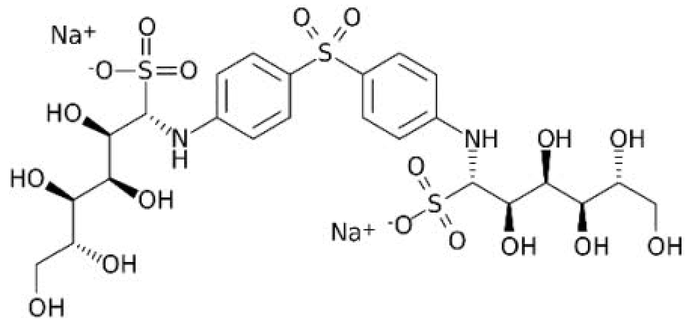
亚马逊云科技

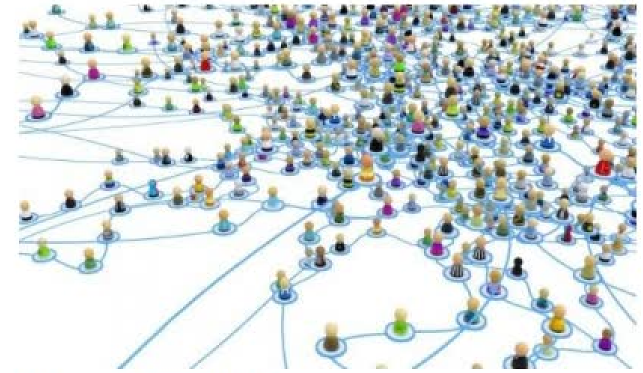
开源图深度学习框架的机遇与挑战

王敏捷　资深应用科学家
亚马逊云科技上海人工智能研究院

图数据无处不在

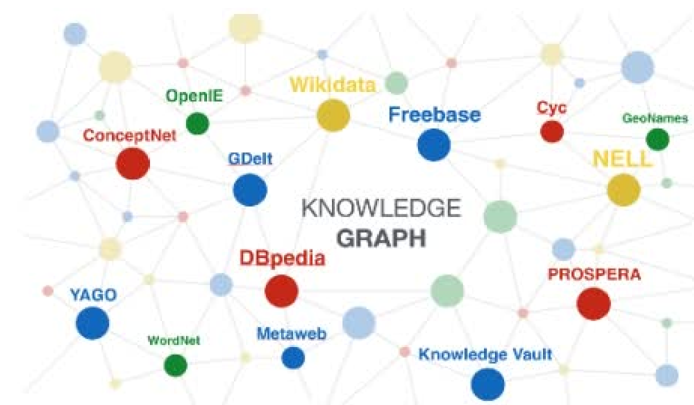


药物和分子结构



社交网络

用户产品交互网络



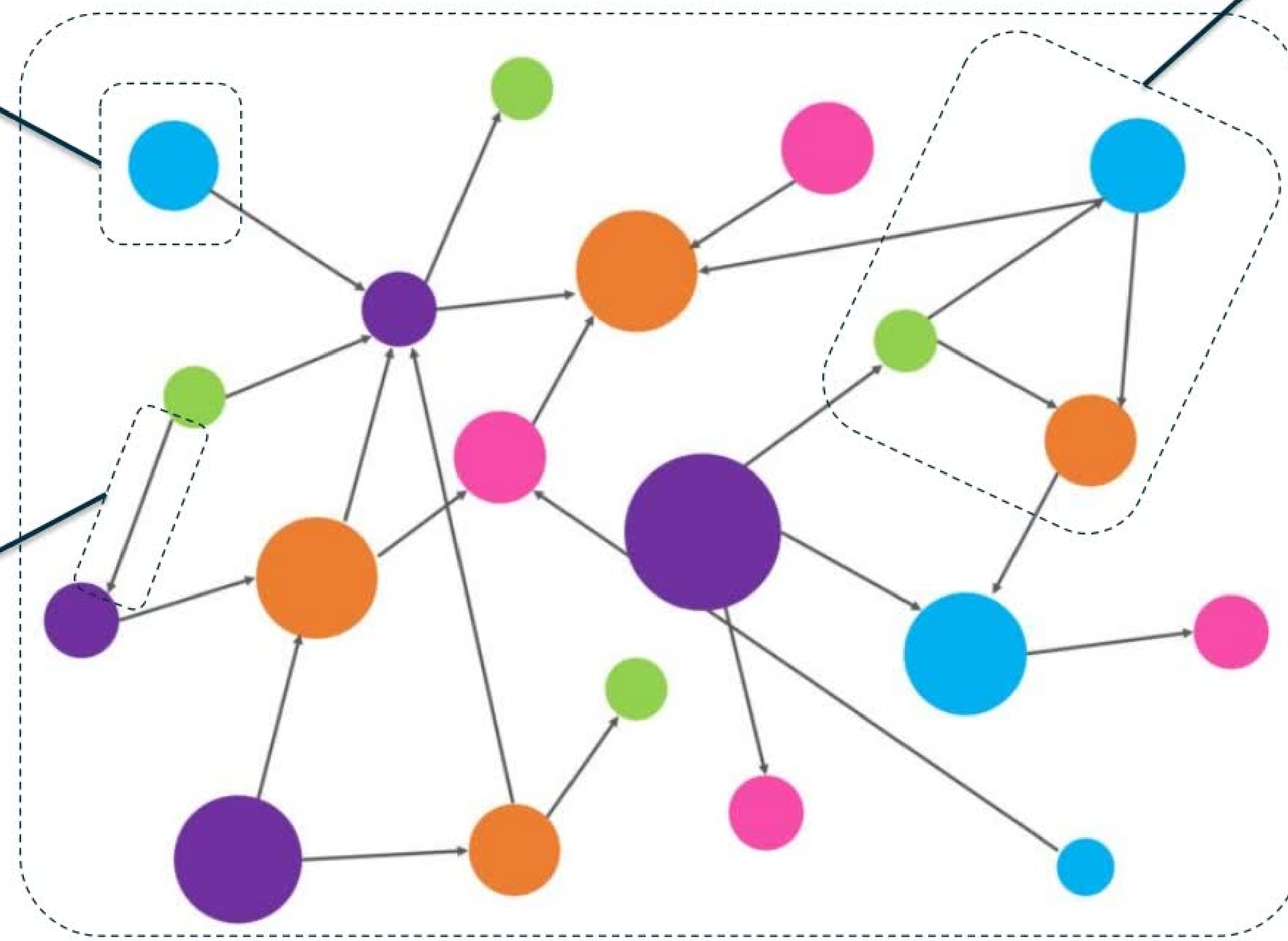
知识图谱

亚马逊云科技

图机器学习任务

节点预测
例：分析一个用户的兴趣爱好

社区（子图）预测
例：检测是否存在可疑的金融欺诈行为



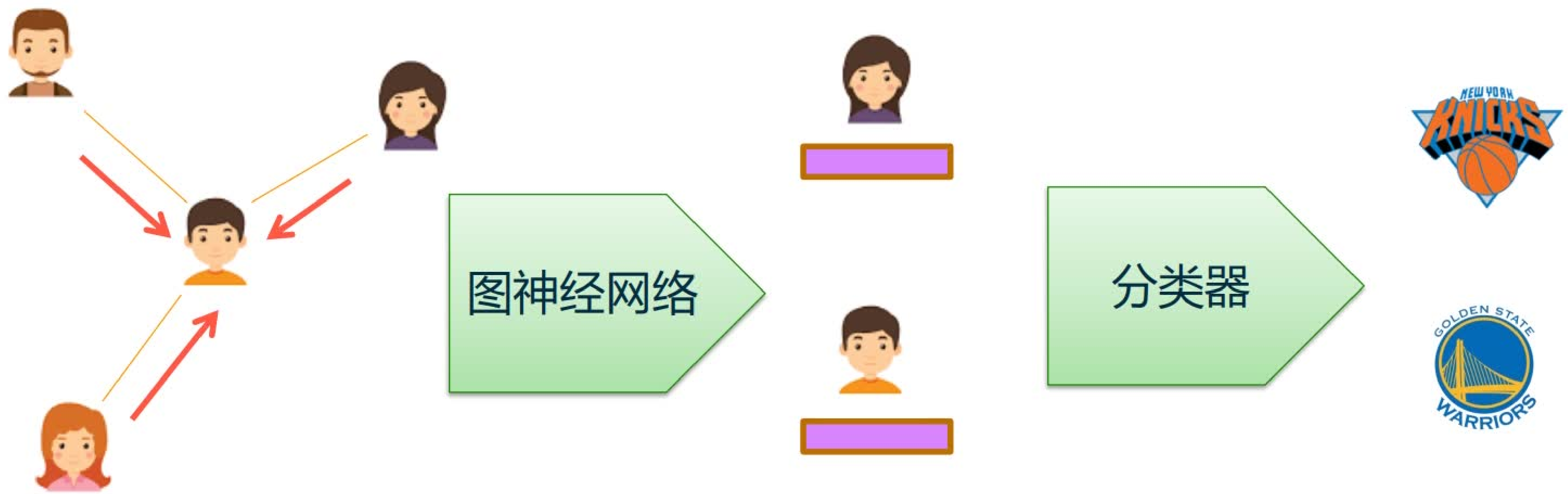
链接预测
例：判断一个交互行为的类型
例：知识图谱补全，推荐系统

图性质预测
例：预测分子或化合物性质
例：图生成模型

亚马逊云科技

图 + 深度学习=> 图神经网络 (GNN)

