通信情報保護學會誌
第4卷 第2號, 1994. 6

SIMD상에서 이산대수 문제에 대한 병렬 알고리즘

Parallel Algorithms for the Discrete Logarithm Problem on SIMD Machines

김 양 희*, 정 창 성**

요 약

고속 계산을 요구하는 분야에서는 여러개의 프로세싱 소자를 사용하여 속도를 증가시키는 병렬 처리의 필요성이 점점 증대되고 있다. 특히 암호처리에서 이산대수 문제나 factorization문제는 많은 시간이 걸리므로 고속계산을 위한 병렬처리가 매우 중요하다. 본 논문에서는 Pohlig-Hellman에 의한 이산대수 알고리즘을 SIMD구조의 병렬 컴퓨터상에서 고속으로 처리할 수 있는 두가지 병렬 이산대수 알고리즘을 제시하며, 이를 16개의 트랜스퓨터로 구성된 병렬 컴퓨터인 KOPS(Korea Parallel System)상에서 구현한 성능평가를 제시한다.

1. 서 론

병렬 컴퓨터는 크게 나누어 각 프로세싱 소자(Processing Element, PE)들의 제어방법(control strategy)에 따라 SIMD(Single Instruction stream multiple Data stream)형과 MIMD(Multiple Instruction stream Multiple data stream)형으로 구분할 수 있으며, 기억장치와 프로세서간의 관계에 따라 기억장치 공유모델(Shared Memory Model, SMM)과 기억장치 분산문델(Distributed Memory Model, DMM)로 나눌 수 있다.

SIMD형은 각각의 프로세서가 하나의 공통된 제어장치(control unit, CU)의 제어를 받아 동기적으로 작동하는 구조의 병렬모델로 한순간에 모든 프로세싱 소자들이 같은 동작을 한다. 기억장치 공유모델은 모든 프로세스가 기억장치(global shared memory)를 공유하는 모델로, 각 프로세서들은 버스(bus)나 스위치(switch)를 이용하여 프로세서가 원하는 기억장치에 접근할 수 있으며 프로세스간의 통신은 공유기억장치의 자료들을 통하여 이루어진다. 기억장치 분산모델은 각각의 PE가 독립적인 기억장치(local memory, LM)를 가지는 모델로 각 프로세서들은 연결망(Interconnection Network)에 의해 연결되며, 프로세스들간의 통신은 연결망을 통해 이루어진다. 기억장치 분산모델에는 선형구조 컴퓨터(Linear Connected Computer, LCC), 메쉬구조 컴퓨터

* 고려대학교 전자공학과
** 고려대학교 전자공학과

(Mesh Connected Computer, MCC), 큐브구조 컴퓨터(Cube Connected Computer, CCC) 등이 있다.

암호처리 분야에 있어서 이산대수 문제나 factorization문제는 매우 큰 숫자를 다루는 문제로 많은 계산 시간이 걸리는 문제들이므로 고속계산을 위한 병렬처리가 매우 적절하다. 이산대수 문제란 주어진 finite field $GF(q)$의 한 primitive element g와 $GF^*(q)$의 한 element y에 대하여 $y = g^x \pmod q$, $0 \le x \le q-2$를 만족하는 x를 구하는 문제이다. 이산대수 문제의 해법은 finite field $GF(q)$의 q값에 의한 분류와 해를 구하는 기법에 의한 분류로 크게 두가지로 나누어 생각할 수 있다. q값에 의해 분류하는 경우에는 q가 prime number인 경우와 2의 지수승(2^n)인 경우의 두가지로 분류할 수 있으며, 해를 구하는 기법에 의해 분류할 경우에는 Pohlig-Hellman이 제시한 알고리즘[8]과 Adleman이 제시한 index calculus 알고리즘[1]으로 분류할 수 있다.

본 논문에서는 Pohlig-Hellman에 의한 이산대수 알고리즘을 SIMD구조의 병렬 컴퓨터상에서 고속으로 처리할 수 있는 두가지 병렬 Pohlig-Hellman 알고리즘을 제시하며, 이를 16개의 트랜스퓨터로 구성된 병렬 컴퓨터인 KOPS(Korea Parallel System)상에서 구현한 성능평가를 제시한다.

