論文92-29B-1-2

多重 루프문의 竝列處理를 위한 타스크 스케쥴링에 관한 硏究

(Study on Task Scheduling for Parallel Processing of Nested Loops)

許 丁 淵*, 孫 潤 求**

(Journg Yourn Hur and Yoon Koo Sohn)

要 約

본 논문에서는 순차처리 프로그램 중 다중 루프문의 병렬처리를 위해 큐잉이론을 이용한 해석적 모델을 제시하고 실 시스템에서의 실행을 위해 4개의 intel 8088 마이크로 프로세서와 2K 바이트 분할 메모리, 8 바이트의 스테터스 및 하드웨어 토큰 링으로 구성된 다중 프로세서 시스템을 제작하여 FORTRAN으로된 다중 루프문의 해석적 모델과 실 시스템에서의 실행결과를 스피드업으로 비교 분석하였다. 해석적 모델과 실 시스템에서의 처리 및 대상 프로그램의 이론적 계산결과 모두는 매우 일치하였다. 제안된 해석적 모델은 다중 반복 루프문의 병렬처리에 평가도구로 사용될 수 있다.

Abstract

This paper is to propose an analytical queuing model for parallel processing of sequential program with nested loops. The analytical results are compared with the results from the implemented multiprocessor system composed of four intel 8088 microprocessor, eight 2KB shared common memories, and a hardware token ring. At results, this study shows that the processed results are almost similar in proposed analytical model and real system. Proposed analytical model can be applied to evaluate parallel processing of sequential program with nested loops.

I. 서 론

과학및 공학의 많은 영역에서 보다 빠른 계산속도를 갖는 컴퓨터가 요구되고 있다.

다중 프로세서 시스템에서 처리속도를 향상하기 위해서는 아래와 같은 문제들이 해결되어야 한다.

첫째 : 현재 사용중인 단일 프로세서에 비해 효과적으로 병렬처리를 위한 동기화(synchronizing) 메카니즘의 개발.[1]

둘째 : 기존 순차 프로그램(sepuential program)을 병렬처리하기 위한 병열성 검출과 프로그램의 재구성.[2]

세째 : 재구성된 프로그램의 분할과 제한된 프로세서의 최적 스케쥴링(scheduling).[3,4]

기존 스케쥴링 방법은 그래프이론, 선형 프로그램 및 휴리스틱 방법으로 연구되어 왔으나 각각의 단점을 갖고 있어 최적 스케쥴링이 되지 못하고 있다.[5,6,7,8]

본 논문은 다중 프로세서 시스템에서의 병렬 처리를 위한 최적 큐잉 모델을 제시하였고 시뮬레이션 및 계산결과를 비교하기 위해 4개의 프로세서로된 다중 프로세서 시스템을 제작하여 실 시스템의 실행결과와 비교하였다.

여기서 비교 분석된 FORTRAN 대상 프로그램의 다중 루프문은 가장 안쪽 실행문들을 단위 타스크로

*正會員, 慶南大學校 電子計算學科
(Dept. of Computer Eng., Kyungnam Univ.)
**正會員, 嶺南大學校 電算工學科
(Dept. of Computer Eng., Youngnam Univ.)
接受日字 : 1991年 10月 10日

하여 구현된 시스템에서는 동일한 구조로된 어셈브리언어로 변환한 후 처리하였다.

Ⅱ. 다중 프로세서 시스템의 구조

다중 프로세서 시스템은 파이프라인 컴퓨터와 어레이 컴퓨터의 단점을 해소할 뿐 아니라 동시에 다수개의 타스크를 실행할 수 있어 처리속도 향상에 매우 적합한 시스템으로 알려져 있다.

그러나 다중 프로세서 시스템에서 최고의 처리속도를 얻기 위해선 아래와 같은 문제들이 해결되어야 한다.

- 순차 프로그램의 재구성과 분할
- 내부 접속망을 이용한 메모리 억세스 전략
- 타스크 상호간 동기화
- 스케쥴링[6,9]

계층구조를 갖는 소결합 멀티프로세서 시스템은 프로세서간 및 분할메모리 억세스 시 응답시간이 너무 늦어 프로그램의 고속 병렬처리 및 실시간 처리에는 적합하지 못하므로 본 논문에서는 밀결합 멀티프로세서 시스템을 구성하였다.

그림 1은 구현한 시스템의 구성도를 보여준다.

