ISSN 1975-8359(Print) / ISSN 2287-4364(Online)
The Transactions of the Korean Institute of Electrical Engineers Vol. 63, No. 12, pp. 1704~1709, 2014
http://dx.doi.org/10.5370/KIEE.2014.63.12.1704

퍼셉트론을 이용하는 멀티코어 프로세서의 성능 연구

A Performance Study of Multi-Core Processors with Perceptrons

이 종 복*
(Jongbok Lee)

Abstract - In order to increase the performance of multi-core system processor architectures, the multi-thread branch predictor which speculatively fetches and allocates threads to each core should be highly accurate. In this paper, the perceptron based multi-thread branch predictor is proposed for the multi-core processor architectures. Using SPEC 2000 benchmarks as input, the trace-driven simulation has been performed for the 2 to 16-core architectures employing perceptron multi-thread branch predictor extensively. Its performance is compared with the architecture which utilizes the two-level adaptive multi-thread branch predictor.

Key Words : Perceptron, Multi-core processor, Multi-thread branch predictor

1. 서 론

최근 멀티코어(multi-core) 프로세서가 스마트폰, 태블릿 PC, 노트북, 데스크탑 등과 같은 컴퓨터 시스템의 성능 향상을 높이기 위하여 광범위하게 이용되고 있다[1-3]. 멀티코어 프로세서에 멀티쓰레드 분기예측을 시행함으로써, 응용 프로그램의 병렬화를 통하여 프로세서의 성능을 높일 수 있다. 멀티쓰레드 분기예측을 통하여 프로그램을 쓰레드로 분할하고 각 코어에 할당할 때, 멀티쓰레드 분기예측 정확도가 성능에 큰 영향을 끼친다. 기존의 멀티쓰레드 분기예측 방식은, 각 쓰레드의 시작주소가 과거에 실제로 실행되었는가의 여부를 히스토리 패턴으로 기록하고, 그 히스토리 패턴을 기반으로 다음 쓰레드의 시작주소를 예측하는 2 단계 적응형 분기예측 방식을 응용한 것이다[4-6].

최근에 이르러 수퍼스칼라 프로세서에 신경망 회로 알고리즘을 도입하여 분기예측의 정확도를 높이는 방안이 시도되었다 [7,8]. 본 논문에서는 신경망 분야에서 활용되는 퍼셉트론을 멀티코어 프로세서의 멀티쓰레드 분기예측에 적용하는 것을 제안하였다. SPEC 2000 벤치마크 프로그램을 대상으로 하여 모의실험을 수행한 결과, 기존의 2 단계 적응형 멀티쓰레드 분기예측법과 비교하여 더욱 높은 성능을 나타낼 수 있었다.

본 논문은 다음과 같이 구성된다. 2장에서 퍼셉트론 멀티쓰레드 분기예측기 및 그 기술에 대한 멀티코어 프로세서에 대한 적용을 살펴보고, 3 장에서 모의실험 환경에 대하여 고찰한다. 그리고 4 장에서 모의실험 결과를 보이며, 5 장에서 결론을 맺는다.

* Corresponding Author : Dept. of Information and Communications Engineering, Hansung University, Korea
E-mail: jblee@hansung.ac..kr
Received : April 25, 2014; Accepted : November 19, 2014

2. 퍼셉트론

2.1. 퍼셉트론의 원리 및 학습 방법

퍼셉트론이란 그림 1에 나타낸 것과 같이, 입력 값들을 가중치와 결합하여 출력을 산출하는 학습 기능을 갖는 신경망이다. 이 때, $x_1,..,x_n$은 입력, $w_0...w_n$은 가중치, y는 출력을 나타내며, 퍼셉트론은 양 또는 음의 정수로 나타나는 가중치를 요소로 갖는 벡터로 구성된다. 한편, 퍼셉트론의 출력 y는 가중치 벡터 $w_{0..N}$과 입력벡터 $x_{0..N}$의 내적 (dot product)으로 계산하며, 수식 1과 같이 표현된다. 이 때, x_0는 항상 1로 설정함으로써, 상관 관계를 학습하기 이전에 초기 바이어스 가중치 w_0는 히스토리와 상관없이 그 사건이 발생하는 것으로 학습한다.

