□ 기술해설 □

범구조적 병렬 계산 모델†

광운대학교 차호정*
고려대학교 김동승*
아주대학교 홍만표*

목 차

1. 서 론

최근 다양한 형태의 고성능 병렬컴퓨터가 개발되었고 병렬 프로그래밍 개념의 보편화가 이루어졌음에도 불구하고 병렬컴퓨팅은 아직 컴퓨터산업에 있어서 주된 흐름이 되지 못하고 있다. 그 주요인으로는 병렬 아키텍처의 다양성과 소프트웨어의 아키텍처 종속성을 들 수 있다. 병렬컴퓨터는 각 시스템마다 구조가 상이하기 때문에 효율적으로 범용 소프트웨어를 개발하기 어렵고 일관성 있는 알고리즘의 분석 및 성능 예측이 쉽지 않다. 급진적으로 발전하는 컴퓨터 제작기술을 고려할때 기존 병렬 소프트웨어의 아키텍처 종속성은 표준화된 병렬 아키텍처 구축과 병렬 프로그램 방식의 수렴에 장애 요소이다.

기존 순차 컴퓨터의 계산 모델은 폰 노이만 (Von Neumann) 모델을 기본으로 한다. 폰 노이만 모델은 순차컴퓨터용 소프트웨어와 하드웨어의 개발을 위한 통합 모델을 제공하여 순차 컴퓨터에 있어서 정형화된 복잡도 이론의 개발을 가능케 하였고, 통합 모델을 기반으로 다양한 알고리즘의 개발과 성능 평가 및 우수한 소프트웨어 개발기술의 향상을 초래하였다. 병렬 컴퓨터에서는 이와 같은 통합 계산 모델이 결여되어 다양한 구조를 가진 병렬 컴퓨터들이 제각기 발전되는 양상이 초래되었다. 결과적으로, 소프트웨어들이 하드웨어에 종속되는 결과를 낳게 되어 이기종간의 소프트웨어 이식이 거의 불가능한 상황에 이르렀고 사용자들은 하드웨어에 구조에 대한 지식을 갖추어야 하는 부담을 안게 되었다. 또한 여러 유형의 병렬 컴퓨터 구조를 반영하는 프로그래밍 언어의 결핍으로 인하여 개발된 프로그램들이 수행될 때 높은 성능을 얻기 어려운 경우가 대부분이다. 즉, 알고리즘의 복잡도와 해당 컴퓨터에서 수행되는 프로그램의 복잡도를 연관시키는 이론의 부족으로 병렬 알고리즘의 성능분석이 어렵고 서로 다른 컴퓨터에서 개발된 알고리즘들의 비교가 쉽지 않다. 병렬 컴퓨터가 안고 있는 이러한 문제점들은 부분적인 개선에 의한

† 본 연구는 한국과학재단의 특정기초연구 연구비 지원에 의한 것임.
*종신회원

해결에는 한계가 있으므로 계산 모델의 정립에서부터 문제 해결의 출발점을 찾아야 한다. 새로운 병렬 계산 모델은 병렬 컴퓨터들의 구조와는 독립적인 '범구조적 병렬 계산 모델'로서 고안되어, 기존의 순차 컴퓨터에서 폰노이만 계산 모델이 담당했던 역할을 병렬 컴퓨터 분야에서 수행함으로써 다양한 구조를 갖는 병렬 컴퓨터와 병렬 프로그램들이 효율적으로 연결될 수 있도록 밑받침되어야 한다.

최근, 병렬컴퓨터에서도 하나의 통합적 계산 모델 개발을 위한 노력이 다양하게 진행되고 있다. 병렬 컴퓨터에서 단일 모델로의 수렴에 대한 부정적인 견해가 있는것은 사실이나, 단일 주소공간을 지원하는 공유메모리 컴퓨터나 스위칭 네트워크를 사용하는 분산메모리 멀티컴퓨터들의 개발과 같은 병렬컴퓨터 제작기술의 복합화 추세와, 통합 계산모델을 사용함으로서 얻어지는 현실적인 이익 요소들로 인하여 최근 범구조적 병렬 계산 모델에 대한 관심이 커지고있다[1, 2]. 범구조적 병렬 계산 모델은 크게 두 방향으로 연구되고 있다. 즉, 기존의 대표적인 병렬 계산 모델인 PRAM 모델의 단점을 보완하여 PRAM의 변형 계산 모델들을 개발하는 연구와, PRAM 모델의 근본적 제약을 탈피하여 실제 구현 가능성이 높은 새로운 계산 모델에 대한 연구가 활발히 이루어지고 있다. PRAM을 탈피한 대표적인 계산 모델로는 HPRAM[3], LogP[4], C^3 [5], BSP[6] 등이 있다. 특히, Valiant가 제안한 BSP 모델은 병렬 소프트웨어와 하드웨어 개발의 기준이 되는 브릿징(Bridging)모델로 평가되어 많은 관심을 끌고 있다.

본 논문에서는 병렬 계산 모델들을 범구조적인 속성 보유여부에 따라 체계적으로 분류할 수 있는 기준을 제시한다. 이 기준을 바탕으로 여러 형태의 병렬 계산 모델들을 구분하며 범구조적병렬 계산 모델의 이론적, 실제적 특성을 기술한다. 본 논문의 구성은 2절에서 병렬 컴퓨터 모델의 일반적인 분류와 본 논문에서 사용하는 범구조적 병렬 계산 모델의 정의를 서술하고, 3절과 4절에서는 2절의 분류에 따른 이상적 병렬 계산 모델들과 범구조적 병렬 계산 모델들에 대해 각각 서술하며, 5절에서 결론을 맺는다.

2. 병렬 모델의 분류와 범구조적 병렬 계산 모델

병렬 컴퓨터 모델은 병렬 컴퓨터에 대한 인식 관점과 추상화 정도에 따라 기계 모델, 아키텍처 모델, 계산 모델, 프로그래밍 모델로 분류할 수 있다[3].

