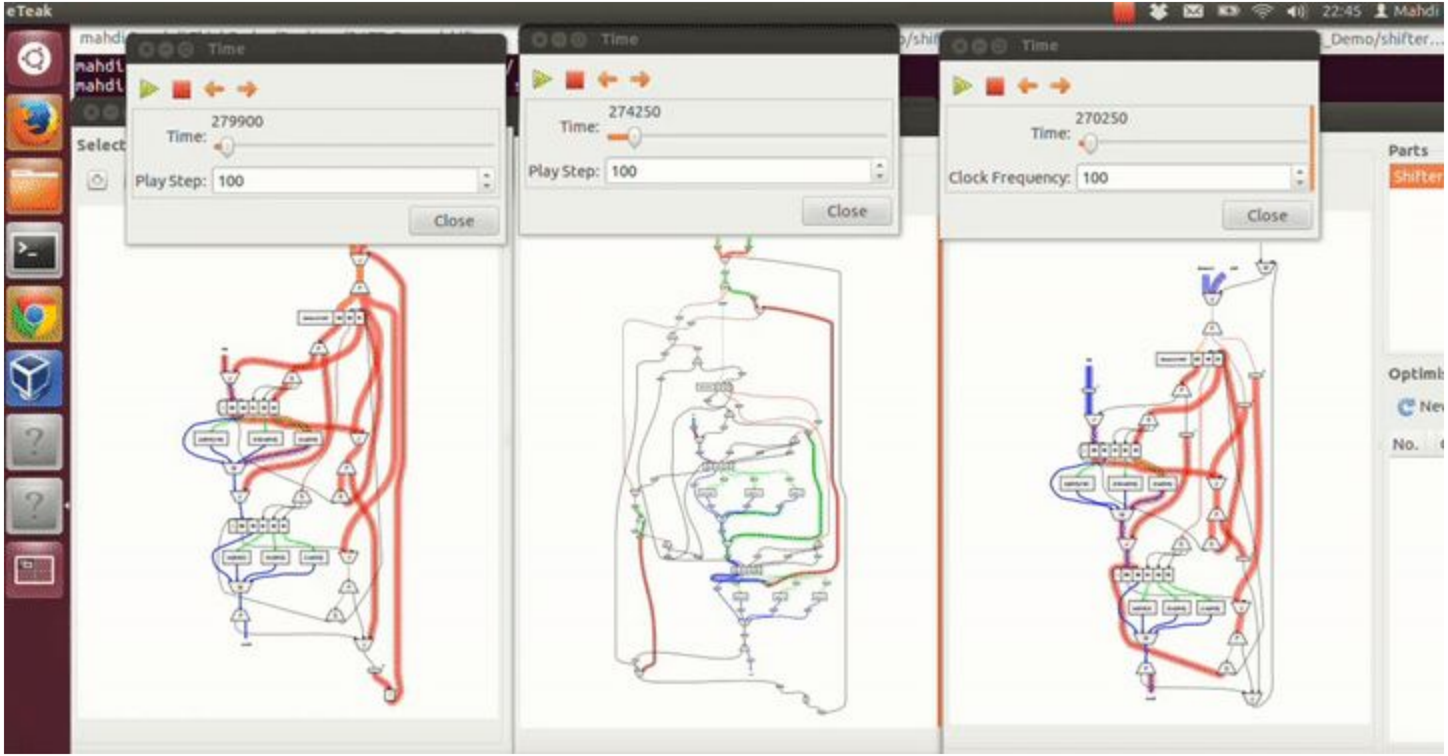
Graph Neural Networks on CPUs: Enabling Affordable and Distributed Training and Inference

Mahdi Jelodari, PhD

BSC , Spain

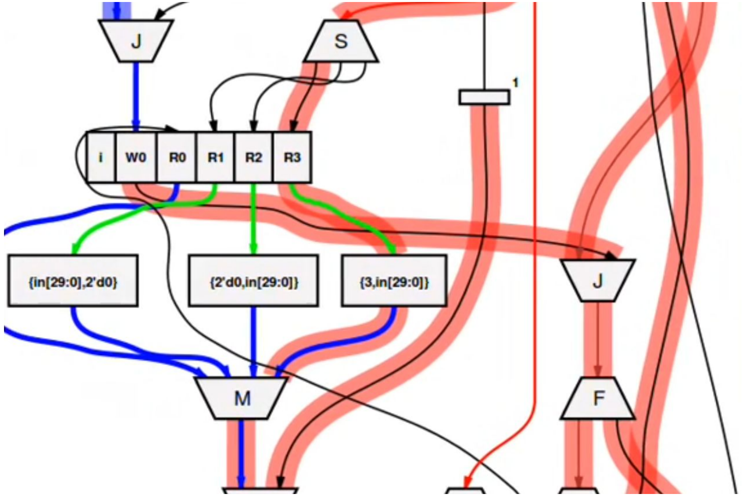
April 2023

Background with Graphs - DFGs



https://github.com/balangs/eTeak
http://apt.cs.manchester.ac.uk/people/mamagham/MJ_Mamaghani_ACSD13.pdf
Visualizer - courtesy of Andrew Bardsley

