The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 14, No. 6, pp.259-265, Dec. 31, 2014. pISSN 2289-0238, eISSN 2289-0246

http://dx.doi.org/10.7236/JIIBC.2014.14.6.259
JIIBC 2014-6-37

멀티코어 프로세서의 통계적 모의실험에 관한 연구

A Study on Statistical Simulation of Multicore Processor Architectures

이종복*

Jongbok Lee*

요 약 현재 널리 이용되는 멀티코어 프로세서 구조의 설계 초기에 그 성능을 분석하기 위하여 명령어 트레이스 모의실험을 이용하는 경우, 시간과 공간을 많이 차지하기 때문에 비실용적이다. 본 논문에서는 프로화일링 기법에 기반하는 통계적 모의실험에 의하여 다양한 하드웨어 사양을 갖는 멀티코어 프로세서의 성능을 측정하는 기법에 대하여 연구하였다. 이것을 위하여 SPEC 2000 벤치마크 프로그램의 특성을 통계적 프로화일링 기법으로 모델링하고 여기서 얻은 통계적 프로화일을 바탕으로 벤치마크 트레이스를 합성하여 멀티코어 프로세서에 대한 모의실험을 수행하였다. 그 결과, 통계적 모의실험에 의하여 측정한 성능이 명령어 트레이스 모의실험에 의하여 측정한 성능에 근접한 결과를 가져왔으며 모의실험 시간을 크게 단축시켰다.

Abstract When the trace-driven simulation is used for the performance analysis of widely used multicore processors in the initial design stage, much time and disk space is necessary. In this paper, statistical simulations are performed for a high performance multicore processor with various hardware configurations. For the experiment, SPEC2000 benchmarks programs are used for profiling and synthesizing new instruction traces. As a result, the performance obtained by our statistical simulation is comparable to that of the trace-driven simulation with the benefit of tremendous reduction in the simulation time.

Key Words : Multi-core processor, Statistical profiling, Statistical simulation

Ⅰ. 서 론

현재 멀티코어 프로세서 구조가 스마트폰, 태블릿 PC, 노트북, 데스크탑 등과 같은 컴퓨터 시스템의 성능 향상을 높이기 위하여 널리 쓰이고 있다[1-5]. 이러한 멀티코어 프로세서의 개발 단계에서 그 성능을 평가하기 위하여 광범위한 모의실험이 행하여지며, 명령어 트레이스 모의실험 (trace-driven simulation)과 실행 위주 모의실험 (execution-driven simulation)이 그 예이다. 명령어 트레이스 모의실험은 모의실험이 적게 걸린다는 장점이 있으나, 비교적 정확도가 떨어지며 디스크 공간이 매우 많이 소요된다. 반면에 실행 위주 모의실험은 정확도가 높지만 모의실험 시간이 과다하게 소요되는 단점이 있다. 두 가지 방법 모두, 하드웨어 사양이 바뀔 때마다 모의실험

*정회원, 한성대학교 정보통신공학과
접수일 : 2014년 11월 3일, 수정완료일 : 2014년 12월 3일
게재확정일 : 2014년 12월 12일

Received: 3 November, 2014 / Revised: 3 December, 2014 / Accepted: 12 December, 2014
*Corresponding Author: jblee@hansung.ac.kr
Dept. of Information and Communications Engineering, Hansung University, Korea

을 다시 반복해야하는 단점이 있다.

