□기술해설□

시스템 독립적인 병렬 프로그래밍 언어 High Performance Fortran

시스템공학연구소 문경덕* · 반난주** · 김태근* · 김중권* · 유여백*

목 차

1. 서 론

거대 과제(Grand Challenges)라 불려지는 유체 역학, 기후 예측, 입자 물리 등 과학 및 공학 분야의 많은 문제들은 불규칙적이고 대용량의 자료 처리를 위하여 고성능 컴퓨팅 기술을 요구한다. 이러한 고성능 컴퓨팅을 위해 80년대 중반 이후 병렬 처리 시스템이 도입되었다. 그러나, 병렬 시스템은 기존의 벡터 슈퍼 컴퓨터에 비하여 뛰어난 장점에도 불구하고, 과학 및 공학 분야에 제한적으로 사용되어 왔다. 병렬 처리 시스템에서 프로그램을 작성하는 것은 하드웨어적인 시스템의 특성과 응용 프로그램의 병렬화를 고려해야하기 때문에 기존의 순차 프로그래밍에 비하여 매우 어렵고, 지원 도구의 개발이 미진하여 일반 사용자가 이용하기 위해서는 많은 노력이 필요하다. 따라서, 병렬 시스템 하드웨어 및 소프트웨어 개발자들은 사용자에게 보다 편리한 프로그래밍 환경을 제공하고, 시스템 독립적으로 이용할 수 있는 고성능 컴퓨팅 언어 및 도구 개발에 관심을 가지게 되었다.

초기의 병렬 프로그래밍 언어에 관한 연구는 벡터화 기법을 이용하여 병렬성을 검사하여 자동적으로 순차 프로그램을 병렬 프로그램으로 변환하는 도구의 개발에 관한 연구가 주류를 이루었으나, 순차 프로그램의 병렬화 도구는 프로그램의 성능 향상에 문제가 있음이 발견되었다. 따라서 사용자가 병렬성에 대한 정보를 표현하고, 컴파일러가 이 정보를 이용하여 병렬 프로그램을 생성하는 방식의 데이터 병렬 프로그래밍 모델이 연구되어 왔다.

현재 상용화된 Thinking Mahines의 CM-5, Intel의 Paragon, IBM의 SP2 등의 병렬 시스템은 프로세서간의 interconnection network과 시스템이 제공하는 통신 라이브러리 등이 서로 달라 시스템간에 프로그램의 호환성이 결여되어 병렬 시스템이 제한적으로 활용되어 왔다. 이에 병렬 프로그래밍의 폭넓은 확산을 위하여

*정회원
**비회원

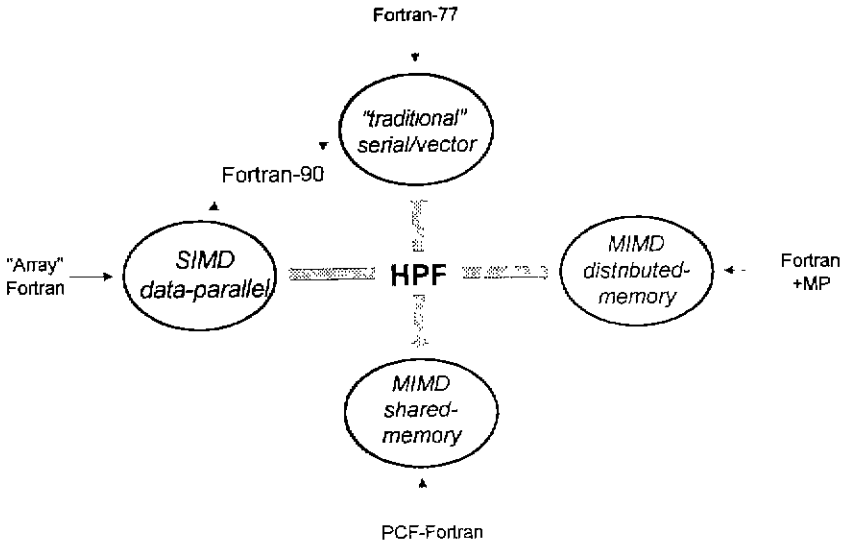

그림 1 High Performance Fortran의 역할

시스템 독립적인 병렬 프로그래밍 언어의 표준화에 산업체 학계 연구 기관이 참여하여 High Performance Fortran(HPF)라는 언어를 제정하게 되었다.

그림 1에서 보여지는 바와 같이 HPF 언어는 순차 및 병렬 프로그래밍 언어의 특성을 고려하여 다양한 구조의 시스템에서 수행할 수 있도록 제안되었다. Fortran 언어의 표준 병렬 프로그래밍 언어로서 제정된 HPF는 Fortran 77과 Fortran 90의 대부분의 문법을 지원하며, 전통적인 순차적 시스템부터 공유 메모리·분산 메모리 등의 시스템에 독립적인 코드를 작성할 수 있도록 하였다.

HPF는 기존의 분산 메모리 병렬 시스템에 대한 언어와는 달리 사용자에게 단일 제어와 전역 주소 공간 체계를 지원한다. 분산 메모리 병렬 시스템에 대한 성능을 최대한 활용하기 위해, 컴파일러 단계에서 프로그램을 분석하여 특성에 따라 데이터와 연산을 각 프로세서에 분할하는 방식을 채택하고 있다. 즉, 컴파일러는 HPF로 작성된 프로그램으로부터 사용자가 정의한 지시어를 분석하여 실제 메세지 패싱을 이용하여 데이터 및 연산을 프로세서에 분할하고 연산에 필요한 데이터의 프로세서간 교환 기능을 삽입한 SPMD 코드를 생성한다.

본 논문에서는 현재 메모리 병렬 시스템에 대한 표준으로 제정중인 HPF 언어를 소개하기 위해, 2장에서는 기존 Fortran 언어의 역사와 특징 및 HPF가 제정된 역사에 대해 논한다. 3장에서는 HPF 언어에서 제공하는 기능 중 병렬화와 관련하여 추가된 사항들에 대하여 논하고 4장에서는 현재 개발되어 있는 HPF 컴파일러에 대해 논한다. 그리고 5장에서는 현재 추진중인 HPF 기능 보완 작업에 대해 살펴보고 마지막으로 6장에서 결론을 맺는다.

2. HPF 역사

병렬 시스템에 적합한 Fortran 언어를 개발하려는 노력은 1970년대초 COMPASS의 Iliac IV SIMD 컴퓨터를 위한 IVTRAN 컴파일러 개발을 시작으로 하여, 80년대 COMPASS와 Thinking Machines에서 정적 데이터 배열 지시어를 이용한 Fortran 8x를 개발하기에 이르렀다. 그러나, 이들 언어는 병렬 시스템을 효율적으로 활용하기에 미비한 부분이 많아, 1990년대초 병렬 시스템을 위한 표준 언어인 Fortran 90을 제정하였다. HPF는 Fortran 90을 확장한 언어로 병렬 프로그래밍에 필요한 기능들을 추가 정의하여 제정한 것이다. 본 장에서는 HPF

의 기반이 되는 Fortran 90의 기능과 HPF의 제정 과정을 중심으로 논의한다.