Data+Control flow Graphs (DCFG)



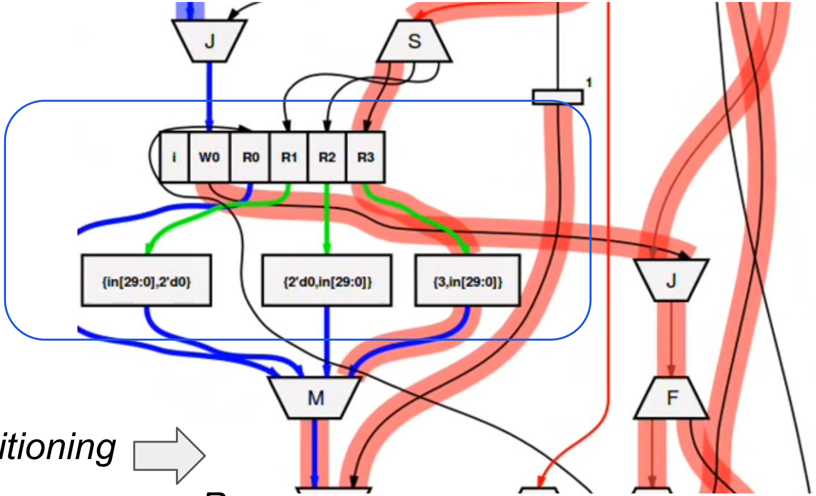

```
import [balsa.types.basic]
import [ror]

--test ror32
procedure test_ror32(output o : 32 bits)
is
  variable i : 5 bits
  channel shiftchan : 32 bits
  channel distchan : 5 bits
begin
  begin
    i:= 1;
    loop
      shiftchan <- 7 || distchan <- i;
      i:= (i+1 as 5 bits)
    while i < 31 end
  end || ror32(distchan, shiftchan, o)
end -- begin
```

https://github.com/balangs/eTeak
http://apt.cs.manchester.ac.uk/people/mamagham/MJ_Mamaghani_ACSD13.pdf
Visualizer - courtesy of Andrew Bardsley

Graph Primitives: Join (J), Fork (F), Steer (S), Merge (M), Operator (O)

Data+Control flow Graphs (DCFG)



Partitioning ⇨

Power
Throughput
Area

```
import [balsa.types.basic]
import [ror]

--test ror32
procedure test_ror32(output o : 32 bits)
is
  variable i : 5 bits
  channel shiftchan : 32 bits
  channel distchan : 5 bits
begin
  begin
    i:= 1;
    loop
      shiftchan <- 7 || distchan <- i;
      i:= (i+1 as 5 bits)
    while i < 31 end
  end || ror32(distchan, shiftchan, o)
end -- begin
```

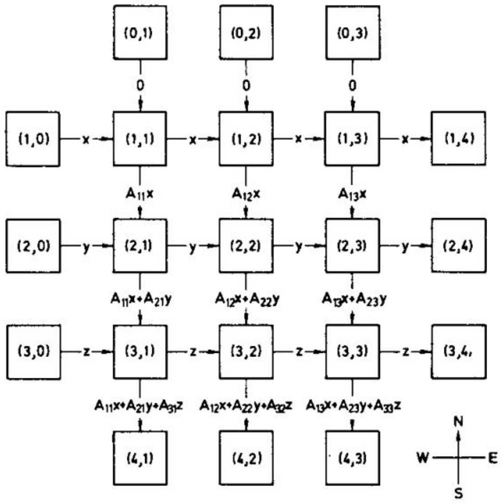
Graph Primitives: Join (J), Fork (F), Steer (S), Merge (M), Operator (O)

https://github.com/balangs/eTeak
http://apt.cs.manchester.ac.uk/people/mamagham/MJ_Mamaghani_ACSD13.pdf
Visualizer - courtesy of Andrew Bardsley

