第7章：分布式训练

(English Version)

DGL采用完全分布式的方法，可将数据和计算同时分布在一组计算资源中。在本节中， 我们默认使用一个集群的环境设置(即一组机器)。DGL会将一张图划分为多张子图， 集群中的每台机器各自负责一张子图(分区)。为了并行化计算，DGL在集群所有机器上运行相同的训练脚本， 并在同样的机器上运行服务器以将分区数据提供给训练器。

对于训练脚本，DGL提供了分布式的API。它们与小批次训练的API相似。用户仅需对单机小批次训练的代码稍作修改就可实现分布式训练。 以下代码给出了一个用分布式方式训练GraphSage的示例。仅有的代码修改出现在第4-7行：1)初始化DGL的分布式模块，2)创建分布式图对象，以及 3)拆分训练集，并计算本地进程的节点。其余代码保持不变，与 mini_cn-batch training 类似， 包括：创建采样器，模型定义，模型训练的循环。

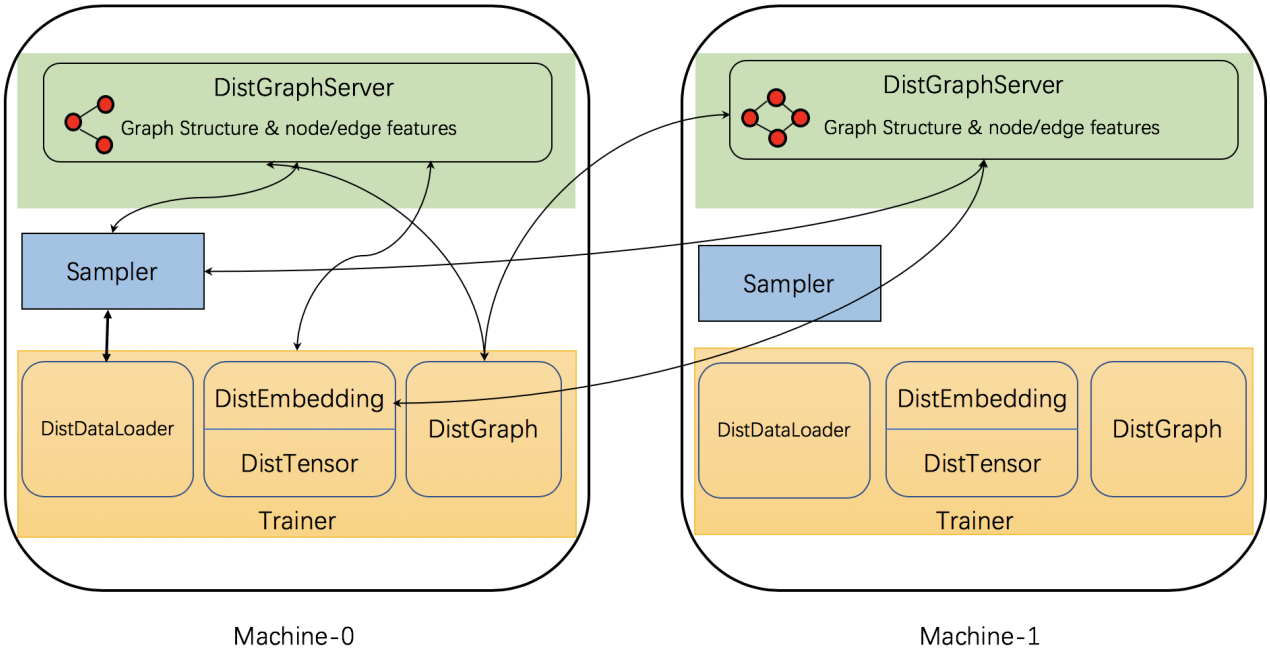
```
import dgl
import torch as th

dgl.distributed.initialize('ip_config.txt')
th.distributed.init_process_group(backend='gloo')
g = dgl.distributed.DistGraph('graph_name', 'part_config.json')
pb = g.get_partition_book()
train_nid = dgl.distributed.node_split(g.ndata['train_mask'], pb, force_even=True)

# 创建采样器
sampler = NeighborSampler(g, [10,25],
                          dgl.distributed.sample_neighbors,
                          device)

dataloader = DistDataLoader(
    dataset=train_nid.numpy(),
    batch_size=batch_size,
    collate_fn=sampler.sample_blocks,
    shuffle=True,
    drop_last=False)

# 定义模型和优化器
model = SAGE(in_feats, num_hidden, n_classes, num_layers, F.relu, dropout)
model = th.nn.parallel.DistributedDataParallel(model)
loss_fcn = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=args.lr)

# 模型训练的循环
for epoch in range(args.num_epochs):
    for step, blocks in enumerate(dataloader):
        batch_inputs, batch_labels = load_subtensor(g, blocks[0].srcdata[dgl.NID],
                                                    blocks[-1].dstdata[dgl.NID])
        batch_pred = model(blocks, batch_inputs)
        loss = loss_fcn(batch_pred, batch_labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

在一个集群的机器上运行训练脚本时，DGL提供了一些工具，可将数据复制到集群的计算机上，并在所有机器上启动训练任务。

Note: 当前版本的分布式训练API仅支持PyTorch后端。

Note: 当前版本的实现仅支持具有一种节点类型和一种边类型的图。

DGL实现了一些分布式组件以支持分布式训练，下图显示了这些组件及它们间的相互作用。



具体来说，DGL的分布式训练具有三种类型的交互进程： *服务器*， *采样器* 和 *训练器*。

- *服务器进程* 在存储图分区数据(这包括图结构和节点/边特征)的每台计算机上运行。 这些服务器一起工作以将图数据提供给训练器。请注意，一台机器可能同时运行多个服务器进程，以并行化计算和网络通信。
- *采样器进程* 与服务器进行交互，并对节点和边采样以生成用于训练的小批次数据。
- *训练器进程* 包含多个与服务器交互的类。它用 `DistGraph` 来获取被划分的图分区数据，用 `DistEmbedding` 和 `DistTensor` 来获取节点/边特征/嵌入，用 `DistDataLoader` 与采样器进行交互以获得小批次数据。

在初步了解了分布式组件后，本章的剩余部分将介绍以下分布式组件：

- 7.1 分布式训练所需的图数据预处理
- 7.2 分布式计算的API
- 7.3 运行分布式训练/推断所需的工具