공유 메모리 기반의 고성능 D-클래스 계산 병렬 알고리즘

신철규, 한재일
국민대학교 전산과학과
e-mail: M2004049@cs.kookmin.ac.kr

A High-performance Parallel Algorithm for D-Class Computation based on Shared Memory

Chul-Gyu Shin, Jae-Il Han
School of Computer Science, Kookmin University
e-mail: M2004049@cs.kookmin.ac.kr

요 약

n × n 불리언 행렬의 집합에서 동치관계를 이용하여 정의된 D-클래스는 개인키나 공개키 암호기술에 사용될 수 있는 가능성을 가지고 있다. 그러나 NP-완전 문제인 계산 복잡도로 인해 D-클래스의 효율적인 계산이 어려워 극히 제한된 크기의 행렬에 대한 D-클래스만이 알려져 있다. D-클래스를 효율적으로 계산하기 위해서는 수식변환, 병렬처리, 순환문 개선 등을 통해 알고리즘을 개선하여야 한다. 본 논문은 D-클래스의 효율적 계산을 위해 공유메모리 기반의 병렬 처리에 적합하도록 수식의 대수적 변환을 이용한 알고리즘의 설계와 실행 결과에 대해 논한다.

1. 서 론

D-클래스는 모든 n × n 불리언 행렬의 집합($M_n(F)$)에서 특정 관계(relation)에 따라 동치(equivalent) 관계에 있는 행렬의 집합으로 정의된다[1]. D-클래스는 개인키나 공개키와 같은 암호기술에 응용될 수 있는 가능성을 가지고 있으나, 계산 복잡도가 NP-완전 문제로서 현재 극히 제한된 크기의 행렬에 대한 D-클래스만이 알려져 있다[1].

D-클래스의 효율적 계산을 위한 알고리즘이 제시되었으나[2,3] 아직도 많은 개선이 필요하다. 본 논문은 D-클래스의 효율적 계산을 위해 공유메모리 기반의 병렬 처리에 적합하도록 수식을 대수적으로 변환하여 얻은 알고리즘의 설계와 실행 결과를 논한다. 본 논문의 구성은 다음과 같다. 2장은 관련연구에 대하여 논하며, 3장에서는 알고리즘의 설계 및 실행결과를 논하며, 4장에서는 결론 및 향후 과제를 기술한다.

2. 관련 연구

집합 F는 0과 1의 두 원소로 구성되며, $M_n(F)$는 행렬의 원소가 F의 원소를 값으로 갖는 모든 n × n 불리언 행렬의 집합이다. $M_n(F)$에 속한 임의의 행렬 A에 대한 D-클래스는 다음과 같이 정의된다.

[정의 1]

$$D_A = \{ B \in M_n(F) : \exists C, X, Y, U, V \in M_n(F) \text{ such that } AX = C,\ CY = A,\ UC = B,\ VB = C \}$$

m=n x n 일 때 $|M_n(F)|$는 2^m이 되고, $M_n(F)$ x $M_n(F)$ 연산은 $2^m \times 2^m$ 번의 행렬 곱셈을 필요로 한다. 그러나 $M_n(F)$에 속한 두 행렬의 곱셈 결과는 다시 $M_n(F)$에 속하게 된다. 이렇게 불리언 행렬의 연산 결과가 중복되는 것을 고려하여 실행시간을 개선시키기 위해 [정의 1]에서 VB = C 식을 CY = A 식에 대입하고 AX = C 식을 UC = B의 식에 대입하여 VBY = A 식과 UAX = B 식이 되며, D-클래스는 다음과 같이 재정의 된다[1].

[정의 2]

$$D_A = \{ B \in M_n(F) : \exists C, X, Y, U, V \in M_n(F) \text{ such that } UAX = B, VBY = A \}$$

[정의 2]에 의해 $M_n(F)$에 속한 임의의 행렬 A에 대한 UAX의 연산 후, 그 결과 값인 B로 VBY 연산을 한다. VBY의 연산 결과 중 A 행렬이 있다면, A 와 B는 D-클래스가 된다. 이 과정에서 UAX의 중복을 제거하여 실행 효율을 개선한 공유 메모리 MIMD구조에서의 병렬 알고리즘을 구현되었다[1,3].

