論文96-33B-6-4

트랜스퓨터 통합환경과 그 응용

(Transputer Integrated Environment and Its Application)

李 孝 鍾 *, 任 訓 徹 **

(Lee Hyo Jong and Hun-Cheal Im)

요 약

트랜스퓨터는 마이크로 프로세서의 일종으로, 일반 논리, 정수 및 실수의 연산 등은 물론이고 프로세서간에 정보교환을 할 수 있는 통신링크 및 그의 제어장치가 하드웨이로 구성되어 있다. 이를 이용해서 프로세서간의 통신을 간단히 할 수 있으며, 저렴한 가격으로 강력한 병렬처리 컴퓨터를 구현할 수 있어서 경제성이 높다. 그러나, 생소한 프로그래밍 언어와 사용자 인터페이스 때문에 트랜스퓨터의 사용은 극히 제한되어 왔다. 본 논문에서는 이러한 트랜스퓨터를 보다 쉽게 이용할 수 있는 도구를 구현하고자 한다. 여러 가지 개발 도구를 한 개의 통합환경으로 구현한 트랜스퓨터 통합환경과 트랜스퓨터를 감시하고 관리할 수 있는 트랜스퓨터 관리기를 개발하였고 이를 이용해서 실제 트랜스퓨터의 성능을 시험할 수 있는 그래픽스 응용프로그램을 개발하였다.

Abstract

A transputer is a powerful micro-processor, which can communicate with other processors with its own communication links and can be utilized into a powerful parallel computer system with easy way and low price. However, its usage has been limited to a few people because of its own programming language and complicated user interfaces. This paper presents programming tools to use a transputer system easily. It includes Transputer Integrated Environment which has many developing tools, and Transputer Manager which can monitor and manage the system and developed programs on it. We also implemented a graphics application program in order to test the feasibility and functionalities of these tools. The Transputer Integrated Environment and Transputer Manager allow parallel programmers easy access to the transputer networks and visualize the performance status of each processor so that they can write efficient parallel programming codes.

I. 서 론

단일 프로세서에만 의존하지 않고 다수의 프로세서를 이용하고자 하는 병렬처리 컴퓨터에 대한 다양한 연구가 소프트웨어와 하드웨어 분야에서 오랫동안 연구되어 왔다. 본 연구는 트랜스퓨터[1,2,3] 시스템 환경에서 이용될 수 있는 병렬처리 프로그램 개발 도구의 구현과 이의 응용에 초점을 두었다. 트랜스퓨터는 독특한 개념의 마이크로 프로세서로서 간단하고 저렴한 가격으로 강력한 병렬처리 컴퓨터를 구현할 수 있다. 프로세서 자체 내에 4개의 통신링크를 내장하고 있으며 이것을 이용해서 많은 수의 프로세서를 연결하면 별도의 하드웨어 없이 병렬시스템을 구현할 수 있다. 또한 전용의 스케줄러와 하드웨어적인 태스크 전환장치를 내장하고 있어 프로그래머가 어셈블리어나 고급언어 레벨에서 직접 멀티태스킹을 구현할 수 있게 해준다. 그러나 이러한 트랜스퓨터의 장점에도 불구하고 트랜스퓨터 시스템의 사용은 극히 제한되어 있으며 잘 알려져 있지 않다.

트랜스퓨터의 사용이 장려되지 않은 이유 중의 하나

* 正會員, 全北大學校 電子工學科
(Chonbuk National University, Department of Electronic Engineering)
** 正會員, 國防科學研究所
(ADD)

接受日字:1995年9月29日, 수정완료일:1996年5月27日

는 순차적인 프로그래밍에 익숙한 프로그래머들이 다중 프로세서의 장점을 살리는 대신, 병렬처리 알고리듬을 구현해야 되는 다중 프로세서 시스템의 사용에 먼저 당황하게 되는 것이다. 또 다른 이유는 순차적인 프로그래밍 환경의 경우 이미 오랜 기간에 걸쳐 많은 프로그램의 개발, 오류정정 및 관리 등을 편리하게 할 수 있는 도구들이 많이 나와 있는 반면, 병렬 프로그래밍 환경에서는 상대적으로 이러한 개발 도구들이 빈약하다고 언급할 수 있다.

현재까지 나와 있는 트랜스퓨터 개발도구로써는 Inmos사에서 개발한 occam 도구로 트랜스퓨터 컴파일러류, 디버거, 메모리 관리가 및 make 파일 생성기 등이 포함되어 있다. 폴딩에디터라는 문서작성기식의 도구도 트랜스퓨터 프로그램 개발을 도와주는데, occam 언어나 병렬 C 언어에서 사용하는 키워드를 기억하고 있어서, 프로그래머들에게 병렬언어의 프로그래밍을 도와주고, 코드의 자연스런 블록구조를 만들어서 프로그램의 이해를 돕도록 하였다. Inmos사 이외에도 트랜스퓨터 컴파일러를 제작한 Tartan, 3LC 또는 Logical System사[8,9] 등에서도 각 제품의 편의를 위하여 개발도구기능을 첨가했으나, 이들 제품의 모든 도구들은 독립적으로 호스트 컴퓨터의 운영체제 내부에서 사용되도록 개발되었다.

이러한 문제들을 해결하기 위해서 본 연구에서는 트랜스퓨터 환경에서 사용할 수 있는 병렬처리 프로그램의 개발 및 관리 등에 필요한 도구들을 개발하였으며, 이들의 유용성을 3가지로 나누어 설명할 수 있다. 첫째는 병렬프로그램을 개발하는데 필요한 여러 가지 도구들을 한 개의 환경으로 통합시켜 놓은 트랜스퓨터 통합환경이다. 프로그래머는 많은 도구들을 통합환경 내에서 쉽게 사용함으로서 전체적인 프로그램의 개발시간을 단축시킬 수 있다. 둘째는 통합환경을 이용하여 성공적으로 작성한 프로그램을 효율적으로 개선하고 관리하는데 도움을 줄 수 있는 트랜스퓨터 관리기이다. 이 관리기를 사용함으로서 트랜스퓨터의 계산결과를 호스트의 그래픽으로 나타낼 수 있으며 프로그램 실행도중 병목현상이나 부하 불균형 문제 등을 파악하여 효율적으로 병렬자원을 이용할 수 있게 도와준다. 셋째는 임의의 구조로 구성된 트랜스퓨터 망의 구조를 파악할 수 있는 툴을 구현하고, 통합환경 내부에 탑재시켜 프로세서 망의 구조와 병렬 알고리듬을 직접 비교할 수 있도록 하였다. 또한 이들 병렬프로그래밍 도구의 성능을 시험하기 위하여 컴퓨터 그래픽스의 한 분야인 광추적[4,5] 알고리듬을 트랜스퓨터 시스템에서 사용할 수 있도록 병렬프로그램 하여 다양한 입력 영상으로 트랜스퓨터 상에서 트랜스퓨터 관리기의 효율성을 실험 분석하였다.

