차기 컴퓨터의 시스템에 관한 현황과 전망

이 근 철

제일전산훈련원 원장

◆ 머리말

최근 세계 유수의 컴퓨터 메이커들간에는 현존 최고의 프로세서라 일컫는 RISC(Reduced Instruction Set Computer) 칩에 만족하지 않고 새로운 형태의 프로세서 개발이 치열하게 전개되고 있다. ECI, CMOS 제조기술과 갈륨비소, 조셉슨 소자를 이용한 칩의 제작이 한창이며 64비트 전송폭과 병렬처리를 이용한 컴퓨터 개발이 빠른 속도로 비밀리에 추진되고 있다.

특히 병렬처리를 이용한 컴퓨터들이 시장에 등장, 무서운 능력의 하드웨어로 평가받고 있다.

세계 선두주자들간에는 병렬처리를 이용한 기종이 차세대 컴퓨터시장을 독주할 하드웨어 기술로 확실시된다는 판단하에 RISC 컴퓨터로 선두자리를 차지한 썬, 밉스, HP 등의 업체들과 그를 만회하려는 IBM, NCR 등의 업체들간에는 이 기술을 놓고 숨막히는 개발경쟁이 한창이다.

본고에서는 차세대 컴퓨터의 병렬처리시스템과 국내 현황에 대하여 기술하고자 한다. 병렬처리란 논리적인 하나의 프로그램을 여러 개의 작은 프로그램으로 분할, 다중처리하여 동시에 해결함으로써 프로그램의 처리속도를 크게 높여 주는 처리 방식이다. 즉, 만약 어떤 프로그램이 3개로 나누어질 수 있고 CPU가 3개 이상 있다면 나누어진 작은 프로그램들이 각각 한 개의 CPU에서 처리돼 CPU를 한 개만 갖고 있는 시스템보다 약 3배 빨리 일을 처리해 내는 방식이다. 즉 병렬처리는 하나의 프로그램을 여러 서브 테스크로 나누어 동시에 수행하는 것으로 작동시간을 줄이기 위한 싱글 테스크에서 기본 설계 개념을 둔 기계구조를 가지고 있다. 이 기계구조는 과학, 수학 엔지니어링 및 인공지능 등에 관한 일을 처리하기 위한 구조이다.

다중처리 방식은 서로 관계없는 많은 다른 프로그램이 있을 때 이들 각각의 프로그램을 각 CPU에서 나누어 여러 가지 프로그램을 처리함으로써 전체적인 결과를 빨리 얻어 낼 수 있는 방식이다. 그러므로 한 프로그램 입장에서 보면 CPU가 아무리 많아도 빨리 처리되지 않으나 여러 프로그램을 처리해야 하는 일(Job)의 입

장에서 보면 몇 배 빨리 해결되는 것이다.

따라서 현재는 회사에 따라 다중처리컴퓨터를 병렬처리컴퓨터라고 혼용해서 부르고 있으나 엄밀히 따지면 다중처리를 한다고 해서 병렬처리컴퓨터라 부를 수는 없다는 것이 전문가들의 견해이다. 최근 들어 이 개념이 점차 구체적으로 확립돼 가고 있는 추세이다.

현재, 일반적으로 이용되고 있는 컴퓨터는 순차처리(SISD)라는 방식에 따라 움직이고 있으며 이 방식에서는 입력된 프로그램에 따라 하나의 프로세서(연산처리장치)가 연산을 하나씩 해나간다. 그러나, 이 방식을 채용하는 한 컴퓨터의 처리속도는 한계가 있다. 그 까닭은 처리에 직접 관계가 되는 전자의 속도가 빛의 속도를 넘어설 수 없기 때문이다.

그래서 연산처리를 하나의 프로세서에게 시키는 것이 아니라, 복수의 프로세서에게 동시에 처리시키는 방법을 생각하게 되었다. 이렇게 하면 한꺼번에 많은 연산을 할 수 있어서 전체적인 처리속도를 빠르게 할 수 있다. 이것을 병렬처리라고 한다.

추론기능을 가진 제5세대 컴퓨터 개발에 병렬처리가 필요한 이유는 두 가지가 있다. 그 하나는 인간이 하는 추론이 병렬적이라고 생각되는 점이고, 또 하나는 추론을 하는데 있어서 대량으로 데이터를 짧은 시간안에 처리할 필요가 있다는 점이다.

