

# 병렬 프로그래밍 중간 보고서

팀명: TEAM\_C\_TENCORE

팀명: 소프트웨어 융합대학 20143148 이호중  
소프트웨어 융합대학 20135357 한승탁  
컴퓨터공학과 20135114 김용화  
콘텐츠 IT 20125345 조준형

목표: Tensor core 기반의 행렬곱 가속화 및 딥러닝 적용 방식 분석

기간: 2018. 09. 01 ~ 2018. 12. 20

## 1. 일정

기간	주요내용
2018.09.01~ 2018.10.29	To implement cpu
2018.10.30~ 2018.11.06	To setup the CUDA development environment To implement general GPU version
2018.11.06~ 2018.11.13	To learn the Tensor Core document
2018.11.14~ 2018.11.26	To implement Tensor Core version
2018.11.27~ 2018.12.03	Experiments and analysis
2018.12.04~ 2018.12.07	To write a report

## 2. 진행사항

미팅 날짜	내용
10월 9일	-Tensor Core 기반의 행렬곱 가속화 및 딥러닝 적용 방식 분석으로 주제를 재확인 하였다. -다음 미팅 일정은 중간고사를 고려하여 10월 29일로 한다. -10월 29일 까지 다음 과제를 완성해야한다. 1.행렬의 연산 법칙 확인 2-1.C/C++ 언어로 CPU상에서 행렬의 연산 함수를 만든다. 2-2.작성한 행렬의 연산 함수들을 병렬화 시키는 방법을 글로 작성한다. 3-1.추가 사항 멀티 쓰레드 이용 2번에서 작성한 함수를 병렬화 해본다. 3-2.행렬의 연산을 간략화시킨 알고리즘들을 알아보고 병렬화가 가능한지 알아본다. - 첨부 2-1 과제 pdf파일 - 첨부 2-2 발표 ppt파일

미팅 날짜	내용
10월 29일	<p>10월 9일 과제 발표</p> <ul style="list-style-type: none"> <li>- CPU를 사용한 연산은 CPU 코어 수 이하로 쓰레드를 생성하도록 변경할 것</li> <li>* CPU 코어수를 초과한 쓰레드는 스케줄링이 되어서 시간이 더 걸리게 된다.</li> <li>* 추가적으로 AXB 이후 C연산시 상호배제, atom 연산 사용할 것</li> </ul> <p>10월 29일 과제</p> <ul style="list-style-type: none"> <li>- CUDA의 CUBLAS 라이브러리를 이용 해당 행렬 A*B+C 연산을 작성할 것</li> </ul> <p>첨부 2-3 미팅 모임 사진</p>

## 첨부 2-1 10월 9일 대학생들 과제 pdf

10.29 일까지 다음과 같이 내용을 완성 바랍니다.

1. Matrix 의 더하기와 곱하기 연산의 법칙을 복습하세요. (글로 정리)
2. C/C++이용하여 CPU 상에 Matrix 의 더함과 곱함 함수를 만들어보세요.

\* 예) matrix\_add()로 A+B 계산함, matrix\_multiplication()는 AxB 계산함.

- matrix 의 shape 가 parameter 로 지정할수 있다.
- 아래 행렬을 계산하고 출력해보세요.

If

$$A = \begin{bmatrix} 1 & 0 & -3 \\ -2 & 4 & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} 2 & -1 \\ 3 & 0 \\ -5 & 2 \end{bmatrix} \text{ and } C = \begin{bmatrix} 3 & -1 \\ -2 & 2 \end{bmatrix}$$

Then

$$A \times B = ? \text{ and } A \times B + C = ?$$

\* 작성한 더함과 곱함 함수들이 병렬화 시키는 방법을 생각해보세요.

- 나중에 보고서를 쓰실때 도움이 될거니까 글로 정리하세요


3. 추가 사항 (Optional):

\* Multi-thread 로 위의 matrix\_add(), matrix\_multiplication()함수 구현해보세요. (win32

api 나 pthread 등 library 이용)

\* 수학자들이 간화시킨 알고리즘들 알아보고 병렬화가 가능한지 생각해보세요.

위의 내용을 정리하고 발표자료 (연산의 법칙, 구현한 코드, 실행결과 캡처, 병렬화의 생각, 어려운점등 포함) 만들어서 발표하세요.



### 병렬프로그래밍 3조 첫번째\_행렬연산

행렬 연산의 법칙

CPU상에서의 행렬 연산

CPU상에서의 행렬 연산 병렬화

추가 - 슈트라센 알고리즘

#### 행렬연산\_행렬 연산의 법칙

**더하기 연산**

크기가  $m \times n$  인 두 행렬 A, B에 대해 각 연산은 동일한 위치상의 성분끼리의 연산으로 정의된다.

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$\Rightarrow A \pm B = \begin{pmatrix} a_{11} \pm b_{11} & a_{12} \pm b_{12} \\ a_{21} \pm b_{21} & a_{22} \pm b_{22} \end{pmatrix}$$

두 행렬의 크기가 서로 다를 경우는 연산할 수 없다.  
교환법칙이 성립한다.

