인텔 KNL 프로세서 사례를 통한 고성능 온칩 메모리의 성능 병목 분석 및 해결 방안 연구

변은규
한국과학기술정보연구원 국가슈퍼컴퓨팅본부
ekbyun@kisti.re.kr

An experimental study on Intel KNL processor to improve the performance of high bandwidth on-chip memory

Eun-Kyu Byun
Division of National Supercomputing, Korea Institute of Science and Technology Information

요 약

나날이 커져가는 데이터 처리량의 수요를 충족시키기 위한 방법의 하나로 수십개의 코어와 여러 채널의 고대역폭 메모리를 탑재한 프로세서가 상위 슈퍼컴퓨터 시스템에 도입되어 사용되고 있다. 이러한 Scale-out 방식은 성능 한계를 크게 끌어올릴 수 있지만 제대로 된 작업 배분이 되지 않았을 때 성능이 떨어질 가능성이 있다. 본 연구에서는 인텔 KNL 프로세서의 고성능 온칩 메모리의 성능 벤치마크를 진행하여 병목 현상이 실제로 존재함을 확인하였다. 또한 이런 성능 저하 패턴을 찾아내고 원인을 분석하여 향후의 시스템에서 이러한 문제를 최소화하기 위해서 하드웨어, 시스템 소프트웨어 수준에의 보완 방안을 제안한다.

1. 서론

나날이 늘어가는 빅데이터 및 인공지능 수요로 인해 필요로 하는 계산 능력 및 데이터 처리량이 폭발적으로 증가하고 있다. 이러한 등장 이전에 GPGPU 가 도입되어 계산성능이 크게 향상되었으나 CPU 와 호스트 메모리 사이의 데이터 이동이 병목지점으로 지적되어 이를 해결하기 위해 공유 메모리 버스 기술 등이 발전하고 있다. 이는 고성능 컴퓨팅을 위해서는 계산능력 뿐 아니라 대용량 데이터의 빠른 이동 및 처리를 위한 메모리 성능이 매우 중요하다는 것을 의미한다. 이를 해결하기 위한 다른 방법으로 최신 고성능 프로세서는 하나의 칩에 수십개의 코어와 고대역폭 메모리를 탑재하는 형태가 나타났다. Intel 에서 출시한 Knights Landing(이하 KNL)프로세서는 최대 74 개의 코어와 DDR 메모리보다 4 배이상의 대역폭을 가진 8 채널의 최대 16GB 의 MCDRAM(multi-channel DRAM)을 탑재하고 있다.[1] 또한 2020 년 6 월 기준 Top500 기준 최고 성능의 슈퍼컴퓨터인 Fugaku 를 구성하는 A64FX 역시 48 개의 코어와 4 채널의 32GB 의 HMB2(high-bandwidth memory)를 프로세서내에 탑재하고 있다.[2]

다수의 코어와 여러 채널의 고대역폭 메모리를 하나의 프로세서 칩에 탑재하는 Scale-out 방식은 현세대의 HPC 시스템에서의 성능을 크게 향상시킬 수 있는 가장 일반적인 방법이다. 하지만 이러한 구조에는 작업의 배치가 제대로 되지 못하여 몇 개의 동시에 일부의 채널의 메모리만 활용되거나 칩내 네트워크의 한 코어가 병목이 되는 상황이 발생하면 프로세서의 성능의 저하될 수 있다. 이를 해결하기 위한 기술은 하드웨어 설계단계에서 뿐만 아니라 시스템 소프트웨어 단계에서 함께 고려되어야 한다.

