■ 2003년 정보과학 논문경진대회 수상작

압축블록의 압축률 분포를 고려해 설계한 내장캐시 및 주 메모리 압축시스템

(An On-chip Cache and Main Memory Compression System Optimized by Considering the Compression rate Distribution of Compressed Blocks)

임 근 수 † 이 장 수 †† 홍 인 표 ††† 김 지 홍 ††††
(Keun Soo Yim) (Jang-Soo Lee) (Inpyo Hong) (Jihong Kim)

김 신 덕 ††††† 이 용 석 †††††† 고 건 ††††††
(Shin-Dug Kim) (Yong Surk Lee) (Kern Koh)

요 약 최근에 프로세서-메모리간 성능격차 문제를 완화하기 위하여 내장캐시의 접근실패율을 낮추고 메모리 대역폭을 확장하는 내장캐시 압축시스템이 제안되었다. 내장캐시 압축시스템은 데이타를 압축해 저장함으로써 내장캐시의 실질적 저장공간을 확장하고, 메모리 버스에서 데이타를 압축해 전송함으로써 실질적 메모리 대역폭을 확장한다. 본 논문에서는 이와 같은 내장캐시 압축시스템을 확장해 기존의 주 메모리 압축시스템과 병합해 설계한 이종 메모리 압축시스템을 제안한다. 주 메모리의 기억공간을 효율적으로 확장하고, 내장캐시의 접근실패율을 낮추고, 메모리 대역폭을 확장하고, 압축캐시의 복원시간을 줄이고, 설계 복잡도를 낮추기 위하여 몇 가지 새로운 기법들을 제시한다. 제안하는 시스템과 비교대상 시스템의 성능은 슈퍼스칼라 구조의 마이크로프로세서 시뮬레이터를 수정하여 실행기반 시뮬레이션을 통해 검증한다. 본 논문에서 사용한 실험방법은 기존의 트레이스기반 시뮬레이션과 비교해 보다 높은 정확도를 갖는다. 실험결과 주 메모리 확장에 따른 이득을 고려하지 않은 경우에 제안하는 시스템은 일반 메모리시스템에 비하여 수행시간을 내장캐시의 크기에 따라 최대 4-23%가량 단축한다. 제안하는 시스템의 데이타 메모리와 코드 메모리의 확장비율은 각각 57-120%와 27-36%이다.

키워드 : 메모리시스템, 메모리압축, 내장캐시, 주 메모리, 프로세서-메모리간 성능격차, 입출력 병목현상

Abstract Recently, an on-chip compressed cache system was presented to alleviate the processor-memory performance gap by reducing on-chip cache miss rate and expanding memory bandwidth. This research presents an extended on-chip compressed cache system which also significantly expands main memory capacity. Several techniques are attempted to expand main memory capacity, on-chip cache capacity, and memory bandwidth as well as reduce decompression time and metadata size. To evaluate the performance of our proposed system over existing systems, we use execution-driven simulation method by modifying a superscalar microprocessor simulator. Our experimental methodology has higher accuracy than previous trace-driven simulation method. The simulation results show that our proposed system reduces execution time by 4-23% compared with conventional memory system without considering the benefits obtained from main memory expansion. The expansion rates of data and code areas of main memory are 57-120% and 27-36%, respectively.

Key words : Memory system, memory compression, on-chip cache, main memory, processor-memory performance gap, and I/O bottleneck

† 학생회원 : 서울대학교 컴퓨터공학부
ksyim@oslab.snu.ac.kr
†† 비 회 원 : IBM poughkeepsie
jangsoo@us.ibm.com
††† 비 회 원 : 연세대학교 전기전자공학부
necross@dubiki.yonsei.ac.kr
†††† 종신회원 : 서울대학교 컴퓨터공학부 교수
jihong@davinci.snu.ac.kr
††††† 정 회 원 : 연세대학교 컴퓨터과학과 교수
sdkim@cs.yonsei.ac.kr
†††††† 비 회 원 : 연세대학교 전기전자공학부 교수
yonglee@yonsei.ac.kr
††††††† 종신회원 : 서울대학교 컴퓨터공학부 교수
kernkoh@oslab.snu.ac.kr
논문접수 : 2003년 6월 2일
심사완료 : 2003년 9월 29일

1. 서 론

지난 20년간 응용프로그램의 평균 메모리 요구량이 매년 50-100%씩 증가해옴에 따라 고성능 컴퓨터시스템에서는 주 메모리의 비용이 크게 증가하고 있다[1]. 고성능 컴퓨터시스템에서 시스템의 성능을 개선하면서도 주 메모리의 비용을 줄일 수 있는 방법 중에 하나는 하드웨어 압축기법에 기반한 주 메모리 압축시스템이다[7-9]. 현재까지 제안된 주 메모리 압축시스템은 압축 페이지를 복원하는 시점에 따라 크게 두 가지로 나뉜다. 한가지 방식은 하드웨어 압축기를 사용해 압축해 주 메모리에 데이타를 압축해 저장하고, 접근시에 외장 L3 캐시에서 복원하는 방식으로 외장 L3 캐시를 사용하는 멀티프로세서 시스템을 위해 설계되었다. 다른 한가지 방식은 페이지 스와핑이 발생하는 경우에 스왑 아웃 대신에 하드웨어 압축기법을 사용해 주 메모리에 설정한 논리적 압축영역에 저장하고, 추후에 접근되는 경우에 복원하는 방식이다. 일반적으로 압축페이지 복원시간이 자기디스크로부터의 페이지 스왑 인 시간에 비해 훨씬 짧다. 이 두 가지 방식의 주 메모리 압축시스템은 페이지 단위로 데이타를 압축함으로써 주 메모리 공간확장을 극대화할 수 있다. 하지만 주 메모리의 공간이 응용프로그램의 총 메모리 요구량보다 클 경우에는 성능상에 이득을 발휘하지 못하며, 전자는 외장 L3 캐시를 사용하기 때문에 싱글프로세서 시스템에 적합하지 않다.

최근에 주 메모리 압축시스템의 복원지점을 내장캐시로 끌어올린 압축 내장캐시 시스템이 제안되었다. 압축 내장캐시 시스템은 주 메모리 공간확장 뿐만 아니라 매년 35-55% 가량씩 증가하는 프로세서-메모리간 성능격차 문제까지도 완화할 수 있다. 현대의 컴퓨터시스템 설계자들은 프로세서-메모리간 성능격차 문제를 완화하기 위해 대형 내장캐시를 사용해 접근실패율을 낮추거나 메모리 대역폭을 확장해 접근실패에 따른 패널티를 낮추는 방식을 사용해왔다. 그러나 대형 내장캐시는 보다 넓은 실리콘면적을 차지해 마이크로프로세서의 생산단가를 높이고, 메모리 대역폭 확장에도 마이크로프로세서의 핀 수와 같은 물리적 한계가 존재한다. 반면에 내장캐시 압축시스템은 물리적 확장 대신에 압축기법을 사용해, 데이타를 압축해 저장함으로써 내장캐시의 기억공간을 실질적으로 확장하고 압축한 데이타를 메모리 버스를 통해 전송함으로써 메모리 대역폭을 논리적으로 확장한다. 이때 메모리 대역폭을 확장하기 위해서는 주 메모리에서도 페이지를 압축해 저장해야 한다. 그러나 기존의 압축 내장캐시 시스템은 프로세서-메모리간 성능격차 문제에 초점을 맞춰 설계해 주 메모리 공간을 효율적으로 확장하지 못한다.

