논문 2005-42CI-1-2

신경망의 분석을 통한 방향 정보를 내포하는 분기 예측 기법

(Direction-Embedded Branch Prediction based on the Analysis of Neural Network)

곽 종 욱*, 김 주 환*, 전 주 식*

(Jong Wook Kwak, Ju-Hwan Kim, and Chu Shik Jhon)

요 약

파이프라인과 슈퍼스칼라 방식 그리고 동적 스케쥴링 기법이 일반화된 시스템 구조 하에서, 분기 명령어에 대한 분기 예측 정확도는 프로세서 입장에서 뿐만 아니라 시스템 전체적인 성능에 있어서 큰 영향을 미친다. 이는 분기 예측이 실패했을 경우 잘못된 분기 예측으로 인한 페널티가 발생하기 때문이며, 이러한 패널티는 파이프라인의 깊이가 깊어지고 더욱 많은 수의 명령어가 동시에 실행되는 환경일수록 더 큰 값을 가진다. 본 논문에서는 분기 예측의 정확도를 높이기 위해서, 분기 예측과 관련된 신경망을 구축하여 이를 통해 분기 예측에 필요한 각 요소별 가중치의 경향을 분석한다. 그 결과, 높은 가중치를 가지는 구성 요소를 기존의 분기 예측 기법에 추가시킨 새로운 형태의 분기 예측 기법을 제안한다. 제안된 새로운 기법은 실행 구동 방식의 시뮬레이터인 SimpleScalar를 통하여 모의실험 되었으며, 실험 결과 본 논문에서 제시한 "분기 명령어의 방향 정보를 내포하는 새로운 기법(direction-gshare)"이 기존의 gshare 기법과 비교하여 동일한 하드웨어 복잡도를 가지면서도 일반적인 Bimodal 기법이나 이단계 적응형 분기 예측 기법 혹은 그의 변형인 gshare 기법에 비하여 분기 예측의 정확도가 최대 4.1%, 평균 1.5% 더 우수한 결과를 보였으며, 최적의 방향 정보 내포량에 대해서는 최대 11.8%, 평균 3.7%의 성능 향상을 보였다.

Abstract

In the pursuit of ever higher levels of performance, recent computer systems have made use of deep pipeline, dynamic scheduling and multi-issue superscalar processor technologies. In this situations, branch prediction schemes are an essential part of modern microarchitectures because the penalty for a branch misprediction increases as pipelines deepen and the number of instructions issued per cycle increases. In this paper, we propose a novel branch prediction scheme, direction-gshare(d-gshare), to improve the prediction accuracy. At first, we model a neural network with the components that possibly affect the branch prediction accuracy, and analyze the variation of their weights based on the neural network information. Then, we newly add the component that has a high weight value to an original gshare scheme. We simulate our branch prediction scheme using SimpleScalar, a powerful event-driven simulator, and analyze the simulation results. Our results show that, compared to bimodal, two-level adaptive and gshare predictor, direction-gshare predictor(d-gshare.3) outperforms, without additional hardware costs, by up to 4.1% and 1.5% in average for the default amount of embedded direction, and 11.8% in maximum and 3.7% in average for the optimal one.

Keywords: 분기 예측, 분기 예측 정확도, 신경망, 퍼셉트론, direction-gshare

I. 서 론

컴퓨터 구조적인 측면에서의 성능 향상을 추구하기 위한 연구와 노력은 지금까지도 활발히 진행되고 있다. 특히 고성능 컴퓨터 시스템에 있어서 명령어 단위의 병렬성(ILP, Instruction Level Parallelism)에 대한 추구는 시스템 전체적인 성능 향상에 필수 조건으로 부각되었다. 이와 같은 ILP를 향상시키기 위한 노력으로, 더욱 세분화된 파이프라인 (Pipeline)의 사용과 아울러 다양한 형태의 슈퍼스칼라 (Superscalar) 방식, 그리고 이를 지원하는 여러 형태의 동적인 스케쥴링 (Dynamic Scheduling) 정책들이 제안되었다. 또한 프로세서의 입

정회원, 서울대학교 전기·컴퓨터 공학부
(Department of Electrical Engineering and Computer Science, Seoul National University)
접수일자: 2004년9월8일, 수정완료일: 2005년1월11일

장에서 뿐만 아니라 메모리적 측면에서도, 시스템 전체적인 IPC(Instruction Per Clock)를 향상시키기 위하여 계층적 메모리 시스템(Hierarchical Memory System)의 사용과 아울러 희생 캐쉬(Victim Cache)나 비중단 캐쉬(Nonblocking Cache)등과 같은 다양한 메모리 기법들이 활용되고 있다. 이러한 프로세서와 메모리적 측면에서의 발전 이외에도 시스템 상에서 수행되는 작업(Workload)들 역시 기존과 비교하여 더욱 세분화되고 다양화되어가고 있으며, 이들의 시스템 계산 능력에 대한 요구치도 점차 증가하는 추세이다. 또한 이를 지원하는 컴파일러나 운영체제 등을 비추어 볼 때, 향상된 시스템 상에서의 응용 프로그램의 수행 패턴 역시 점차 최적화 되어가는 방향으로 변화되어 가리라 예상할 수 있다[1].

이와 같은 최적화된 메모리 구조, 컴파일러, 그리고 운영체제와 같은 다양한 형태의 시스템 지원 요소와 아울러 프로세서의 입장에서는 보다 더 정확한 분기 예측 기법(Branch Prediction Scheme)의 사용이 시스템 전체의 성능에 중요한 영향을 미치게 되었다. 특히 앞서 언급한 바와 같이, 더욱 세분화되어 가는 파이프라인의 사용과 다수개의 명령어들을 동시에 실행되는 슈퍼스칼라 방식, 그리고 Tomasulo 기법과 같은 동적인 스케쥴링 기법들이 사용되는 환경에서는, 일반적으로 프로그램 상에서 분기 명령어가 나타날 때 이에 대한 Taken /NotTaken(이하 T/NT) 여부가 결정될 때까지 명령어의 수행을 중단(Stall)시키지 않고, 예상되는 분기 경로로의 수행(Speculative Execution)을 계속하며 그에 해당하는 다음 명령어를 인출(Fetch)해 오는 방식을 취한다. 이러한 수행 환경에서는 분기 예측 실패로 인한 참조 실패 페널티(Penalty)도 더욱 증가하기 때문에, 정확한 분기 예측 기법이야 말로 프로세서 측면에서의 성능 향상에 가장 중요한 요소 중 하나라 할 수 있다[1].

또한 점차 세분화되고 다양화되어 가는 응용 프로그램의 추세에 따라 최적화되어 가는 프로세서와 메모리 시스템 구조 하에서, 응용 프로그램들의 특징도 점진적으로 바뀌어 가리라 예상 할 수 있다. 그리고 이처럼 응용 프로그램들의 특징이 바뀌어 감에 따라 응용 프로그램 상에서 나타나는 각 분기 명령어 (Branch Instruc-tion)들의 T/NT 여부에 영향을 미치는 구성 요소들도 해당 요소 자체가 바뀌거나 영향을 미치는 정도, 즉 해당 요소들의 가중치의 경향이 변화 되리라 예상할 수 있다.

한편, 인공 지능(Artificial Intelligence)의 한 분야인 신경망(Neural Network)은 일반적으로 결과 예측 능력(Prediction Capability)에 있어서 좋은 정확도를 제시하는 것으로 알려져 있다[2]. 특히 수행 결과의 예측이 각각의 요소가 어느 부분에 속하는가를 결정짓는 분류(Classification) 기법에 있어서는 신경망의 구성을 통한 분석이 단순한 카운터의 임계값 분석을 통한 결정보다는 일반적으로 더 우수한 결과를 가진다[3]. 하지만 이와 같은 신경망은 의사 결정에 있어서 많은 계산 능력과 긴 수행 시간을 요구한다는 단점이 있다. 이러한 어려움은 신경망의 기본 구성 요소인 뉴론(Neuron)의 구현과 관련된 복잡도에서 기인한다. 하지만 Rosenblatt[4] 등에 의해서 제시된 단순한 형태의 뉴론인 퍼셉트론(Perceptron)은 구현과 학습이 용이하며 계산 속도도 상대적으로 더 빠르다고 할 수 있다. 따라서 본 논문에서는 분기 예측에 영향을 미칠 수 있는 각각의 요소를 이와 같은 퍼셉트론을 이용하여 모델링하고 이를 통해 신경망을 구성하고자 한다. 그리고 분기 예측의 정확도를 높이기 위해서 신경망의 구성 설정을 통해 분기 예측에 영향을 미칠 수 있는 각 요소들의 가중치를 분석한다. 이를 바탕으로 모의실험 결과에서 나타난 높은 가중치의 요소에 해당하는 정보를 기존의 분기 예측 기법에 추가하는 방식으로 새롭게 구성된 분기 예측 기법을 제안한다.

이하 본 논문의 구성은 다음과 같다. Ⅱ장에서는 연구 배경 및 관련 연구로서, 본 논문에서 사용하는 신경망과 관련된 배경 지식을 설명하고, 그리고 기존의 시스템에 주로 사용되었던 대표적인 분기 예측 기법에 대한 관련 연구들을 소개 한다. Ⅲ장에서는 본 논문에서 사용되었던 신경망의 구성 설정과 이에 대한 실험 결과를 분석하며, 이를 바탕으로 신경망 정보의 분석을 통한 새로운 형태의 분기 예측 기법을 소개한다. Ⅳ장에서는 Ⅲ장에서 제시된 새로운 분기 예측 기법을 기존의 방식과 비교하여 이에 대한 성능을 평가하고 그 결과를 비교 분석한다. 끝으로 Ⅴ장에서는 본 논문의 결론 및 향후 연구 과제를 제시하고 본 논문을 끝맺는다.

Ⅱ. 연구 배경 및 관련 연구

이 절에서는 본 논문과 관련된 연구 배경 및 관련 연구로서, 신경망에 대한 배경 지식을 설명하며 기존의 신경망을 이용한 분기 예측 기법과 아울러 신경망을 통한 다양한 컴퓨터 구조상의 응용 가능성들을 소개한다. 그리고 기존의 시스템에서 일반적으로 사용되었던 다양

한 형태의 분기 예측 기법에 대한 소개를 한다.

1. 신경망

인공 지능의 한 분야로서의 신경망은 다양한 응용 분야에 이용되어 왔으며, 그 중에서도 특히 패턴 인식(Pattern Recognition), 분류(Classification)[5], 이미지 이해(Image Understanding)[6] 등은 그 대표적인 활용 분야라 할 수 있다. 본 논문의 연구 주제인 분기 예측 기법 역시 각각의 분기들을 T 분기와 NT 분기로 이를 이분화 하는 분류 기법(Classification)에 기반을 두는 신경망의 응용 예라 할 수 있다. 이와 같은 신경망은 세부적으로는 기계 학습(Machine Learning)의 한 분야이다. 신경망의 초기 개념은 1962년 두뇌의 기능을 연구하기 위해 사용된 생물학적 신경망(Biological Neural Network)을 통해 최초로 제안되었으며[4], 오늘날에는 많은 계산량이 요구되는 다양한 예측(Prediction) 기법이나 회귀(Regression) 기법에 적용되고 있다. 이러한 신경망의 기본 개념은 많은 수의 처리 노드(Processing Node), 즉 뉴론(Neuron)이라 불리는 각각의 요소들의 연결을 통해 구현된다. 그리고 연결된 뉴론을 통해 학습의 과정을 거치며 그 결과 해당 뉴론의 값을 강화 혹은 약화하는 방식으로 최종 결과를 수정 반영하게 된다. 하지만 이러한 뉴론은 구현 비용과 계산 요구량이 크다는 단점이 있다. 그래서 오늘날에는 이러한 뉴론의 복잡성을 단순화 시킨 형태인 퍼셉트론이 주로 사용되고 있다. 이러한 퍼셉트론은 뉴론에 비해 개념적으로 이해하기가 쉽고, 구현 및 조정이 용이하며, 학습 속도도 더 빠르다는 장점 때문에 오늘날 다양한 응용에 이용되고 있으며, 현재 다양한 형태의 퍼셉트론들이 존재하고 있다[7].

