nuclear physics theory and computation is central to the DOE/NP mission of improving our understanding of the building blocks of matter, discovering the origins of nuclei, and identifying the forces that transform matter.
- NUCCOR is an application for the computation of the structure and reactions of atomic nuclei, implementing a set of algorithms that solve the quantum mechanical nuclear many-body problem using state-of-the-art nuclear interactions and currents. These include Hartree Fock, Coupled Cluster, and Equation of Motion methods.
- An highly optimized NUCCOR code on Summit will impact the field of low-energy nuclear physics through enabling benchmarks and quality standards for neutrinoless double-beta decay, nuclear structure calculations of experimentally relevant nuclei for guiding, interpreting and predicting experimental research, and enabling nuclear structure and reaction of nuclei and their behaviors with previously unattainable detail.

Summit Science: Building Blocks of Matter

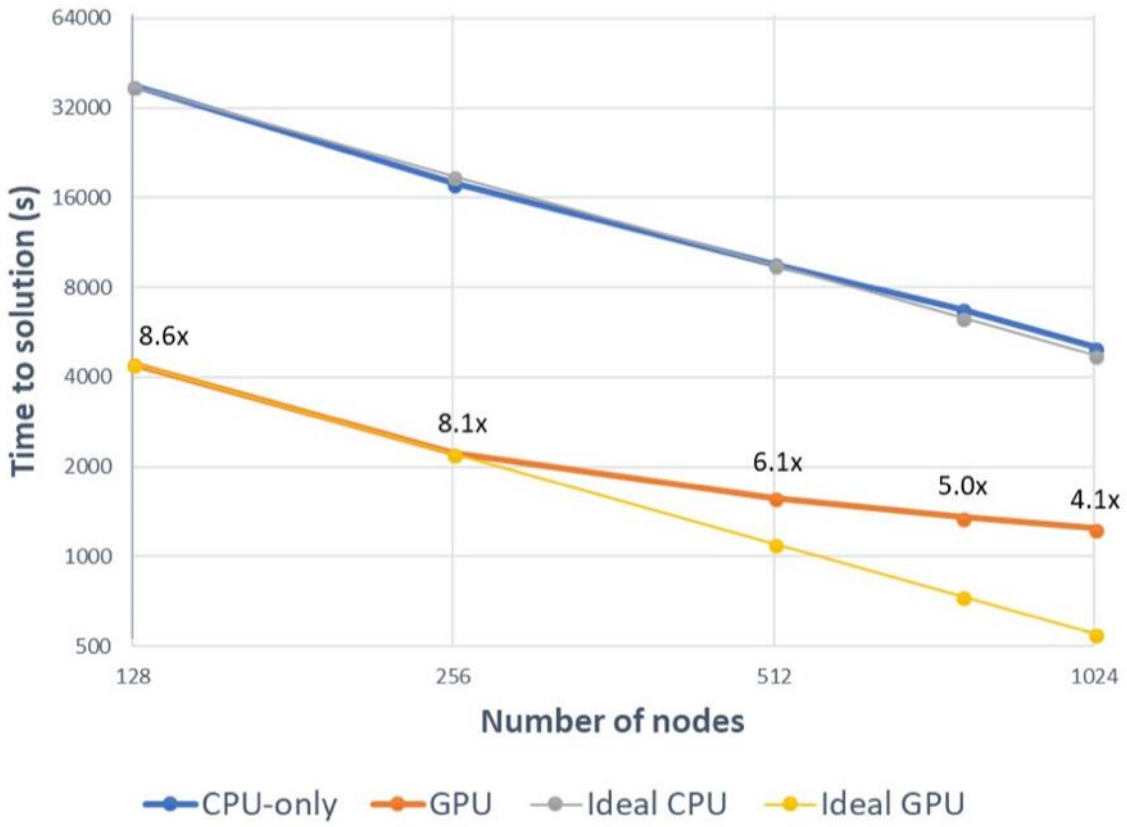
- It is now possible to describe larger nuclei to a higher degree of precision from first principles



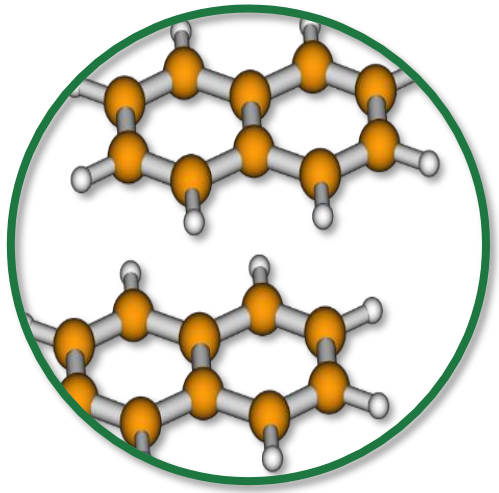
- The spherical tensor contractions at the heart of NUCCOR has been pushed to external libraries to allow multiple versions for different programming models
  - MPI + OpenMP for CPU-only usage
  - MPI + OpenMP + CUDA for GPU accelerated kernels
- Object-oriented design to allow selecting of computational kernels at run-time for easy testing and instrumentation, while retaining the possibility of selecting kernels for optimal performance
- Testing framework to allow unit, integration and regression testing currently covering over 70% of the new libraries

Early Results on Summit

Strong scaling of a full application run to compute a state in an A = 48 system, like ^{48}Ca or ^{48}Ti needed in this project, using up to 1024 nodes on Summit for both the CPU-only version and the GPU version.



Summit Science: Computational Chemistry



Remco Havenith
University of Groningen

Tjerk Straatsma
Oak Ridge National Laboratory

GronOR

- GronOR is a non-orthogonal configuration interaction application based on the factorization method in the General Non-Orthogonal Matrix Element (GNOME) code. GronOR is a substantially refactored, massively MPI-parallelized code base that can take advantage of GPU acceleration. Scalability and load balancing is achieved through use of a task based algorithm. The algorithm is implemented in a fault tolerant way.
- The intended application of GronOR is for small clusters of molecules, with special interest in molecular systems relevant for photovoltaic applications.

