Implement Graph-specific PyTorch Dataloader using DALI and cuGraph

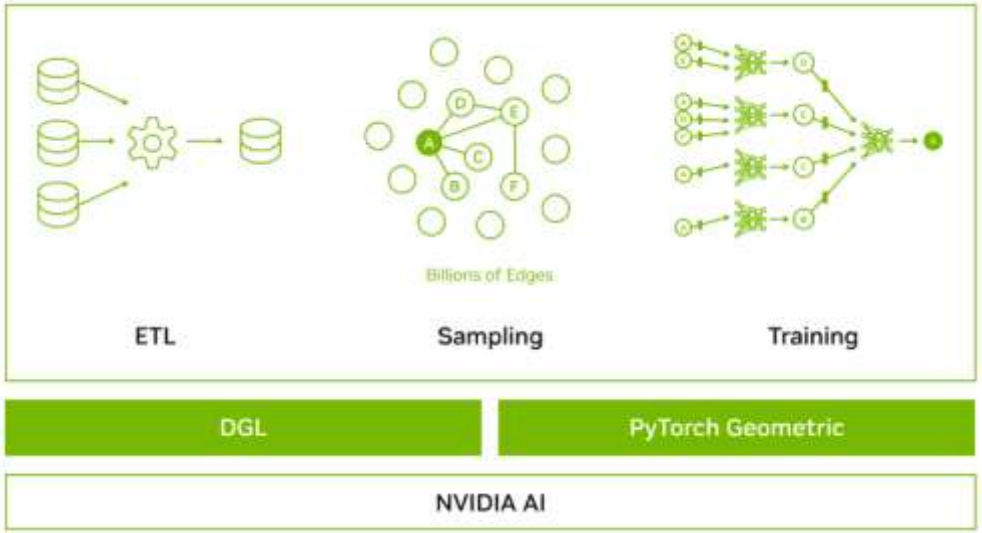
20230531

박현우

Table of Contents

1. NVIDIA Data Loading Library (DALI)

NVIDIA Data Loading Library is a component of NVIDIA Graph Neural Network Frameworks.



1. NVIDIA Data Loading Library (DALI)

Most of preprocessing tasks typically used to run on the CPU,
due to simplicity, flexibility, and availability of libraries.

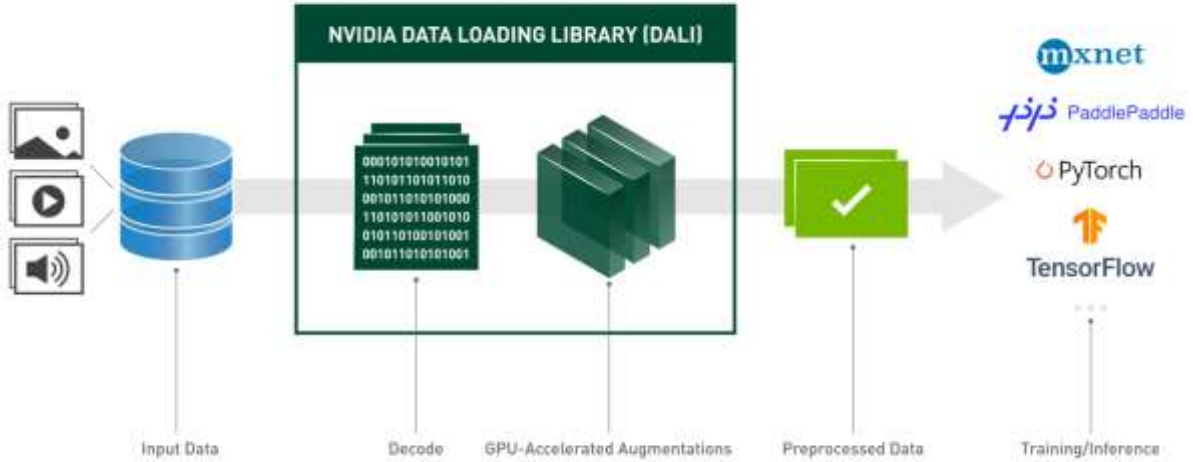
Meanwhile, recent advances in GPU architectures have significantly increased GPU throughput.