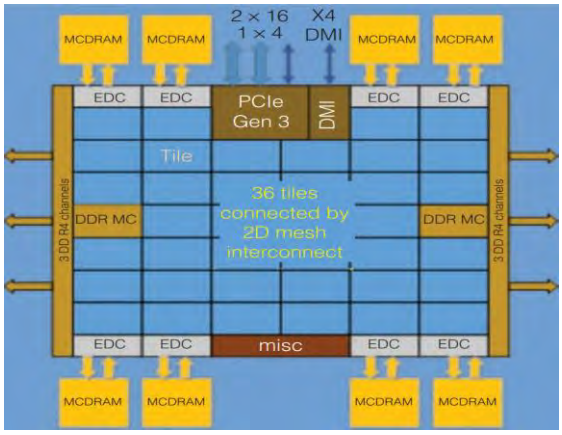
소프트웨어 수준의 최적화를 위해서는 먼저 하드웨어 특성에 대한 충분한 이해가 필요하고 어떤 데이터가 하드웨어로부터 직접 제공되어야 하는지 파악해야 한다. 본 연구에서는 KNL 프로세서를 이용한 다양한 메모리 벤치마크 실험을 통해 MCDRAM 의 성능 특성을 파악하고 병목 상황 분석을 통해 성능을 최적화하기 위해 필요한 사항을 도출하고자 하였다. 이러한 사례 연구가 향후 고성능 프로세서 및 시스템을 설계하고 이를 활용한 최적화에 활용될 수 있을 것이다.

NERSC 의 Cori[3], KISTI 의 누리온[4]를 비롯하여 KNL 프로세서를 이용한 슈퍼컴퓨터가 많이 도입되었고 KNL 성능 분석 연구가 많이 진행되었다. 그러나 기존의 연구는 MCDRAM 의 캐시모드, 클러스터 모드 등의 프로세서 설정에 따른 포괄적인 성능 차이에 관한 내용 위주로 진행되었다.[6] 본 연구에서는 프로세서 내부의 채널 구조에 집중하여 미시적인 성능 저하

요소에 대한 분석을 진행하였다.

2. KNL 프로세서 특성

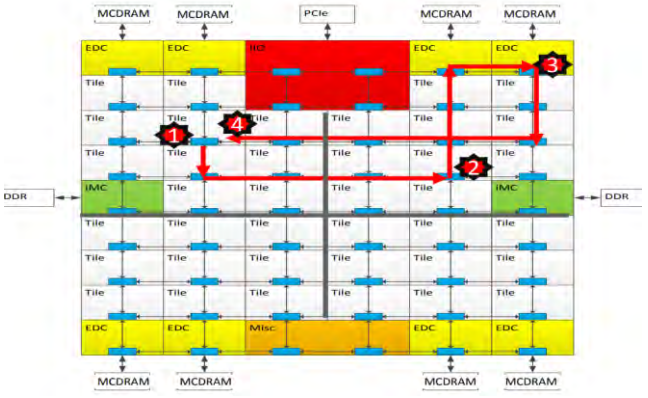
Knights Landing Xeon Phi 프로세서는 2016 년에 Intel 에서 출시한 HPC 용 프로세서로 기존의 가속기 형식의 Xeon Phi 와 다르게 독립적으로 호스트 CPU 로서 동작한다. 72 개에 달하는 계산 코어와 16GB 의 고성능 MCDRAM 을 하나의 칩에 포함하고 있다. 그림 1 은 KNL 의 구조를 나타낸다. 2D mesh 구조의 온칩 네트워크에 계산, 메모리, I/O 장치들이 연결되어 있다. 36 개의 계산 타일 각각은 2 개의 코어와 공유되는 1MB L2 캐시, 그리고 캐시 coherence 와 routing 을 담당하는 Caching/Home Agent(CHA)로 구성되어 있다. 메모리 장치로는 8 개의 MCDRAM 컨트럴러와 각각 3 개의 DDR 채널을 담당하는 2 개의 외부 메모리 컨트롤러가 칩 외부에 분산되어 있다.