본 논문에서는 프로세서-메모리간 성능격차 문제를 완화하고 주 메모리의 공간을 효율적으로 확장하며 멀티프로세서 시스템뿐만 아니라 싱글프로세서 시스템에도 적합한 메모리 압축시스템을 설계하기 위해서, 기존의 주 메모리 압축시스템과 내장캐시 압축시스템을 병합한 형태의 이종 메모리 압축시스템(Hybrid Memory Compression System, HMCS)을 세 가지 새로운 기법을 바탕으로 하여 제시한다. 첫째, 메모리 계층구조를 고려해 다양한 크기의 버켓(bucket, 데이타를 저장하는 자료구조)을 사용함으로써 내부파편을 줄이고 보다 많은 블록에 압축을 적용해 내장캐시와 주 메모리의 공간 그리고 메모리 대역폭의 확장을 최적화한다. 둘째, 압축블록을 그룹화하여 대표치를 메타데이타로 사용함으로써 압축페이지에 대한 메타데이타의 양을 줄이고 보다 효율적인 접근방식을 제공한다. 셋째, 압축 내장캐시에서 데이타 램을 대칭구조로 구성하고 병렬 복원기를 사용해 압축률 상의 손실을 최소화하며 복원시간을 크게 단축시킨다.

성능평가 과정에서는 제안하는 시스템과 비교대상 시스템의 성능을 기존의 트레이스기반 시뮬레이션보다 높은 정확성을 갖는 실행기반 시뮬레이터를 개발해 사용한다. 시뮬레이션 결과 주 메모리 확장에 따른 성능이득을 고려하지 않은 경우에 HMCS는 일반 메모리시스템에 비해서 수행시간을 내장캐시의 크기에 따라 최대 4-23%가량 단축한다. SPEC CINT2000 벤치마크 [11, 12]를 사용해 측정한 제안하는 시스템의 데이타 메모리와 코드 메모리의 확장비율은 각각 57-120%와 27-36%이다. 성능평가를 통해 제안하는 시스템은 하드웨어 압축기법을 내장캐시와 주 메모리에서 사용해 프로세서-메모리간 성능격차 문제를 완화할 뿐만 아니라 주 메모리 공간을 효율적으로 확장해 입출력 병목현상도 크게 개선함을 알 수 있다.

본 논문의 2장에는 관련연구를 기록하며, 3장에는 제안하는 시스템의 전체구조와 동작방식 그리고 새롭게 제안하는 세 가지 기법들의 설계과정과 특성에 대해 기술한다. 4장에서는 실험환경과 방법에 대해 설명하며, 5장에서는 개발한 실행기반 시뮬레이터를 사용해 제안하는 시스템과 비교대상 시스템의 성능을 측정해 평가한다. 그리고 6장에서 결론을 맺는다.

2. 관련 연구

그림 1은 복원지점에 따라 분류한 주 메모리 및 내장캐시 압축시스템의 구조이다. 현재까지 발표된 압축 내장캐시 시스템으로는 그림 1(a)와 같은 구조를 사용해 프로세서-메모리간 성능격차 문제를 효율적으로 완화시키는 *Selective Compressed Memory System*(SCMS)

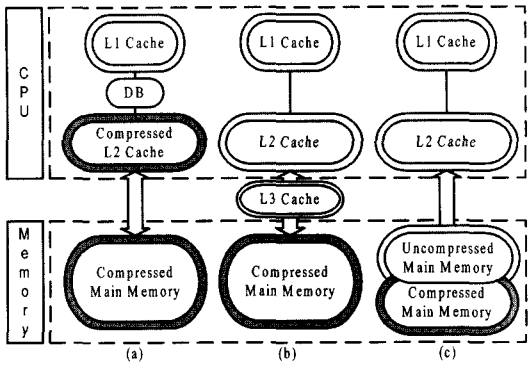


그림 1 복원지점에 따른 주 메모리 및 내장캐시 압축시스템 분류도(a: SCMS, b: MXT, c: CMS)

[2,15]이 유일하다. SCMS는 동적사전 기반 압축알고리즘인 X-RL을 사용한다[3,4]. 왜냐하면 X-RL은 캐시블록과 같은 작은 크기의 블록에 대해서도 높은 압축효율을 보이며, 압축블록을 복원하는 속도가 항상 4바이트/사이클보다 빠르기 때문이다. 그러나 일반적으로 내장 L2 캐시의 접근시간은 해당 캐시블록의 복원시간보다 짧기 때문에 SCMS와 같은 내장캐시 압축시스템에서 복원시간은 성능향상을 저해하는 대표적인 요인이다. SCMS에서는 이와 같은 복원시간을 줄이기 위한 몇 가지 새로운 기법이 제안되었다. 대표적으로 내장 L1 캐시와 L2 캐시사이에 완전연관 구조를 갖는 작은 크기의 복원버퍼(decompression buffer, DB)를 사용하는 방법, 주 메모리에서 블록을 읽어오는 경우 전송시간과 복원시간을 중첩해 복원시간을 감추는 기법, 그리고 선택적 압축기법이 있다.

선택적 압축기법은 캐시블록의 압축률이 임계 압축률보다 낮은 경우에만 해당블록을 압축한 형태로 사용하는 방식으로, 압축에 따른 성능이득이 높고 복원시간이 상대적으로 짧은 블록에만 압축을 적용해 성능향상을 최적화하는 기법이다. 이때 압축률은 압축블록의 크기를 원 블록크기로 나눈 값으로 압축이 잘된 블록이 보다 작은 값을 갖으며, X-RL은 연속된 널(null) 문자를 1사이클에 복원할 수 있어 일반적으로 압축률이 낮은 블록이 높은 블록보다 상대적으로 짧은 복원시간을 갖는다. SCMS는 성능향상을 최적화하고 내장캐시에서 효율적인 압축블록 관리방식을 사용하기 위해서 압축률 50%를 임계 압축률로 사용한다. 임계 압축률 이하로 압축된 캐시블록은 원 블록크기의 절반크기의 버켓에 저장하고 그 외의 캐시블록은 압축하지 않은 형태로 원 블록크기의 버켓에 저장한다. 주 메모리에서는 페이지 내부의 모든 캐시블록이 50% 이하로 압축된 경우에만 원 페이지 크기의 절반인 메모리 버켓을 할당하며 그렇지 않은 경우에는 원 페이지 크기의 메모리 버켓을 할당한다. 이와 같은 SCMS의 버켓팅 방식은 압축블록 및 페이지에 대한 메타데이타의 양을 줄이고 보다 효율적인 접근방식을 지원하기 위해 설계되었다.

