62 1992. 12. 정보과학회지 제 10 권 제 6 호

□特 輯□

Architecture Specialized 병렬 알고리즘 설계

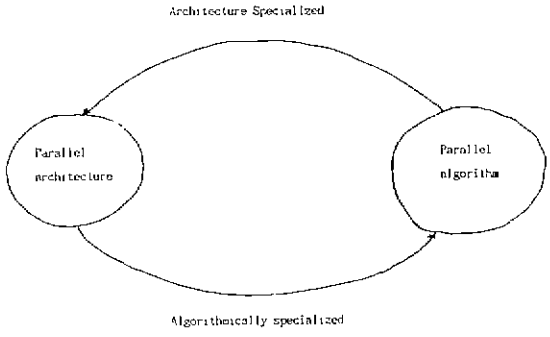
고려대학교 전자공학과 정 창 성*

목 차

I. 서 론

VLSI기술의 발달로 여러 개의 프로세서를 수용할 수 있는 고성능 chip의 개발이 가능하게 되어 이를 대량 사용한 상업용 고도병렬컴퓨터가 등장하고[1], 순차컴퓨터상에서의 성능향상은 물리적인 빛의 속도한계에 도달함에 따라 고속계산을 필요로 하는 분야에서의 병렬처리는 매우 중요하고 필수적인 것으로 인식되고 있고, 특히 병렬컴퓨터상에서의 실제적인 제반문제를 해결하는 병렬알고리즘은 병렬처리의 핵심분야로 이에 관한 많은 연구가 진행되고 있다. 많은 분야에서 대량의 데이타에 대한 고속처리 요구가 증대되고 있기 때문에 병렬알고리즘은 물리, 화학 등의 기초분야와 영상처리, 그래픽스, 수치해석, 인공지능, 계산기하 등의 컴퓨터 과학 전반에 걸친 응용분야에서 개발되고 있고, 많은 산업분야에서 응용되고 있다[2-9].

순차처리와는 달리 병렬처리에서는 하나의 프로세서 대신 여러 개의 프로세서를 사용하여 주어진 문제를 작은 subtask들로 분할하여 동시처리함으로써 시간을 단축시키기 때문에 병렬구조와 병렬알고리즘은 밀접한 연관관계를 가지고 있다. 순차컴퓨터상에서의 알고리즘설계는 low level의 컴퓨터구조에 대한 깊은 지식이 없어도 순차알고리즘을 개발할 수 있지만, 병렬컴퓨터상에서의 병렬알고리즘설계에서는 프로세서들간의 데이타교환을 통해 최종해를 구해야 하기 때문에 프로세서간의 연결방식 및 제어구조에 대한 이해를 필요로 한다. 즉, 병렬구조특성에 따라 병렬알고리즘이 다라지게 되므로, 병렬구조가 주어지면 그 병렬구조에 specialized된 병렬알고리즘을 설계해야 된다(그림 1 참조). 반대로 어떤 특정한 병렬컴퓨터구조가 설계되었다 하더라도 그 위에 효율적인 병렬알고리즘이 구현되지 않으면 그 병렬컴퓨터구조는 아무 효용성이 없으므로 algorithmicalley specialized된 병렬구조의 설계가 필요하다. 이와같이 병렬구조와 병렬알고리즘은 밀접한 상호연관 관계를 갖고 있고, 특히 여러 개의 프로세서상에서 서로 데이타를 교환하면서 문제를 풀어야 하기 때문에 데이타교환을 효율적으로 수행할 수 있는 프로세서상의 데이타 분배