기계 모델은 병렬 컴퓨터 하드웨어를 설계하는 설계자의 관점을 반영하는 모델로서 병렬 컴퓨터를 구체적으로 어떻게 구현할 것인가에 관한 모델이다. 예를 들어, CPU의 구조와 명령 집합, 메모리의 구성, 캐쉬 메모리 기법, 운영 체제등이 정의되며 효율적인 구현을 위한 다양한 기법들이 서술된다. 아키텍처 모델은 기계 모델을 추상화하는 모델로서 병렬 하드웨어의 구조적 측면을 다룬다. 병렬 컴퓨터를 공유 메모리 형태로, 혹은 분산 메모리 형태로 구성할 것인가? 분산 메모리 아키텍처의 경우 상호 연결망의 구성을 어떻게 할 것인가? 패킷 경로 배정은 어떤 알고리즘을 사용할 것인가? 등의 문제를 포함한다. 이 분류에 의한 대표적인 모델로는 UMA(Uniform Memory Access), NUMA(Non Uniform Memory Access) 등의 공유 메모리 아키텍처와 하이퍼큐브, 메쉬 등과 같은 분산 메모리 아키텍처가 있다[7].

한편, 계산 모델은 아키텍처 모델을 기반으로 추상화된 병렬 아키텍처 모델로서 병렬 컴퓨터의 수행 방식과 흐름을 상위 관점에서 정의한다. 이를 이용하여 알고리즘 개발자는 특정 아키텍처에 종속되지 않는 추상화된 모델을 통해 아키텍처 독립적인 알고리즘을 개발하며, 알고리즘 상호간의 성능 비교 및 분석을 할 수 있다. 계산 모델은 단순한 구조를 바탕으로 설계되어 용이한 이론적 연구를 제공해야한다. 또한, 실제 아키텍처상에서의 알고리즘 성능을 반영할 수 있도록 병렬 컴퓨터의 핵심적인 요소들을 포함하여야 한다. 대표적 계산 모델로는 순차 컴퓨터의 RAM(Random Access Machine) 모델과 RAM 모델을 병렬 계산 개념으로 확장한 PRAM(Parallel RAM) 모델이 있

다. 프로그래밍 모델은 프로그래밍 언어 측면에서 병렬 컴퓨터를 서술하며 프로그래머가 소프트웨어를 개발하는 패러다임을 정의한다. 프로그래밍 모델은 계산 모델과 밀접한 관계가 있으며 계산 모델이 제공하는 계산 방식을 토대로 소프트웨어를 작성한다.

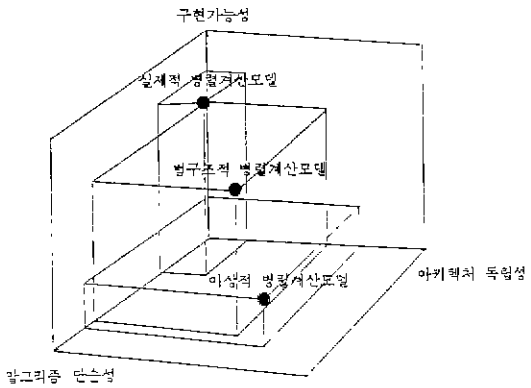


그림 1 병렬 계산 모델의 분류

위 병렬 컴퓨터 모델의 분류중에 '계산 모델'은 아키텍처 모델보다 추상적이고 단순한 관점을 제공하여 알고리즘의 개발이나 성능 분석 등의 연구에 많이 사용되어 왔다. 하지만, PRAM, Circuit 등과 같은 계산 모델은 구현하기 어려운 이상적인 모델로서 현실적인 병렬 소프트웨어 개발의 토대가 되지 못하고 있다. 또한, 현재 대부분의 프로그래밍 모델은 성능 효율을 위해 특정 아키텍처 모델을 계산 모델로 사용하는 아키텍처 종속적인 패러다임을 갖고 있어 이기종 아키텍처간에 소프트웨어 이식성이 적다.

최근에는 고전적인 병렬 컴퓨터 이론과 현실적인 구현 요소들을 함께 고려한 새로운 모델들이 제안되고 있다. 이들 모델들은 기존의 이상적 계산 모델들에 비해 다양한 아키텍처를 포괄하는 '범구조적' 성격을 띠고 있어 범용 병렬 소프트웨어 개발에 효율적인 기초를 제공한다. 본 논문에서 사용하는 범구조적이라는 용어는 Skillicorn[2]이 정의한 'universal'의 의미와 유사하다. Skillicorn은 계산 모델의 범구조성을 모델상의 성능 복잡도와 동일하게(즉, 상수배 차이만으로) 아키텍처상에서 구현 가능할 때, 아키텍처에 대하여 universal하다고 정의하였다. 예를 들어, PRAM 모델은 하이퍼큐브 아키텍처상에서 계산 모델의 복잡도와 동일하게 수행할 수 있으므로 하이퍼큐브 아키텍처에 대하여 universal하다[2]. PRAM 모델은 이론적인 범구조성에도 불구하고 실제 하이퍼큐브 아키텍처상에서 구현할 경우 높은 상수배 요소로 인하여 효율적이지 못하다. 본 논문에서는 복잡도 보존을 의미하는 Skillicorn의 universal 정의에 구현 효율성을 첨가하여 범구조성을 정의한다. 즉,

범구조성 = 복잡도 보존 + 효율성
(혹은, 낮은 상수배 요소)

로 정의한다. 이것을 바탕으로 지금까지 연구되어온 병렬 계산 모델들은 크게 다음의 세 종류로 분류될 수 있다.

- 이상적 계산 모델 : 이상적 아키텍처를 기반으로 하는 비현실적인 계산 모델
- 실제적 계산 모델 : 실제 아키텍처를 기반으로 하는 특정 아키텍처 기반 계산 모델
- 범구조적 계산 모델 : 범용 소프트웨어 개발을 위한 범구조적 아키텍처 기반 계산 모델

위 모델은 (1)알고리즘 단순성(Algorithmic Simplicity), (2)아키텍처 독립성(Architecture Independency), (3)구현 가능성(Implementation Feasibility) 등을 기준으로 구분된다. 알고리즘 단순성은 알고리즘 설계자가 용이하게 알고리즘을 작성할 수 있는 정도를 의미한다. 아키텍처 독립성은 하부 아키텍처 종속적인 요소들의 반영 정도를, 구현 가능성은 실제적인 아키텍처로의 구현 가능성과 성능 예측의 정확성을 반영한다. 그림 1은 이들 요소들을 축으로 하는 계산 모델의 특성을 보여준다. 이상적 계산 모델은 알고리즘 단순성과 아키텍처 독립성은 우수한 반면 구현 가능성이 작고, 실제적 계산 모델은 구현 가능성은 높으나 알고리즘 단순성과 아키텍처 독립성이 낮다. 한편, 범구조적 계산 모델은 알고리즘 단순성과 아키텍처 독립성 면에서 이상적 계산 모델보다 뒤떨어지나 구현 가능성이 높다.

