Intelligent 한 메모리 시스템에서의 Fine-Grained SW Off-loading 을 위한 성능 분석

허인구*, 김용주*, 이진용*, 이지훈*, 이종원*, 백윤흥*,
*서울대학교 전기컴퓨터공학부
e-mail : {igheo, yjkim, jylee, jhlee, jwlee} @sor.snu.ac.kr
ypaek@snu.ac.kr

Performance Analysis for Fine-Grained SW Offloading in Intelligent Memory System

Ingoo Heo*, Yongjoo Kim*, Jinyong Lee*, Jihoon Lee*, Jongwon Lee*, Yunheung Paek*
*Dept. of Electrical Engineering and Computer Science, Seoul National University

요 약

전통적으로 컴퓨터의 성능은 중앙 연산 장치 (CPU)의 성능에 따라 좌지우지 되어 왔다. 하지만 CPU 의 성능이 지속적인 발전을 거듭하여 무어의 법칙을 비교적 충실히 따라가고 있는 반면, 메모리의 성능은 근래 들어 더디게 발전되는 형국이다. 때문에, CPU 와 메모리 간의 성능격차로 인해 메모리의 낮은 성능이 전체 시스템의 성능을 저하시키는 “Memory Wall Problem”은 점점 큰 문제로 대두되고 있다. 이러한 문제를 해결하기 위해 많은 연구에서 메모리 자체의 성능을 발전시키는 것은 물론 메모리 내부에 연산 처리 능력을 추가하여 시스템 전체의 성능을 향상 시키는 시도들을 해왔다. 이 논문에서는 이러한 Intelligent 한 메모리 시스템에서의 SW Off-loading 을 위한 성능 분석을 다룬다. 이전의 연구들이 주로 큰 단위의 Off-load 를 다뤘던 것에 비해 이 논문에서는 작은 단위의 Off-load, 더 정확히는 어셈블리 수준의 Off-load 의 효과에 대해 분석한다. 또한 현재의 어셈블리 수준의 Off-load 의 한계를 지적하고 이를 극복하기 위한 루프 레벨 Off-load, 새로운 Technology 와 아키텍처에 대해서도 소개한다.

1. 서론

현대의 컴퓨터는 대부분 전통적인 Von Neumann 아키텍쳐의 구조를 따르고 있다. 이러한 Von Neumann 구조는 CPU 와 같은 연산 장치와 DRAM 과 같은 기억 장치로 구성되어 있고, 이 두 개의 구성 요소가 가장 중요한 역할을 한다. 하지만 CPU 의 성능이 공정의 발전, 아키텍쳐의 발전에 힘입어 무어의 법칙을 비교적 충실히 따라가며 향상되어 가는 반면, 메모리의 성능은 그렇지 못하다. 메모리의 경우 황의 법칙에 따라 집적도가 높아지면서 전체적인 기억 용량에서는 꾸준한 향상을 이루고 있지만 동작 속도에 있어서는 향상 속도가 더딘 편이다. (그림 1) 이 때문에 메모리의 느린 성능이 전체 시스템의 성능을 저하시키는 “Memory Wall Problem”은 꾸준히 문제가 되어 왔다.

이러한 Memory Wall Problem 을 해결하기 위한 방법으로는 여러 가지가 있을 수 있다. 가장 직관적인 해결 방법은 메모리 자체의 성능을 향상 시키는 것이다. CPU 의 경우처럼 메모리 역시 새로운 공정의 도입을 통해 집적도와 함께 반응 속도를 향상시켜 왔다. 최근에는 3D 집적 기술을 이용한 방법[2]들이 대두되고 있어 메모리 자체의 성능이 크게 증가할 것으로 기대되고 있다.

