Ginex Experiment

2024. 08. 23

Presentation by Suhwan Shin

sshshin@dankook.ac.kr

Contents

Background - Ginex (VLDB'22)

- Motivation
  - 그래프 데이터 셋이 커지면서 메모리 내 GNN 학습은 메모리 용량의 제한의 한계가 있으며, 분산 학습은 메모리를 확장할 수 있지만 비용과 네트워크가 문제임
  - SSD를 메모리 대신 사용하는 것은 비용 측면에서 효율적이지만 낮은 대역폭과 바이트 주소 지정 불가로 I/O 요청을 최소화하기 어려움

- Ginex System
  - Ginex는 GNN 학습 파이프라인을 재구성하여 I/O 작업을 줄이고 메모리 내 캐싱의 사용을 최적화함
  - GNN 학습 파이프라인 재구성: 컴파일러 최적화 기술에서 영감을 받아 Ginex는 학습 과정에서 샘플링과 집합(Gather) 단계를 분리
    - 컴파일러 최적화에서 영감을 얻은 'Inspector-Executor' 모델에 기반을 두고 있으며, 이를 통해 최적의 캐싱 메커니즘을 구현
    - 캐시 교체 정책의 최적화를 위해 Belady의 알고리즘을 사용
      - 미래에 사용될 데이터에 대한 정보를 미리 분석하고 가장 적절한 데이터를 캐시에 유지하도록 함

- Evaluation
  - 기존 방법에 비해 학습 시간을 단축시켰으며, 최대 2.67배 throughput을 처리
  - 클라우드 환경에서 비용 절감

Background - FlashGNN (HPCA'24)

- Motivation
  - 전통적인 CPU나 GPU 기반 시스템은 그래프 데이터셋의 크기가 커짐에 따라 하드웨어 활용도가 떨어지고, 분산 시스템에서는 비용이 문제가 됨
  - SSD를 활용하는 방법은 PCIe 버스를 통한 느린 데이터 전송이 여전히 Bottleneck임

- FlashGNN System
  - FlashGNN은 새로운 노드별 GNN 학습 방법과 효율적인 플래시 요청 스케줄링 알고리즘을 도입함
  - 고성능 서브그래프 생성 방법을 포함하여, 사용할 수 있는 데이터를 최대한 활용함으로써 반복적인 플래시 접근 필요성을 줄임

- Evaluation
  - Ginex보다 좋은 성능을 보이며, 대규모 그래프 데이터 셋에서 4.89~11.83배 속도가 향상되었으며, 57.14 ~ 192.66배 에너지 절감
  - SmartSAGE+ 시스템보다 최대 23.17배 효율적임

Ginex & FlashGNN

- Focus
  - Ginex
    - **DRAM 기반 학습 시스템에서 SSD 기반으로 전환할 때의 성능 저하를 최소화** 하는 것
    - → I/O 작업 최적화와 캐시 관리
  - FlashGNN
    - **SSD의 내부 구조를 활용하여 GNN 학습을 최적화** 하는 것
    - SSD를 저장 장치로 사용하는 것을 넘어, SSD 컨트롤러 내부에서 직접 학습을 수행하여 PCIe 버스 병목 현상 해결
    - → SSD 컨트롤러 내부에서 직접 학습 연산을 수행

