USENIX ATC ’19, Renton, WA, USA

NeuGraph:

Parallel Deep Neural Network Computation on Large Graphs

Lingxiao Ma [†], Zhi Yang [†], Youshan Miao [‡], Jilong Xue [‡], Ming Wu [‡], Lidong Zhou [‡], Yafei Dai [†]

† *Peking University*

‡ *Microsoft Research*

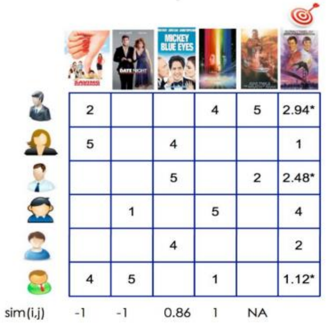
北京大學 PEKING UNIVERSITY

Microsoft Research

* Figures from Internet

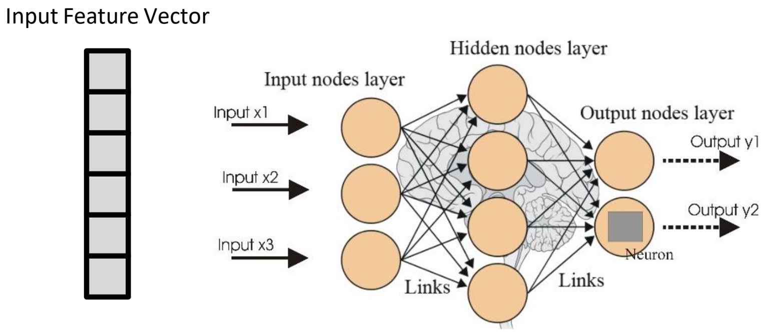
Self-Driving

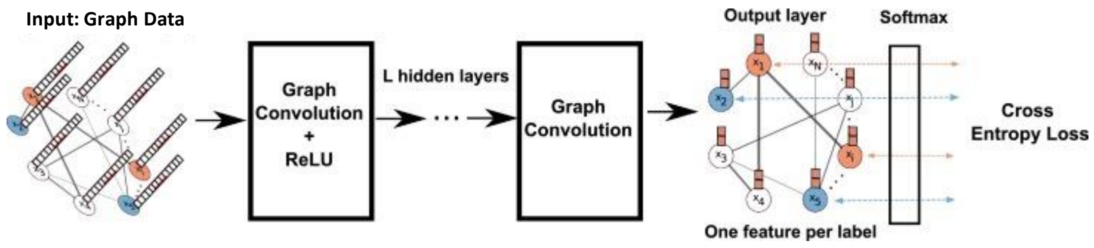
Personal Assistant



Recommendation

Question Answering





Neural Networks

Graph Neural Networks

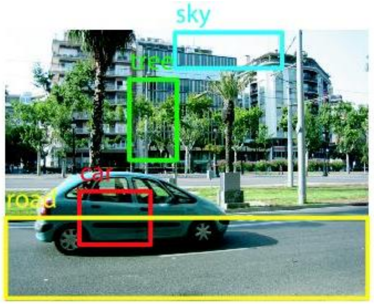
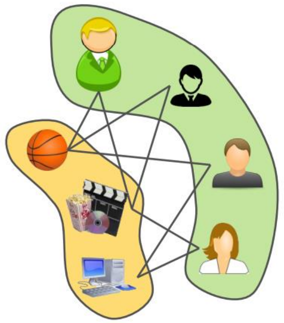
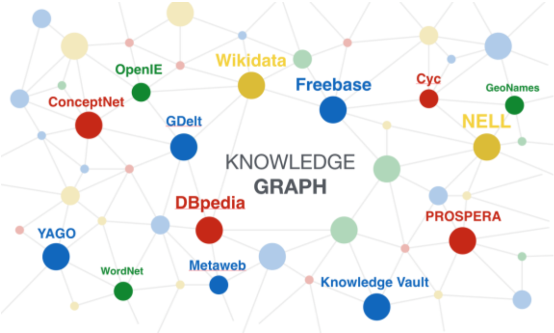


Image Object Detection

Speech Recognition



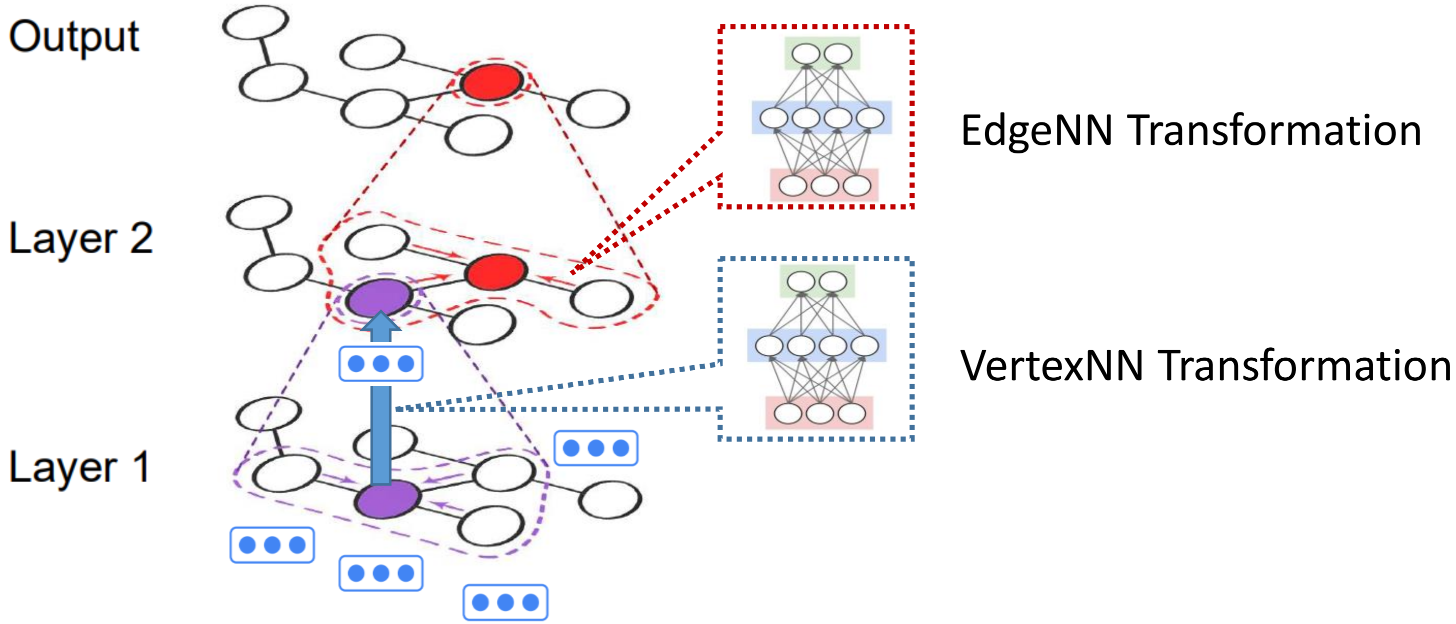
User-Item Graph



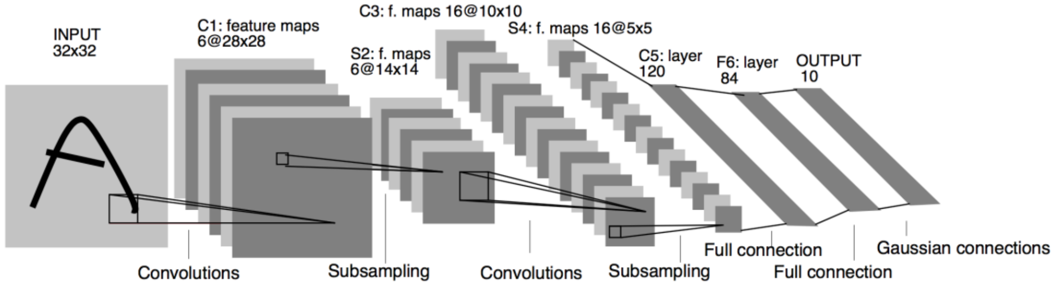
Knowledge Graph

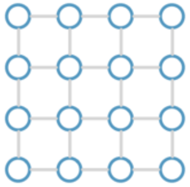
Graph Neural Networks (GNN)

- Information propagation via *Graph*
- Information transformation via *Neural Networks*

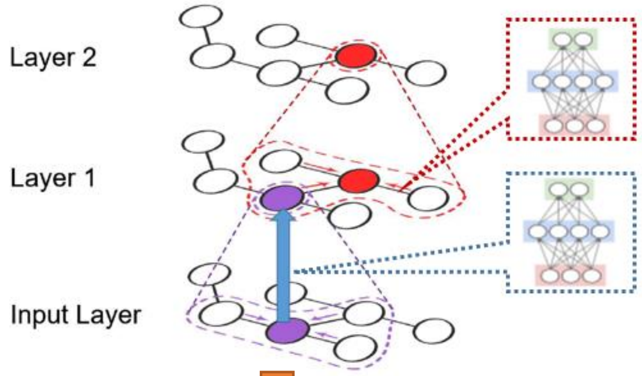


Challenges in Processing GNNs on GPU




2D grid

small and regular




Graphs

large, sparse, and irregular

Scalability Problem
Efficiency Problem

Existing Systems are Insufficient

Deep Learning Systems

TensorFlow

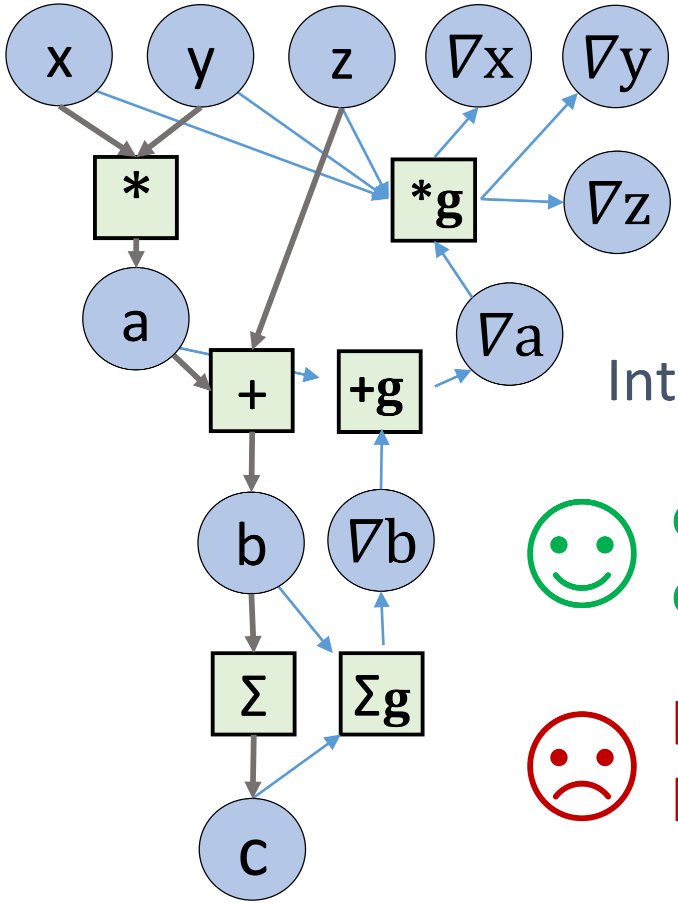
PyTorch mxnet

```
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
z = tf.placeholder(tf.float32)

a = x * y
b = a + z
c = tf.reduce_sum(b)

grad_x,grad_y,grad_z = tf.gradients(c, [x,y,z])

with tf.Session() as sess:
  sess.run([grad_z], feed_dict=values)
```