Ⅱ. 트랜스퓨터의 소개

트랜스퓨터는 Inmos사에서 개발한 메시지 교환 방식[6]의 병렬처리 컴퓨터용 마이크로 프로세서로서 단일프로세서 안에 병렬처리 컴퓨터를 구현하기에 필요한 여러 가지 기능들을 내장하고 있다[1,2,3]. 가장 큰 특징으로서는 자체에 외부 프로세서와 통신할 수 있는 양방향 통신링크를 가지고 있어 하드웨어적으로 두 개의 통신링크만 연결함으로서 인접한 두 프로세서간의 데이터를 교환할 수 있다. 이외에도 하드웨어적으로 작업 전환을 지원하는 작업 전환기, 마이크로 프로그램된 스케줄러, 내부 메모리, 실수연산 프로세서 등의 특징이 있다. 위의 모든 특징들이 한 개의 프로세서 안에 내장됨으로서 여러 개의 트랜스퓨터를 연결하여 하나의 커다란 병렬처리 컴퓨터를 구현하기가 쉽다는 장점이 있다. 그림 1에는 트랜스퓨터를 구성하는 주요 하드웨어와 그 연결이 도식되어 있다.

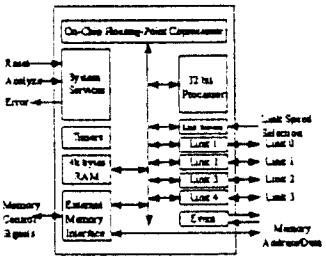


그림 1. 트랜스퓨터의 구조
Fig. 1. Architecture of a transputer.

본 연구에서는 PC상에 카드처럼 끼울 수 있는 트랜스퓨터 보드에 2M 또는 4M바이트의 메모리를 장착한 트랜스퓨터를 사용하고 있다. 이들은 고리, 그물, 트리 및 큐브 구조 등 해결하고자 하는 문제에 가장 적합한 구조를 선택하여 쉽게 상호간에 연결 구성될 수 있다. 트랜스퓨터에서 사용되는 언어로서는 Occam, 병렬-C, 병렬-Pascal, 병렬-Fortran등이 있다. 본 연구에서는 병렬-C를 주 언어로 사용하였으며, 소프트웨어 도구로서는 PC의 MS-DOS 환경에서 구동하는 Logical

Systems 사에서[8,9] 제공하는 전처리기, 컴파일러, 어셈블러, 링커 및 로더 등의 도구를 사용하였다. 병렬-C 프로그래밍 언어는 표준 C프로그래밍 언어가 제공하는 함수 이외에도 트랜스퓨터의 기능을 활용할 수 있는 특별한 라이브러리가 포함되어 있다. 이 라이브러리 함수를 이용하여 프로세서의 생성, 프로세서간의 통신, 프로세서 동기와 및 프로세서 전환 등을 간단히 함수를 사용하여 구현할 수 있다.

Ⅲ. 트랜스퓨터 통합환경 (TIE: Transputer Integrated Environment)

순차프로그램에 익숙한 프로그래머들에게 병렬프로그래밍은 종종 커다란 부담이 되고 있다. 더군다나 트랜스퓨터는 그 독특한 병렬성으로 인하여 프로그래밍이 용이하지 않다. 일반적인 프로그램의 개발 과정을 살펴보면, 프로그래머는 문제해결을 위해 알고리듬이나 방법론을 체계화시킨 다음, 문서작성기를 이용해서 원시코드를 작성하고 컴파일러를 이용해서 작성한 원시코드를 컴파일하여 지정 언어의 구문에 맞게 되었는지를 점검한다. 컴파일 후 컴파일 오류가 발생되면 다시 문서작성기를 이용해서 오류를 수정하게 된다. 비록 성공적으로 컴파일 되었다고 하더라도 링커 오류, 실행오류 등으로 자주 문서작성기를 이용해서 오류를 수정하는 과정을 반복하게 된다. 그림 2(a)에 프로그램을 개발하는 도중에 모든 도구들을 순차적으로 실행시키는 예가 도식되어 있다. 이러한 방식의 프로그래밍 개발과정은 문서작성기, 컴파일러, 링커, 로더 등의 프로그램을 각각 개별적으로 번갈아 가면서 실행시켜야 하므로 불필요한 실행시간과 많은 컴퓨터 자원을 낭비하게 된다. 더욱이 트랜스퓨터의 프로그램은 호스트 컴퓨터를 통하여서만 개발이 가능하므로 이종컴파일러(cross compiler)에 의존할 수밖에 없으며, 실행오류를 호스트 컴퓨터에서 찾아내는 것은 불가능하다.

이 방식의 비효율성을 개선할 수 있는 방법으로서 프로그램 개발시 필요한 모든 프로그래밍 도구들을 한 개의 통합환경이 관리할 수 있도록 하는 방식은 Borland사의 컴파일러에서처럼 순차프로그램의 개발에서도 고려되었다. 이러한 통합환경은 병렬프로그램의 개발에서 프로그래머가 개별적인 개발 도구의 사용에 익숙하지 않더라도 시스템 프로그램들을 쉽게 다룰 수 있도록 도와 줄 수 있다. 그림 2(b)에 제안된 방식으로 프로그램 개발 도구를 통합환경이 관리하여 사용하는 모습을 나타내고 있다.

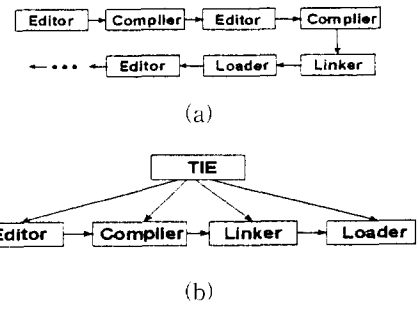


그림 2. 프로그램 개발과정
(a) 일반도구의 사용 (b) 통합환경의 사용
Fig. 2. Process of program development.
(a) Usage of general tools sequentially
(b) Usage of integrated environment

앞서 언급한 것처럼 프로그램의 개발시에 필요한 여러 가지의 도구들을 한 개의 통합환경 안에서 이용하고자 개발한 도구가 트랜스퓨터 통합환경(TIE)이다. 그림 3은 구현된 TIE의 블록 다이어그램을 나타내고 있으며 이 블록들에는 여러 개의 파일을 작성할 수 있는 문서작성기, 컴파일러와 링커 및 로더, 에러메시지 관리기, 프로젝트 관리기가 있다

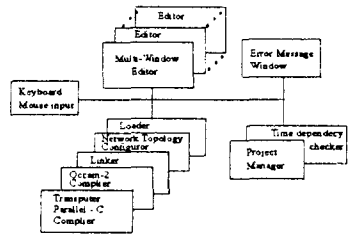


그림 3. TIE의 블록 다이어그램
Fig. 3. Block diagram of TIE.

TIE는 외형상 각각의 객체로 정의된 다수의 창으로 구성되었으며 이들의 화면 구성방법, 정보교환방법 그리고 컴파일 방법을 기술하면 다음과 같다.

