멀티쓰레디드 프로세서를 위한 고성능 분기예측기 구조

배영돈, 박인철
한국과학 기술원 전자전산학과 전기 및 전자공학 전공
donny@ics.kaist.ac.kr, icpark@ee.kaist.ac.kr

Abstract – *Due to the presence of context switches, the accuracy of the branch predictors of multithreaded processors are degraded significantly. This paper presents a novel branch predictor architecture called Selective Sector Replacement (SSR) that improves the prediction accuracy by saving and restoring the predictor state on context switches with negligible timing overhead. A state compression scheme is also presented to reduce the size of the memory storing the predictor state of each thread. The evaluation results show the proposed architecture reduces the misprediction rate by 39% on the average comparing to the conventional branch predictors while the average time overhead to replace the predictor state at each context switch is less than 3% of conventional saving and restoring scheme.*

Keywords: branch prediction, multithreaded processor

1 서론

멀티쓰레딩 기술을 이용하여 프로그램의 실행속도를 높이는 연구가 점점 더 주목을 받고 있다. 2 개의 쓰레드를 지원하는 인텔의 하이퍼쓰레딩(Hyper-Threading)구조를 비롯 상용 마이크로프로세서 중에도 멀티쓰레딩을 지원하는 제품이 많아지고 있다[1-4]. 조건분기명령어들은 파이프라인 플러쉬를 수반하기 때문에 지속적으로 명령어를 읽어 들이는 것(fetch)을 방해하고 성능을 저하시킨다. 따라서, 고성능의 분기예측기로 분기조건을 미리 예측하여 성능저하를 최소화 하는 것이 중요하다. 그러나, 멀리쓰레디드 프로세서는 서로 다른 프로그램이 뒤섞여 실행되기 때문에 기존의 분기예측기를 구조를 사용하여 높은 성능을 얻을 수 없다.

일반적인 마이크로프로세서의 분기예측기에 대한 연구는 활발히 이루어진 반면, 멀티쓰레디드 프로세서를 위한 분기예측기에 대한 연구는 매우 적은 수에 지나지 않았다[5-7].

동적분기예측기(dynamic branch predictor)는 분기방향을 분기예측테이블에 저장하고 최근의 분기기록(branch history)를 토대로 다음 분기 방향을 예측한다[8]. 따라서, 동적분기예측기로 정확한 결과를 얻기 위해서는 충분한 시간 동안의 분기기록을 저장하고 있어야 한다. 따라서, 동적분기예측기는 한가지 프로그램을 계속해서 실행하는 경우에 높은 성능을 보이지만, 실행 프로그램이 자주 바뀌게 되면 그때마다 분기예측테이블을 갱신해야 하고, 갱신 시간 동안 예측실패율이 높기 때문에 전체 성능이 나빠지게 된다. 따라서, 작업간의 전환이 자주 일어나는 멀티쓰레디드 프로세서는 분기예측 성능이 좋지 않다.

멀티쓰레디드 프로세서의 분기예측 성능을 높이는 방법은 크게 세가지로 구분할 수 있다. 첫째는 가장 간단한 방법으로 작업전환시 분기예측기를 초기화 하는 것이다[6]. 다른 프로그램의 분기특성을 저장하고 있는 분기예측기는 간섭(aliasing)이 발생하여 초기값을 저장하고 있는 경우보다도 결과가 좋지 못하다. 따라서, 분기예측기를 초기화하는 것만으로 성능을 향상시킬 수 있다. 두 번째는 기존에 연구된 동적분기예측기중에 멀티쓰레디드 구조에 좋은 성능을 보이는 구조를 사용하는 것이다. 예를 들어 skew 예측기의 경우 associativity 가 많은 경우에 알맞게 설계되어 있기 때문에 좋은 결과를 보이며[9], 분기기록을 이용하지 않기 때문에 멀티쓰레디드 환경에서 성능저하가 적은 bimodal 방식을 혼용하는 hybrid 방식과 bi-mode 방식 또한 좋은 성능을 얻을 수 있다[7,10,11]. 세 번째는 작업전환이 일어날 때마다 분기예측테이블의 내용을 메모리에 저장하고 복원하는(save and restore) 방법이다[6,7]. 이 경우는 분기예측 성능을 최대화(싱글쓰레디드 프로세서 수준으로) 할 수 있는 반면, 작업전환 시 저장하고 불러오는 시간이 필요하고 별도의 메모리가 필요하다는 단점이 있다. 분기예측테이블의 내용을 압축해서 저장할 경우, 성능을 희생하는 대신 이러한 단점을 줄일 수 있게 된다.