Dataflow Graph
as
Intermediate Representation

easy to express NNs
efficient for grid structures

hard to express graph ops
hard to handle large graphs

Existing Systems are Insufficient

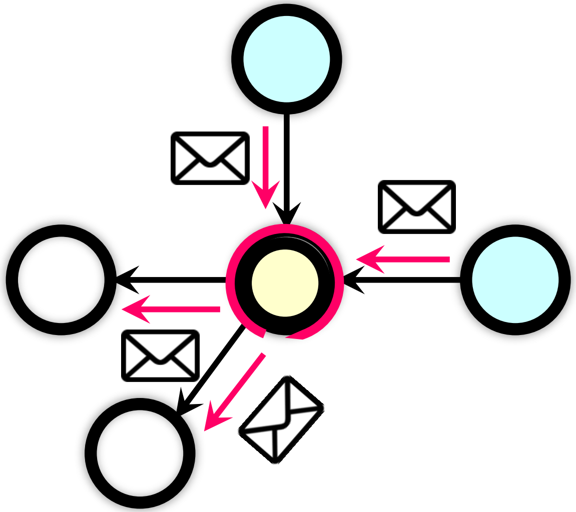
Graph Computing Systems

APACHE GIRAPH GraphLab PowerLyra

```
Gather(D_u, D_(u,v), D_v):
    return D_v.rank / #outNbrs(v)
Sum(a, b):
    return a + b

Apply(D_u, acc):
    r_new = 0.15 + 0.85 * acc
    D_u.delta = (r_new - D_u.rank) / #outNbrs(u)
    D_u.rank = r_new

Scatter(D_u, D_(u,v), D_v):
    if(|D_u.delta|>ε) Activate(v)
    return delta
```



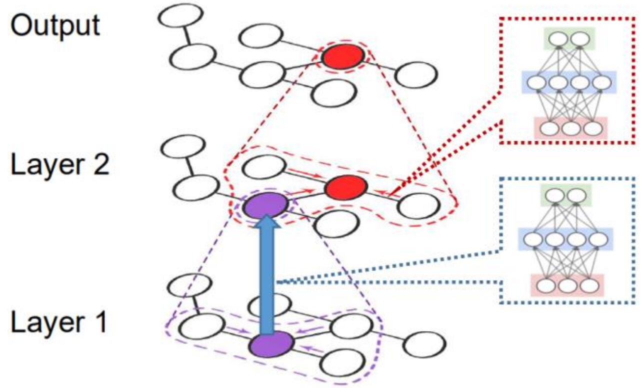
Vertex Programming
e.g.: GAS

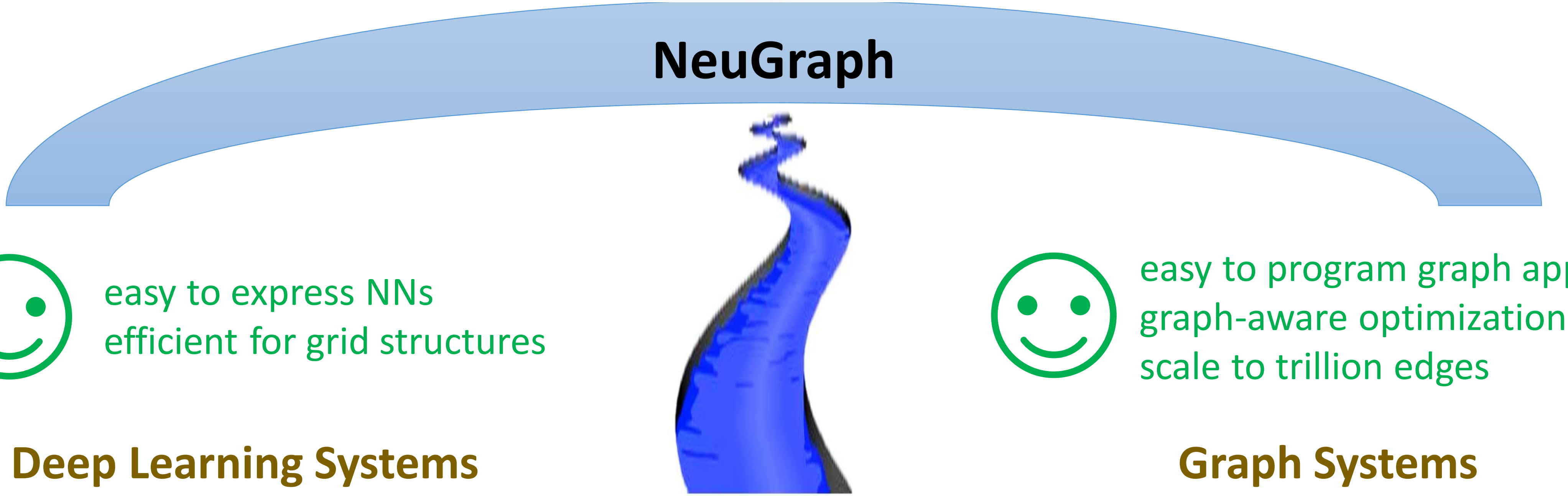
easy to program graph apps
graph-aware optimizations
scale to trillion edges

hard to express NNs (e.g., no backprop.)
insufficient NN execution (e.g., vertex-by-vertex)

We propose: NeuGraph

- Bridge graph and dataflow models to support ***efficient*** and ***scalable*** GNN processing





easy to express NNs
efficient for grid structures

Deep Learning Systems

easy to program graph apps
graph-aware optimizations
scale to trillion edges

Graph Systems

NeuGraph

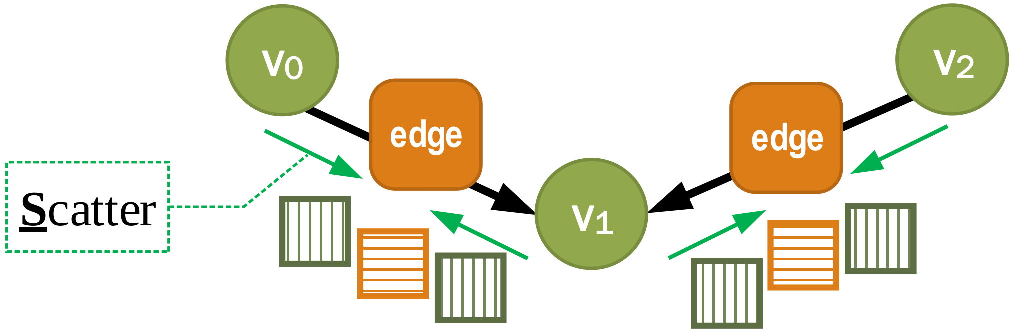
- Bridge graph and dataflow models to support ***efficient*** and ***scalable*** GNN processing
- **Key techniques**
  - SAGA-NN model for graph-based neural networks

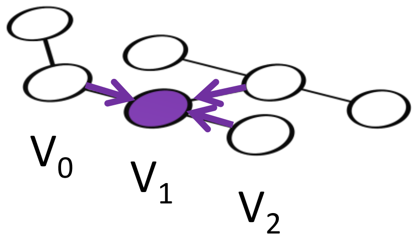
    -> programming GNN apps
  - Chunk-based dataflow graph translation & streaming processing out of GPU core

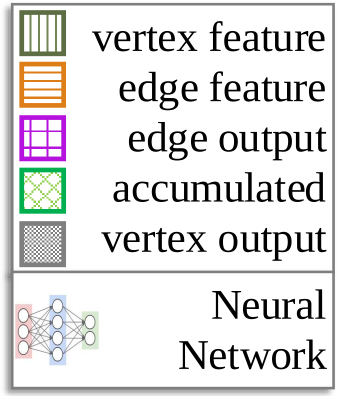
    -> processing graphs larger than GPU memory
  - Highly-optimized graph propagation operators
  - Chain-based parallel streaming

    -> efficient multi-GPU parallel execution
- **Performance**
  - Outperform state-of-the-art frameworks (e.g., TensorFlow and DGL) on small graphs
  - Scale to large real-world graphs with GPUs