(그림 1) 병렬구조 및 병렬알고리즘

* 종신회원

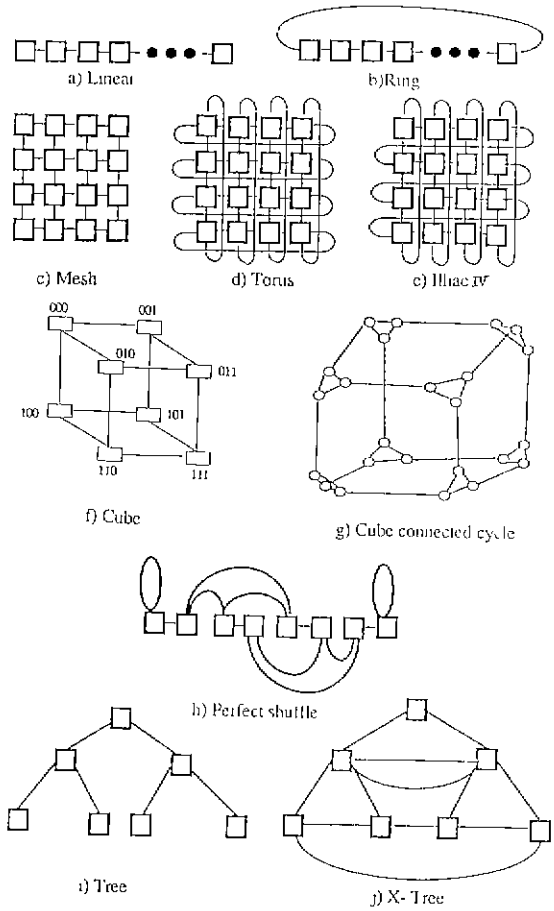
방법과 새로운 병렬성분석이 매우 중요하다.

본 논문에서는 먼저 병렬 계산 모델에 대해 간단히 설명하고, 병렬알고리즘상에서 효율적인 데이타 교환을 위한 데이타 분배 및 병렬성분석을 예를 들면서 설명을 하려고 한다.

II. 병렬계산모델

병렬알고리즘 설계시 주로 사용되는 대표적인 병렬구조모델은 제어방식(control strategy)에 따라 하나의 control unit에 의해 동작되는 SIMD(Single Instruction Stream-Multiple Data Stream)와 각 프로세서가 자신의 control unit에 의해 동작되는 MIMD(Multiple Instruction Stream-Multiple Data Stream) 방식으로 나눌 수 있고, 다시 프로세서간의 연결방식에 따라서 여러 개의 프로세싱 소자가 메모리를 공유하고 있는 SMM(Shared Memory Model)과 각 프로세싱 소자가 local memory를 가지고 일정한 패턴에 따라 연결된 INM(Interconnection Network Model)로 나누어진다. SMM은 임의의 두 개 PE 사이의 데이타 교환이 0(1)에 수행될 수 있다고 가정한 이론적으로 가장 우수한 병렬계산모델로 read나 write의 access conflict에 따라 CRCW(Concurrent Read/Concurrent Write), CREW(Concurrent Rea/Exclusive Write), ERCW(Exclusive Read/Concurrent Write), EREW(Exclusiuc Read/Exclusiue Read/Exclusive Write)로 나누어진다. INM은 다시 연결망에 따라 세분되는데 대표적인 것으로 cube-class, mesh-class, tree class가 있다[8]. Cube connected, cube-connected cycle, PM-21, perfect shuffle 등이 cube class에, linear, ring, mesh, torus, Illiac-IV type이 mesh class에, tree, x-tree 등이 tree class에 속한다(그림 2 참조). 특히 그림 3과 같이 linear connected, mesh connected, ring connected, PM21 connected와 같이 프로세서간에 linear connection을 가진 병렬 computer를 linear class로 분류한다.

프로세싱 소자의 갯수가 n개일 때 각 프로세싱 소자는 0에서 n-1까지 index되어 있고, PE[i]는 i의 index를 가진 프로세싱 소자를 가리킨다. 프로세싱 소자의 index를 이진법으로 나타내었을 때 cube-connected에서는 각 PE가 하나의 bit만 다른 index를 가진 모든 PE에 연결이 되어있다. Cube connected가 한 PE당 연결되는 link 숫자가 logn인데 비해 cube-connected cycle, perfect shuffle 등은 한 PE당 3개로 약간의 overhead는 있지만, cube-connected를 같은 order내에서 simulation할 수

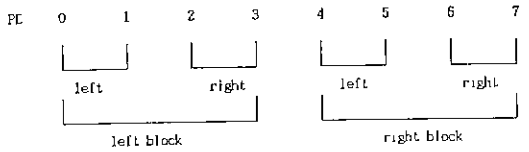


(그림 2) INM 모델

있는 장점이 있다. Mesh는 각 PE가 이웃한 4개의 PE와 연결되어 있고, boundary PE의 연결방식에 따라 torus, Illiac-IV type 등이 있고, 1차원 mesh의 형태가 linear, ring type이 된다. 각 프로세싱 소자는 하나의 arithmetic/logic 명령을 수행하거나 인접한 PE들과 데이타 교환은 0(1)에 수행될 수 있다고 가정한다.