그러나 SCMS의 버켓팅 방식은 내장 캐시와 메모리 대역폭 그리고 주 메모리 공간의 효율적 확장문제와 주 메모리에서의 페이지 재배치 문제를 야기한다. 그림 2는 SimpleScalar[13,14]에서의 내장 L2 데이타 캐시블록의 압축률 분포이다. SCMS에서는 이 분포도 상에 보이는 압축률 51-75%사이의 비교적 많은 양의 블록을 압축하지 않으며, 압축률이 50%에 비해 크게 낮은 블록의 경우에는 저장시 내부파편을 많이 발생시켜 압축에 따른 이득을 감쇄시킨다. 뿐만 아니라 주 메모리상의 압축페이지 내부의 임의의 한 캐시블록의 압축률이 50% 이하에서 그 초과로 변화하면 해당 페이지를 재배치해야 하는 추가비용을 갖는다.

대표적인 주 메모리 압축시스템으로는 IBM에서 제안한 그림 1(b)와 같은 구조의 *Memory Expansion Technology*(MXT)[5]가 있다. MXT는 하드웨어 압축기를 사용해 주 메모리에 페이지를 압축해 저장하고, 접근시에는 고속 병렬복원기[10]를 사용해 외장 L3 캐시에 복원해 저장하는 방식을 사용한다. MXT는 주 메모리를 페이지 단위로 압축해 내장캐시 압축시스템의 캐시블록보다 큰 압축단위를 사용함으로써 주 메모리 공간확장을 극대화한다. 하지만 외장 L3 캐시는 일반적으로 싱글프로세서 시스템에서 사용하지 않는 메모리 계층으로 MXT는 멀티프로세서 시스템에 보다 적합하다.

주 메모리 압축시스템의 다른 분류에는 그림 1(c)와 같은 구조의 *Compressed Memory System*(CMS)[6]가 있다. CMS는 주 메모리의 일정영역을 논리적 압축영역으로 설정해 스왑 아웃되는 페이지를 압축해 저장하고, 추후에 해당 페이지에 접근하는 경우에 하드웨어 복원기와 관련 메모리 관리유닛을 사용해 복원해 보조기억장치로부터의 스왑 인 시간보다 짧은 메모리 접근

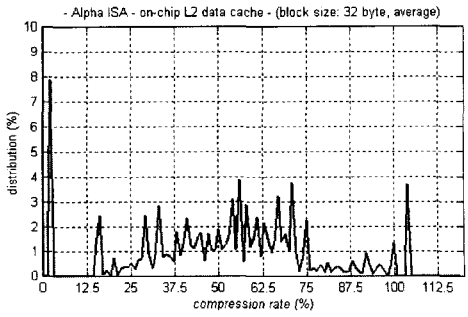


그림 2 SimpleScalar/AlphaISA 구조의 내장 L2 데이타 캐시의 압축률 분포

시간을 제공하여 입출력 병목문제를 완화한다. 하지만 MXT와 CMS 모두 주 메모리 공간이 응용프로그램의 총 메모리 요구량보다 큰 경우에는 성능상의 이득을 발휘하지 못한다.

본 논문에서는 이상에서 살펴본 내장캐시 압축시스템과 주 메모리 압축시스템을 병합해 하나의 내장캐시 및 주 메모리 압축시스템을 설계하여 프로세서-메모리간 성능격차 문제를 완화하고 주 메모리의 공간을 효율적으로 확장하며, 멀티프로세서 시스템뿐만 아니라 싱글프로세서 시스템에도 적합한 메모리시스템을 설계하고 성능을 평가한다. 제안하는 시스템의 핵심적인 특성은 병합구조, 계층적 버캣팅 방식, 블록 그룹화 기법, 그리고 대칭형 내장캐시 구조이다.

3. 이종 메모리 압축시스템(Hybrid Memory Compression System, HMCS)

이 장에서는 HMCS의 전체구조와 구조적 특징들을 소개한다.

3.1 병합 구조

그림 3은 HMCS의 전체구조와 성능상의 이득을 시각화한 것이다. 코드 메모리시스템과 데이타 메모리시스템의 구조는 서로 다른데 그 이유는 명령어 캐시블록의 압축특성과 데이타 캐시블록의 압축특성이 서로 다르기 때문이다.

명령어 캐시에 압축을 적용하면 명령어 캐시블록의 평균 압축률이 캐시블록 크기가 작고 블록내부 데이타 사이에 연관도가 낮아 약 80%로 높아 성능이득은 적은데 반하여 복원비용이 존재해, 전체 메모리시스템의 성능을 저하시킬 수 있기 때문에 명령어 캐시에서는 압축방식을 사용하지 않는다. 일반적으로 압축효율은 압축단위의 크기에 비례해 증가하는데, 주 메모리 코드 페이지의 경우 평균압축률이 70% 이하로 최대 43%까지 공간

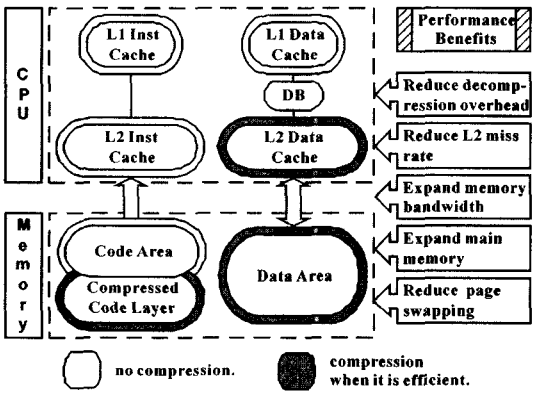


그림 3 HMCS의 전체구조와 성능상의 이득

을 확장할 수 있다. 하지만 코드 페이지의 평균 복원시간은 약 1024 사이클로 상대적으로 길기 때문에 메모리 접근시간을 크게 지연시킬 수 있다. 내장 L2 명령어캐시에서 압축방식을 사용하지 않기 때문에 코드 페이지의 복원시간을 어느 정도 감추기 위해서는 외장 L3 캐시를 복원버퍼로 사용해야 하는데 이는 일반적인 싱글프로세서 시스템에서 사용하지 않는 메모리 계층이다. 그래서 HMCS의 코드 메모리에서는 CMS의 방식을 바탕으로 해서 스왑 아웃되는 페이지를 압축해 주 메모리 상의 논리적인 압축 코드 메모리에 저장하고 추후에 접근되는 경우에 복원해 CPU로 전송하는 방식을 사용한다. 이때 코드 페이지의 복원시간은 자기디스크와 같은 보조 기억장치로부터의 페이지 스왑 인 시간보다 짧다. 코드 메모리를 압축함으로써 얻는 이득이 데이타의 메모리의 그것보다 낮음에도 불구하고, 코드 메모리에 압축기법을 적용한 이유는 SPEC CINT2000 벤치마크의 경우 코드 메모리의 크기가 데이타 메모리의 크기보다 평균적으로 약 3배 가량 커 성능에 큰 영향을 줄 수 있기 때문이다.

