M-RPT : 데이터의 주소 간격을 이용한 적극적인 캐시 선인출 방법

전영숙°, 문현주, 전중남, 김석일
충북대학교 컴퓨터 과학과
yschon@hanmail.net

An Eager Cache Prefetching Scheme Using Stride between Successive Data Reference

Young-Suk Chon, Hyun-Ju Moon, Joongnam Jeon and Sukil Kim

요 약

멀티미디어 응용 프로그램은 방대한 양의 데이터를 실시간으로 고속 처리해야 한다. 적재/저장과 같은 메모리 참조 명령어는 프로세서의 고속 수행에 방해가 되는 주요인이다. 본 논문에서는 메모리 참조 속도를 향상시키기 위해 멀티미디어 데이터의 주소간격이 규칙적으로 참조되는 특성을 활용하여 다음에 참조될 데이터를 미리 캐시로 선인출 함으로써 실행시 캐시 미스율을 줄이고 또한 전체 수행시간을 줄이는 효과적인 방법을 제안한다.

제안한 방법은 캐시 미스율을 줄이는 방법으로서 데이터 선인출 기법을 사용하는데 주소간격을 이용한 기존 연구들에 비해 캐시 미스율에 있어서 평균적으로 27% 향상되었다.

1. 서론

멀티미디어 데이터를 처리하는 응용프로그램은 다량의 재사용성이 적은 메모리를 참조하는 특성이 있다. 메모리 참조 지연시간을 줄이는 것은 전체 수행시간을 줄이는 방법으로서 많은 연구가 진행 중에 있으며, 메모리 참조 속도를 향상시키는 방법으로 캐시 구조가 있다.

캐시 구조에서 선인출 기법은 처리와 메모리 데이터 액세스를 중첩하여 실행함으로써, 데이터가 실제로 필요하기 전에 주기억장치에서 캐시로 인출하여 캐시 미스로 인한 지연시간을 줄이는 방법이다. 최근의 선인출 기법 중에는 멀티미디어 데이터의 주소간격이 일정하게 참조된다는 것을 이용하여 참조 패턴이 안정 상태일 때 연속한 다음위치의 데이터를 선인출을 하게 되며, 이로써 캐시 미스율을 줄이게 되는데, 이러한 방법을 규칙 선인출 기법이라 하며, 데이터가 1차원의 선형 참조인 경우에는 정확하게 선인출 주소를 계산하여 해당 데이터를 캐시로 선인출 하지만 2차원 이상에서는 정확하게 계산하지 못한다는 한계가 있다.

본 논문에서는 규칙 선인출 기법이 갖는 RPT 방식에서 상태 전이 알고리즘을 간단하게 변경하여 선인출을 하게 되며, 멀티미디어 데이터가 갖는 규칙적인 주소 간격 사이에 하나의 불규칙한 패턴 또는 일정한 개수의 규칙 패턴을 갖는 경우에도 선인출 할 주소를 적절하게 계산하게 된다.

실험은 멀티미디어 데이터의 주소를 추적하기위해 ATOM 시뮬레이터를 사용하였고, 캐시의 분석 결과를 생성하기 위해 dinero III 캐시 시뮬레이터를 변형하여 시뮬레이션을 수행하였다.

본 논문은 다음과 같다. 2장에서는 관련 연구로써, 기존 선인출 기법에 관한 연구를 분석한다. 3장에서는 제안한 선인출 알고리즘에 대해서 자세히 제시한다. 4장에서는 실험을 통하여 성능을 평가하고, 기존 알고리즘과 성능 비교한다. 마지막으로 5장에서는 본 논문에서 얻은 결과를 정리한다.

2. 관련 연구

데이터-주소 기반 선인출 방식에는 OBL 방식과 스트림 버퍼 방식이 있는데 OBL방식은 i블록이 캐시에서 미스가 나면 i 블록을 메모리에서 참조할 때 연속한 다음 위치의 블록을 선인출한다. 이 기법은 캐시 미스율을 반으로 줄일 수 있는 장점이 있는 반면, 적절하지 못한 선인출은 유용한 캐시 블록들을 캐시로부터 교체하게 되어 캐시가 오염 될 수 있다. 이러한 문제를 해결하기 위해 선인출 블록을 캐시로 적재하기 전에 큐 형태의 스트림 버퍼에 저장한다. 스트림 버퍼 구조는 캐시오염을 방지하기 위하여 선인출 블록들을 캐시로 적재하기 전에 메모리와 캐시사이의 FIFO 형태의 스트림 버퍼에 저장한다. 스트림 버퍼 구조는 캐시 오염문제는 해결하였고, 연속된 블록을 빠른 속도로 선인출하여 캐시 미스율을 줄일 수 있었다. 그러나 연속된 메모리 블록을 참조하지 않은 경우는 버퍼로 적재하고, 제거하는 일을 반복해야 한다. 이러한 문제를 해결하기위해 여러 개의 스트림 버퍼를 사용하는 다중 스트림 버퍼 기법이 있다.

