TA Session 4

조교 김진표, 박대영, 신준식, 정재훈

pp-ta@aces.snu.ac.kr

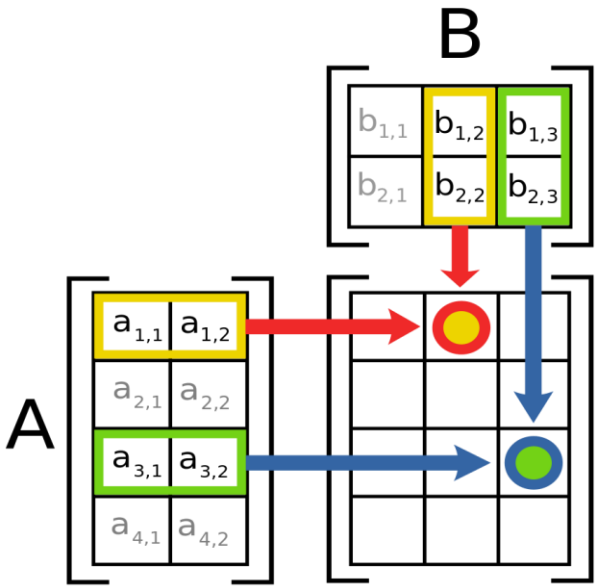
THUNDER Research Group
Seoul National University
서울대학교 천둥 연구실

구성

- 과제4 MPI+OpenMP matmul 소개
  - 채점을 하는 과제임
- 과제4 진행

과제4: MPI + OpenMP matmul

- FP32 행렬 둘을 곱하는 프로그램을 MPI+OpenMP 를 이용해 최적화
  - 4개의 노드 모두 사용
- 과제4 는 채점을 함



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

MPI + OpenMP matmul

- /skeleton/HW4 에 주어진 뼈대 코드를 각자의 홈 디렉토리로 복사해 작업
  - `cp -r /skeleton/HW4 ~/`
  - `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` 만 수정 가능

MPI 환경

- 각 계산 노드는 100Gbps Infiniband 로 연결되어 있음
- 실습 서버에는 OpenMPI 4.1.3 + UCX 1.12.1 이 설치되어 있음

- 주어진 run.sh 을 사용하면 OpenMPI 가 Infiniband 를 사용해 통신

```
1  #!/bin/bash
2
3  : ${NODES:=4}
4
5  salloc -N $NODES --exclusive                    \
6    mpirun --bind-to none -mca btl ^openib -npernode 1 \
7    numactl --physcpubind 0-39                    \
8    ./main $@
9
```

MPI + OpenMP matmul - 실행 예시

- make 로 컴파일
- `run.sh, run_performance.sh, run_validate.sh` 으로 실행
  - salloc + mpirun 을 통해 계산 노드에서 실행하는 커맨드

```
#!/bin/bash

: ${NODES:=4}

salloc -N $NODES --exclusive                                    \
  mpirun --bind-to none -mca btl ^openib -npernode 1 \
  numactl --physcpubind 0-39                                    \
  ./main $@

```

MPI + OpenMP matmul - 실행 예시

- `./run.sh` 은 NODES 환경 변수로 노드 개수 지정
- 예시) `NODES=3 ./run.sh -t 15 -v 1024 1024 1024`

```
kjp4155@login:~/HW4$ NODES=3 ./run.sh -v -t 10 512 512 512
salloc: Granted job allocation 45123
Hello world from processor b1, rank 1 out of 3
Hello world from processor b2, rank 2 out of 3
Hello world from processor b0, rank 0 out of 3
Options:
  Problem size: M = 512, N = 512, K = 512
  Number of threads: 10
  Number of iterations: 1
  Print matrix: off
  Validation: on

[rank 0] Initializing matrix...
[rank 0] Initializing matrix done!
[rank 0] Calculating...(iter=0) 0.059799 sec
Validating...
Result: VALID
[rank 0] Avg. time: 0.059799 sec
[rank 0] Avg. throughput: 4.488966 GFLOPS
salloc: Relinquishing job allocation 45123
kjp4155@login:~/HW4$
```

Hint

- main.cpp 를 읽고 이해할 것
  - rank 0 process 만 A B C 행렬을 생성하고 초기화함
  - 나머지 프로세스에서는 직접 행렬들을 필요한 만큼 할당/초기화 해야 함
- 한 프로세스가 여러 스레드에 작업을 나누어 주었던 것처럼, 여러 프로세스에 작업을 나누어 주면 됨
  - 얼마나 나누어 줄 것인지?
  - 각 프로세스 내의 스레드는 어떻게 나눌 것인지?
  - 데이터 전송을 어떻게 할 것인지?

