北京大學
PEKING UNIVERSITY

Optimization for Deep Learning

Lecture 13-6: BlueFog Libraries

Kun Yuan

BlueFog: An open-source and high-performance python library

BlueFog

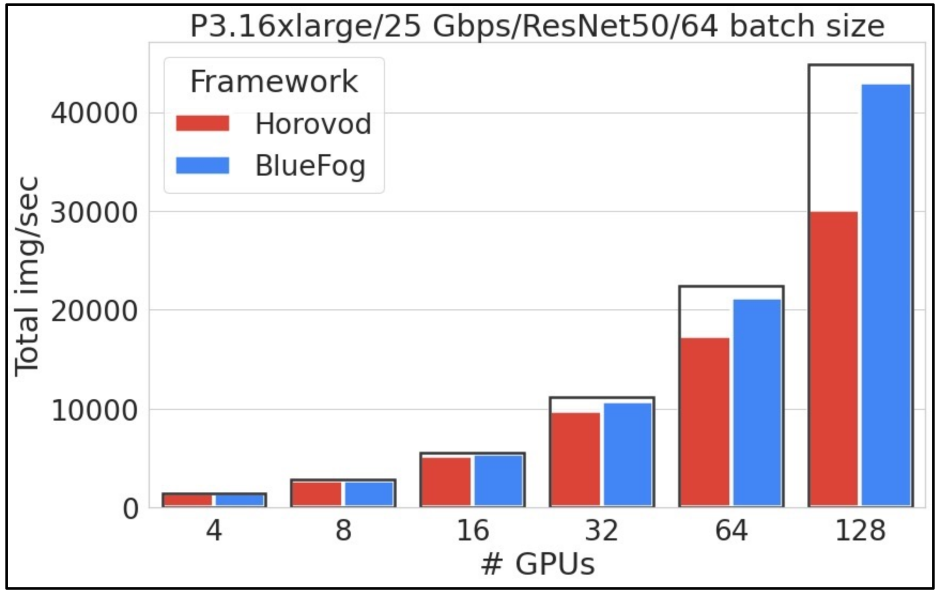
https://github.com/Bluefog-Lib/bluefog

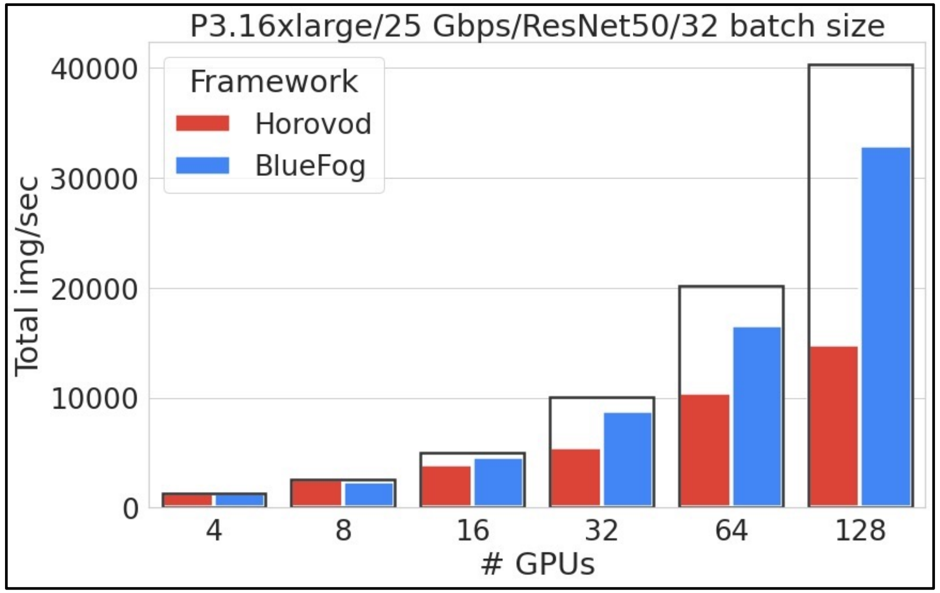
BlueFog

- An open-source library to support decentralized communication in optimization and deep learning
- High-performance
- Easy-to-use

High-performance

- BlueFog has larger throughput than Horovod (the SOTA DL system implementing PSGD) [YYH+21]