일반적으로 대량의 프로세싱 소자를 사용한 고도병렬 컴퓨터에서는 fine grained parallelism을 사용하기 때문에 프로세싱 소자간의 동기화가 하나의 제어장치에 의해 제어되는 SIMD 방식을 많이 사용하고, 프로세싱 소자(PE)들 간의 communication overhead를 줄이고 병렬성을 최대한도로 사용하기 위해 프로세싱 소자들은 여러 크기의 block단위로 분할하여 subtask들을 동시에 수행하게 된다. 표 1과 같이 프로세싱 소자들은 인접한 PE index를 갖는 left block과 right block의 두 block으로 나눌 수 있고, 다시 block들은 recursive하게 left block과

〈표 1〉



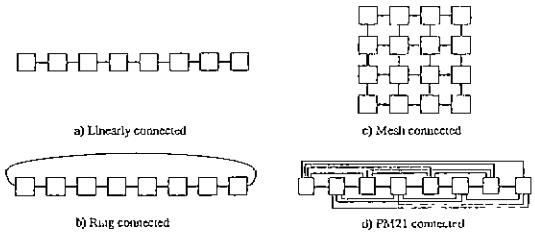
right block으로 나눌 수 있다.

본 논문에서는 대량의 프로세싱 소자를 사용하여 parallelism을 최대한도로 구현할 수 있는 고도병렬컴퓨터상에서의 병렬 알고리즘 설계를 다루고, 이론적인 SMM보다는 실용적인 INM상에서 효율적인 data distribution과 parallelism 분석을 통한 병렬알고리즘 설계를 예를 들면서 3장 및 4장에서 설명하도록 한다.

III. Data Distribution

병렬알고리즘에서는 프로세서에 들어있는 데이타를 서로 교환하면서 문제를 풀어야 하기 때문에 초기에 입력데이타를 프로세서상에 어떻게 분배하느냐가 매우 중요하다. 데이타분배는 주어진 문제의 특성에 따라, 그리고 문제의 해를 구해야 하는 병렬 구조에 따라 달라지므로 문제특성 및 구조특성에 대한 이해와 분석을 필요로 하고, 문제를 푸는 과정에서 프로세서간의 communication overhead를 최대한도로 줄이고 기존의 효율적인 routing방법을 사용할 수 있도록 프로세서상의 데이타분배가 이루어져야 한다. 이론적인 설명보다는 이해를 쉽게하기 위해 실제적인 예를 가지고 설명하기로 한다. 프로세서간에 linear connection을 가진 linear class 병렬컴퓨터(LC) 상에서 TSP(Traveling Salesman Problem)을 simulated annealing 기법을 이용해서 해를 구하는 병렬알고리즘을 생각해 보기로 한다.

TSP는 n개의 도시가 주어졌을 때 임의의 한도시에서 출발하여 모든 도시를 한번만 방문하고 다시 출발했던 도시로 돌아오는 경로중 경로길이가 최소가 되는 경로를 구하는 문제이다. TSP는 많은 계산이 걸리는 NP class 문제로, 이에 대한 많은 해법이 제안되어 왔으나, 여기서는 LC상에서 simulated annealing 기법을 이용해서 고속해를 구하는 방법에 대해 설명한다. Simulated annealing은 주어진 문제의 한해인 초기 configuration에서 시작하여 새로운 configuration을 생성하여 cost function E가 감소할 경우 뿐만아니라 증가할 경우도 $e^{-\frac{\Delta E}{T}}$의 확률에 따라 새로운 상태로 전이하는 move operation을 control parameter T를 감소시키면서 반복함으로써 local