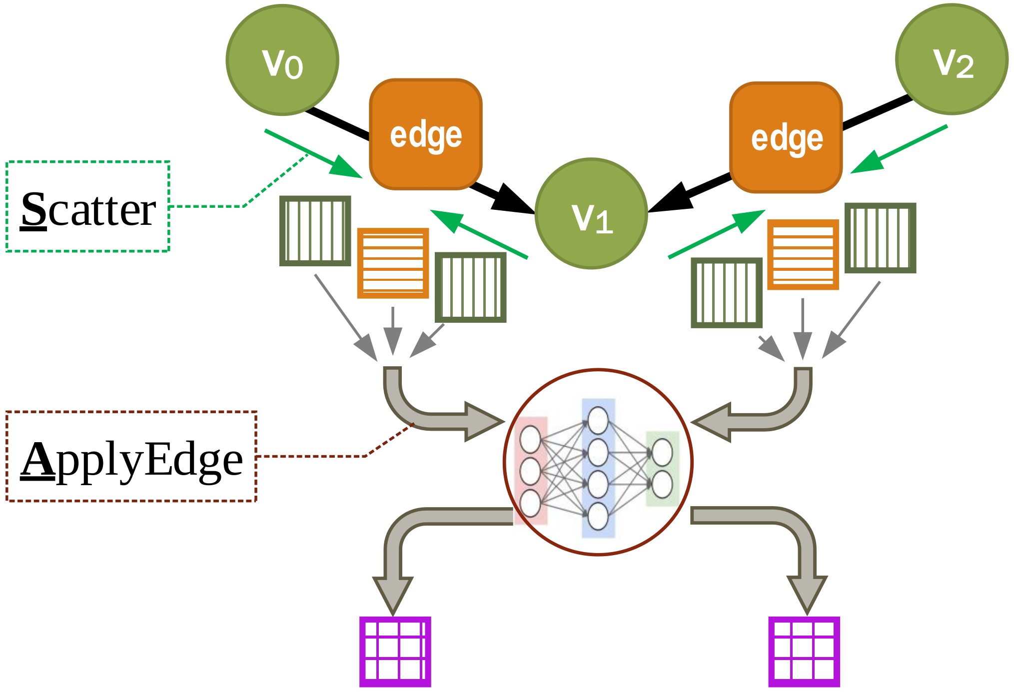
SAGA-NN Abstraction

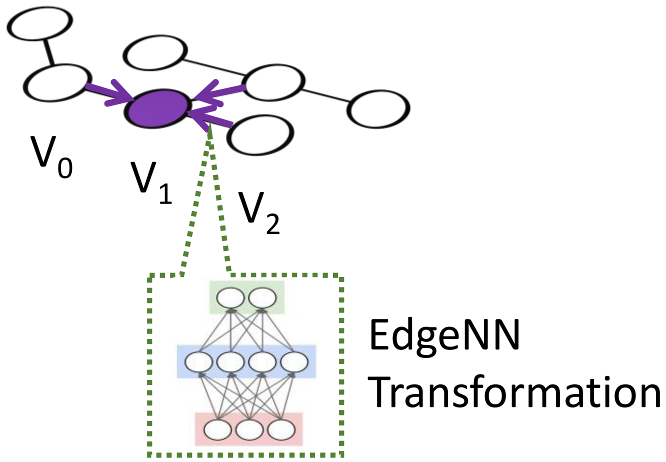


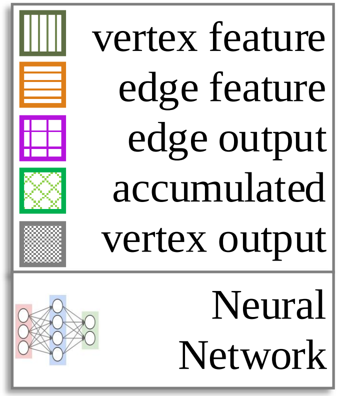




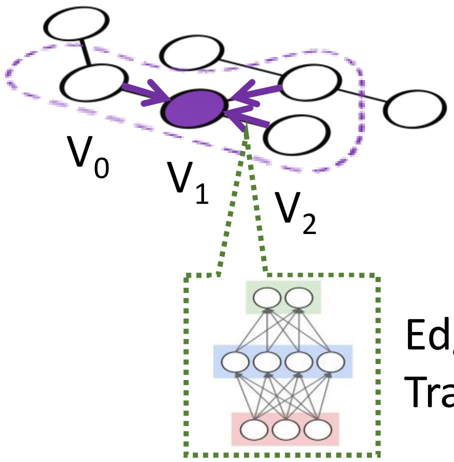
SAGA-NN Abstraction



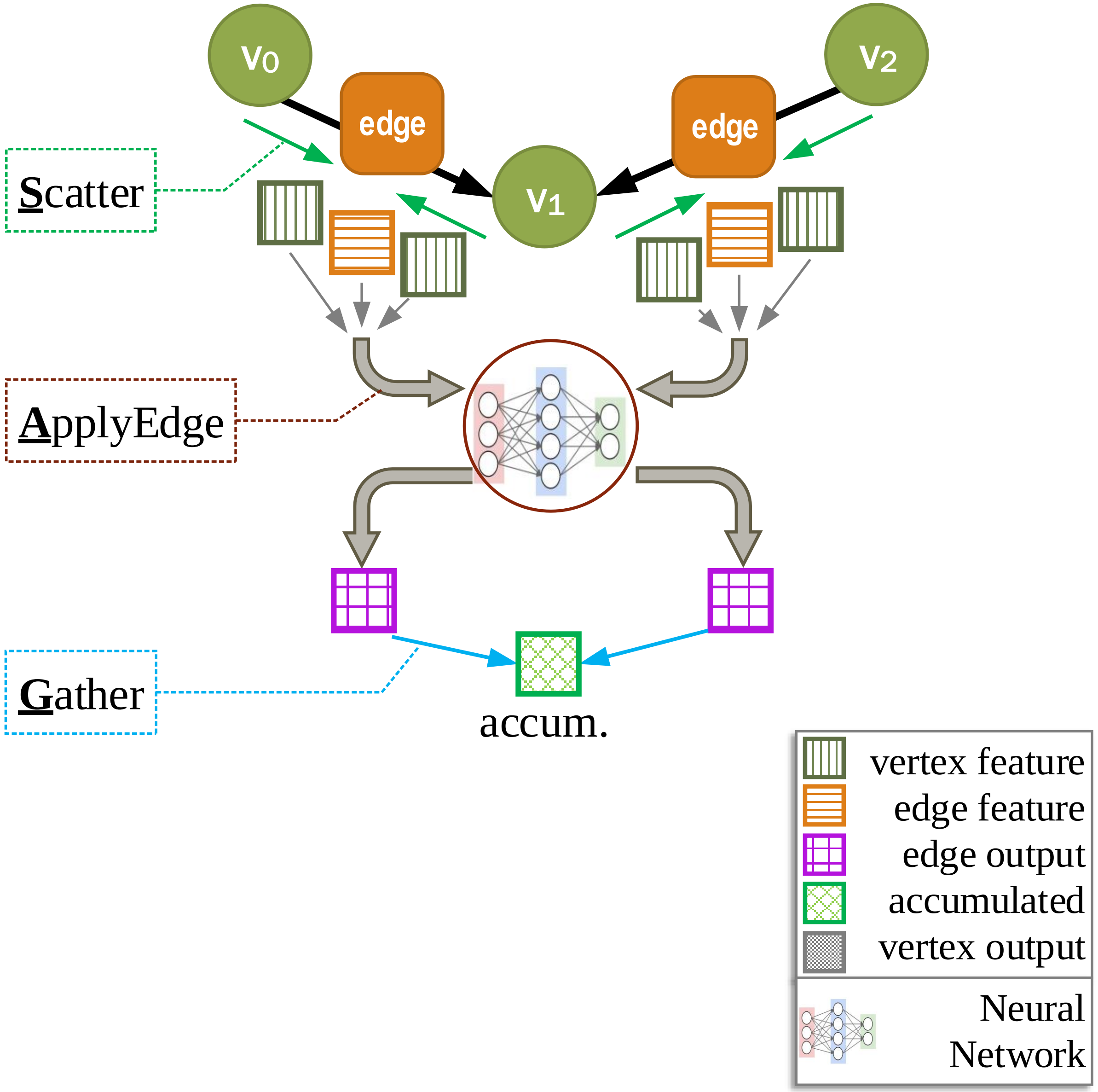




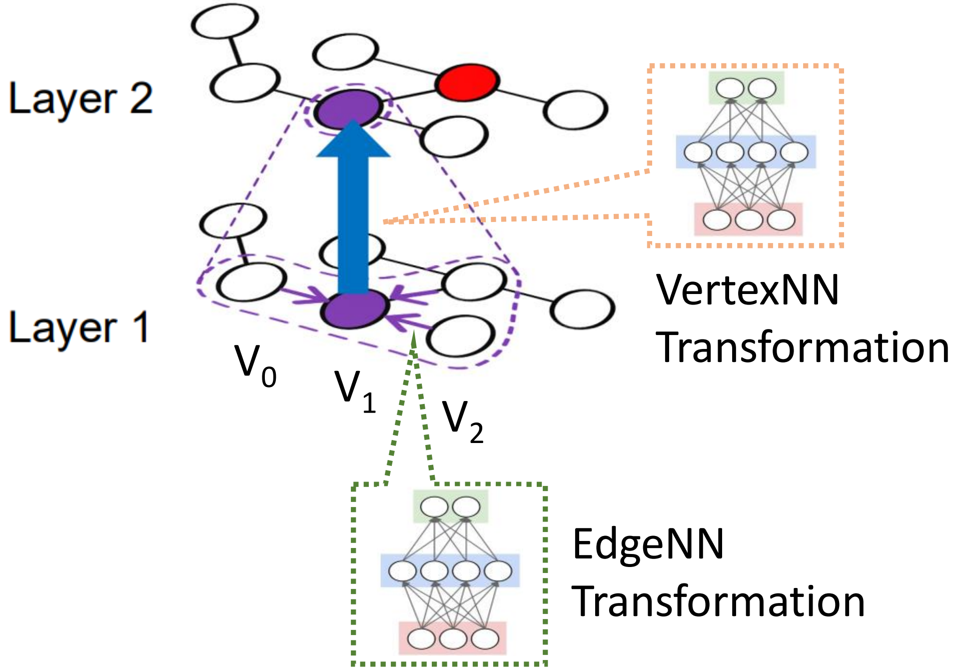
SAGA-NN Abstraction

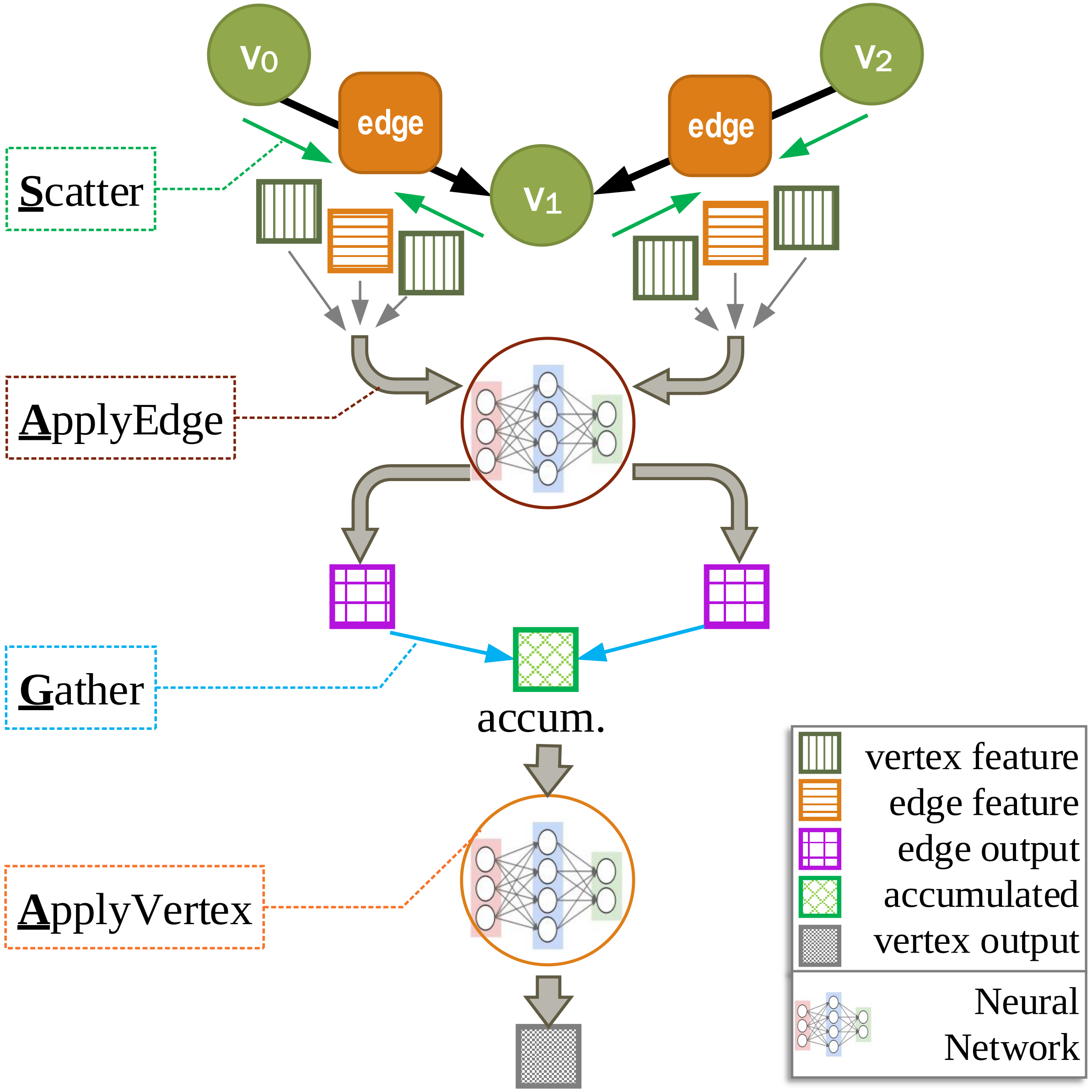


EdgeNN
Transformation

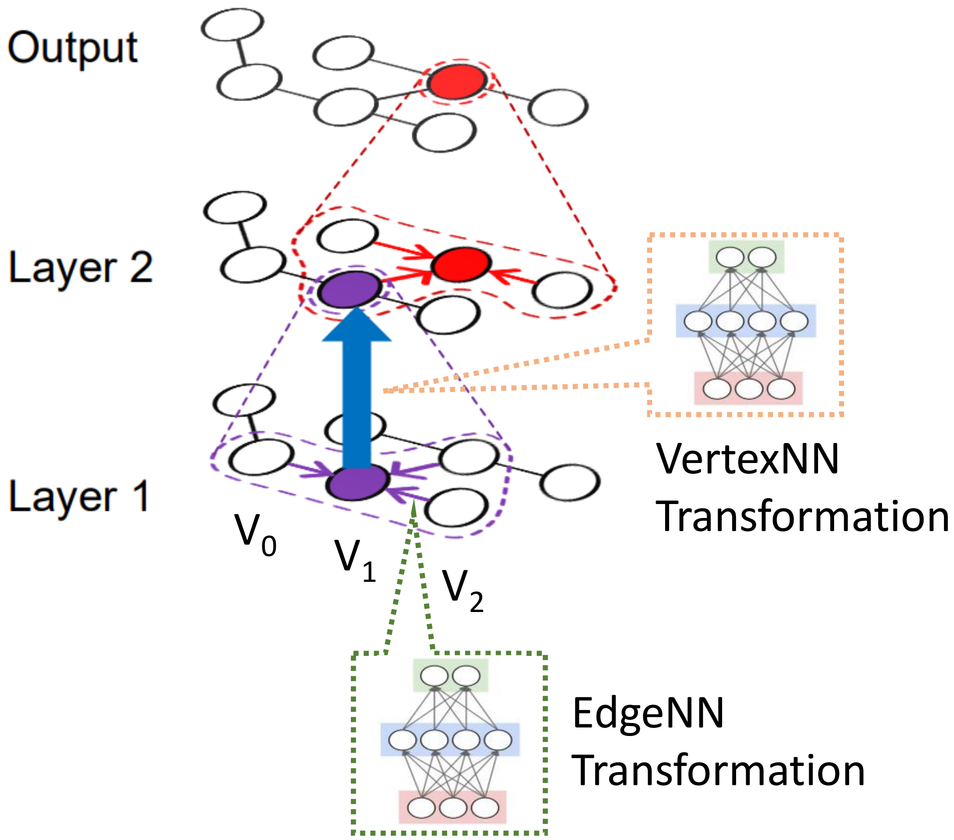


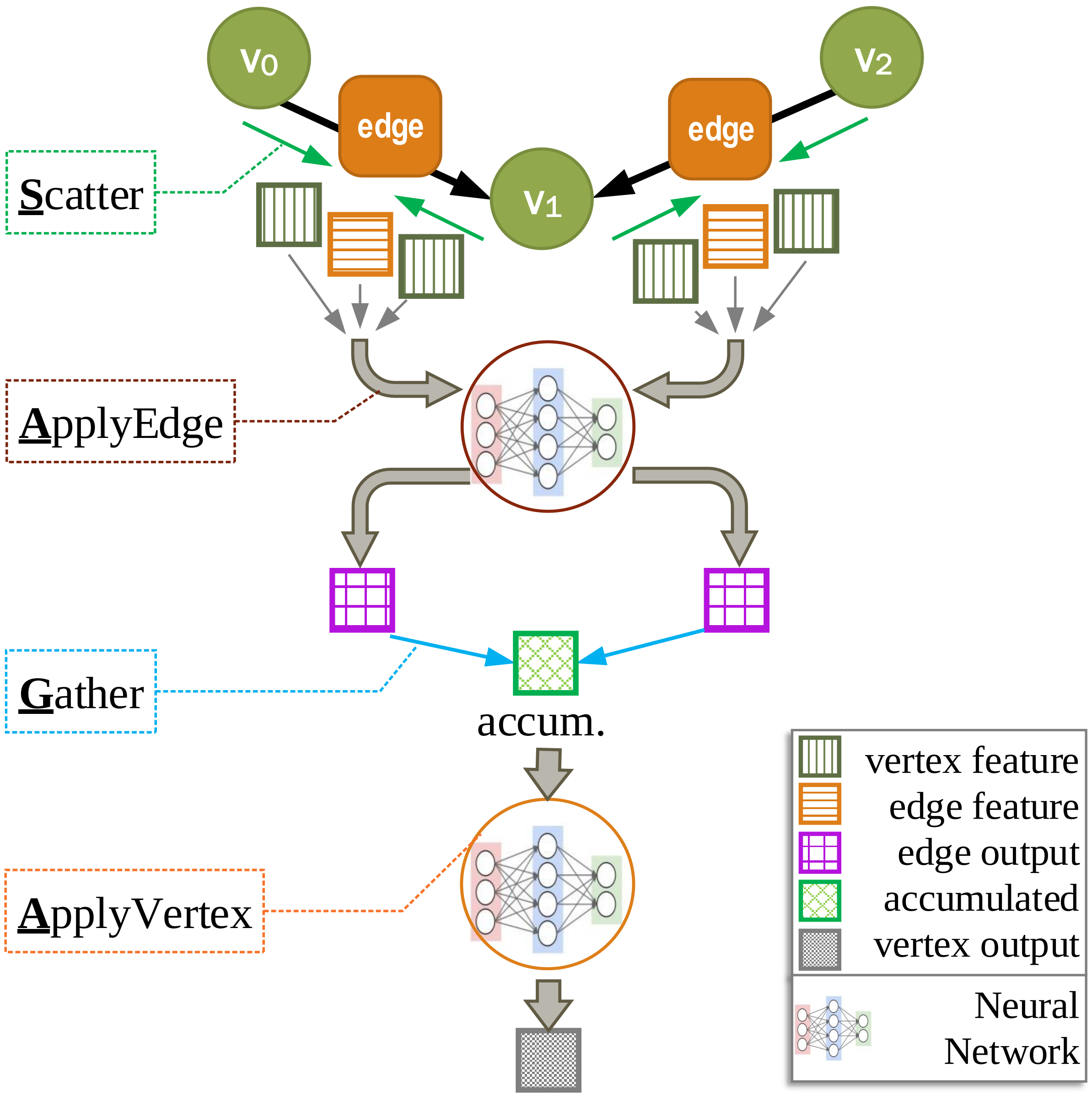
SAGA-NN Abstraction