- All our research progresses are involved in BlueFog

[YYH+21] B. Ying, K. Yuan, H. Hu, Y. Chen, and W. Yin, ``BlueFog: Make Decentralized Algorithms Practical for Optimization and machine learning", arXiv:2111.04287 [GitHub site: github.com/Bluefog-Lib/bluefog]

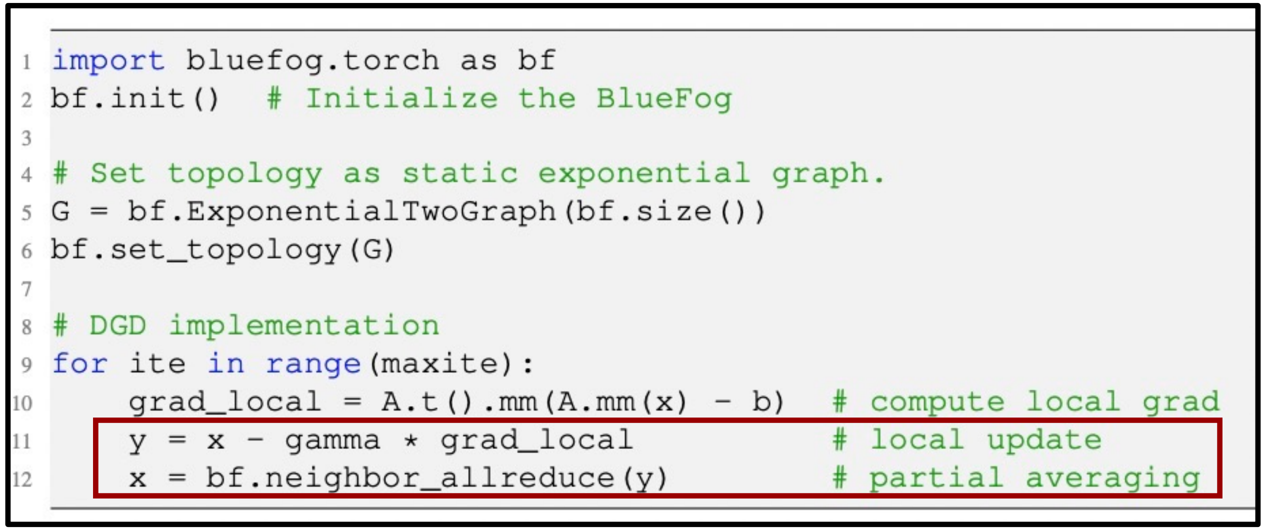
Easy-to-use

- Writing codes for decentralized methods is as easy as writing equations

Decentralized least-square algorithms

$$y_i^{(k)} = x_i^{(k)} - \gamma A_i^T (A_i x_i^{(k)} - b_i)$$

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} y_j^{(k)}$$



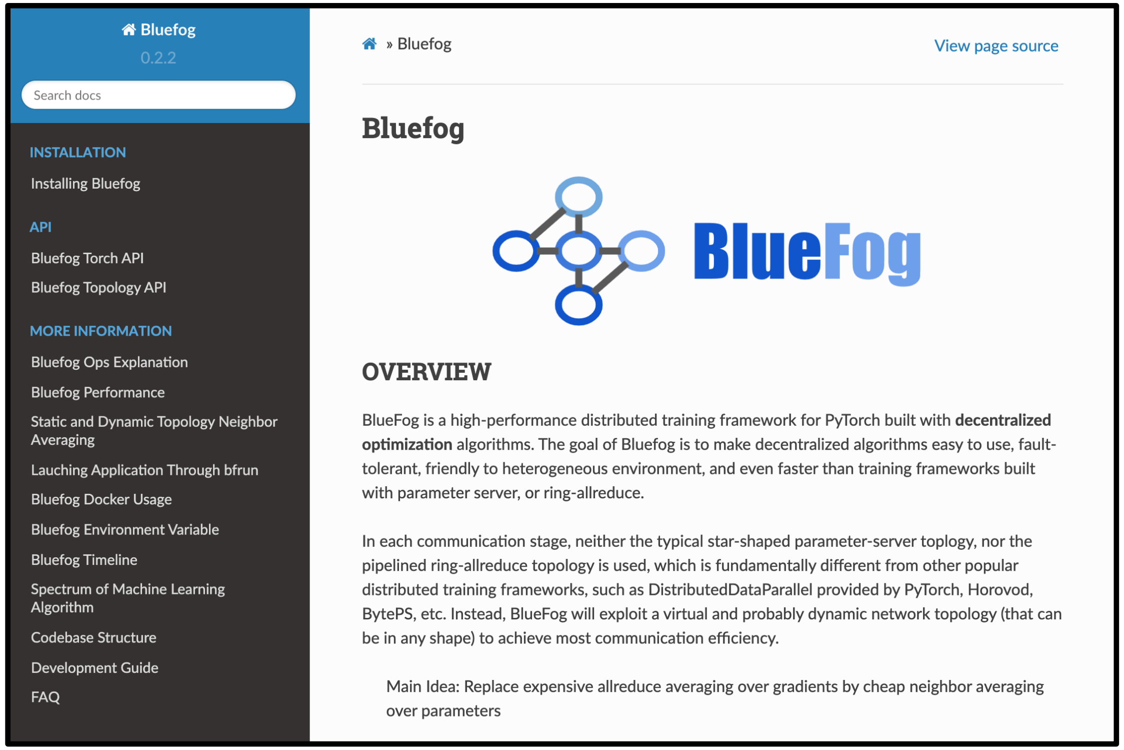
```python
import bluefog.torch as bf
bf.init()  # Initialize the BlueFog

# Set topology as static exponential graph.
G = bf.ExponentialTwoGraph(bf.size())
bf.set_topology(G)

# DGD implementation
for ite in range(maxite):
    grad_local = A.t().mm(A.mm(x) - b)  # compute local grad
    y = x - gamma * grad_local          # local update
    x = bf.neighbor_allreduce(y)        # partial averaging
```

Easy-to-use

Abundant documents



Bluefog
0.2.2
Search docs

INSTALLATION
Installing Bluefog

API
Bluefog Torch API
Bluefog Topology API

MORE INFORMATION
Bluefog Ops Explanation
Bluefog Performance
Static and Dynamic Topology Neighbor Averaging
Lauching Application Through bfrun
Bluefog Docker Usage
Bluefog Environment Variable
Bluefog Timeline
Spectrum of Machine Learning Algorithm
Codebase Structure
Development Guide
FAQ

