□기술해설□

고성능 병렬 언어에서의 자료 병렬성과 태스크 병렬성

안양대학교 박성순*
고려대학교 구미순** · 최성욱**
고려대학교 육현규** · 박명순***

목 차

1. 서 론

상용화된 병렬 컴퓨터 시스템을 지원하기 위한 프로그래밍 도구(programming tool)들은 컴퓨터 회사나 하드웨어 환경에 종속되어 왔다. 그런데 컴퓨터 회사 또는 하드웨어 환경에 종속되지 않는 도구, 즉 이식성(portability)있는 도구의 부족으로 인하여 병렬 컴퓨터 분야에서의 발전이 제한되어 왔다. 그래서 사용자는 하드웨어 분야가 급속히 발전하고 있음에도 불구하고 앞으로 개발될 하드웨어에 적합한 형태로 쉽게 바꿀 수 있는 소프트웨어를 사용하지 못하고 있다.

또한 병렬 컴퓨터 환경에서 프로그래밍하는 것이 너무 어렵고 많은 노력을 필요로 하기 때문에, 사용자는 발전된 하드웨어 환경을 효과적으로 사용하지 못하고, 다만 일반적인 단일 처리기(processor) 시스템에서 프로그래밍하고 있다. 이러한 괴리(乖離)의 원인은 다중 쓰레드(multi-thread) 프로그래밍 모델의 어려움과 순차 프로그램(sequential program)의 자동적인 병렬화(automatic parallelization) 노력의 실패에 기인한다고 볼 수 있다. 다중 쓰레드 형태로 프로그램을 수행하기 위해서는 프로그래머가 다중, 즉 병렬적으로 수행되는 독립적인 명령어(instruction)들을 구분하여 프로그래밍할 수 있어야 하나, 프로그래머가 독립적인 명령어들을 구분하여 프로그래밍하는 것은 쉬운 일이 아니다.

이렇게 병렬적이며 독립적인 쓰레드를 프로그래머 입장에서 구분하여 사용하기 어렵기 때문에 경쟁 상태(race condition)나 교착상태(dead lock)같이 '시간성(timing)'에 민감한 오류를 피할 수 없었다. 따라서 단순히 효율적인 측면은 제외하고라도 시간적인 측면만을 고려해 볼 때, 사용자가 순차 프로그래밍에서 병렬 프로그래밍 환경으로 넘어가는 것이 매우

*정 회 원
**학생회원
***종신회원

어렵다.

그리고 효과적인 병렬 수행을 위해 순차 프로그램을 컴파일하는 것 또한 여전히 어려운 문제로 남아있다. 여기서 발생되는 많은 문제들을 병렬적, 효과적으로 해결하기 위해서는 많은 새로운 알고리즘들을 필요로 하는데, 컴파일러 분야에서 이러한 알고리즘을 제안하고 구현하는 것까지 기대하기는 어렵다. 순차 프로그램의 효과적 병렬 구현이 기본이 되기는 하지만, 현재의 컴파일러는 이러한 구현까지는 이르지 못하고 있다. 이것은 분산 메모리(distributed memory)를 가진 컴퓨터 시스템에 대해서도 적용되며, 이러한 시스템의 컴파일러는 병렬성을 유지해야 할 뿐 아니라 효과적인 자료구조(data structure) 전략도 결정해야 한다[26].

최근 이러한 어려움을 해결하기 위해 다양한 방안들이 제안되었는데, 그 중 하나가 자료 병렬 프로그래밍 모델(data parallel programming model)에 근거한 언어들이다[4, 5, 9, 16, 18, 19, 20, 23, 29, 30]. 자료 병렬 모델은 개념적으로 단일 쓰레드 실행(single thread execution) 방식을 기반으로 하고 있는데, 입력 자료들에 대해 동일 연산을 적용시키는 형태로 병렬성을 제공한다. 즉, 자료 병렬언어에서는 하나의 연산으로 2개의 피연산자를 가지고 결과를 산출하는 벡터형의 덧셈 방법을 제공한다.

SIMD(single instruction multiple data) 방식에서 힌트를 얻은 자료 병렬 형태의 동기화 특성은 앞에서 언급한 시간성 문제로부터 발생할 수 있는 오류 가능성을 제거하여 준다. 또한 자료 병렬 모델은 단일 쓰레드 형태이므로 프로그래머가 디버거(debugger)같은 도구를 사용할 때, 순차 시스템하에서 사용하는 것과 같은 느낌을 갖게 한다. 따라서 순차 프로그래밍에서 병렬 프로그래밍 환경으로 부드럽게 넘어가도록 한다.

자료 병렬 모델의 사용은 좋은 성능의 컴파일러 구현을 위해 필요한 문제를 해결하는 방법이 될 수 있다. 따라서 자료 병렬 모델에 근거한, 그러면서도 컴퓨터 회사 또는 하드웨어에 종속되지 않는 이식가능 병렬 프로그램 도구가 가까운 시일내에 많은 시스템상에서 상용화되리라고 기대된다.

이러한 흐름에 맞춰 HPFF(High Performance Fortran Forum)가 1992년에 구성되었다[20]. 이 포럼은 다양한 종류의 고성능 컴퓨터 시스템에 맞는 강력한 응용 프로그램을 지원하기 위해, 기존의 포트란 90을 확장한 준(準) 표준을 만들기 위해 구성되었다. 여기서 고성능 포트란(High Performance Fortran : 이하 HPF와 혼용)의 준(準) 명세(specification)를 내놓았다[20]. HPF는 포트란 90의 자료 병렬 구조를 이용하고 있고, 포트란 90에서는 제공하지 않는 FORALL 문이나 다양한 지시어(directive)를 제공함으로써 전체 배열에 대해 연산을 가능하게 한다. 자료 병렬성을 제공하는 언어인 HPF를 비롯해 비슷한 시기에 제안된 유사 기능의 언어들인 CM-포트란, 포트란 D, PCF 포트란, 비엔나 포트란등이 있다. 이는 고성능 병렬처리 컴퓨터를 위해 효율적인 컴파일을 허용하기 때문에 고성능 언어(High Performance Language)라고 한다[9, 23, 29, 32]. 이러한 HPF 및 유사 언어들의 제안은 병렬 연산 분야에서 의미있는 사건이었다. 그중 HPF는 컴퓨터 회사나 사용자 모두에게 폭넓게 받아들여진 첫번째 병렬언어이다.

그러나 자료 병렬 프로그래밍 모델을 기반으로 한 고성능 언어들이 모든 프로그래밍 문제를 해결하지는 못한다. 이제까지의 연구가 고성능 연산 환경에서 쉽게 해결할 수 있는 영역에 촛점을 맞추어 왔기 때문에, HPF같은 자료 병렬 프로그램은 자료구조, 연산, 자료 종속성(data dependence) 등에서 동질성(homogeneity)를 찾는데에만 주력해 왔다. 따라서 규칙적인 자료구조를 갖는 요소들의 공통 연산을 적용하는 방식의 계산이 자료 병렬 프로그램을 이용하여 표현될 수 있었다.

그런데 고성능 컴퓨터에서 보다 복잡하고 전문적인 문제를 풀 때, 자료구조, 연산, 자료 종속성 등의 측면에서 볼 수 있는 이질성(heterogeneity)이 존재한다[14]. 즉 '상이한 프로그램과 자료를 수행하기 위해 서로 다른 처리기를 필요로 하는 형태'의 이질적인 응용 프로그램은 자료 병렬 계산을 위한 포트란 언어의 표준형으로 제시된 HPF 및 유사 언어들에서는 효과적으로 수행되지 못한다. 이러한 이질성이 자료

병렬성의 사용을 방해하지는 않지만, 이질성의 정도에 따라 자료 병렬 프로그램은 보다 더 복잡해질 수 있다[19]. 이러한 어려움을 해결하기 위한 방안으로 태스크 병렬 프로그래밍 모델(task parallel programming model)에 근거한 언어들이 다양하게 연구되었다[2, 7, 11, 12, 24, 27].

이러한 태스크 병렬성을 사용하여 병렬 알고리즘을 표현하면 보다 용이하고 효율적인 방안을 발견할 수 있다[14]. 제어 병렬성(control parallelism) 또는 함수형 병렬성(functional parallelism)이라고도 일컫는 이 태스크 병렬성은 프로그래머의 제어하에 프로그램의 동기화(synchronization)나 통신(communication) 등을 나타내는 제어 흐름이나 태스크의 다중 쓰레드를 명시적으로 보이고 있다.

고성능 컴퓨터를 사용하여 보다 복잡한 문제를 해결하거나, 과학자나 공학자가 보다 정교하게 작성된 알고리즘을 구현하고자 할 때, 자료구조 및 연산이나 자료 종속성 측면에 있어 많은 이질성을 가지고 있다. 이러한 이질성이 자료 병렬성의 사용을 방해하지는 않으나 이질성의 정도에 따라 자료 병렬 프로그램이 상당히 어려워질 수도 있고, 비효율적인 형태가 될 수 있다. 따라서 이러한 병렬성을 표현하는 알고리즘을 명시적으로 기술하기 위한, 보다 간단하고 효율적인 방안으로서 태스크 병렬구조를 사용할 수 있다[12].

그러므로 고성능 병렬언어가 이러한 자료 병렬성과 태스크 병렬성을 모두 포함하게 되면, 고성능 언어가 갖추게 되는 융통성(flexibility)과 이식성의 특성을 기반으로 응용 소프트웨어 분야에서 보다 다양한 표현성을 제공하게 될 것이다. 태스크 병렬언어와 자료 병렬언어를 통합하고자 하는 연구들도 시작되어 수행되고 있다[11, 27].

따라서 본 고에서는 현재까지 진행되어 온 고성능 병렬언어를 자료 병렬성을 지원하는 언어와 태스크 병렬성을 지원하는 언어로 구분하여 연구동향을 살펴본다. 그리고 고성능 병렬언어들 중에 HPF, 포트란 D, 비엔나 포트란 등의 자료 병렬언어와 포트란 M, Fx 포트란 등의 태스크 병렬언어에서 표현되는 자료 병렬성과 태스크 병렬성 관련 구조 및 특성을 살펴 보고자 한다.

그리고 좀더 확장된 개념으로 이 두가지 병렬성을 모두 지원하기 위해 통합된 환경을 구축하고자 할 때, 가능한 연구방향 및 기존의 연구, 고려사항 등을 살펴 보고자 한다.