MPI+OpenMP matmul - 평가 기준

- 정확성 점수 50%
  - 무작위 N M K, 노드 개수 및 스레드 개수에 대해 -v 옵션을 주어 값 검증을 통과해야 함
  - 총 10개의 test case 로 채점할 예정
  - N, M, K 각각은 2048 이하
  - 노드 개수는 1~3개
  - 노드당 스레드 개수는 40개 이하

- 성능 점수 50%
  - 3노드, 노드당 20 스레드 사용, 10 iteration, 8192 x 8192 x 8192 기준 **525 GFLOPS** 달성 시 만점
  - 그 이하는 성능에 비례해 점수 부여
  - 답이 틀리면 0점 (-v 옵션)

MPI + OpenMP matmul - 생각해 볼 것

- mpirun 에 사용된 옵션들이 무엇을 의미하는 지
  - 다른 옵션들은 어떤 것이 있는 지
- 노드 개수가 증가함에 따라 성능이 비례해서 증가하는지
- 전체 matmul 시간 중 통신 시간과 연산 시간의 비율이 어떻게 되는 지

MPI + OpenMP matmul - 주의사항

- `mat_mul.cpp` 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 salloc+mpirun 으로 계산 노드에서 실행
- 큐에 너무 많은/오래 걸리는 작업을 제출하지 말 것
  - 구현 단계에서는 작은 문제 크기 & 2노드로 실험
  - 20초 이상 걸리는 작업은 스스로 kill

MPI + OpenMP matmul - 주의사항

- 노드당 num_threads 개의 스레드를 사용해 계산을 수행할 것
  - -t 20 옵션으로 4개의 노드에서 실행하면 총 20 * 4 = 80 개의 스레드를 사용하는 것

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

제출 방법

- 각자의 홈 디렉토리에 `~/submit/HW4` 디렉토리 생성
- 구현한 mat_mul.cpp 파일을 디렉토리 내에 복사

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

```
kjp4155@login:~$ ls ~/submit/HW4
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
------------------------------------
kjp4155@login:~$
```

주의사항

- **제출 기한: 5월 11일 (월) 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

OpenMP + MPI matmul 해설

- 예시 답안을 `/skeleton/HW4-ans` 에 준비해 둠
  - `mat_mul.cpp`
    - 3노드 기준 ~700 GFLOPS

OpenMP + MPI matmul 해설

- mat_mul.cpp
  - 각 노드가 C 의 일부분을 나누어 계산
  - 행 단위로 나눔
  - 각 노드 내에선 다시 담당하는 부분의 일부분을 OpenMP 스레드들이 나누어 계산

OpenMP + MPI matmul 해설

- mat_mul.cpp
  - A, B 를 각 노드 (MPI Rank) 에 나누어 주기 (Scatter, Broadcast)
  - 노드 간 메모리는 공유가 아님

```
60
61    // Scatter A
62    if (mpi_rank == 0) {
63      for (int i = 1; i < mpi_world_size; i++) {
64        MPI_Send(A + is[i] * K, (ie[i] - is[i]) * K, MPI_FLOAT, i, 0,
65                 MPI_COMM_WORLD);
66      }
67    } else {
68      MPI_Recv(A + is[mpi_rank] * K, (ie[mpi_rank] - is[mpi_rank]) * K, MPI_FLOAT,
69               0, 0, MPI_COMM_WORLD, nullptr);
70    }
71
72    // Broadcast B
73    MPI_Bcast(B, K * N, MPI_FLOAT, 0, MPI_COMM_WORLD);
```

OpenMP + MPI matmul 해설

- mat_mul.cpp
  - 계산 결과 C 를 Rank 0 프로세스에 모으기 (Gather)

```
75    mat_mul_omp(is[mpi_rank], ie[mpi_rank]);
76
77    // Gather C
78    if (mpi_rank == 0) {
79      for (int i = 1; i < mpi_world_size; i++) {
80        MPI_Recv(C + is[i] * N, (ie[i] - is[i]) * N, MPI_FLOAT, i, 0,
81                 MPI_COMM_WORLD, nullptr);
82      }
83    } else {
84      MPI_Send(C + is[mpi_rank] * N, (ie[mpi_rank] - is[mpi_rank]) * N, MPI_FLOAT, 0, 0,
85               MPI_COMM_WORLD);
86    }
87  }
```