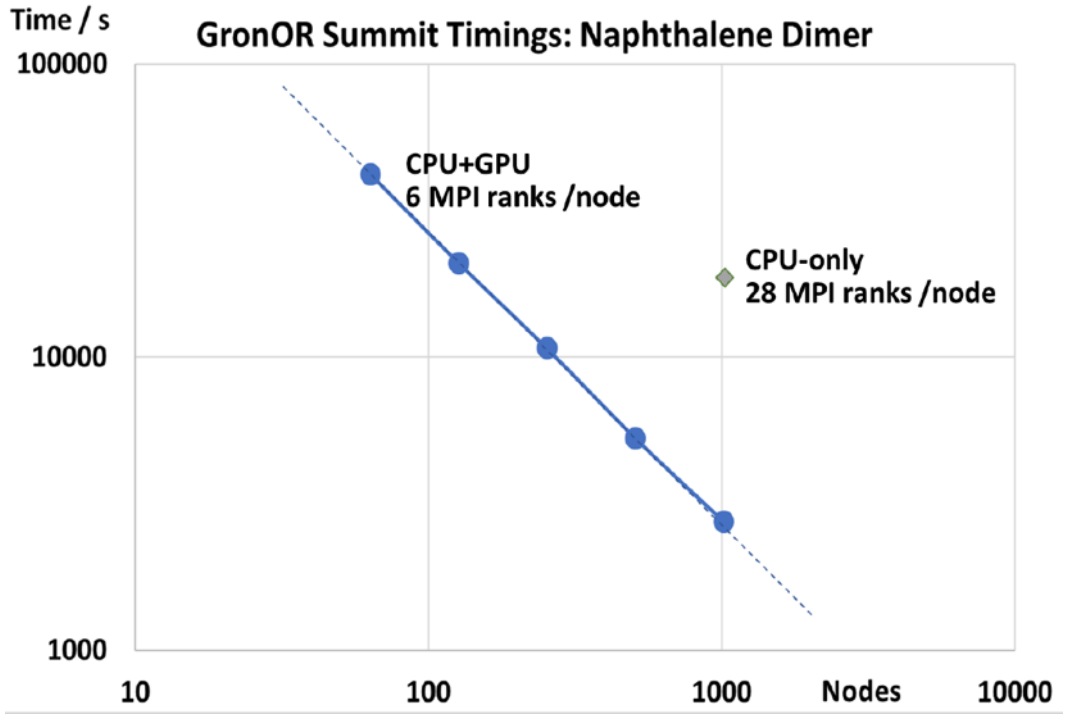
Summit Science: Computational Chemistry

Development work on scalability and load balancing

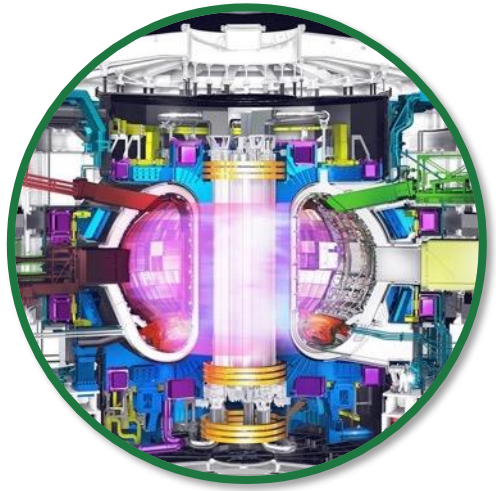
- OpenACC implementation for GPU off-loading
- Master-slave model with task based load balancing
- MPI parallelization with point-to-point non-blocking communication
- Avoid global synchronization and global reduction operations
- Fault resilient implementation
- GAMESS-UK and SYMOL for integrals and CASSCF vectors

Early Results on Summit

- Naphthalene molecules with asymmetric CASSCF configurations 44,88 and 6-311G basis set, leading to 112,867,800 Hamiltonian matrix elements
- GPU+CPU (6 MPI ranks per node) vs. CPU-only (28 MPI ranks per node) performance of 1024 node run on Summit for naphthalene dimer: **6.8x**
- Scalability is near linear on Summit up to 1080 nodes, which is close to the full Phase I system



Summit Science: Plasma Fusion

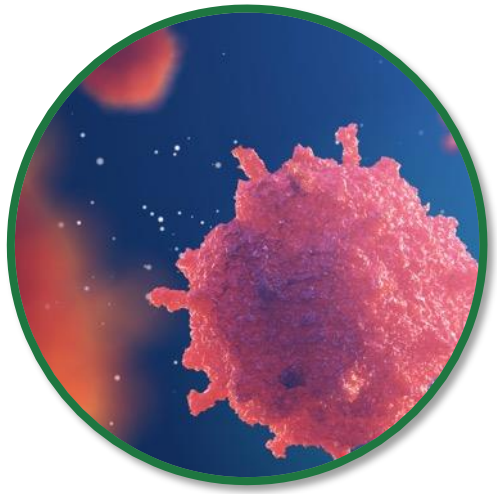


C.S. Chang
Princeton Plasma Physics Laboratory

XGC

- We are developing XGC, coupled with other codes, to be a virtual fusion reactor that models the behavior of plasma-the hot gas medium in which particles generate fusion energy.
- Understanding plasma behavior at a fundamental level is critical for fusion experiments like ITER, which may help meet global energy demand by using seawater to fuel plasma in a fusion device without contributing to the greenhouse effect or producing long-term waste.
- We plan to better predict the performance of fusion experiments by more realistically simulating particle behavior in a part of the plasma known as the edge that interacts with the interior of the device, or tokamak.
- Simulation of the edge part of a fusion reactor requires extreme scale computers due to the complexity of the problem.

Summit Science: Deep Learning for Cancer Researcher



Gina Tourassi
Oak Ridge National Laboratory

Cancer Surveillance

- We all know cancer is a major public health concern. My team is collaborating with the National Cancer Institute to support implementation of a more advanced population-based cancer surveillance program.
- We leverage recent advances in artificial intelligence (AI) to enable automated and accurate capture of important cancer surveillance data elements from clinical text documents.
- We're essentially training computers with large volumes of data to read medical documents and extract important information.
- Such information is used to help doctors determine the best treatment for each patient and improve population health outcomes.
- Summit offers the fastest compute cores; vast amounts of memory, which is important for data-intensive applications such as ours; and efficient communication among cores. These are very important aspects in the Summit architecture for the problem we're solving.