· TIE의 화면구성 : TIE의 화면구성은 그림 4에서와 같이 크게 3개로 구성되며 배경, 화면상단의 메뉴 그리고 배경 안에 있는 여러 개의 창들로 이루어진다. 여기에서 창이라는 것은 정사각형의 독립적인 화면이 한 부분으로 나타나며 움직이거나 크기가 변할 수 있는 특징을 가지고 있다. 이러한 기본적인 창의 특성을 계승하여 사용자가 필요한 창들을 만들어 내는데 이것

들에는 문서작성기 창, 에러메시지 창, 프로젝트 창 등이 있다. 창의 구현은 객체지향기법을 사용하였기 때문에 쉽게 다른 특성을 가지는 창을 구현할 수 있다. 이들의 창은 각각 독립적인 객체로서 작용하며 직접적으로 다른 창에는 영향을 미치지 않는 구조로 되어있다. 예를 들면 다수의 문서작성기는 서로간에 별도로 취급하여 문서를 작성한다. 그리고 이들의 창은 한 개의 활성창과 나머지의 비활성창으로 구분되는데 활성창만이 사용자와 직접적으로 인터페이스를 할 수 있으며, 비활성창은 그 창 내부에 마우스를 위치시키고 버튼을 누름으로서 활성창으로 만들 수 있다.

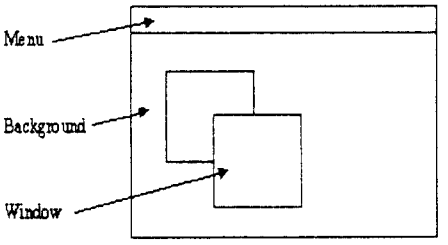


그림 4. TIE의 화면 구성
Fig. 4. Screen configuration of TIE.

· 정보교환 방법 : 프로그래머는 각각의 창과 정보를 교환해야 할 필요가 있는데 정보교환의 방법으로는 메시지 전달 방식을 이용한다. 예를 들면 문서작성기가 원시코드를 작성하고 있을 때 문서작성기의 창 외부에서 파일의 이름을 알고자 할 때 프로그래머는 모든 창으로 메시지를 전달하고 메시지를 받은 창은 메시지를 해석하여 자신에게 해당되는 메시지라는 것을 인식하고 파일의 이름을 메시지 내부에 저장함으로서 정보를 교환한다.

· 컴파일 과정 : TIE에서 가장 많이 쓰는 도구는 원시코드를 컴파일 시키는 것인데 이 방법은 다음과 같다. 먼저 컴파일 명령으로 정의된 키를 사용자가 누르면 TIE는 현재의 활성창에서 작성하고 있는 파일의 이름을 요구한다. 이때 현재의 활성창이 문서작성기이고 파일을 검사하여 컴파일 가능한 파일로 간주되면 컴파일 과정을 호출한다. 병렬 C에서 컴파일 과정은 전처리, 컴파일, 어셈블, 링크의 4단계로 이루어지며 미리 설정된 옵션에 맞게 명령인자를 만들어서 순서대로 4개의 프로그램을 호출한다. 이 과정이 끝난 후 중간의 임시파일은 지우게 되며 컴파일 결과를 별도로 저장하여 에러메시지 창으로 보여주고 프로그래머는 에러메시지 창과 작성하고 있는 원시코드를 대조하면서 컴파일 에러를 정정한다.

TIE는 소프트웨어 공학분야에서 고려되는 코드의 재사용이나 프로그램 개발의 합리성에 중점을 두고 구성되었다. 이들 각 부분의 주요 기능은 다음과 같다.

1) 문서작성기 : 원시코드를 작성하거나 일반 문자형 파일을 작성할 수 있는 범용 문서작성기 구현되었다. 새로운 파일을 생성시키거나 기존에 있던 파일을 읽어들일 때 각 파일은 한 개의 창과 함께 통합환경 상에 나타나게 되며 파일이 변조될 때마다 전체 파일 저장, 부분 선택, 부분 복사, 부분 저장, 문자열 찾기, 문자열 바꾸기 등의 기능을 수행할 수 있다.

2) 코드생성기 : 트랜스퓨터 통합환경 내에서 전처리기, 컴파일러, 링커 및 로더 등의 시스템 프로그램과 대화형으로 접속하여 트랜스퓨터 기계어 코드 및 실행파일을 생성한다. 프로그래머와 시스템의 요구에 맞게 복잡한 시스템 프로그램의 각종 옵션을 상세히 정할 수 있으며 이러한 옵션을 이용하여 문서작성기로 작성한 원시코드를 시스템의 상태에 맞게 최적으로 컴파일 및 링크를 시킬 수 있다.

3) 파일 생성시간 관리 : 파일 생성시간 관리기는 작성한 원시코드를 컴파일하고 링크 시킬 경우 원시코드와 기계어코드, 실행파일의 생성시간을 각각 비교하여 컴파일 할 원시코드를 파악해서 필요한 원시코드만을 컴파일 한다.

4) 에러메시지 관리 : 원시코드를 컴파일 및 링크한 후에 컴파일 오류가 발생하였을 때, 에러메세지를 별도의 파일에 저장하도록 한다.

5) 프로젝트 관리 : 여러 개의 원시코드를 한 개의 프로그램으로 컴파일 시킬 경우 통합환경에서는 프로젝트 파일을 이용하여 관련된 원시코드의 이름만을 포함시켜 프로젝트로 관리할 수 있다.

6) 실행 파일의 실행 : 성공적으로 컴파일되고 링크된 실행파일은 로더를 이용해서 트랜스퓨터로 로드된 후에 호스트 컴퓨터와 접속되어 실행될 수 있다. 이때는 시스템에 관련된 각종 옵션을 미리 정해서 시스템의 상태에 최적으로 로드시킬 수 있다.

그림 5에는 한 개의 프로젝트 파일을 열어놓고 프로젝트에 관계된 파일을 문서작성기를 이용하여 원시코드를 작성하고 이를 컴파일 하는 장면을 보여주고 있다.

TIE는 PC 환경에서 트랜스퓨터를 사용하는 프로그

래머들에게 도움을 주기 위한 도구로서 이를 이용하여 프로그램을 개발할 경우에 개발시간이 현저히 단축되며 주요 기능은 다음과 같다.

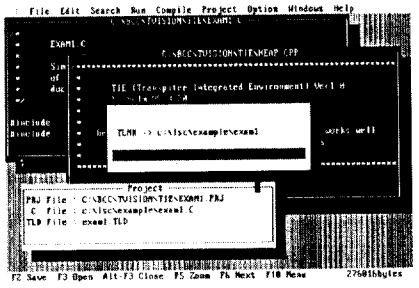


그림 5. TIE의 실행 장면
Fig. 5. Execution process of TIE.

· 병렬프로그램에 익숙치 않은 프로그래머들은 이 분야에서 사용하는 컴파일러의 사용방법에 익숙해져 있지 않다. 또한 옵션의 기능이 복잡하여 단순하게 컴파일만을 시킬 경우에도 많은 시간동안 사용법을 익혀야 하는데, TIE는 미리 정의된 키만을 누름으로서 간편하게 컴파일 시킬 수 있는 방법을 제공한다.