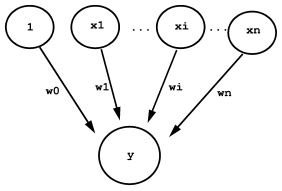

그림 1 퍼셉트론 모델
Fig. 1 The perceptron model

추후에 사건에 대한 실제 결과가 알려지면, 그 결과와 출력 y 값을 가지고 아래의 표 1에 나타낸 학습 알고리즘을 이용하여 가중치 wi를 수정함으로써 퍼셉트론을 학습시킨다.

본 알고리즘에서 사건이 발생하였을 때 t의 값을 +1, 발생하지 않았을 때 t의 값을 -1로 설정하는데, 이 때 Θ는 학습이 충분히 이루어졌는가를 판단하는데 이용하는 임계값이다. 퍼셉트론에 대한 학습 여부는, 출력 y의 부호와 t의 값을 비교하여 그 값이 다르거나, 출력 y의 절대값이 임계값보다 작은 경우에 결정된다. 그리고 학습이 결정되었을 경

우. 퍼셉트론의 가중치 벡터를 수정함으로써 학습이 이루어진다.

표 1 퍼셉트론 학습 알고리즘
Table 1 The perceptron learning algorithm

```
if sign(y_out) != t or |y_out| <= Θ then
        for i = 0 to n do
         w_i = w_i + tx_i
        end for
      end if
```

$$y = w_o + \sum_{i=1}^{n} x_i w_i \quad (1)$$

2.2 멀티코어 프로세서에서의 멀티쓰레드 분기예측법

그림 2는 멀티코어 프로세서에서 프로그램을 실행시킬 때 멀티쓰레드와의 관계를 나타낸 것이다. 임의의 프로그램은 동적 명령어의 흐름 쓰레드 1, 쓰레드 2, ..., 쓰레드 N으로 구성된다. 이 때 각 코어마다 최대 i 개의 쓰레드가 프로그램의 동적 흐름을 예측하여 코어에 할당되어 실행된다.

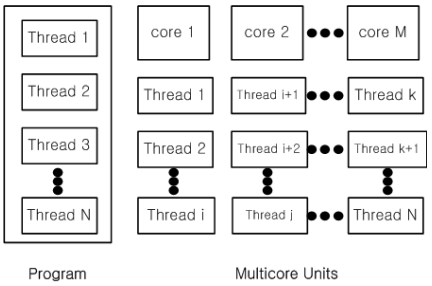

그림 2 멀티코어 프로세서에서 멀티쓰레드의 관계
Fig. 2 The relation between the multi-thread and the multi-core processor

각 코어 내에서 명령어들은 실행 성능을 높이기 위하여 비순차 실행되지만 프로그램의 원 순서를 보존하기 위하여 순차완료(in-order commit) 되며, 각 코어 간에 대해서도 순차완료 된다. 또한, 동적으로 코어 내부 및 코어 간 레지스터 종속과 메모리 모호성 제거 (memory disambiguation)를 검사하여 올바른 실행을 할 수 있다.

각 코어에서 쓰레드를 인출하고 실행하기 위한 멀티쓰레드 분기예측 알고리즘으로 가장 간단한 것은 2 단계 적응형 분기예측법을 멀티쓰레드 분기예측에 응용한 것이다. 2 단계 적응형 멀티쓰레드 분기예측법은 쓰레드 히스토리 레지스터에 최근의 쓰레드 경로 결과를 기록하고, 본 레지스터의 값으로 쓰레드 패턴 히스토리 테이블을 접근하여 2 비트 포화 카운터의 값에 따라 쓰레드 경로의 채택 여부를 예측한다. 이 때, 각 쓰레드의 시작 어드레스를 기록하기 위하여 쓰레드 어드레스 캐쉬를 운용한다. 일반적으로, N 개의 쓰레드 예측을 수행할 때, 첫 번째, 두 번째, ..., N 번째 쓰레드에 대하여, 쓰레드 히스토리 레지스터의 k 비트, k-1 비트, ..., k-N+1 비트로 쓰레드 패턴 히스토리 테이블을 각각 색인하여 다음 쓰레드의 시작 어드레스를 예측한다.

2.3 퍼셉트론을 이용한 멀티쓰레드 분기예측법

본 논문에서는 신경망 회로에서 이용하는 학습 방식을 멀티쓰레드 분기예측 기법에 적용하였으므로, 이것을 퍼셉트론 멀티쓰레드 분기예측법이라 한다. 이것을 위하여 길이 N의 쓰레드 어드레스 히스토리 레지스터에, N 개 쓰레드의 시작 어드레스가 직전 쓰레드의 마지막 어드레스와 불연속인 결과를 1로, 연속인 결과를 -1로 기록한다. 이렇게 하여 얻은 쓰레드 어드레스 N 개의 연속 및 불연속 패턴을, 1 또는 -1로 나타내는 현재 쓰레드 어드레스의 연속 또는 불연속인 결과와 곱하여 길이 N인 가중치 벡터를 생성한다. N 개의 멀티쓰레드에 대하여 생성된 가중치벡터들은 퍼셉트론 가중치벡터 테이블을 구성한다.