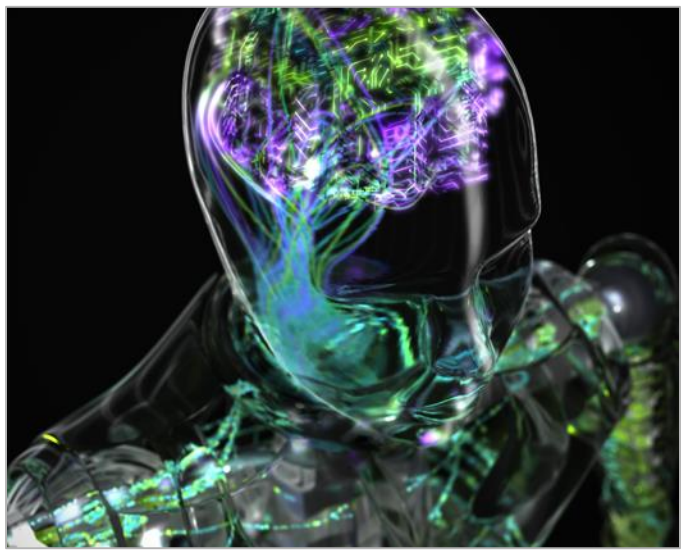
RAPIDS

Product Overview

DATA SCIENCE MARKET SIZE

Analytics

Machine Learning



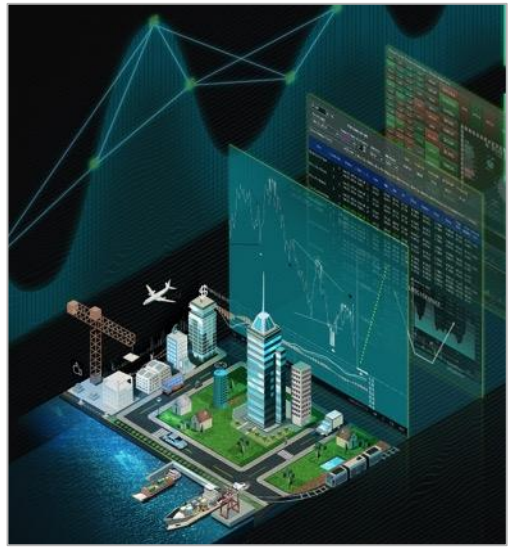
Deep Learning

BIG DATA INDUSTRY VERTICALS

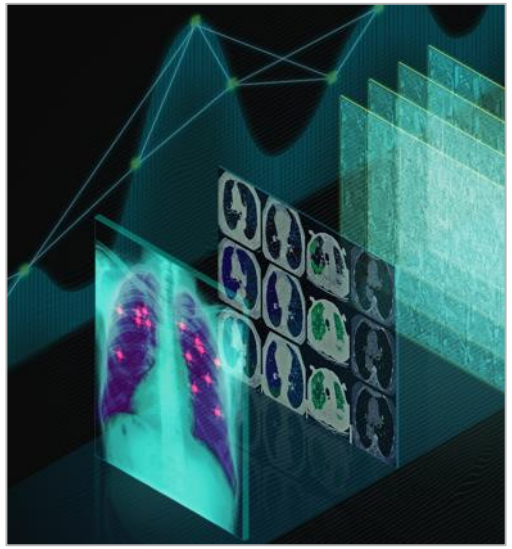
From Business Intelligence to Data Science