(그림 3) Linear class

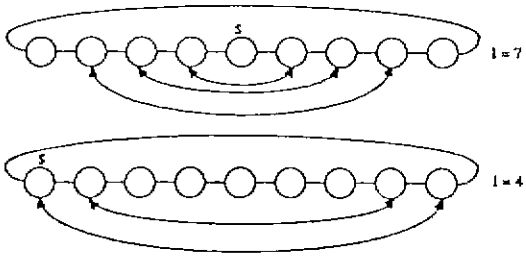
〈표 2〉

```
Algorithm S A
(1)  Start with some initial configuration C
(2)  T = To
(3)  Repeat
(4)  While (not Equilibrium)
(5)      Generate a new configuration Cn
(6)      △ E = E(Cn) - E(C)
(7)      If △ E < 0
(8)      then Replace C with Cn
(9)      else Replace C with Cn with probability e^(-△E/T)
(10) end while
(11) T = αT
(12) Until frozen
```

minima에서 빗어나 global minima에 도달할 수 있는 optimization기법이다. 각 move operation은 다음의 세 과정으로 구성되어 있다(표 2 box 참조). (1) 현재 configuration에서 새 configuration의 generation (2) 두 configuration 사이의 에너지차 계산 (3) 새로운 configuration을 accept할 것인지에 대한 결정. 각 온도 T에서의 move operation은 더이상 새로운 move가 accept되지 않을 때까지 계속되고 T는 점점 감소되어 더 이상 에너지 E의 변화가 없는 frozen state에 들어가면 알고리즘은 종결된다.

Simulated annealing에서는 현재의 move operation이 결정되어야 다음 move operation이 수행될 수 있는 inherent한 순차 성질 때문에 병렬화하는데 많은 어려움이 있다. 여기서는 각 move operation을 병렬처리화함으로써 전체 simulated annealing algorithm을 고속화하는 문제를 다룬다[9].

TSP을 위한 simualated annealing procedure는 다음과 같다. 임의의 도시순서로 이루어진 초기 configura-

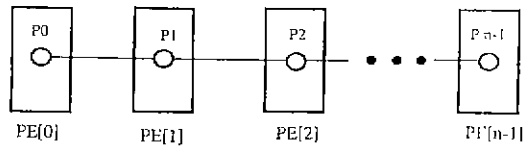


(그림 4) Generation scheme: subtour reverse

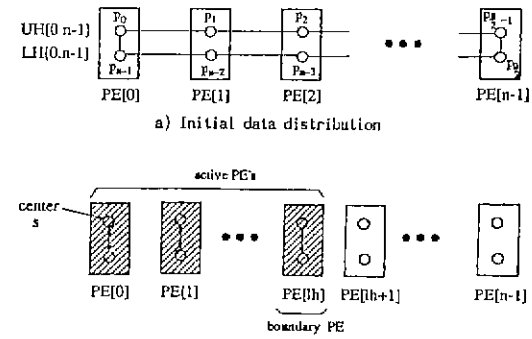
tion을 결정하고 각 configuration의 energy는 경로의 길이로 표현된다. 주어진 온도에서의 move는 다음의 세 단계로 이루어진다. (1) 도시의 방문 순서를 permultation하여 새로운 configuration을 생성한다. (2) 경로길이 차인 energy차 (ΔE)를 계산하다. (3) ΔE가 0보다 작거나 또는 확률 $e^{-\frac{\Delta E}{T}}$로 새로운 configuration을 accept 한다.

n개의 도시 $C=\{c_0, c_1, \cdots, c_{n-1}\}$가 주어지면, C의 경로 P는 방문하는 순서대로 배열된 $\{p_0, p_1, \cdots, p_{n-1}\}$로 표현되고 P는 C를 permutation하여 구해진다. Move operation의 첫번째 단계인 generation 방법은 simulated annealing의 속도증가 및 좋은 해를 구하는데 중요한 역할을 한다. TSP에서의 도시 방문순서를 permutation 하는 generation 방법은 P를 circular ring으로 생각하여 임의의 길이 l을 선택한 후 도시 s를 중심으로 길이 l인 P의 부경로(subtour)를 그림 4와 같이 reverse시킨다. 도시 s는 초기 p_0에서 시작하여 매 move operation마다 $p_1, p_2, \cdots$로 하나씩 증가시키게 된다. 이를 순차알고리즘으로 구현할 경우 각 move operation은 첫번째 step에서 부경로를 reverse 시키는데 0(n)의 시간이 걸리고 두번째와 세번째 step은 0(1) 시간이 걸리므로, 한 move operation을 수행하는데 걸리는 시간은 o(n)이다. 이를 하나의 control unit에 의해 간단한 1 bit flag 데이타를 각 PE에 0(1)에 distribution 할 수 있는 LC SIMD 상에서 구현하는 경우 효율적인 데이타 분배를 통해 move operation을 0(1)에 수행하는 병렬알고리즘을 설계할려고 한다.

