데이터 흐름기 시뮬레이터 설계 및 구현

지수영*, 조용환*, 백두권**
*충북대학교 전자계산기공학과, **고려대학교 전산과학과

The Design and Implementation of a Data Flow Machine Simulator

Su-Young Ji*, Yong-Hwan Cho*,
Doo-kwon Baik**
*Dept. Of Computer Engineering,
Chungbuk National Univ.
**Dept. Of Computer Science, Korea Univ.

ABSTRACT

In this paper, we have planned and realized, as it may be n elementary stage, Data Flow Simulator to be useful, making an experiment on evolution of performance modelling on Data Flow Machine, and then distributed simulation by using uniprocessor.

And we have got a desirable result from the realized Data Flow Simulator by allowing practical examples to have access to it.

1. 서론

최근 VLSI로 대표되는 반도체 기술의 발전으로 고도의 동시성을 지닌 다중 프로세서 시스템의 출현이 가능하게 되었다. 이러한 다중 프로세서 시스템은 작업의 효율적인 할당등에 따른 처리속도의 개선, 신뢰도 향상, 작업 처리량의 증가등 기존의 컴퓨터 시스템에 비하여 뛰어난 성능을 가지고있다. 다중 프로세서 시스템 중의 하나인 데이터 흐름기(Data Flow Machine)는 기존의 Von Neumann 연산 방식 컴퓨터와는 달리 프로그램 카운터와 메모리 개념을 사용하지 않고 프로그램내에 수행 명령들에 대한 입력 데이터로서의 이용가능성에 의해서만 실행되는 고도의 병렬처리 시스템이다[5, 6, 7]. 데이터 흐름기를 구현하는 데에는 많은 시간과 비용이 소요되므로 실제로 이러한 시스템을 구현하기 전에 이 시스템에 대한 컴퓨터 시뮬레이션을 수행할 필요가있다[11].

또한 데이터 흐름기는 분산처리 시스템이므로 분산 시뮬레이션[3, 4]을 하여야 하며, 따라서 이를 위한 분산 시뮬레이터가 필요하다.

본 논문의 제2장에서는 데이터 흐름기에 대한 성능평가 모델링을 한후 아주 간단한 작업(Job)을 Uniprocessor하에서 소프트웨어 접근 방법으로 시뮬레이션 한 결과를 분석하였다.

단일 프로세서를 사용한 이산사건 시뮬레이션에서는 사건 리스트의 순차처리로 인한 병목현상을 초래한다.

그러나, 여러개의 프로세서들을 네트워크로 묶어서 하나의 모델을 형성하여 시뮬레이션을 수행하는 분산 시뮬레이션에서는 프로세서들 내에 존재하는 병행성을 활용할 수 있기 때문에 작업의 처리 속도가 증가됨을 알 수가 있다[14, 15].

제3장에서는 좀더 실제적인 데이터 흐름기 시뮬레이터를 설계하기 위하여 분산 시뮬레이션을 위한 모델을 제시하고 이에 따른 분산 시뮬레이터를 설계하였다.

제4장에서는 실제적으로 데이터 흐름기 시뮬레이터를 8-Bit 마이크로 컴퓨터들로 구현하였으며, 또한 구현된 데이터 흐름기 시뮬레이터위에서 실제 예를들어 분산 시뮬레이션 한 결과를 보였다.

2. 데이터 흐름기 성능평가 모델링

(1) 성능평가 모델 설정

데이터 흐름기에 대한 성능평가 모델은 그림1에 도시한 바와같이 설정하였으며, 이 모델에 대한 가정을 서술하면 다음과 같다.

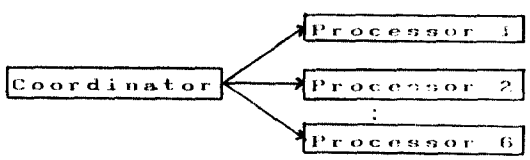

그림1. 데이터 흐름기 성능평가 모델

모델을 위한 가정

ㄱ. Coordinator는 입력된 작업을 순차적으로 프로세서에게 할당한다.

ㄴ. 입력된 작업을 처리하는 프로세서 수는 최소한 1개 이상이어야 한다.

ㄷ. 각 프로세서는 처리한 작업 결과를 다시 Coordinator에게 보낸다.

ㄹ. 각 프로세서는 할당된 작업을 병렬로 처리할 수 있다.

ㅁ. Coordinator에서 프로세서로 할당하는 작업은 이미 데이터 흐름 그래프로 변환 되어져 있다고 가정한다.

