과제 #5

4190.414A 멀티코어 컴퓨팅(001)

Due: 2020년 6월 2일(금) 23시 59분

1 문제1: 한 프로세스에서 Multi GPU 사용하기 (50%)

과제3 에서 GPU 1개를 사용하여 행렬 곱셈을 수행하는 프로그램을 작성하였다. GPU를 최대 4개까지 사용할 수 있도록 프로그램을 수정하라. 뼈대 코드는 과제3의 것을 사용한다. 주의 사항은 다음과 같다.

- `./run.sh` 의 첫 부분인 `SBATCH` 옵션을 바꾸어 GPU 개수를 설정할 수 있다. 주석처럼 보이지만 `sbatch` 커맨드에서 사용하는 옵션들이다.
- 실습 서버 계산 노드 1개에 탑재된 NVIDIA GeForce GTX 2080 GPU 4개를 사용한다. `run.sh` 의 `#SBATCH --gpus=1` 의 값을 바꾸어 여러 GPU를 사용하도록 한다. 채점 시에는 `SBATCH --gpus=4` 옵션을 줄 것이다.
- **메모리 전송을** `mat_mul` **함수 내에서 수행하여야 한다. 즉,** `mat_mul_init` **또는** `mat_mul_final` **함수 내에** `clEnqueueWriteBuffer` **또는** `clEnqueueReadBuffer`**가 있다면** `mat_mul`**로 옮겨주어야 한다. 이는 메모리 전송 시간을 포함하여 성능을 측정하기 위함이다.**
- 실험을 위해 1, 2, 3, 4개의 GPU를 사용할 수 있도록 구현하되, 제출하는 프로그램은 GPU 4개를 사용해야만 한다.
- 여러 스레드를 사용하는 것은 허용한다. OpenMP 의 사용도 허용한다. 단, CPU에서 행렬 곱의 일부를 수행하는 것은 금지한다.
- 과제3 에서는 Makefile 의 수정을 금지했지만, 이번 과제에서는 Makefile 의 수정이 가능하다. `mat_mul.c`, `kernel.cl`, `Makefile`만 수정할 수 있다. 다른 파일은 채점 시에 예시 코드로 덮어씌워진다. 제출 시 `mat_mul.c`, `kernel.cl`, `Makefile` 를 제출하도록 한다. **이 세 파일을** `mat_mul_A` **디렉토리에 넣어서 제출하도록 한다.**
- 이 외의 주의사항은 지난 과제와 같다.

한 OpenCL 프로세스에서 여러 GPU 를 사용하기 위한 힌트로, 다음의 코드를 참고하라.

```
cl_device_id devices[4];
int num_dev = 0;
err = clGetDeviceIDs(platform, CL_DEVICE_TYPE_GPU, 4, devices, &num_dev);
CHECK_ERROR(err);
for (int i = 0; i < num_dev; i++) {
  printf("Device %d: ", i);
  print_device_info(devices[i]);
}
```

보고서에는 다음 내용이 들어가면 좋다.

- 병렬화 방법에 대한 설명
- GPU 1개 사용 대비 2, 3, 4개를 사용할 때의 Speedup

채점 기준은 다음과 같다.

보고서 (20%)

정확성 (40%) 4096 이하의 임의의 M, N, K에 대해서 `-v` 옵션을 통한 validation을 통과해야 한다.

성능 (40%) $M = N = K = 8192$ 옵션을 주고 실행했을 때, 1200 GFLOPS를 넘으면 만점. 그 이하는 비율에 따라 점수를 부여한다. 답이 틀린 경우 0점.

2 문제 2: MPI로 다수의 계산 노드 사용하기 (50%)

행렬 곱셈을 수행하는 MPI 예시 프로그램이 주어진다. 다음은 실행 예시이다.

```
$ sbatch run.sh 1024 1024 1024
(...)
[rank 0] Avg. time: 0.021929 sec
[rank 0] Avg. throughput: 97.928729 GFLOPS
(...)
```

프로그램을 수정하여 성능을 높여보자. 주의 사항은 다음과 같다.

