과제 #06

확장형 고성능 컴퓨팅 (2021년도 2학기, M1522.006700, M3239.002300)

Due: 2021년 12월 5일(일) 23시 59분

1 Matrix Multiplication with CUDA (30점)

$A(M \times K) \times B(K \times N) = C(M \times N)$ 행렬 곱셈을 수행하는 예시 프로그램이 주어진다. 다음은 실행 예시이다.

```
$ make
$ make performance
(...)
Result: VALID
Reference time: 11.786589 sec
Reference throughput: 93.284969 GFLOPS
Your Avg. time: 129.885021 sec
Your Avg. throughput: 8.465269 GFLOPS
(...)
```

이번 과제의 목적은 행렬 곱셈 구현을 수정하여 성능을 높이는 것이다. 주의사항은 다음과 같다.

- `mat_mul.cu`에 행렬 곱셈을 구현하고, `Makefile`에서 컴파일 옵션을 수정할 수 있다. 그 외의 파일은 제출하지 않으며 뼈대 코드를 사용하여 채점된다.
- CUDA를 사용하여 병렬화 한다. 지금까지 배운 병렬화 방법(*e.g.*, OpenMP, MPI)를 섞어서 사용해도 좋다.
- 한 노드에 GPU가 4개 장착되어 있는데, 그 중 한개만 사용한다. 또한, 하나의 노드만 사용하도록 한다. 두 노드 이상 혹은 두개 이상의 GPU를 사용하도록 코드를 수정하여 제출하는 경우 감점될 수 있다.
- `mat_mul_init` 함수에서는 계산에 앞서 하고 싶은 초기화(*e.g.*, 메모리 할당)를 할 수 있다. 이 부분은 실행 시간에 포함되지 않는다.
- 계산량을 줄이는 알고리즘(*e.g.*, Strassen, Winograd)을 사용하는 것은 허용되지 않는다. 그 외의 최적화는 대부분 허용되나, 애매할 경우 조교에게 문의하면 된다.
- 사용하는 노드 개수 이외에 `Makefile`의 옵션은 자유롭게 변경가능하다. 채점은 제출한 Makefile 옵션을 기준으로 진행된다.
- 실습서버의 로그인 노드에 접속 후 소스 코드 디렉토리로 이동. 그 후, `make performance`를 입력하거나 `Makefile`의 내용을 참고하여 SLURM job scheduler에 작업을 enqueue 한다.

(a) (5점) 어떻게 병렬화 하였는지 설명하시오.

(b) (5점) 자유롭게 결과를 분석하시오. (OpenCL과 성능 차이가 있는가? 등)

(c) (10점) 다양한 M(128의 배수), N(128의 배수), K(128의 배수)에 대해 정확한 답이 나와야 한다. `-v` 옵션으로 답이 맞는지 확인할 수 있으며, `make validation`로 몇가지 케이스를 테스트할 수 있다. 채점시에는 다른 케이스를 사용한다.

(d) (10점) $M = N = K = 8192$ 기준으로 128 GFLOPS 이상은 10점, 256 GFLOPS 이상은 20점을 부여한다. `make performance`로 제출 전 성능을 테스트 해볼 수 있다.