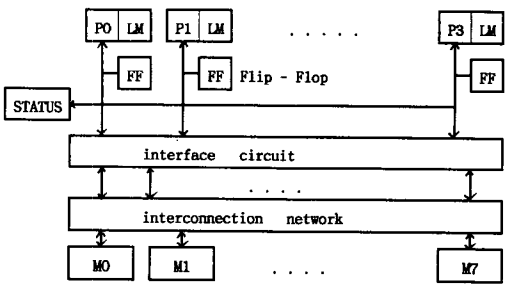


그림 1. 구현한 다중프로세서 시스템의 구성도
Fig. 1. Block diagram of implemented multiprocessor system.

시스템의 회로구성은 크게 각 프로세서와 내부 접속망 사이의 인터페이스 회로와 내부접속망 및 분할메모리 회로, 시스템의 상태를 나타내는 스테터스 및 스테터스의 억세스 권한을 각 프로세서에 부여하는 토큰(token) 회로의 세 부문으로 나뉘어 진다.

시스템은 최대 P0에서 P3까지 4개의 프로세서를 접속할 수 있으며 분할메모리도 M0에서 M7까지 8개를 접속할 수 있도록 구성하였다.

프로세서가 분할메모리 중 한개를 억세스할 경우 먼저 자신의 프립프롭 비트 중 토큰으로 할당된 비트를 확인한 후 억세스토록 하여 스테터스 억세스 시 발생되는 경합을 해결하였다.

각 프로세서가 분할메모리를 억세스할 수 있는 토큰의 전달방법은 프로세서 4개를 대상으로 토큰링을 구성하면 그림 2와 같다. 그림 2에서 토큰은 Q1이되며 Q1이 세트(set)된 프로세서는 분할 메모리를 억세스할 수 있는 권한을 갖는다.

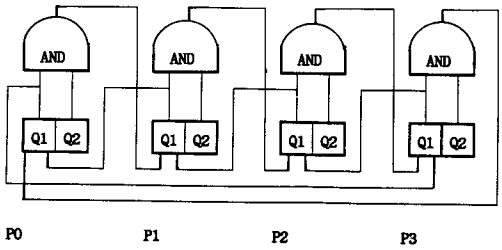


그림 2. 하드웨어 토큰링의 구조도
Fig. 2. Structure of hardware token ring.

프립프롭출력 Q1은 전 프로세서에서 전달된 토큰에 의하여 세트되며 Q2는 현 프로세서가 스테터스 사용을 종료하거나 사용하지 않을 때 세트되어 다음 프로세서에 토큰을 넘겨준다.

멀티프로세서 시스템에서 각 프로세서와 분할된 주메모리 사이의 내부 접속망(IN) 구성형태가 처리결과에 매우 큰 영향을 준다.[6,10,11,12]

제작된 시스템에서는 그림3과 같이 접점수는 크로스바 스위치보다 적고 성능은 거의 동일한 2×2 omega network로 구성하였다.

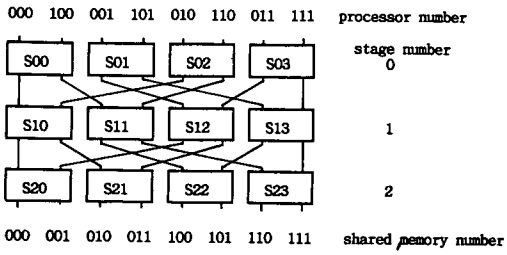


그림 3. Omega network를 이용한 내부접속망 구조
Fig. 3. Structure of interconnection network by omega network.

프로세서와 분할메모리는 각각 고유의 2진수 번지 p_2, p_1, p_0와 m_2, m_1, m_0을 가지며 한 프로세서가 한 메모리들을 억세스 할 때 각 단의 제어 신호는 아래와

같이 생성된다.

$$k = log_2(N) - 1$$
$$(C_0 C_1 \cdots C_K) = (P_0 + M_0)\ (P_1 + M_1) \cdots (P_K + M_K) : \textit{omega network}$$

생성된 제어신호의 첨자는 각 단의 제어신호이며 C_K는 k단의 제어신호가 된다.

스테터스는 A8008부터 A800F까지 8바이트를 할당하여 그림4와 같이 경로충돌 및 분할 메모리 억세스시 충돌을 방지하였다.

