운영체제의 지원 없이 이중 페이지를 지원하는 TLB

이정훈[0] 이장수 김신덕
연세대학교 컴퓨터과학과
{ljh, jslee, sdkim}@kurene.yonsei.ac.kr

A dual TLB supporting two pages without operating system aid

Jung-Hoon Lee[0] Jang-Soo Lee Shin-Dug Kim
Dept. of Computer Science, Yonsei University

요 약

TLB 성능을 높이기 위한 기존의 3가지 주요 연구방향은, ① TLB 엔트리 개수를 최대한 증대시키는 방법, ② 페이지 크기 (page size)를 크게 증대 시키는 방법, ③ 다중 페이지 크기 (multiple page sizes)을 지원하는 방법 등의 연구가 제시되어 왔다. 이러한 방법들 중 다중 페이지 크기를 지원하는 방법이 가장 우수한 성능을 제공하는 방법이지만, 아직 어떠한 운영체제 (operating system)도 다중 페이지를 사용자 (user) 영역까지 지원하고 있지는 않은 상태이다. 따라서 다중 페이지의 효과를 살리기 위해 운영체제의 도움 없이 이중 페이지를 지원하면서 낮은 가격 (low cost)으로 높은 성능 (high performance) 향상을 보일 수 있는 새로운 듀얼 (dual) TLB 구조와 운영 방법을 제안하고자 한다. 제안하는 듀얼 TLB 구조는 작은 페이지 크기 (small page size)를 지원하는 완전 연관TLB와 큰 페이지 크기 (large page size)를 지원하는 완전 연관TLB로 구성된다. 제시된 구조는 기존의 많은 엔트리 개수를 지원하는 TLB와의 성능 비교분석 결과를 통해 볼 때, 작은 엔트리 개수를 가지면서도 거의 같은 성능을 보임을 알 수 있다. 또한 동일한 TLB 면적 크기로 기존 방식의 접근 실패율을 90%정도 감소시키는 성능을 제시하였다.

1. 서론

TLB 성능 향상을 위한 전형적인 방법은 크게 세가지 방식으로 구분되며 이러한 방법은, 1) TLB가 좀더 많은 엔트리 개수를 지원하는 방법, 2) 페이지 크기 (page size)를 크게 증대 시키는 방법, 3) 다중 페이지 크기를 지원하는 방법이다. 그러나 TLB가 지원하는 엔트리 개수가 증대되어진다면, 메모리 참조의 지연 (latency)이라는 역효과가 나타나며, 또한 일반적으로 TLB는 CAM (content addressable memory)으로 구현되어지기 때문에 참조 때마다 매번 많은 엔트리를 비교해야 함으로 전력면에서도 상당히 불리한 요소가 많다. 또한 페이지 크기를 증대 시킬 경우, 메모리의 사상 (mapping)의 적용 정도 (coverage)가 증가한다는 큰 장점을 가지게 되지만, 페이지 내부 단편화 (internal fragmentation)의 증가로 메모리 낭비가 심해지고 사상 되는 페이지의 수를 제한함으로 프로세스의 수가 제한을 받게 된다는 단점도 가지게 된다. 그러므로 현재 TLB 성능을 높이기 위한 가장 좋은 방법으로 다중 페이지 크기를 지원하여 주소 참조 특성에 의거 적정한 페이지 크기를 동적으로 선정하여 운영하는 방법이 효과적이다.

다중 페이지 크기를 지원하는 방법들 중에서 가장 적합한 것은 운영체제나 컴파일러로부터 일정한 정보를 받아서 가장 적합한 페이지 크기를 TLB에 할당하는 것이다. 그러나 운영체제의 시스템 (kurnel) 영역에서는 이러한 방식이 가능하나 사용자 (user) 영역에서는 현실적으로 이러한 방식을 지원하기 어렵기 때문에 우리는 사용자영역에서 운영체제의 지원 없이 이중 페이지를 지원할 수 있는 새로운 TLB 구조를 제안하고자 한다. 이중 페이지의 가장 큰 장점은 내부 단편화를 줄일 수 있으면서도 필요에 따라 큰 페이지의 사용으로 높은 성능 향상을 기대할 수 있다. 그러나 기존의 이중 페이지 지원 방식은 운영 체제의 도움을 받아서 수행되기 때문에 하드웨어적으로 표준화된 구조를 제시하기가 어려웠고 운영체제의 수정과 자원이 필요하였기 때문에 제한된 응용분야에 한정된 TLB구조로 사용되어 왔다.

