MPI Programming for Mandelbrot set

Yun, Junhyeog

# Goal

* Display width = 400, Display height = 400인 Mandelbrot set을 column 방향으로 Display width를 N\_rectangle(=40)개의 block으로 나누어서 계산하고 출력하는 프로그램을 제작한다.
* Static task assignment, dynamic task assignment로 동작하는 2가지 프로그램을 제작한다.
* 각 프로그램은 1, 4, 8개의 process들을 사용하여 실행된다.

# Static task assignment1 – master도 계산에 사용됨

## Implement

1. Structure and global variables

텍스트이(가) 표시된 사진

자동 생성된 설명

Mandelbrot Set을 계산하기 위한 복소수 structure인 complex와 Mandelbrot Set의 좌표와 color값을 한번에 저장하는 structure인 mand를 선언하였다.

텍스트이(가) 표시된 사진

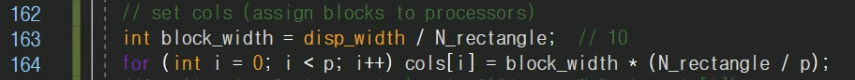
자동 생성된 설명

과제에 제시된 Example of output과 유사한 모양의 Mandelbrot Set을 계산하기 위해 real\_min, real\_max, imag\_min, image\_max 값을 위와 같이 설정하였다. 또한 과제에 제시된 이미지의 크기인 400 \* 400을 disp\_width, disp\_height로 설정해주었다. 과제에서 Display width를 N\_rectangle개의 block으로 나누어서 병렬 계산을 해야 한다고 제시되어 있으므로, N\_rectangle을 정의해주었다. arr은 계산된 Mandelbrot Set이 저장될 공간이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

P는 processor의 개수이며, command-line argument를 통해서 정해진다. Cols는 400개의 col 중에서 각 processor가 몇 개의 col을 맡아서 처리할 것인지를 미리 계산하여 저장할 공간이다. 예를 들어 processor의 개수가 1이면 0번 processor가 모든 block들을 계산할 것이고, processor의 개수가 4이면 각각 10개씩의 block(100개 row)씩 계산하게 된다. P와 Cols를 계산하는 코드는 아래와 같다.





MPI\_Mand는 앞서 정의한 mand structure를 MPI\_Datatype으로 표현한 것이다. 아래와 같은 명령어로 정의가 가능하다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. Bmp 이미지

계산한 Mandelbrot Set을 bmp 이미지로 저장하기 위해 color라는 structure를 정의했고, save라는 함수를 정의했다. color는 color를 의미하는 구조체이고, save함수는 arr 배열에 저장한 값들을 이용해서 bmp 이미지를 저장하는 함수이다. 다만, 중요한 부분이 아니기에 자세한 설명은 생략하도록 하겠다.

1. Cal\_pixel(complex c) 함수

텍스트이(가) 표시된 사진

자동 생성된 설명

Cal\_pixel 함수는 복수수 좌표를 인자로 받아서 color값을 계산하는 함수이다. 이는 수업자료에 있는 함수를 사용하였으며, color 값을 0~255로 바꾸기 위해 return하기 전에 -1을 해주었다.

1. Display 함수

텍스트이(가) 표시된 사진

자동 생성된 설명

위 함수는 root processor가 받은 혹은 계산한 color값들을 arr배열에 저장하는 함수이다. 계산 즉시 출력할 수 없기 때문에 arr 배열에 저장하는 것으로 출력이 되었다고 가정하였다.

1. Calc\_col\_and\_send 함수

텍스트이(가) 표시된 사진

자동 생성된 설명

위 함수는 block의 color값들을 계산하는 함수이다. Calc\_col\_and\_send 함수는 root processor에서 실행되는 함수이다. 계산해야 할 시작 col을 col 매개변수를 통해 전달 받고, 시작 col부터 몇 개의 col을 계산해야 하는지는 자신의 rank를 이용해서 cols 배열에 저장된 값을 통해 알아내고, 그만큼 color 값을 계산한 다음, display 함수 혹은 MPI\_Send 함수를 실행한다. Root processor이면 MPI\_Send를 할 필요없고, 반대로 root processor가 아닌 slave에서는 결과를 계산하고 그 결과를 root processor, master에게 send해야 한다.