다음은 대표적인 이상적 계산 모델 및 현재

까지 제안된 여러 범구조적 계산 모델에 관해 서술한다.

3. 이상적 병렬 계산 모델

이상적 병렬 계산 모델들은 추상화된 모델을 제공하여 병렬 알고리즘 연구의 기초를 제공하는 모델이다. 다음은 대표적인 이상적 계산 모델인 PRAM 모델과 Circuit 모델에 대해 기술한다.

3.1 PRAM 모델

PRAM 모델은 순차 컴퓨터에서 알고리즘의 개발과 분석에 사용하는 RAM 모델을 병렬 개념으로 확장한 것으로서 여러개의 RAM을 공유 메모리를 통해 연결한 것이다(그림 2). 각각의 프로세서들은 고유한 프로세서 ID를 가지며 동일한 클럭에의해 각 스텝마다 동기화되어 진행한다. 각 PRAM 프로세서는 한 스텝에서 다음의 동작을 수행한다.

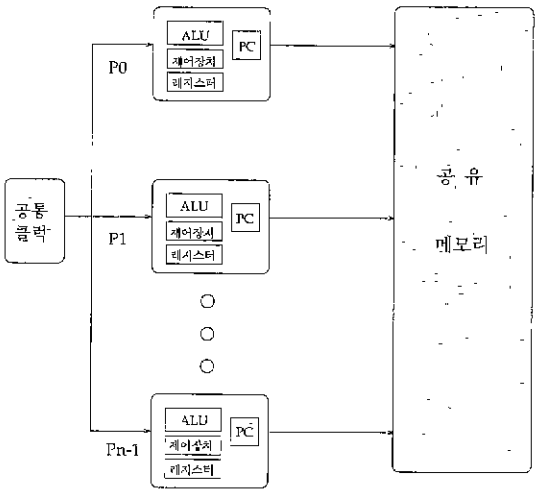


그림 2 PRAM 모델

- 공유 메모리에서 지역 메모리로의 읽기
- 지역 메모리에서 공유 메모리로의 쓰기
- 지역 메모리상에서 단일 연산 수행

그림 3은 PRAM 모델의 동작 과정을 보여준다. PRAM은 클럭을 공유하지만 SIMD 컴퓨터와 달리 프로세서들은 각 스텝에서 다른 명령어를 실행할 수 있는 MIMD 컴퓨터이다.

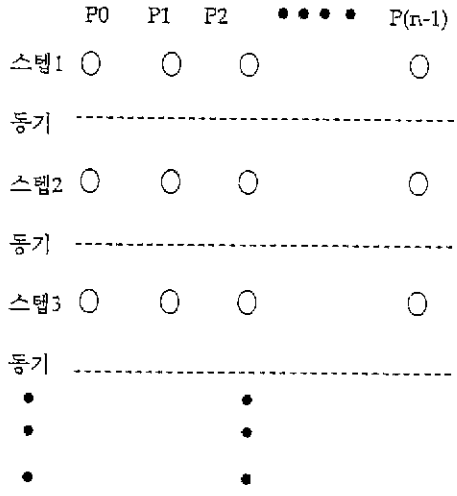


그림 3 PRAM 모델의 동작 과정

프로세서간의 통신은 공유 메모리를 통해 이루어지며 각 프로세서들은 동시에 공유 메모리로 단위 시간내의 접근이 가능하다.

PRAM 모델은 공유 메모리상에 동일한 위치로의 접근 방식에 따라 다시 네 방식으로 분류된다. EREW PRAM 모델은 공유 메모리의 접근시 서로 다른 위치에 동시 접근은 허용하나 동일한 위치로의 동시 접근은 허용하지 않는 모델로 PRAM 모델 중 가장 제한적인 모델이다. 반면, CRCW PRAM 모델은 동시 읽기와 동시 쓰기가 모두 가능하며 CREW PRAM 모델과 ERCW PRAM 모델은 각각 동시 읽기와 동시 쓰기만이 가능한 모델이다. 이중, 동시 쓰기가 가능한 모델은 해결 정책에 따라 다시 공통형, 임의형, 우선 순위형으로 분류된다. 공통형은 동시 쓰기를 원하는 프로세서들이 동일한 값을 기록할 경우 요구를 허용하고, 임의형은 프로세서들 중 임의의 한 프로세서를 선택하여 요구를 허용하며, 우선 순위형은 우선 순위가 높은 프로세서의 값을 기록하는 방식이다.

PRAM 모델을 바탕으로한 간단한 병렬 프로그램의 수행 및 성능 분석 예를 살표보자. 그림 4는 프로세서의 수가 4일때 PRAM 덧셈 알고리즘(*Sum*)의 수행 과정이다. 각 프로세서

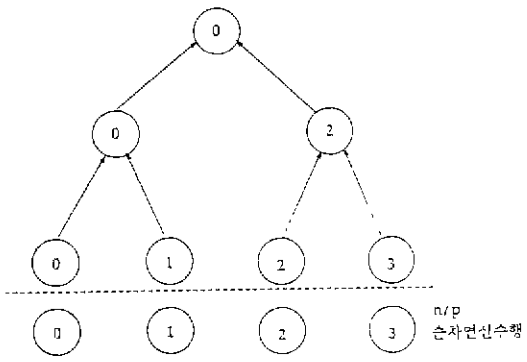


그림 4 PRAM 모델에서의 병렬 덧셈 알고리즘

들은 n/p ($p=4$)개의 데이타를 할당받아 순차 알고리즘을 적용하여 연산한 결과를 공유 메모리에 저장한다. 즉, 0번, 2번 프로세서는 1번, 3번 프로세서의 합을 각각 읽어 자신의 결과와 더한 후 공유 메모리에 결과를 기록한다. 0번 프로세서가 2번 프로세서의 결과를 합하여 기록하면 알고리즘의 수행이 종료된다. 이 과정에서 프로세서의 수가 p일때 $log(p)$번의 연산, 통신 과정이 수행되고 순차 덧셈 알고리즘의 복잡도는 $O(n/p)$이므로 PRAM 모델에서 덧셈 알고리즘의 복잡도는

$$T_{PRAM}(Sum)=O(n/p+log(p))$$

이다.

