임베디드 시스템을 위한 메모리 서브시스템 파라미터의 자동 검출 기법*

하태준○ 서상민 전보성 이재진
티맥스소프트○, 서울대학교 컴퓨터공학부
taejun_ha@tmax.co.kr○, {sangmin, posung, jlee}@aces.snu.ac.kr

Automatic Detection of Memory Subsystem Parameters for Embedded Systems

Taejun Ha○ Sangmin Seo Posung Chun Jaejin Lee
TmaxSoft○, School of Computer Science and Engineering, Seoul National University

1. 서 론

컴퓨터 시스템은 각각의 용도에 따라 다양한 하드웨어를 사용하여 그 성능을 향상시켜 나가고 있다. 그리고 소프트웨어 역시 이러한 다양한 하드웨어의 환경에 적합한 프로그래밍을 통해 시스템의 성능을 향상시켜 나가고 있다. 이런 환경 아래에서 하드웨어 파라미터(parameter)를 정확히 아는 것은 무엇보다도 중요한 요소이다. 특히 성능에 가장 큰 영향을 미치는 메모리 서브시스템의 파라미터를 정확히 알 수 있다면 그에 따라 소프트웨어를 작성하거나 수정하여 메모리 접근을 여러 뱅크에 동등하게 분배하여 수행속도를 향상시킬 수 있을 것이다. 예를 들어, 셀프 최적화(self-optimization)는 프로그램이 실행되는 하드웨어의 특성에 맞게 소프트웨어를 변경해서 성능을 극대화한다. 이때 하드웨어 파라미터를 사용할 수 있다.

본 논문에서는 기본적인 하드웨어 파라미터를 구할 수 있는 알고리즘을 임베디드 시스템에 적용하여 결과를 확인하고, 프로그램의 성능에 크게 영향을 미치는 메모리 뱅크(bank)의 개수를 찾아낼 수 있는 알고리즘을 제안한다.

2. 본 론

하드웨어 파라미터를 검출하는 가장 기본적인 방법은 벤치마크 프로그램을 실행시켜 나타난 결과를 분석해서 값을 추측하는 것이다[1]. 벤치마크 프로그램은 메모리의 특정 주소에 값을 쓰거나 읽어들이고 이때 걸리는 시간을 측정한다. 추가로 캐시 미스(miss), TLB 미스와 같은 값도 같이 측정한다. 이렇게 얻어진 결과를 분석하여 하드웨어 파라미터를 결정한다[2].

캐시의 파라미터를 측정하는 방법에 대해 살펴보면, 캐시 크기의 경우, 적당한 크기의 배열을 할당받아 초기화한 후 배열을 처음부터 차례대로 접근하면서 캐시 미스를 측정한다. 배열의 크기를 점차 증가시키다가 캐시 미스가 처음으로 발생하는 배열의 크기가 캐시의 크기가 된다. 캐시 라인을 측정하려면, 캐시는 메모리에서 데이터를 가져올 때 캐시 라인 단위로 데이터를 가져오기 때문에 특정 캐시 라인의 첫 번째 액세스는 캐시 미스이고 라인의 나머지 액세스는 히트가 된다. 따라서 초기화를 하지 않고, 간격을 1로 한 뒤 캐시보다 작은 크기의 배열을 액세스하면서 캐시 미스를 측정한다. 이때 두 개의 미스 사이에 몇 개의 히트가 발생하였는지를 측정하면 라인 크기를 정할 수 있다. TLB도 캐시와 같은 알고리즘을 이용하여 측정할 수 있다.

메모리 뱅크는 메모리 모듈에서 고속 동작을 구현하려고 독립적으로 동작하는 메모리 그룹(group)을

* 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업[2006-S-040-01, Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술개발]과 한국학술진흥재단의 BK21 사업의 일환으로 수행하였습니다. 또한 이 연구를 위해 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사드립니다.

말한다. 하나의 뱅크에서 데이터를 읽거나 쓰는 동안 다른 뱅크에서 프리차지(precharge)나 리프레시(refresh) 동작을 수행하거나, 새로운 행 주소(row address)를 이용해서 워드 라인(word line)을 선택할 수 있다. 즉, 한 뱅크의 감지 증폭기(sense amplifier)로부터 데이터를 읽는 도중에 다른 뱅크의 워드 라인을 선택할 수 있다. 메모리가 여러 개의 뱅크를 가진 경우, 뱅크 인터리빙 (interleaving)을 사용하면 성능을 향상시킬 수 있다. 뱅크 인터리빙은 각 메모리 뱅크를 파이프라인처럼 이용하는 것이다. 뱅크 인터리빙을 사용하는 경우와 그렇지 않았을 때 메모리의 동작이 확연하게 틀리므로, 뱅크를 검출하려면 인터리빙하는 경우와 하지 않는 두 가지 경우에 대해서 고려하여야 한다. 메모리 뱅크의 하드웨어 파라미터와 위에서 설명한 캐시, TLB의 경우 벤치마크 프로그램의 수행시간을 통해서 측정하는 것이 가장 간단한 방법이다.

