분기 정보의 투기적 사용에 대한 효율적인 복구 기법

곽종욱[O†] 김주환[†] 장성태[††] 전주식[†]
[†]서울대학교 전기컴퓨터공학부
[††]수원대학교 컴퓨터학과
[†]{leoniss[O], if10001, csjhon}@panda.snu.ac.kr, [††]stjhang@suwon.ac.kr

Recovery Modules for Speculative Update Branch History

Jong Wook Kwak[O†] Ju-Hwan Kim[†] Seong Tae Jhang[††] Chu Shik Jhon[†]
[†]Dept. of Electrical Engineering and Computer Science, Seoul National University
[††]Dept. of Computer Science, The University of Suwon

요 약

분기 명령어의 예측 정확도는 시스템 전체 성능에 중대한 영향을 미친다. 여러 분기 예측 방식 가운데 하나인 "분기 정보의 투기적 사용" 은 분기 명령어의 가장 최근 기록을 일관되게 사용할 수 있도록 도와줌으로 해서 분기 예측의 정확도 향상에 크게 기여한다. 하지만 이와 같은 기법은 미완료 분기에 대한 히스토리를 투기적으로 사용하는 방식이다. 따라서 사용되는 정보가 올바르지 못할 수 있으며, 이런 경우 적절한 복구 기법을 필요로 한다. 본 논문에서는 분기 정보의 투기적 사용에 대한 필요성과 효율적인 복구 기법을 제안한다. 제안된 기법은 이전 연구와 비교하여 상당한 하드웨어 요구량의 감소를 가져왔으며, 또한 프로그램 수행의 정확성을 해치지 않으면서 최대 3.3%의 성능향상을 보였다.

1. 서 론

분기 명령어의 분기 예측 정확도는 시스템 전체 성능 향상에 중요한 영향을 미친다. 특히 분기 예측의 실패에서 오는 예측 실패 비용(Miss-Prediction Penalty)은, 최근의 마이크로프로세서 발전 경향과 맞물려 파이프라인(Pipeline)의 단계수가 늘어나고 단위 시간당 제기되는 명령어의 수가 증가할수록 더욱 커지는 추세이다. 이와 같은 분기 예측의 정확도를 높이기 위해 여러 가지 다양한 형태의 분기 예측 기법들이 제안되어 왔으며, 특히 이단계 적응형 분기 예측 기법의 소개 이후로 분기 명령어들의 히스토리(Branch History)는 분기 예측의 중요한 입력 요소 가운데 하나로 자리 잡았다[1].

이와 함께, 분기 명령어의 새로운 입력 요소 제안, 동적인 히스토리 길이의 조절등과 같은 분기 예측 입력 요소들의 효과적 활용에 대한 연구도 진행 되었다. 그 중 하나인, 투기적 분기 정보(Speculative Update Branch History)의 사용은 각 분기 명령어들의 가장 최근 히스토리들을 일관성 있게 사용할 수 있도록 도움을 준다. 일반적으로 현재 예측하고자 하는 분기 명령어와 가까운 최근의 분기 명령어들 일수록 현재의 분기 예측 결과에 보다 더 많은 영향을 준다. 그러한 측면에서 분기 정보의 투기적 사용은 분기 예측의 정확성 향상에 큰 영향을 미친다[2].

하지만 이와 같은 투기적 정보는 정확성이 결여된 정보로서, 미완료 분기에 대한 분기 히스토리 정보를 투기적으로 사용한다. 따라서 사용된 정보가 올바르지 못할 수 있으며, 그럴 경우 적절한 복구 기법을 필요로 한다. 본 논문에서는 이와 같은 분기 정보의 투기적 사용에 대한 성능 향상의 정도를 살피고 분기 정보의 투기적 사용에 대한 필요성을 제시한다. 아울러, 분기 정보의 투기적 사용에서 비롯되는 적절한 복구 기법을 소개한다. 이하 본 논문의 구성은 다음과 같다. 2 절에서는 관련연구 및 배경지식에 대한 설명을 하며, 3 절에서는 분기 정보의 투기적 사용에서 오는 적절한 복구 기법을 소개한다. 그리고 4 절에서는 실험 결과를 제시하며, 끝으로 5 절에서 결론을 맺는다.