그림 1에 기본적인 퍼셉트론 모델이 도시되어 있다. 일반적으로 퍼셉트론은 계산의 목표가 되는 Boolean Function의 결과를 n개의 입력과 그에 대한 각각의 가중치의 변화를 통해 학습의 과정을 거쳐 특정 결과를 출력하게 된다. 그림 1에서 각각의 입력값 x_1, …, x_n은 해당되는 연결선(Edge)을 통해 각각 w_1, …, w_n의 가중치를 가지며, 이를 y의 입력으로 전파(Propagation)하게 된다. 이를 통한 퍼셉트론 y의 출력값은 일반적으로 식 1과 같이 계산된다.

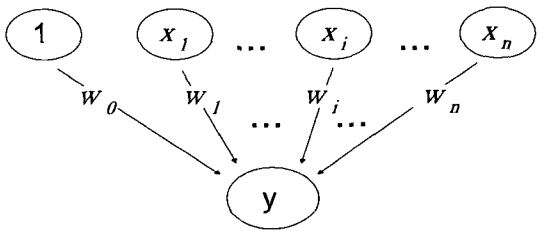

그림 1. 퍼셉트론 모델
Fig. 1. Perceptron Model.

$$y = w_0 + \sum_{i=1}^{n} x_i w_i \tag{1}$$

기본적으로 신경망은 지금까지 주로 여러 인공 지능 분야에서 사용되어 왔다. 하지만 최근 들어 신경망의 이와 같은 특성을 이용한 컴퓨터 구조상의 신경망 적용 연구가 많이 시도되고 있다. 그 중 특정 로드(Load) 연산의 값 예측(Value Prediction), 프로그램 수행시 나타나는 트레이스 예측(Trace Prediction), 그 외에 프로그램의 메모리 접근 패턴을 분석하여 사용되지 않을 캐쉬 블록의 예측을 통한 캐쉬 교체 정책 등의 적용을 대표적인 예로 들 수 있다.

$$w_0 + \sum_{i=1}^{n} x_i w_i = 0 \tag{2}$$

하지만 이와 같은 퍼셉트론의 사용은 모든 응용에 다 적용 가능한 것이 아니라 학습의 결과가 선형 구분성(Linearly Separability)이 있는 함수에만 적용 가능하다는 단점이 있다[5]. 즉 퍼셉트론이 가질 수 있는 가능한 모든 입력값에 대해서 해당 입력값을 경계화 할 수 있는 어떤 초월 평면(Hyperplane)이 존재 하여야 한다는 것이다. 이를 일반적으로 나타내면 식 2와 같이 설명될 수 있으며, 예를 들어 $n = 2$일 경우의 초월 평면은 입력 퍼셉트론을 True 집합과 False 집합으로 구분 지을 수 있는 선(Line)이 된다. 가령 이와 같은 선이 초월 평면이 될 경우, 퍼셉트론이 and 연산을 수행하는 함수는 학습 할 수 있지만, xor 연산을 수행하는 함수는 선형 구분성에 의해 학습 할 수가 없다는 뜻이 된다.

2. 분기 예측 기법

오늘날의 컴퓨터 시스템에서 주로 사용되는 분기 예측 방식은 크게 Bimodal 기법과 이단계 적응형 분기 예측 기법 (Two-Level Adaptive Training Branch Prediction) 및 그의 변형으로 구분 지어 볼 수 있다. 분기 예측 기법에 있어서의 이와 같은 두 가지 큰 분류는, 분기 명령어가 Bimodal 하다는 것과 Corelate 하다는 기본적인 두가지 특성에서 기인한다. 분기 명령어가

Bimodal 하다는 것은, T 분기는 주로 Taken되는 경향이 강하고 NT 분기는 주로 Untaken되는 경향이 강하다는 뜻으로, 분기 명령어가 그 패턴에 따라서 쉽게 통계적으로 이분화 될 수 있다는 것을 뜻한다. 한편 분기 명령어가 Corelate 하다는 것은 특정 분기 명령어의 실행 결과가 해당 분기 명령어 자체의 패턴에 따라서 결정될 수도 있지만, 이와 함께 이전 n개의 분기 명령어와의 상호 관계 속에서 그 특징이 달라 질 수 있다는 것을 뜻한다. 특히 이와 같은 특성은 오늘날의 동적 분기 예측 기법(Dynamic Branch Prediction Scheme)의 기본적인 개념으로 사용되고 있다.

우선 Bimodal 기법이라 불리는 기존의 방식은 분기 예측을 위해서 별도의 분기 예측 테이블(Branch Predic-tor Table)을 사용한다. 이때 해당 테이블의 각 요소들은 해당 분기 명령어의 주소와 함께 일반적으로 n bit 포화 카운터(Saturation Counter)를 가지고 있으며, 해당 분기 명령어의 주소값을 이용하여 분기 예측 테이블을 접근한 후, 해당 주소값의 카운터 값을 이용하여 임계값과 비교한 뒤 분기 예측을 수행한다. 한편 Yeh and Patt의 이단계 적응형 분기 예측 기법[8]은 과거 n개의 분기 명령어의 이전 기록(History)을 Branch History Register(HR)에 쉬프트 레지스터 형태로 보관하면서, 분기 명령어의 예측시마다 이 값을 인덱스로 사용하여 패턴 테이블(Pattern Table)에 접근한다. 이때 패턴 테이블은 Bimodal 기법과 유사 형태의 카운터로 구성되어 있으며, 카운터의 결과값에 따라 분기 예측을 수행한다. 결국 이 두가지 방법의 차이는 분기 예측 테이블 혹은 패턴 테이블을 인덱싱 하는 방법의 차이라 할 수 있다. 일반적으로 이단계 적응형 분기 예측 기법은 분기 명령어의 Corelate한 성질을 반영하여 성능상의 우수한 결과를 보인다. 하지만 이 방법은 분기 명령어의 가명 현상(Aliasing)을 초래하는 단점이 있다[9]. 오늘날 이와 같은 가명 문제를 해결하기 위한 많은 연구가 진행되어 왔으며, 그 중 대표적인 기법들이 Agree Predictor[10]나 Bi-Mode Predictor[11]이다. 이와 함께 1993년 Western Research Laboratory에서 제안한 기법인 gshare 기법[12]도 이들 방식 중 하나이다. 이는 기본적으로 이단계 적응형 분기 예측 기법의 변형이라 할 수 있는데, 분기 예측에 있어서 분기 명령어의 Global Branch History 값과 함께 분기 명령어의 PC 값을 함께 xor 하는 방식으로 분기 예측 테이블을 인덱싱한다. gshare 기법은 이러한 방법으로 기존에 발생할 수 있었던 가명 문제를 해결하고자 하였다.

한편, 신경망을 통한 분기 예측의 성능 향상과 관련된 기존의 연구는 주로 정적 분기 예측(Static Branch Prediction Scheme)에 국한되는 경향이 있었다[13]. 즉, 분기 명령어의 수행 경로를 컴파일러가 미리 분석하여 그 결과를 예측하는 방식이었다. 이와 같은 정적 기법은 분기 예측의 정확도가 80% 정도로 동적 기법에 비해 다소 부족한 점이 있지만, 동적 분기 기법에서 사용될 부가적인 정보를 컴파일 시간에 제공해 줄 수 있어 나름대로 의미가 있다고 하겠다. 최근 들어서는 신경망을 통한 다양한 형태의 동적 분기 예측 기법들이 소개되고 있다[14][15]. 이 또한 대부분 Yeh and Patt의 이단계 적응형 분기 예측 기법의 변형을 시도한 것들이라 할 수 있다. 본 논문에서 제시하고자 하는 신경망을 이용한 분기 예측의 개선 기법 역시 이단계 적응형 분기 예측 기법의 변형 가운데 하나로서, 이와 같은 변형에 대한 힌트로서 신경망의 도출 결과를 이용한 방식이라 할 수 있다.

Ⅲ. 신경망의 정보를 이용한 분기 예측 기법

이 절에서는 분기 예측에 있어서 이에 영향을 미치는 다양한 요소들에 대한 가중치의 정도를 파악하기 위한 분석을 수행하고 그 결과를 논의하며, 이를 반영한 새로운 형태의 분기 예측 기법을 제안한다. 우선 Ⅲ장의 1절에서는 제시될 신경망의 형태를 모델링 한 후, 이를 모의실험하여 각각의 구성 요소들에 대한 가중치의 변화를 분석한다. 2절에서는 실험 결과 신경망의 학습에서 나타나는 입력 샘플의 특성에 대해서 논의한다. 끝으로 3절에서는 1절에서 제시된 높은 영향력을 미치는 구성 요소를 기존의 분기 예측 기법에 추가한 새로운 형태의 분기 예측 기법을 제안한다.

1. 신경망이 구성 및 모의실험

본 논문에서 사용되는 퍼셉트론을 활용한 신경망의 구성은 다음과 같다. 신경망의 입력에 사용되는 입력 노드는 총 30개이다. 이는 각각 가장 최근의 Global Branch History 10개, Branch Direction History 10개, 그리고 분기 명령어의 상위 PC값 10bit의 각각에 해당하는 bit 값을 가지는 10개로서, 이와 같이 총 30개의 입력 노드를 구성한다. 여기서 기존의 Global Branch History와 분기 명령어의 PC 값 이외에 추가로 Branch Direction History를 사용한 것은, 이 또한 분기의 T/NT에 영향을 미치는지 여부를 조사하기 위해 본 논

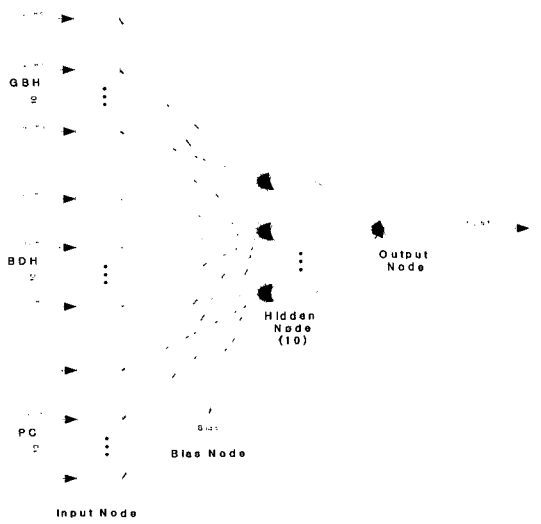

그림 2. 신경망의 구성 설정
Fig. 2. The Structure of Neural Network.