2. 이산대수 문제 알고리즘

이산대수 문제란 주어진 finite field $GF(q)$의 한 primitive element g와 $GF^*(q)$의 한 element y에 대하여 $y = g^x \pmod q$, $0 \le x \le q-2$를 만족하는 x를 구하는 문제이다. 이산대수 문제의 해법을 구하는 알고리즘은 primitive element g의 order가 비교적 작은 소인수를 갖을 때 효과적인 Pohlig-Hellman의 알고리즘[8]과 finite field의 원소들이 비교적 작은 집합의 원소들의 곱으로 표현될 수 있다는 사실을 이용한 index calculus 알고리즘[1]을 들 수 있다. Pohlig-Hellman 알고리즘은 primitive element g의 order가 비교적 작은 소인수를 갖을 때 효과적인 알고리즘으로 다음과 같은 stage로 구성된다: (1) table generation (2) computation of coefficients (3) postcomputation. 우선 $q-1$이

$$q-1 = \prod_{i=1}^{k} p_i^{n_i} \quad (p_1 < p_2 < \cdots < p_k, \ p_i \text{ is prime})$$

로 소인수분해 된다고 가정하면, stage(1)에서 각 prime factor $p_i^{n_i}$에 대하여 $h = g^{\frac{q-1}{p_i}}$를 generator로 하는 $\{h^0, h^1, \ldots, h^{p_i-1}\}$의 table을 만든 다음 computation of coefficients stage에서는 초기 단계에서 구한 table을 이용하여 각각의 $p_i^{n_i}$에 대해 $x = x_i \pmod{p_i^{n_i}}$를 구한 후, 마지막 stage에서 Chinese Remainder Theorem을 이용하여 $x \pmod{q-1}$의 값을 구한다.

Algorithm Pohlig-Hellman

Input. Prime number q with prime factorization of $q-1 = p_1^{n_1} p_2^{n_2} \ldots p_k^{n_k}$, fixed primitive element g of $GF(q)$, and an element $y \in GF^*(q)$.

Output. Discrete logarithm of y to the base g.

1. (Table Gereration)

   For $i = 1$ to k do

   compute $h = g^{\frac{q-1}{p_i}} \pmod q$

   make a table $\{h^0, h^1, \ldots, h^{p_i-1}\}$

2. (Computation of Coefficients)

   For $i = 1$ to k do

   find $x \equiv x_i \pmod{p_i^{n_i}}$ where

   $$x = \sum_{j=0}^{n_i-1} b_j p_i^j \pmod{p_i^{n_i}}$$

3. (Postcomputation)

   find x s.t. $x \pmod{\prod_{i=1}^{k} p_i^{n_i}}$

using the Chinese Remainder Theorem

End Algorithm

❖ 정리 1 만일 $q-1$이 $q-1=\prod_{i=1}^{k} p_i^{n_i}(p_1 < p_2 < \ldots < p_k,\ p_i \text{ is prime})$로 소인수분해 된다고 가정하면, $GF(q)$에 있는 primitive element g와 $y \in GF^*(q)$에 대해 Pohlig-Hellman 알고리즘은 $O(\sum_{i=1}^{k}(p_i \log q + n_i(\log q + p_i)))$의 시간복잡도를 갖는다.

증 명 p_ith root of unity를 계산하는데 $O(\log q)$가 걸리므로, table generation stage는 $O(\sum_{i=1}^{k} p_i \log q)$가 걸린다. Coefficients compuataiton stage는 $O(\sum_{i=1}^{k} n_i(\log q + p_i))$, postcomputation stage는 $O(k \log q)$가 걸리므로, 전체 시간복잡도는 $O(\sum_{i=1}^{k}(p_i \log q + n_i(\log q + p_i)))$가 된다.