병렬처리에서는 이미 'SIMD(Signal Instruction Stream and Multiple Data Stream)'라는 방식이 이용되고 있다. 이 방식은 대량의 데이터에 대하여 정해진 계산을 해가는 처리에 알맞다. SIMD는 크게 벡터 프로세서(Vector Processor) 방식과 어레이(Array) 프로세서 방식의 두 가지로 나눌 수 있다.

벡터 프로세서 방식에서는 처리내용을 4칙연산 수준에서 나누어 이것을 각각의 프로세서가 맡아서 처리한다. 각각의 프로세서는 맡은 연산만을 하고, 그 결과를 다음 프로세서에게 넘겨준다. 이 방식은 수퍼 컴퓨터로 불리는 고속 연산용 컴퓨터에 이용되고 있다.

어레이 프로세서 방식에서는 복수인 프로세서가 같은 연산을 동시에 한다. 이렇게 하면 일정한 시간안에 보다 많은 데이터를 처리하여 그 분량만큼 처리속도가 빨라질 것이다.

그러나, SIMD의 응용범위는 좁고 복잡하여 고도의 프로그램은 처리가 되지 않는다. 벡터 프로세서가 프로그램을 분할하여 처리한다고 하여도 분담할 수 있는 4칙연산과 같이 단순한 것밖에 없다. 어레이 프로세서 방식에서도 프로세서가 처리할 수 있는 내용은 단순한 연산에 한정되어 있다.

1. 병렬처리 머신의 구조

집을 짓고 있는 현장을 생각해 보기로 한다. 그리고 현장에서는 모든 일이 한 사람의 일꾼에게 맡겨져 있다고 생각해 보자.

그 일꾼은 일을 순서에 따라 차례차례로 진행시켜 갈 예정이다. 그는 예를 들면 벽돌을 쌓아 벽을 만들고 수도나 가스의 배관을 한다. 또는 전기배선을 하는 것과 같은 필요한 일을 적절한 순서로, 또 각각의 일의 적당한 때에 한 가지씩 마무리해 간다.

집을 짓는 일을 생각할 경우 이와 같은 방법은 순서대로 하기 때문에 옳은 일일 것 같지만, 자세히 생각해 보면 필요없이 일을 지연시키고 있다는 것을 깨닫게 된다.

벽돌을 쌓아 벽을 만드는 것같은 일에서는 몇 사람이 그 일을 나누어 동시에 하면 훨씬 빨리 끝낼 수 있다. 배선이나 배관같이 완전히 독립적이어서 복수의 팀이 일을 병행할 수 있는 일도 몇 가지가 있다.

한 사람이 한 번에 일을 하나만 처리해 간다고 하는 축차적인 느린 방법은 현재 이용되고 있는 대부분의 계산기의 동작과 아주 흡사하다. 대부분의 계산기는 2개 수의 덧셈, 곱셈, 비교 등의 연산을 하는 프로그램을 작성한다.

프로세서는 연산의 계열을 한 번에 1스텝씩 실행한다. 그런데 이러한 방법은 다음과 같은

두 가지 이유에서 본질적으로 느린 방법이라고 생각된다.

첫째 이유는 각 연산의 페이스(단계)에서 프로세서의 많은 부분이 아무런 일도 하지 않은 상태가 되어 프로세서의 잠재적인 능력을 충분히 발휘할 수 없다는 점이다. 예를 들면 2개의 수를 곱셈하는 경우를 생각해 보자. 이 경우 몇 스텝의 처리가 필요해지는데, 다른 시스템에서는 쓰이지 않는 회로가 있다.

두번째의 이유는 위와 같은 방법에서는 문제가 원래부터 지니고 있는 고속처리의 가능성을 이용하지 않고 있다는 점이다. 즉, 집을 짓는 건축과 마찬가지로 많은 계산으로 나누어 각기 별개의 프로세서를 병행시켜 진행할 수가 있다는 것인데, 그렇지 못하다는 것이다.

선진적인 계산기 아키텍처 구조 설계가들은 계산을 느리게 만들고 있는 문제를 극복하기 위해 다음과 같은 방식의 개발에 전력하고 있다. 그 하나는 다수의 프로세서를 결합시켜 병렬 프로세서, 병렬 계산기라고 불리는 방식을 설계하려고 하는 시도로서 이 시도는 일정한 시간내에 될 수 있는 한 많은 연산을 하는 것을 목표로 하고 있다.

