2003년 대한전자공학회 하계종합학술대회 제26권 제1호

다양한 cache block 크기에 의한 시스템의 성능 변화

이 성 환, 김 준 성
중앙대학교 전자전기공학부

Impacts of multiple cache block sizes on system performance

Seong-Hwan Lee, JunSeong Kim
School of Electrical and Electronics Engineering,
Chung-Ang University
E-mail : lshcau@hanmail.net, junkim@cau.ac.kr

Abstract

본 논문에서는 instruction과 data cache로 나누어지는 L1 cache를 가진 시스템에서 instruction과 data cache 각각의 block 크기 변화가 전체 시스템의 성능에 미치는 영향을 고찰하였다. 이를 위하여 SPEC CPU 벤치마크 프로그램을 입력으로 하는 SimpleScalar를 이용한 시뮬레이션을 수행하였다. 본 연구를 통해서, instruction과 data 각각의 특성에 맞는 cache block 크기를 사용하는 것이 일률적인 cache block 크기를 사용하는 것에 비하여 전체 시스템의 성능을 더욱 향상시켜 준다는 것을 보여준다.

1. 서 론

20세기 중반 이후에 급속히 발전한 컴퓨터 산업은 인류의 생활을 편리하게 하는데 크게 기여했다. 컴퓨터 시스템은 CPU, 메인보드, 메모리 등이 결합되어 구성되므로 구성 요소간의 균형 있는 발전이 필요하다. 그런데 CPU의 처리속도는 매우 빠르게 발전하였고 메인 메모리에서 데이터를 읽고 쓰는 속도는 그에 따르지 못하는 불균형으로 인하여, CPU와 메인 메모리 사이에서 데이터를 주고받는데 병목현상이 발생하였다. 이런 문제점을 개선하기 위해 CPU와 메인 메모리 사이에서 CPU가 원하는 데이터를 빠른 속도로 공급할 수 있는 cache라는 보조 메모리가 도입되었다[1]. 대부분의 마이크로프로세서에 사용되고 있는 cache는 level-1 (이하 L1)과 level-2 (이하 L2)의 두 계층으로 구분되고, L1 cache는 동일한 block 크기를 가지는 instruction과 data cache로 더욱 세분화되어 사용되고 있다.

본 논문에서는 instruction과 data cache로 나누어지는 L1 cache에 있어서 instruction과 data의 서로 다른 특성에 관계없이 동일한 block 크기를 가지는 시스템에 비해서 instruction과 data 각각의 특성에 맞는 서로 다른 block 크기를 가지는 시스템이 전체 시스템의 성능에 어떤 변화를 보이는지를 관찰하고자 한다.

2장에서는 본 연구의 시뮬레이션을 위해서 사용될 시뮬레이션 프로그램인 SimpleScalar와 그 입력으로 사용될 SPEC CPU 벤치마크 프로그램에 대하여 기술하고, 실험에 사용될 시스템의 구성을 간략히 소개한다. 3장에서는 시뮬레이션 결과의 miss rate 그래프를 이용하여 instruction과 data의 서로 다른 특성에 대해서 확인하며, 다양한 cache block 크기에 의해서 시스템의 성능이 어떻게 변하는지 살펴본다. 마지막으로 4장에서 본 연구를 요약하고 결론을 맺는다.

2. 배경 및 실험

본 연구에서는 instruction과 data cache의 다양한

block 크기에 따른 시스템의 성능 변화를 관찰하기 위해서 SimpleScalar 시뮬레이터 버전 3.0에 SPEC CPU 벤치마크 프로그램을 입력으로 하여 실험하였다.

2.1 SimpleScalar

SimpleScalar는 MIPS-IV 구조를 기반으로 하는 execution-driven 시뮬레이터로서 이식성을 증가시키기 위해서 little-endian과 big-endian 버전을 동시에 정의하여 마이크로프로세서 연구에 널리 사용되는 시뮬레이션 프로그램이다[2][3]. SimpleScalar는 sim-fast, sim-safe, sim-cache, sim-profile, sim-outorder의 다섯 가지 시뮬레이터로 구성된다. 이중 sim-outorder 시뮬레이터는 다른 시뮬레이터의 기능을 총 망라하는 것으로, RUU(register update unit)에 기반한 out-of-order issue와 execution을 지원하는 세부적이고 실질적인 프로세서 시뮬레이터이다. 본 연구는 sim-outorder 시뮬레이터를 이용한 시뮬레이션을 통하여 시스템의 구성 요소 중 하나인 cache block 크기의 변화에 따른 시스템의 성능 변화를 고찰한다.