문에서 추가적으로 사용한 요소로써, 각각의 분기 명령어마다 1bit씩을 할당하여, 전방 분기(Forward Branch)이면 1, 후방 분기(Backward Branch)이면 0을 기록한다. 또한 입력 노드 30개와는 별도로, 한단계의 추가적인 내부(Hidden) 노드를 10개 사용하였으며, 끝으로 출력 노드 1개와 초기 Bias용 노드 1개를 사용한다. 이에 대한 내용이 그림 2에 나타나 있다. 그림 2에서와 같이, 입력 노드와 내부 노드, 그리고 내부 노드와 출력 노드 상호간에는 완전 연결 방식으로 구성되어 있다. 그리고 Bias용 노드는 내부 노드와 출력 노드에 각각 연결되어 있는 형태이다. 따라서 본 신경망의 구성에 사용된 노드의 수는 총 42개이며, 사용된 링크수는 321개이다.

이와 같이 구성된 신경망 하에서, 신경망의 학습을 위해 사용되는 전파 (Propagation) 식은 일반적인 Sigmoid Function을 사용하였으며, 후방 전파(Back Propagation)를 위해서는, 각 노드의 delta 값을 계산하여, 이를 통해 가중치의 변화를 조정하는 방식을 사용하였다. 우선 전파를 위해 사용되는 출력 함수로서의 Sigmoid Function은 식 3과 같다.

$$output = \frac{1.0}{1.0 + E^{-x}} \tag{3}$$

위의 식 3을 각각의 노드와 해당 노드와 연결되어 있는 각각의 링크에 적용시켜, 식 4에서처럼 이전단 노드의 출력값과 이전단 노드에서 해당 노드로의 출력 링크의 가중치 값을 서로 곱한 뒤, 각각의 곱을 합하여 그 값을 Sigmoid Function에 대입한다. 이와 같은 방식으로 최종적인 해당 노드의 출력값을 계산한다. 다만 입력 노드 30개의 출력값은, 실제 분기 명령어의 Global History 혹은 Direction History 값이나 PC 값이, 전체 신경망의 입력값으로, 바로 들어가게 된다.

$$\Sigma\ \{(\text{이전단 노드의 출력값})\times(\text{이전단 노드에서 해당 노드로의 출력 링크의 가중치})\} \tag{4}$$

이와 같이 전파를 위해 사용된 각각의 노드의 출력값들은, 최종 출력 노드에서 출력값을 계산하고 난 뒤, 후방 전파를 위해서 그 값을 유지하고 있다. 후방 전파에서는 각각의 노드에서 delta를 먼저 계산하고 그 delta를 이용하여, 각각의 링크에 해당하는 가중치를 조정하는 방법을 사용한다. 우선 delta를 계산하는 방법은 식 5와 같다.

$$delta = (error)\times[(output)\times(1.0 - output)] \tag{5}$$

식 5에서의 delta 값은 최종 출력 노드로부터 그 이전단으로 후방 전파의 형식으로 계산하여 되돌아오게 된다. 식 5에서처럼 delta 값은 전파 과정에서 얻어진 출력값을 이용하여 이를 error 값과 곱하는 방식으로 구해진다.

$$error = (resultof\_TorNT)-(valueof\_outputPerceptron) \tag{6}$$

$$error = \Sigma\ \{(\text{이후단 노드의 delta})\times(\text{이후단 노드에서 해당 노드로의 출력 링크의 가중치})\} \tag{7}$$

한편, delta를 계산하는 식에서의 error를 구하는 방법은 최종 출력 노드와 내부 노드에 있어서 그 방법의 차이가 있다. 우선 최종 출력 노드의 경우 error 값은 다음과 같이 계산된다. 식 6에서 보이듯이 최종 출력 노드의 경우 error 값은, 분기 명령어의 최종적인 T/NT에 해당하는 실제 값에서 최종 출력 노드의 출력값과의 차이를 의미한다. 이때 실제 분기 명령어의 최종적인 결과값으로는 0 혹은 1의 값을 가지게 되며, 최종 출력 노드의 출력값은 0에서 1사이의 값을 가지게 된다. 한편 내부 노드의 경우 error 값은 다음과 같이 계산된다. 위의 식 7에서처럼 내부 노드의 경우는, 이후단 노드들의 delta 값에 이후단 노드에서 해당 노드로의 링크의 가중치를 곱한 후, 각각의 곱을 합한 값이 된다.

이상에서처럼 주어진 식을 통해 해당 노드의 출력값과 delta 값을 계산하였으며, 한편 각각의 링크에 해당하는 가중치 변화량은 위의 식을 바탕으로 얻어진 모든 노드의 delta를 참조하여 아래의 식 8을 이용하여 계산된다.

$$\Delta weight = (learning\_rate \times delta \times output) + (momentum \times \Delta weight') \quad (8)$$

식 8에서 보여 지듯이, 하나의 링크에 가중치의 변화량은 후방 전파 과정을 통해 다음과 같이 계산된다. 즉 이후단 노드의 delta 값과 이전단 노드의 출력값, 그리고 Learning Rate을 서로 곱하여, 이를 이전 학습에서의 가중치의 변화량과 Momentum을 곱한 값을 서로 합하여 계산하게 된다. 이때의 Learning Rate은 가중치 보정에 영향을 미치는 요소로서, 가중치 변경의 정도를 관리하는 역할을 하며, Momentum은 가중치가 향하는 방향을 변경시키는 각 단위의 내부 가중치의 경향, 즉 가중치가 변경되는 방향을 유지하려는 성향이라 할 수 있다. 본 논문에서 주어진 신경망에서는 표 1과 같은 Learning Rate와 Momentum의 값을 사용하여 학습을 수행하였다. 이때 가중치의 초기값은 -0.5에서 0.5사이의 값 중 임의로 할당하였고, 전파와 후방 전파의 경우 입력값으로 1.0 혹은 0.0 중에서 하나로 주어졌다. 본 논문에서는 Learning Rate과 Momentum을 변화시키면서 총 6회의 학습을 반복한다. 이는 동적으로 어떤 요소를 모니터하여 특정 조건을 만족시키면 학습을 중단시키는 방식이 아니라, 정적으로 정해진 횟수만큼의 학습을 반복 수행시킨다는 의미이다. 이때의 Learning Rate과 Momentum의 값은 입력 프로그램들을 수행시키면서 분기 예측의 정확도에 대한 변화를 살펴 그 학습 효율을 높여 학습 속도의 최고치를 얻을 수 있는 값으로 결정하였다.

신경망 학습에 필요한 입력 샘플의 결정은 다음과 같다. 학습용 샘플을 추출하여 신경망을 통해 학습시키는 방법을 사용하는 대신, 시뮬레이터의 분기예측 모듈에 별도의 신경망 코드를 삽입하여 응용 프로그램을 실제로 수행 시키는 과정에서, 분기 예측 모듈이 분기 명령어를 만날 때 마다 삽입된 신경망 코드를 호출하는 방식을 사용하였다. 이때, 학습 입력 샘플의 개수는 각각의 응용 프로그램마다 다르며, 이들이 끝까지 수행되는 동안 만나는 분기 명령어의 개수이다. 아래의 표 2에 입력 샘플의 개수가 나타나 있다.

표 1. 신경망 학습 인자
Table 1. Parameters of the Neural Network.

Learning Rate	0.7	0.3	0.1	0.1
Momentum	0.7	0.5	0.5	0.2
학습 횟수	1	1	1	3

이와 같이 주어진 신경망에서 각 입력 노드의 가중치 절대값의 평균을 나타낸 내용이 그림 3에서 그림 5 사이에 나타나 있다. 사용된 응용은 SPEC CINT/CFP 2000[16]에서 제공하는 응용 프로그램들 중 일부이다. 앞서 언급한 바와 같이, 각각의 그림에서 가로축의 1번부터 10번까지의 입력은 Global Branch History이며, 1번에 가까울수록 최근의 분기를 뜻한다. 그리고 11번부터 20번까지의 입력은 Branch Direction History로서, 이 역시 11번에 가까울수록 최근의 분기를 뜻한다. 끝으로 21번부터 30번까지의 입력은 분기 명령어 PC 값의 상위 비트들로써, 21번에 가까울수록 하위비트(LSB)이며, 30번에 가까울수록 상위비트(MSB)이다. 그리고 세로축은 가중치 절대값의 평균을 나타낸다. 실험에 사용된 응용 프로그램은 art, equake, gcc, gzip, mcf, mesa, twolf, vpr로 총 8개이며, 이들을 모의실험에서 나타난 유사한 패턴의 실험결과로 분류하여 다음과 같이 크게 3가지로 구분하였다.

우선 그림 3에서 나타난 경향 1의 경우는, 분기 명령어의 Branch Direction History의 가중치가 특히 높게 나타난 응용의 예이다. 그리고 그림 4에서 나타난 경향 2의 경우는, 경향 1의 경우처럼 Branch Direction History가 큰 값의 가중치를 가지지는 않지만, Global Branch History와 비교하였을 때, 상호 유사한 정도의 가중치를 가진다고 판단할 수 있는 응용의 예이다. 끝으로 그림 5에서 나타난 경향 3의 경우는, 상대적으로 Branch Direction History의 가중치 경향이 Global Branch History에 비해 미미하게나마 낮은 경우이다.

표 2. 입력 샘플의 개수
Table 2. The Number of Input Sample.

art	135,469,690
equake	149,779,636
gcc	15,371,294
gzip	62,547,186
mcf	30,034,558
mesa	263,123,590
twolf	10,240,535
vpr	1,931,534

각각의 실험 결과를 분석하자면, 우선 분기 명령어의 분기 예측에 영향을 미치는 가중치의 값들 중에서 PC 값은 그림 3에서 그림 5 사이에서 나타나듯이 상대적으로 다른 구성 요소들에 비해서 비교적 높은 값을 가진다는 것을 알 수 있다. 그리고 이러한 PC 값은 상위 혹은 하위 비트의 특별한 구분 없이 대체적으로 높은 값의 가중치를 가지며, 특정 위치별로의 독특한 패턴의 분포를 찾을 수 없다. 이는 전반적으로 모든 응용 프로그램에서 해당되는 사항이며, 분기 명령어의 분기 예측에 있어서 지역 분기 예측 기법(Local Branch Prediction Scheme)의 유효성을 보여주는 실례라 할 수 있다. 즉 지역 분기 예측 기법은 각각의 지역 분기 명령어의 정보만을 파악하여 이를 통해 분기 예측을 수행하며, 이때 사용되는 정보가 각 분기 명령어의 지역 PC 값과 그에 해당하는 지역 분기 명령어의 History인 것이다. 이러한 지역 분기 예측 기법은 McFarling[12]의 연구 결과에서도 나타나 있으며, 또한 해당 연구에서 지역 분기 예측 기법은 나름대로 우수한 성능을 보여 주었다. 본 논문에서는 이와 같이 분기 명령어와 관련된 신경망의 구성 설정을 통해서 이전 연구에 대한 유효성을 재확인 할 수 있었으며, 이에 대한 근거를 신경망을 통한 실험으로 제시했다고 할 수 있다. 하지만 일반적으로 분기 명령어는 지역적 정보와 더불어 전역적 정보를 추가적으로 사용할 때 성능상에 더 긍정적인 영향을 미칠 수 있다는 사실은 이미 널리 알려진 바이다. 다시 말해, PC 값이 상대적으로 큰 가중치의 값을 가지지만, 경우에 따라서는 Global Branch History가 상대적으로 더 높은 가중치를 가질 수 있음을 의미한다. 일반적으로 이단계 적응형 분기 예측 기법이나, gshare 기법이 이러한 두 가지 정보를 모두 이용하는 기법들이다. 이와 같은 상황에 대한 근거 역시 본 논문의 그림 3에서 그림 5에 나타난 모의실험 결과를 통해 확인 할 수 있다. 특히 그림 3의 응용 프로그램들과 더불어 그림 4의 art, mcf가 PC 값에 비해 상대적으로 최근 Global Branch History가 높은 값을 가지는 응용 프로그램의 예이다. 4장 2절의 모의실험 결과에서 다시 다루어지겠지만, 이상 언급된 응용 프로그램들은 상대적으로 이단계 적응형 분기 예측 기법이나 gshare 기법이 분기 명령어의 지역적 정보만 사용한다고 할 수 있는 Bimodal 기법에 비하여 우수한 성능을 보이는 응용의 예이다. 특히 Bimodal 기법이 소규모의 분기 예측 테이블이 사용될 경우 상대적으로 더 우수한 성능을 보이는데 비해, 이들 응용들은 작은 크기의 분기 예측 테이블 하에서도 Bimodal 기법보다 더 우수한 결과를 보였다.

