GPU 编程实验 2--图采样

第一节，邻居采样

背景：

近几年，图神经网络（GNN）成为了机器学习领域一个热门的话题。以 GraphSAGE 为代表的 GNN 模型，在工业界的推荐系统、数据风控、药物合成等领域都有了落地应用。GraphSAGE 通过聚合图（Graph）中的邻域信息，能够获得中心节点的嵌入（embedding，特征向量）。将节点嵌入输入到下游任务中，比如节点分类、链路预测、子图匹配等。

为了聚合图的邻域信息，我们首先需要获取节点的邻域信息，即某一个节点有哪些相邻的节点，在 GraphSAGE 中的该操作即为对图进行采样（Graph Sampling），举个例子，如 Figure1(1)所示：

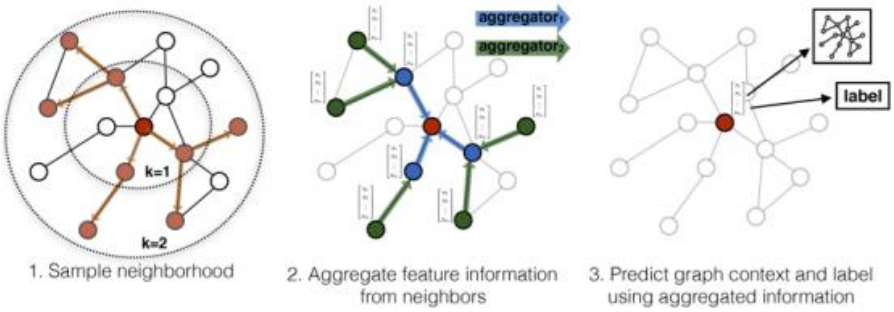

Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

从中心节点 V0 出发，从 V0 的 5 个邻居中随机选取 3 个点作为"第一跳"邻居节点；然后从"第一跳"的每个节点出发，分别获取其 2 个"第二跳"邻居节点。这样，我们就获得了中心节点 V0 的二跳邻域信息。图采样的跳数是 GNN 模型中的一个可变参数，为了简单起见，我们这里仅考虑一跳。

PS：Figure1(1)的过程完成之后，我们取出采样得到的节点的原始特征向量，然后根据采样得到的图中的邻接关系，对特征向量进行聚合，最终可以得到中心节点的 Embedding，如 Figure1(2)所示。然后 Embedding 可以被用于下游的节点分类任务，如 Figure1(3)所示。

实验目标：

本次实验，我们尝试利用 GPU 的并行计算能力，来加速图采样实现。

图采样具体过程：

我们仅考虑最简单的均匀随机采样。一个中心节点 V0，它有 5 个相连的点，也称为邻居，V1，V2，V3，V4，V5，我们对其进行 3 次**均匀**的**随机有放回**的采样。"有放回"也就是说，3 次采样相互**独立**。"均匀"则是代表每个邻居被取到的概率相等，即每次采样，V1-V5 每个点都有 1/5 的概率被取到。

经过 3 次采样后，一种可能的输出就是，我们获得了 V0 的 3 个邻居，V1，V3，V4。也有可能，输出 V1，V1，V3… **如果节点邻居本身不足 3 个，那么必然会有重复被采样到的（为了实现简单，我们这里暂时采用这种设计）**。

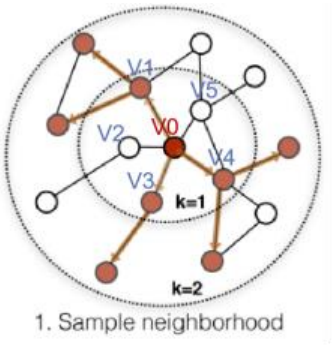

Figure 2

图数据的存储格式：

图数据结构中邻接关系可以由邻接矩阵来表达。比如：

Figure 3

从上面邻接矩阵我们可以看到，ID 为 1 的点，包含 ID 为 0,3,5 的三个邻居。可以看到，图中每个点的邻居数不一定相同，矩阵通常是稀疏的。为了节省存储空间，我们往往采用 CSR 格式来存储邻接矩阵：

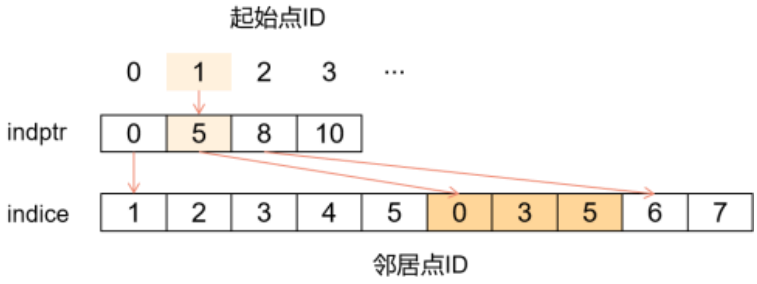

Figure 4

在 CSR 中，我们把所有点的邻居存在一块连续的内存空间中。以 ID 为 1 的点为例，我们查询第一个数组 indptr，得到 1 这个点的邻居在第二个数组 (indice) 中的偏移量 indptr[1] = 5；同时，我们可以得到 ID 为 1 的点的总邻居数 indptr[2]-indptr[1] = 3。然后我们即可知道 1 这个点的所有邻居 ID，indice[5], indice[6], indice[7]。Indptr 的最后一个元素会存储整张图中所有的边数。

代码实现：

那么我们如何用 GPU 来实现图采样呢?

