기술해설

고성능 컴퓨터

채 수 환
(*한국항공대 전자계산학과 부교수)

오늘날 고성능 컴퓨터는 과학, 공학, 국방, 의료 분야 등에 널리 이용되고 있으며 그 적용범위가 점차 확대되어가고 있다. 또한, 하위 레벨의 컴퓨터인 PC(Personal Computer)는 가격이 저하되고 성능은 향상되는 추세에 따라 보편화되어져가고 있으며 그 적용범위도 확대되어 우리 생활에 직접 영향을 미치고 있다. 이리하여 '21세기는 정보화사회가 될 것이다'라는 말에 누구나 수긍하게 되었다. 이는 1946년도에 최초의 범용 전자식 컴퓨터, ENIAC이 완성된 지, 불과 반세기만의 획기적인 변화이다. 이런 변화를 가져올 수 있게 된 동기는 컴퓨터에 관련된 하드웨어, 소프트웨어, 시스템 기술의 급속한 발전에 있다.

컴퓨터 시스템의 속도만을 생각했을 때, ENIAC이 하나의 덧셈을 하는 데, 약 3ms가 소요되었는데 반해 1988년에 발표한 Cray Y-MP/832의 경우 최대처리 속도가 2G FLOPS (FLoating-point Operations Per Second)를 넘는다. 현재 CRAY, nCUBE, INTEL, Hidachi, NEC와 같은 회사에서는 100Giga FLOPS가 넘는 속도를 가진 수퍼컴퓨터의 개발 및 상용화를 서두르고 있으며 금세기 내에 Tera FLOPS를 넘는 컴퓨터시스템이 개발될 것이다.

컴퓨터의 이와 같은 발전에도 불구하고 사용자의 욕구는 더욱 증가하여 컴퓨터의 능력이 사용자의 욕구를 충족시키지 못하고 있다. 아무튼, 컴퓨터의 성능을 향상시키기 위한 연구가 활발하게 이루어지고 있다.

주요대상은 다음과 같다.

(1) 캐쉬메모리(cache memory)
(2) 파이프라인 기법(Pipelining)
(3) 다수의 프로세서 이용(병렬처리)

(1)과 (2)는 기존 순차형 컴퓨터시스템의 성능향상을 위해 사용된 반면에, (3)은 단일프로세서에 의한 성능향상의 한계에 대한 돌파구로서 채택되었고 장래의 주된 연구과제이다. 물론 (1)과 (2)가 (3)에도 사용된다.

여기에서는 고성능 컴퓨터의 실현을 위해서 사용되는 이들 3가지에 대해서 살펴보고자 한다.

1. 캐쉬메모리(Cache Memory)

캐쉬 메모리의 개념은 1965년 Wilkes[1]의 논문에서 두 개의 메모리 계층구조(하나는 일반적인 대용량 메모리, 다른 하나는 고속의 소용량 메모리)를 제안하고 고속의 메모리를 slave memory라고 불렀는데, 이것이 오늘날 캐쉬 메모리이다. 이 캐쉬 메모리를 처음으로 채택한 컴퓨터는 IBM-360/35이고 이후에 보편적으로 사용하게 되었다[2].

오늘날 대부분의 컴퓨터는 그림 1과 같은 메모리 계층구조를 가지고 있는데 이런 계층구조를 갖는 목적은 경제적인 메모리 시스템의 구현에 있다. 다시 말하면, 가능하면 메모리 접근속도(access speed)는 캐쉬메모리의 접근속도에 가까우면서 메모리

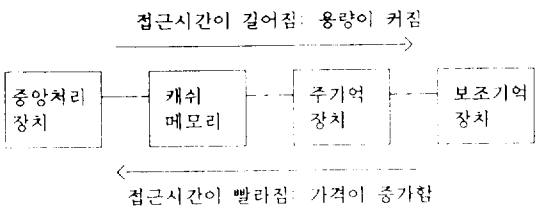

그림 1. 계층구조의 메모리 시스템

시스템 가격은 저렴하게 하려는 것이다. 가격과 성능은 trade-off관계에 있지만, 프로그램 수행이 가지는 특성과 캐쉬메모리의 고속성을 이용하면, 메모리시스템의 성능향상이 가능하다.