PRAM 모델은 구조적으로 매우 단순하여 병렬 알고리즘의 이론적 연구와 분석에 광범위하게 사용되어 왔다. 반면, PRAM 모델의 단점은 메모리 접근을 통한 통신 능력이 매우 막강한 것으로 인하여 현실적으로 구현 가능성이 낮다는 것이다. PRAM은 무한대의 대역폭과 단위 시간에 모든 프로세서들이 동시에 접근할 수 있는 공유 메모리를 가정하며 각 스텝마다 모든 프로세서들이 별도의 수행 부담없이 엄격히 동기화 되어 작동한다. 현재의 기술로 프로세서간의 통신이 단위 시간에 이루어지고 동기화의 부담이 없는 완벽한 PRAM 컴퓨터를 구현하는 것은 불가능하다. 이러한 PRAM의 문제점을 보완하기 위해 PRAM이 고려하지 않는 실제적인 아키텍처 요소들을 반영하여 PRAM을 변형한 유사 모델들이 제안되었으나 PRAM의 한계를 크게 벗어나지는 못하였다. 지금까지 연구된 PRAM의 변형 모델로는 Asynchronous PRAM(A-PRAM)[8], Phase PRAM[9], Local memory PRAM(LPRAM)[10], Block PRAM(BPRAM)[11] 등이 있다. A-PRAM은 PRAM 모델에 비동기성을 첨가한 모델로서 단일 공유 메모리를 통해 상호 통신하는 다수개의 프로세서들로 구성된다. 프로세서들은 PRAM 모델과 달리 비동기적으로 동작하며 공유 메모리 접근시 상호 통신하는 프로세서간에 명시적으로 동기 연산을 사용하여 동기화를 수행한다. Phase PRAM은 PRAM 모델의 엄격한 동기화에 따른 실제적인 비용을 감소시키기 위해 비동기성을 첨가한 모델이다. Phase PRAM 모델은 전체 연산을 다수의 Phase로 나누어 수행하며 Phase안에서는 프로세서들은 비동기적으로 동작하고 동기화는 Phase의 끝에서만 이루어진다. LPRAM은 공유 메모리와 지역 메모리의 접근 비용에 차등을 둔 모델로서 각 프로세서들은 지역 메모리를 갖고 있어 공유 메모리에서 읽은 데이타를 가능한한 지역 메모리상에서 연산을 수행하도록 함으로써 알고리즘의 수행 효율을 높인다. BPRAM은 LPRAM 모델에 블록 파이프라인 특성을 첨가한 모델로서 블록단위의 전송을 이용하여 메시지당 통신비용을 감소시킨다.

한편, PRAM 모델을 변형하여 새로운 모델을 개발하고자 하는 연구와 함께 PRAM을 실제 병렬 컴퓨터로 구현하고자 하는 연구 결과도 보고되었다. 예로써, SB-PRAM[12]은 공통 클럭을 사용하고 공유 메모리를 지원하는 하드웨어 플랫폼으로 PRAM 모델에 근접한 기능을 제공하며 PRAM 형태의 프로그래밍 방식을 제공하는 언어를 수행한다.

3.2 Circuit 모델

서킷(Circuit) 모델[1, 13]은 DAG(Directed Acyclic Graph) 형태로 계산을 표현하는 모델로서 입력 노드, 연산 노드, 출력 노드들로 이루어진다. 입력 아크(화살표)의 수가 0인 노드를 입력 노드, 출력 아크의 수가 0인 노드를 출력 노드라 하고 최대 두개의 입력 아크를 갖는 노드들이 연산 노드이다. 각 노드에서 수행

하는 연산의 수행 시간은 단위 시간에 완료됨을 가정하며 한 노드는 다른 노드의 데이타에 단위 시간에 접근할 수 있다. 서킷 모델은 동일한 문제를 여러 형태의 DAG로 표현할 수 있는데 그림 5는 네 개의 숫자를 합하는 서킷 알고리즘의 예를 보여준다. 서킷 알고리즘의 복잡도는 DAG의 깊이, 즉 입력 노드와 출력 노드 사이의 거리 중 최대값으로 표현된다. 그림 5에서 문제의 크기가 n일때 왼쪽 DAG의 복잡도는 $O(n)$이며 오른쪽 DAG의 복잡도는 $O(log(n))$이다.

서킷 모델은 단순하며 규칙적인 수학적 계산을 분석하는 데 용이한 모델이나 PRAM 모델과 같이 실제성을 고려하지 않는 이상적 모델이다. 서킷 모델은 노드간의 통신이 단위 시간에 수행됨을 가정하며 스케쥴링 문제와 프로세서 할당 문제를 고려하지 않는다. 또한, 알고리즘이 복잡할 경우 분석이 어렵다는 단점을 가지고 있어 특정한 문제의 해결에 주로 사용된다.

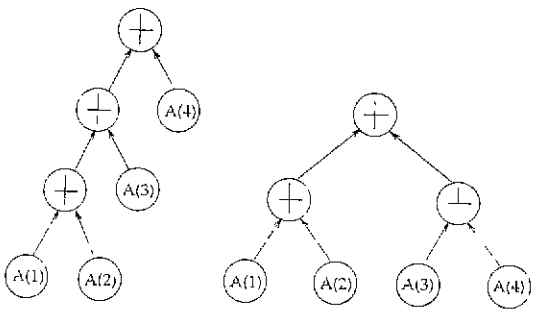


그림 5 Circuit 알고리즘의 예

4. 범구조적 병렬 계산 모델

범구조적 병렬 계산 모델은 크게 두 방향으로 연구되고있다. PRAM 모델을 변형, 일반화하여 구현 가능성이 높은 계산 모델을 개발하는 연구와, PRAM 모델을 변형하지 않고 PRAM 모델을 부모델로 사용하여 상위 레벨에서 병렬 컴퓨터를 표현하는 방법에 대한 연구다. 본 절에서는 PRAM을 일반화한 대표적 모델인 BSP 모델과 그의 변형 모델들을 기술한다. 또한, PRAM 모델을 부모델로 사용하는 H-PRAM 모델에 대해 살펴본다.