2. 관련연구

운영체제의 지원을 받지 않고 이중 페이지를 지원하는 TLB로는 완전 서버블록 (complete subblock)TLB 가 있다 [Hill94]. 이 방식도 낮은 가격 대 성능 비를 고려한 것으로 TLB 하나의 엔트리에 태그 부분은 16k-byte의 페이지를 나타내는 하나의 가상 페이지 번호 (virtual page number: VPN) 와, 그리고 4 개의 물리적인 페이지 번호 (physical page number: PPN) 로 구성되어 진다. 이는 기본 페이지 크기를 4KB 로 두고 16KB 의 효과를 내기 위한 방법이며, 또한 운영체제의 지원을 받지 않는 구조임으로 4 개의 4KB 의 물리적인 페이지 번호를 소유하고 있어야 한다. 만약 가상 페이지 번호가 순차적으로 액세스 되어진다면 좋은 성능을 보일 것으로 기대되지만 항상 엔트리에 물리적인 페이지 번호를 저장할 공간을 마련해 두고 있기 때문에 비순차적일 경우에는 엔트리의 공간 낭비가 우려 된다. 또한 하나의 엔트리가 대치되는 경우 4 개의 4KB 물리적인 페이지 번호 정보가 모두 소실될 우려가 있으며. 이것이 성능 저하를 초래하는 가장 큰 단점으로 지적될 수 있다.

3. 제안된 TLB 구조

제안하고자 하는 듀얼 TLB구조는 작은 페이지 크기 (예로 4KB)을 지원하는 기본적인 완전 연관 TLB (fully-associative TLB)와 큰 페이지 크기 (예로 16KB)을 지원하는 완전 연관 TLB의 두 부분으로 구성되어 진다. 작은 페이지는 가상 페이지를 구성하는 기본 페이지 이다. 이후부터 작은 페이지 크기를 지원하는 TLB를 작은 페이지 TLB, 큰 페이지 크기를 지원하는 TLB를 큰 페이지 TLB라 명명하자.

작은 페이지 TLB의 각 엔트리는 하나의 가상 페이지 번호 (VPN)와 하나의 물리적인 페이지 번호 (PPN)로 구성되며 큰 페이지 TLB는 하나의 가상 페이지 번호와 네 개의 물리적인 페이지 번호로 구성된다. 이처럼 물리적 페이지 번호 네 개를 포함함으로써 운영체제의 지원을 완전히 배제하면서 이중 페이지를 하드웨어적으로 지원하기 위한 구조라 할 수 있다. 큰 페이지 TLB내 하나의 엔트리는 작은 페이지 TLB내 여러 개의 연속적인 엔트리 사상 정보에 해당하며 반드시 작은 페이지 TLB에서 한번 이상 참조된 경우에만 큰 페이지 TLB로 들어올 수 있다. 제시하는 듀얼 TLB를 구성하는 한 가지 설계의 예로 작은 페이지 크기가 4KB이고 큰 페이지 크기가 16KB인 경우를 가정할 때 큰 페이지 TLB의 운용은 다음과 같다. 작은 페이지 TLB내에 한 개의 큰 페이지에 해당하는 4개의 순차적인 작은 페이지 번호 중 3개의 순차적인 가상 페이지가 이미 존재할 경우, 다음에 참조하는 작은 페이지가 해당하는 큰 페이지의 4번째 가상 페이지이고 작은 페이지 TLB에서 접근 실패 (miss)가 발생하는 경우를 고려하자. 이 경우 4개의 순차적인 작은 페이지는 한 개의 큰 페이지로 재구성 가능하며 이를 페이지 승격 (promotion)이라 정의한다. 이때 MMU는 네 번째 작은 페이지에 대한 접근 실패를 처리하게 되고 그 처리 동안 작은 페이지 TLB내의 3개의 순차적 가상 페이지 번호와 접근 실패 처리중인 작은 가상 페이지 번호를 큰 페이지 TLB내에 한 개의 엔트리로 등록하게 된다. 이렇게 한 개의 큰 페이지 엔트리로 묶인 네 개의 작은 페이지 정보들은 페이지 승격과 동시에 작은 페이지 TLB에서 제거 할 수 있기 때문에 작은 페이지 TLB내에 네 개의 엔트리를 다시 사용할 수 있을 뿐만 아니라 큰 페이지 TLB 엔트리 하나로 작은 페이지 TLB 네 개의 엔트리 정보를 가질 수 있기 때문에 시간적 지역성을 보다 효율적으로 이용할 수 있는 장점을 가지고 있다. 이러한 구조로 설계될 경우 얻을 수 있는 최대 사상 효과는 작은 페이지 TLB의 엔트리 개수가 m 개, 큰 페이지 TLB 의 엔트리 개수가 n 개인 경우 $m + n \times$ (큰 페이지 크기 / 작은 페이지 크기)이다. 만약 4KB-16KB 페이지 구성의 경우 $m + 4n$ 이며 4KB-32KB 페이지 구성의 경우 $m + 8n$의 사상 효과를 가질 수 있다.