그런데 어떻게 하면 프로세서의 속도를 향상시킬 수 있을까. 이제까지의 프로세서에서 계산을 느리게 하는 요인의 하나로 메모리 액세스(메모리의 판독이나 써넣기)의 문제가 있다.

즉, 어떤 연산을 하기 전에 데이터나 명령을 메모리에서 판독 즉, 읽어 내어 둘 필요가 있을 경우 프로세서의 모든 장치(계산의 각 스텝을 실행하는 논리회로)가 메모리를 판독할 수 있는 동안 할 일이 없어 쉬고 있는 상태가 되어 버린다.

이러한 상태를 아이들(Idle)이라고 하는데, 자원이 그대로 낭비되고 있는 상태이다. 연산을 끝냈을 때 연산 결과를 메모리에 격납하지 않으면 안될 경우에도 연산장치는 모두 아이들이 되어 버린다.

이 문제는 다음과 같이 설계함으로써 해결될 수 있다. 즉, 어떤 연산을 실행하는 동안에 다른 명령을 메모리에서 판독하여 해석해 놓거나 또는 여러 가지 연산장치에서 실행될 수 있는 연산으로 미리 분해시켜 둔다고 하는 기능을 갖춘 계산기를 설계하는 것이다.

연산을 실행하기 위해 복수의 데이터가 필요한 경우에도 메모리 액세스가 병목현상을 일으킨다. 예를 들면 메모리속에 2개 수의 곱셈을 생각해 보기로 하자. 곱수를 판독하기 위해 프로세서와 메모리간의 통신로가 전용되고 있는 동안은 피승수 즉, 곱해지는 수를 읽어낼 수가 없다.

이러한 문제는 한 번의 복수의 데이터를 메모리에서 읽어낼 수 있으면 해결된다. 그러면 이와 같은 일을 할 수 있는 메모리는 어떻게 설계하면 될까.

메모리 인터리브(Memory Interleave)라고 불리는 방식이 이같은 문제에 대한 한 가지 풀이를 제공해 준다. 이 방식은 수치의 배열, 그것도 큰 배열로, 배열의 요소가 잇따라 사용되어 가는 수치계산에서 특히 효과적이다.

메모리 인터리브(Memory Interleave) 방식을 끌어들인 전형적인 메모리는 소수(예를 들면 8 또는 16)의 각기 독립적으로 판독, 써넣기가 가능한 유닛으로 구성된다.

최초의 메모리 어드레스(즉, 최초의 연산에서 쓰는 수치가 격납되고 있는 장소)는 제1의 메모리 유닛 … 이라는 식으로 된다. 8유닛의 경우에는 9번째 어드레스가 제1의 유닛 내, 열번째의 어드레서는 제2의 유닛 내… 라는 식이 된다.

이와 같은 시스템에서는 독립된 통신로를 중계로 하여 동시에 복수의 메모리 유닛에 대해 액세스할 수 있다. 그렇게 하면 복수의 데이터를 동시에 사용하는 연산에서는 통신로가 비게 되는 것을 기다릴 필요없이 그들 데이터를 동시에 읽어낼 수가 있게 된다.

2. 파이프 라인

계산기의 진행방법 그 자체도 계산기의 속도

를 느리게 하고 있는 원인이다. 2개의 7자리수의 계산을 종이에 연필로 해본 경험이 있는 사람이면 당장 알 수가 있다고 생각되지만, 산수계산을 할 때는 작은 스텝이 여러 개 필요하다. 프로세서가 2개의 부동소수점의 수치(반드시 정수가 아닌 수치)의 곱셈을 할 경우를 생각해 보자. 우선 수치를 가수부(Exponent 로그의 가수)와 지수부(Mantissa : 멱지수)로 분할하지 않으면 안된다.

그 다음으로 지수부를 더해 합쳐 다수 가수부를 올바른 모양이 되도록 조정한다.

부동소수점의 곱셈을 하는 연산장치는 이들 각 처리를 각기 하는 세그먼트(Segment)로 분해할 수가 있다. 그렇게 하면 프로세서는 단순한 구성이 된다. 그리고 어떤 시점에서는 하나의 세그먼트만이 작동하고 그밖의 세그먼트는 모두 아이들되어 프로세서가 원래 지니고 있는 계산능력을 아주 낭비하는 결과가 된다.