상기와 같은 가정에 따른 모델을 정립하기 위해 1) 구성요소(Components), 2) 기술변수(Descriptive Variables), 3) 구성요소들 간의 상호작용(Component Interactions)등의 파라미터로 구분하여 표현하는 비형식적인 모형기술 방법[11]으로 설명하면 다음과 같다.

1) 구성요소(Componets)
Coordinator, Processor, Job, Control Information

2) 기술변수(Descriptive Variables)
Coordinator : Coord-Flag
Processors : Proc-Id, Proc-Flag, Proc-Time
Job : Origin, Destin
Control Information : Origin, Destin, Message
Parameters : Nproc, Empty-Loop, Ave-exe

3) 구성요소들 간의 상호작용(Component Interactions)

ㄱ. Coordinator는 각 프로세서의 실행상태 여부를 검사하여 프로세서의 수를 1개에서 6개까지 증가시키면서 작업을 할당한다.

ㄴ. 작업을 할당받은 각 프로세서는 작업을 처리한 후 다른 프로세서가 그 결과값을 원하는 경우 Coordinator로 다시 결과값을 전달하고 자기 자신은 대기상태에 있는다.

ㄷ. 작업을 할당받은 시뮬레이터들은 데이터 흐름기의 점화규칙(Firing Rule)에 의해 병렬처리될 수 있다.

(2) 성능평가 시뮬레이션

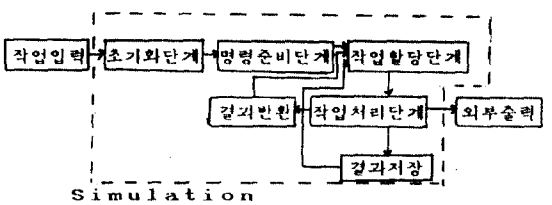

그림2. 성능평가 시뮬레이션 수행과정

성능평가 시뮬레이션 수행과정

1) 초기화 단계: Coordinator는 수행할 명령과 데이터를 매모리상에 준비하고 중간결과와 최종결과를 저장하기 위해 매모리영역을 준비한다.

2) 명령준비단계: Coordinator는 이미결정되어 있는 Dependence Graph에 따라 이용가능한 프로세서에 현재수행 가능한 명령을 할당할 준비를 한다.

3) 작업할당 단계: 이용가능한 프로세서에게 작업을 할당한다.

4) 작업처리 단계: 프로세서들은 할당받은 작업을 병렬로 처리한다.

5) 결과처리 단계: 작업의 처리 결과를 필요에 따라 Coordinator에게 되돌려 보내거나, 또는 잠시 저장하거나, 혹은 최종결과를 출력한다.

(3) 시뮬레이션의 결과분석

데이터 흐름기 성능 평가 모델에 입력하기 위한 기본적이며 간단한 작업은 다음의 식

X1 = (-B+SQRT(B**2-4*A*C))/(2*A)
X2 = (-B-SQRT(B**2-4*A*C))/(2*A)

으로 하였으며, 이 작업을 위해 이미 가정하여 주어진 Dependence Graph와 Data Flow Graph는 다음의 그림3, 그림4와 같다.

1) T1 : = -B
2) T2 : = B**2
3) T3 : = 4*A
4) T4 : = T3*C
5) T5 : = T2-T4
6) T6 : = SQRT(T5)
7) T7 : = T1+T6
8) T8 : = T1-T6
9) T9 : = 2*A
10) X1 : = T7/T9
11) X2 : = T8/T9

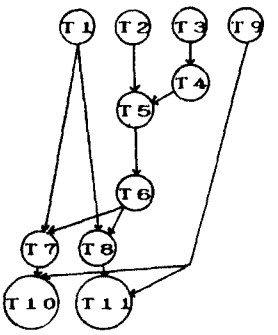

그림3 Dependence Graph

그림4 Data Flow Graph

상기와 같은 실행식 및 조건을 갖고 주어진 입력 작업에 대한 프로세서 수에 따른 작업의 평균 실행시간과 명령실행 단계를 고려한 데이터 흐름기 성능평가 시뮬레이션 수행결과를 보이면 다음의 표1과 같다.

표1. 프로세서 수의 증가에따른 작업의 평균 실행시간

프로세서 수	1	2	3	4	5	6
작업의 평균 실행시간	.214	.115	.118	.136	.117	.115

(단위 : Sec)

상기표1은 프로세서 수의 증가에 따른 작업의 평균 실행시간을 나타낸 것으로 프로세서 수가 3개까지 증가할 때는 작업의 평균 실행시간이 큰 폭으로 감소하지만 프로세서 수가 4개 이상일 때에는 작업의 평균 실행시간이 프로세서 수의 증가에 관계없이 거의 일정함을 알수있다.