우선 뱅크 인터리빙하지 않는 경우에 대해 살펴보면, n 개의 뱅크를 가진 메모리에 최적화된 성능을 보일 수 있는 액세스 패턴을 구한다. 이후 뱅크 번호 k의 값을 1씩 증가시키며 실험결과를 측정하고 그 값을 분석함으로써 뱅크의 개수를 찾아낸다. 하지만, 일반적일 때 뱅크를 사용하면서 인터리빙을 사용하지 않으면 실제로 뱅크를 사용해도 성능의 향상을 기대할 수 없다.

뱅크 인터리빙을 사용하는 경우는 하나의 워드 단위로 뱅크가 배치되어 있다. 따라서 연속된 메모리를 액세스할 경우 가장 최적의 성능이 나오게 된다. 인터리빙을 사용하는 때도 TLB와 캐시, 프리패치의 영향을 최소화하며, 각 뱅크에 최적화된 알고리즘을 적용하면서 뱅크의 개수를 찾는다.

1) 전체 메모리의 뱅크가 1개라고 가정하면 모든 영역에 대해서 같은 수행시간을 가진다.

2) 뱅크 k 개에 최적화된 알고리즘은 k 개의 원소를 차례대로 액세스하고 k+1에서 캐시 라인의 처음으로 돌아와서 액세스한다. 이렇게 되면 k 개의 뱅크를 차례대로 사용하게 되어서 가장 빠른 테스트가 진행되게 된다. 인터리빙을 사용하지 않을 때는 예상 뱅크의 개수가 실제 뱅크의 개수보다 커지면 속도가 느려지는 현상이 발생하였는데 비해, 인터리빙을 사용하면 속도가 같게 나온다. 따라서 수행 시간이 같아지는 지점이 뱅크의 개수를 나타낸다.

위의 알고리즘을 사용해서 Hybus, XHyper270A 임베디드 시스템과 그 외 3가지 시스템을 대상으로 실험하였다. 그 결과 캐시 크기, 캐시 라인 크기, TLB 페이지 크기, 메모리 뱅크는 실제 시스템의 값과 같은 결과가 나왔다. 하지만, TLB entry의 경우 3가지 시스템에서 실제보다 작은 결과를 보였는데, 이것은 리눅스에서 특정 페이지를 피닝(pinning)하기 때문이다.

3. 결 론

본 논문에서 다룬 주제는 크게 두 가지이다. 하나는 기존의 메모리 서브시스템 파라미터 검출 알고리즘이 임베디드 보드에서 정확히 동작하는지 확인하는 것이고, 다른 하나는 메모리 뱅크의 개수를 검출할 수 있는 알고리즘을 제안하고 검증하는 것이다. 캐시, TLB와 같은 기본적인 메모리 서브시스템의 파라미터는 운영체제가 실행 중인 임베디드 보드에서 만족할 만한 결과를 도출할 수 있었고 뱅크의 개수도 정확히 알아낼 수 있었다.

이후의 연구는 하드웨어 파라미터 검출 프로그램을 OS의 커널 프로그램으로 개발하는 것이다. 그리고 하드웨어 파라미터를 검출할 때 퍼포먼스 카운터를 함께 사용하는 방법도 고려하고 있다. 논문의 실험 환경인 Intel PXA270 CPU의 경우 16가지 이벤트를 모니터링 할 수 있는 카운터를 제공하는데, 이를 활용한다면 좀 더 빠르고 정확한 결과를 얻을 수 있을 것이다. 또 다양한 데스크톱 컴퓨팅 환경에서 메모리 뱅크의 개수를 검출할 수 있는 알고리즘도 개발하고자 한다.

참고문헌

[1] Rafael H. Saavedra-Barrera, "CPU Performance Evaluation and Execution Time Prediction Using Narrow Spectrum Benchmarking," Ph.D. Thesis, UC Berkeley, Tech. Rept. No. UCB/CSD-92-684, Feb. 1992.

[2] Kamen Yotov, Keshav Pingali, and Paul Stodghill, "Automatic Measurement of Memory Hierarchy Parameters," SIGMETRICS'05, pp.181-192, June 2005.