» Bluefog View page source

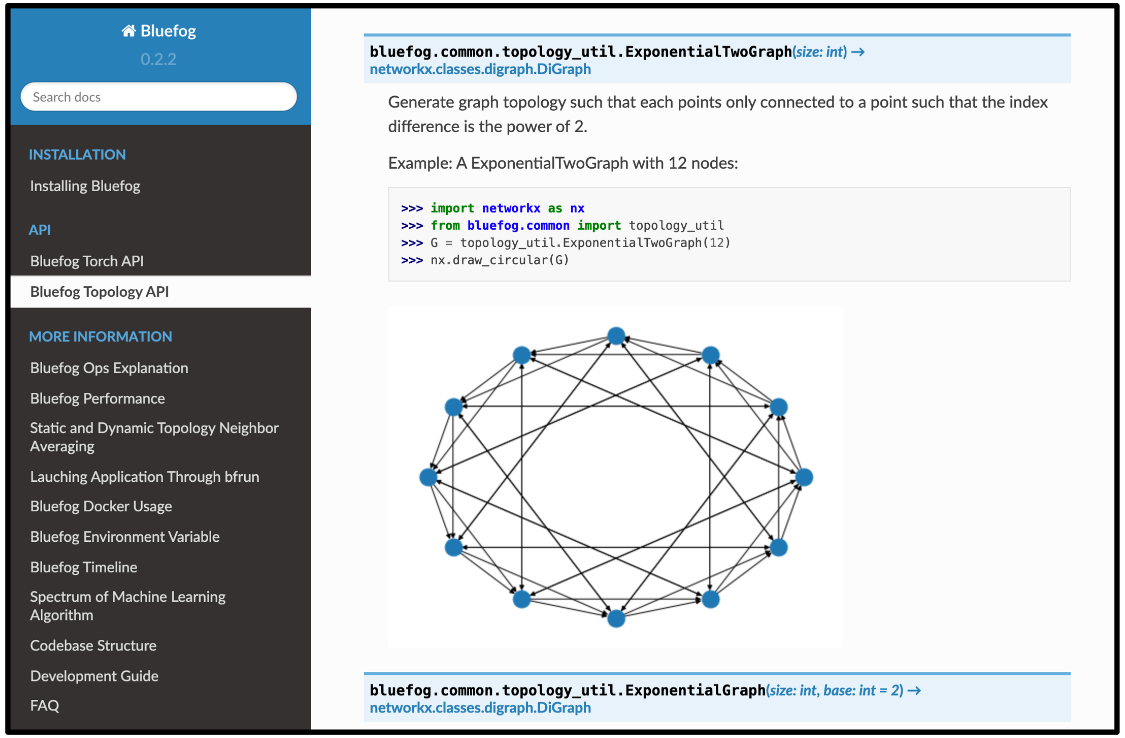
Bluefog

OVERVIEW

BlueFog is a high-performance distributed training framework for PyTorch built with **decentralized optimization** algorithms. The goal of Bluefog is to make decentralized algorithms easy to use, fault-tolerant, friendly to heterogeneous environment, and even faster than training frameworks built with parameter server, or ring-allreduce.

In each communication stage, neither the typical star-shaped parameter-server toplogy, nor the pipelined ring-allreduce topology is used, which is fundamentally different from other popular distributed training frameworks, such as DistributedDataParallel provided by PyTorch, Horovod, BytePS, etc. Instead, BlueFog will exploit a virtual and probably dynamic network topology (that can be in any shape) to achieve most communication efficiency.