用于学习点、边或者整张图的向量表示的一类深度神经网络



图神经网络基于消息传递

消息函数

$$\text{Edge-wise: } \mathbf{m}_e^{(t+1)} = \boxed{\phi}\left(\mathbf{x}_v^{(t)}, \mathbf{x}_u^{(t)}, \mathbf{w}_e^{(t)}\right), (u, e, v) \in \mathcal{E}.$$

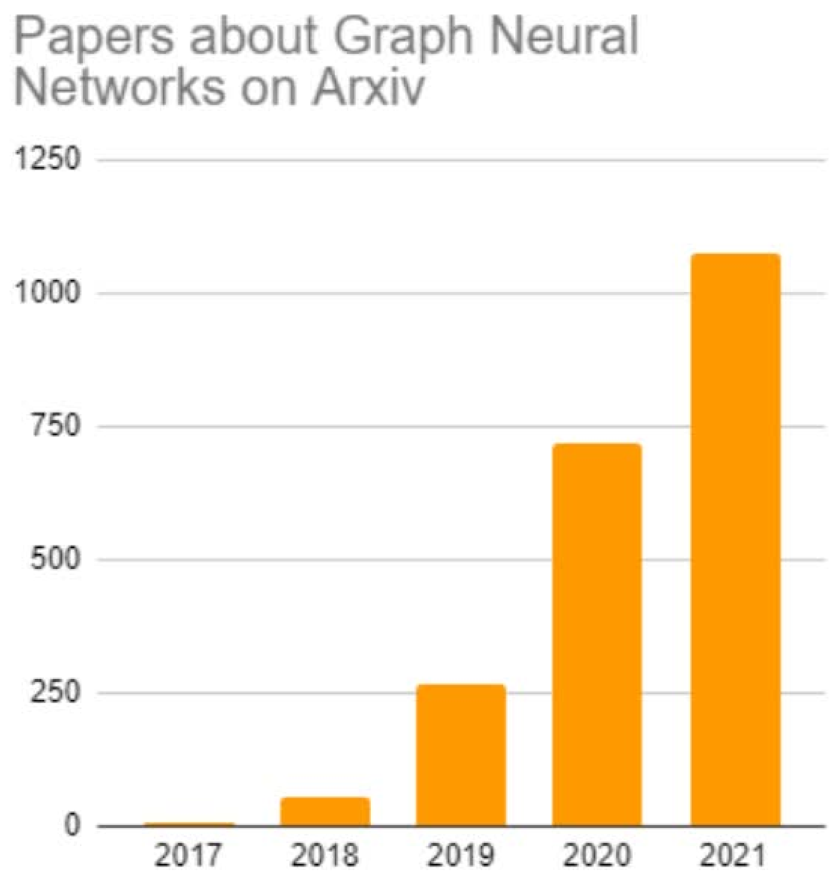
$$\text{Node-wise: } \mathbf{x}_v^{(t+1)} = \boxed{\psi}\left(\mathbf{x}_v^{(t)}, \boxed{\rho}\left(\left\{\mathbf{m}_e^{(t+1)} : (u, e, v) \in \mathcal{E}\right\}\right)\right)$$

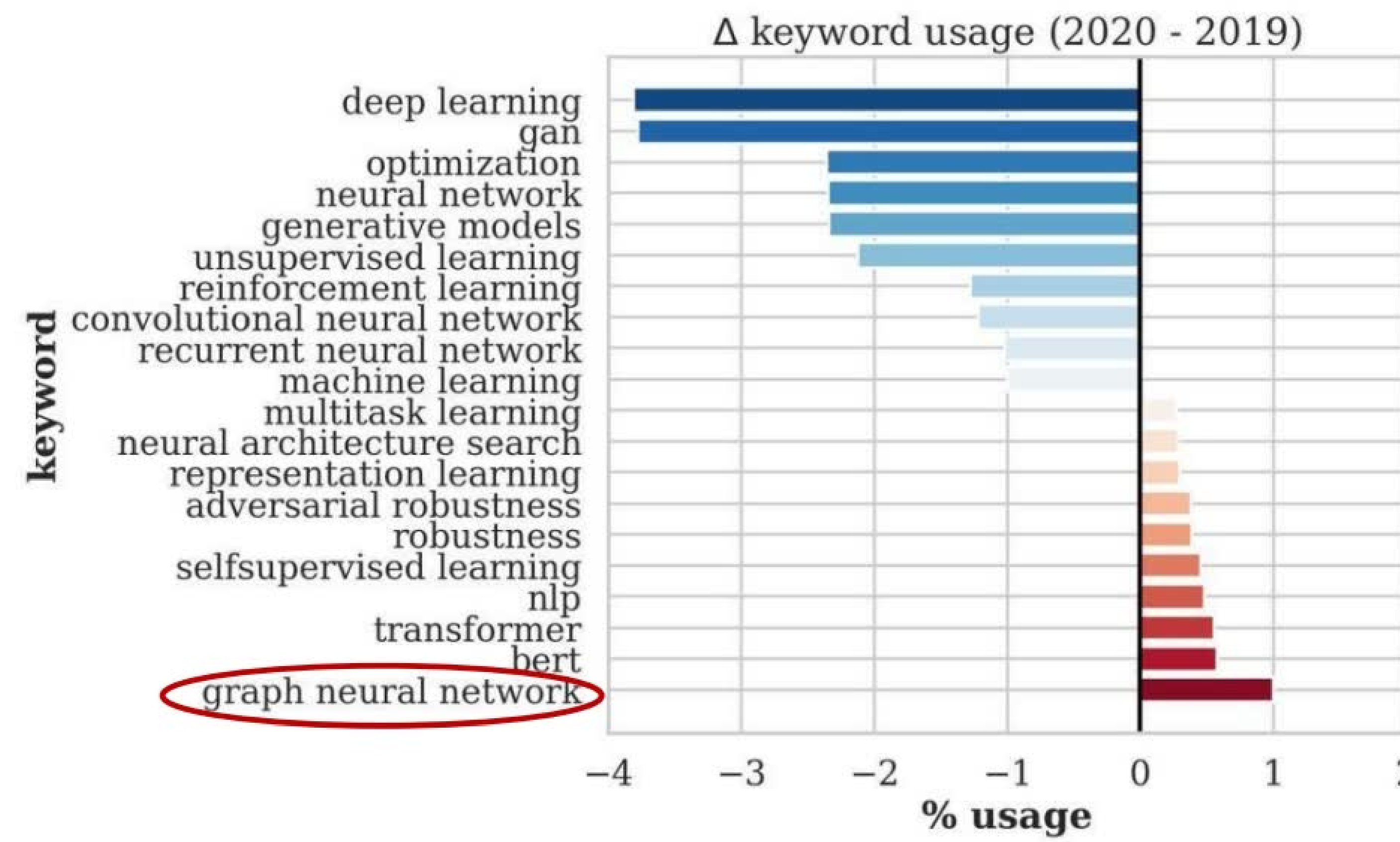
更新函数

累和函数

亚马逊云科技

图神经网络有多火





亚马逊云科技

Graph Neural Networks are the next BIG thing!

re:MARS 2022 keynote by Swami Sivasubramanian

Deep Graph Library (DGL)

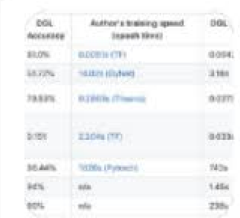
- 面向图结构数据的专用深度学习框架。
- 2018年12月在Neurips大会上宣布开源。
- 开发团队最初主要来自NYU和NYU Shanghai，由张峥教授发起。目前主要开发团队为张峥教授带领的亚马逊云科技上海人工智能研究院。

- 项目上线初就获得广泛关注和好评。
- Github Stars: **9.8K**, Forks: **2.3K**, 贡献者： **206**
- **DGL论文引用数 600+**
- **在学界，DGL是全球领先的图深度学习框架之一；在业界，DGL在使用率上更是全面领先。**

DEEP GRAPH LIBRARY

Yann LeCun @ylecun Follow

Brought to you by NYU, NYU-Shanghai, and Amazon AWS.



PyTorch @PyTorch
DGL (Deep Graph Library) - Clean and efficient library to build graph neural networks including GCN, TreeLSTM and graph generative models. Includes auto-batching and other tricks for speed.