이상의 실험결과를 통해 본 논문에서는, 기존의 지역 분기 예측 기법의 우수한 성능에 대한 근거와, 분기 예측에 있어서 지역적 정보와 전역적 정보를 함께 이용할 때 더 우수한 결과를 보일 수 있다는 사실에 대한 근거를 신경망이라는 기법을 활용하여 이에 대한 가중

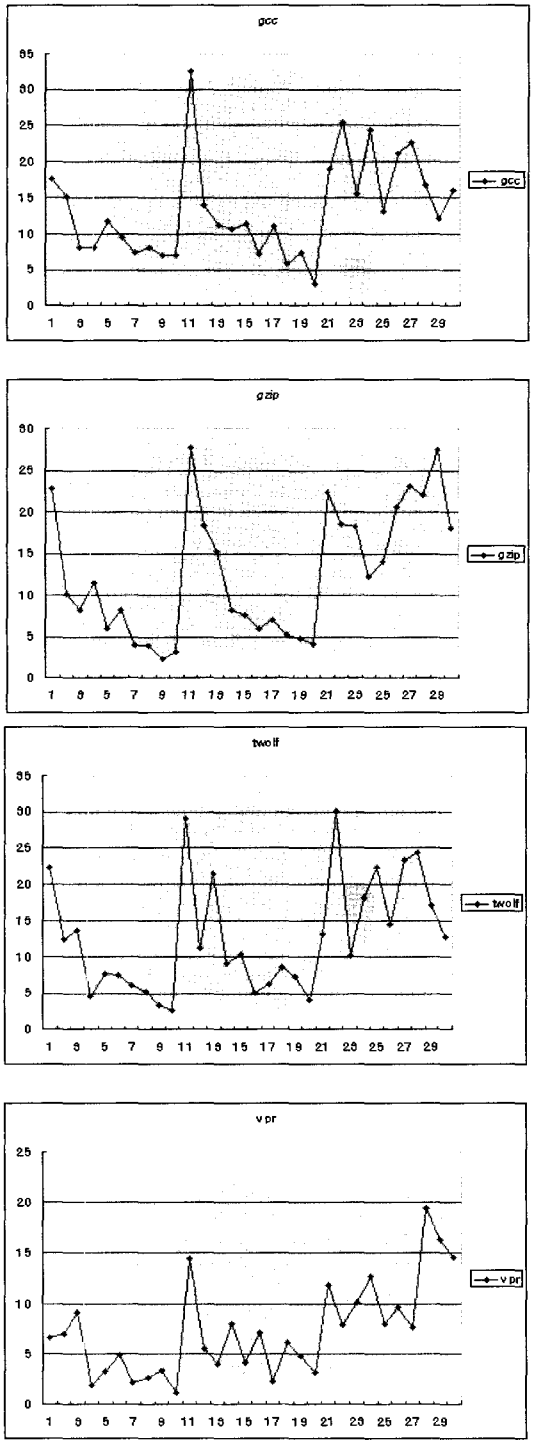

그림 3. 가중치의 상대적 차이 - 경향 1
Fig. 3. Relative Difference in Weight - Pattern 1.

치의 분석을 통해 제시했다고 할 수 있다.

한편, 그림 3에서 나타난 경향 1의 경우는 상대적으로 본 논문에서 추가적으로 사용한 Branch Direction History의 가중치 경향이 특히 높게 나타난 응용의 예들이다. gcc, gzip, twolf, vpr이 이에 해당하는 응용 프로그램들이다. 그림 3에서 보여 지듯이, gcc와 gzip의 경우는 특히 다른 여타의 구성 요소들에 비하여 Branch Direction History의 가중치가 월등히 높은 경향을 가진다. 또한 Global Branch History와 Branch Direction History는 최근의 값일수록 높은 가중치를 가지며, 최근에서 점차 멀어질수록 대체적으로 가중치가 줄어드는 경향을 가진다는 것을 알 수 있다. 특히 주목할만한 사항은, Global Branch History의 경우는 Branch Direction History와 비교하여 상대적으로 다양한 가중치의 변화 정도를 보이지만 대체로 비슷한 값을 가지며 천천히 줄어드는 반면, Branch Direction History는 최근의 값일수록 매우 높은 값을 가지며, 이 값은 최근에서 멀어질수록 빠르게 줄어드는 경향이 있다는 것을 알 수 있다. 특히 11번부터 13번까지의 3개의 노드에 해당하는 가중치의 경향이 특히 더 높다고 할 수 있다. 이상에서 그림 3의 경향 1은 그림 4와 그림 5의 각각의 경향과 비교해 보았을 때 상대적으로 PC 값의 가중치의 정도가 낮다고 할 수 있으며, Global Branch History와 Branch Direction History가 많은 영향을 미친다고 할 수 있다. 특히 최근의 정보일수록 해당 영향력은 그림 4의 경향 2와 그림 5의 경향 3에 비해 더 크다고 할 수 있는 응용 예이다.

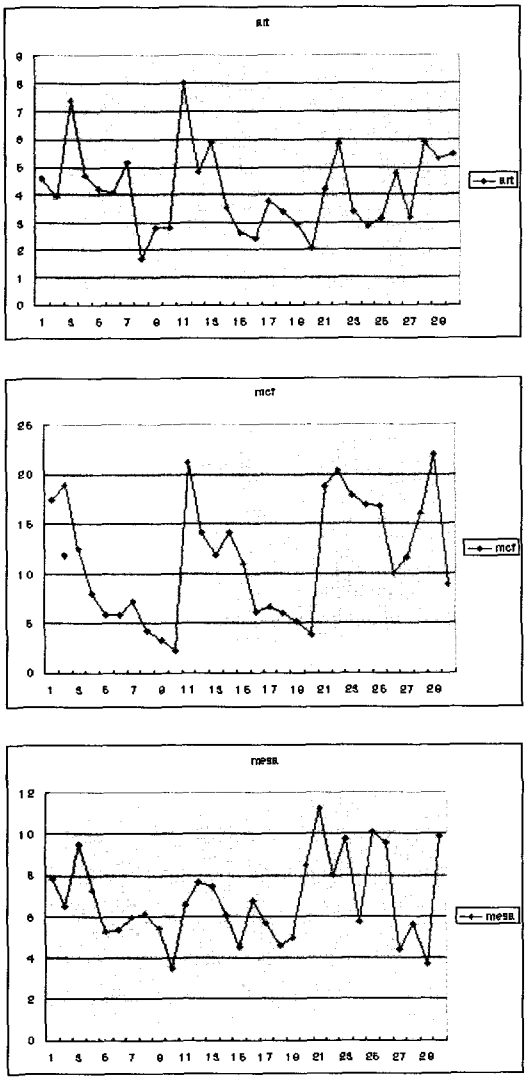

그림 4. 가중치의 상대적 차이 - 경향 2
Fig. 4. Relative Difference in Weight - Pattern 2.

그림 4에서 나타난 경향 2의 경우는, 경향 1에서처럼 Branch Direction History가 월등히 높은 가중치 값을 가지지는 않지만, Global Branch History와 비교 할 경우 상호 유사한 크기의 가중치 값을 가진다고 할 수 있는 응용의 예이다. art, mcf, mesa가 이에 해당하는 응용의 예이다. 그림 4에서 보여주듯이 이러한 응용들은 그림 3과 비교하였을 때, 특히 PC 값에 대한 가중치가 다른 두 구성요소에 비해 상대적으로 더 영향을 미친다는 것을 알 수 있으며, 또한 Branch Direction History와 Global Branch History는 서로 유사한 크기의 가중치 값을 가진다는 것을 알 수 있다. 이를 바꾸어 말하자면, 경향 2에 해당하는 응용 프로그램들의 분기 예측은, 그 예측 결과에 있어서 Branch Direction History와 Global Branch History가 상호 대등한 정도의 영향력을 가질 수 있음을 뜻한다.

끝으로 그림 5에 나타난 경향 3은, 경향 1과 경향 2와 비교하였을 경우 상대적으로 Branch Direction History의 영향력이 작은 경우에 해당하는 응용의 예이다. 실험에 사용된 응용 프로그램들 중에서 equake가 유일하게 이 범주에 해당하였다. equake는 독특하게도

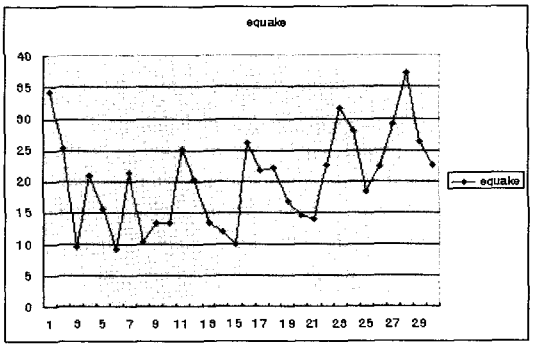

그림 5. 가중치의 상대적 차이 - 경향 3
Fig. 5. Relative Difference in Weight - Pattern 3.

11번부터 감소한 Branch Direction History의 가중치 값이 16번에 이르러 다시 증가하는 형태를 보이다 다시 가중치의 경향이 줄어드는 패턴을 보인다.

이상에서와 같이 본 논문의 신경망 구성 설정을 통한 모의실험에서, 분기 예측의 정확도에 영향을 미치는 각 요소들 간의 상대적인 가중치의 정도를 상호 비교 분석한 것이 본 논문의 의의 중 하나라 할 수 있겠다. 한편, 주어진 모의실험에서 8가지의 응용 프로그램들 중 총 7가지의 실험결과에서 Branch Direction History가 아주 강한, 혹은 적어도 Global Branch History와 대등한 정도의 가중치 값을 가진다는 것을 확인 할 수 있었다. 그리고 또한 평균적으로 11번부터 13번까지의 노드가 상대적으로 높은 가중치 값을 가진다는 것을 알 수 있었다. 이와 같은 신경망의 구성 하에서 나타난 특징을 기반으로 하여, 본 논문에서는 분기 예측에 필요한 각 구성 요소들의 영향력의 정도를 반영한 새로운 형태의 gshare 기법을 제안하고자 한다.

2. 신경망의 학습과 입력 샘플의 특성

본 논문에서 사용되는 입력값으로의 샘플들은 최종적으로 수렴하지는 않는다. 순수하게 학습용 입력 인자 세 가지만을 참고하여 예측을 할 경우 어떠한 방식을 사용하여도 100% 정확한 예측은 불가능하기 때문이다. 또한 입력 인자들의 값이 모두 같다 하더라도 출력은 전혀 상반되는 경우도 발생한다. 따라서 본 논문의 응용에 있어서 학습의 수렴은 불가능하다.