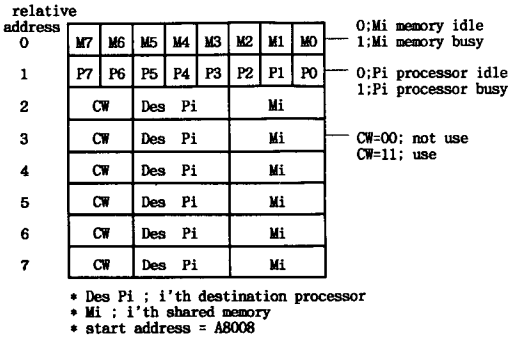


그림 4. 스테터스의 데이타 구조
Fig. 4. Data structure of status.

Ⅲ. 순차프로그램의 병렬처리를 위한 해석적 모델

다중프로세서 시스템에서 각 프로세서에서 실행될 프로그램의 분할 및 분할된 프로그램의 프로세서 할당은 처리속도에 많은 영향을 준다.

큐잉(queuing)시스템에서는 여러개 job의 출력시간, 큐에서의 대기시간, 프로세서의 전송시간 및 각 프로세서에서의 처리시간을 변수로하여 스피드업을 계산할 수 있다.

정적스케쥴링에서는 한개의 job을 여러개의 타스크로 분할한 후 정해진 순서대로 실행되므로 입출력 시간 및 대기시간을 병렬처리를 위한 지연시간으로 나타낼 수 있다.

수치 해석적 모델을 위하여 분할처리되는 루프문은 조건문을 포함하지 않으며 루프문에 실행문을 한개의 프로세서에서 실행할 경우 실행시간이 동일하다고 가정하면 데이터 종속성중 anti-dependence 와 output dependence 및 flow-dependence에 의한 순환방향성 그래프의 생성은 random하게 발생되므로 루프문의 병렬처리시 동기화가 발생할 확률은 Poisson 분포에 따른다.

처리프로그램의 병렬처리 알고리듬에 따라 동기화 없이 독립적으로 처리되는 경우와 random 한 데이터 종속성에 의해 처리되는 경우로 나누어 수치해석적 스케쥴링 모델을 구하기 위해 아래와 같이 용어를 정의하여 사용한다.

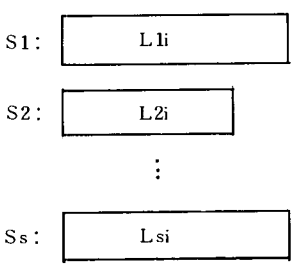
Ti : 큐에 저장된 i번째 타스크
Li : i번째 타스크의 길이
To : 한 프로세서로 타스크를 전송할 때 경로와 프로세서 상태를 조사하기 위한 스테터스 억세스 시간
Tir : 프로세서에서의 단위 타스크의 처리 시간 (1/Li 처리시간)
Tu : 한 프로세서에서 전체의 타스크를 실행할 때 처리시간
Tp : N개의 프로세서에서 M개의 타스크를 병렬로 처리할 때 처리시간
Tc : N개의 프로세서에서 스레이브 프로세서로 타스크를 전송하기 위해 분할메모리에 복사하는 단위 타스크(1/Li) 단위길이 당 복사시간
M : 프로세서에 할당되는 타스크 수(1,2,3,…, M)
N : 프로세서 수(1,2,3,…, N)

만약 각 프로세서가 동종이라 가정하면 타스크를 각프로세서에서 실행시킬 경우 실행시간은 모두 같다.

그림 4와 같이 루프문 내 실행문의 처리시간은 타스크 길이로 표시하였으며 i번째 루프에서 총실행 시간(Li)은 식(1)과 같이 각 do loop 속의 실행문 처리시간의 합으로 나타낼 수 있다.

$$Li = \sum_{K=1}^{S} L_{ki} \qquad (1)$$

do $i = 1, M$



enddo

그림 5. 루프문 내 실행문의 처리시간
Fig. 5. Execution time of execution statement

병렬처리에서 한개의 프로세서에 의한 처리시간(Tu)과 여러 개 프로세서에 의한 병렬처리시간(Tu) 사이에는 Speedup=Tu/Tp관계가 있으므로 먼저 한개의 프로세서에서 모든 타스크의 실 행시간은 식(2)와 같이된다.

$$Tu = M * Tir * Li \qquad (2)$$

길이 Li인 타스크가 분할메모리에 복사될 때 스테터스 억세스시간과 분할메모리 억세스시간의 합(Ts)은 식(3)으로 나타낼 수 있다.

$$Ts = To + Tc * Li \qquad (3)$$

이때 각 프로세서로 할당되는 타스크의 데이타 종속성이 (a)타스크 상호간 데이터 종속성이 없는 독립적인 경우와 (b)데이터 종속성이 랜덤하게 발생할 경우로 구분하였다.