SAGA-NN Abstraction





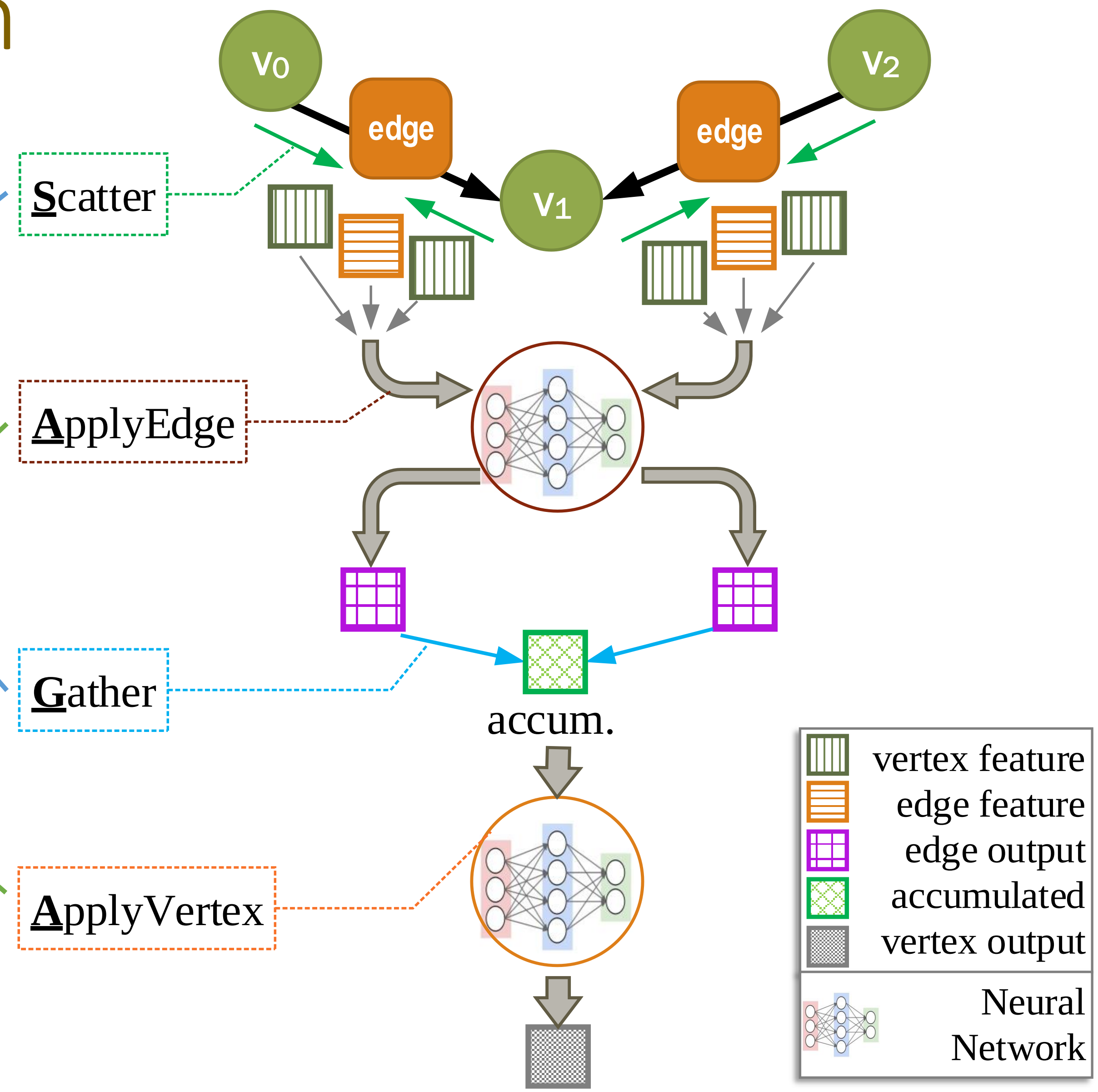
SAGA-NN Abstraction

Graph Operations

- transparent to users

Neural Network Operations

- define NN computation with dataflow



Example: Graph Convolutional Network (GCN)

Gather

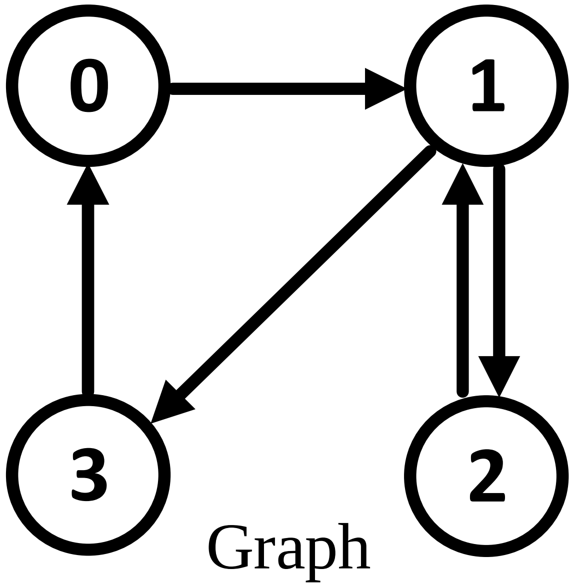
Scatter
ApplyEdge

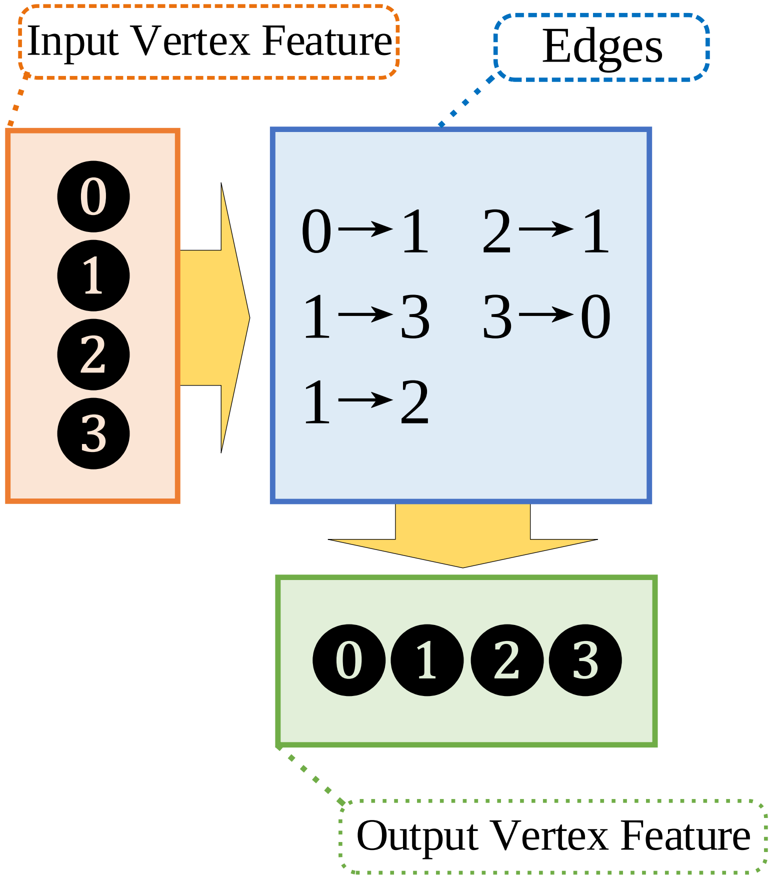
$$\mathbf{h}_u^{\ell+1} = \mathrm{ReLU}\left(W^{\ell} \otimes \left(\sum_{v \to u} e_{vu} \odot \mathbf{h}_v^{\ell} \right) + B^{\ell} \right)$$