2 Double buffering (20점)

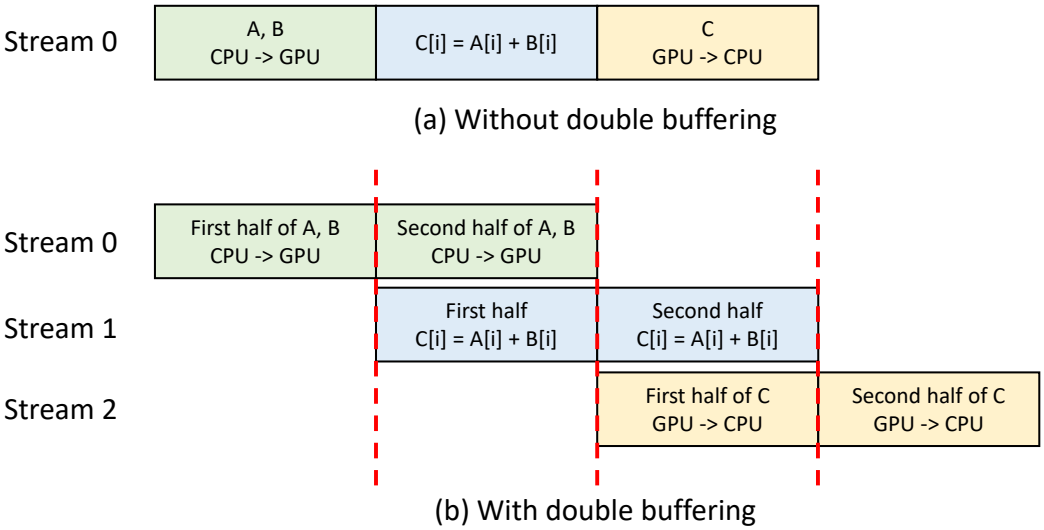

Figure 1: 더블 버퍼링을 적용한 경우와 그렇지 않은 경우의 타임라인

길이 N인 벡터 2개를 더하는 vector addition 프로그램을 작성하고, 그림 1을 참고해서 더블 버퍼링을 적용해보자. 더블 버퍼링을 적용한 경우와 적용하지 않은 경우의 차이를 위주로 결과를 분석하시오.

3 Using Profiler (40점)

본인이 만든 프로그램의 병목을 파악하고 개선할 방법을 찾는데는 프로파일러가 유용하다. NVIDIA가 제공하는 프로파일러로 CUDA 프로그램을 프로파일 해보고 프로젝트에 활용할 수 있도록 연습한다.

3.1 Prerequisite

프로파일러는 여러가지 방식으로 사용할 수 있다.

- 프로파일러와 프로그램을 같은 컴퓨터에서 동시에 실행
- 프로파일러와 프로그램을 다른 컴퓨터에서 동시에 실행
- CLI 버전의 프로파일러와 프로그램을 실행해서 결과를 파일로 추출하고, GUI 버전의 프로파일러에서 해당 파일을 불러와서 결과 분석

일반적으로는 첫 두가지 방법이 편리하지만, 로그인 노드와 계산 노드로 이루어진 클러스터 환경에서는 프로파일러와 프로그램을 연결해서 동시에 실행하기에 번거로운 점이 많기 때문에 마지막 방법을 많이 사용한다. 마지막 방법을 사용하려면 GUI 버전의 프로파일러를 로컬 컴퓨터에 설치해야 한다. (서버에는 이미 설치되어 있다.) 그러므로 다음 두 프로그램을 설치하자.

- NVIDIA Nsight Systems - Download - 회원가입 및 로그인 - OS에 맞는 최신 Host 버전 다운로드 (현재 2021.5.1)
- NVIDIA Nsight Compute - Download - 회원가입 및 로그인 - OS에 맞는 최신 Host 버전 다운로드 (현재 2021.3.0)

3.2 NVIDIA Nsight Systems

Nsight Systems는 프로그램이 실행되면서 호출된 CUDA 함수에 관한 간단한 정보들을 타임라인과 함께 보여주는 프로파일러이다. 프로그램의 전체적인 흐름을 파악할때 유용하다. 기본적인 Nsight Systems의 사용법은 다음과 같다.

```
shpc000@login:~$ /usr/local/cuda/bin/nsys profile <실행 프로그램>
# .bashrc 에 "export PATH=/usr/local/cuda/bin:$PATH" 를 추가하면 매번 전체 경로를 입력할 필요 없이 nsys만 입력하여 사용 가능

shpc000@login:~$ ls
...  report1.qdrep  ...
# qdrep 확장자로 프로파일링 결과 파일이 생성됨

# 해당 qdrep 파일을 로컬로 복사 (scp, sftp 등 사용)
# 로컬에서 NVIDIA Nsight Systems를 실행하고 File > Open > qdrep 파일 선택
```

이제 본인의 1번 문제 코드인 matrix multiplication 코드를 가져와서 실습해보자. 아래는 실습서버에서 Nsight Systems로 matrix multiplication 코드를 프로파일링하는 SLURM 명령어이다.

```
shpc000@login:~/hw6$ salloc --nodes=1 --ntasks-per-node=1 --cpus-per-task=32 --gres=gpu:1 \
                     --partition=shpc mpirun /usr/local/cuda/bin/nsys profile \
                     ./main -n 1 8192 8192 8192

shpc000@login:~/hw6$ ls
...  report1.qdrep  ...
```

(a) (10점) "Timeline View"에서 좌측 "CUDA API"를 우클릭 후 "Show in Events View"를 누르면 하단에서 실행된 CUDA 런타임 함수 및 커널의 시간 정보를 확인할 수 있다. 본인이 작성한 행렬곱 커널은 몇초에 시작하여 몇초동안 실행되었는지 적으시오.

