GPU Computing

Weijie Zhao

02/20/2024

HW2: ANN on DAG

For each test case, at least 50% queries should be correctly answered

Input:

V D E L K A B C M Q

K lines: X[i] (K<=V*D, 0<=X[i]<M)

E lines: u[j] v[j] (u[j] < v[j])

Q lines: start_point max_hop q_0 q_1 q_2 … q_{D-1}

For i = K to V*D-1: X[i] = (A * X[i - 1] + B * X[i - 2]+ C) % M (potential overflow)

Output:

Q lines: vertex id with the smallest L2 distance to q within max_hop from start_point. Ties are broken by ids.

V #vertices	V<=10^5	Q<=10^4	max_hop<=10
D #dimensions	D<=10^3	M<=10^2	
E #edges	E<=10^6	A, B, C non-negative 32-bit int	
L maximum out-degree	L<=63	Due: 2/26/2024 11:59pm EST	

Correct GPU Solution will get 5 pts bonus

Test Environment

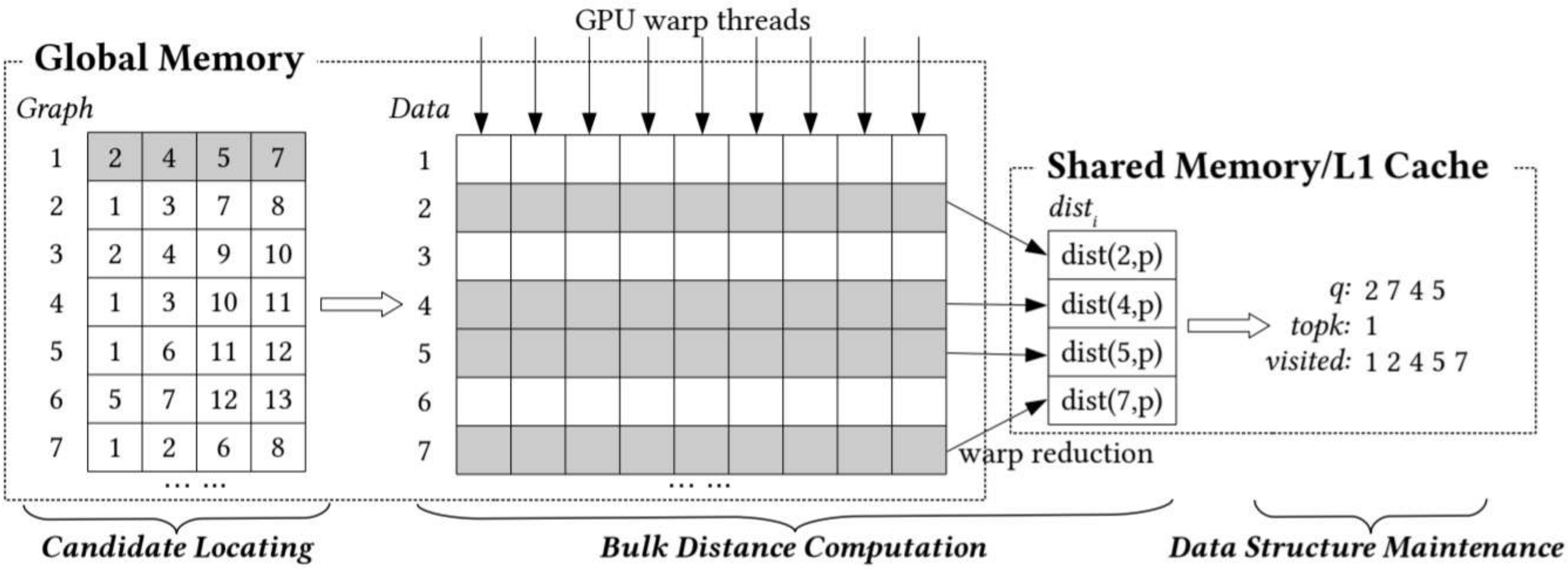
- granger.cs.rit.edu
- weasley.cs.rit.edu
- lovegood.cs.rit.edu

- 8 CPU threads and 1 GPU
- Time limit:
  - 120 seconds compilation time
  - 60 seconds for each test case

```
    edges = new int[V * (L + 1)];
    for(int i = 0;i < V;++i){
        edges[i * (L + 1)] = 0;
    }
    for(int i = 0;i < E;++i){
        int u,v;
        fscanf(fin,"%d%d",&u,&v);
        int degree = edges[u * (L + 1)];
        edges[u * (L + 1) + degree + 1] = v;
        ++edges[u * (L + 1)];
    }
```

```cpp
int nearest_id(int start_point,int max_hop,int* query_data){
    std::queue<std::pair<int,int>> q;
    q.push(std::make_pair(start_point,0));
    int min_d = std::numeric_limits<int>::max();
    int min_id = -1;
    while(!q.empty()){
        auto now = q.front();
        q.pop();
        int id = now.first;
        int hop = now.second;
        int d = squared_l2_dist(X + id * D,query_data,D);
        if((d < min_d) || (d == min_d && id < min_id)){
            min_d = d;
            min_id = id;
        }
        if(hop + 1 <= max_hop){
            int degree = edges[id * (L + 1)];
            for(int i = 1;i <= degree;++i){
                int v = edges[id * (L + 1) + i];
                q.push(std::make_pair(v,hop + 1));
            }
        }
    }
    return min_id;
```

GPU Graph Searching Example



Tensor Computing in Deep Learning

Weijie Zhao

02/20/2024

- Scalar
- Vector
- Matrix
- Tensor
  - Rank
  - Dimension

Tensor Computing in Deep Learning

Weijie Zhao

02/20/2024

- Scalar
- Vector
- Matrix
- Tensor
  - Rank
  - Dimension

Matrix

~~Tensor~~ Computing in Deep Learning

Weijie Zhao

02/20/2024

Matrix ~~Tensor~~ Computing in Deep Learning

- Scalar
- Vector
- Matrix
- Tensor
  - Rank
  - Dimension

Weijie Zhao

02/20/2024

- Matrix multiplication
- Non-linear activation
- Gradient descent

- Scalar
- Vector
- Matrix
- Tensor
  - Rank
  - Dimension

Matrix
~~Tensor~~ Computing in Deep Learning

Weijie Zhao

02/20/2024

- Matrix multiplication
- Non-linear activation
- ~~Gradient descent~~

Graduate student descent

Tensor Operations

- Element-wise add
- Element-wise plus
- Element-wise division
- Hadamard product
- Matrix multiplication
- Batched matrix multiplication
- More linear algebra operations…
- Collect, Scatter, Reduce…

Libraries

- Numpy
- Blas
- cuBlas
- cuSparse
- MKL
- TensorFlow
- PyTorch
- MXNet
- …

Lazy Evaluation and Code Generation

```
c = a + b
d = c * 2

for i = 1 to n do
        c[i] = a[i] + b[i]
for i = 1 to n do
        d[i] = c[i] * 2

for i = 1 to n do
        d[i] = (a[i] + b[i]) * 2
```