- `./run.sh` 의 첫 부분인 `SBATCH` 옵션들을 바꾸어 노드 개수와, 노드당 GPU 개수 등을 설정할 수 있다. 주석처럼 보이지만 `sbatch` 커맨드에서 사용하는 옵션들이다.
- 최대 계산 노드 3개(i.e., `./run.sh` 의 `#SBATCH --nodes` 옵션 값으로 3까지 가능), NVIDIA GeForce GTX 2080 GPU 12개를 사용할 수 있다.
- 생성하는 프로세스의 수는 자유이다. 노드마다 프로세스를 1개씩 생성하여 각 프로세스가 GPU를 4개씩 사용해도 되고, 노드마다 프로세스를 4개씩 생성하여 각 프로세스가 GPU를 1개씩 사용해도 된다 (`run.sh`의 `#SBATCH --tasks-per-node` 옵션 값을 조절하면 된다.). **제출하는 `./run.sh` 스크립트를 `sbatch` 명령어를 통해 채점할 예정이므로 옵션들을 본인의 프로그램에 맞게 기본값을 잘 설정해주어야 한다.**
- 행렬 A, B는 rank가 0인 프로세스에만 주어지므로 MPI를 통해 다른 프로세스에 전달해주어야 한다. 또한, 결과 검증도 rank가 0인 프로세스에서 진행하므로 계산 후 행렬 C를 rank가 0인 프로세스에 모아주어야 한다.
- MPI, OpenCL의 메모리 관련 API 호출은 `mat_mul` 함수 내에서 수행하여야 한다. 즉, `mat_mul_init` 또는 `mat_mul_final` 함수 내에 `clEnqueueWriteBuffer`, `MPI_Send` 등이 있다면 `mat_mul`로 옮겨주어야 한다. 이는 메모리 전송 시간을 포함하여 성능을 측정하기 위함이다.
- 성능 측정에 적합한 버전으로 제출하도록 하자. (예를 들어, 프로그램을 수정하다가 GPU 1개 사용 버전을 제출하지 않도록 주의. 특히 `run.sh` 에서 지정하는 옵션들에 주의.)
- 버퍼에 의해 출력이 뒤섞이므로 디버깅 시에 주의하자.
- 문제1과 마찬가지로, 여러 스레드를 사용하는 것은 허용한다. 단, CPU에서 행렬 곱의 일부를 수행하는 것은 금지한다.

- `mat_mul.c`, `kernel.cl`, `Makefile`, `run.sh` 만 수정할 수 있다. 다른 파일은 채점 시에 예시 코드로 덮어씌워진다. **이 네 파일을 `mat_mul_B` 디렉토리에 넣어서 제출하도록 한다.**

보고서에는 다음 내용이 들어가면 좋다.

- 병렬화 방법에 대한 설명
- 사용하는 GPU 개수에 따른 Speedup
- 노드 개수에 따른 성능 Speedup (e.g., 노드 1개의 GPU 4개, 노드 2개에서 GPU 2개씩, 노드 3개에서 GPU 2개씩 등등)

채점 기준은 다음과 같다.

보고서 (20%)

정확성 (40%) 128 이상 4096 이하의 임의의 M, N, K에 대해서 `-v` 옵션을 통한 validation을 통과해야 한다.

성능 (40%) $M = N = K = 16384$ 옵션을 주고 실행했을 때, 2400 GFLOPS를 넘으면 만점. 그 이하는 비율에 따라 점수를 부여한다. 답이 틀린 경우 0점.

3 제출 방법

- 조교 메일(`jinpyo@aces.snu.ac.kr`)로 보고서를 포함한 모든 파일을 하나의 파일(e.g., .zip, .tar.gz)로 압축 후 첨부하여 제출한다.
- 메일 제목은 [mc2021] 계정이름_HW5으로 한다. (e.g., mc99_HW5)
- 첨부파일명은 [mc2021] 계정이름_HW5.확장자으로 한다. (e.g., mc99_HW5.zip, mc99_HW5.tar.gz)
- **첨부파일의 압축을 풀면 각 문제의 제출 파일들을 담은 `mat_mul_A` 와 `mat_mul_B` 디렉토리가 있어야 한다. 보고서의 위치는 어디든 관계 없다.**
- 제출한 메일은 기계적으로 처리되므로 위의 내용을 지키지 않을 시 누락될 수 있으니 잘 지켜주시기 바랍니다.
- Grace day를 사용하고자 하는 경우에는 메일 내용에 이를 반드시 포함한다.