**Memory Interleaving**

현재 프로세서 클럭이 1GHz를 돌파한 상황에서, 기존 프로세서 클럭 자체가 몇배 성능을 대변하지 못한다는 것과 일맥 상통하는 이야기이다. 실제 펜티엄 III 1GHz가 500MHz에 비해서 두배의 성능을 내지 못한다는 것은 삼척동자도 알만한 이야기인데, 이를 해결하기 위한 시스템 아키텍쳐적인 개혁이 프로세서의 발전만큼 이루어 져야한다는 것을 의미한다. 이를 위한 시도는 x86 프로세서의 진화와 더불어 시스템을 구성하는 칩셋, 그리고 메모리와 각종 컨트롤러의 진화로 이어져 왔다. 이중 최근 인텔의 정책과 맞물려 가장 이슈화가 되고 있는 것이라면 메모리 인터페이스를 꼽을 수 있겠다. 실제 시스템 전체 성능에 프로세서 클럭 만큼이나 민감하게 작용하는 부분이 메모리 클럭(클럭이라기보다 대역폭이라는 표현이 더 맞을 듯 하다)인데, 프로세서가 가진 성능을 최대한 이끌어 내기 위해서 전체 시스템 버스를 늘린다는 측면에서 메모리 대역폭의 증가는 필수 불가결한 것이라고 할 수 있겠다.

메모리의 대역폭을 높이기 위한 방법에는 세 가지 정도가 사용되고 있다. 대역폭은 버스의 폭(width)와 클럭 주파수의 곱으로 표현되는데 첫 번째로 메모리의 입출력 포트를 늘려 대역폭을 증가시키는 방법, 두 번째로 메모리 자체의 클럭을 높여 대역폭을 늘리는 방법, 그리고 마지막으로 메모리를 병렬화하여 접근하는 Interleaving방법이 있다.

메모리 인터리빙은 메모리를 구성하는 각 Bank에 연속된 어드레스를 지정하고 이를 순차적으로 잃어내는 파이프라인 개념에 접근한 방법이다. 이렇게 함으로서 메모리 자체를 고속화하지 않고 기존의 메모리로서 고속 메모리와 같이 대역폭을 늘리는 효과를 볼 수 있으며, 또한 SDRAM의 Precharge time을 상대적으로 줄이는 효과도 볼 수 있다.

아래 그림을 참조로 4뱅크로 구성된 메모리의 인터리빙의 개념을 알아 보도록 하자. 바로 아래그림의 경우 인터리빙을 하지 않는 경우이다. 메모리에서 데이터를 읽어오기 위해서는 순차적으로 각 뱅크에 요청한 데이터를 읽어오고, 각 뱅크를 다시 리프레쉬해야 한다. 즉 순차적으로 한 개의 메모리 읽기 작업이 마무리되기 전까지 기다려야 한다는 것이다.

아래 인터리빙 되지 않은 메모리 뱅크에서 데이터를 읽어오는 것을 대략적으로 정리 해보면 다음과 같다.

|  |
| --- |
| http://images.kbench.com:8080/korean/hardware/info/2001_02/interleaving/diag_noint.jpg |

1. SDRAM 리프레쉬

2.CPU가 어드레스#0을 보낸다.

3. CPU가 요청한 data #0을 받는다.

4. SDRAM 리프레쉬

5. CPU가 어드레스 #1을 보낸다.

6. CPU가 요청한 data#1을 보낸다.

7. SDRAM 리프레쉬

8. CPU가 어드레스#2을 보낸다.

9. CPU가 요청한 data #2을 받는다.

10. SDRAM 리프레쉬

11. CPU가 어드레스 #3을 보낸다.

12. CPU가 요청한 data#3 을 보낸다.

인터리빙을 하지 않을 경우 클럭 사이클의 낭비가 매우 커짐을 알 수 있다.

한편 4개의 뱅크로 구성된 메모리로 위와 같은 작업을 한다고 하면 다음과 같이 된다.

|  |
| --- |
| http://images.kbench.com:8080/korean/hardware/info/2001_02/interleaving/diag_int1.jpg |

1. CPU가 Bank#0에 어드레스 #0을 보낸다.

2. CPU가 어드레스 #1을 Bank#1에 보내고 Data#0을 Bank#0에서 받는다.

3. CPU가 어드레스 #2를 Bank#2에 보내고 Data#1을 Bank#1에서 받는다.

4. CPU가 어드레스 #3을 Bank#3에 보내고 Data#2을 Bank#2에서 받는다.

5. CPU가 Data#3를 Bank#3에서 받는다.

|  |
| --- |
| http://images.kbench.com:8080/korean/hardware/info/2001_02/interleaving/diag_int2.jpg |

즉 한 개의 뱅크가 리프레쉬를 하고 있을 경우 다른 뱅크는 액세스를 하여 순차적인 접근 방법보다 병렬화된, 파이프라인 접근 방법으로 클럭사이클을 줄이면서 데이터를 읽어와 전체적으로 대역폭이 늘어난 효과를 메모리 자체를 고속화 시키거나 입/출력 포트를 늘리지 않고 이루어 낼 수 있는 것이다. 물론 위의 예는 실제 메모리 인터리빙과 완전히 동일하지 않음을 밝힌다. 실제 DRAM의 RAS Access Time, Precharge time, 그리고 Cycle time이 위에서 대략적으로 제시한 것과 같지는 않기 때문이다.

