과제 #6

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

Due: 2024년 12월 8일(일) 23:59:59

1 (100점) Matrix Multiplication Challenge

두 FP32 행렬의 곱을 4개 노드의 모든 자원을 자유롭게 이용해 계산하는 프로그램 `matmul.cu` 및 실행 파일 `run.sh` 를 작성하라. 본 과제에서는 OpenCL 사용을 불허한다. Pthread, OpenMP, MPI 및 CUDA를 사용해 구현하면 된다.

뼈대 코드와 Makefile 이 실습 서버 로그인 노드의 `/shpc/skeleton/hw6/matmul` 디렉토리에 제공된다. 뼈대 코드에는 단일 노드에서 4개의 GPU를 이용하는 프로그램이 구현되어 있다. 뼈대 코드를 이해한 뒤, `matmul.cu` 및 `run.sh` 파일을 작성하면 된다. 이외의 파일은 수정 불가능하다 (채점 시 `matmul.cu, run.sh` 파일을 제외한 파일들은 뼈대코드의 것을 사용함).

한 프로세스에서 여러 GPU 를 사용하기 위해 `cudaSetDevice()` API 를 호출하는 부분을 잘 이해해야 한다. `cudaSetDevice(i)` 호출 이후의 모든 CUDA API 들은 해당 노드 내 `i` 번 GPU에게 명령을 내리는 것임을 이해하자. 한 프로세스가 여러 GPU를 관리하는 경우, 동시에 여러 GPU에게 명령을 내리기 위해 CUDA API 호출을 비동기로 (Asynchronous API) 해야 함에 유의하자.

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

```
$ ./run.sh 4096 4096 4096 -n 5 -v
(a01) Hello world, rank 0 out of 2
(a02) Hello world, rank 1 out of 2
Options:
  Problem size: M = 4096, N = 4096, K = 4096
  Number of iterations: 5
  Print matrix: off
  Validation: on

[rank 0] Initializing matrices...Done!
[rank 0] Number of devices: 4
[rank 1] Number of devices: 4
[rank 0] device 0: NVIDIA TITAN RTX
[rank 1] device 0: NVIDIA TITAN RTX
```

```
[rank 1] device 1: NVIDIA TITAN RTX
[rank 0] device 1: NVIDIA TITAN RTX
[rank 1] device 2: NVIDIA TITAN RTX
[rank 0] device 2: NVIDIA TITAN RTX
[rank 1] device 3: NVIDIA TITAN RTX
[rank 0] device 3: NVIDIA TITAN RTX
[rank 0] Calculating...(iter=0) 0.049669 sec
[rank 0] Calculating...(iter=1) 0.047845 sec
[rank 0] Calculating...(iter=2) 0.047762 sec
[rank 0] Calculating...(iter=3) 0.047841 sec
[rank 0] Calculating...(iter=4) 0.047964 sec
Validating...
Result: VALID
[rank 0] Avg. time: 0.048216 sec
[rank 0] Avg. throughput: 2850.469099 GFLOPS
```

최대 4대의 노드를 사용할 수 있으며, `run.sh`의 $NODES$ 환경변수 값을 수정해 실행할 노드 수를 설정할 수 있다. 각 노드는 2대의 CPU (socket 기준) 와 4대의 GPU로 구성되어 있다. 따라서 총 8대의 CPU와 16대의 GPU를 사용할 수 있다.

모든 계산 자원을 사용해야한다는 제약사항은 없다. 전체 CPU 및 GPU 중 일부만 사용하거나 전혀 사용하지 않더라도 특별한 감점이나 불이익을 없다는 점에 유의하자.

`run.sh` 는 노드당 1개의 MPI 프로세스를 생성하도록 되어 있다. 만약 노드당 여러 개의 MPI 프로세스 (예를 들어, GPU당 1개의 프로세스)를 생성하고 싶으면 `run.sh` 의 `-npernode` 옵션을 수정하면 된다.

보고서에 다음 질문들에 대한 답을 서술하라

- 자신의 병렬화 방식에 대한 설명.
- 성능 최적화를 위한 적용한 방법 및 고려 사항들에 대한 논의.
- `matmul.c`의 각 부분에 대한 설명. `matmul_initialize`, `matmul`, `matmul_finalize` 함수 각각에서 사용하는 CUDA API 및 각 API에 대한 간략한 설명. (API 당 한문장이면 충분).
- 자신이 적용한 최적화 방식을 정리하고, 각각에 대한 성능 실험 결과. (Matrix multiplication은 프로젝트에도 핵심적인 부분이므로 해당 실험을 적극적으로 해보길 권장함.)

본 문제는 성능 평가만 한다. 채점 기준은 다음과 같다.

보고서 (10점) 명시된 질문들에 대한 답으로 평가.

성능 (90점) 실습 서버에서 `run.sh`를 $M = 65536, N = K = 4096$ 옵션을 주고 실행했을 때, 20,000 GFLOPS를 넘으면 만점. 그 이하는 비율에 따라 점수를 부여한다 (e.g., 18,000 GFLOPS인 경우 성능 점수의 90%를 부여). 답이 틀린 경우 0점.

2 제출 방법

- 과제 제출은 실습 서버에서 이루어진다.
- 보고서는 pdf 형식으로 만들어 `report.pdf` 이름으로 제출한다. 제출할 `report.pdf` 파일이 위치한 디렉토리에서 `shpc-submit submit hw6 report.pdf` 명령을 실행한다.
- 제출할 `matmul.cu` 파일이 위치한 디렉토리에서 `shpc-submit submit hw6 matmul.cu` 명령을 실행한다.
- 제출할 `run.sh` 파일이 위치한 디렉토리에서 `shpc-submit submit hw6 run.sh` 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.

3 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.