그러나, 이러한 방식들은 작업전환주기가 긴 coarse-grain 방식의 멀티쓰레디드 프로세서의 경우에는 성능

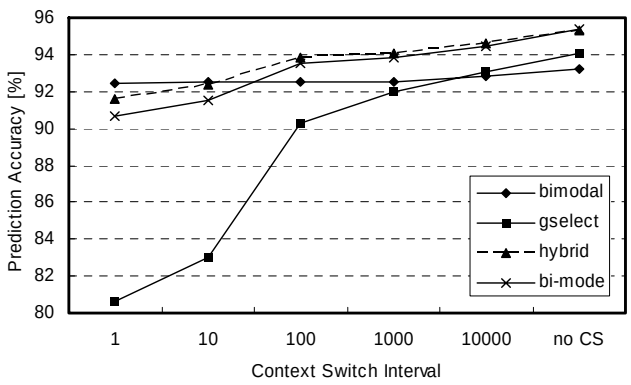


그림 1. 분기예측기 성능비교

향상을 기대할 수 있으나, 작업전환이 자주 일어나는 경우나 특히 매 싸이클 작업전황이 일어나는 fine-grain 방식의 멀티쓰레디드 프로세서에는 적합하지 않다. 그림 1 은 작업전환주기(context switching interval)에 따른 분기예측기의 성능을 나타내고 있다. 작업전환주기가 짧아짐에 따라 gselect[12]방식과 같은 일반적인 2 단계 동적분기예측기들은 매우 큰 성능 저하를 보이게 되며, 기존의 연구 중 멀티쓰레디드 프로세서에 가장 좋은 성능을 보이는 hybrid 와 bi-mode 방식 역시 fine-grain 멀티쓰레디드 프로세서에는 적합하지 않은 것을 알 수 있다.

본 논문에서는 coarse-grain 방식뿐만아니라 fine-grain 방식에 적용할 수 있는 고성능 분기예측기(SSR: Selective Sector Replacement)의 구조를 제안한다. SSR 분기예측기는 작업전환시간(context switching time)에 오버헤드 없이 단일프로그램의 경우와 동일한 성능을 얻을 수 있다.

2 SSR 분기예측기

2.1 SSR 분기예측기의 원리

기존의 저장 및 복원(save & resore)방식[6,7]이 작업전환이 일어날 때마다 분기예측테이블 전체를 저장하는 반면, SSR 분기예측기는 분기예측테이블을 섹터로 구분하고, 분기명령어가 접근하는 섹터만 해당 쓰레드로 전환한다. 이것은 SMT(Simultaneous Multithreading)이 아닌 일반적인 멀티쓰레디드 프로세서의 경우 동시에 실행되는 분기명령어가 한 개이고, 접근되지 않는 분기예측테이블의 내용은 저장/복원할 필요가 없다는 점에서 착안한 것이다.