Consumer Internet

Retail

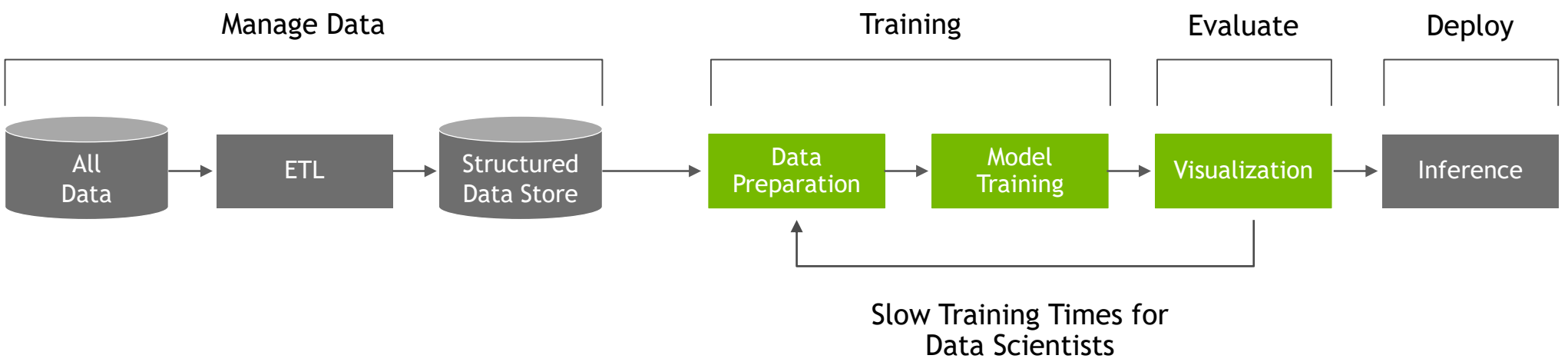


Financial Services



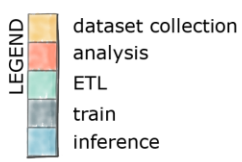
Healthcare

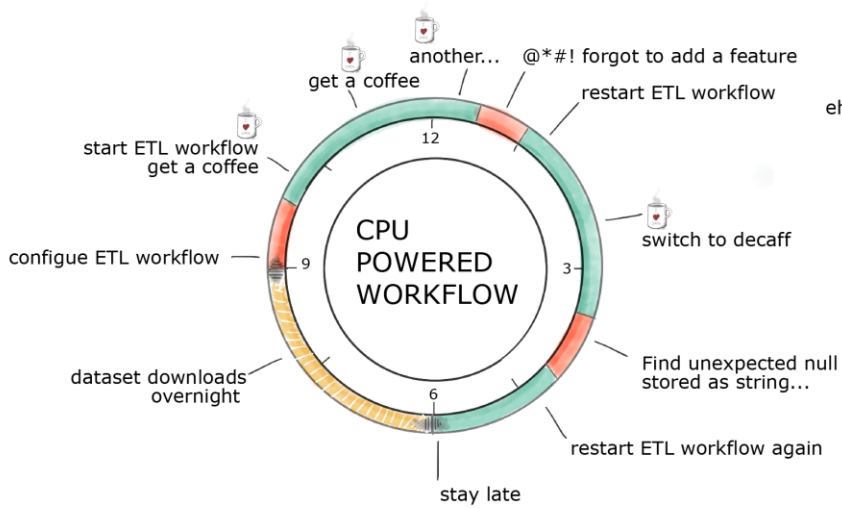
THE BIG PROBLEM IN DATA SCIENCE

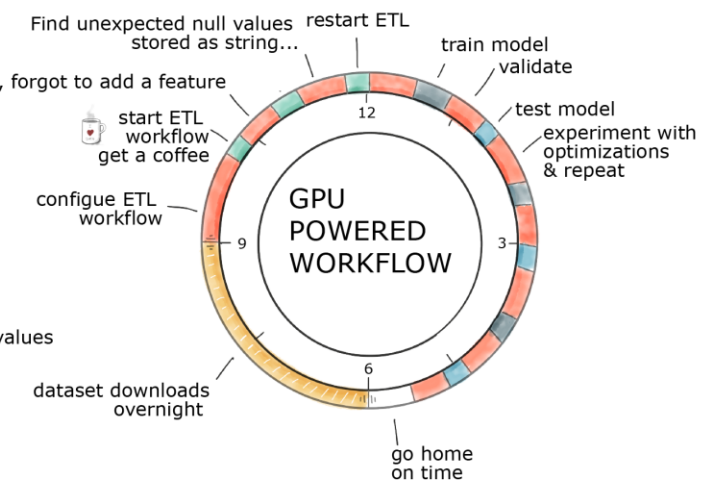


DAY IN THE LIFE OF A DATA SCIENTIST

DAY IN THE
LIFE OF A
DATA SCIENTIST

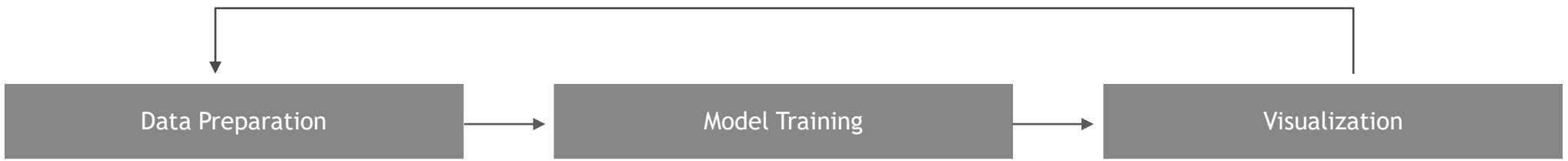


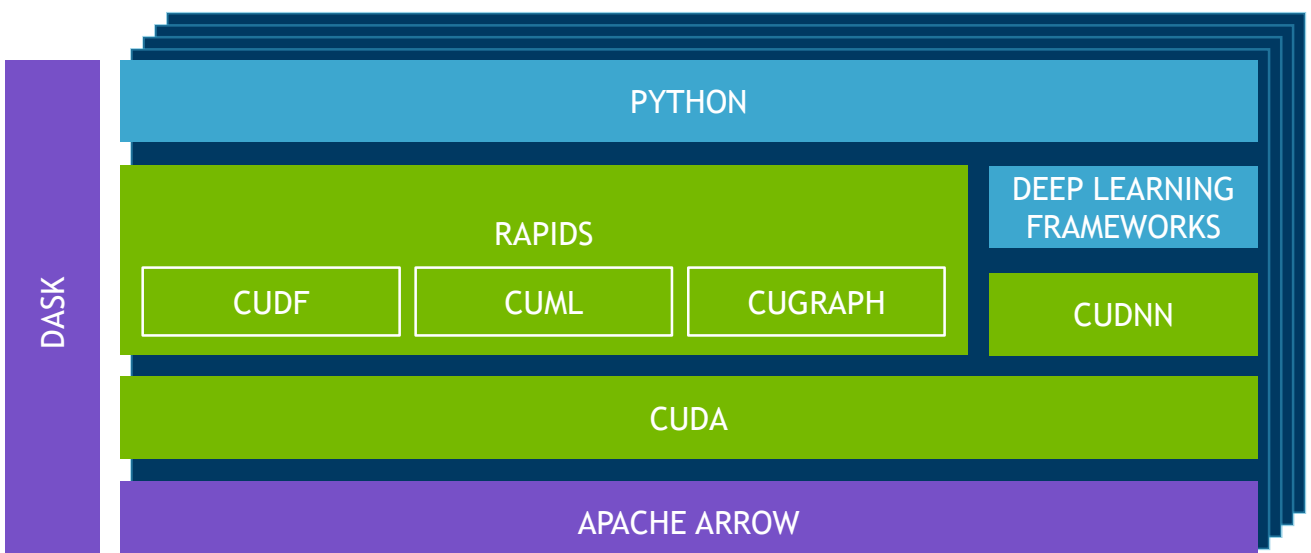




RAPIDS — OPEN GPU DATA SCIENCE

Software Stack





“RAPIDS software has immensely improved how we use data — enabling the most complex models to run at scale and deliver even more accurate forecasting.”

— Jeremy King, EVP & CTO

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.)

daskgdf
Distributed Computing

pygdf
Python Bindings

libgdf
CUDA C++ Implementation

Memory Allocation Requirement

Budget 2-3X dataset size for cuDF working memory

Multi-GPU Multi-node Roadmap

Availability	Multi-GPU	Multi-Node	Peer-to-peer Data Sharing*
Now	Yes	Yes	No
Q4 2018	Yes	Yes	Yes

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

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

Availability	Supported File Formats
Now	CSV
Q4 2018	Parquet, ORC

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

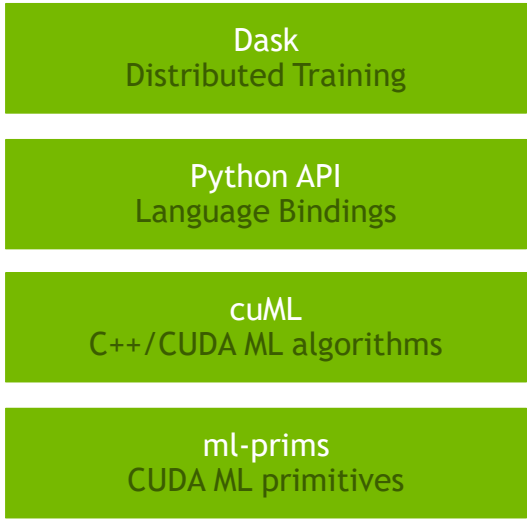
Language Bindings: Python bindings to C++/CUDA based cuML | Uses cuDF DataFrames as input