Hereby, Dense multi-GPU systems train a model much faster than data can be provided by the input pipeline,
leaving the GPUs starved for data.

1. NVIDIA Data Loading Library (DALI)

NVIDIA Data Loading Library (DALI) is a result of our efforts
to find a scalable and portable solution to the data pipeline issues;

i.e. shifting centroid of preprocessing from CPU to GPU



1.1 DALI Concepts

An important feature of DALI is plugins,
which can be used as drop-in replacements for frameworks' native datasets;

by

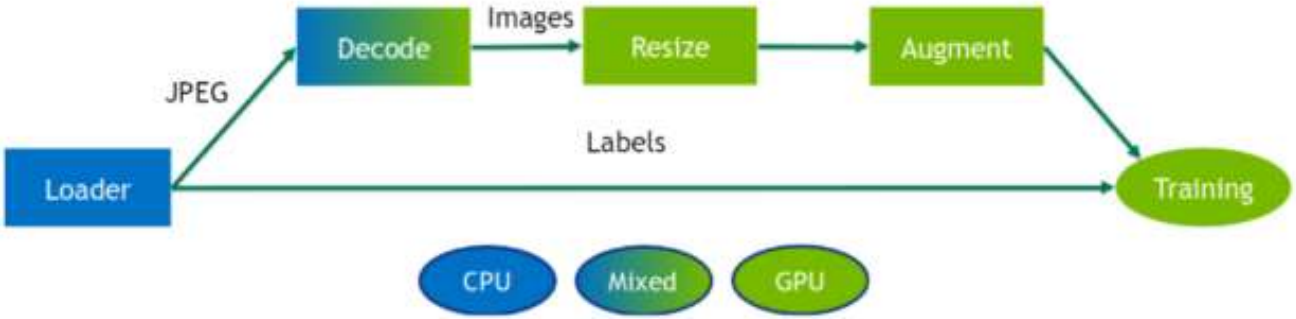
1. Defining DALI pipeline once
2. wrapping it with data iterator wrapper of any supported frameworks.

```
from nvidia.dali.plugin.pytorch import DALIGenericIterator

train_loader = DALIClassificationIterator([pipe], reader_name='Reader')

for i, data in enumerate(train_loader):
    images = data[0]["data"]
    target = data[0]["label"].squeeze(-1).long()
    # model training
```

1.1 DALI Concepts

The main entity in DALI is the data processing pipeline:

A pipeline is defined by a symbolic graph of data nodes connected by operators and run in an asynchronous fashion.

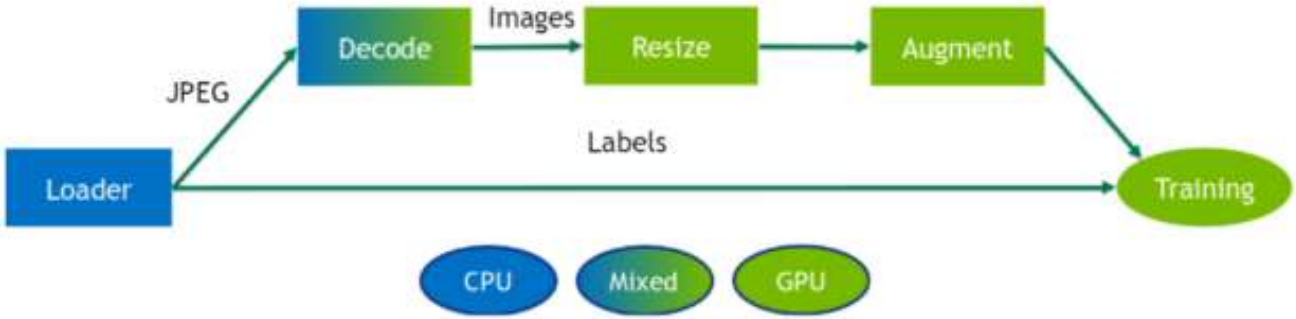


1.1 DALI Concepts

DALI can offload processing to the GPU by utilizing special operators with a Mixed backend,

that receives

1. inputs placed on CPU memory
2. and output data placed in GPU memory.