**곱하기 연산**

크기가  $m \times n$ ,  $o \times p$ 인 두 행렬 A, B에 대해 AB연산을 할 경우  $n$ 과  $o$ 는 동일한 크기여야 하며 연산의 결과 행렬 AB는  $m \times p$ 의 크기를 갖는다.

$$AB = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

연산은 A행렬의 i행과 B행렬의 j열의 내적(스칼라곱)으로 정의된다.  
여기서 내적은 각 행렬의 동일 순서상의 성분끼리 곱한후 각 값을 합하는 것을 의미한다.

행렬의 곱연산은 결합법칙은 성립하나 교환법칙은 성립하지 않는다.

병렬프로그래밍 3조

#### 행렬연산\_CPU상에서의 행렬 연산

```
// 곱셈
void add(vector<vector<int>> &A, vector<vector<int>> &B){
    int row=A.size();
    int col=B[0].size();
    for(int i=0; i<row;i++){
        for(int j=0; j<col;j++){
            (*B)[i][j]=A[i][j];
        }
    }
}

void mult(vector<vector<int>> &A, vector<vector<int>> &B, vector<vector<int>> &C){
    int row=A.size();
    int col=B.size();
    for(int i=0; i<row;i++){
        for(int j=0; j<col;j++){
            (*C)[i][j]=A[i][j]*B[j][j];
        }
    }
}
```

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$\Rightarrow A \pm B = \begin{pmatrix} a_{11} \pm b_{11} & a_{12} \pm b_{12} \\ a_{21} \pm b_{21} & a_{22} \pm b_{22} \end{pmatrix}$$

$$AB = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

#### 행렬연산\_CPU상에서의 행렬 연산

```
void disp(vector<vector<int>> &C){
    int row=C.size();
    int col=C[0].size();
    for(int i=0; i<row; i++){
        for(int j=0; j<col; j++){
            cout<<C[i][j]<<" ";
        }
        cout<<endl;
    }
}

int main() {
    vector<vector<int>> da={{1,0,-3},{-2,4,1}};
    vector<vector<int>> db={{2,-1},{3,0},{-5,2}};
    vector<vector<int>> dc={{3,-1},{-2,2}};
    vector<vector<int>> dd={{0,0},{0,0}}; //결과 저장
```

$$\begin{matrix} \text{행렬 A} & \text{행렬 B} & \text{행렬 C} \\ \begin{pmatrix} 1 & 0 & -3 \\ -2 & 4 & 1 \end{pmatrix} & \begin{pmatrix} 2 & -1 \\ 3 & 0 \\ -5 & 2 \end{pmatrix} & \begin{pmatrix} 3 & -1 \\ -2 & 2 \end{pmatrix} \end{matrix}$$

A와 B를 곱한 행렬  
17 -7  
3 4

결과에 C를 더한 행렬  
20 -8  
1 6

병렬프로그래밍 3조

병렬프로그래밍 3조

#### 행렬연산\_CPU상에서의 행렬 연산 병렬화

```
void add_thread(vector<vector<int>> &A, vector<vector<int>> &B, int i){
    int col=B[0].size();
    for(int j=0; j<col;j++){
        (*B)[i][j]=A[i][j];
    }
}

void mult_thread(vector<vector<int>> &A, vector<vector<int>> &B, vector<vector<int>> &C, int i){
    int row=A.size();
    int col=B.size();
    for(int j=0; j<col;j++){
        for(int k=0; k<row;k++){
            (*C)[i][j]=A[i][k]*B[k][j];
        }
    }
}
```

A. 행렬 연산을 하는 함수에 계산할 행의 정보를 인자로 받을 수 있도록 만들었다.

#### 행렬연산\_CPU상에서의 행렬 연산 병렬화

```
int row=A.size();
thread myThread(row);

// 반복문을 통해 스레드를 생성하고 스레드 배열의 행을 할당하는 인자로 전달한다.
myThread(1) = thread(mult_thread, A, B, A, 1);

for(int i=0; i<row;i++){
    myThread(i).join();
}

disp(A);
size=dc.size();

for(int i=0; i<row;i++){
    myThread(i) = thread(add_thread, A, C, i);
}

for(int i=0; i<row;i++){
    myThread(i).join();
}

disp(C);
return 0;
```

B. 스레드를 계산할 행렬의 행의 개수 만큼 선언한다.

C. 스레드 배열 번호를 호출한 함수로 넘겨준다.

D. 스레드들이 종료된 것을 확인한 후 결과를 출력한다.

병렬프로그래밍 3조

병렬프로그래밍 3조

#### 행렬연산\_슈트라센 알고리즘

1. 행렬을 같은 크기의 정사각형 행렬 4개로 나눈다.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

2. 행렬 M을 다음과 같이 정의한다.

$$\begin{aligned} M_1 &:= (A_{11} + A_{22})(B_{11} + B_{22}) \\ M_2 &:= (A_{21} + A_{22})B_{11} \\ M_3 &:= A_{11}(B_{12} - B_{22}) \\ M_4 &:= A_{22}(B_{12} - B_{22}) \\ M_5 &:= (A_{11} - A_{12})(B_{21} + B_{22}) \\ M_6 &:= (A_{12} - A_{22})(B_{21} + B_{22}) \\ M_7 &:= (A_{11} - A_{22})(B_{21} + B_{22}) \end{aligned}$$

1의 결과  $C_{ij}$ 는 다음과 같이 정의된다.