4.1 BSP 모델

BSP 모델은 PRAM 모델을 일반화한 모델로서 병렬 아키텍처의 실제적인 요소들을 반영하여 효율적인 알고리즘 개발을 가능케 한 모델이다. BSP 모델은 (1)지역 메모리를 가지는 프로세서의 집합, (2)일대일 방식으로 프로세서간에 메시지를 전달하는 상호 연결망, (3)프로세서간의 전역 동기를 위한 Bulk Synchronizer의 세 요소들로 구성된다. BSP 모델은 이중 구조의 메모리 모델로서 각 프로세서는 자신의 지역 메모리와 다른 프로세서의 메모리(비지역 메모리)에 접근할 수 있다. 비지역 메모리로의 접근은 위치에 상관 없이 동일 시각에 접근 가능하다. BSP 모델에서 동기화는 암시적으로 또는 명시적으로 수행할 수 있는데 동기화 연산 사이에 프로세서들에 의해 처리되는 일련의 작업을 '수퍼스텝'이라고 부르며(그림 6), 각 수퍼스텝은 지역 연산과 통신 연산으로 구성된다. 통신 연산은 수퍼스텝 동안에는 결과가 유효하지 않고 동기화가 이루어진 뒤에 유효하게 된다.

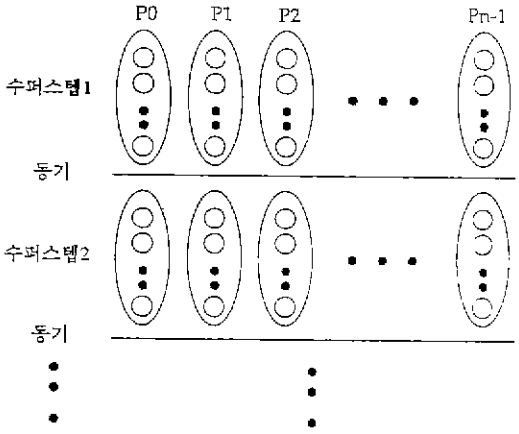


그림 6 BSP 수퍼스텝

BSP 모델은 다양한 형태의 하드웨어와 소프트웨어 아키텍처를 단일 병렬 컴퓨터로 추상화한다. 알고리즘 설계자는 하드웨어의 상세한 부분을 고려할 필요 없이 다음과 같이 정의되는 시스템의 특성 매개변수만을 고려하면 된

다.

- p : 프로세서의 수, 즉 시스템 크기
- s : 프로세서의 처리 속도
- l : 동기화 주기 또는 동기화를 수행하는 데 걸리는 시간
- g : 연산 처리율/통신 처리율

g는 상호 연결망의 대역폭과 관계된 값으로서 프로세서당 h개의 메시지를 전송하고 수신하는 통신 패턴인 h-*relation*의 통신을 $g \times h$ 단위시간에 수행할 수 있음을 의미한다. 매개변수 l과 g의 사용을 통해 다양한 아키텍처의 성능 특성을 표현할 수 있다. 대표적인 병렬 아키텍처의 l과 g의 값은 표 1과 같다.

표 1 l과 g의 예

상호 연결망	l	g
2차원 메쉬	$O(\sqrt{p})$	$O(\sqrt{p})$
버터플라이	$O(log(p))$	$O(log(p))$
하이퍼큐브	$O(log(p))$	$O(1)$

BSP 알고리즘은 다수의 수퍼스텝으로 구성되므로 각 수퍼스텝의 복잡도를 합하여 전체 알고리즘의 복잡도를 구할 수 있다[6]. 수퍼스텝 동안에 지역 연산을 가장 많이 수행하는 프로세서의 지역 연산 수를 n_{comp}, 최대 통신 연산 수를 n_{comm}이라 할 때, 단일 수퍼스텝의 복잡도 $T_{superstep}$은 다음과 같다.

$$T_{superstep} = n_{comp} + n_{comm} \times g + l$$

수퍼스텝의 복잡도로부터 S개의 수퍼스텝으로 이루어진 임의의 BSP 알고리즘 A의복잡도 $T_{BSP}(A)$를 다음과 같이 구할 수 있다.

$$T_{BSP}(A) = T_{superstep1} + \cdots + T_{superstepS}$$
$$N_{comp} + N_{comm} \times g + S \times l$$

여기에서 N_{comp}는 모든 수퍼스텝의 지역 연산 복잡도의 합이고, $N_{comm} \times g$는 통신 복잡도의 합이다. 3.1절에서 기술한 병렬 덧셈 알고리즘을 BSP 모델상에서 살펴보자. BSP 모델에서의 덧셈 알고리즘은 PRAM 모델과 유사하나 계산 방식이 수퍼스텝으로 구성되고 통신과 동기에 대한 비용이 복잡도에 반영되는 점에서 차이가 있다(그림 7). 그림에서 보듯이 각 BSP 프로세서들은 PRAM 모델처럼 $n/p(p=4)$개의 지역 연산을 수행한다. '수퍼스텝 1'에서 1, 3번 프로세서들은 자신의 합을 각각 0, 2번 프로세서에게 전달하고, '수퍼스텝 2'에서 2번 프로세서는 0번 프로세서에게 자신의 합을 전달하고 0번 프로세서는 최종합을 구한다. 이 과정에서, 프로세서의 수가 p일 때 알고리즘상 수퍼스텝의 총수는 $log(p)$이며 지역 연산의 복잡도는 $O(n/p)$이다. 각 수퍼스텝마다 1-*relation*을 수행하고 수퍼스텝의 총수가 $log(p)$이므로 BSP 덧셈 알고리즘의 최종 복잡도는

$$T_{BSP}(Sum) = O(n/p + log(p) \times g + \log(p) \times l)$$

이다($g=1, l=0$인 경우 PRAM 모델에서의 복잡도와 동일함을 볼 수 있다). 2차원 메쉬 형태의 아키텍쳐상에서 알고리즘이 수행될 경우, $g=O(\sqrt{p}), l=O(\sqrt{p})$이므로