(a)각 타스크간 데이터 종속성없이 독립적으로 실행될 경우 M개의 실행타스크가 N개의 프로세서에서 병렬처리되는 총 실행시간(Tp1)은 식(3)으로 나타낼 수 있다.

$$Tp1 = M * Ts + [M/n] * Li * Tir \qquad (4)$$

(b)각 타스크간 데이터 의존성이 랜덤하게 존재하며 동기화에 의한 pipeline 방법으로 실행될 때 길이 Li 타스크의 처리중 동기화가 일어날 확률은 Possion 분포에 의해

$$G(k) = exp(-\lambda) * \lambda * * k/k\,!\ (k = 0.1,2,\cdots, Li)$$

이므로 총 실행시간은 식(5)로 나타낼 수 있다.

여기에서 λ는 평균 동기화 타스크 수가 된다.

$$Tp2 = M * Ts + (2 * To + \sum_{K=1}^{Li} G(k) * k * Tir)/Li +$$
$$[M/N] * \sum_{k=1}^{Li} (Li - G(k) * k) * Tir/Li \qquad (5)$$

식(2),(4) 및 (5)에서 타스크 단위길이는 단위 타스크의 처리 시간으로 단위 타스크의 처리시간과 단위 타스크의 실행시간을 설정하는 기준에 따라 달라진다.

만일 모든 프로세서가 처리할 데이타를 분할메모리에 저장하고 각 프로세서의 로칼메모리에 컴파일된 실행파일을 가지고 각 프로세서가 할당된 타스크를 실행할 경우 마스터 프로세서에 의한 스레이브 프로세서로의 데이터 전송시간(Tc)은 무시할 수 있으며 이 경우 프로세서 동기화에 따른 스테터스 억세스시간(To)만 스피드업에 영향이 없는 경우 스피드업은 To=0.01은 거의 이상적인 경우와 유사하였으나 통신시간이 단위 실행문의 실행시간과 동일한 a4에서는 프로세서가 16일때 약 4배 감소함을 알 수 있다.

데이터 종속이 있는 b의 경우 To=1인 경우 To=0.01에 비해 스피드업이 약 2.8배 감소함을 알 수 있어 데이타 종속이 있는 경우 내부통신 비용의 증가에 따른 스피드업 강하는 상대적으로 줄어듬을 알 수 있다.

표 1에서 a는 loop문내 실행문들 상호간 데이터 의존성이 없는 경우 스피드업을 나타내며 b는 데이터 의존성이 Poisson 분포에 의해 랜덤하게 발생할 경우 스피드업의값을 나타내고 소수점 셋째 자리에서 반올림 하였으며 각 상수 및 변수의 값을 아래와 같이 설정하였다.

Tir=1, Tc=0.01,
Li=30, λ=2.5,
Ot=0.1,0.2,…,1

스피드업에 영향을 주는 주된 요소는 한타스크 실행에 따른 스테터스 억세스 시간 및 억세스 횟수에 의존하게 되며 단위 타스크의 실행시간과 타스크의 복사시간은 상수로서 스피드업에 큰 영향을 주지 않는다.

표 1. 해석적 모델의 계산 결과
Table 1. Results of analytical model.