이러한 문제를 해결하는 방법은 자동차의 조립라인과 많은 점에서 흡사한 방법이 있다. 조립 라인에서 매분 1대 정도의 속도로 차가 잇따라 제조된다. 몇천 명이나 되는 공원들이 전원 동시에 1대의 자동차에 달라 붙어 작업을 한다고 해도 이와 같은 재빠른 작업이 될 수는 없다.

이것은 자동차의 제조공정을 다수의 작은 공정으로 나누어 각 공정을 각각 별개의 작업장소에서 함으로써 비로서 실현될 수 있는 것이다.

자동차는 부분적으로 조립되어 가면서 각 작업소를 거쳐간다. 각 작업장소에는 기능공들이 트렁크의 뚜껑을 볼트로 조인다, 라디오를 끼워 놓는다고 하는 식으로 자기에게 할당된 일을 한다. 조립라인이 본질적으로 빠른 것은 이와 같은 작은 일들을 수없이 많이 되풀이 하고 있기 때문이다.

부동소수점 연산에 대해 조립라인을 만들 수도 있다. 즉, 파이프 라인이라고 불리는 방식이다. 이 방식은 같은 산술연산을 몇 번씩이나 되풀이하는 것과 같을 때에는 특히 그 효과가 크다. 예를 들어 2개씩 조가 된 수치가 다수 있고 각조에 대해 곱셈을 하는 것과 같은 프로그램을 생각해 보자.

여기서는 간단하기 때문에 부동소수점의 곱셈이 10개의 작은 부분으로 연산으로 분해될 수 있으며, 각 부분의 연산에 소요되는 시간은 계산기의 내장클록(Clock)를 1클록이라고 한다. 각 클록에서 새로운 수치의 조가 파이프 라인에 넣어져 최초의 부분연산이 이루어진다.

10클록이 끝나면 최초의 곱셈 결과가 완성되고, 두번째 조의 결과는 이미 곱되는 단계이며, 세번째 조는 다음 2단이 남아 있다. …, 11번째의 조는 파이프라인에 막 들어간 라인의 밖에서 기다리고 있는 상황이 된다.

파이프 라인을 이용하여 계산기의 연산을 얼마만큼 빨리 할 수 있게 하는가 하는 것은 파이프 라인을 어떻게 구성하는가에 달려 있다.

산술연산과 마찬가지로 데이터 전송이나 메모리로부터 데이터를 읽어 내는데 수 클록이 필요하다고 하자. 파이프 라인에서는 프로세서가 차례차례로 데이터를 읽어내기 때문에 각 클록으로 가동을 걸어 놓을 수가 있다. 이렇게 함으로써 메모리에서 최초의 데이터가 도착하는 것을 기다리지 않아도 다음 번의 데이터를 요구할 수 있다.

자동차의 조립라인을 가동시키고 난 다음의 최초의 완성차가 나오기까지에는 얼마간의 시간이 걸린다. 라인의 생산속도가 매분 1대라고 하더라도 어떤 특정한 자동차에 대해 눈여겨 보면 그 자동차가 라인에 들어간 다음 나올 때까지에는 며칠 정도인가가 경과되어 있을 것이다.

공정의 최초 개시시점에서 최초의 제품이 나오기까지의 시간을 파이프 라인의 레이튼시(Latency)라고 한다. 파이프 라인의 레이튼시는 파이프 라인의 공정수와 각 공정에서 소요되는 시간에 의해 결정된다. 레이튼시가 큰 파이프 라인은 처리하여야 할 계산의 양이 많을 경우에만 효율적으로 작동한다.

이것은 조립라인에서도 다수의 차를 제조할

경우에 효율적인 것과 꼭 같다(만일 조립라인을 가동시켜 자동차를 10대 정도밖에 만들지 않은 것과 같은 경우에는 그 차의 생산속도는 매분 1대보다 훨씬 느려진다).

3. 벡터 처리와 VLIW 머신

파이프 라인이 특히 적합한 것으로 벡터처리라고 하는 것이 있다. 벡터는 계산기 과학에서 쓰이는 용어로, 각 데이터간의 순서로 정해져 있는 수치 데이터의 배역을 뜻한다. 기하학적으로 개개의 수치로 좌표를 나타낼 수가 있기 때문에 이와 같은 배열이 벡터라고 불리고 있다. 즉 3개의 수치 배열로 3차원 공간에서의 위치나 방향을 나타내고 4개의 수치로 4차원 공간에서의 위치나 방향을 나타낸다고 하는 식이다.

