□ 特輯 : μ-processor應用

Multiprocessor 컴퓨터 시스템

金 明 桓
(韓國科學技術院 教授)

金 浩 聖
(韓國科學技術院 博士課程)

■ 차 례 ■

1 서 론

컴퓨터의 처리능력을 증가시키고 보다 복잡한 분야에 컴퓨터를 사용하기 위해서는 한개의 프로세서만으로는 그 한계가 있다. 오늘날의 급속히 발전하는 마이크로프로세서 기술은 프로세서의 가격을 급격히 하락시키고 집적도를 증가시킴으로써 Multiprocessor 시스템의 실현을 가능하게 하였다.

시스템 측면에서 컴퓨터의 성능은 동시실행(concurrent execution)의 개념을 이용하여 증대시킬 수 있는데 이것은 하나의 작업을 여러개의 프로세스로 분할하고 실시간 다중작업처리기(Real-time multitasking executive)를 이용하여 이들 프로세스를 schedule 하고 통제와 동기를 시킴으로써 구현된다. 하나의 시스템은 시분할에 의하여 가상적으로 여러개의 프로세스를 갖는 것으로 생각할 수 있으므로 실제로 여러개의 프로세서를 갖는 시스템과 구별없이 concurrency를 갖는다고 말한다. 실제적으로 다수의 프로세서가 있는 경우에 일을 분산시키는 방법에 따라 시스템 동작중에 행하는 dynamic 방법과 시스템 설계시에 정하는 static방법이 있다.

컴퓨터 시스템의 성격을 특징지어주는 것은 명령어(instruction)와 데이타이므로 이 들의 다중성에 의해 시스템을 분류하는 것은 매우 큰 의미를 갖는다.

Flynn[3]은 컴퓨터 시스템을 명령어와 데이타의 흐름에 따라 다음과 같이 4가지로 분류하였다.

i) SISD (Single Instruction Single Data Stream)
 von Neumann 컴퓨터로 순차적으로 명령어를 수행한다.

ii) SIMD (Single Instruction Multiple Data Stream)
 하나의 명령어에 의해 여러개의 데이타가 다수의 처리장치에서 동일한 연산을 한다.

iii) MISD (Multiple Instruction Single Data Stream)
 하나의 데이타에 여러종류의 연산이 차례로 가해지는 형태를 말한다. 예로써 pipeline 프로세서가 있다.

iv) MIMD (Multiple Instruction Multiple Data Stream)
 가장 일반적인 형태로 각자의 독자적인 제어를 갖는 프로세서들을 갖고 있다.

MISD형태의 시스템은 각 프로세서들이 결합된

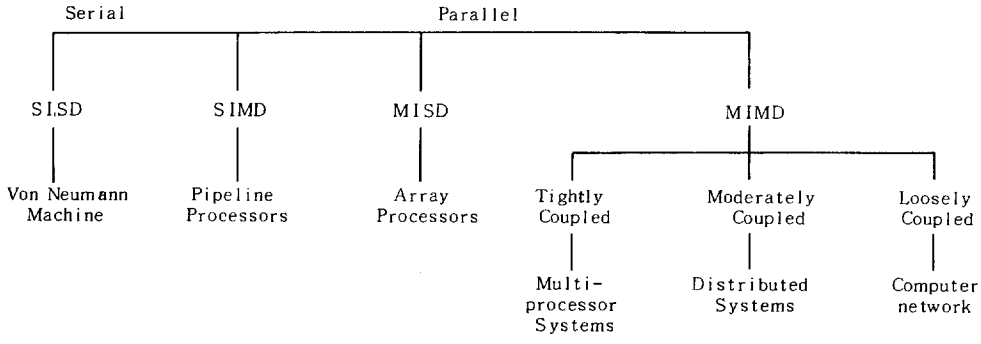

그림 1. 컴퓨터 시스템의 분류

정도에 따라 loosely coupled시스템, tightly coupled 시스템으로 나눌 수 있다.

(1) Loosely coupled 시스템

컴퓨터 네트워크으로도 불리우며 디스크 혹은 테이프등의 I/O를 공유한다.

이러한 시스템에서는 공유하는 장치를 통해 결합되나 본질적으로 프로그램 수행중에 직접적으로 결합되지는 않는다. 컴퓨터 네트워크는 다음과 같은 특징이 있다.

o독자적인 컴퓨터로 이루어 진다.

o통신 규약에 의한 직렬통신.

o네트워크는 통신을 위해서만 사용되고 실질적인 계산은 하나의 node에서만 이루어진다.

o모든 node를 access할 수 있다.