따라서, 이러한 난관을 타개하기 위한 여러가지 방법이 연구되어왔다. 통계적 프로화일링(statistical profiling) 방법은 프로세서와 프로그램의 통계적 특성을 수집하고 그것을 바탕으로 새로운 입력 트레이스를 합성하여, 이것을 확률적으로 모의실험하는 것이다 [6,7]. 통계적 프로화일링을 위하여, 프로그램의 특성과 더불어 프로세서에 대한 특성을 통계적 방법에 의하여 얻는다. 또한, 이러한 통계적 프로화일은 통계적 명령어 트레이스를 새로 합성하는데 이용된다. 합성된 명령어 트레이스는 그 크기가 작지만, 통계적 프로화일링 기법으로 임의로 발생하였기 때문에 각 벤치마크 프로그램의 특성을 함축적으로 잘 대표한다. 따라서 합성된 새로운 명령어 트레이스는 간단한 명령어 트레이스 모의 실험기에서 실행 가능하며, 기존의 일반적인 명령어 트레이스 모의실험 보다 훨씬 짧은 시간에 성능에 대한 예측을 비교적 정확히 할 수 있으므로, 멀티코어 프로세서의 초기 설계 과정에서 성능 평가를 시행할 때 유용하게 이용할 수 있다. 본 논문에서는 일곱 개의 SPEC 2000 벤치마크의 정수형 프로그램을 대상으로 다양한 사양의 멀티코어 프로세서의 성능을 통계적 프로화일링 기법을 이용하여 측정하였다. 그리고 이것을 기존의 일반적인 명령어 트레이스 모의실험으로 측정한 결과와 비교하여 그 정확도를 평가하였다.

본 논문은 다음과 같이 구성된다. 2 장에서는 통계적 프로화일링 기법에 대하여 논하고, 3 장에서는 모의실험 환경을 다룬다. 4 장에서 모의실험결과를 보이고, 5 장에서 결론을 맺는다.

II. 통계적 프로화일링 기법

통계적 모의실험의 전 과정을 자세히 살펴보면 그림 1과 같이 크게 4 단계로 나누어지는데, 첫째, 일반적인 프로그램 트레이스의 발생, 둘째 통계적 프로화일링 기법에 의한 분석 및 통계 데이타의 수집, 셋째, 통계적 트레이스의 합성, 넷째 합성된 트레이스에 대한 통계적 트레이스 모의실험이다.

첫번째의 프로그램 트레이스 발생은 기존의 방법과 동일하다. 특정한 벤치마크 프로그램을 구체적인 명령어 트레이스 발생기를 통하여 명령어 트레이스를 생성한다.

두번째의 통계적 프로화일링을 위한 분석 및 통계 데이타의 수집 단계는 다음과 같다. 우선, 첫 단계에서 얻은 프로그램의 명령어 트레이스를 분석하여 프로그램의 고유한 특성과 지역적 특성에 대한 통계값들을 추출해낸다. 프로그램의 고유한 특성은 프로그램을 구성하는 명령어의 유형별 구성비와 레지스터 피연산자의 개수 및 명령어 간의 데이터 종속에 대한 분포로 구성된다. 이 때, 명령어의 유형별 분포는 원래의 아키텍처가 갖는 명령어 집합에서 총 아홉 개의 간단한 명령어 집합으로 축소시킨다. 한편, 명령어 간의 상호 레지스터 종속은, 유형별 명령어의 소오스 레지스터와 그 레지스터를 목적 레지스터로 하여 종속을 부여하며 선행하는 명령어 간의 쓰레드 내에서의 거리로 정한다. 이러한 특성은 주어진 마이크로 프로세서의 구조와는 무관하고 단지 컴파일러와 마이크로 프로세서의 명령어 집합에만 영향을 받는 고유한 성질이다. 지역적 특성은 쓰레드 예측 미스율이나 캐쉬 미스율 등을 의미하며, 이것은 마이크로 프로세서 하드웨어 구조에 의하여 영향을 받는다. 이 단계는 각 벤치마크 프로그램에 대하여 단 한 번만 시행한다.

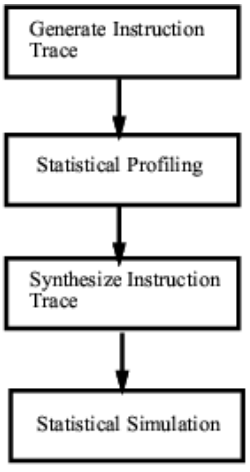

그림 1. 통계적 모의실험의흐름도
Fig. 1. The flow of statistical simulation

세번째, 위에서 얻은 통계적 특성을 기반으로 하여 난수를 발생시켜서 통계적 특성을 갖는 새로운 명령어 트레이스를 합성한다. 통계적 트레이스를 발생시키는 방법은 0부터 1 사이의 난수를 발생시키고 이 값을 통계적 프로화일링에 의하여 얻은 누적 분포 함수에 대응시키는 것이다. 이 방법으로, 명령어의 유형 뿐만이 아니라, 피연산자의 개수, 피연산자의 종속거리를 통계적 프로화일링을 기반으로 결정하여 명령어 코드를 만들어낼 수 있기

때문에 이 작업을 반복하면 프로그램의 특성을 함축적으로 내포하는 짧은 길이의 통계적 트레이스의 합성이 가능하다.