ApplyVertex

```
class GCNLayer(GNNLayer):
    def _init(self):
        self.gather_accumulator = "sum"
        ...  # init variables
    def _apply_vertex(self, vertex, accum):
        ret = tf.matmul(accum, self.vars['weight'])
        ret = ret+self.vars['bias']
        return tf.nn.relu(ret)
    def _apply_edge(self, edge):
        return edge.src * edge.data
class GCNApp(GNNModel):
    def _init(self):
        ...  # init configs, optimizer, loss, et.al.
    def _build(self):
        self.layers.append(GCNLayer(...)) # params
        self.layers.append(GCNLayer(...))
        self.layers.append(GCNLayer(...))
        self.layers.append(GCNLayer(...))
```

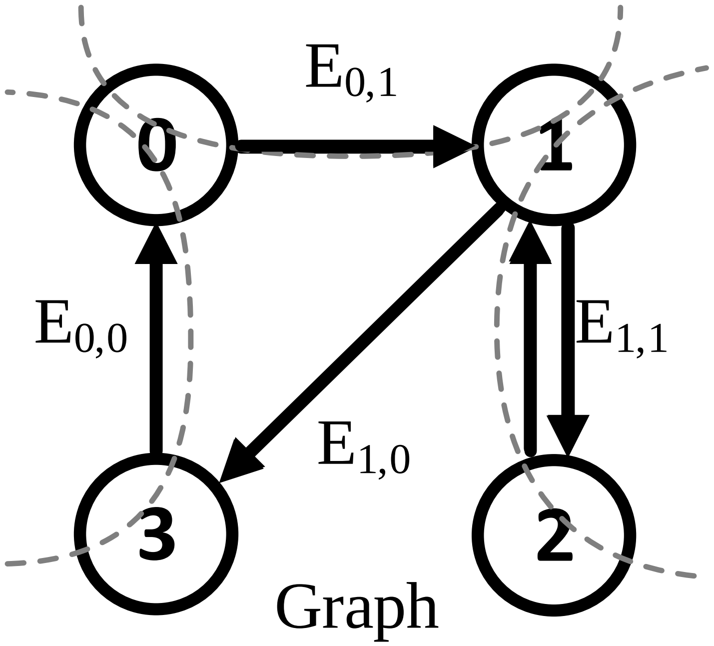
Scaling beyond GPU memory limit

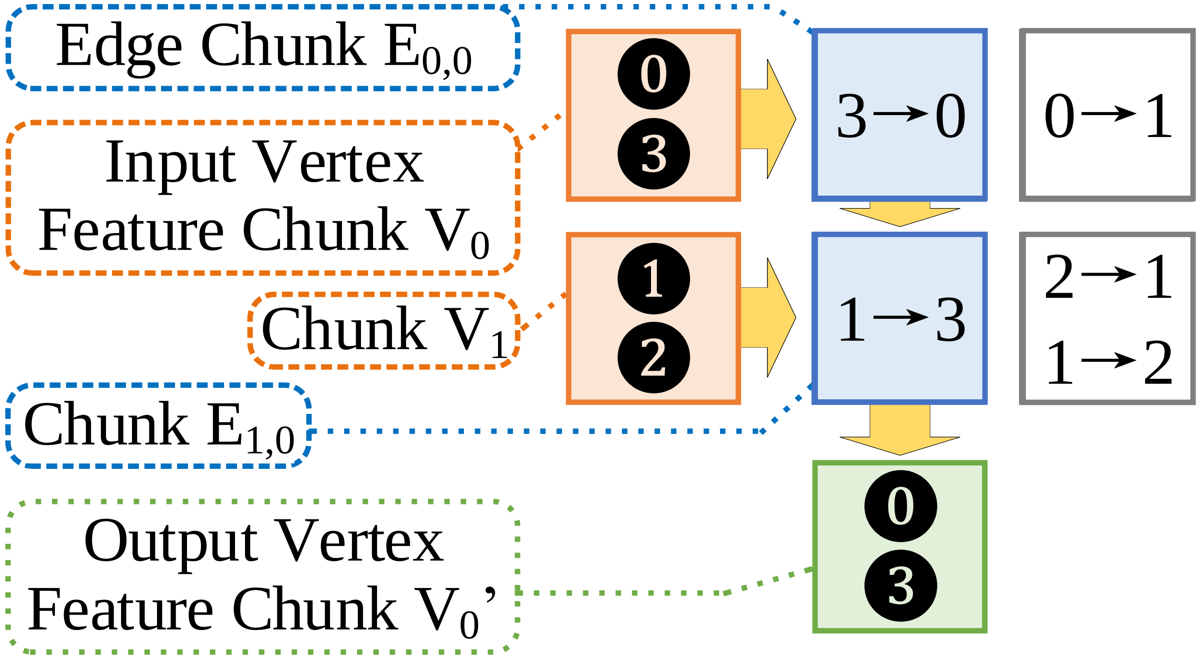


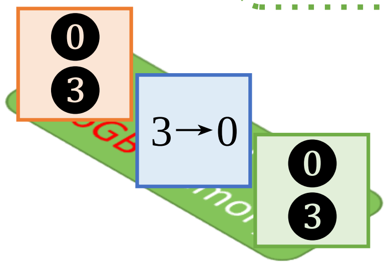


Scaling beyond GPU memory limit

- 2D graph partitioning



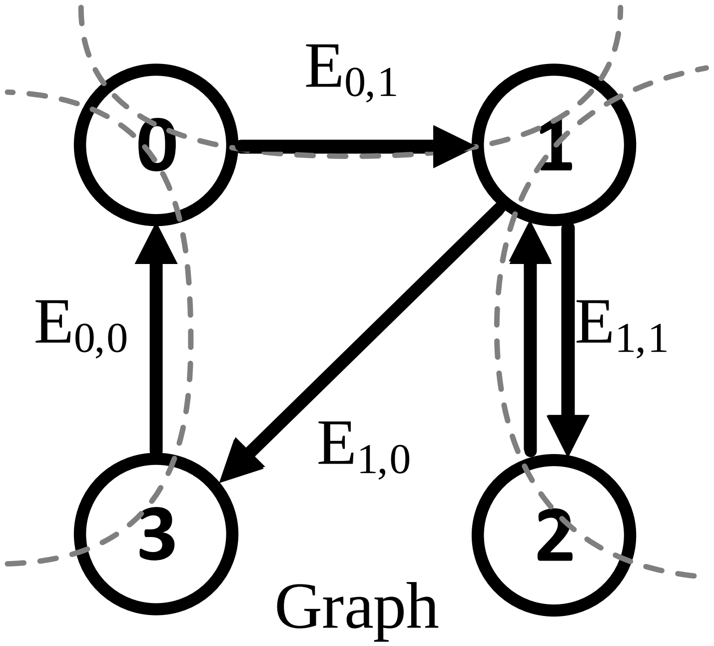


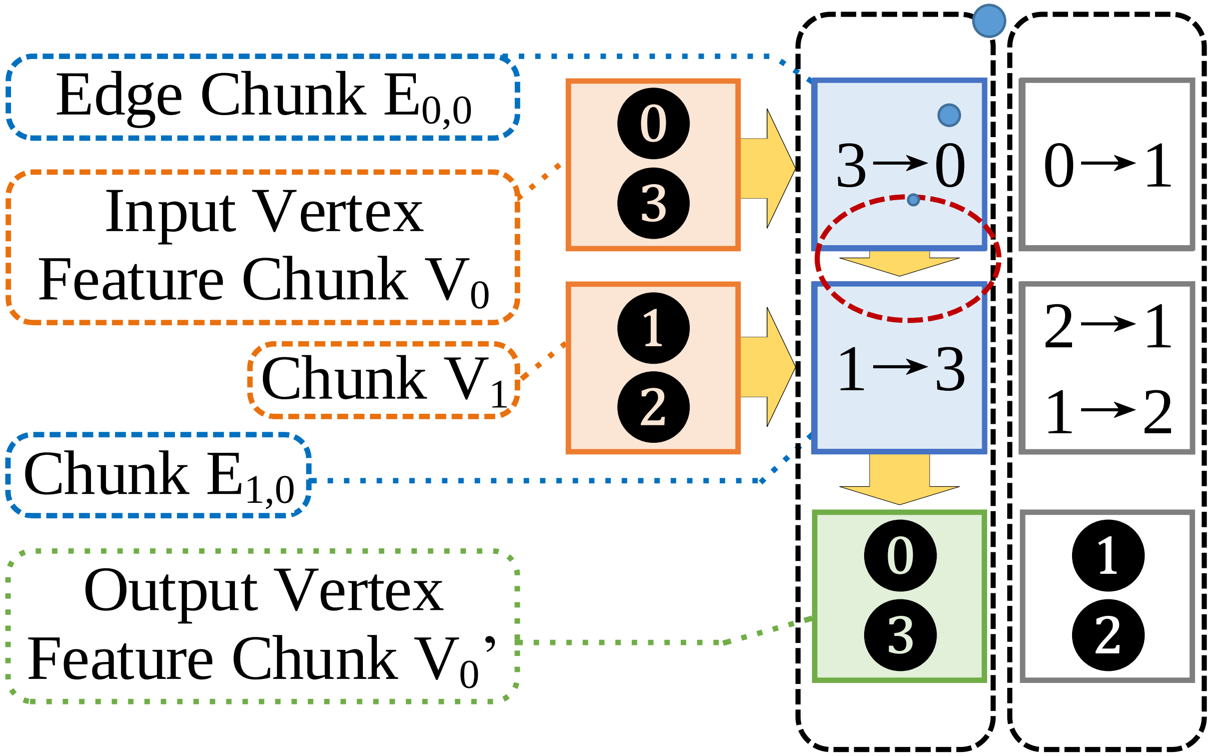


Scaling beyond GPU memory limit

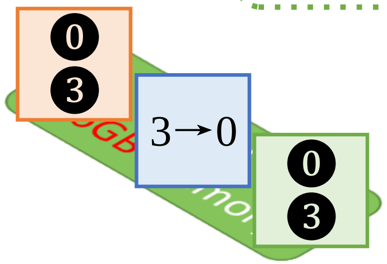
