TA Session 5

조교 김진표, 박대영, 신준식, 정재훈

pp-ta@aces.snu.ac.kr

THUNDER Research Group
Seoul National University
서울대학교 천둥 연구실

실습 및 시험 일정

- 시험은 실습 과제와 비슷한 형식으로 진행할 예정
  - 배운 병렬화 방식 모두 사용 가능
  - 5/9 (월) 공개, 5/18 (수) 23:59 마감
  - 별도의 필기시험은 없음
  - 실습 시간에 matmul 과제와 함께 Q&A 진행

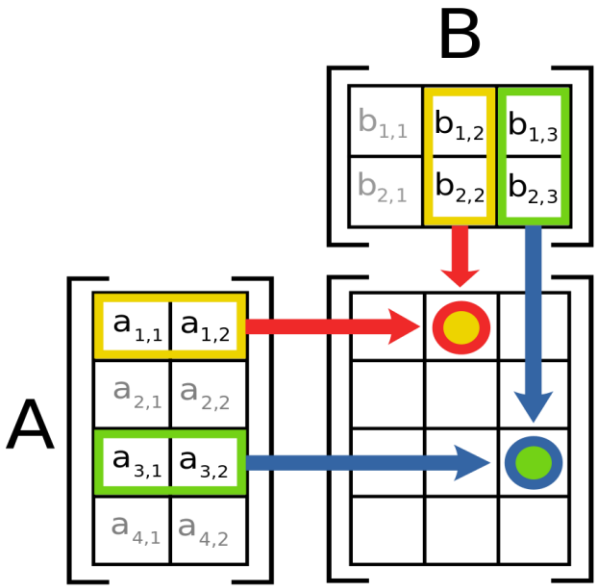
- 앞으로의 일정
  - OpenMP + MPI matmul (5/9 14:29 마감)
  - 5/4 (수) - OpenCL matmul
  - 5/9 (월) - OpenCL matmul (5/13 14:29 마감)
  - 5/11 (수) - CUDA matmul
  - 5/13 (금) - CUDA matmul (5/18 23:59 마감)

구성

- 과제 5 OpenCL matmul 소개
  - 다음주 월요일 실습까지 진행
  - 채점을 하는 과제임
- 과제 5 진행

과제5: OpenCL matmul

- FP32 행렬 둘을 곱하는 프로그램을 OpenCL 를 이용해 최적화
  - 1개 노드의 4개 GPU 사용
  - NVIDIA Geforce RTX 3090
- 과제5 는 채점을 함



https://en.wikipedia.org/wiki/Matrix_multiplication#/media/File:Matrix_multiplication_diagram_2.svg

OpenCL matmul

- /skeleton/HW5 에 주어진 뼈대 코드를 각자의 홈 디렉토리로 복사해 작업
  - `cp -r /skeleton/HW5 ~/`
  - `Makefile, run.sh, …`
  - `make` 로 컴파일이 가능하도록 `Makefile` 을 제공
  - 실행을 위한 `run.sh` 스크립트를 제공

- 뼈대 코드를 수정해 행렬 곱셈을 최적화하는 것이 목표
  - `mat_mul.cpp, kernel.cl` 만 수정 가능

OpenCL matmul - 실행

- make 로 컴파일
- `run.sh, run_performance.sh, run_validate.sh` 으로 실행

- -t 옵션은 더이상 사용하지 않음

OpenCL matmul

```
void mat_mul_init(float *A, float *B, float *C, int M, int N, int K) {
  // Get OpenCL platform
  err = clGetPlatformIDs(1, &platform, NULL);
  CHECK_ERROR(err);
  print_platform_info(platform);

  // Get OpenCL device
  err = clGetDeviceIDs(platform, CL_DEVICE_TYPE_GPU, 1, &device, NULL);
  CHECK_ERROR(err);
  print_device_info(device);

  // Create OpenCL context
  context = clCreateContext(NULL, 1, &device, NULL, NULL, &err);
  CHECK_ERROR(err);

  // Create OpenCL command queue
  queue = clCreateCommandQueue(context, device, 0, &err);
  CHECK_ERROR(err);

  // Compile program from "kernel.cl"
  program = create_and_build_program_with_source(context, device, "kernel.cl");
```