위의 병렬 알고리즘은 행렬의 크기가 커질수록 공유 되는 메모리가 커져 문제점이 발생된다. 이 문제를 해결하기 위해 두 프로세서(Intel Pentium4 2.0GHz)로 구성되는 분산 메모리 구조에서의 병렬 알고리즘을 설계 및 구현 되었다. 이 중 하나는 소켓을 이용한 병렬처리 알고리즘으로, 하나의 프로세서는 UAX 연산을 다른 하나는 VBY 연산을 하여, 결과 값은 소켓통신을 하여 전달된다. 다른 하나는 SAN 기반의 데이터 공유 기술을 이용한 병렬처리 알고리즘으로 첫 번째와 같이 하나의 프로세서는 UAX, 다른 하나는 VBY 연산을 처리하게 되고, 결과 값은 SAN 기반의 데이터 공유 기술을 이용하여 전달된다[1].

행렬 크기의 증가로 그리드 환경에서의 병렬 처리가 필요하며, 병렬 알고리즘 설계 및 구현하기 위해 Globus가 설치 된 클러스터 환경에서 MPICH를 사용하여 병렬 알고리즘을 설계 및 구현 되었다. 클러스터는 컴퓨터의 Processor 가 Intel Pentium4 2.66GHz로 된 것이며, 총 41 대로 구성되었다. Globus 가 설치된 컴퓨터에서 실행이 시작되며, 결과 값을 받아 병합 한다. 다른 40대의 컴

퓨터는 UAX와 VBY 연산의 1/40씩 배분되며, 배분된 연산을 처리하고, 처리된 데이터는 Globus가 설치 된 컴퓨터에 전송한다.

UAX 연산 과정은 각 프로세서에서 fork()를 하여 Child 프로세스 생성 후 Child 프로세서에서 UAX 연산을 한다. Parent 프로세스에서는 Child프로세스의 결과와 현재 ID를 기준으로 -1인 프로세서에서 결과를 받아 병합하고, +1인 프로세서로 결과를 전송한다. 최종 결과 값은 ID:0번 인 프로세서에게 전달되며, VBY 연산을 위해 모든 프로세서로 전달된다. VBY 연산과정은 fork()를 하여 Child 프로세서에서 VBY연산, Parent 프로세스에서는 VBY의 연산 결과, 행렬 A와 B가 D-클래스라는 결과를 얻게 되면, LINE 방식을 이용하여 각 프로세서로 현재 작업 중단 메시지와 다음 작업 실행 메시지를 송수신 한다. 최종 결과는 UAX의 과정과 같이 ID:0번인 프로세서로 전달된다[4,5].

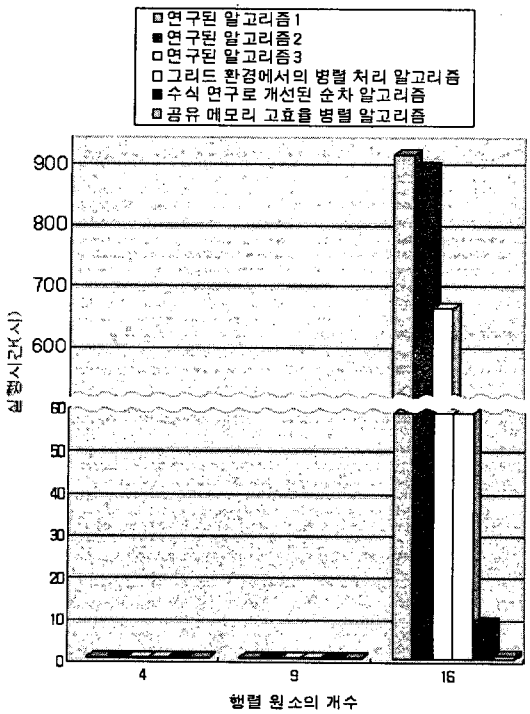
[그림 1]은 연구된 알고리즘의 실행 결과를 보여준다. 2 x 2와 3 x 3행렬의 실행 시간은 2분미만으로 큰 차이가 없다[1]. 하지만, 4 x 4 행렬의 실행 시간은 시간 복잡도로 인해 크게 증가하며 알고리즘에 따라 성능의 차이가 크다. 실행 효율을 비교하면, 그리드 환경에서의 병렬 알고리즘, 공유 메모리 MIMD구조에서의 병렬 알고리즘(연구된 알고리즘3), 분산 메모리 구조에서의 병렬 알고리즘(연구된 알고리즘1,3)의 순서로 실행 효율이 높았다. 이 결과는 프로세서 수의 차이와 메모리 공유 방식 때문이다. 병렬처리에 의해 연산속도가 향상 되었으나, 행렬크기의 증가에 따른 연산 량 증가로 5 x 5 이상의 행렬에서 D-클래스의 결과를 얻기에는 한계가 있다. 따라서 실행 시간을 개선하기 위해, 프로세서 수의 증가와 함께 수식의 연구도 필요하다.