Show this thread

Xavier Bresson @xbresson · Oct 25
I taught my students Deep Graph Library (DGL) in my lecture on "Graph Neural Networks" today. It is a great resource to develop GNNs with @PyTorch. Kudos to the team @GraphDeep!

Thomas Kipf @thomaskipf Follow

By far the cleanest and most elegant library for graph neural networks in PyTorch. Highly recommended! Unifies Capsule Nets (GNNs on bipartite graphs) and Transformers (GCNs with attention on fully-connected graphs) in a single API.

亚马逊云科技

灵活易用的编程接口
以“图”为本，贴近图计算的原生语义。

高效的底层系统设计
使用算子融合等技术对消息传递进行加速

DGL DEEP GRAPH LIBRARY

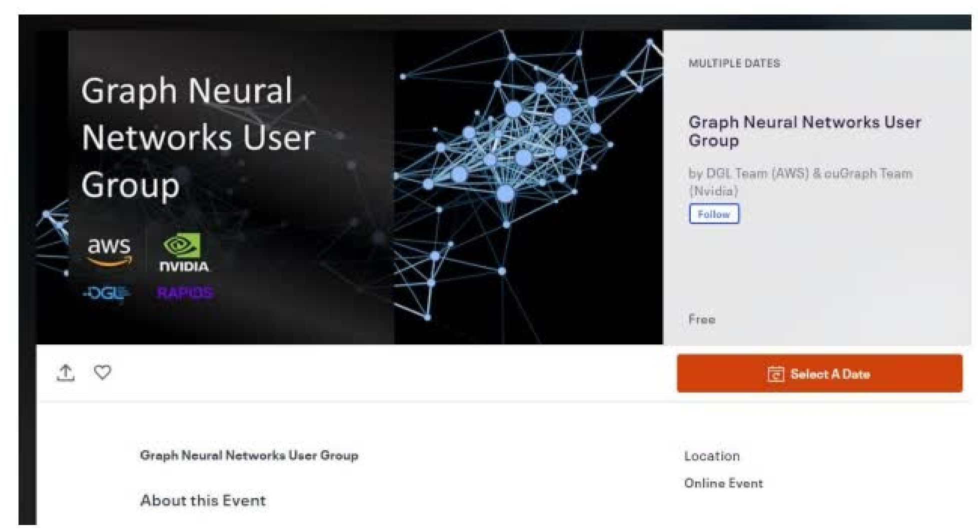
优秀的巨图训练性能
支持十亿级巨图，高效利用多机多GPU集群

完善的开源生态
与许多开源软件有良好互通性，基于DGL的生态也初显成果

DGL开源社区建设

- 广泛的开源合作伙伴
- 每月定期组织用户群分享会。
- 邀请学界和业界的研究者分享图神经网络的最新成果。
- 在学术顶会上举办DGL手把手教程 (GTC’19, KDD’19, WWW’20, KDD’20, GTC’20, WSDM’21)

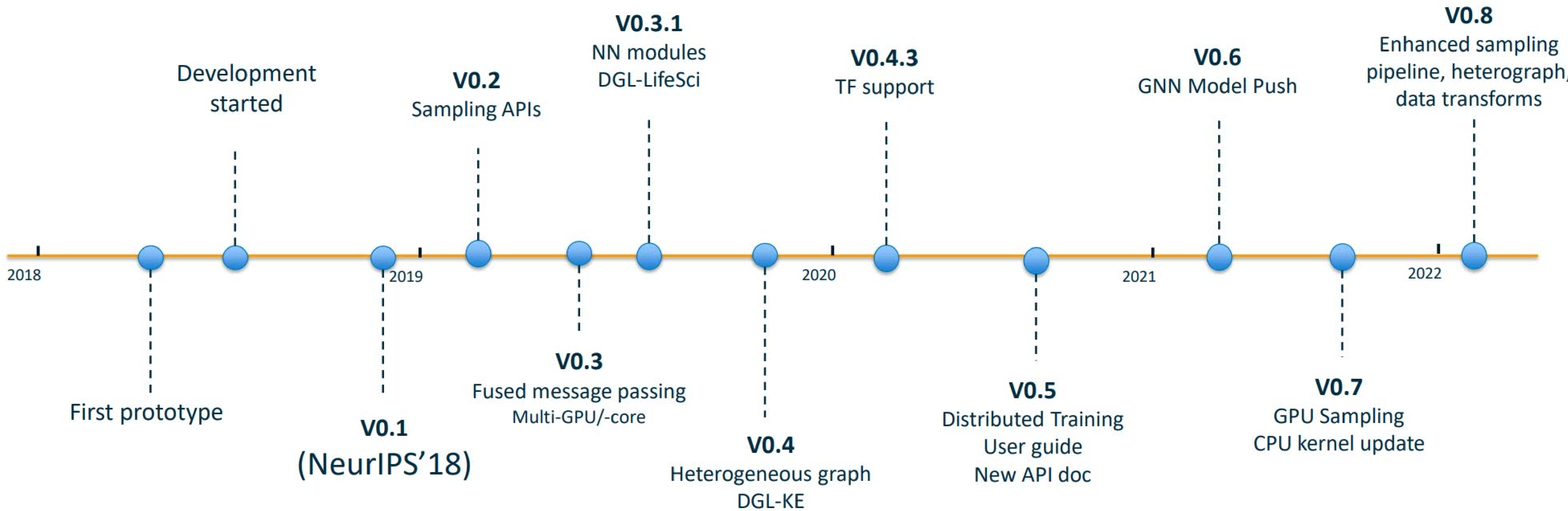
(所有材料都公开在 https://github.com/dglai/)



亚马逊云科技


GNNLens
GRAPHISTRY
可视化
复杂图
OpenHGNN
DGL-KE
知识图谱嵌入
DGL
GNN评测标准
OPEN GRAPH BENCHMARK
deepchem
DGL-LifeSci
生命科学
Amazon Neptune ML
neo4j
数据库
结构化自然语言处理
Graph4NLP
ArangoDB
亚马逊云科技