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

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

SG
Single GPU

MG
Multi-GPU

MGMN
Multi-GPU Multi-Node

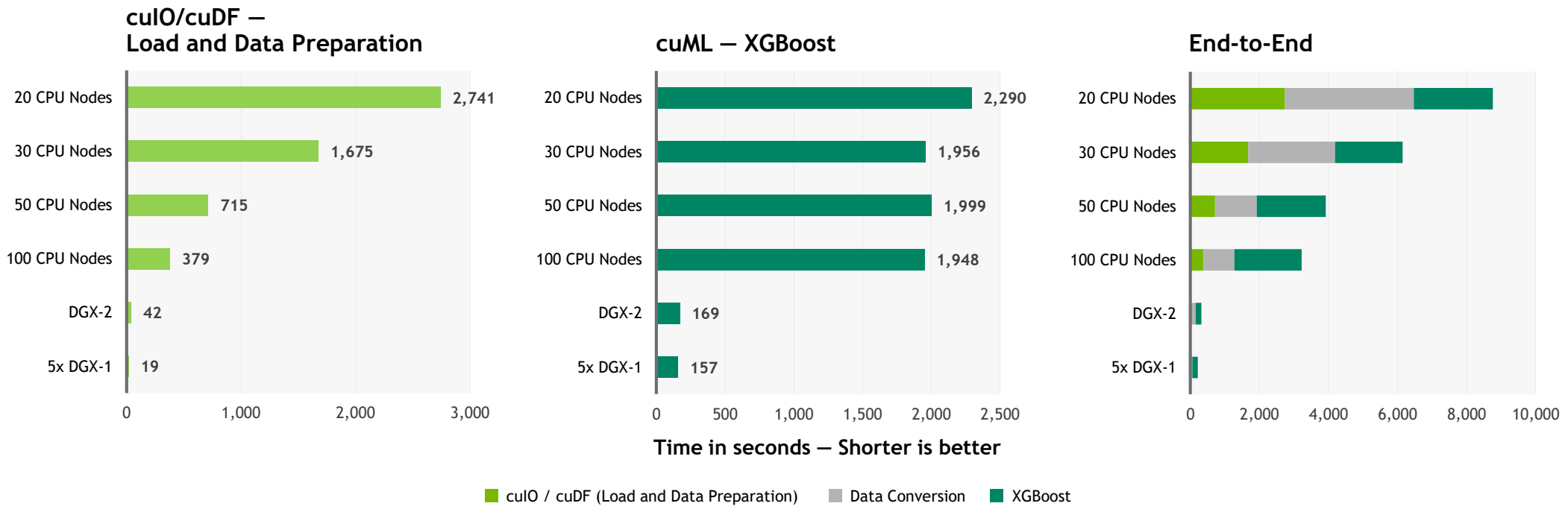
Last updated 10.16.18

cuGRAPH — GRAPH ANALYTICS

GPU Accelerated Unified Graph Analytics

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

BENCHMARKS



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

HIGH PERFORMANCE AND EASY TO USE

Hassle-Free Integration

Accelerate your Python data science toolchain with minimal code changes and no new tools to learn

Scaling Out on Any GPU

Seamless scaling from GPU workstations to multi-GPU servers and multi-node clusters

Top Model Accuracy

Increase machine learning model accuracy by iterating on models faster and deploying them more frequently

Reduced Training Time

Drastically improve your productivity with near-interactive data science

Open Source

Customizable, extensible, interoperable — the open-source software is supported by NVIDIA and built on Apache Arrow