2. 고성능 병렬언어

병렬 컴퓨터에서 보편적으로 사용가능한 프로그래밍 패러다임이 개발되지 않았기 때문에 오늘날 병렬 컴퓨터 사용에 있어 기본적인 문제는 병렬 소프트웨어 개발이 어렵다. 어떤 프로그램은 상호작용하는 large-grain 태스크 집합으로, 어떤 프로그램은 큰 자료구조의 요소들 상에서 수행되는 많은 fine-grain 연산으로 간주된다. 따라서 동일한 패러다임이 한 구조상에 구현되기는 쉬우나 다른 구조상에서도 효율적으로 구현되기는 매우 어렵다.

그래서 병렬 컴퓨터 사용자들은 고성능 병렬 컴퓨터에 효율적으로 적용할 수 있는, 단순하고 컴퓨터 구조에 독립적인 프로그래밍 패러다임을 필요로 한다. 이러한 프로그래밍 패러다임에는 크게 자료 분할의 특성을 가진 자료 병렬언어와 태스크 분할의 특성을 가진 태스크 병렬언어가 있다.

2.1 자료 병렬언어

자료 병렬언어는 과학 및 공학 연산의 특징인 동질성을 이용하는 것으로, 프로그램은 거의 모든 요소들에 방대한 양의 자료구조가 똑같이 적용되는 일련의 연산으로 구성된다. 자료구조를 어떻게 물리적 처리기에 분산할 것인가를 명시적 또는 묵시적인 병렬성으로 나타낸다. 자료 병렬성의 주된 장점은 확장성이다. 많은 자료에 연산이 병렬로 똑같이 적용되기 때문에, 병렬성의 정도는 문제 크기에 따라 달라진다. 많은 양의 계산을 하는 큰 문제를 해결함으로서 보다 높은 병렬성을 얻을 수 있다. 자료 병렬성은 일정하기 때문에, 사용자가 프로세스, 통신, 동기화를 명시적으로 관리할 필요없이 컴파일러에 의해 자동적으로 탐지될 수 있다.

이러한 대표적인 자료 병렬언어로는 포트란

90, 포트란 D, HPF, CMF, 비엔나 포트란, C*, Dataparallel C, Dino, Kali, Paragon 등이 있고[9, 17, 18, 22, 25, 29], 자료 병렬성을 지원하는 컴파일러로는 Adapt, Adaptor, Aspar, Forge90, Id Nouveau, Superb 등이 있다[1, 3, 6, 21, 31].

2.1.1 포트란 D

Rice 대학의 K. Kennedy 등에 의해 제시된 포트란 D는 기존의 포트란에 풍부한 자료 분할 명세기능을 강화한 것이다[16, 29]. 간단하면서도 하드웨어 독립적인 병렬 프로그래밍 모델을 제공하는 것이 포트란 D의 목적이다. 하드웨어에 종속적인 최적화 부담을 컴파일러에게 넘겨 프로그래머는 다양한 병렬 구조상에서 좋은 성능으로 컴파일되고 실행될 수 있는 자료 병렬 프로그램을 쉽게 작성할 수 있다.

포트란 D 자료 병렬 프로그램은 다음의 기본적인 두 단계로 자료 분할문제를 해결한다. 첫번째는 문제 사상(mapping) 단계로서 배열들이 어떻게 서로 정렬되어야 하는가를 다루는 단계이다. 프로그램 실행시 자료 이동요구를 최소화하도록 하며, 이 단계는 하드웨어에 거의 독립적이다. 두번째는 하드웨어 사상 단계이다. 배열들이 실제 병렬 컴퓨터상에 어떻게 분산되어야 하는가를 다루는 단계로서 유한한 실제 컴퓨터 자원으로의 변환을 일으킨다. 하드웨어 사상은 사용되는 컴퓨터의 토폴로지(topology), 통신 매체, 지역메모리의 크기, 처리기 수에 의존하므로 하드웨어 종속적이다. 자료 분할을 나타내기 위한 이 두 단계 전략이 과학자들에게 자연스럽고, 모듈성과 이식성있는 코드를 이끌어내기 쉽다.

2.1.2 HPF

HPF는 포트란 77과 포트란 90을 기본 골격으로 하여 확장된 포트란 언어이며, 포트란 D와 비엔나 포트란의 특성을 포함하고 있다[4, 23]. 자료 병렬성을 고려하여 배열에 대한 각종 연산을 지원하고, FORALL 문과 같은 자료 병렬 문법구조를 가지고 있다. 그리고 다양한 지시어를 통해 사용자가 자료를 분배할 수 있도록 하고 있다. HPF가 포트란 90에 비해 가지고 있는 새로운 특징으로는 새로운 지시어(DISTRIBUTE, ALIGN, etc), 새로운 문법(FORALL 문 등), 고성능 연산에서 필요한 라이브러리 루틴의 추가된 것이다.

HPF 포럼에서는 HPF 컴파일러 구현이 복잡할 것으로 예상하여 빠른 시일내에 초기 컴파일러의 release가 가능하도록 하기위해 HPF 표준 subset을 결정하였다. 이 표준 subset은 고성능 연산과 관련이 없다고 판단되는 부분 -예를 들어 소스 형식의 자유화 -을 제외한 것이다.

2.1.3 Dataparallel C

Dataparallel C는 Rose와 Steele이 설계한 C*언어의 변형으로, C*언어에 가상 토폴로지 개념을 추가하고, 포인터의 개념을 확장한 것이다[18].

Dataparallel C의 개념적 모델은 back-end 병렬 처리기와 연결된 front-end 단일 처리기 형태이다. 일반적인 C 코드를 포함한 Dataparallel C 프로그램의 순차 부분은 front-end에서 실행되고, 일반적인 C에서 지원하지 않는 구조로 작성된 Dataparallel C 프로그램의 병렬 부분은 back-end에서 실행된다. 이때 back-end에서 실행되는 병렬 부분에서 사용할 처리기의 수를 프로그래머가 선택할 수 있도록 하여 융통성을 제공한다. 처리기 수는 실제 하드웨어에서 사용가능한 물리적 처리기 수와는 무관하다. Dataparallel C 프로그램의 순차부분은 C 코드이므로 일반적인 C 언어 시맨틱에 따라 실행된다. Dataparallel C 프로그램의 병렬 부분은 MPC(Master Program Counter)의 제어하에 동기적으로 실행된다.

2.1.4 DINO(Distributed Numerically Oriented language)

Colorado 대학의 R. Schnabel등에 의해 제시된 DINO는 산술 계산을 위해 개발되었던 분산 메모리 다중처리기 환경하에서 병렬 프로그램을 작성하기 위한 언어이다[25]. 분산 메모리상에서 효율성에 지장없이 병렬의 산술 알고리즘을 가능한 한 쉽게 프로그래밍하도록 하는 것이 DINO의 목적이다. 이를 위한 DINO의 주요 기능은 알고리즘 및 자료 분할, 처리기간

통신을 비롯하여, 현재의 메시지 전달 시스템이 제공하는 것보다 고수준의 추상화 형태로 프로세스를 관리하는 태스크 발생 등이 있다. DINO가 제공하는 고수준의 구조들은 분산 병렬 계산에 따르는 저수준의 사소한 작업을 컴파일러에게 맡긴다. 특히 메시지 전달, 프로세스 관리 및 동기화같은 작업들은 프로그래머가 코드로 작성할 필요가 없으며, 이와 관련된 효율성은 컴파일러가 고려하게 된다. 이러한 DINO는 표준 C에 여러 고수준 병렬 구조를 추가한 것이다.

DINO에서는 다음과 같이 프로그래머가 하향식으로 분산 병렬 알고리즘을 기술할 수 있도록 한다. 우선 알고리즘의 주요 자료구조와 통신유형에 적합한 가상 병렬 기계를 정의하고, 이 자료구조를 분배하여 가상 처리기에 복사하는 방법을 명시하고, 동시에 실행될 프로시쥬어를 각 가상 처리기에 제공한다. 분산 산술계산 알고리즘의 자료구조는 대부분이 배열 형태이고 나머지는 트리구조이다. 이 알고리즘은 대부분 자료 병렬성을 나타내므로, 계산의 각 단계에서 자료구조를 조각들로 나누고 각 조각들에 비슷하거나 같은 계산을 동시에 수행한다. 함수 병렬성도 지원하기는 하나 주로 자료 병렬성을 지원한다.

2.1.5 Kali

분산 메모리 구조에서는 전역 주소공간을 제공하지 않으므로 프로그램이 실행할 하드웨어에 종속된다. 따라서 프로그래머는 큰 자료구조를 한 처리기가 소유할 조각들로 나누어야 하고, 자료구조의 다른 부분간 상호작용을 명시적인 메시지 전달 구조로 기술해야 한다. Kali는 이러한 문제들을 해결하기 위해 Rice 대학의 Koelbel 등이 설계한 프로그래밍 환경이다[22]. Kali에서는 분산 메모리 구조에서 전역 주소공간을 지원하는 소프트웨어 층(layer)을 제공하여 계산은 공유 메모리 구조에서처럼 전역 주소공간을 사용하는 병렬 루프의 집합 형태가 된다. 그래서 비공유 메모리 구조의 성능은 유지하면서도 공유 메모리 구조에서와 같이 프로그램 작성이 쉬워진다. Kali에서는 고수준의 분산된 방법으로 병렬 알고리즘을 기술한다. 컴파일러는 이 고수준 명세를 분석하여 메시지 전달 방식으로 통신하는 태스크 시스템으로 변형한다.

2.1.6 Paragon

구조에 독립적인 방법으로 묵시적인 병렬성을 표현하는 프로그래밍 패러다임으로 작성된 프로그램은 어떤 하드웨어에도 쉽게 이식될 수 있다. 이러한 성질을 고려하여 Cornell 대학에서 행해진 연구 프로젝트가 Paragon이다[8]. 현재 구현된 Paragon 프로그래밍 환경은 두 개의 기본적인 구성요소로 이루어져 있다. 첫째, 기본적인 병렬계산 함수집합을 구현하는 실행시간 라이브러리로 이 함수들은 하드웨어 구조에 의존적이다. 둘째, 하드웨어 구조에 독립적인 프로그래밍 언어로 하나의 문제가 실행시간 라이브러리에 의해 구현된 프리미티브로 직접 기술될 수 있다.

Paragon에서의 병렬 연산은 배열 형태로 기술된다. 현재 구현된 Paragon 프로그래밍 환경은 C++ 클래스의 모임으로 제공된다. 객체들로 된 이 라이브러리는 Paragon 시맨틱의 수정을 빠르게 하고, 새로운 프로그래밍 구조의 추가를 상대적으로 쉽게 한다. 개별 연산을 위한 배열 표현, 간단한 자료 치환 함수(shift, rotate, transpose), 자료 복사(replication)와 정리(reduction) 함수, 병렬 자료 선택과 선택적 실행 메카니즘 등의 Paragon 프리미티브 함수들은 포트란 90과 같은 자료 병렬언어 대부분에서 볼 수 있는 공통적인 기본 기능들과 유사하다. Paragon은 병렬 자료에 대한 2단계 추상화 형태를 포함하고 있고, 포트란 90에는 없는 자료사상 구조도 지원한다.