2.2 SPEC CPU 벤치마크 프로그램

본 연구에서 SimpleScalar의 입력으로 사용한 것은 SPEC(Standard Performance Evaluation Corporation)에서 여러 마이크로프로세서들의 성능을 테스트하는 표준을 제공하기 위해서 만든 SPEC CPU 벤치마크 프로그램이다[4][5]. SPEC CPU 벤치마크 프로그램은 프로세서, 메모리 구조, 컴파일러의 성능 비교를 위해 사용되며 크게 정수형 연산을 중심으로 하는 CINT와 실수형 연산을 중심으로 하는 CFP의 두 그룹으로 구성된다.

binary	group	description
go	CINT95	An internationally ranked go-playing program
tomcatv	CFP95	Vectorized mesh generation
fpppp	CFP95	From Gaussian series of quantum chemistry benchmarks
vpr	CINT2000	FPGA circuit placement and routing
gcc	CINT2000	C programming language compiler
vortex	CINT2000	Object-oriented database

표 1. 실험에 사용된 SPEC CPU 벤치마크 프로그램.

실험을 위하여 SPEC CPU95 (8개의 CINT95와 10개의 CFP95로 구성)와 SPEC CPU2000 (12개의 CINT2000과 14개의 CFP2000으로 구성) 벤치마크 프로그램이 고려되었으며, 본 논문에서는 임의로 선택된 6개의 프로그램에 대한 결과를 포함한다. 표 1은 본 연구에서 사용된 벤치마크 프로그램에 대한 간단한 설명을 보여준다.

2.3 시스템의 구성

표 2는 본 연구의 시뮬레이션을 수행하기위한 시스템 구성을 보여준다. 실험에 사용될 시스템의 cache는 L1과 L2의 두 계층으로 구분되며, 서로 다른 L1 cache 크기에 대하여 다양한 조합의 L1 cache block 크기를 고려하여 실험하였다.

L1 cache는 동일한 크기의 instruction과 data cache로 구성된다. 예를 들어, 32Kbyte의 L1 cache는 16Kbyte의 instruction cache와 16Kbyte의 data cache의 조합을 의미한다. 또, Simplescalar에서 제공하는 cache block 크기의 범위는 256byte가 한계이므로, 16byte에서 256byte까지 변화시키면서 실험하였다. L2 cache는 instruction과 data의 구분이 없는 unified cache를 사용하였다.

description	value
L1 cache size (Kbyte - I + D)	32 ~ 512
L2 cache size (Mbyte - Unified)	1
L1 cache block size (byte)	16 ~ 256
L2 cache block size (byte)	256
cache set associativity	4-way
cache replacement	LRU
L1 cache hit latency (clock cycle)	1
L2 cache hit latency (clock cycle)	6
memory bus latency (clock cycle)	18
memory access bus width(byte)	8
instruction TLB table entry	64
data TLB table entry	128
TLB miss latency (clock cycle)	30
branch predictor type	bimodal
BTB table entry	2048
한번에 fetch되는 instruction의 수	4
register update unit의 size	16
load/store queue의 size	8
instruction dedode의 bandwidth (instruction/cycle)	4
instruction issue의 bandwidth (instruction/cycle)	4
instruction commit의 bandwidth (instruction/cycle)	4

표 2. 시스템 구성표

3. 실험 결과 및 분석

3.1 Cache block 크기에 따른 miss rate

그림 1과 2는 32Kbyte 크기의 L1 cache를 갖는 시스템에서 L1 instruction 및 data cache 각각에 대하여 다양한 cache block 크기에 의한 miss rate의 변화를 보여준다. 그래프의 X축은 16~256 byte 범위에서 변화하는 instruction과 data cache의 block 크기를 나타내고, Y축은 각각 instruction과 data cache의 miss rate을 의미하며, 서로 다른 line들은 벤치마크 프로그램의 종류를 나타낸다.