Communicating Sequential Processes (CSP)

High abstraction level - it is CSP based communicating sequential processes

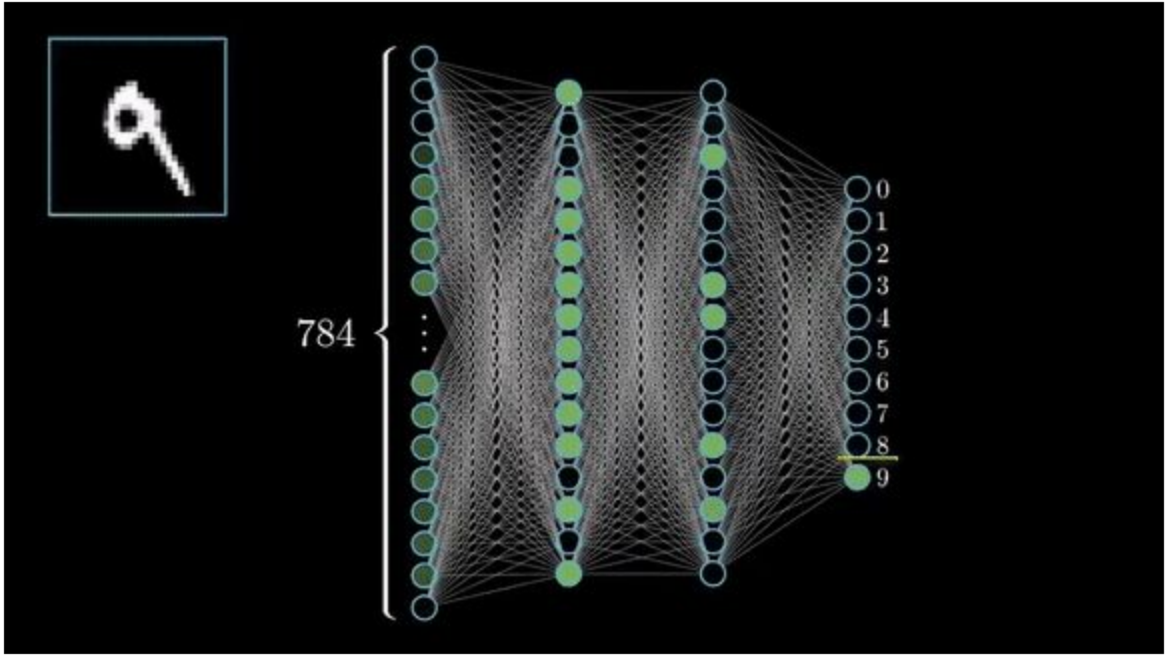
- A square matrix A of order 3 is given
- Three streams representing columns of an array IN are input
- Three streams representing columns of the product matrix IN × A are output
- Results produced at the same rate as input is consumed, requiring high parallelism
- Each non-border node inputs vector component from west and partial sum from north
- Each node outputs vector component to east and updated partial sum to south
- West border nodes produce input data, south border nodes consume desired results
- North border is constant source of zeros and east border is a sink
- No provision needed for termination or changing values of matrix A.