$$T_{BSP}^{Mesh}(Sum) = O(n/p + \log(p) \times \sqrt{p})$$

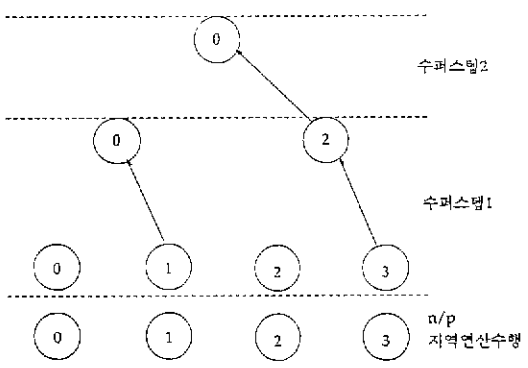


그림 7 BSP 모델에서의 덧셈 알고리즘

4.2 BSP의 변형 모델

BSP 모델은 모델의 단순성을 위해 성능 매개 변수 도입을 최소화한다. 이러한 BSP 모델을 기반으로 성능 매개 변수를 추가로 도입한 여러 모델들이 제안되었는데 그 중 대표적인 모델들을 서술하고 특징을 알아본다.

LogP 모델 LogP 모델[4]은 BSP 모델과 유사하게 병렬 컴퓨터를 지역 메모리를 갖는 다수의 프로세서들이 상호 연결망으로 연결된

형태로 표현한다. LogP 모델은 L, o, g, P의 네 가지 성능 매개 변수로 정의할 구 있는데 각 매개 변수는 다음의 의미를 갖고 있다.

- L : 한 워드(혹은 다수의 워드)를 포함하는 단일 메시지가 시작 프로세서에서 출발하여 목적 프로세서에 도착하는데 소요되는 시간의 상한값을 의미한다.
- o : 성능 부담 요소로서 메시지를 전송하고 수신하기 위해서 프로세서가 소비하는 시간을 의미한다. 이 시간동안에는 프로세서는 다른 연산을 수행할 수 없다.
- g : 한 프로세서에서 연속적으로 전송 혹은 수신하는 메시지들 사이의 최소 시간 간격으로서 g의 역수는 프로세서당 유용한 통신 대역폭을 의미한다.
- P : 프로세서-메모리 쌍의 갯수이다.

LogP 모델에서 각 프로세서들은 비동기적으로 동작하는데, 이러한 모델의 비동기성이 동기적인 특성을 갖는 BSP 모델과 대조된다. LogP 모델은 알고리즘의 성능 예측과 분석면에서 우수한 모델이다. 특히, 통신 스케쥴링을 통해 통신과 연산을 중첩시킴으로써 효율을 높일 수 있다. LogP 모델의 단점으로는 모델의 비동기적 특성에 의해 알고리즘의 단순성이 떨어지고 BSP 모델의 수퍼 스텝 구조와 같은 단순한 구조와는 달리 알고리즘 설계자가 전체 알고리즘을 종합적으로 고려해야 한다는 것이다. 또한, 알고리즘을 작성할때 임의의 순간 통신망에 한 프로세서가 L/g개 이상의 메시지를 전송하거나 수신하지 않도록 해야 하는데 이 한계를 넘으면 알고리즘의 성능이 저하되고 또한 그 성능을 정확히 예측할 수 없게 된다.

C^3 모델 BSP와 LogP 모델에서는 알고리즘 성능 분석에서 계산, 통신, 동기의 측면은 고려하지만 통신 연산시 발생하는 병목현상(congestion)은 고려하지 않는다. C^3 모델[5]은 성능 분석시 링크 병목과 프로세서 병목 효과를 고려함으로써 보다 정확한 성능 예측과 임의의 통신 연산의 성능 평가를 가능케 한다. C^3 모델의 작동 메카니즘은 BSP 모델과 유사한 수퍼스텝 구조로 구성되며 다음과 같은 성능 매개 변수들을 갖는다.

- p : 프로세서의 수
- h : 통신망의 latency로서 두 프로세서간의 평균 거리로 정의되며, $d_{i,j}$를 프로세서 i와 j의 최소 거리라 할 때 $\sum_{0 \le i,j \le p-1} d_{i,j}/p^2$로 표시된다.
- b : 통신망의 bisection 폭을 가리키며 전체 시스템을 동일한 크기의 두 부분으로 분할하기 위해 제거해야 하는 링크의 최소 수를 의미한다.
- s : 한 메시지의 셋업 비용. 메시지는 고정 길이의 패킷들로 구성되며 한 패킷은 두 프로세서 사이의 논리적인 통신 단위이다.
- l : 패킷의 길이를 표시하며 패킷을 구성하는 바이트 수를 의미한다.

WPRAM 모델 WPRAM 모델[14]은 BSP 모델과 달리 아키텍처 계층에서의 전역 동기 연산 지원을 가정하지 않으며 메시지 전송을 이용하여 상위 계층에서 전역 동기를 구현한다. 즉, 하부 아키텍처 상에서의 전역 동기를 가정하지 않으므로 광범위한 아키텍처를 포함할 수 있다. 또한, WPRAM 모델은 동일한 데이타로의 동시 접근을 위한 비동기적 결합(combining) 메카니즘을 가정한다. 결합 메카니즘은 하드웨어 혹은 소프트웨어로 구현할 수 있으며 이를 확장하면 다수의 동시 접근이 가능하다. WPRAM 모델이 제공하는 알고리즘 개발 방법은 알고리즘을 Strong PRAM 모델 상에서 개발한 후 실제적 요소를 고려하여 WPRAM 모델 상으로 알고리즘을 변형하는 것이다. 추상적 레벨에서 알고리즘을 개발함으로써 알고리즘 개발을 보다 용이하게 할 수 있으며 WPRAM 모델의 성능 매개 변수를 이용한 성능 분석을 통해 알고리즘의 효율을 높일 수 있다. WPRAM 모델은 BSP 모델의 성능 매개 변수를 확장하여 프로세스 레벨의 연산 비용도 제공한다. 하부 아키텍처 계층에서의 성능 매개 변수의 값은 표 2와 같다.