DGL三年开发历程

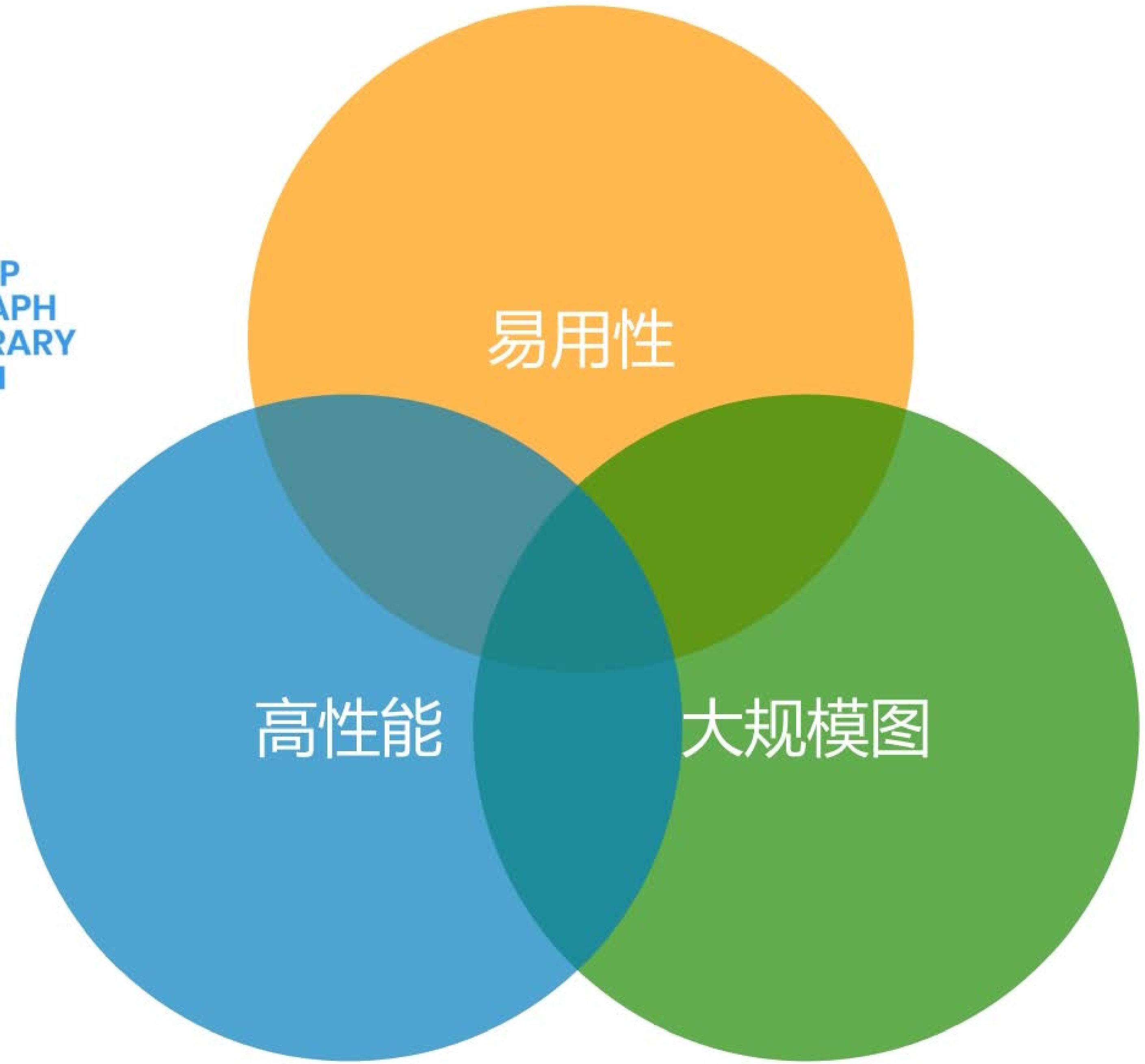


亚马逊云科技

开源图机器学习系统的核心挑战

DEEP GRAPH LIBRARY

PyG



Paddle Graph Learning

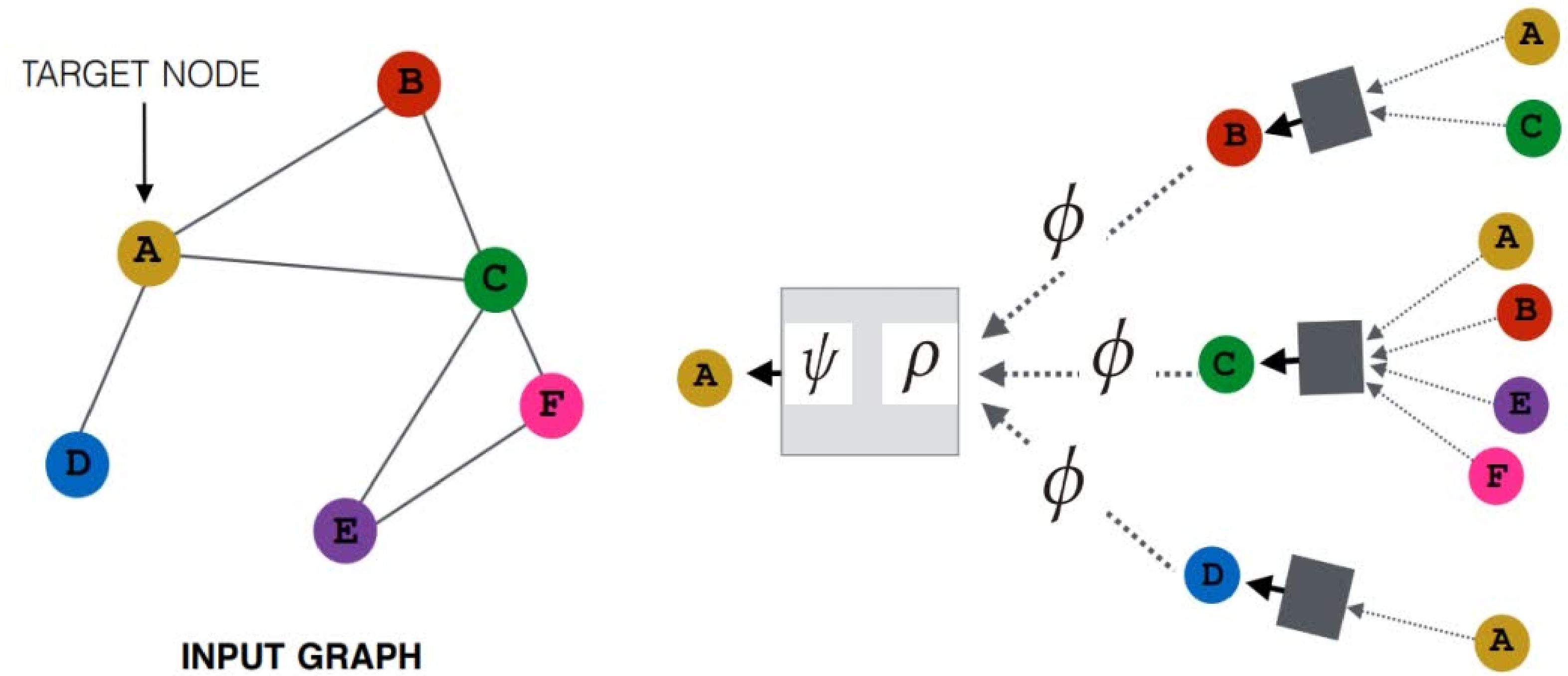
Jraph

graph-learn

亚马逊云科技

易用性+高性能

- 图神经网络入门门槛较高
- 编写高效代码不容易



Message passing in three stages

Message creation: $$m_e = \phi\left(x_u, x_v, w_e\right), (u, e, v) \in \mathcal{E},$$

Message aggregation: $$h_v = \rho\left(\{m_e : (u, e, v) \in \mathcal{E}\}\right),$$

Feature update: $$x_v^{new} = \psi\left(x_v, h_v\right), v \in \mathcal{V}.$$

亚马逊云科技

Graph Attention Network (GAT)

$$
\begin{aligned}
m_{j\to i} &= z_j = Wh_j, && (1)\\
e_{j\to i} &= LeakyReLU(W^{ATT}(z_i||z_j)), && (2)\\
\alpha_{j\to i} &= \frac{\exp(e_{j\to i})}{\sum_{k\in\mathcal{N}(i)}\exp(e_{k\to i})}, && (6)\\
r_i &= \sum_{j\in\mathcal{N}(i)} \alpha_{j\to i} m_{j\to i}, && (7)
\end{aligned}
$$

User-defined Function (UDF)

```python
def message_gat(edges):
    # equation (1)
    z_src, z_dst = edges.src['h'] @ W, edges.dst['h'] @ W
    # equation (2)
    e = leaky_relu(concat(z_src, z_dst, dim=1) @ W_att)
    return {'m': z_src, 'e': e}

def aggregate_func(nodes):
    # equation (6)
    alpha = softmax(nodes.mailbox['e'], dim=1)
    # equation (7)
    r = sum(alpha * nodes.mailbox['m'], dim=1)
    return {'r': r}
```

- **Intuitive**
- The system converts the irregular-shaped graph computation into fixed-shaped tensor computation by data duplication, sharding, etc. **(Less efficient)**
- Suitable for quick prototyping

Performance gap can be 10x ~ 100x !!

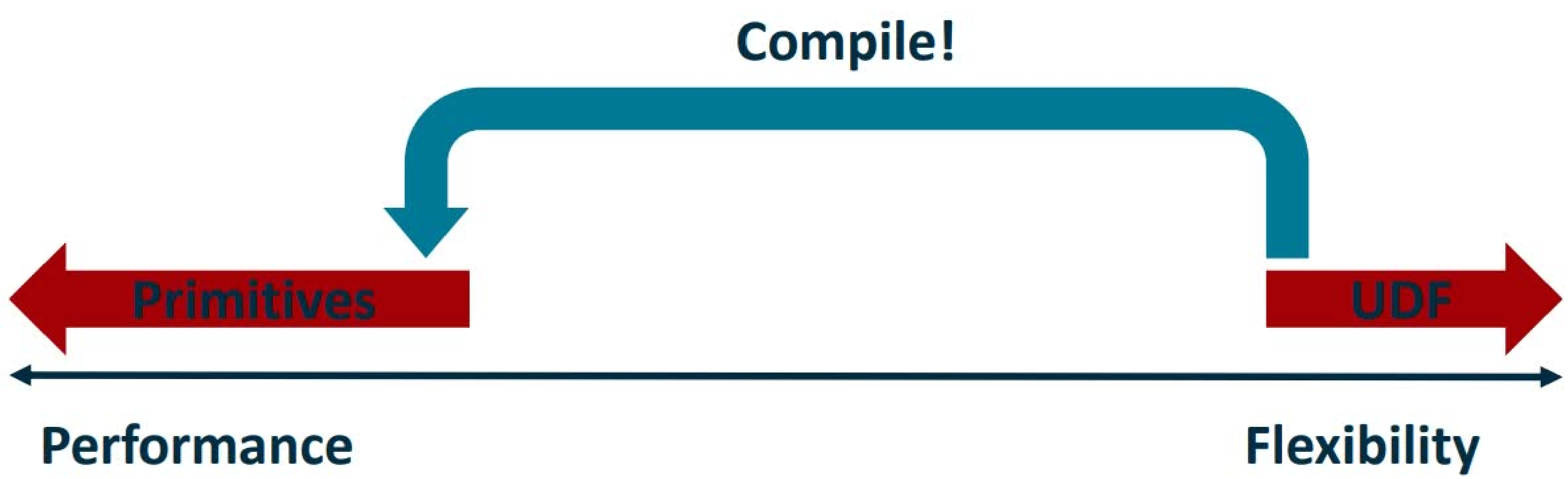
Specialized GNN Primitives

```python
def gat_dgl_primitives(graph, h):
    # equation (1)
    z_src = z_dst = h @ W
    # equation (2)
    el = z_src @ W_att_l
    er = z_dst @ W_att_r
    graph.srcdata.update({'m': z_src, 'el': el})
    graph.dstdata.update({'er': er})
    graph.apply_edges(dgl.u_add_v('el', 'er', 'e'))
    e = leaky_relu(graph.edata.pop('e'))
    # equation (6)
    e_max = dgl.copy_e_max(graph, e)
    e = exp(dgl.e_sub_v(graph, e, e_max))
    e_sum = dgl.copy_e_sum(graph, e)
    graph.edata['alpha'] = dgl.e_div_v(graph, e, e_sum)
    # equation (7)
    graph.update_all(dgl.u_mul_e('m', 'alpha', 'm'), dgl.sum('m', 'r'))
    return graph.dstdata['r']
```