만약 그림 5와 같이 경로 P의 각도시 p_i를 PE[i]에 저장한다면, 임의의 도시 s를 중심으로 길이 l인 부경로의 reverse는 각 프로세서가 linear하게 연결되어 있으므로 0(n)의 시간이 걸린다. 따라서 move operation을 0(1)에 수행하기 위해서는 liner connection의 특성을 잘 이용할 수 있는 데이타분배가 필요하게 된다. 그림 6-a와 같이 경로 P를 반으로 접으로 p_i와 p_{n-1}을 한 PE에 저장하는



(그림 5) Data distribution(I)



(그림 6) Data distribution(II)

데이타 분배방법을 생각해 본다. Generation 방법에서 permutation 시키는 부경로의 중심 s가 항상 p_0에 있다고 생각하면 l이 짝수인 경우에는 각 PE에 있는 두 도시를 교환하고(그림 6-b) l이 홀수인 경우에는 인접한 두 개의 PE에 있는 두 도시를 교환하면(그림 6-c) 부경로를 0(1)에 reverse시킬 수 있다. 부경로의 중심 s는 매 move operation마다 전체성 를 left rotate시켜 항상 첫번째 PE[0]의 p_i에 오게 할 수 있다. 그리고 부경로를 reverse시켰을 때의 경로 길이의 차, 즉 에너지차는 부경로에 속하지 않는 도시들을 저장한 PE와 인접한 boundary PE에서 0(1)에 쉽게 구할 수 있고, 이를 사용하여 accept할 지의 decision을 내린 후 SIMD의 control unit에 의해 모든 PE에 0(1)에 1 bit decision 값을 broadcasting하게 된다. 따라서 각 move operation을 0(1)의 시간에 수행할 수 있고, 전체 simulated annealing을 수행하는데 걸리는 시간은 단지 operation의 반복 회수에 비례하게 된다. MIMD인 경우에는 move operation은 한 프로세서에서 다른 모든 프로세서에 1 bit을 distribu-

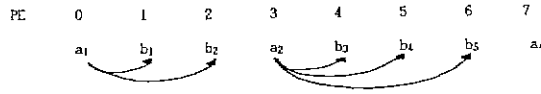
tion하는데 걸리는 시간에 비례하므로 PM-21에서는 0(log n), mesh에서는 $0(\sqrt{n})$의 시간이 소요된다. 위의 예에서 보는 바와 같이 데이타의 분배방법에 따라 프로세서들간의 데이타교환을 최소화 시켜서 최적 병렬 알고리즘을 설계할 수 있다.

IV. Parallelism Analysis

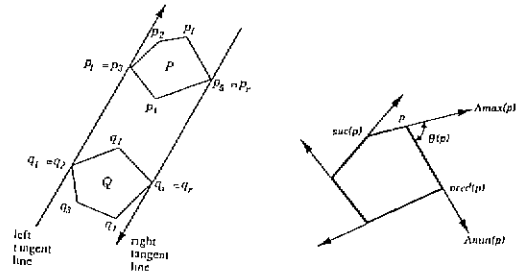
여러 개의 프로세서를 사용하는 병렬컴퓨터상에서 속도를 증가시키기 위해서는 가능한 한 최대의 parallelism을 구현하는 것이 필요하다. 따라서 효율적인 병렬 알고리즘의 개발을 위해서는 병렬구조에 적합한, 주어진 문제가 내재하고 있는 병렬성을 찾아내는 것이 매우 중요하다. 그러나 순차알고리즘이 일반적으로 병렬컴퓨터상에 directly 구현될 수 없고 병렬알고리즘은 순차알고리즘과는 전혀 다른 형태를 가지게 된다. 즉 순차알고리즘을 병렬컴퓨터상에 그대로 구현하려는 경우 시간복잡도는 순차알고리즘과 같거나 더 많은 시간이 걸리게 된다. 주어진 문제의 parallelism은 순차알고리즘상에서 사용한 주요성질에서 쉽게 찾을 수도 있지만, 대부분 주어진 병렬구조상에서의 데이타 분산 및 연결 pattern에 따른 데이타 routing 방식을 고려하여 병렬컴퓨터상에 효율적으로 구현될 수 있는 새로운 algorithmic parallelism을 발견하는 것이 필요하다. 본 절에서는 주어진 문제에 대한 새로운 algorithmic parallelism분석을 발견하는 예로 많은 응용분야에 쓰이는 convex hull 문제를 예로 들면서 설명을 한다[10].

