종속 트리와 상대적 병렬도를 이용하는 수퍼스칼라 프로세서의 정수형 성능 예측 모델

이 종 복
한성대학교 정보통신공학과
jblee@hansung.ac.kr

The Integer Superscalar Processor Performance Model Using Dependency Trees and the Relative ILP

Jongbok Lee
Dept. of Information and Communications Engineering, Hansung University

요 약

최근에 이르러 프로세서의 병렬성을 분석적 기법으로 예측하기 위한 연구가 활발해지면서 프로세서의 성능 예측 모델에 대한 중요성이 대두되고 있다. 그러나 기존의 연구는 현재 광범위하게 사용되고 있는 다중 분기 예측법을 이용하는 프로세서에 대하여 분기 차수와 관계없는 재귀적 성능 모델을 제공해주지 않는다. 본 논문에서는 이것을 해결하기 위하여, 매 싸이클마다 명령어 종속 트리를 구성하고 종속인 명령어 간에 상대적인 병렬도 값을 부여하여 성능 예측 모델 입력 데이타를 측정하였다. 그 결과, 다중 분기 예측법을 사용하는 프로세서에서 정수형 프로그램에 대한 성능을 기존의 성능모델보다 작은 상대 오차로 예측할 수 있다.

1. 서 론

프로세서의 성능 예측 모델을 얻는 방법은 종속분석기를 이용하여 특정한 벤치마크의 명령어 자취에 대한 성능 모델 입력 데이타를 얻은 후에, 이 입력 데이타들을 성능 모델에 입력하므로써 해당 벤치마크의 성능을 예측하는 두 단계로 구성된다. 그러나 기존 연구에서는 현재 광범위하게 사용되고 있는 다중 분기 예측법에 대하여 분기 차수와 관계없는 재귀적 성능모델이 존재하지 않는다. 본 논문에서는 명령어들을 단순 검색하여 병렬성 데이타를 얻는 기존 연구방식에서의 오차를 줄이기 위하여, 매 싸이클마다 윈도우 내 명령어 간의 종속 트리를 구성해서 명령어에 상대적인 종속도를 부여하는 방식을 제안하였다. 이렇게 하여 구한 파라미터를 성능 모델에 입력한 결과, 다중 분기 예측법을 사용하는 정수형 프로그램의 성능을 기존의 방식보다 더욱 정확하게 예측할 수 있다.

2. 기존의 재귀적 성능 예측 모델

충분히 긴 명령어의 흐름에 대하여 두 명령어가 독립일 확률이 일정하다고 가정하면, 두 명령어가 동시에 실행될 확률은 오직 두 명령어가 떨어진 거리에 의해서만 결정된다. 동적 명령어의 흐름 $I_0, I_1, I_2, \cdots, I_{W-1}$로 구성된 크기가 W인 윈도우에 대해서 살펴볼 때, 두 개의 명령어가 조건부 독립일 확률은 오직 두 명령어 간의 거리만의 함수이며, 이것이 p_δ로 일정하다고 가정하였다 [1]. 본 논문에서 p_δ를 이하 병렬도라고 부르기로 한다. 이 때, 윈도우 $I_0, I_1, \cdots, I_{W-1}$에서 정확히 k개의 명령어를 실행하는 확률인 $P(W,k)$는 다음과 같은 재귀함수로 나타낼 수 있다.

$$\begin{aligned} P(W,k) &= P(W-1,k-1) \times p_\delta^W \\ &+ P(W-1,k) \times (1-p_\delta^W) \\ P(W,1) &= 1 \end{aligned}$$

$$\begin{aligned} P(W,0) &= 0 \\ P(W,i) &= 0 \qquad (i > W) \end{aligned} \tag{1}$$

따라서 매 싸이클 마다 실제로 인출되는 명령어의 개수가 g_i이고 그 최대값이 인출율인 k일 때, 수퍼스칼라 프로세서의 ipc는 다음과 같다.

$$ipc = g_k \times P(W,k) + \sum_{i=1}^{k} g_i \times P(W,i) \tag{2}$$

따라서, 윈도우의 크기가 W이며, 인출율이 k인 수퍼스칼라 프로세서에 대한 벤치마크 프로그램의 병렬도가 p_δ일 때, 그 성능을 얻을 수 있다. 그러나 위의 연구에서 실제로는 퇴거되어야 할 명령어들을 그대로 놔둔채 병렬도 p_δ를 측정하므로 이 명령어들이 윈도우의 맨 앞으로 이동하여 밖으로 밀려날 때까지 계속 남아서 병렬도를 낮춘다.