· PC를 트랜스퓨터 프로그램을 개발하는 호스트컴퓨터로 사용하기 때문에 DOS가 지니는 640KB의 메모리 제약을 가지고 있어서, 원시 코드량이 큰 경우 트랜스퓨터로 전송하여 컴파일을 할 수 있도록 하였다. 따라서 32비트 프로세서인 트랜스퓨터가 장착하고 있는 최대메모리까지 사용할 수 있다.

· TIE에서 제공하는 다중창 문서작성기는 2개 이상의 문서를 한 개의 화면에서 작성 및 수정할 수 있다. 병렬프로그램의 경우 두 프로세스간에 통신을 하는 등 두 개이상의 프로세스가 동기를 맞추면서 실행하는 경우가 많은데 이러한 부분을 프로그램할 때 다중창 문서작성기를 효율적으로 사용할 수 있다. 또한 다수의 함수를 가진 프로그램을 작성할 때 다중창 문서작성기를 이용하여 함수부분과 그 함수를 부르는 부분을 한 눈에 보면서 프로그램을 한다.

· 컴파일 후 에러를 별도의 창으로 출력함으로서 현재 원시코드를 작성하고 있는 문서작성기와 대조하면서 에러가 발생한 부분을 빠른 시간에 찾을 수 있다.

· 여러 개의 모듈을 작성하여 컴파일 시킬 때 프로젝트로 등록시킴으로서 한 번에 전체의 컴파일 과정을 수행시킬 수 있고, 또한 고쳐진 모듈만을 재 컴파일하는 방식을 택하기 때문에 컴파일 시간을 단축시킨다.

· 다양한 옵션과 창들을 사용자의 기호에 맞게 배치시키고 사용할 경우 현 상태를 저장시킬 수 있는데 많은 환경을 별도의 이름으로 저장하여 다음에 복귀시킴으로서 TIE를 실행시킬 때마다 다시 환경이나 옵션을 지정할 필요가 없다.

TIE를 이용하여 병렬프로그램을 개발할 경우, 편리성이나 기능성에서 여러 장점이 존재하지만, 몇 가지의 부정적인 요소도 내재하고 있다.

· 크기가 큰 원시코드를 트랜스퓨터 상에서 컴파일 시킬 경우 트랜스퓨터와 PC간의 통신량이 많아져서 일반적으로 컴파일 시간이 매우 느리게 된다. TIE를 사용할 경우 TIE가 DOS의 메모리에 상주하므로 TIE를 사용하지 않을 때 보다 더 작은 크기의 원시코드만을 DOS내에서 직접 컴파일 시킬 수 있고, 그래서 원시코드의 크기가 클 경우 컴파일 시간이 늘어나게 된다.

· TIE는 PC에서 사용할 목적으로 구현되었기 때문에 많은 입출력 부분이나 그래픽 처리 부분이 PC 기반하에서 작성되었다. 따라서 UNIX나 OS/2등 다른 운영체제에서는 직접 사용할 수 없는 호환성 문제가 있다.

완성된 병렬프로그램의 결과적인 측면에서 볼 때 TIE 환경에서 프로그램을 작성하는 것은 사용하지 않았을 때와 원시코드면에서 근본적인 차이는 없다. 그러나 TIE는 효율적으로 원시코드를 작성하는 방식을 제공하기 때문에 이를 이용하면 단위시간내에 더 많은 코드를 작성할 수 있으며 신뢰성 있는 코드를 만들 수 있다. 또한 TIE를 사용하는 환경에 익숙해짐으로서 일단 작성된 프로그램을 수정하거나 개선시키는 등 유지보수면에서 볼 때 TIE를 사용하지 않는 것에 비해 효율적임이 VI절에서 기술한 병렬광추적 알고리듬의 구현을 통하여 밝혀졌다.

Ⅳ. 트랜스퓨터 관리기 (TMAN: Transputer Manager)

트랜스퓨터 관리기는 다중프로세서에서 복수의 프로세서들이 실제로 실행되는 상황들은 문자 및 그래픽으로 가시화 시켜주는 것이 주요목표이다. 아래에 트랜스퓨터를 사용하는 환경에서의 그래픽 사용자 환경의 필요성과 프로세서 감시기의 필요성에 대해 기술하였다.

1) 그래픽 사용자 환경의 필요성

트랜스퓨터 시스템은 자체 내에 실수연산장치를 내

장하고 있으며 다수의 프로세서를 사용하여 병렬처리로 복잡하고 장시간이 소요되는 문제를 빠른 시간에 처리할 수 있다. 특히 그래픽스 응용프로그램은 한 개의 영상을 여러 개로 분할하고 분할된 영상은 직접적인 관련성이 없는 경우가 많으므로 병렬처리를 하기에 적합한 분야라고 할 수 있다. 이러한 그래픽스 응용프로그램의 결과는 단순한 문자나 숫자로 나열하기보다는 실제 계산결과를 즉시 그래픽으로 보여줌으로서 결과를 더욱 빨리 이해할 수 있으며 영상 내에서의 위치 선정 등을 위해 마우스를 통한 입력이 필수적이다. 또한 여러 가지 작업을 하는 프로그램을 개발하고 트랜스퓨터 상에서 실행시킬 때 결과를 독립적인 창을 통해 나타낼 수 있는 그래픽 환경이 호스트에 필요하다.

2) 프로세서 감시기의 필요성

주어진 문제를 해결하기 위해 프로그래머가 원시코드를 작성하고 성공적으로 실행 파일이 생성되었다고 하더라도 실제 시스템에서 실행시킬 때 원하는 결과가 나오지 않는 경우가 많이 있다. 순차처리 프로그램의 경우에는 많은 디버거들이 원시코드의 줄 단위로 디버깅을 행할 수가 있고 현재의 변수 값들을 프로그램의 실행 시간에 관찰하여 프로그램의 오류가 생긴 곳을 쉽게 찾아낼 수 있다. 또한 프로그램의 성능을 개선시키기 위해 시간이 가장 많이 걸리는 비교적 적은 양의 프로그램 부분만을 분석하여 최적화 시킴으로서 성능의 개선을 쉽게 할 수 있다. 그러나 병렬처리 프로그램의 경우 여러 개의 프로세서에서 동시에 프로그램이 수행되므로 원시코드 레벨의 줄 단위로 실행되는 디버거를 기대하기가 어렵다. 비록 원시코드레벨의 디버거를 이용한다고 할지라도 많은 개수의 프로세서마다 한 개씩의 디버거를 필요로 하므로 순차처리 프로그램에서의 디버깅과는 다른 접근 방법이 요구된다. 또한 병렬처리 프로그램의 성능을 개선시키기 위해서는 단순한 원시코드의 최적화보다는 부하의 불균형 문제나 통신상의 오버헤드를 최소화시키는 등의 병렬처리 프로그램에서 나타나는 근본적인 문제를 해결하면서 프로그램을 작성해야 한다. 이러한 병렬처리 프로그램에서 나타나는 문제를 해결하고 프로그램의 성능을 개선시키기 위해서 프로세서의 실행 중에 모든 프로세서의 현재 상태를 외부에서 감시할 수 있는 프로세서의 활동 감시기가 필요하게 된다[10,11]. 트랜스퓨터는 한 개의 중앙처리장치와 4개의 통신링크로 현재의 상태를 나타낼 수 있다. 이들의 현재 상태를 주기적으로 호스트 컴퓨터 상에 나타냄으로서 모든 프로세서의 현재 상태를 감시할 수 있고 부하의 불균형이나 통신상의 오버헤드가 심한 곳을 집중적으로 분석함으로서 프로그램의 성능을 개선하는데 도움을 줄 수 있다.