또 하나의 방법은 Processing-in-Memory(PIM) [3] 같은 방법을 이용 하여 메모리 내부에서 연산이 이루어지게 하는 Intelligent 한 메모리 시스템을 이용하는 것이다. 이러한 방식은 CPU 와 메모리가 버스로 연결되어 데이터를 주고 받는 것보다 훨씬 빠른 속도로 데이터를 전송하는 것이 가능하기 때문에 메모리 접근이 잦은 프로그램을 수행할 때는 매우 효과적일 수 있다. 이 때문에 멀티미디어와 같이 다량의 메모리 접근을 필요로 하는 프로그램이나 그래프 알고리즘과 같이 비정규적 메모리 접근을 주로 하는 프로그램들은 이러한 PIM 방식의 아키텍쳐에서 성능 향상을 크게 얻을 수 있었다 [4].

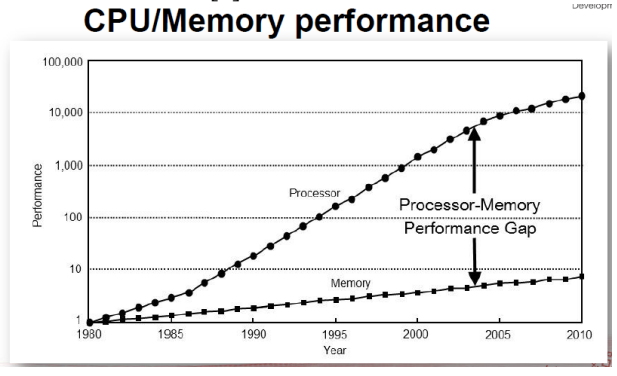

그림 1 CPU/Memory Performance [1]

이러한 방식들은 대부분의 프로그램 코드들을 메모리 내부에 있는 프로세서들에서 수행 했다. 이는 메모리 내부에 집적한 프로세서들이 기본적으로 완전한 프로세서의 기능들을 다 할 수 있기 때문에 가능한 일이기도 했다. 하지만 메모리 내부에 프로세서를 집적한다는 것은 공정 관점에서는 매우 비효율 적인 일이었다. 기본적으로 메모리에 최적화된 공정과 프로세서에 최적화된 공정은 다른 성격을 가지기 때문이다. 이 때문에 메모리 내부에 집적된 프로세서들의 성능은 기존의 프로세서들에 비해 떨어질 수 밖에 없었다. 또한 보편적인 시스템에서 고성능의 CPU 의 존재하는 만큼 메모리 내부에 추가적인 CPU 를 추가하는 것은 비용과 효용성 측면에서 큰 걸림돌일 수 밖에 없었다.

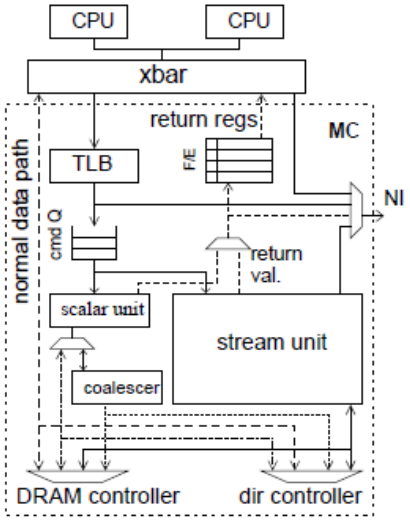

그림 2 Active Memory Unit [5]

이를 극복하기 위해 [5]와 같은 비교적 작은 단위의 기능만을 메모리 쪽에서 수행할 수 있는 방법론도 대두되었다. 이러한 접근 방식에서는 프로그램의 전체 혹은 대부분을 메모리 쪽에서 수행하기 보다는 간단한 Scalar 연산이나 SIMD 연산과 같이 분리하기 쉬운 형태의 기능만을 Off-load 시킨다. 큰 단위(Coarse-Grained)를 Off-load 하는 방식에 비해서는 성능적인 이득이 크다고 볼 수 없지만 다른 면에서 몇 가지 장점을 가진다. 첫째는 기존의 시스템에 무리 없이 적용이 가능하다. 앞서 설명한 바 있듯이, 기존의 컴퓨터 시스템에는 기본적으로 고성능의 CPU 가 존재한다. 이러한 상황에서 CPU 를 대부분 시간 활용하지 않은 채로 메모리 쪽 연산 기능들만을 사용한다는 것은 매우 꺼려지는 일이다. 따라서 기존의 CPU 를 적극 활용하면서도 메모리 쪽 연산 기능을 극대화 할 수 있게 하는 것이 바람직하다. 두 번째는 발열과 전력 소모에서의 장점이다. 메모리의 경우 CPU 와 달리 쿨링 시스템이 미약하다. 때문에 많은 열을 발생 시킬 수 있는 Logic 들을 다수 집적하는 것은 발열 문제를 악화시킬 수 있다. 반도체의 특성상 발열이 늘어나면 전력 소모는 증가하고 동작속도는 느려지므로, 메모리 쪽에서는 작은 단위(Fine-Grained)의 기능만을 담당하는 것이 적당하다.