표2. 프로세서 수의 증가에 따른 명령 실행단계

프로세서 수	1	2	3	4	5	6
명령실행단계	11	7	6	6	6	6

또한, 표2는 프로세서 수의 증가에 따른 명령실행 단계를 나타낸 것으로서 프로세서 수가 1개일때 명령의 실행단계는 11단계로, 프로세서 수가 2개일때 명령의 실행단계는 7단계로 나타났으며, 프로세서 수가 3개이상일때는 명령의 실행단계가 6단계로 동일함을 알 수있다 따라서, 성능평가 시뮬레이션 결과를 분석해보면 표1, 표2 에서 보듯이 주어진 간단한 입력작업에 대하여서는 프로세서 수가 3개일때 전체 시스템 효율이 높음을 알 수있다.

3. 데이터 흐름기를 위한 분산 시뮬레이터의 설계

앞장에서는 데이터 흐름기에 대한 모델링을 한후, 아주 간단한 작업(Job)을 Uniprocessor하에서 소프트웨이칭근 방법으로 시뮬레이션한 결과를 보였다

그러나 데이터 흐름기는 분산 처리시스템이므로 분산 시뮬레이션을 하여야 하며, 따라서 이를 위한 실제적인 분산 시뮬레이터에 대한 연구가 필요하다.

현재 이 분야에 대한 연구가 진행중에 있기 때문에 본장에서는 비록 초보적인 연구에 불과 하지만, 좀더 실제적인 데이터 흐름기 시뮬레이터를 설계하기 위하여 분산 시뮬레이션을 위한 분산모델을 제시하고, 이에 따른 분산 시뮬레이터를 설계하였다.

(1) 분산 시뮬레이션을 위한 분산모델

시뮬레이터의 구조에는 단일 프로세서와 다중 프로세서 두 범주로 나눌 수 있다.

첫번째 범주로서, 단일 프로세서 구조하에서의 분산 시뮬레이터는 이 프로세서 상에서 실행되는 소프트웨어 접근방법으로도 설계가 가능하다.

따라서 이환경 하의 시뮬레이터 구조는 소프트웨어 구조로서가상적인 상호 통신을 위한 프로시듀어로 구성된다 이 범주의 예로서는 Melman과 Livny의 "DISS"와 Wyatt의 "SIMPAS"가있다.

두번째 범주로서 다중프로세서 하에서의 분산시뮬레이터 구조에는 하드웨어 구현방법에 따라 부분 모델링 (Mapping of Submodels)과 함수언어법 (Functional, Language Based)이있다. 함수 언어법은 다양한 처리장치가 사건 리스트 관리 및탐색, 통계자료 수집, 그리고 난수 생성과 같은 시뮬레이션 언어의 특정한 기능을 수행하도록 되어있다. 여기에서 시뮬레이션 언어의 기능들은 모든 기능에 대한 Concurrent한 실행을 위하여 각기 다른 프로세싱 장치들에 분산된다. 함수 언어법의 사용에는 Barel의 "DESC"와 Wyatt의 "계층적 구조"가 있다. 부분 모델링에서는 다양한 부분 모델 들이 다중 처리기로 구성된 네트워크에 사상(Mapping)되는데 일대일 대응일때 이상적이다. 각 처리기들은 사상된 부분 모델들의 명령을 각각 수행한다. 이때의 하드웨어 구조는 Enslow가 분류한 Shared Bus, Cross-bar Switch, Multi-Bus, Multi-Ported Memories 등 으로 나눌수 있다 [2, 3, 4].

(2) 분산 시뮬레이터 설계

다음 그림5에 분산 시뮬레이터의 설계과정을 도시하였다.

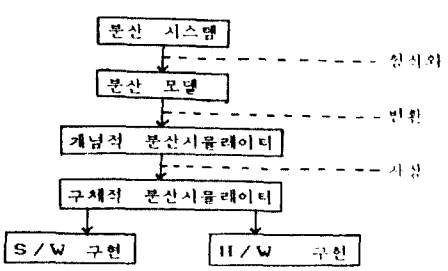

그림5. 분산 시뮬레이터 설계과정

4. 데이터 흐름기 시뮬레이터(DFMS) 구현

3장에서 설명한 바와같이 다중 프로세서 하에서의 데이터 흐름기 분산 시뮬레이터를 비록 초보적인 단계이지만 실제적으로 구현하고자 8-Bit 마이크로 컴퓨터 3대를 인접하여 데이터 흐름기 시뮬레이터를 구현하였다 Coordinator와 Simulator들간의 통신은 Asynchronous Serial Interface 상치를 이용하였다[12].