이와같이 쓰레드 히스토리 레지스터와 퍼셉트론 가중치 벡터 테이블이 마련된 후에, 다음의 쓰레드에 대한 예측을 수행하고자 할 때의 동작은 다음과 같다. 멀티쓰레드 분기예측기는 각 쓰레드의 마지막 어드레스를 이용하여 0 부터 N-1 사이의 인덱스 i를 생성하여 퍼셉트론 가중치 벡터 테이블을 접근한다. 그 결과, i 번째 퍼셉트론을 인출하여 가중치 벡터 $P_{0..N}$을 얻으며, 가중치 벡터와 쓰레드 어드레스 히스토리 레지스터와의 내적으로 출력 y를 계산하여, 그 부호에 따라 다음 쓰레드의 연속 여부를 예측한다. 그 연산 결과가 양이면 다음 쓰레드의 주소를 불연속으로 예측하고, 음이면 그것을 연속으로 예측한다.

추후에 쓰레드의 연속 여부에 대한 결과가 알려지면, 그 결과와 출력 y 값을 가지고 학습 알고리즘을 이용하여 가중치 벡터 $P_{0..N}$의 가중치를 수정한다. 쓰레드가 연속일 때 t의 값을 -1, 불연속일 때 t의 값을 1로 설정하며, 입력 x_i의 값 역시 -1 또는 1이므로, 표 1의 알고리즘에 의하여 쓰레드의 경로가 x_i와 일치할 때 i 번째 가중치를 증가시키고, 일치하지 않을 때 감소시킨다. 따라서, 퍼셉트론의 가중치 벡터가 계속적으로 업데이트되는데, 만일 임의의 쓰레드 어드레스가 계속 불연속이라면 가중치 벡터의 필드값이 점점 커지고, 연속이라면 그 값이 점점 작아지게 된다. 마지막으로 수정된 벡터 $P_{0..N}$을 테이블의 i 번째 항목에 기록하여 그 결과를 반영한다.

그림 3에 본 논문에서 제안하는 퍼셉트론 멀티쓰레드 분기예측기의 하드웨어 블럭도를 나타냈다. 퍼셉트론 멀티쓰레드 분기예측기는 쓰레드 히스토리 레지스터에 유한한 길이의 연속 또는 불연속 행태를 기록하며, 퍼셉트론 가중치 벡터표를 접근하여 다음 쓰레드의 경로에 대한 예측을 수행한다. 특히, M 개의 코어에 대하여 N 개의 쓰레드에 대한 동시 예측을 위하여, 쓰레드 히스토리 레지스터와 더불어 임시 쓰레드 히스토리 레지스터를 이용함으로써, 각 멀티쓰레드 예측의 초기 단계에 쓰레드 히스토리 레지스터의 내용을 그대로 복사한다.