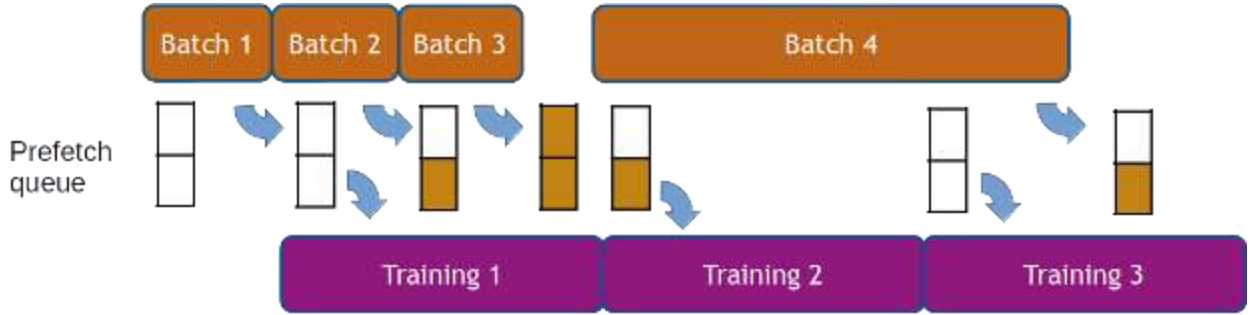


1.2 DALI optimization features

As DALI’s execution is asynchronous, DALI allows for data prefetching,

that is, preparing batches of data ahead of time before they are requested so that the framework has always data ready for the next iteration.

DALI handles data prefetching transparently for the user, with a configurable prefetch queue.



1.2 DALI optimization features

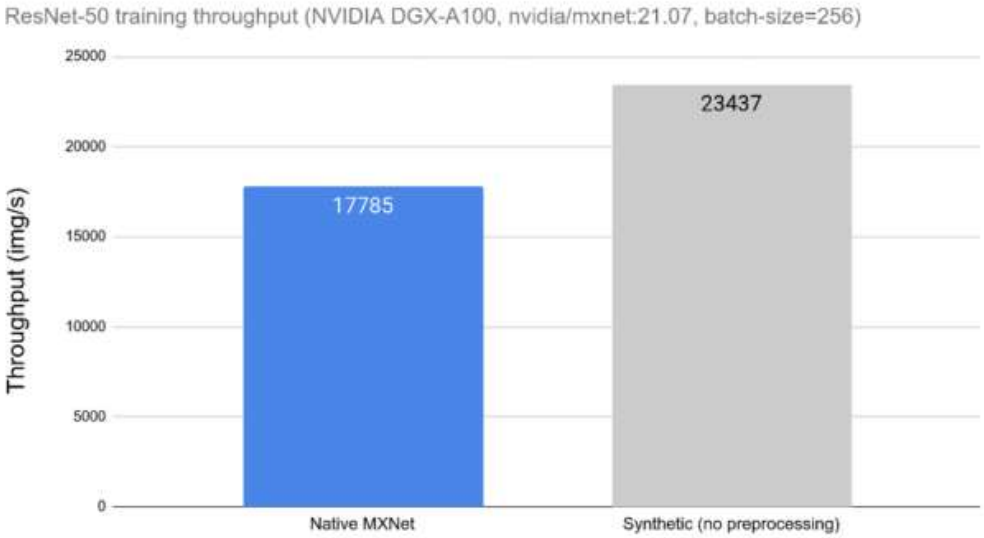
Other performance optimization strategies are supported:

1. **Operator Buffer Presizing**
2. **Prefetching Queue Depth**
3. **File/Data Readers fine-tuning**

@@@ Read on Notion

https://docs.nvidia.com/deeplearning/dali/user-guide/docs/advanced_topics_performance_tuning.html#

1.3 DALI Performance



(Right side)

