Project 5

Due: 12/16/2024 10:30 AM

서론

(1) 각 프로젝트의 파일은 Project <번호> -> Problem <번호> 폴더 구조로 구성한다. 예를 들어, Project 5->Problem 1 의 폴더에 Problem 1 에서 요구하는 모든 파일을 저장한다. 모든 파일은 Project 5 하에서 압축(zip)하고 압축 파일 이름은 학번으로 한다.

(2) 보고서가 필요한 경우는 pdf 형식으로 제출한다.

(3) 프로젝트 제출일의 수업시작(오전 10 시 30 분) 전까지 blackboard 를 통하여 제출한다.

■ **모든 프로젝트는 개인별 프로젝트입니다.**

문제 1(50 점): GPT-2 실행을 위한 환경설정

Project5_overview 자료의 [Environmental Setup]에 제시된 GPT-2 실행환경 구축을 진행한다. 전 과정은 다음과 같이 정리할 수 있다.

(1) Anaconda 설치 및 Python 가상환경 생성.
(2) HuggingFace 를 통한 GPT-2 모델 다운로드.
(3) GitHub 을 통한 PyTorch 소스코드 다운로드.
(4) GCC(GNU C Compiler)의 버전 업그레이드(≥9.3).
(5) PyTorch 빌드 및 설치.

실행환경 구축의 여부는 다음을 통해 평가한다.

- Anaconda 환경을 활성화한 상태에서, workspace directory 에서 "ls -al"을 입력하고, 빌드한 PyTorch 로 run.py 를 실행한 후, 터미널을 캡쳐한 이미지(problem1.jpg).
  - "ls -al"과 "python run.py" 두 command 에 대한 실행결과가 모두 드러나도록 캡쳐해야 한다.
  - run.py 는 생성한 anaconda 가상환경을 사용하여 실행해야 한다.
  - 본인이 지정한 workspace directory(ex. /home/compiler/work/project5)에는 GPT-2 모델과 PyTorch 소스코드, 그리고 이외에 프로젝트에서 제공한 코드가 들어있어야 한다.

문제 2(120점): GPT-2의 실행시간 Breakdown 및 Gemm 연산 최적화

이 프로젝트에서는 생성형 인공지능 모델의 실행시간에서 많은 부분을 차지하는 GeMM(General Matrix Multiplication) 연산을 최적화해보는 것을 목표로 한다. 이를 위해, 우선 최적화를 적용하기 이전의 GeMM 연산에 대한 baseline을 적용한다. 그리고, baseline 코드를 수업시간에 배운 내용을 참고하여 최적화한다. 또한, 최적화 전후에 대한 모델 전체 실행시간을 breakdown해본다. 문제 2의 진행은 Project5_overview 자료의 [Benchmark & Modification] 파트를 참고한다.

(1) (40 점) GeMM 연산의 baseline 코드를 적용하여 PyTorch 를 빌드하고, 이를 통해 GPT-2 모델을 실행하였을 때의 실행시간 breakdown.
- OpenMP thread 수를 1, 2, 4, 8 로 변화시키며 실행시간을 측정한다.
  - ➢ OpenMP 의 thread 수를 설정하는 command: "export OMP_NUM_THREADS=n"
- PyTorch profiler 를 사용하여, PyTorch 의 각 layer 별 실행시간을 구하고, 상위 10 개의 layer 실행시간을 표시하여 캡쳐한다.
- 실행시간 breakdown 결과는 보고서(project5.pdf)에 정리한다.

(2) (50 점) GeMM 연산의 baseline 코드에 적용할 최적화에 대한 자세한 설명과 이를 적용한 optimized 코드를 보고서(project5.pdf)에 제시.

(3) (30 점) Optimized GeMM 코드를 사용하여 PyTorch 를 빌드하고, (1)과 같이 GPT-2 모델을 실행하여 실행시간 breakdown 을 수행.
- (1)과 동일하게 진행한다.

문제 3(80점): Gemm 연산 최적화에 따른 실행성능 변화의 원인 분석

Project5_overview. 자료에 제시한 [Profiling Guide]를 참고하여, 문제 2에서 최적화한 GeMM 연산을 통해 실행성능이 변화한 원인을 분석해본다.

(1) (40 점) OpenMP thread 의 수를 1, 2, 4, 8 로 하고, GeMM 연산의 최적화 전후에 대하여 다음을 측정하여 보고서(project5.pdf)에 제시.

- Cycle 수, instruction 수, stall cycle 수, branch-hit/miss 수, cache-hit/miss 수.
  - Perf-stat 을 사용할 것.
- 각 메모리 계층구조 별 hit rate.
  - Perf-mem 을 사용할 것.

(2) (40 점) 문제 2 와 문제 3-(1)에서 측정한 성능 데이터를 기반으로, 자신이 적용한 코드 최적화가 실행 성능 변화에 어떤 영향을 미쳤는지 보고서(project5.pdf)에 서술.