To \ n		2	3	4	6	8	10	12	14	16
.01	a	2	3	4	5.9	7.9	9.8	11.7	13.6	15.5
	b	1.7	2.4	2.9	3.9	4.6	5.2	5.7	6.1	6.5
.02	a	2	3	3.9	5.9	7.8	9.6	11.4	13.2	15.0
	b	1.7	2.3	2.9	3.8	4.6	5.1	5.6	6.0	6.4
03.	a	2	2.9	3.9	5.8	7.6	9.4	11.2	12.9	14.6
	b	1.7	2.3	2.9	3.8	4.5	5.1	5.5	5.9	6.2
.04	a	2	2.9	3.9	5.7	7.5	9.3	10.9	12.6	14.2
	b	1.7	2.3	2.9	3.7	4.4	5.0	5.4	5.8	6.1
.05	a	2	2.9	3.8	5.7	7.4	9.1	10.7	12.3	13.8
	b	1.7	2.3	2.8	3.7	4.4	4.9	5.4	5.7	6.0
.06	a	2	2.9	3.8	5.6	7.3	8.9	10.5	12.0	13.4
	b	1.7	2.3	2.8	3.7	4.3	4.9	5.3	5.6	5.9
.07	a	1.9	2.9	3.8	5.5	7.2	8.8	10.2	11.7	13.1
	b	1.6	2.3	2.8	3.6	4.3	4.8	5.2	5.5	5.8
.08	a	1.9	2.9	3.8	5.5	7.1	8.6	10.1	11.4	12.7
	b	1.6	2.2	2.8	3.6	4.2	4.7	5.1	5.5	5.7
.09	a	1.9	2.8	3.7	5.4	7.0	8.5	9.9	11.2	12.4
	b	1.6	2.2	2.7	3.6	4.2	4.7	5.1	5.4	5.7
. 1	a	1.9	2.8	3.7	5.4	6.9	8.3	9.7	10.9	12.1
	b	1.6	2.2	2.7	3.5	4.1	4.6	5.0	5.3	5.6
1	a	1.4	1.9	2.2	2.7	3.1	3.3	3.5	3.7	3.8
	b	1.2	1.4	1.6	1.9	2.0	2.1	2.2	2.3	2.3

표 1에서 a는 식(5)에 평균 동기화율을 2.5로 하고 반복실행횟수 N을 100으로 하였을 때 프로세서수에 따른 스피드업의 결과를 보여준다.

표 1에서 a는 루프문내 실행문들 상호간 데이터 의존성이 없는 경우 스피드업을 나타내며 b는 데이터 의존성이 Poisson 분포에 의해 랜덤하게 발생한다는 가정하에 계산된 스피드업의 값을 나타낸다.

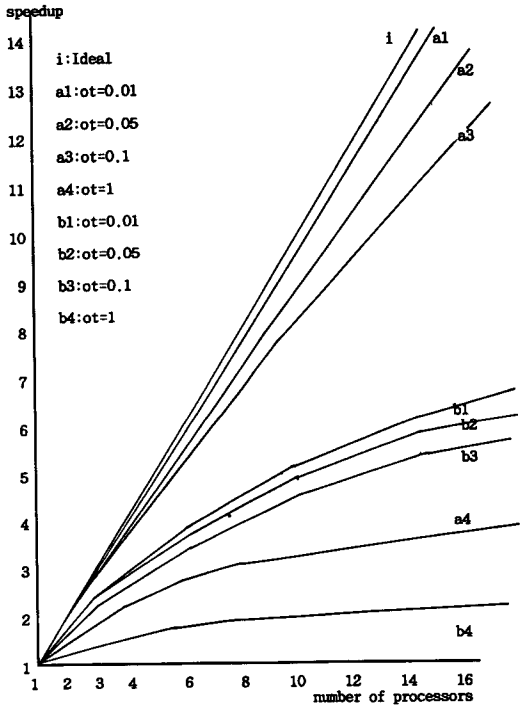
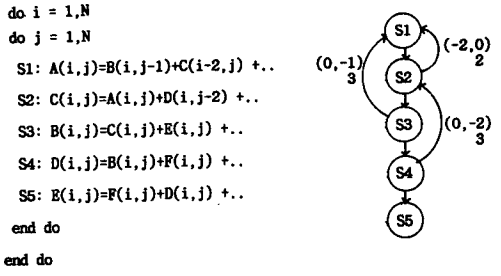


그림 6. 해석적 모델의 스피드업
Fig. 6. Speedup of analytial modeling.

Ⅳ. 시스템에서의 실행결과와 분석

본 장에서는 반복실행되는 FORTRAN의 do-loop에서 검출된 병렬성을 유한 개수로 구성된 다중 프로세서 시스템에서 실행한 후 그 결과를 비교분석 하였다.

반복실행되는 FORTRAN의 do-loop문을 재구성하면 doall문과 doacross문 및 dosequential문중 하나로 된다. doall문은 가장 안쪽 do loop문에 포함된 실행문들을 한타스크로 할당하면 전체 loop 공간에서 타스크들이 독립적으로 실행될 수 있으며 dosequential문은 전 loop공간에서 순차적으로 실행되어야 하므로 본 논문에서 제외시켰다.