2.1. Fortran 90

1990년대 Fortran 90이 분산 메모리 병렬 시스템에 대한 국제 표준으로 제정되었다. Fortran 90은 분산 메모리 시스템에서 데이터 지역성을 이용하여 성능을 향상시킬 수 있다는 개념을 기초로 하여 단일 제어와 전역 데이터 접근방식을 통해 병렬화를 구현하였다. Fortran 90은 순차적 시스템에 대한 표준 언어인 FORTRAN 77의 기능을 확장한 언어로 FORTRAN 77과의 호환성을 지원하고 있다. Fortran 90이 FORTRAN 77에 대하여 추가적으로 지원하는 기능들은 다음과 같다.

- 배열 연산 : 배열의 각 요소들에 대하여 동일한 연산을 수행할 때 DO 루프로 반복하는 코드를 쓰는 대신 전체 배열로 연산 수행을 표현할 수 있도록 하였다. 예를 들어 A = B + C는 배열 B와 C의 각 요소를 더하여 배열 A의 상응하는 위치에 저장한다.
- 향상된 산술 연산 기능 : Fortran 90 표준은 수치의 정밀도 표현에 대한 이식성과 프로그램 수행중 특정 시스템에 대한 수치 표현 특성을 사용자가 결정할 수 있는 내장 함수를 제공한다.
- 데이터형의 확장 : 정수형과 문자형의 종류[1]를 늘리고, packed logical을 추가하였으며 REAL과 COPMLEX에 대하여 정밀도를 증가시켰다.
- 사용자 정의 데이터형 : 사용자가 임의의 데이터 구조와 이 구조에 대한 연산을 정의할 수 있다.
- 데이터와 프로시듀어의 모듈화 : Fortran 90이 지원하는 것중 가장 뛰어난 기능으로 관련된 데이터 선언, 유도형 정의 및 프로시듀어를 MODULE이라는 단위로 구성하는 것을 말한다. 이 모듈은 프로그램의 어느 부분에서든 접근하여 사용할 수 있으며 FORTRAN 77의 ENTRY문을 향상시킨 것이다.
- Fortran 90에서는 모듈화 기능이외에 인터페이스 블럭을 정의하여 다른 서브 프로그램과의 인터페이스를 확인, 최적화하는데 사용될 정보를 명시할 수 있다.
- 기억 장소 종류의 추가 : ALLOCATABLE, AUTOMATIC 그리고 가상형 객체와 같은 새로운 종류의 기억 장소와 포인터 기능들을 추가하여 순차적 메모리 형태에 의존한 FORTRAN 77 언어의 비효율적인 특성[2]들을 감소시켰다.
- 내장 함수의 추가 : 배열에 대한 산술 연산, 수치의 정밀도, bit 조작 등을 수행하는 내장 함수 및 프로시듀어들을 추가하였다.
- 기타 향상된 사항 : Fortran 90에서는 문자열에 "_"를 포함시키고 길이도 31자까지 허용하며, 원시 코드의 형식을 자유롭게 하는등의 표현 형식에서의 제약을 줄였다. 이외에도 CASE문, 내부 서브프로시듀어 등 새로운 기능 및 표현 형식들을 추가시켰다.

2.2. HPF

Fortran 90이 대규모 병렬 시스템의 성능을 최대로 활용하는데 한계점을 나타내자, 새로운 언어를 정의하려는 연구가 수행되었다. 이런 연구들은 모두 사용자가 직접 프로세서간 통신을 수행하는 코드를 작성하는 대신, 사용자가 정의한 방법을 컴파일러가 분석하여 프로세서간 통신을 수행하는 코드를 삽입하는 방식의 데이터 병렬 프로그래밍을 추구하고 있다. 관련 연구로는 Rice 대학과 Syracuse 대학에서 공동으로 수행한 Fortran D와 Vienna 대학의 Vienna Fortran을 비롯하여 ICASE의 Kali 및 Yale 대학의 Yale Extension 등이 있다 [3].

이러한 노력과 더불어 병렬 프로그래밍 환경의 폭넓은 수용을 위해 워크스테이션에서 초병렬 수퍼컴퓨터까지 이식가능하고 시스템의 성능을 최대한 활용할 수 있는 표준 언어 제정이 필요하게 되었다. 이에 1991년 디지탈사는 하드웨어 구조에 독립적이면서도 산업계와 학계 모두 표준 언어로 받아들일 수 있는 고성능 Fortran

1 Short integer, very large character set

2 EQUIVALENCE 문, COMMON 문 그리고 실인수/가인수사이의 배열 형태 변환

언어에 대한 표준 제정을 목표로 Fortran 90과 Fortran D를 기반으로 하는 HPF 프로젝트에 착수하였다.

High Performance Fortran Forum(HPFF)은 1992년초 Rice 대학의 Ken Kennedy를 중심으로 고성능 컴퓨팅을 위한 표준 Fortran 제정을 목적으로 결성되었다. 40여개 이상의 산학 연구기관이 비영리로 결성한 HPFF의 목적은 SIMD(Single Instruction Multiple Data), MIMD(Multiple Instruction Multiple Data) 시스템 및 벡터 프로세서 등의 초병렬 시스템에서 높은 성능을 보이고 이들간에 이식이 가능하도록 기능이 확장된 Fortran을 개발하는 것이었다. 1993년말 HPFF에서는 그들의 모임에서 동의한 사항을 엮어 HPF 언어 사양 1.0 이라는 문서를 발표하였으며, 이 문서에는 HPF fullset과 subset이 정의되어 있다.

1990년대 초병렬 시스템에 대한 표준 언어로 제정된 Fortran 90은 이를 지원하는 컴파일러의 개발이 늦어지면서 널리 활용되지 못하였다. 이에 HPFF는 보다 빠른 시일내에 HPF 컴파일러를 개발하여 HPF 언어를 폭넓게 보급해야 할 것을 절감하였다. 그러나, HPFF에서 제정한 문서 사양 1.0에서 지원되는 모든 기능을 지원하는 HPF 컴파일러 개발은 기술적으로 해결해야 할 요소가 많아 Fortran 90 컴파일러 개발보다 많은 시간이 소요될 것으로 예상되었다. HPFF는 컴파일러의 빠른 개발과 상용화를 가속화시키기 위해 구현시 어려움이 있는 요소를 제외하고 반드시 지원되어야 할 사항을 HPF subset이라 정의하였다.

HPFF 1차 모임에서는 1993년에 "HPF 문서 사양 1.0"을 정의하면서 태스크 분할, 비정규 문제 지원 및 병렬 입 출력에 관련된 기능 등은 기술적인 요인 등으로 인하여 추후에 정의하기로 하고, 이들을 Journal of Development에 남겨두었다. 1994년에 HPFF 1차 모임에 참여한 여러 단체에서는 HPFF 1차 모임에서 정의한 "HPF 문서 사양 1.0"의 기능을 검증하고, Journal of Development에 기술된 기능 및 미비한 기능의 보완이 필요함을 인식하고 HPFF 2차 모임을 결성하였다. HPFF 2차 모임에서는 1996년까지 HPF 문서 사양 1.0의 기능을 검증하고 미비한 기능을 보완하여 HPF 문서 사양 2.0을 정의할 예정이다.