HMCS의 데이타 캐시에서는 명령어 캐시와는 다르게 압축기법을 사용한다. 왜냐하면 데이타 캐시블록의 평균 압축률은 약 50%로 비교적 낮기 때문이다. 복원비용을 줄이기 위해서 내장 L1 캐시와 L2 캐시사이에 완전연관 구조의 복원버퍼를 사용하며, 복원버퍼는 L2 캐시블록 8-32개를 저장하며 내장 L1 캐시와 같은 속도로 접근된다. HMCS의 내장 데이타캐시 구조는 SCMS의 그것과 유사해 HMCS는 기본적으로 프로세서-메모리간 성능격차 문제를 완화한다. 그림 4는 HMCS의 압축 블록 및 페이지의 관리방식을 도식화 것이다. 주 메모리로 데이타 페이지를 로드할 때 내부 캐시블록들을 개별적으로 압축하고 고정길이 버켓을 사용해 페이지 내부에 배치하며, 압축 페이지의 크기는 내부 압축 캐시블록의 버캣크기의 총 합에 의해 결정된다. 그리고 이 압축 페이지 또한 고정길이 메모리 버켓을 사용해 주 메모리에 저장한다. 이때 가변길이 버켓 대신에 고정길이 버켓을 사용하는 이유는 메타데이타의 크기를 줄이고 효율적인 메모리 접근방식을 제공하기 위해서이다. 압축 페이지 내부의 임의의 한 캐시블록에 접근이 이뤄지면, 해당 캐시블록을 압축된 경우에는 압축된 상태로 압축되지 않은 경우에는 압축되지 않은 상태로 내장 L2 캐시로 전송해 저장하고, 복원과정을 거쳐 복원버퍼에 저장하고, 다시 내장 L1 캐시로 전송해 저장하고 CPU 코어로 해당 워드를 전송한다. HMCS는 프로세서와 주 메모리사이에 추가적인 메모리 계층을 사용하지 않기 때문에 싱글프로세서 시스템과 멀티프로세서 시스템 모두에 적합한 방식이다.

3.2 계층적 버케팅 방식

동일크기 블록을 압축해도 압축블록의 크기는 블록내부 데이타의 연관도에 따라 상이하기 때문에 압축블록을 선형주소 공간에 저장하면 내부파편 또는 외부파편이 발생해 압축에 따른 이득이 저하된다. 다양한 크기의 데이타를 선형 주소공간에 저장하는 전형적인 방법에는 가변길이 할당법과 고정길이 할당법이 있다. 가변길이 할당법은 외부파편을 발생시키고, 다양한 크기의 블록으로 인해 관리방식이 복잡해져 내장캐시와 같이 효율적인 접근방식이 필요한 시스템에는 적합하지 않다. 반대로 고정길이 할당법은 내부파편을 발생시키지만 가변길이 할당법에 비하여 훨씬 간단한 관리방식을 사용한다. 이러한 이유로 HMCS와 SCMS는 고정길이 할당법을 사용해 내장캐시 및 주 메모리에서 각각 압축블록과 압축 페이지를 관리한다.

본 논문에서는 버켓을 압축블록을 저장하는 기본 데이타구조로 정의하고 버켓 크기를 압축단위에 대한 비율로 표현한다. 이 정의에 따르면 SCMS는 내장캐시와 주 메모리에서 1/2 크기의 버켓을 사용하는 것이 된다. 세부적으로 SCMS는 내장캐시에서 압축률이 50% 이하인 블록에 대해서 1/2 크기의 캐시 버켓을 할당하고, 주 메모리에서는 페이지 내부의 모든 캐시블록이 1/2 크기의 캐시 버켓을 내장캐시에서 사용하는 경우에만 1/2 크기의 메모리 버켓을 해당 페이지에 할당한다.

표 1은 수식 (1)을 사용해 계산한 압축단위(CU)와 캐시 버켓의 크기(CBS)에 따른 내장캐시의 공간확장 비율이다. 함수 $D(CU, \lambda)$는 압축단위가 CU이고 압축률이 λ인 내장 L2 데이타 캐시블록의 압축률의 분포비율이다. 표 1에서 공간확장 비율의 증가폭이 캐시 버켓의 크기가 1/8 보다 작아지면 점차적으로 줄어들기 때문에 HMCS의 내장 L2 데이타 캐시에서는 1/8 크기의 버켓을 사용한다. 동일한 인덱스 값을 갖는 최대 8개의 압축블록을 하나의 물리 캐시블록에 저장할 수 있지만 하나의 물리 캐시블록에 동시에 5-8개의 압축블록을 저장할 확률은 매우 낮다. 따라서 HMCS 의 내장 L2 데이타 캐시에서는 하나의 물리 캐시블록에 3-5 개의 테크 필드를 내장캐시의 연관정도(associativity)에 따라 사용한다. 그리고 물리 캐시블록 내부에 저장한 압축블록들의 교체 알고리즘은 LRU 기법을 사용한다.

$$R_{\exp}(CU, CBS) = \frac{1}{\sum_{i=1}^{CBS/1} \left(iCBS \int_{(i-1)CBS}^{iCBS} D(CU,\lambda) d\lambda \right)} - 1 \quad (1)$$

HMCS의 주 메모리에서는 1/4 크기의 페이지 버켓을 사용함으로써 메타데이타의 양을 줄이고 페이지 재배치 확률을 낮춘다. SCMS에 비해 HMCS에서 페이지 재배치 확률이 줄어드는 까닭은 캐시 버켓의 크기보다 큰 메모리 버켓을 사용해 주 메모리에서 SCMS의 버켓팅 방식보다 높은 유연성을 제공하기 때문이다. 추가적으로 HMCS에 기반한 컴퓨터시스템의 운영체제는 압축 페이지를 할당하는데 있어서 우선순위에 기반한 욕심쟁이기법(greedy approach)을 사용해 페이지 재배치 확률을 항상 낮은 상태로 유지한다. 그림 4는 이상에서 구성한 계층적 버캣팅 방식을 사용해 일반 페이지를 캐시블록 단위로 압축하고 고정길이 할당법을 사용해 가장 적은 양의 내부파편을 발생시키는 캐시 버켓에 저장하고, 이의 조합인 압축페이지를 메모리 버켓에 저장하는 과정을 제시한다.

표 1 Alpha ISA 구조의 내장 L2 데이타 캐시의 Rexp

CU/CBS	1/2	1/4	1/8	1/16	1/100
032B	23.0%	47.1%	61.9%	69.8%	81.5%
064B	32.4%	60.1%	78.5%	88.2%	103.5%
128B	41.9%	71.4%	93.4%	105.2%	123.7%
256B	51.0%	83.3%	106.2%	119.1%	140.5%

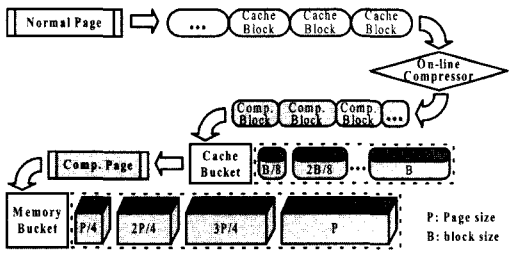


그림 4 계층적 버켓팅 방식을 사용한 압축 블록 및 압축 페이지 관리방식

3.3 압축블록 그룹화 기법

메모리 버켓의 실제크기가 1KB, 2KB, 3KB, 그리고 4KB 중에 하나이기 때문에 가상 메모리상의 페이지 번호를 물리 메모리상의 메모리 버켓번호로 변환하여야 한다. 이를 위해 기존의 가상메모리 페이지 테이블 엔트리 상에 물리 메모리 페이지번호 필드를 2비트 더 확장해 메모리 버켓 번호를 표현하며, 버켓의 크기를 표현하기 위해 2비트 크기의 크기 필드를 추가로 사용한다. 이 가상메모리 페이지와 메모리 버켓간의 주소변환은 4비트를 추가로 사용해 일반적인 구조의 TLB에서 이뤄진다. 부가적으로 HMCS에서는 물리 메모리상에 위치한 압축 페이지의 내부 캐시블록들을 압축해 저장하기 때문에 캐시블록 번호를 캐시 버켓 번호로 변환하고 각각의 캐시 버켓의 크기를 기록해야 한다. 이때 블록 주소 변환은 프로세서 내부에서 L2 캐시 하위계층에서 처리하거나 주 메모리의 컨트롤러 차원에서 처리할 수 있다.

