Accelerating GPU-based Machine Learning in Python using MPI Library: A Case Study with MVAPICH2-GDR

S. Mahdieh Ghazimirsaeed, Quentin Anthony, Aamir Shafi,

Hari Subramoni and Dhabaleswar K. (DK) Panda

Network Based Computing Laboratory

The Ohio State University

{ghazimirsaeed.3, anthony.301, shafi.16, subramoni.1,

panda.2}@osu.edu

THE OHIO STATE UNIVERSITY

Outline

- Introduction
- Motivation
- Overview of the Software Stacks
- MPI-Based Communication Support in cuML
- Performance Evaluation and Characterization
- Conclusion

Introduction

- Unprecedented growth in data generated from diverse sources
- Machine Learning (ML) libraries, tools, and techniques: processing and extracting useful information from this data
- Scikit-learn and Apache Spark's MLlib: natively designed to support the execution of ML algorithms on CPUs.
- GPUs:
  - Popular platform for optimizing parallel workloads
  - Match for ML applications, which require high arithmetic intensity

Introduction (cont.)

- RAPIDS AI: enables end-to-end data science analytic pipelines entirely on GPUs.
- cuML
  - GPU-accelerated ML library
  - GPU-counterpart of Scikit-learn
  - Supports the execution of ML workloads on Multi-Node Multi-GPUs (MNMG) systems

Outline

- Introduction
- Motivation
- Overview of the Software Stacks
- MPI-Based Communication Support in cuML
- Performance Evaluation and Characterization
- Conclusion

Motivation

- Communication stages in cuML:
  - The training data is distributed to all workers
  - The output of the training stage i.e. the model parameters are shared with all workers
- Communication Support in cuML:
  - Point-to-point communication: Dask
  - Collective communication: NVIDIA Collective Communications Library (NCCL)

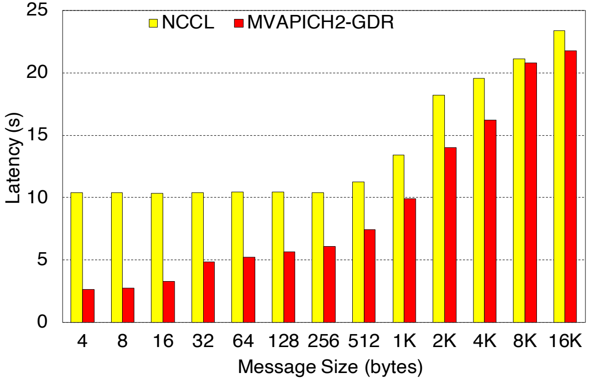
Motivation: Combine Ease-of-use with High-performance

Libraries	GPU Support	MNMG Support	Python Support	High Performance
Scikit-learn	✗	✗	✓	✗
Spark's MLlib	✗	✓	✓	✗
Mahout	✓	✓	✗	✗
PAPIDS cuML	✓	✓	✓	✗
MPI	✓	✓	✗	✓
Our paper	✓	✓	✓	✓

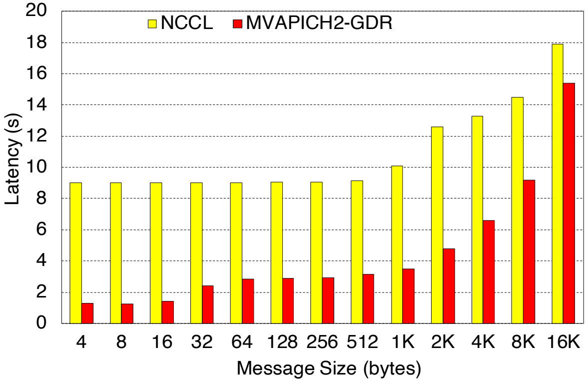
How can we combine the ease-of-use provided by cuML for running ML applications with the high-performance provided by MPI?

Motivation: Support MPI-based Collectives in cuML

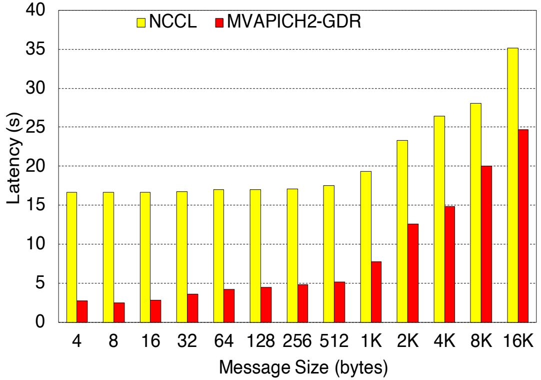
- **MVAPICH2-GDR: Support efficient communication between GPU devices**



Reduce



Bcast



Allreduce

How can we replace NCCL-based collective communications in cuML with MPI-based communications to take advantage of efficient and GPU-aware collective communication designs in MVAPICH2-GDR?