> Main Idea: Replace expensive allreduce averaging over gradients by cheap neighbor averaging over parameters



Bluefog
0.2.2
Search docs

INSTALLATION
Installing Bluefog

API
Bluefog Torch API
Bluefog Topology API

MORE INFORMATION
Bluefog Ops Explanation
Bluefog Performance
Static and Dynamic Topology Neighbor Averaging
Lauching Application Through bfrun
Bluefog Docker Usage
Bluefog Environment Variable
Bluefog Timeline
Spectrum of Machine Learning Algorithm
Codebase Structure
Development Guide
FAQ

bluefog.common.topology_util.ExponentialTwoGraph(*size: int*) → networkx.classes.digraph.DiGraph

Generate graph topology such that each points only connected to a point such that the index difference is the power of 2.

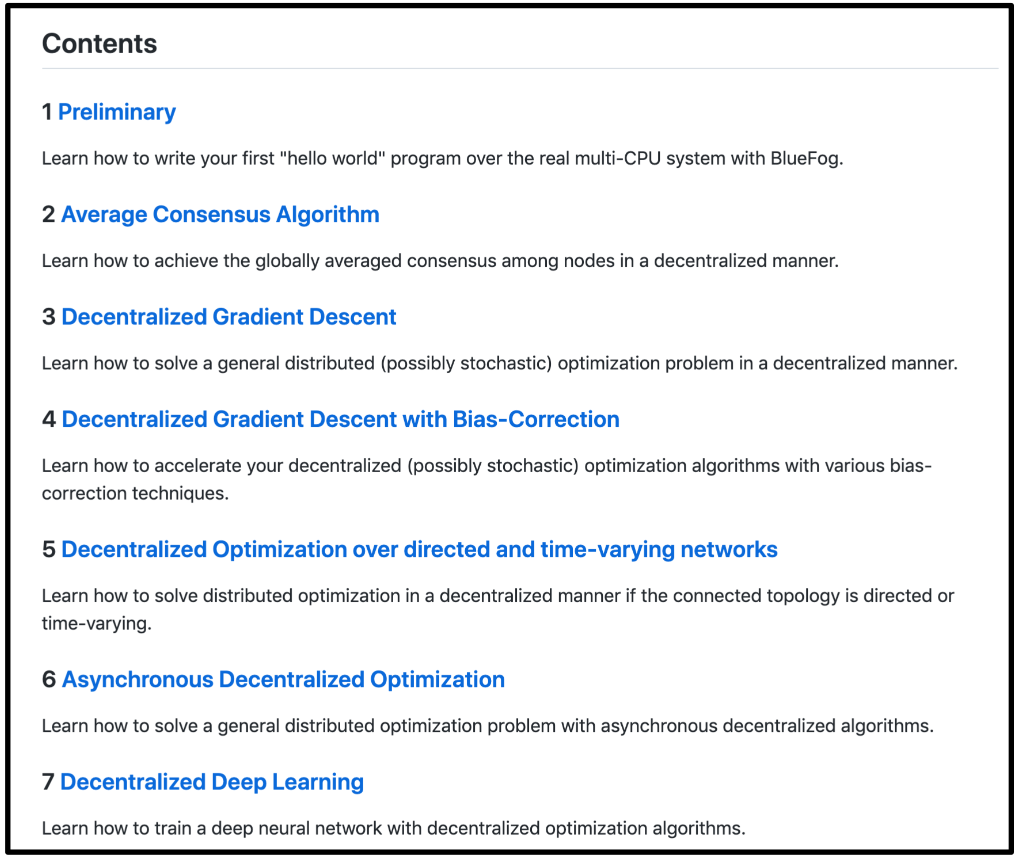
Example: A ExponentialTwoGraph with 12 nodes:

```
>>> import networkx as nx
>>> from bluefog.common import topology_util
>>> G = topology_util.ExponentialTwoGraph(12)
>>> nx.draw_circular(G)
```