예를 들면, 20회를 반복하고 빠져나가는 loop이 있을 경우를 가정해 보자. 만약 이 loop을 15회 반복한 후와 20회 반복수행을 마치고 빠져나가는 순간을 비교해보면, 15회 반복했을 경우 PC 10bit는 해당 loop의 분기 명령어의 주소가 되며(가령, 1100101001), Global Branch History와 Branch Direction History는 각각 Taken(TT...TT)과 Backward(BB...BB)가 된다. 즉 다음과 같은 입력 형태가 된다.

TTTTTTTTTT,BBBBBBBBBB,1100101001

이와 비교하여, 20회의 수행을 마치고 loop을 빠져나가는 순간의 PC 10bit는 해당 loop 분기 명령어의 주소가 되며(가령, 1100101001). Global Branch History는 모두 Taken(TT...TT), Branch Direction History는 모두 Backward(BB...BB)가 된다. 즉 다음과 같은 입력 형태가 생성된다.

TTTTTTTTTT,BBBBBBBBBB,1100101001

이 경우, 두 가지의 입력값은 완전히 동일하지만 출력결과 값은 정반대이다. 즉, 15회 반복했을 경우는 결과값으로 T가 되어야 하며, 20회 반복했을 경우는 NT가 되어야 한다. 이처럼 입력 인자의 값들이 완전히 똑같아도, 결과값은 반대가 되는 경우가 있기 때문에 최종적인 값으로의 수렴은 불가능하다. 하지만 특정 목표값에 대한 근사치는 얻어 낼 수 있다.

한편, 본 논문에서는 분기 예측의 입력 인자로 PC, Global Branch History, Branch Direction History를 사용하였다. 하지만 이러한 인자들과 분기 명령어의 실질적인 Taken 여부와의 관계는 전술한 바와 같이 그 결과를 정확히 예측 할 수는 없다. 따라서 각각의 인자들이 분기 명령어의 Taken 여부에 미치는 영향의 정도를 신경망을 통해 분석해 본 것이 본 논문의 첫번째 의의이기도 하다. 분기 명령어의 Taken 여부의 관계를 앞서 말한 세 가지 입력인자를 통하여 정확히 알 수는 없으나 추측해 볼 수 있는 몇 가지 시나리오는 다음과 같다.

우선, PC는 응용 프로그램에서 특정 분기 명령어를 구분해내는 역할을 한다. 즉, "100번지의 분기 명령어는 주로 Taken이다."라는 식의 정보를 얻을 수 있게 한다. 하지만, 응용 프로그램별로 분기 명령어의 주소 위치는 모두 다르기 때문에 이는 전적으로 응용 프로그램 종속적이다고 할 수 있다. 그리고 이는 본 논문에서의 표기법과 같이 TTTTTNTTNN,BBBBBBFFFB,1110100010의 입력 패턴이 존재한다고 가정하면, 1110100010 주소의 분기 명령어는 loop이며 현재 5번 loop를 수행했다는 것을 추정할 수 있다. 그렇다면, 이 분기 명령어의 이전 번 수행 시 해당 주소의 분기 명령어가 몇번의 loop를 수행하고 빠져나갔는지를 알 수 있다면 T가 몇번까지 나오고 NT로 빠져 나갈 것인지를 예측할 수 있게 된다.

또 다른 예로, 다음과 같은 코드가 있다고 가정 해 보자.

```
if (x >1) goto a; //if1
code;
a:
if (x <-1) goto b; //if2
code;
b:
```

만약 이 경우, 앞쪽에 있는 branch(if1)가 Taken 된

다면 뒤쪽에 있는 branch(if2)는 무조건 Not Taken 된다. 유사한 예로 다음과 같은 코드도 있을 수 있다.

```
for (i = 0; i <= 10; i++)
{
        code;
        if (condition) break; //if1
}
if (i == 10) goto anywhere; //if2
```

이 경우 앞쪽의 loop(for문)가 끝까지 수행된다면, 마지막의 branch(if2)는 Taken이 될 것이고, 중간의 if문(if1)으로 loop를 빠져나가게 된다면 i의 값이 10이 아닌 대부분이 경우 마지막의 branch(if2)는 Not Taken이 될 것이다. 이상의 두 가지 예를 신경망의 입력 인자들로 살펴보면 다음과 같다. 첫 번째 예의 경우는 TTNTNTNTNT,FFBFBFBFBF,1010101010로서, 마지막의 분기 명령어가 Forward이고 Taken일 경우, 1010101010 주소에 있는 뒤쪽의 if문(if2)은 무조건 Not Taken이 된다고 할 수 있다. 두 번째 예의 경우는 NNTNTNTNTN,BFBFBFBFBF,1010101010 식의 패턴이 될 것이며, loop를 끝까지 수행한 경우 1010101010 주소에 있는 마지막 if문(if2)은 무조건 Taken이 된다. 반대로 TTNTNTNTNT,FBFBFBFBFB,1010101010의 패턴이 나오면 중간의 if문(if1)에서 loop를 빠져 나왔다는 것을 의미하므로, 1010101010 주소에 있는 마지막 if문(if2)은 Not Taken일 가능성이 높다고 예측할 수 있다.

3. 방향 정보를 내포하는 분기 예측 기법

이 절에서는 3장의 1절에서 제시된 신경망의 도출 정보를 근거로 하여 새로운 형태의 "분기 방향 정보를 내포하는 분기 예측 기법(direction-gshare)"을 제안한다. 새롭게 제시된 기법은 gshare 기법의 변형이라 할 수 있다. 기존의 gshare 기법은 Global Branch History와 분기 명령어의 PC 정보만을 사용하여, 이를 xor한 결과를 이용해 분기 예측 테이블을 인덱싱 한다. 이에 대한 내용이 그림 6에 나타나 있다. 그림 6에서는 Global Branch History로서 m비트를 사용한다. 즉, 과거 m개의 분기 명령어의 Global Branch History를 이용한다는 뜻이 된다. 또한 분기 명령어의 PC 값으로 n비트를 사용한다. 이때의 n비트 값은 주로 해당 PC 값의 상위(MSB) 부분에서 가져오게 된다. 하위(LSB) 부분이 아니라 상위 부분에서 PC 값 비트를 추출하는 이유는, 상대적으로 하위 부분에 비해 상위 부분의 각 비트 별 분포가 상대적으로 더 고르게 분포(Uniform Distribution)되기 때문이다. 또한 이러한 상황은 분기 예측 테이블의 인덱싱에 있어서 가명 문제를 보다 더 효과적으로 감소시키는 역할을 한다.

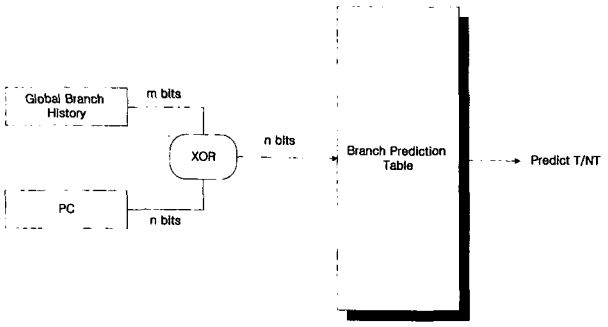

그림 6. gshare 기법
Fig. 6. gshare Predictor.

이와 같이 그림 6에서 보이는 기본적인 gshare 기법을 바탕으로 하여 제안된 새로운 기법은, 기존의 gshare 기법에서 Global Branch History가 차지하던 부분을 Global Branch History와 Branch Direction History를 조합하여 사용하는 방식으로 변형한 형태이다. 그리고 이를 분기 명령어의 PC 값과 xor를 수행하여 최종적으로 분기 예측 테이블을 인덱싱하게 된다. 이때의 PC 값은 기존의 ghsare 기법에서와 동일한 방식으로 추출된다. 이에 대한 내용이 그림 7에 나타나 있다. 본 논문에서는 Branch Direction History로서 기본적으로 3 비트를 이용한다. 이는 3장의 1절에서 제시된 가중치 경향의 분석 결과로부터 내린 결론이다. Branch Direction History로서의 3 비트는, 분기 명령어의 PC 값을 기준으로 해당 분기가 전방 분기이면 1을, 후방 분기이면 0의 값을 할당하게 된다. 이러한 Branch Direction History의 추가적인 사용으로 얻을 수 있는 성능상의 이득은 4장에서 자세히 다룰 것이다. 또한 그림 7에서 보여 지듯이, 제안된 새로운 기법은 기존의 Global Branch History의 일부를 다른 값으로 변형하여 사용하는 방식으로, gshare 기법과 비교하여 추가적인 하드웨어 요구가 없이 동일한 복잡도를 가진다. 그리고 제안된 새로운 기법은 기존의 Global Branch History의 값을 Global Branch History와 Branch Direction History 값으로 양분하여 사용함으로, 분기 예측의 구성 요소 중 하나로써 Global Branch History를 사용하는 기존의 여러 분기 예측기법에 적용가능하다. 더 나아가, 최근 들어 제시되고 있는 복합 예측 기법(Hybrid Prediction Scheme)에서도 기본 구성요소 중 하나로써

그림 7. direction-gshare 기법
Fig. 7. direction-gshare Predictor.

추가적인 하드웨어 부담 없이 적용될 수 있으리라 예상된다.

본 논문에서 제시된 분기 방향 정보를 내포하는 분기 예측 기법이 가질 수 있는 추가적인 장점은 다음과 같다. 우선, 해당 분기 명령어의 추가적인 고유 정보를 사용함으로 인해 분기 예측 테이블을 인덱싱함에 있어서 가명 문제를 보다 더 효율적으로 감소시킬 수 있게 된다. 또한 일반적으로 전역 분기 예측 기법 (Global Branch Prediction Scheme)이나 Global Branch History를 사용하는 다양한 분기 예측 기법에서 문제가 되었던 분기 명령어에 대한 전역 정보 획득용 학습시간 (Warm-Up Time)도 상대적으로 $m-3$으로 감소하게 된다. 그리고 Ⅳ장에서 보이는 바와 같이 내포된 방향 정보에 대한 일반화된 측면에서는, 전체적으로 (m– the number of embedded direction bits) 만큼의 학습시간 단축 효과가 있다.

Ⅳ. 시스템 성능 분석

이 절에서는 본 논문에서 제안한 기법을 평가하기 위한 모의실험을 수행한다. 모의실험에는 SimpleScalar[17]를 사용한다. 이는 비순서 실행(Out-of-Order Issue), 비중단(Non-Blocking) 캐쉬, 임의적 수행(Speculative Execution) 및 최신의 분기 예측 기법 등을 지원하는 강력한 시스템 모의실험 환경이다. 또한 이러한 다양한 형태의 프로세서와 시스템 환경의 묘사가 가능한 SimpleScalar는 이벤트 구동형(Event-Driven) 시뮬레이터로서, 빠른 모의실험과 높은 정확도의 결과를 보장한다.

1. 모의실험 환경 및 벤치마크 프로그램

본 논문에서 사용된 모의실험 환경과 벤치마크 프로그램에 대해서 알아본다. 우선 본 논문에서 제안한 분기 방향 정보를 내포하는 direction-gshare 기법 (d-gshare)과의 비교 대상이 되는 기존의 분기 예측 기법으로 Bimodal 기법(bimod)과 이단계 적응형 분기 예측 기법(2lev), 그리고 gshare 기법(gshare)을 사용하였다. 이와 함께 각 기법에서의 분기 예측 테이블 크기는 512B, 1K, 2K, 4K, 8K, 그리고 16K이다. 분기 예측 부분 이외의 다양한 모의실험 환경은 SimpleScalar에서 제공하는 기본 설정(Default Option)[17]을 사용하였다.