Motivation: Performance Characterization for cuML Algorithms

- Training based on different ML algorithms:
  - K-Means
  - tSVD
  - Random Forest
  - Linear Regression
- Understand cuML to achieve the best performance
  - being a relatively new ML library
  - not studied well by the community

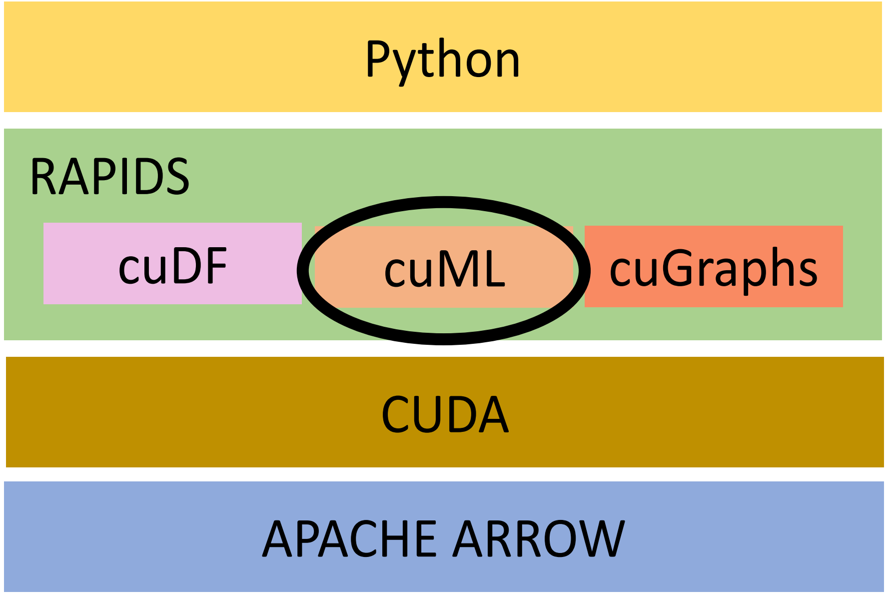
How can we provide performance characterization for GPU-accelerated cuML Algorithms and provide guidelines for data scientists to take the most advantage of them?

Outline

- Introduction
- Motivation
- Overview of the Software Stacks
- MPI-Based Communication Support in cuML
- Performance Evaluation and Characterization
- Conclusion and Future Work

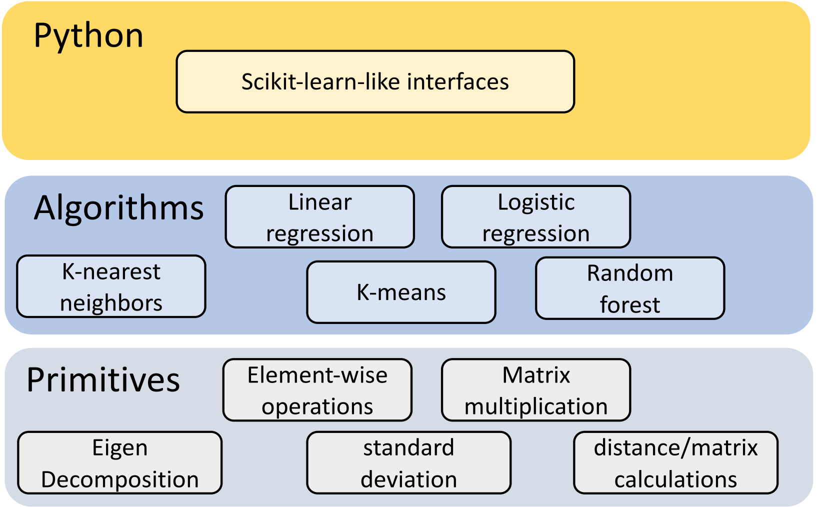
RAPIDS Software Stack

- Built on top of CUDA
- Under the standard specification of Apache Arrow
- Three main components
  - cuDF: data-frame manipulation library
  - cuML: Machine Library library
  - cuGraphs: accelerated graph analytics library