Tony Hoare's paper https://www.cs.cmu.edu/~crary/819-f09/Hoare78.pdf

Artificial Neural Networks (ANN) - Convolutional

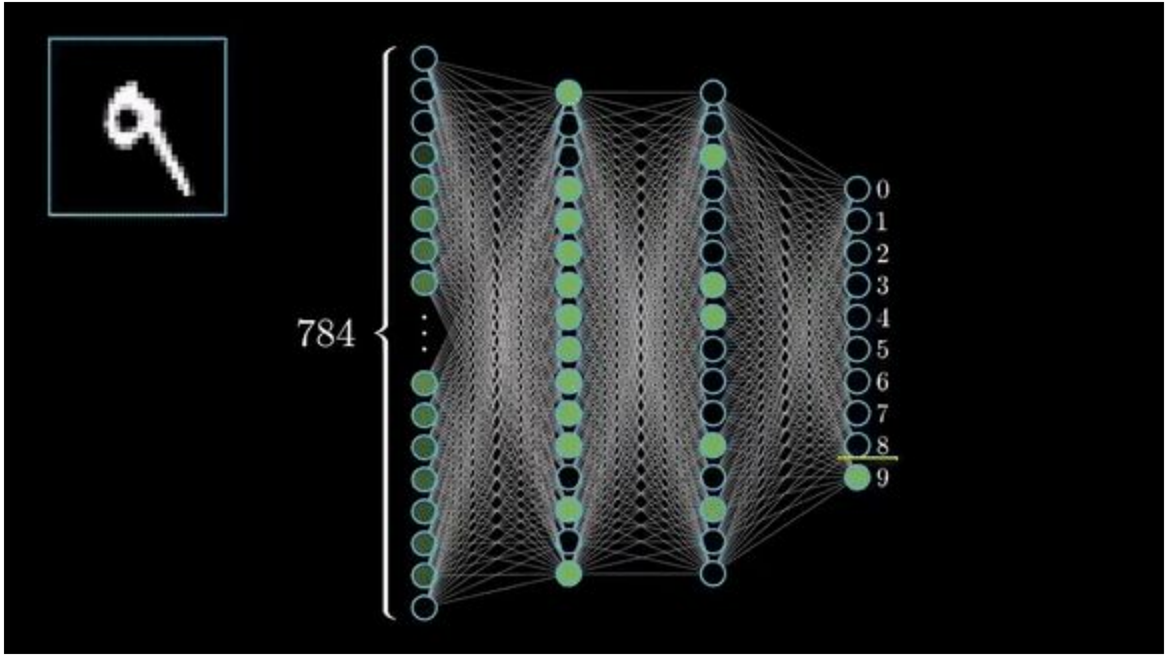
Designed to Structured data such as images and videos.
They are composed of multiple layers of convolutional filters and pooling layers,
which allow them to extract features from input data and classify it into different categories.



Artificial Neural Networks (ANN) - Convolutional

Designed to Structured data such as images and videos.
They are composed of multiple layers of convolutional filters and pooling layers,
which allow them to extract features from input data and classify it into different categories.

- AlexNet (2012)
- VGG (2014)
- GoogLeNet (2014)
- ResNet (2015)
- DenseNet (2016)
- YOLO (2016)
- Mask R-CNN (2017)
- EfficientNet (2019)
- Detectron2 (2020)



Graph Neural Networks (GNN)

Graphs are all around us; real world objects are often defined in terms of their connections to other things. A set of objects, and the connections between them, are naturally expressed as a *graph*. Researchers have developed neural networks that operate on graph data (called graph neural networks, or GNNs) for over a decade. Recent developments have increased their capabilities and expressive power. We are starting to see practical applications in areas such as

- antibacterial discovery ,
- physics simulations ,
- fake news detection ,
- traffic prediction
- recommendation systems .

Graph Neural Networks (GNN)

Graphs are all around us; real world objects are often defined in terms of their connections to other things. A set of objects, and the connections between them, are naturally expressed as a *graph*. Researchers have developed neural networks that operate on graph data (called graph neural networks, or GNNs) for over a decade. Recent developments have increased their capabilities and expressive power. We are starting to see practical applications in areas such as

- antibacterial discovery ,
- physics simulations ,
- fake news detection ,
- traffic prediction
- recommendation systems .

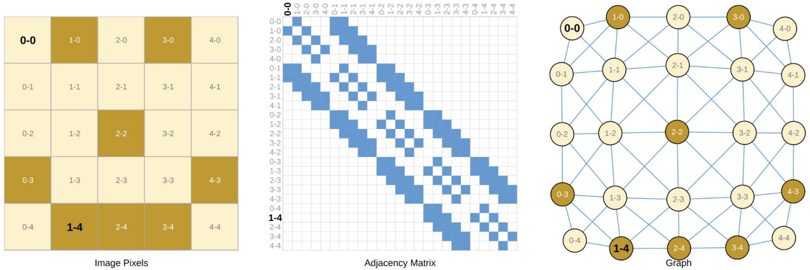
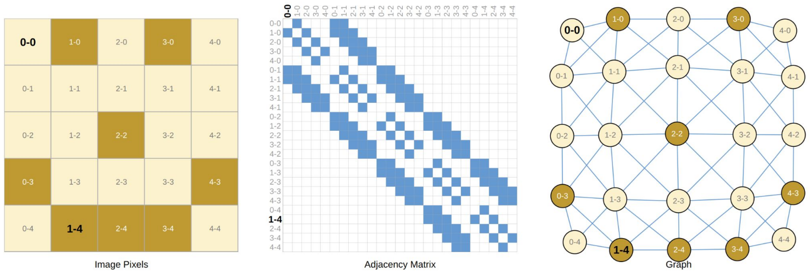