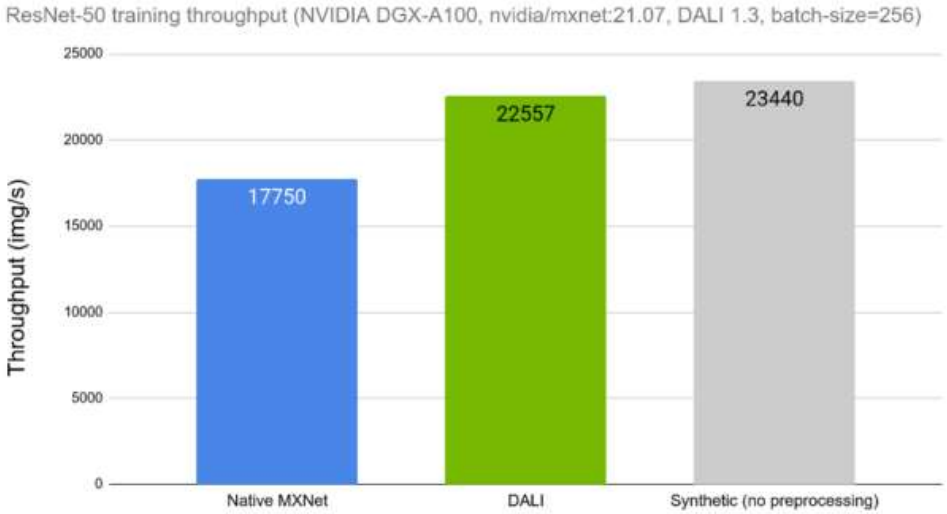
performance of the same network

without the impact of data loading and preprocessing,

by replacing it with synthetic data.

= can be used as a theoretical upper limit.

1.3 DALI Performance

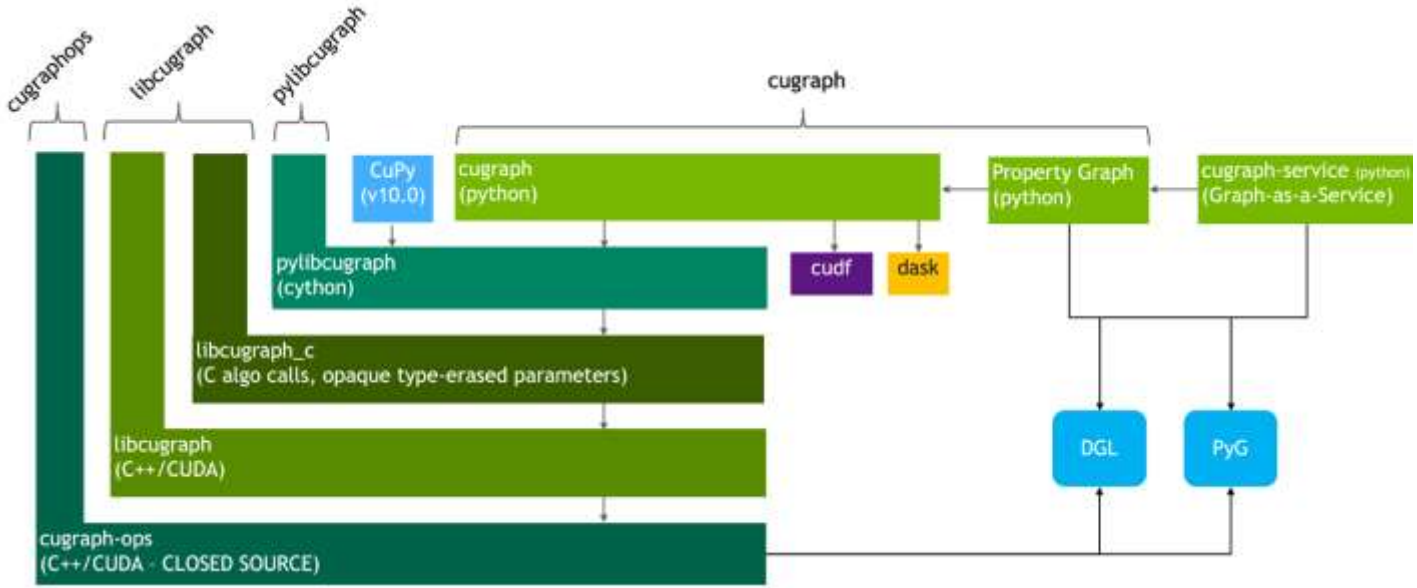


We can see

how the training throughput with DALI is much closer to the theoretical upper limit (synthetic example).