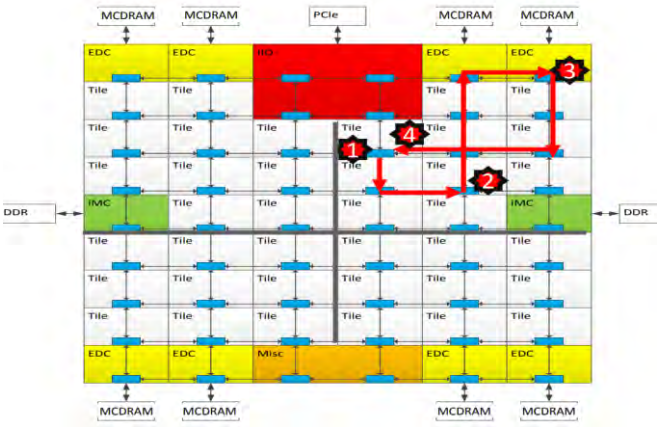
(그림 1) KNL Architecture

KNL 에서 MCDRAM 을 활용하는 방법은 last-level cache 로 사용하는 방법(Cache) 추가 메모리로 사용하는 방법 (Flat), 두 방식을 함께 사용하는 방법(Hybrid)의 3 가지가 있으며 부팅 시점에 설정할 수 있다. Cache 모드로 사용하는 경우 DDR 메모리의 directed mapped cache 방식으로 동작하기 때문에 작업의 메모리 접근 패턴에 따라 Cache thrashing 이 발생할 수 있고, 부팅 후 시간이 지날수록 성능이 저하되는 경향이 있다. 이에 NERSC 에서는 zonesort 라는 커널 모듈을 이용하여 매 작업마다 free page cache list 를 정렬하는 방식으로 성능 저하를 완화하였다.[5] Flat 모드로 MCDRAM 을 사용하는 경우 메모리만 존재하는 NUMA 노드가 추가되는 형태로 메모리 총량이 늘어난다. 응용에 추가적인 코드를 삽입하거나 실행 시 numactl 설정을 추가하는 방법으로 MCDRAM 을 우선 사용할 수 있다.

각 타일의 L2 캐시는 MESIF 프로토콜과 CHA 를 통해 캐시 일관성을 보장한다. L2 캐시에서 miss 가 발생하면 해당주소에의 해시값으로 결정되는 CHA 를 방문하여 캐시 데이터가 존재하는 타일의 위치를 알아내거나 메모리에서 직접 데이터를 읽어온다. CHA 의 위치와 메모리 컨트롤러의 위치에 따라 mesh 네트워크 상의 경로가 길어질 수 있다. 이를 해결하기 위해 CHA 와 메모리 사이의 affinity 를 설정할 수 있는 5 가지 클러스터 모드를 제공한다. All-to-all 은 affinity 없이 메모리 주소가 CHA 들과 물리메모리에 균일하게 분산되어 있다. Hemisphere 와 Quadrant 모드는 칩을 각각 2 개, 4 개의 구역으로 나누고 같은 구역의 CHA 와 물리메모리가 affinity 를 가지게 설정한다. SNC-2 와 SNC-4 모드는 2 개 혹은 4 개로 구역으로 나누고 내부의 코어, L2, CHA, 물리 메모리끼리 affinity 를 가지게 된다. 이 경우 OS 의 관점에서는 마치 2 개 혹은 4 개의 NUMA 노드가 존재하는 것으로 보이게 된다. 클러스터 모드는 마찬가지로 부팅시점에 설정된다. 아래 그림 2, 3 은 각각 Quadrant 와 SNC-4 모드에서 L2 캐시미스 발생시의 동작 과정을 나타낸다.



(그림 2) Quadrant 모드에서 메모리 접근 경로



(그림 3) SNC-4 모드에서 메모리 접근 경로

MCDRAM 모드와 클러스터 모드에 따라 메모리 성능 차이가 발생할 수 있다. 하드웨어 특성 및 응용의 메모리 접근 패턴을 제대로 이해하고 있으면 보다 나은 설정을 활용할 수 있다. 그러나 대부분의 사용자

는 이러한 정보를 정확히 파악하는 것이 불가능하므로 NERSC 같은 슈퍼컴퓨터 운영 기관에서는 Cache node 와 Quadrant 모드를 기본으로 설정하고 특별한 요청이 있을 때 에만 다른 모드를 제공한다.

3. 성능 분석 방법 및 결과

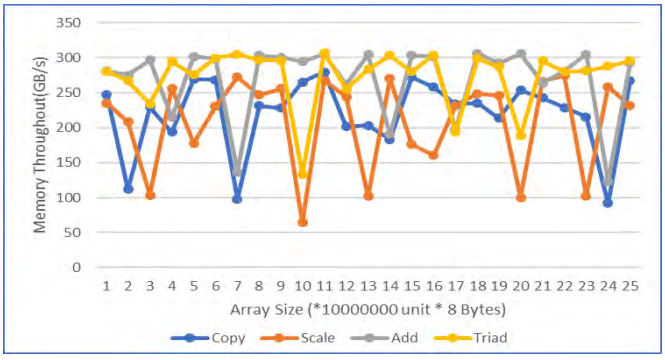
본 연구에서는 KNL 프로세서의 MCDRAM 성능을 측정하기 위해 64 개의 코어와 16GB MCDRAM 을 포함하고 있는 Intel Xeon Phi 7210 Processor 를 사용하고 DDR 메모리는 96GB 를 설치하였다. 코어의 속도는 1.3GHz 이며 MCDRAM 의 제조사 제공 최대 대역폭은 450GB/s 가량이다.