멀티쓰레드 예측법을 시행하기 위하여, M 개의 코어로

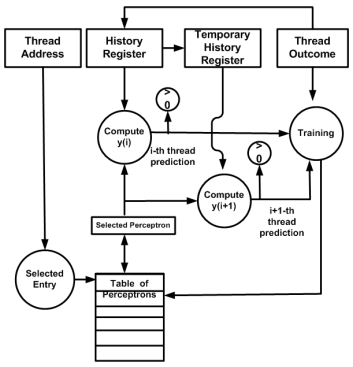

그림 3 퍼셉트론 멀티쓰레드 분기예측기
Fig. 3 The perceptron multi-thread branch predictor

구성되는 시스템에서 멀티쓰레드 i, i+1, ..., i+M-1 개가 존재한다고 가정한다. 이 때, 쓰레드의 주소를 퍼셉트론 가중치 벡터표의 항목 수를 이용하여 해싱한 값으로 특정한 가중치 벡터를 색인하고, 이 값을 쓰레드 히스토리 레지스터와 비트 단위로 곱하여 i 번째 쓰레드의 시작주소를 예측한다. i+1 번째 쓰레드의 시작주소는 i 번째 쓰레드의 시작주소에 대한 예측이 옳다고 가정을 하여 시행하는데, 이것을 위하여 임시 쓰레드 히스토리 레지스터의 i 번째 비트를 갱신하고 색인된 가중치 벡터와 비트 단위 곱셈을 시행한다. 각 멀티코어 프로세서 파이프라인의 실행 단계가 완료하여 쓰레드의 실제 경로가 밝혀지면, 그 결과에 따라서 쓰레드 히스토리 레지스터의 값을 갱신한다. 이 방식과 유사하게, i 번째와 i+1 번째 쓰레드의 시작주소를 기반으로 하여 임시 쓰레드 히스토리 레지스터의 i 번째와 i+1 비트를 갱신함으로써, i+2 번째 쓰레드의 시작주소를 예측한다. 따라서, 한 싸이클에 M 개의 코어에서 N 개의 쓰레드에 대한 시작주소를 동시에 예측하여 각 코어에 할당할 수 있다.

2.4 멀티코어 프로세서의 구조

그림 4는 M개의 코어로 구성되는 공유 메모리 (shared-memory) 멀티코어 프로세서의 일반적인 구조를 나타낸 것이다. 각 코어는 1부터 M까지 구성되는데, 자체적으로 1 차 명령어 캐쉬와 1 차 데이터 캐쉬를 가진다. 또한 각 코어는 메인 메모리와 연결되는 공통의 2 차 통합 캐쉬 (unified cache)를 공유한다. 멀티코어 프로세서를 구성하는 각 코어에 설치된 1 차 데이터 캐쉬의 일관성(cache-coherency)을 위하여 MESI 프로토콜을 이용한다. MESI 프로토콜은 캐쉬 블럭을 변경

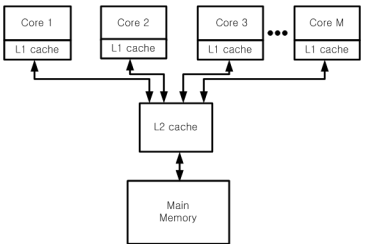

그림 4 멀티코어 프로세서의 구조
Fig. 4 The multi-core processor architecture

(Modified), 독점(Exclusive), 공유(Shared), 무효(Invalid)의 4 가지 상태로 관리한다. 이하 본 논문에서 기술하는 명령어 캐쉬와 데이터 캐쉬는 모두 1 차 캐쉬를 의미한다.

각 코어 내에서의 레지스터 재명명 (register renaming)이 수행되며, 서로 다른 코어 간의 레지스터 재명명을 위하여, 한 코어의 레지스터 값을 다른 코어와 공유한다. 만일에 어느 코어에서 레지스터 값이 새로 기록되지 않는다면 그대로 다음 코어와 공유가 되지만, 기록된다면 이전 레지스터 값의 공유가 중단되고 그 대신 새로운 레지스터 값을 공유한다. 각 코어 내에서의 메모리 종속은 기존의 방법으로 수행되며, 상이한 코어 간의 메모리 종속은 주소 해결 버퍼(Address Resolution Buffer)를 이용하여 관리된다[9].

데이터 캐쉬의 경우, 여러 코어 간의 캐쉬 일관성(Cache Coherency)을 위한 MESI 프로토콜의 적용으로 인하여 캐쉬의 데이터를 무효화 (write-invalidate)하는 경우가 빈번히 발생한다. 본 논문에서는 멀티코어 프로세서 아키텍처 성능의 극대화를 위하여, 명령어 캐쉬와 데이터 캐쉬를 충분한 용량의 2 차 연관 구조로 구성하였다.

3. 모의실험 환경

3.1 멀티코어 프로세서 모의실험기

멀티코어 프로세서 모의실험기는 명령어 트레이스를 기반으로 개발되었다. 모의실험은 제 1 단계 명령어 트레이스의 발생, 제 2 단계 명령어 트레이스에 대한 멀티코어 프로세서의 실행으로 나누어진다. 제 1 단계에서 명령어 트레이스는 Simplescalar를 이용하여 쓰레드 단위 병렬성을 임의의 코어 개수를 갖는 멀티코어에 대응시키는데 적합하도록 발생되었다 [10]. 제 2 단계에서 각 코어가 비순차 수퍼스칼라 프로세서로 동작하여 멀티코어 프로세서가 실행되며, 그림 5에 모의실험기의 내부 순서도를 나타냈다.