3. High Performance Fortran

3.1. HPF의 제정 목표와 범위

효율적인 병렬 프로그래밍을 위해서는 우선 적절한 병렬 알고리즘이 개발되어야 한다. 그러나 병렬 알고리즘의 개발은 결코 용이하지 않으며, 이러한 부담이 병렬 프로그래밍의 확산을 저해하는 요소로 작용해 왔다. 이러한 문제점을 극복하기 위한 방법의 하나로 기존의 알고리즘을 이용하되, 동일한 연산이 수행되는 데이터들을 발견하여 이들을 각 프로세서에 분할하고 동시에 수행하도록 하는 것이었다. 즉 데이터의 병렬화를 기초한 병렬 프로그래밍 언어의 개발이 병렬 처리의 보급에 적절한 해결책이었다.

그러나, 일반 사용자가 각 시스템 업체에서 제공하는 Fortran 컴파일러를 이용하여 프로그램을 개발하는 경우, 먼저 시스템의 특성을 파악하고 프로세서간 통신이 발생되는 부분을 직접 작성해야 한다. 이런 일련의 작업들은 오랜 시간의 노력이 요구되며, 오류 발생 가능성이 높아 효율적인 병렬 프로그램을 개발하는데 장애 요소로 작용해 왔다. 또한, 기존의 Fortran 언어들은 시스템 의존적이므로 동일한 프로그램을 다른 시스템에서 실행시킬 경우 코드를 새로 작성해야 하는 번거로움이 있었다. 이때, 시스템 특성을 파악하고 오류 수정에 걸리는 시간을 줄여 병렬 알고리즘 개발 등을 위한 시간으로 활용하기 위하여, 시스템 의존적 특성을 제거하여 타 시스템과의 호환성을 제공하는 표준 데이터 병렬 프로그래밍 모델에 대한 연구를 수행하였다. 데이터 병렬 프로그래밍 모델에서는 사용자가 데이터 분할 및 정렬에 관련된 지시어를 이용하여 병렬로 수행되는데 필요한 정보를 컴파일러에 제공하면, 컴파일러가 이 정보를 이용하여 프로세서간 통신 부분을 생성하여 준다.

지난 수년간 많은 연구자들은 병렬 시스템에 대한 표준인 Fortran 90의 미비한 기능을 보완하여 시스템 독립적인 데이터 병렬 프로그래밍 모델에 대한 언어를 개발하려는 연구를 수행해

왔다. HPF는 이러한 노력의 일환으로 제정된 표준 데이터 병렬 언어이며, HPF의 제정 목적은 다음 세가지이다.

- 단일 제어(single thread), 전역 이름 공간, 불완전 동기(loosely synchronous) 병렬 계산에 기반을 둔 데이터 병렬 프로그래밍
- MIMD 또는 SIMD와같이 메모리 접근 비용이 일정치 않은 시스템에서 최고 성능 획득
- 다양한 컴퓨터 구조에 적합하게 코드를 조정하는 기능(code tuning)

위의 제정 목적을 만족시키기 위해 Fortran 90이 제공하는 기능외에 데이터 정렬 및 분할문, 병렬 실행문, 내장 함수와 표준 라이브러리의 확장, EXTRINSIC 프로시듀어의 네가지 기능을 추가하였다. 이 중 병렬 실행 가능한 부분을 명시하는 FORALL 구문과 기본적인 내장 함수들은 첫번째와 두번째의 HPF 제정 목적을 만족시키기 위하여 추가되었다. 그리고 데이터 정렬 및 분할문 그리고 EXTRINSIC 프로시듀어는 시스템간에 이식성을 제공하기 위해 추가된 기능이다.

3.2. 데이터의 정렬과 분할

일반적으로 병렬 프로그램의 수행 성능은 연산을 수행하는데 소요되는 시간과 연산에 필요한 데이터를 교환하는데 소요되는 시간에 의해 결정된다. 이는 하나의 연산에 필요한 데이터들은 동일한 프로세서에 있어야하나, 분산 메모리 구조의 병렬 시스템에서는 데이터들이 여러 프로세서에 분할되어 있기 때문이다. 즉 병렬 프로그래밍의 성능은 프로세서간 데이터 통신 시간에 의해 크게 좌우되므로, 하나의 연산에 필요한 데이터는 되도록 같은 프로세서에 놓이도록 정렬하고 분할하는 것이 필요하다.

HPF에서는 사용자가 데이터의 정렬 및 분할 방법을 지정할 수 있는데, !HPF$, *HPF$ 또는 CHPF$ 등의 접두어로 시작되는 지시어를 이용하여 데이터 객체의 분할 방식에 대한 정보를 컴파일러에 알릴 수 있다. ALIGN, TEMPLATE, DISTRIBUTE, PROCESSORS 등은 선언부에 올 수 있는 지시어로 데이터의 정적 정렬 및 분할을 지시하며, REALIGN, REDISTRIBUTE 등은 이미 선언된 데이터의 분할 방식을 동적으로 변경할 수 있는 지시어로 이들은 실행부에 올 수 있고 DYNAMIC으로 선언된 데이터에 한하여 적용할 수 있다.

컴파일러는 사용자가 정의한 데이터 분할과 관련된 정보를 분석하여 그림 2에서와 같이 두 단계 매핑 과정을 거쳐 데이터를 프로세서로 분할하는 기능을 수행한다. 첫번째 단계에서는 선언된 배열을 논리적 배열 공간인 템플리트에 정렬하여 데이터의 접근 형태에 따라 프로세서간 데이터 교환이 적게 발생하도록 데이터 접근 형태를 일치시켜 동일 프로세서에 분할되게 한다. 두번째 단계에서는 정렬된 데이터들을 격자 구조의 논리적 프로세서에 분할한다. 그리고, 논리적 프로세서로부터 물리적 프로세서로의 매핑은 시스템 의존적인 작업이며, 이는 사

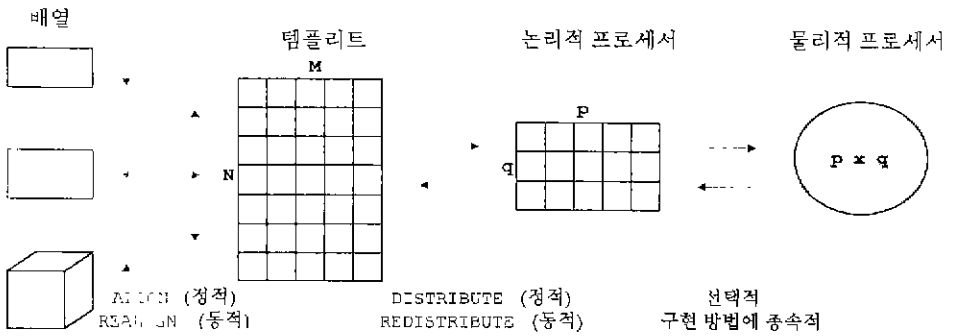

그림 2 데이터를 프로세서에 매핑하는 과정

용자와는 무관하게 컴파일러에 의해 수행된다. HPF에서는 이렇게 데이터를 물리적 프로세서로 매핑하는 대신 논리적 프로세서를 대상으로 매핑을 수행하므로, 논리적 프로세서와 물리적 프로세서 사이의 매핑 부분만 수정하므로 시스템사이의 이식성을 제공한다.