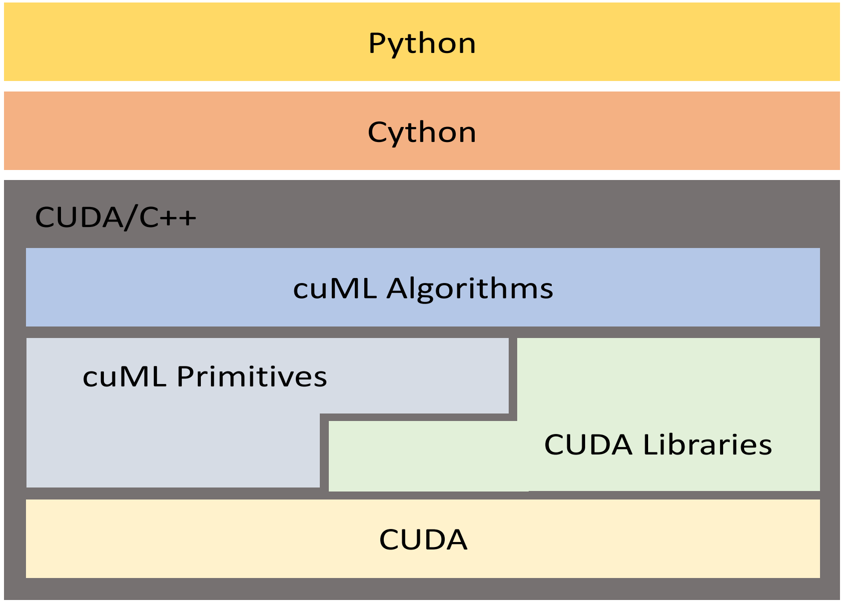
cuML Components

- Three main components
  - Primitives
    - Reusable building blocks for building machine learning algorithms
    - Common for different machine learning algorithms
    - Used to build different machine learning algorithms
  - Machine learning Algorithms
  - Python layer
    - Provides a Scikit-learn like interface
    - Hides the complexities of the C/C++ layer



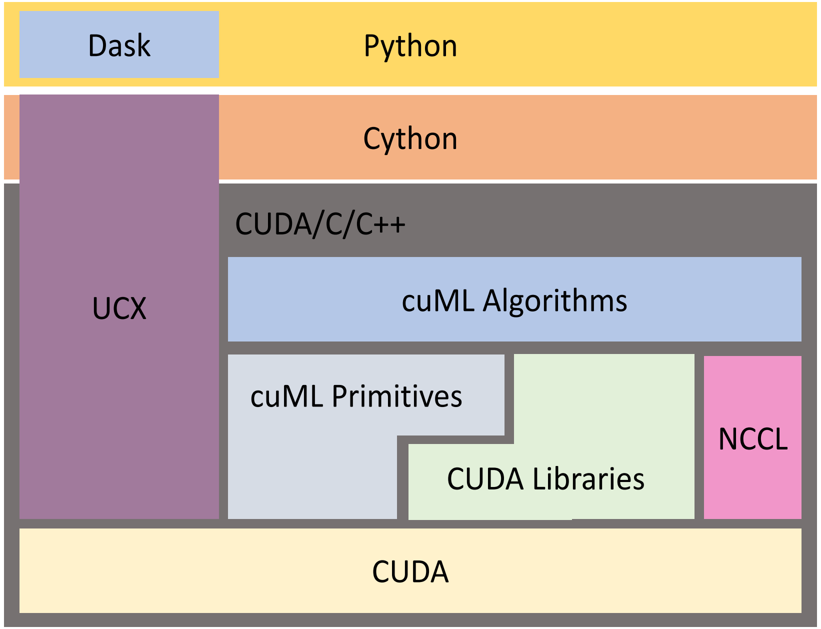
cuML Software Stack

- Software stack of cuML in a system with single GPU
- Primitives and cuML algorithms built on top of CUDA
- The CUDA/C++ layer is wrapped to the Cython layer to expose the cuML algorithms



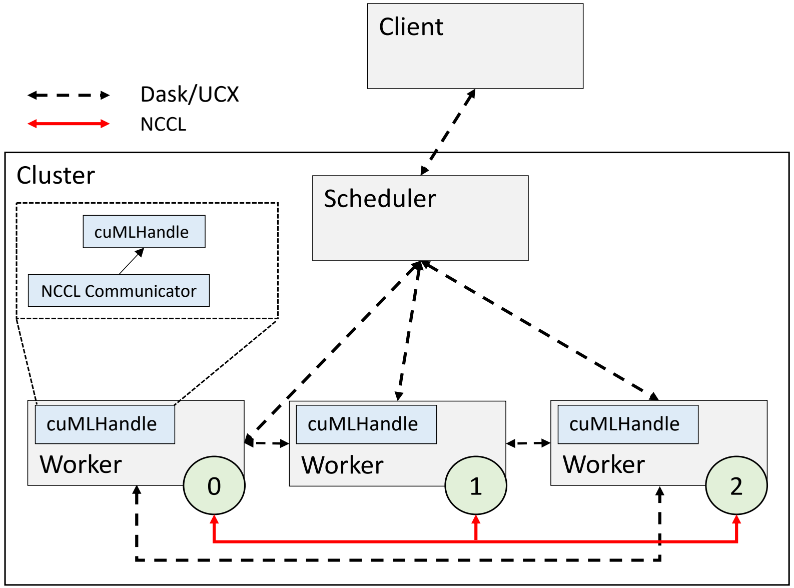
cuML Software Stack in Distributed Setting