3. 성능 모델

본 논문에서는 병렬도를 측정할 때 매 싸이클마다 상호 종속인 명령어 간에 명령어 종속트리를 구성하고, 이것을 이용하여 각 명령어에 상대적인 병렬도를 부여하는 방법을 제안하였다. 이렇게 얻어진 병렬도를 이용하여 최종 ipc를 구하는 것은, 기존의 재귀함수식 1과 2를 이용하였다.

3.1 정수형 종속트리의 정의

각 싸이클마다 윈도우에 있는 모든 명령어 중에서 어느 명령어에도 종속이 아닌 명령어를 뿌리노드라고 명명하며, 뿌리노드가 정수형 명령어인 경우 한 싸이클 만에 실행 가능하므로 모든 뿌리노드의 병렬도는 1이다. 이러한 뿌리노드에서부터 출발하여 이 노드에 종속인 명령어들을 만날 때마다 자식노드로 연결한 트리를 구성한다. 이 때, 각 뿌리노드의 깊이를 1로 정의하

고 자식노드를 만날 때마다 그 깊이를 차례대로 1씩 증가한다. 이와같이 매 싸이클마다 윈도우내 명령어들에 대한 종속트리를 생성한 후에, 각 노드의 상대 병렬도를 계산하였다. 임의의 노드가 뿌리노드에서부터 임의의 경로에 대하여 갖는 지역 병렬도를 p_δ, 깊이를 d라 할 때 이들은 반비례 관계가 있으므로 다음과 같이 놓을 수 있다.

$$p_\delta = \frac{1}{d} \tag{3}$$

한편, 레지스터의 종속관계에 따라 뿌리노드에서 그 노드에 이르는 경로가 2 가지 이상일 때에는 종속도가 큰 것이 그 노드의 종속성을 결정하므로, 식 4와 같이 그 깊이가 최대인 것을 택하였다.

$$p_\delta = \frac{1}{max(d_1, d_2, d_3, \cdots, d_n)} \tag{4}$$

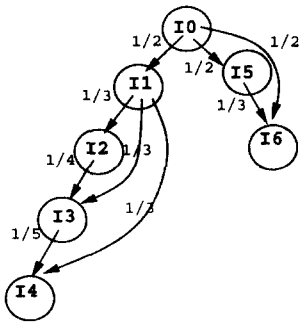

그림 1: 종속트리의 예.

그림 1은 명령어 상대 종속 트리의 예이다. I_0는 뿌리노드이므로 병렬도가 1이고 깊이도 1이다. I_1은 깊이가 2이므로 병렬도가 $\frac{1}{2}$이며, I_2는 깊이가 3이므로 병렬도가 $\frac{1}{3}$이다. 한편, I_4는 깊이가 5인 경로와 3인 경로를 동시에 갖는다. 따라서 병렬도는 깊이가 큰 값을 취해서 $\frac{1}{5}$이다. 이와같이 상대 종속 병렬도를 도입한 종속 트리 방식은 명령어의 단순 통과에 의하여 얻은 병렬도에 의하여 성능이 낮게 나오는 것을 보완한다. 이렇게 해서 얻은 병렬도는 식 1과 2에 입력하므로써 프로세서의 성능을 구할 수 있다.

4. 모의실험 환경 및 방법

4.1 수퍼스칼라 프로세서의 구조

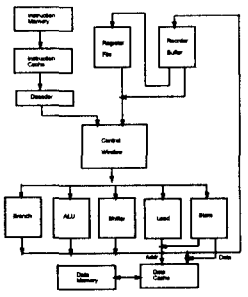

그림 2: Smith and Johnson's superscalar processor model.

본 논문에서는 그림 2에 나타낸 Smith와 Johnson의 수퍼스칼라 프로세서 모델을 기본으로 하며 [2], 한 싸이클에 여러 개의 명령어를 수행시키기 위하여 다중 연산 유닛이 이용된다. 인출된 명령어는 해독 및 재명명 되어 명령어 윈도우에 할당이 되며, 재명명과 정확한 인터럽트 (precise interrupt)를 위하여 리오더 버퍼 (reorder buffer)가 사용된다. 이 때, Tomasulo 알고리즘에 의하여 명령어들이 비순차적으로 연산 유닛으로 이슈 및 수행될 수 있다. 그동안 연구가 수행된 동적 분기 예측법 중에서 이중 적응형 분기 예측법이 비교적 높은 예측 정확도를 나타낸다 [3]. 본 논문에서 사용한 다중 분기 예측법은 전역 히스토리 방식을 사용하는 이중 적응형 분기 예측법을 응용한 것이다 [4].