2. cuGraph

RAPIDS cuGraph is a collection of GPU-accelerated graph creation/manipulation and algorithms.



2.1. Problem of other graph libraries

Currently, by default, DGL (and PyTorch Geometric) depend for graph creation and manipulation on NetworkX and simple PyTorch tensor.

The problem is that NetworkX is merely pure python hashable object;

i.e. not supported by CUDA, thus involving conversion overhead.

> **Note**
>
> DGL internally converts SciPy matrices and NetworkX graphs to tensors to construct graphs. Hence, these construction methods are not meant for performance critical parts.

2.1. Problem of other graph libraries

Although DGL also implements Graph types using PyTorch tensor,
such approach is not fully optimized by CUDA;

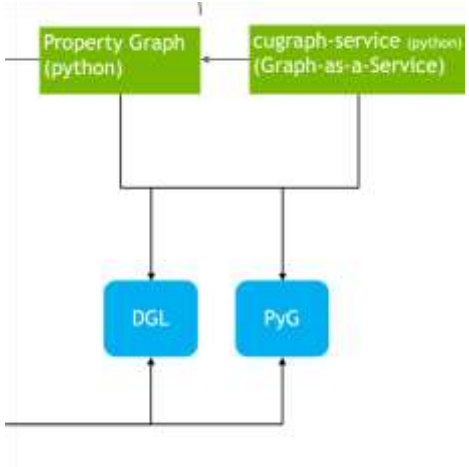
for example, no CUDA optimization for CSR (Compressed Sparse Row).

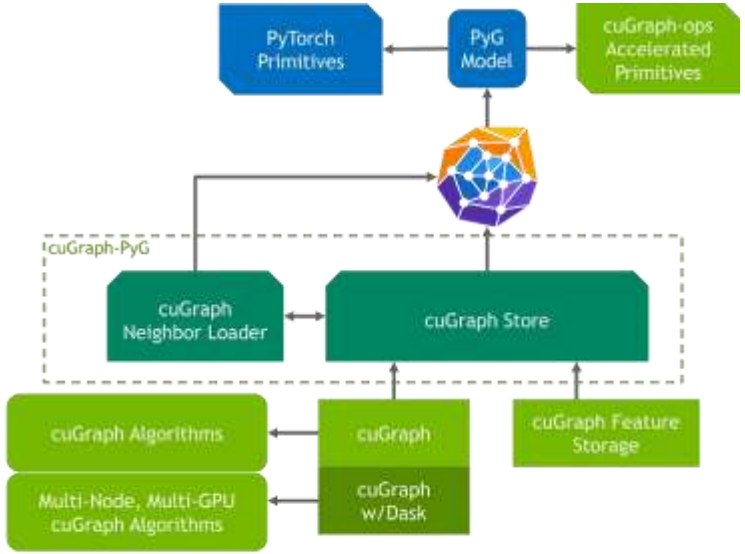
Also, DGL itself lacks of CUDA optimized algorithms;

e.g. Multi-GPU pagerank algorithm or renumbering algorithms

2.2. Concepts of cuGraph

cuGraph is fully supported by CUDA and provides NetworkX-like API,
enabling drop-in replacement for NetworkX-using frameworks (i.e. DGL and PyG).