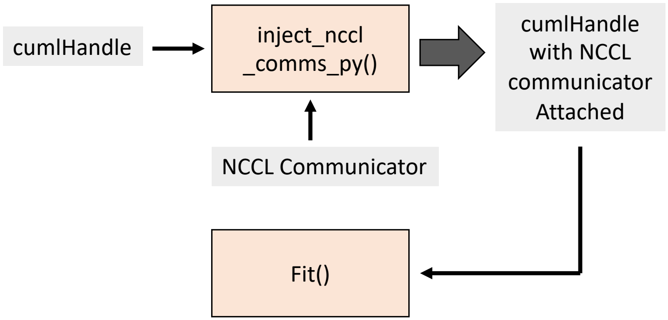
- Two components are added:
  - Dask: for handling point-to-point communications
  - NCCL: for handling collective communications



Dask and NCCL Communication Paths in cuML

- A NCCL communicator is created across the worker processes
- cumlHandle: A class in cuML that is used to manage resources



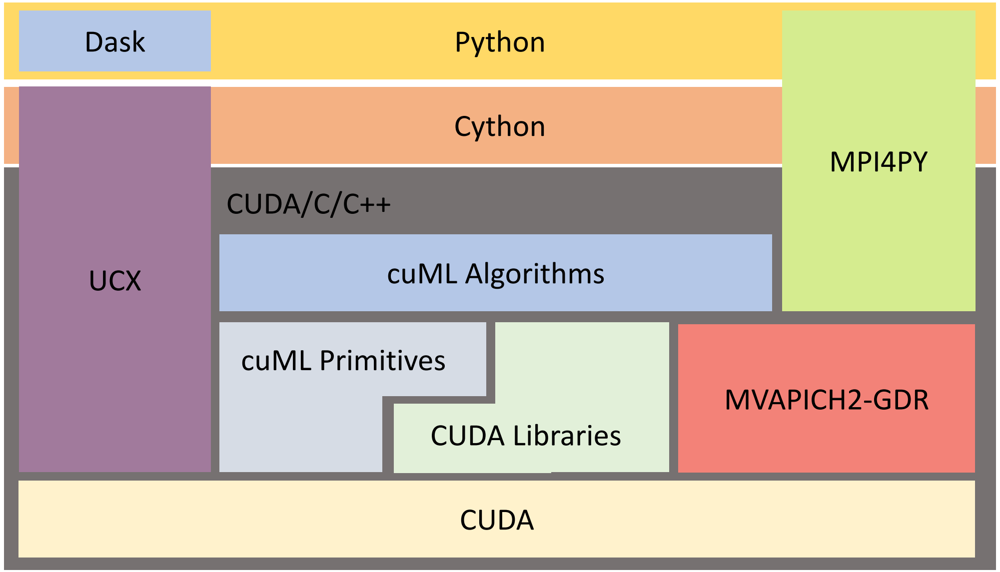


Outline

- Introduction
- Motivation
- Overview of the Software Stacks
- MPI-Based Communication Support in cuML
- Performance Evaluation and Characterization
- Conclusion

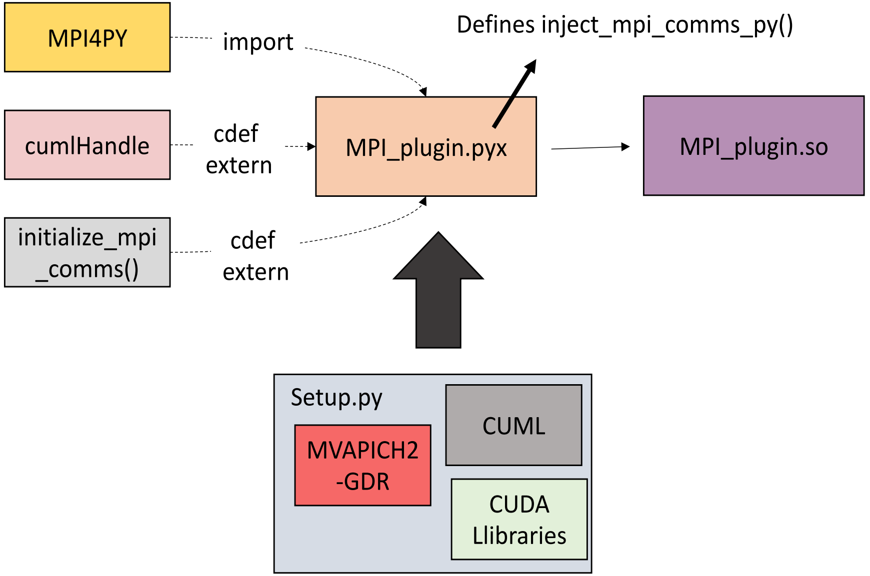
MPI-Based Communication Support in cuML

- MVAPICH2-GDR: for handling collective communications
- MPI4PY: Python binding library for MPI