2.2 태스크 병렬언어

자료구조, 연산, 자료 종속성 등에서 이질적인 연산에는 자료 병렬언어가 적절치 않다. 이질성은 다음과 같은 이유로 발생한다[28]. 첫째, 응용 프로그램이 모듈 내부에서는 자료 병렬성을 가지면서 모듈간에는 태스크 병렬성을 갖는 상이한 모듈로 구성될 수 있다. 둘째, 고속망(high speed network)으로 연결된 이기종 컴퓨터들을 사용하여 병렬 연산을 하는 것이 일반화되고 있다. 각 컴퓨터들은 그 자신이 병

럴 컴퓨터가 될 수도 있고, 워크스테이션에서 수퍼 컴퓨터에 이르기까지 다양한 종류가 있을 수 있다.

독립된 프로그램을 독립된 태스크로 다루는 태스크 병렬성이 소프트웨어 공학 측면에서 유리하며, 구역성을 강화해 캐쉬, 메모리, 통신 대역폭 등의 자원을 보다 효율적으로 이용할 수 있게 한다. 대표적인 태스크 병렬언어로는 PCN, Strand, Linda, Delirium, 포트란 M, Fx 포트란이 있고, 태스크 병렬성을 이용하기 위한 도구로는 Schedule, Hence, CODE가 있다[2, 7, 11, 24, 27].

2.2.1 포트란 M

포트란 M은 태스크 병렬 계산을 표현하기 위해 Argonne과 Caltech 연구원들이 설계한 언어이다. 포트란 M은 포트란 77에 병행 실행 기능을 추가하여 확장한 것이다[11, 12, 13, 15]. 포트란 M에서는 프로그램 모듈을 정의하는 구조인 프로세스를 제공한다. 이 프로세스는 동시에 실행됨을 나타내고, 프로세스간에 정형화된 일대일 통신채널을 구축하고 이 채널을 통해 메시지를 송수신할 수 있다. 프로세스는 포트란 77의 서브루틴 구조로 모형화되며, 병렬 프로그램의 기본단위가 된다. 프로세스와 채널은 동적으로 생성되고 삭제된다. 그러나 여러 프로세스가 포트 변수상에서 동시에 송신하지 못하도록 하고, 수신자는 자료가 사용 가능할 때까지 블럭되도록 하며, 변수상에서 copy-in/copy-out 시맨틱을 프로세스의 매개변수로 전달하게 하여 결정성(determinism)을 보장한다.

포트란 M에서 프로세스가 처리기에 어떻게 사상되는가와 계산 자원이 프로그램의 서로 다른 부분에 어떻게 할당되는가를 프로그래머가 제어할 수 있다. 이 구조는 포트란 77 배열구조로 모형화되어, 프로그래머는 가상 처리기 배열을 정의하고, 이 배열내에 프로세스를 위치시키고 부분배열상에서 부분계산을 하도록 할 수 있다.

2.2.2 Fx 포트란

Fx 포트란의 병렬 프로그램은 자료 병렬 수행이 가능한 태스크 서브루틴 집합으로 구성된다[27, 28]. 태스크는 태스크 서브루틴을 호출해 실행시키는 부분이고 태스크 서브루틴은 분명한 side-effect를 갖는 자료 병렬 서브루틴이다. 태스크 서브루틴을 호출하는 프로그램의 자료 관련성이 분명하기 때문에, 컴파일러는 상이한 처리기상에 태스크를 사상할 수 있으며 자료 일관성을 유지하기 위해 통신도 생성할 수 있다. 이때의 기본 패러다임은 순차실행의 결과와 같은 결과가 얻어지도록 하는 것이다. 태스크들을 처리기 노드로 사상하는 것이 정확성에는 영향을 주지 않으나, 좋은 성능을 얻는데는 중요한 요인이 된다. 프로그래머는 지시어를 사용하여 이 사상을 제어할 수 있다. 효율적인 사상을 자동적으로 선택하거나 프로그래머를 돕는 도구도 개발되고 있다.

2.2.3 Linda

Yale 대학의 D. Gelernter 등에 의해 제시된 Linda는 병렬 프로그래밍의 튜플 영역(tuple space) 모델을 구체화하는 간단한 연산들로 구성된다[7]. 포트란이나 C와 같은 기본적 언어에 튜플 영역 연산을 추가하여 병렬 프로그래밍 언어가 된다.

Linda 모델에서, 두 처리기가 통신하려면 메시지나 공유변수를 교환하는 것이 아니라 자료를 만드는 프로세스가 새로운 자료 객체(튜플이라 함)를 생성하여 튜플 영역에 위치시킨다. 수신 프로세스는 바로 이 튜플에 접근할 수 있다. 프로세스가 동시에 실행되는 다른 프로세스를 생성하려면 "live 튜플"을 생성하여 튜플 영역에 그 프로세스를 위치시킨다. Live 튜플은 자신을 생성한 프로세스와 독립적으로 자신의 연산을 수행한 후 자료 객체 튜플로 돌아간다. 이 기법은 다음과 같은 두가지 의미를 갖는다. 첫째, 통신과 프로세스 생성이 동일한 연산이다. 즉 프로세스를 생성하려면 자료 객체로 돌아가는 live 튜플을 생성하고, 통신하려면 자료 객체 튜플을 생성한다. 둘째, 교환되는 자료는 일시적인 메시지 형식이 아니라 지속적인 객체 형식이다. 수신자는 자료를 생성하는 프로세스가 만든 튜플을 지울 수도 있고, 다른 프로세스들도 읽어야 하는 경우 그대로 남겨둘 수도 있다. 튜플 영역은 계산을 행하는 블랙박스(black-

box) 프로그램의 외부에 존재하여, 임의의 언어로 작성된 블랙박스간 정보 교환에도 사용될 수 있다.

2.2.4 CODE

Texas 대학의 P. Newton에 의해 제시된 CODE 시리즈는 그래픽 병렬 프로그래밍 시스템이다[24]. 프로그래머는 병렬성을 선언적인 고수준의 추상화 형태로 매크로급 단위계산을 합하여 일반화된 종속성 그래프 형태의 병렬 구조로 표현한다. 이 그래프는 명시적 코드로 자동 번역되는데, 노드와 아크의 시맨틱이 대부분 선언적으로 명시되는 다중 그래프가 된다.

CODE 2.0이전의 버전(CODE 1.0과 CODE 1.2)은 쉬운 이용, 재사용성, 효율적 병렬코드 생성 등의 측면에서 상당한 성공을 거두었지만, 단일화된 병렬 계산 모델의 정적인 버전에 대해서만 구현되었다. CODE 2.0은 이전 CODE 버전보다 다양한 병렬 컴퓨터에서 효율적으로 계산모델 표현력이 강화된 것이다.

2.2.5 HeNCE

Tennessee 대학의 A. Beguelin등에 의해 제시된 HeNCE(Heterogeneous Network Computing Environment)는 과학자들이 컴퓨터 망위에서 실행되는 병렬 프로그램 개발하는 것을 돕기 위해 설계되었다[2]. HeNCE는 PVM이라는 소프트웨어 패키지위에서 프로그램 생성, 컴파일, 실행, 분석을 위해 통합된 그래픽 도구로 구성된다. PVM은 이기종 컴퓨터 망에서 프로세스와 통신을 관리하는 소프트웨어 패키지이다. HeNCE는 응용 프로그램이 그래프로 표현되는 병렬 프로그래밍 패러다임을 기본으로 한다. 그래프의 노드(node)는 서브루틴을, 아크(arc)는 자료 종속성을 나타낸다.

프로그래머는 DAG(Directed Acyclic Graph)나 DAG의 변형인 그래프를 명시한다. 그래프의 노드는 프로시쥬어, 에지는 종속성을 나타낸다. 프로그래머가 입력한 프로그램 그래프는 그래픽 인터페이스를 사용한다. 그래프의 노드로 표현된 프로시쥬어는 C나 포트란으로 작성된다. 대부분의 경우 이 프로시쥬어는 기존 코드로부터 얻어질 수 있다. 이러한 소프트웨어 재사용성이 HeNCE의 큰 장점이다.

HeNCE 도구는 사용자가 정의한 가상 병렬 컴퓨터의 다양한 구조상에서 프로시쥬어 편집과 컴파일 기능을 제공한다. HeNCE 프로그램 그래프가 한번 작성되어 컴파일되면 사용자가 정의한 컴퓨터(가상 병렬 컴퓨터)상에서 실행될 수 있다. 프로그래머는 HeNCE를 사용하여 프로그램이 실행되는 다양한 컴퓨터를 명시한다. 가상 병렬 컴퓨터의 한 노드로는 단일 처리기 워크스테이션에서 64K개 처리기의 CM-2까지 가능하다.

3. 자료 병렬성

분산 메모리 기계에는 대용량의 배열 자료를 처리하기 위해 각 처리기마다 존재하는 메모리에 자료들을 분할하여 적절히 분산시키는 '자료 병렬성' 개념이 중요하다. 여기서 각 처리기에서 나뉘어져 처리되도록 분산되어 있는 배열 자료를 분산 배열(distributed array)이라고 부른다.

이와 관련하여 지금까지는 분산 배열을 어떤 방법을 통해 처리기들에게 사상시킬 것인가 하는 문제가 주로 연구되어왔다. 일반적으로 대부분의 응용 프로그램들은 'BLOCK'이나 'CYCLIC' 등으로 불리우는 자료분배 방법을 이용하여 분산 배열들을 처리기로 사상시켜 성능 향상을 보여왔다. 가장 잘 알려진 이러한 자료분배 방법은 자료 병렬성을 고려하여 발표된 HPF나 포트란 D, 비엔나 포트란 등의 언어에서 프로그래머가 배열 원소들을 처리기에 사상시키는 방식으로 제어하도록 한다[10, 16, 23].

프로그래머는 언어 차원에서 이러한 기능을 제공하는 HPF 등의 자료 병렬언어를 사용하여, 매우 큰 자료들을 병렬 처리하되 단일 쓰레드 제어 방식으로 프로그래밍을 진행할 수 있다. 자료 병렬성을 위한 자료분할 및 사상에 필요한 병렬수행, 통신, 동기화 등과 같은 복잡한 문제가 컴파일러에 의해 자동적으로 처리된다.

이 장에서는 HPF, 포트란 D, 비엔나 포트란 등과 같은 자료 병렬언어에서 자료분배(data distribution)를 어떻게 지원하는가와 이들에 대한 병렬 처리를 위해 제공되는 병렬수행 구

조에 대해서 살펴본다.