PC-기반 구조에는 RPT 방식이 있는데, chen과 Baer에 의해 제안되었다. 이 구조는 적재/저장과 같은 메모리 참조 명령어의 동작 정보를 RPT에 저장하여 이전 정보를 근거로 선인출에 대한 주소를 계산하고, 선인출 명령을 발생시킨다. RPT 구조는 그림 1 [1]과 같이 태그 필드, 이전_주소 필드 필드, 주소간격 필드, 상태 필드로 되어 있다.

- 태그 필드 : 적재/저장 명령어의 주소, PC의 값
- 이전 주소 필드: PC의 명령어를 수행 시 이전 반복문에서 참조한 피연산자의 주소
- 주소간격 필드: 이전 피연산자의 주소와 현재 참조한 피연산자의 주소 간격
- 상태 필드 : 두 비트로 인코딩, init 상태 , steady 상태, transient 상태, no-pred 상태

RPT 상태에서 선인출할 메모리 주소(prefetching address)는 이전 주소(prev_addr)와 주소간격(stride)을 더하여 계산되고, 명령어가 처음 수행되는 경우는 선인출 주소는 계산하되, 선인출 명령은 발생하지 않는다. 선인출 명령은 RPT에서 상태가 안정 상태이고 주소간격이 0이 아니면 발생시킨다. 만약 이전 반복문에서 계산된 선인출 주소와 현재 참조한 메모리 주소가 같을 경우 예측 성공(correct), 그렇지 않은 경우 예측 실패(incorrect)가 된다. 상태 필드와 주소 간격 필드의 갱신은

그림 2 [1]와 같다.

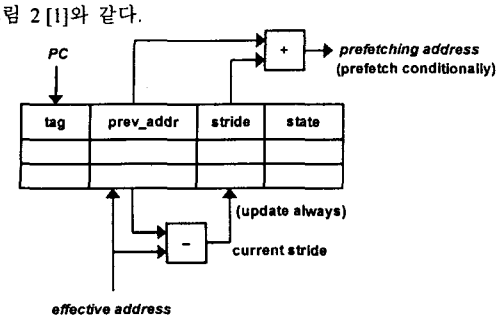

그림 1. RPT 구조

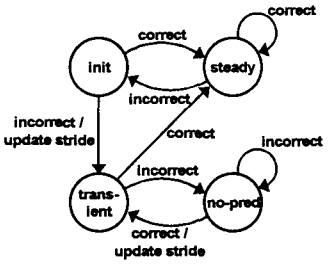

그림 2. RPT 구조의 상태도

RPT 구조는 1차원 선형적 규칙성을 갖는 경우 예를 들어 멀티미디어 응용 프로그램의 많은 양의 배열 처리가 반복적으로 처리되는 경우에 적합하다고 하겠다. 그러나 이 구조는 2차원 이상에서는 한계가 있다.

3. 적극적인 선인출 알고리즘

데이터를 선인출하는 기존의 여러 방법들 중에 RPT 구조는 메모리 참조 패턴이 1차원 규칙성을 갖는 경우에 적합하지만 이런 경우가 아닌 다소 불규칙한 경우에는 한계가 있다. 또한 방대한 양의 멀티미디어 데이터는 자원의 활용성을 높이기 위해 서브 블록으로 참조되는데 이런 경우 1차원 규칙성이 현저히 떨어지므로 선인출 효과가 크게 저하된다. 그림 3은 서브블록 크기를 8x8로 분할하여 연산을 수행하는 경우로 정확한 주소 계산으로 선인출 되는 데이터 영역에 대한 것이다. 진하게 색이 채워진 영역이 선인출 명령에 의해 요청된 데이터가 캐시로 적재되어 캐시에서 히트가 된 경우이다.