계산기의 분야에서는 벡터는 수천 개의 요소를 지닐 수가 있고 기하학적인 해석을 필요로 하지 않는다.

하나의 벡터의 모든 요소에 같은 연산을 하는 경우가 종종 있다. 예를 들면 벡터 곱셈에서는 어떤 벡터의 제1요소와 다른 벡터의 제1요소를 곱한다. 제2요소끼리를 곱한다…고 하는 식의 계산을 한다. 이와 같은 계산은 파이프 라인에 적합하고 벡터연산을 라인화하도록 설계된 하드웨어는 선진적인 계산기 아키텍처의 하나의 기본이 되고 있다.

실제로 개발된 많은 시스템에서는 벡터 요소를 몇 개인가의 그룹으로 나누어 각 그룹을 프로세서 내의 복수의 연산장치로 동시에 처리하고 있다.

이와 같은 머신을 단일 프로세서 머신이라고 하는 것은 적합하지 않다. 이것은 축차형 머신과 병렬 프로세서의 중간단계에 있다고 할 수 있다. 마찬가지로 축차형 머신과 병렬 프로세서의 중간단계에 있는 머신은 가상적인 병렬 계산이라고 불리는 것으로 VLIW 머신(Very Long Instruction Word : 명령어의 길이가 아주 긴 계산기를 뜻한다)이 있다.

VLIW 머신의 프로세서는 질이 고른 연산장치를 복수로 갖추고 있다. 계산기의 각 실행 사이클에서 이제까지의 프로세서에서는 메모리부터 연산 명령을 하나만 읽어내는데 비해, 이 VLIW 머신에서는 복수의 연산을 동시에 읽어낸다. 이 때문에 몇 개인가의 연산을 그룹으로 모아 하나의 명령으로 프로세서가 메모리에서 읽어내는 단위로 하고 있다. 프로세서는 명령을 복수의 연산장치로 동시에 실행하고 어떤 의미에서는 병렬머신과 같이 행동한다.

일반적으로 어떤 연산이 다른 연산을 쓰려고 하였을 경우 어떤 종류의 충돌이 생기는 경우가 있다. VLIW 머신에서는 이러한 충돌을 피하기 위해 연산을 아주 조심스럽게 스케줄해 주지 않으면 안된다.

10 또는 그 이상의 연산을 동시에 실행할 수 있는 머신의 설계도 이루어져 왔다. 파이프 라인 및 벡터 아키텍처는 계산에 어느 정도 규칙성을 요구하기 때문에 불규칙적인 연산을 다룰 경우에는 힘을 충분히 발휘할 수가 없다.

이에 비해 VLIW 아키텍처는 원리적으로 불규칙적인 연산을 실행하는데 적합한 것이다.

이제까지 이야기해온 메모리 인터리브, 파이프라인, 벡터 프로세서, VLIW 머신과 같은 것은 어느 것이나 단일 프로세서의 속도와 효율을 향상시킨다고 하더라도 단일 프로세서로서는 전혀 도움이 되지 않는 경우도 있다.

예를 들어 양자색역학(양성자, 파이 중간자 등을 지배하는 기본적인 역학)에서 실제로 필요한 수치계산에서는 양성자 가까이의 4차원 시공간의 여러 가지 장에 적용하는 힘을 나타내는 수치를 1억개 정도 반복 계산한다. 계산속도를 극한까지 빨리 해가면 계산기 내부의 2점간을 신호가 통과하는 속도가 문제가 된다. 진공속이 광속은 신호가 이동할 수 있는 최고 속도인데, 1나노초당 약 30㎝이다(1나노초는 10억분의 1초).

따라서 가령 프로세서의 회로가 한없이 빨리 작동하고 나아가 물리적으로 소형으로 만들 수 있다고 하더라도, 단일 프로세서의 머신에서는

1초 동안에 수십억개 이상의 명령을 실행할 수 없다. 그리고 이러한 속도로는 양자색역학의 문제를 다루기에는 불충분하다. 이러한 종류의 응용에 이용하려고 하면 다수의 프로세서를 결합해 비로서 그 요구가 충족될 수 있는 것이다.