- Each primitive directly maps to a low-level CUDA kernel **(Very efficient)**
- **Less intuitive**
- Suitable for performance critical scenarios

Graphiler: Optimizing GNN with Message Passing Data Flow Graph

- 利用编译器对用户代码进行无缝转换。



亚马逊云科技

Optimizations

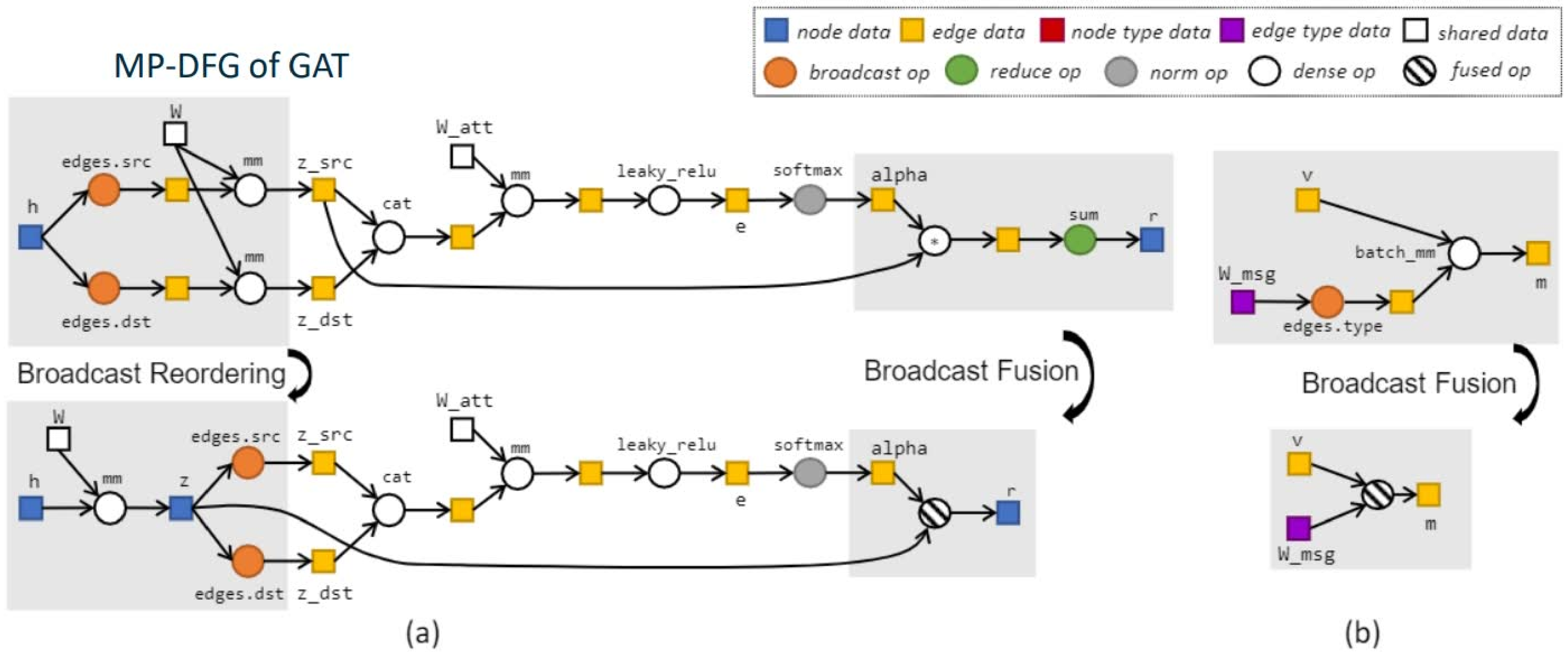


Figure 4: (a) Transformation on the MP-DFG of GAT. (b) Transformation on part of the MP-DFG of HGT

- Perform program optimization by pattern substitution.
- Two broadly applicable pattern substitution rules: *broadcast reordering* and *broadcast fusion*
- Unify the optimization space for homogeneous and heterogeneous GNNs.
- Check out our paper (MLSys'22) for more details!

大规模图

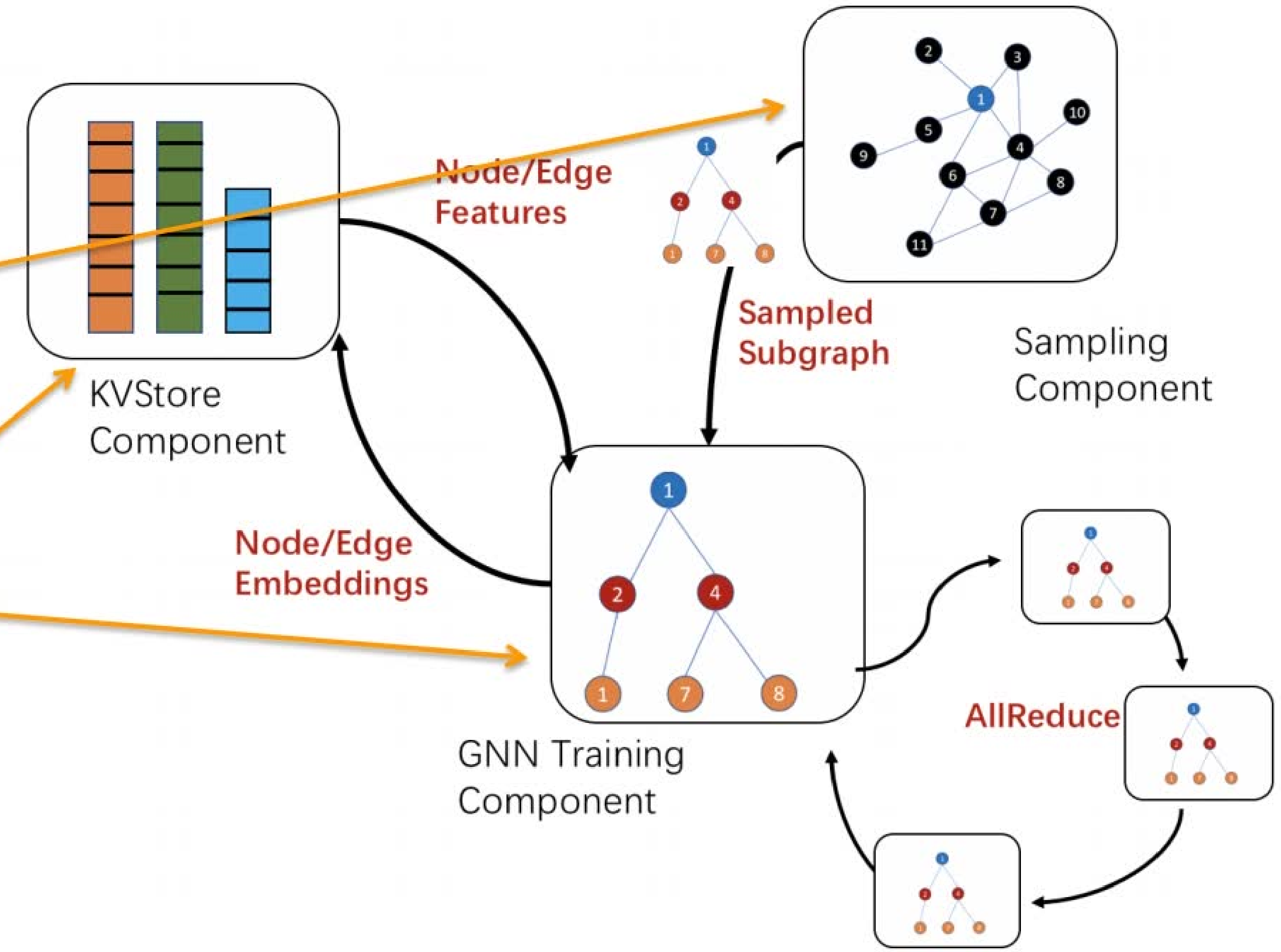
- 学术圈越来越关注大规模图数据
- 工业界图在百亿甚至千亿量级

Dataset	# Nodes	# Edges	Node features	# train nodes	# train links
OGBN-PRODUCT	2.4M	61.9M	100	197K	61.9M
AMAZON [6]	1.6M	264M	200	1.3M	264M
OGBN-PAPERS100M	111M	3.2B	128	1.2M	3.2B
MAG-LSC	240M	7B	756	1.1M	7B