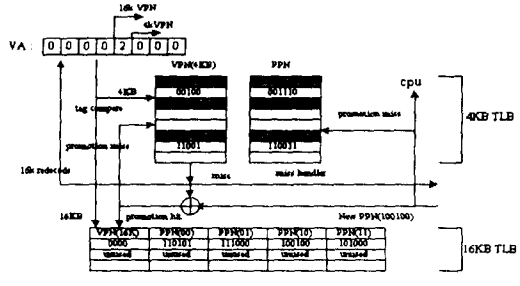

그림 1: 새로운 TLB 구조

제안된 듀얼 TLB 동작 원리는 그림1과 같다. 먼저 가상 주소가 중앙 처리 장치 (CPU)로부터 발생되면 기본 페이지 크기 (예: 4KB)의 페이지를 찾기 위하여 작은 페이지 TLB를 검색하는 동시에 큰 페이지 크기 (예: 16KB)를 검색하기 위하여 큰 페이지 TLB를 검색한다. 이러한 검색은 한 사이클에 완료될 수 있다. 두개의 TLB 검색이 끝나면 다음과 같은 3가지 경우가 발생한다.

1) 작은 페이지 TLB에서 접근 성공 (small TLB hit)
   기본적인 완전 연관 TLB처럼 하나의 물리적인 페이지 번호를 보내주고 CPU가 계속적으로 작업을 수행할 수 있도록 한다.
2) 페이지 TLB에서 접근 성공 (large TLB hit)
   예로 4KB-16KB 페이지 구성의 경우를 가정하면 16KB의 가상 페이지 번호의 비트수가 4KB의 가상 페이지 번호에 비해 2비트 적으므로 이 2비트를 이용하여 4개의 물리적인 페이지 중에 하나만 선별적으로 구동 가능하게 할 수 있다. 4개의 순차적인 작은 가상 페이지가 하나의 큰 페이지 TLB의 엔트리로 승격될 때 PPN(00) 위치에 순차적인 작은 페이지 번호 첫번째에 해당하는 PPN이 그리고 PPN(11) 위치에 마지막 작은 페이지 번호에 해당하는 PPN이 들어가기 때문에 접근 성공을 결정하는 동시에 하나의 PPN이 결정 되어질 수 있다. 이는 전력소모 측면에서도 많은 이득을 볼 수 있다.
3) 두 개의 TLB에서 모두 접근 실패 (small page TLB miss and large page TLB miss)

① 접근 실패가 발생한 가상 페이지 번호에 해당하는 순차적인 3개의 VPN가 작은 페이지 TLB에 존재하지 않는 경우:

먼저 2개의 TLB에서 접근 실패가 발생하면 이를 처리하는 동안 접근 실패가 발생한 VPN에 순차적인 3개의 페이지를 검색하기 위하여 접근 실패가 발생한 작은 TLB의 VPN에서 하위 두 비트를 제거하고 새로운 두 비트 (00, 01, 10, 11) 중에 접근 실패가 발생한 VPN를 제외한 나머지 세 경우에 해당하는 두 비트를 하드웨어적으로 붙여서 작은 페이지 TLB의 재검색을 수행하여 순차적인 3개의 엔트리를 탐색한 후 만약 3개가 존재하지 않으면 기본 페이지 크기에 해당하는 하나의 PPN이 작은 페이지 TLB내에 새로이 등록이 되며 FIFO방식으로 채워진다. 이와 같은 가상 페이지 재구성과 재검색은 운영체제가 접근 실패를 처리 (miss handling)하는 동안 수행되어지기 때문에 부가적인 수행 시간으로 반영되지 않는다.