2. 관련연구 및 배경지식

Yeh and Patt에 의해 제안된 이단계 적응형 분기 예측 기법에 대한 소개 이후로, 분기 명령어의 히스토리는 오늘날의 분기 예측에 있어서 중요한 입력 요소 중 하나로 자리 잡았다[1]. 또한 여기서 한걸음 더 나아가 이와 같은 히스토리들을 어떻게 활용하는가에 대한 연구도 진행되었다. 그 중 대표적인 것이 분기 히스토리에 대한 투기적 사용이다[2]. 다음과 같은 예제 코드를 살펴보자.

```
if (condA)          Br1
...
if (condB)          Br2
...
if (condA & condB)  Br3
...
```

이 경우, 마지막 분기 명령어(Br3)의 분기 방향은 Br1 혹은 Br2의 결과와 강하게 연관되어 있다. 즉 Br1과 Br2의 결과를 알고 있으면, Br3의 결과는 쉽게 예측이 가능하다. 하지만 서론에서 언급했듯이, 파이프라인의 단

계수가 늘어나고 단위 시간당 제기되는 명령어들의 수가 증가함에 따라, 각각의 파이프라인 단계 내부에서 현재 수행 중인 분기 명령어(In-Flight Branch)들이 다수 발생하게 된다. 이럴 경우 해당 분기 명령어들의 분기 정보가 아직까지 분기 히스토리 레지스터(History Register)에 반영되지 못할 수 있다는 문제점이 발생한다.

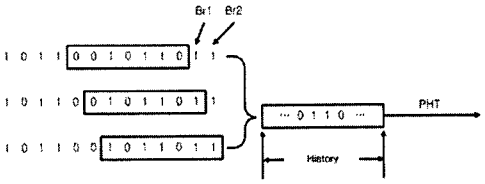

그림 1. 시간문제 사례

또한, 분기 히스토리를 투기적으로 사용하지 못할 경우는 *Hao et al.*에 의해 설명된 시간문제(Timing Problem)가 발생한다[3]. 즉, 동일 분기에 대한 분기 예측 시에 매번 서로 다른 입력 요소의 조합이 사용될 수 있다는 것이다. 그림 1에 이와 같은 내용이 묘사되어 있다. 이러한 시간문제는 동일한 분기 명령어에 대해 반복적으로 수행이 일어난다 하더라도, 캐쉬의 접근실패(Cache Miss), 이전의 다른 분기 명령어에 대한 예측 실패와의 상호 연관성, 명령 윈도우(Instruction Window) 자원의 부족 등, 여러 원인에 의해서 발생한다. 분기 히스토리의 투기적 사용은 이와 같은 문제점들을 효과적으로 해결 할 수 있으며, 그로 인해 분기 예측 정확도의 향상을 가져올 수 있다.

3. 분기 정보의 투기적 사용에 대한 복구 기법

분기 정보의 투기적 사용은 전술 한 바와 같이, 분기 예측의 정확도 향상에 기여한다. 하지만 이와 같은 투기적 정보는 완전히 수행 종료된 분기의 최종적인 정확한 정보를 의미하는 것이 아니다. 따라서 잘못된 투기적 분기 히스토리를 사용했을 경우, 이에 대한 적절한 복구 기법이 필요하다.

그림 2는 전역 히스토리(Global History)를 사용하는 분기 예측 모듈에서, 투기적 분기 정보의 사용과 이에 대한 복구 모듈이다. 본 논문에서는 분기 예측 기법에 있어서 성능 향상의 기본 예측기로 주로 사용되는 *gshare* 예측기를 사용하여 구현하였다[4]. 기존의 *gshare* 방식은 분기 히스토리를 저장하는 레지스터를 하나만 사용하였지만, 본 논문에서는 분기 정보의 투기적 사용을 위해 투기적 정보만을 별도로 저장하는 레지스터(S-GHR)를 추가로 사용한다. 그리고 실제 분기 명령어의 분기 예측 시에, 필요한 S-GHR 비트수 만큼 기존의 오래된 히스토리를 제외한 나머지와 추가적으로 병합하여 사용하게 된다. 이때 제외된 S-GHR 비트수 만큼의 기존 히스토리는 백업(Backup History for Recovery)되고 사용되지 않는다. 이는 큐(Queue)를 이용하여 복구 시 필요한 정보를 백업시킨 관련 연구와 비교하여, 평균 90% 이상의 하드웨어 복잡도 절감효과를 가진다[5].