Experiment Setup - Issue

```
super@super:~/sh/Ginex$ python3 prepare_dataset.py
Loading dataset...
ogbn-papers100M has been updated.
Will you update the dataset now? (y/N)
y
This will download 56.17GB. Will you proceed? (y/N)
y
Using exist file papers100M-bin.zip
Extracting ./dataset/papers100M-bin.zip
Processing...
Loading necessary files...
This might take a while.
Processing graphs...
100%|| 1/1 [00:00<00:00, 19239.93it/s]
Converting graphs into PyG objects...
100%|| 1/1 [00:00<00:00, 4619.28it/s]
Saving...
Done!
/home/super/.local/lib/python3.9/site-packages/ogb/nodeproppred/dataset_pyg.py:69: FutureWarning: You are using `torch.load` with `weights_only=False`
code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the defau
e allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add_safe_globals`. We recommend you st
to this experimental feature.
  self.data, self.slices = torch.load(self.processed_paths[0])
Done!
Creating coo/csc/csr format of dataset...
Done!
Saving indptr...
Done!
Saving indices...
Done!
Saving features...
/home/super/sh/Ginex/prepare_dataset.py:61: DeprecationWarning: __array__ implementation doesn't accept a copy keyword, so passing copy=False failed. _
  features_mmap[:] = features[:]
Done!
Saving labels...
/home/super/sh/Ginex/prepare_dataset.py:68: DeprecationWarning: __array__ implementation doesn't accept a copy keyword, so passing copy=False failed. _
  labels_mmap[:] = labels[:]
Done!
Making conf file...
Traceback (most recent call last):
  File "/home/super/sh/Ginex/prepare_dataset.py", line 88, in <module>
    json.dump(mmap_config, open(conf_path, 'w'))
  File "/usr/lib/python3.9/json/__init__.py", line 179, in dump
    for chunk in iterable:
  File "/usr/lib/python3.9/json/encoder.py", line 431, in _iterencode
    yield from _iterencode_dict(o, _current_indent_level)
  File "/usr/lib/python3.9/json/encoder.py", line 405, in _iterencode_dict
    yield from chunks
  File "/usr/lib/python3.9/json/encoder.py", line 438, in _iterencode
    o = _default(o)
  File "/usr/lib/python3.9/json/encoder.py", line 179, in default
    raise TypeError(f'Object of type {o.__class__.__name__} '
TypeError: Object of type int64 is not JSON serializable
```

[bug] TypeError: Object of type 'int64' is not JSON serializable #7 **New issue**

Open WWWzq-01 opened this issue on Jul 8 · 0 comments

WWWzq-01 commented on Jul 8

```
Traceback (most recent call last):
  File "prepare_dataset.py", line 89, in <module>
    json.dump(mmap_config, open(conf_path, 'w'))
  File "/home/wzq/miniconda3/envs/Ginex/lib/python3.6/json/__init__.py", line 179, in dump
    for chunk in iterable:
  File "/home/wzq/miniconda3/envs/Ginex/lib/python3.6/json/encoder.py", line 430, in _iterencode
    yield from _iterencode_dict(o, _current_indent_level)
  File "/home/wzq/miniconda3/envs/Ginex/lib/python3.6/json/encoder.py", line 404, in _iterencode_dict
    yield from chunks
  File "/home/wzq/miniconda3/envs/Ginex/lib/python3.6/json/encoder.py", line 437, in _iterencode
    o = _default(o)
  File "/home/wzq/miniconda3/envs/Ginex/lib/python3.6/json/encoder.py", line 180, in default
    o.__class__.__name__)
TypeError: Object of type 'int64' is not JSON serializable
```

WWWzq-01 mentioned this issue on Jul 8

fix the bug: `TypeError: Object of type 'int64' is not JSON serializable` #8 **Closed**

Assignees: No one assigned

Labels: None yet

Milestone: No milestone

Development: No branches or pull requests

Notifications: Customize — Subscribe — You're not receiving notifications from this thread.

Experiment (CPU)

- baseline

```
super@super:~/sh/Ginex$ sudo -E PYTHONPATH=/home/super/.local/lib/python3.8/site-packages cgexec -g memory:8gb python3.8 -W ignore run_baseline.py
Epoch 00: 100%|| 1206179/1207179 [01:15<00:00, 15987.56it/s]
Epoch 00, Loss: 1.8398, Approx. Train: 0.4856
Epoch time: 75447.6528 ms
Epoch 01: 100%|| 1206179/1207179 [01:03<00:00, 19081.15it/s]
Epoch 01, Loss: 1.6118, Approx. Train: 0.5369
Epoch time: 63215.2019 ms
Epoch 02: 100%|| 1206179/1207179 [01:08<00:00, 17690.12it/s]
Epoch 02, Loss: 1.5761, Approx. Train: 0.5458
Epoch time: 68185.8287 ms
Epoch 03: 100%|| 1206179/1207179 [01:19<00:00, 15164.61it/s]
Epoch 03, Loss: 1.5608, Approx. Train: 0.5510
Epoch time: 79551.0037 ms
Epoch 04: 100%|| 1206179/1207179 [01:21<00:00, 14841.78it/s]
Epoch 04, Loss: 1.5515, Approx. Train: 0.5537
Epoch time: 81313.6292 ms
```

Thank you

Q&A