잘 알려져 있는 시스템 중에 ARPA network는 전송 대여폭이 50kbps이고 ETHER net은 3Mbps 정도이다. 이 시스템은 실제로 컴퓨터 사이의 거리가 먼 경우나 전송되는 데이타의 양이 많지 않은 경우에 많이 사용된다.

(2) 분산 시스템

Tightly coupled 시스템과 loosely coupled시스템의 중간형태로 상대적으로 독립적인 작업들로 일을 나누어 분산시킨 시스템이다. 이 구성의 특징은 다음과 같다.

o독자적인 처리장치

o프로세서들은 특정작업에 할당되어 있다.

o시스템의 배열은 대칭일 필요가 없다.

o프로세서 사이의 통신은 데이타가 주종을 이루고 명령어를 포함할 수도 있다.

oStatic load sharing

(3) Tightly coupled 시스템

Multiprocessor 시스템으로 불리우며 결합의 정도가 가장 큰 시스템이다. 미국 국가 표준연구소 (ANSI)에서는 multiprocessor 시스템을 종합된 통제 아래에서 여러개의 처리 장치를 갖고 있는 컴퓨터로 정의하고 있다. 종합된 통제란 말은 하나의 종합된 운영체제(operating system)를 갖고 있다는 것으로 multiprocessor 시스템이 요구하고 있는 가장 중요한 요구사항이다. 그러나 이 정의에는 공유와 상호작용(Sharing and interaction)의 개념이 빠져있어 multiprocessor를 정의하기에는 좀 부족하다. multiprocessor시스템은 다음과 같은 특징에서 컴퓨터 네트워크나 분산 시스템과는 그 성격을 달리한다.

ocommon memory : 시스템의 모든 프로세서는 주기억장치(primary memory)를 access할 수 있다.

ocommon OS : 하나의 OS가 모든 프로세서들간의 상호작용을 통제한다.

o자원의 공유 : 입출력 장치 혹은 다른 자원들을 공유할 수 있고 어떤 자원들은 특정 프로세서에 할당될 수도 있다.

o동등한 처리 능력 : 모든 프로세서들은 대칭적으로 배열되어 있고 비슷한 능력들을 보유하고 있다.

odynamic load sharing : 모든 프로세서들에게 동적으로 균일하게 부하를 분산시킨다.

oprocessor autonomy : 각각의 프로세서들은 중요한 계산을 단독적으로 실행할 수 있다.

o동기 : 연관된 프로세서들 사이의 동기(synchronization)가 필요하다.

다중 프로세서 시스템의 주된 한계점은 primary memory access 충돌 문제이다. 이러한 제약 때문에 하나의 OS에 의해 운영될 수 있는 프로세서의 갯수에 한계가 있다.

2 Multiprocessor의 구성

Multiprocessor 시스템 구성의 종류를 구분하는

중요한 요소는 프로세스-메모리의 구성을 형성하는 interconnection 시스템이다. 연결망(interconnection network)에 따른 분류가[5] 여러가지 있으나 여기서는 프로세스-메모리의 구성에 따라 multiprocessor의 구성을 time shared/common bus, Crossbar switch, multiport memory의 세가지로 구분하고 그 각각에 대하여 장단점을 설명한다.

2.1 Time-Shared/Common Bus시스템

이 시스템의 구성은 그림 3.과 같이 하나의 버스에 모든 functional unit들을 연결시켜 버스 인터페이스에 의하여 이들 unit들간의 데이타 전송을 한다. 이 방식은 가격이 저렴하고 구성이 간단하며 하나의 unit을 부가하거나 제거하기가 용이하다. 그러나 시스템의 성능이 버스의 전송속도에 의해 제한되며 버스의 고장이 전 시스템을 못쓰게 만든다. 버스의 전송속도를 높이기 위해 다중버스 시스템이 사용되나 이것은 시스템을 보다 복잡하게 하고 가격을 증가시킨다. 이러한 예로는 IBM STRETCH, CDC 6600등이 있다.