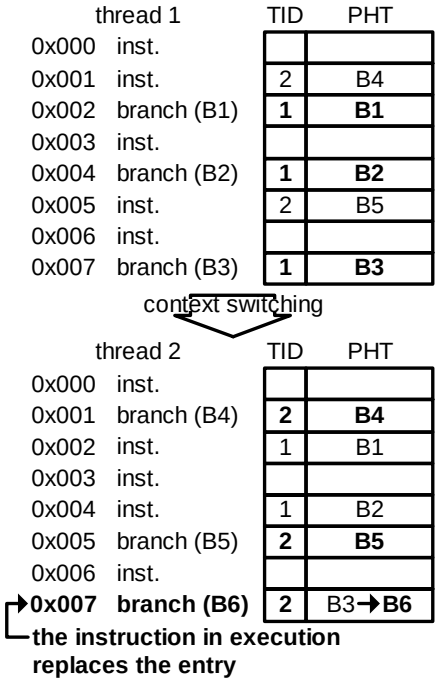


그림 2. SSR 분기예측기의 원리

그림 2 는 SSR 분기예측기의 원리를 보여주고있다. 쓰레드 1 에서 쓰레드 2 로 작업전환이 일어나도 thread 1 에서 접근했던 분기예측테이블의 내용은 그대로 남게 되며, 쓰레드 아이디(TID: thread identifier)정보를 통해 어느 쓰레드의 정보인지 구분할 수 있다. 쓰레드 2 를 실행하는 동안 분기명령어(B1)가 TID 가 일치하지 않는 테이블의 내용(즉, 다른 쓰레드에서 사용했던 테이블의 내용. 그림 2 의 0x007 번지)을 사용하게 되면 이를 별도의 저장공간에 저장하고, 쓰레드 2 에 해당하는 정보를 읽어와서 복원하게 된다. 이와 같이 필요한 테이블의 내용만 섹터단위로 선택적으로 치환하는 방식을 사용하므로 이 분기예측기의 구조를 선택적부분치환(SSR: Selective Sector Replacement)라고 명명하였다. 섹터의 크기는 2^N(N≥0)개의 엔트리이다.

2.2 SSR 의 데이터 압축 방법

이와 같이 부분치환방식으로 작업전환시간의 오버헤드를 제거하더라도 여전히 저장 및 복원방식은 단점이 존재한다. 즉, 각 쓰레드별로 분기예측테이블의 내용을 저장하는 메모리가 필요하다. 이 메모리의 크기를 줄이기 위해선 분기예측테이블의 내용을 압축하는 기술이 필요한데, 이를 위한 압축 방법으로는 테이블마다 1 비트 대표 값을 저장하는 방법(majority bias scheme)과 테이블을 몇 개의 블록으로 나누고 각 블록의 바이어스 값을 계산하여 저장하는 방법(bias scheme), 그리고 각 카운터의 상위비트만 저장하는 방법(snapshot)이 있다[7].

이러한 압축방법을 사용하면 테이블 저장용 메모리의 크기는 줄어드는 반면, 압축을 위한 별도의 회로와 계산 시간이 필요하게 된다. 그러나, SSR 분기예측기는 작업전환시에 데이터 압축을 위한 시간이 필요하지 않다(그림 3).

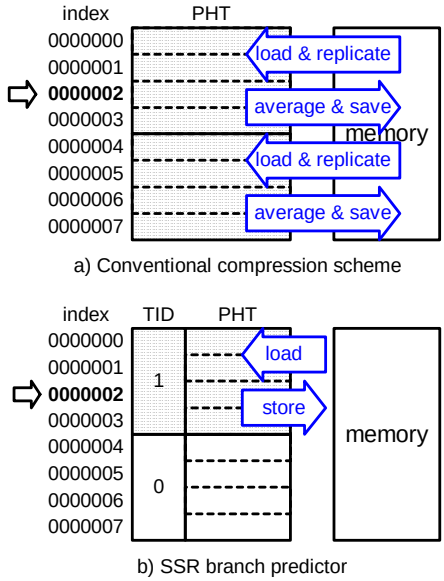
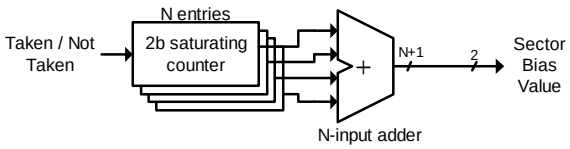
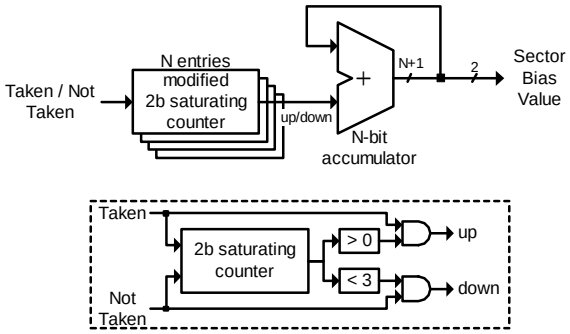