마지막으로, 이렇게 합성된 트레이스를 분기 히트율 및 캐쉬 히트율에 대한 정보와 함께 멀티코어 프로세서 모의실험기로 입력하여 그 성능을 측정한다. 한편, 통계적 프로화일링에 의하여 자료를 확보한 후에는, 하드웨어 조건을 바꿔서 전체 설계 공간에 대하여 다양한 시도를 할 수가 있다. 즉, 코어의 개수, 쓰레드의 크기, 명령어의 인출율, 명령어의 실행 지연 싸이클, 파이프라인 단계의 수를 변화시켜가면서 새로운 결과를 간편하게 얻을 수가 있으므로 모의실험을 다양하고 신속하게 실행할 수 있다.

Ⅲ. 모의실험 환경

1. 멀티코어 프로세서의 구조

그림 2는 N 개의 코어로 구성되는 멀티코어 프로세서의 일반적인 구조를 나타낸 것이다[8-10]. 이 때, 한 개의 코어는 일정한 크기의 쓰레드로 구성되는 순차 (in-order) 수퍼스칼라 프로세서가 이용된다.

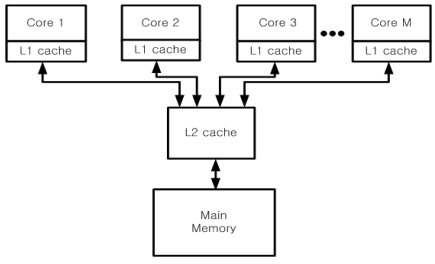

그림 2. 멀티코어 프로세서의 구조
Fig. 2. The multi-core processor architecture

한편, 각 코어는 자체적으로 1 차 명령어 캐쉬와 1 차 데이터 캐쉬를 가지며, 또한 메인 메모리와 연결되는 공통의 2 차 통합 캐쉬를 공유한다. 각 코어에 설치된 1 차 데이터 캐쉬의 일관성(cache-coherency)을 위하여 MESI 프로토콜을 이용하여, 어느 코어에서 공유된 캐쉬 메모리에 쓰기 작업을 하였을 때, 나머지 코어에서는 해당 데이터를 무효화 (write-invalidate) 시킨다. 이하 본 논문에서 기술하는 명령어 캐쉬와 데이터 캐쉬는 모두 1 차 캐쉬를 의미한다.

표 1은 모의실험에 이용된 멀티코어 프로세서 아키텍처의 사양을 나타낸 것이다. 멀티코어의 구조는 듀얼코어, 쿼드코어, 옥타코어를 대상으로 하였다. 각 코어는 수퍼스칼라 방식으로 운영되므로, 매 싸이클 마다 1 개에서 N 개의의 명령어를 인출, 이슈, 실행 및 종료한다. 각 코어의 연산유닛은 정수형 유닛, 로드 스토어 유닛, 분기 유닛으로 구성된다. 명령어 캐쉬와 데이터 캐쉬는 각 코어마다 설치되는데, 64KB의 용량을 갖도록 설정하였으며, 2 차 연관도 (2-way set associativity) 방식을 통하여 접근된다. 그러나, 모든 코어에 의하여 공유되는 2 차 캐쉬는 충분한 용량으로 인하여 100 % 히트가 난다고 가정하였다.