② 접근 실패가 발생한 가상 페이지 번호에 해당하는 순차적인 3개의 VPN가 작은 페이지 TLB에 존재하는 경우:

접근 실패가 발생한 경우 위의 과정처럼 VPN를 재구성함으로써 작은 페이지 TLB의 재검색을 수행한 후 작은 페이지 TLB에서 순차적인3개의 VPN가 존재하게 되면, 운영체제가 접근 실패를 처리하는 동안 3개의 순차적인 PPN을 지정된 큰 페이지 TLB의 한 엔트리를 할당하여 저장시키고 작은 페이지 TLB내에 있는 3개의 엔트리를 무효화 시킨다. 그리고 마지막 새로운 PPN이 들어오면 큰 페이지 TLB의 지정된 위치에 저장시키고 TLB의 운용은 끝나게 된다. 이렇게 하나의 큰 페이지로 승격시킴으로 인해 큰 페이지의 효과를 얻을 수 있고 작은 페이지 TLB 엔트리의 4개를 다시 사용할 수 있으므로 물리적으로 제공되는 엔트리의 수를 더 효과적으로 증대 시킬 수 있다. 또한 반드시 순차적인 데이터만 승격 가능하게 함으로 미리 저장 공간을 확보하고 있는 기존의 완전 서브블록 TLB와의 차별성을 들 수 있다. 또한 큰 페이지 TLB의 각 엔트리 구성 시4개의 PPN을 모두 가지고 있는 것은4개의 PPN을 연속적인 저장 공간에 재할당할 필요를 없애기 위해서 이다.

4. 성능 평가

작은 페이지 크기와 큰 페이지 크기의 최적의 파라 미터를 결정하기 위하여 다양한 시뮬레이션을 수행하였다. 시뮬레이션 결과 작은 페이지 TLB의 엔트리 개수가 작은 경우에는 작은 페이지 크기는 4KB, 큰 페이지 크기는 8KB 가 최적의 성능을 보임을 알 수 있었다. 최근 대부분의 아키텍처에서 사용되고 있는 기본적인 TLB엔트리 개수를 128로 가정하는 경우, 즉 대략 512KB 메모리 사상 크기의 경우에는 작은 페이지 크기로 4KB와 큰 페이지 크기로 16KB의 조합이 최적의 성능을 보임을 알 수 있었다. 객관적인 성능 검증을 위하여 기존의 완전 연관 TLB 구조와 이중 페이지를 지원하는 완전 서브 블록 TLB 그리고 제안하는 듀얼 TLB에 대해서 같은 메모리 사상 크기 (예: 512KB)에 대하여 다양한 구성의 시뮬레이션을 수행하였다. 그 결과는 그림2와 같다. 여기서 "full4k-128entry"는 4KB의 페이지 크기를 가지는 기존의 완전 연관 TLB구조이며 엔트리 개수는 128개이다. 그리고 "4k64-16k16"은 4KB의 페이지 크기를 가지는 작은 페이지 TLB 구조로 엔트리의 개수는 64개이며 16KB의 페이지 크기를 가지는 큰 페이지 TLB 구조의 엔트리 개수는 16개이다. "complete subblock TLB"는 16KB의 페이지 크기를 지원하며 엔트리 개수는 32개이다. 여기서 주목할 것은 기존에 널리 사용되는 완전 연관 TLB (fully-associative TLB)에 비해 엔트리의 수가 대단히 적음에도 불구 하고 접근 실패율 (miss ratio)은 거의 같은 성능 결과를 보임을 알 수 있다. 대부분의 벤치 마크에서 거의 같은 성능을 보이는데, 이는 작은 엔트리의 큰 페이지 TLB에서 많은 적중률 (hit ratio)이 일어 나기 때문이다. 또한 그만큼 순차적 접근이 많이 발생하고 있음을 알 수 있다. 완전 서브 블록 TLB의 경우 같은 사상 크기로 시뮬레이션을 수행하였으므로 엔트리의 수가 너무 적어 전반적으로 낮은 성능을 보이지만 가격 대 성능 비를 따져보면 좋은 성능 효과를 보일 것으로 기대 된다. 다음 절에서는 보다 정확한 성능 비교를 위해 가격 대 성능 비를 측정한 시뮬레이션 결과를 보일 것이다.

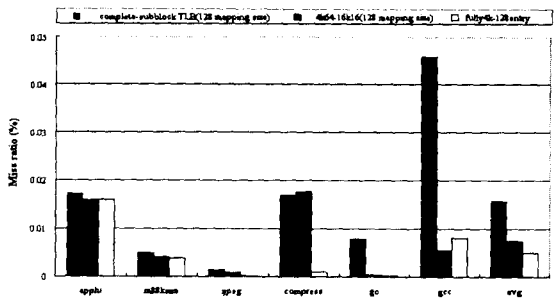

그림 2. 같은 매핑 사이즈를 가지는 세가지 구조.