3. 병렬 Pohlig-Hellman 알고리즘

본 장에서는 SIMD구조의 병렬 컴퓨터상에서 이산대수 문제의 해를 구하는 두가지 병렬 Pohlig-Hellman 알고리즘에 대해 살펴보고자 한다. 병렬 컴퓨터 모델은 1부터 m까지 index된 m개의 프로세싱 소자(Processing Element, PE)들로 구성되어 있다고 가정하며, PE[i]는 index가 i인 PE를 나타낸다. $i_{q-1}\ i_{q-2} \ldots i_1 i_0$가 i의 index이고 각 PE의 index를 binary로 표현할 때 2^p block은 p개의 least significant bit가 다른 연속된 2^p개의 PE들로 구성되어 있다. Bit p가 다른 두개의 2^p block은 합쳐져서 2^{p+1} block을 이루게 된다. 이때 $i_p=0$이면 left 2^p block, $i_p=1$이면 right 2^p block이라 한다. $m=2^k$ 일때 각 PE들은 2^0 block이고, 전체 PE는 2^k block이다. 본 논문에서는 기본 병렬 연산 SUM(각 PE는 모든 PE가 갖고 있는 data의 합을 갖는다)을 이용하며 [7], 이 연산은 LCC상에서는 $O(m)$, MCC상에서는 $O(\sqrt{m})$, CCC상에서는 $O(\log m)$의 시간이 걸린다.

병렬 Pohlig-Hellman I 알고리즘에서 table generation과 coefficients computation stage는 각 프로세싱 소자가 $q-1$의 prime factor 각각을 처리할 수 있도록 병렬화 하고 postcomputation stage에서는 기본 병렬 연산 SUM을 이용하여 Chinese Remainder Theorem을 계산한다.

Parallel Algorithm Pohlig-Hellman_I

Input. Prime number q with prime factorization of $q-1=p_1^{n_1}\ p_2^{n_2}\ldots p_k^{n_k}$, fixed primitive element g of $GF(q)$, and an element $y \in GF^*(q)$. Each PE[i] contains p_i.

Output. Discrete logarithm of y to the base g.

1. (Table generation)

for $i=1$ to k do parallel
compute generator $h=g^{\frac{q-1}{p_i}} \pmod q$
make a table $\{h^0, h^1, \ldots, h^{p_i-1}\}$

2. (Computation of coefficients)

/* find $x \equiv x_i \pmod{p_i^{n_i}}$ for $i=1, \ldots, k$, where $x_i=\sum_{j=0}^{n_i-1} b_j p_i^j \pmod{p_i^{n_i}}$ */

for $i=1$ to k do parallel
/* find coefficient b_0 */
compute $y^{\frac{q-1}{p_i}}$
find b_0 satisfying $y^{\frac{q-1}{p_i}}=h^{b_0}$ using table look up
/* find coefficient $b_1, b_2, \ldots, b_{n_i-1}$ */
for $j=1$ to n_i-1 do

$y_j = y/g^{b_0 + b_1 p_i + \cdots + b_{j-1} p^{j-1}}$

compute $y_j^{\frac{q-1}{p_i^{j+1}}}$

find b_j satisfying $y_j^{\frac{q-1}{p_i}} = h^{b_j}$

using table look up

3. (Postcomputation)

/* Chinese Remainder Theorem:

find x s.t. $x(\mathrm{mod} \prod_1^k p_i^{n_i})$ */

for $i = 1$ to k do parallel

$M = p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$

$M_i = M/p_i^{n_i}$

find N_i satisfying $M_i N_i \equiv 1 (\mathrm{mod}\ p_i^{n_i})$

find $x = \sum_{i=1}^{k} x_i M_i N_i$ by the SUM operation

End Parallel Algorithm I

❖ 정리 2 만일 $q-1$이 $q-1 = \prod_{i=1}^{k} p_i^{n_i} (p_1 < p_2 < \cdots < p_k,\ p_i$ is prime)로 소인수분해 되고 k개의 프로세싱 소자를 사용한다고 가정하면, $GF(q)$에 있는 primitive element g와 $y \in GF^*(q)$에 대해 병렬 Pohlig-Hellman_I 알고리즘은 LCC상에서는 $O(\max(p_i \log q + n_i (\log q + p_i))) + O(k)$, MCC에서는 $O(\max(p_i \log q + n_i(\log q + p_i))) + O(\sqrt{k})$, CCC에서는 $O(\max(p_i \log q + n_i(\log q + p_i))) + O(\log k)$의 시간 복잡도를 갖는다.