본격적인 메모리 성능 평가에 앞서 메모리 주소에 따른 CHA 배치 및 MCDRAM 의 배치 방법을 알아보았다. 좁은 영역의 주소에 쓰기를 수행하고 cache flush 를 반복했을 때 발생하는 memory access 가 어떤 MCDRAM 에 전달되는지를 프로세서 event 인 offcore_response.any_request.mcdram_near 와 offcore_response.any_request.mcdram_far 를 이용해 추적해 보았다.데이터는 cache line(64Byte)별로 다른 CHA 에 할당되며 라운드 로빈 방식이 아니라 해싱을 통해 불규칙적으로 분산되어 한 Page(4KB)의 데이터는 모든 MCDRAM 에 균일하게 분배되고 그 결과 대역폭이 극대화된다.

대용량 메모리 성능 평가에는 STREAM 벤치마크 [8]를 사용하였다. 8byte double type 의 원소를 가지는 3 개의 대용량 배열(A,B,C)을 생성하고 배열간 Copy (C[i]=A[i]), Scale (B[i]=k*C[i]), Add (C[i]=A[i]+B[i]) , Triad(B[i]=A[i]+s*C[i])의 연산을 OpenMP 를 이용하여 병렬로 수행한다. 캐시메모리보다 훨씬 큰 영역을 순차 접근하므로 메모리 자체의 성능을 측정할 수 있다. 또한 병렬 프로그래밍에서 이러한 메모리 접근 패턴과 유사한 상황은 일반적으로 발생한다.

먼저 코어와 MCDRAM 사이의 거리가 성능에 미치는 영향을 알아보기 위해 SNC-4 모드에서 numactl 을 이용하여 계산 코어와 MCDRAM 노드를 지정하여 성능을 측정하였다. 그 결과 동일 노드의 코어와 메모리 사이의 성능과 다른 노드의 코어와 메모리 성능의 차이가 오차범위 이내(1%미만)에서 동일하였다. 이는 온칩 네트워크상의 거리가 메모리 대역폭에 미치는 영향이 미미함을 의미한다.

Quadrant-Cache 모드에서 쓰레드의 개수는 64 개로 고정시키고 배열의 크기를 10M(1 천만개*8byte double) 부터 250M 개까지 10M 단위로 증가시키면서 성능을 측정하였다. 캐시모드의 모든 실험에서는 매 실행전에 zonesort 를 수행하였다. 그림 4 에 나온 실험결과에 따르면 배열의 크기에 따라 메모리 성능이 크게는 3 배 이상의 큰 차이를 보이는 것을 확인할 수 있었다.



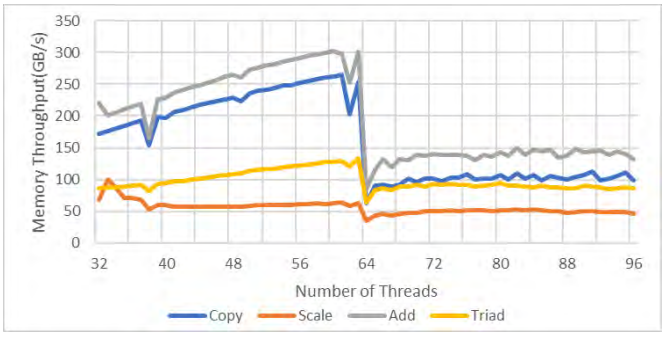
(그림 4) Quadrant 모드에서 배열크기에 따른 메모리 성능

다음으로는 각 배열의 크기를 GB 의 배수로 설정했을 때의 성능을 측정해 보았다. 아래 표 1 에 나타난 것처럼 배열의 크기가 페이지 크기와 정렬되 있으며 일정한 크기의 배수일 경우 성능은 거의 동일하게 측정되었다. 그러나 그 성능이 연산 종류에 따라 하드웨어 성능에 크게 미치지 못하는 것이 확인되었다.