Accumulate in GPU memory

- 2D graph partitioning



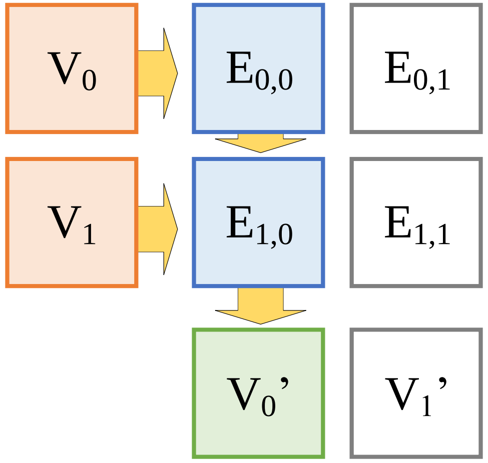


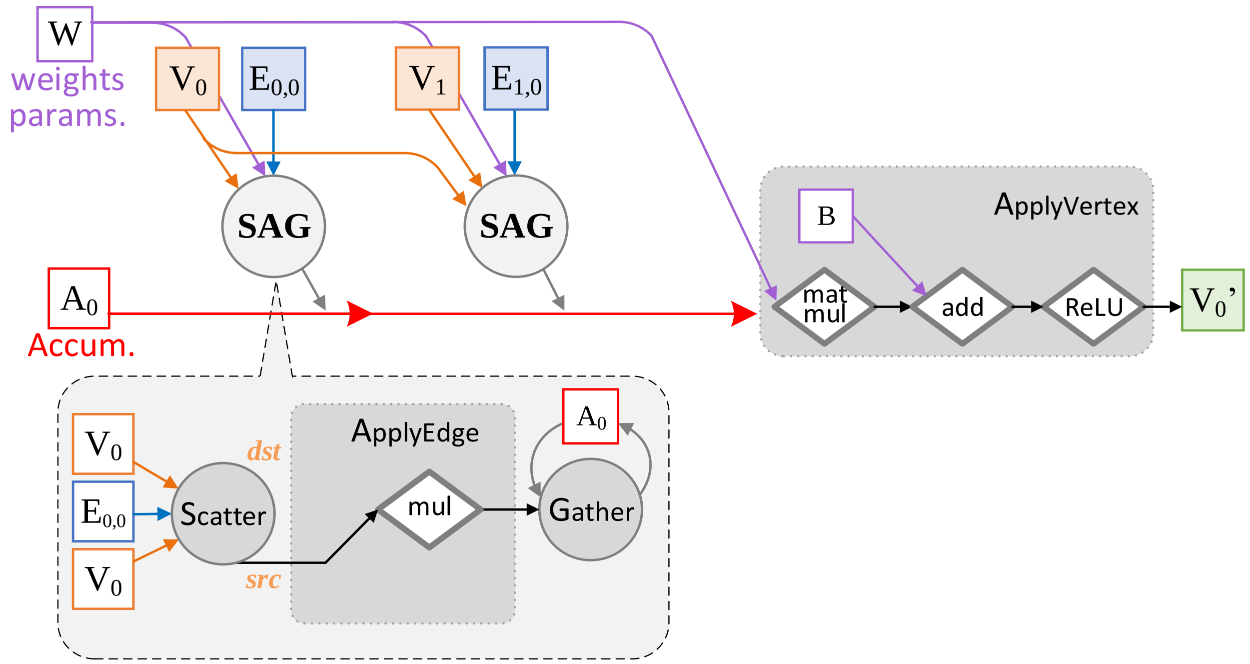
column-by-column chunk scheduling



Chunk-based Dataflow Translation: GCN

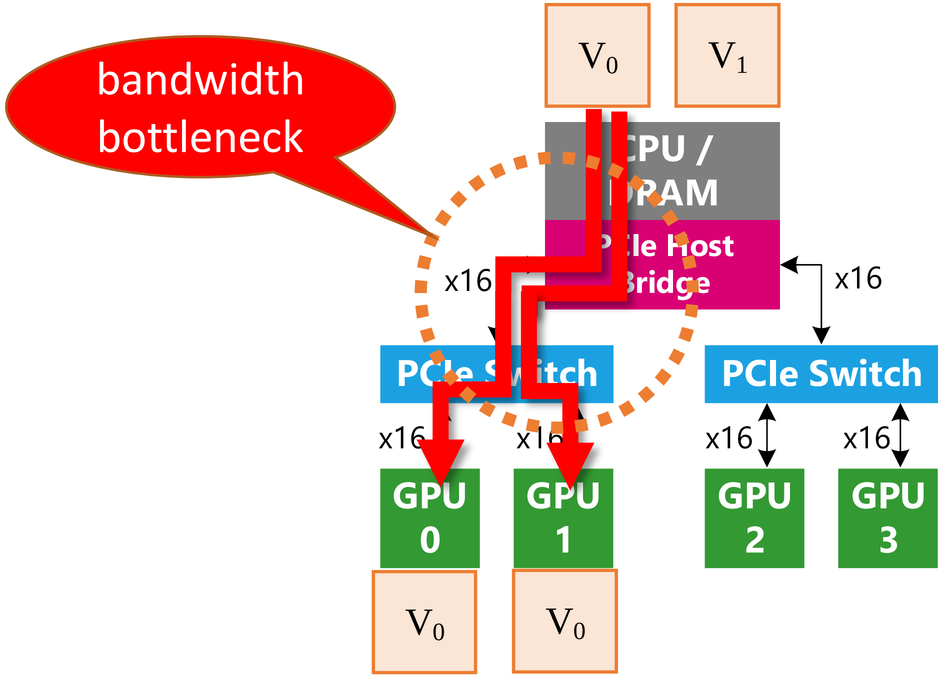
```python
class GCNLayer(GNNLayer):
    def _init(self):
        self.gather_accumulator = "sum"
        ...  # init variables
    def _apply_vertex(self, vertex, accum):
        ret = tf.matmul(accum, self.vars['weight'])
        ret = ret+self.vars['bias']
        return tf.nn.relu(ret)
    def _apply_edge(self, edge):
        return edge.src * edge.data
```

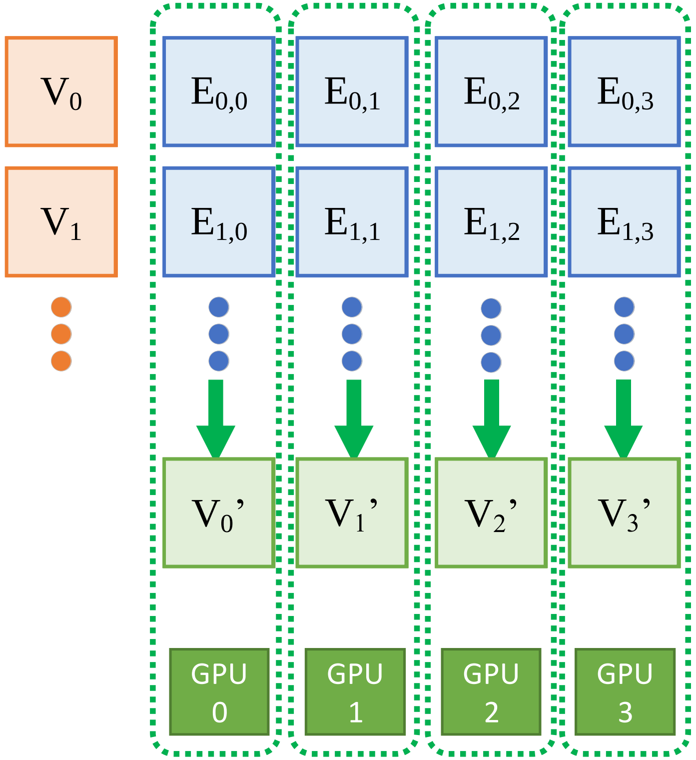




Scaling to multi-GPU

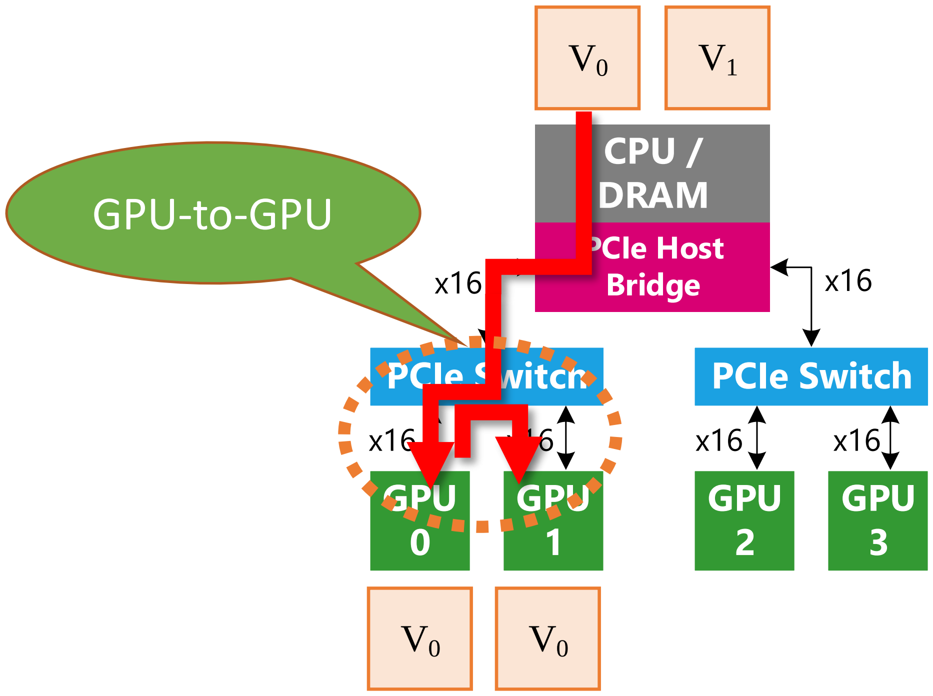
- Naïve solution:
  - Approach: each GPU takes a “column” of edge chunks
  - Problem: redundant data transfer through shared upper links

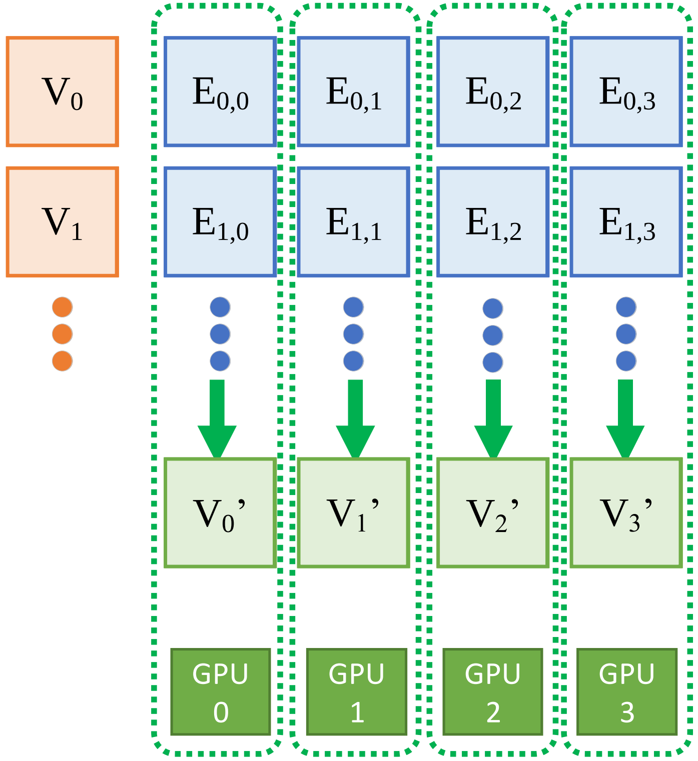




Scaling to multi-GPU

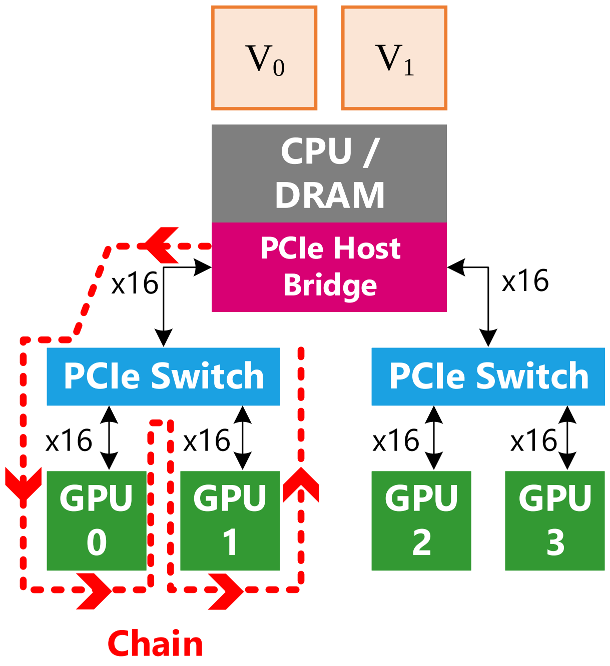
- Observation: link to neighbor GPU is empty
- Approach: directly load from neighbor GPU