巨图训练基于子图采样

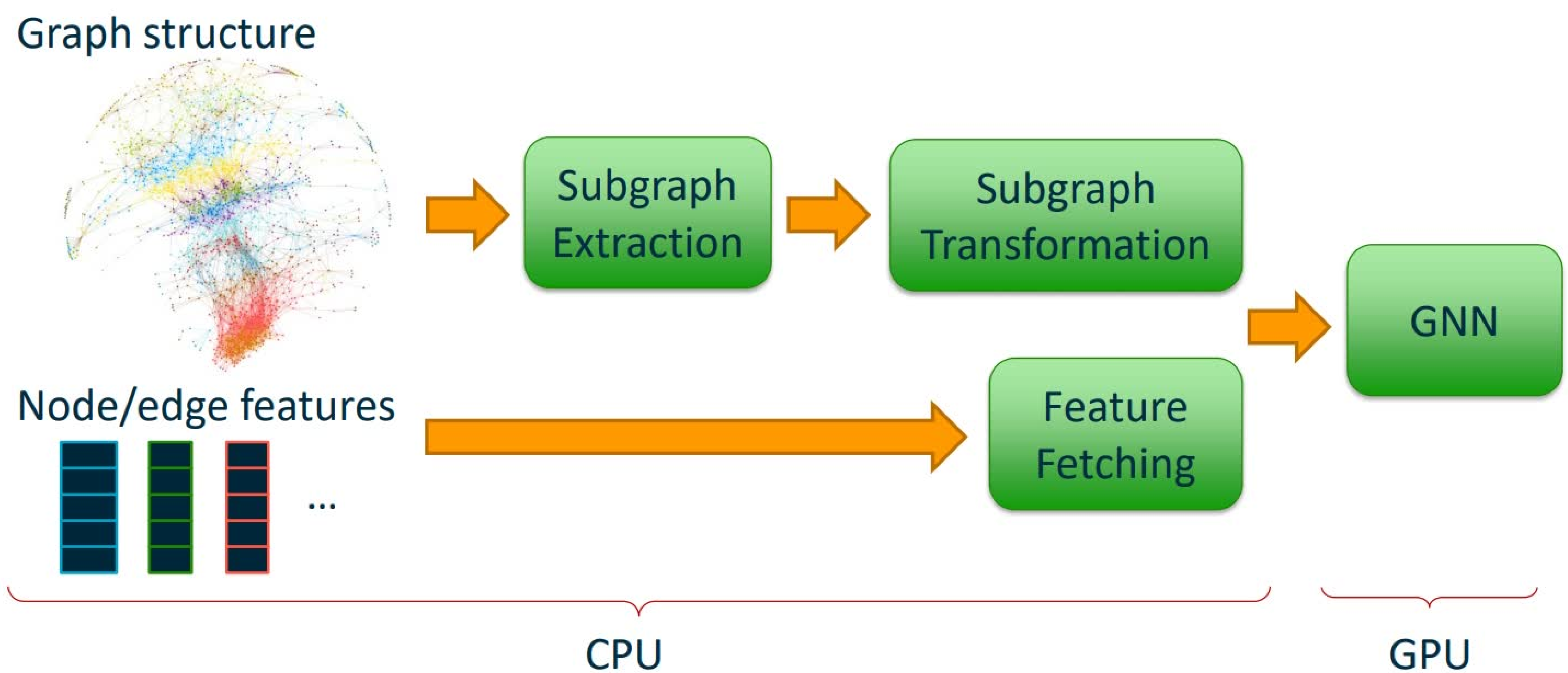
GNN的小批次训练（mini-batch training）基于子图采样

1. 对目标节点随机选取部分邻居节点，并迭代拓展。
2. 抽取采样的边形成子图。
3. 抽取子图特征。
4. 在子图上训练网络并更新参数。
5. 重复步骤 1直至收敛



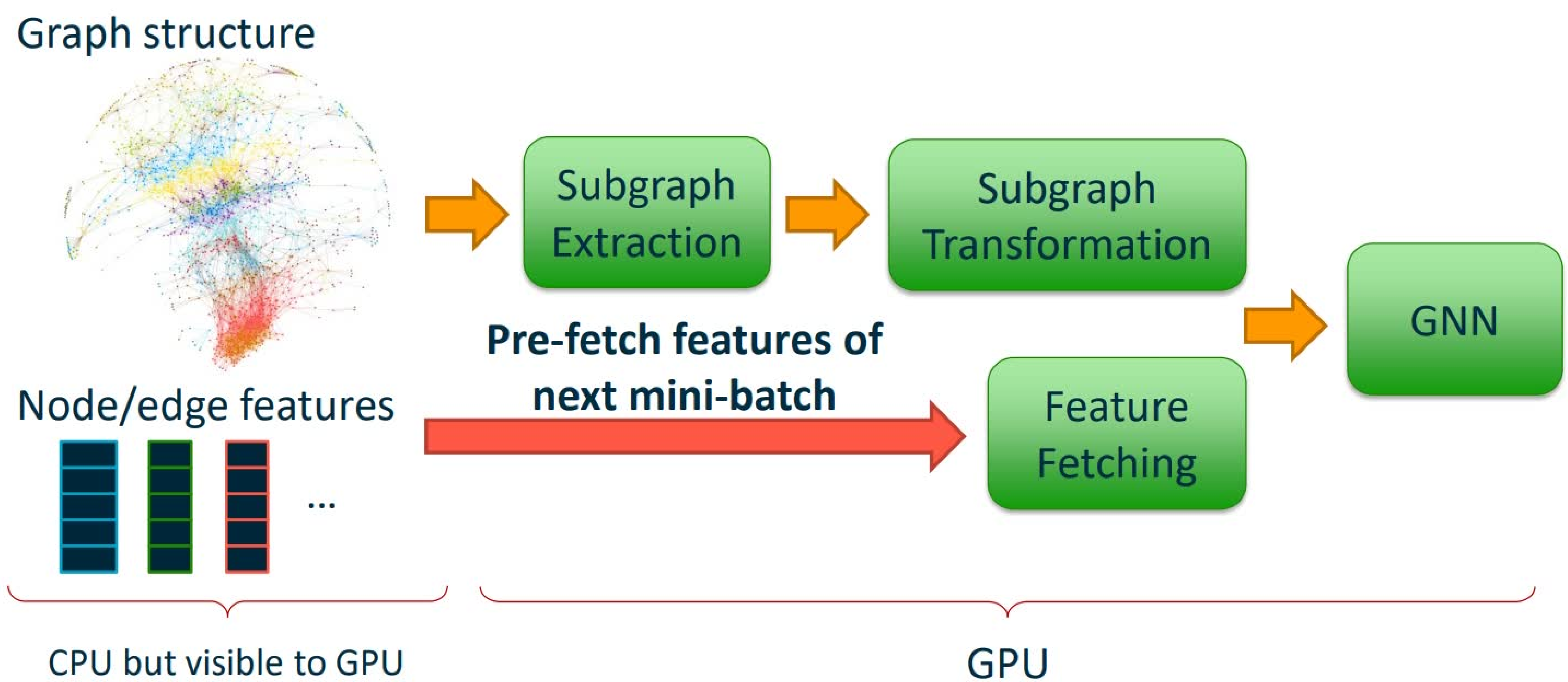
亚马逊云科技

单GPU训练流程



数据拷贝成为训练瓶颈

单GPU训练流程 (v0.8)



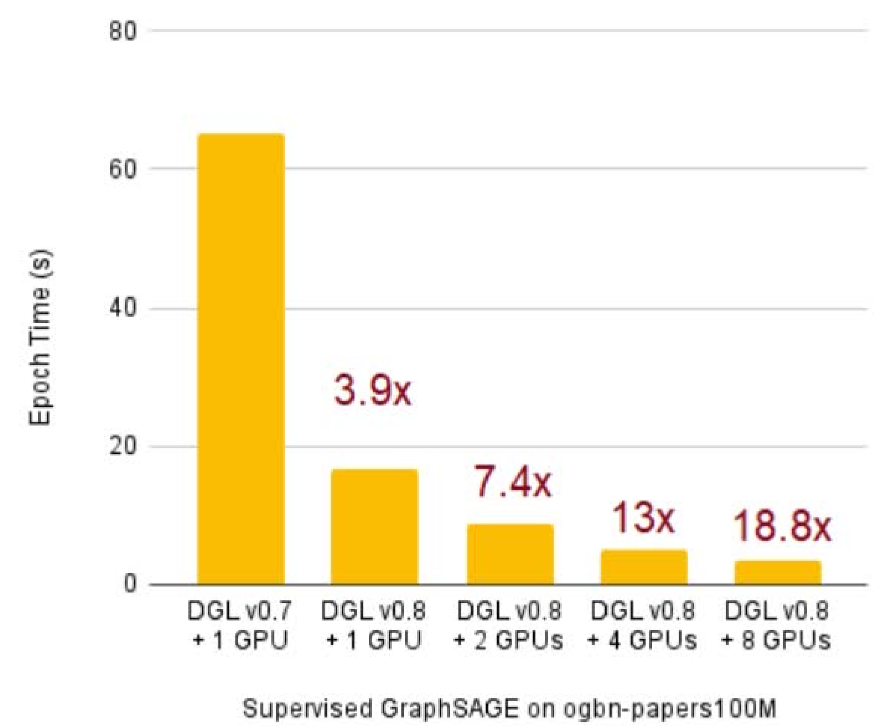
利用GPU加速子图采样

使用CUDA UVA技术

单GPU训练流程 (v0.8)