4.2 수퍼스칼라 프로세서의 사양

본 실험에 사용한 수퍼스칼라 아키텍쳐의 특성은 Sun 사의 Supersparc 프로세서를 모델로 하였다. 명령어 파이프라인은 인출, 해독, 이슈, 실행, 쓰기 및 결과 회부의 6 단계로 구성된다. 프로세서는 이슈 단계에서 윈도우 내의 명령어들을 검색하여 실행을 위하여 연산 유닛으로 보내며, 쓰기와 결과 회부 단계에서 명령어 실행 결과를 각각 리오더 버퍼와 레지스터 화일에 기록한다. 윈도우는 64의 크기를 사용하였으며, 리오더 버퍼의 크기는 완전한 레지스터 재명명이 이루어지도록 충분히 크게 정하였다. 각 싸이클 당 예측되는 분기 명령어의 개수는 1 개, 2 개 및 3 개까지 실험하였다. 각 연산 유닛의 개수는 분기처리 유닛을 제외하고는 무한하다고 가정하였다.

4.3 모의실험 방법

본 논문에서는 수퍼스칼라 프로세서의 성능을 측정하는데 SPEC89 및 SPEC92 벤치마크에 대하여 명령어 자취 모의실험을 이용하였다. 사용된 벤치마크 프로그램들은 *eqntott, espresso, xlisp, gcc, go, ijpeg, mk88sim, perl*의 8 개의 정수형 프로그램이다. 이들을 C로 컴파일하여 얻은 오브젝트 코드를 명령어 자취 발생기인 Shadow [5]에 대한 입력으로 하여 각 벤치마크 마다 어셈블리어 형태의 명령어 자취 일천만 개를 발생시켰다. 이와 같이 생성된 명령어 자취들을 다중 분기 예측법을 사용하는 수퍼스칼라 프로세서 모의 실험기로 입력하였다.

5. 모의실험 결과

5.1 다중 분기 예측 정확도

표 1에 정수형 벤치마크의 분기 예측 정확도를 나타내었다. 각 벤치마크에 대하여 1 차, 2 차, 3 차의 다중 분기 예측을 시행할 때의 정확도를 나타내는데, 2 차 분기 예측인 경우 1 개의 분기 명령어만 예측이 맞을 때 (B_{21}), 2 개의 분기 명령어가 연속적으로 맞을 확률 (B_{22})을 나타내었고, 3 차 분기 예측의 경우에는 각각 1 개 (B_{31}), 2 개 (B_{32}), 3 개의 분기명령어가 연속적으로 맞을 확률 (B_{33})을 나타내었다. 이 분기 예측 정확도는 명령어 자취에 정보로서 포함되어, 분기 예측의 옳고 그름에 따라 병렬도를 측정하는 기본블럭을 포함시키거나 제외시킨다.

5.2 상대적 병렬도

그림 3은 윈도우의 크기가 64일 때 1 차, 2 차, 3 차의 다중 분기 예측법을 시행했을 때의 *xlisp*에 대한 상대적 병렬도를 나타낸다. 각 병렬도는 윈도우 상에서의 위치에 따라, 첫번째 명령어인 경우 1.0에서부터 마지막 명령어인 64로 갈 수록 점차 0.0으로 감소한다. 이들 정수형 벤치마크 중에서 *go, gcc, espresso*가 상대적으로 높은 병렬도를 나타내며, *mk88sim, perl*은 중간 정도, *eqntott, xlisp*는 낮은 병렬도를 나타내었다. 전체 프로그램에 대하여 1 차, 2 차, 3 차의 다중 분기 예측법을 시행함에 따라 그래프가 우측으로 이동하여 병렬도가 증가함을 알 수 있다.

표 1: 다중 분기 예측 정확도(정수형)[%]

벤치마크	1분기	2분기		3분기		
		B_{21}	B_{22}	B_{31}	B_{32}	B_{33}
eq	94.63	89.40	5.17	83.06	5.17	1.45
es	92.56	85.74	6.81	78.13	6.06	2.75
li	95.18	90.66	4.66	86.01	3.64	2.05
gc	90.88	83.44	7.17	77.42	6.32	2.47
go	91.42	83.40	7.90	77.38	6.17	1.29
ij	80.50	62.02	8.60	27.82	68.55	0.33
mk	90.86	80.74	9.26	72.88	6.82	1.23
pe	92.86	85.99	6.34	80.19	5.56	2.51

다중 분기 예측에 의하여 증가된 병렬도는 증가된 프로세서의 성능으로 나타나게 될 것을 예측할 수 있다.