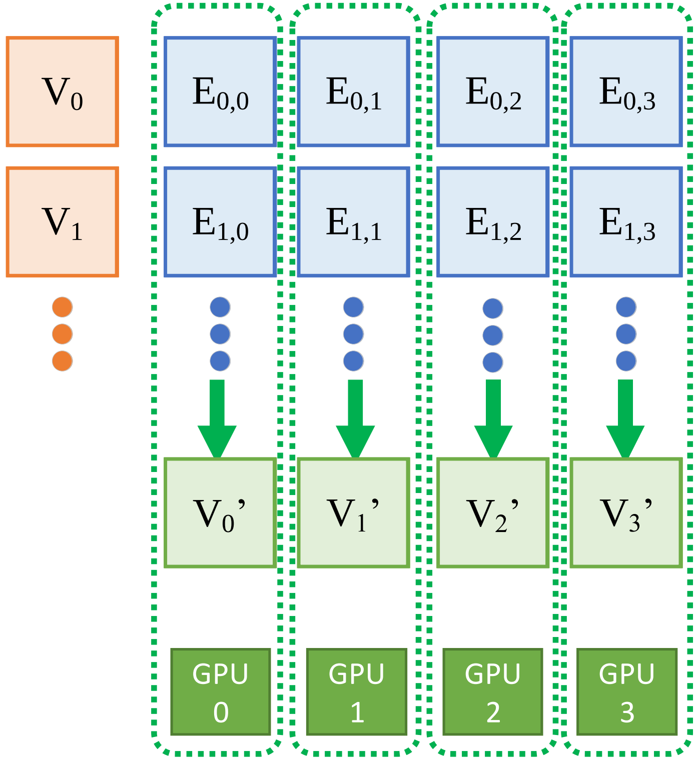




Scaling to multi-GPU

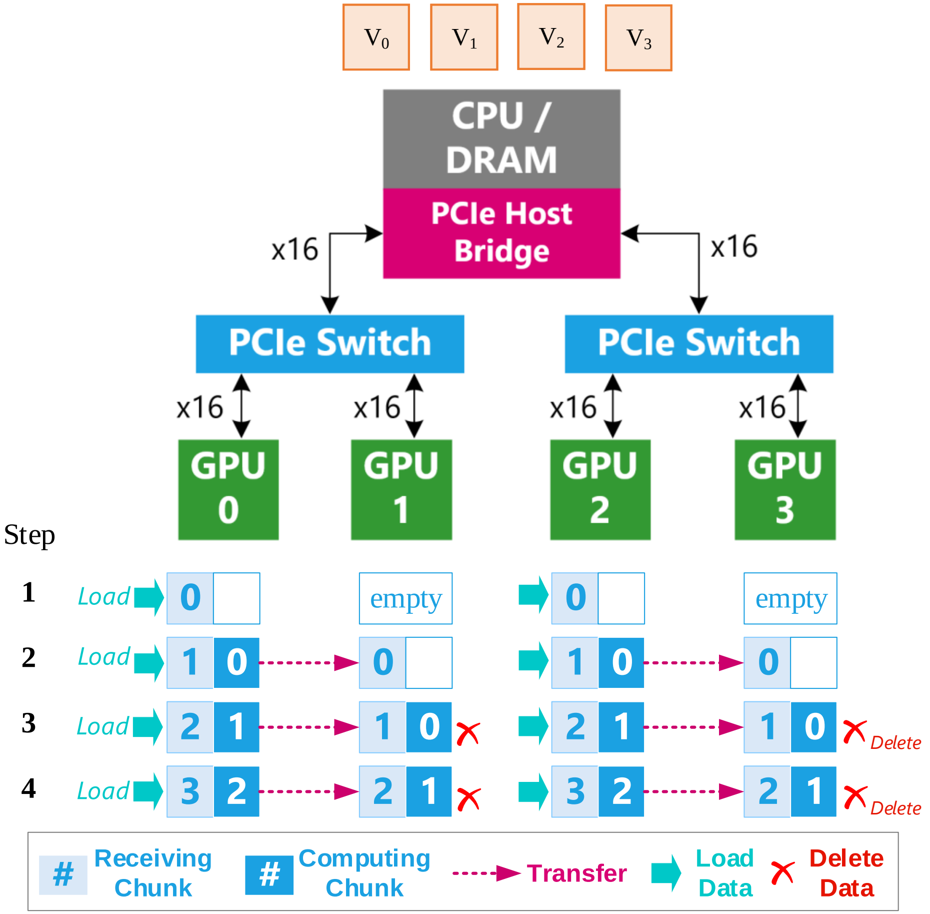
- Observation: link to neighbor GPU is empty
- Approach: directly load from neighbor GPU

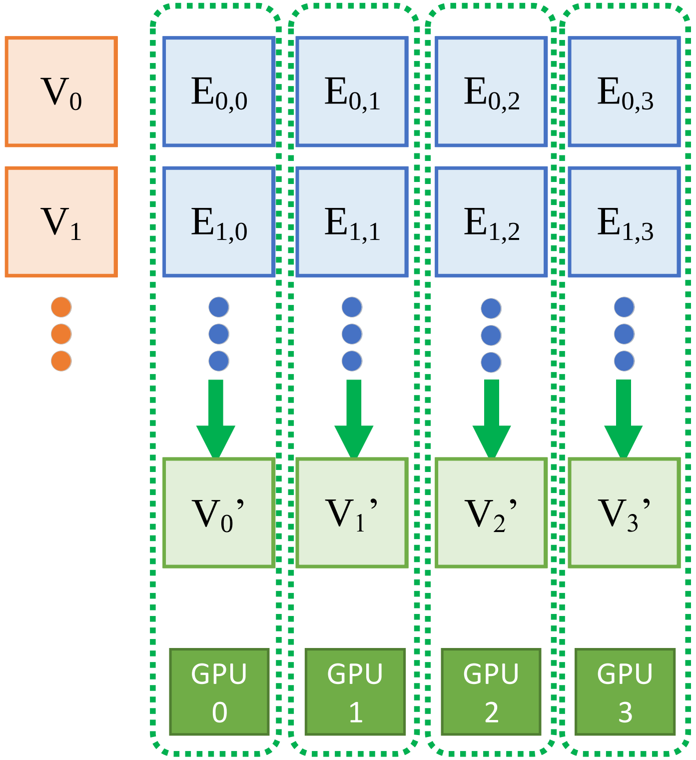




Chain-based Parallel Streaming

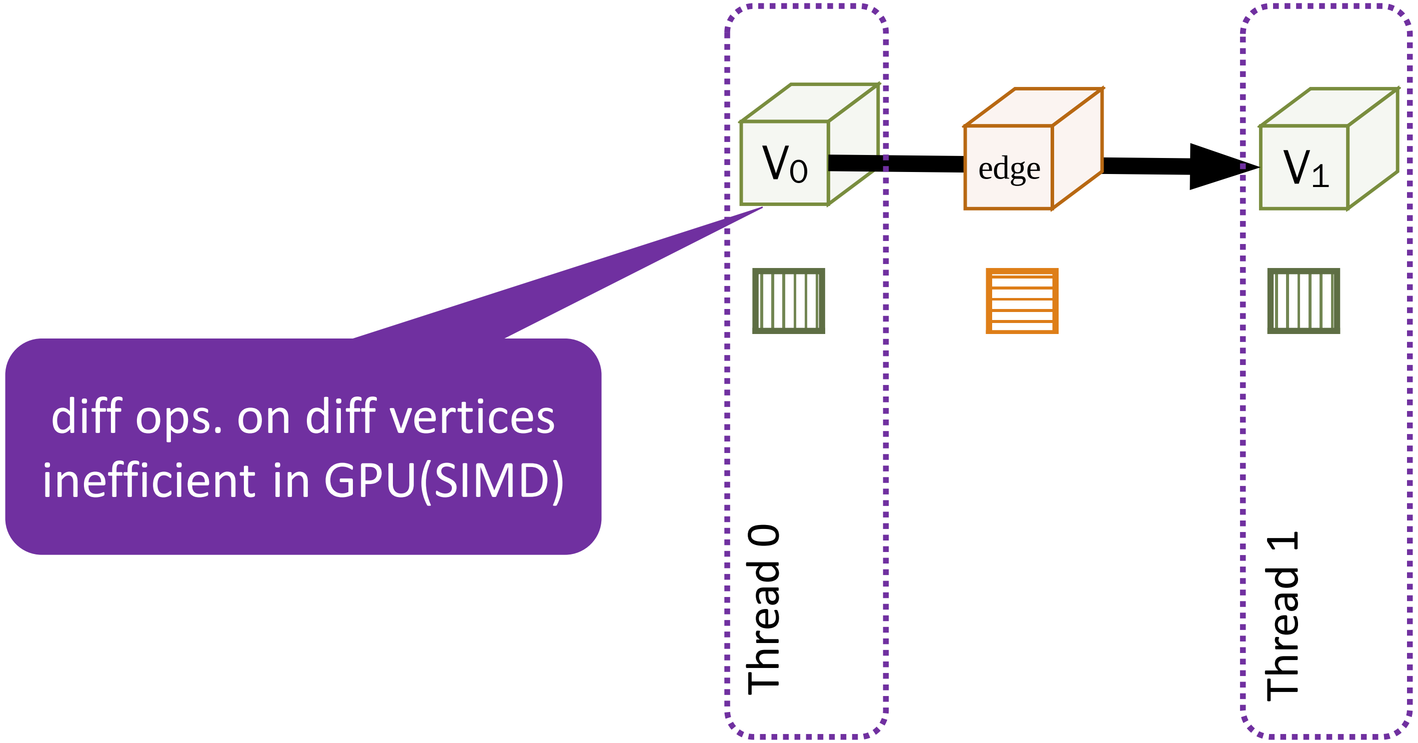
- Each GPU can consume all the chunks in chain for a PCIe switch





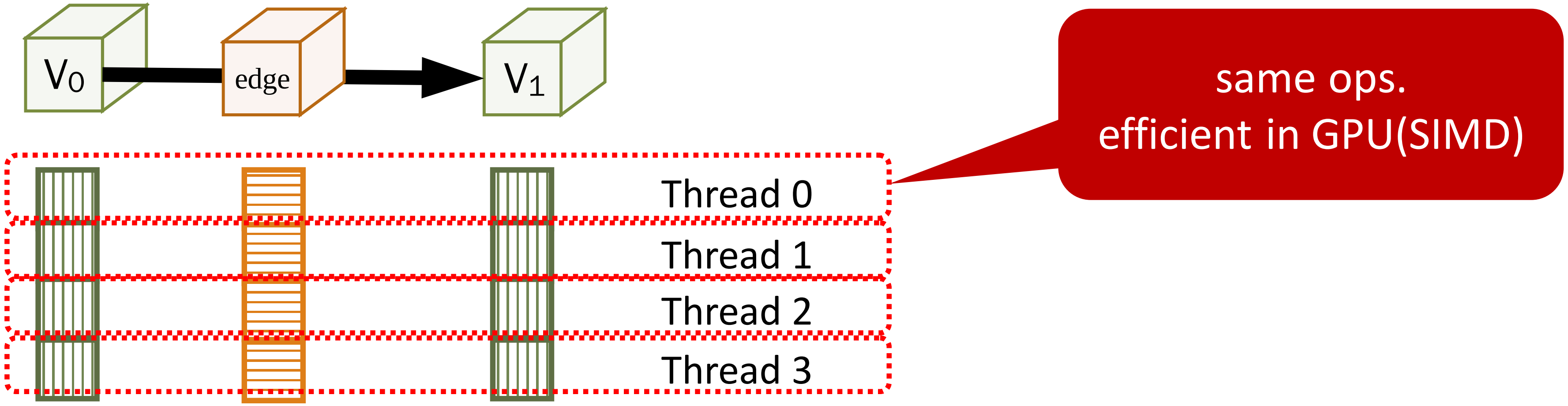
High performance graph propagation kernels

- Traditional multithread parallelization
  - PageRank, SSSP, CC, etc: one-dimension scalar value on vertices
  - A thread takes a vertex/edge



High performance graph propagation kernels

- Observation
  - High-dimension feature
  - More-regular operations along feature-dimension
- Solution: exploit feature-dimension parallelization in GNNs