OpenCL matmul

```
  // Extract kernel from compiled program
  kernel = clCreateKernel(program, "sgemm", &err);
  CHECK_ERROR(err);

  // Create GPU buffers
  a_d = clCreateBuffer(context, CL_MEM_READ_WRITE, M * K * sizeof(float), NULL, &err);
  CHECK_ERROR(err);
  b_d = clCreateBuffer(context, CL_MEM_READ_WRITE, K * N * sizeof(float), NULL, &err);
  CHECK_ERROR(err);
  c_d = clCreateBuffer(context, CL_MEM_READ_WRITE, M * N * sizeof(float), NULL, &err);
  CHECK_ERROR(err);

  // Write to GPU; A (cpu) -> a_d (gpu), B (cpu) -> b_d (gpu)
  err = clEnqueueWriteBuffer(queue, a_d, CL_TRUE, 0, M * K * sizeof(float), A, 0, NULL, NULL);
  CHECK_ERROR(err);
  err = clEnqueueWriteBuffer(queue, b_d, CL_TRUE, 0, K * N * sizeof(float), B, 0, NULL, NULL);
  CHECK_ERROR(err);

  // DO NOT REMOVE; NEEDED FOR TIME MEASURE
  err = clFinish(queue);
  CHECK_ERROR(err);
}
```

OpenCL matmul

```
void mat_mul(float *_A, float *_B, float *_C, int _M, int _N, int _K) {
  A = _A, B = _B, C = _C;
  M = _M, N = _N, K = _K;

  // Setup kernel arguments
  err = clSetKernelArg(kernel, 0, sizeof(cl_mem), &a_d);
  CHECK_ERROR(err);
  err = clSetKernelArg(kernel, 1, sizeof(cl_mem), &b_d);
  CHECK_ERROR(err);
  err = clSetKernelArg(kernel, 2, sizeof(cl_mem), &c_d);
  CHECK_ERROR(err);
  err = clSetKernelArg(kernel, 3, sizeof(int), &M);
  CHECK_ERROR(err);
  err = clSetKernelArg(kernel, 4, sizeof(int), &N);
  CHECK_ERROR(err);
  err = clSetKernelArg(kernel, 5, sizeof(int), &K);
  CHECK_ERROR(err);

  // Setup global work size and local work size
  size_t gws[2] = {(size_t)M, (size_t)N}, lws[2] = {1, 1};
  for (int i = 0; i < 2; ++i) {
    // By OpenCL spec, global work size should be MULTIPLE of local work size
    // Formula below achieve it
    // e.g., gws = 25, lws = 16, then (25 + 16 - 1) / 16 * 16 = 40 / 16 * 16 = 2 * 16 = 32
    gws[i] = (gws[i] + lws[i] - 1) / lws[i] * lws[i];
  }

  // Run kernel
  err = clEnqueueNDRangeKernel(queue, kernel, 2, NULL, gws, lws, 0, NULL, NULL);
  CHECK_ERROR(err);

  // DO NOT REMOVE; NEEDED FOR TIME MEASURE
  err = clFinish(queue);
  CHECK_ERROR(err);
}
```

OpenCL matmul

```
void mat_mul_final(float *A, float *B, float *C, int M, int N, int K) {
  // Read from GPU; c_d (gpu) -> C (cpu)
  err = clEnqueueReadBuffer(queue, c_d, CL_TRUE, 0, M * N * sizeof(float), C, 0, NULL, NULL);
  CHECK_ERROR(err);

  // DO NOT REMOVE; NEEDED FOR TIME MEASURE
  err = clFinish(queue);
  CHECK_ERROR(err);
}
```

Hint

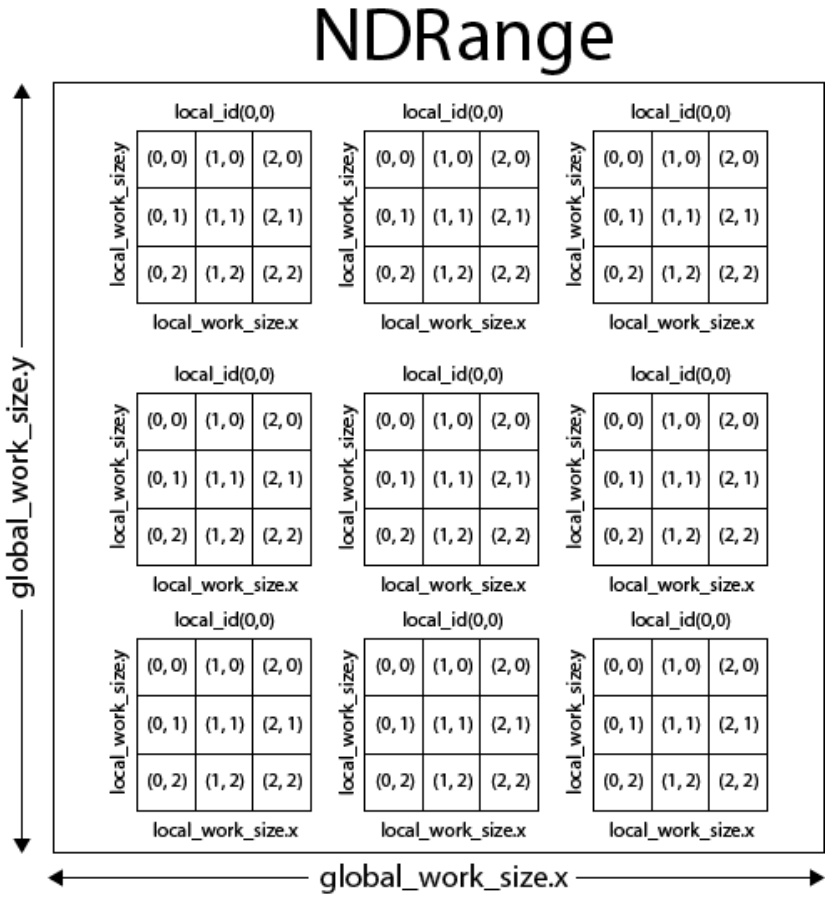
- `mat_mul.cpp` 에 이미 상당 부분이 구현되어 있음
  - `void print_platform_info(cl_platform_id platform)`
  - `void print_device_info(cl_device_id device)`
  - `cl_program create_and_build_program_with_source(cl_context context, cl_device_id device, const char *file_name)`