TMAN은 이러한 필요성에 의해 개발된 트랜스퓨터 관리기이며 호스트에 그래픽 사용자 환경을 구축하고 그 내부에 프로세서 감시기를 구현하였다. 그림 6에는 TMAN의 블록 다이어그램이 나타나 있으며 이 블록들에는 프로세서의 활동을 감시하기 위한 활동 감시기, 실행되는 프로그램의 문자 입출력을 담당하는 콘솔 입출력, 파일의 입출력을 담당하는 파일 서버, 그래픽 입출력을 담당하는 그래픽 서버 및 마우스, 그리고 이들 전체 블록을 트랜스퓨터와 통신하게 해주는 트랜스퓨터 인터페이스로 구성되어 있다.

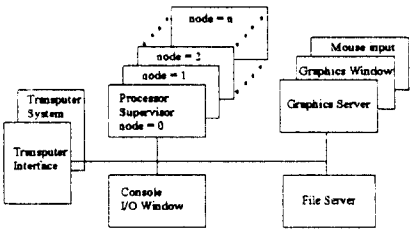


그림 6. TMAN의 블록 다이어그램
Fig. 6. Block diagram of TMAN.

TMAN을 구현하는데 있어서 호스트 컴퓨터와 트랜스퓨터간의 인터페이스, 트랜스퓨터에서의 그래픽 함수 사용방법 그리고 프로세서 감시기의 구현에 관해 기술하면 다음과 같다.

1) 호스트 컴퓨터와 인터페이스

트랜스퓨터와 호스트 컴퓨터와의 인터페이스는 기본적으로 링크 라이브러리를 이용하였다. 본 라이브러리는 트랜스퓨터의 표준 인터페이스로서 시스템 구입시 함께 제공되며 호스트와 트랜스퓨터간에 하위레벨의 데이터 교환 함수를 제공한다. 이 라이브러리를 기본으로 하여 호스트는 파일, 콘솔, 그래픽을 제공하는 서버로 동작하고 트랜스퓨터는 프로그램을 실행시키는 클라이언트로 동작한다. 트랜스퓨터가 호스트의 각종자원을 이용하기 위한 방법으로 C언어의 함수를 사용하며 이 함수는 미리 정해진 형태로 패킷화하여 호스트로 전달되고 호스트 컴퓨터는 이 패킷을 해석하여 작업을 수행한 후 결과를 트랜스퓨터로 돌려준다.

그림 7에 line() 함수를 트랜스퓨터에서 사용할 때의

과정을 도식하였다. 트랜스퓨터는 line()을 호출하며 이것을 일정한 형태의 패킷으로 변환시킨 후 표준인터페이스인 링크 라이브러리를 이용하여 호스트로 전송시킨다. 호스트는 패킷을 받은 후 해석하고 요구하는 작업인 선을 그은 후 결과 값인 0을 패킷화하여 트랜스퓨터로 보내준다. 마지막으로 결과패킷을 트랜스퓨터가 해석함으로서 line() 함수를 사용하는 한 과정을 마치게 된다.

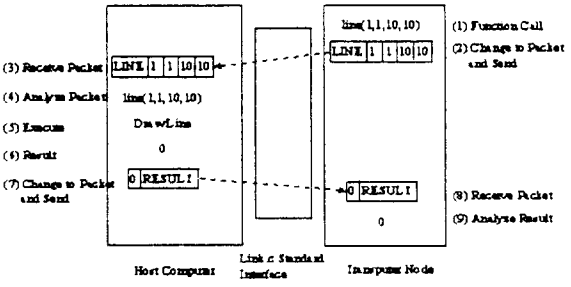


그림 7. 호스트와 트랜스퓨터간의 인터페이스
Fig. 7. Interface between host and Transputer.

2) 그래픽 함수의 사용법

TMAN은 트랜스퓨터에서 그래픽을 사용하기 위한 함수를 추가로 제공하며 보통의 함수를 사용하는 것과 유사한 방식으로 사용한다. 그래픽 함수는 크게 3부분으로 나누어지는데 그래픽 창을 제어하는 부분, 그림을 그리는 부분, 마우스를 제어하는 부분이 있고 각 부분에는 다음과 같은 함수들이 구현되어 있다.

제어 기능	관련 함수
윈도우의 제어	initgraph(), closegraph(), getgstatus(), getmaxxy(), reseterror(), exposedarea()
그래픽스의 제어	vline(), line(), rectangle(), putpixel(), cleargraph()
마우스의 제어	getmousexy()

이 함수들을 이용하여 그래픽 창을 열고, 닫을 수 있으며 그 창위에 점, 선 등을 이용하여 그림을 그리고 원하는 위치를 마우스로 입력받아 사용할 수 있다.

3) 프로세서의 상태를 측정하는 방법

트랜스퓨터에서 프로세서의 상태는 프로세서의 부하율과 프로세서 링크를 통한 통신율로 나타낼 수 있다. 프로세서의 부하율은 프로세서의 휴식(idle) 시간을 측정함으로서 이 시간과 역비례 관계를 이용하였다. 한 개의 프로세서 내에는 여러 개의 실행중인 프로세스가 존재하고 여기에 한 개의 허수아비 프로세스를 삽입함으로서 이 프로세스가 일정한 시간동안에 실행되는 횟수를 셈으로서 간접적으로 프로세서의 휴식시간을 알아내는 방법을 사용하였다. 통신링크상의 통신율을 측정하기 위한 방법으로는 통신을 하는 모든 데이터를 실제로 계산하였다. 트랜스퓨터는 4개의 통신링크를 통해서 데이터를 교환하며 이때마다 전송되는 데이터 크기를 세어서 일정한 영역에 누적시키고 주기적으로 외부에서 이 값을 읽어들인다.

프로세서의 상태를 측정하기 위한 초기화로 더미프로세서를 삽입하고 부하율 및 통신율을 0으로 한다. 그림 8은 3개의 감시기와 3개의 노드를 나타낸 것인데, 여기에서 S1, S2, S3는 TMAN상의 3개의 감시기를 나타내며 각각 N1, N2, N3는 트랜스퓨터의 노드를 나타낸다. TMAN은 각각의 감시기에게 주기적으로 데이터를 갱신하라는 메시지를 보내며 이 메시지를 받은 감시기는 자신이 관리하는 트랜스퓨터의 부하율과 통신율 값을 읽어들이고 이 값으로 자신의 상태를 갱신한다.

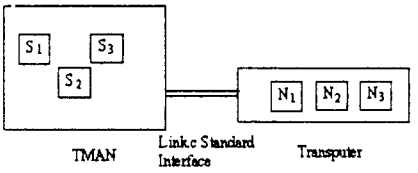


그림 8. 3개의 프로세서를 독립적으로 관리하는 3개의 감시기
Fig. 8. Three supervisors which manage three processors independently.