본 논문에서는 이러한 Fine-Grained Off-load 를 위한 SW 성능 향상에 관한 분석을 다룬다. 기존의 방식들이 주로 Coarse-Grained Off-load 를 해왔던 것과는 달리 본 연구에서는 가장 작은 Off-load 단위인 어셈블리 수준에서의 off-load 를 고려한다. 이를 위해 Intelligent 메모리 시스템은 간단한 수준의 ALU 연산이 가능한 것으로 가정한다. 이러한 가정 하에 Histogram, Breadth-First-Search (BFS), Needleman-Wunsch(NW)와 같은 비정규적 메모리 접근을 하는 프로그램들의 Kernel 에서의 성능 향상을 분석한다. 또한, 이러한 어셈블리 수준 off-load 의 한계를 지적하고 개선점을 미래 연구로 제시한다.

2. Intelligent 메모리 시스템에 적합한 프로그램

이 장에서는 Intelligent 한 메모리 시스템에 적합한 프로그램에 대해서 논하도록 한다. 앞선 연구들에서도 언급했듯이 기본적으로 이러한 메모리 시스템 내부의 연산 처리 능력을 잘 활용하기 위해서는 크게 두 가지 장점에 주목해야 한다.

첫 번째는 메모리 내부에 연산기를 삽입함으로써 얻게 되는 메모리 Latency 의 감소이다. 다시 말해 데이터를 메모리에 요구하고 이를 받게 되는 데 걸리는 시간이 현격하게 감소하는 것이다. 이렇게 줄어든 Latency 를 잘 활용할 수 있는 응용에는 그래프 알고리즘과 같이 빈번하고 비정규적인 메모리 접근을 취하는 프로그램들이 있다. 이러한 프로그램들은 기존의 CPU 에서의 Cache Miss Rate 가 상당히 높기 때문에 CPU 가 메모리에서 데이터를 끌어와 연산해서 쓰는 경우 성능이 매우 나쁘다. 또한 이 저조한 성능의 원인이 예측이 힘든 메모리 접근 패턴에 기인하기 때문에 최적화 하는 것도 쉬운 일이 아니다. 메모리 내부에 연산처리 능력을 삽입하게 될 경우 이 기능을 활용하여 빈번하고 비정규적인 메모리 접근을 가지는 프로그램들을 상당 부분 향상 시킬 수 있다[4].

두 번째 장점은 메모리 내부의 Datapath 를 수정함으로써 얻을 수 있는 넓은 대역폭이다. 이를 이용해 태스크 수준 병렬성 (TLP : Task Level Parallelism) 이나 데이터 수준 병렬성 (DLP : Data Level Parallelism) 을 살린 이전의 연구들도 많다. 이러한 연구들은 주로 SIMD 방식의 연산처리가 가능한 멀티미디어나 과학 관련 응용들이 주를 이루었다.

우리의 연구에서는 주로 첫 번째 장점인 메모리 Latency 와 관련된 응용들을 대상으로 한다. 서론에서 언급했듯이, 메모리 내부에 간단한 수준의 연산 기능만을 집적한다면, TLP 나 큰 수준의 DLP 를 처리하는 것은 힘들기 때문이다.