(a) (b)

그림 7. 데이터 종속성을 갖는 다중 loop문
Fig. 7. Nested loop with data dependence

그림 7(a)는 loop문이 실행될 때마다 동기화가 필요한 do across이며 (b)은 다섯개의 실행문 상호간 데이타 종속관계를 표시한 데이타 종속 그래프를 보여준다.

각 가지에는 타스크가 반복실행될 때 distance vector와 각 실행문들이 실행을 시작하기 전 타스크에서 실행종료되는 할당문까지의 거리를 표시하였으며 반복실행 횟수 N*N은 적절한 스케쥴링에 의해 병렬로 실행될 수가 있다.

그림 8은 그림 7 (a)프로그램에서 각 실행문의 실행시간을 단위시간으로 하고 동기화에 따른 통신비용을 무시하였을 때 N=4인 경우 프로세서수대 스피드업을 각 4가지를 보여준다.

그리고 실 시스템에서의 실행결과를 그래프로 작성하였다.

그림 8에서 실 시스템에서의 각각의 결과는 프로세서 4개까지 일치함을 알 수 있다.

제작된 시스템에서 각 분할 메모리의 억세스를 위해 필요한 스테터스억세스 시간과 프립 프롭의 세트 리세트 시간을 측정한 결과 내부접속망에서 경로충돌과 메모리 충돌이 일어나지 않을 경우 약 0.01ms가 되었다.

각 프로세서가 토큰을 필요로 하지 않을 경우 (스테터스 억세스가 필요하지 않을 경우) 하드웨어로 구성된 토큰 링이 한 프로세서에서 다시 동일한 프로세서로 전달시간은 약 0.01μ초이므로 토큰의 전파시간은 무시하였다.

그림 7 프로그램에서 (통신시간)/(단위실행문의실행시간)=0.1이 되도록 지연을 삽입하여 N=100으로 하여 제작한 멀티프로세서 시스템에 적용하였다.

그림 8은 그림 7 프로그램을 동일한 조건하에서 큐잉이론에 의한 해석적 방법과 각 루프의 지연시간을

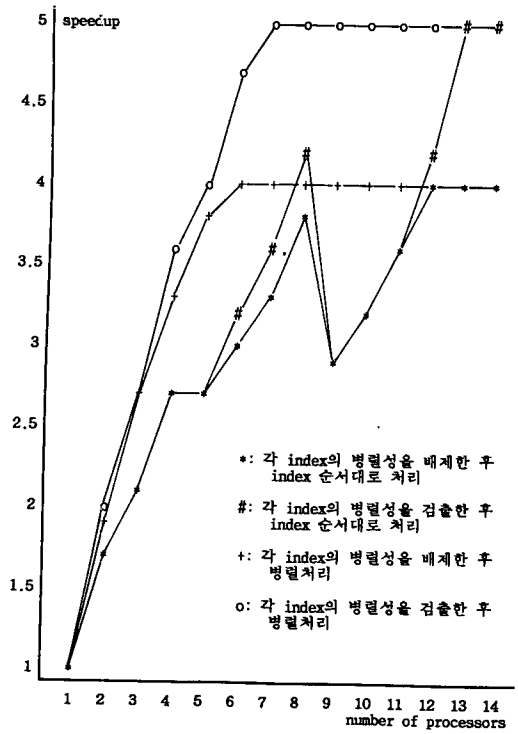


그림 8. N=4일 경우 그림 4 프로그램의 스피드업
Fig. 8. Speedup of program Fig. 4 at N=4.

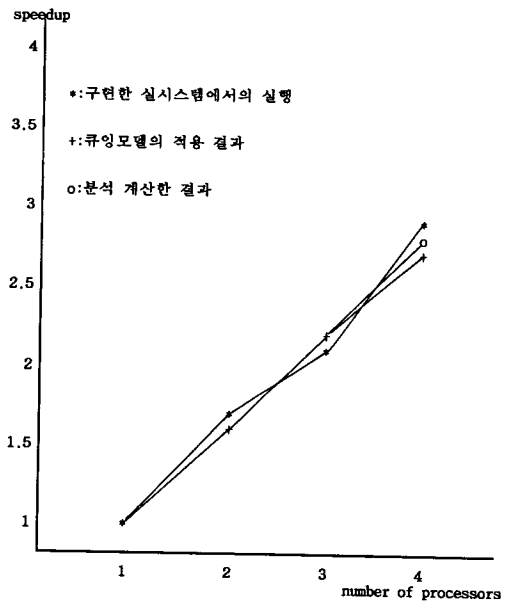


그림 9. 해석적 모델, 실시스템에서의 실행 및 계산 결과의 스피드업
Fig. 9. The execution speedup using queuing model, strict calculation and implemented multiprocessor system.