TMAN 각부의 주요 기능은 다음과 같다.

1) 프로세서 활동 감시기 : 트랜스퓨터의 상태는 프로세서 자체의 부하율과 각각 4개의 통신링크 상에서 나타나는 통신상의 부하율로 나타낼 수 있다. 프로세서 자체의 부하는 0%에서 100%로 나타내어지며 0%일 때는 프로세서가 정지되어 있거나 동기화를 위한 대기 등으로 프로세서가 활동을 하지 않는 상태이며 100% 일때는 프로세서가 100% 활동하고 있다는 것을 의미한다. 통신링크 상에서 나타나는 통신상의 부하는 4개의 통신링크가 물리적으로 연결된 상태를 나타내고 각각의 통신링크에서 단위시간동안 송수신한 데이터 양을 나타낸다. TMAN에서는 개개의 프로세서 모듈을 한 개의 창을 가진 객체로 정의하고, 프로세서의 부하율, 통신링크들간의 연결 상태, 각각 4개의 통신링크들

의 부하를 정의하고 있다. 이를 이용해서 전체적인 프로세서간의 현재 진행상황을 한눈에 파악할 수 있다.

2) 콘솔 입출력 : TMAN은 프로그램이 트랜스퓨터의 프로세서 상에서 실행되고 있을 때 현재 상태를 감시하는 것으로 실행시간에 텍스트의 입출력, 키보드 입력 등이 함께 요구된다. TMAN에는 한 개의 콘솔 창을 가지고 있고 그 창을 통해 트랜스퓨터의 콘솔 입출력을 담당하게 된다.

3) 파일 서버 : 트랜스퓨터 시스템은 각 프로세서들이 독립된 디스크 시스템을 갖지 않고 보드 상에 연결되어 있기 때문에 에드온 보드의 형태로 되어 있고 이 시스템에서 파일을 사용하기 위해서는 호스트 컴퓨터의 파일 시스템을 사용한다. 이를 위해서 호스트 컴퓨터에 전용의 파일 서버가 있어서 트랜스퓨터와의 파일 입출력을 담당한다.

4) 그래픽스 서버 : TMAN에는 한 개의 전용 그래픽 창이 그래픽 초기화 루틴과 함께 생성되며 창의 이동, 크기 변환 등의 기능을 제공하고 있다. 또한 화소 단위에서 화면을 제어할 수 있도록 하였고, 점, 선, 색상 등의 기본 그래픽 요소를 호출할 수 있는 그래픽 함수가 제공되어 결과를 그래픽으로 나타내는데 용이하도록 설계하였다.

사용자의 편의를 도모하기 위하여 키보드 이외에 마우스 드라이버를 장착하였다. 따라서, 마우스를 이용하여 마우스가 가리키고 있는 현재 위치, 마우스 버튼의 누름 상태를 읽어들일 수 있어서 마우스를 통한 입력을 행할 수 있다.

그림 9에 TMAN의 실행 장면을 보여주고 있다. 이 장면은 5개의 프로세서가 장착된 트랜스퓨터 시스템에서 만델브로 세트를 계산하는 프로그램을 실행중이며 Node1 ~Node5의 5개의 활동 감시기, 콘솔 입출력 창, 그래픽 창, 마우스 커서 등을 보여주고 있다.

V. 트랜스퓨터 웜 (TWORM: Transputer Worm)

여러 개의 프로세서로 구성된 병렬처리 시스템에서는 그 전체 시스템의 프로세서들 간의 연결구조를 알아야 한다. 미리 특정한 구조로 연결시켜 놓고 사용자가 연결 상태를 직접 기술하기도 하지만 임의의 개수로 구성된 프로세서들의 망 구조를 일일이 찾아내기란 쉬운 일이 아니다. 혹은 연결 상태가 다른 여러 시스템들 간의 최적화된 프로그램을 만들기 위해 프로그램의 실행 중에 시스템의 연결 상태를 알아내고 시스템 전체를 부팅(booting) 시키는 프로그램을 생각할 수 있는데 이와 같이 무작위로 채널을 이동하면서 시스템 상태를 찾아내는 프로그램을 웜(worm)[3]이라고 한다. Inmos사의 트랜스퓨터사의 개발도구 중의 하나로 제공되어지는 ispy 프로그램은 웜의 대표적인 예이다. 이 프로그램은 독립도구로 개발되어서 호스트컴퓨터 상에서 트랜스퓨터 망의 연결상태를 보여준다.

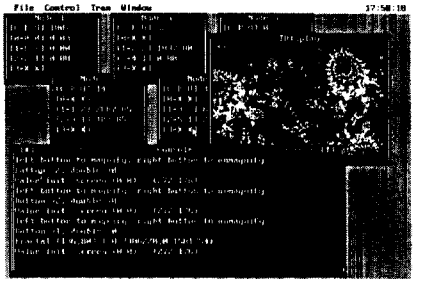


그림 9. TMAN 실행 장면
Fig. 9. Execution process of TMAN.

본 연구에서는 TWORM이라는 트랜스퓨터 상에 웜을 구현하였는바, 그 기능은 ispy와 유사하나, TIE 내부에 탑재되어서 프로그래머는 개발도중 필요시마다 TWORM을 호출할 수 있도록 하였다. 일단 호스트 컴퓨터가 한 개의 루트 노드를 부팅 시키고 웜을 로드시키면 이 웜은 4개의 통신링크를 통해서 연결된 다른 노드들을 찾아낸다. 이 노드들 중 부팅되지 않은 노드를 발견했을 때 그쪽 노드(child node)를 부팅시킨 후 웜을 로드시킨다. 이러한 일련의 과정이 모든 노드로 차례로 전파되며 더 이상의 부팅되지 않은 노드를 찾지 못할 때 자신의 현재 정보를 자신을 부팅시킨 노드(parent node)로 넘겨주게 된다. 호스트 컴퓨터는 모든 노드로부터 노드 번호와 연결 상태의 정보를 종합, 분석하여 전체적인 연결 상태를 알아내게 된다. 이러한 웜을 이용하여 임의의 구조로 연결된 트랜스퓨터 시스템의 연결 상태를 알아보고 이 연결 상태의 정보를 실행될 프로그램에 입력하여 프로그램을 최적으로 로드시키는데 사용될 수 있도록 하였다. 그림 8은 부모 노드가 2개의 자손 노드를 찾아서 부팅시키고 웜을 전파시키며 나머지 한 개의 노드는 부팅이 안된 상태를 나타내고 있다. TWORM은 임의의 구조로 연결된 트랜스퓨터 노드들을 추적하면서 각 노드들이 가지고 있는

4개의 link가 채널을 구성하고 있는 상태, 프로세서들의 메모리 상태 및 프로세서간의 전송속도 등을 측정할 수 있게 하여준다. 따라서 프로그램의 분산뿐만이 아니고 전체적인 병렬시스템의 관리에 필수적인 역할을 한다.