그림 3. SSR 분기예측기의 섹터별 압축



a) 가산기를 이용한 바이어스 값 계산회로



b) SSR 분기예측기의 바이어스값 계산 회로

그림 4. 바이어스값의 계산

이를 위하여 SSR 분기예측기는 바이어스 값을 계산하기 위한 특별한 구조를 사용하고 있다. 기존의 압축방식이 가산기,등을 이용하여 평균값을 계산하는 반면(그림 4-a), SSR 구조는 up/down 신호를 출력하도록 설계된 2 비트 카운터(2-bit saturating counter)와 적분기(accumulator)를 사용하여 매번 분기 명령어가 실행될 때마다 up/down 값을 이용하여 바이어스 값을 갱신한다(그림 4-b). Up/down 신호는 카운터 값이 변경될 때(카운터값이 포화상태가 아닌 경우)만 활성화되기 때문에 매번 이 신호를 적분하고 적분값의 상위 두 비트만 취하면 섹터전체의 바이어스 값을 얻게 된다. 한 비트의 up, down 입력만 있는 적분기를 사용하면 한 번에 여러 개의 카운터 값을 더하는 방식에 비해 하드웨어가 훨씬 적게 들고, 계산과정이 분산되어 계산 속도가 빨라지는 효과가 있다.

2.3 SSR 분기예측기의 구조

그림 5 는 SSR 분기예측기의 전체구조를 나타낸다. SSR 분기예측기는 기존의 모든 동적분기예측방식에 적용할 수 있으며, 그림은 gshare 방식을 나타내고 있다. 각 쓰레드별로 분기기록레지스터(BHR: branch history register)가 있으며 프로세서에서 현재 실행되고있는 쓰레드의 아이디(TID: thread identifier)를 사용해 선택된 분기기록과 프로그램카운터(PC)의 값을 XOR 연산하여 분기예측테이블의 주소를 계산한다. 해당 주소의 분기예측테이블 값의 TID 와 프로세서의 TID 값이 일치하면 테이블에 저장된 값을 사용하며, TID 값이 일치하지 않으면 해당 섹터의 치환작업이 이루어지며, 메모리의 해당번지(테이블주소와 TID 를 병합하여 생성)로부터 읽어온 값을 사용하고 그 값으로 분기예측테이블의 내용을 변경한다. 이와 동시에, 적분기에 저장된 바이어스 값을 메모리에 저장한다.

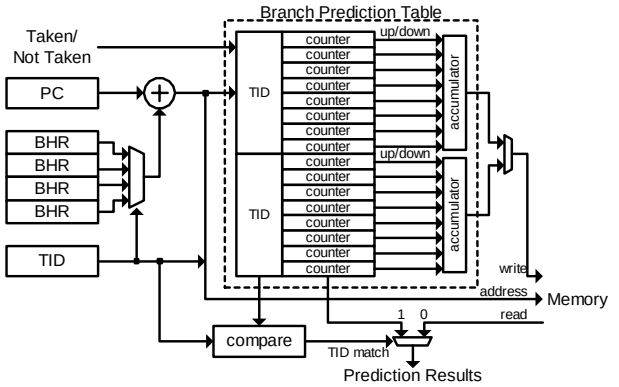


그림 5. SSR 분기예측기의 구조(gshare 방식 적용의 예)

SSR 방식을 사용하기 위한 계산시간의 증가는 TID 를 비교하기 위한 비교기(comparator)와 그 값으로 분기예측테이블값과 메모리에서 읽어 들인 값을 선택하기 위한 멀티플렉서의 지연시간뿐이다. 일반적으로 분기예측테이블의 크기는 캐쉬 메모리보다 크기가 작기 때문에 분기예측테이블을 온칩 메모리에 저장할 경우의 메모리의 동작속도는 프로세서동작속도에 영향을 주지 않는다. 외부 메모리를 사용할 경우에는 분기명령어 중 TID 가 일치하지 않는 경우만 외부메모리의 지연시간만큼 오버헤드가 발생하게 된다.

3 성능 측정

SSR 분기예측기의 성능을 알아보기 위하여 gshare, hybrid, bi-mode 구조에 SSR 방식을 적용하고 MiBench 를 사용하여 결과를 확인하였다.

