**Big Data processing**

**Assignment1 Report**

텍스트, 폰트, 로고, 등록 상표이(가) 표시된 사진

자동 생성된 설명

**32217072 Mobile System Engineering 김도익**

**Big Data Processing Assignment 1**

**목차**

**1. Instruction**

**2. 배경지식**

**2.1) MPI**

**2.2) 사다리꼴적분법**

**3. 직렬 vs 병렬 실행 시간 비교**

**4. 확장성 테스트**

**5. 정확도 테스트**

**6. 결론**

**1. Instruction**

병렬 프로그램이란 하나의 데이터나 작업을 여러 개로 나누어 여러 개의 프로세스가 연산을 처리하는 프로그램이다. 이 과정에서 각 프로세스가 독립적으로 작업을 수행하고, 필요한 경우 프로세스 간 통신을 통해 정보를 교환하여 작업을 완료한다. MPI (Message Passing Interface)는 이러한 병렬 연산 시 프로세스 간 정보 교환을 위한 통신 표준으로, 분산 메모리 환경에서 사용된다. MPI를 사용하면 다양한 하드웨어와 운영체제 환경에서 병렬 프로그램을 구현할 수 있고, 성능 향상을 위해 프로세스 간 효율적 데이터 전달이 가능하다.

이 레포트에서는 사다리꼴 적분법을 이용하여 주어진 함수를 직렬과 병렬로 처리하고, 각 방식의 결과 값과 실행 시간을 측정하여 비교 분석한다. 특히, 문제의 복잡도와 문제 분할 개수 및 프로세스 개수에 따른 성능 변화를 살펴보며, 병렬화가 성능에 미치는 영향을 정량적으로 평가한다. 또한, 실행 시간의 변동과 스피드업을 통해 MPI 병렬 프로그램의 효율성을 검토하고, 병렬화의 장단점을 제시한다.

**2. 배경지식**

본 레포트에서 다루는 내용을 이해하기 위해 필요한 배경 지식으로 MPI와 사다리꼴 적분법에 대해 설명한다.

**2.1) MPI (Message Passing Interface)**

MPI는 병렬 프로그래밍을 위해 개발된 표준화된 메시지 전달 인터페이스로, 여러 프로세스 간에 데이터를 주고받을 수 있게 해주는 기법이다. 고성능 컴퓨터에서 주로 사용되며, 특히 분산 메모리 시스템에서 효과적이다. MPI는 프로세스들 간 직접적인 메시지 전달을 가능하게 하여 데이터 교환을 효율적으로 처리할 수 있게 한다. 이로써 분산된 환경에서도 병렬 연산을 수행할 수 있다. 또한 MPI는 데이터를 여러 프로세스에 분산시키고, 각 프로세스에서 계산된 결과를 하나로 집계할 수 있게 해준다. 이는 큰 데이터셋을 병렬로 처리할 때 연산 량을 줄여준다.

MPI의 단점으로는 복잡한 프로그래밍, 메모리 제한, 확장성 문제, 그리고 장비 의존성이 있다. 다수의 프로세스가 동시에 통신하는 구조를 다루다 보니 프로그래밍과 디버깅이 복잡하고, 프로세스 간 동기화나 데이터 전송 순서에 따라 오류가 발생할 수 있다. 또한, 분산 메모리 환경에서는 각 프로세스가 자체 메모리만 사용할 수 있어 대규모 데이터셋 처리에 제약이 있으며, 매우 큰 클러스터나 복잡한 네트워크 환경에서 통신 오버헤드가 발생할 수 있다. 마지막으로, MPI는 사용되는 하드웨어(CPU, 메모리)나 네트워크 속도, 안정성 등에 의존적이기 때문에 환경에 따라 성능이 크게 달라질 수 있다.