그림 10은 부모 노드가 2개의 자손 노드를 찾아서 부팅시키고 웜을 전파시키며 나머지 한 개의 노드는 부팅이 안된 상태를 나타내고 있다. TWORM은 임의의 구조로 연결된 트랜스퓨터 노드들을 추적하면서 각 노드들이 가지고 있는 4개의 링크가 채널을 구성하고 있는 상태, 프로세서들의 메모리 상태 및 프로세서간의 전송속도 등을 측정할 수 있게 하여준다. 따라서 프로그램의 분산뿐만이 아니고 전체적인 병렬시스템의 관리에 필수적인 역할을 한다.

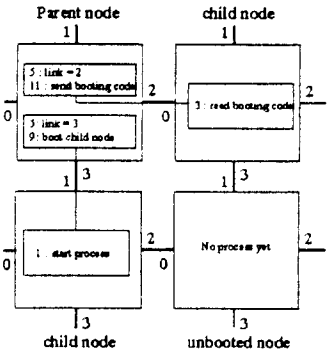


그림 10. Worm의 전파
Fig. 10. Propagation of worm.

VI. 통합환경의 응용사례

트랜스퓨터 상에서 병렬프로그램의 개발을 용이하게 하는 TIE와 TMAN의 구현 목적 중의 하나는 현실적인 컴퓨터 관련 문제들을 병렬화하는데 있었다. 따라서 TIE와 TMAN의 평가를 위해서, 주위에서 자주 이용되면서 집중적인 컴퓨터 처리시간을 요구하는 광추적 알고리듬을 병렬화하고 그 성능을 측정하였다. 광추적 기법[12]은 물체가 조명을 받고 있을 때 그 복합영상을 표현하는 방법으로 컴퓨터 그래픽스 분야에서 오랫동안 연구되어 왔다. 본 연구에서는 빛을 역추적하는 방법으로 추적하고자 하는 광선을 관찰자의 눈에서 출발시켜 화면의 한 점을 통과하게 모델링하였다. 이 빛이 실제 물체들과 교차하는지를 계산하고 교차하면 그 물체의 특성에 따라 반사, 굴절, 투과 현상을 시뮬레이션하고 각기 합당한 빛의 강도를 계산한다. 이러한 반사, 굴절, 투과된 빛들은 또 다시 다른 물체와 교차하는지를 계산하고 이와 같은 과정을 계속 반복하여 결국 빛의 강도가 문턱값 이하로 약해지거나 빛이 아무것과도 교차하지 않았을 때 그 화면에 대한 점의 빛 역추적 과정이 끝나게 된다. 이 것을 화면상에 존재하는 모든 화소에 적용해서 완전한 한 영상을 만들게 된다.

1. 광추적알고리듬의 병렬화

빛을 역추적하는 방법은 화면의 서로 다른 점들 간의 직접적인 관련성이 없으므로 모든 점들을 따로 계산하여 병렬성을 최대로 이용할 수 있다는 장점이 있다. 본 연구에서는 프로세서 farm 모델[3]을 적용해서 병렬광추적알고리듬을 구현하였으며, 이의 자세한 구현은 전문학술지에 게재될 예정이다.[13] 광추적알고리듬은 다음과 같이 간략히 설명될 수 있다. 처음에 프로세서가 부팅이 되고 프로그램이 모든 프로세서로 로드가 된다. 제어기는 영상의 크기 정보를 가지고 있으며 두 개의 프로세서로 분리가 된다. 첫 번째는 일꾼의 요구에 따라 작업을 할당하고 두 번째는 일꾼의 처리 결과를 받아서 조합한 후에 호스트 컴퓨터로 결과를 보내준다. 일꾼은 모든 물체들과 빛의 정보를 가지고 있으며 제어기에 작업을 요구하고 작업을 할당받아서 처리한 뒤 결과를 다시 제어기에 보내는 반복 루프로 구성되어 있다. 작업이 할당되는 크기는 프로세서 수에 따라 바꾸어 질 수 있으며 대표적인 크기는 scan line의 크기(본 연구의 경우는 256 pixels)이다. 이러한 일련의 과정들이 영상 전체를 만들 때까지 제어기에 의해 진행되며 전체 영상을 만들었을 때 제어기는 끝났다는 메시지를 모든 일꾼들에게 보내고 자신도 끝나게 된다.

2. 실험 결과와 분석

TIE와 TMAN을 사용하여 병렬프로그램을 개발할 경우 다른 도구들을 이용하여 프로그램을 개발하는 경우에 소요되는 개발시간의 차이 등을 정량적인 방법으로 분석할 수 는 없었다. 이는 다른 개발시스템과의 정확한 비교분석이 이루어질 수 없음에 기인한다. 그러나, 본 연구에서 개발된 통합환경을 이용하여 병렬광추적 알고리듬을 개발하는 과정은 손쉽고, 비교적 짧은 시간 내에 이루어질 수 있었다. 이는 TIE가 제공하는 개발 도구들이 상호 연관적으로 프로그래머들을 도와주기 때문이었다. 또한 TMAN을 통하여 프로세서의 상태를

그래픽스환경에서 분석하고, 그에 따라 프로세서의 점유율을 높이고 병목현상을 가급적 줄이는 과정등은 병렬프로그램의 효율을 현저하게 높여주는 것을 관찰하였다.

프로세서 farm을 이용한 병렬광추적 알고리듬은 10개의 일꾼 프로세서를 가진 트랜스퓨터 상에서 각기 실행하였다. 입력 영상은 객체관계기술언어(Objective Relationship Descriptive Language)로 표현되었고, 표현된 물체들은 CSG(constructive solid geometry) 방법을 이용하여 최종 형태를 정의하였다. 즉, 실제의 물체를 표현하기 위해서 여러 개의 평면, 구, 실린더 등의 기본요소(primitive)들을 서로 조합하여 원하는 물체를 만들어 내는 방식이다. 출력 영상은 24bit true color로 표현되며 본 실행 예에서는 중간크기(256×256pixels)의 영상을 출력으로 사용하였다. 출력파일은 그래픽스 소프트웨어를 사용하여 화면으로 영상을 볼 수 있다. 입력되는 영상의 종류는 간단히 기본요소를 가지고 있는 영상에서부터 100여개 이상의 기본요소들로 구성된 4개의 영상으로 실험하였다.

보통 응용프로그램을 사용할 때 보다 TMAN에서 실행시켰을 때 오버헤드가 생기는데 이 오버헤드는 주로 결과를 그래픽으로 표시하는 부분과 프로세서 감시기 부분에서 나타난다. 표 1은 TMAN의 프로세서 감시기를 사용하여 병렬 광추적알고리듬의 진행상황을 관측하였을 때의 시간과 오버헤드를 나타낸다.

표 1. TMAN에서 병렬 광추적 계산에 소요된 시간

Table 1. Execution times of parallel ray tracing inside TMAN.