1. Master 함수

텍스트이(가) 표시된 사진

자동 생성된 설명

master함수는 master에서 실행되는 함수로, 먼저 slave들에게 각자 계산해야 할 시작 block을 send한다. 그 다음, 자신에게 할당된 block들을 계산하고, 마지막으로 slave들이 계산하고 master에게 보낸 결과를 MPI\_Recv함수를 이용해서 받고, display한다.

1. Slave함수

텍스트이(가) 표시된 사진

자동 생성된 설명

slave함수는 slave processor들에서 실행되는 함수이다. master로부터 계산해야 할 시작 block을 MPI\_Recv 함수로 전달받고, 그에 해당하는 block들을 계산하고 master에게 결과를 전송한다.

1. 실행방법



위와 같은 명령어를 통해서 프로그램을 실행할 수 있다. -n 옵션은 사용할 processor의 수를 의미한다.

## Result

1. 1 Processor

텍스트이(가) 표시된 사진

자동 생성된 설명

위는 1개의 processor를 사용하여 동작한 stdout의 screendump이다. 40개의 block이 모두 master processor(0)에서 계산되었으며, 전체 runtime은 0.052561초이다.

텍스트, 하얀색, 옅은, 어두운이(가) 표시된 사진

자동 생성된 설명

저장된 bmp 이미지 파일은 위와 같다. 5번 실행한 결과는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 processor | 1 | 2 | 3 | 4 | 5 | Mean |
| Runtime(sec) | 0.052561 | 0.052951 | 0.053560 | 0.052305 | 0.052607 | 0.0527968 |

1. 4 processors

텍스트이(가) 표시된 사진

자동 생성된 설명

위는 4개의 processor를 사용하여 동작한 stdout의 screendump이다. 40개의 block이 각 processor에서 10개씩 계산되었으며, 전체 runtime은 0.046886초이다.

텍스트, 하얀색, 옅은, 어두운이(가) 표시된 사진

자동 생성된 설명

저장된 bmp 이미지 파일은 위와 같다. 5번 실행한 결과는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 processors | 1 | 2 | 3 | 4 | 5 | Mean |
| Runtime(sec) | 0.046886 | 0.044917 | 0.0426966 | 0.043014 | 0.041001 | 0.04370292 |

하나의 processor만 사용했을 때보다 실행시간이 줄어든 것을 확인할 수 있다. 하지만 1/4로는 줄지 않았고 약간의 시간 단축이 있음을 확인할 수 있다.

1. 8 processors

텍스트이(가) 표시된 사진

자동 생성된 설명

위는 8개의 processor를 사용하여 동작한 stdout의 screendump이다. 40개의 block이 각 processor에서 5개씩 계산되었으며, 전체 runtime은 0.059583초이다.

텍스트, 하얀색, 옅은, 어두운이(가) 표시된 사진

자동 생성된 설명

저장된 bmp 이미지 파일은 위와 같다. 5번 실행한 결과는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 8 processors | 1 | 2 | 3 | 4 | 5 | Mean |
| Runtime(sec) | 0.059583 | 0.058959 | 0.056312 | 0.052397 | 0.056521 | 0.0567544 |

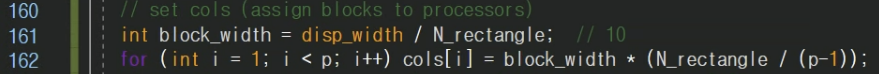
1. 해석

아무래도 400\*400 크기의 Mandelbrot Set을 계산하는 computation 시간이 매우 작아서 communication 시간에 전체 runtime 시간이 영향을 크게 받는다. 따라서 processor의 수가 1에서 4로 증가할 때 실행시간이 소폭 감소한 것이 communication에 드는 시간이 증가한 것보다 computation 시간이 조금 더 줄어들었다고 해석할 수 있으며, 1에서 8로 증가했을 때 시간이 더 증가한 것은 computation 시간이 줄어든 것보다 communication 시간이 훨씬 많이 증가해서 오히려 실행시간이 증가했다고 해석할 수 있다.