- 用户只需添加几行代码
- 在ogbn-papers100M训练GraphSAGE有将近4x性能提升

```
g = ...  # some DGLGraph data
train_nids = ...  # training node IDs
sampler = dgl.dataloading.MultiLayerNeighborSampler(
    fanout=[10, 15],
    prefetch_node_feats=['feat'],   # prefetch node feature 'feat'
    prefetch_labels=['label'],      # prefetch node label 'label'
)
dataloader = dgl.dataloading.NodeDataLoader(
    g, train_nids, sampler,
    device='cuda:0',     # perform sampling on GPU 0
    batch_size=1024,
    shuffle=True,
    use_uvm=True         # turn on UVM optimization
)
```

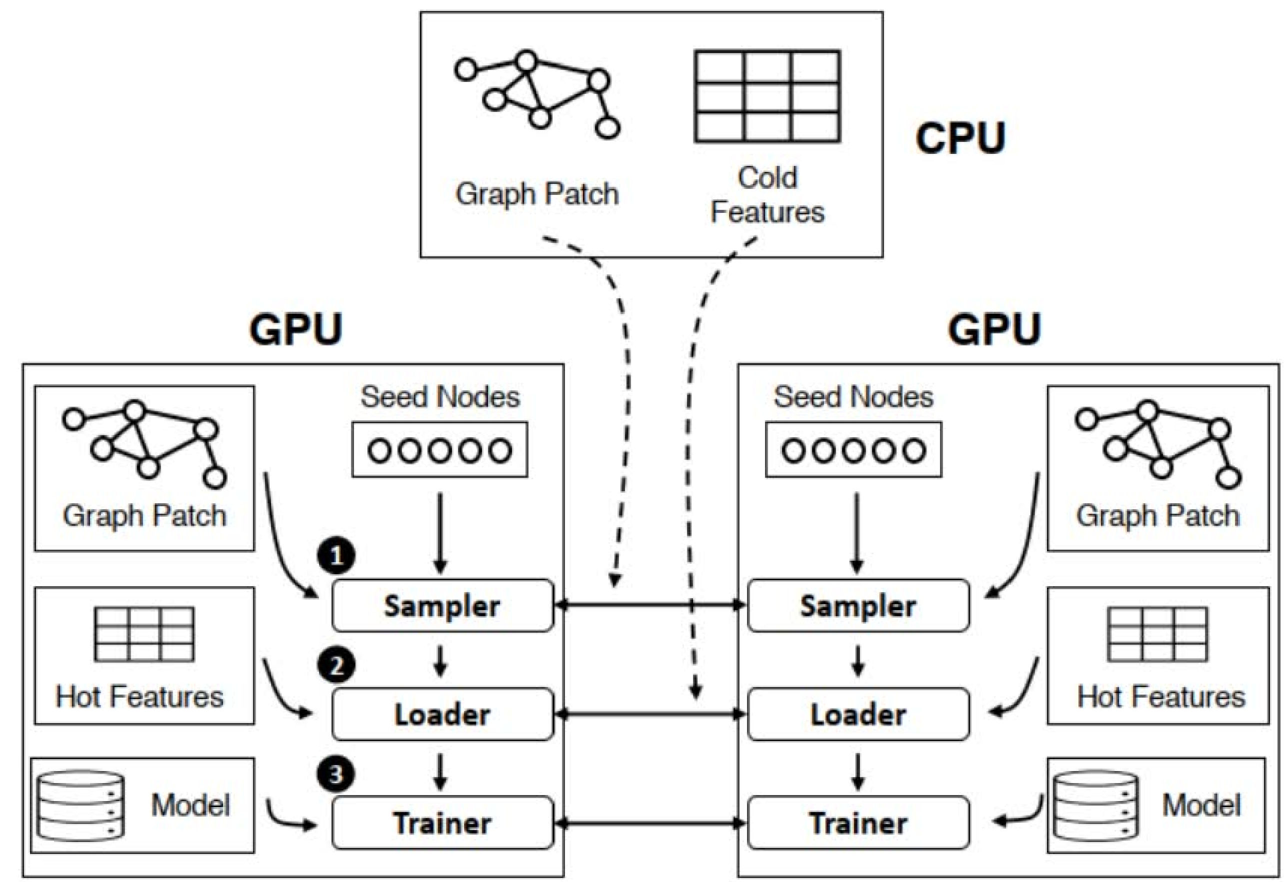


多GPU训练

- 对图进行分割并存储在多块GPU的内存中。 利用多GPU进行并行采样和训练。
- 如何设计高效的多GPU采样算法?
- 如何利用多GPU间的高速带宽?

TABLE I: Aggregate bandwidth (GBps) of NVLinks and PCIe on a DGX-2 GPU server [31]

	1-GPU	2-GPU	4-GPU	8-GPU
PCIe	32	32	64	128
NVLink	0	100	400	1200

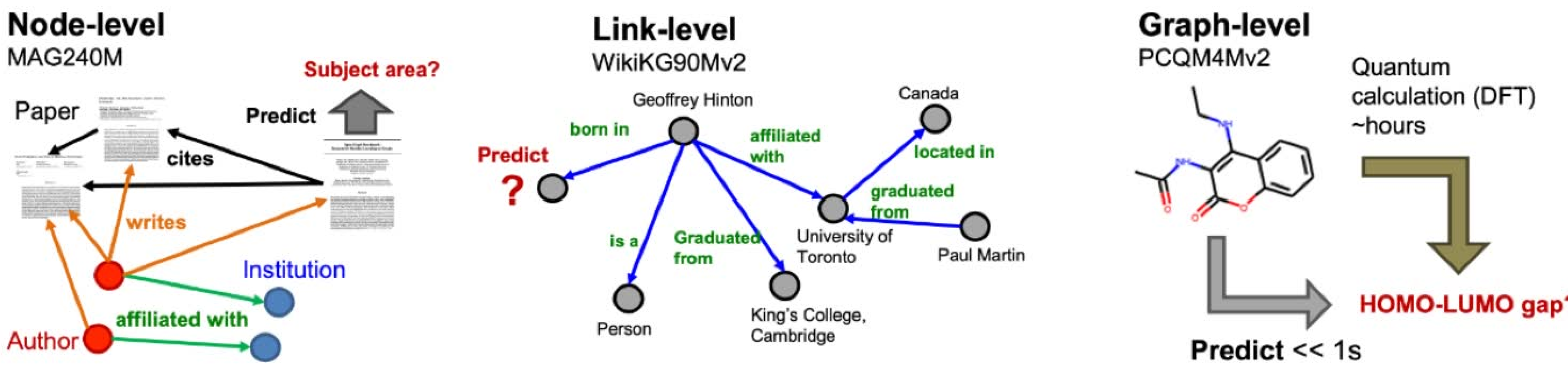


(Paper under submission)

训练成本

Overview of OGB-LSC 2022

We provide three OGB-LSC datasets that are unprecedentedly large in scale and cover prediction at the level of nodes, links, and graphs, respectively. An illustrative overview of the three OGB-LSC datasets is provided below.