2. 멀티코어 프로세서 모의실험기

본 논문에서는 명령어 트레이스 모의실험기를 개발하여 모의실험에 이용하였다 [11]. 멀티코어 프로세서는 제 1 단계 명령어 자취의 발생, 제 2 단계 명령어 자취에 대한 멀티코어 프로세서의 실행으로 나누어진다. 제 1 단계에서 명령어 자취는 Simplescalar를 이용하여 SPEC 벤치마크 프로그램으로부터 임의의 차수의 멀티코어에 적합하도록 발생되었다[12].

표 1. 멀티코어 프로세서 아키텍쳐 하드웨어의 사양
Table 1. The architecture specification of multicore processor architecture

항목	값
멀티코어 수	4, 8, 16
코어의 구조	순차 수퍼스칼라
쓰레드의 크기	4/8/16
인출율, 이슈율, 퇴거율	4/8/16
명령어 캐쉬 및 데이터 캐쉬의 공통 사항	64 KB, 2 차 연관, 16 B 미스 페날티 10 싸이클
연산유닛 개수	산술논리(1/2/4/8), 분기(1), 로드(1/2), 스토어(1),
쓰레드 어드레스 캐쉬	2 K 엔트리
쓰레드 예측기	2 단계 14 비트 전역 히스토리 방식 미스 페날티 6 싸이클
이슈 지연 싸이클	산술논리(1), 분기(1), 로드(1), 스토어(1),
결과 지연 싸이클	산술논리(1), 분기(1), 로드(1), 스토어(1),

제 2 단계에서 각 코어가 순차 수퍼스칼라로 동작하는 멀티코어 프로세서에 입력된다. 이 때, 쓰레드 수준 병렬성이 쓰레드로 매핑되어 각 코어에서 실행된다. 제 2 단계의 과정을 자세하게 기술하면 그림 3에 나타낸 것과 같다.

(1) 명령어 인출, 재명명 및 이슈

초기화 작업을 거친 후에, 각 코어는 매 싸이클마다 1 개 또는 N 개의 명령어를 인출받는다. 인출한 명령어는 재명명 (renaming) 작업을 거치면서 명령어 종속에 의한 타임스탬프(timestamp) 값을 설정받는다. 타임스탬프 방식은 명령어 자취를 이용하는 모의실험에서 데이터 종속성을 신속하고 효율적으로 부여할 수 있는 방법이다. 레지스터 화일의 타임스탬프 값에 의하여, 멀티코어 프로세서 명령어 간의 종속성이 유지되어 성능을 구하는데 반영된다.

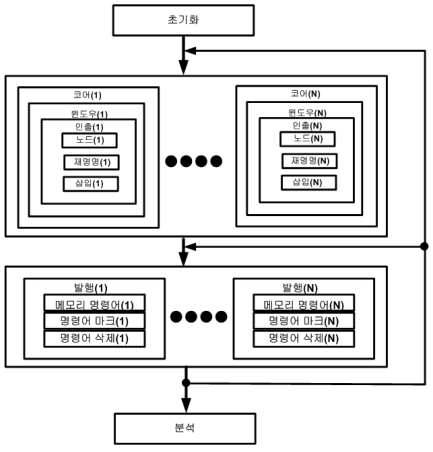

그림 3. 멀티코어 프로세서 모의실험기의 흐름도
Fig 3. The flow chart of digital signal mulicore processor simulator

한편, 재명명을 거친 명령어는 각 코어의 쓰레드에 삽입된다. 싸이클이 증가함에 따라서 쓰레드 내의 명령어는 자체의 타임스탬프 값이 현재 싸이클 보다 작거나 같을 때 실행되어 삭제될 수 있다.

(2) 멀티코어 시뮬레이션

N 개의 멀티코어에 대하여 해당 코어의 윈도우 공간에 적절하게 1 개 또는 M 개의 명령어를 인출해서 채우고, 역시 N 개의 멀티코어에 대하여 각 코어에 대하여 명령어를 실행하면서 종속성에 의하여 부여된 명령어의 타임스탬프가 충족되면 삭제한다. 이 과정은 코어 내부 및 코어 간의 레지스터 종속 및 메모리 종속 검사에 적용되며, 입력으로 주어진 벤치마크 프로그램의 모든 명령어가 소진될 때까지 반복된다.