3.2.1. ALIGN 지시어

효율적인 데이터 병렬 프로그래밍을 위해서는 데이터를 프로세서에 분할하기에 앞서 데이터의 참조 형태를 고려하여 하나의 연산 수행시 필요한 모든 데이터가 동일 프로세서에 있도록 고려한 데이터 정렬 과정이 필요하다. ALIGN 지시어는 데이터 객체들이 정렬될 형태를 정의할 수 있는 기능을 제공한다. 이를 위해 사용자는 서로 다른 참조 형태를 갖는 데이터를 정렬하기 위해 데이터들이 공통으로 사용하는 템플리트라는 논리적 공간을 선언할 수 있다. 즉, 데이터의 지역성을 높이기위해 연산이 수행될 때 가장 관련이 깊은 데이터들이 가까이 위치할 수 있도록 데이터 객체들을 재배열하는 것이다. 또한 정렬된 데이터는 각 요소를 개별적으로 프로세서에 분할하는 번거로움 없이 한번에 프로세서에 분할할 수 있다.

```
!HPF$ ALIGN X(I) WITH B(I+2) ! 이동
!HPF$ ALIGN X(J,K) WITH D2(K, J) !
이항
!HPF$ ALIGN X(:, *) WITH D(:) ! 삭제
!HPF$ ALIGN X(*) WITH D(:, *) ! 복사
!HPF$ ALIGN X(J, K) WITH D2(M-J
+1, N-K+1)
                ! 역순, M, N은 배열의 크기
!HPF$ ALIGN X(J, K) WITH D3(J, *, K)
!HPF$ ALIGN A(I) WITH D(I*2) !
간격유지
!HPF$ ALIGN X(:, :) WITH D2(:, :) !
변동없음
!HPF$ ALIGN (:, :) WITH D2(:, :) :: X
! 위와 같은 의미를 갖는 표현식
```

컴파일러는 정렬문에 정의된 배열들의 첨자 관계를 이용하여 함수식으로 나타내며, 템플리트에 정렬되는 데이터들은 이 함수를 이용하여 그림 3과 같이 가로 혹은 세로 방향의 이동, 축의 복사나 삭제, 요소간 일정 간격 유지, 이항(transpose) 그리고 역순 등의 형태로 정렬될 수 있다. 그림 3의 (a)는 배열 X의 첫번째 요소를 템플리트의 세번째 요소로 정렬하며, (b)는 배열 X를 템플리트 D2에 정렬할 때 행과 열을 바꾸어 정렬한다. (c)는 배열 X의 각 행의 요소들을 템플리트 D의 동일한 요소에 정렬하며, (d)는 1차원 배열 X의 각 요소들은 템플리트 D의 각 열의 상응하는 행으로 동일하게 복사된다.

정렬문에서 ":"는 그 축에 있는 배열의 요소

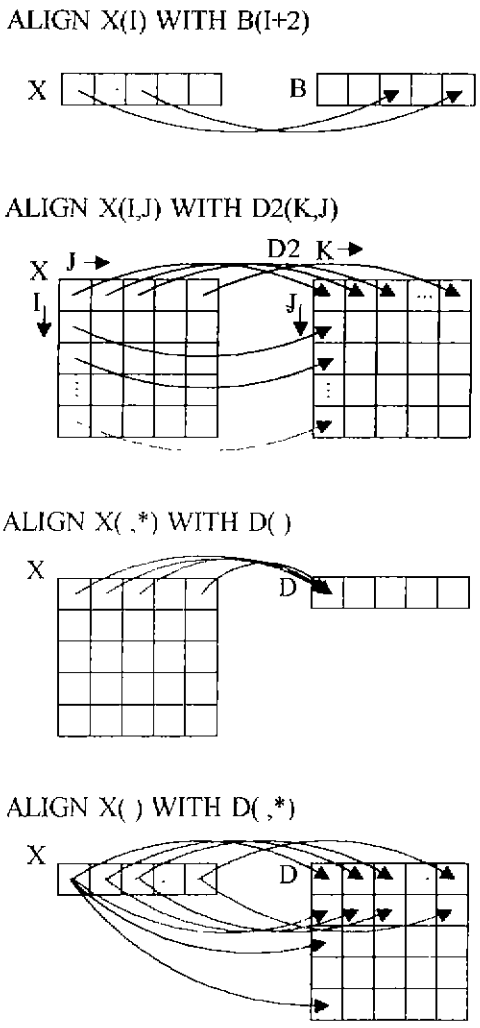

그림 3 데이터 정렬

들이 정렬될 템플리트의 상응하는 축에 순서대로 정렬되는 것을 의미하고, "*"는 해당 축이 정렬될 때 삭제 또는 복사되는 것을 의미한다.

ALIGN 지시어는 선언부에 올 수 있으며, 데이터의 정적 정렬을 지시하고 컴파일할 때 한 번 수행되고 만다. 반면 동적으로 데이터를 정렬해야할 때 실행부에 REALIGN 지시어를 사용하면 동적 정렬이 가능하다. 사전에 DYNAMIC으로 선언되어진 데이터는 REALIGN 지시어를 적용할 수 있다.

3.2.2. DISTRIBUTE 지시어

ALIGN 지시어가 한 연산이 수행될 때 필요한 데이터에 대해 가능한 프로세서간 데이터 교환이 적게 발생하도록 데이터끼리의 관계를 정의하는 반면, 부하 균형과 통신 비용사이의 tradeoff를 잘 유지할 수 있도록 데이터를 논리적 프로세서에 분할하는 것을 정의하는 것이 DISTRIBUTE 지시어이다. 템플리트에 정렬된 데이터는 DISTRIBUTE 지시어를 이용하여 격자 구조의 논리적 프로세서에 매핑할 수 있다.

HPF에서는 데이터를 프로세서에 분할하는 방식으로 BLOCK 과 CYCLIC의 두가지 방식을 지원한다. BLOCK 방식은 전체 데이터 객체를 프로세서수로 나눈 크기의 블럭을 각 프로세서에 분할하며, CYCLIC 방식은 데이터 객체의 각 요소를 순서대로 돌아가면서 프로세서에 분할한다. 또한 이들은 분할될 데이터의 크기를 사용자가 지정하여 BLOCK(SIZE) 또는 CYCLIC(SIZE)로 선언할 수도 있는데 이 두방식은 블럭 사이클릭 방식의 데이터 분할을 선언하는 것과 같은 의미를 지닌다. 그림 4는 아래의 분할문에 의한 데이터 분할 형태를 나타낸 것이다.

```
!HPF$ DISTRIBUTE A(CYCLIC),
B(CYCLIC)
!HPF$ DISTRIBUTE (CYCLIC) :: A, B
!HPF$ DISTRIBUTE C(BLOCK, BLOCK)
!HPF$ DISTRIBUTE D(BLOCK,
CYCLIC(2))
!HPF$ PROCESSORS SEDIUM(16)
```