2차원 convex hull 문제는 평면상에 n개의 점의 집합 S가 주어졌을 때 이들 점들을 포함하는 최소 convex polygon의 꼭지점들을 구하는 문제로서 이를 n개의 프로세싱 소자(PE)를 가진 mesh connected와 cube connected 병렬 컴퓨터상에서 구하는 병렬알고리즘을 설계해 보기로 한다. 이때 효율적인 data routing 방법으로 아려진 selected data broadcasting 기법을 사용한다. Selected data broadcasting은 두 개의 sorted list인 $A=(a_1, a_2, \cdots, a_p)$와 $B=(b_1, b_2, \cdots, b_q)$를 merge한 list C가 주어졌을 때 A의 각 element ai를 $a_i \le b_j < a_{i+1}$인 모든 bj를 저장하는 PE에 보내는 operation으로 MCC에서 $0(\sqrt{n})$, CCC에서 0(log n)이 걸린다[10]. C의 각 점은 각 프로세싱 소자 하나에 들어 있다고 가정한다.

Convex hull을 구하는 방법은 많지만 순차 알고리즘과 병렬 알고리즘의 비교를 쉽게 하기 위해 기본적으로 divide-and-conquer 방식을 사용한다. 순차알고리즘에서는 divide-and-conquer 방식을 사용하여, 우선 점들을 y value에 따라 sorting한 후, 점들을 수평선에 의해 분할된 두개의 집합 P와 Q로 분할한다(그림 8 참조). 이때 P의 점들의 y value는 Q보다 크다. 각 P와 Q에 대해 convex hull CH(P)와 CH(Q)를 구한 후, 이들을 merge하여 최종 convex hull CH(S)를 구하게 된다. Merge process는 CH(P)와 CH(Q)간의 tangent line을 구한 후 내부 점들을 제거하면 구할 수 있다. 순차 알고리즘을 수행하는데 걸리는 시간은 $T(n)=2T(n/2)+M(n)$으로, M(n)은 merge하는 데 걸리는 시간이다. Merge process에서 tangent line은 임의의 두 점을 잇는 직선에서 시작하여 CH(P)와 CH(Q)의 꼭지점을 따라 traverse하면서 구하게 되므로 0(n)의 시간이 걸리고, 최종시간은 T(n)=0(nlogn)이 된다. 병렬컴퓨터상에서는 각 PE에 1개의 점이 들어있다고 가정하고, 우선 S를 예비단계로 sort하면 left block과 right block에 각각 P와 Q가 저장되고 left block과 right bolck에서 각각 CH(P)와 CH(Q)를 동시에 recursive하게 구한 후 이들을 merge한다. 그러나, 순차알고리즘에서의 merge방법을 그대로 사용하면 꼭지점을 traverse하기 위한 점들 간의 communication overhead 때문에 순차알고리즘에서의 merge시간보다 더 많은 시간이 필요하게 된다. 따라서, 순차알고리즘과는 달리 데이타들간의 routing을 효율적으로 수행할 수 있는 새로운 algorithmic parallelism을 발견하는 것이 필요하다. 우선 tangent line을 찾기위해 순차알고리즘과 같이 점들을 traverse하는 대신 tangent line이 지날 수 있는 모든 가능한 점들의 쌍(pair)집합 M을 병렬로 구할 수 있다면 그 중에서 tangent line을 쉽게 찾아낼 수 있으