병렬머신의 설계에서는 기술자는 다수의 선택을 강요당한다. 예를 들면, 다음과 같은 선택이 필요하다. 프로세서를 몇 대로 하면 좋을까? 각 요소 프로세서를 어느 정도의 기능으로 하면 좋은가?

병렬 프로세서의 설계자가 직면하는 이같은 선택을 이해하기 위해서는 집을 짓는 경우를 생각해 여러 가지 문제를 유추해 보는 것이 좋다. 다시 한 번 집을 짓는 현장을 머리에 떠올려 보기로 하자. 다만, 이번에는 모든 작업을 건축회사가 도급으로 청부하였다고 생각한다.

이 경우에는 각 일꾼이 개개의 프로세서의 역할을 맡고, 건축회사의 담당그룹이 전체로서의 역할을 맡는다. 이 '계산기'의 특징은 어떤 것일까?

많은 일꾼이 각각 다른 일을 분담하여 동시에 일하는 것이므로 시스템은 분명히 병렬이지만 균일적이라고는 할 수 없다. 즉, 일꾼들은 기능이 다르면 다른 일을 분담한다(미장이, 전기공 … 등을 떠올리면 된다). 그러나 본질적으로 같은 기능을 지닌 일꾼을 여러 사람 고용하는 것이 가장 좋은 것같은 상황을 생각할 수도 있다.

4. 상호결합 시스템

병렬계산기에서는 다수의 프로세서를 다른 프로세서나 메모리 장치와 어떻게 접속하면 될까? 프로세서 상호간 및 프로세서와 메모리 장치간에 정보를 공유할 수 있게 하는 상호결합 시스템을 구성할 수 있느냐 하는 것이 병렬처리 시스템의 특성을 결정짓는 요인 중에서 가장 중요한 과제중의 하나이다. 공유 메모리라 불리는 이 방식은 상호결합 시스템의 구성방법 중의 하나이고, 이 방식에서 기초한 시스템이 이제까지 많이 제안되고 있다. 이 방식에서는 직접 접속한다.

결합망을 사용한다. 메모리 버스(Bus)를 사용한다는 것 등으로 전 프로세서를 공용 메모리로 접속하고 있다. 버스라는 것은 복수의 프로세서가 정보 요구를 송출하거나 메모리 장치가 프로세서에 데이터를 되돌리거나 하기 위한 통신로이다.

공유 메모리 시스템에서는 모든 프로세서로부터 모든 데이터의 격납 장소에 직접 액세스할 수가 있다. 프로세서가 자신의 처리를 위해 데이터가 필요해졌을 경우 그 데이터가 어디에 있건 메모리에서 간단하게 읽어 낼 수가 있다.

다른 프로세서가 끊임없이 격납되기 이전에, 메모리가 있는 장소에 격납되기 이전에 그 장소로부터 데이터를 읽어내려고 하는 프로세서가 생기지 않도록 주의하지 않으면 안된다. 이러한 문제를 피하기 위해 필요한 조정은 대부분의 경우 소프트웨어에서 이루어지지만, 하드웨어의 도움이 있으면 보다 좋게 실현된다.

예를 들면 어떤 시스템에서는 각 메모리셀에 값이 이미 써 넣어졌을지 어떤지를 나타내기 위한 영역을 메모리에 잡고 있다. 이 영역을 전용으로 사용해 각 메모리 셀에 데이터가 써 넣어지기 전에 읽어내려고 하는 너무 이른 조작이 생기는 것을 막고 있다.

그런데 메모리 방식이 지니고 있는 문제점은 다수의 프로세서가 동시에 액세스할 수 있는 메모리를 만들기 어렵다거나 또는 값이 비싸다고 하는 점이다. 예를 들어 1천개의 프로세서로 된 시스템에서 각 프로세서를 메모리와 직접 접속하였을 경우, 그리고 병목현상을 줄이기 위해 메모리가 1천 뱅크에 분할되어 있을 경우에는 1백만 선의 접속라인을 가지고 있는 시스템을 만들어야 한다.

공유 메모리를 설계할 때 생기는 이와 같은 문제는 출입구가 제한된 고속도로를 관리할 경우의 문제와 흡사하다. 속도제한과 차선수가 주어지면 각 고속도로는 매분 통행할 수 있는 자동차의 대수(용량)가 정해진다.

용량 이상의 대수의 차량의 흐름은 느려지고 정체가 생긴다. 이것을 푸는 한 가지 방법은 차선수를 늘린 고속도로(또는 몇 개의 병행하는 고속도로)를 만드는 것일 것이다.