Experiment Setup

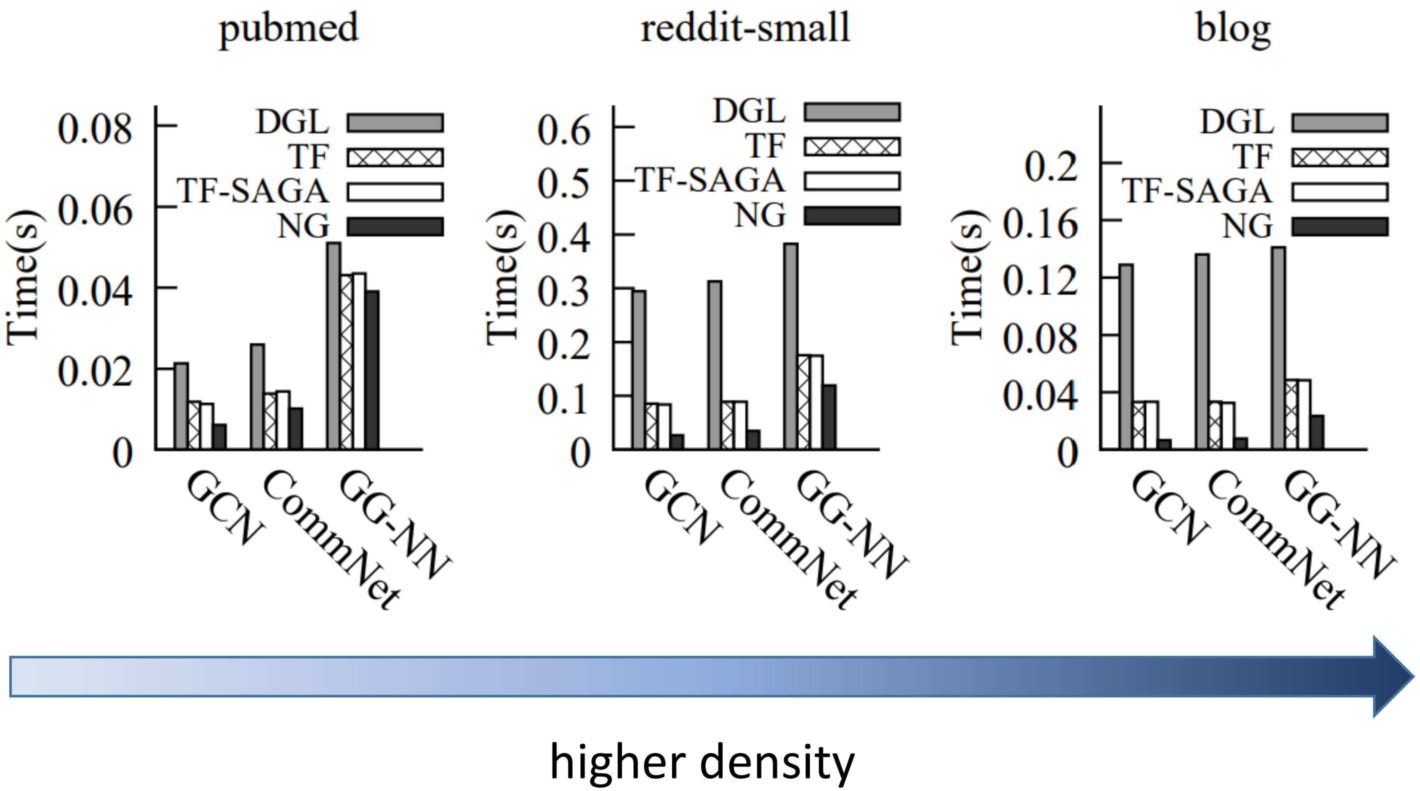
- Typical GNN Applications
  - Communication neural network (**CommNet**) // no computation in ApplyEdge
  - Graph convolutional networks (**GCN**) // element-wise Mul in ApplyEdge
  - Gated graph neural networks (**GG-NN**) // GRU in ApplyVertex
- Testbed
  - 2x E5-2690-v4 (14-core with HT)
  - 512GB Quad-Channel RAM
  - 8x NVIDIA Tesla P100 GPU
  - Ubuntu 16.04
  - CUDA 9.0 + cuDNN 7

dataset	vertex	edge	feature	label	degree	type
blog	10.3K	668.0K	128	39	65	social
pubmed	19.7K	108.4K	500	3	5	cite
reddit_small	58.2K	1.4M	300	41	25	social
reddit_full	2.4M	705.9M	300	50	292	social
enwiki	3.6M	276.1M	300	12	77	knowledge
amazon	8.6M	231.6M	96	22	27	review

- Comparison
  - TF: TensorFlow v1.7 (CPU/GPU)
  - DGL: Deep Graph Library v0.1.3 with PyTorch v1.0 backend
  - TF-SAGA: SAGA-NN with TensorFlow v1.7 backend
  - NG: NeuGraph on top of TensorFlow v1.7

Performance on a Single GPU

Compare with TF, DGL on small graphs



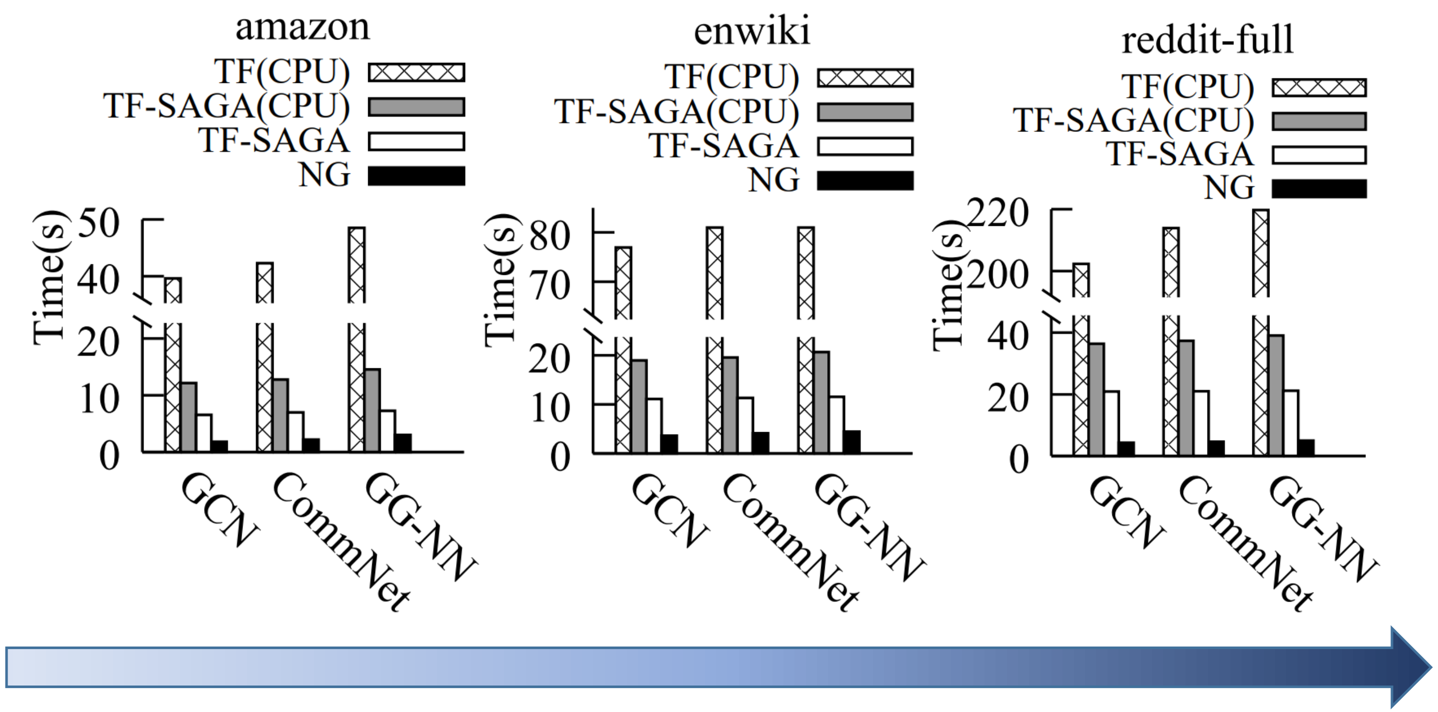
up to 5x speedup vs TF

up to 19x speedup vs DGL

Avg. epoch time; #layer=2, hidden_size=512

Scaling-up on a Single GPU

- Compare with TF(CPU), TF-SAGA(CPU), TF-SAGA on large graphs
  - TF and DGL run out-of-memory (OOM)



higher density

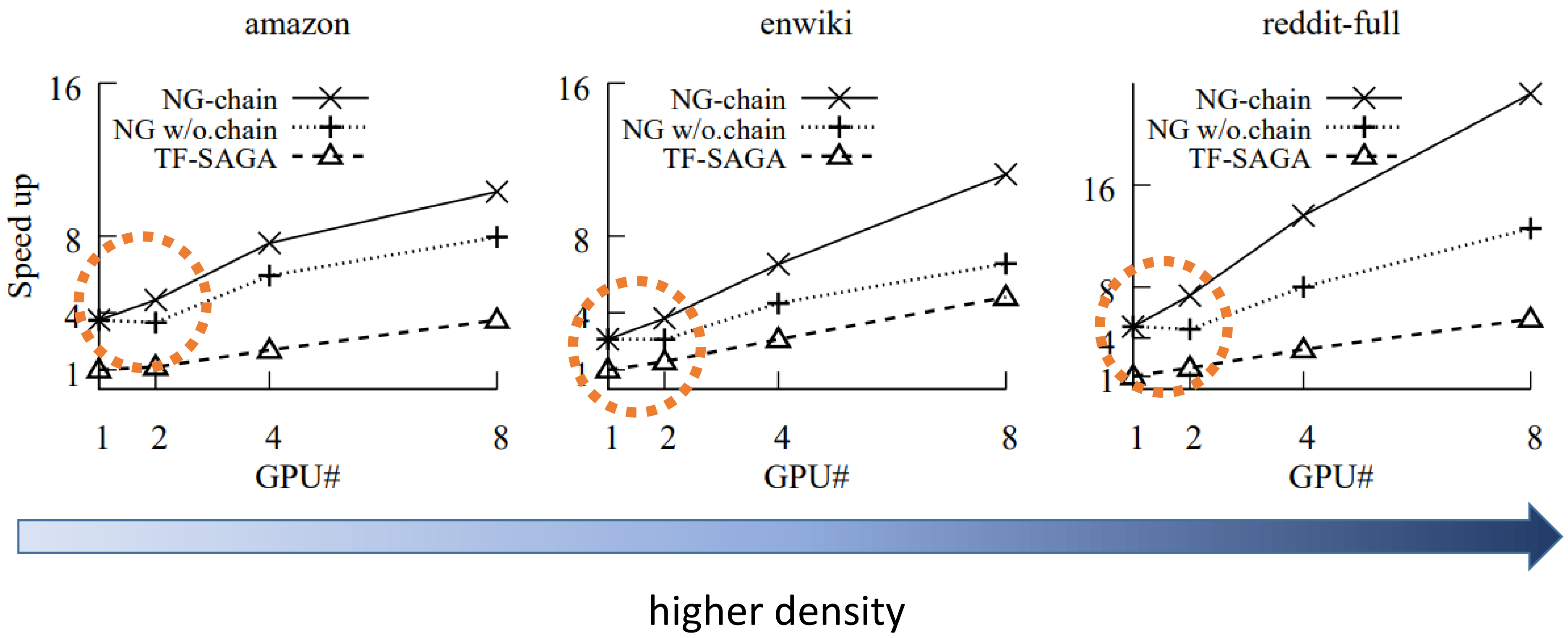
up to 47x speedup vs TF(CPU)

up to 5x speedup vs TF-SAGA

Avg. epoch time; #layer=2, hidden_size=16 (amazon), 32 (enwiki), 64 (reddit-full)

Scaling-out on Multiple GPUs

- GCN on three large graphs on different number of GPUs



Avg. speedup over TF-SAGA-1GPU; #layer=2, hidden_size=16 (amazon), 32 (enwiki), 64 (reddit-full)

Conclusion

NeuGraph: advocate unifying deep learning and graph computing for graph neural networks

- Define a new, flexible SAGA-NN model to express GNNs
- Fuse graph-related optimizations into deep learning frameworks
- Demonstrate NeuGraph achieves efficient and scalable GNN processing

Thank you!
Q&A

For more information, please visit: https://aka.ms/neugraph

北京大學 PEKING UNIVERSITY

Microsoft Research