Support for Multi-Node, Multi-GPU algorithms with *no code change*

2.2. Concepts of cuGraph

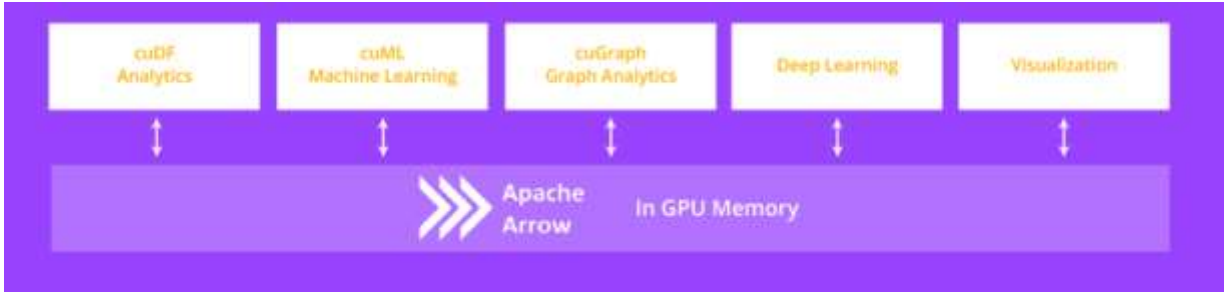
Example of DGL adoption of cugraph:

```
+from cugraph_dgl.convert import cugraph_storage_from_heterograph
+cugraph_g = cugraph_storage_from_heterograph(dgl_g)

sampler = dgl.dataloading.NeighborSampler(
        [15, 10, 5], prefetch_node_feats=['feat'], prefetch_labels=['label'])

train_dataloader = dgl.dataloading.DataLoader(
- dgl_g,
+ cugraph_g,
train_idx,
sampler,
device=device,
batch_size=1024,
shuffle=True,
drop_last=False,
num_workers=0)
```

2.2. Concepts of cuGraph

Specifically, cuGraph operates on GPU DataFrames or cuDF, thereby allowing for seamless passing of data.

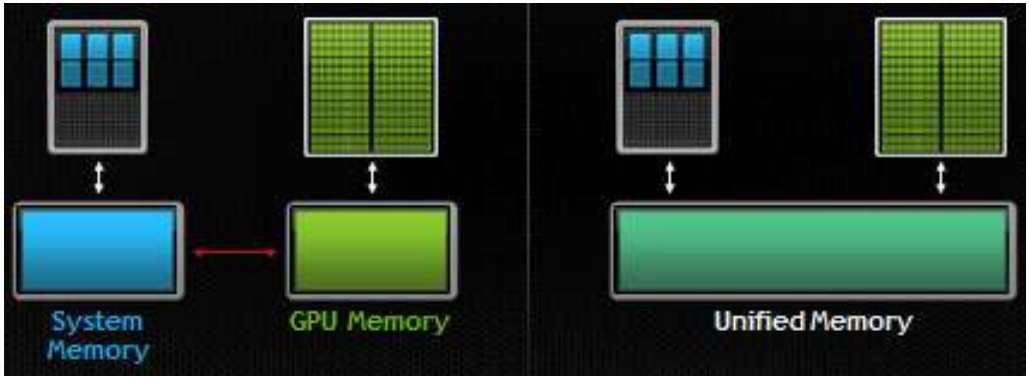
RAPIDS cuDF is GPU version of Apache Arrow and thus eliminates serialization overhead by leveraging Arrow memory format.



2.3. CUDA Unified memory and Oversubscription of cuGraph

Unified memory is a single memory address space accessible from any processor in a system.
i.e. Host memory + Device memory

The data movement is done automatically. (by RAPIDS Memory Manager (RMM))



2.3. CUDA Unified memory and Oversubscription of cuGraph

Oversubscription is simply

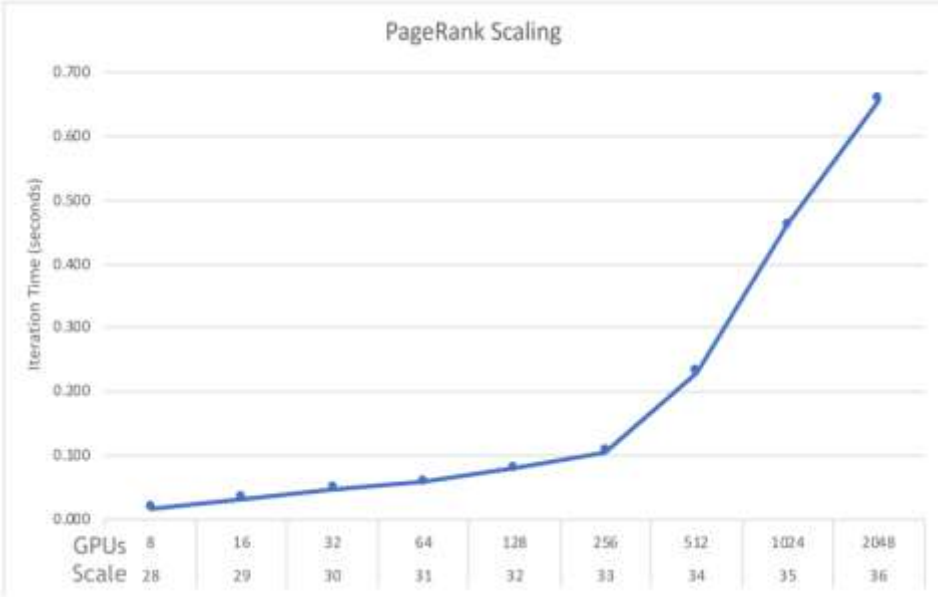
1. the ability to allocate GPU memory larger than what is physically available on the device,
2. and have the GPU automatically page in data as needed.