단순한 예를 들어 계산으로 Access Time이 클럭 사이클의 절반, Precharge Time도 클럭 사이클의 절반으로 계산할 경우 , 4 way 뱅크 인터리빙으로 이론상으로는 기존 메모리를 고속화 시키지 않고도 대역폭은 4배가 늘어나는 효과를 볼 수 있게 된다.

그러나 이 뱅크 인터리빙의 경우 CPU가 요구하는 데이터 어드레스가 같은 뱅크에 위치할 경우에는 당연히 다시 리프레쉬를 거친후에 뱅크에서 데이터를 읽어와야 하기 때문에 항상 높은 대역폭을 얻을 수 있는 것은 아니다.

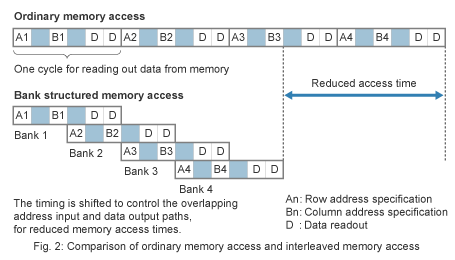
현재 출시중인 대부분의 SDRAM DIMM 모듈을 2 Bank나 4 Bank로 구성되어 있다. 2 Bank 모듈의 경우 대부분 16Mbit의 SDRAM 칩으로 구성되어 32MB의 작은 용량인데, 실제 현재 다수로 사용되는 64/128MB의 SDRAM DIMM의 경우 64Mbit - 256Mbit의 칩으로 구성되어 있으며 4 bank로 구성되어 있다.

하지만 뱅크 인터리빙이 만병 통치약은 아니고, 이를 구현하는데에도 당연히 단점이 따른다. 일단 뱅크수만을 늘린다고 해서 장땡은 아니라는 것이다. 뱅크수를 늘린다고 하면 Memory Depth가 늘어가 나게 되며 이는 전적으로 저렴한 메모리로 고성능 구현이라는 목표에 역행할 수도 있기 때문이다.

메모리의 데이터에 접근하는 방법은 위 그림 처럼, 가로행 주소를 찾고, 세로열 주소를 찾아 확인되는 특정 위치의 값을 읽어 들여 갖는 것으로 이해하면 된다.

The access paths from the memory controller to memory are limited; only one set of paths is available for address and data, respectively. Reading large data volumes from one memory chip can be time-consuming. For this reason, the internal memory architecture is configured in four layers (banks) to increase data read performance.

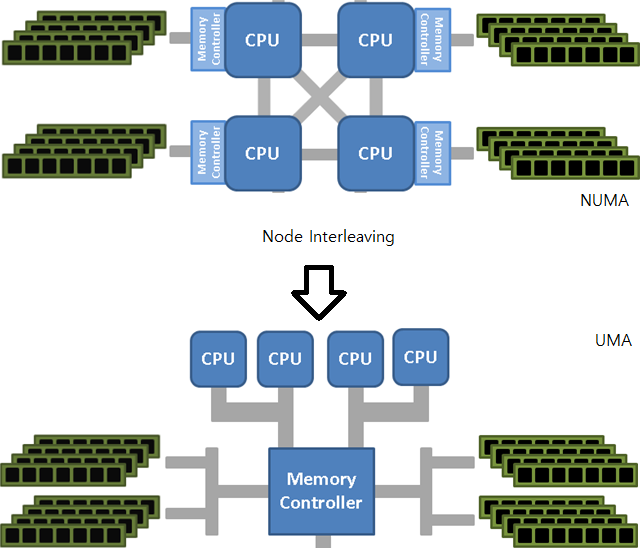
이 때, 기본적으로는 한 번에 한 개의 데이터만 접근할 수 있다. 이 것이 메모리 용량이 작은 경우는 곰방 찾는데, 용량이 큰 경우 검색에 시간이 걸리게 된다. (데이터가 착해서 순차적으로 들어있지 않다면 더욱 더 ...)



Interleaving is the set-up of the physical address space by alternating between multiple memory resources of the same type. First of all, the two memory controllers of a processor are eligible in the case of the Ivy Bridge-EX based servers. The first block of the local address space segment is in the first controller, the second one in the second, the third one back in the first, etc. This principle can then be continued on the level of the four memory channels per controller, and finally on the level of the ranks within the individual memory channel.

인터리빙 기술은 보다 빨리 데이터에 접근하고자 하는 기술로, 데이터가 포함되어 있는 주소만 확인되면, 데이터가 정상적으로 읽어졌다는 정보를 확인하지 않고(그 작업은 기존 Task에 맡겨두고), 바로 다른 데이터에 접근(새로운 Task)을 시작하는 것으로 이해하면 되겠다.