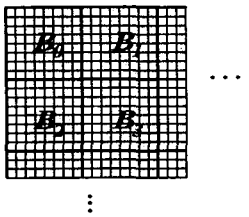

그림 3 RPT구조의 서브블록 참조

새로이 제안한 M-RPT 구조에서는 메모리 참조 패턴이 대다수의 선형적인 규칙성 가운데 일부의 불규칙한 데이터 참조를 갖는 경우 또는 일정한 주기로 반복되는 비선형적 규칙 패턴인 경우 기존의 RPT 구조에서 잘못된 주소로 인하여 캐시 미스가 발생되었던 것을 현저히 줄일 수 있다. M-RPT 방식에서는 선인출 주소를 계산할 때에, 이전 주소간격(prev_stride)과 현재 계산한 주소간격(current stride)이 동일한 경우 선인출할 주소를 현재 계산한 주소간격으로 변경하여 계산을 하고, 주소간격이 동일하지 않은 경우는 저장된 선인출 주소간격(prefet_stride)으로 계산을 하여 선인출 주소를 계산한다. 또한 선인출 명령을 발생시키는 조건도 RPT 구조와 다른데, 주소 간격이 0이 아닌 경우는 선인출 명령을 매번 발생시키고, PREFET 상태에서 주소간격이 0인 경우는 같은 블록을 참조하는 경우로서 INIT 상태가 되면서 선인출 명령을 발생시키지 않는다. 그리고 선인출할 주소는 메모리로부터 캐시로 데이터를 전송시 블록 단위로 전송하므로 블록 주소를 이용하여 구현하였다.

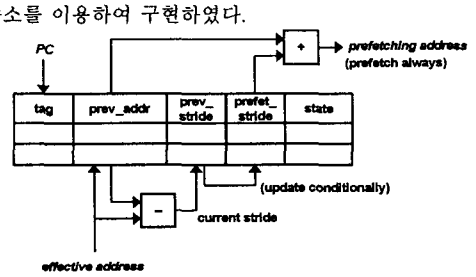

그림 4. M-RPT(modified Reference Prediction Table)

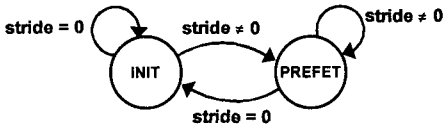

그림 5. 상태 전이도

그림 4는 RPT구조에서 prev_addr필드에 참조되는 데이터 주소가 그대로 저장되는 것과 다르게 블록 주소가 저장되며, stride 필드를 단순히 이전 주소 간격을 저장해 놓기 위한 prev_stride와 실제 선인출할 때에 선인출 주소를 계산하기 위한 prefet_stride로 분리하였다. 상태 필드 부분도 RPT 구조의 4개의 상태에서 INIT, PREFET 2개의 상태로 바뀌었다. 그림 5는 2개의 상태 전이를 나타낸 것이다.

- INIT 상태 : PREFET 상태에서 같은 블록을 참조하는 경우에 INIT 상태로 전이되며 주소간격이 0이 아닐 때에는 다시 PREFET 상태로 전이된다.
- PREFET 상태 : 주소간격이 0이 아닌 경우에는 선인출 주소를 계산하는데 이전 주소 간격과 현재 계산한 주소간격이 같은 경우에는 prefet_stride 값을 갱신한 후 이 값을 이용하여 선인출 주소를 계산하고, 다른 경우에는 저장되어 있는 prefet_stride값을 사용하여 선인출 주소를 계산한다. 주소 간격이 0인 경우는 같은 블록을 참조하는 경우로 해당 블록은 이미 캐시에 적재되어 있으므로 선인출 명령을 발생시키지 않으며 INIT 상태로 전이한다

PREFET 상태에서 선인출 명령을 발생시 선인출 데이터 주소 계산은 다음과 같이 이루어진다.

```
if(prev_stride == current_stride || prev_stride == INIT_VALUE)
        prefet_stride update
    prev_stride update
```

위와 같이 prefet_stride는 조건이 만족하는 동안에 갱신이 되는데, 만약 2번 연속해서 주소간격이 다를 때는 이전

prefet_stride값이 변경 되고, 그렇지 않고 주소간격이 계속해서 같거나, 1번 다른 경우에는 값이 변경되지 않고, 이전 값을 그대로 가지고 선인출 주소를 계산한다

그림 6은 제안한 M-RPT 구조에서 선인출 명령어 발생에 의해 정확하게 캐시에서 히트되는 데이터 영역을 나타낸 것이다.

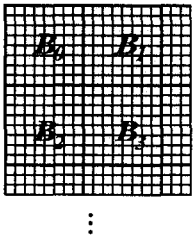

그림 6. M-RPT구조에서의 서브블록 참조

그림과 같이 RPT구조와 비교하여 캐시 미스수가 줄어든 것을 볼 수 있다.

4. 성능 분석

멀티미디어 데이터의 높은 지역성과 규칙적인 참조성을 바탕으로 데이터 선인출의 효과를 분석하기 위해 DEC system의 ATOM 시뮬레이터를 이용하여 트레이스 구동 시뮬레이션을 하였다. 수행결과는 메모리 참조 명령어 적재/저장가 동작할 때 필요한 피연산자의 주소 리스트를 출력한다. 이 결과를 위스콘신 대학에서 개발된 Dinero III를 변형한 캐시 시뮬레이터에 입력으로 하여 캐시의 동작을 분석한다. 표1에 분석 모델을 위한 매개 변수들과 값을 나타내었다. 실험은 대표적인 멀티미디어 벤치마크 MPEG(mpeg2enc, mpeg2dec), JPEG(cjpeg, djpeg)을 대상으로 하였다

표 1 분석 모델을 위한 매개변수의 값

parameter	description
I-cache size	1k
D-cache size	16k ~ 128k
block size	4byte ~ 32byte
word size	4byte
associativity	directed-mapped cache
bus_width	block size
replacement_policy	LRU policy

그림 7은 4개의 벤치마크 프로그램중 mpeg2enc를 데이터 캐시 크기가 64Kbyte이고, 같은 캐시 크기에 블록 크기를 변화시키면서 캐시 미스율을 나타낸 것이다. 그림 8은 mpeg2enc의 전체 버스 사용량을 나타낸 것인데, 그림과 같이 버스 사용량의 차이가 거의 나지 않음을 알 수 있다. 이것은 RPT 구조는 참조 데이터의 주소를 블록 주소가 아닌 해당 주소 그대로 사용하므로 같은 블록을 액세스할 경우 조건이 안정 상태이고 주소간격이 0이 아닌 경우가 만족되면 매번 선인출 명령을 발생시킨다. 이에 반해 M-RPT 구조는 블록 주소를 사용하여 주소간격이 0이 아닌 경우는 선인출 명령을 발생시키고, 0인 경우에는 같은 블록을 참조하는 경우로 선인출 명령을 발생시키지 않으므로 크게 차이가 나지 않는다.

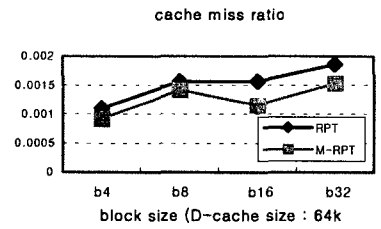

그림 7 mpeg2enc 캐시 미스율

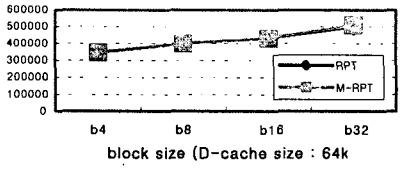

그림 8 mpeg2enc 버스 사용량

앞에서 RPT구조와 M-RPT 구조를 비교해본 결과 캐시 미스율에 있어서는 평균 27% 성능이 향상되었고, 버스 사용량(bus usage) 면에서도 크게 차이가 나지 않음을 알 수 있었다.

5. 결론

본 논문에서는 새로운 하드웨어 기반 선인출 기법을 제안하였다. 멀티미디어 데이터의 메모리 참조 속도를 높이기 위해 사용하는 데이터 선인출 기법은 프로세서가 연산을 수행하는 동안 필요한 피연산자를 미리 캐시로 적재해서 메모리 접근 시간을 줄이는 효과적인 방법이다. 이러한 선인출 기법은 멀티미디어 데이터가 갖는 지역성과 규칙성을 바탕으로 하여 동작한다. M-RPT 구조를 사용한 캐시 메모리의 성능을 분석하기 위해 DEC system의 ATOM 시뮬레이터를 이용하여 명령어 트레이스를 산출하였고, 이 결과를 dinero III 시뮬레이터를 변형하여 만든 캐시 시뮬레이터의 입력으로 사용하였으며 다양한 캐시 크기에 대한 성능을 분석하기 위해 매개 변수로서 데이터 캐시 크기, 블록 크기 등을 가변하여 분석 결과를 얻었다. 기존 선인출 기법들 중에서 성능이 우수한 RPT방식과 비교하여, M-RPT방식이 캐시 미스율이 더 적게 산출되며, 버스 사용면에서도 거의 차이가 없음을 알 수 있었다. 특히 JPEG 보다는 MPEG 프로그램 데이터를 선인출한 것이 캐시 미스율이 더 적게 나오며 평균 27%의 성능 향상이 되었다. 향후 연구과제로는 선인출 결과를 활용하여 버스 트래픽을 감소시키는 연구가 필요하다.

참고문헌

[1] T-Fu Chen, and J-L Baer, "Effective Hardware-Based data prefetching for High-Performance Processors", IEEE Trans. Computers, vol. 44, No. 5, pp. 609-623, May 1995

[2] M-K Tcheun, H-S Yoon, S-R Maeng, "An adaptive sequential prefetching scheme in shared-memory multiprocessors",IEEE

[3] E-D M O, S-T Kofuji, " performance evaluation of the fixed seqeuntialprefetching on a bus-based multiprocessor : preliminary results"