따라서 MPI를 효율적으로 사용하기 위해서는 그 장단점을 정확히 이해하고, 작업 크기와 시스템 환경에 맞춰 적절히 적용하는 것이 중요하다. 적절히 사용하면, 계산 속도를 크게 향상시킬 수 있다. 특히, 데이터 전송과 동기화 구조를 신중하게 설계함으로써 오류 발생 가능성을 줄이고 성능을 극대화할 수 있다. 상황에 맞는 MPI의 활용이 병렬 처리의 성공적인 구현을 좌우한다.

**2.2) 사다리꼴 적분법**

함수 f(x)의 면적을 구하기위해서는 아래와 같은 적분을 사용하면 된다. 하지만 컴퓨터로 이런 연산을 학게되면 굉장히 어렵다.

따라서 본 레포트에서 분할되는 데이터의 수나, 프로세스에 따라, 성능이나 정확도가 어떻게 변하는지를 사다리꼴 적분법에 MPI를 사용하여 알아보도록한다. //

함수 f(x)f(x)f(x)의 면적을 계산하기 위해 일반적으로 다음과 같은 적분을 사용한다:

∫abf(x) dx\int\_a^b f(x) \, dx∫ab​f(x)dx

그러나 이러한 연산을 컴퓨터로 직접 수행하는 것은 계산 비용이 높고 복잡하기 때문에 효율적인 수치적 방법이 필요하다. 이때 **사다리꼴 적분법**은 함수의 곡선을 여러 개의 사다리꼴로 근사하여 면적을 계산하는 유용한 방법이다.

사다리꼴 적분법의 공식은 다음과 같다. 구간 [a,b][a, b][a,b]를 nnn개의 작은 구간으로 나누었을 때, 각 구간의 폭을 h=b−anh = \frac{b - a}{n}h=nb−a​로 정의한다. 이때 전체 적분 값은 다음과 같은 식으로 근사할 수 있다:

∫abf(x) dx≈h2(f(a)+2∑i=1n−1f(a+ih)+f(b))\int\_a^b f(x) \, dx \approx \frac{h}{2} \left( f(a) + 2 \sum\_{i=1}^{n-1} f(a + ih) + f(b) \right)∫ab​f(x)dx≈2h​(f(a)+2i=1∑n−1​f(a+ih)+f(b))

이 방법에서는 구간을 일정한 간격으로 나누고, 각 구간의 양 끝점에서 함수 값을 구하여 사다리꼴의 면적을 합산함으로써 전체 면적을 추정한다.

본 보고서에서는 사다리꼴 적분법에 **MPI**(Message Passing Interface)를 적용하여 병렬 처리를 수행함으로써 계산 속도와 정확도를 높이고자 한다. 데이터를 여러 프로세스로 분할하여 작업을 수행하면 연산 성능과 정확도가 어떻게 변하는지를 평가할 수 있다. 이를 통해 분할 구간의 수와 프로세스 수에 따른 성능 변화를 분석하고, 사다리꼴 적분법을 효율적으로 병렬화하는 방법을 모색한다.

**3. 직렬 vs 병렬 실행 시간 비교**

직렬과 병렬 프로그램의 사다리꼴의 개수를 4096개로 설정하여 그 실행 시간을 측정하고 결과를 비교한다. 직렬 프로그램은 프로세스를 1개를 사용하여 직렬 처리할 수 있도록 하였다.

**3.1) [0, 2]**

프로세스 개수가 1개인 직렬 실행시간과 적분 값은 다음과 같다.

**텍스트, 스크린샷, 폰트, 그래픽이(가) 표시된 사진

자동 생성된 설명**

실행시간은 0.000013초이고 적분 값은 2.666666746139526e+00 이다.

프로세스 개수가 4개인 병렬 실행시간과 적분 값은 다음과 같다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

실행시간은 0.000098초이고 적분 값은 2.666666746139526e+00 이다.