BDM 모델 BDM 모델[15]은 단일 주소 공간을 사용하며 비지역 메모리 접근을 위한 상호 연결망의 성능을 표시하는 매개 변수를 통해 병렬 아키텍처를 추상화한다. BSP 모델과 같이 전역 동기를 지원하며 메모리 지연, 통신 대역폭, 메모리 모듈에 접근시 연속된 m

표 2 WPRAM 모델의 성능 매개 변수 비용

Operation	Cost
network latency	$D=O(log(P))$
network bandwidth	$O(P\times log(p))$
machine granularity	$g=O(1)$
access of X remote words	$O(D+X)$
local operation	$O(1)$

워드를 사용하는 공간 지역성(Spatial Locality)의 성능을 반영한다. 예를 들어, 단일 비지역 메모리 접근에 소요되는 비용은 $\tau+m\sigma$로 표시된다. 여기에서 p는 프로세서의 수, τ는 요구 프로세서가 요구한 패킷을 수신하기 까지의 최대 지연 시간, σ는 한 프로세서가 통신망에 한 워드를 삽입할 수 있는 비율, m은 연속된 m개의 워드를 의미한다. BDM 모델은 단일 메모리 모듈 접근시 공간 지역성과 파이프라인 프리페치를 이용하여 통신 성능을 향상시키고 있으나 모델의 단순성을 위해 프로세서간의 지역성은 이용하지 않는다.

4.3 H-PRAM 모델

Hierarchical PRAM(H-PRAM) 모델[3]은 PRAM 모델의 비현실성을 보완하기 위해 PRAM 모델을 부모델로 사용하여 PRAM의 특성을 유지하는 동시에 동기와 통신 비용을 모델에 반영함으로써 알고리즘의 성능 예측을 가능케 한 모델이다. 또한, H-PRAM 모델은 계층 구조를 통해 전체 시스템을 다수의 부시스템으로 분할하여 부시스템들이 서로 비동기적으로 동작하게 하고 통신이 부시스템 내에서 이루어지도록 한다. 즉, H-PRAM 모델은 프로세서간의 비동기성과 지역성을 이용할 수 있게 한다(그림 8).

H-PRAM 모델은 PRAM 모델의 단순성을 유지함과 동시에 PRAM 모델을 부모델로 사용하여 PRAM 모델상에서 이루어진 연구 결과들을 이용할 수 있다. 계층적 구조를 통해 모델의 구조가 PRAM 모델보다 복잡해지나 모듈 지향적인 알고리즘 개발 방식과 융합될 수 있는 계산 모델이다. H-PRAM 모델을 구현하기 위해서는 하부 아키텍처의 구조가 동적

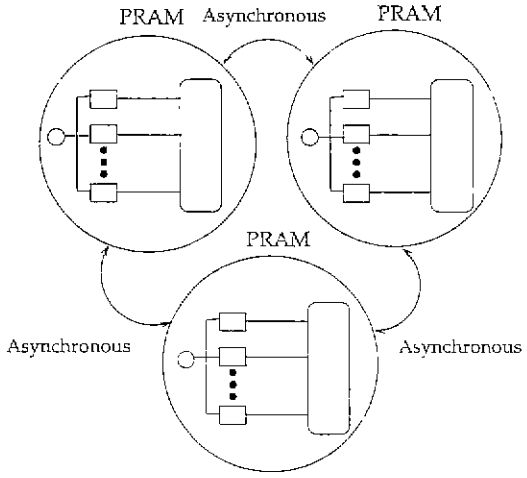


그림 8 H-PRAM 모델

으로 분할 가능해야 하는데 재귀적으로 분할 가능한 아키텍처로는 메쉬, 하이퍼큐브, 스타 그래프등이 있다. H-PRAM 모델의 가장 큰 특징은 전체 시스템을 소규모의 블록으로 분할하여 블록내에서 통신과 동기가 이루어지도록 하여 통신과 동기화의 비용을 최소화 하는 것이다. 일반적으로 병렬 아키텍처들은 시스템 크기가 증가할수록 효율성을 유지하기 위해 통신 하드웨어 성능 향상을 요구한다. H-PRAM 모델은 지역 통신을 수행하는 PRAM 들을 다수개 사용함으로써 필요한 통신 성능의 증가 요구 부담을 줄여준다. 즉, H-PRAM 모델은 소규모의 PRAM들을 사용함으로써 프로세서간의 지역성을 이용하여 성능 향상을 이루는데, 이는 분산메모리 아키텍처상에서 PRAM 모델을 실행할때 발생하는 지역성의 감소로 인한 시스템 성능 저하에 핵심을 둔 것이다.

3.1절에서 기술한 병렬 덧셈 알고리즘을 H-PRAM 모델상에서 살펴보자. H-PRAM 모델에서의 덧셈 알고리즘은 그림 9와 같이 계층적 구조를 갖는 PRAM 그룹들의 연산으로 구성된다. 최하위 레벨의 PRAM 그룹에서는 순차 알고리즘을 사용하여 n/p개의 데이타를 합한 후 상위 레벨의 PRAM 그룹으로 그룹간의 결합이 이루어진다. 상위 레벨에서는 동일한 방법으로 데이타의 교환이 반복되며 최종적으로 단일 PRAM 그룹으로 결합된 후 연산이 종료

한다. 이때 레벨 1의 PRAM 그룹을 제외한 PRAM 그룹들은 소규모의 부시스템을 구성하므로 전체 시스템보다 감소된 통신 지연을 갖는다. 2차원 메쉬 형태의 아키텍처상에서 H-PRAM 덧셈 알고리즘의 복잡도를 구하면 다음과 같다[3].