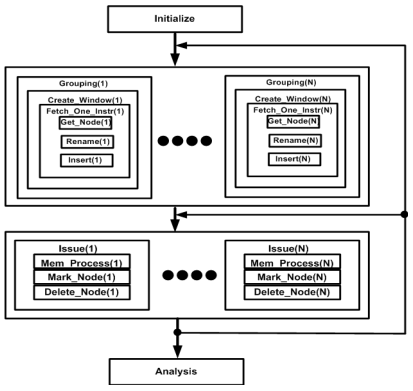

그림 5 멀티코어 프로세서 모의실험기 내부의 순서도
Fig. 5 The internal flow chart of multi-core processor architecture simulator

3.2. 벤치마크 및 멀티코어 프로세서의 사양

표 2는 모의실험에 이용된 SPEC 2000 정수형 벤치마크 프로그램이다. Simplescalar를 통하여 MIPS IV 10억 개의

명령어 트레이스를 발생시키고, 멀티코어 프로세서 모의실험기에서 퍼셉트론 멀티쓰레드 분기예측에 의하여 멀티쓰레드를 각 코어에 할당하여 실행하였다.

표 2 SPEC 2000 정수형 벤치마크 프로그램
Table 2 SPEC 2000 integer benchmark programs

벤치마크	설 명
bzip2	압축
crafty	체스 경기 놀이
gap	그룹 이론 해석기
gcc	C 프로그래밍 언어 컴파일러
gzip	압축
mcf	조합 최적화
paser	워드 프로세서
twolf	배선 및 배치 모의실험기

표 3은 모의실험에 이용된 멀티코어 프로세서 아키텍처의 사양을 나타낸 것이다. 멀티코어의 개수는 1 개, 2 개, 4 개, 8 개 및 16 개를 대상으로 하였다. 각 코어는 최대 2 개의 쓰레드를 할당받아 실행시킬 수 있으며, 각 쓰레드의 길이는 4 개, 8 개, 및 16 개의 3 가지 크기로 설정하였다. 따라서 각 코어는 매 싸이클 마다 최저 8 개부터 최대 32 개의 새로운 명령어를 인출, 이슈, 실행 및 기록한다. 각 코어의 연산유닛은 정수형 유닛, 로드 및 스토어 유닛, 그리고 분기 명령어 유닛으로 구성된다.

명령어 캐쉬와 데이터 캐쉬는 각 코어마다 설치되는데, 64 KB의 용량을 갖도록 설정하였다. 각 캐쉬는 2 차 연관도(2-way set associative) 방식을 통하여 접근된다. 각 캐쉬의 블럭 크기는 16 B로 정하였으며, 1 차 캐쉬 미스가 발생하였을 때는 10 싸이클의 페널티를 갖는다. 그러나, 주 메모리는 별도로 모델링하지 않았기 때문에, 모든 코어에 의하여 공유되는 2 차 통합 캐쉬는 충분한 용량으로 인하여 100 % 히트가 난다고 가정하였다.

퍼셉트론 멀티쓰레드 분기예측기에서 쓰레드 히스토리의 길이는 8로, 쓰레드 패턴 히스토리 테이블의 항목 수를 4096 개로 설정하였다. 퍼셉트론 학습을 위한 임계치 Θ의 값은 수식 2로 정의되었으며, 이 때 THRN은 쓰레드 히스토리 레지스터의 길이를 나타낸다. 퍼셉트론 멀티쓰레드 분기예측과의 성능의 비교를 위한 2 단계 적응형 멀티쓰레드 분기예측에서 전역 히스토리 레지스터의 길이는 14 비트, 테이블의 항목 수를 16,384 개로 하였다. 위의 퍼셉트론 멀티쓰레드 분기예측과 2 단계 적응형 멀티쓰레드 분기예측에서 쓰레드 히스토리의 길이와 패턴 히스토리 테이블의 항목 수는 공정한 비교를 위하여 동일한 하드웨어 비용을 갖도록 설정되었다.