위 과정이 한번 실행될 때 마다 싸이클이 증가하므로, 매 싸이클 당 명령어의 실행 및 삭제가 가장 오래 걸리는 코어가 해당 싸이클 수를 결정한다. 모의실험에 입력으로 쓰인 명령어의 총 개수를 처리하기 위하여 소요된 총 싸이클 수로 나누어, 멀티코어 프로세서 시스템의 성능의 척도인 IPC(Instruction Per Cycle)를 계산할 수 있다.

3. 벤치마크의 사양

표 2는 모의실험에 이용된 일곱 개의 SPEC 2000 벤치마크 프로그램이다. 각 벤치마크 프로그램은 Simplescalar를 통하여 MIPS IV 10억 개의 명령어 자취를 임의의 차수의 멀티코어 프로세서에 적합하도록 발생시켜서 모의실험기에 입력하였다.

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

벤치마크	설 명
bzip2	압축
crafty	체스 경기 놀이
gap	그룹이론 해석기
gcc	GNU C 컴파일러
gzip	압축
paser	워드 프로세서
twolf	배선 및 배치 모의실험기

Ⅳ. 모의실험 및 결과

그림 4에 세 가지의 쓰레드 길이에 대한 듀얼 코어, 쿼드코어 및 옥타코어 프로세서의 성능을, 명령어 트레이스 모의실험으로 측정한 값과, 통계적 모의실험에 의하여 측정한 값을 비교하여 나타냈다.

그림 4(a)와 4(b)는 쓰레드의 길이가 4일 때의 모의실험 결과이다. 듀얼코어일 때, 명령어 트레이스 모의실험에 의하여 측정한 성능의 기하평균은 1.1 IPC를 기록하였으며, 통계적 모의실험에 의한 값은 1.3 IPC를 나타냈다. 쿼드코어일 때는 명령어 트레이스 방법으로 2.0 IPC, 통계적 방법으로 2.4 IPC를 얻을 수 있었다. 옥타코어일 때는 명령어 트레이스와 통계적 모의실험에 의하여 측정한 값이 각각 3.2 IPC와 4.3 IPC를 기록하였다. 듀얼코어와 쿼드코어에서는 최저 21 %의 상대오차를 기록하였으나, 옥타코어에서 오차가 32 %로 증가하였다.