- MAG240M：图结构30GB，节点特征200GB。
- CPU+GPU混合训练：g4dn.metal, 384GB CPU RAM, $7.824/hr
- 全GPU训练：4x g4dn.metal, 32x T4 GPU, $31.296/hr
- Web graph: 4.66B web pages (**>10x !!**)

亚马逊云科技

训练成本

- GNN模型推理代价也很高。

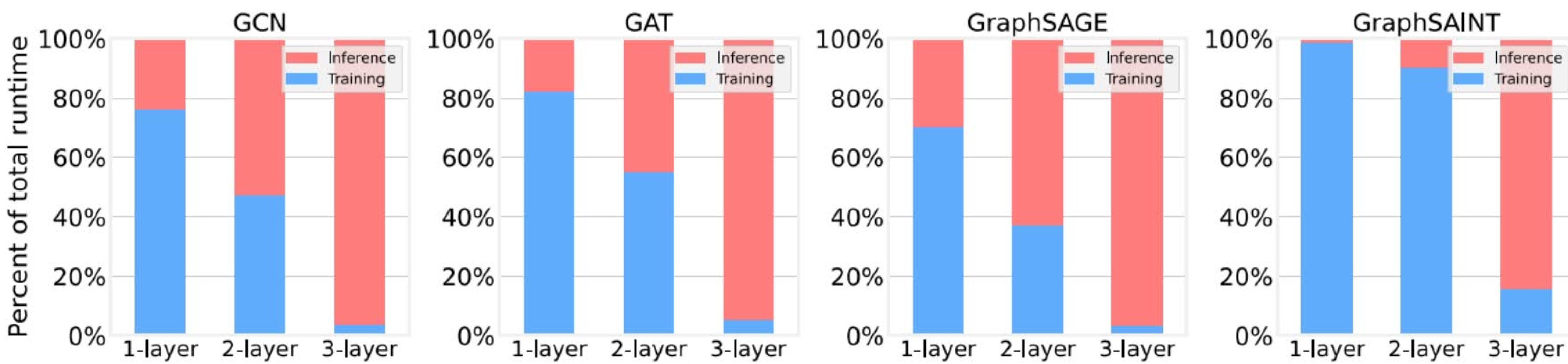
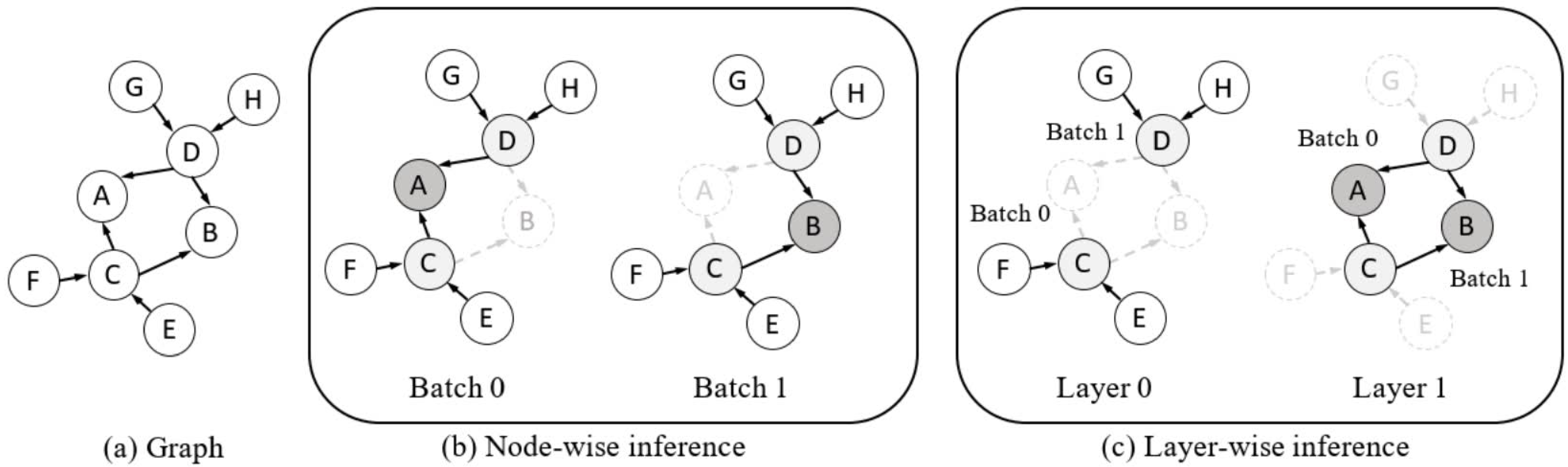


Figure 1: Composition of model training time and node-wise inference time in a model training pipeline for the OGBN-Products graph on a V100 GPU with 32GB memory.

亚马逊云科技

GNN推理



G
H
D
A
B
F
C
E
Batch 0
Batch 1
Layer 0
Layer 1
(a) Graph
(b) Node-wise inference
(c) Layer-wise inference

GNN推理

- 实现高效的GNN推理需要大幅修改原生模型实现。

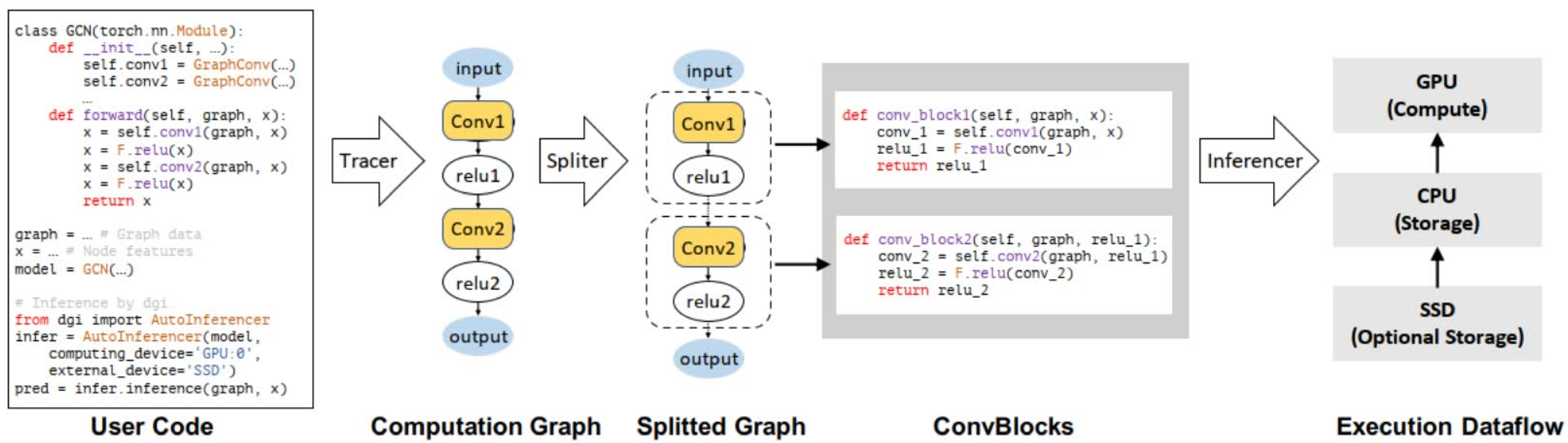
```python
class SAGE(nn.Module):
    def __init__(self, in_feats, n_hidden, n_classes):
        super().__init__()
        self.layers = nn.ModuleList()
        self.layers.append(dglnn.SAGEConv(in_feats, n_hidden, 'mean'))
        self.layers.append(dglnn.SAGEConv(n_hidden, n_hidden, 'mean'))
        self.layers.append(dglnn.SAGEConv(n_hidden, n_classes, 'mean'))
        self.dropout = nn.Dropout(0.5)
        self.n_hidden = n_hidden
        self.n_classes = n_classes

    def forward(self, blocks, x):
        h = x
        for l, (layer, block) in enumerate(zip(self.layers, blocks)):
            h = layer(block, h)
            if l != len(self.layers) - 1:
                h = F.relu(h)
                h = self.dropout(h)
        return h
```

```python
    def inference(self, g, device, batch_size, num_workers, buffer_device=None):
        feat = g.ndata['feat']
        sampler = dgl.dataloading.MultiLayerFullNeighborSampler(1, prefetch_node_feats=['feat'])
        dataloader = dgl.dataloading.DataLoader(
                g, torch.arange(g.num_nodes()).to(g.device), sampler, device=device,
                batch_size=batch_size, shuffle=False, drop_last=False,
                num_workers=num_workers)

        if buffer_device is None:
            buffer_device = device

        for l, layer in enumerate(self.layers):
            y = torch.empty(
                g.num_nodes(), self.n_hidden if l != len(self.layers) - 1 else self.n_classes,
                device=buffer_device, pin_memory=True)
            feat = feat.to(device)
            for input_nodes, output_nodes, blocks in tqdm.tqdm(dataloader):
                # use an explicitly contiguous slice
                x = feat[input_nodes]
                h = layer(blocks[0], x)
                if l != len(self.layers) - 1:
                    h = F.relu(h)
                    h = self.dropout(h)
                # be design, our output nodes are contiguous so we can take
                # advantage of that here
                y[output_nodes[0]:output_nodes[-1]+1] = h.to(buffer_device)
            feat = y
        return y
```


亚马逊云科技

全自动GNN推理

- 通过编译手段分析用户模型并对用户模型进行改写。
- 自动生成高效逐层推理代码。
- 自动搜索推理超参数，自适应底层硬件优化推理速度。



(Paper under submission)

总结

- DGL作为全球领先的开源图神经网络系统的技术特点
- 开源社区介绍
- 图神经网络系统仍然面临着诸如易用性、高性能和大规模图方面的挑战。
- 在编译、分布式等领域的最新研究成果。

亚马逊云科技

欢迎使用并贡献DGL

- 用户论坛，Slack，微信群，知乎专栏
- 或者加入我们！实习岗位常年开放！

cn-ai-intern@amazon.com

亚马逊云科技

亚马逊云科技

谢谢！