DOWNLOAD AND DEPLOY

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

GitHub

NGC

docker

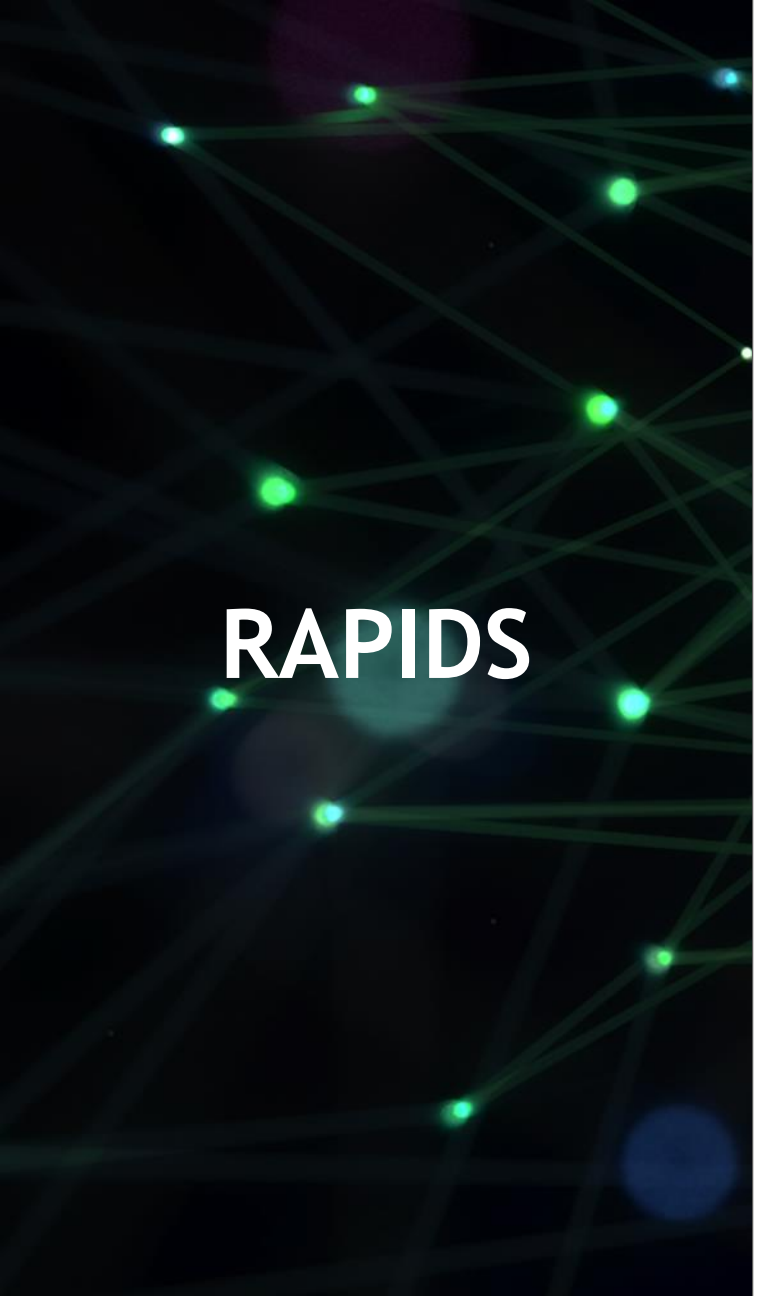
CONDA

python Package Index

Source code, libraries, packages

On-premises

Cloud

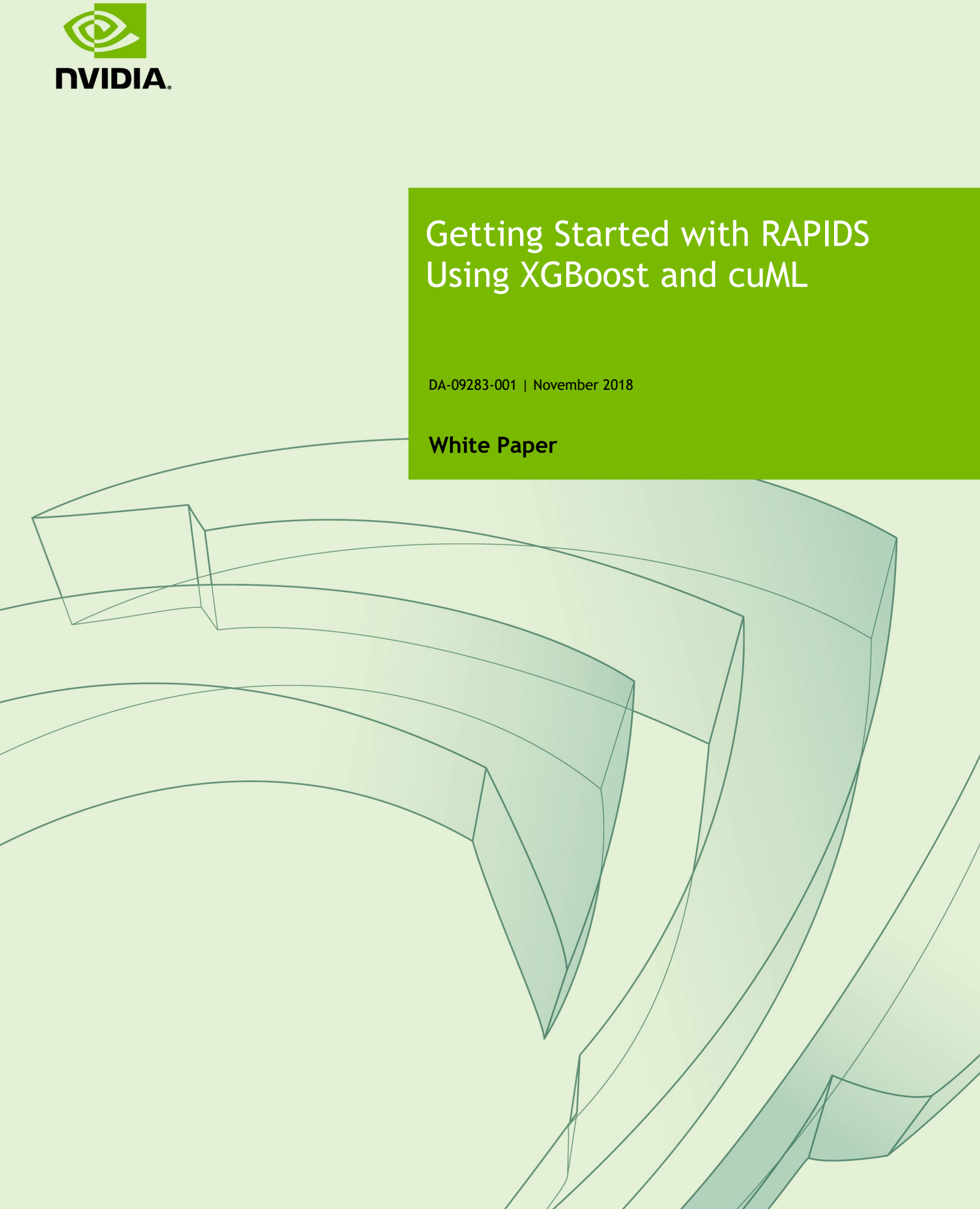


GPU Accelerated Data Science

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

Visit www.rapids.ai


NVIDIA



Getting Started with RAPIDS Using XGBoost and cuML

DA-09283-001 | November 2018

White Paper

Document Change History

DA-09283-001

Version	Date	Authors	Description of Change
01	2018-11-12	Jacci Cenci, Ken Hester, Jeff Weiss, Scott Ellis, Ty Mckercher, Robert Sohigian	Initial release

Abstract

The purpose of this document is to help data scientists leverage the performance, accuracy, and scale of machine learning (ML) algorithms accelerated by RAPIDS. The XGBoost algorithm is featured in this document because it is popular among data scientists for regression and classification prediction problems. The NVIDIA® GPU Cloud (NGC) RAPIDS container includes a mortgage example that demonstrates GPU-accelerated data processing pipelines using the XGBoost algorithm across multi-GPU systems. The current version of RAPIDS supports cuML, the NVIDIA CUDA® machine learning (ML) library, with examples for DBSCAN, PCA, tSVD, and k-NN algorithms. This document provides step-by-step instructions to help data scientists get started using RAPIDS XGBoost and cuML samples.

Contents

Environment Setup

This section describes system requirements for RAPIDS, as well as commands used to collect system information needed for the notebook examples included in the RAPIDS container. Specifically, you will need to determine:

- Minimum software versions
- System network information
- The number of GPUs in a system

Also, details are provided on how to pull the RAPIDS container from the NGC repository.

Table 1 lists the minimum software versions needed for a RAPIDS installation.

Area	Minimum Version
GPU	Pascal
CUDA	9.2
OS Support	Ubuntu 16.04 LTS
Docker	Docker CE v18+
	nvidia-docker v2+

Table 1. Minimum software versions

The IP address of the host machine is needed before running the RAPIDS container. It can be found using:

Note: The `$` prompt signifies that the host machine is being used.

```
$ hostname --all-ip-addresses
```

Select the address of the primary networking device (the first IP address in the list returned from this command, referenced as {IP-ADDR} later in this document).

Also, the number of GPUs in a system can be determined by using:

```
$ nvidia-smi -L | wc -l
```

XGBoost Step-by-Step Guide

This section describes how to use the XGBoost example included in the RAPIDS container. The container is organized with:

- IPython Notebooks (/rapids/notebooks)
- Utility scripts (/rapids/utils)
- A portion of the mortgage risk analysis data set (/rapids/data)
- A conda virtual development environment

For scale testing on multi-GPU systems, the entire mortgage dataset can be downloaded from the RAPIDS Datasets site.

Complete the steps described in Environment Setup before performing the below.