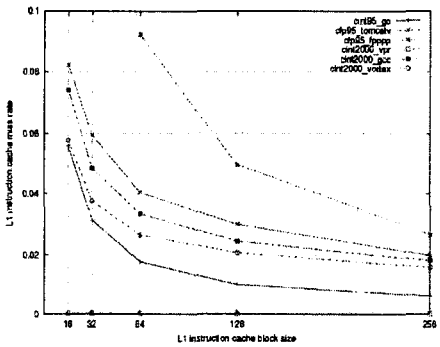

그림1. L1 instruction cache의 miss rate 변화

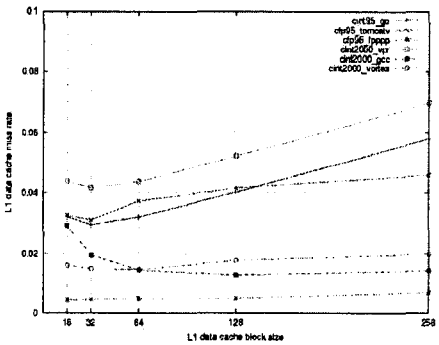

그림2. L1 data cache의 miss rate 변화

그림 1의 instruction cache의 경우, block 크기가 증가할수록 miss rate이 점차로 감소하는 것을 볼 수 있다. 이는 instruction cache의 access pattern이 spatial locality를 충실히 반영하기 때문이다[1]. 반면에 그림 2의 data cache의 경우, block 크기가 증가함에 따라서 miss rate은 감소하다가 다시 증가하는 경향을 보이는 것을 확인할 수 있다. 이는 data cache의 access pattern에 spatial locality와 temporal locality의 trade-off가 반영되기 때문이다[1]. 앞의 결과에서 주목할 점은 instruction과 data cache 각각의 경우 miss rate이 최소가 되는 block 크기가 서로 다르다는 것이다. 이는 instruction과 data의 서로 다른 특성에 의해서 각각의 cache가 최적화되기 때문이며, instruction과 data cache 각각에 서로 다른 block 크기를 사용하는 것이 동일한 block 크기를 사용하는 것에 비해 시스템 성능을 향상시킬 수 있다는 것을 의미한다.

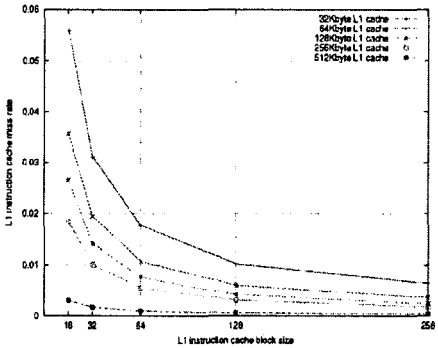

그림 3. go의 L1 instruction cache의 miss rate 변화

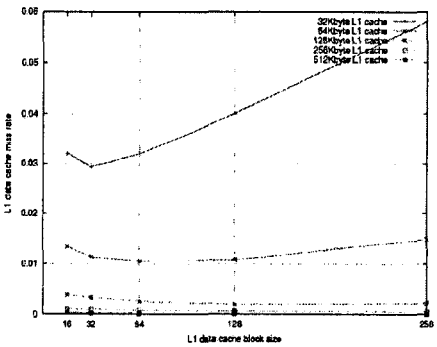

그림 4. go의 L1 data cache의 miss rate 변화

이와 같은 instruction과 data cache에 대한 특성은 다른 크기의 L1 cache에 대하여도 유효하다. 그림 3과 4는 go 벤치마크 프로그램에 대하여 그림 1과 그림2의 instruction과 data cache 각각의 특성을 32K, 64K, 128K, 256K, 512Kbyte의 서로 다른 크기의 L1 cache를 사용하여 재구성하였다. 예상하였던대로, L1 cache의 증가에 따라 instruction과 data cache 모두에서 miss rate이 감소되는 점을 제외하면, instruction과 data cache의 block 크기의 변화에 의한 miss rate의 변화 패턴이 L1 cache 크기에 무관하게 유지됨을 알 수 있다.

3.2 Cache block 크기에 따른 시스템 성능 변화

그림 5와 6은 go와 vortex 벤치마크 프로그램에서 instruction과 data cache 각각의 block 크기를 다양하게 변화시킬 경우 시스템 성능의 상대적인 변화를 보

여준다. 그래프의 X축은 다양하게 조합된 L1 instruction과 data cache의 block 크기를 나타내고, Y축은 L1 instruction과 data cache의 block 크기가 각각 16/16 byte로 동일한 시스템의 성능을 기준으로 하여 다양한 cache block 크기의 조합을 갖는 시스템의 상대적 성능을 나타낸다. 또한, 일반적인 결론을 확인하기 위하여 L1 cache 크기가 32Kbyte (실선표시)와 64Kbyte (점선표시)인 두 가지 경우를 고려하였다.