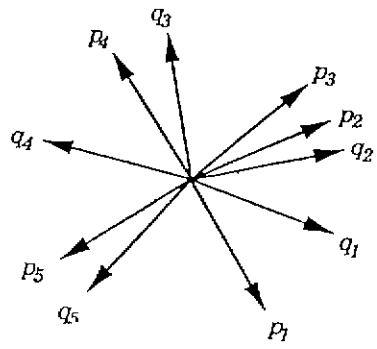


(그림 7) Selected broadcasting



a) P and Q b) θ(P)

(그림 8) Convex hull P and Q



PE	0	1	2	3	4	5	6	7	8	9
v	q_2	p_2	p_3	q_3	p_4	q_4	p_5	q_5	p_1	q_1

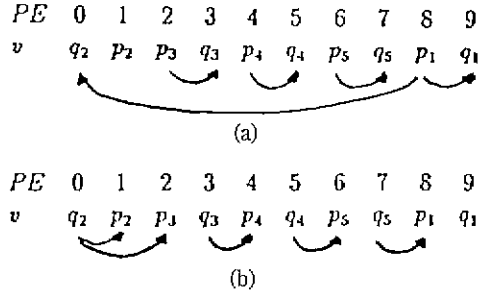
$M_{p_1} = \{(p_1, q_1), (p_1, q_2)\}$ $M_{q_1} = \phi$
$M_{p_2} = \phi$ $M_{q_2} = \{(p_2, q_2), (p_3, q_2)\}$
$M_{p_3} = \{(p_3, q_3)\}$ $M_{q_3} = \{(p_4, q_3)\}$
$M_{p_4} = \{(p_4, q_4)\}$ $M_{q_4} = \{(p_5, q_4)\}$
$M_{p_5} = \{(p_5, q_5)\}$ $M_{q_5} = \{(p_1, q_5)\}$

(그림 9) Computation of M

므로 M을 병렬로 계산할 수 있는 방법을 생각해 보도록 한다.

Convex polygon 각점 p에서 tangent line이 놓일 수 있는 범위는 그림 8의 (b)에서 보는 바와 같이 counterclockwise 방향으로 점을 배열할 때 이전의 점 pred(p)와 이후의 점 suc(p)와 이루는 점 Amin(p)와 Amax(p) 사이의 범위 θ(p)가 된다. 따라서 각각 P와 Q에 속한 임의의 두 점 p, q에 대해 θ(p)∩θ(q)≠0이고 p, q를 잇는 직선이 θ(p)∩θ(q) 사이에 놓이면 p, q간에 tangent line이 존재한다는 것을 쉽게 알 수 있다. 따라서 θ(p)∩θ(q)≠0인 모든 (p, q)의 쌍을 구할 수 있다면 그 중에서 tangent line이 지나는 두점 p, q를 구할 수 있게 된다.

θ(p)∩θ(q)≠0인 모든 (p, q)의 쌍을 병렬로 구하는 방법은 그림 8의 (a)에 대한 예를 들면서 그림 9에서 설명하기로 한다. 우선 각 PE에 들어있는 p와 q의 꼭지점은 Amin에 따라 그림 9과 같이 sort한 후 P의 각 점 p_i에 대해 p_i와 suc(p_i) 사이에 있는 Q의 점들과의 쌍의 집합 M_{p_i}를 구하고 Q의 각 점 q_i에 대해 q_i와 suc(q_i) 사이에 있는 P의 점들과의 쌍의 집합 M_{q_i}를 구한다면 θ(p)∩θ(q)≠0인 모든 쌍을 구할 수 있다는 것을 쉽게 알 수 있다. 따라서, merge step 이전에 이미 sort되어 있는 P와 Q에 있는 점을 merge한 후 그림 10와 같이 P의 각 점 p를 p와 suc(p) 사이에 있는 Q의 점들을 가지고 있는 PE에



(그림 10) Data Routing

보내고, 마찬가지로 Q의 각 점 q를 q와 suc(q) 사이에 있는 P의 점들을 가지고 있는 PE에 보낼 수 있으면 tangent line이 지날 수 있는 쌍의 집합을 구할 수 있는데, 이는 바로 앞에서 설명한 selected broadcasting 방법과 일치하므로 이를 이용하여 직접 구할 수 있다. 따라서, merge 시간 M(n)은 MCC에서 $O(\sqrt{n})$, CCC에서 O(log n)으로 전체적으로 걸리는 시간은 $T(n)=T(n/2)+M(n)$ 이므로 MCC에서는 $O(\sqrt{n})$, CCC에서는 $O(\log^2 n)$이 걸리게 된다. 이와같이 효율적인 routing기법을 이용하여 tangent line이 지날 수 있는 (p, q)의 쌍을 병렬로 하는 새로운 algorithmic parallelism의 발견이 효율적인 병렬알고리즘을 개발하는데 필수적이다.