본 논문에서는 이 캐시블록 주소변환 테이블의 크기를 줄이기 위해서 블록 그룹화 기법을 제안한다. 블록 그룹화 기법은 인접한 압축 캐시블록을 그룹으로 구성

표 2 압축률 편차 (GS: group size)

CU/GS	32B	64B	128B	256B	512B
32B	0%	2.5%	3.0%	3.3%	9.5%
64B	0%	0%	2.1%	4.5%	7.1%

하고 그룹 내부의 캐시블록 중 가장 큰 버켓 크기를 대표치로 고려해 해당 그룹 버켓 크기로 사용하는 방식이다. 표 2는 수식 (2)를 사용해 계산한 그룹내부 압축블록의 압축률 편차이다. 함수 $C_{max}(i,j)$는 i 번째 압축블록부터 j 번째 압축블록 중에서 압축률의 최대치이고, 함수 $C(x)$는 x 번째 블록의 압축률이다. 변수 S는 내장 L2 데이타 캐시블록의 크기를 고려한 압축단위로 나눈 값이며, 변수 L은 고려한 캐시블록의 개수이다. 그룹 크기가 256바이트인 경우 캐시블록의 크기가 32바이트와 64바이트이면 그룹내부의 평균 압축률 차는 약 3.3%와 4.5%로 캐시 버켓의 크기은 1/8(=12.5%)의 절반 값보다 낮다. 그룹 크기가 256바이트이고 페이지 크기가 4KB인 경우에 페이지당 블록 주소변환 테이블의 크기는 페이지 내부에 총 16개의 그룹이 있고 각각의 그룹의 버켓크기는 3비트를 사용해 크기를 표현할 수 있기 때문에 6바이트이다. 이 블록 주소변환 테이블은 가상메모리 페이지테이블 엔트리에 추가된다. 가상메모리 페이지테이블 엔트리의 기본 크기가 4바이트라고 가정하면 HMCS의 페이지테이블 엔트리의 크기는 10바이트가 되며, 만약 메모리 대역폭이 80비트 이상이면 일반 메모리 시스템과 비교해 페이지테이블 엔트리를 전송하기 위한 추가비용을 야기하지 않는다.

$$\Delta(L,S)=\frac{1}{LS}\sum_{i=1}^{L}\sum_{j=1}^{S}(C_{\max}(iS,iS+S-1)-C(iS+j)) \quad (2)$$

3.4 대칭형 내장캐시 구조

SCMS에서는 내장 L2 캐시블록의 크기를 압축단위로 사용한다. 복원시간은 압축단위의 크기에 비례하기 때문에 SCMS에서는 캐시블록의 크기가 큰 경우에 긴 복원시간으로 인해 성능이 크게 저하될 수 있다. 반면 압축단위를 작게하면 압축효율이 낮아져 압축을 통해 얻는 이득이 감소한다. 따라서 압축단위는 임계 압축률과 더불어 내장캐시 압축시스템에 있어서 중요한 설계요소이다.

SimpleScalar/AlphaISA 구조에서 측정한 32바이트 크기와 64바이트 크기의 내장 L2 데이타 캐시블록의 평균 복원시간은 각각 6.5사이클과 12.5사이클이다. 압축내장 캐시의 압축블록에 접근할 확률을 70%라고 가정하면 평균 복원시간은 약 4.6사이클과 8.8사이클로 감소한다. 다시 복원버퍼의 접근성공률을 [2]에 제시된 것과 같이 70%로 고려하면 각각 1.4사이클과 2.6사이클로 줄어든다. 추가적으로 early restart 기법을 사용하면 이 비용은 다시 각각 0.7사이클과 1.3사이클로 줄어든다.

위에서 고려한 압축단위와 같은 32 바이트와 64 바이트 크기의 SimpleScalar/AlphaISA 구조의 내장 L2 데이타 캐시블록의 평균 압축률은 각각 55%와 52.5%로 비교적 효율적이다. 따라서 HMCS에서는 복원비용과 공간확장 비율을 최적화하기 위해 32바이트 또는 64바이트를 압축단위로 사용한다. 설계에 따라 이 압축단위보다 내장 L2 데이타 캐시블록의 크기가 클 수 있다. 이런 경우에는 내장 L2 데이타 캐시블록을 압축단위로 나눠 멀티 뱅크에 저장하고, 각각의 뱅크에 대응하는 병렬압축기[10]를 사용해 작은 압축단위에 따른 압축률상의 손실을 최소화하고 복원시간을 병렬정도에 비례하게 단축시킨다.

이상에서 제시한 기법을 바탕으로 직접사상 구조로 설계한 HMCS의 내장 L2 데이타 캐시 구조는 그림 5와 같다. 해더 필드에는 각 압축블록의 저장위치를 3비트를 사용해 기록한다. 이때 해더 필드의 첫 3비트는 항상 0이다. 이 블록 다이어그램을 통해 HMCS가 비록 SCMS에 비해 복잡한 운영방식을 사용함에도 불구하고 부가적인 지연시간을 야기하지 않음을 알 수 있다.

4. 시뮬레이션 환경 및 방법

압축 메모리시스템의 특성 중의 하나는 쓰기연산을 통해 압축블록의 압축률이 변화할 수 있다는 것이다. 하지만 기존 연구들에서 사용한 트레이스기반 시뮬레이션 방법은 정적인 메모리 이미지 또는 트레이스 데이타를 사용하기 때문에 실행시간에 변화하는 압축률을 반영하지 못한다. 그리고 슈퍼스칼라구조의 마이크로프로세서에서는 압축률 변화에 따라 실행시간에 변화한 메모리 접근시간을 동적 스케줄링을 통해 효율적으로 재구성하는데 트레이스기반 시뮬레이션은 이와 같은 슈퍼스칼라 구조의 마이크프로세서의 기본적인 기능을 지원하지 못한다. 이러한 문제점을 극복하고 보다 압축 메모리시스템의 성능을 정확하게 측정하기 위해서 우리는 대표적인 슈퍼스칼라구조의 마이크로프로세서 시뮬레이터인 SimpleScalar[13,14]를 수정해 실행기반 시뮬레이션을 수행한다. 본 논문의 성능평가 과정은 압축 메모리시스템의 성능을 실행기반 시뮬레이션을 통해 측정한 최초의 실험이다.