2.2 Crossbar Switch시스템

본 시스템의 구성은 그림 4와 같이 메모리 unit와

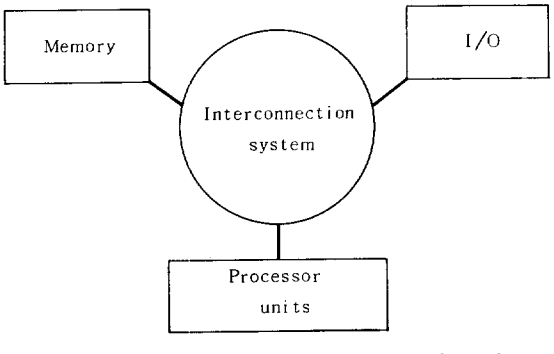

그림 2. 기본적인 multiprocessor의 구성

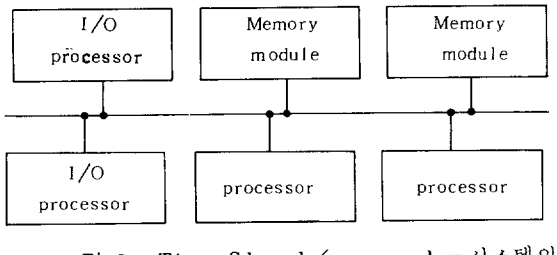

그림 3. Time Shared/common bus 시스템의 구성

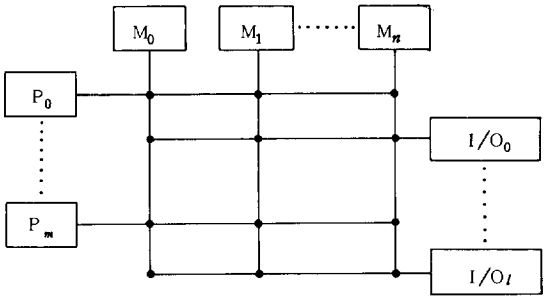

그림 4. Crossbar 스위치 시스템의 구성

프로세서 unit, I/O unit들이 격자 모양의 crossbar 스위치들로 연결되어 있다. 이 방식은 interconnection시스템을 매우 복잡하게 만드나 모든 스위칭 제어가 interconnection 시스템에서 실행되므로 각 functional unit들은 간단하며 데이타의 전송이 각 스위치를 통해 동시에 실행되므로 전송속도가 가장 빠르다. 그러나 연결시킬 unit들이 많아질 수록 스위치가 복잡해지므로 전체 시스템에 대해 스위치가 차지하는 비용의 비율이 높아진다. 앞으로 VLSI의 기술이 발전되어 crossbar 스위치를 쉽게 구현하면 시스템의 확장성, 신뢰성등의 장점이 있어 multiprocessor에 가장 적합한 방식으로 볼 수 있다.

2.3 Multiport메모리

이 방식은 시스템의 성격상, 속도가 메모리 access 시간에 의해서 제한될때 유용한 것으로 스위칭이 메모리 모듈에 집중되어 각 프로세서는 모든 메모리 모듈에 독자적인 버스를 갖고 있으며 메모리 충돌 문제는 각 메모리 part에 할당되어 있는 hard-wired fixed priority에 의해 해결한다.

대부분의 스위칭 로직이 메모리 모듈에 포함되어 있어 메모리 모듈의 가격이 비싸고 많은 수의 케이블과 연결기가 필요하며 시스템의 구성이 메모리 part 의 갯수에 의해 제약을 받게된다.

3 Multiprocessor의 운영체제(OS)

Multiprocessor의 하드웨어가 갖고 있는 parallelism의 특성을 잘 이용하기 위해서는 OS(operating system)의 역할이 매우 중요하다. 개념적으로 일반 컴퓨터의 OS에서 요구하는 것과는 큰 차이가 없지만 multiprocessor 시스템에서는 다음과 같은 능력을 요구하고 있다.

○자원의 분배와 운영

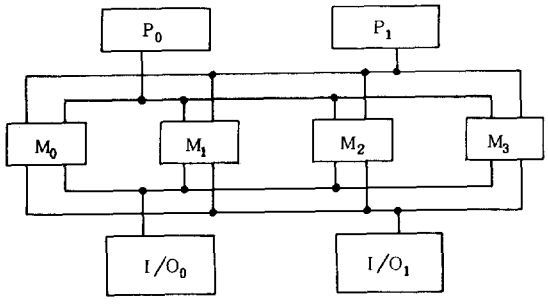

그림 5. Multiport 메모리 시스템의 구성

o테이블과 데이타의 set의 보호
o시스템의 deadlock 방지
o비정상 종결
oI/O 부하의 균등성
o재배치성 (reconfigulation)

이들 특징들 중에 부하의 균등성이 multiprocessor 시스템에서는 매우 중요하여 부가되는 자원을 어떻게 효율적으로 이용하는가의 문제가 multiprocessor OS의 중요한 촛점이 된다. 또한 배분된 작업을 동시에 실행시킬 수 있는 명령어, 예를 들면 DO PARALLEL, IF ALL, IF ANY등의 concurrent 언어가 있어야 한다. 여러개의 명령을 동시에 수행하는 경우에 프로세서의 동기 문제가 항상 따르게 마련인데 이것을 해결하기 위해 제안된 방법으로는 semaphore의 사용, priority에 의한 scheduling 등이 있다.

