REPORT

KU 건국대학교 KONKUK UNIV.

#Assignment2 : Parallel CNN

과목명 | 시스템프로그래밍

담당교수 | 진 현 욱 교수님

학과 | 컴퓨터공학부

학년 | 4학년

학번 | 201714154

이름 | 오 병 현

제출일 | 2020. 11. 25

Design and implementation

1. input_file 로부터 input matrix 크기 읽기

- **Design** : input_file 을 open 하고, 첫줄의 개행 문자까지 read 를 하여 input matrix 크기를 확인할 수 있다. 구체적으로는 1Byte 씩 read 하고 정수로 변환하여 자릿수를 유지해주는 작업이 필요하다. read 를 사용하기 때문에 개행문자를 만나고 나면, offset 의 위치는 input matrix 의 첫번째 위치를 가리키게 되며, lseek 을 이용하여 현재 offset 을 구할 수 있다. 이후의 읽기 작업은 병렬로 Cworker 안에서 수행되기 때문에, 현재 offset 을 유지한채 pread 를 사용하여 필요한 위치에서 읽게 된다.

- **implementation**
  - **int get_inputsize(int infd)**
  - main 함수의 **int size** 에 input matrix 크기 저장
  - **off_t offset = lseek(infd, 0, SEEK_CUR)** 로 현재 offset 저장

2. Parallel Convolution, Max-pooling Layer Architecture

각 레이어 별 결과 크기와 worker 개수는 다음과 같다.

- convolution 결과 크기 = input 크기 – 2 ... int convResSize
- max-pooling 결과 크기 = (input 크기 – 2) / 2
- cworker 개수 = (convolution 결과 크기) * (convolution 결과 크기) ... int cwork_count
- pworker 개수 = (convolution 결과 크기 / 2) * (convolution 결과 크기 / 2)... int pwork_count

또한 cworker 와 pworker 에 주어지는 각 인자는 미리 정의한 구조체 타입으로 주어진다. 두 인자는 동일한 형태이므로 cworker 의 인자를 대표로 보자면, 다음과 같다.

구조체 이름	멤버 변수	설명
conv_arg	**int** pt_num	생성된 pthread 번호이며, 용도는 각 cworker 에서 읽을 3*3 input 의 시작 위치를 계산하기 위함
	int* remained_cworkers	남은 cworker 개수이며, conv layer 에서 pool layer 로 넘어갈 때, 모든 cworker 가 종료되었는지 확인하기 위해 존재
	info* info	모든 worker 가 사용하는 공통 정보이며, info 또한 구조체로 정의된 타입이다.

2.1. Convolution Layer

- **Cworker Design**

전체적인 속도 향상을 위해 main 에서 input_file 을 읽지 않고, 각 cworker 스레드에서 곧바로 3*3 크기의 input 을 읽는다. 스레드 별로 읽어야할 input 의 위치가 다르지만, 같은 프로세스는 offset 을 공유하기 때문에 offset 을 변경하지 않고 읽어야한다. 즉, 1. 에서 구한 시작 offset 을 기준으로 각 스레드가 연산을 수행할 3*3 input 의 시작 offset 을 계산하고, pread 를 이용하여 읽어내야한다. 이를 위해 스레드 별로 0 부터 (cworker 개수 – 1) 까지의 번호를 부여하였으며, 위 표의 pt_num 에 해당한다. 각 스레드가 연산을 수행할 3*3 input 의 시작 offset 을 계산하는 방법은 다음 식을 이용하며, pt_num 에 따라 각 스레드가 읽을 위치가 정해진다.

(input matrix 시작 offset) + (**pt_num** / conv 결과크기) * (3Byte) * (input matrix size)

+ (**pt_num** % conv 결과크기) * (3Byte)

각 cworker 들은 이 위치로부터 읽고, 정수로 변환하면서 input 을 만들어내고, convolution 연산을 수행한다. 이후 스레드는 별도의 return 없이 곧바로 공유하고 있는 convRes 배열의 올바른 위치에 결과 값을 저장한다. 마지막으로 공유하는 동적메모리 변수인 remained_cworkers 를 1 감소시키고 종료한다.

- **Implementation :** **void** *convolution(**void*** **argument**)

- Cworkers Maker Design

pthread_create 로 필요한 cworker 개수만큼 스레드를 생성하여, Convolution 연산이 완전 **병렬**로 일어나도록 한다. 이때 **주의해야할 점**은 pthread_create 직후 pthread_detach 를 이용하여 해당 스레드의 detach state 를 detached thread 가 되도록 변경해야한다. 이는 각 스레드가 종료되었을 때 자동으로 자원 해제가 일어나도록 하기 위함이다. 만약 joinable thread 로 필요한 모든 스레드를 생성한 이후, 해당 개수만큼 pthread_join 으로 직접 자원 해제할 경우, 스레드 생성 도중 일정 수준을 넘어서게 되면 스레드가 더 이상 생성되지 않는 문제가 생긴다. 이는 계속해서 스레드는 생성되지만, reap and kill 과정은 모든 스레드가 생성된 이후에 진행되기 때문에 스레드 개수가 줄어들지 않아서 발생하는 문제이다. 따라서 detached thread 를 이용해야 생성하는 도중에 이미 종료한 스레드들이 자원 해제를 함으로써 계속해서 스레드를 생성할 수 있게 되는 것이다. 그러나 detached thread 를 이용할 경우, Cworker 들을 모두 생성하자 마자 join 과정 없이 바로 Pworker 들을 생성하는 단계로 넘어간다. 이럴 경우, Cworker 가 계산이 덜 된 상태로 진행되면 올바르지 않은 결과가 나올 수 도있다. 따라서 Cworker Layer 와 Pworker Layer 사이에 Cworker 가 모두 작업을 종료하였다는 사실을 보장해주는 단계가 필요하며, 이를 위해 **int*** remained_cworkers 변수를 사용한다. main 에서 remained_cworkers 가 가리키는 값을 cworker 의 개수로 초기화 후 각 cworker 스레드가 종료하기 전에 1 씩 감소시킨다. 이 변수는 동적 할당 메모리를 가리키기 때문에 스레드간 공유되며, race condition 을 방지하기 위해 mutex 를 이용하여 앞뒤로 lock & unlock 을 한다.