3.1 자료 분배

자료를 각 처리기에 분배하는 것은 자료 병렬성을 이용하기 위한 기본적인 방법으로서, 고성능 병렬언어는 지시어(directive)를 제공하여 프로그래머가 자료의 분배를 제어할 수 있도록 한다.

자료분배 방법 중 가장 많이 쓰이는 것은 BLOCK과 CYCLIC이다. BLOCK 자료분배 방식은 프로그래머가 선언한 배열을 블럭단위로 처리기에 할당한다. 블럭은 전체 배열 중 연속적인 원소로 구성된 부분 배열의 형식을 갖고 있다. 예를 들어, 다음과 같이 X, Y와 같은 배열과 BLOCK 분배 방식을 선언했을 경우, X와 Y는 각각 열(column)의 집합과 사각 블럭(rectangular block)으로 나뉜다.

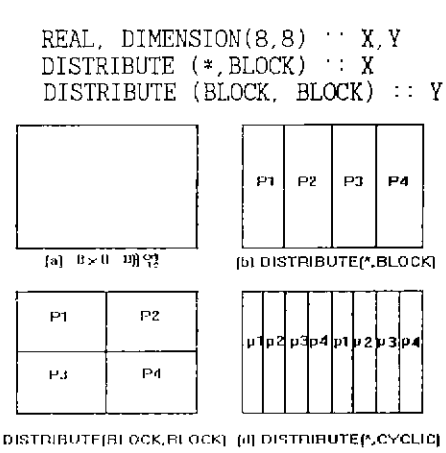


그림 1 8×8 배열에서 자료분배 방식의 예

여기서 DISTRIBUTE는 일종의 지시어로서 X라는 배열에 대해 어떤 방식으로 자료분배가 될 것인가를 선언하는 것이다. 위의 예에서 주어진 자료분배 방식을 그림으로 표현한 것이 그림 1의 (b)와 (c)이다. 여기서는 2차원 배열을 사각형으로 표현하여 배열의 원소에 처리기가 할당된 형태로 나타낸다.

CYCLIC 자료분배 방식은 배열 원소를 처리기에 할당하되, 선언되어 있는 논리적 처리기의 크기 N을 참고하여 배열의 N번째 열(column)마다 동일한 처리기가 할당되도록 한다. 예를 들어 다음은 X라는 배열을 선언하고 논리적 처리기의 크기가 4일 때, X 배열의 4번째 열마다 항상 동일 처리기에 할당되는 것이다.

```
REAL, DIMENSION(8, 8) : : X
PROCESSORS PROC(4)
DISTRIBUTE(CYCLIC, *) ONTO PROC1 : : X
```

CYCLIC 형태의 자료분배 방식은 그림 1의 (d)와 같다.

BLOCK과 CYCLIC 방식을 혼합하여 적용할 수 있는데, 이 경우 응용 프로그램 성격에 따라 성능 차이를 보일 수 있으므로 프로그래머는 프로그램에서 사용하는 자료들에 대한 참조형식(pattern)을 예측하여 적절한 분배방식을 선택해야 한다.

자료 병렬언어중 대표적인 HPF의 경우, 자료분배 방법은 DISTRIBUTE 지시어를 이용하여 자료 사상 관계를 명시하는 것이다. HPF는 앞에서 설명하였던 BLOCK과 CYCLIC 두 가지 형태를 지원한다. HPF에서는 지시어를 삽입할 때 문장앞에 !HPF$, CHPF$, *HPF$ 등을 명시하여 표현한다. 이와 같이 해당 자료에 BLOCK과 CYCLIC 등의 분배 방식을 직접 명시할 수도 있고, 이미 선언되어 있는 배열에 첨자(subscript) 형태로 정의할 수도 있다. 예를 들어 다음은 SALAMI라고 선언된 배열에 BLOCK방식을 적용한다는 것을 표현하고 있다.

```
REAL SALAMI(10000)
!HPF$ DISTRIBUTE SALAMI(BLOCK)
```

HPF는 DISTRIBUTE 이외에 REDISTRIBUTE라는 지시어를 제공한다. REDISTRIBUTE는 이미 분배 방식이 결정되어 있는 자료 객체의 사상 관계를 재정의하는 지시어로서, 속성(attribute)을 명시하기 위한 것이라기보다 재설정하는 지시어이므로 실제 '실행' 성격을 갖는다. REDISTRIBUTE를 이용하여

분배 방식을 변경하고자 할 경우, 해당되는 자료 객체는 미리 DYNAMIC이라는 지시어를 통해 동적 자료임이 표현되어 있어야 한다.

포트란 D는 HPF가 발표되기 이전에 기본 모델로 설정되었던 자료 병렬언어이므로, HPF와 거의 동일한 자료분배 방식을 제공하고 있다. 역시 DISTRIBUTE 지시어를 사용하여 자료분배 방식을 결정하며, 다양한 형식의 문장으로 자료분배가 이루어 진다.

HPF에서 REDISTRIBUTE를 이용하기 위해 배열 선언시 DYNAMIC 지시어로 동적/정적 배열을 구분하듯이, 비엔나 포트란에서도 분산 배열은 정적 배열과 동적 배열로 분류된다. HPF에서 처럼 정적 배열로 분류된 경우에는 배열 선언의 범위(scope)안에서 자료분배 방식이 항상 고정되어 있으며, 동적 배열은 프로그램 수행중에 자료분배 방식이 수정될 수 있는 것을 말한다. 이와 같이 배열을 선언할 때 정적 혹은 동적으로 구분하는 것은 자료분배 방식이 결정됨에 따라 컴파일러가 효율적인 목적 코드를 생성하는데 영향을 미칠 수 있기 때문이다.

자료 병렬언어에는 자료분배 방식을 직접적으로 명시하는 DISTRIBUTE 지시어 이외에 ALIGN 등과 같이 좀더 일반적이고 편리하게 이용될 수 있는 지시어를 제공한다. ALIGN은 자료분배 방식을 각 자료 객체들에게 명시할 필요없이 이미 자료분배 속성을 갖고 있는 객체들을 이용하여, 임의의 자료 객체들을 처리기에 사상할 수 있도록 한다. 서로 align되어 있는 자료 객체들은 동일한 처리기에 사상되므로 align되어 있지 않은 객체들에 비해 효율적인 연산이 가능하다. 이러한 ALIGN 지시어의 지원은 배열 원소들에 대한 자료분배 방식을 한번에 명시할 수 있기 때문에, 자료 병렬성을 고려하는 대부분의 자료 병렬언어에서 이용되고 있다.

HPF, 포트란 D, 비엔나 포트란 등은 모두 aligning을 지원하는데, 이중 비엔나 포트란은 조금 다른 형태의 지시어를 통하여 aligning을 가능하게 한다. HPF, 포트란 D에서 다음과 같이 aligning을 했을 경우 그림 2로 표현할 수 있다.

HPF나 포트란 D와 달리 비엔나 포트란에서

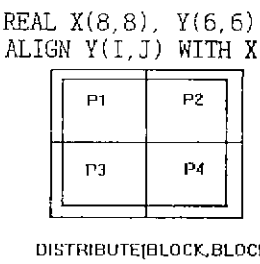


그림 2 8×8 배열과 6×6 배열과의 aligning

는 aligning을 하기 위해 DISTRIBUTION, PERMUTE, TRANSPOSE 등의 정보 제공 함수를 이용한다. DISTRIBUTION(A) 형태로 함수를 호출하면 배열 A의 자료분배 방식에 관한 정보를 알 수 있으며, 이것은 '= A' 등의 문법으로 대체될 수 있다. PERMUTE, TRANSPOSE는 각각 배열 원소들을 순열 또는 대각화등의 방법으로 재배열하여, 이미 분배 방식이 결정된 배열 자료를 aligning할 수 있게 한다. 예를 들어 다음과 같이 선언된 경우를 살펴보자.

```
REAL A(100, 100) DIST(BLOCK, CYCLIC)
REAL B(100, 100) DIST(DISTRIBUTION(A))
REAL C(100, 100) DIST(= A)
REAL D(100) DIST(DISTRIBUTION(A, 1))
REAL E(100) DIST(= (A, 2))
REAL F(100, 100) DIST(TRANSPOSE(A))
```

여기서 배열 B와 C는 A와 동일한 방식으로 align되도록 선언되었으며, 배열 D의 자료분배 방법은 배열 A의 첫번째 dimension 방식인 BLOCK방식을 유지한다. 배열 E는 A의 두번째 dimension 방식인 CYCLIC 방식을 따르며, 배열 F는 TRANSPOSE라는 함수를 이용하므로 배열 A의 대각화 결과인 (CYCLIC, BLOCK) 방식을 적용하게 되는 것이다.

3.2 병렬 구조

HPF와 같은 종류의 자료 병렬언어는 '병렬성'과 '통신'의 두가지 특징을 갖는다. 앞에서 기술하였듯이 자료분배 등을 통한 처리기의 사상은 프로그램내에 존재하는 통신 상태(communication status)를 결정하기 위한 것이다.

이 절에서는 자료 병렬언어에서 병렬성을 어떻게 표현하고 있는가를 기술한다. 특히 자료 병렬성을 지원하는 프로그래밍 언어의 경우 각종 "병렬 구조"를 통해 자료 병렬성을 활용하기 위한 기본적인 연산을 제공한다.

HPF에서는 다음과 같은 방법들을 통해 병렬성을 표현한다.

- 포트란 90의 배열 표현식과 할당문 이용
- FORALL 문 이용
- DO와 FORALL 등에 명시될 수 있는 INDEPENDENT 지시어 이용
- HPF 라이브러리에서 제공하는 배열 관련 라이브러리 함수 이용

본 고에서는 병렬성을 표현하는 방법들중 FORALL 문과 지시어들을 중심으로 서술한다. FORALL 문은 배열 원소들을 블럭 단위로 할당할 수 있도록 하며, 루프처럼 어떤 특별한 순서에 따라 반복(iteration)을 진행하지는 않는다.

FORALL 문은 포트란 90의 배열 치환문(array assigment statement)과 비슷하게, 우변(right-hand side)에 있는 모든 배열들을 미리 계산하여 좌변(left-hand side)에 치환하도록 한다. 그림 3의 FORALL이 포함된 프로그램에 대해 그림 4와 같이 수행순서를 나타내어 DO 문과 비교할 수 있다.

