과제 #3

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

Due: 2024년 10월 27일(화) 23:59:59

1 (60점) Matrix Multiplication using OpenMP

두 FP32 행렬의 곱을 여러 CPU 스레드가 협업해 계산하는 프로그램 `matmul_omp.c` 를 작성하라. CPU 스레드 생성을 위해서 OpenMP 라이브러리를 사용할 것. 병렬화를 위해 Pthread 등 다른 라이브러리를 사용하는 것은 불허한다.

뼈대 코드와 Makefile 이 실습 서버 로그인 노드의 `/shpc/skeleton/hw3/openmp` 디렉토리에 제공된다. 뼈대 코드를 읽고 이해한 뒤, `matmul_omp.c` 파일을 작성하면 된다. `matmul_omp.c` 를 제외한 파일은 수정 불가능하다 (채점 시 `matmul_omp.c` 를 제외한 파일들은 뼈대코드의 것을 사용함).

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

```
$ ./run.sh -v -n 3 -t 1 123 456 789
Options:
  Problem size: M = 123, N = 456, K = 789
  Number of threads: 1
  Number of iterations: 3
  Print matrix: off
  Validation: on

Initializing... done!
Calculating...(iter=0) 0.022331 sec
Calculating...(iter=1) 0.022036 sec
Calculating...(iter=2) 0.020891 sec
Validating...
Result: VALID
Avg. time: 0.021752 sec
Avg. throughput: 4.068816 GFLOPS

$ ./run.sh -v -n 10 -t 20 1024 1024 1024
Options:
```

```
  Problem size: M = 1024, N = 1024, K = 1024
  Number of threads: 20
  Number of iterations: 10
  Print matrix: off
  Validation: on

Initializing... done!
Calculating...(iter=0) 0.037430 sec
Calculating...(iter=1) 0.045339 sec
Calculating...(iter=2) 0.035642 sec
Calculating...(iter=3) 0.039488 sec
Calculating...(iter=4) 0.044530 sec
Calculating...(iter=5) 0.037089 sec
Calculating...(iter=6) 0.046988 sec
Calculating...(iter=7) 0.038763 sec
Calculating...(iter=8) 0.036270 sec
Calculating...(iter=9) 0.041329 sec
Validating...
Result: VALID
Avg. time: 0.040287 sec
Avg. throughput: 53.304776 GFLOPS
```

보고서에 다음 질문들에 대한 답을 서술하라

- 자신의 병렬화 방식에 대한 설명.
- OpenMP는 사용자가 명시적으로 스레드 생성 함수를 호출하지 않는다. OpenMP에서 thread 생성은 어떤 식으로 이루어지는가? 컴파일러와 런타임 시스템이 각각 어떤 역할을 수행하는 지 생각해보자.
- 스레드를 1개부터 256개까지 사용하였을 때의 행렬곱 성능을 측정해 보자 (스레드 개수는 적당한 간격을 두고 측정). 스레드 개수가 늘어남에 따라 일정한 추세로 성능이 증가하는가? 이유는 무엇인가?
- 가장 높은 성능을 보이는 스레드 개수에서 행렬곱 성능은 1번 문제에서 계산한 peak performance 대비 어느 정도인가? 더 높은 성능을 달성하기 위해선 어떤 점을 개선해야 하는가?
- OpenMP의 loop scheduling 방식에 대해 알아보자. `static, dynamic, guided` 방식이 각각 어떤 것인지 서술하고, 실험을 통해 성능을 비교하라.

본 문제는 정확성 및 성능 평가를 한다. 채점 기준은 다음과 같다.

보고서 (15점) 명시된 질문들에 대한 답으로 평가.

정확성 (20점) 64개 이하의 스레드, 4096 이하의 M, N, K에 대해서 `-v` 옵션을 통한 validation을 통과해야 한다. 제공된 `./run_validation.sh` 스크립트를 이용해 채점할 예정임.

성능 (25점) 실습 서버에서 32 스레드, $M = N = K = 4096$ 옵션을 주고 실행했을 때, 80 GFLOPS를 넘으면 만점. 그 이하는 비율에 따라 점수를 부여한다 (e.g., 72 GFLOPS인 경우 성능 점수의 90%를 부여). 답이

틀린 경우 0점. 제공된 `./run_performance.sh` 스크립트를 이용해 채점할 예정임. 주어진 `num_threads` 인자보다 많은 스레드를 생성해 계산하는 경우 0점. `matmul_omp` 이 호출되는 main 스레드는 개수에 포함하지 않음.

2 (40점) Estimating Cache Size

문제 1에서 제공된 프로그램을 활용하여 실습 서버의 계산 노드에 장착된 CPU의 cache (L1, L2, L3) 크기를 추정하는 방법을 설계하라. 각 cache (L1, L2, L3)에 대해 다음 항목을 고려하여 결과를 분석하고 논의하라. 또한, 설계한 방법으로 각 cache가 shared cache인지 private cache인지도 확인할 수 있는지 검토하고, 확인할 수 있다면 어떻게 확인할 수 있는지도 설명하라. (이 문제는 cache 크기의 정확한 추정보다는, 실험을 설계하고 수행하는 방법에 초점을 맞춰 채점됩니다.)

- (10점) L1 cache: 실험을 통해 cache 크기를 추정하고, private인지 shared인지도 확인할 수 있다면 이를 함께 분석하라. 이때 사용한 추정 방법을 설명하라.
- (10점) L2 cache: 실험을 통해 cache 크기를 추정하고, private인지 shared인지도 확인할 수 있다면 이를 함께 분석하라. 이때 사용한 추정 방법을 설명하라.
- (10점) L3 cache: 실험을 통해 cache 크기를 추정하고, private인지 shared인지도 확인할 수 있다면 이를 함께 분석하라. 이때 사용한 추정 방법을 설명하라.
- (10점) 추정값과 실제값 비교: 실험을 통해 알아낸 cache 크기와 인터넷에서 검색을 통해 확인할 수 있는 해당 CPU의 cache 크기를 비교하고 분석하라.

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 hw3 report.pdf` 명령을 실행한다.
- 제출할 `matmul_omp.c` 파일이 위치한 디렉토리에서 `shpc-submit submit hw3 matmul_omp.c` 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.
- OpenMP 가 아닌 다른 병렬화 라이브러리를 사용하지 말 것 (Pthread, CUDA, OpenCL 등).