우리는 SPEC CINT2000[11,12] 세트를 벤치마크 프로그램으로 채택하고, 이 중에서 SimpleScalar로의 포팅 용이도에 따라서 7개를 선택해 사용한다. 포팅과정을 통해 비교적 용이하게 SimpleScalar의 Alpha ISA와 PISA용 실행파일로 생성이 가능한 7가지 벤치마크의 특성은 표 3과 같다. Code와 Data 필드는 각각 가상메모리(VM)상의 위치한 코드 메모리와 데이타 메모리 중에서 1억 개의 명령어를 수행한 이후에 한번이라도 접근된

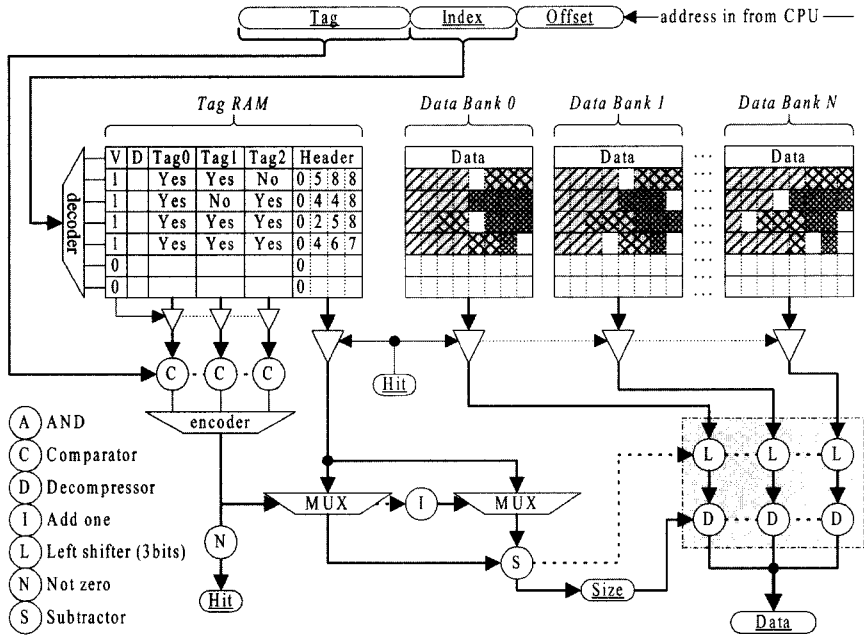


그림 5 HMCS의 내장 L2 데이타 캐시의 블록 다이어그램 (*N = block size / CU*)

표 3 선택한 SPEC CINT2000 벤치마크의 특성

ID	Name	Code (KB)	Data (KB)	VM (KB)	Code CR	Data CR	IPC	IPB	Miss Rate(%)		Inst #	LD #	ST #
									I-L1	D-L1			
1	bzip2	192	143	202,040	87%,72%	56%,37%	0.76	8.25	2.9	12.3	100 M	37 M	12 M
2	gcc	1,944	299	7,628	82%,70%	40%,33%	0.57	6.32	11.5	13.5	100 M	34 M	11 M
3	gzip	208	393	197,252	87%,71%	71%,60%	0.89	10.56	4.2	10.2	100 M	37 M	12 M
4	mcf	160	79	194,608	85%,69%	60%,29%	0.77	6.08	11.5	6.0	100 M	49 M	25 M
5	parser	312	282	9,420	86%,72%	53%,38%	0.82	5.72	4.5	12.2	100 M	35 M	9 M
6	vortex	800	116	25,660	88%,70%	51%,39%	0.24	6.28	15.5	14.9	100 M	29 M	17 M
7	vpr	368	68	3,456	88%,73%	48%,29%	0.22	8.08	9.5	19.4	100 M	22 M	5 M

캐시블록의 총 크기를 의미하며, Code CR과 Data CR 필드는 압축단위가 64바이트와 4KB인 경우에 각각 널 블록을 제외한 코드 메모리와 데이타 메모리의 평균압축률을 의미한다. 다음 두 필드들은 내장 L1 명령어 캐시와 내장 L1 데이타 캐시의 접근실패율이고, 나머지 필드들은 이때 수행한 명령어, 로드 연산, 그리고 쓰기 연산의 수이다. 시뮬레이션시에 고려한 기본 내장캐시 사양은 다음과 같다. 내장 L1 명령어 캐시와 내장 L1 데이타 캐시는 각각 4KB 크기의 직접사상 구조에 블록크기는 16B이고, 내장 L2 명령어 캐시와 내장 L2 데이타 캐시는 각각 16-128KB크기의 2-way 연관구조에 블록크기는 64B이다. 이은 압축 내장 L2 데이타 캐시의 특성을 보다 효율적으로 관찰하기 위한 사양이다. 그리고 HMCS와 SCMS의 복원버퍼의 엔트리는 16개씩이다.

리눅스와 같은 범용 운영체제의 기본 스케줄링 단위는 200ms이다. 우리는 수행시간 단축비율을 측정하기 위하여 매 실험마다 1억 개의 명령어를 수행하는데, 이것은 클럭이 1GHz이고 CPI가 2인 마이크로프로세서가 약 200ms 동안 처리하는 계산량이다. SimpleScalar는 Alpha ISA와 PISA 두 가지 명령어 셋 구조를 지원하지만, 수행시간 단축비율을 제시하는 과정에서는 Alpha ISA의 결과치만 제시한다. 왜냐하면 SimpleScalar/ AlphaISA 모델이 SimpleScalar/PISA 모델보다 실제 시스템과 유사하기 때문이다. 그리고 주 메모리의 확장비율을 제시하기 위해, 우리는 선택한 7개의 벤치마크를 프로그램을 사용해 수정된 SimpleScalar에서 10억 개의 명령어를 수행한 후 저장한 메모리 이미지를 사용한다. 이때 코드 메모리와 데이타 메모리를 분리해 성능을 측정한다.

5. 성능평가

이 장의 1절에서는 주 메모리 확장에 따른 이득을 고려하지 않은 경우의 HMCS의 수행시간 단축비율을 일반 메모리시스템(CS)과 SCMS와 비교해 제시한다. 그리고 2절에서는 HMCS의 주 메모리 확장비율을 제시한다.

5.1 주 메모리 확장에 따른 이득을 고려하지 않은 경우의 수행시간 단축비율

그림 6은 CS와 비교한 내장 L2 데이타 캐시의 상대적 접근실패율이다. SCMS의 내장 L2 데이타 캐시는 압축효율이 좋은 경우에 같은 크기의 두 배의 연관도를 갖는 CS-DA보다 더 높은 논리적 연관도를 갖기 때문에 SCMS의 접근실패율은 CS-DA의 그것보다 항상 낮다. HMCS는 CS와 SCMS와 비교해 내장 L2 캐시의 접근실패율을 각각 최대 66%와 25% 감소시킨다. CS와 SCMS와 비교한 HMCS의 내장 L2 캐시의 접근실패율 감소비율의 평균치는 각각 17%와 8%이다. 이 실험에서는 내장 L2 데이타 캐시의 크기를 128KB로 설정했다. HMCS의 내장 L2 데이타 캐시의 접근실패율의 감소비율은 보다 작은 크기의 캐시를 사용하면 캐시에 걸리는 부하가 많아져 훨씬 크게 나타난다.