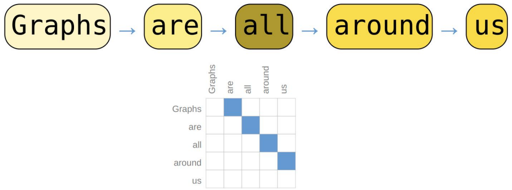
Image Pixels   Adjacency Matrix   Graph

Graph Neural Networks (GNN)

Graphs are all around us; real world objects are often defined in terms of their connections to other things. A set of objects, and the connections between them, are naturally expressed as a *graph*. Researchers have developed neural networks that operate on graph data (called graph neural networks, or GNNs) for over a decade. Recent developments have increased their capabilities and expressive power. We are starting to see practical applications in areas such as

- antibacterial discovery ,
- physics simulations ,
- fake news detection ,
- traffic prediction
- recommendation systems .

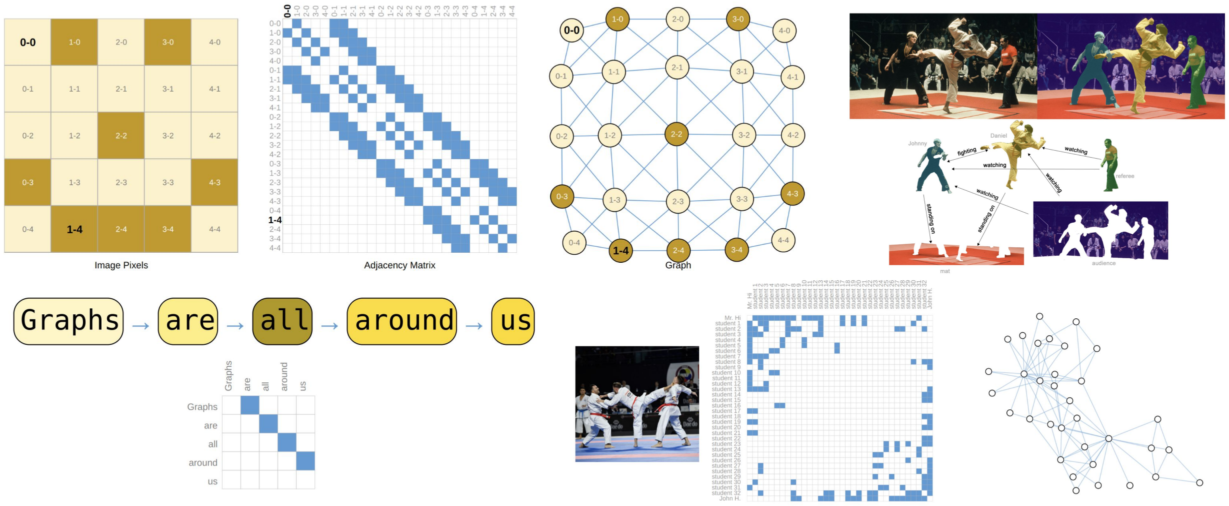




Graph Neural Networks (GNN)

Graphs are all around us; real world objects are often defined in terms of their connections to other things. A set of objects, and the connections between them, are naturally expressed as a *graph*. Researchers have developed neural networks that operate on graph data (called graph neural networks, or GNNs) for over a decade. Recent developments have increased their capabilities and expressive power. We are starting to see practical applications in areas such as

- antibacterial discovery ,
- physics simulations ,
- fake news detection ,
- traffic prediction
- recommendation systems .



Graph Neural Networks (GNN)

DGL DEEP GRAPH LIBRARY

PyG

Creating Message Passing Networks

Generalizing the convolution operator to irregular domains is typically expressed as a *neighborhood aggregation* or *message passing* scheme. With $\mathbf{x}_i^{(k-1)} \in \mathbb{R}^F$ denoting node features of node i in layer $(k-1)$ and $\mathbf{e}_{j,i} \in \mathbb{R}^D$ denoting (optional) edge features from node j to node i, message passing graph neural networks can be described as

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right),$$

Jraph - A library for graph neural networks in jax.