V. 결 론

본 논문에서는 병렬컴퓨터구조에 따라 specialized된 병렬알고리즘설계에 관해 기술하였다. 순차처리와는 달리 프로세서간의 효율적인 데이타 교환이 이루어질 수 있도록 데이타 분배가 이루어져야 하고, 병렬구조 특성을 최대한대로 사용할 수 있도록 새로운 algorithmic parallelism의 분석이 효율적인 병렬알고리즘의 설계를 위해서는 필수적이다. 프로세서상에서의 데이타 분배가 전체병렬알고리즘의 성능에 중요한 factor가 된다는 것을 보이기 위해 liner class 병렬 컴퓨터상에서 TSP를 simulated annealing 방법을 사용하여 해를 구할 경우 효율적인 입력데이타들의 distribution 방법에 대해 예를 들어서 설명을 하였고, 프로세서간의 데이타 routing을 효율적으로 수행할 수 있는, 순차알고리즘과는 전혀 다른 algorithmic parallelism이 중요하다는 것을 mesh-connected와 cube connected 컴퓨터상에서 convex hull 문제를 다루면서 알아보았다. 주어진 병렬구조의 특성에 따라 병렬알고리즘의 기본 structure가 변화하기 때문에 실제적인 병렬알고리즘의 설계는 매우 복잡해지고 어렵지

만, 근본적으로 프로세서간의 데이타교환을 고속으로 수행할 수 있고 병렬 구조 특성을 잘 살릴 수 있는 데이타 분배방법과 algorithmic parallelism을 발견하는데 유념한다면 병렬알고리즘 설계시 많은 도움이 되리라고 생각한다.

참 고 문 헌

1. L. P. Kartasher and S. I. Kartasher, "Second Internationala Conference on Supercomputing." Vol. I, II, III, 1987, International Supercomputing Institute.
2. S. G. Akl, "The Design and Analysis of Parallel Algorithms," Prentice Hall, 1989.
3. L. Uhr, "Parallel Computer Vision," Academic Press, 1987.
4. H. Kobayashi, *et. al.*, "Parallel Processing of an Object Space for Image Synthesis using Ray Tracing," The Visual Computer, pp. 13~22, 1987.
5. M. Marrakchi and Y. Robert, "Optimal Algorithm for Gaussian Elimination on an MIMD computer," Parallel computing 12, pp. 183~194, 1989.
6. F. Dehen and A. Rau-Chaplin, "Implementing Data Structures on a Hypercube Multiporcessor, and Applications in Parallel Computational Geometry," J. Parallel and Distributed Computing 8, pp. 367~375, 1990.
7. Q. F. Stout, K. G. Shin, and T. N. Mudge "International Conference on Parallel Processing," Vol. I, II, III, CRT Press, 1992.
8. H. J. Siegel, "A Model of SIMD Machines and a Comparison of Various Interconnection Networks," IEEE Trans. Comput. C-28, pp. 907~917, 1979.
9. C. S. Jeong and M. H. Kim, "Fast Parallel Simulated Annealing for Traveling Salesman Problem on SIMD machine, with Linear Interconnections, Parallel Computing," pp. 221~228, 1991.
10. C. S. Jeong, and J. J. Choi, "Parallel Enclosing Rectangle on SIMD machines," Parallel Computing, pp. 221~229, 1992.

정 창 성

관심 분야 : 병렬처리

1981 서울대학교 전기과 졸업
1985 Northwestern University Computer Science M S
1987 Northwestern University Computer Science Ph D
1987 ~1992 포항대학교 전자공학과 조교수
1992 ~현재 고려대학교 전자공학과 조교수
1992 ~현재 Editorical Review Board, Parllel algorithms and applications