SHPC Final Project Report
2022-28981 김정수

1. 구조
    1) 노드별 일나누기: 우선 N개의 전체 단어들에 대해 각 노드가 N/4개의 단어를 만드는 일을 수행하게 한다.
    2) 노드 내 일을 배치단위로 진행:
    그리고 그 N/4개에 대해서 배치단위로 일을 진행한다.
    배치의 크기는 8192로 잡고 할당받은 N/4개의 단어를 배치크기 단위로 이제 또 일을 진행하게 된다.
    3) 배치 내:
    각 배치에 대해서 일을 진행할 때에는 SOS를 한번만 받는 게 아니라 배치사이즈만큼 [SOS SOS … SOS] 벡터를 만들어 그 벡터에 대해 GRU모델에 넣게 된다.
    그러면 각 output들이 벡터가 아닌 2차원 행렬이 되기 때문에 weight parameter와 곱할 때 matvec대신 matmul을 쓰게 된다.
    이때 matmul은 시간이 많이 소요되므로, matmul 함수를 gpu 커널함수로 올려 커널함수를 런칭하여 실행한다.
    cuda에서는 행렬곱을 실행할 때 tilesize만큼 타일링(strip mining)한 최적화된 행렬곱을 이용하였다.

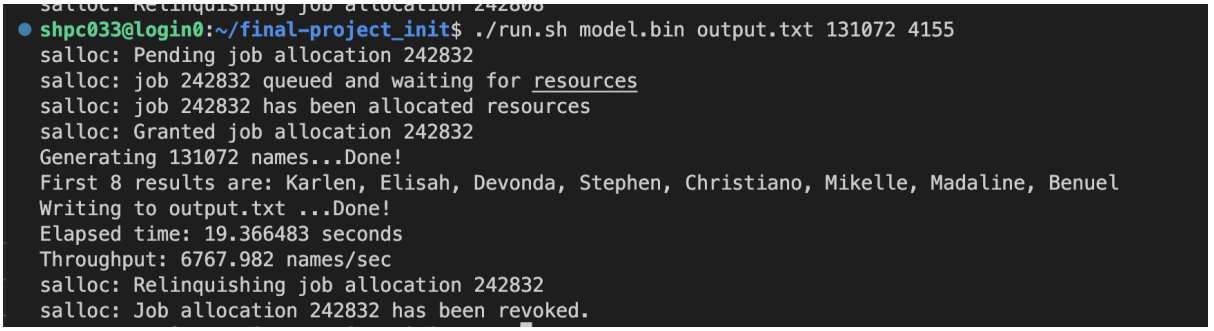
    또한 행렬곱 뿐만 아니라 elementwise add를 할 때 행렬+벡터 브로드캐스트 작업이 필요한데 이 부분은 matrow_add라는 함수를 따로 짜서 진행했다.

    random_select의 경우 배치단위로 진행하되, output과 rfloat에서 실제 절대적위치에 맞게 잘 찾아갈 수 있도록 index를 조정해주었다.

2. 최적화 요소들
    1) MPI로 노드 간 통신
    2) Openmp를 일부 sub함수에서 써서 cpu상의 스레드 간 병렬화를 추구하였다
    3) matmul kernel에 shared memory 차원의 tiling을 하여 행렬곱 최적화를 진행했다.

3. Library: Openmp, MPI, CUDA

4. 성능:



N=65536, Throughput: 5816.346 names/sec

5. 더 해볼만한 것들
    a. 현재 전반적인 데이터 접근 방식이 column wise인 듯 하다. row-wise로 바꾸면 성능증가가 이뤄질 것으로 보인다.
    b. 행렬곱 커널을 모든 행렬곱마다 만들어서 weight parameter를 보낼 때 정적할당으로 하면 더 성능이 좋아질 것이다.
    c. matmul시 두번째 인자를 transpose하면 캐시미스를 줄일 수 있을 것이다.
    d. 현재는 matmul 함수만 커널에 올렸지만 모든 sub 함수들(elemwise_add etc.)을 커널함수화 하면 더 좋을 수도 있을 것이다.