Deepmind https://github.com/deepmind/jraph
Facebook https://pytorch-geometric.readthedocs.io/en/latest/tutorial/create_gnn.html
Amazon AI https://github.com/dmlc/dgl

Message Passing Neural Networks (MPNN)

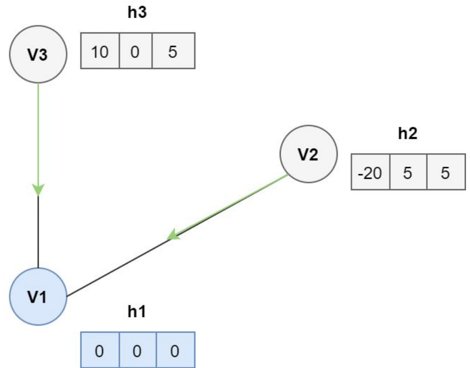
1. MPNNs are able to incorporate information from the local structure of the graph into their representations, allowing them to capture relationships between nodes and their neighbors.
2. MPNNs can be applied to a wide range of tasks, including node classification, link prediction, and graph regression, among others.
3. MPNNs are able to handle graphs of arbitrary size and structure, making them more flexible than traditional neural networks that operate on fixed-size inputs.
4. MPNNs have achieved state-of-the-art performance on several benchmark tasks, demonstrating their effectiveness in learning from graph-structured data.

Message Passing for Node V1
for t = 1

$$m_v^{t+1} = \sum_{w \in N(v)} h_w^t$$

$$h_v^{t+1} = average(h_v, m_v^{t+1})$$

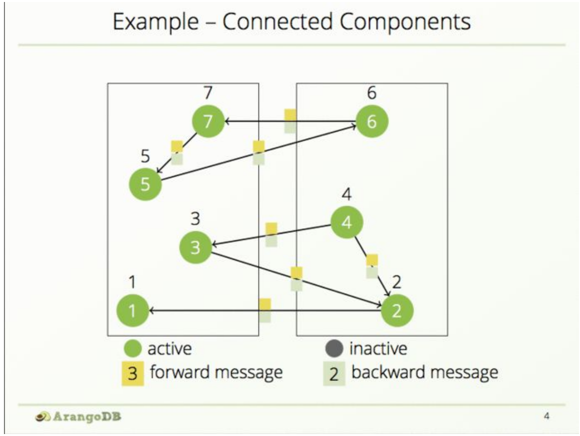
ht - hidden state for each node



‘Think like a vertex’

1. Google's PageRank algorithm: Google originally developed the Pregel programming model for its PageRank algorithm, which is used to rank web pages in search results. The PageRank algorithm involves processing a massive graph of web pages and links between them, making Pregel an ideal tool for the task.
2. Twitter's social network analysis: Twitter has used Pregel to analyze its massive social network graph, which includes billions of nodes and edges. The analysis helps Twitter to identify trends and patterns in user behavior, as well as to develop more effective algorithms for recommending content to users.
3. Facebook's social graph analysis: Facebook has used Apache **Giraph**, a graph processing framework that is based on the Pregel model, to analyze its massive social graph. The analysis helps Facebook to identify patterns in user behavior, to personalize content for users, and to improve its advertising algorithms.

This is specifically from Giraph page:https://cwiki.apache.org/confluence/display/GIRAPH