단위 : 초(″)

영 상	TMAN 미 사용	TMAN 사용		오버헤드
		감시기 미 사용	감시기 사용	
Basicvue	24″	57″	1:05″	170%
Cluster	1:53″	1:54″	1:54″	0.008%
Tetra	4:53″	4:54″	4:55″	0.006%
Wealth	7:41″	7:42″	7:44″	0.005%

표 1의 세로축은 실험에 사용한 4개의 영상을 나타내며 가로축은 각각 TMAN을 사용하지 않았을 때, TMAN을 사용하고 감시기를 사용하지 않았을 때 그리고 감시기를 사용하였을 때와 오버헤드를 나타낸다. 여기에서 TMAN을 사용하고 감시기를 사용하지 않았다는 것은 트랜스퓨터가 TMAN과 인터페이스를 하면서 결과를 TMAN의 콘솔로 쓰는 상태를 나타낸다. 오버헤드를 보면 Basicvue인 경우에 170%로 크게 나타나며 나머지의 경우는 거의 0%에 근접하는 것을 볼 수 있다. Basicvue의 경우에 감시기를 통해서 진행상황을 관측하면 10개의 프로세서 중에서 2~3개만이 활동하는 것을 볼 수 있는데 이것은 Basicvue의 영상이 간단하여 광추적 계산을 하는 시간보다 프로세서간의 정보교환을 하는데 오버헤드가 상대적으로 크기 때문이다. 이러한 오버헤드는 TMAN과 감시기를 사용함으로서 급격하게 증가하는 것을 볼 수 있다. 그러나 나머지의 3개의 영상은 TMAN의 감시기를 통해 관측할 때 10개의 프로세서가 모두 활동하고 있는 것으로 나타나는데 이것은 프로세서간의 정보교환을 하는데 상대적으로 시간을 적게 소모하는 것을 의미하고 또한 TMAN을 사용하는데 대한 오버헤드가 적게 나타나는 것으로 분석된다.

Ⅶ. 결 론

본 논문에서는 트랜스퓨터 시스템에서 병렬 프로그래밍을 효율적으로 도와주는 프로그래밍 도구에 관해 논하였다. 트랜스퓨터 통합환경은 객체 지향적으로 구성되어 있어서 트랜스퓨터에서 사용하는 다른 언어인 Occam, 병렬-Fortran, 병렬-Pascal과도 쉽게 재구성이 되어 다른 형태의 통합환경으로도 사용할 수 있다. 또한 복잡한 망으로 구성되어 있는 프로세서들의 망 구조를 찾아내고 프로세서들을 부팅 시킬 수 있는 웜(worm)을 개발하여 트랜스퓨터의 망 구조를 알아낼 수 있도록 하였다. 시스템의 성능을 실험하기 위하여 광추적 알고리듬을 트랜스퓨터 상에서 병렬적으로 실행될 수 있도록 하였으며, 앞에서 개발한 통합환경을 이용해서 비교적 쉽게 오류를 찾아낼 수 있도록 구현하였다. 병렬 광추적알고리듬을 여러 개의 영상을 통하여 실험하였으며, 그 속도를 측정하고 TMAN의 프로세서 감시기를 사용하였을 때의 오버헤드를 측정하였다. 병렬프로그램의 개발은 일반 통합환경과 비교하여 쉽고 짧은 시간 내에 이루어질 수 있으며, 프로세서 farm을 이용하여 구현된 병렬광추적 프로그램도 이상적인 속도향상율에 가까운 우수한 병렬성을 나타내었

다.

추후의 연구과제로서는 일부 초고속 컴퓨터에서 개발된 바와 같이, 순차처리 프로그램의 내부 루프 등을 자동적으로 해석하여 프로세서의 개수에 의존하지 않고 트랜스퓨터 표준 병렬처리 프로그램으로 만들어 주는 라이브러리를 개발하는 것으로, 이는 트랜스퓨터 프로그램 개발과정을 더욱 단축할 수 있을 것이다. 또한 현재 병렬처리 프로그램의 경우에도 할당되는 작업의 크기와 이에 관련된 통신 오버헤드 등은 좀더 깊은 분석이 필요하다. 이와 같은 병렬처리 프로그램의 도구는 본 연구에서 수행된 통합환경도구와 함께 병렬프로그래밍을 쉽게 구현할 수 있도록 순차프로그램에만 익숙한 소프트기사들도 도와줄 수 있고 가격 대 성능비가 우수한 트랜스퓨터 시스템의 활용을 촉진할 것으로 기대된다.

※ 이 논문은 1994년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음.

참 고 문 헌

[1] Inmos Limited, "Transputer Reference Manual," Prentice-Hall, 1988.

[2] John Roverts, "Transputer Assembly Language Programming," Van Nostrand Reinhold, 1992.

[3] Ronald S. Cok, "Parallel Programms for the Transputer," Prentice-Hall, 1991.

[4] Donald Hearn and M. Pauline Baker, "Computer Graphics," Prentice-Hall, 1986.

[5] Steve Harrington, "Computer Graphics: A programming approach," McGraw-Hill, 1987.

[6] Kai Hwang, "Advanced Computer Architecture," McGraw-Hill, 1993.

[7] Inmos Limited, "Occam 2 Reference Manual," Prentice-Hall.

[8] Logical Systems, "Transputer Toolset: Toolset Reference, Parallel C for the Transputer," 1994.

[9] Logical Systems, "Transputer Toolset: C Library Reference, Parallel C for the Transputer," 1994.

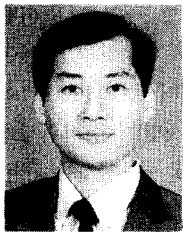
[10] J. Jiang, A. Wagner and S. Chanso, "Tmon: A Real-Time Performance Monitor for Transputer-based Multicomputer," Transputer Research and Applications 4, pages 36-45, Amsterdam, 1990, IOS Press.

[11] Mats Aspnas, Thomas Langbacka, "A Monitoring System for a Transputer-Based Multiprocessor," Transputing '91 Volume 1., pp. 78-93, Amsterdam, 1991, IOS Press.

[12] Andrew Glassner, "An Introduction to Ray Tracing," 1989, McGraw-Hill.

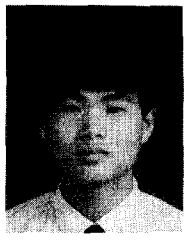
[13] 이효종, 임훈철, "트랜스퓨터 시스템에서의 병렬 광추적 알고리듬," 정보과학회지 게재예정, 1996

저 자 소 개



李 孝 鍾(正會員)

1957년 6월 19일생. 1982년 전북대학교 지구과학과 졸업, 이학사. 1985년 University of Utah 기상학과 졸업, 이학석사. 1986년 동 대학교 컴퓨터과학과 졸업, 공학사. 1988년과 1991년 동 대학원 졸업, 각각 공학석사와 공학박사. 1986 ~ 1989년 University of Utah 컴퓨터과학과 조교. 1989 ~ 1991년 동 대학교 선임프로그래머. 1991년 동 대학교 대기연구소 연구원. 1991년 9월 전북대학교 전자공학과 조교수. 주관심 분야로 병렬처리, 인공지능, 병렬인식알고리즘 개발 및 소프트웨어공학



任 訓 徹(正會員)

1972년 8월 5일생. 1994년 2월 전북대학교 전자공학과 졸업(공학사). 1996년 2월 전북대학교 전자공학과 졸업(석사). 1996년 ~ 현재 국방과학연구소 연구원. 주관심 분야는 컴퓨터그래픽스 및 병렬처리 등임