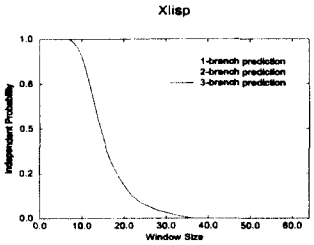

그림 3: Xlisp의 상대적 병렬도

5.3 측정한 성능과 모델에 의하여 산출한 성능의 비교

그림 4는 정수형 벤치마크에 대하여 종속 트리를 이용하지 않은 기존 모델에 의한 성능, 명령어 자취 모의실험에 의하여 측정한 성능 및 본 논문에서 제안하는 종속 트리를 이용한 모델에 의하여 예측한 성능을 함께 나타낸 것이다. 각 프로그램에 대하여 1 차, 2 차, 3 차의 다중 분기 예측을 시행하였을 때 기존 방식에 의한 값을 흰색 막대 그래프로, 측정에 의한 값을 그 차이만큼 연한 색의 막대 그래프로, 그리고 본 논문에서 제안하는 모델에 의한 값을 그 차이만큼 진한색의 막대 그래프로 도시하였다.

전체 프로그램에 대하여 종속 트리를 이용하는 상대 병렬도 모델에 의한 값이 측정에 의한 값보다 일관적으로 높으며, 반면에 종속 트리를 사용하지 않는 기존의 모델에 의한 값이 측정에 의한 값보다 일관적으로 낮음을 알 수 있다. 종속 트리를 사용하지 않는 기존의 성능 예측 모델에 의하면 정수형 벤치마크에서 1 차의 분기예측일 때 측정한 성능에 비하여 20.5 %의 상대오차를 나타낸 반면, 본 논문에서 제안하는 상대 종속 트리 방식은 이보다 작은 15.4 %의 상대오차를 나타냈다. 또한 2 차 및 3 차의 다중 분기 예측에서는 기존의 방식이 각각 29.0 %과 27.0 %의 상대오차를 나타낸 반면에, 상대종속트리 방식은 각각 11.0 %와 20.8 %를 나타냈다. 따라서 본 논문에서 제안하는 방식이 기존의 방식보다 성능을 예측함에 있어서 그 상대오차가 더 작기 때문에 더욱 정밀한 모델임을 알 수 있다.

6. 결론 및 향후 연구과제

본 논문에서는 다중 분기 예측 차수와 관계없이 적용 가능한 수퍼스칼라 프로세서의 정수형 성능 예측 모델을 제안하였다. 매 싸이클 당 윈도우마다 명령어 간의 종속관계를 분석하여 종속 트리를 구성하므로써, 이슈 가능하지 않은 명령어에도 상대 종속 병렬도를 부여하였다. 이와같은 프로세서의 예측 성능 모델 방식을 이용하여 얻은 결과는 측정값 대비 평균 상대오차가 정수형에서 15.7 % 를 나타내어, 25.5 %를 나타낸 기존의 방식보다 낮았다. 향 후의 연구과제로는 실행에 여러 싸이클이 소요되는 실수형 명령어로 구성된 프로그램에 대한 성능 모델을 개발하는 것이다.

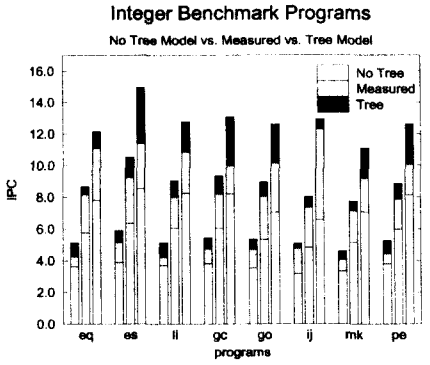

그림 4: 정수형 벤치마크에 대한 결과

참고 문헌

[1] P.K. Dubey, G.B. Adams III, and M. J. Flynn, "Instruction Window Size Trade-Offs and Characterization of Program Parallelism," *IEEE Transactions on Computers*, vol. 43, pp. 431–442, Apr. 1994.

[2] M.D. Smith, M. Johnson, and M.A. Horowitz, "Limits on Multiple Instruction Issue," in *Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems*, Apr. 1989, pp. 290–302.

[3] T-Y. Yeh and Y.N. Patt, "Two-Level Adaptive Branch Prediction," in *The 24th ACM/IEEE Intl Sym and Workshop on Microarchitecture*, Nov. 1991, pp. 51–61.

[4] T-Y. Yeh, D. T. Marr, and Y.N. Patt, "Increasing the Instruction Fetch Rate via Multiple Branch Prediction and a Branch Address Cache," in *International Conference on Supercomputing '93*, 1993, pp. 67–76.

[5] *Introduction to SHADOW*, Sun Microsystems. Inc., Jul. 1989.