앞에서 제시한 병렬 Pohlig-Hellman_I 알고리즘에서 각 프로세서는 prime factor p_i에 대해 precomputation와 coefficient를 계산해야 하므로 prime factor p_i가 커질수록 계산량이 많아져 프로세싱 소자간의 load balancing 이 잘되지 않는다. 그러므로 load balancing을 맞추기 위해 $q-1$의 prime factor들을 합이 거의 같은 크기가 되도록 grouping한 후 각 프로세싱 소자들에 각 group을 할당한다. 즉 $q-1$이 $q-1 = \prod_{i=1}^{k} p_i^{n_i}$일때 다음과 같이 $q-1$의 prime factor를 grouping한다:

$G_1 = \{p_{11}^{n_{11}}, p_{12}^{n_{12}}, \ldots, p_{1l_1}^{n_{1l_1}}\}, \cdots,$

$G_m = \{p_{m1}^{n_{m1}}, p_{m2}^{n_{m2}}, \ldots, p_{ml_m}^{n_{ml_m}}\}$

with $\sum_{i=1}^{l_1} p_{1i}^{n_{1i}} \approx \sum_{i=1}^{l_2} p_{2i}^{n_{2i}} \approx \cdots \approx \sum_{i=1}^{l_m} p_{mi}^{n_{mi}}$ and

$\sum_{j=1}^{m} \sum_{i=1}^{l_j} p_{ji}^{n_{ji}} = \sum_{i=1}^{k} p_i^{n_i}$.

그후 PE[i]에 G_i를 할당한 후 각 group에 대해 병렬로 처리해 줌으로써 각 프로세싱 소자간의 load balancing을 잘 유지해 줄 수 있다.

Parallel Algorithm Pohlig-Hellman_II

Input. Prime number q with prime factorization of $q-1 = p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$ and grouping prime factors of $q-1$ by $G_1, \cdots, G_m$ in such way that the sum of prime factors in each group are almost equal, fixed primitive element g of $GF(q)$, and an element $y \in GF^*(q)$. Each PE[i] contains primes in G_i.

Output. Discrete logarithm of y to the base g.

1. (Table generation)

for $i = 1$ to m do parallel

for each prime $p_{is} (s = 1, \ldots, l_i)$ in G_i do

compute a generator $h = g^{\frac{q-1}{p_{is}}} (\mathrm{mod}\ q)$

make a table $\{h^0, h^1, \ldots, h^{p_{is}-1}\}$

2. (Computation of Coefficients)

/* for each G_i, find $x \equiv x_{is} (\mathrm{mod}\ p_{is}^{n_{is}})$

for $s = 1, \ldots, l_i$, where $x_{is} = \sum_{j=0}^{n_{is}-1} b_j p_i^j$

$(\mathrm{mod}\ p_{is}^{n_{is}})$ */

for $i = 1$ to m do parallel

for each prime $p_{is} (s = 1, \ldots, l_i)$ in G_i do

/* find coefficient b_0 */

compute $y^{\frac{q-1}{p_{is}}}$

find b_0 satisfying $y^{\frac{q-1}{p_{is}}} = h^{b_0}$

using the table look up

/* find coefficient $b_1, b_2, \ldots, b_{n_{is}-1}$ */

for $j = 1$ to $n_{is}-1$ do

$y_j = y/g^{b_0 + b_1 p_{is} + b_2 p_{is}^2}$

compute $y_j^{\frac{q-1}{p_{is}^{j+1}}}$

find b_j satisfying $y_j^{\frac{q-1}{p_{is}^{j+1}}} = h^{b_j}$

using the table look up

3. (Postcomputation)