(1) DFMS 구조

본 논문에서 구현한 DFMS의 구조는 다음과 같다. 다음의 그림6은 하나의 Coordinator와 두개의 Simulator가 데이터 흐름의 원리에 의해 주어진 작업을 처리하도록 되어있는 DFMS의 구조를 보인 것이다.

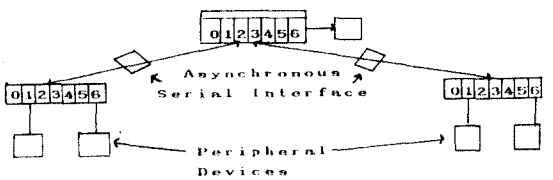

그림6. 데이터 흐름기 시뮬레이터 구조

(2) DFMS에 의한 분산 시뮬레이션

DFMS에의한 분산 시뮬레이션을 수행하여 결과를 얻기까지의 과정을 보이면 그림7과 같다.

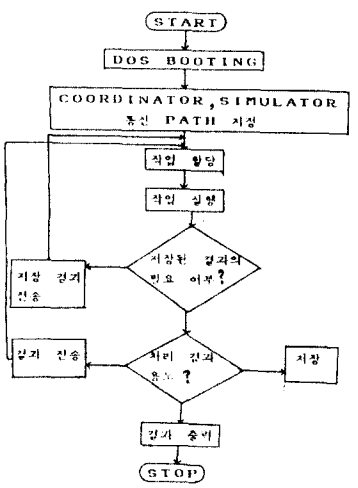

그림7. 분산 시뮬레이션 과정

분산 시뮬레이터의 처리과정

ㄱ. Coordinator에 Dos를 Booting 한다.

ㄴ. Coordinator와 Simulator 간의 통신 Path를 지정한다.

ㄷ. Coordinator는 작업을 각 Simuator에게 할당한다.

ㄹ. 각 Simulator는 할당받은 작업을 처리한다
ㅁ. Coordinator는 Simulator에게 저장된 작업 결과의 필요 여부를 알려준다.
ㅂ. Coordinator는 Simuator에서 처리한 작업 결과의 용도에 따라 작업 결과를 시뮬레이터에 저장하게 하거나 또는 되돌려 받거나 혹은 결과를 출력하게 한다.

(3) 분산 시뮬레이션 적용예

상기의 그림6과 같이 구현된 데이터 흐름기 시뮬레이터 위에서 주어진 작업에 대한 분산 시뮬레이션의 수행결과를 Coordinator와 각 Simulator별로 구분하여 보이면 다음과 같다.

Coordinator

ㄱ. 작업을 받아들임
ㄴ. Sim1-Flag 와 Sim2-Flag 을 검사하여 작업을 보낸다.
ㄷ. 전송 에러 이면 다시 작업을 보낸다.
ㄹ. Sim1, Sim2 으로부터 작업의 결과를 받아들인다.
ㅁ. 받아들인 결과들을 합성하여 새로운 결과를 출력한다.
ㅂ. 새로운 작업이 들어오면 ㄱ~ㅁ과정을 반복한다.
ㅅ. 더 이상의 작업이 없으면 끝마친다.

Simulator1

ㄱ. 작업을 받고 Sim1-Flag을 "Lock"시킨다.
ㄴ. 전송 에러이면 작업을 재 요청하여 받아들인다.
ㄷ. 작업을 처리한다.
ㄹ. 처리 결과를 Coordinator에게 보낸다.
ㅁ. Sim1-Flag을 "Unlock"으로 하고 대기한다.
ㅂ. Coordinator로 부터 요청이 있으면 ㄱ~ㅁ과정을 반복한다.

Simulator2

앞의 Simulator1과 수행과정이 동일함

분산 시뮬레이션의 결과를 분석해보면 데이터 흐름기의 원리에 입각하여 Coordinator의 작업할당은 비교적 원활히 수행되었다. 그러나 작업 할당시에 Coordinator는 한 Simulator가 작업을 처리할때까지 기다린후, 다른 Simulator에게 작업을 할당해야 되기 때문에 많은 대기 시간이 생긴다.

또한 Simulator1과 Simulator2간의 통신은 반드시 Coordinator를 통해서 해야되므로 전송 지연시간으로 인한 시스템 효율이 저하된다.