표 3 멀티코어 프로세서 아키텍쳐 하드웨어의 사양
Table 3 The architecture specification of each core

항목	값				
멀티코어의 수	1	2	4	8	16
코어 당 쓰레드의 최대 수	2				
쓰레드의 길이 및 연산유닛	4		8		16
	산술논리(2), 로드(1), 스토어(1) 분기(2)		산술논리(4), 로드(2), 스토어(2) 분기(2)		산술논리(8), 로드(2), 스토어(2) 분기(2)
명령어 캐쉬 및 데이터 캐쉬의 용량 (KB)	64				
쓰레드 어드레스 캐쉬	2 K 엔트리				
멀티쓰레드 분기예측기	2 단계 적응형		퍼셉트론		
	14 비트 전역 히스토리, 16384 패턴 히스토리 테이블, 미스 페날티 6 싸이클		8 비트 전역 히스토리, 4096 패턴 히스토리 테이블, 미스페날티 6 싸이클		
이슈 지연 싸이클	산술논리(1), 분기(1), 로드(1), 스토어(1)				
결과 지연 싸이클	산술논리(1), 분기(1), 로드(1), 스토어(1)				

$$\theta = 2 \times THRN + 14 \quad (2)$$

4. 모의실험 및 결과

그림 6에 쓰레드의 길이가 4, 8, 16 일 때 1-코어에서 16-코어에 대하여, 2 단계 적응형 멀티쓰레드 분기예측과 퍼셉트론 멀티쓰레드 분기예측법을 각각 이용하는 멀티코어 프로세서의 모의실험 결과를 나타내어 비교하였다. 그림 6(a)와 6(b)는 쓰레드의 길이가 4일 때 성능의 결과를 각각 나타낸 것이다. 2-코어일 때 2 단계 적응형 멀티쓰레드 분기예측에 의한 평균 성능은 2.60 IPC를 나타냈으며, 퍼셉트론 멀티쓰레드 분기예측에 의한 성능은 보다 높은 2.63 IPC를 기록하였다. 16-코어일 때는 2 단계 적응형 멀티쓰레드 분기예측과 퍼셉트론 멀티쓰레드 분기예측이 각각 12.8 IPC와 13.0 IPC를 나타냈다. 퍼셉트론 멀티쓰레드 분기예측법을 이용하는 경우, 코어의 개수가 1-코어에서 2-코어로 증가했을 때 성능이 1.8 배 증가하였으며, 8-코어에서 16-코어가 되었을 때는 성능이 1.7 배로 둔화되었다. 평균적으로, 모든 코어의 수에 대하여 쓰레드의 길이가 4일 때, 퍼셉트론 멀티쓰레드 분기예측 방법이 2 단계 적응형 분기예측을 이용하는 경우보다 평균 1.1 % 우세한 성능을 가져왔다.

그림 6(c)와 6(d)는 쓰레드의 길이가 8일 때 같은 조건으로 비교한 것이다. 2-코어일 때 2 단계 적응형 멀티쓰레드 분기예측에 의한 평균 성능은 4.2 IPC를 나타냈으며, 퍼셉트론 멀티쓰레드 분기예측의 경우 4.3 IPC를 기록하였다. 16-코어일 때는 2 단계 적응형 멀티쓰레드 분기예측과 퍼셉트론 분기예측이 각각 18.5 IPC와 18.9 IPC를 나타냈다. 퍼셉트론을 이용하는 경우, 코어의 개수 2 배가 될수록 성능은 평균 1.7 배 증가하였다. 또한, 쓰레드의 길이가 8일 때의 성능은 쓰레드의 길이가 4일 때보다 평균 1.6 배 증가하였다. 쓰레드의 길이가 8일 때 모든 코어의 수에 대하여, 2 단계 적응형을 이용하는 경우보다 퍼셉트론 멀티쓰레드 분기예측이 평균 2.6 % 우세한 성능을 가져왔다.

마지막으로 그림 6(e)와 6(f)는 쓰레드의 길이가 16일 때, 2 단계 적응형 멀티쓰레드 분기예측과 퍼셉트론 멀티쓰레드 분기예측을 이용하는 멀티코어 프로세서의 각 모의실험 결과이다. 2-코어일 때 2 단계 적응형 멀티쓰레드 분기예측에 의한 평균 성능은 6.5 IPC를 나타냈으며, 퍼셉트론 멀티쓰레드 분기예측은 보다 높은 6.9 IPC를 기록하였다. 16-코어일 때는 2 단계 적응형 멀티쓰레드 분기예측과 퍼셉트론 멀티쓰레드 분기예측이 각각 24.7 IPC와 25.2 IPC를 나타냈다.