bluefog.common.topology_util.ExponentialGraph(*size: int, base: int = 2*) → networkx.classes.digraph.DiGraph

Easy-to-use

Detailed tutorials



Contents

1 Preliminary

Learn how to write your first "hello world" program over the real multi-CPU system with BlueFog.

2 Average Consensus Algorithm

Learn how to achieve the globally averaged consensus among nodes in a decentralized manner.

3 Decentralized Gradient Descent

Learn how to solve a general distributed (possibly stochastic) optimization problem in a decentralized manner.

4 Decentralized Gradient Descent with Bias-Correction

Learn how to accelerate your decentralized (possibly stochastic) optimization algorithms with various bias-correction techniques.

5 Decentralized Optimization over directed and time-varying networks

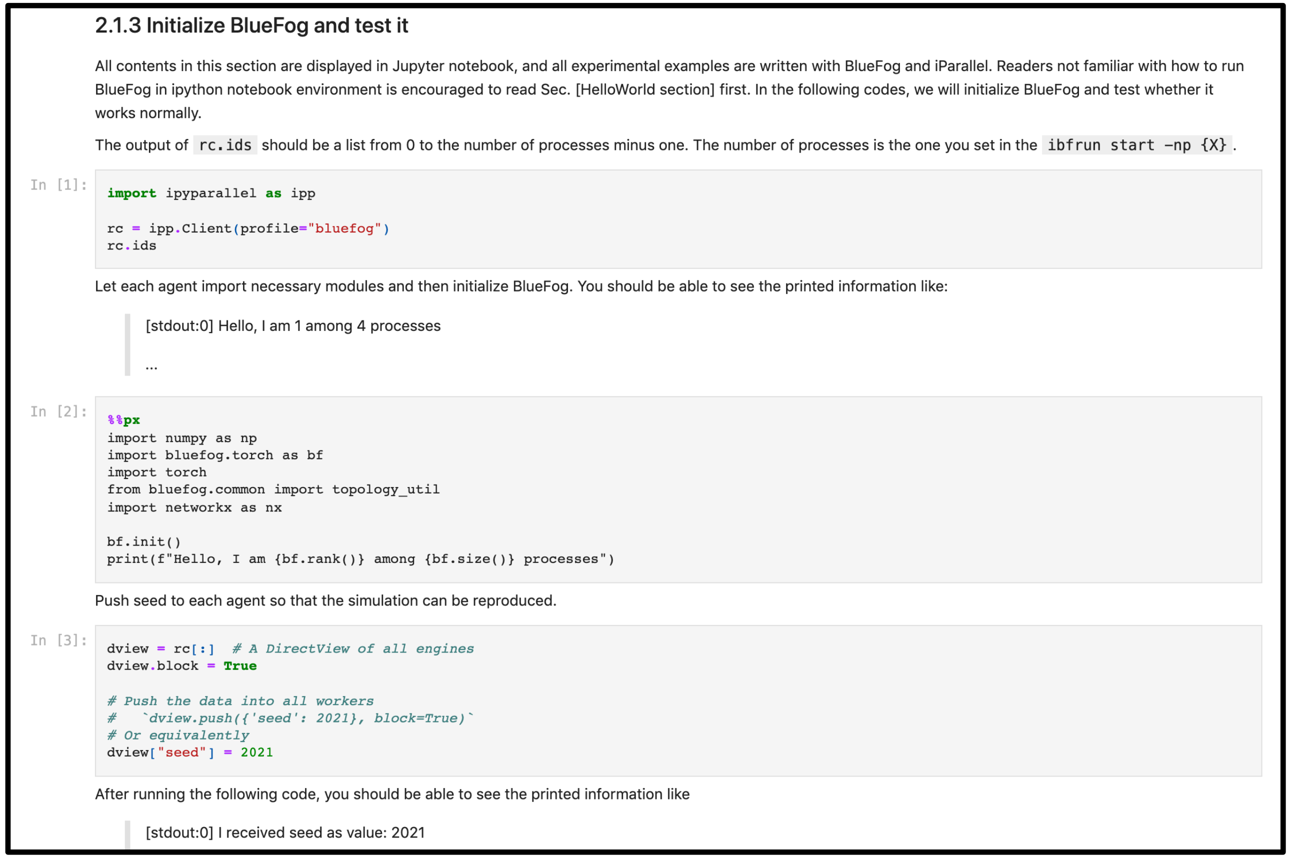
Learn how to solve distributed optimization in a decentralized manner if the connected topology is directed or time-varying.