MPI-Based Communication Support in cuML

- Use a Cython wrapper to inject MPI communicator to cuML handle



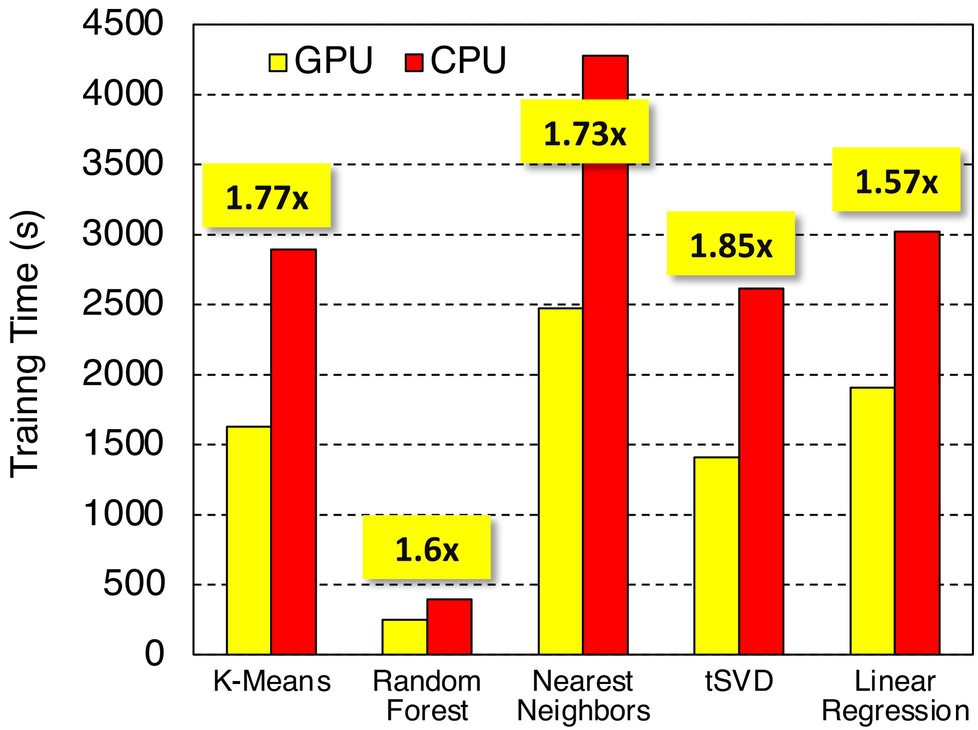
Outline

- Introduction
- Motivation
- cuML Software Stack
- MPI-Based Communication Support in cuML
- Performance Evaluation and Characterization
- Conclusion and Future Work

Experimental Setup

Specification	SDSC Comet
Number of Nodes	36
Processor Family	Xeon Haswell
Processor Model	E5-2680 v3
Clock Speed	2.5 GHz
Sockets	2
Cores per Socket	12
RAM (DDR4)	128 GB
GPU Family	NVIDIA Pascal P100
GPUs	4
GPU Memory	16 GB (HBM2)
Interconnect	IB-EDR (56G)