3. 공유 메모리 기반의 개선된 D-클래스 계산 병렬 알고리즘

위에서 언급된 한계를 극복하기 위해 실행 시간을 개선한 공유 메모리 기반의 D-클래스 계산 병렬 알고리즘을 설계 및 구현하였다. DB를 이용하고 수식을 연구하여 개선된 순차 알고리즘을 병렬처리 한다. DB는 다음과 같은 과정으로 생성된다. $M_n(F)$의 임의의 행렬과 $M_n(F)$의 연산 결과는 $M_n(F)$에 포함 된다. 따라서 UAX와 VBY 연산의 결과도 $M_n(F)$에 포함된다. $M_n(F)$의 임의의 행렬과 $M_n(F)$와의 연산 결과를 파일로 저장 하고, 이 연산에 대한 결과가 필요할 때 파일에서 저장된 데이터 사용한다. 이것을 위해 $M_n(F)$와 $M_n(F)$의 연산 결과를 저장 했으며, 저장된 데이터 파일의 크기는 약 460MB 이다. 수식 연구를 통해, 이 파일을 알고리즘[1]의 VBY 연산과정에 적용하여 순차 알고리즘을 구현했다. VB 연산 결과가 V에 포함되는 것을 이용하여, 실행 시간을 개선하며, 먼저 VY 연산을 하여, 그 결과가 A 행렬이 되는 V 원소를 찾아 메모리에 저장한다. VB의 연산 결과는 V에 포함되므로, UAX 연산 결과로 얻어진 B행렬과 VB연산 결과가 앞서 과정의 결과가 저장된 메모리와 비교하여 최종 결과를 얻을 수 있다.

[그림 2]은 결과를 저장하는 알고리즘이며, DB파일에 연산결과, Index 파일에 선택된 행렬과 결과의 개수를 저장한다. [그림 3]은 수식을 연구하여 구현된 순차 알고리즘을 기반 한 병렬 알고리즘이며, 공유 메모리 MIMD 구조의 Intel Xeon 2.66GHz(FSB533)(L1 Cache: Execution Trace cache, L2 Cache: 512KB Advanced Transfer) 프로세서 2개로 처리한다. 임의의 A 행렬에 대한 D-클래스 연산은 각 프로세서에서 인덱스를 참고하여 전체 데이터의 1/2을 읽고, T_1T_2의 연산 결과로 A 행렬이 있는 T_1을 S_{DB}에 저장한다. UAX 연산 과정에서 UA의 연산결과는 중복된 결과 값을 제거하여 S_A에 저장되고, S_A 와 X와 연산을 하게 된다. 이때, S_A와 X의 연산은 DB파일에 저장 되어있고, DB파일의 데이터를 참고하여 UAX의 결과 값을 SB에 저장한다. VBY의 연산과정은 S_B에 저장된 B값이 선택되어지면, VB 연산을 한다. VB의 연산 결과는 T_1에 포함되고, Y는 T_2로 치환될 수 있다. 따라서 VB 연산 결과가 S_{DB}에 포함되어 있는지를 검사하여, A와 B의 D-클래스가 결정 된다.

[그림 1]은 수식을 연구하여 개선된 순차 알고리즘, 공유 메모리 기반의 D-클래스 계산 병렬 알고리즘과 연구된 알고리즘[1,3]의 실행 결과를 보여준다. 공유 메모리 기반의 D-클래스 계산 병렬 알고리즘, 수식을 연구하여 개선된 순차 알고리즘, 연구된 알고리즘[1,3]의 순으로 효율이 높으며, 가장 효율이 높은 공유 메모리 기반의 D-클래스 계산 병렬 알고리즘은 다른 알고리즘과 비교해 0.6/9에서 0.6/910의 비율로 약15-150배 실행 효율이 향상되었으며, 이것은 $M_n(F)$와 $M_n(F)$의 연산 결과를 저장하고, 저장된 결과를 이용, 반복되는 연산과정이 개선되었기 때문이다.