# Static task assignment2 – master는 계산에 참여하지 않음

## Implement

1. Static task assignment1과의 차이점



텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같이 cols 배열을 선언하는 부분과 master 함수를 약간씩 수정해주면 master는 할당만 하고 1, 4, 8개의 slave processor만 계산에 참여하도록 할 수 있다.

## Result

1. 1 Processor

텍스트이(가) 표시된 사진

자동 생성된 설명

위는 1개의 slave processor를 사용하여 동작한 stdout의 screendump이다. 40개의 block이 모두 하나의 slave processor(1)에서 계산되었으며, 전체 runtime은 0.091363초이다.

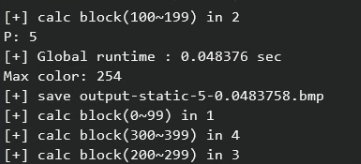
텍스트, 하얀색, 옅은, 어두운이(가) 표시된 사진

자동 생성된 설명

저장된 bmp 이미지 파일은 위와 같다. 5번 실행한 결과는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 processor | 1 | 2 | 3 | 4 | 5 | Mean |
| Runtime(sec) | 0.091363 | 0.0835743 | 0.0891086 | 0.0845438 | 0.0867351 | 0.08706496 |

1. 4 processors



위는 4개의 slave processor를 사용하여 동작한 stdout의 screendump이다. 40개의 block이 각 processor에서 10개씩 계산되었으며, 전체 runtime은 0.048376초이다.

텍스트, 하얀색, 옅은, 어두운이(가) 표시된 사진

자동 생성된 설명

저장된 bmp 이미지 파일은 위와 같다. 5번 실행한 결과는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 processors | 1 | 2 | 3 | 4 | 5 | Mean |
| Runtime(sec) | 0.048376 | 0.050321 | 0.048753 | 0.046538 | 0.047102 | 0.048218 |

하나의 processor만 사용했을 때보다 실행시간이 줄어든 것을 확인할 수 있다. 하지만 1/4로는 줄지 않았고 대략 절반 가까이의 시간 단축이 있음을 확인할 수 있다.

1. 8 processors

텍스트이(가) 표시된 사진

자동 생성된 설명

위는 8개의 slave processor를 사용하여 동작한 stdout의 screendump이다. 40개의 block이 각 processor에서 5개씩 계산되었으며, 전체 runtime은 0.061864초이다.

텍스트, 하얀색, 옅은, 어두운이(가) 표시된 사진

자동 생성된 설명

저장된 bmp 이미지 파일은 위와 같다. 5번 실행한 결과는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 8 processors | 1 | 2 | 3 | 4 | 5 | Mean |
| Runtime(sec) | 0.061864 | 0.060816 | 0.072267 | 0.075925 | 0.067341 | 0.0676426 |

다시 1개의 slave processor만 사용했을 때보다는 아니지만, 약간 실행 시간이 증가했다.

1. 해석

앞선 static task assignment1에서와의 같은 해석이 가능하다. 다만 master가 계산을 맡지 않았으므로 communication 시간이 대폭 줄어서 1개의 processor를 사용했을 때보다는 여러 processor를 사용했을 때 실행시간이 감소함을 알 수 있다.

# Dynamic task assignment

## Implement

1. calc\_block\_and\_send

텍스트이(가) 표시된 사진

자동 생성된 설명

Dynamic task assignment에서는 static에서와는 다르게 calc\_block\_and\_send 함수를 사용한다. 해당 함수는 col에서 시작하는 하나의 block을 계산한 후 한번에 MPI\_Send 함수를 이용하여 master에게 보내는 함수이다.