캐쉬메모리는 소용량이지만 그 접근속도가 주기억장치보다 5-10배 정도 빠르다. 따라서, 프로세서가 다음에 접근할 가능성이 높은 내용들을 고속의 캐쉬메모리에 옮겨 놓아 메모리 접근시간을 단축시킬 수가 있을 것이다. 즉 프로세서는 원하는 정보를 메모리로부터 얻기 위해서 먼저 캐쉬메모리에 원하는 내용이 있는지 확인하고 만약에 있으면(hit 되었으면) 바로 캐쉬메모리로부터 내용을 읽어내고, 없으면(miss 되었으면) 주메모리(main memory)로부터 읽는 방법을 사용한다. 이 때에 캐쉬메모리 시스템의 성능향상은 hit ratio에 의해 좌우된다. Hit ratio를 향상시키기 위해 큰 용량의 캐쉬메모리를 사용할 수 있지만, 비용과 관련이 있기 때문에 적정한 크기에서 정해진다. 아무튼, 반도체기술의 발전에 따라 주 메모리의 용량이 커지므로 캐쉬메모리의 크기도 점차 커지는 경향이다.

캐쉬 메모리의 유용성에 대한 근거는 가상메모리(virtual memory)와 같이 프로그램 수행 상에서 발생하는 순차성과 메모리 참조주소(reference address)의 집중성에 있다. 이를 locality 원리라고 하는데 locality는 크게 공간적인(spatial) locality와 시간적인(temporal) locality로 구분한다. 공간적인 locality는 프로그램 수행이 많은 순차성을 지니고 있으며 서로 관련된 데이타들을 같이 모아 선언하는 프로그래밍 경향에 따라 현재 참조된 부근의 메모리가 장래에 참조될 확률이 높다는 점에 근거를 하고 있다. 따라서 현재 참조된 메모리를 주기억장치로부터 캐쉬로 읽어갈 때, 다음 주소에 있는 일정한 크기의 메모리를 같이 읽어 캐쉬메모리에 저장해두면, 장래에 프로세서가 원하는 내용을 저

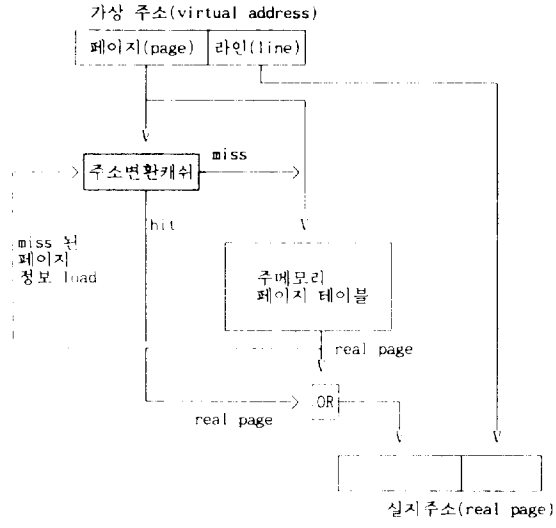

그림 2. 주소변환 캐쉬의 이용과정

속의 주기억장치로부터 읽지 않고 고속의 캐쉬메모리로부터 바로 읽어낼 수 있을 것이다. 시간적인 locality는 프로그램의 수행 시에 스택(stack)이 자주 이용되고 반복적인 수행을 위해 루우프(loop)의 사용이 많다. 따라서 한 번 참조된 메모리 내용은 머지않은 장래에 다시 참조될 확률이 높다는데 그 근거로 하고 있다. 한 번 참조된 내용을 얼마동안 캐쉬메모리에 저장해 놓는 것은 메모리의 접근시간을 단축시킬 수 있는 가능성이 클 것이다.