(표 1) Quad/Cache 모드에서 배열의 크기에 따른 성능

ArraySize	Copy	Scale	Add	Triad
1 GB	261 GB/s	61 GB/s	296 GB/s	128 GB/s
2 GB	261 GB/s	61 GB/s	299 GB/s	131 GB/s
3 GB	258 GB/s	61 GB/s	297 GB/s	128 GB/s
4 GB	256 GB/s	61 GB/s	300 GB/s	129 GB/s
5 GB	224 GB/s	66 GB/s	238 GB/s	130 GB/s

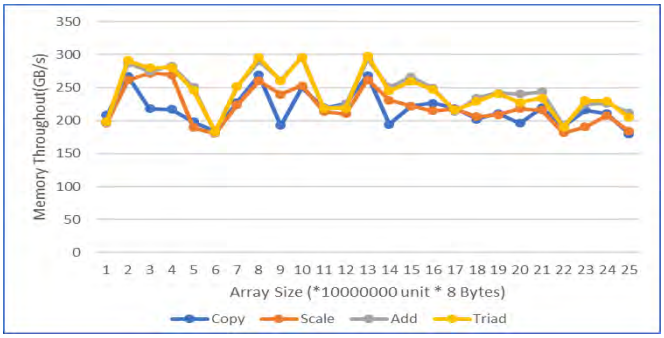
이러한 성능 차이에 쓰레드의 개수가 미치는 영향이 있는지 확인하기 위한 실험을 수행하였다. 그림 5 에는 1GB 의 경우 쓰레드 개수에 따른 성능변화를 보여준다. 64 개까지 쓰레드 개수에 비례하여 상승하다가 64 개를 넘어가면 성능이 떨어지는 것을 확인하였다. 이러한 경향은 모든 배열 크기에서 동일하게 측정되었다. 배열의 크기에 따라 발생하는 성능 저하는 쓰레드 개수와 특별한 상관관계 없이 일관적으로 발생하는 것을 확인하였다.



(그림 5) 쓰레드 개수에 따른 메모리 성능

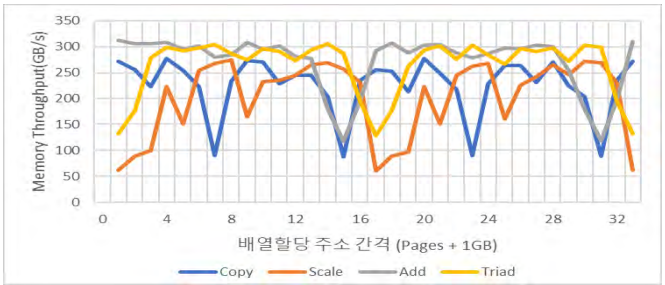
동일한 실험을 Quadrant/Flat 모드에서 진행하였을 때도 마찬가지로 배열 크기에 따라 성능이 크게 바뀌었고 1GB 의 배수일 때 Scale 과 Triad 의 경우 성능이 하락하였다. 즉 성능 저하는 Cache 모드 때문이 아니라 Quadrant 의 CHA 배치 문제일 가능성 크다.

다음으로는 SNC-4 모드에서도 배열 크기에 따라 성능이 변하는지를 확인해 보았다. 그림 6 의 결과와 같이 Quadrant 와 같이 극적인 차이가 나타나지는 않았지만 배열의 크기에 따라 메모리 성능이 크게는 30%이상 차이가 나는 것을 확인할 수 있었다.