1. Pull the RAPIDS container from the NGC container repository (`nvcr.io`) and then launch an interactive pseudo-terminal session. In the following code block, port 8888 is used for JuypterLab communication and both ports 8787 and 8786 are used to monitor scheduling tasks used to distribute data to GPUs.

```
$ docker pull nvcr.io/nvidia/rapidsai/rapidsai:ubuntu1604_cuda92_py35
$ docker run --runtime=nvidia \
   --rm –it \
   -p 8888:8888 \
   -p 8787:8787 \
   -p 8786:8786 \
   nvcr.io/nvidia/rapidsai/rapidsai:ubuntu1604_cuda92_py35
```

2. Activate the virtual environment.

   Note: The # prompt signifies that a RAPIDS container is being used.

```
# cd rapids && source activate gdf
```

3. Unpack the mortgage sample data provided in the container.

   Note: The container prompt now includes `(gdf)`.

```
(gdf)# tar -xzvf data/mortgage.tar.gz --directory=data
mortgage/
mortgage/acq/
mortgage/acq/Acquisition_2000Q1.txt
mortgage/acq/Acquisition_2001Q4.txt
mortgage/acq/Acquisition_2001Q2.txt/
...
mortgage/perf/Performance_2001Q3.txt_0
mortgage/perf/Performance_2001Q2.txt_1
mortgage/names.csv
```

4. Start the JupyterLab.

```
(gdf)# bash utils/start_jupyter.sh
...
(gdf)#
```

5. Access the JupyterLab environment by specifying the following URL in a browser:

```
{IP-ADDR}:8888
```

   Where {IP-ADDR} is the address of the host machine running Docker (identified in the Environment Setup section of this document).

   Using the Chrome browser is recommended.

6. Navigate to the `notebooks` folder on left-hand navigation pane.

   In the folder are an `E2E.ipynb` (End-to-End IPython notebook) and an `ETL.ipynb` IPython notebook. These are used to demonstrate XGBoost machine learning on the mortgage dataset.

7. Double-click on the `E2E.ipynb` to open the notebook.

   Some of the cells in the notebook may need to be edited to adjust for system configuration or to test scalability. More details about how to adjust key parameters in specific cells of the E2E notebook are in the Data Right Sizing and Changing How Much Data is Used sections.

8. Run and edit the cells in the notebook (Shift + Enter) and measure timing results on some of the cells.

The RAPIDS environment is now live. Take some time to explore the E2E notebook – there are many features in RAPIDS.

See the RAPIDS documentation for more information.

Data Right Sizing

Since data requirements typically overrun individual system capacity, it is common for data scientists to split data sets to run in distributed environments. How to split data depends on several factors including data type. A large amount of time in the data science pipeline involves Extract Transform Load (ETL) processing. This section describes how RAPIDS can be used to accelerate data pre-processing and take advantage of multi-GPU systems.

The XGBoost example notebooks make use of Dask, a Python-based library for parallel computing which includes task scheduling capabilities. Cell 2 in the E2E notebook uses a bash shell script (`dask-setup.sh` which is included in the RAPIDS container) to create Dask workers that are associated with each GPU in a multi-GPU system. Replace the default number of GPUs (8, shown in yellow) with the number of GPUs in the system that were collected in the Environment Setup section.

```
%%bash
IPADDR=($(hostname --all-ip-addresses))
bash -c "/rapids/utils/dask-setup.sh 0"
bash -c "/rapids/utils/dask-setup.sh 8 8786 8787 8790 ${IPADDR[0]}
MASTER"
```

The Cells in the E2E notebook used for data processing (Cells 18 and 22) report timing details for each cell. If unusually fast times (sub-millisecond) are measured in these cells, it most likely means there is a problem. Typical issues include not having properly specified the path location for the mortgage data in Cell 4 or not have enough GPU memory.

Here are more details about sizing considerations for the XGBoost examples:

- During ETL processing in the E2E notebook, many copies of data are created in GPU memory, resulting in sporadic spikes of memory utilization
    - Memory utilization which exceeds available GPU resources will stop the Dask worker
    - Because we've disabled experimental features that would enable Dask workers to recover, restart work, or share work, the error is silently propagated forward in the Dask's delayed task graph. This can manifest itself in the form of unusually short times (sub-millisecond timescale) reported in the Cells 18 and 22.
- After Cell 20 is executed, all computed results are migrated from GPU memory back to system memory. The program will crash if there is not enough system memory.
- During execution in Cell 22, data migrates a portion of the data from system memory back into GPU device memory for XGBoost to train against.

- Training processes need a certain amount of available memory to expand throughout processing
    - With XGBoost, the overhead is typically 25% of available GPU memory. This means that we cannot exceed 24 GB of memory utilization on a 32 GB GPU, or 12 GB of memory utilization on a 16 GB GPU

Changing How Much Data is Used

Cell 4 in the E2E notebook is used to define variables that you can adjust (for example, end_year, and part_count noted in yellow) to reduce system and GPU memory requirements. This is useful for testing scalability.

```
acq_data_path = "/rapids/data/mortgage/acq"
perf_data_path = "/rapids/data/mortgage/perf"
col_names_path = "/rapids/data/mortgage/names.csv"
start_year = 2000
end_year = 2002 # end_year is not inclusive
part_count = 11 # the number of data files to train against
```

These variables define the paths to data, the number of years on which to perform ETL, and the number of parts to train against. The entire mortgage dataset is 68 quarters (available from RAPIDS Datasets site). This dataset was split into 112 parts so that each part could fit on various GPU configurations. The size of each part is on average about 1.7 GB.

Included in the RAPIDS container are four parts for year 2000, and seven parts for the year 2001 (Table 2).

Year	Part Count	Part name	Size (GB)
2000	1	/rapids/data/mortgage/perf/Performance_2000Q1.txt	0.995
	2	/rapids/data/mortgage/perf/Performance_2000Q2.txt	0.896
	3	/rapids/data/mortgage/perf/Performance_2000Q3.txt	0.951
	4	/rapids/data/mortgage/perf/Performance_2000Q4.txt	1.109
2001	5	/rapids/data/mortgage/perf/Performance_2001Q1.txt	1.702
	6	/rapids/data/mortgage/perf/Performance_2001Q2.txt_0	1.727
	7	/rapids/data/mortgage/perf/Performance_2001Q2.txt_1	1.727
	8	/rapids/data/mortgage/perf/Performance_2001Q3.txt_1	1.529
	9	/rapids/data/mortgage/perf/Performance_2001Q4.txt_0	1.529
	10	/rapids/data/mortgage/perf/Performance_2001Q4.txt_1	2.039
	11	/rapids/data/mortgage/perf/Performance_2001Q3.txt_0	2.039

Table 2. Performance files from mortgage dataset included in the RAPIDS container

Reducing the number of parts, `part_count`, reduces how much data with which XGBoost trains. Adjusting the `end_year` changes how many years on which to perform ETL.