Benchmarking GNNs

(a) MNIST (b) CIFAR10

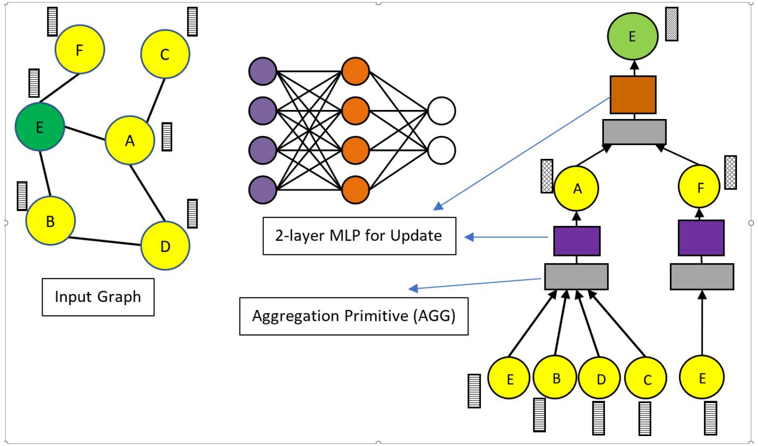
Model	L	#Param	Test Acc.±s.d.	Train Acc.±s.d.	#Epoch	Epoch/Total	#Param	Test Acc.±s.d.	Train Acc.±s.d.	#Epoch	Epoch/Total
MLP	4	104044	95.340±0.138	97.432±0.470	232.25	22.74s/1.48hr	104380	56.340±0.181	65.113±1.685	185.25	29.48s/1.53hr
vanilla GCN	4	101365	90.705±0.218	97.196±0.223	127.50	83.41s/2.99hr	101657	55.710±0.381	69.523±1.948	142.50	109.70s/4.39hr
GraphSage	4	104337	**97.312±0.097**	100.000±0.000	98.25	113.12s/3.13hr	104517	65.767±0.308	99.719±0.062	93.50	124.61s/3.29hr
GCN	4	101365	90.120±0.145	96.459±1.020	116.75	37.06s/1.22hr	101657	54.142±0.394	70.163±3.429	140.50	47.16s/1.86hr
MoNet	4	104049	90.805±0.032	96.609±0.440	146.25	93.19s/3.82hr	104229	54.655±0.518	65.911±2.515	141.50	97.13s/3.85hr
GAT	4	110400	95.535±0.205	99.994±0.008	104.75	42.26s/1.25hr	110704	64.223±0.455	89.114±0.499	103.75	55.27s/1.62hr
GatedGCN	4	104217	**97.340±0.143**	100.000±0.000	96.25	128.79s/3.50hr	104357	**67.312±0.311**	94.553±1.018	97.00	154.15s/4.22hr
GIN	4	105434	96.485±0.252	100.000±0.000	128.00	39.22s/1.41hr	105654	55.255±1.527	79.412±9.700	141.50	52.12s/2.07hr
RingGNN	2	105398	11.350±0.000	11.235±0.000	14.00	2945.69s/12.77hr	105165	19.300±16.108	19.556±16.397	13.50	3112.96s/13.00hr
	2	505182	91.860±0.449	92.169±0.505	16.25	2575.99s/12.63hr	504949	39.165±17.114	40.209±17.790	13.75	2998.24s/12.60hr
	8	506357	Diverged	Diverged	Diverged	Diverged	510439	Diverged	Diverged	Diverged	Diverged
3WLGNN	3	108024	95.075±0.961	95.830±1.338	27.75	1523.20s/12.40hr	108516	59.175±1.593	63.751±2.697	28.50	1506.29s/12.60hr
	3	501690	95.002±0.419	95.692±0.677	26.25	1608.73s/12.42hr	502770	58.043±2.512	61.574±3.575	20.00	2091.22s/12.55hr
	8	500816	Diverged	Diverged	Diverged	Diverged	501584	Diverged	Diverged	Diverged	Diverged

Benchmarking Graph Neural Networks
https://arxiv.org/pdf/2003.00982.pdf
28 Dec 2022

Single-node and Multi-node GNN training

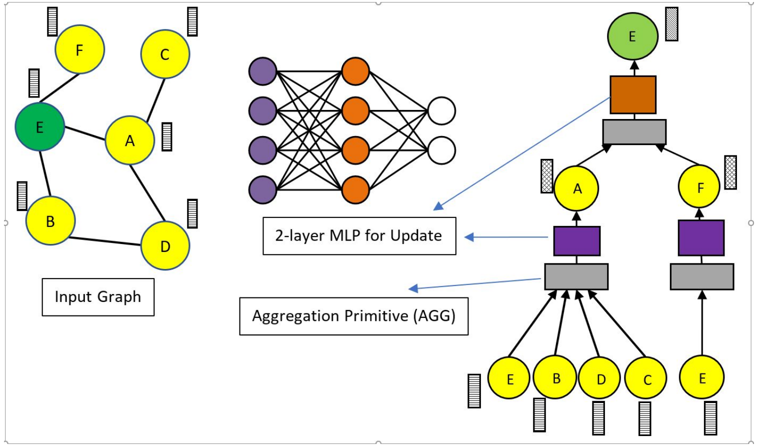
Extensive benchmarking by Intel on Xeon 32 core and 72 core processors

1. Irregular memory accesses to read or *gather* feature-vectors of each vertex and its neighbors from memory as indicated by source vertex indices (**memory bond**)
2. Apply ⊗ on each *Fv*, (*Fv*, *Fv*), or (*Fv*, *Fe*) tuple to perform unary or binary operations, respectively (**compute bond**)
3. Apply ⊕ to reduce each source vertex or edge feature-vector (**compute bond**)
4. Irregular memory accesses to write or *scatter* the resulting feature-vector the Tensor object in memory as indicated by destination vertex indices in the adjacency matrix. (**memory bond**)



DistGNN: Scalable Distributed Training for Large-Scale Graph Neural Networks
https://arxiv.org/pdf/2104.06700.pdf SC'21