이에 따라 최종적으로 우리는 Histogram, BFS, NW 를 대상으로 하였다. Histogram 은 널리 알려진 프로그램으로 데이터를 특정 범위에 해당하는 bin 에 분류하는 프로그램이다. bin 의 값을 읽고 이 값에 1 을 더하고 다시 저장하는 연산이 histogram 구성의 kernel 이다. 이러한 과정에서의 덧셈 1 은 매우 간단한 연산이

므로 메모리 쪽 연산기에서 처리가 가능하다. 또한 bin 의 개수가 많을 경우 메모리 접근 패턴 역시 비정규적이므로 메모리 쪽 연산기의 도움을 많이 받을 수 있을 것이다.

BFS 는 그래프의 각 Node 들의 cost 를 계산하는 유명한 그래프 알고리즘이다. Cost 를 계산하는 과정에서 각 Edge 를 통해 연결된 다른 Node 들을 읽어와야 한다. 그래프의 크기가 매우 클 경우 이렇게 다른 Node 들을 읽어 오는 과정은 매우 비정규적인 메모리 접근을 하게 된다. 따라서 기존의 CPU 에서 이를 수행했을 때의 Cache Miss Rate 도 상당히 높다. 이 때문에 이러한 비정규적 메모리 접근을 하는 부분의 코드들을 메모리 쪽으로 off-load 할 경우 성능 향상이 클 수 있다.

NW 은 유전자 관련 알고리즘이다. 이 응용의 경우 대각선 방향의 어레이 접근을 많이 하다 보니 Cache Miss Rate 가 상당히 높은 편이다. 이 때문에 적용 대상 프로그램으로 선정하였다.

3. 실험 방법 및 결과

3 가지 응용에 대해서 실험을 진행하였다. 기본적인 컴퓨터 시스템은 DDR3 메모리와 Intel Sandy Bridge Processor 3.3GHz 시스템을 가정하였다. 이 시스템에서의 Memory Latency 는 표 1과 같다 [6].

표 1 Cache Miss Penalty

L1 Cache Miss Penalty	12 cycles
L2 Cache Miss Penalty	245 cycles

3 개의 프로그램에서 Kernel 코드들을 분석하여 찾아내었고, 이 코드들을 ARM Development Suite 를 이용하여 ARM ISA 로 변환하였다. Sandy Bridge 의 경우 x86 혹은 amd64 ISA 를 이용하지만 이 경우 메모리 연산과 레지스터 연산을 구분하는 것이 쉽지 않기 때문에 코드 가독성을 고려하여 ARM ISA 로 구분하였다. 하지만 기본적으로 연산의 기능은 동일하기 때문에 성능 분석에 있어서의 차이점은 없다.

메모리 내부에 있는 연산 처리기는 간단한 ALU 연산과, Shift, 곱셈이 가능한 것으로 설정하였고, DRAM 내부에서 메모리를 접근할 경우 메모리 접근에 걸리는 시간은 10 ns 로 설정하였다. 3.3 GHz 프로세서의 경우 33 cycle 에 해당한다. Off-load 된 code 블락의 첫 명령어가 수행되는 데 걸리는 시간은 66 cycle 로 설정하였고 그 이후의 명령어들은 이 Latency 와는 별도로 명령어 자체의 Latency 만을 적용하는 것으로 하였다.

그림 3은 BFS 의 Off-load 예시이다. 주어진 소스코드를 보면 for 문 안에 있는 코드들이 매우 비정규적인 메모리 접근을 하는 것을 알 수 있다. Edge 에 저장된 Node Number 에 따라 다음에 접근하게 될 메모리 영역이 결정되고 이에 따라 cost 가 새로이 계산된다. 이러한 과정의 어셈블리 코드 중 오른쪽 부분들을 메모리 쪽으로 Off-load 시킨다. Branch 의 경우 Predicated Execution 을 이용하여 처리하는 것을 가정하였다. 이렇게 처리할 경우 Edge 에서 다음 Node 를 읽어올 때 발생할 수 있는 Cache Miss 가 일으키는 성능 저하가 사라지기 때문에 성능 향상을 크게 거둘 수 있다.

```
for(int i=0; i<10; i++)
{
        int id = h_graph_edges[i];
        if(!h_graph_visited[id])
        {
                h_cost[id]=h_cost[tid]+1;
                h_updating_graph_mask[id]=true;
        }
}
```

```
ADD     r1,r13,#0x4b0
LDR     r1,[r1,r0,LSL #2]
ADD     r2,r13,#0x320
LDR     r2,[r2,r1,LSL #2]
CMP     r2,#0
BNE     0x44
ADD     r2,r13,#0x190
LDR     r2,[r2,r14,LSL #2]
ADD     r3,r13,#0x190
ADD     r2,r2,#1
STR     r2,[r3,r1,LSL #2]
STR     r12,[r13,r1,LSL #2]
```