- 주어진 뼈대 코드를 잘 이해하고 시작할 것
  - Multi-GPU 프로그래밍 및 시험에 필수적임

- Single-GPU 에서 충분한 성능을 달성한 뒤 Multi-GPU 로 넘어가는 것을 추천

Global/Local work size

- https://stackoverflow.com/questions/16566127/get-global-threadid-in-2-dimensions-in-opencl



OpenCL matmul - 평가 기준

- 정확성 점수 50%
  - 무작위 N M K 에 대해 -v 옵션을 주어 값 검증을 통과해야 함
  - 총 10개의 test case 로 채점할 예정
  - N, M, K 각각은 2048 이하

- 성능 점수 50%
  - **4GPU,** 10 iteration, 8192 x 8192 x 8192 기준 **6000 GFLOPS** 달성 시 만점
  - 그 이하는 성능에 비례해 점수 부여
  - 답이 틀리면 0점 (-v 옵션)
  - ex. 3000 GFLOPS 달성 시 성능 점수의 절반 부여

OpenCL matmul - 생각해 볼 것

- 뼈대 코드로 제공한 OpenCL 환경 설정 코드를 읽고 이해해보기
  - clGetPlatformIDs
  - clCreateContext
  - clCreateCommandQueue
  - ...

- CPU 최적화와 GPU 최적화의 차이점

OpenCL matmul - 주의사항

- `mat_mul.cpp` 및 `kernel.cl` 만 수정 가능
- 다양한 행렬 크기에 대해 답이 맞는지 체크를 할 것 (-v 옵션)
- 로그인 노드에서 성능 측정을 하지 말 것
  - 반드시 주어진 run.sh 혹은 srun 으로 계산 노드에서 실행

OpenCL matmul - 주의사항

- GPU 에서만 연산을 수행할 것
  - CPU-GPU 로드 밸런싱은 본 과제의 범위가 아님

- OpenCL 이외의 병렬화 방식 (Pthread, OpenMP, MPI, CUDA 등) 사용 금지
  - Vector instructin 은 사용 가능

제출 방법

- 각자의 홈 디렉토리에 `~/submit/HW5` 디렉토리 생성
- 구현한 mat_mul.cpp 및 kernel.cl 파일을 디렉토리 내에 복사

- **디렉토리명, 파일명이 정확히 일치하도록**
- 즉, `ls` 명령의 출력이 아래 스크린샷과 같이 나와야 함

```
kjp4155@login:~$ ls ~/submit/HW5
kernel.cl  mat_mul.cpp
kjp4155@login:~$
```

- check-submit 유틸리티 활용

```
kjp4155@login:~$ check-submit
Username: kjp4155
------------------------------------
HW1 prob1.txt   : X file not found
HW1 prob2.txt   : OK
HW1 main.cpp    : X empty file
------------------------------------
HW2 mat_mul.cpp : OK
------------------------------------
HW4 mat_mul.cpp : OK
------------------------------------
HW5 mat_mul.cpp : OK
HW5 kernel.cl   : OK
------------------------------------
kjp4155@login:~$
```

주의사항

- **제출 기한: 5월 11일 (수) 14:29**
  - 기한이 되면 프로그램이 자동으로 제출 파일을 복사해 감
  - 파일명, 디렉토리, 형식을 정확히 지켰는지 체크
  - 평가를 하는 과제

- **조교가 실수를 수정해줄 것이라 기대하지 않을 것**
  - 제출 디렉토리 확인
  - 제출 파일명 확인
  - 제출한 파일 내용 확인
  - 프로그램이 불필요한 출력을 하는 지 확인
  - 디버깅 코드 등 불필요한 코드를 모두 제거했는지 확인