/* Chinese Remainder Theorem: find x such that $x \pmod{\prod_{j=1}^{m} \prod_{i=1}^{l_j} p_{ji}^{n_{ji}}}$ */

for $i = 1$ to m do parallel

$M = p_1^{n_1} p_2^{n_2} \ldots p_k^{n_k}$

$x_i = 0$

for each prime $p_{is} (s = 1, \ldots, l_i)$ in G_i do

$M_{is} = M/p_{is}^{n_{is}}$

find N_{is} satisfying $M_{is}N_{is} \equiv 1 \pmod{p_{is}^{n_{is}}}$

compute $x_i = x_i + x_{is}M_{is}N_{is}$

find $x = \sum_{i=1}^{m} x_i$ by the SUM operation

End Parallel Algorithm_Ⅱ

❖ 정리 3 만일 $q-1$이 $q-1 = \prod_{j=1}^{k} p_i^{n_i} (p_1 < p_2 < \ldots < p_k, p_i$ is prime)로 소인수분해되고 m개의 프로세싱 소자를 사용하며 $G_1 = \{p_{11}^{n_{11}}, p_{12}^{n_{12}}, \ldots, p_{1l_1}^{n_{1l_1}}\}, \cdots, G_m = \{p_{m1}^{n_{m1}}, p_{m2}^{n_{m2}}, \ldots, p_{ml_m}^{n_{ml_m}}\}$ $(\sum_{i=1}^{l1} p_{1i}^{n_{1i}} \approx \sum_{i=1}^{l2} p_{2i}^{n_{2i}} \approx \cdots \approx \sum_{i=1}^{lm} p_{mi}^{n_{mi}}, \sum_{j=1}^{m} \sum_{i=1}^{l_j} p_{ji}^{n_{ji}} = \sum_{i=1}^{k} p_i^{n_i})$이 $q-1$의 prime factor의 grouping이라면, $GF(q)$에 있는 primitive element g와 $y \in GF^*(q)$에 대해 병렬 Pohlig-Hellman_Ⅱ 알고리즘은 LCC상에서는 $O(\max(\sum_{i=1}^{l_j}(p_{ij}\log q + n_{ij}(\log q + p_{ij}))) + O(m)$, MCC상에서는 $O(\max(\sum_{j=1}^{l_i}(p_{ij}\log q + n_{ij}(\log q + p_{ij}))) + O(\sqrt{m})$, CCC 상에서는 $O(\max(\sum_{j=1}^{l_i}(p_{ij}\log q + n_{ij}(\log q + p_{ij}))) + O(\log m)$의 시간복잡도를 갖는다.

4. 성 능 평 가

본장에서는 Pohlig-Hellman에 의한 이산대수 알고리즘을 SIMD구조의 병렬 컴퓨터상에서 처리할 수 있는 두개의 병렬 Pohlig-Hellman 알고리즘을 16개의 트랜스퓨터로 구성된 병렬 컴퓨터인 KOPS(Korea Parallel System)상에서 구현한 성능평가를 제시한다.

〈표1〉과 〈표2〉에서 q = 6381556406866377515470037, $q-1 = 2^2 \cdot 541 \cdot 1223 \cdot 1987 \cdot$

단위(sec)

y	x	Num. of PE	Total time
5	807070627058312953009721	1PE	152.26
		8PE	minimum:6.04
			maximum:90.55
12	219447653245318057546598 3	1PE	331.77
		8PE	minimum:8.10
			maximum:159.40
20	300154715951149352847570 3	1PE	282.46
		8PE	minimum:8.03
			maximum:87.07

표 1 순차알고리즘과 병렬 Pohlig-Hellman_Ⅰ 알고리즘의 수행시간비교(grouping을 안했을 때)

단위(sec)

y	x	Num. of PE	Total time
5	807070627058312953009721	1PE	152.26
		2PE	minimum:82.26
			maximum:100.41
12	219447653245318057546598 3	1PE	331.77
		2PE	minimum:174.68
			maximum:191.37
20	300154715951149352847570 3	1PE	282.46
		2PE	minimum:112.64
			maximum:197.51

표 2 순차알고리즘과 병렬 Pohlig-Hellman_Ⅱ 알고리즘의 수행시간비교(grouping을 했을 때)

$3571 \cdot 6133 \cdot 6997 \cdot 7919$, 그리고 generator $g = 3$일때 순차알고리즘과 병렬알고리즘의 수행시간을 비교한 것으로 〈표2〉는 {2, 541, 1223, 1987, 3571, 6133}와 {6997, 7919}로 grouping하였을 때의 수행결과를 나타낸다.