캐쉬메모리는 주소변환캐쉬(address translation cache)로 이용되고 있다. 가상메모리(virtual memory)시스템은 컴파일(compile)되었을 때에 얻은 가상주소(virtual address) 주소와 실지로 주메모리에 로드(load)될 주소가 일치하지 않기 때문에 프로그램을 수행할 때, 가상주소로부터 실지주소를 얻는 과정이 필요하다. 이 과정은 하나의 프로그램 명령이 수행될 때마다 최소한 한 번 이상 필요하기 때문에 주소변환 시간을 단축시키기 위해 캐쉬메모리가 사용된다. 즉 주메모리에 있는 주소변환테이블(address mapping table)중에서 장래에 사용될 확률이 높은 일부를 주소변환 캐쉬에 넣어둔다. 페이징 메모리(paging memory)개념을 이용한 가상메모리 시스템에서 주소변환 캐쉬의 이용과정을 나타내면 그림 2와 같다[3].

2. 파이프라인 기법(Pipelining)

파이프라인 기법은 자동생산공정에서 널리 사용되고 있는 방법으로 하나의 일(old task)이 끝나기 전에 새로운 일(new task)을 시작하므로서 전체적인 공정시간을 줄이는데 목적이 있다. 그림 3에서 보인 바와 같이 하나의 일을 N개의 과정(step)으로 나누어 수행시킬 수가 있다[4]. 이렇게 시간적인 중첩성을 이용하여 동시에 여러 개의 일을 동시에 처리하는 것을 시간적 병렬성(temporal parallelism)이라고 부른다[5]. 파이프라인을 이용한 일의 수가 과정의 수, N에 비해 훨씬 크다면, 이론적으로는 수행시간은 거의 N배가 단축된다.

지금까지 주류를 이루어온 von Neumann형 컴퓨터에서 프로그램 수행과정이나 벡터(vector)연산과정에서는 순차성이 많기 때문에 파이프라인 기법을 이용하여 컴퓨터의 성능을 크게 향상시킬 수가 있다. 이 기법을 1960년대 초부터 컴퓨터에서 IBM, CDC 등에서 수퍼컴퓨터 구조 설계에 도입하기 시작한 후에 지금은 PC에 이르기 까지 거의 모든 프로세서 설계에 사용되고 있다.

컴퓨터의 명령의 수행은 (1)명령의 fetch, (2)명령의 decode, (3)operand의 주소 생성, (4)operand의 fetch, (5)명령의 수행, (6)수행 결과의 저장, (7)PC(program counter)의 update 등의 과정을 순차적으로 반복된다. 따라서, 하나의 명령이 수행되는 동안 그 다음에 있는 명령들이 과정이 중복되지 않는 범위내에서 동시에 수행이 가능하다. 이를 명령 파이프라인 기법(instruction pipelining)

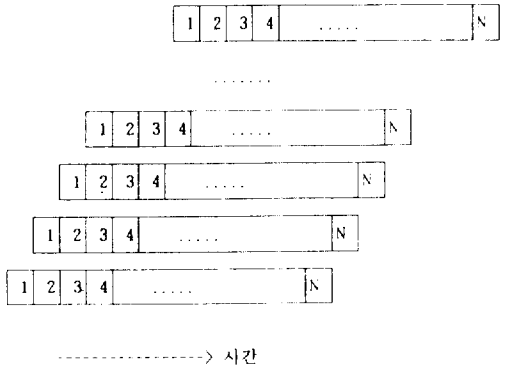

그림 3. N개의 과정으로 나누어진 공정의 수행과정

이라고 부른다. 물론, 이 때에 프로그램 수행이 순차적으로 이루어지지 않을 경우에는 파이프라인 내에 있는 명령들의 수행이 더 이상의 의미를 잃는다.

벡터나 행렬의 연산은 대량의 지속적인 순차성이 보장되기 때문에 파이프라인기법의 좋은 응용분야이다. 이 파이프라인 기법을 이용하여 벡터 및 행렬 연산의 성능향상을 위해 설계된 프로세서를 벡터프로세서라고 부른다. 벡터프로세서는 여러 개의 파이프라인된 기능장치를 이용하는데 Floating point 가산기의 파이프라인된 과정을 예를 들면 다음과 같다. (1)두 개의 실수 값의 입력, (2)두 실수의 지수 값의 비교, (3)작은 지수 값을 가진 실수의 조정(alignment), (4)가산, (5)정규화(normalization), (6)출력.