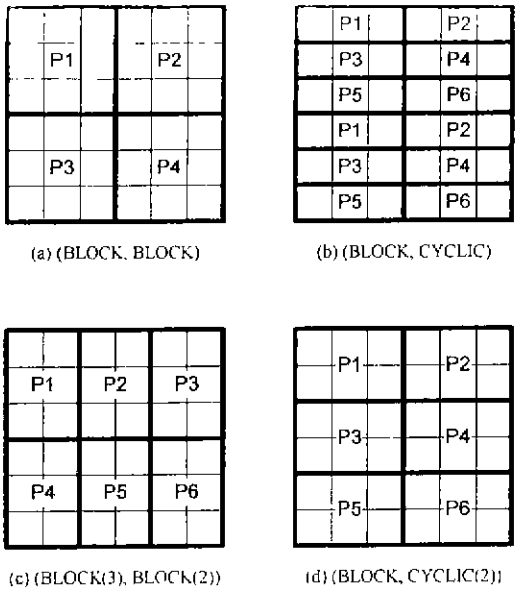

그림 4 데이터 분할에

```
!HPF$ DISTRIBUTE CENTURY
(BLOCK) ONTO SEDIUM
```

데이터를 특정 프로세서에 분할하려면 DISTRIBUTE 지시어를 ONTO절과 함께 사용하면 된다. 만일 ONTO절이 생략되어 있다면 데이터는 컴파일러에 의해 선택된 임의의 프로세서에 분할된다. 마찬가지로 배열 방식을 생략하거나 템플리트를 사용하지 않는 경우는 컴파일러에 의한 임의 지정을 따른다.

그림 4는 데이터가 분할되어지는 형태를 보이는 것으로 이들이 프로세서에 분할되는 원칙은 컴파일러 구현자가 결정할 사항이다. 단, 여기서 중요한 것은 나뉘어진 한 부분의 데이터들은 모두 동일한 프로세서에 분할되어야 한다는 사실이다.

3.2.3. PROCESSORS 지시어

PROCESSORS 지시어는 논리적 프로세서에 대한 이름, 차원, 각 차원의 크기를 정의하여 DISTRIBUTE 지시어에 의해 데이터가 매핑될 격자 구조의 프로세서 배열을 선언한다.

```
!HPF$ PROCESSORS P(N)
```

```
!HPF$ PROCESSORS Q(8, NUMBER_
OF_PROCESSORS()/8)
!HPF$ PROCESSORS BIZARRO
(1972 : 1997, −20 : 17)
!HPF$ PROCESSORS SCALARPROC
```

데이터가 분할될 논리적 프로세서의 배열은 다차원으로 선언될 수 있으며, 사용될 프로세서의 수를 지정하기 위해 HPF 에서 제공하는 NUMBER_OF_PROCESSORS 또는 PROCSSORS_SHAPE과 같이 프로그램 수행에 사용되는 물리적 프로세서의 개수와 형태를 조회하는 내장 함수를 사용할 수 있다. 또한 프로세서 배열에서 차원의 크기는 범위로서 선언할 수도 있으며, 배열의 형태가 선언되지 않은 경우 이는 스칼라 형태의 프로세서를 선언하는 것과 같다.

프로그램 실행 중 두개의 데이터 객체가 어떤 순간에 같은 논리적 프로세서에 매핑되도록 지정되어있는 경우, 이는 두 데이터 객체가 그 순간 같은 물리적 프로세서에 매핑되어 있음을 의미하는 것이다.

3.2.4. DYNAMIC 지시어

REALIGN과 REDISTRIBUTE 지시어는 실행부에 오는 동적 실행문으로, 이를 수행하기 위해서는 앞서 데이터 객체들을 DYNAMIC으로 선언해야 한다.

DYNAMIC 지시어는 데이터 객체가 동적으로 다시 정렬되거나 재분할될 수 있음을 선언한다. 따라서 DYNAMIC로 선언되지 않은 객체에는 REALIGN 이나 REDISTRIBUTE를 적용할 수 없다.

포인터나 ALLOCATABLE 속성을 가지는 변수는 ALIGN 이나 DISTRIBUTE 지시어를 적용할 수 있다. 그러나 그러한 지시어들은 그 즉시 영향을 미치지 않고 ALLOCATE 문에 의해 배열이 할당될 때마다 할당된다.

3.2.5. TEMPLATE 지시어

TEMPLATE 지시어는 데이터들이 정렬되기 위해 사용되는 공동 장소인 템플리트의 이름, 차원, 각 차원의 크기를 지정하여 템플리트를 선언한다. 템플리트는 실제로 할당되는 공간이 아니라 인덱스된 위치에서 배열을 구성하는 논리적 공간으로 생각할 수 있다. 이는 ALIGN 문과 함께 사용되어 분할될 데이터를 논리적으로 정렬할 대상체로써 사용된다.

```
!HPF$ TEMPLATE A(N)
!HPF$ TEMPLATE B(N,N), C(N, 2*N)
```

템플리트는 배열이 선언된 것과 다른 형태로 참조될 때 유용하게 사용되며, 서브프로그램의 인수로써 전달될 수는 없다.

3.3. 병렬 실행문

3.3.1. FORALL

병렬 프로그래밍을 그 목적으로하는 HPF는 데이터 분할 지원이외에 프로그램에서 병렬로 수행될 수 있는 부분을 명시하는 FORALL 구문을 제공하여 사용자에 대한 병렬화를 지원한다. FORALL은 단문의 형태로써 FORALL 문 또는 복수개의 문장을 가지는 FORALL 구문으로 사용될 수 있다. FORALL은 배열의 각 요소들에 대한 지정문을 동시에 실행시키고자 할 때 사용하는데, FORALL 다음에 오는 표현식에는 연산에 필요한 데이터에 대한 시작과 끝 그리고 이용되는 데이터간의 간격을 정의한다.

```
FORALL (I = 1 : N, J = 1 : M) A(I, J)
= I + J
FORALL (I = 1 : M, J = 1 : N : 2) A(I, J)
= I * B(J)
FORALL (i = 2 : 4) x(i) = x(i − 1) + x(i)
+ x(i + 1)
```

FORALL 문에 오는 실행문은 종속성을 포함하지 않아야 한다. 따라서 세번째 표현식의 경우 이를 DO 루프의 순차적 표현으로 바꿀때 다음의 형태로 변환되지는 않는다. FORALL 문을 DO 루프로 변환하는 방법은 [4]에 잘 설명되어 있다.

```
DO I = 2, 4
  x(I) = x(I − 1) + x(I) + x(I)
```

```
ENDDO
```

3.3.2. PURE 속성과 INDEPENDENT 지시어

HPF는 FORALL 문에 배열의 연산 결과를 저장하는 단순한 대입문외에도 프로시듀어의 호출을 허용한다. 단, 그 프로시듀어의 실행 결과가 프로그램의 다른 부분에 영향을 미치지 않아야 한다. 이러한 속성을 '순수(pure)'라하며 PURE 라는 지시어를 통하여 컴파일러에 알린다.

```
REAL PURE FUNCTION f (x, i)
  REAL, INTENT(IN) :: x
  INTEGER, INTENT(IN) :: i
  IF (x>0.0) THEN
    f = x*x
  ELSE IF (i == 1 .OR. I == N) THEN
    f = 0.0
  ELSE
    f = X
    ENDIF
  END FUNCTION
    .....
  REAL a(n)
  INTEGER i
    .....
  FORALL (i = 1 : N) a(i) = f( a(i), i)
```

순수 프로시듀어는 순수 함수와 순수 서브루틴을 포함한다. 순수 함수란 수행된 결과만을 가지고 복귀하는 함수를 말하며, 순수 서브루틴은 인수로 사용된 값이나 포인터를 제외하고는 다른 것에 영향을 미치지 않아야 한다. 순수 프로시듀어에서 호출되는 프로시듀어나 순수 프로시듀어를 참조할 때의 실인수는 반드시 순수해야 한다. '순수'의 속성을 지니는 함수로는 HPF 내장 함수와 HPF 라이브러리들이 있는데 이들에 대해서는 PURE를 선언할 필요가 없다.