그림 7은 메모리 연산의 종류에 따른 메모리 트레픽의 양을 CS와 비교한 것이다. 지연시간(latency)은 첫 번째 워드를 전송하는데까지 걸린 시간을 의미한다. HMCS는 벤치마크 *bzip, gzip*, 그리고 *mcf*의 경우에 내장 L2 데이타 캐시의 접근실패율을 CS와 비교해 거의 단축시키지 못해 이 벤치마크에 경우 읽기와 쓰기 지연시간은 CS의 그것과 거의 동일하다. 하지만 HMCS는 벤치마크 *gcc, parser, vortex*, 그리고 *vpr*에 대한 읽기와 쓰기 지연시간을 CS의 그것과 비교해 각각 약 63%, 7%, 21%, 그리고 16% 단축한다. 전송시간은 전체 캐시블록을 전송하는데 걸리는 시간에서 지연시간을 뺀 값으로, 이 실험에서는 메모리대역폭을 8바이트 내장 L2 데이타 캐시블록의 크기를 64바이트로 설정해, 압축블록 중 압축률이 1/8 이하인 경우에는 하나의 메모리 버스 사이클에 전송할 수 있어 이 경우 전송시간이 0이다. CS 및 SCMS와 비교한 HMCS의 읽기와 쓰기 전송시간의 단축비율은 각각 36-99%와 22-99%이다.

그림 8은 CS와 비교한 CS-DA, SCMS, 그리고 HMCS의 수행시간 단축비율이다. 이는 HMCS-32(압축단위: 32바이트)가 CS 및 SCMS와 비교해 수행시간을 각각 최대 4%와 2.5%가량 단축함을 보인다. 이 실험에서 사용한 내장 L2 데이타 캐시의 크기는 128KB이고 총 내장 L2 캐시의 크기는 256KB로 일반적인 단일 프로세서 시스템에서는 비교적 큰 크기이다. 내장 L2 데이타 캐시의 크기가 64KB, 32KB, 그리고 16KB인 경우

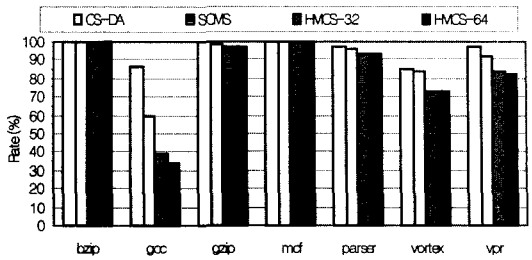


그림 6 CS와 비교한 내장 L2 데이타 캐시의 상대적 접근실패율

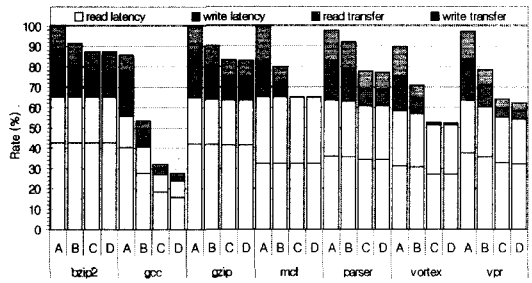


그림 7 CS와 비교한 상대적 메모리 트레픽(A: CS-DA, B: SCMS, C: HMCS-64, D: HMCS-32)

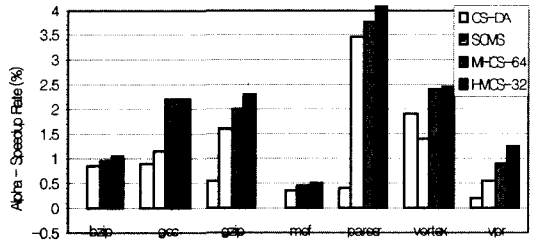


그림 8 주 메모리 확장에 따른 이득을 고려하지 않은 경우의 CS와 비교한 수행시간 단축비율

에 CS와 비교한 HMCS의 수행시간 단축비율은 각각 최대 7%, 13%, 그리고 23%이다.

HMCS에서는 압축률이 7/8 이하인 블록에 대해 모두 압축을 적용하고, SCMS에서 제안된 0.5를 압축 경계로 하는 선택적 압축기법을 사용하지 않지만, 복원비용에 의해 성능이 크게 저하되지 않는 이유는 제안한 대칭구조의 내장캐시가 복원시간을 효율적이고 단축시키기 때문이다. HMCS는 성능을 크게 저하시키지 않으면서 보다 많은 블록에 압축을 적용하고 내부파편을 줄여 주 메모리를 보다 많이 확장할 수 있는 요인을 지닌다.

5.2 주 메모리 확장비율

표 4는 압축방식에 따른 주 메모리의 데이타 영역과 코드 영역의 확장비율이다. 이때 이상적인(ideal) 시스템

표 4 데이타 메모리와 코드 메모리의 확장 비율
(CB: cache bucket size, MB: memory bucket size, CU: 압축단위, AISA: Alpha ISA)

Area	System	CB	MB	CU	AISA	PISA
Data	Ideal	1/∞	1/∞	32B	228%	88%
				64B	257%	99%
	SCMS	1/2	1/2	32B	52%	31%
				64B	57%	32%
	HMCS	1/8	1/4	32B	111%	54%
				64B	120%	57%
Code	Ideal	X	1/∞	4KB	43%	86%
	HMCS	X	1/4	4KB	27%	36%

은 내부파편과 외부파편을 발생시키지 않는 시스템으로 캐시와 메모리의 버켓크기는 1/∞인 경우이다. 이상적인 시스템의 공간 확장비율은 압축률의 역수에서 1을 뺀 값과 같다. HMCS는 압축단위가 64바이트인 경우 데이타 메모리를 SCMS보다 2배 가량 확장하는데 이는 계층적 버켓팅 방식을 사용해 내장캐시와 주 메모리의 내부파편을 효율적으로 줄이고 임계 압축률을 7/8(=87.5%)로 상향 설정했기 때문이다.

HMCS의 데이타 메모리 확장비율은 벤치마크 프로그램에 따라 57-120%이며, SCMS가 확장하지 않던 코드 메모리의 확장비율은 벤치마크 프로그램에 따라 27-36%이다. HMCS는 주 메모리 공간을 효율적으로 확장하기 때문에 실제 HMCS의 수행시간 단축비율을 이장의 1절에서 고려한 것보다 훨씬 크게 나타날 것으로 고려한다.

