과제 #4

M1522.006700 확장형 고성능 컴퓨팅 (001)
M3239.005400 데이터사이언스를 위한 컴퓨팅 2 (001)

Due: 2024년 11일 17일 (일) 23:59:59

1 (20점) GPU 정보 확인하기

보고서에 다음 질문들에 대한 답을 서술하라.

(a) (4점) 다음 커맨드들의 의미와 실행 결과를 답하라. 결과가 너무 길면 적당히 앞부분만 잘라서 첨부하라.

```
srun --partition=class1 --gres=gpu:4 nvidia-smi
```

```
srun --partition=class1 --gres=gpu:4 nvidia-smi -q
```

```
srun --partition=class1 --gres=gpu:4 clinfo
```

(b) (4점) 실습 서버의 **계산 노드**에 장착된 GPU의 모델명과 노드당 장착된 GPU의 개수를 답하라.

(c) (4점) 실습 서버의 **계산 노드**에 장착된 GPU 하나의 메모리 크기를 MiB 단위로 답하라. (MiB: 2^{20} bytes, MB: 10^6 bytes)

(d) (4점) 실습 서버의 **계산 노드**에 장착된 GPU의 maximum power limit(W)과 maximum SM clock speed (Mhz)를 답하라.

(e) (4점) 실습 서버의 **계산 노드**에 장착된 GPU를 OpenCL 을 이용해 사용할 때 Max work item dimension, Max work item size, Max work group size 를 답하라.

2 (80점) Matrix Multiplication with OpenCL

두 FP32 행렬의 곱을 하나의 GPU를 이용해 계산하는 프로그램 `matmul.c, kernel.cl`을 작성하라. 뼈대 코드와 Makefile 이 실습 서버 로그인 노드의 `/shpc/skeleton/hw4/matmul` 디렉토리에 제공된다. 뼈대 코드를 이해한 뒤, `matmul.c` 및 `kernel.cl` 파일을 작성하면 된다. 이외의 파일은 수정 불가능하다 (채점 시 `matmul.c` 및 `kernel.cl` 파일을 제외한 파일들은 뼈대코드의 것을 사용함).

`matmul_initialize` 함수는 이미 대부분 구현되어 있다. 이를 참고해 `matmul` 함수와 `matmul_finalize` 함수를 작성하라.

Slurm 작업 스케줄러에 작업을 제출하는 `./run.sh` 스크립트가 제공된다. 실행 예시는 다음과 같다.

```
$ ./run.sh -v -n 3 1024 1024 1024
```

```
Options:
  Problem size: M = 1024, N = 1024, K = 1024
  Number of iterations: 3
  Print matrix: off
  Validation: on

Initializing... done!
Initializing OpenCL...
Detected OpenCL platform: NVIDIA CUDA
Detected OpenCL device: NVIDIA TITAN RTX
Calculating...(iter=0) 0.005572 sec
Calculating...(iter=1) 0.005567 sec
Calculating...(iter=2) 0.005590 sec
Validating...
Result: VALID
Avg. time: 0.005576 sec
Avg. throughput: 385.106388 GFLOPS
```

Hint: `/shpc/skeleton/vectorio_opencl` 디렉토리에 OpenCL 을 사용해 벡터 내적을 계산하는 프로그램이 있다. OpenCL API 사용법을 참고할 것. 행렬 곱 최적화에 Vector IO 를 사용하라는 것이 아님.

보고서에 다음 질문들에 대한 답을 서술하라

- 자신의 병렬화 방식에 대한 설명.
- 뼈대 코드 `matmul.c`의 각 부분에 대한 설명. `matmul_initialize`, `matmul`, `matmul_finalize` 함수 각각에서 사용하는 OpenCL API 및 각 API에 대한 간략한 설명. (API 당 한문장이면 충분).
- 자신이 적용한 코드 최적화 방식을 분류하고, 각각에 대한 성능 실험 결과. (Matrix multiplication은 향후 프로젝트에 핵심적으로 사용될 예정이므로 해당 실험을 적극적으로 해보길 권장함.)

본 문제는 정확성 및 성능 평가를 한다. 채점 기준은 다음과 같다.

보고서 (10점) 명시된 질문들에 대한 답으로 평가.

정확성 (30점) 4096 이하의 M, N, K에 대해서 `-v` 옵션을 통한 validation을 통과해야 한다. 제공된 `./run_validation.sh` 스크립트를 이용해 채점할 예정임.

성능 (40점) 실습 서버에서 $M = N = K = 4096$ 옵션을 주고 실행했을 때, 1200 GFLOPS를 넘으면 만점. 그 이하는 비율에 따라 점수를 부여한다 (e.g., 1080 GFLOPS인 경우 성능 점수의 90%를 부여). 답이 틀린 경우 0점. 제공된 `./run_performance.sh` 스크립트를 이용해 채점할 예정임. 1개보다 많은 GPU 또는 CPU를 이용해 행렬곱을 계산할 경우 0점.

3 제출 방법

- 과제 제출은 실습 서버에서 이루어진다.

- 보고서는 pdf 형식으로 만들어 `report.pdf` 이름으로 제출한다. 제출할 `report.pdf` 파일이 위치한 디렉토리에서 `shpc-submit submit hw4 report.pdf` 명령을 실행한다.
- 제출할 `matmul.c`, `kernel.cl` 파일이 위치한 디렉토리에서 `shpc-submit submit hw4 matmul.c` 및 `shpc-submit submit hw4 kernel.cl` 명령을 실행한다.
- 파일들이 잘 제출되었는지 확인을 위해 `shpc-submit status` 명령을 실행한다.
- 과제 마감 기한이 지난 뒤 다시 제출 명령을 실행하면 마지막 제출시간이 변경되므로 주의할 것.
- 과제 마감 기한이 지난 뒤 파일이 수정된 경우 grace day 를 사용한 것으로 간주한다.

4 주의 사항

- 뼈대 코드를 각자의 홈 디렉토리로 복사해 가 작업하도록 한다.
- 실습용 서버에서 과제를 수행하도록 한다. 소스 코드를 제출하는 과제의 경우 실습용 서버에서 작동하지 않으면 점수를 받을 수 없다.
- 보고서는 간략하게 필요한 내용만 적는다.