(b) (10점) 타임라인을 캡처하여 보고서에 첨부하시오. (실제로 했는지 확인 용도이므로, 일부만 나와도 됨)

자세한 사용법은 User Guide를 참고하자.

3.3 NVIDIA Nsight Compute

Nsight Systems가 프로그램의 전체적인 흐름을 보여준다면, Nsight Compute는 커널 하나하나의 자세한 수치를 보여주는 프로파일러이다. 기본적인 Nsight Compute의 사용법은 다음과 같다.

```
shpc000@login:~$ /usr/local/cuda/bin/ncu -o profile <실행 프로그램>
# .bashrc 에 "export PATH=/usr/local/cuda/bin:$PATH" 를 추가하면 매번 전체 경로를 입력할 필요 없이 ncu만 입력하여 사용 가능

shpc000@login:~$ ls
...  profile.ncu-rep  ...
# ncu-rep 확장자로 프로파일링 결과 파일이 생성됨

# 해당 ncu-rep 파일을 로컬로 복사 (scp, sftp 등 사용)
# 로컬에서 NVIDIA Nsight Compute를 실행하고 File > Open File > ncu-rep 파일 선택
```

이제 본인의 1번 문제 코드인 matrix multiplication 코드를 가져와서 실습해보자. 아래는 실습서버에서 Nsight Compute로 matrix multiplication 코드를 프로파일링하는 SLURM 명령어이다.

```
shpc000@login:~/hw6$ salloc --nodes=1 --ntasks-per-node=1 --cpus-per-task=32 --gres=gpu:1 \
                     --partition=shpc mpirun /usr/local/cuda/bin/ncu -o profile \
                     ./main -n 1 8192 8192 8192

shpc000@login:~/hw6$ ls
...  profile.ncu-rep  ...
```

(a) (10점) "Page:" 옆의 드롭다운 메뉴를 "Details"로 설정하면 실행된 커널의 자세한 정보와 추천하는 개선 방법이 나온다. 그 중 GPU Speed Of Light는 GPU 내부의 여러가지 하드웨어들이 이론상 최대 성능 대비 얼마나 활용되었는가를 보여주는 항목이다. 본인이 작성한 행렬곱 커널이 계산 자원을 얼마나 잘 활용하였는지 (SOL SM %), 메모리 자원을 얼마나 잘 활용하였는지 (SOL Memory %) 적으시오.

(b) (10점) Details 화면을 캡처하여 보고서에 첨부하시오. (실제로 했는지 확인 용도이므로, 일부만 나와도 됨)

서버에서 사용하는 ncu 프로그램의 자세한 사용법은 CLI User Manual을, 프로파일링 결과 해석의 자세한 방법은 UI User Manual을 참고하자.

4 소감 (10점)

(a) (5점) 대표적인 GPGPU 프레임워크 OpenCL과 CUDA를 써보았다. 비교해서 사용 소감을 간단하게 작성하시오.

(b) (5점) 과제를 하면서 개선됐으면 싶었던 것들을 간단하게 작성하시오. 없으면 "없다"고 적거나 아무 말이나 적는다.

5 Submission Instruction

- `mat_mul` 디렉토리, `double` 디렉토리, `report.pdf`를 한 파일로 압축하여 ETL에 제출한다.
- `mat_mul` 디렉토리에는 `mat_mul.cu`, `Makefile`를 넣는다. 다른 파일은 있더라도 제외하고 채점된다.
- `double` 디렉토리에는 구현한 파일들을 모두 넣는다. 실제로 실험을 했는지 확인하기 위한 용도이며, 조교가 실행해 보지는 않을 것이다.
- 첨부 파일명은 계정이름_`HW06.zip`으로 한다. (*e.g.*, `shpc000_HW06.zip`)
- 채점은 프로그램에 의해 기계적으로 처리되므로 위 사항을 지키지 않은 경우 누락되거나 불이익을 받을 수 있다.
- Grace day를 사용하려면 본인이 과제를 제출한 날에 조교에게 메일(`shpc21@aces.snu.ac.kr`)로 알려야 한다. 메일 없이 제출만 한 경우 다음 과제를 위해 아낀 것으로 판단, 미제출 처리된다. 또한, grace day 사용 시에도 과제 제출은 이메일이 아닌 ETL을 통해 해야한다.