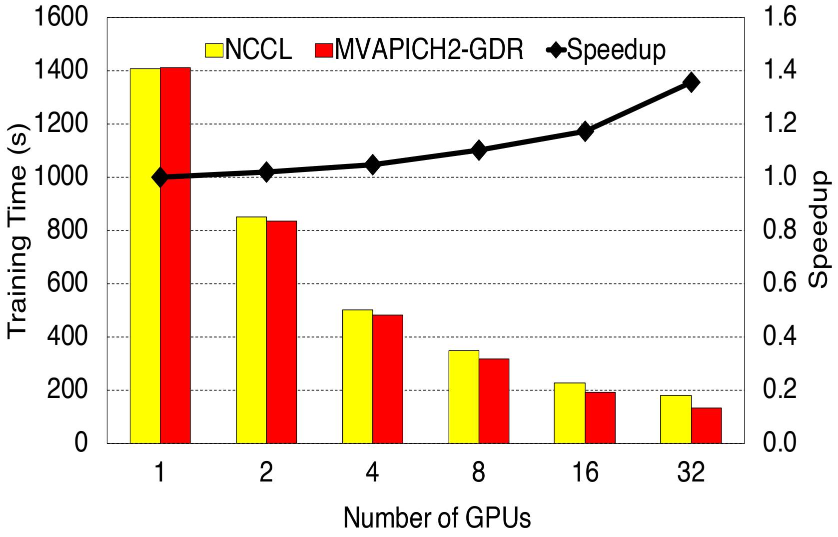
Performance Results



CPU vs. GPU

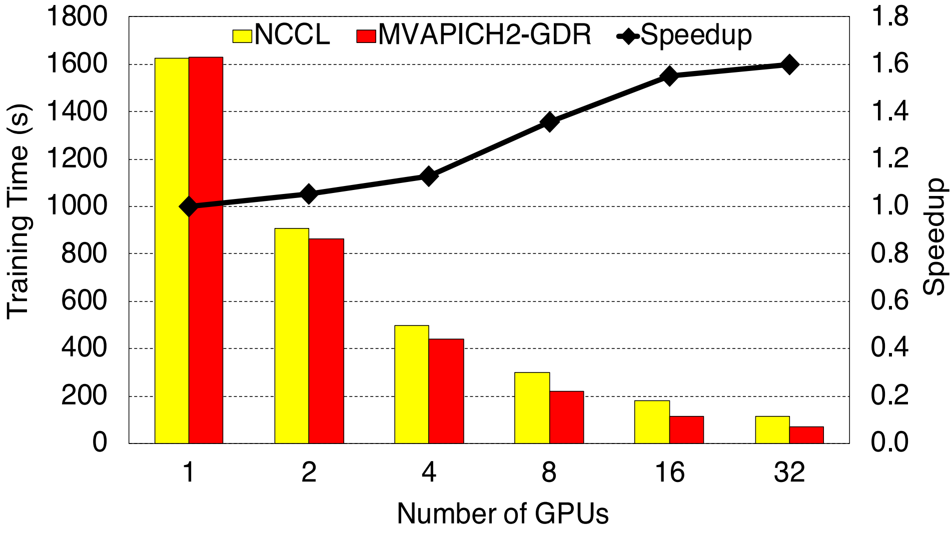
Performance Results

1.38x speedup on 32 GPUs



Truncated SVD

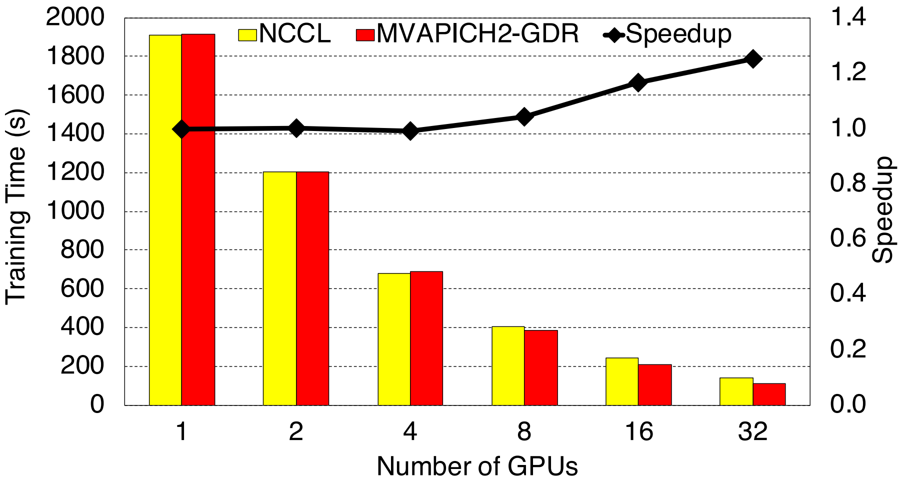
1.6x speedup for 32 GPUs



K-Means

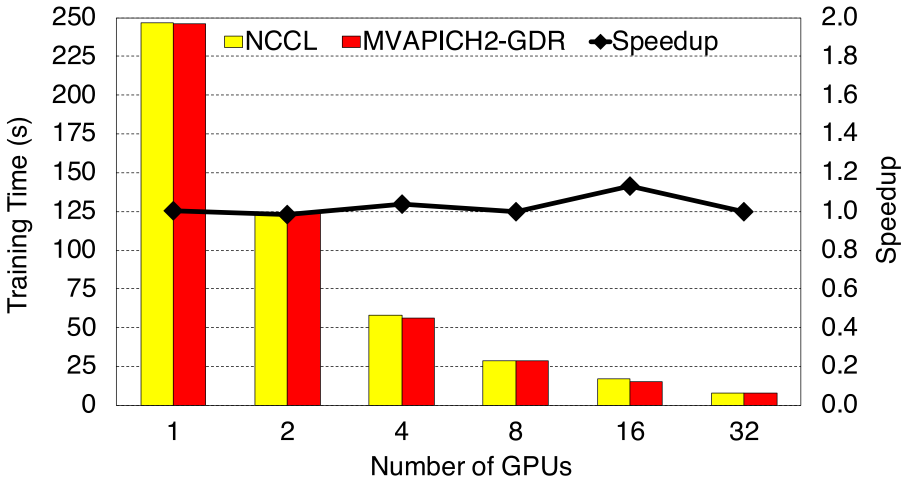
Performance Results

1.25x speedup on 32 GPUs



Linear Regression

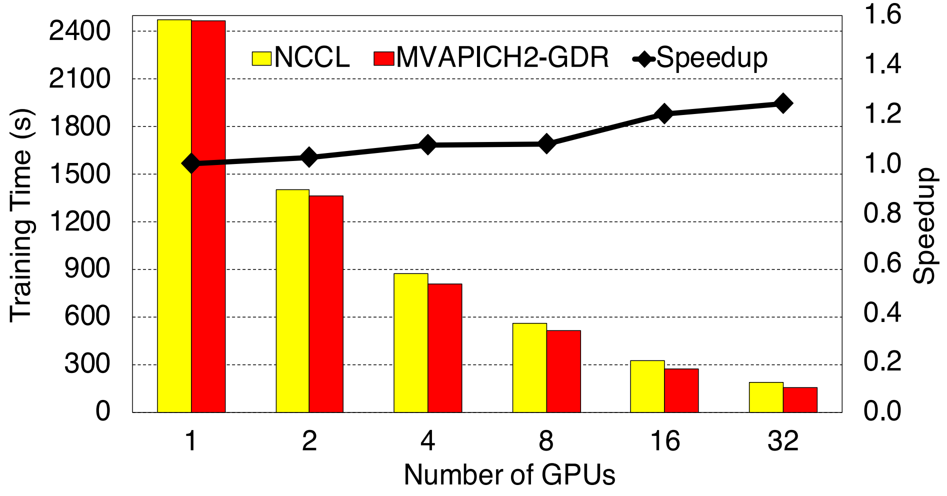
1.1x speedup for 16 GPUs



Random Forest

Performance Results

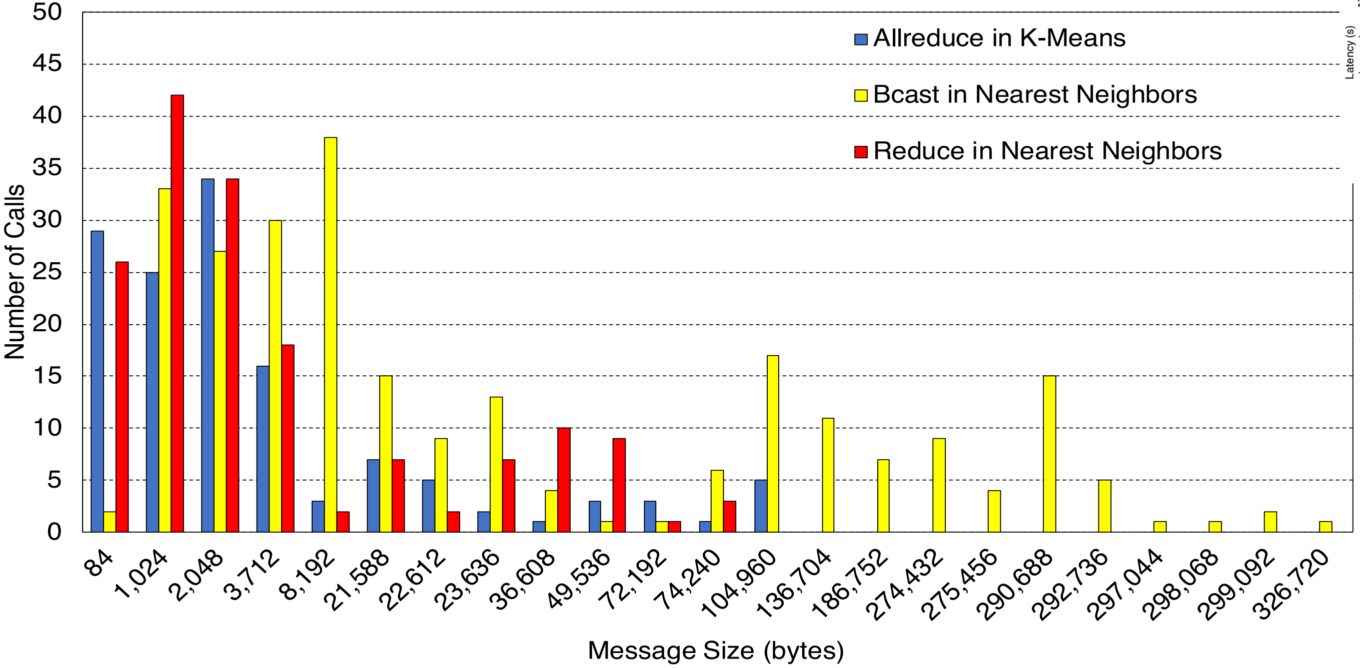
1.24x speedup on 32 GPUs

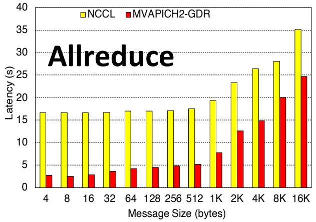


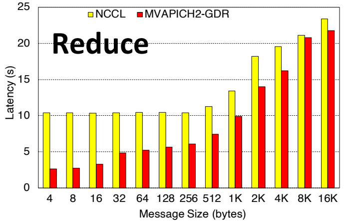
Nearest Neighbors

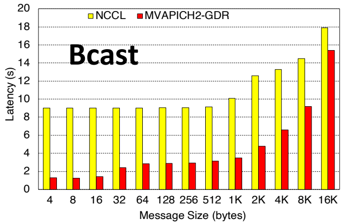