- **Implementation :** **void** make_cworkers(**void*** (***convolution**)(**void***), **conv_arg*** cargs, **int** cwork_count, **int*** remained_cworkers, **info*** info)

2.2. Max-pooling Layer

- **Pworker Design**

Pworker 도 마찬가지로 스레드 마다 주어지는 pt_num 을 이용하여 Max-pooling 연산을 수행할 위치를 계산한다. 다음 식을 이용하여 찾을 수 있다.

행 : (2 * pt_num / (conv 결과 크기)) * 2 / 열 : 2 * pt_num % (conv 결과 크기)

Pworker 는 인자로 conv 결과 배열에 대한 주소(convRes)와 pooling 결과 배열에 대한 주소(poolRes)를 모두 받기 때문에, 각 스레드마다 위의 식을 통해 계산한 convRes 위치로부터 Max-pooling 계산을 수행하고, 별도의 return 없이 바로 poolRes 의 올바른 위치에 결과값을 저장한다. Pworker 도 결과 배열을 output file 에 write 하기 전에 Cworker 때처럼 Pworker 의 모든 작업이 다 끝났다는 보장을 해주어야 한다. 따라서 remained_pworkers 를 사용하며, Pworker 에서 이 변수가 가리키는 값을 1 감소시키고 종료한다. 이는 다른 스레드들과도 공유하는 변수이기 때문에, 앞뒤로 mutex lock & unlock 을 걸어줌으로써 race condition 을 방지한다. 이와 같이 Pworker 의 종료가 보장되지 않고 바로 write 를 할 경우, 결과 배열상에서 구멍이 존재하게 된다.

- **Implementation :** **void** *pooling(**void*** **argument**)

- **Pworkers Maker Design**

Pworker 도 pthread_create 로 생성되고, 바로 해당 스레드에 대하여 pthread_detach 를 호출함으로써, 종료된 스레드에 대하여 자동으로 자원해제 해줌으로써 제한없이 스레드가 생성될 수 있게 해준다.

- **Implementation :** **void** make_pworkers(**void*** (***pooling**)(**void***), **ponv_arg*** pargs, **int** pwork_count, **int*** remained_pworkers, **info*** info)

3. output file 에 결과를 write 하기

- **Design :** 최종 결과 배열은 숫자로 존재하기 때문에 write 를 하기 전에 sprintf 를 이용하여 문자열 타입으로 변경해주어야한다. 또한 한 원소마다 4Byte 로 나타내야 하기 때문에, 4 에서 문자열 길이를 뺀 만큼 공백을 먼저 입력해야한다.

- **Implementation :** **void** write_result(**int convResSize**, **int**** **poolRes**, **int outfd**)

Function description

Parallel CNN 은 아래의 함수 순서대로 호출되어 진행된다.

Function Name	Arguments	Desciption
get_inputsize	infd : int	input file descriptor
	Return Value	input matrix 의 크기 (int)

Function Name	Arguments	Desciption
convolution	argument : void*	conv_arg 타입의 동적할당 구조체를 가리키는 포인터
	Return Value : void	-

Function Name	Arguments	Desciption
make_cworkers	void *convolution(void*)	Convolution 을 수행하는 cworker 의 스레드 함수로 사용되며, 함수포인터로 매개변수를 받는다.
	cargs : conv_arg*	각 스레드 함수에 주어지는 인자를 담고 있는 배열
	cwork_count : int	필요한 cworker 의 개수
	remained_cworkers : int*	남아있는 cworker 의 개수
	info : info*	cworker 와 pworker 에서 사용하는 공통 정보
	Return Value : void	-

Function Name	Arguments	Desciption
pooling	argument : void*	pool_arg 타입의 동적할당 구조체를 가리키는 포인터

	Return Value : void	-

Function Name	Arguments	Desciption
make_pworkers	void* (*pooling)(void*)	Max-pooling 을 수행하는 pworker 의 스레드 함수로 사용되며, 함수포인터로 매개변수를 받는다.
	pargs : pool_arg*	각 스레드 함수에 주어지는 인자를 담고 있는 배열
	pwork_count : int	필요한 pworker 의 개수
	remained_pworkers : int*	남아있는 pworker 의 개수
	info : info*	cworker 와 pworker 에서 사용하는 공통 정보
	Return Value : void	-

Function Name	Arguments	Desciption
write_result	convResSize : int	convolution 결과의 크기이며, 최종 결과 크기가 convResSize/2 이기 때문에, 이 크기만큼 이중 for 문으로 write 를 수행한다.
	poolRes : int**	pooling 까지 마친 최종 결과 행렬을 가리키는 포인터 변수
	outfd : int	output file descriptor
	Return Value : void	-