REPORT

KU 건국대학교 KONKUK UNIV.

#Assignment1 : Parallel CNN

과목명 | 시스템프로그래밍

담당교수 | 진 현 욱 교수님

학과 | 컴퓨터공학부

학년 | 4학년

학번 | 201714154

이름 | 오 병 현

제출일 | 2020. 10. 18

Design and implementation

1. **Input, Convolution result, Max-pooling result 를 저장할 배열 초기화**

   : Input Matrix 는 프로세스 실행시 main 함수의 인자 argv 로 받은 크기만큼 이차원 배열을 동적할당하고, makeMatrix 함수를 이용하여 값을 생성한다.

   Input Matrix 의 크기가 정해지면 Convolution 연산 수행 후 결과의 크기 및 Max-pooling 연산 결과의 크기를 알 수 있다. 사용하는 filter 의 크기가 3 이므로 convolution 결과 및 Max-pooling 결과 행렬의 크기는 다음과 같다.

   convolution 결과 크기 = (Input Matrix 의 크기) – (filter 의 크기) + 1 = (Input Matrix 의 크기) – 2

   Max-pooling 결과 크기 = (convolution 결과 크기) / 2

   따라서 이 크기를 토대로 두 가지 결과를 담을 배열을 동적 할당한다. 이 세가지 배열은 구현한 코드상에서 int** inputMtx, int** convRes, int* result 에 해당하며, result 의 경우 마지막에 출력만 해주면 되기 때문에 편의상 1 차원으로 만들었다. 주의할 점은 프로세스가 종료되기 전에 메모리 누수를 방지하기 위해 free 를 해주어야 한다.

2. **Message Queue 생성 (**Parallel CNN 에서 총 2 개의 POSIX Message Queue 를 사용**)**

   : 하나는 int 형 배열을 담기 위한 큐로서, Input Matrix 를 3x3 으로 겹쳐서 분할한 Convolution Input 과 Convolution 결과를 2x2 로 분할한 Max-pooling Input 을 자식 프로세스로 전송하는데 사용하는 큐이다. 나머지 하나는 int 형 정수를 담기 위한 큐이다. Convolution 결과와 Max-pooling 결과 모두 하나의 정수값이며, 이 값을 자식 프로세스에서 메인 프로세스로 전송해주기 위해서 사용한다. 두 가지 큐는 아래의 속성을 기반으로 open 하였다.

	메인 -> 자식 Queue	자식 -> 메인 Queue
용도	각 연산의 Input 배열 전송	결과 값 전송
attr.mq_maxmsg	10(default)	10(default)
attr.mq_msgsize	36	4

   mq_msgsize 의 경우 큐에 담을 메시지의 크기에 따라 다르게 설정하였다. 메인 -> 자식 Queue 의 경우 최대 3x3 int 형 배열을 전송해야 하므로 9 * (4byte) = 36 으로 설정하였고, 자식 -> 메인 Queue 의 경우 int 형 정수 결과 값을 전송하기 때문에 4byte 로 설정하였다.

3. **자식 프로세스를 생성할 수 있는 최대 개수 설정**

   : Parallel CNN 을 수행하는 방식을 먼저 설명하자면, 미리 필요한 자식 개수만큼 fork 를 하고, 메인에서 Input 을 mq_send 하는 방식이다. 그런데 만약 argv 로 들어오는 숫자가 많이 커져버리면, ulimit -a 에서 출력되는 max user processes 의 값을 초과하게 될 수 있다.

테스트시 실제로 그러했으며, blocking 이 되는 것을 볼 수 있었다. 따라서 자식프로세스가 동시에 존재할 수 있는 최대 개수(int max_prc)를 3000 개로 설정하였다(ulimit -a 상의 max user processes 를 고려하여 설정, 본인 pc 에서는 24000 까지 가능하다). 결과적으로 생성해야하는 자식프로세스의 개수가 만약 17000 개 라면, 3000 개씩 5 번 처리 후, 마지막에 2000 개를 처리하게 되며, 각 처리마다 결과값은 convRes 배열에 앞부분부터 저장되게 된다.