그림 6 은 8 개의 쓰레드를 갖고 있는 경우에 대해 각 방식에 SSR 구조를 적용함으로써 향상된 성능(적중율)의 정도이다. 이 결과는 각 쓰레드별로 별도의 분기예측기를 갖고 있고 전체 분기테이블의 크기가 같을 때를 기준으로 한 것이다. 32byte 부터 8Kbyte 의 분기테이블 크기에 대해 평균 정확도가 89%에서 94%로 약 5%가 향상되었다. 이것은 예측 실패율의 상대적인 크기로 볼 때 48%의 실패율이 감소한 것이다.

쓰레드의 개수가 많아지면 상대적인 성능 향상도 커진다. Bi-mode 방식의 경우 16 개의 쓰레드에 대해 10% 가까운 성능 향상을 보인다. 16 개 쓰레드의 경우에 평균적으로 감소된 실패율은 58%에 이르며, 2 개에서 16 개까지 쓰레드에 대한 전체 평균 값은 약 39%이다.

SSR 구조를 적용하여 이와 같은 성능 향상을 얻기 위해서는 SSR 방식을 사용하기 위해 추가된 하드웨어가 존재한다. gshare 와 같은 일반적인 2 단계 분기예측방식은 그림 8 과 같이 쓰레드가 증가함에 따라 오버헤드가 100%가까이 된다. 그러나, hybrid 나

bi-mode 방식은 20%내외의 하드웨어만 증가한다. 이것은 hybrid 방식과 bi-mode 방식이 bimodal 과 gshare 의 함께 사용하는 구조를 갖고 있기 때문이며, 그림 1 과 같이 bimodal 은 작업전환에 관계없이 거의 일정한 성능을 보이기 때문에 TID 정보를 저장하지 않고도 같은 수준의 성능을 얻을 수 있으며 이로 인해 상대적으로 적은 하드웨어가 추가되는 것이다.

그림 9 에 나타난 것과 같이 SSR 구조를 사용하여 작업 전환 시 분기예측테이블을 저장하고 복원하는 시간을 줄일 수 있다. SSR 방식은 분기예측테이블의 크기에 관계없이 한 싸이클에 작업에 이루어지며, 이로 인해 줄일 수 있는 시간은 기존의 bias 방식를 기준으로 압축을 하지 않을 경우 97%, 4 개의 엔트리 단위로 압축을 하는 경우에 91%이다.

4 결론

본 논문에서는 멀티쓰레디드 프로세서에 적합한 분기예측기 구조를 제안하였다. 제안된 SSR 분기예측기구조는 작업전환에 별도의 시간이 필요하지 않은 저장 및 복원방식으로 기존의 다양한 동적분기예측기 방식에 적용이 가능하며, 또한 섹터단위의 압축방식을 통해 저장되는 분기테이블의 크기를 선택할 수 있다. 분기테이블의 크기가 같은 기존방식에 비해 평균적으로 예측 실패율의 39%가 감소되고, 분기예측테이블을 저장 하고 복원하기 위한 시간의 97%가 줄어들며, 이를 위해 증가된 하드웨어는 20%이다.

참고문헌

[1] Y.-D. Bae and I.-C. Park, "A 4.75GOPS single-chip programmable processor array consisting of a multithreaded processor and multiple SIMD and IO processors," in Proc. Custom Integrated Circuits Conference, pp.583-586, 2004.

[2] Y.-D. Bae, S.-I. Park and I.-C. Park, "A Single-Chip Programmable Platform Based on a Multithreaded Processor and Configurable Logic Clusters," IEEE Journal of Solid-State Circuits, vol. 38, pp.1703-1711, Oct. 2003

[3] E. Norden, "A Multithreading Extension for Low-Power, Low-Cost Applications," in Proc. Embedded Processor Forum, 2003.

[4] S. Ge, X. Tian and Y.-K. Chen, " Efficient multithreading implementation of H.264 encoder on Intel hyper-threading architectures," in Proc. Int. Conf. on Information, Communications and Signal Processing, pp.469-473, 2003.

[5] M. Evers, P.-Y. Chang and Y. N. Patt, "Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches," in Proc. Int. Sym. on Computer Architecture, pp. 3-11, 1996.

[6] A. S. Dhodapkar and J. E. Smith, "Saving and Restoring Implementation Context with co-Designed Virtual Machines," in Workshop on Complexity-Effective Design, 2001, Goteborg, Sweden.