그림 3 Breadth First Search 예시

가정된 실험 환경하에서 3 가지 프로그램에 대해서 Intelligent 한 메모리 시스템을 사용했을 때의 성능 향상을 측정하였다. 기존의 CPU-Memory 시스템을 이용할 경우 성능의 핵심은 Cache Miss Rate 가 쥐고 있다. 이 Cache Miss Rate 는 [7]의 L2 Miss Rate 를 참고하였다. 이 Rate 를 기준으로 Miss 가 날 때 245 cycle 의 메모리 Latency 를 적용하였다. 반대로 L1 에서 hit 가 났을 때의 Latency 는 4 를 적용하였다. Histogram 의 경우 알려진 Miss Rate 가 없었기 때문에 40%를 적용하였다. Bin 의 크기가 매우 클 경우 Miss Rate 는 이 것보다 더 나빠질 수 있다. 기존 아키텍쳐 대비 Intelligent 한 메모리 시스템의 성능 향상은 그림 4와 같다.

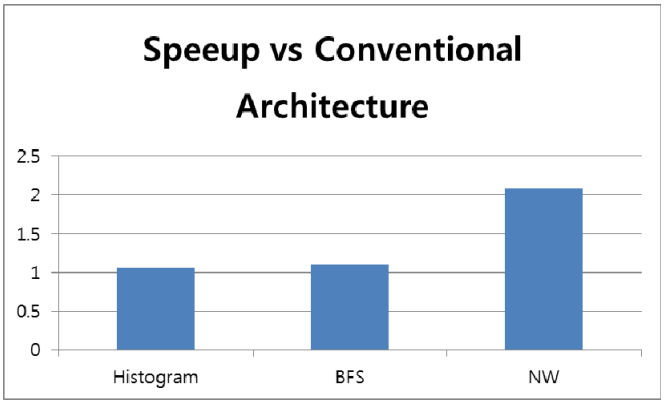

그림 4 성능 향상 비교

NW 와 같이 메모리 연산이 더 많을수록 성능 향상의 폭이 컸다. Off-load 한 Kernel 을 기준으로 메모리 연산 (Load)의 개수는 각각 1, 3, 4 였다. 이 때문에 NW 가 줄어든 메모리의 Latency 의 혜택을 가장 많이 보았다. 또한 세 응용의 L2 Miss Rate 는 각각 40, 21,41.2%로 계산되었는데, Miss Rate 가 가장 높은 NW 가 역시 성능 향상이 가장 컸다. 이를 분석해 보면 앞서 예상했던 바와 같이 메모리 접근 방식이 비정규적이어서 Cache Miss Rate 가 높은 경우가 이득이 가장 큰 것을 알 수 있다.

하지만 이러한 실험 결과는 동시에 문제점을 지적

하고 있다. Histogram 의 경우 Off-load 한 부분의 코드 크기가 가장 작았는데, 이 때 가장 이득이 적었다. 이는 처음 메모리 부분의 연산 기능을 시작하기 위해 필요한 초기화에 드는 66 cycle 에 있다. Off-load 하는 부분의 크기가 어느 정도 큰 BFS 나 NW 의 경우에는 추후에 메모리 연산을 하게 됨으로써 얻어지는 이득이 이 66cycle 의 오버헤드를 상쇄시킬 수 있었다. 하지만 Histogram 의 경우 그러한 상쇄 효과를 얻기가 힘들었다. 따라서, 우리는 조금 더 높은 수준의 Off-load 가 필요하다는 결론을 얻을 수 있다. 이를 위해서는 지금의 어셈블리 수준의 극단적인 Fine-Grained Off-load 보다는 루프 레벨 수준의 Fine-Grained Off-load 에 주목할 필요가 있을 것이다.