Collectives in cuML Algorithms

- K-Means: Allreduce
- Nearest neighbor: Bcast and Reduce



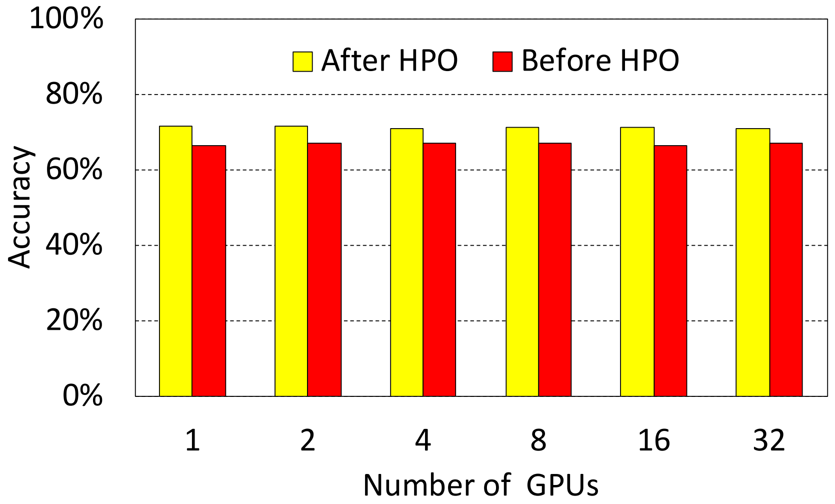




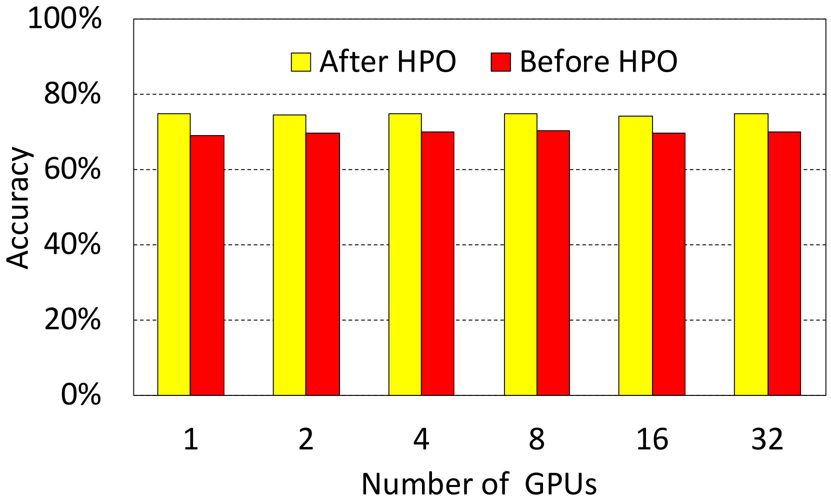


Accuracy

- Hyperparameter optimization (HPO) to the real-world Higgs dataset



K-Means



Random Forest

Outline

- Introduction
- Motivation
- Overview of the Software Stacks
- MPI-Based Communication Support in cuML
- Performance Evaluation and Characterization
- Conclusion

Conclusion

- Add support for MPI-based communications for cuML applications in Python
- Take advantage of MPI collective communication for communication between workers in cuML
- Provide a synthetic benchmarking suite and in-depth analysis of cuML algorithms
- Compare the performance of the proposed MPI-based communication approach with NCCL-based communication design
- Up to 1.6x, 1.25x, 1.25x,and 1.36x speedup for K-Means, Nearest Neighbors, Linear Regression, and tSVD on 32 GPUs
- Will be available to the community

Thank You!

ghazimirsaeed.3@osu.edu

Network-Based Computing Laboratory

http://nowlab.cse.ohio-state.edu/

MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

HiBD
High-Performance Big Data

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

HiDL
High-Performance Deep Learning

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/