사용된 벤치마크 프로그램은 SPEC에서 제공하는 CPU 성능평가 프로그램인 SPEC CINT2000과 CFP2000을 사용하였다. 일반적으로 SPEC에서 제공하는 CPU 성능 평가 프로그램 중, CFP는 분기 예측의 정확도가 아주 높은 특성을 가지고 있다. 따라서 분기 예측의 정확도를 평가하는 논문에서는 주로 CINT 벤치마크 프로그램을 많이 사용하고 있다. 본 논문에서도 CINT에 속하는 벤치마크 프로그램의 성능을 주로 평가하며, 아울러 CFP에 속하는 벤치마크 프로그램의 성능 향상도 소개한다. 또한 사용된 벤치마크 프로그램은 임의로 선정된 프로그램들로서, CINT 7개와 CFP 3개이다. 이들은 각각 gcc 컴파일러를 이용하여 ALPHA binary 형태로 컴파일 되었다. 사용된 벤치마크 프로그램에 관련된 간략한 설명이 표 3에 제시되어 있다.

2. 모의실험 결과 및 분석

분기 예측과 관련된 정책이 바뀌었을 때의 가장 분명한 비교 수치는 분기 예측의 정확도(Branch Prediction Accuracy)이다. 모의실험을 통해 표 3에 제시된 각 응용 프로그램별 분기 예측의 정확도를 나타낸 내용이 그림 8에서 그림 13 사이에 나타나 있다. 각각의 그림에서 가로축은 벤치마크 프로그램들이며, 세로축은 분기 예측의 정확도 이다. 이때 분기 예측 테이블의 크기는 앞서 언급한 바와 같이 512B에서 16K까지 변화를 주었으며, 분기 예측의 비교 대상은 기본적인 Bimodal 기법 (bimod)과 이단계 적응형 분기 예측 기법(2lev)과 그리고 이를 변형한 gshare 기법(gshare), 끝으로 본 논문에서 제시한 분기 명령어의 방향 정보를 내포하는 direction-gshare 기법(d-gshare)이다.

우선, 전체적인 관점에서 보았을 경우 실험결과는 다음과 같다. 그림 8에서부터 그림 13에서 보여 지듯이, 각 응용 프로그램별 분기 예측의 정확도에 있어서는 약간의 차이가 존재하긴 하지만, 이단계 적응형 분기 예측 기법과 gshare 기법은 전반적으로 비슷한 양상을 보

인다는 것을 알 수 있다. 분기 예측 테이블의 크기가 512B의 경우는 평균 89%, 그리고 16K인 경우는 평균 95%의 분기 예측 정확도를 가진다. 이와 같이 분기 예측 테이블의 크기가 커질수록 분기 예측의 정확도가 증가하는 이유는, 분기 예측 테이블의 크기가 커질수록 분기 명령어들 사이에 Corelate한 관계를 찾을 수 있는 충분한 전역 이전 기록(Global History)을 확보해 간다는 의미로 해석되며, 또한 충분한 인덱싱 비트의 사용으로 가명 문제가 점차 줄어든 결과라 할 수도 있겠다. 그림에서 보여 지듯이 각 응용 프로그램별로 다소간의 차이가 있긴 하지만 각 응용 프로그램 별로 gcc, mcf, twolf, equake, 그리고 mesa의 경우에는 이단계 적응형 분기 예측 기법이, 그리고 gzip, vpr, eon, gap, 그리고 swim의 경우에는 gshare 기법이 우수한 성능을 보인다는 것을 알 수 있다. 하지만 average 축에서 보여 지듯이 각 크기별 모든 응용 프로그램에 있어서 평균적인 측면은 모두 gshare가 우위에 있다는 것을 알 수 있다. 이러한 결과는 상대적으로 이단계 적응형 분기 예측 기법과 비교하여, gshare 기법이 가명 문제를 더 우수하게 해결한다는 점에서 그 원인을 찾을 수 있다.

한편, Bimodal 기법은 이단계 적응형 분기 예측 기법과 gshare 기법에 비교하여 전반적으로 성능이 떨어진다는 것을 알 수 있다. 이는 Bimodal 기법이 분기 명령어의 Bimodal하다는 특성을 반영하긴 하지만 상대적으로 분기 명령어의 Corelate하다는 성질을 충실히 반영하지 못한 결과라 할 수 있다. 그리고 이러한 성능상의 차이는 분기 예측 테이블의 크기가 512B에서 16K로 점차 커질수록 더욱 증가한다는 것을 알 수 있다. 하지만 분기 예측 테이블의 크기가 512B일 경우는 오히려 Bimodal 기법이 이단계 적응형 분기 예측 기법과 gshare 기법에 비해 분기 명령어의 예측 정확도가 더 우수하다는 것을 알 수 있다. 다시 말해 이를 반대로 해석하자면, 분기 예측 테이블의 크기가 작아질수록 Bimodal 기법과 이단계 적응형 분기 예측 기법 혹은 gshare 기법 사이의 성능 차이가 줄어든다는 사실을 알 수 있으며, 또한 분기 예측 테이블의 크기가 특정 크기 이하일 경우는 오히려 Bimodal 기법이 성능면에서 우수하다는 사실을 알 수 있다. 이는 바꾸어 해석하자면, 작은 크기의 분기 예측 테이블 환경에서는 이단계 적응형 분기 예측 기법과 gshare 기법이 각 분기 명령어들의 전역적 정보를 획득하기 위한 충분한 크기를 확보하지 못함에서 기인한다고 할 수 있다. 이와 같은 결과는 상대적으로 Bimodal 기법이 분기 예측 테이블의 크기가 작은 소규모 시스템이나 내장형 시스템일 경우에 더 유용한 기법이라는 사실을 보여주는 것이다. 이러한 내용은 McFarling[12]의 논문에서 언급된 실험 결과와 일치하는 내용이라 할 수 있다.

끝으로 본 논문에서 제시된 direction-gshare 기법과 나머지 3가지 기법 사이의 실험 결과를 분석 해 보면 다음과 같다. 우선 direction-gshare 기법이 분기 예측 테이블의 크기에 관계없이 모든 경우에 있어서 성능면에서 우위에 있다는 것을 알 수 있다. 특히 앞서 분석한 바와 같이, Bimodal 기법이 우수한 결과를 보이는 512B나 1K와 같은 작은 크기의 분기 예측 테이블 환경에서도 direction-gshare 기법은 Bimodal 기법과 비교하여 각각 평균 0.7%(512B)와 2.7%(1K)의 성능상의 우위를 보인다. 한편 분기 예측 테이블의 크기가 커질수록 이단계 적응형 분기 예측 기법과 gshare 기법의 정확도가

표 3. 벤치마크 프로그램
Table 3. Benchmark Program.

Category	Benchmark	Description
CINT2000	164.gzip	Data compression utility
	175.vpr	FPGA circuit placement and routing
	176.gcc	C programming language compiler
	181.mcf	Minimum cost network flow solver
	252.eon	Ray tracing
	254.gap	Computational group theory
	300.twolf	Place and route simulator
CFP2000	171.swim	Shallow water modeling
	177.mesa	3D Graphics library
	183.equake	Finite element simulation; earthquake modeling

증가하지만, 분기 예측 정확도에 있어서 direction-gshare 기법에 미치지 못한다. 가장 근소한 경우가 분기 예측 테이블의 크기가 16K인 경우로 direction-gshare 기법이 0.4% 우수한 성능을 보이는 경우이며, 전체적으로 평균 1.5% 정도의 성능 향상을 가진다. 이러한 결과는 direction-gshare 기법이 소규모의 내장형 시스템에서 뿐만 아니라, 비교적 큰 크기의 분기 예측 테이블을 포함하고 있는 중대형 컴퓨터 시스템에서도 다양하게 사용될 수 있음을 의미한다. 일반적으로 소규모 시스템은 모바일 혹은 내장형 시스템에 주로 적용될 수 있으며, 이 경우 하드웨어 자원의 제약 특성으로 인해 일반적으로 낮은 분기 예측 정확도를 가질 수 있으나 상대적으로 저전력 특성과 이동성을 보장받을 수 있게 된다. 이 경우 direction-gshare 기법은 상대적으로 열악한 하드웨어 조건 하에서도 기존의 Bimodal 기법과 비교하여 우수한 성능을 얻을 수 있다. 한편 중대형 컴퓨터 시스템은 서버급이나 워크스테이션으로 활용될 수 있는 시스템을 의미하는 것으로, 많은 하드웨어 자원을 투자해 최고의 성능을 얻고자 함이 주된 목적이다. 이러한 경우 역시, 기존의 이단계 적응형 분기 예측 기법과 gshare 기법을 대체해 direction-gshare 기법이 사용 될 수 있으며, 더 나아가 복합 예측 기법의 기본 요소 중 하나로도 활용 될 수도 있다. 이처럼 본 논문에서 제시된 기법이 분기 예측 정확도적인 측면에서 우수한 이유는 이단계 적응형 분기 예측 기법에서 문제가 되었던 가명 문제를 gshare 기법에서처럼 direction-gshare 기법에서도 같은 방식으로 해결함과 아울러, 신경망의 분석결과에서 나타난 높은 가중치의 요소를 분기 예측 기법에 새롭게 적용한 결과라 하겠다. 또한 이처럼 추가된 분기마다의 고유한 분기 방향 정보가 가명 문제의 해결에 보조적인 역할까지 수행했다고 해석할 수 있을 것이다. 한편 direction-gshare 기법은 기존의 이단계 적응형 분기 예측 기법 혹은 gshare 기법의 문제라 할 수 있는 분기 명령어의 전역 이전 기록에 대한 긴 학습 시간 측면이 있어서도, 분기 방향 정보가 내포되어 있는 만큼 학습 시간이 짧아지기 때문에 Warm-Up 시간적인 측면에서도 유리하다고 할 수 있다. 또한 이와 같은 특성은 실험 환경의 한계로 인해 모의실험에서는 반영되지 않았지만, 실제 구현에 있어서 다양한 응용 프로그램간의 문맥 교환 (Context Switching)이 자주 발생하는 다중 프로그래밍(Multi-Programming) 환경에서 더욱 우수한 장점으로 작용할 수 있다. 이상의 결과에서 보여주듯이 direction-gshare 기법은, 다양한 시스템 환경과 다양한 응용 프로그램과의 상관관계 속에서, 전반적으로 평균 성능면에서 우수한 결과를 보인다는 것을 알 수 있다.

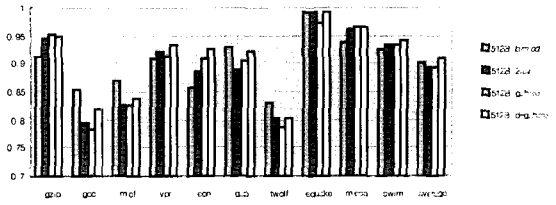

그림 8. 512B Table Size
Fig. 8. 512B Table Size.

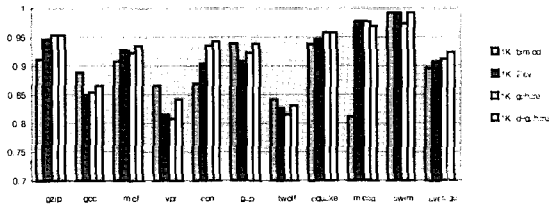

그림 9. 1K Table Size
Fig. 9. 1K Table Size.