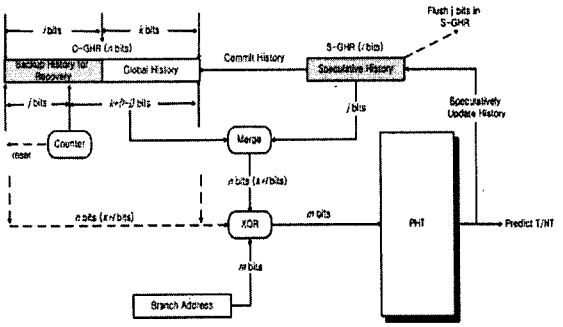

그림 2. 전역 분기 히스토리에 대한 복구 기법

지역 분기 히스토리(Local History)를 사용하는 경우는 이전 기록 기반(History-Based)과 이후 기록 기반(Future-based)의 두 가지 투기적 정보 사용 방식이 있을 수 있다[5]. 그림 3은 그 중에서 이전 기록 기반을 사용하는 방식으로, 분기 정보의 투기적 사용과 이에 대한 복구 모듈을 나타낸다. 제안된 방식은, 분기 예측을 수행할 때, 새롭게 생성된 투기적 정보를 분기 히스토리 테이블(BHT)에 즉시 반영한다. 그리고 이때 가장 오래된 히스토리 1 비트는 큐(Backup History Queue for Recovery)에 저장되게 된다. 그리고 이 정보는 투기적 정보가 잘못 사용 되었다고 판단될 경우 그림 3의 점선에서 나타나듯이, 큐의 역순으로 해당 BHT 엔트리를 복구하는데 사용된다. 이러한 이전 기록 기반 방식은, 각 단계별 복구가 필요하지만, 분기 예측이 용이하다는 장점이 있다. 한편 이후 기록 기반을 사용하는 방식에서는, 상대적으로 BHT가 완료된 히스토리(Committed History)들만을 보관하고 있으며, 분기 명령어의 투기적 히스토리들에 대한 정보는 별도의 큐에 보관하고 있다. 이후 기록 기반 방식은 단순한 복구 기법을 지원하지만 분기 예측 경로를 늘이는 단점이 있다.

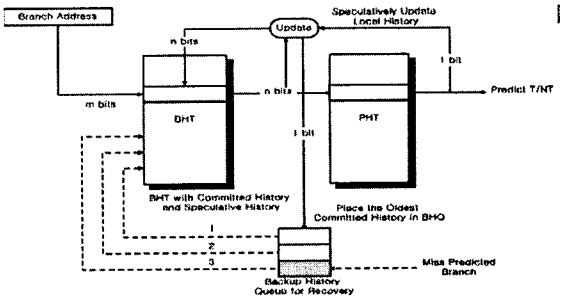

그림 3. 지역 분기 히스토리에 대한 복구 기법

4. 성능 평가

본 논문에서 제안된 방식의 성능을 평가하기 위해 모의실험을 수행하였다. 본 논문에서의 모의실험은 구동 기반 시뮬레이터인 *SimpleScalar*로 진행되었다[6]. 모의실험에 사용된 벤치마크 프로그램은 *SPEC CINT2000*에서 제공되는 프로그램들이다[7].