INDEPENDENT 지시어는 컴파일러가 찾지 못하는 병렬성을 프로그레머가 직접 명시하여 DO문을 병렬적으로 수행할 수 있도록 한다. INDEPENDENT를 사용한 예는 다음과 같다.

```
FORALL(I=1 3)
 a(I) = b(I)
 c(I) = d(I)
END FORALL
```

(a)FORALL문

```
DO I=1,3
    a(I) = b(I)
    c(I) = d(I)
END DO
```

(b)DO문

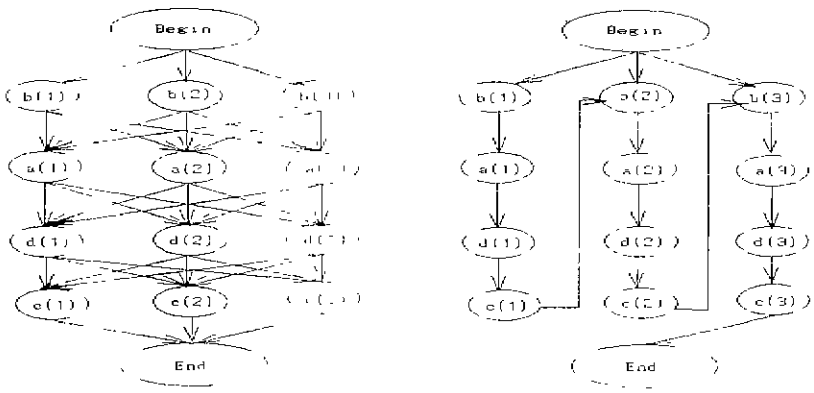
그림 3 FORALL문과 DO문

```
!HPF$ INDEPENDENT, NEW (J, N1)
   DO I = 1, NBLACK
     N1 = IBLOCK_PT(I)
     DO J = INITIAL_RED(N1), LAST_RED(N1)
       X(N1) = X(N1) + A(J) * X(IRED_PT(J))
     END DO
   END DO
```

포트란 D의 경우 FORALL 문의 사용은 HPF와 동일하지만, INDEPENDENT 지시어 등은 제공하지 않는다. 그리고 포트란 D는 복잡한 배열 연산을 하나의 연산으로 표현하는 reduction 함수를 다양하게 지원하고 있다는 특징이 있다.

4. 태스크 병렬성

태스크 병렬 프로그래밍 패러다임에서 프로



(a) FORALL문 (b) DO문

그림 4 FORALL 문과 DO 문과의 순서 관계

그램은 명시적인 통신, 동기화를 통하여 상호 작용하는 일련의 병렬 태스크들로 구성된다. 이러한 태스크 병렬성을 지원하는 경우 다음과 같은 장단점이 있다[14, 17].

태스크 병렬성은 융통성을 제공한다. 프로세스라 불리우는 각 태스크가 명시적으로 동등하기 때문에, 태스크 병렬성은 구조적/비구조적 병렬성을 향상시키는데 사용될 수 있다. 그리고 태스크들간의 상호 작용이 명시적이기 때문에 프로그래머는 컴파일러에 의해 자동적으로 추출되지 않는 형태의 병렬성을 표현하는 프로그램을 작성할 수 있다. 또한 실제 필요하면서도 효율적인 통신과 동기화 기능을 프로그래머가 제공하게 하여, 컴파일러의 최적화 부담을 줄일 수 있고 응용 프로그램을 제어할 수 있다. 일반적으로 태스크 병렬성은 자료 병렬성보다는 응용 컴파일러 기술에 덜 의존적이지만, 많은 경우 목표 기종에서 수행할 적절한 프로그램으로의 번역이 필요하다.

태스크 병렬 프로그래밍 모델은 명시적인 병렬 태스크 생성과 그들간의 통신 및 동기화 관리를 위해 프로그래머의 부가적인 노력이 필요하고 그로인해 프로그래밍이 까다롭게 된다. 또한 통신과 동기화가 명시적이기 때문에 프로그램을 병렬화하는 방법의 변화시 그 프로그램 구문에 확장적인 수정 요구가 있을 수 있다.

이 장에서는 이러한 태스크 병렬성을 지원하는 대표적인 언어인 포트란 M과 Fx 포트란을 중심으로 태스크 병렬성 표현 방안과 유지 형태를 기술한다.

4.1 포트란 M에서의 태스크 병렬성

포트란 M은 기존의 포트란을 확장한 것으로, 정형화된 채널상에서 메시지를 주고 받는 등의 상호작용에 의해 프로세스를 생성하는 메시지 전달 병렬 프로그래밍 모델을 제공한다. 이러한 확장 과정에서 주요 기능은 병행성과 이를 지원하기 위한 프로세스간의 통신이다. 또 이러한 프로세스를 어떻게 처리기에 할당할 것인가를 보여주는 자원 관리 기능이 있다.

4.1.1 병행성과 통신

포트란 M에서는 병행성과 통신을 지원하기 위해 다음과 같은 구조를 제공한다.

- 프로그램 모듈인 프로세스 정의를 위한 구조
- 프로세스의 병렬수행을 명세하는 구조
- 프로세스들 사이의 일대일 통신 채널을 규정하는 구조
- 채널상에서 메시지 교환을 위한 구조

메시지 통신을 위한 SEND와 RECEIVE 연산은 포트란의 화일 I/O문을 사용하여 모형화하고, 장치번호를 사용하기 보다 포트(port) 변수에서 연산한다. 포트란 M 프로그래밍 모델은 동적으로 수행된다. 즉, 프로세스와 채널이 동적으로 생성/제거될 수 있고, 메시지내에 채널을 참조하는 기능을 포함할 수 있다. 연산이 결정적(deterministic)으로 수행됨에도 불구하고, 많은 병렬 프로그래밍 시스템에서 발생되는 경쟁 상태(race condition)가 발생하지 않는다.

그림 5는 포트란 M의 병행성과 통신의 사용예이다.

```
PROGRAM AERODYNAMICS
INPORT (INTEGER, REAL X(10,10), REAL Y(10,20)) PI
OUTPORT (INTEGER, REAL X(10,10), REAL Y(10,20)) PO
  . . .
CHANNEL(IN=PI, OUT=PO)
CHANNEL(IN=QI, OUT=QO)
  . . .
PROCESSES
    PROCESSCALL CONTROLS(PI,QO)
    PROCESSCALL DYNAMICS(QI,PO)
ENDPROCESSES
END

PROCESS CONTROLS(IN,OUT)
INPORT (INTEGER, REAL X(10,10), REAL Y(10,20)) IN
OUTPORT (INTEGER, INTEGER, REAL X(10,10,3)) OUT
  . . .
SEND(OUT) I, J, A
RECEIVE(IN) NSTEP, U, V
  . . .
END
```

그림 5 포트란 M 프로그램 예

그림 5에서, 처음은 지역포트 변수 정의부분으로 입력 포트인 PI는 두 개의 실수형 배열 값을 수신하고, 출력 포트인 PO는 두 개의 실수형 배열값을 송신하는데 사용됨을 나타낸다. 이러한 방식으로 정의되는 메시지 형식은 포트란 M 컴파일러가 효율적인 통신 코드를 생성하고, 이기종 환경하에서 기종에 독립적인 형식으로

변환할 수 있도록 허용한다. 이러한 포트 변수로 2개의 채널을 생성하는 CHANNEL 문과 CONTROLS와 DYNAMICS이라는 두 프로세스를 생성하는 프로세스 블럭(PROCESSES 문과 ENDPROCESSES 문으로 지정된 블럭)이 차례로 나타나 있다. 그 다음은 CONTROLS 프로세스의 내용을 나타내는데, 자료를 포트상에서 매개변수처럼 주고 받기 위해 SEND 문과 RECEIVE 문을 사용하고 있다.

포트란 M의 기본 기능이 태스크 병렬성이긴 하지만 자료 병렬 연산을 일부 지원하고 있어 포트란 D나 HPF의 영향을 받은 자료 분산 문장의 프로그램도 사용가능하다. 그러나 아직 프로토 타입에서는 구현되어 있지 않다.

4.1.2 자원의 관리

포트란 M의 자원 관리 구조는 프로세스와 자료를 어떻게 연결시킬 것인가와 연산용 자원들이 한 프로그램의 서로 다른 부분에 어떻게 할당될 것인가를 프로그래머가 결정할 수 있도록 한다. 포트란 M의 프로세스 배치 구조는 가상 처리기와 같은 가상 컴퓨터 개념을 기반으로 하는데, 가상 컴퓨터는 프로그램을 수행하는 실제 처리기와 일치할 수도 그렇지 않을 수도 있다. 가상 컴퓨터는 N차원의 배열 형태이고, 사상방법은 배열처리 방식과 유사하다.

PROCESSORS 선언은 한 처리기 배열의 형태와 차원을 결정하고 LOCATION 표현은 프로세스들을 이 배열의 특정 원소에 사상하며, SUBMACHINE 표현은 한 프로세스가 그 배열의 부분 집합에서 수행할 수 있도록 결정한다. 예를 들어 다음은 서로 다른 가상 처리기 상에 CONTROLS와 DYNAMICS 프로세스를 배치하는 프로그램이다.

```
PROCESSORS(2)
  . . .
PROCESSES
    PROCESSCALL CONTROLS(...) LOCATION(1)
    PROCESSCALL DYNAMICS(...) LOCATION(2)
ENDPROCESSES
```

반면, 다음은 10개의 가상 처리기들을 갖는 submachine내에 각 프로세스를 배치하는 코드이다. 이 경우 프로세스가 병렬 프로그램이라면 보다 효과적이다.

```
PROCESSORS(2)
  . . .
PROCESSES
    PROCESSCALL CONTROLS(...) SUBMACHINE(1:10)
    PROCESSCALL DYNAMICS(...) SUBMACHINE(11:20)
ENDPROCESSES
```

4.2 Fx 포트란에서의 태스크 병렬성

Fx 포트란은 태스크간의 모든 통신을 제공하는데, 태스크 병렬화 관리 프로그램이 프로그램을 공유 자료공간에 저장하고, 컴파일러가 병렬 시스템의 주소공간을 구분하여 자료 객체를 사상한다.

그런데 Fx 포트란에서 태스크 병렬화는 다음과 같은 제약조건이 존재한다.

- 태스크 구조들간의 통신은 그 구조에 내재하는 자료 병렬성만을 사용하기 때문에, 프로시 쥬어 영역에서 프로시쥬어 매개변수를 통하여서만 허용된다.
- 노드에 대한 태스크 사상은 컴파일 시간에만 이루어진다.

Fx 포트란에서는 태스크 병렬성을 표현하기 위하여 새로운 언어 개념을 추가하기 보다는 기존의 지시어를 사용하고자 하였기 때문에, 태스크 병렬 프로시쥬어 영역인 병렬섹션(parallel section)내에서 자료병렬 구조의 프로시쥬어인 태스크 서브루틴을 호출하는 방식을 취한다. Fx 포트란에서 태스크 병렬구조 및 입출력과 관련된 내용을 살펴보면 다음과 같다.