[7] S. Pasricha, A. Veidenbaum, "Novel Techniques to Improve Branch Prediction Accuracy for Embedded Processors in the Presence of Context Switches", CECS Tech. Report, 2003

[8] T.-Y. Yeh and Y. N. Patt, "Alternative Implementations of Two-Level Adaptive Branch Prediction," in Proc. Int. Sym. on Computer Architecture, pp.124-134, 1992.

[9] P. Michaud, A. Seznec, and R. Uhlig, "Trading Conflict and Capacity Aliasing in Conditional Branch Predictors," in Proc. Int. Sym. on Computer Architecture, pp.292-303, 1997.

[10] S. McFarling, "Combining Branch Predictors", in DEC WRL TN-36, June 1993.

[11] C.-C. Lee, I.-C. Chen, and T. Mudge, "The bi-mode branch predictor", ISM 1997

[12] S.T.Pan, K.So, and J.T.Rahmeh, "Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation," in Proc. Architectural support for programming languages and operating systems, pp.76-84, 1992.

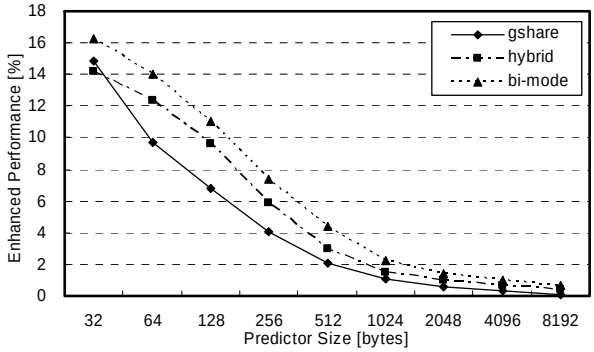


그림 6. 상대적으로 향상된 정확도(쓰레드 8 개의 경우)

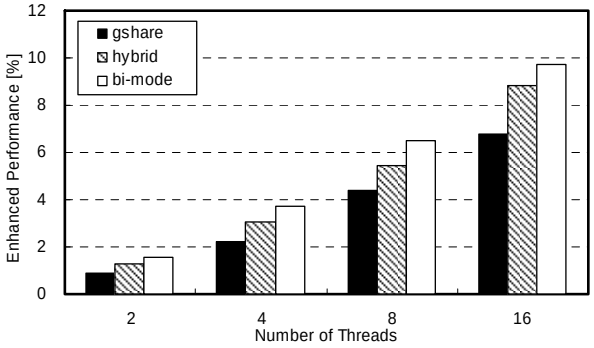


그림 7. 쓰레드 개수와 향상된 정확도

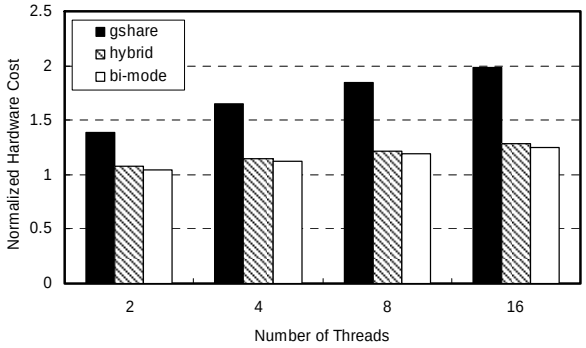


그림 8. 하드웨어 면적 비교

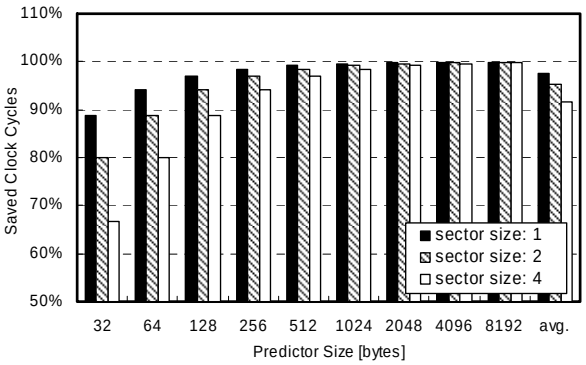


그림 9. 감소된 분기예측테이블의 저장 및 복원 시간