INDEPENDENT 지시어는 DO 루프나 FORALL 문에 선행하며 따라오는 실행문들은 프로그램의 의미를 바꾸지 않으면서 하나의 반복문이 또다른 반복문에 대하여 독립적으로 수행이 가능함을 선언한다. 이 정보는 통신을 병렬화시키거나 재구성하는 등의 최적화를 시도할 때 사용될 수 있다.

```
a(p(1 : 100)) = b(1 : 100)

!HPF$ INDEPENDENT, NEW(I2)
DO i1 = 1, n1
  !HPF$ INDEPENDENT, NEW(i3)
DO i2 = 1, n2
  !HPF$ INDEPENDENT, NEW(i4)
  DO i3 = 1, n3
    DO i4 = 1, n4 ! The inner loop is NOT
                  ! independent!
      a(i1, i2, i3) = a(i1, i2, i3) + &
          b(i1, i2, i4) * c(i2, i3, i4)
    END DO
  END DO
  END DO
END DO
```

3.4. 내장 함수와 라이브러리 루틴

HPF는 병렬 알고리즘 설계에서 필요한 기본적 연산을 위하여 Fortran 90에서 지원하던 내장 함수를 포함하여 새로운 내장 함수들을 정의하였다. 이들은 크게 시스템 구성과 관련하여 프로세서의 수를 조회하는 NUMBER_OF_PROCESSOR, 프로세서의 배열을 조회하는 PROCESSORS_SHPAE 등의 시스템 조회 함수와 계산에 사용되는 산술용 내장 함수가 있다.

3.5. 외부 프로시듀어

HPF 프로그램에서는 다른 언어로 쓰여진 프로시듀어를 외부 프로시듀어로서 호출할 수 있다. 외부 프로시듀어 속성은 다른 언어사이로 쓰여진 프로그램과 HPF 프로그램사이에 인터페이스를 정의하며, 호출될 함수를 EXTRINSIC이라 선언하여 컴파일러에 알린다.

3.6. 순서 조정과 기억 장소 조정

FORTRAN 77과 Fortran 90에서는 COMMON 문이나 SEQUENCE 문을 이용하여 기억 장소의 공유나 데이터간의 순서를 표현할

수 있다. 그러나 이러한 기능은 변수들을 다수의 프로세서에 매핑하여 성능을 향상시키려는 HPF의 특성과는 모순된다. 즉 단일의 순차적 메모리에 적합한 데이터의 순서 조정과 기억장소 할당 기능은 분산 메모리에 데이터를 분할하는 것을 우선 목표로 하는 HPF에는 적합지 않다. 따라서 Fortran의 표현에 어느 정도의 제약을 둘 필요가 있으며, HPF에서는 이를 병렬 프로그래밍에 적합하게 변환시킨다.

HPF에서는 COMMON으로 선언된 데이터를 순차적 데이터로 간주할 것인지의 여부를 사용자가 지정할 수 있도록 SEQUENCE와 NO SEQUENCE 지시어를 제공한다. 이외에도 COMMON으로 선언된 데이터에 대한 기억장소 할당을 어떻게 수행할 것인지를 정의하고 있다.

3.7. HPF Subset

High Performance Fortran 제정에 있어 중요한 목표 중 하나는 HPF 컴파일러의 조기 상용화이었다. 이에 High Performance Fortran Forum은 HPF 전체 사양중 데이터 병렬 프로그래밍에 필요한 최소한의 기능은 포함하면서 컴파일러 구현에 어려움이 있는 부분들을 제외한 HPF subset을 정의하였다. HPF subset에 포함되지 않는 사항은 다음과 같다.

- DYNAMIC, REALIGN, REDISTRIBUTE 지시어
- INHERIT 지시어
- PURE 함수 속성
- FORALL 구문
- HPF 라이브러리와 HPF_LIBRARY 모듈
- EXTRINSIC 함수 속성

3.8. Journal of Development

시간적 제약을 두고 제정된 HPF는 제안된 모든 의견들을 포함할 수는 없었다. 이들 중 HPF의 향상을 위해 이후 고려되어야 할 사항들을 모아 Journal of Development라는 출판물을 엮었다. 여기에 포함된 내용들은 다음과 같다.

- 동일한 논리적 프로세서집합을 다른 형태로 볼 수 있도록 하는 VIEW 지시어
- 사용자 정의 데이터 분할
- 데이터 분할 형식의 부분적 정의
- WHERE문의 중첩
- 주어진 문장을 실행할 프로세서를 지정하

표 1 HPF subset의 구현 여부 비교

특 징	Adaptor	PGHPF	xHPF	Vienna
PROCESSORS 지시어	무시	Yes	무시	Yes
TEMPLATE 지시어	Yes	Yes	Yes	No
단순 정렬	Yes	Yes	Yes	Yes
완전 정렬	No	No	No	Yes
FORALL 문	Canonical	Yes	Yes	VF only
Fortran 90	Yes	Yes	Yes	No
분할되는 차원	1 차원	2 차원	다차원	7 차원
분할 방식	BLOCK	BLOCK, CYCLIC	BLOCK, CYCLIC	BLOCK

표 2 HPF subset에 포함되지 않으나 구현된 항목들

특 징	Adaptor	PGHPF	xHPF	Vienna
재분할	No	No	No	Yes
INHERIT 속성	No	No	Yes	Yes
FORALL 구문	No	No	Yes	No
HPF 라이브러리	only scatter	No	Yes	No
외부 함수	가능	No	Yes	Yes
PURE 속성	Yes	No	No	No

는 EXECUTE-ON_HOME 지시어와 지역 데이터 접근을 명시하는 LOCAL_ACCESS 지시어
- 순수 프로시듀어의 요소적 참조[3]
- 병렬 입 출력
- FORALL-ELSEFORALL 구문

4. HPF 컴파일러

현재 HPF를 지원하는 컴파일러는 많은 대학, 연구소 및 산업계에서 경쟁적으로 수행되고 있다. 이들은 HPF를 주도해 온 Rice 대학과 Syracuse 대학의 Fortran D, Vienna 대학의 Vienna Fortran 등이 프로토타입 시스템으로 구현되어 있으며, Applied Parallel Research의 xHPF 및 Portland Group의 PGHPF 등은 상용화 단계에 있다. 이와 더불어 IBM, Fujitsu 및 Intel 등에서 HPF 컴파일러를 구현 중이며, Cray Research와 HP는 아직 구현하고 있지 않으나, HPF에 대하여 지속적인 관심을 보이고 있다. 이들 시스템은 PGHPF를 제외하고는 모두 HPF subset을 지원하고 있다.

xHPF는 HPF subset으로 작성된 프로그램을 메세지 패싱 호출을 갖는 Fortran 77 프로그램으로 변환하는 컴파일러이다. 이 변환된 프로그램은 다양한 시스템에서 SPMD 방식으로 수행되어 진다. xHPF에서 지원하는 시스템은 IBM SPI, Intel Paragon, Meiko, Cray T3D, nCUBE, CM-5 및 PVM, Express 또는 Linda를 이용하여 네트워크로 연결된 워크스테이션 등이 있다. xHPF의 특징은 다차원 배열의 분할을 지원하기 위해, 다차원 배열을 1차원 배열로 확장하는 데 있다.