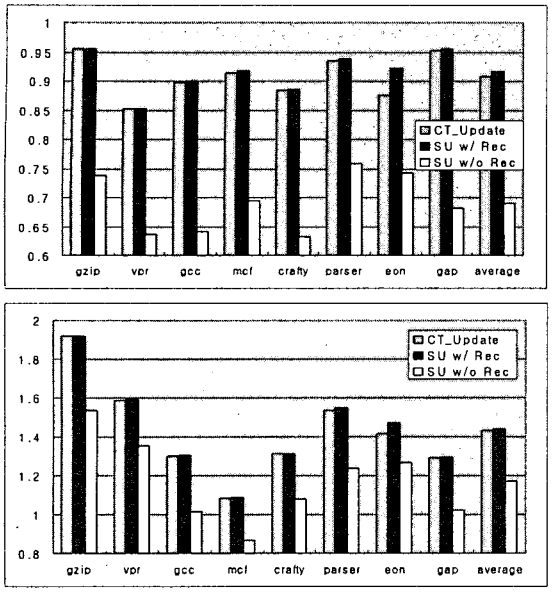

그림 4. 전역 히스토리의 투기적 사용 결과

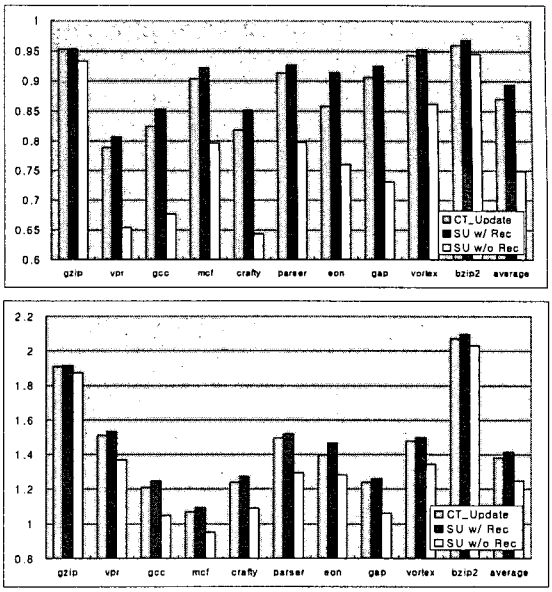

그림 5. 지역 히스토리의 투기적 사용 결과

그림 4는, 그림 2에서 제시된 투기적 분기 히스토리를 사용하며 이에 대한 복구 기법을 적용시켰을 경우의 분기 예측 정확도와 IPC에 대한 실험 결과이다. CT_Update는 투기적 방법을 사용하지 않는 기존의 방식이며, SU w/ Rec는 투기적 기법의 사용과 아울러 본 논문에서 제시된 복구 기법을 사용하는 방식이다. 끝으로 SU w/o Rec는 투기적 기법을 사용하지만 제안된 복구 기법을 사용하지 않는 방식을 뜻한다. 본 논문에서 사용된 PHT의 크기는 1024 이다. 그림 4에서 보이듯이 적절한 복구 기법이 제공된 투기적 분기 히스토리의 사용은 분기 예측의 정확도 측면에서 평균 2.3%의 성능 향상을 가져왔으며, 응용에 따라서는 최대 3.3%의 예측 정확도 향상을 보였다. 끝으로 SU w/o Rec의 결과에서 알 수 있듯이, 적절한 복구 기법이 제공되지 않는 투기적 분기 히스토리의 사용은 의미가 없다는 것을 알 수 있다.

그림 5는 그림 3에서 제시된 방식을 적용했을 경우의 분기 예측 정확도와 IPC의 실험 결과이다. 전체적인 결과의 추세는 그림 4와 유사하다. 다만, 성능 향상의 정도가 상대적으로 작다는 것을 알 수 있다. 이는 지역 분기 히스토리를 사용하는 분기 예측의 경우, 투기적 분기 히스토리가 시스템 전체적인 정보가 아니라 특정 분기 명령어의 지역적 투기 정보이기 때문이다. 이러한 결과는 지역 분기 예측기의 고유한 특징이며, 분기 정보의 투기적 사용이 전역 분기 예측 기법에서 상대적으로 보다 더 효과적일 수 있음을 뜻한다.

5. 결 론

본 논문에서는, 분기 예측의 정확도를 높이기 위한 기법의 하나로 분기 정보의 투기적 사용에 대한 필요성을 설명하고, 이에 대한 적절한 복구 기법을 제안하였다. 그리고 제안된 기법은, 전역 히스토리와 지역 히스토리의 사용 여부에 따라 서로 다르게 적용될 수 있었다.

실험 결과 제시된 기법은 프로그램 수행의 정확성을 해치지 않으면서 분기 정보의 투기적 사용으로 인한 성능 향상을 제공하였다. 또한, 기존의 연구와 비교하여 보다 더 효율적인 하드웨어 구현 복잡도를 나타내었다.

참고 문헌

[1]. T.-Y. Yeh and Y. N. Patt, "Alternative implementations of two-level adaptive branch prediction," In Proc. of the 19th ISCA, pp. 124-134, May, 1992.

[2]. A. R. Talcott,W. Yamamoto,M. J. Serrano, R. C.Wood, andM. Nemirovsky, "The impact of unresolved branches on branch prediction scheme performance," in Proceedings of the 21st Annual International Symposium on Computer Architecture, pp. 12-21, Apr. 1994.

[3]. E. Hao, P.-Y. Chang, and Y. Patt, "The effect of speculatively updating branch history on branch prediction accuracy, revisited," in Proceedings of the 27th Annual International Symposium onMicroarchitecture, pp. 228-232, Nov. 1994.

[4]. McFarling, S., "Combining branch predictors. Tech. Rep. TN-36m", Digital Western Research Lab., June, 1993

[5]. K. Skadron, M. Martonosi, and D. Clark. "Speculative updates of local and global branch history: A quantitative analysis", JILP, vol. 2, Jan. 2000

[6]. D. Burger, T. M. Austin, and S. Bennett, "Evaluating future micro-processors: the SimpleScalar tool set", Tech. Report TR-1308, Univ. of Wisconsin-Madison Computer Sciences Dept., 1997

[7]. SPEC CPU2000 Benchmarks, http://www.specbench.org