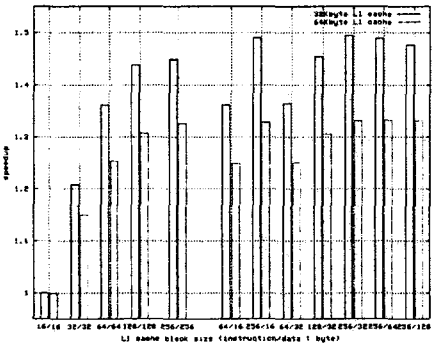

그림 5. go 벤치마크 프로그램의 다양한 cache block 크기에 의한 성능 변화

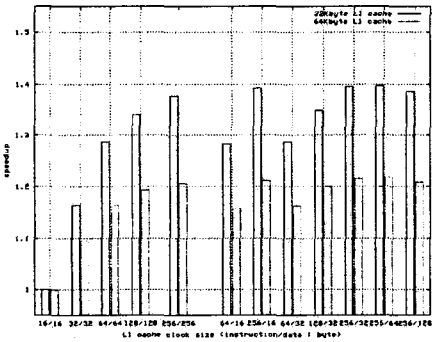

그림 6. vortex 벤치마크 프로그램의 다양한 cache block 크기에 의한 성능 변화

그림 5에서 32Kbyte L1 cache를 가지는 시스템은 L1 instruction과 data cache의 block 크기가 각각 256/32 byte로 구성된 시스템에서 최고의 성능을 보이며, 다음으로 256/16 byte, 256/64 byte, 256/128 byte로 구성된 시스템이 그 뒤를 잇는다. Instruction과 data cache 각각의 block 크기가 서로 다른 256/32 byte로 구성된 시스템은 instruction 과 data cache 각각의 block 크기가 동일한 16/16 byte로 구성된 시스템에 비하여 약 50%의 성능 향상을 가져옴을 알 수 있다. 해당 시스템(256/32 byte)은 역시 instruction과 data cache 각각의 block 크기가 동일한 256/256 byte로 구성된 시스템과 비교하더라도 약 3%의 성능 향상을 가져오는 것을 확인할 수 있다. 이상과 같이 instruction과 data cache의 block 크기가 서로 다른 시스템이 그 block 크기가 동일한 시스템에 비하여 높은 성능을 보이는 현상은 64Kbyte L1 cache를 가지는 시스템에서도 그대로 유효하며, 그림 6의 vortex 벤치마크 프로그램에서도 그대로 유지됨을 알 수 있다. go, vortex 벤치마크 프로그램을 포함하여 실험에 사용된 다른 벤치마크 프로그램인 tomcatv, fpppp, vpr, gcc에서도 마찬가지의 결과가 나오는 것을 확인하였다.

4. 결론

본 논문에서는 cache를 구성하는 요소 중에서 block 크기가 전체 시스템의 성능에 미치는 영향을 알아보기 위해서 SimpleScalar를 이용한 시뮬레이션을 통한 실험을 진행하였다. L1과 L2의 두 레벨로 나누어지는 cache에서 L1 instruction cache는 spatial locality의 영향을 충실히 반영하여 block 크기가 증가할수록 miss rate이 감소하였고, L1 data cache는 spatial locality와 temporal locality의 trade-off에 의해서 block 크기가 크지도 작지도 않은 적당한 지점에서 miss rate이 최소가 되었다. Cache miss rate을 통해서 확인한 instruction과 data cache의 서로 다른 특성을 반영하기 위하여 L1 cache block 크기를 변화시키면서 시스템의 성능을 분석한 결과, instruction과 data cache의 block 크기를 동일하게 구성하는 기존의 L1 cache에 비해, instruction과 data cache의 block 크기를 서로 다르게 구성하는 L1 cache가 시스템의 성능을 향상시킨다는 것을 확인하였다.

Acknowledgement

이 논문은 2002학년도 중앙대학교 학술연구비 지원에 의하여 연구되었음.

참고문헌

[1] John L.Hennessy, David A.Patterson, "Computer Architecture - A Quantitative Approach", Morgan Kaufmann, 2003.

[2] SimpleScalar, "http://www.simplescalar.com/"

[3] Austin T., Larson E., Ernst D., "SimpleScalar: an infrastructure for computer system modeling", IEEE Computer, Feb. 2002 pp. : 59-67

[4]Standard Performance Evaluation Corporation, "http://www.spec.org/"

[5] Henning, J.L., "SPEC CPU2000: measuring CPU performance in the New Millennium", IEEE Computer, July 2000 pp. : 28-35