Syracuse 대학으로부터 HPF 컴파일러 기술을 전수받아 개발된 PGHPF도 HPF subset으로 작성된 프로그램을 메세지 패싱 호출을 갖는 Fortran 77 프로그램으로 변환하는 컴파일러이다. PGHPF에서 지원하는 메세지 패싱 라이브러리는 PVM, MPI, Parmacs, Nx 및 Solaris thread 등이 있으며, Intel Paragon, Meiko CS-2 및 공유 메모리 SPARC 워크스테이션 등을 지원한다. 또한 PGHPF는 90년대 중반에 HPF 문서 사양 1.0에서 정의한 fullset을 지원하는 컴파일러를 발표할 예정이다.

xHPF와 Fortran D의 특징 중 가장 큰 차이는 xHPF는 병렬 실행이 가능한 부분을 찾아 이들을 병렬화시키려 하는데 반하여, Fortran D는 병렬 구문에 표현되어 있는 부분에 대해서만 컴파일러가 처리한다는 것이다. 다음은 Syracuse 대학에서 개발한 Fortran D 컴파일러의 구성단계이며, 단계별 수행 기능은 다음과 같다.

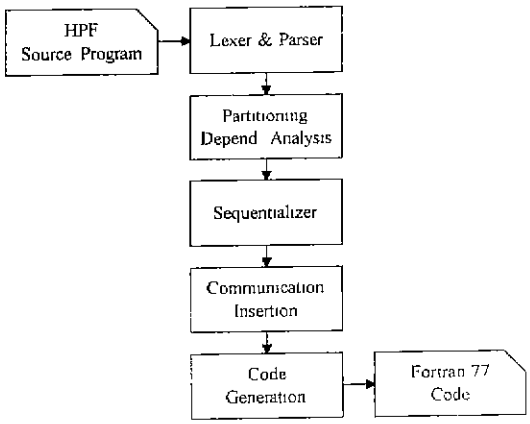

그림 5 Fortran D 컴파일러의 구성

Fortran D 컴파일러는 그림 5와 같이 구문분석, 종속성 분석 및 데이터 분할, 순차화, 통신 라이브러리 삽입 그리고 코드 생성의 다섯 단계로 이루어진다. Fortran D 컴파일러는 HPF로 작성된 원시 프로그램을 입력으로 받아 문법을 점검하고 파스 트리를 생성한다. 다음으로 프로그램내 존재하는 종속성을 분석하고 병렬 지시어와 구문을 참조하여 데이터를 분할한다. 순차화 단계에서는 분할된 데이터에 따라 각 노드에서 수행할 순차적 프로그램을 생성하면, 다음 단계에서 이들 노드 프로그램간에 수행될 통신에 대한 라이브러리들을 삽입하여 병렬 컴퓨터에서 수행할 SPMD 형태의 Fortran 77

3 스칼라 인수에 대해 정의되어 있으나 적합한 형식의 배열 인수에도 적용할 수 있는 프로시듀어를 "요소적 프로시듀어(Elemental Procedure)"라 하고 배열 인수가 요소적 프로시듀어를 참조하는 것을 요소적 참조라 한다.

코드를 생성한다.

5. 병렬 프로그래밍언어의 향후 연구 방향

High Performance Fortran Forum은 HPF 버전 2.0을 1996년까지 제정 발표하기 위해 그들의 모임을 재결성하였다. 이 모임의 목적은 HPF 언어 사양 1.0에서 명확하지 않은 부분에 대한 해석을 강화하고 미흡한 내용을 보완하기 위함이다. 이 모임은 Journal of Development에 남겨둔 사항들을 기반으로 하고 있으며 HPF 2.0에 보완될 기능들은 다음과 같다 [6].

- 매핑기능의 향상
  - 사용자 지정의 비정규 데이터 매핑
  - 링크 구조의 매핑(동적 매핑 지원)
  - 프로세서 부분에 대한 매핑
  - 프로세서 배열에 다중 뷰(VIEW) 허용
- 연산 제어와 태스크 병렬화
  - 연산을 프로세서에 매핑
  - INDEPENDENT DO 루프
  - 동기 조건이 만족될 때 루프를 실행하는 DOACROSS 문
  - 일부 프로세서에서 프로세스를 시작 종료하고 통신할 수 있는 태스크 병렬화
- 입 출력
  - 제한된 형식의 화일 접근을 통한 빠른 병렬 입 출력 제공
  - 요구 페이징을 이용한 큰 배열의 접근
  - 적절한 체크포인트와 재시동
  - 비동기 입 출력과 prefetching 허용
- 통신 최적화
- 언어 프로세서 환경
  - HPF와 다른 언어간의 인터페이스 제공
  - 디버거, 프로파일러 등의 지원 도구를 사용할 수 있는 표준 인터페이스

HPF는 데이터 병렬 프로그래밍 언어로서 병렬 컴퓨팅의 성능을 효율적으로 활용하는데 큰 역할을 하고 있으나 그 한계를 가지고 있다. 예를 들어 "대기 분수계 모델(airshed model)"과 같이 다양한 특성들이 포함된 복잡한 문제들은 데이터의 병렬화만으로는 충분한 성능을 기대할 수 없으며[2], 실제로 많은 경우에 우리가 해결해야 할 문제들이 이와 같은 양상을 띤다.

이와같은 문제 해결에 있어 성능 향상을 위해 요구되는 것이 태스크의 병렬화이다. HPF를 확장하여 다수의 태스크 쓰레드와 다중 제어를 허용하여 태스크 병렬화를 시도하는 연구가 Carnegie Mellon 대학의 Fx와 Argonne National Lab과 Rice 대학의 Fortran M이다 [7].

이들 태스크 병렬 프로그래밍의 이점은 이기종 컴퓨팅을 허용한다는 것이다. 태스크 병렬 프로그래밍은 각 프로세서에서 다른 데이터를 이용하여 다른 연산을 수행하도록 한다. 따라서, 태스크 병렬화를 이용하여 각 연산의 특성에 적합한 프로세서에 태스크를 분할하므로 병렬 시스템의 장점을 최대한 활용할 수 있다.