따라서 이러한 문제점을 개선하기 위한 연구가 절실히 요망된다.

5. 결론

본 연구는 단일 프로세서 하에서 소프트웨어 접근 방법으로 분산 시뮬레이터를 설계하여 데이터 흐름기에 대한 성능평가 시뮬레이션을 수행하여 그 결과를 분석하였다.

실제로 아주 간단한 작업에로 2차 방정식의 근의 공식을 입력한 결과 프로세서 수의 증가에 따른 작업의 평균 실행 시간은 프로세서 수가 3개 일때 최소로 나타났으며, 명령 실행 단계를 보면 프로세서 수가 3개 이상일 때에는 실행 단계가 6단계로 일정함을 알 수 있었다.

또한, 비록 초보적인 연구에 불과 하지만 다중 프로세서 하에서의 좀 더 실제적인 데이터 흐름기 시뮬레이터를 설계하기 위하여 분산 시뮬레이터를 설계하였다. 이를 바탕으로하여 8-Bit 마이크로 컴퓨터 3대를 계층적 트리구조로서 인걸시켜 실제적인 데이터 흐름기 시뮬레이터를 구현하였으며 여기에 간단한 예를 적용시켜 분산 시뮬레이션 하였다.

현재 이 분야에 대한 연구가 진행 중에 있기 때문에 다중 프로세서 하에서의 Loosely Coupled 한 경우에는 데이터 흐름기 분산 시뮬레이터를 구현 하는데에 해결해야 할 문제점이 많이 남아있다. 또한, 데이터 흐름기 자체에 대해 연구해야 할 과제가 방대하다. 예컨대, 통신 인터페이스 장치에 대한 연구, 효율 적인 작업 스케쥴링방침 및 시스템 소프트웨어 등 앞으로 이 분야에 대한 많은 연구가 더욱 절실하다고 하겠다.

참고문헌

[1] Ackerman, W.B., "Data Flow Languages", IEEE. Comp., Feb., 1982
[2] Baik, D-K., "Performance Evaluation of Hierarchical Simulators: Distributed Model Transformations and Mappings". Ph.D. Dissertation, Dept. of Computer Science, Wayne State Univeraity, 1986
[3] Baik, D-K, and Zeigler, B.P., 1985b. "Performance Modelling and Simulation of Hierarchical Distributed Simulators", Technical Report, CSC-85-003, Dept. of Computer Science Wayne State Univ.
[4] Concepcion, A, I., 1985a. "Distributed Simulation on Multiprocessors: Specification, design and architecture, "technical Report, CSC-85-001, Ph D. Dissertation, Dept. of Computer Science Wayne State Univ.
[5] David, P. Misunas, "A Computer Architecture for Data-Flow Computation," MIT/LCS/TM-100, March. 1978
[6] Dennis, J.B., "Data Flow Supercomputer, "IEEE. Comp., Nov, pp. 48-56, 1980
[7] Goatelow, K.P., and Thomas, R.E., "Performance of a simulated Data Flow Computer, "IEEE. Trans. Computers, Vol. C-29, No. 10, pp. 905-919, Oct. 1980
[8] Hwang, K., and, Briges, F.A., "Computer Architecture and Parallel Processing", MCGraw-Hill, pp. 732-768, 1984
[9] Moldovan, D.I., "Modern Parallel Processing", Dept. of Electrical-Engineering Systems, Southern California univ.
[10] Watson, I., and Gurd, J.R., "A Practical Data Flow Computer", Computer, Vol. 15 No. 2, pp. 51~57, Feb., 1982
[11] Zeigler, B.P., "Theory of Modelling and Simulation", A Wiley-Interscienc Publication, pp. 3-25, 1976
[12] Model 7710 Asynchronous Serial Interface Owner's Manual, California Computer System, 1981.
[13] APPLE II Reference Manual, Apple Computer Inc., 1979
[14] 김강현, 백두권, 황종선, "분산 모델링과 분산 시뮬레이션에 관한 연구", 한국 정보과학회 학술 발표집, Vol. 14, No. 1, pp. 41~44, Apr., 1987
[15] 김강현, 백두권, 황종선, "분산 시뮬레이션의 분류법에 관한 연구, "고려대학교 교육대학원 교육논총 제 16, 17 합집, Aug., 1987
[16] 지수영, 조용환, 백두권, "데이터 흐름기 실계를 위한 분산 시뮬레이터에 관한 연구", 한국 정보과학회 '87 추계 학술 발표집, Vol. 14, No. 2, pp. 559~562, Oct., 1987