6 Asynchronous Decentralized Optimization

Learn how to solve a general distributed optimization problem with asynchronous decentralized algorithms.

7 Decentralized Deep Learning

Learn how to train a deep neural network with decentralized optimization algorithms.



2.1.3 Initialize BlueFog and test it

All contents in this section are displayed in Jupyter notebook, and all experimental examples are written with BlueFog and iParallel. Readers not familiar with how to run BlueFog in ipython notebook environment is encouraged to read Sec. [HelloWorld section] first. In the following codes, we will initialize BlueFog and test whether it works normally.

The output of `rc.ids` should be a list from 0 to the number of processes minus one. The number of processes is the one you set in the `ibfrun start -np {X}`.

In [1]:

```
import ipyparallel as ipp

rc = ipp.Client(profile="bluefog")
rc.ids
```

Let each agent import necessary modules and then initialize BlueFog. You should be able to see the printed information like:

> [stdout:0] Hello, I am 1 among 4 processes
>
> ...

In [2]:

```
%%px
import numpy as np
import bluefog.torch as bf
import torch
from bluefog.common import topology_util
import networkx as nx

bf.init()
print(f"Hello, I am {bf.rank()} among {bf.size()} processes")
```

Push seed to each agent so that the simulation can be reproduced.

In [3]:

```
dview = rc[:]  # A DirectView of all engines
dview.block = True

# Push the data into all workers
#   `dview.push({'seed': 2021}, block=True)`
# Or equivalently
dview["seed"] = 2021
```

After running the following code, you should be able to see the printed information like

> [stdout:0] I received seed as value: 2021

Final summary

- Decentralized algorithms save remarkable communication compared to centralized ones
- Sparse and effective topologies make decentralized optimization practical for deep training
- We propose static exponential, one-peer exponential, and one-peer EquiDyn and justify their superiority with strong theoretical and experimental evidences
- We introduced BlueFog to facilitate research and implementation of decentralized methods

References

L Ding, K Jin, B Ying, K Yuan, W Yin, DSGD-CECA: Decentralized SGD with Communication-Optimal Exact Consensus Algorithm, ICML 2023

Z. Song, W. Li, K. Jin, L. Shi, M. Yan, W. Yin, K. Yuan, *Communication-efficient topologies for decentralized learning with O(1) consensus rate*, NeurIPS 2022

B Ying, K Yuan, Y Chen, H Hu, P Pan, W Yin, *Exponential Graph is Provably Efficient for Decentralized Deep Training*, NeurIPS 2021

B Ying, K Yuan, H Hu, Y Chen, W Yin. *BlueFog: Make decentralized algorithms practical for optimization and deep learning*, arXiv:2111.04287, 2021

Thank you!

Kun Yuan homepage: https://kunyuan827.github.io/

BlueFog homepage: https://github.com/Bluefog-Lib/bluefog