그러나, 이러한 방법은 실제상으로는 실현되기 어렵기 때문에 제한이 따른다.

또 한 가지 해결 방법은 국제 전화망과 같은 통신 네트워크를 만드는 것이다. 이세상에는 수억대의 전화기가 있지만 거의 예외없이 온 세계의 전화기는 서로 접속할 수가 있다.

이 경우 전화기의 모든 조합에 대해 전화선이 가설되어 있는 것은 아니다. 전화번호를 돌려 접속요구가 나오면 그 요구는 지정된 전화기에 도달하기까지 여러 교환기를 통해 전파되어 간다.

네트워크 접속이 지니고 있는 결점은 읽어내기 요구가 메모리 장치에 도달하기까지는 네트워크의 몇 개 지점을 통과하지 않으면 안되기 때문에, 메모리에의 액세스 대기시간이 증대하는 것이다.

또 전화망과 마찬가지로 네트워크가 접속요구로 포화상태가 되어 아주 많이 늦어지는 것과 같은 피크가 주기적으로 발생할지도 모르는 점이다. 이렇게 되면 계산이 주기적으로 휴지화된다.

값을 적절하게 유지하고 메모리의 액세스를 높이면서 전체를 최소한으로 줄이도록 여러 가지 네트워크가 설계되어 왔다.

5. 국내 도입 병렬처리 컴퓨터 현황

최근 반도체의 발달로 마이크로 프로세서와 메모리의 성능이 우수해지고 더 빠르고 더 성능좋은 컴퓨터를 바라는 욕구를 최대한 결합해 만든 병렬처리 시스템에 대한 관심이 세계 각국에서 집중되고 있다.

얼마전까지만 해도 대학이나 연구소에서 실험용으로만 사용해오던 병렬처리 시스템들이 최근 들어 RISC와 갈륨비소반도체, 유닉스돌의 기술과 결합해 서서히 나타날 조짐이 보이기 시작한 것이다.

'80년대 중반부터 시장에 선보이기 시작한 병렬처리 시스템은 RISC 칩에 다중처리 시스템과 병렬처리 기능을 첨가시킨 것이 최근의 추세로 소리없이 유저들에게 접근하고 있다.

그러나 국내에서는 병렬처리 컴퓨터가 크게 소개돼 있지 않아 거의 대학이나 연구소에서 실험용으로만 사용되고 있을 뿐 일반인에게 큰 호응을 받고 있지 못한 형편이다. 또 국내에서 병렬처리 컴퓨터가 주장하고 있는 기종들도 어떤 의미에선 완전한 병렬처리 컴퓨터라 볼 수 없어 급속히 발전하고 있는 외국의 신기술을 쫓아 가려면 그 개념 정립부터가 시급하다는 것이 전문가들의 지적이다.

이에 따라 한국정보과학회에선 지난해 산하에 병렬처리 연구회를 설치하고 국내 병렬처리 시스템 시장의 활성화와 범용적 활용을 위한 연구를 꾀하는 등 세계 흐름에 부흥하려는 시도가

<표 1> 국내 병렬처리컴퓨터 기종 현황

공급업체	한국디지털	한국전산계산	대우엔지니어링	컨벡스코리아	쌍용컴퓨터	제철엔지니어링	한국컴퓨터
메 이 커	디 지 털 이퀴프먼트	얼라이언트	엘랙시	컨벡스	스퀀트	컨커런트	텐 덤
모 델 명	VAX9000/440	ALLIAT /FX2800	시스템 6400	C/240	Symmetry 81	3280 EMPS	Nonstop Cyclone
국내출하시기	'89년 7월	'90년 1월	'89년	'88년 11월	'89년 11월	'86년	'89년 12월
최 대 CPU 수	4	28	12	4	30	12	16
OS	VMS	Concentrix	EMBOS	CONVEX UNIX	DYNIX	OS/32	GUARDIAN
주기억용량	2GB	1GB	2GB	2GB	240MB	256MB	2GB

한창이다. 현재 국내 시장에 들어와 있는 병렬처리 컴퓨터라 불리고 있는 기종들은 얼라이언트 FX/2800, 엘렉시의 시스템 6400, 컨벡스의 C240, 스퀀트의 Symmetry 시리즈, 컨커런트의 328EMPS, DEC의 VAX9000 시리즈, 텐덤의 Nonstop Cyclone 등이다. 이들은 국내 대리점 및 현지 법인인 한국전자계산, 대우엔지니어링, 한국디지털, 한국컴퓨터 등을 통해 국내에 자사 특유의 처리기능을 갖고 판매되고 있다.