4.2.1 병렬섹션

태스크 병렬성은 병렬섹션이라 불리우는 코드영역 개념을 사용하여 표현하고, 병렬섹션내의 내용은 태스크 서브루틴과 루프만을 포함할 수 있도록 제한한다. 이렇게 제한을 두는 이유는 컴파일 시간에 공유 자료 및 공유 자원을 효율적으로 관리하기 위함이다. 병렬섹션에는 태스크 서브루틴의 입출력과 자원관리를 규정하기 위한 다른 지시어들과 태스크 서브루틴을 처리기나 다른 자원에 사상하는 내용도 포함된

다. 병렬섹션의 구조는 다음과 같다.

```
C$    begin parallel
                :
      (태스크 서브루틴 호출이나 루프, 다른
       지시어들로 구성된 몸체)
                :
C$    endparallel
```

4.2.2 입출력 매개변수 및 사상

Fx 포트란에서는 모든 태스크 서브루틴의 side-effect를 컴파일 시간에 알 수 있도록 하기 위해, 서브루틴 호출로 인한 side-effect를 정의하는 INPUT, OUTPUT 지시어를 사용한다. 그리고 호출 프로그램내의 모든 변수는 입출력 매개변수 목록에 포함되어야 한다. 입출력 매개변수 목록의 변수는 스칼라 값이거나 배열, 배열섹션 값일 수 있다. 여기서 배열섹션은 포트란 90의 배열섹션과 같다.

Fx 포트란에서는 일련의 처리기나 기계로의 태스크 서브루틴 배치를 프로그래머가 제어한다. 이때 사용되는 지시어나 시맨틱은 기계구조에 의존한다. 그래서 동종(homogeneous)의 처리기 배열 구조하에서는 배열의 크기나 위치가 태스크 서브루틴을 사상하기 위한 충분한 정보가 되지만, 이기종(heterogeneous) 환경하에서는 추가 정보가 필요하다. 서로 다른 태스크 서브루틴은 따로 또는 함께 사상될 수 있다. 또한 단일 태스크 서브루틴이 여러 위치에 존재하면, 태스크 서브루틴은 서로 다른 위치에서 round-robin 방식으로 수행될 수 있다.

Fx 포트란에서는 사상을 위한 지시어가 원시 프로그램상에 나타날 수 있다. 다음은 동종의 2차원 처리기 배열의 iWarp에서 사상을 위해 지시어를 사용하는 예제이다[28].

```
C$    begin parallel
                :
      call foo(a, b, i)
C$    processor (4, 4)
CS    orign (0, 0)
                :
C$    endparallel
```

여기서 processor 지시어는 태스크 서브루틴을 수행하는데 필요한 처리기가 사각형 모양의 블럭구조임을 정의하고, orign 지시어는 2차원 처리기 배열 구조에서 태스크 서브루틴의 위치를 기술한다.

```
C$    begin parallel
       do i = 1,10
         call src(A,B)
C$       output(A,B)
C$       processor(2,4)
C$       origin(0,0)
         call p1(A)
C$       input(A),output(A)
C$       processor(2,4)
C$       origin(0,0)
         call p2(B)
C$       input(B),output(B)
C$       processor(2,2)
C$       origin(2,2),(2,2)
         call sink(A,B)
C$       input(A,B)
C$       processor(2,2)
C$       origin(2,0),(2,2)
       enddo
C$    endparallel
```

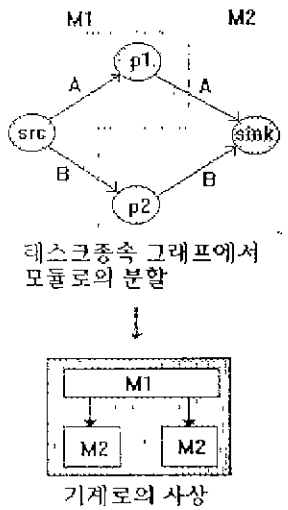


그림 6 태스크 병렬성의 컴파일

4.2.3 Fx 프로그램 예

이상에서 기술한 지시어가 태스크 병렬 프로그램을 작성하는데 어떻게 사용되고, 사상시 어떻게 적용될 수 있는가를 그림 6에서 볼 수 있다[27]. 이 프로그램은 수행시 p1과 p2 루틴에 자료를 제공하는 src 루틴과 그 출력결과를 sink 루틴에 보내는 코드를 포함하고 있다.

태스크 서브루틴 src는 출력 매개변수로 A와 B를, 태스크 서브루틴 p1과 p2는 입출력 매개변수로 각각 A와 B를, 태스크 서브루틴 sink는 입력 매개변수로 A와 B를 가지고 있음을 나타낸다. Fx 컴파일러에서는 서브루틴의 입력과 출력을 연결하는 순차 수행 방식을 사용하므로, 그림 6과 같은 태스크 종속 그래프(task dependence graph)를 구성한다. 태스크 서브루틴인 src와 p1은 동일한 orign 지시어를 가지고 있어, 같은 모듈(M1)에 사상되고, 태스크 서브루틴 p2와 sink는 모듈 M2에 사상된다. M2내 태스크 서브루틴은 orign 지시어에 대해 두 개의 인자를 갖기 때문에, 그 모듈은 중복된다.

5. 태스크 병렬성과 자료 병렬성의 통합

앞에서 기술하였듯이 대부분의 고성능 병렬 언어 분야에서 태스크 병렬성과 자료 병렬성을 지원하는 언어가 각각 독립적으로 연구되어왔다. 그러나 최근에 이르러 태스크 병렬성과 자료 병렬성 기능의 통합이 시도되기 시작하였고, 많은 연구진들이 두 기능을 통합한 언어 개발에 방향을 맞추고 있다[11, 14, 17, 27, 28]. 따라서 이 장에서는 먼저 포트란 M의 자원관리 기능과 HPF의 자료분산 기능을 기반으로 한 통합방식과, Fx 포트란에서의 자료 병렬성과 태스크 병렬성 통합방식에 대해 기술한다. 그리고 이때 고려할 사항들을 기술한다.

5.1 포트란 M과 HPF간의 가능한 통합

M. Chandy와 I. Foster는 포트란 M을 제시하면서 HPF와 같은 자료 병렬 프로그램과 어떻게 통합하여 두 종류의 병렬성을 표현할 수 있는가 하는 방안도 연구하였다[11,13]. 그들은 포트란 M과 같은 태스크 병렬 프로그램과 HPF와 같은 자료 병렬 프로그램을 통합하는 모델을 다음의 3가지 형태로 고려하였다.

5.1.1 포트란 M에서 HPF를 호출

포트란 M과 같은 태스크 병렬 프로그램에서 HPF와 같은 자료 병렬 프로그램을 호출하는 방식에서는 태스크 병렬이 자료 병렬 프로그램의 서로 다른 병행 수행을 조정하는 형태가 된다. 여기서 HPF 프로시쥬어 호출시 SUBMACHINE 지시어를 사용하여 HPF 프로시쥬어가 수행될 가상 컴퓨터를 설정할 수 있다. 이 경우 SUBMACHINE을 사용하지 않으면 HPF 프로시쥬어는 호출할 때와 동일한 가상 컴퓨터상에서 수행된다. 그래서 포트란 M 프로그램에서 호출되는 HPF 프로시쥬어를 위한 자료와 연산은 자료병렬 프로그램에서는 모든 처리기들에게 분산되는데 비해 오직 이 가상 컴퓨터상에만 분산된다.

예를 들어, 다음 프로그램은 크기가 10인 가상 컴퓨터상에서 HPF 프로그램인 CONTROLS와 DYNAMICS를 호출하는 프로그램이다. 여기서 배열 변수 X와 Y는 매개변수로 전달된다.

```
PROGRAM PROG1
PROCESSORS(20)
REAL X(1000, 1000), Y(1000, 1000)
PROCESSES
    HPFCALL CONTROLS(X) SUBMACHINE(1:10)
    HPFCALL DYNAMICS(Y) SUBMACHINE(11:20)
ENDPROCESSES
```

반면, 다음은 동일한 10개의 처리기에 두 HPF 연산을 배치하는 프로그램이다.

```
PROGRAM PROG2
PROCESSORS(10)
REAL X(1000, 1000), Y(1000, 1000)
PROCESSES
    HPFCALL CONTROLS(X)
    HPFCALL DYNAMICS(Y)
ENDPROCESSES
```

그리고 HPF의 매개변수로 전달되는 태스크

병렬 프로그램의 배열 변수들은 HPF 프로그램에서예상되는 방식으로 분산되지 않을 수 있다. 따라서 포트란 M 프로그램의 자료 분산을 HPF 프로그램에서 필요한 형태로 변환하기 위한 재분산(redistribution) 작업이 필요할 수 있다. 위의 예에서 배열 변수 X와 Y는 분산되지 않았으므로, CONTROLS와 DYNAMICS에서 필요한 형태로 다시 분산하는 재분산 작업이 필요하다.

5.1.2 HPF에서 포트란 M 호출

HPF같은 자료 병렬 프로그램은 자연스럽지 않은 자료 병렬 연산을 포함할 수 있다. 이 경우, 순차적으로 수행되어 순차 병목현상(sequential bottleneck)을 유발할 수 있다. 이러한 병목현상을 피하기 위해, 적절한 태스크 병렬 알고리즘으로 구현된 포트란 M 프로그램을 호출하도록 할 수 있다. 이때 포트란 M 프로시쥬어의 호출을 구분하여 표시하기 위해 FM-CALL 같은 특별한 표기를 사용한다. 포트란 M 프로시쥬어가 HPF 프로그램을 호출하는 방식에서처럼 동일한 처리기 상에서 수행될 수 있고, 매개변수로 전달되는 배열에 대한 재분산 연산도 필요하다.

5.1.3 HPF 프로그램간의 통신 방식

태스크 병렬 연산에 의해 초기화되는 두 개의 자료병렬 연산이 병행 수행될 경우, 이들간에 자료교환이 필요할 수 있다. 특히 태스크 병렬 연산이 채널(channel)을 만들고, HPF같은 자료 병렬 프로그램에 적절한 포트를 전달하는 경우 이러한 방식이 사용될 수 있다. 그러나 이들 포트상에서 어떤 자료 병렬 프로그램이 연산되는가에 대한 정의가 필요하다.

이를 위해 다음과 같은 여러 방식을 고려해 볼 수 있다. 첫째, HPF 상에서 모든 SEND와 RECEIVE 연산을 처리하는 하나의 처리기를 정할 수 있다. 이 방안은 개념적으로는 가능하나, 프로세스간의 모든 통신을 순차 처리할 가능성이 있다.