의 경우 직렬과 병렬 실행 모두 결과값이 2.666666746139526e+00으로 같은 것으로 보아 간단한 식의 경우, 직렬과 병렬 실행의 정확도는 차이가 없다는 것을 확인하였다. 또, 직렬의 실행시간은 0.000010 초, 병렬 실행시간은 0.000147초로, 직렬 실행이 병렬 실행보다 성능이 높음을 보인다. 이를 보았을 때, 와 같은 간단한 계산인 경우, 병렬화로 인한 통신 오버헤드 때문에 병렬로 처리하는 것이 효율적이지 않다는 것을 알 수 있다.

**3.2) [0, 2]**

직렬 실행은 다음과 같다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

실행시간은 0.000150 초이고 적분 값은 8. 8.400000397364433e+00 이다.

병렬 실행은 다음과 같다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

실행시간은 0.000116초이고, 적분 값은 8.400000397364245e+00 이다.

함수 의 경우, 병렬 실행 시간이 직렬 실행 시간보다 더 짧게 나타나는 것을 확인할 수 있었다. 이는 계산 복잡도가 높은 함수일수록 병렬 처리의 이점이 두드러지게 나타난다고 볼 수 있다. 또한, 병렬 실행으로 계산한 결과 값이 직렬 실행으로 계산한 값보다 정확한 값인 8.4에 조금 더 가까워지는 경향이 있음을 확인하였다.

따라서, 함수가 복잡할 때, 병렬 처리를 통해 성능뿐만 아니라, 정확도 면에서도 유의미한 향상이 있음을 확인할 수 있었다. 이것은 많은 연산이 필요한 문제에서 병렬 처리가 가지는 이점을 볼 수 있다.

**4. 확장성 테스트**

확장성 테스트 섹션에서는 두 함수에 대해 사다리꼴 MPI 프로그램을 실행한다. MPI 프로그램에서 프로세스의 개수를 1, 2, 4, 8개로 테스트하고, 사다리꼴의 개수를 256, 1024, 4096, 16384개로 테스트하여 성능과 정확도를 비교한다.

**4.1) [0, 2]**

함수에서 사다리 적분법에서 프로세스의 개수와 사다리꼴의 개수에 따라 평가한 speedup과 efficiency이다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | p | 1 | 2 | 4 | 8 |
| 256 | S | 1 | 0.068 | 0.062 | 0.017 |
|  | E | 1 | 0.034 | 0.0155 | 0.002125 |
| 1024 | S | 1 | 0.23 | 0.18 | 0.052 |
|  | E | 1 | 0.115 | 0.045 | 0.0065 |
| 4096 | S | 1 | 0.076 | 0.65 | 0.196 |
|  | E | 1 | 0.038 | 0.1625 | 0.0245 |
| 16384 | S | 1 | 0.336 | 0.3 | 0.40 |
|  | E | 1 | 0.018 | 0.075 | 0.05 |

**4.2) [0, 2]**

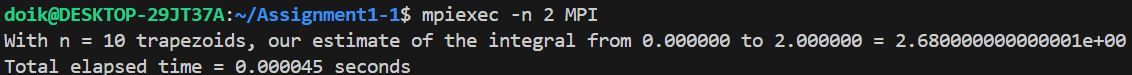
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | p | 1 | 2 | 4 | 8 |
| 256 | S | 1 | 0.15 | 0.24 | 0.01 |
|  | E | 1 | 0.075 | 0.06 | 0.00125 |
| 1024 | S | 1 | 0.35 | 0.19 | 0.12 |
|  | E | 1 | 0.175 | 0.0475 | 0.085 |
| 4096 | S | 1 | 0.73 | 0.77 | 0.77 |
|  | E | 1 | 0.365 | 0.1925 | 0.1925 |
| 16384 | S | 1 | 1.4 | 2.22 | 1.5 |
|  | E | 1 | 0.7 | 0.555 | 0.1875 |

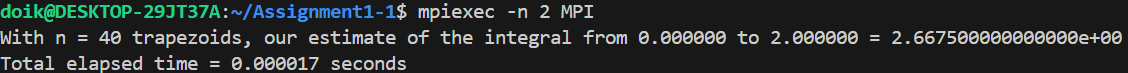
**4. 정확도 테스트**

와함수에 대해 2개의 프로세스를 사용하여 사다리꼴의 개수를 10, 40, 160, 640개로 늘려가며 적분 값을 실제 정확한 값과 비교한다.

