MPI를 활용한 PI(π)값 계산 병렬화 알고리즘

최 민[O] 맹승렬
한국과학기술원 전자전산학과 전산학전공
{mchoi[O], maeng}@kaist.ac.kr

Parallelized PI(π) Calculation Algorithm using MPI

Min Choi[O] SeungRyoul Maeng
Division of Computer Science, Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology

요 약

정확한 π 값의 계산은 자연과학의 여러 분야에 도움을 준다. 이와 같이 π 값을 계산하는 여러 가지 방법이 제안되어 있으며 널리 사용되고 있으나, 본 논문에서는 MPI 라이브러리를 활용한 π 값 계산의 병렬화 알고리즘을 소개한다. $\tan^{-1}(x)$의 정의를 이용하는 π 값 계산 방법은 다항식의 계산과정에서 각 항(term)들의 종속성으로 인하여 병렬화 수행이 힘든 단점이 있다. 본 논문에서는 $\tan^{-1}(x)$를 맥클로린 수열(Maclaurin Series)을 통하여 다항함수로 표현하고, 병렬화 수행에 적합한 적분형태로 변형한다. 따라서, MPI 환경에서 수행하기 적합한 π 값 계산의 병렬화 알고리즘을 제안하고 8노드 클러스터 환경에서 성능을 비교해본다. 또한, 직렬화된 방법에 대한 성능향상(speedup)을 측정한다.

1. 서 론

π 는 지름의 길이에 대한 원의둘레(길이)의 비의 값(비율), 즉 원주율이라고 한다. 따라서, 모든 원에 대하서 원주율은 항상 일정하다. 원주율은 원의 넓이, 부피 등에 대한 계산을 하고자 할 때 반드시 필요한 값이다. 따라서, 정확한 π 값을 구하려는 많은 시도와 노력이 있었고, 심지어는 여기에 사용된 방법을 통해 문명의 발달 정도를 가늠할 수 있다. 원주율을 구하는 가장 간단한 방법은 원주/지름으로 구하면 되는데, 이 때에는 곡선인 원주의 길이를 정확히 측정하기 힘들다는 단점이 있다. 따라서, π 값을 구하는 다양한 방법(다각형법 등)들이 고대 시대부터 제안되었으며, 근래에 들어서는 π 를 무한급수와 공식을 이용해 다각형법을 사용하는 것보다 쉽게 계산할 수 있게 되었다. 현대에는 컴퓨터를 사용하여 충분한 시간과 자원이 뒷받침 되는 경우 소수점 아래 22억 6백 32만 1천 3백 36자리까지 계산할 수 있다. 본 논문에서는 클러스터 환경에서 널리 활용되는 (MPI)Message Passing Interface를 활용한 π 값 계산 병렬화 알고리즘을 제안하고 성능을 분석한다.

2. 무한급수를 이용한 π 값 계산

무한급수를 이용하는 방법은 다각형법을 이용하는 것보다 더 쉽고 정확하게 π 값을 계산할 수 있다. 이를 위해서는 우선 arctan을 맥클로린 수열에 의한 표현으로 나타내야 한다.

$$P_n(x) = f(c) + f'(c)(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \cdots + \frac{f^{(n)}(c)}{n!}(x-c)^n$$

위와 같은 테일러 수열(Taylor Series)의 정의에 대해서 특히, $c=0$ 일 때의 테일러 수열 $\sum_{k=0}^{\infty}\frac{f^k(0)}{k!}$ 를 함수 f 의 맥클로린 수열(Maclaurin Series)이라고 한다. 따라서, 이 식에서 필요로 하는 $f(x)=\tan^{-1}(x)$ 의 n계도함수들을 구해서 맥클로린 수열에 적용하여 정리하면, 다음과 같이 된다.

$$P_n(x) = 0 + x + 0 - \frac{x^3}{3} + 0 + \frac{x^5}{5} + \cdots = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$$

최종적으로 $P_n(x)=\tan^{-1}(x)$ 이므로

$$\tan^{-1}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots + (-1)^n \frac{x^{2n+1}}{2n+1} + \ldots \quad [1]$$

와 같이 $\tan^{-1}(x)$를 맥클로린 수열을 통하여 다항함수로 표현할 수 있다. 즉, π 값을 계산하는데 있어 위의 맥클로린 수열을 풀어서 $\tan^{-1}(x)$ 의 값을 구할 수 있다.