4. 결론 및 미래 연구

본 논문에서는 메모리 내부에 연산 처리기를 포함한 Intelligent 메모리 시스템에서 코드를 off-load 하는 방법의 성능 향상에 대해 알아 보았다. 우리는 기존의 접근 방식과는 다르게 가장 작은 단위의 off-load 인 어셈블리 수준의 Fine-Grained Off-load 를 통한 성능 향상의 효과를 보았다. NW 와 같이 비정규적 메모리 접근 패턴에 의한 Cache Miss Rate 가 높고 Off-load 할 수 있는 코드의 크기가 큰 경우에는 Kernel 에서 2 배가 넘는 성능 향상을 볼 수 있었다. 하지만 Histogram 과 같이 Cache Miss Rate 가 높더라도 off-load 하는 코드의 크기가 작은 경우에는 성능 향상의 크기가 크지 않음을 알 수 있었다.

우리는 이러한 어셈블리 수준의 Fine-Grained Off-load 의 한계를 극복하기 위하여 좀 더 높은 수준인 루프 레벨 수준의 Fine-Grained Off-load 를 미래 연구로 진행할 것이다. Off-load 할 수 있는 코드의 크기가 커짐에 따라 얻을 수 있는 이득이 더 커질 것으로 기대 된다. 다만 이러한 과정에서도 메모리 쪽에 집적하는 연산 처리기의 복잡도는 낮은 수준으로 계속 유지될 수 있어야 한다.

또한 현재 실험에서 가정된 시스템의 경우 메모리의 구조가 2D 이지만 앞으로 3D 집적 기술이 점차 발전하면 3D DRAM 구조를 활용하여 메모리 내부에서의 접근 속도가 더욱 빨라질 것으로 기대된다. 특히 이렇게 3D DRAM 을 사용할 경우 Logic 에 사용되는 die 를 따로 집적할 수 있기 때문에 기존 보다 더 많은 Logic 을 집적하여 간단한 수준의 병렬처리를 더 활용하는 것도 가능하다. 이 역시 우리의 미래연구로서 앞으로 차세대 반도체 기술이 될 3D DRAM 에서의 코드 Off-load 에 대한 연구도 진행하도록 하겠다.

ACKNOWLEDGEMENT

이 논문은 2011 년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(No. 2011-0018609)과 교육과학기술부/한국과학재단 우수연구센터 육성사업의 지원으로 수행되었음 (과제번호 2012-0000470)

참고문헌

[1] J.L.Hennessy, D.A. Patterson, Computer Architecutre : A Quantative Approach, Morgan Kaufmann Publishers

[2] U.Kang, et al., “8 Gb 3-D DDR3 DRAM Using Through-Silicon-Via Technology”, JSSC, vol.45, no. 1, pp. 111-119, 2010.

[3] Kogge, Peter M., Jay B. Brockman, Thomas Sterling, and Guang Gao, "Processing-In-Memory: Chips to Petaflops," Workshop on Intelligent RAM, International Symposium on Computer Architecture, Denver, CO, June 1, 1997

[4] Jeff Draper, Jacqueline Chame, Mary Hall, Craig Steele, Tim Barrett, Jeff LaCoss, John Granacki, Jaewook Shin, Chun Chen, Chang Woo Kang, Ihn Kim, and Gokhan Daglikoca. 2002. The architecture of the DIVA processing-in-memory chip. In Proceedings of the 16th international conference on Supercomputing (ICS '02)

[5] Zhen Fang, Lixin Zhang, John B. Carter, Ali Ibrahim, and Michael A. Parker. 2007. Active memory operations. In Proceedings of the 21st annual international conference on Supercomputing (ICS '07).

[6] http://www.7-cpu.com/cpu/SandyBridge.html

[7] Shuai Che; Boyer, M.; Jiayuan Meng; Tarjan, D.; Sheaffer, J.W.; Sang-Ha Lee; Skadron, K.; , "Rodinia: A benchmark suite for heterogeneous computing," Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on , vol., no., pp.44-54, 4-6 Oct. 2009