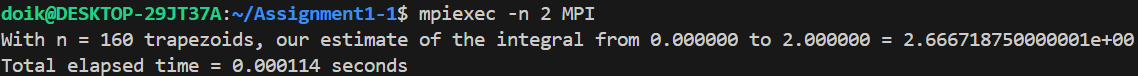
**4.1) [0, 2]**

N 이 10일 때 적분 값은 다음과 같다.

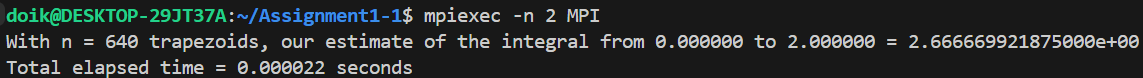


N 이 40일 때 적분 값은 다음과 같다. 

N 이 160일 때 적분 값은 다음과 같다.



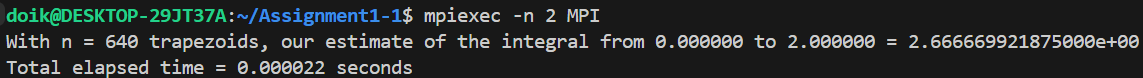
N 이 640일 때 적분 값은 다음과 같다.



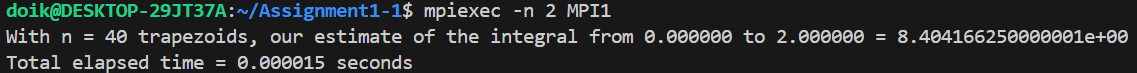
N이 640개 일 때 2.666669921875000e+00으로 가장 정확한 값인 2.6667 과 가장 근접한 값을 보인다.

**4.2) [0, 2]**

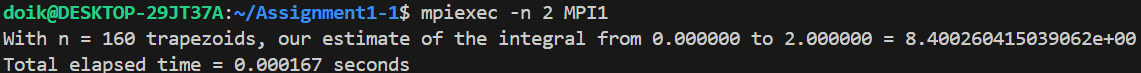
N 이 10일 때 적분 값은 다음과 같다.



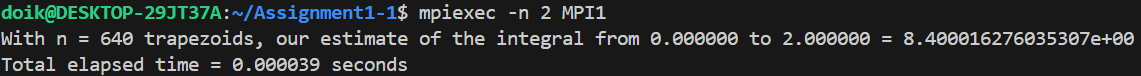
N 이 40일 때 적분 값은 다음과 같다.

****

N 이 160일 때 적분 값은 다음과 같다.

****

N 이 640일 때 적분 값은 다음과 같다.

****

N이 640일 때 정확한 값인 8.4과 가깝다.

사다리꼴 적분에서는 함수를 여러 개의 작은 사다리 꼴로 나눠 면적을 구하는데, 사다리꼴의 수가 늘어나면, 각 구간의 길이가 짧아져 함수 곡선과 직선으로 구성된 사다리꼴 간의 차이가 줄어든다. 이는 함수의 변화를 더 잘 반영하여 더 정확한 면적을 구할 수 있다. 따라서 n이 증가함에 따라 실제 면적에 더 가깝게 수렴하는 것을 볼 수 있다.

**6. 결론**

이렇게 사다리꼴 적분 법 문제를 사용하여 사다리꼴 하나의 연산을 MPI기법을 사용하여 여러 프로세스에서 병렬처리를 실행해보고 그 결과를 분석하여 성능 과 정확도를 분석하였다