위의 〈표1〉과 〈표2〉에서 보는 바와 같이 prime factor들을 grouping하였을때 load balancing이 잘되어 성능이 향상되는 것을 알 수 있다.

5. 결 론

본 논문에서는 Pohlig-Hellman에 의한 이산대수 알고리즘을 SIMD구조의 병렬 컴퓨터상에서 고속으로 처리할 수 있는 두개의 병렬 Pohlig-Hellman 알고리즘을 제시하였으며, 이를 16개의 트랜스퓨터로 구성된 병렬 컴퓨터인 KOPS (Korea Parallel System)상에서 구현한 성능평가를 제시하였다. 처음 제시한 병렬 Pohlig-Hellman_I 알고리즘에서 각 프로세서는 prime factor p_i에 대해 precomputation과 coefficient를 계산해야 하므로 prime factor p_i가 커질수록 계산량이 많아진다. 그러므로 load balancing을 맞추기 위해 $q-1$의 prime factor들을 합이 거의 같은 크기가 되도록 grouping하였다. 그후 각 프로세싱 소자들에 각 group을 할당하여 병렬로 처리함으로써 프로세서의 수가 $O(m)$일때 이산대수 문제를 $O(m)$배 빨리 처리하였다.

본 논문은 EESRC 및 ETRI에서 지원받았음.

참 고 문 헌

[1] L.Adleman, "A subexponential algorithm for the discrete logarithm problem with application to cryptography," Proc.IEEE 20th Annual Symposium on Foundation of Computer Science, 1979, pp.56-60.

[2] I.F.Blake, R.Fuji-Hara, R.C.Mullin and S.A.Vanstone, "Computing logarithms in finite fields of characteristc two," SIAM J.Algebra and Discrete Methods.

[3] Don Coopersmith, "Fast evaluation of logorithms in fields of characteristic two,"IEEE Trans. on Inform. Theory, Vol. IT-30, 1984, pp.472-492.

[4] W.Diffie and M.E.Hellman, "New directions in cryptography,"IEEE Trans. on Information Theory, Vol. IT-22, 1976, pp.644-654.

[5] D.E.Knuth, The Art of Computer Programming, Vol.2, New York: Addison-Wesley, 1971.

[6] N.Koblitz, A course in number theory and cryptography, Springer-Verlag, 1987.

[7] D.Nassimi and S.Sahni, "Data broadcasting in SIMD computers," IEEE Trans. Comput., 1981, pp.101-106.

[8] S.C.Pohlig and M.E.Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance," IEEE Trans. on Inform. Theory, Vol.24, 1978, pp.106-110.

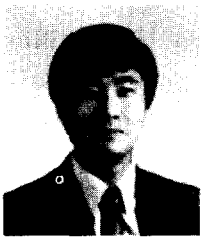
[9] R.Rivest, A.Shamir and L.Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communication of the ACM, Vol.21, 1987, pp.120-126.

□ 著者紹介

김 양 희(金 良 姬)

1982년 이화여자대학교 수학과 졸업(학사)
1984년 서울대학교 대학원 수학과 졸업(석사)
1988년 Univ. of Wisconsin-Madison 대학원 전산과 졸업(석사)
1989년 Univ. of Wisconsin-Madison 대학원 전산과 박사과정 이수
1993년 3월 ~ 현재 고려대학교 전자공학과 박사과정
1990년 3월 ~ 현재 수원전문대학 전산과 전임강사

정 창 성(丁 昌 聲)



1981년 서울대 전기공학과 졸업
1985년 Northwestern Univ. Computer Science, M.S
1987년 Northwestern Univ. Computer Science, Ph.D
1987년 ~ 1992년 포항공과대학 전자계산학과 조교수
1992년 ~ 현재 고려대학교 전자공학과 조교수
관심분야 : 병렬처리, 고속 암호 처리