둘째, 병렬 IMPORT와 OUTPORT 자료형과 자료병렬 SEND와 RECEIVE 문을 제공할 수 있다. 이 방안은 많은 프로세스들이 다른 프로세스 그룹에 자료를 전송하는 과정에서 상호 협력할 수 있도록 하고, 분산된 배열이나 다른 자료구조를 통신하는데 특히 적합하다. 그러나 아직 HPF에서 이러한 연산을 제공하지 않는다.

셋째, 현재 HPF의 extrinsic 프로시쥬어에서 제공하는 매체를 사용할 수 있다. HPF 상에서 포트란 M같은 태스크 병렬 프로그램을 extrinsic 프로시쥬어로 호출할 수 있다. 이것은 HPF 프로시쥬어를 수행하는 처리기에서 포트란 M 프로시쥬어를 호출하는 방식이 되는데, 이때 각 처리기상에 분산되어 있는 배열형태 매개변수의 지역섹션(local section)뿐만 아니라, 모든 스칼라 변수 값도 전달하는 효과를 가질 수 있다.

다음은 HPF에서 포트란 M 프로시쥬어인 FM_PROC을 호출하는 코드이다. 여기서 EXTRINSIC 선언은 컴파일러에게 FM_PROC이 extrinsic 프로시쥬어임을 가리킨다. 호출되는 포트란 M 코드는 분산된 배열 매개변수처럼 전달되는 SEND문과 RECEIVE문을 포트상에 구성할 수 있다.

```
INTERFACE
    EXTRINSIC SUBROUTINE FM_PROC (PI. Y, Z)
    INTEGER PI(10)
    REAL Y(1000)
    REAL Z
    !HPF $ DISTRIBUTE PI(BLOCK), Y(CYCLIC)
    ENDSUBROUTINE
ENDINTERFACE
 . . .
CALL FM_PROC (A, B, C)
```

이러한 형태의 INTERFACE 구성시 문제점은 HPF가 포트 자료 형을 제공하지 않는다는 것이다. 그러나 앞으로 HPF의 완전한 명세가 구현되면 포트란 90의 추상화 자료 형인 IMPORT와 EXPORT 정의를 사용하여 이 문제를 해결할 수 있다. 그리고 완전하지는 않지만 HPF 연산내에 포트 배열의 상용화된 표현인 정수 배열 형태를 사용할 수도 있다.

5.2 Fx 포트란에서의 통합

5.1절이 포트란 M에서의 태스크 병렬성과 자료 병렬성을 통합하려는 제안임에 반해 Fx 포트란은 이미 통합이 어느 정도 진행된 상태이다[27, 28]. 앞에서 기술하였듯이 Fx 포트란을 태스크 병렬언어로 구분하기는 하지만, 태스크 병렬 구조상에서 호출하는 태스크 서브루틴은 자료병렬 구조의 프로시쥬어이다. 따라서 병렬섹션내의 독립된 자료 병렬 프로시쥬어에서 상호 통신되는 매개변수와 처리기에 할당된 상태를 고려하면서 병행적으로 호출하여 수행하는 방식이 된다.

예를들어 그림 7에서 두 개의 자료 병렬 서브루틴을 태스크로 취급하여 호출하는 주 프로그램을 볼 수 있다.

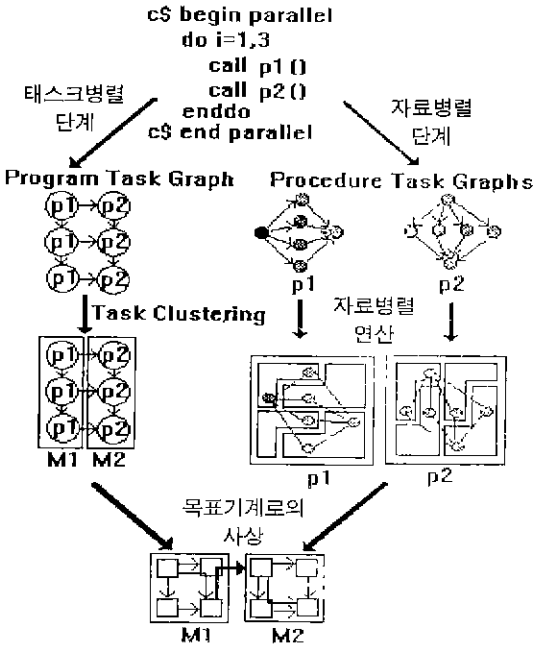


그림 7 Fx 포트란에서의 자료 병렬성과 태스크 병렬성의 예

여기서 컴파일은 자료 병렬 단계와 태스크 병렬 단계가 독립적으로 진행된다. 그래서 자료 병렬 단계는 배열 문장과 병렬 루프를 분석하여 병렬섹션내에서 호출하고 각각의 서브루틴을 위한 자료 병렬 코드를 생성한다.

그리고 태스크 병렬 단계에서는 병렬섹션내의 지시어들을 분석하여 태스크들간의 종속관계를 표현하는 계층적인 균일 태스크 그래프(uniform task graph : UTG)를 생성한다. 이 예에서는 UTG의 최상위 계층에 6개의 태스크가 포함되는데, 각 루프 반복당 2개의 태스크가 배정되어 있음을 보이고 있다. 컴파일러는 UTG를 모듈 단위로 분할한다. 이때 각 노드는 하나의 모듈을 표현하고, 각 아크는 모듈간의 가능한 통신을 표현한다. 한 모듈은 처리기 노드에 배치하는 단위가 되고, 포트란 77 컴파일러에서 컴파일되며 모듈들은 물리적 기계에 사상된다.

5.3 통합시 고려사항

이 장에서 기술한 포트란 M이나 Fx 포트란의 태스크 병렬기능과 HPF의 자료 병렬기능을 통합할 때 인터페이스(interface) 설계하려면 다음 사항들을 고려하여야 한다.

5.3.1 구조 측면

일반적으로 포트란 M같은 태스크 병렬언어는 연산에 할당되는 처리기 수를 수행시간에 결정하는 동적 태스크 병렬 연산형태를 지원하고, 여러 연산이 같은 처리기에 할당될 수도 있다. 따라서 프로그램 수행은 다중 쓰레드 형태이고, 처리기 할당은 수행시간의 매개변수에 의존한다. 반면, 포트란 D나 HPF는 이미 그 수가 결정된 전용 컴퓨터를 고려하므로 처리기당 하나의 쓰레드이고, 처리기 할당이 컴파일 시간에 결정된다.

이러한 차이로 인해 기존의 컴파일러로는 태스크 병렬언어와 자료 병렬언어의 통합이 쉽지 않다. 통합 시스템은 독립된 쓰레드로서 각 처리기에서 수행되는 다중 연산을 지원하여야 한다. 그러나 포트란 D나 HPF 컴파일러는 아직 이러한 기능을 제공하지 못하므로 통합 초기 단계에서는 우선 한 개의 태스크 병렬 프로시쥬어와 자료 병렬 프로시쥬어가 각 실행 처리기에 할당되도록 고려하는 것이 바람직하다.

통합 시스템에서는 태스크 병렬언어의 프로세스를 수행하는 가상 컴퓨터 크기나 실제 위치를 컴파일 시간에 알 필요가 없다. 따라서 태스크 병렬언어의 프로세스는 이 가상 컴퓨터의 표현내용을 호출 루틴인 자료 병렬언어 프로시

쥬어에 보내야 하고, 자료 병렬언어 컴파일러는 이 내용에 의해 확정된 처리기를 사용하는 코드를 생성할 수 있어야 한다.

5.3.2 매개변수로 전달되는 자료의 형

태스크 병렬언어와 자료 병렬언어의 컴파일러와 실행 처리기는 공통적인 자료 표현을 사용하거나, 다른 표현들간의 변환 기능을 제공하여야 하고 재분산 연산도 필요하다. 태스크 병렬언어와 자료 병렬언어의 컴파일러가 프로시쥬어간 분석없이 언어 영역에서 정확한 코드를 생성하기 위해서는 인터페이스(interface) 명세가 필요하다. 자료 병렬언어가 extrinsic 프로시쥬어 기능을 사용하여 태스크 병렬언어 프로그램을 호출하기 위해서는 프로시쥬어의 매개변수 전달을 위한 변환이 필요하다.

특히 포트란 M같은 태스크 병렬언어에서의 모든 프로세스는 독립적인 메모리 영역을 갖는 반면, HPF같은 자료 병렬언어에서는 개념적으로 모든 메모리가 공유된다. 이 경우 포트란 M의 지역 주소를 HPF의 전역 주소로 번역하는 방법이 필요하다.

5.3.3 분석과 최적화

자료 병렬언어의 컴파일러는 프로그램 분석에 의존한다. 예를 들어, Rice 대학의 포트란 77D 컴파일러는 메모리를 할당하고, 연산을 병렬화하고, 통신을 관리하기 위해 intra-, inter-procedural analysis를 구성하고 있다. 이 컴파일러에서는 어떤 프로시쥬어에서 수행하기 시작하고, 어떤 프로시쥬어가 호출되고, 어떤 자료가 호출된 프로시쥬어에 의해 접근되는가 등의 프로그램 관련 정보를 필요로 한다.

통합 시스템에서 컴파일러는 이 두가지 포트란 유형을 직접 분석하고, 분석될 수 없는 루틴에 대해서는 주어진 자료에 의존하여 이 정보를 생성하여야 한다. 이를 위하여 다음을 고려할 수 있다. 우선 수집된 태스크 병렬언어 관련 정보를 보조화일을 사용하여 자료 병렬언어 컴파일러에게 제공할 수 있다. 이를 위해 가능한 상호 연결 방안으로 overlap 영역을 설정할 수 있다.

또 다른 형태는 태스크 병렬언어와 통합될 자료 병렬언어 컴파일러의 효과를 고려한 상호 연결방안이다. 일반적으로 HPF 컴파일러는 한 개의 전용 기계를 위한 코드 또는 공용 기계의 전용 부분을 위한 코드를 생성한다고 가정한다. 따라서 연산과 통신의 granularity의 결정은 그 전용 환경하에서 추정된 성능에 좌우된다. 이러한 가정은 포트란 M과 HPF의 다중 쓰레드를 단일 처리기상에서 수행하는 포트란 M의 쓰레드 환경에서 명확히 나타낼 수 있다. 그러나 자료병렬 컴파일러에 의해 최적화된 코드상에서의 이러한 다중 프로그래밍 효과에 관련된 연구는 아직 없다.