Pascal and later GPU architectures include hardware to accelerate unified memory data movement (paging);

i.e. Paging Migration Engine

2.4. cuGraph Performance

cuGraph scales smoothly from small graphs on 1 GPU to massive graphs with trillions of edges on 2,048 GPUs. (Tested on Selene cluster)

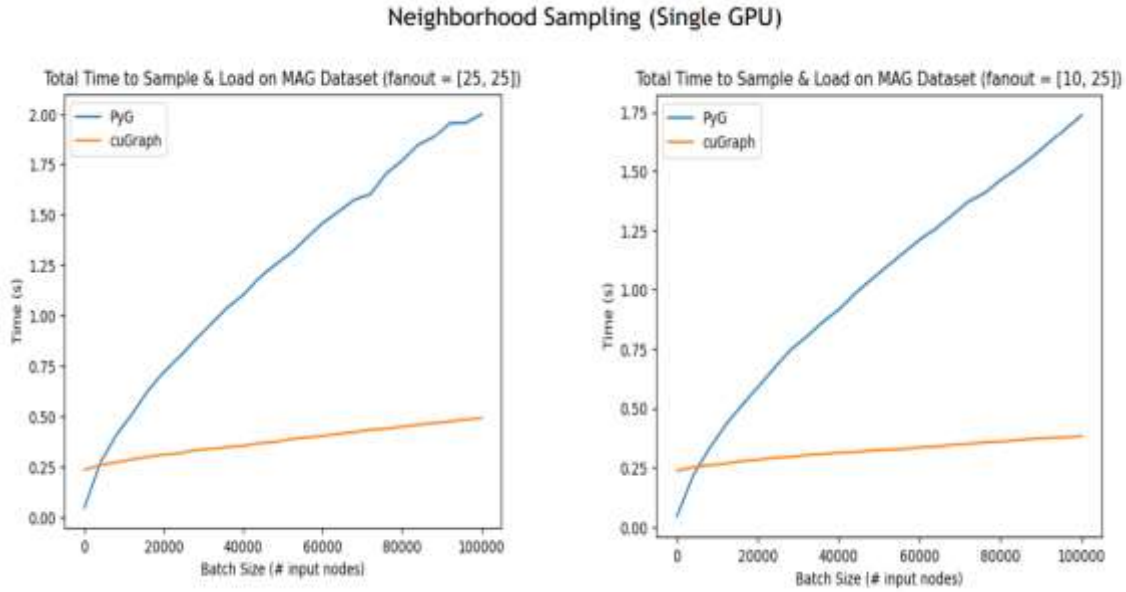


Benchmark of PageRank on Scale 36 dataset (1.1 trillion directed edges):

19.3 seconds (0.66 seconds per iteration, 2,048 GPUs)

2.4. cuGraph Performance

In Single-GPU environment, cuGraph outperforms PyG (NetworkX + PyTorch tensor) on Neighboring sampling algorithms:



3. Proposal: adopt cuGraph into DALI

To sum up,

DALI

1. Plugin nature helps simple integration with many DL frameworks
2. Provide rich pre-built preprocessing operators and optimization strategies.

cuGraph

1. Fully supported by CUDA
2. Provide scalability and CUDA-optimized algorithms
3. Utilize CUDA Unified memory and Oversubscription

3. Proposal: adopt cuGraph into DALI

Currently, DALI supports following storage formats or ExternalSource operator:

1. LMDB,
2. RecordIO (in MXNet),
3. TFRecord (in TensorFlow)