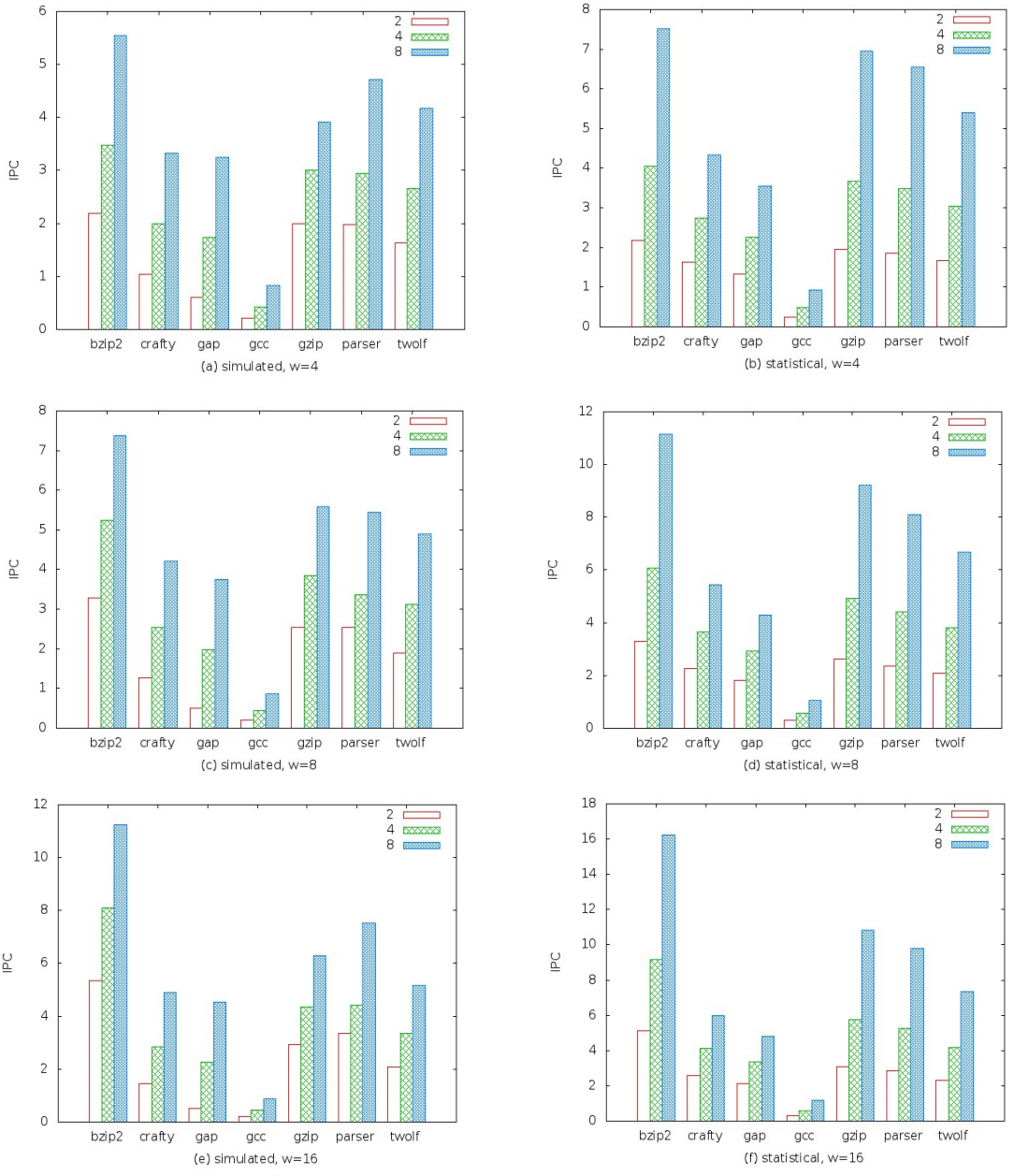

그림 4. 명령어 트레이스 모의실험과 통계적 모의실험에 의하여 측정한 성능의 비교
Fig. 4. The performance comparison of the trace-driven simulation and statistical simulation

그림 4(c)와 4(d)는 쓰레드의 길이가 8일 때의 모의실험 결과를 비교하여 나타낸 것이다. 명령어 트레이스 모의실험에 의하여 측정한 듀얼코어, 쿼드코어, 옥타코어 프로세서의 기하평균값은 각각 1.3 IPC, 2.4 IPC, 4.0 IPC를 나타냈으며, 이에 대응하는 통계적 모의실험에 의하여 측정한 값은 각각 1.8 IPC, 3.1 IPC, 5.4 IPC를 기록하였다. 세 가지 멀티코어 프로세서에서 구한 상대오차의 평균은 35 %이다.

마지막으로, 그림 4(e)와 4(f)는 멀티코어 프로세서를 구성하는 코어의 쓰레드의 길이가 16일 때의 모의실험

결과를 비교하여 나타낸 것이다. 듀얼코어일 때 명령어 트레이스 모의실험이 1.5 IPC, 통계적 모의실험이 2.1 IPC를 가져왔다. 쿼드코어일 때 각 값들은 2.8 IPC와 3.7 IPC를 기록하였으며, 옥타코어일 때는 각각 4.7 IPC와 6.4 IPC로 다소 오차가 증가하였다. 그러나 상대오차의 평균값은 35 %를 넘지 않았다.

위에서 살펴본 것과 같이, 통계적 모의실험에 의하여 측정한 멀티코어 프로세서의 성능이 명령어 트레이스 모의실험에 의하여 측정한 성능과 같은 경향을 보이며, 세 가지 쓰레드의 길이에 대하여 측정한 성능의 정확도가 평균 32 %임을 알 수 있다. 한편, 통계적 모의실험에 소요되는 시간은 명령어 트레이스 모의실험에 소요되는 시간보다 평균 30 배 빠른 것으로 측정되었다. 따라서, 통계적 모의실험의 성능의 정확도는 모의실험에 소요되는 시간에 의하여 보완된다.