벡터 프로세서가 1970년대 초에 상용화된 TI-ASC, CDC Star-100 등에서 사용된 후에 Cray-1, Cyber-205, VP-200 등 수퍼컴퓨터에 널리 쓰이고 있고 다중프로세서를 이용한 대부분의 수퍼컴퓨터에서도 사용되고 있다. 1980년대 초에 제안된 Carnegie Mellon 대학의 시스톨릭(systolic array) 구조도 다수의 프로세서를 이용하여 파이프라인화한 형태이다.

3. 병렬처리(Parallel Processing)

1980년대 후반부터 수퍼컴퓨터 제조회사에서는 단일 프로세서에 의한 컴퓨터시스템의 성능 향상에 한계가 있음을 인식하고 많은 프로세서를 사용하는 (spatial parallelism을 이용하는) 병렬처리컴퓨터 개발에 주력하고 있는 추세이며, 병렬처리 기술의 보편화 및 고성능화가 추진되면서 그 적용범위가 수퍼컴퓨터 뿐 만 아니라, PC에 이르기까지 확대되어 가고 있다.

병렬컴퓨터의 종류를 구분하는 방법이 기준에 따라 여러가지가 있는데 그 중에서 일반적으로 제어방식에 따라 SIMD(Single Instruction stream Multiple Data stream)과 MIMD(Multiple Instruction stream Multiple Data stream)으로 분류하는 방법[6]이 가장 보편적으로 사용되고 있다. 그림 4에 전형적인 SIMD와 MIMD의 구조를 나타냈다[7].

SIMD컴퓨터는 하나의 중앙제어장치와 다수의

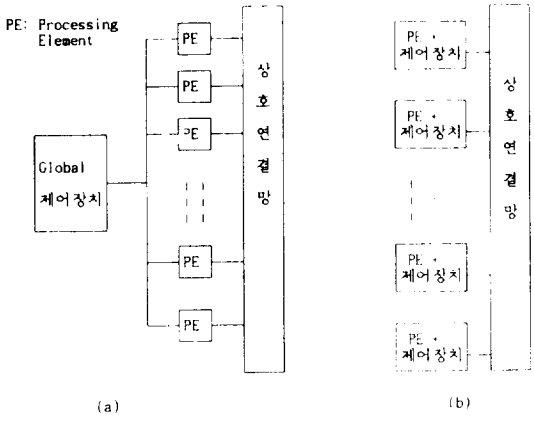

그림 4. SIMD구조(a), MIMD 구조(b)

표 1. SIMD 컴퓨터 발전과정

년 도	시스템	상 태	크 기
1950		von Neuman이 idea 제공	
1958		Unger이 제안	
1963	Solomon, Illiac III	기본연구	
1967	UCPR1	Prototype	20X20
1971	CLIP1	Prototype	10X10
1972	Illiac IV	prototype	8X8
	STARAN	prototype	256X1
1973	CLIP3	prototype	12X16
1976	DAP	prototype	32X32
1980	CLIP4	prototype	96X96
	DAP	commercial	64X64
1980	MPP	prototype	128X128
1985	CM1	commercial	65536
1990년대	AIS	commercial	1024
	DAP	commercial	4196
	MasPar	commercial	16384

프로세서, 하나의 상호연결망으로 구성되어진다. SIMD 컴퓨터는 프로세서의 연결이 배열구조를 가지고 있기 때문에 배열프로세서(array processor)라고 부르기도 하고 대규모의 병렬성을 가지기 때문에 MPC(Massively Parallel Computer)라고 하기도 한다. SIMD컴퓨터는 중앙제어장치에서 하나의 명령을 모든 프로세서에게 전달하면, 프로세서들은 주어진 데이타에 의해 명령을 수행한 후에 그 결과를 제어장치에 보내준다. SIMD컴퓨터의 발전과정은 표 1과 같다[8].

MIMD컴퓨터는 프로세서들이 각자 제어장치를 가지고 있어 독립적으로 명령을 수행할 수 있기 coarse grain한 병렬성을 구현하는데 적합하다. MIMD컴퓨터는 메모리의 구성에 따라 분산메모리 구조와 공유메모리 구조로 나눌 수가 있다[9].

