1.6 在GPU上使用DGLGraph

(English Version)

用户可以通过在构造过程中传入两个GPU张量来创建GPU上的 `DGLGraph` 。另一种方法是使用 `to()` API将 `DGLGraph` 复制到GPU，这会将图结构和特征数据都拷贝到指定的设备。

```
>>> import dgl
>>> import torch as th
>>> u, v = th.tensor([0, 1, 2]), th.tensor([2, 3, 4])
>>> g = dgl.graph((u, v))
>>> g.ndata['x'] = th.randn(5, 3)    # 原始特征在CPU上
>>> g.device
device(type='cpu')
>>> cuda_g = g.to('cuda:0')          # 接受来自后端框架的任何设备对象
>>> cuda_g.device
device(type='cuda', index=0)
>>> cuda_g.ndata['x'].device         # 特征数据也拷贝到了GPU上
device(type='cuda', index=0)

>>> # 由GPU张量构造的图也在GPU上
>>> u, v = u.to('cuda:0'), v.to('cuda:0')
>>> g = dgl.graph((u, v))
>>> g.device
device(type='cuda', index=0)
```

任何涉及GPU图的操作都是在GPU上运行的。因此，这要求所有张量参数都已经放在GPU上，其结果(图或张量)也将在GPU上。此外，GPU图只接受GPU上的特征数据。

```
>>> cuda_g.in_degrees()
tensor([0, 0, 1, 1, 1], device='cuda:0')
>>> cuda_g.in_edges([2, 3, 4])                          # 可以接受非张量类型的参数
(tensor([0, 1, 2], device='cuda:0'), tensor([2, 3, 4], device='cuda:0'))
>>> cuda_g.in_edges(th.tensor([2, 3, 4]).to('cuda:0'))  # 张量类型的参数必须在GPU上
(tensor([0, 1, 2], device='cuda:0'), tensor([2, 3, 4], device='cuda:0'))
>>> cuda_g.ndata['h'] = th.randn(5, 4)                  # ERROR! 特征也必须在GPU上!
DGLError: Cannot assign node feature "h" on device cpu to a graph on device
cuda:0. Call DGLGraph.to() to copy the graph to the same device.
```