계산하여 최적 스케줄링에 의한 스피드업 그리고 실시스템에서의 실행결과를 그래프로 작성하였다.

그림 9에서 실시스템에서의 각각의 결과는 프로세서 4개까지 일치함을 알 수 있다.

V. 결 론

임의의 다중 루프문을 제안한 해석적 모델과 프로그램의 분석에 의한 계산 및 제작한 다중 프로세서 시스템에서의 결과 모두가 일치하므로 제안한 모델이 정확함을 확인하였다.

실험의 분석 결과를 요약하면 아래와 같다.

1) 제안한 큐잉 모델은 순차 처리프로그램중 다중 루프문의 병렬처리에 적용하여 결과를 매우 정확하게 예측할 수 있다.

2) 반복 실행되는 루프들의 실행순서를 빨리 발견하는 알고리듬이 개발되면 최소의 프로세서로 최대의 스피드업을 얻을 수 있다.

3) 유한 프로세서에 의한 순차 프로그램의 병렬처리는 GCD방법으로 루프변수의 병렬성을 검출한 후의 실행이 효과적이다.

4) 프로세서 수가 반복회수의 정수배일 때 매우 효과적이다.

제작된 시스템의 프로세서 수가 4개로 한정되어 프로세서가 많은 경우는 실험이 불가능하였으나 주어진 환경에서의 결과가 매우 정확하므로 다른 언어로 작성된 순차프로그램중 다중 루프문의 병렬처리에도 적용이 가능하다.

參 考 文 獻

[1] V.P. Krothapalli and P. Sadayappan, "An Approach to Synchronization for Parallel Computing," ACM0-89791 272-1/88/0007/0573 p. 573-581, (1988)

[2] G.M. Baudet, R.P. Brent & H.T. Kkung, "Parallel Execution of a Sequence of Tasks on a Asynchronous Multiprocessor," The Australlan Computer Journal, vol. 12, no. 3, p. 105-111, August 1980

[3] DAN I. Moldovan & Jose A.B. Fortes, "Partitioning and Mapping Algorithms into Fixed Size Systolic Array," *IEEE Trans. on Computer.* vol. c-35, no. 1, January 1986.

[4] Utpal Banerjee, Shyh-ching Chen, David J. Kuck & Ross A. Towle, "Time and Parallel Processor Bounds for Fortran-Like Loops," *IEEE Trans. on Computer,* vol. c-28, no. 9, September 1979.

[5] Hai-Bo Chen and Yun-Gui Ci, "Parallel Execution of Non-Do Loops," Dept. of Computer Science Changsha Institute of Technology.
[6] Kai Hwang and Faye A. Briggs, "Computer Architecture and Parallel Processing," 1st. Ed. McGrawhill, New York, pp. 459-556 (1984).
[7] Ravi Sethi, "Scheduling Graphs on Two Processors," *SIAM J. Comput.*, vol. 5, no. 1, March 1976.
[8] Peir, Jih-Kwan, "Program Partititoning and Synahronization on Multiprocessor Systems," Ph.D. Thesis University of Illinois at Urbana-Champaign (1986).
[9] Tse Yun Feng, "A Survey of Interconnection Network," Turorial Interconnection Network for parallel and distributed processing, pp. 5-20(1984).
[10] Lan Jin and YiPan, "A Kind of Interconnection Network with Mixed Static and Dynamic Topologies," Ch2149-3/85/0000/0160, p. 160-166. (1985 IEEE)
[11] E.B. Fernandez and B. Bussell, "Bounds on the Number of Processors and Time for Multiprocessor Optimal Schedules," *IEEE Trans. on Computers,* August 1973.
[12] B.W. Arden and H. Lee, "A Regular Network for Multicomputer System," *IEEE Trans. on Computer,* Jan. p.60-69 (1982)

著 者 紹 介

許 丁 淵(正會員)
1947年 11月 3日生. 1992年 2月 영남대학교 전자공학과 전자계산기 전공. 박사학위 취득예정. 현재 경남대학교 전자계산학과 부교수. 주관심분야는 parallel processing, pattern recognition 등임.

孫 潤 求 (正會員) 현재 영남대학교 전산공학과 교수