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

2의 결과  $C_{ij}$ 는 다음과 같이 정의된다.

$$\begin{aligned} C_{11} &= M_1 + M_4 - M_5 + M_7 \\ C_{12} &= M_3 + M_5 \\ C_{21} &= M_2 + M_4 \\ C_{22} &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

□행렬의 전체 곱셈을 2번 과정처럼 7번의 곱셈과 18번의 덧셈으로 처리할 수 있다.

$$\rightarrow 7 \cdot n^{\log_2 7} - 6 \cdot n^2$$

#### 행렬연산\_슈트라센 알고리즘

알고리즘	시간 복잡도
<pre>void mult(vector&lt;vector&lt;int&gt;&gt; &amp;A, vector&lt;vector&lt;int&gt;&gt; &amp;B, vector&lt;vector&lt;int&gt;&gt; &amp;C){     int row=A.size();     int col=B[0].size();     for(int i=0; i&lt;row;i++){         for(int j=0; j&lt;col;j++){             (*C)[i][j]=A[i][j]*B[j][j];         }     } }</pre>	$O(n^3)$
<p>※문제를 세로로 나누고, 각 행렬을 2개의 부분행렬로 나눈다. (예를 들어, <math>A = \begin{bmatrix} A_{11} &amp; A_{12} \\ A_{21} &amp; A_{22} \end{bmatrix}</math> 라고 하면, <math>A_{11} = \begin{bmatrix} a_{11} &amp; a_{12} \\ a_{21} &amp; a_{22} \end{bmatrix}</math>, <math>A_{12} = \begin{bmatrix} a_{13} &amp; a_{14} \\ a_{23} &amp; a_{24} \end{bmatrix}</math> 라고 할 수 있다.)</p> <p>※이후로 행렬(스칼라) 곱</p> <p><math>C = \begin{bmatrix} C_{11} &amp; C_{12} \\ C_{21} &amp; C_{22} \end{bmatrix}</math>, <math>A = \begin{bmatrix} A_{11} &amp; A_{12} \\ A_{21} &amp; A_{22} \end{bmatrix}</math>, <math>B = \begin{bmatrix} B_{11} &amp; B_{12} \\ B_{21} &amp; B_{22} \end{bmatrix}</math></p> <p>행렬 곱</p> <p><math>C_{11} = A_{11}B_{11} + A_{12}B_{21}</math>  <math>C_{12} = A_{11}B_{12} + A_{12}B_{22}</math>  <math>C_{21} = A_{21}B_{11} + A_{22}B_{21}</math>  <math>C_{22} = A_{21}B_{12} + A_{22}B_{22}</math></p>	$O(n^{2.807})$

병렬프로그래밍 3조

병렬프로그래밍 3조

## 행렬연산\_슈트라센 알고리즘\_병렬화

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}, B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}, C = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$$

A. 행렬을 분할한다.

$$M_1 := (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2})$$

$$M_2 := (A_{2,1} + A_{2,2})B_{1,1}$$

$$M_3 := A_{1,1}(B_{1,2} - B_{2,2})$$

$$M_4 := A_{2,2}(B_{2,1} - B_{1,1})$$

$$M_5 := (A_{1,1} + A_{1,2})B_{2,2}$$

$$M_6 := (A_{2,1} - A_{1,1})(B_{1,1} + B_{1,2})$$

$$M_7 := (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2})$$

B. 각각의  $M_i$ 를 계산하는 작업을 각각 하나의 쓰레드에서 처리하도록 한다.

$$C_{1,1} = M_1 + M_4 - M_5 + M_7$$

$$C_{1,2} = M_3 + M_5$$

$$C_{2,1} = M_2 + M_4$$

$$C_{2,2} = M_1 - M_2 + M_3 + M_6$$

C.  $M_i$ 를 이용  $C_{ij}$ 를 만들고 통합한다.

병렬프로그래밍 3조

[https://github.com/HoJungLee/Parallel\\_Programming\\_2018\\_Fall/blob/master/Capstone/TEAM\\_C\\_TENCORE/matrix.cpp](https://github.com/HoJungLee/Parallel_Programming_2018_Fall/blob/master/Capstone/TEAM_C_TENCORE/matrix.cpp)

첨부 2-3 10월 29일 미팅 사진