3. Motivation

그러나 본 논문에서는 MPI를 활용한 병렬수행을 통해 π 값 계산의 성능을 향상시키고자 하므로, 이러한 방법은 적합하

지 않다. 앞의 무한급수에 의한 표현은 8개의 node(혹은 processing element)에서 작업을 수행하고자 할 때 바로 이전 iteration에서 변경된 데이터가 다음 번 iteration에서 사용되는 순차적으로 종속(sequentially dependent)인 관계가 있다. 즉, 앞 항(term)의 계산결과를 반드시 알아야만 그 다음 항의 계산을 수행하여 전체를 구할 수 있다. 따라서, 클러스터 컴퓨팅 환경에서 아무리 많은 수의 node가 존재한다 할지라도 결국은 serialized된 수행이 되는 것이다. 예를들어, $\tan^{-1}(1)=\frac{\pi}{4}$ 를 계산하고자 할 때 $\tan^{-1}(1)=1-\frac{1^3}{3}+\frac{1^5}{5}-\frac{1^7}{7}+\cdots$ 의 값을 계속해서 구해야 하는데 앞에서 계산한 $1-\frac{1^3}{3}+\frac{1^5}{5}$ 을 알아야 이것에 $-\frac{x^7}{7}$ 을 계산하여 더해줄 수 있다. 따라서, MPI를 사용한 π 값 계산을 위해서는 병렬화 수행가능한 알고리즘이 제안되어야할 필요가 있다.

4. MPI를 활용한 병렬화 수행에 적합한 π 계산 알고리즘

본 논문에서는 약간의 수정을 통해 $\tan^{-1}(x)$ 함수의 급수 표현을 적분표현으로 바꾸어 병렬수행에 매우 적합한 형태로 변경하는 알고리즘을 제안한다. 그 과정은 아래와 같다.

앞의 (1)에 대하여 식의 양변을 미분하면,

$$\frac{d}{dx}\tan^{-1}(x)=1-x^2+x^4-x^6+\cdots\cdots+(-1)^n x^{2n}+\cdots\cdots$$

의 형태가 되고 편의상 x 변수를 t변수로 변경한다.

$$\frac{d}{dt}\tan^{-1}(t)=1-t^2+t^4-t^6+\cdots\cdots+(-1)^n t^{2n}+\cdots\cdots \quad [2]$$

이때 $\frac{1}{1+t^2}=1-t^2+t^4-t^6+\cdots\cdots+(-1)^n t^{2n}+\cdots\cdots$

이므로 [2]식을 다시 쓰면 아래와 같이 나타낼 수 있다.

$$\frac{d}{dt}\tan^{-1}(t)=\frac{1}{1+t^2}$$

이며 이를 다시 적분하면

$$\int_a^b \frac{d}{dt}\tan^{-1}(t)=\int_a^b \frac{1}{1+t^2} \qquad (\forall a,b\in R)$$

$$\tan^{-1}(t)=\int_a^b \frac{1}{1+t^2} \qquad (\forall a,b\in R) \qquad (3)$$

와 같은 최종적인 $\tan^{-1}(t)$ 에 대한 적분 표현이 나온다.

우리는 $\tan(\frac{\pi}{4})=1$ 임을 이미 알고 있고 양변에 arctan을 취하면 $\tan^{-1}(\tan(\frac{\pi}{4}))=\tan^{-1}(1)$ 을 통해 $\frac{\pi}{4}=\tan^{-1}(1)$ 같이 되며 최종적으로 위의 (3)식을 적용하여 아래와 같은 식이 유도된다.

$$\pi=4\tan^{-1}(1)=4\int_a^b \frac{1}{1+t^2}=\int_a^b \frac{4}{1+t^2} \qquad (\forall a,b\in R)$$

우리는 컴퓨터에서 적분연산을 하기 위해서 $-\frac{1}{2}$ 에서 $\frac{1}{2}$ 까지 구간 안의 합을 통해 approximate하게 구현 하도록 한다.

4.1 Decomposition

$\pi=\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{4}{1+t^2}$ 의 계산을 8개 노드에서 나누어 수행해야 한다. 앞에서의 맥클로린 수열과는 다르게 순차적으로 종속이지 않으므로 하지 않으므로 여기서는 병렬수행의 효과를 충분히 얻을 수 있다. 8개의 노드가 구간을 나누어 각각의 부분을 summation하고 이를 다시 reduction을 사용하여 totalsum에 통합하도록 한다. 이 때 PE(processing element)의 수가 8개 이므로 decomposition을 8의 배수가 되도록 수행하는 것이 효과적이다. 그렇지 않은 경우 8개 중 일부 PE는 나머지 task들을 처리해야 하는데 이렇게 되면 task들을 처리하지 않는 PE에게 idle하게 지연되는 시간이 존재한다.

4.2 Assignment

본 논문에서는 프로그램의 수행 전에 문제를 분석하여 정적으로 assignment를 행하는 정적 할당(static assignment) 방법을 사용한다. 정적 할당에서의 block assignment와 cyclic assignment의 두 가지 방법이 이 예제에서 역시 모두 적용 가능하다.

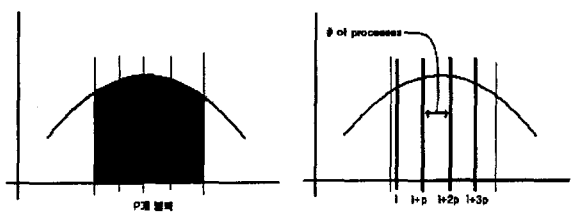

그림 1 static assignment

그림 1과 같이 P(number of processes)개 블록으로 이를 각각의 프로세스들에 할당하는 방법과 프로세스들 사이에 interleaved하게 할당되는 cyclic assignment방식이이 있다. 실제로, 프로세스 i는 i, i+p, i+2p 에 해당하는 x축 좌표

에 대해 계산하게 된다. 여기서는 그림1과 같은 cyclic assignment방법을 사용하도록 한다.