$$T^{Mesh}_{H-PRAM}(Sum)=O(n/p+\sqrt{p})$$

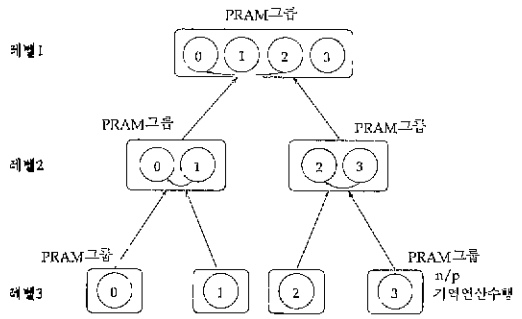


그림 9 H-PRAM 모델에서의 덧셈 알고리즘

5. 결 론

병렬 계산 모델은 알고리즘 개발단계에서 실행 대상의 컴퓨터에 구애받지 않고 개념적인 설계를 가능케 하고 그 결과 및 성능을 가늠할 수 있게 하여 해당 알고리즘/프로그램의 우수성을 사전에 평가해줄 수 있는 역할을 맡는다. 또한 특정 컴퓨터에서 실행되는 경우에 몇개의 파라미터로 성능을 구할 수 있게 함으로써 실현 가능성, 실제 구현시의 효과를 분석, 추정하는데 중요한 역할을 한다. 현 시점에서 바라는 병렬 계산 모델은 (1)알고리즘이나 프로그램 개발때 쉽게 활용할 수 있도록 병렬계산을 단순화하여 나타내고, (2)특정 아키텍처에만 부합하는등의 하드웨어 의존성이 적고, (3)주어진 알고리즘이 실제 병렬컴퓨터로 구현, 적용되는 과정을 정확히 표현할수 있도록 하고, (4)실제 시스템으로의 구현가능성이 높아야 하는 등의 다양하고 또한 상충적인 요구사항을 만족해야 한다. 하지만, 이러한 요건들에 전적으로 부합하는 계산 모델의 고안은 현실적으로 불가능하다. 본 논문에서 소개된 여러 모델은 이상적인 모델에 근접하는 최적모델을 고안하려는 시도로 각각 장점 및 약점을 동시에 갖고 있어 병렬 계산 모델의 현실적 한계성을 보여주는 것이다.

현재 범구조적 병렬 계산 모델들에 대한 연구되고 활발히 진행되고 있으나 전반적인 동향은 이론적인 연구에 치우치고 있으며 제시된 모델을 구현한 병렬컴퓨터에 대해서는 보고된 바가 없다. 이론적인 연구에서도 아직 어떤 병렬 계산 모델이 구현 가능성이 큰 범구조적 계산 모델인가에 대해서 논란이 많다. 따라서 이론적인 연구와 병행하여 제시된 범구조적 병렬 계산 모델들을 면밀히 분석하고 실제와 유사한 컴퓨팅 환경에서 구현하고자 하는 방향의 연구가 절실히 요구된다. 이를 통해서 기존 모델의 제약성 및 우수성을 검증하고 분석하여 보다 현실적인 범구조적 병렬컴퓨터를 위한 최적화 설계 지표를 제시할 수 있을 것이다.

참고문헌

[1] W. F. McColl, General Purpose Parallel Computing, Lectures on Parallel Computation, Edited by A. Gibbson and P. Spirakis, Cambridge University Press, 1993, pp.337-391.

[2] D. B. Skillicorn, 'Architecture-Independent Parallel Computation', IEEE Computer, Vol.23, No.12, December 1990, pp.38-50.

[3] T. Heywood and S. Ranka, 'A Practical Hierarchical Model of Parallel Computation : 1.The Model' Journal of Parallel and Distributed Computing, Vol 16, 1992, pp. 212-232.

[4] D. Culler and et al, 'LogP : Towards a Realistic Model of Parallel Computation', Proc. of 4th ACM SIGPLAN Symp. on Principles and Practices of Parallel Programming, 1993, pp.1-12.

[5] S. E. Hambrusch and A. A. Khokhar, C^3 : A Parallel Model for Coarse grained machines, Technical Report, Purdue University, January 1995.

[6] L. G. Valiant, 'A Bridging Model for Parallel Computation', Communication of the ACM, Vol.33, No.8, August 1990, pp.103-111.

[7] K. Hwang, Advanced Computer Architecture : Parallelism, Scalability, Programmability, McGraw-Hill International Editions, 1993.

[8] P. B. Gibbons, The asymchronous PRAM : A semi-synchronous model for shared memory MIMD machines, Ph.D. Thesis, Computer Science Division, University of California, Berkeley, December. 1989.

[9] P. B. Gibbons, 'A More Practical PRAM Model', In Proeedings of the ACM Symposium on Parallel Algorithms and Architectures), 1989, pp.158-168.

[10] A. Aggarwal, A. K. Chandra and M. Snir, 'Communication Complexity of PRAMs', In Theoretical Computer Science, March 1990, pp.3-28.

[11] A. Aggarwal, A. K. Chandra and M. Snir, 'On Communication Latency in PRAM Computation', In Proceedings of the ACM Symposium on Parallel Algorithms and Architectures, June 1989, pp.11-21.

[12] C. Kebler and H. Seidl, 'Making FORK Practical', Workshop on Models of Parallel Computation, Universiteit Utrecht, January 1995.

[13] J. F. JaJa, 'An Introduction to Parallel Algorithms', Addison-Wesley Publishing Company, 1992, pp.7-9.

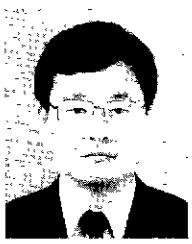
[14] J. M. Nash, P. M. Dew, M. E. Dyer, and J. R. Davy, Parallel Algorithm Design on the WPRAM Model, Technical Report, University of Leeds, July 1994.

[15] J. F. JaJa, K. W. Ryu, The Block Distributed Memory Model, Technical Report, University of Maryland, 1993lpr4.

차 호 정

1985 서울대학교 컴퓨터공학과 학사
1987 서울대학교 컴퓨터공학과 석사
1991 영국 University of Manchester 컴퓨터공학 박사
1991～1993 University of Manchester 연구원
1993～현재 광운대학교 전자계산학과 조교수
관심분야 : 병렬처리 시스템, 멀티미디어 서버, 넷트웍 컴퓨팅

김 동 승



1978 서울대학교 전자공학과 학사
1980 한국과학원 전기및전자공학과 석사
1988 남캘리포니아대 컴퓨터공학 박사
1980～1983 경북대학교 전임강사
1988～1989 남캘리포니아 강사,연구원
1989～1995 포항공대 전산학과 조·부교수
1995～현재 고려대학교 전기·전자 전파공학부 부교수
관심분야 : 병렬처리, 병렬 알고리즘, 컴퓨터 구조

홍 만 표

1981 서울대학교 자연대학 계산통계학과 이학사
1983 서울대학교 자연대학원 계산통계학과 이학석사
1991 서울대학교 자연대학원 전산과학과 이학박사
1983～1985 울산공과대학 전자계산학과 전임강사
1985～현재 아주대학교 정보 및 컴퓨터 공학부 부교수
1993～1994 미네소타 대학 전기전자공학과 교환교수
관심분야 : 병렬처리컴퓨터, 병렬계산모델, 병렬컴파일러