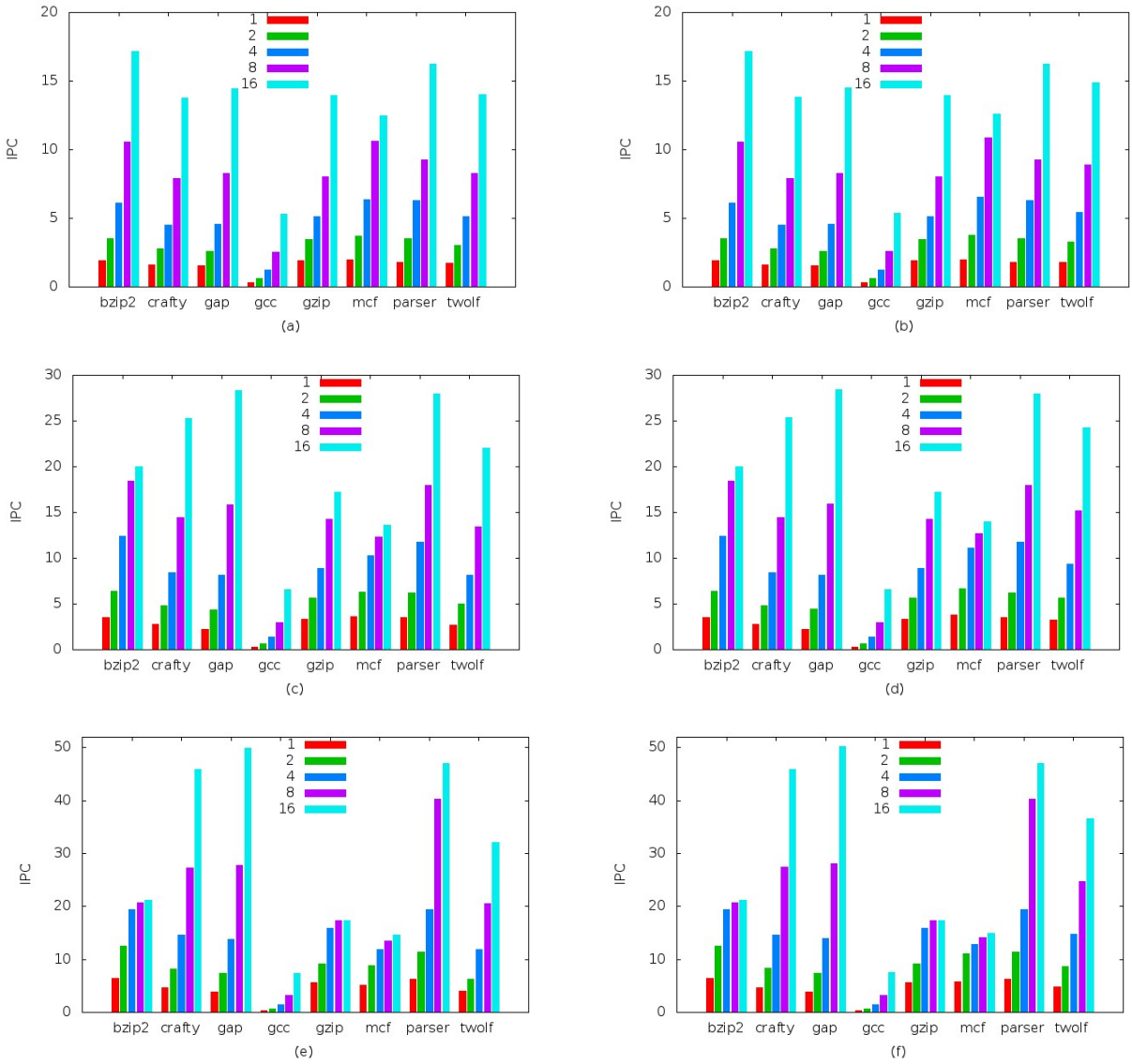

그림 6 2단계 적응형 멀티쓰레드 분기예측과 퍼셉트론 멀티쓰레드 분기예측을 이용하는 멀티코어 프로세서의 성능 비교

Fig. 6 The performance comparison of multi-core processors employing the two-level adaptive multi-thread branch predictor and the perceptron multi-thread branch predictor

퍼셉트론 멀티쓰레드 분기예측의 경우, 코어의 개수가 1-코어에서 2-코어로 증가했을 때 성능이 1.8 배 증가하였으며, 8-코어에서 16-코어가 되었을 때는 성능이 1.4 배를 기록하였다. 쓰레드 길이에 대한 결과를 종합하면, 쓰레드의 길이가 16일 때의 성능은 쓰레드의 길이가 8일 때보다 1.5 배, 4일 때보다 2.4 배 개선되었다. 쓰레드의 길이가 16일 때 모든 코어의 수에 대하여, 퍼셉트론 멀티쓰레드 분기예측이 2단계 적응형 멀티쓰레드 분기예측을 이용하는 경우보다 평균 4.5 % 우세한 성능을 가져왔다.