6. 결 론

병렬 컴퓨터는 경제성, 가격대 성능비, 확장성, I/O 및 메모리 접속 대역폭 등의 장점에도 불구하고 고성능 컴퓨팅을 연구하는 과학자들 위주로 제한적으로 사용되어 왔다. 이는 사용자가 해결해야할 과제 또는 대상에 대한 해를 구하기 위하여 병렬화 기법에 대한 연구 시간을 많이 투자해야하므로, 프로그램 작성에 많은 시간이 걸리고 또한 효율적인 병렬 프로그램을 구현하기 어렵기 때문이다. 실제로 병렬 프로그래밍은 프로세서간의 데이터 통신 및 데이터 분할 방법, 지역적 주소 공간등을 고려해야하기 때문에 기존의 순차 프로그래밍에 비하여 프로그램 작성이 매우 어렵다. 따라서 지난 10여년간 병렬 컴퓨터 개발자뿐만 아니라 병렬 컴퓨터를 이용한 응용 프로그램을 작성하는 과학자 사이에서도 병렬 프로그램을 쉽게 할 수 있고 시스템 독립적인 언어를 개발하는데 관심을 모아왔다. HPF 언어는 이러한 노력의 일환으로 진행되고 있으며, 1997년에 국제 표준 언어를 재정하기 위해 주요 병렬 컴퓨터 개발 업체, 학계, 연구기관이 협의를 진행중이다. 본 고에서는 시스템 독립적으로 사용할 수 있는 데이터 병렬 언어인 HPF의 기능과 HPF 컴파일러에 관하여 살펴 보았다.

HPF 언어의 표준화는 시스템 개발자 측면에서는 병렬 컴퓨터 사용자 수의 증대를 가져올

것이며, 사용자 측면에서는 쉽게 병렬 컴퓨터에 접근할 수 있는 계기가 될 것이다. HPF의 성공 여부는 사용자에게 얼마나 쉽고 성능이 뛰어난 HPF 컴파일러를 제공하느냐에 달려있다. HPF 컴파일러는 HPF로 작성된 프로그램으로부터 사용자가 정의한 병렬 지시어를 검색하고 병렬성과 순차성을 고려하여 데이터 및 연산을 프로세서에 분할하고 연산에 필요한 데이터의 프로세서간 교환 기능을 삽입하여 각 시스템의 특성에 맞는 실행 코드를 생성해야 하기때문에 기존의 Fortran 77 또는 Fortran 90 컴파일러 개발에 비하여 어렵다.

현재, Rice 대학, Syracuse 대학, 그리고 Vienna 대학 등에서의 연구 개발을 기반으로 상용 HPF subset 컴파일러가 개발되고 있으며, 국내에서도 시스템 공학 연구소, 충남 대학교 등에서 이와 유사한 연구가 진행중이다.

참고문헌

[1] Philip J. Hatcher, "The Impact of High Performance Fortran", IEEE Parallel & Distributed Technology, Vol. 2, No. 3, pp. 13-14, 1994,

[2] Thomas Gross, David R. O'Hallaron, and Jaspal Subhlok, "Task Parallelism in a High Performance Fortran Framework", IEEE Parallel & Distributed Technology, Vol. 2, No. 3, pp. 13-14, 1994.

[3] David B. Loveman, "High Performance Fortran", Proceedings of the High Performance Computing Conference '94", September 1994, Singapore.

[4] High Performance Fortran Forum, "High Performance Fortran Language Specification Version 1.0", May 1993.

[5] High Performance Fortan Forum, "High Performance Journal of Development", CRPC-TR93300, May 1993.

[6] High Performance Fortran Forum, "HPF-2 Scope of Activities and Motivating Applications Version 0.8", November 1994.

[7] John M. Levesque, "Applied Parallel Research's xHPF System", IEEE Parallel & Distributed Technology, Vol. 2, No. 3, 1994.

[8] Ian Foster, "Task Parallelism and High-Performance Languages", IEEE Parallel & Distributed Technology, Vol. 2, No. 3, pp. 16-27, 1994.

[9] Vincent J. Schuster, "PGHPF from The Portland Group", IEEE Parallel & Distributed Technology, Vol. 2, No. 3, 1994.

[10] ZeriBozkus, Alok Choudhary, Geoffrey Fox, Tomasz Haupt, and Sanjay Ranka, "Impact of Compilation Technology on Computer Architecture", Kluwer Academic Pubhshers.

[11] Zeki Bozkus, Alok Choudhary, Geoffrey Fox, Tomasz Haupt, and Sanjay Ranka, "Compiling Distribution Directives in a Fortran 90D Compiler," Technical Report SCCS-388, Northeast Parallel Architectures Center.

[12] 문경덕, 김태근, 반난주, 김중권, 유여백, PVM을 이용한 HPF 병렬 프로그래밍 번역기 구현에 관한 연구, 병렬처리시스템 학술발표회 논문집, 제 6권 제 1호, pp. 17-28, 1995년 5월.

문 경 덕

1990 한양대학교 전자계산학과 학사
1992 한양대학교 전자계산학과 석사
1992~현재 시스템공학연구소 연구원
관심분야 : 병렬 컴파일러, 이기종 컴퓨팅, 병렬 처리

반 난 주

1988 아주대학교 전자계산학과 학사
1989~1990 금성 소프트웨어
1994 미주리 주립대학교 전자계산학과 석사
1994~현재 시스템공학연구소 연구원
관심분야 : 병렬 컴파일러, 이기종 컴퓨팅, 병렬 처리

김 태 근

1987 한양대학교 수학과 학사
1990 뉴욕 주립대학교 응용 수학과 석사
1993 뉴욕 주립대학교 응용수학과 박사 병렬처리 알고리즘 전공
1992~현재 시스템공학연구소 선임연구원
관심분야 . 알고리즘, 병렬 컴파일러, 이기종 컴퓨팅, 병렬 처리

김 중 권

1973 서강대학교 전자공학과 학사
1976~1981 KIST 전산센터
1981~1982 대통령 경호실
1985 연세대학교 전자계산학과 석사
1990~현재 아주대학교 전자계산학과 박사과정
1982~현재 시스템공학연구소 책임연구원
관심분야 : 병렬 컴파일러, 이기종 컴퓨팅, 병렬 처리

유 여 백

1970 서울 문리대 학사(천문학)
1972~1974 삼성생명 전산실 Programmer
1974~1977 KIST 전산실 Systems Analyst
1979 워싱턴 주립대 전산학과 석사
1983 워싱턴 주립대 전산학과 박사
1983~1985 뉴욕주립대 전산학과 조교수
1986~1994 몬타나 주립대 전산학과 부교수
1994~현재 시스템공학연구소 슈퍼컴퓨터센터장
관심분야 : 알고리즘, O. R., 병렬 처리

● 제22회 정기총회 및 추계학술발표회 ●

- 행사일정 : 1995년 10월 27일(금)~28일(토)
- 행사장소 : 인하대학교
- 발표논문 접수마감 : 1995년 8월 12일(土)
- 박사학위 논문 발표마감 : 1995년 8월 31일(木)
- 문의 및 접수처 : 한국정보과학회 사무국

Tel : 02-588-9246/7, Fax : 02-521-1352
서울시 서초구 방배3동 984-1(머리재빌딩) ㉾137-063