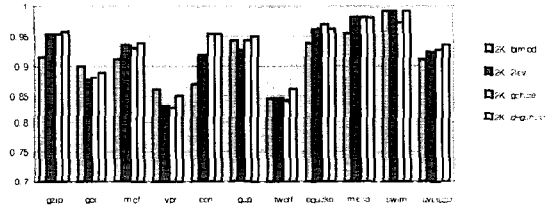

그림 10. 2K Table Size
Fig. 10. 2K Table Size.

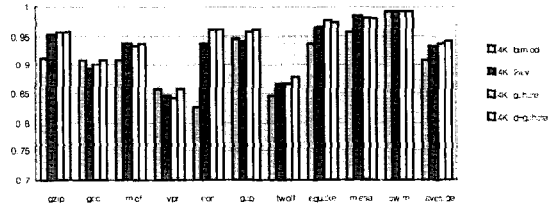

그림 11. 4K Table Size
Fig. 11. 4K Table Size.

한편, 이상의 결과는 Ⅲ장의 3절에서 제시된 분기 방향 정보를 내포하는 분기 예측 기법에 있어서 기본적으로 3 비트의 분기 방향 정보를 이용하였다. 이와 같은

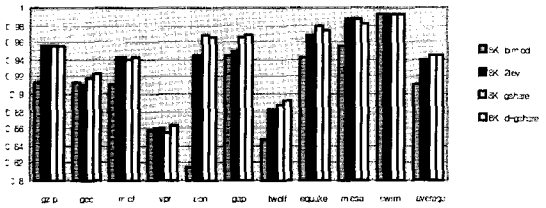

그림 12. 8K Table Size
Fig. 12. 8K Table Size.

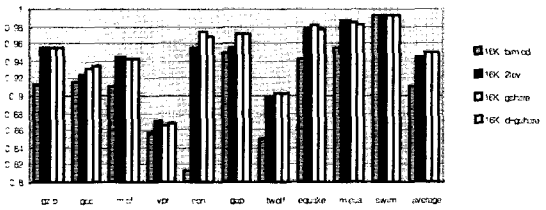

그림 13. 16K Table Size
Fig. 13. 16K Table Size.

비트 값은 III장 1절에서의 신경망 구성 및 그에 대한 모의실험의 분석 결과로 설정된 것이다. 즉, 분기에 영향을 미치는 각 요소별 가중치의 경향 중에서 최근 3개의 Branch Direction History의 가중치 경향이 특히 높게 나타남에서 기인된 결과였다. 하지만 그림 3에서 그림 5 사이에서도 보여 지듯이, 이러한 경향은 각 응용 프로그램들마다 서로 간에 다소의 차이를 보인다. 따라서 내포된 분기 방향 정보에 대한 값의 양을 조절해 가면서 모의실험을 수행 해 볼 필요가 있다. 이에 대한 실험 결과가 각 특성별로 구분되어 그림 14에서 그림 17 사이에 나타나 있다. 모의실험에서는 최소 2 비트(d-gshare.2)에서부터 최대 5 비트(d-gshare.5)까지 분기 방향 정보의 내포 정도를 달리 하였다.

그림 14는 direction-gshare 방향 내포 비율에 대한 특성 1로서, 분기 방향 정보가 내포되어 그 비율이 커질수록 분기 예측 결과의 정확도가 더욱 증가 해 나가는 경향을 나타낸다. gcc, vpr, gzip, gap이 여기에 해당하는 응용 프로그램들이며, d-gshare.5가 최상의 성능을 보이는 환경 설정이다. 이들은 그림 3에서 그림 5 사이에 나타난 결과 중, 경향 1과 경향 2에 해당하는 응용 프로그램들이다. 이러한 결과는, 신경망에서 Branch Direction History의 가중치가 높게 나타난 응용 프로그램들은 실제 분기 예측에 정확도를 높이는데 해당 분기의 Branch Direction History가 더 많은 영향력을 가진다는 의미로 해석 될 수 있다. 참고로 신경망의 구성을

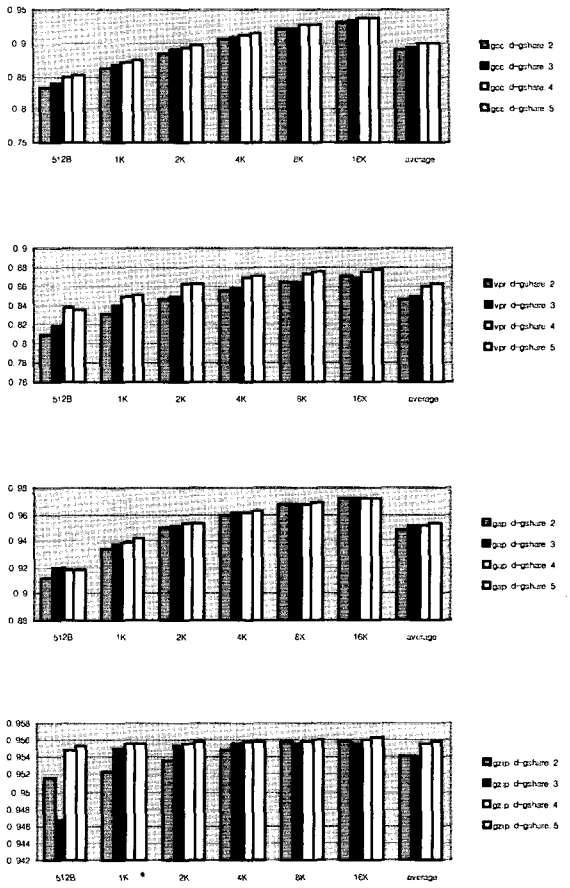

그림 14. direction-gshare 방향 내포 비율에 대한 특성 1, 내포량에 따른 정확성의 선형적 증가, d-gshare.5가 최상의 성능
Fig. 14. direction-gshare with amount of embedded directions pattern 1, Linear improvement of accuracy, Best case: d-gshare.5.

통한 가중치 분석에서 사용되었던 art의 경우는 분기 예측에 있어서 예측 결과의 정확도가 기본적으로 99.5%를 초과하는 응용 프로그램이다. 따라서 신경망의 분석에서는 art를 사용하였지만, 분기 예측 정확도의 분석에서는 이를 배제하였다. 그 외에도 일반적으로 SPEC CFP 계열의 응용 프로그램들은 분기 예측의 정확도가 상당히 높기 때문에 분기 예측과 관련된 연구에서 주로 배제된다.

그림 15는 direction-gshare 방향 내포 비율에 대한 특성 2로서, twolf와 swim이 여기에 해당하는 응용 프로그램들이다. 그림 15에 해당하는 응용 프로그램들도 그림 14에서처럼 분기 방향 정보가 내포되어 그 비율이 커질수록 분기 예측 결과의 정확도가 더욱 증가 해 나가는 경향을 나타낸다. 하지만 그림 14의 응용 프로그램들과 비교하여, 그 증가하는 정도에 있어서 약간의

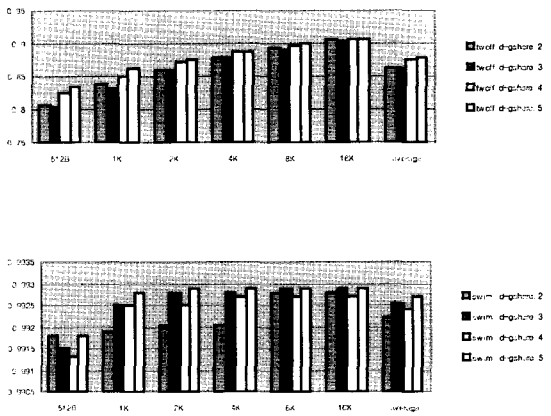

그림 15. direction-gshare 방향 내포 비율에 대한 특성 2, 내포량에 따른 정확성의 일반적 증가, d-gshare.5가 최상의 성능

Fig. 15. direction-gshare with amount of embedded directions pattern 2, General improvement of accuracy, Best case: d-gshare.5.

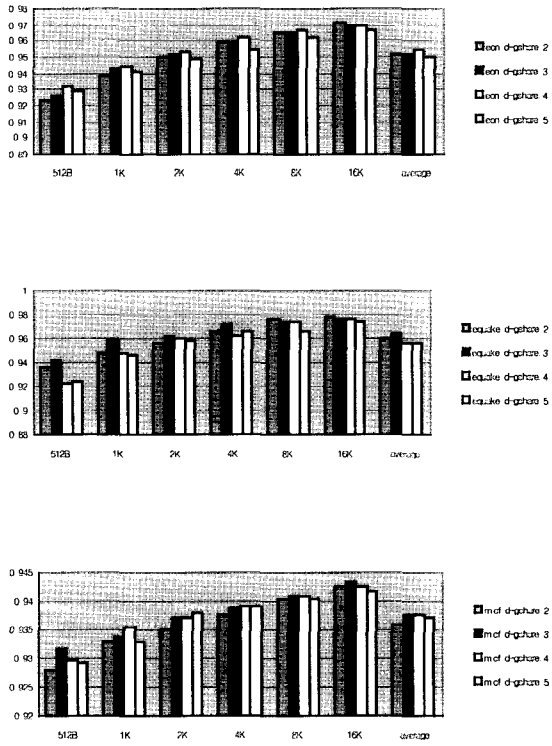

그림 16. direction-gshare 방향 내포 비율에 대한 특성 3, 특정 내포량에서의 최상 성능, d-gshare.4 for eon & d-gshare.3 for equake, mcf

Fig. 16. direction-gshare with amount of embedded directions pattern 3, Best accuracy in case of specific amount of embedded directions, d-gshare.4 for eon & d-gshare.3 for equake, mcf.

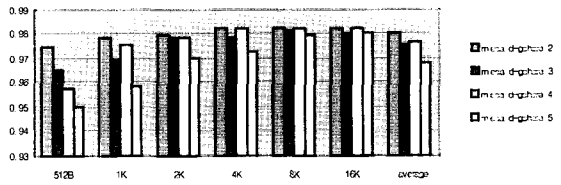

그림 17. direction-gshare 방향 내포 비율에 대한 특성 4, 내포량에 따른 정확성의 감소, d-gshare.2가 최상의 성능

Fig. 17. direction-gshare with amount of embedded directions pattern 4, General decrease of accuracy, Best case: d-ghsare.2.

불규칙성을 가진다. twolf의 경우는 d-gshare.3의 경우에 있어서, 그리고 swim의 경우에 있어서는 d-gshare.4의 경우에 있어서 선형적인 성능 증가 수치를 가지지 못하였다.

하지만 이들 역시 그림 14에서처럼 d-gshare.5가 가장 우수한 성능을 보인다는 점에서는 공통점을 가진다.

그림 16은 direction-gshare 방향 내포 비율에 대한 경향 3으로, 분기 방향 정보가 내포되어 감에 있어서 각 응용 프로그램별 특정한 최적의 방향 정보 내포량을 가지는 경우이다. 그림 16의 eon은 4 비트의 방향 정보 내포량이, equake와 mcf의 경우는 3 비트의 방향 정보 내포량이 각각 최적의 성능을 보였다. 이처럼 그림 16에 해당되는 응용들은 그림 3에서 그림 5 사이에 나타난 결과 중에서 주로 경향 2에 해당하는 응용의 예에서 찾아 볼 수 있다. 즉, Global Direction History와 비교하여 유사한 가중치의 경향을 가지지만, 상대적으로 그렇게 크지 않은 Branch Direction History의 가중치를 가지는 응용의 경우가 주로 그림 16과 같은 특정 방향 정보 내포량의 경우 최적의 성능을 보이는 경우이다.