위의 모의실험결과를 종합하면, 제안하는 퍼셉트론 멀티쓰레드 분기예측법을 멀티코어 프로세서에 적용하였을 때, 2단계 적응형 멀티쓰레드 분기예측법보다 훨씬 높은 정확도로 인하여 성능이 향상되었다. 또한, 멀티코어 프로세서의 성능이 쓰레드의 길이와 코어의 개수가 증가할수록 더욱 개선되는 것을 알 수 있다.

5. 결 론

본 논문에서는 멀티코어 프로세서에 대하여 퍼셉트론 멀티쓰레드 분기예측을 적용하였고, 2 개부터 16 개까지의 멀티코어 프로세서 아키텍처에 대하여 SPEC 벤치마크를 입력으로 하여 모의실험을 통하여 그 성능을 측정하고 분석하였다. 그 결과, 쓰레드의 길이가 4, 8, 16일 때 퍼셉트론 멀티쓰레드 분기예측법은 기존의 2 단계 적응형 멀티쓰레드 분기예측법보다 각각 1.1 %, 2.6 %, 4.5 %의 높은 성능을 나타냈다. 이것은 퍼셉트론의 학습에 의하여, 멀티쓰레드 분기예측의 정확도가 향상되어 멀티코어 프로세서의 성능에 그대로 반영되었기 때문이다.

추후로, 동질 코어(homogeneous core)가 아닌 비동질 코어(heterogeneous core)를 채택하는 비대칭 칩 멀티프로세서(asymmetric chip multiprocessor) 구조에 퍼셉트론 멀티쓰레드 분기예측법을 적용한 성능에 대한 연구가 필요하다. 병행하여, 멀티코어 프로세서 아키텍처에서의 전력 소모량을 계산하여, 성능과 전력을 트레이드 오프할 수 있는 최적화된 하드웨어 사양에 대한 연구를 수행할 예정이다.

감사의 글

본 연구는 한성대학교 교내연구장려금 지원과제 임.

References

[1] R. Kumar, D, M, Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas, “Single-ISA Hetergogeneous Multi-Core Architectures for Multithreaded Workload Performance,” Proceedings of the 31st International Symposium on Computer Architecture, Jun 2004.

[2] S. W. Keckler, K. Olukotun, and H. P. Hofsee, “Multicore Processors and Systems,” Springer. 2009.

[3] Jongbok Lee, “A Performance Study of Multicore Out-of-order superscalar processor architectures,” KIEE, Vol. 61, No. 10, Oct. 2012, pp. 1502-1507.

[4] T. Ungerer, B. Robic, and J. Silk, “Multithreaded Processors,” The Computer Journal, Vol. 45, No. 3, 2002

[5] D. Ortiz-Arroyo and B. Lee. “Dynamic Simultaneous Multithreaded Architecture,” International Conferences on Parallel and Distributed Computing Systems. Aug. 2003.

[6] J. Gummaraju and M. Franklin. “Branch Prediction in Multi-Threaded Processors,” Parallel Architectures and Compilation Techniques, pp.179-188, Oct. 2000.

[7] D. A. Jimenez and C. Lin,“Dynamic Branch Prediction with Perceptrons,” High Performance Computer Architecture, Jun 2001, pp. 197-206

[8] Jongbok Lee, “Multiple Branch Prediction Using Perceptrons,” KIEE, Vol. 58, No. 3, Mar. 2009, pp. 621-626.

[9] M. Frankilin, G. S. Sohi, “ARB: A Hardware Mechanism for Dynamic Reordering of Memory References,” IEEE Transactions on Computers, Vol. 45, No. 5, May 1996.

[10] T. Austin, E. Larson, and D. Ernest, “SimpleScalar : An Infrastructure for Computer System Modeling,” Computer, vol. 35, no. 2, pp. 59-67, Feb. 2002.

저 자 소 개

이 종 복(Jongbok Lee)

1964년 8월 20일생.
1988년 서울대 컴퓨터공학과 졸업.
1998년 동 대학 전기공학부 졸업(공박).
1998~2000 LG반도체 선임연구원.
2000년~현재 한성대 정보통신공학과 교수
Tel : 02-760-4497
Fax : 02-760-4435
E-mail : jblee@hansung.ac.kr