이 시스템들은 병렬처리를 위한 컴파일러를 갖고 있으며 처리속도를 높이기 위해 대부분 대칭구조를 갖고 있고 CPU를 최대한 활용할 수 있는 다이나믹 로드 밸런싱 기능(일의 부하를 조절하는 기능)을 보유하고 있다.

6. 병렬 시스템 발전 전망

오늘날 계산기를 잘 이용하고 있는 과학자들은 선진적인 아키텍처에 관해서도 실제적인 경험을 쌓고 있다. 그러나 경험은 아직은 제한된 것이다. SIMD 머신, 커넥션 머신, 캘리포니아 공과대학의 하이퍼 큐브나 메모리 머신과 같은 유망한 병렬 아키텍처는 광범위한 응용분야에서 연구되고, 그 유효성이 제시되고 있으며 앞으로의 발전가능성이 인정되고 있다.

한편 현재 입수할 수 있는 하드웨어나 소프트웨어는 아직 초보적인 것이다. 각종 머신의 경량적인 비교를 폭넓게 실시한 예가 거의 없다.

병렬 머신은 장차 아주 중요해지고 축차형 머신에 비해 훨씬 큰 성능을 약속해 주리라고 확신하고 있지만, 장치는 수없이 실용화되어 왔다. 단기적으로 보아도 새로운 하드웨어의 미래는 유망하다고 강한 확신을 가질 수 있다. 그러나 소프트웨어에 대해서는 그다지 낙관적이지 못하다. 중요하지만 매력이 결여된 이 분야에 대해 하드웨어와 마찬가지로 정력적으로 투자가 이루어져 왔다고는 생각되지 않는다.

오늘날의 축차형 스프트웨어의 기초를 구축하기 위해 3년 이상의 세월을 소비해 왔다. 교묘한 소프트웨어 기술이 개발되고 그 소프트웨어의 기반을 새로운 머신에도 옮길 수 있게 될는지도 모른다.

그러나 커다란 진전을 가져오게 하려면 실용화 및 연구분야의 기본적인 응용프로그램을 재구성하고 바꿔 쓰는 노력이 요청된다.

병렬 소프트웨어에 대해서도 어떤 종류의 표준을 확립하는 것이 장차의 발전을 위한 하나의 중대한 스텝이 될 것이다. FORTRAN, COBOL, LISP와 같은 언어는 과학기술, 비지니스, 인공지능과 같은 각 분야에 있어서 축차형 머신으로 하는 일의 기본적인 도구가 되고 있다.

이들은 분명히 불완전한 언어이기는 하지만 새로운 하드웨어가 만들어지면 여기에 맞는 프로그램을 바로 얻을 수 있는 축차형 소프트웨어의 기초를 구축하는 데 충분한 일을 해왔다.

그러나 병렬처리의 세계에는 소프트웨어의 표준에 관한 이해나 동의가 아직 얻어지지는 않고 있다.

한편, 새로운 기술의 발전은 아키텍처의 장래에 대해 의심할 여지가 없이 영향을 미칠 것이다. 반도체 기술의 진보, 그리고 새로운 고온 초전도에 기초를 둔 기술의 진보가 하드웨어 설계에 새로운 국면을 가져오리라는 것은 확실한 것이다.

빛으로 상호 결합된 고속 디지털 프로세서와 같은 하이브리드 머신이 틀림없이 나타날 것이다. 이와 같은 진전이 이루어진다고 하여도 우리가 여기에서 이야기해 온 계산기의 기초가 되는 모델은 여전히 적용될 것으로 생각된다.

순수한 광 계산기는 보다 더 혁신적이다. 그러나 계산기는 그보다 더 혁신적이다. 계산기는 오늘날 디지털로만 실현되고 있는 불편하면서도 시간을 낭비하는 방식이 되고 있는 여러 가지 알고리즘을 고속의 아날로그 계산하는 것을 가능하게 할 것이다. 이러한 광 계산기가 신호처리의 분야에 한정된 전용머신이 될는지, 또는 범용머신이 될는지는 현재로서는 예측할 수 없다.