第一，我们需要为 indtpr 和 indice 分别分配一块 GPU 内存，然后我们利用 GPU 线程来访问这两块内存。

第二，为了实现随机选取邻居的功能，我们需要让 GPU 来生成随机数（我们提供函数），并根据随机产生的下标[0, N), N 为邻居数，来访问邻接矩阵；

第三，我们回顾下图采样的原理，发现每个点多次采样是相互独立的，且不同起始点之间也是独立的，那么我们就可以利用 GPU 来做并行采样。

实验要求：

图数据集 reddit，

为了方便大家使用，我们直接提供 CSR 文件以及相应地读取到 host 内存的代码。实验任务是实现 GPU 相关的代码，对 232965 个起始点做 1 跳邻居采样，每个起始点采样 4 次，获得 4 个邻居的 ID，存在结果数组 d_dst_ids 中。然后将采样得到的结果返回 host 端。（如果一个点没有邻居，每次采样返回-1，如果某一个采样起始点 ID 是-1，那么每次采样结果直接返回-1）并输出核函数运行时间。

第二节，统计指定点被采样次数

在第一节实验中，我们已经获得了经过图采样之后的顶点 ID 序列，为了研究其统计规律，我们现在需要统计一些点被采样到的次数。

比如，在第一步我们输出了一串节点的 ID 序列，0,4,1,2,7,3…由于采样的随机性，序列中 ID 是乱序的，然而，我们需要知道其中 0,2,4,6…等偶数节点被访问的次数，那么我们需要对序列数据进行统计。

在本节实验中，我们采用 GPU 来统计节点序列中指定节点的出现次数。我们给出未经优化的 GPU 代码，需要各位同学结合已学的知识进行优化。

提示，可以使用 shared memory 来优化性能。

实验要求：

我们复用第一节的实验代码，以 d_dst_ids 作为输入，然后修改第一次实验中邻居采样的个数，从 4 改为 1000。 要求统计 ID 为 40(0, 40, 80, 120, 160…)整数倍，且 ID 小于 40000 的节点的采样次数，即一共有 1000 个节点需要被统计。将统计结果输出到 host，打印出前 100 个，并计算统计的 GPU Kernel 执行所需时间。

需要大家完成代码编写，并运行成功，以报告的形式提交，第一节指标为 1. 第一跳的前 20 个结果打印截图，2. 图采样 GPU 核函数执行时间截图，3. 提交代码 1。第二节指标为 1.打印前 100 个节点统计次数, 2. GPU 统计核函数执行时间截图，性能需要高于提供的代码, 3. 提交代码 2

实验平台及操作：

服务器配置：

我们提供一台包含 RTX8000GPU 的服务器，大家需要连接校网再登陆服务器。

服务器 IP 地址为: 10.12.86.210，端口为 10027

我们为每个同学都创建了 linux 账户，账户名为 st+学号，初始密码为 rc4ml@ai_course

CUDA 路径/usr/local/cuda/bin

大家需要在当前.bashrc 文件最底下添加一行便于使用 CUDA 编译器:

export PATH=$PATH:/usr/local/cuda/bin

数据集：

数据集位置：/ai_course　可以从此路径中拷贝到和编译出来的可执行文件相同路径中。

GPU 任务执行：

编译，使用 nvcc。

运行，使用 slurm 提交。

由于使用人数较多，我们设置了任务提交系统。大家需要通过 slurm 提交任务，从而保证任务运行时，GPU 资源没有竞争，最终每个人跑出来真实性能。

具体提交方式：

修改脚本 test.sh：只需要修改白色部分,改为你们自己的可执行文件

```
#!/bin/bash

#SBATCH -n 12 # 指定核心数量
#SBATCH -N 1 # 指定node的数量
#SBATCH -t 0-5:00 # 运行总时间，天数-小时数-分钟， D-HH:MM
#SBATCH -p debug # 提交到哪一个分区
#SBATCH --gres=gpu:1 # 需要使用多少GPU，n是需要的数量
./graphsampling
```

递交任务 sbatch test.sh：

```
sunjie@ccai-PowerEdge-T640:~$ sbatch test.sh
Submitted batch job 84
sunjie@ccai-PowerEdge-T640:~$ sbatch test.sh
Submitted batch job 85
sunjie@ccai-PowerEdge-T640:~$ sbatch test.sh
Submitted batch job 86
sunjie@ccai-PowerEdge-T640:~$ sbatch test.sh
Submitted batch job 87
sunjie@ccai-PowerEdge-T640:~$ sbatch test.sh
Submitted batch job 88
sunjie@ccai-PowerEdge-T640:~$ sbatch test.sh
Submitted batch job 89
sunjie@ccai-PowerEdge-T640:~$ sbatch test.sh
Submitted batch job 90
```

递交任务后可以得到任务 id。

查看当前任务为是否完成：运行 squeue 指令：

```
sunjie@ccai-PowerEdge-T640:~$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                86     debug  test.sh   sunjie PD       0:00      1 (Resources)
                87     debug  test.sh   sunjie PD       0:00      1 (Priority)
                88     debug  test.sh   sunjie PD       0:00      1 (Priority)
                89     debug  test.sh   sunjie PD       0:00      1 (Priority)
                90     debug  test.sh   sunjie PD       0:00      1 (Priority)
                85     debug  test.sh   sunjie  R       0:01      1 ccai-PowerEdge-T640
```

任务完成之后程序打印的内容可以在 slurm-jobid.out 中看到

```
sunjie@ccai-PowerEdge-T640:~$ tail slurm-90.out
91 47943
92 50622
93 37085
94 41791
95 38580
96 32363
97 45795
98 33342
99 41094
time cost: 2.19693 ms
```

大家可以通过 nvidia-smi 查看当前 GPU 使用情况，如果发现 GPU 正在被使用，就重新提交一下任务，取多次的数据。

注意：为了保证所有人的使用效率，未遵循作业提交系统的基本操作，一经发现，取消本次实验成绩！其他如果使用方面有问题可以及时联系助教孙杰。