끝으로 그림 17은 direction-gshare 방향 내포 비율에 대한 특성 4로서, 분기 방향 정보가 내포되어 그 비율이 커질수록 분기 예측 결과의 정확도가 오히려 감소하는 경향을 나타내는 경우이다. 본 논문에서 사용된 응용 프로그램들 중에서 mesa만이 유일하게 이와 같은 결과를 나타내었다.

이상의 그림 14에서 그림 17까지에서 보여준 바와 같이, 성능 향상의 정도를 방향 내포 비율에 대한 특성까지 고려하여 결과를 분석하면 다음과 같다. 앞서 설명한 바와 같이, 그림 8에서 그림 13까지의 실험결과는 방향 정보를 내포하는 기본적인 분기 예측 기법으로 d-gshare.3에 해당하는 결과였다. 이 기법은 Bimodal

표 4. 각 기법의 분기 예측 평균 정확도
Table 4. Average Branch Prediction Accuracy.

	bimod	2lev	gshare	d-gshare.2	d-gshare.3	d-gshare.4	d-gshare.5
512B	0.90179	0.89451	0.89392	0.90636	0.90849	0.91174	0.91199
1K	0.89686	0.90948	0.91156	0.9205	0.9226	0.92595	0.92549
2K	0.91204	0.92188	0.92486	0.93053	0.93272	0.93577	0.93486
4K	0.90949	0.93141	0.93626	0.93907	0.94	0.94229	0.94181
8K	0.90952	0.93904	0.94447	0.94565	0.94567	0.94758	0.9469
16K	0.91016	0.94567	0.95007	0.95013	0.9495	0.95114	0.95066

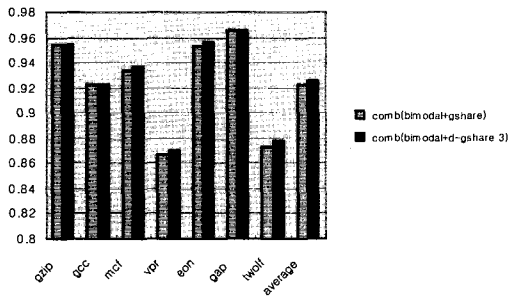

그림 18. 조합 예측 기법
Fig. 18. Combinational Predictor.

기법이나 이단계 적응형 분기 예측 기법 혹은 그의 변형인 gshare 기법에 비하여 최대 4.1%, 평균 1.5% 더 우수한 결과를 가지는 것을 보였다. 하지만 최적의 방향 정보 내포량에 해당하는 실험결과를 통한 비교 수치는 Bimodal 기법이나 이단계 적응형 분기 예측 기법 혹은 그의 변형인 gshare 기법에 비하여 최대 11.8%, 평균 3.7% 더 우수한 결과를 보였다. 이와 같은 결과는 각 응용 프로그램별 최적의 방향 정보 내포량에 대한 선택의 중요성을 보여준다고 할 수 있다.

표 4는 지금까지 서술한 각각의 분기 예측 기법들에 대해서, 응용 프로그램들의 평균 분기 예측 정확도를 direction-gshare의 분기 방향 정보 내포량과 비교하여 함께 나타낸 것이다. 이러한 결과값은 각 응용 프로그램들의 특성에 종속적이지만, 전반적인 결과는 앞서 서술한 바와 유사하다. 결론적으로, 본 논문에서 제안된 분기 방향 정보를 내포하는 분기 예측 기법은 논문에서 사용된 전체 10가지의 응용 프로그램 중 mesa의 경우를 제외한 나머지 90%의 응용 프로그램에서 성능상 보다 더 우수한 결과를 보였으며, 전체의 60%는 일반적으로 분기 방향 정보 내포량의 증가에 따라 더 우수한 분기 예측의 정확성을 보였다. 또한 30%의 경우에 있어서는 적어도 특정 분기 방향 내포량에 있어서 최적의 결과를 보였다. 이를 통해, 분기 방향 정보의 내포량을 각 응용 프로그램별 수행 패턴에 맞게 동적으로 조절하는 기법에 대한 연구도 필요하리라 생각된다.

끝으로 그림 18은 조합 예측 기법의 실험 결과를 보여준다. 최근의 분기 예측 기법은 점차 복합 예측 기법화 되어가는 추세를 보인다. 그림 18도 그 중의 하나로써, 2K 크기의 Bimodal과 1K 크기의 gshare 기법이 복합적으로 사용되는 조합 예측 기법의 SPEC CINT 2000 벤치마크 실험 결과이다. 실험 결과에서도 보여주듯이, 기존의 조합 예측 기법에 비하여 d-gshare.3이 사용된 기법이 모든 응용에 있어서 분기 예측의 정확도가 우수함을 알 수 있다. 즉, 제안된 새로운 기법은 조합 예측 기법의 실험 결과에서도 보여 주듯이 분기 예측의 구성 요소 중 하나로써 Global Branch History를 사용하는 기존의 여러 복합 예측 기법(Hybrid Prediction Scheme)에서도 추가적인 하드웨어 부담 없이 적용되어 성능의 향상을 얻을 수 있으리라 예상된다.

V. 결론 및 향후 과제

파이프라인과 슈퍼스칼라 방식 그리고 동적 스케쥴링 기법이 일반화된 시스템 구조 하에서, 분기 명령어에 대한 분기 예측의 정확도는 프로세서 입장에서 뿐만 아니라 시스템 전체적인 성능에 있어서 큰 영향을 미친다. 특히 프로세서 내부의 파이프라인의 깊이가 깊어지고 동시에 이슈되는 명령어의 개수가 증가할수록, 그리고 보다 더 다양한 스케쥴링 기법이 사용될수록 이러한 중요성은 더욱 커진다고 할 수 있다.

본 논문에서는 이러한 분기 예측의 정확도를 높이기 위하여, 기존의 시스템에서 일반적으로 사용되는 이단계 적응형 분기 예측 기법과 gshare 기법의 새로운 대안으로 direction-gshare 기법을 제안하였다. direction-gshare 기법은 분기 예측에 있어서 중요한 구성 요소인 Global Branch History와 PC 값과 아울러, 추가적인 Branch Direction History를 사용한다. 이러한 구성은 분기 명령어의 분기 예측 결과에 영향을 미치는 각 구

성 요소별 가중치의 분석을 위해 설정된 신경망에 근거하여 도출된 결과를 이용한 것이다. 즉, 분기 예측에 영향을 미치는 각 요소별로의 가중치의 변화를 분석하여, 높은 영향을 미치는 요소를 분기 예측에 새롭게 적용한 기법이다. 실험 결과 제안된 새로운 기법은, 기존의 gshare 기법과 비교하여 추가적인 하드웨어의 요구 없이 대부분의 경우에 있어서 기존의 방법과 비교하여 우수한 성능을 나타내었다. 특히 Bimodal 기법에 비해서 각 응용 프로그램 별로 최대 4.1%, 이단계 적응형 분기 예측 기법과 비교하여 1.5%, 그리고 gshare 기법과 비교하여 1.6%의 성능 향상을 가져왔다. 또한, 최적화된 방향 정보 내포량에 대해서는 최대 11.8%, 평균 3.7% 우수한 결과를 보였다. 또한 제안된 새로운 기법은 기존의 Global Branch History의 값을 Global Branch History와 Branch Direction History 값으로 양분하여 사용함으로, 조합 예측기법의 실험 결과에서도 보여주듯이 분기 예측의 구성 요소 중 하나로써 Global Branch History를 사용하는 기존의 여러 복합 예측 기법(Hybrid Prediction Scheme)에서도 추가적인 하드웨어 부담 없이 적용될 수 있으리라 예상된다.

본 논문의 향후 연구과제로는 다음과 같은 사항을 들 수 있다. 우선, 보다 더 다양한 응용 프로그램을 통한 추가적인 실험이 필요하며, 분기 예측 테이블의 크기도 더욱 다양화해 보아야 할 것이다. 그리고 비교 대상을 더욱 확대하여 하드웨어 비용적인 측면에서 차이가 있긴 하지만 다중 복합 예측 기법(Complex Hybrid Branch Prediction Scheme)이나 토너먼트 예측 기법(Tournament Prediction Scheme)과의 성능상의 차이점도 보다 면밀히 분석해 보아야 할 것이다.

참 고 문 헌

[1] J. L. Hennessy and D.A Patterson, Computer Architecture : A Quantitative Approach, Third Edition, Morgan Kaufmann Publishers, Inc, 20036.

[2] S. Haykin, Neural Networks : A Comprehensive Foundation, Second Edition, Prentice Hall, 1999.

[3] Daniel A. Jimenez and Calvin Lin, Neural Methods for Dynamic Branch Prediction, ACM Transactions on Computer Systems, Vol. 20, No. 4, November 2002.

[4] Rosenblatt, F. Principles of Neurodynamics : Perceptrons and the Theory of Brain Mechanisms. Spartan, New York, 1962.

[5] L. Faucett, Fundamentals of Neural Networks : Architectures, Algorithms and Applications. Prentice-Hall, Englewood Cliffs, NJ, 1994.

[6] Arun D. Kulkarni, Artificial Neural Networks for Image Understanding. Van Nostrand Reinhold, 1993.

[7] Block, H. D., The Perceptron : A Model for Brain Functioning. Rev. Mod. Phy. 34, 123-135, 1962.

[8] Yeh, T. Y. and Patt, Y. N., Two-level adaptive branch prediction. In Proceedings of the 24th ACM/IEEE International Symposium on Micro-architecture, 51-61, 1991.

[9] Sechrest, S., Lee, et al., Corelation and Aliasing in Dynamic Branch Predictors, in Preceedings of the 23rd ISCA, 22-32, 1996.

[10] Eric Sprangle et al., "The Agree Predictor : A Mechanism for Reducing Negative Branch History Interference", IEEE ISCA '97

[11] Chih-Chieh Lee et al., "The Bi-Mode Branch Predictor",International Symposium on Micro-architecture IEEE '97

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

[13] Brad Calder, D. Grunwald, M. Jones, D. Lindsay, J. Martin, M. Mozer, and B. Zorn. Evidence-based static branch prediction using machine learning. ACM Transactions on Programming Languages and Systems, 19(1), 1997.

[14] Jimenez, D. A. and Lin, C., Dynamic branch prediction with percetprons. In Proceedings of the 7th International Symposium on high Performance Computer Architecture, 197-206, 2001.

[15] G. Steven, et al., Dynamic Branch Prediction using Neural Networks, In Proceedings of the Euromicro Symposium on Digital Systems Design, IEEE, 2001.

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

[17] 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

저 자 소 개

곽 종 욱(정회원)
1999년 경북대학교 컴퓨터공학과 학사
2001년 서울대학교 컴퓨터공학과 석사
2002년~현재서울대학교 전기·컴퓨터공학부 박사 과정.
<주관심분야: 컴퓨터 구조, 고성능 병렬 처리, 저전력 내장형 시스템>

전 주 식(정회원)
1974년 서울대학교 수공학과 학사
1976년 한국과학기술원 전산학 석사
1982년 University of Utah 전산학 박사
1982년~1984년 University of Iowa 연구원
1985년~현재 서울대학교 전기·컴퓨터공학부 교수
<주관심분야: VLSI, CAD, 병렬 컴퓨터 구조>

김 주 환(정회원)
2001년 서울대학교 컴퓨터공학과 학사
2001년~현재 서울대학교 전기·컴퓨터공학부 석·박사 통합과정
<주관심분야: 컴퓨터 구조, 멀티프로세서 시스템, VLSI>