6. 결 론

본 고에서는 고성능 병렬 컴퓨터 시스템 환경하에서 효과적인 병렬 수행을 지원하기 위해 제안된 고성능 병렬언어에 촛점을 맞추어, 이들 중 대표적인 연구분야인 자료 병렬언어와 태스크 병렬언어 분야에서의 연구내용을 살펴 보았다. 그리고 대표적인 자료 병렬언어와 태스크 병렬언어들 중 자료 및 태스크 병렬성의 표현, 처리측면에서 그 구조 및 특성을 살펴 보았다.

그런데 자료 병렬성과 태스크 병렬성이 서로 보완적인 관계를 갖고 있으므로 고성능 언어가 두 병렬성을 동시에 표현할 수 있다면, 고성능 언어가 갖추고 있는 융통성과 이식성을 기반으로 응용 소프트웨어 분야에서 보다 다양한 표현성을 제공하게 될 것이다. 따라서 본 고에서는 자료 병렬성과 태스크 병렬성을 통합하고자 하는 측면에서 수행된 연구와 고려사항들도 살펴 보았다.

현재 고성능 병렬언어 분야에서 HPF 같은 자료 병렬언어에서는 태스크 병렬성을 지원하고자 하고, 포트란 M같은 태스크 병렬언어에서는 자료 병렬성을 지원하려는 연구들이 활발하게 진행되고 있다. 또한 이러한 시도들이 새로운 언어 개념을 추가한다기 보다는 기존의 언어 구조 및 지시어를 적절하게 활용하려고 하고 있다. 따라서 본 고에서 기술한 자료 병렬성과 태스크 병렬성의 구조·특성과 두 병렬성의 통합에 관한 내용이 고성능 병렬언어 개발에 중요한 고려사항이 될 것으로 보이며, 이와 같

은 병렬 프로그래밍 패러다임은 대규모 병렬 컴퓨터 환경에서의 소프트웨어 개발에 커다란 영향을 끼치게 될 것이다.

참 고 문 헌

[1] Applied Parallel Research, "Forge 90 distributed memory parallelizer : User's Guide, Version 8.0 Edition," Placerville, CA, 1992.

[2] A. Beguelin, J. Dongarra, G. Geist, R. Manchek, and V. Sunderam, "Graphical Development Tools for Network-Based Concurrent Supercomputing," *Proc. of Supercomputing,* Albuquerque, N.M, pp.435-444, 1991.

[3] T. Brandes, "Automatic Translation of Data Parallel Programs to Message Passing Programs," *Proc. of AP'93 Int'l Workshop on Automatic Distributed Memory Parallelization, Automatic Data Distribution and Automatic Parallel Performance Prediction,* Saarbrucken, Germany, 1993.

[4] T. Brandes, "Evaluation of High Performance Fortran on Some Real Applications," GMD, St. Augustin, Germany, Mar. 1994.

[5] T. Brandes, "Compiling Data Parallel Programs to Message Passing Programs for Massively Parallel MIMD Systems," GMD, St. Augustin, Germany, Mar. 1994.

[6] D. Callahan and K. Kennedy, "Compiling Programs for Distributed-Memory Multiprocessors," *Supercomputing 2,* pp.151-169, 1988.

[7] N. Carriero and D. Gelernter, "Linda in Context," *Communication of the ACM,* Vol. 32, No.4, pp.444-458, 1989.

[8] C. Chase, A. Cheung, A. Reeves and M. Smith, "Paragon : A Parallel Programming Environment for Scientific Applications using Communication Strutures," *Proc. of the Int'l Conf. on Parallel Processing*(II), St. Charles, Illinois, pp.211-218, 1991.

[9] B. M. Chapman, P. Mehrotra and H. Zima, "Vienna Fortran-A Fortran Language Extension for Distributed Memory Multiprocessors," Univ. of Vienna, Austria, 1991

[10] B. Chapman, P. Mehrotra and H. Zima, "Programming in Vienna Fortran," *Scientific Programming,* 1(1) pp.31-50, Fall 1992.

[11] M. Chandy, I. Foster, K. Kennedy, C. Koelbel and C. Tseng, "Intergrated Support for Task and Data Parallelism," *Int'l. Journel of Supercomputer Applications,* 1993.

[12] I. Foster, "Fortran M as a Language for Building Earth System Model," Technical Report CRPC-TR92444, Rice Univ., 1992.

[13] I. Foster and M. Chandy, "Fortran M Language Definition," Technical Report CRPS-TR93429, Rice Univ., Aug, 1993.

[14] I. Foster, "Task Parallelism and High-Performance Languages," *IEEE Parallel & Distributed Technology,* Vol.2, No.3, pp.27-36, Fall 1994.

[15] I. Foster and M. Chandy, "Fortran M : A Language for Mudular Parallel Programming," to appear in *Journel of Parallel and Distributed Computing,* 1994.

[16] G. Fox, S. Hiranandani, K. Kennedy, C. Koelbel, U. Kremer, C.W. Tseng and M. Y. Wu, "Fortran D Language Specification," Technical Report CRPC-TR90079, Rice Univ., Dec. 1990.

[17] T. Gross, D. R. O'Hallaron and J. Subhlok, "Task Parallelism in a High Performance Fortran Framework," *IEEE Parallel & Distributed Technology,* Vol.2, No.3, pp.16-26, Fall 1994.

[18] P. J. Hatcher, "Data-Parallel Programming on MIMD Computers" *IEEE Parallel & Distributed Technology,* Vol.3, NO.2, pp.377-383, Jul. 1991.

[19] P. J. Hatcher, "The Impact of High Performance Fortran," *IEEE Parallel & Distributed Technology,* Vol.2, No.3, pp.13-15, Fall 1994.

[20] High Performance Fortran Forum, "High Performance Fortran Language Specification, Version 1.0," Technical Report CRPC-TR92225, Rice Univ., 1993.

[21] K. Ikudome, G. Fox, A. Kolawa and J. Flower, "An Automatic and Symbolic Parallelization System for Distributed Memory Parallel Computers," *Proc. of the 5th*

Distributed Memory Computing Conference, Charleston, S.C., 1990.

[22] C. Koelbel and P. Mehrotra, "Compiling Global Name-Space Parallel Loops for Distributed Execution," *IEEE Parallel and Distributed Systems,* Vol.2, No.4, pp.440-451, 1991.

[23] C. Koelbel, D. Loveman, R. Schreiber, G. Steele, Jr. and M. Zosel, *The High Performance Fortran Handbook,* MIT Press, 1994.

[24] P. Newton and J. C. Browne, "The CODE 2.0 Graphical Parallel Programming Language," *Proc. of the ACM Int'l Conf. on Supercomputing,* Washington, D.C, pp.167-177, 1992.

[25] M. Rosing, R. Schnabel and R. Weaver, "The DINO Parallel Programming Language," *Journel of Parallel and Distributed Computing,* Vol.13, No.1, pp.30-42. 1991.

[26] H. Srinivasan and M. Wolfe, "Analyzing Programs with Explicit Parallelism," Languages and Compilers for Parallel Computing, No. 589 in Lecture Notes in Computer Science, Springer-Verlag, pp.403-419, 1992.

[27] J. Subhlok, J. M. Stichnoth, D. R. O'Hallaron and T. Gross, "Exploiting Task and Data Parallelism in a Multicomputer," *Proc. of ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming,* ACM Press, New York, pp. 13-22, 1993.

[28] J. Subhlok, D. R. O'Hallaron and T. Gross, "Task Parall Programming in Fx," Technical Report CMU-CS-94-12, Carnegie Mellon Univ., Pittsburgh, 1994.

[29] C. Tseng, "An Optimizing Fortran D compiler for MIMD Distributed Memory Machines," Ph.D thesis, Dept. of Computer Science, Rice Univ., 1992.

[30] B. J. N. Wylie, M. G. Norman and L. J. Clarke, "High Performance Fortran : A Perspective," Technical Report EPCC-TN92-05.07, Univ. of Edinburgh, May 1992.

[31] H. Zima, H. J. Bast and M. Gerndt, "SUPERB : A Tool for Semi-Automatic MIMD/SIMD Parallelization," *Parallel Computing 6,* pp.1-18, 1988.

[32] H. Zima, H. Brezany, B. Chapman, P. Mehrora and A. Schwald, "Vienna Fortran-a Language Specification," In Preparation. Austrian Center for Parallel Computation, Univ. of Vienna, Austria, 1991.

박 성 순

1984 홍익대학교 전자계산학과 학사
1987 서울대학교 계산통계학과 석사
1988~1990 공군사관학교 전산학과 전임강사
1994 고려대학교 전산과학과 박사
1994~현재 안양대학교 전자계산학과 전임강사
관심분야 : 병렬 컴파일러, 병렬 언어 등

구 미 순

1987 고려대학교 수학과 학사
1995~현재 고려대학교 전산과학과 석사과정
관심분야 : 병렬 컴파일러, 병렬 언어 등

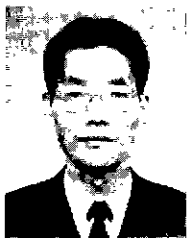
최 성 욱

1994 고려대학교 전산과학과 학사
1994~현재 고려대학교 전산과학과 석사과정
관심분야 : 병렬 컴파일러, 병렬 언어 등

육 현 규

1994 고려대학교 전산과학과 학사
1994~현재 고려대학교 전산과학과 석사과정
관심분야 : 병렬 컴파일러, 병렬 언어 등

박 명 순



1975 서울대학교 전자공학과 학사
1982 University of Utah 전기공학과 석사
1985 Universty of Iowa 전기전산공학과 박사
1975~1980 국방과학연구소 연구원
1985~1987 Marquette 대학교 조교수
1987~1988 포항공과대학 전자전기공학과 조교수
1988~현재 고려대학교 전산과학과 교수

관심분야 : 컴퓨터구조, 병렬 컴파일러 등

논문모집

- 행 사 명 : **제7회 한글 및 한국어 정보처리 학술대회**
- 개최일자 : 1995년 10월 6일(금)~7일(토)
- 개최장소 : 연세대학교
- 주 최 : 한국인지과학회 · 한국정보과학회
- 초록마감 : 1995년 9월 1일(금)
- 원고마감 : 1995년 9월 20일(수)
- 문의 및 제출처 :

C/O KLP '95 papers
서울시 서대문구 신촌동 134
연세대학교 영어영문학과, ㉮120-749
이 익 환 교수
Tel : 02-361-2315, Fax : 02-313-3676
E-mail : ihlee@bubble.yonsei.ac.kr

C/O KLP '95 papers
서울시 서대문구 신촌동 134
연세대학교 컴퓨터과학과, ㉮120-749
이 일 병 교수
Tel : 02-393-2186, Fax : 02-365-2579
E-mail : yblee@csai.yonsei.ac.kr