Multiprocessor 시스템에서 프로세서 수에 대한 성능의 특성은 매우 중요한 측면이다. 일반적으로 완전히 균일한 병렬수행은 얻어질 수 없으며 프로세서의 수가 늘어감에 따라 메모리의 충돌이나 동기등의 overhead가 지배적이 되어 오히려 성능이 감소하는 경우도 생긴다. 따라서 거의 균일한 병렬수행을 얻기 위해서는 언어나 OS의 수정은 불가피한 요소이다. 또한 하드웨어의 확장성에 상응하여 소프트웨어도 그에 따르는 변화에 적응할 수 있도록 설계되어야 한다. multiprocessor의 OS는 다음과 같이 3가지로 분류할 수 있다.

3.1 Master-Slave OS

이 형태의 OS는 구현하기가 가장 용이하며 uniprocessor OS에서 쉽게 확장시킬 수 있다. executive는 항상 일정한 프로세서 즉 master에서 돌고 slave 프로세서들은 master에게 그 서어비스를 요구한다. master-slave OS는 master가 고장날 경우 시스템에 치명적인 영향을 미쳐 신뢰도가 낮고, slave 시스템이 노는 시간이 많으며, 시스템의 구성이 고정되어 있어 유연성이 없다.

그러나 작업의 부하가 잘 정의되어 있고 slave가 master보다 적은 능력을 갖고 있는 비대칭형 시스템의 경우 가장 효율적인 형태라 볼 수 있다.

3.2 분리형 OS

이 방식은 각각의 프로세서 마다 executive가 있어 프로세서 자신의 필요에 따라 executive가 서어비스하며 자신의 테이블을 소유하고 있다. 시스템 코드의 일부분은 reentrant하거나 복사본을 각각의 프로세스에 제공하여야 하며 각각의 OS는 다른 프로세서의 고장에 민감하지는 않으나 고장난 프로세서를 재동작시키기는 어렵다.

3.3 대칭형 OS

모든 프로세서와 자원들을 대칭적으로 다룬다는 것은 매우 힘든 일이다. 그러나 다음과 같은 많은이익을 가져올 수 있다. master가 한 프로세서에서 다른 프로세서로 자유롭게 움직일 수 있어 모든 형태의 자원에 균등하게 부하를 분배할 수 있고 서어비스의 충돌은 priority에 의해 막을 수 있다. 이 형태의 OS의 잠재적 이점은 자원의 효율적 이용과 몇개의 프로세서가 고장났을 경우 점차적으로 성능이 떨어질 뿐 시스템 자체가 정지하지는 않는다는 점이다.

4 앞으로의 전망

하나의 프로세서에 의한 컴퓨터 성능의 향상은 그 한계가 있기 때문에 multiprocessor 시스템의 개발이 불가피하게 되었다. multiprocessor 시스템은 처리 능력을 증가시킨 반면에 프로세서간의 동기와 통신의 overhead 때문에 성능이 이론적인 계산처럼 증가하지는 못한다. 따라서 여러가지 방법들이 대두되는데 그 추세를 보면, 프로세서의 가격이 급격히 하락함에 따라서 점점 더 큰 규모의 multiprocessor 시스템을 추구한다는 것이다. 이러한 예는 뉴욕대학의 NYU 초대형 컴퓨터 프로젝트[4]에서 볼 수 있는데 4,096개의 프로세서를 대략 65,000개의 chip을 사용하여 연결시킬 계획을 세우고 있다. 이러한 경향은 가격에 대한 성능의 비를 개선하고 모듈화와 신뢰도를 자연스럽게 제공하고 있다. 그러나 아직도 작업의 분배, 상호간섭, fault tolerence 등의 문제가 남아

있으며 아직까지 연구할 과제가 많이 있는 상태이다.

참고문헌

1) P. H. Enslow JR; Multiprocessors and parallel Processing, John Wiley & Sons, 1974.

2) Eli T. Fathi and Moshe Krieger; "Multiple Microprocessor System: What, Why and When," Computer, Vol. 16, No. 3, pp. 23-32, Mar. 1983.

3) M. J. Flynn; "Very High-Speed Computing Systems," Proc. of IEEE, Vol. 54, No. 12, pp. 1901-1909, Dec. 1966.

4) A. Gottlieb and R. Grishman; "The NYU ultracomputer-Designing an MIMD shared Memory Parallel Computer," IEEE Trans. Comp. Vol. C-32, No. 2, pp. 175-189, Feb. 1983.

5) G. A. Anderson and E. D. Jensen; "Computer Interconnection Structures; Taxonomy, characteristics and Examples, "Computing Surveys, Vol. 7, No. 4, pp. 197-213, Dec. 1975.