5. 가격 (cost) 대 성능 비 (performance ratio)

일반적으로 TLB 의 태그부분은 각 엔트리의 동시 비교를 위해 CAM (content addressable memory) 을 주로 사용한다. 이러한 CAM 은 각 셀 (cell) 마다 저장과 비교 (comparison) 회로를 가지고 있다. 그렇기 때문에 CAM 의 사이즈가 RAM 의 사이즈보다 보통 두 배의 크기를 가진다.[Muld 91] 우리는 공정한 성능 향상을 위해 같은 면적 (area)을 산출하여 동일한 크기에서 성능을 분석 하고자 한다. 그 단위는 rbe (register bit equiva lents)이고, 전체 면적은 수식 (1)과 같다.

Area = PLA + RAM + CAM --- (식.1)

여기서 제어 회로인 PLA (programmable logic array) 는 130rbe로 RAM 셀은 0.6rbe, CAM 셀은 1.2rbe 로 가정한다. RAM 면적은 수식 (2) 와 같다.

RAM =0.6*(#entries+ #Lsense_amp)*((#data bits + #status bits) + Wdriver) (식 .2)

여기서 Lsense_amp 는 비트라인 센서 앰프의 길이 (length of bit -line sense amplifier) 이고, Wdriver 는 드라이버의 넓이 (width) 이다. 여기서는 모두 6 으로 가정한다. 마지막으로 CAM 의 면적은 수식 (3)과 같다.

CAM=0.6*(√2 * #entries + #Lsense_amp)*(√2 * #tag bits + Wdriver) ---(식. 3)

이러한 수식을 이용하여 128 엔트리 완전 연관 TLB 와 동일한 크기를 가지는 완전 서버 블록 TLB 와 우리가 제안한 성능 평가 결과는 다음 그림 (3)과 같다. 단 우리가 제안한 구조의 비용 (overhead) 을 감안하여 다소 낮은 크기를 두고 시뮬레이션을 수행하였다.

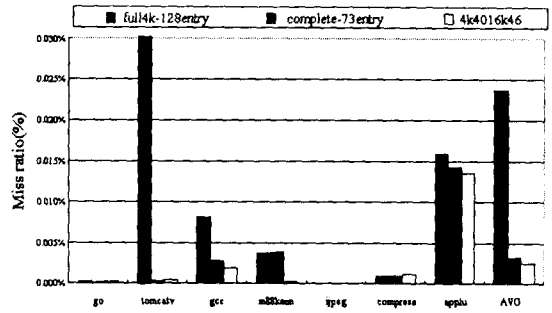

그림 3. 동일한 사이즈를 가진 세가지 구조.

6. 결 론

최근의 TLB 연구는 다중 페이지를 지원하는 추세이다. 운영체제의 지원만 받는다면 다중 페이지 (multi -page)를 지원하는 것이 TLB 성능을 높이는데 가장 좋은 방법이지만 아직 까지 어떠한 운영체제도 사용자 영역까지 지원하고 있지 않다. 하지만 운영체제의 지원이 없는 상태에서 단지 이중 페이지만을 지원하여도 낮은 가격에 높은 성능을 보이고 있고, 엔트리 수의 감소로 매번 많은 엔트리를 접근해야 하는 기존의 TLB 구조보다 전력 면에서도 다소 유리한 면이 있을 것으로 기대된다.

7. 참고 문헌

[Hill94] Mark D Hill and M. Talluri, "Suppassing the TLB Performance of Superpages with Less Operating System Support," *ASPLOS VI-, San Jose, California USA*, pp. 171-182, Oct. 1994.

[Tall92] M Talluri, Shing Kong, Mark D. hill, David A. Patterson, "Tradeoffs in Supporting Two Page Sizes," *the 19th Annual International Symposium on Computer Architecture*, pp. 415-424, May. 1992.

[Muld91] Johannes M.Mulder, N.T. Quach, Michael J.Flynn, "An Area Model for On-Chip Memories and its Applications," *IEEE journal of solid state Circuits*, 26(2) pp. 98-106, Feb. 1991.

[Khal92] Y.A.Khalidi, "Virtual memory support for multiple page sizes," *Proc. of the fourth workshop on workstation operating systems*, Oct. 1993.