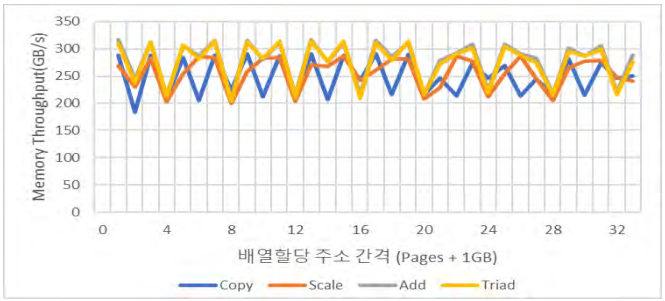


(그림 6) SNC-4 모드에서 배열 크기에 따른 메모리 성능

STREAM 벤치마크의 경우 각 배열의 동일한 인덱스 간의 연산을 반복적으로 수행하고 이는 접근하는 메모리 주소의 간격이 일정하다는 것을 의미한다. 이에 착안하여 각 배열의 할당된 주소의 차이를 1GB 로부터 증가시켜가며 성능 차이를 측정해 보았다. 대용량 배열의 할당은 페이지 단위로 이루어지므로 간격 단위는 페이지로 두었다. 그림 7, 8 를 통해 Quadrant, SNC-4 모드 각각에서 배열간격을 변경시켰을 때의 성능 변화가 주기적으로 반복됨을 알 수 있다. 이러한 현상은 1GB 로부터 시작하지 않는 다른 크기 차이에서도 동일하게 관측되었다. 특히 인접하여 할당되는 두 배열 간의 연산인 Scale 의 경우 Quadrant 모드 경우에는 16 페이지마다 SNC-4 모드의 경우에는 4 페이지마다 성능 변화가 반복된다. 이 차이가 4 배로 모드에서 담당하는 CHA 혹은 MCDRAM 채널의 크기 비율과 동일한 것도 병목과 연관이 있을 것으로 추측된다.



(그림 7) Quadrant 모드에서 배열 간격에 따른 메모리 성능



(그림 8)SNC-4 모드에서 배열 간격에 따른 메모리 성능

4. 결론 - HPC 프로세서의 메모리 성능 향상 방안

KNL 프로세서를 활용한 실험을 통해 특정한 차이를 가지는 주소를 동시에 접근하는 상황이 계속 반복되는 경우 성능이 크게 저하되는 것을 확인하였다. 이는 동일한 CHA 혹은 MCDRAM 채널에 여러 코어로부터의 접근이 몰려서 발생하는 병목현상일 가능성이 있다. 또한 이러한 접근 패턴은 실제 병렬 프로그램에서 충분히 발생 가능하다. 또한 이러한 성능저하를 유발하는 주소차이가 특정한 크기를 가지고 반복된다는 점에서 하드웨어의 메모리 분산 기법과 영향이 있다고 유추할 수 있다.

Scale-out 프로세서에서 데이터 처리의 분배로 인해 발생하는 성능 저하는 완벽하게 제거할 수 없다. 하드웨어 설계 측면에서는 가능한 불규칙적으로 분배함으로써 특별한 상황에서 급격하게 성능이 저하되는 상황을 막아야 하며, 그게 불가능한 상황이면 이러한 성능 저하 가능성에 대한 정보를 최대한 시스템 소프트웨어 개발자에게 제공해야 한다. 이를 통해 OS 의 page 할당 단계 혹은 컴파일러 혹은 병렬 프로그램의 런타임이 가상 메모리에 배열을 배치하는 시점에서 이러한 병목 현상을 충분히 회피할 수 있을 것이다. 향후 국내에서도 HPC 프로세서 및 그를 활용한시스템이 개발될 때 이러한 사항이 고려되기를 기대한다.

본 논문은 대한민국 정부의 재원으로 한국과학기술정보연구원 주요사업의 지원을 받아 수행된 연구임 (과제번호: K-20-L02-C08-S01)

참고문헌

[1] A. Sodani et al., "Knights Landing: Second-Generation Intel Xeon Phi Product," in IEEE Micro, 2016

[2] A64FX Processor in FUGAKU Supercomputer, Fujitsu, https://www.fujitsu.com/global/products/computing/servers/supercomputer

[3] Cori, NERSC, https://docs.nersc.gov/systems/cori/

[4] NURION, KISTI, https://www.ksc.re.kr/ggspcpt/nurion

[5] KNL Cache Mode Performance, https://www.nersc.gov/research-and-development/knl-cache-mode-performance-coe/

[6] S. Rho et al., “Performance Anamysis of Various Multi- and Many-Core Systems Centered on Memory,” 2019 IEEE 4th international Workshops on Foundations and Applications of Self Systems, 2019

[7] McCalpin. John D., "Memory Bandwidth and Machine Balance in Current High Performance Computers", IEEE Computer Society Technical Committee on Computer Architecture Newsletter, 1995