분산메모리 구조에서는 각각의 프로세서가 자기의 메모리를 가지고 있으며 다른 프로세서가 직접 다른 메모리를 접근하지 못하고 메세지(message)를 통해 프로세서 간에 필요한 정보를 교환한다. 이때에 가장 중요한 것이 프로세서 간을 연결하는 상호연결망(interconnection network)이다. 즉 확장성, 연결성(connectivity), 망의 지름(diameter), 링크의 수 등을 고려하여 망구조(network topology)를 선택해야 한다.

Carnegie Mellon 대학교에서 1980년대 초에 만든 Cm*는 버스(bus)구조를 택했지만, 최근 Cosmic Cube, Ametek Series 2010, Intel Pesonal Supercomputer 등에서는 하이퍼 큐브(hypercube)에 의한 방식을 채택하였고 Tetradata의 DBC/1012는 2진트리(binary tree)를 채택하였다.

공유메모리 구조에서는 global 공유메모리를 가지고 있어 프로세서 간의 정보교환이 공유메모리를 통해 이루어지기 때문에 메세지의 교환이 필요 없다. 공유메모리 구조의 경우에는 메세지교환에 따른 통신상의 부하는 줄어들지만, 메모리 공유에 따른 동기문제 및 데이타 coherence 문제가 따른다. 따라서 공유할 데이타가 많은 경우에는 공유메모리 구조가 유리하고 적을 경우에는 분산메모리 구조가 적합하다. 물론 두 가지 메모리 구조를 같이 가지고 있는 컴퓨터시스템이 있는데, 이는 비용을 고려치 않을 경우에 어느 경우에나 다 적용할 수 있을 것이다.

1980년대 공유메모리 구조를 가진 상용화된 컴퓨터는 Flexible Corp. 의 Flex'32와 Encore사의 Multimax등이 있고[9], 1990년대에는 Kendall Square Research의 KRS1이 있다[10].

이외에 fine grain 컴퓨터로서 데이타플로우 컴퓨터와 시스토릭 어레이 컴퓨터를 들 수가 있는데, 데이타플로우 컴퓨터의 제어형태는 MIMD와 같으며 일반 MIMD 컴퓨터와 달리 data-driven이라는 점이 다르다. 시스톨릭 어레이는 위에서 언급한 바와 같이 프로세서와 프로세서 사이를 파이프라인으로 구성한 형태이며 이들 사이에 데이타의 흐름

은 동기적(synchronous)이다.

참 고 문 헌

[1] Wilkes, M. V., "Slave memories and dynamic storage allocation." IEEE Trans. on Electronic Computers, EC-14, no. 4, 1965, pp 270-271.

[2] Liptay, J. S., "Structural aspects of the system/360 Model 85-the cache," IBM Systems Journal. Vol. 7, No. 1, 1968, pp. 15-21.

[3] Barry Wilkinson, Computer Architecture, Prentice Hall, 1991.

[4] Harald S. Stone, High-Performance Computer, Addison-Wesley Publishing Company, Inc. 1993.

[5] Kai Hwang and Fayte A. Brtggs, Computer Architecture and Parallel Processing, McGraw-Hill, 1985.

[6] Flynn, M. J., "Some computer organization and their aspects," IEEE Trans. on Computers, Sept. 1972, pp. 948-960.

[7] V. Kumar, A. Grama, A. Gupta, and G. Karypis, Introduction to Parallel Computing, The Benjamin/Cummings Publishing Company, Inc., 1994.

[8] M. Maresca and T. J. Fountain, "Scanning the Issue-Massively Parallel Computers," Proc. of IEEE, April, 1991. pp. 395-402.

[9] Ralph Duncan, "A Survey of Parallel Computer Architectures," Computer, Feb., 1990, pp. 5-16.

[10] 김 기철, "상용 대규모 병렬처리 컴퓨터의 시스템 구조," 정보과학회지, 1994년 6월, pp. 8-21.

채수환(蔡洙煥)

1950년 10월 28일생. 1973년 한국항공대 전자공학과 졸업. 1977~83년 금성통신연구소 연구원. 1983~88년 미국 Univ. of Alabama 석사및 박사. 현재 한국항공대 전자계산학과 부교수. 주관심분야 : 병렬처리 및 분산처리 시스템.