4. Convolution 연산을 수행하는 Cworkers 프로세스 생성

void makeCworkers(mqd_t mqdes, mqd_t mqdes2, int max_prc, int upper_loop_i, int term)

: 메인에서 Input Matrix 를 3x3 으로 분할하여 큐에 삽입하는 작업을 하기 이전에, Convolution 연산을 수행할 자식 프로세스를 미리 생성한다. 만약 자식 프로세스를 생성하기도 전에 Input Matrix 를 분할하여 큐에 send 하는 작업을 하게 될 경우, 메시지 개수가 10 개가 넘어가는 순간 block 되고, 자식 프로세스를 생성하지도 못한채 멈추게 된다. 따라서 자식 프로세스를 미리 생성해두어야 큐에 메시지를 삽입하는 즉시 자식 프로세스가 receive 를 하여 메인에서도 계속해서 send 를 수행할 수 있다. (자식 하나 생성 후 한 개의 input 을 send 하고 결과를 receive 할 수 도 있겠으나, paralell 이 아니라고 생각하여 이 방식을 선택했습니다.)

fork() 를 통해 생성하게 될 자식 프로세스의 총 개수는 Convolution 연산을 수행하는 횟수 만큼이며, 이는 앞서 1. 에서 계산한 Convolution 결과 크기의 제곱과 같다.

자식 프로세스들은 "배열 담는 큐"에서 receive 한 배열에 대해 Conv 연산을 수행하고, 결과값을 "정수 담는 큐"에 send 한 이후 종료된다. 메인에서 SIGCHILD 에 대한 핸들러를 등록하여 자식 프로세스가 종료한 경우 핸들러가 호출되어 reaping 작업을 수행한다.

5. Input Matrix 를 Filter 크기만큼 분할하여 Message queue 에 저장 (main)

void insertConvInputtoQ(mqd_t mqdes,int max_prc,intterm,intupper_loop_i,int** inputMtx, int c onvResSize)

: 분할 방법은 0 에서 numofmsg(큐에 들어갈 3x3 배열 개수)까지의 정수 값을 convResSize 로 나눈 몫과 나머지를 시작 인덱스로 3x3 Matrix 만큼 분할하여 메시지큐에 send 하는 것이다. 만약 numofmsg 가 3000(max_prc)보다 클 경우, 앞의 3000 개 input 을 먼저 보내고 자식들에서 처리하여 결과를 받은 다음에 뒷부분을 또 다시 mq_send 한다. 구현 상에서 int term 은 실제로 생성하여 동시에 존재하게 될 자식의 개수이다.

※ POSIX Message Queue 의 priority 는 이후에 메인에서 mq_receive 를 수행한 이후 순서를 맞추는데 사용된다. 메인에서 Input Matrix 를 분할하여 큐에 삽입하는 과정은 순차적으로 진행되기 때문에 문제가 되지 않으나, 자식 프로세스들이 Convolution 연산을 수행하고 결과값을 큐에 넣는데 걸리는 시간이 제각각이라 메인에서 이 결과를 receive 할 때, 실제 의도하던 순서와는 다르게 받을 수 있다. 따라서 이러한 순서들을 맞춰주기 위해 메인에서

priority 를 term 으로 초기화하고 3x3 분할 배열을 차례대로 삽입할 때, 1 씩 순차적으로 감소시키면서 mq_send 를 수행한다. 이 priority 값은 메인에서 다시 receive 할 때 까지 유지되어 결과값을 올바른 배열 위치에 저장할 수 있다.

6. Message Queue 에 저장된 convolution 결과 받기 (main)

void receiveConvRes(mqd_t mqdes, int** convRes, int max_prc, int upper_loop_i, int term)

: 위에서 설명한 바와 같이 결과를 받을 때, convRes 배열의 올바른 위치에 저장하기 위해 다음과 같이 priority 값을 이용하여 다음과 같이 작성하였다.

```
convRes[ (numofmsg-prio) / convResSize ][ (numofmsg-prio) % convResSize ] = tmp;
```

이렇게 작성할 경우, 더 높은 우선순위를 가진 값이 늦게 큐로 삽입되어, 이미 그전에 더 낮은 우선순위를 가진 값이 receive 되는 상황이 벌어진다 하더라도 최종적으로 올바른 순서를 유지할 수 있다.

7. Max-pooling 연산을 수행하는 Pworkers 프로세스 생성

void makePworkers(mqd_t mqdes, mqd_t mqdes2, int max_prc, int upper_loop_i, int term)

: Pworker 도 Cworker 와 마찬가지로 최대 3000 개의 자식이 병렬로 처리하게 된다.

Pworker 에서 임시로 Input 2*2 행렬 값을 받아두는 배열의 shape 가 3*3 인 이유에 대해 설명하자면, 처음에 생성한 "메인 -> 자식 Queue" 의 메시지 최대용량이 3*3 배열에 맞춰 36byte 로 설정했는데, 이보다 더 작은 용량의 버퍼를 이용하게 되면 mq_send & mq_receive 과정에서 message too long 에러가 발생하기 때문이다. 아예 2*2 배열 용량에 맞춰서 메시지 큐를 또 open 하기 보다는 전에 사용한 큐를 재사용하는 것을 선택했다.