변수 이름	역할
numprocs	현재 processing element의 개수
N	구간을 몇 개의 조각으로 나누어 계산할 것인지 지정(현재 90000000개의 조각으로 나눔)
Myid	Processing element 중에서 master를 0으로 하고 나머지를 1부터 7까지의 ID를 할당함

표 1 파라미터 변수와 그 역할

Figure 2 를 통해 실행하고자 하는 것은 $x = \frac{1}{n} \times (i - \frac{1}{2})$ 통해서 x를 계산하고 이를 다시 식 (3)에 대입하여 $\tan^{-1}(x)$의 값을 얻고 이 값을 local sum에 일정 비율로 더해 놓는 것이다. 이렇게 더한 것이 모두 모이게 되면 구간 $-\frac{1}{2}$에서 $\frac{1}{2}$ 까지 적분한 결과에 근사하게 된다

```
h = 1.0 / (double) n;          // h = 1/n 계산
   sum = 0.0;                  // sum 초기화

   for (i = myid + 1; i <= n; i += numprocs) {
                               // cyclic assignment
        x = h * ((double)i - 0.5);
        sum += 4.0 / (1.0 + x*x);
   }
   mypi = h * sum;             // local sum을 할당된비
율                             // 만큼 반영
```

그림 2 MPI를 활용한 π 값 계산의 병렬수행 알고리즘

4.3 Orchestration

Message-Passing Model에서의 Orchestration은 explicit하게 send/receive primitives를 이용할 수도 있고 collective communication을 사용하여 할 수도 있다. 여기서는 후자의 방식인 collective communication을 사용하였다. MPI에서 제공되는 MPI_Bcast()함수를 사용하여 변수값 n을 모든 프로세스들에게 전달하였다. (이때 n=900000000이다.) 그 다음 병렬수행을 거쳐 각각의 프로세스들이 mypi 변수에 local sum을 저장해 놓으면 마지막으로 MPI_Reduce()함수를 사용하여 이들 값들이 operator MPI_SUM에 의해 모두 합쳐져서 두번째 변수 pi에 들어오게 된다.

5. 성능평가 및 결론

제안된 MPI를 활용한 π 값 계산의 병렬수행 알고리즘의 성능을 분석하기 위해서 본 논문에서는 π 값 계산을 수행하는 직렬수행 알고리즘과의 성능 비교를 수행한다. 실험환경은 8노드로 구성된 Linux 클러스터이며, 각각의 노드들은 100M Fast Ethernet으로 연결되어 있다.

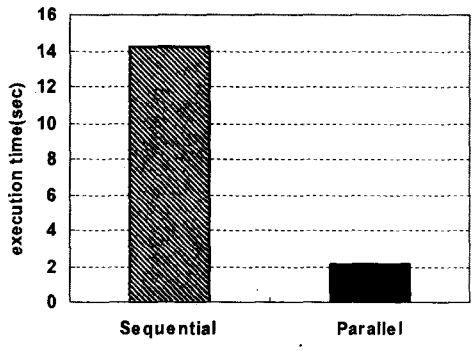

그림 3 MPI를 이용한 병렬화 알고리즘의 실행 시간

실험결과 MPI를 사용하여 병렬화된 알고리즘 2.1초, 직렬수행의 경우 14.2초로 6.7배의 성능향상(speedup)을 보여주었다.

$$Speedup_{fixedproblem}(8\,processors) = \frac{Performance(8\,processors)}{Performance(1\,processor)} = \frac{Time(1\,processor)}{Time(8\,processors)}$$

$$= \frac{14.2(sec)}{2.1(sec)} = 6.7619047619047619047619047619048$$

References

[1] P. B. Borwein, and D. H. Bailey. Ramanujan, Modular Equations, and Approximations to pi American Mathematical Monthly, vol. 96, no. 3 (March 1989), p. 201-220.

[2] D. H. Bailey, P. B. Borwein, and S. Plouffe. A New Formula for Picking off Pieces of Pi, Science News, v 148, p 279 (Oct 28, 1995).

[3] J.M. Borwein and P.B. Borwein. The arithmetic-geometric mean and fast computation of elementary functions. SIAM Review, Vol. 26, 1984, pp. 351-366.

[4] J.M. Borwein and P.B. Borwein. More quadratically converging algorithms for pi . Mathematics of Computation, Vol. 46, 1986, pp. 247-253.

[5] J. Stillwell, Mathematics and Its History, Chapter 8, 9, Springer

[6] D. Blatner, The Joy of π, Walker & Co, 1999

[7] W. Grop, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, 1994