[그림 1] 실행 결과

```
File DB, Index

for T1 in Mn(F)
  write T1 to Index
  for each T2 in Mn(F)
    CheckM = T1T2
    if CheckM not exit from T1 Index in DB
      write CheckM to DB
      IndexNum plus 1
  write IndexNum to Index
  Initialize IndexNum
```

[그림 2] DB 알고리즘

```
Initialize DA , SA , SB , SDB to an empty set

for each A in Mn(F)
fork
  thread 1:
  read CheckM form DB in M_n^0(F)
  if CheckM equal A
    insert CheckM to SDB
  thread 2:
  read CheckM from DB in M_n^1(F)
  if CheckM equal A
    insert CheckM to SDB
join

fork
  thread 1:
  for each U in M_n^0(F)
    T = UA
    if not exist T in SA
      insert T to SA
    for each T in SA
      for each X in M_n^0(F)
        if not exist TX in SB
          insert TX to SB
  thread 2:
    for each U in M_n^1(F)
      T = UA
      if not exist T in SA
        insert T to SA
      for each T in SA
        for each X in M_n^1(F)
          if not exist TX in SB
            insert TX to SB
join

fork
  thread 1:
  for each B in M_n^0(F)
    if B is in SB
    for each V in M_n^0(F)
      T1 = VB
        if T1 in SDB
          insert B to DA
          remove B from M_n^0(F)
  thread 2:
  for each B in M_n^1(F)
    if B is in SB
    for each V in M_n^1(F)
      T1 = VB
        if T1 in SDB
          insert B to DA
          remove B from M_n^1(F)
join
```

[그림 3] DB 이용한 병렬 알고리즘

4. 결론 및 향후 과제

D-클래스는 개인키, 공개키로 이용하여 보안에 응용될 수 있는 가능성을 가지고 있다. 하지만 계산 복잡도가 NP-완전 문제로 현재 극히 제한된 크기의 행렬에 대한 D-클래스만이 알려져 있다. 이런 D-클래스 연산의 효율 높이기 위해 수식 연구와 병렬 알고리즘을 설계 및 구현하였다. 하지만, 연구된 병렬 처리 알고리즘은 5 × 5 이상의 행렬에 대한 결과를 얻기에는 한계가 있다. 이런 문제를 해결하기 위해 공유 메모리 기반의 D-클래스 계산 병렬 알고리즘을 구현 하였으며, 실행 시간이 개선된 결과를 보였다. 하지만, 5 × 5 이상의 행렬의 D-클래스 연산 결과를 얻기 위해 클러스터나 그리드환경에서 병렬 처리가 필요하다. 따라서 클러스터나 그리드 환경에서의 병렬 알고리즘이 요구되며, 최적화를 위해 수식 연구도 필요하다[4,5,6,7].

[참고문헌]

[1] Chul-Gyu Shin, Jae-Il Han, "A Parallel Algorithm the for D-Class Compution ", KIPS Fall Conference, 2004, pp. 965-968

[2] Dock Sang Rim, Jin Bai Kim, "Tables of D-Classes in the semigroup Bn of the binary relations on a set X wit n-elements," Bull. Korea Math. Soc. Vol.20. No. 1, 1983, pp. 9-13

[3] Chul-Gyu Shin, Jae-Il Han, "A Study on the D-Class Computing Algorithm" , KIPS Spring Conference, 2004, pp. 903-906

[4] Barry Wilkinson, Michael Allen, Parallel Programming with MPI, Prentice Hall, 1999

[5] Fran Berman, Geoffrey C. Fox, Anthony J. G. Hey, "Grid Computing-Making the Global Infrastructure a Reality, WILEY, 2001

[6] Gene H. Golub, Charles F. Van Loan. Matrix Computation, The Johns Hopkins' University Press, 1983

[7] Jonn L.Hennessy & David A.Patterson, "Computer Architecture (3rd International Edition) : A Quantitative Approach ", MORGAN KAUFMANN, 2002