1. Master 함수

텍스트이(가) 표시된 사진

자동 생성된 설명

Master 함수에서는 먼저 자신을 제외한 slave processor들에게 처음 계산할 block을 할당해주고, slave processor가 계산한 block의 결과를 받을 때마다 계산할 block이 남아있으면 해당 slave에게 새로운 block을 할당해준다. 더 이상 계산할 block이 남아있지 않다면, terminator\_tag를 보내 slave에게 알린다.

1. Slave 함수

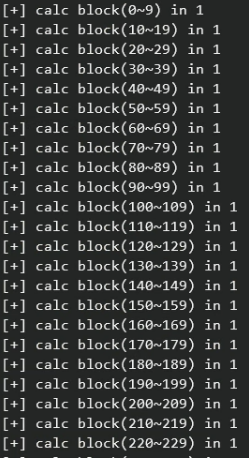
텍스트이(가) 표시된 사진

자동 생성된 설명

slave함수에서는 terminator\_tag가 오기 전까지 master로부터 할당된 block을 계산하고, 그 결과를 master에게 전송한다.

## Result

1. 1 slave processor

텍스트이(가) 표시된 사진

자동 생성된 설명

위는 1개의 slave processor를 사용하여 동작한 stdout의 screendump이다. 40개의 block이 모두 하나의 slave processor(1)에서 계산되었으며, 전체 runtime은 0.054436초이다.

텍스트, 하얀색, 옅은, 어두운이(가) 표시된 사진

자동 생성된 설명

저장된 bmp 이미지 파일은 위와 같다. 5번 실행한 결과는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 processor | 1 | 2 | 3 | 4 | 5 | Mean |
| Runtime(sec) | 0.054436 | 0.056234 | 0.054123 | 0.055430 | 0.056123 | 0.0552592 |

1. 4 slave processors

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

위는 4개의 slave processor를 사용하여 동작한 stdout의 screendump이다. 40개의 block이 각 processor에서 11, 11, 10, 8개씩 계산되었으며, 전체 runtime은 0.013910초이다.

텍스트, 하얀색, 옅은, 어두운이(가) 표시된 사진

자동 생성된 설명

저장된 bmp 이미지 파일은 위와 같다. 5번 실행한 결과는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 processors | 1 | 2 | 3 | 4 | 5 | Mean |
| Runtime(sec) | 0.013910 | 0.014225 | 0.013843 | 0.014124 | 0.013944 | 0.0140092 |

하나의 processor만 사용했을 때보다 실행시간이 대폭 줄어든 것을 확인할 수 있다. 대략 1/4 정도로 줄어들었다. Master가 계산에 참여하지 않고 communication을 전담하여 나머지 4 slave processor가 대부분의 시간을 computation에 사용되었기 때문에 상당한 성능 향상이 이루어졌다.

1. 8 slave processors

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

위는 8개의 slave processor를 사용하여 동작한 stdout의 screendump이다. 40개의 block이 각 processor에서 6, 4, 4, 6, 5, 4, 7, 4개씩 계산되었으며, 전체 runtime은 0.008457초이다.

텍스트, 하얀색, 옅은, 어두운이(가) 표시된 사진

자동 생성된 설명

저장된 bmp 이미지 파일은 위와 같다. 5번 실행한 결과는 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 processors | 1 | 2 | 3 | 4 | 5 | Mean |
| Runtime(sec) | 0.008457 | 0.008255 | 0.008241 | 0.008317 | 0.008569 | 0.0083678 |

4개의 slave processor만 사용했을 때보다 실행시간이 대폭 줄어든 것을 확인할 수 있다. 대략 1/2 정도로 줄어들었다. 4개의 slave를 사용하였을 때와 유사한 이유로, 상당한 성능 향상이 이루어졌다.