There are memory size and GPU configuration differences between servers and workstations. Depending on hardware limitations, the entire dataset might not be able to be loaded or run in a single pass.

Table 3 shows settings for supported platforms that work for this cell in the XGBoost E2E notebook if the entire mortgage dataset was downloaded and unpacked.

Parameter	NVIDIA DGX Station™ Workstation	NVIDIA DGX-1™ Server	NVIDIA DGX-2™ Server	NVIDIA Quadro® Workstation
end_year	2007	2017	2017	2007
part_count	8	16	48	8

Table 3. Guidelines for setting end_year and part_count parameters

Note: Exit the RAPIDS container before proceeding to the next section.

```
(gdf)# exit
```

cuML Step-by-Step Guide

This section describes how to install and test the cuML-based DBSCAN, PCA, tSVD, and k-NN algorithms. The cuML packages can be added while running the RAPIDS base container using RAPIDS packages available on Anaconda Cloud. These packages can also be used for bare metal installs as well.

1. Make sure to exit the RAPIDS container used during the previous section (XGBoost Step-by-Step Guide) if you have not already done so.

```
(gdf)# exit
```

2. After completing the steps described in Environment Setup section of this document, pull the RAPIDS container from NGC (`nvcr.io`), then launch an interactive pseudo-terminal session. Port 8888 is used for JupyterLab communication, and ports 8787, and 8786 are used to monitor scheduling tasks used to distribute data to GPUs.

```
$ docker pull
nvcr.io/nvidia/rapidsai/rapidsai:ubuntu1604_cuda92_py35
$ docker run --runtime=nvidia \
   --rm –it \
   -p 8888:8888 \
   -p 8787:8787 \
   -p 8786:8786 \
   nvcr.io/nvidia/rapidsai/rapidsai:ubuntu1604_cuda92_py35
```

3. Activate the virtual environment.

   Note: The # prompt signifies that a RAPIDS container is being used.

```
# cd rapids && source activate gdf
```

4. From inside the container environment, perform the following steps to add the cuML packages (includes notebook examples for DBSCAN, PCA, tSVD, and k-NN).

   Note: The container prompt now includes (gdf).

```
(gdf)# conda install -y -c rapidsai cudf
(gdf)# conda install -y -c pytorch faiss-gpu cuda92
(gdf)# conda install -y -c rapidsai cuml
(gdf)# export NUMBAPRO_CUDA_DRIVER=/usr/lib/x86_64-linux-
gnu/libcuda.so
(gdf)# export NUMBAPRO_NVVM=/usr/local/cuda-
9.2/nvvm/lib64/libnvvm.so
(gdf)# export NUMBAPRO_LIBDEVICE=/usr/local/cuda-
9.2/nvvm/libdevice/
(gdf)# git clone https://github.com/rapidsai/cuml.git
```

Here is an abbreviated output from executing these commands:

```
Solving environment: done
## Package Plan ##
  environment location: /conda/envs/gdf
  added / updated specs: cudf

The following packages will be downloaded:
package                    |            build
---------------------------|-----------------
cudf-0.2.0                 |         py35_149         175 KB
nvstrings-0.0.0.dev        |   cuda9.2_py35_0         2.5 MB
libgdf-0.2.0               |      cuda9.2_149         8.7 MB
libgdf_cffi-0.2.0          | cuda9.2_py35_149          23 KB
------------------------------------------------------------

The following NEW packages will be INSTALLED:

    cffi:        1.11.5-py35he75722e_1
    cudf:        0.2.0-py35_149           rapidsai
    libgdf:      0.2.0-cuda9.2_149        rapidsai
    libgdf_cffi: 0.2.0-cuda9.2_py35_149   rapidsai
    nvstrings:   0.0.0.dev-cuda9.2_py35_0 rapidsai
    pycparser:   2.19-py35_0
...
## Package Plan ##
  environment location: /conda/envs/gdf
  added / updated specs: cuml

The following packages will be downloaded:

package                    |            build
---------------------------|-----------------
cuml-0.2.0                 |  cuda9.2_py35_17         2.1 MB

The following NEW packages will be INSTALLED:

    cuml: 0.2.0-cuda9.2_py35_17 rapidsai
 ...
Receiving objects: 100% (949/949), 5.57 MiB | 0 bytes/s, done.
Resolving deltas: 100% (439/439), done.
Checking connectivity... done.
```

Once the cuDF and cuML packages are installed inside the container, proceed to the next step to start the Jupyter Lab.

To avoid repeating the cuML install steps between container invocations, reference documentation on docker commit to save the container image for future use.

5. Start the Jupyter Lab.

```
(gdf)# bash utils/start_jupyter.sh
...
(gdf)#
```

6. Access the Jupyter environment by specifying the following URL in a browser:

```
{IP-ADDR}:8888
```

   Where {IP-ADDR} is the address of the host machine running Docker (identified in the Environment Setup section of this document). Using the Chrome browser is recommended.

7. Using the left-hand navigation pane, go to `cuml`, then `python`, then finally the `notebooks` folder.

8. Double-click on any of these notebooks to explore how the RAPIDS function calls closely resemble the familiar scikit-learn function calls:
   - `dbscan_demo.ipynb`
   - `knn_demo.ipynb`
   - `pca_demo.ipynb`
   - `tsvd_demo.ipynb`

9. Run and edit the cells in the notebook (Shift + Enter) and measure timing results and verify accuracy.

Note: These cuML algorithms will evolve to take advantage of multi-GPU systems, so be sure to check NGC container registry often for updated RAPIDS images.

The RAPIDS environment is now live. Take some time to explore the E2E notebook – there are many features in RAPIDS.

See the RAPIDS documentation for more information.

Legal Notices and Trademarks

Notices

The information provided in this specification is believed to be accurate and reliable as of the date provided. However, NVIDIA Corporation (“NVIDIA”) does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This publication supersedes and replaces all other specifications for the product that may have been previously supplied.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and other changes to this specification, at any time and/or to discontinue any product or service without notice. Customer should obtain the latest relevant specification before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer. NVIDIA hereby expressly objects to applying any customer general terms and conditions regarding the purchase of the NVIDIA product referenced in this specification.

NVIDIA products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on these specifications will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do the necessary testing for the application to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this specification. NVIDIA does not accept any liability related to any default, damage, costs or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this specification, or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this specification. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA. Reproduction of information in this specification is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

Trademarks

DGX, Tesla, NVLink, NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright