공유메모리 다중처리기에서 상호연결망의 통신량을 고려하는 선인출 기법

박정우○ 손영철 정한조 맹승렬
한국과학기술원 전산학과

An Adaptive Sequential Prefetching using Traffic Information in Shared-Memory Multiprocessors

Jeong-Woo Park Youngchul Sohn Hanjo Jung Seungryoul Maeng
Dept. of Computer Science
Korea Adavanced Institute of Science and Technology

요 약

상호연결망을 기반으로 하는 공유메모리 다중처리기의 성능은 공유메모리 접근 속도에 많은 영향을 받는다. 선인출 기법은 프로세서의 계산과 데이터의 접근을 중첩시켜 메모리의 접근 속도를 줄인다. 기존의 선인출 기법들은 캐쉬미스 양을 줄이는 것만을 생각하여 상호연결망의 상황을 고려하지 않은 문제점이 있다. 본 논문에서는 응답이 늦은 선인출 이용하여 선인출 양을 조절함으로써 상호연결망의 경쟁을 줄이는 새로운 선인출 기법을 제안하고 프로그램 구동 모의실험을 통해 기존의 선인출 기법[1]에 비해 더 좋은 성능을 나타냄을 보인다.

1. 서론

지난 10년 동안 프로세서 처리속도는 매년 60%씩의 비약적인 발전을 해온 반면, 주 기억장치의 속도향상은 매년 10%미만에 그치고 있다. 이러한 프로세서와 메모리간의 속도차는 컴퓨터의 성능향상에 큰 장애요인이 되고 있다. 상호연결망(interconnection network)을 기반으로 하는 공유메모리 다중처리기에서는, 상호연결망 내에서의 자원경쟁과 지연시간으로 인해 기억장치의 접근시간은 더욱더 증가하므로 공유메모리의 접근시간을 줄이는 것은 중요한 문제이다. 캐쉬는 메모리 접근 시간을 줄이기 위해 사용되는 가장 일반적인 방법이지만 여전히 캐쉬미스(cache miss)때의 메모리 접근 시간은 성능향상에 큰 장애요인으로 남는다[2, 3]. 데이터 선인출 기법은 앞으로 사용될 데이터를 프로세서가 요구하기 전에 미리 캐쉬에 옮겨 캐쉬미스를 줄이는 방법이다. 기존의 선인출 기법들의 문제점은 선인출로 인해 상호연결망의 사용량이 증가하여 상호연결망에서의 지연시간이 길어져 선인출 효과를 경감시키는 것이다. 심지어 상호연결망의 상태에 따라 선인출을 하지 않았을 때보다 좋지 않은 성능을 보이기도 한다[4]. 특히, 선인출 양을 조절하는 하드웨어 선인출 기법들[1, 5]은 다른 선인출 기법들에 비해 캐쉬미스의 양을 효과적으로 줄여 좋은 성능을 나타내지만 상대적으로 선인출 요구량이 많아 상호연결망에서의 지연시간을 더욱더 증가시키는 문제가 있다[6].

본 논문에서는 상호연결망에서의 자원 경쟁과 통신량을 고려하여 선인출 양을 조절하는 새로운 하드웨어 선인출 기법을 제안한다. 기억장치와 상호연결망에 대한 경쟁이 심해지는 경우, 응답이 늦은 선인출(late prefetch)이 증가하게 된다. 본 논문에서는 이런 응답이 늦은 선인출을 이용하여 선인출 정도 K를 조절함으로써 상호연결망에서의 경쟁을 줄여 전체 성능향상을 꾀한다. 본 논문에서 제안한 선인출 기법의 성능은 RSIM(Rice Simulator for ILP Processors)[7] 상에서 SPLASH-2 병렬 프로그램 벤치마크[8]를 수행하여 평가한다. 모의 실험의 결과 제안하는 선인출 기법이 높은 대역폭에서는 기존의 선인출 기법[1]과 같은 성능을 나타내고, 낮은 대역폭에서는 더 좋은 성능을 나타냄을 보인다.

본 논문의 구성은 다음과 같다. 2.절에서는 데이터 선인출 기법들에 대한 기존 연구들을 서술하고, 3.절에서는 본 논문에서 제안하는 선인출 기법에 대하여 설명하고, 4.절에서는 모의 실험 결과를 분석하고, 5.절에서는 결론과 앞으로의 연구 방향을 기술한다.

2. 관련 연구

데이터 선인출 기법은 앞으로 사용될 데이터를 프로세서가 요구하기 전에 미리 캐쉬에 가져다 놓는 방법이다. 데이터 선인출은 단일처리기장치와 다중처리기장치 모두에서 캐쉬미스의 회수를 줄여 주므로 성능향상에 중요한 역할을 한다. 이러한 데이터 선인출 기법은 소프트웨어적인 방법과 하드웨어적인 방법이 있다.이 중에서 선인출 양을 조절하는 하드웨어 선인출 기법은 다른 선인출 기법에 비해 캐쉬미스의 양을 효과적을 줄여 좋은 성능을 나타낸다. 하지만, 각 캐쉬의 캐쉬미스양을 줄이는 것만을 생각해 선인출 양의 증가로 통신량과 자원경쟁이 늘어나는 문제를 안고 있다. 순차적 선인출 기법은 캐쉬미스가 났을 때 연속된 K개의 블럭을 선인출한다. 좋은 선인출 효과를 나타내는 선인출 정도 K는 프로그램마다 다르고, 같은 프로그램이라도 순간마다 변화하게 된다. 이런 K값의 변화를 캐쉬히트/미스(cache hit/miss)를 이용하여 조정하는 선인출 양을 조절하는 순차적 선인출 기법들이 Dahlgren[5]과 Tcheun[1]에 의해 제안되었다. Dahlgren의 방법[5]은 선인출 효율을 계산하여, 선인출 효율이 상위 임계값 이상이 되면 선인출 정도 K를 증가시키고, 하위 임계값 이하이면 K

값을 감소시켜 선인출 양을 조절한다. Tcheun의 방법은 테이블을 이용하여 연속된 메모리 블럭에 대한 순차적 접근 스트림의 K값을 관리한다.

3. 새로운 선인출 기법의 제안

프로그램의 수행 중에 선인출을 요청한 데이터가 아직 캐쉬에 들어오지 않은 상태에서 프로세서가 그 데이터를 사용하려는 경우가 발생할 수 있다. 그런 경우의 선인출을 응답이 늦어지는 선인출(late prefetch)이라고 한다. 프로세서가 연속된 데이터를 요구하는 속도가 빠른 경우나 통신량과 자원 경쟁이 많은 경우에 응답이 늦어 지는 선인출은 늘어난다. 이 절에서는 이러한 응답이 늦은 선인출을 상호연결망의 통신량과 자원 경쟁의 정도를 탐지하는 방법으로 사용하는 새로운 선인출 기법을 제안한다. 응답이 늦은 선인출이 생기는 경우는 프로세서가 요구한 블럭의 정보가 MSHR(Miss State Handling Register)[9]에 기록되어 처리를 기다리고 있는 상태이므로, 일반적인 경우에는 캐쉬히트로 생각하고 별다른 동작을 하지 않는다. 본 논문에서는 응답이 늦은 선인출이 발생하면 상호연결망 내에서 통신량과 자원경쟁이 심화된 상황으로 간주하고 선인출 양을 줄여 선인출로 인하여 상호연결망의 통신량과 자원경쟁이 더욱 많아 지는 경우를 줄이려는 것이다.

제안하는 선인출 기법은 Tcheun의 선인출 양을 조절하는 선인출 기법[1]을 기반으로 한다. 선인출 장치는 선인출 조정기와 네 개의 엔트리를 가지는 작은 크기의 테이블로 구성된다. 선인출 조정기는 캐쉬미스가 났을 때, 테이블로 부터 해당하는 순차적 접근 스트림의 현재의 K 값을 받아서 K개의 블럭을 선인출한다. 캐쉬미스가 난 블럭이 A 라면, $A+1, A+2, ..., A+K$ 블럭들을 선인출한다. 테이블의 각 엔트리에는 각각의 순차적 접근 스트림에 해당하는 블럭 주소 A와 선인출 정도 K가 들어간다.

- 블럭 A에서 캐시미스가 발생하면,
  - 블럭 A를 가져온다.
  - 테이블에서 A를 찾는다.
    - $<A, K>$를 찾았을 때,
      - 블럭 $A+1, A+2, ..., A+K$를 차례로 선인출한다.
      - 테이블의 $<A, K>$를 $<A+K+1, K+1>$로 수정한다.
    - A를 찾지 못했을 때,
      - 블럭 $A+1$을 선인출한다.
      - 테이블에 $<A+2, 2>$를 삽입한다.
- 응답이 늦은 선인출이 발생하면,
  - 테이블의 모든 엔트리의 K에 대해서,
    - $K > 0$ 이면, K를 $K-1$로 수정한다.

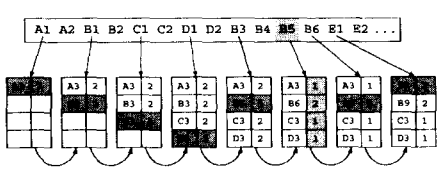

그림 1: 선인출 테이블의 변화

그림 1은 아래와 같은 접근 스트림에서

$A_1, A_2, B_1, B_2, C_1, C_2, D_1, D_2, B_3, B_4, B_5, B_6, E_1, E_2, ...$

블럭 B_5가 응답이 늦은 선인출이 되는 경우, 선인출 테이블의 변화를 나타내는 그림이다. 블럭 B_5의 선인출 도중에 응답이 늦은 선인출이 발생하면, 테이블의 모든 엔트리에 있는 K값들을 1씩 감소시킨다.

4. 성능 평가

4.1 모의 실험 환경

본 논문에서는 모의 실험을 위해 RSIM[7] 모의 실험기를 사용하였다. 연결망은 4×4의 2-D 메쉬구조를 가지는 웜홀 라우팅 방식이다. 의 실험에 사용된 벤치마크 프로그램은 SPLASH-2[8]의 FFT, LU, Water, Ocean 등이다. 실험의 측정 기준은 응답이 늦은 선인출의 양, 상호연결망에서의 지연시간, 캐쉬미스의 양, 실행시간 등이다.

processor	4 way superscalar
cache	L1 4KB, L2 16KB
cache line size	32bytes
network clock speed	4 processor cycles
network bandwidth	8 bytes (high bandwidth)
	4 bytes (medium bandwidth)
	2 bytes (low bandwidth)

표 1: 모의 실험 장치의 기본 구성

모의 실험은 no, asp, more, ours를 각각 높은 대역폭, 중간 대역폭, 낮은 대역폭을 가지는 상호연결망 상황에서 수행하였다. no는 선인출이 없는 경우, asp는 Tcheun의 선인출 기법[1]을 사용하는 경우이다. more는 응답이 늦은 선인출이 발생하면 캐쉬미스로 처리하여 선인출 양을 늘이는 방법이고, ours는 본 논문에서 제안하는 선인출 기법이다.

4.2 새로운 선인출 기법의 성능 평가

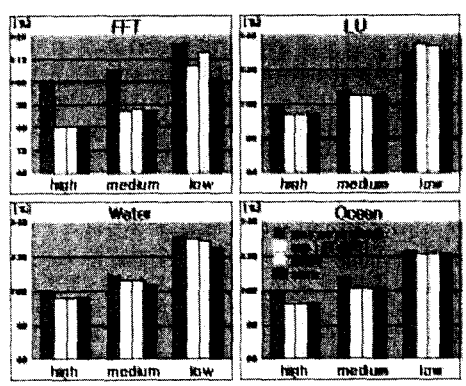

그림 2: 실행시간의 비교

그림 2는 각각의 수행시간을 나타내는 그래프들이다. Ocean을 제외한 모든 벤치마크에서 ours는 가장 좋은 성능을 보이고, Ocean에서는 asp, more, ours가 거의 같은 성능을 보이고 있다. Ocean은 프로그램의 특성상 원격 메모리에 대한 선인출의 효과가 없어 상호연결망의 대역폭과는 관계없이 응답이 늦은 선인출이 거의 발생하지 않기 때

문이다. 높은 대역폭의 상황에서 ours는 asp, more와 같은 성능을 보이고 낮은 대역폭으로 갈수록 ours가 좋은 성능을 보이고 있다. 이런 결과는 아래와 같이 분석할 수 있다.

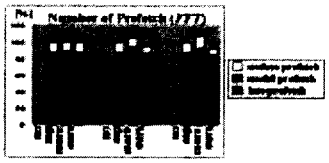

그림 3: 선인출 양의 비교

그림 3은 FFT의 선인출양을 나타내는 그래프이다. Ocean을 제외한 세 개의 벤치마크에서 같은 그림 3과 같은 결과를 얻을 수 있었다. 응답이 늦은 선인출의 빈도에 따라 more는 선인출 양을 늘렸고, ours는 선인출 양을 줄였다. 이렇게 선인출 양을 조절하면 캐쉬미스의 양과 상호연결망에서의 지연시간에 영향을 미치게 된다.

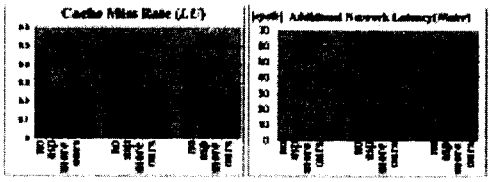

그림 4: 캐쉬미스의 양과 추가되는 지연시간의 비교

그림 4은 LU의 캐쉬미스 양과 Water의 상호연결망에서의 추가적 지연시간을 나타내는 그래프이다. 대역폭에 관계없이 no와 asp의 캐쉬미스의 양은 변화가 없다. 하지만, 대역폭이 줄어듦에 따라 선인출의 양을 늘인 more는 asp보다 줄였고, 선인출의 양을 줄인 ours는 캐쉬미스의 양이 오히려 늘어 났다. 하지만, 그림 4에서 보는 것과 같이 대역폭이 줄어듦에 따라 ours는 asp와 more에 비해 상호연결망에서의 지연시간을 줄였음을 알 수 있다. 즉, ours가 asp와 more에 비해 좋은 성능을 보이는 이유는 캐쉬미스의 양은 증가하지만 상호연결망에서의 지연시간을 줄이기 때문이다.

5. 결론

데이터 선인출 기법은 앞으로 사용될 데이터를 프로세서가 요구하기 전에 미리 캐쉬에 옮겨 공유메모리 다중 처리기의 성능을 향상시킨다. 기존의 하드웨어 선인출 기법들은 캐쉬미스 양을 줄이는 것 만을 생각하여 선인출에 의해 늘어난 상호연결망에서의 지연시간은 고려하지 않았다. 본 논문에서는 상호연결망에서의 통신량과 자원경쟁을 고려하여 선인출 양을 조절하는 하드웨어 선인출 기법을 제안하고 평가하였다. 상호연결망에서의 지연시간이 길어지면 응답이 늦은 선인출이 늘어난다. 본 논문에서는 제안하는 선인출 기법은 이러한 응답이 늦은 선인출이 발생하면 선인출 양을 줄여 선인출 양을 조절하는 방법이다.

모의 실험을 통해 본 논문에서 제안하는 선인출 기법은 상호연결망의 대역폭이 충분한 경우에는 다른 선인출 기법들과 같이 동작하고, 상호연결망의 대역폭이 줄어들면 선인출 양을 조절함으로써 기존의 선인출 기법보다 좋음을 보였다. 이는 선인출 양이 줄어들면 캐쉬미스는 증가하지만, 상호연결망에서의 지연시간이 줄어들기 때문이다. 낮은 대역폭의 상호연결망에서는 상호연결망의 지연시간이 전체 성능에 더 중요한 역할을 함을 알 수 있었다. 앞으로의 연구과제는 보다 실제적인 환경에서의 실험을 통한 정확하게 분석하는 것과 캐쉬미스의 양을 늘이지 않으면서도 상호연결망에서의 지연시간을 줄일 수 있는 방법을 고안하는 것이다.

참고 문헌

[1] M. K. Tcheun, H. Yoon, and S. R. Maeng, "An Adaptive Sequential Prefetching Scheme in Shared Memory Multiprocessors," in *International Conference on Parallel Processing*, pp. 306–313, IEEE, Sep. 1997.

[2] S. P. VanderWiel and D. J. Lilja, "When Caches Aren't Enough: Data Prefetching Techniques," *IEEE Computer*, vol. 30, pp. 23–30, July 1997.

[3] T. C. Mowry, "Tolerating Latency in Multiprocessors through Compiler-inserted Prefetching," *ACM Transactions on Computer Systems*, vol. 16, pp. 55 92, Feb. 1998.

[4] S. Kim and A. V. Veidenbaum, "The Effect of Limited Network Bandwidth and its Utilization by Latency Hiding Techniques in Large-Scale Shared Memory Systems," in *Parallel Architectures and Compilation Techniques - Conference Proceedings*, pp. 40–51, Nov. 1997.

[5] F. Dahlgren and M. Dubois, "Sequential Hardware Prefetching in Shared-Memory Multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, pp. 733 746, July 1995.

[6] S. P. VanderWiel and D. J. Lilja, "Data Prefetch Mechanisms," *ACM Computing Surveys*, 1998.

[7] V. S. Pai, R. Ranganathan, and S. V. Adve, "RSIM Reference Manual version 1.0," tech. rep., Rice University, Aug. 1997.

[8] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," in *Procedings of the 22nd Annual Internatinal symposium on Computer Architecture*, pp. 24 36, June 1995.

[9] D. Croft, "Lockup-free instruction fetch/prefetch cache organization," in *Procedings of the 8th Annual Internatinal symposium on Computer Architecture*, pp. 81 87, 1981.