**Node Interleaving**

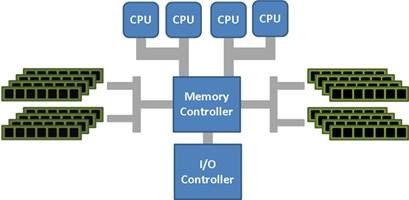


Node Interleaving disables the NUMA architecture properties of the system. All operating systems supported by this platform support NUMA architectures. In most cases, you can obtain optimum performance by disabling Node Interleaving. When this option is enabled, memory addresses are interleaved across the memory installed for each processor and some workloads may experience improved performance.

IMPORTANT: Enabling Node Interleaving may impact operating system performance. Enabling Node Interleaving requires that all nodes have equal memory sizes.

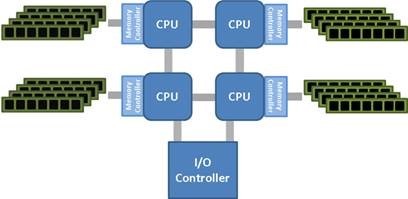
과거 UMA 아키텍쳐에서는 메모리 컨트롤러가 시스템 보드의 메인 칩셋(North Bridge)에 존재 했다.

메모리 컨트롤러가 별도로 존재했기에, 시스템의 CPU는 North Bridge를 통해 모든 메모리 영역을 접근/사용 가능했다.



NUMA 아키텍쳐에서는 메모리 컨트롤러가 CPU 내부에 존재하여, CPU 별로 메모리가 함께 그룹되어 사용된다.

메모리 컨트롤러가 CPU 내부에 존재하기 때문에, 각 CPU는 다른 CPU의 동의(?) 없이는 다른 CPU와 연결된 메모리 영역을 접근/사용할 수 없다.

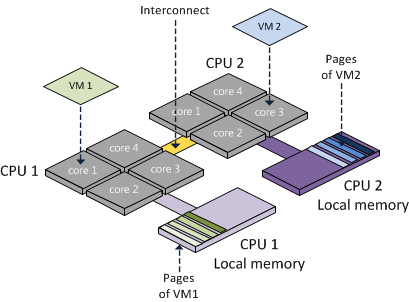


NUMA 아키텍쳐의 메모리 접근은,

특정 CPU에 대한 자료는 연결된 메모리 그룹에 상주하기 때문에, 빠르게 접근이 가능하고, 각 CPU가 작업 중 불필요한 I/O를 줄일 수 있다.

그러나, 단점일 수 있는 것은, 특정 CPU에 연결된 메모리 그룹에 할당된 용량만 사용이 가능하고, 특정 CPU에서 하던 작업이 다른 CPU로 이동되는 경우 메모리 데이터 역시 전반적으로 재구성(I/O) 발생된다.

또한, 동일 데이터가 여러 메모리에 중복 상주하는 이벤트가 발생할 수 있다.



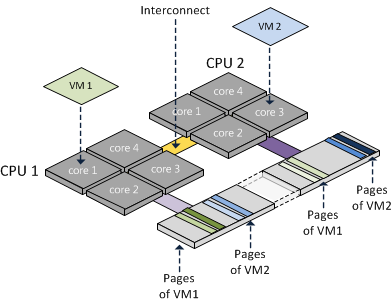
“Node Interleaving”은 NUMA아키텍쳐상의 메모리 Addressing을 UMA 아키텍쳐와 같이 변환(에뮬레이트?) 해주는 기능으로 시스템에 있는 모든 메모리를 하나의 그룹으로 구성해주는 것이다.

이 기능의 장점은 각 CPU가 다른 CPU에 연결된 메모리에 접근 가능하게 된다는 것.

(메모리를 보다 많이 사용할 수 있고(시스템에 장착된 최대 용량만큼), 또한 NUMA가 아닌 것처럼 동작한다는 것)

단점으로는, 위에 언급된 것처럼, 연산에 필요한 데이터가 직접 연결된 CPU 공간에 있다면 문제되지 않겠지만, 다른 CPU에 연결된 메모리에 존재한다면, 해당 CPU를 통해 메모리를 접근해야 하기 때문에, 지연이 발생하게 된다.

(해당 CPU가 작업 중이라면 더욱 그러하다. (작업 끝날때까지 기다려야 하고, 데이터를 주고 받는 동안은 해당 CPU는 다른 작업을 못한다. 그냥 접근하는 것이 아니라, 메모리를 관리하는 CPU가 자료를 찾아 필요한 CPU에 QPI를 통해 전달해주는 형태로 동작하기 때문. CPU가 워낙 빠르고, QPI 역시 빠르기 때문에 인지할 수 없을지라도, 컴퓨터는 인지하겠 ... 쿨럭 ...) 따라서, 이러한 작업(다른 CPU에 연결된 메모리에 들어있는 데이터 요청)이 많아지면 CPU가 본연의 기능(연산)이 아니라 데이터 찾는 역할을 하느라 성능에 영향을 줄 수 있다는 것이겠다.



출처: <http://infoages.tistory.com/1499> [정보와 지식의 올바른 공유 - http://infoages.com]