V. 결 론

본 논문에서는 순차 수퍼스칼라로 구성되는 멀티코어 프로세서 아키텍처에 대하여 통계적 모의실험을 통하여 성능을 측정하고 결과를 분석하였다. 통계적 모의실험은, 입력으로 이용되는 벤치마크 프로그램의 명령어 트레이스에 대한 사전 프로화일링을 통하여 합성된 새로운 명령어 자취를 이용하기 때문에, 명령어 데이터 크기가 작고 모의실험 시간을 대폭 단축할 수 있다는 장점이 있다. 모의실험 결과, 통계적 모의실험 방법이 비록 명령어 트레이스에 의한 측정값보다 평균 32 %의 오차가 발생하였으나, 모의실험 시간을 30 배 빠르게 개선시킬 수 있었다.

향 후의 연구과제는, 통계적 모의실험에 의한 성능의 정확도를 더욱 높이는 방법을 모색하고, 임베디드 및 디지털 신호처리 멀티코어 프로세서에 대한 통계적 모의실험을 시행하는 것이다.

References

[1] Jongbok Lee, “A Study of Trace-driven Simulation for Multi-core Processor Architectures,” The Journal of the Institute of Internet, Broadcasting and Communication, Vol. 12, No. 3, Jun. 2012, pp. 9-13.

[2] Jongbok Lee, “Performance Analysis of Multicore Processor Architectures Based On Cache Size Effects,” The Journal of the Institute of Internet, Broadcasting and Communication, Vol. 12, No. 6, Dec. 2012, pp. 175-180.

[3] Jongbok Lee, “Performance Study of Multi-core In-Order Superscalar Processor Architectures,” The Journal of the Institute of Internet, Broadcasting and Communication, Vol. 12, No. 5, Oct. 2012, pp. 123-128.

[4] Jongbok Lee, “A Performance Study of Embedded Multicore Processor Architectures,” The Journal of the Institute of Internet, Broadcasting and Communication, Vol. 13, No. 1, Feb. 2013, pp. 163-169.

[5] Jongbok Lee, “Performance Study of Multicore Digital Signal Processor Architectures,” The Journal of the Institute of Internet, Broadcasting and Communication, Vol. 13, No. 14, Aug. 2013, pp. 171-177.

[6] Jongbok Lee, “The Processor Performance Model Using Statistical Simulation,” The Jounral of the Korean Institute of Information Scientists and Engineers, Vol. 33, No. 5-6, Jun. 2006, pp. 297-305.

[7] R. Carl and J. E .Smith, "Modeling Superscalar Processors via Statistical Simulation," Workshop on Performance Analysis and Its Impact on Design, Jun. 1998.

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

[9] G. S. Sohi, S. E. Breach, and T. N. Vijaykumar, "Multiscalar Processors," Proceedings of the 22nd annual international symposium on Computer architecture, pp. 414-425, May 1995.

[10] T-Y. Yeh and Y. N. Patt, "Alternative Implementations of Two-Level Adaptive Branch Prediction," in Proceedings of the 19th International Symposium on Computer Architecture, pp.124-134, May. 1992.

[11] A. Rico, A. Duran. F. Cabarcas, Y. Etsion, A.

Ramirex, and M. Valero, "Trace-driven Simulation of Multithreaded Applications," ISPASS, 2011.

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

저자 소개

이 종 복(정회원)

- 1964년 8월 20일생.
- 1988년 : 서울대 컴퓨터공학과 졸업.
- 1998년 : 동 대학 전기공학부 졸업 (공박).
- 1998년 ~ 2000년 : LG반도체 선임연구원.
- 2000년 ~ 현재 : 한성대 정보통신공학과 교수

- Tel : 02-760-4497
- Fax : 02-760-4435
- E-Mail : jblee@hansung.ac.kr

※ 본 연구는 한성대학교 교내학술연구비 지원과제 임.