8. Convolution 결과 배열을 2x2 형태로 분할하여 Message queue 에 저장 (main)

void insertPoolInputtoQ(mqd_t mqdes, int** convRes, int term, int max_prc, int upper_loop_i, int convResSize);

: 위에서 설명한 바와 같이 2x2 지만, 메시지 큐 용량에 맞추기 위해 3x3 배열을 사용하였으며, 좌측상단 2x2 만큼만 값을 저장하여 mq_send 를 한다. 이때도 향후 올바른 순서로 결과값 저장을 위해 priority 를 이용한다.

9. Message Queue 로부터 Parallel CNN 연산의 최종 결과 받기

: Max-pooling 까지 마친 결과 값들을 Message Queue 로부터 받는다. 이때도 다음과 같이 priority 를 이용하여 result 배열의 올바른 위치에 값을 저장한다.

```
result[prio-1] = tmp;
```

이후 이 값들을 순서대로 출력하고 프로세스를 종료하게 된다.

Function description

Parallel CNN 은 아래의 함수 순서대로 호출되어 진행된다.

Function Name	Arguments	Desciption
makeCworkers	mqdes : mqd_t	메인으로부터 3x3 분할 배열을 받기 위해 사용하는 Message Queue Descriptor
	mqdes2 : mqd_t	메인으로 Convolution 결과를 전송하기 위해 사용하는 Message Queue Descriptor
	max_prc : int	동시에 존재할 수 있는 자식프로세스의 최대 개수
	upper_loop_i : int	상위 루프의 인덱스
	term : int	상위 루프의 턴마다 생성할 자식프로세스의 개수
	Return Value : void	-

Function Name	Arguments	Desciption
insertConvInputtoQ	mqdes : mqd_t	자식으로 3x3 분할 배열을 전송하기 위해 사용하는 Message Queue Descriptor
	max_prc : int	동시에 존재할 수 있는 자식프로세스의 최대 개수
	term : int	상위 루프의 턴마다 생성할 자식프로세스의 개수
	upper_loop_i : int	상위 루프의 인덱스
	inputMtx : int**	Input Matrix 를 의미하며, 이것을 분할하는 과정을 거친다.
	convResSize : int	Convolution 결과 배열의 크기를 나타내며, Input Matrix 를 분할하는데 사용된다.
	Return Value : void	-

Function Name	Arguments	Desciption
receiveConvRes	mqdes : mqd_t	자식으로부터 Convolution 결과를 받기 위해 사용하는 Message Queue Descriptor
	convRes : int**	Message Queue 로부터 받은 Convolution 결과 정수 값을 저장할 배열이다.
	max_prc : int	동시에 존재할 수 있는 자식프로세스의 최대 개수
	upper_loop_i : int	상위 루프의 인덱스
	term : int	상위 루프의 턴마다 생성할 자식프로세스의 개수
	convResSize : int	Convolution output 배열의 크기이다.
	Return Value : void	-

Function Name	Arguments	Desciption
makePworkers	mqdes : mqd_t	메인으로부터 2x2 분할 배열을 받기 위해 사용하는 Message Queue Descriptor
	mqdes2 : mqd_t	메인으로 Max-pooling 결과를 전송하기 위해 사용하는 Message Queue Descriptor
	max_prc : int	동시에 존재할 수 있는 자식프로세스의 최대 개수
	upper_loop_i : int	상위 루프의 인덱스
	term : int	상위 루프의 턴마다 생성할 자식프로세스의 개수
	Return Value : void	-

Function Name	Arguments	Desciption
insertPoolInputtoQ	mqdes : mqd_t	자식으로 2x2 분할 배열을 전송하기 위해 사용하는 Message Queue Descriptor
	convRes : int**	Convolution output 배열이며, 이것을 2x2 로 분할하여 전송한다.
	term : int	상위 루프의 턴마다 생성할 자식프로세스의 개수
	max_prc : int	동시에 존재할 수 있는 자식프로세스의 최대 개수
	upper_loop_i : int	상위 루프의 인덱스
	convResSize : int	Convolution output 배열의 크기이다.
	Return Value : void	-

Function Name	Arguments	Desciption
receiveFinalRes	mqdes : mqd_t	자식으로부터 최종적인 연산 결과 값을 받기위해 사용되는 Message Queue Descriptor 이다.
	max_prc : int	동시에 존재할 수 있는 자식프로세스의 최대 개수
	upper_loop_i : int	상위 루프의 인덱스
	term : int	상위 루프의 턴마다 생성할 자식프로세스의 개수
	result : int*	큐에서 받은 결과 값을 저장할 배열이다.
	Return Value : void	-