6. 결 론

본 논문에서는 프로세서-메모리간 성능격차 문제를 완화하고 주 메모리의 공간을 효율적으로 확장하는 이종 메모리 압축시스템을 네 가지 새로운 기법을 바탕으로 하여 설계하였다. 첫째, 메모리시스템을 코드 메모리와 데이타 메모리로 구분하고 각각의 압축특성에 맞게 독립적으로 설계함으로써 전체 메모리시스템의 성능향상을 극대화하였다. 둘째, 메모리 계층구조를 고려해 다양한 크기의 버켓을 내장캐시와 주 메모리에서 사용해 압축블록을 관리함으로써 내부파편을 줄이고 보다 많은 블록에 압축을 적용해 궁극적으로 내장캐시의 공간과 메모리 대역폭 그리고 주 메모리 공간의 확장을 최적화하였다. 셋째, 압축블록을 그룹화하여 대표치를 메타데이타로 사용함으로써 압축페이지에 대한 메타데이타의 양을 줄이고 이를 통해 보다 효율적인 접근방식을 제공하도록 설계하였다. 넷째, 압축 내장캐시에서 데이타 램을 분할해 대칭구조로 구성하고 병렬 복원기 사용함으로써 압축률 상의 손실을 최소화하며 복원시간을 크게 단축시키도록 설계하였다.

성능평가 과정에서는 기존의 트레이스기반 시뮬레이션보다 높은 정확성을 갖는 실행기반 시뮬레이터를 개발해 사용하였다. 시뮬레이션 결과 제안하는 이종 메모리 압축시스템은 주 메모리 확장에 따른 성능향상을 고려하지 않은 경우에도 내장 L2 데이타캐시의 크기에 따라 수행시간을 내장캐시의 크기에 따라 4-23%가량 단축시킨다. SPEC INT2000 벤치마크를 사용한 제안하는 이종 메모리 압축시스템의 주 메모리의 데이타 영역과 코드 영역의 확장비율은 각각 57-120%와 27-36%이다. 이를 통해 본 논문에서 제시한 기법들을 바탕으로 하드웨어 압축기법을 내장캐시와 주 메모리에 적용함으로써 프로세서-메모리간 성능격차 문제를 완화하고 주 메모리 공간을 효율적으로 확장해 입출력 병목현상도 크게 개선함을 알 수 있다.

참 고 문 헌

[1] J.L. Hennessy and D.A. Patterson, Computer Architecture - *A Quantitative Approach,* 3rd Edition, Chapter 5, Morgan Kaufmann Publishers, 2002.

[2] J.S. Lee, W.K. Hong, and S.D. Kim. "Design and Evaluation of On-Chip Cache Compression Technology," *In Proceedings of the 17th IEEE International Conference on Computer Design,* pp. 184~191, 1999.

[3] S. Bunton and G. Borriello, "Practical Dictionary Management for Hardware Data Compression," *Communications of the ACM,* Vol. 35, No. 1, pp. 95~104, 1992.

[4] M. Kjelso, M. Gooch, and S. Jones, "Design and Perform-ance of a Main Memory Hardware Data Compressor," *In Proceedings the 22nd Euromicro Conference, IEEE Computer Society Press,* pp. 422~430, 1996.

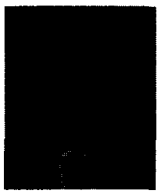
[5] B. Abali, Shen Xiaowei, H. Franke, D.E. Poff, and T.B. Smith, "Hardware Compressed Main Memory: Operating System Support and Performance

Evaluation," *IEEE Transactions on Computers,* Vol. 50, Issue 11, pp. 1219~1233, 2001.
[6] M. Kjelso, M. Gooch, and S. Jones, "Performance Evaluation of Computer Architecture with Main Memory Data Compression," *Journal of Systems Architecture,* Vol. 45, pp. 571~590, 1999.
[7] F. Douglis, "The Compression Cache: Using On-line Compression to Extend Physical Memory," *In Proceedings of the Winter 1993 USENIX Conference,* pp. 519~529, 1993.
[8] D. Citron and L. Rudolph, "Creating a Wide Bus Using Caching Techniques," *In Proceedings of the first International Symposium on High-Performance Computer Architecture,* pp. 90~99, 1995.
[9] J. Yang, Y. Zhang, and R. Gupta, "Frequent Value Compression in Data Caches," *In Proceedings of the 33rd ACM/IEEE International Symposium on Microarchitecture,* pp. 258~265, 2000.
[10] P.A. Franaszek, J. Robinson, and J. Thomas, "Parallel Compression with Cooperative Dictionary Construction," *In Proceedings of the 6th IEEE Data Compression Conference,* pp. 200~209, 1996.
[11] J.L. Henning, "SPEC CPU2000: Measuring CPU Performance in the New Millennium," *IEEE Computer,* Vol. 33, Issue 7, pp. 28~35, 2000.
[12] SPEC CPU2000, http://www.spec.org/cpu2000/.
[13] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: an Infrastructure for Computer System Modeling," *IEEE Computer,* Vol. 35, Issue 2, pp. 59~67, 2002.
[14] SimpleScalar, http://www.simplescalar.com/.
[15] 이장수, 홍원기, 김신덕, "고성능 컴퓨팅을 위한 압축 메모리 구조 설계", *정보과학회 논문지, 시스템 및 이론,* 26권, 2호, pp. 242~260, 1999.

임 근 수
2003년 연세대학교 컴퓨터과학 학사 및 전기전자공학 학사. 현재 서울대학교 전기컴퓨터공학부 석사과정. 관심분야는 고성능 메모리 구조, 플래시메모리 저장장치, 센서 네트워크, 네트워크 스택 구조, 고급 운영체제 등임

이 장 수
1994년 한양대학교 전자계산학 학사 1998년 연세대학교 컴퓨터과학 석사 2004년 연세대학교 컴퓨터과학과 박사 2001년 - 현재 IBM Poughkeepsie 연구원. 관심분야는 고성능 컴퓨터 구조, 프로세서-메모리 집적구조, 성능평가 등임

홍 인 표
1999년 연세대학교 전자공학 학사. 2001년 연세대학교 전기전자공학 석사. 현재 연세대학교 전기전자공학부 박사과정. 주 관심분야는 마이크로프로세서와 네트워크프로세서 등임



김 지 홍
1986년 서울대학교 계산통계학과 학사 1988년 Univ. of Washington 컴퓨터과학과 석사. 1995년 Univ. of Washington 컴퓨터과학 및 공학과 박사. 현재 서울대학교 컴퓨터공학부 부교수. 관심분야는 컴퓨터구조, 내장형 시스템, 저전력 시스템, 멀티미디어 시스템임

김 신 덕
1981년 연세대학교 전자공학 학사. 1987년 Univ. of Oklahoma 전기컴퓨터공학 석사. 1991년 Purdue Univ. 전기컴퓨터공학 박사. 현재 연세대학교 컴퓨터과학과 교수. 관심분야는 고성능 컴퓨터 구조, 지능형 캐시 메모리, 병렬처리 시스템, Grid computing임

이 용 석
1973년 연세대학교 전기공학 학사. 1977년 Univ. of Michigan 전기공학 석사 1981년 Univ. of Michigan 전기공학 박사. 현재 연세대학교 전기전자공학부 교수. 주 관심분야는 마이크로프로세서 설계임

고 건
1974년 서울대학교 응용물리학 학사 1979년 Univ. of Virginia 전산학 석사 1981년 Univ. of Virginia 전산학 박사 현재 서울대학교 컴퓨터공학부 교수 관심분야는 운영체제, 컴퓨터 구조, 컴퓨터 시스템 성능평가, 분산 컴퓨터시스템임