Single-node and Multi-node GNN training

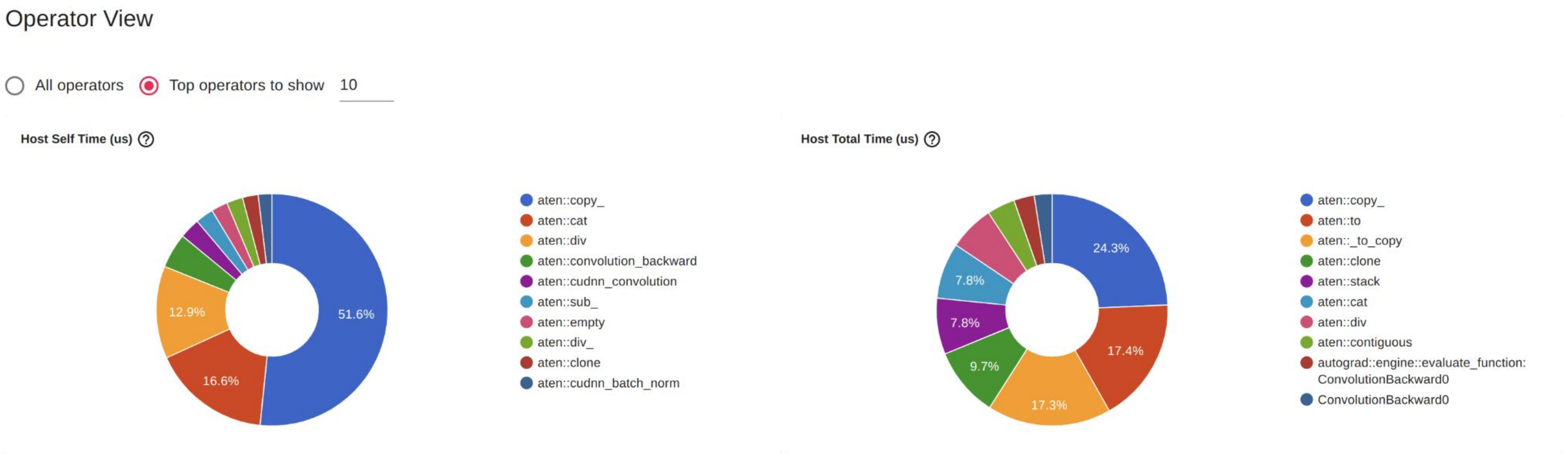


1. Irregular memory accesses to read or *gather* feature-vectors of each vertex and its neighbors from memory as indicated by source vertex indices (**memory bond**)
2. Apply ⊗ on each *Fv*, (*Fv*, *Fv*), or (*Fv*, *Fe*) tuple to perform unary or binary operations, respectively (**compute bond**)
3. Apply ⊕ to reduce each source vertex or edge feature-vector (**compute bond**)
4. Irregular memory accesses to write or *scatter* the resulting feature-vector the Tensor object in memory as indicated by destination vertex indices in the adjacency matrix. (**memory bond**)

apply cache blocking on the source vertex feature-vector tensor, potentially accessing the whole destination vertex feature-vector tensor repeatedly -> 4x faster training
Available under DGL

DistGNN: Scalable Distributed Training for Large-Scale Graph Neural Networks
https://arxiv.org/pdf/2104.06700.pdf SC'21

Operator-Level Execution Time



Pytorch profiler + Tensorboard https://pytorch.org/tutorials/recipes/recipes/profiler_recipe.html

Support from the community

Posted by u/perone 7 months ago

[P] Tutorial on using LLVM to JIT PyTorch graphs to native code (x86/arm/RISC-V)

Project

Just made a tutorial on how to create a simple JIT compiler for PyTorch graphs using LLVM for those who might be interested in compilers for ML. I did a small detour talking about PyTorch's NNC and how to generate native code for different architectures such as x86, ARM and RISC-V for those who are interested [here is the link](#).

Next Steps - Under Excelencia Severo Ochoa Grant

Incorporate in-house acceleration paradigms*

Advancements on RISC-V vector processing units

Look into establishing a benchmark for health data: video + text + EHR

VIA: A Smart Scratchpad for Vector Units with Application to Sparse Matrix Computations
https://upcommons.upc.edu/bitstream/handle/2117/346345/BSC_DS-2021-19_VIA%20A%20Smart%20Scratchpad.pdf?sequence=1&isAllowed=y

RISC-V Instruction Set Extension for Graph Applications
https://carrv.github.io/2022/papers/CARRV2022_paper_8_Yenimol.pdf

Thank you, Please stay in touch!

Linkedin https://www.linkedin.com/in/mahdijelodari/

Twitter: https://twitter.com/MJcomp86